UUM

Universiti Utara Malaysia

SCHOOL OF COMPUTING
COLLEGE OF ARTS AND SCIENCES

SEMESTER A242 SESSION 2024/2025

STTHK3113 SENSOR-BASED SYSTEMS

MIDTERM REPORT

PREPARED FOR
AHMAD HANIS BIN MOHD SHABLI

PREPARED BY

NAME	MATRIC
MUHAMMAD KHAIRUL INAS BIN SHAIFULRIZAL	294690

1. Objective

To develop a sensor-based monitoring system that:

- Captures temperature and humidity data every 10 seconds
- Stores the data in a SQL relational database
- Activates a relay if thresholds (user can set) are exceeded
- Displays real-time readings in a graph on a mobile/web app

2. Backend

2.1. Backend Functions

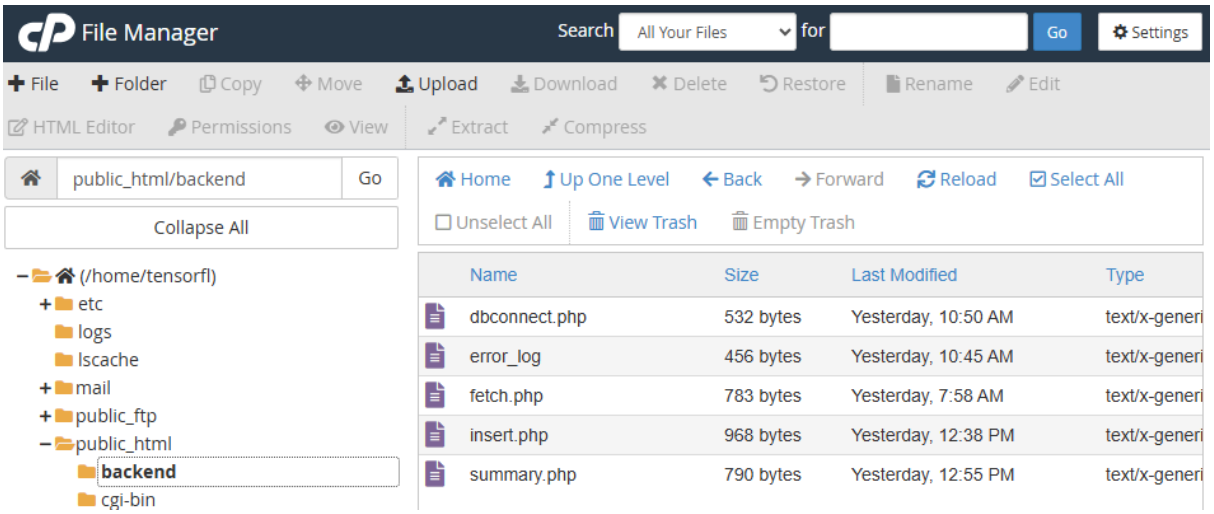

Figure 2.0: List Of Backend PHP Files

2.1.1. Accept sensor data every 10 seconds via API

```php
// Read POST body
$data = json_decode(json: file_get_contents(filename: "php://input"), associative: true);
```

Figure 2.1: Accept sensor data via API (insert.php)

2.1.2. Store data with timestamp

```php
// Insert into database
$stmt = $conn->prepare(query: "INSERT INTO sensor_data (temperature, humidity, timestamp) VALUES (?, ?, NOW())");
$stmt->bind_param(types: "dd", var: &$temp, vars: &$hum);

if ($stmt->execute()) {
    echo json_encode(value: [
        "success" => true,
        "message" => "Data inserted successfully."
    ]);
} else {
    echo json_encode(value: [
        "success" => false,
        "message" => "Insert failed: " . $stmt->error
    ]);
}
```

Figure 2.2: PHP Inserts Temperature and Humidity into Database (insert.php)

