UNIVERSITY OF VOCATIONAL TECHNOLOGY

Faculty of Engineering Technology

Department of Electro-Mechanical Technology

EE402040

Internet of Things (IoT)

IoT- Based Tank Water Level Monitoring System

Project Report

Group Members

Name	**Reg. no**
D.D.Bandara	: MAN/22/B1/58
J.A.D.C. Yasiru Jayasuriya	: MAN/22/B1/23

Program	: B.Tech in Manufacturing Technology
Submission Date	: 21st June 2025

Instructed by: Mr. Janith Kasun

Contents

1. Introduction

Problem Statement

Water scarcity and inefficient water management are significant global challenges. In many residential, commercial, and industrial settings, manual monitoring of water tank levels is common. This method is time-consuming, prone to human error, and lacks real-time data, often leading to overflow, wastage, or unexpected water shortages.

Furthermore, in large or distributed environments, physical inspection of multiple tanks becomes impractical and costly. The lack of proactive monitoring can result in significant financial losses due to water damage, increased utility bills, and operational disruptions. There is a pressing need for an automated, reliable, and real-time solution to efficiently manage water resources.

Motivation

The motivation behind developing an IoT-based tank water level monitoring system stems from the desire to address the inefficiencies and challenges associated with traditional water management. By leveraging IoT technologies, we aim to provide a solution that offers real-time visibility into water levels, enabling timely action and informed decision-making. This system will promote water conservation by preventing overflows, reduce the burden of manual checks, and enhance the overall efficiency of water usage. The integration of ESP32 microcontrollers, a robust communication protocol, and a user-friendly web interface aligns with the principles of smart living and sustainable resource management, contributing to a more connected and intelligent environment.

Aim & Objectives

Aim: To design, develop, and implement a reliable and efficient IoT-based system for real-time monitoring of water tank levels, accessible via a web application.

Objectives (SMART Goals):

- Specify and select appropriate hardware components, including two ESP32 microcontrollers and a suitable water level sensor.
- Measure water levels accurately using the selected sensor and transmit data wirelessly from the sensor node ESP32 to the communication node ESP32 within a range of 20 meters.
- Automate the transmission of water level data from the communication node ESP32 to a cloud-based platform using an efficient communication protocol (MQTT/HTTP) every 5.
- Realize a Python-based web application that displays real-time water levels, historical data, and provides customizable alerts to users, with a functional prototype.
- Test and optimize the system for reliability, data accuracy, and user experience, ensuring 95% system uptime and reliable data transmission for a continuous 24-hour

2. Literature Review

Existing Water Level Monitoring Solutions

Existing water level monitoring solutions range from simple mechanical float switches to more advanced ultrasonic and radar-based systems. Mechanical float switches, while inexpensive, only provide binary (full/empty) indications and are prone to mechanical failures. Ultrasonic sensors offer continuous level measurement and are widely used in industrial applications due to their non-contact nature. Examples include systems employing Arduino boards with HC-SR04 ultrasonic sensors for basic monitoring. More sophisticated industrial solutions often utilize Programmable Logic Controllers (PLCs) with various sensor types, offering high precision but at a significant cost and complexity.

In the realm of IoT, several projects have explored water level monitoring using various microcontrollers like Arduino, Raspberry Pi, and ESP series. These often integrate with cloud platforms such as AWS IoT, Google Cloud IoT Core, or open-source alternatives like Node-RED and Thingspeak. Communication protocols commonly employed include Wi-Fi, LoRa, and cellular networks. For instance, some projects utilize Wi-Fi-enabled microcontrollers to send data to a cloud server, which then processes and visualizes the data on a web dashboard. Others focus on low-power wide-area networks (LPWAN) like LoRa for long-range, low-data-rate transmissions, suitable for remote water tanks.

