UNIVERSITI UTARA MALAYSIA

COLLEGE OF ARTS AND SCIENCES (CAS)

SCHOOL OF COMPUTING (SOC)

FIRST SEMESTER 2024/2025 (A241)

STTH3113 Sensor-Based System (A242)

GROUP (A)

Midterm Exam

SUBMITTED TO:

Ahmad Hanis Bin Mohd Shabli

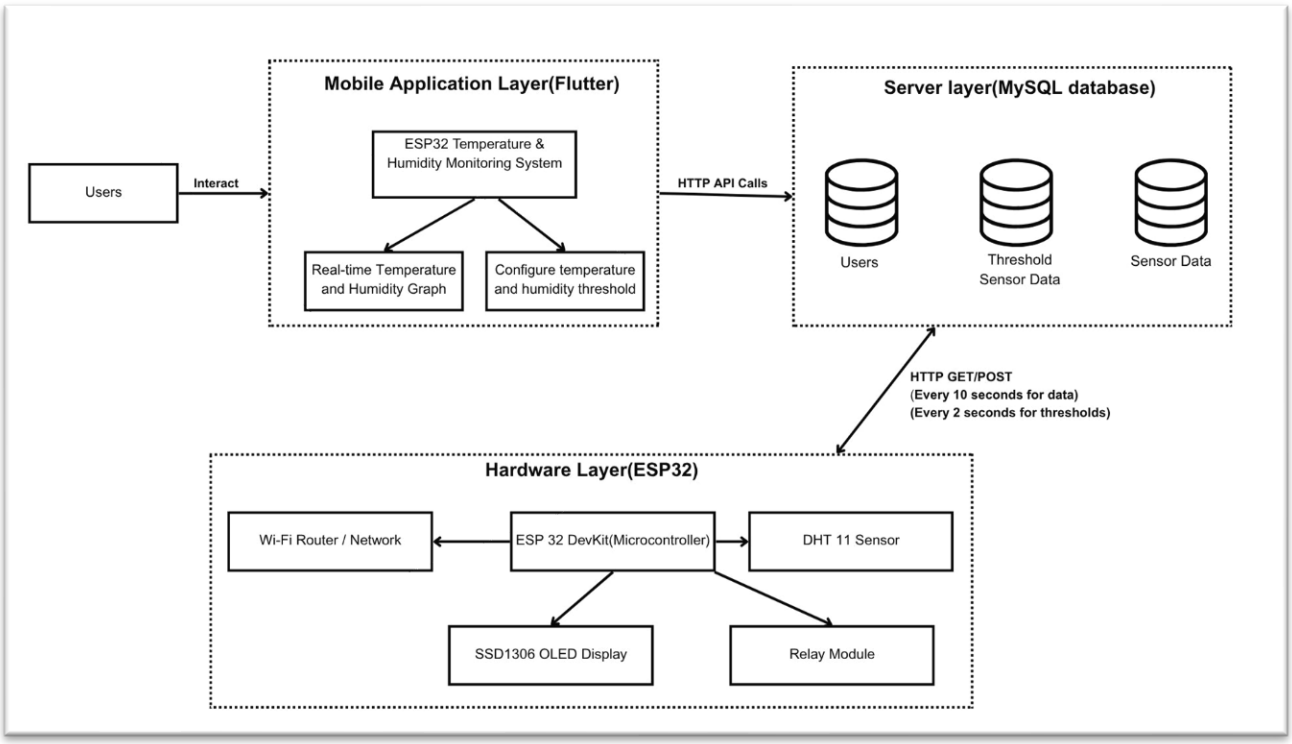
SUBMITTED BY:

KOY CHANG WEI 295972

SUBMISSION DATE:

30 MAY 2025

1.0 System Architecture Diagram



2.0 Setup & Deployment Steps

2.1 Hardware

Components Used:

- ESP32 DevKit microcontroller
- DHT11 temperature & humidity sensor
- SSD1306 OLED display (128x32, I2C)
- Relay module (5V)
- Trainingboard, jumper wires, power supply

ESP32 Pinout:
- GPIO 4 → DHT11 Data
- GPIO 21 → OLED SDA
- GPIO 22 → OLED SCL
- GPIO 25 → Relay IN
- 3.3V → VCC (Sensors)
- GND → GND (Common)

Assembly Steps:

1. Connect the DHT11 sensor's data pin to GPIO 4, positive pin to 3.3V, and negative pin to GND.
2. Connect the OLED display's SDA to GPIO 21, SCK to GPIO 22, VCC to 3.3V and GND to GND.
3. Connect the relay module's IN to GPIO 25, DC+ to 3.3V, and DC- to GND.
4. Double-check all connections for correctness and power on the ESP32.

