

BÁO CÁO THỰC HÀNH

CÔNG NGHỆ INTERNET OF THINGS HIỆN ĐẠI

Lab 4

THỰC HÀNH: BUILD A SIMPLE IoT DASHBOARD

GVHD: **Phan Trung Phát**

Lớp: **NT532.021**

Họ và tên	MSSV
Nguyễn Thành Đăng	21520683
Nguyễn Trần Bảo Quốc	21520421

ĐÁNH GIÁ KHÁC (*):

Nội dung	Kết quả
Tổng thời gian thực hiện bài thực hành trung bình (1)	2 tuần
Link Video thực hiện (2) (nếu có)	Link drive demo
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.



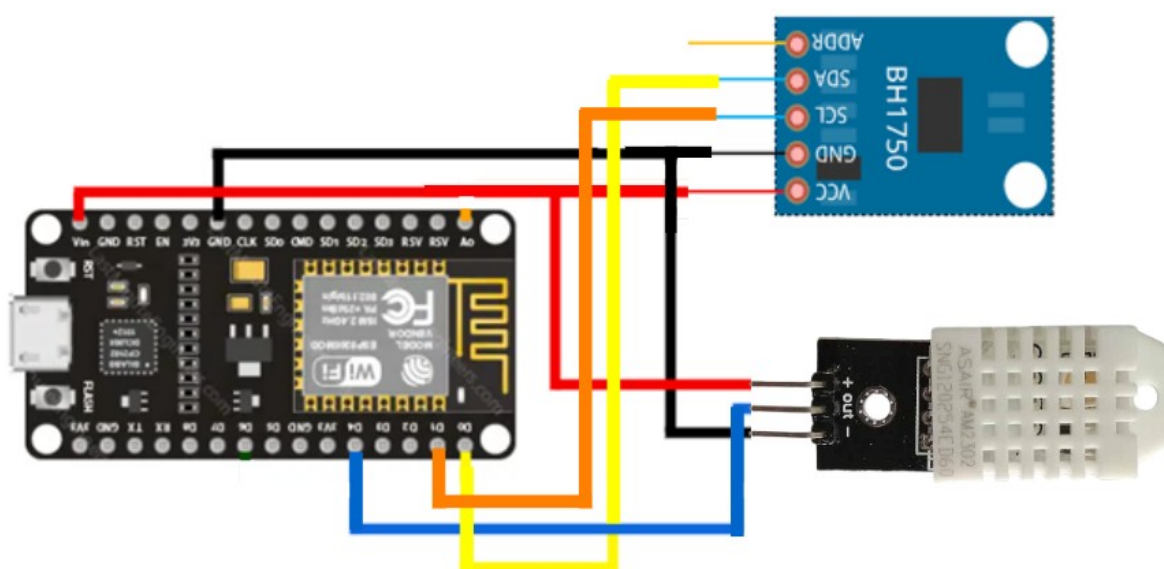
LƯU HÀNH NỘI BỘ



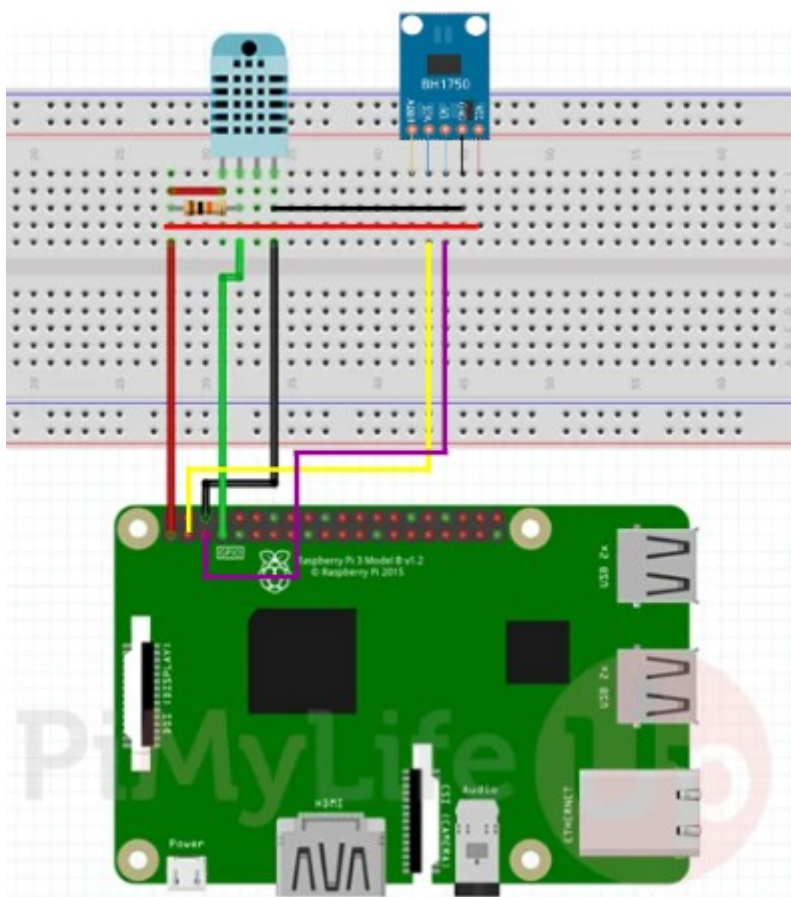
Câu hỏi 1. Device: Sinh viên thực hiện sử dụng 2 loại thiết bị cảm biến DHT22/DHT11 và BH-1750 để lấy 3 loại dữ liệu cảm biến là nhiệt độ, độ ẩm và ánh sáng. Trên Wemos D1 sử dụng cảm biến BH-1750, đồng thời lập trình để gửi các giá trị cảm biến đến hệ thống thông qua giao thức HTTP/MQTT. Trên Raspberry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

