TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN ĐIỆN

----------*****----------

BÁO CÁO BÀI TẬP LỚN

MÔN HỌC: VI XỬ LÝ

ĐỀ TÀI: HỆ THỐNG GIÁM SÁT VÀ ĐIỀU KHIỂN TỦ BẢO QUẢN THEO CÁC THÔNG SỐ NHIỆT ĐỘ VÀ ĐỘ ẨM

Giảng viên hướng dẫn : ThS. Trần Văn Tuấn

Sinh viên thực hiện : Trần Tuấn Anh 20172949

Vũ Tiến Anh 20172955

Bùi Minh Hiếu 20173114

Nguyễn Xuân Phương 20172097

Đỗ Lê Ngọc Tân 20173582

Lớp : KSCLC - K62

Hà Nội, tháng 9 năm 2021

LỜI NÓI ĐẦU

Ngày nay, cùng với sự phát triển của khoa học kỹ thuật là sự phát triển vượt bậc của các thiết bị IOT. IOT xuất hiện ở khắp mọi nơi, mọi hoạt động trong đời sống hiện tại và đang trở thành một xu hướng mới ở cuộc cách mạng công nghệ 4.0.

Vì vậy, với những kiến thức đã học của ngành điện, nhóm của chúng em xin chọn đề tài: "**Hệ thống giám sát và điều khiển tủ bảo quản theo các thông số nhiệt độ và độ ẩm**".

Do kiến thức còn hạn hẹp nên trong quá trình thực hiên đề tài không thể tránh những sai sót rất mong quý thầy cô bỏ qua và có hướng giúp đỡ để em có hướng đi cao hơn sau này trong lĩnh vực nghiên cứu khoa học.

Em xin chân thành cám ơn:

- ThS. Trần Văn Tuấn, giảng viên viện Điện là người đã giúp cho em có những kiến thức cần thiết thông qua học phần "Vi xử lý" và trực tiếp hướng dẫn nhiệt tình cho chúng em trong suốt quá trình thực hiện đề tài này.

MỤC LỤC

CHƯƠNG 1: GIỚI THIỆU CHUNG

I. Giới thiệu tổng quan về đề tài

Trong dự án này, chúng em muốn từ những kiến thức đã được học về điện, điện tử, lập trình ứng dụng, IOT,… của mình để xây dựng một mạch có chức năng theo dõi độ ẩm và nhiệt độ của tủ, hiển thị lên màn hình LCD và giao diện trên local webserver. Có thể điều chỉnh các giá trị offset và tới hạn của nhiệt độ/ độ ẩm thông qua bàn phím hoặc nhập từ webserver, từ đó tự động điều khiển các tải hoạt động cho phù hợp. Tuy nhiên do tình hình dịch bệnh nên nhóm em chưa có đủ linh kiện để làm phần điều khiển qua bàn phím.

II. Phân công công việc

Tên	Nhiệm vụ
Trần Tuấn Anh	Tạo local webserver và xử lý HTTP request để điều khiển tải
Nguyễn Xuân Phương	Tìm hiểu lý thuyết và viết chương trình cho phần hiển thị màn hình LCD
Đỗ Lê Ngọc Tân	Tìm hiểu lý thuyết và viết chương trình đọc nhiệt độ, độ ẩm từ cảm biến DHT11
Bùi Minh Hiếu	Tìm hiểu lý thuyết và tạo giao diện webserver, tạo HTTP request để điều khiển từ xa
Vũ Tiến Anh	Tìm hiểu lý thuyết và lập trình kết nối với wifi, xử lý đọc ghi SPI flash

CHƯƠNG 2: LÝ THUYẾT

I. Giới thiệu về vi xử lý ESP32

ESP32 là một vi điều khiển giá rẻ, năng lượng thấp có hỗ trợ WiFi và dual-mode Bluetooth (tạm dịch: Bluetooth chế độ kép). Dòng ESP32 sử dụng bộ vi xử lý Tensilica Xtensa LX6 ở cả hai biến thể lõi kép và lõi đơn, và bao gồm các công tắc antenna tích hợp, RF balun, bộ khuếch đại công suất, bộ khuếch đại thu nhiễu thấp, bộ lọc và module quản lý năng lượng.

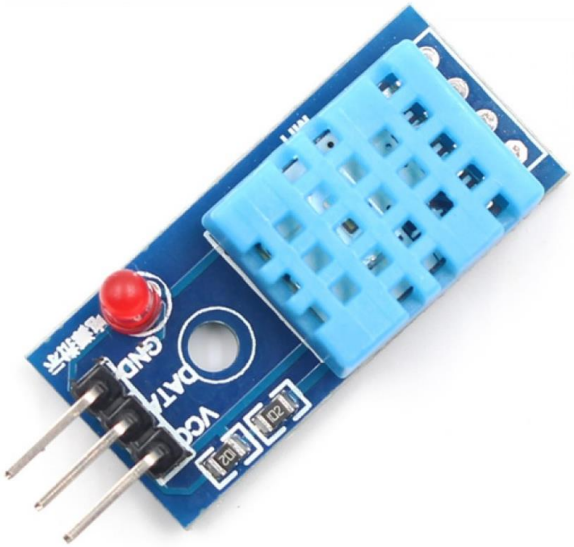
ESP32 được chế tạo và phát triển bởi Espressif Systems, một công ty Trung Quốc có trụ sở tại Thượng Hải, và được sản xuất bởi TSMC bằng cách sử dụng công nghệ 40 nm. ESP32 là sản phẩm kế thừa từ vi điều khiển ESP8266.

Hình 1. Board ESP32 Devkit

II. Cảm biến DHT11

- DHT11 là một cảm biến nhiệt độ và độ ẩm kỹ thuật số cơ bản.
- DHT11 dựa trên giao thức truyền tin 1 dây nhằm cung cấp các giá trị độ ẩm và nhiệt độ.
- Cảm biến DHT11 cung cấp giá trị độ ẩm tương đối theo phần trăm (20 đến 90% RH) và các giá trị nhiệt độ theo độ C (0 đến 50°C).

Chân	Tên chân	Mô tả
1	VCC	Điện áp vào từ 3.3 đến 5.5 V DC
2	DATA	Lấy giá trị đầu ra
3	NC	Không sử dụ0ng
4	GND	Nối vào nguồn 0V



Hình 2. Cảm biến DHT11

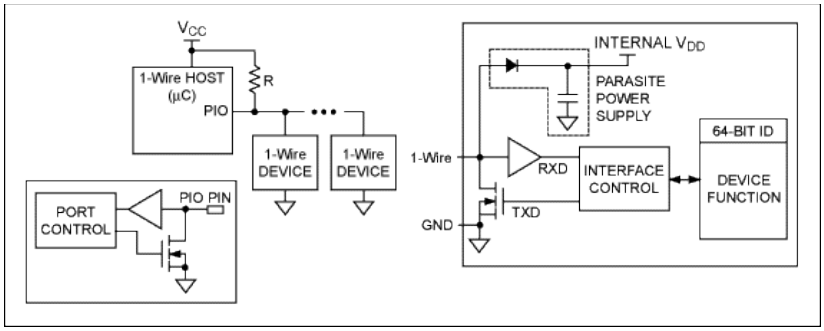
III. Giao thức truyền tin 1 dây (One-wire protocol)

1-Wire là một hệ thống bus giao tiếp với thiết bị được thiết kế bởi Dallas Semiconductor Corp.

