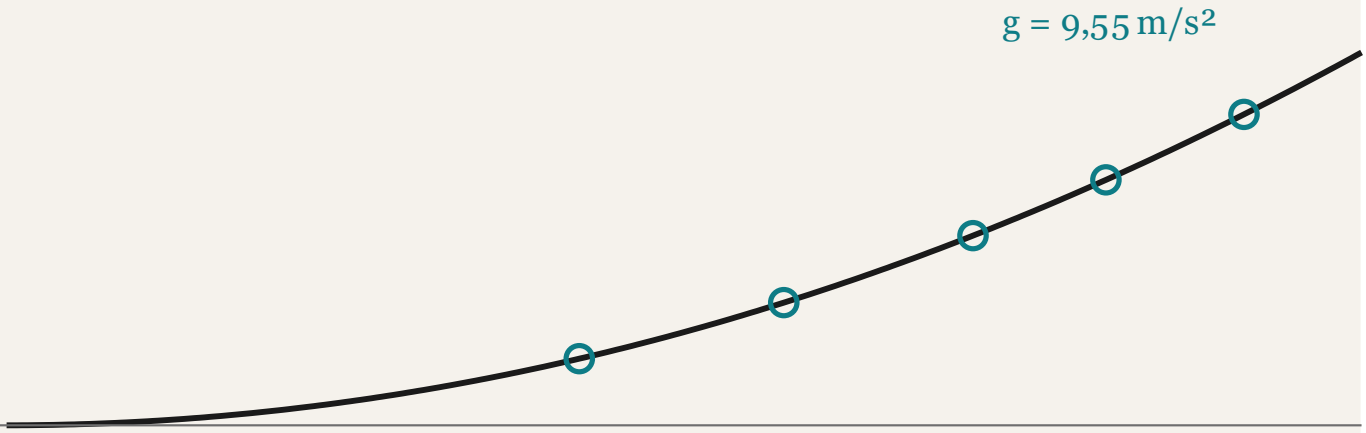
FÍSICA · INSTRUMENTACIÓN · AULA



Experimentación en física

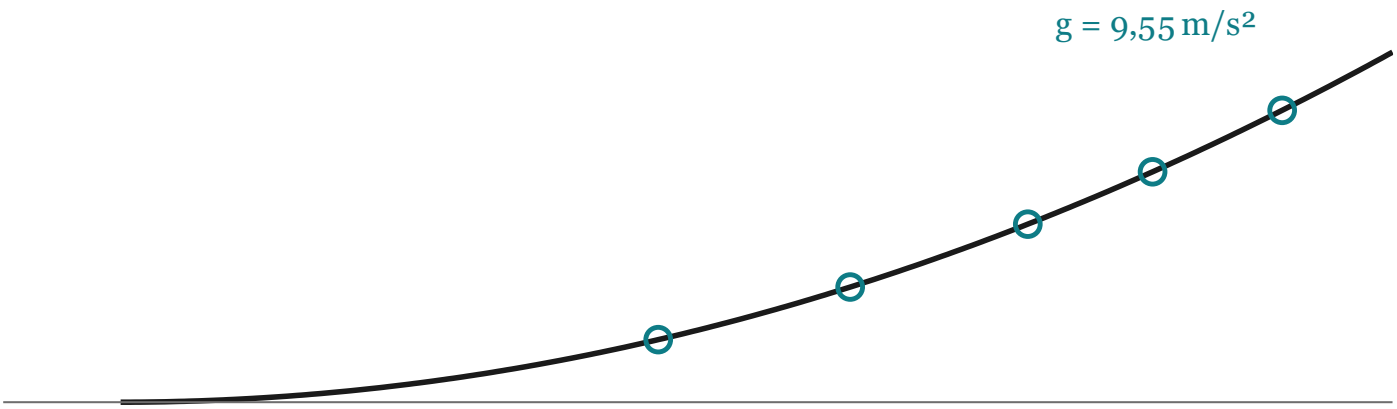
con Arduino y ESP32

Más de 30 prácticas de laboratorio con sensores de bajo coste

J. Lorenzo Ramírez Castro

experimentacioliure.com · Licencia CC BY-SA 4.0

FÍSICA · INSTRUMENTACIÓN · AULA



Experimentación en física

con Arduino y ESP32

Más de 30 prácticas de laboratorio con sensores de bajo coste

J. Lorenzo Ramírez Castro

experimentacioliure.com · Licencia CC BY-SA 4.0

J. Lorenzo Ramírez Castro, 2026
http://experimentaciolliure.com/
http://experimentaciolliure.wordpress.com/



https://creativecommons.org/licenses/by-sa/4.0/
Este documento está bajo una licencia Creative Commons Reconocimiento-Compartir-Igual 4.0 Internacional (CC BY-SA 4.0)
Los dibujos que ilustran los títulos de los diferentes capítulos y dos más en el movimiento circular están creados con inteligencia artificial (se indica en el pie de cada uno de ellos).
Las imágenes de las placas microprocesadoras y de los sensores en el capítulo de la maquinaria están sacadas de las publicaciones de las respectivas casas comerciales.
El resto de los dibujos y fotografías son del autor.

Índice de capítulos

Introducción 9

Maquinaria 13

Microcontroladores y microprocesadores: Arduino y ESP32 15

Los sensores 23

IDE de Arduino: instalación, configuración y uso 51

Tarjetas microSD: almacenamiento de datos en experimentos autónomos 59

Arduino Nano 33 BLE Sense: conexión con Phyphox y Arduino Science Journal 63

Mecánica. Movimientos 67

Seguimiento del movimiento de personas 69

Movimiento de carritos con sensor de distancia 73

Movimiento de carritos con barreras ópticas 77

Medida de la aceleración de la gravedad 80

Cálculo de g con un péndulo 86

Caída y rebote de una pelota 89

Estudio del movimiento circular: velocidad angular constante y aceleración angular 91

Oscilaciones y vibraciones 99

Oscilaciones de un muelle: medida de fuerzas con célula de carga 101

Oscilaciones de un muelle: determinación de la constante elástica mediante sensor de distancia 109

Frecuencia de vibración de una varilla empotrada 113

Oscilaciones amortiguadas de una regla con sensor LDR 118

Mecánica de fluidos 121

Presión hidrostática y medida del nivel de agua 123

Variación de la presión atmosférica con la altura 127

Presión en el interior de un globo 130

Medida de la presión en el interior de una burbuja 135

Medida de masa mediante presión sobre un cojín de aire 138

Calor y Temperatura 141

Medida de la temperatura: curva de enfriamiento con sensor DS18B20 143

Calor latente de fusión del hielo 146

Disoluciones exotérmicas y endotérmicas 149

Absorción de la radiación solar o de una bombilla por superficies de diferente color 152

Combinación de las leyes de los gases: Boyle, Charles y Gay-Lussac 155

Estación meteorológica 162

Luz y Color 165

Irradiancia y ley de Lambert 167
Ley de Beer-Lambert 172
Polarización de la luz 174
Análisis de la frecuencia de la corriente alterna mediante un fototransistor y Phyphox 176
Espectroscopia con el sensor de color TCS34725 180
Simulación de tránsito planetario 184
Medida de las variables ambientales durante un eclipse solar 186
Electromagnetismo 195
Carga y descarga de un condensador 197
Conductividad eléctrica del grafito 201
Variación del campo magnético de un imán con la distancia 204
Medida del campo magnético terrestre 207
Inducción magnética en una espira 214
Permitividad eléctrica de materiales 217
Conductividad eléctrica del agua 219
Medida del pH de disoluciones acuosas 226
Anexos 231
Glosario de términos 233
Índice Alfabético 249
Referencias bibliográficas 254

Índice general

Introducción 9
Maquinaria 13
Microcontroladores y microprocesadores: Arduino y ESP32 15
La placa Arduino UNO 15
Las placas con ESP32: Wemos D1 R32 16
Otras placas con ESP32 18
Arduino Nano 33 BLE Sense 18
Comparativa de placas 20
¿Qué placa elegir para cada experimento? 20
Alimentación eléctrica de los microcontroladores 21
Los sensores 23
1. Sensores de distancia 23
1.1 Sensor de ultrasonidos HC-SR04 23
1.2 Sensor VL53L0X (láser infrarrojo) 24
1.3 Sensor de distancia por infrarrojos SHARP GP2Y0A21 25
2. Sensores de temperatura 25
2.1 Sensor DS18B20 25
2.2 Sensor LM35 26
2.3 Termistores NTC 27
2.4 Sensor DHT11/DHT22 28
2.5 Sensores AHT2X (AHT20/AHT21) 28
3. Sensores de presión (y temperatura y humedad) 30
3.1 BMP280 y BME280 30
4. Sensores de luz y color 31
4.1 BH1750 31
4.2 TSL2591 32
4.3 Fotoresistencia (LDR) 33
4.4 Fototransistor 33
4.5 Sensor de color TCS34725 34
5 Sensores de proximidad por infrarrojos 37
5.1 Módulo Flying-fish 37
5.2 Módulo KY-032 37
6. Barreras de luz 39
6.1 Fotopuertas 39

6.2 Optoacoplador IR de ranura 40
7. Sensor de pH 41
8. Sensor de fuerza/peso (célula de carga + HX711) 42
9. Sensores magnéticos 43
9.1 Sensor de efecto Hall A3144 (KY-003) 43
9.2 Magnetómetro HMC5883L / QMC5883L / QMC5883P 44
9.3 Sensor de flujo magnético SS49E 44
10. Conductividad y nivel de agua 45
10.1 Sensor de conductividad 45
10.2 Sensor de nivel de agua 47
11. Otros sensores para experimentos de física 47
11.1 Sensor de corriente ACS712 47
11.2 Sensor de sonido MAX9814 / KY-038 48
11.3 Sensor de flujo de agua YF-S201 48
12. Consideraciones generales sobre conexión de sensores 49
IDE de Arduino: instalación, configuración y uso 51
Instalación del software 51
Reconocimiento de las placas en el IDE 51
Arduino UNO 51
Placas con ESP32 (Wemos D1 R32, DOIT ESP32 DEVKIT V1, etc.) 51
Arduino Nano 33 BLE Sense 52
Gestión de librerías 52
Instalar librerías desde el Gestor de bibliotecas (método recomendado) 53
Instalar librerías desde un archivo ZIP 53
Estructura de un sketch (proyecto) 53
Visualización de datos 55
Monitor serie del IDE de Arduino 55
Serial Plotter (visor gráfico integrado) 55
RealTerm 55
Serial USB Terminal (dispositivos Android) 56
Serial Bluetooth Terminal (dispositivos Android) 56
HM10 Bluetooth Serial Lite (Dispositivos iOS) 56
Phyphox 56
Formato de los datos 57
Solución de problemas comunes 57
Alternativas al IDE de Arduino 58
Tarjetas microSD: almacenamiento de datos en experimentos autónomos 59
El módulo lector de tarjetas microSD 59

Librerías necesarias 59
Ejemplo completo: registro de datos de un sensor BME280 en tarjeta microSD 60
Arduino Nano 33 BLE Sense: conexión con Phyphox y Arduino Science Journal 63
Instalación de la placa en el IDE de Arduino 63
Sensores integrados y microcontrolador 63
Librerías necesarias 64
Conexión con Phyphox 64
Conexión con Science Journal 65
Mecánica. Movimientos 67
Seguimiento del movimiento de personas 69
Movimiento de carritos con sensor de distancia 73
Movimiento de carritos con barreras ópticas 77
Medida de la aceleración de la gravedad 80
Cálculo de g con un péndulo 86
Caída y rebote de una pelota 89
Estudio del movimiento circular: velocidad angular constante y aceleración angular 91
Oscilaciones y vibraciones 99
Oscilaciones de un muelle: medida de fuerzas con célula de carga 101
Oscilaciones de un muelle: determinación de la constante elástica mediante sensor de distancia 109
Frecuencia de vibración de una varilla empotrada 113
Oscilaciones amortiguadas de una regla con sensor LDR 118
Mecánica de fluidos 121
Presión hidrostática y medida del nivel de agua 123
Variación de la presión atmosférica con la altura 127
Presión en el interior de un globo 130
Medida de la presión en el interior de una burbuja 135
Medida de masa mediante presión sobre un cojín de aire 138
Calor y Temperatura 141
Medida de la temperatura: curva de enfriamiento con sensor DS18B20 143
Calor latente de fusión del hielo 146
Disoluciones exotérmicas y endotérmicas 149
Absorción de la radiación solar o de una bombilla por superficies de diferente color 152
Combinación de las leyes de los gases: Boyle, Charles y Gay-Lussac 155
Estación meteorológica 162
Luz y Color 165
Irradiancia y ley de Lambert 167
Variación de la iluminancia con la inclinación 170
Ley de Beer-Lambert 172

Polarización de la luz 174
Análisis de la frecuencia de la corriente alterna mediante un fototransistor y Phyphox 176
Espectroscopia con el sensor de color TCS34725 180
Simulación de tránsito planetario 184
Medida de las variables ambientales durante un eclipse solar 186
Electromagnetismo 195
Carga y descarga de un condensador 197
Conductividad eléctrica del grafito 201
Variación del campo magnético de un imán con la distancia 204
Medida del campo magnético terrestre 207
Inducción magnética en una espira 214
Permitividad eléctrica de materiales 217
Conductividad eléctrica del agua 219
Medida del pH de disoluciones acuosas 226
Anexos 231
Glosario de términos 233
Índice Alfabético 249
Referencias bibliográficas 254
Datasheets y documentación técnica 263
Librerías 266

Introducción

En mi interés por conseguir instrumentación asequible y versátil para la toma de datos en los laboratorios escolares de Educación Secundaria y Bachillerato, he estado ocupado años atrás explorando las enormes posibilidades de los dispositivos móviles, como son los teléfonos y tabletas inteligentes. De hecho, en la primavera de 2022 concluí una recopilación de experimentos y publiqué en la red la segunda versión del libro **Experimentación en Física con dispositivos móviles 2** (Ramírez, 2022).

Desde entonces he estado dando vueltas a cómo completar aquellas propuestas mediante la utilización de «arduinos», abriendo el abanico de sensores de los que podemos disponer a muy bajo coste, como los de temperatura sumergibles, distancia por láser, barómetros de alta precisión o sensores de conductividad eléctrica y pH.

Hablo de «arduinos» de la misma forma que habitualmente decimos «danones» para referirnos a cualquier yogur, empleando el término de forma genérica para englobar a las distintas familias de microcontroladores y placas de desarrollo que propongo utilizar en esta obra (Arduino UNO, ESP32 y Arduino Nano 33 BLE Sense).

Logo de la empresa creadora de los microcontroladores Arduino

Si en vuestros centros docentes disponéis de sistemas comerciales de adquisición de datos (*dataloggers*) como los de Fourier Systems (Einstein) o firmas similares, quizás no sea imprescindible complicarse la vida ensamblando componentes. Sin embargo, si no contáis con ellos, o si queréis superar las restricciones de los entornos cerrados, espero que este libro os anime a embarcaros en las propuestas que presento.

El sistema formado por los microcontroladores y su ecosistema de sensores acoplables ofrece unas prestaciones de grado científico a una fracción testimonial de su coste comercial.

Es importante remarcar que este libro no busca competir con el uso de los teléfonos móviles como laboratorios portátiles, sino complementarlos. Si bien los *smartphones* son herramientas extraordinarias gracias a sus sensores internos (acelerómetros, giroscopios, magnetómetros y fotómetros), presentan limitaciones evidentes cuando queremos introducir un sensor en el interior de una disolución química, medir la presión en un globo, registrar el enfriamiento de un líquido a alta temperatura o medir campos magnéticos y corrientes con sondas cableadas. Es justamente en este terreno donde los microcontroladores Arduino y ESP32 demuestran su valor insustituible.

Logo de la empresa creadora de los microprocesadores ESP32

Debo empezar confesando que en el ámbito de los microprocesadores y la programación me considero un principiante entusiasta. Las soluciones que ofrezco en estas páginas son las que he conseguido hacer funcionar tras muchas horas de ensayo y laboratorio, aunque no sean siempre las más elegantes ni las más eficientes desde el punto de vista de la ingeniería de software.

Mis propuestas nacen de adaptar y combinar el trabajo de muchos otros docentes y divulgadores que, afortunadamente, comparten abiertamente su conocimiento.

En este sentido, debo expresar mi profundo agradecimiento al fantástico profesor de Tecnología y gran amigo Pere Picoy, quien me introdujo en este

fascinante mundo a través de un curso de Arduino impartido desinteresadamente en el marco del seminario CTM (Ciencias, Tecnología y Matemáticas) que organizamos en Lleida en 2015.

Asimismo, quiero reconocer las aportaciones de otros autores de referencia en cuyos trabajos me he apoyado constantemente:

Luis Llamas (https://www.luisllamas.es/), ingeniero industrial de Zaragoza que mantiene desde hace años un blog extraordinario con cursos de electrónica, Arduino y ESP32, ofrecidos generosamente bajo licencia Creative Commons (CC BY-NC-SA).

Giovanni Organtini, profesor de la Sapienza Università di Roma y pionero en la introducción rigorosa de Arduino en el laboratorio de física universitaria y secundaria (Organtini, 2016, 2020).

Beatriz Padón Romero, profesora de enseñanza secundaria, autora de un curso sintético pero impecable de física con Arduino en su plataforma web (Padón, 2019).

Un capítulo aparte merece la irrupción de la **Inteligencia Artificial generativa (IA)**. Durante el desarrollo de este libro, el uso de asistentes de IA ha pasado de ser una curiosidad a convertirse en un copiloto técnico fundamental, de manera que la IA me ha permitido en este último año resolver atascos en el código de los microcontroladores

Esta tecnología democratiza el acceso a la programación (aunque también presenta otras derivadas que son motivo de fuerte preocupación) y permite que cualquier docente de física o química, sin formación previa en ciencias de la computación, pueda diseñar experiencias avanzadas de adquisición de datos centrándose exclusivamente en lo importante, que no es sino la enseñanza y la comprensión del fenómeno físico.

Estructura del libro

Esta obra está pensada como un manual práctico. Tras este texto introductorio, el libro se divide en dos grandes bloques:

Bloque I: La Maquinaria (Capítulos 1 a 5)
Recoge los fundamentos técnicos imprescindibles para llevar a cabo las experiencias. Está diseñado para consultarse de forma independiente como catálogo de referencia, y contiene:

Microcontroladores y microprocesadores: Arduino y ESP32. Análisis descriptivo y comparativo de las tres placas de desarrollo empleadas en la obra (Arduino UNO, Wemos D1 R32 con ESP32, y Arduino Nano 33 BLE Sense), sus especificaciones de memoria, arquitectura de pines y criterios de elección para cada experimento.

Los sensores. Presentación detallada del catálogo de sensores de bajo coste empleados (distancia ultrasonidos/láser, temperatura, presión, luz/color, barreras ópticas, células de carga, campo magnético, pH, conductividad y flujo). Se detallan sus principios físicos de funcionamiento, rangos de medida y esquemas de conexión.

IDE de Arduino: instalación, configuración y uso. Guía paso a paso para la instalación del entorno de desarrollo, gestión de tarjetas y librerías, anatomía de un programa (*sketch*) y uso del Monitor Serie y el Plotter Serie para la inspección de datos.

Tarjetas microSD: almacenamiento de datos en experimentos autónomos. Explicaciones para conectar módulos de lectura/escritura, almacenamiento en archivos CSV y registro continuado de datos en experiencias de campo.

Arduino Nano 33 BLE Sense: conexión con Phyphox y Arduino Science Journal. Aprovechamiento de la conectividad Bluetooth Low Energy (BLE) y la integración directa con aplicaciones móviles para la toma inalámbrica de datos en tiempo real de una tarjeta que integra diversos sensores.

Bloque II: Propuestas Experimentales
Constituye el núcleo práctico del libro, con más de 30 experimentos distribuidos en seis bloques temáticos:

Mecánica y Movimientos. Estudio de la cinemática de personas y carritos con sensores de distancia por ultrasonidos y láser, barreras ópticas, determinación precisa de g mediante péndulos y caídas libres, rebotes de pelotas y dinámica del movimiento circular.

Oscilaciones y Vibraciones. Análisis del movimiento armónico simple en muelles utilizando

células de carga (fuerza) y sensores de distancia (posición), oscilaciones amortiguadas con fotoresistencias LDR y frecuencia de vibración en varillas empotradas.

Mecánica de Fluidos. Hidrostática y nivel de agua, variación barométrica de la presión atmosférica con la altitud, presión en la pared de globos y pompas de jabón, y pesaje de masas mediante cojín de aire.

Calor y Temperatura. Curvas de enfriamiento con sondas digitales DS18B20, determinación del calor latente de fusión del hielo, termoquímica en disoluciones endo/exotérmicas, irradiancia y absorción diferencial de la radiación solar por colores, verificación experimental de las leyes de los gases (Boyle, Charles y Gay-Lussac) y construcción de una estación meteorológica.

Luz y Color. Verificación de la Ley de Lambert y la Ley de Beer-Lambert, experimentos de polarización, espectroscopia cualitativa, medida de la frecuencia de la red eléctrica mediante fototransistores y simulación de tránsitos planetarios y eclipses.

Electromagnetismo. Carga y descarga de condensadores, conductividad eléctrica del grafito y del agua, perfil espacial de campos magnéticos, medida del campo magnético terrestre, inducción electromagnética en espiras (Ley de Faraday), permitividad eléctrica de dieléctricos.

Química. Conductividad eléctrica del agua y medida del pH en disoluciones acuosas.

Nota sobre las experiencias de Acústica y Sonido: Observaréis que este libro no incluye un capítulo específico dedicado al sonido. Esta decisión se debe a que el análisis acústico (frecuencias, espectrogramas, velocidad del sonido, efecto Doppler) se realiza de forma precisa y sencilla aprovechando el micrófono integrado en los teléfonos inteligentes mediante las aplicaciones Phyphox o Physics Toolbox.

Consejos para el uso del libro

Para obtener el máximo partido de esta obra en nuestra práctica docente, me atrevo a dar algunas recomendaciones surgidas de la experiencia en el aula:

No es necesario leer el libro de principio a fin. Os sugiero hojear directamente el bloque de los experimentos, seleccionar la práctica que se ajuste a vuestro currículo actual y acudir después a los capítulos de La Maquinaria para resolver los esquemas de conexión y la configuración de las placas.

Para evitar reiteraciones que podrían hacer la lectura pesada, los experimentos no repiten en cada capítulo todos los esquemas y figuras de patillaje de cada sensor. En su lugar, hay que ir a las figuras y tablas del capítulo de los sensores.

Este no es un manual de programación pura ni de arquitectura de software. El código suministrado en cada experimento está pensado para funcionar de inmediato («cargar y medir»). Se puede animar al alumnado a modificar parámetros sencillos (tiempos de muestreo, umbrales), perder el miedo al entorno de desarrollo y utilizar asistentes de IA si se desea adaptar los programas a nuevas variantes experimentales.

Hay que tener paciencia ya que no solo es que preparar actividades experimentales de más trabajo, sino que la informática y la electrónica en el trabajo cotidiano de los laboratorios escolares son impredecibles por cosa nimias, por un cable que hace mal contacto, por un carácter que se ha borrado en el código, etc., así que paciencia y pedir ayuda a los compañeros de Tecnología.

Este tampoco es un texto de didáctica de los trabajos prácticos, ni de cómo incardinarlos en el currículum, que lo dejo a vuestro saber como docentes, pero está escrito por mi fuerte convencimiento de la importancia de la experimentación en la enseñanza de las Ciencias.

Todos los experimentos cuentan con una breve fundamentación teórica que entronca directamente con los contenidos curriculares de Educación Secundaria y Bachillerato. Si necesitáis profundizar en el modelo físico de alguna experiencia, podéis recurrir a plataformas de referencia libre en línea como *HyperPhysics* (Nave y Olmo, 2010), el *Curso Interactivo de Física en Internet* de Ángel Franco (Franco, 2015) o los textos de *Física Universitaria* de OpenStax (Ling et al., 2021).

De lo que no me he preocupado en los experimentos es de la cuantificación de los errores en los

resultados de los valores experimentales, sino de que fueran del orden de magnitud o lo más cercanos posibles a los valores reales. Para el estudio de los errores podéis consultar, por ejemplo, el documento *Errors en les mesures* (Ramírez, 2012).

Espero que las páginas que siguen os sirvan de estímulo, os ahorren horas de ensayo en el laboratorio y, sobre todo, os ayuden a contagiar a vuestros estudiantes el placer de la física experimental.

Maquinaria

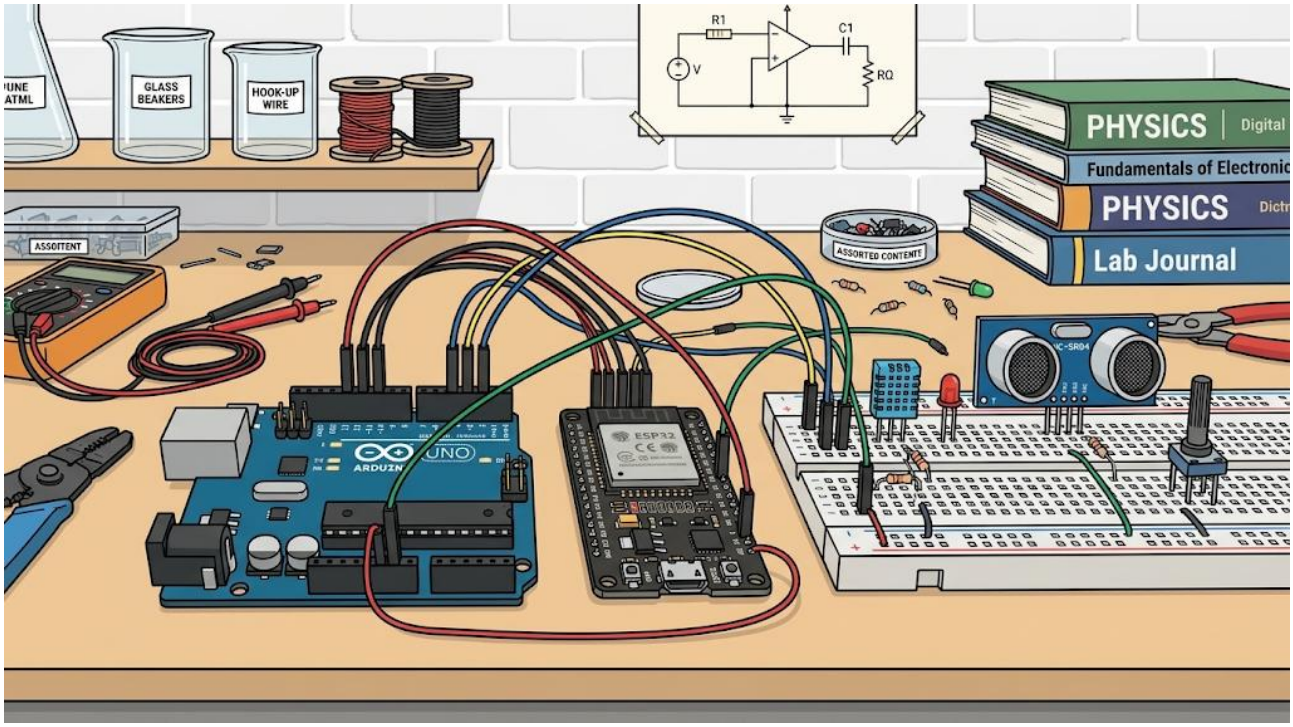


Ilustración realizada con Gemini. Esté pensada para adornar, aunque el alumnado puede entretenerse en buscar los errores evidentes que contiene.

Microcontroladores y microprocesadores: Arduino y ESP32

Normalmente, cuando hablamos de arduino no nos referimos a una única placa, sino a todo un ecosistema de hardware y software de código abierto. El proyecto Arduino ha generado en el seno de la empresa multitud de placas con diferentes microcontroladores diseñadas para hacer más simple y accesible el diseño de circuitos electrónicos.

Pero la idea Arduino ha saltado al mundo mundial y el mercado ha florecido, sobre todo en China, con multitud de clónicos y variantes. En particular, la empresa Expressif ha creado un microprocesador, el ESP32, que montado en infinidad de placas supera en muchos aspectos a los originales arduinos.

En este libro utilizaremos básicamente tres tipos de placa:

- Arduino UNO R3: la placa más extendida, ideal para iniciarse y para experimentos que no requieran conexión inalámbrica.
- Wemos D1 R32 (y otras placas con microprocesador ESP32): incorporan WiFi y Bluetooth, permitiendo enviar datos a un teléfono móvil o tableta sin cables.
- Arduino Nano 33 BLE Sense: una placa de tamaño reducido que integra una gran cantidad de sensores y comunicación Bluetooth BLE.

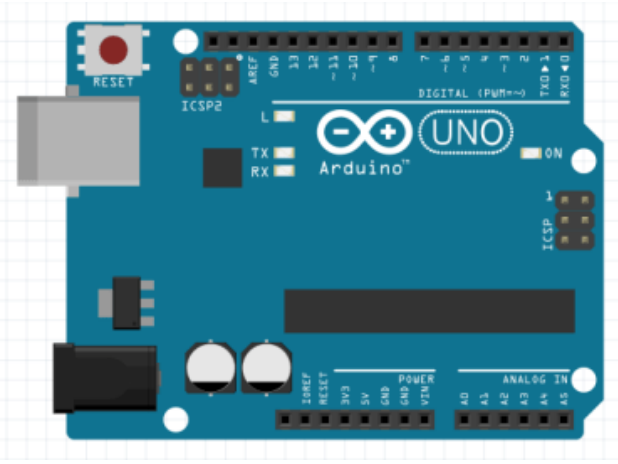
La placa Arduino UNO

Arduino UNO es una placa de circuito impreso basada en un microcontrolador ATmega (ATmega328P en la versión más común). Dispone de todos los elementos necesarios para conectar periféricos a sus entradas y salidas.



Características principales

- Microprocesador: ATmega328P
- Tensión de funcionamiento: 5 V
- Alimentación de la placa: mediante el conector jack (diámetro exterior de 5,5 mm y un diámetro interior de 2,1 mm, rango 7-12 Vcc, por ejemplo, un pac de 6 pilas de 1,5 V) o mediante el puerto USB tipo B (5 Vcc, como un cargador de móvil). También conectando una fuente externa directamente al pin VIN (7 a 12 V)
- Conexión al ordenador: USB tipo B (como los de las impresoras) o C, según el modelo de placa.
- Entradas analógicas: 6 (conversor analógico-digital de 10 bits, valores de 0 a 1023)
- Pines digitales: 14 (6 de ellos con capacidad PWM)
- Frecuencia de reloj: 16 MHz
- Memoria flash: 32 KB
- Precio orientativo: desde 4 € (versiones clónicas)



El Arduino UNO es una de las placas más utilizadas para montajes en Tecnología y es muy probable que los compañeros de esta especialidad puedan facilitaros una y ayudaros con ella.

Ventajas del Arduino UNO

Es un ecosistema enorme en el que existe una cantidad ingente de tutoriales, librerías, shields y ejemplos disponibles. Es la placa con mayor comunidad de soporte.

Es ideal para empezar. Su simplicidad la convierte en la mejor opción para quienes se inician en electrónica con Arduino.

Dispone de entradas analógicas puras, de manera que, a diferencia del ESP32, sus entradas analógicas son auténticas y lineales de 0 a 5 V, lo que resulta esencial para experimentos como el pH-metro o sensores analógicos de precisión.

Existen cientos de shields (módulos apilables) disponibles que amplían sus capacidades: Ethernet, control de motores, pantallas LCD, etc.

Es compatibles con 5 V. La mayoría de sensores y actuadores del mercado funcionan a 5 V, por lo que se conectan directamente sin necesidad de conversores de nivel.

Limitaciones del Arduino UNO
Dispone de poca memoria (solo 32 KB de flash y 2 KB de RAM), suficiente para la mayoría de proyectos de Física, pero insuficiente para programas complejos o grandes volúmenes de datos.

No tiene conectividad inalámbrica nativa, no tiene WiFi ni Bluetooth. Para añadirlos hay que usar módulos externos (como el módulo bluetooth HC-06) que complican el montaje.

El procesador es relativamente lento (16 MHz frente a los 240 MHz del ESP32), lo que limita la frecuencia de muestreo en experimentos rápidos.

La resolución de su ADC (Convertidor Analógico a Digital) es de 10 bits (0-1023), que es suficiente para muchos experimentos, pero inferior a los 12 bits del ESP32.

Casos de uso ideales
Experimentos de laboratorio que requieran entrada analógica pura (pH-metro, sensores de conductividad analógicos).

Primeros contactos con la programación de microcontroladores en entornos educativos.

Proyectos en centros docentes donde ya se disponga de kits de Arduino UNO.

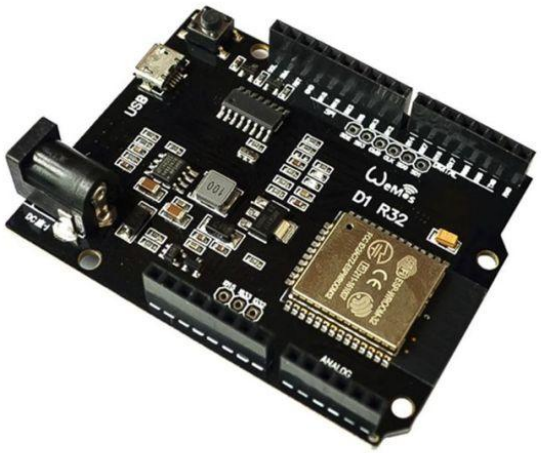
Experimentos donde sea suficiente que los datos se visualicen por el puerto serie en el ordenador.

Conversión analógico-digital en Arduino UNO
Las placas Arduino disponen de un conversor analógico-digital para transformar en ceros y unos el voltaje que le llega por las entradas analógicas. En el caso del Arduino UNO, el conversor es de 10 bits, lo que significa que los 0-5 V se convierten en valores de 0 a 1023 (2^{10} = 1024). Por eso, en los proyectos, para convertir el nivel leído en voltaje se ha de dividir por 1023.

Las placas con ESP32: Wemos D1 R32

El microprocesador ESP32, fabricado por Espressif Systems, es más potente que el ATmega del Arduino UNO e incorpora de serie conexiones WiFi y Bluetooth. Existe una versión de Arduino que incorpora WiFi, pero es mucho más cara que las placas que montan el chip ESP32. También es posible conectar un módulo Bluetooth a un Arduino UNO, pero complica las conexiones. Además, el precio de las placas con ESP32 es similar a las de un Arduino UNO clónico (a partir de 4 € en tiendas online).

La placa Wemos D1 R32

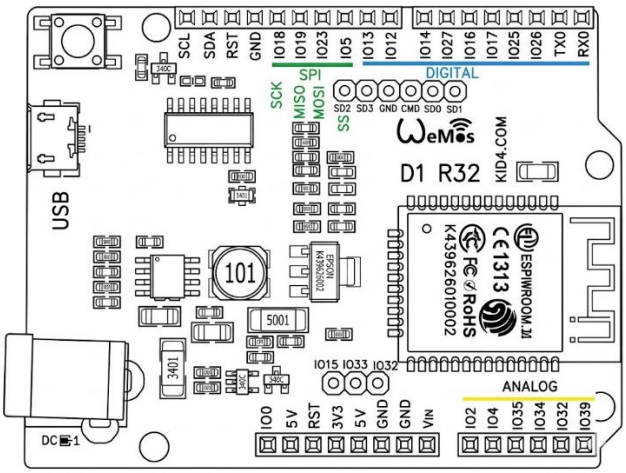


La placa Wemos D1 R32 incorpora un microprocesador ESP32 con WiFi y Bluetooth pero mantiene los tamaños y la compatibilidad con Arduino UNO, por lo que se pueden utilizar los sensores, proyectos (sketches) y shields diseñados para Arduino.

Características principales
- Microprocesador: ESP32-WROOM-32D
- Alimentación de la placa: mediante el conector jack (diámetro exterior de 5,5 mm y un diámetro interior de 2,1 mm, rango 5-12 Vcc, por ejemplo,

un pack de 6 pilas de 1,5 V) o mediante el puerto USB (microUSB o C, según el que disponga, a 5 Vcc, como un cargador de móvil).
También conectando una fuente externa directamente al pin VIN (5 a 12 V)

- Conexión al ordenador: USB micro o C, según el modelo de placa
- Tensión de funcionamiento: 3,3 V (los pines de entrada/salida)
- Tensión de alimentación a sensores: 3,3 y 5 V (dispone de dos pines que suministran 5 V, algo fundamental para sensores como el HC-SR04 que necesitan 5 V)
- Entradas analógicas: varias, con conversor analógico-digital de 12 bits (valores de 0 a 4095)
- Conectividad: WiFi y Bluetooth (BLE)
- Precio orientativo: desde 4 €



Ventajas del ESP32
Dispone de WiFi y Bluetooth integrados, lo que permite enviar datos a un teléfono móvil (Phyphox), a una página web o a la nube sin hardware adicional.

Tiene mayor potencia que el Arduino. Dispone de un procesador dual-core a 240 MHz, mucho más rápido que el ATmega328P.

La resolución de su ADC (Convertidor Analógico a Digital) es de 12 bits (2^{12} = 4096), cuatro veces más fina (0-4095) que la del Arduino UNO (0-1023).

El microprocesador es Dual-core, lo que permite ejecutar dos tareas simultáneamente (por ejemplo, leer sensores y enviar datos por bluetooth al mismo tiempo).

A diferencia de otras placas ESP32, esta tiene pines que suministran 5 V, compatibles con sensores que requieren esta tensión.

Su precio es imbatible. Por menos de 5 € se tiene una placa con WiFi, Bluetooth y más potencia que un Arduino UNO.

Limitaciones del ESP32
Su ADC no es lineal, de manera que las entradas analógicas del ESP32 no son tan lineales ni precisas como las del Arduino UNO. Además, como todos los ESP32, el rango de voltaje máximo en el que trabaja es de 0-3,2 V (no 5 V). Esto hace que no sea adecuado para experimentos que requieran medidas analógicas de precisión (como el pH-metro).

Su uso puede supones mayor complejidad. Por ejemplo, la configuración del entorno de desarrollo (instalación del soporte ESP32 en el IDE de Arduino) es un paso adicional que puede resultar confuso para principiantes.

A veces dan problemas en su reconocimiento en el IDE de Arduino y en la subida de los proyectos a la placa.

El consumo de energía es algo mayor, aunque sigue siendo bajo. El ESP32 consume más corriente que el ATmega328P (~80 mA frente a ~30 mA).

Casos de uso ideales
Experimentos en los que se interesante enviar los datos por Bluetooth a los teléfonos móviles.

Experimentos que se beneficien de la alta frecuencia de muestreo del procesador.

Permite la toma de datos remota con registro en tarjeta SD y transmisión por WiFi periódica. Por ejemplo, en una estación meteorológica autónoma que publique datos en una web.

Realización de proyectos IoT donde se necesite conexión a Internet.

Precauciones al usar ESP32
Aunque se puedan conectar sensores a 5 V, las entradas analógicas del ESP32 solo soportan hasta 3,2 V (a diferencia del Arduino UNO que soporta hasta 5 V). Si se utilizan shields que emplean la

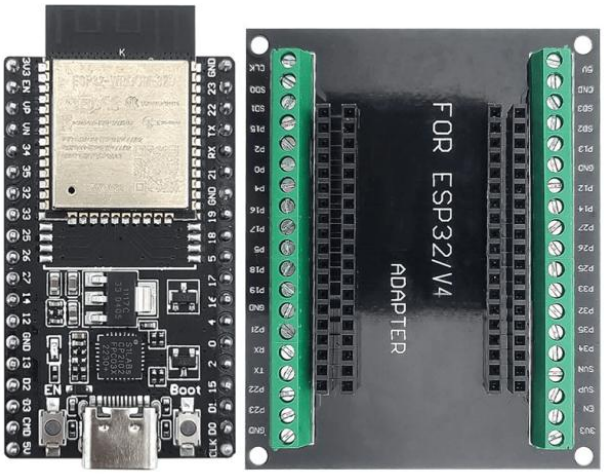
entrada analógica, es posible que no funcionen correctamente.

Puede ser necesario en el código tener en cuenta que el ESP32 funciona a 3,3 V.

Como el conversor ADC es de 12 bits, para convertir el nivel leído en voltaje se debe dividir por 4095 (2^{12} = 4096).

Otras placas con ESP32

Además de la Wemos D1 R32, existen muchas otras placas que montan el ESP32: DOIT ESP32 DEVKIT V1, ESP32 Dev Module, etc. En el IDE de Arduino se puede seleccionar cualquiera de ellas (a menudo es compatible usar el módulo genérico ESP32 Dev Module).



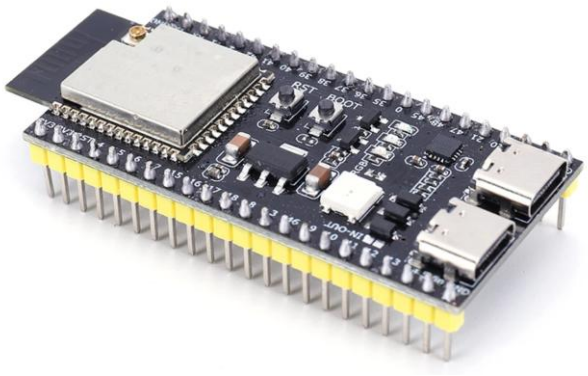
Placa genérica ESP32 con los pines soldados y un adaptador para pincharla y que resulte más fácil la conexión con los sensores.

También han cobrado gran relevancia las placas basadas en la evolución de este chip: el **ESP32-S3**. Estas placas (como la ESP32-S3-WROOM-1 o otros modelos de diversos fabricantes) mantienen el procesador de doble núcleo y la conectividad Wi-Fi y Bluetooth (ahora Bluetooth 5 LE), pero añaden mejoras significativas (aunque yo todavía no les he encontrado uso en la experimentación), manteniendo un precio muy similar a las clásicas:

Soporte USB nativo (OTG): Permiten la conexión directa sin necesidad de un chip conversor USB-Serie, abriendo la puerta a emular periféricos (como teclados o ratones) directamente desde el código.

Mayor cantidad de pines GPIO disponibles y más memoria (SRAM y a menudo PSRAM), ideal para capturar y procesar grandes conjuntos de datos de sensores.

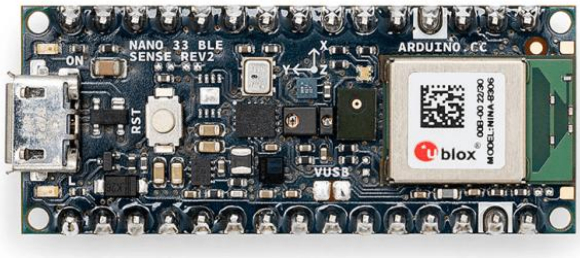
Instrucciones vectoriales orientadas a redes neuronales y aprendizaje automático (TinyML), que permiten procesar datos en el propio microcontrolador.



ESP32-S3

Arduino Nano 33 BLE Sense

Esta placa, diseñada por el proyecto Arduino, está basada en el Arduino Nano 33 BLE, al que incorporaron una gran cantidad de sensores. El nombre lo indica: **Nano** (dimensiones reducidas), **33** (funciona a 3,3 V), **BLE** (Bluetooth Low Energy) y **Sense** (cantidad de sensores).



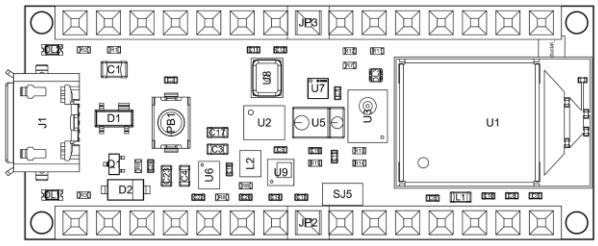
Arduino Nano 33 BLE Sense Rev2

Características principales

- Microprocesador: nRF52840 (ARM Cortex-M4F)
- Tensión de funcionamiento: 3,3 V
- Alimentación: por USB o por VIN (4,5 a 21 V); con dos pilas CR2032 de 3 V es suficiente
- Conectividad: Bluetooth BLE
- Precio: unos 45 € la versión Rev2, que es la que venden ahora en la tienda oficial de Arduino

Sensores integrados
La gran ventaja de esta placa es que integra en una tarjeta minúscula los siguientes sensores:

- Sensor de luz, proximidad, color y gestos (APDS9960)
- Micrófono digital (MP34DT05)
- Acelerómetro y giroscopio (LSM9DS1)
- Magnetómetro (LSM9DS1)
- Sensor de temperatura y humedad (HTS221 en la versión 1, HS300x en la Rev2)
- Sensor de presión (LPS22HB)



Board topology top

Ref.	Description	Ref.	Description
U1	NINA-B306 Module Bluetooth® Low Energy 5.0 Module	U6	MP2322GQH Step Down Converter
U2	BMI270 Sensor IMU	PB1	IT-1185AP1C-160G-GTR Push button
U3	MP34DT06JTR MEMS Microphone	U8	HS3003 Humidity Sensor
U7	BMM150 Magnetometer IC	DL1	Led L
U5	APDS-9660 Ambient Module	DL2	Led Power
U9	LPS22HBTR Pressure Sensor IC		

Ubicación de los sensores en el Arduino Nano 33 BLE Sense Rev2

Ventajas del Nano 33 BLE Sense
Es un todo en uno al integrar múltiples sensores, lo que evita tener que comprarlos, cablearlos y configurarlos por separado.

Tiene un tamaño mínimo. Es un formato Nano (45 × 18 mm), ideal para proyectos portátiles o para incorporar en dispositivos compactos.

La comunicación bluetooth es BLE (Bluetooth Low Energy), que consume muy poca energía, permitiendo que funcione con pilas pequeñas durante horas.

Los sensores vienen calibrados de fábrica, lo que garantiza medidas fiables sin necesidad de calibración manual.

Además, es una placa con numerosos pines de entrada y salida, así que se le pueden conectar los mismos sensores externos que a cualquier otra placa microcontroladora.

Es ideal para llevarla puesta, ya que su reducido tamaño y bajo consumo la hacen perfecta para experimentos de movimiento corporal, ritmo cardíaco, etc.

Limitaciones del Nano 33 BLE Sense
Precio elevado: unos 45 €, frente a los 4 € de un Arduino UNO o un ESP32.

No tiene WiFi, solo dispone de Bluetooth BLE. Si se necesita conexión a Internet, hay que recurrir a otra placa.

No tiene conexiones para 5 V y exclusivamente todos los pines trabajan a 3,3 V. No es compatible directamente con shields o sensores diseñados para 5 V (aunque algunos funcionen igual).

Requiere Mbed OS. El ecosistema de software es diferente al Arduino tradicional, de manera que algunas librerías estándar de Arduino no son compatibles directamente.

Aunque tiene un termómetro integrado, su lectura viene condicionada por el calentamiento de la propia placa al funcionar, así que sus valores no son nada fiables.

Casos de uso ideales
Experimentos de movimiento con acelerómetro y giroscopio integrados.

Proyectos que se comuniquen con la aplicación Phyphox mediante BLE. Es compatible con las aplicaciones Phyphox y Arduino Science Journal a las que puede enviar los datos de los sensores, lo que se explica en un capítulo específico.

Experimentos de fonocardiografía o análisis de sonido con el micrófono integrado.

Medida de valores con los diversos sensores que integra, como por ejemplo del campo magnético terrestre con el magnetómetro LSM9DS1.

Comparativa de placas

Característica	Arduino UNO R3	Wemos D1 R32 (ESP32)	Arduino Nano 33 BLE Sense
Microcontrolador	ATmega328P	ESP32-WROOM-32D	nRF52840
Voltaje de trabajo	5 V	3,3 V	3,3 V
ADC	10 bits (0-1023)	12 bits (0-4095)	12 bits
WiFi	No	Sí	No
Bluetooth	No	Sí (BLE)	Sí (BLE)
Sensores integrados	No	No	Sí (varios)
Precio orientativo	~4 €	~4 €	~45 €

¿Qué placa elegir para cada experimento?

Si se necesitan entradas analógicas pura (pH, conductividad con señaos débiles) usad Arduino UNO R3. Los ADC son auténticos y lineales en todo el rango de 0-5 V.

Si se necesita bluetooth para Phyphox, por ejemplo, usad Wemos D1 R32 (ESP32) o Arduino Nano 33 BLE Sense. El ESP32 es más económico, pero el Nano 33 BLE Sense es más compacto y tiene sensores integrados.

Si se necesita WiFi usad Wemos D1 R32 (ESP32). Es la opción más económica con WiFi integrado.

Si se usan sensores I²C cualquiera de las tres placas es compatible. Preferid ESP32 por su mayor versatilidad.

Si el presupuesto es limitado el ESP32 (Wemos D1 R32) ofrece la mejor relación calidad-precio, con WiFi, Bluetooth y suficiente potencia para la mayoría de experimentos.

Al final la elección de la placa adecuada depende del tipo de experimento, los sensores involucrados y la necesidad de conectividad inalámbrica. La siguiente tabla os puede ayudar a seleccionar la placa más apropiada:

Tipo de experimento	Placa recomendada	Motivo
Experimentos contemplados en este libro	La que se indique en cada caso	Es la placa o placas que mejor se ajustan a la situación.
Medidas analógicas de precisión (pH, conductividad con señal débil)	Arduino UNO R3	Entradas analógicas puras y lineales de 0-5 V. El ESP32 no es fiable para este tipo de medidas.
Experimentos con envío de datos a Phyphox	Wemos D1 R32 o Nano 33 BLE Sense	Ambos disponen de Bluetooth BLE. El Nano 33 BLE Sense es más compacto; el ESP32 es más barato.
Estación meteorológica autónoma con registro remoto	Wemos D1 R32 (ESP32)	WiFi integrado permite subir datos a la nube. Suficiente memoria para logging prolongado.
Experimentos de movimiento y aceleración del cuerpo	Arduino Nano 33 BLE Sense	Lleva acelerómetro y giroscopio integrados y calibrados. Tamaño mínimo para llevar puesto.
Experimentos con múltiples sensores I²C	Cualquiera (preferir ESP32)	El ESP32 ofrece más flexibilidad y velocidad, aunque el UNO también es compatible.
Experimentos escolares básicos (sin inalámbrico)	Arduino UNO R3	La placa más común en centros educativos. Más sencilla para empezar.
Toma de datos de alta frecuencia	Wemos D1 R32 (ESP32)	Procesador a 240 MHz permite mayores frecuencias de muestreo.
Proyecto compacto y portátil con pilas	Arduino Nano 33 BLE Sense	Tamaño reducido, bajo consumo y sensores integrados.

Alimentación eléctrica de los microcontroladores

En los montajes experimentales, a menudo resulta restrictivo mantener el microcontrolador conectado al ordenador permanentemente, ya que algunos experimentos se desarrollan al aire libre, en movimiento o exigen mediciones ininterrumpidas durante días, situaciones donde la conexión a un PC no es viable. Por ello, os propongo algunas opciones de alimentación autónoma para las placas Arduino y ESP32.

Cualquiera de las placas de las que hablamos puede funcionar de forma independiente si se le suministra la corriente y el voltaje adecuados. Existen diversas estrategias para alimentarlas:

Alimentación mediante puerto USB (5V)

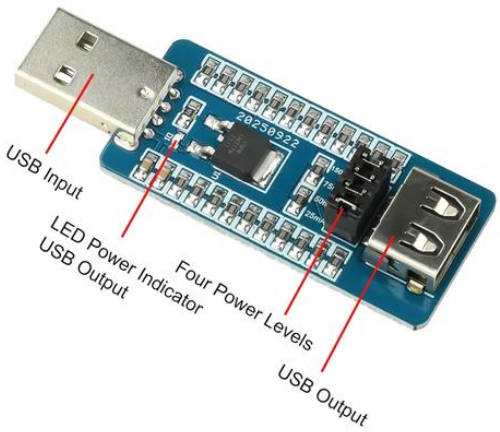
Es el método más directo y seguro, ya que la placa recibe exactamente los 5 V que necesita para operar, puenteando o minimizando el trabajo de su regulador interno.

Las **baterías portátiles** (Powerbanks) diseñadas para cargar teléfonos móviles son excelentes para suministrar energía a los microcontroladores en experimentos portátiles ya que proporcionan 5 V estables mediante una conexión USB.

Pero ¡Ojo!, muchos powerbanks modernos tienen una función de apagado automático si la corriente de salida es muy baja (típicamente inferior a 50-100 mA). Como un Arduino UNO o un ESP32 en reposo pueden consumir menos de esa cantidad, la batería puede apagarse a los pocos minutos, interrumpiendo el experimento.

Para solucionarlo se pueden usar powerbanks diseñados para baja corriente (para dispositivos IoT, auriculares) que son mucho más caros que los estándar, o añadir a estos últimos una pequeña carga (como un LED) que incremente el consumo por encima del umbral de apagado.

Una alternativa al led de arriba son los módulos de carga virtual antiapagado de powerbaks, que se conectan entre la batería portátil y el microprocesador, y que tienen precios asequibles (unos 3 €).



Módulo de carga virtual antiapagado de powerbaks

Los **cargadores de teléfono a la red eléctrica** son ideales si el experimento requiere varios días o semanas (por ejemplo, una estación meteorológica o la monitorización del pH en un acuario) y se dispone de una toma de corriente cercana. Un cargador estándar de 5 V con salida USB garantiza una alimentación suficiente e ininterrumpida.

Alimentación externa por clavija (Jack DC) o pines VIN

Las placas de tamaño estándar, como el Arduino UNO R3 y la tarjeta Wemos D1 R32 (ESP32), incluyen una clavija de alimentación negra (Jack de barril de 2,1 mm) y pines etiquetados como **VIN** (Voltaje de Entrada).

Estas entradas pasan a través del regulador de voltaje interno de la placa, el cual se encarga de reducir el voltaje suministrado al nivel operativo de la lógica (5 V en Arduino UNO; 3,3 V y 5 V para el ESP32 en la placa Wemos).

Es recomendable suministrar al Arduino Uno un voltaje entre 7 V y 12 V. Si se aplica menos de 7 V (por ejemplo, 5 V o 6 V), el regulador interno puede tener una caída de tensión (dropout) que deje el voltaje operativo final por debajo del requerido, causando inestabilidad. Si se aplica más de 12 V, el regulador tendrá que disipar el exceso de energía en forma de calor, pudiendo sobrecalentarse y dañarse.

La alimentación más adecuada para estas entradas es un portapilas (AA o AAA) con 6 pilas que proporciona unos 9 V (1,5 V x 6). Esta es una solución fiable y robusta, con buena capacidad de corriente para motores o sensores exigentes.

Aunque es común ver conectores para pilas de 9 V (tipo bloque/PP3) a Jack DC, su uso es desaconsejable en general y más para experimentación prolongada o exigente. Tienen una capacidad muy baja (aprox. 500 mAh) y no pueden suministrar los picos de corriente elevados que requiere un ESP32 cuando escribe en una tarjeta SD de datos o activa su módulo WiFi, provocando reinicios inesperados.

Consideraciones sobre el tipo de alimentación y los valores obtenidos

Al cambiar el método de alimentación de USB a alimentación externa (VIN o Jack DC), se debe prestar especial atención a las mediciones de los sensores analógicos.

El conversor analógico-digital (ADC) de placas como el Arduino UNO utiliza por defecto el voltaje de alimentación (Vcc) como voltaje de referencia (los 5 V nominales). Cuando la placa se alimenta desde el puerto USB del ordenador, el voltaje real suele estar muy próximo a 5,0 V. Sin embargo, al usar el regulador interno desde VIN, este voltaje puede ser ligeramente distinto (por ejemplo, 4,9 V o 5,1 V).

Esta pequeña desviación altera el valor de referencia del ADC, provocando que las lecturas en voltios calculadas en el código difieran ligeramente del valor real. Por rigor metodológico, se deben calibrar los sensores utilizando siempre la misma fuente de alimentación que se empleará durante el experimento definitivo.

Si se va a medir alimentando la placa con un powerbank, la calibración previa y la determinación de las constantes físicas también debe hacerse alimentando el circuito con ese mismo powerbank.

Los sensores

En este capítulo presento los sensores que utilizaremos a lo largo del libro (y alguno más) para la captación de datos en los experimentos de Física y Química. Todos ellos son sensores de bajo coste, fácilmente disponibles en tiendas de electrónica o plataformas online, y compatibles con las placas Arduino y ESP32.

Cada sensor se describe siguiendo una estructura homogénea: principio físico de funcionamiento, características técnicas, conexionado, calibración (cuando proceda) y aplicaciones prácticas en experimentos de física.

Al final del libro, en las referencias bibliográficas, se encuentran los enlaces a sus diferentes fichas de datos (datasheet) y también a las diferentes librerías convenientes para la simplificación del código de los correspondientes proyectos (sketch).

A pesar de que todos los sensores se describen de manera similar, no todos son igual de importantes para la experimentación en física y no todos los utilizo por igual. Dejadme que os proponga el ranquin de los más interesantes:

Sensores de distancia
Tanto el sensor de ultrasonidos HC-SR04 como el de infrarrojo VL53L0X.

Sensor de temperatura
El sensor DS18B20 es el rey.

Sensor de presión
Sensor de presión (y temperatura y humedad) BME280.

Sensores de luz
Sensor TSL2591, las fotorresistencias y los fototransistores.

Sensores de proximidad y barreras de luz
Sensor KY-032 y el led y fototransistor IR

Sensor de fuerza
Célula de carga más amplificador HX711

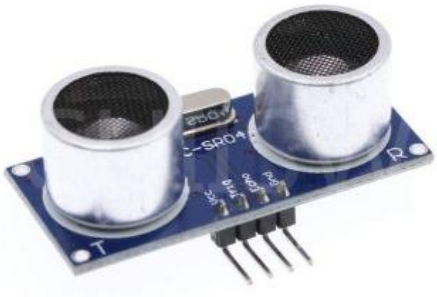
Sensores magnéticos
Sensor Hall SS49E y el magnetómetro HMC5883L.

No incluyo sensores de sonido ya que en este ámbito los smartphones y los ordenadores son imbatibles.

1. Sensores de distancia

Estos sensores son de los más interesantes ya que no tienen alternativa en los sensores de los teléfonos móviles.

1.1 Sensor de ultrasonidos HC-SR04



Principio de funcionamiento
El HC-SR04 es un sensor de distancia por ultrasonidos. Su funcionamiento se basa en la ecolocalización: emite un pulso de sonido de alta frecuencia (40 kHz), no audible para el ser humano, que se propaga por el aire, rebota en los objetos cercanos y regresa al sensor. Midiendo el tiempo transcurrido entre la emisión y la recepción del eco, y conociendo la velocidad del sonido en el aire, se calcula la distancia.

La velocidad del sonido en condiciones normales (20 °C, 50 % humedad, 1 atm) es de 343 m/s. El sonido tarda 29,15 µs en recorrer 1 cm. La distancia se obtiene mediante:

$$\text{distancia (cm)} = \text{tiempo (µs)} / 58{,}3$$

$$d(cm) = v(m/s)*t(\mu s)*10^{-6}*100/2 =$$
$$= 343*5*10^{-5}* t(\mu s) = t(\mu s)/58{,}3$$

Se divide entre dos porque el tiempo medido corresponde al recorrido de ida y vuelta.

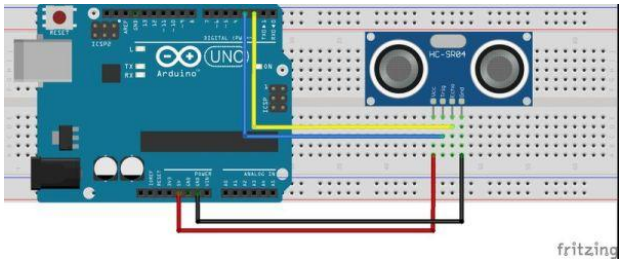
Características técnicas

- Dimensiones: 43 × 20 × 17 mm
- Tensión de alimentación: 5 Vcc
- Frecuencia de trabajo: 40 kHz
- Distancia máxima: 4 m
- Distancia mínima: 2 cm
- Ángulo de trabajo óptimo: <15°
- Duración mínima del pulso de disparo (TTL): 10 µs
- Duración del pulso de eco (TTL): 100-25000 µs
- Tiempo mínimo entre medidas: 20 ms
- Pines: VCC (5 Vcc), Trig (disparo), Echo (recepción), GND (tierra)
- Precio: desde 2 €

Este sensor no necesita librerías para su programación.

Pines de conexión

Pin	Descripción
VCC	Alimentación, 5 V
GND	GND. Tierra, 0 V
Trig	Pin digital (D3)
Echo	Pin digital (D2)



Usos en experimentos de física

Medición de distancias y desplazamientos en cinemática
Seguimiento de la posición de un móvil en caída libre o plano inclinado
Determinación de la velocidad del sonido en el aire
Estudio del movimiento periódico de un péndulo
Medición del nivel de líquidos por reflexión acústica

1.2 Sensor VL53L0X (láser infrarrojo)

Principio de funcionamiento

El VL53L0X es un sensor TOF (Time of Flight, tiempo de vuelo) que mide con gran precisión el tiempo que tardan los pulsos de luz láser infrarroja (940 nm) en viajar desde el emisor hasta el objeto más cercano y regresar al receptor.

A diferencia del HC-SR04, que usa ondas sonoras, el VL53L0X emplea luz, lo que le confiere mayor precisión y una velocidad de muestreo mucho más alta. Incorpora una matriz de SPAD (Singos Photon Avalanche Diodes) capaz de detectar fotones individuales, lo que permite medir distancias con resolución milimétrica.

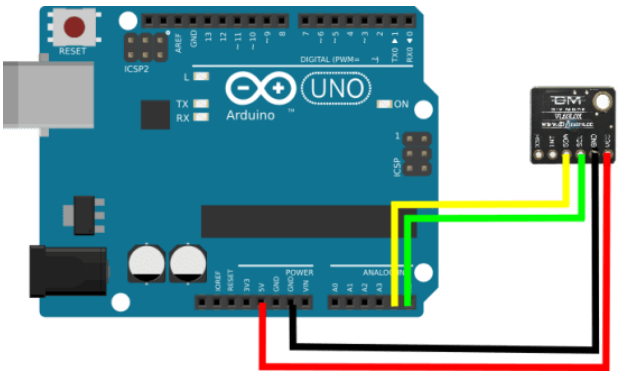


Características técnicas

- Tensión de alimentación: 2,6 a 5,5 V
- Rango de medida: 40 mm a 2 m
- Resolución: 1 mm
- Precisión: ±3 % en condiciones óptimas, >±10 % en condiciones menos favorables
- Interfaz: I²C
- Campo de visión (FOV): 25°
- Precio: 3-4 €

Pines de conexión

Pin	Descripción
VIN	Alimentación (2,6-5,5 V)
GND	Tierra (0 V)
SCL	Línea de reloj I²C
SDA	Línea de datos I²C
XSHUT	Apagado (normalmente no se usa)
INT	Interrupción (opcional)

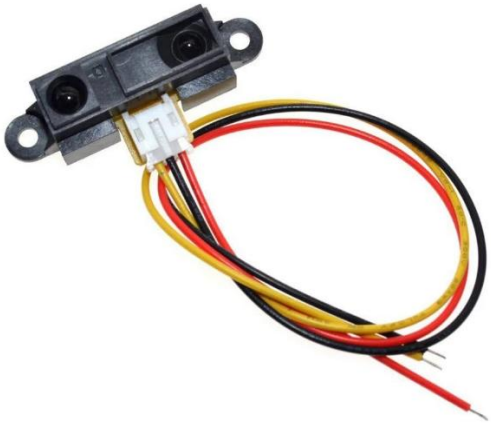


La librería que normalmente utilizo para este sensor es la Adafruit VL53L0X.

Usos en experimentos de física
Seguimiento de objetos en movimiento con alta precisión (caída libre, rebotes)
Medida de la velocidad instantánea en cinemática
Experimento de conservación de la energía: seguimiento de la altura de un rebote
Calibración de otros sensores de distancia

1.3 Sensor de distancia por infrarrojos SHARP GP2Y0A21

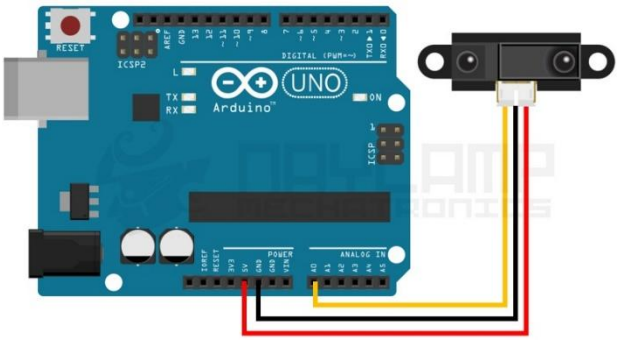
Principio de funcionamiento
El SHARP GP2Y0A21 es un sensor de distancia por infrarrojos que proporciona una **salida analógica** inversamente proporcional a la distancia. A diferencia de los sensores de proximidad digitales (Flying-fish, KY-032), el GP2Y0A21 mide la distancia real de forma continua. Utiliza triangulación óptica: se emite un haz IR, se refleja en el objeto y llega a un PSD (Position Sensitive Detector). La posición del punto de luz en el PSD varía con la distancia del objeto, produciendo un voltaje analógico.



Características técnicas
- Rango: 10-80 cm (el GP2Y0A02, 20-150 cm)
- Salida: analógica (2,8 V a 10 cm, 0,4 V a 80 cm)
- Tiempo de respuesta: ~30 ms
- Precio: ~6 €

Pines de conexión

Pin	Descripción
Vin	Alimentación (4,5-5,5 V)
GND	Tierra (0 V)
Vout	Pin analógico, A0



Usos en experimentos de física
Medida de distancia sin contacto con salida continua (analógica)
Experimento de cinemática (caída libre, plano inclinado)
Robot seguidor de paredes
Medición del nivel de líquidos por reflexión

Comparativa de sensores de distancia

Caracte-rística	HC-SR04	VL53L0X	GP2Y0A21
Tipo	Ultrasoni-dos	Láser IR (TOF)	IR
Rango	2 cm - 4 m	4 cm - 2 m	10-80 cm
Precisión	±3 mm		±1 cm
Ángulo	15°	25°	
Alimentac ión	5 V	2,6-5,5 V	4,5-5,5 V
Interfaz	Digital Trig/Echo	I²C	Analógica
Precio	~2 €	~3 €	~6 €

2. Sensores de temperatura

Los sensores de temperatura son también de gran interés ja que su alternativa en los teléfonos móviles se restringe a un termómetro en su interior que viene determinado por el calentamiento del propio móvil. Los sensores de temperatura que se pueden utilizar con los micrcontroladores no sólo son exteriores, sino que algunos de ellos se pueden sumergir en líquidos.

2.1 Sensor DS18B20

Principio de funcionamiento
El DS18B20 es un sensor digital de temperatura que utiliza un oscilador de cristal cuya frecuencia varía con la temperatura. Internamente cuenta con un conversor analógico-digital de 12 bits que digitaliza

la señal del sensor térmico. Se comunica mediante el protocolo **1-Wire**, que permite transmitir datos y alimentación por un solo cable (más tierra). Cada DS18B20 tiene una dirección única de 64 bits grabada de fábrica, lo que permite conectar múltiples sensores en el mismo bus y leerlos individualmente.

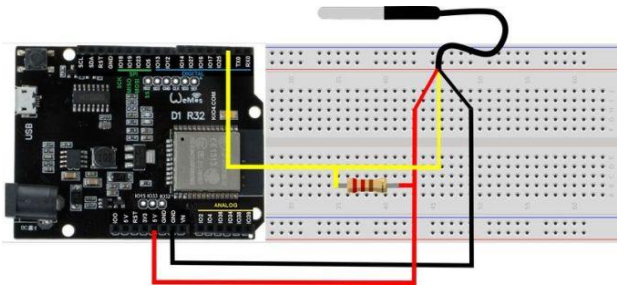
Características técnicas

- Carcasa: acero inoxidable (6×50 mm), resistente al agua
- Alimentación: 3,0 a 5,5 V
- Rango de temperatura: -55 a +125 °C
- Precisión (-10 a 85 °C): ±0,5 °C
- Precisión (resto del rango): ±2 °C
- Resolución programable: 9 a 12 bits (12 bits por defecto)
- Protocolo: 1-Wire (un solo pin para datos)
- Dirección única de 64 bits (permite conectar varios sensores en el mismo bus)
- Precio: ~2 €



Cables de conexión

Pin	Descripción
Rojo	Alimentación (3,0-5,5 V)
Negro	GND, Tierra (0 V)
Amarillo	Datos, pin One-wire
Requiere una resistencia pull-up de 4,7 kΩ entre los cables de voltaje y de datos.	



Usos en experimentos de física

Medición de temperatura en líquidos (al ser sumergible)
Calorimetría: determinación del calor específico de sustancias
Estudio de curvas de calentamiento y enfriamiento
Experimento de calor latente de fusión del hielo
Seguimiento de reacciones endotérmicas y exotérmicas
Medición de temperatura en estaciones meteorológicas

2.2 Sensor LM35

Principio de funcionamiento

El LM35 es un sensor analógico de temperatura cuya salida de voltaje es linealmente proporcional a la temperatura en grados Celsius. Proporciona 10 mV por cada grado Celsius, sin necesidad de calibración externa. A 0 °C entrega 0 V y a 100 °C entrega 1,0 V. Está basado en un circuito integrado que explota la dependencia térmica de la tensión base-emisor de un transistor.





Características técnicas

- Alimentación: 4-30 V
- Salida: 10 mV/°C
- Rango: 0-100 °C (LM35), -55 a 150 °C (LM35D)
- Precisión: ±0,5 °C a 25 °C
- Encapsulado: TO-92 (no impermeable)
- Precio: <1 €

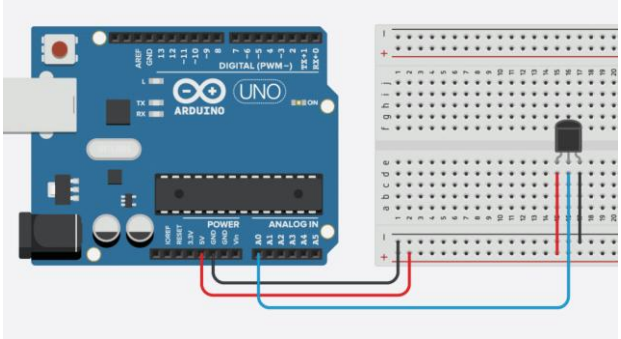
Usos en experimentos de física

Medición de temperatura ambiente
Calibración de termistores NTC
Experimento de enfriamiento de Newton

Estación meteorológica básica (con un DHT11 para humedad)

Pines de conexión

Pin	Descripción
VCC (izquierdo)	Alimentación (4-30 V)
VOUT (centro)	Salida analógica (10 mV/°C)
GND (derecho)	Tierra



Advertencia: el LM35 no es sumergible ya que su encapsulado TO-92 no está sellado contra la humedad. Para líquidos, usad DS18B20 en cápsula de acero inoxidable.

2.3 Termistores NTC

Principio de funcionamiento

Un termistor NTC (Negative Temperature Coefficient) es una resistencia cuyo valor disminuye al aumentar la temperatura. Está fabricado con óxidos semiconductores (manganeso, níquel, cobalto) cuya resistencia sigue una relación exponencial con la temperatura, descrita por la ecuación de Steinhart-Hart:

$$1/T = A + B \cdot ln(R) + C \cdot \left(ln(R)\right)^3$$

donde T es la temperatura en Kelvin, R la resistencia y A, B, C coeficientes específicos de cada termistor. Al ser altamente no lineales, requieren calibración o el uso de curvas características proporcionadas por el fabricante.

Características técnicas

- NTC MF52 (encapsulado perla): para temperatura ambiente (-40 a 120 °C)

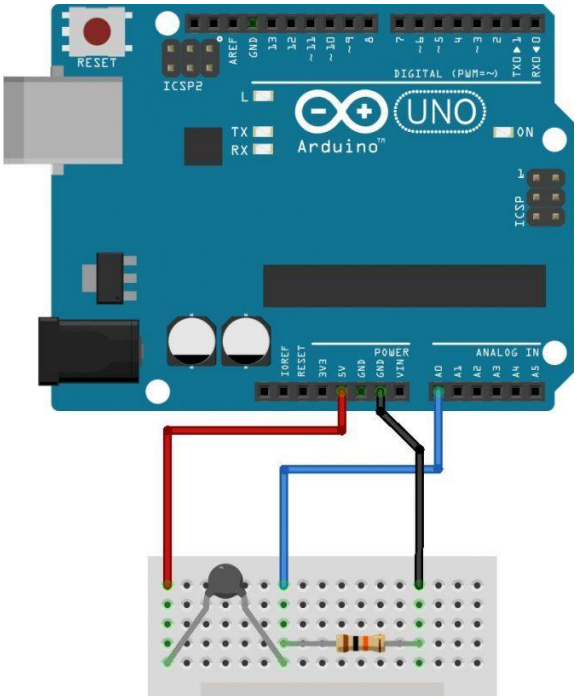


- NTC 3950 (funda metálica impermeable): para líquidos (-30 a 120 °C)



- Precio: <0,50 €

Conexiones



Usos en experimentos de física

Medición de temperatura de bajo coste con calibración

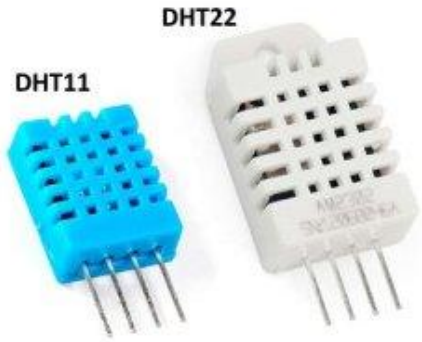
Estudio de la relación resistencia-temperatura en semiconductores

Práctica de calibración de sensores

2.4 Sensor DHT11/DHT22

Principio de funcionamiento

El DHT11 es un sensor digital que integra un termistor NTC para la medida de temperatura y un sensor capacitivo de humedad. El sensor capacitivo consiste en un dieléctrico que absorbe agua, variando su capacidad en función de la humedad relativa. Un microcontrolador interno procesa las señales y las envía por un bus digital serie de un solo cable con protocolo propietario.

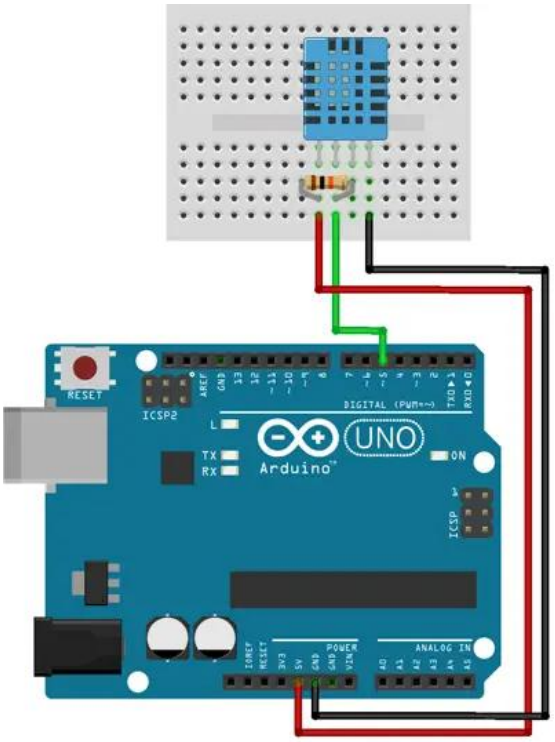


Características técnicas

Característica	DHT11	DHT22
Rango temperatura	0-50 °C	-40 a 80 °C
Precisión temperatura	±2 °C	±0,5 °C
Rango humedad	20-90 %	0-100 %
Precisión humedad	±5 %	±2 %
Frecuencia muestreo	1 Hz	2 Hz
Precio	<1 €	~3 €

Pines de conexión

Pin	Descripción
Vin	Alimentación (5 V)
GND	Tierra (0 V)
Vout	Pin digital, D5
Requiere una resistencia pull-up de 10 kΩ entre los cables de voltaje y de datos.	

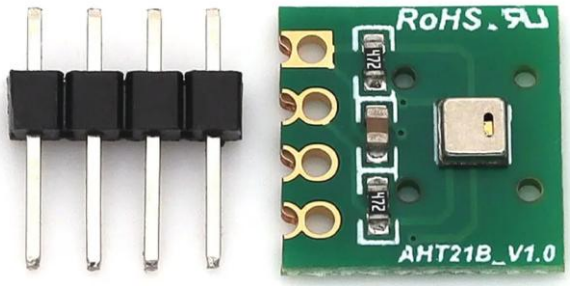


Usos en experimentos de física

Estación meteorológica doméstica
Estudio de la humedad relativa y su relación con la temperatura
Experimento de punto de rocío
Control ambiental en invernaderos y terrarios

2.5 Sensores AHT2X (AHT20/AHT21)

Los sensores AHT20 y AHT21 (también AHT30), fabricados por Aosong Electronics, el mismo fabricante de los anteriores (y más populares) DHT11/DHT22, representan su evolución natural. A diferencia de sus predecesores, toda la familia AHT2X emplea un bus I²C estándar, elimina los problemas de temporización del protocolo propietario de un solo cable del DHT22 y ofrece una estabilidad a largo plazo muy superior.



Principio de funcionamiento

El AHT20/AHT21 integra en un único encapsulado dos transductores independientes:

Uno es el sensor de temperatura que consta de un termistor de silicio de coeficiente negativo (NTC) cuya resistencia varía con la temperatura. El microcontrolador interno digitaliza la señal mediante un ADC de alta resolución y aplica una curva de compensación lineal almacenada en la memoria OTP (One-Time Programmable) calibrada de fábrica.

Otro es el sensor de humedad relativa que es un sensor capacitivo basado en un dieléctrico polímero que absorbe reversiblemente moléculas de agua del ambiente. La constante dieléctrica del polímero (ε_r) varía con la cantidad de agua adsorbida, modificando la capacidad del condensador. El ADC interno convierte esta variación capacitiva en un valor digital de humedad relativa (%HR).

El AHT20 y el AHT21 son funcionalmente idénticos en su interfaz I²C y registro interno, salvo en su precisión ya que el AHT21 ofrece ±0,2 °C frente a ±0,3 °C del AHT20 en el rango de 0–60 °C, y una histéresis de humedad ligeramente inferior.

Características técnicas

Parámetro	AHT20	AHT21	DHT22
Tensión alimentación	2,2 – 5,5 V	2,2 – 5,5 V	3,3 – 5,5 V
Rango temperatura	–40 °C a +85 °C	–40 °C a +85 °C	–40 °C a +80 °C
Precisión Tª	±0,3 °C (0–60 °C)	±0,2 °C (0–60 °C)	±0,5 °C (0–50 °C)
Resolución Tª	0,01 °C	0,01 °C	0,1 °C
Rango humedad	0 – 100 %HR	0 – 100 %HR	0 – 100 %HR
Tiempo de respuesta	< 8 s	< 5 s	< 15 s
Frecuencia de muestreo	Hasta 20Hz (cada 50 ms)	Hasta 20 Hz (cada 50 ms)	0,5 Hz (cada 2 s)
Interfaz	I²C (0x38)	I²C (0x38)	1-Wire propietario
Longitud máxima cable	limitación I²C ~1 m	limitación I²C ~1 m)	~20 m (con resistencia pull-up)
Calibración fábrica	Sí (OTP, compensación Tª)	Sí (OTP, compensación Tª)	Sí (OTP básica)
Precio orientativo	~1,5 €	~2 €	~3 €

Ventajas clave del AHT2X frente al DHT22

- Utiliza el protocolo I²C estándar de manera que no requiere librerías específicas. Se integra directamente con Wire.h y puede compartir el bus con otros sensores I²C (BMP280, BH1750, TCS34725, etc.).
- Su velocidad de muestreo es de hasta 20 lecturas por segundo (50 ms por lectura), frente a 1 lectura cada 2 segundos del DHT22. Esto permite estudiar fenómenos térmicos transitorios de evolución rápida.
- Presenta estabilidad a largo plazo ya que el encapsulado sellado y la calibración OTP con compensación de temperatura reducen la deriva temporal. El DHT22 es notoriamente propenso a la deriva en humedad tras meses de uso continuo.
- Tiene menor consumo: 19 µA de medida frente a ~1 mA del DHT22, lo que es crítico en aplicaciones alimentadas por batería.
- Es menos sensible a interferencias electromagnéticas que el protocolo de un solo cable del DHT22, cuyo flanco de temporización puede ser perturbado por cables largos o motores próximos.

Limitaciones del AHT2X frente al DHT22

- La longitud de cable que se puede utilizar es más corta ya que I²C está diseñado para comunicaciones intra-placa o de corta distancia (~1 m). El DHT22, al usar un protocolo serie asíncrono de un solo cable, puede funcionar con cables de hasta 20 m si se colocan resistencias pull-up adecuadas, lo que lo hace más adecuado para sensores remotos.
- Aunque dispone de librerías maduras (Adafruit AHTX0, SparkFun AHT20), el ecosistema del DHT22 es aún más extenso por su antigüedad.

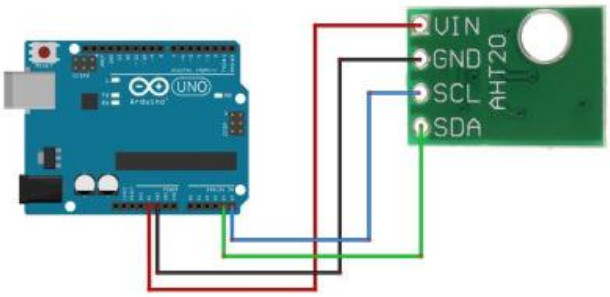
Pines de conexión

El módulo AHT20/AHT21 más común (Gravity o genérico) dispone de 4 pines:

Pin	Conexión al microcontrolador
VCC (Alimentación)	3,3 V o 5 V
GND (Tierra)	GND
SCL (Reloj I²C)	SCL (Arduino UNO: A5; ESP32: GPIO 22)
SDA (Datos I²C)	SDA (Arduino UNO: A4; ESP32: GPIO 21)

La dirección I²C por defecto del AHT20/AHT21 es 0x38 (7 bits). Ninguno de los dos modelos permite cambiar la dirección mediante pines físicos, por lo

que no es posible conectar dos AHT2X en el mismo bus I²C sin un multiplexor (como el TCA9548A).



Usos en experimentos de física

Estación meteorológica de alta velocidad. Gracias a su tasa de 20 Hz, permite registrar la respuesta térmica a perturbaciones rápidas (apertura de una nevera, pulso de calor radiante, viento de un ventilador) con una resolución temporal que el DHT22 no puede alcanzar.

Curvas de enfriamiento (ley de Newton del enfriamiento). La alta tasa de muestreo permite capturar con precisión la evolución exponencial de la temperatura de un cuerpo que se enfría, ajustando la constante de tiempo $\tau = Rt \cdot C$ (resistencia térmica × capacidad calorífica) con menos error que con sensores lentos.

Calor latente de evaporación. Midiendo temperatura y humedad simultáneamente a 20 Hz durante la evaporación de un líquido (agua, etanol) sobre una superficie, se puede correlacionar el descenso térmico (enfriamiento evaporativo) con el aumento de humedad relativa en el entorno inmediato.

Perfiles térmicos en experimentos de conducción. Midiendo temperatura en múltiples puntos (con varios AHT2X multiplexados o combinados con DS18B20) a lo largo de una barra metálica sometida a un gradiente térmico, para verificar la ley de Fourier de conducción del calor.

Comparativa de algunos sensores de temperatura

Característica	DS18B20	LM35	NTC MF52	DHT11
Tipo	Digital	Analógico	Analógico	Digital
Rango	-55 a 125 °C	0-100 °C	-40 a 120 °C	0-50 °C
Precisión	±0,5 °C	±0,5 °C	Variable	±2 °C
Sumergible	Sí	No	Sí	No
Interfaz	1-Wire	Analógica	Analógica	Digital
Precio	~2 €	<1 €	<1 €	<1 €

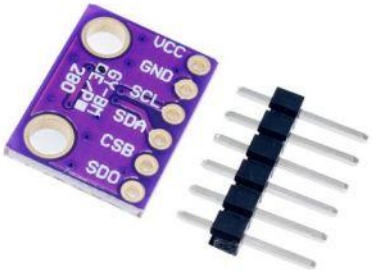
3. Sensores de presión (y temperatura y humedad)

Estos sensores que miden la presión atmosférica con gran sensibilidad son muy interesantes como alternativa a los teléfonos móviles que disponen de sensor de presión ya que de estos últimos hay pocos modelos y son los más caros, y además el tamaño de estos sensores para Arduino y ESP32 es muy reducido y permite introducirlos donde no entra un móvil.

3.1 BMP280 y BME280

Principio de funcionamiento

El **BMP280** y el **BME280** son sensores MEMS (Micro-Electro-Mechanical Systems) que miden presión atmosférica mediante una membrana piezorresistiva, donde la presión del aire deforma la micro-membrana de silicio, y esa deformación se traduce en un cambio de resistencia eléctrica. Además, los dos incorporan un sensor de temperatura para compensar las medidas de presión y el **BME280** añade un sensor capacitivo de humedad.



El **BMP280** mide solo presión y temperatura. El **BME280** mide presión, temperatura y humedad.

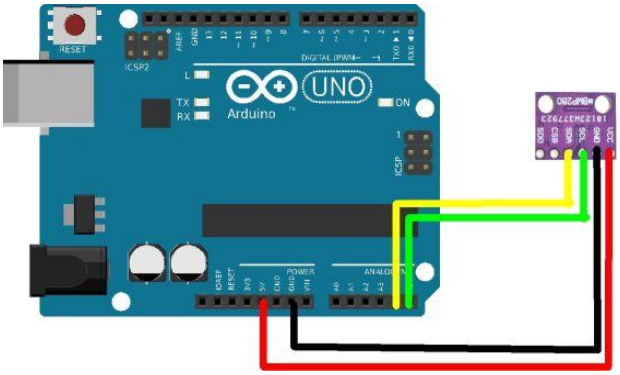
Características técnicas

Característica	BMP280	BME280
Dimensiones	15×12 mm	15×12 mm
Voltaje	1,8-5 Vcc	1,8-5 Vcc
Interfaz	I²C (3,4 MHz) o SPI (10 MHz)	I²C (3,4 MHz) o SPI (10 MHz)
Temperatura	-40 a +85 °C (resolución 0,01 °C, precisión ±1 °C)	-40 a +85 °C (resolución 0,01 °C, precisión ±1 °C)
Humedad		0-100 % (resolución 0,008 %, precisión ±3 %)
Presión	300-1100 hPa (resolución 0,18 Pa, precisión ±1 hPa)	300-1100 hPa (resolución 0,18 Pa, precisión ±1 hPa)
Dirección I²C por defecto	0x77 (0x76 si SDO se conecta a GND)	0x77 (0x76 si SDO se conecta a GND)
Precio	~3 €	~12 €

Los chips de los BME280 y BMP280 funcionan a 3,3 V pero los módulos comerciales aceptan hasta 5 V porque incorporan un regulador de voltaje

Pines de conexión

Pin	Descripción
VCC/VIN	Alimentación (5 V o 3,3 V)
GND	Tierra
SCL/SCK	Reloj serie (I²C: SCL; SPI: SCK)
SDA/SDI	Datos serie (I²C: SDA; SPI: SDI)
CSB	Selección de chip (SPI)
SDO	Selección dirección I²C



Usos en experimentos de física

Medida de presión atmosférica y su variación con la altura (barometría). Determinación experimental de la relación presión-altitud

Estación meteorológica completa (con BME280)

Estudio de la ley de Boyle-Mariotte (presión-volumen del aire)

Medición de la presión en función de la profundidad en un líquido

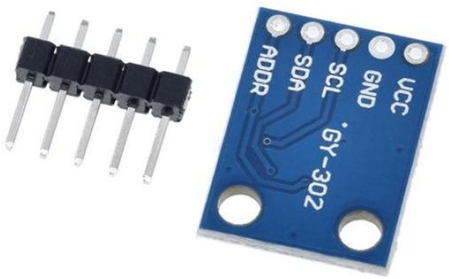
4. Sensores de luz y color

En el libro propongo diversos experimentos con sensores de luz, pero en algunas ocasiones los sensores de luz de los teléfonos móviles son mejor alternativa.

4.1 BH1750

Principio de funcionamiento

El BH1750 es un sensor digital de luz ambiente que mide la **iluminancia** en lux (lumen/m²), la unidad estándar del Sistema Internacional. Internamente utiliza un fotodiodo cuya corriente es proporcional a la luz incidente, un conversor analógico-digital de 16 bits y un microcontrolador que procesa la señal aplicando una curva de respuesta espectral calibrada para imitar la del ojo humano (fotópica). se comunica por I²C.

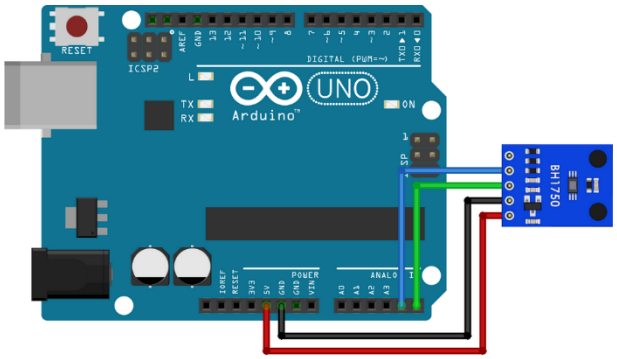


Características técnicas

- Tensión: 2,6 a 5,5 V
- Rango: 1-65535 lux
- Resolución: hasta 0,5 lux
- Interfaz: I²C (direcciones: 0x23, 0x5C)
- Modos de medida: alta resolución, baja resolución, continuo, única
- Precio: ~2 €

Pines de conexión

Pin	Descripción
VIN	Alimentación (3,3 o 5 V)
GND	Tierra
SCL	SCL (A5 en Arduino UNO, 22 en ESP32)
SDA	SDA (A4 en Arduino UNO, 21 en ESP32)
ADDR	Dejar libre (define dirección I²C)



Usos en experimentos de física

Medida de la intensidad luminosa en función de la distancia (ley del cuadrado inverso)
Verificación de la ley de Beer-Lambert en disoluciones coloreadas
Estudio de la transmitancia y absorbancia de materiales
Medida de la luz solar a lo largo del día
Estudio de la polarización de la luz

4.2 TSL2591

Principio de funcionamiento

El TSL2591 es un sensor de luz digital de alta sensibilidad que combina dos fotodiodos: uno sensible a luz visibles e infrarroja (canal full), y otro sensible solo a infrarrojo (canal IR). Un conversor ADC de 16 bits digitaliza ambas señales. El microcontrolador puede restar el componente IR del canal full para obtener una medida de luz visible pura, lo que proporciona una precisión muy superior a la de sensores más simples.

Características técnicas

- Rango: 188 µlux a 88 000 lux
- Resolución ADC: 16 bits
- Rangos de ganancia: 1x, 25x, 428x (programables)
- Tiempo de integración: 100-600 ms (programable)
- Interfaz: I²C (dirección 0x29)
- Precio: ~3 €



Pines de conexión

Pin	Descripción
VIN	Alimentación (3,3 o 5 V)
GND	Tierra
SCL	SCL (A5 en Arduino UNO, 22 en ESP32)
SDA	SDA (A4 en Arduino UNO, 21 en ESP32)
3VO	Salida de 3,3 V (suministra hasta 100 mA)
INT	Pin de interrupción opcional

Usos en experimentos de física

Medidas de iluminancia en condiciones de muy baja luz (atenuación extrema)
Determinación precisa de la ley de Beer-Lambert
Estudio de filtros ópticos
Medida de la transmitancia de materiales semitransparentes
Medida de la luz solar

Comparativa BH1750 vs TSL2591

En la práctica, se observa que el BH1750 tiende a dar lecturas entre un 20-30 % más altas que el TSL2591 bajo la misma fuente de luz. Esto no implica que uno de ellos funcione incorrectamente ya que el TSL2591 utiliza un filtro más sofisticado que resta el componente infrarrojo, mientras que el BH1750 está optimizado para emular la respuesta del ojo humano.

Para experimentos de física donde se requiera alta precisión en un rango dinámico amplio (desde 188 µlux hasta > 80000 lux), el TSL2591 es más adecuado. Para medidas generales de iluminancia en el rango 1-65535 lux, el BH1750 ofrece una excelente relación calidad-precio y facilidad de uso.

4.3 Fotoresistencia (LDR)

Principio de funcionamiento

Una fotorresistencia o LDR (Light Dependent Resistor) es un componente optoelectrónico cuya resistencia eléctrica disminuye al aumentar la intensidad de luz incidente. Está fabricada con sulfuro de cadmio (CdS), un material semiconductor que presenta fotoconductividad: los fotones incidentes proporcionan energía a los electrones para saltar de la banda de valencia a la banda de conducción, aumentando la conductividad del material.

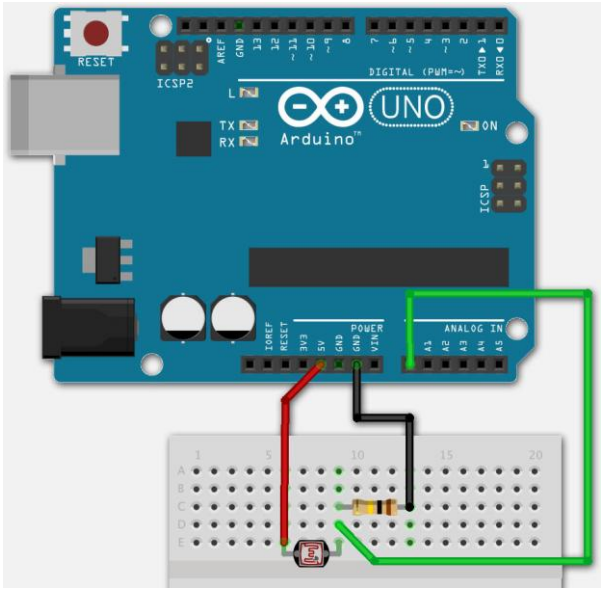


Características técnicas

- Resistencia en oscuridad: ~10 MΩ
- Resistencia con luz brillante: ~5 kΩ
- Tiempo de respuesta: ~20 ms
- Precio: ~0,05 € (20 unidades por 1 €)

Conexiones

No necesita librería para funcionar: se lee como un divisor de tensión con una resistencia fija de 10 kΩ conectada a una entrada analógica. La salida no es lineal y no proporciona valores en lux sin calibración previa.



Usos en experimentos de física

Detector de luz de bajo coste

Demostración del efecto fotoeléctrico interno (fotoconductividad)

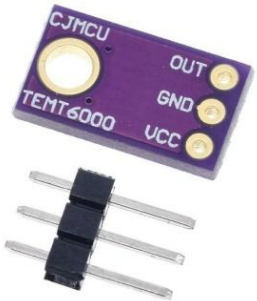
Medida cualitativa de la variación de intensidad luminosa

Estudio de la respuesta temporal de materiales semiconductores a la luz

4.4 Fototransistor

Principio de funcionamiento

Un fototransistor es un transistor bipolar cuya base se excita mediante fotones en lugar de corriente eléctrica. La luz incidente genera pares electrón-hueco en la región de la base, que son amplificados por la ganancia del transistor, produciendo una corriente de colector proporcional a la intensidad luminosa.



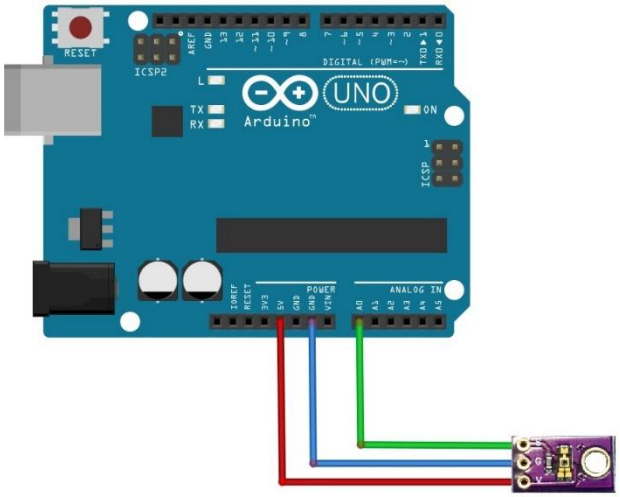
El modelo TEMT6000 integra el fototransistor con una resistencia de carga, proporcionando una salida analógica directamente conectables al microcontrolador.

Características técnicas (TEMT6000)

- Tensión de alimentación: 3,3-5 V
- Salida: analógica (proporcional a la luz)
- Tiempo de respuesta: ~15 µs (mucho más rápido que una LDR)
- Longitud de onda pico: 570 nm (verde-amarillo)
- Precio: ~1 €

Pines de conexión

Pin	Descripción
V	Alimentación (3,3 o 5 V)
G	GND, Tierra
S	Pin analógico, A0



Usos en experimentos de física

Barreras de luz rápidas (fotopuertas) para medida de tiempos en cinemática

Detección de pulsos luminosos de corta duración

Medida de la frecuencia de parpadeo (flicker) de fluorescentes y LEDs.

Lectura de códigos de barras y señales ópticas

4.5 Sensor de color TCS34725

Principio de funcionamiento

El TCS34725 es un sensor de color digital fabricado por AMS (Austria Mikro Systeme) que integra en un encapsulado compacto un array de 3×4 fotodiodos con filtros espectrales de paso de banda para los canales rojo (R), verde (G) y azul (B), así como fotodiodos claros (C, sin filtro) para medir la intensidad lumínica total. Cada fotodiodo convierte la luz incidente en una corriente eléctrica proporcional a la intensidad recibida en su banda espectral.



El integrado dispone de dos convertidores analógico-digitales (ADC) de 16 bits que digitalizan simultáneamente las lecturas de los canales de color y el canal claro. Un filtro de bloqueo de infrarrojos (IR-cut) integrado en el propio chip minimiza la interferencia de la radiación IR contenida en la luz natural o artificial, mejorando drásticamente la precisión de la medición del color real del objeto.

La comunicación con el microcontrolador se realiza a través de un bus I²C estándar (dirección por defecto: 0x29), lo que simplifica el cableado y permite integrar el sensor en redes de múltiples dispositivos I²C.

Analogía física: El TCS34725 funciona como un «ojo electrónico» que separa la luz blanca en sus tres componentes fundamentales (RGB), de forma análoga a cómo los tres tipos de conos de la retina humana permiten percibir el color. El filtro IR-cut desempeña la función del pigmento macular del ojo, que absorbe selectivamente la luz azul-violeta y reduce la dispersión cromática.

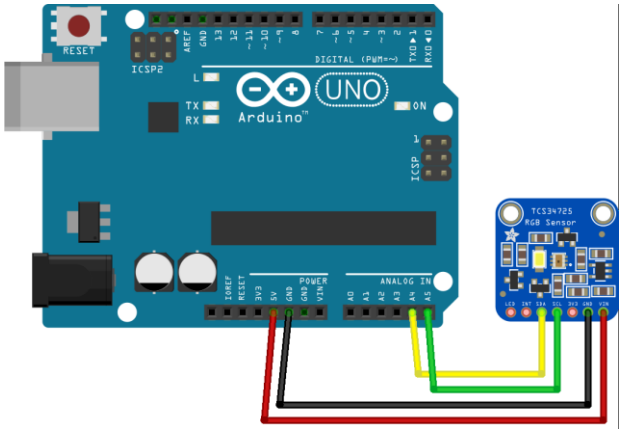
Características técnicas TCS34725 (AMS)

Parámetro	Valor
Tensión de alimentación	2,7 – 3,6 V (el módulo incorpora regulador a 3,3 V para 5 V)
Interfaz	I²C (dirección 0x29)
Resolución ADC	16 bits por canal (R, G, B, C)
Rango dinámico	3.800.000:1
Filtro IR integrado	Sí (bloqueo óptico en el chip)
Frecuencia de muestreo	Configurable hasta ~700 Hz
Rango espectral	~270 – 1000 nm (con picos ~470, ~525, ~640 nm RGB)
Tiempo de integración	Configurable de 2,4 ms a 700 ms
Ganancia	Programable (1×, 4×, 16×, 60×)
Corriente en reposo	65 µA (modo inactivo)
Dimensiones encapsulado	2,0 × 2,4 × 1,0 mm (DFN-8)
Precio módulo	~ 4 € (GY-33/TCS34725)

Conexión

El módulo más común basado en el TCS34725 (GY-33 o similar) dispone de cuatro pines, además de dos pines opcionales para el LED iluminador blanco integrado:

Pin del módulo	Conexión Arduino UNO	Conexión ESP32 (Wemos D1 R32)
VIN	5 V	5 V o 3,3 V
GND	GND	GND
SCL	A5	D22 (GPIO 22)
SDA	A4	D21 (GPIO 21)
LED (opcional)	3,3 V a través de resistencia 220 Ω	3,3 V a través de resistencia 220 Ω
INT (opcional)	Cualquier pin digital	Cualquier pin digital



El LED blanco integrado en algunos módulos (como el GY-33) permite iluminar la superficie del objeto en condiciones de baja luz ambiental, garantizando lecturas consistentes. Se activa mediante una salida digital del microcontrolador.