1. Minh chứng:

Link demo bài 1: [Tại đây](#)



Hình 1. Sơ đồ kết nối sensors với ESP



Hình 2. Mô hình kết nối raspberry pi với sensors

2. Giải thích:

```
esp.ino > esp > test.ino > topic
1  #include <ESP8266WiFi.h>
2  #include <PubSubClient.h>
3  #include <WiFiClientSecure.h>
4  #include <ArduinoJson.h>
5  #include "DHT.h"
6  #include <BH1750.h>
7
8  BH1750 lightMeter;
9
10 // WiFi
11 const char *ssid = "TheLight"; // Tên WiFi của bạn
12 const char *password = "20022003"; // Mật khẩu WiFi của bạn
13
14 // MQTT Broker
15 const char *topic = "iot/value";
16 const char *topicrec = "iot/detected";
17 const char *topicmac = "iot/mac";
18 const char *mqtt_broker = "192.168.119.74";
19 const char *mqtt_username = "thanh dang";
20 const char *mqtt_password = "123456";
21 const int mqtt_port = 1883;
```

Hình 3. Khai báo thư viện cho ESP

Khai báo thư viện và các thông tin kết nối đến MQTT Broker (Broker này là raspberry pi) tương tự như bài lab3, ở đây thì thêm thư viện BH1750.h để lấy giá trị của sensor.



```
24 DHT dht(DHTPIN, DHTTYPE);
25 const int DHTPIN = 5;
26 const int DHTTYPE = DHT22;
27
28 const int led = 16;
29 bool state = false;
30 int lastPostTime = 0;
31 const unsigned long post_interval = 10000;
32 String macAddr = "";
33 const String macRasp ="1";
34 const String returnRasp ="";
35 WiFiClient espClient;
36 PubSubClient client(espClient);
37
```

Hình 4. Các biến phục vụ cho chương trình

Ở đây các biến phục vụ cho chương trình tương tự hoàn toàn so với câu 5 Lab3, nhưng thêm macAddr và macRasp để phục vụ định danh thiết bị gửi (mặc định ESP này sẽ gửi tới raspberry là 1). Do lab 4 có 7 câu phát triển dần lên nên một số biến dù khai báo nhưng không hề liên quan đến chính câu này.

```
40 void setup() {
41   Serial.begin(9600);
42   WiFi.begin(ssid, password);
43   Serial.println("Connecting to WiFi...");
44   while (WiFi.status() != WL_CONNECTED) {
45     delay(500);
46     Serial.print(".");
47   }
48   Serial.println("\nConnected to the WiFi network");
49   macAddr =WiFi.macAddress();
50   Serial.print("Địa chỉ MAC: ");
51   Serial.println(macAddr);
52   client.setServer(mqtt_broker, mqtt_port);
53   client.setCallback(callback);
54   while (!client.connected()) {
55     if (client.connect("ESP8266Client", mqtt_username, mqtt_password)) {
56       Serial.println("Connected to MQTT broker");
57       client.subscribe(topic);
58       client.subscribe(topicrec);
59       client.subscribe(topicmac);
60     } else {
61       Serial.print("Failed to connect to MQTT broker, rc =");
62       Serial.print(client.state());
63       Serial.println(" retrying in 5 seconds");
64       delay(5000);
65     }
66   }
67   StaticJsonDocument<200> sensorMAC;
68   sensorMAC["macAddr"] = macAddr;
69   sensorMAC["macRasp"] = macRasp;
70   char jsonBuffer[256];
71   serializeJson(sensorMAC, jsonBuffer);
72   client.publish(topicmac, jsonBuffer);
73   dht.begin();
74   lightMeter.begin();
75   pinMode(led, OUTPUT);
76 }
```

