ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN

ĐỒ ÁN TỐT NGHIỆP
CAPSTONE PROJECT

NGÀNH: KỸ THUẬT ĐIỀU KHIỂN VÀ TỰ ĐỘNG HÓA

ĐỀ TÀI:

THIẾT KẾ DATALOGGER CHO HỆ THỐNG QUAN TRẮC MÔI TRƯỜNG KHÔNG KHÍ

Người hướng dẫn: **TS. NGUYỄN QUỐC ĐỊNH**
Sinh viên thực hiện:
NGUYỄN TUẤN MINH – MSSV: 105200459 – LỚP: 20TDHCLC3
NGUYỄN LÊ NGHĨA – MSSV: 105200373 – LỚP: 20TDHCLC1
HỒ ĐẮC THẮNG – MSSV: 105200471 – LỚP: 20TDHCLC3

Đà Nẵng, 6/2025

TÓM TẮT

Tên đề tài: Thiết kế Datalogger cho hệ thống quan trắc môi trường không khí.

SVTH: Nguyễn Tuấn Minh	MSSV: 105200459	Lớp : 20TDHCLC3
Nguyễn Lê Nghĩa	MSSV: 105200373	Lớp : 20TDHCLC1
Hồ Đắc Thắng	MSSV: 105200471	Lớp : 20TDHCLC3

Đề tài nghiên cứu này tập trung vào việc thiết kế một hệ thống datalogger nhằn giám sát và thu thập dữ liệu môi trường (cụ thể là các dữ liệu không khí) phục vụ cho các ứng dụng quan trắc trong công nghiệp và bảo vệ môi trường. Với tình trạng biến đổi khí hậu cũng như nhu cầu quản lý dữ liệu môi trường ngày càng tăng nên việc phát triển hệ thống này là vô cùng cấp thiết.

Mục tiêu của nghiên cứu là xây dựng một hệ thống datalogger ứng dụng vào quan trắc chất lượng môi trường không khí có khả năng thu thập các thông số từ các cảm biến, lưu trữ dữ liệu, hỗ trợ theo dõi và phân tích dữ liệu theo thời gian thực. Hệ thống sử dụng Arduino Uno để giao tiếp với các cảm biến bụi PM2.5, PM10, nhiệt độ, độ ẩm, khí CO và NO_2; sau đó truyền dữ liệu về Raspberry Pi 5 để xử lý và lưu trữ. Ngoài việc tính toán chỉ số AQI, hệ thống còn tích hợp mạng nơ-ron LSTM để dự đoán xu hướng chất lượng không khí. Dữ liệu được hiển thị trực quan thông qua WebApp, cho phép truy cập từ xa qua mạng Wi-Fi hoặc Ethernet.

Các kết quả thực nghiệm cho thấy thiết bị datalogger hoạt động ổn định, có khả năng thu thập dữ liệu từ các cảm biến khác nhau và truyền tải dữ liệu đến hệ thống giám sát từ xa.

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

TT	Họ tên sinh viên	Số thẻ SV	Lớp	Ngành
1	Nguyễn Tuấn Minh	105200459	20TDHCLC3	Kỹ thuật điều khiển và tự động hoá
2	Nguyễn Lê Nghĩa	105200373	20TDHCLC1	Kỹ thuật điều khiển và tự động hoá
3	Hồ Đắc Thắng	105200471	20TDHCLC3	Kỹ thuật điều khiển và tự động hoá

1. *Tên đề tài đồ án:*

Thiết kế Datalogger cho hệ thống quan trắc môi trường không khí

2. *Đề tài thuộc diện: ☐Có ký kết thỏa thuận sở hữu trí tuệ đối với kết quả thực hiện*
3. *Các số liệu và dữ liệu ban đầu:*

……..……..……...
...……
…..……..……..……

4. *Nội dung các phần thuyết minh và tính toán:*

a. Phần chung:

TT	Họ tên sinh viên	Nội dung
1	Nguyễn Tuấn Minh	Tìm hiểu tổng quan đề tài : mục tiêu, phạm vi, nội dung thực hiện đề tài Tìm hiểu cơ sở lý thuyết : bộ thu thập dữ liệu, các ngưỡng đánh giá chỉ số, sơ đồ hệ thống, các công cụ hỗ trợ thiết kế phần mềm
2	Nguyễn Lê Nghĩa	
3	Hồ Đắc Thắng	

b. Phần riêng:

TT	Họ tên sinh viên	Nội dung
1	Nguyễn Tuấn Minh	Thiết kế huấn luyện mô hình mạng nơ-ron dự đoán thông số AQI và đưa ra cảnh báo trong tương lai Thiết kế giao diện Webapp
2	Nguyễn Lê Nghĩa	Tìm hiểu yêu cầu, kiến trúc phần cứng, lựa chọn, phân tích các cảm biến, các bộ điều khiển, mạch nguồn và lắp đặt mô hình thực tế

3	Hồ Đắc Thắng	Nghiên cứu kiến trúc, cách thu thập, xử lí, lưu trữ, truyền, hiển thị dữ liệu, lập trình trên các bộ điều khiển

5. Các bản vẽ, đồ thị (ghi rõ các loại và kích thước bản vẽ):

a. Phần chung:

TT	Họ tên sinh viên	Nội dung
1	Nguyễn Tuấn Minh	
2	Nguyễn Lê Nghĩa	
3	Hồ Đắc Thắng	

b. Phần riêng:

TT	Họ tên sinh viên	Nội dung
1	Nguyễn Tuấn Minh	Sơ đồ kiến trúc mạng nơ-ron, Sơ đồ khối phần mềm quản lý dữ liệu
2	Nguyễn Lê Nghĩa	Sơ đồ cụ thể hệ thống Sơ đồ đấu nối các cảm biến
3	Hồ Đắc Thắng	Lưu đồ thuật toán đọc và tính toán dữ liệu trên Arduino Lưu đồ thuật toán Raspberry đọc dữ liệu từ Arduino Lưu đồ quá trình thu thập và ghi dữ liệu cảm biếm vào cơ sở dữ liệu

6. Họ tên người hướng dẫn:	*Phần/ Nội dung:*
TS Nguyễn Quốc Định	Định hướng đề tài Mô hình huấn luyện mạng nơ-ron Thu thập dữ liệu để huấn luyện

7. Ngày giao nhiệm vụ đồ án: 6/3/2025

8. Ngày hoàn thành đồ án: 26/6/2025

Đà Nẵng, ngày tháng 6 năm 2025

Trưởng Bộ môn Tự động hóa **Người hướng dẫn**

TS. Giáp Quang Huy

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

PHIẾU KIỂM SOÁT TIẾN ĐỘ LÀM ĐỒ ÁN TỐT NGHIỆP

(Phiếu dành cho người hướng dẫn/sinh viên)

Họ tên sinh viên:

Nguyễn Tuấn Minh	MSSV: 105200459	Lớp : 20TDHCLC3
Nguyễn Lê Nghĩa	MSSV: 105200373	Lớp : 20TDHCLC1
Hồ Đắc Thắng	MSSV: 105200471	Lớp : 20TDHCLC3

Tên đề tài ĐATN:

Thiết kế Datalogger cho hệ thống quan trắc môi trường không khí

Họ tên người HD: TS Nguyễn Quốc Định

Đơn vị: Khoa Điện Trường Đại học Bách khoa – Đại học Đà Nẵng

Tuần	**Ngày**	**Khối lượng**		**GVHD ký tên**
		đã thực hiện (%)	**tiếp tục thực hiện (%)**	
1	6/3	Nhận đề tài và nghiên cứu cơ bản đề tài (10%)	Nghiên cứ tài liệu, phân tích yêu cầu, xây dựng hướng thiết kế	
2	10/3	Phân tích hệ thống, lập sơ đồ tổng quan hệ thống (20%)	Tìm hiểu, thiết kế phần cứng	
3	18/3	Duyệt lần 1: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
4	24/3	Tìm hiểu phần mềm và các ngôn ngữ lập trình (30%)	Tìm hiểu lựa chọn kiến trúc phần cứng	
5	2/4	Duyệt lần 2: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
6	7/4	Lựa chọn kiến trúc phần cứng (35%)	Tìm hiểu kết nối phần cứng	
7	14/4	Hoàn thiện kết nối phần cứng (45%)	Tìm hiểu lập trình phần mềm đọc dữ liệu cảm biến	
8	21/4	Duyệt lần 3: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		

9	28/4	Nghiên cứu xử lý và lưu trữ dữ liệu (55%)	Hoàn thiện truyền dữ liệu và hiển thị dữ liệu	
10	5/5	Tìm hiểu mô hình mạng nơ-ron hỗ trợ dự đoán chất lượng không khí (70%)	Hoàn thiện mô hình mạng nơ-ron và thực hiện kết nối	
11	15/5	Duyệt lần 4: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
12	1/6	Hoàn thiện kết nối, huấn luyện và đánh giá hiệu suất mô hình (85%)	Thiết kế giao diện Web và quản lý dữ liệu	
13	8/6	Hoàn thiện giao diện Web và tổng hợp kết quả đề tài (95%)	Kiểm tra lại các phần và thực hiện viết báo cáo hoàn chỉnh	
14	16/6	Hoàn thiện báo cáo và kiểm tra đầy đủ các phần (100%)	Chuẩn bị bảo vệ ĐATN	
15	24/6	Bảo vệ ĐATN, hoàn thành nhiệm vụ.		

LỜI NÓI ĐẦU

Trong những năm gần đây, ô nhiễm không khí đang trở thành vấn đề đáng lo ngại tại các khu công nghiệp và đô thị lớn. Việc xây dựng hệ thống quan trắc môi trường tự động, hoạt động ổn định và có khả năng cảnh báo sớm là yêu cầu cấp thiết trong công tác quản lý và bảo vệ môi trường. Đề tài "Thiết kế Data Logger cho hệ thống quan trắc môi trường không khí" được thực hiện với mong muốn ứng dụng kiến thức chuyên ngành vào việc xây dựng một thiết bị có khả năng thu thập, lưu trữ và truyền dữ liệu từ các cảm biến không khí, phục vụ việc giám sát chất lượng không khí một cách hiệu quả và liên tục.

Trong quá trình thực hiện, em đã tìm hiểu, lựa chọn giải pháp phần cứng phù hợp với nền tảng Raspberry Pi, tích hợp các giao thức truyền thông và cảm biến môi trường nhằm xây dựng một mô hình Datalogger có tính ứng dụng cao. Mặc dù đã nỗ lực hoàn thành tốt nhất trong khả năng, nhưng do thời gian và kinh nghiệm thực tế còn hạn chế, đồ án khó tránh khỏi thiếu sót. Em kính mong nhận được sự góp ý từ quý thầy cô để hoàn thiện hơn trong tương lai.

Em xin chân thành cảm ơn quý thầy cô trong khoa đã tận tình giảng dạy và đặc biệt là giáo viên hướng dẫn đã hỗ trợ, định hướng và tạo điều kiện thuận lợi để em hoàn thành đồ án này

LỜI CẢM ƠN

Để có thể hoàn thành đồ án tốt nghiệp của chúng tôi, bên cạnh sự cố gắng của mỗi cá nhân, chúng tôi xin được gửi lời cảm ơn chân thành đến quý thầy cô Trường Đại học Bách Khoa – Đại học Đà Nẵng đã tạo điều kiện thuận lợi cho chúng tôi học tập và thực hiện đề tài nghiên cứu này. Đặc biệt chúng tôi xin bày tỏ lòng biết ơn sâu sắc đến TS. Nguyễn Quốc Định đã chỉ dạy và hướng dẫn chúng tôi vô cùng tận tình trong suốt quá trình thực hiện đề tài. Đồng thời chúng tôi nhận được rất nhiều sự quan tâm, giúp đỡ từ gia đình và bạn bè đã không ngừng động viên trong quá trình hoàn thành đề tài.

Trong suốt quá trình thực hiện đề tài tốt nghiệp, nhờ vào kiến thức chuyên ngành trong suốt thời gian học tập mà chúng tôi có thể vận dụng vào quá trình thực hiện đề tài này, từ đó giúp chúng tôi học hỏi được rất nhiều điều bổ ích. Những kiến thức mà chúng tôi tiếp thu được sẽ là hành trang giúp chúng tôi vững bước trong tương lai.

Tuy nhiên, do vốn kiến thức còn hạn hẹp nên bài nghiên cứu khó có thể tránh khỏi những thiếu sót. Chúng tôi mong có thể nhận được những góp ý quý báu từ thầy cô cũng như các bạn học để đề tài nghiên cứu của chúng tôi có thể hoàn thiện hơn nữa.

Chúng tôi xin chân thành cảm ơn !

LỜI CAM ĐOAN LIÊM CHÍNH HỌC THUẬT

Chúng tôi xin cam đoan bản thuyết minh này được hoàn thành dựa trên các kết quả nghiên cứu của chúng tôi trong khuôn khổ của đề tài : 'Thiết kế datalogger cho hệ thống quan trắc môi trường không khí'. Dự án có quyền sử dụng kết quả của bản thuyết minh này để phục vụ cho dự án. Các số liệu, kết quả trình bày trong bản thuyết minh là trung thực và chưa từng được ai công bố trong bất kì bản thuyết minh nào trước đây.

Sinh viên thực hiện 1 Sinh viên thực hiện 2 Sinh viên thực hiện 3

MỤC LỤC

DANH SÁCH CÁC BẢNG

DANH SÁCH HÌNH ẢNH

DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT

Chữ/ký hiệu viết tắt	Diễn giải đầy đủ
IoT	Internet of Things
LoRa	Long Range
AI	Artificial Intelligence
CAN	Controller Area Network
GPIO	General Purpose Input/Output
SQL	Structured Query Language
GPRS	General Packet Radio Service
AQI	Air Quality Index
PC	Personal Computer
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
API	Application Programming Interface
PHP	Hypertext Preprocessor
VSCode	Visual Studio Code
CO	Carbon Monoxide
NO_2	Nitrogen Dioxide
IDE	Integrated Development Environment
ADC	Analog to Digital Converter
UART	Universal Asynchronous Receiver Transmitter
SD	Secure Digital
TTL	Transistor-Transistor Logic
RX	Receive
TX	Transmit
MEMS	Micro-Electro-Mechanical Systems
SKU	Stock Keeping Unit
I2C	Inter-Integrated Circuit
RH	Relative Humidity

Chữ/ký hiệu viết tắt	Diễn giải đầy đủ
SDA	Serial Data
SCL	Serial Clock
SBC	Single Board Computer
RAM	Random Access Memory
LPDDR4X	Low Power Double Data Rate 4X
ARM	Advanced RISC Machines
CSV	Comma-Separated Values
JSON	JavaScript Object Notation
USB	Universal Serial Bus
GND	Ground
AC	Alternating Current
DC	Direct Current
UHS	Ultra High Speed (thường dùng với thẻ SD)
LED	Light Emitting Diode
UTF	Unicode Transformation Format
DB	Database
LSTM	Long Short-Term Memory
RNN	Recurrent Neural Network
GRU	Gated Recurrent Unit
CNN	Convolutional Neural Network
ReLU	Rectified Linear Unit
MSE	Mean Squared Error
MAE	Mean Absolute Error
Adam (AME)	Adaptive Moment Estimation

MỞ ĐẦU

Ô nhiễm không khí ngày càng trở nên nghiêm trọng, đặc biệt tại các khu công nghiệp và đô thị hóa nhanh. Trong khi đó, các hệ thống quan trắc truyền thống thường có chi phí cao và thiếu tính linh hoạt. Trước thực trạng đó, việc xây dựng một hệ thống giám sát môi trường chi phí thấp, dễ triển khai và có khả năng cảnh báo sớm là rất cần thiết.

Đề tài "Thiết kế Datalogger cho hệ thống quan trắc môi trường không khí" nhằm xây dựng một hệ thống có khả năng thu thập, lưu trữ và truyền dữ liệu từ các cảm biến môi trường như bụi PM2.5, PM10, khí CO, NO_2, nhiệt độ và độ ẩm. Hệ thống sử dụng Arduino Uno để đọc dữ liệu cảm biến, truyền về Raspberry Pi 5 để xử lý, tính toán AQI và hiển thị qua WebApp. Đặc biệt, mạng nơ-ron LSTM được tích hợp để dự đoán xu hướng chất lượng không khí.

Mục tiêu của đề tài là thiết kế một hệ thống hoạt động ổn định, dễ mở rộng và tích hợp, phù hợp với môi trường công nghiệp. Đề tài được thực hiện thông qua các bước: nghiên cứu lý thuyết, thiết kế phần cứng – phần mềm, lập trình, và thử nghiệm mô hình thực tế.

Kết quả đạt được góp phần cung cấp một giải pháp giám sát môi trường chủ động, kết hợp giữa IoT và trí tuệ nhân tạo, hướng tới khả năng triển khai thực tế trong các khu công nghiệp và ứng dụng cộng đồng.

Cấu trúc của đồ án tốt nghiệp gồm các nội dung sau

- Tổng quan đề tài
- Cơ sở lý thuyết
- Thiết kế phần cứng
- Thiết kế phần mềm
- Mô hình mạng nơ-ron và giao diện Webapp
- Kết quả
- Kết luận và hướng phát triển

CHƯƠNG 1 : TỔNG QUAN ĐỀ TÀI

1.1. Đặt vấn đề

- Trong bối cảnh biến đổi khí hậu và sự gia tăng các hoạt động công nghiệp, môi trường không khí tại các khu công nghiệp đang phải đối mặt với nhiều thách thức nghiêm trọng. Các nguồn phát thải từ nhà máy, phương tiện giao thông, và hoạt động sản xuất công nghiệp đang gây ra tình trạng ô nhiễm không khí, ảnh hưởng tiêu cực đến sức khỏe con người và môi trường xung quanh. Việc giám sát và thu thập dữ liệu chất lượng không khí đóng vai trò quan trọng trong việc đánh giá tác động và đưa ra các giải pháp kiểm soát kịp thời. Tuy nhiên, hiện nay, các hệ thống quan trắc không khí thường gặp hạn chế về khả năng tùy biến, chi phí cao và khó triển khai trên diện rộng, đặc biệt trong các khu công nghiệp có điều kiện kinh tế còn hạn chế.
- Trong bối cảnh đó, việc thiết kế một hệ thống quan trắc môi trường không khí khu chuyên dụng, có khả năng thu thập và lưu trữ dữ liệu môi trường một cách chính xác, ổn định, và dễ dàng tích hợp với các công nghệ hiện đại là cấp thiết. Hệ thống này không chỉ hỗ trợ giám sát chất lượng không khí theo thời gian thực mà còn cung cấp cơ sở dữ liệu đáng tin cậy để phân tích, dự đoán và đưa ra các cảnh báo kịp thời.
- Với mục tiêu đáp ứng các vấn đề trên, đề tài "Thiết kế datalogger cho hệ thống quan trắc môi trường không khí" được thực hiện nhằm phát triển một giải pháp hiệu quả, ứng dụng được trong nhiều lĩnh vực như kiểm soát ô nhiễm công nghiệp, bảo vệ môi trường và sức khỏe cộng đồng. Hệ thống được kỳ vọng sẽ đóng góp vào việc nâng cao năng lực giám sát môi trường, từ đó giúp các cá nhân, tổ chức đưa ra các quyết định quản lý bền vững và hiệu quả.

1.2. Lịch sử phát triển

Hệ thống quan trắc không khí đã được nghiên cứu và phát triển trong nhiều thập kỷ nhằm hỗ trợ việc giám sát, đánh giá và quản lý chất lượng không khí. Ban đầu, các hệ thống này chủ yếu dựa trên các phương pháp đo đạc thủ công với thiết bị cơ học hoặc điện tử đơn giản. Tuy nhiên, sự phát triển nhanh chóng của khoa học công nghệ đã thúc đẩy các bước tiến lớn trong lĩnh vực này.

Giai đoạn đầu (Trước thập niên 1990):

- Các hệ thống quan trắc không khí chủ yếu sử dụng cảm biến điện tử đơn lẻ, kết hợp với các thiết bị ghi dữ liệu cơ học như băng giấy hoặc bộ nhớ đơn giản.
- Khả năng lưu trữ và xử lý dữ liệu còn hạn chế, không thể đáp ứng nhu cầu phân tích thời gian thực hoặc tích hợp hệ thống.

Thời kỳ chuyển giao (Thập niên 1990 - 2000):

- Sự ra đời của máy tính cá nhân và công nghệ vi xử lý đã mở ra hướng mới cho các hệ thống quan trắc.
- Các datalogger sử dụng vi xử lý tích hợp đã xuất hiện, cho phép thu thập và lưu trữ dữ liệu tự động trong thời gian dài.
- Tuy nhiên, các hệ thống này vẫn còn hạn chế về khả năng kết nối, khiến việc truyền xuất và chia sẻ dữ liệu gặp khó khăn.

Giai đoạn phát triển (Từ 2000 đến nay):

- Công nghệ IoT (Internet of Things) bùng nổ, cùng với các tiến bộ về cảm biến không dây (Wi-Fi, LORA, Zigbee) và lưu trữ dữ liệu trên nền tảng đám mây.
- Các datalogger hiện đại không chỉ nhận dữ liệu mà còn cung cấp khả năng giám sát từ xa, xử lý dữ liệu theo thời gian thực và đưa ra cảnh báo kịp thời.
- Sự phổ biến của các nền tảng mã nguồn mở như Raspberry Pi, Arduino đã giúp giảm chi phí và tăng tính linh hoạt trong thiết kế hệ thống.

Xu hướng hiện tại và tương lai:

- Hướng đến hệ thống datalogger có khả năng xử lý thông minh, tích hợp trí tuệ nhân tạo (AI) để phân tích và dự đoán xu hướng ô nhiễm.
- Nâng cao khả năng bảo mật dữ liệu và tối ưu hóa năng lượng cho các thiết bị hoạt động trong khu vực rộng lớn, vùng xa.
- Mở rộng ứng dụng trong các lĩnh vực như công nghiệp sản xuất, kiểm soát khí thải nhà máy, và giám sát biến đổi chất lượng không khí theo thời gian thực.

Với bối cảnh đó, đề tài "Thiết kế datalogger cho hệ thống quan trắc môi trường không khí" được xây dựng dựa trên các tiến bộ công nghệ hiện đại, kế thừa những thành tựu từ các nghiên cứu trước đây và tập trung giải quyết các hạn chế hiện có. Hệ thống hướng đến một giải pháp hiệu quả, giúp các cơ quan và tổ chức dễ dàng triển khai trong thực tế.

1.3. Mục tiêu và phạm vi.

1.3.1. Mục tiêu của đề tài

- Mục tiêu của đề tài là thiết kế và xây dựng một datalogger có hiệu năng cao, hoạt động ổn định, linh hoạt trong giao tiếp, dễ tích hợp và mở rộng trong môi trường

công nghiệp. Phạm vi nghiên cứu tập trung ở quy mô mô hình thử nghiệm, ưu tiên các thông số môi trường cơ bản, chưa triển khai diện rộng. Đối tượng nghiên cứu bao gồm phần cứng Raspberry Pi 5, Arduino Uno, các cảm biến môi trường, module giao tiếp và phần mềm xử lý, lưu trữ, truyền dữ liệu.

- Trong đó, phần cứng được xây dựng với các cổng giao tiếp linh hoạt, hỗ trợ các giao thức truyền thông công nghiệp, đảm bảo khả năng tích hợp với các loại cảm biến không khí hiện có trên thị trường.
- Phần mềm được thiết kế với giao diện giám sát từ xa, cho phép người dùng theo dõi các thông số chất lượng không khí theo thời gian thực trên nền tảng website, đồng thời hỗ trợ lưu trữ dữ liệu để phục vụ phân tích và quản lý lâu dài.
- Tích hợp mạng nơ-ron nhân tạo LSTM để dự đoán xu hướng chỉ số AQI trong tương lai, giúp mở rộng khả năng của hệ thống từ giám sát sang cảnh báo sớm.
- Điểm nhấn của đề tài là ứng dụng hệ thống này vào khu công nghiệp, nhằm giám sát các thông số quan trọng như nồng độ bụi PM2.5, PM10, khí CO, NO_2, độ ẩm và nhiệt độ không khí, từ đó đánh giá mức độ ô nhiễm và ảnh hưởng đến sức khỏe con người. Chỉ tiết các mục tiêu này sẽ được trình bày trong các phần tiếp theo để làm rõ tính ứng dụng, giúp xác định cái nhìn chi tiết hơn về đề tài.

1.3.2. Phạm vi đề tài và ứng dụng

Đề tài "Thiết kế Datalogger cho hệ thống quan trắc môi trường không khí" được thực hiện với mục đích xây dựng một hệ thống có khả năng thu thập, lưu trữ và truyền dữ liệu từ các cảm biến môi trường như nhiệt độ, độ ẩm, bụi mịn PM10, PM2.5, khí CO, NO_2,… nhằm phục vụ công tác giám sát và cảnh báo chất lượng không khí tại các khu công nghiệp.

Hệ thống sử dụng nền tảng Raspberry Pi 5 làm trung tâm xử lý, cho phép kết nối linh hoạt với nhiều loại cảm biến thông qua các chuẩn giao tiếp như UART, I2C hoặc SPI, đồng thời hỗ trợ truyền dữ liệu về máy chủ hoặc hệ thống giám sát tập trung thông qua Wi-Fi, Ethernet hoặc các giao thức mở rộng như RS485/LoRa. Dữ liệu được tổ chức lưu trữ theo thời gian thực trong cơ sở dữ liệu MySQL và hiển thị thông qua WebApp. Ngoài ra, hệ thống còn tích hợp mô hình mạng nơ-ron LSTM để dự đoán chỉ số AQI, góp phần nâng cao khả năng cảnh báo sớm.

Phạm vi nghiên cứu tập trung ở quy mô mô hình thử nghiệm trong phòng, ưu tiên các thông số môi trường cơ bản và chưa triển khai diện rộng ngoài thực tế. Hệ thống mới chỉ hoạt động trong điều kiện có nguồn điện ổn định, chưa tích hợp nguồn năng lượng dự phòng hoặc tái tạo.

Đối tượng nghiên cứu bao gồm phần cứng Raspberry Pi 5, Arduino Uno, các cảm biến môi trường, module giao tiếp và phần mềm xử lý, lưu trữ, truyền dữ liệu, WebApp hiển thị và mô hình học máy LSTM.

Phạm vi ứng dụng của hệ thống bao gồm:

- Giám sát môi trường không khí tại các khu công nghiệp, cụm nhà máy sản xuất.
- Ứng dụng trong giáo dục và nghiên cứu tại các trường đại học, phòng thí nghiệm.
- Là tiền đề để phát triển các hệ thống quan trắc môi trường giá rẻ cho cộng đồng, trường học, đô thị hoặc các vùng dân cư bị ô nhiễm.

