TRƯỜNG ĐẠI HỌC KHOA HỌC
ĐẠI HỌC HUẾ

KHOA CÔNG NGHỆ THÔNG TIN

BÀI TIỂU LUẬN

Đề tài:

GIÁM SÁT CHẤT LƯỢNG KHÔNG KHÍ VỚI ESP32 VÀ CẢM BIẾN MQ

PHÁT TRIỂN ỨNG DỤNG IOT - 2024-2025.2.TIN4024.005

GIẢNG VIÊN HƯỚNG DẪN: VÕ VIỆT DŨNG

Huế, tháng 4/202

Danh mục các từ viết tắt

CO: Khí carbon monoxit (Carbon Monoxide)

CO_2: Khí carbon dioxide (Carbon Dioxide)

PM2.5: Bụi mịn có đường kính ≤ 2,5 μm (Particulate Matter 2.5)

ESP32: Vi điều khiển 32-bit tích hợp WiFi & Bluetooth của hãng Espressif

IoT: Internet of Things – Mạng lưới vạn vật kết nối Internet

OLED: Organic Light-Emitting Diode – Đi-ốt phát quang hữu cơ (màn hình hiển thị)

ADC: Analog-to-Digital Converter – Bộ chuyển đổi tín hiệu tương tự sang số

UART: Universal Asynchronous Receiver/Transmitter – Giao tiếp nối tiếp bất đồng bộ

ppm: parts per million – đơn vị đo nồng độ khí (phần triệu)

$\mu g/m^3$: microgram trên mét khối – đơn vị đo mật độ bụi trong không khí

Mục lục

1. Phần mở đầu

Trong những năm gần đây, vấn đề ô nhiễm không khí ngày càng trở nên nghiêm trọng ở nhiều đô thị lớn tại Việt Nam như Hà Nội, TP. Hồ Chí Minh, Đà Nẵng... Theo thống kê từ Tổng cục Môi trường và các báo cáo của Tổ chức Y tế Thế giới (WHO), chỉ số bụi mịn PM2.5 và nồng độ các khí độc như CO, NO_2 thường xuyên vượt ngưỡng an toàn cho phép, ảnh hưởng trực tiếp đến sức khỏe của người dân, đặc biệt là trẻ em và người cao tuổi.

Từ thực tế đó, nhu cầu giám sát chất lượng không khí tại nhà, trường học, bệnh viện hay văn phòng làm việc trở nên rất cần thiết.

Hiện nay, với sự phát triển của công nghệ IoT, việc xây dựng các hệ thống theo dõi chất lượng không khí trở nên dễ tiếp cận hơn bao giờ hết. Các thiết bị như vi điều khiển ESP32, cảm biến khí MQ-135, cảm biến bụi PMS5003 có chi phí khá rẻ, dễ mua tại các cửa hàng linh kiện điện tử trong nước, nhưng vẫn đáp ứng tốt yêu cầu đo lường cơ bản.

Thêm vào đó, sử dụng mã nguồn mở Wokwi để mô phỏng phần cứng với ứng dụng Blynk giúp kết nối thiết bị với điện thoại qua Internet, giúp cho sinh viên có thể triển khai một hệ thống giám sát chất lượng không khí từ xa với chi phí thấp nhưng hiệu quả cao.

Với nhu cầu học tập, đề tài: “Giám sát chất lượng không khí với ESP32 và cảm biến MQ” là một chủ đề thiết thực, có tính ứng dụng cao trong đời sống. Dự án này này sử dụng cảm biến MQ-135 để đo nồng độ các khí (đặc biệt là khí CO), kết hợp với cảm biến PMS5003 để theo dõi nồng độ bụi PM2.5 là hai yếu tố ảnh hưởng trực tiếp đến chất lượng không khí. Các giá trị đo được sẽ được hiển thị trên màn hình OLED, giao diện web và truyền về nền tảng Blynk để người dùng theo dõi từ xa qua điện thoại.

Mục tiêu của bài tiểu luận là tìm hiểu cách hoạt động của hệ thống đo chất lượng không khí sử dụng ESP32, các loại cảm biến, lập trình mô phỏng trên Wokwi và tích hợp gửi dữ liệu lên Blynk. Từ đó, củng cố kiến thức về lập trình nhúng, kỹ thuật cảm biến và triển khai ứng dụng IoT thực tế trong đời sống.

2. Phần Giới thiệu

2.1. Giới thiệu chung

Internet of Things (IoT) là một mạng lưới kết nối các thiết bị như cảm biến, máy móc hoặc đồ dùng trong nhà với nhau thông qua Internet. Nhờ đó, chúng có thể thu thập dữ liệu, chia sẻ thông tin và điều khiển từ xa một cách thông minh [1]. Trong hệ thống IoT, các thiết bị được gắn vi mạch và cảm biến, có khả năng giao tiếp qua mạng. Ví dụ, một cảm biến không khí có thể gửi thông tin về chất lượng môi trường xung quanh lên điện thoại hoặc máy chủ để người dùng theo dõi. Hiện nay, IoT được ứng dụng rất nhiều trong đời sống như nhà thông minh, nông nghiệp thông minh, và đặc biệt là giám sát môi trường.

Đề tài này áp dụng mô hình IoT để đo và theo dõi hai yếu tố gây ô nhiễm không khí phổ biến là khí CO và bụi mịn PM2.5. Các cảm biến sẽ kết nối với vi điều khiển ESP32 – một bo mạch có hỗ trợ WiFi – nhằm truyền dữ liệu đến nền tảng Blynk. Nhờ đó, người dùng có thể xem thông tin chất lượng không khí từ xa ngay trên điện thoại của mình. Điều này rất tiện lợi cho việc theo dõi và bảo vệ sức khỏe hằng ngày.

