

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN ĐIỆN TỬ - VIỄN THÔNG



REPORT

SENSOR AND UPLOAD DATA WITH MQTT

Họ tên: ĐOÀN QUANG LƯU
MSSV: 20203884

Hà Nội, April 7, 2023

MỤC LỤC

DANH MỤC TỪ VIẾT TẮT	i
DANH MỤC HÌNH VẼ	ii
DANH MỤC BẢNG BIỂU	iii
TÓM TẮT BÁO CÁO	iv
CHƯƠNG 1. CHƯƠNG MỞ ĐẦU	1
CHƯƠNG 2. PHÂN TÍCH ĐỀ TÀI	2
2.1 Chỉ tiêu kỹ thuật	2
2.1.1 Chỉ tiêu chức năng	2
2.1.2 Chỉ tiêu phi chức năng	2
2.2 Block Diagram	3
2.3 Lựa chọn giải pháp	3
2.3.1 Lựa chọn phần cứng	3
2.3.2 Lựa chọn phần mềm	8
2.4 Linh kiện sử dụng	9
2.5 Nguyên lý hoạt động cơ bản của linh kiện	9
2.5.1 Nguyên lý cảm biến	9
2.5.2 Giao thức MQTT	11
CHƯƠNG 3. THIẾT KẾ	12
3.1 Khối cảm biến	12
3.1.1 Cảm biến nhiệt/ độ ẩm với ESP32	12
3.1.2 Cảm biến bụi với ESP8266	14
3.2 Cài đặt wifi và gửi dữ liệu lên server	15
3.2.1 Cài đặt wifi	15

3.2.2	Gửi dữ liệu lên server	15
3.3	Khôi server	16
3.3.1	MQTT Broker	16
3.3.2	Data Bridge	17
3.3.3	Database	18
3.3.4	Front-end	18
CHƯƠNG 4. THỰC HIỆN MẠCH		19
4.4	Kiểm tra chức năng của hệ thống	19
4.5	Kết quả đo được	20
CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN		21
TÀI LIỆU THAM KHẢO		22

DANH MỤC TỪ VIẾT TẮT

IoT	Internet of Thíngs
MQTT	Message Queuing Telemetry Transport
SCADA	Supervisory Control And Data Acquisition

DANH MỤC HÌNH VẼ

Hình 2.1	Sơ đồ khối của hệ thống	3
Hình 2.2	Cảm biến nhiệt và đồ thị của trổ phụ thuộc vào nhiệt độ	9
Hình 2.3	Cảm biến độ ẩm	10
Hình 2.4	Cảm biến độ bụi và đồ thị điện áp ra phụ thuộc vào nồng độ bụi	10
Hình 2.5	Cơ chế hoạt động cơ bản của MQTT	12
Hình 3.1	Broker với HiveMQ	16
Hình 3.2	Node-red trên Pi	17
Hình 3.3	Node-red trên Pi	18
Hình 3.4	Hiển thị dữ liệu bằng grafana	19
Hình 4.1	Kết quả cuối cùng	20

DANH MỤC BẢNG BIỂU

Bảng 2.1	Vi xử lý khối cảm biến	4
Bảng 2.2	Linh kiện cảm biến nhiệt độ/ độ ẩm	5
Bảng 2.3	Linh kiện cảm biến bụi	6
Bảng 2.4	Linh kiện	9

TÓM TẮT BÁO CÁO

Internet of Things (viết tắt là IoT) là một kịch bản của thế giới, khi mà mỗi đồ vật, con người được qua một mạng duy nhất mà không cần đến sự tương tác trực tiếp giữa người với người, hay người với máy tính. IoT đã phát triển từ sự hội tụ của công nghệ không dây, công nghệ vi cơ điện tử và Internet. Nói đơn giản là một tập hợp các thiết bị có khả năng kết nối với nhau, với Internet và với thế giới bên ngoài để thực hiện một công việc nào đó. Thời đại IoT yêu cầu một giao thức kết nối mới để đảm bảo hỗ trợ đầy đủ cho các thiết bị vật lý thực tế. Để giải quyết vấn đề này, Message Queuing Telemetry Transport (MQTT) đang dần trở nên phổ biến. Sản phẩm được thực hiện với mục đích gửi dữ liệu lên server và database có sẵn bằng thiết bị vi xử lý. Sản phẩm trong báo cáo sử dụng vi điều khiển: Esp8266 có tác dụng cảm biến và Raspberry Pi Zero 2W với chức năng tạo server, tín hiệu thu thập được lập trình bằng Arduino IDE và gửi dữ liệu bằng giao thức MQTT. Bài báo cáo gồm 3 phần chính: giới thiệu hệ thống, chi tiết và kết luận.

CHƯƠNG 1. CHƯƠNG MỞ ĐẦU

Dữ liệu là một tập hợp các dữ kiện, chẳng hạn như số, từ, hình ảnh, nhầm đo lường, quan sát hoặc chỉ là mô tả về sự vật. Vì vậy, để phân tích và rút ra kết luận một sự vật, hiện tượng nào đó, người ta cần dựa vào một hệ thống dữ liệu để làm cơ sở phân tích. Như vậy, nếu như dữ liệu sau khi thu thập không được lưu trữ lại có thể khiến cho dữ liệu ấy trở nên vô nghĩa và không có một kết luận chính xác. Qua đó, việc lưu trữ trở thành việc tối thiểu khi thu thập dữ liệu. Người ta có thể lưu trữ dữ liệu trong bộ nhớ máy tính, tuy nhiên để thuận tiện và nhiều người có thể truy cập, phân tích dữ liệu đó, dữ liệu thường được lưu trữ trên server. Vì vậy, hệ thống tập trung vào lập trình gửi dữ liệu lên server.

Máy chủ (Tiếng anh là Server) là một máy tính được kết nối với một mạng máy tính hoặc internet, có IP tĩnh, có năng lực xử lý cao và trên đó người ta cài đặt các phần mềm để phục vụ cho các máy tính khác truy cập để yêu cầu cung cấp các dịch vụ và tài nguyên. Như vậy về cơ bản máy chủ cũng là một máy tính, nhưng được thiết kế với nhiều tính năng vượt trội hơn, năng lực lưu trữ và xử lý dữ liệu cũng lớn hơn máy tính thông thường rất nhiều. Máy chủ thường được sử dụng cho nhu cầu lưu trữ và xử lý dữ liệu trong một mạng máy tính hoặc trên môi trường internet. Máy chủ là nền tảng của mọi dịch vụ trên internet, bất kỳ một dịch vụ nào trên internet muốn vận hành cũng đều phải thông qua một máy chủ nào đó. Server được xem là thành phần không thể thiếu trong hệ thống lưu trữ máy chủ. Các hệ thống Server hiện nay được cung cấp dưới nhiều giải pháp khác nhau.

