Final Report
Weather Station

Group
Jarn Yam

Members

Sippakorn	Thunyahan	6631355721
Krissada	Singhakachain	6632007521
Thanakrit	Bunrueng	6632092821
Boonyakorn	Tanrattanakorn	6632111021

Presented to
Asst. Prof. Dr. Pitchaya Sitthi-amorn

2110366 Embedded System Lab I
Semester 1/2025

Department of Computer Engineering, Faculty of Engineering,
Chulalongkorn University

Contents

1 Objective **2**

2 Equipment List **2**
2.1 Setup Picture . . . 3

3 Hardware Configuration **4**
3.1 Pin Connections . . . 4
3.2 Additional Configuration Details . . . 5
3.2.1 ADC Configuration . . . 5
3.2.2 UART Configuration . . . 5
3.2.3 Timer Configuration . . . 5
3.2.4 Power Management . . . 5

4 System Architecture **6**
4.1 Weather Station Subsystem . . . 6
4.1.1 Sensors . . . 6
4.1.2 STM32 to ESP32S3 Communication . . . 6
4.1.3 MQTT Communication . . . 6
4.1.4 Additional Feature . . . 6
4.2 Data Collection Subsystem . . . 6
4.2.1 Data Flow Diagram . . . 7
4.2.2 MQTT Broker . . . 7
4.2.3 Backend Server . . . 7
4.2.4 Database . . . 8
4.3 Web Application Subsystem . . . 8
4.3.1 Frontend Architecture . . . 8
4.3.2 Pages and Features . . . 8
4.3.3 Deployment . . . 9

5 Role and Responsibility **10**

6 Project Timeline **10**
6.1 Week 1 (Nov 10-16) . . . 10
6.2 Week 2 (Nov 17-23) . . . 10
6.3 Week 3 (Nov 24-30) . . . 10
6.4 Week 4 (Dec 1-8) . . . 10
6.5 Key Milestones . . . 10

7 Appendix **11**

8 References **11**
8.1 Equipment Images . . . 11

1 Objective

The objective of this project is to design and implement a weather station system capable of monitoring and reporting various environmental parameters, including temperature, humidity, light intensity, and liquid precipitation. The system will utilize microcontroller units (MCUs) to interface with sensors, process the collected data, and transmit it to a cloud server for remote access and visualization through a web application.

2 Equipment List

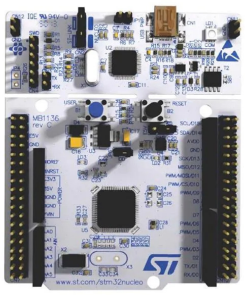

Figure 1: STM32 Nucleo F411RE Board

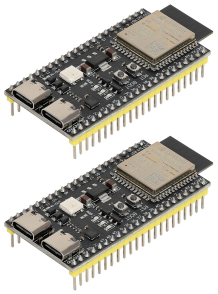

Figure 2: ESP32-S3 Development Board

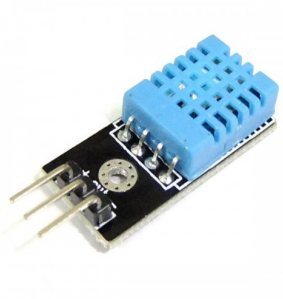

Figure 3: DHT11 Temperature and Humidity Sensor

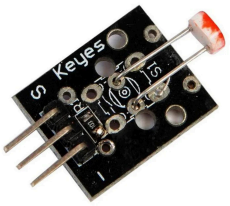

Figure 4: Light Dependent Resistor (LDR)

