

BÁO CÁO THỰC HÀNH

CÔNG NGHỆ INTERNET OF THINGS HIỆN ĐẠI

Lab 4

Xây dựng giao diện ứng dụng để tương tác với mô hình IoT

GVHD: Phan Trung Phát

Lớp: NT532.021.MMCL.2

Họ và tên	MSSV
Tổng Võ Anh Thuận	21522652
Trịnh Vinh Đại	21521915
Lê Huỳnh Quan Vũ	21522797

ĐÁNH GIÁ KHÁC (*):

Nội dung	Kết quả
Tổng thời gian thực hiện bài thực hành trung bình (1)	
Link Video thực hiện (2) (nếu có)	Code n Video
Ý kiến (3) (nếu có) + Khó khăn + Đề xuất ...	
Điểm tự đánh giá (4)	10/10

(*): phần (1) và (4) bắt buộc thực hiện.

Phần bên dưới là báo cáo chi tiết
của nhóm/cá nhân thực hiện.



LƯU HÀNH NỘI BỘ



Lựa chọn 1 nền tảng IoT platform mã nguồn mở và thực hiện xây dựng ứng dụng IoT cơ bản đầy đủ các thành phần và chức năng, bao gồm:

The IoT platform we will be using for this lab is **ThingsBoard Cloud**. It is free and open source (it is also recommended by the instructor). Documentation is available here: <https://thingsboard.io/docs/paas/>

1. Device: sinh viên thực hiện sử dụng 2 loại thiết bị cảm biến DHT22/DHT11 và MQ-135 để lấy 3 loại dữ liệu cảm biến là nhiệt độ, độ ẩm và ánh sáng. Trên Wemos D1 sử dụng cảm biến BH-1750, đồng thời lập trình để gửi các giá trị cảm biến đến hệ thống thông qua giao thức HTTP. Trên Raspberry Pi sử dụng DHT22/DHT11, đồng thời lập trình để gửi các giá trị cảm biến đến hệ thống thông qua giao thức MQTT. Các dữ liệu này cũng được lưu trữ tại hệ thống IoT platform.

a) Wemos D1

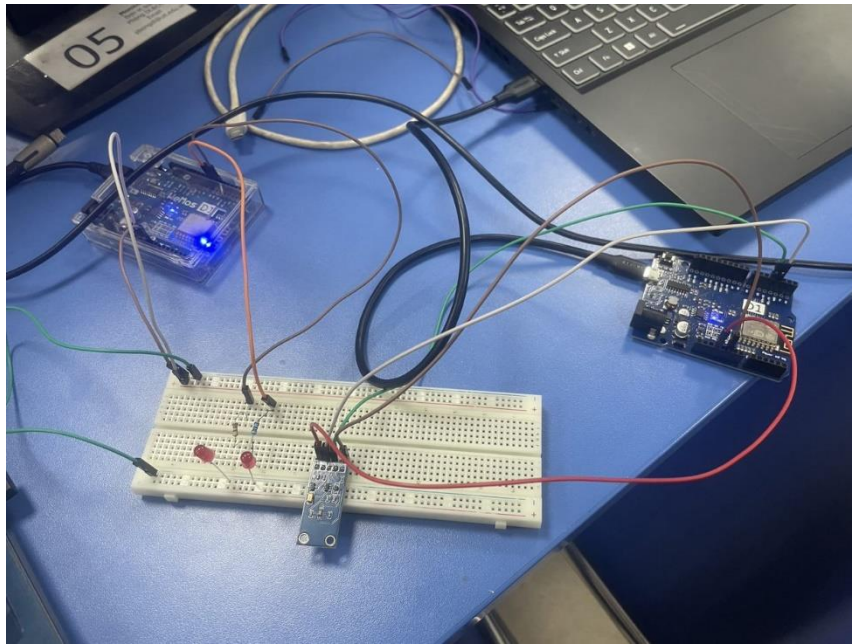
Set up:

Components in used:

- 2 Wemos D1 R2.
- 1 BH1750 sensor.
- 2 LEDs.
- 2 resistors.
- 6 jumper wires

Wire connection:

- 2 LEDs:
 - o Anode: Use digital pins D3 and D4 on Wemos 2. Those pins control LED output. Each LED connects with one resistor.
 - o Cathode: Connect to GND zone on the breadboard. This zone grounds the LED.
- BH1750 sensor:
 - o Use digital pins SCL/D1 and SDA/D2 on Wemos 1.
 - o Anode and Cathode: Connect to GND zone and VCC zone on the breadboard.