Usos en experimentos de física

Es un sensor muy polivalente que puede servir, por ejemplo, para:

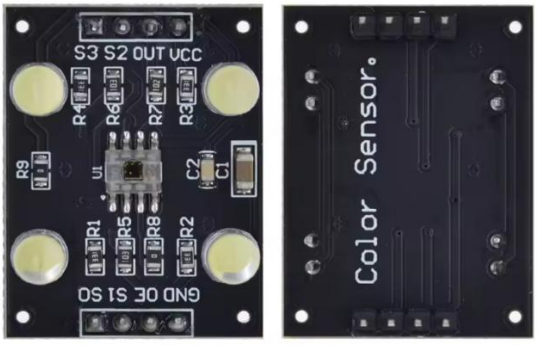
- Espectroscopia básica: análisis de la composición espectral de fuentes de luz LED, incandescente, fluorescente o solar midiendo las componentes RGB relativas.
- Determinación de concentraciones (Ley de Beer-Lambert): midiendo la atenuación de luz que atraviesa una disolución coloreada de concentración conocida; la absorbancia es proporcional a la concentración.
- Estudio de la reflectancia de superficies: caracterización del color de pigmentos, tejidos y papeles en función del ángulo de incidencia.
- Estudio de la temperatura de color: la relación entre los canales R, G y B permite estimar la temperatura de color de una fuente luminosa (en kelvin).
- Cinemática y medida de intervalo: usando una barrera óptica con el canal claro (C) para detectar cambios bruscos de iluminación —objetos que cruzan un haz de luz— midiendo tiempos con precisión (hasta ~2,4 ms con integración mínima).
- Fotometría básica: cuantificación de la intensidad lumínica relativa mediante el canal claro (C) en experimentos de inversa del cuadrado de la distancia.
- Monitoreo de decaimiento de fósforos y fluorescencia: seguimiento temporal de la emisión tras excitación luminosa.
- Estudio de polarización de la luz: midiendo la intensidad transmitida a través de filtros polarizadores en distintos ángulos.

Sensores alternativos al TCS34725

A continuación se describen otros sensores de color disponibles en el mercado que pueden cumplir funciones similares o complementarias al TCS34725:

TCS3200 (TAOS / AMS)

- Principio: Matriz de 8×8 fotodiodos con filtros RGB y claros; la salida es una onda cuadrada de frecuencia proporcional a la intensidad de cada canal, seleccionable mediante pines S0-S1 (escala de frecuencia) y S2-S3 (selección de filtro).



- Ventajas: bajo precio (~6 €); respuesta rápida; fácil de leer con cualquier microcontrolador sin necesidad de ADC ni I²C.
- Inconvenientes: Carece de filtro de bloqueo IR, por lo que la luz infrarroja contamina las lecturas; requiere tres pines digitales más dos de selección (5 pines en total); las lecturas deben convertirse de frecuencia a intensidad con temporizadores internos; menor precisión que el TCS34725.

- Recomendado para: Experimentos cualitativos, presupuestos muy ajustados, o cuando se desee conectar a microcontroladores sin interfaz I²C.

GY-31 (TCS3200 en módulo con LEDs)

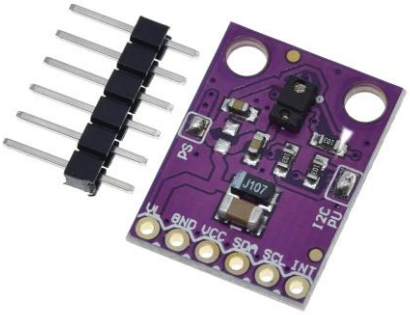
- Principio: Esencialmente un TCS3200 montado en una placa con 4 LEDs blancos de iluminación integrados. La lectura es por frecuencia, al igual que el TCS3200.



- Ventajas: Incluye iluminación propia, lo que estabiliza las condiciones de medida; precio bajo (~2 – 4 €).
- Inconvenientes: Mismas limitaciones que el TCS3200 (sin filtro IR, conectividad por frecuencia). Los LEDs pueden calentar la muestra y alterar la medida si no se gestiona correctamente el tiempo de encendido.
- Recomendado para: Experimentos de reflectancia en superficies opacas con iluminación controlada, clasificación de objetos por color en entornos educativos.

APDS-9960 (Broadcom / Avago)

- Principio: Sensor multispectral RGB y detección de gestos por retroiluminación IR (4 fotodiodos direccionales). Incluye medición de luz ambiental (ALS) y proximidad.



- Ventajas: Integra detección de gestos, proximidad y color en un solo encapsulado (4 en 1); interfaz I²C y muy compacto.
- Inconvenientes: Rango de color menos preciso que el TCS34725. El filtro IR no es tan eficaz para aplicaciones colorimétricas puras; precio bajo (~2 €). La funcionalidad de gestos requiere procesamiento adicional.
- Recomendado para: Proyectos que necesiten combinar detección de gestos o proximidad con medición básica de color (interacción humano-máquina, robótica educativa).

Cuadro resumen de sensores de color

Característica	TCS34725	TCS3200	APDS-9960	GY-31
Tipo de salida	I²C digital	Frecuencia (onda cuadrada)	I²C digital	Frecuencia (onda cuadrada)
Resolución	16 bits por canal	Depende del temporizador	16 bits (ALS)	Depende del temporizador
Filtro IR integrado	Sí	No	Parcial	No
Iluminación integrada	Opcional (LED externo)	No	No	Sí (4 LEDs blancos)
Canales	R, G, B, C (claro)	R, G, B, C (seleccionables)	R, G, B, ALS, Gestos	R, G, B, C (seleccionables)
Velocidad máxima	~700 Hz	kHz (ondas cuadradas)	~100 Hz	kHz (ondas cuadradas)
Alimentación	2,7 – 3,6 V (regulable a 5 V)	2,7 – 5,5 V	2,4 – 3,6 V	2,7 – 5,5 V
Precio orientativo	~3– 4 €	~1,5 – 2,5 €	~2 – 3 €	~2 – 4 €
Dificultad de uso	Baja (I²C, librería)	Media (medición de frecuencia)	Baja (I²C, librería)	Media (medición de frecuencia)
Precisión colorimétrica	Alta	Media-Baja	Media	Media-Baja
Funciones adicionales	—	—	Gestos, proximidad	LEDs iluminadores

5 Sensores de proximidad por infrarrojos

Los sensores de proximidad por infrarrojos detectan la presencia de objetos cercanos sin contacto físico, mediante la emisión de un haz de luz IR y la detección de su reflexión. A diferencia de los sensores de distancia (HC-SR04, VL53L0X), estos dispositivos solo proporcionan una señal digital de presencia/ausencia. Utilizan luz infrarroja en el entorno de 940 nm, invisible al ojo humano.

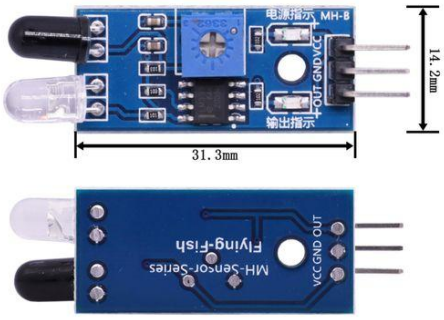
Son sensores (junto con las barreras de luz) imprescindibles para el estudio de los movimientos

5.1 Módulo Flying-fish

Principio de funcionamiento

Los módulos de la serie **MH-Sensor** son sensores de obstáculos IR de diseño más sencillo y económico. El modelo típico (MH-B, por ejemplo) utiliza un LED emisor IR y un fototransistor receptor montados en el mismo plano, orientados hacia delante.

El LED IR emite un haz de luz que, al incidir sobre un objeto cercano, se refleja difusamente y una fracción de esa luz es captada por el fototransistor. La señal del fototransistor se compara con un umbral ajustable mediante un comparador LM393, que proporciona una salida digital: nivel LOW cuando detecta un obstáculo y HIGH en reposo. Un potenciómetro de ajuste permite fijar la distancia de detección.

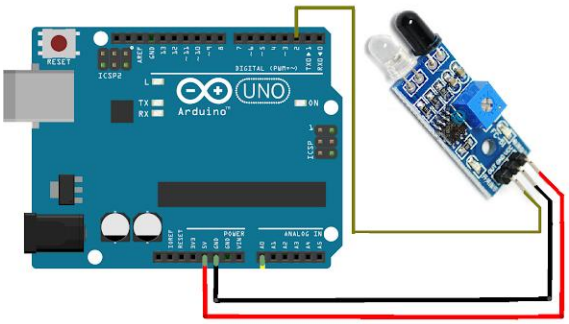


Características técnicas

- Tensión de alimentación: 3,3-5 V
- Distancia ajustable: 2-30 cm (mediante potenciómetro)
- Salida: digital
- Indicador LED de detección (se enciende cuando detecta obstáculo)
- Dimensiones: ~30 × 20 mm
- Precio: ~1 €

Pines de conexión

Pin	Descripción
VCC	Alimentación (3,3-5 V)
GND	Tierra
OUT	Salida digital (LOW cuando detecta obstáculo, HIGH en reposo)



Usos en experimentos de física

Detección de paso de objetos o personas en cinemática
Contador de objetos (cuentarrevoluciones)
Puerta de inicio/parada para cronometraje de experimentos
Sistemas de barrera óptica simple
Medición del tiempo de caída de un cuerpo
Estudio del movimiento uniforme y uniformemente acelerado
Cronometraje de precisión en experimentos de péndulo

5.2 Módulo KY-032

Principio de funcionamiento

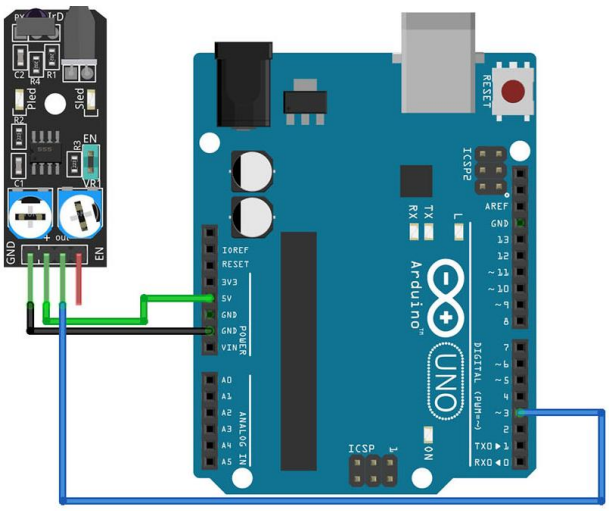
El KY-032 es un módulo de detección de obstáculos por infrarrojos de la familia de sensores KY (Keyes). Su principio es similar al Flying-fish: un LED emisor IR (938 nm) y un fototransistor receptor dispuestos frontalmente. Incorpora también un comparador LM393 para digitalizar la señal, pero añade un oscilador interno (a ~40 kHz) que modula el haz IR emitido.



La modulación permite al receptor distinguir la señal emitida de la luz IR ambiental (como la del sol o bombillas incandescentes), mejorando la inmunidad al ruido óptico. Al igual que el Flying-fish, dispone de un potenciómetro para ajustar la distancia de detección y un LED indicador.

Características técnicas

- Tensión de alimentación: 3,3-5 V
- Distancia ajustable: 2-40 cm (mediante potenciómetro)
- Salida: digital (LOW cuando detecta obstáculo, HIGH en reposo)
- Frecuencia de modulación: ~40 kHz (reduce interferencias ambientales)
- Indicador LED de detección
- Dimensiones: ~40 × 15 mm
- Precio: ~1,5 €



Pines de conexión

Pin	Descripción
VCC	Alimentación (3,3-5 V)
GND	Tierra
OUT	Salida digital (LOW al detectar obstáculo)
EN	Enable (activo bajo; dejar a GND o sin conectar para funcionamiento normal)

Usos en experimentos de física

Detección de paso en condiciones de alta luminosidad ambiental
Barreras de luz en exteriores o cerca de ventanas
Experimentos de robótica educativa
Sistemas de conteo resistentes a falsos positivos por luz ambiente

Comparativa de sensores de proximidad IR

Caracterís-tica	Flying-fish MH-Sensor	KY-032
Tipo de salida	Digital (LM393)	Digital (LM393 + modulación)
Distancia ajustable	2-30 cm	2-40 cm
Inmunidad a luz ambiente	Baja	Alta (modulación ~40 kHz)
Flexibilidad de umbral	Fija por potenciómetro	Fija por potenciómetro
Precio	~1 €	~1,5 €

Calibración de los sensores de proximidad IR

La calibración de estos sensores puede ser necesaria para adaptarlos a las condiciones específicas de cada experimento.

Para ajustar del potenciómetro se ha de girar lentamente con un destornillador de precisión mientras se coloca un objeto a la distancia deseada. Girar en sentido horario aumenta la distancia de detección; en sentido antihorario la reduce.

El procedimiento consiste en colocar un objeto blanco (cartulina) a la distancia a la que se desea disparar la detección. Después se ha de girar el potenciómetro hasta que el LED indicador se encienda (o cambie de estado). Al retirar el objeto el LED debe apagarse. Repetir la operación para verificar la distancia.

Se puede verificar por el puerto serie leyendo el pin digital y comprobando que el estado cambia al aproximar/alejar el objeto. Si no hay cambio, ajustar de nuevo el potenciómetro.

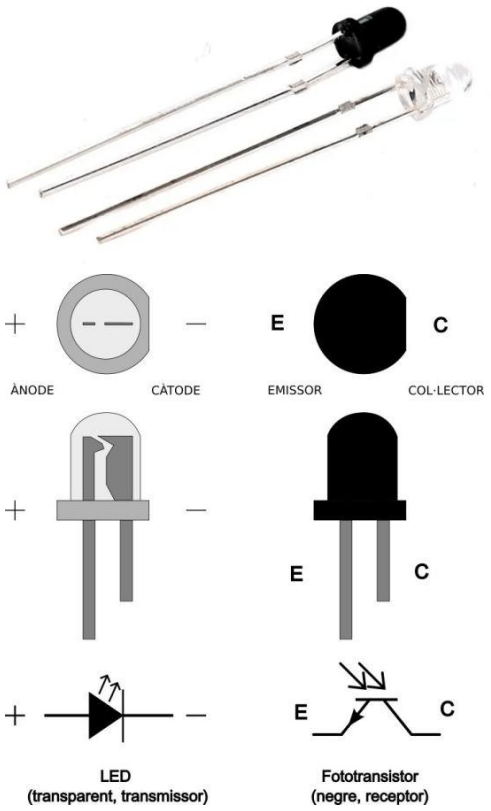
6. Barreras de luz

Las barreras de luz (junto con los sensores de proximidad) son sensores imprescindibles para el estudio de los movimientos.

6.1 Fotopuertas

Principio de funcionamiento

Una barrera de luz o fotopuerta está constituida por un emisor de luz (generalmente un LED infrarrojo) y un receptor (fototransistor, fotodiodo o fotoresistencia) enfrentados. Cuando un objeto interrumpe el haz luminoso, el receptor detecta la caída de intensidad y genera un cambio de señal que el microcontrolador interpreta como un evento. Es el principio de las puertas ópticas utilizadas en laboratorios de física para medir intervalos de tiempo con alta precisión.



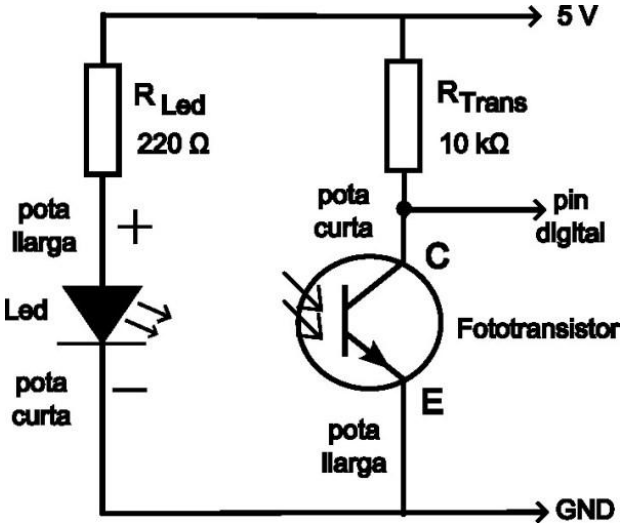
Aquí utilizaremos la pareja LED (transparente) infrarrojo (940 nm) y fototransistor IR (color negro) de dos patas (colector y emisor), por su rapidez de respuesta y bajo coste.

El fototransistor es un transistor y los hay de tres patas, es decir, también con la pata base, pero aquí no hace falta ya que la base está en el capuchón y controlada por la luz que recibe.

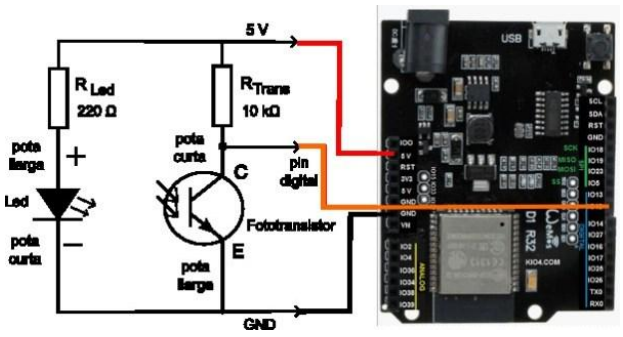
Montaje

El montaje requiere:

- LED IR (transparente) + resistencia limitadora de 220 Ω
- Fototransistor IR (negro, encapsulado de dos patas) + resistencia de 10 kΩ (pull-down)
- Conectar el ánodo del LED (con resistencia) a 5 V y el cátodo a GND
- Conectar el colector del fototransistor a VCC (5 V) y el emisor a tierra a través de la resistencia de 10 kΩ; el punto de unión emisor-resistencia se conecta a un pin digital del microcontrolador



Para indicar al microprocesador si el fototransistor está iluminado, o no, se debe conectar la unión entre el fototransistor y su resistencia al pin que se indique en el código de cada proyecto; en el caso de la figura de abajo en el pin digital 12 del ESP32 (Wemos D1 R32).

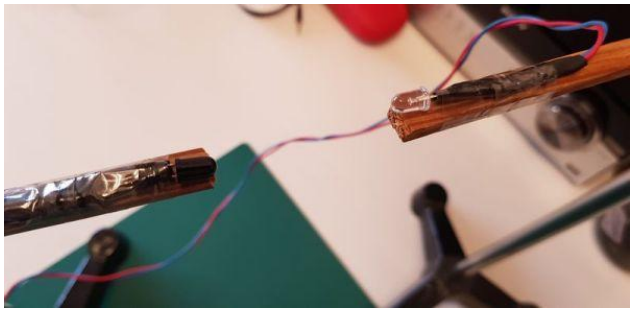


Los pines a los que se ha de conectar el colector del fototransistor han de ser digitales, ya que lo que aquí nos interesa no es medir el voltaje de esta pata para

saber si el transistor está más o menos iluminado, sino si en el colector hay voltaje (HIGH), o no (LOW).

Se deben colocar los dos elementos enfrentados 2 – 5 cm de manera que la luz del led llegue directamente al fototransistor ya que su ángulo de visión es de unos 30º, y se debe dejar un espacio entre ellos para que puedan pasar los objetos de los que estudiamos el movimiento.

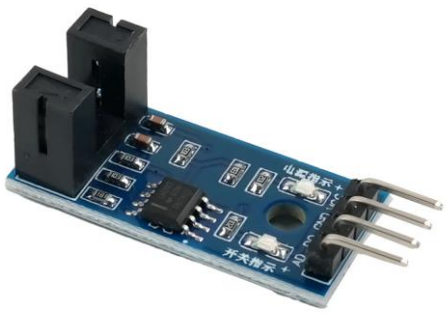
Al interrumpir el móvil el haz, el fototransistor deja de conducir y la entrada digital del microcontrolador pasa a LOW.



Usos en experimentos de física
Medición del tiempo de caída de un cuerpo
Determinación de la velocidad instantánea en un plano inclinado
Cronometraje de precisión en experimentos de péndulo
Medida de la velocidad de proyectiles
Estudio del movimiento uniforme y uniformemente acelerado

6.2 Optoacoplador IR de ranura

El optoacoplador de ranura, también conocido como sensor de velocidad FC-03 o HC-020K, es un sensor óptico que integra en un solo encapsulado en forma de «U» un emisor LED infrarrojo y un fototransistor receptor enfrentados.



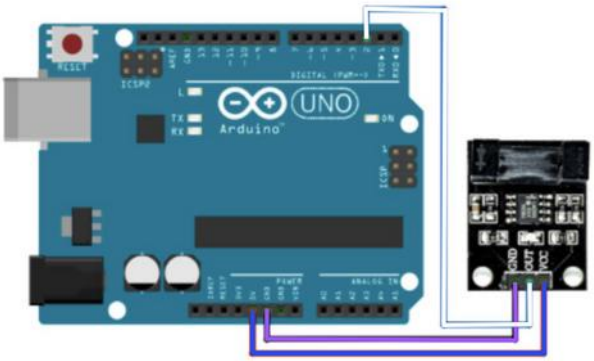
Este sensor, por su salida digital y su bajo coste, es muy interesante para experimentos de dinámica rotacional en el laboratorio de física.

Principio de funcionamiento
El haz infrarrojo emitido por el LED (940 nm) atraviesa la ranura libremente y es detectado por el fototransistor. Al introducir un obstáculo en la ranura, el haz se interrumpe y el fototransistor deja de conducir. El comparador LM393 convierte esta transición en una señal digital nítida, eliminando el ruido y el rebote mecánico.

Características técnicas
- Tensión de alimentación: 3,3-5 Vcc
- Salida digital (DO): LOW cuando hay obstrucción, HIGH cuando el haz está despejado (configurable según módulo)
- Salida analógica (AO): opcional, proporcional a la cantidad de luz recibida
- Comparador: LM393 integrado (señal sin rebotes)
- Ancho de ranura típico: 5 mm (admite discos ranurados de hasta ~3 mm de espesor)
- Dimensiones del módulo: ~32 × 13 mm
- Frecuencia máxima de conmutación: >1 kHz (suficiente para miles de RPM)
- Indicador LED de obstrucción en la propia placa
- Precio: ~1 €

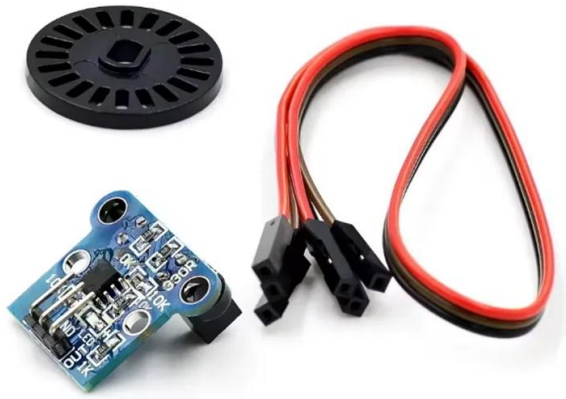
Pines de conexión

Pin	Descripción
VCC	Alimentación (3,3-5 V)
GND	Tierra
DO	Salida digital
AO	Salida analógica (opcional, no presente en todos los módulos)



Usos en experimentos de física
Contador de objetos pequeños.

Medida de frecuencia de oscilación: un péndulo o varilla que atraviesa la ranura permite medir su periodo de oscilación.
Medida de velocidad de rotación
La aplicación más potente de este sensor es la medida de velocidad de rotación mediante un disco ranurado (encoder) acoplado al eje de un motor o mecanismo rotatorio.



Montaje experimental
1. Acoplar un disco ranurado (con N ranuras equiespaciadas, típicamente 20) al eje de rotación.
2. Colocar el sensor de ranura de modo que el disco pase libremente por la ranura.
3. Conectar DO a un pin digital del microcontrolador.
4. Configurar una interrupción en ese pin para contar cada flanco descendente (o ascendente).

7. Sensor de pH

Principio de funcionamiento
Un electrodo de pH mide la actividad de los iones hidrógeno (H^+) en una disolución. El electrodo de vidrio genera un potencial eléctrico proporcional al pH según la ecuación de Nernst:

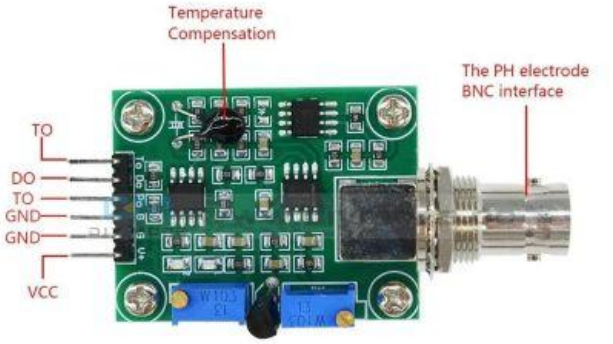
$$E = E_0 + (RT/nF) \cdot ln[H^+]$$

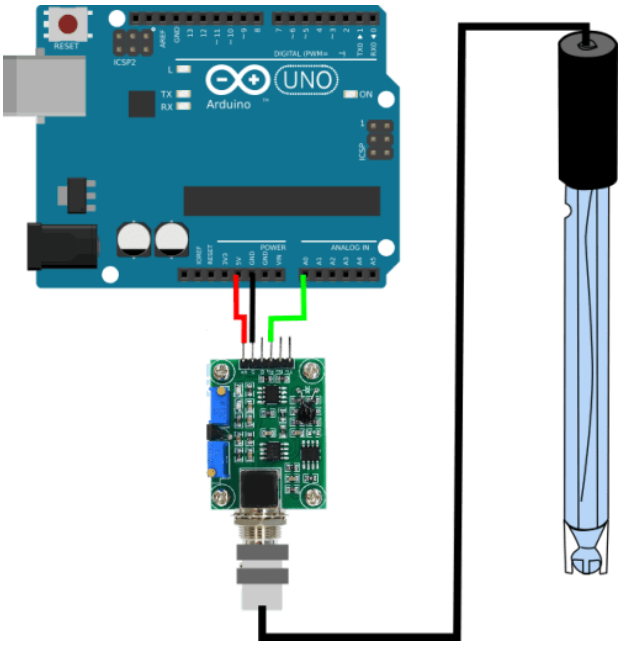
donde R es la constante de los gases, T la temperatura, n la carga del ion (n=1 para H^+) y F la constante de Faraday. A 25 °C, el potencial cambia aproximadamente 59 mV por unidad de pH.

El kit de pH para Arduino consta de una sonda de pH con conector BNC y una placa electrónica (PH-4502C) que acondiciona la señal —amplifica y filtra— para que el microcontrolador pueda leerla a través de una entrada analógica.

Características de la placa PH-4502C
- Voltaje de trabajo: 5 V
- Rango de pH: 0-14
- Tiempo de estabilidad: ≤ 60 s (necesita calentarse para dar valores estables)
- Temperatura de trabajo: -10 a 50 ºC (temperatura nominal de 20 ºC)
- Salida: analógica (Po)
- Precio del kit: desde 9 €

Características de la sonda
- Rango de pH: 0-14
- Punto cero: 7 ± 0,5 pH
- Temperatura de funcionamiento: 0 – 60 ºC
- Tiempo de respuesta: ≦ 1min
- Vida media: 1,5-2 años
- Conector BNC adecuado para la mayoría de controladores de pH





Pines de conexión

Pin	Descripción
V+	Alimentación (5 V)
G	Masa del circuito analógico
G	Massa de la alimentación
To	Temperatura
Do	Señal del límite de pH
Po	Valor de pH en V (La salida es analógica)

Advertencia importante

El pH-metro funciona correctamente con Arduino UNO (entradas analógicas puras), pero **NO funciona bien con placas ESP32**, ya que sus entradas analógicas no son auténticas (su ADC es no lineal). se recomienda usar Arduino UNO, añadiendo un módulo Bluetooth HC-06 si se necesita conexión inalámbrica.

Usos en experimentos de física y química

Medición del pH de disoluciones ácidas y básicas
Estudio de la variación del pH en reacciones de neutralización
Determinación de constantes de acidez (Ka) por valoración potenciométrica
Seguimiento del pH en reacciones químicas
Monitorización de la calidad del agua

8. Sensor de fuerza/peso (célula de carga + HX711)

Principio de funcionamiento

Una célula de carga es un transductor que convierte una fuerza mecánica (peso) en una señal eléctrica. Internamente contiene galgas extensométricas (resistencias metálicas que cambian su valor al ser deformadas) dispuestas en configuración de puente de Wheatstone.

Cuando se aplica una fuerza sobre la célula, las galgas se deforman ligeramente, variando su resistencia de forma proporcional a la fuerza aplicada. El puente de Wheatstone convierte esta pequeña variación de resistencia en una diferencia de voltaje del orden de milivoltios.

Para poder leer esta señal tan débil con un microcontrolador, es necesario amplificarla. Aquí entra el **HX711**, un ADC de 24 bits específicamente diseñado para células de carga.



Células de carga

Hay células de carga de todos los valores, pero para el laboratorio la más interesante es la de 1 kg para experimentos con masas pequeñas (pesitas, objetos ligeros) ya que tiene el rango más versátil para la mayoría de experimentos.
Precio orientativo de la célula: ~2 €.

Amplificador HX711

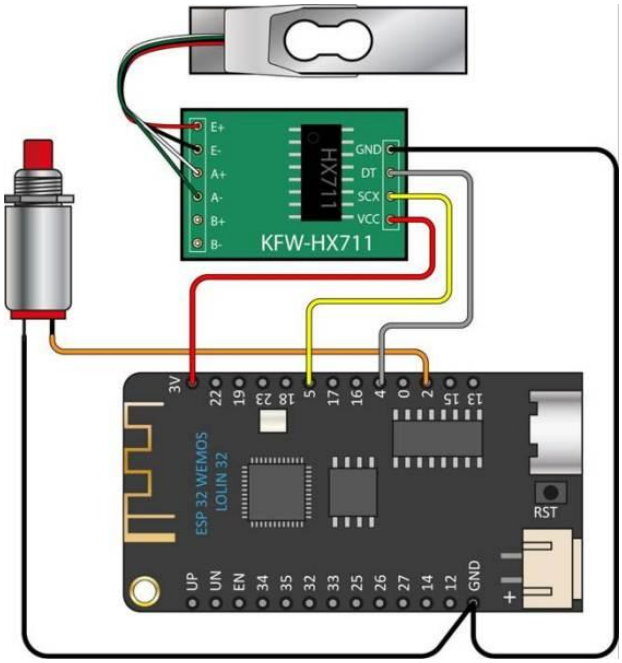
El HX711 es un conversor analógico-digital de 24 bits con un amplificador de instrumentación integrado, diseñado para leer células de carga y balanzas, que permite detectar variaciones de fuerza del orden de 0,1 g en una célula de 1 kg.

Características técnicas

- Resolución: 24 bits
- Canales: 2 (A y B)
- Ganancia seleccionable: 128 o 64 (canal A), 32 (canal B)
- Interfaz: digital serie (2 pines: DT y SCK)
- Voltaje de alimentación: 2,7-5,5 V
- Precio del módulo: ~1 €

Pines de conexión

Pin HX711	Conexión
VCC	5 V (o 3,3 V)
GND	Tierra
DT (DOUT)	Pin digital del microcontrolador
SCK (PD_SCK)	Pin digital del microcontrolador



Conexión entre célula de carga y HX711
La célula de carga tiene cuatro cables que se conectan directamente al módulo HX711:

Cable	Función	Conexión HX711
Rojo	Excitación+ (E+)	E+
Negro	Excitación- (E-)	E-
Blanco	Señal+ (S+)	A+
Verde	Señal- (S-)	A-

Para Arduino/ESP32 se utiliza la librería HX711 Arduino Library de Bogdan Necula, disponible en el Gestor de Librerías del IDE de Arduino.

Usos en experimentos de física
Medida de masas y fuerzas (peso)
Verificación de la ley de Hooke (fuerza-elongación en un muelle)
Estudio de las oscilaciones de un muelle
Medida de la tensión superficial de líquidos
Determinación de la densidad de sólidos y líquidos por el principio de Arquímedes

9. Sensores magnéticos

Los hay de tres tipos: Digitales (A3144), que indican la existencia o no de campo magnético, analógicos (SS49E), que permiten medir variaciones del campo magnético y digitales con interfaz I2C (HMC5883L), que permiten hacer medidas cuantitativas absolutas del campo.

La alternativa de usar el sensor magnético del que disponen la mayoría de los teléfonos móviles muchas veces es una opción igual de buena y mucho más cómoda..

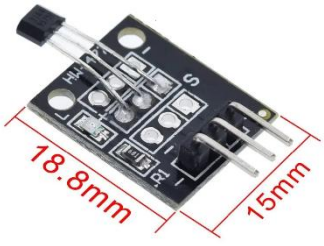
9.1 Sensor de efecto Hall A3144 (KY-003)

Principio de funcionamiento
El sensor de efecto Hall A3144, como su nombre indica, explota el efecto Hall que consiste en que cuando una corriente eléctrica circula por un material semiconductor y se aplica un campo magnético perpendicular a la dirección de la corriente, los portadores de carga (electrones o huecos) son desviados por la fuerza de Lorentz, generando una diferencia de potencial transversal (tensión Hall).

El A3144 integra un elemento Hall que genera una pequeña tensión proporcional al campo magnético, y un comparador Schmitt trigger que convierte esa tensión en una señal digital (LOW cuando detecta un campo magnético, HIGH en reposo).

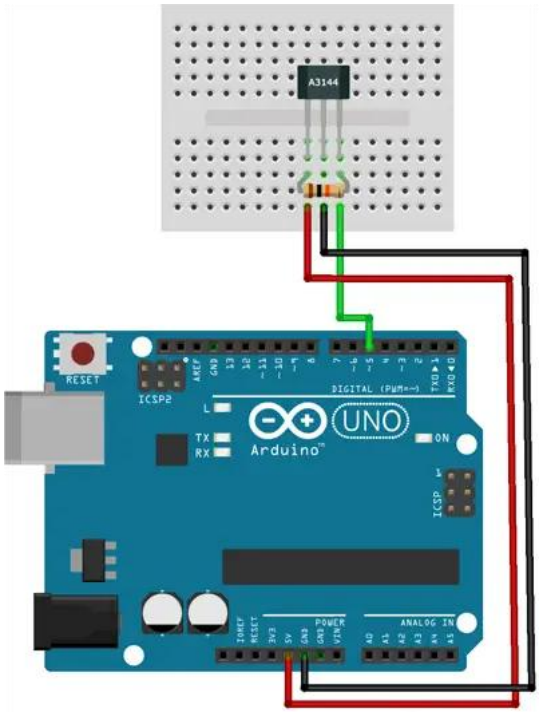


Características técnicas
- Alimentación: 4,5-24 V
- Salida: digital (LOW cuando detecta campo, HIGH en reposo)
- Frecuencia de conmutación: hasta 100 kHz
- Tipo: unipolar (responde solo a un polo magnético)
- Precio: ~1 € (módulo KY-003)



Pines de conexión

Pin	Descripción
Vin (izquierdo)	Alimentación (5 V)
GND (centro)	Tierra (0 V)
S (derecho)	Pin digital, D5
Requiere una resistencia pull-up de 10 kΩ entre los cables de voltaje y de datos.	



Usos en experimentos de física

Detección de paso de un imán (cuentarrevoluciones, medición de velocidad angular)
Interruptor magnético de proximidad
Experimentos de inducción magnética
Medición de la frecuencia de rotación de un motor o disco

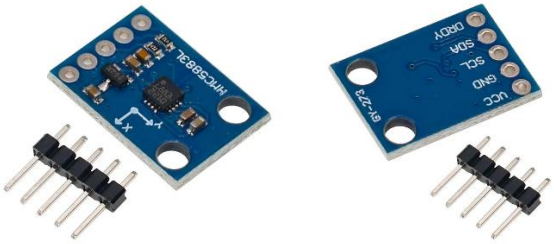
9.2 Magnetómetro HMC5883L / QMC5883L / QMC5883P

Principio de funcionamiento

El HMC5883L (y sus versiones mejoradas QMC5883L y QMC5883P) es un magnetómetro de tres ejes que mide la intensidad y dirección del campo magnético ambiental. Internamente utiliza sensores Magnetoresistivos (AMR, Anisotropic Magneto-Resistance) en forma de puentes de Wheatstone fabricados con materiales ferromagnéticos (permalloy) cuya resistencia varía con la orientación y la intensidad del campo magnético externo. Esto permite medir campos en los tres ejes espaciales (X, Y, Z) con alta sensibilidad. Se comunica mediante el bus I²C.

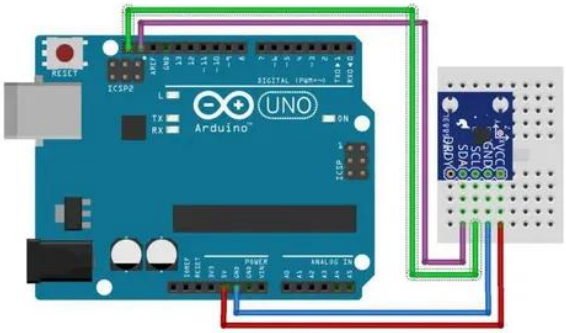
Características técnicas

- Rango: 800 µT (8 gauss) configurable
- Resolución: hasta 0,2 µT (2 miligauss)
- Interfaz: I²C (dirección 0x1E para HMC5883L, 0x0D para QMC5883L, 0x2C para QMC5883P)
- Alimentación: 2,16-3,6 V (los módulos con regulador aceptan 5 V)
- Precio: ~2 € el HMC5883L, ~6 € el QMC5883L o el QMC5883P



Pines de conexión

Pin	Descripción
VCC	Alimentación (3,3-5 V)
GND	Tierra
SCL	SCL, Reloj I²C
SDA	SDA, Datos I²C



Usos en experimentos de física

Medición del campo magnético terrestre (~25-65 µT según la latitud)
Determinación de la permeabilidad magnética de materiales
Estudio de la ley de Biot-Savart: campo magnético en función de la distancia a un conductor
Brújula digital (orientación magnética)
Mapeo de campos magnéticos en el laboratorio

9.3 Sensor de flujo magnético SS49E

Principio de funcionamiento

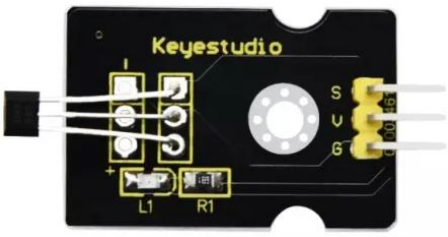
El SS49E es un sensor de efecto Hall lineal que proporciona una salida analógica proporcional a la intensidad del campo magnético (también podría ser el A1302KUA). A diferencia del A3144 (digital), el SS49E permite medir la magnitud del campo, no solo su presencia. Su funcionamiento se basa en un elemento Hall que genera una tensión diferencial,

amplificada por un amplificador operacional interno para proporcionar una salida lineal en el rango de alimentación.



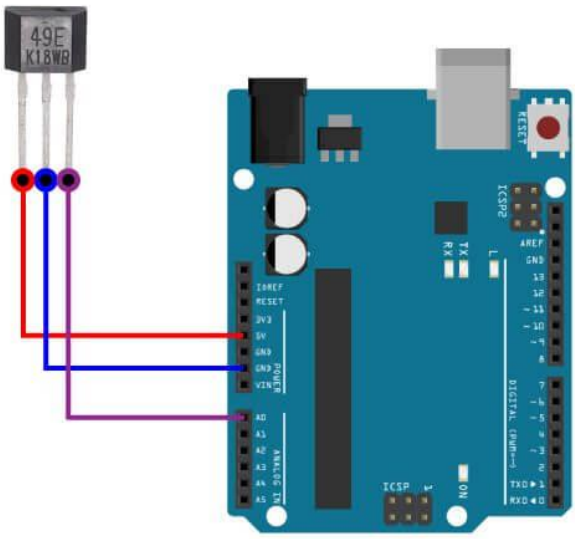
Características técnicas

- Alimentación: 2,7-6,5 V
- Salida: analógica (proporcional al campo)
- Sensibilidad: 14 mV/mT típica
- Rango: ±100 mT
- Salida en reposo (0 mT): VCC/2
- Precio: ~1 €



Pines de conexión

Pin	Descripción
V (izquierdo)	Alimentación (5 V)
G (centro)	Tierra (0 V)
S (derecho)	Pin analógico, A0



Usos en experimentos de física

Medida de la intensidad del campo magnético en función de la distancia a un imán

Verificación de la ley de Biot-Savart con espiras y solenoides

Determinación experimental de la permeabilidad magnética del vacío (μ_0)

Estudio de la fuerza magnética entre imanes

10. Conductividad y nivel de agua

10.1 Sensor de conductividad

Principio de funcionamiento

La conductividad eléctrica del agua mide su capacidad para transportar corriente eléctrica, que depende directamente de la concentración de iones disueltos (sales, minerales). Este principio se rige por la ley de Kohlrausch en la que la conductividad molar de un electrolito es la suma de las contribuciones individuales de cada ion.

El sensor de conductividad casero consiste en dos electrodos (de acero inoxidable o grafito) separados una distancia fija, sumergidos en la disolución problema. Al aplicar un voltaje conocido entre ellos, se mide la corriente que circula a través del agua. Mediante la ley de Ohm y la constante de celda (que depende de la geometría de los electrodos), se obtiene la conductividad en µS/cm o mS/cm.

Para evitar la electrólisis y la polarización de los electrodos, es recomendable usar corriente alterna de baja frecuencia en lugar de corriente continua. Una forma sencilla de implementarlo es alternar la polaridad de la medida con dos pines digitales del microcontrolador.



Se puede comprar un sensor de conductividad 51/STM32 para Arduino como el de la fotografía de

arriba por entre 40 y 70 € con las siguientes características:

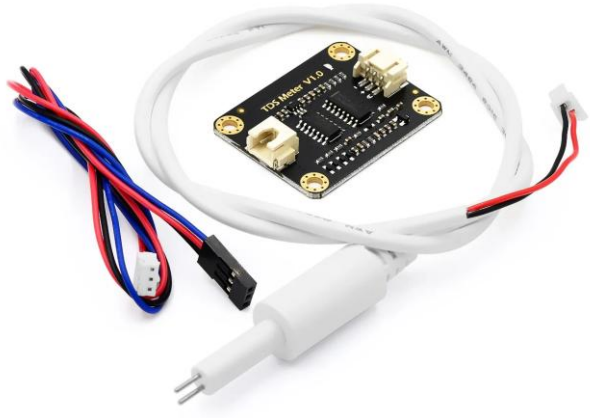
Características técnicas

- Tensión de alimentación: 5.0V
- Voltaje de salida de señal analógica: 0~3.4V
- Precisión de la medida: ±5% F.S.
- Interfaz de electrodo: BNC
- Interfaz del sensor de temperatura: XH 2.54
- Tamaño del tablero: 42mm*32mm

Para el electrodo de conductividad:

- Electrodo tipo de laboratorio, DJS-1
- Constante de la célula de conductividad: 1.0 (K=1)
- Rango de medida: 0 20mS/cm
- Rango de medida recomendado: 1~15mS/cm
- Temperatura de medida: 0-40°C
- Vida útil del electrodo: >0.5 años (la vida real está relacionada con el tipo y la frecuencia de uso)
- Longitud del cable: 100cm

Alternativamente, si no se necesita conocer exactamente la conductividad) se puede comprar un sensor TDS (Total Dissolved Solids) de conductividad del agua que mide cuántos miligramos de sólidos están disueltos en 1 litro de agua (ppm), que es mucho más barato. En términos generales, cuanto mayor sea el valor de TDS, más sustancias disueltas contenidas en el agua y menos limpias será. Por lo tanto, el valor de TDS se puede utilizar como una de las bases para reflejar la limpieza del agua.



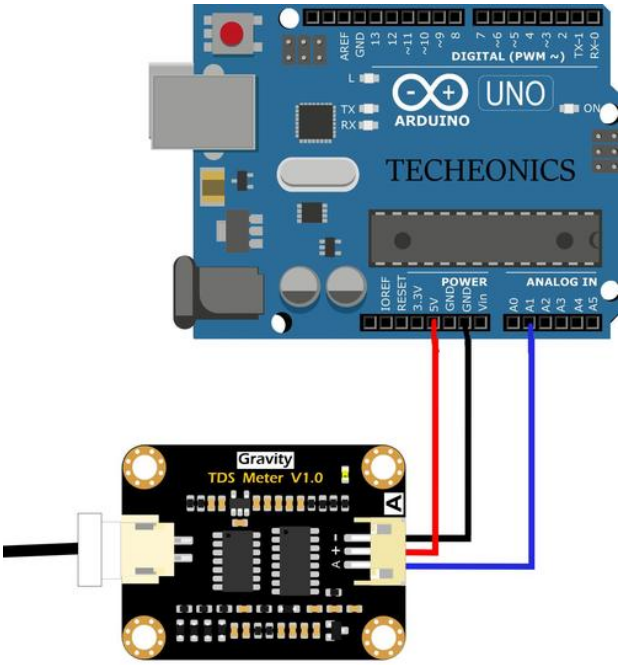
La fuente de excitación para la medida utiliza señales de CA, que pueden prevenir eficazmente la polarización de la sonda, prolongar su vida útil y aumentar la estabilidad de la señal de salida. La sonda TDS evidentemente es impermeable para que se pueda sumergir en agua.

Características técnicas

- Tensión de entrada: 3.3~5.5V
- Señal de salida: 0~2.3V
- Rango de medida TDS: 0~1000ppm
- Precisión de la medida TDS: ±10% F.S. (25°C)
- Tamaño: 42*32mm
- Interfaz del módulo: XH2.54-3P
- Interfaz de electrodo: XH2.54-2P
- Sonda TDS impermeable
- Precio: ~7 €

Pines de conexión

Pin	Descripción
+	Alimentación (3,5-5,5 V)
-	Tierra (0 V)
A	Pin analógico, A1



Usos en experimentos de física y química

Determinación de la concentración de sales en disolución

Estudio de la conductividad en función de la temperatura

Verificación de la ley de Kohlrausch

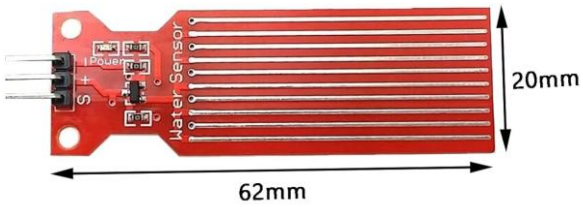
Medida de la mineralización del agua (agua destilada vs. agua del grifo)

Seguimiento de reacciones de precipitación por cambio de conductividad

10.2 Sensor de nivel de agua

El sensor de nivel de agua permite detectar si el agua ha alcanzado una determinada altura. Existen dos aproximaciones principales:

Sensor de electrodos (conductivo): dos electrodos colocados a la altura deseada. Cuando el agua alcanza ese nivel, cierra el circuito entre ambos electrodos, permitiendo el paso de una pequeña corriente que el microcontrolador detecta como señal HIGH en una entrada digital.

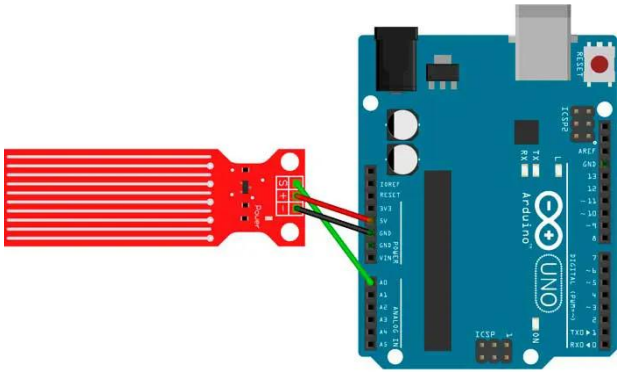


Características técnicas

- Voltaje de funcionamiento: 5V
- Corriente de funcionamiento: menos de 20mA
- Tipo de sensor: Analógico
- Área de detección :40mmx16mm
- Temperatura de trabajo :10-30 ºC
- Humedad de funcionamiento: 10%-90% sin condensación
- Precio: < 1 €

Pines de conexión

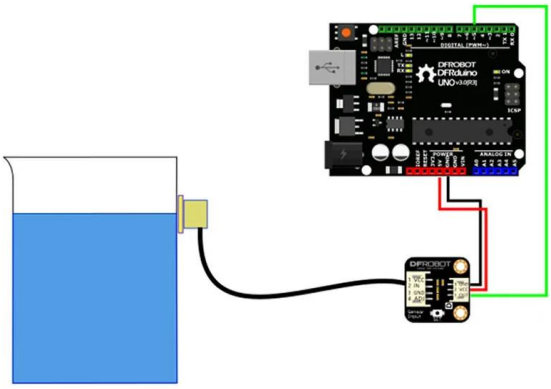
Pin	Descripción
+	Alimentación (3-5 V)
-	Tierra (0 V)
S	Pin analógico, A0



Sensor capacitivo: mide el cambio en la capacitancia entre dos placas (o entre una placa y la masa de agua) cuando el nivel del agua varía. No tiene contacto directo con el agua, lo que evita problemas de corrosión y electrólisis. Su principio es el de un condensador de placas paralelas donde el agua actúa como dieléctrico.

Características técnicas

- Chip: XKC-Y25-T12V
- Interfaz: Gravity Digital
- Voltaje de funcionamiento (InVCC): 5~24V DC
- Voltaje de salida (nivel alto): InVCC
- Voltaje de salida (nivel bajo): 0V
- Temperatura de funcionamiento: 0~105 grados Celsius
- Rango para el espesor de la inducción (sensibilidad): 0~13 mm
- Humedad: 5% ~ 100%
- Precio: 15-23 €



Usos en experimentos de física

Control de nivel en depósitos y acuarios
Medida de lluvia acumulada (pluviómetro)
Experimentos de hidrostática: presión en función de la profundidad
Estudio de la relación entre volumen y altura en recipientes de diferentes formas

11. Otros sensores para experimentos de física

Además de los sensores descritos en las secciones anteriores, existen otros dispositivos que pueden resultar de interés para experimentos de física con Arduino y ESP32.

11.1 Sensor de corriente ACS712

Principio de funcionamiento

El ACS712 es un sensor de corriente basado en el efecto Hall. La corriente a medir circula por un

conductor interno de cobre, generando un campo magnético proporcional a la intensidad de corriente. Un chip Hall integrado en el mismo encapsulado detecta este campo y produce un voltaje linealmente proporcional a la corriente. Esto permite medir corriente continua y alterna sin necesidad de interrumpir el circuito.



Características técnicas

- Rango: ±5 A, ±20 A o ±30 A según versión
- Sensibilidad: 185 mV/A (versión 5 A), 100 mV/A (20 A), 66 mV/A (30 A)
- Alimentación: 5 V
- Salida: analógica (VCC/2 para 0 A)
- Precio: ~2 €

Usos en experimentos de física

Medida de corriente en circuitos eléctricos
Estudio de la ley de Ohm
Verificación de las leyes de Kirchhoff
Medición del consumo de dispositivos eléctricos

11.2 Sensor de sonido MAX9814 / KY-038

Principio de funcionamiento

Los sensores de sonido utilizan un micrófono de condensador (electret) que convierte las variaciones de presión sonora en variaciones de voltaje. El MAX9814 incorpora un amplificador con ganancia controlada automáticamente (AGC), lo que permite captar desde susurros hasta sonidos intensos sin saturación.

El KY-038 es más simple: usa un micrófono electret y un amplificador LM393, proporcionando salida analógica (envolvente de la onda) y digital (comparador con umbral ajustable).



Características técnicas

- MAX9814: ganancia ajustable (40, 50, 60 dB), AGC, respuesta de 20 Hz a 20 kHz, precio ~4 €
- KY-038: salida analógica y digital, umbral ajustable por potenciómetro, precio ~1 €

Usos en experimentos de física

Medida de la velocidad del sonido mediante retardo entre dos micrófonos
Estudio de ondas sonoras y frecuencia (tono)
Análisis de la atenuación del sonido con la distancia
Visualización de formas de onda

11.3 Sensor de flujo de agua YF-S201

Principio de funcionamiento

El YF-S201 es un sensor de caudal de agua basado en el efecto Hall, ya que el flujo de agua hace girar una turbina interna que contiene un imán; un sensor Hall detecta los pasos del imán y genera un pulso por cada revolución. Midiendo la frecuencia de los pulsos se determina el caudal.



Características técnicas

- Rango: 1-30 L/min
- Precisión: ±10 %
- Pulso por litro: ~450 (F = 7,5 Q, donde F es frecuencia en Hz y Q caudal en L/min)

- Alimentación: 5-24 V
- Precio: ~3 €

Usos en experimentos de física
Medición de caudal en fluidos
Estudio de la ecuación de continuidad
Experimento de Torricelli (velocidad de salida por un orificio)
Calibración de bombas de agua

12. Consideraciones generales sobre conexión de sensores

Conexión en Arduino UNO
Los sensores se pueden conectar directamente a los pines de 5 V, GND, y las entradas analógicas (A0-A5) o digitales correspondientes.

Conexión en ESP32 (Wemos D1 R32)
Los sensores se pueden conectar a 3,3 V o 5 V (la Wemos D1 R32 dispone de pines de 5 V). Para sensores I²C, SCL se conecta al pin 22 y SDA al pin 21. Para entradas analógicas hay que tener en cuenta que el ESP32 solo soporta hasta 3,2 V en estos pines.

Soldadura de pines
Muchos sensores se venden con los pines sin soldar. Para soldarlos, lo más fácil es pincharlos en una protoboard y colocar la plaquita del sensor encima, soldando por la parte inferior, como lo explican los técnicos de Adafruit (https://learn.adafruit.com/adafruit-bme280-humidity-barometric-pressure-temperature-sensor-breakout/assembly).

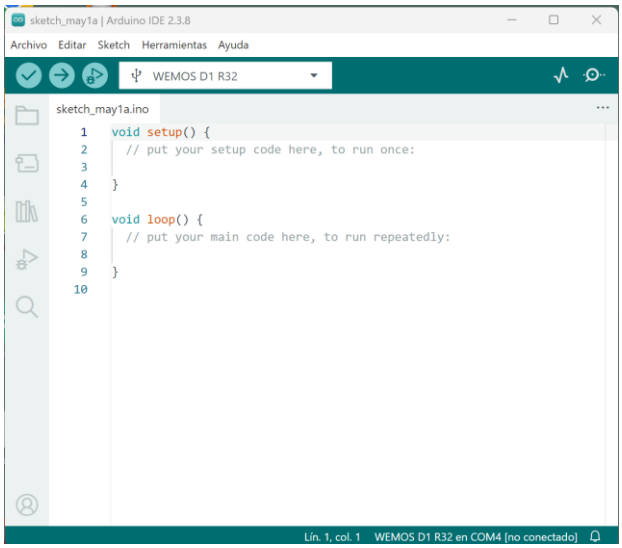
IDE de Arduino: instalación, configuración y uso

El IDE (Entorno de Desarrollo Integrado) de Arduino es el software que utilizaremos para escribir el código de los proyectos (sketches), comprobarlos y cargarlos en las placas Arduino y ESP32.

Instalación del software

Se puede descargar gratuitamente desde la página oficial de Arduino. Para ello hay que ir a https://www.arduino.cc/, descargar la última versión del IDE (en 2026, Arduino IDE 2.3) e instalar el programa siguiendo los pasos del asistente de instalación.

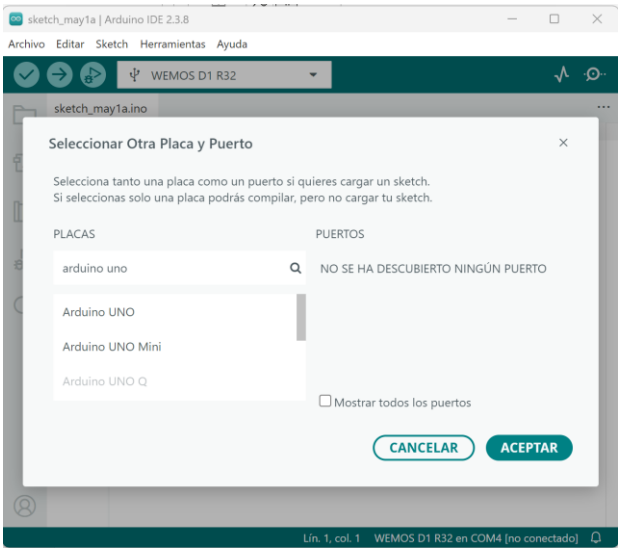
Una vez instalado, al abrir el IDE aparece la interfaz que consta de un editor de texto para escribir el código, una barra de herramientas horizontal con los botones principales y una vertical con las placas y las librerías. También un área de mensajes en la parte inferior y un monitor serie para visualizar los datos



Reconocimiento de las placas en el IDE

Arduino UNO

La placa Arduino UNO está reconocida por defecto en el IDE de Arduino. Solo es necesario conectar la placa al ordenador mediante un cable USB y, en el cajetín de placas, seleccionar la placa y el puerto a la que está conectada al ordenador.



Alternativamente se puede hacer a través del menú Herramientas → Placa, seleccionar Arduino UNO, y en Puerto, seleccionar el puerto COM asignado a la placa.

Windows reconoce automáticamente las placas Arduino y les asigna un puerto serie COM.

Placas con ESP32 (Wemos D1 R32, DOIT ESP32 DEVKIT V1, etc.)

La placa ESP32 no la fabrica Arduino (es de la empresa china Espressif Systems, https://www.espressif.com/), por lo que el IDE de Arduino no la reconoce por defecto. Es necesario añadir el soporte siguiendo estos pasos:

Paso 1: Añadir la URL del gestor de placas

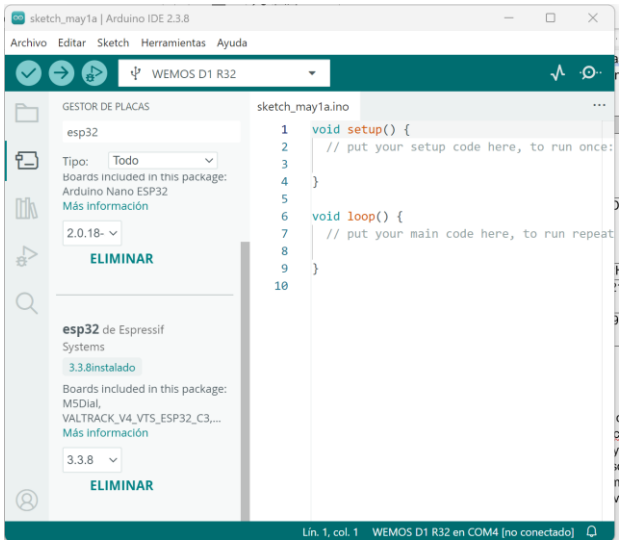
- Abrir el IDE de Arduino.
- Ir a Archivo → Preferencias.
- En la pestaña Ajustes, localizar el campo Gestor de URLs Adicionales de Tarjetas.
- Pegar la siguiente dirección URL:

`https://espressif.github.io/arduino-esp32/package_esp32_index.json`

- Pulsar OK para guardar.

Paso 2: Instalar el soporte para ESP32 desde el gestor de placas

- Ira la barra de herramientas vertical y clicar en el icono de tarjetas, o alternativamente, ir a Herramientas → Placa → Gestor de tarjetas.
- En la ventana de búsqueda, escribir esp32.
- Aparecerá esp32 by Espressif Systems. Pulsar en Instalar.
- Esperar mientras se descargan las definiciones de tarjetas (puede tardar unos minutos).
- Durante la instalación, preguntará si se desean instalar los controladores; responder que sí.



Paso 3: Seleccionar la placa y el puerto

- Conectar la placa al ordenador mediante un cable USB.
- En el cajetín de placas seleccionar la placa y el puerto, o alternativamente, ir a Herramientas → Placa y → puerto.
- Wemos D1 R32: seleccionar DOIT ESP32 DEVKIT V1 o ESP32 Dev Module.

Paso 4 (también podría ser el Paso 1): Instalar el driver USB para Windows
Es posible que al conectar el esp32 al ordenador lo reconozca directamente el sistema operativo, pero también es posible que no, porque todas las placas que tienen un esp32 no tienen el mismo controlador usb.

En este último caso se debe ir al administrador de dispositivos y mirar el nombre que aparece en los puertos COM a los que el sistema no ha encontrado los controladores (drivers) e instalar el controlador correspondiente.

Si la placa es la Wemos D1 R32 el controlador del circuito integrado de conexión USB es el CH340/CH341, y se ha se descargar el driver desde la web de Wemos:
https://www.wemos.cc/en/latest/ch340_driver.html

Si es otra ESP32 que necesita el controlador CP210x USB to UART bridge, entonces se ha de descargar de:
https://www.silabs.com/software-and-tools/usb-to-uart-bridge-vcp-drivers?tab=downloads

Después de instalar el driver, se puede comprobar en el Administrador de dispositivos de Windows qué puerto COM ocupa la placa (aparecerán como "USB-SERIAL CH341 o CP210x USB to UART bridge).

Arduino Nano 33 BLE Sense

A pesar de ser de Arduino esta placa no está instalada por defecto, así que hay que Instalarla.

- Abrir el IDE de Arduino (sin la placa conectada).
- Ir a Herramientas → Placa → Gestor de tarjetas (o usar el icono de placas en la barra lateral).
- Escribir Mbed OS en el buscador.
- Instalar Arduino Mbed OS Nano Boards.
- Durante la instalación, preguntará si se desean instalar los controladores; responder que sí.
- Una vez terminado, conectar la placa al ordenador mediante un cable USB.
- En el IDE, seleccionar Arduino Nano 33 BLE y el puerto COM correspondiente.

Al instalar la placa se instalan los controladores, de manera que Windows ja la reconocerá automáticamente y le asignará un puerto serie.

Gestión de librerías

Las librerías (también conocidas como bibliotecas) son conjuntos de código preescrito que añaden funcionalidades a los proyectos Arduino, permitiendo interactuar con sensores, pantallas o protocolos de comunicación (como I²C o SPI) sin necesidad de programar a bajo nivel desde cero.

Instalar librerías desde el Gestor de bibliotecas (método recomendado)

La mayoría de las librerías se pueden instalar directamente desde el Gestor de bibliotecas.
1. Ir a Programa → Incluir librería → Administrar bibliotecas (o usar el icono de librerías en la barra lateral).
2. Escribir el nombre de la librería en el buscador.
3. Seleccionar la librería deseada y pulsar Instalar.

Las librerías que vienen de serie con el IDE de Arduino (como `Wire` para I²C o `SPI` para comunicación SPI) no necesitan instalarse.

Instalar librerías desde un archivo ZIP

Algunas librerías se han de instalar manualmente desde un archivo ZIP.
1. Descargar el archivo ZIP de la librería (normalmente desde GitHub).
2. Ir a Programa → Incluir librería → Añadir biblioteca .ZIP.
3. Seleccionar el archivo ZIP descargado.
4. El IDE lo instalará automáticamente.

Algunas librerías utilizadas en este libro

Sensor/ Dispositivo	Librería	Cómo se instala
Sensor VL53L0X	Adafruit_VL53L0X	ZIP (GitHub) o Gestor
Sensor DS18B20	OneWire + DallasTemperature	Gestor de bibliotecas
Sensor BME280/BMP280	Seeed_BME280 o Adafruit_BME280	ZIP (GitHub) o Gestor
Sensor BH1750	BH1750	ZIP (GitHub) o Gestor
Bluetooth ESP32	BluetoothSerial	Viene con el soporte ESP32
Phyphox BLE	phyphoxBle	ZIP (GitHub) o Gestor
Tarjeta microSD (ESP32)	FS, SD, SPI	Vienen con el soporte ESP32
Tarjeta microSD (Arduino)	SD	Gestor de bibliotecas

Instalación de librerías de los sensores del Arduino Nano 33 BLE Sense

Puesto que la placa integra múltiples sensores, es necesario instalar sus librerías desde el gestor de bibliotecas (Programa → Incluir librería → Administrar bibliotecas, (o usar el icono de librerías en la barra lateral)).

Actualmente hay dos versiones de la placa, V1 y la más actual V2, y en esta última han cambiado el sensor de temperatura y humedad, así que hemos de tenerlo en cuenta.

Sensor	Librería	Autor
Bluetooth + Phyphox	phyphoxBLE	RWTH Aachen University
Acelerómetro, giroscopio, magnetómetro	Arduino_LSM9DS1	Arduino
Barómetro	Arduino_LPS22HB	Arduino
Temperatura y humedad V1	Arduino_HTS221	Arduino
Temperatura y humedad V2	Arduino_ HS300x	Arduino
Luz y color	Arduino_APDS9960	Arduino

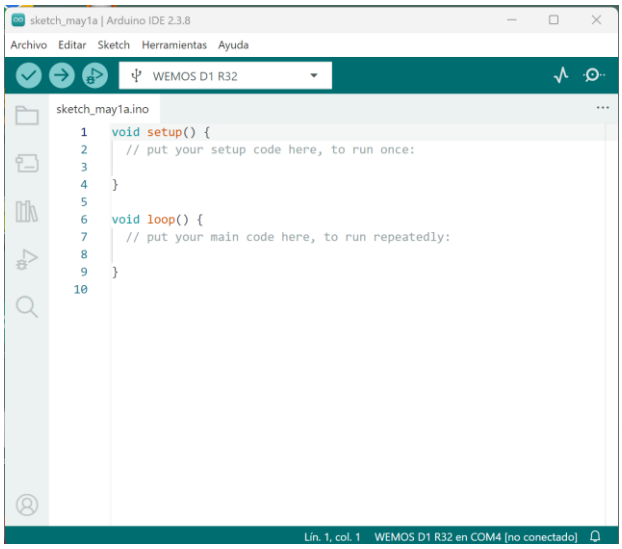
Estructura de un sketch (proyecto)

Un proyecto en Arduino se denomina sketch. Los sketches son programas en formato texto (código) que deben ir en una carpeta con el mismo nombre y tienen la extensión ".ino". Al hacer doble clic, el IDE de Arduino los abre automáticamente

Estructura básica
Todo sketch de Arduino hay al menos dos funciones:

```
void setup() {
// Se ejecuta una sola vez al inicio
// Aquí se configuran los pines, se inician sensores, etc. Por ejemplo:
Serial.begin(9600); //Inicia la comunicación serie
}
void loop() {
// Se ejecuta repetidamente
// Aquí se leen sensores, se envían datos, etc. Por ejemplo:
```

```
delay(1000); // Espera 1 segundo entre repeticiones
}
```



Un sketch de Arduino no solo contiene funciones, sino también la inclusión de librerías, la declaración de variables y la configuración de las comunicaciones. A continuación se detalla cada parte:

Comentarios

Los comentarios son notas para el programador. El compilador los ignora por completo, por lo que no ocupan espacio en la memoria del microcontrolador.

Los comentarios de una sola línea se escriben poniendo dos barras diagonales //. Todo lo que siga en esa línea será ignorado. Los comentarios de múltiples líneas se abren con /* y se cierran con */.

```
// Esto es un comentario de una sola línea
/* Esto es un comentario
   de varias líneas. Muy útil para
   explicar bloques complejos de código */
```

Inclusión de librerías y definición de variables globales

Las librerías se incluyen al principio del código mediante la instrucción #include. Esto permite usar funciones preprogramadas para sensores complejos. Al principio también se incluyen las variables globales.

```
#include <Wire.h> // Librería para comunicación I2C
#include <Adafruit_Sensor.h> // Librería base de sensores

int pinLED = 13; // Variable global: accesible en todo el programa
```

Función setup()

La función setup() se ejecuta una sola vez cuando el microcontrolador se enciende o se reinicia. Se utiliza para configurar pines, inicializar sensores y abrir el puerto serie.

```
void setup() {
  // Configuración de pines
  pinMode(pinLED, OUTPUT);

  // Inicialización de la comunicación serie
  // El número 9600 indica la velocidad en baudios (bits por segundo)
  Serial.begin(9600);

  // Imprimir un mensaje inicial en el Monitor Serie
  Serial.println("Iniciando experimento...");
}
```

Función loop()

La función loop() se ejecuta continuamente, en un bucle infinito, mientras la placa tenga energía. Aquí reside la lógica principal: lectura de sensores, cálculos físicos y envío de datos.

```
void loop() {
  // Enviar datos por el puerto serie para visualizarlos en el Monitor Serie o Serial Plotter
  Serial.print("Dato medido: "); // Imprime texto sin salto de línea
  Serial.println(25.4); // Imprime el valor y añade un salto de línea

  delay(1000); // Pausa la ejecución durante 1000 milisegundos (1 segundo)
}
```

Botones principales de la barra de herramientas

Icono	Función
✓ (Verificar)	Comprueba que el código no tenga errores de sintaxis
→ (Subir)	Verifica el código y lo carga en la placa
🔍 (Monitor serie)	Abre una ventana para ver los datos enviados por la placa

Cada vez que se sube un proyecto nuevo a una placa microcontroladora se borra en la placa el sketch anterior que se hubiera subido.

Visualización de datos

Las tarjetas tienen dos formas básicas de enviar los datos capturados por los sensores: a través del cable USB (puerto serie) o por bluetooth. A continuación se indican algunas de las aplicaciones que permiten capturarlos.

Monitor serie del IDE de Arduino

La forma más sencilla de ver los datos que envía la placa es a través del monitor serie:
1. Cargar el proyecto en la placa.
2. Pulsar el icono de la lupa (🔍) en la esquina superior derecha.
3. Asegurarse de que la velocidad de transmisión (baudios) coincide con la configurada en el código (Serial.begin(9600)).

Serial Plotter (visor gráfico integrado)

El IDE de Arduino incluye una herramienta llamada Serial Plotter que permite visualizar gráficamente los datos en tiempo real, sin necesidad de instalar programas externos.

1. Cargar el proyecto en la placa y abrir el monitor serie para verificar que los datos llegan correctamente.
2. Pulsar el icono de la gráfica en la esquina superior derecha, o ir a Herramientas → Serial Plotter (o pulsar Ctrl+Mayús+L).
3. Seleccionar la misma velocidad de transmisión configurada en el código (Serial.begin(9600)).
4. Los datos numéricos aparecerán representados como gráficas de líneas en tiempo real, con colores distintos para cada valor separado por espacios o tabuladores.

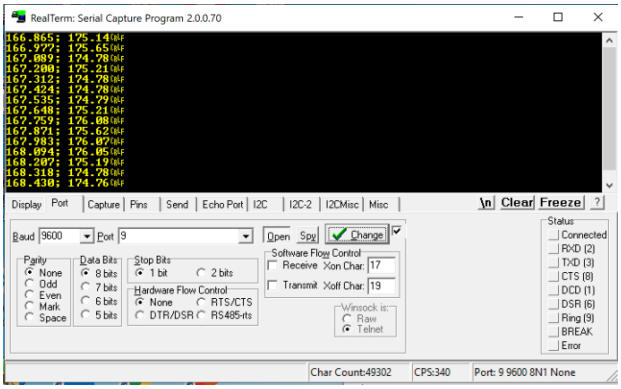
El Serial Plotter del IDE de Arduino tiene la ventaja de que está integrado en el IDE y no requiere instalación adicional.

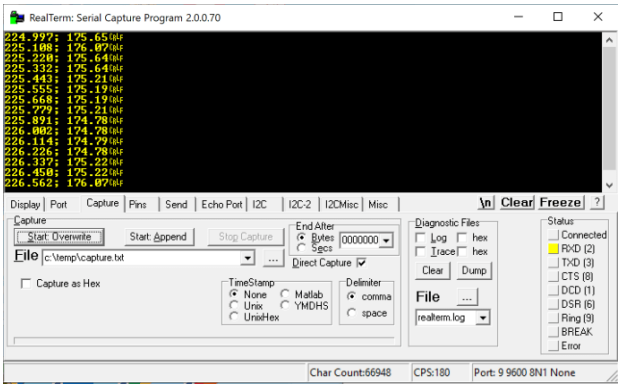
Por otro lado permite visualizar múltiples variables simultáneamente (hasta 10 canales) por lo que es ideal para ver tendencias, oscilaciones y comportamientos dinámicos de los sensores.

Tiene la limitación de que no guarda los datos en un archivo. Para ello se necesita RealTerm u otro programa de captura serie, si no, se han de seleccionar, copiar y pegar a mano donde interese.

RealTerm

RealTerm es un programa gratuito que permite capturar los datos del puerto serie y guardarlos en un archivo de texto.
1. Descargar de: https://realterm.sourceforge.io/
2. Abrir el puerto serie correspondiente (pestaña Port), por ejemplo 9600 baudios.
3. En la pestaña Capture, seleccionar la ruta del archivo donde se guardarán los datos.
4. Pulsar Start Overwrite (o Start Append si se quiere añadir a datos existentes).
5. Si Direct Capture está seleccionado, se dejarán de ver los datos en pantalla pero se acelera el proceso.





KST
KST (disponible en https://kst-plot.kde.org/) es un programa multiplataforma, gratuito y de código abierto que permite representar gráficamente en

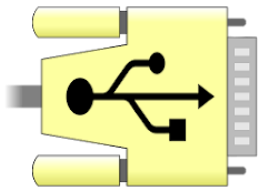
tiempo real los datos que se reciben por el puerto serie a través de RealTerm.

Uso básico de KST:
1. Ejecutar KST.
2. Abrir el asistente de datos (icono correspondiente).
3. Seleccionar el archivo que RealTerm está actualizando.
4. Configurar el formato ASCII (separadores, nombres de columna, etc.).
5. Seleccionar las columnas a representar.
6. En el rango de datos, seleccionar "count from end" con 200-500 puntos para tiempo real.

Serial USB Terminal (dispositivos Android)

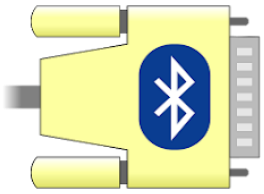
Esta app permite visualizar texto y guardar registros de los datos capturados, por lo que es muy útil para trabajar en el laboratorio sin ordenador. Para visualizar los datos en un dispositivo Android se le ha de conectar la placa directamente mediante un cable OTG (On-The-Go).

1. Descargar la aplicación “Serial USB Terminal” desde Google Play.
2. Conectar la placa (Arduino o ESP32) al teléfono o tablet Android usando un adaptador OTG (y, si hace falta, un cable USB).
3. Abrir la aplicación y conceder los permisos para acceder al dispositivo USB.
4. En la configuración de la app, aseguraos de que la velocidad (Baud rate) coincida con la configurada en el código del microcontrolador (por ejemplo, 9600 o 115200).
5. Pulsar el icono de conexión en la parte superior para empezar a recibir datos, que después se pueden guardar o enviar para su estudio posterior.

Serial USB Terminal	
	

Serial Bluetooth Terminal (dispositivos Android)

Para visualizar los datos en un dispositivo Android cuando la placa transmite mediante Bluetooth.
1. Descargar la aplicación Serial Bluetooth Terminal desde Google Play.
2. Emparejar el teléfono con la placa desde los ajustes de Bluetooth de Android (aunque a veces la propia app conecta directamente sin emparejamiento previo).
3. Seleccionar el dispositivo Bluetooth de la placa en la aplicación.

Serial Bluetooth Terminal	
	

HM10 Bluetooth Serial Lite (Dispositivos iOS)

Para iPhone/iPad:
1. Descargar BLESerial HM10 Lite desde la App Store.
2. Conectar con la placa Bluetooth.

BLRSerial HM10	
	

Phyphox

Phyphox (Physical Phone Experiments, https://phyphox.org/) es una aplicación gratuita para Android e iOS desarrollada por la Universidad RWTH Aachen que permite visualizar y procesar datos en gráficos personalizados recibidos fundamentalmente por Bluetooth de Baja Energía (BLE).

Para usar Phyphox con una placa:

1. Cargar en la placa un proyecto que incluya la librería phyphoxBle.h.
2. Abrir Phyphox en el teléfono.
3. Pulsar el botón + y seleccionar Añadir experimento por Bluetooth.
4. Seleccionar el dispositivo con el nombre asignado en el código.
5. Los datos aparecerán en gráficos en tiempo real.

Phyphox



En ocasiones es preferible cargar el proyecto directamente en Phyphox a partir de un código QR:
1. Cargar en la placa el proyecto.
2. Abrir Phyphox en el teléfono.
3. Pulsar el botón + y seleccionar Añadir experimento por QR.
4. Guardar el experimento.
5. Al iniciar la toma de datos, el teléfono se conectará a la placa.

Nota. Para que Phyphox detecte la placa por Bluetooth, el teléfono debe tener activados el Bluetooth y la localización.

Formato de los datos

En los códigos de este libro, los datos se envían separados por punto y coma (;) i no por coma (,), con el punto como separador decimal (notación americana). Esto facilita su tratamiento posterior en hojas de cálculo o programas de representación gráfica.

Si se desea importar los datos a un programa en castellano o catalán que use la coma como separador decimal, podéis hacer una sustitución automática de puntos por comas en cualquier procesador de textos.

Solución de problemas comunes

El puerto serie no aparece

Aseguraros de que el cable USB transmite datos (no solo carga). Algunos cables económicos solo tienen los hilos de alimentación.

Probar con otro puerto USB del ordenador.

En Windows, abrir el Administrador de dispositivos y comprobar si aparece el dispositivo en "Puertos (COM y LPT)". Si aparece con un icono de advertencia (triángulo amarillo), el driver no está instalado correctamente.

Ya he comentado que en el caso de algunas placas ESP32 se ha de instalar el driver correspondiente (ver la sección del ESP32). Para Arduino Nano 33 BLE Sense, aseguraros de que se han instalado los controladores durante la instalación de la placa (Mbed OS).

Error de compilación: librería no encontrada

Verificar que la librería está instalada (Programa → Incluir librería → Administrar bibliotecas).

Comprobar que el nombre de la librería en el código (`#include`) coincide exactamente con el nombre instalado (mayúsculas/minúsculas).

Algunas librerías dependen de otras. Por ejemplo, DallasTemperature necesita OneWire. Instalar ambas.

Error "Board not found" al subir el código

Comprobar que la placa correcta está seleccionada en Herramientas → Placa, y verificar que el puerto COM seleccionado es el correcto.

Hay veces que hay que desconectar el sensor para que se cargue el sketch.

En placas ESP32, puede ser necesario mantener pulsado el botón BOOT mientras se inicia la subida, y soltarlo cuando empiece a cargar.

Algunas placas ESP32 requieren pulsar el botón EN (reset) después de cargar el código.

Otros errores al subir el código

Así como en los arduinos no suele haber problemas, en los ESP32, por su variedad y por no ser nativos

del IDE de Arduino, a veces aparecen problemas cuando, una vez compilado el código, intenta subirlo a la placa. Muchas veces se soluciona este problema haciendo alguna de las siguientes cosas:

Pulsar el botón Boot en el momento de la carga, en las placas que dispongan de él.

Seleccionar el nombre de otra placa, por ejemplo el genérico ESP32 Dev Module.

Desconectar el sensor de la placa y volver a conectarlo una vez cargado el código.

Disminuir la velocidad de transferencia (Upload Speed) en la carga. Para reducir la velocidad de transferencia en el IDE de Arduino, ir a Herramientas > Upload Speed (o Velocidad de subida). Si está muy alto (por ejemplo a 921600) hay que bajarlo a un valor mucho más estable y conservador, como 115200. Tardará un poco más en cargar el código, pero la estabilidad de la señal cuadrada estará garantizada.

Datos ilegibles en el monitor serie

La velocidad de transmisión (baudios) en el monitor debe coincidir con la del código. Si el código usa Serial.begin(115200), seleccionar 115200 en el monitor. Si aparecen caracteres extraños (ʕʔʔʔ), la velocidad suele ser incorrecta.

El ESP32 no se conecta por Bluetooth

La conexión bluetooth muchas veces da problemas, así que no hay que desesperar.

Para Bluetooth Classic: usar la aplicación Serial Bluetooth Terminal (Android) o HM10 Bluetooth Serial Lite (iOS).

Para Phyphox (BLE): aseguraros de que el Bluetooth y la localización del teléfono están activados.
Comprobar que la librería phyphoxBle.h está correctamente instalada.
Verificar que el nombre del dispositivo Bluetooth en el código coincide con el que se busca en la aplicación.

Alternativas al IDE de Arduino

Aunque en este libro utilizamos exclusivamente el IDE de Arduino por su simplicidad y facilidad de uso, existen entornos más avanzados para programar microcontroladores:

PlatformIO: Extensión para Visual Studio Code (VS Code, de Microsoft) que permite programar Arduino, ESP32 y cientos de placas más con un gestor de librerías integrado, depuración avanzada y soporte para múltiples frameworks (Arduino, ESP-IDF, etc.). Es gratuito y de código abierto (PlatformIO Lab).

Para usarlo:
1. Instalar Visual Studio Code desde https://code.visualstudio.com/
2. Abrir VS Code e ir a la pestaña de extensiones (Ctrl+Mayús+X) y buscar e instalar PlatformIO IDE.
3. Crear un nuevo proyecto seleccionando la placa y el framework (Arduino).
4. Escribir el código y usar los botones de Build y Upload de PlatformIO.

Todos los proyectos de este libro pueden realizarse con PlatformIO si se prefiere, ya que es compatible con el mismo código Arduino, pero se ha de llamar al principio del código a la librería Arduino:

```
#include <Arduino.h>
```

ESP-IDF: El framework oficial de Espressif para programar ESP32 directamente, sin pasar por el ecosistema Arduino. Ofrece máximo control y rendimiento, pero su curva de aprendizaje es más pronunciada.

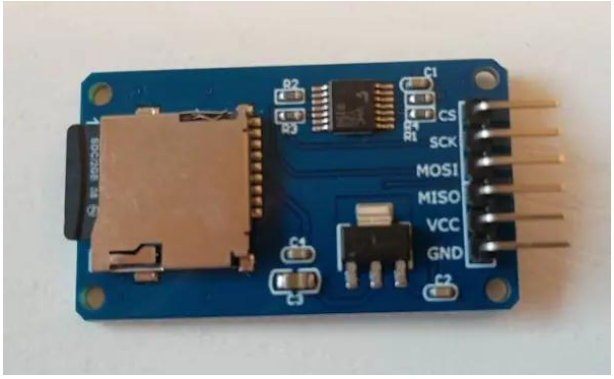
Tarjetas microSD: almacenamiento de datos en experimentos autónomos

En capítulos anteriores he explicado cómo conectar sensores a los microprocesadores Arduino y ESP32 y cómo programarlos para visualizar y guardar los datos que se obtengan, tanto en el ordenador como en el teléfono móvil. En este capítulo explico cómo recoger los datos captados por los sensores en una tarjeta microSD para posteriormente recuperarlos y analizarlos.

Muchas veces el experimento que realizamos tiene lugar en un tiempo corto y va bien recoger los valores que proporciona Arduino mientras se van produciendo, pero hay ocasiones en las que queremos datos a lo largo de mucho tiempo (la evolución de la temperatura y la humedad en un invernadero o en un eclipse de Sol por ejemplo) o fuera del alcance de nuestro ordenador o teléfono (si subimos Arduino con un dron en el aire, por ejemplo) y es recomendable que se vayan registrando en el propio hardware. Aquí es cuando puede venir bien una tarjeta microSD que los guarde.

El módulo lector de tarjetas microSD

El módulo lector de tarjetas microSD se comunica con las tarjetas mediante el protocolo SPI, un bus de comunicación síncrono que utiliza cuatro líneas: SCK (reloj de sincronización), MOSI (datos del microcontrolador al periférico), MISO (datos del periférico al microcontrolador) y CS (selección de chip, para activar el dispositivo).



En la tabla siguiente se indican los pines SPI predeterminados a los que se puede conectar el módulo lector de tarjetas microSD a los ESP32:

Pin del módulo	Conexión en ESP32 (Wemos D1 R32)
VCC	5 V
CS	GPIO5
MOSI	GPIO23
CLK (SCK)	GPIO18
MISO	GPIO19
GND	GND

Aunque algunos fabricantes indican que el lector de tarjetas SD funciona a 3,3 V y 5 V, la mayoría de los módulos incorporan un regulador de voltaje que necesita al menos 4,3 V en la entrada para funcionar de forma estable. Por ello, en la práctica es recomendable conectar VCC a 5 V.

El pin GPIO5 actúa como CS (selección de chip); tened en cuenta que este pin participa en el arranque del ESP32 (strapping), aunque con la tarjeta SD en reposo (CS=HIGH durante el reset) no suele causar problemas.

Para introducir la tarjeta microSD en el lector, presionad suavemente el muelle que se encuentra dentro. Hay una posición en la que entra y otra en la que no. Previamente debéis comprobar que la tarjeta microSD esté formateada como FAT32. Esto debéis hacerlo con un ordenador que tenga o al que se acople un lector de tarjetas. Tened en cuenta que FAT32 tiene un límite de 4 GB por archivo, y el sistema de archivos soporta un máximo de 65536 entradas en el directorio raíz.

Librerías necesarias

Las librerías para el lector de tarjetas vienen de serie con el IDE de Arduino si están actualizadas las tarjetas Arduino y ESP32. En el caso de las placas con microprocesador ESP32, la biblioteca necesaria es la de Espressif en su versión 2.0 o superior.

Para instalar el soporte ESP32 en el IDE de Arduino, debéis añadir la siguiente URL en Archivo → Preferencias → Gestor de URLs Adicionales de Tarjetas:

`https://espressif.github.io/arduino-esp32/package_esp32_index.json`

Luego, en Herramientas → Placa → Gestor de tarjetas, buscad esp32 by Espressif Systems e instaladlo. Esto instalará automáticamente las librerías necesarias para el lector de tarjetas microSD (FS, SD, SPI).

Ejemplo completo: registro de datos de un sensor BME280 en tarjeta microSD

En este ejemplo se conecta un sensor BME280 (presión, temperatura y humedad) a una placa Wemos D1 R32 (ESP32) y se registran los datos en una tarjeta microSD. El sensor BME280 se comunica por I²C.

Conexiones

BME280: VCC a 3,3 V, GND a GND, SCL a SCL (GPIO22), SDA a SDA (GPIO21)

Lector microSD: según la tabla de la sección anterior

Código

```
/*
  Registro de datos de temperatura, humedad y presión en tarjeta microSD
  Sensor: BME280 (I2C)
  Placa: Wemos D1 R32 (ESP32)

  Los datos se guardan en data.txt en la tarjeta microSD con estructura:
Tiempo(s);Temperatura(°C);Humedad(%);Presion(Pa);Altitud(m)
*/

// Librerías para tarjeta SD (vienen con el soporte ESP32)
#include "FS.h"
#include "SD.h"
#include <SPI.h>

// Librerías para el sensor BME280 (I2C)
#include "Seeed_BME280.h"  // Hay que instalarla
#include <Wire.h>  // Viene por defecto

BME280 bme280;

// Variables del sensor
float temp;      // Temperatura en °C
float hum;       // Humedad en %
uint32_t pres;   // Presión en Pa (getPressure() retorna uint32_t)
float tiempoSegundos;
float altitud;   // Altitud calculada en m

String dataMessage;

// Inicializa la tarjeta SD
void initSDCard() {
  if (!SD.begin()) {
    Serial.println("Error: no se pudo montar la tarjeta SD");
    return;
  }
  uint8_t cardType = SD.cardType();

  if (cardType == CARD_NONE) {
    Serial.println("No hay tarjeta SD conectada");
    return;
  }
  Serial.print("Tipo de tarjeta SD: ");
  if (cardType == CARD_MMC) {
    Serial.println("MMC");
  } else if (cardType == CARD_SD) {
    Serial.println("SDSC");
  } else if (cardType == CARD_SDHC) {
    Serial.println("SDHC");
  } else {
    Serial.println("DESCONOCIDO");
  }
  uint64_t cardSize = SD.cardSize() / (1024 * 1024);
  Serial.printf("Tamaño de tarjeta: %lluMB\n", cardSize);
}

// Escribe un archivo nuevo en la SD
void writeFile(fs::FS &fs, const char *path, const char *message) {
  Serial.printf("Escribiendo archivo: %s\n", path);
  File file = fs.open(path, FILE_WRITE);
  if (!file) {
    Serial.println("Error al abrir archivo para escritura");
    return;
  }
  if (file.print(message)) {
    Serial.println("Archivo escrito correctamente");
  } else {
    Serial.println("Error de escritura");
  }
```

```
  file.close();
}

// Añade datos al final de un archivo existente
void appendFile(fs::FS &fs, const char path, const
char message) {
  File file = fs.open(path, FILE_APPEND);
  if (!file) {
    Serial.println("Error al abrir archivo para añadir
datos");
    return;
  }
  if (file.print(message)) {
    Serial.println("Datos añadidos");
  } else {
    Serial.println("Error al añadir datos");
  }
  file.close();
}

void setup() {
  Serial.begin(115200);

  // Inicializar sensor BME280
  if (!bme280.init()) {
    Serial.println("Error al inicializar el sensor
BME280");
  }

  // Inicializar tarjeta SD
  initSDCard();

  // Si el archivo data.txt no existe, crearlo con
cabeceras
  File file = SD.open("/data.txt");
  if (!file) {
    Serial.println("Creando archivo data.txt...");
    writeFile(SD, "/data.txt",
"Tiempo(s);Temperatura(°C);Humedad(%);Presion
_Pa;Altitud(m)\r\n");
  } else {
    Serial.println("El archivo data.txt ya existe");
  }
  file.close();
}

void loop() {
  // Leer sensores
  tiempoSegundos = millis() / 1000.0;
  temp = bme280.getTemperature(); // °C
  hum = bme280.getHumidity(); // %
  pres = bme280.getPressure(); // Pa
  altitud = bme280.calcAltitude(pres); // m

  // Preparar línea de datos (separador punto y
coma)
  dataMessage = String(tiempoSegundos) + ";" +
                String(temp) + ";" +
                String(hum) + ";" +
                String(pres) + ";" +
                String(altitud) + "\r\n";

  Serial.print("Guardando: ");
  Serial.println(dataMessage);

  // Añadir datos al archivo en la SD
  appendFile(SD, "/data.txt", dataMessage.c_str());

  // Mostrar datos por el monitor serie
  Serial.print("Tiempo(s)=");
  Serial.print(tiempoSegundos);
  Serial.print(", Temperatura=");
  Serial.print(temp);
  Serial.print(", Humedad=");
  Serial.print(hum);
  Serial.print(", Presion=");
  Serial.print(pres);
  Serial.print(", Altitud=");
  Serial.println(altitud);

  delay(1000);  // Esperar 1 segundo entre lecturas
}
```

Alimentación autónoma

Aunque el código también envía los datos por el puerto serie, la idea es que la placa pueda funcionar de forma autónoma. Para ello, como explicaba en el capítulo de los microcontroladores, la podéis alimentar con:

- Un powerbank (batería externa para móviles) conectado por USB
- Un cargador de móvil conectado a la red eléctrica
- Un pack de 6 pilas (9 V) conectadas al jack en Arduino UNO, o al jack o a VIN en el ESP32.

Durabilidad de la tarjeta SD

Las tarjetas SD tienen un número limitado de ciclos de escritura (~100 000 para una tarjeta de calidad estándar). En experimentos de larga duración que escriben datos cada segundo, la tarjeta podría alcanzar este límite en cuestión de días. Para uso continuado, considerad reducir la frecuencia de

escritura (por ejemplo, escribiendo un promedio cada 60 segundos) o usar tarjetas SD de alta resistencia (High Endurance), diseñadas para aplicaciones de grabación continua.

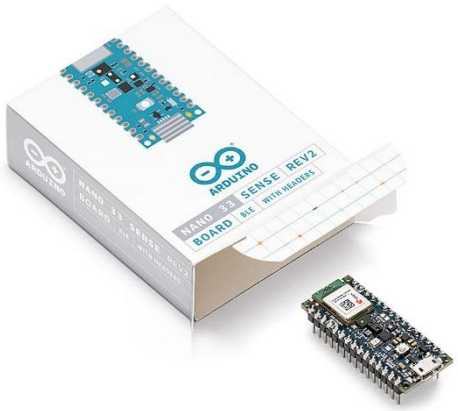
Además, durante la escritura la tarjeta puede consumir picos de hasta 100 mA, por lo que es conveniente asegurar que la fuente de alimentación pueda suministrar esa corriente. Por ejemplo, no es conveniente utilizar una pila de 9 V clásica ja que no rinde en general lo suficiente.

Formato de los datos

Los datos se guardan en un archivo de texto (`data.txt`) en la tarjeta microSD con formato CSV (separados por punto y coma). Esto permite importarlos directamente a una hoja de cálculo o a programas de representación gráfica como KST.

Arduino Nano 33 BLE Sense: conexión con Phyphox y Arduino Science Journal

La placa Arduino Nano 33 BLE Sense es una de las más versátiles del mercado para la experimentación en física (aunque no es barata, unos 45 €). En un formato reducido integra una gran cantidad de sensores y comunicación Bluetooth Low Energy (BLE), lo que permite enviar datos a un teléfono móvil o tableta sin necesidad de cables adicionales.



Instalación de la placa en el IDE de Arduino

Antes de conectar la placa al ordenador, debéis instalar vuestro soporte en el IDE de Arduino:
1. Abrid el IDE de Arduino (sin la placa conectada).
2. Vaya a Herramientas → Placa → Gestor de tarjetas.
3. Escribid Mbed OS en el buscador.
4. Instalad Arduino Mbed OS Nano Boards.
5. Durante la instalación, preguntará si se desea instalar los controladores; responder que sí.
6. Una vez terminado el proceso, conectad la placa al ordenador mediante un cable USB. Windows reconocerá la placa automáticamente.
7. En el IDE, seleccionad Arduino Nano 33 BLE en el menú de placas y elegid el puerto COM correspondiente.

Sensores integrados y microcontrolador

La gran ventaja de esta placa es que incorpora los siguientes sensores sin necesidad de conexiones externas:

Sensor	Componente	Magnitud que mide
Luz, proximidad, color y gestos	APDS9960	Iluminancia, color RGB, proximidad, gestos
Micrófono digital	MP34DT05	Presión sonora (señal PDM digital)
Acelerómetro, giroscopio y magnetómetro (9 ejes)	LSM9DS1	Aceleración (m/s²), velocidad angular (°/s), campo magnético (µT)
Temperatura y humedad V1	HTS221	Temperatura (°C), humedad relativa (%)
Temperatura y humedad V2	HS300x	Temperatura (°C), humedad relativa (%)
Presión barométrica	LPS22HB	Presión atmosférica (hPa)

Microcontrolador:

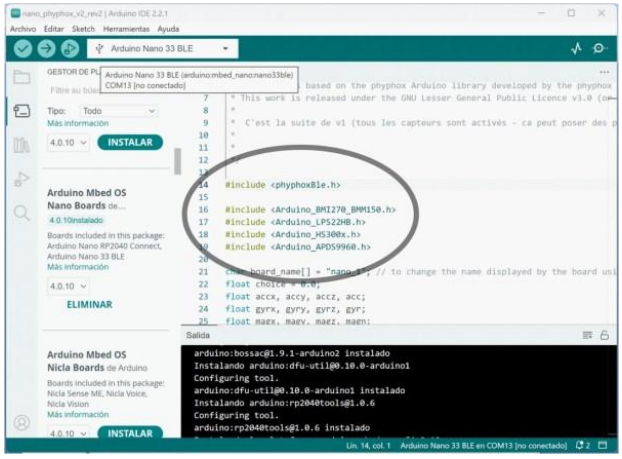
La placa monta un nRF52840 de Nordic Semiconductor (ARM Cortex-M4F a 64 MHz, 1 MB flash, 256 KB RAM) con aceleración BLE por hardware. Todos sus pines GPIO funcionan a 3.3 V. No hay que aplicar nunca 5 V directamente en ningún pin, o se dañará la placa.

Los sensores integrados se comunican con el nRF52840 a través del bus I²C, compartiendo las líneas SDA y SCL. Cada sensor tiene su propia dirección I²C: 0x5C (HTS221), 0x5D (LPS22HB), 0x39 (APDS9960), 0x6A y 0x6C (LSM9DS1).

Librerías necesarias

Para utilizar los sensores de la placa, se deben instalar las siguientes librerías desde el Gestor de bibliotecas del IDE de Arduino (Programa → Incluir librería → Administrar bibliotecas):

Sensor	Librería	Autor
Bluetooth + Phyphox	phyphoxBLE	RWTH Aachen University
Acelerómetro, giroscopio, magnetómetro	Arduino_LSM9DS1	Arduino
Barómetro	Arduino_LPS22HB	Arduino
Temperatura y humedad V1	Arduino_HTS221	Arduino
Temperatura y humedad V2	Arduino_HS300x	Arduino
Luz, color y gestos	Arduino_APDS9960	Arduino

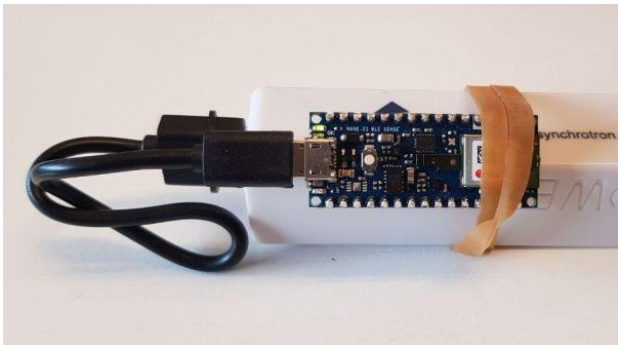


Conexión con Phyphox

Phyphox (https://phyphox.org/) es una aplicación gratuita para Android e iOS que permite visualizar datos en gráficos personalizados recibidos por Bluetooth BLE.

Procedimiento

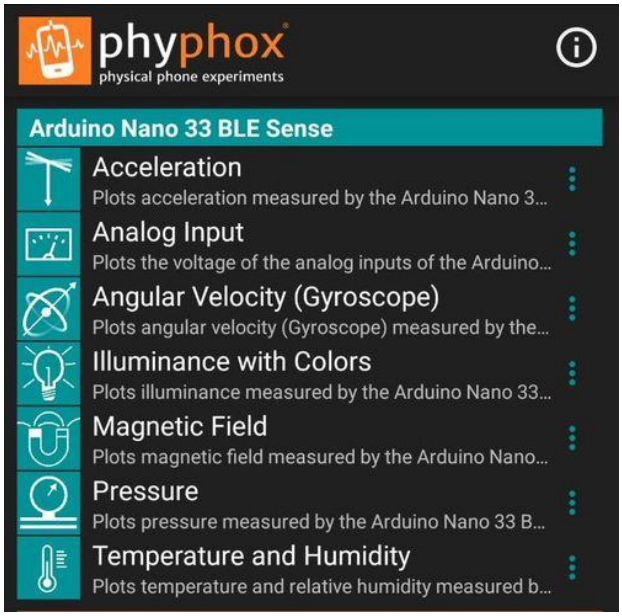
1. Descargad el proyecto que proponen desde la página de La physique autrement (https://vulgarisation.fr/projet/arduino_nano/). Ver también Ramírez (2024).
2. Abridlo en el IDE de Arduino. Por defecto, el proyecto asigna a la placa el nombre nano_6 (se puede cambiar por el nombre que se prefiera en el código).
3. Cargad el proyecto en la placa pulsando el botón Subir (flecha hacia la derecha).
4. Una vez cargado el proyecto, desconectad la placa del ordenador y enchufarla a una batería externa (powerbank) para alimentación autónoma.



5. Abrid la aplicación Phyphox en vuestro teléfono móvil.
6. Pulsad el botón + y seleccionad Añadir experimento desde un código QR.
7. Enfocad el código QR proporcionado por el proyecto (https://vulgarisation.fr/projet/arduino_nano/).

8. Una vez que aparezca el conjunto de experimentos, seleccionad Agregar todos.
9. Seleccionad un experimento (por ejemplo, Pressure).

10. La app pedirá escoger un dispositivo Bluetooth; seleccionad nano_6 (o el nombre que esté asignado).
11. Pulsad el botón de reproducción para empezar a recoger datos.



Importante: para que la aplicación Phyphox detecte la placa a través de Bluetooth, el teléfono debe tener activados el Bluetooth y la localización. Aunque el teléfono no sea capaz de vincular la tarjeta al Bluetooth, no es necesario preocuparse ja que la aplicación Phyphox reconoce la placa y se conecta aunque no esté vinculada al teléfono.

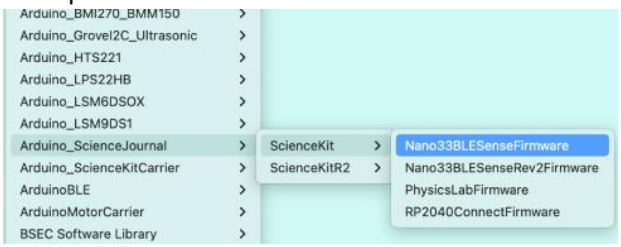
Conexión con Science Journal

Arduino Science Journal (https://science-journal.arduino.cc/) es una aplicación gratuita desarrollada por Arduino (originalmente por Google) que permite recoger datos de los sensores de la placa de forma sencilla. Está disponible para Android e iOS.

