ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA

TIỂU LUẬN

ĐO LƯỜNG VÀ ĐIỀU KHIỂN BẰNG MÁY TÍNH

ĐỀ TÀI:

ỨNG DỤNG IOT SỬ DỤNG HIVEMQ COMMUNITY EDITION ĐỂ XÂY DỰNG HỆ THỐNG GIÁM SÁT NHIỆT ĐỘ VÀ ĐỘ ẨM CHO VƯỜN ƯƠM CÂY TRỒNG

Giảng viên hướng dẫn: TS. NGUYỄN TRỌNG TÀI

LỚP L04 --- NHÓM 29 --- HK222

Thành phố Hồ Chí Minh – 2023

BÁO CÁO PHÂN CÔNG NHIỆM VỤ VÀ KẾT QUẢ THỰC HIỆN

STT	MSSV	Họ và tên	Nhiệm vụ	Hoàn thành
1	2012294	Nguyễn Tấn Trình	Thiết kế, thi công phần cứng.	100%
2	2013693	Phan Thành Lộc	Thiết kế giao diện bằng Visual Studio.	100%
3	2013963	Lê Hữu Uy Nhân	Giao tiếp ESP32 với MQTT truyền nhận dữ liệu.	100%

MỤC LỤC

I. GIỚI THIỆU ĐỀ TÀI:

1. Lý do chọn đề tài:

Cùng với sự phát triển của khoa học công nghệ trong thời đại 4.0, đặc biệt là sự phát triển về mặt số hóa hay IOT (Internet of Things – Vạn vật kết nối). IOT có thể coi là một xu hướng hiện tại của tự động hóa và trao đổi dữ liệu. Nó là sự kết hợp giữa hệ thống mạng vật lý, mạng Internet kết nối vạn vật và điện toán đám mây. Hiện nay, IOT ngày càng phổ biến và được ứng dụng rộng rãi trong nhiều lĩnh vực của đời sống và sản xuất.

Việc ứng dụng IOT vào việc giám sát, điều khiển các thiết bị từ xa thông qua các thiết bị di động có thể xem là một ứng dụng phổ biến hiện nay. Đặc biệt, việc giám sát nhiệt độ, độ ẩm để duy trì trạng thái ổn định trong vườn ươm đảm bảo cho cây trồng phát triển ổn định, có thể xem là một ứng dụng mang lại lợi ích thiết thực điển hình.

Do vây, nhóm lựa chọn tìm hiểu và thực hiện đề tài ứng dụng IOT sử dụng Hivemq Community Edition để xây dựng hệ thống giám sát nhiệt độ và độ ẩm cho vườn ươm. Để có thể hiểu biết và ứng dụng được việc xây dựng một hệ thống IOT đơn giản thông qua việc kết nối với điện toán đám mây thông qua giao thức MQTT.

2. Nội dung đề tài:

Tìm hiểu các kiến thức liên quan và xây dựng một hệ thống giám sát nhiệt độ và độ ẩm cho vườn ươm với các tính năng như sau:

- Đọc dữ liệu từ các cảm biến, thực hiện cảnh báo nếu giá trị đọc được nằm ngoài ngưỡng an toàn.
- Truyền các dữ liệu đọc từ cảm biến lên Broker Hivemq thông qua giao thức MQTT để có thể giám sát từ các thiết bị.
- Giám sát dữ liệu đọc về từ cảm biến thông qua màn hình giao diện trên laptop.

Hệ thống dùng loại cảm biến nhiệt độ và độ ẩm để thu thập dữ liệu về nhiệt độ và nồng độ khí gas để so sánh với ngưỡng cho phép và sử dụng quạt và vòi để đảm bảo duy trì môi trường phù hợp.

II. THIẾT KẾ PHẦN CỨNG:

1. Danh mục linh kiện:

STT	Tên linh kiện	Giá thành
1	Module ESP-WROOM32 DevKit	170.000 vnđ
2	Module 2 relay 5V	32.000 vnđ
3	Cảm biến nhiệt độ DHT11	25.000 vnđ

1.1. Module ESP-WROOM32 DevKit:

a. ESP-WROOM32 DevKit là gì?

ESP32 là một series các vi điều khiển trên một vi mạch giá rẻ, năng lượng thấp:

- Sử dụng bộ vi xử lý Tensilica Xtensa LX6.
- Có tích hợp WIFI và dual-mode Bluetooth (Bluetooth chế độ kép).
- Bao gồm công tắc antenna tích hợp, bộ khuếch đại công suất, bộ khuếch đại thu nhiễu thấp, bộ lọc và module quản lý năng lượng.

ESP-WROOM32 là một module với nhiều tính năng cải tiến hơn các module dòng ESP8266

Hình 1.1 Module ESP-WROOM32 DevKit

b. Cấu hình của ESP-WROOM32 DevKit:

Bộ xử lý:

- CPU: Xtensa Dual-Core LX6 microprocessor.
- Chạy hệ 32 bit.
- Tốc độ xử lý từ 160 MHz đến 240 MHz.
- ROM: 448 Kb.
- Tốc độ xung nhịp từ 40 Mhz ÷ 80 Mhz (có thể tùy chỉnh khi lập trình).

- RAM: 520 Kb SRAM liền chip. Trong đó 8 Kb RAM RTC tốc độ cao – 8 Kb RAM RTC tốc độ thấp (dùng ở chế độ DeepSleep).

Giao tiếp không dây:

- Wi-Fi: 802.11 b/g/n/e/i
- Bluetooth: v4.2 BR/EDR và BLE

Giao tiếp có dây:

- 2 bộ DAC 8 bit
- 18 bộ ADC 12 bit
- 2 cổng giao tiếp I^2C
- 3 cổng giao tiếp UART
- 3 cổng giao tiếp SPI (1 cổng cho chip FLASH)
- 2 cổng giao tiếp I^2S
- 10 kênh ngõ ra điều chế độ rộng xung (PWM)
- SD card/SDIO/MMC host
- Ethernet MAC hỗ trợ chuẩn: DMA và IEEE 1588
- CAN bus 2.0
- IR (TX/RX)

Cảm biến tích hợp:

- 1 cảm biến Hall (cảm biến từ trường).
- 1 cảm biến đo nhiệt độ.
- Cảm biến chạm (điện dung) với 10 đầu vào khác nhau.

Bảo mật:

- Hỗ trợ tất cả các tính năng bảo mật chuẩn IEEE 802.11, bao gồm WFA, WPA/WPA2 và WAPI
- Khởi động an toàn (Secure boot).
- Mã hóa flash (Flash encryption).
- 1024-bit OTP, lên đến 768-bit cho khách hàng.
- Tăng tốc phần cứng mật mã: AES, SHA-2, RSA, mật mã đường cong elliptic (ECC – elliptic curve cryptography), bộ tạo số ngẫu nhiên (RNG – random number generator).

Nguồn điện hoạt động:

- Điện áp hoạt động: 2,2V ÷ 3,6V
- Nhiệt độ hoạt động: $-40^oC \div +85^oC$
- Số cổng GPIO: 36

c. Môi trường lập trình:

Lập trình ESP32 thông qua: ESP-IDF và VS Code (sử dụng ngôn ngữ C)

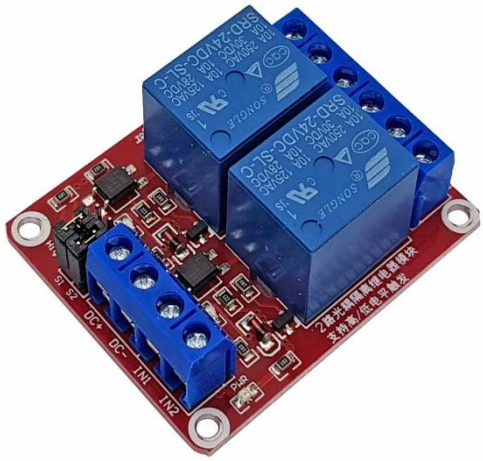
Với những ưu điểm:

- Có khả năng tự hoàn thiện code, gợi ý.
- Có thể xem được các thư viện include vào. Do đó, biết cách dùng thư viện đó như thế nào, và có thể mở ngay trong VS Code không phải dùng 1 trình soạn thảo khác để mở.
- Có thể tìm tới hàm gốc. Khi lập trình bạn quên mất việc truyền tham số nào vào thì với VS Code chỉ cần thao tác đơn giản là Ctr + Click, nó sẽ mở file chứa hàm gốc đó lên.

1.2. Module 2 relay 5V

a. Module 2 relay 5V là gì?

Module 2 relay 5V với opto cách ly kích H/L với opto cách ly nhỏ gọn, có opto và transistor cách ly giúp cho việc sử dụng trở nên an toàn với board mạch chính, mạch được sử dụng để đóng ngắt nguồn điện công suất cao AC hoặc DC, có thể chọn đóng khi kích mức cao hoặc mức thấp bằng Jumper.