2.1.3. Provide endpoint to fetch data for graphing

```php
// Validate limit
if ($limit <= 0 || $limit > 1000) {
    $limit = 100;
}

// Query from database
$sql = "SELECT temperature, humidity, timestamp FROM sensor_data ORDER BY timestamp DESC LIMIT ?";
$stmt = $conn->prepare(query: $sql);
$stmt->bind_param(types: "i", var: &$limit);
$stmt->execute();

$result = $stmt->get_result();
```

Figure 2.3: Fetch Most Recent N Records (fetch.php)

3. Mobile / Web App

3.1. Fetches temperature and humidity data from backend

```dart
Future<void> fetchSensorData() async {
  final response = await http.get(
    Uri.parse('https://tensorflowtitan.xyz/backend/fetch.php'),
  );
```

Figure 3.0: Pull Sensor Data from PHP Backend (mainscreen.dart)

3.2. Displays values in a graph/chart (Use packages like fl_chart, syncfusion_flutter_charts, or equivalent)

```dart
import 'package:flutter/material.dart';
import 'package:fl_chart/fl_chart.dart';
import 'summaryscreen.dart';
import 'dart:convert';
import 'package:http/http.dart' as http;
```

Figure 3.1: Use LineChart to Plot temperatureData & humidityData (mainscreen.dart)

3.3. Optional: Display latest value or status message (e.g., “Alert: High Temp!”)

```dart
statusMessage =
    lastTemp > 35.0 ? "⚠️ Alert: High Temp!" : "✅ Temperature Normal";
humidityStatus =
    lastHum > 97 ? "⚠️ High Humidity!" : "✅ Humidity Normal";
```

Figure 3.2: Indicator Display (mainscreen.dart)

4. System Behaviour

4.1. Data Interval: Every 10 seconds, ESP32 sends:

```
// Read + send every 10s
if (millis() - lastUpdate > 10000) {
  lastUpdate = millis();

  float temp = readTemperature();
  float hum = readHumidity();
  checkAndTriggerRelay(temp, hum);
  postSensorData(temp, hum);
}

// Blink OLED status every 1s
if (millis() - lastBlink > 1000) {
  lastBlink = millis();
  blinkOn = !blinkOn;

  float temp = readTemperature();
  float hum = readHumidity();
  bool relayActive = digitalRead(RELAY_PIN);
```

Figure 4.0: Send every 10 seconds (arduino.ino)

4.2. Relay Trigger (user able to configure): Activated when:

```
#define TEMP_THRESHOLD 35.0
#define HUMIDITY_THRESHOLD 97.0

DHT dht(DHTPIN, DHTTYPE);

// Initialize DHT sensor and relay pin
void initDHT() {
  dht.begin();
  pinMode(RELAY_PIN, OUTPUT);
  Serial.println("DHT11 & Relay initialized");
}

// Read temperature in Celsius
float readTemperature() {
  return dht.readTemperature();
}

// Read humidity in percentage
float readHumidity() {
  return dht.readHumidity();
}

// Check threshold and activate relay if needed
bool checkAndTriggerRelay(float temp, float hum) {
  bool trigger = temp > TEMP_THRESHOLD || hum > HUMIDITY_THRESHOLD;
  digitalWrite(RELAY_PIN, trigger ? HIGH : LOW);  // Relay ON if HIGH (check if yours is active HIGH or LOW)
  return trigger;
}
```