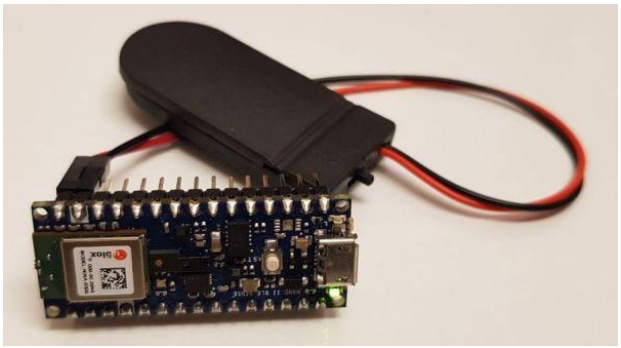
Procedimiento

La placa Nano 33 BLE Sense necesita un firmware (código) específico para funcionar con Science Journal. Sin él, la aplicación no detectará los sensores:

1. Instalad la librería Arduino Science Journal en el IDE de Arduino.
2. A continuación, en Archivo, Ejemplos, Ejemplos de bibliotecas personalizadas, Arduino_sciencejournal, abrid el sketch Nano33BleSenseFirmware o el Nano33BleSenseRev2Firmware, según la versión de la placa.



Alternativamente, descargad e instalad el firmware oficial desde: https://github.com/arduino/Arduino-Science-Journal-Android
3. Seleccionad la placa Arduino Nano 33 BLE y el puerto COM correspondiente.
4. Cargad el firmware pulsando el botón Subir.
5. Una vez cargado, si interesa, se puede desconectar la placa del ordenador y alimentar con un powerbank o una batería externa. El bajo consumo de esta placa permite alimentarla con dos baterías botón de 3 V.



6. Abrid la aplicación Arduino Science Journal en el teléfono.
7. Conectad la placa mediante Bluetooth desde la propia aplicación. Para ello se ha de abrir un experimento y, en sensores, clicar en el engranaje de configuración que permite seleccionar los sensores que interesen, así como llamar a placas Arduino.
8. Seleccionad los sensores que deseéis monitorizar.



9. Los datos se mostrarán en tiempo real y se pueden exportar en formato CSV para su análisis.

El firmware de Science Journal reemplaza cualquier otro programa en la placa. Para volver a usar la placa con vuestros propios sketches, simplemente cargad otro programa. La placa no se daña.

Science Journal es compatible con la Nano 33 BLE Sense en Android e iOS. Aseguraos de tener la última versión de la app y de que el firmware de la placa esté actualizado.

Mecánica. Movimientos

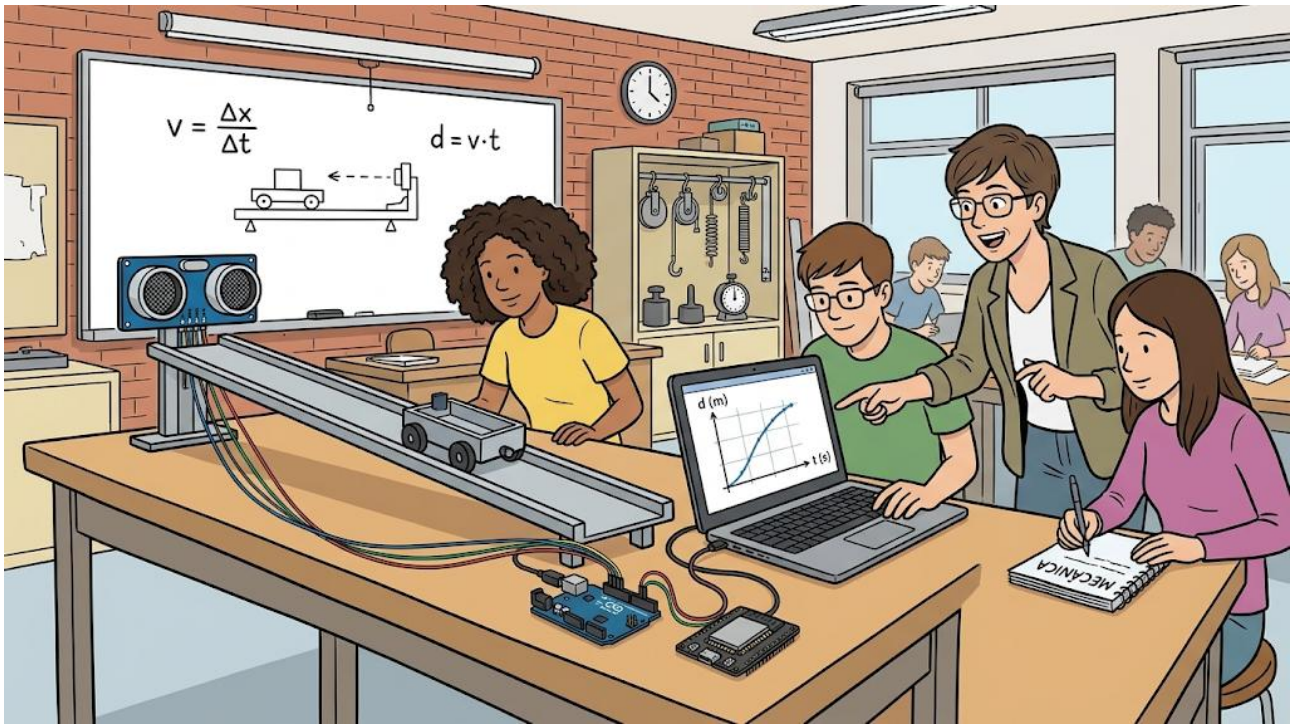
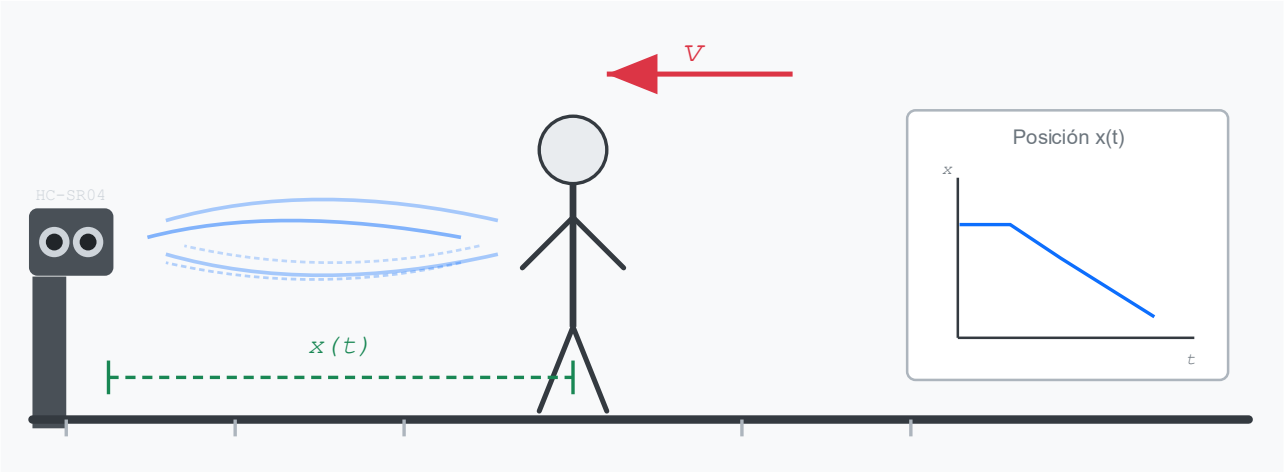


Ilustración realizada con Gemini. Esté pensada para adornar, aunque el alumnado puede entretenerse en buscar los errores evidentes en las conexiones que contiene.

Seguimiento del movimiento de personas



Para empezar a trabajar con sensores y arduinos, en este primer experimento mostraré cómo medir distancias con el sensor de ultrasonidos HC-SR04, que es fácil de conectar y que se puede utilizar sin librerías.

Lo utilizaremos para registrar el movimiento de una persona que camina y analizar su perfil de velocidad y aceleración a partir de los datos de distancia en función del tiempo, igual que lo hacían los alumnos de Secundaria y de Bachillerato en el IES "Leonardo Da Vinci" de Alicante (2008a).

Esta forma de abordar el experimento no es sino una sofisticación del clásico en el que un alumno camina por un pasillo en el que se han marcado en el suelo sucesivas distancias con tiza ayudándose de una cinta métrica. Los compañeros han de tomar tiempos en los que pasa por cada marca (2, 4, 6 m, etc.) con un cronómetro o un teléfono móvil.

Fundamento físico

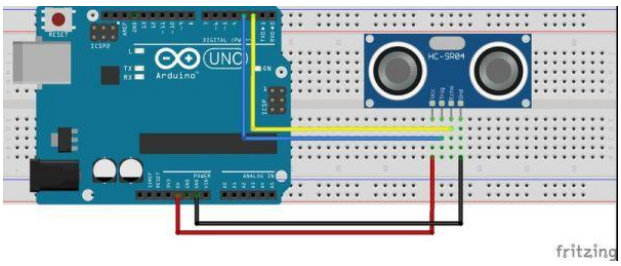
El movimiento de una persona al caminar no es totalmente uniforme ja que durante el ciclo de la marcha se alternan fases de aceleración y deceleración, pero la distancia recorrida en función del tiempo, d(t), permite calcular la velocidad media a partir de la ecuación de la línea de tendencia.

Al representar d(t), se observan un movimiento aproximadamente uniforme (pendiente constante) aunque pueda haber pequeños cambios de velocidad. La velocidad típica de una persona caminando está entre 1,0 y 1,5 m/s (entre 3,6 y 5,4 km/h).

Materiales y montaje experimental

Se necesita una placa Wemos D1 R32 (ESP32) o Arduino UNO, un sensor de ultrasonidos HC-SR04P, una protoboard, cables Dupont, un espacio despejado de unos 5 metros de longitud y un ordenador con el IDE de Arduino.

Para más detalles sobre el sensor HC-SR04, consultad el capítulo de los sensores. Se ha de conectar VCC a 5 V, GND a GND, Trig a un pin digital (D3) y Echo a otro pin digital (D2).

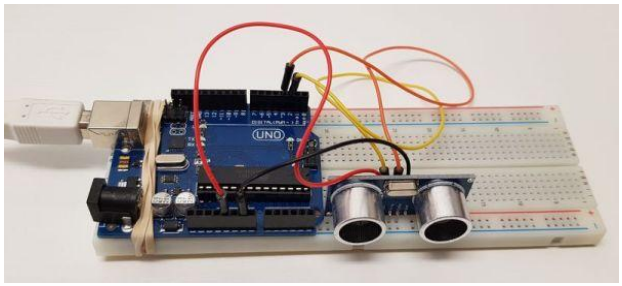


El sensor mide el tiempo que tarda en ir y venir el pulso de ultrasonido (el eco) en microsegundos. Para transformar este tiempo en la distancia a la que se encuentra el objeto se ha de tener en cuenta que el camino es de ida y vuelta, se ha de dividir por dos:

$$d(cm) = v(m/s)*t(\mu s)*10^{-6}*100/2 =$$
$$= 343*5*10^{-5}* t(\mu s) = t(\mu s)/58,3$$

Donde consideramos que 343 m/s es la velocidad del sonido en el aire.

La placa Wemos, que se describe en el primer capítulo, se conecta después al ordenador y se carga el código.



Procedimiento

Fijad el sensor HC-SR04 a una altura de aproximadamente 1 metro del suelo, mirando horizontalmente hacia el pasillo o espacio donde caminará la persona. Aseguraros de que no haya obstáculos entre el sensor y la zona de paso.

Situad a la persona en el punto de inicio (más lejano al sensor) y comenzad la toma de datos. La persona debe caminar hacia el sensor a velocidad constante (lo más uniforme posible). Continuad registrando hasta que llegue a la marca.

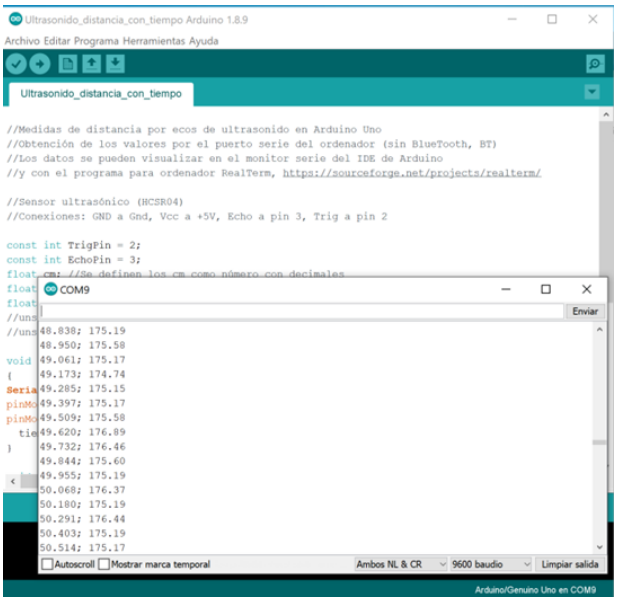
Podéis repetir el experimento con diferentes velocidades (marcha lenta, normal y rápida) y con cambios de sentido en el movimiento (acercándo y alejándose).

Resultados esperados

Los resultados se pueden visualizar en el IDE de Arduino por el puerto serie del ordenador, tal y como se explica en el tercer capítulo del libro. Nos aparecerán dos columnas. La primera el tiempo en milisegundos y la segunda la distancia en cm.

Si se utiliza la placa Wemos con el código para ella, los datos se pueden visualizar en la aplicación Phyphox. La gráfica de distancia frente al tiempo mostrará una línea descendente (la persona se acerca al sensor) cuya pendiente corresponde a la velocidad de la marcha.

Si posteriormente se realiza la gráfica en una hoja de cálculo o en un programa de análisis gráfico se puede añadir a los puntos la línea de tendencia y la ecuación correspondiente, en este caso lineal.



La curva puede presentar pequeñas oscilaciones debidas al balanceo natural del cuerpo al caminar, pero nos dará la velocidad media.

Código para ArduinoUNO

```
//Código para realizar medidas de distancia por ecos de ultrasonido en Arduino Uno
//Se obtienen los valores por el puerto serie del ordenador, de manera que los datos
//Se pueden visualizar en el monitor serie del IDE de Arduino y con el programa para ordenador RealTerm, https://sourceforge.net/projects/realterm/

//Sensor ultrasónico HC-SR04
//Conexiones: GND a Gnd, Vcc a +5V, Echo a pin 3, Trig a pin 2

const int TrigPin = 2;
const int EchoPin = 3;
float cm; //Se definen los cm como número con decimales
float tiempo; // Se define el tiempo como numero con decimales
float tiempoSegundos; // Se define el tiempo en segundos como número con decimales

void setup() {
Serial.begin(9600); // Se inicia el monitor serie para mostrar el resultado
pinMode (TrigPin, OUTPUT);
pinMode( EchoPin, INPUT);
```

```
tiempo = millis(); //mide el tiempo en milisegundos desde que se pone en marcha Arduino
}

void loop() {
digitalWrite(TrigPin, LOW); //para generar un pulso limpio ponemos a LOW 2µs
delayMicroseconds(2);
digitalWrite(TrigPin, HIGH); //Genera un pulso ultrasónico...
delayMicroseconds(10); //...que dura 10 micro segundos
digitalWrite(TrigPin, LOW);

cm = pulseIn(EchoPin, HIGH) / 58.3; // El sensor mide el tiempo que tarda el eco en microsegundos y se transforma en distancia en cm, dividiendo por 58,3

tiempo = millis(); //Actualiza el tiempo actual
tiempoSegundos = tiempo/1000; //cambia los milisegundos a segundos
Serial.print(tiempoSegundos, 3); //Escribe el tiempo en segundos con tres decimales
Serial.print(";");
//Serial.print("s"); //Añade s, de segundos, al valor numérico. Mejor no ponerlo si se desea tratar después los datos

Serial.print(cm, 2); //Escribe la distancia en cm con dos decimales
Serial.println();
//Serial.print("cm"); //Añade cm al valor numérico. Mejor no ponerlo si se desea tratar después los datos

delay(50); //Tiempo de espera entre medidas, en milisegundos
}
```

Código para Wemos y Phyphox

```
//Código para realizar medidas de distancia por ecos de ultrasonido en Wemos D1R32 (ESP32). Seleccionar la placa ESP32 Dev Module.
//Se obtienen los valores por el puerto serie del ordenador, de manera que los datos
se pueden visualizar en el monitor serie del IDE de Arduino y con el programa para ordenador RealTerm, https://sourceforge.net/projects/realterm/
//También se envían por bluetooth y se pueden recoger en la app Phyphox.

//Sensor ultrasónico HC-SR04P
//Conexiones: GND a Gnd, Vcc a +5V, Echo a pin 13, Trig a pin 12
//Se ha de instalar la librería phyphoxBle.

#include <phyphoxBle.h>

// Definición de pines para Wemos D1 R32
const int TrigPin = 12;
const int EchoPin = 13;

float cm;
float tiempoSegundos;
unsigned long tiempoInicial;

void setup() {
  Serial.begin(115200);
  pinMode(TrigPin, OUTPUT);
  pinMode(EchoPin, INPUT);

  // Iniciamos el Bluetooth con el nombre que verás en el móvil
  PhyphoxBLE::start("Wemos_Radar");

  // Creamos la estructura del experimento
  PhyphoxBleExperiment experimento;
  experimento.setTitle("Radar de Distancia");
  experimento.setCategory("Mis Experimentos");
  experimento.setDescription("Mide la distancia frente al tiempo usando un HC-SR04.");

  // Creamos una pestaña para visualizar los datos
  PhyphoxBleExperiment::View vistaPrincipal;
  vistaPrincipal.setLabel("Gráfica y Datos");

  // Añadimos el bloque para ver el número exacto
  PhyphoxBleExperiment::Value distanciaActual;
  distanciaActual.setLabel("Distancia actual");
  distanciaActual.setPrecision(2); // Fuerza a mostrar dos decimales
  distanciaActual.setUnit("cm");
  distanciaActual.setColor("ff0000"); // Color rojo para el texto
  distanciaActual.setChannel(2); // Leerá los datos del canal 2

  // Añadimos la gráfica (X = Tiempo, Y = Distancia)
  PhyphoxBleExperiment::Graph grafica;
  grafica.setLabel("Distancia frente al Tiempo");
  grafica.setUnitX("s");
  grafica.setUnitY("cm");
  grafica.setLabelX("Tiempo");
  grafica.setLabelY("Distancia");
```

```
  grafica.setColor("0000ff"); // Línea de la gráfica en
azul
  grafica.setChannel(1, 2); // Eje X asume el canal
1, Eje Y asume el canal 2

  // Ensamblamos todas las piezas y las adjuntamos
a Phyphox
  vistaPrincipal.addElement(distanciaActual);
  vistaPrincipal.addElement(grafica);
  experimento.addView(vistaPrincipal);
  PhyphoxBLE::addExperiment(experimento);

  tiempoInicial = millis();
}

void loop() {
  // Generamos el pulso
  digitalWrite(TrigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(TrigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(TrigPin, LOW);

  // Medimos el eco y convertimos a cm
  cm = pulseIn(EchoPin, HIGH) / 58.0;

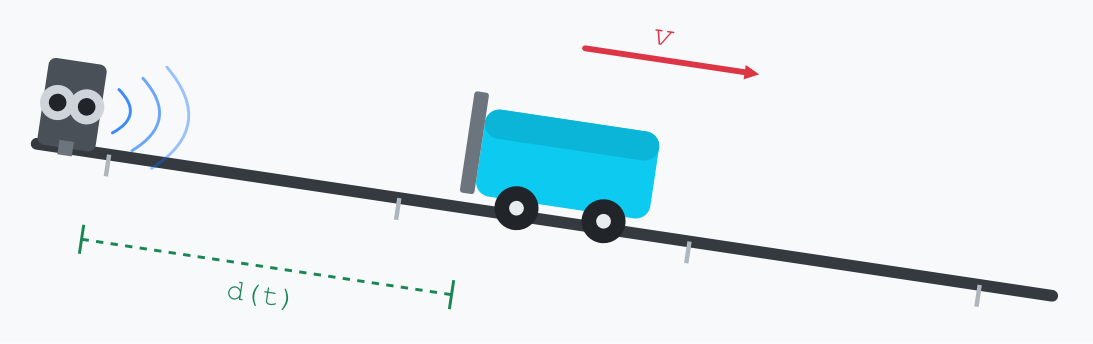
  // Calculamos el tiempo en segundos desde que
inició
  tiempoSegundos = (millis() - tiempoInicial) /
1000.0;

  // ENVIAMOS LOS DATOS A PHYPHOX
  // La función write asocia en orden: Canal 1
(tiempo), Canal 2 (distancia)
  PhyphoxBLE::write(tiempoSegundos, cm);

  // También lo enviamos al monitor serie clásico
para depuración
  Serial.print(tiempoSegundos, 2);
  Serial.print(" s | ");
  Serial.print(cm, 2);
  Serial.println(" cm");

  // //Tiempo de espera entre medidas, en
milisegundos
  delay(50);
}
```

Movimiento de carritos con sensor de distancia



Queremos medir la posición en función del tiempo de un carrito que se desplaza sobre un plano inclinado, utilizando el sensor de distancia por láser infrarrojo VL53L0X y, a partir de los datos obtenidos, determinar la aceleración del movimiento y comprobar la segunda ley de Newton.

Este es un experimento clásico que tradicionalmente se realizaba dejando caer un carrito o una bola por un plano inclinado y midiendo con un cronómetro los tiempos que tardaba el móvil en recorrer determinadas distancias marcadas en el plano (CDECT, 2002).

Más recientemente, con equipos EXAO (Experimentación Asistida por Ordenador) y dattalogers, ya se ha utilizado un sensor de distancia para estudiar el movimiento uniformemente acelerado en un plano inclinado (alumnos de Secundaria y de Bachillerato en el IES "Leonardo Da Vinci", 2008b, Amador, 2025).

También se puede hacer utilizando Arduino como aquí, pero con el sensor de distancia HC-SR04 por ultrasonidos como hacen Domínguez y otros (2019) de la Escuela Superior de Ingeniería de Puerto Real (Cádiz).

Fundamento físico

Cuando un objeto se desplaza sobre un plano inclinado sin rozamiento, la única fuerza que actúa sobre él en la dirección del movimiento es la componente del peso paralela al plano: $F = mg\sin\theta$, donde m es la masa del objeto, g la aceleración de la gravedad y θ el ángulo de inclinación.

Aplicando la segunda ley de Newton, $\sum F = m \cdot a$, la aceleración teórica es $a = g\sin\theta$.

La posición en función del tiempo para un movimiento uniformemente acelerado (MRUA) viene dada por la ecuación cinemática:

$$s(t) = s_0 + v_0 t + \frac{1}{2}at^2$$

Donde s_0 es la posición inicial y v_0 la velocidad inicial.

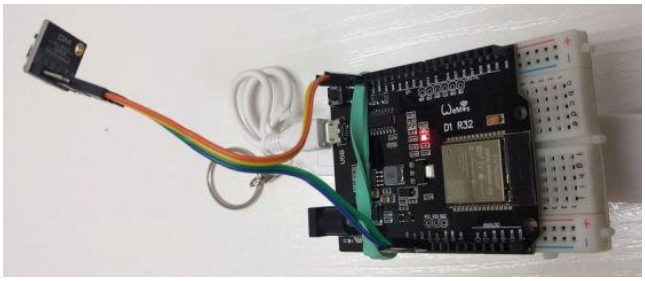
Midiendo la distancia s(t) con el sensor VL53L0X y ajustando los datos a esta ecuación, podemos extraer experimentalmente la aceleración a y compararla con el valor teórico g·sinθ.

Materiales montaje experimental

Se necesita una placa Wemos D1 R32 (ESP32), un sensor de distancia VL53L0X, un carrito o coche de juguete, un perfil de aluminio o de madera para la rampa, una protoboard, cables Dupont y una powerbank o cables USB para alimentación.

Conectad el sensor VL53L0X siguiendo el esquema del capítulo 2. Las conexiones I²C son las siguientes: el pin VIN del sensor a 3,3 V o 5 V de la placa, GND a GND, SCL al pin GPIO22 (SCL del ESP32) y SDA al pin GPIO21 (SDA del ESP32).

Se ha de instalar la librería del sensor VL53L0X y la librería Phyphox BLE (phyphoxBle.h), tal como se describe en el capítulo del IDE de Arduino.



Ahora, colocad el sensor en un extremo del plano inclinado, de modo que el haz láser apunte hacia el carrito.

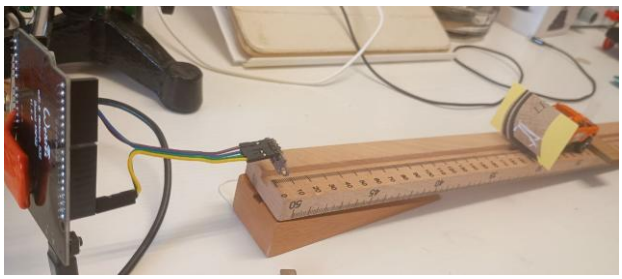
Aseguraros de que el carrito se desplace dentro del rango de medida del sensor (de 4 cm a 2 m) y que no haya obstáculos entre el sensor y el carrito.

Es conveniente pegar al carrito una pequeña tarjeta de cartón o plástico para favorecer la reflexión del haz infrarrojo que emite el sensor.

Procedimiento

En primer lugar, conectad la placa al ordenador mediante un cable USB y subid el código. Abrid el monitor serie del IDE de Arduino para verificar que el sensor mide correctamente la distancia.

Después, de cargar el código y abrir la aplicación Phyphox, para obtener las distancias se deben seguir los mismos pasos que siempre, y ya se puede empezar a medir distancias, como se ve en la captura de pantalla de abajo.



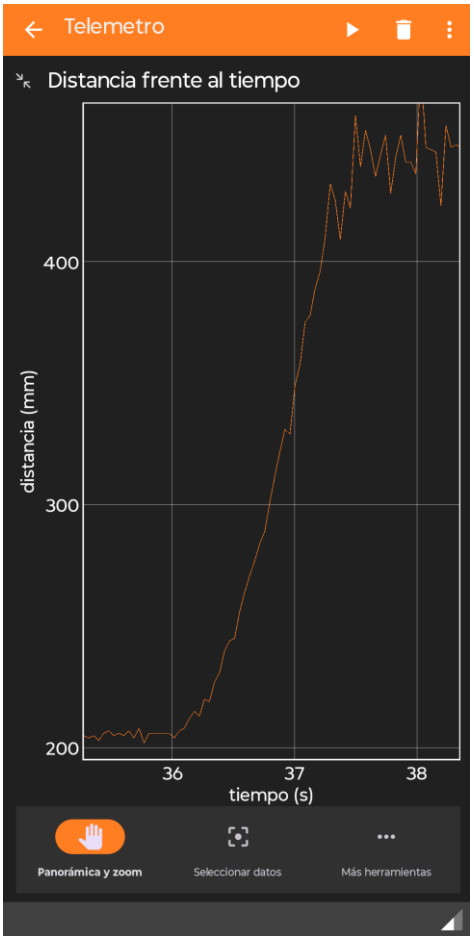
A continuación, inclinad el plano y soltad el carrito desde una posición conocida cerca del sensor. Los datos de distancia y tiempo se enviarán automáticamente a la aplicación Phyphox en su móvil a través de Bluetooth. Repetid la medida para diferentes ángulos de inclinación del plano.

Resultados esperados

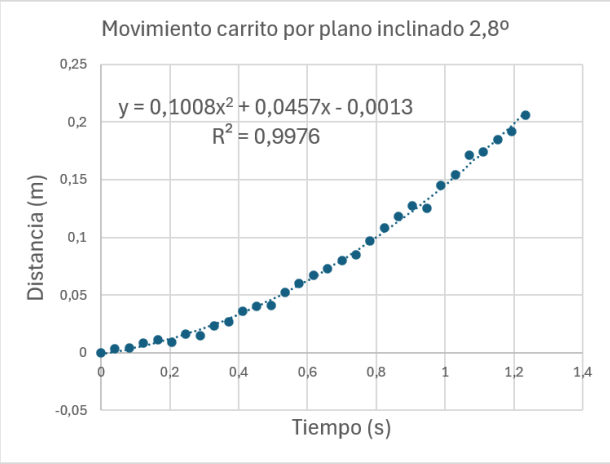
Al dejar caer el carrito por el plano inclinado, se obtiene en Phyphox una gráfica de distancia frente al tiempo. Dado que el movimiento es uniformemente acelerado, la gráfica será una parábola. La distancia aumentará a medida que el carrito se aleje del sensor en el extremo del plano.

A partir de los datos de distancia y tiempo, se puede calcular la velocidad media entre dos instantes consecutivos mediante $v = \Delta s / \Delta t$.

Alternativamente, se pueden exportar los datos a Excel (por ejemplo) y ajustar la curva s(t) a una función cuadrática $s(t) = At^2 + Bt + C$, donde $A = a/2$, $B = v_0$ y $C = s_0$.



Phyphox con el sensor de distancia VL53L0X



Para un plano inclinado sin rozamiento, la aceleración teórica es a = g sinθ, donde θ es el ángulo de inclinación. Comparando el valor experimental obtenido del ajuste con el valor teórico se puede estimar el efecto del rozamiento y evaluar la precisión del experimento.

Ampliación

Repetid el experimento variando el ángulo de inclinación y representad la aceleración obtenida en función del seno del ángulo. La pendiente de la recta de regresión debería ser g (9,81 m/s²). También se puede probar con diferentes masas del carrito para verificar que la aceleración es independiente de la masa en ausencia de rozamiento apreciable.

Código

```
/*
Medida de distancia con sensor VL53L0X (Adafruit).
Placa Wemos D1 R32 (ESP32). Seleccionar ESP32 DevModule.
Conexiones del sensor: GND a GND, VCC a 3,3 V o 5 V, SCL a SCL, SDA a SDA
Los datos se visualizan en el monitor serie del IDE de Arduino y en la aplicación Phyphox en el móvil a través de Bluetooth.

Se han de instalar las librerías phyphoxBle y Adafruit_VL53L0X.
*/

#include <phyphoxBle.h> // Librería Phyphox BLE
#include <Wire.h> // Para comunicación I2C
#include <Adafruit_VL53L0X.h> // Librería del sensor VL53L0X de Adafruit

Adafruit_VL53L0X lox = Adafruit_VL53L0X();

void setup() {
  Serial.begin(9600);  // Iniciar monitor serie

  // Iniciar Bluetooth BLE con nombre "Telemetro"
  PhyphoxBLE::start("Telemetro");

  // Configurar experimento en Phyphox
  PhyphoxBleExperiment experimento;
  experimento.setTitle("Telemetro");
  experimento.setCategory("Arduino Experiments");
  experimento.setDescription("Mide la distancia por tiempo de vuelo de infrarrojos");

  PhyphoxBleExperiment::View view;

  // Elemento para mostrar el valor numérico de la distancia
  PhyphoxBleExperiment::Value value;
  value.setLabel("Distancia actual");
  value.setPrecision(0);
  value.setUnit("mm");
  value.setColor("FFFFFF");
  value.setChannel(2); // Canal 2: distancia

  // Elemento para mostrar la gráfica
  PhyphoxBleExperiment::Graph graph;
  graph.setLabel("Distancia frente al tiempo");
  graph.setUnitX("s");
  graph.setUnitY("mm");
  graph.setLabelX("tiempo");
  graph.setLabelY("distancia");
  graph.setChannel(1, 2);  // Canal 1: tiempo, Canal 2: distancia

  view.addElement(value);
  view.addElement(graph);
  experimento.addView(view);
  PhyphoxBLE::addExperiment(experimento);

  // Inicializar sensor VL53L0X
  Serial.println("Iniciando VL53L0X...");
  if (!lox.begin()) {
    Serial.println("Error al iniciar el sensor VL53L0X. Revisa las conexiones");
    while(1);
  }
  Serial.println("VL53L0X iniciado correctamente. Empezando a medir...");
}
```

```
void loop() {
  VL53L0X_RangingMeasurementData_t measure;

  // Realizar lectura de distancia
  lox.rangingTest(&measure, false);

  float t = 0.001 * (float)millis(); // Tiempo en
segundos
  float distance = 0;

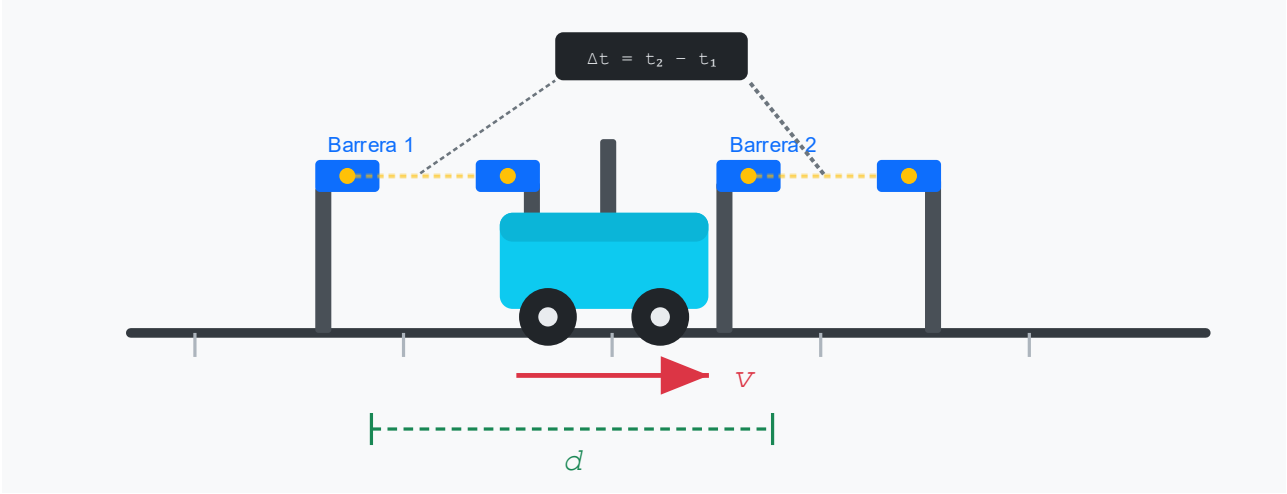
  // Si la lectura es válida (estado != 4 indica error
de fase/fuera de rango)
  if (measure.RangeStatus != 4) {
    distance = measure.RangeMilliMeter;  //
Distancia en mm
  } else {
    // Si el sensor no puede determinar la distancia o
está fuera de rango
    distance = 0;
  }

  // Enviar datos a Phyphox
  PhyphoxBLE::write(t, distance);

  // Mostrar por monitor serie (formato:
tiempo;distancia)
  Serial.print(t, 3);
  Serial.print("; ");
  Serial.println(distance);

  PhyphoxBLE::poll();  // Necesario para mantener
la comunicación BLE
}
```

Movimiento de carritos con barreras ópticas



En este experimento se trata de medir la velocidad de un carrito o una bola que se desplaza sobre una superficie plana horizontal o inclinada utilizando una barrera de luz (fotopuerta) y analizar los diferentes tipos de movimiento, uniforme y uniformemente acelerado.

Existen fotopuertas comerciales con las que tradicionalmente se hacía el estudio de los movimientos de bolas y carritos (Amador, 2025, The Vernier blog, 2026), pero aquí las fabricaremos con leds infrarrojos y fototransistores.



Fundamento físico

Una barrera de luz consiste en un emisor (LED infrarrojo) y un receptor (fototransistor) enfrentados. Cuando un objeto interrumpe el haz, el microcontrolador detecta un cambio en la señal. Si conocemos la longitud del objeto (o una banda opaca colocada sobre él), podemos calcular la velocidad en ese punto como $v = \Delta x/\Delta t$, donde Δx es la longitud del objeto y Δt el tiempo que permanece interrumpido el haz.

Colocando dos barreras separadas una distancia conocida, podemos medir la velocidad en dos puntos y calcular la aceleración media:

$$a = (v_2^2 - v_1^2)/(2d).$$

Materiales y montaje experimental

Necesitaréis una placa Wemos D1 R32 (ESP32) o Arduino UNO, dos pares de LED IR + fototransistor (para las barreras de luz), dos resistencias de 220 Ω (para los LED) y dos de 10 kΩ (para los fototransistores), una protoboard, cables Dupont, una superficie plana o carril de laboratorio, un carrito con una banda opaca de longitud conocida (por ejemplo, 10 cm) y un ordenador con el IDE de Arduino.

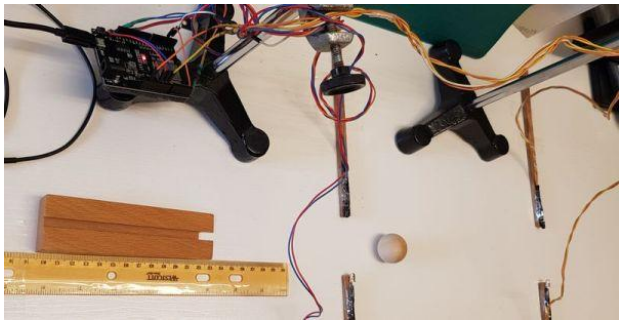
Para más detalles sobre las barreras de luz, consultad el capítulo de los sensores.

Montad dos barreras de luz separadas una distancia conocida, d, (por ejemplo, 50 cm) a lo largo del carril. Cada barrera consiste en un LED IR con resistencia de 220 Ω en serie conectado a 5 V y GND, y un fototransistor con resistencia de 10 kΩ en configuración pull-down, conectado a un pin digital.

Conectad el primer fototransistor a D2 y el segundo a D3 y aseguraros de que los LED y fototransistores estén perfectamente alineados y de que el carrito pase entre ellos sin tocarlos.

Procedimiento

Se conecta la placa y se carga el código. Podéis realizar tres tipos de experimentos:
1 De movimiento uniforme, impulsando el carrito o la bola sobre una superficie horizontal y midiendo el tiempo de paso por cada barrera.



2 De movimiento acelerado, dejando caer el carrito o la bola por una rampa.
3 De movimiento acelerado con fuerza constante si se ata un hilo al carrito, pasándolo por una polea y colgando una pesa del otro extremo.

Resultados esperados

Para movimiento uniforme, $v_1 \approx v_2$ y la aceleración será aproximadamente 0. Para movimiento acelerado (rampa o pesa), $v_2 > v_1$ y la aceleración será constante. Con una inclinación de 10° en la rampa, la aceleración teórica es $g \cdot \text{sen}(10°) \approx 1{,}70$ m/s².

Calculad la velocidad media y la aceleración para cada configuración. Representad la velocidad frente al tiempo si realizáis medidas con diferentes separaciones entre barreras.

Comparad la aceleración medida con la teórica (para la rampa, $a = g \cdot \text{sen}\ \theta$, y para la pesa, $a = m_{pesa} \cdot g/(m_{carrito} + m_{pesa})$. Calculad los errores relativos.

Ampliación

Añadid una tercera barrera para tener tres medidas de velocidad y verificar la aceleración constante.

Puede usar también una barrera simple conectada a un cronómetro para medir el período de un péndulo o el tiempo de caída libre.

Código para un carrito de 10 cm de largo

```
// Medida de velocidad de un carrito de 10 cm con dos barreras de luz
// Fototransistor 1 a D2, Fototransistor 2 a D3

#define BARRERA1 2
#define BARRERA2 3
#define LONGITUD_BANDA 0.10  // Los 10 cm del carrito en metros

void setup() {
  Serial.begin(9600);
  pinMode(BARRERA1, INPUT);
  pinMode(BARRERA2, INPUT);
}

void loop() {
  // Esperar a que el carrito interrumpa la primera barrera
  while (digitalRead(BARRERA1) == HIGH) {}
  unsigned long t1_inicio = micros();
  while (digitalRead(BARRERA1) == LOW) {}
  unsigned long t1_fin = micros();

  // Esperar a que interrumpa la segunda barrera
  while (digitalRead(BARRERA2) == HIGH) {}
  unsigned long t2_inicio = micros();
  while (digitalRead(BARRERA2) == LOW) {}
  unsigned long t2_fin = micros();

  // Calcular velocidades
  float v1 = LONGITUD_BANDA / ((t1_fin - t1_inicio) / 1000000.0);
  float v2 = LONGITUD_BANDA / ((t2_fin - t2_inicio) / 1000000.0);
  float distancia = 0.50;  // metros entre barreras

  Serial.print("v1;");
  Serial.print(v1, 3);
  Serial.print(";v2;");
  Serial.print(v2, 3);
  Serial.print(";a;");
  Serial.println((v2*v2 - v1*v1)/(2*distancia), 3);

  delay(1000);
}
```

Código para bolita

```
//Tiempo en pasar entre dos barreras de luz
// Fototransistor 1 a 12, Fototransistor 2 a 13
```

```
int sensor1 = 12; // se define el fototransistor 1 en el
pin 12
int sensor2 = 13; // se define el fototransistor 2 en el
pin 13

unsigned long t1 = 0; //se define el tiempo al pasar
por el primer sensor
y se le asigna de entrada el valor 0
unsigned long t2 = 0; // tiempo al pasar por el
segundo sensor
unsigned long t12 = 0;

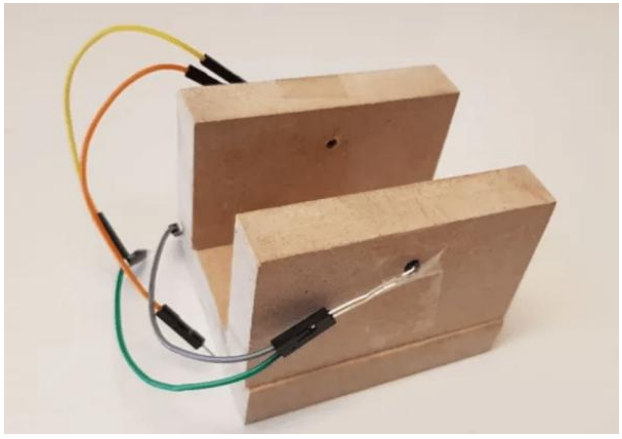
void setup() {
Serial.begin(9600);
pinMode(sensor1,INPUT); // se define el sensor1
como de entrada
pinMode(sensor2,INPUT);
}

void loop() {
while(digitalRead(sensor1) == LOW); // espera que
pase por el 1er sensor. Dejará de recibir luz
infrarroja
t1 = micros(); // guarda el tiempo en useg
(microsegundos)
while(digitalRead(sensor2) == LOW); // espera
hasta que pase por el 2do sensor
t2 = micros();
// Envía por el puerto serie los valores de t1, t2 y
t12
Serial.print("\nT1: ");
Serial.print(t1);
Serial.print("\nT2: ");
Serial.print(t2);
t12 = t2 - t1;
Serial.print("\nTiempo total: ");
Serial.print(t12);
Serial.print(" useg");
}
```

Medida de la aceleración de la gravedad

En esta ocasión se pretende medir la aceleración de la gravedad de caída libre de un cuerpo utilizando barreras de luz (fotopuerta), como en el caso anterior del movimiento de los carritos, y/o con sensores de proximidad.

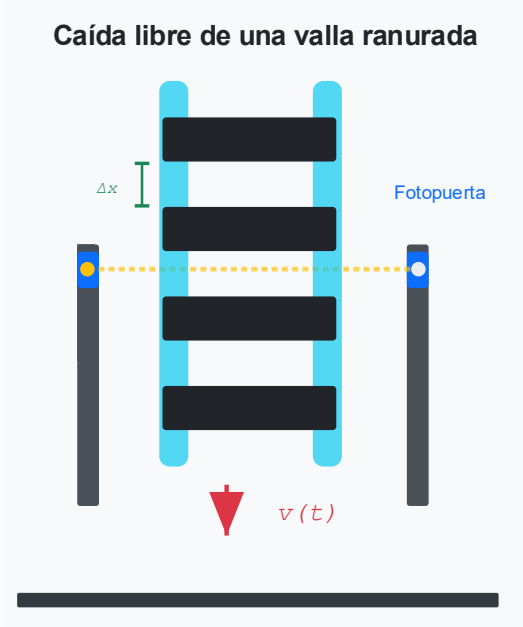
La aceleración de la gravedad (g) es una constante física fundamental a cuya medida con precisión es un ejercicio clásico en los laboratorios de física, que tradicionalmente se realiza con péndulos, como veremos en el siguiente experimento, o cuerpos en caída libre cronometrados con fotopuertas, por ejemplo.



Montaje de una fotopuerta con un led y un fototransistor IR enfrentados

Esto último ya lo propuse en una entrada de mi blog (Ramírez, 2023b) donde utilizaba, como veremos aquí a continuación, una barrera de luz formada por un led IR enfrentado a un fototransistor trabajando ambos en el infrarrojo (940 nm), dejando caer entre los dos una valla con franjas alternas opacas y transparentes, lo que no era sino hacer lo mismo que en *Movimiento de caída de una regla* (Aparicio y Lozano, 2003, Physics with Vernier, 2026) en el que se utilizaba un datalogger, o en *Estudio de movimientos con fotocélula* (Ramírez, 2017) que se hacía con un teléfono móvil y una fotorresistencia.

En este experimento también propongo un método un poco diferente al de la barrera de luz IR que combina la misma valla, pero ahora con un sensor de barrera óptica tipo Flying-fish. La valla se deja caer verticalmente ante el sensor que tiene, tanto el led IR como el fototransistor IR, uno al lado del otro, de manera que si pasa un objeto por delante suyo la luz infrarroja emitida por el led se refleja y es recogida por el fototransistor. Cada vez que una franja transparente pasa a ser opaca pasando por delante del sensor, la situación cambia de estado. Registrando los instantes de cada cambio, podemos reconstruir la ley del movimiento de caída uniformemente acelerado y extraer el valor de g.



La técnica que propongo aquí presenta dos ventajas respecto a la de la barrera de luz. Por un lado el montaje es mucho más simple ya que no hace falta más que comprar el sensor (no hay que construirlo a partir de sus elementos) y es de bajo coste (3 €), y no hace falta más que un soporte. Por otra el software consiste en un sketch (programa para el IDE de Arduino) un poco más elaborado que el de la barrera, que ahora incorpora un filtro antirrebote (debounce) lo que evita falsos cambios de estado, y además funciona en un microcontrolador ESP32 (no sólo con un Arduino).

Fundamento físico

Un cuerpo (la valla, en nuestro caso) en caída libre cerca de la superficie terrestre, despreciando la resistencia del aire, se mueve con aceleración constante g. Si elegimos un eje y vertical hacia abajo, la posición en función del tiempo viene dada por:

$$y(t) = y_0 + v_0*t + 1/2\ g*t^2$$

donde y_0 es la posición inicial y v_0 la velocidad inicial a la que llega la valla al sensor.

El objeto que cae será una valla que tiene una sucesión de franjas opacas y transparentes de igual anchura, o no, que deben medirse lo mejor posible. Cuando la valla cae, cada transición opaca-transparente (o transparente-opaca) sucede cuando el borde de una franja coincide con el haz del sensor. Si se mide el tiempo en el que se produce cada transición se puede graficar las sucesivas distancias recorridas por la valla en su caída ante el tiempo y encontrar la ecuación del movimiento, de donde obtendremos el valor de g (la aceleración de la caída). Este será el procedimiento de análisis que seguiremos después de capturar los datos experimentales.



Aunque la anchura de las franjas sea igual los intervalos de tiempo entre transiciones consecutivas no serán constantes, sino que disminuirán a medida que la valla acelere.

Materiales y montaje experimental con fotopuerta

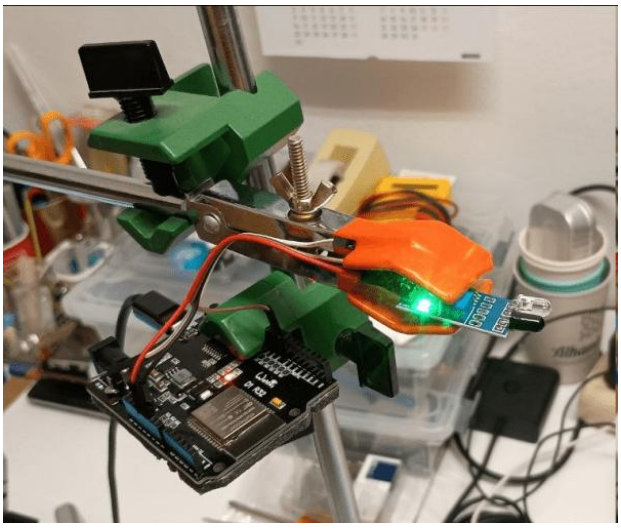
Sólo con una fotopuerta se pueden realizar varios experimentos. Por ejemplo se puede medir la velocidad a la que pasa una bola o un cochecito por el punto en el que está el sensor. Si es una bola que se mueve por una mesa y cae, se puede predecir la distancia a la que tocará en el suelo.

Si es una valla con franjas transparentes y opacas se puede dejar caer, como la de cartón en la foto de al lado y obtener el tiempo de paso de cada franja, lo que permite estudiar su movimiento y calcular la aceleración de la gravedad, como proponían Aparicio y Lozano (2003a y b) utilizando Multilog en *Movimiento de caída de un regle*.

En la misma foto se ve que se hace pasar la valla por una barrera de madera en la que los sensores están empotrados en unos agujeros en lados enfrentados hechos con un taladro, y así quedan fijos.

Materiales y montaje experimental con sensor de proximidad

Fije el sensor a un soporte (por ejemplo, una pinza de laboratorio o un bloque de madera) de manera que el haz infrarrojo quede horizontal. El sensor debe colocarse de manera que lo hiciera incida en la valla perpendicularmente, a una distancia constante durante toda la caída.



Aquí utilizaremos, por eso, un sensor tipo Flying-fish como el que se ve en la fotografía anterior.

Como ya he comentado, la valla de franjas actúa como un "obstáculo móvil" que alterna estados transparente (haz no interrumpido, y que se pierde) y opaco (haz interrumpido, y que se refleja). Coloque la valla de manera que pueda caer verticalmente justo delante del sensor, a una distancia de un par de centímetros del frente del encapsulado (para garantizar que las franjas opacas interrumpan completamente el haz), y que caiga libremente, sin frotar el sensor ni otros obstáculos. Puede sujetarse con los dedos por su extremo superior y dejarla ir limpiamente.

Materiales necesarios
Placa ESP32 (cualquiera) y sensor Flying-fish o KY-032 IR .

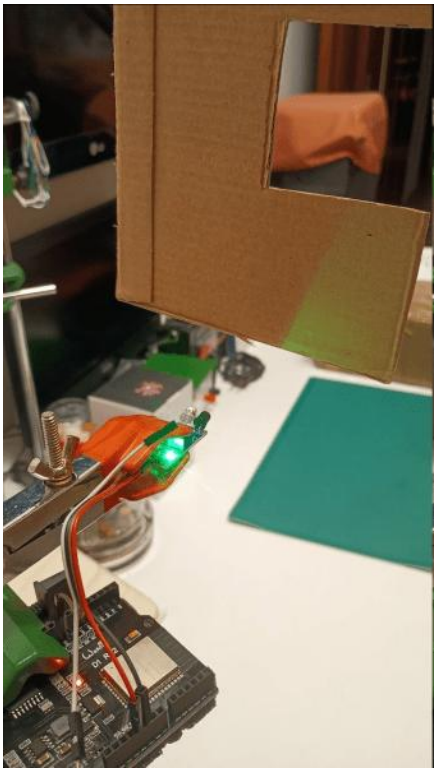
Valla impresa o dibujada y recortada con al menos tres franjas opacas y dos transparentes, cada una de anchura conocida, mejor si son iguales, de un mínimo de 2 cm. La valla se imprime en papel o cartón, con anchos idénticas (o no) para las zonas opacas y transparentes (por ejemplo, a L = 4,0 cm). Si es ligera fuera convenían pegarle un par de agujas de extender la ropa a la par inferior para minimizar el efecto del rozamiento con el aire.

Soporte rígido (tres pies y pinza) para sujetar el sensor de manera que la valla pueda caer libremente delante de él. También cables dupont para las conexiones.

Procedimiento

Una vez subido el sketch al ESP32, si se abre el terminal serie del Arduino configurado en 115200 bps, veréis que muestra el mensaje «SISTEMA INICIALIZADO – Esperando primera transición...». Entonces, dejad ir la valla sin imprimirle velocidad lateral ni rotación.

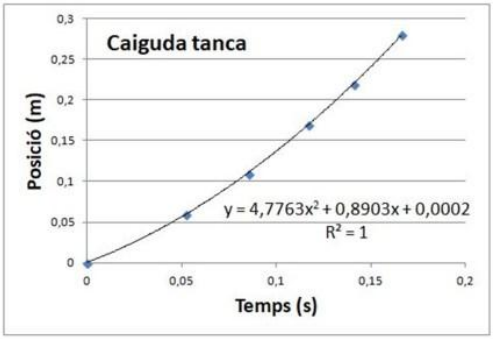
El sensor detectará la primera transición (cerca de la primera franja) y enviará por serie la línea correspondiente. La caída continuará y se imprimirán los datos de cada transición sucesiva y cuando la valla haya pasado completamente por el sensor podéis recoger los datos impresos en el terminal. Ahora, se copian los valores en una hoja de cálculo o un programa de tratamiento gráfico de datos y se analizan.



Resultados esperados en el caso de la fotopuerta

Dejando caer libremente la valla de cartón a través de la barrera de luz, y utilizando una placa ESP32 con el código que indico más abajo he obtenido los valores del costado de la gráfica. Considerando que la valla corta y deja pasar la luz a 0, 6, 11, 17, 22 y 28 cm se puede dibujar la gráfica posición/tiempo y obtener la ecuación de la línea de tendencia.

```
T1: 976436206
T2: 976488120
Temps total: 51914 µseg
T1: 976520898
T2: 976552872
Temps total: 31974 µseg
T1: 976576983
T2: 976601914
Temps total: 24931 µseg
```



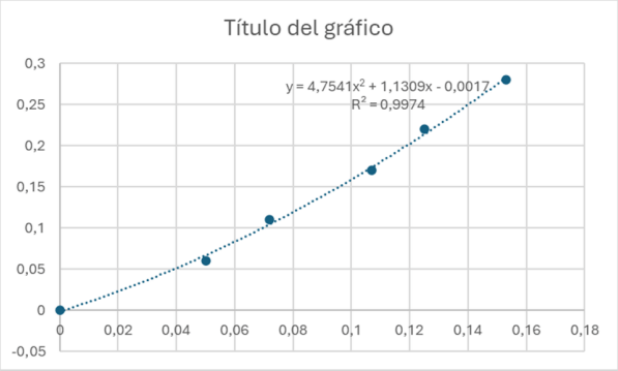
La ecuación muestra una aceleración de caída de a = 2×4,776 = 9,55 m/s^2, lo que es un valor muy aceptable para la aceleración de la gravedad.

Resultados esperados en el caso del sensor de proximidad

Pueden ser, por ejemplo, los de un caso real en el que las franjas no tienen la misma anchura:

Transición	Tiempo (ms)	Tiempo (s)	Distancia (cm)	Distancia (m)
1	0	0	0	0
2	50	0,05	6	0,06
3	72	0,072	11	0,11
4	107	0,107	17	0,17
5	125	0,125	22	0,22
	153	0,153	28	0,28

Si ahora graficamos en una hoja de cálculo la distancia en función del tiempo se obtiene la gráfica de abajo en la que está representada la línea de tendencia, la ecuación a la que se ajusta y el valor de la calidad del ajuste, R^2.



Como se ve, aquí la aceleración de la gravedad medida es g = 4,754 x 2 = 9,51 m/s^2, que tampoco está mal. Aquí las principales fuentes de error podrían ser:

- Error sistemático en la medida de las longitudes ya que con una regla milimetrada se puede ajustar a 0.5 mm, como mucho.
- Errores aleatorios en los tiempos. La resolución de 'millis()' es de 1 ms, pero la incertidumbre en la detección de las esquinas puede ser similar al tiempo de eliminación de rebotes (debounce) (10 ms).
- Desviación de la caída libre ideal: rozamiento con el aire, posible rozamiento lateral con el soporte, velocidad inicial no nula., etc.

Por lo tanto, es posible medir la aceleración de la gravedad con materiales de bajo coste y fáciles de montar con el tradicional método de la valla con franjas que a mí tanto me gusta.

El código para el ESP32

Para subir el sketch a la placa ESP32 abrid un sketch al IDE de Arduino y pegad el código del final), seleccionar la placa ESP32 Dev Module (genérica ESP32) y pulsar el botón de subir (flecha hacia la derecha). Después de que se cargue, abrid el Monitor Serie (115200 bps) para ver los mensajes de inicio.

Hay veces que el IDE de Arduino, una vez que ha compilado el código, da error al intentar cargarlo en la placa. En este caso desconectad del ESP32 el pin de salida de datos del sensor y volved a intentar cargar el sketch. Una vez cargado, volved a conectar el sensor al pin 12 (el que hemos escogido).

El sketch dispone de un filtro antirrebote (debounce) de manera que el código ignora cualquier cambio que dure menos de 'DEBOUNCE_MS' (10 ms), un tiempo suficiente para eliminar ruido sin perder transiciones leídas, ya que sino, pueden aparecer cambios falsos.

La función millis() proporciona el número de milisegundos transcurridos desde el arranque del ESP32 (se podría haber utilizado micros()). La diferencia entre dos llamadas a millis() proporciona el intervalo con una precisión de ±1 ms, más que suficiente para intervalos de caída libre que suelen ser de decenas de milisegundos. Los datos de cada transición se imprimen en dos formatos: una línea con los valores pelados y una línea descriptiva de los tiempos transcurridos.

Código

```
/*
SKETCH PARA ESP32 CON SENSOR FLYINGFISH - DETECCIÓN DE VALLAS OPACAS/TRANSPARENTES
Descripción física del experimento: Un sensor Flyingfish (barrera óptica) montado en el pin GPIO12 del ESP32 detecta el paso de una valla que alterna zonas transparentes y opacas de igual longitud.
Cada cambio de opacidad (transparente → opaca, opaca → transparente) produce una transición en la señal digital del sensor.
```

```
Este sketch mide el tiempo absoluto (ms) desde la primera transición en que pasa cada transición y el intervalo de tiempo (ms) entre transiciones consecutivas.
Los datos se envían por el puerto serie en formato CSV para su análisis posterior.
 */
// CONFIGURACIÓN DEL HARDWARE (EDITAR SEGÚN EL MONTAJE REAL)
const uint8_t PIN_SENSOR = 12; // Pin digital donde está conectado el sensor
const bool SENSOR_ACTIVE_STATE = LOW;  // Estado lógico cuando el sensor no detecta su propia luz (HIGH = luz detectada, LOW = obstáculo)
const unsigned long DEBOUNCE_MS = 10; // Tiempo de estabilización para evitar falsas transiciones
const unsigned long SERIAL_BAUD = 115200; // Velocidad del puerto serie (bits por segundo)

// VARIABLES GLOBALES (ESTADO DEL EXPERIMENTO)
bool lastStableState; // Último estado estable del sensor (después del debounce)
bool lastRawState; // Última lectura cruda (para detección de cambios)
unsigned long lastDebounceTime = 0; // Momento en que se inició el período de debounce
unsigned long experimentStartTime = 0; // Tiempo de la primera transición (ms desde el inicio del sketch)
unsigned long lastTransitionTime = 0; // Tiempo de la transición anterior (ms desde experimentStartTime)
unsigned int transitionCount = 0; // Número total de transiciones detectadas
bool experimentStarted = false; // Bandera que indica si ya ha ocurrido la primera transición

// PROTOTIPOS DE FUNCIONES
void printTransitionData(unsigned long currentTime, unsigned long interval);
void sendHeader();

//SETUP: INICIALIZACIÓN DEL SISTEMA
void setup() {
  // 1. Inicializar comunicación serie para depuración y captura de datos
  Serial.begin(SERIAL_BAUD);
  while (! Serial) {
    ; // Esperar a que el puerto esté listo (en placas con USB nativo)
  }
  // 2. Configurar el pin del sensor como entrada digital
  pinMode(PIN_SENSOR, INPUT);
  // 3. Leer el estado inicial del sensor y almacenarlo como referencia
  lastRawState = digitalRead(PIN_SENSOR);
  lastStableState = lastRawState;
  // 4. Enviar cabecera explicativa por el puerto serie
  sendHeader();
  // 5. Informar por serie que el sistema está listo
  Serial.println("SISTEMA INICIALIZADO");
  Serial.println("Esperando primera transición...");
  Serial.println();
}
// BUCLE PRINCIPAL DE DETECCIÓN
void loop() {
  // Leer el estado actual del sensor (nivel lógico HIGH/LOW)
  bool currentState = digitalRead(PIN_SENSOR);

  // Detectar un cambio crudo (sin debounce) respecto a la última lectura
  if (currentState != lastRawState) {
  // Reiniciar el temporizador de debounce
    lastDebounceTime = millis();
    lastRawState = currentState;
  }

  // Comprobar si ha pasado el tiempo desde el último cambio crudo
  if ((millis() - lastDebounceTime) >= DEBOUNCE_MS) {
// El estado ya está estable; verificar si difiere del estado estable anterior
    if (currentState != lastStableState) {
  // ¡Se ha producido una transición válida (flanco)!
      lastStableState = currentState;
// Obtener el tiempo actual absoluto (ms desde el arranque del microcontrolador)
      unsigned long now = millis();
// Gestionar la primera transición (inicio del experimento)
      if (!experimentStarted) {
        experimentStarted = true;
        experimentStartTime = now;
        lastTransitionTime = now;
        transitionCount = 1;

Serial.println("=========================");
Serial.println("PRIMERA TRANSICIÓN DETECTADA");
```

```
Serial.println("Inicio del cronometraje experimental.");
Serial.println("=========================");
Serial.println();
Serial.println("Transición, Tiempo desde inicio (ms), Interval (ms)");
// La primera transición tiene intervalo cero (no hay anterior)
printTransitionData(0, 0);
 } else {
// Para transiciones posteriores, calcular tiempos relativos
      unsigned long timeSinceStart = now - experimentStartTime;
      unsigned long interval = now - lastTransitionTime;
//Actualizar el instante de la última transición
      lastTransitionTime = now;
      transitionCount++;

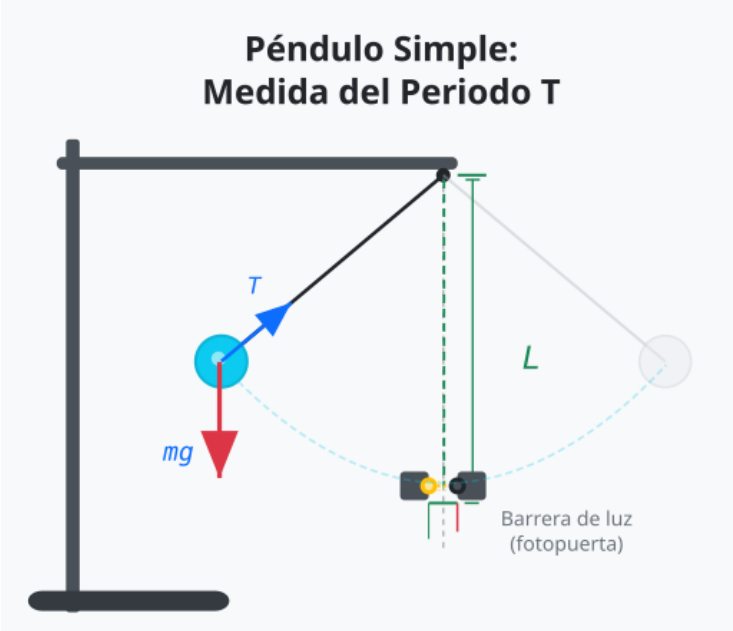
// Enviar los datos de la transición por el puerto serie
      printTransitionData(timeSinceStart, interval);
    }
   }
  }
}
// ENVÍO DE CABECERA EXPLICATIVA
void sendHeader() {
  Serial.println("=========================");
  Serial.println("EXPERIMENTO: Valla opaca/transparente con sensor Flyingfish");
  Serial.println("Microcontrolador: ESP32");
  Serial.println("Pin del sensor: GPIO" + String(PIN_SENSOR));
  Serial.println("Estado activo del sensor: " + String(SENSOR_ACTIVE_STATE == HIGH ? " HIGH (luz)" : "LOW (obstáculo)"));
  Serial.println("Debounce: " + String(DEBOUNCE_MS) + " ms");

  Serial.println("=========================");
  Serial.println();
  Serial.println("OBJETIVO:");
  Serial.println(" - Registrar el instante de cada transición (opaco↔transparente).");
  Serial.println(" - Medir el intervalo entre transiciones consecutivas.");
  Serial.println(" - Los tiempos se expresan en milisegundos (ms).");
  Serial.println();
  Serial.println("FORMATO DE DATOS (CSV):");
  Serial.println(" Transición, Tiempo desde inicio (ms), Intervalo (ms)");
  Serial.println("=========================");
}
// IMPRIMIR DATOS DE UNA TRANSICIÓN
void printTransitionData(unsigned long timeSinceStart, unsigned long interval) {
// Formato CSV simple: número de transición, tiempo desde inicio, intervalo
  Serial.print(transitionCount);
  Serial.print(", ");
  Serial.print(timeSinceStart);
  Serial.print(", ");
  Serial.println(interval);

// Para mayor claridad, también se puede imprimir una línea descriptiva (opcional, comentar si sólo se desea CSV puro)
  Serial.print(" -> Transición ");
  Serial.print(transitionCount);
  Serial.print(" a los ");
  Serial.print(timeSinceStart);
  Serial.print(" ms (interval: ");
  Serial.print(interval);
  Serial.println(" ms)");
}
```

Cálculo de g con un péndulo

Vamos a calcular el valor de la aceleración de la gravedad a través de determinar experimentalmente el periodo de oscilación de un péndulo simple y a comprobar su dependencia con la longitud del hilo y la independencia de la masa. Para medir el tiempo de oscilación con precisión utilizaremos una barrera de luz (fotopuerta) conectada a Arduino.



Este experimento se ha propuesto infinidad de veces, tanto de forma tradicional midiendo tiempos con un cronómetro (CDECT, 2002b), como midiéndolos con una fotopuerta (Departamento de Física, 2017) o con Arduino (Adiati, R.F. y otros, 2025), así que aquí hay poca novedad.

Fundamento físico

Un péndulo simple consiste en una masa puntual suspendida de un hilo inextensible y sin masa. Cuando se separa de su posición de equilibrio y se suelta, oscila bajo la acción de la gravedad. Para pequeñas amplitudes (menores de 15º), el movimiento es armónico simple y el periodo de oscilación viene dado por:

$$T = 2\pi\sqrt{\frac{L}{g}}$$

donde L es la longitud del hilo y g la aceleración de la gravedad. Es importante destacar que el periodo es independiente de la masa del péndulo y solo depende de la longitud. Elevando al cuadrado obtenemos una relación lineal:

$$T^2 = \frac{4\pi^2}{g} \cdot L$$

que permite calcular g experimentalmente a partir de la pendiente de la recta T^2 frente a L.

Materiales y montaje experimental

Necesitaréis una placa Arduino UNO o Wemos D1 R32 (ESP32), una barrera de luz formada por un LED infrarrojo y un fototransistor IR, hilo y una masa (arandela o tuerca), un soporte para sujetar el péndulo, una protoboard, resistencias de 220 Ω y 10 kΩ, y cables Dupont.

Montad el circuito de la barrera de luz siguiendo las indicaciones del capítulo de los sensores. Conectad el LED IR (ánodo a 5 V a través de una resistencia de 220 Ω, cátodo a GND) y el fototransistor IR (colector a 5 V a través de una resistencia de 10 kΩ, emisor a GND). La unión entre el colector y la resistencia del fototransistor se conecta a una entrada digital de la placa, por ejemplo el pin 2.

Colocad la barrera de luz de modo que el péndulo, al oscilar, interrumpa el haz en el punto más bajo de su trayectoria. La masa del péndulo debe pasar justo entre el LED y el fototransistor. Aseguraos de que el hilo cuelgue verticalmente en reposo y que la masa pueda oscilar libremente sin obstáculos.

Procedimiento

En primer lugar, conecte la placa al ordenador y subid el código. Abrid el monitor serie para verificar que la barrera de luz detecta correctamente el paso del péndulo. Separad el péndulo de la vertical un ángulo pequeño (menos de 15º) y soltadlo. El programa registrará el tiempo entre dos pasos

consecutivos por el punto más bajo y mostrará el periodo en el monitor serie.

Medid el periodo para al menos cinco longitudes diferentes del hilo (por ejemplo, 0,25 m, 0,50 m, 0,75 m, 1,00 m y 1,25 m). Para cada longitud, realizad al menos 10 medidas y calculad el valor promedio del periodo. Anotad los resultados en una tabla.

Resultados esperados

Para un péndulo de longitud L = 1,0 m, el periodo teórico es:

$$T = 2\pi\sqrt{\frac{1{,}0}{9{,}81}} \approx 2{,}01\ \text{s}$$

La siguiente tabla muestra los valores esperados orientativos:

Longitud (m)	Periodo teórico (s)	T^2 (s^2)
0,25	1,00	1,01
0,50	1,42	2,01
0,75	1,74	3,02
1,00	2,01	4,03
1,25	2,24	5,04

Representad gráficamente T^2 frente a L. Los puntos deberían ajustarse a una línea recta cuya pendiente es $4\pi^2/g$. Despejando:

$$g = \frac{4\pi^2}{\text{pendiente}}$$

Calculad el valor experimental de g y comparadlo con el valor teórico de 9,81 m/s². Determinad el error relativo y discutid las posibles fuentes de error: rozamiento del aire, amplitud inicial no infinitesimal, imprecisión en la medida de la longitud del hilo, etc.

Ampliación

Repetid el experimento con masas diferentes (manteniendo la misma longitud) para comprobar que el periodo no depende de la masa. Utilizad ángulos de separación mayores de 15º para observar la desviación del comportamiento ideal del péndulo simple.

Código

```
/*
Medida del periodo de un péndulo con barrera de luz
Placa: Arduino UNO o Wemos D1 R32 (ESP32)

La barrera de luz detecta el paso del péndulo.
El periodo se calcula como el tiempo entre dos interrupciones consecutivas
del haz en el mismo sentido.

Para la instalación de librerías, ver Capítulo 3 (IDE de Arduino).
*/

const int barreraPin = 2;   // Pin digital para la barrera de luz
unsigned long tiempoAnterior = 0;
unsigned long tiempoActual = 0;
float periodo = 0;
bool bloqueado = false;

void setup() {
  Serial.begin(9600);
  pinMode(barreraPin, INPUT);
  Serial.println("Iniciando medicion del pendulo...");
  Serial.println("Tiempo(ms);Periodo(s)");
}

void loop() {
  int estado = digitalRead(barreraPin);

  // Detectar flanco descendente (haz interrumpido -> LOW)
  if (estado == LOW && !bloqueado) {
    bloqueado = true;

    // Calcular periodo si no es la primera interrupción
    tiempoActual = millis();
    if (tiempoAnterior > 0) {
      unsigned long diferencia = tiempoActual - tiempoAnterior;
      periodo = diferencia / 1000.0;  // Convertir a segundos

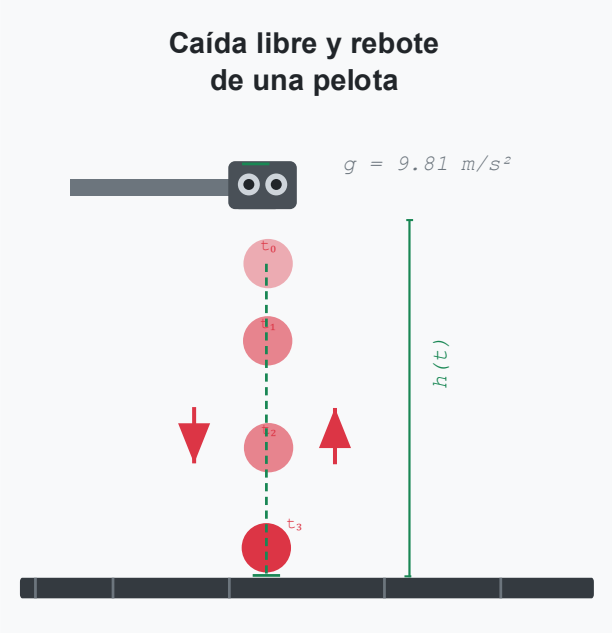
      // Mostrar datos
      Serial.print(tiempoActual);
      Serial.print(";");
      Serial.println(periodo, 4);  // 4 decimales para precisión
    }
    tiempoAnterior = tiempoActual;
  }
```

```
  // Detectar flanco ascendente (haz recuperado ->
HIGH)
  if (estado == HIGH && bloqueado) {
    bloqueado = false;
  }

  delay(1);  // Pequeña pausa para evitar rebotes
}
```

Caída y rebote de una pelota

Lo que se pretende en este experimento es estudiar el movimiento de caída libre de una pelota y analizar la pérdida de energía mecánica durante los sucesivos rebotes. Para ello mediremos las sucesivas posiciones de la pelota y la altura máxima que alcance después de cada impacto mediante un sensor de distancia láser VL53L0X (también se puede utilizar el ultrasónico HC-SR04).



Este es un caso que ya se ha abordado mediante sensores de posición (Aparicio y Lozano, 2002e y f), o grabación del sonido de los sucesivos choques de la pelota con el suelo (Ramírez, 2016, Phyphox, 2018)

Fundamento físico

Cuando una pelota se deja caer desde una altura inicial h_0, su energía potencial gravitatoria ($m \cdot g \cdot h_0$) se transforma en energía cinética durante la caída.

Al impactar contra el suelo, parte de la energía se disipa en forma de deformación, calor y sonido, por lo que la pelota no recupera su altura original. La relación entre la altura después del rebote (h_1) y la altura inicial (h_0) se conoce como coeficiente de restitución $e = \sqrt{(h_1/h_0)}$.

Este coeficiente es una medida de la elasticidad del impacto y depende del material de la pelota y de la superficie de rebote. En una caída libre ideal, sin resistencia del aire, la posición vertical sigue la ecuación $y(t) = h_0 - \frac{1}{2} \cdot g \cdot t^2$, donde $g = 9{,}81$ m/s² es la aceleración de la gravedad.

Materiales y montaje experimental

Necesitaréis una placa Wemos D1 R32 (ESP32) o un Arduino UNO, un sensor de distancia VL53L0X (láser infrarrojo TOF), una protoboard, cables Dupont, una pelota grande que rebote (de básquet, por ejemplo) y un soporte para fijar el sensor apuntando hacia abajo y un ordenador con el IDE de Arduino instalado.

Colocad el sensor VL53L0X en un soporte rígido, a una altura conocida (por ejemplo, 100 cm) del suelo, con el emisor láser apuntando verticalmente hacia abajo.

Conectad el sensor a la placa siguiendo el esquema I²C: VIN a 5 V (o 3,3 V), GND a GND, SCL al pin SCL (A5 en Arduino UNO, pin 22 en ESP32) y SDA al pin SDA (A4 en Arduino UNO, pin 21 en ESP32).

La pelota debe colocarse justo debajo del sensor antes de soltarla, de modo que el haz láser incida sobre su superficie superior durante la caída y el rebote.

Procedimiento

En primer lugar, conecte la placa al ordenador mediante un cable USB. Abrid el IDE de Arduino y cargad el código que se muestra al final de este experimento. A continuación, abrid el monitor serie (9600 baudios) para visualizar los datos en tiempo real. Sosteniendo la pelota justo debajo del sensor, anotad la distancia inicial.

Soltad la pelota sin impulso adicional y observad cómo el monitor serie registra la distancia en función del tiempo. Dejad que la pelota rebote varias veces hasta que se detenga. Copiad los datos del monitor serie a un archivo de texto para su posterior análisis.

Resultados esperados

Al representar gráficamente la distancia (eje Y) frente al tiempo (eje X), observaréis una secuencia de parábolas descendentes y ascendentes. La primera parábola corresponde a la caída inicial, seguida de parábolas cada vez más pequeñas que representan los rebotes sucesivos.

La altura máxima de cada rebote disminuye progresivamente, típicamente entre un 20-40% respecto al rebote anterior para una pelota de goma sobre una superficie dura.

Identificad el instante de cada impacto (mínimo de cada parábola) y la altura máxima alcanzada en cada rebote. Calculad el coeficiente de restitución e = $\sqrt{(h_1/h_0)}$ para el primer rebote. Repetid para los rebotes sucesivos y observad si se mantiene constante.

Representad la altura máxima en función del número de rebote en una gráfica semilogarítmica; la pendiente os dará la tasa de disipación de energía.

Para un análisis más preciso, ajustad cada parábola a la ecuación $y(t) = h_{max} - ½·g·(t - t_0)^2$ para obtener el valor experimental de g.

Ampliación

Repetid el experimento con pelotas de diferentes materiales (de futbol, de plástico) y comparad los coeficientes de restitución. Variad la altura inicial y estudiad cómo afecta al coeficiente.

Código

```
// Medida de distancia con sensor VL53L0X para experimento de caída y rebote
// Conexiones: VIN a 5V, GND a GND, SCL a A5 (UNO) o 22 (ESP32), SDA a A4 (UNO) o 21 (ESP32)

#include "Adafruit_VL53L0X.h"  // Librería del sensor

Adafruit_VL53L0X lox = Adafruit_VL53L0X();

void setup() {
  Serial.begin(9600);
  // Iniciar sensor
  if (!lox.begin()) {
    Serial.println("Error al iniciar VL53L0X");
    while (1);
  }
}

void loop() {
  VL53L0X_RangingMeasurementData_t measure;
  lox.rangingTest(&measure, false);

  if (measure.RangeStatus != 4) {
    // Enviar tiempo (ms) y distancia (mm) separados por punto y coma
    Serial.print(millis());
    Serial.print(";");
    Serial.println(measure.RangeMilliMeter);
  }
  delay(10);  // Aproximadamente 100 muestras por segundo
}
```

Estudio del movimiento circular: velocidad angular constante y aceleración angular

Mediremos la velocidad angular (ω) de un disco en movimiento circular uniforme (MCU) movido por un motorcito y utilizando 12 tiras opacas en la periferia, una cada 30º y una fotopuerta o sensor de proximidad.

También mediremos la aceleración angular (α) del movimiento circular uniformemente acelerado (MCUA) mediante un sistema de peso que cuelga de una polea acoplada al eje.



Verificaremos que ($\omega = \Delta\theta/\Delta t$\) es constante en el MCU analizando los 12 intervalos por vuelta y comprobando que los doce valores de Δt son aproximadamente iguales.

En el caso del MCUA se puede determinar experimentalmente el momento de inercia (I) del disco que gira aplicando la segunda ley de Newton rotacional: ($\tau = I \cdot \alpha$).

Por ejemplo, así lo hacen con una fotopuerta comercial en el Laboratorio de mecánica y termodinámica de Universidad Carlos III de Madrid (Departamento de Física, 2026), aunque tradicionalmente se hacía contando vueltas con un cronómetro (ENOSA, 1982) o utilizando un cronovibrador.

Fundamento físico

Cinemática rotacional

En el movimiento circular, la **posición angular** (θ) se mide en radianes (rad). Una vuelta completa corresponde a 2π rad. En este experimento el disco tiene N = 12 tiras opacas pegadas en la periferia, separadas uniformemente cada 30º. Cada vez que una tira interrumpe el haz de la fotopuerta, se genera un pulso que indica que el disco ha avanzado un ángulo:

$$\Delta\theta = \frac{2\pi}{N} = \frac{2\pi}{12} = \frac{\pi}{6}\ \text{rad} \approx 0,5236\ \text{rad}$$

La **velocidad angular instantánea** para el intervalo entre dos pulsos consecutivos es:

$$\omega_i = \frac{\Delta\theta}{\Delta t_i} = \frac{\pi/6}{\Delta t_i}\ (\text{rad/s})$$

donde Δt_i es el tiempo (en segundos) entre pulsos.

En el MCU (velocidad angular constante) todos los Δt_i de una vuelta son aproximadamente iguales. La velocidad angular media de la vuelta es:

$$\omega = \frac{1}{12}\sum_{i=1}^{12}\frac{\pi/6}{\Delta t_i}$$

En el MCUA (aceleración angular constante) los Δt_i disminuyen progresivamente porque el disco gira cada vez más rápido. Las ecuaciones cinemáticas angulares son análogas a las del MRUA lineal:

$$\omega(t) = \omega_0 + \alpha\, t, \qquad \theta(t) = \theta_0 + \omega_0 t + \frac{1}{2}\alpha\, t^2$$

Relación con magnitudes lineales

En el borde del disco (radio R):

$$v = \omega\, R\ (\text{velocidad lineal})$$

$$a_c = \omega^2 R = \frac{v^2}{R}\ (\text{aceleración centrípeta})$$

En MCUA aparece además una aceleración tangencial:

$$a_t = \alpha\, R$$

Dinámica rotacional (MCUA)

Cuando un peso de masa (m) cae tirando de un hilo enrollado en el eje del disco (radio r), el par (torque) aplicado sobre el disco es:

$$\tau = r \cdot F \approx r \cdot m\, g$$

La segunda ley de Newton rotacional relaciona el par neto con la aceleración angular:

$$\sum \tau = I \cdot \alpha$$

Combinando ambas expresiones obtenemos el momento de inercia experimental del disco:

$$I_{\text{exp}} = \frac{\tau}{\alpha} = \frac{m\, g\, r}{\alpha}$$

Para un disco macizo homogéneo de masa M y radio R, el momento de inercia teórico de referencia es:

$$I_{\text{teórico}} = \frac{1}{2} M R^2$$

Tabla de analogía traslación-rotación

Magnitud lineal	Símbolo	Unidad	↔	Magnitud angular	Símbolo	Unidad
Posición	x	m	↔	Posición angular	θ	rad
Velocidad	v	m/s	↔	Velocidad angular	ω	rad/s
Aceleración	a	m/s²	↔	Aceleración angular	α	rad/s²
Masa	m	kg	↔	Momento de inercia	I	kg·m²
Fuerza	F	N	↔	Par (torque)	τ	N·m
Segunda ley Newton	F = m a	N	↔	Segunda ley rotacional	τ = I· α	N·m

Materiales y montaje experimental

- Placa Arduino UNO (opcionalmente módulo Bluetooth) o Wemos D1 R32 (ESP32) para enviar datos a Phyphox).
- Fotopuerta IR: LED IR (ej. KY-008) + fototransistor (ej. TEMT6000 o SFH309), resistencia 220 Ω (para el LED IR), resistencia 10 kΩ (pull-up para el fototransistor), o alternativamente un sensor de proximidad.

Al usar un módulo Flying-Fish con comparador conectaremos VCC a 5V, GND a GND y DO a pin digital 2. Ajustad el potenciómetro de sensibilidad con un destornillador hasta que el LED indicador del módulo se encienda cuando el haz está interrumpido y se apague cuando no lo está.

Alternativa a la fotopuerta es utilizar un sensor de velocidad (encoder fotoelectrico) como el HC-020K (Andromina Robot, 2016).

- Disco de cartón o plástico de aproximadamente 20 cm de diámetro, con 12 tiras opacas adhesivas (cinta aislante negra, ~1×3 cm cada una) pegadas radialmente en la periferia, separadas 30º.
- Motor DC con reductor (para MCU), que puede ser un motor reciclado de un juguete o de un kit educativo. Fuente de alimentación regulable para el motor de corriente continua en el MCU.
- Poleas, cuerda fina y resistente y pesas calibradas (10 g, 20 g, 50 g, 100 g) o masas marcadas.
- Soporte universal con pinzas y nueces, protoboard, cables Dupont y cinta métrica o calibre.

Preparación del disco

Si se dispone de un equipo de mecánica para enseñanzas medias utilizad el disco metálico y los soportes que normalmente contienen, sino recortad un disco de cartón o plástico de aproximadamente 20 cm de diámetro, haced un agujero en el centro para el eje y marcad 12 puntos equiespaciados en la periferia, cada 30º, para lo que podéis usar un transportador de ángulos.

Pegad una tira de cinta aislante negra (opaca) de ~1×3 cm en cada marca, orientada radialmente. Las tiras deben sobresalir ligeramente del borde del disco para interrumpir el haz IR al pasar. Acoplad el disco al eje del motor (MCU) o a un eje libre montado sobre soportes (MCUA).

Configuración MCU (velocidad angular constante)

Acoplad el disco al eje del motor DC. Fijad el motor al soporte universal. Conectad el motor a una fuente de alimentación regulables (3–12 Vcc); a mayor voltaje, mayor velocidad angular ω.

Colocad la fotopuerta IR enfrentada al borde del disco, de modo que las tiras opacas interrumpan el haz al pasar. La distancia entre el LED IR y el fototransistor debe de ser de 1–3 cm para un funcionamiento fiable.



Simulación del montaje elaborada con Gemini

Configuración MCUA (aceleración angular)

Montad el disco en un eje libre (sin motor) sobre soportes con rodamientos o casquillos de baja fricción. Enrollad un hilo en el eje del disco y, si conviene, pasad el hilo por una polea pequeña y colgad un portapesas en el extremo libre.

Colocad la fotopuerta IR enfrentada al borde del disco, igual que en la configuración MCU. Aseguraos de que el disco gira libremente de manera que un ligero toque debería hacerlo girar varias vueltas antes de detenerse.



Simulación del montaje elaborada con Gemini

Procedimiento para MCU

Conectad la placa al ordenador mediante un cable USB. Abrid el monitor serie del IDE de Arduino (Herramientas → Monitor Serie) para visualizar los datos en tiempo real.

Cargad el Código A (MCU) en la placa. Sin alimentar el motor, girad el disco manualmente unas cuantas vueltas para comprobar que la fotopuerta detecta correctamente los doce pulsos por vuelta. Ajustad la posición de la fotopuerta si es necesario.

Alimentad el motor DC con una tensión baja (~3 V). Observad los valores en el monitor serie. Cada vuelta completa deben aparecer doce intervalos Δt_1 a Δt_{12}.

Verificad que los doce valores de Δt son aproximadamente iguales $\Delta t_1 \approx \Delta t_2 \approx \Delta t_{12}$. Si algún valor se desvía significativamente, revisad que las tiras opacas tengan el mismo ancho y estén bien pegadas.

Variad la tensión del motor (5 V, 7 V, 9 V) y repetid las mediciones. Para cada tensión, anotad la ω_{media} y comprobad que $(\omega \propto V)$ (en un motor DC ideal la velocidad es proporcional a la tensión aplicada, aunque en la práctica existe una tensión umbral de arranque).

Resultados esperados MCU

Los valores de la tabla siguiente son orientativos y dependen del motor concreto, del diámetro del disco y de la tensión aplicada. Valores típicos con disco de ~20 cm de diámetro y motor DC

V (V) motor	ω (rad/s)	T (s)	f (Hz)	v_{borde} (m/s)	a_c (m/s²)
3,0	4,2	1,50	0,67	0,42	0,88
5,0	8,5	0,74	1,35	0,85	3,61
7,0	12,8	0,49	2,04	1,28	8,19
9,0	17,1	0,37	2,72	1,71	14,6

Calculad la velocidad angular media por vuelta y la desviación típica.

Comprobad también la relación $(\omega \propto V)$ representando la ω media frente a la tensión del motor.

Procedimiento para MCUA

Cargad el Código B (MCUA) en la placa. Este código registra los instantes de todos los pulsos durante la caída del peso en un buffer interno de 200 posiciones (~16 vueltas completas).

Sin peso, comprobad que el disco gira libremente con baja fricción de manera que un toque ligero debería hacerlo girar varias vueltas.

Colocad una masa pequeña (10 g) en el portapesas. Mantened el disco sujeto con la mano para que no gire, y después soltad el disco para que el peso caiga. El código registrará los instantes de cada pulso durante toda la caída y al alcanzar las 200 pulsaciones (o al detenerse el disco), el código envía los datos por el monitor serie en formato CSV.

Repetid con masas de 20 g, 50 g y 100 g. Para cada masa, realizad tres mediciones y calculad el valor medio de α.

Para determinar el momento de inercia (I), conocidos el radio del eje (r), la masa (m) y la aceleración angular (α) medida, calculad:

$$I_{\text{exp}} = \frac{m\,g\,r}{\alpha}$$

Si se mede la masa (M) del disco con una balanza y el radio (R) con una cinta métrica se puede calcular el valor teórico $\left(I_{\text{teórico}} = \frac{1}{2}MR^2\right)$ y comparar con el valor experimental.

Como los datos se envían al monitor serie en formato CSV con tres columnas, tiempo absoluto t (s), número de pulso y velocidad angular ω (rad/s) se pueden copiar estos datos a una hoja de cálculo para representar ω(t) y obtener α como la pendiente de la recta de regresión.

Resultados esperados MCUA

El momento de inercia experimental suele ser **mayor que el teórico** porque incluye la contribución del eje, los soportes y la fricción. Además, el disco real no es perfectamente homogéneo. Valores típicos (disco de cartón, (R = 10 cm), (M ≈ 50 g), (r_{eje} = 0,5 cm).

Masa (g)	α (rad/s²)	I_{exp} (kg·m²)	$I_{teórico}$ (kg·m²)
10	0,82	5,98 x 10^{-4}	2,50 x 10^{-4}
20	1,58	6,21 x 10^{-4}	2,50 x 10^{-4}
50	3,71	6,61 x 10^{-4}	2,50 x 10^{-4}

Exportad los datos CSV generados por el Código B a una hoja de cálculo y representad ω(t) de manera que se obtiene una recta de pendiente α (rad/s²).

El momento de inercia experimental se obtiene a partir de:

$$I_{\text{exp}} = \frac{m\,g\,r}{\alpha}$$

Y se puede comparar como antes con el valor teórico para un disco macizo homogéneo:

$$I_{\text{teórico}} = \frac{1}{2}MR^2$$

Discutid las posibles fuentes de discrepancia entre ambos valores: Fricción en el eje y en la polea, que el disco de cartón no sea perfectamente macizo ni homogéneo, contribución del eje (varilla roscada, tuercas) al momento de inercia total, etc.

Ampliación

Si al disco se le añaden masas, por ejemplo, pegando masas conocidas (monedas, arandelas) en la periferia del disco y se repite el experimento ¿Cómo cambia el momento de inercia (I)? ¿aumenta con el cuadrado de la distancia de las monedas al eje?

Cálculo del trabajo y la energía en el caso del MCUA, donde el trabajo realizado por el peso al caer una altura (h) es W = mgh y la energía cinética rotacional del disco es $\left(E_c = \frac{1}{2} I\omega^2\right)$. Verificad la conservación de la energía: $(W \approx E_c + \text{pérdidas por fricción})$.

Código A — MCU: Medición de 12 intervalos por vuelta

```
/*Código para MCU con disco de 12 tiras opacas
Sensor: fotopuerta IR en pin digital 2 (interrupción 0)
Con N = 12 tiras, cada pulso corresponde a Δθ = π/6 rad.
El código mide los 12 intervalos Δt_i y calcula ω_i = (π/6) / Δt_i.
Al completar una vuelta (12 pulsos), envía los 12 valores por serial junto con la ω media, la desviación típica y el coeficiente de variación.
 */

const int SENSOR_PIN = 2;          // Pin digital para la fotopuerta (soporta interrupción)
const int N_STRIPS = 12;           // Número de tiras opacas en el disco
const float DELTA_THETA = PI / 6;  // Ángulo entre tiras: 30° = π/6 rad

volatile unsigned long t_anterior = 0;     // Instante (µs) del último pulso detectado
volatile float intervalos[N_STRIPS];       // Almacena los 12 Δt (en segundos) de la última vuelta
volatile int pulso_idx = 0;                // Índice del pulso actual (0 a 11)
volatile bool vuelta_completa = false;     // Flag: true cuando se han medido 12 intervalos

void setup() {
  // Configuración inicial: se ejecuta una vez al arrancar la placa
  Serial.begin(115200);                    // Iniciamos comunicación serie a 115200 baud
  pinMode(SENSOR_PIN, INPUT_PULLUP);       // Pin como entrada con resistencia pull-up interna (10 kΩ)

attachInterrupt(digitalPinToInterrupt(SENSOR_PIN), on_pulse, FALLING);
  // Configuramos interrupción por flanco de bajada: el fototransistor conduce (LOW) cuando el haz IR se interrumpe, y deja de conducir (HIGH) cuando el haz llega al fototransistor
  Serial.println("Iniciando MCU...");     // Mensaje de inicio en el monitor serie
}

void on_pulse() {
  // Rutina de interrupción: se ejecuta en cada flanco de bajada del sensor
  unsigned long t_actual = micros();       // Leemos el instante actual en microsegundos (resolución 4 µs)
  unsigned long dt = t_actual - t_anterior; // Tiempo entre este pulso y el anterior (µs)
  t_anterior = t_actual;                   // Actualizamos la referencia para el próximo pulso

  if (pulso_idx > 0) {                     // El primer pulso no tiene Δt válido (no hay pulso anterior)
    intervalos[pulso_idx - 1] = dt / 1.0e6;  // Convertimos microsegundos a segundos y almacenamos
  }

  pulso_idx++;                             // Avanzamos al siguiente pulso

  if (pulso_idx >= N_STRIPS + 1) {         // Si hemos completado una vuelta (12 intervalos medidos)
    pulso_idx = 0;                         // Reiniciamos el contador de pulsos
    vuelta_completa = true;                // Señalizamos al loop() que hay datos disponibles
  }
}

void loop() {
  // Bucle principal: se ejecuta continuamente mientras la placa esté alimentada
  if (vuelta_completa) {                   // Si hay datos de una vuelta completa disponibles
    vuelta_completa = false;               // Reseteamos el flag para la siguiente vuelta

    // Calcular ω para cada uno de los 12 intervalos y la ω media de la vuelta
    float omega[N_STRIPS];
    float suma_omega = 0;
    for (int i = 0; i < N_STRIPS; i++) {
      omega[i] = DELTA_THETA / intervalos[i];  // ω_i = (π/6) / Δt_i (rad/s)
      suma_omega += omega[i];
    }
```

```
  float omega_media = suma_omega / N_STRIPS;
// Velocidad angular media de la vuelta

  // Calcular la desviación típica de ω (indica la
calidad del MCU)
  float suma_desv = 0;
  for (int i = 0; i < N_STRIPS; i++) {
    float dif = omega[i] - omega_media;
    suma_desv += dif * dif;
  }
  float desv_tip = sqrt(suma_desv / (N_STRIPS -
1));  // Desviación típica muestral (rad/s)
  float cv = desv_tip / omega_media;                  //
Coeficiente de variación (adimensional)

  // Enviar resultados al monitor serie para su
análisis
  Serial.print("ω_media: ");
Serial.print(omega_media, 4); Serial.print(" rad/s\t");
  Serial.print("σ: "); Serial.print(desv_tip, 4);
Serial.print(" rad/s\t");
  Serial.print("CV: "); Serial.print(cv * 100, 2);
Serial.println(" %");
  Serial.print("T = "); Serial.print(2 * PI /
omega_media, 4); Serial.println(" s");    // Período
  Serial.print("f = "); Serial.print(omega_media / (2
* PI), 4); Serial.println(" Hz");   // Frecuencia

  // Enviar también los 12 valores de ω_i para
análisis detallado (posibles tendencias)
  Serial.print("ω_i: ");
  for (int i = 0; i < N_STRIPS; i++) {
    Serial.print(omega[i], 4);
    if (i < N_STRIPS - 1) Serial.print(", ");
  }
  Serial.println();
  Serial.println("---");  // Separador entre vueltas
consecutivas
 }
}
```

Código B — MCUA: Registro temporal de pulsos durante la aceleración

```
/*Código para MCUA con disco de 12 tiras opacas
y peso que cae
Sensor: fotopuerta IR en pin digital 2
Registra el instante de CADA pulso durante la
caída en un buffer interno.
ωi = (π/6) / Δti. La pendiente de ω(t) da la
aceleración angular α.
Máximo 200 pulsos registrados (~16 vueltas
completas).
Los datos se envían en formato CSV para su
posterior análisis en hoja de cálculo.
*/

const int SENSOR_PIN = 2;          // Pin digital para
la fotopuerta (soporta interrupción)
const int N_STRIPS = 12;           // Tiras opacas en
el disco
const float DELTA_THETA = PI / 6;  // Ángulo entre
tiras: 30° = π/6 rad
const int MAX_PULSOS = 200;        // Máximo
número de pulsos a registrar en el buffer

volatile unsigned long tiempos[MAX_PULSOS];  //
Array que almacena los instantes de cada pulso
(µs)
volatile int pulso_count = 0;              // Contador de
pulsos registrados hasta el momento
volatile bool muestreo_completo = false;     // Flag:
true cuando se alcanza MAX_PULSOS
volatile bool nuevo_pulso = false;         // Flag para
indicar al loop() que hay un pulso nuevo

void setup() {
 // Configuración inicial: se ejecuta una vez al
arrancar la placa
 Serial.begin(115200);                 // Iniciamos
comunicación serie a 115200 baud
 pinMode(SENSOR_PIN, INPUT_PULLUP);      //
Pin como entrada con resistencia pull-up interna
(10 kΩ)

attachInterrupt(digitalPinToInterrupt(SENSOR_PIN)
, on_pulse, FALLING);
 // Interrupción por flanco de bajada: se dispara
cuando una tira opaca interrumpe el haz IR
 Serial.println("Iniciando MCUA...");    // Mensaje de
inicio
 Serial.println("Soltad el disco! Registrando
pulsos...");
}

void on_pulse() {
 // Rutina de interrupción: se ejecuta en cada
flanco de bajada del sensor
 if (pulso_count < MAX_PULSOS) {        // Si aún
no hemos llenado el buffer de 200 posiciones
   tiempos[pulso_count] = micros();      //
Guardamos el instante actual en microsegundos
   pulso_count++;                      // Incrementamos
el contador de pulsos
   nuevo_pulso = true;                 // Señalizamos al
loop() que hay un nuevo pulso disponible
```

```
  } else if (!muestreo_completo) {
    muestreo_completo = true;             // Buffer lleno, activamos el flag de finalización
  }
}

void loop() {
  // Bucle principal: se ejecuta continuamente mientras la placa esté alimentada
  if (nuevo_pulso) {                      // Si hay un nuevo pulso registrado por la interrupción
    nuevo_pulso = false;
    if (pulso_count >= 2) {               // Necesitamos al menos 2 pulsos para calcular un Δt
      // Calcular ω para este intervalo entre el pulso actual y el anterior
      unsigned long dt = tiempos[pulso_count - 1] - tiempos[pulso_count - 2];  // Δt en µs
      float omega = DELTA_THETA / (dt / 1.0e6);  // ω = (π/6) / Δt (rad/s)
      float t = tiempos[pulso_count - 1] / 1.0e6;  // Tiempo absoluto desde el inicio (s)

      // Enviar datos en formato CSV: t(s); pulso#; ω(rad/s)
      // Este formato permite copiar directamente a una hoja de cálculo para análisis
      Serial.print(t, 6); Serial.print("; ");
      Serial.print(pulso_count - 1); Serial.print("; ");
      Serial.println(omega, 6);
    }
  }

  if (muestreo_completo) {                // Si el buffer de 200 pulsos se ha llenado
    Serial.println("--- FIN MUESTREO ---");

    // Calcular α por regresión lineal simplificada de ω vs t
    // Ecuación del MCUA: ω(t) = ω₀ + α·t  →  α es la pendiente de la recta
    // Método simplificado: α = (ω_final - ω_inicial) / (t_final - t_inicial)
    // Para mayor precisión, copiad los datos CSV a una hoja de cálculo y usad ESTIMACION.LINEAL
    if (pulso_count >= 10) {              // Necesitamos suficientes puntos para un ajuste fiable
      int idx_ini = pulso_count * 0.2;    // Ignoramos el primer 20% (transitorio inicial)
      int idx_fin = pulso_count * 0.9;    // Ignoramos el último 10% (efectos de fricción)

      // Calcular tiempos relativos al segundo pulso (índice 1)
      unsigned long dt_ini = tiempos[idx_ini] - tiempos[1];   // Tiempo desde el inicio (µs)
      unsigned long dt_fin = tiempos[idx_fin] - tiempos[1];
      float t_ini = dt_ini / 1.0e6;       // Convertimos a segundos
      float t_fin = dt_fin / 1.0e6;

      // Velocidad angular en los puntos inicial y final (usando el Δt local)
      float dt_local_ini = (tiempos[idx_ini] - tiempos[idx_ini - 1]) / 1.0e6;  // Δt local en s
      float dt_local_fin = (tiempos[idx_fin] - tiempos[idx_fin - 1]) / 1.0e6;
      float w_ini = DELTA_THETA / dt_local_ini;   // ω en el punto inicial (rad/s)
      float w_fin = DELTA_THETA / dt_local_fin;   // ω en el punto final (rad/s)

      float alpha = (w_fin - w_ini) / (t_fin - t_ini);  // Aceleración angular α (rad/s²)
      Serial.print("α ≈ "); Serial.print(alpha, 4); Serial.println(" rad/s²");

      // NOTA: Si se conoce la masa m (kg) y el radio del eje r (m):
      // I = m · g · r / α  →  momento de inercia experimental del disco (kg·m²)
    }

    muestreo_completo = false;            // Reseteamos el flag para una nueva medición
    pulso_count = 0;                      // Reiniciamos el contador de pulsos
    Serial.println("Listo para nueva medición. Soltad el disco!");
  }
}
```

Oscilaciones y vibraciones

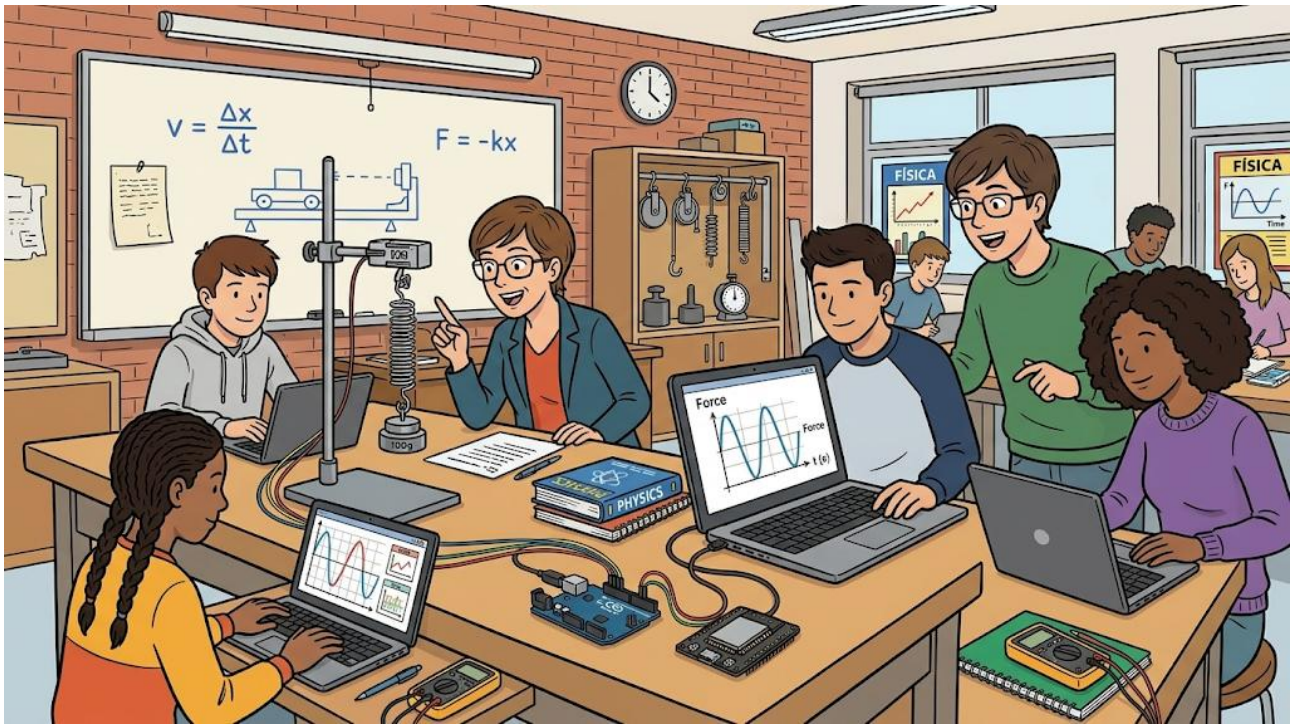


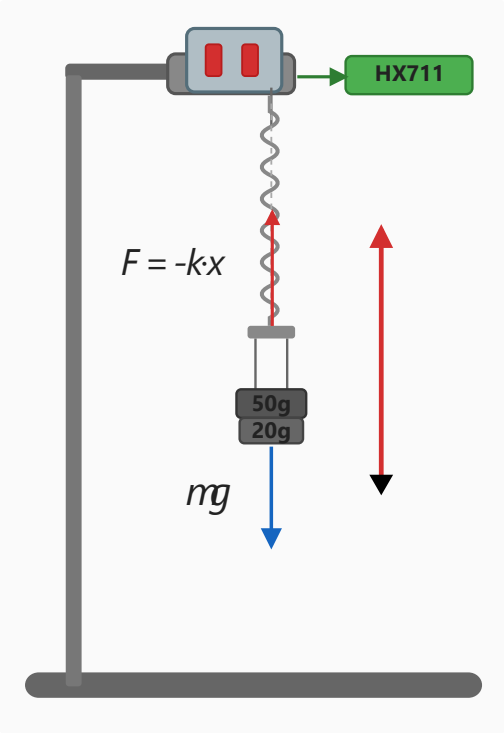
Ilustración realizada con Gemini. Esté pensada para adornar, aunque el alumnado puede entretenerse en buscar los errores evidentes en las conexiones que contiene.