1.4. Nội dung thực hiện

Để đáp ứng nhu cầu giám sát môi trường không khí trong khu công nghiệp, đảm bảo khả năng hoạt động lâu dài, giám sát liên tục và có tính mở rộng, đề tài được triển khai theo các nội dung chính sau:

- Thiết kế phần cứng: Xây dựng sơ đồ nguyên lý kết nối giữa Raspberry Pi 5, Arduino Uno và các cảm biến môi trường (PM2.5, PM10, CO, NO_2, DHT22). Thiết kế mạch nguồn ổn định, hỗ trợ các cổng giao tiếp chuẩn công nghiệp như UART, I2C, SPI và có thể mở rộng với RS485 hoặc LoRa.
- Phát triển phần mềm: Lập trình Arduino để thu thập dữ liệu cảm biến, xử lý sơ bộ và gửi về Raspberry Pi. Xây dựng chương trình trên Raspberry Pi để lưu trữ dữ liệu vào cơ sở dữ liệu MySQL, xử lý và tính toán chỉ số AQI. Thiết kế giao diện WebApp thân thiện, hiển thị dữ liệu theo thời gian thực và cho phép truy xuất lịch sử.
- Tích hợp trí tuệ nhân tạo: Sử dụng mô hình mạng nơ-ron hồi tiếp LSTM để dự đoán xu hướng chỉ số AQI từ dữ liệu thu thập. Mô hình được huấn luyện từ dữ liệu thực tế và đánh giá bằng các chỉ số MSE, MAE để kiểm chứng độ chính xác.
- Thử nghiệm và đánh giá: Tiến hành kiểm tra hệ thống trong môi trường mô phỏng khu công nghiệp. Thu thập dữ liệu liên tục từ cảm biến, so sánh với giá trị tham chiếu để đánh giá hiệu quả hoạt động và độ ổn định của hệ thống. Kiểm tra tính khả dụng của WebApp và tốc độ phản hồi của mô hình AI.

Với cách tiếp cận toàn diện từ phần cứng đến phần mềm và AI, hệ thống không chỉ đảm bảo tính khả thi trong việc giám sát chất lượng không khí mà còn hướng đến khả năng ứng dụng thực tế, mở rộng quy mô và tích hợp vào các hệ thống quản lý môi trường thông minh.

CHƯƠNG 2 : CƠ SỞ LÝ THUYẾT

2.1. Quan trắc môi trường

2.1.1. Khái niệm

Quan trắc môi trường là hoạt động theo dõi, giám sát chất lượng môi trường định kỳ thông qua các chỉ tiêu về tính chất vật lý và hóa học của thành phần môi trường. Quá trình đo lường sẽ cung cấp các đánh giá cần thiết về những tác động và chuyển biến của môi trường ở từng khoảng thời gian khác nhau. Mục đích của việc thực hiện quan trắc môi trường nhằm phân tích môi trường đang bị ảnh hưởng ở mức độ nào và hoạt động sản xuất kinh doanh có tác động như thế nào đến môi trường.

Hình 2.1 Quan trắc môi trường

Trong đó, trạm quan trắc môi trường được thực hiện ở nhiều không gian và các hình thức đa dạng khác nhau như quan trắc môi trường nước, quan trắc môi trường nước thải, quan trắc môi trường không khí xung quanh và quan trắc môi trường khí thải. Từ đó, đạt được những mục tiêu chung trong việc đánh giá diễn biến của mọi khía cạnh môi trường trong phạm vi quốc gia hay nắm bắt tình hình cụ thể của từng môi trường để đưa ra những giải pháp cụ thể và có những cảnh báo kịp thời tới những diễn biến bất thường hoặc có nguy cơ gây ô nhiễm, ảnh hưởng tới thực trạng môi trường chung.

2.1.2. Vai trò

Hệ thống quan trắc môi trường không khí đóng vai trò quan trọng trong việc giám sát chất lượng môi trường, hỗ trợ công tác quản lý và đưa ra các biện pháp kiểm soát ô nhiễm hiệu quả.

- Giám sát và đánh giá chất lượng không khí: Hệ thống quan trắc cung cấp dữ liệu chính xác và liên tục về các thông số môi trường như nồng độ bụi mịn (PM2.5, PM10), khí thải công nghiệp (CO, NO_2). Những thông tin này giúp đánh giá mức độ ô nhiễm và tác động của hoạt động công nghiệp đối với môi trường xung quanh.
- Cảnh báo sớm và phòng ngừa ô nhiễm: Hệ thống quan trắc không khí có khả năng phát hiện các xu hướng ô nhiễm bất thường, từ đó đưa ra cảnh báo sớm để kịp thời xử lý, giảm thiểu tác động tiêu cực đến sức khỏe con người và hệ sinh thái. Việc kết hợp giữa quan trắc không khí và mô hình dự báo giúp đưa ra các biện pháp kiểm soát hiệu quả.
- Xác định nguồn phát thải ô nhiễm: Dữ liệu từ hệ thống quan trắc giúp xác định chính xác nguồn gốc và mức độ ô nhiễm do các nhà máy, khu công nghiệp gây ra. Từ đó, các cơ quan quản lý có thể đưa ra quy định kiểm soát khí thải phù hợp, yêu cầu các doanh nghiệp áp dụng công nghệ xử lý khí thải tiên tiến để giảm thiểu ô nhiễm.
- Hỗ trợ xây dựng chính sách bảo vệ môi trường: Thông tin thu thập được từ hệ thống quan trắc không khí là cơ sở quan trọng để xây dựng các chính sách và quy định về bảo vệ môi trường trong khu công nghiệp. Việc phân tích dữ liệu quan trắc giúp đánh giá hiệu quả của các biện pháp kiểm soát ô nhiễm và đề xuất các giải pháp cải thiện chất lượng không khí.

Hệ thống quan trắc môi trường không khí khu không chỉ đóng vai trò đo lường mà còn là công cụ hỗ trợ ra quyết định trong quản lý môi trường. Việc tích hợp công nghệ IoT và trí tuệ nhân tạo vào hệ thống quan trắc giúp nâng cao hiệu quả giám sát, cảnh báo sớm và đưa ra các giải pháp bảo vệ môi trường một cách kịp thời.

2.2. Bộ thu thấp dữ liệu

2.2.1. Giới thiệu về datalogger

Datalogger là thiết bị tự động thu thập, ghi nhận và lưu trữ dữ liệu từ các cảm biến hoặc nguồn dữ liệu khác. Với khả năng hoạt động liên tục trong thời gian dài, datalogger đóng vai trò quan trọng trong việc giám sát các thông số môi trường, công

nghiệp và nông nghiệp. Dữ liệu được lưu trữ có thể được phân tích trực tiếp hoặc truyền về máy chủ qua các giao thức không dây như Wi-Fi, GPRS hoặc LORA để quản lý và xử lý từ xa.

2.2.2. Chức năng

Hệ thống datalogger đảm nhận nhiều chức năng quan trọng để đảm bảo việc giám sát và quản lý dữ liệu môi trường. Cụ thể, các chức năng của hệ thống bao gồm:

- Thu thập dữ liệu từ các cảm biến.
- Lưu trữ dữ liệu tạm thời hoặc lâu dài.
- Truyền dữ liệu về máy chủ hoặc đám mây thông qua các giao thức như không dây như Wi-Fi, LORA, GPRS.
- Hỗ trợ cảnh báo sự cố, bao gồm các trường hợp mất kết nối hoặc khi giá trị vượt ngưỡng.

2.2.3. Các tiêu chí thiết kế datalogger

Bên cạnh các chức năng cơ bản, hệ thống datalogger cần đáp ứng một số tiêu chí thiết kế quan trọng để đảm bảo hiệu suất và tính linh hoạt trong ứng dụng. Các tiêu chí này bao gồm:

- Độ bền và khả năng hoạt động liên tục.
- Kết nối không dây ổn định.
- Tính tương thích với nhiều loại cảm biến và các giao thức truyền thông khác nhau.
- Khả năng mở rộng và nâng cấp linh hoạt về cả phần cứng và phần mềm.

2.3. Các ngưỡng chỉ số đánh giá

2.3.1. Các ngưỡng đánh giá chỉ số

a) Thang đo AQI

Bảng 2.1 Bảng giá trị AQI theo nồng độ của các chất ô nhiễm

AQI	Mức độ không khí	PM2.5 (µg/m³)	PM10 (µg/m³)	NO_2 (µg/m³)	CO (mg/m³)
0 - 50	Tốt	0 - 12	0 - 54	0 - 40	0 - 5
51 - 100	Trung bình	12.1 - 35.4	55 - 154	41 - 100	5 - 10
101 - 150	Kém	35.5 - 55.4	155 - 254	101 - 200	10 - 20

AQI	Mức độ không khí	PM2.5 (µg/m³)	PM10 (µg/m³)	NO_2 (µg/m³)	CO (mg/m³)
151 - 200	Xấu	55.5 - 150.4	255 - 354	201 - 400	20 - 30
201 - 300	Rất xấu	150.5 - 250.4	355 - 424	>400	>30
>300	Nguy hại	>250.5	>424	-	-

b) Công thức tính AQI theo từng chất ô nhiễm

AQI của từng chất được tính theo công thức:

$$AQI_p = \frac{(I_{high} - I_{low})}{(C_{high} - C_{low})} \times (C - C_{low}) + I_{low} \qquad (1)$$

Trong đó:

- AQI_p : chỉ số AQI của chất ô nhiễm p
- C : Nồng độ đo được của chất ô nhiễm
- C_{low}, C_{high} : Ngưỡng nồng độ gần nhất bao quanh C (theo bảng AQI chuẩn)
- I_{low}, I_{high} : Giá trị AQI tương ứng với C_{low} và C_{high}

Trong đó: AQI Tổng được lấy theo chỉ số AQI cao nhất trong các chất đo được

c) Đánh giá mức độ ô nhiễm theo từng thông số quan trắc

- Bụi :

Bảng 2.2 Bảng giá trị bụi lơ lửng (TSP, PM10, PM2.5)

Thông số	Tốt	Trung bình	Ô nhiễm nhẹ	Ô nhiễm nặng	Nguy hại
TSP (µg/m³)	<100	100 - 200	200 - 300	300 - 400	>400
PM10 (µg/m³)	<50	50 - 100	100 - 150	150 - 250	>250
PM2.5 (µg/m³)	<25	25 - 50	50 - 100	100 - 150	>150

- Các khí gây ô nhiễm :

Bảng 2.3 Bảng giá trị các khí ô nhiễm

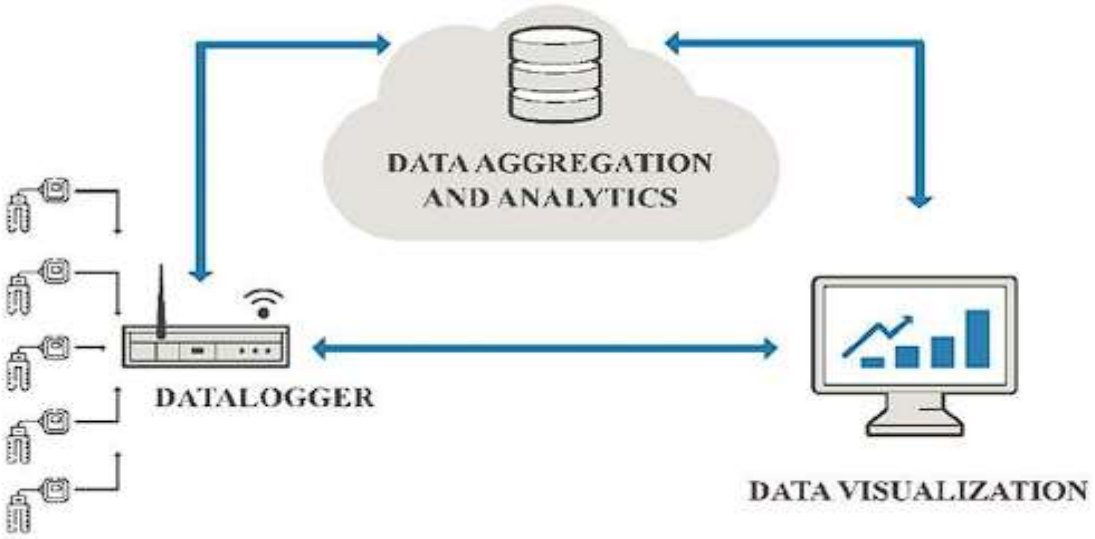
Thông số	Tốt	Trung bình	Ô nhiễm nhẹ	Ô nhiễm nặng	Nguy hại
NO_2 (µg/m³)	<40	40 - 100	100 - 200	200 - 400	>400
CO (mg/m³)	<5	5 - 10	10 - 20	20 - 30	>30

- Khí hậu :

Bảng 2.4 Bảng giá trị nhiệt độ, độ ẩm

Thông số	Tốt	Chấp nhận được	Không tốt	Nguy hại
Nhiệt độ (°C)	18 - 27	27 - 30	30 - 35	>35
Độ ẩm (%)	40 - 60	30 - 40, 60 - 70	<30 hoặc >70	-

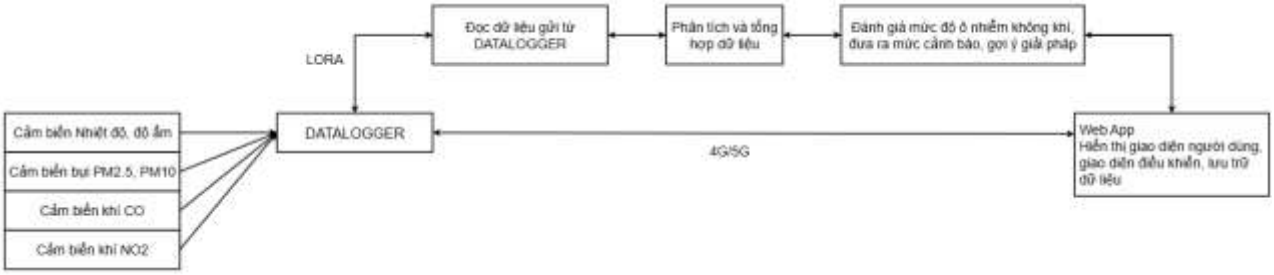
2.4. Sơ đồ hệ thống



Hình 2.2 Sơ đồ tổng quan hệ thống

- Hệ thống Datalogger quan trắc không khí ứng dung công nghê IoT, hoạt động theo mô hình sever-client
- Mỗi bộ datalogger đóng vai trò client, thực hiện thu thập dữ liêu từ các cảm biến, xử lí dữ liệu và gửi lên server.
- Server đảm nhiệm vai trò nhận dữ liệu từ các datalogger, lưu trữ và cung cấp khả năng truy xuất dữ liệu đến hệ thống giám sát

- Người dung giám sát thông số không khí và hỗ trợ cấu hình từ xa qua PC hay laptop thông qua webapp



Hình 2.3 Sơ đồ cụ thể hệ thống

2.5. Phần mềm

2.5.1. Ngôn ngữ đánh dấu siêu văn bản HTML

HTML được định nghĩa là một ngôn ngữ đánh dấu (markup language). Đánh dấu (markup) là một phương pháp để chỉ định thông tin về nội dung, Thông tin ở đây, chính là các thông tin về việc định dạng của một phần tử nội dung như hiển thị đậm, nghiêng, các thuộc tính về màu sắc,… Mục tiêu của ngôn ngữ này là tách phần định dạng với phần nội dung được định dạng. Sự tách rời này có tính tương đối và có thể có nhiều mức độ khác nhau tùy thuộc vào từng phiên bản của ngôn ngữ này

2.5.2. CSS

CSS (Cascading Style Sheet) là ngôn ngữ định kiểu theo tầng, mô tả cách trình bày của các tài liệu bằng HTML. Bao gồm cả màu sắc, bố cục và phông chữ. Với những ưu điểm như:

- Tiết kiệm thời gian: Chúng ta có thể viết CSS một lần và sau đó tái sử dụng trong nhiều trang HTML.
- Tách biệt giữa nội dung và trình bày: Bảng kiểu CSS có thể được lưu trên một tệp tin có phần mở rộng .css và sau đó sử dụng vào các trang HTML.
- Tải các trang Web về trình duyệt nhanh hơn: Việc sử dụng CSS giúp chúng ta không cần phải viết lại các thuộc tính cho các thẻ HTML.
- Dễ dàng bảo trì: Khi thực hiện một sự thay đổi toàn bộ Website, chỉ cần thay đổi trong file CSS, tất cả các phần tử trong trang Web sẽ được cập nhật tự động.
- Tạo ra trang Web đa dạng hơn trong cách trình bày.
- Tương thích nhiều thiết bị: Các bảng kiểu cho phép nội dung tối ưu hóa với nhiều thiết bị.

2.5.3. JavaScript

JavaScript là một ngôn ngữ kịch bản phổ biến dùng để viết các script cho máy khách của Web. Các script phía máy khách được thực thi trên trình duyệt trong khi phía máy chủ được thực thi trên server. Chúng được viết để tăng thêm tính tương tác của các trang Web. Chúng ta có thể viết các script để thay đổi nội dung và vị trí các phần tử từ trên một trang một cách động để phản hồi lại tương tác của người dùng. Một trong những trường hợp sử dụng JavaScript như:

- JavaScript có thể thay đổi nội dung HTML. Một trong nhiều phương thức JavaScript HTML là getElementById().
- Thay đổi giá trị thuộc tính HTML, kiểu HTML(CSS).
- Có thể ẩn và hiển thị các phần tử HTML

2.5.4. Framework

Framework là các đoạn code đã được viết sẵn, cấu thành nên một bộ khung và các thư viện lập trình được đóng gói. Chúng cung cấp các tính năng có sẵn như mô hình, API và các yếu tố khác để tối giản cho việc phát triển các ứng dụng web phong phú, năng động. Các framework giống như là chúng ta có khung nhà được làm sẵn nền móng cơ bản, bạn chỉ cần vào xây dựng và nội thất theo ý mình.
Hiện nay chưa có quy ước về cách phân loại các framework, tuy nhiên để dễ hiểu có thể chia như sau:

- Theo ngôn ngữ: JavaScript frameworks, HTML & CSS frameworks, .NET Framework,...
- Theo hệ điều hành hỗ trợ: Windows framework, Android framework, iOS framework,...
- Theo ứng dụng: Web framework, Mobile framework, Front-end framework, Back-end framework,...

2.5.5. MySQL

MySQL là một hệ thống quản lý cơ sở dữ liệu quan hệ nguồn mở do Oracle cung cấp. Nhà phát triển có thể tải xuống và sử dụng MySQL mà không phải trả phí cấp phép. Họ có thể cài đặt MySQL trên nhiều hệ điều hành hoặc máy chủ đám mây khác nhau. MySQL là một hệ thống cơ sở dữ liệu phổ biến cho các ứng dụng web.

2.5.6. SQL

Ngôn ngữ truy vấn có cấu trúc (SQL) là một ngôn ngữ lập trình phục vụ việc lưu trữ và xử lý thông tin trong cơ sở dữ liệu quan hệ. Cơ sở dữ liệu quan hệ lưu trữ thông tin dưới dạng bảng có các hàng và cột đại diện cho những thuộc tính dữ liệu và nhiều mối quan hệ khác nhau giữa các giá trị dữ liệu. Bạn có thể sử dụng các câu lệnh SQL để lưu trữ, cập nhật, loại bỏ, tìm kiếm và truy xuất thông tin từ cơ sở dữ liệu. Bạn cũng có thể sử dụng SQL để duy trì và tối ưu hóa hiệu suất cơ sở dữ liệu.

2.5.7. Ngôn ngữ PHP

PHP là viết tắt của từ Hypertext Preprocessor. Đây là một ngôn ngữ lập trình kịch bản (scripting language) mã nguồn mở và là tập hợp con của các ngôn ngữ script như JavaScript. Ngôn ngữ này chủ yếu được dùng để phát triển các ứng dụng từ phía server, bên cạnh đó còn được sử dụng cho frontend và backend.

2.5.8. Visual Studio Code

Đây là một trình soạn thảo mã nguồn miễn phí được Microsoft phát triển, hỗ trợ trên Windows, macOS và Linux. Được đánh giá cao bởi các chuyên gia công nghệ thông tin, VS Code kết hợp các tính năng của một IDE và một code editor. Nhờ đó mang đến sự linh hoạt và hiệu suất mạnh mẽ cho các lập trình viên.

VS Code không chỉ hỗ trợ các ngôn ngữ như JavaScript, TypeScript và Node.js mà còn cung cấp một hệ sinh thái mở rộng phong phú cho nhiều ngôn ngữ lập trình khác. Với dung lượng nhẹ và yêu cầu tài nguyên hệ thống thấp, đây là một công cụ lý tưởng cho việc xây dựng các ứng dụng web. Giúp lập trình viên làm việc hiệu quả mà không bị cản trở bởi những tính năng không cần thiết

CHƯƠNG 3 : THIẾT KẾ PHẦN CỨNG

3.1. Yêu cầu dữ liệu quan trắc

Để hiện thực hóa khả năng quan trắc chất lượng không khí toàn diện và chính xác cho khu công nghiệp, hệ thống datalogger được thiết kế phải có khả năng thu thập dữ liệu từ nhiều thông số môi trường quan trọng. Cụ thể, chúng tôi cần theo dõi và đo lường liên tục các chỉ số chính sau:

- **Các chỉ số bụi mịn:** PM2.5 và PM10 – những hạt siêu nhỏ có khả năng đi sâu vào hệ hô hấp, ảnh hưởng trực tiếp đến sức khỏe con người và làm giảm tầm nhìn trong môi trường công nghiệp.
- **Các khí độc hại:** Carbon Monoxide (CO) và Nitrogen Dioxide (NO2) – những chất gây ô nhiễm phổ biến, phát sinh từ các hoạt động công nghiệp và giao thông, tiềm ẩn nguy cơ gây hại nghiêm trọng cho sức khỏe.
- **Thông số khí hậu:** Nhiệt độ và Độ ẩm – các yếu tố môi trường đóng vai trò quan trọng trong việc ảnh hưởng đến sự phân tán của chất ô nhiễm và cũng cần được theo dõi liên tục để đánh giá điều kiện tổng thể.

Do đó, việc lựa chọn và tích hợp các cảm biến phù hợp để đo lường các thông số trên là bước nền tảng, định hình khả năng thu thập dữ liệu và hiệu quả hoạt động của toàn bộ hệ thống.

3.2. Kiến trúc phần cứng

Để tối ưu hóa việc thu thập dữ liệu từ đa dạng các loại cảm biến và đảm bảo tính linh hoạt, ổn định trong quá trình phát triển và vận hành, hệ thống datalogger của chúng tôi được thiết kế theo một kiến trúc phân tán .

Trong kiến trúc này, Arduino Uno đóng vai trò là bộ điều khiển trung gian (slave controller) không thể thiếu. Vai trò này được xác định rõ ràng bởi một yếu tố kỹ thuật then chốt: Do Raspberry Pi không có bộ chuyển đổi ADC (Analog-to-Digital Converter) tích hợp, nó không thể đọc trực tiếp các tín hiệu analog từ một số loại cảm biến, điển hình như các cảm biến khí CO và NO_2 mà chúng tôi sử dụng. Vì vậy, Arduino Uno, với khả năng ADC mạnh mẽ và tích hợp sẵn, là lựa chọn lý tưởng để:

- Đọc dữ liệu cảm biến: Giao tiếp trực tiếp với các cảm biến khác nhau, bao gồm PMS7003 (UART), DHT22 (1-Wire), và đặc biệt là đọc tín hiệu analog từ các cảm biến khí CO và NO_2 thông qua các chân ADC của nó.

- Xử lý sơ bộ dữ liệu: Thực hiện các tác vụ như chuyển đổi đơn vị, tính toán trung bình, lọc nhiễu ban đầu (nếu cần) để cung cấp dữ liệu sạch hơn và đã được số hóa cho Raspberry Pi.
    - Giao tiếp với Raspberry Pi: Truyền dữ liệu đã xử lý từ các cảm biến đến Raspberry Pi thông qua một giao thức truyền thông đáng tin cậy.

Sau đó, Raspberry Pi – bộ điều khiển chính (master controller) – sẽ tiếp nhận dữ liệu này, thực hiện các tác vụ xử lý sâu hơn (như phân tích nâng cao, tính toán chỉ số AQI), lưu trữ vào bộ nhớ (thẻ SD) và truyền tải dữ liệu lên máy chủ để phân tích và hiển thị trực quan.

Việc phân chia nhiệm vụ một cách rõ ràng giữa Arduino Uno và Raspberry Pi mang lại nhiều ưu điểm vượt trội:

- Giảm tải cho Raspberry Pi: Cho phép Raspberry Pi tập trung vào các tác vụ phức tạp hơn như quản lý hệ thống, lưu trữ dữ liệu dung lượng lớn và truyền thông mạng, thay vì phải xử lý các tín hiệu cấp thấp từ cảm biến.
- Đơn giản hóa tích hợp cảm biến và khắc phục hạn chế phần cứng: Tận dụng được hệ sinh thái phong phú, dễ dàng lập trình và các thư viện sẵn có của Arduino, đặc biệt là khả năng đọc tín hiệu analog, giải quyết trực tiếp hạn chế về ADC của Raspberry Pi.
- Tính linh hoạt và khả năng mở rộng: Dễ dàng thêm hoặc thay thế cảm biến mà không ảnh hưởng lớn đến toàn bộ hệ thống, đồng thời tăng cường khả năng gỡ lỗi và bảo trì.

3.3. Lựa chọn và phân tích các cảm biến quan trắc

Để hiện thực hóa khả năng quan trắc chất lượng không khí toàn diện và cung cấp dữ liệu chính xác cho các chỉ số PM2.5, PM10, CO, NO_2, nhiệt độ và độ ẩm, việc lựa chọn cảm biến đóng vai trò then chốt. Quá trình này được thực hiện cẩn trọng dựa trên các tiêu chí về độ chính xác, dải đo, độ ổn định, khả năng tương thích giao tiếp với Arduino Uno, và hiệu quả chi phí. Mỗi cảm biến được tích hợp vào hệ thống đều được đánh giá kỹ lưỡng để đảm bảo khả năng hoạt động tin cậy trong môi trường khu công nghiệp.

