SENSORES Y ACTUADORES – TST 2024

Tp8-e6 #66. Control de Nivel de un Tanque de Almacenamiento de Agua

- Descripción: Implementar el control de nivel de un tanque de almacenamiento de agua, asegurando que el sistema de riego funcione correctamente cuando hay suficiente agua disponible.
- Responsable: Un integrante

Tarea:

- ☑ 1. Configurar los sensores para medir el nivel del agua en el tanque.
- ☑ 2. Implementar la lógica de control para que el sistema de riego se desactive si el nivel de agua es bajo.
- ☑ 3. Validar el funcionamiento del control de nivel.

Descripción General del Proyecto

El objetivo es mantener un sistema de riego automatizado, controlado por el nivel de agua en un tanque. La bomba solo debe funcionar cuando hay suficiente agua en el tanque. Si el nivel de agua es bajo, el sistema de riego debe desactivarse para evitar dañar la bomba o trabajar en seco.

Componentes Necesarios

1. **ESP32**: Microcontrolador para gestionar el control de la bomba y los sensores.
2. **Bomba Elevadora Centrífuga de Agua QC100 Vasser**: Encargada de bombear el agua para el riego.
3. **Sensor de Nivel de Agua** (ej. sensor ultrasónico o sensor de nivel de agua por flotador).
4. **Relé de Estado Sólido** o **Relé electromecánico**: Para controlar la activación/desactivación de la bomba.
5. **Fuente de alimentación** para el ESP32 y la bomba.
6. **Cables de conexión**.

Tareas Desglosadas

1. Configurar los Sensores para Medir el Nivel del Agua

a. Sensor ultrasónico (HC-SR04)

Un sensor ultrasónico es una buena opción para medir el nivel de agua. El HC-SR04 mide la distancia entre el sensor y la superficie del agua. A partir de esa distancia, se puede calcular el nivel de llenado del tanque.

Conexión del sensor al ESP32:

Pin del Sensor	Pin del ESP32
VCC	3.3V
GND	GND
TRIG	GPIO 5
ECHO	GPIO 18

Código para leer el sensor ultrasónico:

```
cpp
Copiar código
#define TRIG_PIN 5
#define ECHO_PIN 18

void setup() {
  Serial.begin(115200);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
}

long medirDistancia() {
  // Enviar pulso al TRIG
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  // Leer el tiempo del ECHO
  long duration = pulseIn(ECHO_PIN, HIGH);

  // Calcular la distancia en cm
  long distance = duration * 0.034 / 2; // Velocidad del sonido: 0.034 cm/us

  return distance;
}

void loop() {
  long distancia = medirDistancia();
  Serial.print("Distancia: ");
  Serial.print(distancia);
  Serial.println(" cm");
  delay(1000);
}
```

b. Sensor de nivel de agua por flotador

Otra opción es utilizar un sensor de flotador que activa o desactiva un interruptor dependiendo del nivel de agua. Este sensor es más simple de configurar, ya que solo envía una señal alta o baja.

2. Implementar la Lógica de Control

Debes configurar el sistema para que la bomba se encienda si el nivel de agua está por encima de un cierto umbral (por ejemplo, el 30% del tanque lleno) y se apague si el nivel cae por debajo de ese umbral.

El control de la bomba se hace a través de un relé que será activado o desactivado según la lectura del sensor.

Conexión del relé al ESP32 y la bomba:

Pin del Relé	Pin del ESP32	Descripción
VCC	3.3V	Alimentación relé
GND	GND	Tierra
IN	GPIO 16	Control desde ESP

Código para control de la bomba:

```
cpp
Copiar código
#define RELAY_PIN 16
#define LOW_WATER_THRESHOLD 30 // Umbral bajo del agua (en cm o porcentaje)
#define TRIG_PIN 5
#define ECHO_PIN 18

void setup() {
  Serial.begin(115200);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(RELAY_PIN, OUTPUT);
  digitalWrite(RELAY_PIN, LOW); // La bomba comienza apagada
}

long medirDistancia() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  long duration = pulseIn(ECHO_PIN, HIGH);
  long distance = duration * 0.034 / 2; // Distancia en cm
  return distance;
}

void controlarBomba() {
  long distancia = medirDistancia();
  Serial.print("Distancia: ");
  Serial.print(distancia);
  Serial.println(" cm");
```

```
  if (distancia > LOW_WATER_THRESHOLD) {
    // Si el nivel de agua está bajo, apagar bomba
    digitalWrite(RELAY_PIN, LOW);
    Serial.println("Nivel bajo: Bomba apagada");
  } else {
    // Si hay suficiente agua, encender bomba
    digitalWrite(RELAY_PIN, HIGH);
    Serial.println("Nivel suficiente: Bomba encendida");
  }
}

void loop() {
  controlarBomba();
  delay(5000); // Verificar cada 5 segundos
}
```

3. Validar el Funcionamiento del Control de Nivel

a. Validación con monitoreo serial:

Puedes observar el nivel de agua medido y el estado de la bomba en el monitor serial para verificar si el sistema responde correctamente.

b. Pruebas en la realidad:

- Llena el tanque de agua a diferentes niveles.
- Observa si la bomba se activa/desactiva de acuerdo al nivel.
- Verifica que la bomba no funcione cuando el nivel es bajo (por debajo del umbral).

Consideraciones de Seguridad

1. **Bomba y Relé:** Trabaja con la bomba con mucho cuidado, ya que maneja corriente alterna de alto voltaje. Asegúrate de que el relé sea adecuado para la carga de la bomba.
2. **Sensores:** Protege los sensores del agua y asegúrate de que estén adecuadamente conectados para evitar lecturas erróneas.

Recursos Adicionales

- **Bomba QC100**: Consulta el manual de tu bomba para verificar los requisitos eléctricos y asegúrate de que el relé sea capaz de manejar la carga.
- **IoT**: Si planeas integrar este sistema con una aplicación IoT, puedes usar la red Wi-Fi del ESP32 para enviar los datos de nivel de agua a un servidor o dashboard (como Blynk, MQTT, etc.).

Monitoreo de nivel de agua sin contacto basado en IoT con ESP32 y HC-SR04

En este proyecto de IoT, crearemos un servidor web de monitoreo de nivel de agua utilizando un sensor ultrasónico HC-SR04 y ESP32. Será un sistema de medición de nivel de agua sin contacto. Primero, aprenderemos a interconectar HC-SR04 con ESP32. Después de eso, veremos cómo programar nuestra placa ESP32 con el sensor ultrasónico para construir nuestro servidor web de monitoreo de agua. El servidor web mostrará el nivel de agua actual medido como la distancia en cm por HC-SR04. ¡Comencemos!

