UNIVERSITY OF VOCATIONAL TECHNOLOGY

Faculty of Engineering Technology
Department of Electro-Mechanical Technology

EE402040 Internet of Things

(IoT)

Project Proposal

Water Heater Temperature Control and Monitoring System by using ESP32-IoT

<u>Group Members</u>

Name	: H.M.Janith odithya bamunugedara (MEC/22/B1/22)
	: N.M.A.M.Shehan maduranga (MAN/22/B1/09)
Program	: B. tech in Mechatronics Technology
Submission Date	: 21st June,2025

Instructed by: Mr. Janith Kasun

Contents

3.Introduction

❖ Problem Statement

Conventional water heaters typically operate without intelligent control or remote monitoring capabilities. This leads to several challenges:

- **Energy Waste:** Heaters often run longer than needed, resulting in unnecessary electricity consumption.
- **Inconsistent Temperature:** Manual operation can cause water to be too hot or too cold, impacting comfort and safety.
- **Safety Risks:** Overheating can lead to scalding or even fire hazards, especially when heaters are left unattended.
- **Lack of Convenience:** Users must be physically present to monitor or adjust the heater, which is inconvenient in modern, fast-paced lifestyles.

❖ Motivation

The increasing adoption of smart home technologies and the Internet of Things (IoT) highlights a growing demand for intelligent, energy-efficient, and remotely accessible appliances. Water heating is a significant contributor to household and commercial energy bills. By integrating IoT with water heaters, users can:

- Remotely monitor and control water temperature, ensuring hot water is available when needed.
- Reduce energy consumption by scheduling or automating heater operation based on actual usage patterns.
- Enhance safety by implementing overheat protection and real-time alerts.
- Improve convenience and user experience, especially for people managing multiple properties or facilities.

❖ Aim & Objectives

Aim:

To design and implement an IoT-based Water Heater Temperature Control and Monitoring System utilizing ESP32, enabling efficient, safe, and convenient management of water heating.

Objectives (SMART):

- **Specific:** Develop a prototype system that allows users to remotely monitor and control their water heater's temperature via a mobile/web app.
- **Measurable:** Achieve temperature regulation within ±1°C of the user-set threshold, with real-time status updates.
- **Achievable:** Use readily available hardware (ESP32, DS18B20 sensor, relay module) and open-source platforms (Blynk or custom web app).
- **Relevant:** Address key issues of energy efficiency, safety, and user convenience in water heating.
- **Time-bound:** Complete research, design, implementation, and testing within 10 weeks.

Solution Overview

This project proposes a system where:

- The **ESP32 microcontroller** continuously reads water temperature using a DS18B20 sensor.
- If the temperature falls below a user-defined threshold, the ESP32 activates a relay to turn on the heating element.
- Once the desired temperature is reached, the relay is deactivated to stop heating.
- All temperature data and control options are accessible via the **Blynk IoT platform** (or a custom web app), allowing users to monitor and adjust settings from anywhere using a

smartphone or computer.

- The system can send real-time alerts for over-temperature situations, ensuring both safety and peace of mind.

Applications

The proposed system is highly versatile and can be applied in:

- **Residential homes** for smart water heating and energy savings.
- **Hotels and commercial buildings** for centralized, remote management of multiple heaters.
- **Industrial settings** where precise temperature control is crucial for processes.
- **Healthcare and elderly care facilities** to maintain safe water temperatures and prevent accidents.
- **Remote or off-grid locations** where efficient energy use and remote monitoring are critical.

Significance

By leveraging affordable IoT hardware and cloud platforms, this project addresses real-world problems of energy waste, safety, and user inconvenience in water heating. It demonstrates the practical benefits of IoT in daily life and provides a scalable template for further smart appliance development.

4.Literature Review

❖ What Exists Already?

Several research works and projects have demonstrated the use of the ESP32 microcontroller integrated with IoT platforms for temperature control and monitoring systems, particularly for water heaters and similar thermal applications.