2.2. Giới thiệu về các thành phần của hệ thống

- ESP32: Đây là một vi điều khiển tích hợp sẵn WiFi và Bluetooth, phù hợp cho các dự án IoT. ESP32 có khả năng xử lý mạnh mẽ với hai lõi 32-bit, tốc độ lên tới 240 MHz và bộ nhớ đủ lớn để lưu trữ chương trình [2]. Board ESP32 có nhiều chân giao tiếp như UART, I2C, ADC… giúp nó kết nối được với nhiều loại cảm biến khác nhau. Nhờ tích hợp WiFi, ESP32 có thể dễ dàng gửi dữ liệu lên mạng hoặc nhận lệnh điều khiển từ xa.

- Cảm biến MQ-135: Đây là cảm biến chuyên dùng để phát hiện các khí gây ô nhiễm trong không khí như amoniac (NH_3), khí CO, CO_2, khói, cồn (rượu) v.v. MQ-135 hoạt động dựa trên việc thay đổi điện trở khi tiếp xúc với các khí này. Trong không khí sạch, điện trở của cảm biến rất lớn, nhưng khi có khí độc, điện trở sẽ giảm xuống [3]. Cảm biến này cho ra tín hiệu analog (điện áp) tỷ lệ với nồng độ khí đo được. MQ-135 có ưu điểm giá rẻ, phản hồi nhanh, phù hợp với các ứng dụng đo chất lượng không khí trong nhà. Tuy nhiên, nếu muốn đo chính xác theo đơn vị ppm, cần phải hiệu chuẩn cẩn thận trước khi sử dụng.

- Cảm biến PMS5003: Đây là cảm biến đo bụi mịn trong không khí, hoạt động bằng nguyên lý laser để đếm hạt bụi. Không khí đi qua cảm biến, các hạt bụi sẽ làm tán xạ tia laser, từ đó thiết bị tính toán số lượng và kích thước hạt bụi, đặc biệt là các loại bụi PM1.0, PM2.5 và PM10 [4]. Cảm biến PMS5003 giao tiếp với vi điều khiển qua UART và gửi dữ liệu liên tục mỗi giây. PMS5003 có độ chính xác cao và được sử dụng phổ biến trong các thiết bị đo bụi chuyên nghiệp. Nhược điểm của nó là giá thành cao hơn và tiêu thụ điện năng nhiều hơn so với các cảm biến dùng LED hồng ngoại.

- Nền tảng Blynk: Blynk là một nền tảng giúp người dùng giám sát và điều khiển từ xa các thiết bị IoT qua điện thoại. Chỉ với thao tác kéo thả các widget trong ứng dụng Blynk, người dùng có thể tạo giao diện hiển thị các số liệu như: biểu đồ, đồng hồ, nút bấm… mà không cần tự viết ứng dụng di động [5].

- Hệ thống Blynk gồm ba phần chính:

1. ứng dụng trên điện thoại,
2. máy chủ Blynk
3. thư viện Blynk chạy trên vi điều khiển.

Trong đề tài này, Blynk được dùng để hiển thị các giá trị nồng độ khí CO và bụi PM2.5, giúp người dùng theo dõi chất lượng không khí bất kỳ lúc nào.

- Các linh kiện phần cứng khác:

+ **Điện trở:** Linh kiện dùng để hạn chế dòng điện chạy qua, nhằm bảo vệ các thành phần. Điện trở còn dùng để tạo cầu phân áp. Ví dụ, trong module MQ-135 có một biến trở (potentiometer) để điều chỉnh ngưỡng cảnh báo bằng cách thay đổi điện trở trên mạch so sánh.

+ **Đèn LED:** LED là loại diode phát sáng khi có dòng điện chạy qua. Đèn LED thường được dùng làm đèn báo trạng thái của mạch: chẳng hạn đèn nguồn sáng khi mạch có điện, hoặc đèn cảnh báo sáng khi nồng độ khí vượt ngưỡng cho phép.

+ **Nguồn cấp:** Hệ thống sử dụng nguồn một chiều 5V. ESP32 có thể lấy nguồn từ cổng USB máy tính hoặc nguồn ngoài (adapter, pin). Trên board ESP32 có sẵn mạch ổn áp hạ 5V xuống 3.3V để cung cấp điện ổn định cho chip vi điều khiển.

+ Breadboard và dây nối: Breadboard và dây nối giúp lắp ráp mạch linh hoạt mà không cần hàn. Các dây nối dùng để kết nối nhanh các chân của ESP32 với cảm biến MQ-135, PMS5003... thuận tiện cho việc thử nghiệm và điều chỉnh mạch.

+ **Màn hình OLED (giao tiếp I2C):** Màn hình OLED 0.96 inch sử dụng giao tiếp I2C (2 dây SDA và SCL) để hiển thị dữ liệu đo được từ cảm biến ngay trên thiết bị. Màn hình này kết nối với ESP32 qua hai chân I2C mặc định (GPIO21 cho SDA, GPIO22 cho SCL). Nhờ có OLED, người dùng có thể quan sát trực tiếp giá trị nồng độ CO và PM2.5 trên thiết bị mà không cần mở điện thoại hay ứng dụng.

3. Phần nội dung

3.1 Nguyên lý hoạt động

3.1.1. Nguyên lý hoạt động của ESP32

- ESP32 đóng vai trò là bộ não của hệ thống, đảm nhận việc đọc dữ liệu từ cảm biến và gửi dữ liệu lên máy chủ. Khi chạy, ESP32 thực thi chương trình đã được nạp bằng Arduino C++ với một vòng lặp liên tục. Trong mỗi vòng lặp, ESP32 thực hiện các công việc sau:

1. **Đọc tín hiệu cảm biến:** ESP32 sử dụng bộ ADC tích để đọc điện áp analog từ cảm biến MQ-135 (qua chân ADC GPIO34). Điện áp này tỷ lệ với nồng độ khí CO mà MQ-135 phát hiện được. Đồng thời, ESP32 nhận dữ liệu từ cảm biến bụi PMS5003 qua giao tiếp UART (cổng Serial). Cảm biến PMS5003 định kỳ gửi các gói dữ liệu chứa nồng độ PM1.0, PM2.5, PM10; chương trình trên ESP32 sẽ tách giá trị PM2.5 từ chuỗi dữ liệu nhận được để sử dụng.
2. **Kết nối WiFi và gửi dữ liệu:** Nhờ có module WiFi 2.4GHz tích hợp, ESP32 có thể kết nối trực tiếp đến mạng WiFi. Trong hàm khởi động (setup()), ESP32 được lập trình để kết nối WiFi bằng SSID (tên mạng) và password đã cấu hình. Sau khi vào Internet, ESP32 dùng thư viện Blynk để kết nối đến máy chủ đám mây của Blynk. Mỗi khi có dữ liệu cảm biến mới, ESP32 sẽ gửi dữ liệu này lên Blynk qua các Virtual Pin bằng lệnh Blynk.virtualWrite(). Thư viện Blynk đóng gói giá trị và truyền lên máy chủ Blynk qua kết nối TCP.
3. **Nhận lệnh từ xa (nếu có):** Mặc dù hệ thống chủ yếu gửi dữ liệu lên cho người dùng, ESP32 cũng có khả năng nhận lệnh điều khiển từ xa. Ví dụ, nếu người dùng nhấn một nút trên ứng dụng Blynk (trên điện thoại) để bật cảnh báo, máy chủ Blynk sẽ chuyển lệnh đó đến ESP32 qua kết nối WiFi. Thư viện Blynk trên ESP32 sẽ gọi hàm xử lý (callback) tương ứng do lập trình viên định nghĩa để thực thi lệnh, chẳng hạn bật đèn LED cảnh báo gắn trên thiết bị. (Trong phạm vi đề tài này, chức năng nhận lệnh từ xa chưa được sử dụng, nhưng nguyên lý của Blynk có hỗ trợ giao tiếp hai chiều như vậy.)
4. **Lặp vòng và kiểm tra:** ESP32 liên tục lặp lại các bước đọc và gửi dữ liệu nêu trên. Để tránh quá tải, chương trình thường chèn một khoảng trễ ngắn (vài trăm mili-giây đến vài giây) giữa các lần đọc/gửi. Ngoài ra, ESP32 có thể được lập trình để giám sát ngưỡng: ví dụ nếu nồng độ CO vượt mức cho phép, ESP32 có thể chủ động kích hoạt ngay một chân GPIO điều khiển còi hoặc đèn cảnh báo tại chỗ.

3.1.2. Nguyên lý hoạt động của cảm biến MQ-135

Hình 1: Mô-đun cảm biến khí MQ-135 (module cảm biến chất lượng không khí).

- Cảm biến MQ-135 là một loại cảm biến phổ biến dùng để đo chất lượng không khí tổng quát. Đây là một cảm biến có khả năng phát hiện nhiều loại khí gây ô nhiễm như amoniac (NH_3), khí nitơ oxit (NOx), cồn, benzen, khói, và cả CO_2. Trong một số trường hợp, cảm biến này cũng phản ứng được với khí CO nên có thể sử dụng để đánh giá mức độ ô nhiễm liên quan đến khí CO ở mức tương đối.

- Nguyên lý hoạt động của MQ-135 thuộc loại cảm biến điện trở hóa học. Cảm biến có hai thành phần chính: phần tử cảm biến và bộ phận đốt nóng bên trong. Khi hoạt động, bộ phận đốt nóng làm nóng lớp bán dẫn đến một nhiệt độ nhất định, giúp tăng khả năng phản ứng với khí. Khi các phân tử khí như CO, NH_3 hay khói bay vào bề mặt cảm biến, điện trở của lớp bán dẫn sẽ thay đổi. Cụ thể, nếu nồng độ khí tăng lên thì điện trở giảm xuống và ngược lại.

- Đầu ra của MQ-135 là điện áp analog. Vi điều khiển như ESP32 có thể đọc giá trị điện áp này qua chân ADC để đánh giá mức độ khí trong không khí.

- Tuy nhiên, MQ-135 không hiển thị giá trị ppm cụ thể mà chỉ cho biết sự biến động tương đối. Vì vậy, trong các ứng dụng thực tế nếu muốn đo chính xác nồng độ khí người dùng cần hiệu chuẩn cảm biến bằng cách đo trong môi trường sạch và so sánh với mẫu khí chuẩn. Còn trong dự án mô phỏng này, tôi chỉ sử dụng giá trị ADC thô hoặc chia theo mức để đánh giá chất lượng không khí là tốt, trung bình hay kém.

- Và cảm biến MQ-135 phản ứng với nhiều loại khí khác nhau, nên nó thường được coi là cảm biến tổng hợp không chuyên biệt cho một loại khí nào. Cho nên, nếu sử dụng MQ-135 để đo khí CO thì chỉ nên dùng để theo dõi xu hướng tăng giảm thay vì so sánh với tiêu chuẩn nồng độ CO cụ thể. Ngoài ra, cảm biến này cũng cần một thời gian làm nóng ban đầu để cho kết quả ổn định.

3.1.3. Nguyên lý hoạt động của cảm biến PMS5003

Hình 2: Cảm biến bụi mịn PMS5003 (hộp kim loại) kèm dây cáp kết nối UART.

- Cảm biến PMS5003 là một loại cảm biến chuyên dùng để đo nồng độ bụi mịn trong không khí. Đây là loại bụi có kích thước rất nhỏ, rất dễ xâm nhập vào đường hô hấp và gây ảnh hưởng đến sức khỏe, đặc biệt là bụi PM2.5. Cảm biến PMS5003 được sử dụng để đo chất lượng không khí nhờ khả năng đo chính xác và phản hồi nhanh.

- Nguyên lý hoạt động của PMS5003 dựa trên hiện tượng tán xạ ánh sáng bằng laser. Bên trong cảm biến có một diode laser phát ra tia sáng chiếu qua luồng không khí được hút vào nhờ quạt mini. Khi trong không khí có bụi, các hạt bụi sẽ làm tán xạ tia laser. Một cảm biến quang bên trong sẽ ghi lại mức tán xạ và từ đó vi xử lý bên trong cảm biến sẽ tính số lượng và kích thước của hạt bụi, sau đó chuyển đổi thành nồng độ bụi theo đơn vị $\mu g/m^3$.