3.3.1. Cảm biến bụi mịn PMS7003

- Giới thiệu và nguyên lý hoạt động: Cảm biến PMS7003 thuộc dòng PMSx003 của Plantower, là một module đo lường bụi mịn sử dụng công nghệ tán xạ laser tiên tiến. Cảm biến được thiết kế để phát hiện và định lượng nồng độ các

hạt vật chất lơ lửng trong không khí có kích thước từ 0.3 micromet trở lên. Bên trong, một nguồn laser chiếu sáng dòng không khí mẫu, và khi các hạt bụi đi qua chùm tia này, ánh sáng sẽ bị tán xạ. Một bộ phận cảm quang nhạy bén sẽ thu nhận các xung ánh sáng tán xạ, từ đó, vi xử lý tích hợp trong cảm biến sẽ phân tích và tính toán chính xác nồng độ khối lượng của các hạt PM1.0, PM2.5 và PM10 theo thời gian thực.

Hình 3.1 Cảm biến đo nồng độ bụi

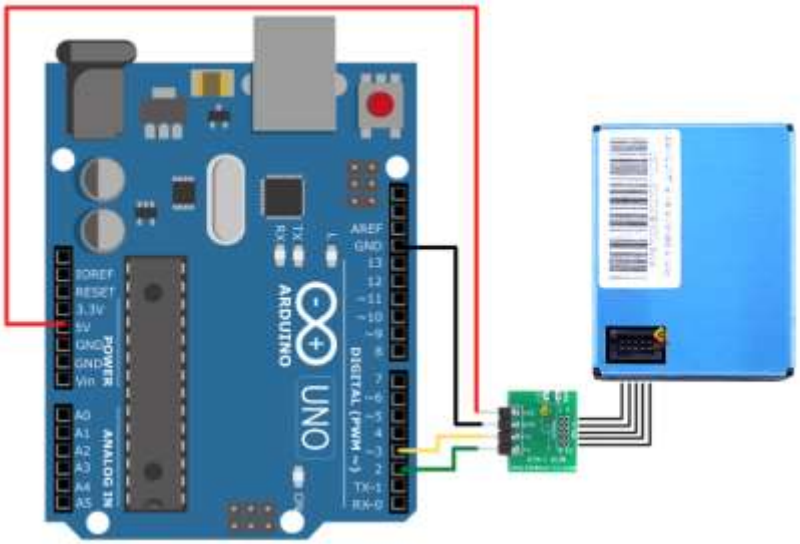
➢ THÔNG SỐ KỸ THUẬT:

- Điện áp sử dụng: 4.5~5.5VDC
- Điện áp hoạt động: 4.5–5.5VDC
- Dòng tiêu thụ: ≤100 mA
- Dòng nghỉ: 200 µA
- Giao tiếp: UART (tín hiệu TTL 3.3V)
- Kích thước đo được: 0,3–10 µm
- (chia thành 3 dải: 0,3–1,0 / 1,0–2,5 / 2,5–10 µm)
- Hiệu suất đếm: 50% @ 0,3 µm, 98% @ ≥ 0,5 µm
- Kết quả đo dải (PM2.5): 0–500 µg/m³
- Tối đa dải đo: ≥1000 µg/m³
- Độ phân giải: 1 µg/m³
- Sai số: ±10% (100–500 µg/m³), ±10 µg/m³ (0–100 µg/m³)
- Thời gian phản hồi: <1 giây (đơn lẻ), ≤10 giây (tổng)
- Nhiệt độ làm việc: -10 đến +60°C
- Độ ẩm làm việc: 0–99% RH
- Kích thước: 48 × 37 × 12 mm

- Lý do lựa chọn: PMS7003 được chọn lọc nhờ vào độ tin cậy tương đối cao trong phân khúc cảm biến dân dụng/bán công nghiệp, cung cấp dữ liệu ổn định và phù hợp

cho việc giám sát xu hướng chất lượng không khí. Giao tiếp UART đơn giản của cảm biến hoàn toàn tương thích với cổng Serial của Arduino Uno, giúp việc tích hợp và lập trình trở nên dễ dàng mà không cần thêm mạch chuyển đổi phức tạp. Khả năng cung cấp đồng thời cả PM2.5 và PM10 đáp ứng đầy đủ yêu cầu về chỉ số bụi mịn của đề tài.

- Kết nối với Arduino Uno: Cảm biến PMS7003 được kết nối với Arduino Uno thông qua giao tiếp UART, trong đó chân TX của cảm biến nối với chân RX của Arduino và chân RX của cảm biến nối với chân TX của Arduino (thường sử dụng SoftwareSerial để không chiếm cổng Serial chính cho debug). Nguồn cấp 5V được lấy trực tiếp từ Arduino Uno.



Hình 3.2 Sơ đồ đấu nối cảm biến bụi với Arduino

3.3.2. Cảm biến Nhiệt độ và Độ ẩm DHT22

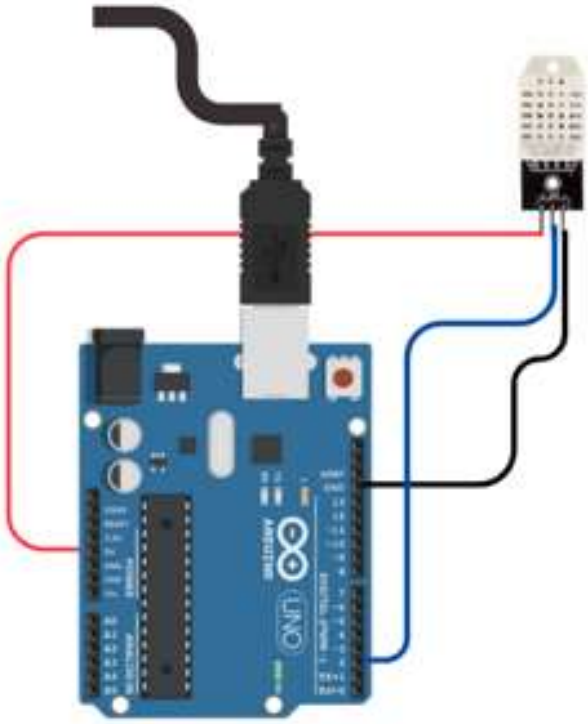
- Giới thiệu và nguyên lý hoạt động: DHT22 là một module cảm biến kỹ thuật số được sử dụng rộng rãi để đo lường đồng thời nhiệt độ và độ ẩm tương đối của môi trường. Đây là một lựa chọn phổ biến trong các dự án IoT và quan trắc môi trường do sự cân bằng giữa độ chính xác, độ ổn định và chi phí. Cảm biến tích hợp một bộ phận cảm biến độ ẩm điện dung và một thermistor để đo nhiệt độ. Một chip tích hợp bên trong sẽ chuyển đổi các giá trị đo được thành tín hiệu kỹ thuật số và truyền tải dữ

liệu thông qua một giao thức 1-Wire độc quyền, giúp đơn giản hóa việc kết nối chỉ với một chân dữ liệu.

- THÔNG SỐ KỸ THUẬT
  - Điện áp hoạt động: 5V
  - Khoảng đo độ ẩm: 0% – 100% RH sai số 2% RH
  - Khoảng đo nhiệt độ: -40 ~ 80 độ C sai số 0.5% độ C
  - Tần số lấy mẫu tối đa 0.5Hz (2 giây / lần)
  - Kích thước: 28mm x 12mm x 10mm

Hình 3.3 Cảm biến DHT22

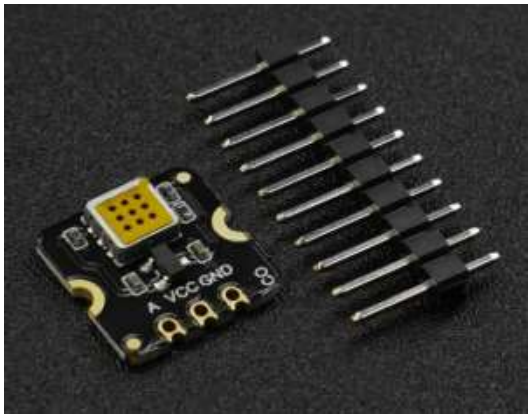
- Lý do lựa chọn: DHT22 mang lại độ chính xác tương đối tốt và độ ổn định cao hơn so với phiên bản DHT11, hoàn toàn phù hợp cho việc giám sát điều kiện môi trường trong bối cảnh đề tài. Cảm biến này có giá thành hợp lý, dễ dàng tìm kiếm và có nhiều thư viện hỗ trợ lập trình cho Arduino Uno, giúp quá trình tích hợp nhanh chóng và hiệu quả. Việc sử dụng tín hiệu đầu ra số cũng giúp giảm thiểu nhiễu và đơn giản hóa việc đọc dữ liệu.
- Kết nối với Arduino Uno: Cảm biến DHT22 được kết nối với Arduino Uno thông qua một chân GPIO duy nhất để truyền dữ liệu. Một điện trở kéo lên (pull-up resistor) 4.7kΩ thường được khuyến nghị giữa chân dữ liệu và VCC để đảm bảo tín hiệu ổn định. Nguồn cấp 5V được lấy từ Arduino Uno.



Hình 3.4 Sơ đồ đấu nối DHT22 với Arduino

3.3.3. Cảm biến khí CO DFRobot Fermion: MEMS Carbon Monoxide CO Gas Detection Sensor

- Giới thiệu và nguyên lý hoạt động: Để phát hiện nồng độ Carbon Monoxide (CO), hệ thống sử dụng cảm biến thuộc dòng DFRobot Fermion, tích hợp công nghệ MEMS (Micro-Electro-Mechanical Systems). Công nghệ MEMS giúp cảm biến đạt được kích thước cực kỳ nhỏ gọn, giảm thiểu đáng kể mức tiêu thụ năng lượng và mang lại khả năng phản hồi nhanh hơn đáng kể so với các loại cảm biến MOX bán dẫn oxit kim loại truyền thống. Cảm biến hoạt động dựa trên sự thay đổi điện trở của vật liệu cảm biến khi tiếp xúc với phân tử CO trong không khí. Một bộ phận gia nhiệt siêu nhỏ tích hợp sẵn trong cấu trúc MEMS sẽ duy trì nhiệt độ tối ưu cho quá trình phản ứng hóa học, đảm bảo độ nhạy và độ chọn lọc cao nhất. Tín hiệu đầu ra là điện áp analog, phản ánh sự thay đổi điện trở của cảm biến theo nồng độ CO, và được đưa vào bộ chuyển đổi ADC tích hợp trên Arduino để xử lý.

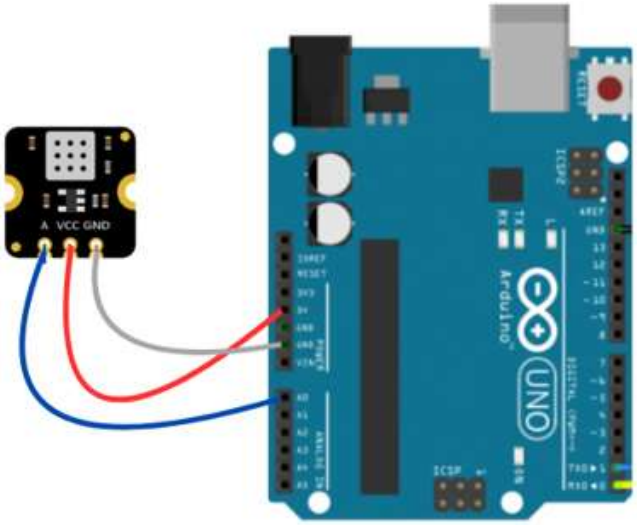


Hình 3.5 Cảm biến khí CO

➢ THÔNG SỐ KỸ THUẬT

- Mã sản phẩm (SKU): SEN0564
- Loại khí phát hiện: CO (Carbon Monoxide – khí CO)
- Dải phát hiện: 5–5000 ppm
- Điện áp hoạt động: 3.3–5V
- Dòng điện hoạt động: < 20 mA
- Tín hiệu đầu ra: Điện áp analog
- Độ nhạy: R_0 (trong không khí) / R_s (trong môi trường có 150 ppm CO) ≥ 3
- Nhiệt độ hoạt động: -10 đến 50°C
- Độ ẩm hoạt động: 15–90% RH (không ngưng tụ)
- Tuổi thọ: ≥ 5 năm (trong không khí)
- Kích thước: 13 × 13 × 2.5 mm / 0.51 × 0.51 × 0.1 inch

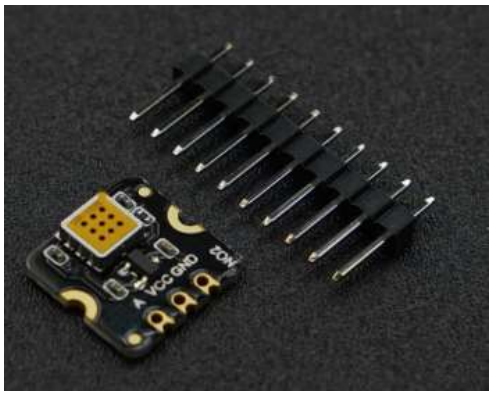
- Lý do lựa chọn: Việc lựa chọn cảm biến CO DFRobot Fermion dựa trên các ưu điểm vượt trội của công nghệ MEMS, mang lại hiệu suất cao, kích thước nhỏ gọn và tiêu thụ điện năng thấp, rất lý tưởng cho một thiết bị datalogger. Tín hiệu đầu ra dạng analog giúp dễ dàng tích hợp với các chân ADC trên Arduino Uno, không yêu cầu giao tiếp phức tạp như I2C hoặc SPI. Quan trọng hơn, độ nhạy và độ chọn lọc cao của cảm biến này là yếu tố then chốt để đo lường nồng độ CO với độ chính xác cần thiết trong môi trường công nghiệp phức tạp
- Kết nối với Arduino Uno: Cảm biến khí CO được cấp nguồn từ chân 5V của Arduino Uno. Tín hiệu đầu ra analog được kết nối vào chân A0, cho phép Arduino đọc giá trị điện áp tương ứng với nồng độ CO và xử lý qua bộ ADC tích hợp.



Hình 3.6 Sơ đồ đấu nối cảm biến CO với Arduino

3.3.4. Cảm biến khí NO2 DFRobot Fermion: MEMS Nitrogen Dioxide NO2 Gas Detection Sensor

- Giới thiệu và nguyên lý hoạt động: Tương tự như cảm biến CO, cảm biến khí NO2 này cũng là một thành viên của dòng DFRobot Fermion, tận dụng công nghệ MEMS để phát hiện nồng độ Nitrogen Dioxide (NO2). NO2 là một trong những chất gây ô nhiễm không khí phổ biến, phát sinh chủ yếu từ quá trình đốt cháy nhiên liệu hóa thạch và các hoạt động công nghiệp. Cảm biến hoạt động dựa trên nguyên lý tương tự như cảm biến MEMS CO, với một vật liệu cảm biến được tối ưu hóa đặc biệt để phản ứng với NO2. Khi NO_2 hấp phụ lên bề mặt cảm biến, điện trở thay đổi và được chuyển đổi thành tín hiệu điện áp analog, sau đó đưa vào Arduino Uno để xử lý thông qua bộ ADC tích hợp.



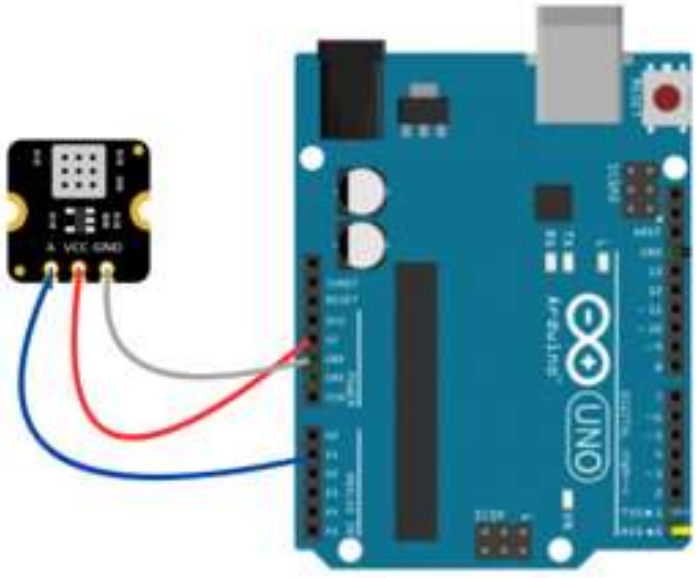
➢ THÔNG SỐ KỸ THUẬT

- Mã sản phẩm (SKU): SEN0574
- Loại khí phát hiện: NO_2 (Nitơ Dioxit)
- Dải phát hiện: 0.1–10 ppm
- Điện áp hoạt động: 3.3–5V
- Dòng điện hoạt động: < 20 mA
- Tín hiệu đầu ra: Điện áp analog
- Độ nhạy: R_0 (trong không khí) / R_s (trong môi trường có 5 ppm NO_2) ≥ 0.5
- Nhiệt độ hoạt động: -10 đến 50°C

Hình 3.7 Cảm biến khí NO2

- Độ ẩm hoạt động: 15–90% RH (không ngưng tụ)
- Tuổi thọ: ≥ 5 năm (trong không khí)
- Kích thước: 13 × 13 × 2.5 mm / 0.51 × 0.51 × 0.1 inch

- Lý do lựa chọn: Lựa chọn cảm biến NO2 DFRobot Fermion được thúc đẩy bởi ưu điểm của công nghệ MEMS về hiệu suất, kích thước và hiệu quả năng lượng. Cảm biến xuất tín hiệu analog nên dễ dàng tích hợp vào các chân analog A1 của Arduino Uno mà không cần giao tiếp phức tạp. Đặc biệt, độ nhạy và độ chọn lọc cao của cảm biến này rất quan trọng trong việc đo lường chính xác nồng độ NO_2 trong môi trường công nghiệp để phục vụ đánh giá chất lượng không khí.
- Kết nối với Arduino Uno: Cảm biến NO_2 được cấp nguồn từ chân 5V của Arduino Uno. Tín hiệu đầu ra analog được đưa vào chân A1, Arduino sẽ đọc tín hiệu và xử lý thông qua ADC nội bộ.



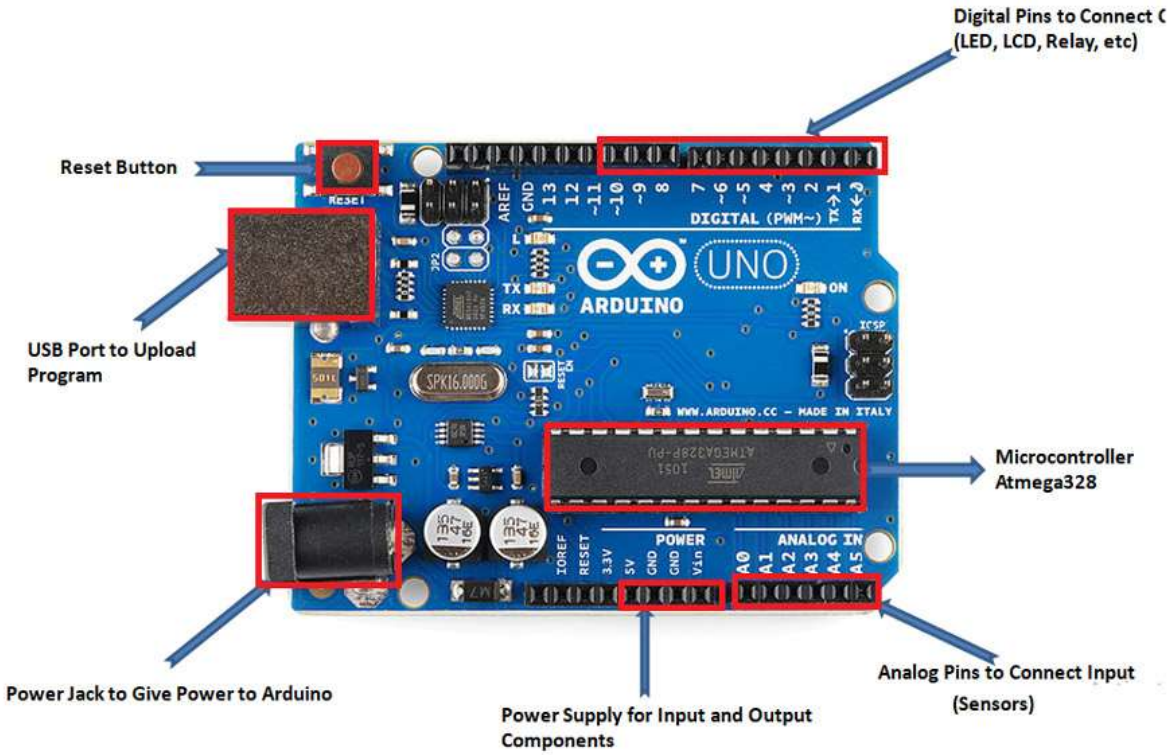
Hình 3.8 Sơ đồ đấu nối cảm biến NO2 với Arduino

3.4. Bộ điều khiển trung gian

Trong hệ thống datalogger này, chúng tôi sử dụng Arduino Uno như một bộ điều khiển phụ để đọc dữ liệu từ các cảm biến. Việc dùng Arduino Uno có vai trò quan trọng trong việc thu thập và chuẩn bị dữ liệu trước khi gửi về Raspberry Pi.

3.4.1. Tổng quan về Arduino

Arduino Uno là một bo mạch vi điều khiển phổ biến, dùng chip ATmega328P. Nó được chọn vì dễ sử dụng, có nhiều tài liệu hướng dẫn và thư viện hỗ trợ. Điều này giúp chúng tôi dễ dàng lập trình để giao tiếp với các loại cảm biến khác nhau.



Hình 3.9 Arduino Uno

3.4.2. Vai trò của Arduino Uno trong hệ thống

Arduino Uno chịu trách nhiệm chính trong việc thu thập dữ liệu từ các cảm biến và xử lý sơ bộ. Cụ thể:

- Đọc dữ liệu từ cảm biến: Arduino Uno kết nối trực tiếp với tất cả các cảm biến:
  - Cảm biến bụi PMS7003: Giao tiếp qua cổng Serial (UART) để đọc nồng độ PM2.5 và PM10.
  - Cảm biến nhiệt độ và độ ẩm DHT22: Giao tiếp qua giao thức 1-Wire để lấy giá trị nhiệt độ và độ ẩm.
  - Cảm biến khí CO và NO_2 (DFRobot Fermion): Cả hai đều xuất tín hiệu analog, được Arduino đọc qua các chân analog (A0, A1…).
- Xử lý tín hiệu analog**:** Đây là lý do chính khiến chúng tôi sử dụng Arduino Uno thay vì chỉ dùng Raspberry Pi. Raspberry Pi không có bộ ADC tích hợp, nên

không thể đọc tín hiệu analog trực tiếp từ các cảm biến khí. Arduino Uno giải quyết vấn đề này bằng cách chuyển đổi tín hiệu analog thành dữ liệu số.

- Tiền xử lý dữ liệu: Sau khi thu thập, Arduino Uno thực hiện các bước xử lý sơ bộ như:
  - Chuyển đổi đơn vị đo.
  - Lọc nhiễu, làm mượt dữ liệu (trung bình trượt).
  - Đóng gói dữ liệu từ tất cả các cảm biến thành chuỗi định dạng sẵn.
- Gửi dữ liệu đến Raspberry Pi: Gói dữ liệu sau xử lý sẽ được Arduino truyền về Raspberry Pi thông qua giao tiếp Serial (UART), giúp Raspberry Pi tập trung vào việc lưu trữ, hiển thị và truyền dữ liệu lên server.

3.4.3. Các kết nối chính trên Arduino

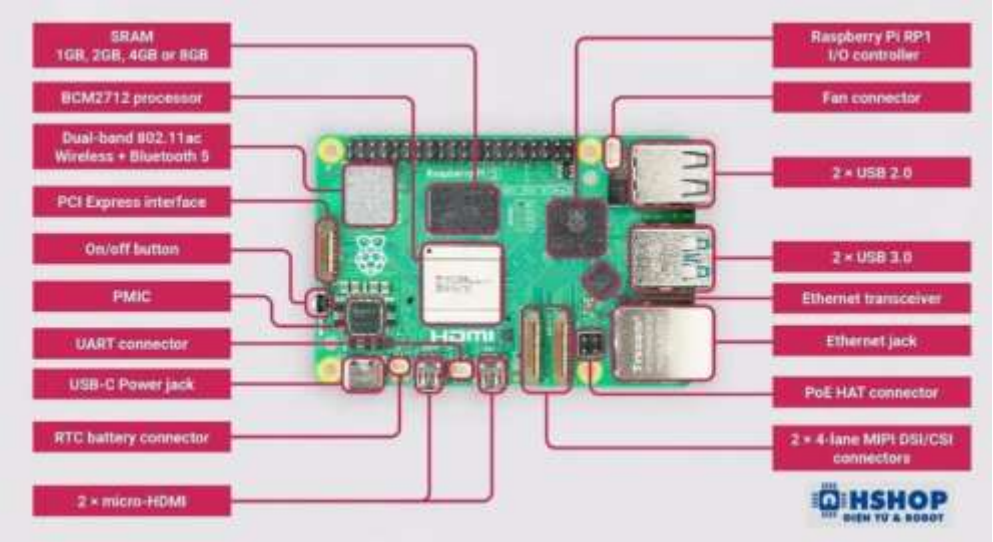
- UART (Serial): Dùng để nhận dữ liệu từ cảm biến PMS7003 và gửi dữ liệu đã xử lý đến Raspberry Pi.
- 1-Wire (GPIO Digital): Dùng để giao tiếp với cảm biến DHT22 qua một chân digital (ví dụ D2).
- ADC (Analog Input): Các chân A0, A1 được sử dụng để đọc tín hiệu analog từ cảm biến khí CO và NO_2.
- GND/5V: Cấp nguồn chung cho các cảm biến từ Arduino

3.5. Bộ điều khiển chính

Trong kiến trúc phân tán của hệ thống datalogger, sau khi Arduino Uno hoàn thành nhiệm vụ thu thập và tiền xử lý dữ liệu cảm biến, Raspberry Pi 5 sẽ tiếp nhận vai trò của bộ điều khiển chính – đóng vai trò là "bộ não" trung tâm và cổng kết nối toàn cầu của toàn hệ thống. Với hiệu năng vượt trội và khả năng kết nối đa dạng, Raspberry Pi 5 là chìa khóa để hiện thực hóa các tác vụ xử lý dữ liệu phức tạp, lưu trữ quy mô lớn và truyền thông tin liên tục tới máy chủ đám mây.

3.5.1. Tổng quan về Raspberry pi 5

Raspberry Pi 5 là dòng máy tính bo mạch đơn (Single-Board Computer - SBC) mới nhất và mạnh mẽ nhất của Raspberry Pi Foundation tính đến thời điểm hiện tại. Được trang bị bộ xử lý Broadcom BCM2712 Quad-core ARM Cortex-A76 (ARMv8) hoạt động ở xung nhịp 2.4GHz, cùng với RAM LPDDR4X dung lượng [4GB/8GB], Raspberry Pi 5 mang lại hiệu năng xử lý cao hơn đáng kể so với các thế hệ trước. Nó chạy hệ điều hành Raspberry Pi OS (dựa trên Debian Linux), cung cấp một môi trường linh hoạt và mạnh mẽ cho việc phát triển phần mềm phức tạp.