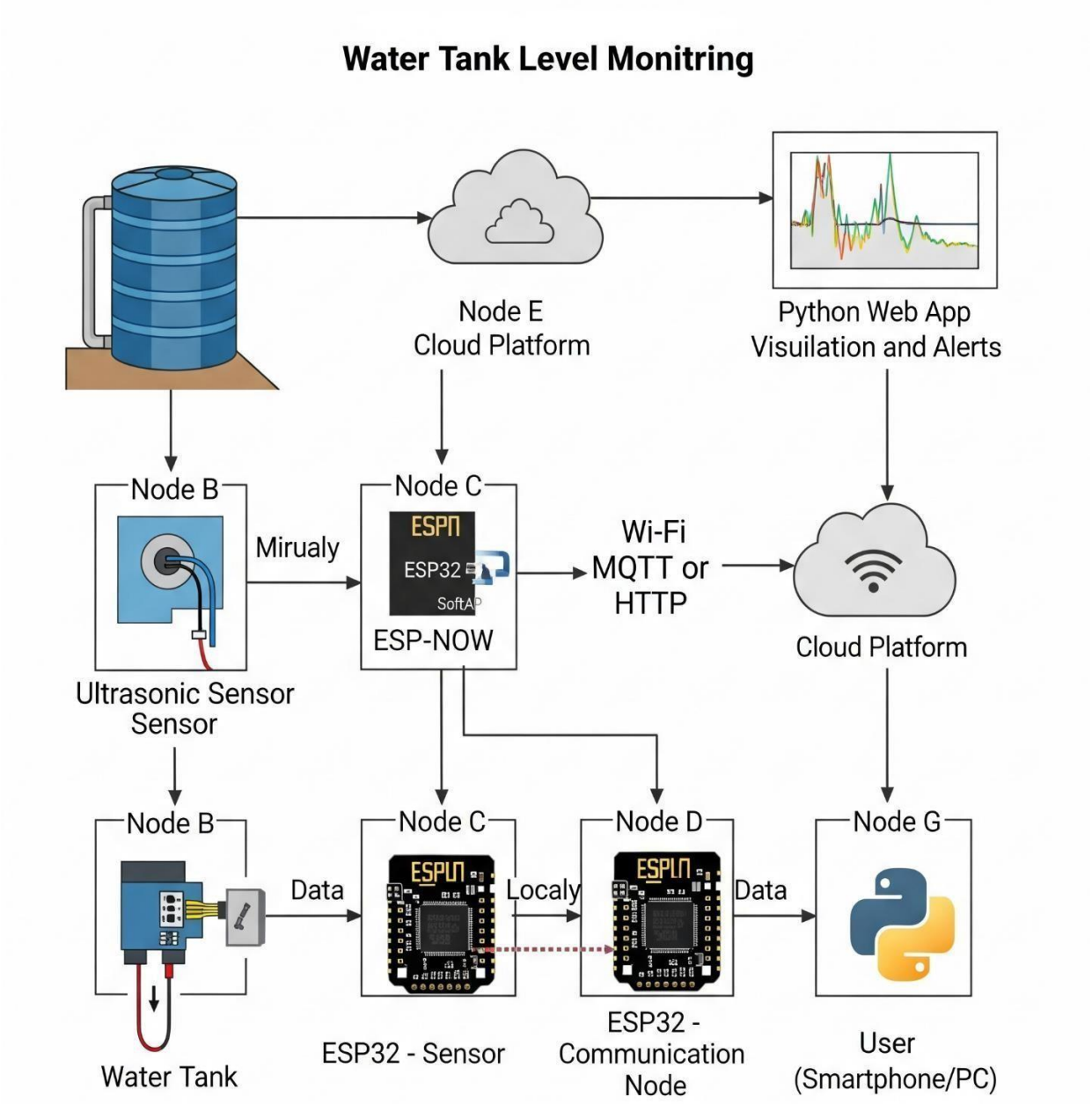
Gaps in Existing Solutions

While existing solutions provide valuable insights, several gaps and limitations persist. Many low-cost DIY solutions lack robustness, scalability, and advanced features such as historical data analysis, predictive analytics, or sophisticated alert mechanisms. Industrial solutions, while robust, are often proprietary, expensive, and not easily adaptable for smaller-scale or residential applications. The common use of a single microcontroller for both sensing and communication can sometimes lead to resource contention and reduced efficiency, especially in scenarios requiring rapid data acquisition and concurrent data transmission. Furthermore, many systems lack a dedicated communication layer, which could enhance reliability and provide a more robust data pipeline, particularly in environments with fluctuating network conditions or when the sensor node is in a less optimal location for direct Wi-Fi connectivity. The integration of two ESP32 boards, one for sensing and local processing and another for dedicated communication, offers a unique opportunity to address these limitations by creating a more modular, efficient, and resilient system architecture. This separation of concerns can improve performance, reduce latency, and enhance the overall reliability of the data transmission.

3. Methodology & System Design

Architecture Diagram

The system will employ a layered architecture comprising a sensing layer, a communication layer, a cloud layer, and a presentation layer.



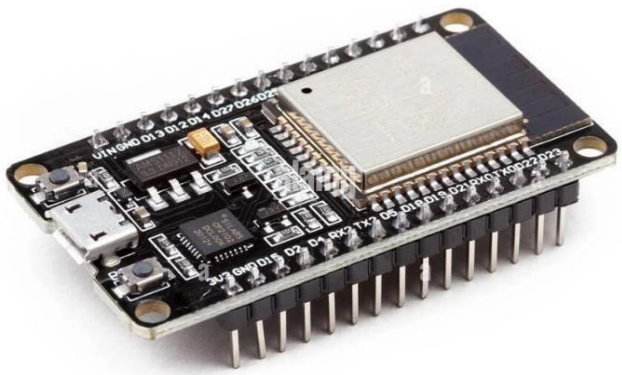
Description

The Sensing Layer involves the ultrasonic sensor measuring water levels within the tank. The ESP32 Sensor Node reads data from the sensor and performs initial processing. The Communication Layer consists of the dedicated ESP32 Communication Node, which receives processed data from the sensor node and transmits it to the cloud. The Cloud Layer encompasses a cloud platform for data ingestion, storage, and processing. Finally, the Presentation Layer is a Python web application that visualizes the data and provides user interaction.