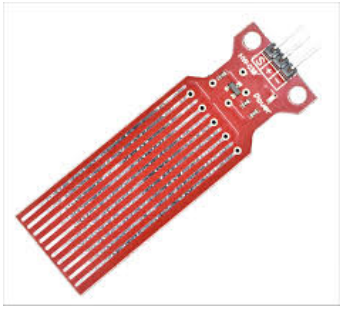

Figure 5: HW-038 Water Level Sensor

Figure 6: SG90 Servo Motor

Figure 7: Jumper Cable

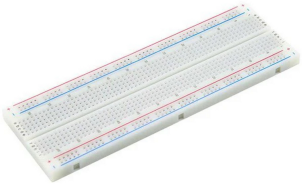

Figure 8: Breadboard

Figure 9: USB Mini Cable

Figure 10: USB Type-C Cable

2.1 Setup Picture

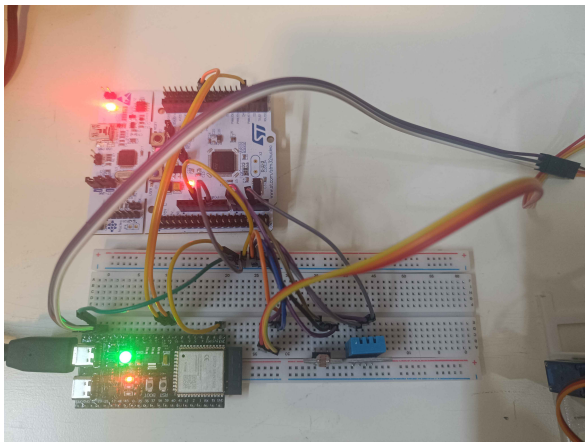

Figure 11: Hardware Setup

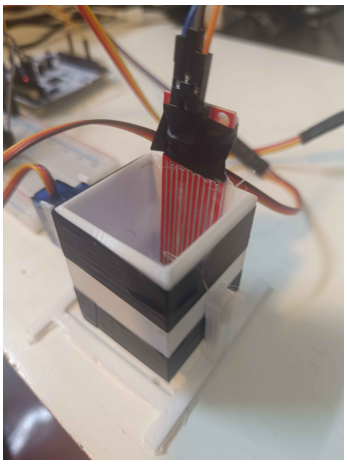

Figure 12: Servo Actuated Rain Gauge

3 Hardware Configuration

3.1 Pin Connections

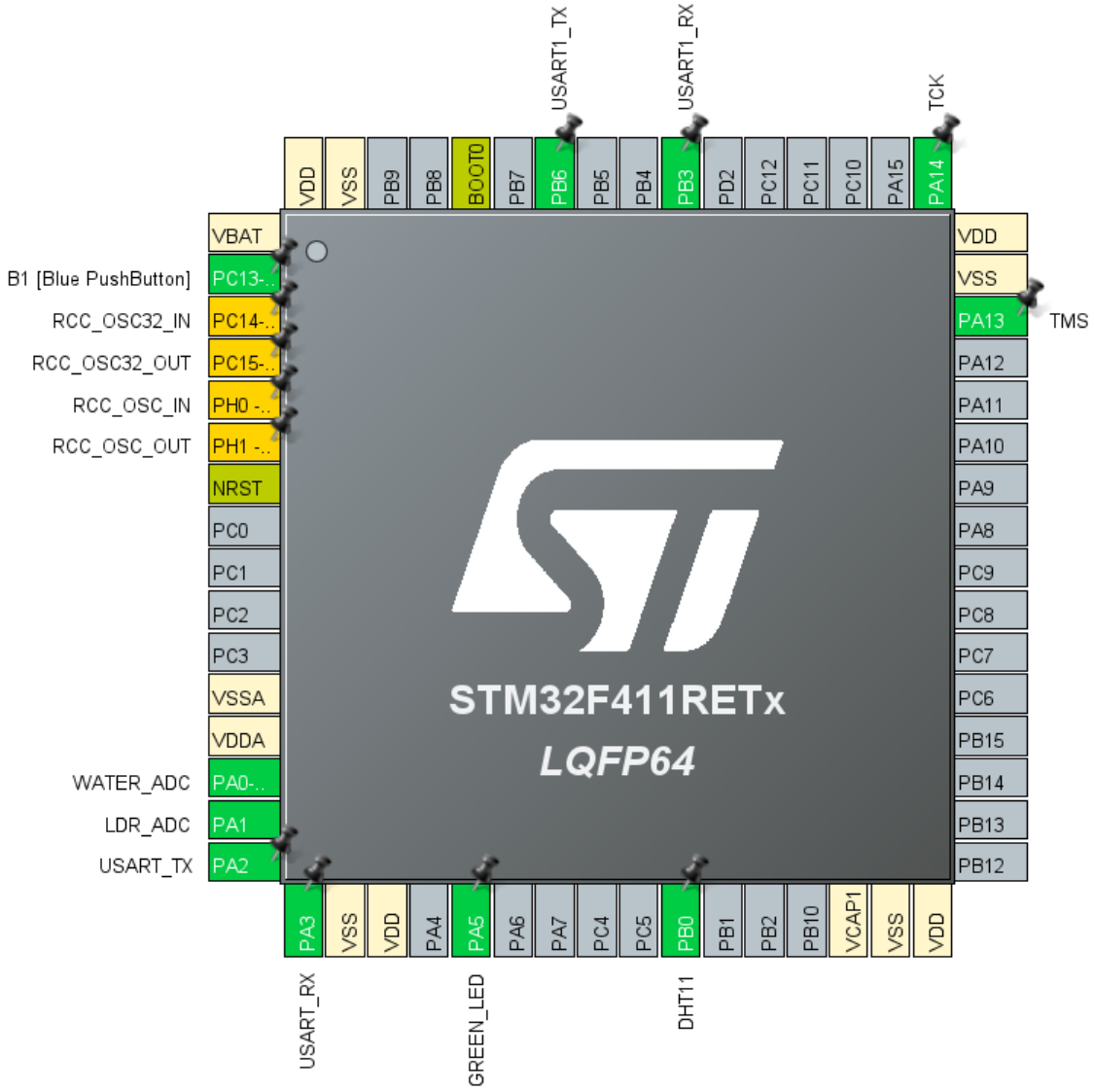

Figure 13: STM32 Nucleo F411 Pin Configuration in STM32CubeMX

Table 1: STM32 Nucleo F411 Pin Mapping and Configuration

MCU Pin	Board Pin	Pin Purpose	Configuration	Connected To
PA0	**A0**	Water Level Sensor ADC Input	Analog Input	HW-038 Water Level Sensor
PA1	**A1**	LDR Light Sensor ADC Input	Analog Input	Light Dependent Resistor (LDR)
PA5	**-**	Status LED	Digital Output (Push-Pull)	On-board LED indicator
PB0	**A3**	DHT11 Temperature/Humidity Sensor	Digital I/O	DHT11 data pin
PB3	**D3**	USART1 RX (ESP32 Communication)	Alternate Function	ESP32 UART TX
PB6	**D10**	USART1 TX (ESP32 Communication)	Alternate Function	ESP32 UART RX

Table 2: ESP32-S3 Pin Mapping and Configuration

GPIO Pin	Pin Purpose	Configuration	Connected To
GPIO 16	UART RX (STM32 Communication)	Serial Input	STM32 UART TX
GPIO 17	UART TX (STM32 Communication)	Serial Output	STM32 UART RX
GPIO 14	Servo Motor Control	PWM Output	Rain Gauge Servo Motor
GPIO 48	RGB Status LED	NeoPixel Data	WS2812B RGB LED

3.2 Additional Configuration Details

3.2.1 ADC Configuration

- **ADC1** is used for both analog sensors (LDR and Water Level)
- Resolution: 12-bit (0-4095 range)
- Channels dynamically configured in software
- LDR uses ADC Channel 1, Water sensor uses ADC Channel 0

3.2.2 UART Configuration

- **USART1**: 115200 baud, 8N1, TX/RX mode - Primary communication with ESP32
- **USART2**: 115200 baud, 8N1, TX/RX mode - Debug output via ST-Link VCP

3.2.3 Timer Configuration

- **TIM1** is configured for microsecond timing delays required by DHT11 sensor protocol

