Universidad de Costa Rica
Escuela de Ingeniería Eléctrica
IE-0624 Laboratorio de Microcontroladores

EIE
Escuela de
Ingeniería Eléctrica

Sistema Distribuido de Medición de Nivel y Alerta por Inundaciones usando Arduino y comunicación IoT

UNIVERSIDAD DE
COSTA RICA

Grupo 01

Profesor: MSc. Marco Villalta Fallas

Estudiante: Luis Brenes Campos — **Carné:** C21324
Estudiante: Elizabeth Matamoros — **Carné:** C04652

I SEMESTRE 2025

Índice

1. Introducción **3**

2. Objetivos **4**
2.1. Objetivo General . . . 4
2.2. Objetivos Específicos . . . 4

3. Alcances **4**

4. Justificación **4**

5. Nota teórica **5**
5.1. Información general del Arduino Uno . . . 5
5.2. Sensor ultrasónico HC-SR04 . . . 8
5.3. Información general del ESP32 WROOM-32 . . . 9
5.4. Comunicación IoT . . . 11
5.5. ThingsBoard . . . 11
5.6. Diseño del circuito . . . 11
5.7. Lista de componentes y precios . . . 12

6. Análisis de programa **13**
6.1. Programación del Arduino Uno . . . 13
6.1.1. Definición de pines . . . 13
6.1.2. Parámetros de mapeo . . . 13
6.1.3. Configuración inicial (`setup()`) . . . 13
6.1.4. Bucle principal (`loop()`) . . . 14
6.2. Programación del ESP32 WROOM-32 . . . 15

7. Análisis Electrónico **18**

8. Git y enlace a video de demostración de funcionamiento **20**

9. Conclusiones y recomendaciones **20**
9.1. Conclusiones . . . 20
9.2. Recomendaciones . . . 20

1. Introducción

Este proyecto consiste en el desarrollo de un sistema de monitoreo distribuido para la detección temprana de posibles inundaciones, utilizando dos microcontroladores Arduino Uno con sensores ultrasónicos HC-SR04 y un módulo ESP32 WROOM-32 con conectividad WiFi. El sistema está diseñado para medir el nivel de agua en dos puntos distintos de forma simultánea y activar alarmas locales cuando se supera un umbral crítico.

Por otro lado, cada Arduino actúa como nodo sensor y cuenta con su propio buzzer, que se activa de manera inmediata cuando el nivel de agua detectado es alto. Los datos de ambos nodos son enviados al ESP32, el cual centraliza la información y la transmite a la plataforma ThingsBoard. Allí se encuentra configurado un dashboard con indicadores de nivel y alertas visuales para el monitoreo remoto del sistema.

Es importante recalcar que para la validación funcional, se utilizó una maqueta compuesta por vasos con distintos niveles de agua, lo cual permite simular el comportamiento del sistema ante escenarios de riesgo. Esto facilita la verificación de los sensores, la respuesta del sistema de alarmas y la transmisión correcta de los datos hacia la nube.

Finalmente, el proyecto permitió desarrollar una solución práctica, accesible y replicable para la detección temprana de inundaciones, integrando sensores físicos, microcontroladores y una plataforma IoT. La experiencia evidenció el potencial de este tipo de sistemas para aplicaciones reales, y al mismo tiempo brindó una oportunidad valiosa para fortalecer las habilidades aprendidas en el curso.

2. Objetivos

2.1. Objetivo General

Crear un sistema distribuido de monitoreo de nivel de agua y generación de alertas ante posibles inundaciones, utilizando microcontroladores Arduino Uno interconectados y un ESP32 WROOM-32 para la transmisión de datos a una plataforma IoT, con el fin de simular una red de alerta temprana funcional y accesible.

2.2. Objetivos Específicos

- Diseñar una red distribuida de monitoreo utilizando dos microcontroladores Arduino Uno y un ESP32 WROOM-32, donde los dos arduinos funcionen como nodos sensores de nivel de agua y el ESP32 como nodo central con conectividad WiFi.
- Desarrollar un sistema de comunicación y alerta, que permita la transmisión de datos desde los nodos sensores al nodo central, y de este a una plataforma IoT.
- Simular la detección temprana de condiciones de riesgo por acumulación de agua mediante la medición de niveles en dos puntos distintos, activando alarmas locales y remotas cuando se superen umbrales definidos.

3. Alcances

- El sistema será capaz de medir el nivel de agua en dos puntos distintos mediante sensores ultrasónicos HC-SR04 conectados a microcontroladores Arduino Uno, simulando una red de monitoreo en tiempo real para la detección de niveles críticos.
- Se establecerá una comunicación serial unidireccional entre los nodos sensores y el ESP32, permitiendo la transmisión continua de datos para su análisis centralizado.
- El módulo ESP32 se encargará de recibir los datos de nivel y enviarlos a la plataforma IoT ThingsBoard, donde podrán visualizarse remotamente desde cualquier dispositivo con acceso a Internet.
- Se implementarán mecanismos de alerta local (buzzers) en cada nodo sensor, así como notificaciones remotas configuradas en la plataforma ThingsBoard, que se activarán automáticamente al superar un umbral de seguridad definido.

4. Justificación

Las inundaciones constituyen un riesgo constante en numerosos entornos, donde variaciones súbitas en el nivel de agua pueden generar daños significativos y poner en peligro vidas humanas. Contar con un sistema de monitoreo distribuido capaz de detectar a tiempo aumentos críticos en el caudal y emitir alertas inmediatas resulta esencial para anticipar escenarios de riesgo y facilitar la toma de decisiones preventivas.

Este proyecto implementa una solución práctica que integra sensores ultrasónicos HC-SR04, dos microcontroladores Arduino Uno y un ESP32 con conectividad WiFi para medir niveles de agua en simultáneo, activar alarmas

locales y transmitir datos en tiempo real a la plataforma ThingsBoard. La maqueta experimental, conformada por vasos con diferentes alturas de agua, permite validar el comportamiento del sistema y ajustar umbrales de alerta con precisión.

