VIETNAM NATIONAL UNIVERSITY
HO CHI MINH UNIVERSITY OF TECHNOLOGY
COMPUTER SCIENCE & ENGINEERING FACULTY

EXPERIMENT REPORT

CO3038
INTERNET OF THINGS
(L01)

Teacher: Cao Tiến Đạt

Student: Nguyễn Thanh Hiền ID: 2111203

Ho Chi Minh City, March 2025

TABLE OF CONTENT

LIST OF IMAGES

LIST OF TABLE

1. OVERVIEW

Student name: Nguyen Thanh Hien
Student ID: 2111203
Class: L01
Source Code: https://github.com/amyranotamirror/CO3038-iot/tree/main/lab-1

1.1. Introduction

Temperature & Humidity sensors solution was suitable for multiple applications. In this lab, you are supposed to get an interactive dashboard that can observe real-time data.

1.2. Requirements

Design and implement an IoT dashboard that displays temperature and humidity in real-time.

1.3. Definition of Done

- Design and implementation an IoT dashboard.
- Measure temperature and humidity in real-time.
- Define test plan, perform, and report test result.
- List technical specifications of your temperature and humidity solution.
- Write instructions and notable points when deploying the solution.

2. SYSTEM DESIGN

2.1. System Architecture

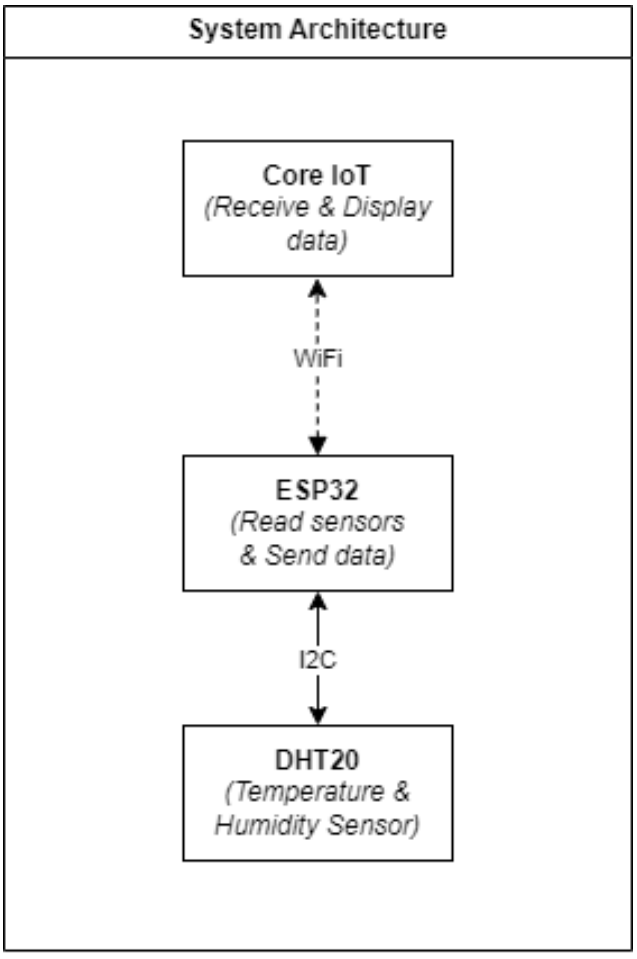

Figure 1. System Architecture

This IoT system follows a three-layer architecture:

1. Perception Layer (Data collection)
    - DHT20 sensors periodically collects temperature & humidity data.
    - ESP32 reads data and processes it before transmitting.
2. Network Layer (Data transmission)
    - Data is transmitted via WiFi using MQTT protocol.
3. Application Layer (Dashboard)
    - The Core IoT platform receives and displays real-time data on monitoring dashboard.

2.2. Technical Risks

- Sensors malfunction or fails to operate correctly.
- ESP32 malfunctions or fails to operate correctly.
- Wiring between ESP32 and DHT20 is unstable or disconnected.
- WiFi connection is unstable or disconnected.
- CoreIoT connection is unstable or disconnected.

2.3. Hardware Components

- ESP32 UNO R3
- DHT20 temperature and humidity sensors.

2.4. Dashboard Design

With the requirement to monitor temperature and humidity, the student decided to monito:

- Latest updated data: View immediate result while performing tests.
- Historical data (displayed as line graph): View data changes over time.

