

# Medidor de Nivel de Agua con ESP32-S3 y Escalamiento a Capa de Transporte

**Integrantes:** Rodríguez Matías

Rodríguez Facundo

Córdoba Diego

Lanfranco Carolina

Lanfranco Julia

**Fecha:** 17/09/2025

**Materia:** Aproximación al mundo del trabajo

**Docente:** Mainero Alejandro

## Contenido

<b>Introducción .....</b>	3
<b>Objetivo del proyecto .....</b>	3
<b>Breve descripción del sistema .....</b>	3
<b>Marco teórico .....</b>	4
<b>Sensor HC-SR04 y principio de medición .....</b>	4
<b>ESP32-S3 y ventajas .....</b>	5
<b>Protocolo MQTT y capa de transporte .....</b>	5
<b>LCD HD44780 vía I2C.....</b>	5
<b>Base de datos MySQL como capa de almacenamiento .....</b>	6
<b>Desarrollo del sistema .....</b>	6
<b>Diagrama de conexión .....</b>	6
<b>Código explicado por secciones.....</b>	7
<b>1. Conexión WiFi .....</b>	7
<b>2. Medición de nivel.....</b>	7
<b>3. Publicación MQTT .....</b>	8
<b>4. Recepción de comandos .....</b>	8
<b>Justificación de pines, componentes y protocolos .....</b>	8
<b>Capturas del LCD y del broker MQTT.....</b>	9
<b>Explicación de la base de datos .....</b>	10
<b>Resultados .....</b>	11
<b>Conclusiones .....</b>	13
<b>Qué aprendimos.....</b>	13
<b>Qué desafíos enfrentamos.....</b>	13
<b>Qué mejoraríamos o escalaríamos.....</b>	13

# Introducción

El acceso y control eficiente del agua es un desafío creciente en la actualidad. Los sistemas IoT permiten desarrollar soluciones inteligentes para monitoreo y automatización de recursos. En este proyecto se presenta el desarrollo de un sistema de medición de nivel de agua en un tanque, escalado desde la capa física hasta la capa de transporte y almacenamiento de datos.

El repositorio completo del proyecto, incluyendo el código fuente, diagramas y documentación, está disponible en:

[EQUIPO-LANFRANCO-RODRIGUEZ-CORDOBA/proyecto-medidor-agua-tanque: proyecto de medición de nivel de agua de un tanque](#)

Puede encontrar el **material audiovisual** explicativo haciendo [click aquí](#).

## Objetivo del proyecto

El objetivo principal de este proyecto es desarrollar un sistema de monitoreo de nivel de agua en un tanque utilizando un microcontrolador ESP32-S3, un sensor ultrasónico HC-SR04 y una pantalla LCD I2C. El sistema busca escalar desde la capa física hasta la capa de transporte, integrando comunicación inalámbrica mediante el protocolo MQTT y almacenamiento de datos en una base de datos MySQL. Además, se implementa una alerta visual por nivel bajo y control remoto de iluminación mediante mensajes MQTT.

## Breve descripción del sistema

El sistema se compone de un ESP32-S3 que mide la distancia entre el sensor ultrasónico y el pelo de agua, calculando así el nivel del tanque. Esta información se muestra en tiempo real en una pantalla LCD 16x2 conectada por I2C. Si el nivel cae por debajo de 100 cm, se activa una alerta luminosa mediante un LED rojo.

El ESP32-S3 se conecta a una red WiFi simulada en Wokwi y publica el nivel de agua en un tópico MQTT (tanque/nivel) utilizando el broker público test.mosquitto.org. Además, está suscripto al tópico tanque/luz, desde el cual puede recibir comandos para encender o apagar un LED verde de forma remota.

En paralelo, un script en Python ejecutado en Visual Studio Code escucha los mensajes publicados por el ESP32-S3 y almacena los valores en una base de datos MySQL, permitiendo el registro histórico de las mediciones. Esta arquitectura modular permite escalar el sistema hacia aplicaciones más complejas como control de bombas, monitoreo remoto o integración con plataformas IoT.

## Marco teórico

### Sensor HC-SR04 y principio de medición

El sensor ultrasónico HC-SR04 permite medir distancias mediante el principio de eco. Emite un pulso ultrasónico a través del pin TRIG y mide el tiempo que tarda en recibir el rebote en el pin ECHO. Este tiempo se convierte en distancia utilizando la fórmula:

$$\text{distancia (cm)} = \text{tiempo (\mu s)} / 58$$

En este proyecto, el sensor se ubica a 400 cm sobre el fondo del tanque, por lo que el nivel de agua se calcula como:

$$\text{nivel} = 400 - \text{distancia}$$

Este tipo de sensor es económico, preciso en rangos cortos y fácil de integrar con microcontroladores como el ESP32.

