EDUDEMOS

Demostrador 3×1:

Energía Solar, eólica y captación de agua en un solo proyecto educativo

EDUDEMOS

El proyecto "EduDemoS" está financiado por la Unión Europea. No obstante, los puntos de vista y opiniones expresados son exclusivamente los del autor o autores y no reflejan necesariamente los de la Unión Europea ni los de la Agencia Ejecutiva en el Ámbito Educativo y Cultural Europeo (EACEA). Ni la Unión Europea ni la EACEA pueden ser consideradas responsables de las mismas.

Este primer bloque contendrá toda la información necesaria para entender el proyecto, su financiación, objetivos y participantes. Los apartados de este primer bloque de la guía serán:

1. INTRODUCCIÓN A EDUDEMOS

1. 1. ¿Qué es EduDemos?

EduDemos es un proyecto educativo que desarrolla demostradores sostenibles basados en energías renovables, como el agua, el sol y el aire. Estos demostradores son herramientas interactivas y replicables utilizadas para enseñar a jóvenes y profesores sobre sostenibilidad y competencias digitales, integrando el aprendizaje práctico con la conciencia ambiental.

1. 2. Objetivos

El proyecto EduDemos tiene como objetivo principal sensibilizar a las nuevas generaciones, tanto jóvenes como profesores, sobre la sostenibilidad mediante el uso de demostradores basados en energías renovables como el sol, el agua y el aire. Además, busca mejorar las competencias digitales en el ámbito educativo, proporcionando herramientas tecnológicas prácticas y replicables. Otro de sus propósitos es fomentar una red de colaboración entre educadores para el intercambio de conocimientos y experiencias prácticas, promoviendo la adopción de prácticas sostenibles y contribuyendo a un futuro más verde en Europa.

1. 3. Metodología

La metodología del proyecto EduDemos se estructura en torno a la implementación práctica de demostradores sostenibles que utilizan fuentes de energía renovable como el agua, el sol y el aire. Estos demostradores no son solo herramientas teóricas; están diseñados para ser construidos y experimentados por los estudiantes, fomentando un aprendizaje activo y práctico.

- **Desarrollo de modelos:** Se diseñan modelos de demostradores que integran tecnologías de energías renovables. Estos modelos sirven como ejemplos prácticos que los estudiantes pueden replicar.

- **Guías de construcción:** EduDemos proporciona guías detalladas para la construcción de estos demostradores, asegurando que los estudiantes y docentes puedan seguir paso a paso el proceso. Estas guías son accesibles y están diseñadas para ser usadas en un entorno educativo, facilitando el aprendizaje de conceptos técnicos y científicos de manera práctica.

- **Formación de docentes y estudiantes:** Se implementan programas de formación dirigidos tanto a profesores como a estudiantes. Estos programas están orientados a desarrollar competencias tanto en sostenibilidad como en habilidades digitales, preparando a los participantes para enfrentar los desafíos del cambio climático. La formación incluye talleres, cursos, y recursos educativos que complementan el aprendizaje teórico con la aplicación práctica.

- **Colaboración europea:** El proyecto fomenta la colaboración entre distintas instituciones educativas y organizaciones en Europa. Esta red de cooperación permite el intercambio de conocimientos, experiencias y recursos, enriqueciendo el proceso educativo. Los socios en el proyecto trabajan en conjunto para adaptar y mejorar los modelos y guías según las necesidades locales y regionales, promoviendo una educación más inclusiva y eficaz en toda Europa.

1. 4. Partners

El proyecto EduDemos cuenta con varios socios clave que aportan su experiencia y recursos para el éxito del proyecto:

- **Gerda Stetter Stiftung (Alemania):** Esta fundación se especializa en proyectos educativos y tecnológicos, enfocándose en el desarrollo sostenible. Su participación asegura que los aspectos técnicos y pedagógicos del proyecto sean de alta calidad.

- **Fundación Sergio Alonso (España):** Una fundación que apoya iniciativas educativas y sociales, especialmente en Canarias. Su contribución es crucial para la implementación del proyecto en España y para asegurar su relevancia local.

- **FINNOVAREGIO (Bélgica):** Esta organización se enfoca en la innovación regional y el desarrollo sostenible. Su rol en EduDemos es clave para integrar el proyecto en redes europeas de innovación y sostenibilidad.

- **GBS St.Gallen (Suiza):** Esta institución educativa suiza aporta su experiencia en formación profesional y técnica, especialmente en el ámbito de las energías renovables. Participa en el desarrollo de los contenidos educativos y la metodología del proyecto.

- **IES El Rincón (España):** Un instituto de educación secundaria en Gran Canaria, España, que actúa como uno de los centros piloto donde se implementan y prueban los demostradores sostenibles del proyecto. Su participación es fundamental para ajustar el proyecto a las necesidades de los estudiantes y profesores.

Cada uno de estos socios aporta su experiencia específica, lo que permite que EduDemos sea un proyecto integral que aborda la educación en sostenibilidad desde diferentes perspectivas y en varios contextos europeos. La colaboración entre estos partners asegura que los recursos desarrollados sean diversos, accesibles y aplicables en distintos entornos educativos.

2. DEMOSTRADOR 3x1

2. 1. Descripción de los modelos

2. 1. 1. Objetivos

El objetivo de los demostradores es enseñar y concienciar a los estudiantes sobre cómo se genera energía a partir del viento, el sol y la captación de agua, promoviendo una comprensión práctica de las energías renovables y la

importancia de la sostenibilidad medioambiental. Estos demostradores buscan involucrar a los estudiantes en la construcción y experimentación con tecnologías sostenibles, fomentando habilidades técnicas y competencias relacionadas con la sostenibilidad.

Estos demostradores están dirigidos principalmente a estudiantes de la etapa educativa de formación profesional, pero también pueden ser utilizados en el nivel de secundaria . Concretamente abarca desde la educación secundaria obligatoria hasta el bachillerato y formación profesional, donde los estudiantes ya tienen una base en ciencias y tecnología que les permite comprender y experimentar con conceptos más avanzados como la energía eólica o solar.

Su valor educativo radica en su capacidad para ofrecer una experiencia de aprendizaje práctico y contextualizado en torno a las energías renovables. A través de la construcción y el uso del Demostrador 3x1, los estudiantes desarrollan una comprensión más profunda de cómo se genera la energía a partir del viento, el sol y la captación de agua, además de fomentar habilidades técnicas, competencias digitales y un mayor compromiso con la sostenibilidad ambiental. Esto facilita el aprendizaje activo y el desarrollo de competencias clave en ciencia y tecnología.

2. 1. 2. Competencias educativas

Los demostradores facilitan el desarrollo de competencias clave en estudiantes que se encuentran realizando una formación profesional, especialmente los que están cursando estudios de familias profesionales técnicas, pero también para estudiantes de 1º a 4º de ESO. A través de su construcción y análisis los estudiantes adquieren habilidades esenciales para su desarrollo, además de fomentar el trabajo en equipo, aplicar la creatividad y el pensamiento crítico para resolver problemas, también comprenderán el impacto de las tecnologías sostenibles en la sociedad. Este enfoque fomenta una formación integral que prepara a los estudiantes para los desafíos actuales y futuros.

Competencias educativas a adquirir con el Demostrador 3x1 para **1º ESO - 4º ESO**:

1. **Competencia en comunicación lingüística (CCL):**
   Desarrollo de habilidades de comunicación: Los estudiantes aprenderán a comunicar sus ideas y resultados de manera efectiva en diferentes formatos (oral, escrito, visual), empleando un lenguaje técnico adecuado y respetuoso. Esto se aplica tanto en la representación de soluciones tecnológicas como en la elaboración de documentación técnica básica.

2. **Competencia matemática y en ciencia, tecnología e ingeniería (STEM):**
   Diseño y fabricación de soluciones: Los estudiantes aplicarán conocimientos de matemáticas, ciencia y tecnología para diseñar y construir el Demostrador 3x1, utilizando herramientas de diseño asistido por ordenador y técnicas de fabricación digital.

3. **Competencia digital (CD):**
   Uso de herramientas digitales: Los estudiantes emplearán aplicaciones y plataformas digitales para gestionar el proyecto, desde la ideación hasta la presentación final.. Los estudiantes aprenderán a utilizar aplicaciones digitales para representar esquemas, circuitos y diseñar componentes del Demostrador 3x1.

4. **Competencia personal, social y de aprender a aprender (CPSAA):**
   Trabajo en equipo y gestión de proyectos: A lo largo del proyecto, se fomentará la capacidad de los estudiantes para trabajar en equipo de manera efectiva, compartiendo responsabilidades y colaborando para resolver problemas. También se promoverá la autorreflexión y la mejora continua del proceso de aprendizaje mediante la evaluación crítica de las soluciones propuestas.

5. **Competencia emprendedora (CE):**
   Creatividad e innovación: Los estudiantes serán desafiados a desarrollar soluciones tecnológicas innovadoras que generen valor para la comunidad, utilizando estrategias de resolución de problemas y fomentando un espíritu emprendedor. La planificación y ejecución de proyectos sostenibles y accesibles serán clave en este proceso.

6. **Competencia en conciencia y expresión culturales (CCEC):**

Valorización cultural y tecnológica: Los estudiantes aprenderán a expresar y difundir sus propuestas tecnológicas de manera creativa y efectiva, utilizando diversos medios y técnicas, y reflexionarán sobre la importancia de las tecnologías sostenibles en diferentes contextos culturales, reconociendo el valor del patrimonio tecnológico y su impacto en la sociedad.

Estas competencias se adquieren a través de un enfoque práctico y contextualizado en torno al Demostrador 3x1, para que los estudiantes no solo adquieran conocimientos técnicos, sino también habilidades transversales y una mentalidad crítica y sostenible, que son relevantes tanto en su vida académica como personal.

De esta manera, se contribuye ampliamente a las competencias establecidas en el Perfil de salida del alumnado al término de la enseñanza básica, favoreciendo la transversalidad del aprendizaje del alumnado, y la concreción de los principios y fines del sistema educativo español en este periodo educativo.

2. 1. 3. Conceptos básicos

- Demostrador de viento

El demostrador de viento es una herramienta educativa que permite a los estudiantes construir un pequeño aerogenerador para comprender los principios básicos de la energía eólica y su aplicación en la generación de electricidad. A través de la interacción con el dispositivo, los estudiantes aprenden cómo la energía cinética del viento se transforma primero en energía mecánica, mediante el movimiento de las palas, y luego en energía eléctrica gracias a un generador.

Este demostrador incluye componentes esenciales como palas, un eje y un generador, lo que facilita la comprensión práctica de conceptos como la aerodinámica de las palas, la relación entre la velocidad del viento y la eficiencia energética, y el impacto ambiental de la energía eólica frente a otras fuentes de energía. Además, permite explorar cómo el diseño y la orientación de las palas influyen en la maximización de la generación de energía.

Inspirado en los "Juguetes del Viento" de César Manrique, este dispositivo fusiona arte y funcionalidad educativa, conectando la herencia cultural de Canarias con la enseñanza de las energías renovables. Es una herramienta que no solo fomenta el aprendizaje técnico, sino que también celebra la armonía entre naturaleza, tecnología y cultura local.

- Demostrador solar

El demostrador solar es una herramienta educativa diseñada para que los estudiantes comprendan cómo la luz solar se convierte en electricidad mediante paneles fotovoltaicos. Inspirado en el movimiento de los girasoles, que optimizan su posición para maximizar la captación de energía solar, este dispositivo permite a los estudiantes observar cómo las células solares captan la luz y la transforman en corriente eléctrica, ilustrando de manera práctica el proceso del efecto fotovoltaico.

El demostrador incluye un panel solar ajustable, que ayuda a los estudiantes a explorar cómo la orientación del panel y el ángulo de incidencia de la luz solar afectan la eficiencia energética. Al igual que los girasoles siguen el sol a lo largo del día para maximizar su exposición, el demostrador destaca la importancia de la correcta alineación de los paneles solares para obtener un rendimiento óptimo.

Además de enseñar los principios técnicos de la energía fotovoltaica, el demostrador ofrece una experiencia educativa que conecta a los estudiantes con la naturaleza y fomenta una visión práctica y eficiente del aprovechamiento de recursos sostenibles.

- Demostrador de agua

El demostrador de agua es una herramienta educativa que permite a los estudiantes comprender cómo se puede capturar y gestionar el agua de lluvia de manera eficiente. A través de este dispositivo, los estudiantes observan el proceso de recolección de agua, desde la captación en una superficie similar a un tejado hasta su almacenamiento en un depósito. Esto les permite explorar el ciclo del agua y aplicar estrategias de conservación hídrica en la vida cotidiana.

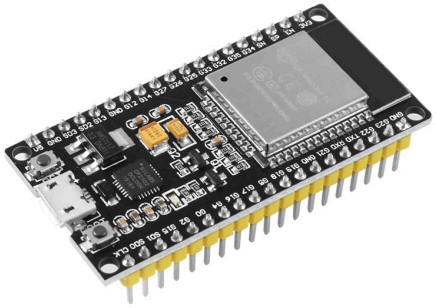
El demostrador ilustra cómo el agua de lluvia, un recurso renovable y gratuito, puede ser aprovechada para diversas aplicaciones prácticas, como el riego de plantas. Además, los estudiantes aprenden sobre la importancia de mantener limpios los sistemas de captación y almacenamiento, así como sobre la relevancia de la sostenibilidad en el uso eficiente del agua, especialmente en zonas con escasez de recursos hídricos.

Inspirado en la estética y el funcionamiento del Bejeque, una suculenta autóctona de Canarias, este recolector de agua conecta la naturaleza y la tecnología moderna, demostrando cómo las soluciones basadas en la biomimética pueden preservar los recursos hídricos. Siguiendo la filosofía de integración artística y natural de César Manrique, el recolector fusiona arte, ecología y tecnología educativa, promoviendo el respeto por el medio ambiente y la cultura local, mientras enseña a los estudiantes a valorar y aplicar soluciones sostenibles inspiradas en la naturaleza para el futuro.

3. PROGRAMACIÓN DE LOS DEMOSTRADORES

3. 1. Requisitos mínimos

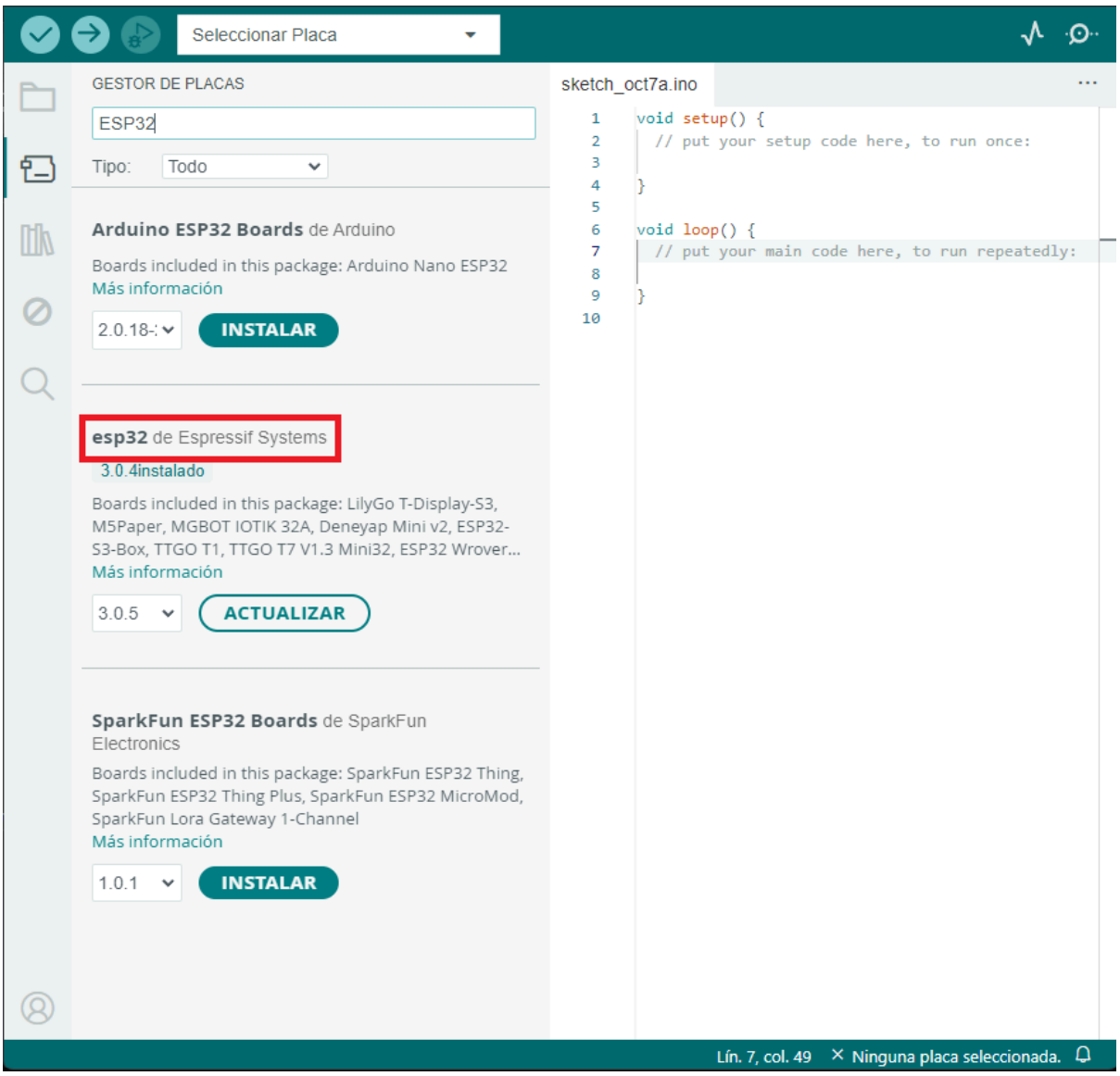
Los demostradores emplean una placa programable llamada NodeMCU, que está basada en el microcontrolador ESP32. Para poder empezar a programar esta placa desde el PC, debemos tener en cuenta una serie de aspectos.



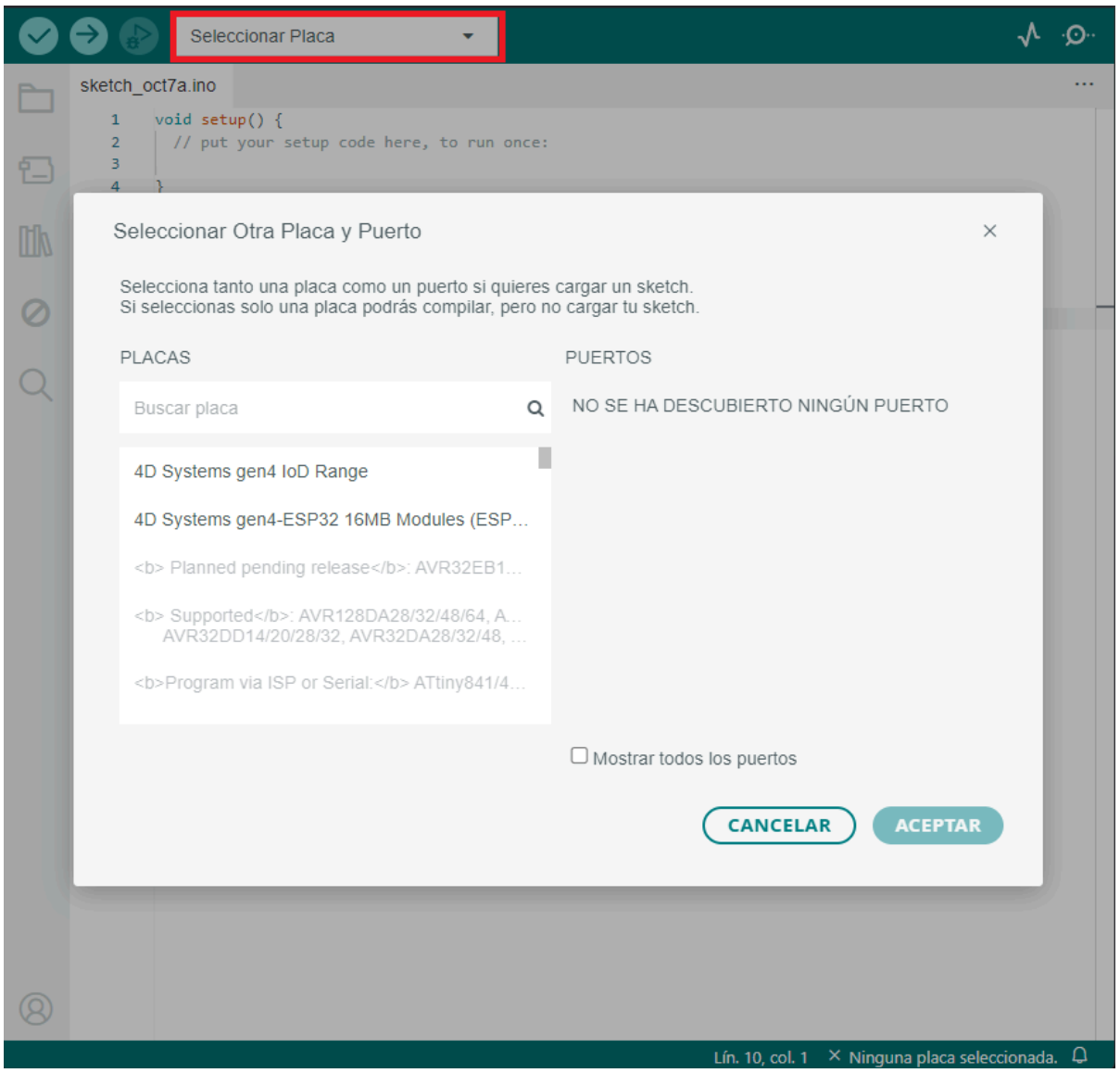
En primer lugar, **debemos tener instalado el IDE** (Integrated Development Environment), que no es más que la interfaz que nos permite programar nuestra placa. El NodeMCU es compatible con el IDE de Arduino, que es bastante conocido y fácil de usar. Para descargar la aplicación tenemos que ir a la página oficial de Arduino. Para los usuarios de PC, la opción más sencilla es pinchar sobre la opción que corresponda según el sistema operativo que tengamos en nuestro ordenador. Esto nos descarga un archivo ejecutable que instalará la herramienta de desarrollo.

En segundo lugar, la aplicación se comunica con la placa a través de uno de los puertos USB del ordenador, empleando para ello un protocolo de comunicación serie. Para poder realizar esa comunicación, la placa dispone de un pequeño integrado encargado de esa función. En la placa NodeMCU con ESP32, esta función corresponde al chip CP2102. Si el ordenador no detecta automáticamente la placa al conectarla por USB, **necesitaremos instalar los drivers correspondientes.** Estos drivers podemos descargarlos desde la página del fabricante.

En tercer lugar, antes de poder programar el NodeMCU, **deberás agregar soporte para la placa ESP32** en el IDE de Arduino. Para ello, ve a Herramientas > Placa > Gestor de Tarjetas y busca "ESP32". Instala el paquete de soporte hecho por el fabricante del microcontrolador, “Espressif Systems”, para esa placa. Ahora deberías tener disponible un listado de placas para escoger entre las que se encuentra "**ESP32-WROOM-DA**" que es la que utilizaremos a lo largo de la guía.



Por último, solamente necesitas seleccionar el puerto en el que se encuentra la placa conectada al ordenador **mediante un cable micro USB a USB tipo A**. Los puertos seriales se seleccionan en la parte superior del IDE de Arduino, en la pestaña Tools > Port seleccionamos el puerto donde se ha conectado la placa.

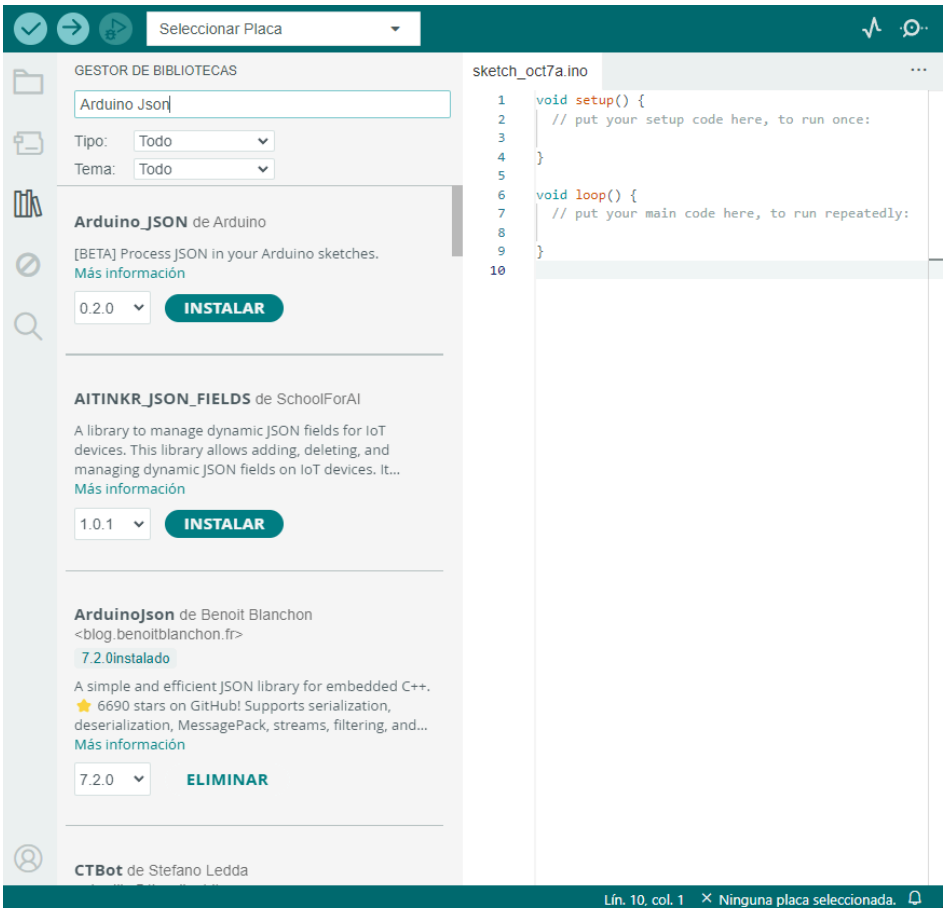


Si no sabes con certeza qué puerto está asignado al NodeMCU, puedes desconectar y volver a conectar la placa para identificarlo. En Windows, el puerto aparecerá como COMX (donde X es el número de puerto). En otros sistemas operativos, como macOS o Linux, los puertos pueden mostrarse como /dev/ttyUSBX o /dev/ttySX. Al desconectar y reconectar la placa, observa qué puerto aparece o desaparece de la lista para identificar el correspondiente.