Hình 5. Hàm setup của ESP

Luồng code: Kết nối wifi → lấy địa chỉ MAC của ESP → kết nối đến MQTT Broker qua thông tin khai báo ở trên → Nết kết nối thành công thì subscribe topic (cụ thể ý nghĩa từng topic thì thấy xem mô tả bằng video của em để rõ hơn) → gửi thông tin MAC kết nối đến MQTT → các khởi tạo sensor và led.



```
98 void loop() {
99     client.loop();
100     if (state && returnRasp)
101     {
102         tone(buzzer, 1000, 400);
103         digitalWrite(led,HIGH);
104         delay(500);
105     }
106     digitalWrite(led,LOW);
107     delay(150);
108
109     float h = dht.readHumidity();
110     float t = dht.readTemperature();
111     float lux = lightMeter.readLightLevel();
112     double hval = round(h * 100) / 100.0;
113     double tval = round(t * 100) / 100.0;
114     double lval = round(t * 100) / 100.0;
115
116     StaticJsonDocument<200> sensorData;
117
118     sensorData["temper"] = tval;
119     sensorData["humid"] = hval;
120     sensorData["light"] = lval;
121     sensorData["macAddr"] = macAddr;
122     char jsonBuffer[256];
123     serializeJson(sensorData, jsonBuffer);
124
125     if (millis() - lastPostTime >= post_interval) {
126         client.publish(topic, jsonBuffer);
127         lastPostTime = millis();
128     }
129 }
```

Hình 6. Gửi dữ liệu

Cách gửi cũng giống hệt bài 5 lab3 nhưng thêm các trường macAddr và light vào Json để gửi đi.

```
78 void callback(char *topicrec, byte *payload, unsigned int length) {
79     Serial.print("Message arrived in topic: ");
80     Serial.println(topicrec);
81     Serial.print("Message:");
82     for (int i = 0; i < length; i++) {
83         Serial.print((char)payload[i]);
84     }
85     Serial.println();
86     String data = "";
87     for (int i = 0; i < length; i++) {
88         data.concat((char)payload[i]);
89     }
90     DynamicJsonDocument doc(256);
91     DeserializationError error = deserializeJson(doc, data);
92     state = doc["state"];
93     returnRasp = doc["macRasp"];
94     Serial.println();
95     Serial.println("-----");
96 }
```

Hình 7. Đọc dữ liệu trả về

Hàm xử lý dữ liệu được trả về (nhận được từ topicrec).
Xử lý gửi dữ liệu sensors bằng Raspberry pi.


```

test4.py > on_connect
1 import time
2 import adafruit_dht
3 import board
4 from machine import I2C
5 import paho.mqtt.client as paho
6 from paho import mqtt
7
8 mqtt_broker = "192.168.119.74"
9 mqtt_port = 1883
10 mqtt_topic = "iot/detected"
11 mqtt_username = "thanhdang"
12 mqtt_password = "123456"
13
14 def on_connect(client, userdata, flags, rc, properties=None):
15     print("CONNACK received with code %s." % rc)
16 def on_publish(client, userdata, mid, properties=None):
17     print("mid: " + str(mid))
18 def on_subscribe(client, userdata, mid, granted_qos, properties=None):
19     print("Subscribed: " + str(mid) + " " + str(granted_qos))
20 def on_message(client, userdata, msg):
21     print(msg.topic + " " + str(msg.qos) + " " + str(msg.payload))
22
23 client = paho.Client(client_id="", userdata=None, protocol=paho.MQTTv311)
24 client.on_connect = on_connect
25
26 client.tls_set(tls_version=mqtt.client.ssl.PROTOCOL_TLS)
27 client.username_pw_set(mqtt_username, mqtt_password)
28 client.connect(mqtt_broker, mqtt_port)
29 client.on_subscribe = on_subscribe
30 client.on_message = on_message
31 client.on_publish = on_publish
32 client.subscribe(mqtt_topic)
33 dht_device = adafruit_dht.DHT11(board.D4)
34 i2c = I2C(scl=machine.Pin(22), sda=machine.Pin(21), freq=100000) BH1750_ADDR = 0x23
35 i2c.writeto(BH1750_ADDR, bytes([0x10]))
36
37 while True:
38     try:
39         data = i2c.readfrom(BH1750_ADDR, 2)
40         temperature_c = dht_device.temperature
41
42         humidity = dht_device.humidity
43         lux = (data[0] << 8 | data[1]) / 1.2
44         client.publish(mqtt_topic, payload={"macRasp": 1, "humid":{humidity}, "temp":{temperature_c}, "light": {lux} })
45     except RuntimeError as err:
46         print(err.args[0])
47
48     time.sleep(10.0)

```

Hình 8. Code gửi dữ liệu trực tiếp bằng raspberry pi

Khai báo thư viện (line 1-6).