Hardware Components

List of Hardware Components:

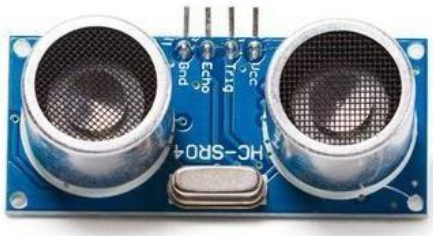
1. **ESP32 Development Board (x2):** One ESP32 will serve as the "Sensor Node," responsible for reading sensor data. The second ESP32 will function as the "Communication Node," dedicated to managing cloud communication.



- ESP32 boards are chosen for their integrated Wi-Fi and Bluetooth capabilities, dual-core processor, sufficient GPIO pins, and low power consumption, making them ideal for IoT applications requiring network connectivity and local processing. Their robust community support and development tools (Arduino IDE) facilitate rapid prototyping and deployment.

2. **Ultrasonic Sensor (e.g., HC-SR04 or JSN-SR04T):** To measure the distance to the water surface.

- Ultrasonic sensors provide non-contact, accurate distance measurements, which can be translated into water level. They are relatively inexpensive and widely available. The JSN-SR04T version is preferred for its waterproof transducer, making it suitable for direct immersion or use in humid environments.



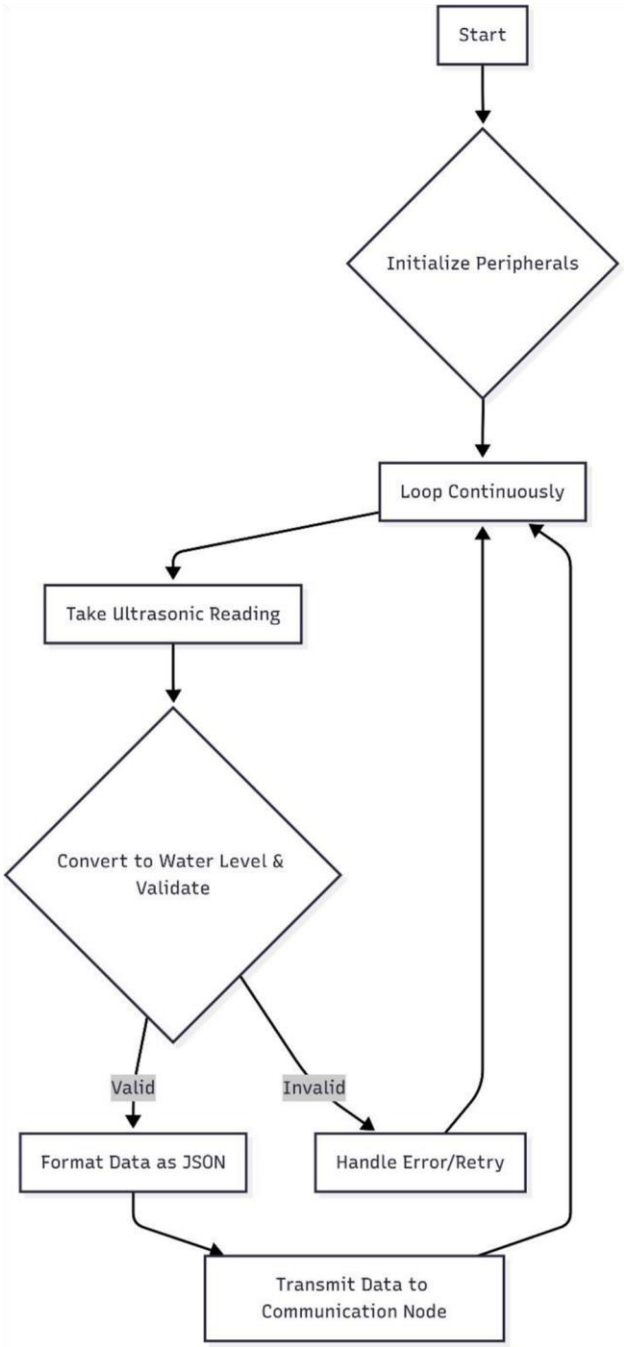
3. **Power Supply (5V DC adapter, portable power bank):** For powering both ESP32 boards and the sensor.
    - A stable power supply is crucial for continuous operation. Options will be explored for both mains power and battery backup for reliability.

