SCHOOL OF COMPUTING

UUM COLLEGE OF ARTS AND SCIENCES

STTHK3113 SENSOR-BASED SYSTEMS

SEMESTER 6 (A242)

INDIVIDUAL ASSIGNMENT :

MIDTERM EXAM A242

LECTURED BY :

AHMAD HANIS BIN MOHD SHABIL

PREPARED BY :

NAME	MATRIC NO.
NURSYASYA AINA BINTI ABDUL HAMID	295153

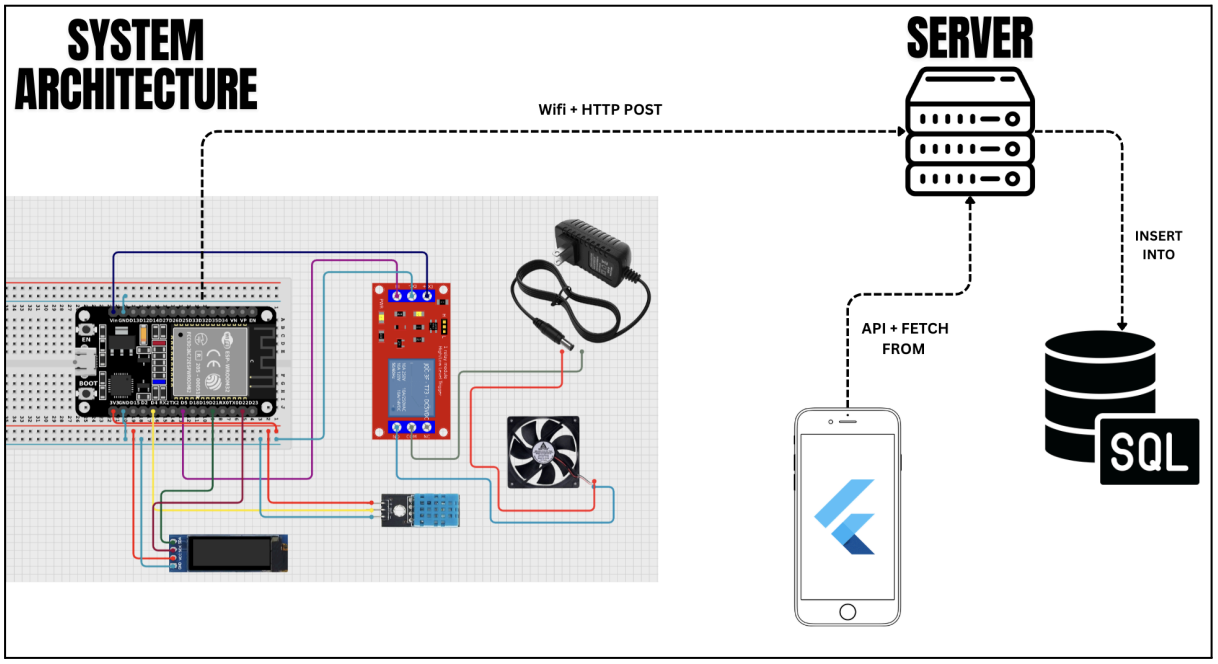
SUBMISSION DATE :

30 MAY 2025

1.0 INTRODUCTION

This project, SmartSense, is an IoT-based environmental monitoring system developed. The system uses an ESP32 microcontroller and a DHT11 sensor to continuously measure temperature and humidity. It features real-time display on an OLED screen, automatic control of a relay for devices like a fan, and remote data uploading to a web server. WiFi and user setup are simplified through a captive portal that can be accessed from any phone or laptop.