Bằng cách sử dụng hệ thống này, có thể nhận dữ liệu từ cảm biến và hiển thị một cách trực quan lên trình duyệt từ bất cứ đâu có kết nối internet, rất phù hợp với các dự án giám sát các tham số môi trường từ xa.

Nội dung của báo cáo gồm:

- Phần mở đầu giới thiệu đê tài.
- Giới thiệu về linh kiện sản phẩm sử dụng.
- Giải thích nguyên lý và triển khai sản phẩm.
- Kết quả thu được.
- Hướng phát triển và kết luân.

CHƯƠNG 2. PHÂN TÍCH ĐỀ TÀI

2.1 Chỉ tiêu kỹ thuật

2.1.1 Chỉ tiêu chức năng

Sản phẩm hoạt động ổn định, cảm biến và giữ dữ liệu lên server, quản lý và hiển thị dữ liệu dễ nhìn, dễ hiểu.

Input:

- Điện áp vào sensor: 3.3VDC
- Tín hiệu vào Raspberry Pi Zero: 5VDC-3A
- Độ ẩm: Khoảng 0-100 %
- Nhiệt độ: khoảng $0^{\circ}\text{C} - 50^{\circ}\text{C}$
- Bụi: khoảng $200-800 \mu\text{g}/\text{m}^3$

Ouput: Dữ liệu tương ứng với nhiệt độ và độ ẩm được gửi lên database và được biểu thị dưới dạng đồ thị/ biểu đồ.

Quan hệ giữa Input và Output: Cơ bản thì dữ liệu output thu thập được có giá trị tương ứng với giá trị nhiệt độ và độ ẩm thu được. Sai số như nhau:

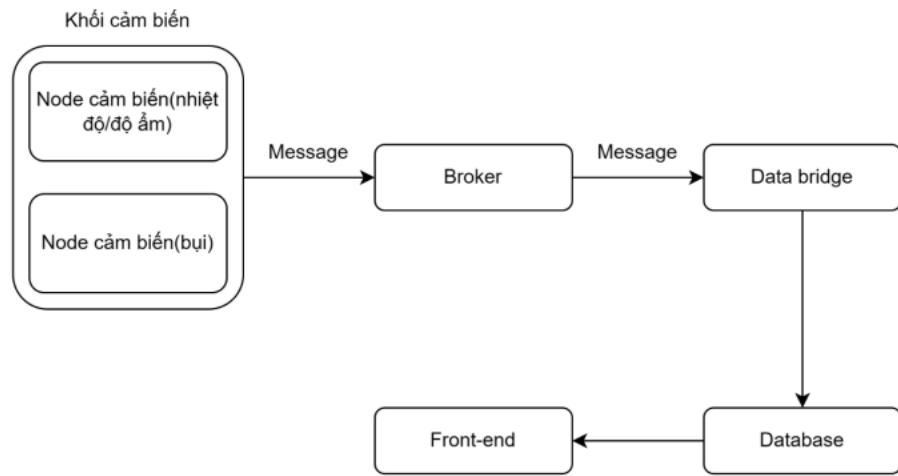
- Độ ẩm: Sai số $\pm 5\%$
- Nhiệt độ: Sai số $\pm 2^{\circ}\text{C}$
- Bụi: Sai số $\pm 15\%$

2.1.2 Chỉ tiêu phi chức năng

Một số chỉ tiêu của sản phẩm:

- Kích thước: khoảng 15x25x6 cm
- Màu: đen
- Vật liệu: Nhựa
- Giá: $\sim 2.000.000$ VND

2.2 Block Diagram



Hình 2.1 Sơ đồ khái của hệ thống

Vai trò của các khôi:

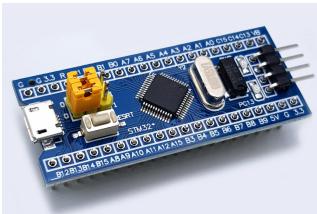
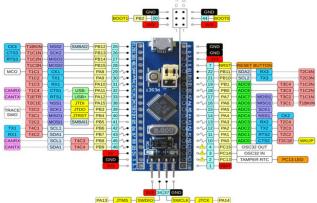
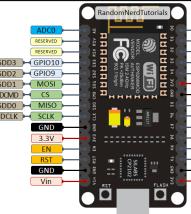
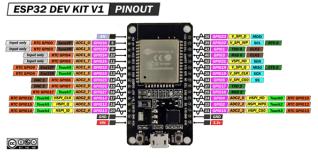
- *Khối cảm biến*: Gửi thông tin dựa trên hiện trạng môi trường xung quanh(trong vùng hoạt động của khối cảm biến)
- *Broker*: Broker nhận những thông tin subscribe (Subscriptions) từ publisher, nhận thông điệp, chuyển những thông điệp đến các Subscriber tương ứng dựa trên Subscriptions từ client.
- *Data bridge*: Đưa luồng thông tin nhận được từ broker đến Database và phân loại, xử lý dữ liệu nhận được.
- *Database*: Nhận thông tin nhận được từ data bridge và lưu trữ.
- *Front-end*: Lấy được dữ liệu từ database và thể hiện dữ liệu cho người dùng động của khối cảm biến.

2.3 Lựa chọn giải pháp

Hệ thống có khá nhiều linh kiện/ phần mềm để lựa chọn, vì vậy việc thống kê và phân tích là quan trọng. Sau đây là bảng thống kê các linh kiện trên thị trường và lựa chọn linh kiện:

2.3.1 Lựa chọn phần cứng

Khối cảm biến

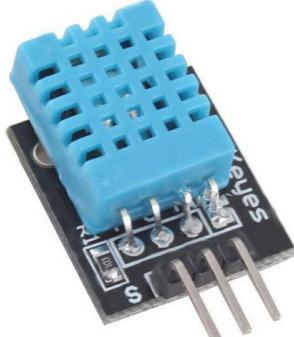
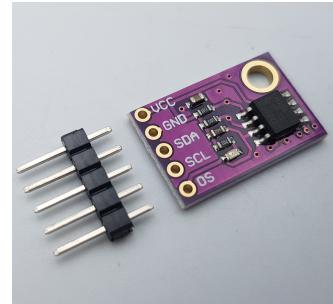
Thông số	STM32	ESP8266	ESP32
Giá thành	~ 100.000VND	~ 120.000VND	~ 170.000VND
Kiến trúc	ARM Cortex-M3	RISC 32-bit	SoC dual Xtensa LX6 32-bit
Bộ nhớ	Flash 64KB, SRAM 20KB	Flash 1MB, SRAM 80KB	Flash 4MB, SRAM 520KB
Điện áp vào	2.0V - 3.6VDC	2.5V-3.6VDC	2.2V-3.6VDC
Tốc độ	72MHz	80MHz	240MHz
Hình ảnh			
Chân GPIO			

Bảng 2.1 Vi xử lý khối cảm biến

Qua bảng, lựa chọn :

- **ESP8266**: có sẵn, đáp ứng đủ chỉ tiêu yêu cầu.
- **ESP32**: tốc độ cao, giá cả phù hợp.