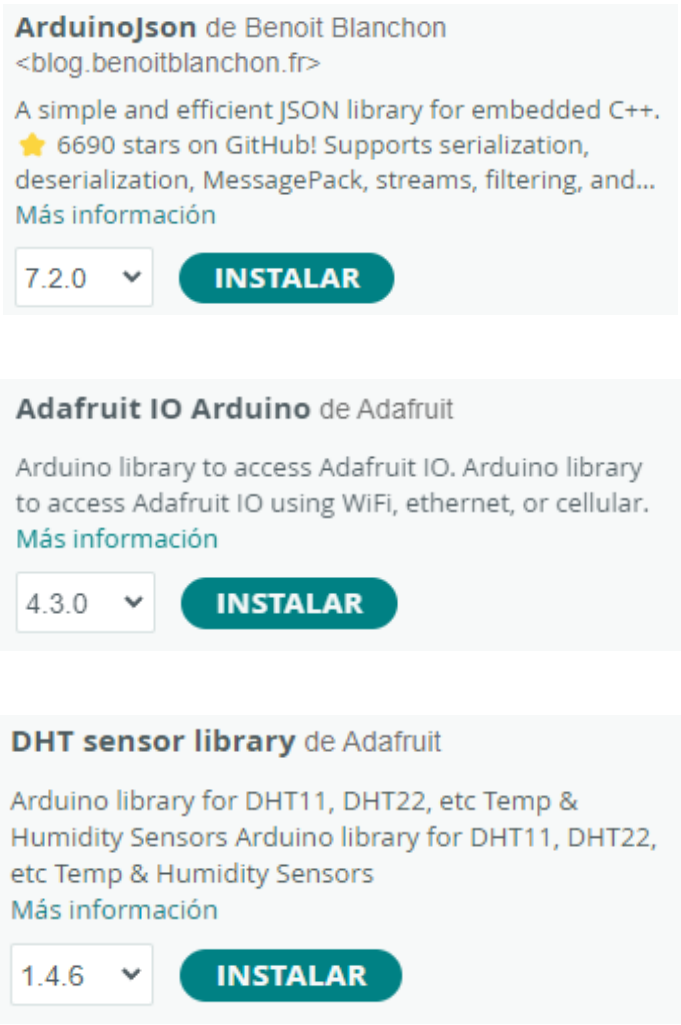
3. 2. Librerías necesarias

Una librería de Arduino IDE es un conjunto de código preescrito que facilita la programación al proporcionar funciones y procedimientos listos para usar. Estas librerías agrupan comandos y herramientas que permiten interactuar con componentes y módulos específicos, como sensores, motores, pantallas, y más, sin necesidad de escribir todo el código desde cero. Al incluir una librería en tu proyecto, puedes simplificar tareas complejas y hacer que tu código sea más limpio y fácil de entender.

En este punto, necesitarás instalar una serie de librerías que son necesarias para que el código del microcontrolador que está disponible (en el próximo apartado explicaremos cómo descargarlo) funcione correctamente. Estas librerías permitirán obtener adecuadamente las lecturas de los sensores y facilitará la conexión de los Demostradores con internet.



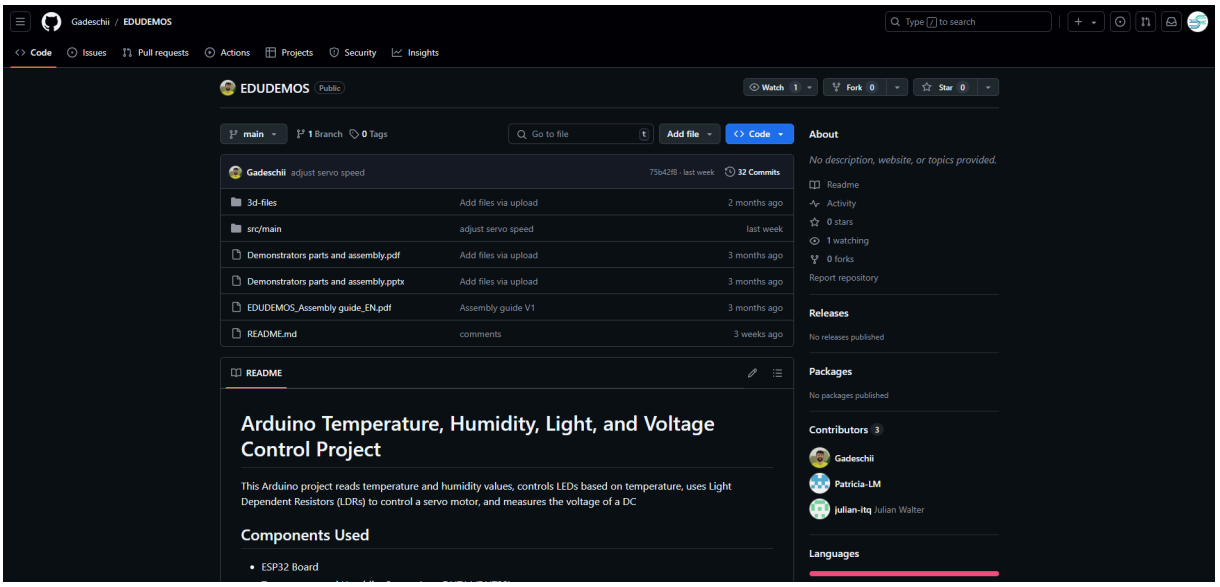
Dentro del IDE de Arduino, dirígete a la pestaña Tools > Manage libraries, busca las librerías que son necesarias para el código; ***Arduino Json, Adafruit_WiFi, DHT sensor library***, ***ESP32Servo,*** e instálalas.



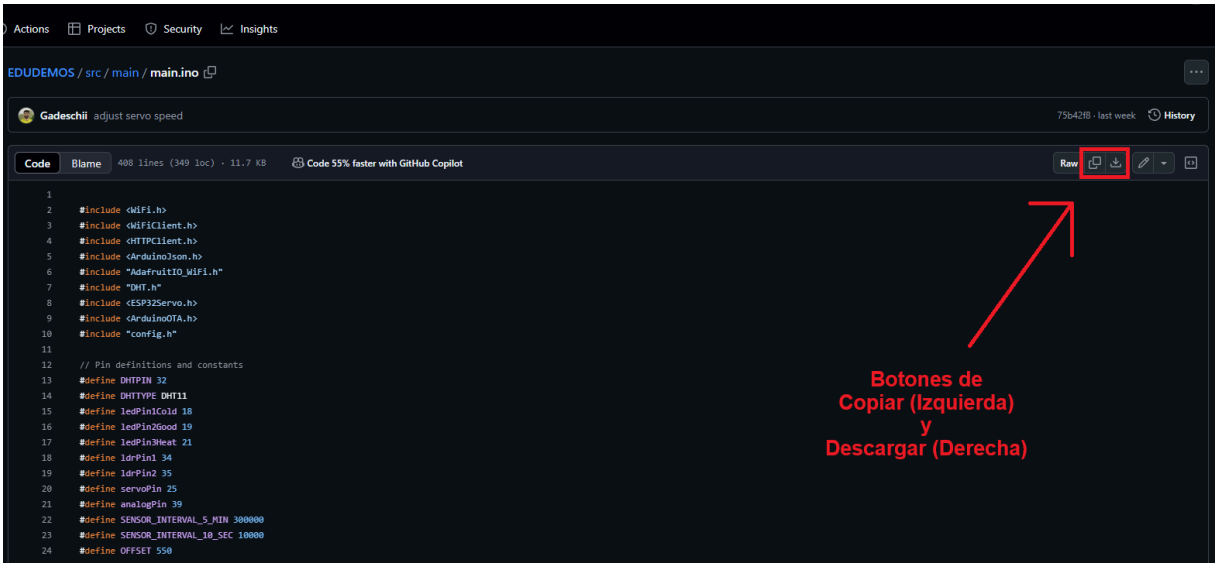
3. 3. Encontrar y descargar el código

El código de los Demostradores puedes encontrarlo en el **repositorio oficial de EDUDEMOS** en GitHub, que es una plataforma en línea que permite a desarrolladores almacenar, gestionar y colaborar en proyectos de programación. Los repositorios de los usuarios sirven para alojar sus códigos, colaborar con otros desarrolladores y gestionar versiones del software.

Visita la página oficial de GitHub y busca el repositorio de EDUDEMOS. Deberías ver una página como la siguiente:



Podrás encontrar el código fuente de los Demostradores dentro de la carpeta ***src/main/*** en el archivo ***main.ino.*** Dentro de GitHub puedes abrirlo para visualizar el programa. Puedes obtener el código de dos maneras, descargando el archivo con el botón de descarga o copiando las líneas de código y pegándolas en un archivo vacío de tu Arduino IDE con el botón de copiar.



En el repositorio dispones de un archivo llamado README.md que explica el código y las funciones que utiliza. Puedes consultarlo cuando quieras.

3. 4. Funcionamiento del código

El código fuente implementado, para un microcontrolador ESP32 en este caso, permite la obtención de las medidas de temperatura y humedad, control de LEDs en función del valor de la temperatura, el uso de LDR para controlar un servomotor y medir un voltaje de corriente continua generado. A continuación, se desglosa y explica el código por partes para dar una visión general del funcionamiento del mismo.

En la función *setup()*, se inicializan las comunicaciones seriales, se configura la conexión WiFi, se habilitan las actualizaciones OTA (Over The Air), y se preparan tanto el sensor DHT como los LEDs y el servomotor para su uso. Además, se crean tareas de FreeRTOs que gestionan la lectura de datos del sensor y del voltaje generado por el motor.

```
void setup() {
  Serial.begin(115200); // Start serial communication at 115200 baud rate
  analogReadResolution(12); // ESP32 has a 12-bit ADC
  analogSetAttenuation(ADC_11db); // Set attenuation for higher sensitivity
  pinMode(4, INPUT);

  setupWiFi(); // Setup WiFi connection
  setupOTA(); // Setup OTA for remote updates
  setupDHT(); // Setup DHT sensor
  setupLEDsAndServo(); // Setup LED pins and servo

  // Create tasks
  xTaskCreatePinnedToCore(
    readSensorTask1,    // Task function
    "ReadSensorTask1", // Name of the task
    4096,              // Stack size
    NULL,               // Task input parameter
    1,                  // Priority of the task
    NULL,               // Task handle
    0                   // Core where the task should run
  );
```

```
  xTaskCreatePinnedToCore(
    readSensorTask2,    // Task function
    "ReadSensorTask2", // Name of the task
    4096,              // Stack size
    NULL,               // Task input parameter
    1,                  // Priority of the task
    NULL,               // Task handle
    1                   // Core where the task should run
  );

  xTaskCreatePinnedToCore(
    readDcMotorVoltageTask,    // Task function
    "readDcMotorVoltageTask", // Name of the task
    4096,              // Stack size
    NULL,               // Task input parameter
    1,                  // Priority of the task
    NULL,               // Task handle
    1                   // Core where the task should run
  );

  xTaskCreatePinnedToCore(
    readSolarVoltageTask,    // Task function
    "readSolarVoltageTask", // Name of the task
    4096,              // Stack size
    NULL,               // Task input parameter
    1,                  // Priority of the task
    NULL,               // Task handle
    1                   // Core where the task should run
  );
}
```

El ciclo principal del programa, la función *loop()*, se encarga de manejar las actualizaciones OTA y de mantener la conexión a Adafruit IO de manera continua.

```
void loop() {
  handleOTA(); // Handle OTA updates
  io.run(); // Maintain connection to Adafruit IO
}
```

La función *readDcMotorVoltage()* es responsable de leer la tensión generada por un motor de corriente continua mediante un sensor de voltaje, convirtiendo esta medida en microvoltios. Los datos se envían a Adafruit IO después de realizar múltiples lecturas para reducir el error, y se aplica un umbral mínimo para filtrar el ruido no deseado. De manera similar, la función *readSolarVoltage()* está diseñada para medir la salida de detector de tensión al que está conectado un panel solar. Al igual que en la función anterior, se realizan múltiples lecturas para minimizar el ruido y mejorar la precisión de los datos.

```
void readDcMotorVoltage() {
  // Take multiple readings to reduce noise
  const int numReadings = 1000;
  unsigned long sum = 0;

  for (int i = 0; i < numReadings; i++) {
    int reading = analogRead(DcMotorPin);
    sum += reading;
  }

  float averageReading = sum / (float)numReadings;
  Serial.print("Average ADC reading: ");
  Serial.println(averageReading);  // Display the average value

  // Calculate voltage
  float voltage = (averageReading / 4095.0) * 3.3;  // For 12-bit ADC (0-4095)
  Serial.print("Measured voltage (V): ");
  Serial.println(voltage);  // Display the measured voltage

  // Adjust for voltage divider
  float inputVoltage = voltage / (R2 / (R1 + R2));

  // Convert to microvolts
  float microvoltsMotorDc = inputVoltage * 1000000.0;
  Serial.print("Input voltage (µV): ");
  Serial.println(microvoltsMotorDc);

  // Apply a minimum threshold to filter noise
  const float threshold = 10.0;  // Threshold in microvolts
  if (microvoltsMotorDc < threshold) {
    microvoltsMotorDc = 0;
  }
```

```
  // Check if the microvolts value has changed
  if (microvoltsMotorDc != lastMicrovolts) {
    Serial.print("Measured voltage: ");
    Serial.print(microvoltsMotorDc, 2);  // Display with 2 decimal places
    Serial.println(" µV");

    // Send microvolts to Adafruit IO
    microvoltsMotorDcFeed->save(microvoltsMotorDc);

    // Update the last microvolts value
    lastMicrovolts = microvoltsMotorDc;
  }
}

void readSolarVoltage() {
  // Take multiple readings to reduce noise
  const int numReadings = 1000;
  unsigned long sum = 0;

  for (int i = 0; i < numReadings; i++) {
    int reading = analogRead(SolarPin);
    sum += reading;
  }

  float averageReading = sum / (float)numReadings;
  Serial.print("Average ADC reading: ");
  Serial.println(averageReading);  // Display the average value

  // Calculate voltage
  float voltage = (averageReading / 4095.0) * 3.3;  // For 12-bit ADC (0-4095)
  Serial.print("Measured voltage (V): ");
  Serial.println(voltage);  // Display the measured voltage

  // Adjust for voltage divider
  float inputVoltage = voltage / (R2 / (R1 + R2));

  // Convert to microvolts
  float microvoltsSolar = inputVoltage * 1000000.0;
  Serial.print("Input voltage (µV): ");
  Serial.println(microvoltsSolar);

  // Apply a minimum threshold to filter noise
  const float threshold = 10.0;  // Threshold in microvolts
  if (microvoltsSolar < threshold) {
```

```
    microvoltsSolar = 0;
  }

  // Check if the microvolts value has changed
  if (microvoltsSolar != lastMicrovolts) {
    Serial.print("Measured voltage: ");
    Serial.print(microvoltsSolar, 2);  // Display with 2 decimal places
    Serial.println(" µV");

    // Send microvolts to Adafruit IO
    microvoltsSolarFeed->save(microvoltsSolar);

    // Update the last microvolts value
    lastMicrovolts = microvoltsSolar;
  }
}
```

El programa utiliza varias tareas de FreeRTOs. La tarea *readSensorTask1()* se ejecuta periódicamente para leer y procesar los datos de temperatura y humedad desde el sensor DHT. Estos datos se envían a Adafruit IO y, según la temperatura medida, se controlan los LEDs. Por su parte, la tarea *readSensorTask2()* lee valores de dos LDRs, calcula la diferencia de luz entre ellos y ajusta el servomotor en consecuencia. También envía los valores promedio de los LDRs a Adafruit IO. Otra tarea, *readDcMotorVoltageTask()*, se dedica a leer y procesar periódicamente el voltaje del motor y a enviar estos datos a Adafruit IO. Por último, *readSolarVoltageTask()*, realiza los mediciones obtenidas del detector de tensión y las actualiza en la plataforma con el resto de la información.

```
void readSensorTask1(void * parameter) {
  for (;;) {
    Serial.println("----------------------- Measured Temperature / Humidity
-----------------------  ");
    updateSensorData(); // Read and process sensor data
    vTaskDelay(SENSOR_INTERVAL_5_MIN / portTICK_PERIOD_MS); // Delay for 5 minutes
  }
}

void readSensorTask2(void * parameter) {
  for (;;) {
    Serial.println("---------------------------- Measured LDR and Servo
----------------------------  ");
    updateLDRAndServo(); // Read LDR values and control servo
```

```
    vTaskDelay(SENSOR_INTERVAL_10_SEC / portTICK_PERIOD_MS); // Delay for 10
seconds
  }
}

void readSolarVoltageTask(void * parameter) {
  for (;;) {
    Serial.println("----------------------------- Measured Solar (V)
-----------------------------  ");
    readSolarVoltage(); // Read and process voltage
    vTaskDelay(SENSOR_INTERVAL_10_SEC / portTICK_PERIOD_MS); // Delay for 10
seconds
  }
}

void readDcMotorVoltageTask(void * parameter) {
  for (;;) {
    Serial.println("----------------------------- Measured DC Motor (V)
-----------------------------  ");
    readDcMotorVoltage(); // Read and process voltage
    vTaskDelay(SENSOR_INTERVAL_5_MIN / portTICK_PERIOD_MS); // Delay for 10 seconds
  }
}
```

En cuanto a la conectividad, la función *setupWiFi()* conecta el ESP32 a una red WiFi y mantiene la conexión con Adafruit IO. Para las actualizaciones remotas, la función *setupOTA()* inicializa el sistema de actualizaciones OTA utilizando la librería ArduinoOTA, definiendo los controladores para gestionar estos eventos. La función *handleOTA()* se encarga de manejar las actualizaciones OTA al invocar *ArduinoOTA.handle()*.

```
void setupWiFi() {
  int retries = 0;
  const int maxRetries = 5;

  WiFi.begin(WIFI_SSID, WIFI_PASS);
  Serial.print("Connecting to WiFi");

  while (WiFi.status() != WL_CONNECTED && retries < maxRetries) {
    delay(1000);
    Serial.print(".");
    retries++;
  }
```

```
  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("\nConnected to WiFi");
    // Send microvolts to Adafruit IO
    wifiFeed->save(WIFI_SSID);
  } else {
    Serial.println("\nWiFi connection failed. Restarting...");
    ESP.restart();
  }

  connectToAdafruitIO();
}

void handleOTA() {
  ArduinoOTA.handle(); // Handle OTA updates

  //check if there are new client
  // Verificar si hay un nuevo cliente telnet
  if (telnetServer.hasClient()) {
    if (!telnetClient || !telnetClient.connected()) {
      if (telnetClient) telnetClient.stop();
      telnetClient = telnetServer.available();
      Serial.println("New Telnet client connected");
    } else {
      telnetServer.available().stop();
    }
  }

  //Send data to the telnet client server
  if (telnetClient && telnetClient.connected()) {
    while (Serial.available()) {
      telnetClient.write(Serial.read());
    }
    while (telnetClient.available()) {
      Serial.write(telnetClient.read());
    }
  }

}

void setupOTA() {
  // Start telnet server
  telnetServer.begin();
  telnetServer.setNoDelay(true);
```

```
  // Initialize ArduinoOTA for over-the-air updates
  ArduinoOTA.setHostname("ESP32-OTA");
  ArduinoOTA.setPassword("admin"); // Uncomment to set a password for OTA updates
  // Define OTA event handlers
  ArduinoOTA.onStart([]() {
    String type;
    if (ArduinoOTA.getCommand() == U_FLASH) {
      type = "sketch";
    } else { // U_SPIFFS
      type = "filesystem";
    }
    Serial.println("Start updating " + type);
  });
  ArduinoOTA.onEnd([]() {
    Serial.println("\nOTA Update Finished");
  });
  ArduinoOTA.onProgress([](unsigned int progress, unsigned int total) {
    Serial.printf("Progress: %u%%\r", (progress / (total / 100)));
  });
  ArduinoOTA.onError([](ota_error_t error) {
    Serial.printf("Error[%u]: ", error);
    if (error == OTA_AUTH_ERROR) {
      Serial.println("Auth Failed");
    } else if (error == OTA_BEGIN_ERROR) {
      Serial.println("Begin Failed");
    } else if (error == OTA_CONNECT_ERROR) {
      Serial.println("Connect Failed");
    } else if (error == OTA_RECEIVE_ERROR) {
      Serial.println("Receive Failed");
    } else if (error == OTA_END_ERROR) {
      Serial.println("End Failed");
    }
  });
  ArduinoOTA.begin(); // Start the OTA service
  Serial.println("OTA Ready");
}
```

Dentro del propio código existen unas variables constantes definidas que contienen las credenciales para la red Wi-Fi a utilizar y las claves para la plataforma Adafruit IO correspondientes. Los valores de estas variables puedes cambiarlos en la línea 30 en adelante, como se muestra a continuación, si lo necesitas.

```
// Set up your wifi Credencial
#define WIFI_SSID "wifi_name"
#define WIFI_PASS "Wifi_Password"

// Set up your API key for Adafruit IO
#define AIO_USERNAME "Adafuit_name"
#define AIO_KEY "Adafruit_Password"
```

El sensor DHT es inicializado en la función *setupDHT()*, y los pines de los LEDs y el servomotor se configuran en la función *setupLEDsAndServo()*, que también conecta el motor a su pin correspondiente.

```
void setupDHT() {
  dht.begin(); // Start the DHT sensor
}
```

Finalmente, hay dos funciones clave para la actualización de datos: *updateSensorData()* lee los valores de temperatura y humedad del sensor DHT, envía esta información a Adafruit IO y controla los LEDs en función de la temperatura, y *updateLDRAndServo()* lee los valores de dos LDRs, ajusta el servomotor según la diferencia de luz y envía los valores promedio de las LDRs a Adafruit IO.

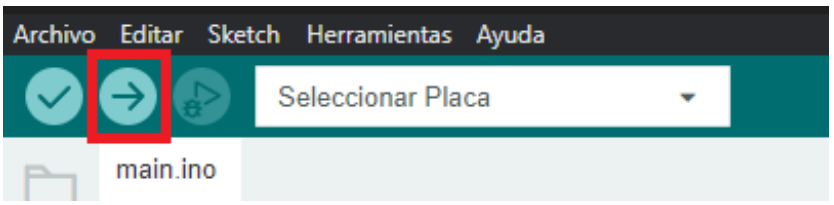
```
void updateSensorData() {
  // Read values from DHT sensor
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  // Check if readings are valid
  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  // Send values to Adafruit IO
  temperature->save(t);
  humidity->save(h);

  // Print sensor values
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print(" °C, Humidity: ");
```

```
  Serial.print(h);
  Serial.println(" %");

  // Example LED control based on temperature
  if (t < 20) {
    digitalWrite(ledPin1Cold, HIGH);
    digitalWrite(ledPin2Good, LOW);
    digitalWrite(ledPin3Heat, LOW);
     // Send to the corresponding feed (Cold LED)
    ledColdFeed->save(1);  // 1 means the LED is ON
    ledGoodFeed->save(0);  // OFF
    ledHighFeed->save(0);  // OFF
    Serial.println("Cold is HIGH");

  } else if (t >= 20 && t < 25) {
    digitalWrite(ledPin1Cold, LOW);
    digitalWrite(ledPin2Good, HIGH);
    digitalWrite(ledPin3Heat, LOW);
    // Send to the corresponding feed (Good LED)
    ledColdFeed->save(0);  // OFF
    ledGoodFeed->save(1);  // ON
    ledHighFeed->save(0);  // OFF
       Serial.println("Good is HIGH");
  } else {
    digitalWrite(ledPin1Cold, LOW);
    digitalWrite(ledPin2Good, LOW);
    digitalWrite(ledPin3Heat, HIGH);
    // Send to the corresponding feed (Heat LED)
    ledColdFeed->save(0);  // OFF
    ledGoodFeed->save(0);  // OFF
    ledHighFeed->save(1);  // ON
    Serial.println("Heat is HIGH");
  }
}

void updateLDRAndServo() {
  // Read LDR values
  ldrValue1 = analogRead(ldrPin1) ;
  ldrValue2 = analogRead(ldrPin2)- OFFSET;

  // Invert readings so low values mean darkness and high values mean light
  ldrValue1 = 4095 - ldrValue1;
  ldrValue2 = 4095 - ldrValue2;
  averageLdrValue = (ldrValue1 + ldrValue2) / 2;
```

```
    // Print readings for debugging
    Serial.print("LDR 1: ");
    Serial.print(ldrValue1);
    Serial.print(" | LDR 2: ");
    Serial.println(ldrValue2);
    Serial.print("Average LDR: ");
    Serial.println(averageLdrValue);

    // Send the average value to Adafruit IO
    ldrFeed->save(averageLdrValue);
    ldr1Feed-> save (ldrValue1);
    ldr2Feed -> save (ldrValue2);

    // Compare LDR values and move the servo based on the difference
    if (ldrValue1 > ldrValue2 + tolerance) {
      if (servoPos > 0) {
        servoPos -= servoStep;  // Move the servo to the left
        Serial.println("Moving left");
      }
    } else if (ldrValue2 > ldrValue1 + tolerance) {
      if (servoPos < 180) {
        servoPos += servoStep;  // Move the servo to the right
        Serial.println("Moving right");
      }
    }

    // Ensure the servo stays within the [0, 180] range
    servoPos = constrain(servoPos, 0, 180);

    // Move the servo to the new position only if it has changed
    myServo.write(servoPos);

    // Print the servo angle
    Serial.print("Servo angle: ");
    Serial.println(servoPos);

    // Pause to give the servo time to move
    delay(30);  // Increase the delay for more response time for the servo
}
```

3. 5. Programación del microcontrolador

Para programar la placa del ESP32 con el código facilitado, lo primero será tener el NodeMCU conectado al ordenador con un cable Micro USB a USB tipo A. Posteriormente, como ya se ha indicado, seleccionaremos el puerto COM correspondiente a nuestra placa y usaremos el botón de *Subir código* del IDE de Arduino, que compila el programa antes de enviarlo a la placa programable. Justo a su izquierda, con el símbolo del *tick*, está el botón de solamente compilar. La compilación sirve para convertir el código escrito en un lenguaje de programación como es C/C++ en un formato que el microcontrolador pueda entender y ejecutar, revisando y señalando durante el proceso de que no haya errores de sintaxis o problemas que impidan que el programa se ejecute.



Si te aparece algún error relacionado con el proceso de uploading o carga del código en la placa, puedes verificar lo siguiente:

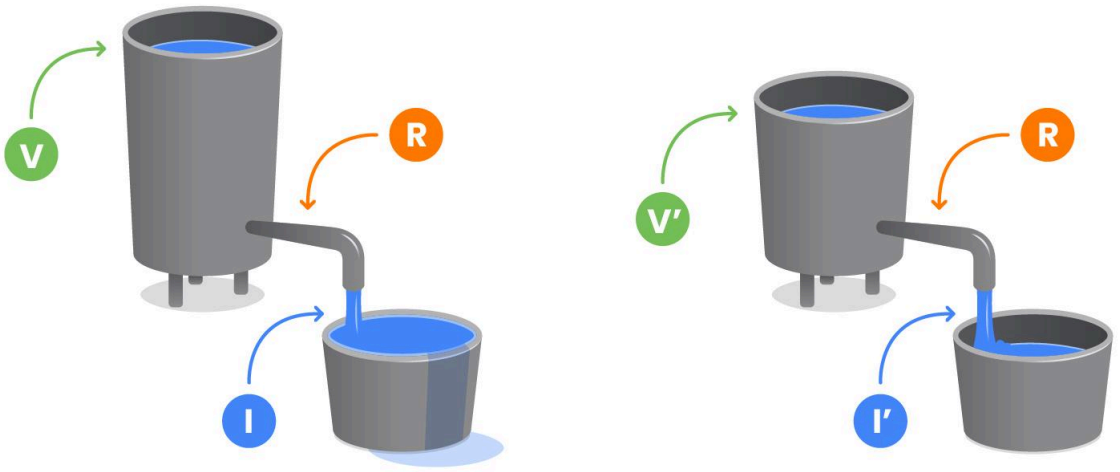
- Comprobar que el cable USB esté bien conectado en ambos extremos.
- Intenta utilizar otro cable USB, algunos cables solo sirven para cargar, no transmiten datos.
- Intenta utilizar otro puerto USB de tu ordenador.
- Asegúrate de que has seleccionado el puerto COM adecuado.
- Algunos modelos de NodeMCU necesitan que se les ponga en modo programación de manera manual accionando el botón de BOOT mientras comienza el proceso de carga del código.
- Comprueba que tienes instalado correctamente el driver necesario para programar la placa NodeMCU (el CP210x).

