**Smart Water Management System**

**Design Document**

**Version 1.1**



**Group Id: F2302894E5**

**Student Name: Usama Ahmad**

**Supervisor Name:** **Dr. Israr Ullah**

**Revision History**



|  |  |  |  |
| --- | --- | --- | --- |
| Date (dd/mm/yyyy) | Version | Description | Author |
| 18//01/2024 | 1.1 | This IoT-based project aims to develop a smart water management system that automates the monitoring, filling, and pump operation process of a home water tank. The system utilizes sensors to continuously monitor the water level and automatically start and stop the water pump based on predefined thresholds. A companion mobile app allows users to monitor the current water level and manually control the pump. | BC200406015 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

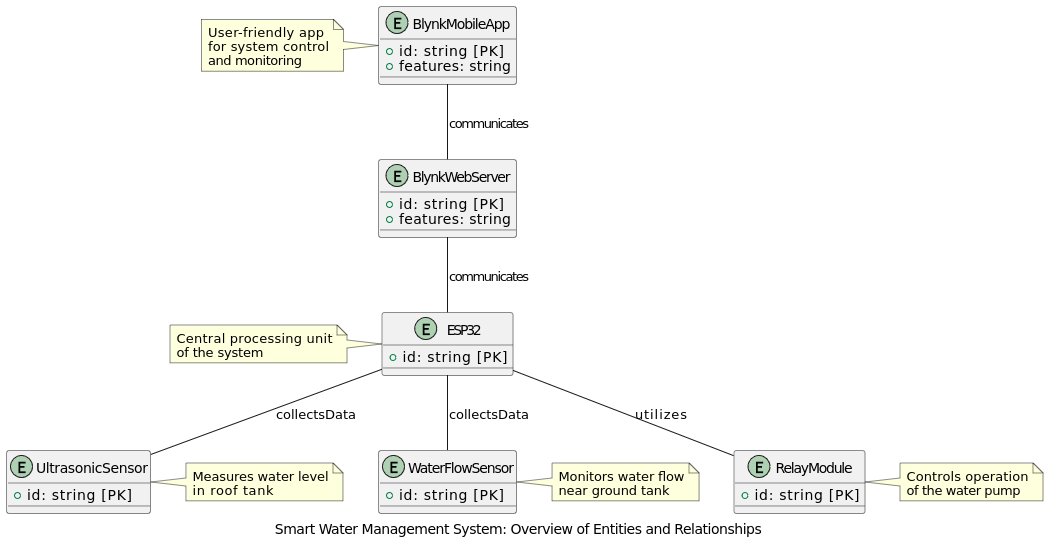
**Table of Contents**

1. [Introduction of Design Document](#One)
2. [Entity Relationship Diagram (ERD)](#ERD)
3. [Sequence Diagrams](#Six)
4. [Architecture Design Diagram](#Seven)
5. System State Machine
6. Circuit Design
7. [Interface Design](#interfacedesign) for Mobile App
8. [Test Cases](#testcases)