4. **Jumper Wires and Breadboard:** For prototyping and connecting components.

5. **Waterproof Enclosure:** For protecting the sensor and ESP32 boards from environmental elements, especially for outdoor tank installations.
    - Essential for ensuring the longevity and reliability of the electronic components in a potentially wet environment.

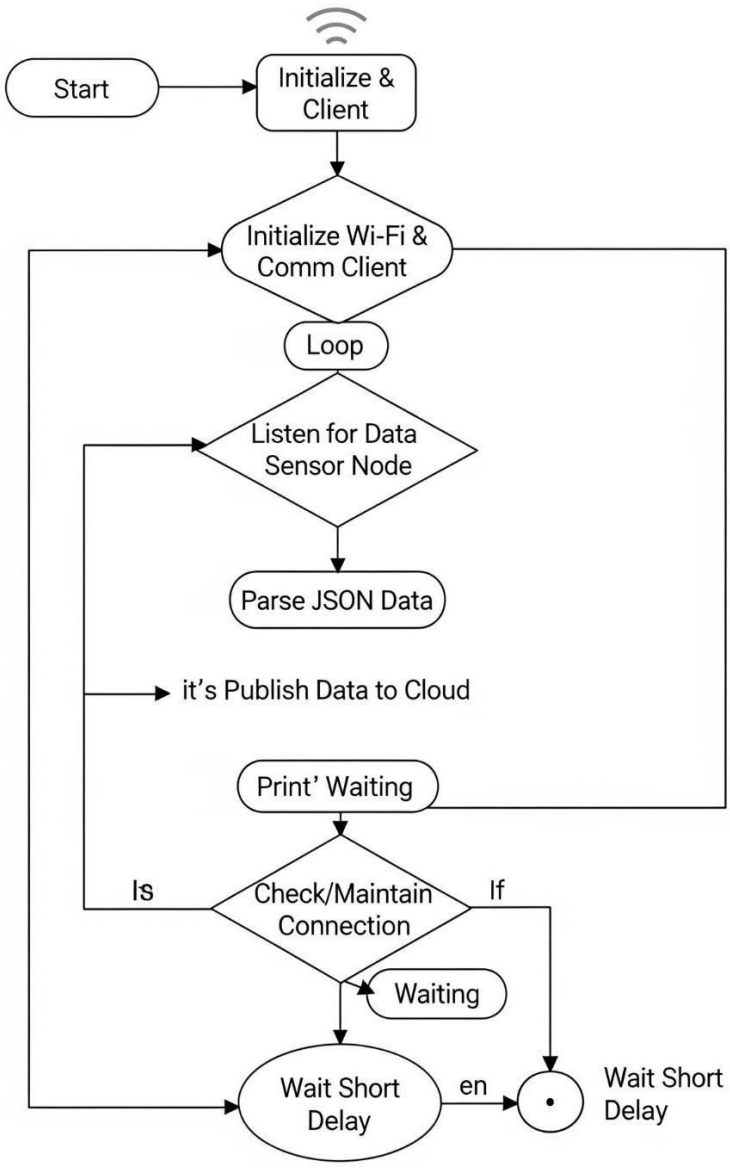
ESP32 (Sensor Node) Flowchart

This flowchart illustrates the operational logic of the ESP32 Sensor Node. It begins with initialization, then enters a continuous loop to read sensor data, process it, format it as JSON, and send it to the Communication Node. Error handling is included for invalid sensor readings.



ESP32 (Communication Node) Flowchart

This flowchart depicts the operational flow for the ESP32 Communication Node. It involves initializing Wi-Fi and the communication client, then continuously listening for data from the Sensor Node. Upon receiving data, it parses the JSON and publishes it to the cloud. It also manages connection maintenance.



Communication Protocol (MQTT/HTTP) Selection

Choice: HTTP (Hypertext Transfer Protocol)

Reason:

1. **Universality and Broad Compatibility:**

   HTTP is the foundational protocol of the World Wide Web and is universally understood by virtually all networked devices, including web servers, client applications (web browsers, mobile apps), and many IoT platforms.

   This wide compatibility ensures that your IoT device can easily communicate with a wide range of services and applications, making integration simpler.

2. **Simplicity of Implementation for Basic Communication:**
   - For simple request-response interactions, HTTP is straightforward to implement on both the IoT device (client) and the server side.
   - Many IoT devices might primarily need to send data (e.g., sensor readings) to a server or receive simple commands, which HTTP's `POST` and `GET` methods handle efficiently.
3. **Firewall and Network Friendliness:**
   - HTTP (and particularly HTTPS) traffic typically uses standard ports (80 for HTTP, 443 for HTTPS) that are usually open by default in network firewalls. This reduces configuration complexities and improves the chances of successful communication, especially in diverse network environments.
4. **Existing Infrastructure and Tooling:**
   - There's a vast ecosystem of tools, libraries, frameworks, and established best practices for developing and securing HTTP-based applications. This makes development, debugging, and maintenance easier.
   - Cloud IoT platforms often provide HTTP/HTTPS endpoints for device connectivity, simplifying onboarding.