Hình 1.2 Module 2 relay 5V

b. Thông số kỹ thuật:

Điện áp nuôi mạch: 5VDC.

Dòng tiêu thụ: khoảng 200mA/1Relay

Tín hiệu kích: High (5VDC) hoặc Low (0VDC) chọn bằng Jumper.

Relay trên mạch:

- Nguồn nuôi: 5VDC.
- Tiếp điểm đóng ngắt max: 250VAC-10A hoặc 30VDC-10A

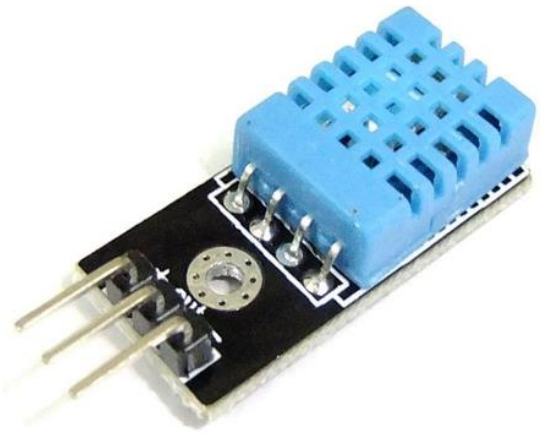
Kích thước: 52 (L) * 41(W) * 19 (H) mm.

1.3. Cảm biến nhiệt độ DHT11:

a. Cảm biến nhiệt độ DHT11 là gì?

DHT11 là một cảm biến kỹ thuật số giá rẻ để cảm nhận nhiệt độ và độ ẩm. Cảm biến này có thể dễ dàng giao tiếp với bất kỳ bộ vi điều khiển vi nào như Arduino, Raspberry Pi, ... để đo độ ẩm và nhiệt độ ngay lập tức.

DHT11 là một cảm biến độ ẩm tương đối. Để đo không khí xung quanh, cảm biến này sử dụng một điện trở nhiệt và một cảm biến độ ẩm điện dung.



Hình 1.3 Cảm biến nhiệt độ DHT11

b. Thông số kỹ thuật:

Điện áp hoạt động: 3V - 5V (DC)

Dòng sử dụng: 2.5mA max (khi truyền dữ liệu)

Dãi độ ẩm hoạt động: 20% - 90% RH, sai số ±5%RH

Dãi nhiệt độ hoạt động: 0°C ~ 50°C, sai số ±2°C

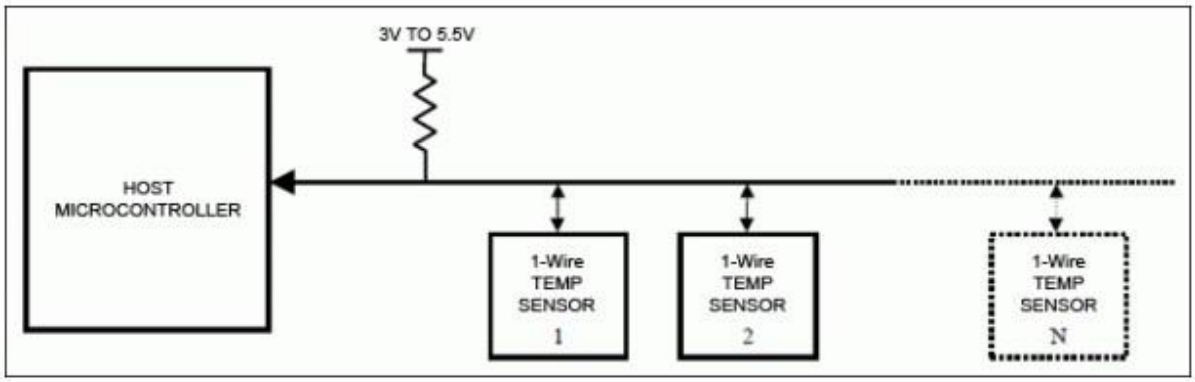
Tần số lấy mẫu tối đa: 1Hz (1 giây / lần)

Khoảng cách truyền tối đa: 20m

c. Giao tiếp One Wire (1 Wire):

Cảm biến DHT11 giao tiếp với vi điều khiển thông qua giao thức 1 Wire. Giao thức 1 Wire là một hệ thống bus giao tiếp với thiết bị, hỗ trợ truyền dữ liệu tốc độ thấp (16.3 kbit/s), truyền tín hiệu, và nguồn nuôi qua cùng một chân tín hiệu đơn.

Mô hình kết nối:



Hình 1.4 Sơ đồ kết nối của giao thức 1-Wire

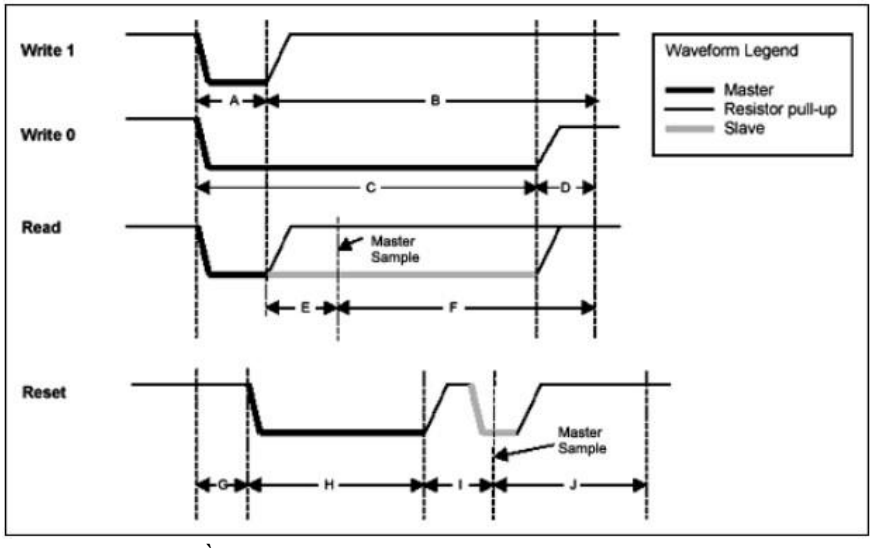
Phương thức hoạt động: có 4 phương thức thường sử dụng là: Reset, Write bit 0, Write bit 1, Read.

+ Reset: Chuẩn bị giao tiếp . Master cấu hình chân data là Ouput, kéo xuống 0 một khoảng H rồi nhả ra để trở treo kéo lên mức 1. Sau đó cấu hình Master là chân Input, delay I (us) rồi đọc giá trị slave trả về. Nếu = 0 thì cho phép giao tiếp. Nếu data = 1 đường truyền lỗi hoặc slave đang bận.

+ Write 1 : truyền đi bit 1 : Master kéo xuống 0 một khoảng A(us) rồi về mức 1 khoảng B (us).

+ Write 0 : truyền đi bit 0 : Master kéo xuống 0 khoảng C rồi trả về 1 khoảng D (us).

+ Read : Đọc một Bit : Master kéo xuống 0 khoảng A rồi trả về 1. delay khoảng E(us) rồi đọc giá trị slave gửi về. Các bạn phải bắt được tín hiệu trong khoảng F. Sau khoảng đó sẽ Slave sẽ nhả chân F về trạng thái 1 (IDLE – Rảnh rỗi).



Hình 1.5 Sơ đồ phương thức hoạt động của giao thức 1-Wire

Khung truyền của cảm biến DHT11: gồm 5 byte

+ Byte 1: giá trị phần nguyên của độ ẩm (RH%)

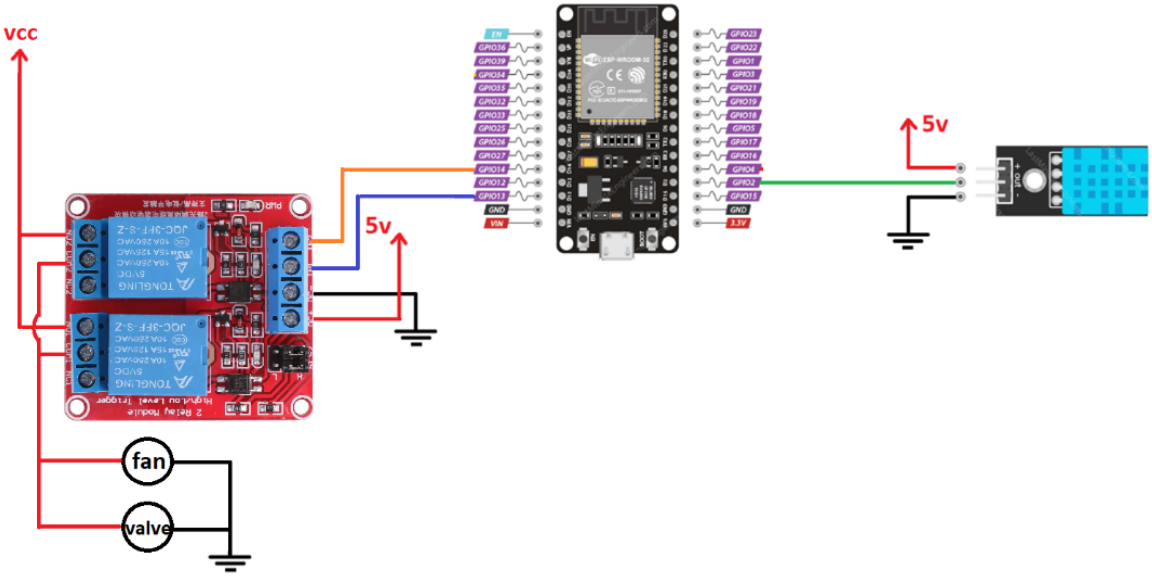
+ Byte 2: giá trị phần thập phân của độ ẩm (RH%)

