



# **ACE6263-EMBEDDED IoT SYSTEMS & APP**

## **Assignment Report**

### **Flood Gate Monitoring System**

**Group 10:**

<b>Name</b>	<b>Student ID</b>	<b>Majoring</b>
<b>Yousef Mohamed</b>	<b>1201102720</b>	<b>CE</b>
<b>Sohaila Khalid</b>	<b>1201102260</b>	<b>CE</b>
<b>Amro Ali</b>	<b>1201100877</b>	<b>CE</b>
<b>Ebrahim Lutf</b>	<b>1211306409</b>	<b>CE</b>

# **Chapter 1: Overview**

## **1.1 Introduction**

Flooding is one of the most devastating natural disasters affecting many regions around the world, especially in areas with inadequate drainage or unpredictable rainfall. With increasing climate variability, communities located near rivers or low-lying areas are particularly vulnerable. Traditional flood control mechanisms often rely on manual monitoring and delayed human response, which may not be efficient in emergency situations.

This project presents an IoT-based floodgate and river monitoring system designed to automate the detection of water level, rainfall, and flow rate using sensors. The system is powered by an ESP32 microcontroller, with 3 different sensors and actuators and uses the Blynk IoT platform as a real-time dashboard for monitoring and control. It aims to prevent flooding by automatically activating a gate mechanism based on sensor readings and issuing alerts through the dashboard using a dc motor.

## **1.2 Problem statement**

In many parts of the world, flood control still depends on someone physically checking water levels and manually operating gates, a slow, outdated approach that leaves little room for quick action. When storms hit, sending workers out to adjust floodgates isn't just inefficient, it's dangerous. Without real-time data, communities are left guessing until it's too late, leading to preventable damage, ecological harm, and even tragic loss of life. What we need is a smarter system—one that can sense rising waters before disaster strikes, act on its own, and alert the right people in time to make a real difference.

## **1.3 Objectives**

- To design and implement an automated floodgate control system using ESP32 and IoT sensors.
- To measure environmental variables such as water level and rainfall intensity.
- To use Blynk IoT dashboard for real-time monitoring and control.
- To alert users via visual (LED) and audio (buzzer) alarms in critical situations.

## **1.4 Proposed solution**

To address the problem of delayed flood response and ineffective manual monitoring, this project proposes an automated, real-time River & Floodgate Monitoring System powered by an ESP32 microcontroller and integrated with IoT capabilities. The system is designed to detect the danger of floods and raise appropriate alarms.

## Chapter 2: Methodology

### 2.1 System Architecture

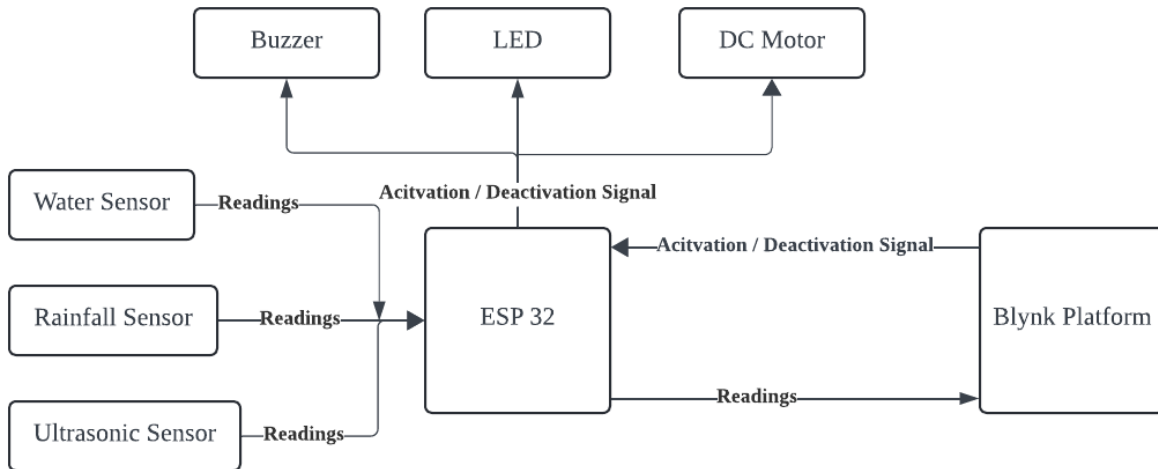


Figure 2.1: System Block Diagram

### 2.2 Sensors and actuators

#### Sensors Used

- Ultrasonic Sensor: Measures river water level by calculating the distance between the sensor and the water surface.
- Rain Detector Sensor: Detects presence and intensity of rainfall.
- Water Level Sensor: Provides backup detection by sensing the actual water level.