5. **Security (with HTTPS):**
    - While raw HTTP is not secure, using HTTPS (HTTP over SSL/TLS) provides encryption for data in transit, ensuring confidentiality and integrity, and authentication of the server. This is crucial for IoT projects where sensitive data might be exchanged or where command integrity is vital.
6. **Stateless Nature (and its benefits):**
    - HTTP is inherently stateless, meaning each request is independent. For many IoT scenarios, such as sending periodic sensor data, this statelessness is perfectly adequate and can simplify server design as the server doesn't need to maintain complex session information for each device.
7. **Web Integration:**
    - If your IoT project involves a web-based dashboard or a mobile application for user interaction, HTTP/HTTPS provides a seamless way for the IoT devices to interact with these front-end applications, either directly or through an intermediate API gateway.

Example JSON Structure for Water Level Data:

```
{
 "sensorId": "TANK_001",
 "timestamp": "2025-06-21T12:30:00Z",
 "waterLevel_cm": 150,
 "tankCapacity_cm": 200,
 "percentageFull": 75.0,
 "status": "normal"
}
```

Field Descriptions:

- sensorId: Unique identifier for the water tank sensor.
- timestamp: UTC timestamp of when the data was recorded.

- waterLevel_cm: Current water level in centimeters from the bottom of the tank.
- tankCapacity_cm: Total height/capacity of the tank in centimeters.
- percentageFull: Water level expressed as a percentage of tank capacity.
- status: Current status of the tank (e.g., "normal", "low", "high", "overflow").

Web Application (Python) Features and Design

The Python web application will serve as the primary user interface for monitoring and managing the water level system. It will be developed using a suitable web framework.

Features:

1. **Real-time Dashboard:** Displays the current water level, percentage full, and status of connected tanks in real-time.
2. **Historical Data Visualization:** Interactive charts and graphs showing water level trends over time (hourly, daily, weekly, monthly).
3. **Alerts and Notifications:** Configurable thresholds for low, high, and overflow levels. Users will receive notifications (e.g., on the dashboard, email, or push notifications - as a future enhancement) when these thresholds are crossed.
4. **Multi-tank Support:** Ability to monitor multiple water tanks, each with its unique ID and customizable parameters.
5. **User Authentication:** Secure login for authorized users to access their tank data.
6. **Responsive Design:** Optimized for viewing on various devices (desktop, tablet, mobile).

Network Connectivity

The overall system network connectivity will involve several key stages:

1. **Sensor Node to Communication Node:** A local wireless connection will be established between the two ESP32 boards. This could be achieved using:

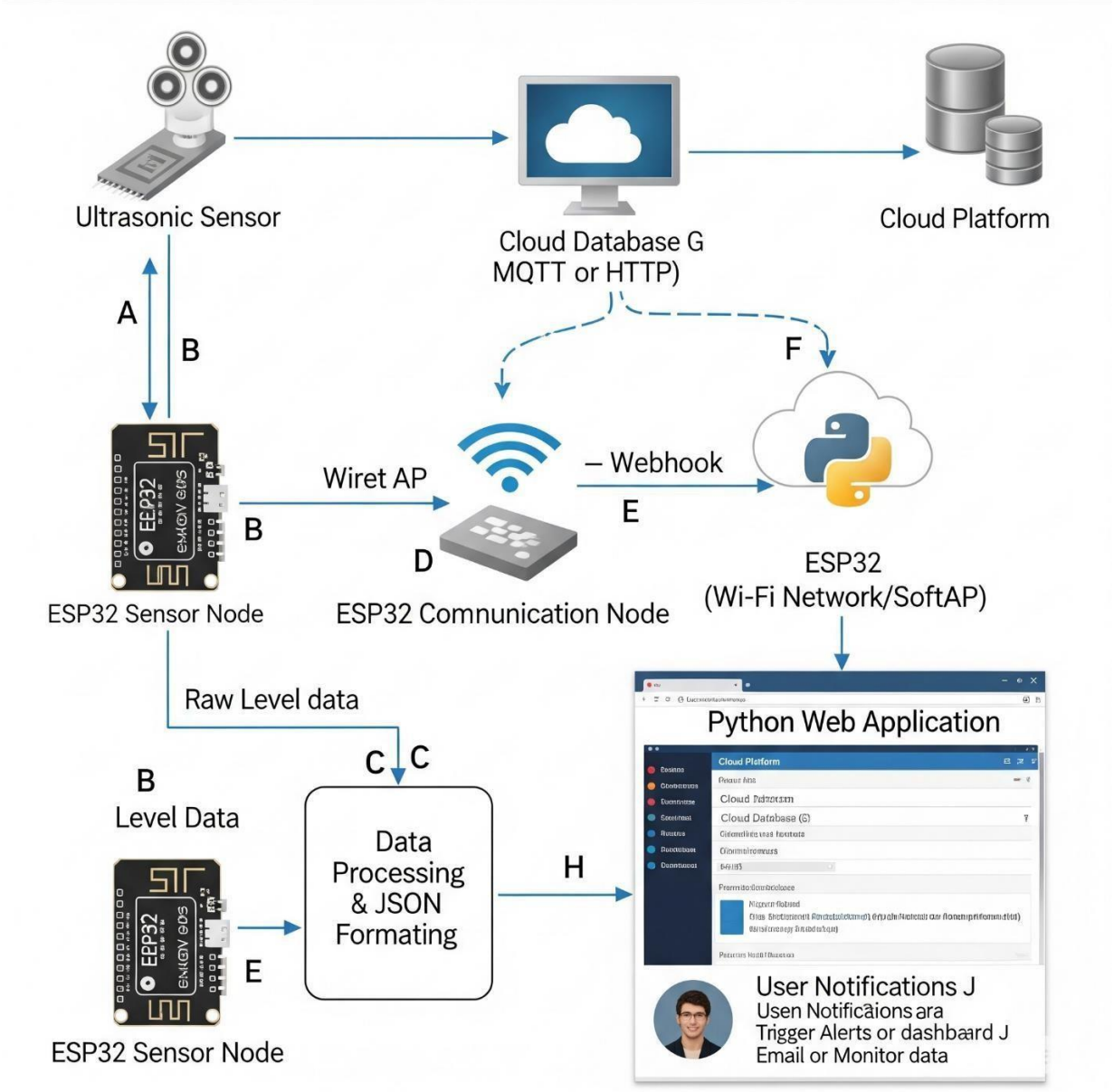
- **ESP-NOW:** A connectionless communication protocol that allows devices to communicate directly without a router, ideal for short-range, fast data exchange.
- SoftAP (Access Point) Mode: The Communication Node ESP32 can act as an access point, and the Sensor Node ESP32 connects to it as a station. This provides a simple local network.

  The chosen method will ensure robust and low-latency data transfer between the two ESP32s.

2. **Communication Node to Cloud Platform:** The Communication Node ESP32 will connect to the local Wi-Fi network (router/hotspot) and then use either MQTT or HTTP over Wi-Fi to send data to the chosen cloud platform.
3. **Cloud Platform to Web Application:** The cloud platform will act as a central data broker. The Python web application will access this data via the cloud platform's provided APIs (e.g., REST API, MQTT subscription to the cloud broker) to retrieve real-time and historical water level information.
4. **User to Web Application:** Users will access the web application through a standard web browser on their smartphones, tablets, or personal computers, provided they have internet access.

Data Flow Diagram

This diagram illustrates the overall flow of data within the system, from the sensor acquisition to user interaction.



Description:

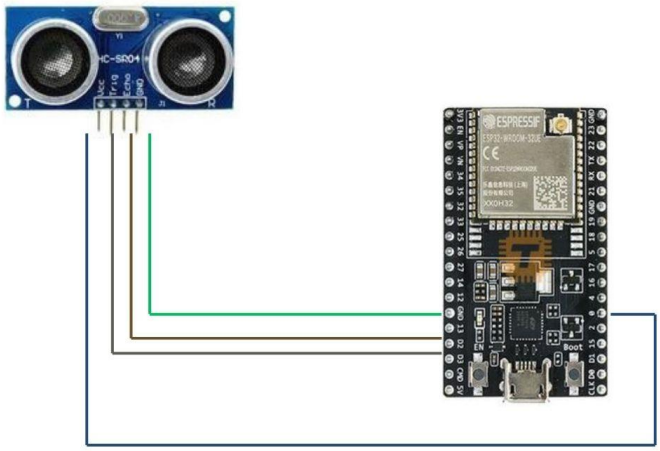
1. Ultrasonic Sensor measures the distance, sending raw data to the ESP32 Sensor Node.
2. ESP32 Sensor Node processes the raw data, converts it into water level, and formats it into a JSON payload.