## ESP32-S3 y ventajas

El ESP32-S3 es un microcontrolador de alto rendimiento con conectividad WiFi y Bluetooth, múltiples pines GPIO y soporte para programación en MicroPython. Sus ventajas incluyen:

- Procesamiento rápido y eficiente
- Conectividad inalámbrica integrada
- Compatibilidad con sensores, actuadores y pantallas
- Ideal para proyectos IoT escalables

En este sistema, el ESP32-S3 se encarga de medir el nivel, mostrarlo en pantalla, publicar los datos por MQTT y recibir comandos remotos.

## Protocolo MQTT y capa de transporte

MQTT (Message Queuing Telemetry Transport) es un protocolo ligero de mensajería diseñado para comunicaciones máquina a máquina (M2M) en entornos con recursos limitados. Funciona sobre TCP/IP y se basa en un modelo publicador/suscriptor:

- El ESP32 publica el nivel en el tópico tanque/nivel
- El script en Python se suscribe y guarda los datos en la base
- El ESP32 también se suscribe al tópico tanque/luz para recibir comandos

Esta capa de transporte permite escalar el sistema hacia monitoreo remoto, control distribuido y aplicaciones IoT más complejas.

## LCD HD44780 vía I2C

La pantalla LCD 16x2 con controlador HD44780 permite mostrar información en tiempo real. Se conecta mediante el chip PCF8574 que traduce señales paralelas a I2C, reduciendo el uso de pines.

En este proyecto, se utiliza una clase modular (LcdApi + I2cLcd) que abstrae los comandos del controlador y permite mostrar el nivel de agua con claridad. También se implementa una alerta visual mediante LEDs.

## Base de datos MySQL como capa de almacenamiento

MySQL es un sistema de gestión de bases de datos relacional que permite almacenar, consultar y analizar datos estructurados. En este proyecto, se utiliza para:

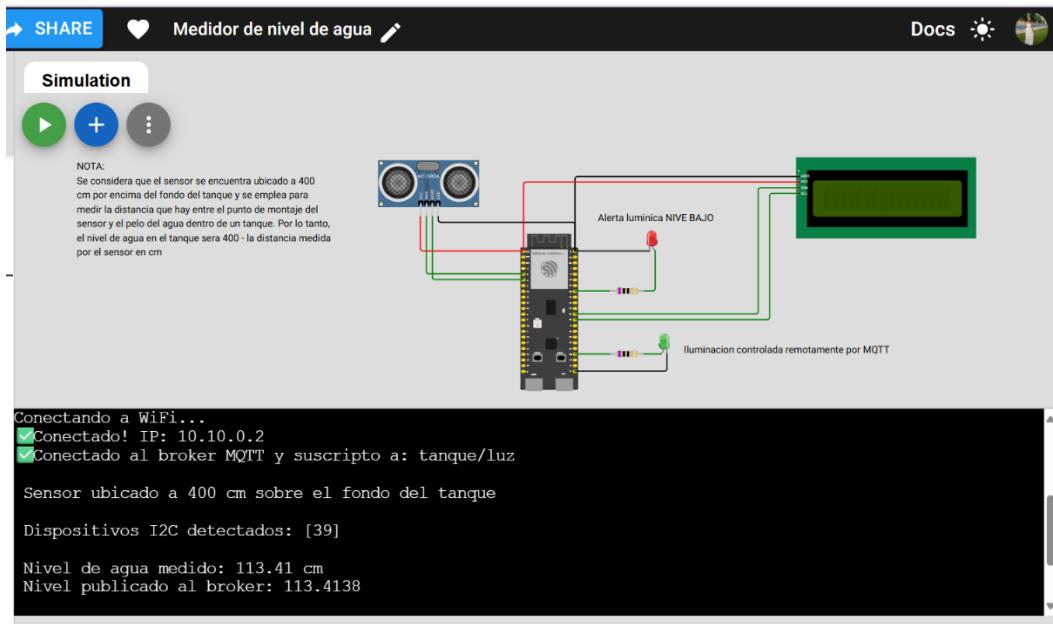
- Registrar cada medición de nivel publicada por el ESP32
- Asociar cada dato con una marca de tiempo
- Permitir trazabilidad y análisis histórico

La conexión se realiza desde un script en Python usando mysql-connector-python, y la inserción se gestiona mediante una función modular (crear\_tabla\_e\_insertar) que garantiza la persistencia de los datos.