3. TECHNICAL SPECIFICATION

3.1. Hardware Specification

- Temperature & Humidity Sensor: **DHT20** [1]
    - Input voltagee: 3.3VDC
    - Humidity range: 0 ~ 100% RH
    - Humidity accuracy: ± 3% RH (25 ℃)
    - Temperature range: -40 ~ + 80 ℃
    - Temperature accuracy: ± 0,5 ℃
- Microcontroller board: **Kit Wifi BLE SoC ESP32 WeMos D1 R32** [2]
    - Power supply:
        - 5VDC via USB port
        - 5~12V DC via round DC power jack or Vin pin.
    - Central Processing Unit: Wireless Module – ESP-WROOM-32, based on Espressif ESP32 dual-core Tensilica LX6 processor with 802.11 b/g/n WiFi and Bluetooth 4.2 LE.
    - Pin Configuration & Compatibility: Full ESP32 pinout, designed with GPIO pin headers and dimensions similar to Arduino Uno.
    - Integrated Components: Built-in CH340 UART communication and programming circuit.
- **Arduino UNO extended shield** [3]
    - Port: 4 Analog, 4 I2C, 4 Digital

[1] Ohstem. 1. Cảm biến nhiệt độ độ ẩm DHT20. Access via: https://docs.ohstem.vn/en/latest/module/cam-bien/dht20.html (Access date: March 20th, 2025).

[2] HSHOP. Kit Wifi BLE SoC ESP32 WeMos D1 R32 (Arduino Compatible). Access via: https://hshop.vn/kit-arduino-wifi-ble-soc-esp32-wemos-d1-r32 (Access date: March 20th, 2025).

[3] Ohstem. Mạch mở rộng cho Arduino UNO. https://ohstem.vn/product/mach-mo-rong-cho-arduino/ (Access date: March 20th, 2025).

  - 11 servo channel
  - Switch between 3.3V to 5V

3.2. Data Collection

- Measurement frequency: Every 1 second (both temperature and humidity)
- Temperature serial print
  - Frequency: Every 1,5 second
  - Format: `Temperature: x °C`, with `x` is the temperature data.
- Humidity serial print:
  - Frequency: Every 1 second
  - Format: `Humidity: x %`, with `x` is the humidity data.

3.3. Data Transmission

- Protocol: MQTT
- Payload format: JSON
- Transmission interval: Every 10 second (both temperature and humidity)
- Connection monitoring:
  - Check WiFi and CoreIoT connection every 4 seconds
  - Automatically reconnects if disconnection is detected.

3.4. IoT Platform

- CoreIoT platform: ThingsBoard-based, Cloud-hosted
- Topic: v1/devices/me/telemetry
- Visualization: Real-time dashboards with card widgets

4. SYSTEM IMPLEMENTATION

4.1. Development Environment

4.1.1. IDE

- VSCode (PlatformIO)

4.1.2. Library

- `Arduino.h`: Provides core functions for Arduino programming.
- `Wire.h`: Provides communication over I2C protocol, used by DHT20 sensor.
- `WiFi.h`: Provides support for WiFi connectivity on ESP32 board.
- `DHT20.h`: Provides pecific library to interface with the DHT20 temperature and humidity sensor.
- `Arduino_MQTT_Client.h`: Provides an MQTT client for Arduino-based board.
- `ThingsBoard.h`: Provides connectivity from ESP32 to ThingsBoard