1. A comprehensive system using ESP32, DS18B20 temperature sensors, relay modules, and the Blynk IoT platform has been developed for remote monitoring and control of water heater temperature. This system allows users to set temperature thresholds, remotely switch the heater on/off, and visualize real-time temperature data via a mobile app. It targets residential, commercial, and industrial applications, aiming to optimize energy use and improve user convenience through automation and remote management[1].

2. Another project emphasizes energy efficiency by employing advanced algorithms with ESP32 to dynamically regulate temperature and minimize energy consumption. This system includes real-time monitoring and adaptive control strategies, focusing on sustainable and resource-conscious temperature control applications[2].

3. An automated geyser temperature control system using ESP32 incorporates dual-relay control for managing water flow and heating, along with an LCD display for local monitoring. It integrates digital sensors and remote access via the Blynk app, aiming to reduce power consumption and water wastage by automating heating cycles[3].

4. A remote thermostat control and temperature monitoring system utilizes ESP32 Thing boards and an open-source IoT platform (openHAB) to create a low-cost, low-power consumption solution for thermal energy storage. It uses MQTT protocol for communication and allows users to visualize sensor data on a home server dashboard, enhancing remote control capabilities[4].

5. Practical implementations like the ESP32 IoT Water Heater project use MQTT-based communication with cloud platforms such as Cayenne IoT Cloud. These projects enable scheduling, temperature monitoring with multiple DS18B20 sensors, and remote switching of heating elements, focusing on power saving and user convenience.

❖ What's Missing (That You'll Fix)?

Despite these advancements, several gaps remain in existing systems:

- **Comprehensive Integration of Safety Features:** Many current systems focus on temperature control and remote monitoring but lack integrated safety mechanisms such as over-temperature protection, fault detection, and emergency shutdown protocols.
- **Enhanced User Interface and Customization:** While platforms like Blynk and openHAB provide basic control and visualization, there is limited customization for userspecific preferences, alerts, and predictive maintenance notifications.
- **Energy Consumption Analytics:** Few systems offer detailed analytics on energy usage patterns to help users optimize consumption further.
- **Scalability and Multi-Device Management:** Existing solutions often target single water heaters or small setups. There is a need for scalable architectures that manage multiple devices across large residential or commercial buildings with centralized control.
- **Robustness in Diverse Environments:** Many projects do not address environmental challenges such as varying water quality, sensor calibration drift, or network reliability in IoT communication.

❖ Cited Sources

1."Water Heater Temperature Control and Monitoring System by using ESP32 IoT," ArduinoExpert,2025.[Online].Available: https://arduinoexpert.com/water-heater-temperature-monitoring-and-control-by-using-esp32-iot/

2.2 S. Author, "ESP32 Temperature Control Project With An Emphasis On Energy Efficiency,"SSRN,May2024.[Online].
Available: https://papers.ssrn.com/sol3/Delivery.cfm/SSRN_ID4841107_code6767600.pdf?abstractid=4841107&mirid=1&type=2

3.3 A. Author, "Design and Implementation of an Automated Geyser Temperature Control SystemUsingESP32,"IJRASET.[Online].Available: https://www.ijraset.com/best-journal/design-and-implementation-of-an-automated-geyser-temperature-control-system-using-esp32-349