#### Actuators Used

- DC Motor: Controls the physical movement of the floodgate based on sensor inputs.
- Buzzer: Emits an audible alarm when critical water levels or rainfall are detected.
- LED Indicator: Provides a visual alert to indicate high-risk situations.

## 2.3 Hardware design

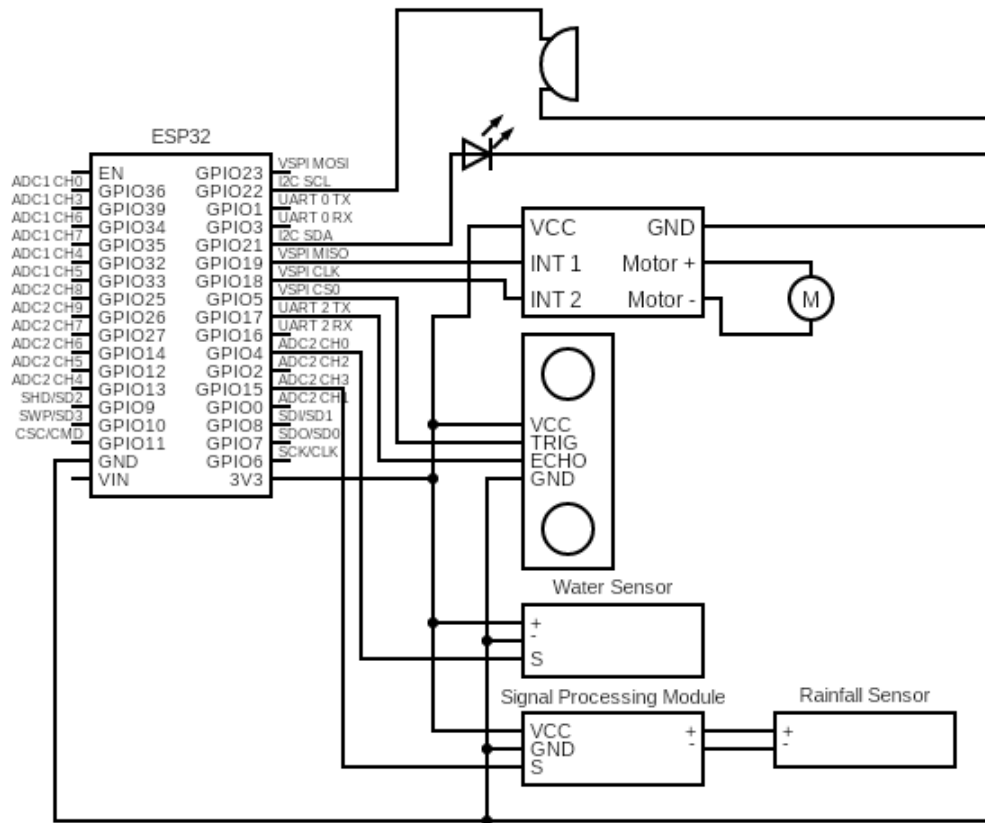


Figure 2.2: Circuit Diagram

## 2.4 Software design

### 2.4.1 Introduction

The software component of this project was designed to manage real-time monitoring, decision-making, and control logic of the floodgate automation system. The ESP32 microcontroller was programmed using Arduino IDE and communicates with the Blynk IoT platform to send and receive data from various sensors and actuators.

Blynk was chosen for its user-friendly mobile interface and real-time dashboard capabilities, allowing both automatic and manual control over the system. A total of seven data streams were configured in Blynk, each corresponding to a specific sensor or actuator. These data streams are mapped to virtual pins (V0–V6) and allow for two-way communication between the hardware and the dashboard.

