Solar-Powered Iot-Based Barangay Drainage Canal Water Level Monitoring and Overflow Warning Simulation System Using ESP32 and Python Application

Kzel Gale D. Austria[1], Ronchester M. Batac[2], Jose Jr. C. Felipe[3], Prince August H. Luciano[4], Joyrish Ashanti B. Santos[5], Khertly A. Vosotros[6]

Faculty of Department of Computer Engineering Fulfillment of the Requirements for the Subject Software Design

DOI: https://doi.org/10.47772/IJRISS.2026.100600842

Received: 15 June 2026; Accepted: 20 June 2026; Published: 07 July 2026

ABSTRACT

Localized flooding in urban community drainage systems remains a recurring environmental hazard in the Philippines, driven primarily by insufficient infrastructure maintenance and obstructed waterways. This special project presents the design, fabrication, and validation of a Solar-Powered IoT-Based Barangay Drainage Canal Water Level Monitoring and Overflow Warning Simulation System, an independent engineering solution tailored for community-level flood preparedness. The system architecture is split into a standalone Power Track—utilizing a 6V/6W polycrystalline solar panel, a TP4056 charging module, two 3.7V/3500mAh Lithium-Ion batteries in parallel, and an LM2596S buck converter—and a wireless Data Telemetry Track driven by an ESP32 microcontroller and an HC-SR04 ultrasonic sensor. Real-time acoustic reflection distances are parsed over a localized network socket interface to a standalone Python Tkinter desktop application named *HydroSense System*. Empirical testing within a simulated glass aquarium drainage model demonstrated a 100% functional success rate for telemetry transmission at 115200 baud, with no data packet drops. The software successfully categorized water levels into Normal, Warning, and Danger thresholds, instantly updating color-coded GUI alert themes (Green, Yellow, and Red) while simultaneously driving physical LED arrays and a passive buzzer alarm. Furthermore, an automated background engine flawlessly appended timestamped logs to local CSV spreadsheets for community record-keeping. The findings prove that integrating sustainable energy harvesting with localized IoT frameworks delivers an affordable, power-autonomous, and dependable early-warning system that significantly enhances local barangay disaster risk management without grid reliance or complex server overhead.

Keywords — Internet of Things (IoT), ESP32, Python Tkinter, Flood Early Warning System, Solar Energy Harvesting, Disaster Risk Reduction.

INTRODUCTION

Flooding is a recurring environmental hazard in the Philippines, particularly in urban areas with insufficient or poorly maintained drainage systems. Short rainfall events often lead to water accumulation and localized flooding, causing property damage and transportation disruptions. Research indicates that these incidents are frequently caused by insufficient drainage maintenance and obstructed waterways (Varela et al., 2022). While drainage canals are essential for directing excess rainwater, they are often compromised by improper waste disposal, which necessitates efficient monitoring (Jadhav et al., 2025).

The advancement of the Internet of Things (IoT) has enabled the development of smart environmental monitoring systems that facilitate automated data collection. Research demonstrates that microcontrollers like the ESP32, integrated with ultrasonic sensors, provide effective real-time water level measurements and warning notifications (Rahman et al., 2025). However, a significant limitation of field-deployed IoT devices is power sustainability. Standard battery-operated systems require frequent maintenance and are prone to downtime. To

ensure continuous operational longevity and minimize manual intervention, integrating renewable energy harvesting—specifically solar power—is necessary for autonomous system functionality.

In response, the researchers propose a **Solar-Powered IoT-Based Barangay Drainage Canal Water Level Monitoring and Overflow Warning Simulation System Using ESP32 and Python Application**. This prototype is designed for community-level monitoring and features a self-sustaining power architecture utilizing a polycrystalline solar panel and a rechargeable battery management system. The system employs an ultrasonic sensor for water level detection, an ESP32 for data processing, and physical alerts via LEDs and buzzers. Real-time visualization and historical logging are managed by a Python-based application called the *HydroSense System.*

The system classifies water levels into **Normal**, **Warning**, and **Danger** categories to support barangay officials' disaster response efforts. Although the prototype focuses on water level changes rather than direct waste detection, it demonstrates how IoT technology, coupled with sustainable energy harvesting, can enhance local flood preparedness and monitoring efficiency (Jadhav et al., 2025).

LITERATURE REVIEW

Flooding and Drainage Canal Conditions in Communities

According to Varela et al. (2022), flooding in communities is often due to insufficient or clogged drainage canals, with a need for additional canals and maintenance. High drainage density is associated with minimal flooding. Additionally, Many low-income countries find that stormwater drainage is one of the most urgent infrastructure needs in urban areas. Many urban areas are situated in coastal regions with the highest average rainfall and are subject to flooding. Tropical rainfall is more intense than in temperate climates. (Cairncross & Ea, 1991). Moreover, Flooding is a serious concern in cities of developing countries because of its scale of damage to inhabitants and built infrastructure. Climate change and extremes increase flood hazards, enhancing the vulnerability of communities, particularly those residing in flood-prone areas and slums. Technocratic technical measures have long been adopted to protect people behind the structures, but the scope of protection is limited, and it is even more hazardous when the structures collapse. Subsequently, contemporary measures with flood warning and rescue emerge. (Nur & Shrestha, 2017)

Internet of Things (IoT) In Environmental Monitoring

The integration of the Internet of Things (IoT) into environmental monitoring is transforming the way we collect, analyze, and respond to environmental data. IoT technologies, including sensors and real-time data analytics, allow for continuous monitoring of critical environmental parameters, such as air and water quality, temperature, humidity, and pollutant levels. By enabling real-time data collection, IoT empowers governments, industries, and researchers to make more informed decisions to mitigate environmental risks and promote sustainable development. (Carter, 2022). The growing environmental challenges, such as pollution, climate change, and resource depletion, have highlighted the need for efficient, real-time monitoring solutions. Traditional methods often lack accuracy, scalability, and automation. The advancement of the Internet of Things (IoT) and sensor technologies has shown innovative approaches for tracking environmental parameters like air quality, water pollution, and temperature variations. These smart systems authorize continuous data collection, real-time analysis, and automated responses to environmental risks. (Lala&Vugar, 2025)

Water Level Monitoring Systems

Ultrasonic sensors, which use high-frequency ultrasonic waves to detect the level of liquids or solids, are often used to monitor water levels. These sensors are mounted at the top of a tank and transmit waves; the time it takes for the return signal to be received by the sensor is measured. The proposed project aims to use a web server for the internal analysis of drainage canal. and municipality water towers to save energy and improve efficiency. (Bagal et al., 2023). According to Lei Jing-feng, (2010) demonstrates that water-level monitoring systems contribute to understanding real-time drainage operation status, helping dispatch decisions and providing early warning for emergency flood control. Furthermore conventional manual monitoring of water bodies has several

disadvantages, including high manual overhead, low accuracy, excessive time in measurements, high communication delay, high cost, life-threatening risks during measurements, etc. The objective of the proposed work is to automatically monitor and control the smart water bodies to sort out the disadvantages of conventional monitoring. (Perumal et al., 2017)

ESP32 Microcontroller

The ESP32 microcontroller is widely adopted for IoT and embedded systems due to its dual-core architecture and wireless connectivity, with applications in smart agriculture, healthcare, smart city infrastructure, industrial IoT, and environmental monitoring. (Rahman et al., 2026) In addition, the ESP32 is a versatile and cost-effective microcontroller widely used in IoT projects, including water reservoir level monitoring, which can improve management processes and reduce water waste. (Fernandes et al., 2024) In addition, Core 1 executes application-specific code that includes sensor fusion, filtering algorithms, and control logic, Core 0 is in charge of the Wi-Fi and Bluetooth stacks. Preprocessing comprises recalibration, statistical outlier-detection methods, and moving-average filtering for noise reduction. The final data is created in JSON format for effective packing and storage. (Pangarkar et al., 2025). Known for its low power consumption, built-in Wi-Fi and Bluetooth, and robust processing power, the ESP32 is an ideal platform for real-time sensing and communication in IoT applications. By leveraging its features, the system can continuously monitor water levels and transmit data over a wireless network to a remote server or cloud-based platform, enabling authorities to respond swiftly to potential flooding or infrastructure failures. (Al-shareeda et al., 2025)

Ultrasonic Sensor for Water Level Detection

Robotic technologies are so varied and impactful, future scientists and engineers benefit immensely from building robots independently, beginning with foundational designs that leverage affordable sensors. (Zhmud et al., 2018) To increase efficiency, sustainability, and product quality, the manufacturing industry is increasingly embracing data collection and analysis to inform decision-making. This is a component of the fourth industrial revolution, which is expected to culminate in Industry 4.0, characterized by completely integrated marketplaces, supply chains, and processes in which intelligent, automated decision-making responds instantly to requests. The implementation of industrial digital technologies (IDTs), including edge computing, cloud computing, smart sensors, the internet of things (IoT), and machine learning (ML) will bring about this change. This calls for both online and in-line sensors that don't need human operators; online measurements employ automated sampling techniques, while in-line technologies measure the process stream directly. By adding features such as wireless IoT connectivity or processing the collected data to reduce its complexity, sensors can be made into smart sensors. Process interconnectivity, such as cloud computing, where data is moved to a centralized cloud site, or edge computing, where compute nodes are situated near end devices, needs hardware solutions. ML may be used to analyze data and make decisions automatically at all levels, from individual sensors to cloud-based data centers. (Bowler et al., 2022)

Warning and Alert Systems

Lias and Kumaran, (2024) developed an IoT-based flood monitoring system using ESP32 that utilized three LED indicators and a buzzer to classify flood severity levels. Their system assigned different alert conditions to green, yellow, and red indicators and generated alarm signals when dangerous water levels were detected, demonstrating the effectiveness of color-coded warnings and audible alerts in flood monitoring. Additionally, Cadeliña and Perin, (2022) designed a flood alarm system using ultrasonic sensors and a buzzer to warn communities located near rivers. The study emphasized that audible alarms are effective in notifying both authorities and residents when water levels reach danger thresholds, thereby improving preparedness and response during flood events. Moreover, Lengkong et al. (2024) implemented an IoT-based water-level early warning system that transmitted real-time notifications through Telegram. Their findings showed that automated notifications provide timely information to users even when they are away from the monitoring site, highlighting the importance of remote warning mechanisms in modern flood monitoring systems. Furthermore, Diriyana et al., (2019) developed a water-level monitoring and flood early warning system that continuously monitored water levels and delivered warnings through online platforms. The researchers concluded that real-time alerts

improve community awareness and enable faster responses to potential flooding situations. Ultimately, Hanan et al., (2019) developed a water-level detection system using ultrasonic sensors, ESP8266, and buzzer communication media. The study demonstrated that combining visual monitoring with buzzer alarms enhances the effectiveness of flood detection systems by providing both visual and audible warnings when water levels exceed predefined thresholds.