Khai báo thông tin kết nối MQTT Broker (line 8 – line 12)

Kết nối đến MQTT Broker và topic bằng code chuẩn được cung cấp từ **paho** (line 14 – line 32).

Đọc và publish dữ liệu đến topic (line 40 – line 45).

Chỉ cần chạy code python và đã nối dây giống sơ đồ thì dữ liệu đã được gửi tới MQTT Broker.



```
models > JS Data.js > dataSchema > temp
1  const mongoose = require('mongoose');
2
3  const dataSchema = new mongoose.Schema(
4    {
5      temp: {
6        type: Number,
7      },
8      humid: {
9        type: Number,
10     },
11     light: {
12       type: Number,
13     },
14     macAddr:
15     {
16       type: String,
17     },
18     timestamp: {
19       type: Date,
20       default: Date.now,
21     },
22   }
23 );
24 module.exports = mongoose.model("Data", dataSchema)
```

Hình 9. Model dữ liệu lưu trữ

Model lưu trữ dữ liệu sensors sử dụng MongoDB.

```
67 function handleMessage(topic, message, server, listData) {
68   if (topic === process.env.MQTT_TOPIC_SEND) {
69     const parsedMessage = JSON.parse(message.toString());
70     const newData = new Data({
71       macAddr: parsedMessage.macAddr,
72       temp: parsedMessage.temper,
73       humid: parsedMessage.humid,
74       light: parsedMessage.light
75     });
76     newData.save()
77       .then(() => {
78         console.log(parsedMessage)
79         const currentDate = new Date();
80         const hours = currentDate.getHours();
81         const minutes = currentDate.getMinutes();
82         listData.push({ name: `${hours}:${minutes}`, humid: parsedMessage.humid, temp: parsedMessage.temper, light: parsedMessage.light, macAddr: parsedMessage.macAddr });
83         const latestData = listData.slice(Math.max(listData.length - 20, 0));
84         server.sockets.emit('message', latestData);
85         server.sockets.emit('value', { humid: parsedMessage.humid, temp: parsedMessage.temper, light: parsedMessage.light, macAddr: parsedMessage.macAddr });
86         processStatus(parsedMessage, server);
87       })
88       .catch(err => console.error('Error saving data:', err));
89   }
90 }
```

Hình 10. Lưu lại dữ liệu vào model

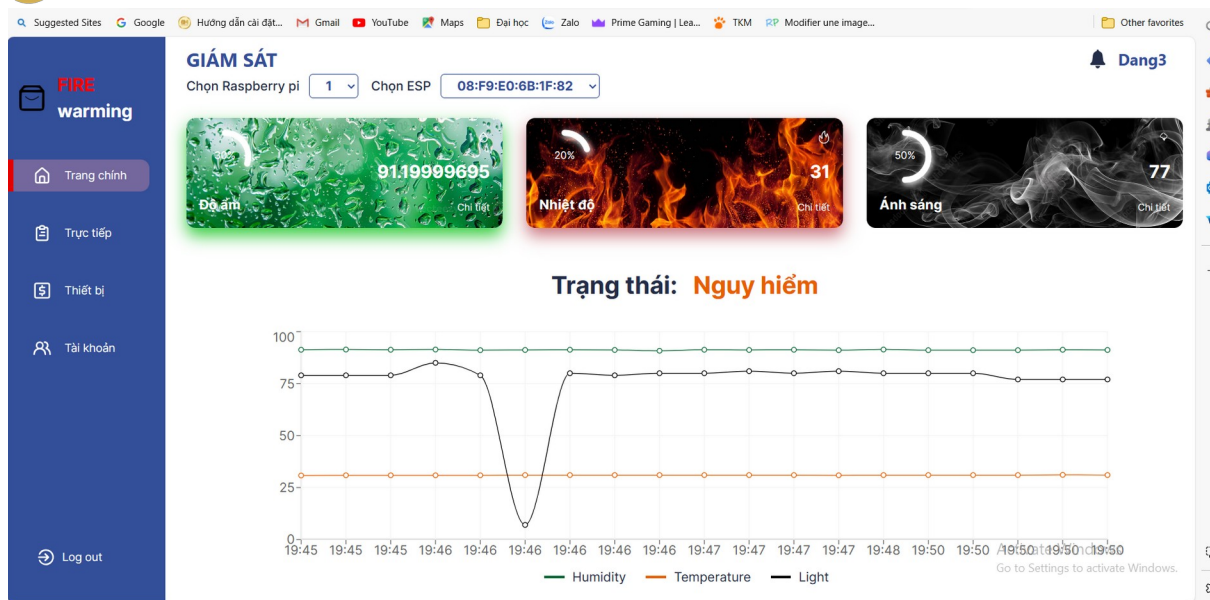
Sẽ có rất nhiều thông tin trong code do có liên quan đến các câu sau của bài lab nên em sẽ giải thích nó chi tiết trong video riêng (video giải thích không phải demo).