Cảm biến nhiệt

Thông số	DHT11	SHT71	CJMCU-75
Giá thành	~ 20.000VND	~ 500.000VND	~ 40.000VND
Khoảng đo nhiệt	0°C – 50°C	-40°C – 123.8°C	-55°C – 125°C
Khoảng đo ẩm	20% - 90%	0 - 100%RH	None
Sai số nhiệt	±2°C	±0.4°C	±0.125°C
Sai số ẩm	±5%	±3%	
Điện áp vào	3V - 5,5VDC	2,1V - 3,6VDC	2.8V - 5.5VDC
Hình ảnh			

Bảng 2.2 Linh kiện cảm biến nhiệt độ/ độ ẩm

Việc up dữ liệu càng đa dạng càng đánh giá được kết quả tốt, vì vậy linh kiện có thể cảm biến được cả nhiệt độ và độ ẩm được ưu tiên, ngoài ra mục tiêu chính của sản phẩm là up dữ liệu lên server, cảm biến nhiệt không cần thực sự chính xác, hay độ rộng cảm biến cao, bởi DHT11 có giá thành rẻ, vì vậy trong bài tập này **DHT11** được lựa chọn.

Cảm biến bụi

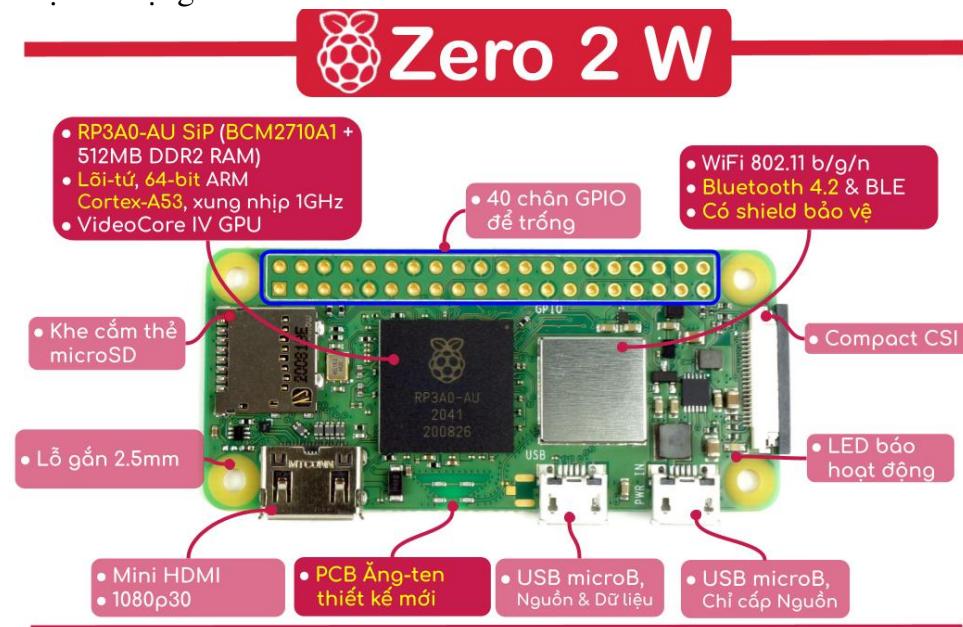
Thông số	GP2Y10	PMS7003
Giá thành	200.000 VND	400.000 VND
Khoảng đo	$200\text{-}800 \mu\text{g}/\text{m}^3$	$0\text{ }500 \mu\text{g}/\text{m}^3$
Độ chính xác	$\pm 15\%$	$\pm 10\%$
Điện áp vào	5VDC	5VDC
Hình ảnh		
Chân		

Bảng 2.3 Linh kiêm cảm biến bụi

Cũng tương tự lí do với cảm biến nhiệt, cảm biến **GP2Y10** được sử dụng để thu thập dữ liệu về bụi của môi trường.

Khối server:

Với mục tiêu làm quen với hệ điều hành Linux và Raspberry Pi, Raspberry Pi Zero 2W được sử dụng.



2.3.2 Lựa chọn phần mềm

- *Broker*: HiveMQ: Hoàn toàn có thể thay bằng các broker khác như Mosquitto, ActiveMQ, Mosca,... Tuy nhiên HiveMQ được sử dụng bởi khả năng hoạt động online khá tiện để sử dụng.
- *Data bridge*: Node-RED: Lựa chọn Node-RED do Node-RED giúp việc sử dụng giao thức MQTT dễ dàng hơn
- *Database*: InfluxDB: hoàn toàn có thể thay thế bằng các database khác như MySQL, MongoDB hay MariaDB.
- *Front-end* : Grafana: giao diện đẹp, dễ sử dụng với nhiều phần mềm Database.

2.4 Linh kiện sử dụng

Sau khi chọn các vi xử lý, cảm biến được sử dụng, linh kiện được tổng hợp lại trong bảng sau:

Bảng 2.4 Linh kiện sử dụng trong sản phẩm

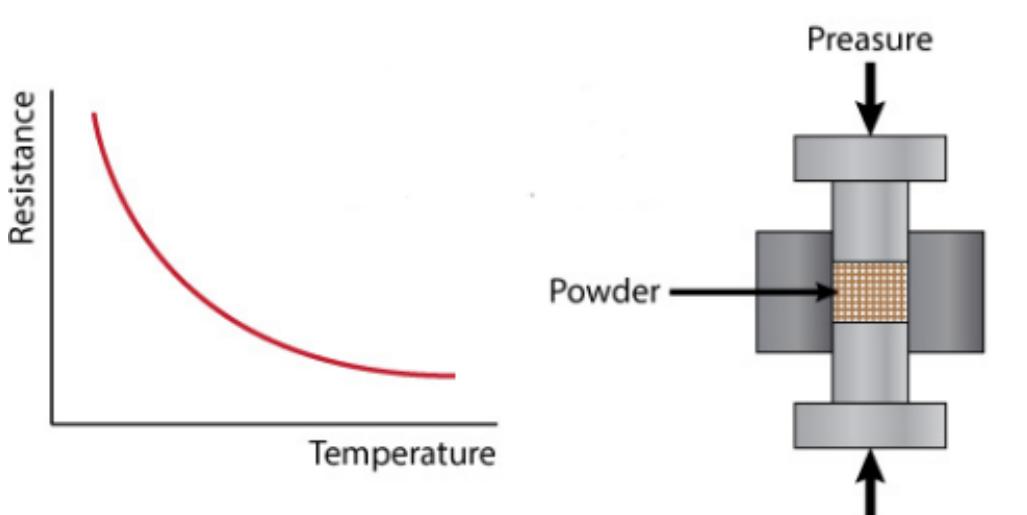
Linh kiện	Số lượng
DHT11	1
GP2Y10	1
Breadboard 5x5cm	1
ESP8266	1
ESP32	1
Raspberry Pi Zero 2W	1
Tụ gốm $220\mu F$	1
Trở 140R	1