4. INTRODUCCIÓN A ELECTRÓNICA

La electrónica es una rama de la física y la ingeniería que se centra en el estudio, diseño y aplicación de dispositivos y sistemas que funcionan mediante el control de electrones y otras partículas cargadas eléctricamente. A través de la electrónica, se construyen circuitos que permiten la manipulación de señales, lo que facilita la creación de tecnologías esenciales como computadoras, teléfonos, automóviles y una amplia variedad de dispositivos de uso diario.

4. 1. Ley de Ohm

En el ámbito de la electrónica y la electricidad, hay una ley clave conocida como la **Ley de Ohm**, que establece la relación entre las tres magnitudes más importantes de un circuito: **el voltaje (tensión), la corriente (intensidad) y la resistencia**. Entender el significado de estas magnitudes y cómo se relacionan entre sí puede parecer complicado al principio, pero se vuelve mucho más fácil si se compara un circuito eléctrico con un sistema de agua, utilizando una analogía para explicar su funcionamiento.



- **Voltaje (Tensión):** Es la presión del agua en las tuberías. El voltaje es lo que "empuja" a los electrones (la corriente) a moverse por el circuito. Se mide en **voltios (V)**.

- **Corriente (Intensidad):** Es el caudal de agua que fluye por las tuberías. En un circuito eléctrico, es el flujo de electrones que se mueve gracias al voltaje. Se mide en **amperios (A)**.

- **Resistencia:** Es el tamaño o el estrechamiento de la tubería. La resistencia es lo que "dificulta" o "frena" el paso de la corriente. Si la resistencia es alta, menos corriente puede pasar. Se mide en **ohmios (Ω)**.

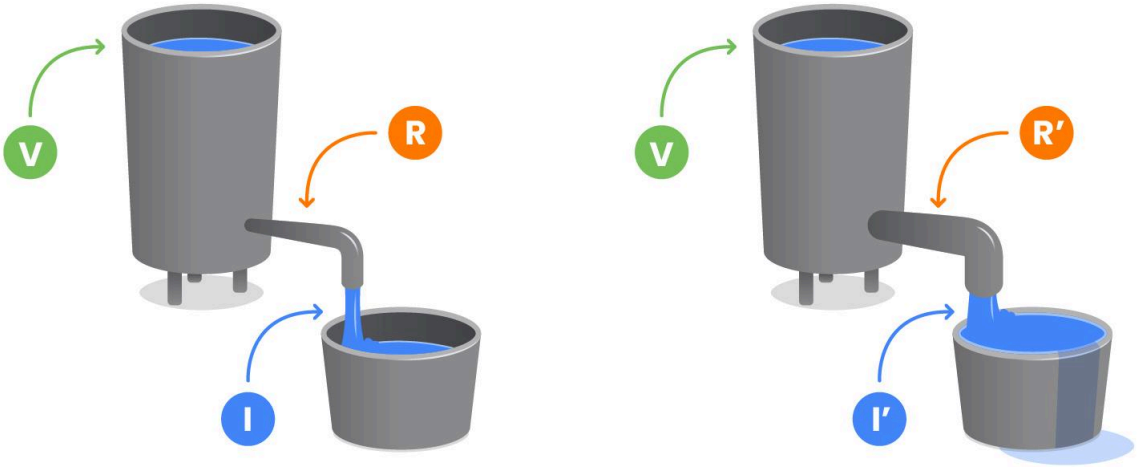
La Ley de Ohm dice que la relación entre estas tres magnitudes sigue una regla sencilla:

$$V = I \cdot R$$

O de otra manera:

- Si aumentas el voltaje (la presión), más corriente fluirá.

- Si aumentas la resistencia (estrechas la tubería), menos corriente pasará, a menos que aumentes también el voltaje.



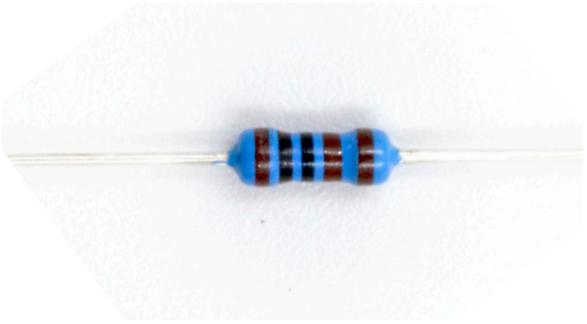
4. 2. Componentes electrónicos

Los componentes electrónicos son los elementos básicos que constituyen los circuitos eléctricos y electrónicos. Cada uno de ellos cumple una función específica dentro del circuito, permitiendo controlar y manipular el flujo de corriente eléctrica. Entre los componentes más comunes se encuentran las resistencias, que limitan el paso de corriente; los condensadores, que almacenan y liberan energía; los diodos, que permiten el paso de corriente en una sola dirección; y los transistores, que actúan como interruptores o amplificadores. Estos elementos, al combinarse en diversas configuraciones, hacen posible el funcionamiento de dispositivos electrónicos que usamos a diario.

4. 2. 1. Componentes básicos

- Resistencias

En electrónica, una resistencia (R) es un componente cuya función es limitar el flujo de corriente eléctrica dentro de un circuito. Su propósito es regular la cantidad de corriente (I) que circula por los demás componentes, protegiéndolos y asegurando el correcto funcionamiento del circuito.



El valor de la resistencia se indica mediante unas bandas de colores impresas en el propio componente. Estas combinaciones de colores permiten identificar el valor en ohmios de las resistencias, siguiendo el patrón establecido en la siguiente tabla:

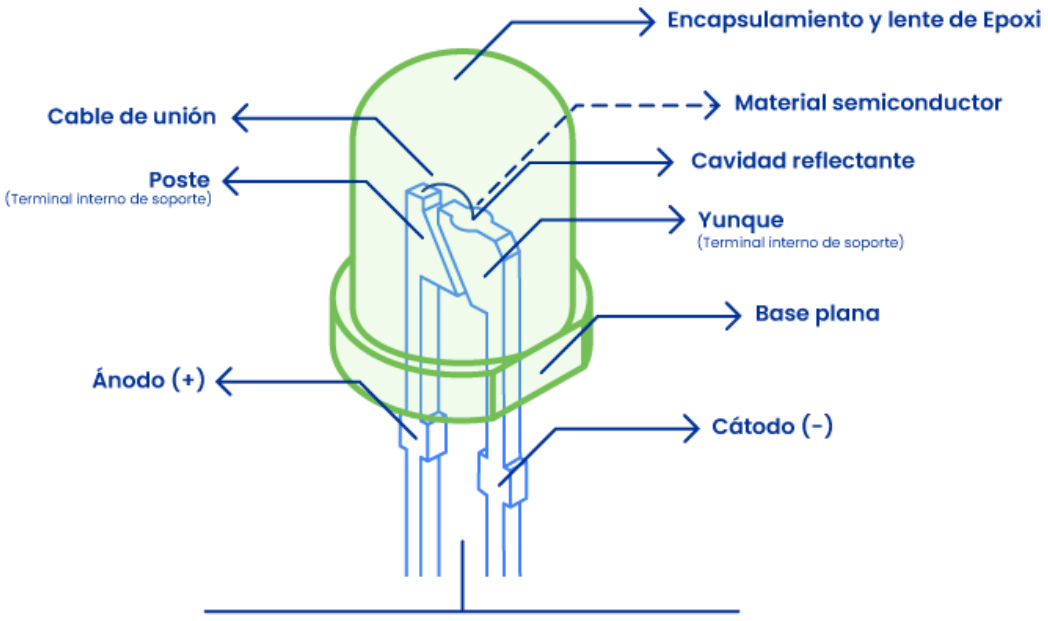
COLOR	BANDA 1	BANDA 2	BANDA 3	MULTIPLICADOR	TOLERANCIA
Negro	0	0	0	1 Ω	-
Marrón	1	1	1	10 Ω	± 1%
Rojo	2	2	2	100 Ω	± 2%
Naranja	3	3	3	1 kΩ	-
Amarillo	4	4	4	10 kΩ	-
Verde	5	5	5	100 kΩ	± 0.5%
Azul	6	6	6	1 MΩ	± 0.25%
Morado	7	7	7	10 MΩ	± 0.10%
Gris	8	8	8	-	± 0.05%
Blanco	9	9	9	-	-
Dorado	-	-	-	-	± 5%
Plateado	-	-	-	-	± 10%

Ten en cuenta que hay resistencias que tienen 3 bandas de color (Banda 1 y Multiplicador, además de la Tolerancia) y otras 4 bandas de color (Banda 1, Banda 2 y Multiplicador, además de la Tolerancia), incluso de 5 bandas. La Tolerancia se refiere al margen de error del valor real de la resistencia respecto al valor indicado por las bandas de colores.

- Diodos emisores de luz

Un LED (Diodo Emisor de Luz) es un tipo de diodo con la particular característica de que es capaz de emitir luz. Un diodo es un componente electrónico que permite la circulación de corriente en un solo sentido.

Está formado internamente por un material semiconductor con unión tipo p-n, por tanto, cuando se polariza en directa, es decir, con mayor tensión en el ánodo que en el cátodo, el diodo conduce electricidad y cuando se polariza en inversa bloquea el paso de corriente. Esta manera de funcionar del diodo hace que en cierta forma se parezca a un interruptor controlado por tensión. El diodo LED nos indica la polaridad de la conexión a simple vista con el tamaño de sus patas de conexión. La más larga coincide con el ánodo, o positivo, y la más corta se trata del cátodo, o negativo. En el caso particular del LED, se trata de un diodo que al conducir la electricidad su material semiconductor libera energía en forma de fotones, lo que hace que se ilumine.



Cuando un LED conduce electricidad ofrece muy poca resistencia al paso de corriente, por lo que la fuente de alimentación va a suministrar toda la corriente posible de forma descontrolada, lo que equivale a un cortocircuito que como poco supondrá un **alto riesgo de quemar el LED**. Para evitar esto, siempre debemos conectar una resistencia de valor adecuado (220 ohm o superior normalmente) en serie con el LED. Así controlamos el paso de corriente y evitamos que nada se rompa.

- Condensadores

Un condensador es un componente electrónico que almacena energía en forma de carga eléctrica. Consiste en dos placas conductoras separadas por un material aislante llamado dieléctrico. Cuando se aplica una tensión a través de las placas, se acumula carga en ellas, creando un campo eléctrico.

Los condensadores son utilizados en diversas aplicaciones, como en filtros, fuentes de alimentación, circuitos de temporización y para suavizar señales eléctricas. Su capacidad para almacenar y liberar energía rápidamente los hace esenciales en el diseño de circuitos electrónicos. Se mide en faradios (F), aunque comúnmente se utilizan unidades más pequeñas como microfaradios (μF) y nanofaradios (nF).

4. 2. 2. Alimentación de un circuito

La alimentación de un circuito electrónico se refiere al proceso de proporcionar la energía necesaria para que el circuito funcione correctamente. Esta energía puede provenir de diferentes fuentes, como baterías, adaptadores de corriente, paneles solares o fuentes de alimentación de red. Los aspectos clave relativos a la alimentación de un circuito electrónico son:

- **Tensión (Voltaje):** Es la fuerza que impulsa a los electrones a través del circuito. Cada componente electrónico tiene un rango específico de voltaje que necesita para operar. Por ejemplo, algunos microcontroladores funcionan a 5V, mientras que otros pueden necesitar 3.3V.

- **Corriente (Amperaje):** Es la cantidad de electrones que fluye por el circuito. La fuente de alimentación debe ser capaz de proporcionar suficiente

corriente para satisfacer las necesidades de todos los componentes del circuito.

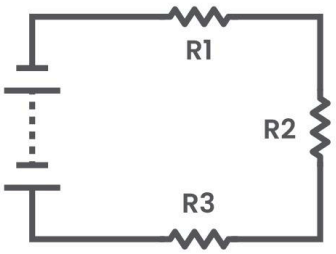
- **Estabilidad:** La alimentación debe ser estable y constante para evitar fluctuaciones que puedan dañar los componentes electrónicos o afectar su rendimiento. Esto es especialmente importante en circuitos sensibles.

- **Regulación:** Algunos circuitos requieren un voltaje específico y constante, por lo que se utilizan reguladores de voltaje para mantener la tensión en el nivel adecuado, independientemente de las variaciones en la carga o en la fuente de alimentación.

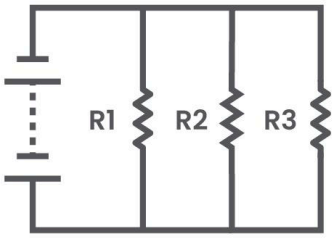
- **Tipos de alimentación:** Los circuitos pueden alimentarse con corriente continua (CC), como la que proporcionan las baterías, o corriente alterna (CA), como la que se obtiene de la red eléctrica. Algunos circuitos pueden necesitar un conversor para transformar la corriente alterna en corriente continua.

4. 2. 3. Configuración de componentes en un circuito

En electrónica, los componentes en un circuito se pueden conectar de diversas maneras, y cada configuración afecta el comportamiento del circuito. Las configuraciones más comunes son en **serie**, en **paralelo** y **mixtas**.



Circuito serie



Circuito paralelo

En una configuración en **serie**, los componentes se conectan uno tras otro, formando un solo camino para la corriente. Esto significa que la misma corriente fluye a través de todos los componentes.

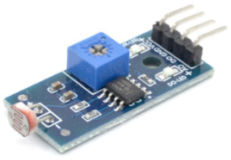
En una configuración en **paralelo**, los componentes se conectan de manera que cada uno forma su propio camino para la corriente.

En una configuración **mixta**, los componentes están conectados en una combinación de series y paralelo. Esta configuración permite una mayor flexibilidad en el diseño del circuito y se utiliza para satisfacer requisitos específicos de voltaje y corriente.

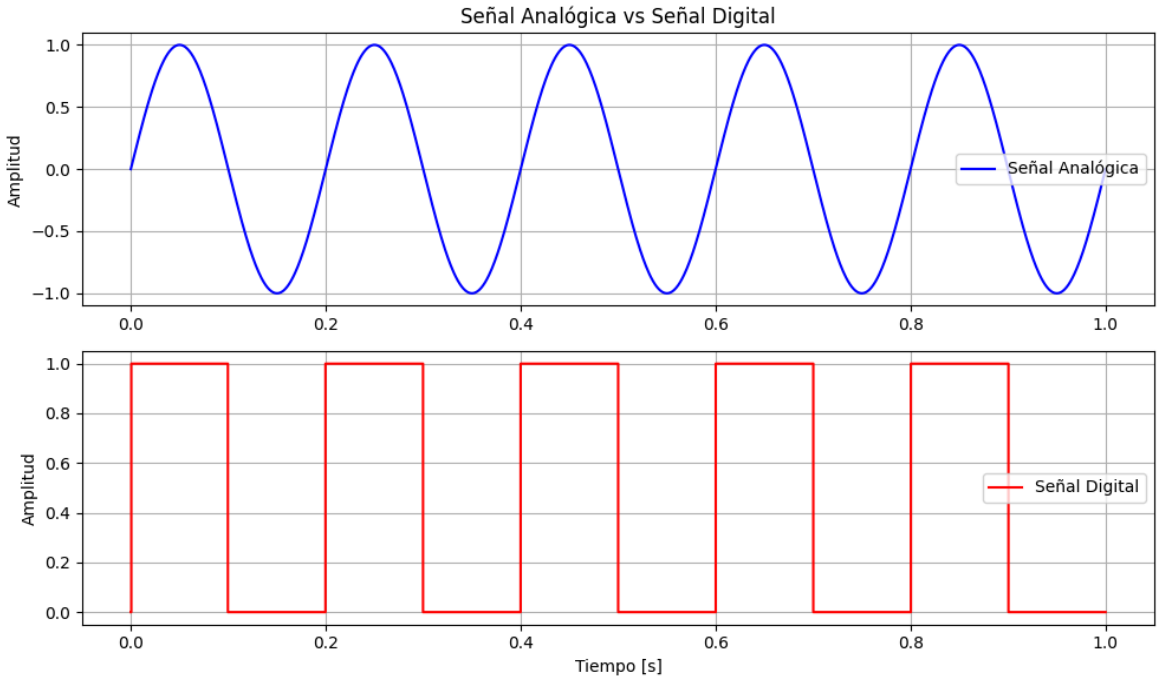
4. 2. 4. Sensores

Un sensor es un dispositivo que detecta cambios en el entorno y convierte esa información en una señal eléctrica o en datos que pueden ser interpretados por un sistema electrónico. Los sensores son fundamentales en una amplia variedad de aplicaciones, desde sistemas de automatización y robótica hasta dispositivos de medición y monitoreo.



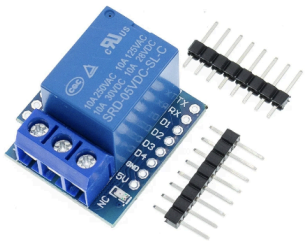
Los sensores son capaces de captar diferentes tipos de estímulos, como luz, temperatura, presión, humedad, movimiento, sonido, y más. Cada tipo de sensor está diseñado para medir una magnitud específica actuando como transductores, es decir, convirtiendo una forma de energía (como la energía térmica, mecánica o luminosa) en una señal eléctrica que puede ser procesada por otros dispositivos, como microcontroladores. Esta señal generada puede ser analógica (una variación continua, como un voltaje) o digital (un valor discreto, como un "sí" o "no").



Los sensores se utilizan en una amplia gama de aplicaciones, como en la domótica para controlar el clima de una casa, en automóviles para sistemas de seguridad, en dispositivos portátiles para monitorear la salud y en la industria para el control de procesos. Por ejemplo, un DHT-11 o DHT-22 es un tipo de sensor de temperatura que entrega con una señal digital los valores medidos, mientras que una Fotorresistencia, es un tipo de sensor que varía su resistencia eléctrica en función de la cantidad de luz que incide sobre ella, pudiendo obtener una señal de tipo analógico.

4. 2. 5. Actuadores

Un actuador es un dispositivo que convierte una señal eléctrica de control en movimiento o una acción física dentro de un sistema. Los actuadores trabajan en conjunto con los sensores en los sistemas de control. Mientras los sensores detectan y miden cambios, los actuadores responden a esa información para ejecutar acciones y modificar el entorno o el sistema controlados mediante señales que pueden ser analógicas o digitales, enviadas por un sistema de control como un microcontrolador o un ordenador. Estas señales indican al actuador qué tarea debe realizar, ya sea abrir una válvula, mover un brazo robótico o girar un motor.

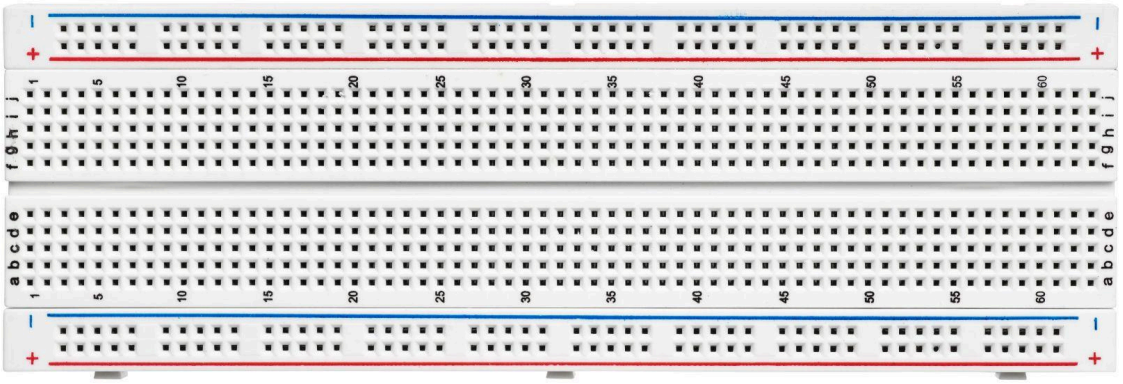
Existen diferentes tipos de actuadores:

- **Actuadores eléctricos:** Utilizan energía eléctrica para generar movimiento, como motores eléctricos, solenoides y servomotores.
- **Actuadores hidráulicos:** Funcionan con presión de fluidos para mover objetos, utilizados en sistemas que requieren gran fuerza, como en maquinaria pesada.
- **Actuadores neumáticos:** Emplean aire comprimido para generar movimiento, común en aplicaciones industriales y sistemas de automatización.

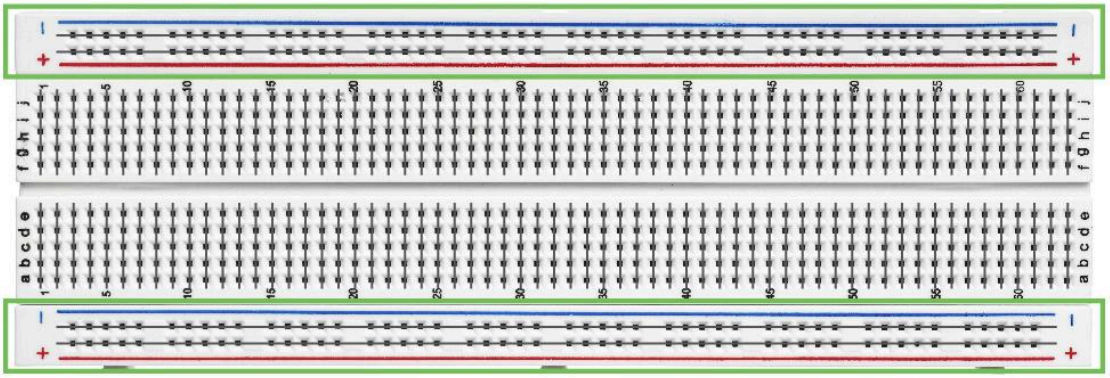
Los actuadores son esenciales en una amplia variedad de aplicaciones. En robótica, por ejemplo, permiten que los brazos o las ruedas se muevan, mientras que en la domótica controlan elementos como persianas o puertas automáticas. También están presentes en los automóviles, donde gestionan el movimiento de los asientos o la apertura de válvulas.

4. 3. Protoboard

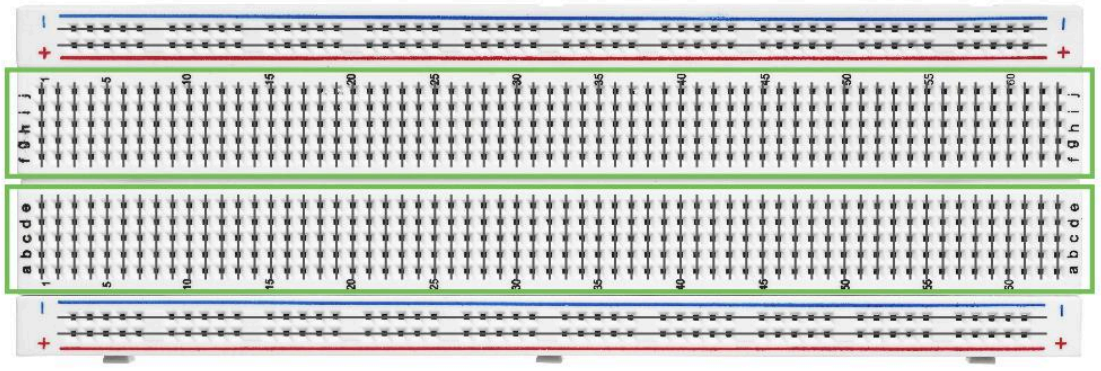
Una placa protoboard es una especie de "campo de trabajo" para crear circuitos electrónicos. Es un tablero lleno de pequeños orificios en los que puedes insertar diversos componentes como resistencias, LEDs, cables, … Estos orificios están conectados internamente de forma que, al colocar un componente en uno de ellos, se conecta automáticamente con otros agujeros siguiendo un patrón establecido.



La protoboard te permite experimentar con distintas configuraciones de circuitos sin necesidad de soldar los componentes de manera permanente. Puedes mover los elementos, reemplazarlos o quitarlos fácilmente sin causar daño.



La imagen muestra cómo están conectados internamente los agujeros de una protoboard. Las láminas de cobre en su interior permiten que ciertos grupos de agujeros estén conectados entre sí. En este caso, las líneas horizontales, que suelen ser los rieles de alimentación ubicados en los bordes de la placa, están conectadas eléctricamente a lo largo de toda su longitud. Esto significa que cualquier componente o cable que se inserte en esos agujeros compartirá la misma conexión de alimentación o tierra.

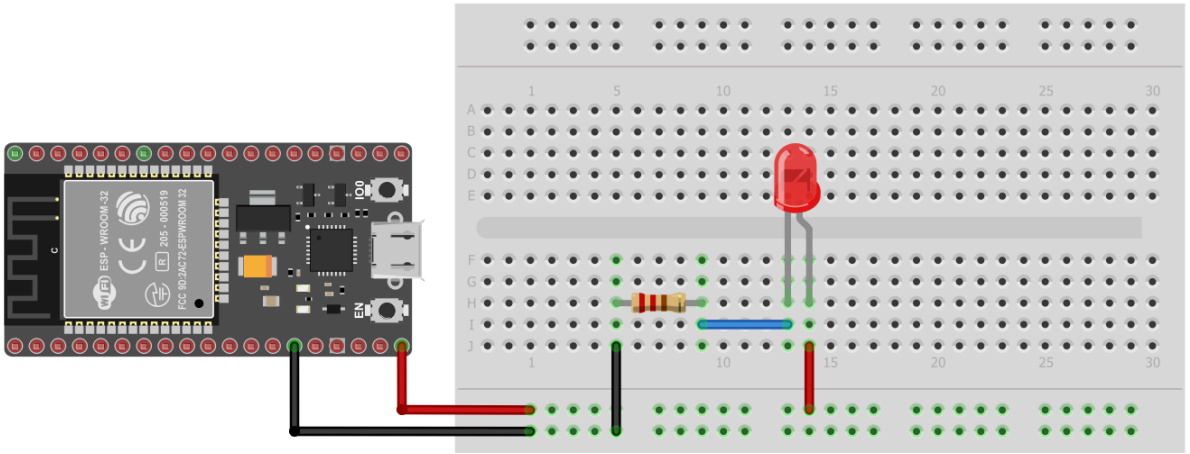


Por otro lado, los grupos de líneas verticales, que se encuentran en el área central de la protoboard, también están unidas eléctricamente entre sí. Estos grupos de conexiones están aislados entre sí, pero dentro de cada grupo, los componentes que coloques compartirán la misma conexión, facilitando la creación de circuitos complejos sin la necesidad de soldar. Este diseño de conexiones internas es lo que permite una fácil manipulación y modificación de circuitos experimentales en la protoboard.

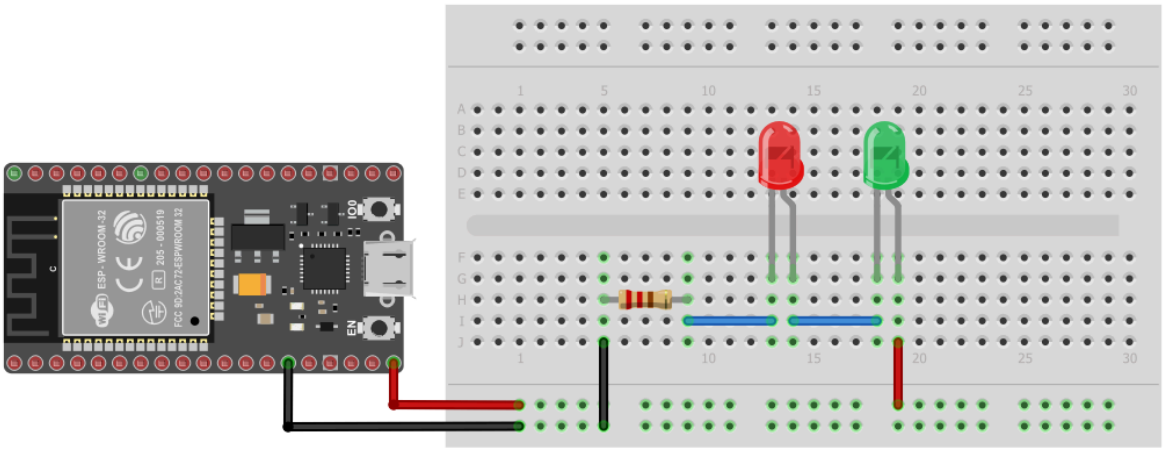
4. 3. 1. Actividades con Protoboard

En esta primera actividad, el objetivo es aprender a utilizar la protoboard y comprender el flujo de corriente en un circuito simple. Para lograrlo, puedes utilizar los pines de VIN y GND para dar alimentación al circuito en la placa de pruebas con los rieles de alimentación laterales, teniendo en cuenta que VIN es el terminal positivo y GND el negativo. Luego necesitarás un LED como indicador visual de energía y una resistencia para limitar la corriente. Conecta el LED en serie con la resistencia en cualquier zona de la placa de pruebas, asegurándote de conectar en la orientación correcta el diodo LED. Al completar el circuito y conectar el NodeMCU al ordenador, el LED debería encenderse, lo que indicará que la corriente está fluyendo correctamente.

