|  |
| --- |
| Ảnh có chứa quảng trường  Mô tả được tạo tự độngBAN CƠ YẾU CHÍNH PHỦ  **HỌC VIỆN KỸ THUẬT MẬT MÃ**  ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯  Logo HvKTMM |
| **BÁO CÁO KẾT THÚC HỌC ĐỒ ÁN 3**  **NGHIÊN CỨU, THIẾT KẾ TRẠM KHÍ TƯỢNG GIÁM SÁT VÀ CẢNH BÁO TỪ XA** |
| *Sinh viên thực hiện*:  Nguyễn Tiến Dũng DT030107  Ngô Đức Vinh DT030150 |
| **Hà Nội, 2023** |

LỜI CẢM ƠN

Chúng em xin bày tỏ lòng biết ơn sâu sắc nhất tới thầy Đặng Văn Hải, thầy đã tận tình hướng dẫn và giúp đỡ chúng em trong suốt quá trình làm đề tài này. Với sự chỉ bảo của thầy, chúng em đã có những định hướng tốt trong việc triển khai và thực hiện các yêu cầu trong quá trình làm đề tài.

Chúng em xin chân thành cảm ơn sự dạy bảo và giúp đỡ của các thầy giáo, cô giáo Học viện Kỹ thuật Mật mã đã trang bị những kiến thức cơ bản nhất để chúng em có thể hoàn thành tốt báo cáo này.

Chúng em xin chân thành cảm ơn các thầy giáo, cô giáo trong khoa Điện tử viễn thông của Học viện đã góp ý và tạo điều kiện cho chúng em trong việc tìm hiểu và thực hiện đề tài này.

Xin cảm ơn tới những người thân trong gia đình đã quan tâm trong suốt quá trình học tập và làm đề tài.

Xin gửi lời cảm ơn tất cả bạn bè, đặc biệt là các bạn trong lớp DT3A đã giúp đỡ và đóng góp ý kiến để chúng mình hoàn thành đề tài.

MỤC LỤC

[LỜI CẢM ƠN 2](#_Toc131874067)

[MỤC LỤC 3](#_Toc131874068)

[DANH MỤC HÌNH ẢNH 5](#_Toc131874069)

[DANH MỤC BẢNG BIỂU 6](#_Toc131874070)

[CHƯƠNG I: TỔNG QUAN VỀ ĐỀ TÀI 7](#_Toc131874071)

[1.1. Tổng quan và đưa ra bài toán 7](#_Toc131874072)

[1.2. Giải pháp thiết kế 8](#_Toc131874073)

[CHƯƠNG II: CƠ SỞ LÝ THUYẾT ĐỂ THIẾT KẾ ĐỀ TÀI 11](#_Toc131874074)

[2.1. Công nghệ phần mềm 11](#_Toc131874075)

[2.1.1. Giới thiệu về Visual Studio 2019 11](#_Toc131874076)

[2.1.2. Phần mềm Arduino IDE 11](#_Toc131874077)

[2.1.3. Phần mềm Sql 2019 12](#_Toc131874078)

[2.1.4. Sơ lược về MQTT 13](#_Toc131874079)

[2.1.5. Phần mềm MQTTX 14](#_Toc131874080)

[2.2. Công nghệ phần cứng 15](#_Toc131874081)

[2.2.1. Kit ESP32 15](#_Toc131874082)

[2.2.2. Module cảm biến độ ẩm, nhiệt độ DHT22 17](#_Toc131874083)

[2.2.3. Module cảm biến chất lượng không khí MQ-135 19](#_Toc131874084)

[2.2.4. Module cảm biến mưa 20](#_Toc131874085)

[CHƯƠNG III: XÂY DỰNG VÀ THẾT KẾ SẢN PHẨM 22](#_Toc131874086)

[3.1. Thiết kế phần cứng 22](#_Toc131874087)

[3.1.1. Lựa chọn linh kiện 22](#_Toc131874088)

[3.1.2. Sơ đồ nguyên lý 22](#_Toc131874089)

[3.2. Thiết kế phần mềm 23](#_Toc131874090)

[3.3. Thực nghiệm 23](#_Toc131874091)

[CHƯƠNG IV: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 27](#_Toc131874092)

[4.1. Kết luận 27](#_Toc131874093)

[4.2. Hướng phát triển 27](#_Toc131874094)

[TÀI LIỆU THAM KHẢO 28](#_Toc131874095)

[PHỤ LỤC 29](#_Toc131874096)

DANH MỤC HÌNH ẢNH

[Hình 1.1.1 Sơ đồ khối giải pháp 9](#_Toc131874097)

[Hình 2.1.1.1 Hình ảnh về phần mềm Visual Studio 2019 11](#_Toc131874098)

[Hình 2.1.2.1 biểu tượng của Arduino IDE 12](#_Toc131874099)

[Hình 2.1.3.1 Giao diện của SQL Server 2019 13](#_Toc131874100)

[Hình 2.1.4.1 Giao thức MQTT 13](#_Toc131874101)

[Hình 2.1.5.2 Hình ảnh về MQTTx 15](#_Toc131874102)

[Hình 2.2.1.1 Hình ảnh về ESP32 16](#_Toc131874103)

[Hình 2.2.2.1 Hình ảnh về module cảm biến DHT22 18](#_Toc131874104)

[Hình 2.2.2.2 Giản đồ thời gian của module DHT22 19](#_Toc131874105)

[Hình 2.2.3.1 Hình ảnh về module cảm biến chất lượng không khí MQ-135 19](#_Toc131874106)

[Hình 2.2.4.1 Hình ảnh về cảm biến mưa 21](#_Toc131874107)

[Hình 3.1.2.1 Sơ đồ nguyên lý toàn hệ thống 22](#_Toc131874108)

[Hình 3.2.1 Lưu đồ thuật toán toàn hệ thống 23](#_Toc131874109)

[Hình 3.3.1 Sản phẩm hoàn thiện sau khi lắp ráp 24](#_Toc131874110)

[Hình 3.3.2 Dữ liệu được gửi đến mqtt borker 25](#_Toc131874111)

[Hình 3.3.3 Dữ liệu được gửi đến phần mềm SQL 25](#_Toc131874112)

[Hình 3.3.4 Các thông số thời tiết hiển thị trên web server 26](#_Toc131874113)

DANH MỤC BẢNG BIỂU

[Hình 2.2.1.2 Sơ đồ chân ESP32 14](#_Toc131874114)

[Bảng 2.2.1.1 Thông số kĩ thuật của ESP32 15](#_Toc131874115)

[Bảng 2.2.2.1 Sơ đồ chân DHT22 16](#_Toc131874116)

[Bảng 2.2.2.2 Thông số kĩ thuật của module cảm biến DHT22 16](#_Toc131874117)

[Bảng 2.2.3.1 Sơ đồ chân của module MQ-135 18](#_Toc131874118)

[Bảng 2.2.3.2 Thông số kỹ thuật module cảm biến MQ135 18](#_Toc131874119)

[Bảng 2.2.4.2 Bảng thông số kĩ thuật của module cảm biến mưa 19](#_Toc131874120)

[Bảng 3.1.1.1 Các linh kiện cần chuẩn bị 20](#_Toc131874121)

CHƯƠNG I: TỔNG QUAN ĐỒ ÁN

1.1. Mục tiêu đồ án

Mục tiêu của đề tài "Nghiên cứu, thiết kế trạm khí tượng giám sát và cảnh báo từ xa" là tạo ra một hệ thống hiện đại và đáng tin cậy để thu thập dữ liệu khí tượng và cung cấp cảnh báo từ xa cho mục đích giám sát và dự báo thời tiết. Đồ án này nhằm đáp ứng nhu cầu ngày càng tăng của các tổ chức khí tượng và các đơn vị quan trắc khí tượng về việc thu thập và phân tích dữ liệu thời tiết một cách hiệu quả.

Mục tiêu cụ thể của dự án là thiết kế và phát triển một trạm khí tượng giám sát tự động, được trang bị các cảm biến đa chức năng nhằm đo đạc và ghi nhận các yếu tố khí tượng quan trọng như nhiệt độ, độ ẩm, lượng mưa và chất lượng không khí. Trạm khí tượng sẽ được lắp đặt và hoạt động một cách tự động, đảm bảo tính liên tục và chính xác của dữ liệu thu thập được. Trong khi đó, sử dụng ESP32 - một vi điều khiển nhỏ gọn có khả năng kết nối Wi-Fi và Bluetooth, với tính năng xử lý tín hiệu cao và chi phí thấp, là giải pháp tối ưu cho việc thiết kế một trạm khí tượng thông minh.