Python-Based Monitoring Systems

Python is also widely used for collecting, storing, and analyzing historical datasets generated by IoT devices. Zulhakim et al., (2023) demonstrated that sensor data processing techniques can improve the reliability and quality of monitoring systems by reducing noise and enhancing measurement accuracy. Historical datasets enable trend analysis, performance evaluation, and future decision-making, all essential for flood monitoring and prediction systems. In addition, Python provides robust libraries for serial communication, enabling seamless data exchange between computers and microcontrollers such as the ESP32. Rahman et al., (2026) explained that the ESP32 supports multiple communication protocols and is highly compatible with software platforms used in IoT applications. Through serial communication, sensor readings can be transmitted directly from the ESP32 to a Python application for processing, visualization, and storage. The effectiveness of combining ESP32 devices with software-based monitoring platforms was further demonstrated by Al-Shareeda et al., (2025), who developed a secure real-time water-level monitoring system that utilized ESP32 technology for continuous data acquisition and remote monitoring. Their findings emphasized the importance of real-time data presentation and reliable communication in ensuring the effectiveness of monitoring systems for critical infrastructure. Furthermore, Bowler et al., (2022) highlighted the growing importance of smart sensors, cloud computing, edge computing, and data analytics in modern monitoring systems. The researchers noted that software platforms capable of processing and visualizing real-time sensor data are essential components of Industry 4.0 applications. This supports the use of Python as a tool for integrating sensor measurements, data analysis, and visualization into a unified monitoring platform such as HydroSense. Overall, the literature indicates that Python provides an effective environment for developing monitoring systems due to its support for dashboard creation, real-time visualization, historical data management, and serial communication with IoT devices. These capabilities justify its use in HydroSense as the primary software platform for monitoring water levels and generating meaningful information for users.

Solar Energy Harvesting and Battery Management in IoT Nodes

The power supply for sensor nodes is a challenge, as they are typically battery-powered. Batteries used include primary (non-rechargeable) and secondary (rechargeable) batteries. Primary batteries are typically used as the power source for many sensor networks, and the lifetime of these sensor nodes is the time required to discharge below the minimum charge level required by each sensor node. Although batteries have high energy density, they have a limited lifetime. For long-term deployments, regular battery replacements are required. Battery replacement can be expensive and not feasible in a case like remote environmental monitoring. (Wu et al., 2023). Oluremi et al., (2025) investigated solar energy harvesting techniques for IoT sensor nodes and emphasized the importance of intelligent energy management in maintaining long-term operation.

Their study proposed adaptive energy prediction and management mechanisms to optimize the utilization of harvested solar energy under changing environmental conditions. The researchers concluded that solar energy harvesting can significantly improve the sustainability of remote IoT deployments by reducing dependence on battery replacement and maintenance. Furthermore, Salman et al., (2026) developed a smart charging framework for solar-powered ESP32-based IoT systems operating in remote environments. The framework employed Maximum Power Point Tracking (MPPT) and battery state-of-charge management to optimize charging efficiency and prevent excessive battery degradation.

Their findings revealed that limiting battery discharge and implementing intelligent charging strategies increased battery cycle life by approximately four times, demonstrating the importance of battery management in long-term field deployments. Moreover, Velliangiri, (2025) proposed a low-power IoT node architecture utilizing ESP32 microcontrollers and deep-sleep protocols for remote sensor networks. Through adaptive wake-sleep

scheduling, the system reduced power consumption by more than 75% compared to continuously active devices. The study highlighted that deep-sleep operation is one of the most effective methods for extending battery life in environmental monitoring applications where sensor readings are collected periodically rather than continuously. Additionally, Castro Garcia et al., (2025) introduced I-Canopy, an energy-autonomous environmental monitoring platform based on the ESP32. The system incorporated energy-aware operation, local data storage, and cloud synchronization to maintain functionality under limited power availability and intermittent connectivity.

The researchers demonstrated that combining solar power with intelligent energy management strategies allows IoT monitoring systems to operate reliably in remote and rural environments. Overall, Sahani et al., (2026) presented a high-efficiency solar energy harvesting and power management unit designed specifically for ultra-low-power IoT nodes. The proposed system integrated adaptive Maximum Power Point Tracking and power regulation mechanisms to maximize energy extraction from solar panels while maintaining stable operation of sensor nodes. Their findings indicate that advanced energy harvesting circuits can further improve the reliability and autonomy of self-powered IoT monitoring systems.

METHODOLOGY

Research Design

The researchers utilized an Applied Engineering Research Design focused on Functional Prototyping. This design is highly appropriate for the project as it aims to deliver a tangible, operational solution—specifically, a solar-powered IoT device—to address a localized environmental challenge: real-time flooding within community drainage canals.

Using this methodology, the development process was structured into the following systematic phases: system logic mapping, hardware circuit design, software development (including the ESP32 firmware and Python GUI), and final system integration. This engineering framework enabled an iterative design process, allowing the researchers to continuously evaluate, debug, and refine the prototype. This ensured maximum accuracy in sensor calibration, robust serial communication, and a highly stable power management system prior to the final system demonstration.

System Architecture and Quick Design Phase

The proposed system's operational framework is divided into two distinct functional tracks: the Power Track and the Data Telemetry Track. This architecture was systematically planned across three development stages: system mapping, physical containment design, and electrical circuit routing, providing a high-level conceptual blueprint prior to hardware fabrication.

The standalone energy collection sequence is illustrated in Figure 1. This loop is engineered for grid-independent field operation, beginning at the solar panel, which converts ambient solar radiation into electrical current. This harvested energy continuously replenishes the rechargeable Li-Ion batteries. To protect downstream components from voltage fluctuations, an LM2596S Buck Converter steps down and stabilizes the current into a steady 5V power rail, which directly drives the main ESP32 microcontroller.

Concurrently, the communication and logging sequence is mapped out in Figure 2. The pipeline begins with the HC-SR04 ultrasonic sensor, which measures the instant water level inside the drainage canal simulation model using acoustic wave reflections. The ESP32 processes this raw data and transmits the telemetry stream wirelessly over a local Wi-Fi network interface using network sockets to the host computer executing the HydroSense Python application. The software instantly parses the data against predefined alert thresholds, updates the GUI display for the on-duty barangay officials, and appends the timestamped records to a local CSV file for permanent historical logging.