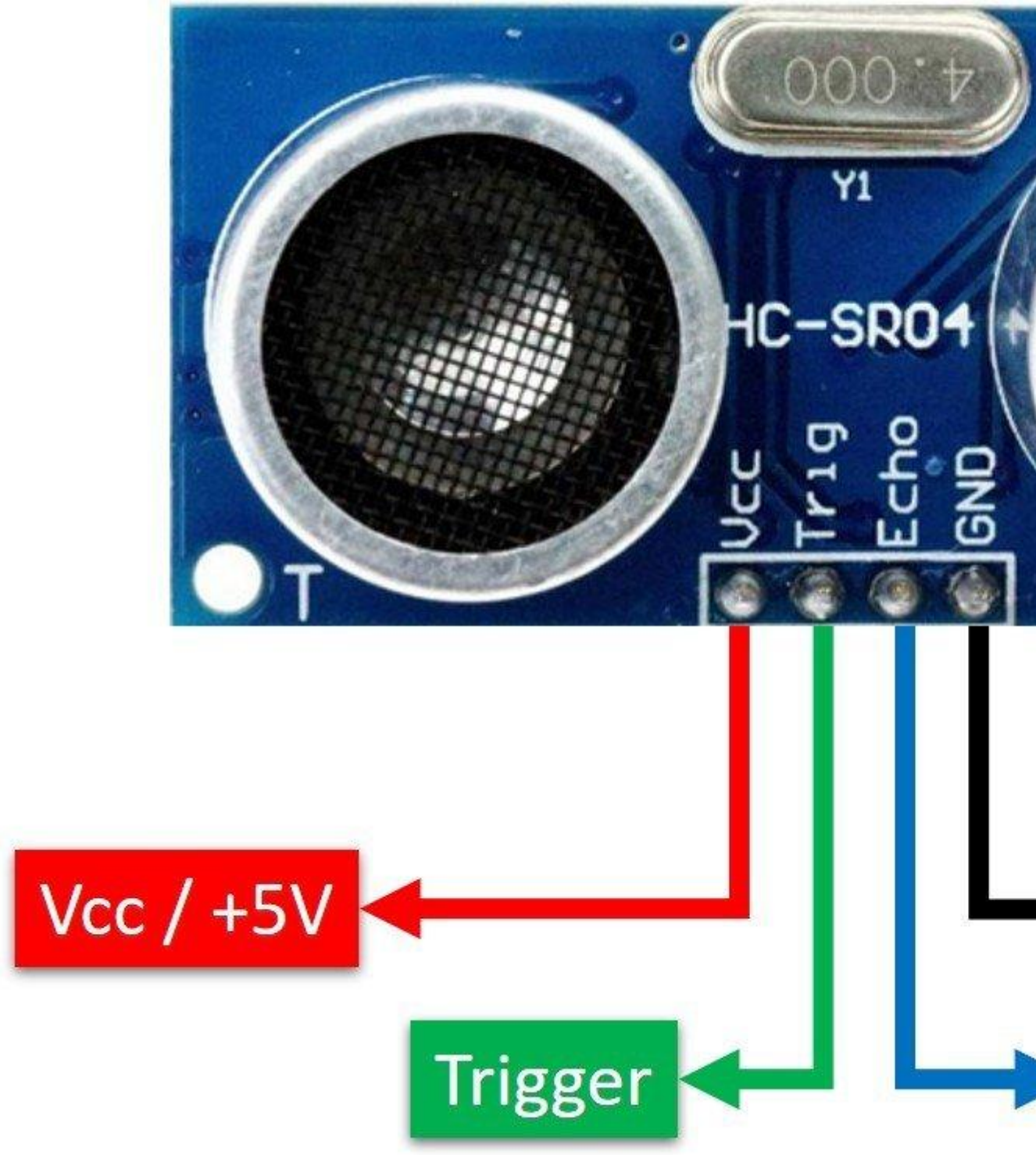
Ahora, primero analicemos los distintos componentes del sistema de monitoreo de nivel de agua sin contacto basado en IoT, como el sensor ultrasónico, el diagrama de conexión con ESP32 y el boceto de Arduino. Al final, veremos una demostración en video.

Introducción al sensor ultrasónico HC-SR04

Para interconectar el sensor ultrasónico HC-SR04 con ESP32, debemos conocer la funcionalidad de cada pin del sensor ultrasónico. Al conocer la funcionalidad de los pines de entrada y salida, podremos identificar qué pines GPIO del ESP32 se deben usar para interconectar con HC-SR04.

Distribución de pines del modelo HC-SR04

La figura que se muestra a continuación muestra la configuración de pines de un sensor ultrasónico. Consta de cuatro pines: Vcc, tierra, disparador y pin de eco.



Vcc y Ground se utilizan para alimentar el sensor. Debemos suministrar 5 voltios al pin Vcc y conectar el pin GND con el terminal de tierra de la fuente de alimentación.
Disparador: es un pin de entrada. El pin disparador se utiliza para activar el sensor ultrasónico para iniciar la medición de distancia o el cálculo de distancia.

Cuando los usuarios desean obtener mediciones de distancia del sensor, aplicamos un pulso de 10 µs a este pin.
Eco: Este es un pin de salida de pulso. El pin de eco produce un pulso como salida. El ancho del pulso o el tiempo de activación del pulso depende de la distancia entre el sensor ultrasónico y el obstáculo que se coloca frente al sensor HC-SR04. En condiciones de inactividad, este pin permanece en un nivel bajo activo.
En la siguiente sección se proporcionan más detalles sobre el funcionamiento del sensor ultrasónico.