Hơn nữa, hệ thống sẽ được kết nối với mạng không dây hoặc mạng di động, cho phép truyền dữ liệu từ trạm khí tượng về một trung tâm giám sát từ xa. Trung tâm này sẽ có khả năng xử lý và phân tích dữ liệu một cách tự động, tạo ra thông báo cảnh báo dựa trên các thông số khí tượng và cung cấp thông tin thời tiết chính xác và kịp thời cho người dùng.

Mục tiêu cuối cùng của đồ án là cung cấp cho các tổ chức khí tượng, người nghiên cứu và công chúng một công cụ mạnh mẽ để theo dõi và dự báo thời tiết. Sự cung cấp thông tin thời tiết chính xác và đáng tin cậy từ trạm khí tượng giám sát và cảnh báo từ xa sẽ hỗ trợ quyết định trong các lĩnh vực như nông nghiệp, hàng hải, hàng không, quản lý môi trường và địa chính.

Tóm lại, mục tiêu đồ án là xây dựng một trạm khí tượng giám sát tự động và hệ thống cảnh báo từ xa chất lượng cao, đảm bảo tính liên tục và chính xác của dữ liệu thời tiết, nhằm cung cấp thông tin thời tiết đáng tin cậy và hỗ trợ quyết định cho các tổ chức và cá nhân quan tâm đến lĩnh vực khí tượng và thời tiết

1.2. Phạm vi nghiên cứu đồ án

Phạm vi nghiên cứu của đồ án bao gồm sự tập trung vào việc áp dụng và phát triển các công nghệ hiện đại và tiên tiến trong lĩnh vực khí tượng và truyền thông từ xa. Các công nghệ được sử dụng trong đồ án của bạn bao gồm MQTT (Message Queuing Telemetry Transport) và ASP.NET.

MQTT là một giao thức truyền thông đáng tin cậy và nhẹ nhàng được thiết kế để truyền dữ liệu giữa các thiết bị trong mạng IoT (Internet of Things). Trong đồ án, bạn sử dụng công nghệ MQTT để truyền dữ liệu giữa ESP32 (một thiết bị nền tảng IoT) và máy chủ web (webserver). Giao thức MQTT đảm bảo khả năng truyền thông hiệu quả và đáng tin cậy giữa hai bên, cho phép việc thu thập và chuyển giao dữ liệu thời tiết một cách hiệu quả và không gây lãng phí tài nguyên.

ASP.NET là một nền tảng lập trình phát triển ứng dụng web mạnh mẽ của Microsoft. Bằng cách sử dụng ASP.NET, bạn có khả năng lập trình và phát triển máy chủ web để xử lý và hiển thị dữ liệu thời tiết từ trạm khí tượng và cung cấp giao diện người dùng tương tác cho việc theo dõi và cảnh báo từ xa. ASP.NET cung cấp các công cụ và thư viện phong phú để xây dựng ứng dụng web linh hoạt, bảo mật và có hiệu suất cao.

Ngoài ra, nghiên cứu của bạn có thể tìm hiểu và tham khảo các công nghệ và giải pháp khác đang được sử dụng trong lĩnh vực khí tượng và truyền thông từ xa. Điều này có thể bao gồm việc nghiên cứu các giao thức truyền thông khác như HTTP, WebSockets, hay các công nghệ khác như data analytics, và machine learning để phân tích và dự báo thời tiết. Việc tìm hiểu và áp dụng các công nghệ tiên tiến này sẽ nâng cao tính hiệu quả và chất lượng của trạm khí tượng giám sát và hệ thống cảnh báo từ xa của bạn.

1.2.1 Giới thiệu về MQTT

1. Giới thiệu về MQTT (Message Queuing Telemetry Transport)

MQTT (Message Queuing Telemetry Transport) là một giao thức truyền thông nhẹ được phát triển để đạt hiệu suất cao, tiết kiệm băng thông và có khả năng hoạt động ổn định trong mạng không đáng tin cậy. Nó được thiết kế cho các ứng dụng M2M (Machine-to-Machine) và IoT (Internet of Things) để truyền tải dữ liệu từ các thiết bị nhúng và cung cấp thông tin theo thời gian thực.

A picture containing text, diagram, screenshot, line

Description automatically generated

Hình 1.2.1 Mô hình hoạt động của MQTT

1. Ưu điểm của MQTT

* Độ nhẹ và và tiết kiệm băng thông: MQTT sử dụng định dạng tin nhắn nhỏ gọn và header tối thiểu, giúp giảm lượng dữ liệu truyền tải. Điều này rất hữu ích trong các mạng có băng thông hạn chế và giúp tiết kiệm năng lượng trong các thiết bị IoT.
* Giao tiếp đa nền tảng: MQTT hỗ trợ nhiều nền tảng và ngôn ngữ lập trình, giúp dễ dàng tích hợp vào các hệ thống sẵn có và cho phép giao tiếp giữa các thiết bị chạy trên các nền tảng khác nhau.
* Mô hình publish-subscribe: MQTT sử dụng mô hình publish-subscribe, trong đó các thiết bị gửi (publish) tin nhắn đến các kênh (topic), và các thiết bị khác đăng ký (subscribe) để nhận các tin nhắn từ các kênh tương ứng. Điều này tạo ra một mô hình linh hoạt và phân tán cho việc truyền tải dữ liệu trong hệ thống.
* Khả năng hoạt động ổn định: MQTT được thiết kế để hoạt động trong mạng không đáng tin cậy và có khả năng tự động kết nối lại khi mất kết nối. Nó hỗ trợ cơ chế lưu trữ tin nhắn (message persistence) để đảm bảo rằng các tin nhắn không bị mất đi trong quá trình truyền tải.

1. Nhược điểm của MQTT

* Không đảm bảo giao hàng chính xác: MQTT không đảm bảo rằng tất cả các tin nhắn sẽ được gửi và nhận chính xác. Nó chỉ cam kết gửi tin nhắn đi và đảm bảo rằng các thiết bị đã đăng ký sẽ nhận được tin nhắn đó, nhưng không đảm bảo rằng tất cả các tin nhắn đều đến đích.
* Bảo mật yếu: MQTT có các cơ chế bảo mật cơ bản nhưng không mạnh mẽ như một số giao thức truyền thông khác. Việc truyền tải dữ liệu qua MQTT cần phải được bảo vệ bằng các phương pháp mã hóa và xác thực bổ sung để đảm bảo tính riêng tư và an toàn.

Tóm lại, MQTT là một giao thức truyền thông nhẹ, linh hoạt và tiết kiệm băng thông được sử dụng rộng rãi trong các ứng dụng M2M và IoT. Nó cho phép truyền tải dữ liệu trong mạng không đáng tin cậy và hỗ trợ tính năng publish-subscribe giữa các thiết bị khác nhau. Tuy nhiên, MQTT cần được kết hợp với các biện pháp bảo mật bổ sung để đảm bảo tính an toàn và riêng tư của dữ liệu.

1.2.2 Giới thiệu về ASP.NET

a) Giới thiệu về ASP.NET

ASP.NET là một nền tảng phát triển ứng dụng web phía máy chủ (server-side) được phát triển bởi Microsoft. Nó cung cấp một mô hình lập trình linh hoạt và mạnh mẽ để xây dựng các ứng dụng web đa dạng từ các trang web tĩnh đơn giản đến các ứng dụng doanh nghiệp phức tạp.

b) Ưu điểm của ASP.NET

* Hiệu suất cao: ASP.NET sử dụng mô hình lập trình mã nguồn mở Just-In-Time Compilation (JIT) để biên dịch trực tiếp mã nguồn thành mã máy khi ứng dụng được thực thi. Điều này giúp tăng hiệu suất và tăng tốc độ phản hồi của ứng dụng.
* Bảo mật mạnh mẽ: ASP.NET cung cấp các tính năng bảo mật tích hợp như xác thực người dùng, quản lý phiên, mã hóa dữ liệu và kiểm soát truy cập. Nó hỗ trợ cả bảo mật dựa trên vai trò (role-based security) và bảo mật dựa trên chứng chỉ (certificate-based security).
* Tích hợp tốt với công cụ phát triển Visual Studio: ASP.NET được tích hợp chặt chẽ với Visual Studio, môi trường phát triển tích hợp (IDE) của Microsoft. Điều này cung cấp cho lập trình viên các công cụ mạnh mẽ để phát triển, gỡ lỗi và triển khai ứng dụng web dễ dàng hơn.
* Tích hợp dễ dàng với công nghệ Microsoft: ASP.NET được tích hợp mạnh mẽ với các công nghệ của Microsoft như SQL Server, Azure, Active Directory và nhiều hệ thống Microsoft khác. Điều này giúp phát triển và triển khai ứng dụng web trên nền tảng Microsoft dễ dàng hơn.

c) Nhược điểm của ASP.NET

* Hạn chế đối với các hệ thống không phải Microsoft: Mặc dù ASP.NET Core có sẵn trên nhiều nền tảng, phiên bản truyền thống của ASP.NET có hạn chế khi phát triển trên các hệ thống không phải Microsoft. Điều này có thể hạn chế khả năng mở rộng và tương thích với môi trường không Microsoft.
* Tính khái quát hóa thấp: ASP.NET trước đây là một nền tảng phát triển web cụ thể dành cho môi trường Windows, do đó không được phổ biến rộng rãi như một số nền tảng khác như PHP hoặc Node.js.