## 2.4.2 Blynk IOT

### Datastreams








ID	Name	Pin	Color	Data Type	Units	Is Raw	Min	Max
1	Water level	V2		Integer	cm	false	0	100
2	rainfall	V1		Integer		false	0	1
3	buzzer	V0		Integer		false	0	1
4	Open gate	V3		Integer		false	0	1
5	flood warning	V4		Integer		false	0	1
6	warning light	V5		Integer		false	0	1
7	close gate	V6		Integer		false	0	1

Figure 2.3: Data streams used in Blynk

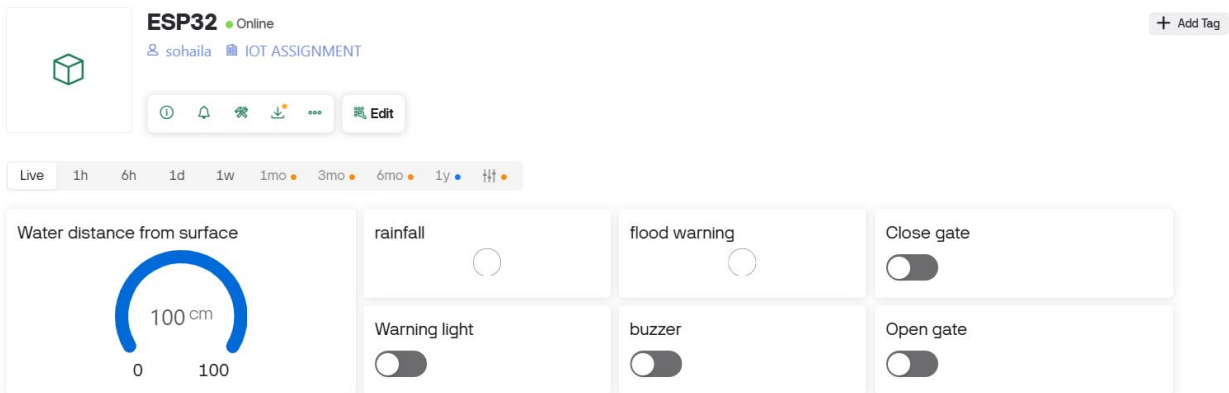


Figure 2.4: Web dashboard

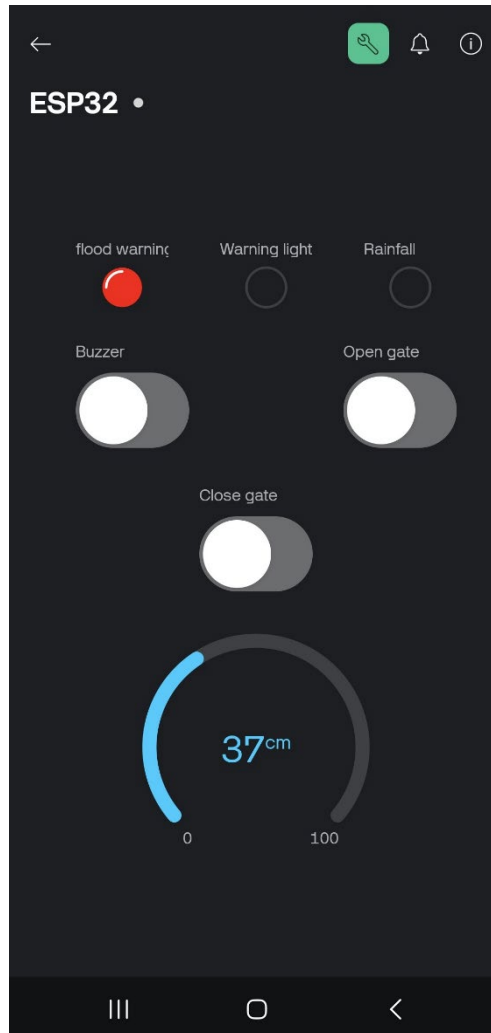


Figure 2.5: Mobile dashboard

#### 2.4.3 Dashboard Functions:

- The water level is displayed in centimeters and continuously updated in real-time using an ultrasonic sensor.
- Rainfall detection (via rain sensor) is triggered to red when rainfall is detected.
- Manual control switches for opening and closing the gate, controlled by the DC motor, allow for testing purposes.
- A warning light and buzzer can be activated automatically or manually when floods are detected.
- The flood warning indicator using the water level sensor turns red when critical water levels are reached.