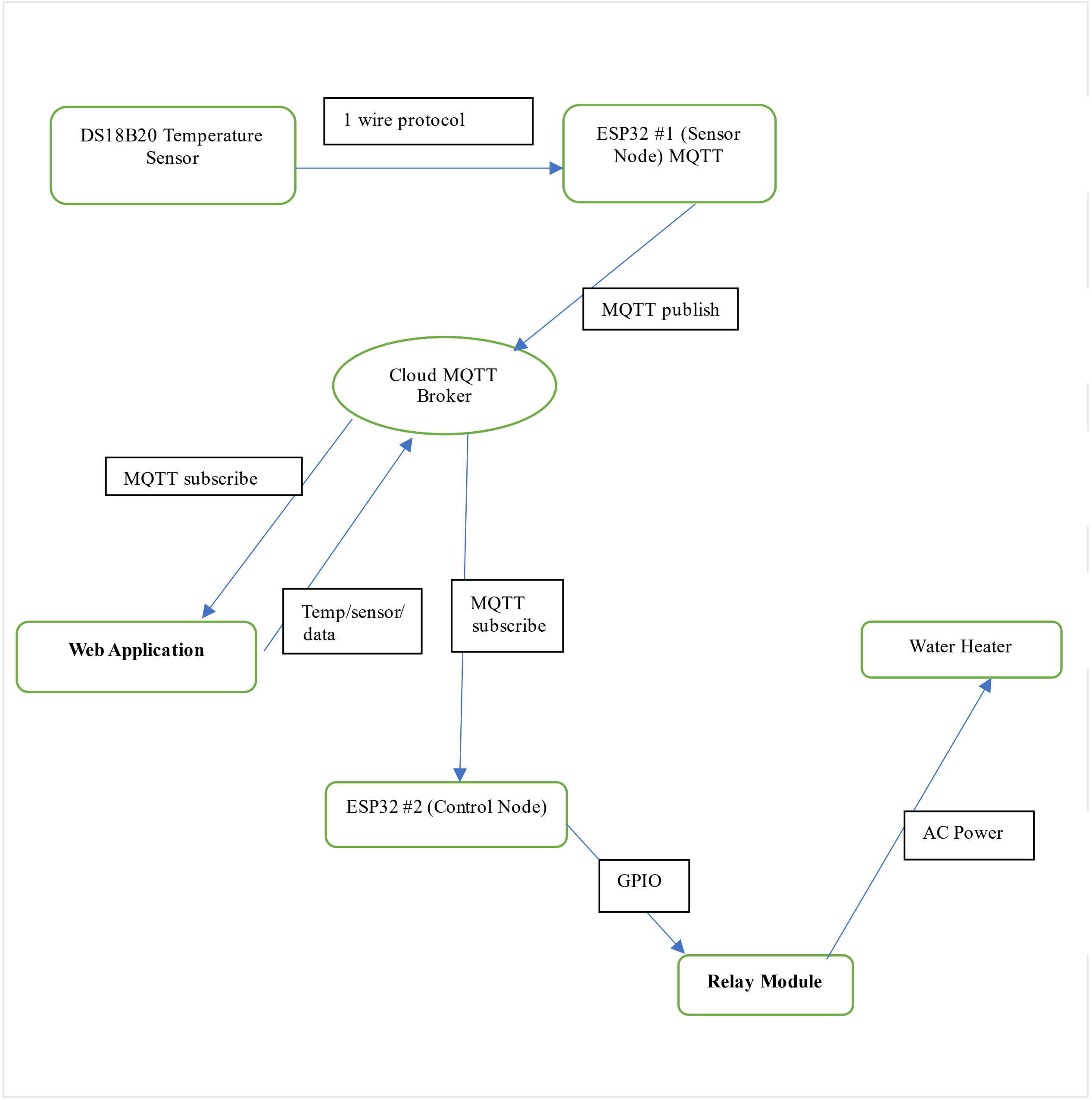
4. 4 M. Author, "A Remote Thermostat Control and Temperature Monitoring System," EJECE,vol.4,no.5,Sep.2020.[Online].Available: https://pdfs.semanticscholar.org/35c6/8b378ef199881163e2a1098fc315d26a3b3e.pdf

5. 6 J.Slabbert,"ESP32IoTWaterHeater,"Instructables.[Online]. Available: https://www.instructables.com/Cayenne-IoT-Geyser/