1.3 Tổng quan đồ án

Từ những lý do ở mục 1.2 nên chúng em chọn đồ án “NGHIÊN CỨU, THIẾT KẾ TRẠM KHÍ TƯỢNG GIÁM SÁT VÀ CẢNH BÁO TỪ XA”.

1.3.1 Sơ đồ khối hệ thống

Ta sẽ tiến hành thiết kế mô hình Trạm Khí Tượng có tác dụng thu thập dữ liệu cảm biến hiển thị lên webserver và cảnh báo thời tiết cho người sử dụng, ta sử dụng các module cảm biến DHT22, MQ-135, mưa để thu thập dữ liệu xong đó gửi dữ liệu tổng hợp đến ESP32. Dữ liệu thu thập được từ môi trường sẽ được hiển thị và kèm theo cảnh báo lên web server đồng thời lưu trữ dữ liệu đó.

Ta có sơ đồ khối giải pháp:

A picture containing text, diagram, plan, technical drawing

Description automatically generated

Hình 1.3.1 Sơ đồ khối giải pháp

Chức năng của các khối:

**Khối nguồn:** Cung cấp nguồn hoạt động cho toàn bộ hệ thống

**Khối cảm biến:** Thu thập dữ liệu từ môi trường xung quang và gửi dữ liệu đó đến khối xử lý trung tâm.

**Khối xử lý trung tâm:** Điều khiển các quá trình giao tiếp, nhận và xử lý dữ liệu từ khối cảm biến và gửi dữ liệu đến khối server.

**Khối server:** Nhận dữ liệu từ khối xử lý trung tâm và hiển thị thông tin, đồng thời đưa ra cảnh báo cho người dùng quan sát.

**Khối broker:** Máy chủ trung gian giữa các thiết bị truyền nhận.

**Khối OpenAI**: Nhận dữ liệu từ webserver và trả lại lời khuyên.

1.3.2 Nguyên lý hoạt động của hệ thống

Nguyên lý hoạt động của hệ thống dự án được mô tả như sau: Hệ thống sử dụng Esp32 để thu thập dữ liệu từ các cảm biến thông qua các chân dữ liệu. Sau đó, Esp32 nén dữ liệu thành gói tin và sử dụng giao thức MQTT để gửi gói tin đến webserver.

Tại webserver, gói tin được nhận và giải nén để khôi phục lại dữ liệu ban đầu. Sau đó, dữ liệu được chuyển đến quy trình xử lý và lưu trữ thông tin tương ứng. Quá trình xử lý này có thể bao gồm các bước như phân tích, tính toán, lưu trữ vào cơ sở dữ liệu, và xử lý logic khác tùy theo yêu cầu của dự án.

Cuối cùng, thông tin được trả về cho người dùng thông qua giao diện người dùng của ứng dụng web. Các thông tin hiển thị có thể được trình bày dưới dạng các biểu đồ, bảng dữ liệu hoặc các hiển thị tương tác khác để cung cấp thông tin một cách trực quan và dễ hiểu cho người dùng.

Như vậy, dự án kết hợp giữa việc thu thập dữ liệu từ các cảm biến, sử dụng giao thức MQTT để truyền dữ liệu, xử lý và lưu trữ thông tin tại webserver, và cuối cùng cung cấp các thông tin hiển thị cho người dùng. Qua đó, dự án mang lại giải pháp toàn diện và tiện ích trong việc giám sát và quản lý dữ liệu từ các cảm biến.

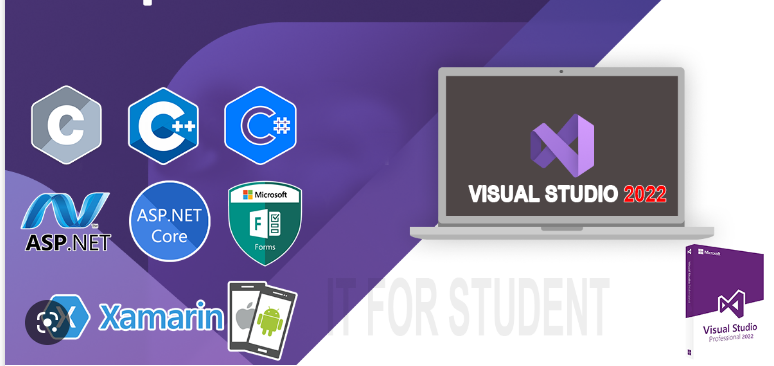
CHƯƠNG II: GIẢI PHÁP CÔNG NGHỆ THIẾT KẾ

2.1. Công nghệ phần mềm

Ta sử dụng các phần mềm Visual Studio 2019 để thiết kế thuật toán để phát triển web, Sql 2018 phát triển database và phần mềm Arduino IDE cho KIT ESP 32.

2.1.1. Giới thiệu về Visual Studio 2019

Visual Studio Code 2019 (hay còn gọi là VS Code) là một trình biên tập mã nguồn mở, phát triển bởi Microsoft. Được phát hành lần đầu vào năm 2015, VS Code đã nhanh chóng trở thành một trong những trình biên tập mã phổ biến nhất hiện nay. VS Code hỗ trợ nhiều ngôn ngữ lập trình như C++, C#, Java, JavaScript, Python, Ruby, HTML, CSS và nhiều hơn nữa.



Hình 2.1.1.1 Hình ảnh về phần mềm Visual Studio 2019

2.1.2. Phần mềm Arduino IDE

Arduino IDE là một phần chính hãng, giúp bạn viết code để nạp vào bo mạch một cách nhanh chóng, dễ dàng và hoàn toàn miễn phí.

Thông tin của Arduino IDE:

**Nền tảng:** [Windows](https://www.thegioididong.com/game-app/windows), [MacOS](https://www.thegioididong.com/game-app/macos), [Linux](https://simple.wikipedia.org/wiki/Linux)

**Dung lượng:** 530MB

**Nhà phát hành:** [Arduino Software](https://www.arduino.cc/en/software)

Một số tính năng chính của Arduino IDE:

Là phần mềm lập trình mã nguồn mở miễn phí.

Hỗ trợ lập trình tốt cho bo mạch Arduino.

Thư viện hỗ trợ phong phú và đa dạng.

