

# Required Components

## 1. Hardware Components

### a. Raspberry Pi Pico W

- Main microcontroller used to read sensor data and connect to the internet.
- Pico **W** preferred because it has **built-in WiFi** for ThingSpeak and MQTT.

### b. Ultrasonic Sensor (HC-SR04)

- Measures the distance between the sensor and water surface.
- Used to calculate water level percentage / height.

### c. OLED Display (0.96" I2C – SSD1306)

- Displays:
  - Water level (cm / %)
  - System status (Safe / Warning / Critical)
  - Connectivity status (ThingSpeak / MQTT)

### d. LEDs

- **Green LED** → Safe
- **Red LED (fast blink)** → Warning
- **Red LED (solid)** → Critical
- LED resistors: **220Ω or 330Ω**

### e. Buzzer

- Audible alert for critical water levels.

### f. Connecting Wires (Male–Male / Male–Female)

- For connecting Pico to sensors and modules.

### g. Breadboard

- Easy circuit assembly and testing.

#### **h. Power Supply**

- USB cable / power bank / 5V adapter.
- 

## **2. Software Components**

#### **a. Thonny IDE**

- Main IDE used to program Raspberry Pi Pico in **MicroPython**.
- Supports uploading code directly via USB.
- Easy to install and beginner-friendly.

#### **b. MicroPython Firmware for Pico / Pico W**

- Required to run MicroPython code on the microcontroller.

#### **c. Required Python Libraries (installed in Thonny)**

- machine
  - time
  - ssd1306
  - umqtt.simple (for MQTT)
  - urequests (for ThingSpeak – optional)
  - network (WiFi connection)
- 

## **3. IoT & Cloud Platforms**

#### **a. ThingSpeak**

- For cloud dashboard
- Stores water level data and shows graphs

**b. MQTT Broker (e.g., HiveMQ / EMQX / Mosquitto)**

- For real-time alerts
- Pico publishes messages; MyMQTT app receives notifications

**c. MyMQTT Mobile App**

- Used to receive alerts:
  - Warning level
  - Critical level
  - Water level updates