HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA KỸ THUẬT ĐIỆN TỬ I

BÁO CÁO HỌC PHẦN THAY THẾ

XỬ LÝ TÍN HIỆU TRONG HỆ THỐNG TRUYỀN THÔNG

Đề tài: "Hệ thống giám sát môi trường sử dụng ESP32 và ESP8266 (Nhiệt độ, Độ ẩm, Ánh sáng, CO2, Báo cháy)"

Giảng viên: Trương Minh Đức

Nhóm môn học: 03

Nhóm báo cáo: 02

Sinh viên thực hiện:

1. Lương Anh Đức - B19DCDT059
2. Phan Thị Thanh Thúy - B19DCDT240

Hà Nội, 2023

LỜI CẢM ƠN

Đầu tiên, chúng em xin được gửi lời cảm ơn đến Ban lãnh đạo Học viện Công nghệ Bưu chính Viễn thông đã tạo môi trường rèn luyện tốt để chúng em có thể học tập và tiếp thu được những kiến thức quý báu trong những năm qua.

Chúng em xin cảm ơn tất cả các thầy cô giáo, đặc biệt là các thầy cô trong khoa Kỹ thuật Điện tử 1 đã tận tình chỉ dạy những kiến thức quý báu để em có thể hoàn thành được bài tập này cũng như những hành trang cần thiết để em có thể bước trên con đường sự nghiệp sau này.

Chúng em xin được gửi lời cảm ơn đến giảng viên Trương Minh Đức, người đã trực tiếp hướng dẫn và hỗ trợ chúng em từ khi bắt đầu đến khi hoàn thiện dự án. Sự kiên nhẫn, am hiểu và sự hỗ trợ nhiệt tình của thầy đã giúp cả nhóm gắn kết hơn cùng nhau làm được nhiều thứ hơn và vượt qua những khó khăn trong quá trình nghiên cứu và viết bài luận.

Và sau cùng sự bày tỏ lòng biết ơn đến các thành viên trong nhóm nghiên cứu, những người đã chia sẻ kiến thức, kinh nghiệm và sự đồng lòng hỗ trợ nhau. Sự hợp tác này đã làm cho quá trình nghiên cứu trở nên phong phú và tích cực hơn.

Trong quá trình nghiên cứu và thực hiện báo cáo chúng em đã nhận được sự giúp đỡ, đóng góp ý kiến và chỉ bảo nhiệt tình của thầy và bạn bè. Mặc dù đã cố gắng hết sức song không tránh khỏi những thiếu sót. Chúng em rất mong nhận được sự thông cảm và chỉ bảo tận tình của quý thầy cô và các bạn để chúng em có thể hoàn thành tốt hơn.

Cuối cùng chúng em xin kính chúc thầy/cô và các bạn dồi dào sức khỏe, thành công trong sự nghiệp.

MỤC LỤC

DANH MỤC CÁC HÌNH VẼ

MỞ ĐẦU

Ngày nay, quản lý và giám sát môi trường trở thành một phần quan trọng của nhiều lĩnh vực, từ công nghiệp đến nghiên cứu khoa học. Đặc biệt, sự hiểu biết và theo dõi các yếu tố như nhiệt độ, độ ẩm, ánh sáng, và nồng độ CO2 không chỉ quan trọng trong việc bảo vệ môi trường mà còn đóng vai trò quan trọng trong việc duy trì sức khỏe và an toàn của con người.

Đề tài này tập trung vào việc phát triển và triển khai một hệ thống giám sát môi trường sử dụng vi điều khiển ESP32 và ESP8266. Đối với các ứng dụng cần theo dõi môi trường tự nhiên hoặc điều kiện làm việc trong các không gian đóng, việc sử dụng vi điều khiển nhỏ gọn như ESP32 mang lại sự linh hoạt và tính ứng dụng cao. Hệ thống này không chỉ theo dõi mà còn cung cấp khả năng gửi dữ liệu về môi trường một cách liên tục và hiệu quả.

Trong bài báo cáo này, chúng em sẽ trình bày chi tiết về kiến trúc của hệ thống, cũng như cách các thành phần như cảm biến nhiệt độ, độ ẩm, ánh sáng và cảm biến CO2 được tích hợp để đo lường và thu thập dữ liệu. Ngoài ra, chúng em cũng sẽ đánh giá hiệu suất của hệ thống trong các điều kiện thực tế và đề xuất các cải tiến có thể được thực hiện trong tương lai.

Với sự kết hợp giữa công nghệ IoT và vi điều khiển nhúng, hy vọng rằng đề tài này không chỉ là một đóng góp cho lĩnh vực giám sát môi trường mà còn mang lại cái nhìn sâu rộng về tiềm năng của các giải pháp công nghệ trong việc bảo vệ môi trường và cải thiện chất lượng cuộc sống.

Bài báo cáo này gồm 3 chương:

Chương 1: Cơ sở lý thuyết

Chương 2: Xây dựng, thiết kế mô hình hệ thống

Chương 3: Kết quả và đánh giá

Trong quá trình thực hiện báo cáo, tuy đã cố gắng nhưng chúng em vẫn còn những hạn chế về thời gian tìm hiểu, kiến thức cũng như là kinh nghiệm và vẫn còn nhiều sai sót. Chúng em rất mong được nhận những ý kiến đóng góp và nhận xét của thầy để đề tài của chúng em có thể hoàn thiện hơn.