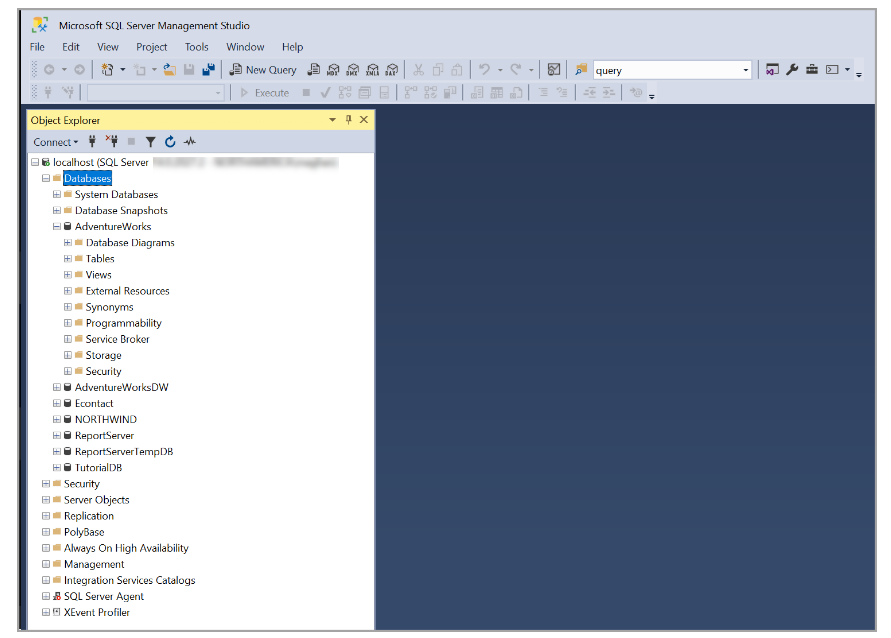


Hình 2.1.2.1 biểu tượng của Arduino IDE

2.1.3. Phần mềm Sql 2019

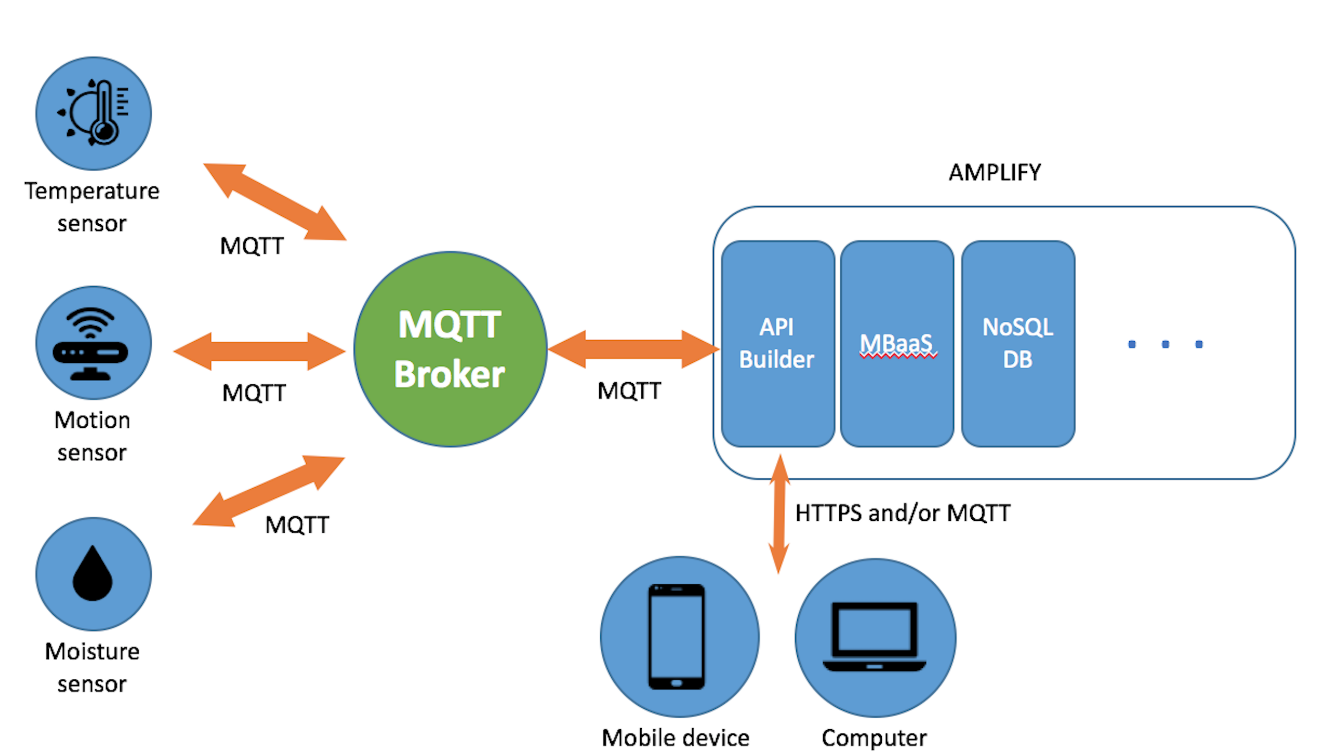
SQL Server 2019 là một hệ quản trị cơ sở dữ liệu quan hệ được phát triển bởi Microsoft. Được phát hành vào năm 2019, SQL Server 2019 là phiên bản mới nhất của SQL Server.

SQL Server 2019 hỗ trợ nhiều ngôn ngữ lập trình như C++, Java, Python và .NET. Nó cũng hỗ trợ các hệ điều hành phổ biến như Windows và Linux.



Hình 2.1.3.1 Giao diện của SQL Server 2019

2.1.4. Sơ lược về MQTT



Hình 2.1.4.1 Giao thức MQTT

MQTT (Message Queuing Telemetry Transport) là một giao thức truyền thông nhẹ và độ tin cậy cao, được sử dụng để truyền dữ liệu giữa các thiết bị IoT (Internet of Things). MQTT được thiết kế để hoạt động trên các mạng không đáng tin cậy, có băng thông thấp hoặc không ổn định, vì vậy nó là lựa chọn phổ biến cho các ứng dụng IoT.

**Phương thức truyền MQTT:**

Phương thức truyền MQTT sử dụng mô hình publisher-subscriber, trong đó các thiết bị (hay các ứng dụng) được phân loại thành hai nhóm: publisher (người gửi) và subscriber (người nhận). Publisher sẽ gửi các thông điệp (message) tới một chủ đề (topic) nhất định, trong khi đó subscriber sẽ đăng ký để nhận các thông điệp từ một hoặc nhiều chủ đề.

Các thông điệp được truyền bằng giao thức TCP/IP, với các thông điệp được mã hóa và giải mã bằng cách sử dụng các giao thức mã hóa khác nhau (ví dụ như TLS hoặc SSL). MQTT cung cấp ba cấp độ chất lượng dịch vụ (Quality of Service - QoS) khác nhau để đảm bảo tính tin cậy của việc truyền thông điệp. Các cấp độ này bao gồm:

QoS 0: Truyền thông điệp một lần, không đảm bảo tính tin cậy.

QoS 1: Truyền thông điệp ít nhất một lần, đảm bảo tính tin cậy nhưng có thể gây ra sự trùng lặp trong quá trình truyền.

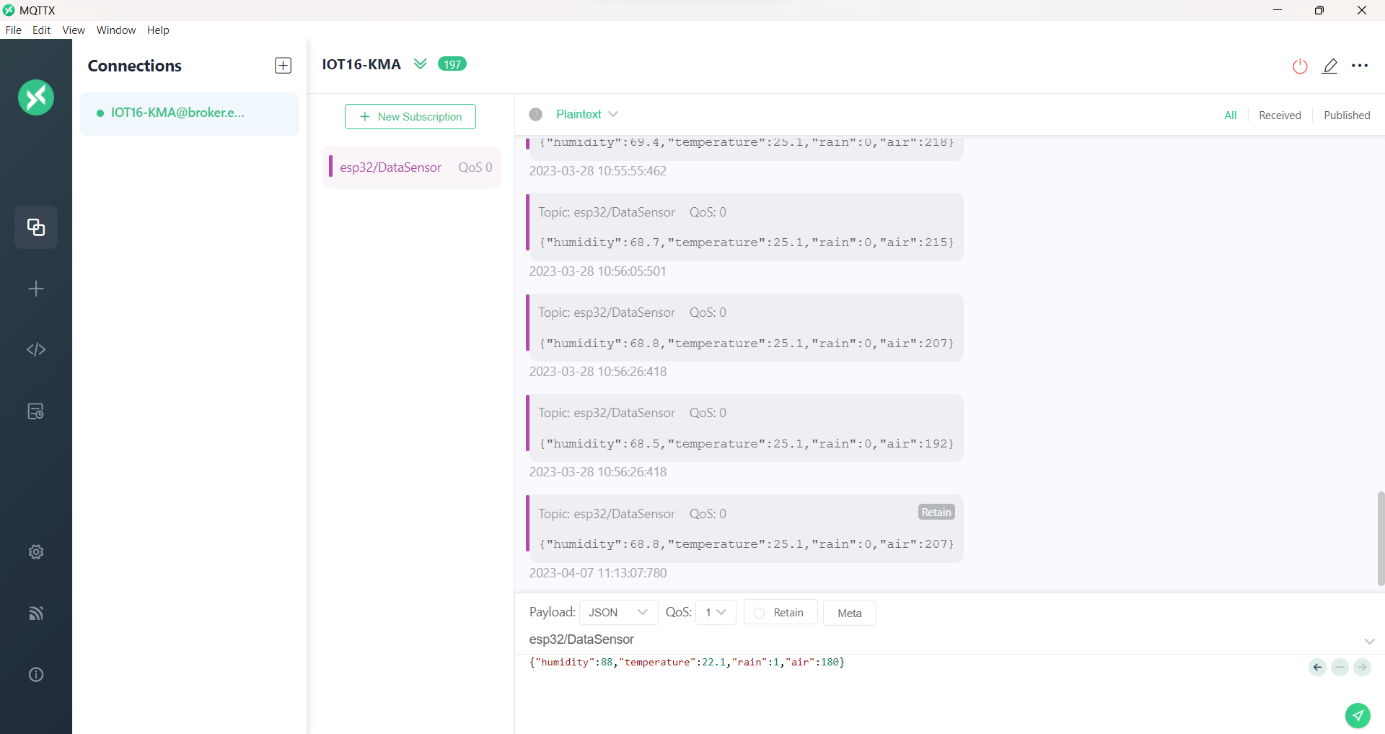
QoS 2: Truyền thông điệp chính xác một lần, đảm bảo tính tin cậy tuyệt đối nhưng tốn nhiều băng thông hơn.