5.Methodology & System Design

❖ Architecture

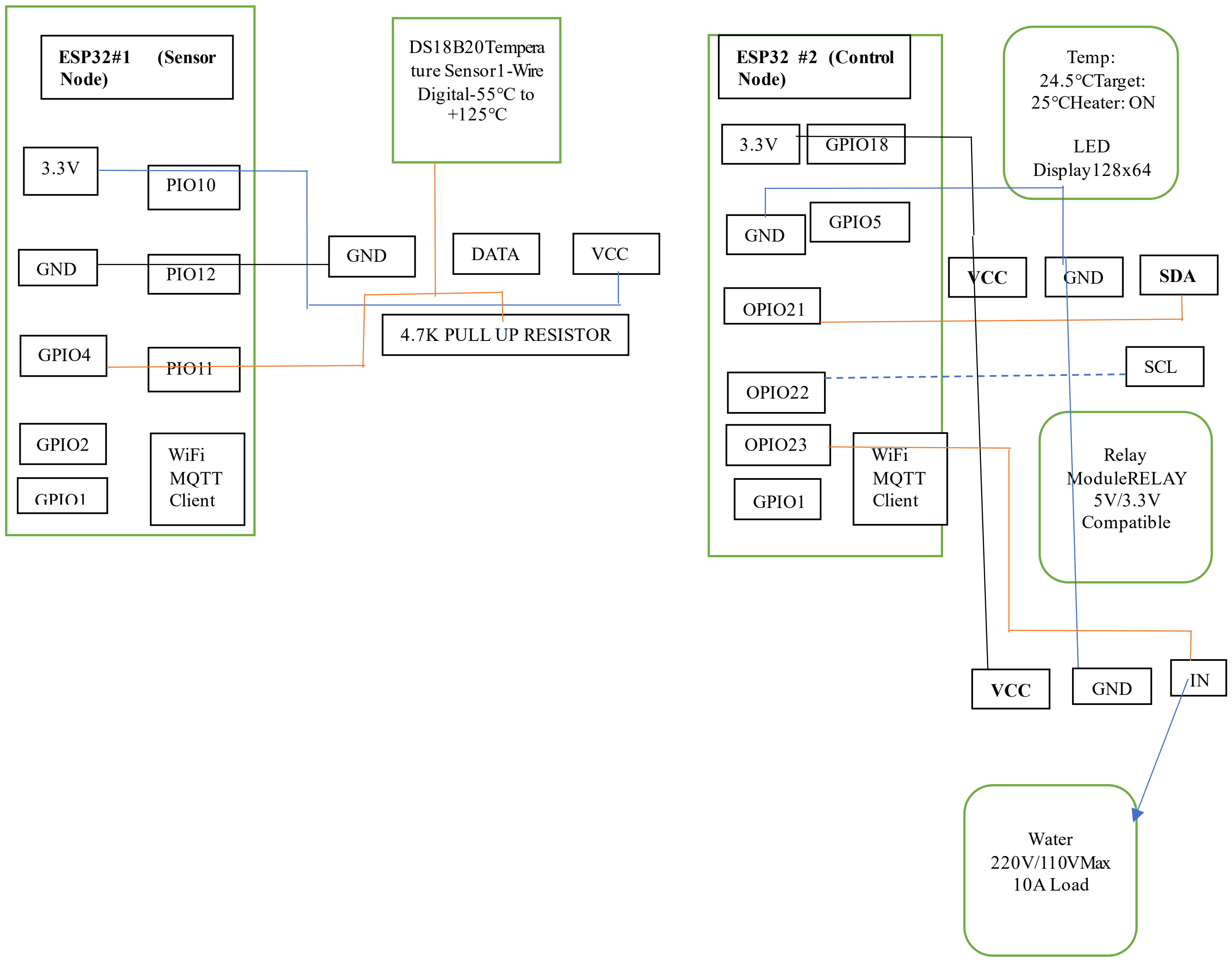
High-Level Block Diagram:



❖ Hardware

Component	Quantity	Justification
ESP32 Boards	2	Dual-processor redundancy for fault tolerance; Node #1 handles sensing, Node #2 handles control
DS18B20 Sensor	1	Waterproof, ±0.5°C accuracy, 1-Wire protocol simplifies wiring
5V Relay	1	Optically isolated for safe AC switching (supports up to 250V/10A)
OLED Display	1	Local system status monitoring (attached to ESP32 #2)

Preliminary Circuit Diagrams



❖ SOFTWARE

ESP32 (C++) Pseudocode for Water Heater Temperature Control System

Pseudocode

cpp

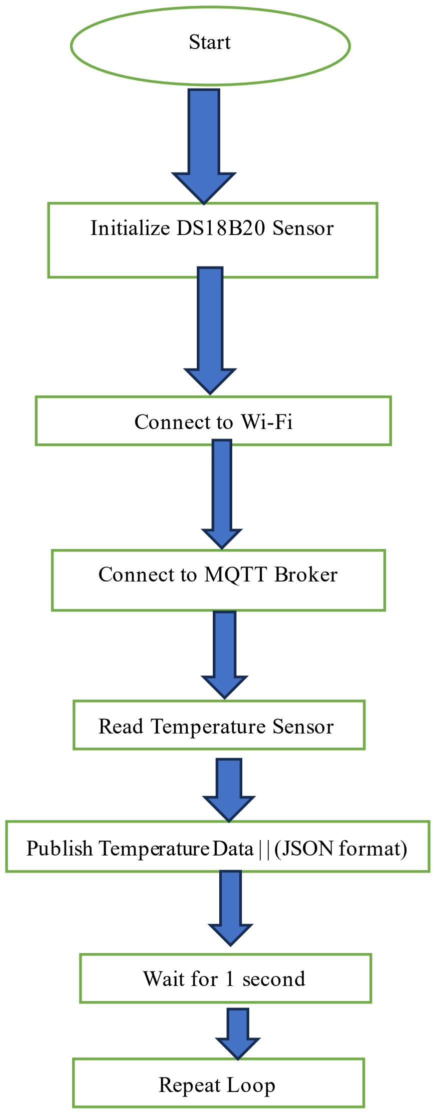
```
// Sensor ESP32 Code


setup() {
   initializeDS18B20();
   connectWiFi(ssid, password);
   connectMQTT(brokerAddress);
}


loop() {
   float temperature = readTemperature();
   String jsonPayload = createJSON(device_id, temperature, currentTimestamp());
   publishMQTT("waterheater/sensor/temperature", jsonPayload);
   delay(1000); // 1 second delay
}
```