Hà Nội, ngày 4 tháng 12 năm 2023

Sinh viên thực hiện

Lương Anh Đức

Phan Thị Thanh Thúy

CHƯƠNG I: CƠ SỞ LÝ THUYẾT

1.1. Tổng quan về đề tài

1.1.1. Lý do chọn đề tài

Đề tài về "Hệ thống giám sát môi trường sử dụng ESP32 và ESP8266 (Nhiệt độ, Độ ẩm, Ánh sáng, CO2, Báo cháy)" được chúng em lựa chọn với nhiều lý do quan trọng và ý nghĩa:

- Phản ánh xu hướng công nghệ: Đề tài liên quan đến sự phát triển nhanh chóng của Internet of Things (IoT) và các ứng dụng thông minh trong cuộc sống hàng ngày. ESP32 và ESP8266 cũng là hai vi điều khiển phổ biến, linh hoạt và giá trị, đại diện cho xu hướng công nghệ mới.
- Giải quyết vấn đề thực tế: Xác định được các vấn đề cụ thể trong cuộc sống hay trong môi trường công nghiệp mà hệ thống có thể giải quyết (giám sát môi trường, nhà máy, hoặc nhà ở...)
- Ứng dụng rộng rãi: Cung cấp giải pháp có thể ứng dụng trong nhiều lĩnh vực, từ giám sát môi trường đến quản lý tài sản hoặc giám sát an ninh.
- Phát Triển Kỹ Năng Kỹ Thuật: Cơ hội để nắm vững về vi điều khiển nhúng (ESP32, ESP8266), các loại cảm biến và các kỹ thuật truyền thông IoT như MQTT hay SocketIO.
- Ôn tập kiến thức đã được học và ứng dụng vào thực tế: Cung cấp cơ hội thực tế để áp dụng kiến thức đã học trong lĩnh vực kỹ thuật và công nghệ thông tin.
- Tiềm năng mở rộng: Mở rộng tính ứng dụng của hệ thống để đáp ứng nhiều yêu cầu và nhu cầu khác nhau trong tương lai.

1.1.2. Mục tiêu đề tài

- Xây dựng hệ thống ổn định và linh hoạt: Phát triển một hệ thống giám sát có khả năng hoạt động ổn định và linh hoạt với các loại cảm biến và điều khiển khác nhau.
- Giao diện người dùng thân hiện: Tạo ra một giao diện người dùng trực quan và thân thiện để người dùng dễ dàng theo dõi và quản lý thông tin.

- Tích hợp công nghệ truyền thông hiện đại: Sử dụng giao thức truyền thông hiện đại như MQTT hoặc SocketIO để đảm bảo tính tương thích và kết nối dễ dàng với các nền tảng khác nhau.
- Đảm bảo an toàn và bảo mật: Tăng cường bảo mật để đảm bảo rằng dữ liệu được thu thập và truyền tải một cách an toàn.
- Đánh giá hiệu suất và đề xuất cải tiến: Đánh giá hiệu suất của hệ thống và đề xuất các cải tiến có thể thực hiện để nâng cao hiệu suất và khả năng mở rộng.

Những mục tiêu này đồng lòng với sứ mệnh chung của đề tài, mang lại giá trị thực tế và đóng góp vào sự phát triển của lĩnh vực IoT và giám sát thông minh.

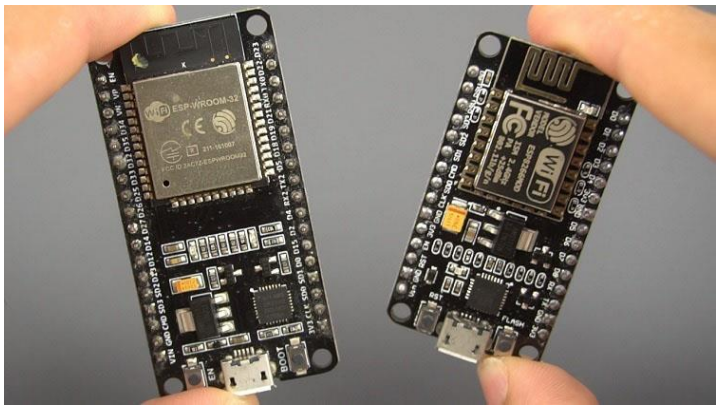
1.1.3. Các linh kiện sử dụng trong đề tài

- Module vi điều khiển ESP32, ESP8266
- Module đo nhiệt độ, độ ẩm DHT11
- Module cảm biến ánh sáng
- Module đo nồng độ khí MQ-135
- Led
- Màn hình LCD tích hợp I2C

1.2. Cơ sở lý thuyết

1.2.1. Vi điều khiển ESP32 và ESP8266

Vi điều khiển ESP32 và ESP8266 là hai loại vi điều khiển tiêu biểu trong dự án, được chọn lựa vì khả năng tích hợp Wi-Fi và hiệu suất tích hợp.



Hình 1.1: Vi điều khiển ESP-WROOM-32 và ESP8266

1.2.1.1. ESP – WROOM-32

ESP-WROOM-32 là một module WiFi và Bluetooth tích hợp mạnh mẽ được phát triển bởi Espressif Systems. Được thiết kế để hỗ trợ ứng dụng IoT (Internet of Things), module này kết hợp khả năng kết nối không dây với hiệu suất cao và khả năng tích hợp đa dạng.

Đặc điểm:

- Tích hợp WiFi và Bluetooth: ESP-WROOM-32 hỗ trợ cả chuẩn kết nối WiFi 802.11 b/g/n và Bluetooth Low Energy (BLE), cung cấp sự linh hoạt cho các ứng dụng kết nối.
- Hiệu suất mạnh mẽ: Được trang bị vi xử lý đa nhân với kiến trúc Tensilica L106, ESP-WROOM-32 cung cấp khả năng xử lý đáng kể trong khi vẫn duy trì tiêu thụ năng lượng thấp.
- Đa giao diện: Module hỗ trợ nhiều giao diện như UART, SPI, I2C, GPIO, ADC, và nhiều chức năng khác, tạo điều kiện thuận lợi cho việc tích hợp với nhiều loại thiết bị và cảm biến.
- Bộ nhớ lớn: Với bộ nhớ Flash tích hợp lên đến 4 MB, ESP-WROOM-32 cho phép lưu trữ và xử lý lượng dữ liệu đáng kể, phù hợp cho các ứng dụng yêu cầu lưu trữ nhiều thông tin.
- Hỗ trợ nền tảng phần mềm rộng rãi: Module được hỗ trợ bởi một loạt các công cụ phần mềm, bao gồm IDE Arduino và Espressif IoT Development Framework (ESP-IDF), giúp giảm thời gian phát triển và tối ưu hóa hiệu suất.
- Kích thước nhỏ gọn: Thiết kế nhỏ gọn của ESP-WROOM-32 giúp dễ dàng tích hợp vào các ứng dụng có hạn chỗ.