Oscilaciones de un muelle: medida de fuerzas con célula de carga

El estudio de la ley de Hooke y del movimiento armónico simple con muelles es uno de los casos en los que se centraban ya los experimentos clásicos con pesas, metros y cronómetros (Departamento de Física. Universidad de Jaen, 2021), también con dataloggers hace años (Aparicio y Lozano, 2002a y 2002b) o con teléfonos móviles más recientemente (Ramírez, 2022).

En este experimento nos detendremos en el estudio del movimiento armónico simple (MAS) de una masa suspendida de un muelle midiendo directamente la fuerza recuperadora con una célula de carga (galgas extensométricas + HX711).

La cuestión está en determinar la constante elástica (k) del muelle mediante la ley de Hooke (F = -k x) y mediante la relación entre el período y la masa $\left(T = 2\pi\sqrt{m/k}\right)$ y comparar ambos métodos, tanto en el caso de oscilar la pesa en el aire como de hacerlo sumergida en agua



En el siguiente experimento (Oscilaciones de un muelle: determinación de la constante elástica mediante sensor de distancia) volveremos a enfrentarnos a este mismo problema, pero entonces midiendo los desplazamientos con sensores de distancia.

Este experimento se ha hecho tradicionalmente midiendo los periodos de oscilación con un cronómetro (CDECT, 2002c y d), pero también utilizando un sensor de fuerza comercial (Aparicio y Lozano, 2002g y h), o con la ayuda de teléfonos móviles (Marco , 2017)

Fundamento físico

Ley de Hooke y fuerza recuperadora

Un muelle ideal cumple la **ley de Hooke**: la fuerza recuperadora es proporcional al desplazamiento respecto a la posición de equilibrio y de sentido opuesto:

$$F = -k \cdot x$$

donde (F) es la fuerza recuperadora (N), (k) la constante elástica (N/m) y (x) el desplazamiento desde la posición de equilibrio (m). El signo negativo indica que la fuerza se opone al desplazamiento, tendiendo a devolver la masa a su posición de equilibrio.

Ecuación diferencial del MAS

Si aplicamos la segunda ley de Newton ($F = m \cdot a$) al sistema masa-muelle:

$$m\frac{d^2x}{dt^2} = -kx$$

Y la reorganizamos

$$\frac{d^2x}{dt^2} + \frac{k}{m}x = 0$$

Tenemos la ecuación diferencial del oscilador armónico simple, cuya solución general es:

$$x(t) = A\cos(\omega t + \phi), \qquad \omega = \sqrt{\frac{k}{m}}$$

Donde (x) es la posición de la masa que oscila, (A) es la amplitud, (m) la masa colgada, (ω) la frecuencia angular (rad/s) y (π) la fase inicial (rad).

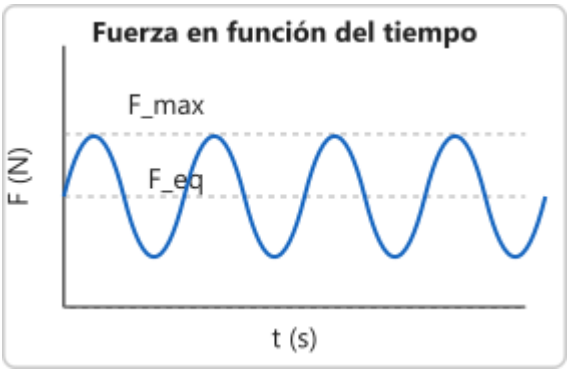
Si se deriva la posición respecto al tiempo se obtiene la velocidad:

$$v(t) = -A\,\omega\,\mathrm{sen}(\omega t + \phi)$$

Si se vuelve a derivar tendremos la aceleración:

$$a(t) = -A\,\omega^2\cos(\omega t + \phi)$$

De manera que en los extremos la aceleración es máxima ($\pm A\,\omega^2$), y la fuerza también, y en el centro de la oscilación (cuando x = 0) la aceleración de la masa que cuelga es cero y la fuerza que padece también.



Período de oscilación

El período (T) (tiempo que tarda el sistema en completar una oscilación completa) se relaciona con la frecuencia angular mediante *ω = 2π/T*. Sustituyendo:

$$T = 2\pi\sqrt{\frac{m}{k}}$$

Este resultado indica que el período es independiente de la amplitud (isoacronía del MAS), siempre que nos mantengamos en el régimen lineal de la ley de Hooke (pequeñas oscilaciones).

Energía en el MAS

La energía total del oscilador se conserva (en ausencia de rozamiento) y se intercambia entre energía potencial elástica y energía cinética:

$$E_{\text{total}} = E_p + E_c = \frac{1}{2}kx^2 + \frac{1}{2}mv^2 = \frac{1}{2}kA^2 \text{ (constante)}$$

Oscilaciones amortiguadas

En un sistema real siempre hay fuerzas disipativas (rozamiento con el aire, fricción interna del muelle, etc.) de manera que la ecuación diferencial del movimiento debe incorporar un término proporcional a la velocidad (amortiguamiento viscoso):

$$m\frac{d^2x}{dt^2} + b\frac{dx}{dt} + kx = 0$$

donde b es el coeficiente de amortiguamiento viscoso (N·s/m). La solución general a esta ecuación es:

$$x(t) = A_0 e^{-\gamma t}\cos(\omega' t + \phi)$$

con $\gamma = b/(2m)$ (coeficiente de amortiguamiento, s^{-1}) y $\omega' = \sqrt{\omega_0^2 - \gamma^2}$ (frecuencia angular amortiguada, rad/s), siendo $\omega_0 = \sqrt{k/m}$ la frecuencia natural del sistema.

A su vez, la amplitud de las oscilaciones decrece exponencialmente y esta variación se puede medir con el **decremento logarítmico** (δ), que cuantifica esta atenuación:

$$\delta = \ln\left(\frac{A_n}{A_{n+1}}\right) = \gamma T'$$

donde A_n y A_{n+1} son amplitudes de oscilaciones consecutivas y $(T' = 2\pi/\omega')$ es el período amortiguado. Midiendo δ a partir de los valores de A de la gráfica experimental, podemos obtener el valor del coeficiente d amortiguamiento $(\gamma = \delta/T')$.

En función del valor de γ podemos encontrarnos con diferentes tipos de regímenes de amortiguamiento:

- Subcrítico $(\gamma < \omega_0)$: el sistema oscila con amplitud decreciente. Es el caso más común en nuestros experimentos.
- Crítico $(\gamma = \omega_0)$: el sistema retorna al equilibrio en el menor tiempo posible, sin oscilar. Es el régimen ideal para amortiguadores de vehículos.
- Sobreamortiguado $(\gamma > \omega_0)$: el sistema retorna al equilibrio lentamente, sin oscilar.

Veamos ahora que pasa con las oscilaciones en un fluido viscoso (más o menos). Al sumergir la masa oscilante en un fluido (agua, aceite, glicerina), el amortiguamiento aumenta drásticamente por dos efectos adicionales:

El empuje de Arquímedes establece que el fluido ejerce una fuerza vertical hacia arriba igual al peso del volumen desalojado:

$$F_e = \rho_{\text{fluido}} \cdot V_{\text{sumergido}} \cdot g$$

Esto reduce la tensión en el muelle y desplaza la posición de equilibrio. La tensión neta en el muelle será ahora:

$$F_{\text{muelle}} = mg - \rho V g.$$

Aparece una fuerza de fricción viscosa que para una esfera de radio (r) moviéndose a baja velocidad (ley de Stokes) en un fluido de viscosidad (η) es:

$$F_d = 6\pi\eta r v$$

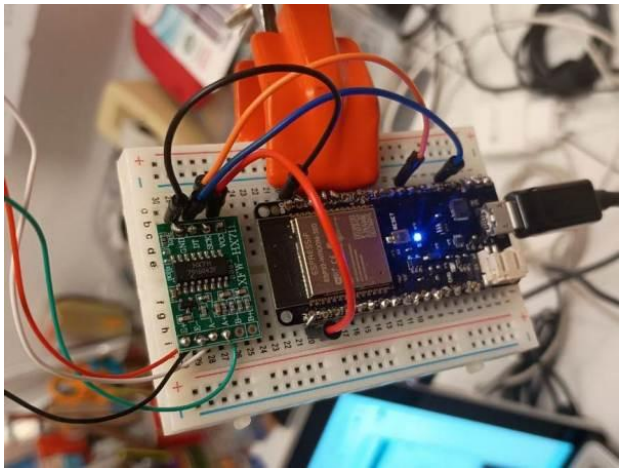
La viscosidad dinámica del agua $(1{,}0 \times 10^{-3} Pa \cdot s)$ es aproximadamente 55 veces mayor que la del aire $(1{,}8 \times 10^{-5} Pa \cdot s)$. Por tanto, el coeficiente de amortiguamiento (γ) en agua es del orden de 0,5–1,0 s^{-1}, frente a ~0,02 s^{-1} en aire. Esto significa que en aire se observan más de 100 oscilaciones antes de extinguirse, mientras que en agua apenas se apreciarán de 5 a 10 oscilaciones completas.

Materiales y montaje experimental

Se necesita una placa Wemos D1 R32 (ESP32) o Arduino UNO, una **célula de carga** de 1 kg (o 5 kg) con amplificador **HX711**, una protoboard, cables Dupont, un muelle helicoidal (k entre 5 y 20 N/m), masas calibradas (juego de pesas de 10 g a 100 g con portapesas), un soporte universal con nuez y pinza.

Como se indica en el apartado del código, las células de carga requieren **calibración** con masas conocidas para establecer la relación entre los valores digitales del HX711 y la fuerza real en newtons, ya que sin calibración solo obtenemos valores relativos adimensionales.

Para más detalles sobre la célula de carga y el HX711 y como conectarlos, consultad el capítulo de los sensores.

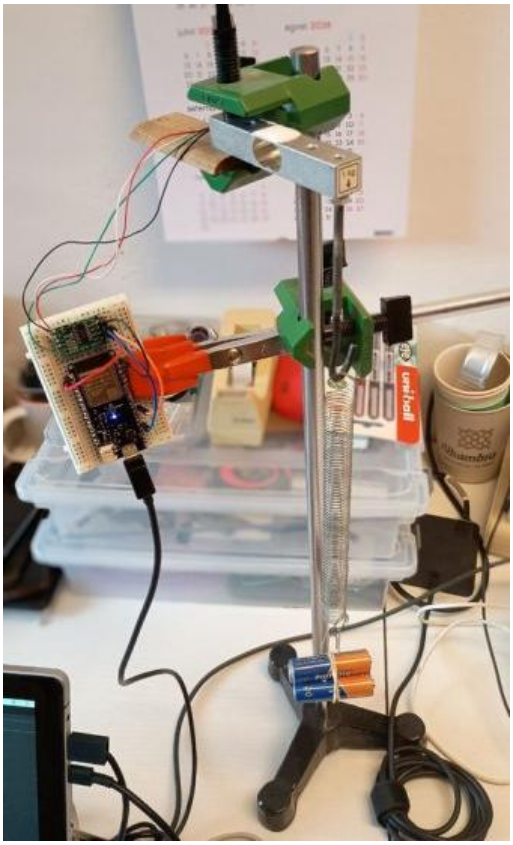


En la entrada del HX711 están soldados los cables de la célula de carga y en la salida están soldadas las patitas y pinchadas en placa de pruebas.

Recuerdo que el HX711 se conecta a la placa:
DT (data) a pin digital (GPIO4 en ESP32, D2 en UNO), SCK (clock) a pin digital (GPIO5 en ESP32, D3 en UNO), VCC a 5V (el HX711 necesita 5V para el ADC) y GND a GND.
Y la célula de carga se conecta al HX711 tal que los cuatro cables van: rojo a E+, negro a E-, blanco a A- y verde = A+.

Para la variante con amortiguamiento en fluido se necesita además un recipiente transparente (vaso de precipitados o bote de cristal de 500 mL), agua destilada (o del grifo), hilo de pescar (sedal) o hilo fino de unos 15–20 cm de largo, y opcionalmente aceite de cocina o glicerina para comparar viscosidades.

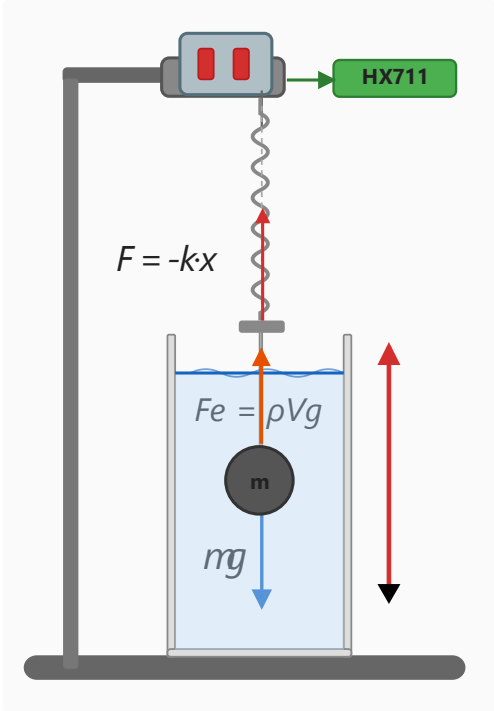
Montaje base (oscilaciones en aire)
1. Fijad la célula de carga al soporte universal mediante la pinza, de modo que quede suspendida verticalmente.
2. Colgad el muelle del extremo inferior de la célula de carga (el muelle debe colgar libremente, sin rozar con nada).
3. Colgad el portapesas del extremo inferior del muelle.



Aseguraos de que las oscilaciones sean estrictamente verticales, sin balanceo lateral, ya que este introduce componentes de fuerza no deseadas y falsifica las medidas.

Variante con amortiguamiento en fluido

1. Alargad lo que cuelga la masa con un hilo fino de unos 10–15 cm entre el muelle y las pesas, de forma que la masa quede suspendida por el hilo, de manera que se pueda sumergir sin mojar el muelle.
2. Colocad el recipiente con agua (o el fluido elegido) debajo de la masa, de modo que al bajarla quede completamente sumergida.
3. Ajustad la altura del soporte para que la masa quede sumergida aproximadamente 2–3 cm por debajo de la superficie, sin tocar el fondo ni las paredes del recipiente.
4. Proteged la célula de carga y el HX711 de salpicaduras: cubridlos con una bolsa de plástico o film transparente. El agua puede dañar las galgas extensométricas y el circuito electrónico.



5. Registrad la nueva posición de equilibrio: el empuje de Arquímedes reduce la tensión en el muelle. Si la masa tiene un volumen (V) y la densidad del agua es $(\rho = 1000\,\mathrm{kg/m^3})$, la reducción de tensión es $(\Delta F = \rho V g)$. Si lo que cuelga es una esfera de 2 cm de diámetro, $(\Delta F \approx 0{,}041\,\mathrm{N})$ (equivalente a ~4,2 g).

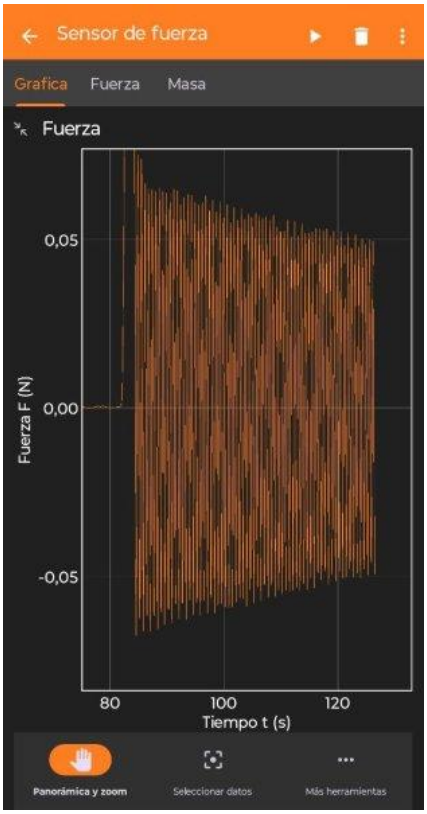
Procedimiento

Medida estática (ley de Hooke)

1. Para cada masa colgada (10 g, 20 g, ..., 100 g), registrad la lectura de fuerza en equilibrio. Se puede hacer a través del puerto serie del ordenador mirando los datos en el IDE de Arduino y apuntándolos de uno en uno.
2. Después representad F frente al alargamiento (x) y la pendiente de la recta (F = k x) será la constante elástica (k), por ejemplo 7,1 N/m

Medida dinámica (período vs masa)

1. Se cuelga una determinada masa (por ej. 50 g), se separa de la posición de equilibrio unos 2-3 cm y se suelta.
2. Registrad la fuerza en función del tiempo durante al menos 10 oscilaciones completas. Preferiblemente hacedlo con la aplicación Phyphox ya que directamente se obtiene la gráfica F(t) que permite medir el tiempo entre esas 10 oscilaciones
3. Si se desea se pueden repetir las medidas para diferentes masas (20 g, 50 g, 80 g, 100 g).
4. El período (T) se determina entre dos máximos consecutivos de la fuerza (la fuerza pasa por su valor máximo cuando la elongación es máxima, y por su valor mínimo (cercano a cero) cuando la masa pasa por la posición de equilibrio), o entre diez máximos y dividiendo por diez. Recordad que la célula mide la tensión del muelle, no el peso de la masa.)



Captura de pantalla del gráfico fuerza frente al tiempo de la oscilación en el aire de un muelle del que pende una masa

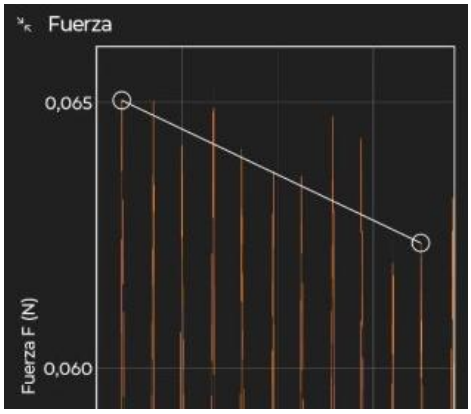


Gráfico ampliado en el que se puede ver el tiempo que ha trascurrido en 10 oscilaciones del muelle

Por ejemplo, si al medir el tiempo que ha trascurrido entre 10 oscilaciones de una masa de 70,9 g el resultado es de 0,63 s, tenemos que la frecuencia es 15,9 Hz.

Como $\omega = \mathrm{f}\,2\pi = \sqrt{\frac{k}{m}}$, se puede despejar la constante del muelle, sustituir los valores y calcularla. En este caso concuerda el valor obtenido por este método dinámico con el obtenido anteriormente por el método de la ley de Hooke.

Medida con amortiguamiento en agua

1. Con la masa sumergida y en reposo, reiniciad la placa microcontroladora para que la fuerza se ponga a cero (es la nueva posición de equilibrio, con el empuje de Arquímedes descontado).
2. Separad la masa unos 2 - 3 cm de la posición de equilibrio tirando hacia arriba o empujando hacia abajo el hilo y soltándolo suavemente.
3. Registrad la fuerza en función del tiempo. Las oscilaciones se extinguirán en 2–5 segundos debido al fuerte amortiguamiento viscoso.
4. Se puede repetid con otros fluidos (aceite de cocina, glicerina diluida) para comparar el efecto de la viscosidad.

Como en el caso anterior, la fuerza oscilará entre un mínimo cercano a 0 N (en la posición de equilibrio, donde la velocidad es máxima y la aceleración nula) y un máximo igual al peso aparente en el punto de máxima elongación. La gráfica F(t) será una sinusoide mucho más amortiguada (apenas 5–10 oscilaciones apreciables) de amplitud decreciente en mucho menos tiempo

El período amortiguado (T') será ligeramente mayor que (T), pero la envolvente en el caso del agua cae a (1/e) de la amplitud inicial en un tiempo, $t = 1/\gamma, \mathrm{de}\ 1\ \acute{o}\ 2$ segundos

Ampliaciónes

Se pueden **comparar los resultados** y calcular el error relativo entre ambos métodos (Hooke y oscilación):

$$\varepsilon = \frac{\left|k_{\mathrm{Hooke}} - k_{\mathrm{período}}\right|}{k_{\mathrm{promedio}}} \times 100$$

Valores típicos: $(\varepsilon < 10\%)$ si el muelle es ideal.

Estudio de los amortiguamientos en aire y en el agua.

Se pueden ajustad las envolventes de las oscilaciones a una exponencial decreciente: $(A(t) = A_0 e^{-\gamma t})$, y calcular el coeficiente de amortiguamiento a partir del decremento logarítmico.

$$\delta = \ln\left(\frac{A_n}{A_{n+1}}\right)$$

con $\delta = \gamma \cdot T$ para el aire y $\delta = \gamma \cdot T'$ para el agua.

El coeficiente de amortiguamiento (γ) (s^{-1}) refleja las pérdidas por fricción interna del muelle y rozamiento con el aire o el agua. Para un muelle metálico típico, $\gamma_{aire} \approx 0{,}01 - 0{,}03\ s^{-1}$

Ahora se puede comparar la relación de coeficientes de amortiguamiento con la relación de viscosidades:

$$\left[\frac{\gamma_{\mathrm{agua}}}{\gamma_{\mathrm{aire}}} \approx \frac{\eta_{\mathrm{agua}}}{\eta_{\mathrm{aire}}} \approx \frac{1{,}0 \times 10^{-3}}{1{,}8 \times 10^{-5}} \approx 55\right]$$

Y ver si se cumple esta proporcionalidad dentro del error experimental. Tened en cuenta en el análisis las posibles situaciones reales como la geometría no esférica de las pesas, el flujo turbulento a altas velocidades o la contribución de la fricción interna del muelle.

Ampliación

Amortiguamiento en otros fluidos viscosos. Repetid la variante con agua utilizando otros fluidos de diferente viscosidad: aceite de cocina $(\eta \approx 0{,}07\ Pa \cdot s, \sim \times 70\ la\ del\ agua)$, $glicerina\ (\eta \approx 1{,}5\ Pa \cdot s, \sim \times 1500\ la\ del\ agua)$. Ojo!: la glicerina produce un amortiguamiento tan intenso que el sistema puede pasar a régimen sobreamortiguado (sin

oscilaciones). ¿A partir de qué viscosidad desaparecen las oscilaciones?

Amortiguamiento forzado. Acercad un imán potente al muelle (si es metálico) para introducir amortiguamiento magnético. ¿Cómo cambia (γ)?

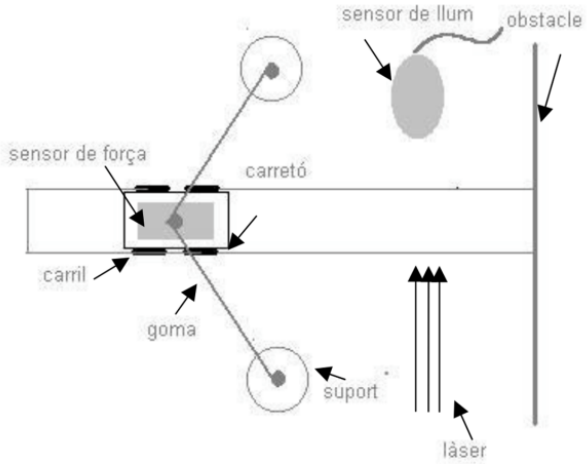
Isoacronía. Repetid con diferentes amplitudes (1, 2, 3 cm) para la misma masa. ¿Cambia el período? Verificad que T es independiente de la amplitud dentro del error experimental.

Muelles en serie y paralelo. Conectad dos muelles iguales en serie $\left(1/k_{\text{eq}} = 1/k_1 + 1/k_2\right)$ y en paralelo $\left(k_{\text{eq}} = k_1 + k_2\right)$. Verificad experimentalmente las leyes de combinación.

Geometría de la masa. Se puede repetir la experiencia si se dispone de masas de diferente forma (cilíndrica, cúbica, esférica) pero igual masa y volumen, y estudiar cómo afecta la forma al coeficiente de arrastre y, por tanto, a (γ). Hay que tener en cuenta que la ley de Stokes $(F_d = 6\pi\eta r v)$ solo es válida para esferas y que para otras geometrías el coeficiente de forma cambia.

El empuje de Arquímedes. Al sumergir la masa en agua, la posición de equilibrio cambia por el empuje, lo que permite determinar experimentalmente el volumen de la masa sumergida usando la célula de carga, para lo que no habría más que comparar la lectura de fuerza en el aire y el agua (la diferencia es $(\Delta F = \rho V g)$).

Más experimentos. El estudio del movimiento armónico simple es sólo una posibilidad de utilización del sensor de fuerza para realizar estudios experimentales. Se puede estudiar el choque de un vehículo, como en *Impuls i quantitat de moviment* (Aparicio y Lozano, 2002c y d), o cómo en *Com fer un salt de bungee-jumping* segur* (Aparicio y Lozano, 2004ª y b)).



Esquema del montaje para estudiar el choque de una carretilla

También se puede analizar cómo evoluciona una reacción química (BBC Science, 2026) en la que hay desprendimiento de gases como por ejemplo la reacción del bicarbonato de sodio con vinagre (ácido acético), donde el dióxido de carbono gases liberado provoca una disminución en la masa del vaso de reacción, o del zinc y el ácido clorhídrico, etc. En este último caso iría mejor una balanza sobre la que se colocaría un vaso de precipitados o un erlenmeyer en el que se fuera produciendo la reacción.

Código

Código completo para ESP32 o Arduino UNO con HX711:

```
/*Sensor de fuerza.
Se necesita una celda de carga de 1 kg y el
amplificador de señal Hx711.
También un microprocesador esp32 que incorpora
bluetooth.

Se han de instalar las librerias HX711_ADC.h (de
Olav Kallhovd, para el amplificador de señal)
y phyphoxBle.h (de RWTH para la comunicación
con la aplicación Phyphox via bluetooth).

La celda de carga se ha de conectar al amplificador
de señal con el cable rojo a E+,
el cable negro a E-, el cable blanco a A- y el cable
verde a A+
El amplificador de señal se conecta: VCC a 3,3 V,
GND a GND, DT al pin 4 y SCK al pin 5.
*/

#include <HX711_ADC.h>
```

```
#include <phyphoxBle.h>

#define BUTTON_PIN 2
#define SENSOR_NAME "Sensor de fuerza M"

// Pines del HX711: DT = 4, SCK = 5
HX711_ADC LoadCell(4, 5);

// Factor de calibración (para gramos)
const float calFactor = 1119.22;

// Variables de control
unsigned long t_offset = 0;

void setup() {

  Serial.begin(115200);

  // --- Configuración Phyphox ---
  PhyphoxBLE::start(SENSOR_NAME);
  PhyphoxBLE::experimentEventHandler =
&newExperimentEvent;
  PhyphoxBLE::printXML(&Serial);

  PhyphoxBleExperiment exp;
  exp.setTitle("Sensor de fuerza");
  exp.setCategory("Arduino Experiments");
  exp.setDescription("Dibuja la gráfica de la fuerza y
la masa medidas por la celda de carga.");

  // Vistas
  PhyphoxBleExperiment::View viewGraph;
  viewGraph.setLabel("Gráfica fuerza");

  PhyphoxBleExperiment::View viewForce;
  viewForce.setLabel("Fuerza");

  PhyphoxBleExperiment::View viewMass;
  viewMass.setLabel("Masa");

  //vista para la gráfica de masa
  PhyphoxBleExperiment::View viewMassGraph;
  viewMassGraph.setLabel("Gráfica masa");

  // Gráfico de fuerza
  PhyphoxBleExperiment::Graph graphF;
  graphF.setLabel("Fuerza");
  graphF.setUnitX("s");
  graphF.setUnitY("N");
  graphF.setLabelX("Tiempo t");
  graphF.setLabelY("Fuerza F");
  graphF.setChannel(1, 2);  // tiempo, fuerza

  // gráfico de masa
  PhyphoxBleExperiment::Graph graphM;
  graphM.setLabel("Masa");
  graphM.setUnitX("s");
  graphM.setUnitY("g");
  graphM.setLabelX("Tiempo t");
  graphM.setLabelY("Masa m");
  graphM.setChannel(1, 3);  // tiempo, masa

  // Valor fuerza
  PhyphoxBleExperiment::Value force;
  force.setLabel("F");
  force.setPrecision(3);
  force.setUnit("N");
  force.setColor("FFFFFF");
  force.setChannel(2);
  force.setXMLAttribute("size=\"2\"");

  // Valor masa
  PhyphoxBleExperiment::Value mass;
  mass.setLabel("m");
  mass.setPrecision(1);
  mass.setUnit("g");
  mass.setColor("FFFFFF");
  mass.setChannel(3);
  mass.setXMLAttribute("size=\"2\"");

  // Añadir elementos a vistas
  viewGraph.addElement(force);
  viewGraph.addElement(graphF);

  viewForce.addElement(force);

  viewMass.addElement(mass);

  //añadir gráfica de masa
  viewMassGraph.addElement(mass);
  viewMassGraph.addElement(graphM);

  // Añadir vistas al experimento
  exp.addView(viewGraph);
  exp.addView(viewForce);
  exp.addView(viewMass);
  exp.addView(viewMassGraph);

  PhyphoxBLE::addExperiment(exp);

  // --- Configuración HX711 ---
  LoadCell.begin();
  LoadCell.start(2000);
  LoadCell.setCalFactor(calFactor);

  pinMode(BUTTON_PIN, INPUT_PULLUP);
```

```
}

void loop() {

  // --- Tara con botón ---
  if (digitalRead(BUTTON_PIN) == LOW) {
    LoadCell.tareNoDelay();
    while (!LoadCell.getTareStatus()) {
      LoadCell.update();
    }
  }

  // --- Lectura del HX711 ---
  if (LoadCell.update()) {

    float t = (millis() - t_offset) / 1000.0;  // tiempo en
segundos
    float grams = LoadCell.getData();  // masa en
gramos
    float forceN = grams * 9.81 / 1000.0;  // fuerza en
Newtons

    // Enviar a Phyphox: (t, fuerza, masa)
    PhyphoxBLE::write(t, forceN, grams);

    // Enviar por serie
    Serial.print(t);
    Serial.print(",");
    Serial.print(forceN, 3);
    Serial.print(",");
    Serial.println(grams, 1);
  }
}

void newExperimentEvent() {
  // Reiniciar tiempo al iniciar o reiniciar experimento
  if (PhyphoxBLE::eventType == 1 ||
PhyphoxBLE::eventType == 2) {
    t_offset = millis();
  }
}
```

Calibración de la célula de carga

Para calibrar la célula de carga se ha de determinar el factor de calibración experimentalmente. En el código aparece “const float calFactor = 1119.22;”, y ese número es el que hay que ajustar. Para ello:

1. Poned en marcha el sensor sin carga de manera que indique cero. Cada vez que se resetea la placa el sensor automáticamente se tara a cero.

2. Colocad una masa conocida m (por ejemplo, 100 g = 0.1 kg o cualquier otra de la que se conozca su valor porque se haya pesado sobre una balanza fiable) sobre la célula de carga.

3. La fuerza real que cuelga será 100 g, pero el sensor nos indicará otro valor, por ejemplo 80 g.

4. El nuevo factor de calibración se calcula: Factor nuevo = factor viejo * masa aparente/masa real (por ejemplo, 1119.22*80/100 = 895.38)

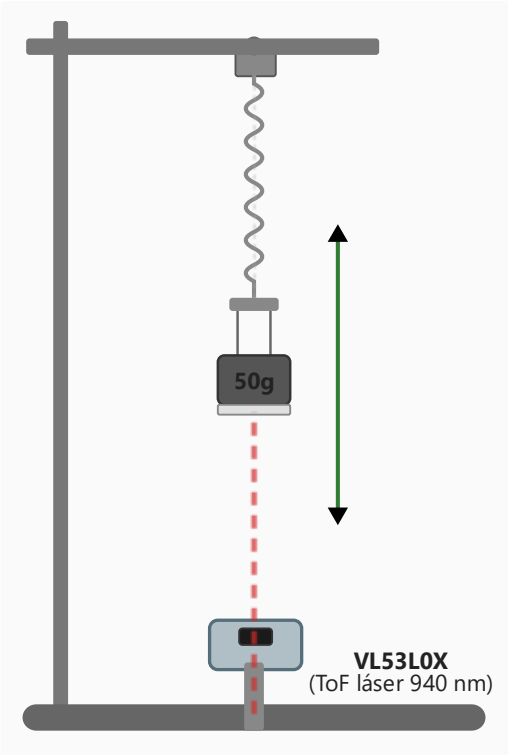
5. Sustituid el nuevo factor de calibración en el código con este valor (en el ejemplo, 895.38) y volver a cargar el código en la placa.

Oscilaciones de un muelle: determinación de la constante elástica mediante sensor de distancia

Aquí os propongo el mismo experimento del caso anterior (*Oscilaciones de un muelle: medida de fuerzas con célula de carga*) pero midiendo las posiciones del objeto que oscila mediante un sensor de distancia. Este planteamiento es complementario al caso del movimiento de un muelle en el aire y se puede realizar a la vez que se mide la fuerza que soporta el muelle en las oscilaciones.

Es decir, vamos a estudiar el movimiento armónico simple (MAS) de una masa suspendida de un muelle midiendo la posición en función del tiempo con un sensor de distancia VL53L0X (ToF láser) o HC-SR04 (ultrasonidos).

Determinaremos después la constante elástica (k) del muelle mediante la ley de Hooke y mediante la relación $T = 2\pi\sqrt{m/k}$ en el caso de las oscilaciones. También podemos verificar la isoacronía del MAS.



Con anterioridad ya se ha hecho este experimento utilizando un sensor de distancia comercial (Aparicio y Lozano, 2002g y h)

Fundamento físico

Ley de Hooke y fuerza recuperadora

Un muelle ideal cumple la ley de Hooke, que indica que la fuerza recuperadora es proporcional al desplazamiento respecto a la posición de equilibrio y de sentido opuesto:

$$[F = -k\,x]$$

donde (F) es la fuerza recuperadora en N, (k) la constante elástica en N/m y (x) el desplazamiento desde la posición de equilibrio en m. El signo negativo indica que la fuerza se opone al desplazamiento, tendiendo a devolver la masa a su posición de equilibrio.

Ecuación diferencial del MAS

Si aplicamos la segunda ley de Newton ($F = m{\cdot}a$) al sistema masa-muelle:

$$m\frac{d^2x}{dt^2} = -kx$$

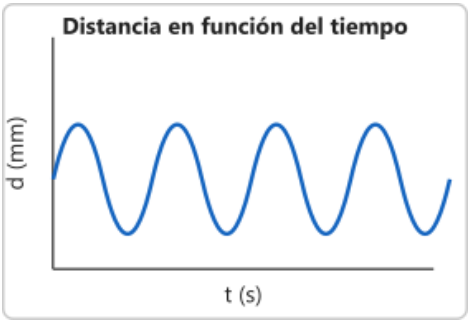
Y la reorganizamos

$$\frac{d^2x}{dt^2} + \frac{k}{m}x = 0$$

Tenemos la ecuación diferencial del oscilador armónico simple, cuya solución general es:

$$x(t) = A\cos(\omega t + \phi), \qquad \omega = \sqrt{\frac{k}{m}}$$

Donde (x) es la posición de la masa que oscila, (A) es la amplitud, (m) la masa colgada, (ω) la frecuencia angular en rad/s y (π) la fase inicial en rad.



Si se deriva la posición respecto al tiempo se obtiene la velocidad:

$$v(t) = -A\,\omega\,\mathrm{sen}(\omega t + \phi)$$

Si se vuelve a derivar tendremos la aceleración:

$$a(t) = -A\,\omega^2\cos(\omega t + \phi)$$

De manera que en los extremos la aceleración es máxima ($\pm A\,\omega^2$), y la fuerza también, y en el centro de la oscilación (cuando x = 0) la aceleración de la masa que cuelga es cero y la fuerza que padece también.

El período de oscilación (T) (tiempo que tarda el sistema en completar una oscilación completa) se relaciona con la frecuencia angular mediante $\omega = 2\pi/T$. Sustituyendo:

$$T = 2\pi\sqrt{\frac{m}{k}}$$

Este resultado indica que el período es independiente de la amplitud (isoacronía del MAS), siempre que nos mantengamos en el régimen lineal de la ley de Hooke (pequeñas oscilaciones).

La energía total del oscilador se conserva (en ausencia de rozamiento) y se intercambia entre energía potencial elástica y energía cinética:

$$E_{\text{total}} = E_p + E_c = \frac{1}{2}kx^2 + \frac{1}{2}mv^2 = \frac{1}{2}kA^2 \text{ (constante)}$$

Materiales y montaje experimental

Se necesita una placa Wemos D1 R32 (ESP32) o Arduino UNO, un sensor de distancia VL53L0X (recomendado) o un HC-SR04, una protoboard, cables Dupont, un muelle helicoidal (k entre 5 y 20 N/m), masas calibradas (juego de pesas de 10 g a 50 g con portapesas), un soporte universal con nuez y pinza, y una pequeña placa reflectante (papel blanco o cartulina) para mejorar la reflexión del láser.

Para más detalles sobre los sensores de distancia, consultad el capítulo “Los sensores”. En todo caso recuerdo:

El VL53L0X es un sensor de distancia basado en tiempo de vuelo (ToF, Time of Flight) de un láser VCSEL (Vertical-Cavity Surface-Emitting Laser) de 940 nm en el infrarrojo cercano (invisible al ojo humano). Emite pulsos láser y mide el tiempo que tarda la luz en reflejarse en el objeto y volver al sensor:

$$d = \frac{c \cdot t}{2}$$

donde d es la distancia en m, c la velocidad de la luz $(3\ \times 10\text{^}8\ m/s)$ y t el tiempo de ida y vuelta del pulso en s. Su rango de medida es de 30 a 1200 mm, con precisión de ±3% y resolución de 1 mm.

El HC-SR04 es un sensor de ultrasonidos que emite pulsos acústicos de 40 kHz (fuera del rango audible humano) y mide el tiempo que tarda el eco en regresar. Su rango es de 2 a 400 cm, con precisión de ±3 mm. Es más lento y menos preciso que el VL53L0X, pero suficiente para estos experimentos y más barato, y funciona sobre cualquier superficie (no requiere una superficie reflectante específica pero funciona mejor si se encuentra con un cuadradito de cartón).

1. Para montar el sistema de medida empezad suspendiendo el muelle de un soporte mediante una nuez y una pinza., y colgad el portapesas del extremo inferior del muelle.

Si utilizáis el sensor VL53L0X, pegad un pequeño cuadrado de papel blanco o cartulina en la parte inferior de las pesas. El VL53L0X funciona mejor con superficies claras y difusas; las superficies oscuras o muy reflectantes (metales brillantes) pueden dar lecturas erráticas.

2. Fijad el sensor de distancia en la base del soporte, apuntando verticalmente hacia la parte inferior de las pesas (o al portapesas).

Colocad el sensor de modo que la distancia al portapesas en equilibrio sea de 10-20 cm (dentro del rango óptimo del VL53L0X: 3-120 cm).

3. Conectad el sensor a la placa:
VL53L0X (I²C):
VCC → 3.3V o 5V (según módulo), GND → GND, SCL → A5 (UNO) o GPIO22 (ESP32), SDA → A4 (UNO) o GPIO21 (ESP32)

HC-SR04:
VCC → 5V, GND → GND, TRIG → pin digital, ECHO → pin digital (con divisor de tensión si es para ESP32 de 3.3V)

Aseguraos de que el sensor tenga visión despejada hacia las pesas y de que las oscilaciones sean verticales, sin balanceo lateral.

Medida estática de k (ley de Hooke)

1. Medid la distancia de equilibrio sin carga adicional (solo portapesas). Esta es vuestra referencia.
2. Añadid masas de 10 g en 10 g hasta 100 g, registrando la distancia de equilibrio para cada masa.
3. La distancia disminuirá al añadir masa (el muelle se estira). Calculad el alargamiento como: $(x = d_0 - d_m)$, donde (d_0) es la distancia sin carga y (d_m) la distancia con masa.
4. La fuerza aplicada es $(F = m \cdot g)$. Representad (F) frente a (x). La pendiente es (k).

Ejemplo de resultados esperables para un muelle con k ≈ 10 N/m

Masa (g)	Fuerza (N)	Alargamiento (mm)	k = F/x (N/m)
10	0,0981	~10	9,8
20	0,1962	~20	9,8
30	0,2943	~30	9,8
50	0,4905	~49	10,0
80	0,7848	~78	10,1
100	0,9810	~97	10,1

Medida dinámica (oscilaciones)

1. Para una masa fija (ej. 50 g), separad la masa unos 2-3 cm hacia abajo de la posición de equilibrio y soltadla.

2. Registrad la distancia en función del tiempo durante al menos 10 oscilaciones completas. La gráfica (x(t)) será una sinusoide amortiguada.

3. El período \(T\) se determina entre dos máximos consecutivos de la distancia (elongación máxima) o entre dos cruces por la posición de equilibrio. Para un muelle con k ≈ 10 N/m y masa m = 50 g se pueden esperar valores como el siguiente:

$$T = 2\pi\sqrt{0.050/10} \approx 0.44\,\text{s}$$

4. Repetid para diferentes masas (20 g, 50 g, 80 g, 100 g). Para cada masa, medid (T) como el intervalo entre dos máximos consecutivos de la distancia (o entre dos cruces por la posición de equilibrio). Al representar (T^2) frente a (m) la pendiente es $(4\pi^2/k)$. Despejad $k = 4\pi^2/\text{pendiente}$.

Como en el experimento anterior, ahora se puede calcular el error relativo entre ambos métodos:

$$\varepsilon = \frac{|k_{\text{Hooke}} - k_{\text{período}}|}{k_{\text{promedio}}} \times 100$$

Que ha de dar valores de menos de $(\varepsilon < 10\%)$.

Y también calcular el coeficiente de amortiguamiento ajustando la envolvente de las oscilaciones a una exponencial decreciente: $(A(t) = A_0 e^{-\gamma t})$. El coeficiente de amortiguamiento (γ) (s^{-1}) refleja las pérdidas por fricción interna del muelle y rozamiento con el aire.

Ampliación

Se pueden plantear los mismos temas de ampliación que en el experimento anterior. Por ejemplo:

Mostrar la Isoacronía del fenómeno repitiendo con diferentes amplitudes (1, 2, 3 cm) para la misma masa para ver si cambia el período. Verificad que (T) es independiente de la amplitud dentro del error experimental.

Estudiar la composición de dos muelles en serie y paralelo conectando dos muelles iguales en serie $(1/k_{\text{eq}} = 1/k_1 + 1/k_2)$ y en paralelo $(k_{\text{eq}} = k_1 + k_2)$. Verificad experimentalmente las leyes de combinación.

Código

```
/*
Oscilaciones de un muelle: medida de distancia con VL53L0X

Mide la posición de una masa oscilante usando un sensor
de distancia láser VL53L0X (ToF) vía I²C.

Conexiones (I²C):
VL53L0X
VCC -> 3.3V o 5V (según módulo)
GND -> GND
SCL -> A5 (UNO) o GPIO22 (ESP32)
SDA -> A4 (UNO) o GPIO21 (ESP32)

Librería: Adafruit_VL53L0X
https://github.com/adafruit/Adafruit_VL53L0X
 */

#include "Adafruit_VL53L0X.h"  // Librería de
```

```
Adafruit para el sensor VL53L0X

Adafruit_VL53L0X lox = Adafruit_VL53L0X();  // Creamos el objeto para el sensor

void setup() {
  Serial.begin(115200);  // Iniciamos comunicación serie a 115200 baudios

  // Inicializamos el sensor VL53L0X
  // El método begin() busca el sensor en el bus I²C y configura los parámetros
  if (!lox.begin()) {
    Serial.println("Error: VL53L0X no detectado. Verificad las conexiones I²C.");
    while (1);  // Si falla la inicialización, detenemos el programa
  }

  Serial.println("VL53L0X inicializado correctamente.");
  Serial.println("Formato: tiempo_ms;distancia_mm");
}

void loop() {
  VL53L0X_RangingMeasurementData_t measure;
  // Estructura que almacena el resultado de la medición:
  //   measure.RangeMilliMeter  -> distancia en mm
  //   measure.RangeStatus      -> 0 = válido, 4 = fuera de rango/error

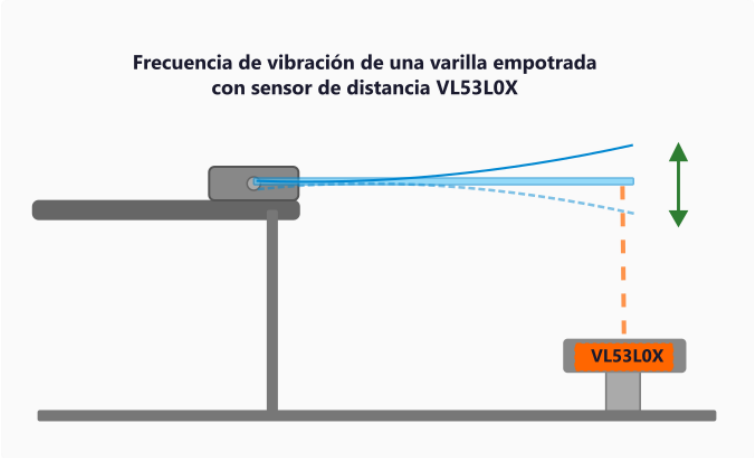
  // Realizamos una medición de distancia
  // El segundo parámetro (false) indica que no se usa depuración por serie
  lox.rangingTest(&measure, false);

  // Solo enviamos datos si la medición es válida
  // RangeStatus = 4 indica que el sensor no pudo medir (fuera de rango o superficie
  // demasiado reflectante/absorbente)
  if (measure.RangeStatus != 4) {
    Serial.print(millis());          // Tiempo en ms desde el arranque
    Serial.print(";");               // Separador CSV (compatible con hojas de cálculo)
    Serial.println(measure.RangeMilliMeter);  // Distancia en mm
  }

  // Frecuencia de muestreo: 100 muestras/s (delay(10))
  // Para un MAS con período típico de 0.5-1.5 s, obtenemos 50-100 puntos por período
  //, suficiente para identificar máximos con precisión
  delay(10);
}
```

Frecuencia de vibración de una varilla empotrada

Determinaremos la frecuencia fundamental de vibración de una varilla metálica (fleje) empotrada en un extremo (voladizo) mediante un sensor de distancia. Verificaremos la relación ($f \propto 1/L^2$) entre la frecuencia y la longitud libre de la varilla, y estimaremos el módulo de Young (E) del material.



Este experimento ya lo proponía Salvador Gil (Gil, 2016), y yo me hacía eco de él en una entrada más extensa (Ramírez, 2021) sobre la medida de la frecuencia de vibración de varillas fijadas por un extremo. Como indico allí, la frecuencia de vibración puede medirse como aquí, pero también de otras maneras y con otros sensores: de campo magnético, con micrófono piezoeléctrico, con fotointerruptor, etc.

La elección del sensor no es indiferente ya que con un sensor de posición solo se pueden seguir vibraciones lentas, mientras que para las rápidas (decenas o cientos de hercios) conviene un sensor que entregue la vibración ya convertida en frecuencia (fotointerruptor, micrófono, pastilla piezoeléctrica), como se comenta en el apartado «Hasta qué frecuencia puede medir el sensor».

Fundamento físico

Una varilla delgada empotrada en un extremo y libre en el otro se comporta como una viga en voladizo y sus vibraciones transversales se rigen por la ecuación de Euler-Bernoulli:

$$EI\frac{\partial^4 y}{\partial x^4} + \rho A\frac{\partial^2 y}{\partial t^2} = 0$$

Donde (E) es el módulo de Young del material (Pa), (I) es el momento de inercia de la sección transversal (m^4), (ρ) es la densidad del material (kg/m^3), (A) es el área de la sección transversal (m^2) y (y(x,t)) es el desplazamiento transversal en la posición (x) y tiempo (t).

Frecuencias de resonancia

Aplicando las condiciones de contorno de una viga empotrada-libre la frecuencia fundamental del primer modo de vibración es:

$$f_1 = \frac{k}{L^2}\sqrt{\frac{EI}{\rho A}}$$

Lo que implica el resultado clave de que la frecuencia fundamental es inversamente proporcional al cuadrado de la longitud libre:

$$f_1 \propto \frac{1}{L^2}$$

Esto significa que el producto ($f_1 \cdot L^2$) debe permanecer constante para una misma varilla. Esta relación es fácilmente verificable experimentalmente variando la longitud (L) de la varilla que sobresale del soporte.

Materiales y montaje experimental

La varilla puede ser un fleje de acero (o aluminio) de aproximadamente 30 cm × 1 cm × 0,5 mm. También puede usarse una regla metálica fina.La varilla puede ser un fleje de acero (o aluminio) de aproximadamente 30 cm × 1 cm × 0,5 mm. También puede usarse una regla metálica fina. Anotad siempre el material del fleje y las dimensiones reales, porque la frecuencia depende de ellas.

Soporte puede ser una pinza de mesa (o un tornillo de banco) para empotrar firmemente la varilla.El soporte puede ser una pinza de mesa o un tornillo de banco, con el que empotrar firmemente la varilla.

Placa Wemos D1 R32 (ESP32) o Arduino UNO, sensor de distancia VL53L0X, protoboard, cables

Dupont y una regla o cinta métrica.Placa Wemos D1 R32 (ESP32) o Arduino UNO, sensor de distancia VL53L0X, protoboard, cables Dupont, una regla o cinta métrica y un pie de rey (calibre) para medir el espesor del fleje.

Para más detalles sobre el VL53L0X, consultad el capítulo de los sensores.

Fijad un extremo de la varilla firmemente en la pinza del soporte universal o en un tornillo de banco. La varilla debe quedar horizontal, con el otro extremo libre.

Medid con una regla la distancia (L) desde el borde de la mordaza (punto de empotramiento) hasta el extremo libre. Esta es la longitud que determina la frecuencia de vibración.La varilla no debe deslizarse ni tener juego. Un empotramiento deficiente falsifica los resultados (la frecuencia medida será menor de la esperada).

Colocad el sensor VL53L0X perpendicular al plano de vibración por encima o por debajo del extremo libre de la varilla, a una distancia de 5–10 cm, apuntando directamente a la cara de la varilla.



Conectad el VL53L0X al microcontrolador mediante I²C:
SDA → GPIO21 (ESP32) o A4 (UNO)
SCL → GPIO22 (ESP32) o A5 (UNO)
VCC → 3,3 V (el VL53L0X funciona a 3,3 V)
GND → GND

La varilla debe oscilar libremente sin tocar el sensor en ningún momento.

Hasta qué frecuencia puede medir el sensor
Cada medida del VL53L0X dura del orden de 30–50 ms, y si a ello se añade el delay(20) del código, el intervalo real entre muestras es de 50–70 ms, es decir, una tasa de 15–20 muestras por segundo (no las 50 que indica el comentario del programa).

Con 4 o 5 muestras por oscilación, que es el mínimo razonable para localizar los extremos con cierta precisión, la frecuencia más alta que se puede medir ronda los 10 Hz.

Para un fleje de acero de 0,5 mm, la tabla de resultados esperados da 20 Hz para L = 25 cm y frecuencias mucho mayores para longitudes menores, de modo que con este sensor solo son utilizables las longitudes largas: L entre 15 y 50 cm. Para longitudes menores y frecuencias más altas hay que cambiar de sensor: fotointerruptor, micrófono, pastilla piezoeléctrica o sensor de efecto Hall con un imán pegado a la varilla (Ramírez, 2021).

Procedimiento

1. Ajustad la longitud libre de la varilla a L = 40 cm (medida desde el borde de la mordaza al extremo libre). Con este sensor, longitudes menores de unos 20 cm dan frecuencias que no se pueden muestrear con suficiente resolución.
2. Desplazad el extremo libre unos 5–10 mm de su posición de equilibrio y soltadlo.
3. El código leerá la distancia en función del tiempo y detectará los máximos para calcular el período.
4. Repetid para longitudes decrecientes: 25, 20, 15, 12, 10, 8, 6 y 5 cm.4. Repetid para longitudes decrecientes: 40, 35, 30 y 25 cm (y longitudes menores solo si disponéis de otro sensor).
5. Para cada longitud, realizad 3 medidas y calculad el promedio.
6. Anotad la frecuencia (f = 1/T) para cada longitud.6. Anotad la frecuencia (f = 1/T) para cada longitud y comprobad que el producto f·L² se mantiene constante.

Resultados esperados

Tabla de frecuencias fundamentales
La siguiente tabla muestra las frecuencias esperadas para un fleje de acero $(\rho = 7800\ \mathrm{kg/m^3})$, $(E = 2{,}0 \times 10^{11}\ \mathrm{Pa})$, de sección 10 mm × 0,5 mm, calculadas con la fórmula de Euler-Bernoulli. El coeficiente 3,516 es la primera raíz de la ecuación característica del modo fundamental de una viga empotrada-libre, no un factor de corrección empírico. Para este fleje, $f_1 \cdot L^2 \approx 0{,}41$ Hz·m².

$$f_1 = \frac{3{,}516}{2\pi L^2}\sqrt{\frac{E \cdot I}{\rho \cdot A}}$$

Frecuencia de oscilación (teórica) para un fleje de acero de 0,5 mm			
L (cm)	**1/L² (m⁻²) (L en m)**	**f_1 (Hz)**	**T (ms)**
50	4,00	1,6	610
45	4,94	2,0	495
40	6,25	2,6	391
36	7,72	3,2	317
30	11,1	4,5	220
25	16,0	6,5	153
20	25,0	10,2	98
15	44,4	18,2	55
10	100	41	24

La frecuencia real puede variar ±15 % según el material, el espesor exacto y la calidad del empotramiento.

El producto f·L² debe ser aproximadamente constante para todas las longitudes si se trata de la misma varilla y del mismo modo de vibración. Para el acero de la tabla ($f_1 \cdot L^2 \approx 0{,}41$ Hz·m²):

$$f \cdot L^2 \approx 1{,}24 \text{ Hz·m}^2$$

He verificado experimentalmente esta relación, obteniendo 10,8 Hz para L = 17,5 cm, lo que da f·L² = 10,8 × 0,175² = 0,33 Hz·m². La diferencia con el valor teórico se debe al espesor real del fleje y a las condiciones de contorno del empotramiento, sobre todo, como he comentado anteriormente, la sujeción de la varilla y la vibración de la mesa; el producto f·L² se mantiene constante dentro de la incertidumbre de las medidas.

Se puede estimar el módulo de Young a partir del valor experimental del producto f·L² = a, despejándolo de la ecuación de la frecuencia:

$$E = \frac{4\pi^2 a^2 \rho A}{3{,}516^2 \cdot I}$$

Donde el momento de inercia de la sección rectangular respecto a la flexión es I = w·e³/12 y el área es A = w·e (w, ancho; e, espesor)

Se puede comprobar la coherencia ya que si se sustituye el valor teórico de la tabla, a = 0,41 Hz·m², se obtiene E = 2,0×10¹¹ Pa, el valor de referencia del acero.

Ampliación

Podríais repetir el experimento si dispusieseis de varillas del mismo espesor pero de diferente material (aluminio, latón, madera, plástico). La frecuencia fundamental depende de $\left(\sqrt{E/\rho}\right)$. ¿Qué material produce la frecuencia más alta? ¿Coincide con el que tiene mayor módulo de Young específico (E/ρ)?

Si se acopla una masa conocida en el extremo libre de la varilla con cinta adhesiva la frecuencia a la que vibra la varilla disminuye. Para masas pequeñas, la corrección puede modelarse como una masa puntual (M) en el extremo de una viga sin masa:

$$f \approx \frac{1}{2\pi}\sqrt{\frac{3EI}{ML^3}}$$

¿Para qué relación (M/m) (masa añadida/masa de la varilla) deja de ser válida esta aproximación?

Código

```
/*
  Frecuencia de vibración de una varilla empotrada (viga en voladizo)
  Sensor de distancia VL53L0X (ToF) sobre I2C.

  El sensor se coloca PERPENDICULAR al plano de vibración (por encima o por
  debajo del extremo libre), nunca frente a la cara del extremo.

  Conexiones:
    SDA -> GPIO21 (ESP32) o A4 (UNO)
    SCL -> GPIO22 (ESP32) o A5 (UNO)
    VCC -> 3,3 V (comprobad si el módulo admite 5 V en VIN)
    GND -> GND

  Librería: Adafruit VL53L0X
  https://github.com/adafruit/Adafruit_VL53L0X

  La tasa real de adquisición es de 15-20 muestras/s (conversión del sensor
   + delay(20)), suficiente solo para vibraciones de menos de 5 Hz. Para
  frecuencias mayores hay que usar otro sensor (fotointerruptor, micrófono).
*/

#include <Wire.h>
#include "Adafruit_VL53L0X.h"

Adafruit_VL53L0X sensor = Adafruit_VL53L0X();
```

```
// --- Parámetros de detección de extremos ---
// min_amplitud_mm: recorrido mínimo (pico a pico) para considerar que hay
// oscilación. Si no se alcanza, no se detectan extremos (evita el ruido).
float min_amplitud_mm = 6.0;
// Un extremo se acepta cuando la distancia se aleja del último extremo más
// de esta fracción de la amplitud observada.
const float histAmp = 0.10;

// --- Filtro de mediana de 5 muestras (elimina picos aislados del sensor) ---
const int N_MED = 5;
int buf[N_MED];
int nBuf = 0;

// --- Detección de extremos ---
bool subiendo = false;          // true si la distancia va en aumento
float d_extremo = 0.0;          // distancia en el último extremo aceptado
unsigned long t_extremo = 0;    // instante del último extremo aceptado
float amplitud = 0.0;           // amplitud pico a pico observada (mm)

// --- Estimación del período ---
const int N_PERIODOS = 3;       // períodos que se promedian
unsigned long periodos[N_PERIODOS];
int nPeriodos = 0;

float mediana() {
  int v[N_MED];
  for (int i = 0; i < N_MED; i++) v[i] = buf[i];
  for (int i = 1; i < N_MED; i++) {        // ordenación por inserción
    int x = v[i], j = i - 1;
    while (j >= 0 && v[j] > x) { v[j + 1] = v[j]; j--; }
    v[j + 1] = x;
  }
  return (float)v[N_MED / 2];
}

void setup() {
  Serial.begin(115200);
  if (!sensor.begin()) {
    Serial.println("Error: VL53L0X no detectado. Verificad las conexiones I2C.");
    while (1);
  }
  // Presupuesto de tiempo bajo = más muestras por segundo (menos alcance).
  // Ajustadlo según la hoja de características de vuestro módulo.
  sensor.setMeasurementTimingBudgetMicroSeconds(20000);

  Serial.println("tiempo_ms;distancia_mm;periodo_ms;frecuencia_Hz");
  delay(1000);
}

void loop() {
  VL53L0X_RangingMeasurementData_t medida;
  sensor.rangingTest(&medida, false);

  // Solo se procesan medidas válidas (RangeStatus 4 = fuera de rango/error)
  if (medida.RangeStatus == 4) {
    delay(20);
    return;
  }
  int d = medida.RangeMilliMeter;
  unsigned long t = millis();

  // Filtro de mediana
  buf[nBuf % N_MED] = d;
  nBuf++;
  if (nBuf < N_MED) { delay(20); return; }
  float df = mediana();

  // Seguimiento de la amplitud pico a pico (con olvido lento)
  static float dmin = df, dmax = df;
  dmin = min(dmin, df);
  dmax = max(dmax, df);
  amplitud = 0.9 * amplitud + 0.1 * (dmax - dmin);

  static bool iniciado = false;
  if (!iniciado) {                  // primer extremo: punto de partida
    d_extremo = df; t_extremo = t; iniciado = true;
    Serial.print(t); Serial.print(';'); Serial.print(df);
    Serial.println(";;");
    delay(20);
    return;
  }

  // Cambio de sentido con histéresis y amplitud suficiente
  if (subiendo && df < d_extremo - histAmp * max(amplitud, 1.0f)
```

```
    && (d_extremo - dmin) > min_amplitud_mm) {
   subiendo = false;
   unsigned long periodo = t - t_extremo;
   if (periodo > 0) {
    periodos[nPeriodos % N_PERIODOS] =
periodo;
    nPeriodos++;
    float suma = 0; int n = min(nPeriodos,
N_PERIODOS);
    for (int i = 0; i < n; i++) suma += periodos[i];
    float t_medio = suma / n;                  // período
medio (ms)
    Serial.print(t); Serial.print(';'); Serial.print(df);
Serial.print(';');
    Serial.print(t_medio, 0); Serial.print(';');
    Serial.println(1000.0 / t_medio, 1);
   }
   t_extremo = t; d_extremo = df; dmin = df; dmax =
df;
  } else if (!subiendo && df > d_extremo + histAmp *
max(amplitud, 1.0f)
           && (dmax - d_extremo) >
min_amplitud_mm) {
   subiendo = true;
   t_extremo = t; d_extremo = df; dmin = df; dmax =
df;
  }

  delay(20);  // ciclo total ~50-70 ms: 15-20
muestras/s
}
```

Oscilaciones amortiguadas de una regla con sensor LDR

A diferencia del experimento anterior en el que no nos deteníamos en el amortiguamiento de las vibraciones, aquí trataremos de registrar y analizar las oscilaciones amortiguadas de una regla flexible utilizando un sensor de luz LDR como sensor de posición sin contacto, midiendo la variación de intensidad luminosa reflejada por un espejo adherido a la regla.En el experimento anterior no nos deteníamos en el amortiguamiento de las vibraciones al suponer que la oscilación era poco amortiguada, que el período no dependía de la amplitud y simplemente lo medíamos.

Aquí el amortiguamiento es justamente el fenómeno que queremos estudiar i para ello registraremos las oscilaciones amortiguadas de una regla flexible utilizando un LDR como sensor de posición sin contacto, midiendo la variación de intensidad luminosa reflejada por un espejo adherido a la regla, y analizaremos cómo decae la amplitud con el tiempo.



El montaje sigue el descrito por Castro, Lago y Mondaini (2015) en el artículo «Damped Harmonic Oscillator with Arduino», con los ajustes necesarios para hacerlo en el laboratorio de un centro de secundaria (alimentación, captura de datos y calibración).

Fundamento físico

Un sistema oscilante real experimenta fuerzas de fricción que disipan energía mecánica progresivamente, dando lugar a oscilaciones amortiguadas. Para pequeñas oscilaciones, el movimiento se describe por $x'' + 2\gamma x' + \omega_0^2 x = 0$, donde $\gamma = b/(2m)$ es la constante de amortiguamiento (s^{-1}) y $\omega_0 = \sqrt{(k/m)}$ la frecuencia angular natural. Cuando el amortiguamiento es débil (régimen subamortiguado, $\gamma < \omega_0$), la solución es:

$$x(t) = A_0 e^{(-\gamma t)} \cdot \cos(\omega_d t + \varphi)$$

con $\omega_d = \sqrt{(\omega_0^2 - \gamma^2)}$

En este montaje, la regla actúa como una lámina flexible (un oscilador de viga en voladizo). Al desplazar el extremo libre y soltarlo, la regla oscila verticalmente con una frecuencia característica determinada por su longitud, sección y módulo de Young, y su punta se comporta, para pequeñas amplitudes, como un oscilador armónico amortiguado.

Para medir el movimiento se pega un espejo plano pequeño en el extremo libre de la regla, que refleja la luz de un LED hacia el LDR, y al oscilar la regla, el punto luminoso reflejado se desplaza sobre el fotorresistor, cambia la resistencia y, con ella, la tensión leída.

Para ángulos pequeños, el desplazamiento del punto luminoso es proporcional al desplazamiento de la punta de la regla, con un factor de amplificación $G \approx (3/2)\cdot(d/L)$, donde d es la distancia del espejo al LDR y L la longitud libre de la regla: un movimiento de pocos milímetros de la punta produce un desplazamiento de varios centímetros del punto luminoso, y la sensibilidad crece al aumentar d.

Materiales

Se necesita una placa Arduino UNO o una Wemos D1 R32 (ESP32), una regla de plástico flexible de 30–50 cm (anotad material y espesor), un LDR (fotoresistencia, tipo GL55), una resistencia fija de 10 kΩ, un espejo plano pequeño de unos 2×2 cm, un LED o una linterna LED, una pinza o soporte de laboratorio (nuez y varilla) para sujetar la regla, una protoboard, cables Dupont, cinta adhesiva o pegamento termofusible y un pie de rey. Conviene además disponer de un tubo opaco corto para limitar el campo de visión del LDR.

Montaje experimental y procedimiento

Pegad el espejo plano en el extremo libre de la regla, con cinta adhesiva o pegamento termofusible, cuidando que quede bien sujeto y plano (su orientación determina hacia dónde se refleja la luz). Sujetad el otro extremo de la regla firmemente con la pinza en el soporte de laboratorio, dejando la regla horizontal y con una longitud libre L conocida (medida desde el borde de la pinza).

Montaje del circuito electrónico. Formad un divisor de tensión con el LDR y la resistencia de 10 kΩ. Conectad una pata del LDR a la alimentación positiva (5 V en el UNO, 3,3 V en el ESP32), la otra pata a una entrada analógica (A0 en el UNO, GPIO34 en el ESP32) y la resistencia de 10 kΩ entre esa misma entrada y GND.

Al aumentar la luz, la resistencia del LDR disminuye y la tensión leída en A0 aumenta; la respuesta no es lineal ni independiente del nivel de iluminación, de ahí la conveniencia de calibrarla.

Colocad el LED por encima del espejo, iluminándolo desde arriba con un haz que llegue oblicuo, y el LDR junto al LED (o ligeramente desplazado), orientado hacia el espejo para recoger el haz reflejado; ambos deben quedar fijos al soporte, de modo que lo único que se mueva sea el espejo solidario a la regla.

Evitad que el LED ilumine directamente el LDR, y para ello interponed un pequeño tubo opaco o una pantalla, de manera que el fotorresistor vea solo la luz que llega del espejo. Después ajustad la posición hasta que, con la regla en reposo, el punto luminoso caiga en el centro de la zona activa del LDR y, al oscilar la regla, se desplace lateralmente sin salirse de ella.

Conectad la placa Arduino al ordenador y cargad el código. Abrid el monitor serie o RealTerm para visualizar los datos. Desplazad ligeramente el extremo libre de la regla hacia abajo (unos 2–3 cm) con el dedo y soltadlo. La regla comenzará a oscilar verticalmente con amplitud decreciente. Capturad los datos durante 5–10 segundos (unas 3000 muestras a 100 Hz). Guardad los datos en un archivo CSV.

Repetid el experimento variando la longitud de la regla que sobresale del soporte (por ejemplo, 30, 25, 20 y 15 cm) y observad cómo cambia la frecuencia de oscilación. También se puede añadir una pequeña masa (un clip, un trozo de plastilina) en el extremo de la regla para estudiar su efecto sobre el período.

Resultados esperados

La gráfica de los datos mostrará una oscilación amortiguada típica: la amplitud de la señal del LDR disminuye exponencialmente con el tiempo. Para una regla de plástico de 30 cm con 20 cm libres, el período teórico es del orden de 0,4 s (unos 2–3 Hz), calculado con la fórmula del experimento anterior y el espesor real de la regla.

El amortiguamiento depende del material y del montaje: en plásticos suele ser moderado y la oscilación se extingue en unos segundos (del orden de 10–30 oscilaciones). Se observará cierta asimetría en la envolvente debida a la no linealidad del LDR y a que su respuesta no es simétrica respecto al punto de equilibrio; es menos crítica para amplitudes pequeñas y dentro de la zona lineal de la curva de calibración.

Los datos pueden organizarse en una tabla como la siguiente:

Longitud libre L (cm)	Período medido T (s)	Frecuencia f (Hz)	$1/L^2$ (m^{-2})
30			
25			
20			
15			

Análisis

Identificad los picos (máximos y mínimos) en la señal. Calculad el período T midiendo el tiempo

entre picos sucesivos y promediando varias medidas. Determinad la frecuencia f = 1/T y la frecuencia angular ω = 2π/T.

Para el análisis del amortiguamiento, representad la amplitud de los picos (valor absoluto respecto al nivel de equilibrio) en función del tiempo en una gráfica semilogarítmica. La pendiente de la recta os dará el coeficiente de amortiguamiento γ.

Determinad el factor de calidad que es Q = ω_d/(2γ) = π/δ. Indica el número de radianes que tarda la amplitud en reducirse a 1/e de su valor inicial, es decir, Q/π oscilaciones, de manera que si Q = 20, la amplitud cae a 1/e tras unas 6 oscilaciones.

Discutid las fuentes de error: la no linealidad del sensor LDR, la posible vibración del soporte, la influencia de corrientes de aire, y las limitaciones de la tasa de muestreo (por ejemplo, 100 Hz es suficiente para un sistema de 10 Hz, pero estaría en el límite).

Ampliación

Se puede estudiar cómo varía el período al cambiar la longitud libre de la regla (debería crecer con L², como en el experimento anterior) o al añadir masa en el extremo.

Código

```
/*
  Registro de oscilaciones amortiguadas de una regla con LDR
  LDR en divisor de tensión con resistencia de 10 kΩ.
  La luz de un LED se refleja en un espejo pegado al extremo de la regla y se
  recoge en el LDR: al oscilar la regla, el punto luminoso se desplaza y varía
  la resistencia del LDR.

  Conexiones (Arduino UNO):
    LDR entre 5 V y A0
    Resistencia de 10 kΩ entre A0 y GND
  En ESP32: LDR entre 3,3 V y GPIO34; 10 kΩ entre GPIO34 y GND.

  Tasa de muestreo nominal: 500 Hz (2000 µs entre muestras).
  La tasa real la fija el enlace serie: a 115200 baudios, una línea de unos
  12 caracteres tarda ~1 ms, así que se obtienen del orden de 300-400
  muestras/s. Comprobadlo con los tiempos del archivo capturado.
*/

const int ldrPin = A0;
const unsigned long intervalo_us = 2000UL;  // 500 Hz nominales
const float alpha = 0.3;                    // filtro paso bajo (0 = sin filtrar)

float ldrFiltrado = 0.0;

void setup() {
  Serial.begin(115200);
  Serial.println("tiempo_ms;LDR");
}

void loop() {
  static unsigned long t0 = micros();
  int valorLDR = analogRead(ldrPin);

  // Filtro exponencial sencillo para reducir el ruido del ADC
  ldrFiltrado = alpha * valorLDR + (1.0 - alpha) * ldrFiltrado;

  // Tiempo en ms con resolución de microsegundos
  unsigned long t = micros() - t0;
  Serial.print(t / 1000.0, 3);
  Serial.print(';');
  Serial.println((int)ldrFiltrado);

  // Esperar hasta completar el intervalo, compensando el tiempo empleado
  while (micros() - t0 < intervalo_us) { }
  t0 += intervalo_us;
}
```

Mecánica de fluidos

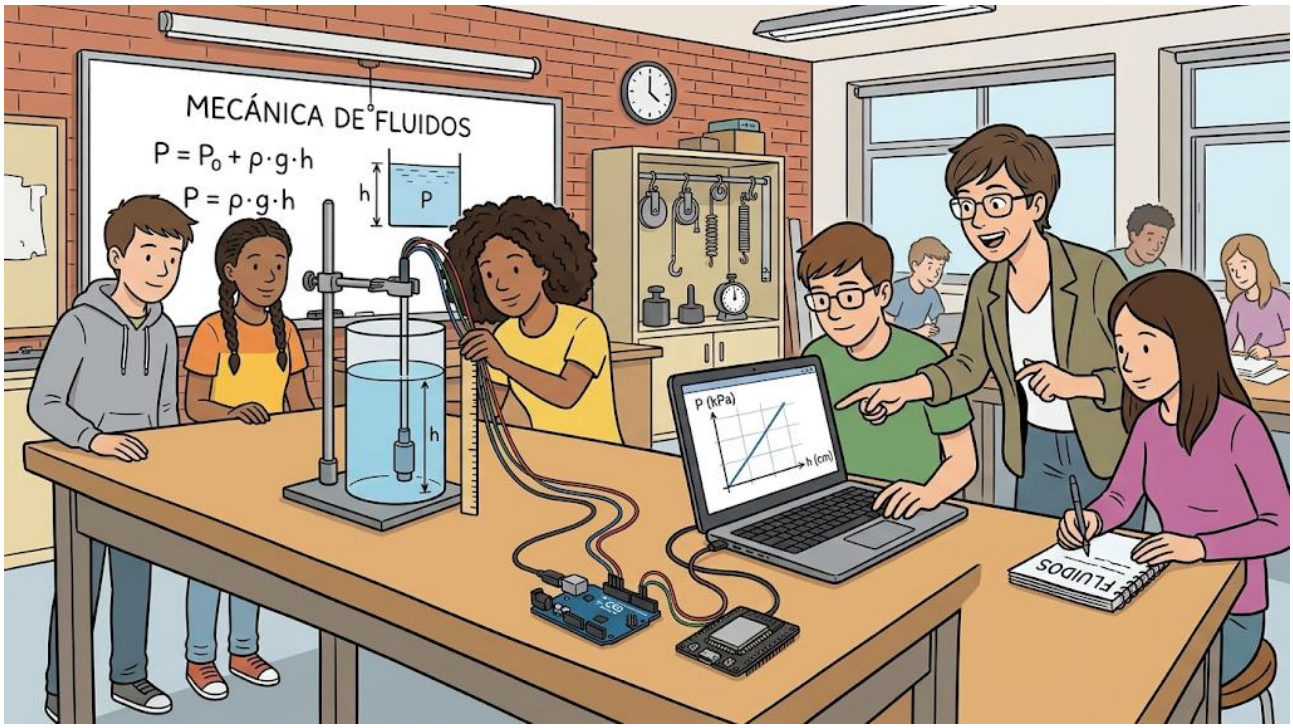
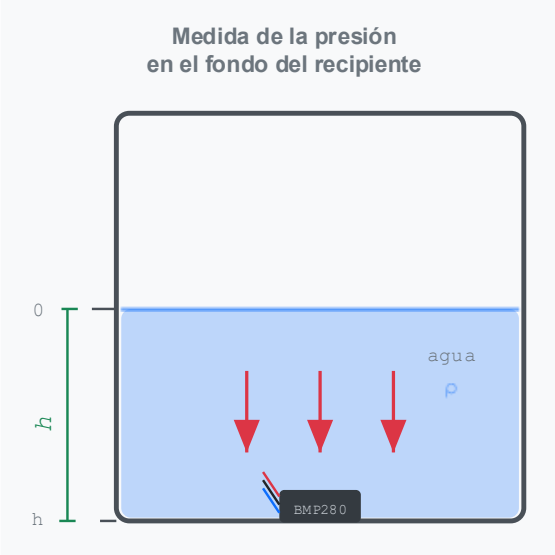


Ilustración realizada con Gemini. Esté pensada para adornar, aunque el alumnado puede entretenerse en buscar los errores evidentes en las conexiones que contiene.

Presión hidrostática y medida del nivel de agua

Determinaremos experimentalmente la relación lineal entre la presión hidrostática y la profundidad en un fluido en reposo, verificando el principio fundamental de la hidrostática ($P = \rho \cdot g \cdot h$). Adicionalmente, utilizar esta relación para medir el nivel de agua o estimar la densidad de distintos fluidos.



Tradicionalmente, este experimento se hacía con un manómetro de agua, como todavía lo hacen los alumnos (Peralta y otros, 2024) de la Universidad de Cuenca en Ecuador.

Si se dispone de sensor de presión comercial se puede utilizar el protocolo de Rosa Mª Melià (2007) para el CDEC.

Fundamento físico

La presión hidrostática a una profundidad (h) en un fluido de densidad (ρ) en reposo depende únicamente de dicha profundidad y de la aceleración de la gravedad (g), según la ecuación de la estática de fluidos:

$$P = P_0 + \rho \cdot g \cdot h$$

Donde (P_0) es la presión atmosférica en la superficie del líquido. Al registrar la presión atmosférica antes de sumergir el sensor, podemos restar este valor de las lecturas obtenidas bajo el agua para encontrar la presión manométrica:

$$P_{man} = P - P_0 = \rho \cdot g \cdot h$$

Esta ecuación lineal indica que, al representar gráficamente P_{man} frente a h, deberíamos obtener una recta que pasa por el origen. La pendiente de esta recta corresponde al producto ρ·g.

Para el agua dulce $\rho \approx 1000$ kg/m^3 y g = 9,81 m/s^2, la pendiente teórica es aproximadamente 9810 Pa/m. Esto significa que cada 10 cm (0,1 m) de columna de agua añaden aproximadamente 981 Pa de presión manométrica. Este principio (descubierto por Arquímedes y formalizado por Pascal) permite no solo calcular la presión en el fondo de embalses o tanques, sino también diseñar sensores de nivel industriales que miden cuánto líquido hay basándose en la presión hidrostática.

Materiales y montaje experimental

Placa Arduino UNO o ESP32 (Wemos D1 R32) y sensor de presión barométrica BMP280 o BME280, protoboard y cables Dupont.
Tubo rígido de plástico o un globo pequeño de látex (para impermeabilizar el sensor).
Probeta cilíndrica alta, probeta graduada o recipiente transparente (al menos 30 cm de altura).
Regla o cinta métrica.
Agua dulce (y opcionalmente agua salada o aceite).

Conexión electrónica
Conectad el sensor BMP280/BME280 a la placa usando el bus I²C:
VCC a 3.3V (o 5V si el módulo tiene regulador incorporado).
GND a GND.
SCL al pin SCL (A5 en Arduino UNO, pin 22 en ESP32).
SDA al pin SDA (A4 en Arduino UNO, pin 21 en ESP32).

Impermeabilización del sensor
Los sensores BMP280 y BME280 no son resistentes al agua. Sumergirlos directamente los cortocircuitará y los destruirá. Hay dos enfoques para realizar la lectura bajo el agua:

Método del tubo de aire: Introducid el sensor dentro de un tubo de ensayo o un tubo largo de plástico. Sellad herméticamente la salida de los cables (con

silicona caliente o masilla epoxi) dejando el extremo del tubo abierto hacia abajo y el sensor en el otro extremo dentro del tubo, completamente seco. Al sumergir el tubo, la presión del agua se transmite a través del material o del aire atrapado.



Método del globo: Introducid el sensor en un globo o preservativo sin lubricante, pasad los cables por la boquilla y selladla fuertemente con bridas o cinta aislante. El látex flexible transmitirá perfectamente la presión del agua al sensor encapsulado. Fijad este encapsulado a una regla o soporte vertical rígido.

Procedimiento

1. Mantened el sensor en el exterior del recipiente, a la altura de la superficie del agua, y dejad que registre la presión base (P_0).
2. Introducid agua en la probeta hasta una altura suficiente (por ejemplo, 35 cm).
3. Sumergid el sensor encapsulado hasta una profundidad exacta de 5 cm. Esperad un par de segundos a que la lectura se estabilice y registrad la presión.
4. Continuad hundiendo el sensor en incrementos de 5 cm (10, 15, 20, 25, 30 cm), anotando las presiones correspondientes.
5. Descargad los datos del monitor serie al ordenador para su procesamiento en una hoja de cálculo.

Para cada profundidad h, calculad la presión manométrica restando el valor inicial P_0.

$$P_{man} = P - P_0.$$

Representad en una gráfica los valores de P_{man} (en el eje Y, en Pascales) frente a la profundidad h (en el eje X, en metros).

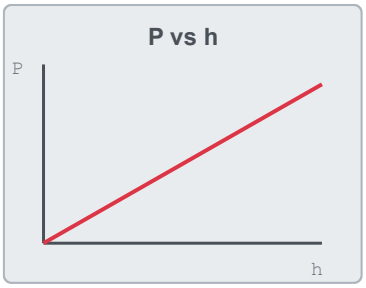
Realizad un ajuste lineal de los puntos experimentales. La recta debería pasar muy cerca del origen (0,0).

A partir de la pendiente de la recta de regresión ρ·g, despejad la densidad del fluido experimental (ρ = pendiente/9,81)

Comparad el valor obtenido con la densidad teórica del agua dulce (1000 kg/m^3) y calculad el error relativo. Evaluad posibles fuentes de desviación: dilatación del recipiente, la tensión superficial si usasteis el método del globo, y la resolución de la regla.

Resultados esperados

Al representar la presión manométrica (presión medida menos presión atmosférica) frente a la altura de la columna de agua, obtendréis una recta que pasa por el origen. La pendiente de esa recta corresponde a ρ·g ≈ 9810 Pa/m para el agua dulce.



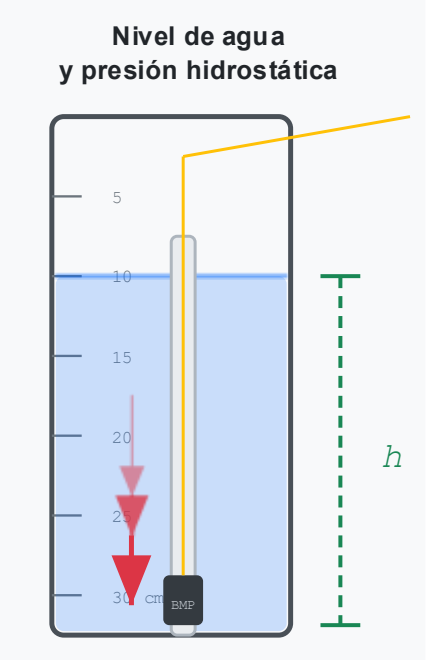
Ampliación

Como la presión hidrostática a una profundidad (h) en un fluido de densidad (ρ) es:

$$P = P_0 + \rho g h$$

donde (P_0) es la presión atmosférica en la superficie, se puede calcular la profundidad a la que está sumergido el sensor simplemente despejando:

$$h = \frac{P - P_0}{\rho g}$$



Para comprobar la universalidad de la ecuación, repetid las medidas sustituyendo el agua dulce por agua saturada en sal (donde la densidad puede subir hasta $\approx 1150\ \text{kg/m}^3$) o aceite vegetal ($\approx 900\ \text{kg/m}^3$). Las pendientes de las nuevas gráficas serán, de forma directa, visiblemente distintas, demostrando el efecto de la densidad en la presión en el fondo de un recipiente.

Código

El siguiente programa lee constantemente la presión en pascales y la envía a través del puerto serie. Se utiliza la biblioteca estándar Adafruit_BMP280, compatible con los sensores barométricos más comunes en este tipo de montajes. No hace falta enviarlas por bluetooth a Phyphox ya que las medidas que se necesitan son puntuales para cada altura sumergida.

```
/*
Medida de presión hidrostática y nivel de líquido
Este código lee los datos de un sensor BMP280 y los envía al monitor serie
en formato CSV (separado por puntos y comas) para facilitar su importación a Excel u hojas de cálculo.

Conexiones I2C:
VCC -> 3.3V o 5V
GND -> GND
SCL -> a SCL - A5 (Arduino UNO) o GPIO 22 (ESP32)
SDA -> a SDA - A4 (Arduino UNO) o GPIO 21 (ESP32)
*/

#include <Wire.h>
#include <Adafruit_BMP280.h>

Adafruit_BMP280 bmp; // Creamos el objeto del sensor barométrico

void setup() {
  Serial.begin(9600); // Iniciamos el puerto serie

  // Inicializamos el sensor en la dirección típica I2C (0x76 o 0x77)
  if (!bmp.begin(0x76)) {
    Serial.println("¡Error! No se encuentra el sensor BMP280. Revisa las conexiones.");
    while (1) {
      // Detenemos el programa si hay un error
      delay(10);
    }
  }

  // Configuración del sensor recomendada para bajo ruido
bmp.setSampling(Adafruit_BMP280::MODE_NORMAL,     // Modo continuo
Adafruit_BMP280::SAMPLING_X2,  // Sobremuestreo temperatura
Adafruit_BMP280::SAMPLING_X16,  // Sobremuestreo presión (máximo)
Adafruit_BMP280::FILTER_X16,  // Filtrado (reduce el ruido del agua)
Adafruit_BMP280::STANDBY_MS_500); // Medio segundo entre medidas

  Serial.println("--- Inicio del experimento de hidrostática ---");
  Serial.println("Tiempo(ms);Presion(Pa)");
}

void loop() {
  // Leemos el tiempo transcurrido
unsigned long tiempo = millis();

  // Leemos la presión absoluta en Pascales
float presion_Pa = bmp.readPressure();

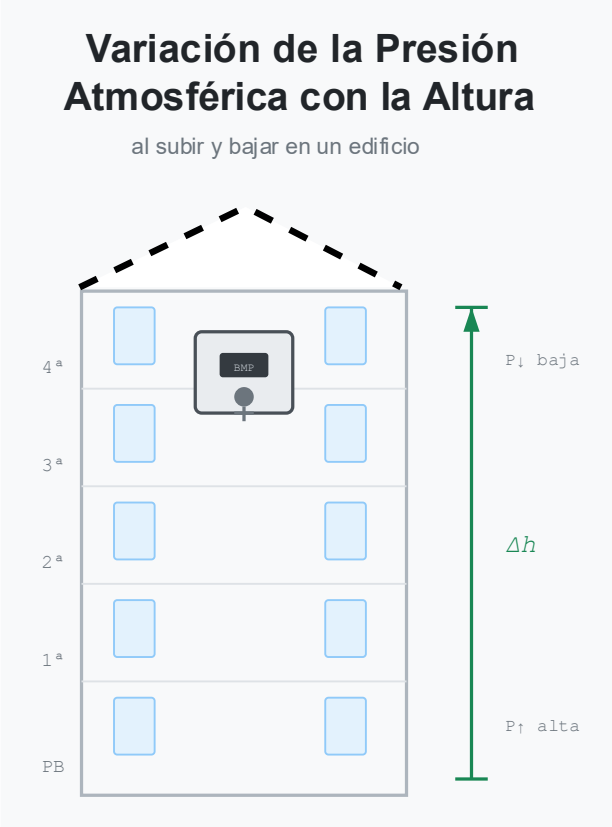
  // Imprimimos los valores separados por punto y coma
```

```
  Serial.print(tiempo);
  Serial.print(";");
  Serial.println(presion_Pa, 1); // 1 decimal de
precisión

  delay(500); // Tomamos dos muestras por
segundo
}
```

Variación de la presión atmosférica con la altura

Medir la variación de la presión atmosférica con la altura utilizando un sensor de presión barométrica (BMP280 o BME280) conectado a un Arduino o ESP32, desplazándoos entre diferentes alturas (por ejemplo, distintos pisos de un edificio) y verificando la relación proporcional entre presión y altitud.



Si se dispone de un teléfono con sensor barométrico (solo lo tienen los caros) se puede hacer este experimento de la misma forma (Ramírez, 2018a).

Fundamento físico

La presión atmosférica es la fuerza por unidad de área que ejerce la columna de aire sobre la superficie terrestre. Disminuye con la altura porque la cantidad de aire que hay por encima cada vez es menor. Para pequeñas variaciones de altura (hasta unos 1000 m), la relación entre presión y altitud se puede aproximar mediante la fórmula barométrica simplificada:

$$p(h) = p_0 \cdot e^{(-h/H)}$$

donde p_0 es la presión a nivel del mar (≈ 1013,25 hPa), h la altura sobre el nivel del mar, y H = RT/(Mg) la altura de escala (≈ 8400 m a 15 °C), siendo R la constante de los gases ideales, T la temperatura absoluta, M la masa molar media del aire y g la aceleración de la gravedad.

Para desniveles pequeños (decenas de metros, lo que será normalmente nuestro caso), la variación es aproximadamente lineal:

$$\Delta p \approx -\rho \cdot g \cdot \Delta h$$

donde ρ es la densidad del aire (≈ 1,225 kg/m³ a nivel del mar). Esto implica que al subir 8 m, la presión disminuye aproximadamente 1 hPa.

El sensor BMP280 mide presión barométrica con una precisión de ±1 hPa y la temperatura, comunicándoos por I²C o SPI. Incorpora un sensor de temperatura para compensar las variaciones térmicas del propio sensor, mejorando la precisión.

Materiales

Necesitaréis una placa Wemos D1 R32 (ESP32) o Arduino UNO, un sensor BMP280 (o BME280, que incluye además humedad), cables Dupont, una protoboard, un power bank o batería portátil para alimentar la placa durante los desplazamientos, y opcionalmente un teléfono móvil con Phyphox para visualizar los datos en tiempo real.

Montaje experimental y procedimiento

Conectad el BMP280 a la placa por I²C: VCC a 3,3V (o 5V según el módulo), GND a GND, SCL al pin SCL (A5 en Arduino UNO, pin 22 en ESP32) y SDA al pin SDA (A4 en UNO, pin 21 en ESP32). Si el módulo tiene un pin CS, dejadlo sin conectar (o a 3,3V para seleccionar modo I²C).

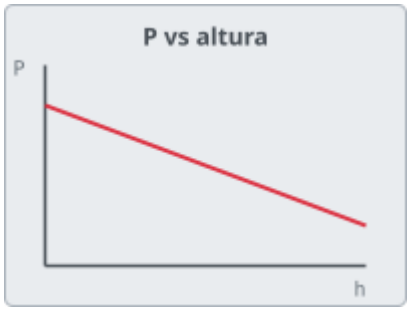
Cargad el código con conectividad BLE y, a continuación, montad la placa y el sensor en un soporte portátil (una caja pequeña o directamente sobre la protoboard) y conectad la alimentación mediante el power bank. Si usáis ESP32 con

Phyphox, podréis observar la presión en tiempo real en el móvil mientras se suben y bajan pisos andando o en ascensor. La gráfica mostrará una disminución al subir y un aumento al bajar.

Resultados esperados

Al subir del primer al último piso de un edificio típico (unos 20–30 m de desnivel), la presión atmosférica disminuirá aproximadamente 2,5–3,5 hPa, y la altitud calculada por el sensor aumentará aproximadamente en el mismo desnivel.

La relación entre pisos consecutivos (unos 3 m cada uno) mostrará una disminución de presión de unos 0,35–0,40 hPa por piso. La gráfica de presión frente al tiempo mostrará una forma escalonada si os paráis en cada piso, con una caída continua durante el ascenso en ascensor y una meseta cuando la puerta se abre.



Representad la presión en función de la altitud (o del número de piso). Calculad la diferencia de presión entre el punto más bajo y el más alto.

Para el análisis lineal (válido para pequeños desniveles), calculad $\Delta p/\Delta h$ y comparad con el valor teórico $-\rho \cdot g \approx -0{,}12$ hPa/m. Discutid las fuentes de error: cambios de temperatura durante el trayecto, corrientes de aire en el hueco del ascensor, precisión del sensor (±1 hPa), deriva térmica del sensor.

Ampliación

Repetid el experimento en un ascensor de un edificio alto (más de 10 pisos) y comparad la precisión de la estimación de altura con la obtenida mediante un sensor de distancia.

Combinad el BMP280 con un GPS para medir altitud por satélite y comparad ambos métodos.

También se puede usar el sensor para detectar cambios de presión debidos al paso de frentes meteorológicos dejando el dispositivo fijo durante 24 horas y registrando la presión cada minuto (cambiando el delay a 60000).

Código

```
/*
Medida de presión atmosférica con BMP280 y
Phyphox. Placa: ESP32 DEV Module

Conexiones I²C:
VCC → 3,3V
GND → GND
SCL → pin 22 (ESP32)
SDA → pin 21 (ESP32)
*/

#include <Wire.h>
#include <Adafruit_BMP280.h>
#include <phyphoxBle.h>

Adafruit_BMP280 bmp; // Sensor BMP280 por I²C

void setup() {
  Serial.begin(9600);
  Wire.begin();

  if (!bmp.begin(0x76)) {  // Dirección I²C típica del
BMP280 (0x76 o 0x77)
    Serial.println("Sensor BMP280 no encontrado");
    while (1);
  }

  // Configuración por defecto: presión y
temperatura

bmp.setSampling(Adafruit_BMP280::MODE_NOR
MAL,
                Adafruit_BMP280::SAMPLING_X2,
                Adafruit_BMP280::SAMPLING_X16,
                Adafruit_BMP280::FILTER_X16,
                Adafruit_BMP280::STANDBY_MS_500);

  PhyphoxBLE::start("Barometro");

  PhyphoxBleExperiment experimento;
  experimento.setTitle("Presión atmosférica");
  experimento.setCategory("Arduino Experiments");
  experimento.setDescription("Presión barométrica
(hPa) frente al tiempo.");
```

```
  PhyphoxBleExperiment::View view;

  PhyphoxBleExperiment::Graph graphPresion;
  graphPresion.setLabel("Presión");
  graphPresion.setUnitX("s");
  graphPresion.setUnitY("hPa");
  graphPresion.setLabelX("tiempo");
  graphPresion.setLabelY("Presión");
  graphPresion.setChannel(1, 2);

  PhyphoxBleExperiment::Graph graphAltura;
  graphAltura.setLabel("Altitud");
  graphAltura.setUnitX("s");
  graphAltura.setUnitY("m");
  graphAltura.setLabelX("tiempo");
  graphAltura.setLabelY("Altitud");
  graphAltura.setChannel(1, 3);

  view.addElement(graphPresion);
  view.addElement(graphAltura);
  experimento.addView(view);
  PhyphoxBLE::addExperiment(experimento);
}

void loop() {
  float t = 0.001 * millis();
  float presion = bmp.readPressure() / 100.0; //
Convertir Pa a hPa
  float altitud = bmp.readAltitude(1013.25);  // Altitud
(m) con referencia a “nivel del mar”

  PhyphoxBLE::write(t, presion, altitud);

  Serial.print(t, 1);
  Serial.print("; ");
  Serial.print(presion, 1);
  Serial.print(" hPa; ");
  Serial.print(altitud, 1);
  Serial.println(" m");

  delay(100);
}
```

Presión en el interior de un globo

Todos hemos comprobado alguna vez que es más fácil seguir hinchando un globo una vez que ya está un poco inflado En este experimento vamos a medir la presión en el interior de un globo mientras se infla y se desinfla, analizando la relación entre la presión interna, el volumen y la elasticidad de la membrana del globo. A partir de aquí se puede verificar cualitativamente la ley de Laplace para membranas elásticas.



Esto ya lo había hecho utilizando el sensor de presión de un teléfono móvil que introduje (con muchas dificultades) en un globo de caucho (Ramírez, 2018b).

La mayor parte de los teléfonos móviles no disponen de sensor de presión (y los que los tienen son los más caros) y además los sensores de presión BME280(BMP280) que utilizaremos en este caso son muy pequeños, baratos y se introducen fácilmente en cualquier globo, así que esta es una mucho mejor opción para estudiar la presión en el interior de globos.

Fundamento físico

Ley de Laplace para membranas elásticas

Para una membrana esférica delgada sometida a una presión interna (Pi) y externa (Pe), la diferencia de presión (ΔP) se relaciona con la tensión superficial de la membrana (σ) y el radio (R):

$$\Delta P = P_i - P_e = \frac{2\sigma}{R}$$

En un globo, la tensión σ no es constante, sino que varía con el grado de estiramiento del material (caucho) conforme se infla o desinfla. Esto produce un comportamiento peculiar:

Al iniciar el inflado se necesita mucha presión para empezar a estirar el caucho (alta rigidez inicial). Una vez inflado la presión necesaria disminuye a medida que el globo se hace más grande, hasta un mínimo. A partir de aquí y hasta que estalla la presión vuelve a aumentar hasta el límite elástico del material.

Ecuación de estado del gas interior

Suponiendo que el aire dentro del globo se comporta como un gas ideal se rige por ecuación de estado de los gases

$$P_i V = nRT$$

Donde (Pi) es la presión absoluta interior, (V) el volumen, (n) la cantidad de sustancia, (R) la constante de los gases y (T) la temperatura, de manera que al variar el producto P·V cambia también su temperatura.

Materiales y montaje experimental

Placa ESP32, Sensor BMP280 / BME280, globos de caucho de diferente tamaño y grosor, protoboard y cables dupont.

También una cámara de fotos o un móvil para fotografiar o gravar el globo (con escala, por ejemplo una regla).



Hay que introducir el sensor BMP280 dentro del globo. Una vez acabado el experimento sacad el sensor del globo ya que cuando se infla soplando se introduce humedad (el sensor soporta humedad, pero conviene no abusar).

Conectad el sensor al microcontrolador según la tabla siguiente y cargad el experimento en Phyphox

BMP280	ESP32
VCC	3,3 ó 5V
GND	GND
SCL	GPIO22
SDA	GPIO21

Con el globo vacío y abierto (presión interior = presión atmosférica) se registra el valor de (P_0) del sensor. Esta lectura servirá como referencia de presión cero ($\Delta P = 0$).

Se infla un poco el globo y se vuelve a anotar la presión. A la vez se hace una foto o se empieza a gravar un vídeo del globo con un objeto al lado de dimensiones conocidas, por ejemplo una regla.

Se repite el proceso con volúmenes crecientes hasta que el globo esté muy hinchado.

Una vez hinchado el globo se pega un trocito de cello en su superficie y se pincha con un alfiler para dejar escapar el aire lentamente mientras se registra la presión en función del tiempo y se sigue grabando en video para poder posteriormente con el software Tracker medir el radio del globo y, por tanto, su volumen aproximado.

No conviene dejar escapar el aire por la boca del globo ya que por el efecto de acción-reacción tiende a colocarse el sensor en la boca por donde sale el aire a gran velocidad y, por tanto (por el efecto Venturi) disminuyendo mucho la presión del aire que sale y falsificándose los datos de la presión dentro del globo.