MQTT cũng hỗ trợ các tùy chọn "last will and testament" để cung cấp khả năng xử lý các trường hợp mất kết nối. Khi một thiết bị bị mất kết nối với broker, nó có thể gửi một thông điệp đến các subscriber thông qua một chủ đề đặc biệt được đăng ký trước đó.

2.1.5. Phần mềm MQTTX

MQTTX là một phần mềm mã nguồn mở được sử dụng để kiểm tra và gửi tin nhắn MQTT (Message Queuing Telemetry Transport) từ các thiết bị IoT (Internet of Things) hoặc máy tính. MQTT là một giao thức truyền thông định hướng nhẹ nhàng cho các ứng dụng IoT, cho phép các thiết bị IoT gửi và nhận dữ liệu thông qua Internet.

MQTTX cung cấp một giao diện đồ họa đơn giản và dễ sử dụng cho người dùng, cho phép họ kết nối với một máy chủ MQTT, đăng ký và đăng nhập vào các kênh, gửi và nhận tin nhắn MQTT, và theo dõi các hoạt động MQTT.

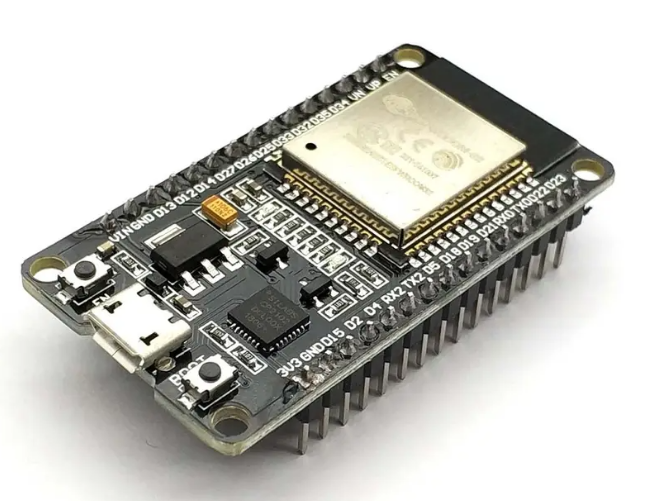


Hình 2.1.5.2 Hình ảnh về MQTTx

2.2. Công nghệ phần cứng

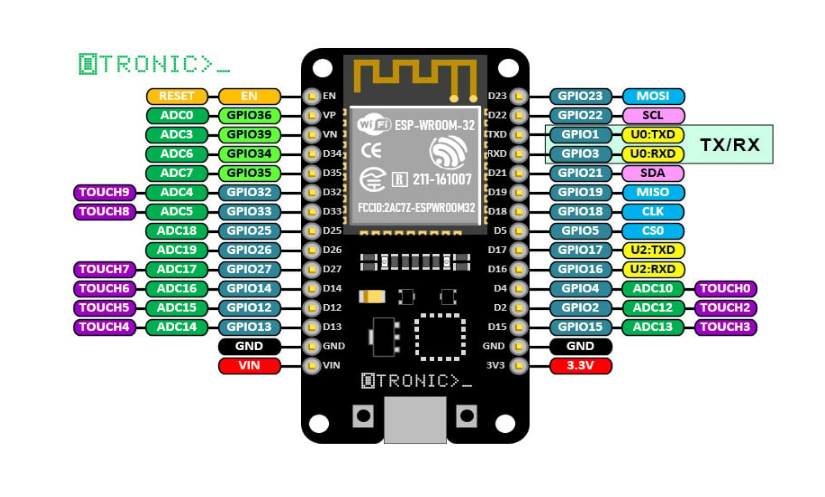
2.2.1. Kit ESP32

ESP32 là một dòng sản phẩm vi điều khiển dựa trên kiến trúc RISC-V, được sản xuất bởi Tập đoàn Espressif Systems. Đây là phiên bản nâng cấp của ESP8266, có khả năng xử lý cao hơn và tính năng mạng phát triển hơn. Sản phẩm được ra mắt vào năm 2016 và được sử dụng rộng rãi trong các ứng dụng Internet of Things (IoT)



Hình 2.2.1.1 Hình ảnh về ESP32

Sơ đồ chân ESP32



Hình 2.2.1.2 Sơ đồ chân ESP32

Sơ đồ chân của ESP32 có tổng cộng 30 chân, bao gồm 2 chân đất và 2 chân nguồn. Các chân còn lại được phân thành các nhóm chức năng khác nhau như GPIO, I2C, UART, SPI, ADC, DAC, PWM và các kết nối nguồn khác. Bên cạnh đó, ESP32 còn có các chân đặc biệt để hỗ trợ chức năng Wi-Fi và Bluetooth.

Thông số kỹ thuật của ESP32 bao gồm:

|  |  |
| --- | --- |
| CPU | Dual-core, 32-bitLX6 microprocessors |
| Tốc độ xử lý | 240 MHz |
| Bộ nhớ | 4MB flash và 520KB SRAM |
| Wi-Fi | chuẩn Wi-Fi 802.11 b/g/n, hỗ trợ các chế độ Wi-Fi: AP, STA, AP+STA |
| Bluetooth | chuẩn Bluetooth 4.2 và Bluetooth BLE |
| Ngõ vào điện áp | 5V DC qua cổng micro-USB hoặc VCC pin |
| Điện áp hoạt động | 3.3V DC |

Bảng 2.2.1.1 Thông số kĩ thuật của ESP32

Ứng dụng của ESP32:

Internet of Things (IoT): ESP32 được sử dụng trong các thiết bị như các cảm biến, thiết bị đo lường, thiết bị theo dõi, máy tính nhúng,..

Công nghiệp: ESP32 có thể được sử dụng để giám sát và kiểm soát các thiết bị công nghiệp.

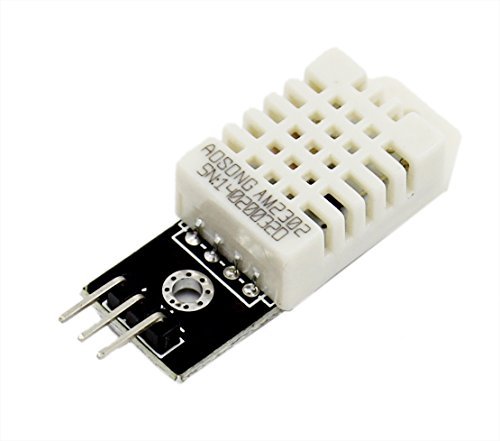
Thiết bị thông minh: ESP32 có thể được sử dụng trong các thiết bị như camera an ninh, bảng điều khiển thông minh,…

Điện tử DIY: ESP32 có thể được sử dụng để phát triển các dự án điện tử DIY như điều khiển đèn thông minh, thiết bị âm thanh,…

ESP32 là một sản phẩm vi điều khiển mạnh mẽ và tiện ích, với các tính năng đa dạng và ứng dụng rộng rãi.

2.2.2. Module cảm biến độ ẩm, nhiệt độ DHT22

Module cảm biến độ ẩm, nhiệt độ DHT22 là một trong những loại cảm biến đo độ ẩm và nhiệt độ phổ biến trong các ứng dụng IoT và các dự án liên quan đến đo lường môi trường.



