IoT Water Level Monitoring System - Engineering Report

Role: Embedded Systems Architect

Date: October 26, 2023

Version: 1.1

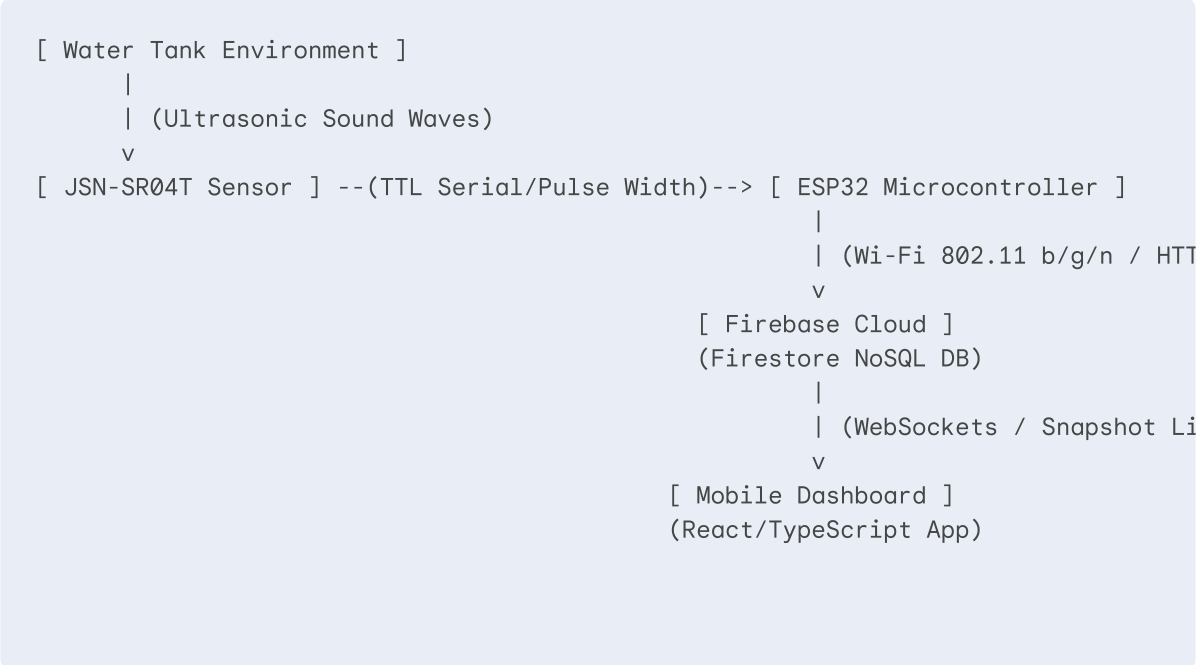
1. Executive Summary

This document outlines the detailed architecture for a robust, low-cost, real-time water level monitoring system designed for residential and industrial applications. The system utilizes an ESP32 microcontroller for high-performance edge processing and wireless connectivity, paired with an industrial-grade ultrasonic sensor for non-contact measurement in corrosive or humid environments. Data synchronization is handled by Google Firebase, providing low-latency updates to a Progressive Web App (PWA). This architecture ensures that users receive instant visual feedback and alerts on any mobile device without the need for native app installation, prioritizing reliability, scalability, and ease of maintenance.

2. System Architecture

2.1 High-Level Block Diagram

The system follows a standard IoT edge-to-cloud topology. The sensor acts as the input node, the ESP32 as the gateway/processor, and the mobile device as the client.



2.2 Data Flow & Processing Pipeline

1. **Acquisition:** The ESP32 triggers the JSN-SR04T ultrasonic sensor every 2 seconds. This interval balances real-time responsiveness with power efficiency, preventing sensor overheating and reducing CPU load.

2. **Signal Processing:** The ESP32 measures the pulse width of the 'Echo' signal to calculate the time-of-flight (ToF) of the sound wave. The distance is derived using the speed of sound in air (~343 m/s), adjusted for the 2x travel distance (down and up).
3. **Noise Filtering:** Raw ultrasonic data often contains "jitter" caused by water ripples during filling or condensation on the sensor head. A **Median Filter** collects 5 consecutive readings, sorts them, and selects the middle value. This effectively eliminates statistical outliers without the lag introduced by simple averaging.
4. **Normalization:** The physical distance (cm) is mapped to a logical percentage (0-100%) based on the specific tank geometry. This abstraction allows the frontend application to remain agnostic of the physical tank size; it simply displays a percentage.
5. **Transmission Strategy:** To minimize bandwidth and database write costs, data is only pushed to Firestore if the value changes by more than a **2% Deadband threshold** or if a "heartbeat" interval (e.g., 5 minutes) has passed.
6. **Visualization:** The Mobile App utilizes Firestore `onSnapshot` listeners. Unlike traditional REST polling (which wastes battery by asking "is there new data?" repeatedly), this WebSocket connection pushes data to the phone immediately upon change, triggering an instant UI re-render.