3.2.4 Power Management

- Most unused pins are configured as analog inputs (high impedance) to minimize power consumption
- ADC peripherals are started/stopped dynamically to conserve power during sensor readings

4 System Architecture

4.1 Weather Station Subsystem

The Weather Station Subsytem is responsible for collecting weather data from the environment, namely temperature, humidity, light intensity, and liquid precipitation.

The subsystem utilizes STM32 Nucleo F411 board as the primary microcontroller unit (MCU) to interface with the sensors. The STM32 then processes the raw data from the sensors and transmits the processed data to ESP32S3 via UART communication protocol. Finally, the ESP32S3 sends the data to the cloud server through MQTT to be displayed on the web application.

4.1.1 Sensors

Table 3: Weather Station Sensors

Sensor	Type	Interface	Measured Data
DHT22	Temperature/Humidity	One-wire digital	Temperature (℃), Humidity (%)
LDR	Light Intensity	Analog voltage	Light Intensity (%)
Water Level Sensor	Liquid Precipitation	Analog voltage	Water Level (mm)

4.1.2 STM32 to ESP32S3 Communication

The STM32 microcontroller communicates with the ESP32S3 using UART protocol. The ESP32S3 then sends a char 'R' to request data from the STM32. Upon receiving the request, the STM32 respond with the latest data packet. The data packet is sent as a raw byte stream with the following structure:

Table 4: UART Data Packet Structure

Field	Type	Size (bytes)
Start Byte	uint8_t	1
Temperature	uint8_t	1
Humidity	uint8_t	1
Light Intensity	float	4
Water Level	float	4
Checksum	uint8_t	1

4.1.3 MQTT Communication

The ESP32S3 connects to a Wi-Fi network and establishes a connection to the MQTT broker. It will periodically publish the weather data received from the STM32 to the "sensor/data" topic in CSV format. The ESP32S3 also subscribes to the "sensor/action" topic and which will allows for the reset of the rain gauge when "resetRainGauge" command is received.