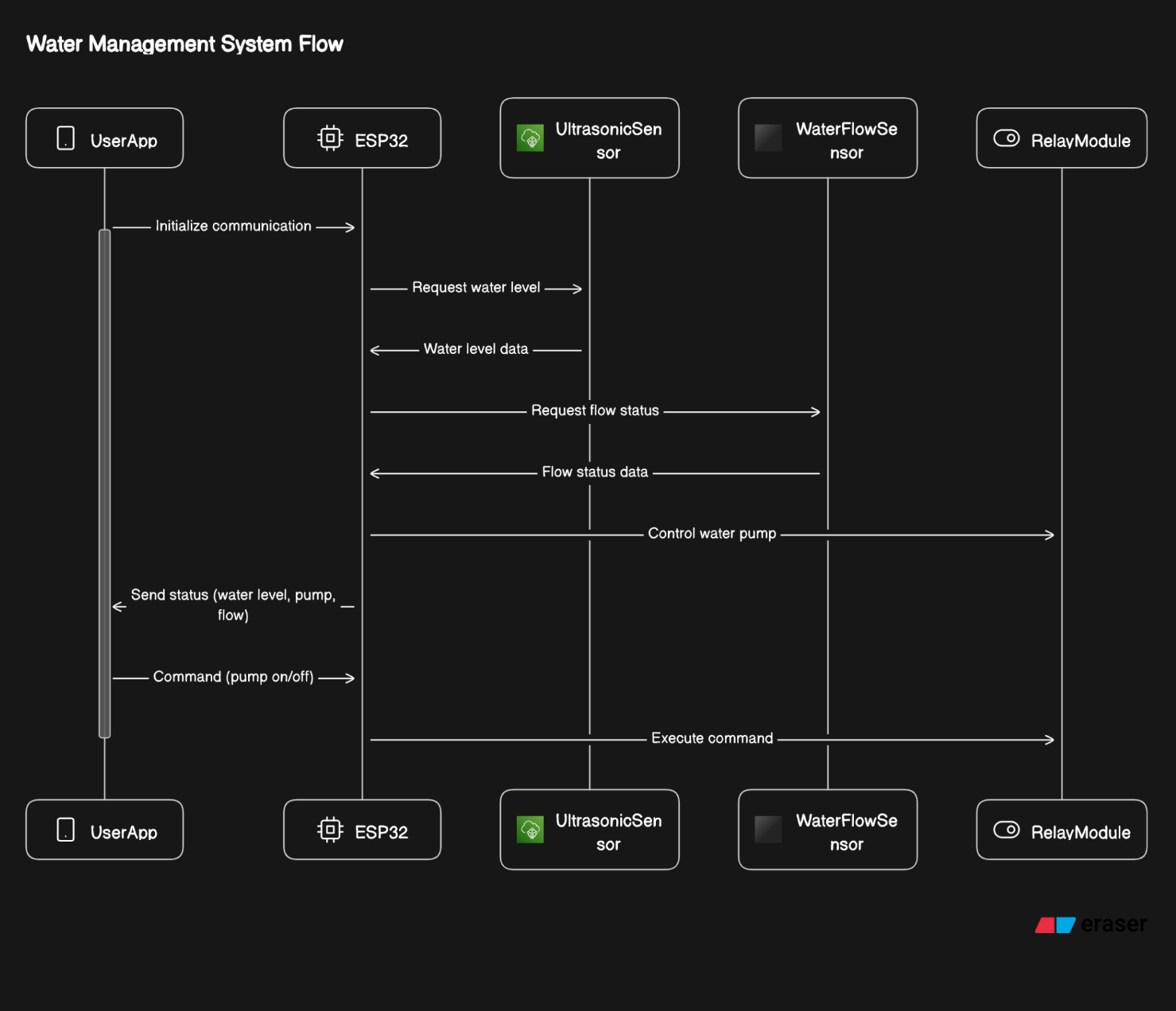
1. Introduction of Design Document

This Design Document outlines the architecture, specifications, and requirements for the Smart Water Management System (SWMS). SWMS is an IoT-based solution designed to automate water tank monitoring and pump operation, with the goal of improving water conservation and convenience in households. This document provides detailed information about the system components, requirements, tools, and expected deliverables.