3. Hardware Requirements (BOM)

Component	Specification	Detailed Reason for Selection	Est. Cost
Microcontroller	**ESP32-WROOM-32**	Selected for its dual-core Xtensa LX6 processor (allowing Wi-Fi stack to run separately from sensor logic), 12-bit ADC, and deep-sleep capabilities for future battery operation.	$5.00
Sensor	**JSN-SR04T** (Waterproof)	**Critical Selection:** Unlike the hobbyist HC-SR04, this sensor features a detached, hermetically sealed probe. It resists corrosion from humidity and condensation inside closed tanks, ensuring long-term survivability. Note: It has a minimum blind zone of ~20cm.	$8.00
Power Supply	5V 1A Adapter / 18650 Li-Ion	The ESP32 can draw significantly current peaks (>250mA) during Wi-Fi transmission. A stable 1A supply prevents brownout resets.	$3.00
Enclosure	IP65 Rated Box	Essential for housing the non-waterproof control board (ESP32 + Sensor PCB). Protects against rain, dust, and insects.	$4.00
Buck Converter	LM2596 (Optional)	High-efficiency step-down module. Required if the power source is a 12V/24V solar battery system, converting it to a stable 5V with minimal heat loss.	$1.50

3.1 Connection Diagram (Wiring)

- **JSN-SR04T VCC** -> ESP32 VIN (5V) - *Must be 5V, 3.3V is insufficient for the sensor transducer.*
- **JSN-SR04T GND** -> ESP32 GND
- **JSN-SR04T TRIG** -> ESP32 GPIO 5 - *Output pin to initiate measurement.*
- **JSN-SR04T ECHO** -> ESP32 GPIO 18 - *Input pin to read pulse width.*

4. Firmware Logic (Edge Computing)

The firmware is architected using a non-blocking state machine approach. Instead of using `delay()`, which halts the processor, we use `millis()` timers. This ensures the microcontroller can maintain the Wi-Fi stack background tasks (like keeping the connection alive) while waiting for sensor readings.

Core Algorithm:

1. **Connectivity Check:** Continuously monitor Wi-Fi status. If disconnected, enter a reconnection routine without blocking the main loop.
2. **Trigger Sequence:** Send a 10µs HIGH pulse to the TRIG pin to activate the piezoceramic transducer.
3. **Echo Capture:** Use interrupts or `pulseIn()` to measure the duration the ECHO pin stays HIGH.
4. **Distance Calculation:** `Distance (cm) = Duration (µs) * 0.0343 / 2`.
5. **Data Sanitization:** Apply the median filter. Clamp values to defined `TANK_HEIGHT` limits to prevent negative percentages or values >100% due to sensor glitches.
6. **Conditional Upload:** Compare the new percentage with the last uploaded value.
   - *If* `abs(New - Old) > 2%`: Upload immediately.
   - *If* `TimeSinceLastUpload > 5 mins`: Upload "Heartbeat" to confirm system health.

5. Software/Cloud Requirements

- **Protocol:** HTTPS (REST) for initial auth, followed by persistent connections via the Firebase SDK (gRPC/WebSockets).
- **Backend:** Firebase Firestore (NoSQL).
  - *Collection Structure:* `artifacts/{appId}/public/data`
  - *Document:* `water_level`
  - *Fields:*
    - `level` (number): The normalized percentage.
    - `timestamp` (serverTimestamp): Used for historical graphing.
    - `status` (string): "Normal", "Low", "Critical".
- **Frontend:** React (TypeScript) styled with Tailwind CSS. The app is responsive, functioning as a dashboard on desktops and a touch-friendly app on mobiles.

6. Implementation Plan (Step-by-Step)

Phase 1: Bench Testing

1. **Wiring:** Connect the ESP32 and JSN-SR04T on a breadboard. Ensure common ground between the 5V sensor supply and the 3.3V ESP32.
2. **Firmware Flash:** Flash the firmware with `Serial.begin(115200)`.
3. **Verification:** Point the sensor at a flat object (box/wall). Move the object and verify the Serial Monitor readings match a tape measure. *Note: The JSN-SR04T cannot read distances closer than 20cm (Blind Zone).*

Phase 2: Calibration

1. **Total Depth Measurement:** Measure the distance from the intended sensor mounting point to the bottom of the empty tank (e.g., 200cm).
2. **Offset Measurement:** Determine the "Full" water line. Because of the 20cm blind zone, the sensor must be mounted at least 20cm above the maximum water level.
3. **Configuration:** Input these values into the Firmware constants (`TANK_HEIGHT_CM`, `SENSOR_GAP_CM`) to ensure 0% and 100% are calculated correctly.

Phase 3: Enclosure & Deployment

1. **Lid Preparation:** Drill a 20-22mm hole in the tank lid. The JSN-SR04T probe screws in like a bolt.
2. **Sealing:** Mount the probe. Ensure the wire passes through a waterproof cable gland into the IP65 box containing the ESP32. This prevents moisture from wicking down the wire into the electronics.
3. **Power:** Route the USB cable or power wires through a separate gland.
4. **Final Test:** Power up and monitor the dashboard while manually lifting the float or simulating water level changes.

7. Scalability & Future Improvements

1. **Multi-Tank Fleet Management:** Update the data schema to include a `tank_id` property. The Dashboard can then be upgraded to a "Fleet View," allowing a facility manager to monitor dozens of tanks on a single screen.
2. **Pump Automation (Hysteresis):** Integrate a 5V Relay Module. Implement software hysteresis to prevent rapid toggling:
    - *Turn ON* when level drops below 20%.
    - *Turn OFF* only when level reaches 90%.
    - *Safety:* Add a max-runtime timer to cut power if the pump runs for >2 hours (indicating a leak or sensor failure).

3. **LoRaWAN Integration:** For agricultural deployments lacking Wi-Fi, replace the communication layer with LoRaWAN (using an SX1276 chip). This allows data transmission over 5-10km to a central gateway, albeit with lower data frequency to adhere to duty cycle regulations.