Khái niệm cơ bản của công nghệ 1 dây: là một giao thức nối tiếp sử dụng một đường dây dữ liệu duy nhất với tham chiếu đất để giao tiếp. Thiết bị chủ (Master) của giao thức 1 dây sẽ khởi tạo và điều khiển giao tiếp của một hoặc nhiều thiết bị phụ (Slave) trên hệ thống bus. Mỗi thiết bị phụ 1 dây có 64 bit ID duy nhất, không thể thay đổi, được lập trình hướng đối tượng, đóng vai trò là địa chỉ trên bus một dây. Mã vạch 8 bit, một tập hợp con của ID 64 bit, xác định loại thiết bị và chức năng. Thông thường, thiết bị phụ 1 dây có 4 dải điện áp sau:

- 1.71V (min) to 1.89V (max)
- 1.71V (min) to 3.63V (max)
- 2.97V (min) to 3.63V (max)
- 2.8V (min) to 5.25V (max)

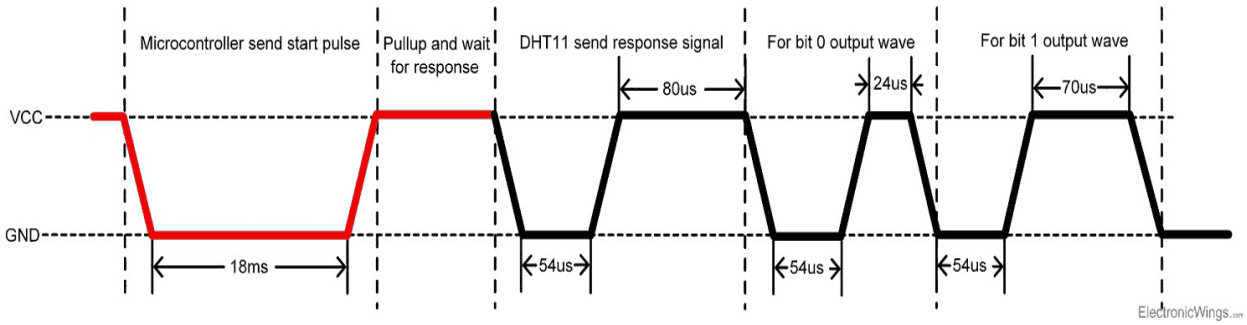
Hầu hết các thiết bị 1 dây không có chân cho nguồn điện, chúng lấy nguồn điện từ bus 1 dây.



- Giao diện chủ (Simple host interface): Thường sẽ có chân cổng dự phòng.
- Truyền thông nối tiếp 2 hướng: Chế độ truyền bán song công (việc truyền tín hiệu có thể xảy ra 2 hướng, nhưng 2 thiết bị được kết nối sử dụng luân phiên kênh (chỉ được chuyền theo một chiều tại một thời điểm)).
- Nguồn cung cấp mạch nhiễu: lấy được từ tín hiệu truyền thông nối tiếp và lưu lại trên tụ bên trong.

1. Giao tiếp truyền tin với vi xử lý

- Các mức điện áp với giá trị thời gian nhất định sẽ xác định logic 1 hoặc logic 0 trên chân kết nối.
- Giao tiếp sẽ được chia làm 3 bước, đầu tiên là gửi request tới cảm biến, sau đó cảm biến sẽ gửi tới xung phản hồi, tiếp đến cảm biến sẽ gửi dữ liệu khoảng 40 bit tới vi xử lý.



2. Xung bắt đầu

- Để bắt đầu giao tiếp, đầu tiên chúng ta sẽ gửi xung mẫu bắt đầu theo dạng hình thang tới cảm biến. Lý do là xung hình thang: do có độ trễ thời gian cấp điện,...
- Để cung cấp xung xuất mẫu bắt đầu, kéo xuống chân dữ liệu (DATA) tối thiểu 18ms và sau đó sẽ kéo lên, được hiển thị tại đường chéo trong hình.

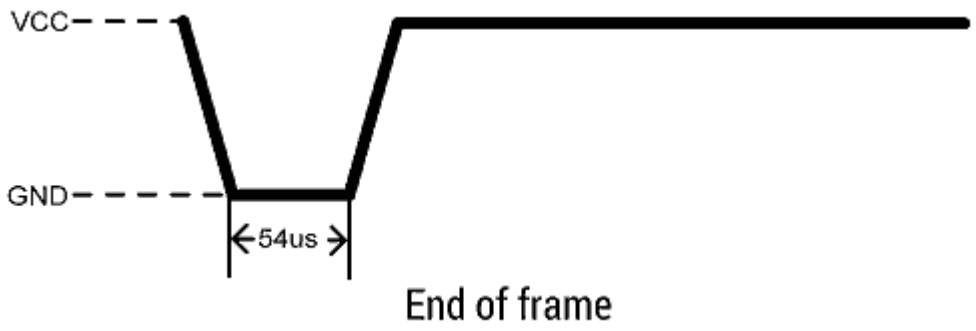
3. Xung phản hồi

- Sau khi xung mẫu bắt đầu, cảm biến sẽ gửi lại xung phản hồi để thấy được rằng cảm biến đã nhận về xung bắt đầu.
- Xung phản hồi thấp vào khoảng 54us, trong khi xung phản hồi cao vào khoảng 80us.

4. Dữ liệu

- Sau khi gửi xung phản hồi, cảm biến DHT11 sẽ gửi dữ liệu lên CPU, nơi mà chứa giá trị nhiệt độ và độ ẩm, cùng lúc đó sẽ kiểm tra giá trị.
- Khung dữ liệu có độ dài 40 bit, chứa 5 đoạn (byte) và mỗi đoạn dài 8 bit.
- Trong 5 byte trên, 2 byte đầu sẽ ghi lại giá trị độ ẩm dưới dạng số nguyên thập phân. Giá trị này sẽ cho chúng ta biết được độ ẩm phần trăm tương đối. 8 bit đầu là phần nguyên còn 8 bit sau là phân số.
- 2 byte kế tiếp sẽ chứa giá trị nhiệt độ theo dạng số nguyên thập phân. Giá trị này sẽ cho chúng ta biết được nhiệt độ theo độ C.
- Byte checksum sẽ xác minh xem giá trị tổng 4 byte trước có giống với giá trị kiểm tra hay không. Nếu không giống nhau, sẽ hiện ra một số lỗi dữ liệu nhận được, cụ thể trong code là lỗi 301.
- Khi dữ liệu đã được nhận, chân DHT11 sẽ ở chế độ tiêu thụ điện năng thấp cho đến khi có tín hiệu xung bắt đầu tiếp theo.

5. Xung kết thúc



- Sau khi gửi 40 bit dữ liệu, cảm biến DHT11 gửi xung thấp khoảng 54us và lên xung cao. Ngay sau đó, cảm biến chuyển về chế độ duy trì (Sleep mode).

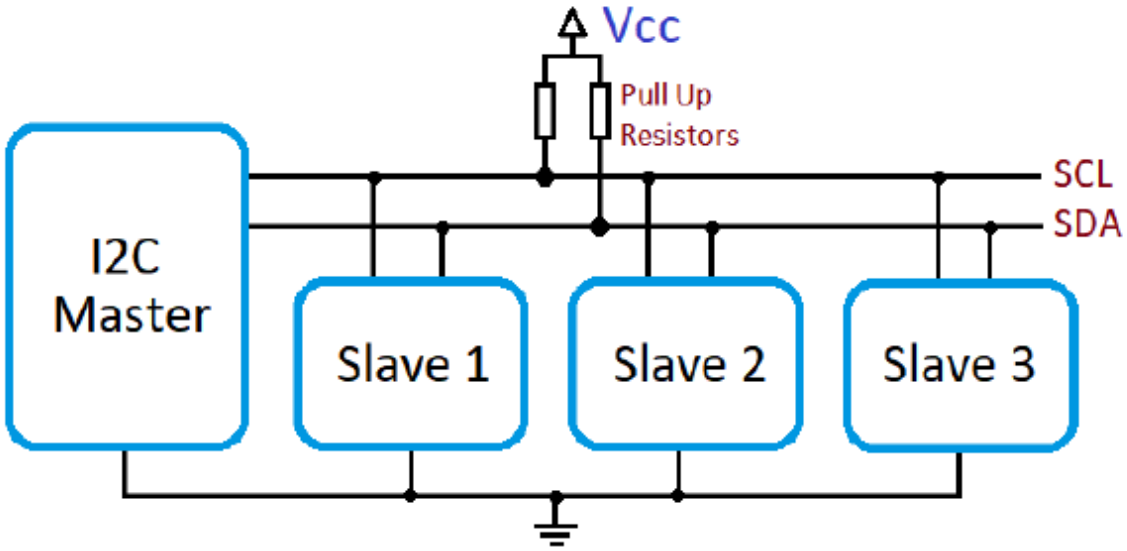
IV. Giao tiếp I2C

1. Định nghĩa

I2C(Inter - Intergrated Circuit) là 1 giao thức giao tiếp nối tiếp đồng bộ được phát triển bởi Phillips Semiconductors, sử dụng để truyền nhận dữ liệu giữa các IC với nhau chỉ sử dụng hai đường truyền tín hiệu. I2C theo giao tiếp Master-Slave.