+ Byte 3: giá trị phần nguyên của nhiệt độ (TC)

+ Byte 4 : giá trị phần thập phân của nhiệt độ (TC)

+ Byte 5 : kiểm tra tổng (Check Sum) là tổng của 4 byte phía trước cộng lại

2. Sơ đồ kết nối phần cứng:



Hình 1.6 Sơ đồ thiết kế phần cứng

Chân IN1 và IN2 của module relay lần lượt được nối vào chân GPIO14 và GPIO13 của module ESP32 để nhận tín hiệu điều khiển các thiết bị điện.

Ngõ ra cảm biến DHT11 được nối vào chân GPIO2 của module ESP32 để thu thập dữ liệu về nhiệt độ và độ ẩm.

III. THIẾT KẾ FIRMWARE:

1. Tổng quan về MQTT:

1.1. Giao thức truyền thông MQTT:

a. Khái niệm:

MQTT (Message Queuing Telemetry Transport) là giao thức truyền thông điệp (message) theo mô hình publish/subscribe (tạm dịch: xuất bản – đăng ký), 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.

Giao thức này hoạt động trên nền tảng TCP/IP, được thiết kế cho việc truyền tải dữ liệu đến các thiết bị ở xa hay vi điều khiển nhỏ có tài nguyên hạn chế.

b. Ưu nhược điểm của MQTT:

- **Ưu điểm:**

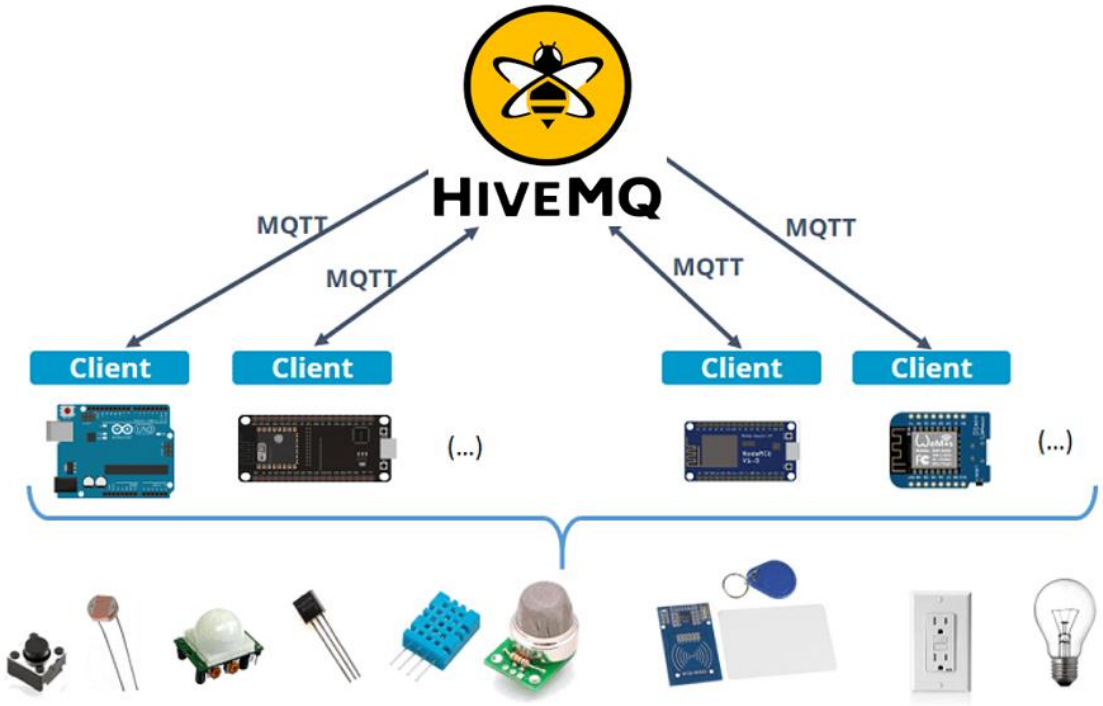
- Phân phối thông tin hiệu quả hơn.
- Tăng khả năng mở rộng.
- Giảm đáng kể tiêu thụ băng thông mạng.
- Giảm tốc độ cập nhật.
- Phù hợp cho việc điều khiển.
- Tối đa hóa băng thông đang sử dụng.
- Chi phí đầu tư cực kỳ thấp.
- Rất an toàn và bảo mật.
- Giao thức publish/subscribe thu thập nhiều dữ liệu hơn và tốn ít băng thông hơn so với giao thức cũ.

- **Nhược điểm:**

- MQTT có chu kỳ truyền chậm hơn tương đối so với CoAP.
- Tài nguyên của MQTT hoạt động dựa trên subscribe động. Còn CoAP sẽ sử dụng hệ thống tài nguyên tĩnh, có tính ổn định hơn.
- MQTT không được mã hóa. Nó chỉ sử dụng TLS/SSL để mã hóa bảo mật.
- MQTT là giao thức truyền thông nhưng khó để tạo ra được mạng mở rộng toàn cầu.

c. Vị trí của MQTT trong mô hình IOT:

Dữ liệu sau khi được cảm biến thu thập chuyển về vi điều khiển / vi xử lý thí lúc này MQTT đóng vai trò như một cầu nối để truyền những dữ liệu đó lên MQTT Broker và được lưu trữ để sử dụng cho các mục đích sau này.



Hình 2.1 Vị trí của MQTT trong hệ thống IoT

d. Tính năng và đặc điểm nổi bật:

Dạng truyền thông điệp theo mô hình Pub/Sub cung cấp việc truyền tin phân tán một chiều, tách biệt với phần ứng dụng.

Việc truyền thông điệp là ngay lập tức, không quan tâm đến nội dung được truyền.

Sử dụng TCP/IP là giao thức nền.

Tồn tại ba mức độ tin cậy cho việc truyền dữ liệu (QoS: Quality of service)

- QoS 0: Broker/client sẽ gửi dữ liệu đúng một lần, quá trình gửi được xác nhận bởi chỉ giao thức TCP/IP.
- QoS 1: Broker/client sẽ gửi dữ liệu với ít nhất một lần xác nhận từ đầu kia, nghĩa là có thể có nhiều hơn 1 lần xác nhận đã nhận được dữ liệu.
- QoS 2: Broker/client đảm bảo khi gửi dữ liệu thì phía nhận chỉ nhận được đúng một lần, quá trình này phải trải qua 4 bước bắt tay.

Phần bao bọc dữ liệu truyền nhỏ và được giảm đến mức tối thiểu để giảm tải cho đường truyền.

1.2. Cơ chế hoạt động của MQTT:

a. MQTT Broker:

MQTT Broker hay máy chủ mô giới 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, ….

MQTT Broker được cung cấp dưới dạng mã nguồn mở hoặc các phiên bản thương mại giúp người dùng có thể tự cài đặt và tạo broker riêng. Ngoài ra các bạn cũng có thể sử dụng Broker trên điện toán đám mây với các nền tảng IOT như hive broker, amazone,….

b. MQTT Client:

MQTT Client là các thiết bị / ứng dụng kết nối đến Broker để thực hiện truyền nhận dữ liệu .Client thì được chia thành hai nhóm là Publisher và Subscriber. Một Client có thể có 1 trong 2 nhiệm vụ hoặc cả 2.