Desde el punto de vista formativo, el desarrollo de esta arquitectura distribuida refuerza competencias clave del curso, al involucrar lectura de sensores físicos, comunicación serial, programación en C/C++, configuración de módulos WiFi y manejo de dashboards IoT.

Además, la plataforma ThingsBoard utilizada facilita la incorporación de análisis de datos y generación de reportes automáticos, lo que abre la posibilidad de extender el proyecto hacia modelos de predicción de crecidas y patrones de comportamiento hídrico. Esta capacidad de almacenar y procesar historiales de nivel de agua brinda un enfoque más profundo al monitoreo, permitiendo optimizar umbrales de alerta y mejorar la confiabilidad del sistema ante variaciones ambientales.

5. Nota teórica

5.1. Información general del Arduino Uno

El Arduino Uno es una placa de desarrollo de código abierto que integra dos microcontroladores: el ATmega328P para el cómputo principal y el ATmega16U2 como interfaz USB–serie. Esto permite un prototipado rápido y una programación directa desde el Arduino IDE sin hardware adicional.

A continuación se presentan sus características más relevantes:

- **Microcontrolador principal (ATmega328P):**
  - Arquitectura AVR de 8 bits a 16 MHz
  - Memoria Flash: 32 KB (0,5 KB reservados para el bootloader)
  - SRAM: 2 KB
  - EEPROM: 1 KB
  - Periféricos: 2 temporizadores de 8 bits, 1 temporizador de 16 bits, UART, SPI, I^2C/TWI, comparador analógico, Watchdog Timer, Brown-Out Detection
- **Microcontrolador USB–serie (ATmega16U2):**
  - Arquitectura AVR de 8 bits
  - Flash ISP: 16 KB; SRAM: 512 B; EEPROM: 512 B
  - Actúa como puente USB tipo B $\rightarrow$ UART, eliminando convertidores externos
- **Entradas/Salidas:**
  - 14 pines digitales (D0–D13), 6 con PWM
  - 6 entradas analógicas (A0–A5)
  - Pines de alimentación: 5 V, 3,3 V, VIN, GND ($\times$3), AREF

  - Cabeceras ICSP (SPI) de 6 pines

- **Alimentación y condiciones de operación:**
  - Rango lógico: 2,7 V – 5,5 V
  - Alimentación USB: 5 V
  - Entrada externa VIN: 6 V – 20 V (recomendado 7–12 V)
  - Reguladores internos para 5 V y 3,3 V
  - Temperatura de operación: –40 °C a 85 °C
- **Otras características:**
  - Oscilador cerámico de 16 MHz
  - Botón de reset integrado
  - Orificios de montaje (ø 3 mm) y factor de forma estándar Arduino

Capacidades de comunicación

- **UART:** Pines 0 (RX) y 1 (TX)
- **USB–serie:** A través del ATmega16U2
- **I^2C/TWI y SPI:** Interfaces de bus para comunicación con sensores y actuadores

Programación

- Entorno: *Arduino IDE*
- Bootloader en ATmega328P para carga directa de código

Diagrama de bloques y de pines

El diagrama de pines es el siguiente [2]:

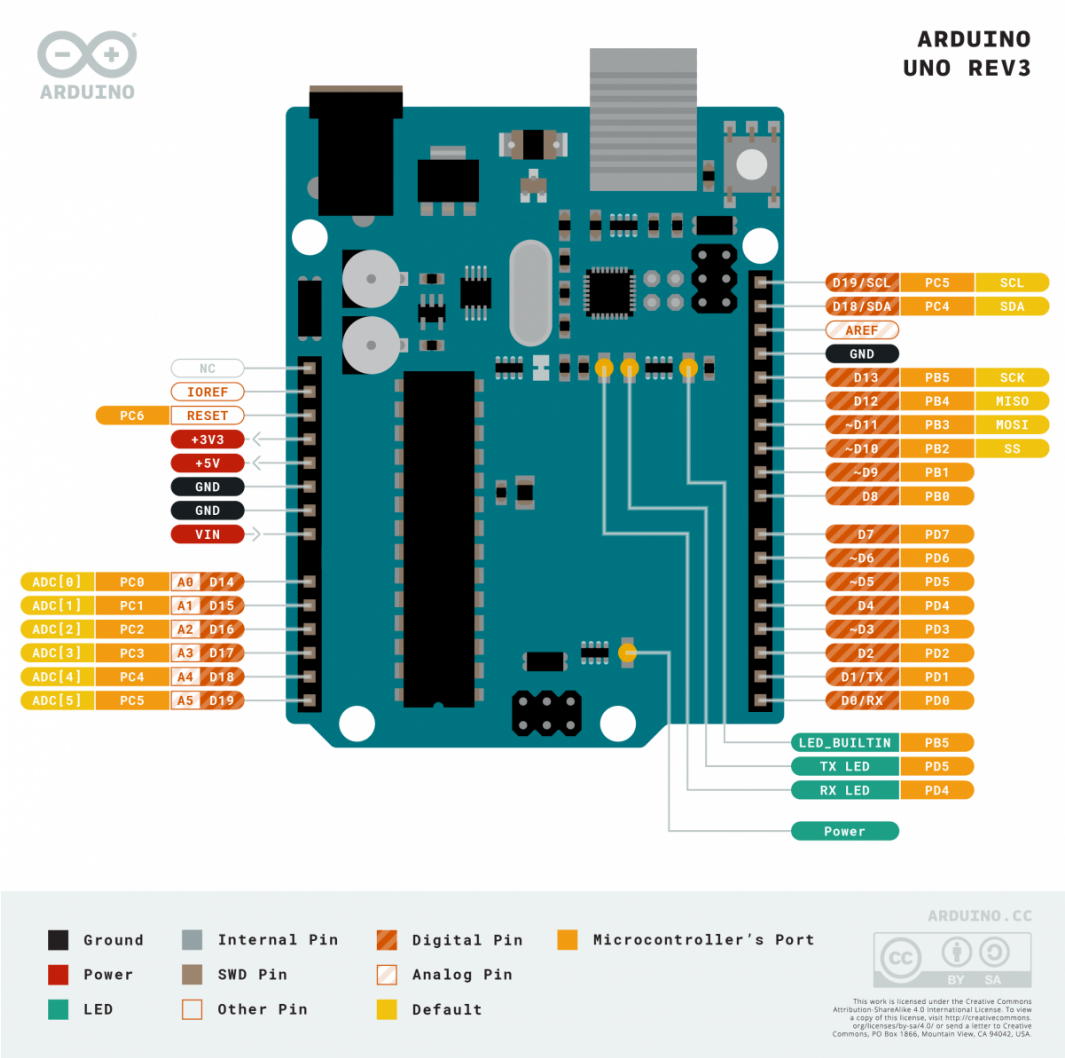


Figura 1: Diagrama de pines de Arduino Uno

El diagrama de bloques (esquemático) es el siguiente. Este se encuentra separado en las 3 partes principales: alimentación, comunicación USB y procesamiento [3]:

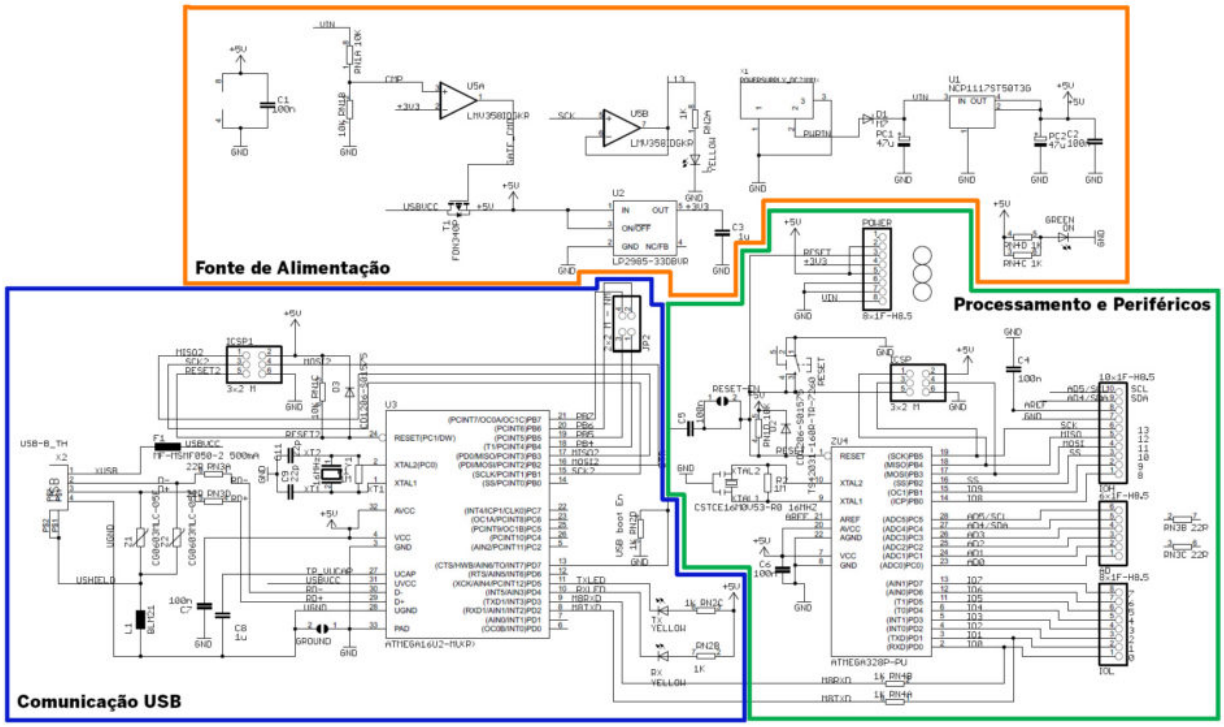


Figura 2: Esquemático de Arduino Uno

5.2. Sensor ultrasónico HC-SR04

El sensor ultrasónico HC-SR04 es un dispositivo utilizado para medir distancias mediante la emisión y recepción de ondas sonoras de alta frecuencia. Su funcionamiento se basa en el principio del tiempo de vuelo: al enviar una señal de control al pin `TRIG`, el sensor emite una ráfaga de ocho pulsos ultrasónicos a 40 kHz. Luego, el pin `ECHO` se activa y permanece en nivel alto mientras espera el rebote de la onda en un objeto. El tiempo que transcurre entre la emisión y la recepción del eco es directamente proporcional a la distancia entre el sensor y el objeto [4]. Dicha distancia se calcula usando la fórmula:

$$d = \frac{t}{2} \times v$$

donde t es el tiempo en segundos que el pin `ECHO` permanece en alto y v es la velocidad del sonido en el aire, estimada en 343 m/s a temperatura ambiente.

El HC-SR04 opera con una tensión de alimentación de 5 V DC y tiene un consumo aproximado de 15 mA durante la medición. El rango de detección efectivo se encuentra entre los 2 cm y los 400 cm, con una resolución de aproximadamente 3 mm. Su ángulo de apertura es de alrededor de 15°, lo que permite una medición relativamente focalizada. El módulo cuenta con cuatro pines: `VCC`, `GND`, `TRIG` y `ECHO`, y su encapsulado es compacto, lo que facilita su integración en maquetas o sistemas embebidos [4].

Es importante recalcar que gracias a su precisión y simplicidad de uso, el sensor HC-SR04 es ampliamente utilizado en diversas aplicaciones.

Figura 3: HC-SR04

5.3. Información general del ESP32 WROOM-32

El ESP32-WROOM-32 es un módulo de comunicación inalámbrica diseñado por Espressif Systems, ampliamente utilizado en aplicaciones de IoT debido a su versatilidad, alto rendimiento y bajo consumo energético. Este módulo integra un microprocesador Xtensa LX6 de doble núcleo a 32 bits, con una frecuencia de operación de hasta 240 MHz, capaz de ejecutar múltiples tareas de forma simultánea, lo cual lo hace ideal para aplicaciones embebidas que requieren procesamiento local y conectividad en tiempo real [5].