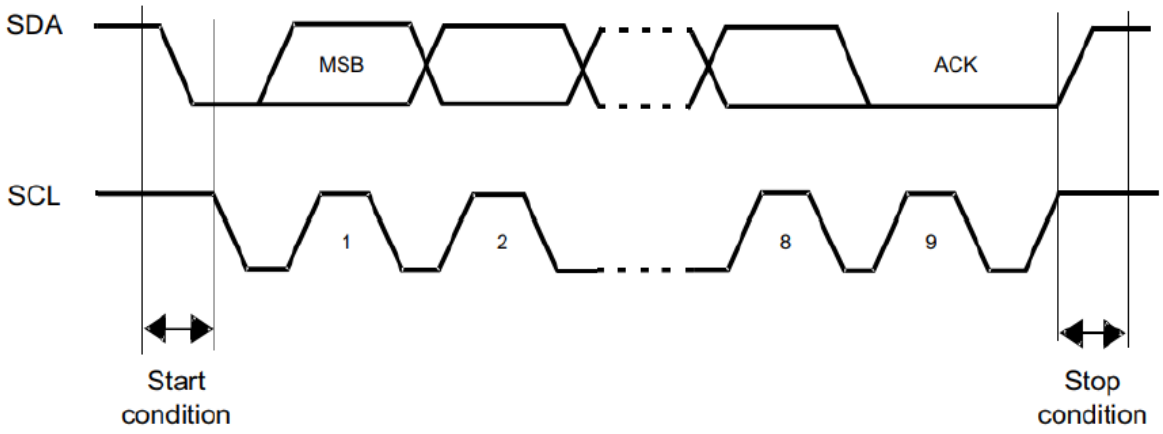
I2C sử dụng hai đường truyền tín hiệu:

- Serial Clock Line (SCL): Tạo xung nhịp đồng hồ do Master phát đi.
- Serial Dataline (SDA): Đường truyền nhận dữ liệu.



Hình 3. Giao tiếp I2C

2. Hoạt động của giao tiếp I2C



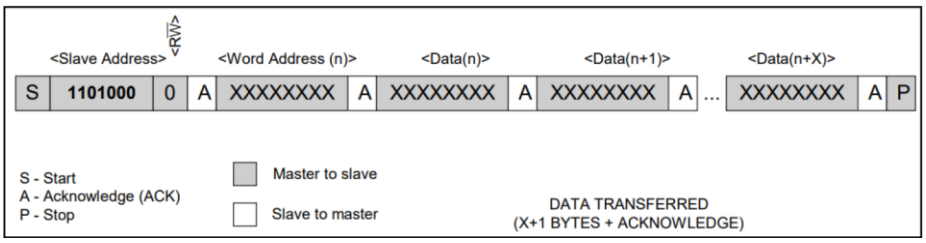
Bắt đầu với Start Condition(điều kiện khởi tạo): khi tín hiệu Master từ dây SDA chuyển tín hiệu từ mức cao xuống mức thấp, dây SCL sẽ bắt đầu phát tín hiệu Clock từ Master chuyển đến Slave.

Kết thúc với Stop Condition(điều kiện dừng): sau khi dây SCL truyền đủ 8-bit tín hiệu và mức tín hiệu sẽ được chuyển từ mức thấp lên mức tín hiệu cao, và tín hiệu của dây SDA cũng sẽ được chuyển thành mức tín hiệu cao.

Bit ACK(Acknowledge) trong I2C: kéo đường bus SDA xuống mức tín hiệu thấp.

a. Quá trình truyền dữ liệu đến thiết bị I2C

- **Bước 1: Master** gửi tín hiệu start I2C.
- **Bước 2: Master** gửi địa chỉ của thiết bị I2C (7 bit) kèm bit **Write** (bit 0) - tùy loại IC như IC thời gian thực thì mới có thể ghi dữ liệu.
- **Bước 3: Master** gửi địa chỉ của thanh ghi dữ liệu của Slave muốn ghi giá trị.
- **Bước 4: Master** gửi giá trị mà muốn ghi vào thanh ghi ở bước 3.
- **Bước 5: Master** tạo tín hiệu Stop.

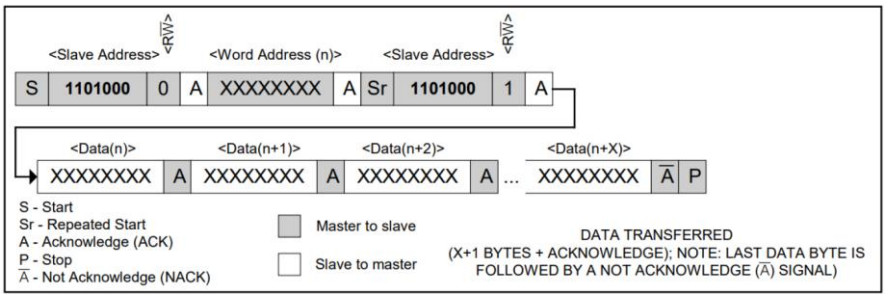


Số slave tối đa có thể kết nối là 2^7 - 1 thiết bị

Sau khi gửi đủ giá trị 1 byte thì Slave phản hồi 1 bit ACK.

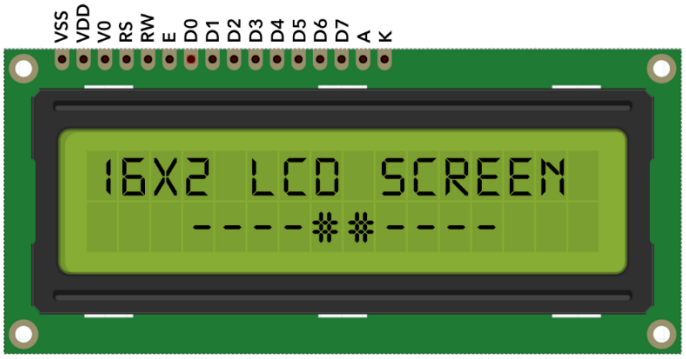
b. Quá trình nhận dữ liệu từ thiết bị I2C

- **Bước 1: Master** gửi tín hiệu start I2C.
- **Bước 2: Master** gửi địa chỉ của thiết bị I2C (7 bit) kèm bit **Write** (bit 0).
- **Bước 3: Master** gửi đia chỉ thanh ghi của Slave mà muốn đọc dữ liệu.
- **Bước 4: Master** gửi tín hiệu **Repeated Start.**
- **Bước 5: Master** gửi địa chỉ của thiết bị I2C (7 bit) kèm bit **Read** (bit 1).
- **Bước 6: Master** đọc dữ liệu chứa trong thanh ghi ở **bước 3** từ Slave gửi về.
- **Bước 7: Master** tạo tín hiệu Stop.



V. Màn hình LCD

1. LCD 16×2



No	Symbol	Function
1	VSS	Ground
2	VDD	5V +
3	V0	Contrast
4	RS	Register
5	RW	Read/Write
6	E	Enable
7	D0	Data bus
8	D1	Data bus
9	D2	Data bus
10	D3	Data bus
11	D4	Data bus
12	D5	Data bus
13	D6	Data bus
14	D7	Data bus
15	A	Anode (5V+)
16	K	Cathode (GND)

Hình 4. Màn hình LCD 16x2

LCD 16×2 có 16 chân trong đó:

- VSS: nguồn 0V
- VCC: Nguồn 5V
- VEE: Chỉnh độ sáng màn hình
- RS: chọn thanh ghi
- RW: Chọn chế độ đọc ghi
- E: chân enable
- D0-D7: chân data
- A/K: nguồn cho led màn hình

Để giao tiếp với LCD, ta cần phải gửi các lệnh cấu hình và data thông qua các chân D0-D7:

- Để gửi lệnh cấu hình, ta sẽ kéo chân RS, RW xuống mức 0. Tiếp theo là ghi giá trị lệnh vào các chân data, sau đó tạo 1 xung low-to-high trên chân E để gửi tín hiệu vào các thanh ghi trong LCD.
- Để gửi data, ta kéo chân RW xuống mức 0, RS lên mức 1 và sau đó làm tương tự như lệnh gửi cấu hình.