Hình 3.110 Raspberry Pi 5

3.5.2. Vai trò và chức năng chính của Raspberry Pi 5 trong hệ thống

Raspberry Pi 5 đảm nhiệm các nhiệm vụ quan trọng, tận dụng tối đa sức mạnh phần cứng của mình:

- Thu thập dữ liệu từ Arduino Uno: Raspberry Pi 5 tiếp nhận các gói dữ liệu đã được tiền xử lý và đóng gói từ Arduino Uno thông qua giao tiếp Serial (UART). Việc này đảm bảo dữ liệu đến Pi đã ở định dạng chuẩn, giảm tải cho Pi trong việc xử lý tín hiệu cấp thấp từ cảm biến.
- Xử lý dữ liệu nâng cao: Với sức mạnh tính toán vượt trội, Raspberry Pi 5 có khả năng thực hiện các thuật toán xử lý dữ liệu phức tạp hơn như:
    - Hiệu chuẩn dữ liệu: Áp dụng các phương pháp hiệu chuẩn nâng cao để cải thiện độ chính xác của dữ liệu cảm biến.
    - Tính toán chỉ số chất lượng không khí (AQI): Chuyển đổi các nồng độ khí và bụi mịn thành chỉ số AQI dễ hiểu, cung cấp cái nhìn tổng quan về chất lượng không khí.

    - Phân tích xu hướng: Chạy các script Python phân tích dữ liệu để phát hiện các xu hướng ô nhiễm theo thời gian.
- Lưu trữ dữ liệu cục bộ: Dữ liệu quan trắc sau khi được xử lý sẽ được lưu trữ cục bộ trên thẻ nhớ MicroSD của Raspberry Pi 5. Dung lượng lưu trữ lớn của thẻ nhớ cho phép ghi lại lịch sử dữ liệu dài hạn, đảm bảo không mất mát thông tin ngay cả khi mất kết nối mạng. Dữ liệu thường được lưu dưới dạng tệp văn bản (ví dụ: CSV, JSON) để dễ dàng truy xuất và phân tích sau này.
- Truyền dữ liệu lên máy chủ đám mây: Đây là một trong những chức năng cốt lõi của datalogger. Raspberry Pi 5 sử dụng khả năng kết nối mạng tích hợp (Wi-Fi 5GHz và Gigabit Ethernet) để truyền dữ liệu đã được xử lý và lưu trữ cục bộ lên máy chủ đám mây hoặc webserver. Trong giai đoạn thử nghiệm, chúng tôi sử dụng nền tảng Localtonet để tạo một đường hầm (tunnel) an toàn, cho phép truy cập dữ liệu từ xa mà không cần cấu hình mạng phức tạp. Điều này giúp dễ dàng kiểm tra và đánh giá hệ thống trước khi triển khai chính thức.
- Quản lý và điều phối hệ thống: Raspberry Pi 5 đóng vai trò quản lý toàn bộ hệ thống, điều phối hoạt động giữa các module, quản lý thời gian thực, và có thể hỗ trợ các tính năng bổ sung như giao diện người dùng cục bộ (nếu có) hoặc các tác vụ lập lịch.

3.5.3. Các kết nối chính của Raspberry pi 5

Raspberry Pi 5 cung cấp một loạt các giao tiếp vật lý để kết nối với Arduino Uno và mạng lưới:

- UART (Serial): Giao tiếp chính giữa Raspberry Pi và Arduino Uno. Sử dụng chân GPIO 14 (TX) và GPIO 15 (RX), cấu hình baudrate đồng bộ với Arduino (thường là 9600 hoặc 115200).
- Ethernet (Gigabit): Kết nối mạng có dây ổn định, lý tưởng cho môi trường công nghiệp.
- Wi-Fi (Dual-band 802.11ac): Cho phép kết nối mạng không dây linh hoạt, phù hợp với khu vực không kéo dây.
- USB 3.0/2.0: Kết nối các thiết bị ngoại vi như chuột, bàn phím hoặc các thiết bị lưu trữ.
- Thẻ MicroSD: Dùng để cài hệ điều hành và lưu trữ dữ liệu từ datalogger.

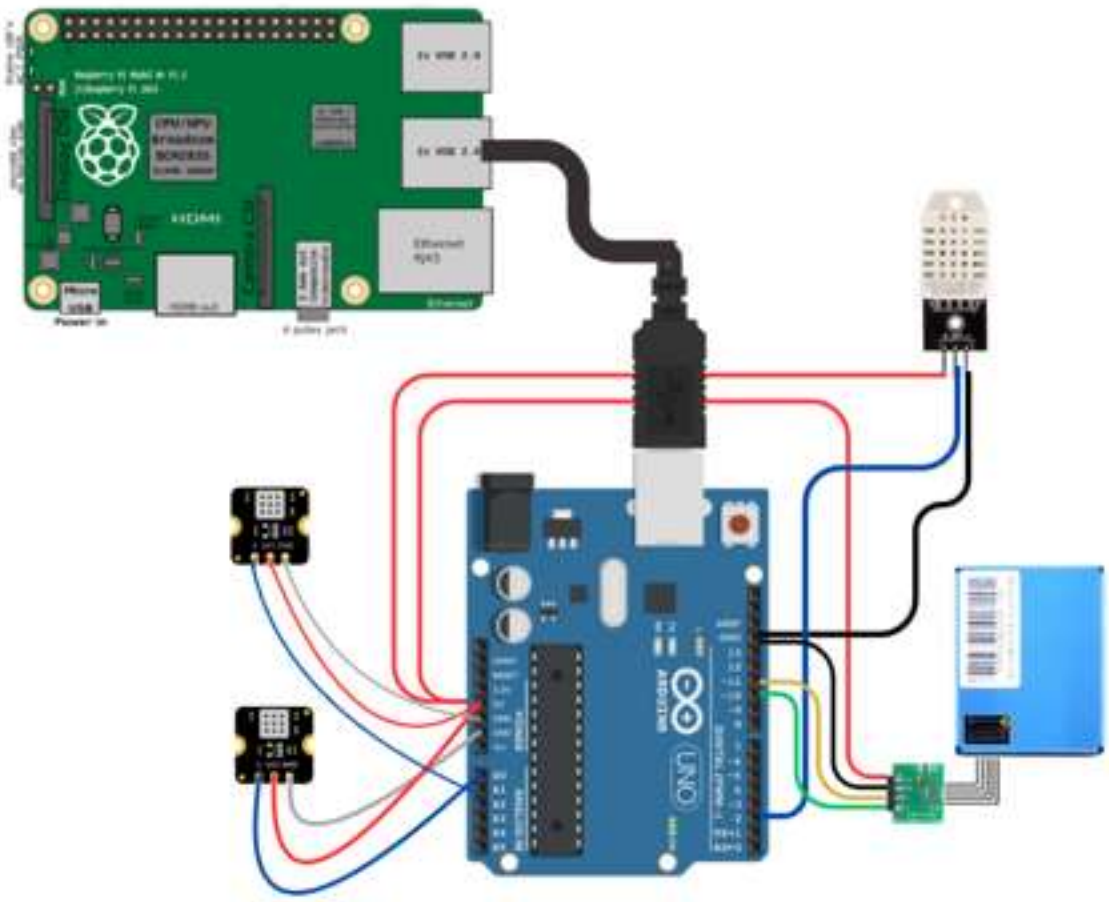
Danh sách kết nối:

// - DHT22: VCC - 5V, GND - GND, DATA – D2 (arduino)

// - PMS7003: VCC - 5V, GND - GND, TX – D10, RX – D11 (arduino)

// - CO MEMS: VCC - 5V, GND - GND, OUT - A0 (arduino)

// - NO2 MEMS: VCC - 5V, GND - GND, OUT - A1 (arduino)



Hình 3.11 Kết nối phần cứng

3.6. Mạch nguồn và các thành phần phụ trợ

Để đảm bảo hoạt động liên tục, ổn định và đáng tin cậy cho toàn bộ hệ thống datalogger trong môi trường công nghiệp, việc thiết kế một mạch nguồn hiệu quả và tích hợp các

thành phần phụ trợ cần thiết là vô cùng quan trọng. Phần này sẽ trình bày chi tiết về các giải pháp cung cấp năng lượng và các module hỗ trợ, giúp hệ thống hoạt động trơn tru.

3.6.1. Mạch nguồn cung cấp năng lượng

Hệ thống datalogger của chúng tôi yêu cầu các mức điện áp khác nhau cho các thành phần: Raspberry Pi 5 cần nguồn 5V DC với dòng điện cao, Arduino Uno hoạt động ở 5V DC, và các cảm biến cũng yêu cầu 5V DC (PMS7003, DHT22) hoặc có thể hoạt động ở 3.3V/5V (DFRobot Fermion). Để đảm bảo cung cấp năng lượng ổn định và hiệu quả, mạch nguồn được thiết kế với các nguyên tắc sau:

- Nguồn vào: Hệ thống được cấp nguồn thông qua bộ chuyển đổi AC-DC (adapter) với điện áp đầu ra 12V DC và dòng điện đủ lớn (ví dụ: 2A hoặc hơn tùy tổng công suất tiêu thụ thực tế của Pi 5 và các thiết bị khác). Việc sử dụng 12V DC làm nguồn vào trung gian giúp linh hoạt trong việc cấp nguồn cho các bộ chuyển đổi hạ áp.
- Bộ chuyển đổi hạ áp Step-down (Buck Converter):
    - Chuyển đổi 12V xuống 5V cho Raspberry Pi 5 và Arduino Uno: Sử dụng module chuyển đổi DC-DC loại buck (ví dụ: LM2596-based module hoặc module chuyên dụng cho Pi 5 với khả năng cấp dòng cao). Module này sẽ hạ điện áp 12V từ nguồn adapter xuống 5V để cấp cho Raspberry Pi 5 và Arduino Uno. Việc này đặc biệt quan trọng vì Raspberry Pi 5 yêu cầu nguồn điện sạch và ổn định, có khả năng cung cấp dòng điện lớn (có thể lên đến 3A hoặc hơn khi hoạt động tải nặng).
        - Lý do lựa chọn: Module buck có hiệu suất cao hơn đáng kể so với ổn áp tuyến tính (linear regulator) khi có sự chênh lệch lớn giữa điện áp vào và ra, giúp giảm tổn thất năng lượng và tản nhiệt.
        - Lưu ý thiết kế: Cần chọn module có khả năng cấp dòng đủ lớn, có các tụ lọc đầu vào/đầu ra để làm mượt điện áp và giảm nhiễu.
- Ổn áp tích hợp trên Arduino Uno: Arduino Uno tự có ổn áp tích hợp để cung cấp 5V và 3.3V cho các linh kiện trên bo mạch và các cảm biến nhỏ. Các cảm biến PMS7003 và DHT22 sẽ được cấp nguồn trực tiếp từ các chân 5V của Arduino Uno.
- Nguồn cấp cho cảm biến DFRobot Fermion: Các cảm biến DFRobot Fermion (CO, NO_2) có thể hoạt động ở cả 3.3V và 5V. Để đơn giản hóa và đồng bộ với Arduino Uno (với mức logic 5V), chúng sẽ được cấp nguồn 5V từ Arduino.

3.6.2. Module lưu trữ dữ liệu cục bộ

- Thẻ nhớ MicroSD: Raspberry Pi 5 sử dụng thẻ nhớ MicroSD làm nơi lưu trữ hệ điều hành (Raspberry Pi OS) và đồng thời là bộ nhớ chính để lưu trữ dữ liệu quan trắc.
  - Dung lượng: Thẻ nhớ MicroSD với dung lượng [ví dụ: 16GB hoặc 32GB] được lựa chọn để đảm bảo đủ không gian lưu trữ cho hệ điều hành, các ứng dụng và lịch sử dữ liệu quan trắc trong thời gian dài (ví dụ: nhiều tháng hoặc năm).
  - Tốc độ đọc/ghi: Việc lựa chọn thẻ nhớ Class 10 hoặc UHS (Ultra High Speed) là cần thiết để đảm bảo tốc độ đọc/ghi dữ liệu nhanh, giúp hệ thống hoạt động mượt mà và không bị nghẽn cổ chai khi ghi dữ liệu liên tục.
- Định dạng lưu trữ: Dữ liệu từ các cảm biến sẽ được Raspberry Pi 5 tổ chức và ghi vào các tệp tin dưới định dạng CSV (Comma Separated Values) hoặc JSON (JavaScript Object Notation). Các định dạng này dễ đọc, dễ phân tích và tương thích với nhiều công cụ xử lý dữ liệu khác. Dữ liệu được lưu trữ kèm theo dấu thời gian (timestamp) để phục vụ việc phân tích lịch sử.

3.6.3. Module truyền thông mạng

Khả năng kết nối mạng là cốt lõi để datalogger có thể gửi dữ liệu lên máy chủ và cho phép truy cập từ xa. Raspberry Pi 5 tích hợp sẵn các module truyền thông mạnh mẽ:

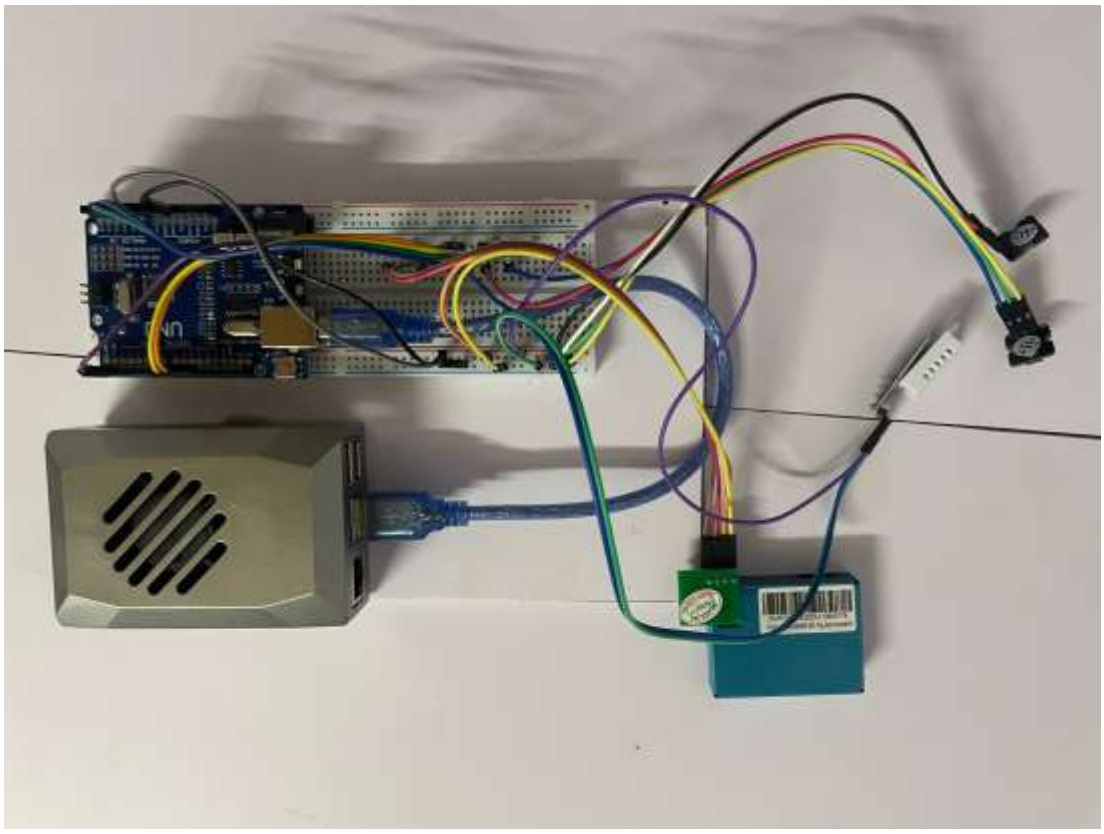
- Wi-Fi (Dual-band 802.11ac): Cung cấp khả năng kết nối không dây tốc độ cao và linh hoạt. Điều này đặc biệt hữu ích cho việc triển khai ở những vị trí khó kéo dây mạng, cho phép datalogger gửi dữ liệu qua mạng Wi-Fi hiện có trong khu công nghiệp hoặc từ một hotspot di động.
- Ethernet (Gigabit): Cổng Ethernet tích hợp cung cấp kết nối mạng có dây ổn định và tốc độ cao (1 Gbps), là lựa chọn ưu tiên cho môi trường công nghiệp yêu cầu độ tin cậy cao và khả năng chống nhiễu tốt.

3.6.4. Các thành phần phụ trợ khác

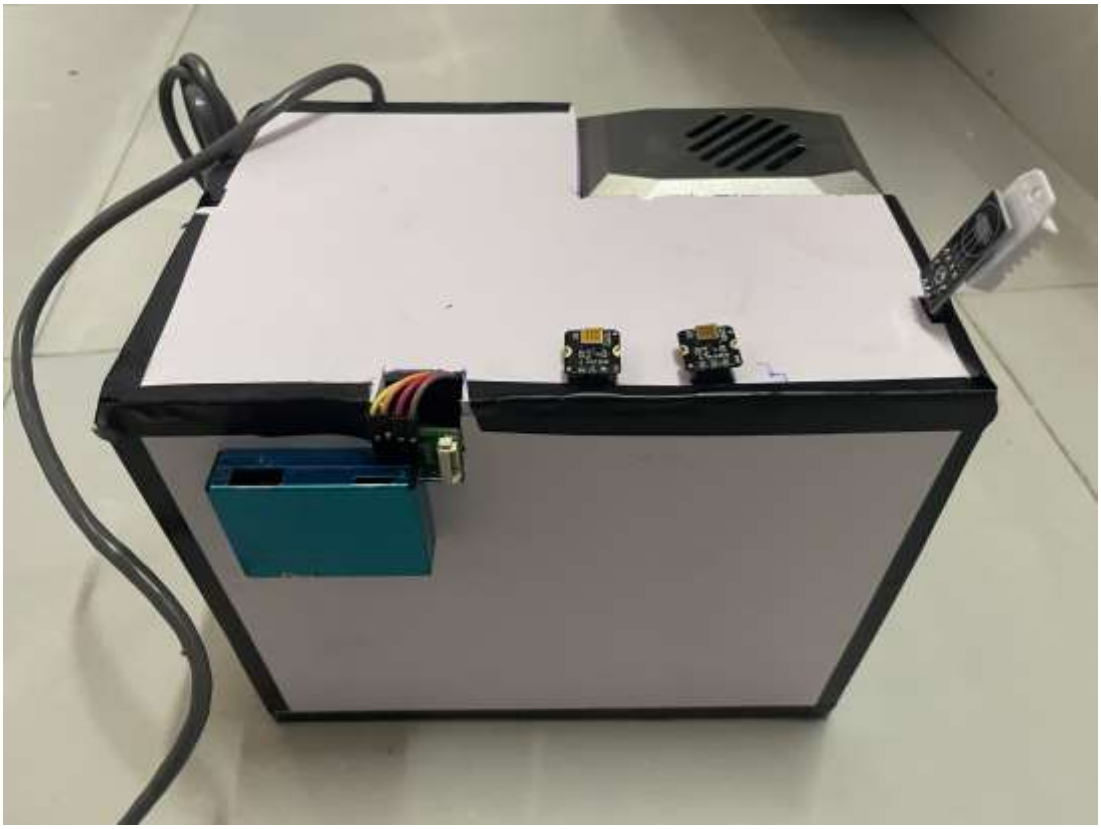
- LED báo trạng thái: Các đèn LED có thể được sử dụng để báo hiệu trạng thái hoạt động của hệ thống (ví dụ: nguồn, kết nối mạng, lỗi cảm biến), giúp dễ dàng theo dõi trực quan mà không cần màn hình.

- Nút nhấn Reset/Chế độ: Một nút nhấn có thể được tích hợp để reset hệ thống hoặc chuyển đổi giữa các chế độ hoạt động (ví dụ: chế độ hiệu chuẩn, chế độ ngủ).
- Vỏ bảo vệ: Để bảo vệ các thành phần điện tử khỏi bụi bẩn, độ ẩm và các yếu tố môi trường khắc nghiệt của khu công nghiệp, một vỏ bảo vệ đạt chuẩn công nghiệp (ví dụ: IP65) sẽ được sử dụng. Vỏ phải có khả năng tản nhiệt tốt, có kèm quạt nhỏ và các lỗ thông hơi để cảm biến khí có thể lấy mẫu không khí.

3.7. Lắp đặt phần cứng thực tế



Hình 3.12 Đấu nối phần cứng thực tế



Hình 3.13 Mô hình phần cứng thực tế

CHƯƠNG 4 : THIẾT KẾ PHẦN MỀM

4.1. Tổng quan về kiến trúc phần mềm

Phần mềm trong hệ thống quan trắc chất lượng không khí của chúng tôi được thiết kế để đảm bảo dữ liệu được thu thập, xử lý, lưu trữ và hiển thị một cách liên tục và hiệu quả. Mục tiêu là xây dựng một phần mềm ổn định, tận dụng khả năng của Arduino Uno và Raspberry Pi để cung cấp thông tin chất lượng không khí.

Cấu trúc phần mềm của hệ thống được chia thành các lớp (giai đoạn) chính, mỗi lớp có nhiệm vụ riêng biệt. Dữ liệu sẽ đi theo một cụ thể như sau:

4.1.1. Lớp thu thập dữ liệu

- Đây là lớp đầu tiên, nơi các cảm biến (PMS7003, DHT22, DFRobot Fermion CO, DFRobot Fermion NO_2) tiếp xúc trực tiếp với không khí.
- Arduino Uno có nhiệm vụ chính là đọc dữ liệu từ cảm biến. Chương trình trên Arduino liên tục đọc tín hiệu từ các cảm biến qua các kiểu kết nối khác nhau (UART, I2C, 1-Wire). Đặc biệt, Arduino Uno sẽ chuyển đổi tín hiệu analog từ các cảm biến khí CO và NO_2 sang dạng số.
- Sau khi đọc, Arduino sẽ xử lý dữ liệu bước đầu như chuyển đổi đơn vị, làm mượt dữ liệu để giảm nhiễu, và đóng gói chúng thành một định dạng dễ gửi đi (ví dụ: một chuỗi văn bản như CSV). Việc này giúp giảm bớt công việc cho bộ xử lý chính sau này.

4.1.2. Lớp xử lý và lưu trữ dữ liệu cục bộ

- Raspberry Pi 5 là phần "điều khiển trung tâm" của phần mềm. Nó sẽ nhận dữ liệu từ Arduino Uno và thực hiện các bước xử lý tiếp theo.
- Chương trình trên Raspberry Pi 5 (viết bằng ngôn ngữ Python) có các nhiệm vụ sau:
  - Nhận dữ liệu: Đọc dữ liệu gửi từ Arduino Uno qua cổng Serial (UART).
  - Xử lý dữ liệu: Xử lý thêm dữ liệu đã nhận, bao gồm việc tính toán chỉ số chất lượng không khí VN_AQI dựa trên các thông số đã thu thập.
  - Lưu trữ dữ liệu: Dữ liệu sau khi được xử lý sẽ được lưu vào cơ sở dữ liệu MySQL chạy trên Raspberry Pi 5. Dữ liệu được lưu trên thẻ nhớ MicroSD để đảm bảo không bị mất ngay cả khi mất mạng. Bảng trong MySQL sẽ có các cột: id, ngày giờ, VN_AQI, Nhiệt độ, độ ẩm, CO, NO2, PM10, PM25.

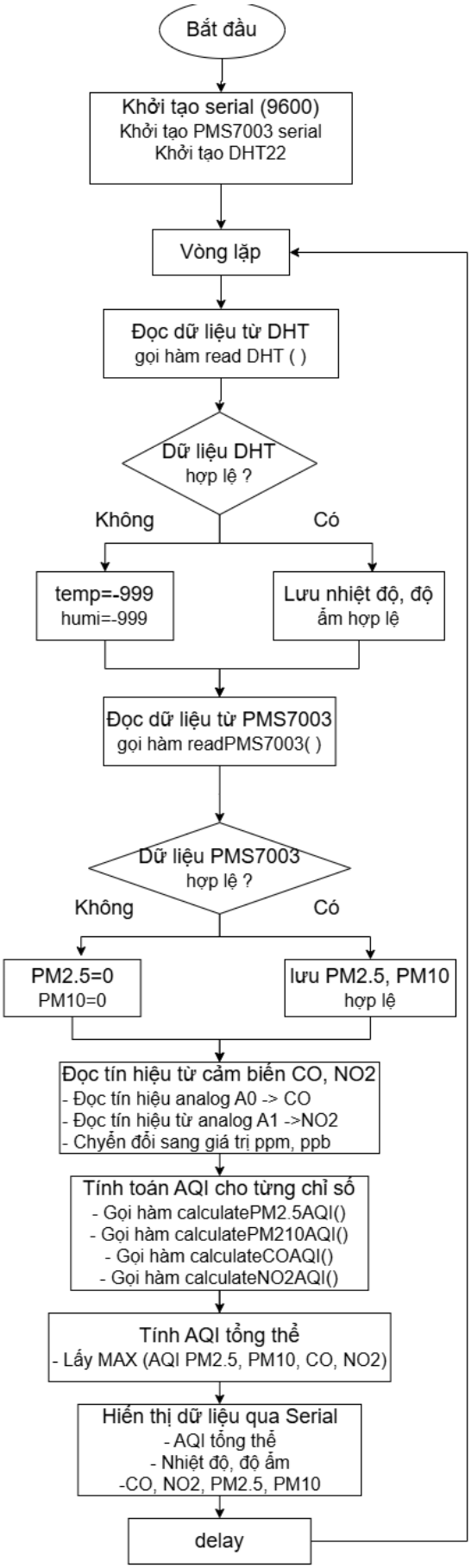
- Cung cấp dữ liệu (API Web): Raspberry Pi 5 chạy một ứng dụng web server nhỏ dùng framework Flask (Python). Ứng dụng này tạo ra các giao diện lập trình (API) để các ứng dụng khác (như trang web hiển thị) có thể yêu cầu và lấy dữ liệu từ MySQL.