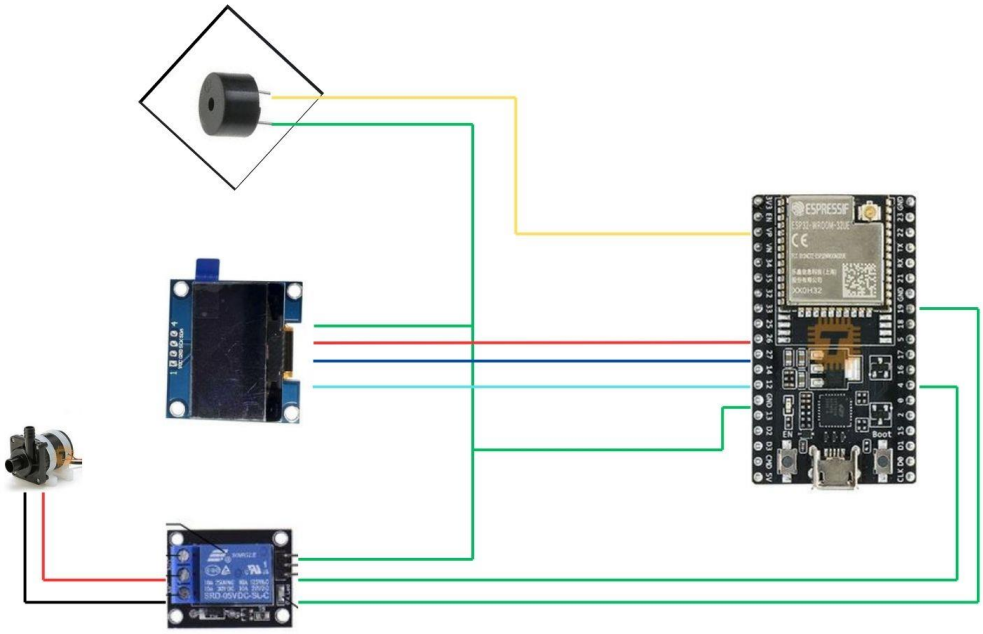
3. The JSON data is sent wirelessly (e.g., via ESP-NOW) to the ESP32 Communication Node.
4. The ESP32 Communication Node receives the JSON data, connects to the local Wi-Fi Network/Router, and publishes the data to the Cloud Platform using MQTT/HTTP.
5. The Cloud Platform acts as a central hub, ingesting the data and storing it in a Cloud Database.
6. The Python Web Application retrieves real-time and historical data from the Cloud Platform via its API or webhook.
7. The Web Application renders the data on a Web Interface, which is accessed by the User Device.
8. Based on predefined thresholds, the Web Application triggers User Notifications (e.g., on the dashboard or via email).

4. CODE

https://github.com/yasiru758/water-level-monitoring-system-

5. Wiring Diagram





6.Implementation Plan & Timeline

Task Breakdown and Responsibilities

Phase	Key Tasks	Responsibility
1. Planning & Setup	Project proposal, hardware procurement, environment setup, cloud research.	Both
2. Sensor Node Hardware & Firmware	Sensor interface, C++ firmware, basic sensor testing.	Both
3. Local Communication Implementation	Implement ESP-NOW/SoftAP on ESP32s, test local data transfer.	Both
4. Communication Node & Cloud Integration	Configure Wi-Fi, implement MQTT client, test E2E cloud data.	Both
5. Web Application Backend Development	Flask setup, python web	Both
6. Web Application Frontend & Features	Dashboard design, alert system, user authentication.	Both

7. System Integration, Testing & Documentation	Testing, debug/optimize, final report.	Both

7.Chalenges We Faced

- **Sensor Accuracy and Environmental Factors:**
    - Ultrasonic sensors can be affected by factors like humidity, temperature fluctuations, and tank irregularities (e.g., internal structures, foam on the water surface) which might lead to inaccurate readings. The proposal mentions the preference for the JSN-SR04T for its waterproof transducer, implying awareness of environmental challenges.
    - "Testing and optimize the system for reliability, data accuracy, and user experience" is an objective, suggesting that achieving accuracy might be a challenge.
- **Wireless Communication Reliability (ESP-NOW/SoftAP between ESP32s):**
    - Maintaining a stable and low-latency wireless connection between the "Sensor Node ESP32" and the "Communication Node ESP32" within the specified 20-meter range could pose challenges, especially if there are obstacles or interference.
    - The proposal highlights the benefit of separating concerns by using two ESP32s to "enhance reliability and provide a more robust data pipeline, particularly in environments with fluctuating network conditions or when the sensor node is in a less optimal location for direct Wi-Fi connectivity," implying these are potential issues with single-microcontroller setups.

- **Web Application Development and Real-time Display:**
    - Developing a "Python-based web application that displays real-time water levels, historical data, and provides customizable alerts to users" requires significant effort. Challenges could include:
        - Achieving true real-time data visualization with minimal latency.
        - Implementing efficient data retrieval and processing from the cloud database.
        - Designing a user-friendly and responsive interface.
        - Setting up "configurable thresholds for low, high, and overflow levels" and delivering timely notifications.
- **System Integration and Debugging:**
    - Integrating various hardware and software components (sensors, two ESP32s, communication protocols, cloud platform, web application) can be complex, and identifying and resolving issues across different layers (sensing, communication, cloud, presentation) will likely require extensive debugging. The proposal includes "System Integration, Testing & Documentation" as a crucial phase, indicating its complexity.