En cuanto a su arquitectura interna, el ESP32-WROOM-32 cuenta con 520 KB de SRAM interna, 448 KB de ROM, y puede integrar hasta 4 MB de memoria Flash externa mediante su sistema de empaquetado [5]. Además, incluye periféricos digitales que permiten una amplia gama de aplicaciones:

- Hasta 3 interfaces UART
- 3 SPI, 2 I^2C, 2 I^2S
- Convertidores ADC de 12 bits (hasta 18 canales) y DAC de 8 bits (2 canales)
- Temporizadores, watchdog, sensor de temperatura interno y comparadores analógicos
- Capacidad de uso de hasta 34 GPIOs configurables

A nivel de conectividad, el módulo soporta Wi-Fi 802.11 b/g/n (2.4 GHz) y Bluetooth 4.2 (BR/EDR y BLE), lo que permite tanto comunicación de largo alcance como de baja potencia. Su sistema de antena puede ser externa o integrada mediante una antena PCB, según el diseño del fabricante. En cuanto a la alimentación, opera a 3.3 V y tiene un consumo que varía según el modo de operación: en modo activo puede superar los 200 mA, mientras que en modo deep sleep puede reducirse hasta 10 μA, siendo una opción eficiente para sistemas autónomos con restricciones de energía [5].

En el contexto de este proyecto, el ESP32 cumple la función de nodo central. Recibe datos desde los microcontroladores Arduino Uno que miden el nivel de agua a través de sensores ultrasónicos HC-SR04. Posteriormente, el ESP32 reenvía esa información mediante Wi-Fi a la plataforma ThingsBoard, donde se han configurado dashboards con indicadores visuales y alarmas remotas. Este diseño permite un monitoreo distribuido y remoto del sistema de niveles de agua, así como una respuesta rápida ante condiciones de riesgo.

Diagrama de pines

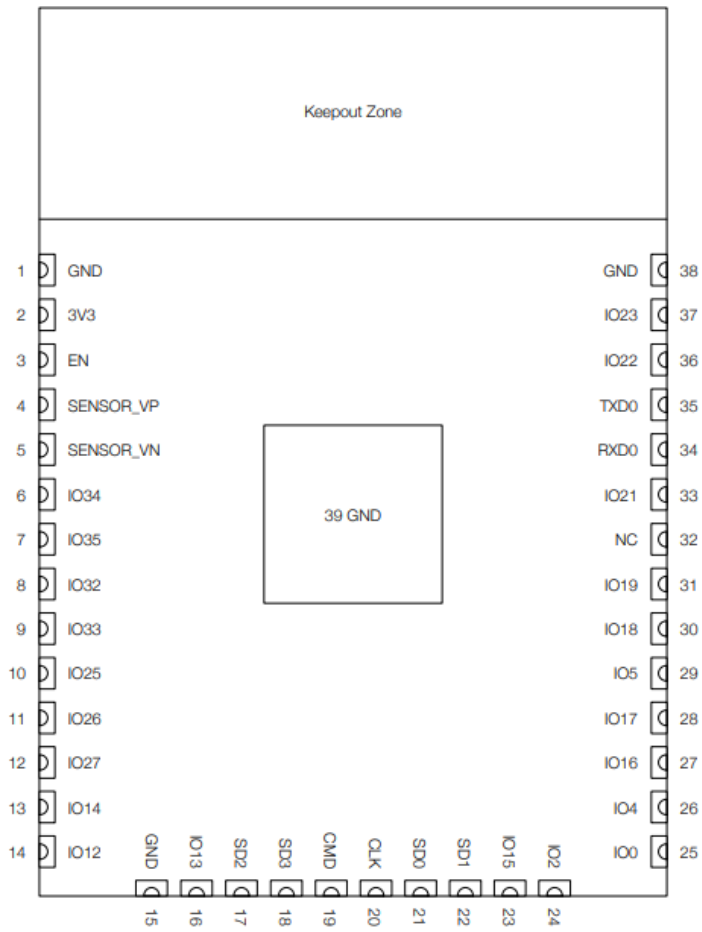


Figura 4: Caption

5.4. Comunicación IoT

La Comunicación IoT (Internet of Things) es el proceso mediante el cual dispositivos físicos (como sensores, actuadores o microcontroladores) intercambian datos a través de una red, normalmente utilizando internet. Esta comunicación permite que los dispositivos recolecten información del entorno (por ejemplo, temperatura, nivel de agua, humedad, etc.) y la envíen a plataformas remotas para ser visualizada, analizada o utilizada para tomar decisiones automáticas [6].

Los protocolos de comunicación más comunes en IoT incluyen HTTP, MQTT, y CoAP. En este proyecto se utilizó HTTP, lo que permite enviar datos en formato JSON a un servidor web o plataforma IoT como ThingsBoard [7].

La comunicación IoT es clave para lograr monitoreo remoto en tiempo real, automatización de procesos, generación de alarmas, y almacenamiento histórico de datos. En contextos educativos o de prototipado, como este laboratorio, permite simular escenarios reales de supervisión y control a distancia.

5.5. ThingsBoard

ThingsBoard es una plataforma de código abierto diseñada específicamente para gestionar soluciones IoT. Sirve como una interfaz intermedia entre los dispositivos físicos y el usuario final, proporcionando herramientas para [8]:

- Recibir datos de sensores a través de protocolos como MQTT o HTTP.
- Almacenar datos en series temporales.
- Crear dashboards interactivos para visualizar esos datos.
- Definir reglas de automatización o alarmas cuando se detectan condiciones críticas.
- Administrar múltiples dispositivos IoT desde una sola plataforma.