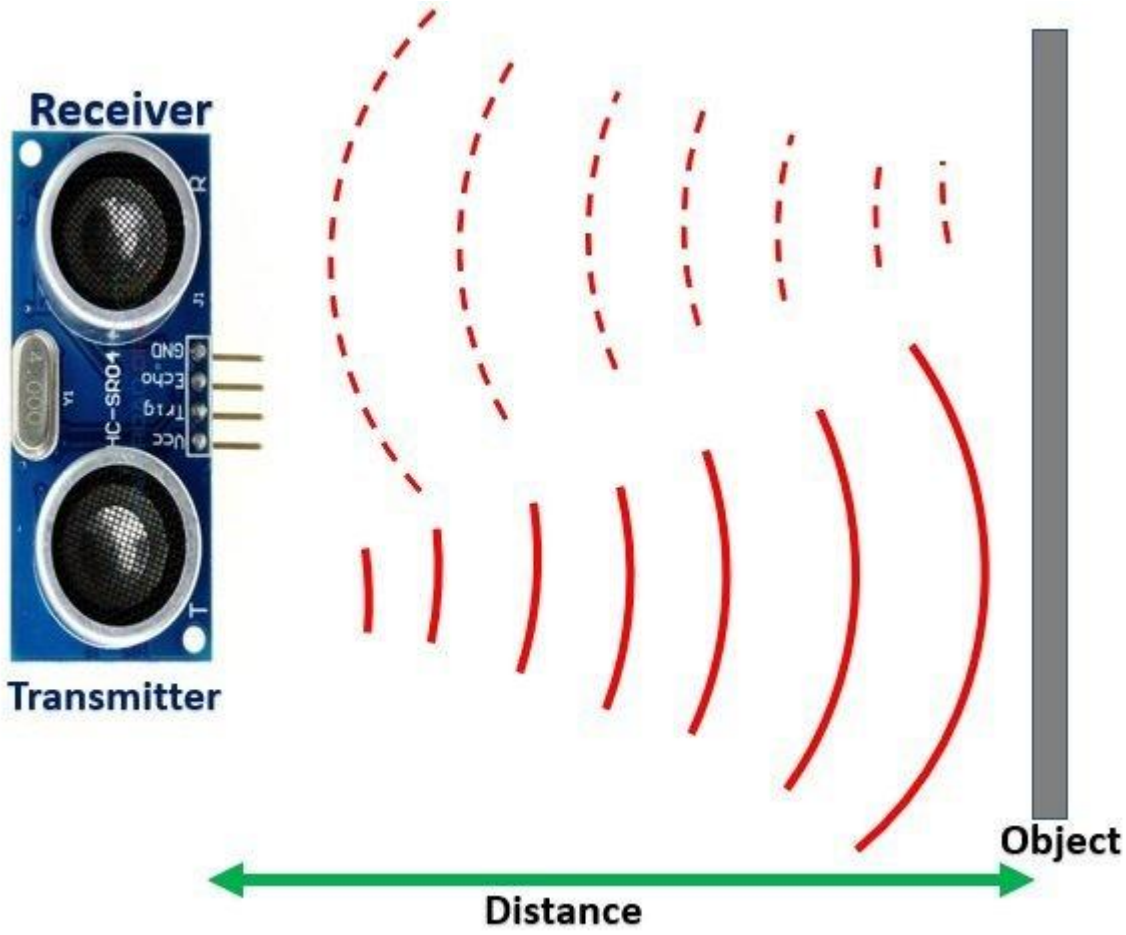
¿Cómo funciona el sensor HC-SR04?

El sensor ultrasónico HC-SR04 mide la distancia mediante ondas ultrasónicas inaudibles de una frecuencia de 40 KHz. Al igual que las ondas sonoras, las ondas ultrasónicas viajan por el aire y, si hay algún obstáculo frente a ellas, se reflejan según su ángulo de incidencia. Además, si se coloca un objeto en paralelo a un transmisor ultrasónico, las ondas ultrasónicas se reflejan exactamente en un ángulo de 180 grados. Por lo tanto, para la medición de distancia con el sensor HC-SR05, colocamos el objeto bajo prueba exactamente en una posición paralela a un sensor ultrasónico como se muestra en la siguiente figura.


Receiver
R
J1
GND
Echo
Trig
Vcc
HC-SR04
Y1
4.000
T
Transmitter

El sensor ultrasónico HC-SR05 consta de dos módulos básicos, como son el transmisor ultrasónico y el módulo receptor ultrasónico. El circuito transmisor convierte una señal eléctrica en una ráfaga de 40 KHz de 8 pulsos de ondas de sonar. La señal eléctrica de entrada al circuito transmisor es una entrada de pulso de 10 µs al pin de activación del sensor HC-SR04. Como mencionamos anteriormente, aplicamos esta señal de entrada de activación a través de ESP32 o cualquier microcontrolador. Por otro lado, el circuito receptor ultrasónico escucha estas ondas ultrasónicas que son producidas por el circuito transmisor.

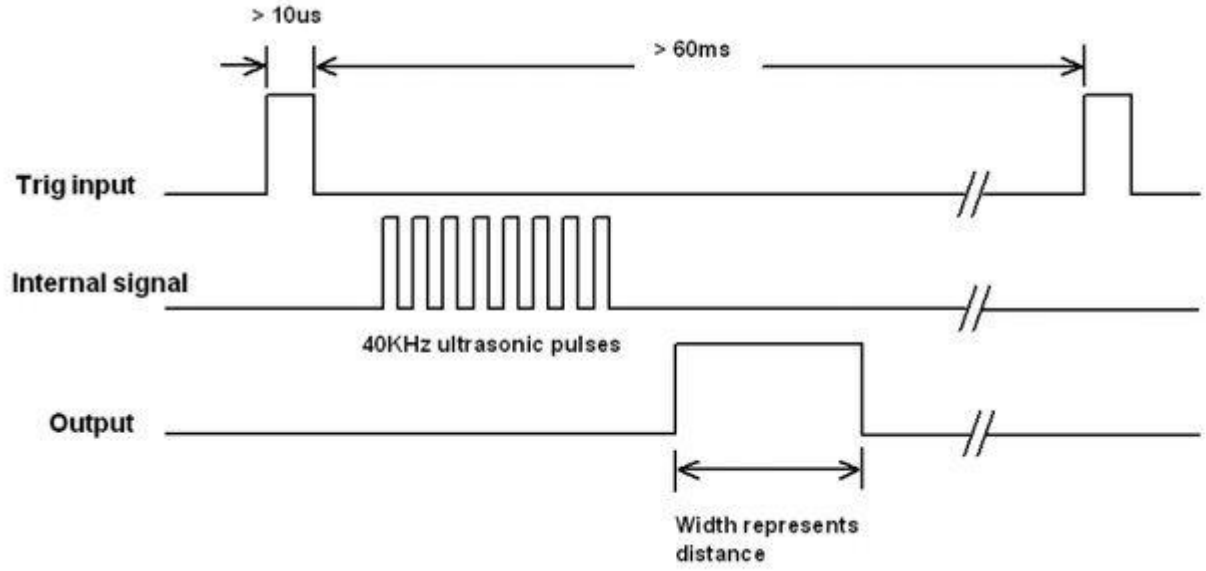
Medir el tiempo de pulso de eco HC-SR04 con ESP32



- Para comenzar a medir con HC-SR04, primero aplicamos un pulso de 10 µs al pin de disparo del sensor HC-SR04 desde el pin de salida digital ESP32.
- Tan pronto como la señal de activación de entrada de 10 µs se activa a un nivel bajo, el circuito transmisor produce una ráfaga de 8 pulsos de sonar ultrasónico. Al mismo tiempo, el pin Echo también realiza una transición de un nivel lógico bajo a un nivel lógico alto.
- Cuando el pin Echo se activa, comenzamos a medir el tiempo con la función de medición de duración ESP32.