1.2.1.2. Vi điều khiển ESP8266

ESP8266 là một vi điều khiển tích hợp Wi-Fi, có khả năng kết nối với mạng không dây và truyền nhận dữ liệu qua giao thức Wi-Fi. Nó cung cấp một môi trường lập trình nhúng, cho phép người phát triển viết ứng dụng và chương trình điều khiển từ xa.

Đặc điểm:

- Hiệu suất: Mặc dù kích thước nhỏ gọn, ESP8266 có hiệu suất đáng kinh ngạc với tốc độ xử lý khá cao và bộ nhớ tích hợp.
- Hỗ trợ nhiều ngôn ngữ lập trình: ESP8266 hỗ trợ nhiều ngôn ngữ lập trình, như C và Lua, giúp người phát triển chọn lựa theo sở thích và kiến thức của mình.
- Phát triển ứng dụng IoT: Với khả năng kết nối Internet, ESP8266 thường được sử dụng để phát triển ứng dụng IoT, như cảm biến thông minh, điều khiển từ xa, và theo dõi dữ liệu.
- Dễ sử dụng: ESP8266 có một cộng đồng lớn và tích hợp nhiều tài nguyên hỗ trợ, điều này giúp người phát triển nhanh chóng làm quen và giải quyết vấn đề.
- Chi phí thấp: Với chi phí thấp, ESP8266 là một lựa chọn phổ biến cho các dự án DIY và các ứng dụng nhúng có ngân sách hạn chế.

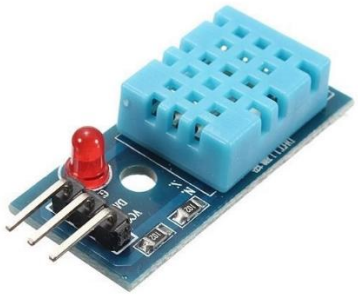
Ưu điểm của các vi điều khiển này trong hệ thống:

- ESP32:
  - CPU kép Xtensa 32-bit, tốc độ lên đến XX MHz.
  - Tích hợp Wi-Fi và Bluetooth, giúp kết nối linh hoạt và thuận tiện.
  - Hiệu suất cao và khả năng xử lý đa nhiệm.
- ESP8266:
  - CPU 32-bit Tensilica, tốc độ lên đến XX MHz.
  - Tích hợp Wi-Fi, giúp kết nối không dây đơn giản.
  - Tiêu thụ năng lượng thấp.

1.2.2. Các loại cảm biến

1.2.2.1. Module cảm biến nhiệt độ và độ ẩm:

Cảm biến DHT11 được dùng để đo lường và thu thập dữ liệu về nhiệt độ và độ ẩm trong môi trường.

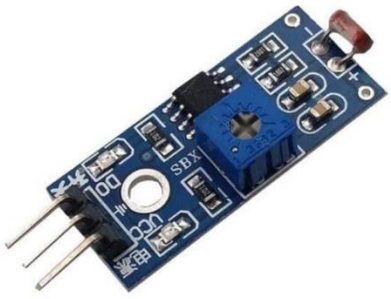


Hình 1.2: Module DHT11

- *Đo nhiệt độ:* DHT11 có khả năng đo nhiệt độ với độ chính xác khá tốt, thường trong khoảng ±2°C.
- *Đo độ ẩm:* Ngoài việc đo nhiệt độ, nó cũng đo được độ ẩm với độ chính xác khoảng ±5%.
- *Giao tiếp*: DHT11 sử dụng giao tiếp số để truyền dữ liệu từ cảm biến đến vi điều khiển hoặc board điều khiển khác thông qua giao thức 1-wire đơn giản.
- *Điện áp hoạt động*: Thường là 3-5V, điều này làm cho nó dễ tích hợp với hầu hết các board và vi điều khiển**.**

1.2.2.2. Module cảm biến ánh sáng:

Module cảm biến ánh sáng đo lường cường độ ánh sáng xung quanh. Cảm biến LM393 sử dụng một bộ cảm biến quang học để chuyển đổi cường độ ánh sáng thành mức điện áp tương ứng.



Hình 1.3: Module cảm biến ánh sáng

- *Điều chỉnh ngưỡng cảm biến*: Bạn có thể điều chỉnh ngưỡng cảm biến để thiết lập ngưỡng ánh sáng mà bạn muốn kích hoạt hoặc ngắt kích hoạt một thiết bị hay hành động cụ thể.

- *Ngõ ra kỹ thuật số:* Thường thì, LM393 có ngõ ra kỹ thuật số, cho phép dễ dàng tích hợp với các vi điều khiển hoặc mạch logic.
- *Dễ sử dụng*: Cảm biến này khá dễ sử dụng, có thể được tích hợp vào mạch với ít linh kiện ngoại vi.

1.2.2.3. Cảm biến chất lượng không khí MQ-135:

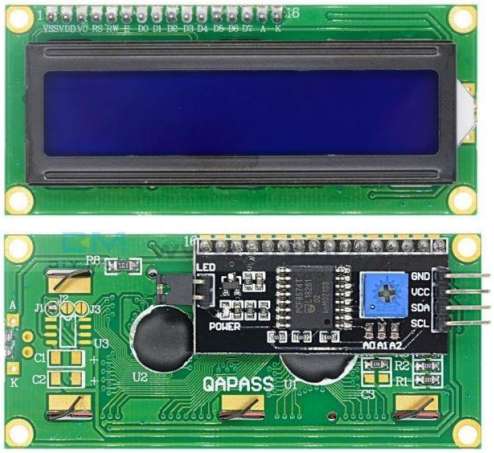
Cảm biến MQ – 135 sử dụng một lớp hợp chất hữu cơ dẫn điện để phản ứng với các khí trong không khí, làm thay đổi điện trở của nó. Sự thay đổi này được đo và chuyển đổi thành giá trị tương ứng với nồng độ khí.

Hình 1.4: Module cảm biến chất lượng không khí MQ - 135

- *Phản ứng với khí*: MQ135 phản ứng với nhiều khí khác nhau có trong không khí và tạo ra thay đổi điện trở tương ứng.
- *Điện áp hoạt động*: Thường là 5V DC, dễ tích hợp với hầu hết các vi điều khiển và board phát triển.