8. How We solved them

- **Sensor Accuracy and Environmental Factors:**
    - **Challenge:** Ultrasonic sensors can be affected by environmental factors, leading to inaccurate readings.
    - **Solution:** The proposal specifically mentions preferring the JSN-SR04T version of the ultrasonic sensor for its waterproof transducer, making it suitable for direct immersion or use in humid environments. The "ESP32 (Sensor Node) Flowchart" also includes a "Convert to Water Level & Validate" step with an "Invalid" path leading to "Handle

Error/Retry," indicating a planned approach to deal with potentially unreliable readings. Additionally, the objective to "Test and optimize the system for reliability, data accuracy, and user experience" implies a rigorous testing phase to ensure accurate measurements.

- **Wireless Communication Reliability (ESP-NOW/SoftAP between ESP32s):**
    - **Challenge:** Maintaining a stable and low-latency wireless connection between the two ESP32 nodes, especially in environments with fluctuating network conditions or less optimal sensor node locations.
    - **Solution:** The system's architecture with two ESP32s is specifically designed to enhance reliability and provide a more robust data pipeline by separating the sensing and communication concerns. The proposal suggests using "ESP-NOW" or "SoftAP (Access Point) Mode" for this local wireless connection, with ESP-NOW being ideal for "short-range, fast data exchange" and SoftAP providing a "simple local network". The "Local Communication Implementation" phase in the timeline is dedicated to implementing and testing this data transfer.
- **Web Application Development and Real-time Display:**
    - **Challenge:** Displaying real-time water levels, historical data, and providing customizable alerts effectively.
    - **Solution:** The "Python web application" will feature a "Real-time Dashboard" to display current data, "Historical Data Visualization" with interactive charts, and an "Alerts and Notifications" system with configurable thresholds. The objective is to "Realize a Python-based web application that displays real-time water levels, historical data, and provides customizable alerts to users, with a functional prototype".
- **System Integration and Debugging:**

- **Challenge:** The complexity of integrating various hardware and software components.
- **Solution:** The "Implementation Plan & Timeline" includes a dedicated "System Integration, Testing & Documentation" phase , which will involve "Testing, debug/optimize, final report". This systematic approach is designed to address integration issues. The "ESP32 (Sensor Node) Flowchart" also includes error handling for invalid sensor readings.

9.Expected Outcomes & Deliverables

At the culmination of this project, the following outcomes and deliverables are expected:

1. Functional Prototype System: A working IoT-based tank water level monitoring system comprising:
    - One ESP32 Sensor Node with an integrated ultrasonic sensor, capable of accurately measuring water levels and transmitting data locally.
    - One ESP32 Communication Node, capable of receiving data from the Sensor Node and publishing it to a cloud platform via Wi-Fi using MQTT.
2. Real-time Web Application: A Python-based web application with a user-friendly interface that displays live water level readings, historical data, and provides customizable alerts.
3. Source Code: Well-documented and modular source code for:
    - ESP32 firmware (C++).
    - Python web application (backend and frontend).
4. Project Report: A comprehensive report detailing the problem, motivation, literature review, system design, implementation details, testing procedures, results, challenges faced, and future enhancements. This report will adhere to academic formatting standards.
5. Technical Documentation: Diagrams (architecture, data flow), circuit diagrams, component specifications, and setup instructions to enable future replication or expansion of the system.

6. Python based web dashboard

Detailed Schedule (Gantt Chart - Conceptual)

No	Task Name	June									July													
		22	23	24	25	26	27	28	29	30	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Project Planning & Setup																							
2	Sensor Node Hardware & Firmware																							
3	Local Communication Implementation																							
4	Communication Node & Cloud Integration																							
5	Web Application Backend Development																							
6	Web Application Frontend & Features																							
7	System Integration, Testing & Documentation																							

10.Budget

The estimated budget and required resources for this project are outlined below. Efforts will be made to utilize readily available and cost-effective components.

Hardware:

Item	Quantity	Estimated Price (LKR per unit)	Total Estimated Cost (LKR)
ESP32 Development Boards	2	1,200	2,400
Ultrasonic Sensor	1	800	800
Jumper Wires	1 set	500	500
Breadboard	1	400	400
Power Supply (5V DC adapter)	1	950	950
Waterproof Enclosure	1	1000	1000
Miscellaneous (resistors, etc.)	-	500	500
Subtotal Hardware	-	-	**6,550**

11. Youtube Demo Link : https://youtu.be/F7NY0R4gzGg

References

- M. K. Sharma, *Internet of Things: Concepts and Applications*. CRC Press, 2021.
- Espressif Systems, "ESP32 Technical Reference Manual." [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf (Accessed: June 21, 2025).
- https://www.daraz.lk/
- https://tronic.lk/