2.2 Frontend(Mobile App)

Technology: Flutter (Dart)Setup Steps:

1. Install Flutter SDK and required dependencies.
2. Clone the project repository.
3. Navigate to the project directory and run: "flutter pub get"
4. Update the server URL in "lib/myconfig.dart":

```
class MyConfig {
  static const String server = "http://your-server-domain.com";
}
```

5. Connect a mobile device or start an emulator.

Run the app: “ flutter run”

7. Register a new user or log in to access the dashboard, real-time charts, and threshold configuration.

2.3 Backend

Technology: PHP (7.4+), MySQL (8.0+). Hosted on JomHostingSetup Steps:

1. Upload all PHP files from the server/ directory to your JomHosting.
2. Create a new MySQL database and user.
3. Import the required tables:

- tbl_dht (sensor data)
- tbl_threshold (threshold configuration)
- users (authentication)

4. Update database credentials in server/dbconnect.php:

```
$servername = "localhost";
$username = "your_db_username";
$password = "your_db_password";
$dbname = "your_database_name";
```

5. Ensure the hosting supports PHP and allows remote connections from the ESP32.
6. Test API endpoints using a browser or curl to confirm the backend is running.

3.0 Screenshots (backend, relay, app)

3.1 Backend

dbconnect.php

```
<?php
$servername = "localhost";
$username   = "threenqs_koy_chang_wei";
$password   = "6gspmd70(**O";
$dbname     = "threenqs_iottraining_db_kcw";

$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
?>
```

dht11.php

```
<?php
include_once("dbconnect.php");

// get parameters (fall back to 0 if relay not provided)
$temp  = isset($_GET['temp'])  ? $_GET['temp']  : '';
$hum   = isset($_GET['hum'])   ? $_GET['hum']   : '';
$relay = isset($_GET['relay']) ? $_GET['relay'] : 0;

// make sure you have a `relay_status` column in your table
$sql = "
  INSERT INTO `tbl_dht`
    (`temperature`, `humidity`, `relay_status`)
  VALUES
    ('$temp', '$hum', '$relay')
";

if ($conn->query($sql) === TRUE) {
    echo "success";
} else {
    echo "failed: " . $conn->error;
}

$conn->close();
?>
```

get_sensor_data.php

```php
<?php
header('Content-Type: application/json');
header('Access-Control-Allow-Origin: *');
header('Access-Control-Allow-Methods: GET, POST, OPTIONS');
header('Access-Control-Allow-Headers: Content-Type');

include_once("dbconnect.php");

try {
    // Get the limit parameter from query string, default to 50
    $limit = isset($_GET['limit']) ? intval($_GET['limit']) : 50;

    // Validate limit (between 1 and 500)
    if ($limit < 1) {
        $limit = 50;
    } elseif ($limit > 500) {
        $limit = 500;
    }

    // Get the total count of all records in the table
    $countStmt = $conn->prepare("SELECT COUNT(*) as total_count FROM tbl_dht");
    $countStmt->execute();
    $countResult = $countStmt->get_result();
    $totalCount = $countResult->fetch_assoc()['total_count'];

    // Get the records with the specified limit ordered by timestamp
    $stmt = $conn->prepare("SELECT id, temperature, humidity, timestamp, relay_status FROM tbl_dht ORDER BY timestamp DESC LIMIT ?");
    $stmt->bind_param("i", $limit);
    $stmt->execute();
    $result = $stmt->get_result();

    $sensorData = [];
    while ($row = $result->fetch_assoc()) {
        $sensorData[] = [
            'id' => $row['id'],
            'temperature' => floatval($row['temperature']),
            'humidity' => floatval($row['humidity']),
            'timestamp' => $row['timestamp'],
            'relay_status' => $row['relay_status']
        ];
    }

    // Reverse the array to show chronological order (oldest first)
    $sensorData = array_reverse($sensorData);

    echo json_encode([
        'status' => 'success',
        'data' => $sensorData,
        'count' => count($sensorData),
        'total_records' => intval($totalCount),
        'limit' => $limit
    ]);

} catch (Exception $e) {
    echo json_encode([
        'status' => 'error',
        'message' => 'Database error: ' . $e->getMessage()
    ]);
}

$conn->close();
?>
```