4.1.4 Additional Feature

In addition to MQTT communication, the ESP32S3 also controls a servo motor with PWM signal to reset the rain gauge and an RGB LED to indicate system status.

Table 5: ESP32S3 Status LED Color Mapping

Status	Color	RGB Value	Meaning
Normal Operation	Green	(0, 255, 0)	All systems OK
Wi-Fi Not Connected	Red	(255, 0, 0)	Wi-Fi connection lost or unavailable
MQTT Not Connected	Blue	(0, 0, 255)	MQTT broker not connected
Sensor Error	Yellow	(255, 255, 0)	Sensor communication or data error

4.2 Data Collection Subsystem

The Data Collection Subsystem serves as the central hub for receiving, processing, storing, and distributing weather data from the ESP32S3 device to the web application. This subsystem consists of an MQTT broker, a backend server, and a database system.

4.2.1 Data Flow Diagram

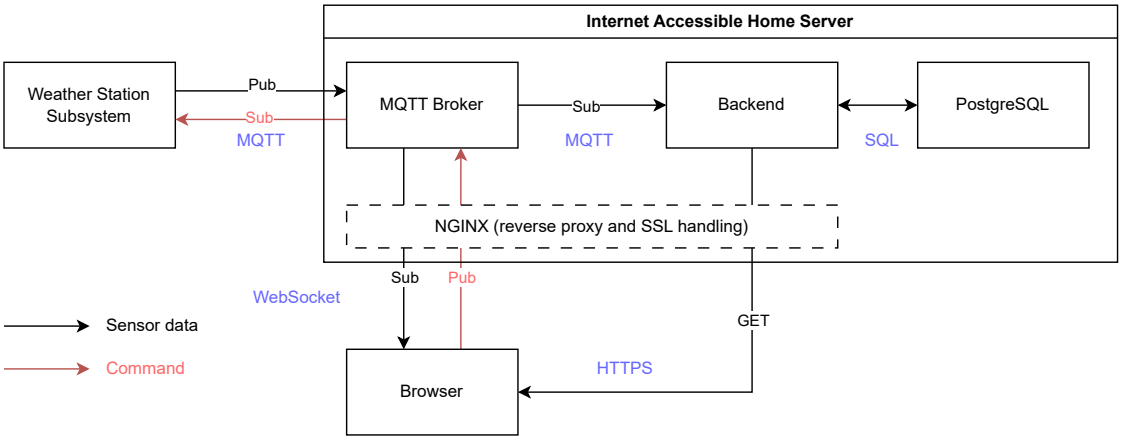

Figure 14: Diagram depicting flow of data between each major components

4.2.2 MQTT Broker

The system utilizes an MQTT broker as the primary communication protocol between the ESP32S3 and the web application. MQTT was selected for its lightweight nature, low bandwidth requirements, and publish-subscribe messaging pattern, which is ideal for IoT applications.

Due to limitation on the browser's ability to directly connect via MQTT protocol, Eclipse Mosquitto was chosen for its ability to open a WebSocket interface, allowing the web application to communicate with the broker directly.

- **Broker**: Eclipse Mosquitto
- **Protocol**: MQTT and WebSocket
- **Topics**:
  - `sensor/data`: Published by ESP32S3 containing weather readings in CSV format
  - `sensor/action`: Subscribed by ESP32S3 to receive `resetRainGauge` command

4.2.3 Backend Server

The backend server subscribes to the MQTT broker to receive sensor data, stores it in the database, and provides historical data for the web application. It has a single API endpoint which is `/api/history`.