cpp

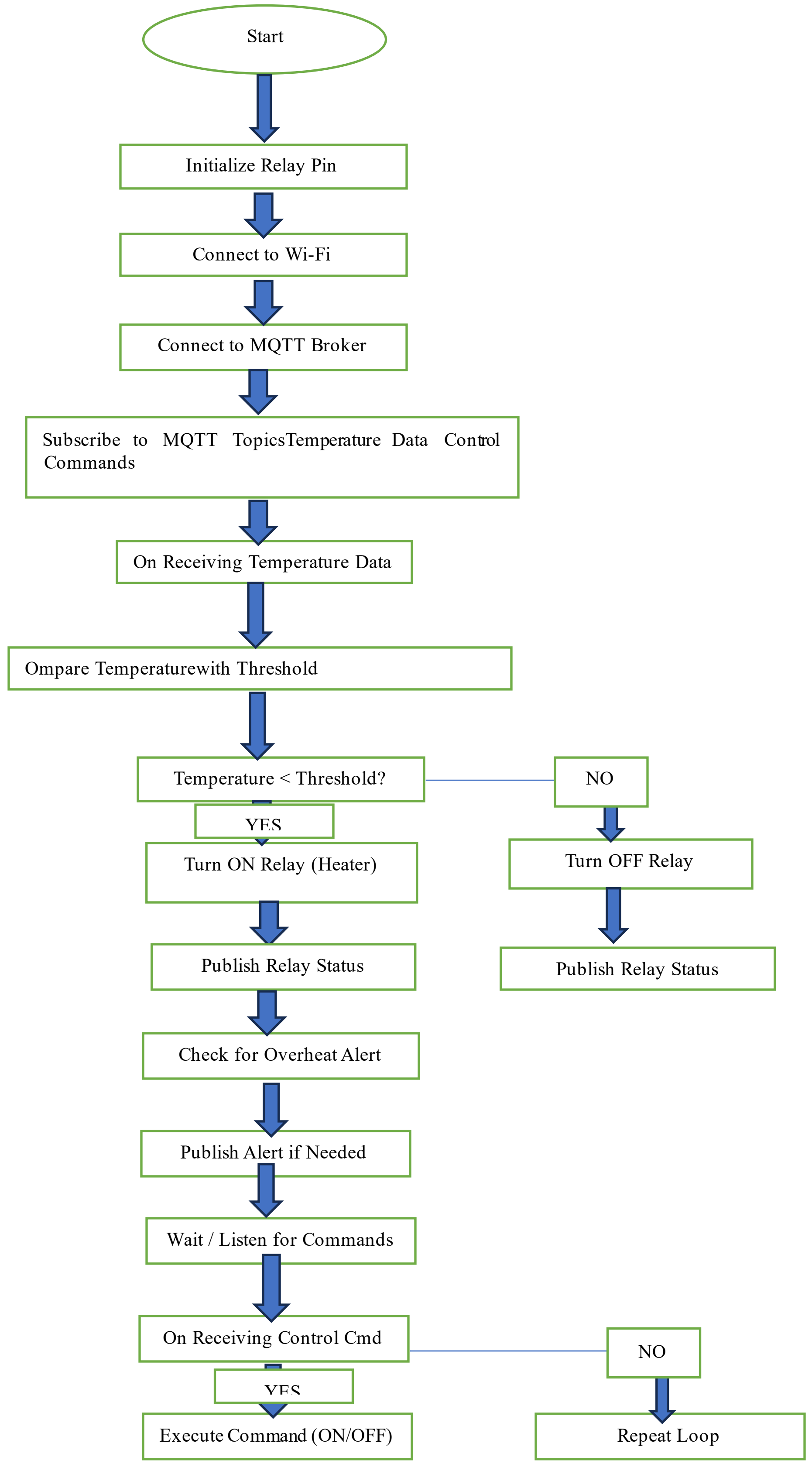
```
// Controller ESP32 Code


setup() {
   initializeRelayPin();
```

```
    connectWiFi(ssid, password);
    connectMQTT(brokerAddress);
    subscribeMQTT("waterheater/sensor/temperature");
    subscribeMQTT("waterheater/control/command");
}

onMQTTMessage(topic, payload) {
    if (topic == "waterheater/sensor/temperature") {
        float temp = parseTemperature(payload);
        if (temp < desiredThreshold) {
            turnRelayOn();
            relayState = "ON";
        } else {
            turnRelayOff();
            relayState = "OFF";
        }
        publishMQTT("waterheater/status/relay",createJSON(relayState,
currentTimestamp()));
        if (temp > safetyLimit) {
            publishMQTT("waterheater/alerts/overheat", createAlertJSON(temp));
        }
    } else if (topic == "waterheater/control/command") {
        String command = parseCommand(payload);
        if (command == "turn_on") {
            turnRelayOn();
            relayState = "ON";
        } else if (command == "turn_off") {
            turnRelayOff();
            relayState = "OFF";
        }
        publishMQTT("waterheater/status/relay",createJSON(relayState,
currentTimestamp()));
    }
}

loop() {
    // Handle MQTT client loop to process incoming messages
    mqttClient.loop();
}
```