get_threshold.php

```php
<?php
header('Content-Type: application/json');
header('Access-Control-Allow-Origin: *');
header('Access-Control-Allow-Methods: GET, POST, OPTIONS');
header('Access-Control-Allow-Headers: Content-Type');

include_once("dbconnect.php");

try {
    // Get the threshold values from the database
    $stmt = $conn->prepare("SELECT sensor_id, sensor_name, threshold_temp, threshold_humidity, timestamp FROM tbl_threshold_relay ORDER BY sensor_id");
    $stmt->execute();
    $result = $stmt->get_result();

    $thresholds = [];
    while ($row = $result->fetch_assoc()) {
        $thresholds[] = [
            'sensor_id' => intval($row['sensor_id']),
            'sensor_name' => $row['sensor_name'],
            'threshold_temp' => floatval($row['threshold_temp']),
            'threshold_humidity' => floatval($row['threshold_humidity']),
            'timestamp' => $row['timestamp']
        ];
    }

    // If no thresholds exist, create default ones
    if (empty($thresholds)) {
        $defaultThresholds = [
            [
                'sensor_id' => 1,
                'sensor_name' => 'dht11',
                'threshold_temp' => 26.0,
                'threshold_humidity' => 70.0
            ]
        ];

        foreach ($defaultThresholds as $threshold) {
            $insertStmt = $conn->prepare("INSERT INTO tbl_threshold_relay (sensor_id, sensor_name, threshold_temp, threshold_humidity) VALUES (?, ?, ?, ?)");
            $insertStmt->bind_param("isdd", $threshold['sensor_id'], $threshold['sensor_name'], $threshold['threshold_temp'], $threshold['threshold_humidity']);
            $insertStmt->execute();
            $insertStmt->close();
        }

        // Fetch the newly created thresholds
        $stmt = $conn->prepare("SELECT sensor_id, sensor_name, threshold_temp, threshold_humidity, timestamp FROM tbl_threshold_relay ORDER BY sensor_id");
        $stmt->execute();
        $result = $stmt->get_result();

        while ($row = $result->fetch_assoc()) {
            $thresholds[] = [
                'sensor_id' => intval($row['sensor_id']),
                'sensor_name' => $row['sensor_name'],
                'threshold_temp' => floatval($row['threshold_temp']),
                'threshold_humidity' => floatval($row['threshold_humidity']),
                'timestamp' => $row['timestamp']
            ];
        }
    }

    echo json_encode([
        'status' => 'success',
        'data' => $thresholds
    ]);

} catch (Exception $e) {
    echo json_encode([
        'status' => 'error',
        'message' => 'Database error: ' . $e->getMessage()
    ]);
}

$conn->close();
?>
```

get_threshold_for_arduino.php

```php
<?php
header('Content-Type: application/json');
header('Access-Control-Allow-Origin: *');

include_once("dbconnect.php");

try {
    // Get the threshold values for sensor_id = 1 (default DHT11)
    $stmt = $conn->prepare("SELECT threshold_temp, threshold_humidity FROM tbl_threshold_relay WHERE sensor_id = 1");
    $stmt->execute();
    $result = $stmt->get_result();

    if ($row = $result->fetch_assoc()) {
        // Return thresholds in a simple format for Arduino
        echo json_encode([
            'temp_threshold' => floatval($row['threshold_temp']),
            'humidity_threshold' => floatval($row['threshold_humidity']),
            'status' => 'success'
        ]);
    } else {
        // Return default values if no thresholds found
        echo json_encode([
            'temp_threshold' => 26.0,
            'humidity_threshold' => 70.0,
            'status' => 'default'
        ]);
    }

} catch (Exception $e) {
    // Return default values on error
    echo json_encode([
        'temp_threshold' => 26.0,
        'humidity_threshold' => 70.0,
        'status' => 'error'
    ]);
}

$conn->close();
?>
```