ISSN No. 2454-6186 | DOI: 10.47772/IJRISS | Volume X Issue VI June 2026

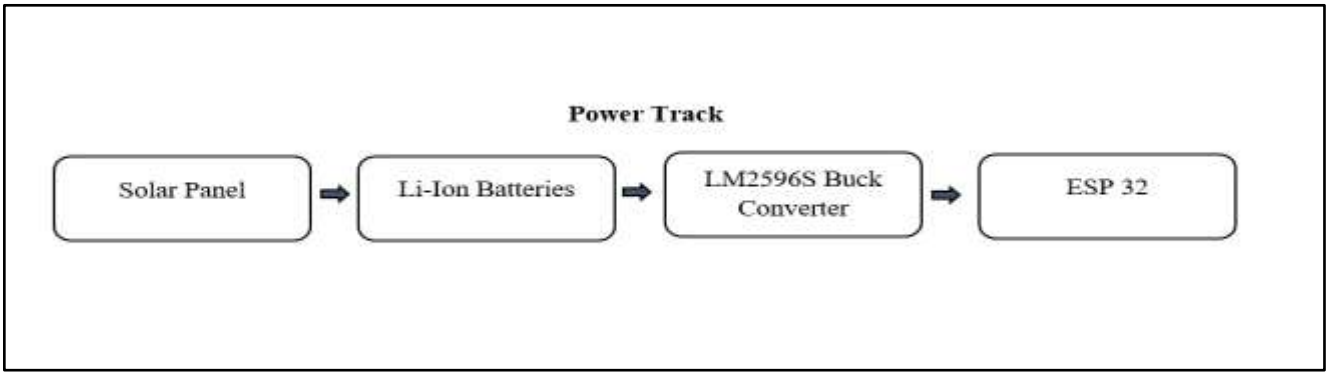


Figure 1. Power Track Diagram

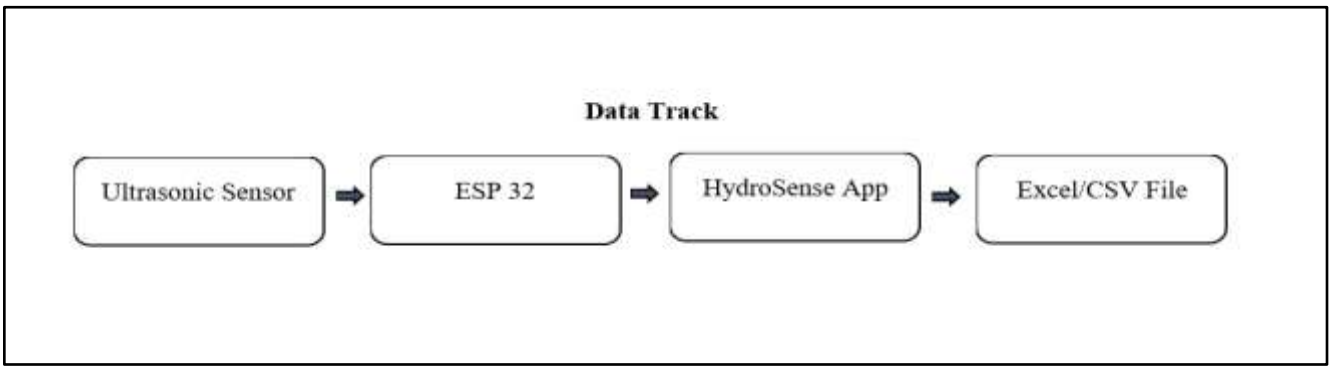


Figure 2. Data Telemetry Track

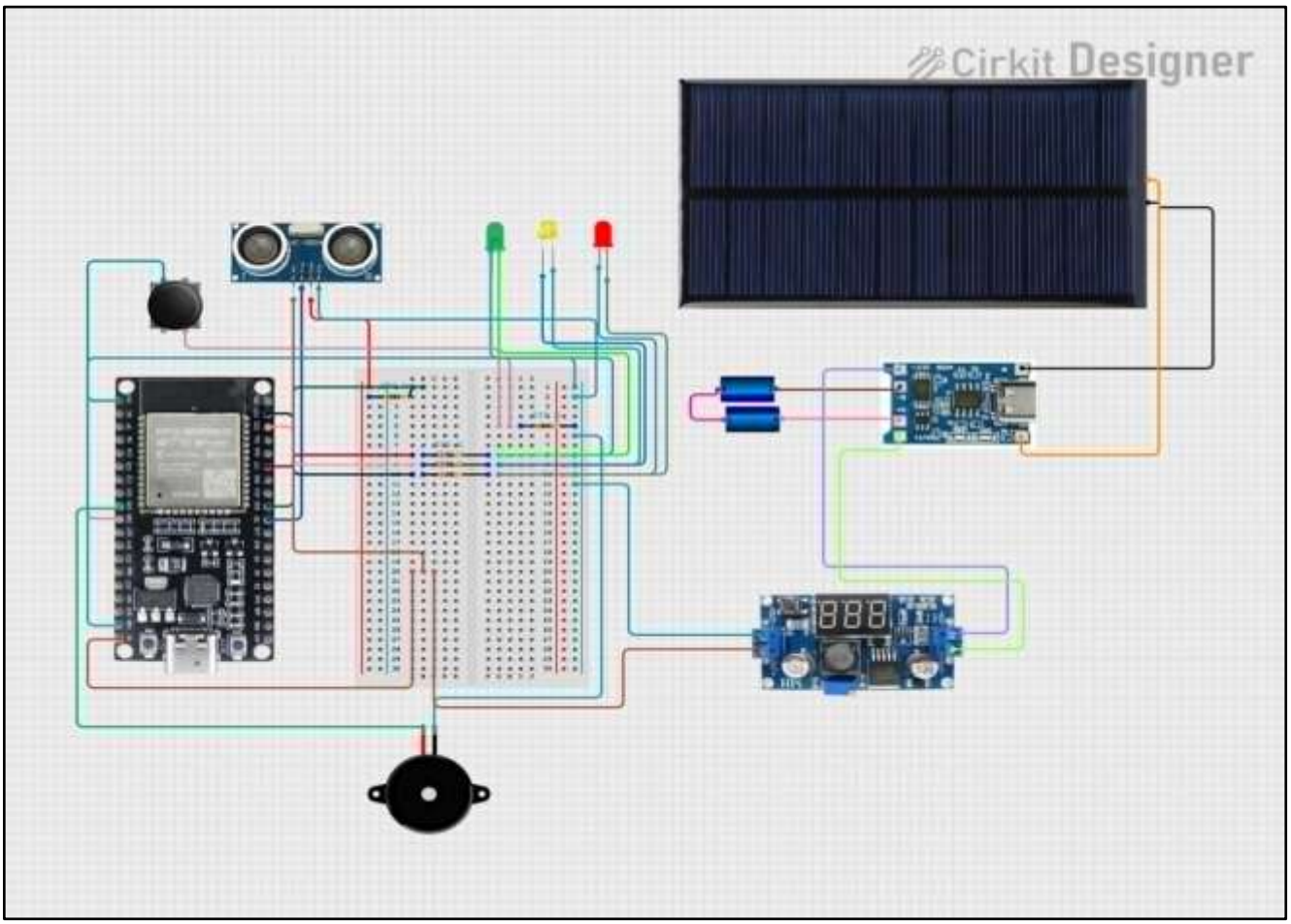


Figure 3. System Circuit Schematic Diagram

Software Architecture and Development

The software component developed for this project is named HydroSense, a dedicated desktop application engineered in Python using the Tkinter library for its Graphical User Interface (GUI). The software employs a modular design architecture, cleanly separating the backend logical processing from the frontend visual interfaces. This structure ensures high maintainability, efficient data handling, and reliable performance during continuous operations.

HydroSense Deployment Setup and Configuration

To ensure data integrity, accountability, and system security, the *HydroSense* application implements a restricted-access control framework via the HydroSense Deployment Setup interface. Because the system is exclusively designed as a localized utility for barangay personnel and authorized disaster monitoring officers rather than general residents, the initial system initialization requires the operator to configure several critical parameters before the primary dashboard can be accessed:

- **Location Information:** The specific barangay name, zone, or street classification where the physical IoT node is deployed, ensuring precise tracking of which drainage canal is being monitored.
- **Assigned Personnel:** The official name and identification details of the personnel currently on duty, establishing a clear audit trail for accountability.
- **Drainage Information and Thresholds:** The physical dimensions (total depth and width) of the specific drainage canal simulation model along with the customized centimeter limits assigned for the Normal, Warning, and Danger statuses.
- **Network Connection Configuration:** The input of the active IP Address assigned to the ESP32 microcontroller and the corresponding network port to establish a secure, wireless communication pipeline over the local Wi-Fi network between the host computer and the hardware node.

Wi-Fi Data Parsing and Risk State Logic

Once the deployment setup is finalized and authenticated, the software utilizes network communication libraries to continuously listen to the incoming wireless data stream transmitted from the hardware over the local area network (LAN). The raw distance string received from the ESP32 is parsed, filtered for errors, and converted into an active numerical value representing centimeters. The system then applies a fundamental telemetry formula to calculate the actual height of the water:

Water Level = Total Canal Depth - Sensor Distance Reading

The resulting value is fed into an execution block containing *if/elif/else* conditional statements. This logic dynamically determines whether the software triggers a state change, shifting the GUI color themes (Green, Yellow, or Red) and sending immediate low-level execution commands back to the ESP32 to toggle the physical LEDs or active buzzer accordingly.

Historical Data Logging and Excel/CSV Integration

For long-term trends analysis, post-disaster evaluation, and official community record-keeping, the *HydroSense* system features an automated, background-running logging subsystem. Every time a new data packet is received from the hardware node, the application does not merely refresh the screen; it instantly appends and saves the structured information directly into an external Excel-compatible (.csv) spreadsheet file for permanent data storage and historical analysis.

Solar Energy Harvesting Architecture

To achieve long-term field operational longevity without relying on grid power, the prototype implements a

localized, self-sustaining solar energy harvesting architecture. The primary energy collection unit chosen for this design is a **6V/6W polycrystalline solar panel**. The selection of this specific rating and physical footprint was guided by crucial practical constraints: first, its compact dimensions are perfectly proportional to the scaled-down miniature simulation prototype; and second, it offered a highly cost-effective option that conformed to the rigid budget limitations of the project without compromising the fundamental energy demands of the circuit.

The solar panel is routed directly through a TP4056 Lithium-Ion charging network to safely replenish two Model Li-IP-3.7V/3500 Lithium-Ion batteries configured in parallel. This parallel configuration maintains a nominal system voltage of 3.7V while combining cell capacities to maximize operational runtime. Because the voltage inputs from renewable energy harvesting fluctuate significantly with real-time ambient weather conditions, the power management sub-circuit integrates regulator modules to step down and stabilize the peak energy from the solar panel into a continuous, clean 5V power rail. This power rail meets the operational voltage requirements of the ESP32 microcontroller and its attached ultrasonic sensors, safeguarding the active microelectronics against unexpected voltage spikes.