Câu hỏi 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,...).

1. Minh chứng:


Thầy có thể xem báo cáo toàn bộ các câu qua [demo](#) (do các câu em kết hợp vào 1 bài).





Hình 11. Dashboard


 FIRE


warming

 Trang chính

 Trực tiếp

 Thiết bị

 Tài khoản

 Log out

Các RASPBERRY PI đã kết nối

Kết nối với Raspberry pi để theo dõi

Thêm

STT	Raspberry Pi ID	Ngày tạo	Xóa
1	1	2024-05-16T09:00:31.113Z	Xóa

Các ESP đã được phép kết nối

STT	Raspberry Pi ID	MAC ESP	Ngày tạo	Xóa
-----	-----------------	---------	----------	-----

Các ESP đang chờ sự cấp phép

Raspberry Pi ID	MAC ESP	Ngày tạo	Chấp nhận	Xóa
-----------------	---------	----------	-----------	-----

Activate Windows

Go to Settings to activate Windows.

Hình 12. Đa kết nối ESP

2. Giải thích:



```
models > js MacESP.js > macSchema
1  const mongoose = require('mongoose');
2
3  const macSchema = new mongoose.Schema(
4    {
5      macAddr: {
6        type: String,
7        unique: true
8      },
9      macRasp:
10     {
11       type: String,
12     }
13   }, {timestamps:true}
14 );
15 module.exports = mongoose.model("MacESP", macSchema)
```

Hình 13. Model lưu trữ các thông tin về ESP đã kết nối (backend)

Cấu trúc cơ sở dữ liệu mongo lưu trữ thông tin của một ESP.

```
18
19  getAllESP: async(req,res) =>
20  {
21    try {
22
23      const macEsps = await MacESP.find({macRasp: req.params.macRasp})
24      res.status(200).json(macEsps);
25    } catch (error) {
26      res.status(500).json('Lỗi')
27    }
28  },
29  getAll: async(req,res) =>
30  {
31    try {
32
33      const macEsps = await MacESP.find()
34      res.status(200).json(macEsps);
35    } catch (error) {
36      res.status(500).json('Lỗi')
37    }
38  },
```

Hình 14. Lấy thông tin của các ESP theo id của Rasp và tất cả Rasp (backend)

Route xử lý việc lấy tất cả các ESP lọc theo địa chỉ Rasp và Route xử lý lấy toàn bộ ESP mà cơ sở dữ liệu có.

Câu hỏi 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.

1. Minh chứng:

Thầy có thể xem báo cáo toàn bộ các câu qua [demo](#) (do các câu em kết hợp vào 1 bài).

2. Giải thích:



```
67 function handleMessage(topic, message, server, listData) {
68   if (topic === process.env.MQTT_TOPIC_SEND) {
69     const parsedMessage = JSON.parse(message.toString());
70     const newData = new Data({
71       macAddr: parsedMessage.macAddr,
72       temp: parsedMessage.temper,
73       humid: parsedMessage.humid,
74       light: parsedMessage.light
75     });
76     newData.save()
77       .then(() => {
78         console.log(parsedMessage)
79         const currentDate = new Date();
80         const hours = currentDate.getHours();
81         const minutes = currentDate.getMinutes();
82         listData.push({ name: `${hours}:${minutes}`, humid: parsedMessage.humid, temp: parsedMessage.temper, light: parsedMessage.light, macAddr: parsedMessage.macAddr });
83         const latestData = listData.slice(Math.max(listData.length - 20, 0));
84         server.sockets.emit('message', latestData);
85         server.sockets.emit('value', { humid: parsedMessage.humid, temp: parsedMessage.temper, light: parsedMessage.light, macAddr: parsedMessage.macAddr });
86         processStatus(parsedMessage, server);
87       })
88       .catch(err => console.error('Error saving data:', err));
89   }
90 }
```

Hình 15. Hàm gửi data dạng realtime cho client (server)

Đây chính là hàm lưu lại data vào model ngay khi nhận từ sensors. Và nó ngay khi nhận data từ sensors thì lập tức gửi các data này đến client thông qua socket để đạt được realtime (line 84).

```
12 useEffect(() => {
13   const socket = socketIOClient(process.env.REACT_APP_SOCKET);
14   socket.on('value', (newData) => {
15     let dt = []
16     dt.push(newData.humid);
17     dt.push(newData.temp);
18     dt.push(newData.light);
19     setdtReal(dt);
20   });
21 }, [])
```

Hình 16. Realtime bằng card từng giá trị (client)