Como elemento de repaso, ¿podrías decir el valor de la resistencia utilizada en el esquema?

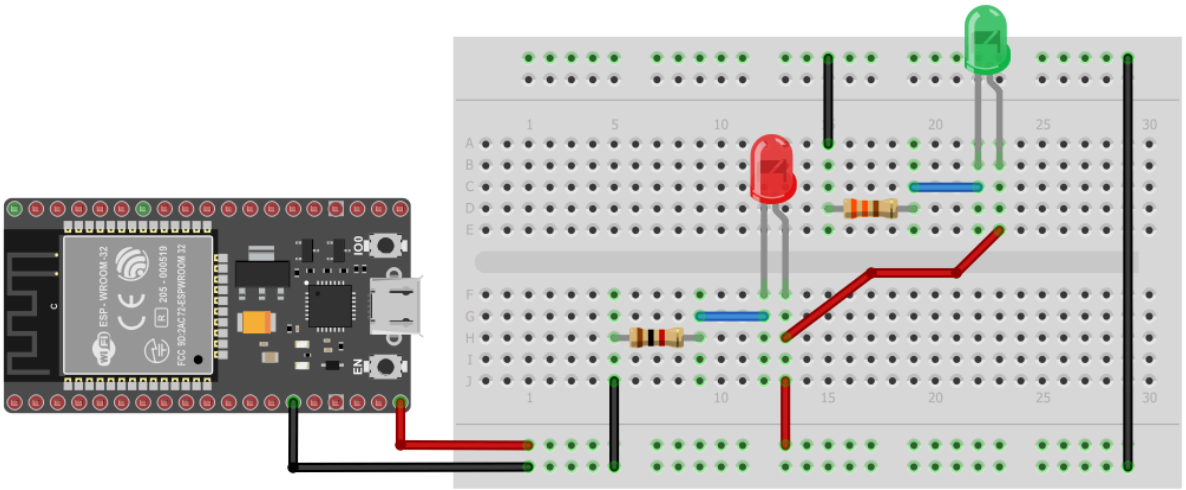


Esta siguiente actividad consiste en conectar en serie dos diodos LED junto con su resistencia limitadora. Une el terminal negativo de un diodo LED con el terminal positivo del siguiente, y posteriormente coloca la resistencia, uniendo ambos extremos del circuito a las líneas de alimentación. Ten en cuenta la orientación de ambos LEDs.



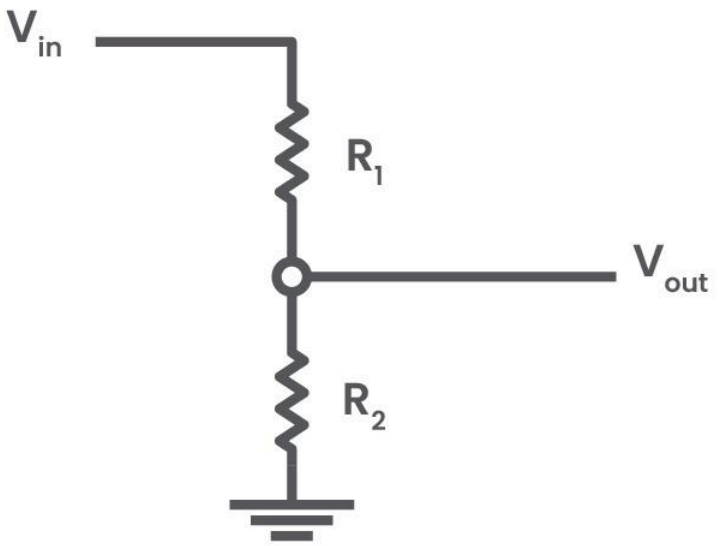
Para continuar practicando con la protoboard, vamos a montar los dos mismos LEDs del ejercicio anterior, pero en paralelo y con diferentes valores de resistencias limitadoras cada uno.

¿Podrías decir el valor de las resistencias que se han usado en esta ocasión viendo el esquema?



4. 3. 2. Divisor de tensión

El divisor de tensión es un circuito muy simple y útil que permite obtener una fracción del voltaje de entrada mediante el uso de dos resistencias conectadas en serie. Es común en aplicaciones donde se necesita reducir un voltaje para adaptarlo a un nivel que pueda ser utilizado por otros componentes.



Cuando se conectan dos resistencias en serie a una fuente de voltaje, la corriente que pasa por ellas es la misma, pero el voltaje se divide entre las resistencias de acuerdo con sus valores. El divisor de tensión aprovecha esta propiedad para "dividir" el voltaje de entrada en una parte más pequeña. La tensión a la salida del divisor se puede calcular usando la siguiente expresión:

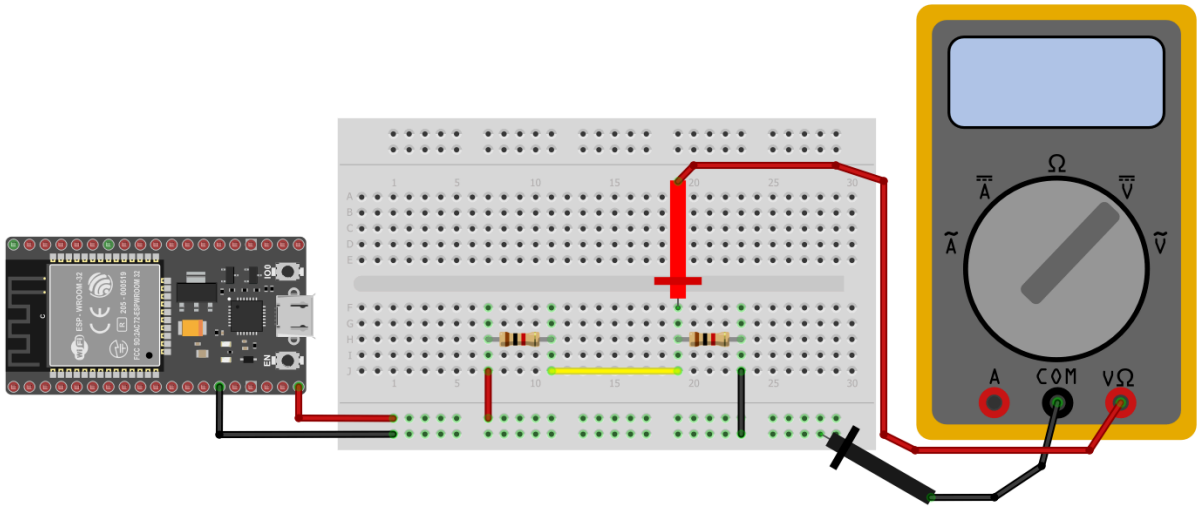
$$V_{out} = \frac{R_2}{R_1 + R_2} * V_{in}$$

Si tienes una fuente de 5V, y usas dos resistencias en serie de 1 kΩ (R1) y 2 kΩ (R2), el voltaje a la salida del partidor de tensión se puede calcular como:

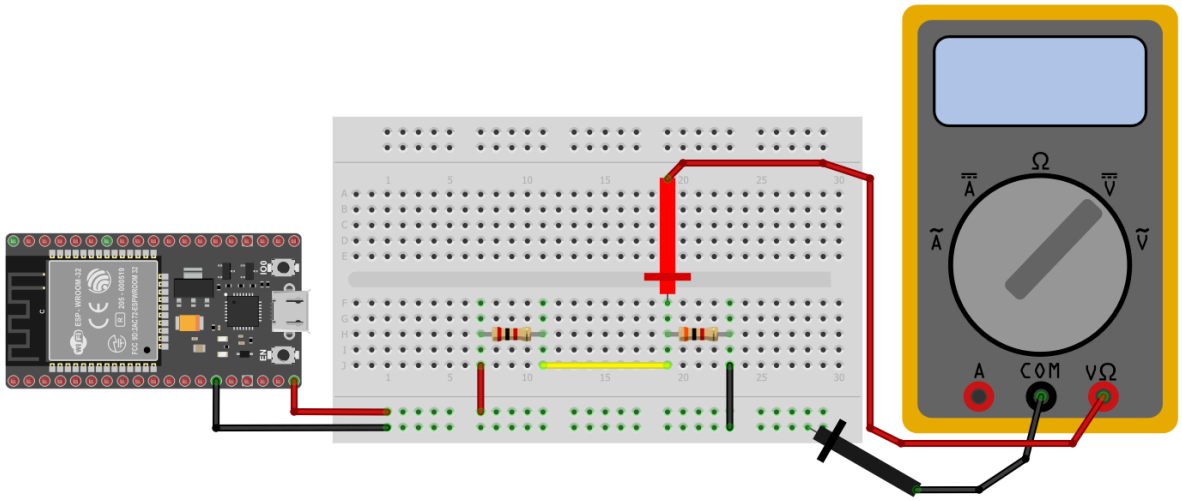
$$V_{out} = 5 * \frac{2k\Omega}{2k\Omega + 1k\Omega} = 3{,}33V$$

4. 3. 3. Actividades con divisores de tensión

Para practicar el concepto de partidor de tensión, monta dos resistencias de 1 kΩ en serie en la placa de pruebas y utiliza de nuevo el propio NodeMCU para alimentar el circuito (recuerda que VIN es la alimentación positiva y GND la negativa), tal y como se ve en el esquema siguiente. Mide la tensión en el punto medio entre las dos resistencias y verifica que coincide con el valor calculado usando la fórmula del divisor de tensión.



Ahora crea un circuito con un divisor de tensión que utilice una resistencia de 2 kΩ (R1) y otra de 3 kΩ (R2) conectadas en serie a la alimentación del NodeMCU. Calcula el voltaje de salida del partidor y comprueba que está correcto midiendo la tensión en el punto indicado. A continuación, tienes el esquema del partidor del ejercicio.



5. ENSAMBLAJE DEL DEMOSTRADOR 3x1

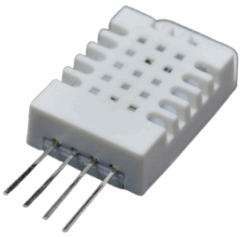
El montaje del Demostrador 3x1 se realizará de forma modular, abordando cada sección por separado en los apartados correspondientes de esta guía. Este enfoque busca simplificar el proceso y facilitar su comprensión, permitiendo que el montaje sea claro y accesible para quienes lo lleven a cabo. Cada paso está diseñado con el objetivo de garantizar un resultado funcional y efectivo, incluso para quienes no cuenten con experiencia previa en ensamblaje de demostradores.

Uno de los objetivos principales de este proyecto es fomentar la autonomía de los centros educativos, permitiéndoles crear sus propios demostradores si cuentan con impresoras 3D en sus instalaciones. Por esta razón, hemos puesto a disposición los archivos de impresión necesarios para la fabricación de los componentes, los cuales pueden descargarse a través del siguiente enlace: Archivos de impresión 3D. Con esta herramienta, buscamos empoderar a las instituciones educativas para que integren estos demostradores en sus actividades de forma independiente y eficiente.

5. 1. Montaje del Demostrador de Agua + Solar

5. 1. 1. Montaje electrónico (Parte de Agua)

Comenzaremos realizando el montaje del captador de agua. Para realizar este montaje completo necesitarás el siguiente material:



1 x Sensor DHT-22

Encargado de medir la temperatura y la humedad ambientales.

3 x Resistencias de 1 kΩ

Usadas para limitar la corriente que fluye por los diodos LED.

3 x LEDs de Colores (Rojo, Verde y Azul preferiblemente)

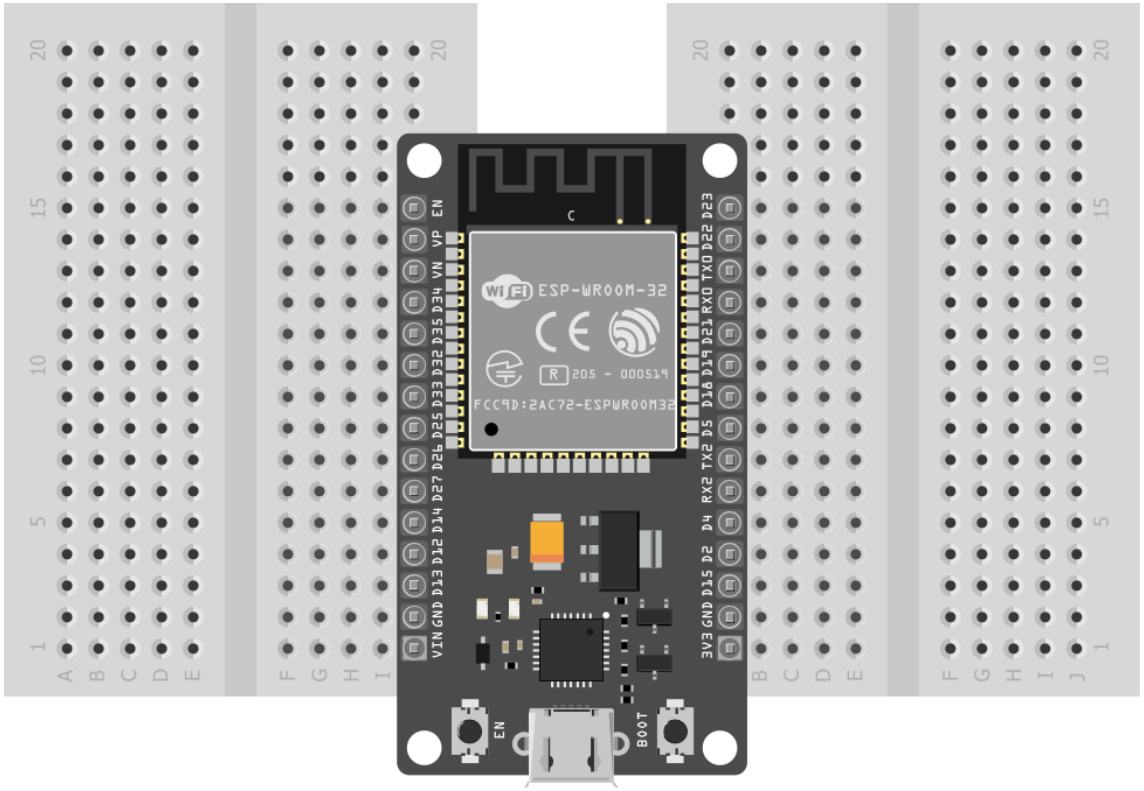
Empleados como indicadores visuales del rango de la temperatura medida.

Cables dupont (Macho-Macho)

Necesarios para realizar las conexiones entre los diferentes componentes en la protoboard.

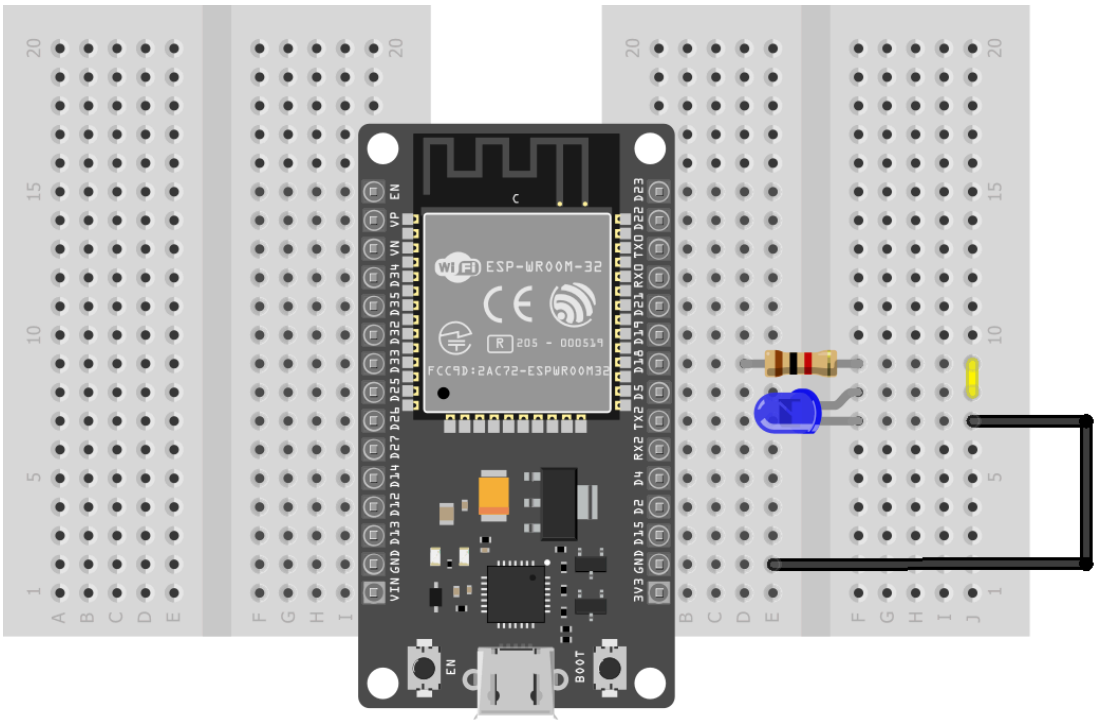
El objetivo del montaje será obtener el valor de la temperatura y la humedad ambientales y, en función del valor de la temperatura, encender un LED u otro como indicadores de peligro por alta o baja temperatura. De esta manera si la temperatura está por debajo de los 20ºC se enciende el LED Azul, si se está por encima de los 25ºC se enciende el LED Rojo y si está entre ambos valores (entre 20 y 25ºC) se enciende el LED Verde.

El primer paso es colocar la placa con ESP32 sobre dos protoboard pequeñas para tener espacio suficiente para luego conectar los componentes que vamos a necesitar. A continuación se puede ver el resultado:



Una vez tengamos el microcontrolador colocado en las dos protoboards, empezaremos a conectar los LEDs de colores con sus resistencias limitadoras. Vamos a empezar colocando el LED Azul y una de las resistencias de 1 kΩ en serie. Conectaremos el positivo del LED en el **pin D18**, y utilizaremos cualquier pin GND

para conectar el negativo del circuito. A continuación se puede ver el montaje indicado.



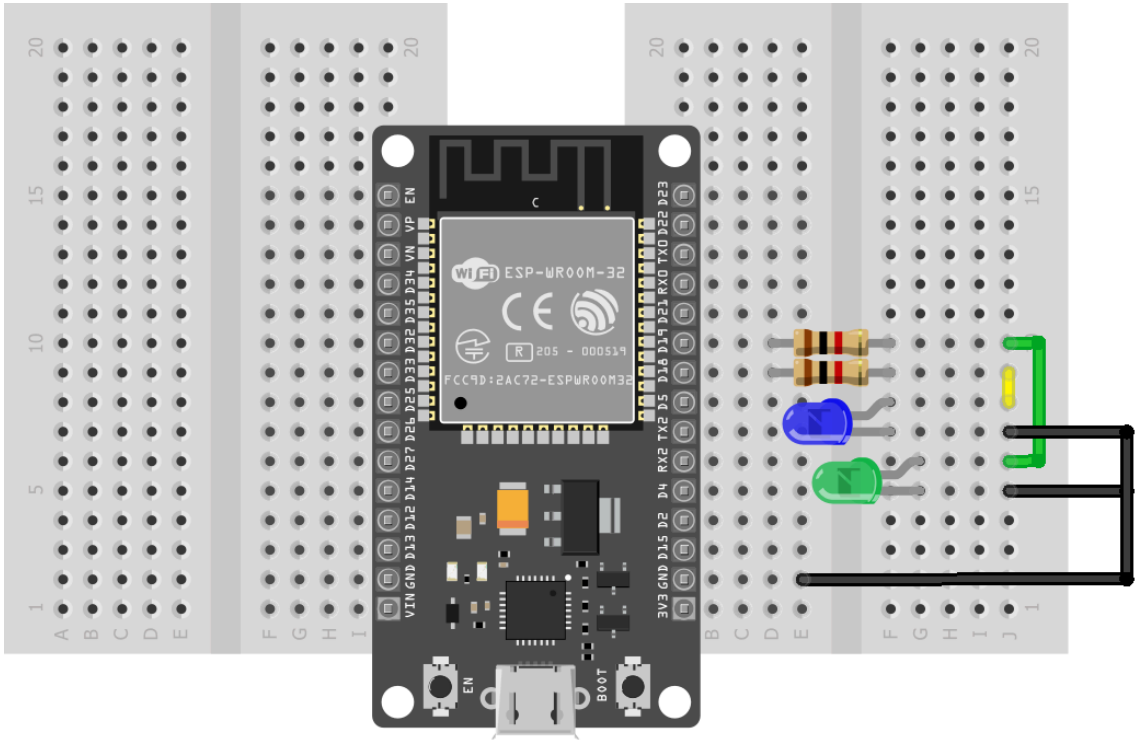
Para comprobar el correcto funcionamiento del montaje, podemos cargar el siguiente código en el NodeMCU:

```
#define ledPin1Cold 18 // Definimos el pin donde está conectado el LED Azul

// Bloque de inicialización
void setup() {

    // Configuramos el modo de funcionamiento del pin del LED como salida
    pinMode(ledPin1Cold, OUTPUT);

}

// Bucle principal del código
void loop() {

    digitalWrite(ledPin1Cold, LOW); // Apagamos el LED
    delay(1000); // Esperamos 1 segundo = 1000 milisegundos
    digitalWrite(ledPin1Cold, HIGH); // Encendemos el LED
```

```
    delay(1000); // Esperamos 1 segundo

}
```

Este pequeño programa hace parpadear el LED Azul, cambiando su estado entre encendido y apagado cada segundo. Si no funciona correctamente, comprueba que el LED se encuentre bien orientado (recuerda que el terminal más largo es el positivo, mientras el corto es el negativo) y conectado adecuadamente con la resistencia limitadora.

El siguiente paso del montaje será añadir un segundo LED. Esta vez colocaremos y conectaremos el LED de color Verde en el **pin D19** con su resistencia. Puedes ver el esquema de montaje indicado en la siguiente imagen.

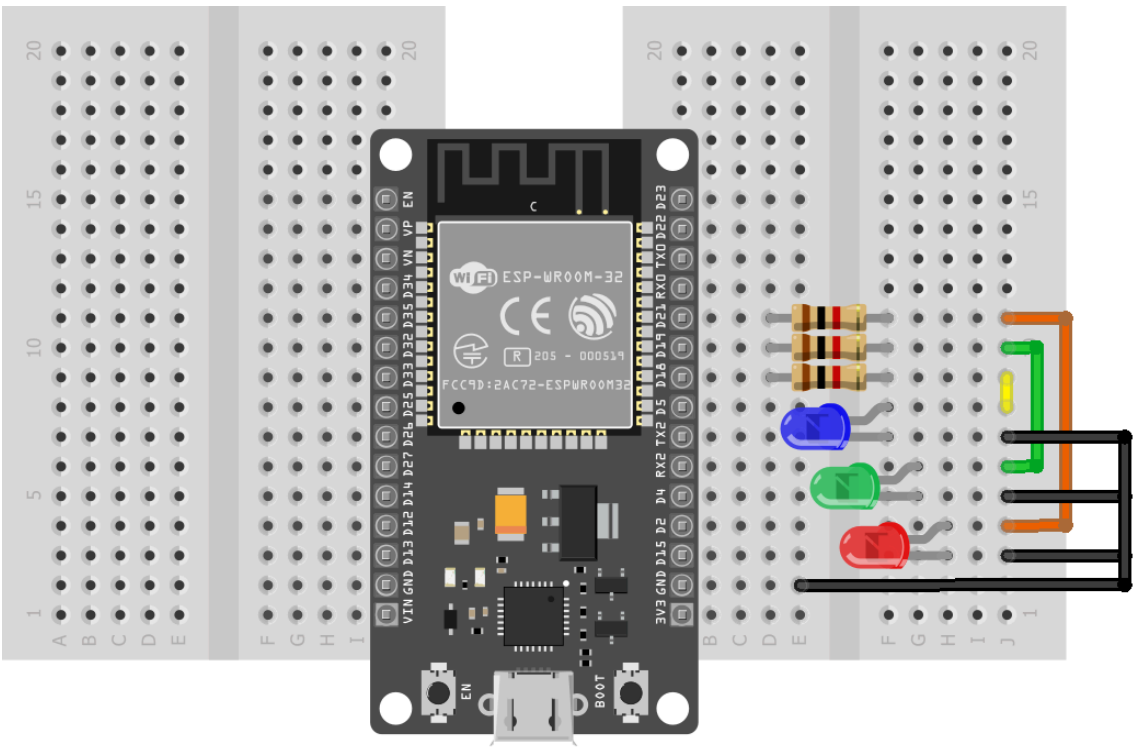


Vamos a comprobar de nuevo que el montaje es correcto. Vamos a utilizar ahora el código que encontrarás a continuación para verificar que el nuevo LED también está adecuadamente colocado.

```
#define ledPin1Cold 18 // Definimos el pin donde está conectado el LED Azul
#define ledPin2Good 19 // Definimos el pin donde está conectado el LED Verde

// Bloque de inicialización
void setup() {

    // Configuramos el modo de funcionamiento de los pines de los LED como salida
    pinMode(ledPin1Cold, OUTPUT);
    pinMode(ledPin2Good, OUTPUT);

}

// Bucle principal del código
void loop() {

    digitalWrite(ledPin1Cold, LOW); // Apagamos el LED Azul
    digitalWrite(ledPin2Good, LOW); // Apagamos el LED Verde
    delay(1000); // Esperamos 1 segundo = 1000 milisegundos
    digitalWrite(ledPin1Cold, HIGH); // Encendemos el LED Azul
    digitalWrite(ledPin2Good, HIGH); // Encendemos el LED Verde
    delay(1000); // Esperamos 1 segundo

}
```

Como puedes ver, el objetivo del código es el mismo: encender y apagar, esta vez, ambos LED cambiando de estado cada segundo. Si el LED Azul y el Verde cambian de estado correctamente, podemos dar por válido el montaje actual y continuar con el siguiente paso.

Para terminar de montar los LEDs de colores que necesitamos para cumplir con el objetivo establecido con esta parte del Demostrador, necesitamos conectar el último LED de color Rojo en el **pin D21**. Procede a montar el diodo LED y la última resistencia correspondiente, como se muestra en la siguiente imagen.