System Requirements

Table 1: Hardware Specifications and Functions

Hardware	Specification	Description
Laptop/Computer	Intel Core i3 or higher, 4GB RAM minimum	Used for programming, monitoring, and running the Python application
Solar Panel	Model 136X110-3 (6V, 6W, Polycristalline)	Captures solar radiation and converts it into electrical energy (approx. 330mA max current) to power the circuit and recharge the system during peak daylight hours.
Rechargeable Battery	Li-1P-3.7V/3500 (2X Parallel Configuration), Lithium-Ion 18650 cells, combined capacity of 7000mAh, output voltage 3.7V nominal.	Provides a steady current reserve to the ESP32 microcontroller and sensor modules during nighttime, overcast periods, or low-daylight conditions.
Battery Charging Module	TP4056 Lithium-Ion	Functions as the core energy regulation safety link between the solar harvesting panel and the storage bank. It safely regulates raw incoming solar voltage to charge the parallel-connected 18650 Lithium-Ion cells, preventing deep over-charging or battery draining below safe operating thresholds.
DC-DC Step-Down Power Module	LM2596S Adjustable Buck Converter	Steps down the fluctuating voltage coming from the solar panel or battery configuration to a stable 5V power rail suitable for the ESP32 microcontroller.
Diagnostic & Assembly Tools	Multi-tester, Soldering Iron, Soldering Wire	Equipment used for component validation, connection resilience testing, circuit debugging, and permanent hardware prototyping.
ESP32 Microcontroller	ESP32 Wi-Fi Development Board	Serves as the main controller that processes sensor data and transmits information
Ultrasonic Sensor	HC-SR04	Measures the water level by detecting the distance between the sensor and the water surface
LEDs	Red, Yellow, Green LEDs	Provide visual warning indicators for

		different water levels
Buzzer	Passive Buzzer	Produces audible warnings during danger levels
Push Button	Standard Push Button Switch	Allows the user to manually turn off the buzzer
Aquarium	Glass or Acrylic Tank	Represents the miniature drainage canal simulation environment
Jumper Wires	Male-to-Male, Male-to-Female, and Female-to-Female	Used for component connections
Resistor	220 Ohms	Limits electrical current and protects circuit components
Breadboard	Standard Breadboard	Used for circuit prototyping
USB Cable	Type-C	Used for initial programming, uploading firmware to the ESP32, and bench testing.

Table 2: Software Specifications and Functions

Software	Specification	Description
Operating System	Windows 10/11	Required for running the development tools and monitoring software
Arduino IDE	Latest Version	Used for programming and uploading code to the ESP32
Python	Python 3.x	Used for developing the HydroSense Intelligent Hydraulic Telemetry monitoring dashboard
Python Libraries	Tkinter	Used for graphical interface, communication, data logging, and visualization
Code Editor	VSCode	Used for Python development

Build Prototype Phase

The fabrication and software integration of the proposed prototype were executed through a systematic workflow divided into the following phases:

1. **Preliminary Research and Scope Definition:** The researchers conducted comprehensive technical reviews to map out the hardware limitations and establish a controlled project scope. To maintain an optimized and manageable data infrastructure, it was determined that **only designated barangay personnel or local disaster risk reduction officers would be granted authorized access** to the *HydroSense Deployment*. Limiting the deployment scope to localized community officials prevents complexities regarding data security, authorization protocols, and potential server overhead that would typically arise from open-access public distribution.

2. **Procurement of System Requirements:** Following the initial research phase, all critical hardware components cataloged in the engineering requirements were gathered. This inventory included an ESP32 development board, an HC-SR04 ultrasonic sensor, a polycrystalline solar panel, an LM2596S buck converter, indicator LEDs, a passive buzzer, breadboards, jumper wires, and current-limiting resistors.

3. **Pin Configuration Mapping and Wiring Logic:** To prevent short circuits and ensure proper hardware interfacing, an online-assisted technical mapping process was conducted. The researchers analyzed technical datasheets and schematic modeling tools to plan out the General Purpose Input/Output (GPIO) pin assignments of the ESP32 microcontroller. This involved mapping the exact digital lines for the ultrasonic sensor's Echo and Trigger pins, as well as the output channels for the status LEDs and the passive buzzer alarm.

4. **Firmware Uploading and Breadboard Prototyping:** Prior to permanent assembly, the electronic components were temporarily structured on a prototype breadboard. The Arduino IDE environment was

utilized to write, compile, and upload the core C++ firmware to the ESP32 microcontroller. Early sensor distance readouts and hardware states were monitored through the serial terminal to validate data integrity and confirm circuit continuity before physical fabrication.

System Fabrication and Assembly Method

The project's structural construction was executed as a simulated environment designed to validate the practical feasibility of deploying localized IoT-based monitoring nodes within real-world municipal drainage infrastructure. To provide a highly observable, transparent simulation environment for water-level fluctuations, a **glass aquarium tank** was selected as the central drainage model. The use of an aquarium tank was highly strategic: it enabled efficient filling and draining during stress testing, and its wide-open top aperture provided an optimal layout for mounting the ultrasonic sensor directly above the water surface without acoustic interference.

To ensure a highly professional, streamlined, and safe presentation, all intricate electrical circuitry—including the breadboard matrix, the LM2596S buck converter, and the core ESP32 microcontroller—was housed in a dedicated containment enclosure **at the base of the prototype**. This centralized layout also minimized the risk of component degradation caused by accidental water splashes during simulation routines.

A miniature utility pole was constructed on the right side of the tank to serve as a physical alert console, housing three color-coded LED indicators (Red, Yellow, Green) and a buzzer alarm for instantaneous, localized signaling. Concurrently, the connecting wires for the ultrasonic sensor were seamlessly routed through the rear facade of the aquarium downward into the base enclosure, ensuring that the front layout remains free of cluttered wiring and completely accessible for visual analysis.

RESULTS AND DISCUSSION

The two components of implementation are software and hardware.

Hardware Implementation

The physical construction of the proposed project was successfully fabricated into a functional, self-contained simulation prototype. Figure 4 illustrates the numbered, multi-angle breakdown of the hardware materials used in the construction, pinpointing the physical location of each component. Furthermore, the complete, integrated operational setup, showing the hardware prototype functioning side-by-side with the HydroSense desktop interface, is presented in Figure 5.

Figure 4. Numbered Multi-Angle Breakdown of Material Components

1. Ultrasonic Sensor
2. ESP32
3. Breadboard
4. Resistor
5. LM2596S Adjustable Buck Converter
6. Batteries
7. TP4056 Lithium-Ion Battery Charging Module
8. Jumper Wires
9. LEDs
10. Buzzer
11. Aquarium Tank
12. Solar Panel

Figure 5. Final Operational System Prototype and Desktop Application Setup

During testing, the hardware components responded according to their intended functions, wherein the ultrasonic sensor successfully detected water level changes and triggered the corresponding warning indicators based on the identified threshold level.

Software Implementation

The *HydroSense* software application was successfully built and executed using Python and the Tkinter library to serve as a centralized monitoring terminal for local disaster risk management. To maximize field usability, the raw script was compiled into a standalone executable application (.exe), allowing it to launch natively on any local computer at the barangay hall without needing Python pre-installed. The sub-sections below discuss the practical operational performance and interfaces of each software component verified during system execution.

HydroSense Deployment Setup Interface

Upon execution, the system prompts the operator with the HydroSense Deployment Setup Interface (Figure 6 and Figure 7). This window functions as a localized access control and system initialization portal. During testing, the input text fields properly accepted and validated parameters including specific Location Information (e.g., City/Municipality and Barangay), Assigned Personnel identification details, and custom drainage dimensions. Crucially, the interface successfully validated the network configuration fields, allowing the user to input the active local IP Address and Network Port assigned to the ESP32 microcontroller to establish the wireless telemetry pipeline over the local network interface.