2.5 Nguyên lý hoạt động cơ bản của linh kiện

2.5.1 Nguyên lý cảm biến

Cảm biến nhiệt:

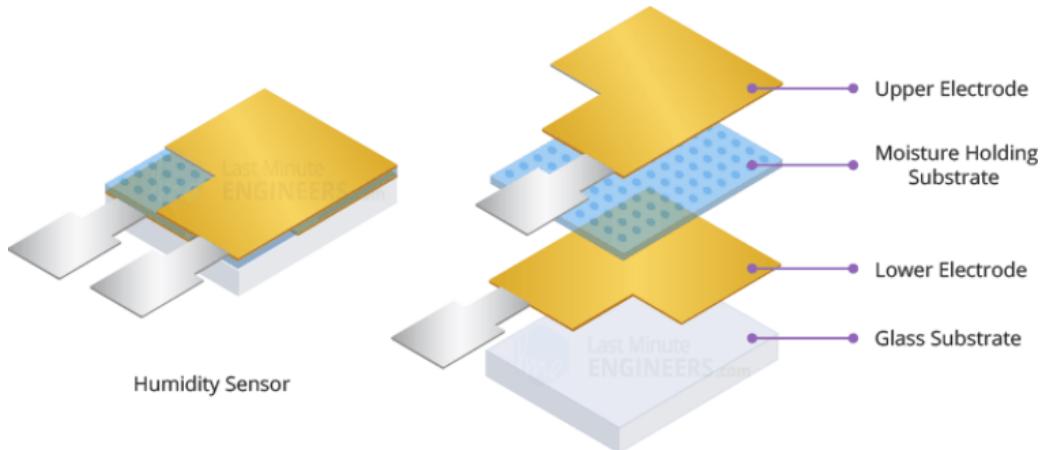
Để đo nhiệt độ, DHT22 sử dụng cảm biến nhiệt độ NTC hay điện trở nhiệt. Một nhiệt điện trở thực chất là một biến trở có thể thay đổi điện trở của nó khi nhiệt độ thay đổi. Các cảm biến này được chế tạo bằng cách nung kết các vật liệu bán dẫn như gốm sứ hoặc polyme để tạo ra những thay đổi lớn hơn về điện trở chỉ với những thay đổi nhỏ về nhiệt độ. Khi nhiệt độ tăng lên, giá trị của điện trở sẽ giảm.



Hình 2.2 Cảm biến nhiệt và đồ thị của trở phụ thuộc vào nhiệt độ

Cảm biến độ ẩm:

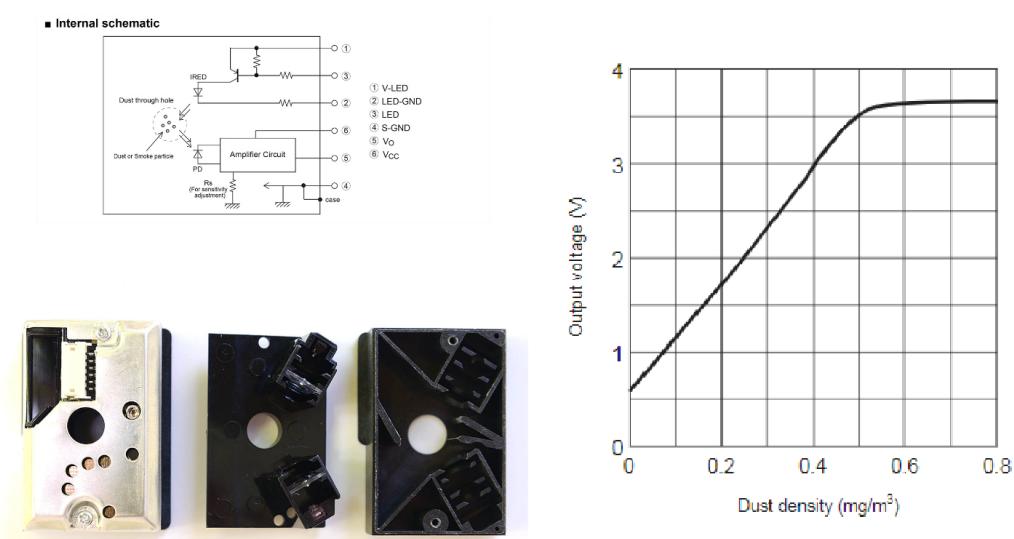
Để đo độ ẩm, bộ phận cảm biến tương ứng có hai điện cực với chất nền giữ ẩm giữa chúng. Khi độ ẩm thay đổi, độ dẫn của chất nền thay đổi hay điện trở giữa hai điện cực này sẽ thay đổi. Sự thay đổi điện trở này được đo và xử lý bởi IC giúp cho vi điều khiển luôn sẵn sàng để đọc.



Hình 2.3 Cảm biến độ ẩm

Cảm biến bụi:

Ở đây có thể thấy sẽ có 02 bộ phận dùng để truyền và nhận hồng ngoại (IR LED và Phototransistor). 02 bộ phận này được đặt chênh gốc với nhau. Khi có bụi bay vào, tia hồng ngoại từ IR LED sẽ bị dội vào Phototransistor, lúc này điện áp từ phototransistor sẽ được đưa đến mạch khuếch đại (Amplifier) và xuất ra chân Vo.



Hình 2.4 Cảm biến độ bụi và đồ thị điện áp ra phụ thuộc vào nồng độ bụi

2.5.2 Giao thức MQTT

[1] **Khái niệm:** MQTT (Message Queuing Telemetry Transport) là giao thức truyền thông điệp (message) theo mô hình publish/subscribe (cung cấp / thuê bao), được sử dụng cho các thiết bị IoT với băng thông thấp, độ tin cậy cao và khả năng được sử dụng trong mạng lưới không ổn định. Nó dựa trên một Broker (tạm dịch là “Máy chủ môi giới”) “nhẹ” (khá ít xử lý) và được thiết kế có tính mở (tức là không đặc trưng cho ứng dụng cụ thể nào), đơn giản và dễ cài đặt.

Ưu điểm: Một số ưu điểm nổi bật của MQTT như: băng thông thấp, độ tin cậy cao và có thể sử dụng ngay cả khi hệ thống mạng không ổn định, tốn rất ít byte cho việc kết nối với server và connection có thể giữ trạng thái open xuyên suốt, có thể kết nối nhiều thiết bị (MQTT client) thông qua một MQTT server (broker). Bởi vì giao thức này sử dụng băng thông thấp trong môi trường có độ trễ cao nên nó là một giao thức lý tưởng cho các ứng dụng IoT.