Nhận giá trị realtime này, sẽ được biểu diễn trên các thẻ thông số tương ứng.

```
92 <Card
93   key={card.id}
94   title={card.title}
95   color={card.color}
96   barValue={card.barValue}
97   value={dtReal[index]}
98   png={card.png}
99   series={dataSeries[index].series}
100 />
101 </div>
```

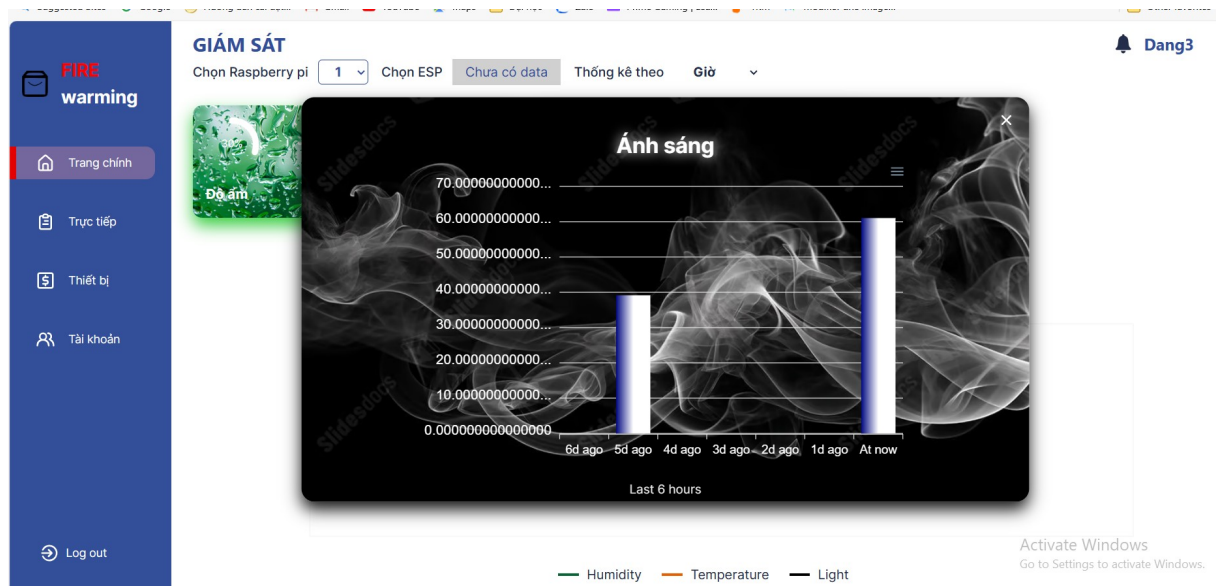
Hình 17. Hiển thị giá trị realtime

Truyền giá trị nhận được từ socket ở trên vào component Card để người dùng có thể quan sát.

Câu hỏi 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..

1. Minh chứng:

Thầy có thể xem báo cáo toàn bộ các câu qua [demo](#) (do các câu em kết hợp vào 1 bài).



Hình 18. Biểu đồ

2. Giải thích:

```
67 function handleMessage(topic, message, server, listData) {
68   if (topic === process.env.MQTT_TOPIC_SEND) {
69     const parsedMessage = JSON.parse(message.toString());
70     const newData = new Data({
71       macAddr: parsedMessage.macAddr,
72       temp: parsedMessage.temper,
73       humid: parsedMessage.humid,
74       gas: parsedMessage.gas
75     });
76     newData.save()
77       .then(() => {
78         console.log(parsedMessage)
79         const currentDate = new Date();
80         const hours = currentDate.getHours();
81         const minutes = currentDate.getMinutes();
82         listData.push({ name: `${hours}:${minutes}`, humid: parsedMessage.humid, temp: parsedMessage.temper, gas: parsedMessage.gas, macAddr:
83           const latestData = listData.slice(Math.max(listData.length - 20, 0));
84           server.sockets.emit('message', latestData);
85           server.sockets.emit('value', { humid: parsedMessage.humid, temp: parsedMessage.temper, gas: parsedMessage.gas, macAddr: parsedMessage
86             processStatus(parsedMessage, server);
87           });
88           .catch(err => console.error('Error saving data:', err));
89     }
90 }
```

Hình 19. Hàm gửi data dạng realtime cho client (server)

Đây chính là hàm lưu lại data vào model ngay khi nhận từ sensors. Và nó ngay khi nhận data từ sensors thì lập tức gửi các data này đến client thông qua socket để đạt được realtime (line 85).



```
14
15     useEffect(() => {
16         if (clear)
17         {
18             setData([])
19             setClear(false)
20         }
21         const socket = socketIOClient('http://localhost:9000');
22         socket.on('message', (newData) => {
23             if(newData.macAddr == props.macAddr)
24                 setData(newData);
25         });
26         socket.on('status', (newData) => {
27             if (newData.macAddr == props.macAddr)
28             {
29                 setMsg(newData.msg);
30                 setStatus(newData.status)
31             }
32         });
33     }, []);
34     //useEffect(() => {
```