- Cảm biến PMS5003 giao tiếp với vi điều khiển thông qua giao thức UART. Mỗi giây, nó sẽ gửi về một gói dữ liệu gồm các giá trị đo được, trong đó có thông tin về PM1.0, PM2.5 và PM10. Dữ liệu này ở dạng số (digital) nên vi điều khiển như ESP32 có thể xử lý trực tiếp mà không cần chuyển đổi tương tự như với cảm biến analog. Nhưng để đảm bảo kết quả đo chính xác, cảm biến này cũng cần thời gian làm nóng sau khi cấp nguồn để ổn định quạt và buồng đo.

- PMS5003 là một cảm biến dễ sử dụng, có độ chính xác cao, phù hợp để đo bụi mịn trong môi trường thực tế. Trong dự án này, vì không có điều kiện mô phỏng PMS5003 trực tiếp trên nền tảng Wokwi, nên tôi sử dụng cách sinh ngẫu nhiên các giá trị bụi PM2.5 để thay thế, nhưng vẫn đảm bảo vẫn kiểm tra được luồng dữ liệu và giao diện hiển thị.

3.1.4. Nguyên lý hoạt động của Blynk

- Blynk hoạt động theo mô hình client-server. Ở đây, ESP32 đóng vai trò gửi dữ liệu tới máy chủ đám mây Blynk, còn ứng dụng Blynk trên điện thoại cũng kết nối đến máy chủ đó để nhận dữ liệu và hiển thị cho người dùng. Nhờ có máy chủ trung gian này, thiết bị và điện thoại không cần kết nối trực tiếp với nhau mà vẫn giao tiếp dược với nhau trên đám mây.

- Quy trình hoạt động cơ bản của hệ thống Blynk như sau: Khi ESP32 kết nối WiFi thành công, nó sử dụng thư viện Blynk (với Auth Token được cấp riêng) để đăng ký một phiên làm việc với Blynk Cloud. Về phía người dùng, sau khi tạo một dự án trên ứng dụng Blynk và thêm các widget cần thiết (ví dụ đồng hồ Gauge hiển thị giá trị CO, biểu đồ cho PM2.5), ứng dụng điện thoại cũng mở kết nối tới Blynk Cloud và đăng ký nhận dữ liệu từ thiết bị (dựa trên Auth Token, Device ID, v.v.).

- Khi ESP32 gọi Blynk.virtualWrite(Vpin, value) để gửi dữ liệu, thư viện sẽ đóng gói cặp Virtual Pin & Value rồi gửi lên máy chủ Blynk. Blynk Server tiếp nhận, lưu giá trị vào cơ sở dữ liệu và đẩy ngay bản cập nhật tới tất cả ứng dụng đang mở dự án đó. Widget tương ứng trên app đã được gán với Virtual Pin đó sẽ nhận giá trị mới và cập nhật hiển thị tức thì.

- Ngược lại, nếu người dùng thao tác trên app, ứng dụng sẽ gửi lệnh lên máy chủ, sau đó máy chủ chuyển lệnh xuống ESP32 qua Virtual Pin hoặc chân điều khiển đã được thiết lập. Thư viện Blynk trên ESP32 sẽ nhận lệnh này và gọi hàm xử lý tương ứng do lập trình viên viết sẵn. Trong dự án giám sát không khí, chức năng gửi lệnh ngược có thể dùng để reset giá trị min/max đo được, hoặc điều khiển thiết bị phụ (ví dụ bật quạt thông gió khi PM2.5 quá cao).

- Blynk còn cung cấp giao diện rất trực quan để theo dõi dữ liệu lịch sử. Ví dụ: widget SuperChart có thể vẽ biểu đồ thời gian thực và lưu trữ dữ liệu đo. Ở phiên bản Blynk 2.0 mới, người dùng còn có trang web dashboard và các tính năng nâng cao như tự động hóa, thông báo sự kiện, quản lý thiết bị... Tuy nhiên, cốt lõi nguyên lý Blynk vẫn là: thiết bị kết nối internet → gửi dữ liệu → máy chủ → ứng dụng hiển thị. Nhờ đó, người dùng không cần hiểu sâu về mạng hay lập trình giao diện vẫn có thể làm chủ dự án IoT của mình một cách dễ dàng.

3.1.5. Nguyên lý hoạt động của các thành phần còn lại (điện trở, LED, nguồn…)

- Điện trở (Resistor): Điện trở hoạt động theo định luật Ohm, có tác dụng cản trở dòng điện và tạo ra sụt áp trên nó. Trong mạch, điện trở thường được dùng để hạn dòng cho LED tránh LED bị quá dòng dẫn đến hỏng và làm điện trở kéo (pull-up/pull-down) hoặc cầu chia áp cho tín hiệu. Nguyên tắc chung: điện trở càng lớn thì dòng qua mạch càng nhỏ, điện trở càng nhỏ thì dòng càng lớn. Chọn đúng giá trị điện trở sẽ giúp các linh kiện khác hoạt động trong giới hạn an toàn.

- Đèn LED: LED là một loại diode bán dẫn phát sáng khi có dòng điện chạy qua theo chiều thuận từ cực dương sang cực âm. LED chỉ dẫn điện một chiều và có một điện áp ngưỡng thông thường khoảng 1.8V–3.3V tùy màu sắc. Trong mạch, LED thường được

dùng làm đèn báo: ví dụ LED nguồn sáng khi mạch được cấp điện, hoặc LED cảnh báo sáng khi chất lượng không khí kém. Trên board ESP32 DevKit, có sẵn một đèn LED màu xanh dương nối với chân GPIO2, có thể lập trình nhấp nháy để báo hiệu WiFi đã kết nối thành công. Độ sáng LED tăng khi dòng qua nó tăng, nhưng nếu vượt quá giới hạn thì LED sẽ bị cháy. Vì vậy, luôn phải có điện trở nối tiếp để giới hạn dòng cho LED.