Mô hình Pub/Sub và Cơ chế hoạt động của MQTT:

Client:

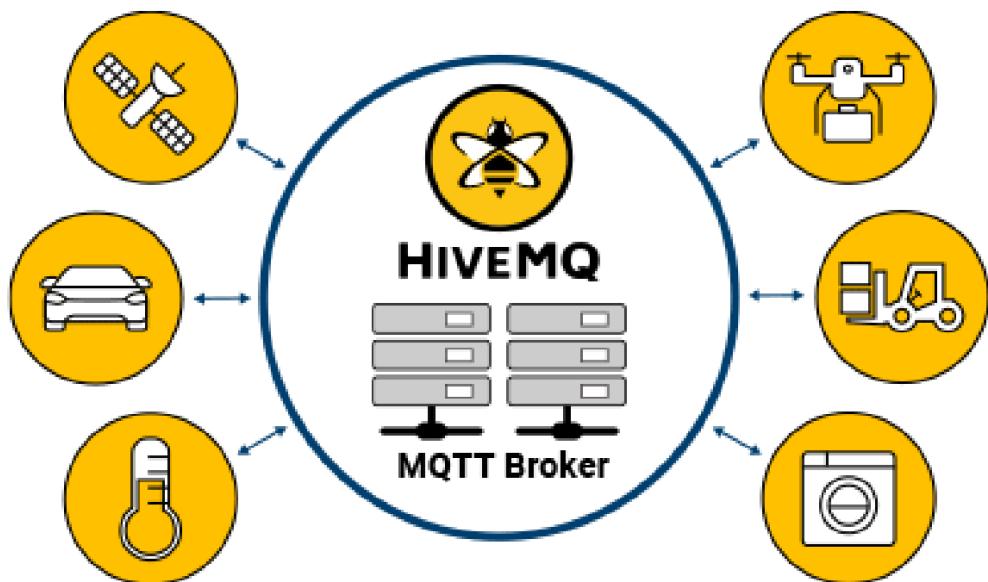
Publisher - Nơi gửi thông điệp

Subscriber - Nơi nhận thông điệp

Broker - Máy chủ môi giới:

Trong đó Broker được coi như trung tâm, nó là điểm giao của tất cả các kết nối đến từ Client (Publisher/Subscriber). Nhiệm vụ chính của Broker là nhận thông điệp (message) từ Publisher, xếp vào hàng đợi rồi chuyển đến một địa điểm cụ thể. Nhiệm vụ phụ của Broker là nó có thể đảm nhận thêm một vài tính năng liên quan tới quá trình truyền thông như: bảo mật message, lưu trữ message, logs,

Client thì được chia thành hai nhóm là Publisher và Subscriber. Client chỉ làm ít nhất một trong 2 việc là publish các thông điệp (message) lên một/nhiều topic cụ thể hoặc subscribe một/nhiều topic nào đó để nhận message từ topic này.



Hình 2.5 Cơ chế hoạt động cơ bản của MQTT

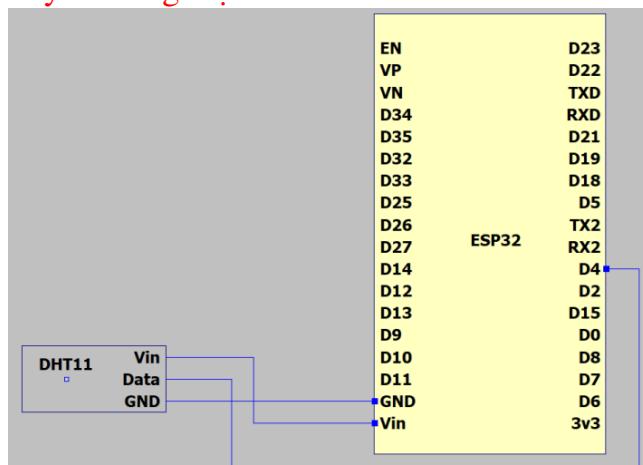
CHƯƠNG 3. TRIỂN KHAI

[2]

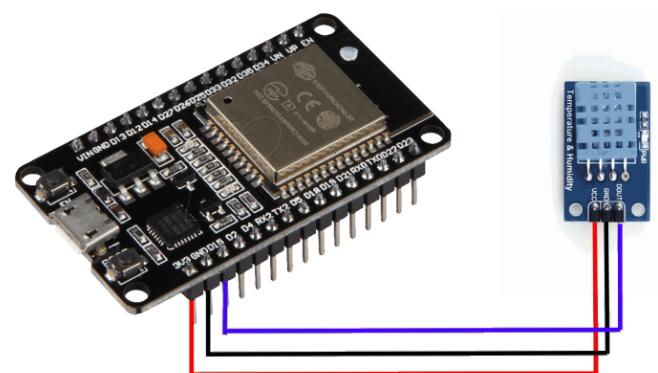
3.1 Khối cảm biến

3.1.1 Cảm biến nhiệt/ độ ẩm với ESP32

Layout bảng mạch



Layout mạch cảm biến nhiệt



Mạch thực tế

Lập trình với IDE

- Thư viện sử dụng: [DHTesp.h](#)
- Lập trình phần cảm biến:

```

bool getTemperature() {
    TempAndHumidity newValues = dht.getTempAndHumidity();
    if (dht.getStatus() != 0) {
        Serial.println("DHT11 error status: " + String(dht.getStatusString()));
        return false;
    }

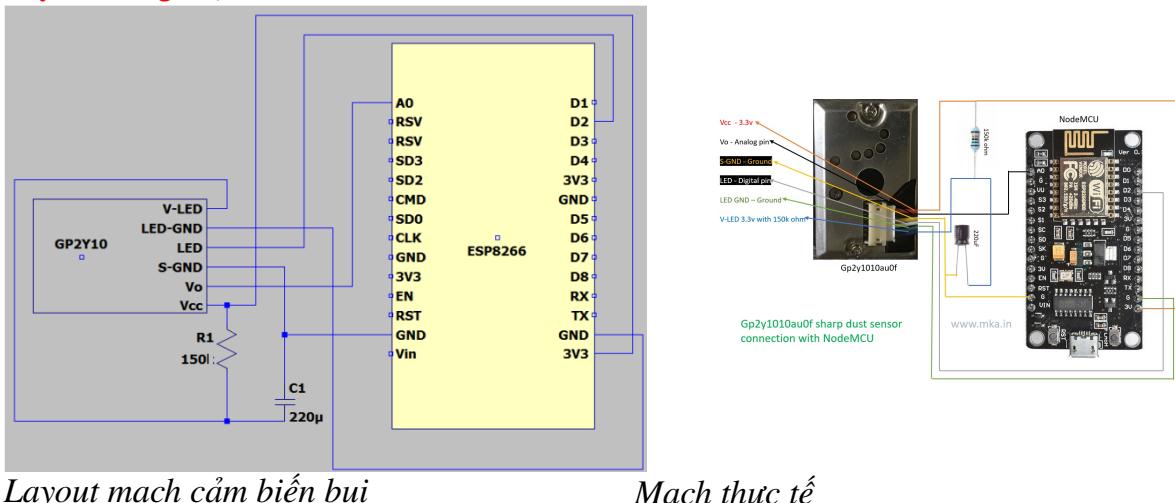
    float heatIndex = dht.computeHeatIndex(newValues.temperature, newValues.humidity);
    float dewPoint = dht.computeDewPoint(newValues.temperature, newValues.humidity);
    float cr = dht.getComfortRatio(cf, newValues.temperature, newValues.humidity);

    String comfortStatus;
    switch(cf) {
        case Comfort_OK:                     comfortStatus = "Comfort_OK";           break;
        case Comfort_TooHot:                 comfortStatus = "Comfort_TooHot";         break;
        case Comfort_TooCold:                comfortStatus = "Comfort_TooCold";        break;
        case Comfort_TooDry:                 comfortStatus = "Comfort_TooDry";         break;
        case Comfort_TooHumid:               comfortStatus = "Comfort_TooHumid";       break;
        case Comfort_HotAndHumid:            comfortStatus = "Comfort_HotAndHumid";     break;
        case Comfort_HotAndDry:              comfortStatus = "Comfort_HotAndDry";       break;
        case Comfort_ColdAndHumid:            comfortStatus = "Comfort_ColdAndHumid";     break;
        case Comfort_ColdAndDry:              comfortStatus = "Comfort_ColdAndDry";       break;
        default:                           comfortStatus = "Unknown:";                  break;
    };
    Serial.println(" T:" + String(newValues.temperature) + " H:" + String(newValues.humidity)
    + " I:" + String(heatIndex) + " D:" + String(dewPoint) + " " + comfortStatus);
    return true;
}