Hình 2.2.2.1 Hình ảnh về module cảm biến DHT22

|  |  |
| --- | --- |
| Tên chân | Mô tả |
| Vcc | Nối nguồn |
| Data | Đầu ra cả nhiệt độ độ ẩm thông qua dữ liệu nối tiếp |
| GND | Nối đất |

Bảng 2.2.2.1 Sơ đồ chân DHT22

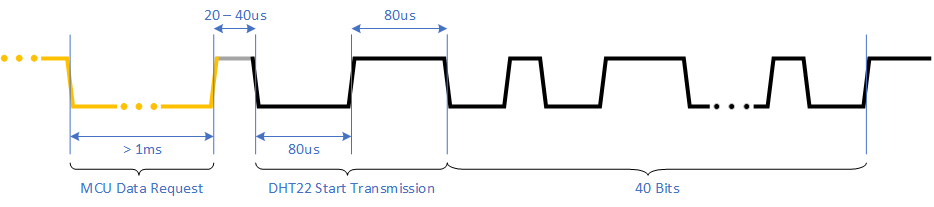
Thông số kỹ thuật của cảm biến DHT22 bao gồm:

|  |  |
| --- | --- |
| Điện áp hoạt động | 3.3V đến 5V |
| Dòng tiêu thụ | 2.5mA |
| Dải đo nhiệt độ | Từ -40°C đến 80 °C |
| Dải đo độ ẩm | Từ 0 - 100%RH |
| Sai số đo lường | ±2% RH và ±0.5°C |
| Chu kì đo | Tối đa 2 lần / giây |

Bảng 2.2.2.2 Thông số kĩ thuật của module cảm biến DHT22

Cách hoạt động: khi người dùng yêu cầu đo điện áp sẽ được cấp vào đầu dò và cảm biến sẽ đo nhiệt độ, độ ẩm của môi trường xung quanh. Sau đó giá trị nhiệt độ, độ ẩm sẽ được chuyển đổi thành tín hiệu số và được truyền tải đến một vi điều khiển hoặc các thiết bị khác qua giao thức 1 wire.

Đầu ra được cung cấp bởi chân dữ liệu sẽ theo thứ tự dữ liệu số nguyên độ ẩm 8bit + 8bit dữ liệu thập phân độ ẩm + dữ liệu số nguyên nhiệt độ 8 bit + dữ liệu nhiệt độ phân số 8 bit + bit chẵn lẻ 8 bit. Để yêu cầu mô-đun DHT11 gửi những dữ liệu này, chân I/O phải được đặt ở mức thấp trong giây lát và sau đó được giữ ở mức cao.



Hình 2.2.2.2 Giản đồ thời gian của module DHT22

2.2.3. Module cảm biến chất lượng không khí MQ-135

Module cảm biến không khí MQ-135 là một loại cảm biến khí đa năng, sử dụng để đo lường chất lượng không khí trong môi trường xung quanh. Nó được phát triển bởi tập đoàn nghiên cứu và sản xuất đo lường khí của Trung Quốc.



Hình 2.2.3.1 Hình ảnh về module cảm biến chất lượng không khí MQ-135

Sơ đồ chân:

|  |  |
| --- | --- |
| Tên chân | Mô tả |
| Vcc | Nối nguồn |
| A0 | Đầu ra tín hiệu analog |
| D0 | Đầu ra tín hiệu digital |
| GND | Nối đất |

Bảng 2.2.3.1 Sơ đồ chân của module MQ-135

Thông số kỹ thuật của module cảm biến MQ-135 bao gồm:

|  |  |
| --- | --- |
| Điện áp hoạt động | 5VDC |
| Dòng thiêu thụ | 150mA |
| Thời gian làm nóng | 20 giây |
| Nhiệt độ hoạt động | -10°C đến 50°C |
| Độ ẩm hoạt động | 10% đến 90% RH |

Bảng 2.2.3.2 Thông số kỹ thuật module cảm biến MQ135

Cách hoạt động: khi không khí bị nhiễm bởi các chất độc hại, chúng sẽ phản ứng với một lớp màng chuyển tiếp dẫn điện trên bề mặt của cảm biến MQ135, thay đổi trở kháng của cảm biến và tạo ra một tín hiệu điện.

2.2.4. Module cảm biến mưa

Module cảm biến nước mưa được sử dụng để phát hiện mưa, nước hoặc các dung dịch dẫn điện tiếp xúc với bề mặt cảm biến sẽ phát ra tín hiệu để làm các ứng dụng tự động như phát hiện mưa, báo mực nước tự động.



Hình 2.2.4.1 Hình ảnh về cảm biến mưa

Module cảm biến mưa gồm có 4 chân: Vcc – nối nguồn, GND – nối đất, A0 – đầu ra tín hiệu analog, D0 – đầu ra tìn hiệu digital.

Thông số kỹ thuật của cảm biến mưa bao gồm:

|  |  |
| --- | --- |
| Điện áp hoạt động | 5VDC |
| Dòng tiêu thụ | 20mA |
| Đầu ra tín hiệu | Kỹ thuật số hoặc tương tự |
| Nhiệt độ hoạt động | 0°C đến 50°C |
| Độ ẩm hoạt động | 10% đến 90% RH |

Bảng 2.2.4.2 Bảng thông số kĩ thuật của module cảm biến mưa

Module cảm biến mưa hoạt động dựa trên nguyên lý điện trở, khi mưa rơi vào bề mặt của cảm biến, nước sẽ làm giảm khả năng dẫn điện của cảm biến, điều này được đọc bởi bộ điều chỉnh để sản xuất tín hiệu tương ứng.