Một số mã lệnh thường dùng:

Mã lệnh	Chức năng	T_{exe}
0x01	Xoá toàn bộ nội dung đang hiển thị trên màn hình.	1.52ms
0x02	Di chuyển con trỏ về vị trí đầu màn hình.	1.52ms
0x06	Tự động di chuyển con trỏ đến vị trí tiếp theo mỗi khi xuất ra LCD 1 ký tự.	37us
0x0C	Bật hiển thị và tắt con trỏ	37us
0x0E	Bật hiển thị và bật con trỏ	37us
0x80	Di chuyển con trỏ về đầu dòng 1	37us
0xC0	Di chuyển con trỏ về đầu dòng 2	37us
0x38	Giao tiếp 8 bit, hiển thị 2 dòng, kích thước font 5x7	37us
0x28	Giao tiếp 4 bit, hiển thị 2 dòng, kích thước font 5x7	37us

2. Module mở rộng chân cho giao tiếp LCD

LCD có quá nhiều nhiều chân gây khó khăn trong quá trình đấu nối và chiếm dụng nhiều chân trên vi điều khiển.

Thay vì phải mất 6 chân vi điều khiển để kết nối với LCD 16×2 (RS, EN, D7, D6, D5 và D4) thì **module I2C** chỉ cần tốn 2 chân (SCL, SDA) để kết nối. Ở đây nhóm chọn module PCF8574:

Hình 5. Module I2C PCF8574

Module này hoạt động dựa trên giao tiếp I2C đã trình bày ở trên. Thay vì việc dùng 8 chân của vi điều khiển nối với 8 chân data, ta sẽ gửi dữ liệu qua 1 chân duy nhất nối với chân SDA của module. Thông qua chân này có thể gửi lệnh cũng như data để hiển thị lên màn hình LCD.

VI. Timer

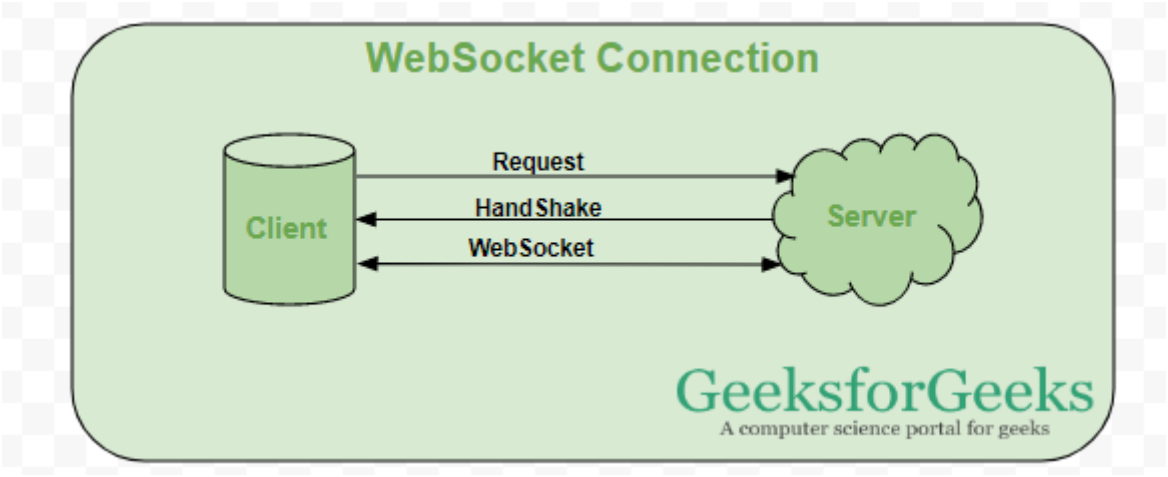
Timer là 1 ngoại vi cơ bản của bất kỳ 1 vi điều khiển nào. Bản chất timer là 1 mạch digital logic có vai trò đếm mỗi chu kỳ clock (đếm lên hoặc đếm xuống). Tốc độ của bộ đếm sẽ phụ thuộc vào tần số của nguồn xung. Trong mỗi vi xử lý sẽ có 2 loại nguồn xung chính là từ bộ dao động nội hoặc lấy từ bên ngoài.Trong ESP32, sẽ có 2 hardware timer. Tất cả chúng đều là bộ đếm thời gian chung 64 bit dựa trên bộ chia trước 16 bit và bộ đếm lên / xuống 64 bit có khả năng tự động tải lại (auto-reload).

Ngoài ra ta còn có thể sử dụng các software timer khi sử dụng thư viện FreeRTOS. Trong project này, nhóm sẽ dùng timer cứng của vi điều khiển và timer sẽ có vai trò dùng để định thời các sự kiện.

VII. Websocket

WebSocket là một giao thức giúp truyền dữ liệu hai chiều giữa server-client qua một kết nối TCP duy nhất. WebSocket giúp client và server (máy chủ) có thể cùng gửi yêu cầu và trả về dữ liệu cùng lúc song song với nhau. Mặc dù được thiết kế để chuyên sử dụng cho các ứng dụng web, công nghệ này có thể được đưa vào bất kì loại ứng dụng nào.

Kết nối socket được thiết lập theo mô hình dưới đây:

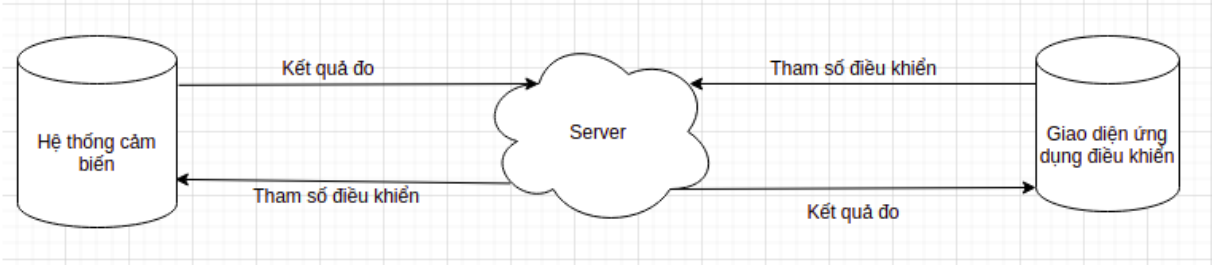


Việc sử dụng giao thức WebSocket để kết nối Client (Hệ thống cảm biến, vi xử lý) với Server giúp tối ưu tài nguyên tính toán, cụ thể:

- **Tiết kiệm chi phí thiết lập kết nối.** Do dữ liệu được gửi liên tục từ Client lên Server (10s / lần), việc duy trì kết nối hai chiều giúp giảm chi phí cho việc thiết lập kết nối.
- **Giảm dung lượng của thông điệp trao đổi.**
- **Giảm số lượng request từ client lên server.** Trong trường hợp kết nối không được duy trì. Client buộc phải liên tục request sau mỗi khoảng thời gian cố định để Client có thể lấy các tham số điều khiển từ Server. Kết nối 2 chiều của WebSocket cho phép Server có thể gửi thông điệp điều khiển cho Client mỗi khi tham số điều khiển thay đổi. Điều này tiết kiệm đáng kể tài nguyên tính toán của cả Client và Server, vừa giúp điều khiển trong thời gian thực.

Trong đề tài do nhóm xây dựng, WebSocket được sử dụng để kết nối giữa Hệ thống cảm biến với Server và đồng thời Server với Giao diện web của ứng dụng điều khiển.

- Khi Server nhận được thông điệp chứa kết quả đo từ Hệ thống cảm biến, Server lưu lại các thông số này, đồng thời chuyển tiếp thông điệp này để Ứng dụng điều khiển hiển thị lên giao diện.
- Khi người dùng thay đổi tham số điều khiển trên Ứng dụng, Ứng dụng điều khiển sẽ gửi các tham số này cho Server, Server sau đó sẽ chuyển tiếp thông điệp này cho Hệ thống cảm biến.

