Smart Tank Monitoring System

IoT Assignment #03

El Berni Karim   Fabbri Luca   Dellasantina Luca

February 5, 2026

Abstract

The Smart Tank Monitoring System is a modular IoT solution designed to monitor rainwater levels in a tank and control a water channel valve. The system operates in two main modes: AUTOMATIC (software-controlled) and MANUAL (operator-controlled). It integrates four subsystems: an ESP32-based monitoring unit (TMS), an Arduino-based actuator unit (WCS), a Central Control Unit (CUS), and a Web Dashboard (DBS), communicating via MQTT, Serial, and HTTP protocols.

Contents

1 System Overview 3

2 System Architecture 3

2.1 1. Tank Monitoring Subsystem (TMS) . . . 3

2.2 2. Water Channel Subsystem (WCS) . . . 4

2.3 3. Control Unit Subsystem (CUS) . . . 4

2.4 4. Dashboard Subsystem (DBS) . . . 5

3 Control Logic and FSMs 6

3.1 Mode Management . . . 6

3.2 Automatic Control Policy . . . 6

3.3 Finite State Machines (FSM) . . . 6

3.3.1 TMS FSM (ESP32) . . . 6

3.3.2 WCS FSM (Arduino) . . . 6

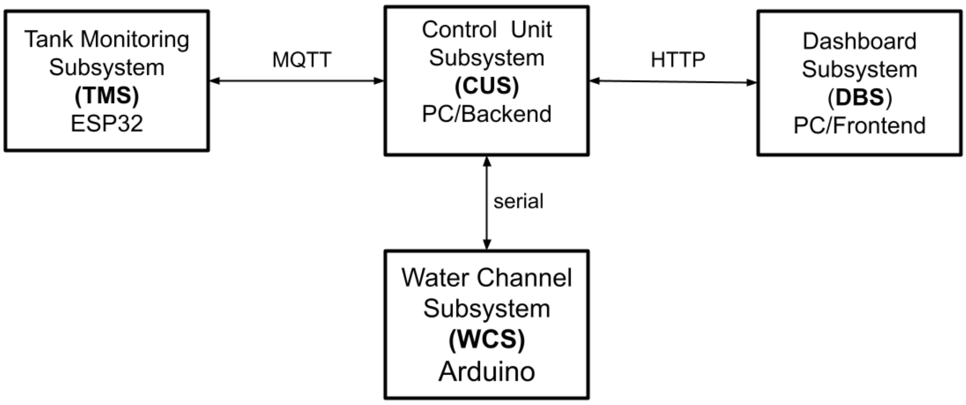
4 Hardware Implementation 7

4.1 TMS Connections (ESP32) . . . 7

4.2 WCS Connections (Arduino UNO) . . . 7

1 System Overview

The system creates a distributed architecture for tank monitoring and control. It addresses the requirement of managing water levels to prevent overflow or dry states by controlling a discharge valve.


Tank Monitoring Subsystem (TMS) ESP32
MQTT
Control Unit Subsystem (CUS) PC/Backend
HTTP
Dashboard Subsystem (DBS) PC/Frontend
serial
Water Channel Subsystem (WCS) Arduino

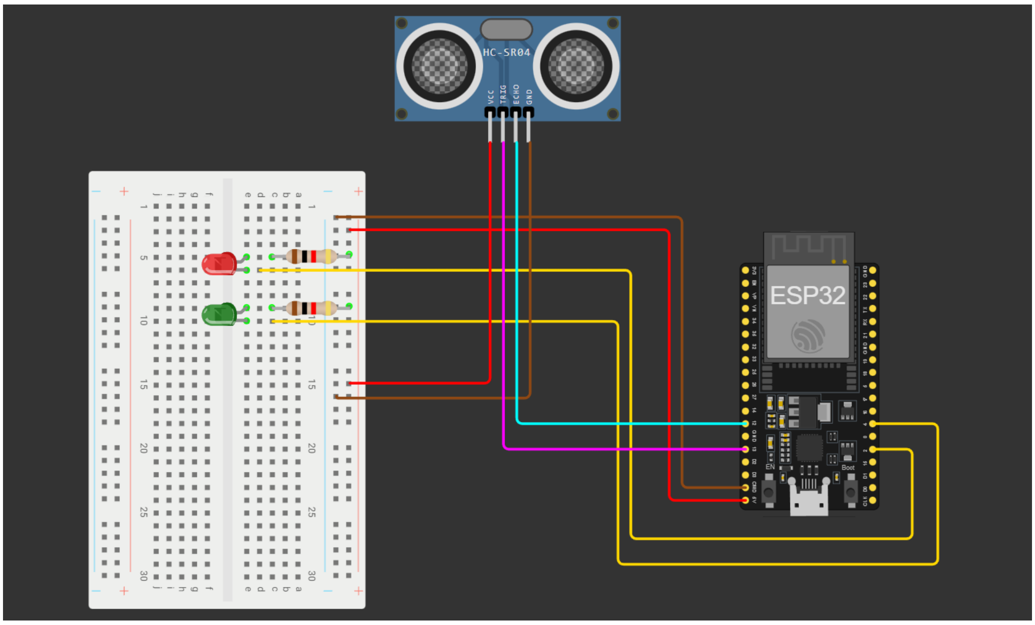
The core functionalities are:

- **Monitoring**: Continuous reading of water levels using sonar.
- **Control**: Automated valve actuation based on defined logic policies or manual override.
- **Visualization**: Real-time dashboard for remote supervision.
- **Alerting**: Visual indicators (LEDs) and system states (Warning, Alarm).

2 System Architecture

The system is divided into four main subsystems:

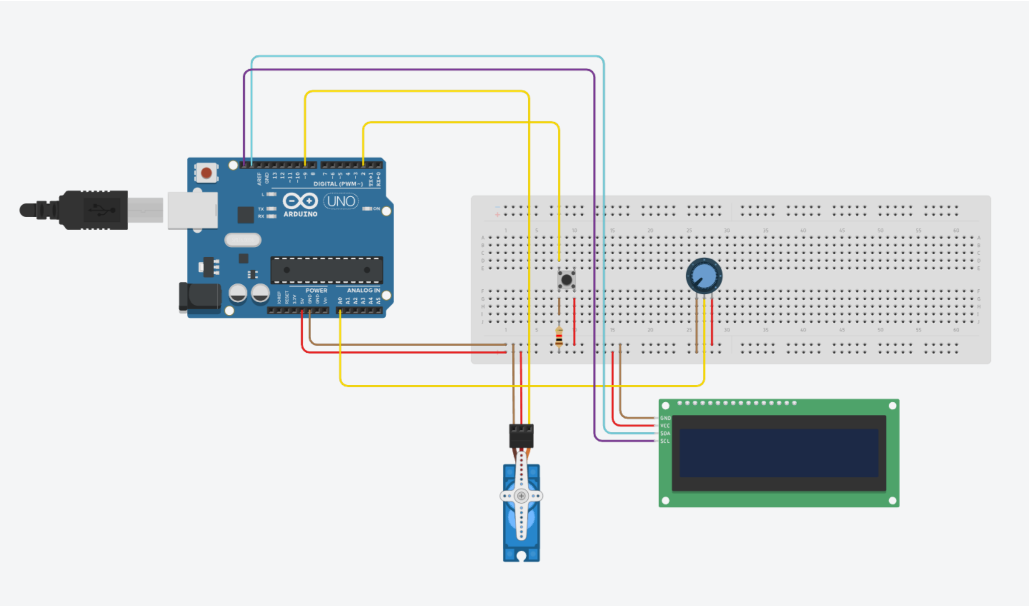
2.1 1. Tank Monitoring Subsystem (TMS)



Hardware: ESP32 based microcontroller.
Role: Captures environmental data and handles network connectivity.
Functionality:

- Measures water level using an HC-SR04 sonar sensor.
- Publishes data to the `tank/level` MQTT topic.
- Indicates connection status via Green (Connected) and Red (Error) LEDs.

2.2 2. Water Channel Subsystem (WCS)



Hardware: Arduino UNO.
Role: Physical actuation and local user interface.
Functionality:

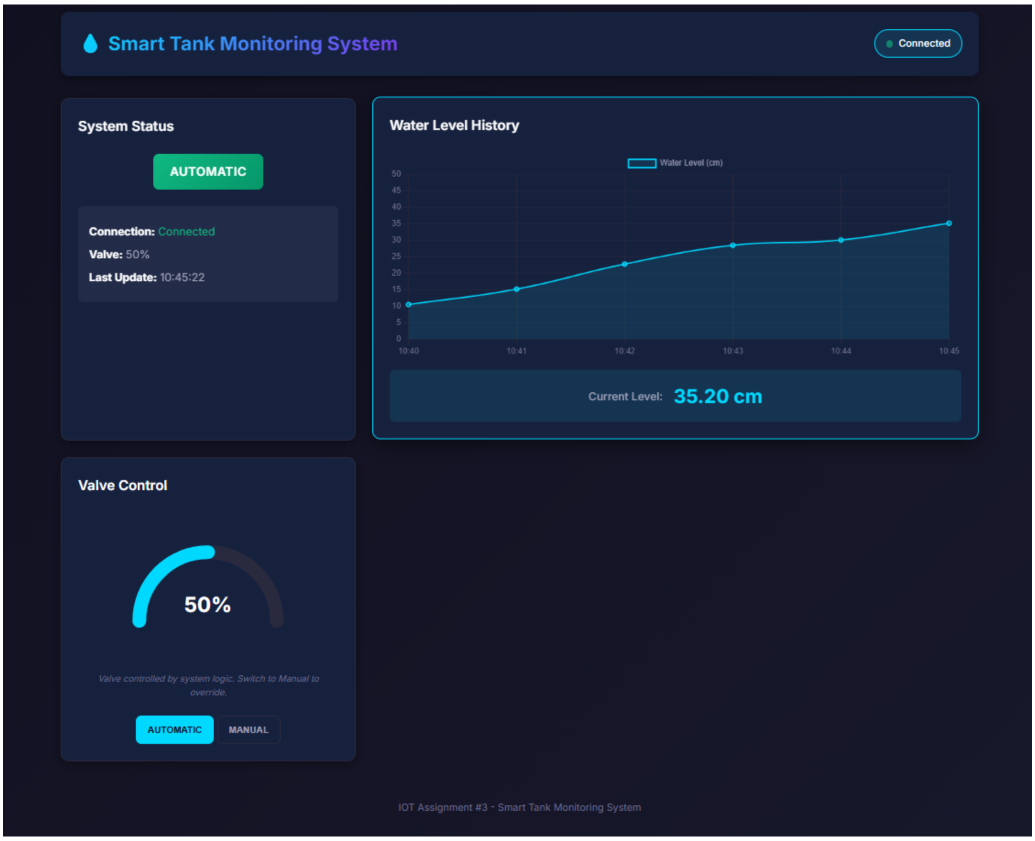
- Controls the servo motor for valve opening (0° to 180° mapped to 0-100%).
- Provides local manual control via a Potentiometer.
- Displays current status on an LCD screen.
- Communicates with the CUS via Serial (JSON protocol).

2.3 3. Control Unit Subsystem (CUS)

Software: Java Application running on a PC/Server.
Role: The brain of the system, implementing the control logic.
Functionality:

- Acts as a bridge between MQTT (TMS), Serial (WCS), and HTTP (DBS).
- Implements the automatic control policy based on thresholds $L1, L2$ and times $T1, T2$.
- Manages the global system state.

2.4 4. Dashboard Subsystem (DBS)



Connection: Connected
Valve: 50%
Last Update: 10:45:22

Current Level: 35.20 cm

Valve controlled by system logic. Switch to Manual to override.

IOT Assignment #3 - Smart Tank Monitoring System

Software: Web Application (HTML/CSS/JS).
Role: Remote monitoring and control interface.
Functionality:

- Visualizes real-time water level graphs.
- Allows mode switching (Automatic ↔ Manual).
- Provides remote manual control of the valve.

3 Control Logic and FSMs

3.1 Mode Management

The system operates in two mutually exclusive modes:

- **AUTOMATIC**: The CUS determines the valve opening based on sensor readings.
- **MANUAL**: The user controls the valve setting via the Potentiometer (local) or the Dashboard (remote).

A timeout mechanism ensures safety: if the TMS stops sending data for $T2$ seconds, the system enters an **UNCONNECTED** state.

3.2 Automatic Control Policy

The automatic logic uses two water level thresholds ($L_1 < L_2$) and a timing threshold (T_1).

Condition	**Valve Action**	**Description**
$Level < L_1$	**0% (Closed)**	Normal operation.
$L_1 < Level < L_2$	**50% (Half)**	*Warning state.* Valve opens after duration $\geq T_1$.
$Level \geq L_2$	**100% (Open)**	*Alarm state.* Valve opens immediately.

Table 1: Automatic Control Logic Table

3.3 Finite State Machines (FSM)

3.3.1 TMS FSM (ESP32)

- `STATE_INITIALIZING`: Hardware setup.
- `STATE_CONNECTING_WIFI`: Connecting to WiFi network.
- `STATE_CONNECTING_MQTT`: Connecting to the MQTT broker.
- `STATE_CONNECTED`: Normal operation, sensing and publishing.
- `STATE_NETWORK_ERROR`: Fallback state on connection failure.

3.3.2 WCS FSM (Arduino)

- `MODE_UNCONNECTED`: Safe state, waiting for CUS heartbeat.
- `MODE_AUTOMATIC`: Actuator follows CUS commands.
- `MODE_MANUAL`: Actuator follows Potentiometer/Dashboard input.

4 Hardware Implementation

4.1 TMS Connections (ESP32)

- **Sonar Trigger**: GPIO 13
- **Sonar Echo**: GPIO 12
- **Green LED**: GPIO 2
- **Red LED**: GPIO 4

4.2 WCS Connections (Arduino UNO)

- **Servo Motor**: Pin 9 (PWM)
- **Potentiometer**: Pin A0 (Analog Input)
- **Button**: Pin 2 (Digital Input with Interrupt)
- **LCD Display**: I2C Bus (SDA=A4, SCL=A5)