Code explanation:

Wemos 1: connect with BH1750 sensor.

```
6   const char* ssid = "Daivjppro";
7   const char* password = "123456780";
8   const char* mqtt_server = "mqtt.thingsboard.cloud";
9   const char* topic = "v1/devices/me/telemetry";
10  const String username = "1n2DNxT7sDDA0ycBziSe";
11
```

This code declares Wifi name, password, MQTT server address, topic and username.

```
59 void setup() {
60   Serial.begin(115200);
61   Wire.begin(D2, D1); // Khởi động giao tiếp I2C với chân SDA là D2 và chân SCL là D1
62   if (!lightMeter.configure(BH1750::CONTINUOUS_HIGH_RES_MODE)) {
63     Serial.println("BH1750 configuration failed!");
64   } else {
65     Serial.println("BH1750 configured successfully!");
66   }
67   setup_wifi(); // Kết nối Wifi
68   client.setServer(mqtt_server, 1883); // Thiết lập máy chủ MQTT
69 }
70
71 void loop() {
72   if (!client.connected()) {
73     reconnect(); // Kết nối lại nếu mất kết nối MQTT
74   }
75   client.loop();
76
77   // Đọc giá trị từ cảm biến BH1750
78   float lux = lightMeter.readLightLevel();
79
80   // Gửi giá trị lên MQTT
81   char msg[50];
82   snprintf(msg, 50, "{\"lux\":%.2f}", lux);
83   Serial.print("Publish message: ");
84   Serial.println(msg);
85   client.publish(topic, msg);
86
87   delay(2000); // Chờ 2 giây trước khi gửi lại
88 }
```



This code snippet is to start I2C communication, set up the MQTT Server, try to connect to the MQTT Server again if the connection fails, read the BH1750 sensor value, and send data to the Server.

Wemos 2: connect with 2 LEDs.

```
5   #define WIFI_AP "Daivjppro"
6   #define WIFI_PASSWORD "123456780"
7
8   #define TOKEN "1n2DNxT7sDDA0ycBziSe"
9
10  #define GPIO0 0
11  #define GPIO2 2
12
13  #define GPIO0_PIN 3
14  #define GPIO2_PIN 5
15
16  char thingsboardServer[] = "mqtt.thingsboard.cloud";
17  const char* topic = "v1/devices/me/rpc/request/+"; //
18  WiFiClient wifiClient;
```

This code declares Wifi name, password, MQTT server address, topic, define GPIO pins and define pin codes for those 2 GPIO pins.

```
47 void on_message(char* topic, byte* payload, unsigned int length) {
48
49     Serial.println("On message");
50
51     char json[length + 1];
52     strncpy (json, (char*)payload, length);
53     json[length] = '\0';
54
55     Serial.print("Topic: ");
56     Serial.println(topic);
57     Serial.print("Message: ");
58     Serial.println(json);
59
60     // Decode JSON request
61     DynamicJsonDocument jsonBuffer(200); // Sử dụng DynamicJsonDocument
62     DeserializationError error = deserializeJson(jsonBuffer, json);
63
64     if (error)
65     {
66         Serial.println("deserializeJson() failed");
67         return;
68     }
69
70     // Check request method
71     String methodName = jsonBuffer["method"].as<String>();
72
73     if (methodName.equals("getGpioStatus")) {
74         // Reply with GPIO status
75         String responseTopic = String(topic);
76         responseTopic.replace("request", "response");
77         client.publish(responseTopic.c_str(), get_gpio_status().c_str());
78     } else if (methodName.equals("setGpioStatus")) {
79         // Update GPIO status and reply
80         set_gpio_status(jsonBuffer["params"][0].as<int>(), jsonBuffer["params"][1].as<boolean>());
81         String responseTopic = String(topic);
82         responseTopic.replace("request", "response");
83         client.publish(responseTopic.c_str(), get_gpio_status().c_str());
84         client.publish("v1/devices/me/attributes", get_gpio_status().c_str());
85     }
86 }
87
```