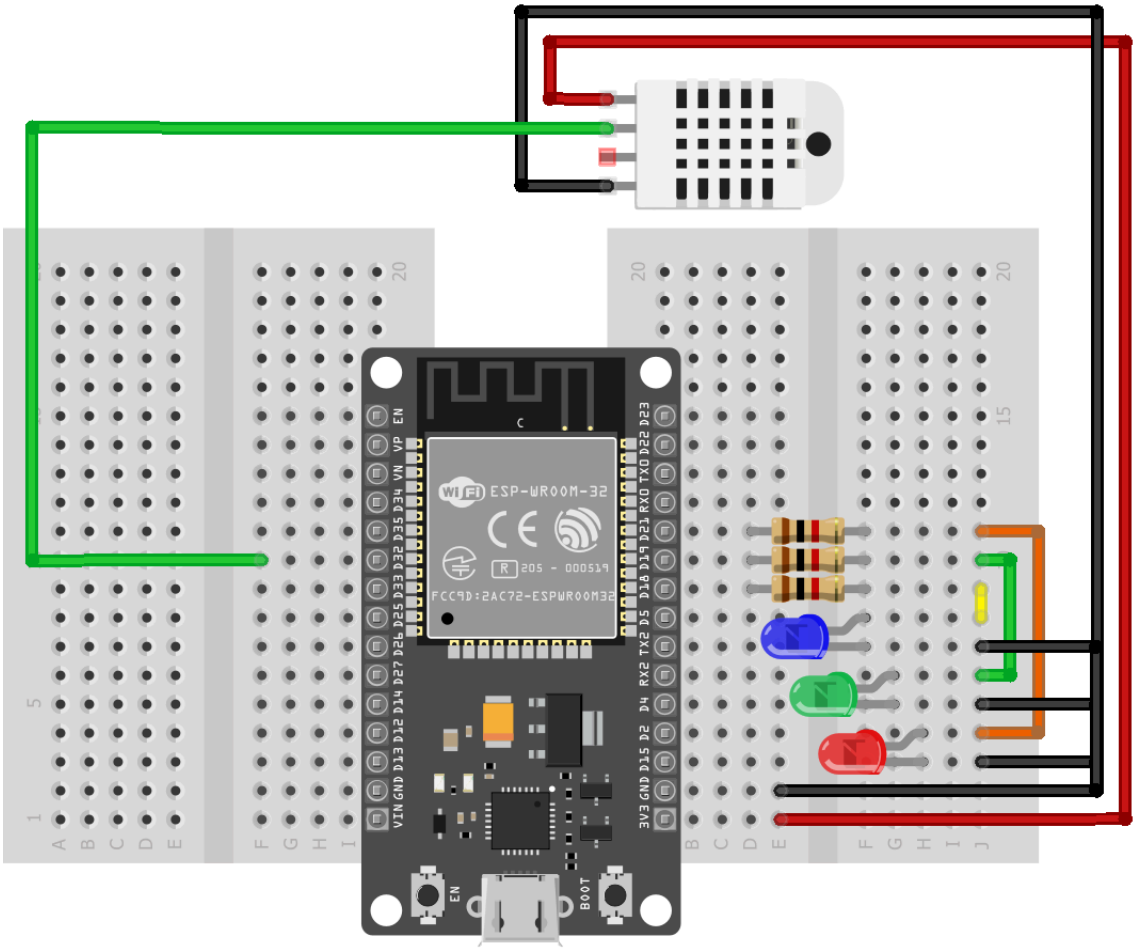


Comprobaremos que está correcto realizando la misma programación que en los casos anteriores, generando un parpadeo, ahora, en los tres LEDs de colores. El código para ello es el que sigue.

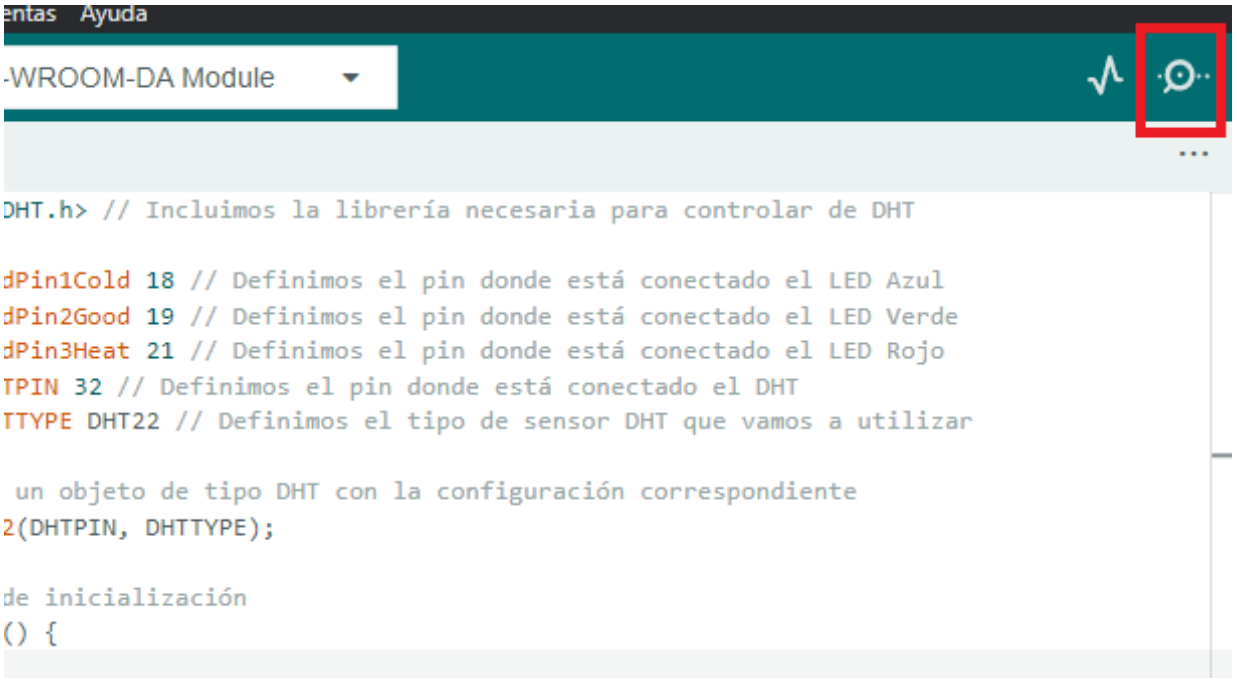
```
#define ledPin1Cold 18 // Definimos el pin donde está conectado el LED Azul
#define ledPin2Good 19 // Definimos el pin donde está conectado el LED Verde
#define ledPin3Heat 21 // Definimos el pin donde está conectado el LED Rojo

// Bloque de inicialización
void setup() {

    // Configuramos el modo de funcionamiento de los pines de los LED como salida
    pinMode(ledPin1Cold, OUTPUT);
    pinMode(ledPin2Good, OUTPUT);
    pinMode(ledPin3Heat, OUTPUT);

}

// Bucle principal del código
void loop() {
```

```
    digitalWrite(ledPin1Cold, LOW); // Apagamos el LED Azul
    digitalWrite(ledPin2Good, LOW); // Apagamos el LED Verde
    digitalWrite(ledPin3Heat, LOW); // Apagamos el LED Rojo
    delay(1000); // Esperamos 1 segundo = 1000 milisegundos
    digitalWrite(ledPin1Cold, HIGH); // Encendemos el LED Azul
    digitalWrite(ledPin2Good, HIGH); // Encendemos el LED Verde
    digitalWrite(ledPin3Heat, HIGH); // Encendemos el LED Rojo
    delay(1000); // Esperamos 1 segundo

}
```

El próximo paso es agregar y conectar el sensor DHT22 en el **pin D32**, ya sea como un componente individual o en un módulo. A continuación se presenta el montaje realizado hasta este momento junto con las conexiones del sensor DHT22.



Pasamos a comprobar el correcto funcionamiento del sensor DHT-22. A continuación, dispones de un código de prueba de todo el montaje realizado para esta parte del montaje. Carga el programa y verás cómo, además de que los LEDs de colores parpadean como han hecho hasta ahora, la placa con ESP32 enviará por el Monitor Serial (puedes ver el botón donde abrirlo en la siguiente imagen) cada par de segundos las medidas obtenidas de temperatura ambiental (en ºC) y humedad relativa del aire (en %).




```
#include <DHT.h> // Incluimos la librería necesaria para controlar de DHT

#define ledPin1Cold 18 // Definimos el pin donde está conectado el LED Azul
#define ledPin2Good 19 // Definimos el pin donde está conectado el LED Verde
#define ledPin3Heat 21 // Definimos el pin donde está conectado el LED Rojo
#define DHTPIN 32 // Definimos el pin donde está conectado el DHT
#define DHTTYPE DHT22 // Definimos el tipo de sensor DHT que vamos a utilizar

// Creamos un objeto de tipo DHT con la configuración correspondiente
DHT myDHT22(DHTPIN, DHTTYPE);

// Bloque de inicialización
void setup() {

    // Configuramos el modo de funcionamiento de los pines de los LED como salida
```

```
    pinMode(ledPin1Cold, OUTPUT);
    pinMode(ledPin2Good, OUTPUT);
    pinMode(ledPin3Heat, OUTPUT);

    // Inicializamos el monitor Serial
    Serial.begin(115200);

    // Inicializamos el sensor DHT
    myDHT22.begin();

}

// Bucle principal del código
void loop() {

    digitalWrite(ledPin1Cold, LOW); // Apagamos el LED Azul
    digitalWrite(ledPin2Good, LOW); // Apagamos el LED Verde
    digitalWrite(ledPin3Heat, LOW); // Apagamos el LED Rojo
    delay(1000); // Esperamos 1 segundo = 1000 milisegundos
    digitalWrite(ledPin1Cold, HIGH); // Encendemos el LED Azul
    digitalWrite(ledPin2Good, HIGH); // Encendemos el LED Verde
    digitalWrite(ledPin3Heat, HIGH); // Encendemos el LED Rojo
    delay(1000); // Esperamos 1 segundo

    //Obtenemos las medidas del sensor DHT
    float temperatura = myDHT22.readTemperature();
    float humedad = myDHT22.readHumidity();

    // Mostramos los valores por monitor Serial
    Serial.print("Temperatura: ");
    Serial.println(temperatura);
    Serial.print("Humedad: ");
    Serial.println(humedad);

}
```

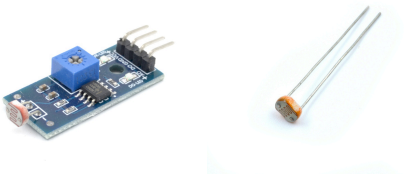
Con el monitor abierto, comprueba que el desplegable que se ubica más a la derecha sobre el propio Monitor tenga seleccionado el valor “115200”. Ahora podrás ver los datos que envía el NodeMCU tomados del DHT-22 de manera constante y verificar que el montaje hasta este punto funciona correctamente.

5. 1. 2. Montaje electrónico (Parte Solar)

Comenzaremos realizando el montaje del componente solar. Para realizar este montaje completo necesitarás el siguiente material:

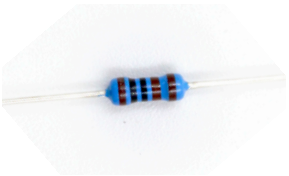
1 x Servomotor SG-90

Utilizado para mover el Demostrador solar y orientarlo hacia la luz solar.



2 x Fotoresistencias en Módulo o independientes

Empleados para medir el nivel de luminosidad recibida.



2 x Resistencias del mismo valor que las LDR

Utilizadas solamente junto con las LDR independientes para montar un partidor de tensión controlado por intensidad lumínica.

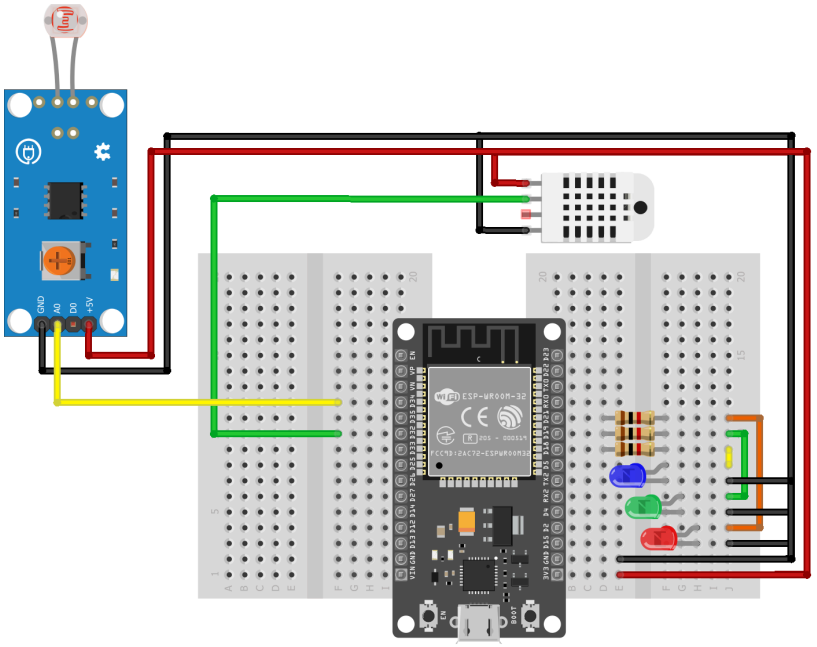
Cables dupont Macho-Macho

Necesarios para realizar las conexiones entre los diferentes componentes en la protoboard.

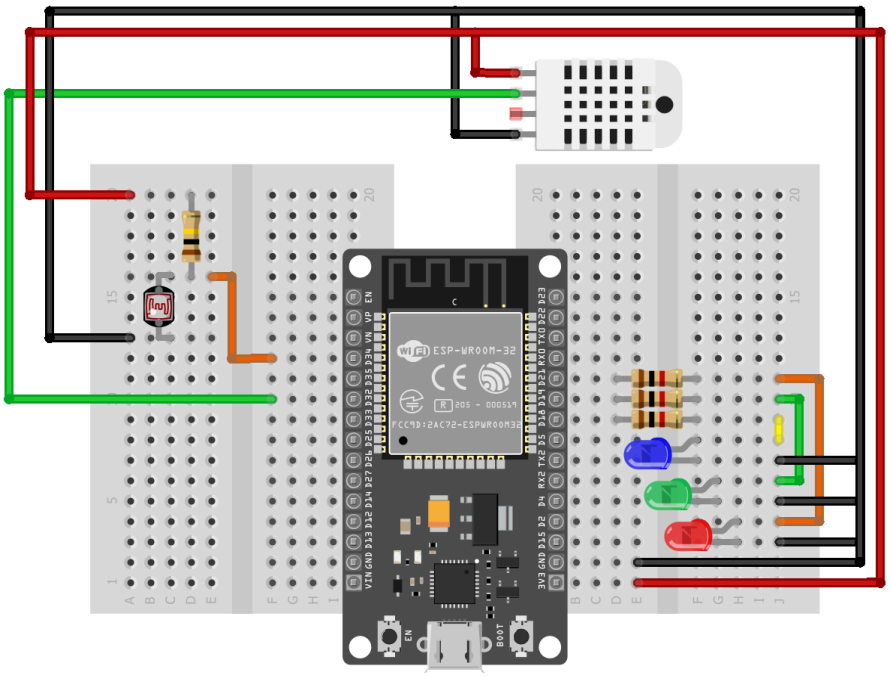
1 x Panel solar	1 x Detector de tensión
Empleado para detectar luz solar.	Usado junto al panel solar para detectar luz solar.

El objetivo del montaje será obtener la orientación de la incidencia de la luz solar con las LDR y rotar el Demostrador para que incida la mayor cantidad de rayos solares sobre el panel.

El montaje del componente solar va a comenzar desde el último montaje que se vio en el apartado anterior. En este montaje se añade el módulo de LDR o fotorresistencia. Para obtener el valor del sensor se va a utilizar la salida digital del mismo, que se va a conectar al **pin D34** del ESP32. El montaje se puede ver a continuación:



Este esquema también se puede sustituir por el montaje de una LDR y una resistencia en serie, formando un partidor de tensión controlado por la intensidad lumínica, como el que se muestra a continuación:

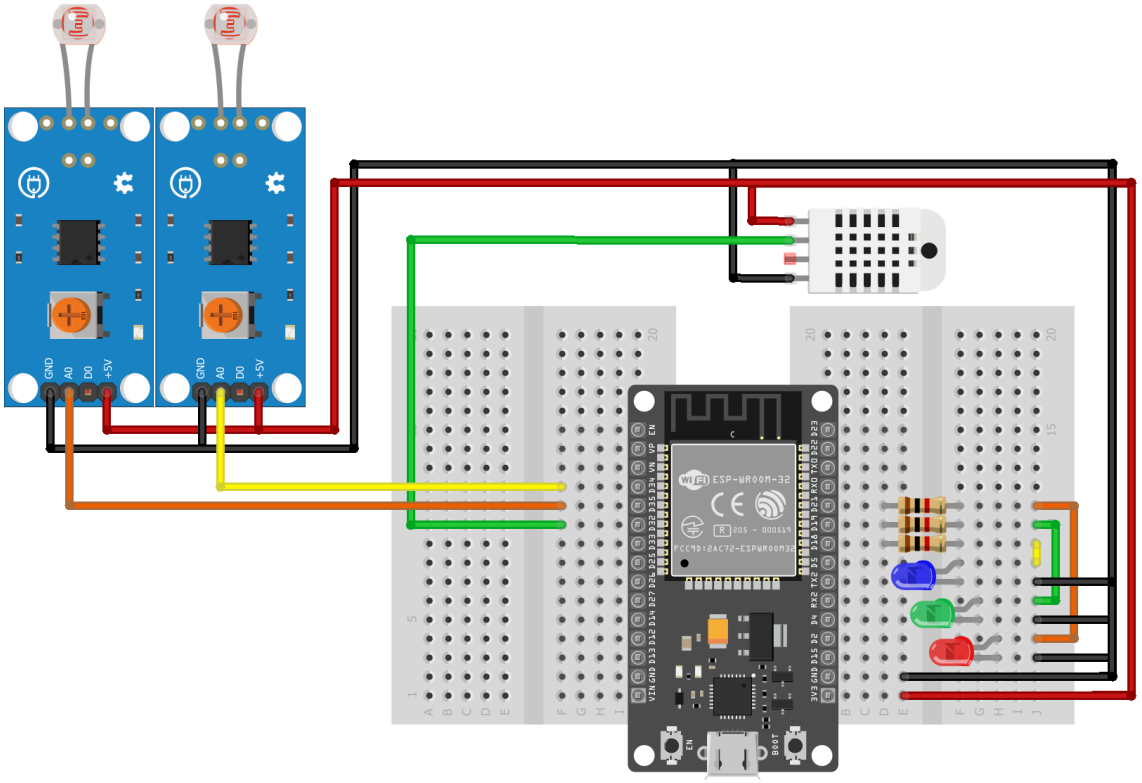


Para comprobar el funcionamiento de los nuevos elementos añadidos al montaje, vamos a emplear un código que solamente tome la medida de la LDR para encender el LED Rojo con mayor o menor intensidad según la luminosidad medida por la fotorresistencia. Puedes ver el código a cargar a continuación.

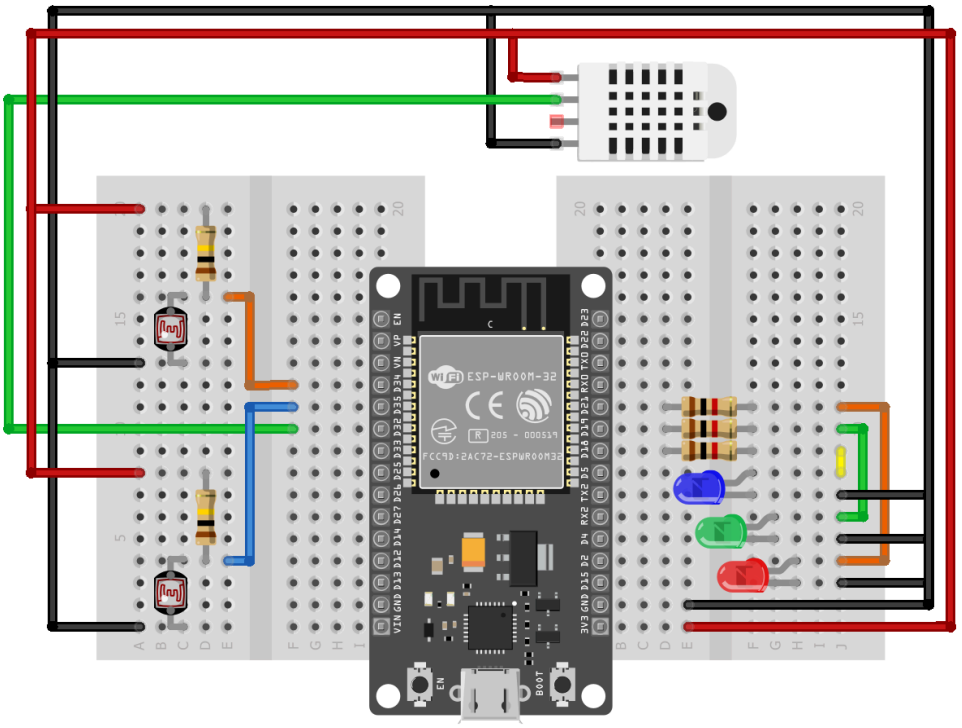
```
#define ldrPin1 34 // Definimos el pin donde está conectado la LDR

// Bloque de inicialización
void setup() {

    // Inicializamos el monitor serial
    Serial.begin(115200);

}

// Bucle principal del código
void loop() {
```

```
    // Tomamos la lectura del pin donde se encuentra la LDR
    int lectura_ldr_1 = analogRead(ldrPin1);
    // Mostramos el valor por el monitor Serial
    Serial.print("Luminosidad 1: ");
    Serial.println(lectura_ldr_1);

}
```

A continuación se debe añadir el segundo módulo de LDR. Para obtener el valor de la lectura de este sensor se va a utilizar la salida digital, como en el caso anterior, pero esta se va a conectar al **pin D35**. A continuación se puede ver el montaje con ambas LDR para la parte actual del Demostrador.



A continuación se muestra el montaje anterior pero, en lugar de emplear las LDR en módulo, haciendo uso de dos LDR colocadas como partidores de tensión.

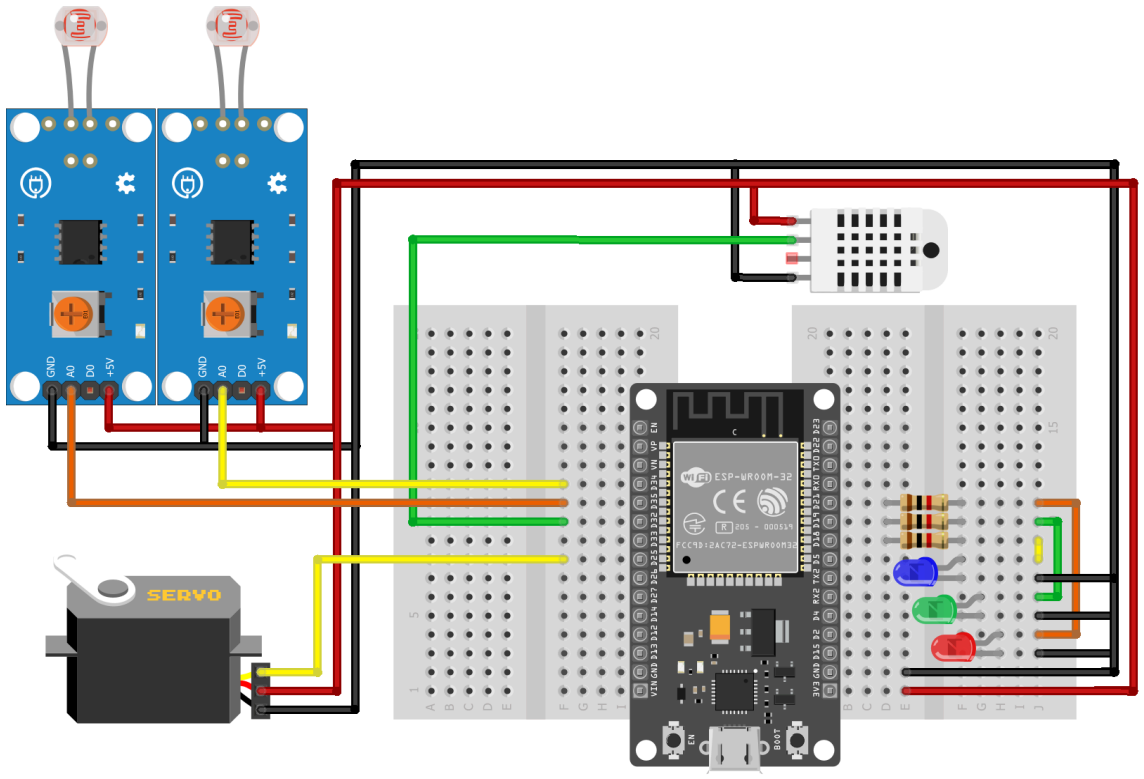


Comprobemos que se ha conectado correctamente empleando el siguiente código, que es muy similar al anterior pero mostrando el valor de la medida de ambas fotorresistencias.

```
#define ldrPin1 34 // Definimos el pin donde está conectado la primera LDR
#define ldrPin2 35 // Definimos el pin donde está conectado la segunda LDR

// Bloque de inicialización
void setup() {

    // Inicializamos el monitor serial
    Serial.begin(115200);

}

// Bucle principal del código
void loop() {

    // Tomamos las lecturas de los pines donde se encuentran las LDR
    int lectura_ldr_1 = analogRead(ldrPin1);
    int lectura_ldr_2 = analogRead(ldrPin2);
    // Mostramos el primer valor por el monitor Serial
```

```
        Serial.print("Luminosidad 1: ");
        Serial.println(lectura_ldr_1);
        // Mostramos el segundo valor por el monitor Serial
        Serial.print("Luminosidad 2: ");
        Serial.println(lectura_ldr_2);

}
```

El siguiente componente que se va a añadir en el montaje es un servomotor. Este se va a conectar a ambas lineas de alimentación (cable rojo a la positiva y cable negro a la negativa) y el cable amarillo (que es la señal de control) se va a conectar al **pin D25**.

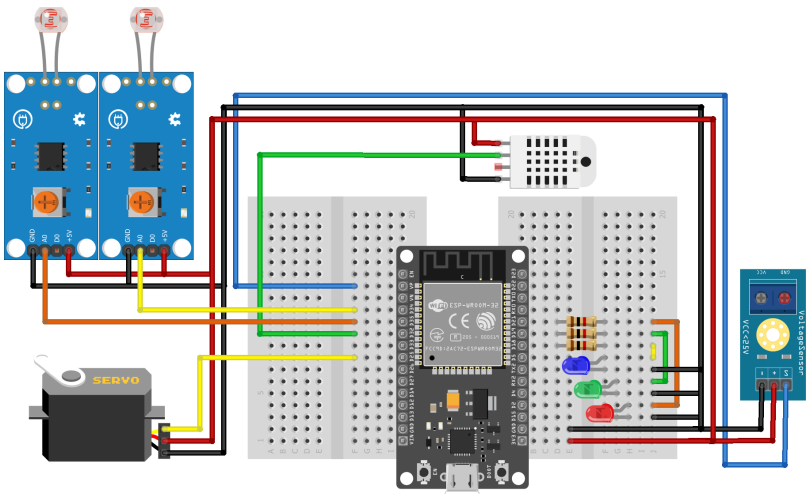


Comprobaremos el funcionamiento del servomotor cargando el siguiente código en la placa NodeMCU que realiza un barrido moviendo el servo entre las posiciones de 0º, 90º y 180º, en bucle, una y otra vez.