Hình 20. Realtime qua ChartLine (Client)

Nhận giá trị và cập nhật các giá trị này vào Chart dạng line.

```
57     </> <data></data> </> </>
58     <LineChart width={1100} height={300} data={data} margin={{ top: 5, right: 20, left: 10, bottom: 5 }}>
59         <XAxis dataKey="name" />
60         <YAxis />
61         <Tooltip />
62         <CartesianGrid stroke="#f5f5f5" />
63         <Line type="monotone" dataKey="humid" name="Humidity" stroke="#076836" yAxisId={0} />
64         <Line type="monotone" dataKey="temp" name="Temperature" stroke="#e96000" yAxisId={0} />
65         <Line type="monotone" dataKey="light" name="Gas" stroke="#000000" yAxisId={0} />
66     </LineChart>
67
```

Hình 21. Cập nhật giá trị realtime vào biểu đồ LineChart

Truyền data vào biểu đồ realtime, và dùng các datakey để lấy ra các giá trị cụ thể cho từng line của biểu đồ LineChart này.

Câu hỏi 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,....

1. Minh chứng:

Thầy có thể xem báo cáo toàn bộ các câu qua [demo](#) (do các câu em kết hợp vào 1 bài).

2. Giải thích:



```
1  const mongoose = require('mongoose');
2
3  const logSchema = new mongoose.Schema(
4    {
5      msg: {
6        type: String
7      },
8      macAddr:
9        {
10         type: String,
11       },
12      timestamp: {
13        type: Date,
14        default: Date.now,
15      },
16    }
17  );
18  module.exports = mongoose.model("Log", logSchema)
```

Hình 22. Model lưu trữ log

Model log lưu lại message khi xử lý bất thường thay vì log toàn data như thấy yêu cầu vì mặc định data của em đã được lưu vào model Data khi nhận dữ liệu (Hình 10).

```
18      let msg;
19      if (state == 3)
20      {
21        msg = `Dữ liệu thu được trên ${parsedMessage.macAddr} cực kì nguy hiểm`;
22      } else if (state == 2)
23      {
24        msg = `Dữ liệu thu được trên ${parsedMessage.macAddr} nguy hiểm`;
25      }
26      lastProcessTime = currentTime;
27      (async () => {
28        const newLog = new Log( {
29          msg: msg,
30          macAddr: parsedMessage.macAddr
31        })
32        await newLog.save()
```

Hình 23. Lưu log

Code này sẽ bỏ trong điều kiện kiểm tra bất thường, giả sử nếu nó bất thường thì sẽ lưu lại message của Rasp đó.

Câu hỏi 6: 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,...

1. Minh chứng:

Thầy có thể xem báo cáo toàn bộ các câu qua [demo](#) (do các câu em kết hợp vào 1 bài).