En este proyecto, ThingsBoard se utilizó para recibir los datos de nivel de agua enviados por el ESP32, visualizarlos en tiempo real a través de indicadores tipo dial, y activar alarmas cuando los niveles superaban ciertos umbrales predefinidos. Esta plataforma facilita el análisis remoto y la toma de decisiones rápidas, replicando una solución funcional usada en aplicaciones reales de IoT.

5.6. Diseño del circuito

El sistema implementado consta de dos nodos sensores independientes, cada uno basado en una placa Arduino Uno. Cada uno de estos nodos está conectado a un sensor ultrasónico HC-SR04, encargado de medir el nivel de agua. Cuando el nivel detectado supera un umbral crítico, el Arduino activa un buzzer y un LED como alerta local inmediata.

Para cada nodo, el pin `TRIG` del sensor HC-SR04 se conecta a un pin digital de salida del Arduino, y el pin `ECHO` a una entrada digital para la medición del pulso de retorno. El buzzer y el LED se conectan también a pines digitales configurados como salida. Se emplean resistencias limitadoras para el LED, así como divisores de tensión

para adaptar las señales de salida del Arduino (5 V) al nivel de entrada permitido por el ESP32 (3.3 V), previniendo posibles daños por sobrevoltaje.

Cada divisor de tensión se implementa con un potenciómetro de 10 kΩ, ajustado para reducir el voltaje de 5 V a aproximadamente 3.3 V. Este divisor se utiliza tanto para la señal de comunicación serial como para adaptar la alimentación al ESP32 cuando se alimenta el sistema desde una batería externa conectada a uno de los Arduinos.

Finalmente, el ESP32 WROOM-32 actúa como nodo central del sistema. Está conectado a las salidas de los divisores de tensión y se encarga de recibir las lecturas de nivel desde ambos Arduinos mediante comunicación serial (modo unidireccional). Luego, transmite estos datos a la plataforma ThingsBoard mediante Wi-Fi. El ESP32 también está conectado a tierra común con el resto del sistema para asegurar una referencia estable.

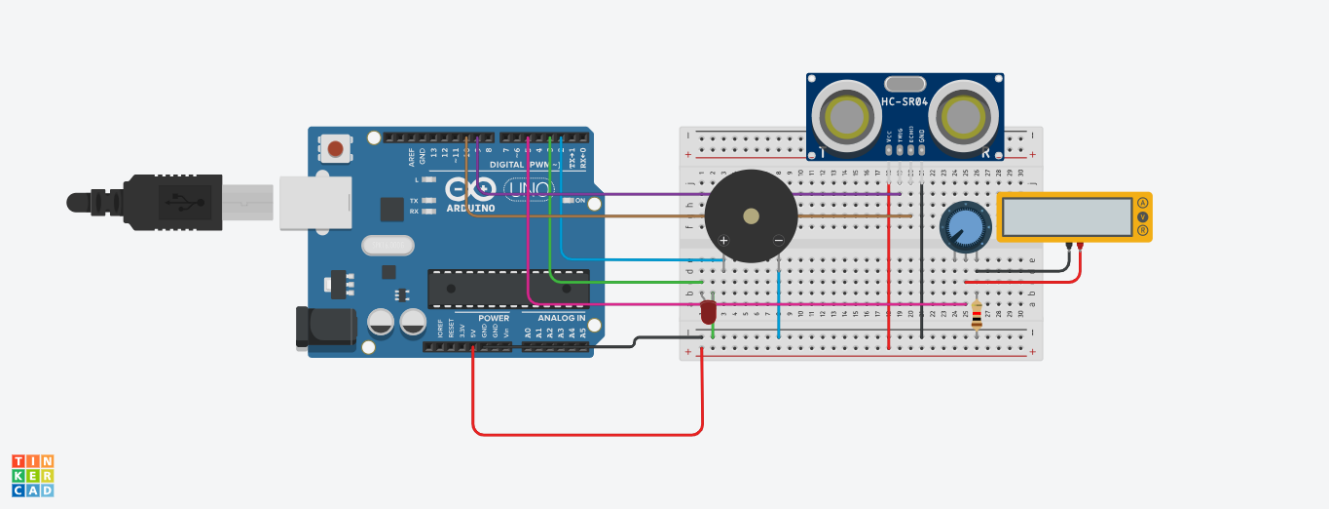


Figura 5: Esquema de conexión para el diseño de circuito propuesto

5.7. Lista de componentes y precios

Tabla 1: Lista de componentes y precios estimados en Costa Rica

Componente	**Descripción**	**Precio unitario (CRC)**
Arduino Uno R3	Placa microcontroladora ATmega328P	₡18 615
Sensor HC-SR04	Módulo ultrasónico 2cm–4m	₡3 002
ESP32-WROOM-32	Módulo Wi-Fi y Bluetooth 3.3V	₡8 500
Buzzer	Piezoeléctrico 5V	₡500
LED rojo 5mm	Diodo emisor de luz	₡100
Potenciómetro 10kΩ	Ajuste manual de voltaje	₡350
Resistencias varias	Kits 220Ω, 1kΩ	₡200

6. Análisis de programa

6.1. Programación del Arduino Uno

6.1.1. Definición de pines

En esta sección se definen los pines del Arduino Uno que se emplearán para el sensor ultrasónico HC-SR04, la señal PWM y los dispositivos de alerta:

```
const int led     = 3;   % LED indicador
const int buzzer  = 2;   % Buzzer activo
const int trigPin = 9;   % Trigger del HC-SR04
const int echoPin = 10;  % Echo del HC-SR04
const int pwmPin  = 5;   % Salida PWM al ESP32
```

- `led (pin 3)`: indicador visual que se enciende en alarma.
- `buzzer (pin 2)`: sirena acústica para alerta.
- `trigPin (pin 9)`: emite el pulso ultrasónico.
- `echoPin (pin 10)`: mide la duración del eco.
- `pwmPin (pin 5)`: salida PWM proporcional a la distancia.

6.1.2. Parámetros de mapeo

```
const float maxDistance = 100.0;  % Distancia máxima considerada (cm)
```

Se fija en 100 cm el valor máximo que se mapeará al rango PWM, para evitar saturaciones cuando el objeto esté muy lejos y aprovechar los bits para distancias pequeñas.