- **Technology Stack**: Go with Fiber framework
- **MQTT Client Library**: Eclipse Paho

API Response Format
The `/api/history` endpoint returns a JSON object containing an array of historical sensor readings:

```
{
  "data": [
    {
      "id": 123,
      "timestamp": "2025-12-09T10:30:00Z",
      "temperature": 25.5,
      "humidity": 60.2,
      "light": 75.8,
      "rain": 12.3
    },
    ...
  ],
  "count": 100
```

```
}
```

Table 6: API Response Fields

Field	Type	Description
data	Array	Array of historical data points
count	Integer	Total number of records returned
id	Integer	Unique identifier for each reading
timestamp	String (ISO 8601)	Time when data was recorded
temperature	Float	Temperature in Celsius degrees
humidity	Float	Relative humidity in percentage
light	Float	Light intensity in percentage
rain	Float	Precipitation level in millimeters

4.2.4 Database

The database stores historical weather data for visualization and analysis.

- **Database System**: PostgreSQL 17
- **Data Schema**:

Table 7: Database Schema for Weather Data

Field	Data Type
id	SERIAL PRIMARY KEY
timestamp	TIMESTAMP
temperature	DOUBLE PRECISION
humidity	DOUBLE PRECISION
light	DOUBLE PRECISION
rain	DOUBLE PRECISION

4.3 Web Application Subsystem

The Web Application Subsystem provides a user-friendly interface for monitoring real-time weather data, viewing historical trends, and controlling device functions. SvelteKit is selected for its lightweight nature and ease of use.

4.3.1 Frontend Architecture

- **Framework**: SvelteKit and Tailwind CSS
- **State Management**: Svelte Store for application state
- **Data Visualization**: D3.js for graphs and Three.js for 3D elements

4.3.2 Pages and Features

1. **Real-time Dashboard**
    - Stunning 3D house scene with dynamic lighting and rain animation
    - Display current temperature, humidity, light intensity, and precipitation levels
    - “DRAIN” button for manually issuing the board to drain the rain gauge
    - Updates in real-time as new data arrives

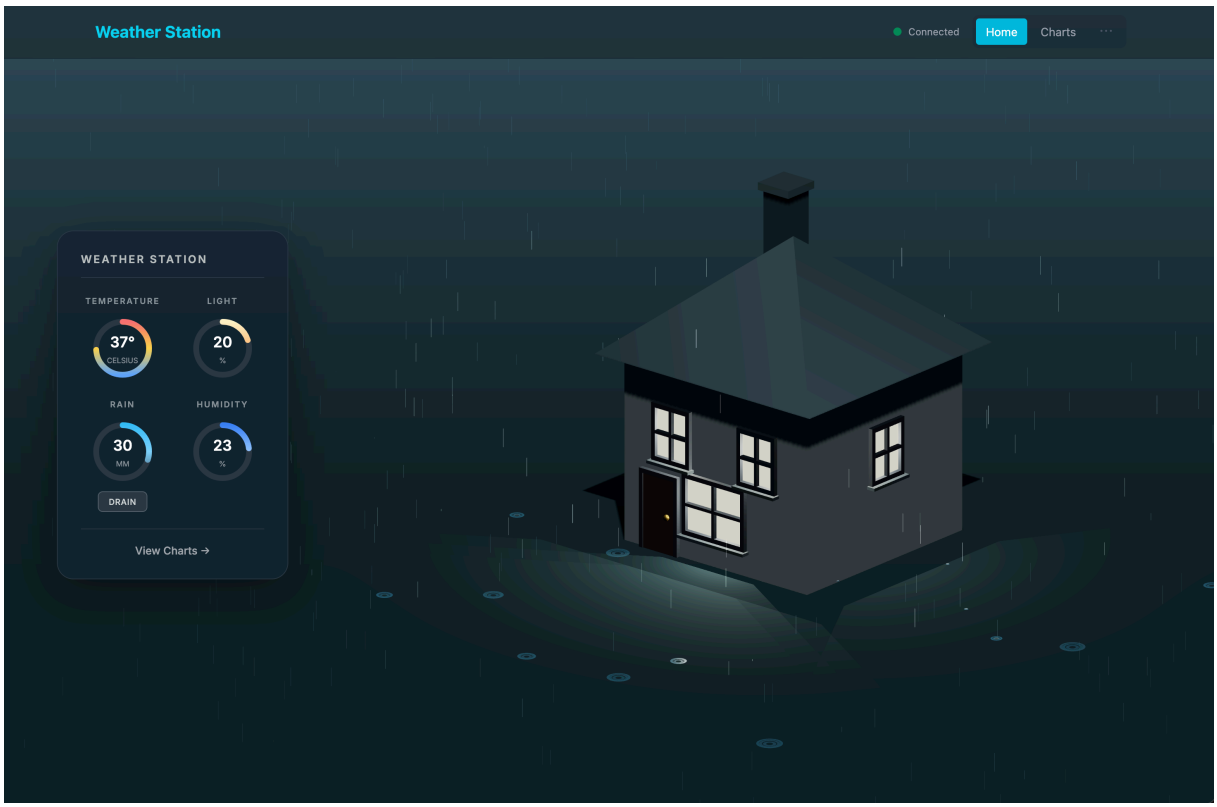

Figure 15: Screenshot of Real-time Dashboard Page

2. **Historical Data Visualization**

   - Line charts showing collected sensor data over time
   - Date range selector for custom time periods
   - Fetches historical data from backend API
   - Updates in real-time as new data arrives