## Desarrollo del sistema

### Diagrama de conexión

El circuito fue diseñado y simulado en Wokwi. Se utilizó un ESP32-S3, un sensor ultrasónico HC-SR04, una pantalla LCD 16x2 con interfaz I2C, y dos LEDs para señalización. El LED rojo indica nivel bajo y el LED verde puede ser controlado remotamente por MQTT. La conexión se realizó respetando los pines recomendados por la documentación de cada componente.



Simulación del sistema de medición de nivel de agua en wokwi

## Código explicado por secciones

### 1. Conexión WiFi

```
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(ssid, password)
```

Se establece la conexión a la red WiFi simulada en Wokwi para habilitar la comunicación MQTT.

### 2. Medición de nivel

```
def medir():
    disparo.value(1)
    utime.sleep_us(10)
    disparo.value(0)
    tiempo_eco = time_pulse_us(eco, 1, 23200)
    distancia_cm = tiempo_eco / 58
```

```
    return distancia_cm
```

Se mide la distancia entre el sensor y el pelo de agua. El nivel se calcula como 400 - distancia.

### 3. Publicación MQTT

```
client.publish(TOPIC_PUB, f"{nivel:.2f}")
```

El ESP32 publica el nivel en el tópico tanque/nivel usando el broker público test.mosquitto.org.

### 4. Recepción de comandos

```
python
def mensaje_llegado(topic, msg):
    if msg.decode() == "prender":
        led_verde.value(1)
```

El ESP32 se suscribe al tópico tanque/luz y enciende o apaga el LED verde según el mensaje recibido.

## Justificación de pines, componentes y protocolos

Se eligió el ESP32-S3 por su capacidad de procesamiento y conectividad WiFi. El sensor HC-SR04 se conectó a los pines GPIO 5 (TRIG) y 6 (ECHO). La pantalla LCD se conectó por I2C a los pines GPIO 35 (SCL) y 36 (SDA), utilizando el chip PCF8574. Los LEDs se conectaron a los pines GPIO 40 (rojo) y GPIO 20 (verde). Se utilizó MQTT como protocolo de transporte por su ligereza y eficiencia en sistemas embebidos.

## Capturas del LCD y del broker MQTT



Pantalla LCD mostrando el nivel medido por el sensor ultrásónico HC-SR04

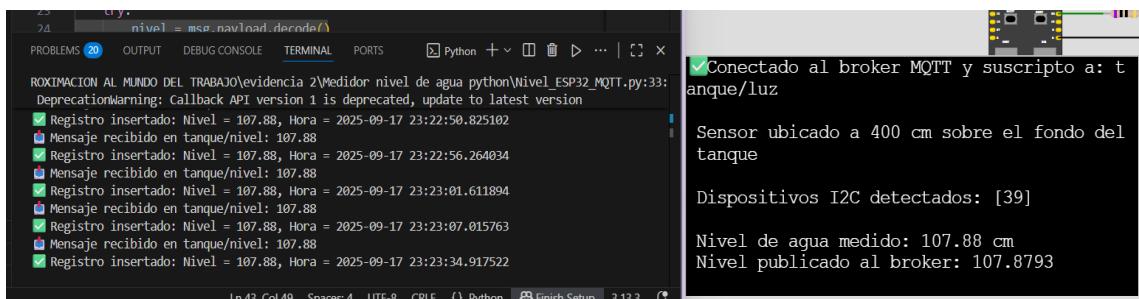
```
Conectando a WiFi...
✓Conectado! IP: 10.10.0.2
✓Conectado al broker MQTT y suscripto a: tanque/luz
```

Sensor ubicado a 400 cm sobre el fondo del tanque

Dispositivos I2C detectados: [39]

Nivel de agua medido: 113.50 cm  
Nivel publicado al broker: 113.5

Conexion exitosa del esp32-S3 a WiFi y broker MQTT, con medición de nivel de agua



*Visualización simultánea: Consola del esp32-S3 en Wokwi y Backend python insertando datos en MySQL*

## Explicación de la base de datos

```
CREATE DATABASE nivel_tanque;
```

```
USE nivel_tanque;
```

```
CREATE TABLE mediciones (
```

```
    id INT AUTO_INCREMENT PRIMARY KEY,
    nivel FLOAT NOT NULL,
    fecha DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

Se creó una base de datos llamada `nivel_tanque` con una tabla `mediciones` que almacena el valor del nivel y la fecha de recepción. El script Python se conecta mediante `mysql-connector-python` y ejecuta la función `crear_tabla_e_insertar(nivel)` para registrar cada medición.

The screenshot shows the MySQL Workbench interface. In the top-left, the 'Navigator' pane displays the database schema with tables like 'club\_futbol', 'esp32', and 'nivel'. The central area shows a 'Query 1' tab with the following SQL code:

```

1 • SELECT `id`,`hora`,`nivel`
2   FROM `esp32`.`nivel`;
3

```

The 'Result Grid' below displays the data from the 'nivel' table:

	id	hora	nivel
11		2025-09-17 22:12:27	107.8793
12		2025-09-17 22:12:32	107.9198
13		2025-09-17 22:13:37	107.9455
14		2025-09-17 22:13:43	107.8793
15		2025-09-17 22:13:48	107.8793
16		2025-09-17 22:41:30	113.53
17		2025-09-17 22:41:47	107.86
18		2025-09-17 22:41:47	107.88
19		2025-09-17 22:41:52	107.97

The bottom section shows the 'Query History' with the following log entries:

Action	Time	Message	Duration / Fetch
1	22:11:26	Apply changes to esp32	0.000 sec / 0.000 sec
2	22:13:13	SELECT `id` FROM `esp32`.`nivel` LIMIT 0, 1000	8 row(s) returned
3	22:13:31	SELECT `id`,`hora`,`nivel` FROM `esp32`.`nivel` LIMIT 0, 1000	12 row(s) returned
4	22:13:43	SELECT `id`,`hora`,`nivel` FROM `esp32`.`nivel` LIMIT 0, 1000	14 row(s) returned
5	22:42:07	SELECT `id`,`hora`,`nivel` FROM `esp32`.`nivel` LIMIT 0, 1000	19 row(s) returned

Consulta SQL y visualización de registros en MySQL Workbench

## Resultados

### Ejemplo de medición

Durante la simulación en Wokwi, el sensor HC-SR04 registró una distancia de 85 cm entre el pelo de agua y el sensor. Dado que el sensor está ubicado a 400 cm del fondo del tanque, el nivel de agua calculado fue:

$$\text{Nivel} = 400 - 85 = 315 \text{ cm}$$

**Este valor se mostró correctamente en la pantalla LCD:**

Nivel medido:

315.0 cm

**Además, se imprimió en consola:**

Nivel de agua medido: 315.00 cm

### Publicación en MQTT

El ESP32-S3 publicó el nivel en el tópico tanque/nivel usando el broker público test.mosquitto.org.

#### En consola se registró:

Nivel publicado al broker: 315.00

#### Desde otro dispositivo o script Python, se recibió correctamente el mensaje:

Mensaje recibido en tanque/nivel: 315.00

Este valor fue almacenado en la base de datos **MySQL** junto con la marca de tiempo.

#### Recepción de comandos

El ESP32-S3 se suscribió al tópico tanque/luz y respondió correctamente a los comandos enviados desde el broker:

##### Al enviar "prender":

*Mensaje en: tanque/luz => prender*

Se encendió el LED verde.

##### Al enviar "apagar":

*Mensaje en: tanque/luz => apagar*

Se apagó el LED verde.

Esto demuestra que el sistema puede ser controlado remotamente mediante MQTT.

#### Alarma por nivel bajo

Cuando el nivel medido fue inferior a 100 cm, el sistema activó la alerta visual:

*Nivel de agua medido: 85.00 cm*

 PRECAUCIÓN: Nivel bajo (< 100 cm)

El LED rojo se encendió automáticamente, indicando que el tanque está en estado crítico. Esta funcionalidad permite actuar rápidamente ante situaciones de riesgo.

# Conclusiones

## Qué aprendimos

Este proyecto nos permitió integrar conocimientos de electrónica, programación en MicroPython, protocolos de comunicación y bases de datos. Aprendimos a escalar un sistema desde la capa física (sensor y actuadores) hasta la capa de transporte (MQTT) y almacenamiento (MySQL), logrando una solución funcional y modular. También reforzamos habilidades de trabajo en equipo, documentación técnica y presentación profesional.

## Qué desafíos enfrentamos

Durante el desarrollo, enfrentamos varios desafíos técnicos, como la configuración de pines en Wokwi, la sincronización entre el ESP32-S3 y el broker MQTT, y la conexión con la base de datos desde Python. También tuvimos que resolver problemas de sintaxis, errores de conexión y conflictos en el repositorio. Estos obstáculos nos ayudaron a mejorar nuestra capacidad de depuración, organización y colaboración.

## Qué mejoraríamos o escalaríamos

Como mejora futura, proponemos escalar el sistema para incluir:

- Control automático de bomba: encendido/apagado según el nivel medido
- Notificaciones móviles: alertas por Telegram, WhatsApp o correo
- Panel web o app: visualización remota del nivel y control manual
- Historial gráfico: visualización de datos almacenados en la base
- Sensores múltiples: monitoreo de varios tanques en paralelo

De esta manera, este trabajo demuestra cómo un sistema sencillo puede escalarse desde la capa física hasta la capa de transporte y almacenamiento, sentando las bases para aplicaciones IoT más complejas.”