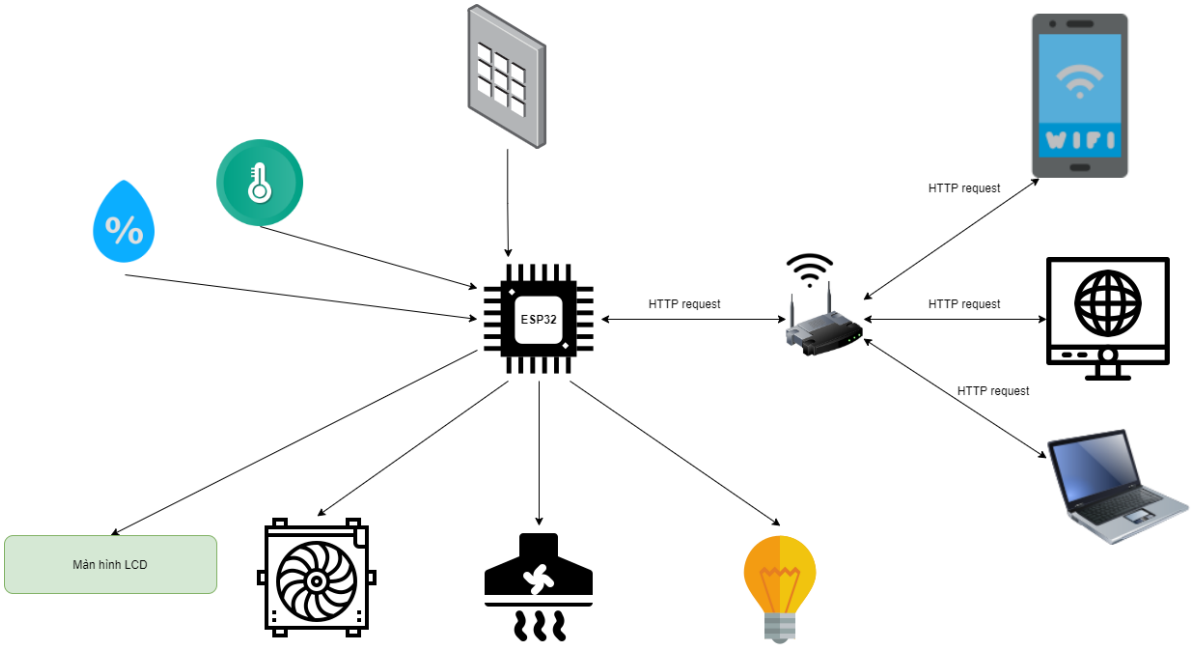


Hình 6. Hoạt động của websocket

CHƯƠNG 3: THIẾT KẾ HỆ THỐNG

I. Tổng quan hệ thống

1. Sơ đồ khối

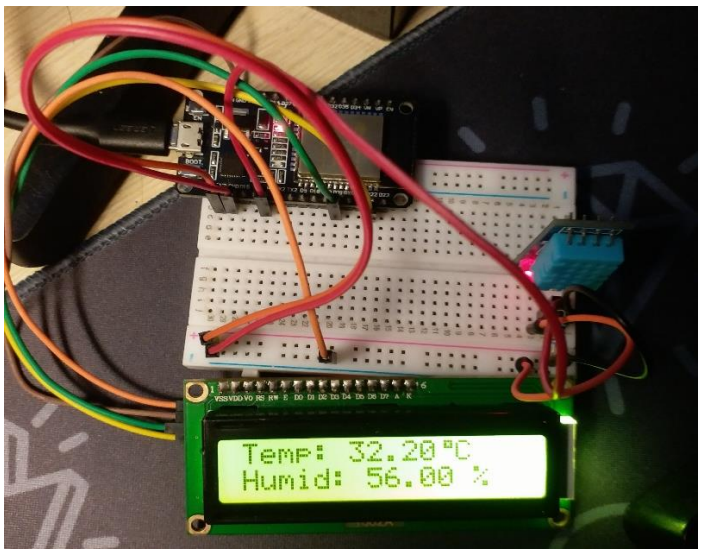


Hình 7. Sơ đồ mô tả hệ thống

Hệ thống gồm 2 khối hoạt động chính:

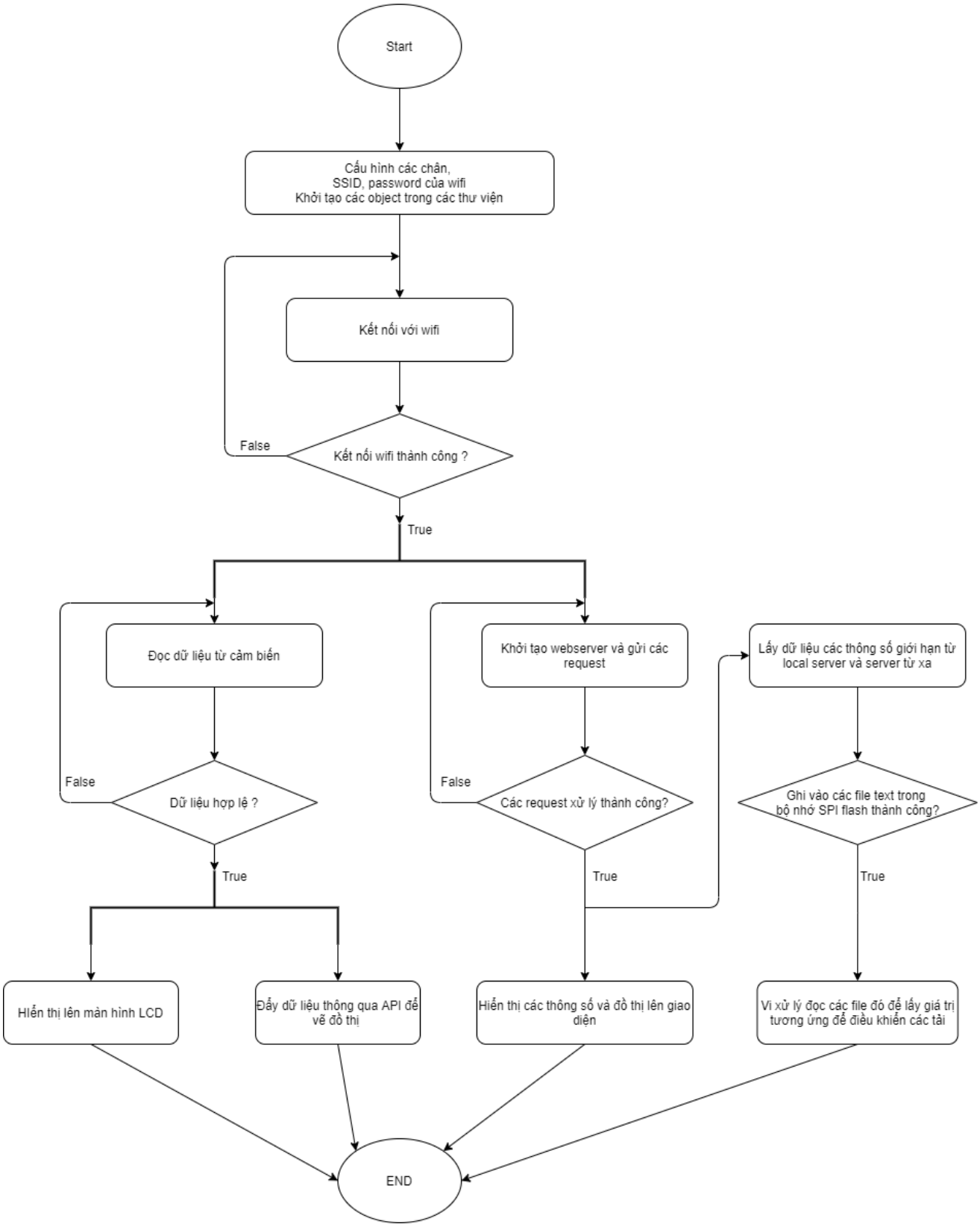
- Khối thiết bị: bao gồm bộ vi xử lý kết nối với cảm biến, màn hình hiển thị, các tải như đèn, quạt, …
- Khối phần mềm điều khiển: webserver để điều khiển và theo dõi từ xa