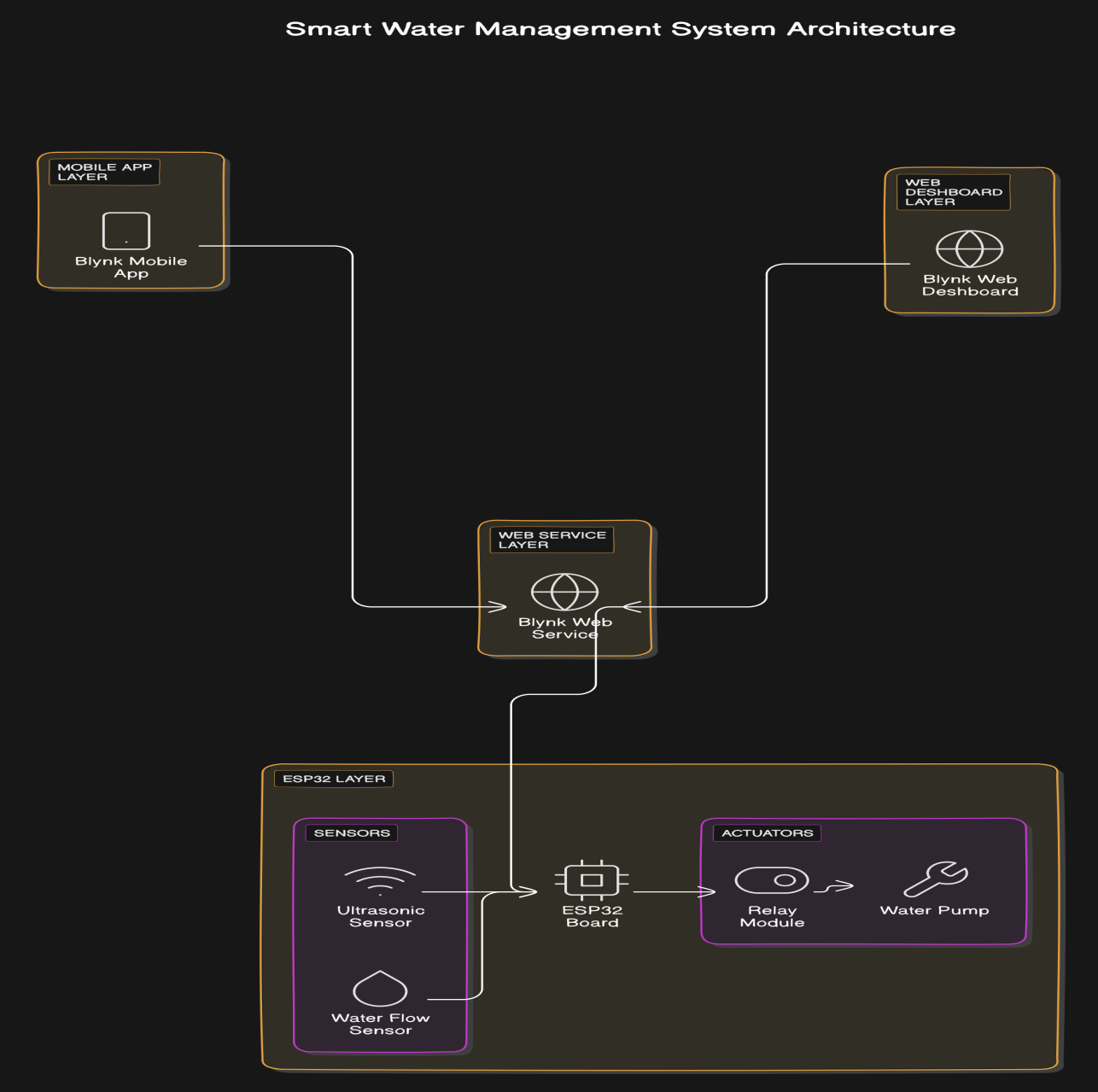
1. Entity Relationship Diagram (ERD) (To be developed using Microsoft Visio or any other drawing software of your choice)



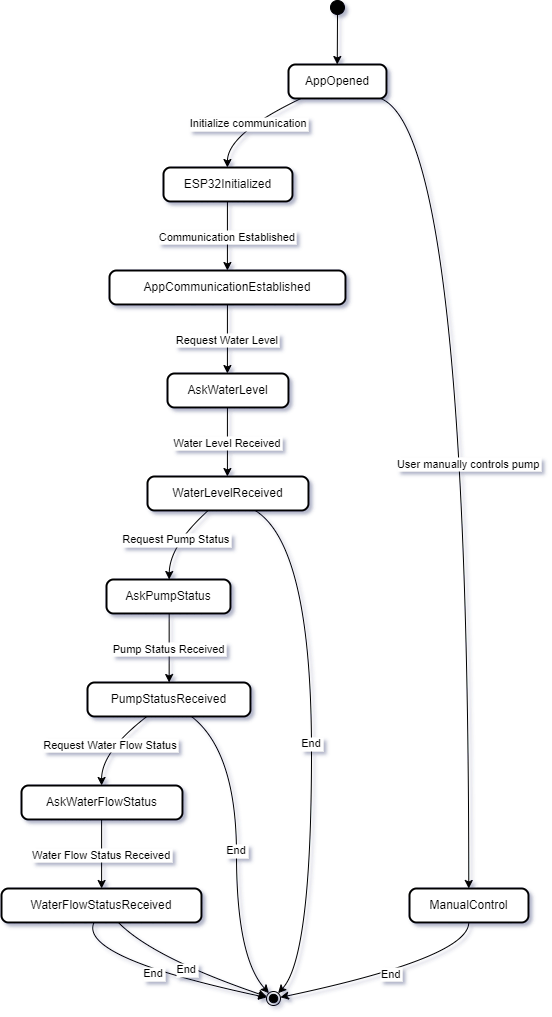
1. Sequence Diagrams (To be developed using Rational Rose or any other drawing software of your choice)