This code will do the following:



- Create a function to receive messages from MQTT Server.
- Handle the payload: Create a char array of length equal to payload + 1 to contain the JSON message, then copy the payload content into the JSON array and add the string terminator character “\0” to make it a valid string.
- Print to Serial port the information received.
- JSON decoding: Use DynamicJsonDocument to create a 200-byte dynamic JSON document, then decode the JSON string and check for decoding errors.
- Handle request method:
- Reply to request:

```
88  String get_gpio_status() {
89      // Prepare gpio's JSON payload string
90      DynamicJsonDocument jsonBuffer(200); // Sử dụng DynamicJsonDocument
91      JsonObject data = jsonBuffer.to<JsonObject>();
92      data[String(GPIO0_PIN)] = gpioState[0] ? true : false;
93      data[String(GPIO2_PIN)] = gpioState[1] ? true : false;
94      String strPayload;
95      serializeJson(data, strPayload);
96      Serial.print("Get gpio status: ");
97      Serial.println(strPayload);
98      return strPayload;
99  }
100
101  void set_gpio_status(int pin, boolean enabled) {
102      if (pin == GPIO0_PIN) {
103          // Output GPIOs state
104          digitalWrite(GPIO0, enabled ? HIGH : LOW);
105          // Update GPIOs state
106          gpioState[0] = enabled;
107      } else if (pin == GPIO2_PIN) {
108          // Output GPIOs state
109          digitalWrite(GPIO2, enabled ? HIGH : LOW);
110          // Update GPIOs state
111          gpioState[1] = enabled;
112      }
113  }
114
115  void InitWiFi() {
116      Serial.println("Connecting to AP ...");
117      // attempt to connect to WiFi network
118
119      WiFi.begin(WIFI_AP, WIFI_PASSWORD);
120      while (WiFi.status() != WL_CONNECTED) {
121          delay(500);
122          Serial.print(".");
123      }
124      Serial.println("Connected to AP");
125  }
```

- Creates and returns a JSON string containing the current state of the GPIO pins.
- Set the state for a specific GPIO pin and update this state in the "gpioState" array.
- Connect the device to the WiFi network.

b) Jetson Nano

Reason for the alternative choice:

1. Both Jetson Nano and Raspberry Pi are both single-board computers (SBCs), so the purpose of use remains unchanged.

2. They run the same OS distro, which is Debian, so the development environment should be the same.
3. Jetson Nano has a better performance since it has 4 GB 64-bit LPDDR4 with a strong CPU.
4. We own Jetson Nano locally but not Raspberry Pi so we are flexible to complete the LAB on time.

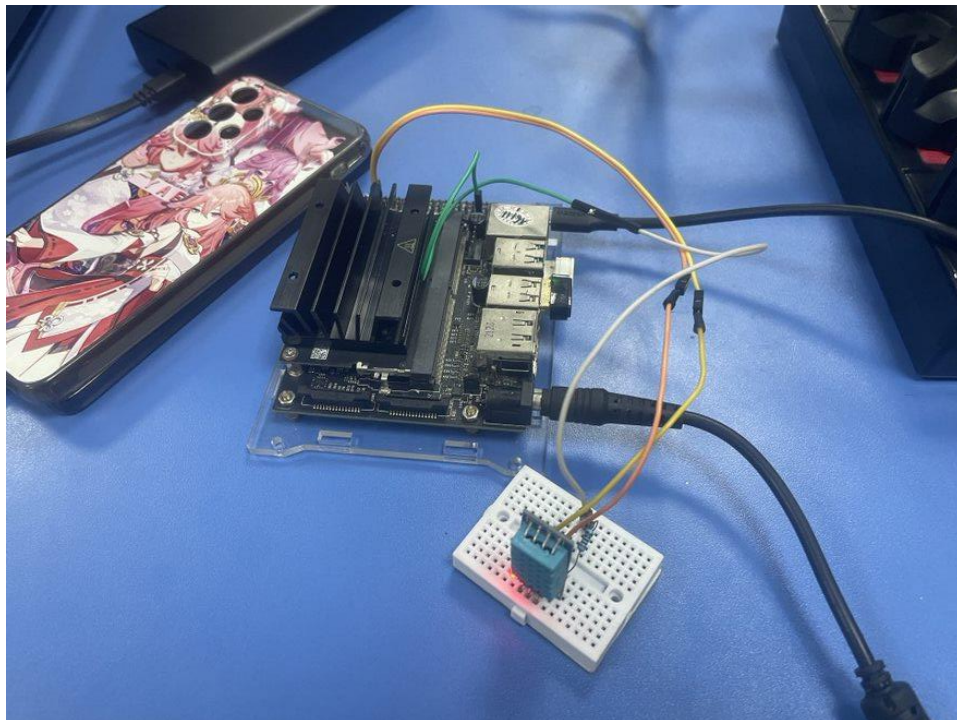
Set up:

Components in use:

- 1 Jetson Nano B1.
- 1 DHT11 sensor.
- 1 resistor.
- 6 jumper wires

Wire connection:

- DHT11 sensor:
 - GND: connect to GND pin on Jetson Nano. This helps ground sensor.
 - Data: connect to pin 33, which is GPIO13 on Jetson Nano. This help transfer data to Jetson Nano.
 - VCC: connect to 5V pin through a resistor on Jetson Nano. This helps power sensor.



Code explanation:

```
import C_CHT
import time
from tb_device_mqtt import TBDeviceMqttClient, TBPublishInfo

while True:
```



```
data = C_DHT.readSensorDHT11(0);
# data = {temperature, NULL, humidity, NULL}
telemetry = {"temperature": data[0], "humidity": data[2]}
client = TBDeviceMqttClient("THINGBOARD-SERVER", "THINGBOARD-TOKEN")
# Connect to ThingsBoard
client.connect()
# Sending telemetry without checking the delivery status
client.send_telemetry(telemetry)
# Sending telemetry and checking the delivery status (QoS = 1 by default)
result = client.send_telemetry(telemetry)
# get is a blocking call that awaits delivery status
success = result.get() = TBPublishInfo.TB_ERR_SUCCESS
# Disconnect from ThingsBoard
client.disconnect()
time.sleep(1)
```

Library in use:

1. C_CHT: This is a C-to-Python library, which is built specifically for Jetson Nano to read DHT11/22 sensor.
2. TBDeviceMqttClient, TBPublishInfo: Those are Thingboards Python libraries that are used to connect and publish data to the Thingboards Server through MQTT.
3. Time: Normal Python library, to delay loop function.

Main code: This code acts as a cron job, which executes every 1 second.

- First, Jetson Nano read data from DHT11 in first option (0). There are 2 pin (33, 18) on Jetson Nano that can read DHT sensor, so that option (0) and option (1) respectively.
- Second, extract data and assign it to telemetry for sending to the server.
- Third, connect Jetson Nano to ThingBoards Server by providing the server address and authentication token.
- Fourth, send telemetry to the server and get response from the server.
- Finally, release the connection and repeat the process.