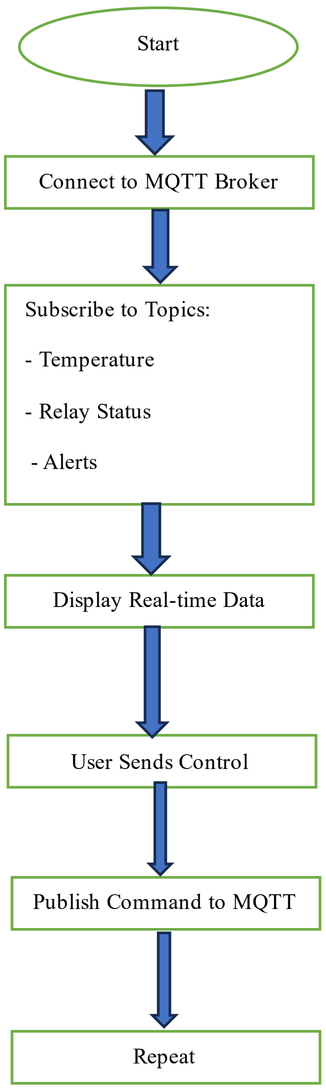
Flowchart 1: Sensor ESP32 - Temperature Sensing and Publishing



Flowchart 2: Controller ESP32 - Relay Control and Monitoring



Flowchart 3: Web Application Interaction (Simplified)



❖ Comms (MQTT/HTTP): Why You Chose It, Topics/Endpoints

For the Water Heater Temperature Control and Monitoring System using two ESP32 boards, MQTT was chosen as the communication protocol due to its lightweight nature, efficiency in low-bandwidth IoT environments, and support for real-time, asynchronous messaging. MQTT's publish-subscribe model allows decoupled communication between the sensor ESP32, control ESP32, and the web application, ensuring reliable and timely data exchange. HTTP was considered less suitable because it is request-response based and incurs higher overhead, making it less efficient for continuous monitoring and control.

❖ MQTT Topics and Endpoints:

1. waterheater/sensor/temperature — Sensor ESP32 publishes temperature readings here.
2. waterheater/control/command — Web app sends heater ON/OFF commands here for the control ESP32 to subscribe.
3. waterheater/status/relay — Control ESP32 publishes relay status updates here.

❖ Data (JSON): Show Your Data Structure

The system uses JSON format to structure data for easy parsing and interoperability.

Temperature Reading Example:

json

```
{
  "device_id": "ESP32_SENSOR_01",
  "timestamp": "2025-06-21T15:00:00Z",
  "temperature_celsius": 55.3
}
```

Control Command Example:

json

```
{
  "command": "turn_on",
  "target_device": "ESP32_CONTROLLER_01"
}
```

Relay Status Example:

json

```
{
  "device_id": "ESP32_CONTROLLER_01",
  "relay_state": "ON",
  "timestamp": "2025-06-21T15:01:00Z"
}
```

❖ Web App (Python): Features, Framework, Design

- **Framework:** Developed using the Flask Python framework for simplicity and flexibility.
- **Features:**
  - Real-time temperature monitoring dashboard.
  - Control interface to turn the water heater ON or OFF remotely.
  - Alert notifications for over-temperature conditions.
  - Historical data visualization for temperature trends.
- **Design:** Responsive web interface accessible from desktop and mobile devices, using MQTT client libraries to subscribe and publish to MQTT topics, enabling seamless interaction with the ESP32 devices.

❖ Network

How It All Connects

- Both ESP32 boards connect to the local Wi-Fi network.
- An MQTT broker (hosted locally or on cloud) manages message routing between devices and the web app.
- The sensor ESP32 publishes temperature data to the broker.
- The control ESP32 subscribes to commands from the web app and controls the relay accordingly.
- The web app subscribes to sensor data and relay status topics to provide real-time updates to the user.