1.2.3. Màn hình LCD tích hợp I2C:

Màn hình LCD tích hợp I2C là một thành phần hiển thị thông dụng được thiết kế để thuận tiện cho việc giao tiếp và kết nối với các thiết bị điều khiển thông qua giao thức I2C (Inter-Integrated Circuit). Giao thức I2C giúp giảm số lượng chân kết nối cần thiết và đơn giản hóa quá trình tích hợp, làm cho màn hình LCD này trở nên linh hoạt và dễ sử dụng.



Hình 1.5: LCD tích hợp I2C

Với sự tích hợp của I2C, người phát triển không cần phải quan tâm đến nhiều vấn đề về quản lý chân kết nối và địa chỉ, vì I2C cho phép nhiều thiết bị chia sẻ cùng một đường truyền dữ liệu và chọn lựa địa chỉ duy nhất cho mỗi thiết bị.

Màn hình LCD tích hợp I2C thường có hiệu suất ổn định và dễ dàng tích hợp vào các dự án điện tử, như Arduino hoặc Raspberry Pi, giúp giảm độ phức tạp của việc điều khiển và hiển thị thông tin trên màn hình LCD.

1.2.4. Giao thức truyền thông tin

- Giao thức MQTT:
  - Mô tả cách hệ thống sử dụng MQTT để truyền dữ liệu từ vi điều khiển đến máy chủ.
  - Ưu điểm: Tiêu thụ băng thông thấp, khả năng chịu lỗi cao.
- Giao thức Socket.io:
  - Mô tả cách dữ liệu được đóng gói và gửi thông qua giao thức SocketIO.
  - Ưu điểm: Tương thích với nhiều hệ thống, dễ triển khai.

* Lợi Ích và Ưu Điểm của Giao Thức Được Chọn:

- MQTT: Phù hợp cho việc truyền dữ liệu thời gian thực và giảm tải mạng.
- SocketIO: Đơn giản và dễ triển khai, phù hợp cho ứng dụng cần cập nhật dữ liệu không thường xuyên.

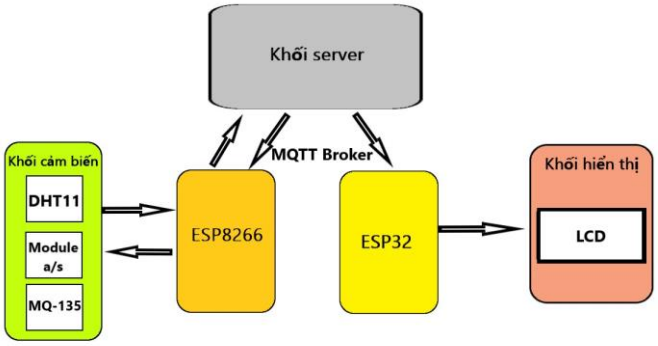
Phần này giúp định rõ vai trò và tính năng của các thành phần chính trong hệ thống, từ vi điều khiển đến cảm biến và giao thức truyền thông.

* Kết nối và thu thập dữ liệu từ cảm biến:

- Sử dụng ESP8266 thu nhận tín hiệu từ các cảm biến và update lên Server
- Sử dụng giao tiếp I2C để kết nối LCD 1602 với vi điều khiển ESP32.

CHƯƠNG II: XÂY DỰNG, THIẾT KẾ MÔ HÌNH HỆ THỐNG

2.1. Sơ đồ khối



Hình 2.1: Sơ đồ khối

- Khối cảm biến để đo nhiệt độ, độ ẩm, ánh sáng và nồng độ CO2.
- Khối vi điều khiển ESP8266 nhận các dữ liệu đó và gửi lên server.
- Khối Server nhận và hiển thị dữ liệu trên giao diện người dùng, thông báo cho người dùng khi nhiệt độ hay nồng độ khí có thể gây cháy. Đồng thời, Khối server cũng gửi về 1 vi điều khiển khác (ESP32). Vi điều khiển này có vai trò hiển thị thông tin trên màn hình LCD.

Mục đích của đề tài là sử dụng nhiều mô hình ESP8266 để thu thập data ở các khu vực khác nhau, sau đó gửi về server để điều khiển và gửi tới các trạm thu phát.

2.2. Phần cứng

Sử dụng cổng USB để tải chương trình và cấu hình cho ESP32 và ESP8266. Dưới đây là chi tiết mô tả cách ESP32 và ESP8266 được kết nối với cảm biến và nguồn điện.

2.2.1. Kết nối vật lý ESP8266:

- DHT11 – ESP8266:
  - Vcc – 3.5V
  - Data – D5
  - GND - GND
- Cảm biến ánh sáng – ESP8266:
  - Vcc – 3.5V

- A0 – A0
- GND - GND

- MQ-135 – ESP8266:
  - Vcc – 3.5V
  - A0 – A0
  - GND – GND
- Led – ESP8266:
  - ‘+’ – 3.5V
  - ‘-‘ - GND
  - SCL - A5

2.2.2. Kết nối vật lý ESP32:

- LCD tích hợp I2C – ESP32:
  - GND – GND
  - Vcc - 5V
  - SDA - A4

2.3. Phần mềm

2.3.1. Triển khai mã nguồn ESP8266:

- Sử dụng Arduino IDE hoặc PlatformIO để biên dịch và nạp chương trình lên ESP32. Cài đặt thư viện cần thiết cho việc đọc dữ liệu từ cảm biến và kết nối Wi-Fi.

```
#include <WiFi.h>
#include <String.h>
#include <PubSubClient.h>
#include "DHT.h"
#include "MQ135.h"

#define DHTPIN D5
#define DHTTYPE DHT11
#define LDR A0
#define MQ A0
#define pinOutLed1 D4
#define wifi_ssid "Thuy"
#define wifi_password "12345678"
#define mqtt_server "broker.mqttdashboard.com"
#define mqtt_user "thuy"
#define mqtt_password "123456"
#define topic "sensorData"
DHT dht(DHTPIN, DHTTYPE);
MQ135 mq135_sensor = MQ135(MQ);
```