## 2.5 System Flowchart

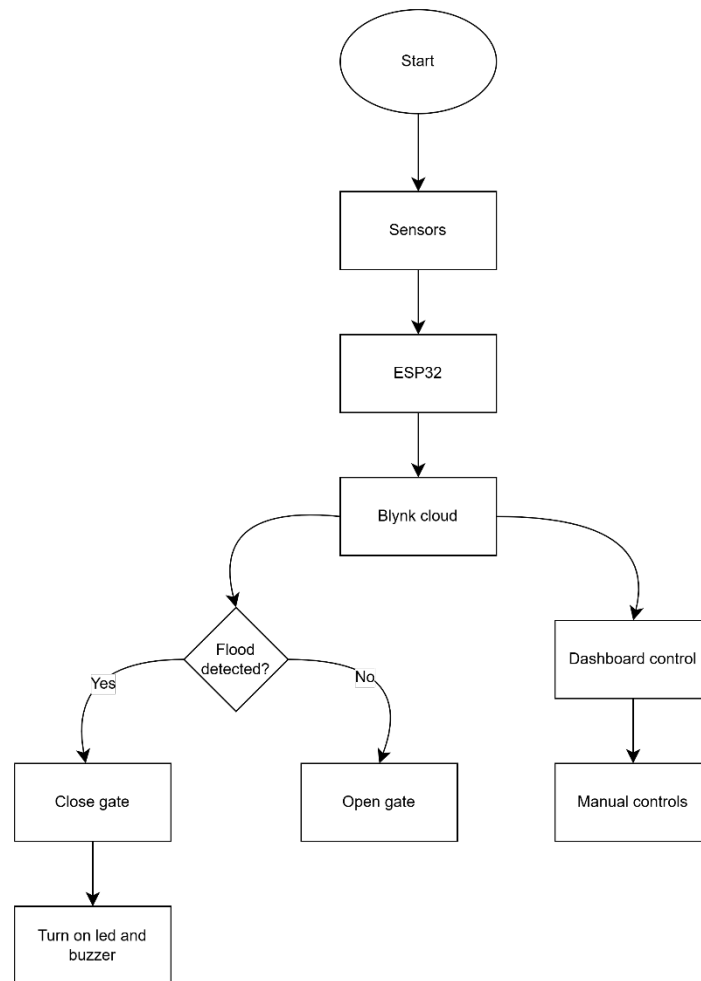


Figure 2.6: Flowchart

## 2.6 Source Code

```
#define BLYNK_TEMPLATE_ID "TMPL6h30rB3zU"
#define BLYNK_TEMPLATE_NAME "Flood gate"
#define BLYNK_AUTH_TOKEN "ip7FeQwiBZLgA2V21HioiNA01GVsY_PW"

#define ULTRASONIC_ECHO_PIN 17
#define ULTRASONIC_TRIG_PIN 5

#define RAINFALL_SENSOR_PIN 15

#define WATER_SENSOR_PIN 4

#define BUZZER_PIN 22
#define LED_PIN 21

#define MOTOR_OPEN_PIN 19
#define MOTOR_REVERSE_PIN 18

#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

// network credentials
char ssid[] = "Sohaila's A55";
char pass[] = "am55sikiak12";
char auth[] = BLYNK_AUTH_TOKEN;

// variable to store ultrasonic sensor readings
long duration, distance;

void setup() {
  Serial.begin(115200);

  Blynk.begin(auth, ssid, pass);

  pinMode(ULTRASONIC_TRIG_PIN, OUTPUT);
  pinMode(ULTRASONIC_ECHO_PIN, INPUT);

  pinMode(RAINFALL_SENSOR_PIN, INPUT);

  pinMode(WATER_SENSOR_PIN, INPUT);
```



```

digitalWrite(WATER_SENSOR_PIN, LOW);

pinMode(BUZZER_PIN, OUTPUT);
digitalWrite(BUZZER_PIN, LOW); // Buzzer OFF by default

pinMode(LED_PIN, OUTPUT);
digitalWrite(LED_PIN, LOW); // LED OFF by default

pinMode(MOTOR_OPEN_PIN, OUTPUT);
pinMode(MOTOR_REVERSE_PIN, OUTPUT);
// motor off by default
digitalWrite(MOTOR_OPEN_PIN, LOW);
digitalWrite(MOTOR_REVERSE_PIN, LOW);
}

void loop() {
  Blynk.run();
  ultrasonic();
  detectRain();
  detectWater();
}

//Ultrasonic function to calculate the distance and send to blynk
void ultrasonic() {
  digitalWrite(ULTRASONIC_TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(ULTRASONIC_TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(ULTRASONIC_TRIG_PIN, LOW);

  duration = pulseIn(ULTRASONIC_ECHO_PIN, HIGH);
  distance = duration / 58.2;
  String disp = String(distance);

  Blynk.virtualWrite(V2, distance);
  delay(1000);
}

void detectRain(){
  // LOW if water detected, HIGH if not
  int rainDetected = digitalRead(RAINFALL_SENSOR_PIN);