- Nguồn điện: Nguồn điện cung cấp năng lượng cho toàn hệ thống. Mô hình này sử dụng nguồn một chiều +5V. Nguồn có thể lấy từ cổng USB máy tính, adapter điện 5V, hoặc pin. Trên board ESP32 DevKit có một bộ ổn áp tuyến tính để hạ 5V xuống 3.3V nuôi vi điều khiển. Các cảm biến MQ-135 và PMS5003 đều dùng điện áp 5V. Nếu dùng pin, cần có mạch sạc và mạch boost tăng áp nếu cần để duy trì 5V khi hoạt động. Trong mô phỏng Wokwi, nguồn 5V được tự động cung cấp khi nối linh kiện vào chân 5V của ESP32, và Wokwi cũng mô phỏng sẵn bộ ổn áp 3.3V trên board, do đó chỉ cần cấp 5V đầu vào là đủ.

3.2 Cấu trúc hệ thống giám sát

- Hệ thống giám sát chất lượng không khí trong đề tài bao gồm các thành phần chính:

1. Cảm biến MQ-135 (dùng để đo khí CO)
2. Cảm biến PMS5003 (đo nồng độ bụi PM2.5)
3. Vi điều khiển ESP32 (đóng vai trò xử lý trung tâm và kết nối WiFi), mạng WiFi (giúp ESP32 truyền dữ liệu lên Internet)
4. Nền tảng Blynk (để giám sát từ xa qua ứng dụng điện thoại).

Quá trình hoạt động của hệ thống diễn ra theo các bước như sau:

3.2.1. Thu thập dữ liệu từ cảm biến:

- Cảm biến MQ-135 xuất tín hiệu analog được kết nối vào chân GPIO34 của ESP32 để đọc ADC, vi điều khiển có thể đọc giá trị điện áp analog tương ứng với nồng độ khí CO trong không khí.

Trong mô hình mô phỏng, vì Wokwi chưa hỗ trợ cảm biến MQ-135, nên tôi sử dụng một biến trở (Potentiometer) làm đầu vào giả lập cho tín hiệu cảm biến với:

1. Chân tín hiệu (SIG) của biến trở → GPIO34
2. Hai chân còn lại là GND và VCC nối lần lượt vào 3.3V và GND của ESP32

- Cảm biến PMS5003 kết nối với ESP32 thông qua giao tiếp UART kết nối với chân GPIO16 và GPIO17 cho TX/RX, giúp truyền dữ liệu số về nồng độ bụi PM2.5. ESP32 sẽ đọc các cảm biến này theo chu kỳ, ví dụ mỗi 2 giây một lần.

- Trong mô hình mô phỏng, vì Wokwi chưa hỗ trợ cảm biến PMS5003, nên tôi sử dụng một biến trở (Potentiometer) làm đầu vào giả lập cho tín hiệu cảm biến với:

1. Chân tín hiệu (SIG) của biến trở → GPIO32
2. Hai chân còn lại là GND và VCC nối lần lượt vào 3.3V và GND của ESP32

3.2.2. Xử lý và hiệu chỉnh dữ liệu:

- Sau khi nhận được dữ liệu, ESP32 sẽ xử lý các giá trị đầu vào. Đối với MQ-135, vi điều khiển sẽ chuyển đổi giá trị điện áp analog thành một mức ước lượng của khí CO (tính theo ppm nếu có hiệu chỉnh, hoặc đánh giá tương đối). Đối với PMS5003, ESP32 sẽ giải mã gói dữ liệu nhận được để lấy ra giá trị nồng độ bụi PM2.5 (đơn vị $\mu g/m^3$).

3.2.3. Hiển thị cục bộ (OLED):

- Hệ thống cho phép người dùng quan sát trực tiếp các giá trị đo được thông qua màn hình OLED 128x64, kết nối với ESP32 qua giao tiếp I2C (GPIO21 là SDA, GPIO22 là SCL).

3.2.4 Gửi dữ liệu lên nền tảng Blynk:

- Song song với việc hiển thị cục bộ, ESP32 sẽ truyền dữ liệu lên nền tảng Blynk thông qua Internet. Blynk là một ứng dụng IoT phổ biến, cho phép gửi và nhận dữ liệu giữa phần cứng (ESP32) và ứng dụng di động. Khi ESP32 kết nối WiFi thành công, nó sẽ khởi tạo Blynk bằng Auth Token. Mỗi khi có dữ liệu mới, ESP32 sẽ gửi các giá trị lên Virtual Pin tương ứng trên Blynk.

3.2.5. Giám sát từ xa bằng ứng dụng Blynk:

- Người dùng có thể tải và cài đặt ứng dụng Blynk trên điện thoại Android hoặc iOS, sau đó đăng nhập và thêm thiết bị ESP32 vào ứng dụng. Tại đây, người dùng có thể tạo giao diện riêng bằng cách thêm các widget như hiển thị số, biểu đồ hoặc đồng hồ đo. Các widget sẽ hiển thị giá trị được gửi từ ESP32 theo thời gian thực, cho phép người dùng dễ dàng theo dõi chất lượng không khí dù ở bất kỳ đâu miễn có Internet. Ngoài ra, Blynk cũng hỗ trợ Web Dashboard giúp người dùng truy cập và giám sát qua trình duyệt trên máy tính.

3.3 Sơ đồ mô phỏng trên Wokwi

- Để phát triển và kiểm thử hệ thống mà không cần sử dụng phần cứng thực tế, tôi đã sử dụng nền tảng Wokwi, một công cụ mô phỏng phần cứng trực tuyến cho vi điều khiển như Arduino và ESP32. Đây là một ứng dụng rất tiện lợi, cho phép lập trình và chạy thử trực tiếp trên trình duyệt với nhiều loại linh kiện phổ biến như cảm biến, màn hình, động cơ, và thậm chí có thể mô phỏng kết nối WiFi.