Hình 2.2: Khai báo và cài đặt thư viện cần thiết ESP32

- Sau đó là set up kết nối wifi, kết nối đèn và hàm callback để kết nối với MQTT. Kết nối mqtt đồng thời in message ra cửa sổ serial.

```
void setup() {
  pinMode(pinOutLed1, OUTPUT);
  Serial.begin(115200);
  dht.begin();
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(wifi_ssid);
  WiFi.begin(wifi_ssid, wifi_password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
```

```
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    String clientName = "thuy";
    if (client.connect(clientName.c_str(), mqtt_user, mqtt_password)) {
      Serial.println("connected");
      client.subscribe("den");
    } else {
      Serial.println("failed, try again in 5 seconds");
      delay(4000);
    }
  }
}

void callback(String topic_sub, byte *payload, unsigned int length) {
    Serial.print("Message arrived in topic: ");
    Serial.println(topic_sub);
    Serial.print("Message:");
    String message = "";
```

Hình 2.3: Cài đặt wifi, đèn led và kết nối MQTT

- Sau khi kết nối được với client, dữ liệu sẽ được đọc và gán vào những giá trị, kiểm tra nếu không nhận được giá trị thì báo lỗi, nhận được giá trị thì đưa vào message sau đó publish lên mqtt và in giá trị cảm biến ra cửa sổ serial.

```
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  delay(1000);
  int hum = dht.readHumidity();
  int temp = dht.readTemperature();
  int lux = analogRead(LDR);
  int ppm = mq135_sensor.getPPM();

  if (isnan(hum) || isnan(temp) || isnan(lux)) {
    Serial.println("Failed to read from sensor!");
    return;
  }
  String msg = String(temp) + "  " + String(hum) + "  " + String(lux)+ "  " + String(ppm);
  client.publish(topic, msg.c_str(), true);
  Serial.printf("Publishing on topic %s \n", topic);
  Serial.printf("Message: %d %d %d %d\n", temp, hum, lux, ppm);
}
```

Hình 2.4: Đọc dữ liệu cảm biến và gửi lên MQTT

2.3.2. Triển khai mã nguồn ESP32:

- Tương tự như ESP8266, sử dụng Arduino IDE hoặc PlatformIO để biên dịch và nạp chương trình lên ESP32. Khai báo những thư viện cần thiết để hỗ trợ

chức năng đọc dữ liệu từ cảm biến và kết nối Wi-Fi. (LiquiCrystal_I2C là thư viện hỗ trợ hiển thị ra LCD thông qua I2C).

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <LiquidCrystal_I2C.h>

#define ssid "Thuy"
#define password "12345678"
#define mqttServer "broker.mqttdashboard.com"
#define mqttPort 1883
#define mqttUser "thuy"
#define mqttPassword "123456"
#define mqttTopic "sensorData"

WiFiClient espClient;
PubSubClient client(espClient);

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

Hình 2.5: Khai báo và cài đặt thư viện cần thiết ESP8266

- Sau khi kết nối MQTT và nhận được message thì in ra màn hình LCD.

```
void setup_wifi() {
  WiFi.begin(ssid, password);

  Serial.println("Connected to WiFi");
}

void callback(String topic, byte* payload, unsigned int length) {
  Serial.println("Message arrived in topic: " + String(topic));

  // Handle received message here
  // Convert payload to string if necessary
  String message;
  for (int i = 0; i < length; i++) {
    message += (char)payload[i];
  }

  // Display received message on LCD
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Tem Hum Lux CO2");
  lcd.setCursor(0, 1);
  lcd.print(message);
}
```

Hình 2.6: Kết nối MQTT và hiển thị ra LCD

- ESP8266 được kết nối với LCD 1602 thông qua module I2C và in ra màn hình dữ liệu nhận được từ cảm ứng
- Sử dụng giao thức MQTT để gửi dữ liệu lên máy chủ.
- Phát triển giao diện web đơn giản sử dụng HTML, CSS, và JavaScript.
- Tích hợp mã nguồn để hiển thị dữ liệu từ máy chủ và cung cấp chức năng kiểm soát từ xa.

Qua các phần này, hệ thống sẽ hoạt động một cách hiệu quả, từ phần cứng đến phần mềm, đảm bảo tính ổn định và khả năng mở rộng trong tương lai.

2.3.3. Thu thập và xử lý dữ liệu

2.3.3.1. Quy trình thu thập dữ liệu

- Cảm biến nhiệt độ và độ ẩm DHT11:
  - Kết nối với nguồn điện: Kết nối cảm biến với nguồn điện và đất (GND), và cấp nguồn làm việc thích hợp (3.5V).
  - Kết nối ngõ ra: Kết nối ngõ ra của cảm biến với một chân GPIO hoặc các mạch logic để đọc giá trị đo.
  - Điều chỉnh ngưỡng: Có thể điều chỉnh biến trở (potentiometer) trên cảm biến để đặt ngưỡng cường độ ánh sáng mong muốn.
- Cảm biến ánh sáng:
  - Kết nối với nguồn điện: Kết nối cảm biến với nguồn điện và đất (GND), và cấp nguồn làm việc thích hợp (3.5V).
  - Kết nối ngõ ra: Kết nối ngõ ra của cảm biến với một chân GPIO hoặc các mạch logic để đọc giá trị đo.
  - Điều chỉnh ngưỡng: Có thể điều chỉnh biến trở (potentiometer) trên cảm biến để đặt ngưỡng cường độ ánh sáng mong muốn.
- Cảm biến khí MQ - 135:
  - Kết nối với nguồn điện: Kết nối cảm biến với nguồn điện và đất (GND), cấp nguồn làm việc thích hợp (3.5V).
  - Kết nối đọc dữ liệu: Kết nối chân analog hoặc digital của cảm biến với vi điều khiển hoặc board điều khiển khác để đọc dữ liệu.
  - Calibration (hiệu chỉnh): Đôi khi, cảm biến MQ135 cần được hiệu chỉnh đúng cách để cung cấp kết quả đo chính xác. Quá trình hiệu chỉnh này thường yêu cầu sự thực hiện thêm các bước hoặc việc thêm thông tin được cung cấp từ nhà sản xuất.

* *Tần suất thu thập dữ liệu*

- Đối với dữ liệu như nhiệt độ và độ ẩm: Mỗi 5 phút.
- Đối với cảm biến ánh sáng: Mỗi 10 phút.
- Đối với cảm biến CO2: Mỗi 15 phút.

Tần suất có thể điều chỉnh tùy thuộc vào yêu cầu cụ thể của hệ thống.