4.2. Dashboard Implementation

4.2.1. Basic Dashboard

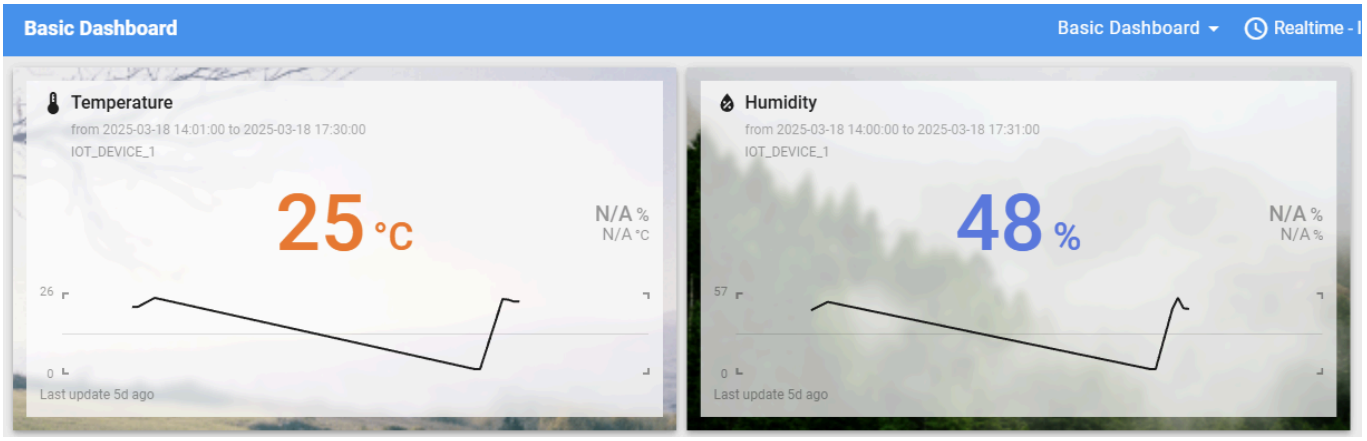

Figure 2. Dashboard Design

4.2.2. Device Connection

Create a public device called "IOT-DEVICE-1", then get the "Access token" to connect the physical device with the dashboard telemetry channel, which captures the temperature and humidity data.

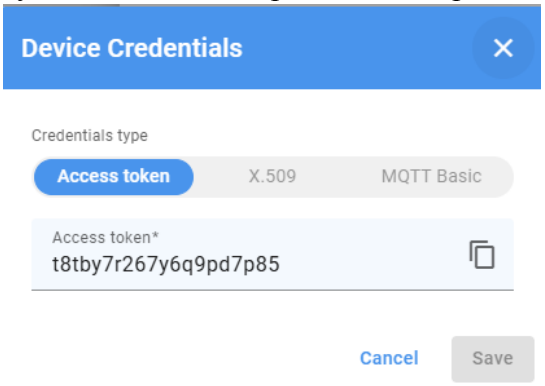

Figure 3. Access Token

```
constexpr char TOKEN[] = "t8tby7r267y6q9pd7p85";
constexpr char THINGSBOARD_SERVER[] = "app.coreiot.io";
constexpr uint16_t THINGSBOARD_PORT = 1883U;
```

4.3. Task Scheduling

The student defined 5 tasks running independently to implement the solution:

- `TaskTemperature`: Print the temperature to serial monitor every 1.5 second.
- `TaskHumidity`: Print the humidity to serial monitor every 1 second.
- `TaskReadDHT20`: Read and store the temperature and humidity data every 1 second.
- `TaskCheckConnection`: Check connection and reconnect if necessary every 4 seconds.
- `TaskSendData`: Send temperature and humidity data to the dashboard every 10 seconds.

5. SYSTEM DEPLOYMENT

5.1. Hardware Setup

5.1.1. Installing the Sensor (DHT20 or Similar)

- Identify an appropriate location for installation based on environmental requirements:

  - **Indoor Deployments**: Avoid placing sensors near air vents, windows, or direct heat sources to prevent distorted readings.
  - **Outdoor Deployments:** Protect the sensor with a weatherproof enclosure to shield it from rain, dust, and extreme temperatures.
  - **Industrial Deployments:** Ensure placement in areas free from excessive vibration, contaminants, or electromagnetic interference.
- Securely mount the sensor using appropriate brackets or adhesive to prevent movement.

5.1.2. Installing Power Supply

- **For Stable Indoor Use:**
  - Connect to a 5V USB adapter with surge protection.
- **For Remote/Outdoor Use:**
  - Utilize a 5–12V DC adapter with regulated voltage to ensure continuous operation.
  - Deploy a battery pack (e.g., LiPo with voltage stabilization) in areas where power sources are unreliable.
  - Ensure power stability by testing voltage levels at the deployment site.

5.2. Software Deployment

5.2.1. Connecting to CoreIoT Platform

- Confirm that the CoreIoT server (ThingsBoard-based) is active and accessible.
- Register the sensor device on CoreIoT and generate an access token for authentication.

5.2.2. Flashing the Microcontroller (ESP32 or Equivalent)

- Flash the microcontroller with firmware that includes:
  - WiFi credentials (SSID and password) specific to the deployment environment.
  - Access token linked to the CoreIoT platform.
  - Configurations for MQTT data transmission.
- Secure the microcontroller inside a protective casing to prevent damage or environmental exposure.
- Power on the device and verify automatic connection to WiFi and data streaming.