login.php

```php
<?php
include_once("dbconnect.php");

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $email = trim($_POST['email'] ?? '');
    $password = trim($_POST['password'] ?? '');

    if (empty($email) || empty($password)) {
        echo json_encode(['status' => 'error', 'message' => 'Email and password required']);
        exit;
    }

    $stmt = $conn->prepare("SELECT tbl_password FROM user_login WHERE tbl_email = ?");
    $stmt->bind_param("s", $email);
    $stmt->execute();
    $stmt->bind_result($hashed_password);
    if ($stmt->fetch() && password_verify($password, $hashed_password)) {
        echo json_encode(['status' => 'success', 'message' => 'Login successful']);
    } else {
        echo json_encode(['status' => 'error', 'message' => 'Invalid email or password']);
    }
    $stmt->close();
} else {
    echo json_encode(['status' => 'error', 'message' => 'Invalid request']);
}
?>
```

register.php

```php
<?php
include_once("dbconnect.php");

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $email = trim($_POST['email'] ?? '');
    $password = trim($_POST['password'] ?? '');

    if (empty($email) || empty($password)) {
        echo json_encode(['status' => 'error', 'message' => 'Email and password required']);
        exit;
    }

    // Check if email already exists
    $stmt = $conn->prepare("SELECT tbl_id FROM user_login WHERE tbl_email = ?");
    $stmt->bind_param("s", $email);
    $stmt->execute();
    $stmt->store_result();
    if ($stmt->num_rows > 0) {
        echo json_encode(['status' => 'error', 'message' => 'Email already registered']);
        exit;
    }
    $stmt->close();

    // Hash the password
    $hashed_password = password_hash($password, PASSWORD_DEFAULT);

    // Insert new user
    $stmt = $conn->prepare("INSERT INTO user_login (tbl_email, tbl_password) VALUES (?, ?)");
    $stmt->bind_param("ss", $email, $hashed_password);
    if ($stmt->execute()) {
        echo json_encode(['status' => 'success', 'message' => 'Registration successful']);
    } else {
        echo json_encode(['status' => 'error', 'message' => 'Registration failed']);
    }
    $stmt->close();
} else {
    echo json_encode(['status' => 'error', 'message' => 'Invalid request']);
}
?>
```

update_threshold.php

```php
<?php
header('Content-Type: application/json');
header('Access-Control-Allow-Origin: *');
header('Access-Control-Allow-Methods: GET, POST, OPTIONS');
header('Access-Control-Allow-Headers: Content-Type');

include_once("dbconnect.php");

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    try {
        $sensor_id = intval($_POST['sensor_id'] ?? 1);
        $threshold_temp = floatval($_POST['threshold_temp'] ?? 26.0);
        $threshold_humidity = floatval($_POST['threshold_humidity'] ?? 70.0);

        // Validate input ranges
        if ($threshold_temp < 0 || $threshold_temp > 100) {
            echo json_encode([
                'status' => 'error',
                'message' => 'Temperature threshold must be between 0°C and 100°C'
            ]);
            exit;
        }

        if ($threshold_humidity < 0 || $threshold_humidity > 100) {
            echo json_encode([
                'status' => 'error',
                'message' => 'Humidity threshold must be between 0% and 100%'
            ]);
            exit;
        }

        // Check if threshold exists for this sensor
        $checkStmt = $conn->prepare("SELECT sensor_id FROM tbl_threshold_relay WHERE sensor_id = ?");
        $checkStmt->bind_param("i", $sensor_id);
        $checkStmt->execute();
        $checkResult = $checkStmt->get_result();

        if ($checkResult->num_rows > 0) {
            // Update existing threshold
            $updateStmt = $conn->prepare("UPDATE tbl_threshold_relay SET threshold_temp = ?, threshold_humidity = ?, timestamp = NOW() WHERE sensor_id = ?");
            $updateStmt->bind_param("ddi", $threshold_temp, $threshold_humidity, $sensor_id);

            if ($updateStmt->execute()) {
                echo json_encode([
                    'status' => 'success',
                    'message' => 'Threshold updated successfully',
                    'sensor_id' => $sensor_id,
                    'threshold_temp' => $threshold_temp,
                    'threshold_humidity' => $threshold_humidity
                ]);
            } else {
                echo json_encode([
                    'status' => 'error',
                    'message' => 'Failed to update threshold'
                ]);
            }
            $updateStmt->close();
        } else {
            // Insert new threshold
            $insertStmt = $conn->prepare("INSERT INTO tbl_threshold_relay (sensor_id, sensor_name, threshold_temp, threshold_humidity) VALUES (?, 'dht11', ?, ?)");
            $insertStmt->bind_param("idd", $sensor_id, $threshold_temp, $threshold_humidity);

            if ($insertStmt->execute()) {
                echo json_encode([
                    'status' => 'success',
                    'message' => 'Threshold created successfully',
                    'sensor_id' => $sensor_id,
                    'threshold_temp' => $threshold_temp,
                    'threshold_humidity' => $threshold_humidity
                ]);
            } else {
                echo json_encode([
                    'status' => 'error',
                    'message' => 'Failed to create threshold'
                ]);
            }
            $insertStmt->close();
        }
        $checkStmt->close();

    } catch (Exception $e) {
        echo json_encode([
            'status' => 'error',
            'message' => 'Database error: ' . $e->getMessage()
        ]);
    }
} else {
    echo json_encode([
        'status' => 'error',
        'message' => 'Invalid request method'
    ]);
}

$conn->close();
?>
```