4.1.3. Lớp truyền dữ liệu và hiển thị

- Truyền dữ liệu lên mạng: Để đưa dữ liệu quan trắc ra ngoài mạng nội bộ, Raspberry Pi 5 dùng kết nối Internet (Wi-Fi hoặc Ethernet) và dịch vụ Localtonet. Localtonet giúp tạo một đường dẫn an toàn để truy cập từ xa vào Flask Web API trên Pi mà không cần cấu hình mạng phức tạp. Điều này tiện lợi cho việc kiểm tra và đánh giá hệ thống.
- Giao diện Người dùng Web: Dữ liệu từ Flask API sẽ được hiển thị cho người dùng qua một trang web được xây dựng bằng HTML, CSS và JavaScript. Trang web này sẽ hiển thị:
  - Thông số tổng quát: Chỉ số VN_AQI tổng thể kèm theo màu sắc cảnh báo (Tốt, Trung bình, Kém, Xấu, Rất xấu, Nguy hại) và tóm tắt nhanh về mức độ ô nhiễm.
  - Thông số chi tiết: Nồng độ cụ thể của PM2.5, PM10, CO, NO2, cùng với nhiệt độ và độ ẩm.
  - Biểu đồ và Lịch sử: Dữ liệu được hiển thị theo thời gian thực dưới dạng biểu đồ đường (theo ngày, giờ, phút). Người dùng có thể xem lại dữ liệu cũ, so sánh mức độ ô nhiễm và theo dõi xu hướng.
  - Tải dữ liệu: Cung cấp chức năng cho phép người dùng tải về dữ liệu đã thu thập để phân tích thêm.

4.2. Lập trình trên Arduino Uno

Phần mềm trên Arduino Uno có nhiệm vụ chính là thu thập dữ liệu thô từ các cảm biến, xử lý sơ bộ và gửi dữ liệu đã chuẩn bị này đến Raspberry Pi. Chương trình được viết bằng ngôn ngữ lập trình Arduino (dựa trên C/C++) trong môi trường Arduino IDE.


Bắt đầu
Khởi tạo serial (9600)
Khởi tạo PMS7003 serial
Khởi tạo DHT22
Vòng lặp
Đọc dữ liệu từ DHT
gọi hàm read DHT ( )
Dữ liệu DHT
hợp lệ ?
Không
Có
temp=-999
humi=-999
Lưu nhiệt độ, độ
ẩm hợp lệ
Đọc dữ liệu từ PMS7003
gọi hàm readPMS7003( )
Dữ liệu PMS7003
hợp lệ ?
Không
Có
PM2.5=0
PM10=0
lưu PM2.5, PM10
hợp lệ
Đọc tín hiệu từ cảm biến CO, NO2
- Đọc tín hiệu analog A0 -> CO
- Đọc tín hiệu từ analog A1 ->NO2
- Chyển đổi sang giá trị ppm, ppb
Tính toán AQI cho từng chỉ số
- Gọi hàm calculatePM2.5AQI()
- Gọi hàm calculatePM210AQI()
- Gọi hàm calculateCOAQI()
- Gọi hàm calculateNO2AQI()
Tính AQI tổng thể
- Lấy MAX (AQI PM2.5, PM10, CO, NO2)
Hiển thị dữ liệu qua Serial
- AQI tổng thể
- Nhiệt độ, độ ẩm
-CO, NO2, PM2.5, PM10
delay

Hình 4.1 Lưu đồ thuật toán đọc và tính toán dữ liệu trên Arduino

4.2.1. Đọc và xử lí dữ liệu cảm biến

Arduino Uno sẽ tuần tự đọc dữ liệu từ tất cả các cảm biến đã kết nối. Mỗi cảm biến có cách giao tiếp và đọc dữ liệu riêng.

a. Đọc dữ liệu từ cảm biến bụi PMS7003:

Cảm biến PMS7003 giao tiếp qua Serial (UART). Chúng tôi sử dụng thư viện SoftwareSerial để tạo một cổng Serial ảo trên các chân GPIO của Arduino, tránh xung đột với cổng Serial vật lý dùng để giao tiếp với máy tính (hoặc Raspberry Pi).

- Logic đọc :

1) Khởi tạo cổng SoftwareSerial cho PMS7003.
2) Gửi lệnh yêu cầu đọc dữ liệu đến cảm biến (nếu cần, tùy chế độ hoạt động của PMS7003).
3) Đọc gói dữ liệu 32 byte mà PMS7003 gửi về.
4) Phân tích cú pháp (parse) gói dữ liệu để trích xuất các giá trị PM2.5 và PM10. Các giá trị này thường ở dạng số nguyên, cần chia cho một hệ số nhất định để có đơn vị $\mu g/m^3$.

b. Đọc dữ liệu từ cảm biến nhiệt độ và độ ẩm DHT22:

Cảm biến DHT22 giao tiếp qua giao thức 1-Wire. Chúng tôi sử dụng thư viện DHT.h để đơn giản hóa việc đọc.

- Logic đọc:

1) Khởi tạo cảm biến DHT22 với chân dữ liệu đã kết nối.
2) Sử dụng hàm readHumidity ( ) và readTemperature ( ) của thư viện để lấy giá trị.
3) Kiểm tra lỗi đọc (ví dụ: cảm biến không phản hồi).

c. Đọc dữ liệu từ cảm biến khí DFRobot Fermion (CO, NO_2) qua ADC:

Hai cảm biến CO và NO_2 của DFRobot Fermion trong hệ thống này sử dụng ngõ ra tín hiệu analog. Chúng tôi kết nối đầu ra của từng cảm biến vào các chân analog của Arduino Uno (ví dụ A0 cho CO và A1 cho NO_2), sau đó sử dụng hàm analogRead() để đọc giá trị.

- Logic đọc :

1) Kết nối đầu ra tín hiệu của từng cảm biến vào chân analog tương ứng của Arduino.

2) Sử dụng hàm analogRead() để đọc giá trị ADC (từ 0 đến 1023 ứng với điện áp 0–5V).
3) Chuyển đổi giá trị ADC thành điện áp thực tế
4) Dựa vào công thức hoặc biểu đồ đặc trưng trong datasheet của từng cảm biến để tính toán nồng độ khí CO và NO_2 (ppm hoặc ppb). Mỗi loại cảm biến sẽ có hệ số quy đổi riêng.

4.2.2. Tính toán chỉ số AQI

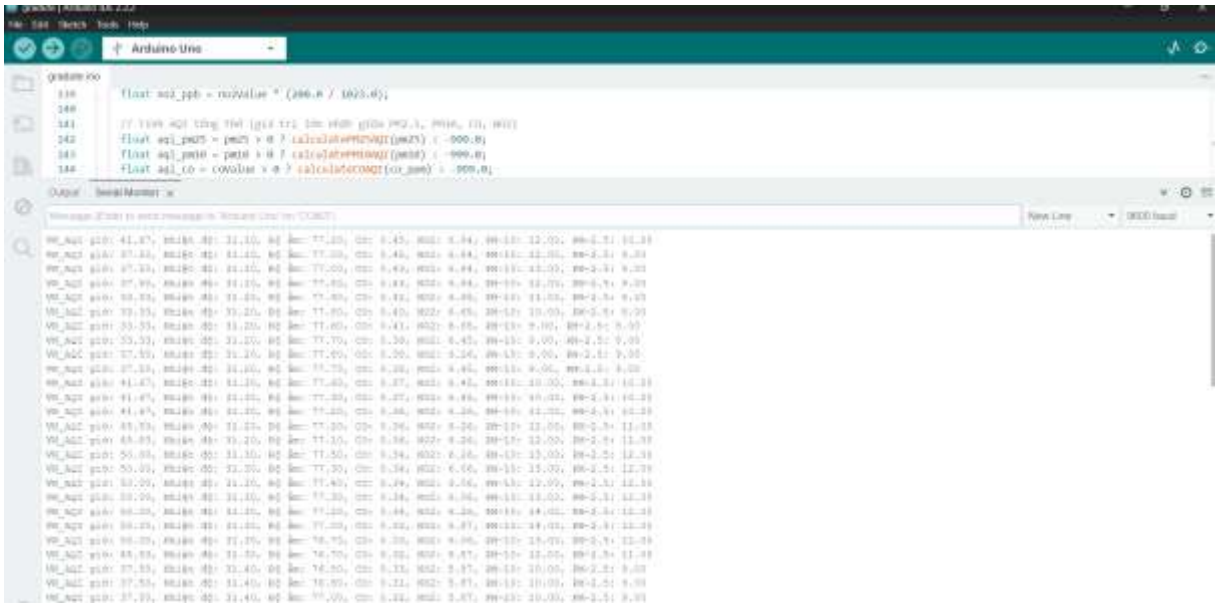
Với mỗi thông số (PM2.5, PM10, CO, NO2), so sánh nồng độ đo được với các ngưỡng trong bảng AQI (Bảng 2.1) để tìm ra chỉ số riêng cho từng thông số
Tính toán chỉ số AQI riêng cho từng thông số với công thức (1)
Chỉ số AQI tổng sẽ là giá trị AQI cao nhất trong các chỉ số AQI riêng lẻ.

4.2.3. Giao tiếp với Raspberry Pi

Sau khi đọc và tiền xử lý tất cả dữ liệu từ cảm biến và tính AQI, Arduino Uno sẽ đóng gói các giá trị này và gửi đến Raspberry Pi thông qua giao tiếp Serial (UART).

- Định dạng dữ liệu: Dữ liệu sẽ được đóng gói dưới dạng một chuỗi ký tự có cấu trúc rõ ràng để Raspberry Pi dễ dàng phân tích. Định dạng phổ biến là CSV (Comma Separated Values), CSV đơn giản và nhẹ hơn cho Arduino.
    - Ví dụ CSV : PM2.5, PM10, CO, NO2, Nhiệt độ, Độ Ẩm, AQI
    - Dữ liệu thực tế : 23.5, 45.2, 5.1, 0.08, 28.7, 65.4, 75
- Logic gửi dữ liệu :

1) Tạo chuỗi dữ liệu từ các biến đã đọc được.
2) Sử dụng Serial.print ( ) để gửi chuỗi qua cổng Serial vật lý của Arduino (chân TX/RX, được nối với chân RX/TX của Raspberry Pi).
3) Gửi dữ liệu định kỳ .



Hình 4.2 Kết quả đo và tính toán trên Arduino

4.3. Lập trình trên Raspberry Pi 5

Raspberry Pi 5 đóng vai trò là "bộ não" của hệ thống datalogger, tiếp nhận dữ liệu đã được tiền xử lý từ Arduino Uno, thực hiện các tính toán phức tạp hơn, lưu trữ vào cơ sở dữ liệu và cung cấp giao diện để dữ liệu có thể được truy cập và hiển thị. Toàn bộ phần mềm trên Raspberry Pi 5 được phát triển chủ yếu bằng ngôn ngữ Python, tận dụng sự mạnh mẽ, linh hoạt và hệ sinh thái thư viện phong phú của nó.

4.3.1. Nhận dữ liệu từ Arduino Uno

- Tổng quan và Vai trò:
  - Phần này hoạt động như một cầu nối dữ liệu, được triển khai trên máy chủ Raspberry Pi, để nhận và xử lý dữ liệu môi trường từ vi điều khiển Arduino.
  - Nó đảm bảo việc thu thập dữ liệu liên tục và đáng tin cậy từ cổng Serial, chuyển đổi định dạng và sẵn sàng cho các bước xử lý tiếp theo như lưu trữ vào cơ sở dữ liệu, hiển thị trên giao diện người dùng hoặc gửi lên đám mây.
- Raspberry Pi 5 giao tiếp với Arduino Uno thông qua cổng Serial (UART). Dữ liệu được gửi từ Arduino dưới dạng một chuỗi ký tự đã được định dạng (CSV).
- Kết nối Serial:
  - Sử dụng thư viện pyserial để thiết lập kết nối nối tiếp với vi điều khiển Arduino.
  - Cấu hình cổng Serial /dev/ttyUSB0 (đây là cổng thường thấy trên Linux cho các thiết bị USB-to-Serial), tốc độ baud 9600 và timeout 1 giây để tránh bị treo khi không có dữ liệu.
- Đọc và Xử lý Dữ liệu:

- Đọc từng dòng dữ liệu từ cổng Serial.
- Giải mã dòng dữ liệu từ byte sang chuỗi UTF-8 và loại bỏ ký tự xuống dòng.
- Lọc dữ liệu không cần thiết: Bỏ qua các dòng thông báo khởi động từ Arduino hoặc các dòng không chứa định dạng dữ liệu mong muốn.
- Phân tích chuỗi dữ liệu: Tách chuỗi dữ liệu nhận được thành các cặp khóa-giá trị (ví dụ: "Nhiet do: 25.5") dựa trên ký tự phân cách ", " và ": ".
- Chuyển đổi giá trị thành kiểu số thực (float) để phục vụ việc tính toán hoặc lưu trữ.

➢ Xác thực Dữ liệu:

- Kiểm tra sự hiện diện của tất cả các khóa dữ liệu cần thiết (Nhiệt độ, Độ ẩm, PM2.5, PM10, CO, NO2, AQI Tổng thể) để đảm bảo dữ liệu đầy đủ và hợp lệ.
- Thông báo lỗi nếu dữ liệu bị thiếu hoặc không đúng định dạng.

➢ Hiển thị Dữ liệu (Console):

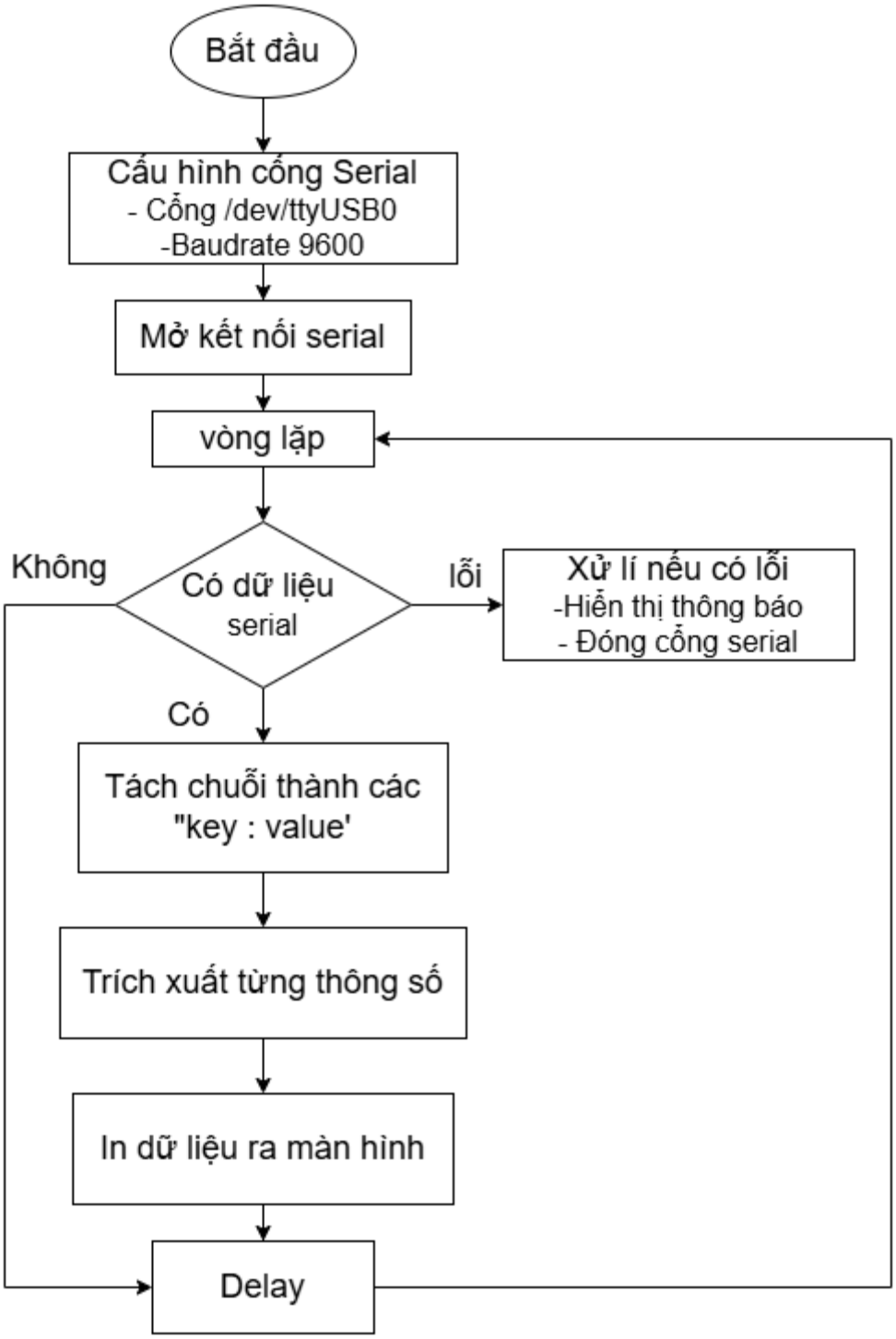
- In các giá trị dữ liệu đã được phân tích và xác thực ra màn hình console theo định dạng dễ đọc, hữu ích cho việc gỡ lỗi và kiểm tra hoạt động của hệ thống.

➢ Xử lý Lỗi:

- Sử dụng khối try-except để bắt và xử lý các ngoại lệ có thể xảy ra trong quá trình đọc và phân tích dữ liệu, như lỗi ValueError (khi chuyển đổi kiểu dữ liệu) hoặc các lỗi chung khác khi giao tiếp Serial.

➢ Luồng hoạt động chính:

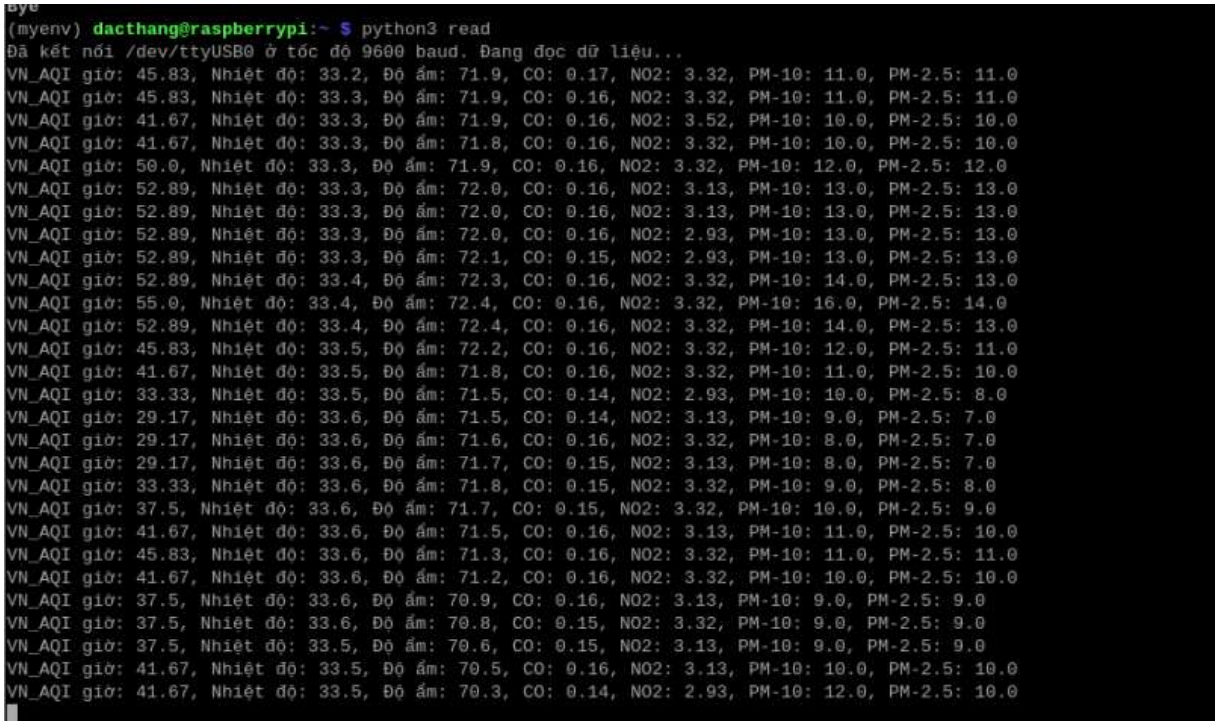
- Chương trình chạy trong một vòng lặp vô hạn (while True), liên tục gọi hàm read_sensor_data() để đọc dữ liệu mới từ Arduino, đảm bảo thu thập dữ liệu theo thời gian thực.



Hình 4.3 Lưu đồ thuật toán Raspberry đọc dữ liệu từ Arduino

- Thư viện sử dụng: Module pyserial của Python là lựa chọn tiêu chuẩn để giao tiếp Serial trên Raspberry Pi.
- Logic nhận dữ liệu :

1) Khởi tạo cổng Serial: Mở cổng Serial tương ứng với kết nối Arduino Uno (thường là /dev/ttyACM0 hoặc /dev/ttyUSB0 trên Raspberry Pi), với tốc độ baudrate phù hợp (ví dụ: 9600 bps) đã cấu hình trên Arduino.
2) Đọc dữ liệu: Chương trình Python sẽ liên tục lắng nghe và đọc từng dòng dữ liệu (kết thúc bằng ký tự xuống dòng \n) từ cổng Serial.
3) Phân tích cú pháp (Parsing): Sau khi nhận được một dòng dữ liệu, chuỗi đó sẽ được phân tích cú pháp để tách các giá trị riêng lẻ (PM2.5, PM10, CO, NO2, Nhiệt độ, Độ ẩm). Nếu dữ liệu gửi từ Arduino là CSV, Python sẽ dùng hàm split (,) để chia chuỗi thành các phần tử. Các giá trị này sau đó được chuyển đổi từ kiểu chuỗi sang kiểu số (float hoặc int) để có thể xử lý toán học.
4) Xử lý lỗi: Bao gồm các cơ chế kiểm tra lỗi để đảm bảo dữ liệu nhận được hợp lệ (ví dụ: kiểm tra số lượng phần tử, giá trị có nằm trong khoảng hợp lý không).



Hình 4.4 Kết quả hiển thị trên Raspberry

4.3.2. Lưu trữ dữ liệu vào MySQL

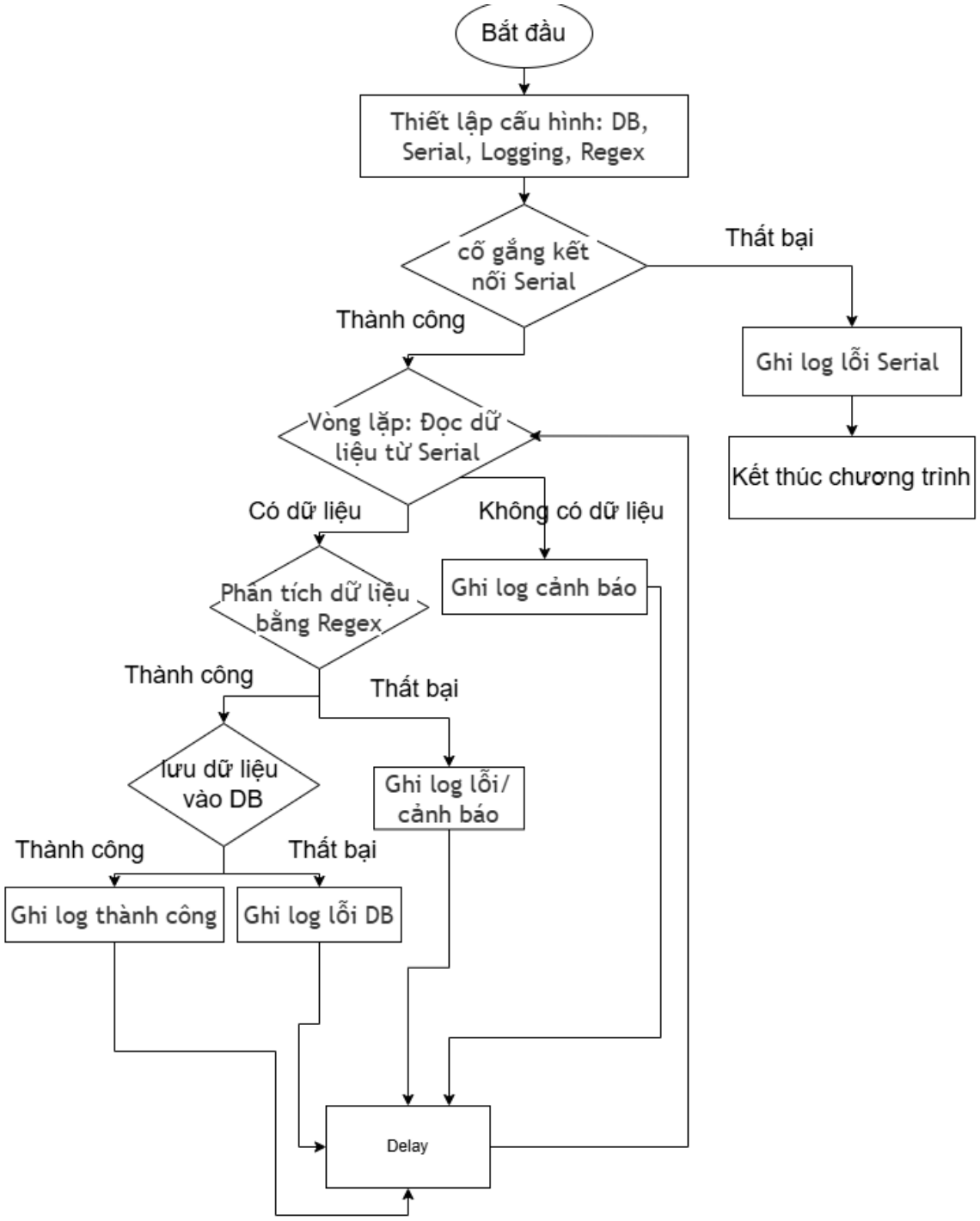
- Dữ liệu đã được xử lý, bao gồm cả VN_AQI, sẽ được lưu trữ vào cơ sở dữ liệu MySQL cục bộ trên Raspberry Pi 5.
- Thiết lập cơ sở dữ liệu :
- Cài đặt MySQL Server trên Raspberry Pi
- Tạo một cơ sở dữ liệu và một người dùng với quyền truy cập phù hợp.