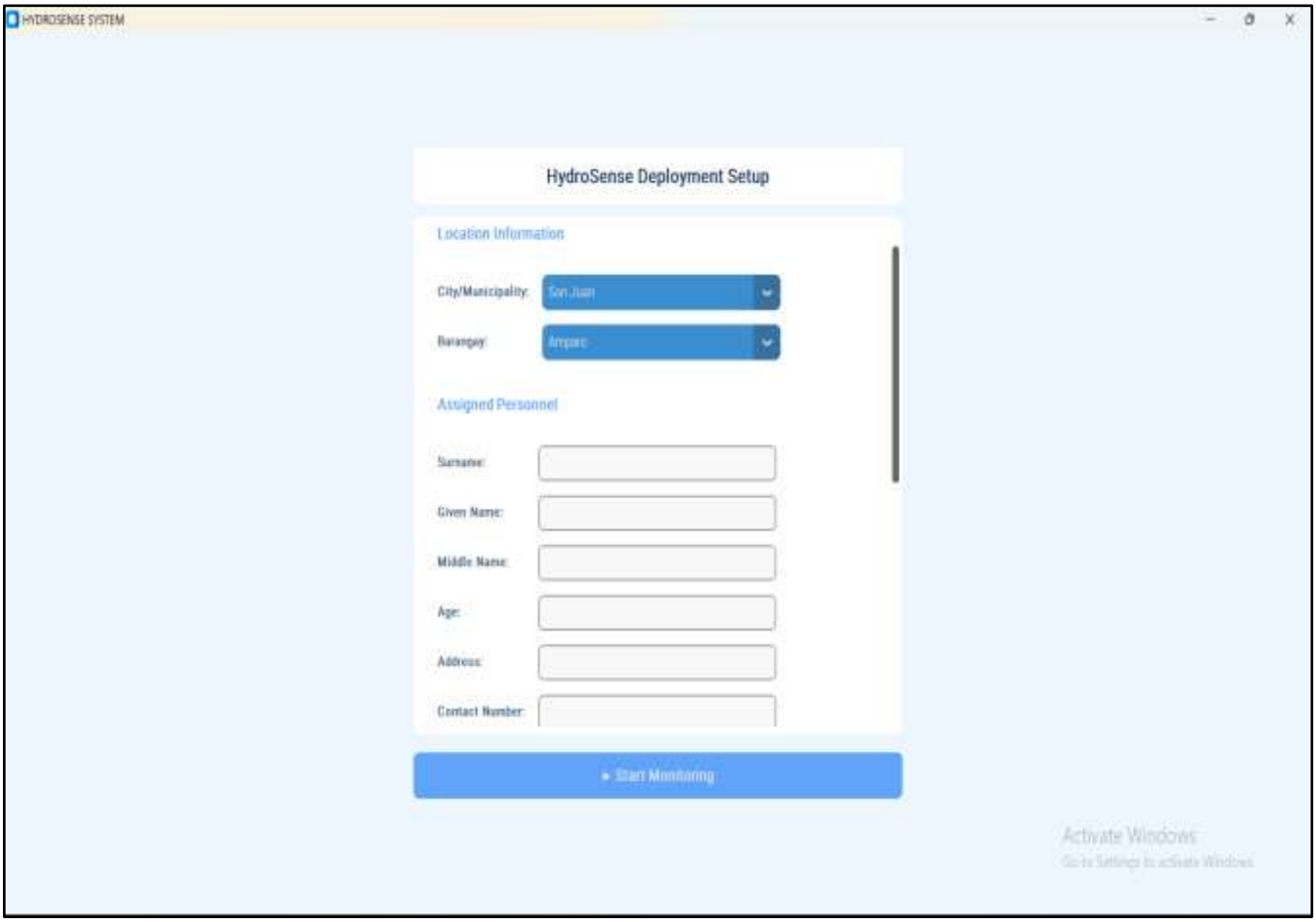


Figure 6. HydroSense Deployment Setup

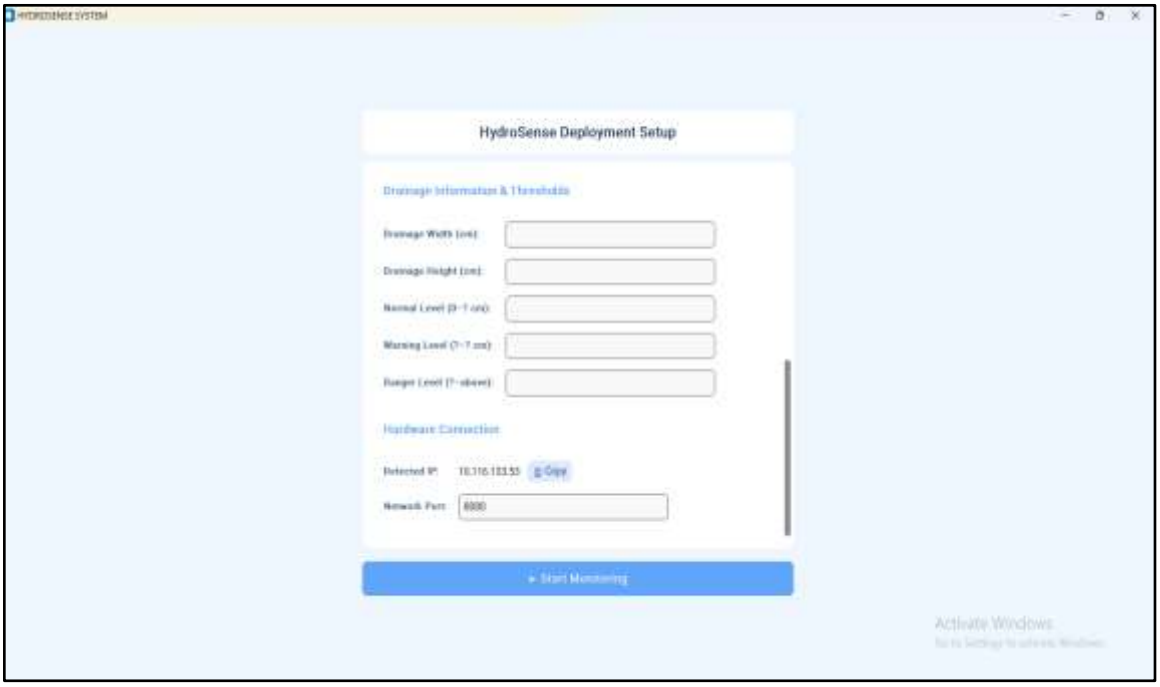


Figure 7. HydroSense Deployment Setup

Normal State Interface

The *Normal State Interface* (Figure 8) represents the ongoing baseline monitoring loop. During prolonged operation under low-water environments, the graphical tracking metrics, telemetry logs, and status readouts remain fully synchronized and stable. No visual artifacts or buffer delays were observed, confirming that the modular data-parsing routines seamlessly process continuous loops of low-fluid-height data packets transmitted over the shared local network.

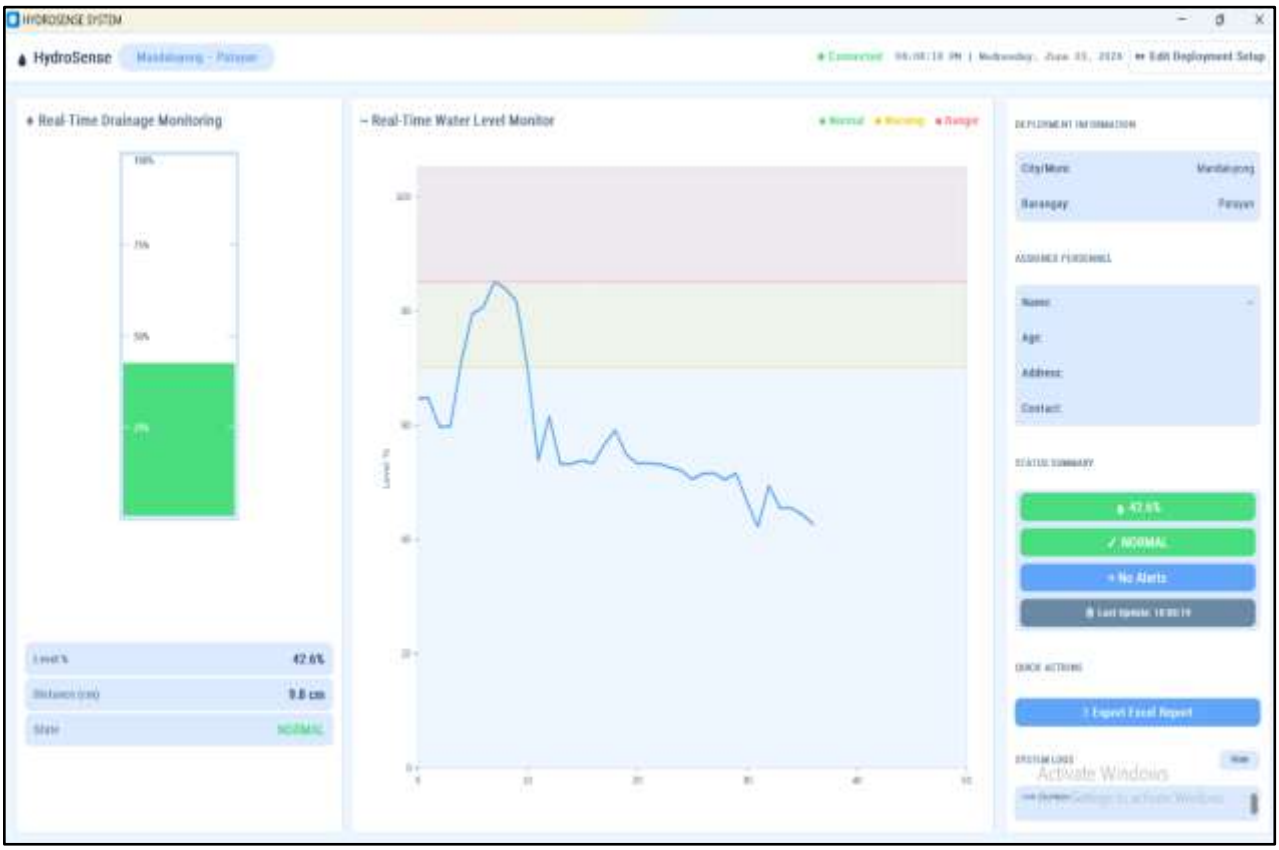


Figure 8. Normal State Interface

Warning State Interface

As fluid is introduced into the simulation tank and breaches the first user-defined boundary, the application shifts into the *Warning State Interface* (Figure 9). The backend logic registers the threshold violation and immediately switches the dashboard theme to a bright Yellow color alert. This color shift visually draws the on-duty barangay official's attention, signaling a rising trend of water accumulation within the drainage infrastructure that requires close situational awareness.