tbl_dht

	id	temperature	humidity	timestamp	relay_status
编辑 复制 删除	1	22.5	81	2025-05-29 12:49:58	1
编辑 复制 删除	2	22.5	79	2025-05-29 12:50:08	1
编辑 复制 删除	3	22.5	79	2025-05-29 12:50:25	1
编辑 复制 删除	4	22.4	79	2025-05-29 12:50:35	1
编辑 复制 删除	5	22.4	79	2025-05-29 12:50:45	1
编辑 复制 删除	6	22.4	79	2025-05-29 12:50:56	1
编辑 复制 删除	7	22.4	79	2025-05-29 12:51:05	1
编辑 复制 删除	8	28.2	84	2025-05-29 12:51:16	1
编辑 复制 删除	9	29.5	95	2025-05-29 12:51:26	1
编辑 复制 删除	10	29.2	95	2025-05-29 12:51:36	1
编辑 复制 删除	11	27.7	95	2025-05-29 12:51:46	1
编辑 复制 删除	12	26.7	95	2025-05-29 12:51:56	0
编辑 复制 删除	13	25.8	95	2025-05-29 12:52:06	0
编辑 复制 删除	14	25.2	95	2025-05-29 12:52:16	0
编辑 复制 删除	15	24.7	95	2025-05-29 12:52:26	0
编辑 复制 删除	16	24.3	95	2025-05-29 12:52:36	0
编辑 复制 删除	17	24	95	2025-05-29 12:52:46	0
编辑 复制 删除	18	23.8	95	2025-05-29 12:52:56	0
编辑 复制 删除	19	23.5	95	2025-05-29 12:53:06	1
编辑 复制 删除	20	23.3	95	2025-05-29 12:53:16	1
编辑 复制 删除	21	23	95	2025-05-29 12:53:26	1
编辑 复制 删除	22	22.9	95	2025-05-29 12:53:36	1
编辑 复制 删除	23	22.8	95	2025-05-29 12:53:47	1
编辑 复制 删除	24	22.7	95	2025-05-29 12:53:57	1
编辑 复制 删除	25	22.5	95	2025-05-29 12:54:07	1

tbl_threshold_relay

	sensor_id	sensor_name	threshold_temp	threshold_humidity	timestamp
编辑 复制 删除	1	dht11	100	100	2025-05-29 14:55:50

user_login

	tbl_id	tbl_email	tbl_password	timestamp
编辑 复制 删除	3	k@gmail.com	$2y$10$PZdlPvCCGJoJleBeM3uT2.1DV4qPBlTKnS3yzi3HeaE...	2025-05-28 23:20:28
编辑 复制 删除	4	a@gmail.com	$2y$10$fQA/E/l/Uci482LKTuDebO.wx02YYx1u412w.mGAgKy...	2025-05-28 23:27:37

3.2 Relay

```
void loop() {
  // Fetch thresholds every 2 seconds
  if (millis() - fetchThresholdPrevMillis > 2000 || fetchThresholdPrevMillis == 0) {
    fetchThresholdPrevMillis = millis();
    fetchThresholds();
  }

  // Send sensor data every 10 seconds
  if (millis() - sendDataPrevMillis > 10000 || sendDataPrevMillis == 0) {
    sendDataPrevMillis = millis();

    // Read DHT
    hum  = dht.readHumidity();
    temp = dht.readTemperature();
    if (isnan(hum) || isnan(temp)) {
      Serial.println("Failed to read from DHT sensor!");
      return;
    }

    // Apply relay logic with dynamic thresholds - OR condition
    bool relayOn = (temp > TEMP_THRESHOLD || hum > HUM_THRESHOLD);
    digitalWrite(RELAY_PIN, relayOn ? HIGH : LOW);

    Serial.printf("Temp=%.1f°C (>%.1f)  Hum=%.1f%% (>%.1f)  Relay=%s\n",
                  temp, TEMP_THRESHOLD, hum, HUM_THRESHOLD, relayOn ? "ON" : "OFF");

    // Refresh OLED with dynamic threshold display
    updateDisplay(temp, hum, relayOn);
```