Figure 16: Screenshot of Historical Data Visualization Page

4.3.3 Deployment

Both Web Application Subsystem and Data Collection Subsystem subsystems are deployed on a home server using Docker containers for portability.

5 Role and Responsibility

1. Sippakorn Thunyahan (Project Manager): Overall project coordination, document review, project setup
2. Krissada Singhakachain (Backend Engineer): Server setup, web application development
3. Thanakrit Bunrueng (System Architect): Design system architecture, design communication interface
4. Boonyakorn Tanrattanakorn (Embedded System Programmer): Implement sensor integration, embedded firmware development

6 Project Timeline

Duration: November 10 – December 8, 2025 (4 weeks)

6.1 Week 1 (Nov 10-16)

- **Sippakorn:** Project kickoff, repository setup
- **Krissada:** Research MQTT brokers, setup dev environment
- **Thanakrit:** Design system architecture, define protocols
- **Boonyakorn:** Study STM32 HAL, setup IDE, test basic I/O

6.2 Week 2 (Nov 17-23)

- **Sippakorn:** Review architecture, track milestones
- **Krissada:** Deploy MQTT broker, build web app, create APIs
- **Thanakrit:** Define MQTT topics, design data packets
- **Boonyakorn:** Implement DHT11 driver, configure ADC, develop UART

6.3 Week 3 (Nov 24-30)

- **Sippakorn:** Integration testing, update documentation
- **Krissada:** Integrate MQTT client, add data logging, build UI
- **Thanakrit:** Validate protocols
- **Boonyakorn:** Integrate sensors, implement ESP32 firmware, add servo/LED

6.4 Week 4 (Dec 1-8)

- **Sippakorn:** Final testing, check results, prepare presentation
- **Krissada:** Deploy production
- **Thanakrit:** Performance validation, finalize documentation
- **Boonyakorn:** Bug fixes, power optimization

6.5 Key Milestones

- **Nov 16:** Architecture approved, dev environment ready
- **Nov 23:** Individual subsystems functional
- **Nov 30:** Full system integration complete
- **Dec 8:** Final testing and delivery

7 Appendix

- Source code repository: https://github.com/orgs/Embedded-Lab-Project-2025/repositories
- MQTT URL: ubunsin.krissada.com:1883 (MQTT), https://ws.jarnyam.krissada.com (WebSocket)
- Backend URL: https://api.jarnyam.krissada.com
- Web Application URL: https://jarnyam.krissada.com

8 References

8.1 Equipment Images

- STM32 Nucleo F411RE Board: https://th.mouser.com/ProductDetail/STMicroelectronics/NUCLEO-F411RE?qs=Zt3UNFD9mQjdEJg18RwZ2g%3D%3D
- ESP32-S3 Development Board: https://www.amazon.com/YEJMKJ-Development-ESP32-S3-DevKitC-1-N16R8-ESP32-S3-WROOM-1-Microcontroller/dp/B0CRRNLTPD
- DHT11 Temperature and Humidity Sensor: https://www.genlogic.co.th/product/60/dht11-temperature-and-humidity-sensor-module
- Light Dependent Resistor (LDR): https://kitsguru.com/products/photosensitive-resistor-sensor-module-for-arduino
- HW-038 Water Level Sensor: https://www.majju.pk/product/analog-water-level-sensor-module-for-arduino/
- SG90 Servo Motor: https://th.cytron.io/p-sg90-micro-servo
- Jumper Cable: https://www.robotsiam.com/product/16/jumper-f2m-cable-wire-40pcs-2-54mm-20cm-female-to-male
- Breadboard: https://www.pishop.us/product/full-sized-breadboard-830-points/
- USB Mini Cable: https://th.rs-online.com/web/p/usb-cables/1862818
- USB Type-C Cable: https://www.torchdirect.co.uk/products/usb-type-c-to-usb-type-c-1-5-m-data-sync-and-charging-cable