2.0 SYSTEM ARCHITECTURE



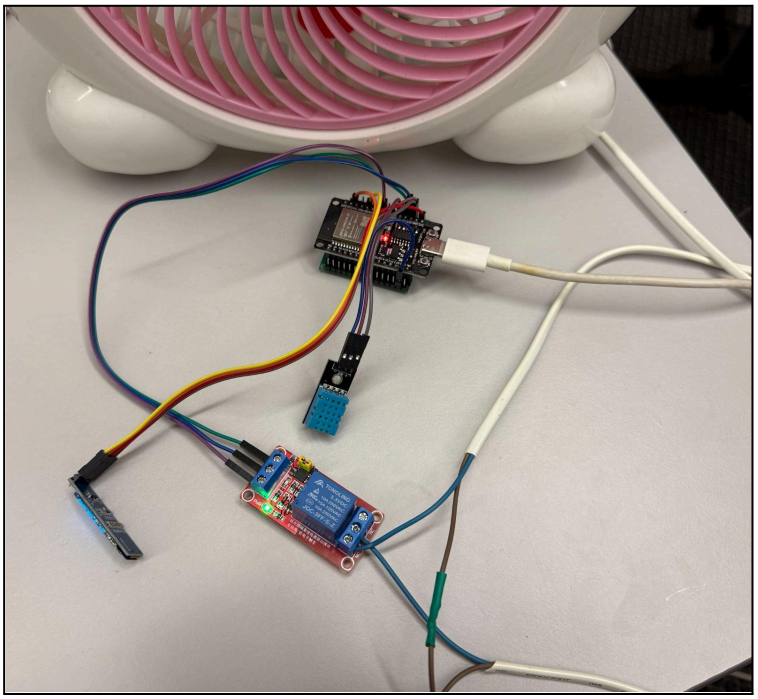
This diagram illustrates the complete flow of the IoT monitoring system:

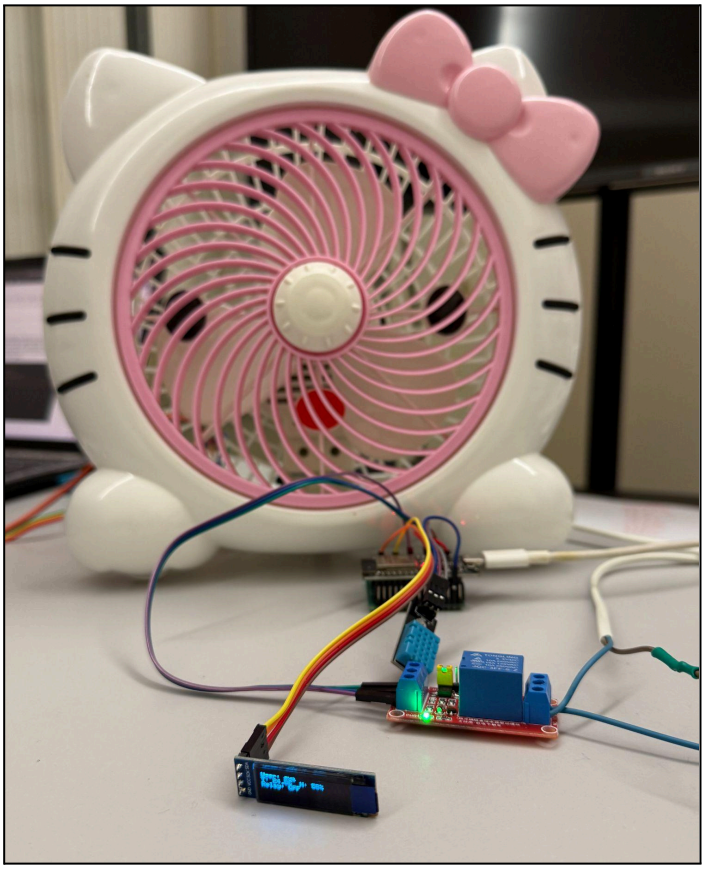
- ESP32 DevKit V1 acts as the main controller, collecting temperature and humidity data from the DHT11 sensor, displaying the info on the OLED, and controlling a relay (for the fan).
- Relay Module allows the ESP32 to switch the fan on or off based on sensor readings.
- WiFi Adapter (built-in ESP32) uploads sensor data every 10 seconds via HTTP POST to a remote server.
- Server (running PHP + MySQL) receives and stores the data, making it accessible for analysis and visualization.
- SQL Database logs every reading with a timestamp.
- Flutter Mobile App fetches the latest sensor data from the server API and displays it with graphs and status indicators for the user.