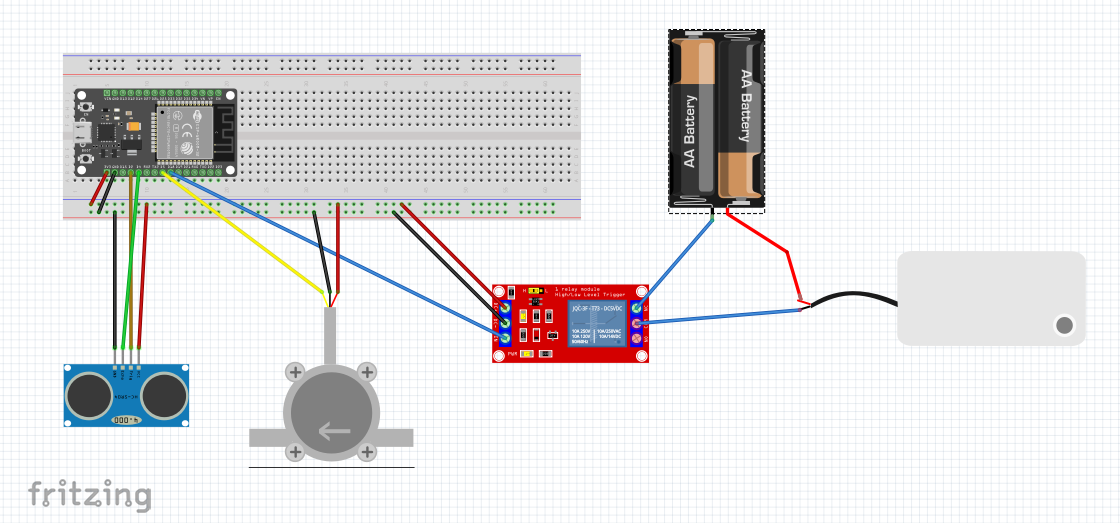
1. Architecture Design Diagram



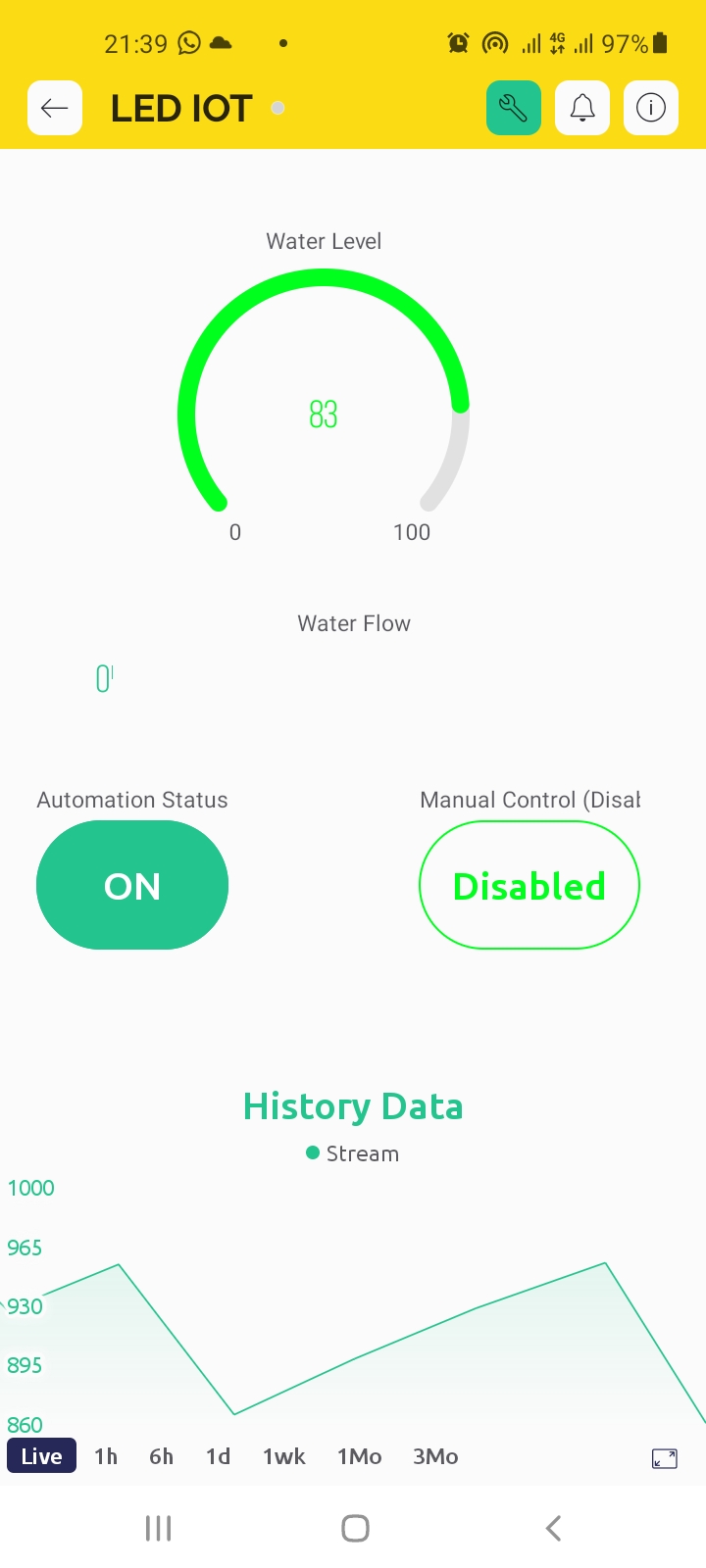
1. System State Machine



1. Circuit Design



1. Interface Design for Mobile App



1. Test Cases

**Test Case 1: Monitor Water Level (UC-1)**

**Actions:**

1. The ultrasonic sensor measures the water level in the roof tank.
2. The ESP32-Micro-Controller reads the sensor data and sends it to the Blynk mobile app.

**Expected Results:**

* The water level is displayed on the mobile app.

**Post-Conditions:**

* The water level is displayed on the mobile app.

**Test Case 2: Start Water Pump (UC-2)**

**Actions:**

1. The water level falls below a predefined threshold.
2. The ESP32-Micro-Controller sends a signal to the relay module to turn on the water pump.

**Expected Results:**

* The water pump turns on and starts filling the ground tank.

**Post-Conditions:**

* The ground tank is filled with water.

**Test Case 3: Stop Water Pump (No Flow) (UC-3)**

**Actions:**

1. The water flow sensor detects no water flow in the pipe.
2. The ESP32-Micro-Controller sends a signal to the relay module to turn off the water pump.

**Expected Results:**

* The water pump turns off.

**Post-Conditions:**

* The water pump is off.

**Test Case 4: Stop Water Pump (Overflow) (UC-4)**

**Actions:**

1. The water level rises above a predefined threshold.
2. The ESP32-Micro-Controller sends a signal to the relay module to turn off the water pump.

**Expected Results:**

* The water pump turns off.

**Post-Conditions:**

* The water tank is not overflowing.

**Test Case 5: Turn On Water Pump Manually (UC-5)**

**Actions:**

1. The user presses the "Turn on pump" button on the mobile app.
2. The Blynk mobile app sends a signal to the ESP32-Micro-Controller.
3. The ESP32-Micro-Controller sends a signal to the relay module to turn on the water pump.

**Expected Results:**

* The water pump turns on.

**Test Case 6: Turn Off Water Pump Manually (UC-6)**

**Actions:**

1. The user presses the "Turn off pump" button on the mobile app.
2. The Blynk mobile app sends a signal to the ESP32-Micro-Controller.
3. The ESP32-Micro-Controller sends a signal to the relay module to turn off the water pump.

**Expected Results:**

* The water pump turns off.

The End

Prototype Phase