CHƯƠNG III: XÂY DỰNG VÀ THẾT KẾ SẢN PHẨM

3.1. Thiết kế phần cứng

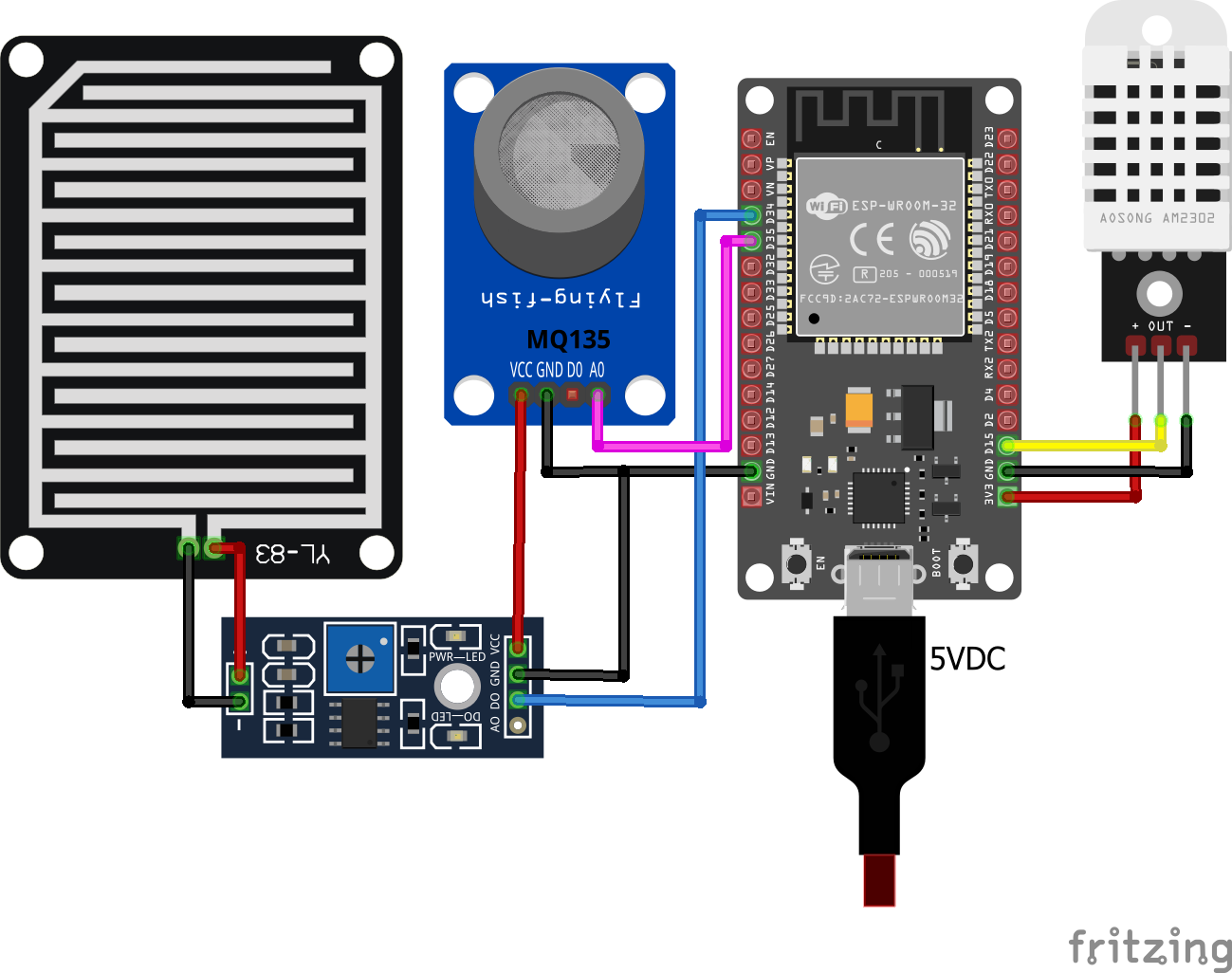
3.1.1. Lựa chọn linh kiện

Để thực hiện đề tài ta cần các linh kiện sau đây:

|  |  |  |
| --- | --- | --- |
| STT | Tên | Số Lượng |
| 1 | Kit ESP32 | 1 |
| 2 | Module cảm biến MQ135 | 1 |
| 3 | Module cảm biến DHT22 | 1 |
| 4 | Module cảm biến mưa | 1 |
| 5 | Adapter 5V 1A | 1 |
| … | … | … |

Bảng 3.1.1.1 Các linh kiện cần chuẩn bị

3.1.2. Sơ đồ nguyên lý



Hình 3.1.2.1 Sơ đồ nguyên lý toàn hệ thống

3.2. Thiết kế phần mềm

Sau khi hoàn thành xong phần cứng của sản phẩm, ta tiến hành vẽ lưu đồ thuật toán của toàn hệ thống và chuyền dữ liệu mqtt. Tiếp theo là viết chương trình cho esp32 trên phần mềm Arduino IDE.

A picture containing diagram, text, plan, technical drawing

Description automatically generated

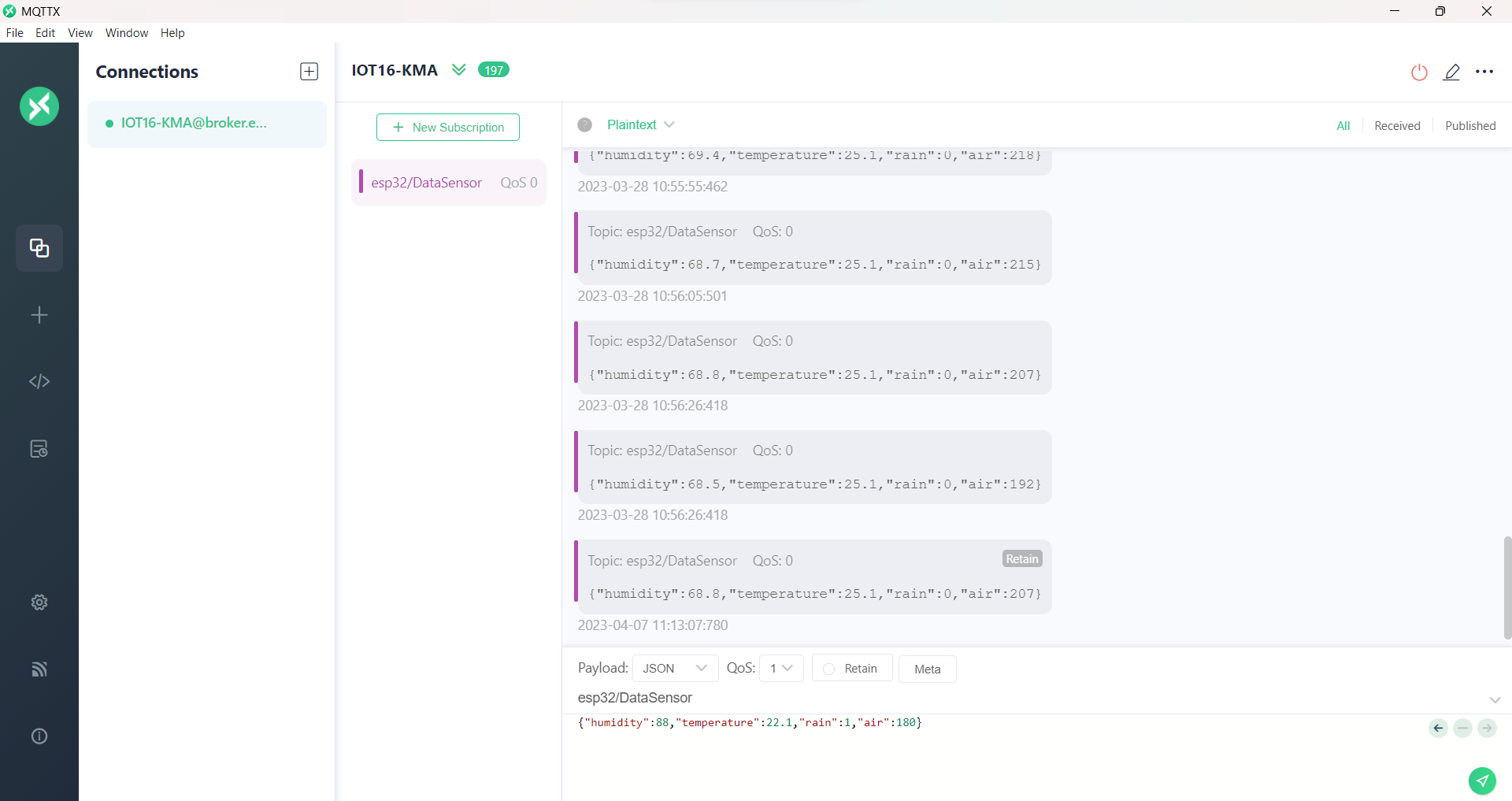
Hình 3.2.1 Lưu đồ thuật của khối xử lý trung tâm

3.3. Thực nghiệm

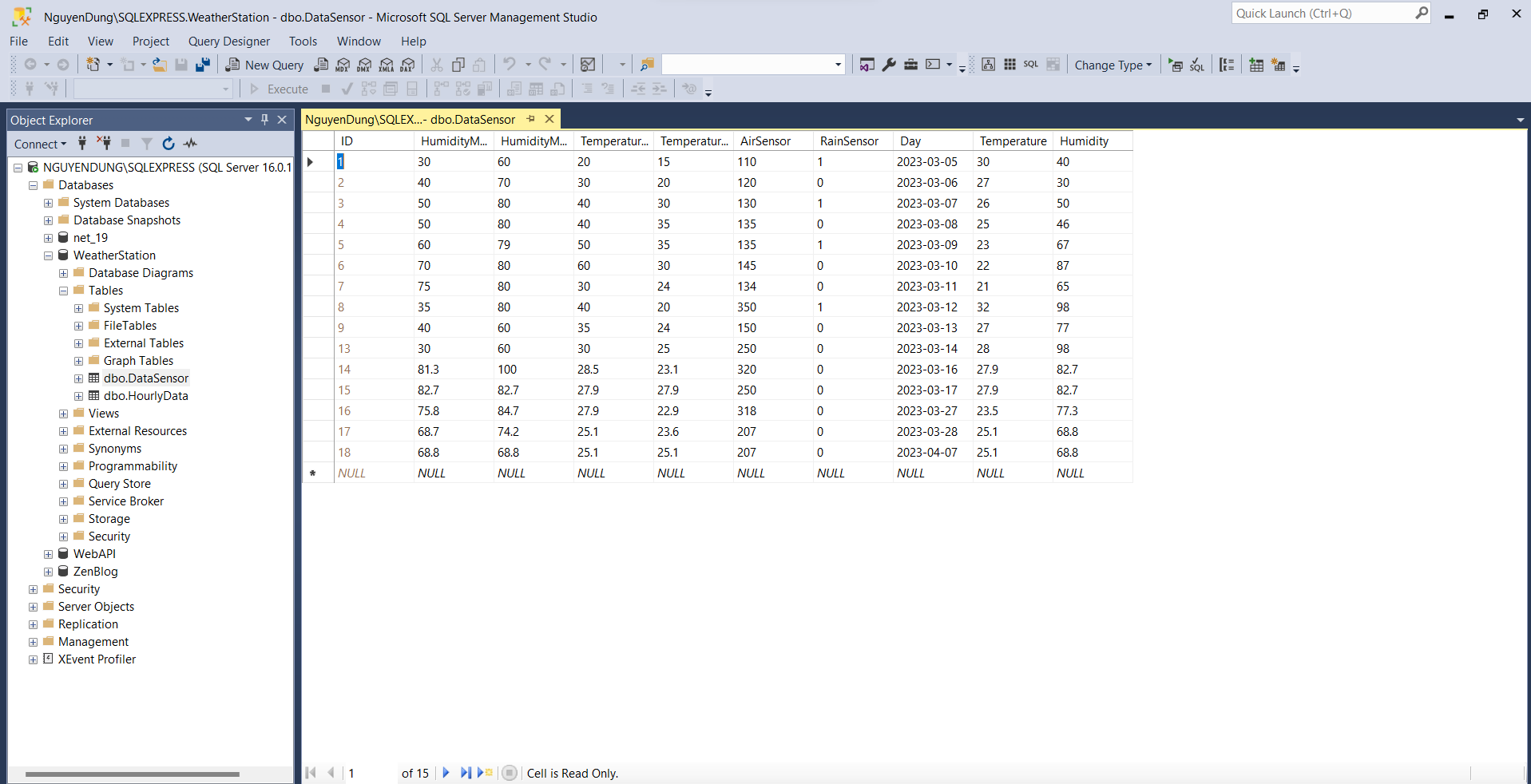
Ta tiến hành nạp code và cấp nguồn cho sản phẩm và theo dõi màn hình hiển thị trên webserver.