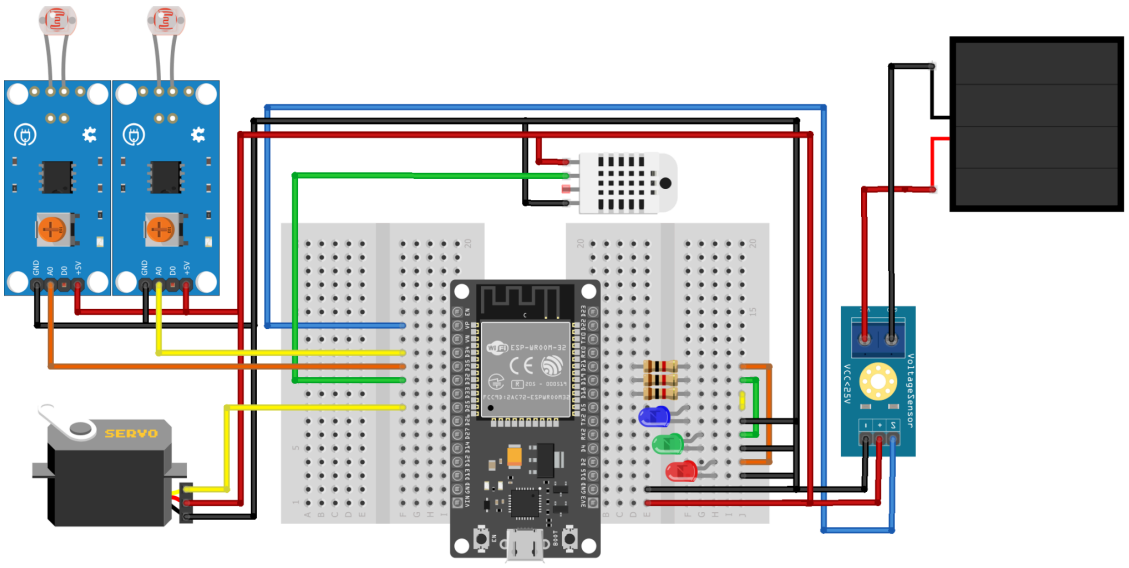
```
#include <Servo.h> // Incluir la librería necesaria para controlar el servomotor

#define servoPin 25 // Definimos el pin donde está conectado el servomotor
```

```
// Creamos un objeto de tipo Servo para controlarlo
Servo myServo;

// Bloque de inicialización
void setup() {

    // Asociamos el servo que creamos al pin donde está conectado
    myServo.attach(servoPin);
    // Colocamos el servo en una posición inicial de 0°
    myServo.write(0);

}

// Bucle principal del código
void loop() {

    myServo.write(0); // Movemos el servomotor a 0°
    delay(1000); // Esperamos 1 segundo = 1000 milisegundos
    myServo.write(90); // Movemos el servomotor a 90°
    delay(1000); // Esperamos 1 segundo
    myServo.write(180); // Movemos el servomotor a 180°
    delay(1000); // Esperamos 1 segundo

}
```

El último componente que vamos a conectar al ESP32 será un detector de tensión. Este módulo se debe alimentar a ambas líneas de alimentación (Vcc y GND). La línea de señal se va a conectar al **pin D36**.



El último paso que falta para terminar el montaje electrónico es conectar el panel solar al detector de tensión. A continuación se puede ver cómo quedaría el montaje completo.



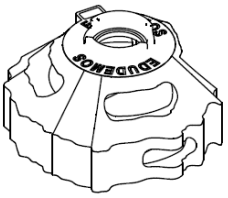
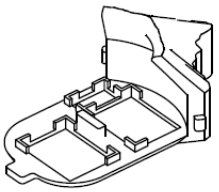
Vamos a comprobar el funcionamiento del detector solar completo haciendo uso del siguiente código que simplemente lee la señal del módulo detector de tensión y enciende el LED Azul si detecta voltaje.

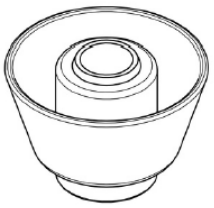
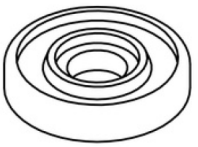
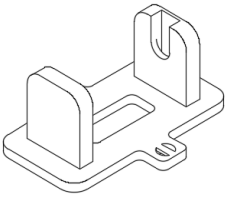
```
#define ledPin1Cold 18 // Definimos el pin donde está conectado el LED Azul
#define detectorPin 36 // Definimos el pin donde está conectado el Detector de
Tensión

// Bloque de inicialización
void setup() {

    // Configuramos el modo de funcionamiento del pin del LED como salida
    pinMode(ledPin1Cold, OUTPUT);

}

// Bucle principal del código
void loop() {

    // Tomamos la lectura del Módulo Detector
    bool tension_detectada = digitalRead(detectorPin);
```

```
    // Comprobamos si hay tensión o no
    if(tension_detectada) { // Sí hay tensión

        digitalWrite(ledPin1Cold, HIGH); // Encendemos el LED Azul

    } else { // No hay tensión

        digitalWrite(ledPin1Cold, LOW); // Apagamos el LED Azul

    }

}
```

5. 1. 3. Montaje Mecánico del Demostrador de Agua + Solar

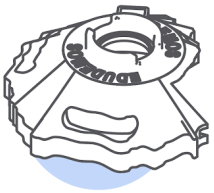
Los elementos que componen este demostrador son los siguientes:

Elemento	Cantidad	Descripción	Imagen
Base	1	Es la parte superior que asegura los componentes móviles, como las palas y el eje del motor. Proporciona estabilidad y permite que el sistema funcione suavemente, facilitando la correcta transferencia de energía desde las palas al generador.	
Cajón	1	Es la base que conecta las diferentes secciones del dispositivo, proporcionando soporte estructural y asegurando una correcta alineación entre el motor, las palas y otros componentes esenciales para su funcionamiento.	

Contenedor de agua	1	Esta parte es capaz de almacenar el agua que cae en él.	
Hoja	8	Hoja que permite que los líquidos entren en el contenedor de agua	
Disco Pequeño	3	Es un componente estático que no se mueve. Su función principal es actuar como parte del soporte estructural del dispositivo, ayudando a mantener la correcta alineación y estabilidad de los elementos.	
Disco Estático	2	Disco que se mantiene fija para tapar y mantener la estabilidad de la estructura.	
Placa Sol	1	Parte que permite el movimiento del panel solar para su correcta orientación	
Disco Solar	1	Disco que permite conectar el girador solar con un servomotor instalado y juntarlo con el resto de la estructura.	

A continuación se muestra como se realiza el montaje del demostrador:

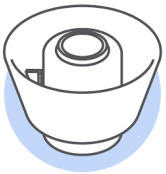
BASE



1 x1

2 x1

WATER



3 x1

4 x8

5 x2

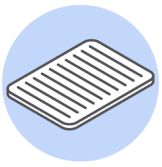
6 x2

SUN

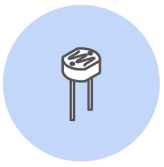
7 x1

8 x1

ELECTRONIC



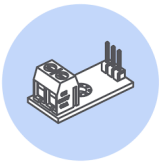
a x1



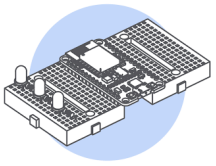
b x2

c x1

d x1

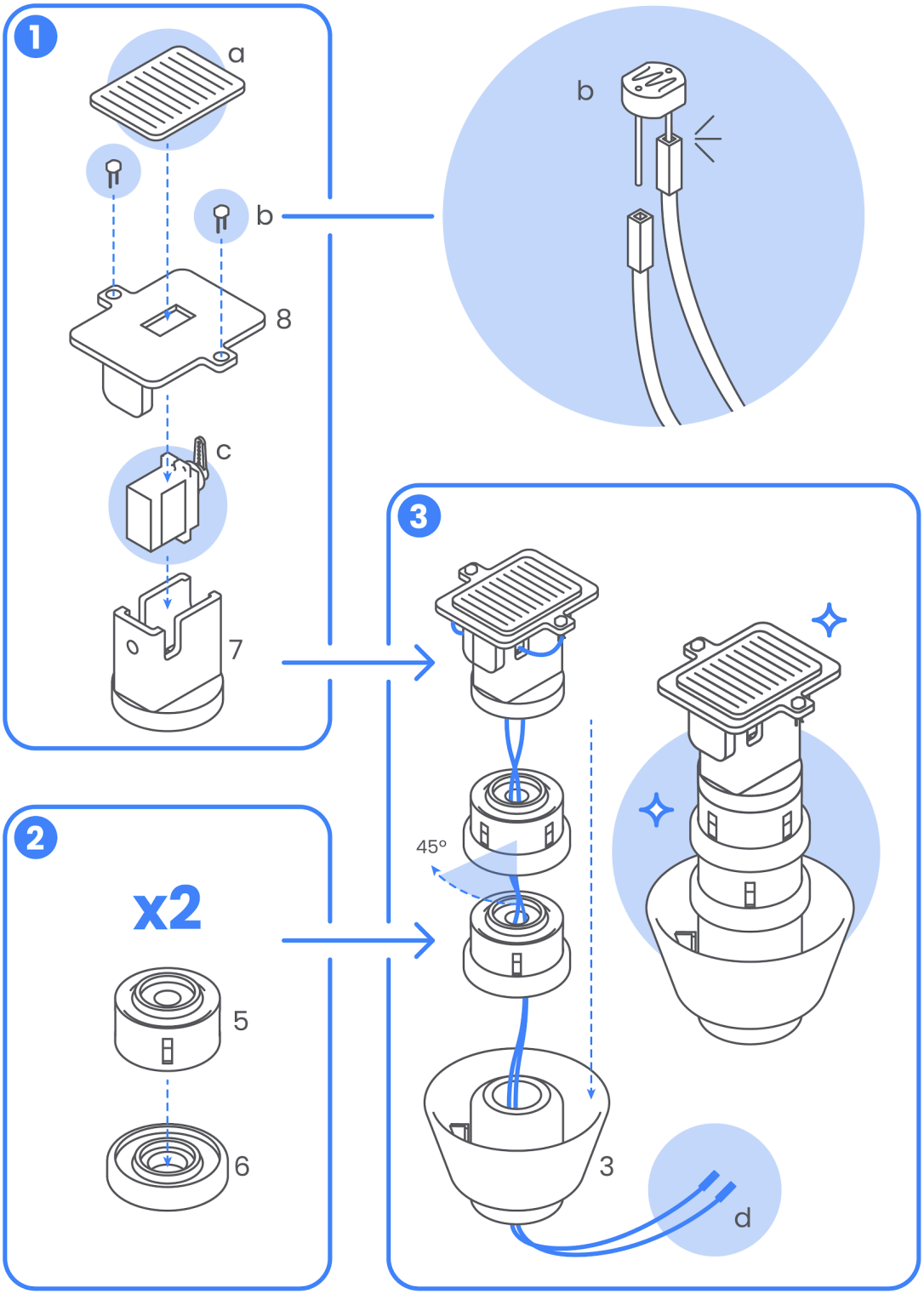


e x1

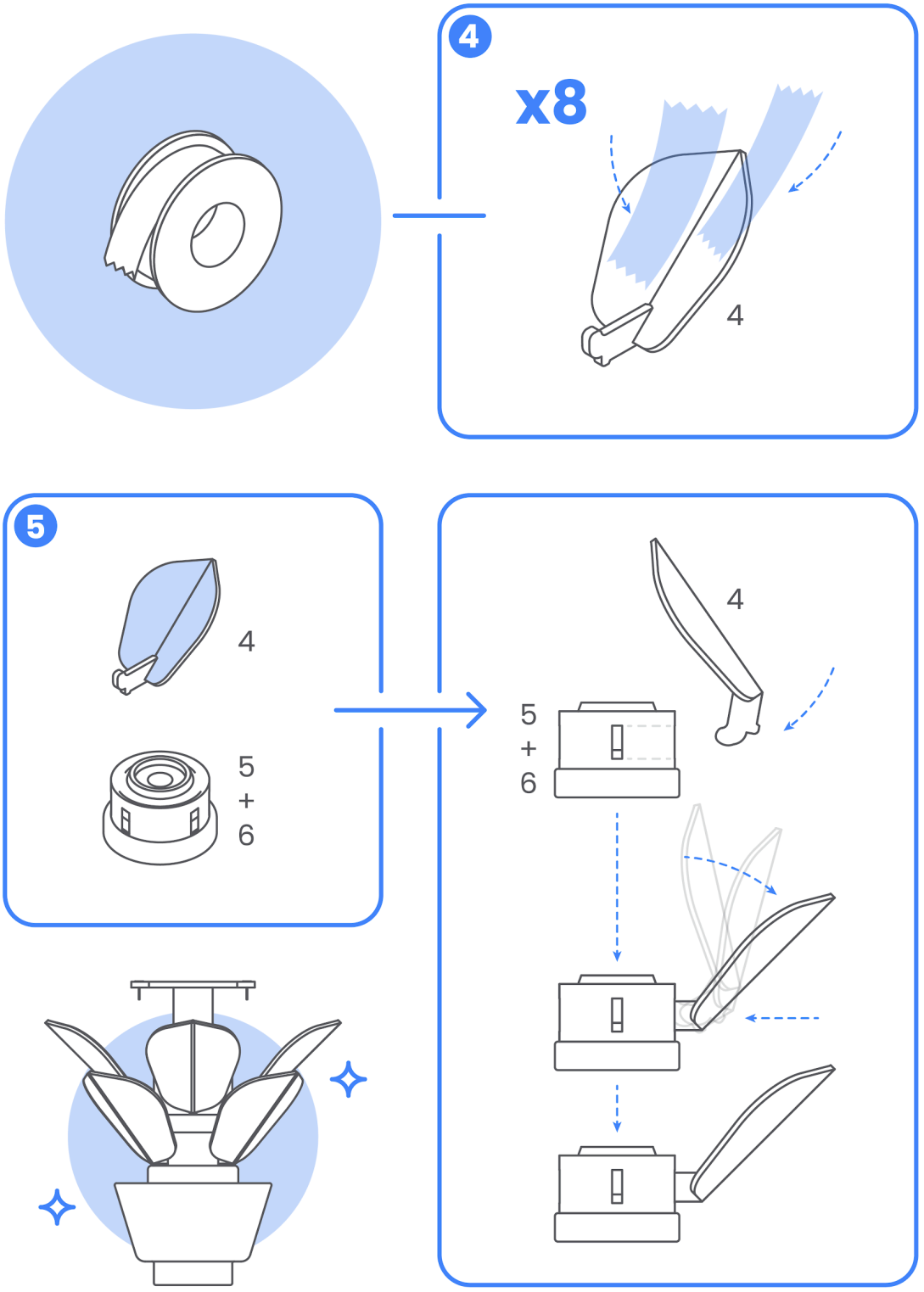


f x1

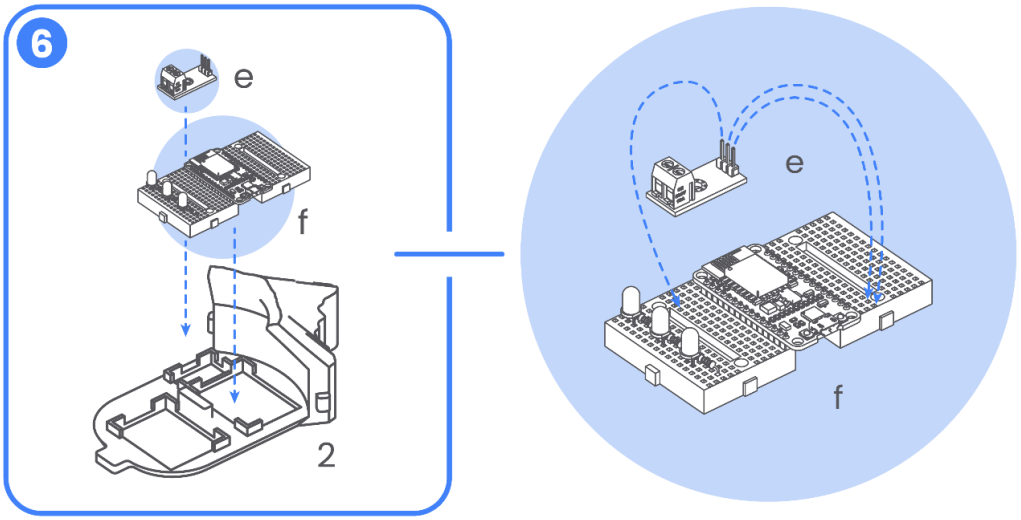
g x1

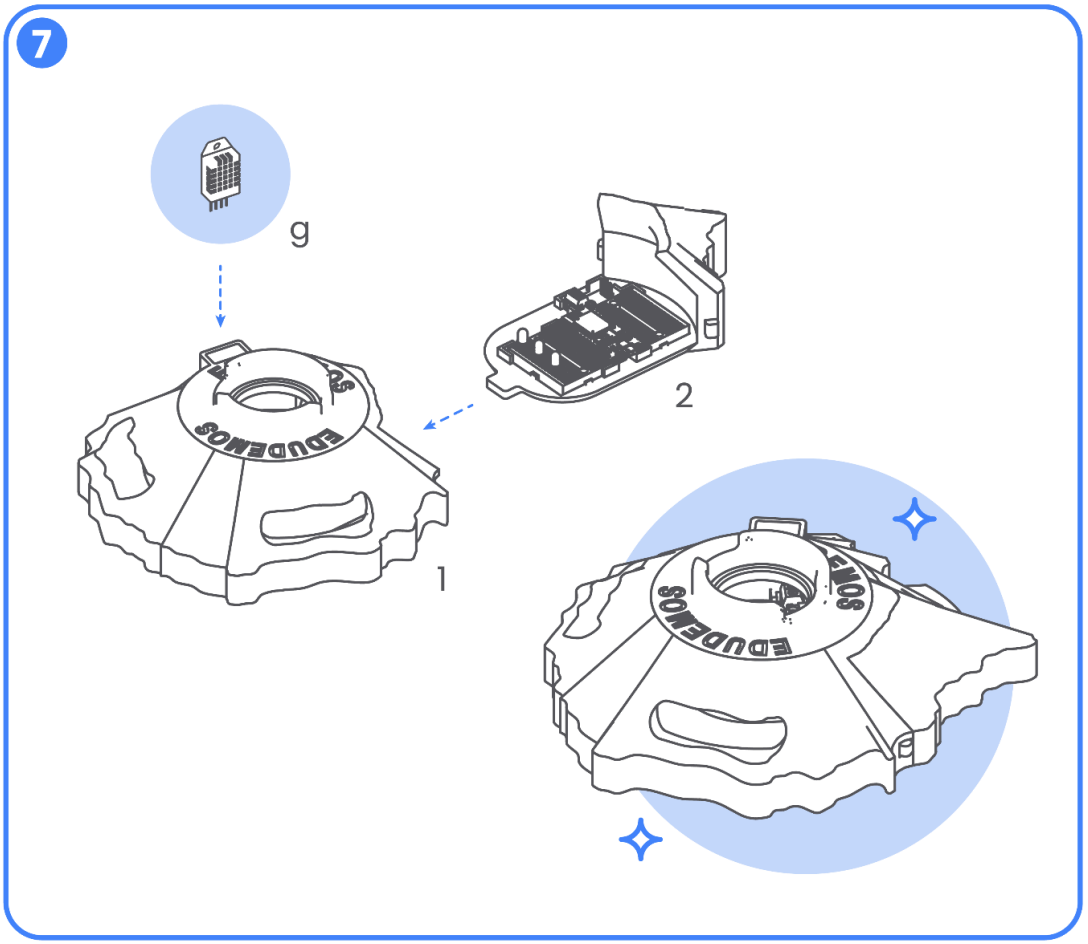


Funded by the European Union
Gerda Stetter Stiftung
Technik macht Spaß!
I.E.S. El Rincón
FUNDACIÓN SERGIO ALONSO
Finnova
gbs sg.ch

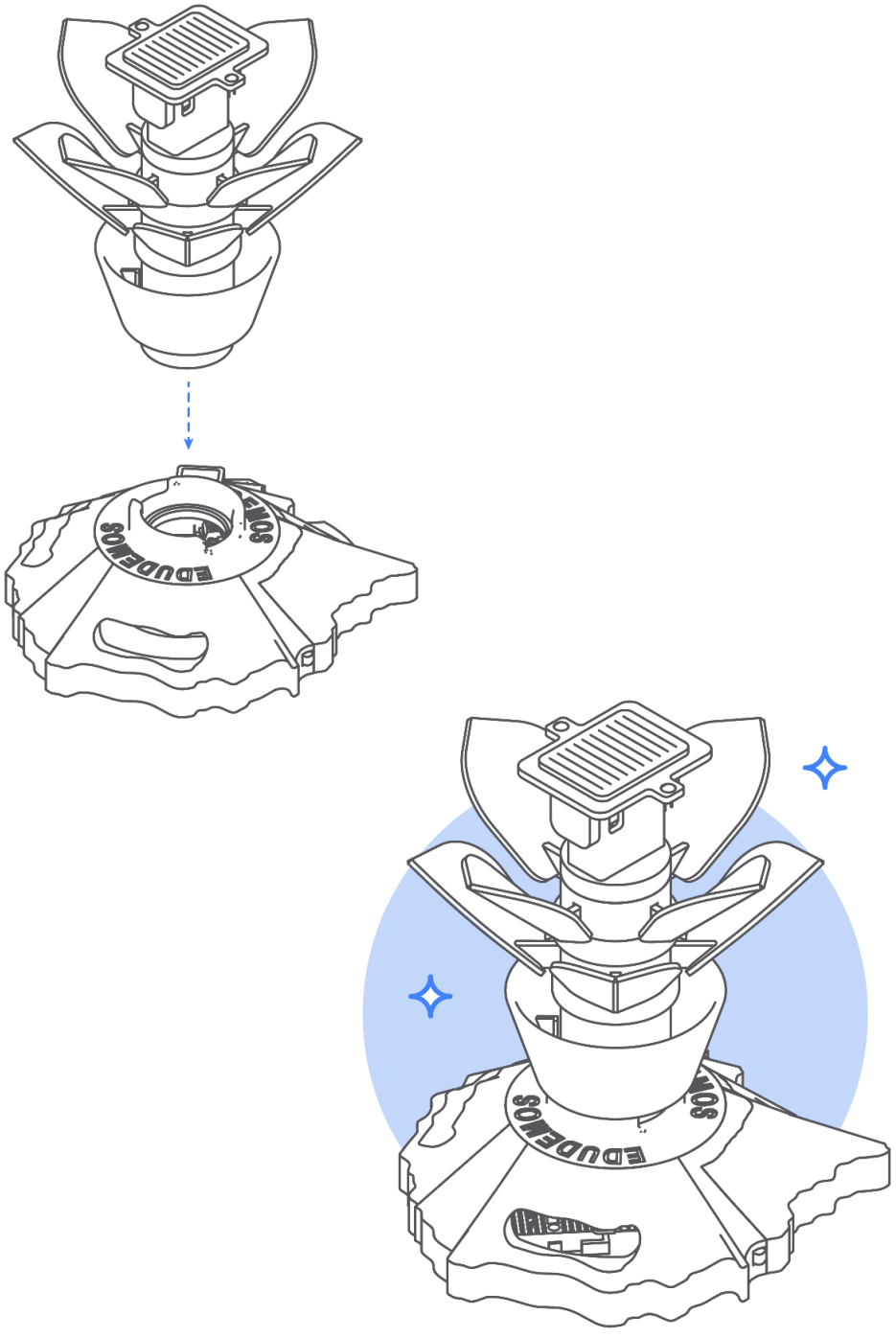


Funded by the European Union
Gerda Stetter Stiftung
Technik macht Spaß!
I.E.S. El Rincón
FUNDACIÓN SERGIO ALONSO
Finnova
gbs sg.ch


6
e
f
2
e
f


7
g
2
1

8

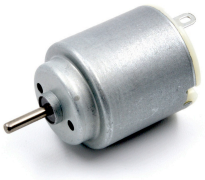


Funded by the European Union
Gerda Stetter Stiftung Technik macht Spaß!
FUNDACIÓN SERGIO ALONSO
Finnova
gbs sg.ch

5. 2. Montaje del Demostrador de viento

5. 2. 1. Montaje electrónico

Comenzaremos realizando el montaje del Demostrador de viento. Para realizar este montaje completo necesitarás el siguiente material:

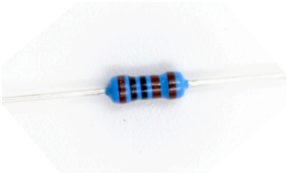


1 x Motor de Corriente Continua de 5V

Empleado para generar voltaje con la fuerza del viento.

1 x Condensador de 100nF / 1uF

Utilizados para eliminar ruidos indeseados generados por el Motor de CC.

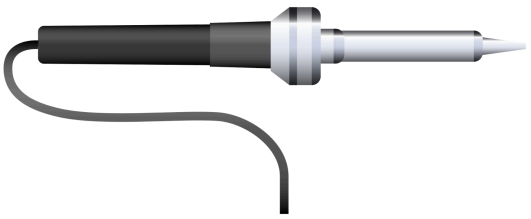


2 x Resistencias de 1 kΩ

Utilizados en el partidor de tensión que permite tomar la lectura de tensión generada por el Motor de CC.

Cables dupont Macho-Macho

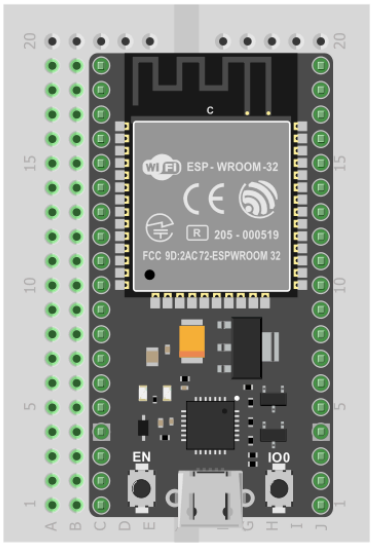
Necesarios para realizar las conexiones entre los diferentes componentes en la protoboard.

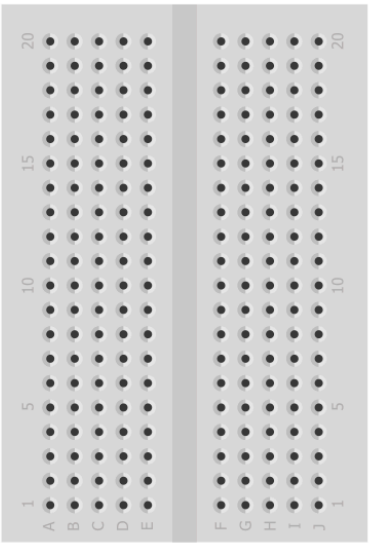


Soldador de estaño (sólo si el motor no tiene cables incluidos)

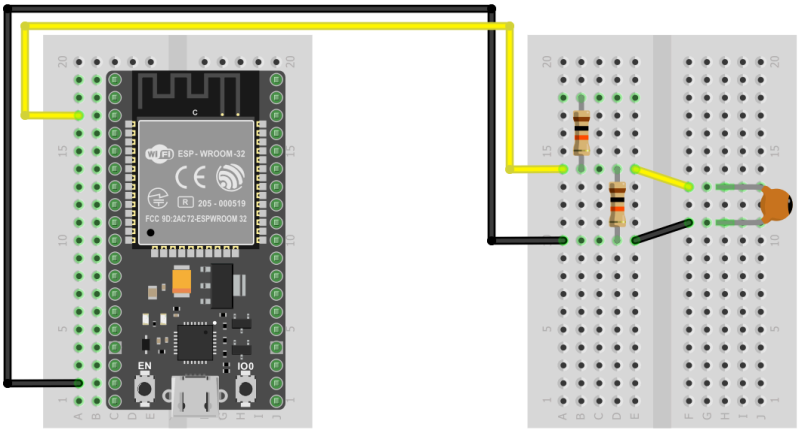
El objetivo del montaje será obtener la medida de la cantidad de voltaje generado por el motor, que irá unido a la estructura mecánica que girará por efecto de la fuerza del viento. Para poder realizar la medición con el NodeMCU de la tensión obtenida por el movimiento de las aspas del Demostrador, debemos montar un partidor de tensión que reduzca el nivel máximo de tensión a obtener (5V) a uno que sea apto para el ESP32 (que trabaja a 3'3V). Con esto podremos obtener cuánto voltaje se está generando por efecto del viento.