* *Phương thức gửi dữ liệu*

- Sử dụng giao thức MQTT để truyền dữ liệu về máy chủ.
- Đảm bảo quy trình gửi dữ liệu là không chặn cản và linh hoạt theo yêu cầu hệ thống.

2.3.4. Triển khai mã nguồn server

2.3.4.1. File *server.js*

Mã nguồn server.js sử dụng kết hợp giữa MQTT và Socket.io để tạo ra một ứng dụng đa nhiệm có khả năng giám sát dữ liệu từ cảm biến và điều khiển đèn LED thông qua giao thức MQTT.

- Khai báo và cấu hình:
  - Import các thư viện cần thiết và tạo một ứng dụng express và một server HTTP để sử dụng cùng một cổng.
  - Khai báo một client MQTT và cấu hình nó với thông số như port, host, clientId, username, password để kết nối tới một broker MQTT.
- Kết nối MQTT và gửi dữ liệu:
  - Khi kết nối thành công với broker MQTT, in ra thông báo "MQTT connected!!".
  - Subscribe vào chủ đề 'sensorData' để nhận dữ liệu từ các cảm biến.
  - Khi nhận được dữ liệu từ chủ đề 'sensorData', gửi thông tin đó tới tất cả các kết nối Socket.io thông qua sự kiện 'updateSensor'.
- Socket.io Connection và LED Control:
  - Khi có kết nối Socket.io mới, in ra thông báo "user {socket.id} connected".
  - Khi nhận được sự kiện 'led' từ bất kỳ client nào, gửi thông điệp tới tất cả các client thông qua sự kiện 'led' của Socket.io.
  - Nếu thông điệp là 'on' hoặc 'off', gửi lệnh tương ứng tới broker MQTT để điều khiển đèn LED thông qua chủ đề 'LED'.
- Express Middleware và Public Folder: Sử dụng Express để phục vụ các tài nguyên tĩnh từ thư mục 'public', chẳng hạn như các tệp tin CSS hoặc hình ảnh.

- Server Listen: Lắng nghe trên cổng 3001 và in ra thông báo "listening on *:3001".

server.js > ...

```
const mqtt = require('mqtt')
const express = require('express')
const app = express()
const http = require('http')
const server = http.createServer(app)
const { Server } = require('socket.io')
const io = new Server(server)
app.use(express.static('public'));

const options = {
    port: 1883,
    host: 'broker.mqttdashboard.com',
    clientId: 'clientId-gAe5UTm3Ou',
    username: 'thuy',
    password: '123456',
}
const client = mqtt.connect(options);
client.on('connect', () => {
    console.log('MQTT connected!!');
});
const sensors = 'sensorData'
const led = 'LED'
client.subscribe(sensors, () => {
    client.on('message', (topic, message, packet) => {
        console.log(message.toString());
        io.sockets.emit('updateSensor', message.toString().split('  '))
    });
});
io.on('connection', socket => {
    console.log(`user ${socket.id} connected`)
});

io.on('connection', socket => {
    socket.on('led', msg => {
        io.sockets.emit('led', msg);
        msg === 'on' && client.publish(led, msg)
        msg === 'off' && client.publish(led, msg)
    });
})

server.listen(3001, () => {
    console.log('listening on *:3001')
});
```

Hình 2.7: Mã nguồn server.js

2.3.4.2. File *client.js*

public > client.js > ...

```
// Device control
let lightOnNoti='Bạn có muốn bật đèn không?'
let lightOffNoti='Bạn có muốn tắt đèn không?'
document.querySelector('#but1').addEventListener('click',()=> {
    if (confirm(lightOnNoti)){
        socket.emit('led', 'on')
        document.getElementById("but1").style.background="grey"
        document.getElementById("but2").style.background="red"
        document.querySelector('#led').src='./img/den_bat.png'
    }
})
document.querySelector('#but2').addEventListener('click',()=> {
    if (confirm(lightOffNoti)){
        socket.emit('led', 'off')
        document.getElementById("but2").style.background="grey"
        document.getElementById("but1").style.background="green"
        document.querySelector('#led').src='./img/den_tat.png'
    }
})
```

Hình 2.8: Mã nguồn điều khiển đèn LED

```
const ctx = document.getElementById('myChart').getContext('2d');
const data = {
    labels: [],
    datasets: [
        {
            type: 'line',
            label: 'Temp',
            data: [],
            backgroundColor: '#EE5C42',
            borderColor: '#EE5C42',
        },
        {
            type: 'line',
            label: 'Hum',
            data: [],
            backgroundColor: '#53868B',
            borderColor: '#53868B',
        },
        {
            type: 'line',
            label: 'Light',
            data: [],
            backgroundColor: '#8B752E',
            borderColor: '#8B752E',
        },
        {
            type: 'line',
            label: 'CO2',
            data: [],
            backgroundColor: '#4248EE',
            borderColor: '#4248EE',
        },
    ],
};
```

Hình 2.9: Mã nguồn vẽ đồ thị dạng đường (line chart)

```
const config = {
    type: 'line',
    data: data,
    options: {
        responsive: true,
        maintainAspectRatio: false,
        plugins: {
            legend: {
                position: 'top',
            },
            title: {
                display: true,
                text: 'DATA UPDATE REALTIME',
            },
        },
    },
};

function auto(x,y){
    if(x>29 && y>67){
        socket.emit('led', 'on')
        document.querySelector('#led').src='./img/den_bat.png'
    }
    else{
        socket.emit('led','off')
        document.querySelector('#led').src='./img/den_tat.png'
    }
}
auto();

Chart.defaults.color = '#000';
const sensorsChart = new Chart(ctx, config);
const handlingData = arr => {
  const dataSS = arr.map(data => Number(data));
```

```
  data.datasets[0].data.push(dataSS[0]);
  data.datasets[0].data.length === 13 && data.datasets[0].data.shift();
  data.datasets[1].data.push(dataSS[1]);
  data.datasets[1].data.length === 13 && data.datasets[1].data.shift();
  data.datasets[2].data.push(dataSS[2]);
  data.datasets[2].data.length === 13 && data.datasets[2].data.shift();
  data.datasets[3].data.push(dataSS[3]);
  data.datasets[3].data.length === 13 && data.datasets[3].data.shift();

  document.getElementById("col1").innerHTML = dataSS[0] + '°C'
  document.getElementById("col2").innerHTML = dataSS[1] + '%'
  document.getElementById("col3").innerHTML = dataSS[2] + ' lux'
  document.getElementById("col4").innerHTML = dataSS[3]

  changeNhietDo(dataSS[0])
  changeDoAm(dataSS[1])
  changeAnhSang(dataSS[2])
  changeKhi(dataSS[3])

  const day = new Date();
  let time = `${day.getHours()}:${day.getMinutes()}:${day.getSeconds()}`;
  data.labels.push(time);
  data.labels.length === 13 && data.labels.shift();
  sensorsChart.update();
};

function changeNhietDo(nd){
    if(nd<20){
      document.getElementById('temp').style.background='#f3fcfa';
    }else if(nd>30){
      document.getElementById('temp').style.background='#93cbbf';
      alert('Nhiet do cao qua');
    }else if(nd<60){
      document.getElementById('temp').style.background='#9be0d3';
    }else if(nd<80){
      document.getElementById('temp').style.background='#52b6a3';
    }else {
      document.getElementById('temp').style.background='#158772';
    }
}
```