6.1.3. Configuración inicial (`setup()`)

```
void setup() {
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(buzzer, OUTPUT);
  pinMode(led,    OUTPUT);
  pinMode(pwmPin, OUTPUT);
  digitalWrite(buzzer, LOW);
  digitalWrite(led,    LOW);
  analogWrite(pwmPin,  0);
}
```

- `Serial.begin(9600)`: inicializa el puerto serie para depuración.

- `pinMode(...)`: configura cada pin como entrada o salida.
- `digitalWrite/analogWrite iniciales`: aseguran que buzzer y LED estén apagados y PWM en 0 %.

6.1.4. Bucle principal (`loop()`)

El bucle repite el siguiente flujo cada 300 ms:

1. **Disparo del ultrasonido:**

```
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(15);
digitalWrite(trigPin, LOW);
```

   Se genera un pulso de 15 µs para que el HC-SR04 emita la onda ultrasónica.

2. **Medición del eco:**

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

   Mide en microsegundos el tiempo de ida y vuelta del pulso reflejado.

3. **Cálculo de distancia:**

$$\text{distance} = \frac{\text{duration} \times 0.034}{2} \quad [\text{cm}]$$

   Con 0.034 cm/µs (velocidad del sonido) y división por 2 para distancia de ida.

4. **Envío por Serial:**

```
Serial.print("Distancia: ");
Serial.print(distance);
Serial.println(" cm");
```

5. **Lógica de alarma:**

```
if (distance <= 3) {
  digitalWrite(buzzer, HIGH);
  digitalWrite(led,    HIGH);
} else {
  digitalWrite(buzzer, LOW);
  digitalWrite(led,    LOW);
}
```

Se activa LED y buzzer si la distancia es $\leq 3\,\text{cm}$.

6. **Mapeo a PWM proporcional:**

```
float d = constrain(distance, 0, maxDistance);
int duty = map(int(d), 0, int(maxDistance), 0, 255);
analogWrite(pwmPin, duty);
```

- `constrain()`: recorta entre 0 y `maxDistance`.
- `map()`: escala lineal a $[0, 255]$ para `analogWrite()`.

7. **Pausa:**

```
delay(300);
```

Espera 300 ms antes de la siguiente medición.

6.2. Programación del ESP32 WROOM-32

El módulo ESP32-WROOM-32 actúa como nodo central del sistema, encargado de recibir las mediciones de nivel de agua desde los sensores conectados a los Arduinos Uno, calcular su valor equivalente en centímetros mediante el análisis de la señal PWM recibida, y transmitir estos datos a una plataforma IoT para su visualización y análisis remoto.

El programa implementado se basa en el uso de la librería `WiFi.h` para la conexión inalámbrica y `HTTPClient.h` para la transmisión de datos mediante el protocolo HTTP POST. En la fase de inicialización (`setup()`), el ESP32 se conecta a la red Wi-Fi configurada con el SSID y contraseña del entorno de pruebas. Una vez establecida la conexión, el sistema entra en un ciclo continuo de adquisición y envío de datos.

La lectura de los niveles se realiza a partir de las señales PWM provenientes de los Arduinos. Estas señales se reciben en los pines GPIO14 y GPIO27 del ESP32. Se mide el tiempo en alto y en bajo del ciclo PWM con la función `pulseIn()`, y a partir de ello se calcula el ciclo de trabajo. Multiplicando este valor por la distancia máxima esperada (100 cm), se obtiene una estimación del nivel de agua en cada punto:

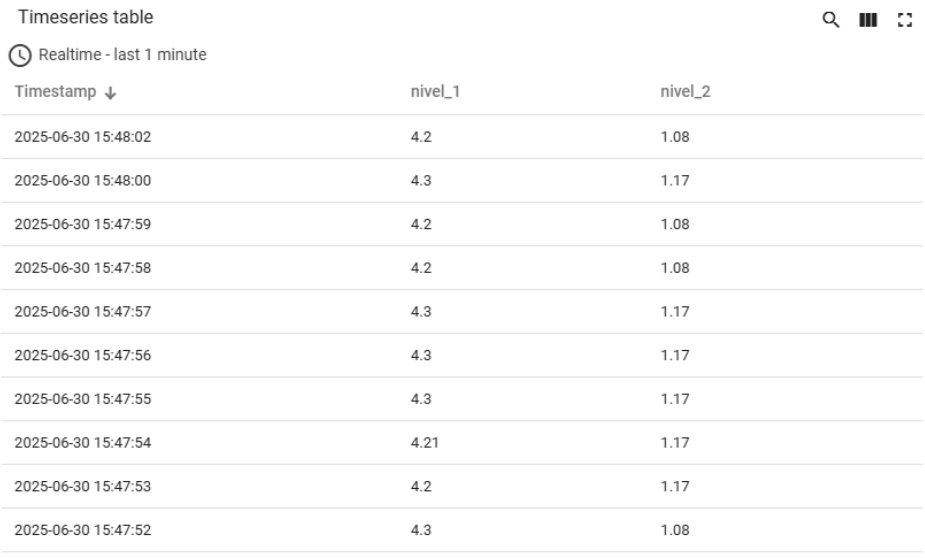
$$\text{distancia} = \left(\frac{t_{alto}}{t_{alto} + t_{bajo}} \right) \times 100 \text{ cm} \tag{1}$$

Posteriormente, estos datos se formatean en una cadena JSON con las claves `"nivel_1"` y `"nivel_2"`, y se envían a ThingsBoard mediante una solicitud POST HTTP al endpoint correspondiente, usando el token de acceso proporcionado por la plataforma.