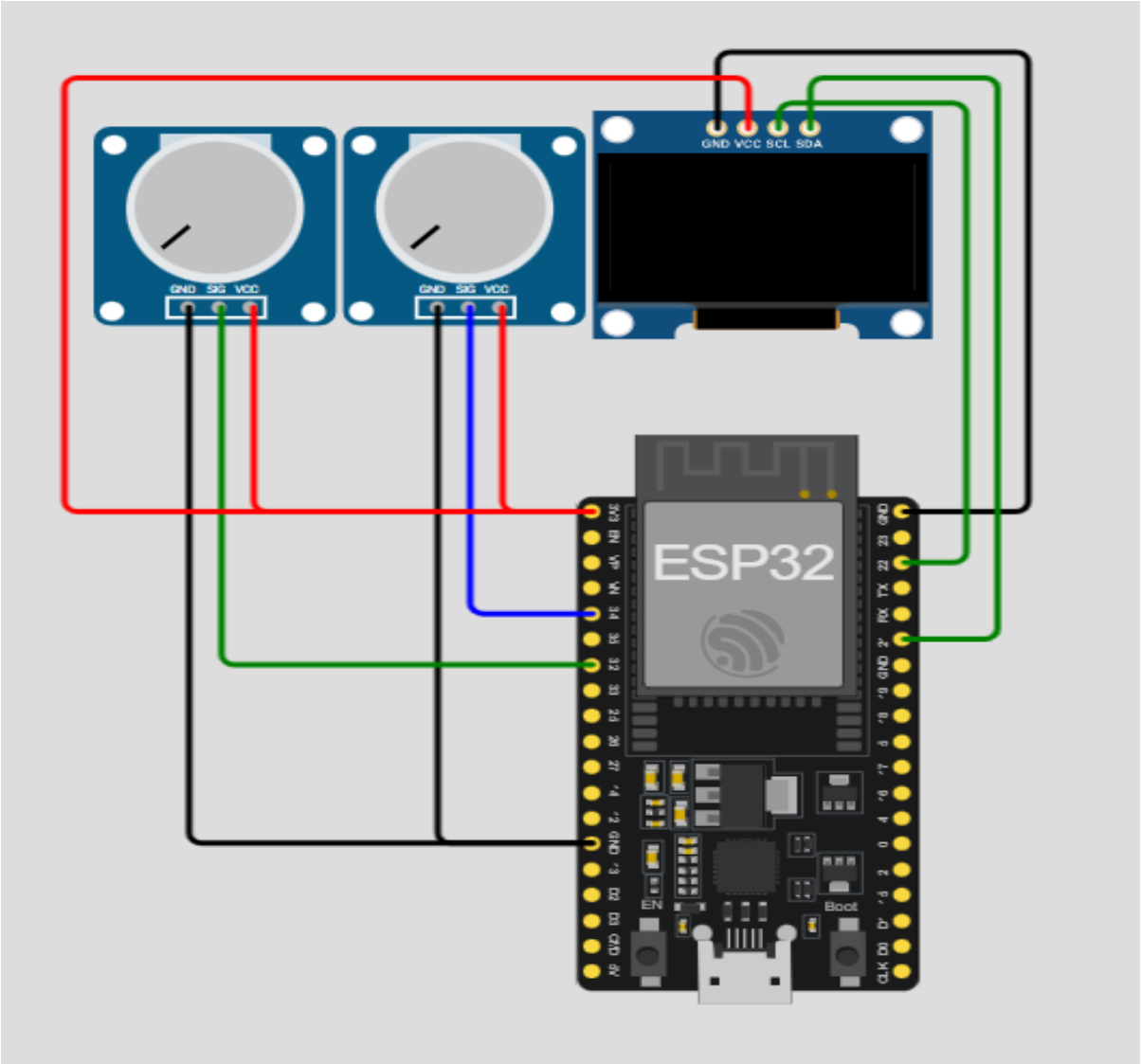
- Trong dự án này, tôi sử dụng board ESP32 làm trung tâm xử lý và kết nối các thiết bị ngoại vi. Do Wokwi hiện chưa hỗ trợ mô phỏng cảm biến MQ-135 nên tôi thay thế bằng một potentiometer (biến trở) để giả lập tín hiệu đầu ra dạng analog. Cụ thể, biến trở được kết nối với chân GPIO34 (chân ADC) của ESP32, còn hai chân còn lại nối với nguồn 3.3V và GND. Khi xoay biến trở, điện áp đầu ra thay đổi từ 0 đến 3.3V – tương đương với các mức nồng độ khí CO mà MQ-135 có thể đo trong thực tế.

Tương tự, cảm biến PMS5003 cũng chưa được hỗ trợ mô phỏng trên Wokwi do sử dụng giao tiếp UART khá phức tạp. Cho nên, tôi tạo giá trị PM2.5 bằng một potentiometer (biến trở) để giả lập tín hiệu đầu ra dạng analog. Cụ thể, biến trở được kết nối với chân GPIO32 (chân ADC) của ESP32, còn hai chân còn lại nối với nguồn 3.3V

và GND. Khi xoay biến trở, điện áp đầu ra thay đổi từ 0 đến 3.3V – tương đương với các mức nồng độ khí PM2.5 mà PMS5003 có thể đo trong thực tế. Cách làm này giúp quá trình mô phỏng vẫn đi đúng luồng xử lý dữ liệu mà không cần cảm biến thực.

- Để hiển thị dữ liệu đo được, tôi sử dụng một màn hình OLED 128x64 sử dụng giao tiếp I2C, cũng được mô phỏng trực tiếp trên Wokwi. Màn hình này được nối với ESP32 thông qua hai chân: SDA (GPIO21) và SCL (GPIO22). Nguồn cấp và GND của màn hình OLED nối với 3.3V và GND. Mỗi 2 giây, dữ liệu CO và PM2.5 từ biến trở sẽ được cập nhật lên màn hình OLED để người dùng dễ quan sát.

- Ngoài ra, ESP32 còn kết nối với Serial Monitor ảo của Wokwi, cho phép in log (tốc độ baud 115200). Nhờ đó, tôi có thể quan sát rõ ràng các giá trị đang được xử lý qua từng vòng lặp của chương trình.