2. Sơ đồ mạch thực tế



Hình 8. Sơ đồ mạch thực tế

3. Lưu đồ hoạt động của hệ thống

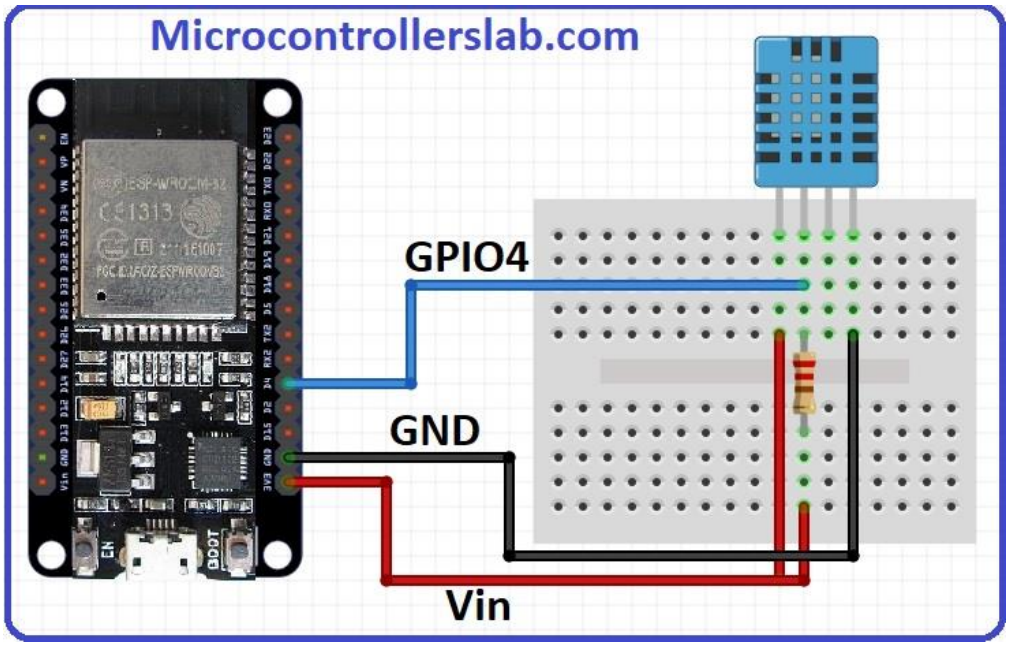


Hình 9. Lưu đồ hoạt động của hệ thống

II. Khối thiết bị

1. Cảm biến DHT11

Như nguyên lý hoạt động đã trình bày ở trên thì ở đây sẽ thực hiện kết nối như hình sau:



Hình 10. Kết nối vi xử lý với cảm biến DHT11

Trong Arduino IDE, cần thêm thư viện "DHT.h" để thực hiện giao tiếp với cảm biến này. Để sử dụng các hàm cho việc giao tiếp, trước khi gọi các hàm sẽ phải khai báo 1 object như sau:

```
DHT dht(DHTPIN, DHTTYPE);
```

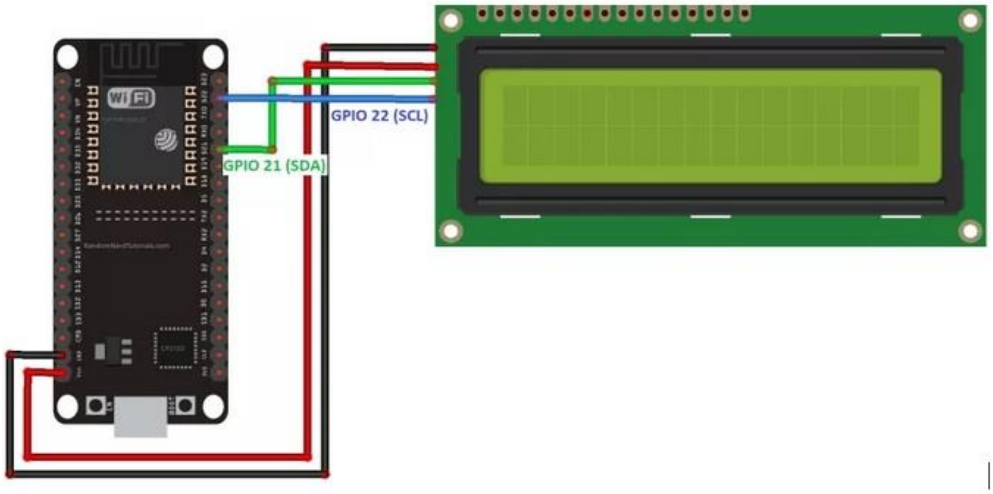
Trong đó ***dht*** là tên object, ***DHTPIN*** là chân data, ***DHTTYPE*** là loại cảm biến sử dụng (dht11)

Các hàm sử dụng trong thư viện DHT:

- Khởi tạo giao tiếp với dht11: **dht11.begin()**
- Đọc độ ẩm: **dht.readHumidity()**
- Đọc nhiệt độ: **dht.readTemperature()**
- Kiểm tra xem dữ liệu đọc được hay không: **isnan()**

2. Giao tiếp màn hình LCD với module I2C

Kết nối màn hình với vi điều khiển như hình dưới đây:



Hình 11. Kết nối vi xử lý với màn hình LCD

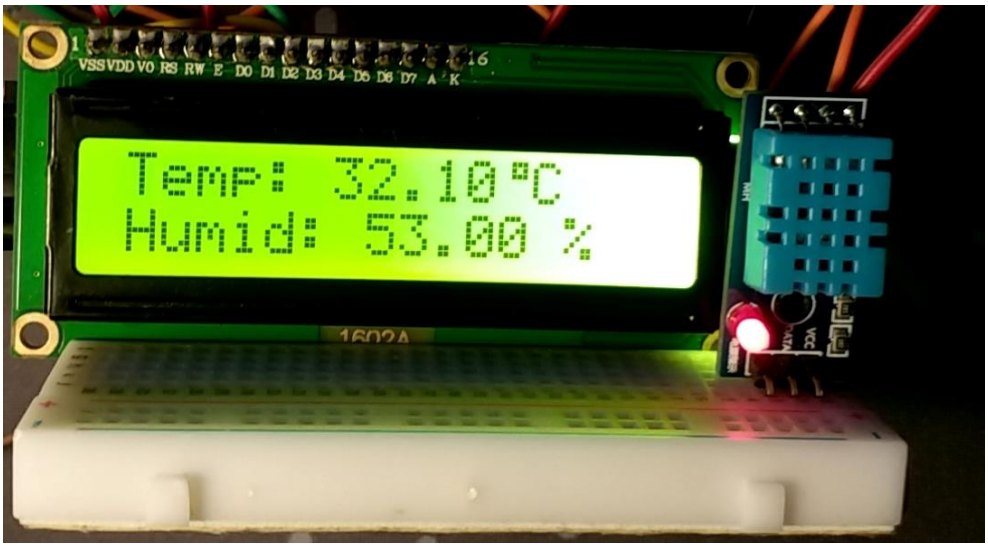
Trong Arduino IDE, cần thêm thư viện "**LiquidCrystal_I2C.h**" để thực hiện giao tiếp với màn hình. Để sử dụng các hàm cho việc giao tiếp, trước khi gọi các hàm sẽ phải khai báo 1 object như sau:

```
LiquidCrystal_I2C lcd(0x27, 16, 2);
```

Với ***lcd*** là tên object, **0x27** là địa chỉ của thiết bị trong mạng I2C, 16 và 2 lần lượt là số hàng và số cột hiển thị của LCD.

Trong project này, em đã tạo 1 ký tự nhiệt độ bằng cách ghi 1 mảng gồm các giá trị nhị phân tương ứng với các pixel trên lcd.

Sau khi đọc được dữ liệu cảm biến, màn hình LCD sẽ hiển thị số liệu như hình:



Hình 12. Hiển thị giá trị nhiệt độ và độ ẩm lên LCD

3. Kết nối với wifi

Trong Arduino IDE, cần thêm thư viện "**WiFi.h**" để thực hiện kết nối wifi.