```

3.1.2 Cảm biến bụi với ESP8266

Layout bảng mạch



Layout mạch cảm biến bụi

Mạch thực tế

Lập trình với IDE

- Thư viện sử dụng: None- tính toán giá trị trực tiếp tại chân GPIO
- Lập trình phần cảm biến:

Theo datasheet, một lần đo của cảm biến sẽ mất khoảng 10ms để đo.

Trong 10ms sẽ có: 0.32ms: Trong khoảng thời gian này, IR LED sẽ được bật lên và chúng ta tiến hành đọc giá trị. Tuy nhiên, chỉ được phép đọc giá trị sau 0.28ms => cần làm trong 0.32ms này đó là: Bật IR LED. Delay 0.28ms. Tắt IR LED. Delay 0.04ms. 9.680ms: Thời gian này cảm biến sẽ không làm gì cả => chỉ cần delay 9.680ms. Vì vậy, ta có code như sau:

```

digitalWrite(ledPower, LOW); // Bật IR LED
delayMicroseconds(samplingTime); //Delay 0.28ms
voMeasured = analogRead(measurePin); // Đọc giá trị ADC V0
delayMicroseconds(deltaTime); //Delay 0.04ms
digitalWrite(ledPower, HIGH); // Tắt LED
delayMicroseconds(sleepTime); //Delay 9.68ms

// Tính điện áp từ giá trị ADC
calcVoltage = voMeasured * (5.0 / 1024); //Điện áp Vcc của cảm biến (5.0 hoặc 3.3)

// Linear Equation http://www.howmuchsnow.com/arduino/airquality/
// Chris Nafis (c) 2012
dustDensity = 0.17 * calcVoltage - 0.1;

```

3.2 Cài đặt wifi và gửi dữ liệu lên server

- Các thư viện được sử dụng: WiFi.h, PubSubClient.h, ArduinoJson.h

3.2.1 Cài đặt wifi

- Lập trình với IDE:

```
void setup_wifi() {  
  
    delay(10);  
    // We start by connecting to a WiFi network  
    Serial.println();  
    Serial.print("Connecting to ");  
    Serial.println(ssid);  
  
    WiFi.mode(WIFI_STA);  
    WiFi.begin(ssid, password);  
  
    while (WiFi.status() != WL_CONNECTED) {  
        delay(500);  
        Serial.print(".");  
    }  
  
    randomSeed(micros());  
  
    Serial.println("");  
    Serial.println("WiFi connected");  
    Serial.println("IP address: ");  
    Serial.println(WiFi.localIP());  
}
```

3.2.2 Gửi dữ liệu lên server

- Dữ liệu được gửi lên là dạng json-vì vậy cần 1 công đoạn chuyển từ dạng số qua dạng json để publish

- Lập trình với IDE:

```
// Serialize data to Json format  
StaticJsonDocument<1024> doc;  
char output[100];  
doc["dust"] = dustDensity;  
Serial.println("Read");  
serializeJson(doc, output);  
Serial.println(output);  
  
client.publish("home/esp8266", output); // publish to home/esp8266  
Serial.println("Sent");  
delay(5000);
```

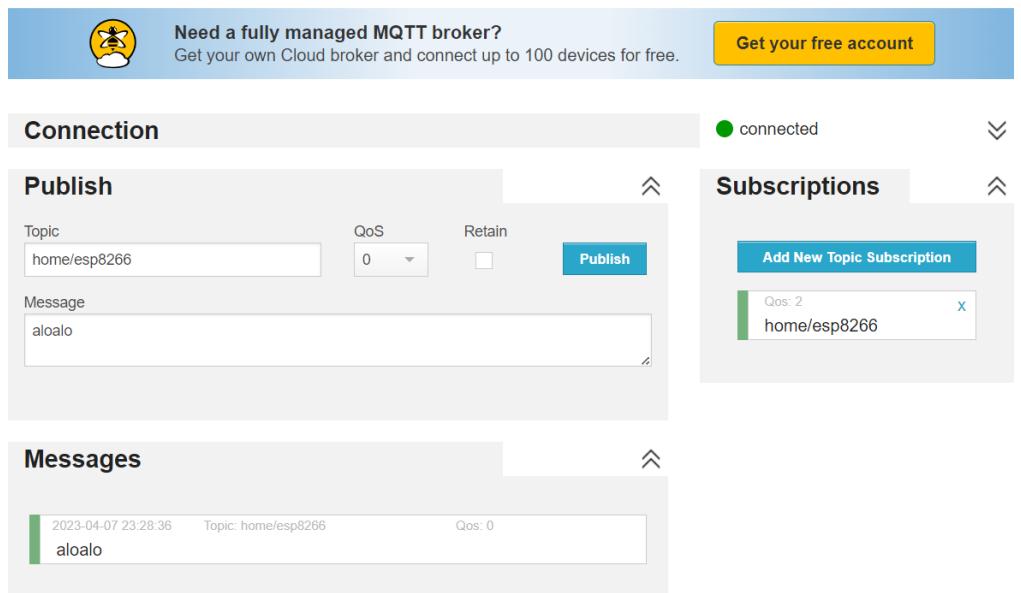
3.3 Khởi server

- Bài toán cảm biến không khí đặt ra yêu cầu cần sử dụng trong một không gian không quá lớn như nhà ở, phòng làm việc,... Chính vì thế các link online HiveMQ(MQTT Broker), Grafana(Front-end), InfluxDB(Database) được sử dụng, về Node-Red(Data bridge) thì tải trực tiếp trên Pi Zero
- Trước tiên ta update các phần mềm và OS của Raspberry Pi bằng lệnh “sudo apt update upgrade”
- Tiếp đến ta thực hiện tải Node-red:
bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)
- Để bắt đầu mở Node-Red, sử dụng lệnh node-red-start

3.3.1 MQTT Broker

Hivemq Broker là một nền tảng truyền nhận dữ liệu dựa trên giao thức MQTT, được thiết kế với đặc tính nhanh, hiệu quả, độ tin cậy cao khi truyền dữ liệu 2 chiều giữa các thiết bị Internet of Things.