2. Dashboard: hiển thị tình trạng thiết bị đang kết nối đến hệ thống (Wemos D1, Raspberry Pi,...).

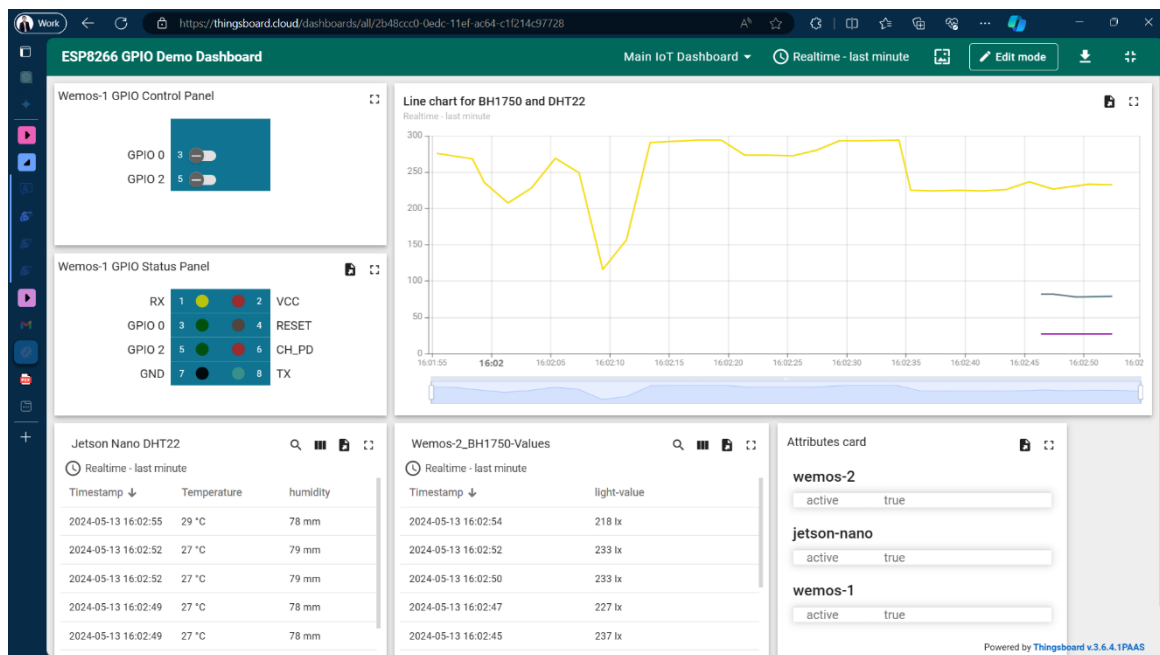
Lưu ý: số lượng thiết bị kết nối đến hệ thống phải từ 2 trở lên (2 Wemos D1,...).

As you can see in the dashboard below, we have “Attribute card” (bottom right) which shows the connection status of 3 currently active IoT devices. Besides, we also have:

- GPIO Control and Status Panel to control the LEDs of the ESP8266 device.
- A line chart to visualize data from the 2 sensors BH1750 and DHT11



- Two time-series tables for the 2 sensors as well (Humidity is incorrectly set as mm for unit due to a mistake).



All the explanations after this section will use the above dashboard image as a reference.

3. Màn hình chính: dùng để thể hiện các tương tác với hệ thống, bao gồm: thể hiện các giá trị cảm biến ở thời điểm hiện tại (tên giá trị, giá trị của cảm biến), chức năng điều khiển 2 đèn LED trên Wemos D1 thông qua giao diện ứng dụng. Chức năng này phải thể hiện thời gian thực.

For LED controlling, both the dashboard configuration and the Wemos code is referenced here: <https://thingsboard.io/docs/samples/esp8266/gpio/>

There are two widgets used for LED controlling in the dashboard:

- ESP8266 GPIO Control Panel (row 0, col 0):
 - This widget allows you to control individual GPIO pins (3 and 5) on the ESP8266 device.
 - It uses an RPC (Remote Procedure Call) type for communication.
 - Clicking a switch on the widget sends a request to the device to change the pin state (on/off).
- ESP8266 GPIO Status Panel (row 0, col 5):
 - This widget displays the current status (on/off) of various GPIO pins on the ESP8266 device.
 - It uses a "latest" type, indicating it shows the most recent data received.
 - Each pin has a label (e.g., GPIO 0, VCC) and a configurable color for better visualization.

4. Biểu đồ: dùng để thể hiện các giá trị cảm biến thu thập được từ phần thiết bị một cách trực quan bằng biểu đồ, bảng biểu. Cụ thể như sau: các giá trị cảm biến bao gồm 3 loại (nhiệt độ, độ ẩm, ánh sáng). Các biểu đồ này thể hiện dưới dạng thời gian thực.



In order to draw the line chart like in the dashboard image. We simply create a line chart widget with the datasource being “lab04-nhom01” which includes all 3 IoT devices (2 Wemos, 1 Jetson Nano). Then we set the key, label, color, and unit for the sensor values for lux, humidity, and temperature respectively.