- Tạo một bảng để lưu trữ dữ liệu.
- Cấu trúc bảng MySQL

id	Ngày giờ	VN_AQI giờ	Nhiệt độ	Độ ẩm	CO	NO2	PM-10	PM-2.5

- Logic lưu trữ:

1) Kết nối đến DB: Chương trình Python sẽ thiết lập kết nối đến cơ sở dữ liệu MySQL bằng thông tin đăng nhập.
2) Chèn dữ liệu: Tạo câu lệnh SQL để chèn các giá trị cảm biến và AQI đã xử lý vào bảng
3) Xử lý lỗi: Bắt các ngoại lệ (exceptions) để xử lý các lỗi kết nối hoặc lỗi ghi dữ liệu vào DB.
4) Đóng kết nối DB sau khi hoàn tất thao tác hoặc sử dụng with statement để tự động quản lý kết nối



Hình 4.5 Lưu đồ quá trình thu thập và ghi dữ liệu cảm biếm vào cơ sở dữ liệu

```
MariaDB [sensor_data]> SELECT * FROM pass;
```

id	Ngày giờ	VN_AQI giờ	Nhiệt độ	Độ ẩm	CO	NO2	PM-10	PM-2.5
1	2025-06-14 00:02:43	2.95	28.40	79.00	0.160	3.130	1.00	0.00
2	2025-06-14 00:04:53	2.95	28.40	79.00	0.160	3.130	1.00	0.00
3	2025-06-14 00:08:38	2.95	28.40	79.00	0.160	3.130	1.00	0.00
4	2025-06-14 00:09:39	2.95	28.40	79.00	0.160	3.130	1.00	0.00
5	2025-06-14 00:10:39	2.95	28.40	79.00	0.160	3.130	1.00	0.00
6	2025-06-14 00:11:39	2.95	28.40	79.00	0.160	3.130	1.00	0.00
7	2025-06-14 00:12:39	2.95	28.40	79.00	0.160	3.130	1.00	0.00
8	2025-06-14 00:13:39	2.95	28.40	79.00	0.160	3.130	1.00	0.00
9	2025-06-14 00:14:39	2.95	28.40	79.00	0.160	3.130	1.00	0.00
10	2025-06-14 02:48:30	2.95	28.40	79.00	0.160	3.130	1.00	0.00
11	2025-06-14 02:49:19	2.95	28.40	79.00	0.160	3.130	1.00	0.00
12	2025-06-14 02:50:19	2.95	28.40	79.00	0.160	3.130	1.00	0.00
13	2025-06-14 02:51:19	2.95	28.40	79.00	0.160	3.130	1.00	0.00
14	2025-06-14 02:52:19	2.95	28.40	79.00	0.160	3.130	1.00	0.00
15	2025-06-14 02:53:19	2.95	28.40	79.00	0.160	3.130	1.00	0.00
16	2025-06-14 02:54:19	2.95	28.40	79.00	0.160	3.130	1.00	0.00
17	2025-06-14 02:55:19	2.95	28.40	79.00	0.160	3.130	1.00	0.00
18	2025-06-14 02:56:19	2.95	28.40	79.00	0.160	3.130	1.00	0.00
19	2025-06-14 03:21:50	112.08	28.50	86.20	0.100	2.150	49.00	40.00
20	2025-06-14 03:22:50	112.08	28.50	86.20	0.110	2.150	50.00	40.00
21	2025-06-14 03:23:50	109.62	28.50	86.20	0.110	2.150	48.00	39.00
22	2025-06-14 03:25:15	104.69	29.00	84.60	0.120	2.350	45.00	37.00
23	2025-06-14 03:26:15	109.62	29.00	84.50	0.120	2.540	46.00	39.00
24	2025-06-14 03:27:15	107.16	29.00	84.50	0.120	2.350	45.00	38.00
25	2025-06-14 03:28:15	104.69	29.00	84.40	0.130	2.350	44.00	37.00
26	2025-06-14 03:29:15	107.16	29.10	84.40	0.120	2.540	45.00	38.00
27	2025-06-14 03:30:15	104.69	29.10	84.40	0.130	2.540	44.00	37.00
28	2025-06-14 03:31:15	104.69	29.10	84.30	0.130	2.350	44.00	37.00
29	2025-06-14 13:22:34	3.69	30.70	78.40	0.220	3.910	0.00	0.00
30	2025-06-14 13:41:49	67.61	31.20	76.80	0.150	3.130	20.00	20.00
31	2025-06-14 13:42:49	67.61	31.20	77.00	0.150	2.740	20.00	20.00
32	2025-06-14 13:43:49	67.61	31.10	77.10	0.150	3.130	20.00	20.00
33	2025-06-14 13:44:49	67.61	31.10	77.20	0.140	2.930	20.00	20.00
34	2025-06-14 13:45:49	67.61	31.10	77.20	0.140	2.930	20.00	20.00
35	2025-06-14 13:46:49	65.51	31.10	77.30	0.150	2.930	19.00	19.00
36	2025-06-14 16:04:13	4.17	27.10	62.70	0.080	1.760	1.00	1.00
37	2025-06-14 16:05:13	4.17	27.10	62.60	0.090	1.960	1.00	1.00
38	2025-06-14 16:06:13	4.17	27.10	62.70	0.080	1.760	1.00	1.00
39	2025-06-14 16:07:13	4.17	27.10	62.70	0.080	1.760	2.00	1.00
40	2025-06-16 10:12:52	37.50	32.90	72.70	0.170	3.130	10.00	9.00

Hình 4.6 Kết quả hiển thị trên MySQL

CHƯƠNG 5 : MÔ HÌNH MẠNG NƠ-RON VÀ GIAO DIỆN WEBAPP

5.1. Lý do lựa chọn mạng LSTM cho bài toán dự đoán AQI

5.1.1. Bản chất của bài toán dự báo AQI

Là một bài toán chuỗi thời gian (time series prediction), dữ liệu AQI là các chuỗi giá trị theo thời gian (theo giờ hoặc theo ngày) và có quy luật biến đổi tuần hoàn hoặc theo xu hướng

Mỗi điểm dữ liệu không độc lập với nhau, mà phụ thuộc vào các giá trị trước đó: Ví dụ: AQI lúc 10h sáng sẽ phụ thuộc vào nhiệt độ, độ ẩm, khí thải và AQI của các giờ trước đó

5.1.2. Tổng quan về mạng LSTM (Long Short-Term Memory)

LSTM là một loại RNN nâng cao

- LSTM được thiết kế để khắc phục nhược điểm của mạng RNN cơ bản – đặc biệt là hiện tượng vanishing gradient khi huấn luyện chuỗi dài
- LSTM có cấu trúc bao gồm 3 cổng điều khiển:
    - Forget gate (cổng quên): Quyết định thông tin nào trong trạng thái cũ cần loại bỏ
    - Input gate (cổng đầu vào): Chọn lọc thông tin mới cần thêm vào bộ nhớ
    - Output gate (cổng đầu ra): Điều chỉnh thông tin cần xuất ra từ bộ nhớ

Nhờ vào bộ nhớ có điều khiển, LSTM có khả năng ghi nhớ thông tin quan trọng trong chuỗi dài

5.1.3. Lí do chọn LSTM

- Khả năng ghi nhớ dài hạn:
    - AQI biến động theo ngày, theo tuần (chu kỳ), và bị ảnh hưởng bởi điều kiện thời tiết, giao thông, v.v.
    - Các biến động này cần được ghi nhớ trong nhiều giờ, nhiều ngày → LSTM rất phù hợp
- Khắc phục vấn đề gradient biến mất
    - RNN thường "quên" thông tin cũ khi chuỗi quá dài (do gradient gần như bằng 0 khi truyền ngược qua nhiều bước)

- LSTM khắc phục điều này thông qua cell state được truyền dài hạn, giúp giữ thông tin xuyên suốt thời gian

- Khả năng xử lý dữ liệu tuần tự (sequence-dependent)
  - Các mạng khác như MLP (Perceptron truyền thẳng), CNN (mạng tích chập) không được thiết kế để xử lý trình tự thời gian
  - Mạng LSTM hoạt động theo thứ tự thời gian, nên học được sự phụ thuộc tạm thời giữa các bước thời gian
- Hỗ trợ huấn luyện theo cửa sổ thời gian (sliding window)
- Khả năng xử lí bài toán vượt trội hơn so với các kiến trúc khác

Bảng 5.1 So sánh các mạng nơ-ron

Mạng nơ-ron	**Xử lý tuần tự**	**Ghi nhớ dài hạn**	**Phù hợp chuỗi thời gian**	**Độ phức tạp**	**Ghi nhớ thông tin xa**
MLP	Không	Không	Không	Thấp	Không
CNN	Một phần	Không	Giới hạn	Trung bình	Không tốt
RNN	Có	Có nhưng yếu	Có	Trung bình	Có nhưng dễ quên
LSTM	**Có**	**Có mạnh**	**Rất phù hợp**	**Cao hơn**	**Tốt**

Qua các yếu tố so sánh trên ta thấy **LSTM** vượt trội khi xử lý các chuỗi dài có tính quy luật phức tạp như AQI theo giờ/ngày

5.1.4. Số lớp LSTM và số lượng nơ-ron trong mỗi lớp

Cấu trúc mạng chi tiết phụ thuộc vào dữ liệu thực nghiệm, nhưng có một số quy tắc chung:

- Số lớp ẩn: Thông thường bắt đầu với 1–2 lớp LSTM/GRU. Việc xếp chồng (stack) nhiều lớp giúp học đặc trưng cao cấp hơn nhưng dễ gây quá khớp và tốn thời gian huấn luyện. Ví dụ, có thể thử 2–3 lớp LSTM nối tiếp với nhau
- Số neuron (đơn vị) mỗi lớp: Phổ biến là lựa chọn lũy thừa của 2 (32, 64, 128, 256). Có thể bắt đầu với ~50–100 đơn vị mỗi lớp như ví dụ minh hoạ trong các tài liệu hướng dẫn, sau đó tăng/giảm để cân bằng độ chính xác và tính toán

- Hàm kích hoạt: LSTM thường dùng hàm tanh trong các đơn vị nhớ; lớp đầu ra (Dense) dùng linear cho bài toán dự báo giá trị liên tục AQI. Nếu dùng CNN, thường dùng ReLU sau mỗi tầng tích chập để tạo phi tuyến tính
- Dropout và regularization: Để tránh quá khớp, nên chèn dropout (ví dụ 0.2–0.5) giữa các lớp LSTM hoặc sau lớp Dense. Có thể sử dụng L2 regularization nếu cần
- Hàm mất mát và tối ưu: Đối với bài toán hồi quy AQI, thường chọn hàm lỗi MSE (mean squared error) hoặc MAE (mean absolute error), tối ưu bằng thuật toán Adam với tốc độ học (learning rate) ban đầu khoảng 0.001
- Batch size và epochs: Kích thước batch (ví dụ 32–64) và số epoch (50–200) được điều chỉnh tùy dữ liệu. Kỹ thuật như early stopping (dừng khi không cải thiện trên tập xác thực) cũng nên áp dụng

Lựa chọn kiến trúc gồm 2 lớp LSTM để học sâu hơn các mối quan hệ thời gian:

- Lớp đầu tiên học mối quan hệ ngắn hạn trong chuỗi dữ liệu (theo từng giờ)
- Lớp thứ hai học các xu hướng dài hạn hoặc tính mùa vụ (vd: ô nhiễm tích lũy, dao động ngày – đêm, hiệu ứng cuối tuần)

Cân bằng độ phức tạp và khả năng tổng quát:

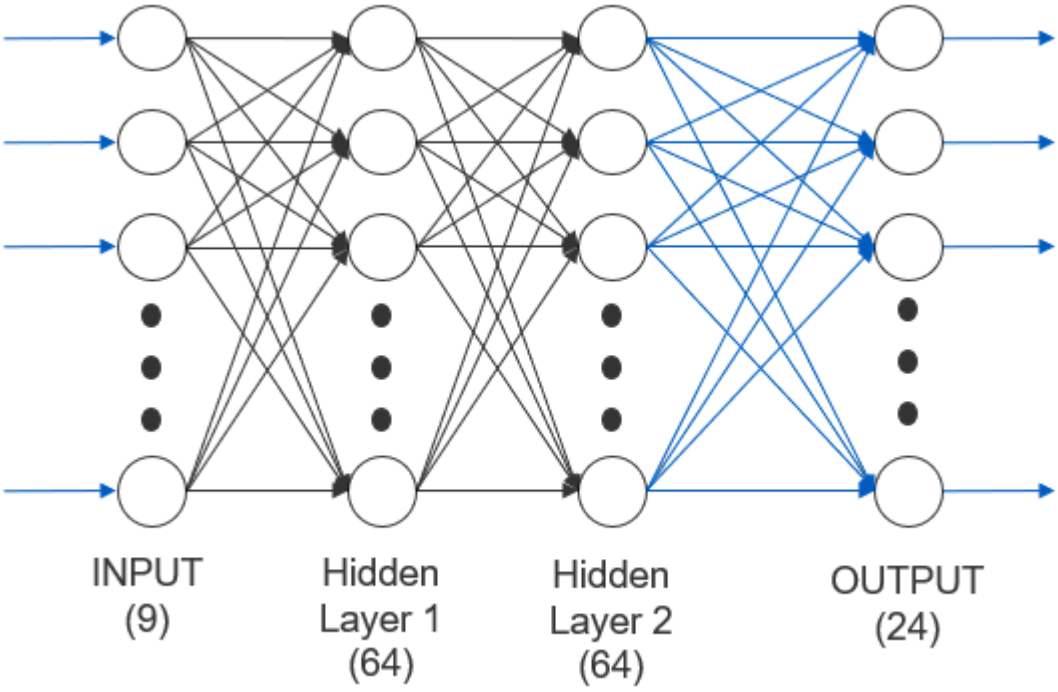
- Một lớp có thể không đủ khả năng học toàn bộ xu hướng
- Hơn 2 lớp dễ gây overfitting nếu không có nhiều dữ liệu và yêu cầu tính toán cao hơn

→ Chọn 2 lớp là điểm tối ưu giữa khả năng biểu diễn và độ ổn định huấn luyện

Chọn 64 nơ-ron mỗi lớp:

- 64 nơ-ron cho phép mô hình ghi nhớ được nhiều chiều đặc trưng tại mỗi bước thời gian, giúp cải thiện khả năng biểu diễn chuỗi dữ liệu phức tạp, gồm:
    - Sự thay đổi nhanh của AQI
    - Mối quan hệ giữa các chất ô nhiễm và điều kiện thời tiết
    - Dự báo ảnh hưởng cộng dồn
- Đã được kiểm chứng thực nghiệm: Qua quá trình thử nghiệm thì mỗi lớp có 64 nơ-ron sẽ cho hiệu quả tối ưu nhất: Sai số MSE giảm đều theo epoch, MAE dưới 0.12 sau 30 epoch → phù hợp yêu cầu thực tế
- Thử với lớp ẩn gồm 32 nơ-ron: Mô hình có tốc độ học chậm hơn

- Thử với lớp ẩn gồm 128 nơ-ron: sai số có giảm nhưng không đáng kể, nhưng thời gian huấn luyện lại dài gấp đôi



Hình 5.1 Kiến trúc mạng nơ-ron

5.1.5. Hàm mất mát (Loss function) và Thuật toán tối ưu (Optimaze)

a) Hàm mất mát MSE

Được sử dụng để đo độ sai lệch trung bình bình phương giữa giá trị dự đoán và giá trị thực tế. Công thức tổng quát như sau:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad (2)$$

Trong đó:

y_i: giá trị thực tế

$\hat{y}_i$: giá trị dự đoán của mô hình

n: số lượng mẫu đầu ra (trong bài toán này là 24 giá trị AQI mỗi lần dự đoán

MSE trừng phạt sai số lớn mạnh hơn do bình phương sai số

Nhạy cảm với các dự đoán lệch nhiều → giúp mô hình học tốt hơn trong việc hạn chế sai số lớn

MSE là một hàm trơn, khả vi, rất phù hợp với các thuật toán tối ưu như Adam, SGD, giúp huấn luyện mô hình dễ dàng và ổn định

b) Lựa chọn MSE là hàm mất mát cho bài toán

Phù hợp với bài toán hồi quy: Bài toán dự đoán AQI theo chuỗi thời gian là một bài toán hồi quy đa đầu ra (multi-output regression). MSE là hàm mất mát phổ biến và tiêu chuẩn nhất cho hồi quy, đặc biệt là khi đầu ra là các giá trị liên tục và có ý nghĩa vật lý, như chỉ số chất lượng không khí

- MSE ưu tiên sai số nhỏ, trừng phạt mạnh sai số lớn:
  - Trong bài toán dự đoán AQI, những sai lệch lớn có thể gây ra hệ quả nghiêm trọng trong việc cảnh báo sai mức độ ô nhiễm không khí
  - MSE giúp mô hình tránh các dự đoán cực sai bằng cách trừng phạt chúng mạnh hơn
- Tương thích tốt với mô hình LSTM:
  - MSE là một hàm lồi và liên tục, giúp thuật toán học sâu hội tụ nhanh và ổn định hơn so với một số hàm mất mát khác
  - Trong PyTorch, nn.MSELoss() được tối ưu rất tốt cho các mô hình như nn.LSTM

Bảng 5.2 So sánh các hàm mất mát

Hàm mất mát	**Ưu điểm chính**	**Nhược điểm**
MSE	**Nhấn mạnh sai số lớn, phù hợp hồi quy**	**Nhạy cảm với outlier**
MAE	Ít nhạy cảm với outlier	Không trơn tại 0 → khó tối ưu hơn
Huber Loss	Kết hợp MSE + MAE, chống outlier tốt hơn	Phức tạp hơn, cần chọn thêm tham số delta
SmoothL1	Ổn định, mượt mà	Kết hợp nhưng hiệu quả không vượt trội MSE trong dự đoán AQI

Trong bài toán dự đoán AQI, MSE là lựa chọn hợp lý nhất, vừa đảm bảo độ chính xác cao, vừa đơn giản hóa huấn luyện

c) Thuật toán tối ưu Adam

Adam (Adaptive Moment Estimation) là một thuật toán tối ưu hóa sử dụng trong học sâu, kết hợp ưu điểm của Momentum và RMSProp. Adam được thiết kế để tự động điều chỉnh tốc độ học (learning rate) cho từng tham số, giúp quá trình huấn luyện nhanh hơn và ổn định hơn, đặc biệt hiệu quả trong mô hình LSTM với dữ liệu chuỗi thời gian như AQI

Cách hoạt động của Adam:

Adam duy trì hai giá trị trung bình động :

- m_t - Trung bình cộng cấp 1 (Giống Monentum)

$$m_t = \beta_1.m_{t-1} + (1-\beta_1).g_t \qquad (3)$$

Trong đó g_t là gradient tại thời điểm t

- v_t - Trung bình cộng cấp 2 (giống RMSProp – bình phương gradient)

$$v_t = \beta_2.v_{t-1} + (1-\beta_2).g_t^2 \qquad (4)$$

Adam sau đó sử dụng 2 giá trị này để điều chỉnh bước cập nhật tham số :

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \hat{v}_t = \frac{v_t}{1-\beta_2^t} \qquad (5)$$

$$\theta_{t+1} = \theta_t - \alpha.\frac{\hat{m}_t}{\sqrt{v_t + \varepsilon}} \qquad (6)$$

Các tham số chính trong Adam:

Bảng 5.3 Các tham số chính trong Adam

Tham số	**Ý nghĩa**	**Giá trị khuyến nghị (mặc định)**
learning_rate (α)	Tốc độ học – quy định kích thước bước cập nhật	0.001
beta1 (β1)	Hệ số suy giảm của trung bình động cấp 1 (momentum)	0.9
beta2 (β2)	Hệ số suy giảm của trung bình động cấp 2 (bình phương grad)	0.999
epsilon (ϵ)	Hằng số rất nhỏ để tránh chia cho 0	1e-8
weight_decay (nếu có)	Hệ số điều chuẩn L2 – chống overfitting	0 (có thể dùng 1e-5)

Lí do lựa chọn Adam để huấn luyện LSTM:

- Tự động điều chỉnh learning rate cho từng tham số → nhanh hội tụ
- Làm việc tốt với dữ liệu chuỗi thời gian không ổn định
- Phù hợp với mạng LSTM có nhiều tham số và độ sâu
- Ít cần tinh chỉnh so với SGD → thuận tiện khi huấn luyện mô hình thực tế
- Tránh bị "mắc kẹt" trong các vùng gradient nhỏ hoặc dao động mạnh

Kết luận: Trong quá trình huấn luyện mô hình LSTM để dự đoán chỉ số AQI theo chuỗi thời gian, nhóm sử dụng thuật toán tối ưu Adam với tốc độ học (learning rate) là 0.001. Adam được lựa chọn vì khả năng kết hợp giữa Momentum và RMSProp giúp cập nhật các trọng số một cách hiệu quả và ổn định, đặc biệt phù hợp với mô hình LSTM có nhiều lớp và dữ liệu chuỗi có biến động. Các siêu tham số như $\beta_1 = 0.9$ và $\beta_2 = 0.999$ giữ nguyên theo mặc định vì đã được chứng minh hiệu quả trong nhiều nghiên cứu học sâu

5.2. Huấn luyện và đánh giá mô hình

5.2.1. Tiền xử lí dữ liệu và tạo chuỗi thời gian

Dữ liệu môi trường được lấy trực tiếp từ cơ sở dữ liệu MySQL bằng SQLAlchemy. Sau khi kết nối thành công, dữ liệu được xử lý để tách thời gian thành các đặc trưng riêng và chuẩn hóa về cùng một thang đo. Tiếp đó, dữ liệu được chia thành các chuỗi 24 giờ liên tục theo kỹ thuật sliding window, phục vụ cho bài toán dự đoán chuỗi thời gian. Cụ thể, mỗi chuỗi đầu vào là tập dữ liệu của 24 giờ gần nhất và đầu ra là chỉ số AQI dự đoán cho 24 giờ kế tiếp. Việc tích hợp dữ liệu trực tiếp từ MySQL giúp mô hình có khả năng dự đoán theo thời gian thực, phù hợp với yêu cầu ứng dụng thực tiễn về cảnh báo chất lượng không khí.

Tuy nhiên, thực tế trong mô hình này tập dữ liệu huấn luyện chỉ bao gồm khoảng hơn 300 mẫu được lấy với tần suốt 1 giờ 1 lần, đây là số lượng dữ liệu rất nhỏ để huấn luyện mô hình học sâu như LSTM, thì tình trạng overfitting có thể xảy ra và dự báo không chính xác trên dữ liệu thực tế mới

Dữ liệu được chia thành hai phần theo tỷ lệ 8:2, tương ứng với 80% để huấn luyện mô hình và 20% để kiểm tra mô hình sau khi huấn luyện. Quá trình chia dữ liệu được thực hiện sau bước chuẩn hóa bằng “MinMaxScaler” và sau khi tạo chuỗi đầu vào – đầu ra theo cửa sổ trượt (sliding window). Việc không xáo trộn dữ liệu là bắt buộc do tính chất chuỗi thời gian, nhằm đảm bảo mô hình học được xu hướng và quy luật thời gian một cách chính xác

"MinMaxScaler" là một kỹ thuật chuẩn hóa dữ liệu tuyến tính, thuộc nhóm feature scaling. Phương pháp này đưa toàn bộ giá trị của một đặc trưng (feature) về một khoảng giá trị nhất định, thường là từ 0 đến 1.

Công thức :

$$x_{scaled} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \quad (7)$$

Trong đó:

x : Giá trị gốc cần chuẩn hóa

$x_{\min}$, $x_{\max}$: Giá trị nhỏ nhất và lớn nhất của đặc trưng đó trong tập dữ liệu

x_{scaled} : Giá trị chuẩn hóa

a) Tách thời gian thành đặc trưng riêng

Sau khi lấy dữ liệu từ bảng environmental_data cột "Thời gian" sẽ được xử lí để tách ra thành các đặc trưng riêng

```
df['Ngày giờ'] = pd.to_datetime(df['Ngày giờ'])
df['Năm'] = df['Ngày giờ'].dt.year
df['Tháng'] = df['Ngày giờ'].dt.month
df['Ngày'] = df['Ngày giờ'].dt.day
df['Giờ'] = df['Ngày giờ'].dt.hour
```

Trong đó bao gồm "Giờ", "Ngày", "Tháng", "Năm"

b) Chuẩn hóa dữ liệu đầu vào

Sau đó dữ liệu sẽ được làm sạch và chuẩn hóa về khoảng [0;1] trước khi đưa vào mô hình huấn luyện sử dụng "MinMaxScaler" để chuẩn hóa dữ liệu. Dữ liệu được chia thành các chuỗi liên tiếp bằng kỹ thuật cửa sổ trượt (sliding window) với mỗi chuỗi dữ liệu 24 giờ đầu vào, mô hình sẽ học để dự đoán 24 giờ AQI tiếp theo. Điều này mô phỏng chính xác cách vận hành trong thực tế, nơi dữ liệu môi trường liên tục được ghi nhận và sử dụng để dự đoán trước.