Đây là một dịch vụ miễn phí trên nền tảng Cloud, giúp các bạn có thể kết nối MQTT ở bất cứ đâu, không kể địa lý, phù hợp khi học lập trình IOT

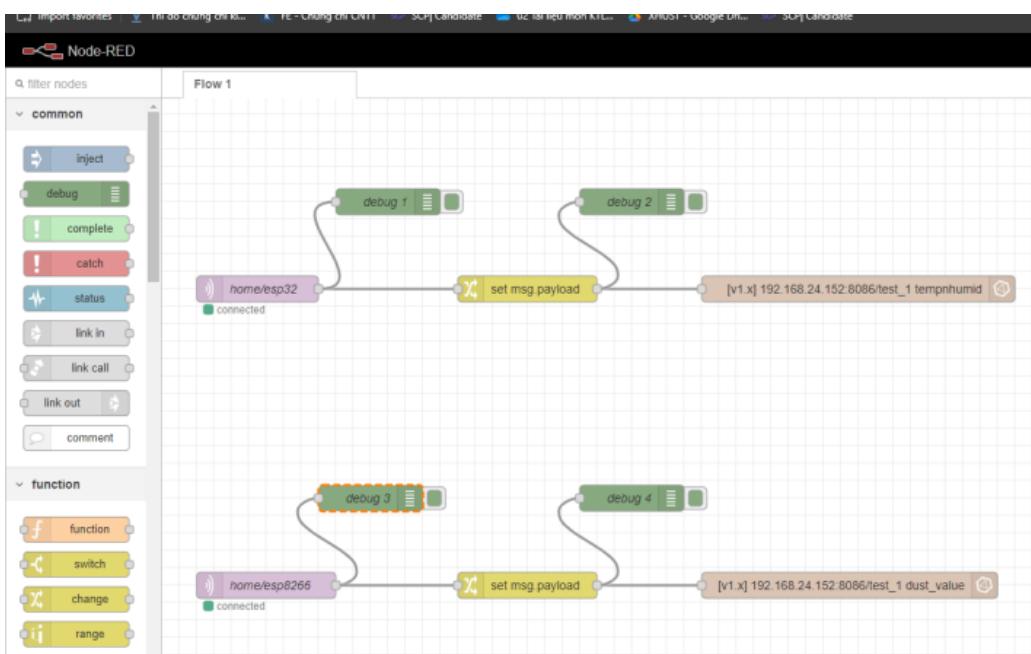


Hình 3.1 Broker với HiveMQ

3.3.2 Data Bridge

[3] - Node RED là một công cụ lập trình dùng để kết nối các thiết bị phần cứng, API và các dịch vụ trực tuyến với nhau. Về cơ bản, đây là một công cụ trực quan được thiết kế cho IoT (Internet of Things), nhưng cũng có thể được sử dụng cho các ứng dụng khác nhằm liên kết nhanh các luồng (flow) dịch vụ khác nhau.

- Node-RED cho phép người dùng kết hợp các dịch vụ Web và phần cứng bằng cách thay thế các tác vụ mã hóa cấp thấp phổ biến (như một dịch vụ đơn giản giao tiếp với một cổng nối tiếp) và điều này có thể được thực hiện với giao diện kéo thả trực quan. Các thành phần khác nhau trong Node-RED được kết nối với nhau để tạo ra một luồng (flow). Hầu hết mã lệnh (code) cần thiết được tạo tự động.

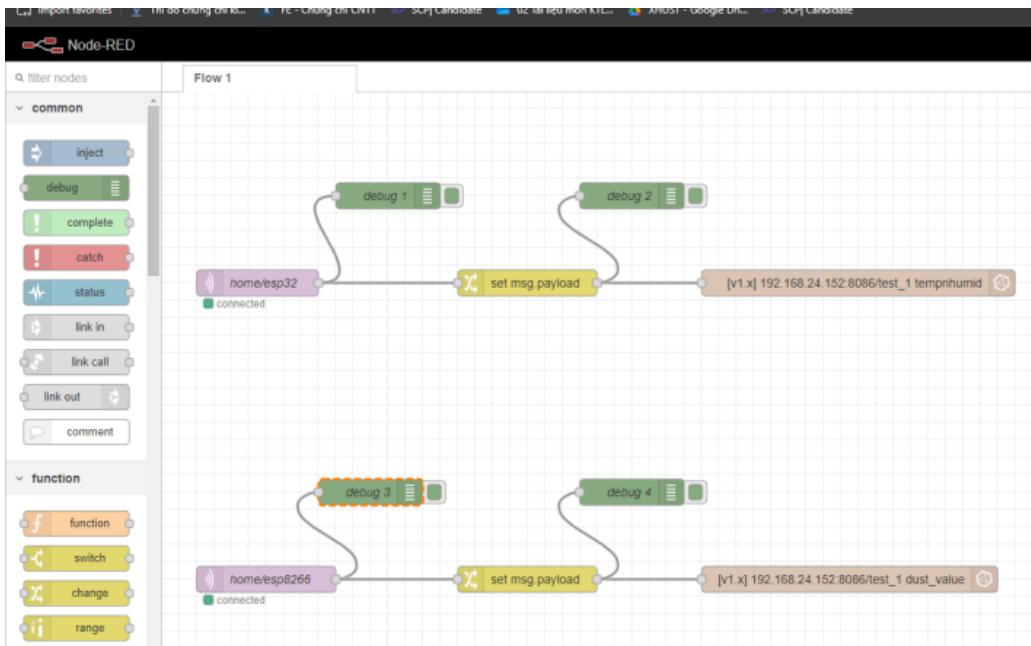


Hình 3.2 Node-red trên Pi

- Đầu tiên ta truy cập vào địa chỉ của database dưới dạng: địa chỉ hostserver:8086
- Sau đó ta set các node MQTT là home/esp32 và home/esp8266 để nhận được các message từ các publishers là các node cảm biến.
- Tiếp đến ta set các node set msg.payload để chuyển các gói message về thành dạng JSON file.
- Cuối cùng là các JSON file được chuyển về node database được tạo ra từ trước. Ta cũng cần đồng thời tạo ra một database mới trước khi chuyển dữ liệu vào và tạo các measurement ở đây là các thông số đo. Mục đích của việc này là để tạo được các measurement nhằm hiển thị trên giao diện front-end của Grafana.