- Estas ondas viajan a través del aire y si hay algún objeto colocado paralelo al sensor, estas ondas se reflejan después de una colisión con el objeto.
- Tan pronto como las ondas ultrasónicas recibidas por el circuito receptor chocan con un objeto, el pin de eco pasa a nivel bajo. El ESP32 detecta esta transición de la señal de salida de eco de un nivel alto activo a un nivel bajo activo y detiene la medición.

En resumen, al medir el tiempo de activación de la señal de pulso de salida del eco, podemos medir la distancia. La siguiente figura ilustra la señal de salida del eco con respecto a la señal de activación de entrada y 8 pulsos del sonar.



La duración durante la cual la señal de salida del eco permanece alta depende de la distancia entre el sensor ultrasónico y el objeto que colocamos frente al sensor. Cuanto mayor sea la distancia, mayor será el tiempo que las ondas del sonar tardarán en llegar de regreso al circuito receptor ultrasónico. Esto se debe a que las ondas ultrasónicas viajan a través del aire a la velocidad del sonido y la velocidad permanece constante.

Cómo convertir la duración del tiempo en distancia

En la siguiente sección, veremos cómo medir la duración del pulso usando ESP32. Supongamos que hemos medido el pulso de salida en el tiempo (t) con ESP32. Ahora la pregunta es cómo convertir este tiempo medido en distancia. Bueno, esta es la parte más obvia de este tutorial. En la escuela secundaria, todos estudiamos una ecuación de distancia-tiempo bien conocida que es S = vt. Podemos convertir la duración del pulso (t) en distancia (S) usando esta ecuación.

Distance (S) = Speed (v) * t //distance in meters

Aquí v es la velocidad de las ondas ultrasónicas en el aire. La velocidad de las ondas ultrasónicas en el aire es igual a la velocidad del sonido, que es 340 m/s (metros por segundo).

La ecuación anterior dará como resultado la distancia en unidades de metros. Pero, si desea la distancia en unidades de centímetros, multiplique 340 por 100. Por lo tanto, la ecuación anterior se convierte en:

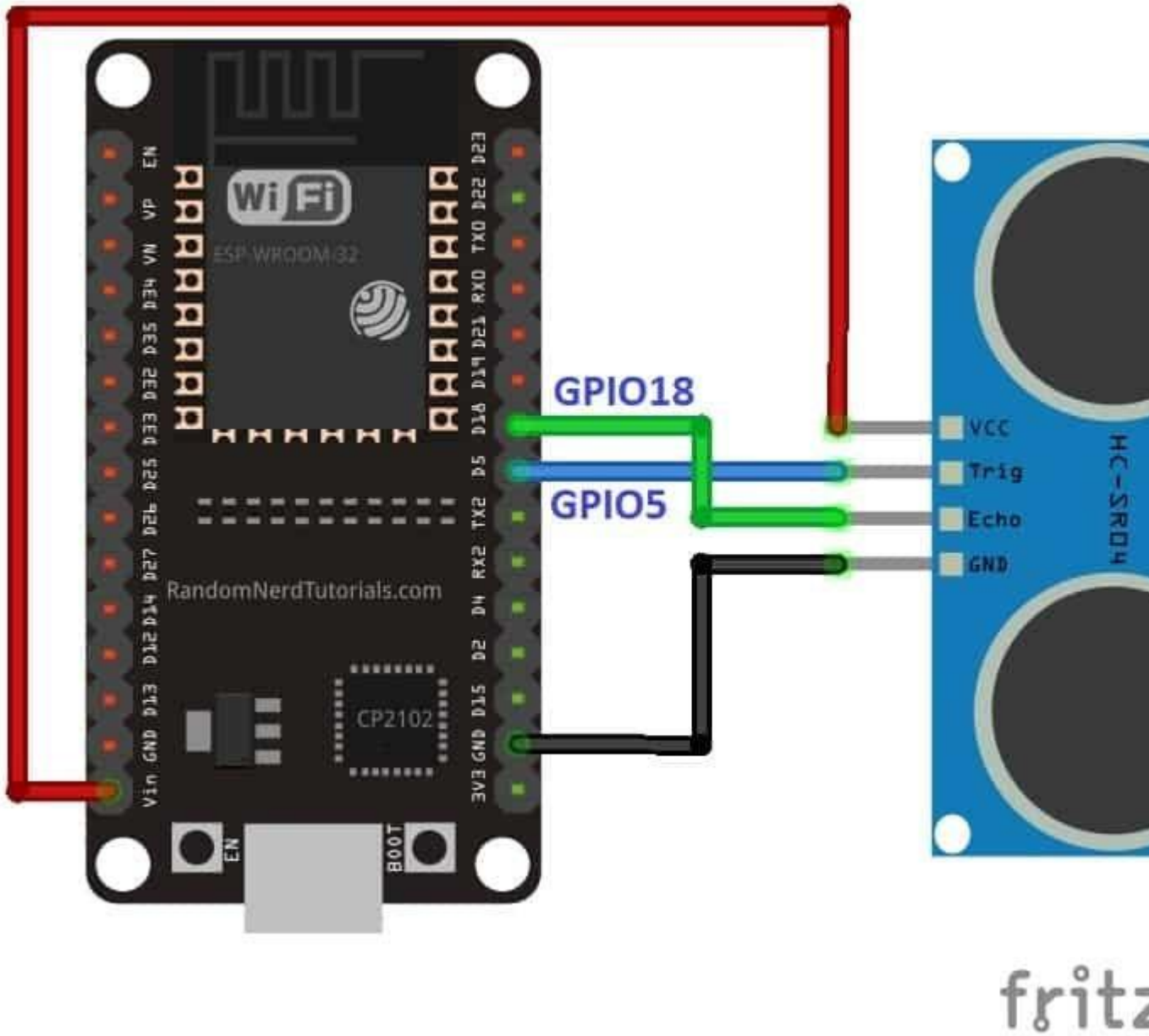
```
S = 34000 * t   // distance in cm
```

El tiempo indicado en la fórmula anterior también debe dividirse por dos, ya que las ondas ultrasónicas viajan desde el transmisor hasta el obstáculo y luego se reflejan de nuevo en el circuito receptor recorriendo la misma distancia. Queremos encontrar la distancia entre HC-SR04 y el objeto únicamente. Por lo tanto, la fórmula para calcular la distancia es:

```
S = 17000 * t    // distance in cm
```

Cómo conectar el sensor ultrasónico HC-SR04 con ESP32

Hasta ahora hemos visto el funcionamiento del sensor ultrasónico y los detalles de los pines. Ahora sabemos que para interconectar un sensor HC-SR04 con ESP32, necesitamos cuatro pines, de los cuales dos son pines de fuente de alimentación y dos son pines de entrada y salida digitales. Un pin GPIO del ESP32 se utilizará como pin de salida digital para proporcionar una señal de activación al sensor ultrasónico. De manera similar, un pin GPIO se utilizará como pin de entrada digital para capturar la señal de salida de eco del sensor de salida.