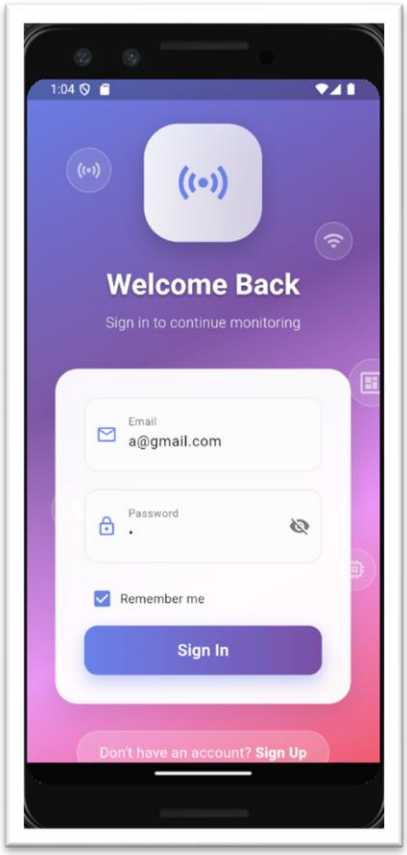
```

    // Send to server
    if (WiFi.status() == WL_CONNECTED) {
      WiFiClient client;
      HTTPClient http;
      String url = serverName + "dht11.php?id=101"
                   + "&temp="  + String(temp, 1)
                   + "&hum="   + String(hum, 1)
                   + "&relay=" + String(relayOn ? 1 : 0);
      http.begin(client, url);
      int code = http.GET();
      if (code > 0) {
        Serial.printf("HTTP Response code: %d\n", code);
        Serial.println(http.getString());
      } else {
        Serial.printf("HTTP Error code: %d\n", code);
      }
      http.end();
    }
  }
}
```