Los datos de presión obtenidos en Phyphox (o a través del puerto serie) se exportan a una hoja de cálculo o un programa de tratamiento de datos y se calcula la sobrepresión ($\Delta P = P_i - P_0$) en el interior del globo.

Se analiza el video con Tracker (usa una referencia de escala) y se miden los diámetros de del globo para diferentes tamaños y se relacionan con las diferentes presiones

También se pueden relacionar las diferentes presiones con la temperatura interna.

Resultados esperados

Lo esperado se puede resumir en las siguientes tablas:

Presión–radio del globo

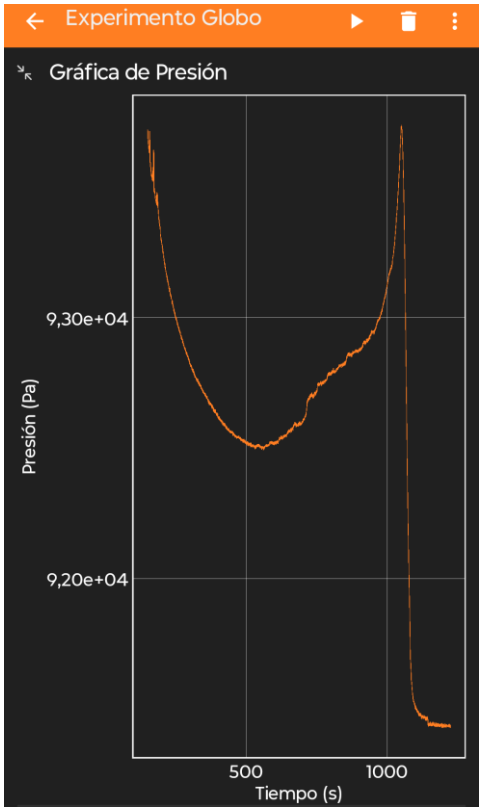
Estado del globo	ΔP esperada	Comportamiento
Vacío (R pequeño)	Alta (arranque)	La rigidez inicial del caucho requiere mayor presión
Semihinchado (R medio)	Mínima	El caucho está más distendido
Muy hinchado (R grande)	Creciente	El caucho se acerca al límite elástico

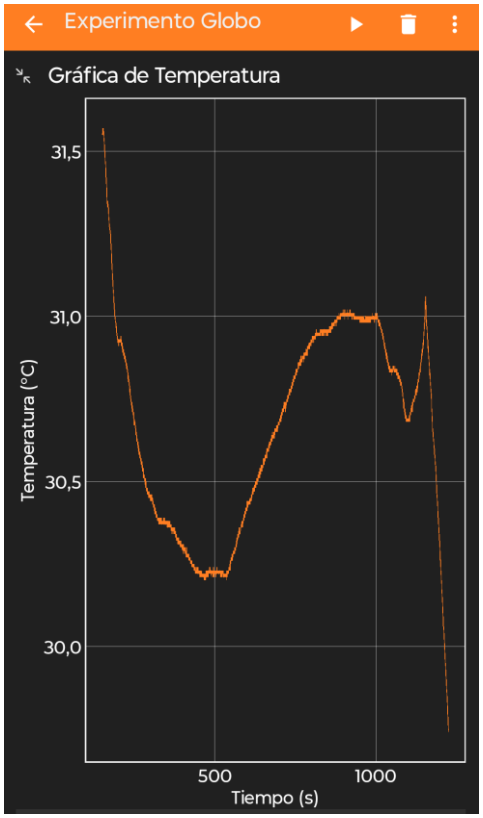
Ejemplo de tabla de datos

Volumen inyectado (mL)	Presión absoluta (Pa)	ΔP (Pa)	Radio (cm)
0	101 325	0	—
50	102 100	775	3,2
100	102 450	1 125	4,0
150	102 300	975	4,6
200	102 800	1 475	5,1

En un caso concreto he obtenido la curva de la captura de la pantalla de Phyphox siguiente donde se ve la gráfica de la presión en el interior de un globo a lo largo del tiempo mientras se está deshinchando.

Comienza con la presión alta cuando el globo está muy hinchado, disminuye mientras va disminuyendo su tamaño hasta que es suficientemente pequeño para que el grosor de la goma sea suficientemente grande para que vuelva a aumentar la presión, hasta que llega un momento en el que se deshincha del todo y la presión interior baja rápidamente hasta la presión atmosférica.





Gráfica de la temperatura en el interior del globo conforme se está deshinchando. Conforme la presión y el volumen van descendiendo lo hace a la vez la temperatura, que vuelve a subir cuando lo hace la presión.

Análisis de los datos

Si se representa la gráfica de la sobrepresión (ΔP) frente al radio del globo se puede identificar donde se encuentra el mínimo.

Si se calcula $(\sigma = \Delta P \cdot R/2 \; - \; \text{ley de Laplace})$ para diversos puntos se observa cómo varía la tensión de la membrana con el estiramiento.

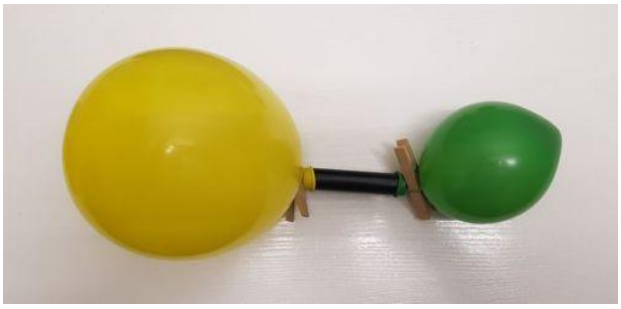
Ampliación

Se puede comparar los valores del inflado con los de desinflado para comprobar que el caucho presenta histéresis mecánica.

Si se dispone de diferentes tipos de globo, incluso de diferente composición material, también se puede hacer la comparación entre ellos

Si se introduce un globo pequeño dentro de uno grande se puede medir la presión en el espacio entre ellos.

¿Qué ocurre si conectas dos globos iguales pero diferentemente hinchados (diferente radio) mediante un tubo? El flujo de aire va del grande al pequeño o al revés.



Modelo termodinámico: Midir la temperatura, la presión y el volumen del globo inmediatamente después de inflar rápidamente el globo (aproximación adiabática) y comprobar hasta qué punto se cumple la ley de los gases $(\mathrm{P_i V = nRT})$.

Código

```
/*
Mide presion absoluta con BMP280 para
experimento con globo
y envía los datos por Bluetooth a Phyphox
*/

#include <Wire.h>              // Libreria para
comunicacion I2C
```

```
#include <Adafruit_BMP280.h>    // Libreria del sensor de presion BMP280
#include <phyphoxBle.h>         // Libreria para enviar datos a Phyphox via Bluetooth

Adafruit_BMP280 bmp;            // Creamos el objeto para el sensor BMP280

void setup() {
  // Configuracion inicial
  Serial.begin(9600);
  while (!Serial);

  if (!bmp.begin(0x76)) {
    Serial.println("BMP280 no encontrado. Verifica conexiones I2C.");
    while (1);
  }

  // Configuramos el modo de medida del BMP280 en modo NORMAL
  // (Cambiado de FORCE a NORMAL para que tome lecturas continuas sin tener que forzarlas manualmente)

bmp.setSampling(Adafruit_BMP280::MODE_NORMAL,
                Adafruit_BMP280::SAMPLING_X1,
                Adafruit_BMP280::SAMPLING_X1,
                Adafruit_BMP280::FILTER_OFF);

  Serial.println("Tiempo(s);Presion(Pa);Temp(C)");

  // CONFIGURACIÓN DE PHYPHOX
  PhyphoxBLE::start("Globo_BMP280"); // Nombre que aparecerá en el Bluetooth

  // Creamos el experimento que se enviará a la app
  PhyphoxBleExperiment experimento;
  experimento.setTitle("Experimento Globo");
  experimento.setCategory("Experimentos");
  experimento.setDescription("Mide la presión y temperatura durante el vuelo del globo.");

  // Creamos una vista (pestaña) en la app
  PhyphoxBleExperiment::View vistaPrincipal;
  vistaPrincipal.setLabel("Datos en vivo");

  // 1. Bloque de valor absoluto: Presión
  PhyphoxBleExperiment::Value valPresion;
  valPresion.setLabel("Presión Absoluta");
  valPresion.setUnit("Pa");
  valPresion.setChannel(2); // Leerá el dato del canal 2

  // 2. Gráfica: Presión vs Tiempo
  PhyphoxBleExperiment::Graph graphPresion;
  graphPresion.setLabel("Gráfica de Presión");
  graphPresion.setUnitX("s");
  graphPresion.setUnitY("Pa");
  graphPresion.setLabelX("Tiempo");
  graphPresion.setLabelY("Presión");
  graphPresion.setChannel(1, 2); // Eje X: Canal 1 (Tiempo), Eje Y: Canal 2 (Presión)

  // 3. Bloque de valor absoluto: Temperatura
  PhyphoxBleExperiment::Value valTemp;
  valTemp.setLabel("Temperatura");
  valTemp.setUnit("°C");
  valTemp.setChannel(3); // Leerá el dato del canal 3

  // 4. Gráfica: Temperatura vs Tiempo
  PhyphoxBleExperiment::Graph graphTemp;
  graphTemp.setLabel("Gráfica de Temperatura");
  graphTemp.setUnitX("s");
  graphTemp.setUnitY("°C");
  graphTemp.setLabelX("Tiempo");
  graphTemp.setLabelY("Temperatura");
  graphTemp.setChannel(1, 3); // Eje X: Canal 1 (Tiempo), Eje Y: Canal 3 (Temp)

  // Añadimos los elementos a la vista (el orden importa para la visualización)
  vistaPrincipal.addElement(valPresion);
  vistaPrincipal.addElement(graphPresion);
  vistaPrincipal.addElement(valTemp);
  vistaPrincipal.addElement(graphTemp);

  // Añadimos la vista al experimento y el experimento a Phyphox
  experimento.addView(vistaPrincipal);
  PhyphoxBLE::addExperiment(experimento);
}

void loop() {
  // Capturamos el tiempo en segundos (mejor para las gráficas en Phyphox)
  float t_segundos = millis() / 1000.0;

  float presion = bmp.readPressure();
  float temperatura = bmp.readTemperature();

  // Mantenemos el envío por puerto serie
  Serial.print(t_segundos, 2);
```

```
  Serial.print(";");
  Serial.print(presion, 1);
  Serial.print(";");
  Serial.println(temperatura, 2);

  // Enviamos los datos a Phyphox por Bluetooth
  // Canal 1: t_segundos | Canal 2: presion | Canal
3: temperatura
  PhyphoxBLE::write(t_segundos, presion,
temperatura);

  delay(100);
}
```

Medida de la presión en el interior de una burbuja

Este experimento es casi igual al anterior en el que medíamos la presión en el interior de un globo según se iba hinchando y deshinchando. Ahora pretendemos medir la sobrepresión en el interior de una burbuja de jabón en función de su radio y verificar la ley de Laplace para burbujas esféricas, así como determinar el coeficiente de tensión superficial del líquido jabonoso.



Fundamento físico

Una burbuja de jabón tiene dos superficies (interior y exterior de la película líquida). La diferencia de presión entre el interior y el exterior viene dada por la ley de Laplace para una película esférica de dos caras:

$$\Delta P = P_i - P_e = \frac{4\sigma}{R}$$

Donde (ΔP) es la sobrepresión (Pa), (σ) es la tensión superficial del líquido jabonoso (N/m) y (R) es el radio de la burbuja (m).

Para una burbuja de gas en un líquido (una sola superficie), la expresión es:

$$\Delta P = \frac{2\sigma}{R}$$

La tensión superficial surge de las fuerzas de cohesión entre las moléculas del líquido. En el agua pura a 20 °C, $(\sigma \approx 0{,}073\ \mathrm{N/m})$. El jabón reduce esta tensión a ~0,025–0,030 N/m, estabilizando la burbuja.

La ley de Laplace predice que burbujas más pequeñas tienen mayor sobrepresión. Esto explica por qué una burbuja pequeña conectada a una grande se vacía dentro de la grande: el aire fluye de mayor presión a menor presión.

Materiales y montaje experimental

Arduino UNO o ESP32, sensor BMP280/BME280, solución jabonosa (agua + jabón de platos + glicerina, o de compra), tubo fino (pajita rígida, 3–5 mm Ø) para soplar burbujas, protoboard y cables dupont

También una cámara de fotos o un móvil para fotografiar o gravar las burbujas (con escala, por ejemplo una regla).

Conexiones del sensor a la placa

BMP280	Arduino UNO	ESP32
VCC	3,3 V	3,3 V
GND	GND	GND
SCL	A5 (SCL)	GPIO22
SDA	A4 (SDA)	GPIO21

Conectad el sensor a la placa según las conexiones anteriores y abrid el terminal serie del IDE de Arduino para ver los valores del sensor.

Con el sistema abierto a la atmósfera, registrad la presión ambiente (P_0).

Sobre una mesa mojada con la solución jabonosa generad con la pajita las burbujas cerca de una regla. Con un móvil gravad la burbuja que generáis.

Ahora hay que mojar el sensor en el líquido jabonoso e introducirlo directamente en la burbuja. Acabado el experimento secad rápidamente el sensor (el sensor soporta humedad, pero conviene no abusar).

Se pueden repetir las medidas para media docena radios diferentes, desde burbujas pequeñas (R ≈ 3 cm) hasta grandes (R ≈ 10 cm).

Otra opción es la de inflar sobre el sensor una burbuja lentamente desde R ≈ 0 hasta su tamaño máximo (con el sistema grabando). La presión variará de forma continua y se capturarán los datos en tiempo real.

Resultados esperados

Relación (ΔP) vs radio

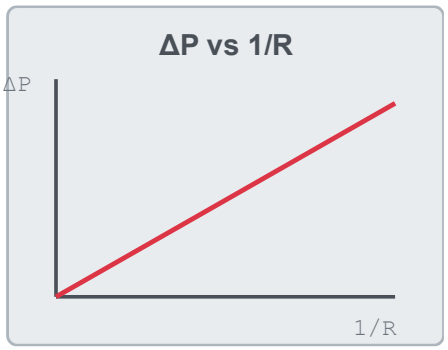
Radio (mm)	(ΔP) teórica en Pa para $(\sigma = 0,028\ \mathrm{N/m})$
3	37,3
5	22,4
10	11,2
15	7,5
20	5,6
30	3,7

Determinación de (σ)

Ajustando los datos experimentaos a la ley de Laplace:

$$\Delta P = \frac{4\sigma}{R} \Rightarrow \sigma = \frac{\Delta P \cdot R}{4}$$

Cada par $(R, \Delta P)$ da un valor de (σ). El promedio proporciona la tensión superficial de la solución jabonosa. También se pueden graficar los valores (ΔP) en Pa frente a (1/R) en m^{-1}. La relación debe ser lineal con pendiente (4σ). Si comparamos con agua pura el valor de (σ) para la solución jabonosa debe ser 2–3 veces menor que el del agua (0,073 N/m).



La estudiante Paula Gómez (Gómez, 2023) en su trabajo de investigación de fin de bachillerato, *Bombolles... més que un divertiment*, midió con este método la presión en el interior de burbujas de jabón obteniendo diferencias de presión del orden de los 10 Pa para radios de 9,0 cm.

Ampliación

¿Qué ocurre si conectas dos burbujas de diferente radio mediante un tubo? El flujo de aire va de la grande a la pequeña o al revés.

Observad si (ΔP) disminuye en el tiempo mientras la burbuja permanece estable (por difusión de gas a través de la película).

Se pueden preparar soluciones con diferentes concentraciones de jabón y comparar los valores de (σ).

Medir el efecto de la temperatura, por ejemplo calentando la solución jabonosa (30–50 °C) y midiendo cómo disminuye (σ).

Código

```
/*
Mide la sobrepresión en burbujas de jabón con el sensor BMP280
Salida de datos por el puerto serie
*/

#include <Wire.h>  // Libreria para comunicacion I2C (bus del BMP280)
#include <Adafruit_BMP280.h>  // Libreria del sensor de presion BMP280

Adafruit_BMP280 bmp;  // Creamos el objeto para el sensor BMP280

void setup() {  // Configuracion inicial: se ejecuta una vez al arrancar la placa
Serial.begin(9600);  // Iniciamos comunicacion serie a 9600 baudios
while (!Serial);  // Esperamos a que el puerto serie se conecte

if (!bmp.begin(0x76)) {  // Intentamos inicializar el BMP280 en direccion I2C 0x76
Serial.println("BMP280 no detectado.");  // Si falla, mostramos mensaje de error
while (1);  // Detenemos el programa si no se encuentra el sensor
}
```

```
 // Configuramos el modo de medicion forzado (bajo consumo, medicion bajo demanda)
bmp.setSampling(Adafruit_BMP280::MODE_NORMAL,
Adafruit_BMP280::SAMPLING_X1,
Adafruit_BMP280::SAMPLING_X1,
Adafruit_BMP280::FILTER_OFF);

 // Cabecera CSV: tiempo, presion absoluta, presion relativa (respecto a la inicial), temperatura
Serial.println("Tiempo(ms);Presion_abs(Pa);Presion_rel(Pa);Temp(C)");
}

void loop() {  // Bucle principal: se repite indefinidamente
 // Variables estaticas: mantienen su valor entre llamadas a loop()
static float p0 = 0;  // Presion de referencia (calibracion inicial)
static bool calibrado = false;  // Indicador de que la calibracion se ha completado

unsigned long t = millis();  // Tiempo actual en milisegundos
float presion = bmp.readPressure();  // Leemos presion absoluta en pascales
float temperatura = bmp.readTemperature();  // Leemos temperatura en grados Celsius

 // FASE 1: Calibracion inicial (primeros 2 segundos)
 // Establecemos la presion ambiental como referencia p0 para medir diferencias
if (!calibrado) {
if (t > 2000) {  // Esperamos 2 segundos a que el sensor se estabilice
p0 = presion;  // Guardamos la presion ambiente como referencia
calibrado = true;  // Marcamos la calibracion como completada
Serial.println("--- CALIBRADO ---");  // Indicamos por serie que la calibracion ha finalizado
}
return;  // Salimos sin enviar datos hasta estar calibrados
}

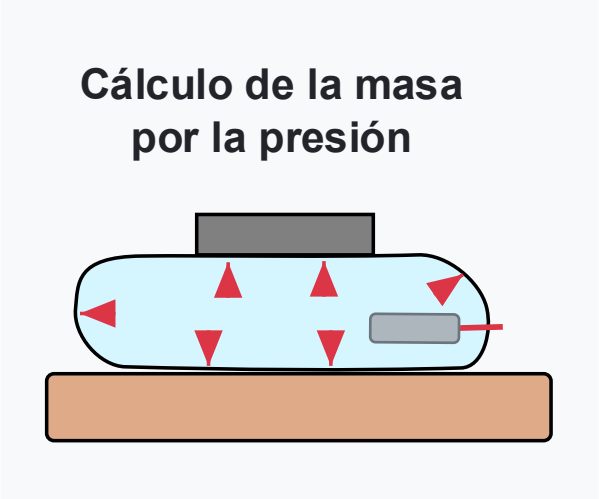
 // FASE 2: Medida continua (despues de la calibracion)
float deltaP = presion - p0;  // Calculamos la presion relativa (diferencia respecto a la inicial)

 // Enviamos los datos por el puerto serie
Serial.print(t);  // Tiempo en ms
Serial.print(";");
Serial.print(presion, 1);  // Presion absoluta con 1 decimal
Serial.print(";");
Serial.print(deltaP, 1);  // Sobrepresion (diferencia) con 1 decimal
Serial.print(";");
Serial.println(temperatura, 2);  // Temperatura con 2 decimales

delay(100);  // Esperamos 100 ms entre muestras (10 lecturas/segundo para capturar la dinamica rapida de la burbuja)
}
```

Medida de masa mediante presión sobre un cojín de aire

Se trata de medir la masa de objetos utilizando la variación de presión en un cojín de aire sellado al relacionar el aumento de presión con la fuerza ejercida por el peso de los objetos sobre el cojín.



Este experimento responde a una idea del profesor Juien de la Universidad Paris Sud – Orsay (Bobroff, 2020) que la llevaba a cabo con teléfonos móviles y que yo repliqué al año siguiente (Ramírez, 2021b).

Fundamento físico

Al colocar una masa (m) sobre un cojín de aire sellado de área (A), la presión interior aumenta según:

$$\Delta P = \frac{F}{A} = \frac{mg}{A}$$

Despejando:

$$m = \frac{\Delta P \cdot A}{g}$$

Materiales y montaje experimental

Tarjeta ESP32 con un sensor BMP280/BME280.
Bolsa de plástico sellable (tipo ziplock) o una bolsa de plástico simple que se cerrará herméticamente.
Objetos de masa conocida (pesas de 50, 100, 200 g), regla o calibre para medir el área de contacto (plaquita rígida), powerbank y cables

BMP280	ESP32
VCC	3,3 ó 5 V
GND	GND
SCL	SCL / GPIO22
SDA	SDA / GPIO21

Introduce el sensor BMP280 conectado a la tarjeta ESP32 enchufada a un powerbank dentro de la bolsa sellable. Sella la bolsa con una pequeña cantidad de aire en su interior. Coloca la bolsa sobre una superficie plana.



Colocad una cajita rígida, por ejemplo de cartón o plástico, encima de la bolsa y medid la presión inicial (P_0) sin carga. Se ha de intenta que la superficie de la caja ocupe casi toda la superficie de la bolsa para que los valores que se obtengan directamente del peso se aproximen lo más posible al valor real.



La presión se mide a través de la aplicación Phyphox conectada a la placa ESP32 a través de bluetooth.

Sobre la caja se van colocando masas de valor conocido para las que se registra la presión en cada caso.

Por ejemplo, en el caso de la bolita de la fotografía de arriba la presión antes y después era de 91265 y 91 296 Pa respectivamente, para un área de la caja de 0,14x0,18 = 0,0 252 m^2. Estos valores dan una masa de 78 g, que es más pequeña que la masa real de 117 g.

Si se calcula $(\Delta P = P - P_0)$ para cada masa y se grafica (ΔP) vs (m), a partir de la ecuación de la línea de tendencia lineal se puede determinar la pendiente (m/gA) y compararla con el área medida.

Ahora ya se puede medir la masa desconocida de objetos cotidianos.

Código

```
#include <Wire.h>
#include <Adafruit_BMP280.h>
#include <phyphoxBle.h>  // Librería para la comunicación con Phyphox

Adafruit_BMP280 bmp;  // Creamos el objeto para el sensor de presion BMP280

void setup() {
  Serial.begin(9600);  // Iniciamos comunicacion serie a 9600 baudios
  bmp.begin(0x76);     // Inicializamos el sensor en la direccion I2C 0x76

  Serial.println("Presion(Pa)");  // Cabecera para el monitor serie

  // INICIO DE CONFIGURACIÓN PHYPHOX
  PhyphoxBLE::start("BMP280_Presion"); // Nombre del dispositivo en Bluetooth

  PhyphoxBleExperiment experimento;
  experimento.setTitle("Presión Simple");

  PhyphoxBleExperiment::View vista;
  vista.setLabel("Mediciones");

  // 1. Mostrar el valor numérico
  PhyphoxBleExperiment::Value valPresion;
  valPresion.setLabel("Presión actual");
  valPresion.setUnit("Pa");
  valPresion.setChannel(2); // La presión irá en el canal 2

  // 2. Mostrar una gráfica
  PhyphoxBleExperiment::Graph graphPresion;
  graphPresion.setLabel("Presión vs Tiempo");
  graphPresion.setUnitX("s");
  graphPresion.setUnitY("Pa");
  graphPresion.setChannel(1, 2); // Canal 1 = Tiempo(X), Canal 2 = Presión(Y)

  // Añadimos los elementos y cargamos el experimento
  vista.addElement(valPresion);
  vista.addElement(graphPresion);
  experimento.addView(vista);
  PhyphoxBLE::addExperiment(experimento);
}

void loop() {
  float p = bmp.readPressure();  // Leemos la presion atmosferica actual en pascales (Pa)
  float t = millis() / 1000.0;   // Calculamos el tiempo en segundos para la gráfica

  Serial.println(p, 1);  // Enviamos la presion con 1 decimal seguida de un salto de linea

  // Enviamos los datos por Bluetooth a Phyphox
  // Canal 1: tiempo (t) | Canal 2: presión (p)
  PhyphoxBLE::write(t, p);

  delay(500);  // Esperamos 500 ms (1/2 segundo) entre mediciones
}
```

Calor y Temperatura

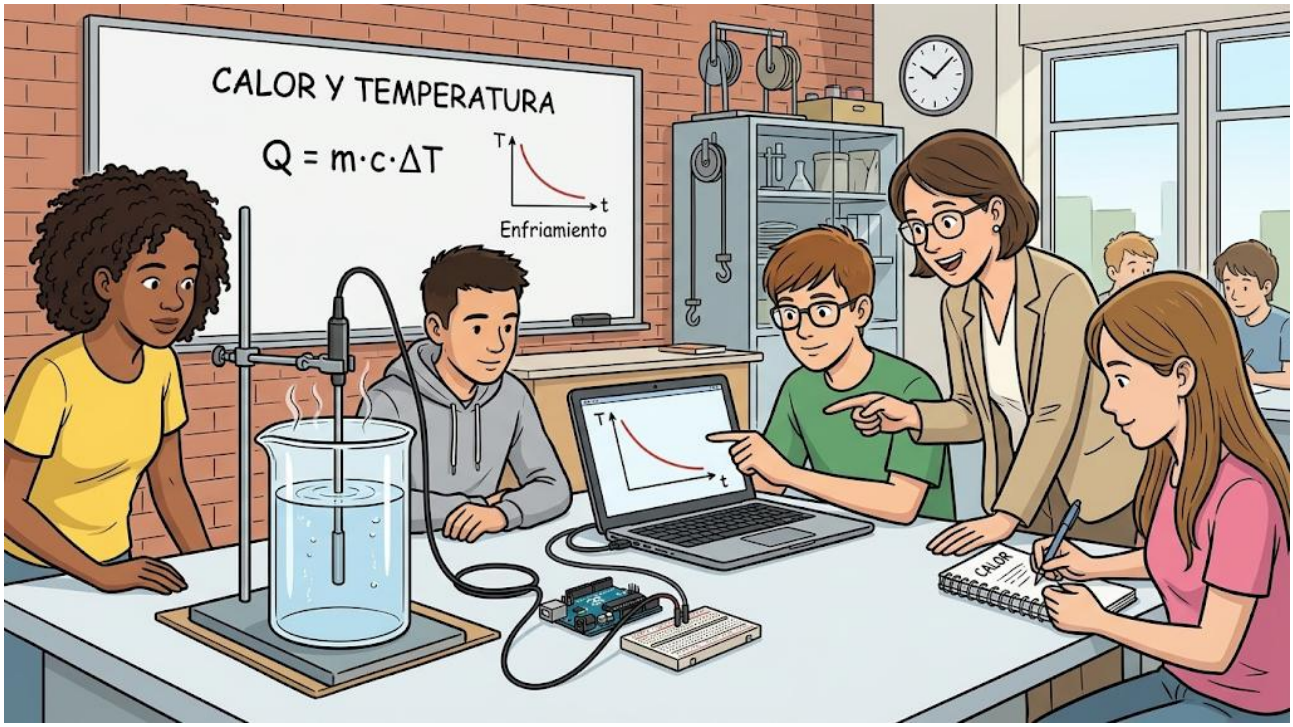
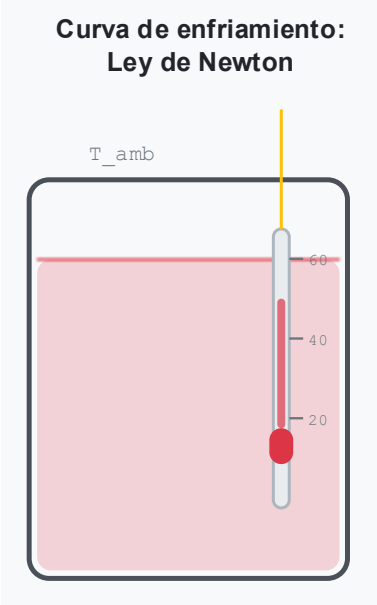


Ilustración realizada con Gemini. Esté pensada para adornar, aunque el alumnado puede entretenerse en buscar los errores evidentes en las conexiones que contiene.

Medida de la temperatura: curva de enfriamiento con sensor DS18B20

Vamos a registrar la temperatura de un líquido a lo largo del tiempo utilizando el estupendo sensor digital DS18B20, que puede sumergirse en líquidos, y analizar la curva de enfriamiento para determinar la constante de tiempo térmica del sistema según la ley de enfriamiento de Newton.



Este experimento tradicionalmente se realizaba con un cronómetro (reloj), un termómetro de alcohol y un cuaderno de laboratorio donde se iban apuntando los valores (Martínez i otras, 2012).

Ahora se utilizan sondas de temperatura (Molas y otras, 2009), pero F. Bouquet y otros (Bouquet y otros, 2019) ya han propuesto realizar esta actividad utilizando sensores de temperatura y Arduino en un artículo sobre experimentos de bajo coste con objetos cotidianos en el que también incluían otros tres: Oscilaciones mecánicas de una goma de pollo con un sensor acelerómetro, transmisión de la luz con el sensor correspondiente y deformación de una regla aprovechando un sensor de efecto Hall.

Fundamento físico

La ley de enfriamiento de Newton establece que la velocidad de pérdida de calor de un cuerpo es proporcional a la diferencia de temperatura entre el cuerpo y el entorno:

$$\frac{dT}{dt} = -k\,(T - T_{\text{amb}})$$

donde T es la temperatura del cuerpo, T_{amb} la temperatura ambiente y k la constante de enfriamiento (inversa de la constante de tiempo térmica $\tau = 1/k$). Integrando esta ecuación diferencial obtenemos:

$$T(t) = T_{\text{amb}} + (T_0 - T_{\text{amb}})\,e^{-kt}$$

donde T_0 es la temperatura inicial. Esta función exponencial decreciente describe cómo se enfría un cuerpo cuando está más caliente que su entorno. Ajustando los datos experimentaos a esta ecuación podemos determinar la constante de tiempo térmica τ, que depende de la masa del líquido, su calor específico y las propiedades aislantes del recipiente.

Materiales y montaje experimental

Necesitaréis una placa Wemos D1 R32 (ESP32), un sensor DS18B20 en cápsula de acero inoxidable (resistente al agua), una resistencia pull-up de 4,7 kΩ, agua caliente (no hirviendo) y un recipiente, también una protoboard y cables Dupont, y opcionalmente un powerbank para alimentación autónoma.

Conectad el sensor DS18B20 de manera que el cable rojo del sensor (VCC) se conecta a 5 V de la placa, el cable amarillo (datos) al pin GPIO26 junto con una resistencia pull-up de 4,7 kΩ a VCC, y el cable negro (GND) a GND.

Una vez conectado, se sube el código y se abre el monitor serie para verificar que el sensor lee correctamente la temperatura ambiente.

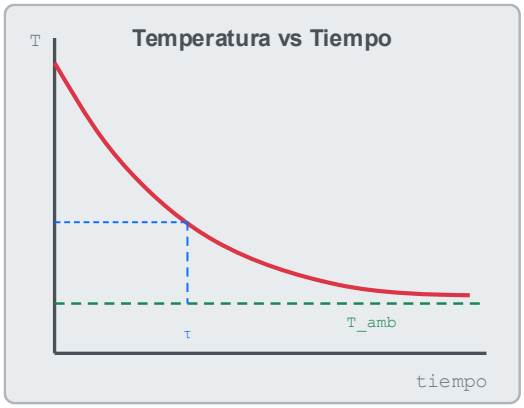
El sensor DS18B20 en cápsula de acero inoxidable es resistente al agua, por lo que puede introducirse directamente en el líquido.

Calentad agua en un recipiente hasta unos 60-70 °C (sin llegar a ebullición) y sumergid la sonda del DS18B20 en el agua. A continuación, dejad el sistema enfriarse de forma natural mientras el microprocesador registra la temperatura cada segundo. Los datos se enviarán por Bluetooth al móvil y se leerán con Phyphox. Continuad la toma de datos hasta que la temperatura se estabilice cerca de la temperatura ambiente.

Resultados esperados

La gráfica de temperatura frente al tiempo mostrará una curva exponencial decreciente que comienza en la temperatura inicial del agua caliente y se aproxima asintóticamente a la temperatura ambiente. Por ejemplo, si la temperatura inicial es de 60 °C, la temperatura ambiente de 22 °C y la constante de tiempo $\tau = 1/k \approx 300$ s, la temperatura tras 5 minutos será aproximadamente:

$$T(300) = 22 + (60 - 22) \cdot e^{-300/300}$$
$$= 22 + 38 \cdot 0{,}368 \approx 36\,^{\circ}C$$



Para determinar la constante de tiempo térmica τ, representad el logaritmo natural de la diferencia de temperatura frente al tiempo:

$$\ln(T - T_{\text{amb}}) = \ln(T_0 - T_{\text{amb}}) - kt$$

Si el enfriamiento sigue la ley de Newton, los puntos deberían alinearse en una recta cuya pendiente es $-k = -1/\tau$. Calculad τ a partir de la pendiente y discutid su significado físico y su dependencia de la masa de agua, el calor específico, el área superficial del recipiente y el coeficiente de transferencia térmica.

Ampliación

Repetid el experimento con diferentes volúmenes de agua para comprobar que la constante de tiempo τ aumenta con la masa. Probad también con diferentes recipientes (metal, plástico, vidrio) para estudiar la influencia del material en la velocidad de enfriamiento.

También se puede aislar el recipiente con un envoltorio térmico y comparar la constante de tiempo obtenida.

Otro experimento puede consistir en medir la capacidad calorífica, por ejemplo, de tuercas (midiendo la disminución de la temperatura del agua caliente cuando las tuercas hexagonales se sumergen en ella).

Además se puede medir la potencia de un calentador eléctrico sumergible (siguiendo el aumento de la temperatura del agua con el tiempo).

Código

Se ha de instalar la librería DallasTemperature.

```
/*
Medida de temperatura con sonda DS18B20 y
Bluetooth para Phyphox
Placa: Wemos D1 R32 (ESP32)

Conexiones: Rojo a 5V, Amarillo a GPIO26 con
pull-up 4,7k, Negro a GND

Los datos se visualizan por el monitor serie del
ordenador
y por Bluetooth en la app Phyphox.
*/

#include <OneWire.h>
#include <DallasTemperature.h>
#include <phyphoxBle.h> // Sustituimos
BluetoothSerial por la librería de Phyphox

// Configurar bus OneWire en pin GPIO26
OneWire ourWire(26);
DallasTemperature sensors(&ourWire);

void setup() {
  Serial.begin(115200);         // Monitor serie
  sensors.begin();              // Inicializar sensor
DS18B20

  Serial.println("Temperatura inicializada");

  // CONFIGURACIÓN DE PHYPHOX
  PhyphoxBLE::start("ESP32_Temp"); // Nombre
que aparecerá al buscar desde la app

  // Creamos el experimento para Phyphox
  PhyphoxBleExperiment experimento;
```

```
  experimento.setTitle("Sensor DS18B20");
  experimento.setCategory("Mis Sensores");

  // Creamos la vista (pestaña)
  PhyphoxBleExperiment::View vista;
  vista.setLabel("Datos térmicos");

  // 1. Mostrar el valor numérico absoluto
  PhyphoxBleExperiment::Value valTemp;
  valTemp.setLabel("Temperatura Actual");
  valTemp.setUnit("°C");
  valTemp.setChannel(2); // La temperatura viajará por el canal 2

  // 2. Mostrar la gráfica de Temperatura frente a Tiempo
  PhyphoxBleExperiment::Graph graphTemp;
  graphTemp.setLabel("Evolución de la temperatura");
  graphTemp.setUnitX("s");
  graphTemp.setUnitY("°C");
  graphTemp.setLabelX("Tiempo");
  graphTemp.setLabelY("Temperatura");
  graphTemp.setChannel(1, 2); // Eje X: Canal 1 (Tiempo) | Eje Y: Canal 2 (Temperatura)

  // Añadimos los elementos a la vista y al experimento
  vista.addElement(valTemp);
  vista.addElement(graphTemp);
  experimento.addView(vista);
  PhyphoxBLE::addExperiment(experimento);
}

void loop() {
  sensors.requestTemperatures();  // Solicitar lectura al sensor
  float temp = sensors.getTempCByIndex(0);  // Obtener temperatura en °C

  if (temp != -127.0 && temp != 85.0) {  // Descartar lecturas erróneas

    // Usamos float para los segundos para que la gráfica fluya con decimales
    float t = millis() / 1000.0;

    // Mostrar por monitor serie
    Serial.print(t, 2);
    Serial.print("; ");
    Serial.println(temp, 2);

    // Enviar datos por Bluetooth a Phyphox
    // Canal 1: tiempo (t) | Canal 2: temperatura (temp)
    PhyphoxBLE::write(t, temp);
  }

  delay(1000);  // Leer cada segundo (Phyphox actualizará la gráfica cada segundo)
}
```

Calor latente de fusión del hielo

Determinaremos experimentalmente el calor latente de fusión del hielo (L_f) midiendo la variación de temperatura al añadir hielo a una masa conocida de agua caliente, aplicando el principio de conservación de la energía calorífica.



Este es un experimento clásico que se ha hecho tanto utilizando un termómetro analógico (Laboratorio de Física, 2014), como digital (Departamento de Física Aplicada, 2006a), y aquí lo haremos con un sensor de temperatura conectado a arduinos.

Fundamento físico

Cuando el hielo funde, absorbe una cantidad de energía llamada calor latente de fusión sin que su temperatura varíe (se mantiene a 0 °C durante todo el cambio de fase). La energía absorbida es

$$Q = m_{hielo} \cdot L_f,$$

donde L_f = 334 kJ/kg para el agua.

Al añadir hielo a agua caliente, el calor cedido por el agua al enfriarse $\left(Q_{cedido} = m_{agua} \cdot c_e \cdot \Delta T\right)$ se emplea en fundir el hielo $(m_{hielo} \cdot L_f)$ y en calentar el agua procedente del hielo desde 0 °C hasta la temperatura de equilibrio $(m_{hielo} \cdot c_e \cdot \Delta T_{eq})$. El balance energético permite despejar L_f experimentalmente.

Ce es el calor específico del agua líquida que vale 4180 J/(kg · K). Si en los cálculos se considera el calentamiento del hielo desde la temperatura del congelador a los 0º, se ha de utilizar el calor específico del hielo, que es aproximadamente 2090 J/(kg · K).

Materiales y montaje experimental

Necesitaréis una placa Wemos D1 R32 (ESP32) o Arduino UNO, un sensor de temperatura DS18B20 con funda metálica impermeable, una resistencia pull-up de 4,7 kΩ, una protoboard, cables Dupont, un vaso de precipitados (o recipiente similar), un termómetro de referencia para calibración (opcional), hielo (preferiblemente triturado), agua y una balanza digital.

Conectad el sensor DS18B20 a la placa: el cable rojo (VCC) a 5 V, el negro (GND) a GND y el amarillo (datos) a un pin digital (D2 en Arduino UNO, GPIO26 en ESP32) con una resistencia pull-up de 4,7 kΩ entre VCC y datos.

Introducir el sensor en el agua asegurando que la funda metálica quede completamente sumergida pero sin tocar las paredes ni el fondo del recipiente.

Calentad agua hasta unos 40-50 °C y verted una masa conocida (por ejemplo, 200 g) en el vaso de precipitados. Registrad la temperatura inicial estable del agua. Pesad una cantidad conocida de hielo (por ejemplo, 30 g) y añadidlo rápidamente al agua. Removed suavemente y registrad la temperatura cada segundo hasta que se estabilice (el hielo se habrá fundido completamente). Anotad la temperatura final de equilibrio.

Resultados esperados

La temperatura del agua descenderá rápidamente al añadir el hielo y se estabilizará cuando todo el hielo se haya fundido. Con 200 g de agua a 45 °C y 30 g de hielo, la temperatura de equilibrio esperada es de aproximadamente 28 °C.

Aplicad la ecuación de balance energético:

$$m_{agua} \cdot c_e \cdot (T_{inicial} - T_{eq}) = m_{hielo} \cdot L_f + m_{hielo} \cdot c_e \cdot (T_{eq} - 0)$$

Despejad L_f y comparad con el valor teórico de 334 kJ/kg.

Calculad el error relativo y discutid las fuentes de error, como pueden ser no considerar el calentamiento del hielo antes de empezar afundirse, las pérdidas de calor al ambiente, el calor específico del recipiente, la precisión de las masas y temperaturas, etc.

Ampliación

Repetid el experimento con diferentes masas de hielo y agua para evaluar la consistencia del resultado.

Código

```
/*
Medida de temperatura con sonda DS18B20 y
Bluetooth para Phyphox
Placa: Wemos D1 R32 (ESP32) — Seleccionar:
DOIT ESP32 DEVKIT V1

Conexiones: Rojo a 5V, Amarillo a GPIO26 con
pull-up 4,7k, Negro a GND

Los datos se visualizan por el monitor serie del
ordenador
y por Bluetooth en la app Phyphox.
*/

#include <OneWire.h>
#include <DallasTemperature.h>
#include <phyphoxBle.h> // Sustituimos
BluetoothSerial por la librería de Phyphox

// Configurar bus OneWire en pin GPIO26
OneWire ourWire(26);
DallasTemperature sensors(&ourWire);

void setup() {
  Serial.begin(115200);         // Monitor serie
  sensors.begin();              // Inicializar sensor
DS18B20

  Serial.println("Temperatura inicializada");

  // CONFIGURACIÓN DE PHYPHOX
  PhyphoxBLE::start("ESP32_Temp"); // Nombre
que aparecerá al buscar desde la app

  // Creamos el experimento para Phyphox
  PhyphoxBleExperiment experimento;
  experimento.setTitle("Sensor DS18B20");
  experimento.setCategory("Mis Sensores");

  // Creamos la vista (pestaña)
  PhyphoxBleExperiment::View vista;
  vista.setLabel("Datos térmicos");

  // 1. Mostrar el valor numérico absoluto
  PhyphoxBleExperiment::Value valTemp;
  valTemp.setLabel("Temperatura Actual");
  valTemp.setUnit("°C");
  valTemp.setChannel(2); // La temperatura viajará
por el canal 2

  // 2. Mostrar la gráfica de Temperatura frente a
Tiempo
  PhyphoxBleExperiment::Graph graphTemp;
  graphTemp.setLabel("Evolución de la
temperatura");
  graphTemp.setUnitX("s");
  graphTemp.setUnitY("°C");
  graphTemp.setLabelX("Tiempo");
  graphTemp.setLabelY("Temperatura");
  graphTemp.setChannel(1, 2); // Eje X: Canal 1
(Tiempo) | Eje Y: Canal 2 (Temperatura)

  // Añadimos los elementos a la vista y al
experimento
  vista.addElement(valTemp);
  vista.addElement(graphTemp);
  experimento.addView(vista);
  PhyphoxBLE::addExperiment(experimento);
}

void loop() {
  sensors.requestTemperatures();  // Solicitar
lectura al sensor
  float temp = sensors.getTempCByIndex(0);  //
Obtener temperatura en °C

  if (temp != -127.0 && temp != 85.0) {  // Descartar
lecturas erróneas

    // Usamos float para los segundos para que la
gráfica fluya con decimales
    float t = millis() / 1000.0;

    // Mostrar por monitor serie
```

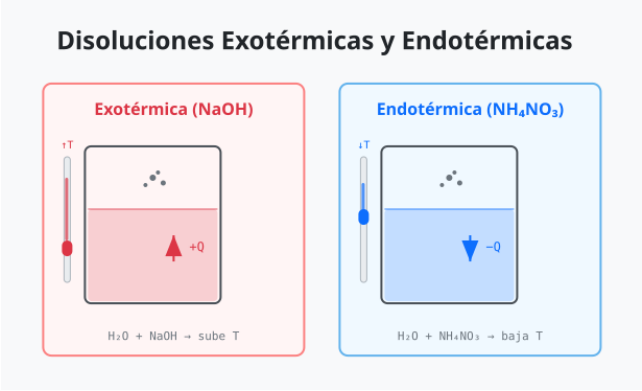
```
    Serial.print(t, 2);
    Serial.print("; ");
    Serial.println(temp, 2);

    // Enviar datos por Bluetooth a Phyphox
    // Canal 1: tiempo (t) | Canal 2: temperatura
(temp)
    PhyphoxBLE::write(t, temp);
  }

  delay(1000);  // Leer cada segundo (Phyphox
actualizará la gráfica cada segundo)
}
```

Disoluciones exotérmicas y endotérmicas

Ahora vamos a medir la variación de temperatura al disolver diferentes sales en agua (NaOH: exotérmica, NH_4NO_3: endotérmica) y calcular el calor de disolución.



Esto, y además el cálculo de entalpia de reacciones, es lo que ya proponían hace años Aparicio, Corominas y Lozano (2003a, b, c y d).

Fundamento físico

El proceso de disolución de una sal en agua implica una serie de cambios energéticos asociados a las interacciones entre el soluto y el disolvente. La entalpía de disolución (ΔH_{dis}) es la energía absorbida o liberada a presión constante cuando una cantidad determinada de soluto se disuelve completamente en un disolvente.

Físicamente, este proceso puede dividirse en dos etapas conceptuales fundamentales:
1. Ruptura de la red cristalina del soluto: Requiere energía para vencer las fuerzas electrostáticas que mantienen unidos a los iones en el cristal sólido (proceso endotérmico). Esta energía se relaciona directamente con la energía reticular (U).

2. Hidratación de los iones: Los iones liberados interaccionan con los dipolos de las moléculas de agua circundantes, formando una capa de solvatación. Este proceso de atracción ión-dipolo libera energía (proceso exotérmico), conocida como energía de hidratación (ΔH_{hidr}).

La variación total de entalpía (ΔH_{dis}) viene dada por el balance de ambas contribuciones:

$$\Delta H_{dis} = U + \Delta H_{hidr}$$

Si la energía de hidratación liberada es mayor que la energía reticular (magnitud), el proceso neto es exotérmico ($\Delta H_{dis} < 0$), elevando la temperatura de la disolución, como ocurre con el hidróxido de sodio (NaOH). El valor teórico para el NaOH es de aproximadamente -44.5 kJ/mol.

Si la energía requerida para romper el cristal es mayor que la energía liberada en la hidratación, el proceso neto es endotérmico ($\Delta H_{dis} > 0$), disminuyendo la temperatura de la disolución, como es el caso del nitrato de amonio (NH_4NO_3). Su entalpía teórica de disolución es de unos +25.7 kJ/mol.

El calor intercambiado (Q) a presión constante se puede calcular experimentalmente utilizando la calorimetría, midiendo el cambio de temperatura del agua:

$$Q = m \cdot c_e \cdot \Delta T$$

donde m es la masa total de la disolución (soluto + disolvente), c_e es el calor específico de la disolución (que podemos aproximar al del agua, 4.18 $J/(g \cdot ^\circ C)$ para disoluciones diluidas) y ΔT es la variación de temperatura $(T_{final} - T_{inicial})$. A partir del calor, la entalpía molar experimental es $\Delta H = -Q/n$, donde n son los moles de soluto (el signo menos indica que si el agua absorbe calor, Q>0, la disolución es exotérmica, $\Delta H < 0$).

Materiales y montaje experimental

Arduino UNO o ESP32 y sensor DS18B20 (impermeable y sumergible en agua).
Agua destilada, balanza y vaso de precipitados (100 mL).
Hidróxido de sodio (NaOH, sosa cáustica), con solubilidad en agua muy alta, de unos 111 g por cada 100 mL de agua a 20 °C. **Precaución**: Sustancia altamente corrosiva; usar guantes y gafas protectoras.
Nitrato de amonio (NH_4NO_3), muy soluble en agua, alcanzando unos 150 g a 190 g por cada 100 ml a 20 °C y subiendo muy rápidamente al aumentar la temperatura del agua.

Conectad el DS18B20: VCC (rojo) → 3,3 V, GND (negro) → GND, DATA (amarillo) → D2 con resistencia 4,7 kΩ a VCC.

Añadir una masa conocida (ej. 50 g) de agua destilada en el vaso de precipitados e iniciar la toma de datos con el microcontrolador y medir la temperatura inicial del agua antes de añadir el soluto.
Pesar con precisión una masa conocida del compuesto químico (ej. 2 g a 5 g de NaOH o NH_4NO_3).
Añadir la sal al agua y agitar continuamente mientras se está registrando la temperatura cada segundo.
Continuar la medición durante un par de minutos, hasta identificar claramente el máximo o mínimo térmico (equilibrio térmico antes de que empiece a perder/ganar calor con el ambiente).

Para tomar la temperatura con el sensor de forma segura, se pudiera pensar en introducirlo protegido con una funda fina de plástico, aunque el DS18B20 impermeable de acero inoxidable suele resistir bien inmersiones cortas en bases.

Análisis y resultados

Una vez registrada la curva de temperatura vs. tiempo calculad el $\Delta T_{max} = T_{final} - T_{inicial}$ y después el calor intercambiado

$$Q = m_{total} \cdot c_e \cdot \Delta T_{max}.$$

Determinad también el calor molar de disolución experimental:

$$\Delta H_{exp} = -Q/n_{soluto}$$

Comparad el valor obtenido de ΔH_{exp} con los valores teóricos (~ -44.5 kJ/mol para NaOH y $\sim +25.7$ kJ/mol para NH_4NO_3) i calculad el error relativo.

Los valores experimentales suelen ser menores en magnitud debido a las pérdidas de calor hacia el entorno (vaso y aire) al no usar un calorímetro ideal (aislado). Se recomienda debatir estas fuentes de error en el análisis, así como la capacidad calorífica del propio sensor y del recipiente.

Ampliación

Probad la determinación de entalpías de otras sustancias, por ejemplo con el carbonato de sodio anhidro (Na_2CO_3) frente al decahidratado ($Na_2CO_3 \cdot 10H_2O$), para observar el efecto de la red ya hidratada frente a la anhidra.

Determinación de la entalpía estándar de neutralización ácido-base, como la reacción del ácido clorhídrico (HCl aq) y el hidróxido de sodio (NaOH), o la entalpía de reacciones redox espontáneas como el sulfato de cobre pentahidratado ($CuSO_4 \cdot 5H_2O$) y el cinc (Zn) metálico.

Código

```
/*
Medida de temperatura con sonda DS18B20 y Bluetooth para Phyphox
Placa: Wemos D1 R32 (ESP32)

Conexiones: Rojo a 5V, Amarillo a GPIO26 con pull-up 4,7k, Negro a GND

Los datos se visualizan por el monitor serie del ordenador y por Bluetooth en la app Phyphox.
*/

#include <OneWire.h>
#include <DallasTemperature.h> // librería del termómetro
#include <phyphoxBle.h> // librería de Phyphox

// Configurar bus OneWire en pin GPIO26
OneWire ourWire(26);
DallasTemperature sensors(&ourWire);

void setup() {
  Serial.begin(115200);  // Monitor serie
  sensors.begin();  // Inicializar sensor DS18B20

  Serial.println("Temperatura inicializada");

  // CONFIGURACIÓN DE PHYPHOX
  PhyphoxBLE::start("ESP32_Temp"); // Nombre que aparecerá al buscar desde la app

  // Creamos el experimento para Phyphox
  PhyphoxBleExperiment experimento;
  experimento.setTitle("Sensor DS18B20");
  experimento.setCategory("Mis Sensores");

  // Creamos la vista (pestaña)
  PhyphoxBleExperiment::View vista;
  vista.setLabel("Datos térmicos");
```

```
  // 1. Mostrar el valor numérico absoluto
  PhyphoxBleExperiment::Value valTemp;
  valTemp.setLabel("Temperatura Actual");
  valTemp.setUnit("°C");
  valTemp.setChannel(2); // La temperatura viajará por el canal 2

  // 2. Mostrar la gráfica de Temperatura frente a Tiempo
  PhyphoxBleExperiment::Graph graphTemp;
  graphTemp.setLabel("Evolución de la temperatura");
  graphTemp.setUnitX("s");
  graphTemp.setUnitY("°C");
  graphTemp.setLabelX("Tiempo");
  graphTemp.setLabelY("Temperatura");
  graphTemp.setChannel(1, 2); // Eje X: Canal 1 (Tiempo) | Eje Y: Canal 2 (Temperatura)

  // Añadimos los elementos a la vista y al experimento
  vista.addElement(valTemp);
  vista.addElement(graphTemp);
  experimento.addView(vista);
  PhyphoxBLE::addExperiment(experimento);
}

void loop() {
  sensors.requestTemperatures();  // Solicitar lectura al sensor
  float temp = sensors.getTempCByIndex(0);  // Obtener temperatura en °C

  if (temp != -127.0 && temp != 85.0) {  // Descartar lecturas erróneas

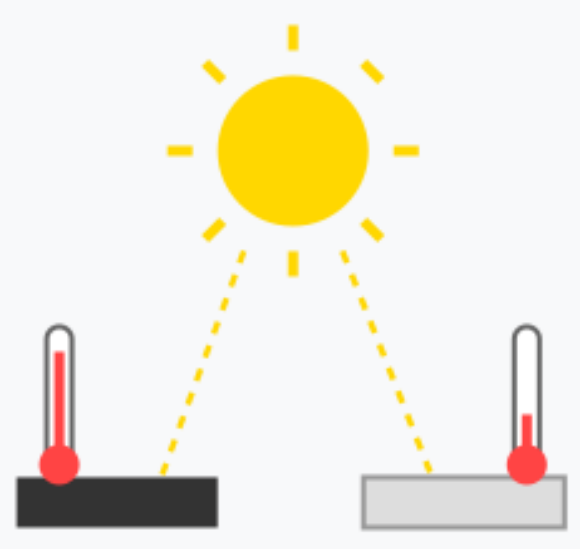
    // Usamos float para los segundos para que la gráfica fluya con decimales
    float t = millis() / 1000.0;

    // Mostrar por monitor serie
    Serial.print(t, 2);
    Serial.print("; ");
    Serial.println(temp, 2);

    // Enviar datos por Bluetooth a Phyphox
    // Canal 1: tiempo (t) | Canal 2: temperatura (temp)
    PhyphoxBLE::write(t, temp);
  }
  delay(1000);  // Leer cada segundo (Phyphox actualizará la gráfica cada segundo)
}
```

Absorción de la radiación solar o de una bombilla por superficies de diferente color

Estudiar cómo el color de una superficie influye en la absorción de la radiación térmica (luz visible e infrarroja) midiendo la temperatura alcanzada por superficies pintadas de negro, blanco y otros colores cuando se exponen a una fuente de luz (bombilla incandescente o luz solar), utilizando termistores NTC conectados a un Arduino.



Elórtegui (2024) lo hace utilizando termómetros clásicos y lo explica en la entrada “Emisión y absorción de energía por radiación” de su web *Ciencia de cada día*.

Fundamento físico

La transferencia de calor por radiación es el proceso por el cual la energía se transmite en forma de ondas electromagnéticas. Todos los cuerpos emiten radiación térmica según la ley de Stefan-Boltzmann:

$$P = \varepsilon \cdot \sigma \cdot A \cdot T^4,$$

donde P es la potencia radiada, ε la emisividad ($0 \leq \varepsilon \leq 1$), σ la constante de Stefan-Boltzmann ($5{,}67\times10^{-8}$ $W\cdot m^{-2}\cdot K^{-4}$), A el área y T la temperatura absoluta.

La absorción de la radiación incidente depende del color de la superficie. Las superficies oscuras (como el negro) absorben la mayor parte de la radiación en todo el espectro visible e infrarrojo, mientras que las superficies claras (como el blanco) reflejan la mayor parte.

La emisividad y la absortividad están relacionadas por la ley de Kirchhoff de la radiación térmica en la que para una misma longitud de onda, la emisividad es igual a la absortividad ($\varepsilon(\lambda) = \alpha(\lambda)$). Por tanto, una superficie negra, que absorbe más radiación, también emite más radiación cuando se calienta, alcanzando un equilibrio térmico a mayor temperatura que una superficie blanca expuesta a la misma fuente.

Para medir la temperatura se utilizan termistores NTC (Negative Temperature Coefficient), cuya resistencia disminuye al aumentar la temperatura. La relación resistencia-temperatura sigue aproximadamente la ecuación de Steinhart-Hart.

En este experimento no es necesario calibrar absolutamente los termistores, ya que el interés está en la comparación relativa entre superficies.

Materiales y montaje experimental

Se necesita una placa Arduino UNO (o cualquier placa compatible), dos termistores NTC de 500 Ω (o de valor similar), dos resistencias fijas de 4,7 kΩ, una protoboard, cables Dupont.
Además, dos pequeñas superficies metálicas (como moldes metálicos de cocina o chapas), pasta térmica, pintura acrílica negra mate y blanca (y opcionalmente otros colores), un soporte de poliestireno expandido (corcho blanco) o madera para aislar, una lámpara incandescente de 150 W (o foco LED potente), un soporte para la lámpara y un cronómetro.

Cada termistor NTC se ha de conectar en serie con una resistencia fija de 4,7 kΩ entre 5V y GND, formando un divisor de tensión. El punto medio de cada divisor (entre el NTC y la resistencia) se conecta a una entrada analógica del Arduino (A0 y A1 respectivamente).

Para preparar las superficies se ha de pintar una de las caras de cada placa metálica de un color distinto

(negro mate, blanco, y opcionalmente amarillo, rojo, etc.). Dejad secar bien. En la cara opuesta (sin pintar), aplicad una pequeña cantidad de pasta térmica y fijad el termistor NTC con cinta adhesiva o pegamento térmico, asegurando un buen contacto térmico.

Montad las superficies pintadas en el soporte de poliestireno expandido, enfrentando las caras pintadas hacia la lámpara. La lámpara ha de estar a la misma distancia de todas las superficies (unos 15–20 cm). Aislad la parte trasera de los sensores con más poliestireno para minimizar las pérdidas por convección y conducción.

Una vez todo montado, conectad la placa Arduino al ordenador y cargad el código. Iniciad la toma de datos con el monitor serie (o con RealTerm para capturar los datos en un archivo) y durante los primeros 30 segundos, registrad la temperatura de referencia sin la lámpara encendida.

A continuación, encended la lámpara y continuad registrando durante 3–5 minutos hasta que las temperaturas se estabilicen (se alcance el equilibrio dinámico). Apagad la lámpara y seguid registrando durante otros 2–3 minutos mientras las superficies se enfrían hasta la temperatura ambiente.

Repetid el experimento con diferentes pares de colores (blanco vs. negro, blanco vs. amarillo, etc.) para comparar la capacidad de absorción de cada color.

Resultados esperados

Al encender la lámpara, la superficie negra se calentará más rápidamente y alcanzará una temperatura de equilibrio mayor que la superficie blanca (típicamente 5–15 °C más, dependiendo de la potencia de la lámpara y la distancia).

Al apagar la lámpara, ambas superficies se enfriarán tendiendo a la temperatura ambiente, pero la superficie negra, al haber alcanzado mayor temperatura, tardará más en enfriarse y mostrará una tasa de enfriamiento inicial mayor (ley de Newton del enfriamiento).

Las superficies de colores intermedios (amarillo, rojo, verde) mostrarán un comportamiento intermedio entre el blanco y el negro.

Representad en una misma gráfica la temperatura (o el valor crudo del ADC) frente al tiempo para ambas superficies, indicando los instantes de encendido y apagado de la lámpara. Calculad la diferencia de temperatura de equilibrio entre las superficies.

Ajustad la curva de calentamiento a una exponencial

$$T(t) = T_eq - (T_eq - T^0) \cdot e^{(-t/\tau)}$$

para obtener la constante de tiempo térmica τ de cada superficie. Ajustad también la curva de enfriamiento.

Discutid cómo la diferencia de temperatura se relaciona con la diferencia de absortividad/emisividad entre los colores.

Ampliación

Repetid el experimento con luz solar directa en lugar de la lámpara. Usad un sensor de intensidad luminosa (BH1750) junto con los termistores para correlacionar la irradiancia con la temperatura alcanzada.

También se puede medir la emisividad de las superficies comparando la tasa de enfriamiento en ausencia de fuente de calor externa (la emisividad determina la capacidad de irradiar calor).

Para experimentos más avanzados, calibrad los termistores NTC usando un termómetro de referencia y la ecuación completa de Steinhart-Hart con los tres coeficientes.

Código

```
/*
Medida de absorción de radiación térmica en
superficies de diferente color
Utiliza dos termistores NTC como sensores de
temperatura, montados en divisores de tensión con
resistencias de 4,7 kΩ.

Conexiones:
NTC_negro (A0): resistencia fija 4,7k entre A0 y 5V,
NTC entre A0 y GND
NTC_blanco (A1): resistencia fija 4,7k entre A1 y
5V, NTC entre A1 y GND

Los valores leídos son proporcionales a la
```

```
temperatura, en unidades arbitrarias (no
calibradas).
Para calibración absoluta, usar la ecuación de
Steinhart-Hart con los parámetros del fabricante.
*/

const int pinNegro = A0;
const int pinBlanco = A1;
const float Rfija = 4700.0; // Resistencia fija del
divisor (Ω)
const float R_NTC_25 = 500.0; // Resistencia del
NTC a 25°C (Ω)
const float B = 3950.0; // Coeficiente beta del NTC
(típico)
const float T0 = 298.15; // 25°C en Kelvin

void setup() {
  Serial.begin(9600);
  Serial.println("tiempo(ms); negro(u.a.);
blanco(u.a.); negro(°C); blanco(°C)");
}

float leerTemperatura(int pin) {
  int valorADC = analogRead(pin);
  if (valorADC == 0) valorADC = 1; // Evitar división
por cero

  // Calcular la resistencia del NTC a partir del
divisor de tensión
  float Vout = (valorADC / 1023.0) * 5.0;
  float R_NTC = Rfija / (5.0 / Vout - 1.0);

  // Ecuación de Steinhart-Hart simplificada (modelo
beta)
  float steinhart = log(R_NTC / R_NTC_25) / B + 1.0
/ T0;
  float temperaturaK = 1.0 / steinhart;
  return temperaturaK - 273.15; // Convertir a °C
}

void loop() {
  unsigned long t = millis();

  float T_negra = leerTemperatura(pinNegro);
  float T_blanca = leerTemperatura(pinBlanco);

  // Valor crudo del ADC (unidades arbitrarias,
proporcional a temperatura)
  int rawNegro = analogRead(pinNegro);
  int rawBlanco = analogRead(pinBlanco);

  Serial.print(t);
  Serial.print("; ");
  Serial.print(rawNegro);
  Serial.print("; ");
  Serial.print(rawBlanco);
  Serial.print("; ");
  Serial.print(T_negra, 1);
  Serial.print("; ");
  Serial.println(T_blanca, 1);

  delay(500); // 2 muestras por segundo
}
```

Combinación de las leyes de los gases: Boyle, Charles y Gay-Lussac

Un clásico de la físico-química es el verificar experimentalmente las tres leyes fundamentaos que rigen el comportamiento de los gases ideales: la **ley de Boyle-Mariotte** (relación presión-volumen a temperatura constante), la **ley de Charles** (relación volumen-temperatura a presión constante) y la **segunda ley de Gay-Lussac** (relación presión-temperatura a volumen constante). Estas tres leyes se combinan en la ecuación de estado del gas ideal: (PV = nRT), que constituye uno de los pilares de la termodinámica clásica.

Fundamento físico

La ecuación de estado del gas ideal establece que para una cantidad fija de gas (número de moles (n) constante):

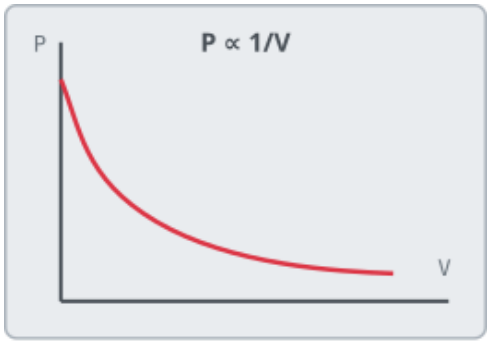
$$PV = nRT$$

donde (P) es la presión absoluta (en Pa), (V) el volumen (en m³), (T) la temperatura absoluta (en K, donde $(T[\mathrm{K}] = T[^\circ\mathrm{C}] + 273{,}15)$ y (R = 8,314 J/(mol·K) es la constante universal de los gases.

De esta ecuación se derivan tres relaciones particulares cuando se mantiene una de las variables fija:

Ley de Boyle-Mariotte (T = cte)
Robert Boyle (1662) y Edme Mariotte (1676) establecieron de forma independiente que, si la temperatura de un gas permanece constante, la presión es inversamente proporcional al volumen:

$$P_1V_1 = P_2V_2 \Longrightarrow P \propto \frac{1}{V}$$

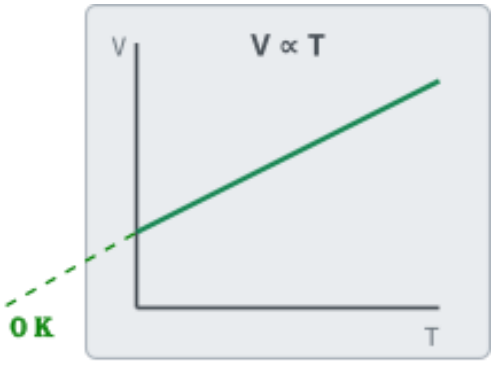


T = constante

Primera ley de Charles (P = cte)
Si la presión permanece constante, el volumen es directamente proporcional a la temperatura absoluta:

$$\frac{V_1}{T_1} = \frac{V_2}{T_2} \Longrightarrow V \propto T$$

Jacques Charles (1787) descubrió esta relación al observar que los gases se expanden al calentarse.

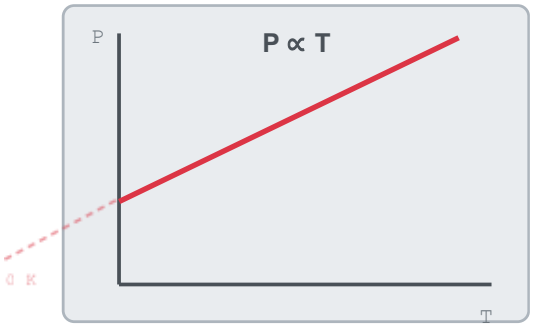


P = constante

Segunda ley de Gay-Lussac (V = cte)
Si el volumen permanece constante, la presión es directamente proporcional a la temperatura absoluta:

$$\frac{P_1}{T_1} = \frac{P_2}{T_2} \Longrightarrow P \propto T$$

Joseph Louis Gay-Lussac (1809) estableció esta ley calentando gases en recipientes rígidos. Es responsable del aumento de presión en un aerosol expuesto al calor y el fundamento del termómetro de gas a volumen constante.



V = constante

Aplicación combinada
Las tres leyes pueden unificarse en la expresión:

$$\frac{P_1V_1}{T_1} = \frac{P_2V_2}{T_2}$$

que permite predecir el estado final de un gas cuando cambian simultáneamente presión, volumen y temperatura.

Materiales y montajes experimentales

Para los tres montajes experimentales se necesita:

- Una placa ESP32 Wemos D1 R32 (o un Arduino Nano 33 BLE Sense)
- Un sensor BME280 (presión, temperatura y humedad) o BMP280 (presión y temperatura). El BME280 ofrece además humedad relativa.
- Conectad el sensor BME280 a cuatro cables Dupont finos (VCC, GND, SCL, SDA).
- Protoboard pequeña
- Powerbank o batería externa USB para alimentación autónoma
- Phyphox instalado en un smartphone (Android/iOS)
- Cinta aislante o masilla de sellado (tipo Blu-Tack o silicona caliente)
- Termómetro digital de respaldo

Conexiones eléctricas para los tres montajes (BME280 → ESP32):

Pin BME280	Pin ESP32 Wemos D1 R32
VCC	3,3 V (pin 3V3)
GND	GND
SCL	GPIO22
SDA	GPIO21

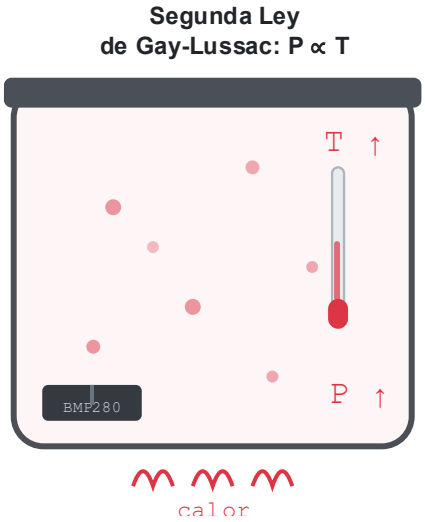
Para más información sobre la placa y el sensor, consultad el capítulo de microcontroladores y el de los sensores.

Material y montaje 1. Verificación de la segunda ley de Gay-Lussac, P vs T a V constante

- Un bote de vidrio hermético tipo tarro de conserva (500 - 800 mL) con tapa de rosca metálica
- Una fuente de calor (placa calefactora, cocina eléctrica o mechero Bunsen)
- Un recipiente grande para baño maría (cazo u olla)
- Agua del grifo y hielo

El montaje que aquí expongo es similar al que utilicé para estudiar esta misma ley, pero utilizando un termómetro digital y un teléfono móvil que disponía de sensor barométrico (Ramirez, 2021c).

Este montaje consiste en un recipiente rígido y hermético (bote de vidrio) cuyo interior contiene el sensor BME280. Al sumergir el bote en un baño maría, la temperatura del gas interior cambia y, al ser el volumen constante, la presión varía proporcionalmente.

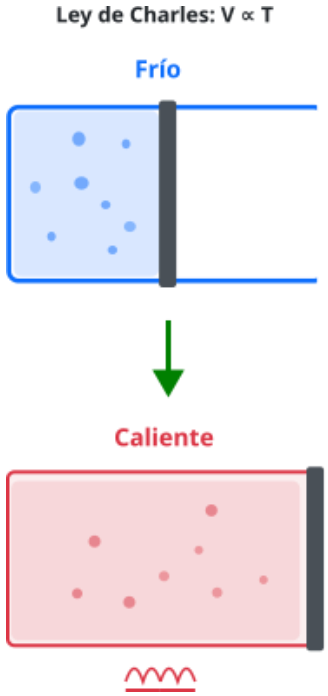


Se ha de usar un ESP32 con batería integrada (tipo TTGO T-Call) o un powerbank conectado al sensor y colocando todo dentro del bote con la placa funcionando, y cerrando bien la tapa. El ESP32 se conectará con Phyphox por bluetooth.

Se pone el bote de vidrio al baño maría en un cazo grande que le cubra casi todo con agua lo más fría posible y se va calentando. No se ha de sumergir la tapa del bote en el agua, solo la parte de vidrio.

Material y montaje 2. Verificación de la ley de Charles, V vs T a P constante

- Un vaso de precitados grande, de 600 – 1000 mL
- Mechero Bunsen o de alcohol
- Rejilla metálica y trípode o soporte
- Un tubo o jeringuilla de plástico transparente grande de 50–60 mL (tipo jeringa de alimentación o veterinaria, sin aguja).



Ya he explicado en mi bloc alguna forma clásica de realizar esta experiencia en la que se muestra la expansión térmica del aire al calentarse (Ramirez, 2021c), por ejemplo con un tubo de Mariotte.

Aquí propongo que el gas (el aire) esté en un tubo o jeringuilla grande de plástico que tiene el sensor en su interior en un extremo tapado y por el otro extremo está abierto.



Se introduce verticalmente en el agua contenida en un vaso de precipitados con el nivel interior y exterior a la misma altura (hacia la mitad del volumen del tubo, cuidando de que el sensor no se moje). El agua debe de estar lo más fría posible.

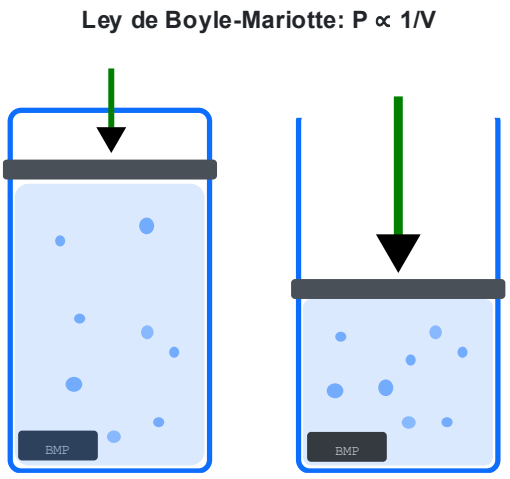
El vaso se coloca sobre un trípode o aro con rejilla metálica, sobre un Bunsen con la llama baja para que vaya calentando el agua poco a poco.

El sensor BME280 registrará la temperatura y presión interior. La presión debe permanecer prácticamente constante, para lo que iremos subiendo el tubo de manera que el agua esté al mismo nivel dentro y fuera.

Material y montaje 3: Verificación de la ley de Boyle-Mariotte, P vs V a T constante

- Jeringuilla grande de 50-60 mL (tipo jeringa de alimentación o veterinaria, sin aguja).
- Tubo de silicona o plástico flexible de 5-8 mm de diámetro interno (unos 15 cm)
- Vaselina o grasa de silicona para el sellado del émbolo
- Regla graduada o cinta métrica para medir el desplazamiento del émbolo

Una forma tradicional de llevar a cabo esta experiencia es la de utilizar una jeringuilla grande con un manómetro neumático acoplado en la salida (Ramirez, 2021c. Relació entre pressió i volum d'un gas amb xeringa i manòmetre), pero también se ha hecho con la jeringuilla y un sensor de presión conectado a un dattaloguer (Aparicio y Lozano, 2003c y d)



El montaje que aquí propongo también utiliza una jeringuilla grande como cámara de volumen variable, pero mediremos la presión interior mientras se comprime o expande el aire manualmente con el sensor BME280.

Una posibilidad es la de colocar el sensor BME280 en el exterior, unido a la jeringuilla mediante un tubo de silicona de unos 10 cm. La boquilla de la jeringuilla se conecta al tubo, y el otro extremo del tubo se conecta a una pequeña cámara sellada (un tapón de plástico) que aloja el sensor. De esta forma, cuando el émbolo se mueve, el aire se desplaza por el tubo y la presión es la misma en todo el sistema (principio de Pascal).Aquí la dificultad está en construir la cámara sellada con un par de tapones de plástico.

Yo prefiero pasar unos cables dupont por la boquilla de la jeringuilla y colocar el sensor dentro, aunque ocupe un poco del volumen. La boquilla por la que pasan los cables se sella con una pistola de silicona caliente.

Engrasad ligeramente el émbolo con vaselina o grasa de silicona para asegurar un deslizamiento suave y un sellado hermético. Introducid el embolo en la jeringuilla con un cordoncito de manera que el aire puede pasar entre el embolo y la pared de la jeringuilla. Cuando esté el embolo en la posición inicial de la experiencia (a volumen 60 mL, por ejemplo) se retira el hilo y la jeringuilla queda herméticamente cerrada.

Procedimiento y resultados esperados

Experimento 1: Gay-Lussac (P vs T a V constante)

Colocad el bote de vidrio (con el sensor dentro ya en marcha) en el recipiente grande lleno con agua fría (con hielo si es posible) hasta cubrir 3/4 del bote.

Conectad la app Phyphox al ESP32 e id registrando la temperatura y presión mientras se va calentando el agua lentamente con la placa calefactora o el bunsen.

Continuad hasta que el agua alcance unos 60-70 °C (no es necesario hervir). Se puede repetir el experimento una vez más para verificar la reproducibilidad.

Al representar la presión absoluta (P) (medida por el BME280 en Pa) frente a la temperatura absoluta (T) (en K, calculada como (T[°C] + 273,15), debe obtenerse una relación lineal que pasa por el origen (0, 0):

$$P = k \cdot T$$

donde (k = nR/V) es constante al ser (n) y (V) fijos.

En la práctica, la recta de regresión tendrá una ordenada en el origen cercana a cero. La pendiente (k) permite estimar el número de moles de aire dentro del bote si se conoce su volumen.

Valores aproximados para (V = 300) mL de aire a nivel del mar

T (°C)	T (K)	P esperada (hPa)
10	283,15	~1013
25	298,15	~1066
40	313,15	~1120
55	328,15	~1173
70	343,15	~1227

Experimento 2: Charles (V vs T a P constante)

Colocad el matraz Kitasato con el globo y el sensor (Montaje 2) sobre el trípode.
2. Tomad la temperatura ambiente inicial y el diámetro del globo en reposo.
3. Encended el mechero Bunsen y colocadlo debajo del matraz.
4. Cada 30 segundos, mientras el mechero calienta: Anotad la temperatura leída por el BME280 (mostrada en Phyphox o monitor serie).
Medid el diámetro del globo con la cinta métrica o fotografiadlo con una referencia de escala.
1. Continuad calentando hasta que el globo esté visiblemente inflado (unos 3-5 minutos).
2. Apagad el mechero y continuad midiendo durante el enfriamiento (otros 3-5 minutos).
3. Calculad el volumen aproximado del globo usando la fórmula de la semiesfera: ($V_{\text{globo}} \approx \frac{2}{3}\pi r^3$).

Al representar el volumen (V) dentro del tubo frente a la temperatura absoluta (T), debe obtenerse una relación lineal:

$$V = k' \cdot T$$

con (k' = nR/P) constante.

La extrapolación de la recta a (V = 0) debe cortar el eje de temperaturas en (T = -273,15) °C (cero absoluto), aunque la incertidumbre experimental puede dar valores entre -250 °C y -300 °C, dependiendo de la precisión de las medidas volumétricas.

Experimento 3: Boyle (P vs V a T constante)

Conectad el ESP32 con el sensor BME280 al sistema de jeringuilla y abrid Phyphox o el monitor serie para monitorizar la presión en tiempo real.

Colocad el émbolo de la jeringuilla en la marca de 60 mL (máximo volumen) y comenzad a registrad la presión (que inicialmente será la atmosférica ambiente).

Empujad el émbolo hasta 40–30 mL, de dos en dos mL. A continuación, realizad el proceso inverso: tirad del émbolo en pasos de 2 mL hasta los 60 mL, registrando la presión.

Se puede volver a realizar la serie completa (compresión + expansión) para evaluar la reproducibilidad.

Realizad el proceso lentamente (5 segundos entre escalones) para que el gas pueda equilibrarse térmicamente con el ambiente. Si el émbolo se mueve rápido, el aire se calienta por compresión adiabática y los datos no reflejarán la ley de Boyle (necesitáis un proceso isotermo).

Al representar la presión (P) frente al inverso del volumen (1/V), debe obtenerse una relación lineal que pasa por el origen:

$$P = k'' \cdot \frac{1}{V}$$

donde (k" = nRT) es constante.

Si en lugar de (P) vs (1/V) representamos (P) vs (V), debe obteneros una hipérbola característica de la relación de proporcionalidad inversa.

V (mL)	1/V (mL⁻¹)	P esperada (hPa)
50	0,020	1013
40	0,025	1266
30	0,033	1688
25	0,040	2026
20	0,050	2533

Análisis de datos para los tres experimentos

Exportad los datos desde Phyphox (formato CSV con separador punto y coma) o copiadlos del monitor serie.

Convertid las temperaturas de °C a K sumando 273,15.

Calculad la regresión lineal de la variable dependiente frente a la independiente:

- Gay-Lussac: (P) frente a (T) pendiente (k = nR/V)
- Charles: (V) frente a (T) pendiente (k' = nR/P)
- Boyle: (P) frente a (1/V) pendiente (k" = nRT)

Calculad el coeficiente de determinación (R^2) para evaluar la bondad del ajuste (valores > 0,95 indican buena concordancia con la ley).

Estimad el número de moles (n) de aire en el sistema:

- Para Gay-Lussac: ($n = k \cdot V / R$)
- Para Boyle: ($n = k''/(R \cdot T)$)

Comparad el valor experimental de (n) con el valor teórico:

$$n_{\text{teórico}} = \frac{P_{\text{atm}} \cdot V_{\text{sistema}}}{R \cdot T_{\text{ambiente}}}$$

donde ($P_{\text{atm}} \approx 101325$) Pa y ($V_{\text{sistema}}$) se mide en m³.

Calculad el error relativo:

$$\varepsilon_r = \frac{|n_{\text{exp}} - n_{\text{teórico}}|}{n_{\text{teórico}}} \times 100\%$$

Ampliación

Cálculo del cero absoluto
Usando los datos del experimento de Charles (Montaje 2), extended la recta de regresión hasta (V = 0) y encontrad la intersección con el eje (T). El resultado debe aproximarse a (-273,15) °C. Discutid las fuentes de desviación.

Efecto de la humedad
El BME280 mide humedad relativa. ¿Influye la humedad en la presión del gas? Investigar cómo el vapor de agua (gas no ideal) modifica la presión

parcial de la mezcla usando la ley de Dalton: $\left(P_{\text{total}} = P_{\text{aire seco}} + P_{\text{vapor}}(T)\right)$.

Corrección por gas real
Para presiones altas (émbolo comprimido a 20 mL), la presión puede alcanzar ~2500 hPa, donde los gases reales empiezan a desviarse del modelo ideal. Calculad el factor de compresibilidad (Z = PV/nRT) y comparadlo con (Z = 1) del gas ideal.

Experimento combinado
Usad la jeringuilla (Montaje 3) para variar el volumen, pero esta vez sumergiendola en agua caliente para cambiar también la temperatura. Aplicad la ley combinada: $(P_1V_1/T_1 = P_2V_2/T_2)$.

Código

Código para ESP32. Este código funciona para los tres montajes. Lee del BME280 la presión, temperatura en ºC y Kelvins y humedad relativa y los envía al monitor serie y por Bluetooth BLE a la app Phyphox en el smartphone.

```
/*
Verificación de las leyes de los gases

Placa: ESP32 Wemos D1 R32
Sensor: BME280
Comunicación: Bluetooth BLE → Phyphox y Serial (formato CSV)
*/

#include <Wire.h>                 // Biblioteca para comunicación I2C
#include <Adafruit_Sensor.h>      // Biblioteca base unificada de sensores Adafruit
#include <Adafruit_BME280.h>      // Biblioteca específica para el sensor BME280
#include <phyphoxBle.h>           // Biblioteca para comunicación BLE con Phyphox

Adafruit_BME280 bme;

// Variables para almacenar las lecturas
float presion_Pa;
float presion_hPa;
float temperatura_C;
float temperatura_K;

void setup() {
  Serial.begin(115200);

  // Cabecera para la hoja de cálculo. Solo imprimimos esto al inicio.

Serial.println("Tiempo(s);Presion(hPa);Temperatura(C);Temperatura(K)");

  Wire.begin(21, 22);  // SDA = GPIO21, SCL = GPIO22

  if (!bme.begin(0x76)) {
    Serial.println("ERROR:No_BME280");
    while (1) { delay(100); }
  }

  // Inicializar la comunicación Bluetooth BLE
  PhyphoxBLE::start("Exp_Gases");

  PhyphoxBleExperiment experimento;
  experimento.setTitle("Leyes de los Gases");
  experimento.setCategory("Termodinámica");

  PhyphoxBleExperiment::View view;

  // --- GRÁFICO 1: Presión vs Tiempo ---
  PhyphoxBleExperiment::Graph graficoP;
  graficoP.setLabel("Presión vs Tiempo");
  graficoP.setUnitX("s");
  graficoP.setUnitY("hPa");
  graficoP.setChannel(0, 1);       // Canal 0 → X (tiempo app), Canal 1 → Y (presión)
  view.addElement(graficoP);

  // --- GRÁFICO 2: Temperatura vs Tiempo ---
  PhyphoxBleExperiment::Graph graficoT;
  graficoT.setLabel("Temperatura vs Tiempo");
  graficoT.setUnitX("s");
  graficoT.setUnitY("°C");
  graficoT.setChannel(0, 2);       // Canal 0 → X (tiempo app), Canal 2 → Y (Temp °C)
  view.addElement(graficoT);

  // --- GRÁFICO 3: Presión vs Temperatura (Ley de Gay-Lussac) ---
  PhyphoxBleExperiment::Graph graficoPT;
  graficoPT.setLabel("Presión vs Temperatura");
  graficoPT.setUnitX("K");         // Eje X: Temperatura Absoluta
  graficoPT.setUnitY("hPa");       // Eje Y: Presión
  graficoPT.setChannel(3, 1);      // Canal 3 → X (Temp K), Canal 1 → Y (presión)
  view.addElement(graficoPT);
```

```
  experimento.addView(view);
  PhyphoxBLE::addExperiment(experimento);
}

void loop() {
  // --- LECTURA Y CÁLCULO ---
  presion_Pa = bme.readPressure();
  presion_hPa = presion_Pa / 100.0;

  temperatura_C = bme.readTemperature();
  temperatura_K = temperatura_C + 273.15; // Conversión a Kelvins

  // Capturar el tiempo actual en segundos para la hoja de cálculo
  float tiempo_s = millis() / 1000.0;

  // --- MOSTRAR EN MONITOR SERIE (Formato CSV puro) ---
  // Imprime: Tiempo(s); Presion(hPa); Temperatura(C); Temperatura(K)
  Serial.print(tiempo_s, 2);
  Serial.print(";");
  Serial.print(presion_hPa, 2);
  Serial.print(";");
  Serial.print(temperatura_C, 2);
  Serial.print(";");
  Serial.println(temperatura_K, 2); // println para el salto de línea

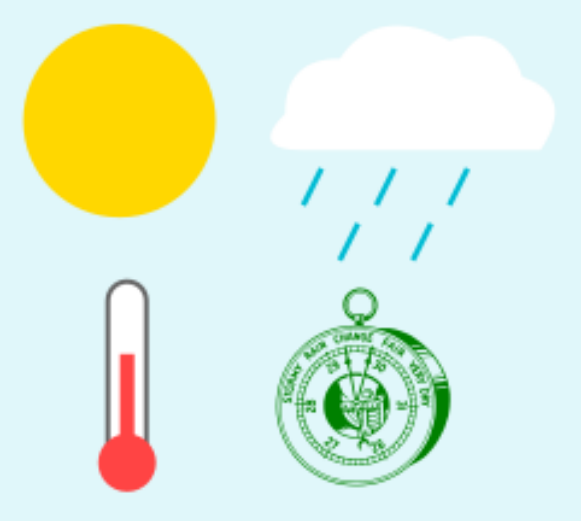
  // --- ENVIAR POR BLUETOOTH A PHYPHOX ---
  // Canal 1 (Presión), Canal 2 (Temp °C), Canal 3 (Temp K)
  PhyphoxBLE::write(presion_hPa, temperatura_C, temperatura_K);

  // Esperar 500 ms (2 muestras por segundo)
  delay(500);

  // Mantener activa la comunicación BLE
  PhyphoxBLE::poll();
}
```

Estación meteorológica

Vamos a construir una estación meteorológica autónoma que registre temperatura, humedad, presión atmosférica y altitud aproximada, almacenando los datos en una tarjeta microSD para su posterior análisis, y opcionalmente transmitiéndolos por Bluetooth a un dispositivo móvil.



En internet encontrareis infinidad de propuestas para crear vuestra estación meteorológica con diferentes tipos de sensores, y en particular con el sensor BME280, que es el que vuelvo a proponeros utilizar. Por ejemplo, esta que propone leer los valores que se van obteniendo en una pequeña pantalla (Marcelo, 2021), o esta que envía los datos por wifi a la plataforma ThingSpeak para que puedan ser leídos de allí en cualquier sitio (Larios, 2023).

Fundamento físico

Una estación meteorológica mide parámetros atmosféricos fundamentales: la temperatura del aire (relacionada con la energía cinética media de las moléculas), la humedad relativa (porcentaje de vapor de agua respecto al máximo que puede contener el aire a esa temperatura), la presión atmosférica (peso de la columna de aire por unidad de superficie, que varía con la altitud según la ecuación barométrica ($P = P_0 \cdot e^{-M \cdot g \cdot h/(R \cdot T)}$), y la altura aproximada (derivada de la presión mediante la fórmula hipsométrica).

El registro continuo de estos datos permite estudiar patrones climáticos, predecir cambios meteorológicos y comprender la relación entre las variables atmosféricas.

Materiales y montaje experimental

Necesitaréis una placa Wemos D1 R32 (ESP32), un sensor BME280 (presión, temperatura y humedad), un módulo lector de tarjetas microSD (SPI), una tarjeta microSD, una protoboard, cables Dupont, una batería externa (powerbank) para alimentación autónoma o un cargador de móvil y un ordenador con el IDE de Arduino.

Para más detalles sobre el sensor BME280, consultad el capítulo de los sensores y para la tarjeta microSD, ver el capítulo correspondiente.

Conectad el sensor BME280 a la placa por I²C: VCC a 5 V, GND a GND, SCL a pin 22 (ESP32) y SDA a pin 21 (ESP32). conecte el módulo microSD por SPI: VCC a 5 V, GND a GND, CS a pin 5, MOSI a pin 23, MISO a pin 19 y SCK a pin 18.

Colocad el sensor BME280 en un lugar ventilado pero protegido de la radiación solar directa y la lluvia. La placa y el módulo SD deben ir mejor en una caja estanca.

Formatead la tarjeta microSD en FAT32. conectad la placa al ordenador, cargad el código y verificad que se crea el archivo de datos en la SD. Desconectad la placa y alimentadla con el powerbank. Dejad la estación registrando datos durante al menos 24 horas, con un intervalo de muestreo de 5 minutos. Retirad la tarjeta SD y copiad los datos al ordenador para su análisis.

Resultados esperados

En un día estable, la temperatura mostrará un ciclo diario con un máximo al mediodía y un mínimo al amanecer. La presión atmosférica variará típicamente entre 1010 y 1030 hPa al nivel del mar, descendiendo antes de las borrascas. La humedad relativa será inversamente proporcional a la

temperatura (más humedad por la noche, menos al mediodía).

Representad gráficamente las tres variables frente al tiempo. Calculad la media, máximo y mínimo de cada una. Observad la correlación inversa entre temperatura y humedad. Determinad la altura sobre el nivel del mar a partir de la presión media usando la ecuación barométrica. Comparad sus datos con los de una estación meteorológica oficial cercana.

Ampliación

Añadid un sensor de luz BH1750 para medir la irradiancia solar al estilo de lo que propongo más adelante en la actividad “Medida de las variables ambientales durante un eclipse solar”.

Implementad la transmisión WiFi de los datos a una página web local o a un servicio como ThingSpeak, como hace Eder Larios (2023).

Código

```
// Estación meteorológica con BME280 y ESP32, y registro en microSD
// Conexiones BME280 (I2C): VCC a 5V, GND a GND, SCL a 22, SDA a 21
// Conexiones microSD (SPI): VCC a 5V, GND a GND, CS a 5, MOSI a 23, MISO a 19, SCK a 18

#include "Seeed_BME280.h" // Libreria del sensor BME280 (presion, temperatura, humedad) via I2C
#include <Wire.h>  // Libreria para comunicacion I2C (bus del BME280)
#include <SD.h>  // Libreria para tarjetas microSD via SPI

#define CS_SD 5 // Pin Chip Select (CS) para la tarjeta microSD en el modulo SPI

BME280 bme280; // Creamos el objeto para el sensor BME280
File archivoDatos; // Manejador para el archivo de datos en la microSD

void setup() {
  // Configuracion inicial: se ejecuta una vez al arrancar la placa
  Serial.begin(9600); // Iniciamos comunicacion serie a 9600 baudios

  // Iniciamos el sensor BME280 por el bus I2C
  if (!bme280.init()) { // Si la inicializacion falla...
    Serial.println("Error BME280"); // ...mostramos mensaje de error
    while (1) { delay(100); } // ...y salimos del setup sin continuar
  }

  // Iniciamos la tarjeta microSD en el bus SPI
  if (!SD.begin(CS_SD)) { // Si no se detecta la tarjeta...
    Serial.println("Error SD"); // ...mostramos mensaje de error
    return; // ...y continuamos sin SD (solo datos por serie)
  }

  // Creamos el archivo de datos con cabecera de columnas
  archivoDatos = SD.open("meteo.txt", FILE_WRITE); // Abrimos (o creamos) el archivo en modo escritura

  if (archivoDatos) { // Si se abrió correctamente...
    // Añadimos la columna 'altitud_m' a la cabecera CSV

archivoDatos.println("tiempo_ms;temp_C;presion_Pa;humedad_%;altitud_m");
    archivoDatos.close(); // Cerramos el archivo para liberar el buffer
  }
}

void loop() {
  // Bucle principal: se repite indefinidamente
  float temp = bme280.getTemperature();     // Leemos temperatura del BME280 en grados Celsius
  float presion = bme280.getPressure();     // Leemos presion atmosferica en pascales (Pa)
  float humedad = bme280.getHumidity();     // Leemos humedad relativa en porcentaje (0-100%)

  // Calculamos la altitud sobre el nivel del mar usando la presión leída
  float altitud = bme280.calcAltitude(presion);

  // Mostramos los datos por el monitor serie (para visualizacion en tiempo real)
  Serial.print(millis()); // Tiempo transcurrido desde el arranque en milisegundos
  Serial.print(";");
```

```
  Serial.print(temp, 2); // Temperatura con 2
decimales
  Serial.print(";");
  Serial.print(presion, 1); // Presion con 1 decimal
  Serial.print(";");
  Serial.print(humedad, 1); // Humedad con 1
decimal
  Serial.print(";");
  Serial.println(altitud, 2); // Altitud con 2 decimales
(ultimo campo, con salto de linea)

  // Guardamos los mismos datos en la tarjeta
microSD (para registro historico)
  archivoDatos = SD.open("meteo.txt",
FILE_APPEND); // Abrimos en modo anadir (sin
sobrescribir)

  if (archivoDatos) { // Si se abrio correctamente...
    archivoDatos.print(millis()); // Escribimos cada
campo en la tarjeta SD
    archivoDatos.print(";");
    archivoDatos.print(temp, 2);
    archivoDatos.print(";");
    archivoDatos.print(presion, 1);
    archivoDatos.print(";");
    archivoDatos.print(humedad, 1);
    archivoDatos.print(";");
    archivoDatos.println(altitud, 2); // Registramos la
altitud y hacemos salto de línea
    archivoDatos.close(); // Cerramos el archivo para
asegurar que se escriban los datos
  }

  delay(300000); // Esperamos 5 minutos (300000
ms) entre muestras para uso como estacion
meteorologica
}
```

Luz y Color

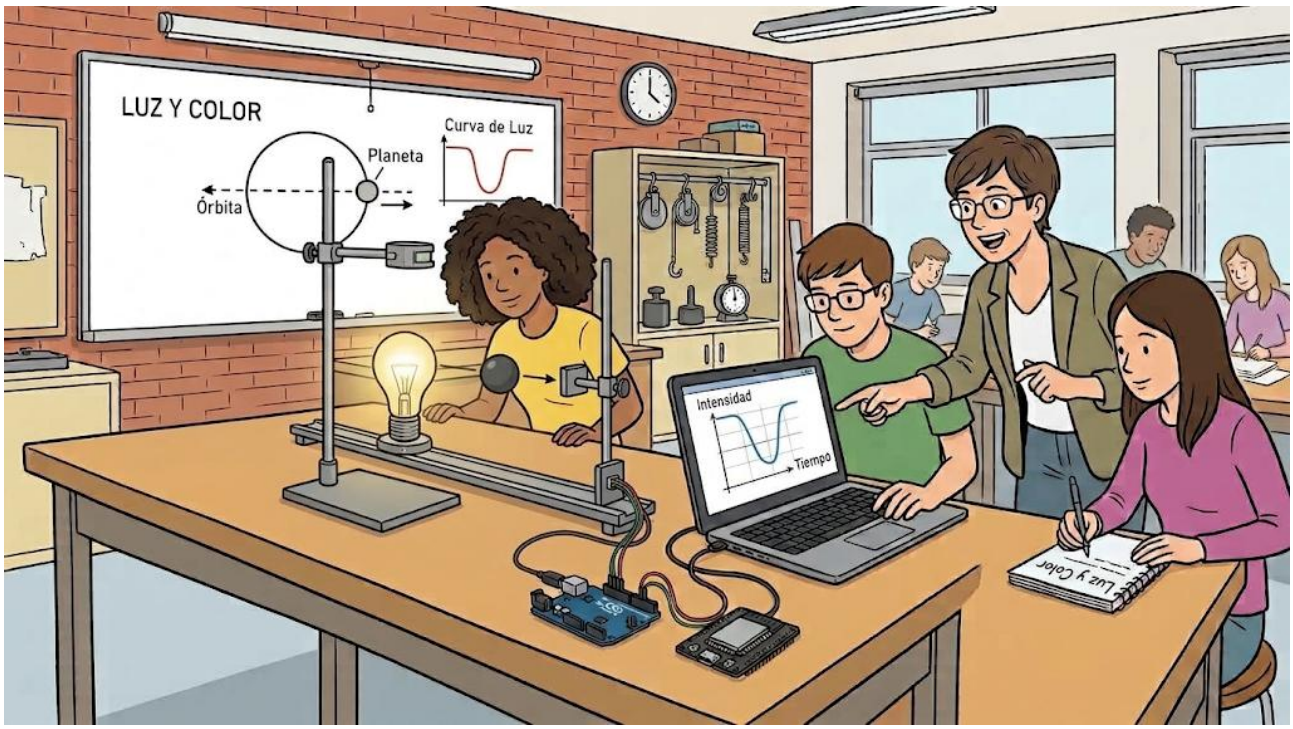
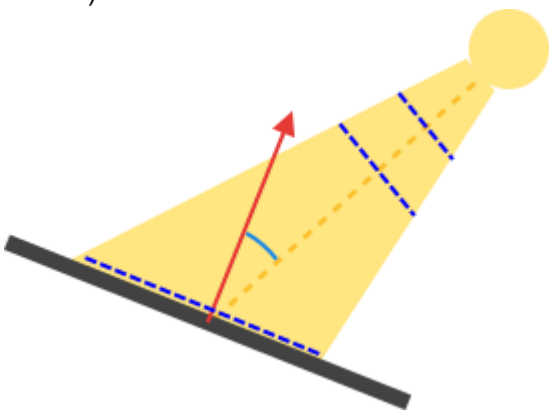


Ilustración realizada con Gemini. Esté pensada para adornar, aunque el alumnado puede entretenerse en buscar los errores evidentes que contiene.

Irradiancia y ley de Lambert

Conocer y comparar diferentes sensores de luz para Arduino y ESP32 —el sensor digital BH1750 (o el TSL2591), la fotorresistencia LDR y el fototransistor TEMT6000—, aprender a conectarlos y programarlos, y utilizarlos para medir la irradiancia luminosa y estudiar la ley de Lambert (variación de la intensidad con la distancia y el ángulo de incidencia).



Este experimento se puede realizar satisfactoriamente con el sensor de luz de un teléfono móvil (Ramirez, 2017b), así que debéis valorar si merece la pena hacerlo con arduinos.

Fundamento físico

La irradiancia o iluminancia es la potencia luminosa por unidad de área que llega a una superficie. La ley de Lambert establece que la intensidad de la luz emitida por una fuente puntual disminuye inversamente con el cuadrado de la distancia ($I \propto 1/d^2$), y que la iluminancia sobre una superficie es proporcional al coseno del ángulo de incidencia de los rayos luminosos respecto a la normal ($E = E_0 \cdot \cos\theta$).

Los sensores de luz convierten la energía luminosa en una señal eléctrica. El BH1750 es un sensor digital que mide la iluminancia en lux con una resolución de hasta 0,5 lux mediante un fotodiodo con respuesta espectral similar al ojo humano, comunicándoos por I²C.

La fotorresistencia LDR (Light Dependent Resistor) es un componente cuyo valor resistivo disminuye al aumentar la intensidad de luz incidente (valores típicos: 10 MΩ en oscuridad, 5 kΩ con luz brillante), basándoos en el efecto fotoeléctrico interno en un semiconductor de CdS.

El fototransistor TEMT6000 es un transistor sensible a la luz que genera una corriente proporcional a la irradiancia, con una velocidad de respuesta superior a la de la LDR.

Materiales y montaje experimental

Necesitaréis una placa Wemos D1 R32 (ESP32) o Arduino UNO, un sensor digital BH1750 (módulo GY-302 o BH1750FVI), una fotorresistencia LDR (GL55 o similar), un fototransistor TEMT6000, una resistencia de 10 kΩ, una protoboard, cables Dupont, una linterna o foco LED de luz blanca, una regla, un transportador de ángulos y, opcionalmente, una batería externa para funcionamiento autónomo.

Comenzad con el sensor BH1750: Conectad VIN a 5V (o 3,3V), GND a GND, SCL al pin SCL (A5 en Arduino UNO, pin 22 en ESP32) y SDA al pin SDA (A4 en UNO, pin 21 en ESP32). El pin ADDR se deja sin conectar (dirección I²C 0x23).

BH1750	Arduino UNO	ESP32
VCC	3,3 ó 5 V	3,3 ó 5 V
GND	GND	GND
SCL	A5	pin 22
SDA	A4	pin 21

Para la fotorresistencia LDR, montad un divisor de tensión: se conecta una pata de la LDR a 3,3V (o 5V) y la otra pata a una entrada analógica (A0 en Arduino UNO, pin 2 en ESP32). Conectad una resistencia de 10 kΩ entre esa misma entrada analógica y GND. También se puede usar el módulo KY-018 que ya incluye la resistencia integrada.