El siguiente fragmento muestra el formato del cuerpo JSON enviado:

```
{
  "nivel_1": 4.21,
  "nivel_2": 1.17
}
```

ThingsBoard recibe esta información y la presenta en un tablero interactivo que incluye indicadores tipo dial y una tabla de series temporales Figura 6. Dentro de la plataforma se configuraron reglas de alarma para que, al superar o descender por debajo de ciertos valores umbral, se generen alertas en estado `critical`. Estas alarmas se visualizan en tiempo real en el panel de eventos, lo cual facilita la detección inmediata de condiciones de riesgo Figura7.



Timeseries table

Realtime - last 1 minute

Timestamp ↓	nivel_1	nivel_2
2025-06-30 15:48:02	4.2	1.08
2025-06-30 15:48:00	4.3	1.17
2025-06-30 15:47:59	4.2	1.08
2025-06-30 15:47:58	4.2	1.08
2025-06-30 15:47:57	4.3	1.17
2025-06-30 15:47:56	4.3	1.17
2025-06-30 15:47:55	4.3	1.17
2025-06-30 15:47:54	4.21	1.17
2025-06-30 15:47:53	4.2	1.17
2025-06-30 15:47:52	4.3	1.08

Figura 6: Recepción de datos en ThingsBoard desde el ESP32

Figura 7: Dashboard configurado en ThingsBoard con visualización y alarmas

Este enfoque permite el monitoreo remoto en tiempo real, la generación automática de alarmas ante condiciones críticas y el almacenamiento histórico de datos para su análisis posterior. Además, la implementación de comunicación IoT mediante el protocolo HTTP y el uso de una plataforma como ThingsBoard refuerzan los conocimientos adquiridos sobre integración de sistemas embebidos con servicios en la nube aprendidos en el curso.

7. Análisis Electrónico

El sistema propuesto está compuesto por dos nodos sensores basados en microcontroladores Arduino Uno y un nodo central conformado por un módulo ESP32 WROOM-32. Cada Arduino se conecta a un sensor ultrasónico HC-SR04 para medir niveles de agua de manera independiente, activando una alerta local (buzzer y LED) si el nivel supera un umbral crítico. Los datos recogidos por los sensores son representados como señales PWM, las cuales son enviadas al ESP32 para su posterior transmisión a la nube a través de comunicación IoT. La Figura8 muestra el montaje físico completo del sistema.

En primer lugar, para la configuración general del sistema, cada nodo sensor cuenta con un Arduino UNO conectado a un sensor HC-SR04, un LED y un buzzer. Ambos Arduinos operan de forma autónoma, alimentados por cable USB o batería externa. El nodo central ESP32 se encarga de recibir las señales PWM desde los Arduinos y transmitir los valores de nivel procesados hacia la plataforma ThingsBoard mediante comunicación HTTP.

Ahora bien, para el subsistema de sensado, los sensores HC-SR04 miden la distancia entre el emisor ultrasónico y la superficie del agua. Internamente, estos sensores generan una señal PWM proporcional a la distancia detectada, que se transmite a través de un pin digital. En este proyecto, dichas señales PWM son capturadas por el ESP32 a través de los pines GPIO14 y GPIO27, donde se calcula el ciclo útil para estimar el nivel de agua correspondiente a cada sensor.

Por otro lado, para el subsistema de alerta local cuando el nivel medido por cada sensor supera un umbral predefinido, el Arduino activa de forma inmediata una alerta compuesta por un buzzer y un LED rojo. Este mecanismo permite alertar de forma local sin depender de la conexión a Internet, proporcionando una capa adicional de seguridad.

En relación con el acondicionamiento de señal para el ESP32, puesto que los pines digitales del Arduino operan a 5V y los del ESP32 a 3.3V, se implementaron divisores de tensión utilizando potenciómetros de 10kΩ para adaptar las señales PWM antes de ingresarlas al ESP32. Este paso es fundamental para evitar daños en los pines del microcontrolador, ya que el ESP32 no es tolerante a niveles superiores a 3.3V.

Además, la comunicación entre los nodos se realiza de forma unidireccional. Los Arduinos únicamente envían señales PWM al ESP32, sin recibir datos de vuelta. Esta simplicidad evita conflictos por sincronización o direccionamiento, y es suficiente para la función principal de transmisión de datos de nivel hacia la nube.

Finalmente, el montaje físico del sistema se realizó sobre una protoboard. La Figura8 muestra la disposición de los componentes: dos Arduinos UNO, un módulo ESP32, sensores HC-SR04, dos buzzers, dos LEDs y potenciómetros para los divisores de tensión. La alimentación se realiza mediante conexión USB y batería, y el cableado fue organizado para facilitar pruebas funcionales y garantizar conexiones estables durante la simulación del sistema.