Figure 4.1: Relay Trigger

4.3. App graph must reflect these values clearly

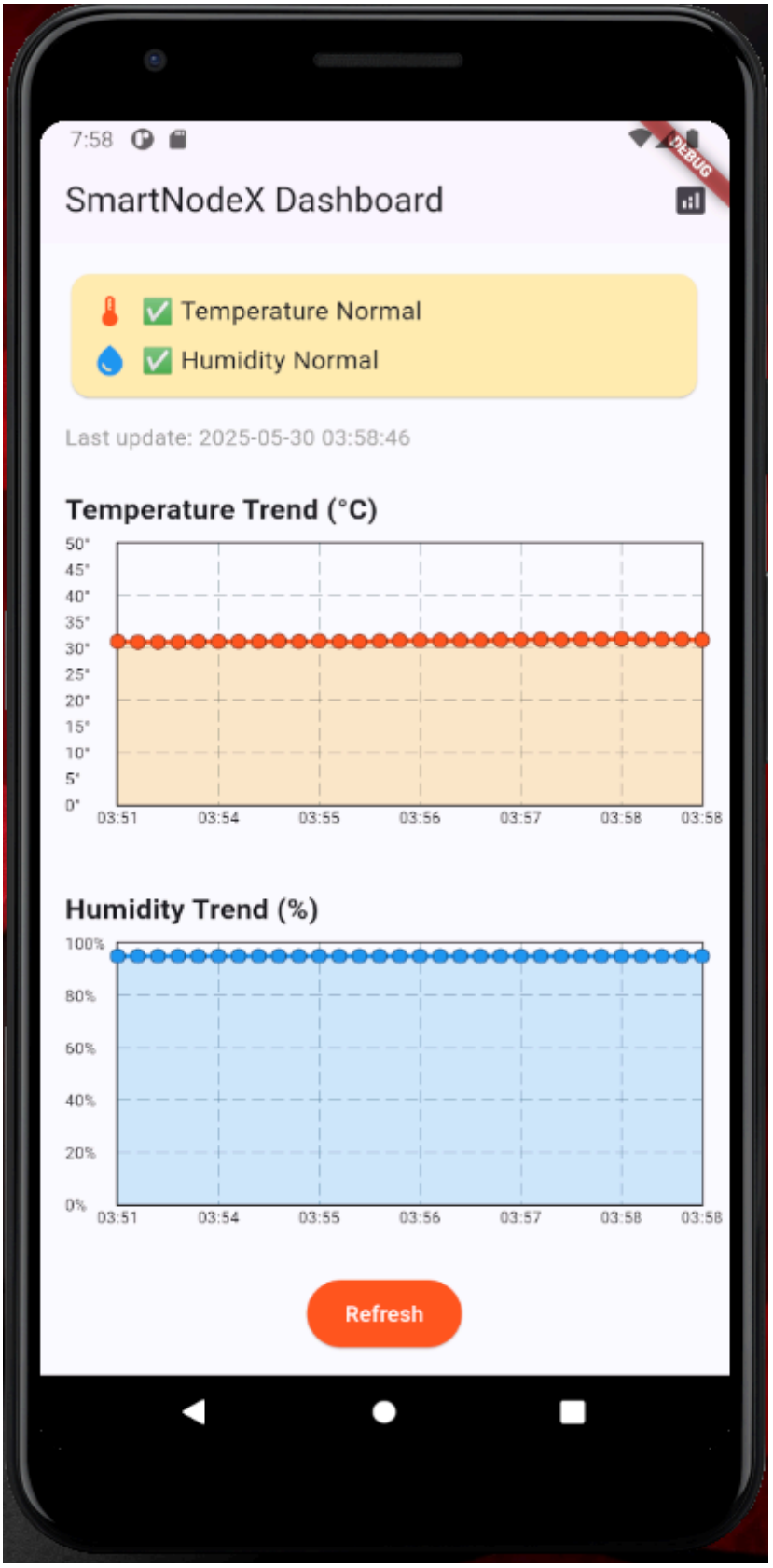

Figure 4.2: Graph Visualization in Mobile App

5. Challenges and Improvements

- **Real-Time Data Handling**

  One challenge was handling real-time data smoothly. The system needed to send and display sensor readings every 10 seconds without lag. To solve this, millis() was used on the ESP32 for timing, and setState() was used in Flutter to update the screen without freezing.

- **Data Synchronization Between ESP32 and Backend**

  Another challenge was making sure the ESP32 could send data to the backend without errors. Sometimes, bad network connections or incorrect JSON caused data loss. This was fixed by checking HTTP responses and making sure the data was properly formatted.

- **Relay Threshold Control**

  The last challenge was the relay control. The temperature and humidity limits were hardcoded, so changing them meant reprogramming the ESP32. A better way would be to use a web page or Firebase so users can change the limits easily.

6. Links

Youtube Links: https://youtu.be/c5kjIi7bbTY
Github Links: https://github.com/krulinas/SmartNodeX