Publisher là thiết bị gửi bản tin lên broker

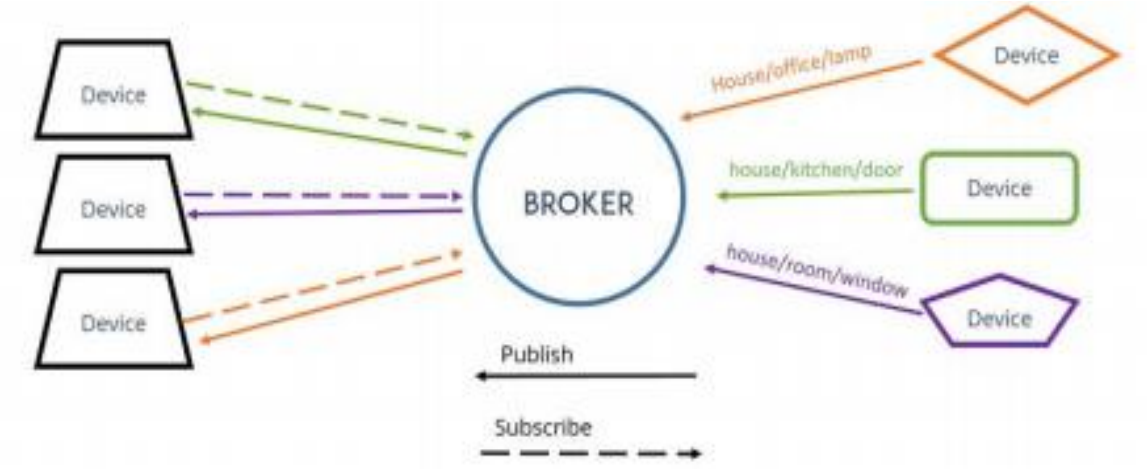
Subscriber là người nhận bản tin mỗi khi có bản tin mới gửi lên Broker.

c. Mô hình publish/subscribe:

Mô hình này gồm có Broker và Client đóng vai trò tương ứng là Publisher và Subscriber.

Broker sẽ là trung tâm nhận thông điệp từ Publisher xếp theo thứ tự và gửi đến địa điểm cụ thể.

Client chỉ làm 1 trong 2 việc là publish các thông điệp lên topic cụ thể hoặc là subscribe topic để nhận thông điệp.



Hình 2.2 Mô hình publish / subscribe

d. Cơ chế hoạt động của giao thức MQTT:

Một phiên MQTT được chia thành bốn giai đoạn: kết nối, xác thực, giao tiếp và kết thúc.

Giai đoạn 1: Kết nối

Client (máy khách) tạo kết nối Transmission Control Protocol / Internet Protocol (TCP/IP) tới broker bằng cách sử dụng cổng tiêu chuẩn hoặc cổng tùy chỉnh được xác định bởi các nhà phát triển broker.

Giai đoạn 2: Xác thực

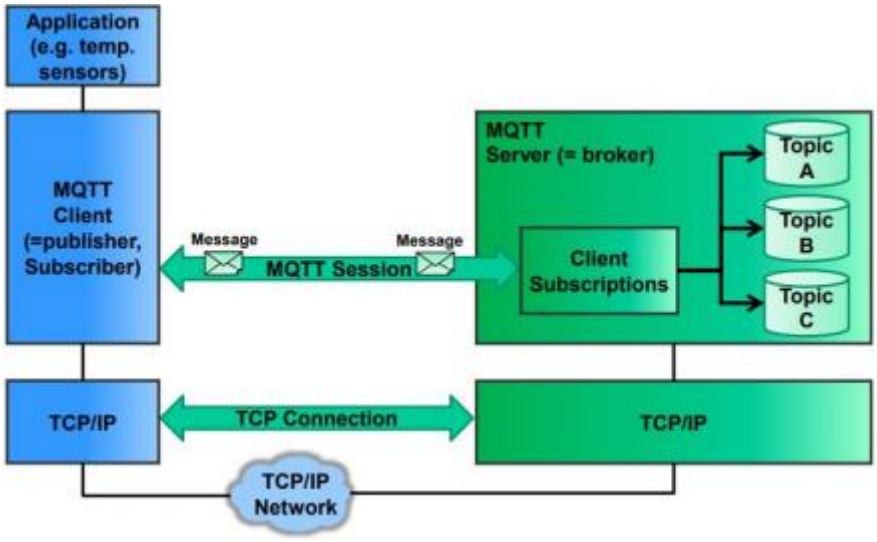
Các cổng tiêu chuẩn là port 1883 cho giao tiếp không mã hóa và port 8883 cho giao tiếp được mã hóa – sử dụng lớp cổng bảo mật (SSL) / bảo mật lớp truyền tải (TLS). Trong quá trình giao tiếp SSL/TLS, máy khách cần kiểm chứng và xác thực máy chủ.

Giai đoạn 3: Giao tiếp

Client sẽ gửi bản tin lên broker nếu là Publisher hoặc nhận bản tin từ broker về nếu là Subscriber. Quá trình kết nối này sẽ được giữ đến khi kết thúc kết nối.

Giai đoạn 4: Kết thúc

Sau khi kết thúc để có thể truyền nhận MQTT, tiếp tục quay lại các bước trên.



Hình 2.3 Cơ chế hoạt động của giao thức MQTT

2. Chương trình firmware:

2.1. Thư viện kết nối MQTT trên Arduino IDE:

```
//-------Khai bao cac bien dieu khien cho he thong-----
float temperature = 0, humidity = 0;
```

```
float setTemp = 40.0, setHumid = 60.0;
int operation_mode = 1, valveStatus = 0, fanStatus = 0;

//mqtt broker config
char broker[] = "broker.hivemq.com";
int port = 1883;

//wifi config
char ssid[] = "my wifi";
char password[] = "123456788";

//client sub topics
char sub_topic[] = "/control-dkmt";

//client pub topic
char pub_topic[] = "/data-dkmt";

int serialBaudrate = 9600;

//----------Khai bao doi tuong MQTT Client và WifiClient
WiFiClient net;
MQTTClient client;
auto timer = timer_create_default();
DHT dht(DHTPIN, DHTTYPE);
```

2.2. Lập trình kết nối của ESP32 với Broker HIVEmq:

Thực hiện cấu hình kết nối trong hàm setup.

```
void setup()
{
   pinMode(valvePin, OUTPUT);
   pinMode(fanPin, OUTPUT);
   dht.begin();

   Serial.begin(serialBaudrate);
   Serial.print("Connecting to ");
   Serial.println(ssid);
   WiFi.begin(ssid, password);

   while (WiFi.status() != WL_CONNECTED)
   {
      delay(500);
      Serial.print(".");
    }
   Serial.println("");
   Serial.println("WiFi connected");
   Serial.println("IP address: ");
   Serial.println(WiFi.localIP());

   client.begin(broker, port, net);
   client.onMessage(messageReceived);
   connect();
   timer.every(2000, sendData);
```

```
    timer.every(500, readDHT);
}
```

Hàm connect thực hiện kết nối với MQTT Broker:

```
void connect()
{
    while (!client.connected())
    {
        Serial.print("Attempting MQTT connection...");
        String clientId = "ESP_dkmtBTL";
        clientId += String(random(0xffff), HEX);
        if (client.connect(clientId.c_str()))
        {
            Serial.println("connected \n");
            Serial.println(clientId);
            client.subscribe(sub_topic);
        }
        else
        {
            Serial.print("failed, rc=");
            // Serial.print(client.state());
            Serial.println(" try again in 2 seconds");
            delay(2000);
        }
    }
    Serial.println("\nConnected!");
}
```

Hàm sendData thực hiện nhiệm vụ gửi các giá trị thông số hệ thống đến broker theo định dạng JSON.

```
bool sendData(void *)
{
    char value[160] = "";
    char temperature_string[50];
    char humidity_string[50];
    char mode_string[20];
    char fan_string[20];
    char valvePin_string[20];

    sprintf(temperature_string, "\"temperature\":\"%0.2f\",", temperature);
    sprintf(humidity_string, "\"concentration\":\"%0.2f\",", concentration);
    sprintf(mode_string, "\"mode\":\"%d\",", operation_mode);
    sprintf(fan_string, "\"fan\":\"%d\",", fanStatus);
    sprintf(valvePin_string, "\"valve\":\"%d\"", valveStatus);

    strcat(value, "{");
    strcat(value, temperature_string);
    strcat(value, humidity_string);
    strcat(value, mode_string);
    strcat(value, fan_string);
    strcat(value, valvePin_string);
    strcat(value, "}");
```

```
    client.publish(pub_topic, value);
    Serial.println(value);
    return true;
}
```

2.3. Lập trình nhận dữ liệu từ cảm biến về ESP32:

Đọc dữ liệu từ cảm biến nhiệt độ DHT11:

```
bool readDHT(void *)
{
    float t = dht.readHumidity();
    float t = dht.readTemperature();
    float f = dht.readTemperature(true);
    if (isnan(t))
    {
        temperature = 0;
    }
    temperature = t;
    humidity = h;
    return true;
}
```

2.4. Lập trình điều khiển các thiết bị cảnh báo:

Điều khiển các thiết bị điện: khi nhiệt độ lớn hơn setTemp thì bật quạt; khi độ ẩm lớn hơn setHumid thì bật valve phun sương tạo độ ẩm:

```
void control()
{
    if (humidity > setHumid)
    {
        digitalWrite(valvePin, HIGH);
        valveStatus = 1;
    }
    else if (temperature > setTemp)
    {
        digitalWrite(fanPin, HIGH);
        fanStatus = 1;
    }
    else
    {
        digitalWrite(valvePin, LOW);
        digitalWrite(fanPin, LOW);
        valveStatus = 0;
        fanStatus = 0;
    }
}
```