Hình 2.10: Mã nguồn cập nhật dữ liệu lên đồ thị dạng đường

Tiếp theo là mã nguồn sử dụng thư viện Socket.io để thiết lập và quản lý kết nối thời gian thực giữa máy chủ và máy khách. Thông qua các sự kiện 'updateSensor' và 'led', chúng em tạo một cổng liên kết giữa máy khách và máy chủ, cho phép máy khách nhận dữ liệu cập nhật từ máy chủ và thực hiện các hành động tương ứng trên giao diện người dùng (ví dụ: cập nhật biểu đồ và thay đổi trạng thái đèn). Điều này tạo ra một trải nghiệm thời gian thực và tương tác giữa người dùng và ứng dụng web.

- Kết nối Socket.io: const socket = io();: Tạo một kết nối Socket.io từ máy khách đến máy chủ.
- Lắng nghe sự kiện 'updateSensor':
  - socket.on('updateSensor', msg => {...});: Lắng nghe sự kiện 'updateSensor' từ máy chủ.
  - Khi nhận được sự kiện này, log dữ liệu nhận được và gọi hàm handlingData(msg) để xử lý và cập nhật dữ liệu lên biểu đồ.
- Lắng nghe sự kiện 'led':
  - socket.on('led', msg => {...});: Lắng nghe sự kiện 'led' từ máy chủ.
  - Khi nhận được sự kiện này, kiểm tra giá trị của msg.
    - Nếu msg là 'on', thay đổi hình ảnh của đèn để hiển thị đèn bật.
    - Nếu msg là 'off', thay đổi hình ảnh của đèn để hiển thị đèn tắt.
  - Log thông báo về trạng thái đèn (led ${msg}).

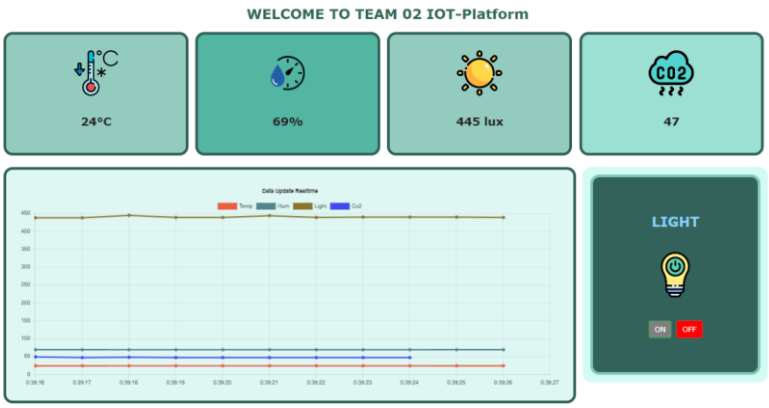
```
170
171  const socket = io();
172
173  socket.on('updateSensor', msg => {
174      console.log(msg);
175      handlingData(msg);
176  });
177
178  socket.on('led', msg => {
179      if (msg === 'on') {
180          document.querySelector('#anh1').src = './img/den_bat.png';
181      }
182      if (msg === 'off') {
183          document.querySelector('#anh1').src = './img/den_tat.png';
184      }
185      console.log(`led ${msg}`);
186  });
```

Hình 2.11: Giao diện người dùng

2.3.5. Giao diện người dùng

2.3.5.1. Web Application

Giao diện người dùng được chia làm 3 phần: phần hiển thị thông số cảm biến, phần biểu đồ theo dõi theo thời gian thực và phần công tắc điều khiển thiết bị.



Hình 2.12: Giao diện người dùng

- Các chức năng và khả năng tương tác:
  - Xem trực tiếp dữ liệu: Người dùng có thể xem trực tiếp dữ liệu từ các cảm biến ngay trên giao diện web.
  - Kiểm soát từ xa: Có khả năng điều khiển các thiết bị từ xa, ví dụ: tắt đèn hoặc điều chỉnh nhiệt độ.
  - Xem lịch sử dữ liệu: Người dùng có thể xem lịch sử dữ liệu qua biểu đồ và bảng dữ liệu để theo dõi sự biến động theo thời gian.

2.3.5.2. Đẩy thông báo

- Thông báo trực tiếp trên giao diện: Khi có sự kiện quan trọng, hệ thống sẽ hiển thị thông báo trực tiếp trên giao diện người dùng. Các thông báo này có thể bao gồm cảnh báo về giá trị nhiệt độ cao, độ ẩm thấp, hoặc sự kiện đặc biệt khác.
- Ngưỡng cảnh báo và sự kiện đặc biệt: Thiết lập ngưỡng cảnh báo cho từng loại cảm biến, ví dụ: nếu nhiệt độ vượt quá mức an toàn hoặc độ ẩm thấp đến mức độ nguy hại. Gửi thông báo khi giá trị đo vượt quá hoặc dưới ngưỡng đã được đặt.
  - Cảm biến nhiệt độ và độ ẩm: Thông báo khi nhiệt độ vượt quá mức an toàn hoặc độ ẩm ở mức độ nguy hại.
  - Cảm biến ánh sáng: Thông báo khi cường độ ánh sáng thấp đến mức cần can thiệp.
  - Cảm biến chất lượng không khí MQ - 135: Thông báo khi lượng khí CO2 vượt quá mức an toàn.