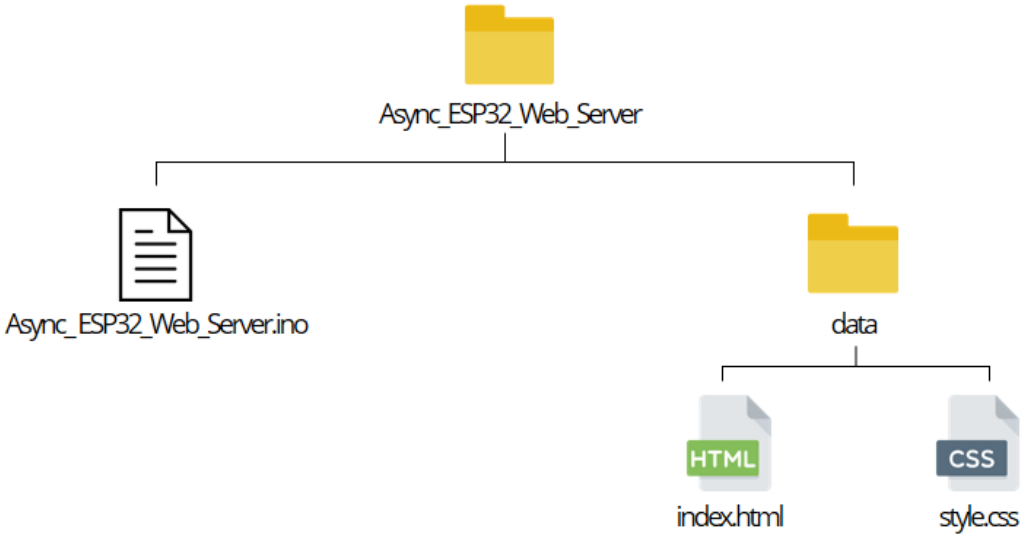
Các hàm sử dụng trong chương trình:

- **WiFi.status():** kiểm tra xem đã có kết nối wifi chưa
- **WiFi.mode(WIFI_STA):** thực hiện cài đặt kết nối ở chế độ station (truy cập vào wifi phát ra từ router)
- **WiFi.begin(ssid, password):** Cài đặt tên và mật khẩu wifi cần truy cập

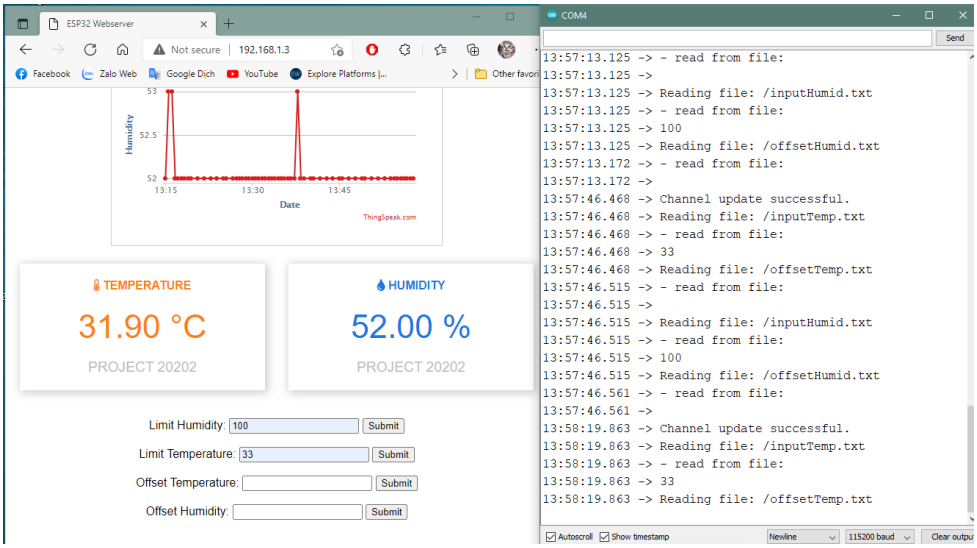
4. SPI flash

Em lựa chọn sử dụng bộ nhớ ngoài là để phòng trường hợp bị mất điện thì dữ liệu sẽ không bị mất. Trong bộ nhớ SPI flash, em sẽ tạo ra 4 file text để lưu các giá trị set up về nhiệt độ và độ ẩm, cùng với đó là 1 file chứa mã HTML của giao diện webserver

Để sử dụng bộ nhớ SPI flash có sẵn, ta phải tổ chức thư mục của project như sau:



Trong chương trình, em tự đã xây dựng các hàm với chức năng đọc/ ghi file trong bộ nhớ. Khi ghi thành công sẽ có thông báo như sau:



III. Khối phần mềm điều khiển

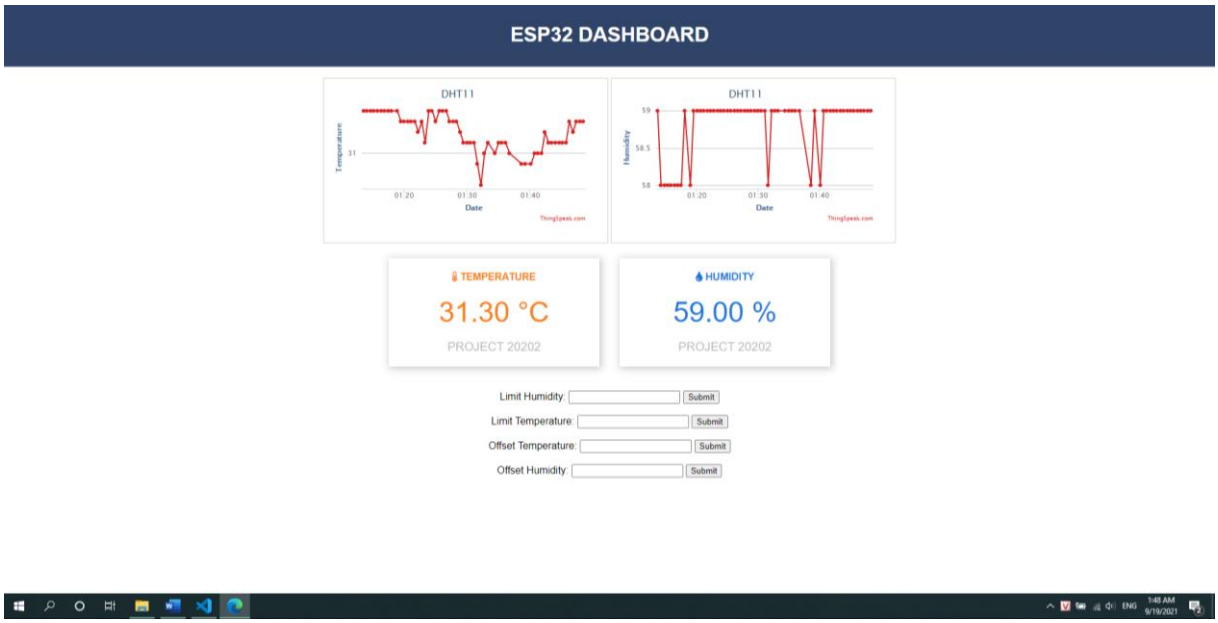
1. Local webserver

Sau khi kết nối ở chế độ station, ta sẽ có 1 địa chỉ IP. Ta chỉ cần kết nối thiết bị (laptop, điện thoại di động,..) vào cùng wifi với ESP32. Sau đó điền địa chỉ IP trên vào 1 trình duyệt web bất kỳ.