Para este nuevo montaje, será necesario crear un circuito desde el principio. Comenzaremos colocando el ESP32 en dos mini protoboards. A continuación, se muestra el montaje correspondiente.

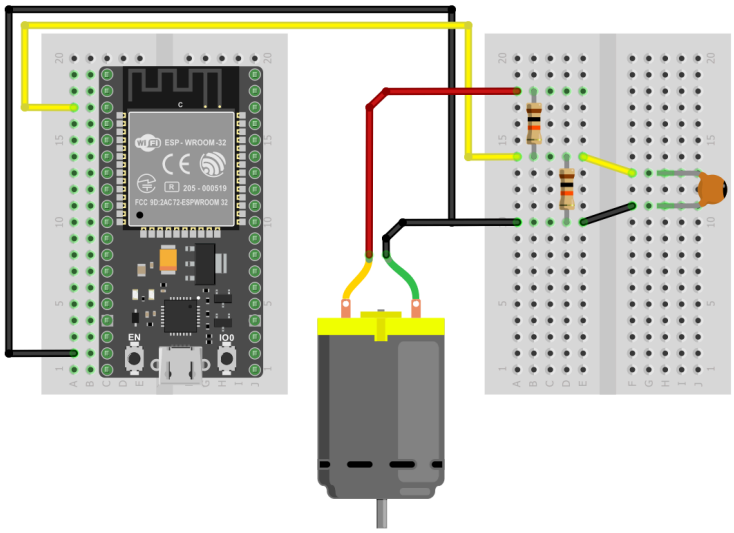




A continuación, se procederá a montar un divisor de tensión. El punto intermedio de este divisor se conectará al pin **D39 (VN)**. Además, es necesario conectar un condensador entre el terminal de este pin y GND, que servirá de filtro para las variaciones indeseadas en la medida, tal y como se muestra en la imagen siguiente.



El último paso en el montaje electrónico de este demostrador es conectar el motor al conjunto anterior, de manera que uno de sus terminales se une a GND y el otro terminal a la entrada del partidor de tensión montado antes. A continuación, se presenta este montaje terminado.

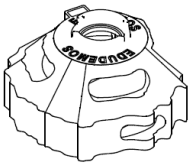


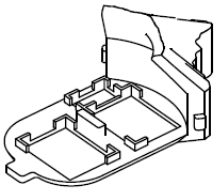
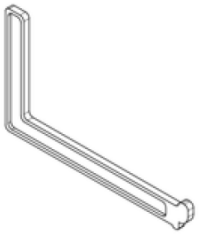
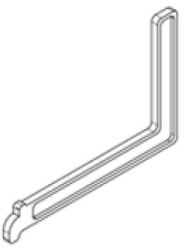
Vamos a comprobar que el montaje de este Demostrador funciona correctamente haciendo uso del siguiente fragmento de código en el que solamente leemos la salida del partidor de tensión que está unida al ESP32. Girando el rotor del motor, veremos como la medición va dando diferentes valores a través del Monitor Serial.

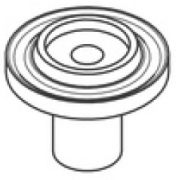
```
// Bloque de inicialización
void setup() {

    // Inicializamos el Monitor Serial
    Serial.begin(115200);

}

// Bucle principal del código
void loop() {

    // Tomamos la medida del pin que va a la salida del partidor de tensión
    int medicion = analogRead(39);
    // Mostramos la medida por el monitor serial
    Serial.println(medicion);

}
```

5. 2. 2. Montaje Mecánico

Los elementos que componen este demostrador son los siguientes:

Elemento	Cantidad	Descripción	Imagen
Base	1	Es la parte superior que asegura los componentes móviles, como las palas y el eje del motor. Proporciona estabilidad y permite que el sistema funcione suavemente, facilitando la correcta transferencia de energía desde las palas al generador.	

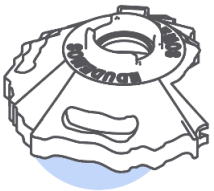
Cajón	1	Es la base que conecta las diferentes secciones del dispositivo, proporcionando soporte estructural y asegurando una correcta alineación entre el motor, las palas y otros componentes esenciales para su funcionamiento.	
Soporte Motor	1	Esta pieza se encargará de sostener y brindar apoyo al motor, cuyo eje estará conectado al disco de rodamiento. Este diseño permitirá la conexión de los pines de salida, que emitirán señales indicando la presencia de movimiento.	
Hoja Grande en L	6	Es una hoja fija que no se mueve. Está diseñada para formar parte de la estructura estática del demostrador, contribuyendo a la estabilidad y el equilibrio del sistema en su conjunto.	
Hoja Pequeño en L	6	Es una hoja fija que no se mueve, de menor tamaño que la pieza anterior. Está diseñada para formar parte de la estructura estática del demostrador, contribuyendo a la estabilidad y el equilibrio del sistema en su conjunto.	

Pala Cuadrada	8	Es una de las palas móviles clave que gira cuando es impulsada por el viento. Su diseño cuadrado permite capturar eficientemente la energía cinética del viento y convertirla en movimiento rotatorio.	
Disco grande	2	Es un componente clave que actúa como soporte estructural para las palas móviles. Este disco está diseñado para girar cuando las palas capturan el viento, transfiriendo el movimiento rotatorio hacia el eje central del sistema.	
Disco Pequeño	3	Es un componente estático que no se mueve. Su función principal es actuar como parte del soporte estructural del dispositivo, ayudando a mantener la correcta alineación y estabilidad de los elementos móviles, como las hojas y los discos grandes.	
Disco rodamiento	1	Es un componente crucial que se utiliza para reducir la fricción entre las partes móviles del sistema. Este disco actúa como un soporte rotatorio que permite que los componentes, como el Disco Grande y las palas, giren de manera suave y eficiente alrededor del eje central.	

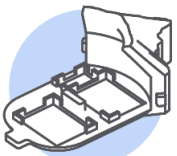
Tapa	1	Es un componente estructural que se coloca en la parte superior del conjunto, asegurando que todos los elementos internos estén protegidos y correctamente alineados.	
Rodamiento	1	Es un componente mecánico que permite que los elementos giratorios, como los discos y las palas, se muevan con mínima fricción.	
Espigas Madera	3	Es un componente estructural utilizado para conectar y asegurar las diferentes piezas de madera del dispositivo.	

A continuación se muestra como se realiza el montaje del demostrador:

BASE



1 x1



2 x1

3 x1

VIENTO

4 x3

5 x2

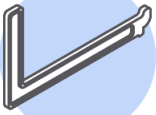
6 x1

7 x1

8 x3

9 x1

10 x6

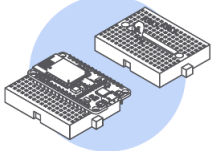


11 x6

12 x8

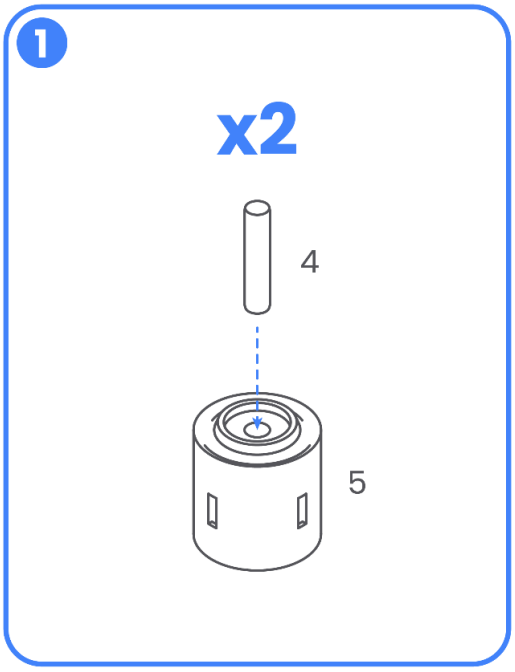
ELECTRÓNICA

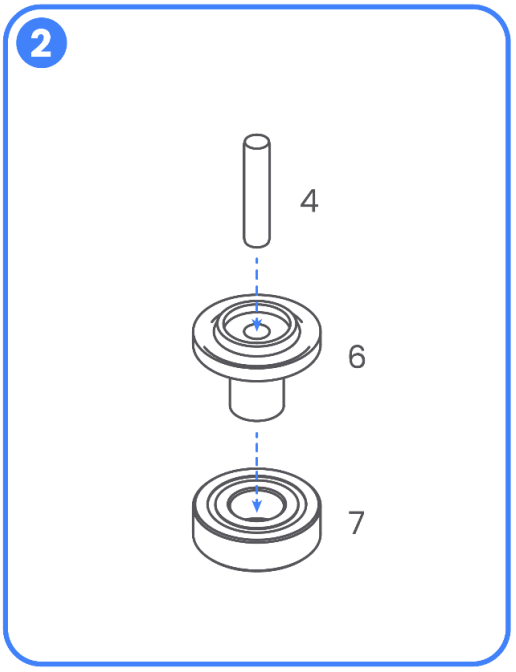
a x1

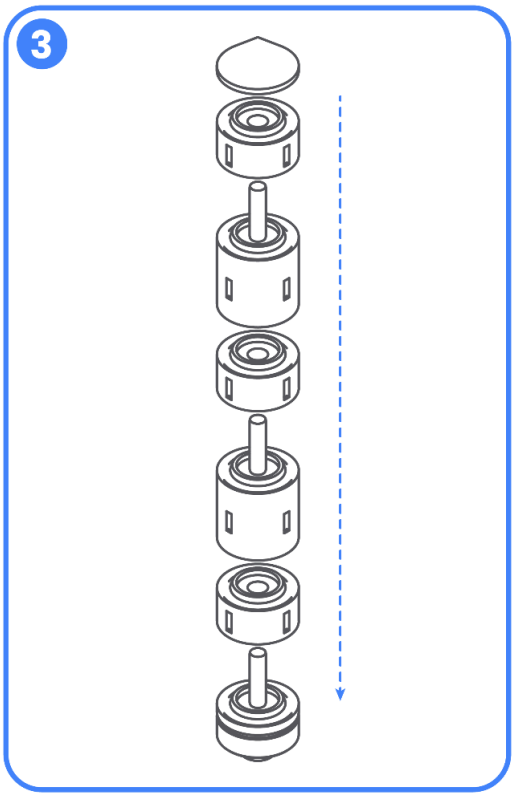


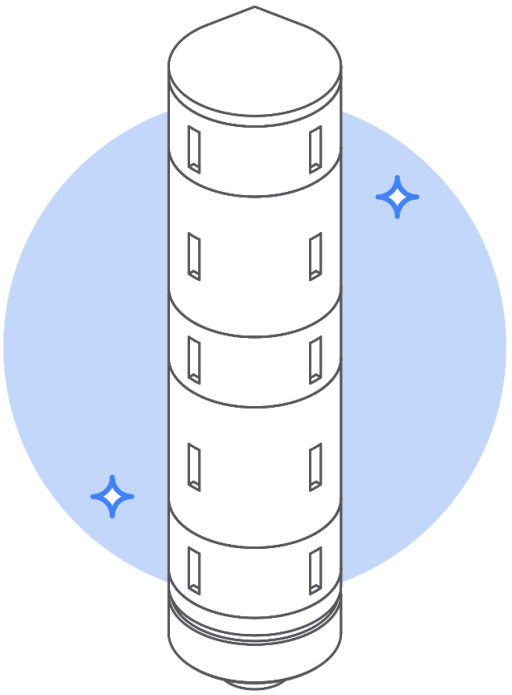
b x1

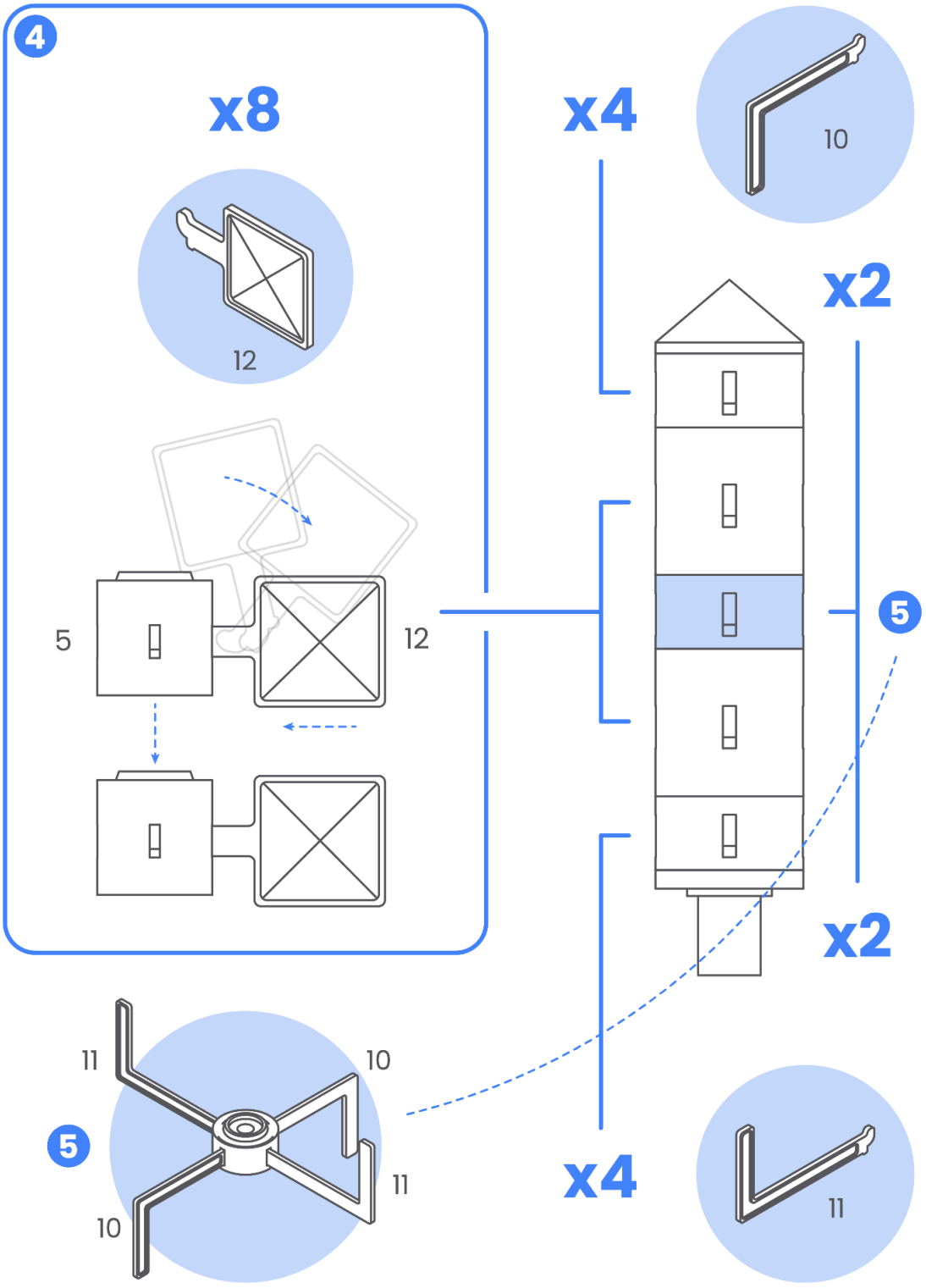
c

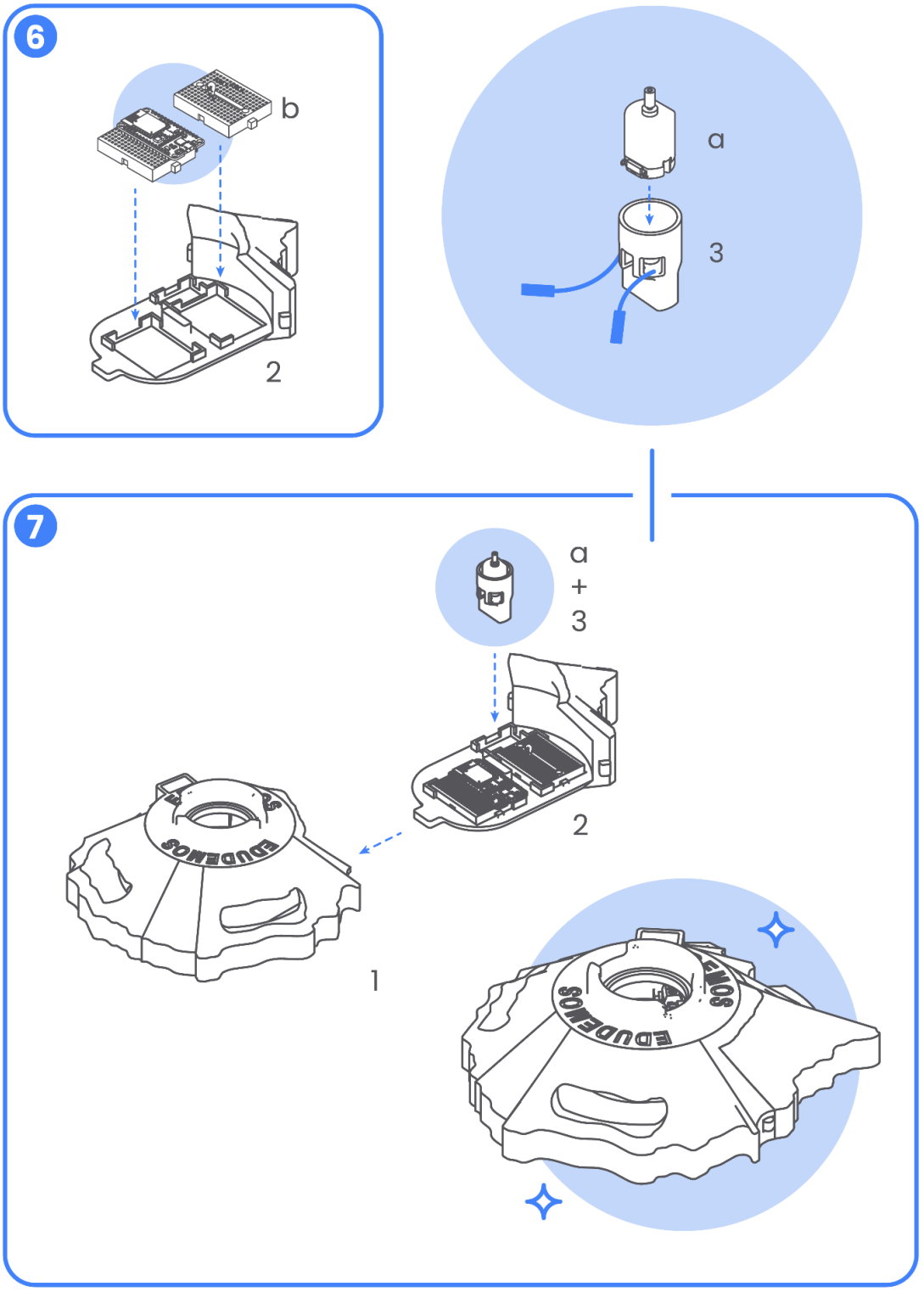

1
x2
4
5


2
4
6
7


3

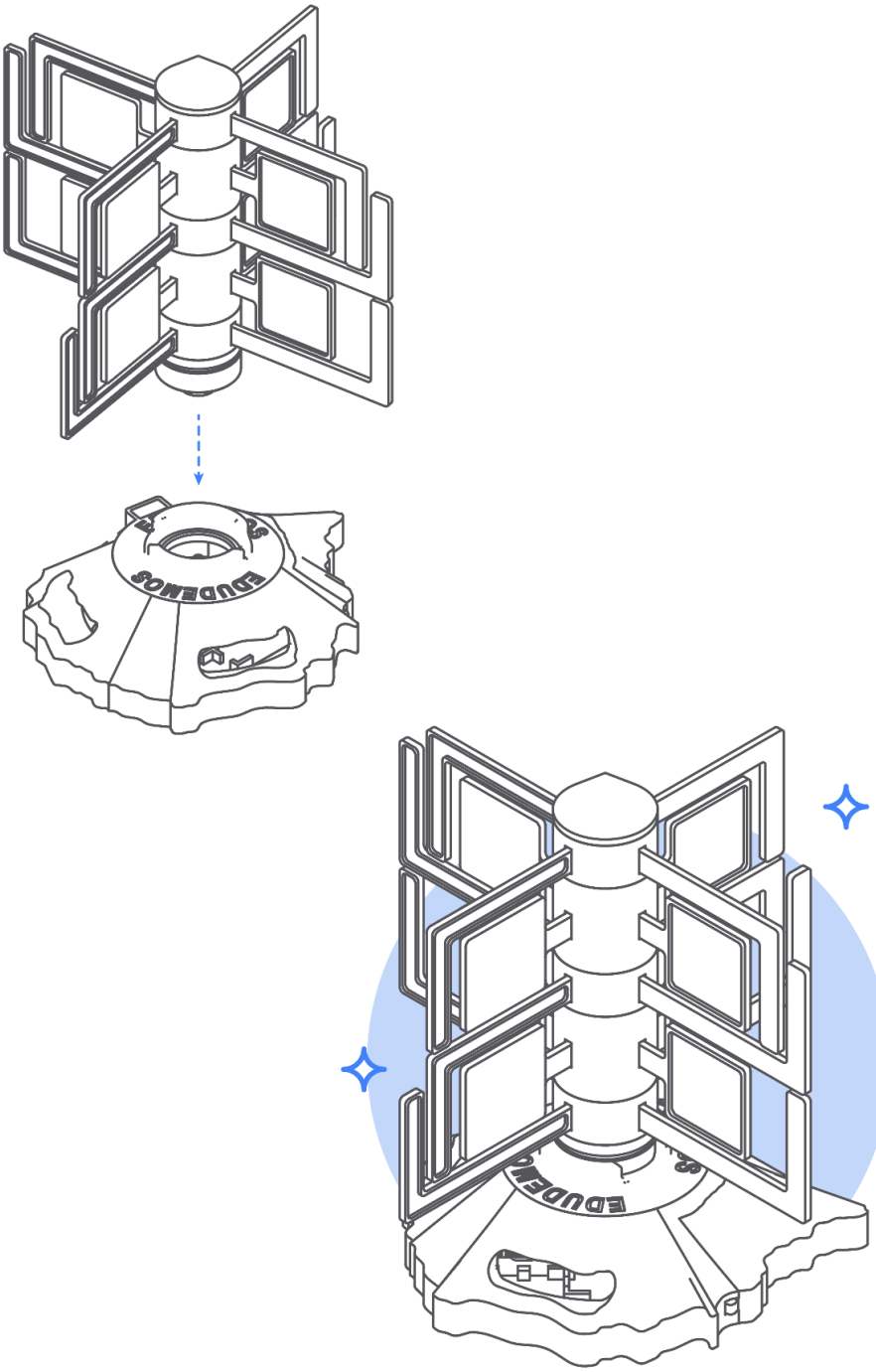



4
x8
12
5
12
x4
10
x2
5
x2
11
10
5
10
11
x4
11



Funded by the European Union
Gerda Stetter Stiftung
Technik macht Spaß!
FUNDACIÓN SERGIO ALONSO
Finnova
gbs sg.ch

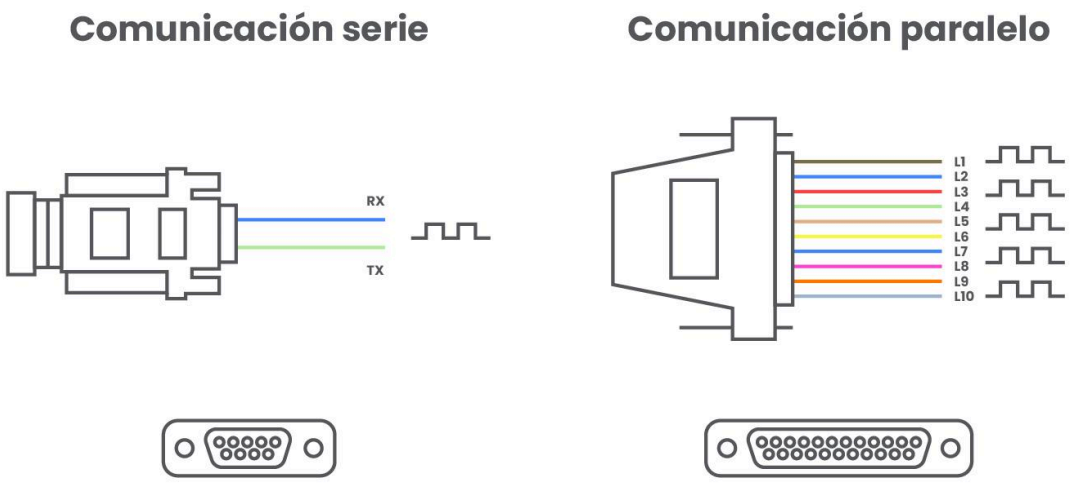
8



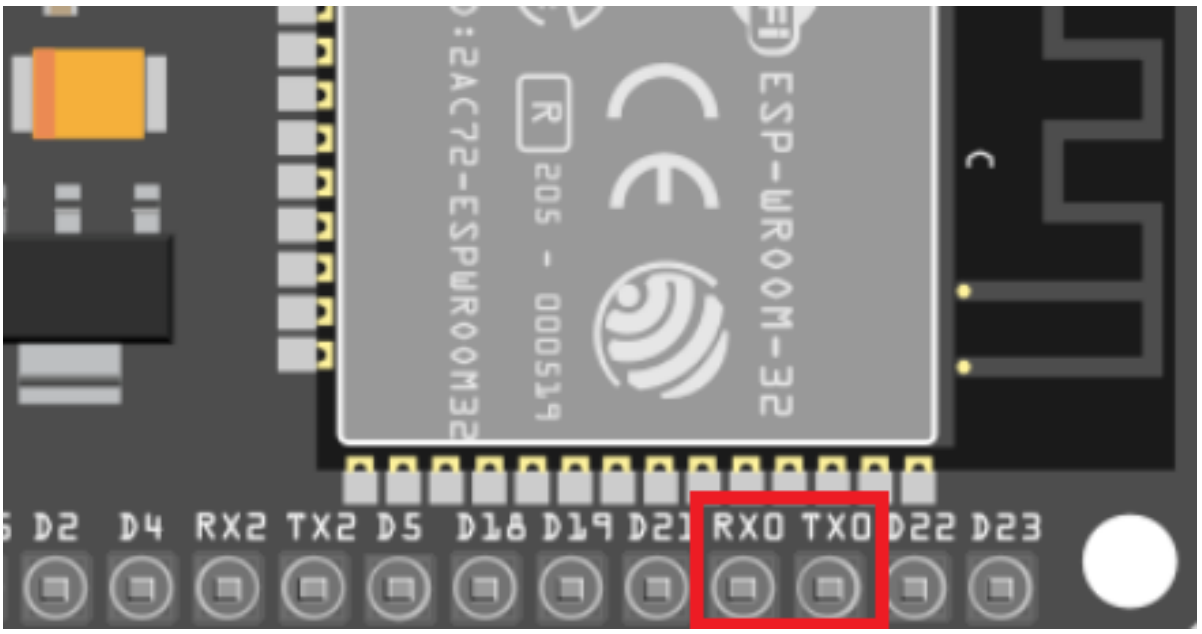
6. DATOS DE LOS SENSORES

6. 1. Puerto Serie

La comunicación a través del puerto serie es uno de los métodos más utilizados para que los dispositivos se conecten entre sí. Es también la principal vía por la cual el microcontrolador del NodeMCU, el ESP32, se comunica tanto con el ordenador como con otros componentes. En un puerto serie, los datos se transmiten y reciben secuencialmente, enviando bits de uno en uno a través de un único canal. En contraste, un puerto paralelo transmite información simultáneamente por varios canales.