3.0 STEP GUIDELINES

3.1 HARDWARE SETUP

- **ESP32 DevKit V1:** Main controller for data collection and automation.
- **DHT11 Sensor:** Measures temperature and humidity.
- **0.91" OLED SSD1306 (I2C):** Displays live data and user info.
- **Relay Module:** Switches external devices (like a fan).





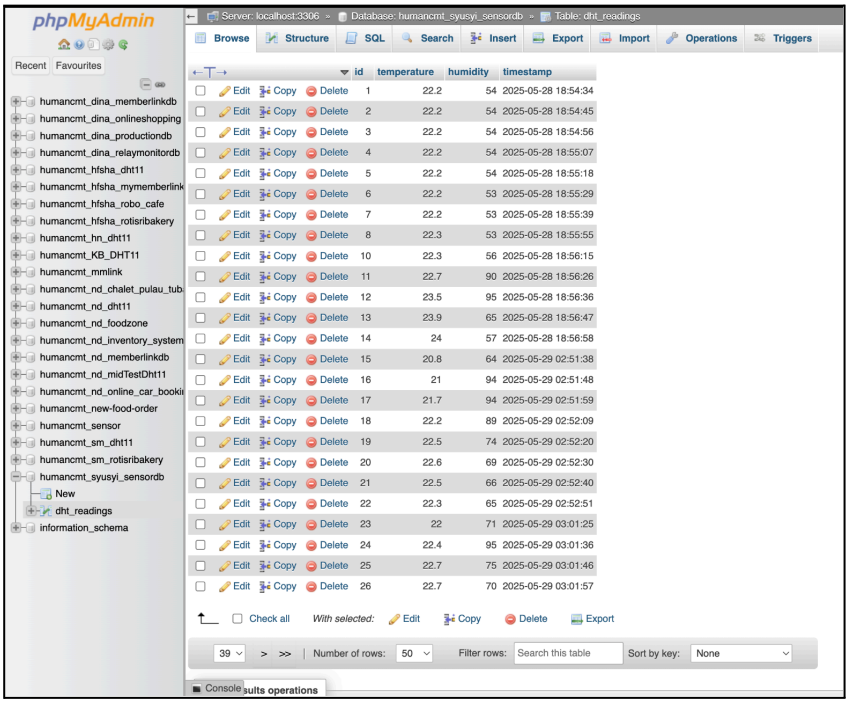
3.2 SOFTWARE SETUP

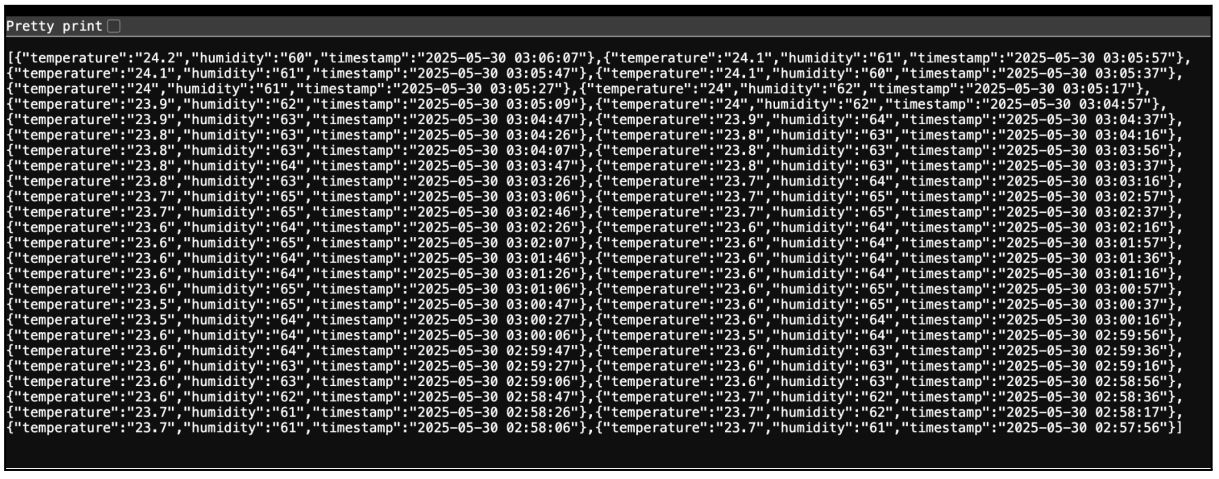
- The ESP32 was programmed using Arduino IDE and several essential libraries (WiFi, EEPROM, DHT, Adafruit SSD1306).
- WiFi credentials and the username are saved in EEPROM for persistence after resets.
- A built-in captive portal lets users easily set up WiFi and username using a phone or laptop.
- Every 10 seconds, the ESP32 reads data from the DHT11 sensor and sends temperature and humidity readings to a backend server using HTTP POST.
- The OLED screen shows the username, sensor readings, and the relay status in real-time.
- The relay is controlled automatically:
    - For example, it turns ON if the temperature > 26°C or humidity > 70%.
    - otherwise, it stays OFF.

```
281
282     float temp = dht.readTemperature();
283     float hum = dht.readHumidity();
284
285     if (isnan(temp) || isnan(hum)) {