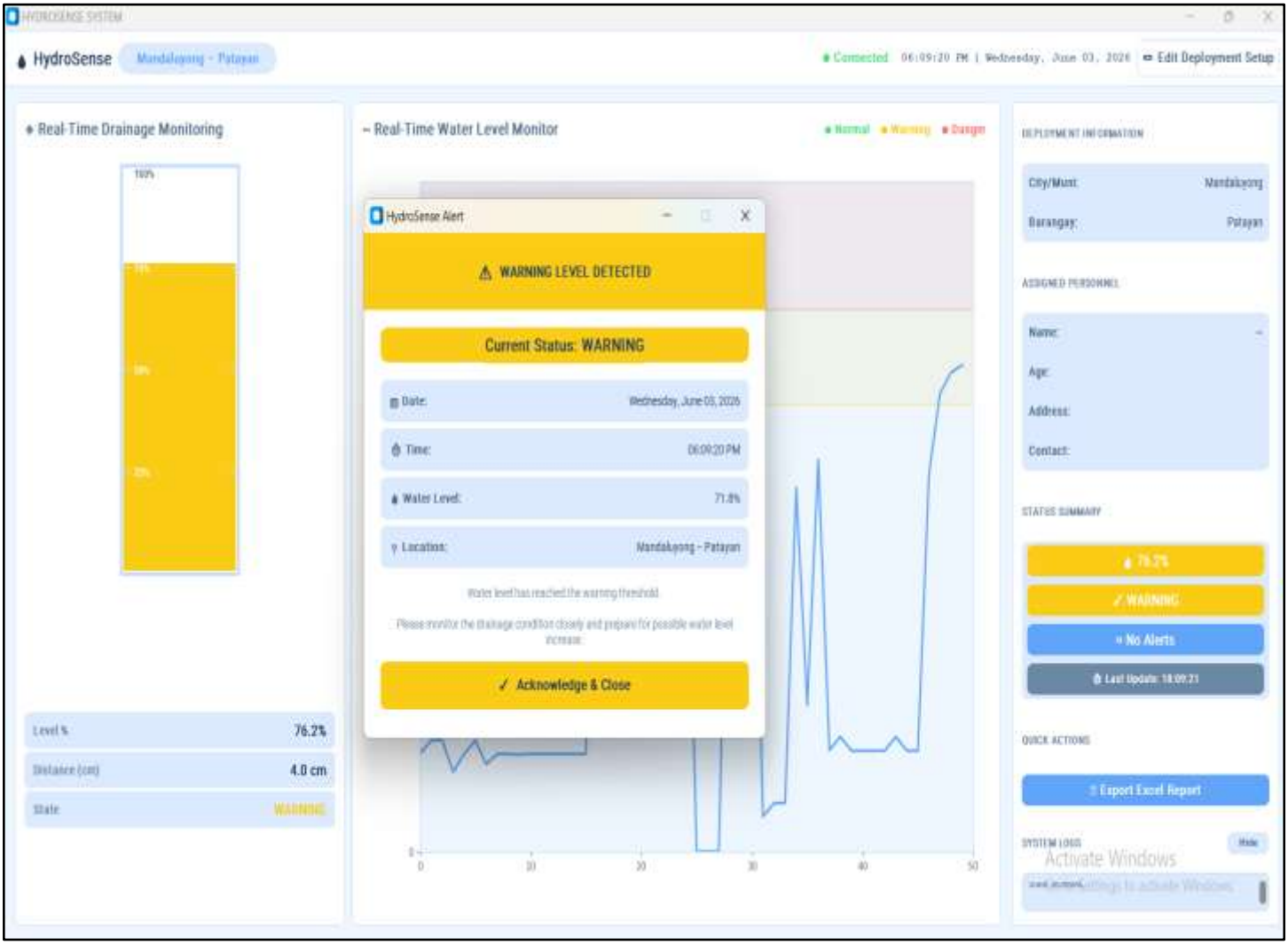


Figure 9. Warning State Interface

Danger State Interface

When water reaches a danger level, simulating an imminent canal overflow, the system triggers the *Danger State Interface* (Figure 10). The software interface flashes bright Red while simultaneously transmitting a low-level command string back over the network communication link. This software action forces the physical ESP32 node to energize its passive buzzer and Red LED array. This rapid dual-alert mechanism provides an undeniable audible and visual indicator that the community drainage line has reached a danger flood state.

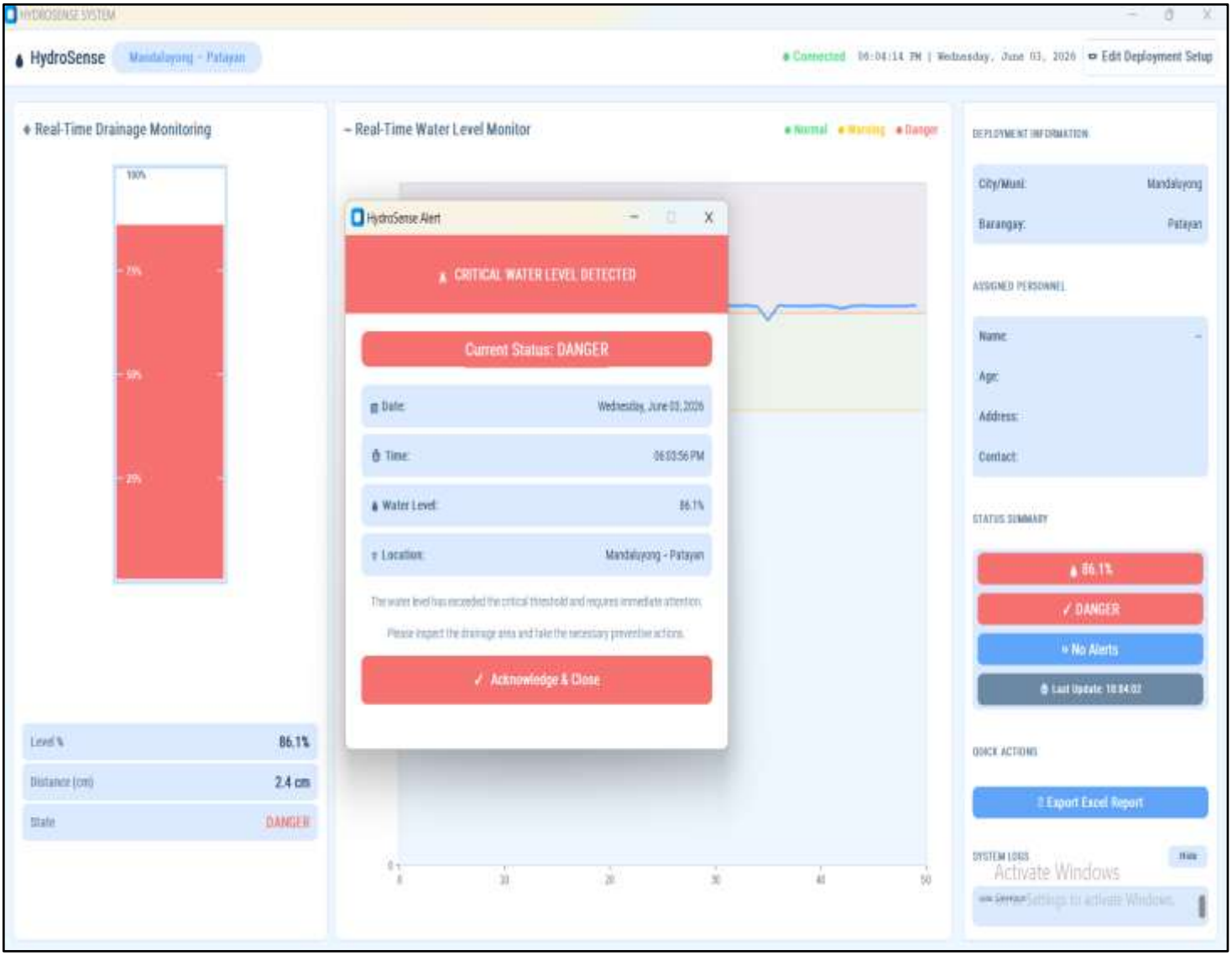
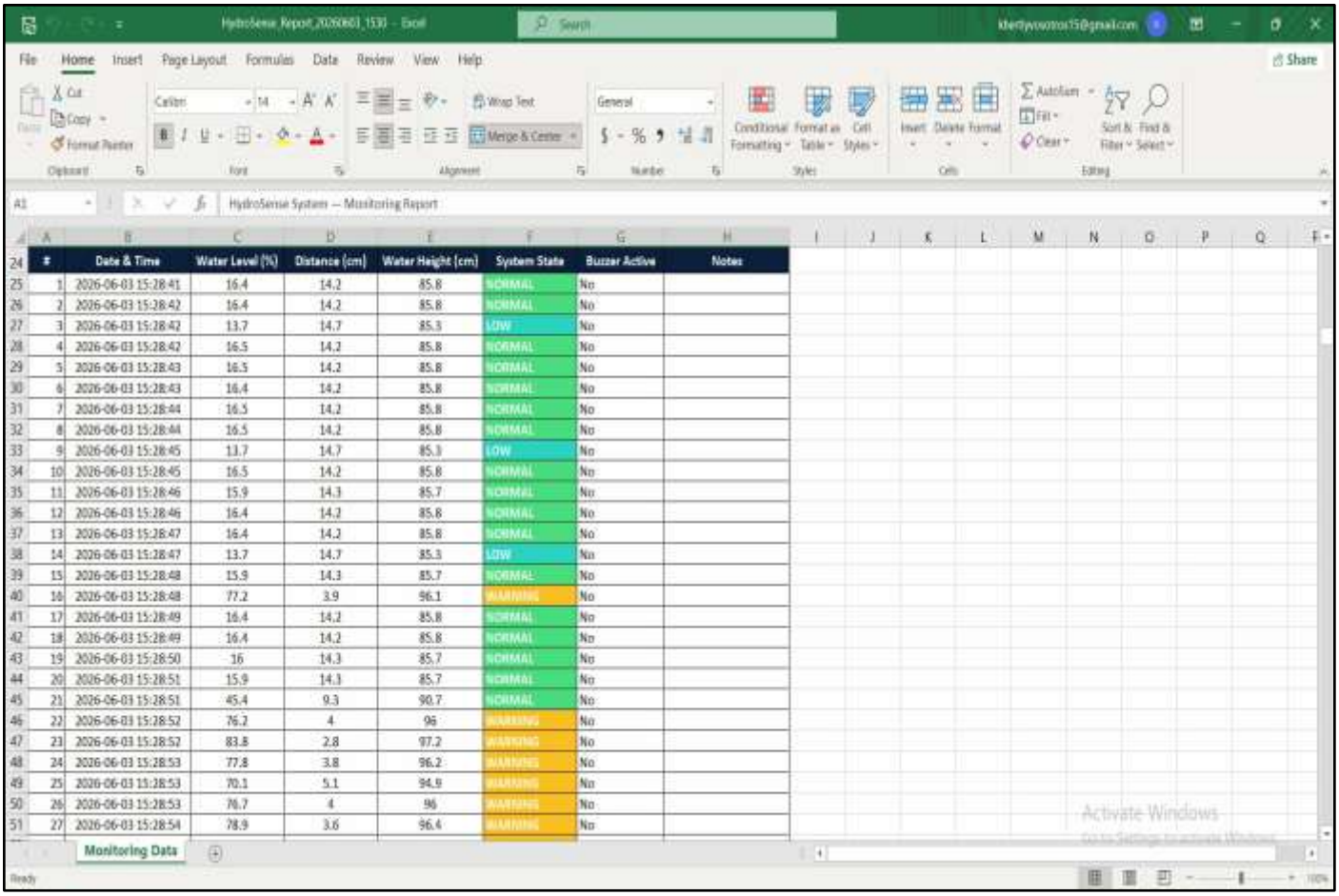