Hình 3.3.1 Sản phẩm hoàn thiện sau khi lắp ráp



Hình 3.3.2 Dữ liệu được gửi đến mqtt borker



Hình 3.3.3 Dữ liệu được gửi đến phần mềm SQL

A screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidenceA screen shot of a computer

Description automatically generated with low confidence

Hình 3.3.4 Các thông số thời tiết hiển thị trên web server

CHƯƠNG IV: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

4.1. Kết luận

Đạt được:

Vận dụng được các kiến thức đã học từ môn học phát triển ứng dụng iot để đáp ứng các yêu cầu chức năng của đề tài.

Xây dựng thành công các chức năng cơ bản như đọc dữ liệu từ các cảm biến, xử lý và lưu trữ dữ liệu, truyền dữ liệu qua mạng wifi và hiển thị dữ liệu trên web server.

Đề tài đã thiết lập thành công một web server để hiển thị các thông số của trạm khí tượng, đồng thời hỗ trợ cập nhật dữ liệu liên tục.

Thiếu sót:

Chưa có thử nghiệm và kiểm tra đầy đủ để đảm bảo tính ổn định và độ tin cậy của hệ thống.

Chưa tối ưu hoá được hiệu suất của hệ thống để đáp ứng được một số yêu cầu đặc biệt như xử lý dữ liệu nhanh chóng hoặc tiết kiệm năng lượng.

Tính thẩm mỹ của sản phẩm còn chưa cao, chưa đạt yêu cầu mong muốn.

4.2. Hướng phát triển

Nghiên cứu và tích hợp thêm các tính năng mới như cảm biến môi trường khác, kết nối với các thiết bị thông minh khác như smartphone.

Phát triển các tính năng mở rộng như tạo biểu đồ và báo cáo phân tích dữ liệu, tích hợp hệ thống cảnh báo hoặc tích hợp với hệ thống quản lý.

Tối ưu hoá hiệu suất của hệ thống để đáp ứng được các yêu cầu đặc biệt như tiết kiệm năng lượng hay xử lý dữ liệu nhanh chóng.

Phát triển ứng dụng di động để người dùng có thể xem các thông số của trạm khí tượng bất cứ khi nào và ở bất cứ đâu.

TÀI LIỆU THAM KHẢO

Datasheet for DHT22 sensor. (n.d.). Components101. Retrieved from <https://components101.com/sensors/dht22-pinout-specs-datasheet>

MQ135 Gas Sensor for Air Quality. (n.d.). Components101. Retrieved [from https://components101.com/sensors/mq135-gas-sensor-for-air-quality](file:///C:\Users\nguye\Downloads\from%20https:\components101.com\sensors\mq135-gas-sensor-for-air-quality)

Rain Drop Sensor Module. (n.d.). Components101. Retrieved from <https://components101.com/sensors/rain-drop-sensor-module>

ESP32 Devkit GPIO Pinout. (2018, December 31). Circuits4You. Retrieved from <https://circuits4you.com/2018/12/31/esp32-devkit-esp32-wroom-gpio-pinout/>

MQTT Server là gì? Tìm hiểu chi tiết về giao thức MQTT (28, 11 2022). Retrieved from <https://www.bkns.vn/mqtt-server-la-gi-tim-hieu-chi-tiet-ve-giao-thuc-mqtt.html>

PHỤ LỤC

Chương trình trên Arduino IDE sử dụng cho esp32

|  |
| --- |
| #include <WiFi.h>  #include <PubSubClient.h>  #include <ArduinoJson.h>  #include<MQ135.h>  #include <DHT.h>  #include <stdlib.h>  // WiFi  const char \*ssid = "nDung1930";  const char \*password = "NDung2001@";  // MQTT Broker  const char \*mqtt\_broker = "broker.emqx.io";  const char \*mqtt\_username = "admin";  const char \*mqtt\_password = "1234";  const int mqtt\_port = 1883;  const char \*DataSensor = "esp32/DataSensor";  // Pin of Sensor  #define PIN\_MQ135 35  #define DHTPIN 15  #define RAIN\_SENSOR 34  #define DHTTYPE DHT22  //Variable Sensor  float humidity = 0, temperature = 0, AirSensor = 0;  int RAIN\_SENSOR\_Value = 0;  //JsonSend - send to Broker  String jsondata = "";  // JsonReceive - receive to Broker  String JsonReceive = "";  WiFiClient espClient;  PubSubClient client(espClient);  DHT dht(DHTPIN, DHTTYPE);  void ConnectToWifi() {  WiFi.begin(ssid, password);  while (WiFi.status() != WL\_CONNECTED) {  delay(500);  Serial.println("Connecting to WiFi..");  }  Serial.println("Connected to the WiFi network");  }  void ConnectToBorker() {  client.setServer(mqtt\_broker, mqtt\_port);  client.setCallback(callback);  while (!client.connected()) {  String client\_id = "clientId-F0SIVDycNX";  //client\_id += String(WiFi.macAddress());  Serial.printf("The client %s connects to the public mqtt broker: ", client\_id.c\_str());  if (client.connect(client\_id.c\_str(), mqtt\_username, mqtt\_password)) {  Serial.println("Mqtt broker connected");  } else {  Serial.print("failed with state: ");  Serial.print(client.state());  delay(2000);  }  }  }  void setup() {  Serial.begin(115200);  dht.begin();  pinMode(RAIN\_SENSOR, INPUT);  pinMode(PIN\_MQ135, INPUT );  ConnectToWifi();  ConnectToBorker();  }  //function callback: receive data from broker  void callback(char \*topic, byte \*payload, unsigned int length) {  Serial.print("Message arrived in topic: ");  Serial.println(topic);  String topicStr = topic;  for (int i = 0; i < length; i++) {  JsonReceive += (char) payload[i];  }  Serial.print("JsonReceive: ");  Serial.println(JsonReceive);  Serial.println("-----------------------");  JsonReceive = "";  }  void getValueOfSensor() {  AirSensor = analogRead(PIN\_MQ135);  RAIN\_SENSOR\_Value = digitalRead(RAIN\_SENSOR);  humidity = dht.readHumidity();  temperature = dht.readTemperature();  }  void send\_data() {  getValueOfSensor();  DynamicJsonBuffer jBuffer;  JsonObject& root = jBuffer.createObject();  root["humidity"] = humidity;  root["temperature"] = temperature;  root["rain"] = RAIN\_SENSOR\_Value;  root["air"] = AirSensor;  root.printTo(jsondata);  client.publish(DataSensor, jsondata.c\_str(), true );  Serial.println(jsondata);  jsondata = "";  }  void loop() {  client.loop();  send\_data();  delay(10000);  } |