Kết quả sau khi xử lí:

- Dữ liệu đầu vào (X): có dạng số lượng mẫu × 24 × 9 (24 giờ, 9 đặc trưng)
- Dữ liệu đầu ra (y): là chuỗi dự đoán AQI cho 24 giờ tiếp theo

- Chuẩn hóa và giải nén: chỉ số AQI được chuẩn hóa riêng và có thể khôi phục về giá trị gốc sau dự đoán.
- Lưu dữ liệu gốc: Sau xử lý, dữ liệu gốc cũng được lưu ra file Excel (du_lieu_goc.xlsx) để phục vụ minh họa và kiểm chứng

	CO	NO2	PM-10	PM-2.5
count	311.000000	311.000000	311.000000	311.000000
mean	0.456806	0.188200	0.486685	0.307130
std	0.184379	0.169882	0.204208	0.173061
min	0.000000	0.000000	0.000000	0.000000
25%	0.333333	0.096386	0.307692	0.191860
50%	0.466667	0.144578	0.474359	0.267442
75%	0.566667	0.228916	0.602564	0.386628
max	1.000000	1.000000	1.000000	1.000000

Hình 5.2 Dữ liệu sau khi được chuẩn hóa về đoạn (0;1)

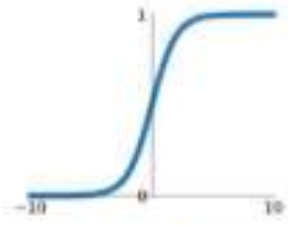
Trong PyTorch LSTM sẽ sử dụng các hàm kích hoạt ngầm định trong cấu trúc mạng

Bảng 5.4 Các hàm kích hoạt trong LSTM

LSTM	Hàm kích hoạt
Input gate	Sigmoid
Forget gate	Sigmoid
Output gate	Sigmoid
Cell gate	Tanh
Hidden state output	Tanh

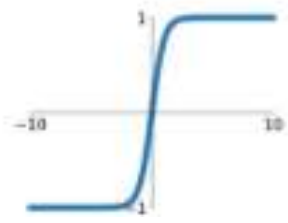
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



tanh

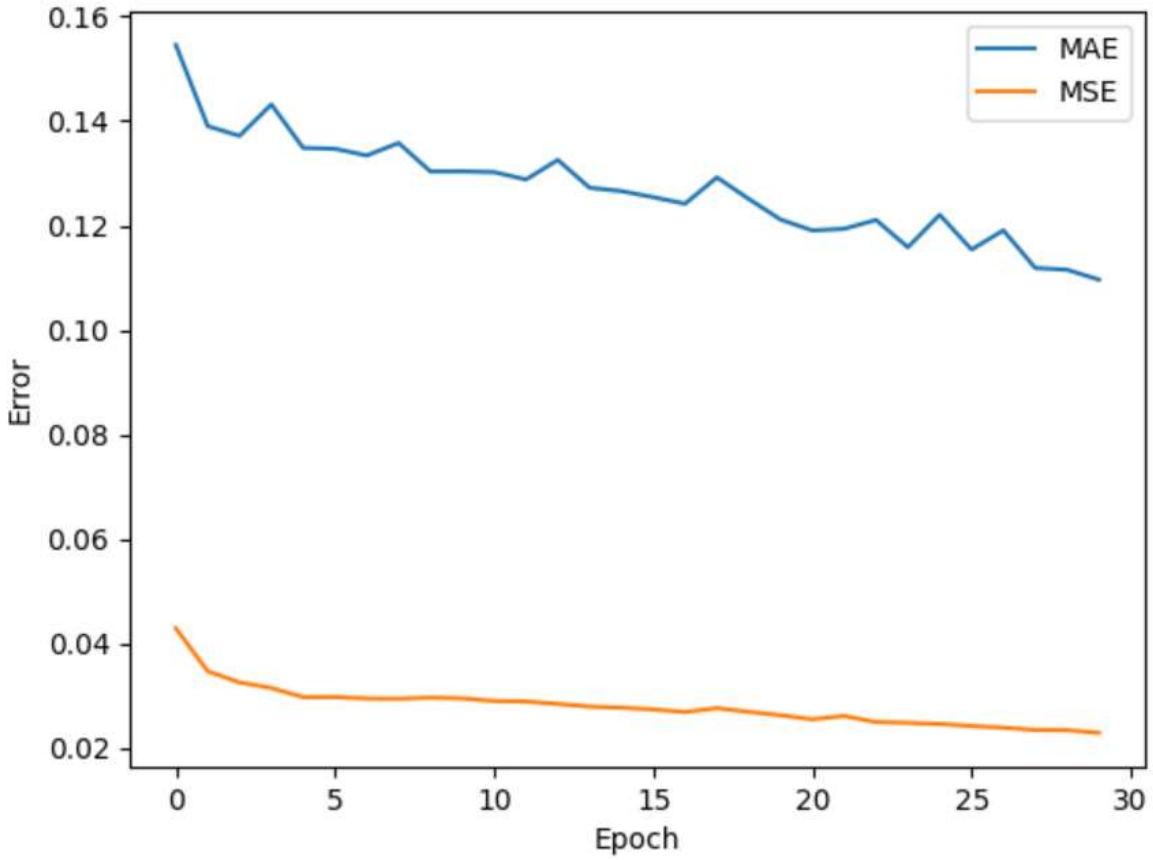
$$\tanh(x)$$



Hình 5.3 Hàm kích hoạt Sigmoid và Tanh

5.2.2. Đánh giá mô hình

a) Đánh giá trên tập huấn luyện



Hình 5.4 Đồ thị giá trị MSE & MAE

Tệp dữ liệu được chia thành 30 Epoch để huấn luyện với 80% để học và 20% để kiểm tra

Chỉ số	Giá trị ban đầu (Epoch 1)	Giá trị tốt nhất (Epoch 30)	Mức độ cải thiện
MSE	0.0430	**0.0230**	↓ 46.5%
MAE	0.1545	**0.1097**	↓ 29.0%
Loss	0.0812	**0.0180**	↓ 77.8%

Epoch 1-10 (Giai đoạn hội tụ đầu)

- MSEloss giảm mạnh từ 0.0812 → 0.0269
- MSE giảm đều từ 0.043 → 0.0296
- MAE dao động nhẹ nhưng giảm xuống từ 0.1545 → 0.1303
- Mô hình hội tụ nhanh trong các epoch đầu tiên

Epoch 11-20 (Ổn định)

- MSEloss giảm từ 0.0255 → 0.0223
- MAE: giảm chậm, thấp nhất ở epoch 20: 0.1211
- Mô hình đang tối ưu tốt hơn

Epoch 21-30 (Tối ưu hiệu quả)

- MSEloss giảm từ 0.0212 → 0.018
- MAE: Từ 0.119 → 0.1097
- MSE: Thấp nhất tại epoch 30: 0.023

Epoch	MSE	MAE	Nhận xét
1	0.0430	0.1545	Mở đầu, sai số còn lớn
10	0.0296	0.1303	Giảm mạnh, học ổn định
20	0.0263	0.1211	Cải thiện rõ ràng
24	0.0249	0.1158	Bắt đầu chạm mốc thấp
28	0.0235	0.1119	Xu hướng hội tụ rõ nét
30	0.0230	0.1097	Tối ưu nhất toàn bộ quá trình

Bảng 5.5 Đánh giá hiệu suất

Tiêu chí	Đánh giá
Hiệu suất tổng thể	Rất tốt
Mức độ hội tụ	Mượt, ổn định, không overfit
MSE < 0.023	→ Sai số chỉ khoảng ±10 AQI sau khi chuyển lại AQI
MAE < 0.11	→ Dự báo khá gần thực tế, tin cậy trong ứng dụng cảnh báo

Trong một hệ thống học sâu, việc ghi lại quá trình huấn luyện và lưu mô hình là vô cùng quan trọng, nhằm mục tiêu:

- Theo dõi chất lượng huấn luyện qua từng epoch như loss, MAE, MSE
- Phân tích hiệu quả mô hình theo thời gian, phát hiện sớm dấu hiệu overfitting/underfitting
- Tái sử dụng mô hình đã huấn luyện cho việc dự đoán về sau mà không cần huấn luyện lại từ đầu (tiết kiệm thời gian và tài nguyên)
- So sánh giữa các lần huấn luyện khác nhau để chọn ra phiên bản tốt nhất

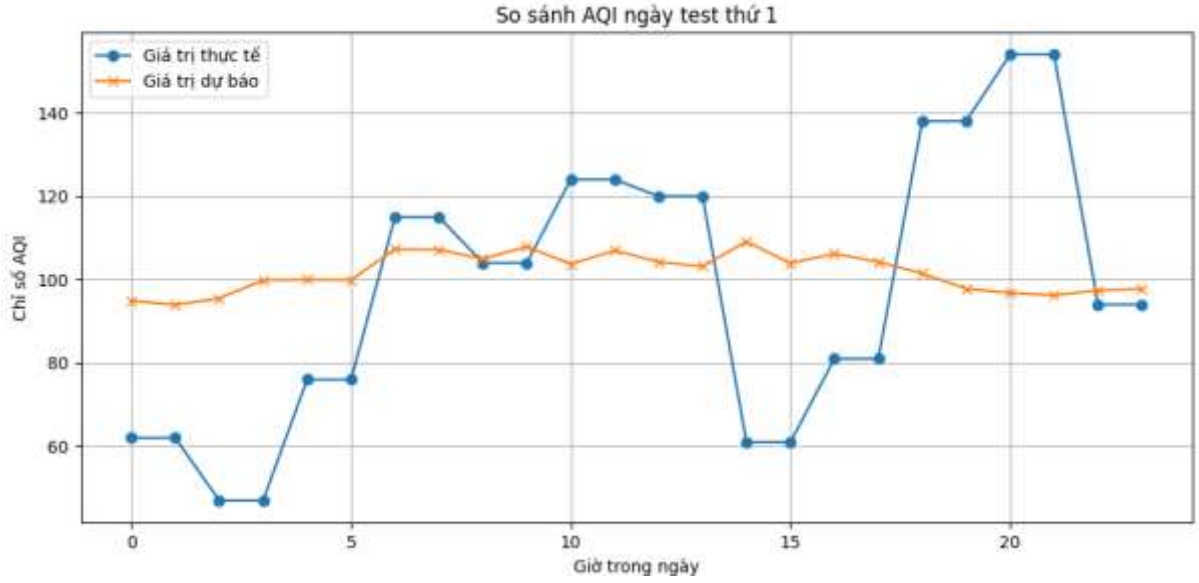
Sau khi huấn luyện xong, mô hình được lưu dưới dạng file .pth, file này lưu lại trạng thái trọng số (weights) của mô hình LSTM, sau này không cần huấn luyện lại, chỉ cần tải lại mô hình để dùng

Sau mỗi epoch huấn luyện, hệ thống sẽ tự động ghi lại các chỉ số đánh giá mô hình như Loss, MAE và MSE vào tệp training_log.csv. Tệp này có thể được mở bằng Excel để trực quan hóa quá trình huấn luyện, giúp phát hiện mô hình có đang học hiệu quả hay không. Sau khi huấn luyện hoàn tất, mô hình LSTM được lưu lại dưới dạng file air_quality_forecast_24h.pth. Nhờ đó, trong các lần sử dụng sau, hệ thống có thể tải lại mô hình này để dự báo chỉ số AQI mà không cần huấn luyện lại, giúp tiết kiệm thời gian và tài nguyên tính toán

b) Đánh giá trên tập kiểm tra

Sau khi huấn luyện mô hình trên 80% dữ liệu, nhóm tiến hành đánh giá trên 20% dữ liệu còn lại (test set) để đảm bảo mô hình hoạt động tốt trên dữ liệu chưa từng thấy. Mô hình được sử dụng để dự đoán AQI cho 24 giờ tiếp theo tại mỗi bước, sau đó kết quả dự báo được so sánh với giá trị thực tế bằng biểu đồ và được đánh giá bằng các chỉ số như MSE và MAE.

Biểu đồ dưới đây minh họa sự tương quan giữa giá trị thực tế và giá trị dự báo trong một ngày bất kỳ của tập test:



Hình 5.5 So sánh kết quả tập kiểm tra và dự đoán

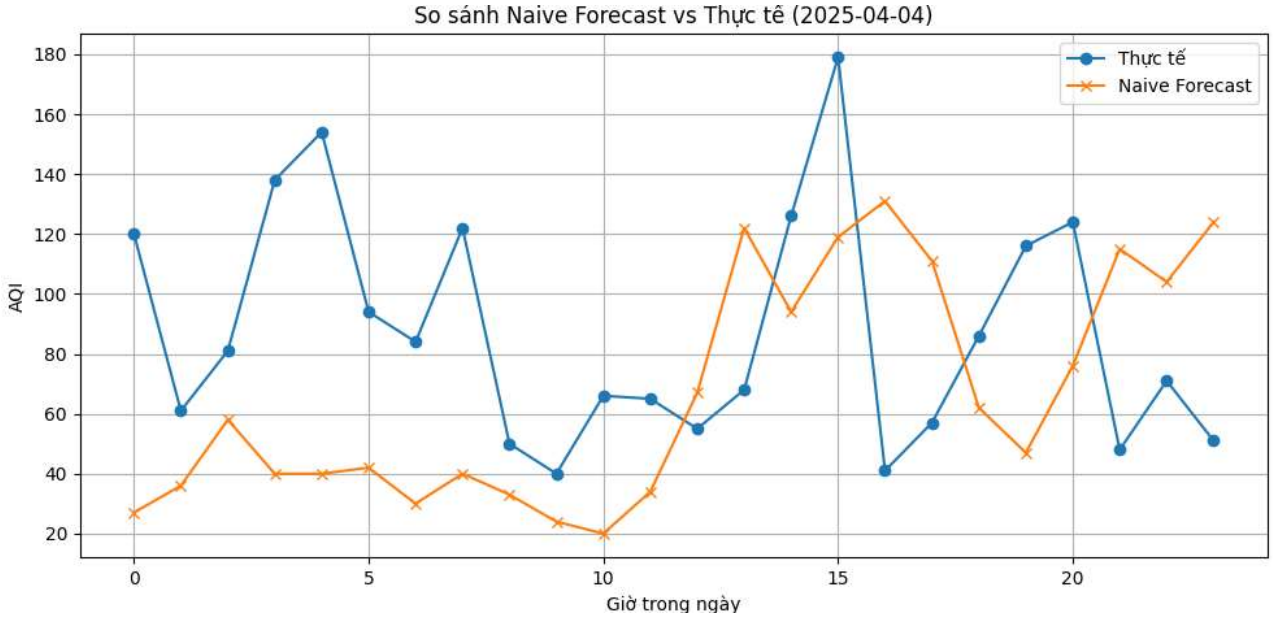
Biểu đồ so sánh giữa giá trị thực tế và giá trị dự báo cho thấy mô hình LSTM hoạt động khá tốt trong việc dự đoán xu hướng chung của chỉ số AQI trong ngày. Tuy nhiên, mô hình chưa thể hiện tốt khả năng dự báo các điểm biến động lớn, đặc biệt vào các khung giờ cao điểm hoặc các thời điểm có sự thay đổi môi trường đột ngột. Điều này là dễ hiểu

vì hàm mất mát MSE có xu hướng làm mượt kết quả dự báo để giảm sai số trung bình, thay vì ưu tiên các điểm cực trị.

Một hạn chế quan trọng ảnh hưởng đến kết quả dự báo là số lượng dữ liệu huấn luyện chưa đủ lớn, dẫn đến mô hình LSTM chưa học được hết các xu hướng đột biến trong thực tế. Đặc biệt, các tình huống AQI tăng vọt vào giờ cao điểm hoặc thời điểm bất thường thường không được mô hình phản ánh chính xác, vì chưa từng xuất hiện trong dữ liệu huấn luyện. Điều này khiến đường dự báo có xu hướng "làm mượt" để giảm sai số trung bình, làm mất đi tính chính xác tại các thời điểm quan trọng.

c) Đánh giá LSTM với Baseline model

So sánh với Baseline model (Navie Forecast), mô hình sẽ tự động tìm 2 ngày có đủ 24 giờ dữ liệu để so sánh với nhau



Hình 5.6 So sánh Thực tế với Navie Forecast

Biểu đồ cho thấy sự khác biệt đáng kể giữa giá trị AQI thực tế và Naive Forecast. Đường Naive (cam) gần như phẳng và dao động nhẹ → cho thấy nó chỉ đơn giản lặp lại giá trị cũ, không bắt kịp biến động. Đường thực tế (xanh) dao động rất mạnh, có đỉnh trên 180, cho thấy sự biến động lớn trong ngày, đặc biệt là giờ 3–4h, 15–16h. Trong khi đó sai số của LSTM chỉ nằm trong khoảng ±10 cho thấy độ chính xác của LSTM là cao hơn rất nhiều

Bảng 5.6 Đánh giá Navie Forecast

Yếu tố đánh giá	Nhận xét
Độ chính xác	Kém – mô hình không phản ánh được các đỉnh hoặc đáy AQI thực tế
Khả năng bắt xu hướng	Thấp – Naive không thay đổi theo điều kiện môi trường mới
Đơn giản, dễ triển khai	Ưu điểm – không cần huấn luyện, không cần mô hình
Tốc độ dự báo	Nhanh – gần như tức thời

Bảng 5.7 So sánh LSTM với Navie Forecast

Tiêu chí	Naive Forecast	LSTM
Bắt xu hướng tăng/giảm	Không	Có
Dự báo các đỉnh bất thường	Không	Có thể có (tuỳ huấn luyện)
MAE, MSE	Cao	Thấp hơn rõ rệt
Khả năng tổng quát hóa	Rất thấp	Tốt hơn đáng kể
Ứng dụng thực tế	Không đủ chính xác để cảnh báo	Có thể dùng để cảnh báo

Kết luận: Từ biểu đồ trên, có thể thấy phương pháp Naive Forecast không đáp ứng được yêu cầu dự báo chất lượng không khí trong các điều kiện biến động. Đường dự báo không phản ánh được các mức đột biến trong AQI, đặc biệt trong các giờ cao điểm. Do đó, mặc dù đơn giản và nhanh chóng, mô hình cơ sở này chỉ mang tính tham khảo. Khi so sánh với mô hình LSTM, độ chính xác của Naive Forecast thấp hơn rõ rệt, thể hiện ở việc không thể theo kịp các xu hướng phức tạp trong chuỗi thời gian thực tế

STT	Ưu điểm	Giải thích
1	**Ứng dụng mô hình học sâu (LSTM)**	Mô hình có khả năng ghi nhớ chuỗi dữ liệu theo thời gian, phù hợp với tính chất liên tục của AQI, giúp cải thiện độ chính xác dự báo.

STT	Ưu điểm	Giải thích
2	**Hiển thị kết quả trực quan**	Dự báo AQI được trình bày dưới dạng bảng dữ liệu 24 giờ, giúp người dùng dễ dàng theo dõi theo từng giờ trong ngày.
3	**Tự động phân loại cảnh báo AQI**	Hệ thống gán cảnh báo theo mức độ ảnh hưởng sức khỏe, từ "Tốt" đến "Nguy hại" dựa trên giá trị AQI trung bình.
4	**Tích hợp giao diện web**	Dễ sử dụng, không cần cài đặt phần mềm chuyên biệt. Có thể chạy trực tiếp trên trình duyệt và chia sẻ cho người dùng khác.
5	**Dữ liệu dễ quản lý và mở rộng**	Dữ liệu được lưu ở định dạng Excel hoặc cơ sở dữ liệu, dễ kiểm tra, cập nhật và chuyển đổi.
6	**Kiến trúc linh hoạt và mở rộng được**	Có thể tích hợp thêm các mô hình khác, mở rộng API, hoặc kết nối với cảm biến ngoài để dự báo thời gian thực.

STT	Nhược điểm	Giải thích
1	**Phụ thuộc vào dữ liệu lịch sử**	Nếu dữ liệu thu thập bị thiếu hoặc không liên tục, mô hình có thể dự báo sai lệch hoặc không học được xu hướng.
2	**Chưa theo kịp biến động đột ngột**	Mô hình LSTM có xu hướng "làm mượt" dự báo, dẫn đến khó phát hiện các đột biến AQI như giờ cao điểm hoặc thời tiết thay đổi bất thường.
3	**Giao diện còn đơn giản**	Hiện chưa có biểu đồ, bản đồ hoặc chức năng cảnh báo trực tiếp; người dùng cần đọc bảng kết quả để hiểu diễn biến.

⇨ Hệ thống dự báo AQI mà nhóm xây dựng có nhiều ưu điểm như khả năng học chuỗi thời gian, tích hợp giao diện web dễ dùng, và có thể mở rộng. Tuy nhiên, hệ thống vẫn còn một số hạn chế như chưa xử lý tốt các biến động đột ngột, chưa có khả năng cập nhật theo thời gian thực và còn phụ thuộc nhiều vào chất lượng dữ liệu đầu vào. Những điểm này sẽ được cải thiện trong các phiên bản nâng cấp sau.

5.3. Xây dựng giao diện Web hiện thị dữ liệu và dự báo

5.3.1. Mục tiêu xây dựng Web

Nhằm giúp người dùng không chuyên có thể dễ dàng theo dõi chỉ số chất lượng không khí và kết quả dự báo từ mô hình, nhóm đã xây dựng một giao diện web đơn giản sử dụng Flask – một framework nhẹ của Python. Giao diện này có khả năng:

- Tự động đọc dữ liệu đầu vào từ các file Excel
- Hiển thị lịch sử chỉ số AQI và các chất ô nhiễm
- Hiển thị kết quả dự báo AQI cho 24 giờ của ngày mai
- Cho phép lọc dữ liệu theo khoảng thời gian tuỳ chọn

5.3.2. Công nghệ sử dụng

Thành phần	**Công cụ / Thư viện**
Backend	Python, Flask
Giao diện	HTML + CSS
Dữ liệu	Excel (.xlsx) sử dụng pandas và openpyxl
Trình duyệt	Google Chrome (mở tự động khi chạy ứng dụng)

5.3.3. Cấu trúc hoạt động

a) Đọc dữ liệu

Ứng dụng sử dụng pandas để đọc dữ liệu từ hai tệp Excel:

du_lieu_goc.xlsx: chứa dữ liệu AQI lịch sử theo từng giờ

du_doan_24h_ngay_mai.xlsx: chứa dự báo AQI cho 24 giờ của ngày tiếp theo

```
def load_data():
    df1 = pd.read_excel("du_doan_24h_ngay_mai.xlsx")
    df2 = pd.read_excel("du_lieu_goc.xlsx")
```

b) Giao diện chính

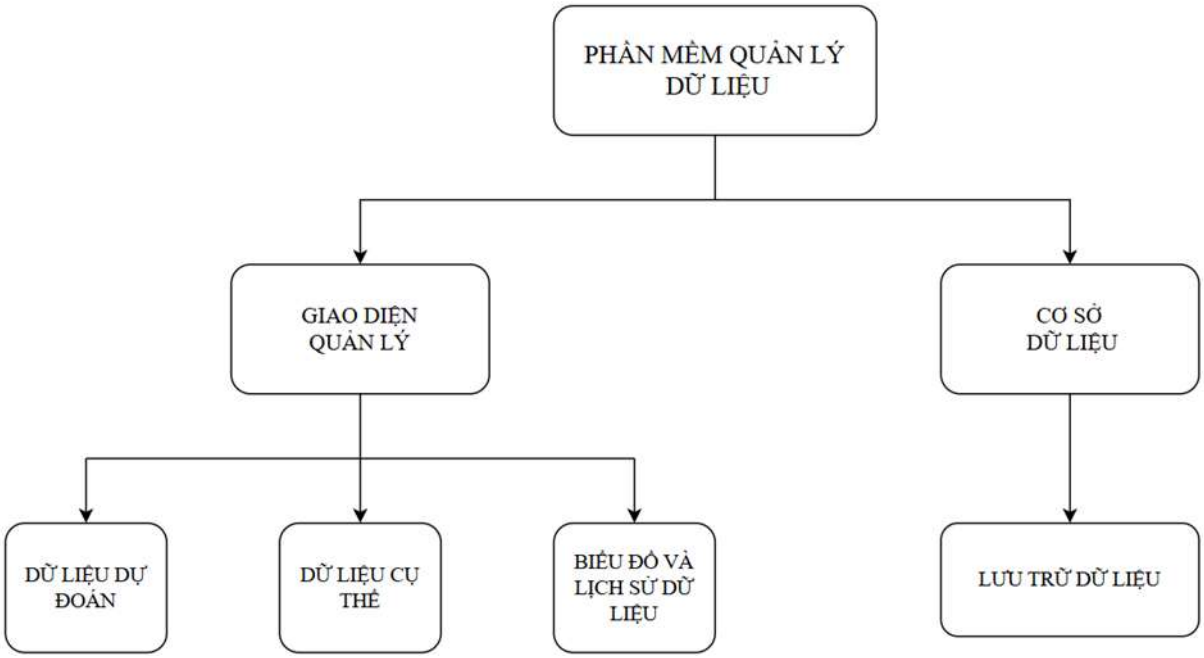
Trang chính (/) trả về một giao diện HTML với ba bảng:

- Dự báo 24h ngày mai
- Lịch sử AQI các giờ
- Biểu đồ

Tất cả dữ liệu được truyền sang giao diện bằng render_template

Khi chạy chương trình thì trình duyệt sẽ tự động mở trang web tại localhost:5000

5.3.4. Kiến trúc phần mềm



Hình 5.7 Sơ đồ khối phần mềm quản lý dữ liệu

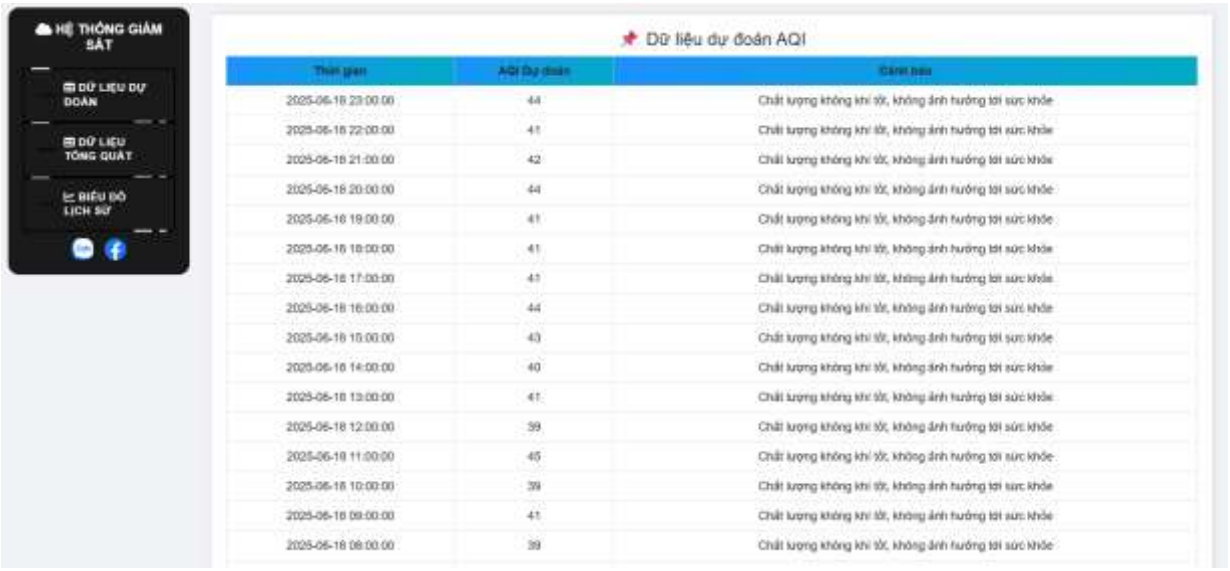
Khối "PHẦN MỀM QUẢN LÝ DỮ LIỆU": Đây là phần trung tâm điều phối, tích hợp các chức năng xử lý, lưu trữ, và hiển thị dữ liệu. Có nhiệm vụ đọc và xử lý dữ liệu, hiển thị thông tin dự báo và lịch sử, kết nối với cơ sở dữ liệu

Nhánh "GIAO DIỆN QUẢN LÝ": Giao diện được xây dựng bằng Flask (Python), cho phép người dùng tương tác trực tiếp với dữ liệu. Gồm 3 chức năng con: Dữ liệu dự đoán, Dữ liệu cụ thể, Biểu đồ và Lịch sử dữ liệu

Nhánh "CƠ SỞ DỮ LIỆU": Đây là thành phần chịu trách nhiệm lưu trữ và tổ chức dữ liệu