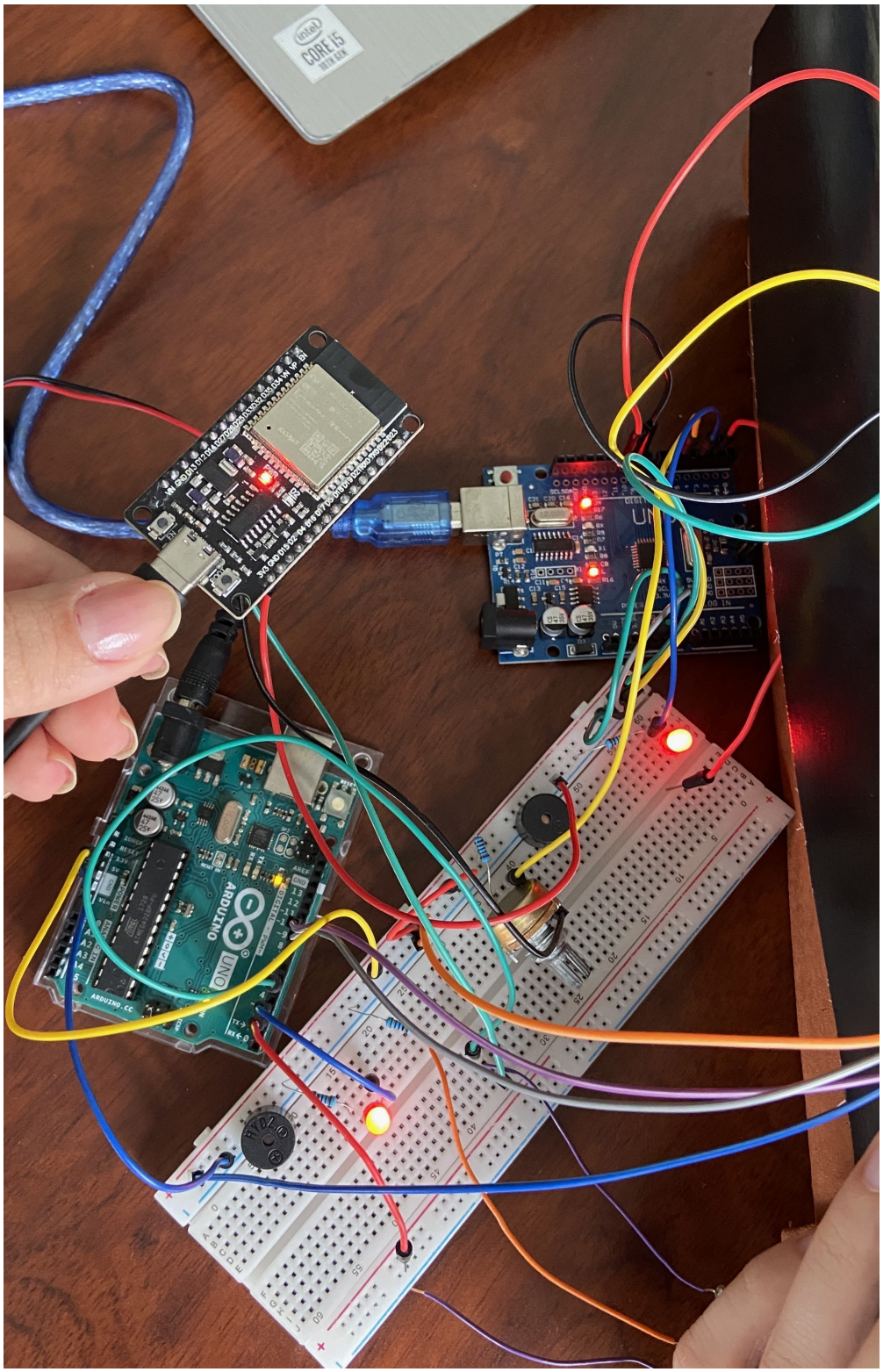


Figura 8: Montaje físico del sistema con dos Arduinos, ESP32 y periféricos

8. Git y enlace a video de demostración de funcionamiento

El desarrollo completo del laboratorio, que incluye el código fuente, las simulaciones y la documentación del laboratorio, se encuentra disponible en el repositorio oficial del curso en Git UCR, en la carpeta correspondiente a Proyecto `https://git.ucr.ac.cr/laboratorio_microcontroladores_i2025/proyecto.git`.

Además, en el siguiente enlace se encuentra un video que comprueba el funcionamiento del proyecto `https://youtu.be/Si661SCKUW8?si=7dmRznF2M2Ja89yW`.

9. Conclusiones y recomendaciones

9.1. Conclusiones

- Se logró que los módulos Arduino Uno adquirieran de manera fiable los datos de los sensores ultrasonidos HC-SR04 y los enviaran al ESP32 mediante señales PWM.
- El ESP32, actuando como microcontrolador principal, recibe correctamente las señales PWM de cada Arduino y las transmite exitosamente por Wi-Fi a ThingsBoard.
- Se diseñaron y configuraron de forma satisfactoria los dashboards en ThingsBoard, incluyendo indicadores y alarmas visuales cuando los niveles de agua superan los umbrales críticos.

9.2. Recomendaciones

- Para un prototipo más representativo, aplicar el sistema en un entorno real con una fuente de agua (por ejemplo, un río) y validar su comportamiento bajo variaciones naturales de nivel.
- Implementar conectividad Wi-Fi directa en cada nodo Arduino, de modo que todos puedan transmitir datos autonomamente al servidor principal sin depender de un ESP32 intermedio.
- Incorporar sensores climáticos (temperatura, humedad, velocidad del viento) que permitan correlacionar las variaciones del nivel de agua con condiciones meteorológicas.

Referencias

[1] STMicroelectronics, "LSM9DS1: iNEMO inertial module: 3D accelerometer, 3D gyroscope, 3D magnetometer," *Datasheet*, rev. 2, July 2015. Disponible en: https://www.st.com/resource/en/datasheet/lsm9ds1.pdf

[2] González, Ó. *Guía de modelos Arduino y sus características: Arduino UNO*. BricoGeek Lab. Disponible en: https://lab.bricogeek.com/tutorial/guia-de-modelos-arduino-y-sus-caracteristicas/arduino-uno

[3] Teix, R. *Conhecendo o Esquemático do Arduino Uno – Parte 1*. Ricardo Teix. Disponible en: https://www.ricardoteix.com/conhecendo-o-esquematico-do-arduino-uno-parte-1/

[4] HC-SR04, *Ultrasonic Sensor HC-SR04 Datasheet*, [Online]. Available: https://components101.com/sensors/hc-sr04-ultrasonic-sensor [Accessed: 29-Jun-2025].

[5] Espressif Systems, *ESP32-WROOM-32 Datasheet*, Version 3.9, Feb. 2020. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

[6] Wikipedia, "Internet of Things," *Wikipedia, la enciclopedia libre*, Disponible en: https://en.wikipedia.org/wiki/Internet_of_things. [Consultado: junio 2025].

[7] G. P. Naik y U. A. Bapat, "Application Layer Protocols of Internet of Things: MQTT, CoAP, AMQP and HTTP," *International Journal of Computer Science and Mobile Computing (IJCSMC)*, vol. 9, no. 9, pp. 70–76, 2020.

[8] ThingsBoard, "ThingsBoard Community Edition Documentation," Disponible en: https://thingsboard.io/docs/. [Consultado: junio 2025].