❖ Data Flow

Map the Data's Journey

1. **Temperature Sensing:** Sensor ESP32 reads temperature from DS18B20 sensors.
2. **Data Publishing:** Sensor ESP32 publishes JSON-formatted temperature data to waterheater/sensor/temperature topic on MQTT broker.
3. **Data Reception:** Web app subscribes to temperature topic, receives data, and updates the dashboard.
4. **User Command:** User sends ON/OFF command via web app interface, which publishes JSON command to waterheater/control/command topic.
5. **Command Execution:** Control ESP32 subscribes to command topic, receives command, and actuates the relay to control the water heater.
6. **Status Update:** Control ESP32 publishes relay status to waterheater/status/relay topic, which the web app subscribes to for real-time status display.

7. **Alerts:** If temperature exceeds threshold, control ESP32 publishes alert messages to waterheater/alerts/overheat topic for immediate user notification.

6.Implementation Plan & Timeline

❖ Tasks and Responsibilities

Task	Description	Assigned To
Requirement Analysis	Define system requirements, hardware and software needs	Team Member 1& 2
Hardware Procurement and Setup	Purchase ESP32 boards, DS18B20 sensors, relay modules, power supplies	Team Member 1& 2
ESP32 Sensor Node Programming	Develop code for temperature sensing and MQTT publishing	Team Member 1& 2
ESP32 Controller Node Programming	Develop relay control, MQTT subscription, and command handling	Team Member 1& 2
MQTT Broker Setup and Testing	Configure MQTT broker, test communication between ESP32 nodes and web app	Team Member 1& 2
Web Application Development	Design and implement Python Flask app for monitoring and control	Team Member 1& 2
Integration and System Testing	Integrate hardware and software components, perform functional testing	Team Member 1& 2
Documentation and Report Preparation	Prepare final report, user manual, and presentation	Team Member 1 & 2

❖ Detailed Schedule

Week	Task
1	Requirement analysis, hardware setup
1	ESP32 sensor node programming
2	ESP32 controller node programming
2	MQTT broker setup and communication testing
2	Web app development and integration
3	System testing and debugging
3	Documentation and final report

❖ Gantt Chart

Task Name	Week 01	Week 02	Week 03
Requirement Analysis	⬛		
Hardware Procurement and Setup	⬛		
ESP32 Sensor Node Programming		🟦	
ESP32 Controller Node Programming		🟦	
MQTT Broker Setup and Testing, Web Application Development			🟥
Integration and System Testing			🟥

Documentation and Report Preparation			🟥

7.Expected Outcomes & Deliverables

- **Prototype:** Fully functional water heater temperature control and monitoring system using two ESP32 boards.
- **Source Code:** Complete ESP32 firmware for sensor and controller nodes, and Python web application code.
- **Documentation:** Detailed project report including system design, implementation details, testing results, and user guide.
- **Presentation:** Final presentation demonstrating system functionality and project learnings.

8.Budget/Resources

Item	Quantity	Estimated Cost (Rs)	Purpose
ESP32 Development Boards	2	Rs. 2,200.00.	Microcontroller units for sensing and control
DS18B20 Temperature Sensors	2-3	Rs. 990.00	Accurate water temperature measurement
Relay Modules	1-2	Rs.260.00	Control heating element power supply
Power Supply Units	1-2	Rs.860.00	Power ESP32 and relay modules
Miscellaneous Components	-	Rs.500.00	Wires, connectors, breadboards, enclosures

Software Tools	-	Free/Open-source	Arduino IDE, Python, MQTT broker software

9.References

1. Arduino Expert. "Water Heater Temperature Monitoring and Control by Using ESP32 IoT." 2025.
2. Instructables. "ESP32 IoT Water Heater." 2024.
3. Vishvas Natu, Vedant Hinge, Harsh Shankar. "Design and Implementation of an Automated Geyser Temperature Control System Using ESP32." International Journal of Research in Advent Technology, 2025. DOI: 10.22214/ijraset.2025.72481
4. Reddit. "HA ESP32 Dallas Temp Sensor Tado Hot Water Control." /r/homeassistant, 2024.
5. GitHub. "Smart Solar Heating Project." UnknownHero99, 2021.
6. SSRN. "ESP32 Temperature Control Project With An Emphasis On Energy Efficiency." 2024.