Ahora realice la conexión del ESP32 con el sensor HC-SR04 según este diagrama de conexión. En este diagrama esquemático, utilizamos el pin GPIO5 del ESP32 para proporcionar una señal de activación y GPIO18 para capturar el pulso de salida de eco.

HC-SR04	ESP32
CCV	Vino
Tierra	Tierra

Desencadenar	GPIO5
Eco	GPIO18

Instalaremos el sensor ultrasónico encima de un recipiente que transportará el agua. De esta manera podremos determinar el nivel del agua obteniendo la distancia (cm) desde el sensor y enviándola al servidor web del monitor de nivel de agua ESP32.
Así es como se ve la configuración:

Quizás te interese leer:

- Sensor ultrasónico MicroPython HC-SR04 con ESP32 y ESP8266
- Sensor ultrasónico HC-SR04 con ESP32: medición de distancia

Configuración del IDE de Arduino

Usaremos Arduino IDE para programar nuestra placa de desarrollo ESP32. Por lo tanto, debes tener la última versión de Arduino IDE. Además, también debes instalar el complemento para la placa que usarás.

Si su IDE no tiene el complemento instalado, puede visitar el siguiente enlace:

- Instalar la biblioteca ESP32 en Arduino IDE y cargar el código.

Servidor web de monitoreo de nivel de agua Arduino Sketch ESP32

Abra su IDE de Arduino y vaya a **Archivo > Nuevo** para abrir un nuevo archivo. Copia el código que se encuentra a continuación en ese archivo. Deberás reemplazar las credenciales de red para poder conectarte correctamente a tu WiFi local.

```
#include <WiFi.h>
#include <WiFiClient.h>
#include <WebServer.h>

int trigger_pin = 5;
int echo_pin   = 18;

// Replace with your network credentials
const char* ssid = "MASTERS";
const char* password = "english123";

WebServer server(80);

String page = "";
int distance_cm;

void setup() {
  Serial.begin(115200);
  pinMode(trigger_pin, OUTPUT);
  pinMode(echo_pin, INPUT);
  delay(1000);

  WiFi.begin(ssid, password);
  Serial.println("");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  server.on("/", []() {
    page = "<head><meta http-equiv=\"refresh\" content=\"3\"></head><center><h1>Ultasonic Water
Level Monitor</h1><h3>Current water level:</h3> <h4>" + String(distance_cm) + "</h4></center>";
    server.send(200, "text/html", page);
  });
  server.begin();
  Serial.println("Web server started!");
}
```

```
void loop() {
  digitalWrite(trigger_pin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigger_pin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigger_pin, LOW);
  long duration = pulseIn(echo_pin, HIGH);
  distance_cm = (duration / 2) / 29.09;
  Serial.println(distance_cm);
  server.handleClient();
  delay(3000);
}
```

¿Cómo funciona el código?

En primer lugar, incluiremos las librerías necesarias. Como tenemos que conectar nuestro ESP32 a una red inalámbrica, necesitamos la librería WiFi.h para ello. La librería WebServer.h será necesaria para construir el servidor web.

```
#include <WiFi.h>
#include <WiFiClient.h>
#include <WebServer.h>
```

A continuación definiremos los nombres de los pines del ESP32 que hemos conectado con los pines Trigger y Echo del sensor. La siguiente línea define que los pines GPIO5 y GPIO18 del ESP32 se utilizan para controlar los pines trigger y echo del sensor HC-SR04 respectivamente.

```
int trigger_pin = 5;
int echo_pin   = 18;
```

A continuación, crearemos dos variables globales, una para el SSID y otra para la contraseña. Estas contendrán nuestras credenciales de red que se utilizarán para conectarnos a nuestro enrutador inalámbrico. Reemplace ambas con sus credenciales para garantizar una conexión exitosa.

```
// Replace with your network credentials
const char* ssid = "YOUR_SSID";
const char* password = "YOUR_PASSWORD";
```

El objeto WebServer se utilizará para configurar el servidor web ESP32. Pasaremos el puerto HTTP predeterminado, que es 80, como entrada al constructor. Este será el puerto en el que el servidor escuchará las solicitudes.

```
WebServer server(80);
```

La variable entera 'distance_cm' contendrá el valor de la distancia en cm.

```
int distance_cm;
```

configuración()

Dentro de la función setup() abriremos la comunicación serial a una velocidad de 115200.

```
Serial.begin(115200);
```

Además, inicialice el trigger_pin como un pin de salida digital y el echo_pin como un pin de entrada digital utilizando la función pinMode().

```
 pinMode(trigger_pin, OUTPUT);
 pinMode(echo_pin, INPUT);
```

La siguiente sección de código conectará nuestra placa ESP32 con la red local cuyas credenciales de red ya especificamos anteriormente. Una vez establecida la conexión, la dirección IP de la placa ESP32 se imprimirá en el monitor serial. Esto nos ayudará a realizar una solicitud al servidor.

```
WiFi.begin(ssid, password);
 Serial.println("");
 while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
 }
 Serial.println("");
 Serial.print("IP address: ");
 Serial.println(WiFi.localIP());
```

Las siguientes líneas crean el servidor web HTML y lo inician. Al usar on() en nuestro objeto de servidor, manejaremos la URL /. La página HTML se creará con tres encabezados que muestran "Monitor de nivel de agua ultrasónico", "Nivel de agua actual" y la distancia en cm. Además, la página web se actualizará automáticamente cada 3 segundos para mostrar la lectura actual.

Además, se utilizará el método send() para devolver la respuesta. Este método acepta tres parámetros. El primer parámetro es el código de respuesta, que especificaremos como 200. Es el código de respuesta HTTP para ok. El segundo parámetro es el tipo de contenido de la respuesta, que especificaremos como “text/html”, y el tercer parámetro es el mensaje real que se nos enviará como respuesta HTTP.