Para el fototransistor TEMT6000, la conexión es directa: GND a GND, VCC a 3,3V o 5V, y OUT a una entrada analógica (A0 en UNO, pin 2 o 34 en ESP32).

Colocad la fuente de luz fija en un extremo de la regla y el sensor en el otro extremo, enfrentados. Para los experimentos con ángulos, usad el transportador para inclinar el sensor respecto a la dirección del haz.

Procedimiento 1: Luxómetro con BH1750 y Phyphox.
Cargad el primer código (BH1750 + Phyphox) en la placa ESP32. Abrid la aplicación Phyphox en el móvil, activad Bluetooth y localización, pulse el botón «+» y seleccionad «Añadir por Bluetooth». Buscad el dispositivo «Luxómetro».
Tomad medidas de iluminancia ambiental en distintas condiciones: luz solar directa, sombra, interior con fluorescentes, oscuridad.
Medid la iluminancia a diferentes distancias de una fuente puntual (10 cm, 20 cm, 40 cm, 80 cm) y verificad la ley de inversa del cuadrado.

Procedimiento2: Fotoresistencia LDR con Phyphox.
Cargad el código de la LDR con Phyphox. Observad cómo varía la lectura (en unidades arbitrarias) al tapar y destapar el sensor.
Probad distintos valores de la resistencia de calibración para ajustar la sensibilidad.

Procedimiento3: Fototransistor TEMT6000.
Repetid los experimentos 2 y 3 con el fototransistor. Comparad la velocidad de respuesta y la calidad de la señal con la LDR.

Resultados esperados y análisis

Con el BH1750 obtendréis medidas precisas hasta 65.000 lux (luz solar directa ≈ 30.000–100.000 lux, día nublado ≈ 1.000–5.000 lux, interior con fluorescentes ≈ 300–500 lux, atardecer ≈ 10–50 lux, oscuridad < 1 lux).

Al variar la distancia, el producto lux·x distancia² debe permanecer aproximadamente constante, verificando la ley de inversa del cuadrado.

Con la LDR y el fototransistor, las lecturas en unidades arbitrarias seguirán la misma tendencia.

Representad gráficamente la iluminancia en función de la distancia y ajustad a una función $E = k/d^2$.

Calculad el valor de k y su incertidumbre. Para el experimento angular, representad E frente a cos θ y verificad la linealidad.

Discutid las ventajas e inconvenientes de cada sensor: el BH1750 da valores absolutos en lux pero es lento (120 ms en alta resolución), la LDR es barata y rápida pero no calibrada en lux, el fototransistor es más rápido que la LDR pero también da valores relativos.

Considerad también el efecto del ruido eléctrico en las lecturas del ESP32 (ADC no lineal, interferencias de la antena Bluetooth).

Ampliación

Construid un fotómetro para medidas de transmitancia colocando un LED y el sensor enfrentados con una cubeta entre ambos, para estudiar la absorción de disoluciones coloreadas, lo que haremos en el siguiente experimento (ley de Beer-Lambert).

También se puede usar el sensor APDS9960 del Arduino Nano 33 BLE Sense para medidas de color e intensidad más precisas.

Código

Código 1: BH1750 + Phyphox (ESP32)

```
/*
 * Luxómetro con sensor BH1750 y Phyphox vía BLE
 * Adaptado del ejemplo BH1750test y Rangefinder de Phyphox BLE
 *
 * Conexiones:
 *   VCC  → 3V3 o 5V
 *   GND  → GND
 *   SCL  → SCL (pin 22 en ESP32)
 *   SDA  → SDA (pin 21 en ESP32)
 *   ADDR → desconectado (dirección 0x23)
 *
 * Placa: ESP32 DEV Module o DOIT ESP32 DEVKIT V1
 */

#include <BH1750.h>
#include <Wire.h>
#include <phyphoxBle.h>

BH1750 lightMeter;

void setup() {
  Serial.begin(9600);
  Wire.begin();

  // Modo continua baja resolución (4 lux, 16 ms) →
```

```
más medidas/s

lightMeter.begin(BH1750::CONTINUOUS_LOW_RES_MODE);
  // Alternativas:
  // CONTINUOUS_HIGH_RES_MODE     (1 lux, 120 ms)
  // CONTINUOUS_HIGH_RES_MODE_2   (0,5 lux, 120 ms)
  // ONE_TIME_HIGH_RES_MODE, etc.

  Serial.println(F("BH1750 Test begin"));

  PhyphoxBLE::start("Luxometro");

  PhyphoxBleExperiment experimento;
  experimento.setTitle("Luxómetro");
  experimento.setCategory("Arduino Experiments");
  experimento.setDescription("Dibuja la gráfica de la iluminancia (lux) frente al tiempo.");

  PhyphoxBleExperiment::View view;
  PhyphoxBleExperiment::Graph graph;
  graph.setLabel("Iluminancia");
  graph.setUnitX("s");
  graph.setUnitY("lux");
  graph.setLabelX("tiempo");
  graph.setLabelY("Iluminancia");
  graph.setChannel(1, 2); // Canal 1: tiempo, Canal 2: lux

  view.addElement(graph);
  experimento.addView(view);
  PhyphoxBLE::addExperiment(experimento);
}

void loop() {
  float t = 0.001 * (float)millis(); // Tiempo en segundos
  float lux = lightMeter.readLightLevel();

  PhyphoxBLE::write(t, lux); // Envía los datos a Phyphox

  Serial.print("Iluminancia: ");
  Serial.print(lux);
  Serial.println(" lx");

  // delay(1000); // Descomentar para espaciar las medidas
}
```

Código 2: LDR + Phyphox (ESP32)

```
/*
 * Medida de luminosidad con fotoresistencia LDR y Phyphox
 *
 * Conexiones: LDR en serie con resistencia de 10 kΩ
 *   LDR entre pin 2 (ESP32) y VCC (3,3V)
 *   Resistencia 10k entre pin 2 y GND
 *
 * Placa: ESP32 DEV Module
 */

#include <phyphoxBle.h>

const long A = 10000;   // Resistencia LDR en oscuridad (kΩ)
const int B = 15;       // Resistencia LDR con luz brillante (kΩ)
const int Rc = 10;      // Resistencia de calibración (kΩ)
const int LDRPin = 2;   // Pin analógico del LDR

void setup() {
  PhyphoxBLE::start("Sensor de luz");

  PhyphoxBleExperiment experimento;
  experimento.setTitle("Sensor de luz LDR");
  experimento.setCategory("Arduino Experiments");
  experimento.setDescription("Luminosidad (u.a.) frente al tiempo.");

  PhyphoxBleExperiment::View view;
  PhyphoxBleExperiment::Graph graph;
  graph.setLabel("Luminosidad");
  graph.setUnitX("s");
  graph.setUnitY("u.a.");
  graph.setLabelX("tiempo");
  graph.setLabelY("Luminosidad");
  graph.setChannel(1, 2);

  view.addElement(graph);
  experimento.addView(view);
  PhyphoxBLE::addExperiment(experimento);

  Serial.begin(9600);
}

void loop() {
  float t = 0.001 * (float)millis();
  float V = analogRead(LDRPin);
  // Fórmula con LDR entre VCC y pin (resistencia a GND)
```

```
  float luminosidad = ((long)V * A * 10) / ((long)B *
Rc * (4095 - V));

  PhyphoxBLE::write(t, luminosidad);

  Serial.print(t, 3);
  Serial.print("; ");
  Serial.println(luminosidad);

  PhyphoxBLE::poll(); // Solo necesario para Nano
33 IoT, pero inofensivo en otras placas
}
```

Código 3: Fototransistor TEMT6000 + Phyphox (ESP32)

```
/*
 * Medida de luminosidad con fototransistor
TEMT6000 y Phyphox
 *
 * Conexiones:
 *   GND → GND
 *   VCC → 3,3V o 5V
 *   OUT → pin 2 (ESP32)
 *
 * Placa: ESP32 DEV Module
 */

#include <phyphoxBle.h>

const int LDRPin = 2; // Pin analógico

void setup() {
  PhyphoxBLE::start("Fototransistor");

  PhyphoxBleExperiment experimento;
  experimento.setTitle("Sensor de luz
Fototransistor");
  experimento.setCategory("Arduino Experiments");
  experimento.setDescription("Luminosidad (u.a.)
frente al tiempo.");

  PhyphoxBleExperiment::View view;
  PhyphoxBleExperiment::Graph graph;
  graph.setLabel("Luminosidad");
  graph.setUnitX("s");
  graph.setUnitY("u.a.");
  graph.setLabelX("tiempo");
  graph.setLabelY("Luminosidad");
  graph.setChannel(1, 2);

  view.addElement(graph);
  experimento.addView(view);
  PhyphoxBLE::addExperiment(experimento);

  Serial.begin(9600);
}

void loop() {
  float t = 0.001 * (float)millis();
  float V = analogRead(LDRPin);

  PhyphoxBLE::write(t, V);

  Serial.print(t, 3);
  Serial.print("; ");
  Serial.println(V);

  PhyphoxBLE::poll();
}
```

Variación de la iluminancia con la inclinación

Vamos ahora a medir cómo varía la iluminancia sobre una superficie en función del ángulo de incidencia de la luz y verificar la ley del coseno de Lambert: ($E = E_0 \cos\theta$).

Fundamento físico

Cuando una superficie plana recibe luz con un ángulo θ respecto a la normal, la iluminancia efectiva disminuye según:

$$E = E_0 \cos\theta$$

donde (E_0) es la iluminancia a incidencia normal ($\theta = 0°$).

Materiales y montaje experimental

Arduino UNO o ESP32
Sensor BH1750
Transportador de ángulos
Soporte giratorio
Fuente de luz direccional (linterna LED)
Protoboard y cables

Fijad el BH1750 sobre un soporte que pueda girar en un plano vertical y colocad una linterna a una distancia fija (~50 cm) apuntando al sensor.

Medid (E_0) con el sensor perpendicular a la luz ($\theta = 0°$) e id girando el sensor en incrementos de 10º hasta 80º.

El dispositivo ira registrando (E) para cada ángulo. Al final copiad los valores y graficad (E/E_0) vs ($\cos\theta$) en una hoja de cálculo o programa específico de análisis de datos.

Análisis

Ajustad la línea de tendencia a $E = E_0 \cos\theta$ y visualizad el coeficiente de determinación (R^2). Analizad si nos alejamos de lo esperado cuando los ángulos son más altos ($\theta > 70°$).

Código

Se puede utilizar el código 1 para enviar los datos a Phyphox o este más sencillito que solo envía los valores por el puerto serie.

```
#include <Wire.h>
#include <BH1750.h>

BH1750 luxometro;  // Creamos el objeto para el sensor de luz BH1750

void setup() {  // Configuracion inicial: se ejecuta una vez al arrancar la placa
   Serial.begin(9600);  // Iniciamos comunicacion serie a 9600 baudios para enviar datos al PC
   Wire.begin();  // Iniciamos el bus de comunicacion I2C para conectar sensores
   luxometro.begin();  // Inicializamos el sensor con la direccion I2C por defecto
   Serial.println("Angulo(deg);Iluminancia(lx)");  // Enviamos el tiempo de carga al monitor serie
}

void loop() {  // Bucle principal: se repite indefinidamente mientras la placa este alimentada
   float lux = luxometro.readLightLevel();  // Leemos la iluminancia actual del sensor BH1750 en lux
   Serial.println(lux);  // Enviamos el valor de iluminancia al monitor serie
   delay(500);  // Esperamos 500 ms (1 segundo) entre mediciones
}
```

Ley de Beer-Lambert

Verificar experimentalmente la ley de Lambert-Beer, que relaciona la absorbancia de una disolución con la concentración del soluto y la longitud del camino óptico, midiendo la intensidad de luz transmitida a través de disoluciones de diferente concentración con un sensor de luz.

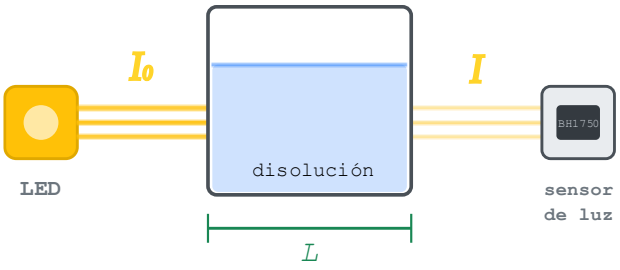
Medir la absorción de luz al atravesar diferentes materiales (vidrio, plástico, agua coloreada) y verificar la ley de Beer-Lambert.

Este experimento se puede realizar satisfactoriamente con el sensor de luz (Ramirez, 2020) o con la cámara de un teléfono móvil (Hubbard, 2021), así que debéis valorar si merece la pena hacerlo con arduinos.

Fundamento físico

Según la ley de Beer-Lambert la intensidad de un haz luminoso disminuye exponencialmente al atravesar un medio:

$$I = I_0 \cdot 10^{-\varepsilon c L}$$



El exponente de la ecuación se denomina absorbancia (A) de la disolución y es directamente proporcional a la concentración (c) del soluto absorbente y a la longitud del camino óptico (L):

$$A = \varepsilon \cdot c \cdot L,$$

donde ε es el coeficiente de absorción molar (una constante que depende de la sustancia y de la longitud de onda).

La absorbancia se define como $A = -\log_{10}(I/I_0)$, siendo I_0 la intensidad de luz incidente (disolvente puro) e I la intensidad transmitida a través de la disolución problema. Esta ley es fundamental en espectrofotometría y permite determinar concentraciones desconocidas por comparación con patrones.

Materiales y montaje experimental

Necesitaréis una placa Wemos D1 R32 (ESP32) o Arduino UNO, un sensor de luz digital BH1750, una linterna LED de luz blanca de alta luminosidad como fuente de luz, una protoboard y cables Dupont.

También una cubeta o vaso de precipitados pequeño y transparente, colorante alimentario (azul o rojo), agua destilada, una pipeta o jeringa graduada y una regla.



Conectad el BH1750 a la placa por I²C: VIN a 5 V, GND a GND, SCL a A5 (UNO) o pin 22 (ESP32) y SDA a A4 (UNO) o pin 21 (ESP32).

Colocad el LED blanco en un soporte fijo apuntando hacia el sensor BH1750, con la cubeta en medio, de modo que el haz de luz atraviese la cubeta horizontalmente antes de llegar al sensor. Hay que asegurarse de que el sistema esté con poca luz para evitar interferencias de la luz ambiental.

Procedimiento

Preparad una disolución madre de colorante alimentario (por ejemplo, 10 gotas en 100 mL de agua). Llenad la cubeta con agua destilada y medid la intensidad de luz transmitida (I_0). A continuación, medid la intensidad para concentraciones crecientes: 1, 2, 3, 4 y 5 mL de disolución madre añadidos a la cubeta (completad con agua destilada hasta el mismo volumen total cada vez).



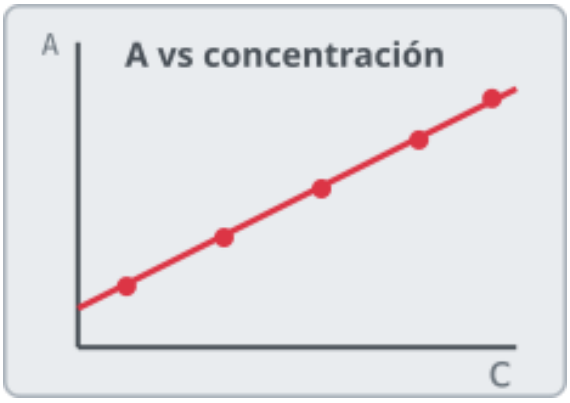
Registrad la intensidad de luz transmitida para cada concentración.

Resultados esperados

Al aumentar la concentración de colorante, la intensidad de luz transmitida disminuirá exponencialmente. La absorbancia $A = -\log_{10}(I/I_0)$ aumentará linealmente con la concentración.

Para un colorante azul, la absorbancia será mayor para la luz roja (absorbida por el azul) que para la luz azul.

Calculad la absorbancia para cada concentración. Representad A frente a la concentración c y realizad un ajuste lineal. La pendiente dará el producto ε·L (absortividad molar por camino óptico). Verificad si la relación es lineal en todo el rango o si se desvía a altas concentraciones (limitación de la ley de Lambert-Beer).



Discutid las fuentes de error: dispersión de luz por partículas, reflexiones en las paredes de la cubeta, variaciones en la intensidad del LED.

Ampliación

Repetid el experimento variando la longitud del camino óptico (superponiendo cubetas una detrás de otra con la misma disolución) para verificar la dependencia lineal con L. También se pueden usar diferentes colorantes y comparar sus coeficientes de absorción molar.

Código

```
// Medida de absorbancia con sensor BH1750
// Conexiones I2C: VIN a 5V, GND a GND, SCL a A5 (UNO) o 22 (ESP32), SDA a A4 (UNO) o 21 (ESP32)

#include <Wire.h>
#include <BH1750.h>

BH1750 sensorLuz;

void setup() {
  Serial.begin(9600);
  Wire.begin();
  sensorLuz.begin();
}

void loop() {
  float lux = sensorLuz.readLightLevel();

  Serial.print(millis());
  Serial.print(";");
  Serial.println(lux, 1);

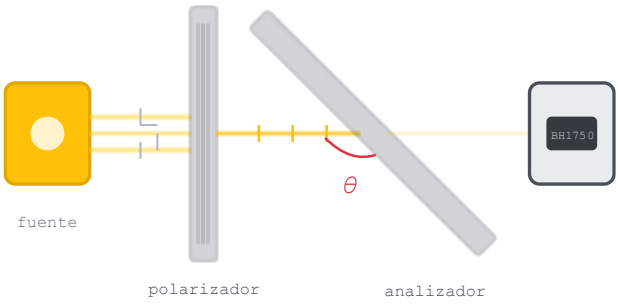
  delay(1000);
}
```

Polarización de la luz

Estudiaremos la polarización de la luz midiendo la intensidad transmitida a través de un polarizador en función del ángulo de rotación (ley de Malus: $(I = I_0 \cos^2 \theta)$.

Un sensor conectado a arduinos medirá la iluminancia (lux) tras el analizador para distintos ángulos, permitiendo verificar experimentalmente la relación con el cuadrado del coseno.

Este experimento también se puede hacer muy bien con el sensor de luz de un teléfono móvil (Monteiro, 2016).



Fundamento físico

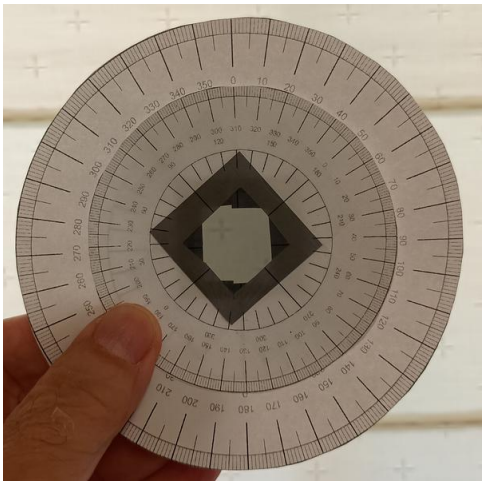
La luz natural no polarizada vibra en todas las direcciones perpendiculares a su dirección de propagación. Al pasar por un polarizador lineal, solo la componente del campo eléctrico paralela al eje de transmisión atraviesa el filtro, resultando en luz polarizada linealmente. Si esta luz incide sobre un segundo polarizador (analizador) girado un ángulo (θ) respecto al primero, la intensidad transmitida sigue la ley de Malus:

$$I(\theta) = I_0 \cos^2 \theta$$

donde (I_0) es la intensidad máxima (cuando ambos ejes están alineados, (θ = 0º) e (I_θ) la intensidad transmitida para un ángulo (θ) cualquiera. La intensidad se anula cuando los ejes están cruzados (θ = 90º).

Materiales y montaje experimental

Arduino UNO o ESP32 y sensor BH1750.
2 filtros polarizadores lineales en un soporte rotatorio con escala angular.
Fuente de luz (LED blanco)



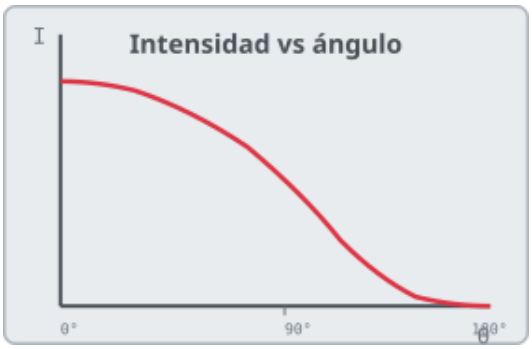
Un polarizador sobre otro formando un ángulo de 50º

Alinead el LED, el primer polarizador (fijo), el segundo polarizador (giratorio) y el sensor BH1750 en línea recta.

Girad el segundo polarizador en incrementos de 10° y registrad la intensidad. Después graficad (I/I_0) vs (θ).

Análisis

Ajustad a $(I = I_0 \cos^2(\theta - \phi))$ y verificad la ley de Malus. Comprobareis que el ángulo de extinción corresponde a 90º.



Código

```
#include <Wire.h>          // Libreria para comunicacion I2C
#include <BH1750.h>        // Libreria del sensor de luz digital BH1750
```

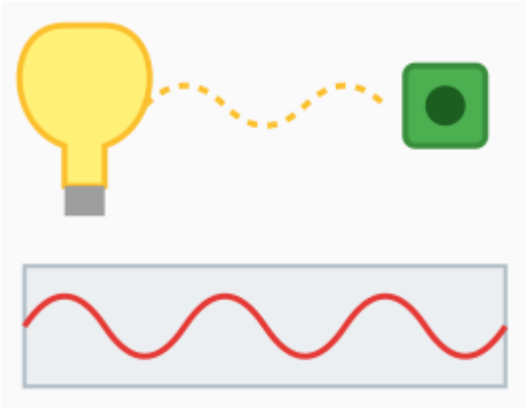
```
BH1750 sensor;            // Creamos el objeto para el
sensor de luz BH1750

void setup() {            // Configuracion inicial: se
ejecuta una vez al arrancar
   Serial.begin(9600);   // Iniciamos comunicacion
serie a 9600 baudios
   Wire.begin();         // Iniciamos el bus I2C
   sensor.begin();       // Inicializamos el sensor
BH1750
}

void loop() {             // Bucle principal: se repite
continuamente
   Serial.println(sensor.readLightLevel());  //
Leemos la iluminancia (lux) y la enviamos al PC
   delay(500);           // Esperamos 500 ms entre
mediciones
}
```

Análisis de la frecuencia de la corriente alterna mediante un fototransistor y Phyphox

Vamos a determinar experimentalmente la frecuencia de la corriente eléctrica alterna (50 Hz en Europa) midiendo el parpadeo de la luz emitida por una bombilla incandescente, halógena o LED, utilizando un fototransistor TEMT6000 conectado a un ESP32 con transmisión de datos por Bluetooth a Phyphox.



Esta actividad, que me gusta mucho, tradicionalmente la había realizado con sensores de luz conectados a dattalogers (y ordenador), por ejemplo, con los equipos EXAO de Adolf Cortel (1992) o MultiLog de Fourier Education (Aparicio y Lozano, 2004c y d). Más adelante la hacía con un teléfono móvil y la difunta aplicación Audiotime+, y más recientemente, después de darle muchas vueltas, lo he conseguido con un fototransistor y un ESP32 (Ramírez, 2026b), que es la propuesta que hago aquí.

Fundamento físico

Las bombillas conectadas a la red eléctrica de corriente alterna (CA) reciben un voltaje que varía sinusoidalmente con el tiempo: $V(t) = V_0 \cdot \text{sen}(2\pi ft)$, donde f es la frecuencia de la red (50 Hz en Europa, 60 Hz en América).

La potencia disipada por la bombilla es proporcional al cuadrado del voltaje: $P(t) \propto V^2(t) \propto \text{sen}^2(2\pi ft)$. Dado que $\text{sen}^2(\omega t) = (1 - \cos(2\omega t))/2$, la potencia (y por tanto la intensidad luminosa) oscila al doble de la frecuencia de red, es decir, a 100 Hz (o 120 Hz en América).

La luz parpadea, pues, 100 veces por segundo, aunque el ojo humano no percibe esta variación por el fenómeno que falsamente se ha dado en denominar persistencia retiniana.

Para detectar estas oscilaciones rápidas se necesita un sensor con velocidad de respuesta inferior a 1 ms y una tasa de muestreo de al menos 1000 muestras por segundo. El fototransistor TEMT6000 responde en microsegundos, por lo que es adecuado.

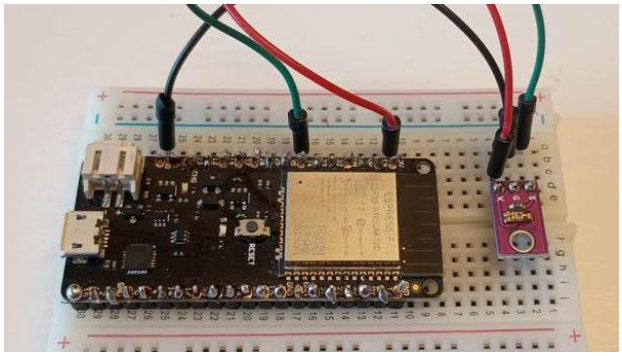
El ESP32 presenta el desafío de que su ADC es ruidoso y las transmisiones Bluetooth generan microcaídas de tensión que pueden enmascarar la señal. La solución consiste en usar los dos núcleos del ESP32: uno para la adquisición de datos (con sincronización precisa por microsegundos) y otro para la transmisión BLE por lotes, y opcionalmente añadir un condensador de 100 nF entre la salida del sensor y GND como filtro paso bajo para reducir el ruido de alta frecuencia.

Materiales y montaje experimental

Se necesita una placa Wemos D1 R32 o cualquier placa con ESP32 que tenga Bluetooth, un fototransistor TEMT6000, una placa de pruebas, cables Dupont, un condensador cerámico de 100 nF a 1 µF (opcional, para filtrar ruido) y varias bombillas para probar: incandescente, halógena, fluorescente compacta y LED. Necesitaremos también un teléfono móvil con la aplicación Phyphox instalada.

Conectad el fototransistor TEMT6000 a la placa ESP32: el pin GND a GND, el pin VCC a 3,3V o 5V, y el pin OUT a la entrada analógica 34 (GPIO34) del ESP32. Esta entrada es exclusivamente analógica (no tiene función digital), lo que evita conflictos.

Si observáis ruido excesivo en la señal, se ha de conectar un condensador cerámico de 100 nF entre el pin OUT del sensor y GND, lo más cerca posible del pin de la placa; esto actuará como filtro paso bajo suavizando los picos de ruido eléctrico de la antena Bluetooth.



Colocad el sensor apuntando hacia la bombilla que queráis analizar, a una distancia de 10–30 cm. Aseguraros de que no haya otras fuentes de luz variable interfiriendo.

Procedimiento

Cargad el código en el ESP32 seleccionando la placa ESP32 DEV Module en el IDE de Arduino.

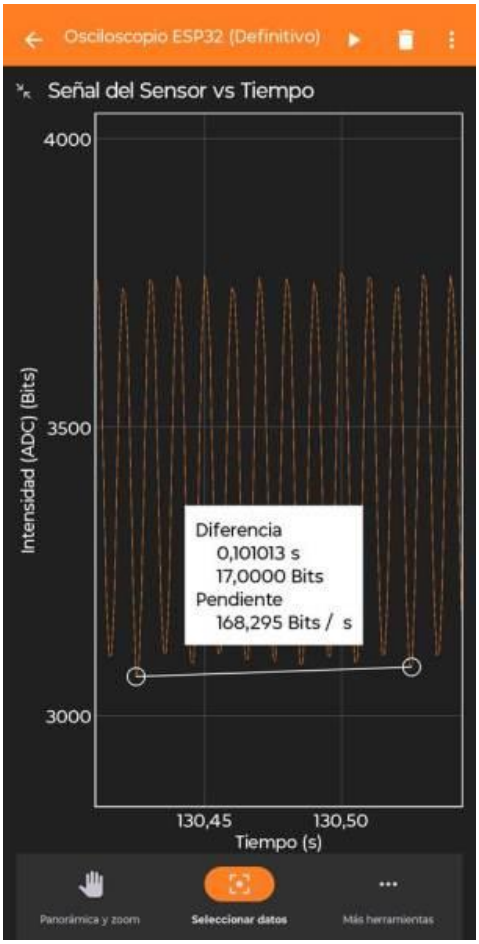
Abrid Phyphox en el móvil y configurad el experimento importando el archivo de configuración usando el código QR que he puesto antes del código para el ESP32.

Activad Bluetooth y localización en el teléfono y comenzad la toma de datos apuntando el sensor hacia una bombilla incandescente. Observaréis una onda sinusoidal con una frecuencia de 100 Hz. Medid el tiempo entre 10 picos consecutivos y calculad el período y la frecuencia.

Repetid con bombillas halógenas, fluorescentes compactas y LED. Tomad nota de las diferencias: algunas bombillas LED con electrónica de conmutación rápida pueden producir frecuencias mucho más altas (20–40 kHz) que aparecerán como ruido o una señal casi continua.

Resultados esperados y análisis

Para una bombilla incandescente o halógena conectada a la red de 50 Hz, la gráfica mostrará una oscilación sinusoidal con un período de 10 ms, correspondiente a una frecuencia de 100 Hz (el doble de la frecuencia de red). La amplitud de la oscilación será proporcional a la intensidad luminosa de la bombilla. La forma de onda no será perfectamente sinusoidal debido a la inercia térmica del filamento, que suaviza parcialmente las variaciones.



Para bombillas fluorescentes compactas y LED, el resultado dependerá de la calidad de su electrónica de alimentación. Las bombillas LED de buena calidad pueden tener una frecuencia de conmutación de 20–40 kHz, invisible para el sensor a 1000 muestras/s, por lo que darán una señal casi continua. Las de baja calidad pueden mostrar un parpadeo notable a 100 Hz.

Medid el tiempo entre 11 picos consecutivos en la gráfica de Phyphox y dividid por 10 para obtener el período T. Calculad la frecuencia f = 1/T. Comparad con el valor esperado de 100 Hz. Calculad el error relativo. Observad que la frecuencia obtenida es el doble de la frecuencia de la red eléctrica, y explicad por qué sucede esto a partir de la relación $P \propto V^2$.

Discutid las fuentes de error: el ruido del ADC del ESP32, las interferencias de la antena Bluetooth, la limitación de 1000 muestras/s (que permite ver señales de hasta 500 Hz según el teorema de Nyquist), y la posible distorsión armónica introducida por la electrónica de la bombilla.

Ampliación

Utilizad un montaje adaptado para medir la velocidad de un objeto que pasa frente al sensor, colocando el fototransistor enfrentado a una fuente de luz fija (un puntero laser, por ejemplo) y haced pasar un objeto (una bolita, una regla) entre ambos; el tiempo de oscurecimiento permitirá calcular la velocidad.

También se pueden sincronizar dos fototransistores para medir tiempos de vuelo o aceleraciones, como hemos hecho en los experimentos correspondientes.

Código

Este experimento no se define en Phyphox a partir del código de abajo cargado en la placa ESP32 sino que debe subirse directamente a la aplicación haciendo clic en el botón más y en Añadir experimento desde el código QR que aparece a continuación.



Una vez cargado el experimento se puede guardar como un experimento más de la misma aplicación para no tener que acudir cada vez al QR.

El código que viene a continuación sí que se ha de cargar en la placa microcontroladora.

El sensor TEMT6000 no necesita ninguna librería para funcionar pero el Bluetooth sí, pero las librerías ya están instaladas por defecto cuando se instalan las placas ESP32 de Espressif en el IDE de Arduino.

Los datos solo se envían por Bluetooth para evitar retardos, no por puerto serie, por lo que para visualizarlos se ha de usar Phyphox con el experimento descargado.

```
/*
Proyecto para medir variaciones rápidas de luminosidad producidas
por bombillas conectadas a la red eléctrica de 50 Hz.

Se necesita un fototransistor TEMT6000 y una placa ESP32 con Bluetooth.

Conexiones:
GND → GND
VCC → 3,3V o 5V
OUT → pin 34 (ADC del ESP32)

El código usa los dos núcleos del ESP32:
Núcleo 1 (prioridad 1): adquisición de datos con sincronización
Núcleo 0 (prioridad 2): transmisión BLE por lotes de 10 datos

Los datos solo se envían por Bluetooth, no por puerto serie.
Para visualizarlos, usar Phyphox con el experimento descargable.
*/

#include <Arduino.h>
#include <BLEDevice.h>
#include <BLEServer.h>
#include <BLEUtils.h>
#include <BLE2902.h>

#define SERVICE_UUID        "19B10000-E8F2-537E-4F6C-D104768A1214"
#define CHARACTERISTIC_UUID "19B10001-E8F2-537E-4F6C-D104768A1214"

BLECharacteristic *pCharacteristic;
bool deviceConnected = false;

const int ldrPin = 34;      // Pin analógico del fototransistor

// Lote de 10 datos para evitar saturar Bluetooth
#define BATCH_SIZE 10

struct SensorBatch {
  float tiempo[BATCH_SIZE];
  float valor[BATCH_SIZE];
};
```

```
QueueHandle_t dataQueue;

// Callbacks de conexión BLE
class MyServerCallbacks: public
BLEServerCallbacks {
   void onConnect(BLEServer* pServer) {
deviceConnected = true; }
   void onDisconnect(BLEServer* pServer) {
deviceConnected = false; }
};

// Tarea de adquisición en el núcleo 1
void adcTask(void *pvParameters) {
  SensorBatch batch;
  int index = 0;
  unsigned long nextSampleTime = micros(); //
Sincronización estricta

  for(;;) {
    batch.tiempo[index] = micros() / 1000000.0;
    batch.valor[index] = (float)analogRead(ldrPin);
    index++;

    // Cuando el lote está completo, se envía a la
cola
    if (index >= BATCH_SIZE) {
      xQueueSend(dataQueue, &batch, 0);
      index = 0;
    }

    // Siguiente muestra en exactamente 1000 µs →
1000 muestras/s
    nextSampleTime += 1000;
    long waitTime = nextSampleTime - micros();
    if (waitTime > 0) {
      delayMicroseconds(waitTime);
    } else {
      nextSampleTime = micros(); // Resincronizar si
hubo retraso
    }
  }
}

// Tarea de transmisión BLE en el núcleo 0
void bleTask(void *pvParameters) {
  SensorBatch batch;
  for(;;) {
    if (xQueueReceive(dataQueue, &batch,
portMAX_DELAY)) {
      if (deviceConnected) {
        // Enviar 80 bytes (10 flotantes × 2 canales × 4
bytes) por BLE
        pCharacteristic->setValue((uint8_t*)&batch,
sizeof(SensorBatch));
        pCharacteristic->notify();
      }
    }
  }
}

void setup() {
  Serial.begin(115200);

  // Cola para 50 lotes de 10 datos
  dataQueue = xQueueCreate(50,
sizeof(SensorBatch));

  // Configuración BLE
  BLEDevice::init("ESP32_Luz");
  BLEDevice::setMTU(256);

  BLEServer *pServer = BLEDevice::createServer();
  pServer->setCallbacks(new MyServerCallbacks());

  BLEService *pService = pServer-
>createService(SERVICE_UUID);
  pCharacteristic = pService->createCharacteristic(
                      CHARACTERISTIC_UUID,

BLECharacteristic::PROPERTY_NOTIFY
                    );
  pCharacteristic->addDescriptor(new BLE2902());
  pService->start();

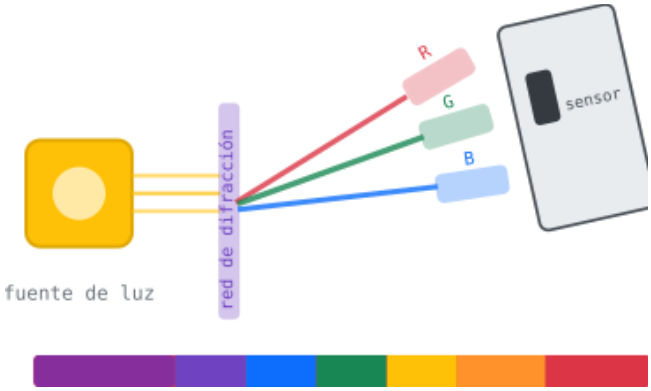
  BLEAdvertising *pAdvertising =
BLEDevice::getAdvertising();
  pAdvertising->addServiceUUID(SERVICE_UUID);
  pServer->getAdvertising()->start();

  // Tarea de adquisición en núcleo 1, prioridad 1
  xTaskCreatePinnedToCore(adcTask, "Lectura",
4096, NULL, 1, NULL, 1);
  // Tarea de Bluetooth en núcleo 0, prioridad 2
(mayor)
  xTaskCreatePinnedToCore(bleTask, "Envio",
4096, NULL, 2, NULL, 0);
}

void loop() {
  vTaskDelete(NULL); // loop() no se usa
}
```

Espectroscopia con el sensor de color TCS34725

Vamos a simular un espectroscopio utilizando el sensor de color TCS34725 para analizar la composición espectral de diferentes fuentes de luz (LED blanco, incandescencia, fluorescente, luz solar) y, con ayuda de una red de difracción, vamos a determinar experimentalmente la frecuencia y longitud de onda de las bandas de color del espectro visible.



Este experimento es una modificación sustancial de los métodos tradicionales de análisis espectral cuantitativo que se pueden realizar con una red de difracción, la cámara de un teléfono móvil y la aplicación web SpectralWorkbench (Romero y otros, 2022).

Las posibilidades de este sensor para hacer estudios de la composición de la luz se pueden ampliar con los artículos de Luis Llamas (2026) y de CATEDU (2026).

Fundamento físico

Naturaleza de la luz

La luz visible es una onda electromagnética cuya frecuencia (f) y longitud de onda (λ) se relacionan mediante:

$$c = \lambda f$$

donde (c = 3,00·10^8 m/s) es la velocidad de la luz en el vacío. La luz blanca está compuesta por una mezcla de frecuencias que van desde el rojo $(f \approx 4{,}3 \times 10^{14}\ \text{Hz})$, $(\lambda \approx 700\ \text{nm})$ hasta el violeta $(f \approx 7{,}5 \times 10^{14}\ \text{Hz})$, $(\lambda \approx 400\ \text{nm})$.

Descomposición espectral mediante red de difracción

Al hacer pasar un haz de luz a través de una red de difracción (conjunto de rendijas paralelas equiespaciadas), cada longitud de onda se desvía un ángulo distinto según la ecuación de la red:

$$d \cdot sen\theta = n\lambda$$

donde (d) es la distancia entre rendijas (por ejemplo, para 500 líneas/mm, $(d = 2{,}0 \times 10^{-6}\ \text{m})$), (θ) el ángulo de desviación, (λ) la longitud de onda y (n) el orden de difracción $(n = 0, \pm1, \pm2, \ldots)$. Esto permite separar físicamente los distintos colores del espectro.

Funcionamiento del sensor TCS34725

Como se explica en el capítulo dedicado a los sensores, el TCS34725 (AMS) es un sensor de color digital que contiene un array de 3×4 fotodiodos con filtros espectrales de paso de banda para rojo, verde, azul y un canal claro (sin filtro). Cada fotodiodo convierte la luz incidente en una corriente proporcional a la intensidad recibida en su banda espectral.

Como vemos el TCS34725 no es un espectrómetro real, sino un sensor tri-cromático con canales de luz roja, verde, azul (RGB) y transparente (Clear). No mide un espectro continuo de longitud de onda, pero lo podemos usar para espectroscopia básica o colorimetría combinándolo con una fuente de luz y un elemento dispersivo como una red de difracción o un prisma.

Materiales y montaje experimental

- Placa Arduino UNO o ESP32 (Wemos D1 R32) y el sensor de color TCS34725 (GY-33 o similar)
- Protoboard y cables Dupont
- Linterna led y otras fuentes de luz: incandescente, fluorescente, vela, etc.
- Red de difracción: la que se disponga: 500, 600, 1000 líneas/mm, o alternativamente una casera obtenida de un CD o DVD (Ramírez, 2009) (opcional pero recomendada)
- Cartulina negra para construir un colimador y una pantalla o cartulina blanca para proyectar el espectro

- Regla y transportador de ángulos (si se usa red de difracción)

La comunicación con el TCS34725 se realiza exclusivamente por I²C, lo que reduce el cableado a cuatro hilos (más otro opcional para el LED iluminador). Conectad según la siguiente tabla:

Módulo TCS34725 (GY-33)	Arduino UNO	ESP32 (Wemos D1 R32)
VIN	5 V	5 V o 3,3 V
GND	GND	GND
SCL	A5 (pin 19)	D22 (GPIO 22)
SDA	A4 (pin 18)	D21 (GPIO 21)
LED (opcional)	Apagado a GND	Apagado a GND

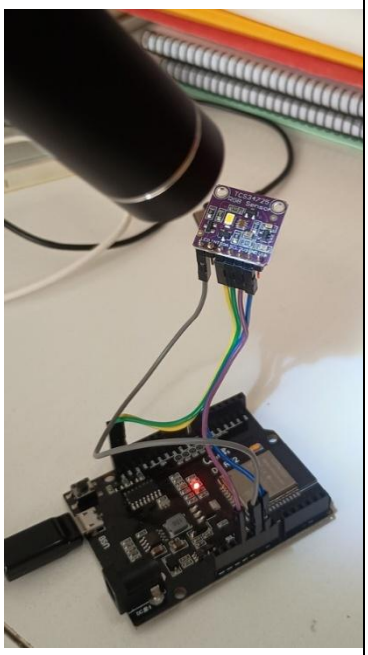
El pin led de la placa tiene una resistencia pull-up interna conectada al voltaje de alimentación. Esto significa que si se deja el pin desconectado, el circuito interno recibe automáticamente una señal ALTA (HIGH) y enciende el led por defecto para garantizar que haya iluminación al leer los colores, lo que va muy bien para analizar los colores de los objetos, lo que no es nuestro caso ahora.

Para mantener apagado el led siempre se conecta el pin led del sensor directamente a un pin GND (tierra) del microcontrolador. Al forzar la señal a BAJO (LOW), el LED se apagará físicamente.

Empezad por construid un espectroscopio con cartulina negra (o tubo de plástico o de cartón del papel de cocina) haciendo un cilindro de ~10 cm de largo con una rendija estrecha (~1 mm) en un extremo y la red de difracción en el otro extremo



Colocad la fuente de luz frente a la rendija del espectroscopio y proyectad el espectro sobre una pantalla blanca situada a una distancia conocida (por ejemplo, 20 cm).



Desplazad el sensor TCS34725 para que mida la luz de cada banda de color proyectada sobre la pantalla. El sensor ha de estar de cara a la luz incidente y las luces reflejadas en la pantalla sirven de referencia.

Para el análisis directo de fuentes de luz (sin red de difracción), simplemente apuntad el sensor hacia la fuente, asegurando que no haya luz parásita que contamine la medida.

Procedimiento para medir las bandas espectrales de la red de difracción

Una vez montado el sistema óptico anterior (colimador + red de difracción + pantalla) y proyectado el espectro del LED blanco sobre la pantalla, medid la distancia desde la red a la pantalla (L).

Ahora, para cada banda de color (rojo, naranja, amarillo, verde, azul, violeta) medid la distancia horizontal (x) desde el máximo central (orden 0) hasta el centro de la banda en la pantalla y calculad el ángulo de desviación: $\theta = \arctan(x/L)$.

Con el ángulo θ se puede calcular aproximadamente la longitud de onda de cada color:

$\lambda = d \cdot sen\theta / n$ (con n = 1 para primer orden).

Conectad la placa y cargad el código proporcionado más abajo. Si se coloca el sensor TCS34725 sobre la banda se registrarán los valores R, G, B, C. Repetid todo el proceso para el otro lado del espectro (orden n = -1) y promediad.

Para el análisis directo de fuentes de luz (sin red de difracción) situad el sensor TCS34725 apuntando directamente a cada fuente de luz (LED blanco, incandescente, fluorescente, luz solar, vela) y registrad los valores de los cuatro canales (R, G, B, C) para cada fuente.

Solo queda normalizar los valores respecto al canal claro: r = R/C, g = G/C y b = B/C.

Análisis de datos

Longitud de onda de las bandas espectrales
Completad la tabla siguiente usando los resultados obtenidos:

$$f = \frac{c}{\lambda}$$

Banda de color	(x) (m)	(θ) (grados)	(λ) (nm)	(f) (×10¹⁴ Hz)
Rojo				
Naranja				
Amarillo				
Verde				
Azul				
Violeta				

Composición espectral de fuentes de luz (sin red de difracción)
A partir de los valores R, G y B suministrados por el sensor calculad las fracciones relativas para cada fuente:

$$\%R = \frac{R}{R+G+B} \times 100, \quad \%G = \frac{G}{R+G+B} \times 100,$$
$$\%B = \frac{B}{R+G+B} \times 100$$

Fuente de luz	% Rojo	% Verde	% Azul
LED blanco			
Incandescente			
Fluorescente			
Luz solar			
Vela			

La luz incandescente debería mostrar un predominio del rojo (temperatura de color ~2700 K). La fluorescente presenta picos característicos en las líneas del mercurio (verde ~546 nm, azul ~436 nm). La luz solar tiene una distribución más equilibrada, aunque con mayor intensidad en el verde (pico de sensibilidad del ojo humano).

Resultados esperados
- Los valores de R, G, B obtenidos con el TCS34725 (16 bits, 0–65535) serán directamente proporcionales a la intensidad de cada componente espectral.
- Para el LED blanco, las tres componentes aparecerán relativamente equilibradas, con un ligero pico en azul si es un LED de luz fría (~6500 K) o en rojo si es cálido (~3000 K).
- La red de difracción separará las bandas con una dispersión angular apreciable (por ejemplo, para (L = 30 cm), la separación entre el rojo y el violeta será del orden de 5–10 cm en la pantalla).
- Las longitudes de onda calculadas mediante $(d \cdot sen\theta = \lambda)$ deberían coincidir con los valores de referencia dentro de un margen del 5–10 %, limitado principalmente por la precisión de la medida angular y la anchura de la rendija.

Ampliación
Espectro de absorción: Interponed entre la fuente y el sensor diferentes disoluciones coloreadas (permanganato, sulfato de cobre, clorofila) y observad qué componentes del espectro se atenúan.

Ley de Beer-Lambert: Preparad diluciones seriadas de un colorante alimentario y medid la absorbancia $(A = -\log_{10}(I/I_0))$ en función de la concentración para una banda específica.

Temperatura de color: Usad la relación entre los canales R y B para estimar la temperatura de color de distintas fuentes: $(T \propto (R/B)^k)$.

Sensor alternativo: Repetid el experimento con un APDS-9960 (Arduino Nano 33 BLE Sense) para comparar la precisión y linealidad.

Código

```
/*
Espectroscopia con sensor de color TCS34725
Lee los cuatro canales (R, G, B, C) del TCS34725 mediante I2C
y los envía por puerto serie en formato CSV.

Conexiones:
TCS34725 (GY-33)  ->  Arduino UNO/Nano  ->  ESP32 (Wemos D1 R32)
VIN       ->  3,3V          ->  3,3V
GND       ->  GND           ->  GND
SCL       ->  A5 (pin 19)   ->  D22 (GPIO 22)
SDA       ->  A4 (pin 18)   ->  D21 (GPIO 21)
LED……>  GND (apagado

Requiere la librería Adafruit_TCS34725.
*/

#include <Wire.h>  // Libreria para comunicacion I2C (incluida en el IDE de Arduino)
```

```
#include <Adafruit_TCS34725.h>  // Libreria de Adafruit para el sensor de color TCS34725

/*
Creamos una instancia del sensor con los parámetros por defecto:
- Tiempo de integración: 50 ms
- Ganancia: 16x
Otros valores posibles:
Integración: 2.4ms, 24ms, 50ms, 101ms, 154ms, 700ms
Ganancia: 1x, 4x, 16x, 60x
Para luz débil, aumentar el tiempo de integracion o la ganancia, i viceversa.
*/

Adafruit_TCS34725 tcs = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_50MS, TCS34725_GAIN_16X);

void setup() {
Serial.begin(9600);  // Iniciamos la comunicacion serie a 9600 baudios para enviar datos al ordenador

delay(500);

// Verificamos que el sensor TCS34725 está conectado y responde en la direccion I2C 0x29
if (!tcs.begin()) {
// Si el sensor no se detecta, mostramos un mensaje de error y detenemos la ejecucion
Serial.println("ERROR: No se encontro el sensor TCS34725. Verifica las conexiones I2C.");
while (1) { delay(10); }  // Bucle infinito: el programa se detiene aqui
}

// Si el sensor se ha inicializado correctamente
Serial.println("TCS34725 iniciado correctamente.");
// Enviamos la cabecera de la tabla CSV con los nombres de las columnas
Serial.println("tiempo_ms;R;G;B;C");
}

void loop() {
// Variables donde se almacenaran los valores leidos del sensor
// Los valores son enteros sin signo de 16 bits (0 - 65535)
uint16_t r, g, b, c;

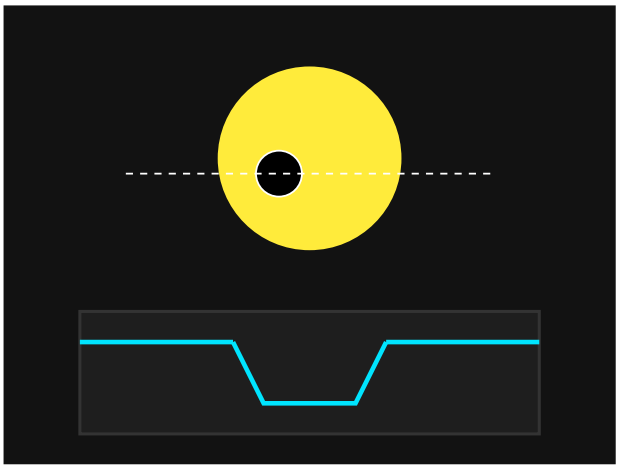
/*
Leemos los cuatro canales del sensor en una sola llamada.
- r: intensidad del canal ROJO
- g: intensidad del canal VERDE
- b: intensidad del canal AZUL
- c: intensidad del canal CLARO (sin filtro, luz total)
*/
tcs.getRawData(&r, &g, &b, &c);

/*
Enviamos los datos por puerto serie en formato CSV (punto y coma como separador).
*/
   Serial.print(millis());  // Tiempo transcurrido desde el arranque (ms)
   Serial.print(";");
   Serial.print(r);  // Canal rojo (0-65535)
   Serial.print(";");
   Serial.print(g);  // Canal verde (0-65535)
   Serial.print(";");
   Serial.print(b);  // Canal azul (0-65535)
   Serial.print(";");
   Serial.println(c);  // Canal claro (0-65535) - ultimo campo con salto de linea

// Esperamos 500 ms entre mediciones (2 muestras por segundo)
delay(500);
}
```

Simulación de tránsito planetario

Es muy interesante y sencillo simular el tránsito de un planeta frente a su estrella (como en los exoplanetas) midiendo la disminución de la intensidad luminosa al pasar un objeto oscuro frente a una fuente de luz.



Esta misma simulación se puede realizar utilizando el sensor de luz de los teléfonos móviles (Monteiro, 2017).

Fundamento físico

El método de tránsito es una de las técnicas más fructíferas para la detección de exoplanetas. Cuando un planeta pasa por delante de su estrella anfitriona (tránsito), bloquea parcialmente la luz estelar, produciendo una disminución periódica y característica del brillo aparente. La fracción de luz bloqueada es proporcional al cociente de áreas:

$$\frac{\Delta F}{F} = \left(\frac{R_p}{R_s}\right)^2$$

donde $(\Delta F/F)$ es la profundidad del tránsito (caída relativa de flujo), (R_p) el radio del planeta y (R_s) el radio de la estrella. Por ejemplo, Júpiter frente al Sol produce $(\Delta F/F \approx 1\%)$ y la Tierra apenas $(\approx 0,01\%)$.

En esta simulación de laboratorio, un LED blanco o una bombilla incandescente actúa como estrella y una esfera oscura como planeta. El sensor BH1750 mide la iluminancia a medida que la esfera cruza la trayectoria del haz, registrando la curva de luz característica del tránsito.

Materiales y montaje experimental

- ESP32 y sensor BH1750 o TSL2591
- LED blanco o bombilla incandescente (la "estrella")
- Esfera oscura (bola de plastilina, canica pintada) como "planeta" y cinta métrica.

Colocad el foco de luz en un extremo y el sensor en otro, alineados. Deslizad la esfera entre ellos simulando el tránsito.

Procedimiento

Se comienza midiendo la luz F sin obstáculo (luz directa) y se desliza la esfera lentamente por delante del haz mientras Phyphox registra y grafica la caída de intensidad.

Se puede repetir la experiencia con esferas de diferente diámetro.

Análisis

Calculad $(\Delta F/F)$ para cada esfera y comparad con $\left(\left(R_p/R_s\right)^2\right)$. Discutid también las limitaciones de la simulación (difracción, dispersión).

Código

Se ha de instalar en el IDE de Arduino las librería Phyphox BLE y BH1750 de Chistopher Laws.

```
/*
Código para simular un tránsito planetario con
ESP32 / Nano 33 BLE.
Los datos se envían por puerto serie al PC y por
Bluetooth a Phyphox.
Se genera automáticamente una gráfica de Tiempo
vs Iluminancia.
*/

#include <Wire.h>         // Librería para
comunicación I2C
#include <BH1750.h>       // Librería del sensor
(Autor: Christopher Laws)
#include <phyphoxBle.h>   // Librería oficial de
Phyphox
```

```
BH1750 sensor;            // Objeto para el sensor BH1750

void setup() {
  Serial.begin(9600);
  Wire.begin();
  sensor.begin();

  // 1. Iniciamos el Bluetooth con el nombre del dispositivo
  PhyphoxBLE::start("Transito_Planetario");

  // 2. Creamos y configuramos el experimento para Phyphox
  PhyphoxBleExperiment experimento;
  experimento.setTitle("Tránsito Planetario");
  experimento.setCategory("Mis Experimentos");
  experimento.setDescription("Curva de luz de un tránsito planetario.");

  // Creamos una "vista" o pestaña dentro de la app
  PhyphoxBleConfigView vista;
  vista.setLabel("Datos en directo");

  // Configuramos la gráfica
  PhyphoxBleConfigGraph grafica;
  grafica.setLabel("Curva de luz");         // Título de la gráfica
  grafica.setUnitX("s");                    // Unidad del eje X
  grafica.setLabelX("Tiempo transcurrido");  // Etiqueta del eje X
  grafica.setUnitY("lux");                  // Unidad del eje Y
  grafica.setLabelY("Iluminancia");         // Etiqueta del eje Y

  // Asignamos qué dato va a qué eje (Canal 1 al X, Canal 2 al Y)
  grafica.setChannel(1, 2);

  // Añadimos la gráfica a la vista, y la vista al experimento
  vista.addElement(grafica);
  experimento.addView(vista);

  // Cargamos el experimento en el Bluetooth
  PhyphoxBLE.addExperiment(experimento);
}

void loop() {
  // Calculamos el tiempo en segundos (millis() da milisegundos)
  float tiempo = millis() / 1000.0;

  // Leemos la luz
  float luz = sensor.readLightLevel();

  // --- ENVÍO AL PC POR CABLE ---
  // Lo enviamos en formato "Tiempo,Luz" para que el Serial Plotter pueda dibujarlo
  Serial.print(tiempo);
  Serial.print(",");
  Serial.println(luz);

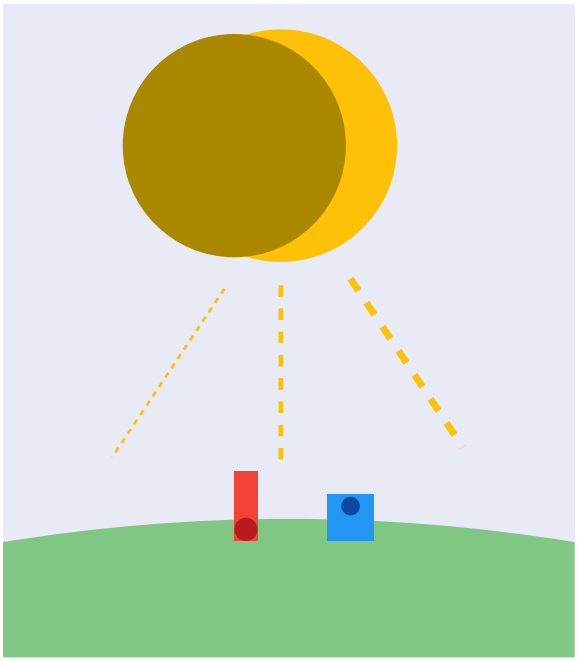
  // --- ENVÍO AL MÓVIL POR BLUETOOTH ---
  // Enviamos primero el canal 1 (tiempo) y luego el canal 2 (luz)
  PhyphoxBLE.write(tiempo, luz);

  delay(50); // 20 muestras por segundo
}
```

Medida de las variables ambientales durante un eclipse solar

Registrar la evolución temporal de las variables ambientales —temperatura, humedad relativa, presión atmosférica e iluminancia— durante un eclipse solar, y analizar cómo la ocultación progresiva del disco solar afecta a cada una de ellas.

El experimento está diseñado (Ramírez, 2026c) para funcionar de forma autónoma en exteriores, almacenando los datos en una tarjeta microSD y transmitiéndolos simultáneamente por Bluetooth a la aplicación Phyphox para visualización en tiempo real.



Fundamento físico

El eclipse solar

Un eclipse solar ocurre cuando la Luna se interpone entre la Tierra y el Sol, proyectando una sombra sobre la superficie terrestre. Durante la fase de totalidad, la radiación solar directa se reduce drásticamente, lo que provoca:

- Caída de la iluminancia (Ev): desde valores típicos de ~100 000 lx (sol directo) hasta ~1 lx o menos durante la totalidad.
  La iluminancia se define como el flujo luminoso por unidad de área:

$$E_v = \frac{\Phi}{A}$$

  Unidad: lux (lx = lm/m²). La radiación solar directa en un día despejado alcanza ~110 000 lx. Durante un eclipse parcial, la fracción de área solar oculta determina la iluminancia residual.

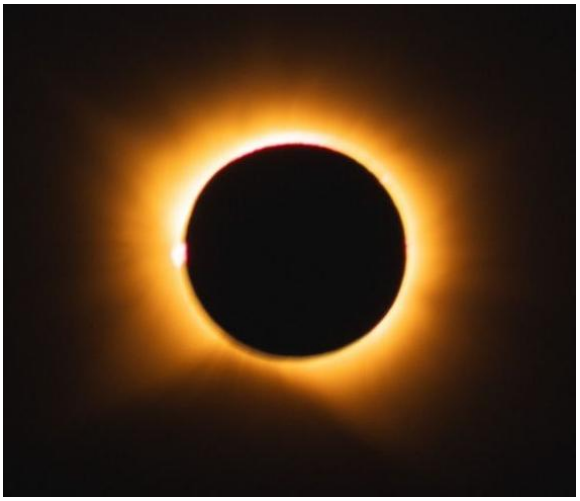
- Descenso de la temperatura ambiente: la interrupción de la radiación infrarroja solar reduce la temperatura del aire en varios grados (típicamente 2–6 °C).

- Variación de la humedad relativa: al descender la temperatura, la humedad relativa aumenta (la presión de vapor saturado disminuye con (T)).
  La humedad relativa (HR) depende de la temperatura según la ecuación de Clausius-Clapeyron aproximada:

$$HR = \frac{e}{e_s(T)} \times 100\%$$

  donde ($e_s(T)$) es la presión de vapor saturante, que disminuye al bajar (T). Por tanto, si el contenido absoluto de vapor de agua se mantiene constante, un descenso de (T) provoca un aumento de (HR).

- Mínima variación de la presión atmosférica. La presión atmosférica no tiene porqué sufrir variaciones durante un eclipse.

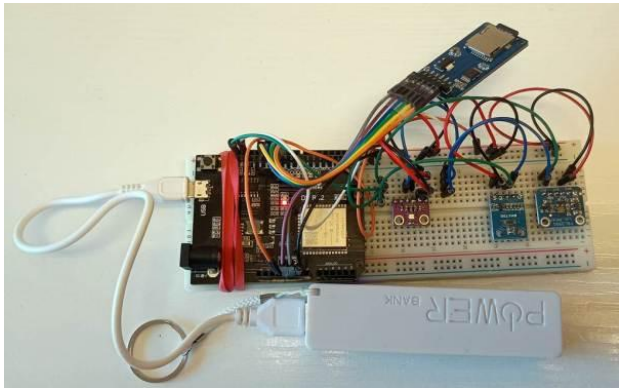


Fotografía de Xavier Franch del eclipse solar total de 2026

Materiales y montaje experimental

En la tabla siguiente se listan los materiales necesarios:

Componente	Cantidad	Notas
Placa ESP32 (Wemos D1 R32 o similar)	1	También sirve Arduino Nano 33 BLE Sense rev2
Sensor BME280 (T, H, P)	1	Comunicación I²C, dirección 0x76 o 0x77
Sensor BH1750 (lux)	1	I²C, rango 1–65535 lx
Sensor TSL2591 (lux)	1 (op.)	I²C, rango 188 µlx–88000 lx
Adaptador microSD	1	SPI, formato FAT32
Tarjeta microSD	1	Capacidad ≥ 512 MB
Protoboard	1	Para conexiones sin soldadura
Cables dupont (MM, MF, FF)	~20	Varias longitudes
Power bank	1	Para alimentación autónoma en exterior
Cables USB-OTG	1 (op.)	Para conectar a móvil/tableta



Conexiones específicas

BME280:

Pin BME280	Conexión
VCC/VIN	3,3 V o 5 V
GND	GND
SCL	GPIO22 (ESP32) / A5 (Arduino UNO)
SDA	GPIO21 (ESP32) / A4 (Arduino UNO)

Si se deja SDO desconectado, la dirección I²C es 0x77. Conectando SDO a GND la dirección cambia a 0x76.

BH1750:

Pin BH1750	Conexión
VIN	3,3 V o 5 V
GND	GND
SCL	GPIO22
SDA	GPIO21
ADDR	Sin conectar

TSL2591:

Pin TSL2591	Conexión
VIN	3,3 V o 5 V
GND	GND
3VO	Salida 3,3 V (hasta 100 mA)
INT	Opcional — interrupción por umbral
SCL	GPIO22
SDA	GPIO21

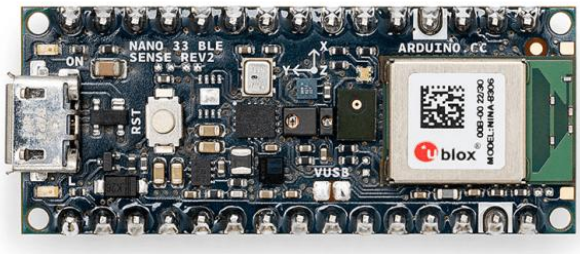
Adaptador microSD (SPI en ESP32):

Pin SD	Conexión
VCC	5 V (el regulador interno lo baja a 3,3 V)
GND	GND
MISO	GPIO19
MOSI	GPIO23
SCK	GPIO18
CS	GPIO5

Sensores integrados del **Arduino Nano 33 BLE Sense**:

V1: HTS221 (T + H), LPS22HB (P), APDS9960 (luz)

V2: HS300x (T + H), LPS22HB (P), APDS9960 (luz)



El Arduino Nano 33 BLE Sense no necesita sensores externos, ya que los integra todos (APDS9960 para luz, LPS22HB para presión, HTS221 o HS300x para T + H). Basta con conectar la placa a la alimentación.

Procedimiento

Días previos al eclipse

1. Probad del sistema: monta el dispositivo y cárgalo con el sketch al menos una semana antes del eclipse.
2. Verificación de la tarjeta SD: aseguraros de que esté formateada en FAT32 (no exFAT ni NTFS).
3. Calibración relativa: colocad los sensores de luz bajo la misma fuente y registra las diferencias entre BH1750 y TSL2591 (el BH1750 suele dar valores un 20–30 % más altos).
4. Prueba de autonomía: dejad el sistema funcionando durante 2–3 horas y verifica que los datos se guardan correctamente en la SD y que el Bluetooth no se desconecta.

Durante el eclipse

1. Colocad el dispositivo en un lugar abierto, sin obstáculos que proyecten sombra sobre los sensores de luz.
2. Conectad la alimentación (power bank).
3. Opcional: abrid Phyphox en el móvil, añadid el experimento por Bluetooth y comienzad la grabación.
4. Dejad que el sistema registre datos automáticamente durante todo el evento.
5. Tomad notas manuales de los momentos clave (inicio del eclipse, totalidad, final).



El dispositivo el día del eclipse solar de 2026

Después del eclipse

1. Apagad el sistema y extraed la tarjeta microSD.
2. Copiad el archivo `eclipse.csv` al ordenador.
3. Abrid los datos en una hoja de cálculo (sustituyendo puntos decimales por comas si el sistema usa coma decimal).
4. Generad gráficas: iluminancia vs tiempo, temperatura vs tiempo, HR vs tiempo.

Resultados esperados

Iluminancia durante un eclipse

Fase	Iluminancia (lx)	Comportamiento
Sol despejado	~100 000–110 000	Valor de referencia a mediodía
Inicio del eclipse parcial	100 000 → 50 000	Caída gradual
50 % de ocultación	~30 000–40 000	La luz se reduce más rápido de lo lineal por dispersión atmosférica
90 % de ocultación	~1 000–5 000	Caída pronunciada
Totalidad	0,1–10	Solo luz crepuscular + corona solar
Salida de la totalidad	1 000–100 000	Recuperación simétrica

Temperatura

Descenso térmico: 2–6 °C durante la totalidad, dependiendo de la estación, nubosidad y hora del día.

Hay un retardo respecto al mínimo de luz ya que la temperatura responde con inercia (~5–15 min de retraso). La recuperación es más lenta que la caída, ya que el suelo necesita reabsorber radiación.

Humedad relativa

Aumento entre 5 y 15 puntos porcentuales durante la totalidad, por el descenso de temperatura.

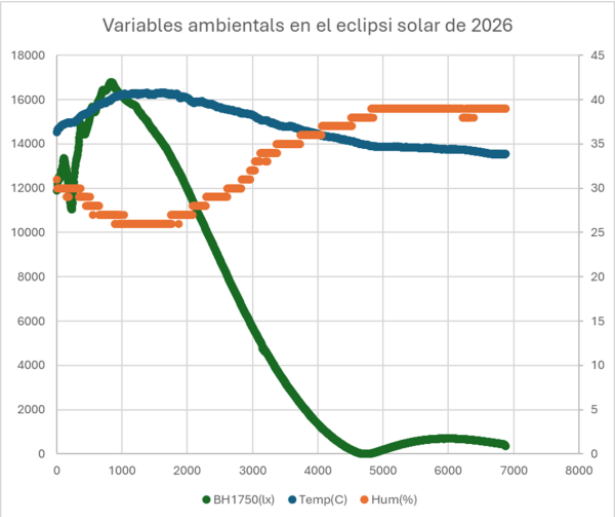
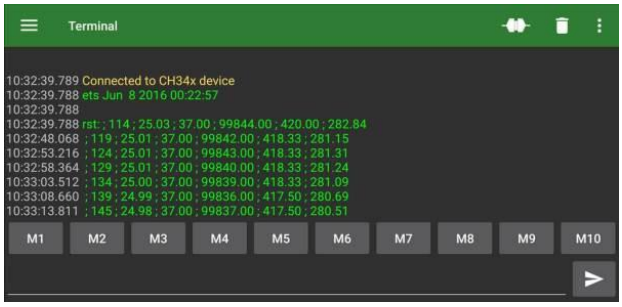
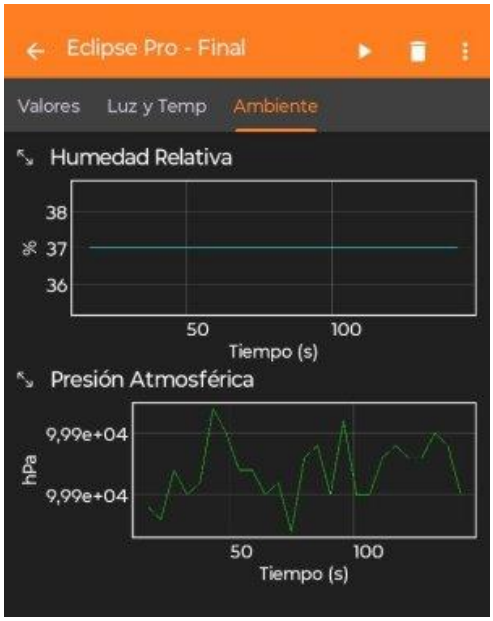


Gráfico con los valores de iluminancia (verde), temperatura (azul) y humedad (marrón) durante el eclipse solar de 2026

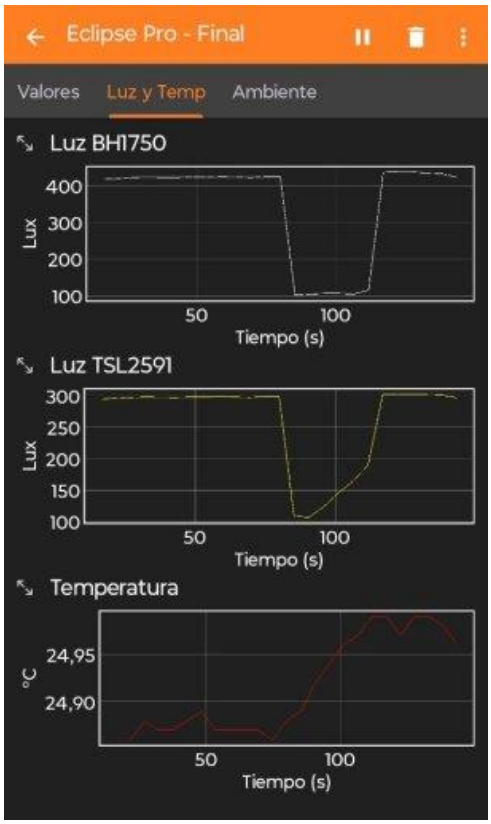
Presión atmosférica
Tendrá una variación mínima (< 10 Pa) atribuible al eclipse, menor que las fluctuaciones diarias que suelen ser mayores.

En las siguientes capturas de pantalla se ve cómo los teléfonos capturan los valores ambientales.









Análisis de los datos
1. Normalización temporal: sincronizad todos los datos respecto al instante de la totalidad (t = 0).

2. Ajuste de curvas: modelad la caída de iluminancia como función del área de ocultación del disco solar. ¿Por qué la iluminancia no cae linealmente con el porcentaje de ocultación del Sol?

3. Comparativa de sensores: representad simultáneamente las lecturas del BH1750 y el TSL2591. Calculad la diferencia porcentual media entre ambos.
¿Qué fracción de la caída de temperatura se debe a la reducción de radiación infrarroja y cuánta a la reducción de convección?

4. Correlación T vs HR: calculad el coeficiente de correlación de Pearson entre temperatura y humedad relativa. La relación debe ser inversa ($r \approx -0{,}7\ a\ -0{,}9$).

5. Inercia térmica: determinad el desfase temporal entre el mínimo de iluminancia y el mínimo de temperatura mediante correlación cruzada.

¿Es significativa la variación de presión atmosférica durante el eclipse o está dentro del ruido instrumental?
¿Cómo afecta la nubosidad parcial al perfil de iluminancia? ¿Y al perfil térmico?

Ampliación

Comparación con datos oficiales
Obtened los datos de iluminancia y temperatura de una estación meteorológica cercana (AEMET, Meteocat) y comparadlos con los registros obtenidos.

Estudio del FreeDSM
El proyecto FreeDSM (Free Dark Sky Meter) (https://github.com/gaia4sustainability/FreeDSM) de Gaia4Sustainabily propone un dispositivo de bajo coste para medir el brillo del cielo nocturno y contribuir al estudio de la contaminación lumínica.

Se puede reciclar el montaje de este experimento para estudiar también los eclipses solares.

Seguimiento remoto por IoT
Añadid un módulo WiFi para enviar los datos a una plataforma como ThingSpeak o Blynk durante el eclipse, permitiendo el seguimiento en tiempo real desde cualquier lugar.

Estudio de la corona solar
Hay que acceder a imágenes del Sol capturadas durante las fases del eclipse y correlacionar el área de ocultación visual con los datos de iluminancia.

Análisis espectral
Se puede añadir el TCS34725 como sensor de color para determinar cómo cambia la composición espectral de la luz durante las diferentes fases del eclipse (efecto de enrojecimiento en la totalidad).

Código

Phyphox
Hay que cargar el experimento en Phyphox y el skech en la palca ESP32. Para cargar el experimento en Phyphox solo hay que añadir en la aplicación un nuevo experimento mediante código QR y enfocar al siguiente.



Código para ESP32
Sketch a cargar en la placa

```
/*
Seguimiento de eclipse solar con ESP32
Sensores: BME280 + BH1750 + TSL2591 +
microSD

Librerías necesarias:
- Grove - Barometer Sensor BME280 (Seeed
Studio)
- BH1750 (Christopher Laws)
- Adafruit TSL2591 Library (Adafruit)
- BLEDevice, BLEUtils, BLE2902 (incluidas en
ESP32)
 */

#include <BLEDevice.h>
#include <BLEServer.h>
#include <BLEUtils.h>
#include <BLE2902.h>
#include <Wire.h>
#include "Seeed_BME280.h"
#include <BH1750.h>
#include <Adafruit_TSL2591.h>
#include <SPI.h>
#include <SD.h>

// UUIDs para BLE Phyphox
#define SERVICE_UUID      "c0c0c0c0-c0c0-c0c0-
c0c0-c0c0c0c0c0c0"
#define CHAR_UUID_DATOS    "c0c0c0c1-c0c0-
c0c0-c0c0-c0c0c0c0c0c1"
#define CHAR_UUID_TIEMPO   "c0c0c0c2-c0c0-
c0c0-c0c0-c0c0c0c0c0c2"

const int chipSelect = 5;  // CS para microSD en
ESP32
File dataFile;
```

```
BLECharacteristic *pCharDatos;
BLECharacteristic *pCharTiempo;
bool deviceConnected = false;

BME280 bme;
BH1750 bh1750;
Adafruit_TSL2591 tsl = Adafruit_TSL2591(2591);
bool hasBH1750 = false, hasTSL2591 = false;

float pressure, temperature, humidity;
float luxBH = -1, luxTSL = -1;

// Callbacks de conexión BLE
class MyServerCallbacks : public
BLEServerCallbacks {
   void onConnect(BLEServer* pServer) {
deviceConnected = true; }
   void onDisconnect(BLEServer* pServer) {
      deviceConnected = false;
      BLEDevice::startAdvertising();
   }
};

void setup() {
   Serial.begin(115200);

   // I2C en pines 21 (SDA) y 22 (SCL) del ESP32
   Wire.begin(21, 22);
   bme.init();

   hasBH1750 =
bh1750.begin(BH1750::CONTINUOUS_HIGH_RES
_MODE);
   hasTSL2591 = tsl.begin();

   // Inicializar tarjeta microSD
   Serial.print("Iniciando SD...");
   if (!SD.begin(chipSelect)) {
      Serial.println("Fallo en la SD o no presente.");
   } else {
      Serial.println("SD lista.");
      File file = SD.open("/eclipse.csv",
FILE_APPEND);
      if (file) {

file.println("Tiempo(ms),Temp(C),Hum(%),Pres(Pa),
BH1750(lx),TSL2591(lx)");
         file.close();
      }
   }

   // Configuración BLE
   BLEDevice::init("Eclipse_ESP32_Pro");
   BLEServer *pServer =
BLEDevice::createServer();
   pServer->setCallbacks(new
MyServerCallbacks());

   BLEService *pService = pServer-
>createService(SERVICE_UUID);

   pCharDatos = pService->createCharacteristic(
      CHAR_UUID_DATOS,
      BLECharacteristic::PROPERTY_NOTIFY
   );
   pCharDatos->addDescriptor(new BLE2902());

   pCharTiempo = pService->createCharacteristic(
      CHAR_UUID_TIEMPO,
      BLECharacteristic::PROPERTY_NOTIFY
   );
   pCharTiempo->addDescriptor(new BLE2902());

   pService->start();
   BLEDevice::getAdvertising()-
>addServiceUUID(SERVICE_UUID);
   BLEDevice::getAdvertising()->start();

   Serial.println("Esperando conexión BLE...");
}

void loop() {
   static unsigned long lastMillis = 0;
   unsigned long now = millis();

   if (now - lastMillis < 5000) return;  // Muestreo
cada 5 segundos
   lastMillis = now;

   // Lectura de sensores
   temperature = bme.getTemperature();
   humidity = bme.getHumidity();
   pressure = bme.getPressure();
   luxBH = hasBH1750 ? bh1750.readLightLevel() :
-1;

   if (hasTSL2591) {
      uint32_t lum = tsl.getFullLuminosity();
      uint16_t ir = lum >> 16;
      uint16_t full = lum & 0xFFFF;
      luxTSL = tsl.calculateLux(full, ir);
   } else {
      luxTSL = -1;
   }

   // Salida por puerto serie
```

```
  Serial.print(now); Serial.print(";");
  Serial.print(temperature); Serial.print(";");
  Serial.print(humidity); Serial.print(";");
  Serial.print(pressure); Serial.print(";");
  Serial.print(luxBH); Serial.print(";");
  Serial.println(luxTSL);

  // Guardar en microSD
  if (SD.begin(chipSelect)) {
    File file = SD.open("/eclipse.csv",
FILE_APPEND);
    if (file) {
      file.print(now); file.print(",");
      file.print(temperature); file.print(",");
      file.print(humidity); file.print(",");
      file.print(pressure); file.print(",");
      file.print(luxBH); file.print(",");
      file.println(luxTSL);
      file.close();
    }
  }

  // Enviar por BLE si hay conexión
  if (deviceConnected) {
    float sensores[5] = {temperature, humidity,
pressure, luxBH, luxTSL};
    pCharDatos->setValue((uint8_t*)sensores,
sizeof(sensores));
    pCharDatos->notify();

    uint32_t tiempo = now;
    pCharTiempo->setValue((uint8_t*)&tiempo,
sizeof(tiempo));
    pCharTiempo->notify();
  }
}
```

Sketch para Arduino Nano 33 BLE Sense rev2

El experimento se carga en Phyphox a través de bluetooth.




```
/*
Seguimiento de eclipse solar con Arduino Nano 33
BLE Sense rev2

Librerías necesarias:
- Arduino_LPS22HB (presión)
- Arduino_HTS221 o Arduino_HS300x (T + H
según versión)
- Arduino_APDS9960 (luz)
- Phyphox BLE (RWTH Aachen University)
 */

#include <Arduino_LPS22HB.h>
#include <Arduino_APDS9960.h>
#include <phyphoxBle.h>

// Seleccionar según versión de la placa:
// #include <Arduino_HTS221.h>   // V1
#include <Arduino_HS300x.h>      // V2

void setup() {
  Serial.begin(115200);
  while (!Serial);

  // Inicializar sensores integrados
  if (!BARO.begin()) {
    Serial.println("Fallo LPS22HB");
    while (1);
  }
  if (!APDS.begin()) {
    Serial.println("Fallo APDS9960");
    while (1);
  }
  // Según versión:
  if (!HTS.begin()) {  // o HS300x.begin()
    Serial.println("Fallo sensor T+H");
    while (1);
  }

  PhyphoxBLE::start("Eclipse_Nano33");
  Serial.println("Esperando conexión Phyphox...");
}

void loop() {
  static unsigned long lastMillis = 0;
  unsigned long now = millis();

  if (now - lastMillis < 5000) return;
  lastMillis = now;

  float temp = HTS.readTemperature();    // o
HS300x.readTemperature()
  float hum = HTS.readHumidity();
```

```
  float pres = BARO.readPressure();

  int r, g, b, a;
  float lux = 0.0;
  if (APDS.readAmbientLight(a)) {
    // Conversión aproximada APDS9960 → lux
    lux = a * 0.076;  // Factor empírico
  }

  Serial.print(now); Serial.print(";");
  Serial.print(temp); Serial.print(";");
  Serial.print(hum); Serial.print(";");
  Serial.print(pres); Serial.print(";");
  Serial.println(lux);

  // Enviar por BLE a Phyphox
  PhyphoxBLE::write(temp, hum, pres, lux);

  delay(5000);
}
```

Electromagnetismo

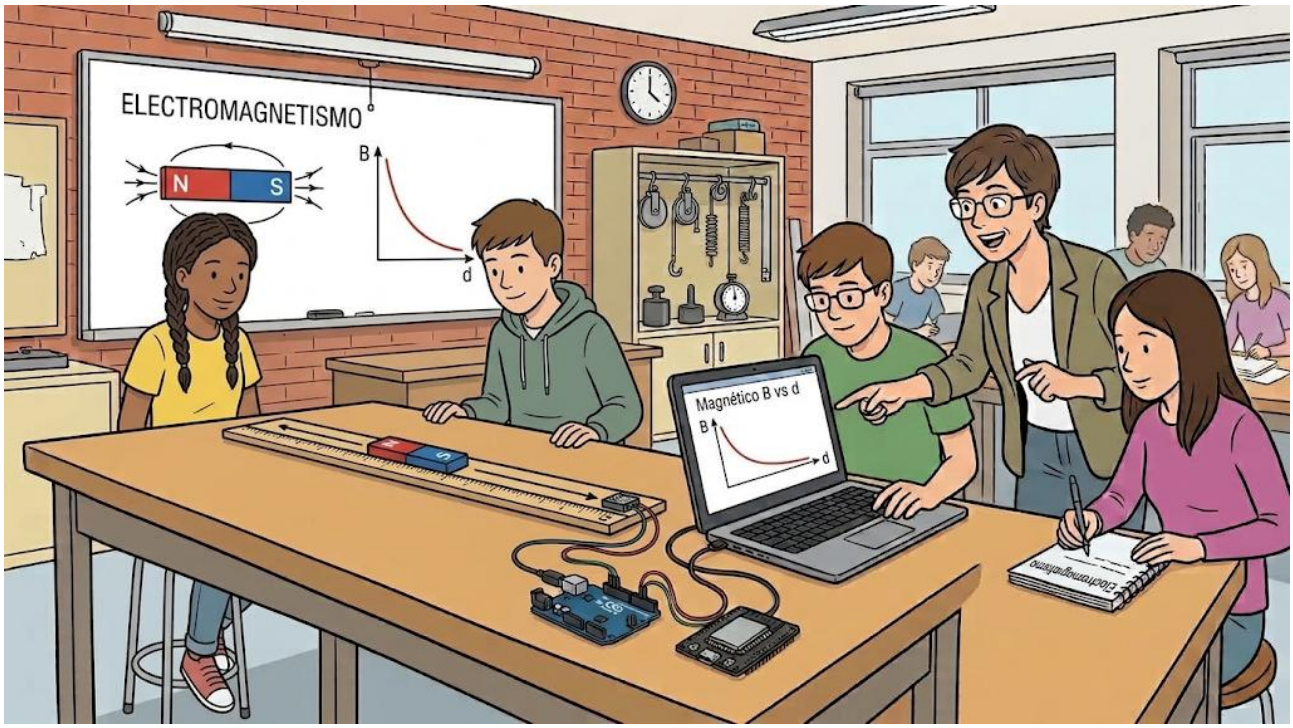
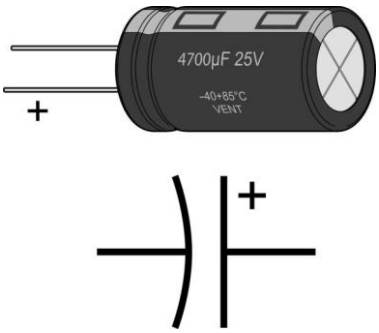


Ilustración realizada con Gemini. Esté pensada para adornar, aunque el alumnado puede entretenerse en buscar los errores evidentes que contiene.

Carga y descarga de un condensador

Se trata de registrar la curva de carga y descarga de un condensador electrolítico en función del tiempo, determinar experimentalmente la constante de tiempo RC del circuito y verificar que la constante de tiempo medida coincide con el producto $R \cdot C$.



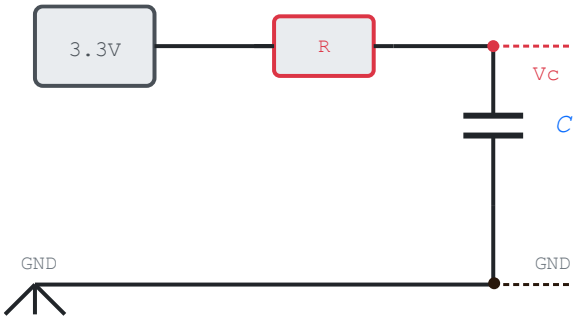
Esta es una de la prácticas más extendidas en los laboratorios de física. Por ejemplo, en la Universidad de Las Palmas de Gran Canaria proponen realizarla con un osciloscopio (Departamento de Informática y Sistemas, 2026) y en la de Almería con un voltímetro y un amperímetro (García, 2009), pero aquí lo haremos con ayuda de un Arduino (Ramírez, 2021).

Fundamento físico

Cuando un condensador se carga a través de una resistencia desde una fuente de tensión continua, la tensión en sus bornes sigue una ley exponencial. Durante la carga, la tensión $V_C(t)$ viene dada por:

$$V_C(t) = V_0\left(1 - e^{-t/RC}\right)$$

donde V_0 es la tensión de la fuente, R la resistencia en serie y C la capacidad del condensador.



Durante la descarga (cortocircuitando el condensador a través de una resistencia), la tensión decae según:

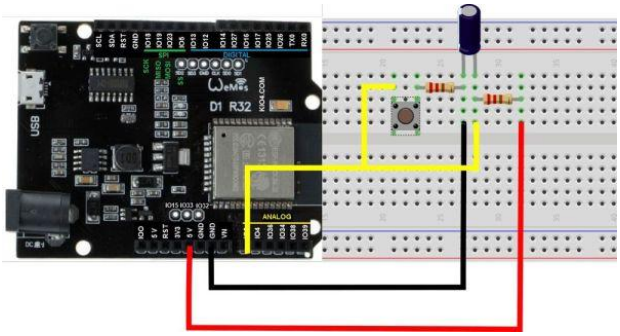
$$V_C(t) = V_0\, e^{-t/RC}$$

El producto $\tau = RC$ se denomina constante de tiempo del circuito y representa el tiempo necesario para que la tensión alcance el 63,2 % de su valor final durante la carga, o disminuya al 36,8 % durante la descarga. Midiendo la curva experimental y ajustándola a estas ecuaciones, podemos determinar τ y comprobar que coincide con el valor teórico $R \cdot C$.

Materiales y montaje experimental

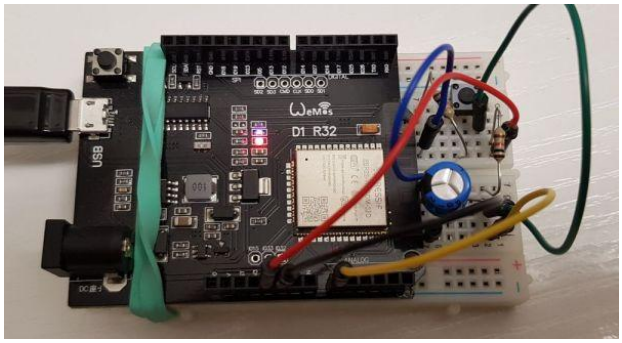
Necesitaréis una placa Arduino UNO o Wemos D1 R32 (ESP32), un condensador electrolítico de 220 µF (16 V), dos resistencias de 10 kΩ, un pulsador (interruptor momentáneo), una protoboard y cables Dupont.

Antes de montar el circuito, tened en cuenta que si el condensador ha sido usado con anterioridad puede estar cargado. Descargadlo cortocircuitando sus terminales antes de conectarlo a la placa Arduino. Las placas Arduino no aguantan más de 5 V en sus entradas, por lo que trabajaremos a 3,3 V.



Montad el circuito como se ve en la imagen de arriba: se conecta una resistencia de 10 kΩ en serie con el condensador de 220 µF. El conjunto se conecta entre 3,3 V (o 5 V) y GND. La pata negativa del condensador (la más corta) va a GND. El punto común entre la resistencia y el condensador se conecta a una entrada analógica: en Arduino UNO al pin A0, en ESP32 al pin GPIO2.

Colocad un pulsador en paralelo con el condensador, con una resistencia de 10 kΩ en serie, para descargarlo de forma controlada.



Procedimiento

Conectad la placa al ordenador y subid el código. Abrid el monitor serie del IDE de Arduino. Inicialmente el condensador está descargado, por lo que la tensión leída será próxima a 0 V.

Pulsad el botón de inicio en el monitor serie (o enviad cualquier carácter) para comenzar la carga. La placa empezará a registrar la tensión en el condensador a intervalos regulares. Observaréis cómo la tensión aumenta rápidamente al principio y luego se ralentiza, aproximándoos asintóticamente a la tensión de alimentación.

Para la descarga, pulsad el botón físico (pulsador) que cortocircuita el condensador a través de la resistencia, y observad cómo la tensión decae exponencialmente.

Resultados esperados



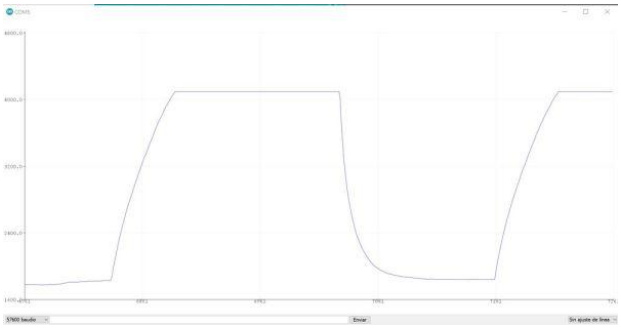
La constante de tiempo teórica del circuito es:

$$\tau = R \cdot C = 10\,000\,\Omega \times 220 \times 10^{-6}\,\mathrm{F} = 2{,}2\,\mathrm{s}$$

Durante la carga, al cabo de $\tau = 2{,}2$ s la tensión debería ser aproximadamente el 63,2 % de V_0:

$$V_C(\tau) = V_0 \cdot (1 - e^{-1}) \approx 0{,}632 \cdot V_0$$

Para V_0 = 3,3 V, esto corresponde a unos 2,09 V. Durante la descarga, al cabo de τ la tensión debería caer al 36,8 % de su valor inicial.

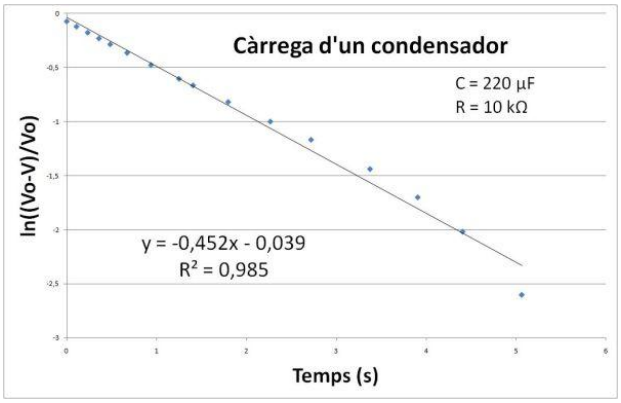


Carga i descarga de un condensador en una placa esp32 visualizada con el IDE de Arduino



Carga i descarga de un condensador en una placa esp32 visualizada en Phyphox

Para determinar experimentalmente la constante de tiempo, representad $\ln(1 - V_C/V_0)$ frente al tiempo durante la carga. Si el comportamiento es exponencial, los puntos se alinearán según una recta de pendiente $-1/\tau$.



Alternativamente, durante la descarga, representad $\ln(V_C/V_0)$ frente al tiempo, que también dará una recta de pendiente $-1/\tau$.

$-1/\tau$ = 0,452 → τ = 2,2 s

Calculad τ a partir de la pendiente y comparadlo con el valor teórico de 2,2 s. Determinad el error experimental y discutid las posibles causas: tolerancia de la resistencia y el condensador, resistencia interna del circuito, precisión del conversor ADC.

Ampliación

Repetid el experimento con diferentes valores de resistencia (por ejemplo, 4,7 kΩ y 22 kΩ) manteniendo el mismo condensador, y comprobad que τ varía proporcionalmente a R.

Probad también con diferentes capacidades (100 µF, 470 µF) para verificar la dependencia lineal con C.

Se puede conectar el circuito a una entrada analógica del ESP32 y visualizar la curva en tiempo real usando Phyphox para un análisis más preciso.

Código 1. Puerto serie

```
/*
Carga y descarga de un condensador electrolítico
Placa: Arduino UNO o ESP32
Para ver los datos por el puerto serie

Conexiones: Condensador en serie con R de 10k
Resistencia a GND, condensador a +3,3 V o +5 V
Punto común en A0 (Arduino UNO) o GPIO2 (ESP32)
Pulsador con R de 10k en paralelo al condensador

El voltaje está activado para un ESP32. Si es un Arduino UNO se ha de activar para su voltaje
*/

const int pinCondensador = 2;  // GPIO2 en ESP32 (A0 en Arduino UNO)
float voltaje;

void setup() {
  Serial.begin(115200);
  pinMode(pinCondensador, INPUT);

  Serial.println("Iniciando medida de carga/descarga de condensador");
  Serial.println("Envie cualquier caracter para comenzar la carga...");

  // Esperar a que el usuario inicie la medida
  while (!Serial.available()) {
    delay(100);
  }
  // Limpiar buffer
  while (Serial.available()) {
    Serial.read();
  }

  Serial.println("Cargando condensador...");
  Serial.println("t(ms);V(V)");
}

void loop() {
  unsigned long t = millis();

  // Leer tensión en el condensador
  // En ESP32: ADC de 12 bits (0-4095), Vref = 3,3 V
  // En Arduino UNO: ADC de 10 bits (0-1023), Vref = 5 V
  int raw = analogRead(pinCondensador);
  voltaje = (raw / 4095.0) * 3.3;
  // voltaje = (raw / 1023.0) * 5;

  // Mostrar datos
  Serial.print(t);
  Serial.print("; ");
  Serial.println(voltaje, 3);

  delay(50);  // 20 muestras por segundo
}
```

Código 2. Phyphox

```
//Carga y descarga de un condensador electrolítico
//en una placa con microprocesador esp32, que dispone de Bluetooth
//Se ha de seleccionar la placa ESP32 DEV Module o DOIT ESP32 DEVKIT V1
//Los datos se pueden visualizar en el monitor serie del IDE de Arduino
//y con el programa para ordenador RealTerm, https://sourceforge.net/projects/realterm/
//Los datos también se pueden visualizar en el teléfono móvil o la tableta
//con la aplicación Phyphox (https://phyphox.org/) a través de BT (BlueTooth)
//Este sketch és una adaptación del sketch Rangefinder, ejemplo incluído en la librería Phyphox BLE.

//Conexiones: Condensador en serie con resistencia de 10K
//Resistencia a GND, condensador a +3.3 o +5 V,
```

```
//común de la resistencia y el condensador a A0 (Arduino) o a 2 (esp32)
//en paralelo al condensador, una resistencia de 10K y un pulsador conector

#include <phyphoxBle.h>   //Phyphox BLE library

const int condensador = 02;   //Pin que recoje los datos del condensador en el esp32
float voltaje;
float V;

void setup() {
   PhyphoxBLE::start("Condensador");

   //Experiment
   PhyphoxBleExperiment experiment;

   experiment.setTitle("Carga y descarga de un condensador");
   experiment.setCategory("Arduino Experiments");
   experiment.setDescription("Dibuja la gráfica del voltaje del condensador con relación al tiempo.");

   //View
   PhyphoxBleExperiment::View view;

   //Graph
   PhyphoxBleExperiment::Graph graph;
   graph.setLabel("Voltaje del condensador");
   graph.setUnitX("s");
   graph.setUnitY("V");
   graph.setLabelX("tiempo");
   graph.setLabelY("Voltaje");

   //In contrast to other examples, we will not generate the timestamp on the phone.
   //For experiments with a high data rate, we can achieve a better temporal
   //accuracy if we generate the timestamp on the Arduino and send it in pairs with the
   //measured values.

   graph.setChannel(1,2);

   view.addElement(graph);               //Attach graph to view
   experiment.addView(view);             //Attach view to experiment
   PhyphoxBLE::addExperiment(experiment); //Attach experiment to server

   Serial.begin(9600);  //inicia el terminal serie
}

void loop() {
   float t = 0.001*millis();       //Tiempo en segundos
   voltaje = analogRead(condensador);  //Voltaje del condensador

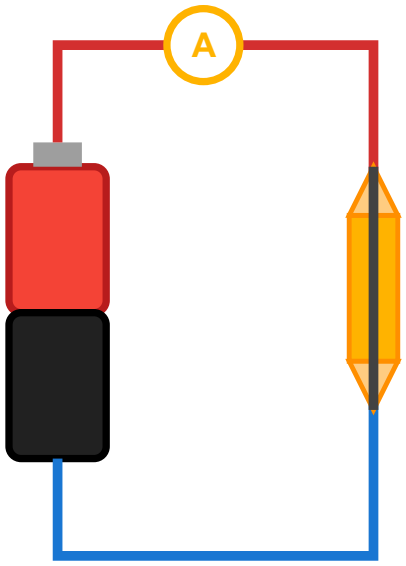
   V = voltaje*3.3/4095;
   PhyphoxBLE::write(t, V); // envia los datos a phyphox

   Serial.print(t,3); //Escribe el tiempo en segundos con tres decimales
   Serial.print("; "); //separa los valores de tiempo y distancia con un punto y coma
   Serial.println(voltaje*3.3/4095); //Escribe el voltaje del condensador en voltios
                                     //si se ha conectado a 3,3 V en un esp32
   //Serial.println(voltaje*5/1023); //Escribe el voltaje del condensador en voltios
                                     //si se ha conectado a 5V en un Arduino UNO
   delay(10);

   PhyphoxBLE::poll(); //Only required for the Arduino Nano 33 IoT, but it does no harm for other boards.
}
```

Conductividad eléctrica del grafito

Medir la resistividad eléctrica del grafito (mina de lápiz) en función de su longitud y sección transversal, verificando la segunda ley de Ohm ($R = \rho \cdot L/S$) que relaciona la resistencia de un conductor con sus dimensiones geométricas y su resistividad (Ramírez, 2025).



Fundamento físico

La resistencia eléctrica de un conductor depende de tres factores: la resistividad del material (ρ), la longitud (L) y la sección transversal (S), según la ecuación $R = \rho \cdot L/S$.

El grafito, una forma alotrópica del carbono, es un conductor moderado con una resistividad típica de $3{,}5\times10^{-5}$ Ω·m (mucho mayor que la del cobre: $1{,}7\times10^{-8}$ Ω·m).

Al dibujar una línea de grafito sobre un papel, creamos un conductor de sección aproximadamente rectangular cuyo espesor depende de la dureza del lápiz y de la presión aplicada. Además, la estructura de esta pista no es sólida, sino que está formada por micropartículas de grafito superpuestas. Esto provoca que cualquier deformación mecánica del sustrato (como doblar o estirar el papel) altere el contacto entre las partículas, modificando drásticamente su resistencia eléctrica (un efecto piezorresistivo básico).

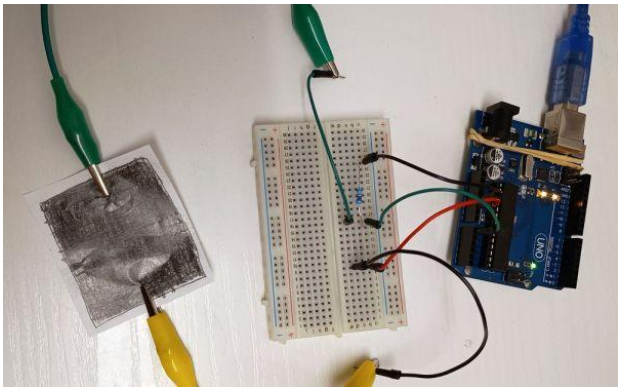
Midiendo la resistencia para diferentes longitudes y anchuras, podemos determinar experimentalmente la resistividad del grafito y su comportamiento dinámico.

Materiales y montaje experimental

Necesitaréis una placa Wemos D1 R32 (ESP32) o Arduino UNO, una resistencia de referencia conocida (1 kΩ o 10 kΩ), cables Dupont con pinzas de cocodrilo, lápices de grafito (preferiblemente blando, tipo 2B o superior), papel blanco liso, una regla milimetrada y un multímetro digital para verificar las medidas.

Dibujad sobre el papel una línea de grafito uniforme de 1 cm de ancho y 10 cm de largo, repasando varias veces con el lápiz para obtener una capa homogénea.

Conectad un extremo de la línea de grafito a 5 V (o 3,3 V) a través de la resistencia de referencia y el otro extremo a GND, y además el punto medio entre la línea de grafito y la resistencia a una entrada analógica (A0). Esencialmente, construís un divisor de tensión donde una de las resistencias es la línea de grafito.



Procedimiento

Conectad la placa al ordenador y cargad el código. Medid la caída de tensión en la línea de grafito para diferentes longitudes (recortando progresivamente la línea: 10 cm, 8 cm, 6 cm, 4 cm, 2 cm). Para cada longitud, realizad tres medidas y anotad el valor

medio. Repetid el experimento con líneas de diferente anchura (0,5 cm, 1 cm, 1,5 cm) para estudiar la dependencia con la sección.