5.3. Dashboard Setup & Monitoring

- Access the CoreIoT dashboard via web or mobile device.
- Validate real-time sensor readings (temperature, humidity, etc.) and confirm accurate data logging.
- Adjust the data refresh rate to balance real-time monitoring and network bandwidth usage.

5.4. Other Considerations

5.4.1. Environmental Considerations

- Install the sensor in a location that reflects true environmental conditions for accurate monitoring.
- Avoid placements near:
  - Vents or air conditioning units, as airflow can cause temperature fluctuations.
  - Direct sunlight, unless the goal is to measure high heat exposure.
  - Enclosed spaces, which may restrict airflow and affect readings.

5.4.2. Ensuring Network Stability

- Test WiFi signal strength at the deployment site to confirm stable connectivity.
- For large areas or weak signal zones, consider using WiFi extenders or mesh networks.

5.4.3. Power Backup & Redundancy

- Implement uninterruptible power supplies (UPS) for critical deployments.
- Periodically inspect power sources to prevent unexpected failures and ensure long-term functionality.

5.4.4. Securing the Hardware

- Protect the sensor and microcontroller against tampering and theft by using:
  - Lockable enclosures in public areas.
  - Secure mounting mechanisms to prevent unauthorized removal.
- Select an enclosure material that balances protection while maintaining accurate sensor readings.

5.4.5. Post-Deployment Validation & Maintenance

- Initial Testing: Monitor the first 5–10 minutes of sensor data to detect any connectivity or calibration issues.
- Scheduled Maintenance:
  - Perform monthly inspections to clean sensors, check connections, and update firmware.
  - Verify continuous data transmission and connectivity to the CoreIoT platform.
- Firmware & Software Updates:
  - Regularly check for firmware updates to enhance performance and security.

5.4.6. Scaling for Multiple Deployments

- Assign unique device identifiers in CoreIoT to efficiently track and manage multiple units.
- Use a visual dashboard (e.g., map widgets) to monitor sensor locations across different sites.

6. TESTING & VALIDATION

6.1. Approach

Monitor the temperature value displayed on the dashboard. Then, attempt to change the temperature reading by using breath or finger. Observe any changes in the displayed value.

6.3. Test Cases & Result

ID	Test case	Test step	Expected result	Evidence	Result
TC1	Board Disconnected	1. Power off the ESP32 board. 2. Serial monitor for sensor reading.	- Compile and flashing result failed.	Compile result: Image Serial monitor: Image	Pass
TC2	Sensor Disconnected	1. Power on the ESP32 and connect to WiFi. 2. Disconnect the DHT20 sensor. 3. Serial monitor for sensor reading.	- Compile and flashing result succeed. - Serial monitor displays error code “236” with i2cRead	Compile result: Image Serial monitor: Image	Pass
TC3	WiFi Disconnected	1. Power on the ESP32 and connect the sensor. 2. Turn off WiFi. 3. Monitor the ESP32 logs for retry attempts. 4. Reconnect WiFi and observe recovery.	- Compile and flashing result succeed. - The ESP32 detects WiFi disconnection and attempts to reconnect. - Data transmission resumes after WiFi is restored.	Compile result: Image Serial monitor: Image	Pass
TC4	Normal Operation	1. Power on the ESP32 with the sensor connected and WiFi enabled. 2. Monitor the CoreIoT dashboard for real-time data updates.	- Compile and flashing result succeed. - Temperature & humidity values update every 1 second on the serial monitor. - Temperature & humidity values update every 10 seconds on CoreIoT dashboard. - No errors appear in the serial monitor.	Compile result: Image Serial monitor: Image Dashboard: Image	Pass

Table 1. Test cases & result

REFERENCE

1. Ohstem. 1. Cảm biến nhiệt độ độ ẩm DHT20. Access via: https://docs.ohstem.vn/en/latest/module/cam-bien/dht20.html (Access date: March 20th, 2025).
2. HSHOP. Kit Wifi BLE SoC ESP32 WeMos D1 R32 (Arduino Compatible). Access via: https://hshop.vn/kit-arduino-wifi-ble-soc-esp32-wemos-d1-r32 (Access date: March 20th, 2025).
3. Ohstem. Mạch mở rộng cho Arduino UNO. https://ohstem.vn/product/mach-mo-rong-cho-arduino/ (Access date: March 20th, 2025).