The screenshot shows the configuration for a line chart widget. At the top, the 'Datasource' is set to 'lab04-nhom01'. Below, the 'Series' section contains a table with three rows for 'lux', 'humidity', and 'temperature'. Each row has columns for Key, Label, Y axis, Color, Units, and Decimals. The 'lux' row has a yellow color and 'lx' units. The 'humidity' row has a blue color and '%' units. The 'temperature' row has a red color and '°C' units. There are also icons for settings, deletion, and expansion for each series.

Key	Label	Y axis	Color	Units	Decimals
lux	lux	default	Yellow	lx	0
humidity	humidity	default	Blue	%	Set
temperature	temperature	default	Red	°C	Set

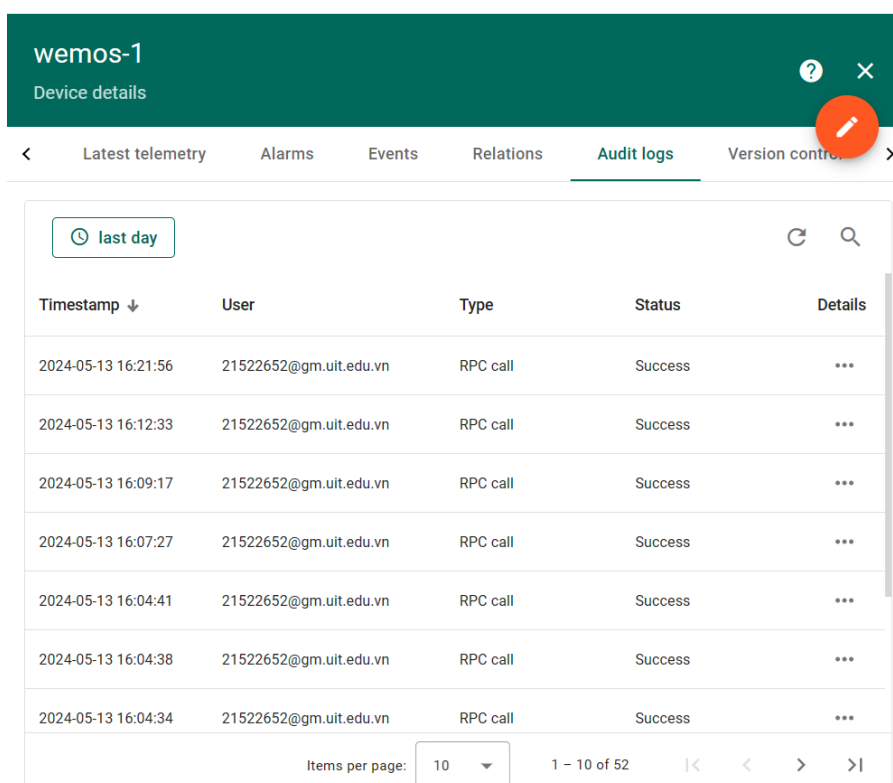
5. Logs: thể hiện các log của hệ thống để người quản trị có thể dễ dàng tương tác, quản lý đối với hệ thống. Cụ thể như sau: ghi lại các thông tin của hệ thống, giá trị cảm biến từ phần thiết bị gửi đến, các trường dữ liệu cần phải có: ID của thiết bị, tên thiết bị, tên giá trị, giá trị cảm biến, thời gian nhận dữ liệu.

To see logs of changes made to the system, on the web UI, go to the Security section, choose Audit logs. The result will be similar to the image below.

The screenshot shows the 'Audit logs' page in the ThingsBoard web UI. The page has a header with 'Security > Audit logs' and a sidebar with 'last day' and a search icon. The main content is a table with columns: Timestamp, Entity type, Entity name, User, Type, Status, and Details. The table lists several log entries for devices 'wemos-1' and 'wemos-2', showing RPC calls and telemetry deletions. The last entry is for the 'Main IoT Dashboard' being updated. The footer shows 'Items per page: 10' and '1 - 10 of 57'.