Resultados esperados

La resistencia de la línea de grafito será del orden de decenas de kiloohmios para una línea de 10 cm × 1 cm ($\rho_{efectiva} \approx 10^{-2}$ Ω·m). Al reducir la longitud a la mitad, la resistencia debería disminuir aproximadamente a la mitad. Al duplicar la anchura (sección), la resistencia debería reducirse a la mitad. Estos comportamientos verifican la segunda ley de Ohm.

Análisis

Representad gráficamente la resistencia R frente a la longitud L para cada anchura. Deberíais obtener una relación lineal que pase cerca del origen. Para cada anchura, calculad la pendiente ΔR/ΔL y, conociendo la sección (espesor × anchura), determinad la resistividad ρ. El espesor de una línea de grafito típica es de unos 10-20 µm.

Discutid las fuentes de error: irregularidades en el grosor del grafito, resistencia de contacto de las pinzas, precisión de las medidas geométricas.

Ampliación

Influencia de la dureza del lápiz: Repetid el experimento con lápices de diferentes durezas (HB, 2B, 4B, 6B) y comparad las resistividades obtenidas. Cuanto más blando es el lápiz, mayor cantidad de grafito deposita y menor es la resistencia.



Utilizar un Energy Stic para comprobar la conductividad del grafito: Una forma más espectacular y puede ser más divertida de mostrar al alumnado la diferencia de conductividad de las minas de lápices en función del contenido menos o mayor de grafito es lo que propone Gabriel Pinto, profesor de Química en la Universidad politécnica de Madrid.

Lo que propone es dibujar sobre una cartulina A5 una H con lápiz de dureza 2H y una B sobre otra cartulina pero ahora con lápiz 2B y comprobar la diferencia de conductividades de las letras. Para ello se ayuda de la conductividad humana y de un juguete que le llaman Energy Stic (entre 10 y 20 €) que proporciona una pequeña diferencia de potencial entre sus extremos (6 V) y que si se cierra el circuito es capaz de detectar corrientes eléctricas de muy baja intensidad, reaccionando con luces y musiqueta.

Coloca a los jóvenes en círculo, dos personas se enlazan con el Energy Stic y otras dos con la letra correspondiente. En el caso de la letra B el aparato hace musiquita y con el H no reacciona, indicando la diferencia de conductividad de las dos minas de lápiz.

Efecto de la deformación (Piezorresistencia): Con el circuito conectado midiendo en tiempo real, proceded a doblar suavemente el papel (creando una curvatura cóncava o convexa) o a tensarlo tirando ligeramente de los extremos sin romperlo. Observaréis cómo varía la resistencia en el monitor serie.

Al estirar o curvar el papel, la distancia microscópica entre los granos de grafito cambia, alterando la red de percolación eléctrica. Este es el principio físico subyacente en las galgas extensiométricas y sensores de flexión (flex sensors) comerciales.

Código

```
// Medida de resistencia de una línea de grafito mediante divisor de tensión
// R_referencia conectada entre 3.3V y A0; línea de grafito entre A0 y GND

const float R_REF = 10000.0;  // Resistencia de referencia en ohmios (10 kΩ)
const float VCC = 3.3;        // Tensión de alimentacion (3.3V para ESP32; usar 5.0V si es Arduino UNO)

void setup() {                // Configuracion inicial: se ejecuta una vez al arrancar la placa
```

```
  Serial.begin(9600);                  // Iniciamos
comunicacion serie a 9600 baudios para enviar
datos al PC
}

void loop() {                          // Bucle principal: se
repite indefinidamente mientras la placa este
alimentada
  int valorADC = analogRead(A0);       // Leemos el
valor analogico del pin A0 (0-4095 en ESP32 de 12
bits)
  float Vout = valorADC * (VCC / 4095.0);  //
Convertimos el valor ADC a voltios (resolucion 12
bits del ESP32)
  // float Vout = valorADC * (5.0 / 1023.0); // Linea
alternativa para Arduino UNO (resolucion 10 bits,
Vref=5V)

  // Calculamos la resistencia del grafito despejando
R_grafito de la formula del divisor de tension:
  // Vout = VCC * R_grafito / (R_ref + R_grafito)  =>
R_grafito = R_ref * Vout / (VCC - Vout)
  float R_grafito = R_REF * Vout / (VCC - Vout);

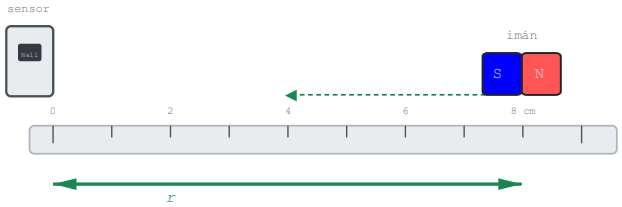
  Serial.print(millis());              // Tiempo transcurrido
desde el arranque en milisegundos
  Serial.print(";");                   // Separador punto y
coma entre campos
  Serial.println(R_grafito, 1);        // Resistencia
calculada en ohmios con 1 decimal

  delay(1000);                         // Esperamos 1
segundo entre mediciones
}
```

Variación del campo magnético de un imán con la distancia

Se va a medir el campo magnético generado por un imán permanente en función de la distancia r medida desde el centro del imán hasta el elemento sensible de un sensor de efecto Hall. El objetivo experimental es verificar que, a distancias grandes comparadas con el tamaño del imán, el campo decae según la ley del dipolo magnético:

$$B(r) \propto 1/r^3$$



Tradicionalmente se ha llevado a cabo con una brújula ayudada del campo magnético terrestre (rsef, 2007). Ahora, además de lo que se propone aquí este experimento se puede hacer más fácilmente utilizando cualquier teléfono móvil, ja que la mayoría de ellos disponen de sensor magnético (Ramírez, 2017c).

Fundamento físico

Un imán permanente se comporta aproximadamente como un dipolo magnético. A diferencia del campo eléctrico de una carga puntual que decae como $(1/r^2)$, el campo magnético de un dipolo disminuye con el cubo de la distancia. Para un dipolo puntual de momento magnético (m, en A·m²), el campo a lo largo de su eje es:

$$B(r) = \frac{\mu_0}{4\pi} \cdot \frac{2m}{r^3}$$

donde $(\mu_0 = 4\pi \times 10^{-7}\ \mathrm{T{\cdot}m/A})$ es la permeabilidad del vacío y (r) la distancia desde el centro del dipolo. Esta dependencia cúbica implica que el campo se reduce a 1/8 al duplicar la distancia.

Con un sensor Hall analógico (SS49E) o un magnetómetro digital (HMC5883L) conectado a Arduino, se mide (B) a distancias crecientes y se puede verificar la ley $(B \propto 1/r^3)$ mediante una gráfica y su línea de tendencia.

Materiales y montaje experimental

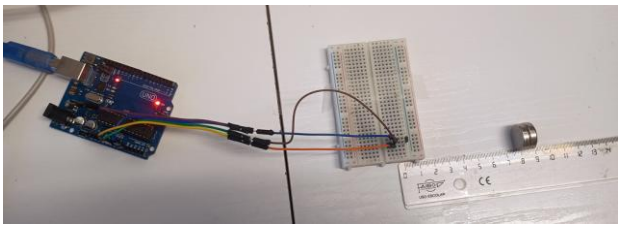
- Arduino UNO
- Sensor Hall SS49E o HMC5883L
- Condensador 100 nF (104)
- Imanes de ferrita o de neodimio (diferentes tamaños)
- Cables y placa de pruebas
- Regla o cinta métrica

Pines de conexión

Pin	Descripción
V (izquierdo)	Alimentación (5 V)
G (centro)	Tierra (0 V)
S (derecho)	Pin analógico, A0
Condensador 104	Entre A0 y GND

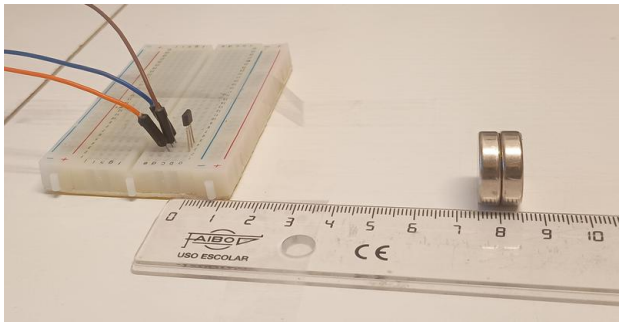
Es muy recomendable poner un condensador cerámico de 100 nF (suele poner 104), entre la pata de salida (señal) del sensor SS49E y la pata de GND (Tierra). Eso absorberá casi todo el ruido eléctrico y las lecturas serán muchísimo más estables.





Para llevar a cabo el experimento se fija el sensor Hall SS49E encima de una mesa en un punto de la regla graduada, con su cara sensible perpendicular al eje del imán, y se va deslizando el imán sobre la regla a distancias conocidas. En cada distancia se registra el valor del campo y se van apuntando los datos en una tabla de valores.

No se ha de acercar el imán a menos de 1 cm del sensor de forma prolongada, ya que un campo intenso puede magnetizar ligeramente el encapsulado.



Ejemplo de resultados a obtener

r (cm)	B (mT)	log (r)	log (B)
2,0	60,6	0,30	1,78
3,0	23,7	0,48	1,37
4,0	11,8	0,60	1,07
5,0	6,9	0,70	0,84
6,0	4,5	0,78	0,65
7,0	3,1	0,85	0,49
8,0	2,4	0,90	0,38
9,0	1,7	0,95	0,23
10,0	1,4	1,00	0,15

Análisis

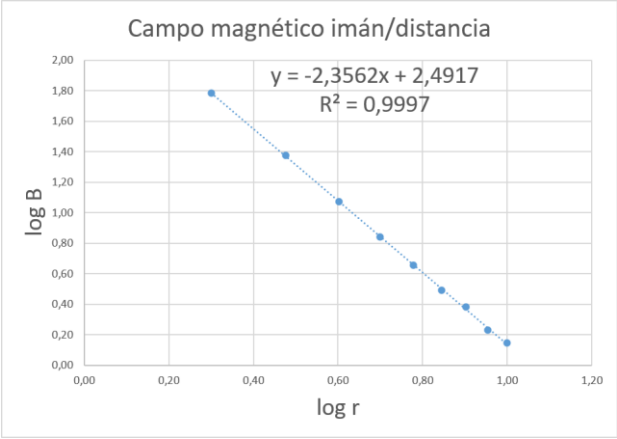
Para verificar la ley con una regresión lineal se toman logaritmos decimales en ambos miembros de la ecuación del campo magnético:

$$\log(B) = \log(\mu_0 \cdot m / 2\pi) - 3 \cdot \log(r)$$

Esta expresión tiene la forma de una recta

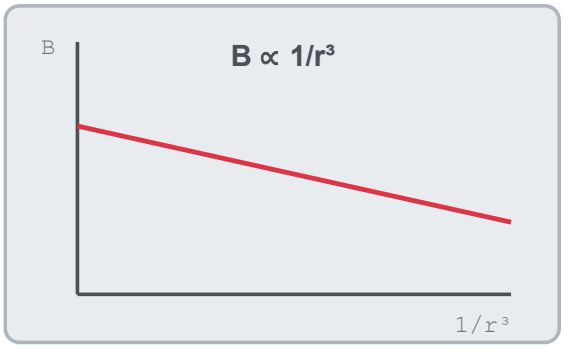
$$y = a + b \cdot x,$$

con: $y = \log(B)$, $x = \log(r)$, $b = -3$ (pendiente teórica) y $a = \log(\mu_0 \cdot m/2\pi)$ (ordenada en el origen)



Si la pendiente experimental resulta compatible con −3, el modelo de dipolo quedaría confirmado, lo que no es en el caso (-2,4) con los valores de ejemplo.

También se puede comprobar la ley representando B frente a $1/r^3$. Si el modelo es correcto, los puntos se alinean sobre una recta que pasa por el origen, cuya pendiente es $C = \mu_0 \cdot m/2\pi$. Esta forma es más sencilla porque evita los logaritmos.



Podéis comparar resultados obtenidos con diferentes imanes.

Ampliación

Explicar por qué la gráfica de B frente a r no es una recta y por qué habitualmente el exponente está entre -2 y – 3. ¿Qué ocurre si se hace $r \to 0$ en la ecuación del dipolo? ¿Es físicamente razonable?

Estudiar cómo cambia la ley si se mide el campo en dirección perpendicular al eje del imán en lugar de sobre el eje.

¿Qué efecto tiene acercar un imán de nevera al montaje mientras se toman medidas? ¿Por qué se han retirar objetos metálicos de la mesa i procurar que la mesa no tenga patas metálicas?

Si se acerca mucho el imán al sensor y se vuelve a alejar, ¿recupera la lectura inicial? ¿Qué nos dice esto sobre la histéresis y la magnetización residual?

Estudiar la elasticidad de los materiales con una viga (regla de plástico) apoyada con carga central (Bouquet y otros, 2019).

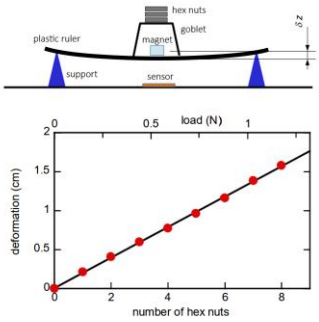


Figure 4. Top figure: experimental setup. A plastic ruler is placed on two supports, the magnet at its center. A goblet is used to keep the hex nuts apart from the magnet and avoid their magnetization. Bottom figure: results. The deformation is linear with the load, and the slope gives the Young modulus.

Estudiar la vibración de una varilla incrustada en un extremo con un imán enganchado en el otro. Para esto se debe modificar al código para que también proporcione el paso del tiempo.

Código

```
/*
VARIACIÓN DEL CAMPO MAGNÉTICO CON LA DISTANCIA
Placa: Arduino UNO
Sensor: Honeywell SS49E (Efecto Hall lineal, 5V)
El código se autocalibra
Se desprecian las medidas inferiores a 0,5 mT ya que entran dentro del error de medida del sensor en Arduino (14 mV/mT).
*/

const int PIN_HALL = A0;
const int N_MUESTRAS = 100; // Utilizamos 100 muestras para aplanar mejor el ruido

// Parámetros del sensor
const float V_REF = 5.0;
const float SENS_V_MT = 0.014;
float v_offset_real = 0;

void setup() {
  Serial.begin(9600);

  Serial.println("Calibrando sensor (No acerques imanes)...");
  delay(2000); // tiempo de calibración

  long suma_cal = 0;
  for (int i = 0; i < N_MUESTRAS; i++) {
    suma_cal += analogRead(PIN_HALL);
    delay(5);
  }
  float adc_medio_cal = (float)suma_cal / N_MUESTRAS;

  v_offset_real = adc_medio_cal * (V_REF / 1023.0);

  Serial.println("¡Calibrado!");
  Serial.println("Distancia | Campo Magnético (mT)");
  Serial.println("-----------------------------------");
}

void loop() {
  long suma = 0;

  for (int i = 0; i < N_MUESTRAS; i++) {
    suma += analogRead(PIN_HALL);
    delay(2);
  }
  float adc_media = (float)suma / N_MUESTRAS;

  float v_adc = adc_media * (V_REF / 1023.0);
  float B_mT = abs((v_adc - v_offset_real) / SENS_V_MT);

  // --- Filtro de ruido ---
  // Si la lectura es menor a 1.5 escalones del ADC (~0,5 mT), consideramos que es cero.
  if (B_mT < 0.5) {
    B_mT = 0.0;
  }

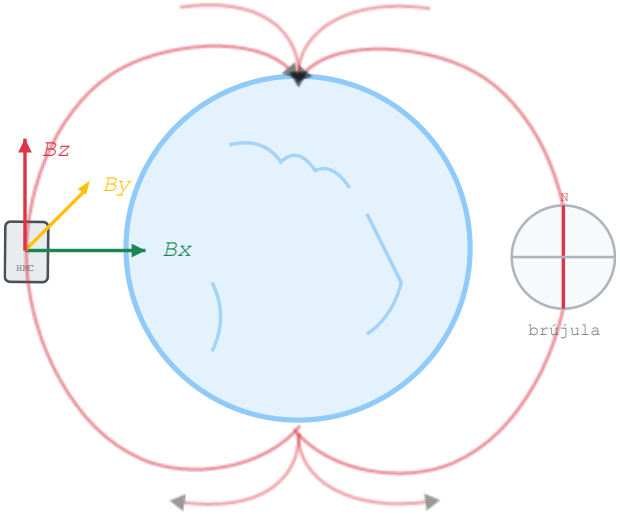
  Serial.print("Campo magnético: ");
  Serial.print(B_mT, 2);
  Serial.println(" mT");

  delay(1000);
}
```

Medida del campo magnético terrestre

Vamos a medir el vector campo magnético terrestre en el laboratorio con un magnetómetro de tres ejes (QMC5883L). A partir de sus tres componentes se calculará el módulo del campo total, la componente horizontal, la inclinación magnética y la declinación magnética local.

El resultado se comparará con el modelo de referencia IGRF (International Geomagnetic Reference Field) para la latitud y longitud del centro.



El campo magnético terrestre es un ejemplo extraordinario de campo vectorial débil: apenas 25 μT en su componente horizontal, frente a los 50 mT de un imán de nevera, es decir, unas 2000 veces menor.

Este experimento se puede hacer más fácilmente con cualquier teléfono móvil y, por ejemplo, la aplicación Phyphox, ja que la mayoría de ellos disponen de sensor magnético (Ramírez, 2017d).

Fundamento físico

En primera aproximación, el campo magnético terrestre puede describirse como el de un dipolo magnético cuyo eje está inclinado unos 11.5° respecto al eje de rotación de la Tierra. En un punto cualquiera próximo a la superficie se expresa mediante tres componentes en un sistema de coordenadas local:

$$B = (B_x, B_y, B_z)$$

B_x: componente hacia el norte magnético
B_y: componente hacia el este magnético
B_z: componente vertical (positiva hacia abajo en el hemisferio norte)

Módulo del campo total

El módulo del campo total es:

$$B = \sqrt{B_x^2 + B_y^2 + B_z^2}$$

Valores típicos para latitudes medias (Lleida, 41° 36′ 12″ (41.6167) Norte, 0° 37′ 04″ (0.6222) Este):

Componente	Valor típico (μT)
B_x (norte)	24 – 26
B_y (este)	3 – 6
B_z (vertical)	38 – 42
B_{total}	45,5

Inclinación magnética

La inclinación magnética I es el ángulo que forma el vector campo con el plano horizontal:

$I = arctan(B_z/B_H)$, con $B_H = \sqrt{({B_x}^2 + {B_y}^2)}$

Para Lleida la inclinación esperada es de unos 56° a 60° (el vector apunta hacia el suelo).

Declinación magnética

La declinación magnética D es el ángulo entre el norte magnético (proyección horizontal del campo) y el norte geográfico:

$$D = arctan(B_y/B_x)$$

En la península ibérica este ángulo es pequeño, del orden de 1,7° a 2° hacia el este para Lleida.

Materiales y montaje experimental

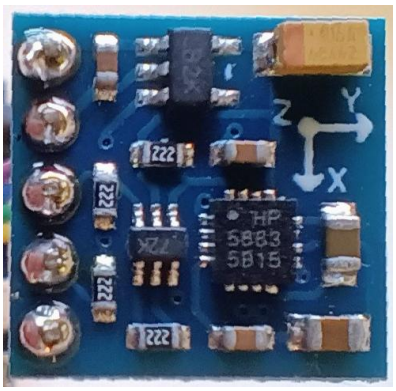
- Arduino UNO (o ESP32)
- Magnetómetro QMC5883P (o QMC5883L)
- Brújula o app de brújula (para comparación)
- Soporte nivelado

HMC5883L	Arduino	ESP32
VCC	3,3 V	3.3V
GND	GND	GND
SCL	A5	GPIO22
SDA	A4	GPIO21

Un magnetómetro puede sufrir errores sistemáticos debidos principalmente a campos permanentes generados por materiales ferromagnéticos que haya en las proximidades que producen un desplazamiento constante de las lecturas, de modo que el círculo que debería describir el extremo del vector campo al girar el sensor aparece centrado en un punto distinto del origen. Se corrigen restando el offset estimado:

$$B_x, offset = (B_x, máx + B_x, mín)/2$$

El QMC5883L es un magnetómetro magnetorresistivo que incluye una secuencia interna de set/reset que invierte periódicamente la magnetización del elemento sensible para cancelar derivas térmicas y offsets.



En todo caso es conveniente conectar el sensor alejando el montaje de ordenadores, altavoces, transformadores y estructuras metálicas.

Al conectar el sensor o al resetearlo hay que ejecutar el protocolo de calibración girando el sensor haciendo ochos durante 20 segundos.

Análisis

Los resultados que podemos obtener a través del monitor serie son del siguiente orden:

t_ms;Bx_uT;By_uT;Bz_uT;Btotal_uT;Incl_deg;AzimutMag_deg;RumboReal_deg

21737;-7.67;16.20;40.73;44.50;66.3;115.3;116.5

Donde se ve que el campo magnético terrestre tota da 44,5 μT, la inclinación 66,3º y el código ya considera la declinación 1,2º (116,5 – 115,3).

Se pueden comparar estos datos con el modelo del Campo de referencia geomagnético internacional (IGRF). El IGRF es un modelo de referencia que permite calcular el campo teórico en función de latitud, longitud, altitud y fecha (NCEI, 2026, Geomagnetism, 2026). Para una localidad de la península ibérica el modelo predice valores de B_{total} en torno a 46. La comparación cuantitativa entre la medida y el modelo permite discutir el efecto del entorno sobre las medidas geomagnéticas.

Ampliación

Realizar el mapa magnético del aula apuntando B en una cuadrícula y dibujando isolíneas.

Medir B en distintos pisos del centro y ver si varia con la altura.

Medir B en el patio y en un entorno con estructuras metálicas (dentro del centro) y comparar lo que sería la variación en las medidas entre entornos abiertos y urbanos.

Comparar la medida del QMC5883P con la del magnetómetro del smartphone mediante la app Phyphox y analizar las diferencias.

Código

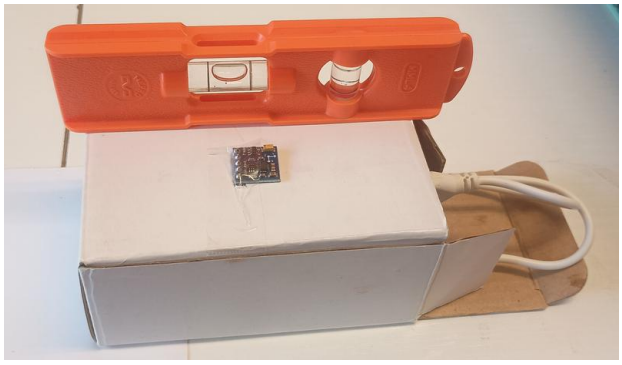
Entorno de medición y proceso de calibración
Alejaros de ordenadores, monitores, llaves, teléfonos, fuentes de alimentación y mesas con patas de hierro. Lo ideal es ir al centro de un aula libre de metales o, aún mejor, al aire libre. Procurad que el sensor esté separado al menos unos centímetros de la placa Arduino/ESP32, ya que la placa misma emite ruido magnético.

Conectad el Arduino/ESP32 y abrid inmediatamente el Monitor Serie a 115200 baudios. Se ve que empieza una cuenta atrás de 20 segundos.

Durante estos 20 s coged la protoboard+sensor en la mano y movedlo en el aire describiendo la forma de un "8" tridimensional. Hay que rotar el sensor sobre todos sus ejes (boca arriba, boca abajo, de lado a la izquierda, de lado a la derecha, apuntando al techo, apuntando al suelo). El objetivo es que el sensor experimente el campo terrestre desde absolutamente todas las orientaciones posibles. Cuando termine el tiempo, imprimirá los valores OFF_X, OFF_Y, OFF_Z.

Una vez terminada la calibración, el monitor serie empieza a escribir los valores separados por puntos y comas.

Para obtener lecturas válidas poned el sensor completamente plano sobre una superficie no metálica (una caja de cartón o madera). Si no está plano, los cálculos de azimut y rumbo real (Norte) tendrán un error trigonométrico importante (error por cabeceo/alabeo).



Después girad la placa suavemente sobre la mesa. Cuando la columna $RumboReal_{degr}$ marque 0.0, el eje X del sensor (suele venir marcado con una flechita en el PCB) estará apuntando exactamente al Norte geográfico. Y a esperar que los valores sean buenos 😉.

Código

Recordar que al encender o resetear la placa ESP32 o Arduino Uno se produce la calibración del sensor durante 20 segundos. En este tiempo se ha de estar moviendo el sensor en el aire haciendo ochos en diferentes direcciones del espacio.

```
/*
MEDIDA DEL CAMPO MAGNÉTICO TERRESTRE
Sensor: QMC5883P (Módulo clon "HP 5883" - Dirección 0x2C)
Con autocalibración
Envío de datos por el monitor serie y a Phyphox
*/

#include <Arduino.h>
#include <Wire.h>
#include <phyphoxBle.h>

const uint8_t QMC_ADDR = 0x2C;

// Parámetros de conversión
const float LSB_POR_GAUSS = 3000.0f;
const float DECLINACION_GRADOS = 1.2f;

// Factor para corregir la caída de sensibilidad (x4) típica de este sensor
const float FACTOR_CORRECCION = 4.0f;

float OFF_X = 0.0f;
float OFF_Y = 0.0f;
float OFF_Z = 0.0f;

const unsigned long T_MUESTREO_MS = 200;

void escribirRegistro(uint8_t reg, uint8_t valor) {
  Wire.beginTransmission(QMC_ADDR);
  Wire.write(reg);
  Wire.write(valor);
  Wire.endTransmission();
}

void configurarSensor() {
  escribirRegistro(0x0B, 0x01);
  escribirRegistro(0x0A, 0xCF);
}

bool leerCrudo(int16_t &rx, int16_t &ry, int16_t &rz)
{
  Wire.beginTransmission(QMC_ADDR);
  Wire.write(0x01);
  if (Wire.endTransmission() != 0) return false;
  if (Wire.requestFrom(QMC_ADDR, (uint8_t)6) != 6) return false;

  uint8_t xl = Wire.read(); uint8_t xh = Wire.read();
  uint8_t yl = Wire.read(); uint8_t yh = Wire.read();
  uint8_t zl = Wire.read(); uint8_t zh = Wire.read();

  rx = (int16_t)((xh << 8) | xl);
  ry = (int16_t)((yh << 8) | yl);
  rz = (int16_t)((zh << 8) | zl);
  return true;
}

float aMicroteslas(int16_t crudo) {
  // Aplicamos el factor de corrección para escalar los valores a la realidad
  return ((crudo / LSB_POR_GAUSS) * 100.0f) * FACTOR_CORRECCION;
}

void calibrarSensor() {
  float min_x = 100000, max_x = -100000;
```

```
  float min_y = 100000, max_y = -100000;
  float min_z = 100000, max_z = -100000;

  Serial.println(F("\n--- CALIBRANDO (20s) - Dibuja un 8 ---"));

  unsigned long t_inicio = millis();
  unsigned long ultimo_print = millis();

  while (millis() - t_inicio < 20000) {
    int16_t rx, ry, rz;
    if (leerCrudo(rx, ry, rz)) {
      if (rx != 0 || ry != 0 || rz != 0) {
        float bx = aMicroteslas(rx);
        float by = aMicroteslas(ry);
        float bz = aMicroteslas(rz);

        if (bx < min_x) min_x = bx; if (bx > max_x) max_x = bx;
        if (by < min_y) min_y = by; if (by > max_y) max_y = by;
        if (bz < min_z) min_z = bz; if (bz > max_z) max_z = bz;
      }
    }

    if (millis() - ultimo_print > 1000) {
      Serial.print(20 - ((millis() - t_inicio) / 1000));
      Serial.println(F("s restantes..."));
      ultimo_print = millis();
    }
    delay(20);
  }

  OFF_X = (max_x + min_x) / 2.0;
  OFF_Y = (max_y + min_y) / 2.0;
  OFF_Z = (max_z + min_z) / 2.0;

  Serial.println(F("\n--- CALIBRACION COMPLETADA ---"));
  delay(1000);
}

void setup() {
  Serial.begin(115200);
  while (!Serial && millis() < 3000);

  Wire.begin();
  configurarSensor();
  delay(100);

  calibrarSensor();

  // --- CONFIGURACION PHYPHOX ---
  PhyphoxBLE::start("Mag_ESP32");

  PhyphoxBleExperiment config;
  config.setTitle("Brujula Digital");
  config.setCategory("Sensores ESP32");

  // PESTAÑA 1: DATOS (5 Canales: X, Y, Z, BTotal, Inclinación)
  PhyphoxBleExperiment::View vistaDatos;
  vistaDatos.setLabel("Magnetometro");

  PhyphoxBleExperiment::Value vBx;
vBx.setLabel("Eje X"); vBx.setUnit("uT");
vBx.setChannel(1); vistaDatos.addElement(vBx);
  PhyphoxBleExperiment::Value vBy;
vBy.setLabel("Eje Y"); vBy.setUnit("uT");
vBy.setChannel(2); vistaDatos.addElement(vBy);
  PhyphoxBleExperiment::Value vBz;
vBz.setLabel("Eje Z"); vBz.setUnit("uT");
vBz.setChannel(3); vistaDatos.addElement(vBz);
  PhyphoxBleExperiment::Value vBtot;
vBtot.setLabel("Campo Total"); vBtot.setUnit("uT");
vBtot.setChannel(4); vistaDatos.addElement(vBtot);
  PhyphoxBleExperiment::Value vIncl;
vIncl.setLabel("Inclinacion"); vIncl.setUnit("deg");
vIncl.setChannel(5); vistaDatos.addElement(vIncl);

  config.addView(vistaDatos);

  // PESTAÑA 2: GRÁFICAS (Usando el Canal 0 para el Tiempo interno de la app)
  PhyphoxBleExperiment::View vistaGraficas;
  vistaGraficas.setLabel("Graficas");

  PhyphoxBleExperiment::Graph gBtot;
  gBtot.setLabel("Campo Magnetico Total");
  gBtot.setUnitX("s");
  gBtot.setUnitY("uT");
  gBtot.setLabelX("Tiempo");
  gBtot.setLabelY("B Total");
  gBtot.setChannel(0, 4); // Canal 0 (Tiempo de la app) vs Canal 4 (Btotal)
  vistaGraficas.addElement(gBtot);

  PhyphoxBleExperiment::Graph gIncl;
  gIncl.setLabel("Inclinacion Magnetica");
  gIncl.setUnitX("s");
  gIncl.setUnitY("deg");
  gIncl.setLabelX("Tiempo");
  gIncl.setLabelY("Inclinacion");
  gIncl.setChannel(0, 5); // Canal 0 (Tiempo de la app) vs Canal 5 (Inclinacion)
```

```
  vistaGraficas.addElement(gIncl);

  config.addView(vistaGraficas);

  PhyphoxBLE::addExperiment(config);

  // Imprimir cabecera CSV para el ordenador

Serial.println(F("\nt_ms;Bx_uT;By_uT;Bz_uT;Btotal
_uT;Incl_deg;AzimutMag_deg;RumboReal_deg"));
}

void loop() {
  static unsigned long t_previo = 0;
  PhyphoxBLE::poll();

  if (millis() - t_previo >= T_MUESTREO_MS) {
    t_previo = millis();

    int16_t rx, ry, rz;
    if (!leerCrudo(rx, ry, rz)) return;

    float Bx = aMicroteslas(rx) - OFF_X;
    float By = aMicroteslas(ry) - OFF_Y;
    float Bz = aMicroteslas(rz) - OFF_Z;

    // Cálculo de magnitudes
    float BH = sqrt(Bx * Bx + By * By);
    float Btotal = sqrt(Bx * Bx + By * By + Bz * Bz);

    float incl = atan2(Bz, BH) * 180.0f / PI;

    float azimut = atan2(By, Bx) * 180.0f / PI;
    if (azimut < 0) azimut += 360.0f;

    float rumbo_real = azimut +
DECLINACION_GRADOS;
    if (rumbo_real >= 360.0f) rumbo_real -= 360.0f;
    if (rumbo_real < 0) rumbo_real += 360.0f;

  // ENVÍO A PHYPHOX (Máximo 5 elementos)
  // Canales: 1=Bx, 2=By, 3=Bz, 4=Btotal,
5=Inclinación
    float datos_phyphox[5] = {Bx, By, Bz, Btotal,
incl};
    PhyphoxBLE::write(datos_phyphox, 5);

  // ENVÍO COMPLETO AL MONITOR SERIE
    Serial.print(millis());      Serial.print(';');
    Serial.print(Bx, 2);         Serial.print(';');
    Serial.print(By, 2);         Serial.print(';');
    Serial.print(Bz, 2);         Serial.print(';');
    Serial.print(Btotal, 2);     Serial.print(';');
    Serial.print(incl, 1);       Serial.print(';');
    Serial.print(azimut, 1);     Serial.print(';');
    Serial.println(rumbo_real, 1);
  }
}
```

```
/*
MEDIDA DEL CAMPO MAGNÉTICO TERRESTRE
Sensor: QMC5883L (I2C)
Placas: ESP32 / Arduino
Con autocalibración
Salida de datos solo por el puerto serie
 */

#include <Arduino.h>
#include <Wire.h>

// DIRECCIONES Y REGISTROS DEL QMC5883L
const uint8_t QMC_ADDR   = 0x0D;
const uint8_t REG_DATA   = 0x00;
const uint8_t REG_CR1    = 0x09;
const uint8_t REG_SETRES = 0x0B;
const uint8_t CR1_VALUE  = 0x45; // OSR=128,
RNG=±2G, ODR=50Hz, Modo Continuo

// PARÁMETROS DE CONVERSIÓN Y DE TU
UBICACIÓN
const float LSB_POR_GAUSS = 12000.0f;

// Declinación magnética (Lleida, España ~ +1.2
grados)
const float DECLINACION_GRADOS = 1.2f;

// Variables para guardar los offsets (ahora no son
constantes)
float OFF_X = 0.0f;
float OFF_Y = 0.0f;
float OFF_Z = 0.0f;

const unsigned long T_MUESTREO_MS = 200; // 5
Hz

// FUNCIONES DE CONTROL DEL SENSOR
void escribirRegistro(uint8_t reg, uint8_t valor) {
  Wire.beginTransmission(QMC_ADDR);
  Wire.write(reg);
  Wire.write(valor);
  Wire.endTransmission();
}

void configurarSensor() {
  escribirRegistro(REG_SETRES, 0x01);
```

```
  escribirRegistro(REG_CR1, CR1_VALUE);
}

bool leerCrudo(int16_t &rx, int16_t &ry, int16_t &rz)
{
  Wire.beginTransmission(QMC_ADDR);
  Wire.write(REG_DATA);
  if (Wire.endTransmission() != 0) return false;
  if (Wire.requestFrom(QMC_ADDR, (uint8_t)6) !=
6) return false;

  uint8_t xl = Wire.read(); uint8_t xh = Wire.read();
  uint8_t yl = Wire.read(); uint8_t yh = Wire.read();
  uint8_t zl = Wire.read(); uint8_t zh = Wire.read();

  rx = (int16_t)((xh << 8) | xl);
  ry = (int16_t)((yh << 8) | yl);
  rz = (int16_t)((zh << 8) | zl);
  return true;
}

float aMicroteslas(int16_t crudo) {
  return (crudo / LSB_POR_GAUSS) * 100.0f;
}

// RUTINA DE CALIBRACIÓN HARD-IRON
void calibrarSensor() {
  float min_x = 100000, max_x = -100000;
  float min_y = 100000, max_y = -100000;
  float min_z = 100000, max_z = -100000;

  Serial.println(F("\n=============="));
  Serial.println(F(" INICIANDO CALIBRACIÓN (20
segundos)"));
  Serial.println(F("================"));
  Serial.println(F("GIRA EL SENSOR EN TODAS
LAS DIRECCIONES"));
  Serial.println(F("Dibuja un '8' en el aire cubriendo
todos los ejes."));

  unsigned long t_inicio = millis();
  unsigned long ultimo_print = millis();

  while (millis() - t_inicio < 20000) { // 20 segundos
    int16_t rx, ry, rz;
    if (leerCrudo(rx, ry, rz)) {
      float bx = aMicroteslas(rx);
      float by = aMicroteslas(ry);
      float bz = aMicroteslas(rz);

      if (bx < min_x) min_x = bx;
      if (bx > max_x) max_x = bx;
      if (by < min_y) min_y = by;
      if (by > max_y) max_y = by;
      if (bz < min_z) min_z = bz;
      if (bz > max_z) max_z = bz;
    }

    // Cuenta atrás en el monitor serie
    if (millis() - ultimo_print > 1000) {
      Serial.print(20 - ((millis() - t_inicio) / 1000));
      Serial.println(F(" seg restantes..."));
      ultimo_print = millis();
    }
    delay(20); // Leer aprox a 50Hz
  }

  // El centro de la esfera magnética medida nos da
el Offset
  OFF_X = (max_x + min_x) / 2.0;
  OFF_Y = (max_y + min_y) / 2.0;
  OFF_Z = (max_z + min_z) / 2.0;

  Serial.println(F("\n--- CALIBRACIÓN
COMPLETADA ---"));
  Serial.print(F("OFF_X: ")); Serial.print(OFF_X, 2);
Serial.println(F(" uT"));
  Serial.print(F("OFF_Y: ")); Serial.print(OFF_Y, 2);
Serial.println(F(" uT"));
  Serial.print(F("OFF_Z: ")); Serial.print(OFF_Z, 2);
Serial.println(F(" uT"));
  Serial.println(F("------------------------------\n"));
  delay(3000); // Pausa para que el usuario pueda
leer los resultados
}

void setup() {
  Serial.begin(115200);
  while (!Serial && millis() < 3000);

  Wire.begin();
  configurarSensor();

  // Ejecuta la calibración al arrancar
  calibrarSensor();

  // Imprime la cabecera CSV

Serial.println(F("t_ms;Bx_uT;By_uT;Bz_uT;Btotal_u
T;Incl_deg;AzimutMag_deg;RumboReal_deg"));
}

void loop() {
  static unsigned long t_previo = 0;

  if (millis() - t_previo >= T_MUESTREO_MS) {
```

```
    t_previo = millis();

    int16_t rx, ry, rz;
    if (!leerCrudo(rx, ry, rz)) return;

    // Aplicar la calibración calculada en el Setup
    float Bx = aMicroteslas(rx) - OFF_X;
    float By = aMicroteslas(ry) - OFF_Y;
    float Bz = aMicroteslas(rz) - OFF_Z;

    // Magnitudes derivadas
    float BH     = sqrt(Bx * Bx + By * By);
    float Btotal = sqrt(Bx * Bx + By * By + Bz * Bz);

    // Inclinación magnética (Pitch del campo)
    float incl = atan2(Bz, BH) * 180.0f / PI;

    // Azimut magnético (respecto al plano XY,
asumiendo sensor plano sobre la mesa)
    float azimut = atan2(By, Bx) * 180.0f / PI;
    if (azimut < 0) azimut += 360.0f;

    // Rumbo Real Geográfico
    float rumbo_real = azimut +
DECLINACION_GRADOS;
    if (rumbo_real >= 360.0f) rumbo_real -= 360.0f;
    if (rumbo_real < 0) rumbo_real += 360.0f;

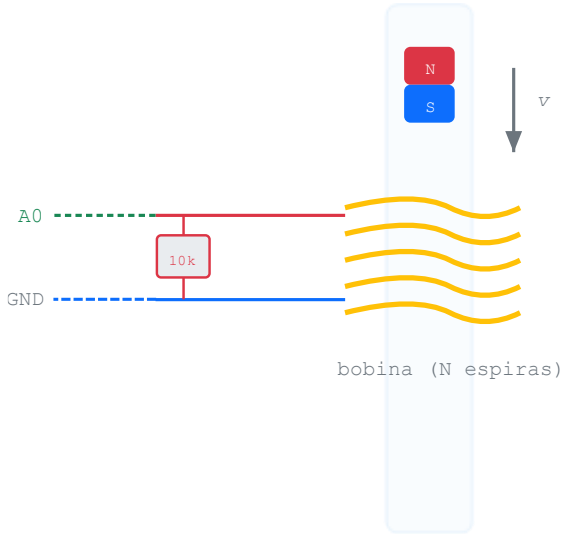
    // Envío CSV
    Serial.print(millis());      Serial.print(';');
    Serial.print(Bx, 2);         Serial.print(';');
    Serial.print(By, 2);         Serial.print(';');
    Serial.print(Bz, 2);         Serial.print(';');
    Serial.print(Btotal, 2);     Serial.print(';');
    Serial.print(incl, 1);       Serial.print(';');
    Serial.print(azimut, 1);     Serial.print(';');
    Serial.println(rumbo_real, 1);
  }
}
```

Inducción magnética en una espira

Se va a inducir una fuerza electromotriz (FEM) en una espira conductora haciendo pasar un imán de neodimio a través de ella, y se registrará la forma de onda de la FEM con el ADC del microcontrolador.

Con los datos se verificará cuantitativamente la ley de Faraday-Lenz y se comprobará que el área neta bajo el pulso de FEM es prácticamente nula:

$$\varepsilon = -N \cdot d\Phi_B/dt$$



Una alternativa consiste en recoger la variación de voltaje que se produce en la bobina en un ordenador, tableta o teléfono móvil a través del Jack de los auriculares y estudiar la variación con un programa como Audacity o una aplicación como AudioTodo (Ramírez, 2021e).

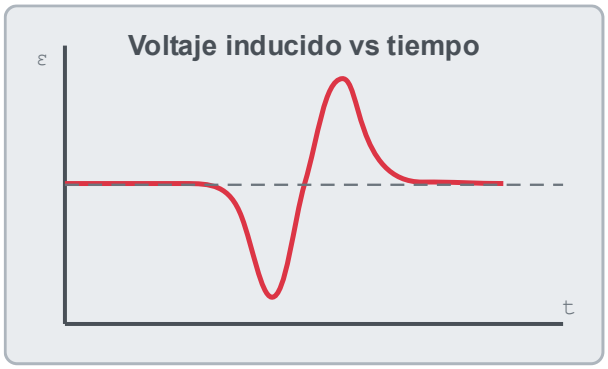
Fundamento físico

La ley de inducción de Faraday establece que la fuerza electromotriz (fem) inducida en un circuito es igual a la derivada temporal del flujo magnético que lo atraviesa, cambiada de signo (ley de Lenz):

$$\mathcal{E} = -\frac{d\Phi_B}{dt}$$

donde ($\Phi_B = \int B \cdot dA$) es el flujo magnético (Wb). El signo negativo indica que la fem inducida se opone a la variación de flujo que la produce. Al desplazar un imán permanente a través de una bobina, el flujo magnético a través de las espiras varía con el tiempo, generando un voltaje inducido que Arduino puede registrar a través de una entrada analógica protegida con una resistencia limitadora.

Este fenómeno es el fundamento de generadores eléctricos, transformadores y sensores inductivos. La magnitud de la fem depende de la velocidad del imán, del número de espiras \(N\) y de la intensidad del campo magnético.



Si la espira tiene N vueltas, la fuerza electromotriz inducida será:

$$\varepsilon = -N \cdot d\Phi_B/dt$$

El flujo magnético a través de una superficie se define como:

$$\Phi_B = \int B \cdot dA = B \cdot A \cdot cos\theta$$

Y para una espira plana de área A perpendicular al campo (θ = 0):

$$\Phi_B = B \cdot A$$

Si el imán se aproxima a la espira con velocidad v, el campo que la atraviesa depende de la distancia r(t):

$$\Phi_B(t) = B\big(r(t)\big) \cdot A$$

y la FEM resulta:

$$\varepsilon(t) = -N \cdot A \cdot (dB/dr) \cdot (dr/dt)$$
$$= -N \cdot A \cdot (dB/dr) \cdot v$$

La FEM es proporcional a la velocidad con la que se mueve el imán: cuanto más rápido, mayor es la FEM inducida.

Al acercarse y alejarse el imán se obtiene un pulso doble: una parte positiva (al entrar) y otra negativa (al salir), de modo que el área neta bajo la curva es aproximadamente cero.

Integrando la ley de Faraday entre dos instantes:

$$\int \varepsilon dt = -N \cdot [\Phi_B(t_2) - \Phi_B(t_1)]$$

Si el imán parte del reposo lejos de la espira y vuelve al reposo lejos de ella, el flujo inicial y el final son ambos nulos, de modo que:

$$\int \varepsilon dt = 0$$

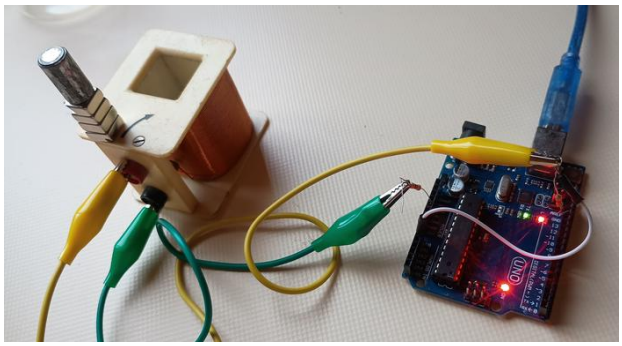
El área positiva debe ser igual al área negativa. Esta comprobación es una excelente forma de validar experimentalmente la ley.

Materiales y montaje experimental

- Arduino UNO (o ESP32)
- Bobina de 50–500 espiras
- Imanes de neodimio
- Tubo de caída para el imán de cartón o plástico (no magnético)
- Resistencia 10 kΩ (protección)

Señal	Arduino UNO/Nano
Alimentación del divisor	5V
Nodo de medida (punto medio del divisor)	A0
Bobina Terminal A	Resistencia
Bobina Terminal B	GND

Conectad una resistencia de entre 1 a 10 kΩ a 5 V en serie con otra resistencia igual de 1 a 10 kΩ y la bobina. Conectad el punto de unión de las dos resistencia a A0 y la bobina a tierra (GND). Dejad caer el imán por el interior de la bobina.



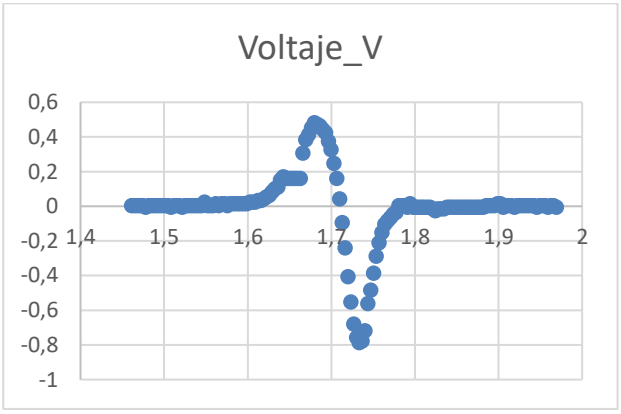
Utilizar una bobina de 400 espiras de laboratorio o bobinar 50 vueltas (o más) ajustadas de hilo de cobre sobre un tubo de cartón o plástico de 3 cm de diámetro.

Con el imán lejos, y una vez cargado el código, observad la tensión de reposo (debe estar próxima a 0 V).

Dejar caer el imán por el tubo, atravesando la espira, desde distintas alturas, registrar los pulsos y exportar los valores.

Análisis

El voltaje inducido muestra un pico positivo y otro negativo al pasar el imán. Se puede integrar la curva para estimar el flujo total.



Para calcular la integral de los datos discretos en una hoja de cálculo, el método más sencillo y preciso es la Regla del Trapecio. Este método calcula el área bajo la curva dividiendo el gráfico en "trapecios" verticales entre cada par de puntos y luego sumando todas las áreas.

Imaginad que tenéis los datos organizados en dos columnas en Excel:
Columna A: El Tiempo (t) en segundos. (Ejemplo: celdas A2 a A10).
Columna B: El voltaje (V) que hay que integrar. (Ejemplo: celdas B2 a B10).

La fórmula matemática para calcular el área de un trapecio entre dos puntos es:

$$\text{Área} = \frac{(y_1 + y_2)}{2} \times (t_2 - t_1)$$

En Excel, se puede hacer de la siguiente manera:
Primero hay que calcular el área del primer intervalo:
En la celda **C3** (al lado del segundo dato del voltaje), escribid la siguiente fórmula:

=(B2+B3)/2*(A3-A2)

A continuación copiad la fórmula hacia abajo haciendo clic en la esquina inferior derecha de la celda C3 arrastrándola hacia abajo hasta llegar al último dato (celda C10). Esto calculará el área individual de cada pequeño tramo de tiempo.

Por último, hay que sumar todas las áreas (la integral total): En la celda donde quieras ver el resultado final, utiliza la función suma para juntar todos los tramos:

=SUMA(C3:C10)

El resultado de esa suma es el valor aproximado de la integral del voltaje en ese periodo de tiempo, que debe dar aproximadamente cero. En el ejemplo de la gráfica de arriba la suma da -0,0016.

Ampliación

Se pueden contestar las siguientes preguntas
¿Por qué la FEM cambia de signo al invertir el movimiento del imán?
¿Qué ocurre con la FEM si se mueve el imán más despacio? ¿Y si se mueve más rápido? Justifícalo con la ley de Faraday.
Si se duplica el número de espiras, ¿qué le sucede a la FEM? ¿Por qué?
¿Por qué la integral de la FEM a lo largo de todo el tránsito es nula?
¿Qué diferencias habría si en lugar de una espira se usara un solenoide largo?

Código

```
/*
EXPERIMENTO : Inducción Magnética en una Espira (Ley de Faraday-Lenz)
 PLATAFORMA : Arduino UNO R3 (ATmega328P, 16 MHz, ADC 10 bits)
 CIRCUITO   : 5V -- [1kΩ] -- A0 -- [Bobina] -- [1kΩ] -- GND
*/

const uint8_t PIN_ADC = A0;
const float VCC = 5.0f;
const float VOLTS_POR_CUENTA = VCC / 1024.0f;

// Variable de offset auto-calibrada en reposo
float v_offset_medido = 2.51f;
unsigned long tiempo_inicial = 0;

void setup() {
  Serial.begin(115200);

  // Promediar 100 lecturas iniciales en reposo para fijar el offset exacto de 2.5V
  long suma_raw = 0;
  for (int i = 0; i < 100; i++) {
    suma_raw += analogRead(PIN_ADC);
    delay(2);
  }
  v_offset_medido = (suma_raw / 100.0f) * VOLTS_POR_CUENTA;

  // Cabecera para Serial Plotter / Monitor Serie / Excel
  Serial.println(F("Tiempo_s,Voltaje_V"));
  tiempo_inicial = micros();
}

void loop() {
  unsigned long tiempo_actual = micros();
  float t_segundos = (tiempo_actual - tiempo_inicial) / 1000000.0f;

  int lectura_raw = analogRead(PIN_ADC);
  float voltaje_A0 = (float)lectura_raw * VOLTS_POR_CUENTA;

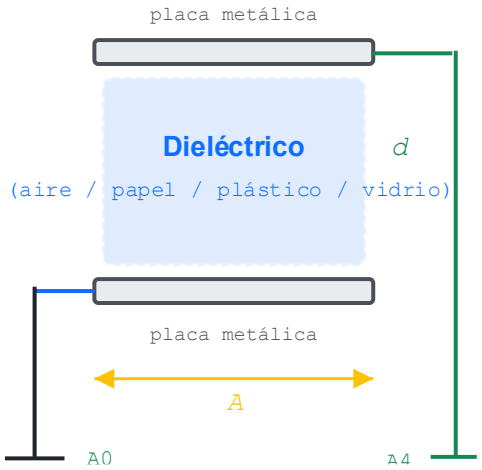
  // Multiplicamos por 2.0 para compensar el divisor Thévenin (0.5 * E) y recuperar E(t) real
  float fem_inducida = (voltaje_A0 - v_offset_medido) * 2.0f;

  Serial.print(t_segundos, 4);
  Serial.print(F(","));
  Serial.println(fem_inducida, 4);

  delay(2); // ~250-300 Hz de muestreo real debido al overhead del Serial
}
```

Permitividad eléctrica de materiales

Mediremos la constante dieléctrica (ε_r) de diferentes materiales (aire, papel, plástico, vidrio) a partir de la capacidad de un condensador de placas paralelas conectado a Arduino como capacímetro, y no a un capacímetro comercial (UPC, 2026).



Este experimento ya lo había propuesto con anterioridad (Ramírez, 2022b).

Fundamento físico

Un condensador de placas paralelas almacena carga eléctrica cuando se aplica una diferencia de potencial entre sus placas. Su capacidad depende de la geometría y del material dieléctrico interpuesto:

$$C = \varepsilon_0 \varepsilon_r \frac{A}{d}$$

donde (C) es la capacidad (F), ($\varepsilon_0 = 8,85 \times 10^{-12}\ \mathrm{F/m}$) la permitividad del vacío, (ε_r) la permitividad relativa (constante dieléctrica) del material, (A) el área de las placas (m²) y (d) la separación entre ellas (m). Al insertar un dieléctrico, la capacidad aumenta en un factor (ε_r).

El tiempo característico de carga del condensador a través de una resistencia (R) es la constante de tiempo ($\tau = RC$). Arduino mide el tiempo que tarda la tensión en alcanzar un umbral conocido (por ejemplo, el 63,2% del voltaje final, equivalente a una constante de tiempo). Comparando el tiempo de carga con y sin material dieléctrico se obtiene (ε_r) experimental.

Materiales y montaje experimental

- Arduino UNO
- Condensador de placas paralelas (ver mas adelante)
- Muestras: papel, plástico, vidrio, cartón

Se necesita un condensador de láminas planas y paralelas regulable en distancia y de una superficie relativamente grande (más de 200 cm^2) para que su capacidad tenga un valor suficientemente grande que permita su medida. El dispositivo se puede comprar en casas de material didáctico a un precio elevado, pero también se puede construir uno con dos planchas de metal (aluminio, acero, etc) de unos 20×20 cm y que sean bien planas.

Una alternativa aún más asequible es la de fabricar el condensador con dos planchas de madera o plástico (siempre bien planas) a las que pegar con pegamento papel de aluminio, y colocar luego una sobre otra.

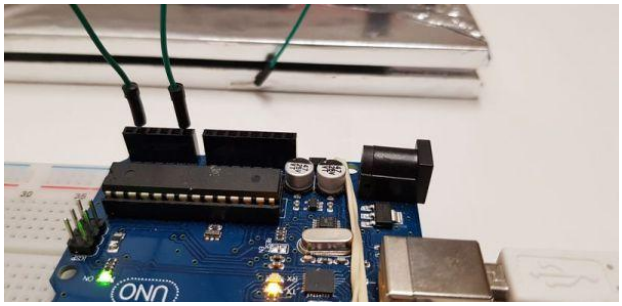


Para conseguir que no haya contacto entre las placas, cuando se quiere que el dieléctrico sea el aire, se ponen cuadritos de papel o plástico en las cuatro esquinas. También debe conectarse un cable conductor a cada una de las placas pegándolo con cinta adhesiva.

En función de la cantidad de papelitos que se pongan entre las placas la distancia entre ellas será mayor o menor. Por ejemplo se pueden poner 30 papeles en cada esquina, lo que puede suponer una separación de 3,09 mm.

Para medir la capacidad del condensador se puede utilizar un capacímetro, o usar un Arduino como tal, que es lo que haremos aquí.

Como se ve en la fotografía de abajo, lo que debe hacerse es conectar una de las placas del condensador en el pin A0 del Arduino y la otra placa en el pin A4. Es indiferente cuál a cuál puesto que el condensador plano es simétrico y no tiene polaridad como los electrolíticos.



Análisis

Con unas superficies de las placas del condensador de 21,0 x 21,0 cm = 441 cm^2 y un resultado de C = 116 pF para el aire, la permitividad sale:

$$\varepsilon_0 = \frac{C \cdot d}{S} = \frac{116 \cdot 10^{-12} \cdot 3{,}09 \cdot 10^{-3}}{4{,}41 \cdot 10^{-2}} = 8{,}13 \cdot 10^{-12} F/m$$

Lo que no se aleja mucho de lo establecido oficialmente ($8{,}854 \cdot 10^{-12}$ F/m).

Ampliación

Se puede repetir el experimento introduciendo entre las placa metálicas láminas de diferentes materiales y/o de diferentes grosores y volver a medir la nueva capacidad del condensador para calcular la permitividad de cada material, y después compararla con los valores oficiales.

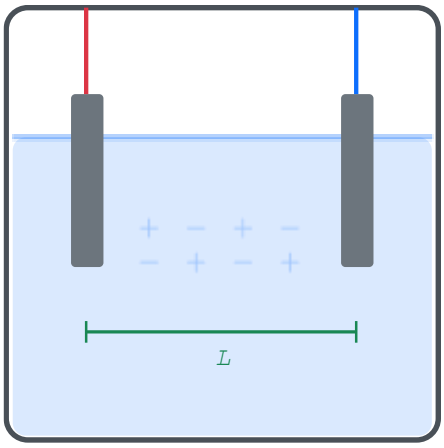
Código

```
//Capacitor para picofaradios (pF)
//Creado por Electgpl
//http://electgpl.blogspot.com/2018/02/pico-capacimetro-con-arduino-y-display.html
//Conectar una pata del condensador al pin A0 y la otra al pin A4
//y ver el resultado por el puerto serie

const int OUT_PIN = A4;
const int IN_PIN = A0;
long int acumulado=0;
long int actualizado=0;
void setup(){
  Serial.begin(9600);
  pinMode(OUT_PIN, OUTPUT);
  pinMode(IN_PIN, OUTPUT);
}
void loop(){
  for(int i=0; i<100; i++){
    pinMode(IN_PIN, INPUT);
    digitalWrite(OUT_PIN, HIGH);
    int val = analogRead(IN_PIN);
    digitalWrite(OUT_PIN, LOW);
    pinMode(IN_PIN, OUTPUT);
    float capacitancia = (float)val * 24.5 / (float)(1023 - val);
    acumulado = capacitancia + acumulado;
  }
  actualizado = acumulado/100;
  acumulado=0;
  Serial.print("Valor en pF: ");
  Serial.println(actualizado);
  delay(100);
}
```

Conductividad eléctrica del agua

En este experimento vamos a medir la conductividad eléctrica del agua del grifo, destilada y salada (con diferentes sales disueltas) mediante un sensor TDS (Total Dissolved Solids) de conductividad del agua que mide cuántos miligramos de sólidos están disueltos en 1 litro de agua (ppm).



destilada · grifo · salada

Cristina Velasco (Velasco, 2023) en su trabajo de fin de carrera, *Guardianes del Aire y del Agua: Estaciones de medición de calidad de aire y agua*, utiliza este sensor para medir la calidad del agua con suficiente éxito como para intentarlo también nosotros.

Fundamento físico

El agua destilada es un mal conductor de la electricidad porque en ella hay muy pocos objetos capaces de transportar carga aunque se autoionize de forma espontánea:

$$H2O \rightleftharpoons H^+ + OH^-$$

Ese equilibrio está muy desplazado hacia la izquierda: a 25 °C solo hay del orden de 10^{-7} mol/L de cada ion. Hay portadores de carga, sí, pero poquísimos. De ahí que su conductividad sea minúscula (del orden de 0,05 µS/cm) y que un TDS-metro comercial marque prácticamente cero.

Cuando disolvéis una sal en agua, el cristal se rompe y los iones se separan, rodeados de moléculas de agua (hidratados). Para la sal común (cloruro de sodio):

$$NaCl(s) \rightarrow Na^+(aq) + Cl^-(aq)$$

Si nos fijamos en la estequiometría: cada mol de NaCl que se disuelve aporta dos moles de portadores de carga, un catión Na^+ y un anión Cl^-. Por eso el agua con sal conduce órdenes de magnitud mejor que el agua destilada, y por eso la conductividad crece de forma aproximadamente proporcional a la concentración de sal disuelta: el doble de sal, el doble de portadores, el doble de conductividad (mientras la disolución siga siendo diluida).

Una disolución que conduce la corriente porque contiene iones de un electrolito. El agua potable, el agua de mar y el agua de un acuario son disoluciones electrolíticas débiles o fuertes, según cuánta sal mineral lleven disuelta.

En un cable de cobre, la corriente eléctrica es un desplazamiento de electrones libres, de manera que la mayor parte de la materia (los núcleos atómicos) se queda quieta y solo se mueven las cargas negativas. En una disolución iónica el mecanismo es completamente distinto: la carga la transporta la propia materia. Los cationes (cargas positivas, como Na^+) migran hacia el electrodo negativo o cátodo, y los aniones (cargas negativas, como Cl^-) migran hacia el electrodo positivo o ánodo. Los dos flujos suman: la corriente total es la suma del transporte de cargas positivas en un sentido y negativas en el otro.

La consecuencia práctica es que para medir la conductividad de un líquido hay que sumergir dos electrodos y analizar cómo se deja atravesar por esa migración iónica. Esa es exactamente la función de la sonda del sensor TDS.

Qué son los sólidos disueltos totales (TDS)

TDS son las siglas inglesas de Total Dissolved Solids, sólidos disueltos totales. Se expresan en miligramos por litro (mg/L), unidad que en disoluciones acuosas diluidas coincide numéricamente con las partes por millón:

$$1 \text{ ppm} \equiv 1 \text{ mg/L}$$

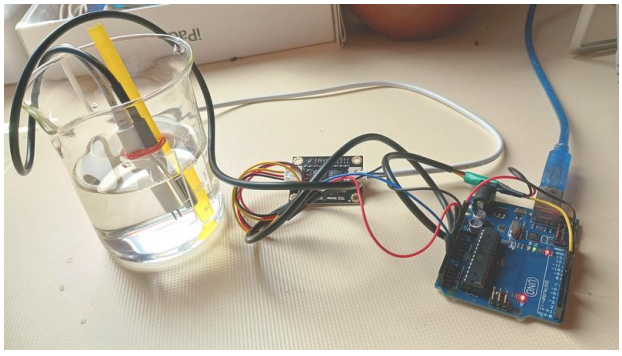
El TDS-metro no pesa sólidos: mide conductividad y aplica un factor de conversión. El aparato no tiene ninguna balanza dentro; lo único que hace es medir la facilidad con que la disolución conduce la corriente y el código multiplica por un número. Ese número es el factor de conversión, k:

TDS (ppm) = k · Ce (µS/cm),

Sal / Sustancia	Fórmula	Factor de conversión (TDS = Ce × Factor)
Cloruro de Sodio	NaCl	0.47 - 0.50
Cloruro de Potasio	KCl	0.50 - 0.53
Cloruro de Calcio	$CaCl_2$	0.50 - 0.55
Nitrato de Potasio	KNO_3	0.62 - 0.64
Bicarbonato de Sodio	$NaHCO_3$	0.65 - 0.67
Carbonato de Calcio	$CaCO_3$	0.66 - 0.71
Sulfato de Magnesio	$MgSO_4$	0.70 - 0.75
Mezcla "442"*	Varias	0.65 - 0.70

*La mezcla 442 es un estándar industrial compuesto por 40% de sulfato de sodio, 40% de bicarbonato de sodio y 20% de cloruro de sodio que podría asociarse al “agua natural”.

El valor k ≈ 0,50 para el NaCl no es una constante de la naturaleza, es una convención para sales de tipo NaCl. Distintas sales (y distintos iones) transportan distinta cantidad de carga por unidad de masa, de modo que una disolución de sulfato de calcio y otra de cloruro de sodio con idéntica masa de sólido disuelto marcarán TDS diferentes.



La conductividad se mide en siemens/metro (o en nuestro caso en µS/cm) El siemens (S) mide la conductancia eléctrica, que equivale a la inversa del ohmio (Ω^{-1} o mho), ya que la conductancia es inversa a la resistencia eléctrica. Es decir, una disolución tiene una conductancia de un siemens cuando un voltio de potencial produce una corriente de un amperio.

Conductividad (µS/cm) que genera una cantidad específica de sal en agua (ppm) a 25 °C

Concentración (ppm o mg/L)	NaCl	KCl	Mezcla "442"
10 ppm	~ 21	~ 19	~ 15
50 ppm	~ 105	~ 96	~ 72
100 ppm	~ 210	~ 192	~ 143
500 ppm	~ 1020	~ 950	~ 714
1000 ppm	2000	1895	1450
2000 ppm	4000	3850	3000

Por qué hay que compensar la temperatura

La conductividad iónica es un fenómeno de transporte, y todo transporte en un líquido está frenado por la viscosidad. Al calentar el agua, la viscosidad disminuye, los iones se mueven con más facilidad y la disolución conduce más. En aguas naturales ese efecto es aproximadamente del 2 % por cada grado centígrado: A 30 °C el mismo agua marca un valor más alto que a 20 °C.

La variación es de un 20 % entre 15 °C y 25 °C: muchísimo más que el error de la electrónica.

Por eso todo el mundo (fabricantes, laboratorios, normativa) refiere las medidas a una temperatura de referencia única: T_R= 25,0 °C, con un coeficiente térmico α_t = 0,02 (es decir, 2 %/°C). La compensación se aplica sobre la tensión medida, no sobre el resultado final, con esta expresión:

$$V_{25} = V_{medido} \div \left(1 + 0{,}02 \cdot (T_{medida} - 25)\right)$$

Se divide porque a temperatura superior a 25 °C el agua conduce más, y por tanto el módulo entrega más tensión de la que entregaría a 25 °C. Para referir esa lectura al patrón de 25 °C hay que descontar el exceso, y eso se hace dividiendo por un factor mayor que la unidad.

Si la muestra está a 20 °C (más fría, menos móvil, menos tensión), el factor del denominador es menor que 1 y la división aumenta la tensión: corregimos hacia arriba la lectura empobrecida por el frío. La ecuación funciona igual en ambos sentidos, que es

justo lo que se espera de una buena ley de compensación.

Por qué el módulo del sensor excita los electrodos con corriente alterna (AC)

Aquí está la razón física que justifica comprar un módulo y no resolverlo con dos cables y una pila. Si excitásemos los electrodos con corriente continua pura (DC) ocurrirían tres cosas malas a la vez:

Polarización de la doble capa. En la superficie de cada electrodo se forma una fina capa de iones de signo contrario (la doble capa eléctrica), de manera que la tensión medida no se estabiliza, deriva lentamente y la lectura se va perdiendo con el tiempo.

Electrólisis. Por encima de aproximadamente 1,23 V (más el sobrepotencial de cada electrodo) el agua empieza a descomponerse: aparecen burbujas de O2 en el ánodo y de H2 en el cátodo. Las burbujas se adhieren al electrodo y modifican la superficie de contacto.

Corrosión del electrodo. La corriente continua en un solo sentido disuelve progresivamente el metal del ánodo y contamina la muestra con iones metálicos que además falsean la medida.

La excitación en corriente alterna evita las tres: al invertir el sentido muchas veces por segundo, la doble capa se descarga en cada semiciclo, las reacciones de electrólisis no llegan a progresar y no se acumula transporte neto de metal. El módulo TDS que vamos a usar ya incorpora internamente toda esta electrónica de excitación alterna y de acondicionamiento: no hace falta montar ningún circuito externo. Simplemente, no lo alimentéis con una fuente que él no gestione.

Material necesario

Componente	Especificación
Placa microcontroladora	Arduino UNO R3 (5 V, ADC 10 bits) (o ESP32 (3,3 V, ADC 12 bits))
Módulo sensor TDS analógico	Tipo Gravity/DFRobot SEN0244, alimentación 3,3–5,0 V, consumo ~6 mA, salida analógica 0–2,3 V
Sonda TDS	Dos electrodos de acero inoxidable con línea de inmersión máxima marcada, conector jack XH
Sensor de temperatura DS18B20	Encapsulado TO-92, −55 a +125 °C, precisión ±0,5 °C en −10 a +85 °C
Resistencia de pull-up	4,7 kΩ (obligatoria entre DQ y VCC del bus 1-Wire)
Condensadores cerámicos de desacoplo	2 de 100 nF
Protoboard y cables Dupont	Macho-hembra y macho-macho
Recipientes de vidrio	250 mL
Probeta o matraz aforado	Para enrasar las disoluciones patrón
Balanza de precisión	Resolución 0,01 g
Cloruro de sodio (sal común)	NaCl
Agua destilada	Para patrones y para el enjuague final
Termómetro de referencia (opcional)	Para contrastar la lectura del DS18B20

Esquema de conexiones

Pin del componente	Pin del Arduino UNO R3	Color de cable sugerido
Módulo TDS VCC	5V	Rojo
Módulo TDS GND	GND	Negro
Módulo TDS AOUT	A0	Amarillo
DS18B20 VCC (pin 3)	5V	Rojo
DS18B20 GND (pin 1)	GND	Negro
DS18B20 DQ (pin 2)	D2	Verde
Resistencia 4,7 kΩ Entre DQ y 5V del DS18B20		
Condensador 100 nF Entre VCC y GND del módulo TDS (optativo)		
Condensador 100 nF Entre VCC y GND del DS18B20 (optativo)		

Hay que usar preferentemente un Arduino UNO ya que el sensor TDS tiene una salida de datos analógica.

En el sensor DS18B20 se necesita una resistencia (Pull-up) de 4,7 kΩ en el bus 1-Wire (entre los 5 V y la entrada de datos) ya que sin la resistencia entre DQ y VCC, el bus nunca alcanza el nivel ALTO de

reposo y no hay comunicación posible: el sensor simplemente no se encuentra.

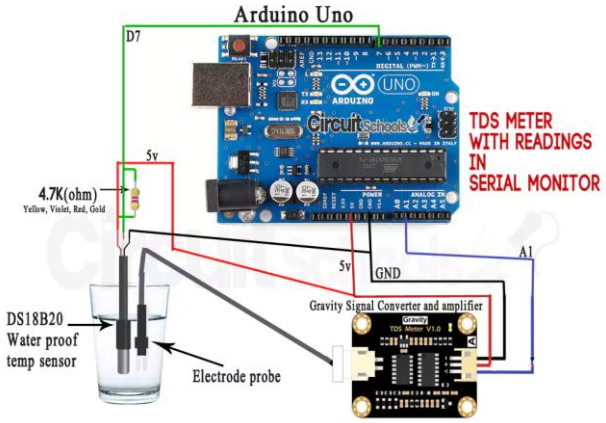


Imagen de Circuits Schools (2025)

Puede ser interesante colocar un condensador cerámico de 100 nF entre VCC y GND de cada sensor puesto lo más cerca posible de sus pines, ya que absorberán los picos de corriente que el sensor demanda y evitarán que ese ruido viaje por los cables hasta el conversor analógico.

No se ha de mojar el conector de la sonda, solo los electrodos, y nunca por encima de la línea de inmersión máxima marcada.

Procedimiento experimental

Preparación de las disoluciones patrón por pesada. Trabajad siempre con agua destilada como disolvente.

Disolución patrón	Masa de NaCl pesada	Concentración nominal	TDS nominal
Agua destilada	0,00 g	0 mg/L	0 ppm
Patrón 1	0,25 g en 1 L	250 mg/L	250 ppm
Patrón 2	0,50 g en 1 L	500 mg/L	500 ppm
Patrón 3	1,00 g en 1 L	1000 mg/L	1000 ppm

Cargad el programa con el IDE de Arduino, y abrid el monitor serie a 9600 baudios. Al arrancar debe aparecer una línea con la temperatura, la tensión, la tensión compensada, la conductividad y el TDS.

Si en el lugar de la temperatura aparece el aviso de que no se encuentra el DS18B20, el programa sigue midiendo con la temperatura fija de 25 °C, de modo que el experimento no se interrumpe pero la compensación térmica queda desactivada de hecho.

Sumergid la sonda en agua destilada hasta justo la línea de inmersión máxima, sin mojar el conector, y esperad al menos 30 segundos sin tocar nada ya que la doble capa del electrodo necesita establecerse y la primera lectura siempre está algo alta. Ese tiempo de cebado no se debe saltar.

Para cada patrón, agitad muy suavemente la sonda para renovar la capa de líquido en contacto con los electrodos (sin formar remolinos ni burbujas) y esperad unos segundos. Anotad la tensión V leída, la temperatura T y el TDS calculado.

Entre un patrón y el siguiente, sacad la sonda, enjuagadla con agua destilada y secadla suavemente con papel absorbente limpio. No basta con enjuagar: si arrastráis una gota del patrón anterior, esa gota contiene sal y contaminará el siguiente patrón. En una disolución de 250 ppm, una gota contaminante puede falsear la medida varios puntos porcentuales.

Repetid cada medida al menos dos veces, volviendo siempre al patrón desde el enjuague. Comparad las dos lecturas: si difieren más de un 5 %, hay un problema de estabilización, de burbujas o de temperatura.

Tabla de registro de datos

Anotad todas las medidas en una tabla como la siguiente.

La columna V25 es la tensión corregida a 25,0 °C con la expresión $V25 = V \div \left(1 + 0{,}02 \cdot (T - 25)\right)$.

La desviación se calcula respecto al TDS nominal de la disolución patrón, en valor absoluto:
desviación (%) = (TDS medido − TDS nominal) x 100 ÷ TDS nominal

Disolución patrón	Masa pesada (g/L)	C nominal (ppm)	T (°C)	V medida (V)	V25 (V)	Conductividad µS/cm	TDS calculado (ppm)
Agua destilada	0,000	0					
Agua del grifo			26,3	0,557	0,543	441,1	(441,1x0,7) = 308,7
Patrón 1	0,250	250					
Patrón 2	0,500	500					
Patrón 3	1,000	1000					
Repetición de patrón 1	0,250	250					
Repetición de patrón 2	0,500	500					
Repetición de patrón 3	1,000	1000					

Ampliación

Si se duplica la masa de sal disuelta sin cambiar el volumen se puede pensar que aumentarán al doble los portadores de carga, pero ¿qué le ocurrirá a la tensión leída a temperatura constante? ¿Por qué la relación no es exactamente lineal en todo el rango de concentraciones?

El factor k no es una constante física. El TDS medido casi nunca coincide exactamente con la masa de sal pesada. Explicad por qué: el factor k ≈ 0,50 es una convención del fabricante para sales de tipo NaCl, no una constante de la naturaleza. Cada ion transporta distinta cantidad de carga por unidad de masa; la movilidad del Na^+ no es la del Ca^{2+} ni la del $SO4^{2-}$. Si se mide una disolución de sulfato de calcio, ¿con que valores nos encontraremos?

Haced los cálculos de la compensación por la temperatura. Tomad la lectura de un patrón tomada a temperatura distinta de 25 °C y calculad a mano cuánto vale V25 y comparad los TDS resultantes. ¿Cuántos ppm de diferencia hay? ¿Justifica ese error la corrección térmica que hemos introducido en el código?

Código

```
/*
 * MEDIDOR DE CONDUCTIVIDAD ELÉCTRICA (µS/cm) Y TDS (ppm)
 * Placa: Arduino UNO
 * Sensor conductividad/TDS: Pin Analógico A0
 * Sensor temperatura (DS18B20): Pin Digital 2
 */

#include <OneWire.h> // Librería de Jim Studt
#include <DallasTemperature.h> //Librería para el sensor de temperatura de Miles Burton

// 1. CONFIGURACIÓN DE PINES Y HARDWARE
const int PIN_TDS = A0;          // Pin donde se conecta la señal analógica del módulo TDS
const int PIN_DS18B20 = 2;       // Pin donde se conecta el cable de datos del sensor de temperatura

// Constantes físicas del Arduino UNO
const float V_FONDO_ESCALA = 5.0; // El Arduino UNO funciona a 5V (referencia del ADC)
const int ADC_MAX = 1023;        // Resolución del conversor analógico-digital (10 bits = 0 a 1023)

// 2. PARÁMETROS DEL EXPERIMENTO
const int N_MUESTRAS = 25;       // Tomaremos 25 lecturas seguidas para hacer una media y eliminar ruido
const float T_REFERENCIA = 25.0;  // Temperatura estándar a la que se normaliza la conductividad (25 °C)
const float ALPHA_TERMICO = 0.02; // El agua varía su conductividad un 2% por cada grado centígrado

// Configuración de las librerías para el sensor de temperatura
OneWire bus(PIN_DS18B20);
DallasTemperature termometro(&bus);

void setup() {
  // Iniciamos la comunicación con el ordenador a 9600 baudios
  Serial.begin(9600);

  // Encendemos y configuramos el sensor de temperatura
  termometro.begin();
  termometro.setResolution(12); // Máxima precisión térmica (resolución de 0,0625 °C)

  Serial.println("Iniciando medidor de Conductividad y TDS...");
}

void loop() {

  // PASO 1: LECTURA Y FILTRADO DEL SENSOR TDS
  long suma = 0;

  // Tomamos varias muestras rápidamente para estabilizar la lectura
  for (int i = 0; i < N_MUESTRAS; i++) {
    suma += analogRead(PIN_TDS);
    delay(2); // Pequeña pausa de 2 ms para que el conversor del Arduino descanse
  }

  // Calculamos la media matemática de las muestras tomadas
  float promedio_adc = suma / (float)N_MUESTRAS;

  // Convertimos ese número crudo (de 0 a 1023) a un valor real en Voltios (de 0.0V a 5.0V)
```

```
  float v_medido = promedio_adc *
(V_FONDO_ESCALA / (ADC_MAX + 1));

  // PASO 2: LECTURA DE LA TEMPERATURA
  termometro.requestTemperatures(); // Pedimos al
sensor que mida (este proceso tarda unos 750 ms)
  float t_leida = termometro.getTempCByIndex(0); //
Guardamos los grados Celsius

  float t_medida = T_REFERENCIA;    // Valor por
defecto (25 °C) en caso de que el sensor falle

  // Protección contra errores: Si el sensor se
desconecta, la librería devuelve -127 °C.
  // Usar -127 °C rompería las matemáticas, así que
solo usamos la lectura si es mayor a -100 °C.
  if (t_leida > -100.0) {
    t_medida = t_leida; // La lectura es válida, la
usamos
  } else {
    Serial.println("AVISO: DS18B20 no detectado.
Asumiendo 25 °C.");
  }

  // PASO 3: COMPENSACIÓN TÉRMICA
  // Como el agua caliente conduce mejor la
electricidad, calculamos qué voltaje
  // habría devuelto el sensor si el agua estuviese
exactamente a 25 °C.
  float v25 = v_medido / (1.0 + ALPHA_TERMICO *
(t_medida - T_REFERENCIA));

  // PASO 4: CÁLCULO DE CONDUCTIVIDAD Y
TDS
  // Fórmula del fabricante (polinomio) que convierte
el voltaje compensado a
  // Conductividad Eléctrica (CE) medida en
microSiemens por centímetro (µS/cm)
  float ce = 133.42 * v25 * v25 * v25 - 255.86 * v25 *
v25 + 857.39 * v25;

  // El TDS (Sólidos Disueltos) se estima
multiplicando la conductividad por 0.5
  // (Factor estándar para agua común basándose
en sales de sodio)
  float tds = ce * 0.5;

  // PASO 5: MOSTRAR RESULTADOS EN
PANTALLA
  Serial.print("Temp: ");
  Serial.print(t_medida, 2);
  Serial.print(" C  |  ");

  // Mostramos el voltaje real leído por el pin A0
  Serial.print("V. Medido: ");
  Serial.print(v_medido, 3);
  Serial.print(" V  |  ");

  // Mostramos el voltaje tras aplicar la
compensación por temperatura
  Serial.print("V. a 25C (V25): ");
  Serial.print(v25, 3);
  Serial.print(" V  |  ");

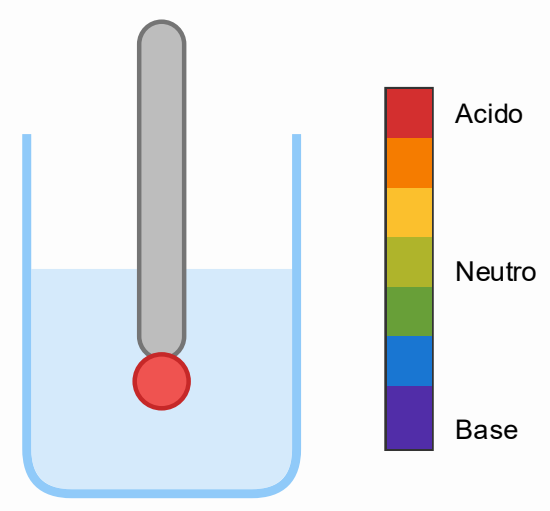
  Serial.print("Conductividad: ");
  Serial.print(ce, 1);
  Serial.print(" uS/cm  |  ");

  Serial.print("TDS: ");
  Serial.print(tds, 1);
  Serial.println(" ppm");

  // Pausa obligatoria. La lectura del sensor de
temperatura a 12 bits ya tarda 750 ms,
  // con este delay de 1000 ms garantizamos que el
sistema no se sature.
  delay(1000);
}
```

Medida del pH de disoluciones acuosas

Medir el pH de diferentes disoluciones acuosas utilizando un kit de pH para Arduino, comparando los valores obtenidos con los teóricos y analizando la relación entre la concentración de iones H^+ y el valor del pH (Ramírez, 2020b).



Fundamento físico

El pH es una medida de la acidez o basicidad de una disolución acuosa, definido como el logaritmo decimal negativo de la actividad de los iones hidronio:

$$\mathrm{pH} = -\log_{10}[\mathrm{H}^+]$$

En una disolución neutra a 25 °C, $[H^+] = 10^{-7}$ M y pH = 7. Las disoluciones ácidas tienen pH < 7 y las básicas pH > 7.

La escala de pH es logarítmica, lo que significa que una diferencia de una unidad de pH corresponde a un cambio de un orden de magnitud en la concentración de H^+.

El electrodo de pH mide la diferencia de potencial entre un electrodo de referencia (plateado) y un electrodo sensible a H^+ (vidrio), diferencia que es proporcional al pH según la ecuación de Nernst.

Materiales y montaje experimental

Necesitaréis una placa Arduino UNO R3 (IMPORTANTE: el pH-metro no funciona correctamente con ESP32), un kit de pH para Arduino (sonda con conector BNC + placa PH-4502C), una protoboard, cables Dupont, varios recipientes pequeños (vasos de precipitados), agua destilada, disoluciones patrón de pH conocido (pH 4, 7 y 10 para calibración), vinagre, zumo de limón, agua del grifo, bicarbonato sódico disuelto y lejía diluida.



Conectad la placa PH-4502C al Arduino UNO: el pin VCC de la placa a 5 V, GND a GND y el pin Po (salida analógica) a la entrada A0 del Arduino.

Conectad la sonda de pH al conector BNC de la placa. Fijad la sonda a un soporte vertical para que los electrodos queden sumergidos en las disoluciones sin tocar el fondo del recipiente. Antes de cada medida, lavad la sonda con agua destilada y secadla suavemente con papel absorbente (sin frotar el bulbo de vidrio).

Las sondas de pH necesitan ser tratadas con cuidado, pero incluso así después de un año y medio o dos ya no funcionan bien y deben sustituirse.
Conservación de los electrodos de pH

Si ha comprado un pH-metro y al retirar la tapa protectora de la punta aparecen depósitos de sal no es necesario alarmarse, es normal y desaparecerán al limpiar el electrodo con agua destilada.

Para almacenar los electrodos de pH se debe utilizar siempre la tapa o capuchón protector lleno de líquido, ya que la membrana del electrodo nunca debe quedar seca, y mantenerlos a una temperatura ambiente de entre 15 ºC a 25 ºC. Como líquido se utilizará una solución de almacenamiento de electrodos, normalmente cloruro de potasio de 3 mol/L, o solución tampón de pH 4.01. ¡Nunca se ha de utilizar agua destilada!



Procedimiento

En primer lugar, conectad la placa al ordenador (sin la sonda d y cargad el código de calibración. Es necesario calibrar el sensor sumergiendo la sonda en la disolución patrón de pH 7 y anotando el valor leído. Repetid con los patrones de pH 4 y pH 10. Con estos tres puntos se puede establecer la recta de calibración (voltaje frente a pH).

Se observa que en el circuito electrónico existen dos potenciómetros de color azul. El más cercano al conector BNC de la sonda es el que regula el offset, el otro es el del límite de pH.



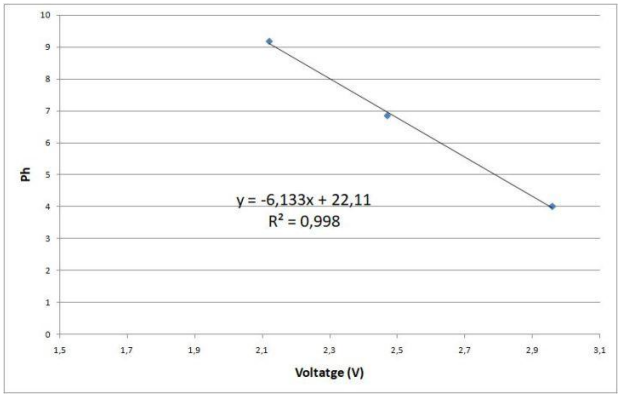
Offset: El rango de medidas de la sonda de pH oscila entre valores negativos y positivos y el 0 representa un pH de 7,0. Para poder utilizar la sonda con Arduino, el circuito electrónico modifica el valor medido por la sonda y envía sólo valores positivos de tensión.

Límite de pH: Este potenciómetro, que no utilizaremos, es para establecer un valor límite del circuito del sensor de pH que hace que el LED rojo se encienda y la señal de pin Do se ponga en ON.
Para medir bien los valores de pH de las disoluciones patrón se deben hacer dos cosas. La primera es cortocircuitar con un cable la parte interna del conector BNC con la exterior y modificar el valor que da el pin Po, que es el que se ve en el monitor serie del IDE de Arduino, ajustando con un destornillador el potenciómetro Offset para que sea 2,5 V.

La segunda es medir el voltaje que da el sensor de pH para dos o tres valores de referencia de pH, una vez que se ha conectado la sonda y calcular la línea de calibración. Lo más recomendable es utilizar soluciones de calibración en sobre. Estas soluciones se venden (a buen precio, 4€ el lote) de diferentes valores aunque los más comunes son pH 4.01, pH 6.86 y pH 9.18.

Utilizando sobres con pH 4,01, pH 6,86 y pH 9,18 se pueden obtener voltajes de 2,96, 2,47 y 2,12 respectivamente, por ejemplo. El sensor es lineal por lo que la fórmula para calcular el pH es y = mx + b, que puede obtenerse de la ecuación de la línea de tendencia al representar gráficamente los valores en una hoja de cálculo.

El resultado que da es y = -6,13 x + 22,11, donde y = pH y x = voltaje, y es la ecuación que he introducido en el código del proyecto. Cada uno debe hacer lo mismo con el sensor y la sonda de que disponga y modificar el código para que se ajuste a su maquinaria.



A continuación, ya se puede medir el pH de cada una de las disoluciones problema (vinagre, limón, agua del grifo, bicarbonato, lejía) introduciendo la sonda y esperando a que la lectura se estabilice (unos 30 segundos).

Resultados esperados

Se puede esperar pH ácido para el vinagre (pH ≈ 2-3) y el zumo de limón (pH ≈ 2-3), pH aproximadamente neutro para el agua del grifo (pH ≈ 6-8), pH básico para la disolución de bicarbonato (pH ≈ 8-9) y pH fuertemente básico para la lejía diluida (pH ≈ 11-12). Las lecturas pueden variar ±0,5

unidades de pH debido a la precisión limitada del sensor.

Análisis

Para cada disolución, calculad la concentración de iones H^+ a partir del pH medido:

$$[H^+] = 10^{-pH}$$

Ordenad las disoluciones de menor a mayor acidez y comparad con vuestras expectativas basadas en el conocimiento cotidiano de esas sustancias. Discutid las fuentes de error: calibración del electrodo, temperatura de la disolución (el pH depende de la temperatura), tiempo de estabilización y posible deriva del electrodo.

Ampliación

Realizad una valoración ácido-base: añadid gota a gota una base (NaOH diluido) a una disolución de vinagre y registrad el pH en función del volumen añadido. Representad la curva de valoración y determinad el punto de equivalencia.

Código de calibración

```
//Calibrar sensor pH
//Sensor PH-4502C
//Proyecto basado en entrada de David Cervantes Caballero
//https://scidle.com/es/como-usar-un-sensor-de-ph-con-arduino/

const int analogInPin = A0;
int sensorValue = A0;
unsigned long int avgValue;
float b;
int buf[10],temp;

void setup()
{
 Serial.begin(9600);
}
void loop()
{
  //Promediar diez medidas
 for(int i=0;i<10;i++)
 {
  buf[i]=analogRead(analogInPin);
  delay(10);
 }
 for(int i=0;i<9;i++)
 {
  for(int j=i+1;j<10;j++)
  {
   if(buf[i]>buf[j])
   {
    temp=buf[i];
    buf[i]=buf[j];
    buf[j]=temp;
   }
  }
 }
 avgValue=0;
 for(int i=2;i<8;i++)
 avgValue+=buf[i];

 float pHVol=(float)avgValue*5.0/1024/6;

 Serial.println(pHVol);

 delay(2000);
}
```

Código de medidas

```
//Medir pH
//Sensor PH-4502C
//Proyecto basado en la entrada de David Cervantes Caballero
//https://scidle.com/es/como-usar-un-sensor-de-ph-con-arduino/

const int analogInPin = A0;
int sensorValue = A0;
unsigned long int avgValue;
float b;
int buf[10],temp;

void setup()
{
 Serial.begin(9600);
}
void loop()
{
  //Promediar diez medidas
 for(int i=0;i<10;i++)
 {
  buf[i]=analogRead(analogInPin);
  delay(10);
 }
 for(int i=0;i<9;i++)
 {
  for(int j=i+1;j<10;j++)
```

```
 {
  if(buf[i]>buf[j])
  {
   temp=buf[i];
   buf[i]=buf[j];
   buf[j]=temp;
  }
 }
}
avgValue=0;
for(int i=2;i<8;i++)
avgValue+=buf[i];

float pHVol=(float)avgValue*5.0/1024/6;
float phValue = -6.13 * pHVol + 22.11;

Serial.print("pH = ");
Serial.println(phValue);

delay(2000);
}
```

Anexos

Glosario de términos

1-Wire: véase OneWire.

A3144: sensor de efecto Hall con salida digital y comparador interno, detecta la presencia de un campo magnético superando un umbral ajustable; se comercializa en módulo como KY-003.

Aceleración angular (α): variación de la velocidad angular por unidad de tiempo, medida en rad/s²; en el estudio del movimiento circular describe el frenado del disco por el rozamiento.

ACS712: sensor de corriente basado en el efecto Hall, mide corriente continua y alterna hasta 5 A, 20 A o 30 A según el modelo, con aislamiento galvánico.

ADC (Conversor Analógico-Digital): circuito electrónico que convierte una señal analógica (tensión continua) en un valor digital (número entero) que el microcontrolador puede procesar.

AGC (Automatic Gain Control): control automático de ganancia del amplificador MAX9814, que ajusta la amplificación según el nivel sonoro para captar desde susurros hasta sonidos intensos sin saturación.

AHT20 / AHT21: sensores digitales de temperatura y humedad relativa con calibración de fábrica (OTP) y compensación de temperatura, comunicación I²C, más precisos y estables que el DHT22.

AHT2X: familia de sensores que agrupa los modelos AHT20 y AHT21.

Amortiguamiento: disipación progresiva de la energía de un oscilador por una fuerza proporcional a la velocidad (rozamiento viscoso), que reduce su amplitud de forma exponencial con el tiempo.

Amplificador de instrumentación: amplificador diferencial de muy alta ganancia y elevado rechazo al modo común, integrado en el HX711, que amplifica los milivoltios diferenciales que entrega una célula de carga.

Amplificador operacional: amplificador de ganancia muy elevada usado como bloque de acondicionamiento de señal (amplificación, comparación, filtrado); el SS49E lo incorpora internamente para dar salida lineal.

analogRead(): función de Arduino que realiza una conversión con el ADC y devuelve un entero entre 0 y 1023 (UNO, 10 bits) o 0 y 4095 (ESP32, 12 bits) proporcional a la tensión de entrada.

APDS9960: sensor óptico multifunción que integra detección de gestos, proximidad, color RGB e intensidad de luz ambiental, comunicación I²C.

Arduino: plataforma de hardware y software libre basada en microcontroladores, diseñada para facilitar el prototipado de proyectos electrónicos.

Arduino IDE: entorno de desarrollo oficial de Arduino (editor, compilador, gestor de tarjetas y librerías) con el que se escriben, compilan y cargan los sketches.

Arduino Nano 33 BLE Sense: placa de formato reducido basada en el nRF52840 que integra acelerómetro, giroscopio, magnetómetro, micrófono digital, sensor de luz/color/gestos, humedad, presión y temperatura, con conectividad BLE.

Arduino UNO: placa de referencia de la familia Arduino basada en el ATmega328P, con 14 pines digitales (6 PWM), 6 entradas analógicas y ADC de 10 bits; sus entradas analógicas son lineales de 0 a 5 V.

Arduino UNO R3: revisión más extendida de la placa Arduino UNO, utilizada en el libro para experimentos que requieren entrada analógica pura.

Arquímedes (principio de): todo cuerpo sumergido en un fluido experimenta un empuje vertical ascendente igual al peso del fluido desalojado.

Atenuación: disminución de la intensidad de una señal (luz o tensión) al atravesar un medio o con la distancia; determina la medida en condiciones de muy baja iluminancia.

ATmega328P: microcontrolador de 8 bits de Microchip (16 MHz, 32 KB de memoria flash y 2 KB de RAM) empleado en el Arduino UNO.

attachInterrupt(): función de Arduino que asocia una rutina (ISR) al flanco de un pin digital,

permitiendo medir pulsos de fotopuertas con latencia de microsegundos.

Balanza: instrumento de laboratorio para medir masa; se emplea como patrón para calibrar la célula de carga (masa conocida → fuerza en newtons).

Barrera óptica: dispositivo formado por un emisor y un receptor de luz enfrentados que detecta el paso de un objeto al interrumpirse el haz; base de la medida de tiempos.

Baudios: unidad de velocidad de transmisión del puerto serie (símbolos por segundo); 9600 baudios es el valor habitual empleado en el libro.

BH1750: sensor digital de iluminancia (lux) con fotodiodo de respuesta espectral similar a la del ojo humano, ADC de 16 bits y comunicación I²C.

BLE (Bluetooth Low Energy): versión del protocolo Bluetooth optimizada para bajo consumo energético, ideal para dispositivos alimentados por pilas.

BLE2902: descriptor estándar de BLE que habilita las notificaciones de una característica; se declara en los sketches para enviar datos a Phyphox o Science Journal.

Bluetooth: tecnología de radio de corto alcance en la banda de 2,4 GHz que permite la comunicación inalámbrica entre la placa y un teléfono móvil.

BME280: versión mejorada del BMP280 que además mide humedad relativa.

BMP280: sensor de presión atmosférica y temperatura fabricado por Bosch, comunicación I²C o SPI.

Boyle (ley de): a temperatura constante, el producto presión × volumen de una masa de gas ideal permanece constante ($P \cdot V = \text{cte}$); también llamada ley de Boyle-Mariotte.

Bucle: estructura de control que repite un bloque de instrucciones; en Arduino, la función loop() es un bucle infinito que ejecuta la lógica de medida.

Buffer: aquí, disolución tampón de pH conocido y estable (pH 4,01 / 7,00 / 10,01) empleada como patrón en la calibración del pH-metro.

Bus I²C: bus de comunicación serie síncrono que utiliza dos líneas (SCL y SDA) para conectar múltiples dispositivos usando direcciones únicas.

Byte: unidad de información digital compuesta por 8 bits, equivalente a un carácter alfanumérico.

C++: lenguaje de programación de los sketches de Arduino y ESP32; tipado, compilado y con estructuras de control, funciones y variables de distintos tipos.

Caída libre: movimiento vertical de un cuerpo bajo la única acción de la gravedad ($a \approx 9{,}81\ m/s^2$) cuando el rozamiento con el aire es despreciable.

Calor: energía en tránsito entre dos cuerpos a distinta temperatura; se mide en julios (J) o en calorías (1 cal = 4,184 J).

Calor específico: calor necesario para elevar un grado la temperatura de la unidad de masa de una sustancia, en J/(kg·K); el del agua es 4180 J/(kg·K).

Calorímetro: recipiente térmicamente aislado que permite medir calores específicos y latentes minimizando las pérdidas al entorno.

Calor latente: calor absorbido o cedido por unidad de masa durante un cambio de fase sin variación de temperatura; el de fusión del hielo es 334 kJ/kg.

Campo eléctrico (E): región del espacio en la que una carga eléctrica experimenta una fuerza; su dirección define la oscilación de la luz polarizada.

Campo magnético (B): magnitud vectorial que describe la fuerza ejercida sobre cargas en movimiento y sobre materiales magnéticos; se mide en teslas (T).

Capacidad calorífica (C): calor necesario para elevar un grado la temperatura de un cuerpo completo, $C = m \cdot c$, en J/K; su cociente con la resistencia térmica da la constante de tiempo del sensor.

Carga eléctrica (Q): propiedad fundamental de la materia que origina las interacciones electromagnéticas; se mide en culombios (C).

Caudal: volumen de fluido que atraviesa una sección por unidad de tiempo; el YF-S201 lo obtiene contando pulsos de efecto Hall (≈ 4,5 Hz por L/min).

CdS (sulfuro de cadmio): semiconductor con fotoconductividad empleado en las LDR; su resistencia disminuye al aumentar la luz incidente.

Célula de carga: transductor que convierte una fuerza mecánica en una señal eléctrica mediante galgas extensométricas en configuración de puente de Wheatstone.

Cero absoluto: temperatura teórica mínima (0 K = −273,15 °C) a la que cesa el movimiento térmico de las partículas.

CH340: circuito integrado convertidor USB-serie de las placas Wemos D1 R32, cuyo driver debe instalarse en Windows para que aparezca el puerto COM.

Charles (ley de): a presión constante, el volumen de una masa de gas es directamente proporcional a su temperatura absoluta (V/T = cte).

Cinemática: parte de la mecánica que describe el movimiento (posición, velocidad, aceleración y tiempo) sin atender a las causas que lo producen.

Coeficiente de amortiguamiento (b): constante que cuantifica la fuerza de rozamiento viscoso proporcional a la velocidad, F = −b·v, medida en N·s/m.

Coeficiente de correlación (r de Pearson): número entre −1 y 1 que mide el grado de relación lineal entre dos variables, como temperatura y humedad relativa.

Coeficiente de determinación (R^2): fracción de la varianza de los datos explicada por la recta de ajuste; $R^2 = 1$ indica ajuste perfecto.

Coeficiente de restitución (e): cociente entre la velocidad de separación y la de aproximación en un choque; para un rebote se obtiene como $e = \sqrt{(h_1/h_0)}$.

Cojín de aire: recinto cerrado de aire (bolsa o globo) cuya sobrepresión interior permite medir una masa a partir de P = F/S.

Columna de agua: altura de líquido que por hidrostática genera una presión de ≈ 9810 Pa por metro, es decir, 981 Pa cada 10 cm.

Compilador: programa que traduce el código C++ del sketch a instrucciones máquina del microcontrolador; ignora los comentarios, que no ocupan memoria.

Componente horizontal: proyección del campo magnético terrestre sobre el plano horizontal; junto con la vertical define la inclinación magnética.

Compresión: disminución del volumen de un gas al aumentar la presión; junto con la expansión permite evaluar la reproducibilidad de la ley de Boyle.

Concentración: cantidad de soluto por unidad de volumen de disolución; se expresa en mol/L (molaridad), g/L o ppm.

Condensación: cambio de fase de vapor a líquido con cesión de calor latente; a cada temperatura define la presión de vapor saturado y por tanto la humedad relativa.

Condensador: componente electrónico que almacena energía en forma de campo eléctrico, formado por dos conductores separados por un dieléctrico.

Conducción: transferencia de calor por contacto directo entre cuerpos o a través de un material, sin transporte de materia.

Conductividad: medida de la capacidad de un material para conducir corriente eléctrica, expresada en siemens por metro (S/m) o en µS/cm en disoluciones.

Constante de celda: factor geométrico de una celda de conductividad (cociente entre separación y área de los electrodos) que relaciona la conductancia medida con la conductividad de la disolución.

Constante de los gases (R): constante universal de los gases ideales, R = 8,314 J/(mol·K), presente en la ecuación de estado y en la ecuación de Nernst.

Constante de tiempo (τ): tiempo que tarda un sistema de primer orden (circuito RC, sensor térmico, LDR) en alcanzar el 63,2 % de su variación total.

Constante elástica (k): fuerza por unidad de alargamiento de un muelle, en N/m; corresponde a la pendiente de la recta F-x de la ley de Hooke.

Convección: transporte de calor por el movimiento de un fluido que, al calentarse, se expande, disminuye su densidad y asciende.

Conversor analógico-digital: véase ADC.

Corriente alterna (CA): corriente cuyo sentido y magnitud varían periódicamente; la red eléctrica española lo hace a 50 Hz.

Corriente continua (CC): corriente de sentido y magnitud constantes, como la que suministran las pilas, el puerto USB o el regulador de la placa.

Corriente eléctrica: flujo de carga eléctrica a través de un conductor, medida en amperios (A).