Existen varios tipos de puertos serie, siendo el más popular el puerto USB (Universal Serial Bus). Sin embargo, también es posible encontrar el antiguo puerto serie RS-232, así como otros protocolos de comunicación serie como RS-485, I2C, SPI, entre otros. La comunicación serie es fundamental en los microcontroladores, ya que todos ellos cuentan con al menos una unidad UART (Universal Asynchronous Receiver/Transmitter). Este tipo de puerto serie utiliza dos canales: uno para transmitir datos (TX) y otro para recibirlos (RX).



La placa NodeMCU puede comunicarse con el ordenador a través de un puerto USB, el mismo que se utiliza para programarla. Esto establece un canal de comunicación bidireccional que permite tanto el envío como la recepción de datos. La herramienta que nos permite ver los datos enviados desde el NodeMCU y, a su vez, enviar información desde el ordenador al NodeMCU es **el Monitor Serial, que ya hemos nombrado y utilizado anteriormente**. Esta interfaz está integrada en el Arduino IDE y se puede localizar en la esquina superior derecha del programa, representada por el siguiente icono:

Al abrirlo, el Monitor Serial del Arduino IDE se presenta de la siguiente manera:

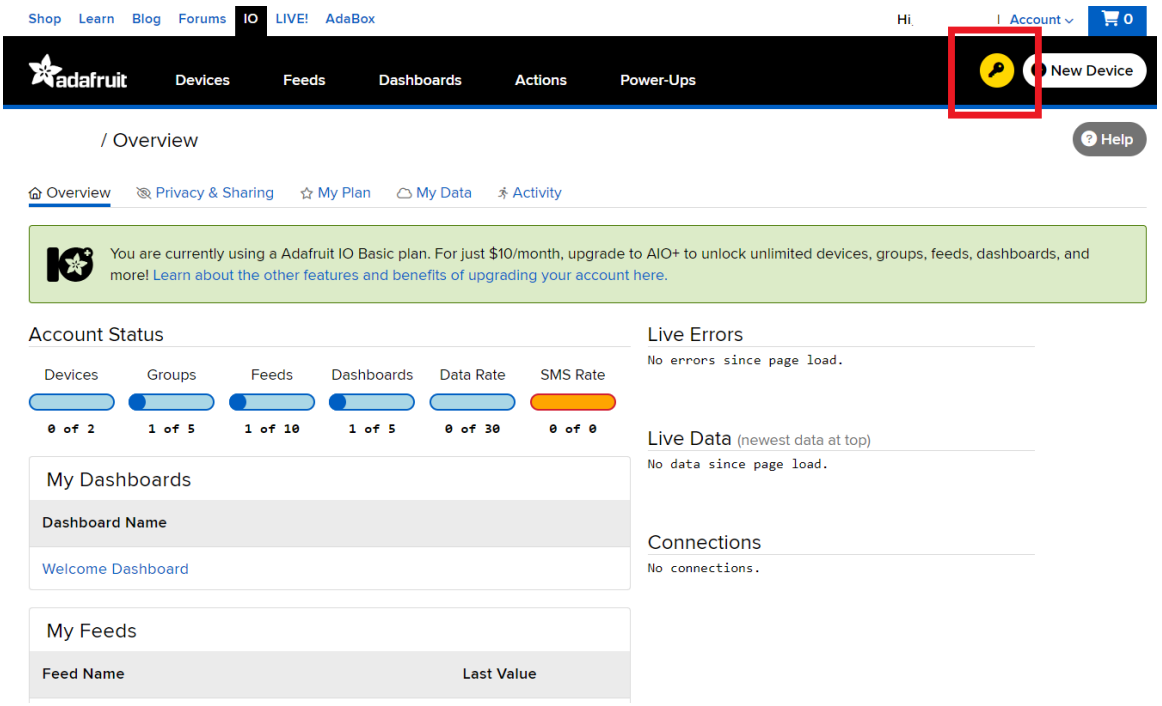
Como puedes observar, la interfaz es bastante simple y fácil de usar, con los siguientes elementos principales:

- **Envío de datos:** En el campo superior en blanco, puedes escribir y enviar datos desde el PC al ESP32.
- **Recepción de datos:** La mayor parte del monitor está dedicado a mostrar los datos recibidos desde el ESP32.
- **Autoscroll:** Permite activar o desactivar el desplazamiento automático en la ventana de recepción de datos.
- **Mostrar timestamp:** Opción para ver la hora exacta en que se recibe cada dato.
- **Ajuste de línea:** Agrega un tipo de ajuste de línea al final del mensaje enviado.
- **Baud Rate o Tasa de baudios:** Define la velocidad de transmisión en baudios (símbolos por segundo). Es esencial que la tasa de baudios sea la misma tanto en el emisor como en el receptor para una comunicación correcta.

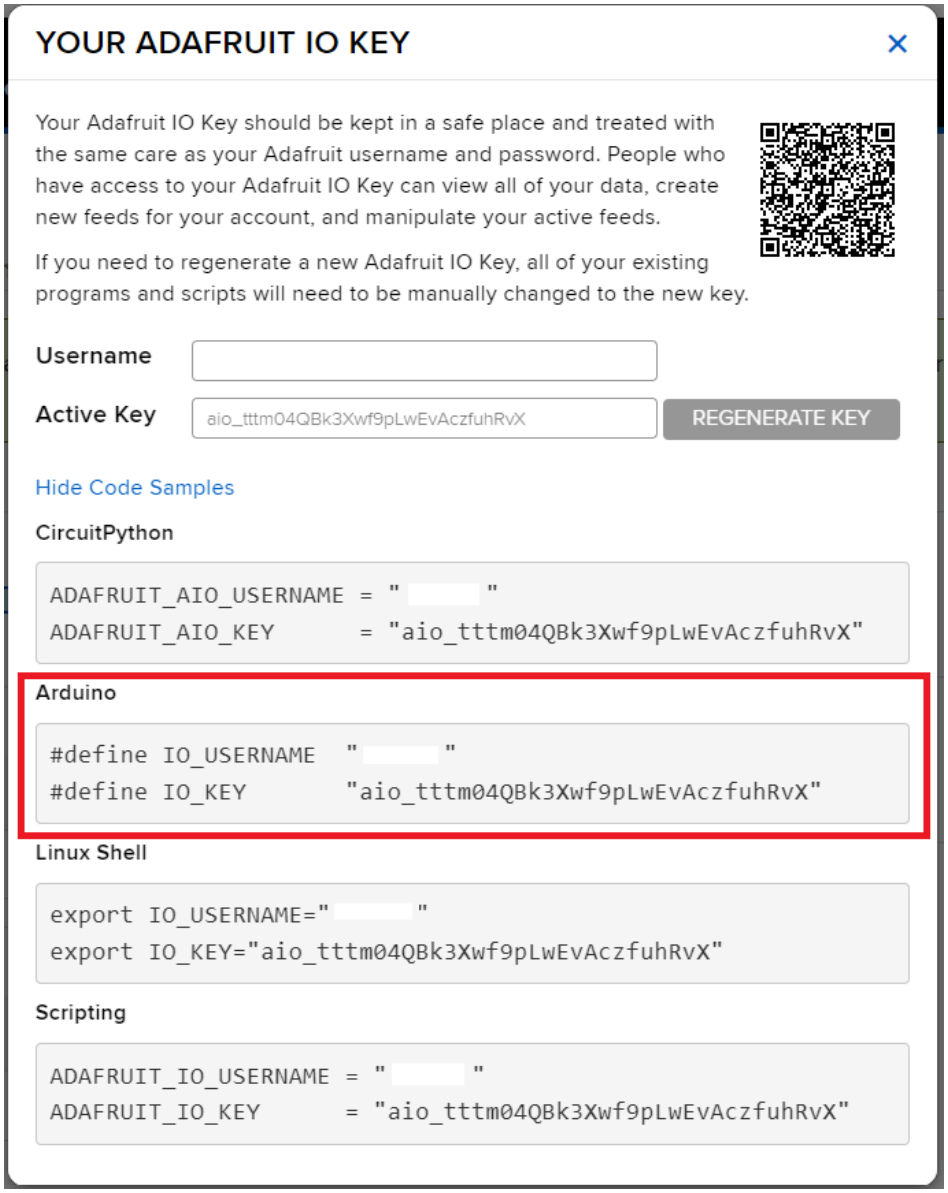
6. 2. Adafruit IO

Adafruit IO es una plataforma de Internet de las Cosas (IoT) que permite a los usuarios recopilar, visualizar y gestionar datos de dispositivos conectados a la red. Ofrecida por Adafruit, una empresa conocida por desarrollar hardware y software de código abierto, Adafruit IO facilita la conexión y el control de proyectos de hardware como microcontroladores (Arduino, ESP32, Raspberry Pi, etc.) con la nube.

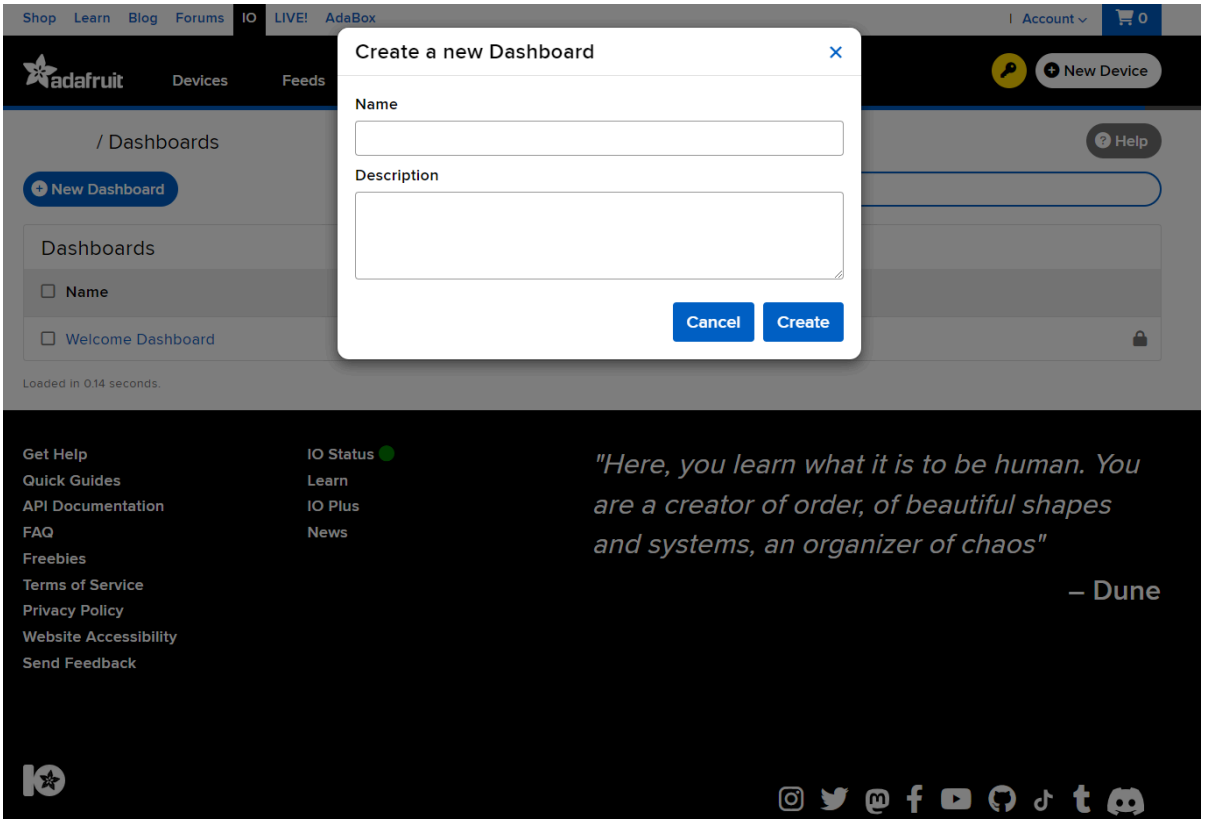
Para utilizar esta plataforma deberemos crearnos una cuenta de usuario en su página web. Una vez tengamos una cuenta, pincharemos en la parte superior en la sección que pone IO, junto a Shop, Blog y Forum. Tendremos la siguiente ventana a la vista, y deberemos pinchar sobre el icono con la llave amarilla:



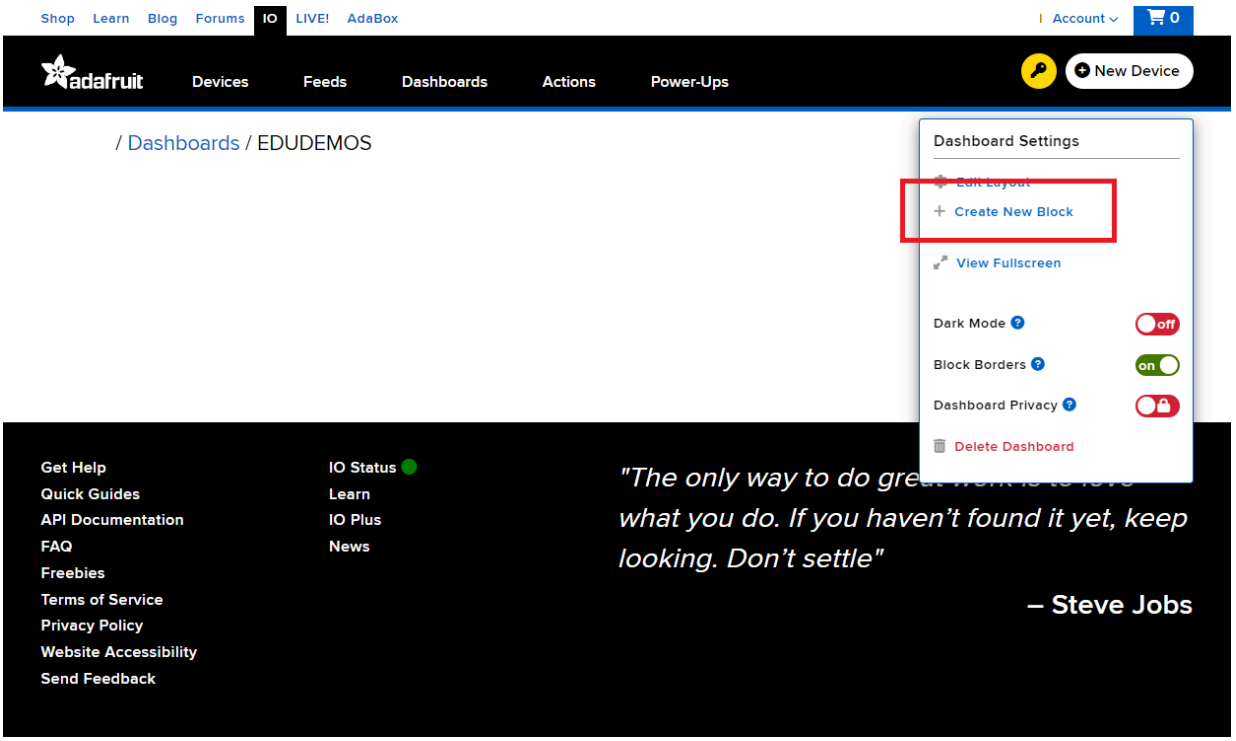
Cuando pinchemos ahí, se abrirá una ventana como la de la siguiente imagen, donde necesitaremos ir a la sección de "***Arduino***" para tomar los datos que aparecen ahí y modificar las mismas variables en el código de configuración (*config.h*) de los Demostradores que mencionamos anteriormente.



Lo siguiente es crear un ***Dashboard***. Para ello vamos a pinchar en el menú de la parte superior "*Dashboards*" y luego en el botón grande y azul que pone "*New Dashboard*". Veremos entonces una ventana emergente que nos pedirá darle nombre y una descripción a nuestro nuevo Dashboard. Vamos a crear uno con el nombre que queramos y podemos añadirle una descripción de para qué lo usaremos.



Ahora vamos a proceder a entrar al *Dashboard* que hemos creado, donde veremos una página prácticamente vacía. Para añadir representaciones de los datos que llegarán desde nuestro NodeMCU en el Demostrador deberemos pinchar sobre el icono de la tuerca situado arriba a la derecha, que nos abrirá un menú donde tendremos la opción de "***Create New Block***", es decir, Crear un Nuevo Bloque.



Cuando intentamos crear un nuevo bloque, se abre un listado de posibles gráficos e indicadores que incluir en nuestro *Dashboard*. Para representar los datos en Adafruit IO, debemos crear lo que llaman Feeds, o fuente de datos. Estos almacenan la información enviada o recibida desde diferentes dispositivos conectados a la plataforma. Los Feeds que necesitaremos son: ***temperature, humidity, ldr, ldr1Feed, ldr2Feed, microvoltsMotor, microvoltsSolar, ledHighFeed, ledGoodFeed y ledColdFeed.***

Utilizaremos los gráficos de lineales, los medidores y los indicadores para representar la información recibida desde el ESP32 del NodeMCU que estamos utilizando de la siguiente manera:

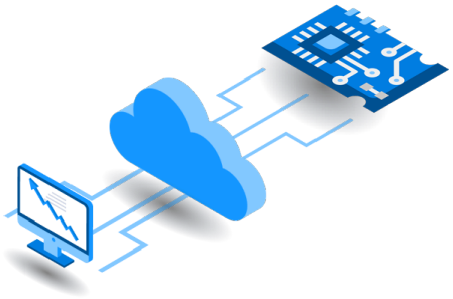
- Gráfico lineal para representar la Temperatura en ºC a lo largo del tiempo.
- Gráfico lineal para representar la Humedad relativa en % del aire a lo largo del tiempo.
- Gráfico lineal para representar la tensión generada por el Motor de CC en uV a lo largo del tiempo.
- Gráfico lineal para representar la tensión generada por el Panel Solar en uV a lo largo del tiempo.

- Gráfico medidor para representar la Temperatura actual en ºC.
- Gráfico medidor para representar la Humedad relativa actual en %.
- Gráfico medidor para representar la tensión generada por el Motor de CC actual en uV.
- Gráfico medidor para representar la tensión generada por el Panel Solar actual en uV.
- Gráfico medidor para representar la Intensidad Lumínica media actual.
- Indicador para representar el estado del LED de Cold.
- Indicador para representar el estado del LED de Good.
- Indicador para representar el estado del LED de Heat.

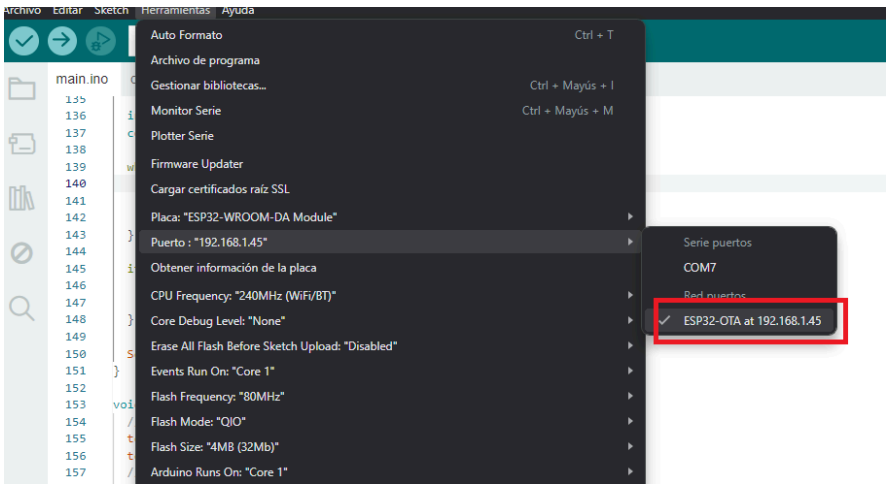
El objetivo será obtener un Dashboard al final como el siguiente:

7. OTA (Over-The-Air) UPDATES

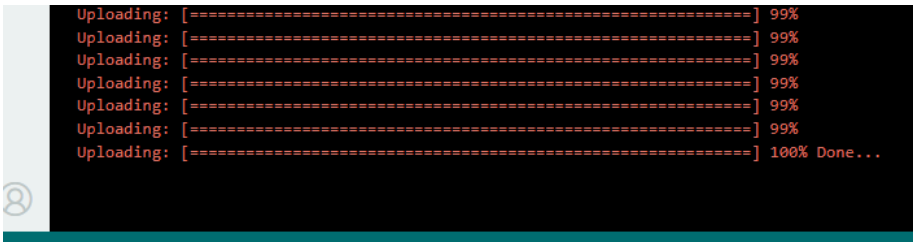
Las actualizaciones OTA permiten **actualizar el firmware de un dispositivo de manera inalámbrica**, sin necesidad de conectarlo físicamente a una computadora. Esto es particularmente útil en dispositivos IoT, como el NodeMCU, que suelen estar ubicados en lugares remotos o de difícil acceso. Con las actualizaciones OTA, puedes cargar nuevo código o mejoras en el dispositivo sin tener que desmontarlo o conectarlo mediante un cable USB.



Para utilizar las actualizaciones OTA, debemos cargar una primera vez el código completo de los Demostradores a nuestra placa con ESP32. Una vez subido y establecida la conexión con la red Wi-Fi y la plataforma Adafruit IO, podremos seleccionar entre los puertos del Arduino IDE (en *Tools* > *Port*) un puerto OTA que corresponderá con el NodeMCU que estamos utilizando (tendrá de nombre algo como "***ESP32-OTA at***" y la dirección IP del dispositivo en la red Wi-Fi.



Ahora, si quieres subir un código a través de actualizaciones OTA, simplemente pincha en el botón de subir código. Se te solicitará una contraseña (que por defecto es *admin*) y posteriormente el programa comenzará a compilarse. La primera vez que utilices OTA, el Sistema Operativo te puede solicitar que concedas permiso para que pueda llevarse a cabo la tarea. Simplemente permite que la aplicación se ejecute y podrás subir códigos a tu placa con ESP32 conectada a la red WiFi sin necesidad de cables USB todas las veces que lo necesites.



Si en algún momento necesitas cambiar el nombre de la placa cuando es listada en los puertos disponibles o la contraseña para poder realizar la actualización remota, simplemente dirígete al código fuente de los Demostradores y sustituye los valores entre paréntesis y comillas dobles que puedes ver en las líneas que se muestran a continuación (están ubicadas dentro de la función ***setupOTA()***):

```
// Initialize ArduinoOTA for over-the-air updates
ArduinoOTA.setHostname("ESP32-OTA");
ArduinoOTA.setPassword("admin");
```

8. PROBLEMAS TÍPICOS

Problema	Comprobación	Solución
No puedo cargar el programa a la placa NodeMCU.	Comprueba que el cable USB esté bien conectado en ambos extremos.	Desconecta y vuelve a conectar adecuadamente el cable en ambos extremos.
	Comprueba que el cable que estás utilizando no sea solamente de carga.	Sustituye el cable por otro que permite transferir datos.
	Comprueba que el puerto USB de tu ordenador	Cambia el puerto USB.

Me da error al compilar el programa.	funcione correctamente.	
	Comprueba que has seleccionado el puerto COM adecuado para la placa NodeMCU.	Selecciona el puerto COM donde se encuentra conectado el NodeMCU.
	Comprueba que tienes instalado los drivers necesarios (el CP210X).	Instala los drivers como se indica en el apartado *3.1 Requisitos mínimos*.
	Comprueba que el error está relacionado con alguna librería.	Instala las librerías necesarias indicadas en el apartado *3.2 Librerías necesarias*.
	Comprueba que el error está relacionado con la sintaxis del código.	Revisa las líneas donde te da error y cambia lo que encuentres erróneo comparando con el código original que tienes en GitHub.
El ESP32 no se conecta a la red WiFi.	Comprueba las credenciales definidas para conectarse al WiFi.	Edita las credenciales (SSID y contraseña) de la red WiFi a los datos que quieres utilizar.
El ESP32 no envía los datos a Adafruit IO.	Comprueba las credenciales definidas para conectarse a la plataforma.	Edita las credenciales (*Key* y nombre de usuario) de la plataforma a los datos que quieres utilizar.
La placa NodeMCU no enciende.	Comprueba que está conectada por cable a una fuente de alimentación adecuada.	Conecta la placa a un puerto USB u otra fuente de 5V.
Los LEDs no funcionan correctamente.	Comprueba que están conectados adecuadamente en la protoboard.	Conecta cada terminal del LED en el lugar correspondiente como indican los esquemas.
	Comprueba que están bien polarizados.	Conecta el terminal más largo del LED al lado positivo y el más corto al lado negativo o GND.
	Comprueba que tienen la resistencia limitadora conectada adecuadamente.	Conecta una resistencia de al menos 220 ohmios en serie con el LED.
El sensor DHT no entrega lecturas.	Comprueba que está correctamente alimentado.	Alimenta el DHT con 3'3V en el terminal positivo y GND en el negativo.
	Comprueba que el terminal de datos esté conectado al mismo pin de la placa que se	Conecta el terminal de datos al pin correspondiente (D32).

	indica en el programa.	
	Comprueba que el modelo del sensor DHT sea el adecuado.	Utiliza un sensor DHT-22 o cambia el tipo de sensor DHT en el programa.
Las LDR no entregan medidas razonables.	Comprueba que las LDR estén conectadas adecuadamente.	Conecta las LDR adecuadamente en un partidor de tensión si las utilizas independientes o directamente a los pines de la placa correspondientes si las utilizas en módulo.
	Comprueba que las LDR estén conectadas a los pines de la placa adecuados.	Conecta las LDR a los pines de la placa que permiten realizar lecturas analógicas (D34 y D35).
El Servomotor no funciona.	Comprueba la alimentación del Servomotor.	Conecta la alimentación positiva del Servomotor al pin de 5V y la negativa a GND.
	Comprueba que el Servomotor esté bien conectado al pin de control.	Conecta el terminal de señal de control del Servomotor adecuadamente al D25.
El detector de tensión no entrega medidas razonables.	Comprueba que el módulo está bien alimentado.	Alimenta el módulo detector de tensión con 3'3V y GND.
	Comprueba que el terminal de datos está conectado adecuadamente.	Conecta correctamente la señal de salida analógica del módulo al pin D36.
Las medidas obtenidas con el panel solar no son lógicas.	Comprueba la orientación del panel solar.	Conecta el panel adecuadamente al módulo detector de tensión haciendo coincidir los terminales positivos y negativos.

PROBLEMAS MECÁNICOS		
Las aspas en forma de L no se mueven de forma independiente	Comprueba la separación entre discos.	Si es necesario, saca un poco las espigas de madera de los discos grandes para reducir su fricción.
Quiero desmontar las aspas y horas de mi demostrador	Comprueba levantar las piezas y luego sacarlas.	Para desmontar las aspas y las hojas de los

		demostradores, realiza los mismos pasos que usaste para montarlo pero en el orden inverso.
No me encaja la base con el cajón del demostrador	Comprueba que el cajón está bien alineado con la base.	Alinea el cajón con el hueco de la base para poder encajarlos.
El motor no tiene cables para conectarlo en mi montaje.	Comprueba que el motor disponga de terminales para soldar cables.	Suelda con el soldador de estaño dos cables, uno en cada terminal del motor de CC.
El motor no entrega tensión alguna cuando se mueve.	Comprueba que está conectado adecuadamente.	Conecta el motor adecuadamente conectando un terminal a la entrada del partidor de tensión y el otro terminal a GND.

EDUDEMOS

EDUcating through Sustainable DEMOnstrators

Funded by the European Union

info@edudemos.eu

@edudemos

@edudemos

www.edudemos.eu