Timestamp	Entity type	Entity name	User	Type	Status	Details
2024-05-13 16:21:56	Device	wemos-1	21522652@gm.uit.edu.vn	RPC call	Success	...
2024-05-13 16:19:38	Device	wemos-2	21522652@gm.uit.edu.vn	Telemetry deleted	Success	...
2024-05-13 16:12:33	Device	wemos-1	21522652@gm.uit.edu.vn	RPC call	Success	...
2024-05-13 16:09:17	Device	wemos-1	21522652@gm.uit.edu.vn	RPC call	Success	...
2024-05-13 16:07:27	Device	wemos-1	21522652@gm.uit.edu.vn	RPC call	Success	...
2024-05-13 16:04:41	Device	wemos-1	21522652@gm.uit.edu.vn	RPC call	Success	...
2024-05-13 16:04:38	Device	wemos-1	21522652@gm.uit.edu.vn	RPC call	Success	...
2024-05-13 16:04:34	Device	wemos-1	21522652@gm.uit.edu.vn	RPC call	Success	...
2024-05-13 16:04:28	Device	wemos-1	21522652@gm.uit.edu.vn	RPC call	Success	...
2024-05-13 16:02:50	Dashboard	Main IoT Dashboard	21522652@gm.uit.edu.vn	Updated	Success	...

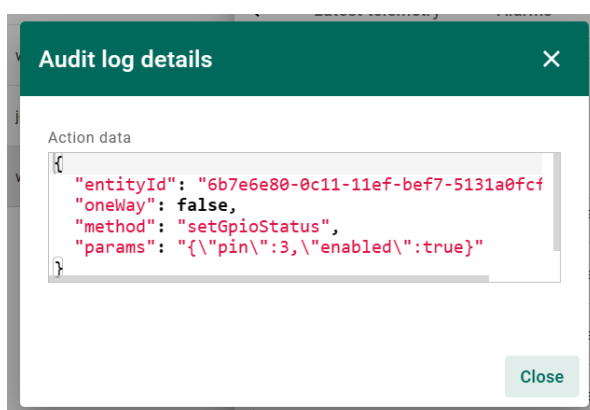
You can also go to the Devices page to see the audit logs of each device:



wemos-1				
Device details				
< Latest telemetry Alarms Events Relations <u>Audit logs</u> Version control >				
🕒 last day 🔄 🔍				
Timestamp ↓	User	Type	Status	Details
2024-05-13 16:21:56	21522652@gm.uit.edu.vn	RPC call	Success	...
2024-05-13 16:12:33	21522652@gm.uit.edu.vn	RPC call	Success	...
2024-05-13 16:09:17	21522652@gm.uit.edu.vn	RPC call	Success	...
2024-05-13 16:07:27	21522652@gm.uit.edu.vn	RPC call	Success	...
2024-05-13 16:04:41	21522652@gm.uit.edu.vn	RPC call	Success	...
2024-05-13 16:04:38	21522652@gm.uit.edu.vn	RPC call	Success	...
2024-05-13 16:04:34	21522652@gm.uit.edu.vn	RPC call	Success	...

Items per page: 10 1 - 10 of 52 < >

To see details of each log entry, click on the “...” button on the far right and the result will be shown in JSON format, including the ID of the entity you are viewing along with some attributes:



For logs related to sensor values, this type of data is available in the widget data which is the same content for Question 7 (for both viewing and downloading) so it will not be mentioned here.

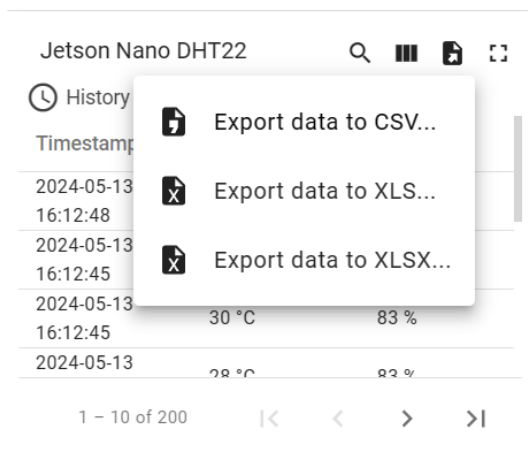
6. (Nâng cao) thực hiện triển khai IoT platform đó ở local.