Figure 10. Danger State Interface

Historical Records and Excel Logging

The final module of the application focuses on automated data preservation through the *Historical Records and Excel Logging Interface* (Figure 11). As telemetry streams enter the host computer, the background engine instantly logs and appends each packet into a local Excel-compatible comma-separated value (.csv) spreadsheet file. The logging engine automatically formats the columns into structured fields: *Timestamp, Location, Personnel, Water Level (cm), and Status Classification*. Testing verified that even during rapid state changes, the spreadsheet writes immediately without file lock conflicts or data corruption.



#	Date & Time	Water Level (%)	Distance (cm)	Water Height (cm)	System State	Buzzer Active	Notes
1	2026-06-03 15:28:41	16.4	14.2	85.8	NORMAL	No	
2	2026-06-03 15:28:42	16.4	14.2	85.8	NORMAL	No	
3	2026-06-03 15:28:42	13.7	14.7	85.3	LOW	No	
4	2026-06-03 15:28:42	16.5	14.2	85.8	NORMAL	No	
5	2026-06-03 15:28:43	16.5	14.2	85.8	NORMAL	No	
6	2026-06-03 15:28:43	16.4	14.2	85.8	NORMAL	No	
7	2026-06-03 15:28:44	16.5	14.2	85.8	NORMAL	No	
8	2026-06-03 15:28:44	16.5	14.2	85.8	NORMAL	No	
9	2026-06-03 15:28:45	13.7	14.7	85.3	LOW	No	
10	2026-06-03 15:28:45	16.5	14.2	85.8	NORMAL	No	
11	2026-06-03 15:28:46	15.9	14.3	85.7	NORMAL	No	
12	2026-06-03 15:28:46	16.4	14.2	85.8	NORMAL	No	
13	2026-06-03 15:28:47	16.4	14.2	85.8	NORMAL	No	
14	2026-06-03 15:28:47	13.7	14.7	85.3	LOW	No	
15	2026-06-03 15:28:48	15.9	14.3	85.7	NORMAL	No	
16	2026-06-03 15:28:48	77.2	3.9	96.1	WARNING	No	
17	2026-06-03 15:28:49	16.4	14.2	85.8	NORMAL	No	
18	2026-06-03 15:28:49	16.4	14.2	85.8	NORMAL	No	
19	2026-06-03 15:28:50	16	14.3	85.7	NORMAL	No	
20	2026-06-03 15:28:51	15.9	14.3	85.7	NORMAL	No	
21	2026-06-03 15:28:51	45.4	9.3	90.7	NORMAL	No	
22	2026-06-03 15:28:52	76.2	4	96	WARNING	No	
23	2026-06-03 15:28:52	83.8	2.8	97.2	WARNING	No	
24	2026-06-03 15:28:53	77.8	3.8	96.2	WARNING	No	
25	2026-06-03 15:28:53	70.1	5.1	94.9	WARNING	No	
26	2026-06-03 15:28:53	76.7	4	96	WARNING	No	
27	2026-06-03 15:28:54	78.9	3.6	96.4	WARNING	No	

Figure 11. Historical Records and Excel Logging

System Testing

The system testing phase was executed to validate the operational synergy, communication reliability, and overall responsiveness of the integrated hardware-software ecosystem. Testing focused heavily on network synchronization, data transmission accuracy via the shared local area network (LAN), physical sensor calibration under changing water levels, and the dynamic rendering of the *HydroSense* dashboard components.

Network Synchronization and Serial-Over-Wi-Fi Communication

Initial system testing focused on verifying the wireless data pipeline established between the edge-processing hardware and the centralized desktop software. Both the ESP32 microcontroller and the host computer executing the Python application were configured to connect to a unified local Wi-Fi network interface. Testing confirmed that the ESP32 successfully established a stable socket or serial transmission link over the shared network, allowing it to seamlessly broadcast parsed sensor payloads to the *HydroSense* host environment without data packet drops, buffer overflows, or significant latency.

Physical Simulation and Multi-Stage Water Level Testing

To evaluate the calibration and precision of the tracking hardware under real-world conditions, empirical testing was conducted using the glass aquarium simulation tank. Water was manually introduced into the reservoir at varying volume intervals to simulate rising flood heights within a drainage asset.

The HC-SR04 ultrasonic sensor consistently captured the acoustic reflections of the fluid surface at different heights, accurately computing the distance shifts. The hardware successfully mapped these physical variations to the corresponding local alert nodes, triggering the correct sequence of the physical Green, Yellow, and Red LED channels, along with the active buzzer, precisely at the designated threshold boundaries.

Dashboard Functionality and Dynamic UI Interface Validation

After verifying the physical components, the software interface was audited to confirm its visual responsiveness to live environmental data. The testing matrices (detailed in Tables 3 and 4) demonstrated that the HydroSense dashboard flawlessly parsed incoming network packets and adjusted its user interface elements in real time.

When water levels exceeded the predefined limits, the dashboard correctly rendered the corresponding Normal (Green), Warning (Yellow), and Danger (Red) structural themes. All technical information, including live numerical data, warning alerts, and operator logs, appeared correctly on the main console screen, validating the end-to-end functionality of the special project.

Table 3. Water Level Classification Testing

Test Scenario	Water Level Condition	Expected Output	Actual Output	Result
Test 1	Low	Green LED activated	Green LED activated	Passed
Test 2	Medium	Yellow LED activated	Yellow LED activated	Passed
Test 3	High	Red LED + buzzer activated	Red LED + buzzer activated	Passed

Table 3 presents the test results for the water-level classification system. During testing, the HydroSense prototype successfully identified varying water levels and activated the corresponding warning indicators at predefined threshold values. The green indicator was activated during normal conditions, while the yellow and red indicators were activated during warning and danger conditions, respectively. Additionally, the buzzer alarm functioned properly once danger-level conditions were detected.

Table 4. Dashboard Functionality Testing

Function Tested	Expected Behavior	Actual Behavior	Result
Real-time water display	Dashboard updates water level	Water level updated	Passed
Warning classification	Correct status shown	Correct status shown	Passed
Notification display	Warning shown during high level	Notification displayed	Passed
Historical logging	Records saved	Records stored	Passed

Table 4 presents the functional testing results for the HydroSense application across multiple trials. The empirical findings demonstrate that the system achieved 100% success in dynamically updating the graphical user interface, computing accurate water levels, and executing automated data logging sequences without system latency.

Summary of Testing Results

Based on the conducted testing procedures, the HydroSense prototype successfully performed its intended function of monitoring drainage canal water levels. The system consistently classified water levels into **Normal, Warning, and Danger** conditions, activated corresponding warning indicators, and updated the HydroSense dashboard in real time. Furthermore, the historical monitoring feature successfully recorded drainage canal conditions for monitoring purposes. These findings demonstrate the functionality of the proposed prototype for barangay-level monitoring of drainage canal water levels.

Hardware Fabrication and Testing

The hardware assembly was completed using a **soldering iron and high-conductivity soldering wire** to ensure permanent, low-resistance connections between the ESP32 development board, the sensor modules, and the power management sub-circuit. Prior to deployment, a **digital multimeter was used to perform continuity testing, verify the LM2596S buck converter's output voltage, and check the ESP32's current draw**. This

diagnostic process ensured that the power rails did not exceed the microcontroller's operational limits, mitigating the risk of component burnout during long-duration simulations.