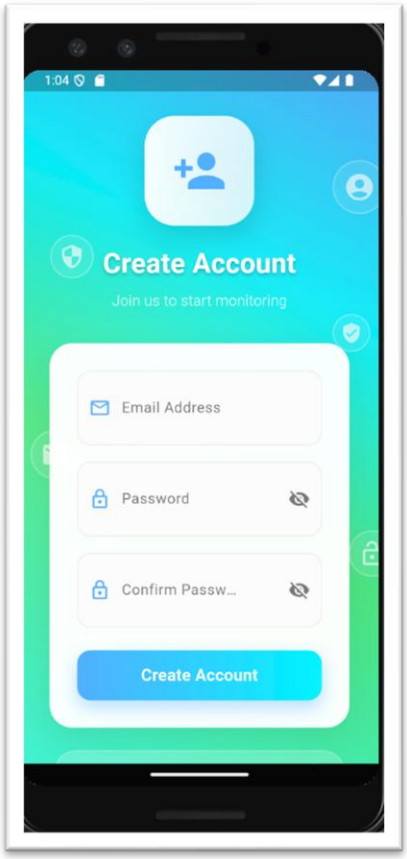
3.3 App

Splashscreen

Login Screen



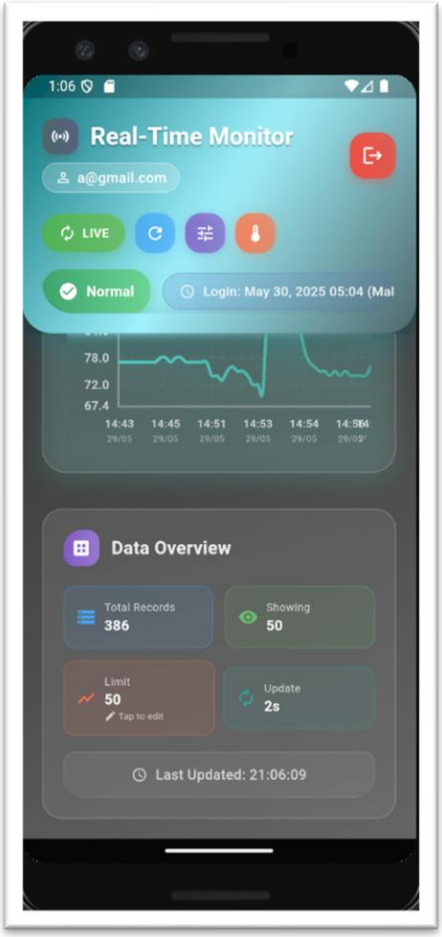
Register Screen

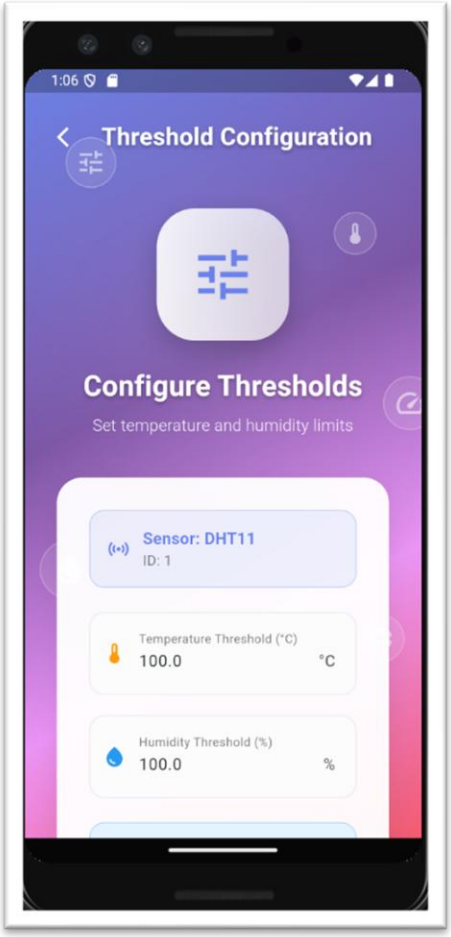


Dashboard

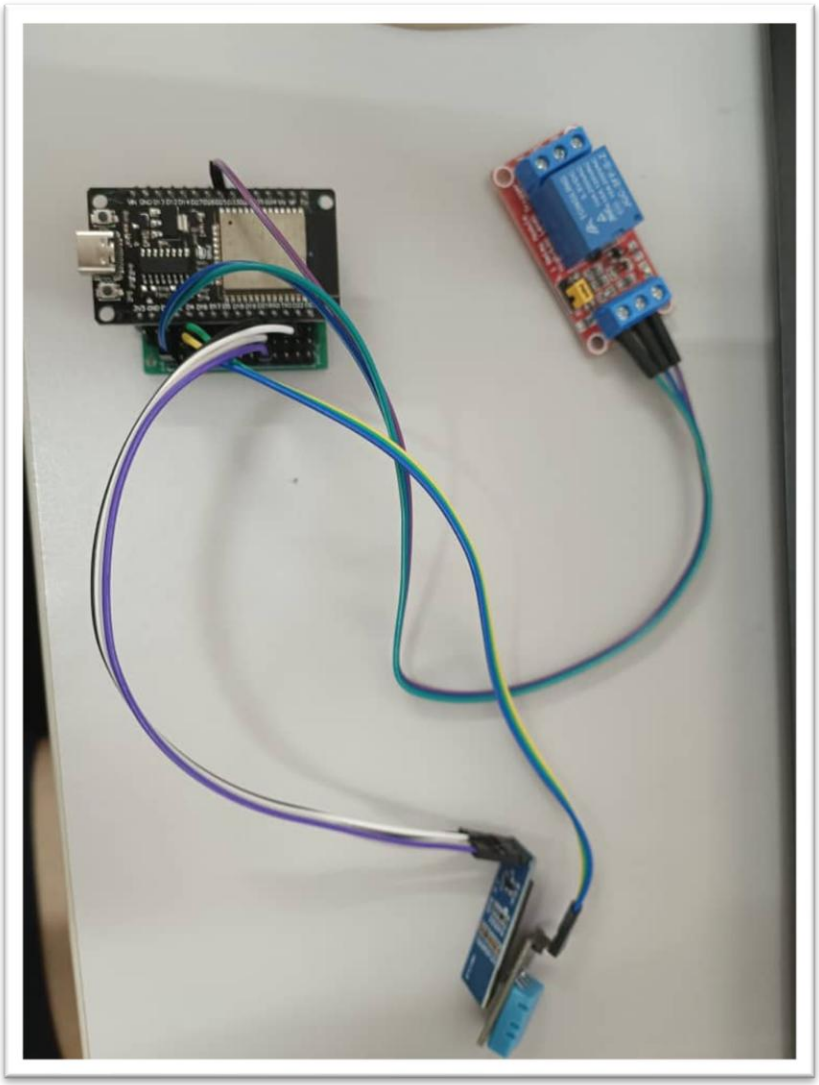

1:04
Real-Time Monitor
a@gmail.com
LIVE
Normal
Login: May 30, 2025 05:04 (Mal
°C
Temperature
23.6 °C
%
Humidity
76.0 %
Quick Analytics
Avg Temp
23.2°C
Avg Humidity
78.2%
Temp Range
22.2 - 25.2°C
Humidity Range
71.0 - 95.0%


1:05
Real-Time Monitor
a@gmail.com
LIVE
Normal
Login: May 30, 2025 05:04 (Mal
Temperature Trend
50 data points
LIVE
23.6°C
25.6
25.0
24.0
23.0
22.0
21.8
14:43
14:45
14:51
14:53
14:54
Humidity Trend
50 data points
LIVE
76.0%


1:06
Real-Time Monitor
a@gmail.com
LIVE
Normal
Login: May 30, 2025 05:04 (Mal
78.0
72.0
67.4
14:43
14:45
14:51
14:53
14:54
Data Overview
Total Records
386
Showing
50
Limit
50
Tap to edit
Update
2s
Last Updated: 21:06:09


1:06
Threshold Configuration
Configure Thresholds
Set temperature and humidity limits
Sensor: DHT11
ID: 1
Temperature Threshold (°C)
100.0
°C
Humidity Threshold (%)
100.0
%

3.4 ESP 32 DevKit



4.0 Challenges and improvements

The development of the ESP32 Temperature & Humidity Monitoring System was successful, yet it presented several challenges and highlighted areas for future improvements. One major issue encountered was the sensitivity of the DHT11 humidity sensor, which often reported a constant 95% humidity reading, only changing under extreme conditions. While this indicated that the sensor was functioning, it unfortunately resulted in biased humidity data and skewed visualizations on the dashboard, limiting the system's ability to effectively detect subtle changes in the environment.

Another critical area requiring careful consideration was the reliability of hardware connections. Ensuring accurate and secure wiring of GPIO pins, 3.3V power supply, and ground connections was essential to avoid reverse currents or short circuits that could potentially damage the ESP32 and its peripherals. This added a layer of complexity and required meticulous attention during assembly.