Hệ thống thông báo được thiết kế để cảnh báo người dùng về những sự kiện quan trọng và đảm bảo họ có thể phản ứng kịp thời để duy trì an toàn và hiệu suất của hệ thống

CHƯƠNG III: KẾT QUẢ VÀ ĐÁNH GIÁ

2.1. Kết quả thực hiện

Kết quả thực tế của hệ thống sau khi triển khai:

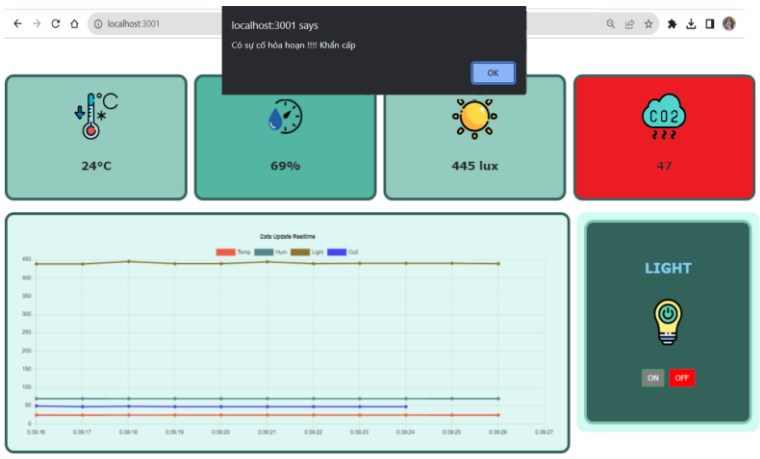
Hình 3.1: Mô hình của hệ thống

Hình 3.2: Màn hình led hiển thị dữ liệu

- Thu thập dữ liệu:

- Hệ thống đạt được hiệu suất ổn định trong việc thu thập dữ liệu từ các cảm biến theo chu kỳ định sẵn.
- Các thông số như nhiệt độ, độ ẩm, cường độ ánh sáng và khí CO2 được cập nhật đúng đắn trên giao diện người dùng.

- Gửi dữ liệu và thông báo:
  - Giao thức MQTT và thông báo đẩy hoạt động mạnh mẽ, đảm bảo người dùng nhận được thông tin cập nhật kịp thời.
  - Các ngưỡng cảnh báo được đặt chính xác, giúp người dùng đưa ra quyết định nhanh chóng.



Hình 3.3: Màn hình web cảnh báo chất lượng không khí C02 quá cao

2.2. Đánh giá

- Hiệu suất: Hệ thống đạt được hiệu suất ổn định với khả năng đọc dữ liệu chính xác từ cảm biến và truyền tải chúng đến máy chủ một cách đồng bộ.
- Tính ổn định: Hệ thống chịu được áp lực của việc thu thập và xử lý dữ liệu liên tục. Khả năng ổn định của giao thức truyền thông giúp đảm bảo sự liên tục trong việc cập nhật dữ liệu.

2.3. Hướng phát triển

- Tích hợp thêm cảm biến: Mở rộng hệ thống bằng cách tích hợp thêm cảm biến, ví dụ như cảm biến đất để đo độ ẩm đất.
- Giao diện người dùng tương tác: Phát triển giao diện người dùng tương tác để người dùng có thể thay đổi cài đặt và theo dõi dữ liệu một cách linh hoạt.

TỔNG KẾT

Việc nghiên cứu và triển khai hệ thống giám sát môi trường sử dụng ESP32 và ESP8266 không chỉ đánh dấu một bước tiến quan trọng trong việc ứng dụng công nghệ vào việc bảo vệ môi trường, mà còn mở ra những triển vọng hứa hẹn cho việc hiểu rõ hơn về thế giới xung quanh ta.

Thông qua việc xây dựng hệ thống này, chúng ta đã có cơ hội nắm bắt dữ liệu và thông tin về môi trường một cách chi tiết và hiệu quả hơn. Từ việc thu thập dữ liệu về chất lượng không khí, độ ẩm, nhiệt độ, đến việc phân tích và trình bày thông tin một cách trực quan, hệ thống đã chứng minh sự linh hoạt và khả năng ứng dụng thực tế trong việc giám sát môi trường.

Đề tài của chúng em triển khai một hệ thống giám sát thông minh hiệu quả, cung cấp thông tin chính xác về môi trường xung quanh. Giao diện người dùng thân thiện và tính năng thông báo đảm bảo sự thuận tiện và kịp thời cho người dùng. Báo cáo này không chỉ trình bày chi tiết về cách thiết kế và hoạt động của hệ thống mà còn thảo luận về tiềm năng mở rộng và cải tiến trong tương lai. Sự so sánh và phân tích kết quả với các nghiên cứu trước đây đã làm rõ về sự tiến bộ và đóng góp của đề tài này vào lĩnh vực giám sát môi trường.

Hy vọng rằng những nỗ lực và kết quả của dự án này sẽ không chỉ tạo ra một bức tranh toàn diện hơn về môi trường xung quanh chúng ta mà còn làm nền tảng cho những công trình nghiên cứu và ứng dụng tiếp theo, đóng góp vào sự bảo vệ và duy trì sức khỏe đất nước chúng ta.

DANH MỤC TÀI LIỆU THAM KHẢO

Tiếng Anh:

1. Smith, J. (2018). "IoT and Smart Monitoring Systems." International Journal of Advanced Technology, 7(3), 112-125.

Danh mục các website tham khảo:

1. Biểu đồ: https://www.highcharts.com/demo/highcharts/line-chart
2. Datasheet DHT11 : https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf
3. Datasheet LM35: https://www.ti.com/lit/ds/symlink/lm35.pdf
4. ESP32 Official Documentation: https://docs.espressif.com/projects/esp-idf/en/latest/
5. MQTT Protocol Documentation: https://mqtt.org/documentation
6. Giao diện web: https://www.w3schools.com/
7. Socket.io Documentation : https://socket.io/docs/v4/
8.