```

```

    if (rainDetected == LOW) {
        Blynk.virtualWrite(V1, 1);
        Serial.print("rain detected\n");
    } else {
        Blynk.virtualWrite(V1, 0);
        Serial.print("no rain\n");
    }

    delay(1000);
}

void detectWater(){
    int waterLevel = digitalRead(WATER_SENSOR_PIN);

    Serial.print("Water Sensor Value: ");
    Serial.println(waterLevel);

    if (waterLevel == HIGH) {
        Blynk.virtualWrite(V4, 1);
        activateWarning();
    } else {
        Blynk.virtualWrite(V4, 0);
    }

    delay(500);
}

void activateWarning() {
    Blynk.virtualWrite(V0, 1);
    Blynk.virtualWrite(V5, 1);
    digitalWrite(LED_PIN, HIGH);
    digitalWrite(BUZZER_PIN, HIGH);
}

// Functions to control the relay from the Blynk app

// Buzzer functionality
BLYNK_WRITE(V0) {
    if (param.asInt() == HIGH) {
        digitalWrite(BUZZER_PIN, HIGH);
    } else {
        digitalWrite(BUZZER_PIN, LOW);
    }
}

```

```
}

// LED functionality
BLYNK_WRITE(V5) {
    if(param.asInt() == HIGH) {
        digitalWrite(LED_PIN, HIGH);
    } else {
        digitalWrite(LED_PIN, LOW);
    }
}

// open gate functionality
BLYNK_WRITE(V3) {
    if(param.asInt() == HIGH) {
        digitalWrite(MOTOR_OPEN_PIN, HIGH);
    } else {
        digitalWrite(MOTOR_OPEN_PIN, LOW);
    }
}

// close gate functionality
BLYNK_WRITE(V6) {
    if(param.asInt() == HIGH) {
        digitalWrite(MOTOR_REVERSE_PIN, HIGH);
    } else {
        digitalWrite(MOTOR_REVERSE_PIN, LOW);
    }
}
```

## 2.7 Budget and bills of materials

Item	Quantity	Price (Per one)	Total	Justification
Buzzer	2	1.40	2.80	To alert the danger audibly
Jumper wires	1	4.50	4.50	To connect the circuit together
Water level sensor	2	4.60	9.20	Detects rising water levels as a core input for flood condition assessment.
Ultrasonic sensor	2	3.20	6.40	Measures distance from water surface to detect river level changes accurately.
Raindrop sensor	1	6.50	6.50	To detect the rain fall which is one of our system objectives
Breadboard	1	3.80	3.80	Used as a prototyping platform to mount and connect all electronic components.
LED	1	1.20	1.20	Used to alert danger visually
Dc motor	4	2.80	11.20	Powers the mechanical movement of the floodgate to regulate water flow.
Plastic Gears with different sizing	4	0.3-0.7	2.5	Transfers motor torque to the gear rack, enabling gate movement.
Esp32	1	16.99	16.99	Acts as the main controller, handling sensor

				input, actuator control, and IoT link.
Gear Rack	1	0.50	0.50	Converts motor rotation into linear motion for gate sliding.
Total	65.60			

## Chapter 3: Results

### 3.1 Results

The Floodgate Monitoring System was successfully implemented and tested under controlled conditions. All components performed as expected, and the system met its functional objectives.

- The ultrasonic sensor, rain detector, and water level sensor consistently provide accurate and reliable readings during testing.
- The DC motor correctly activated to open or close the floodgate depending on water level inputs.
- The buzzer and LED were triggered appropriately during alert conditions, providing clear audible and visual warnings.
- The Blynk IoT dashboard effectively visualizes all live sensor data, displaying water level, rainfall detection, and system status through user-friendly widgets.
- Remote control features on the Blynk dashboard allow manual activation of all actuators (motor, buzzer, LED), confirming reliable two-way communication between the cloud and hardware.