GIÁM SÁT

Chọn Raspberry pi

1

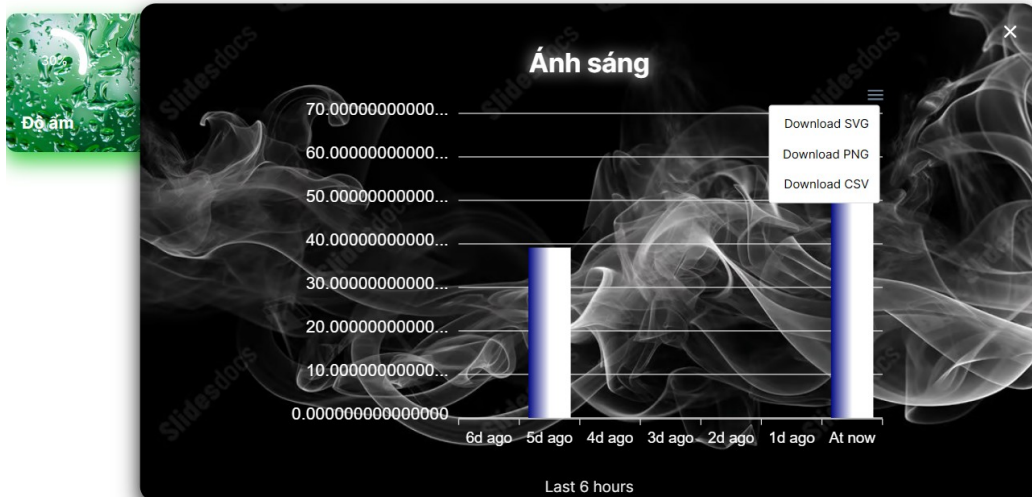
Chọn ESP

Chưa có data

Thống kê theo

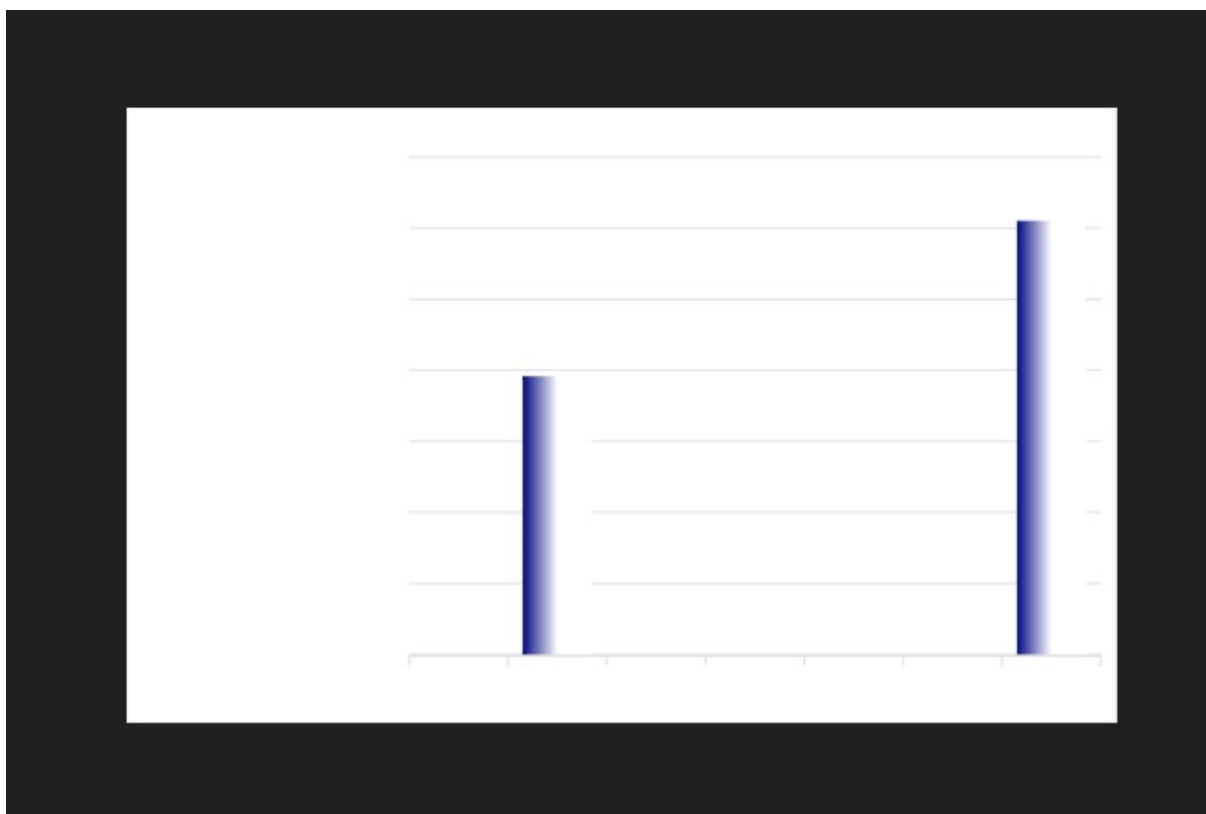
Giờ

Dan



Activate Windows

Hình 24. Download logs



Hình 25. Log dạng png

2. Giải thích:

```
7 import ReactApexChart from "react-apexcharts";  
8 import ApexCharts from 'apexcharts';
```

Hình 26. Thư viện sử dụng biểu đồ hỗ trợ tải data



```
62 function ExpandedCard({ param, setExpanded }) {
63   const data = {
64     options: {
65       chart: {
66         id: 'chartData',
67         type: "bar",
68         height: "auto",
69       },
70       dropShadow: {
71         enabled: false,
72         enabledOnSeries: undefined,
73         top: 0,
74         left: 0,
75         blur: 3,
76         color: "#000",
77         opacity: 0.35,
78       },
79
80       fill: {
81         colors: ["#fff"],
82         type: "gradient",
83       },
84       dataLabels: {
85         enabled: false,
86         style: {
87           fontSize: '12px',
88           colors: ["#304758"]
89         }
90       },
91       stroke: {
```

Hình 27. Option tùy chỉnh trong biểu đồ

Sử dụng thuộc tính type: “bar” để biểu đồ được import có thể cấp phép tải dạng **png**, **csv** và **svg**.

```
169 <ReactApexChart className={text-black} options={data.options} series={param.series} type="bar"/>
```

Hình 28. Áp dụng nó vào Biểu đồ