Problems Encountered and Solutions

Problems	Solutions
Hardware Integration and Wire Management: The initial assembly of the ESP32, sensors, and breadboard resulted in a cluttered wiring layout, affecting the overall system appearance and structural stability.	The wiring connections were systematically reorganized and routed through the rear portion of the prototype to ensure a neater, more stable, and well-organized presentation.
Sensor Responsiveness and Threshold Tuning: The ultrasonic sensor occasionally experienced calibration errors, resulting in inconsistent triggering of the LED indicators and the buzzer alarm during testing.	Multiple calibration tests were conducted to establish precise, consistent distance thresholds for the Normal, Warning, and Danger water level classifications.
Software Development Complexity: Integrating real-time visualization, historical dataset recording, and hardware communication within the Python-based monitoring application posed significant programming difficulties.	Additional technical documentation, open-source forums, and systematic debugging methods were utilized to refine the script and resolve data transmission errors.
Component Damage and Hardware Constraints: Hardware malfunctions, such as damaged microcontrollers and burned LEDs from extended operational testing, caused unexpected delays.	Defective items were replaced immediately, and protective resistors were added to the circuit to regulate current and safeguard the active elements.

DISCUSSION OF FINDINGS

The empirical testing matrices and operational evaluations of the Solar-Powered IoT-Based Barangay Drainage Canal Water Level Monitoring System demonstrate the successful integration of hardware and software components. During testing, the ESP32 microcontroller and the HydroSense application maintained stable communication through a shared local Wi-Fi network. Sensor data was transmitted continuously at 115200 baud, enabling the HydroSense application to receive and process water-level information in real time. The system successfully displayed monitoring data, updated warning classifications, and recorded historical logs throughout the testing procedures.

Physical testing conducted using the glass aquarium simulation model showed that the HC-SR04 ultrasonic sensor consistently detected changes in water level and transmitted the corresponding measurements to the ESP32 microcontroller. Based on the predefined threshold values, the system correctly classified water levels into Normal, Warning, and Danger conditions. The associated visual indicators (green, yellow, and red LEDs) and the audible buzzer were activated based on the detected water level. The synchronization between the hardware indicators and the HydroSense dashboard demonstrates the system's capability to provide real-time monitoring and warning notifications under simulated drainage canal conditions.

Furthermore, the solar-powered architecture successfully supplied power to the monitoring system through the integration of a solar panel, rechargeable batteries, a TP4056 charging module, and an LM2596S buck converter. The power management subsystem provided a stable voltage supply for the ESP32 and connected components during testing. Implementing a localized monitoring platform also reduces reliance on complex cloud infrastructure and enables barangay personnel to access monitoring information via a dedicated desktop application. These findings suggest that the proposed prototype can serve as a practical, cost-effective solution for community-level drainage canal water-level monitoring and early warning applications.

CONCLUSION

The researchers successfully designed, fabricated, and validated the Solar-Powered IoT-Based Barangay Drainage Canal Water Level Monitoring and Overflow Warning Simulation System. By leveraging an ESP32 microcontroller, an HC-SR04 ultrasonic sensor, and a custom Python Tkinter application, the project achieved

a unified, low-latency, and highly responsive local disaster risk mitigation environment. The system effectively eliminates common barriers to community technology deployment by utilizing a standalone executable application structure that allows local barangay officials to quickly initialize, customize, and operate the flood tracking network without technical specialized programming knowledge.

Empirical testing across simulated water level cycles confirmed that the hardware and software communicate reliably over a shared Wi-Fi network. The system maintains total reading accuracy, properly logging and appending timestamped information into structured local Excel spreadsheets for long-term data archiving.

Additionally, the implementation of an independent solar charging track and an adjustable LM2596S buck converter ensures that the field edge node remains power-autonomous, grid-independent, and protected from voltage surges.

Ultimately, this special project proves that combining sustainable energy harvesting with localized IoT architecture creates an affordable, robust, and dependable early-warning flood management system that can significantly improve local disaster preparedness and community safety.

REFERENCES

1. Al-Shareeda, M. A., Ali, A. M., Hammoud, M. A., Kazem, Z. H. M., & Hussein, M. A. (2025). Secure IoT-based real-time water level monitoring system using ESP32 for critical infrastructure. Journal of Cyber Security and Risk Auditing, 2025(2), 44–52. https://doi.org/10.63180/jcsra.thestap.2025.2.4
2. Bagal, S. A., Bhisikar, T., Nimje, A., Nandanwar, Y., & Patil, T. (2023). Design of automatic domestic water quality monitoring & distribution control. International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering, 11(4). https://doi.org/10.17148/IJIREEICE.2023.11405
3. Bowler, A. L., Bakalis, S., & Watson, N. J. (2022). A review of in-line and on-line measurement technologies to support process monitoring and control in food manufacturing. Foods, 11(16), 2294. https://doi.org/10.3390/foods11162294
4. Cadeliña, M. R. L.,&Perin, M. A. D. (2022). Flood alarm: Arduino-based water level indicator with buzzer for flood warning purposes.
5. Castro Garcia, L., Wang, J., Wang, S., Qiao, X., Kuhl, S., Zhang, H., Rangel de la Tejera, R., & Reed, D. (2025). I-Canopy: A resilient IoT platform for adaptive environmental monitoring at the edge. Smart Agricultural Technology, 12, 101611. https://doi.org/10.1016/j.atech.2025.101611
6. Diriyana, A., Darusalam, U., & Natasha, N. D. (2019). Water level monitoring and flood early warning using microcontroller with IoT based ultrasonic sensor. Jurnal Teknik Informatika C.I.T Medicom, 11, 22–28. https://doi.org/10.35335/cit.Vol11.2019.9.pp22-28
7. Fernandes, J. A. T., Loureiro, G. F., & Giovanelli, L. B. (2024). Development of a prototype for water reservoir level monitoring using an ESP32 board. International Journal of Geoscience, Engineering and Technology, 10(1), 46–53. https://doi.org/10.70597/ijget.v10i1.545
8. Hanan, Gunawan, A. A. N., & Sumadiyasa, M. (2019). Water level detection system based on ultrasonic sensors HC-SR04 and ESP8266-12 modules with Telegram and buzzer communication media. Instrumentation Mesure Métrologie, 18(3). https://doi.org/10.18280/i2m.180311
9. Lei, J.-F. (2010). Application of water-level monitoring system to modern management of drainage system. China Water&Wastewater, 26(18), 145–147.
10. Lengkong, A. G., Salaki, D. T.,&Alfonsius, E. (2024). Implementation of an Internet of Things (IoT)-based water level early warning system with Telegram notification. https://doi.org/10.33365/jatika.v6i2.349
11. Lias, J., & Kumaran, G. (2024). IoT-based flood monitoring system using ESP32. Evolution in Electrical and Electronic Engineering.
12. Nur, I., & Shrestha, K. K. (2017). An integrative perspective on community vulnerability to flooding in cities of developing countries. Procedia Engineering, 198, 958–967. https://doi.org/10.1016/j.proeng.2017.07.141

13. Oluremi, D., Chinedu, P., & Adeyemo, D. O. (2025). Optimizing solar energy harvesting in IoT nodes using deep self-organizing maps. ResearchGate Preprint.
14. Perumal, P., Raj, A., Bharathi, B., Raju, G. M., & Yogeswari, K. (2017). UAV assisted automated remote monitoring and control system for smart water bodies. In 2017 Second International Conference on Recent Trends and Challenges in Computational Models (ICRTCCM) (pp. 269–273). IEEE. https://doi.org/10.1109/ICRTCCM.2017.40
15. Rahman, M. H., Shihab, M., Rahman, M. A., & Naderuzzaman, M. (2026). ESP32 microcontroller: A review of architecture, communication protocols, applications and research challenges. Open Access Journal on Engineering Applications, 1(2), 19–33. https://doi.org/10.64886/oajea.0102.00
16. Sahani, S., et al. (2026). A high-efficiency inductor-less solar energy harvesting system for IoT end nodes. Results in Engineering. https://doi.org/10.1016/j.rineng.2026.110342
17. Salman, K., Ahmed, S. T., Acharya, T., Annamalai, A., & Chouikha, M. (2026). Robust smart charging framework for resilience IoT deployments in remote environments. In 2025 IEEE Symposium on Wireless Technology&Applications (ISWTA). https://doi.org/10.1109/ISWTA68114.2025.11329928
18. Varela, B. D., Tan, S. M., Jr., Jao, E. B., Jr., Espiña, M. B., & De Asis, C. A. (2022). Assessment of drainage conditions in the Poblacion barangays of Catarman, Northern Samar: A pilot study. International Journal of Environment and Climate Change, 12(10), 1337–1345. https://doi.org/10.9734/ijecc/2022/v12i1031156
19. Velliangiri, A. (2025). Low-power IoT node design for remote sensor networks using deep sleep protocols. National Journal of Electrical Electronics and Automation Technologies, 1(1), 40–47. https://doi.org/10.17051/JEEAT/01.01.05
20. Zhmud, V., Trubin, V., Dimitrov, L., & Karpov, E. (2018). Educational robotics: From simple sensors to complex robotic systems. Proceedings of the International Conference on Robotics and Automation.
21. Zulhakim, A. M., Abdullah, W. F. H., Abdul Halim, I. S., Haji Mamat, R., Muslan, M. I. A., & Ahmad, N. S. (2023). Smoothing sensor data in a controlled IoT framework with moving averages. In 2023 IEEE Regional Symposium on Micro and Nanoelectronics (RSM) (pp. 153–156). IEEE. https://doi.org/10.1109/RSM60397.2023.1049754