## Chapter 4: Conclusion

A river floodgate monitoring system using an ESP32 microcontroller was successfully developed. Integrated with it are multiple sensors, ultrasonic, rainfall, and water level sensors all to assist in detecting potential flood conditions. By incorporating a DC motor for gate control, an LED and buzzer for local alerts, and real-time monitoring through the Blynk IoT platform, the system demonstrates a practical and cost-effective approach to automated flood prevention. The integration of IoT ensures remote access and timely alerts, enhancing safety and responsiveness. This prototype lays the foundation for scalable, automated flood management solutions in vulnerable regions.

## **Reflections and Contributions: Amro Ali**

To begin with, working on the river floodgate system was a highly rewarding and educational experience. I gained a new insight into the real-world applications of IoT systems. I have also learned a lot about how river floodgate monitoring systems work, and the science and engineering behind them. In addition, whilst conducting research for the project, I learned about the devastating effects that flooding has on the affected communities and the importance of implementing such systems to alleviate the dangers at best and act as a cautionary system at worst.

Moving on, I have mainly contributed to both the hardware and software development of the project, the specifics of my contributions are as follows:

- Helped in acquiring components.
- Participated in circuit design.
- Participated in making circuit connections.
- Participated in testing and debugging circuit components.
- Participated in developing code for ESP32.
- Participated in testing and debugging code for ESP 32.

## **Reflections and Contributions: Yousef Mohamed**

Working on the river floodgate system project was a solid hands-on experience that helped bridge the gap between theory and real-world IoT applications. It gave me a better understanding of how critical systems like flood monitoring operate and why they matter. Through this project, I became more aware of how damaging floods can be to communities and how systems like these aren't just technical exercises; they have real potential to save lives and minimize disaster impact.

My main role was focused on developing the prototype and I had direct involvement in both the hardware and software aspects of the project, including:

- Assisting with sourcing and selecting required components
- Contributing to the circuit design process
- Contributing to building and wiring the prototype on the breadboard
- Contributing to testing, troubleshooting, and refining circuit connections
- Participated in writing and debugging the ESP32 control code
- Contributing to Running tests to ensure proper system behavior
- Helped in assembling and building the full prototype setup

## **Reflections and Contributions: Ebrahim Lutf**

Working on the river floodgate monitoring system was a valuable learning experience that deepened my understanding of how embedded systems and mechanical components can be integrated to address real-world challenges. It was rewarding to see how a system we built from scratch could respond to environmental conditions and make automatic decisions, especially in a safety-critical scenario like flood control.

My main responsibility was designing and implementing the mechanical prototype of the floodgate using a DC motor. This involved integrating the motor with a gear rack to convert rotational motion into linear movement, allowing the gate to open and close smoothly. I contributed to:

- Designing the floodgate movement mechanism using gears and racks
- Integrating the DC motor into the hardware system
- Calibrating the gate movement to respond to control signals from the ESP32
- Testing and adjusting the motor control logic for accurate gate positioning
- Supporting the team in assembling the final prototype setup



## **Reflections and Contributions: Sohaila Khalid**

This project allowed me to apply my theoretical knowledge of IoT and embedded systems to a real-world safety application. I also gained valuable insight into how flood response systems function and how IoT can be leveraged to enable timely and efficient monitoring. Most importantly, this project highlighted the life-saving potential of such systems, especially in regions prone to flooding, where early warnings and automated responses can make a significant difference.

My primary contribution was in developing and integrating the Blynk IoT dashboard, which serves as the user interface for remote monitoring. This dashboard enables users to visualize sensor data such as water level, flow rate, and rainfall in real time. It also allows control over the floodgate mechanism through actuator triggers.

I contributed to the project in the following specific areas:

- Designed and configured the Blynk IoT dashboard layout
- Integrated sensor readings (water level, flow, rainfall, and temperature) into the dashboard
- Developed and tested communication between the ESP32 and the Blynk app via Wi-Fi
- Enabled real-time data visualization for remote monitoring
- Collaborated with teammates to ensure the dashboard aligned with the overall system logic and hardware behavior

This experience has strengthened my skills in IoT platform integration, real-time data communication, and user interface design for embedded systems.