We couldn't do this due to skill issue so there's that (the platform on local doesn't have the download function built-in for us).



7. (Nâng cao) Người dùng có thể lọc lại các dữ liệu mà họ mong muốn hoặc download toàn bộ dữ liệu dưới các định dạng như csv, excel,...

To do this, go to the dashboard and choose the widget data that you want to save. Click on the download icon on the top right corner of the widget and you will see some download options like below:



The csv file once downloaded:

A1 : X ✓ fx Timestamp;Temperature;humidity							
	A	B	C	D	E	F	
1	Timestamp;Temperature;humidity						
2	2024-05-13 16:11:46;30;81						
3	2024-05-13 16:11:46;30;81						
4	2024-05-13 16:11:49;30;82						
5	2024-05-13 16:11:49;30;82						
6	2024-05-13 16:11:52;30;83						
7	2024-05-13 16:11:54;30;82						
8	2024-05-13 16:11:54;30;82						
9	2024-05-13 16:11:57;30;80						
10	2024-05-13 16:11:57;30;80						
11	2024-05-13 16:12:00;28;82						
12	2024-05-13 16:12:00;28;82						

If you want to filter data of a certain type, you can do so with the column display option. In the image below, I remove temperature from the display and thus, after downloading, only humidity data shall be included.



Jetson Nano DHT22

History - last 30 days

Timestamp ↓	humidity
2024-05-13 16:12:48	79 %
2024-05-13 16:12:45	83 %
2024-05-13 16:12:45	83 %
2024-05-13 16:12:42	83 %

Columns to Display

- ☒ Timestamp
- ☐ Temperature
- ☒ humidity

1 - 10 of 200

The content of the downloaded csv file only contains data of humidity as shown below:

A1 : X ✓ fx Timestamp;humidity

	A	B	C	D	E
1	Timestamp;humidity				
2	2024-05-13 16:07:09;83				
3	2024-05-13 16:07:11;78				
4	2024-05-13 16:07:11;78				
5	2024-05-13 16:07:14;79				
6	2024-05-13 16:07:14;79				
7	2024-05-13 16:07:17;78				
8	2024-05-13 16:07:20;81				
9	2024-05-13 16:07:20;81				
10	2024-05-13 16:07:22;83				
11	2024-05-13 16:07:23;83				
12	2024-05-13 16:07:25;80				
13	2024-05-13 16:07:25;80				
14	2024-05-13 16:07:28;80				
15	2024-05-13 16:07:28;80				



YÊU CẦU CHUNG

1) Đánh giá

- Chuẩn bị tốt các yêu cầu đặt ra trong bài thực hành.
- Sinh viên hiểu và tự thực hiện được bài thực hành, trả lời đầy đủ các yêu cầu đặt ra.
- Nộp báo cáo kết quả chi tiết những đã thực hiện, quan sát thấy và kèm ảnh chụp màn hình kết quả (nếu có); giải thích cho quan sát (nếu có).
- Sinh viên báo cáo kết quả thực hiện và nộp bài.

2) Báo cáo

- File **.PDF** hoặc **.docx**. Tập trung vào nội dung, giải thích.
- Nội dung trình bày bằng Font chữ **Cambria hoặc Times New Roman** (tự nhiên, phải chuyển đổi hết báo cáo này sang 1 font chữ thống nhất) – cỡ chữ **13. Canh đều (Justify) cho văn bản. Canh giữa (Center) cho ảnh chụp.**
- Đặt tên theo định dạng: Mã lớp-LabX_MSSV1_MSSV2. (trong đó X là Thứ tự buổi Thực hành).
Ví dụ: NT532.021.1-Lab01_25520001_25520002
- Nếu báo cáo có nhiều file, nén tất cả file vào file **.ZIP** với cùng tên file báo cáo.
- Không đặt tên đúng định dạng – yêu cầu, sẽ **KHÔNG** chấm điểm bài thực hành.
- Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

Bài sao chép, trễ, ... sẽ được xử lý tùy mức độ vi phạm.

HẾT