Para iniciar el servidor, llamaremos a begin() en nuestro objeto de servidor.

```
server.on("/", []() {
   page = "<head><meta http-equiv=\"refresh\" content=\"3\"></head><center><h1>Ultasonic Water Level Monitor</h1><h3>Current water level:</h3> <h4>" + String(distance_cm) + "</h4></center>";
   server.send(200, "text/html", page);
 });
 server.begin();
 Serial.println("Web server started!");
```

bucle()

Dentro de la función loop(), primero proporcionaremos un pulso de 10 µs al pin de activación. Esto es para habilitar la medición de distancia de los datos del sensor HC-SR04. Iniciará el proceso de toma de muestras de distancia.

```
 digitalWrite(trigger_pin, LOW);
 delayMicroseconds(2);
 digitalWrite(trigger_pin, HIGH);
 delayMicroseconds(10);
 digitalWrite(trigger_pin, LOW);
```

Tan pronto como aplicamos un pulso de 10 µs al sensor ultrasónico, este produce ondas de sonar de 40 KHz y eleva la señal de salida de eco a un estado activo alto. La señal de salida de eco permanece activa alta hasta que estas ondas de sonar se reflejan de nuevo en el transmisor ultrasónico. Tan pronto como el circuito receptor recibe estas ondas, la señal de salida de eco pasa a un nivel activo bajo.

Medimos el tiempo durante el cual la señal de salida permanece en un estado activo alto utilizando la función pulseIn(). Pasamos el echo_pin como primer parámetro y su estado 'HIGH' como segundo parámetro. Esta vez lo guardaremos en la variable 'duration'.

```
long duration = pulseIn(echo_pin, HIGH);
```

La función pulseIn() se utiliza para medir la duración del pulso y devuelve la duración del tiempo en microsegundos. Pero la fórmula de la relación distancia-tiempo que derivamos anteriormente funciona si tanto la velocidad como el tiempo tienen las mismas unidades en términos de tiempo, como segundos, microsegundos o milisegundos. En esta ecuación, la unidad de velocidad es centímetros por segundo.

S = 17000 * t

Para convertir la velocidad en centímetros por microsegundo, divida esta ecuación por 10^-6. Ahora, las unidades de velocidad y tiempo son compatibles entre sí.

S = 0,017*t

Luego calcularemos la distancia en cm utilizando el cálculo que se indica a continuación. Esto se guardará en la variable 'distancia_cm' que definimos previamente.

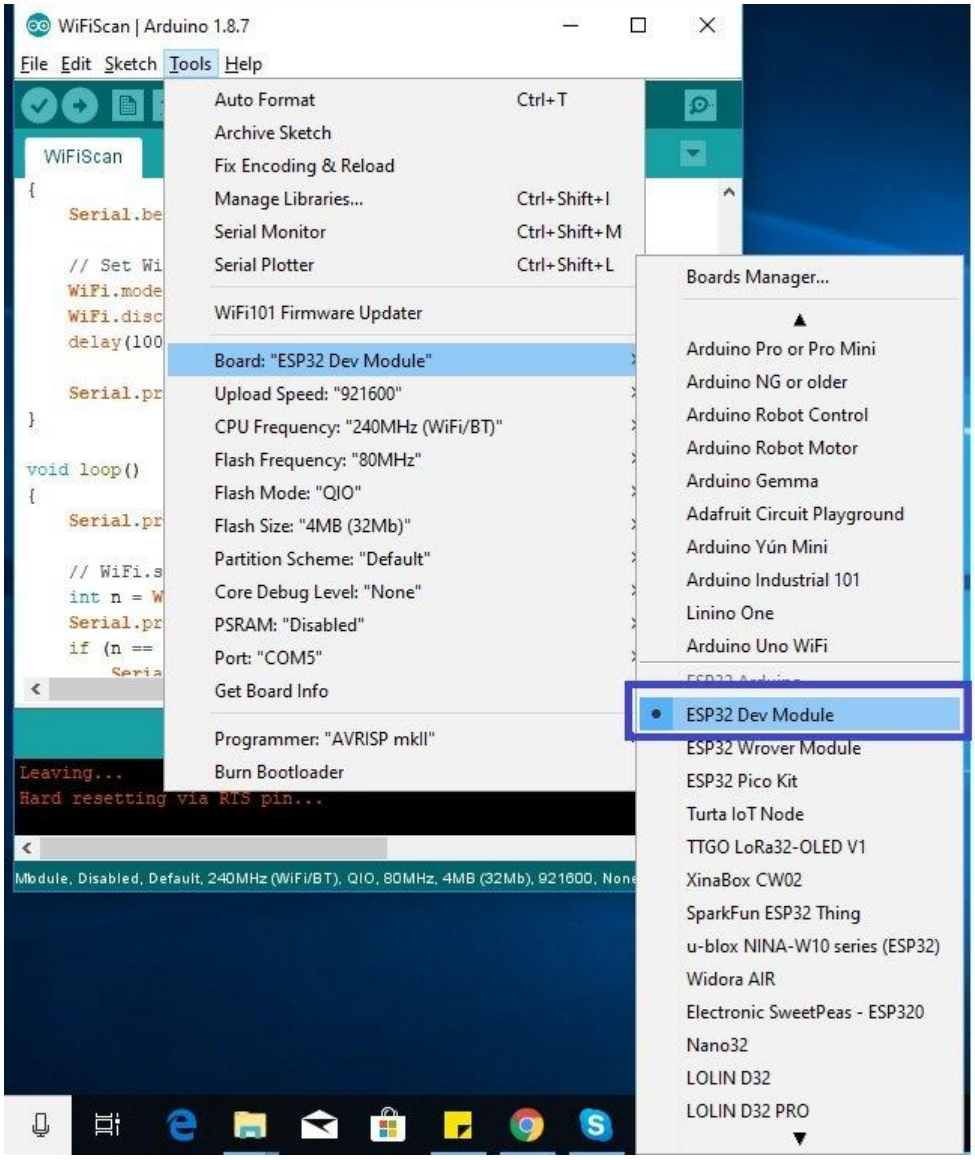
```
 distance_cm = (duration / 2) / 29.09;
```

Este valor se imprime en el monitor serial y también se envía al servidor web ESP32.