286       Serial.println("❌ Failed to read from DHT sensor!");
287       showOLED("DHT11 Error", "Check wiring", "");
288       return;
289     }
290
291     temperature = String(temp, 1);
292     humidity = String(hum, 0);
293
294     sendToPHP(temp, hum);
295
296     bool isRelayOn = (temp > 26 || hum > 70);
297     digitalWrite(RELAY_PIN, isRelayOn ? HIGH : LOW);
298
299     String relayStatus = isRelayOn ? "Relay: ON" : "Relay: OFF";
300     showOLED(
301       "User: " + username,
302       "T: " + temperature + "C  H: " + humidity + "%",
303       relayStatus
304     );
305   }
```

3.3 BACKEND SETUP

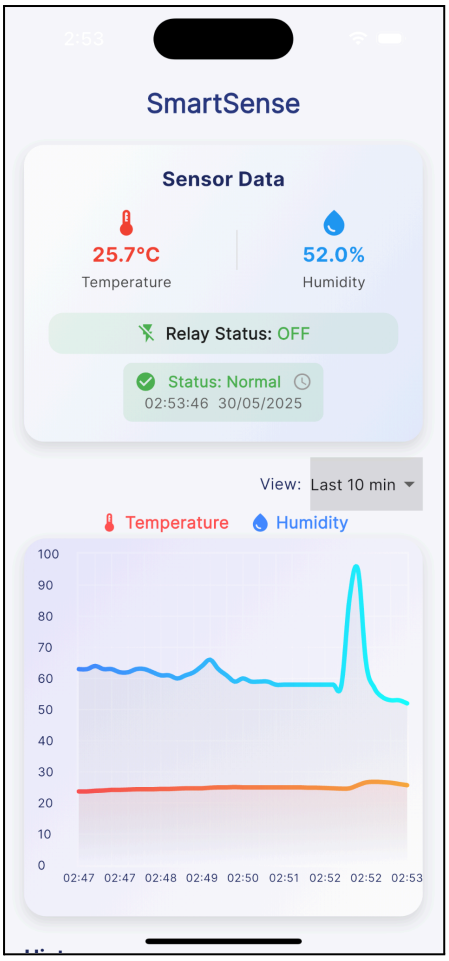
- Platform: Uses PHP as the backend language with a MySQL database for storing data.
- Main API Scripts:
    - **dbconnect.php**: Manages the database connection details and credentials for the PHP scripts, so they can access the data.
    - **submit.php**: Accepts POST data sent from the ESP32 (temperature and humidity) and logs each reading into the MySQL table.
    - **fetch.php**: Provides the most recent 50 sensor readings in JSON format. The mobile app relies on this to show live graphs and history.
- SQL Database:
  All sensor readings are stored in a structured table for further analysis and retrieval.