Hàm xử lý khi nhận được tín hiệu:

```
void messageReceived(String &topic, String &payload)
{
    Serial.println(payload);
```

```
    DynamicJsonDocument control(256);
    deserializeJson(control, payload);
    int control_mode = control["mode"];
    operation_mode = control_mode;
    if (operation_mode == 0)
    {
      int valve = control["valve"];
      int fan = control["fan"];
      if (valve == 1)
      {
        digitalWrite(valvePin, HIGH);
        buzzStatus = 1;
        Serial.println("valve on");
      }
      else
      {
        digitalWrite(valvePin, LOW);
        buzzStatus = 0;
        Serial.println("valve off");
      }

      if (fan == 1)
      {
        digitalWrite(fanPin, HIGH);
        fanStatus = 1;
        Serial.println("fan on");
      }
      else
      {
        digitalWrite(fanPin, LOW);
        fanStatus = 0;
        Serial.println("fan off");
      }
    }
}
```

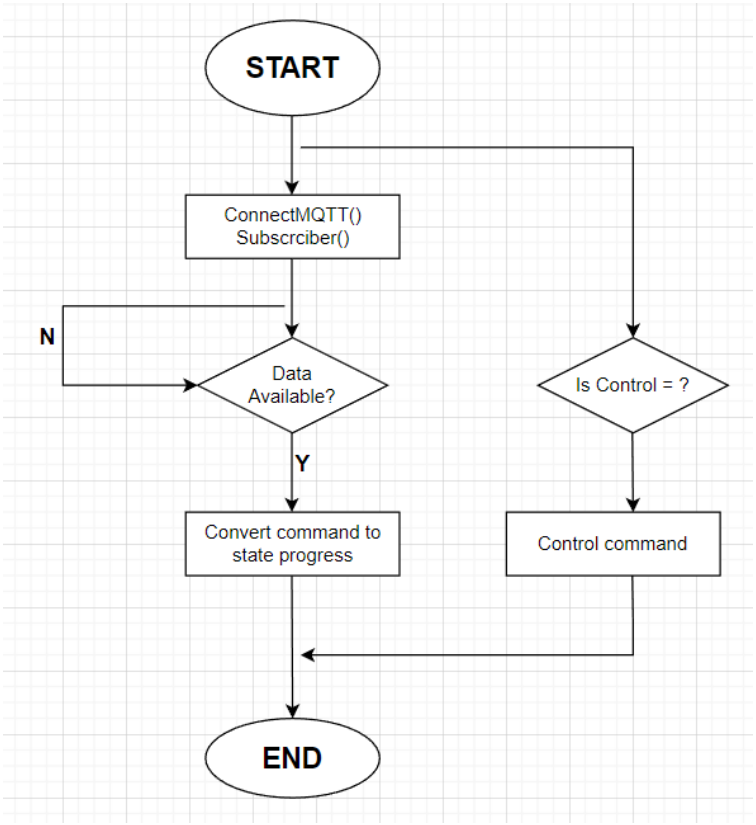
Gọi phương thúc loop trong hàm loop của chương trình chính

```
void loop()
{
  timer.tick();
  client.loop();
  delay(10);

  if (!client.connected())
  {
    connect();
  }

  if (operation_mode == 1)
  {
    control();
```

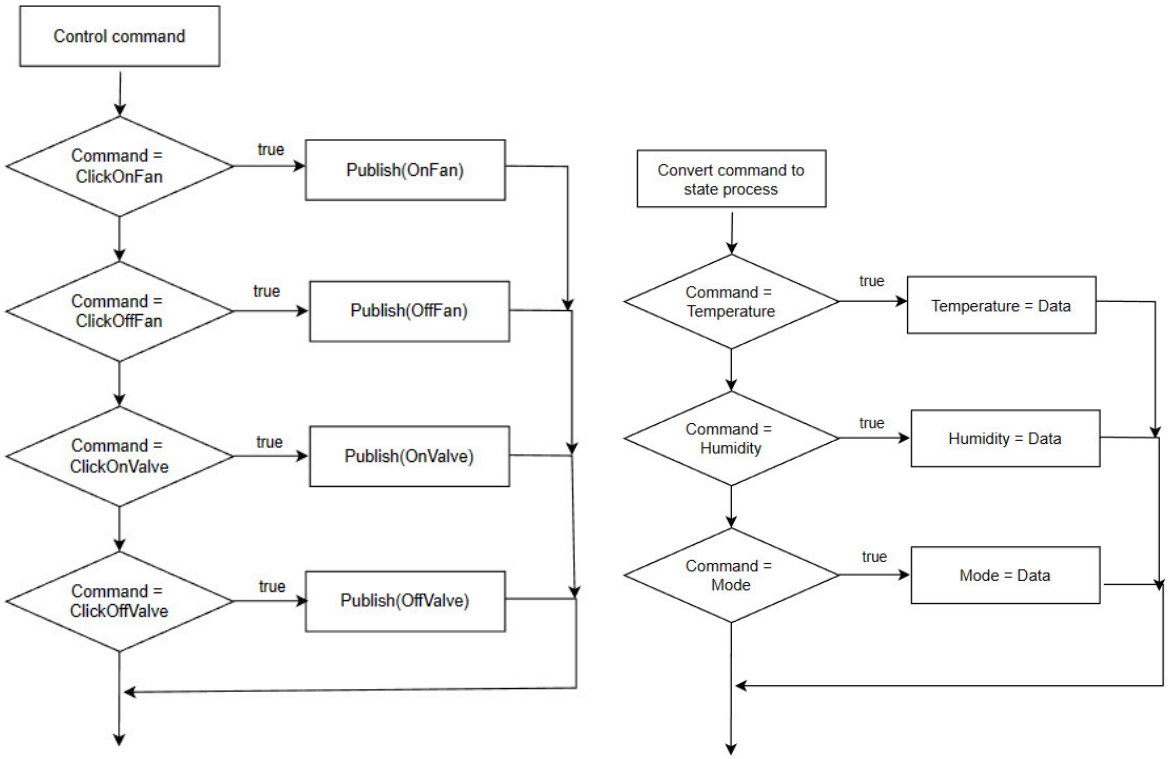
```
        }
    }
```

IV. THIẾT KẾ PHẦN MỀM:

1. Giải thuật thực hiện:



Hình 3.1 Lưu đồ giải thuật chung



Hình 3.2 Lưu đồ giải thuật truyền / nhận dữ liệu

2. Xây dựng giao diện giám sát:

2.1. Tổng quan giao diện:

Hình 3.3 Giao diện quan sát và điều khiển

Thông tin giám sát trên giao diện:

- Chế độ được lựa chọn hiện tại.
- Nhiệt độ hiện tại.
- Độ ẩm không khí hiện tại.

Thông tin điều khiển từ giao diện:

- Điều khiển quạt.
- Điều khiển vòi nước.

2.2. Chương trình xây dựng giao diện:

a. Các thư viện hỗ trợ:

- **Thư viện GunaUI:**

Với Guna2.UI, có thể tạo ra các giao diện người dùng chuyên nghiệp, hiện đại và dễ dàng tùy chỉnh. Bất kể bạn đang phát triển ứng dụng desktop nào, Guna2.UI sẽ là một công cụ hữu ích để nâng cao trải nghiệm người dùng và tăng tính thẩm mỹ của ứng dụng.

Chi tiết về thư viện tại: Thư viện GunaUI

- **Thư viện mqttnet:**

MQTTnet là một thư viện mã nguồn mở miễn phí thực thi giao thức MQTT cho các ứng dụng .NET hoạt động đa nền tảng (Windows, Linux, macOS).

Thư viện MQTTnet cung cấp các tính năng như:

- Quản lý kết nối và đăng ký với máy chủ MQTT.
- Gửi và nhận các tin nhắn MQTT với các cấp độ QoS khác nhau.
- Hỗ trợ mã hóa TLS/SSL và xác thực người dùng.
- Hỗ trợ các tính năng mới của MQTT 5.0, như thuộc tính tin nhắn, chủ đề được chia sẻ và quyền truy cập chủ đề.
- Hỗ trợ các giao diện lập trình ứng dụng (API) đồng bộ và không đồng bộ.
- Hỗ trợ các phiên bản MQTT 3.1, 3.1.1 và 5.0.
- Cung cấp các gói mở rộng hỗ trợ thêm cho MQTT, như Managed Client (với các tính năng nâng cao), RPC (gọi hàm từ xa).

Chi tiết về thư viện tại: Thư viện mqttnet

- **Thư viện Newtonsoft.Json:**

Bộ thư viện này cho phép chúng ta chuyển đổi một object của C# thành một chuỗi ký tự định dạng theo quy ước của JSON (JavaScript Object Notation) cũng như chuyển đổi ngược chuỗi JSON về object của C#. Đây là một trong những bộ thư viện có lượt download lớn nhất trên NuGet.