CPU (Central Processing Unit): unidad central de procesamiento, el "cerebro" del microcontrolador que ejecuta las instrucciones del programa.

Cronómetro: instrumento de medida de intervalos de tiempo, en el libro se sustituye por fotopuertas y marcas de tiempo del microcontrolador.

CSV (Comma-Separated Values): formato de archivo de texto que almacena datos tabulares separados por comas o punto y coma, compatible con hojas de cálculo.

Curva de calibración: representación de la señal de un sensor frente al valor conocido del patrón, cuya recta de ajuste permite convertir lecturas en magnitudes físicas.

Curva de enfriamiento: registro de la temperatura de un cuerpo frente al tiempo mientras cede calor al ambiente; su ajuste exponencial permite estimar la constante de tiempo térmica.

Datalogger: sistema de adquisición que registra datos de forma autónoma (por ejemplo placa + tarjeta microSD), sin necesidad de un ordenador conectado.

Datasheet (hoja de datos): documento técnico del fabricante con las especificaciones eléctricas, rangos de medida, patillaje y protocolo de un componente.

Debounce (antirrebote): tratamiento por hardware o software que elimina las conmutaciones falsas producidas por el rebote mecánico de un interruptor o de una barrera de luz.

Declinación magnética: ángulo entre el norte magnético y el norte geográfico en un punto; en la Península Ibérica es de pocos grados hacia el este.

Decremento logarítmico (δ): logaritmo neperiano del cociente entre dos amplitudes sucesivas, $\delta = \ln(A_n/A_{n+1}) = \gamma \cdot T$, que cuantifica la atenuación de una oscilación amortiguada.

Densidad: masa por unidad de volumen de una sustancia, en kg/m³; la del aire al nivel del mar es ≈ 1,225 kg/m³ y gobierna la variación barométrica con la altura.

Deriva: variación lenta y no deseada de la lectura de un sensor con el tiempo o con la temperatura, ajena a la magnitud medida.

Desfase: retardo temporal (diferencia de fase) entre dos señales periódicas, como el que introduce la inercia térmica entre el mínimo de iluminancia y el de temperatura en un eclipse.

Desplazamiento: vector que describe el cambio de posición de un móvil entre dos instantes de tiempo.

Desviación típica (σ): medida estadística de la dispersión de un conjunto de medidas respecto a su valor medio.

DHT11: sensor digital de temperatura y humedad de bajo coste, rango 0-50 °C y 20-90 % HR, con resolución de 1 °C y 1 % HR.

DHT22: versión mejorada del DHT11, con rango −40 a +80 °C, resolución de 0,1 °C y 0,1 % HR, pero limitado a ≈ 0,5 Hz de muestreo.

Dieléctrico: material aislante que al situarse entre las placas de un condensador se polariza y aumenta su capacidad según su permitividad relativa εr.

Diferencia de potencial: trabajo por unidad de carga entre dos puntos de un circuito; en el efecto

Hall aparece como tensión transversal proporcional al campo magnético.

Difracción: desviación de la luz al atravesar una abertura o una red de líneas del orden de su longitud de onda; es la base física de la espectroscopia.

Diodo: componente semiconductor que conduce la corriente en un solo sentido; los diodos IR emiten luz y los fotodiodos generan corriente al iluminarse.

Disco ranurado: disco opaco con ranuras equidistantes que, acoplado al eje de un motor, interrumpe cíclicamente un optoacoplador y permite medir la velocidad de rotación (encoder).

Disolvente: componente mayoritario de una disolución, que disuelve al soluto; en las experiencias del libro es el agua destilada.

Divisor de tensión: montaje de dos resistencias que reduce una tensión en una proporción fija; se usa para adaptar señales de 5 V a entradas de 3,3 V.

DOIT ESP32 DEVKIT V1: placa de desarrollo económica con ESP32, 30 pines y alimentación por USB, alternativa al Wemos D1 R32.

DS18B20: sensor digital de temperatura con protocolo 1-Wire, encapsulado en funda metálica impermeable, rango −55 a +125 °C y resolución de 12 bits.

Dual-core: arquitectura del ESP32 con dos núcleos de procesamiento a 240 MHz, que permite repartir tareas de medida rápida y de comunicación.

Ecuación de estado: relación $P \cdot V = n \cdot R \cdot T$ entre presión, volumen, cantidad de sustancia y temperatura absoluta de un gas ideal.

Ecuación de Nernst: relación entre el potencial de un electrodo y la concentración de iones; en el electrodo de vidrio del pH-metro $E = E_0 + (RT/nF) \cdot \ln[H^+]$.

Efecto Hall: aparición de una tensión transversal en un conductor por el que circula corriente y que está sometido a un campo magnético perpendicular, por desviación de los portadores de carga.

Elasticidad: propiedad de un material de recuperar su forma tras una deformación; el coeficiente de restitución mide la del impacto de una pelota.

Electret: micrófono de condensador con dieléctrico polarizado permanentemente, empleado en los módulos de sonido MAX9814 y KY-038.

Electrodo: conductor eléctrico utilizado para establecer contacto con una parte no metálica de un circuito, como una disolución electrolítica.

Electrodo de pH: electrodo de vidrio cuya membrana genera un potencial proporcional al pH de la disolución, medido frente a un electrodo de referencia de Ag/AgCl.

Electrólisis: descomposición química de una disolución por el paso de corriente eléctrica; explica la polarización de los electrodos durante la medida de conductividad.

Electrolito: sustancia que al disolverse se disocia en iones y hace conductor el medio; su concentración determina la conductividad eléctrica del agua.

Electrón: partícula elemental de carga negativa; su desviación por campos magnéticos (fuerza de Lorentz) origina el efecto Hall.

Emisión: radiación de energía por parte de un cuerpo; todo cuerpo emite radiación térmica según la ley de Stefan-Boltzmann.

Emisor IR: LED que emite luz infrarroja (≈ 940 nm) en las barreras ópticas y en los sensores de proximidad por reflexión.

Encoder: codificador de posición o velocidad basado en un disco ranurado y un optoacoplador, que genera un pulso por cada paso de ranura.

Energía cinética: energía asociada al movimiento, $Ec = \tfrac{1}{2}mv^2$; en el rebote de una pelota se transforma en potencial gravitatoria y parte se disipa en cada impacto.

Energía mecánica: suma de las energías cinética y potencial de un sistema; su pérdida entre rebotes cuantifica la energía disipada en el choque.

Energía potencial (gravitatoria): energía asociada a la posición de un cuerpo en un campo gravitatorio, Ep = m·g·h.

Entalpía (H): magnitud termodinámica cuya variación, a presión constante, cuantifica el calor intercambiado: ΔH < 0 en procesos exotérmicos y ΔH > 0 en endotérmicos.

Error relativo: cociente entre el error absoluto y el valor de referencia, expresado en porcentaje; permite comparar la calidad de distintas medidas.

ESP32: microprocesador de Espressif Systems con WiFi y Bluetooth integrados, dual-core a 240 MHz y ADC de 12 bits, ampliamente utilizado en IoT.

Espectro: descomposición de una radiación en sus longitudes de onda; el espectro visible abarca aproximadamente de 380 a 750 nm.

Espectrómetro: instrumento que separa la luz con una red de difracción o un prisma y mide su intensidad en función de la longitud de onda.

Espectroscopia: técnica de análisis que estudia la luz emitida o absorbida por una muestra para identificar sustancias o determinar concentraciones.

Espectro visible: intervalo del espectro electromagnético percibido por el ojo humano, entre el ultravioleta y el infrarrojo; las superficies oscuras lo absorben casi por completo.

Evaporación: paso de líquido a vapor que ocurre en la superficie a cualquier temperatura y que absorbe calor latente del entorno.

Extrapolación: prolongación de una recta de ajuste más allá del intervalo medido, como la que corta el eje de temperaturas en −273,15 °C.

Factor de calidad (Q): parámetro de un oscilador amortiguado, $Q = \omega_0/(2\gamma)$, proporcional al número de oscilaciones apreciables antes de que se disipe la energía.

Faraday-Lenz (ley de): la fuerza electromotriz inducida en un circuito es igual a la variación del flujo magnético que lo atraviesa cambiada de signo, $\varepsilon = -d\Phi/dt$.

Fase: argumento angular de una oscilación que indica el punto del ciclo en que se encuentra el sistema; también designa cada etapa intermedia de un proceso.

FAT32: sistema de archivos compatible con tarjetas SD/SDHC hasta 32 GB, con límite de 4 GB por archivo y 512 entradas en el directorio raíz.

FEM (Fuerza electromotriz): tensión generada por una fuente o inducida en un circuito; se mide en voltios.

Filtro paso bajo: circuito o algoritmo (RC, media móvil) que atenúa las componentes de alta frecuencia y deja pasar las lentas, reduciendo el ruido de la señal.

Firmware: software grabado en la memoria del microcontrolador que controla su funcionamiento.

Float: tipo de dato de C++ en coma flotante de 32 bits (≈ 7 cifras significativas) empleado para las magnitudes físicas en los sketches.

Flotante: entrada digital que no está fijada a un nivel definido por no tener resistencia de pull-up o pull-down; provoca lecturas aleatorias.

Flujo de agua: volumen de agua que circula por unidad de tiempo (caudal), medido contando los pulsos del sensor de efecto Hall YF-S201.

Flujo magnético (Φ): número de líneas de campo magnético que atraviesan una superficie, $\Phi = B \cdot A \cdot \cos\theta$; se mide en webers (Wb).

Flying-fish: sensor de proximidad por infrarrojos con salida digital, basado en un LED IR y un fototransistor con comparador LM393, distancia ajustable 2-30 cm.

Fotoconductividad: aumento de la conductividad de un semiconductor (como el CdS) al absorber fotones que liberan portadores de carga.

Fotodiodo: unión PN polarizada inversamente que genera una corriente proporcional a la luz incidente; es el elemento sensible de los sensores de luz digitales.

Fotómetro: instrumento que mide la intensidad luminosa o la iluminancia de una fuente de luz.

Fotopuerta: dispositivo formado por un emisor de luz y un fotodetector enfrentados, utilizado para detectar la interrupción de un haz luminoso.

Fotoresistencia: véase LDR.

Fototransistor: transistor sensible a la luz, cuya corriente de colector varía con la intensidad luminosa incidente.

Frecuencia: número de ciclos completos por unidad de tiempo de una onda periódica, medida en hercios (Hz).

Frecuencia angular (ω): pulsación de un movimiento armónico, $\omega = 2\pi/T$, medida en rad/s.

Frecuencia de muestreo: número de lecturas por segundo que adquiere el sistema, en Hz; limita la resolución temporal y la máxima frecuencia observable (teorema de Nyquist).

Fricción: véase Rozamiento.

Fuerza de Lorentz: fuerza que actúa sobre una carga en movimiento dentro de un campo magnético, $F = q(v \times B)$; es responsable del efecto Hall y de la inducción magnética.

Fuerza electromotriz: véase FEM.

Fusión: cambio de fase de sólido a líquido que absorbe calor latente a temperatura constante (0 °C en el hielo).

Galga extensométrica: sensor resistivo que varía su resistencia al deformarse mecánicamente, utilizado en células de carga y medidores de fuerza.

Ganancia: factor de amplificación de una señal; en el TSL2591 es programable (1x, 25x, 428x) y en el MAX9814 la ajusta automáticamente el AGC.

Gases ideales: modelo de gas con moléculas sin volumen propio y sin interacción entre ellas, que cumple la ecuación de estado $P \cdot V = n \cdot R \cdot T$.

Gay-Lussac (ley de): a volumen constante, la presión de una masa de gas es directamente proporcional a su temperatura absoluta ($P/T = \text{cte}$).

GND (Ground): referencia de tensión común (0 V) en un circuito electrónico.

GP2Y0A21: sensor analógico de distancia por infrarrojos de Sharp, rango de 10 a 80 cm, salida de tensión inversamente proporcional a la distancia.

GPIO (General Purpose Input/Output): pines de propósito general que pueden configurarse como entrada o salida digital.

Gráfica: representación de los datos experimentales frente al tiempo o entre variables; herramienta de análisis para verificar leyes físicas y ajustar rectas.

Grafito: forma alotrópica del carbono, blando, laminar y buen conductor; la mina de lápiz se emplea como resistencia de resistividad conocida.

Gravedad (g): aceleración de la caída libre en la superficie terrestre, $g \approx 9{,}81\ \text{m/s}^2$, determinable con un péndulo o con una pelota que cae.

GY-33: módulo comercial basado en el sensor de color TCS34725, con LED de iluminación blanca integrado y comunicación I^2C.

HC-020K: módulo encoder de ranura con salida digital (comparador LM393) que genera un pulso por cada ranura de su disco; mide velocidad de rotación.

HC-06: módulo Bluetooth serie clásico que añade conectividad inalámbrica a un Arduino UNO a través del puerto serie.

HC-SR04: sensor de distancia por ultrasonidos con pines Trig y Echo, rango de 2 cm a 4 m y precisión de unos 3 mm; no requiere librerías externas.

Hercio (Hz): unidad de frecuencia del Sistema Internacional, equivalente a un ciclo por segundo.

Hidrostática: parte de la mecánica de fluidos que estudia los líquidos en reposo; la presión depende solo de la profundidad, $P = P_0 + \rho \cdot g \cdot h$.

Histéresis: diferencia entre la lectura de subida y de bajada de una magnitud; también describe el retardo de respuesta del electrodo de vidrio del pH-metro.

HMC5883L: magnetómetro de 3 ejes con comunicación I^2C utilizado para medir el campo

magnético terrestre; está descatalogado y se sustituye por el QMC5883L.

Hoja de cálculo: programa (Excel, LibreOffice Calc) que permite tabular, ajustar y representar gráficamente los datos exportados en formato CSV.

Hooke (ley de): la deformación de un cuerpo elástico es proporcional a la fuerza aplicada, F = k·x, dentro de su límite elástico.

HTS221: sensor capacitivo de humedad relativa y temperatura integrado en la placa Arduino Nano 33 BLE Sense, comunicación I²C.

Humedad relativa (HR): porcentaje de vapor de agua presente en el aire respecto al máximo que admite a esa temperatura (saturación).

HX711: conversor analógico-digital de 24 bits específico para células de carga, con amplificador de instrumentación integrado.

I²C (Inter-Integrated Circuit): bus de comunicación serie síncrono desarrollado por Philips, utiliza dos líneas (SCL y SDA) y direcciones únicas por dispositivo.

IDE (Integrated Development Environment): entorno de desarrollo integrado que incluye editor de código, compilador y gestor de librerías.

IGRF (International Geomagnetic Reference Field): modelo de referencia del campo magnético terrestre con el que se compara el valor medido en el laboratorio.

Iluminancia: flujo luminoso que incide por unidad de superficie, medida en lux (lm/m²); es la magnitud que miden el BH1750 y el TSL2591.

Imán de neodimio: imán permanente de gran remanencia (NdFeB) usado para inducir una fuerza electromotriz en una espira o como fuente de campo conocido.

Incertidumbre: intervalo de valores dentro del cual se estima que se encuentra el valor verdadero de una medida.

Inclinación: ángulo entre el haz luminoso y la normal a la superficie; su coseno gobierna la iluminancia recibida (ley del coseno de Lambert).

Inclinación magnética: ángulo entre el campo magnético terrestre y el plano horizontal en un punto; mide la relación entre la componente vertical y la horizontal.

Inducción magnética: generación de una fuerza electromotriz en un circuito al variar el flujo magnético que lo atraviesa (ley de Faraday-Lenz).

Interferencia: superposición de ondas que produce máximos y mínimos de intensidad; herramienta de medida en óptica.

Interrupción: mecanismo que detiene temporalmente el programa para atender un evento externo (un pulso en un pin) con latencia de microsegundos.

Irradiancia: potencia radiante que incide por unidad de superficie, en W/m²; en el libro se mide con LDR y sensores de luz para verificar la ley de Lambert.

Kelvin (K): unidad de temperatura absoluta del SI, con el cero en el cero absoluto; T(K) = t(°C) + 273,15.

KST: aplicación libre de visualización de datos en tiempo real que lee el archivo CSV que RealTerm escribe desde el puerto serie.

KY-003: módulo comercial que integra el sensor de efecto Hall A3144 con salida digital y umbral fijo.

KY-032: módulo de detección de obstáculos por infrarrojos con LED emisor y receptor, potenciómetro de umbral y salida digital.

KY-038: módulo de sonido con micrófono electret, comparador LM393 y salidas analógica y digital con umbral ajustable.

Lambert (ley del coseno de): la iluminancia que recibe una superficie plana es proporcional al coseno del ángulo de incidencia de la radiación.

LDR (Light Dependent Resistor): fotoresistencia que disminuye su valor resistivo al aumentar la intensidad de luz incidente.

LED (Light Emitting Diode): diodo que emite luz cuando circula corriente eléctrica en sentido directo.

Ley de Beer-Lambert: la absorbancia de una disolución es proporcional a la concentración del soluto y al camino óptico recorrido por la luz.

Ley de Ohm: la intensidad que circula por un conductor es proporcional a la tensión aplicada e inversamente proporcional a su resistencia, V = I·R.

Librería: conjunto de funciones y rutinas predefinidas que facilitan la programación de periféricos y sensores.

Linealidad: grado en que la respuesta de un sensor se ajusta a una recta en todo su rango de medida, como la del LM35 (10 mV/°C).

LM35: sensor analógico de temperatura que proporciona 10 mV/°C, encapsulado TO-92 y no sumergible.

LM393: comparador de tensión doble empleado en los módulos digitales (fotopuertas, Flying-fish, KY-038) para convertir una señal analógica en un nivel lógico con umbral ajustable.

Longitud de onda (λ): distancia entre dos puntos consecutivos en fase de una onda; la luz visible abarca de 380 nm (violeta) a 750 nm (rojo).

LPS22HB: sensor piezo-resistivo de presión atmosférica integrado en la placa Arduino Nano 33 BLE Sense, rango 260-1260 hPa.

LSM9DS1: unidad de medición inercial (IMU) de 9 ejes que integra acelerómetro, giroscopio y magnetómetro, comunicación I²C.

Lumen (lm): unidad de flujo luminoso del SI; el lux equivale a un lumen por metro cuadrado.

Lux (lx): unidad de iluminancia del SI (lm/m²); el BH1750 mide de 1 a 65 535 lx y el TSL2591 de 188 µlx a 88 000 lx.

Luxómetro: instrumento destinado a medir iluminancia; en el libro se implementa con los sensores BH1750 o TSL2591.

Magnetómetro: sensor que mide la intensidad y la dirección del campo magnético; con el QMC5883L se determinan las componentes del campo terrestre.

Magnetorresistencia (AMR): variación de la resistencia de un material ferromagnético con el ángulo entre el campo magnético y la corriente; principio de funcionamiento del QMC5883L.

Magnitud física: propiedad medible de un sistema (longitud, masa, tiempo, temperatura…) que se expresa como un número acompañado de su unidad.

Malus (ley de): la intensidad transmitida a través de un analizador es $I = I_0 \cdot \cos^2\theta$, donde θ es el ángulo entre polarizador y analizador.

Manómetro: instrumento que mide diferencias de presión respecto a la atmosférica, es decir, presión manométrica.

Masa (m): medida de la cantidad de materia de un cuerpo y de su inercia; unidad del SI, el kilogramo (kg).

Masa molar (M): masa de un mol de sustancia, en g/mol; la media del aire es ≈ 28,96 g/mol y aparece en la fórmula barométrica.

Matraz Kitasato: matraz de fondo plano con tubuladura lateral, usado con un globo para estudiar la ley de Charles a presión constante.

MAX9814: amplificador de micrófono con control automático de ganancia (AGC), ideal para capturar señales de audio con micrófonos MEMS o electret.

Mbed OS: sistema operativo en tiempo real (RTOS) utilizado por las placas Arduino basadas en ARM Cortex, incluyendo la Nano 33 BLE Sense.

Mechero Bunsen: quemador de gas de laboratorio empleado para calentar el agua en el experimento del calor latente de vaporización.

Media aritmética: cociente entre la suma de un conjunto de medidas y su número; es la estimación más habitual del valor representativo.

Menisco: curvatura de la superficie libre de un líquido en un recipiente estrecho; en el enrase de un matraz aforado se lee la base del menisco.

microcontrolador: circuito integrado programable que contiene CPU, memoria y periféricos de entrada/salida en un solo chip.

microprocesador: circuito integrado que contiene únicamente la CPU, requiere componentes externos (memoria, periféricos) para funcionar.

microSD: formato de tarjeta de memoria flash basada en el estándar SD, utilizada en experimentos autónomos para almacenar datos en formato CSV.

millis(): función de Arduino que devuelve los milisegundos transcurridos desde el arranque; se usa como marca de tiempo para registrar datos.

Mina de lápiz: varilla de grafito y arcilla que sirve como resistencia lineal de resistividad conocida para verificar $R = \rho \cdot L/A$.

Módulo de Young (Y): cociente entre el esfuerzo y la deformación unitaria de un material; determina la frecuencia de vibración de una varilla empotrada.

Módulo lector de tarjetas microSD: placa que aloja la tarjeta y la conecta al microcontrolador por bus SPI para escribir los datos en un archivo CSV.

Mol (mol): cantidad de sustancia que contiene $6{,}022 \cdot 10^{23}$ entidades (número de Avogadro); es una de las unidades básicas del SI.

Momento de inercia (I): resistencia de un cuerpo a cambiar su velocidad angular, $I = \Sigma m \cdot r^2$, en $kg \cdot m^2$; interviene en la dinámica del movimiento circular.

Monitor Serie: ventana del IDE de Arduino que muestra los datos enviados por el puerto serie, habitualmente a 9600 baudios.

Movimiento armónico simple (MAS): oscilación periódica cuya aceleración es proporcional y de signo contrario al desplazamiento, con solución $x = A \cdot \text{sen}(\omega t + \varphi)$.

Movimiento circular: movimiento de trayectoria circular, caracterizado por la velocidad angular ω y la aceleración angular α.

MP34DT05: micrófono digital MEMS con salida PDM (Pulse Density Modulation), integrado en la placa Arduino Nano 33 BLE Sense.

Muelle: elemento elástico que cumple la ley de Hooke y constituye el oscilador armónico de los experimentos de oscilaciones.

Multímetro: instrumento que mide tensión, corriente y resistencia; en el libro se emplea como polímetro para verificar conexiones y valores de componentes.

Newton (N): unidad de fuerza del SI; 1 N = 1 $kg \cdot m/s^2$.

Newton (ley de enfriamiento de): la velocidad de enfriamiento de un cuerpo es proporcional a la diferencia de temperatura con el ambiente, lo que da lugar a un decaimiento exponencial.

Nivel de agua: sensor que detecta si el líquido alcanza una altura determinada, aprovechando la conductividad entre electrodos sumergidos.

Nivel lógico: tensión que representa un 1 o un 0 digital; es 5 V en el Arduino UNO y 3,3 V en el ESP32 y la Nano 33 BLE Sense.

nRF52840: microcontrolador de Nordic Semiconductor basado en ARM Cortex-M4F a 64 MHz, con 1 MB de flash y 256 KB de RAM, con aceleración BLE por hardware. Es el chip principal de la Arduino Nano 33 BLE Sense.

NTC (Negative Temperature Coefficient): termistor cuya resistencia disminuye al aumentar la temperatura.

Nyquist (teorema de): para reconstruir una señal hay que muestrear al menos al doble de su frecuencia máxima; limita la frecuencia observable con una tasa de muestreo dada.

Offset: tensión o desviación de referencia que hay que restar a la lectura de un sensor para obtener el valor verdadero.

OneWire: protocolo de comunicación serie que utiliza un solo cable de datos (más tierra) para conectar múltiples dispositivos.

Optoacoplador: dispositivo formado por un LED y un fototransistor enfrentados en un mismo encapsulado; en la versión de ranura detecta la interrupción del haz.

Oscilación amortiguada: movimiento oscilatorio cuya amplitud decrece exponencialmente por la disipación de energía debida al amortiguamiento viscoso.

Oscilador armónico: sistema cuya ecuación del movimiento es la del movimiento armónico simple, como una masa unida a un muelle ideal.

Pascal (Pa): unidad de presión del SI; 1 Pa = 1 N/m² y 1 atm = 101 325 Pa.

Pascal (principio de): la presión aplicada a un fluido encerrado se transmite íntegramente a todos sus puntos; fundamenta la medida de masa con un cojín de aire.

Patillaje: distribución y función de los pines de un componente o placa (VCC, GND, datos); su respeto es imprescindible para no dañar el sensor.

Patrón: muestra, disolución o instrumento de valor conocido y trazable que se emplea como referencia en la calibración.

Péndulo: masa suspendida de un hilo que oscila con periodo $T = 2\pi\sqrt{(L/g)}$; la medida de T y L permite determinar g.

Periodo (T): tiempo que tarda un ciclo completo de una oscilación, T = 1/f, medido en segundos.

Permeabilidad magnética (μ): propiedad del medio que cuantifica su respuesta al campo magnético; en el vacío $\mu_0 = 4\pi \cdot 10^{-7}$ T·m/A.

Permitividad eléctrica (ε): propiedad del medio que determina la fuerza entre cargas y la capacidad de un condensador, $\varepsilon = \varepsilon r \cdot \varepsilon_0$.

pH: medida logarítmica de la acidez o basicidad de una disolución, definida como $-\log_{10}[H^+]$.

pH-metro: instrumento que mide el pH a partir del potencial del electrodo de vidrio y se calibra con disoluciones tampón de pH conocido.

Phyphox: aplicación móvil desarrollada por la Universidad RWTH Aachen que permite visualizar datos de sensores enviados por Bluetooth.

Pin: terminal de conexión de un circuito integrado o placa, que puede ser de entrada, salida o ambos.

Pipeta: instrumento de laboratorio que permite transferir volúmenes pequeños y precisos de líquido, como el colorante en la cubeta de la ley de Beer-Lambert.

Placa de pruebas (protoboard): tablero con líneas conductoras internas que permite montar circuitos y soldar módulos sin realizar pistas.

Plano inclinado: superficie inclinada un ángulo θ sobre la que la componente m·g·senθ de la gravedad acelera el móvil; método clásico para medir g.

Polaridad: signo de la tensión o sentido de conexión de un componente; respetarla es imprescindible para no dañar sensores y condensadores electrolíticos.

Polarización (de la luz): restricción de la dirección de oscilación del campo eléctrico de una onda electromagnética; se consigue con láminas polarizadoras.

Polarizador: filtro que transmite únicamente la componente del campo eléctrico paralela a su eje de transmisión.

Polímetro: véase Multímetro.

Portapesas: soporte que cuelga del muelle o del hilo y sobre el que se colocan masas calibradas para aplicar fuerzas conocidas.

Posición: coordenada que localiza un móvil respecto a un sistema de referencia; su registro temporal describe la cinemática del movimiento.

Potencia (P): energía transferida o transformada por unidad de tiempo, medida en vatios (W).

Potenciómetro: resistencia variable de tres terminales; en los módulos de infrarrojos ajusta el umbral de detección del comparador.

ppm (partes por millón): unidad de concentración equivalente a 1 mg de soluto por litro de disolución acuosa (1 ppm ≈ 1 mg/L).

Precisión: grado de concordancia entre medidas repetidas de la misma magnitud; se cuantifica con la desviación típica de las medidas.

Presión (P): fuerza ejercida por unidad de superficie, en pascales (Pa); en un fluido en reposo depende de la profundidad y de la densidad.

Presión absoluta: presión medida respecto al vacío, $P = P_0 + \rho gh$; es la que registra el sensor BME280.

Presión de vapor: presión del vapor en equilibrio con su líquido; crece con la temperatura y determina la humedad de saturación.

Presión hidrostática: presión que ejerce un fluido en reposo por su propio peso, $P = \rho gh$, proporcional a la profundidad.

Presión manométrica: diferencia entre la presión absoluta y la atmosférica; es la que se obtiene al restar P_0 a la lectura del sensor sumergido.

Probeta: recipiente cilíndrico graduado de laboratorio, empleado para contener la columna de agua del experimento de hidrostática.

Promedio: véase Media aritmética.

PSRAM: memoria RAM externa opcional de algunas placas ESP32, útil para almacenar grandes búferes de datos de sensores rápidos.

Puente de Wheatstone: configuración de cuatro resistencias en forma de puente que permite medir pequeñas variaciones de resistencia, utilizada en galgas extensométricas y células de carga.

Puerto COM: puerto serie virtual que Windows asigna a la placa al conectarla por USB; debe seleccionarse en el IDE para cargar el programa.

Pull-down: resistencia que fija una entrada digital a nivel bajo (0 V) cuando no hay señal activa, evitando que quede flotante.

Pull-up: resistencia que fija una línea digital a nivel alto; es obligatoria en el bus I²C, en la línea OneWire y en la salida del fototransistor.

PWM (Pulse Width Modulation): técnica de modulación por ancho de pulso que permite simular una señal analógica mediante una onda cuadrada de ciclo de trabajo variable.

QMC5883L: magnetómetro de 3 ejes con comunicación I²C, alternativa mejorada al HMC5883L, utilizado para medir campo magnético terrestre.

Radiación infrarroja: radiación electromagnética de longitud de onda mayor que la visible (≈ 700 nm a 1 mm); transporta la mayor parte de la energía térmica de la luz solar.

Radiación solar: energía electromagnética procedente del Sol; su absorción diferencial según el color de la superficie se estudia midiendo la temperatura alcanzada.

Radiación térmica: energía emitida por todo cuerpo en función de su temperatura; sigue la ley de Stefan-Boltzmann, $P = \varepsilon \cdot \sigma \cdot A \cdot T^4$.

Radián (rad): unidad de ángulo plano del SI; 2π rad equivalen a 360°.

RAM (Random Access Memory): memoria de acceso aleatorio donde se almacenan temporalmente variables y datos durante la ejecución del programa.

Rango de medida: intervalo de valores de entrada en el que un sensor garantiza su exactitud; fuera de él la lectura no es válida (por ejemplo 4 cm a 2 m en el VL53L0X).

Rapidez: módulo o valor absoluto de la velocidad, sin indicación de la dirección del movimiento.

RealTerm: programa de terminal serie para Windows que captura los datos del puerto en un archivo de texto reutilizable por otras aplicaciones.

Red de difracción: superficie con miles de líneas paralelas por milímetro que separa la luz en sus longitudes de onda por difracción.

Reflectancia: fracción de la radiación incidente que una superficie refleja; depende del color y del acabado, y explica la absorción diferencial de la radiación.

Regla graduada: instrumento de medida de longitudes con divisiones milimétricas, empleado para calibrar y medir desplazamientos.

Regresión: ajuste de un modelo matemático (normalmente una recta) a un conjunto de datos experimentales minimizando los residuos.

Regulador: circuito integrado que mantiene constante la tensión de salida (5 V o 3,3 V) a partir de una alimentación superior.

Relé: interruptor electromecánico que permite conmutar cargas de potencia gobernadas por una señal de baja tensión.

Repetibilidad: grado de concordancia entre medidas sucesivas de la misma magnitud realizadas en idénticas condiciones.

Reproducibilidad: grado de concordancia entre medidas de la misma magnitud obtenidas en condiciones distintas (montaje rearmado, otro día, otro operador).

Resistencia: componente electrónico que se opone al paso de la corriente eléctrica, medida en ohmios (Ω).

Resistividad (ρ): resistencia específica de un material, $\rho = R \cdot A/L$, en Ω·m; caracteriza el grafito y las disoluciones electrolíticas.

Resolución: cambio mínimo de la magnitud de entrada que produce una variación detectable en la salida (10 bits → 4,9 mV por paso en 0-5 V).

Resonancia: máxima amplitud de respuesta de un oscilador cuando se excita a su frecuencia natural.

Revolución: giro completo de 2π radianes; el sensor de caudal YF-S201 genera un pulso por cada revolución de su rotor.

RGB (Red, Green, Blue): modelo de color aditivo con el que el TCS34725 descompone la luz en sus tres componentes fundamentales.

Rotación: movimiento de un cuerpo alrededor de un eje fijo, descrito por el ángulo girado y su velocidad angular.

Rozamiento: fuerza que se opone al movimiento relativo entre superficies o dentro de un fluido; su componente viscosa amortigua las oscilaciones mecánicas.

Ruido: fluctuación aleatoria de la señal que se superpone a la medida; se reduce con filtrado, promediado y buen apantallamiento.

Saturación: estado en que un sensor deja de responder proporcionalmente por exceso de la magnitud medida; el AGC del MAX9814 la evita en señales de audio fuertes.

Science Journal: aplicación de Google que registra y grafica los datos que la placa envía por BLE, empleada con la Arduino Nano 33 BLE Sense.

Semiesfera: mitad de una esfera, de volumen $(2/3)\pi r^3$; se usa para estimar el volumen del globo inflado en el experimento de Charles.

Sensibilidad: cambio de la señal de salida por unidad de cambio de la magnitud medida; en el LM35 es de 10 mV/°C.

Sensor: dispositivo que detecta magnitudes físicas o químicas y las convierte en señales eléctricas medibles.

Sensor analógico: sensor cuya salida es una tensión continua proporcional a la magnitud medida (LM35, GP2Y0A21, SS49E) y que requiere un ADC.

Serial: canal de comunicación que envía texto entre el microcontrolador y el ordenador; Serial.begin() fija la velocidad y Serial.print()/Serial.println() escriben en él.

Serial Plotter: visor gráfico integrado en el IDE de Arduino que representa en tiempo real los valores numéricos enviados por el puerto serie.

setup(): función de Arduino que se ejecuta una sola vez al arrancar la placa y en la que se inicializan el puerto serie, los sensores y la tarjeta SD.

Shield: placa de expansión apilable sobre Arduino UNO que añade funcionalidades específicas (pantallas, control de motores, etc.).

Silicio: semiconductor base de los sensores de temperatura integrados y de la electrónica del microcontrolador.

Sketch: nombre que recibe un programa escrito para Arduino en el IDE oficial.

Solenoide: bobina cilíndrica que genera un campo magnético uniforme en su interior al circular corriente y en la que un imán en movimiento induce una fem.

Soluto: componente minoritario de una disolución, que se disuelve en el disolvente; su concentración determina propiedades como la absorbancia o la conductividad.

Solvatación: envoltura de los iones por moléculas de agua (atracción ion-dipolo) que libera energía y explica el carácter exotérmico de algunas disoluciones.

Sonda: elemento sensor que se introduce en el medio (aire, agua o disolución) para medir temperatura, pH o conductividad.

SPI (Serial Peripheral Interface): bus de comunicación serie síncrono full-duplex que utiliza cuatro líneas (MOSI, MISO, SCK, CS).

SRAM: memoria RAM estática del microcontrolador donde residen las variables y los búferes durante la ejecución del programa.

SS49E: sensor de efecto Hall con salida analógica lineal proporcional a la intensidad del campo magnético.

Stefan-Boltzmann (ley de): la potencia radiada por unidad de superficie es proporcional a la cuarta potencia de la temperatura absoluta, $P = \varepsilon \cdot \sigma \cdot A \cdot T^4$.

Stokes (ley de): fuerza de rozamiento viscoso sobre una esfera que se mueve lentamente en un fluido, $F = 6\pi\eta r v$; permite determinar la viscosidad.

String: tipo de dato de Arduino para cadenas de texto, empleado al construir las líneas CSV que se escriben en la tarjeta microSD.

TCS3200 / TCS34725: sensores de color RGB con convertidor de luz a frecuencia (TCS3200) o a digital (TCS34725), utilizados para espectroscopia y detección de color.

TDS (Total Dissolved Solids): sólidos disueltos totales; su estimación a partir de la conductividad (ppm) informa sobre la salinidad de un agua.

Temperatura ambiente: temperatura del entorno, referencia ineludible en las curvas de enfriamiento y en la calibración de termistores.

Temperatura de color: temperatura del cuerpo negro cuya radiación presenta la misma cromaticidad que una fuente luminosa; se expresa en kelvin.

TEMT6000: sensor de luz ambiental basado en fototransistor con salida analógica proporcional a la iluminancia.

Termistor: resistencia cuyo valor varía significativamente con la temperatura; puede ser NTC o PTC.

Termodinámica: rama de la física que estudia la energía, el calor y el trabajo; los experimentos de gases y de calor latente son aplicaciones directas.

Termómetro: instrumento para medir temperatura; en el libro se emplean termómetros digitales (DS18B20, LM35, AHT2X) y de mercurio como referencia.

Tesla (T): unidad de densidad de flujo magnético del SI; 1 T = 1 Wb/m². El campo magnético terrestre es del orden de 50 µT.

ThingSpeak: plataforma IoT en la nube que recibe y grafica datos enviados por wifi desde una placa ESP32.

Tiempo de integración: intervalo durante el cual el sensor acumula luz para promediar y mejorar la relación señal-ruido (100-600 ms en el TSL2591).

TOF (Time of Flight): técnica de medición de distancia basada en el tiempo que tarda un pulso de luz (láser) en ir y volver del objeto.

Tolerancia: margen de variación admisible en el valor de un componente (±5 % en una resistencia), fuente habitual de error en los cálculos.

Torricelli (ley de): la velocidad de salida de un líquido por un orificio situado a una profundidad h es $v = \sqrt{(2gh)}$.

Transformador: dispositivo de dos bobinas acopladas magnéticamente que eleva o reduce la tensión alterna por inducción mutua.

Transistor: dispositivo semiconductor de tres terminales que amplifica o conmuta señales; está presente en los fototransistores y en los comparadores.

Transmitancia (T): fracción de la luz que atraviesa una muestra, $T = I/I_0$, relacionada con la absorbancia mediante $A = -\log T$.

Trayectoria: conjunto de posiciones sucesivas que describe un móvil a lo largo de su movimiento.

Trípode: soporte metálico de tres patas sobre el que se apoyan la rejilla y el matraz en los montajes de calentamiento.

TSL2591: sensor digital de luz de muy alto rango dinámico (188 µlux a 88 000 lux), ADC de 16 bits y ganancias programables, comunicación I²C.

UART: interfaz de comunicación serie asíncrona (líneas RX y TX) que la placa emplea para hablar con el ordenador o con módulos Bluetooth.

uint16_t: tipo de dato entero sin signo de 16 bits (0 a 65535) usado, por ejemplo, para los cuatro canales que entrega el TCS34725.

Ultrasonidos: ondas sonoras de frecuencia superior al límite audible humano (>20 kHz), utilizadas en sensores de distancia.

USB tipo B: conector de datos y alimentación (5 V) del Arduino UNO, similar al de las impresoras.

UUID (Universally Unique Identifier): identificador de 128 bits que designa cada característica BLE enviada a Phyphox o a Science Journal.

Vaporización: cambio de fase de líquido a gas que absorbe calor latente (2257 kJ/kg en el agua a 100 °C y presión normal).

Vaso de precipitados: recipiente de laboratorio de paredes rectas usado como cubeta transparente en los experimentos de hidrostática.

Velocidad angular (ω): ángulo girado por unidad de tiempo, en rad/s; relaciona la velocidad lineal con el radio mediante $v = \omega \cdot r$.

Velocidad instantánea: velocidad de un móvil en un instante concreto, que se aproxima midiendo intervalos de tiempo muy cortos entre dos fotopuertas.

Velocidad media: cociente entre el desplazamiento total y el tiempo empleado, $v = \Delta x/\Delta t$.

Verificación: comprobación experimental de que una ley física se cumple con los datos dentro de su incertidumbre (Faraday-Lenz, Lambert, Boyle…).

Vibración: oscilación mecánica de un cuerpo; en una varilla empotrada su frecuencia depende de la longitud, la sección y el módulo de Young.

Viscosidad: resistencia interna de un fluido a fluir; su coeficiente dinámico η se determina midiendo la velocidad límite de caída de una esfera (ley de Stokes).

Visual Studio Code: editor de código sobre el que se instala PlatformIO como extensión para compilar y subir el firmware a la placa.

VL53L0X: sensor láser TOF de STMicroelectronics para medición de distancia, rango 40 mm a 2 m, precisión ±1 mm.

Volatile: calificador de C++ que indica al compilador que una variable puede cambiar fuera del flujo normal (rutina de interrupción), impidiendo optimizaciones erróneas.

Voltaje: diferencia de potencial eléctrico entre dos puntos de un circuito, medida en voltios (V).

Voltímetro: instrumento que mide diferencias de potencial; se conecta siempre en paralelo con el elemento de interés.

Voltio (V): unidad de tensión o diferencia de potencial del SI, equivalente a un julio por culombio.

Weber (Wb): unidad de flujo magnético del SI; 1 Wb = 1 T·m².

Wemos D1 R32: placa con ESP32 que mantiene el formato y la compatibilidad de pines del Arduino UNO, con WiFi y Bluetooth integrados.

Wheatstone (puente de): véase Puente de Wheatstone.

WiFi: tecnología de comunicación inalámbrica basada en el estándar IEEE 802.11 para redes de área local.

YF-S201: sensor de caudal de agua con efecto Hall, genera pulsos proporcionales al volumen de agua circulante (frecuencia ≈ 1 L/min = 4,5 Hz).

Índice Alfabético

1-Wire — véase **OneWire**

A

A3144 — **42**, 43, 235
Absorbancia — **31**, 34, 171, 172, 182, 243
Aceleración angular — **90–92**, 96, 235
Ácido — **106**, 228
ACS712 — **47**, 235
Adafruit_VL53L0X — **24**, 74, 89, 111, 114
ADC — **15–17**, 19, 235
AGC — **47**, 235
AHT20 — **27**, 28, 235
AHT21 — **27**, 235
AHT2X — **27**, 28, 235
Amortiguamiento — **101–105**, 110, 117–119, 235
Amperio — 220
Amplificador de instrumentación — **235**
Amplificador operacional — 44, 235
analogRead — **119**
APDS9960 — **18**, 35, 62, 167, 188, 192, 193, 235
Arduino — **9**, 10, 12–21, 235
Arduino IDE — **50**, 235
Arduino Nano — **10**, 14, 17–19, 51, 56, 235
Arduino Nano 33 BLE Sense — **10**, 14, 17, 62, 182, 188, 192, 235
Arduino UNO — **9**, 14–16, 19–21, 28, 31, 41, 50, 69, 85, 86, 88, 91, 106, 235
Arduino UNO R3 — **14**, 216, 227, 235
ARM — **17**, 62, 244
Atenuación — **31**, 34, 47, 101, 235
ATmega328P — **14**, 16, 216, 235
attachInterrupt — **94**, 95, 235

B

Balanza — **93**, 106, 145, 148, 220, 236
Barrera óptica — **36**, 79, 82, 236
Baudios — **53**, 54, 236
BH1750 — **28**, 30–32, 152, 162, 166–174, 184, 185, 187, 188, 190–192, 236
BLE — **9**, 10, 14, 16–19, 51, 52, 55, 57, 62–64, 72, 74, 75, 126, 235
BLE2902 — **177**, 178, 236
Bluetooth — **10**, 14–19, 41, 54–57, 62, 64, 70, 73, 74, 236
Bluetooth Low Energy — **10**, 17, 62, 236
BME280 — **22**, 29, 30, 52, 59, 60, 122, 126, 129, 134, 137, 155–159, 161, 162, 187, 190, 191, 236
BMP280 — **28–30**, 122, 124, 126, 127, 129, 130, 132, 134–138, 155, 236
BNC — **40**, 45, 227, 228
Buffer — **93**, 95, 96, 162, 236
Bus I²C — **27**, 33, 43, 62, 122, 162, 174, 236
Bus SPI — **162**
Byte — **236**

C

C++ — **14**, 16, 22, 236
Caída libre — **24**, 77, 79, 82, 88, 236
Calor — **10**, 20, 25, 29, 88, 140, 142, 145, 146, 148, 149, 151, 152, 154, 155, 236
Calor específico — **25**, 145, 148, 236
Calor latente — **25**, 29, 145, 236
Calorímetro — 149, 236
Campo eléctrico — **173**, 204, 236
Campo magnético — **10**, 42–44, 112, 204–211, 214, 236
Capacidad calorífica — **29**, 236
Carga eléctrica — **236**
Caudal — **47**, 48, 237
CdS — **32**, 166, 237
Célula de carga — **41**, 42, 100, 102, 103, 107, 108, 235
Cero absoluto — **237**
CH340 — 51
Cinemática — **10**, 24, 34, 72, 237
Coeficiente de amortiguamiento — **101**, 105, 110, 119, 237
Coeficiente de correlación — 237
Coeficiente de determinación — **158**, 170, 237
Coeficiente de restitución — **89**, 237
Cojín de aire — **10**, 137, 237
Columna de agua — **123**, 237
Conversor analógicodigital — 14, 21, 30, 41, 235
Compilador — **235**
Componente horizontal — **207**, 237
Compresión — **158**, 237
Concentración — **44**, 45, 171, 172, 219, 227, 229, 237
Condensación — 237
Condensador — **28**, 46, 175, 197–200, 204, 217, 218, 221, 237
Conducción — **29**, 152, 237
Conductancia — 220
Conductividad — **9–11**, 19, 32, 44, 45, 201, 202, 219, 220, 222–224, 237
Conductividad eléctrica — **9–11**, 44, 201, 219, 223
configurarSensor — 209, 210
Constante de celda — 44, 237

Constante de los gases — **40**, 237
Constante de tiempo — **29**, 142, 152, 236
Constante elástica — **100**, 108, 238
Convección — **190**, 238
Conversor analógicodigital — véase **ADC**
Corriente alterna — **175**, 221, 238
Corriente continua — **44**, 91, 221, 235
Corriente eléctrica — **42**, 44, 175, 219, 237
CPU — **238**
CreateServer — 178, 191
Cronómetro — **68**, 72, 77, 85, 142, 238
Culombio — 249
Curva de calibración — **238**
Curva de enfriamiento — **142**, 238

D

Datalogger — 79, 238
Datasheet — **22**, 238
Debounce — **79**, 82–84, 238
Declinación magnética — **207**, 211, 238
Decremento logarítmico — **101**, 238
Densidad — **42**, 112, 122–124, 126, 238
Deriva — **100**, 109, 127, 229, 238
Desfase — 190, 238
Desviación típica — **95**, 238
DHT11 — **26**, 27, 238
DHT22 — **27**, 28, 235
Dieléctrico — **27**, 28, 46, 217, 238
Diferencia de potencial — **238**
Difracción — **180–182**, 184, 239
Diodo — **239**
Disco ranurado — **40**, 239
Disolución — **9**, 34, 40, 148, 171, 172, 219, 220, 222, 223, 227–229, 236
Disolvente — **148**, 239
Divisor de tensión — **110**, 119, 151, 202, 203, 239
DOIT ESP32 DEVKIT V1 — **199**, 239
DS18B20 — **10**, 22, 24, 26, 142, 144–146, 148, 149, 221–224, 239
Dualcore — **16**, 239

E

Ecuación de continuidad — 48
Ecuación de estado — **129**, 154, 237
Ecuación de Nernst — **40**, 227, 237
Efecto Doppler — 11
Efecto fotoeléctrico — 32, 166
Efecto Hall — **42**, 43, 47, 142, 204, 206, 235
Elasticidad — **88**, 129, 205, 239
Electret — **47**, 239
Electrodo — **40**, 45, 219, 221, 222, 227, 229, 239
Electrodo de pH — **40**, 239
Electrólisis — **44**, 46, 221, 239
Electrolito — **44**, 219, 239
Electrón — **239**
Emisión — **22**, 36, 151, 239
Emisividad — **151**, 152
Emisor IR — **36**, 239
Empuje — **101**, 105, 235
Encoder — **40**, 91, 239
Energía cinética — **88**, 101, 109, 161, 239
Energía mecánica — **88**, 239
Energía potencial — **88**, 240
Entalpía — **148**, 149, 240
Equilibrio térmico — 149, 151
Error relativo — **110**, 146, 159, 240
ESP32 — **9**, 10, 12–36, 235
Espectro — **151**, 180–182, 240
Espectro visible — **151**, 240
Espectrómetro — 240
Espectroscopia — **10**, 34, 180, 182, 239
Evaporación — **29**, 240
Extrapolación — 158, 240

F

Factor de calidad — 119, 240
Fase — **75**, 100, 108, 136, 145, 186, 236
FAT32 — **58**, 161, 188, 240
Fuerza electromotriz (FEM) — **214**
Filtro paso bajo — **119**, 240
Firmware — **64**, 65, 240
Flujo de agua — **47**, 240
Flujo magnético — **43**, 214, 240
Flujo turbulento — 105
Flyingfish — **24**, 36, 37, 79, 81, 84, 91, 240
Fotoconductividad — **32**, 240
Fotodiodo — **30**, 33, 38, 240
Fotómetro — **167**, 240
Fotopuerta — **38**, 76, 79, 80, 85, 90–92, 94, 95, 241
Fotoresistencia — véase **LDR**
Fototransistor — **22**, 32, 36, 38, 39, 76–79, 85, 91, 94, 166, 167, 169, 175, 177, 241
Frecuencia — **10**, 14, 23, 24, 33–35, 37, 39, 40, 42–45, 47, 60, 95, 101, 104, 109, 111–115, 117, 119, 175, 176, 180, 241
Frecuencia angular — **101**, 109, 117, 241
Frecuencia de muestreo — **111**, 241
Fricción — véase **Rozamiento**
Fuerza de Lorentz — **42**, 241
Fuerza de rozamiento — 237
Fuerza electromotriz — véase **FEM**
Fusión — **10**, 25, 145, 236

G

Galga extensométrica — 241
Gases ideales — **126**, 154, 237
GND — **23**, 26, 28, 34, 241
GP2Y0A21 — **24**, 241
GPIO — **17**, 28, 34, 62, 84, 124, 181, 182, 241
Grado Celsius — **25**
Grafito — **10**, 44, 201–203, 241
Gravedad — **72**, 79–81, 85, 88, 122, 126, 236
GY33 — **33**, 34, 180, 241

H

HC020K — **39**, 91, 241
HC06 — **41**, 241
HCSR04 — **22**, 23, 36, 68–70, 72, 88, 108–110, 241
Hercio — **241**
Hidrostática — **10**, 46, 122–124, 241
Histéresis — **116**, 205, 241
HM10 — **55**, 57
HMC5883L — **22**, 42, 43, 204, 241
Hoja de cálculo — **61**, 82, 93, 95, 96, 159, 170, 188, 215, 228, 242
HS300x — **18**, 188, 192, 193
HTS221 — **18**, 62, 188, 192, 242
Humedad relativa — **27–29**, 155, 159, 161, 162, 186, 235
HX711 — **41**, 42, 100, 102, 103, 106, 107, 235

I

I²C — **19**, 23, 27–31, 33–35, 42, 43, 48, 51, 52, 59, 62, 72, 74, 235
IDE — **10**, 16, 17, 42, 50–52, 54, 57, 58, 62, 63, 68–70, 235
IGRF — **207**, 208, 242
Iluminancia — **30**, 31, 166–170, 174, 184–186, 188–190, 235
Imán de neodimio — 242
Inclinación magnética — **207**, 210, 213, 237
Inducción magnética — **214**, 216, 241
Intensidad luminosa — **31**, 175
Interferencia — **242**
Interrupción — **40**, 94–96, 241
Irradiancia — **10**, 152, 166, 242

J

Joule, **Julio** — **236**

K

Kelvin — **26**, 34, 242
KST — **55**, 61, 242
KY003 — **42**, 235
KY032 — **22**, 24, 36, 242
KY038 — **47**, 239

L

LDR — **10**, 32, 33, 117–119, 166–168, 237
Ley de BeerLambert — **31**, 34, 171, 182, 243
Ley de Boyle — **158**, 237
Ley de BoyleMariotte — **30**, 154, 157
Ley de Charles — **154**, 243
Ley de enfriamiento de Newton — **142**
Ley de Faraday — **10**, 215, 216
Ley de FaradayLenz — **216**, 242
Ley de GayLussac — **154**
Ley de Hooke — **42**, 100, 101, 103, 104, 108–110, 238
Ley de Lambert — **10**, 166, 242
Ley de Lenz — 214
Ley de Malus — **173**
Ley de Newton — **29**, 72, 91, 100, 108
Ley de Ohm — **44**, 201, 243
Ley de Stefan Boltzmann — **151**
Ley de Stokes — **105**
Ley de Torricelli — **48**
Ley del coseno de Lambert — **242**
Librería — **24**, 32, 243
LM35 — **25**, 26, 243
LM393 — **36**, 37, 39, 47, 240
Longitud de onda — **33**, 151, 180, 181, 240
LPS22HB — **18**, 62, 188, 192, 243
LSM9DS1 — **18**, 62, 243
Lumen — **30**
Lux — **30–32**, 166–168, 170, 172–174, 185–187, 193, 236
Luxómetro — **167**, 168, 170, 243
Luz polarizada — 236

M

Magnetómetro — **18**, 22, 43, 207, 208, 210, 235
Magnetorresistencia — **243**
Magnitud física — 243
Manómetro — 157, 243
Masa molar — 126, 243
Matraz Kitasato — 157, 243
MAX9814 — **47**, 235
Mbed OS — **18**, 51, 56, 243
MCU — **90–92**, 94, 95
Mechero Bunsen — **155**, 156, 158, 243

Media aritmética — **243**
MEMS — **29**
Menisco — **243**
microcontrolador — **14, 20, 235**
microprocesador — **14–17**, 38, 240
microSD — **10**, 58, 59, 61, 161, 162, 186–188, 191, 192, 238
millis — **60**, 244
Mina de lápiz — **241**
MISO — **58**, 161, 162, 248
Módulo de Young — **112**, 114, 244
Módulo lector de tarjetas microSD — **58**, 161, 244
Mol — **148**, 149, 154, 219, 237
Molaridad — 237
Momento de inercia — **90**, 91, 93, 112, 114, 244
Monitor Serie — **10**, 50, 53, 54, 60, 69, 244
MOSI — **58**, 161, 162, 248
Movimiento armónico simple — **100, 244**
Movimiento circular — **10**, 90, 235
Movimiento circular uniforme — 10
MP34DT05 — **18**, 244
Multímetro, **Polímetro** — **201**, 244

N

Neutralización — 41, 149
Nivel de agua — **10**, 46, 244
Nivel lógico — **244**
nRF52840 — **17**, 62, 235

O

Offset — **106**, 107, 206, 244
OneWire — **56**, 146, 149, 223, 235
Optoacoplador — **39**, 239
Oscilación amortiguada — **244**
Oscilador armónico — **244**
OTG — **17**, 55
OTP — **28**, 235

P

Péndulo — **40**, 85, 86, 241
Periodo — **77**, 83, 85, 86, 95, 100, 101, 103–105, 109–111, 113, 115–119, 176, 245
Permeabilidad magnética — **43**, 245
Permitividad eléctrica — **10**, 217, 245
Peso aparente — 104
pH — **10**, 19, 20, 40, 41, 226–230, 236
pHmetro — **15**, 16, 41, 227, 236
Phyphox — **16**, 18, 19, 55–57, 62–64, 70–72, 236
Pipeta — 171, 245
Placa de pruebas — **102**, 175, 245
Plano inclinado — **72–74**, 245
Polaridad — **245**
Polarización — **10**, 31, 34, 44, 173, 221, 239
Polarizador — **173**, 243
Polímetro — véase **Multímetro**
Potencia — **16**, 19, 151, 152, 166, 175, 242
Potencial eléctrico — 40
Potenciómetro — **37**, 47, 91, 228, 242
Presión — **9**, 10, 18, 22, 29, 30, 46, 47, 59, 60, 122–138, 148, 154–163, 186, 188, 190, 192, 201, 236
Presión absoluta — **124**, 129, 132, 136, 154, 157, 246
Presión atmosférica — **10**, 30, 122–124, 126, 127, 130, 161, 162, 186, 236
Presión de vapor — **186**, 237
Presión hidrostática — **122**, 123, 246
Presión manométrica — **122**, 123, 246
Principio de Arquímedes — **42**, 101, 103–105, 235
Principio de Pascal — **157**
Probeta — **122**, 246
Promedio — véase **Media aritmética**
PSRAM — 17, 246
Puente de Wheatstone — **41**, 246
Puerto COM — **50**, 51, 246
Puerto serie — **15**, 37, 50, 51, 53, 54, 56, 69, 241
Pulldown — **38**, 76, 240
Pullup — **25**, 27, 28, 42, 91, 94, 95, 107, 142, 145, 221, 240
PWM — **235**

Q

QMC5883L — **43**, 207, 208, 211, 242

R

Radiación infrarroja — **186**, 190, 246
Radiación solar — **10**, 151, 161, 186, 246
Radiación térmica — **151**, 152, 239
RAM — **15**, 62, 235
RealTerm — **54**, 55, 69, 70, 118, 152, 199, 246
Red de difracción — **180–182**, 240
Reducción — **190**
Reflectancia — **34**, 246
Reflexión — **23**, 36, 73, 109, 239
Relé — 247
Resistividad — **201**, 202, 241
Resonancia — 247
RGB — **33**, 35, 62, 180, 235
Rotación — **40**, 43, 81, 173, 207, 239

Rozamiento, Fricción — **72**, 74, 81, 82, 86, 101, 105, 109, 110, 235

S

Science Journal — **10**, 18, 64, 65, 247
Serial Plotter — **53**, 54, 185, 216, 247
Shield — 247
Sketch — **10**, 237
Solenoide — **216**, 248
Solvatación — 248
SS49E — **22**, 42, 43, 204, 206, 235

T

Tarjeta microSD — **58**, 59, 61, 161, 162, 186, 188, 191, 238
TCS3200 — **34, 248**
TCS34725 — **28**, 33–35, 180–183, 190, 241
TDS — **45**, 219–224, 248
Temperatura ambiente — **142**, 152, 158, 186, 248
Temperatura de color — **34**, 182, 248
TEMT6000 — **32**, 33, 91, 166, 167, 169, 175, 248
Tensión superficial — **42**, 123, 134, 135
Teorema de Nyquist — **241**
Termistor — **26**, 151, 152, 248
Termodinámica — **159**, 240
Termómetro — **18**, 24, 142, 145, 152, 154, 155, 223, 224, 248
ThingSpeak — **161**, 162, 248
Tiempo de integración — **31**, 183, 248
TOF — **23**, 24, 88, 108, 109, 111, 248
Transformador — **248**
Transistor — **25**, 32, 38, 39, 166, 241
Transmitancia — **31**, 249
TSL2591 — **22**, 31, 32, 184, 188, 190, 191, 241

U

UART — **51**
Ultrasonidos — **22**, 68, 108, 109, 241
USB tipo B — **14**
UUID — **177**, 249

V

Vaporización — **243**
Varilla empotrada —115
Velocidad angular — **43**, 90, 92, 93, 95, 96, 244
Velocidad instantánea — **24**, 249
Velocidad media — **69**, 73, 77, 249
Vibración — **112–114**, 119, 206, 244
Viscosidad — **102**, 104, 105, 220, 248
Visual Studio Code — **57**
VL53L0X — **22–24**, 36, 72, 74, 75, 88, 89, 108–111, 113–115, 246
Volatile — **94**, 95, 249
Voltímetro — 249

W

Wemos D1 R32 — **10**, 14, 15, 17, 19, 20, 48, 237
Wheatstone (puente de) — véase **Puente de Wheatstone**

Y

YFS201 — **47**, 237

Referencias bibliográficas

Adafruit Learning System (2023). Adafruit VL53L0X Time of Flight Micro-LIDAR Distance Sensor Breakout.
Disponible en: https://learn.adafruit.com/adafruit-vl53l0x-micro-lidar-distance-sensor-breakout

Adiati, R.F. y otros (2025), Design of an Automatic Pendulum Velocity Measuring Device Using Light Sensors. *Indonesian Physical Review*. Vol. 08, núm. 01.
Disponible en: http://ipr.unram.ac.id/index.php/ipr/article/download/404/172

Alumnos de Secundaria y de Bachillerato (2008a). Gráficas de movimientos sencillos. IES "Leonardo Da Vinci" de Alicante.
Disponible en: https://www.dedfisica-rsef.com/recursosdidacticos/LaboratorioCarritos/Movimientos/Movimientos.htm

Alumnos de Secundaria y de Bachillerato (2008b). Rodadura de un carrito por un plano inclinado (I). IES "Leonardo Da Vinci" de Alicante.
Disponible en: https://www.dedfisica-rsef.com/recursosdidacticos/LaboratorioCarritos/inclinado-sensores.htm

Amador, A. (2025). Determinación de g a partir del movimiento de un objeto en un plano inclinado. Departamento de Física, Universidad de Buenos Aires-Exactas.
Disponible en: https://asignaturas.df.uba.ar/l1-amador/wp-content/uploads/sites/23/2025/02/2025-Verano-Clase-TP-4-Amador-1.pdf

Andromina Robot, 2016, Encoder para robot. Tutorial sobre el encoder fotoeléctrico HC-020K, usado en un robot Arduino.
Disponible en: https://androminarobot.blogspot.com/2016/04/tutorial-sobre-el-encoder-fotoelectrico.html

Aparicio, J., Corominas, J. y Lozano, M. (2003a). Entalpia d'una reacció endotérmica. CDECT.
Disponible en: https://experimentaciolliure.com/wp-content/uploads/2023/02/endotermica.pdf

Aparicio, J., Corominas, J. y Lozano, M. (2003b). Entalpia d'una reacció endotérmica. Material per al professorat. CDECT.
Disponible en: https://experimentaciolliure.com/wp-content/uploads/2023/02/pendotermica.pdf

Aparicio, J., Corominas, J. y Lozano, M. (2003c). Entalpia d'una reacció exotérmica. CDECT.
Disponible en: https://experimentaciolliure.com/wp-content/uploads/2023/02/exotermica.pdf

Aparicio, J., Corominas, J. y Lozano, M. (2003d). Entalpia d'una reacció exotérmica. Material per al professorat. CDECT.
Disponible en: https://experimentaciolliure.com/wp-content/uploads/2023/02/pexotermica.pdf

Aparicio, J. y Lozano, M. (2002a). Protocolo de prácticas con Multilog: El moviment armònic simple. Activitats de l'alumnat.
Disponible en: https://experimentaciolliure.com/wp-content/uploads/2023/02/harmonic.pdf

Aparicio, J. y Lozano, M. (2002b). Protocolo de prácticas con Multilog: El moviment armònic simple. Material per al professorat.
Disponible en: https://experimentaciolliure.com/wp-content/uploads/2023/02/pharmonic.pdf

Aparicio, J. y Lozano, M. (2002c). Protocolo de prácticas con Multilog: Impuls i quantitat de moviment. Activitats de l'alumnat.
Disponible en: https://experimentaciolliure.com/wp-content/uploads/2023/02/impuls.pdf

Aparicio, J. y Lozano, M. (2002d). Protocolo de prácticas con Multilog: Impuls i quantitat de moviment. Material per al professorat.
Disponible en: https://experimentaciolliure.com/wp-content/uploads/2023/02/pimpuls.pdf

Aparicio, J. y Lozano, M. (2002e). Protocolo de prácticas con Multilog: Energia disipada en el bot d'una pilota. Activitats de l'alumnat.
Disponible en: https://experimentaciolliure.com/wp-content/uploads/2023/02/dissipada.pdf

Aparicio, J. y Lozano, M. (2002f). Protocolo de prácticas con Multilog: Energia disipada en el bot d'una pilota. Material per al professorat.
Disponible en: https://experimentaciolliure.com/wp-content/uploads/2023/02/pdissipada.pdf

Aparicio, J. y Lozano, M. (2002g). El moviment harmònic simple.
Disponible en: https://experimentaciolliure.com/wp-content/uploads/2023/02/harmonic.pdf

Aparicio, J. y Lozano, M. (2002g). El moviment harmònic simple. Material per al professorat.
Disponible en: https://experimentaciolliure.com/wp-content/uploads/2023/02/pharmonic.pdf

Aparicio, J. y Lozano, M. (2003a). Moviment de caiguda d'un regle.
Disponible en: https://experimentaciolliure.com/wp-content/uploads/2023/02/regle.pdf

Aparicio, J. y Lozano, M. (2003b). Moviment de caiguda d'un regle. Material per al professorat.
Disponible en: https://experimentaciolliure.com/wp-content/uploads/2023/02/pregle.pdf

Aparicio, J. y Lozano, M. (2003c). Llei de Boyle. Centre de Documentació i Experimentació en Ciències i Tecnologia.
Disponible en: https://experimentaciolliure.com/wp-content/uploads/2023/02/boyle.pdf

Aparicio, J. y Lozano, M. (2003d). Llei de Boyle. Material per al professorat. Centre de Documentació i Experimentació en Ciències i Tecnologia.
Disponible en: https://experimentaciolliure.com/wp-content/uploads/2023/02/pboyle.pdf

Aparicio, J. y Lozano, M. (2004a). Protocolo de prácticas con Multilog: Com fer un salt de bungee-jumping segur. Activitats de l'alumnat.
Disponible en: https://experimentaciolliure.com/wp-content/uploads/2023/02/bungee.pdf

Aparicio, J. y Lozano, M. (2004b). Protocolo de prácticas con Multilog: Com fer un salt de bungee-jumping* segur. Material per al professorat.
Disponible en: https://experimentaciolliure.com/wp-content/uploads/2023/02/pbungee.pdf

Aparicio, A. y Lozano, M.T., 2004c. Estudi de la llum emesa per un fluorescent. Centre de Documentació i Experimentació en Ciències i Tecnologia.
Disponible en: https://experimentaciolliure.com/wp-content/uploads/2023/02/fluorescent.pdf

Aparicio, A. y Lozano, M.T., 2004d. Estudi de la llum emesa per un fluorescent. Material per al professorat. Centre de Documentació i Experimentació en Ciències i Tecnologia.
Disponible en: https://experimentaciolliure.com/wp-content/uploads/2023/02/pfluorescent.pdf

Arduino (2021). Arduino Science Journal. Aplicación móvil.
Disponible en: https://science-journal.arduino.cc

ARM Limited (2022). Mbed OS Documentation.
Disponible en: https://os.mbed.com/docs

BBC Science (2026). Monitoring the rate of a reaction.Bitesize.
Disponible en: https://www.bbc.co.uk/bitesize/guides/zct4fcw/revision/7

Bobroff. J., 2020. Tutos Smartphones: mesurer la masse avec la presión. La Physique Autrement.
Disponible en: https://www.youtube.com/watch?v=lAp7UWu6NF8

Bouquet, F., Dauphin, C., Bernard, F. y Bobroff, J., 2019. Low-Cost Experiments with Everyday Objects for Homework Assignments. *Physics Education*, 54.
Disponible en: https://www.researchgate.net/publication/330116374_Low-cost_experiments_with_everyday_objects_for_homework_assignments
https://arxiv.org/pdf/1807.03203

CATEDU, 2026. Práctica 13: Cómo percibir el color con Arduino. Librería CATEDU.
Disponible en: https://libros.catedu.es/books/arduino-y-pure-data-ondas-color-y-sonido/page/practica-13-como-percibir-el-color-con-arduino

CDECT (2002a). El pla inclinat: moviment rectilini uniformement accelerat. Departament d'Ensenyament de Catalunya. Centre de documentació en ciències i tecnología.
Disponible en: https://experimentaciolliure.com/wp-content/uploads/2020/05/inclinat.pdf

CDECT (2002b). Pèndol simple. Departament d'Ensenyament de Catalunya. Centre de documentació en ciències i tecnología.
Disponible en: https://experimentaciolliure.com/wp-content/uploads/2020/05/pend_simp.pdf

CDECT (2002c). Determinació estàtica de la constant d'una molla. https://experimentaciolliure.com/wp-content/uploads/2020/05/hooke.pdf

CDECT (2002d). Determinació dinàmica de la constant d'una molla. Departament d'Ensenyament de Catalunya. Centre de documentació en ciències i tecnología.
Disponible en: https://experimentaciolliure.com/wp-content/uploads/2020/05/molla_din.pdf

Circuit Schools Staff, 2025. TDS Meter with TDS Sensor and Arduino for water quality monitoring in Realtime
Disponible en: https://www.circuitschools.com/tds-meter-with-tds-sensor-and-arduino-for-water-quality-monitoring/

Cortel, A., 1992. EXAO1: Un equip d'experimentació assistida per ordinador. Revista de Física. RSEF.
Disponible en: https://raco.cat/index.php/RevistaFisica/article/view/2038

de Castro, L. H. M., Lago, B. L. y Mondaini, F. (2015). Damped Harmonic Oscillator with Arduino. Journal of Applied Mathematics and Physics.
Disponible en:
https://www.researchgate.net/publication/279184279_Damped_Harmonic_Oscillator_with_Arduino

Departamento de Física, 2006a. Calor latente de fusión del hielo. Universidad de Cantabria.
Disponible en:
https://personales.unican.es/lopezqm/IFE/laspracticas/experimentosPDF/calorpdf/11bCalorLatHielo(11b).pdf

Departamento de Física, 2017, Péndulo simple. Facultad de Ciencias. Universidad Católica del Norte. Chile.
Disponible en: http://www.fisica.ucn.cl/wp-content/uploads/2017/09/GL2-PENDULO-SIMPLE.pdf

Departamento de Física (2021). Determinación de la constante de recuperación elástica de un resorte. Universidad de Jaen
Disponible en:
https://www.ujaen.es/departamentos/fisica/sites/departamento_fisica/files/uploads/node_seccion_de_micrositio/2021-02/Anexo%20Experimento%20muelle%20resuelto.pdf

Departamento de Física, 2026. Movimiento circular. Laboratorio de mecánica y termodinámica (grado en ciencias). Universidad Carlos III. Madrid
Disponible en: https://laboratoriofisica.uc3m.es/guiones_esp/mt/3.movcirc.pdf

Departamento de Informática y Sistemas (2026). Práctica número 2. Carga y descarga de un condensador. Universidad de Las Palmas de Gran Canaria
Disponible en: https://www2.ulpgc.es/hege/almacen/download/40/40814/practica_numero_2.pdf

Domínguez, M. y otros, 2019. Mejorando una práctica del laboratorio docente de Física mediante el uso de Arduino y de sensores de bajo coste. Escuela Superior de Ingeniería, Puerto Real (Cádiz).
Disponible en: https://indoc.uca.es/articulos/sol-201800111587-tra.pdf

Elórtegui, 2024. Emisión y absorción de energía por radiación. Bloc Ciencia de cada día.
Disponible en: https://www.elortegui.org/ciencia/datos/1FICESO/134Ondas/3Pract/03/03%20Radiacion.html

ENOSA, 1982. Movimiento circular uniforme. Mecánica EE. MM. (Enseñanzas Medias). Empresa Nacional de Óptica. Madrid.
Disponible en: https://experimentaciolliure.wordpress.com/wp-content/uploads/2026/07/enosa-circular.pdf

Franco, A., 2015, Curso Interactivo de Física en Internet.
Disponible en: http://www.sc.ehu.es/sbweb/fisica3/

García Salinas, M.J., 2009. Carga y Descarga de un Condensador Eléctrico. Departamento de Física Aplicada. Universidad de Almería.
Disponible en: https://w3.ual.es/~mjgarcia/cargaydescarga.pdf

Geomagnetism, 2026. International Geomagnetic Reference Field (IGRF), 14th Generation Calculator
Disponible en: https://geomag.bgs.ac.uk/data_service/models_compass/igrf_calc.html

Giancoli, D. C., 2009. Física: principios con aplicaciones (6.ª ed.). Pearson Educación.

Gil, S., 2016. Experimentos de Física de bajo costo, usando TIC's. Parte 2, pág. 249.
https://www.academia.edu/27797975/Experimentos_de_F%C3%ADsica_de_bajo_costo_usando_TICs_Parte_2

Gómez, P., 2023. Demostració experimental de la diferència de pressions entre l'interior i l'exterior d'una bombolla de sabó (pág 65) en *Bombolles... més que un divertiment*. Treball de Recerca Segon de Batxillerat. Ies Guindàvols. Lleida.
Disponible en: https://drive.google.com/file/d/1bi0Pnrz_529cXMuKcSQRRSM3UuSJb10Q/view

Hubbard, K., 2021. Mobile Phone Spectrophotometer Experiments.
Disponible en: https://katharinehubbard.blog/2021/04/20/mobile-phone-spectrophotometer-experiments/

IEEE (2020). IEEE Standard for Low-Power Wireless Personal Area Networks (Bluetooth). IEEE Std 802.15.1-2020.

IUPAC (2019). Compendium of Chemical Terminology (Gold Book). Versión 2.3.3.
Disponible en: https://goldbook.iupac.org

Laboratorio de Física, 2014. Entalpía de fusión del hielo. CC Físicas, UCM.
Disponible en: https://fisicas.ucm.es/data/cont/media/www/pag-36931/prac14-1314.pdf

Larios, E., 2023. Construyendo tu Propia Estación Meteorológica IoT con ESP32: Monitoreo en Tiempo Real y Datos en ThingSpeak.
Disponible en: https://medium.com/@eder88559/construyendo-tu-propia-estaci%C3%B3n-meteorol%C3%B3gica-iot-con-esp32-monitoreo-en-tiempo-real-y-datos-en-58505fbffa18

Latorre, E. (2020). Sensor BME280 con Arduino (SPI e I²C).
Disponible en: https://www.enrique-latorre.com

Ling, S.J., Sanny, J. y Moebs, W., 2021a, *Física Universitaria.Volumen 1* (Mecánica, Ondas y Acústica), OpenStax Rice University.
Disponible en: https://openstax.org/details/books/f%C3%ADsica-universitaria-volumen-1

Ling, S.J., Sanny, J. y Moebs, W., 2021b, *Física Universitaria.Volumen 2* (Termodinámica, Electricidad y Magnetismo), OpenStax Rice University.
Disponible en: https://openstax.org/details/books/f%C3%ADsica-universitaria-volumen-2

Ling, S.J., Sanny, J. y Moebs, W., 2021c, *Física Universitaria.Volumen 3* (Óptica, Física Moderna), OpenStax Rice University.
Disponible en: https://openstax.org/details/books/f%C3%ADsica-universitaria-volumen-3

Llamas, L. (2019). Medir distancia con precisión con Arduino y sensor láser VL53L0X y VL6180X. Luis Llamas Ingeniería.
Disponible en: https://www.luisllamas.es/medir-distancia-con-arduino-y-sensor-laser-vl53l0x/

Llamas, L., 2026. Medir valores RGB con Arduino y sensor de color TCS34725. Web del autor
Disponible en: https://www.luisllamas.es/arduino-sensor-color-rgb-tcs34725/

Marcelo, 2021. Estación meteorológica portátil con ESP32.
Disponible en: https://otroblogdemarcelo.wordpress.com/2021/09/27/estacion-meteorologica-portatil-con-esp32/

Marco, E.A., 2017. Visualització del moviment harmònic simple aplicant les noves tecnologies. *Ciències*, núm 34.
Disponible en: https://revistes.uab.cat/ciencies/article/view/n34-arnau/4

Martínez, N., Diaz, A., Rosas, Y. y López, N., 2012. Experimento: "¿Cuánto esperar para tomar el café?". Universidad Politécnica del Valle de México.
Disponible en: https://ingenieriaindustrialupvmtareasytrabajos.wordpress.com/wp-content/uploads/2012/07/practica-ecuaciones-ley-de-enfriamiento-de-newton.pdf

Melià, R.M., 2007, Pressió hidrostàtica amb sensors Multilab. CDEC. Departament d'Educació. Generalitat de Catalunya.
Disponible en: https://experimentaciolliure.com/wp-content/uploads/2023/02/pressi_hidrostatica_multilog.zip

Molas, M.C., Rodriguez, Fl. y Leibovich, D., 2009. Ley de enfriamiento de Newton considerando reservorios finitos. Universidad Favaloro, C. A. de Buenos Aires.
Disponible en: https://www.fisicarecreativa.com/informes/infor_termo/reserv_finitos.pdf

Monteiro, M., 2016. Ley de Malus con Smartphone. Blog Física Martín.
Disponible en: https://fisicamartin.blogspot.com/2016/05/ley-de-malus-con-smartphone.html

Monteiro, M., 2017. Astronomía con Smartphone. Blog Física Martín.
Disponible en: https://fisicamartin.blogspot.com/2017/10/astronomia-con-smartphone.html

Nave, C.R. y Olmo, M., 2010, *Hyperphysics*.
Disponible en: http://hyperphysics.phy-astr.gsu.edu/hbasees/hph.html

NCEI, 2026. International Geomagnetic Reference Field (IGRF). National Centers for Environmetal Information. NOAA/NCEI.
Disponible en: https://www.ncei.noaa.gov/products/international-geomagnetic-reference-field

Organtini, G., 2016, *Scientific arduino programming. Arduino programming for scientists*. Sapienza Università di Roma & INFN–Sez. di Roma.
Disponible en: https://www.roma1.infn.it/people/organtini/publications/scientificArduino.pdf

Organtini, G., 2020, *Fisica con Arduino*. Scienze Zanichelli
Disponible en: https://giovanniorgantini.wordpress.com/

Padín, B., 2019, *Física con Arduino*.
Disponible en: https://fisica-arduino.gitbook.io/sensores

Peralta, A. y otros, 2024. Presión. Universidad de Cuenca. Ecuador
Disponible en: https://www.collegesidekick.com/study-docs/15482938

Phyphox, 2018, (In)elastic collision. Universidad de Aachen
Disponible en: https://phyphox.org/wiki/index.php?title=Experiment:_Inelastic_Collision

Physics with Vernier, 2026, Picket Fence Free Fall. Vernier Science Education.
Disponible en: https://www.vernier.com/experiment/pwv-5_picket-fence-free-fall/

Ramírez, L., 2009. Difracció, interferència i espectroscòpia visible I: Xarxes de difracció.
Disponible en: https://experimentaciolliure.com/2009/12/19/difraccio-interferencia-i-espectroscopia-visible-i-xarxes-de-difraccio/

Ramírez, L., 2012. Errors en les mesures. Bloc Experimentació Lliure.
Disponible en: https://experimentaciolliure.com/constants/errors-en-les-mesures/

Ramírez, L., 2016, Experiments de mecànica amb l'ajut del sensor de so dels telèfons mòbils i tauletes. *Recursos de Flisica*, núm 18.
Disponible en: https://rrfisica.cat/num/num18/article_num=33&pos=5&total=8&art=191.html

Ramírez, L., 2017, Experimentació amb tecnologia mòbil. Estudi de moviments amb fotocèl•lula. Bloc Experimentació Lliure.
Disponible en: https://experimentaciolliure.com/2017/09/29/experimentacio-amb-tecnologia-mobil-estudi-de-moviments-amb-fotocel%e2%80%a2lula/

Ramirez, L., 2017b. Experimentació amb tecnologia mòbil. Mesurant amb el sensor de llum. Bloc Experimentació Lliure.
Disponible en: https://experimentaciolliure.com/2017/11/28/experimentacio-amb-tecnologia-mobil-mesurant-amb-el-sensor-de-llum/#intensitat-de-la-llum-i-la-distancia

Ramírez, L., 2017c. Experimentació amb tecnologia mòbil. El sensor magnètic i el camp magnètic d'un imant. Bloc Experimentació Lliure.
Disponible en: https://experimentaciolliure.com/2017/04/11/experimentacio-amb-tecnologia-mobil-el-sensor-magnetic-i-el-camp-magnetic-dun-imant/

Ramírez, L., 2017d. Experimentació amb tecnologia mòbil. El sensor magnètic i el camp magnètic de la Terra. Bloc Experimentació Lliure.
Disponible en: https://experimentaciolliure.com/2017/03/07/experimentacio-amb-tecnologia-mobil-el-sensor-magnetic-i-el-camp-magnetic-de-la-terra/

Ramírez, L., 2018a, Experimentació amb tecnologia mòbil. Variació de la pressió amb l'altura en un ascensor. Bloc Experimentació Lliure.
Disponible en: https://experimentaciolliure.com/2018/11/12/experimentacio-amb-tecnologia-mobil-variacio-de-la-pressio-amb-laltura-en-un-ascensor/

Ramírez, L., 2018b, Experimentació amb tecnologia mòbil. La pressió a l'interior d'un globus. Bloc Experimentació Lliure.
Disponible en: https://experimentaciolliure.com/2018/12/05/experimentacio-amb-dispositius-mobils-la-pressio-a-linterior-dun-globus./

Ramírez, L., 2020. Llei d'absorció de la llum de Beer-Lambert amb telèfon móvil. Bloc Experimentació Lliure.
Disponible en: https://experimentaciolliure.com/2020/11/26/llei-dabsorcio-de-la-llum-de-beer-lambert-amb-telefon-mobil/

Ramírez, L., 2020b. Un pH-metre al laboratori de secundaria. Bloc Experimentació Lliure.
Disponible en: https://experimentaciolliure.com/2020/01/16/un-ph-metre-al-laboratori-de-secundaria/

Ramírez, L., 2021. Mesura de la freqüència pel desplaçament de la vareta en la vibració. Experimentació Lliure.
Disponible en: https://experimentaciolliure.com/2021/03/29/varetes-fixades-per-un-extrem-mesura-de-la-frequencia-de-vibracio/#despla%C3%A7ament

Ramírez, L., 2021b. Mesurar la massa d'un objecte per la pressió que realitza. Experimentació Lliure.
Disponible en: https://experimentaciolliure.com/2021/05/04/mesurar-la-massa-dun-objecte-per-la-pressio-que-realitza/

Ramírez, L., 2021c. Lleis dels gasos. Del tub de Mariotte a Arduino. Experimentació Lliure.
Disponible en: https://experimentaciolliure.com/2021/02/24/lleis-dels-gasos-del-tub-de-mariotte-a-arduino

Ramírez, L,, 2021d. Càrrega i descarrega d'un condensador amb Arduino. Experimentaciò Lliure.
Disponible en: https://experimentaciolliure.com/2021/11/23/carrega-i-descarrega-dun-condensador-amb-arduino/

Ramírez, L,, 2021e. Estudi de la velocitat de caiguda d'un imant per un tub. Física amb Audacity i aplicacions similars. Experimentaciò Lliure.
Disponible en: https://experimentaciolliure.com/tallers-i-treballs-practics/experimentacio-assistida-per-ordinador-exao/fisica-amb-audacity-i-aplicacions-similars/#imanttub

Ramírez, L., 2022. *Experimentación en Física con dispositivos móviles 2*. Bloc Experimentació Lliure.
Disponible en: https://experimentaciolliure.com/2022/06/27/experimentacion-en-fisica-con-dispositivos-moviles-2-segona-edicio-del-llibre/

Ramírez, L., 2022b. Determinació experimental de la permeabilitat magnètica, amb telèfon. Bloc Experimentació Lliure.
Disponible en: https://experimentaciolliure.com/2022/01/30/determinacio-experimental-de-la-permeabilitat-magnetica-amb-telefon/

Ramírez, L., 2022b. Determinar experimentalment amb Arduino la permitivitat elèctrica de l'aire. Bloc Experimentació Lliure.
Disponible en: https://experimentaciolliure.com/2022/02/17/determinar-experimentalment-amb-arduino-la-permitivitat-electrica-de-laire/

Ramírez, L., 2023a, Propostes de treballs pràctics EXAO. Bloc Experimentació lliure.
Disponible en: https://experimentaciolliure.com/tallers-i-treballs-practics/experimentacio-assistida-per-ordinador-exao/propostes-de-treballs-practics-exao/

Ramírez, L., 2023b, Estudi de moviments amb barreres de llum controlades per Arduino. Bloc Experimentació lliure.
Disponible en: https://experimentaciolliure.com/2023/05/25/estudi-de-moviments-amb-barreres-de-llum-controlades-per-arduino/#temps-que-triga-un-objecte-en-passar-per-una-barrera-de-llum

Ramírez, L., 2024, Nova versió d'Arduino Nano 33 BLE Sense REV2 amb Phyphox i Science Journal. Bloc Experimentació Lliure.
Disponible en: https://experimentaciolliure.com/2024/02/02/nova-versio-darduino-nano-33-ble-sense-rev2-amb-phyphox-i-science-journal/

Ramírez, 2025. El grafit no sempre condueix igual. Experimentació Lliure.
Disponible en: https://experimentaciolliure.com/2025/01/15/el-grafit-no-sempre-condueix-igual/

Ramirez, L. (2026). Experimentació Lliure. Blog de divulgación científica con experimentos de física.
Disponible en: https://experimentaciolliure.com

Ramírez, L., 2026b. Anàlisi de la freqüència de la llum amb un fototransistor i Phyphox. Blog Experimentació Lliure.
Disponible en: https://experimentaciolliure.com/2026/02/26/analisi-de-la-frequencia-de-la-llum-amb-un-fototransistor-i-phyphox/

Ramírez, L., 2026c. Mesura de les variables ambientals en un eclipsi solar. Blog Experimentació Lliure.
Disponible en: https://experimentaciolliure.com/2026/04/13/mesura-de-les-variables-ambientals-en-un-eclipsi-solar/

Romero, A.R., Castellanos H.E. y Alvarez-Salazar, C.E., 2022. Using homemade spectrometers to perform accurate measurements of discrete and continuous spectra. Physics Education.
Disponible en: https://arxiv.org/abs/2201.07110

RSEF, 2007. P Exp. Campo magnético de un imán y campo magnético terrestre. Real Sociedad Española de Física. Olimpiada de Física.
Disponible en: https://rsef.es/wp-content/uploads/2025/07/P-EXPERIMENTAL-OEF-2007.pdf

RWTH Aachen University (2023). Phyphox: Physical Phone Experiments. Aplicación móvil.
Disponible en: https://phyphox.org

Santos, S. y Santos, R., 2021, ESP32: Guide for MicroSD Card Module using Arduino IDE.
Disponible en: https://randomnerdtutorials.com/esp32-microsd-card-arduino/

Serway, R. A. y Jewett, J. W. (2018). Física para ciencias e ingeniería (10.ª ed.). Cengage Learning.
SparkFun Electronics (2023). *SparkFun HX711 Load Cell Amplifier Hookup Guide*.
Disponible en: https://learn.sparkfun.com/tutorials/load-cell-amplifier-hx711-breakout-hookup-guide

The Vernier blog (2026). 3 Hands-On Experiments Using the Go Direct Photogate. Vernier Science Education.
Disponible en: https://www.vernier.com/blog/3-hands-on-experiments-using-the-go-direct-photogate/

Tipler, P. A. y Mosca, G. (2010). Física para la ciencia y la tecnología (6.ª ed.). Reverté.

UPC, 2026. Práctica de Laboratorio A: Condesadores y capacidad. Dieléctricos: medida de la constante dieléctrica. Universidad Politécnica de Cataluña
Disponible en: https://upcommons.upc.edu/server/api/core/bitstreams/f12b9969-91aa-4154-bd48-e75eba0078dd/content

Velasco, C., 2023. *Guardianes del Aire y del Agua: Estaciones de medición de calidad de aire y agua*. Proyecto Fin de Carrera, E.T.S.I. y Sistemas de Telecomunicación (UPM), Madrid.
Disponible en: https://oa.upm.es/78588/1/PFG_VELASCO_RUA_CRISTINA.pdf

Young, H. D. y Freedman, R. A. (2013). Física universitaria (13.ª ed.). Pearson.

Datasheets y documentación técnica

Allegro MicroSystems (2016). *A3144 Hall-Effect Switch Datasheet*. Rev. Disponible en: https://www.alldatasheet.com/view.jsp?Searchword=A3144

Allegro MicroSystems (2020). *ACS712 Fully Integrated, Hall Effect-Based Linear Current Sensor IC Datasheet*. Rev. 6. Disponible en: https://www.allegromicro.com/-/media/files/datasheets/acs712-datasheet.pdf

ams-OSRAM (2020). *TCS34725 Color Light-to-Digital Converter with IR Filter Datasheet*. Rev. 1.5. Disponible en: https://ams-osram.com/products/sensor-solutions/ambient-light-color-spectral-proximity-sensors/ams-tcs34725-color-sensor

ams-OSRAM (2020). *TSL2591 High-Sensitivity Light-to-Digital Converter Datasheet*. Rev. 1.4. Disponible en: https://ams-osram.com/support/download-center?search=TSL2591

ams-OSRAM / módulos GY-33 (s.f.). *TCS3200 Datasheet*. Disponible en: https://ams-osram.com/support/download-center?search=TCS3200

Aosong Electronics (2019). *DHT11 Humidity & Temperature Sensor Datasheet*. Disponible en: https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf

Aosong Electronics (2019). *DHT22 (AM2302) Humidity & Temperature Sensor Datasheet*. Disponible en: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf

Aosong Electronics (2020). *Datasheet AHT20 — Temperature and Humidity Sensor*. Rev. 1.0. https://www.aosong.com/userfiles/files/media/Data%20Sheet%20AHT20.pdf

Aosong Electronics (2021). *Datasheet AHT21 — Temperature and Humidity Sensor*. Rev. 1.0. https://www.aosong.com/userfiles/files/media/Data%20Sheet%20AHT21.pdf

Arduino (2021). *Arduino Nano 33 BLE Sense Product Reference Manual*. Disponible en: https://docs.arduino.cc/hardware/nano-33-ble-sense

Arduino (2023). *Arduino Nano 33 BLE Sense Rev2 Product Reference Manual*. Disponible en: https://docs.arduino.cc/hardware/nano-33-ble-sense-rev2/

Arduino / Microchip (Atmel) (s.f.). *Arduino Nano Datasheet*. Disponible en: https://docs.arduino.cc/hardware/nano/

Arduino / Microchip (Atmel) (s.f.). *Arduino UNO R3 Datasheet*. Disponible en: https://docs.arduino.cc/hardware/uno-rev3/

Atmel Corporation (2015). *ATmega328P Datasheet*. Rev. 2015. Disponible en: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

Avia Semiconductor (2016). *HX711 24-Bit Analog-to-Digital Converter Datasheet*. Disponible en: https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf

Bosch Sensortec (2018). *BME280 Datasheet*. Rev. 1.6. Disponible en: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme280-ds002.pdf

Bosch Sensortec (2021). *BMP280 Digital Pressure Sensor Datasheet*. Rev. 2.6. Disponible en: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bmp280-ds001.pdf

Broadcom (2016). *APDS-9960 Digital Proximity, Ambient Light, RGB and Gesture Sensor Datasheet*. Rev. 1.8. Disponible en: https://docs.broadcom.com/doc/AV02-4191EN

DFRobot (2020). *Gravity: Analog pH Meter Pro Kit Datasheet*. Disponible en: https://www.dfrobot.com/product-1780.html

DFRobot (s.f.). *Sensor de conductividad / TDS Datasheet*. Disponible en: https://wiki.dfrobot.com/Gravity__Analog_TDS_Sensor___Meter_For_Arduino_SKU__SEN0244

ElecFreaks (2019). *HC-SR04 Ultrasonic Sensor Datasheet*. Disponible en: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf

Espressif Systems (2023). *ESP32-WROOM-32D Datasheet*. Rev. 2.5. Disponible en: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf

Honeywell (2010). *HMC5883L 3-Axis Magnetometer Datasheet*. Disponible en: https://cdn-shop.adafruit.com/datasheets/HMC5883L_3-Axis_Digital_Compass_IC.pdf

Honeywell (2020). *SS49E Linear Hall-Effect Sensor Datasheet*. Disponible en: https://www.mouser.com/datasheet/2/187/honeywell-sensing-ss49e-linear-hall-effect-sensor-d-1228541.pdf

JNHuaMao (chip Texas Instruments CC2541) (s.f.). *Módulo Bluetooth HM-10 Datasheet*. Disponible en: https://prod-edam.honeywell.com/content/dam/honeywell-edam/sps/siot/ja/products/sensors/magnetic-sensors/linear-and-angle-sensor-ics/common/documents/sps-siot-ss39et-ss49e-ss59et-product-sheet-005850-3-en-ciid-50359.pdf

Maxim Integrated (2015). *MAX9814 Microphone Amplifier with AGC and Low-Noise Microphone Bias Datasheet*. Rev. Disponible en: https://www.analog.com/media/en/technical-documentation/data-sheets/max9814.pdf

Maxim Integrated (2019). *DS18B20 Programmable Resolution 1-Wire Digital Thermometer*. Rev. 7. Disponible en: https://www.analog.com/media/en/technical-documentation/data-sheets/ds18b20.pdf

Nordic Semiconductor (2021). *nRF52840 Product Specification*. Rev. 2.2. Disponible en: https://infocenter.nordicsemi.com/pdf/nRF52840_PS_v1.2.pdf

ROHM / Avago (módulos GY-302) (s.f.). *BH1750FVI Datasheet*. Disponible en: https://www.mouser.com/datasheet/2/348/bh1750fvi-e-186247.pdf

Sea YF Sensor (2019). *YF-S201 Hall Effect Flow Sensor Manual*. Disponible en: https://cdn-shop.adafruit.com/product-files/828/C898+datasheet.pdf

Sharp Corporation (2016). *GP2Y0A21YK0F Distance Measuring Sensor Unit Datasheet*. Rev. 1.3. Disponible en: https://www.alldatasheet.com/view.jsp?Searchword=Gp2y0a21yk0f

Shenzhen Shengsheng Sensing Technology (2020). *QMC5883L 3-Axis Magnetic Sensor Datasheet*. Disponible en: https://www.tinytronics.nl/image/data/Producten/Sensoren/Magnetisch%20veld/Datasheet-QMC5883L-1.0%20.pdf

Songle (s.f.). *Módulo relé Datasheet*. Disponible en: https://www.circuitbasics.com/wp-content/uploads/2015/11/SRD-05VDC-SL-C-Datasheet.pdf

STMicroelectronics (2016). *VL53L0X Datasheet*. Rev. 5. Disponible en: https://www.st.com/resource/en/datasheet/vl53l0x.pdf

STMicroelectronics (2017). *HTS221 Capacitive Digital Sensor for Relative Humidity and Temperature Datasheet*. Rev. 4. Disponible en: https://www.st.com/resource/en/datasheet/hts221.pdf

STMicroelectronics (2018). *LSM9DS1 iNEMO Inertial Module: 3D Accelerometer, 3D Gyroscope, 3D Magnetometer Datasheet*. Rev. 4. Disponible en: https://www.st.com/resource/en/datasheet/lsm9ds1.pdf

STMicroelectronics (2019). *LPS22HB Piezoresistive Absolute Pressure Sensor Datasheet*. Rev. 5. Disponible en: https://www.st.com/resource/en/datasheet/lps22hb.pdf

STMicroelectronics (2019). *MP34DT05-A Miniature MEMS Digital Output Microphone Datasheet*. Rev. 2. Disponible en: https://www.st.com/resource/en/datasheet/mp34dt05-a.pdf

Texas Instruments (2020). *LM35 Precision Centigrade Temperature Sensors Datasheet*. Rev. X. Disponible en: https://www.ti.com/lit/ds/symlink/lm35.pdf

Texas Instruments (s.f.). *LM393 Datasheet*. Disponible en: https://www.ti.com/lit/ds/symlink/lm393.pdf

Varios (clones del módulo CSR BC417) (s.f.). *Módulo Bluetooth HC-06 Datasheet*. Disponible en: https://agelectronica.lat/pdfs/textos/H/HC-06-BT-MODULE.PDF

Varios (módulos KY) (s.f.). *Módulo KY-032 Datasheet*. Disponible en: https://arduinomodules.info/ky-032-infrared-obstacle-avoidance-sensor-module/

Varios (módulos KY) (s.f.). *Módulo KY-038 Datasheet*. Disponible en: https://arduinomodules.info/ky-038-microphone-sound-sensor-module/

Varios (s.f.). *Codificador óptico HC-020K Datasheet*. Disponible en: https://mrelectrobot.com/wp-content/uploads/2021/09/HC-020K.pdf

Varios (s.f.). *LDR (fotoresistencia) Datasheet*. Disponible en: https://www.handsontec.com/dataspecs/sensor/GL55-LDR.pdf

Varios (s.f.). *Módulo de obstáculos IR Flying-fish Datasheet*. Disponible en: https://agelectronica.lat/pdfs/textos/O/OKY3127.PDF

Varios (s.f.). *Módulo lector microSD Datasheet*. Disponible en: https://docs.arduino.cc/learn/programming/sd-guide/

Varios (s.f.). *Sensor de nivel de agua Datasheet*. Disponible en: https://curtocircuito.com.br/datasheet/sensor/nivel_de_agua_analogico.pdf

Varios (s.f.). *Fototransistor (generico) Datasheet*. Disponible en: https://img.ozdisan.com/ETicaret_Dosya/1055037_1215163107..pdf

Varios (Vishay, TDK, Epcos) (s.f.). *Termistor NTC Datasheet*. Disponible en: https://www.vishay.com/docs/29049/ntcle100.pdf

Vishay Semiconductors (2018). *TEMT6000 Ambient Light Sensor Datasheet*. Rev. 1.6. Disponible en: https://www.vishay.com/docs/81579/temt6000.pdf

Vishay Semiconductors (s.f.). *LED infrarrojo Datasheet*. Disponible en: https://www.vishay.com/docs/81009/tsal6100.pdf

WCH (Nanjing Qinheng Microelectronics) (s.f.). *Convertidor USB-serie CH340 Datasheet*. Disponible en: https://www.wch-ic.com/downloads/CH340DS1_PDF.html

Wemos / Espressif (s.f.). *Wemos D1 R32 Datasheet*. Disponible en: https://handsontec.com/dataspecs/module/ESP/WeMos%20D1%20R32.pdf

Librerías

Adafruit (2023). *Adafruit AHTX0 Library*. https://github.com/adafruit/Adafruit_AHTX0

Adafruit (2023). *Adafruit LPS2X Library* (incluida en el núcleo Mbed OS para Nano 33 BLE Sense). Repositorio: https://github.com/adafruit/Adafruit_LPS2X

Adafruit (2023). *Adafruit LSM9DS1 Library*. Repositorio GitHub: https://github.com/adafruit/Adafruit_LSM9DS1

Adafruit (2023). *Adafruit TCS34725 Library*. Repositorio GitHub: https://github.com/adafruit/Adafruit_TCS34725

Adafruit (2023). *Adafruit TSL2591 Library*. Repositorio GitHub: https://github.com/adafruit/Adafruit_TSL2591_Library

Adafruit (2023). *Adafruit_BH1750 Library*. Repositorio GitHub: https://github.com/adafruit/Adafruit_BH1750

Adafruit (2023). *Adafruit_BME280.h*. Repositorio GitHub: https://github.com/adafruit/Adafruit_BME280_Library

Adafruit (2023). *Adafruit_BMP280.h*. Repositorio GitHub: https://github.com/adafruit/Adafruit_BMP280_Library

Adafruit (2023). *Adafruit_Sensor.h*. Repositorio GitHub: https://github.com/adafruit/Adafruit_Sensor

Adafruit (2023). *Adafruit_VL53L0X Library*. Repositorio GitHub: https://github.com/adafruit/Adafruit_VL53L0X

Arduino (2023). *Arduino_APDS9960.h*. Repositorio GitHub: https://github.com/arduino-libraries/Arduino_APDS9960

Arduino (2023). *Arduino_HS300x.h*. Repositorio GitHub: https://github.com/arduino-libraries/Arduino_HS300x

Arduino (2023). *Arduino_HTS221.h*. Repositorio GitHub: https://github.com/arduino-libraries/Arduino_HTS221

Arduino (2023). *Arduino_LPS22HB.h*. Repositorio GitHub: https://github.com/arduino-libraries/Arduino_LPS22HB

Arduino (2023). *SD.h*. Repositorio GitHub: https://github.com/arduino-libraries/SD

Arduino (s.f.). *Arduino.h*. Repositorio GitHub: https://github.com/arduino/ArduinoCore-avr/blob/master/cores/arduino/Arduino.h

Arduino (s.f.). *SPI.h*. Repositorio GitHub: https://github.com/arduino/ArduinoCore-avr/tree/master/libraries/SPI

Arduino (s.f.). *Wire.h*. Repositorio GitHub: https://github.com/arduino/ArduinoCore-avr/tree/master/libraries/Wire

Arduino Science Journal. Firmware: https://github.com/arduino/Arduino-Science-Journal-Android

Bogdan Necula (2021). *HX711 Arduino Library*. Repositorio GitHub: https://github.com/bogde/HX711

Christopher Laws (s.f.). *BH1750.h*. Repositorio GitHub: https://github.com/claws/BH1750

Claus Kühnel (2022). *BluetoothSerial Library* (incluida en el núcleo ESP32 para Arduino).

Espressif Systems (2023). *BLEDevice.h*. Repositorio GitHub: https://github.com/espressif/arduino-esp32/tree/master/libraries/BLE

Espressif Systems (2023). *FS.h*. Repositorio GitHub: https://github.com/espressif/arduino-esp32/tree/master/libraries/FS

Miles Burton (2022). *DallasTemperature Library* (gestor de librerías Arduino). Repositorio: https://github.com/milesburton/Arduino-Temperature-Control-Library

Paul Stoffregen (s.f.). *OneWire.h*. Repositorio GitHub: https://github.com/PaulStoffregen/OneWire

Rob Tillaart (2022). *DHTlib Library* (gestor de librerías Arduino). Repositorio: https://github.com/RobTillaart/DHTlib

RWTH Aachen University (2023). *phyphox BLE Library*. Repositorio GitHub: https://github.com/phyphox/phyphox-arduino

Seeed Studio (2020). *Grove_BME280 Library*. Repositorio GitHub: https://github.com/Seeed-Studio/Grove_BME280

SparkFun Electronics. *SparkFun AHT20 Library*. Repositorio GitHub: https://github.com/sparkfun/SparkFun_Qwiic_Humidity_AHT20_Arduino_Library

Vale