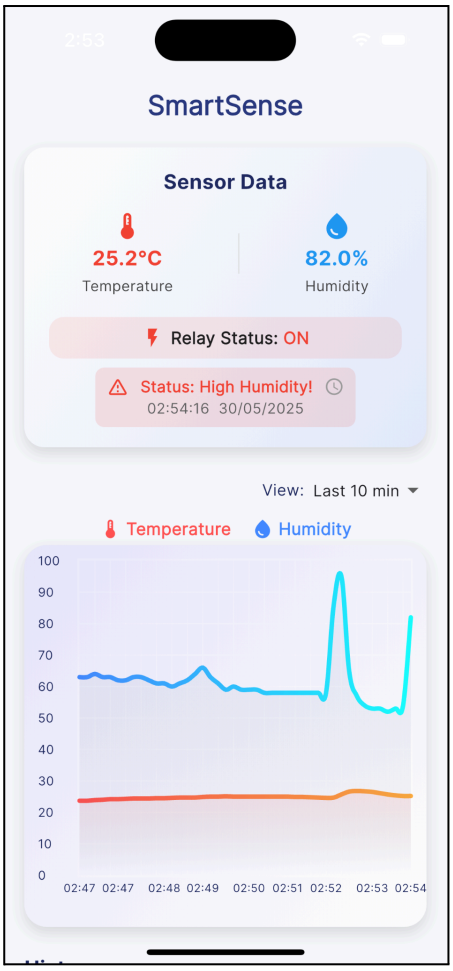


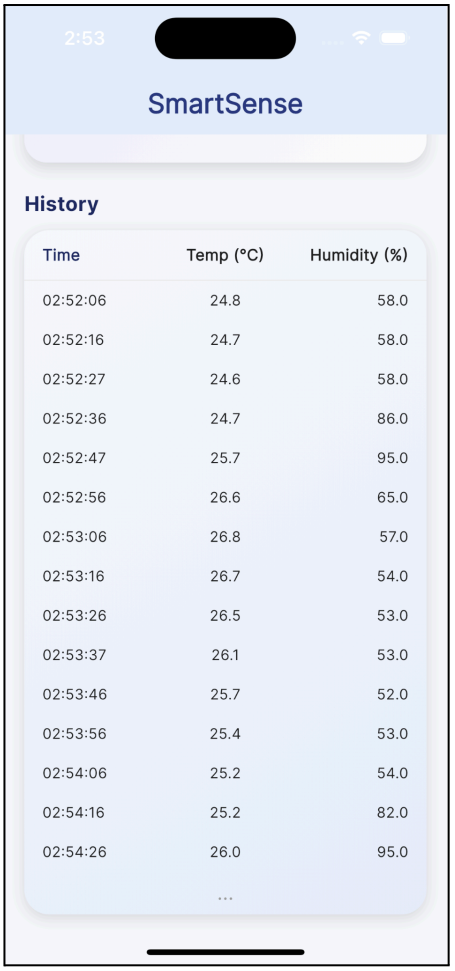


3.4 MOBILE APP

- Flutter app displays real-time and historical sensor data
- Shows relay status and alerts for high temperature/humidity