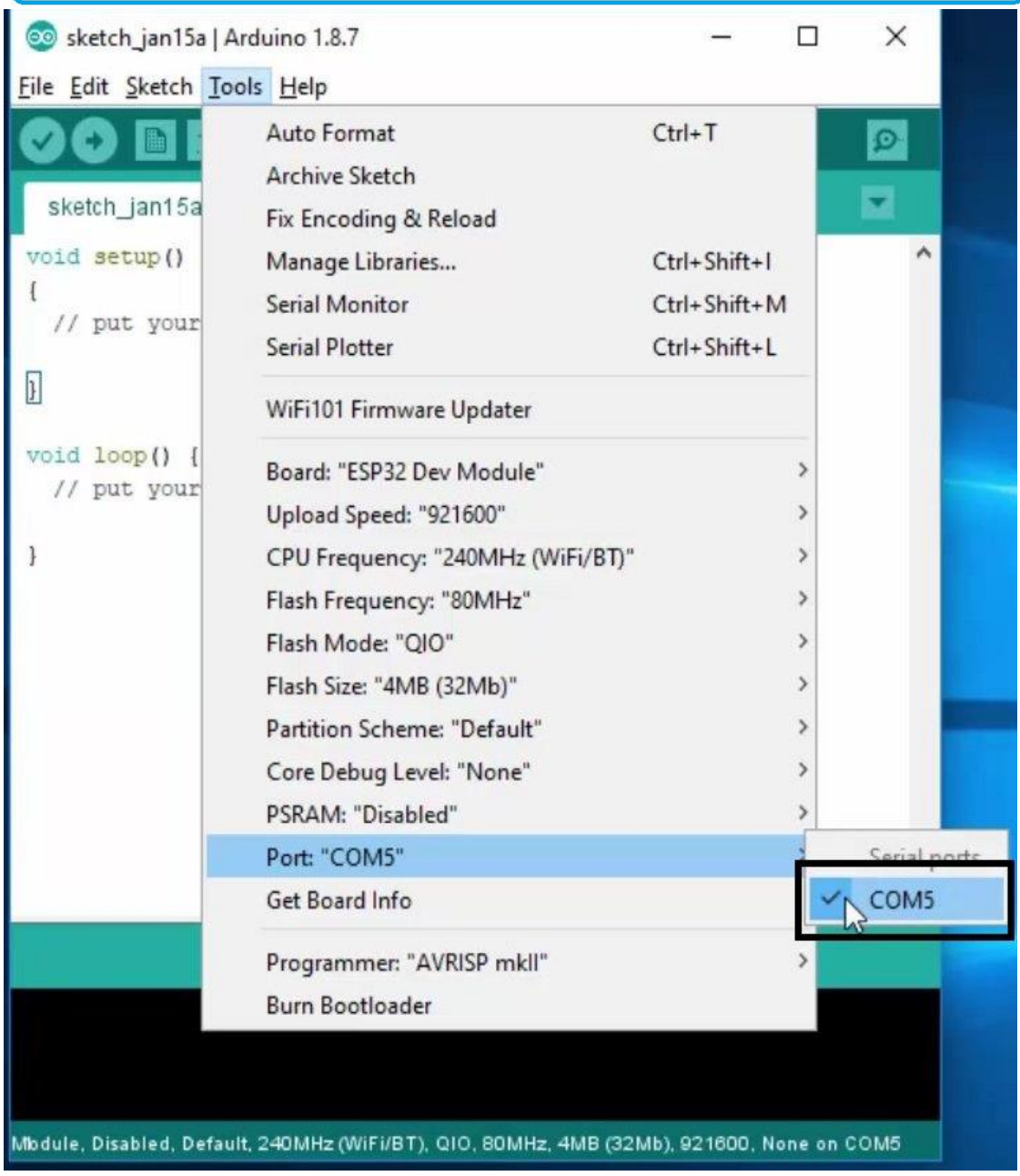
```
 Serial.println(distance_cm);
  server.handleClient();
  delay(3000);
```

Demostración de monitoreo del nivel de agua basado en IoT

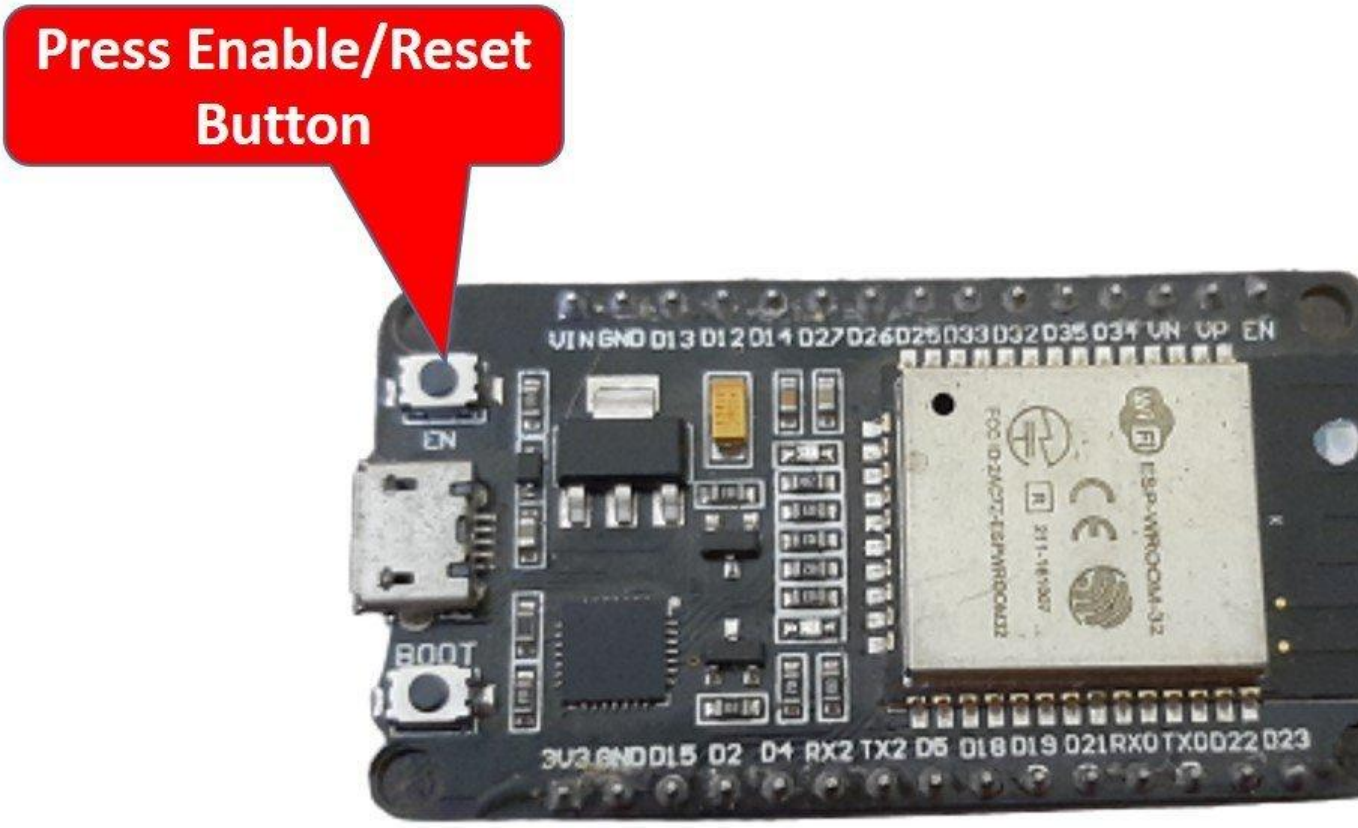
Asegúrate de elegir la placa y el puerto COM correctos antes de cargar el código en la placa. Ve a Herramientas > Placa y selecciona Módulo de desarrollo ESP32.