3.3.3 Database

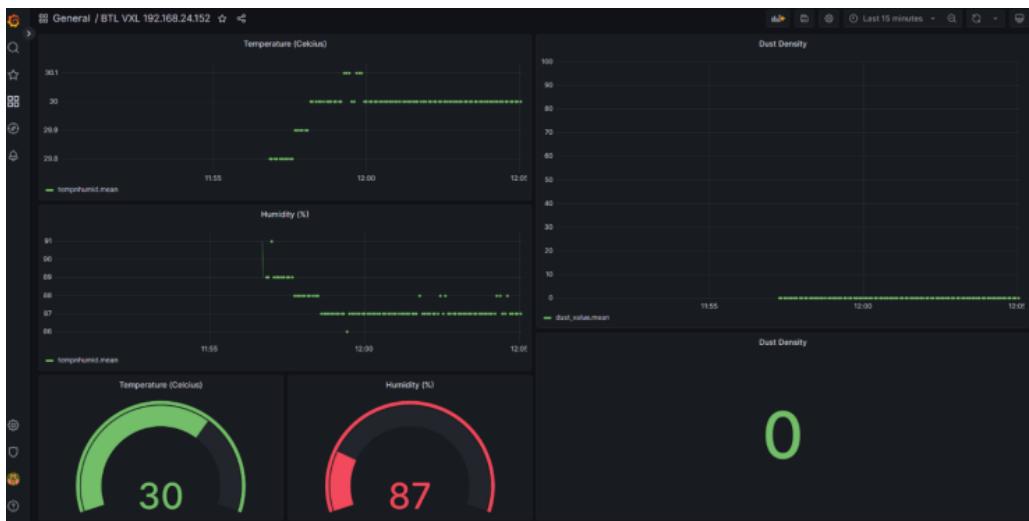
- InfluxDB là một database được tối ưu hóa để xử lý dữ liệu chuỗi thời gian (các dãy số được lập chỉ mục theo thời gian). InfluxDB được sử dụng để lưu các dữ liệu cho các trường hợp liên quan đến một lượng lớn time-stamped data, bao gồm DevOps monitoring, log data, application metrics, IoT sensor data, và real-time analytics. Nó có thể tự động xóa các dữ liệu cũ, không cần thiết và cung cấp một ngôn ngữ giống SQL để tương tác với dữ liệu.
- Influxdb có lợi thế lớn trong việc xử lý lượng time-stamped data lớn. Tuy vậy, giới hạn về kiểu dữ liệu là một hạn chế đối với Influxdb.



Hình 3.3 Node-red trên Pi

3.3.4 Front-end

- Grafana là một nền tảng open-source chuyên phục vụ mục đích theo dõi và đánh giá các số liệu thu được. Grafana hỗ trợ rất nhiều loại dashboard và các loại graph để người dùng dễ dàng theo dõi.
- Ưu điểm của Grafana là có thể truy xuất dữ liệu theo dòng thời gian và có khả năng hiển thị trên nhiều loại đồ thị và biểu đồ. Vì vậy nên người dùng có thể đưa ra thông tin và cái đặt cảnh báo dễ dàng trên Grafana.
- Với Grafana, chúng ta có thể tiếp cận với các thao tác dễ dàng:
- Cài đặt và đăng nhập vào Grafana
- Lựa chọn Datasource (trong bài tập lần này là InfluxDB)
- Tạo Dashboard để hiển thị dữ liệu từ database



Hình 3.4 Hiển thị dữ liệu bằng grafana

CHƯƠNG 4. KIỂM TRA-THỬ NGHIỆM

4.4 Kiểm tra chức năng của hệ thống

Đại lượng cần đo:

- Độ bụi (mg/m^3)
- Nhiệt độ ($^{\circ}C$)
- Độ ẩm(%)

Cách thiết lập hệ thống đo:

- Kết nối vi xử lý qua máy tính đang sử dụng phần mềm Arduino IDE.
- Thực hiện nối mạch như trên layout bảng mạch.
- Phát sóng wifi 2.4GHz
- Setup server và giao diện người dùng qua Node-RED và Grafana

Kiểm tra :

- Nạp code cho các vi xử lý được nêu trên
- Setup server trên Raspberry Pi
- Setup Node-RED và giao diện trên Grafana
- Kiểm tra thông tin đo được của cảm biến qua giao diện Grafana.

4.5 Kết quả đo được

Hệ thống kết nối hoạt động đúng theo yêu cầu đề ra, cảm biến đo bụi có thông số vẫn chưa chuẩn, có thể là do cảm biến bị hỏng; thông số cảm biến nhiệt độ và độ ẩm khá đúng, gần với mức nhiệt độ thực tế.



Hình 4.1 Kết quả cuối cùng

CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Mặc dù chỉ là project đơn giản không quá khó, tuy nhiên cũng không hề dễ để hiểu đối với sinh viên mới làm quen với hệ thống mạng/điện tử nói chung, bởi hầu như mọi thứ đều cần tìm hiểu và làm quen. Việc thực hiện project có thể coi là tạo dựng nền kiến thức nền tảng cơ bản về các giao thức mạng, cách hoạt động của Raspberry Pi, hệ điều hành Linux...

Sản phẩm đã đạt được tương đối chỉ tiêu được đặt ra, với chức năng cảm biến và gửi dữ liệu lên server, sau đó quản lý dữ liệu với database và hiển thị qua các biểu đồ dễ nhìn, dễ hiểu. Với khả năng hoạt động trên tốc độ cao, nhẹ, giá thành mạch phù hợp với sinh viên. Đây có thể là project phù hợp với sinh viên muốn tìm tòi, học hỏi...

Tương lai, hoàn toàn có thể cải thiện sản phẩm bằng việc thêm, cải thiện bảo mật cho sản phẩm, tăng thêm số lượng cảm biến, mở rộng chỉ tiêu sản phẩm bằng việc thêm cảm biến, từ đó giúp quản lý dữ liệu với các ứng dụng như: nhà thông minh, quản lý không khí... Ngoài ra, bởi hiện tại nền tảng database Influx Db được sử dụng là chỉ giới hạn free trong 30 ngày, vì vậy tương lai cần tự tạo ra database mới để sử dụng.

TÀI LIỆU THAM KHẢO

- [1] C. Bernstein, “Mqtt (mq telemetry transport),” [Online]. Available: <https://www.techtarget.com/iotagenda/definition/MQTT-MQ-Telemetry-Transport>.
- [2] L. E. Systems, “Easy raspberry pi iot server,” [Online] . Available : <https://randomnerdtutorials.com/how-to-install-mosquitto-broker-on-raspberry-pi/>, 2022.
- [3] I. C. Simplified, “Installing node red on raspberry pi zero w (32bit),” [Online]. Available: <https://www.youtube.com/watch?v=wYLO8wMOD7M>, 2023.