Chi tiết về thư viện tại: Thư viện Newtonsoft

b. Xây dựng giao diện giám sát:

- **Kết nối MQTT Broker:**

```
string brokerHostname = "broker.hivemq.com";
int brokerPort = 1883;

var options = new MqttClientOptionsBuilder()
    .WithTcpServer(brokerHostname, brokerport)
    .Build();
mqttClient = mqttFactory.CreateManagedMqttClient();
mqttClient.ConnectedHandler = new MqttClientConnectedHandlerDelegate (OnConnected);
mqttClient.ApplicationMessageReceivedHandler = new
MqttApplicationMessageReceivedHandlerDelegate (OnMessageReceived);

await mqttClient.StartAsync(new ManagedMqttClientOptionsBuilder()
    .WithAutoReconnectDelay(TimeSpan. FromSeconds(5))
```

```
    .WithClientOptions(options)
    .Build());


private void OnConnected(MqttClientConnectedEventArgs eventArgs)
{
// khi kết nối thành công, đăng ký đọc dữ liệu từ một topic trong HiveMQ string topic = "/data-dkmt";
mqttClient.SubscribeAsync(topic);
}
```

- **Nhận dữ liệu và xử lý cho giao diện:**

```
private void OnMessageReceived(MqttApplicationMessageReceivedEventArgs eventArgs)
      {
         // Khi nhận được tin nhắn từ HiveMQ, hiển thị nội dung lên TextBox
         string message = Encoding.UTF8.GetString(eventArgs.ApplicationMessage.Payload);

         DataReiceive dataReiceive = JsonConvert.DeserializeObject<DataReiceive>(message);
        // textBox1.Invoke(new Action(() => textBox1.Text = dataReiceive.mode));
         guna2CircleButton1.Invoke(new Action(() => guna2CircleButton1.Text = dataReiceive.temperature + "°C"          ));
         guna2CircleProgressBar1.Invoke(new Action(()=> guna2CircleProgressBar1.Value = (int)double.Parse(dataReiceive.temperature)));

         guna2CircleButton2.Invoke(new Action(() => guna2CircleButton2.Text = dataReiceive.concentration + "%"));
         guna2CircleProgressBar2.Invoke(new Action(() => guna2CircleProgressBar2.Value = (int)double.Parse(dataReiceive.concentration)));

         guna2ToggleSwitch3.Invoke(new Action(() => {
            if (dataReiceive.mode == "1")
            {
               guna2ToggleSwitch3.Checked = true;
               label6.Text = "Auto";
               StatusMode = "1";
            }
            else
            {
               guna2ToggleSwitch3.Checked = false;
               label6.Text = "Manual";
               StatusMode = "0";
            }
            }));

         guna2ToggleSwitch1.Invoke(new Action(() => {
            if (dataReiceive.valve =="1")
            {
               guna2ToggleSwitch1.Checked = true;
               label2.Text = "On";
               guna2Panel3.FillColor = Color.Blue;
```

```
                StatusValve = "1";

            }
            else
            {
                guna2ToggleSwitch1.Checked = false;
                label2.Text = "Off";
                guna2Panel3.FillColor = Color.Silver;
                StatusValve = "0";
            }
        }
            ));
        guna2ToggleSwitch2.Invoke(new Action(() =>
        {
            if (dataReiceive.fan =="1")
            {
                guna2ToggleSwitch2.Checked = true;
                label5.Text = "On";
                guna2Panel4.FillColor = Color.Blue;
                StatusFan = "1";

            }
            else
            {
                guna2ToggleSwitch2.Checked = false;
                label5.Text = "Off";
                guna2Panel4.FillColor = Color.Silver;
                StatusFan = "0";

            }
        }
            ));
    }
```

- **Khi nhấn nút điều khiển và truyền dữ liệu đi:**

```
private async void OnPublishData()
{
    if (mqttClient != null && mqttClient.IsConnected)
    {
        string topic="/control-dkmt";
        string message = $"{{\"fan\":\"{StatusFan}\",
\"speak\":\"{StatusValve}\",\"mode\":\"{StatusMode}\"}}";

        var mqttMessage = new MqttApplicationMessageBuilder()
            .WithTopic (topic)
            .WithPayload(message)
            .WithExactlyOnceQoS()
```

```
            .WithRetainFlag()
            .Build();
        await mqttclient. PublishAsync(mqttMessage);
    }
}
private void guna2ToggleSwitch3_Click(object sender, EventArgs e)
{
    if (guna2ToggleSwitch3. Checked)
    {
        label6.Text = "Auto";
        StatusMode = "1";
    }
    else
    {
        label6.Text = "Manual";
        StatusMode = "0";
    }
    OnPublishData();
}
```

Khi chưa chuyển chế độ Manual sẽ không cho gửi dữ liệu:

```
private void guna2ToggleSwitch2_Click(object sender, EventArgs e)
{
    if (guna2ToggleSwitch3. Checked)
    {
        guna2ToggleSwitch2. Checked = false;
        System.Windows.MessageBox.Show("Vui lòng chuyển sang chế độ Manual để sử dụng
chức năng này!", "Thông báo", MessageBoxButton.OKCancel, MessageBoxImage. Warning);
    else
    {
        if (guna2ToggleSwitch2.Checked)
        {
            labels.Text = "On";
            guna2Panel4.FillColor = Color.Blue;
            StatusFan = "1";
        }
        else
        {
            label5.Text = "Off";
```

```
            guna2Panel4.FillColor = Color.Silver;
            StatusFan = "0";
        }
    OnPublishData();
    }
}
```

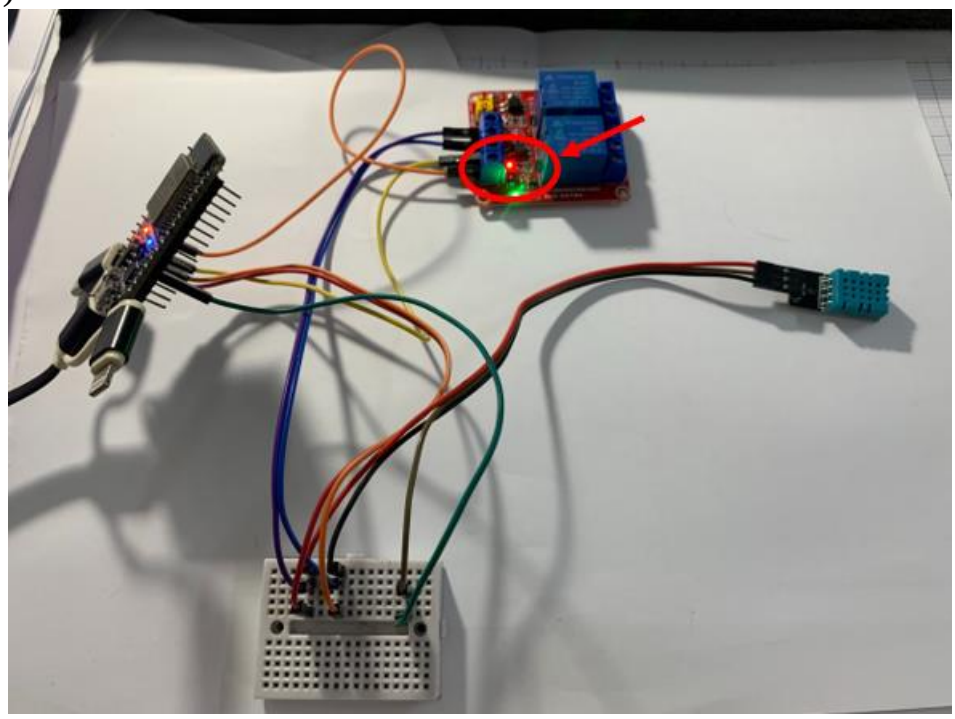
V. KẾT QUẢ THỰC HIỆN:

1. Phần cứng:

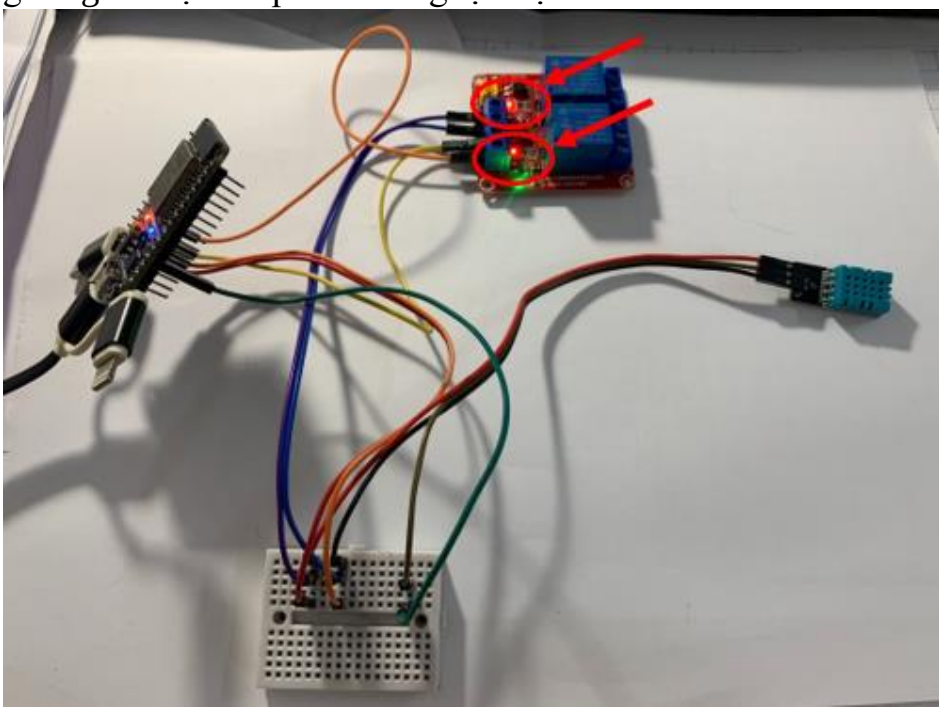
- **Kết quả phần cứng:**

Bật quạt khi có tín hiệu điều khiển từ xa:

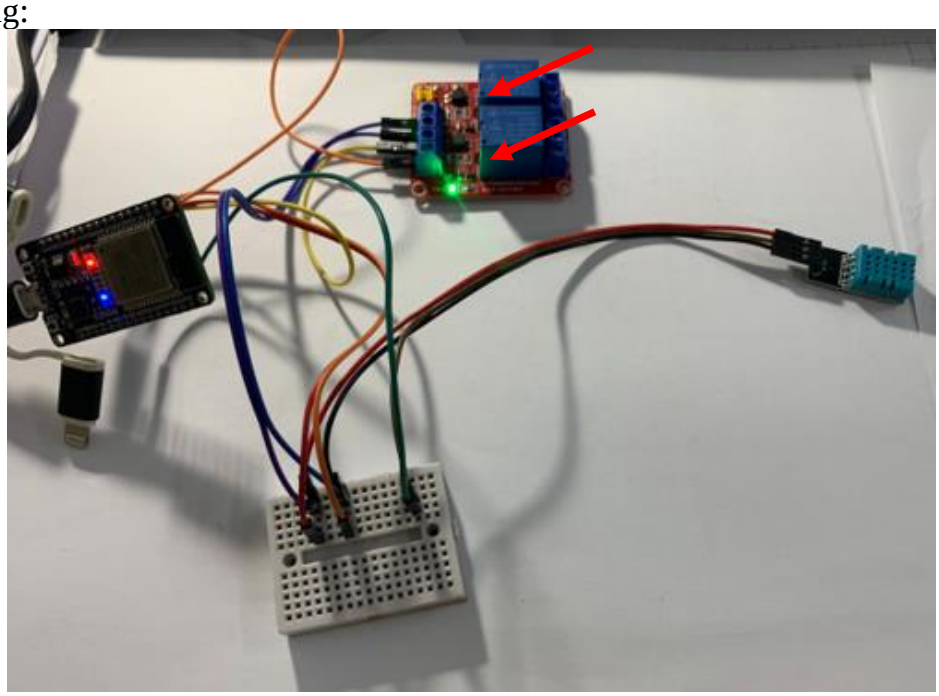
(Đèn led đỏ trên module relay báo hiệu xuất tín hiệu điều khiển vòi phun nước)



Ở chế độ auto: Khi nhiệt độ cao hơn ngưỡng => bật quạt và khi độ ẩm thấp hơn ngưỡng => bật vòi phun sương tạo độ ẩm.



Tắt quạt khi nhiệt độ nhỏ hơn ngưỡng đặt và tắt vòi phun khi độ ẩm cao hơn ngưỡng:



- **Quan sát kết quả ESP32 trên cửa số Arduino IDE:**

Kết nối wifi và broker thành công:

```
WiFi connected
IP address:
192.168.137.82
Attempting MQTT connection...connected
```

Khi hệ thống hoạt động ở chế độ auto:


```
{"temperature":"31.30","humidity":"75.00","mode":"1","fan":"0","valve":"1"}
{"temperature":"31.30","humidity":"75.00","mode":"1","fan":"0","valve":"1"}
{"temperature":"31.30","humidity":"75.00","mode":"1","fan":"0","valve":"1"}
{"temperature":"31.30","humidity":"75.00","mode":"1","fan":"0","valve":"1"}
{"temperature":"31.30","humidity":"75.00","mode":"1","fan":"0","valve":"1"}
{"temperature":"31.30","humidity":"75.00","mode":"1","fan":"0","valve":"1"}
{"temperature":"31.50","humidity":"77.00","mode":"1","fan":"0","valve":"1"}
{"temperature":"31.80","humidity":"80.00","mode":"1","fan":"0","valve":"0"}
{"temperature":"32.20","humidity":"83.00","mode":"1","fan":"0","valve":"0"}
{"temperature":"32.80","humidity":"83.00","mode":"1","fan":"0","valve":"0"}
{"temperature":"33.30","humidity":"81.00","mode":"1","fan":"0","valve":"0"}
{"temperature":"34.20","humidity":"81.00","mode":"1","fan":"0","valve":"0"}
{"temperature":"34.70","humidity":"79.00","mode":"1","fan":"0","valve":"1"}
{"temperature":"35.20","humidity":"77.00","mode":"1","fan":"0","valve":"1"}
{"temperature":"35.60","humidity":"77.00","mode":"1","fan":"0","valve":"1"}
{"temperature":"36.40","humidity":"75.00","mode":"1","fan":"0","valve":"1"}
{"temperature":"37.00","humidity":"74.00","mode":"1","fan":"0","valve":"1"}
{"temperature":"38.00","humidity":"72.00","mode":"1","fan":"0","valve":"1"}
{"temperature":"38.50","humidity":"72.00","mode":"1","fan":"0","valve":"1"}
```

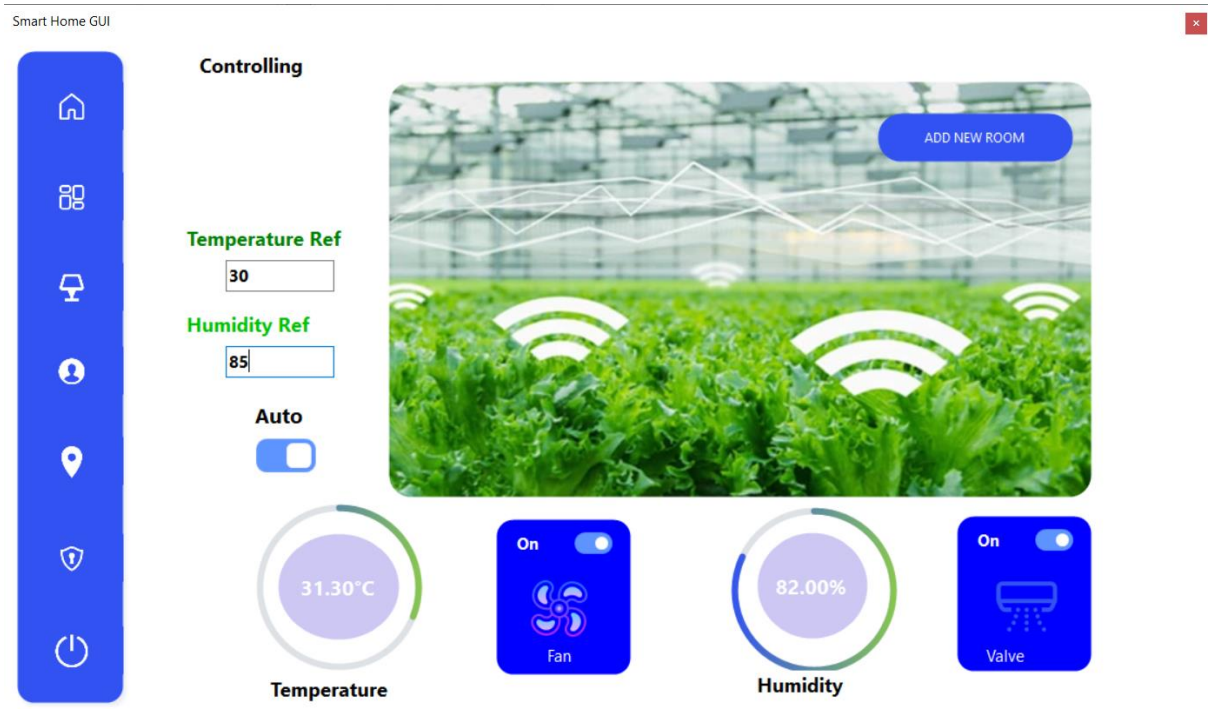
Khi hệ thống hoạt động ở chế độ manual dữ liệu được gửi từ broker xuống ESP32:

```
{"temperature":"31.30","humidity":"79.00","mode":"1","fan":"0","valve":"1"}
{"temperature":"31.30","humidity":"79.00","mode":"1","fan":"0","valve":"1"}
{"fan":"0","valve":"1","mode":"0","setTemp":"27","setHumid":"30"}
valve on
fan off
{"temperature":"31.30","humidity":"79.00","mode":"0","fan":"0","valve":"1"}
{"fan":"1","valve":"1","mode":"0","setTemp":"27","setHumid":"30"}
valve on
fan on
{"temperature":"31.30","humidity":"79.00","mode":"0","fan":"1","valve":"1"}
{"fan":"0","valve":"1","mode":"0","setTemp":"27","setHumid":"30"}
valve on
fan off
{"temperature":"31.30","humidity":"79.00","mode":"0","fan":"0","valve":"1"}
{"temperature":"31.30","humidity":"79.00","mode":"0","fan":"0","valve":"1"}
{"fan":"0","valve":"1","mode":"1","setTemp":"27","setHumid":"30"}
{"temperature":"31.30","humidity":"79.00","mode":"1","fan":"1","valve":"0"}
{"temperature":"31.30","humidity":"79.00","mode":"1","fan":"1","valve":"0"}
{"temperature":"31.30","humidity":"79.00","mode":"1","fan":"1","valve":"0"}
{"temperature":"31.30","humidity":"79.00","mode":"1","fan":"1","valve":"0"}
{"fan":"1","valve":"0","mode":"0","setTemp":"27","setHumid":"30"}
valve off
fan on
{"temperature":"31.30","humidity":"79.00","mode":"0","fan":"1","valve":"0"}
{"fan":"1","valve":"0","mode":"1","setTemp":"27","setHumid":"30"}
{"temperature":"31.30","humidity":"79.00","mode":"1","fan":"1","valve":"0"}
{"temperature":"31.30","humidity":"79.00","mode":"1","fan":"1","valve":"0"}
```