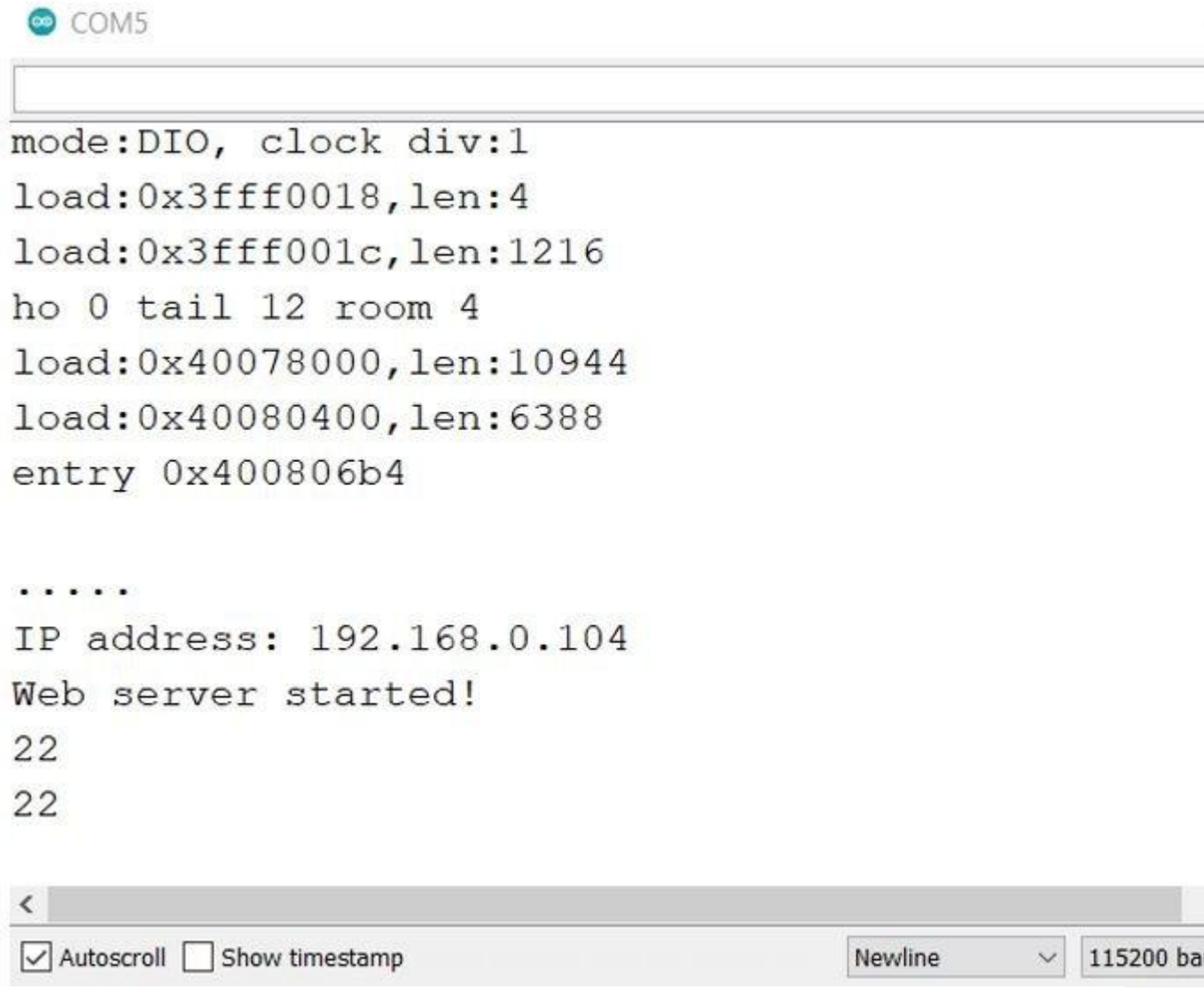
A continuación, vaya a Herramientas > Puerto y seleccione el puerto apropiado a través del cual está conectada su placa.



Haga clic en el botón de carga para cargar el código en la placa de desarrollo ESP32.
Una vez que haya cargado el código en la placa de desarrollo, presione el botón HABILITAR.



En su IDE de Arduino, abra el monitor serial y podrá ver la dirección IP de su módulo ESP32 así como las lecturas de distancia.



Monitor en serie

Escriba esta dirección IP en un nuevo navegador web y presione Enter. Se abrirá la siguiente página web.

Puede ver la lectura actual del nivel de agua. En este momento, es de 23 cm porque el contenedor está vacío y 23 cm es la longitud total del contenedor.
Ahora empieza a verter agua en el recipiente y el valor comenzará a disminuir. En el nivel medio, el valor es de aproximadamente 11 cm.
Asimismo, a medida que aumenta el nivel del agua la distancia (cm) disminuye.

REFERENCIAS BIBLIOGRAFICAS

1. Espressif Systems. (2019). *ESP32 series datasheet*. Espressif Systems. https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
2. SparkFun Electronics. (s.f.). *HC-SR04 ultrasonic distance sensor*. SparkFun. https://www.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf
3. Random Nerd Tutorials. (2018). *ESP32 relay module – control AC appliances*. Random Nerd Tutorials. https://randomnerdtutorials.com/esp32-relay-module-ac-control/
4. Aziz, M. (2020). *Control de bomba de agua con Arduino y sensor de nivel de agua*. Circuit Digest. https://www.circuitdigest.com/microcontroller-projects/automatic-water-level-controller-using-arduino
5. Motorarg S.A. (s.f.). *Manual de instalación y mantenimiento de bombas QC100 Vasser*. Motorarg. (Disponible a través de su sitio web oficial o distribuidor).
6. Monitoreo de nivel de agua sin contacto basado en IoT con ESP32 y HC-SR04

   https://microcontrollerslab.com/iot-contactless-water-level-monitoring-esp32-hc-sr04/