Furthermore, integrating the complete technology stack, including the PHP backend hosted on JomHosting, the MySQL database, and the Flutter-based mobile application, presented a significant learning curve. Familiarizing oneself again with data handling procedures across multiple platforms inevitably led to time constraints, especially when debugging issues across interconnected components.

Moving forward, several enhancements could significantly boost the system's performance and improve the user experience. Firstly, addressing data accuracy by upgrading from the DHT11 to a more precise and responsive sensor, such as the Bosch BME280 or Sensirion SHT31, would be beneficial. These sensors not only offer improved temperature and humidity accuracy but, in the case of the BME280, also provide barometric pressure measurements. While this upgrade would require minor hardware adjustments and firmware modifications, it would deliver more reliable and detailed environmental data.

Secondly, introducing Over-the-Air (OTA) firmware updates would substantially enhance the system's maintainability. OTA updates allow firmware modifications to be deployed remotely over Wi-Fi, eliminating the necessity for physical USB connections and simplifying the rollout of bug fixes or new features, particularly beneficial for managing multiple deployed devices. Implementing this could be efficiently achieved using existing libraries such as ArduinoOTA or leveraging cloud-based IoT services.

Lastly, improving the relay module for practical, real-world applications and easier user configuration would add significant value. Transitioning from a basic relay module to an industrial-grade or smart relay system could offer improved durability, higher current handling capabilities, and built-in safety measures. Additionally, developing a more user-friendly interface perhaps within the mobile app or a dedicated web dashboard for defining more sophisticated relay activation rules would enhance usability. This could include features such as time-based scheduling, hysteresis control to prevent rapid relay switching or integration with external triggers, making the system versatile enough for various control scenarios, ranging from simple alarms to comprehensive environmental management solutions.