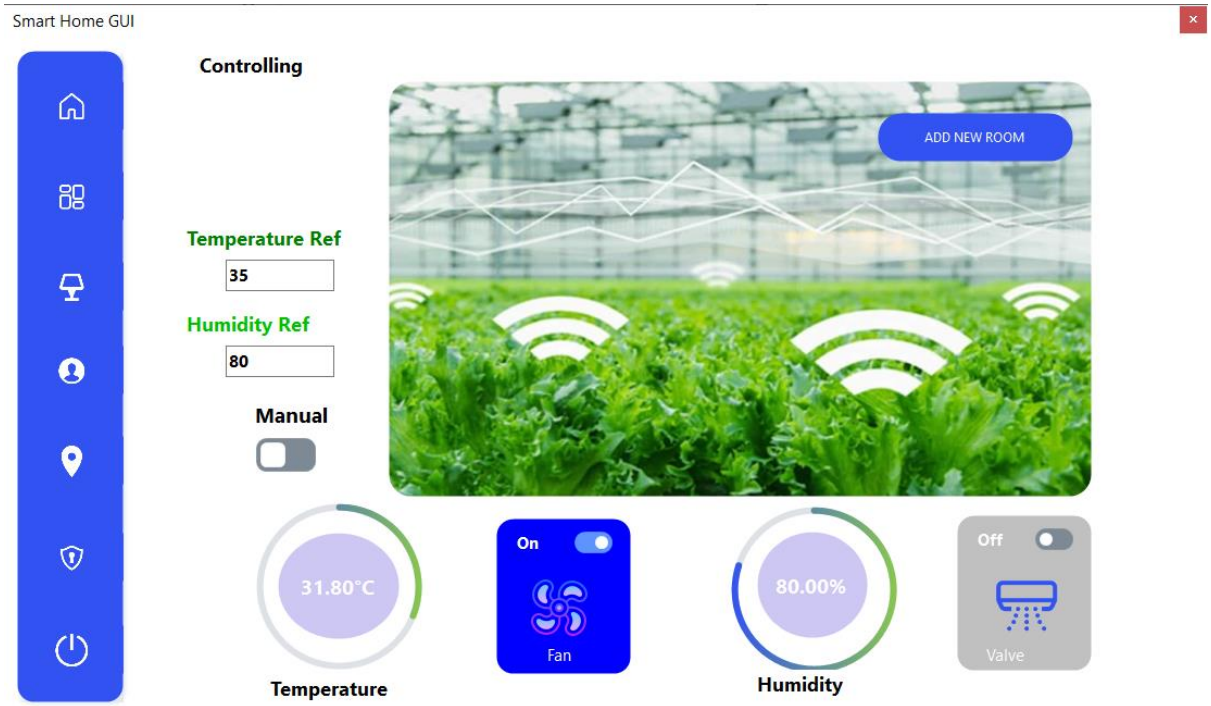
2. Giao diện giám sát:

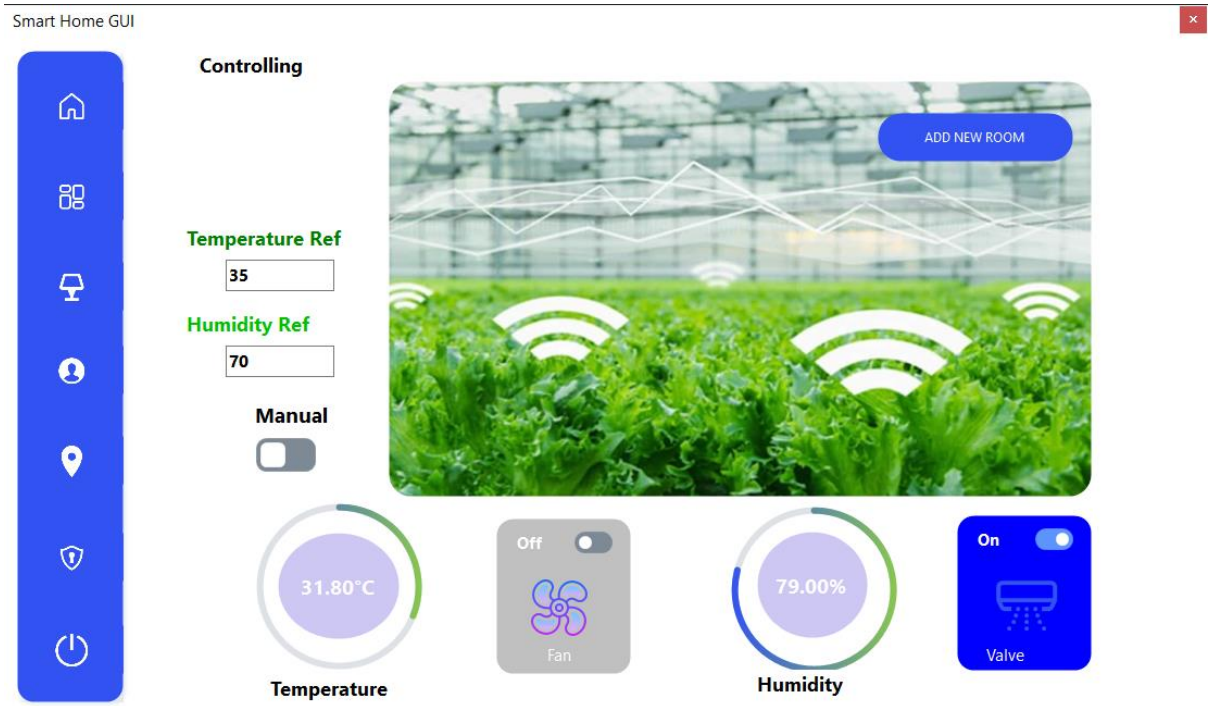
Phần mềm khi hoạt động ở chế độ auto:

Khi nhiệt độ vườn tăng cao và độ ẩm không khí thấp:



Ở chế độ manual có thể tùy chon điều khiển quạt hoặc vòi phun:




Smart Home GUI
Controlling
ADD NEW ROOM
Temperature Ref
35
Humidity Ref
70
Manual
31.80°C
Temperature
Off
Fan
79.00%
Humidity
On
Valve

TÀI LIỆU THAM KHẢO

1. Điện tử tương lai, *ESP32 là gì?*, link truy cập: https://dientutuonglai.com/esp32-la-gi.html
2. Khuê Nguyễn (30/07/2021), Lập trình STM32 với DHT11 theo chuẩn 1 Wire, link truy cập: https://khuenguyencreator.com/lap-trinh-stm32-voi-dht11-theo-chuan-1-wire/#:~:text=DHT11%20s%E1%BB%AD%20d%E1%BB%A5ng%20giao%20th%E1%BB%A9c,ch%C3%BAt%20v%E1%BB%81%20chu%E1%BA%A9n%20n%C3%A0y%20nh%C3%A9
3. Thanh Thư (11/9/2020), *MQTT là gì? Vai trò của MQTT trong IoT,* link truy cập: https://viblo.asia/p/mqtt-la-gi-vai-tro-cua-mqtt-trong-iot- V3m5WL3bKO