Hình 3. sơ đồ mô phỏng trên Wokwi

```
{
  "version": 1,
  "author": "Capi.Robust",
  "editor": "wokwi",
  "parts": [
  { "type": "board-esp32-devkit-c-v4", "id": "esp", "top": 0, "left": -4.76, "attrs": {} },
  {
    "type": "wokwi-potentiometer",
    "id": "pot1",
    "top": -116.5,
    "left": -77,
    "attrs": { "value": "0.5" }
  },
  {
    "type": "wokwi-potentiometer",
    "id": "pot2",
    "top": -116.5,
    "left": -153.8,
    "attrs": { "value": "0.5" }
  },
  {
    "type": "board-ssd1306",
    "id": "oled1",
    "top": -121.66,
    "left": 0.23,
    "attrs": { "i2cAddress": "0x3c" }
  }
 ],
 "connections": [
  [ "esp:TX", "$serialMonitor:RX", "", [] ],
  [ "esp:RX", "$serialMonitor:TX", "", [] ],
  [ "pot1:GND", "esp:GND.1", "black", [ "v0" ] ],
  [ "pot1:VCC", "esp:3V3", "red", [ "v0" ] ],
  [ "pot1:SIG", "esp:34", "blue", [ "v0" ] ],
  [ "pot2:GND", "esp:GND.1", "black", [ "v0" ] ],
  [ "pot2:VCC", "esp:3V3", "red", [ "v0" ] ],
  [ "pot2:SIG", "esp:32", "green", [ "v0" ] ],
  [ "oled1:GND", "esp:GND.2", "black", [ "v-28.8", "h96", "v172.8" ] ],
  [ "oled1:VCC", "esp:3V3", "red", [ "v-19.2", "h-211.05", "v163.2" ] ],
  [ "oled1:SCL", "esp:22", "green", [ "v-9.6", "h57.9", "v172.8" ] ],
  [ "oled1:SDA", "esp:21", "green", [ "v-19.2", "h57.67", "v211.2" ] ]
 ],
 "dependencies": {}
}
```

Hình 4. Sơ đồ mô phỏng trên Wokwi

Potentiometer (giả lập MQ-135) → chân tín hiệu nối vào GPIO34

Potentiometer (giả lập PMS5003) → chân tín hiệu nối vào GPIO32

Màn hình OLED SSD1306 → kết nối qua SDA (GPIO21) và SCL (GPIO22)

Nguồn và GND của tất cả các linh kiện đều nối với 3.3V và GND của ESP32

ESP32 được cấp nguồn tự động trong môi trường mô phỏng

- Chương trình mô phỏng được viết bằng ngôn ngữ Arduino C++. Trong phần setup(), tôi khởi tạo Serial, OLED và khai báo chân ADC. Trong loop(), ESP32 đọc giá trị từ potentiometer, sau đó hiển thị dữ liệu lên OLED và in ra Serial Monitor. Chu kỳ cập nhật lặp lại sau mỗi 2 giây.

- Khi chạy mô phỏng, người dùng có thể xoay núm biến trở để thay đổi giá trị analog giả lập đầu ra của MQ-135 và PMS5003 thì giá trị CO và PM2.5 trên màn hình OLED sẽ thay đổi theo.

3.4 Giao tiếp giữa ESP32 và nền tảng Blynk

- Trong dự án này, tôi sử dụng nền tảng Blynk để truyền dữ liệu từ thiết bị ESP32 lên Internet, giúp người dùng có thể theo dõi chất lượng không khí từ xa thông qua điện thoại hoặc máy tính. Blynk cung cấp một hệ sinh thái gồm ba phần chính:

1. Blynk Cloud (máy chủ lưu trữ và trao đổi dữ liệu)
2. Ứng dụng Blynk trên điện thoại (giao diện người dùng)
3. Thư viện Blynk để lập trình tích hợp trên các vi điều khiển như ESP32

- Ưu điểm của Blynk là người dùng không cần tự triển khai máy chủ hay viết ứng dụng riêng, mà chỉ cần sử dụng các Virtual Pin để gửi và nhận dữ liệu giữa thiết bị và nền tảng đám mây. Giao diện thân thiện, cho phép kéo thả widget, đặt tên, chọn đơn vị hiển thị,… phù hợp với người học hoặc các dự án nhỏ.

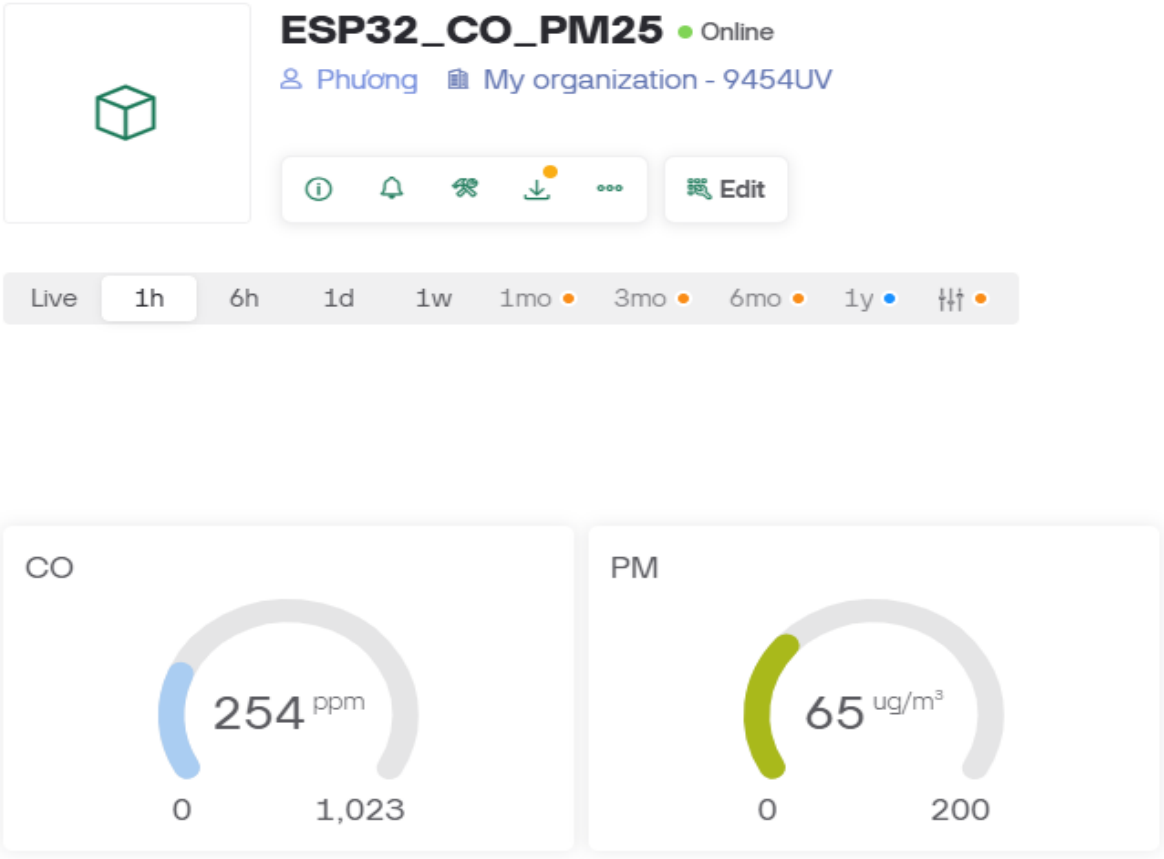
- Để ESP32 có thể giao tiếp với Blynk, trước hết tôi cần thêm thư viện Blynk vào chương trình. Khi tạo dự án trên ứng dụng Blynk, hệ thống sẽ cung cấp một chuỗi mã xác thực riêng. Tôi khai báo mã này cùng với tên mạng WiFi và mật khẩu trong chương trình. Khi thiết bị khởi động, nó sẽ sử dụng lệnh Blynk.begin(auth, ssid, pass) để kết nối tới Blynk Cloud. Nếu kết nối thành công, TCP sẽ liên tục trao đổi dữ liệu giữa ESP32 và máy chủ Blynk.