Ta sẽ tạo 1 webserver thông qua thư viện "ESPAsyncWebServer.h" trong arduino IDE. Ta sẽ gọi 1 object server để khởi tạo 1 webserver ở port 80:

```
AsyncWebServer server(80);
```

Sau khi kết nối thành công thì trên webserver sẽ hiển thị giao diện như sau:



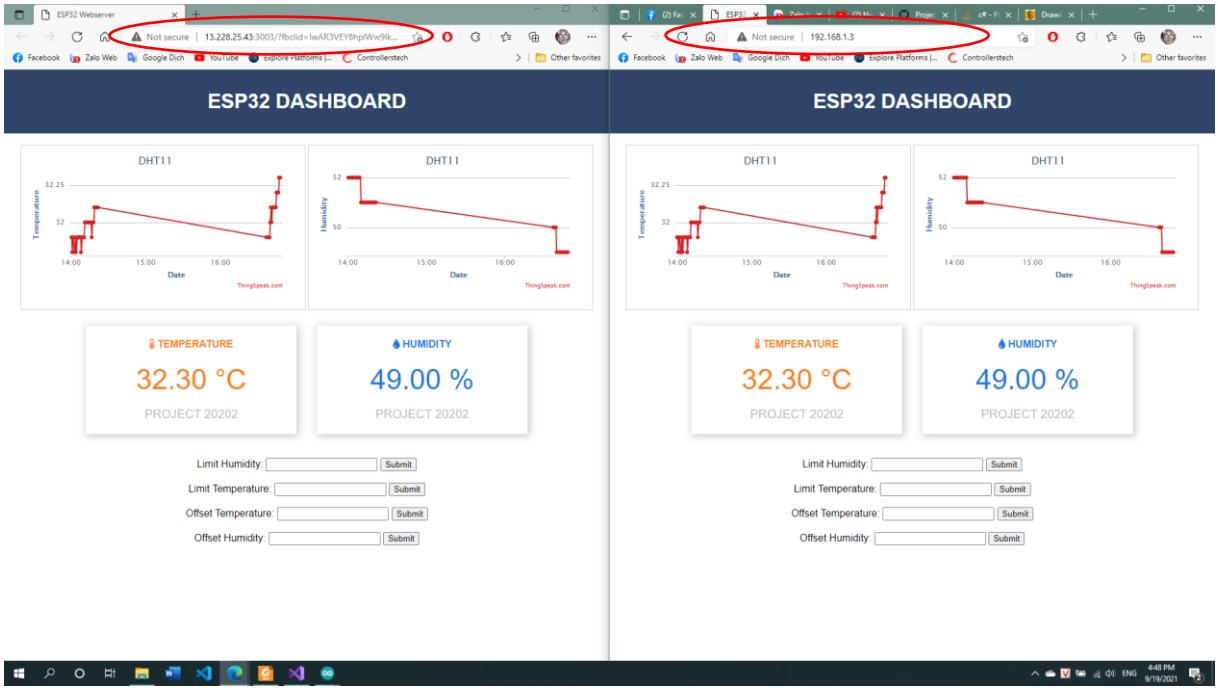
Hình 13. Giao diện webserver local

Phần giao diện webserver gồm các phần:

- Các thẻ hiển thị nhiệt độ, độ ẩm hiện tại
- 2 đồ thị nhiệt độ và độ ẩm theo thời gian để người dùng thuận tiện theo dõi
- Các form để ghi các giá trị giới hạn của nhiệt độ và độ ẩm

2. Kết nối từ xa

Khi khởi tạo 1 webserver ở local, ta có thể truy cập từ xa thông qua 1 webserver kết nối với wifi khác nhưng có các chức năng tương tự như local webserver. Miễn là thiết bị (ESP32) có kết nối wifi.



Hình 14. Giao diện webserver từ xa so sánh với local webserver

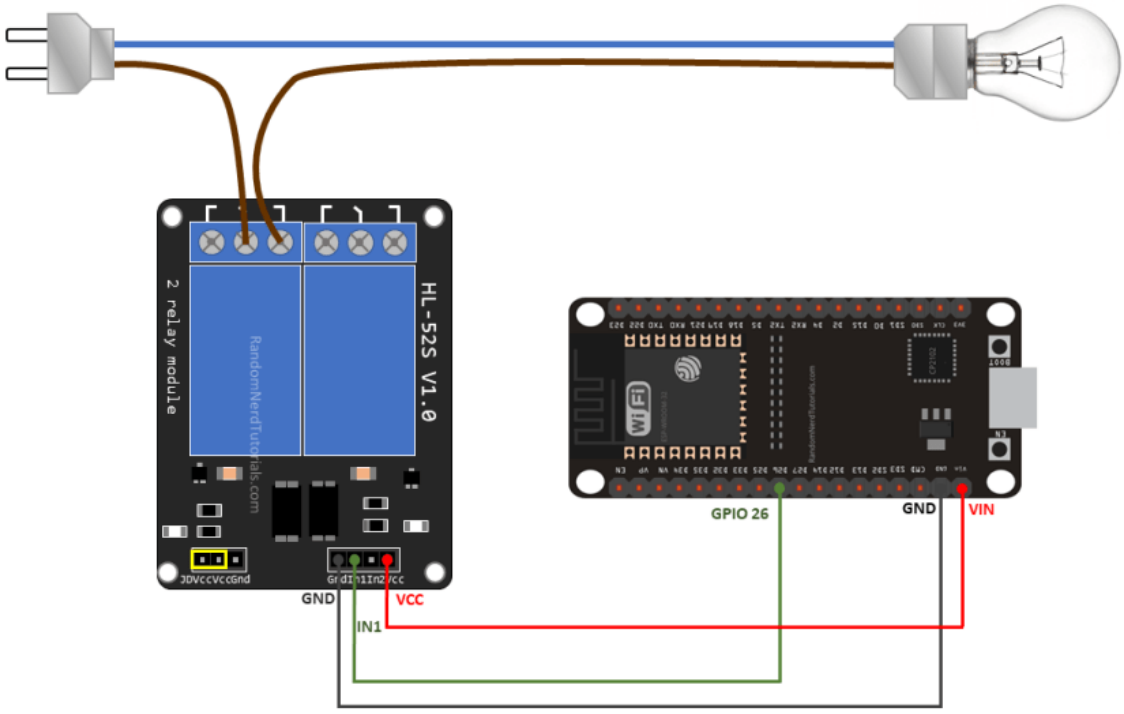
3. Điều khiển các tải

Sau khi ghi các dữ liệu từ webserver vào bộ nhớ flash ngoại, vi xử lý sẽ tiến hành đọc các file đó. Với dữ liệu trong file, vi xử lý sẽ thực hiện so sánh với giá trị nhiệt độ, độ ẩm hiện tại để thực hiện bật/ tắt các tải cho phù hợp. Việc đọc file sẽ diễn ra liên tục, cứ mỗi 30s vi xử lý sẽ thực hiện đọc và so sánh 1 lần

```
-> Reading file: /inputTemp.txt
-> - read from file:
-> 33
-> Reading file: /offsetTemp.txt
-> - read from file:
->
-> Reading file: /inputHumid.txt
-> - read from file:
-> 100
-> Reading file: /offsetHumid.txt
-> - read from file:
->
```

Hình 15. Vi xử lý thực hiện đọc file trong bộ nhớ flash ngoại

Trong thực tế, ta sẽ dùng relay để điều khiển các tải thông qua các mức logic được xuất ra từ chân của vi xử lý.



Hình 16. Kết nối vi xử lý với relay để điều khiển tải

CHƯƠNG 4: TỔNG KẾT

I. Kết quả đạt được

Về cơ bản đã phân tích, thiết kế được 1 hệ thống IoT. Dù đã hoàn thành project nhưng chúng em không tránh được sai xót.

Tổng kết, nhóm đã hoàn thành được những nội dung:

- Chế tạo được 1 thiết bị theo dõi và điều khiển các thông số nhiệt độ, độ ẩm từ xa miễn là nơi đó có kết nối wifi
- Phân tích chức năng của từng khối hoạt động
- Củng cố và áp dụng được các kiến thức tổng hợp về điện, điện tử, lập trình và các kiến thức về giao thức mạng, IOT

II. Hướng phát triển

- Tối ưu các chức năng, giao diện cho hệ thống
- Xây dựng được 1 hệ thống có thể theo dõi được nhiều thiết bị cùng 1 lúc
- Làm được mạch PCB để tiết kiệm chi phí cho thiết bị
- Hướng đến 1 ứng dụng đa nền tảng để tiện lợi cho người sử dụng

TÀI LIỆU THAM KHẢO

https://viblo.asia/p/networking-socket-hoat-dong-nhu-the-nao-aWj53LxYK6m

https://randomnerdtutorials.com/esp32-web-server-spiffs-spi-flash-file-system/

https://randomnerdtutorials.com/esp32-esp8266-i2c-lcd-arduino-ide/

https://www.maximintegrated.com/en/design/technical-documents/tutorials/1/1796.html