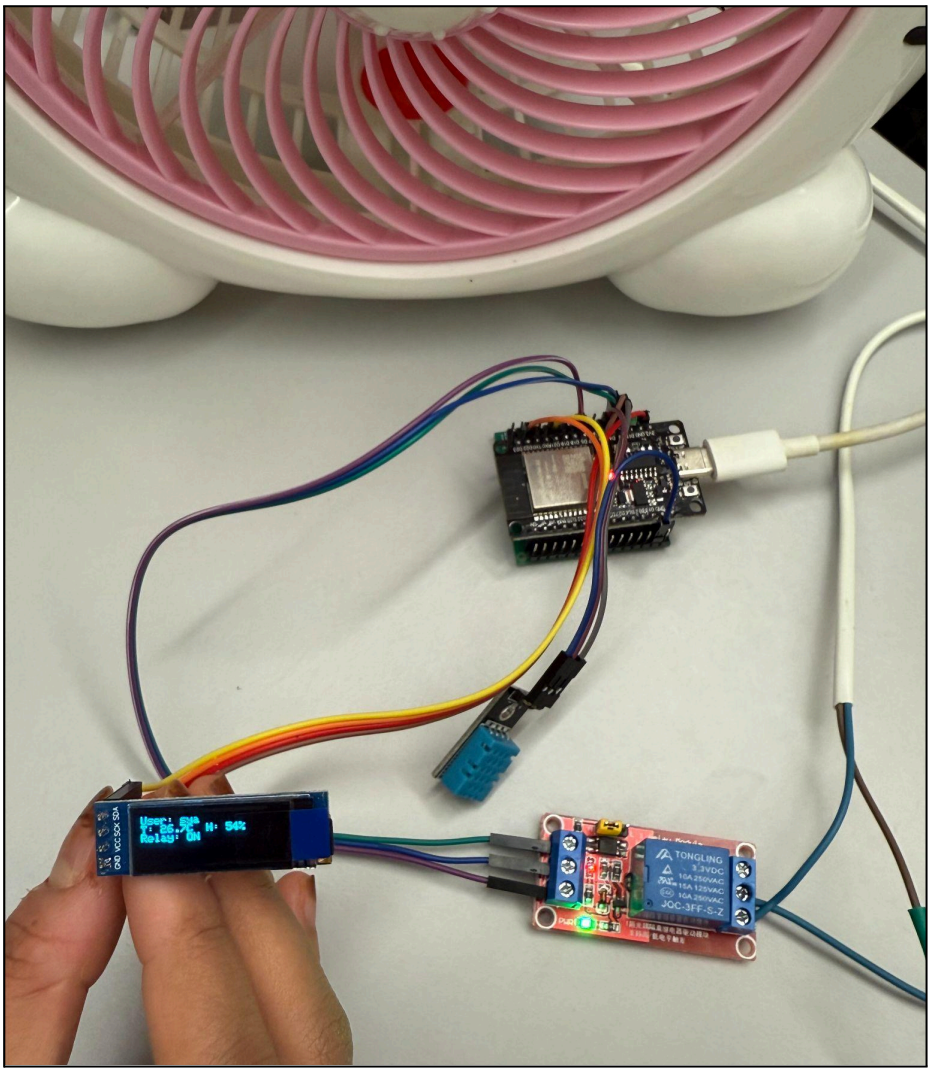






3.5 SYSTEM FLOW

1. ESP32 powers up, if no WiFi, starts AP mode for setup.
2. User connects and enters WiFi/username.
3. ESP32 uploads sensor data to the server.
4. Server stores the data of the temperature and humidity from the dht11 sensor.
5. Mobile app fetches and visualizes information of the data stored.
6. OLED displays username, readings, and relay status whether it is on or off.



4.0 CHALLENGES AND IMPROVEMENTS

Challenges:

- Making sure all wiring, especially for the relay and DHT11 sensor, was correct to avoid damaging the ESP32 or components.
- The DHT11 sensor sometimes gave inaccurate or inconsistent readings, especially at high humidity levels.
- Powering both the fan and relay from the same adapter required careful planning to prevent voltage drops or resets.
- Integrating the ESP32 with the backend server (PHP + MySQL) involved troubleshooting data formatting and communication issues.

Improvements:

- Use a DHT22 or another more accurate sensor for better temperature and humidity readings.
- Add notification features in the mobile app to alert users of high temperature or humidity.
- Implement user authentication for better data security and access control.

5.0 DEMONSTRATION VIDEO

Youtube Video Link : https://youtu.be/ePdiog3kFaY
Github Link : https://github.com/syasy00/Temperature-and-Humidity-Monitoring-Project.git