- Sau mỗi chu kỳ đo, ESP32 sẽ gửi dữ liệu cảm biến lên Blynk bằng hàm Blynk.virtualWrite(pin, value). Tôi gán chân V1 cho giá trị CO và V2 cho giá trị bụi PM2.5. Cụ thể, ESP32 sẽ thực hiện lệnh Blynk.virtualWrite(V1, co_value) và Blynk.virtualWrite(V2, pm25_value) để cập nhật dữ liệu cho người dùng.

- Phía người dùng, khi thiết kế giao diện trên app Blynk, tôi có thể thêm các widget như Value Display, Gauge, Chart,… và gán chúng với các Virtual Pin tương ứng. Ví dụ, widget Gauge hiển thị giá trị CO từ V1 và hiển thị giá trị PM2.5 từ V2.

- Việc sử dụng Blynk trong dự án giúp cho hệ thống ESP32 dễ dàng kết nối Internet và tương tác với người dùng. Dữ liệu từ cảm biến được cập nhật liên tục lên nền tảng đám mây, giúp người dùng theo dõi chất lượng không khí mọi lúc mọi nơi.

Hình 5. Hiển thị thông tin nhận được lên OLED



Hình 6. Hiển thị thông tin nhận được lên Blynk

4. Kết luận và nhận xét

4.1. Kết luận

Đề tài "Giám sát chất lượng không khí với ESP32 và cảm biến MQ" đã xây dựng được một hệ thống IoT cơ bản có khả năng theo dõi hai thông số quan trọng của không khí là nồng độ khí CO và bụi mịn PM2.5.

Thông qua việc tích hợp các cảm biến MQ-135 và PMS5003 với vi điều khiển ESP32, Hệ thống có khả năng đo lường tương đối nồng độ khí CO và bụi mịn PM2.5, hiển thị thông tin một cách trực quan trên màn hình OLED, đồng thời đồng bộ dữ liệu lên nền tảng Blynk để người dùng theo dõi từ xa qua ứng dụng di động. Kết quả mô phỏng trên Wokwi cho thấy hệ thống vận hành ổn định: các giá trị cảm biến được đọc và cập nhật định kỳ, giao tiếp WiFi (trong code) và gửi dữ liệu lên Blynk thực hiện đúng logic thiết kế. Đề tài đã đạt được mục tiêu đề ra về mặt chức năng, và đồng thời giúp tôi hiểu sâu hơn về quy trình xây dựng một ứng dụng IoT hoàn chỉnh từ phần cứng cảm biến, lập trình firmware, đến khai thác dịch vụ IoT như Blynk.

4.2. Nhận xét

Hệ thống giám sát chất lượng không khí với ESP32, MQ-135 và PMS5003 đã đáp ứng được các yêu cầu cơ bản về chức năng. Đề tài tận dụng hiệu quả các kiến thức liên ngành: từ cảm biến phần cứng, lập trình nhúng trên vi điều khiển, đến mạng WiFi và dịch vụ IoT đám mây. Đây là minh chứng cho thấy khoa Công nghệ thông tin của trường Đại học khoa học, Đại học Huế giúp cho sinh viên có thể ứng dụng kiến thức lập trình của mình trong lĩnh vực IoT và khoa học dữ liệu môi trường.

5. Tài liệu tham khảo

[1]: Tài liệu tham khảo về IoT: https://aws.amazon.com/vi/what-is/iot/

[2]: ESP32 Programming for Beginners: https://www.instructables.com/Make-Your-Own-ESP32-Simple/

[3]: Tài liệu hướng dẫn sử dụng cảm biến MQ-135: https://www.codrey.com/electronic-circuits/how-to-use-mq-135-gas-sensor/

[4]: Tài liệu hướng dẫn sử dụng cảm biến PMS5003: https://soldered.com/learn/hum-plantower-pms5003/?srsltid=AfmBOopcCqiFWohxUeBq9G6o_b4eUPDGnADZrubTl08WjyDn1wvJ7ZC5

[5]: Tài liệu Blynk: https://docs.blynk.io/

- Tài liệu học tập, source code tham khảo của thầy Võ Việt Dũng, https://github.com/vvdung-husc/2024-2025.2.TIN4024.004