5.3.5. Kết quả

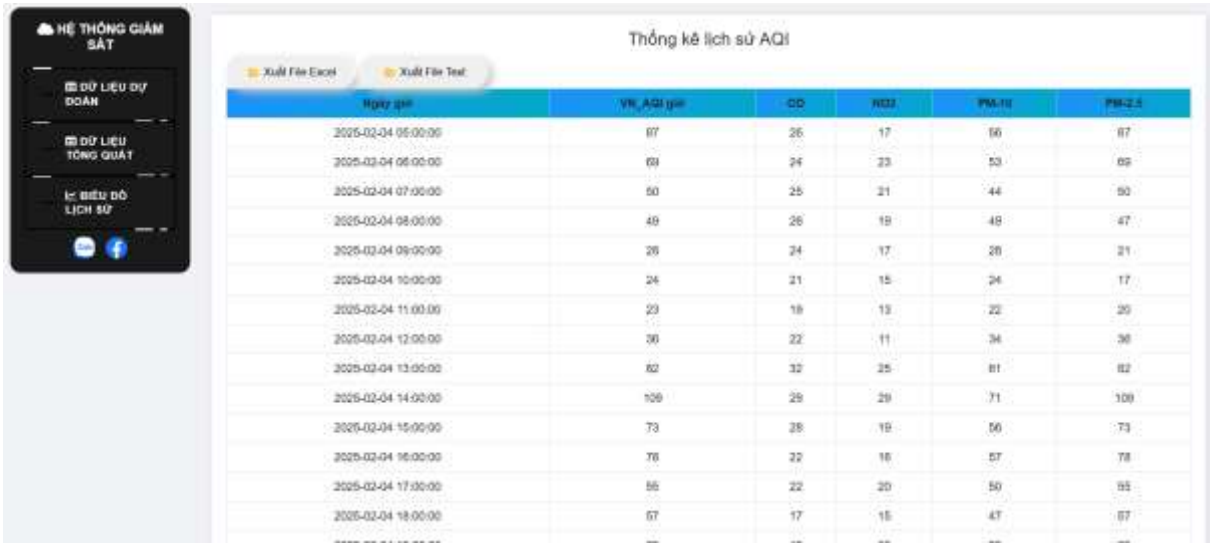
- **Giao diện quản lý:**
  - Dữ liệu dự đoán: Hiển thị thông số AQI dự đoán và đưa ra cảnh báo cụ thể theo thời gian trong tương lai



Hình 5.8 Hiển thị dữ liệu dự đoán ngày hôm sau

Kết quả được hiển thị là kết quả được mô hình LSTM dự đoán sau đó được đưa lên web, kết quả dự đoán còn có sai số lớn và mang tính chất tham khảo

- Dữ liệu cụ thể: Nồng độ cụ thể các chất ô nhiễm (PM2.5, PM10, CO, NO2) và nhiệt độ, độ ẩm cụ thể



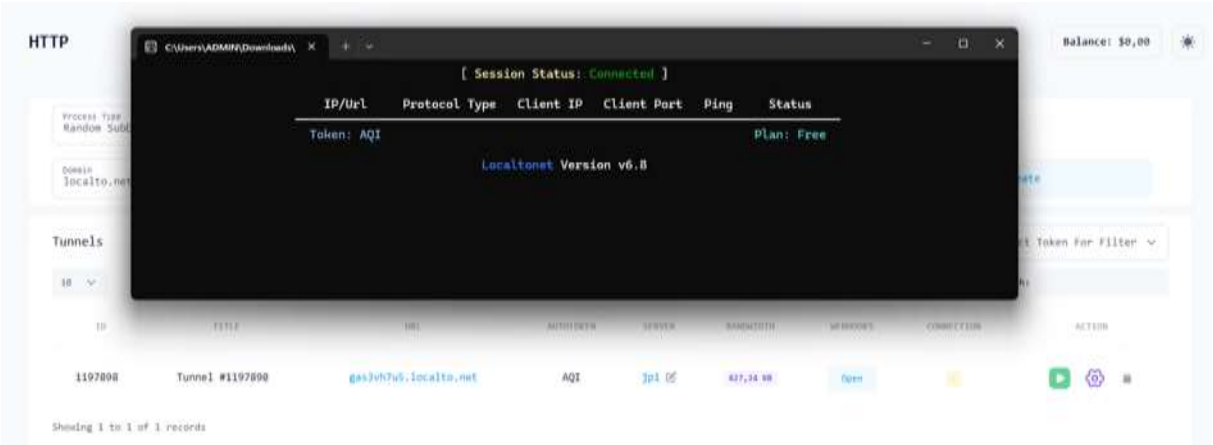
Hình 5.9 Dữ liệu cụ thể

- Người dùng có thể xuất dữ liệu dưới dạng file Excel hoặc văn bản để phục vụ nghiên cứu hoặc lưu trữ
- Biểu đồ và lịch sử dữ liệu: Biểu đồ dữ liệu thời gian thực hiện theo ngày, giờ, phút, có thể lựa chọn xem lại dữ liệu cũ, có thể có thêm so sánh mức độ ô nhiễm giữa các khoảng thời gian và xu hướng chất lượng không khí

Hình 5.10 Biểu đồ lịch sử

- Cung cấp cho người dùng cái nhìn tổng quan và xu hướng của từng chất gây ô nhiễm có thể đưa ra nhận định về chất lượng không khí, bên cạnh đó còn có công cụ để người dùng có thể truy xuất thông tin theo từng khoảng thời gian cụ thể

Hosting và Tên miền: Vì đang trong giai đoạn thử nghiệm nên nhóm chưa sử dụng các công cụ lưu trữ có tốc độ đọc ghi cao mà đang sử dụng trên nền tảng Localtonet để có thể sử dụng và đưa ra đánh giá trước khi đưa vào thực tế.



Hình 5.11 Giao diện theo dõi Localtonet

5.4. Kết quả đạt được

Sau quá trình nghiên cứu, thiết kế và thử nghiệm, hệ thống datalogger quan trắc không khí do nhóm thực hiện đã hoàn thiện ở mức độ mô hình hoạt động thực tế và thu được những kết quả cụ thể sau:

- Hoạt động ổn định: Hệ thống thực hiện thu thập liên tục các thông số môi trường bao gồm PM2.5, PM10, CO, NO_2, nhiệt độ và độ ẩm. Dữ liệu được cập nhật theo thời gian thực và hiển thị đồng thời qua WebApp.
- Lưu trữ dữ liệu hiệu quả: Dữ liệu thu thập được ghi vào cơ sở dữ liệu MySQL với định dạng chuẩn (CSV, JSON), kèm dấu thời gian, thuận tiện cho việc phân tích và truy xuất sau này.
- Tính toán chỉ số AQI: Các giá trị môi trường được xử lý để chuyển đổi thành chỉ số AQI theo quy chuẩn Việt Nam. Chỉ số được thể hiện rõ ràng trên giao diện người dùng.
- Ứng dụng trí tuệ nhân tạo: Mô hình mạng nơ-ron LSTM được huấn luyện trên tập dữ liệu cảm biến nhằm dự đoán xu hướng AQI. Mô hình hoạt động ổn định với sai số nhỏ trong phạm vi chấp nhận được.
- Xây dựng giao diện WebApp: Giao diện trực quan, cho phép người dùng xem dữ liệu hiện tại, truy xuất lịch sử và đánh giá xu hướng biến đổi chất lượng không khí theo thời gian.

5.5. Ưu điểm

- Chi phí thấp, dễ tiếp cận: Sử dụng các thiết bị phần cứng phổ biến như Arduino Uno, Raspberry Pi 5 và cảm biến giá rẻ nhưng đủ độ chính xác cho mục đích quan trắc cơ bản.
- Dễ mở rộng và tùy biến: Hệ thống có thể thêm các loại cảm biến mới hoặc thay đổi mô hình AI mà không cần thay đổi toàn bộ cấu trúc.
- Ứng dụng được trí tuệ nhân tạo (AI): Việc tích hợp mô hình học sâu (LSTM) giúp hệ thống không chỉ giám sát mà còn có khả năng dự đoán và cảnh báo.
- Giao diện thân thiện, dễ truy cập: WebApp được xây dựng cho phép người dùng theo dõi từ xa qua Wi-Fi hoặc Ethernet.
- Linh hoạt trong triển khai: Mô hình có thể ứng dụng tại các khu công nghiệp, phòng thí nghiệm hoặc khu dân cư cần giám sát chất lượng không khí.

5.6. Nhược điểm

- Độ chính xác cảm biến còn hạn chế: Một số cảm biến giá rẻ có thể sai lệch trong môi trường thực tế nếu không được hiệu chuẩn kỹ.
- Chưa có hệ thống cảnh báo tự động: Hệ thống mới dừng lại ở mức giám sát, chưa có chức năng cảnh báo khi AQI vượt ngưỡng nguy hiểm.
- Chưa tối ưu về năng lượng: Hệ thống vẫn phụ thuộc vào nguồn điện trực tiếp, chưa có nguồn dự phòng hoặc năng lượng mặt trời.
- Bảo mật WebApp chưa cao: Giao diện hiện tại chưa có cơ chế đăng nhập hoặc phân quyền người dùng.
- Dữ liệu huấn luyện AI còn hạn chế: Tập dữ liệu còn nhỏ, chưa đủ đa dạng để đảm bảo độ chính xác cao khi triển khai ở nhiều địa điểm.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

a) Kết luận chung

Trong bối cảnh ô nhiễm không khí ngày càng trở thành vấn đề cấp thiết, đặc biệt tại các khu công nghiệp và đô thị hóa nhanh, việc nghiên cứu và phát triển các hệ thống quan trắc chất lượng không khí là yêu cầu mang tính thời sự. Đề tài "Thiết kế Datalogger cho hệ thống quan trắc môi trường không khí" đã hoàn thành các mục tiêu đề ra và thu được những kết quả thực tế đáng khích lệ.

Hệ thống được xây dựng dựa trên nền tảng phần cứng gồm Arduino Uno và Raspberry Pi 5, kết nối với các cảm biến bụi mịn (PMS7003), nhiệt độ - độ ẩm (DHT22), khí CO và NO_2 (DFRobot Fermion). Hệ thống đã thu thập dữ liệu môi trường theo thời gian thực, tính toán chỉ số chất lượng không khí (AQI), và lưu trữ dữ liệu có tổ chức vào cơ sở dữ liệu MySQL. Đặc biệt, mô hình mạng nơ-ron LSTM được huấn luyện để dự đoán xu hướng AQI trong tương lai, mở rộng vai trò của hệ thống từ giám sát sang cảnh báo sớm. Dữ liệu được hiển thị thông qua WebApp với giao diện đơn giản, dễ sử dụng và có thể truy cập từ xa.

Những kết quả và đóng góp nổi bật của đề tài:

- Thiết kế và triển khai thành công một hệ thống datalogger môi trường tích hợp phần cứng và phần mềm, hoạt động ổn định trong điều kiện thử nghiệm.
- Áp dụng trí tuệ nhân tạo (AI) để nâng cao khả năng dự báo và xử lý dữ liệu môi trường.
- Đề xuất mô hình kiến trúc phân tán, tách riêng vai trò thu thập – xử lý sơ bộ – lưu trữ – hiển thị dữ liệu, đảm bảo khả năng mở rộng linh hoạt.
- Góp phần xây dựng giải pháp chi phí thấp, dễ triển khai, phù hợp với quy mô phòng thí nghiệm, hộ dân hoặc cụm công nghiệp nhỏ.

Đề xuất và kiến nghị:

- Hệ thống hiện tại mới được triển khai ở mức thử nghiệm mô hình. Cần được hiệu chuẩn cảm biến kỹ hơn và kiểm nghiệm ngoài thực địa trong điều kiện môi trường khắc nghiệt để đánh giá tính bền vững và chính xác lâu dài.
- Cần phát triển thêm các tính năng như hệ thống cảnh báo tự động (qua email, Telegram) khi AQI vượt ngưỡng nguy hiểm.
- Đề xuất hướng phát triển tiếp theo bao gồm tích hợp nguồn năng lượng tái tạo (như năng lượng mặt trời), hỗ trợ kết nối LoRaWAN để phù hợp với các mô hình phân tán không cần mạng internet.

- Kiến nghị các cơ sở giáo dục và doanh nghiệp nhỏ quan tâm nhiều hơn đến giải pháp IoT môi trường vì tính ứng dụng cao, khả năng thương mại hóa và phù hợp với xu hướng chuyển đổi số ngành tài nguyên – môi trường.

b) Hướng phát triển của đề tài

- Tích hợp cảnh báo thời gian thực: Cảnh báo AQI qua email, Zalo, Telegram hoặc còi/bảng LED khi vượt ngưỡng cho phép.
- Phát triển phiên bản sử dụng năng lượng tái tạo: Thiết kế mạch nguồn sử dụng pin năng lượng mặt trời và UPS mini để đảm bảo hoạt động 24/7.
- Tối ưu thuật toán AI: Thử nghiệm thêm các mô hình khác như GRU, CNN hoặc Hybrid Model, kết hợp LSTM với dữ liệu thời tiết để tăng độ chính xác dự đoán.
- Phát triển hệ thống quản lý người dùng và phân quyền: Nâng cấp WebApp để phân quyền truy cập, thêm tính năng xuất báo cáo tự động.
- Triển khai thực tế dài hạn: Lắp đặt hệ thống ngoài trời với hộp bảo vệ chuẩn IP65 để đánh giá độ bền và khả năng hoạt động liên tục.
- Xây dựng nền tảng đa điểm: Kết nối nhiều datalogger lại thành mạng cảm biến phân tán, quản lý tập trung qua hệ thống SCADA mini hoặc Dashboard tổng hợp.

TÀI LIỆU THAM KHẢO

[1] C. H. Jung, "Time Series Forecasting for Air Quality with Structured and Unstructured Data Using Artificial Neural Networks," *Atmosphere*, vol. 16, no. 3, p. 320, 2023. [Online]. Available: https://www.mdpi.com/2073-4433/16/3/320

[2] J. Duan, Y. Gong, J. Luo and Z. Zhao, "Air-quality prediction based on the ARIMA-CNN-LSTM combination model optimized by dung beetle optimizer," *Scientific Reports*, vol. 13, no. 1, 2023. [Online]. Available: https://www.nature.com/articles/s41598-023-36620-4

[3] StackExchange, "How to select number of hidden layers and number of memory cells in an LSTM?" *AI Stack Exchange*. [Online]. Available: https://ai.stackexchange.com/questions/3156/how-to-select-number-of-hidden-layers-and-number-of-memory-cells-in-an-lstm

[4] W. Zhou, "Algorithms for Hyperparameter Tuning of LSTMs for Time Series Forecasting," *Remote Sensing*, vol. 15, no. 8, p. 2076, 2023. [Online]. Available: https://www.mdpi.com/2072-4292/15/8/2076

[5] AQICN, "Thang đo chỉ số chất lượng không khí và chú giải màu sắc," [Online]. Available: https://aqicn.org/scale/vn/

[6] S. V. Vasanthi and A. Brindha, "Low Cost, Multi Pollutant Sensing System Using Raspberry," *Sustainability*, vol. 13, no. 1, p. 370, 2021. [Online]. Available: https://www.mdpi.com/2071-1050/13/1/370

PHỤ LỤC

PHỤ LỤC A. MÃ CHƯƠNG TRÌNH ARDUINO

1. Chương trình đọc giá trị cảm biến

```
#include <DHT.h>
#include <SoftwareSerial.h>

#define DHTPIN 2
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

SoftwareSerial pmsSerial(10, 11); // TX, RX cho PMS7003

// Cấu trúc lưu dữ liệu PMS7003
struct PMSData {
  uint16_t pm25, pm10;
};
// Hàm tính AQI dựa trên nồng độ và bảng chuẩn EPA
float calculateAQI(float conc, float c_low, float c_high, float i_low, float i_high) {
  return ((i_high - i_low) / (c_high - c_low)) * (conc - c_low) + i_low;
}
// Hàm tính AQI cho PM2.5
float calculatePM25AQI(float pm25) {
  if (pm25 <= 12.0) return calculateAQI(pm25, 0.0, 12.0, 0, 50);
  else if (pm25 <= 35.4) return calculateAQI(pm25, 12.1, 35.4, 51, 100);
  else if (pm25 <= 55.4) return calculateAQI(pm25, 35.5, 55.4, 101, 150);
  else if (pm25 <= 150.4) return calculateAQI(pm25, 55.5, 150.4, 151, 200);
  else if (pm25 <= 250.4) return calculateAQI(pm25, 150.5, 250.4, 201, 300);
  else if (pm25 <= 500.4) return calculateAQI(pm25, 250.5, 500.4, 301, 500);
  else return 500;
}
// Hàm tính AQI cho PM10
float calculatePM10AQI(float pm10) {
  if (pm10 <= 54) return calculateAQI(pm10, 0, 54, 0, 50);
  else if (pm10 <= 154) return calculateAQI(pm10, 55, 154, 51, 100);
  else if (pm10 <= 254) return calculateAQI(pm10, 155, 254, 101, 150);
```

```
    else if (pm10 <= 354) return calculateAQI(pm10, 255, 354, 151, 200);
    else if (pm10 <= 424) return calculateAQI(pm10, 355, 424, 201, 300);
    else if (pm10 <= 604) return calculateAQI(pm10, 425, 604, 301, 500);
    else return 500;
}
// Hàm tính AQI cho CO (8h average)
float calculateCOAQI(float co) {
    if (co <= 4.4) return calculateAQI(co, 0.0, 4.4, 0, 50);
    else if (co <= 9.4) return calculateAQI(co, 4.5, 9.4, 51, 100);
    else if (co <= 12.4) return calculateAQI(co, 9.5, 12.4, 101, 150);
    else if (co <= 15.4) return calculateAQI(co, 12.5, 15.4, 151, 200);
    else if (co <= 30.4) return calculateAQI(co, 15.5, 30.4, 201, 300);
    else if (co <= 50.4) return calculateAQI(co, 30.5, 50.4, 301, 500);
    else return 500;
}
// Hàm tính AQI cho NO2 (1h average)
float calculateNO2AQI(float no2) {
    if (no2 <= 53) return calculateAQI(no2, 0, 53, 0, 50);
    else if (no2 <= 100) return calculateAQI(no2, 54, 100, 51, 100);
    else if (no2 <= 360) return calculateAQI(no2, 101, 360, 101, 150);
    else if (no2 <= 649) return calculateAQI(no2, 361, 649, 151, 200);
    else if (no2 <= 1249) return calculateAQI(no2, 650, 1249, 201, 300);
    else if (no2 <= 2049) return calculateAQI(no2, 1250, 2049, 301, 500);
    else return 500;
}
// Hàm đọc dữ liệu từ PMS7003 với retry
PMSData readPMS7003() {
    PMSData data = {0, 0};
    const int maxRetries = 5;
    int retry = 0;

    while (retry < maxRetries) {
        while (pmsSerial.available()) pmsSerial.read();

        unsigned long startTime = millis();
        while (millis() - startTime < 2000 && pmsSerial.available() < 32) {
            delay(10);
        }
```

```
      if (pmsSerial.available() >= 32) {
        if (pmsSerial.read() == 0x42 && pmsSerial.read() == 0x4d) {
          uint8_t buffer[30];
          if (pmsSerial.readBytes(buffer, 30) == 30) {
            uint16_t sum = 0x42 + 0x4d;
            for (int i = 0; i < 28; i++) {
              sum += buffer[i];
            }
            uint16_t checksum = (buffer[28] << 8) | buffer[29];
            if (sum == checksum) {
              data.pm25 = (buffer[10] << 8) | buffer[11];
              data.pm10 = (buffer[12] << 8) | buffer[13];
              return data;
            }
          }
        }
      }
      retry++;
      delay(100);
    }
    return data;
}
// Hàm đọc DHT22 với retry
bool readDHT(float &temp, float &humi) {
    const int maxRetries = 3;
    for (int i = 0; i < maxRetries; i++) {
      temp = dht.readTemperature();
      humi = dht.readHumidity();
      if (!isnan(temp) && !isnan(humi)) {
        return true;
      }
      delay(2000);
    }
    return false;
}
void setup() {
    Serial.begin(9600);
    pmsSerial.begin(9600);
    dht.begin();
```

```
    delay(30000);
}
void loop() {
    // Đọc dữ liệu từ DHT22
    float temp = 0.0, humi = 0.0;
    if (!readDHT(temp, humi)) {
        temp = -999.0;
        humi = -999.0;
    }
    // Đọc dữ liệu từ PMS7003
    PMSData pms = readPMS7003();
    float pm25 = pms.pm25;
    float pm10 = pms.pm10;
    // Đọc dữ liệu từ DFROBOT CO & NO2 (giả sử kết nối analog)
    int coValue = analogRead(A0);
    int no2Value = analogRead(A1);
    float co_ppm = coValue * (10.0 / 1023.0);
    float no2_ppb = no2Value * (200.0 / 1023.0);
    // Tính AQI tổng thể (giá trị lớn nhất giữa PM2.5, PM10, CO, NO2)
    float aqi_pm25 = pm25 > 0 ? calculatePM25AQI(pm25) : -999.0;
    float aqi_pm10 = pm10 > 0 ? calculatePM10AQI(pm10) : -999.0;
    float aqi_co = coValue > 0 ? calculateCOAQI(co_ppm) : -999.0;
    float aqi_no2 = no2Value > 0 ? calculateNO2AQI(no2_ppb) : -999.0;
    float aqi = -999.0;
    if (aqi_pm25 != -999.0 || aqi_pm10 != -999.0 || aqi_co != -999.0 || aqi_no2 != -999.0) {
        aqi = max(max(aqi_pm25, aqi_pm10), max(aqi_co, aqi_no2));
    }
    // Hiển thị dữ liệu theo định dạng yêu cầu
    Serial.print("VN_AQI giờ: "); Serial.print(aqi); Serial.print(", ");
    Serial.print("Nhiệt độ: "); Serial.print(temp); Serial.print(", ");
    Serial.print("Độ ẩm: "); Serial.print(humi); Serial.print(", ");
    Serial.print("CO: "); Serial.print(co_ppm); Serial.print(", ");
    Serial.print("NO2: "); Serial.print(no2_ppb); Serial.print(", ");
    Serial.print("PM-10: "); Serial.print(pm10); Serial.print(", ");
    Serial.print("PM-2.5: "); Serial.println(pm25);
    delay(5000);
}
```

PHỤ LỤC B. MÃ CHƯƠNG TRÌNH PYTHON

1. Chương trình mô hình mạng nơ-ron và dự báo

```
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import Dataset, DataLoader
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sqlalchemy import create_engine
import matplotlib.pyplot as plt
import os
import csv
from datetime import datetime, timedelta
import random
from openpyxl.styles import Font

def set_seed(seed=42):
    random.seed(seed)
    np.random.seed(seed)
    torch.manual_seed(seed)
    torch.cuda.manual_seed_all(seed)
    torch.backends.cudnn.deterministic = True
    torch.backends.cudnn.benchmark = False
# Mô hình LSTM
class AirQualityLSTM(nn.Module):
    def __init__(self, input_size=9, hidden_size=64, num_layers=2, output_size=24):
        super(AirQualityLSTM, self).__init__()
        self.lstm = nn.LSTM(input_size, hidden_size, num_layers, batch_first=True)
        self.fc = nn.Linear(hidden_size, output_size)
    def forward(self, x):
        out, _ = self.lstm(x)
        out = out[:, -1, :]
        out = self.fc(out)
        return out
# Dataset từ chuỗi
class SequenceAQIDataset(Dataset):
    def __init__(self, sequences, labels):
        self.sequences = sequences
        self.labels = labels
```

```
    def __len__(self):
        return len(self.sequences)
    def __getitem__(self, idx):
        return (
            torch.tensor(self.sequences[idx], dtype=torch.float32),
            torch.tensor(self.labels[idx], dtype=torch.float32)
        )
# Nhãn cảnh báo từ trung bình AQI
def classify_aqi_level(mean_aqi):
    if mean_aqi <= 50:
        return "Chất lượng không khí tốt, không ảnh hưởng tới sức khỏe"
    elif mean_aqi <= 100:
        return "Chất lượng không khí khá tốt, ảnh hưởng nhẹ đến sức khỏe trẻ em"
    elif mean_aqi <= 150:
         return "Chất lượng không khí ở mức chấp nhận được. Tuy nhiên, đối với những người nhạy cảm (Người già, trẻ em, người mắc các bện hô hấp tim mạnh...) có thể chịu những tác động nhất định tới sức khỏe"
    elif mean_aqi <= 200:
         return "Những người bình thường bắt đầu có các ảnh hưởng tới sức khỏe, nhóm người nhạy cảm có thể gặp những vấn đề về sức khỏe nghiêm trọng hơn"
    elif mean_aqi <= 300:
        return "Chất lượng không khí xấu, ảnh hưởng tới sức khỏe của tất cả mọi người"
    else:
        return "Nguy hại"
# Đọc dữ liệu và tạo chuỗi
def read_data_from_mysql_sequence():
    engine = create_engine("mysql+pymysql://tuanminh:06062002@localhost/sensor_data")
    df = pd.read_sql("SELECT * FROM environmental_data", engine)
    df['Ngày giờ'] = pd.to_datetime(df['Ngày giờ'])
    df = df.sort_values(by='Ngày giờ')
    df['Năm'] = df['Ngày giờ'].dt.year
    df['Tháng'] = df['Ngày giờ'].dt.month
    df['Ngày'] = df['Ngày giờ'].dt.day
    df['Giờ'] = df['Ngày giờ'].dt.hour
    feature_cols = ['VN_AQI giờ', 'CO', 'NO2', 'PM-10', 'PM-2.5', 'Năm', 'Tháng', 'Ngày', 'Giờ']
    df[feature_cols] = df[feature_cols].apply(pd.to_numeric, errors='coerce').fillna(0)
    # Chuẩn hóa
    scaler_features = MinMaxScaler()
    scaler_aqi = MinMaxScaler()
```

```
    scaler_aqi.fit(df[['VN_AQI giờ']])  # Fit riêng AQI gốc
    df[feature_cols] = scaler_features.fit_transform(df[feature_cols])
    sequence_length = 24
    input_sequences, output_sequences = [], []
    for i in range(len(df) - 2 * sequence_length):
        input_seq = df[feature_cols].iloc[i:i + sequence_length].values
        output_seq = df['VN_AQI giờ'].iloc[i + sequence_length:i + 2 * sequence_length].values
        input_sequences.append(input_seq)
        output_sequences.append(output_seq)
    # Giải nén dữ liệu về giá trị gốc chưa chuẩn hóa
    df_copy = df.copy()
    df_copy[['VN_AQI giờ']] = scaler_aqi.inverse_transform(df[['VN_AQI giờ']])
    df_copy[feature_cols] = scaler_features.inverse_transform(df[feature_cols])
    df_copy = df_copy.drop(columns=["Năm", "Tháng", "Ngày", "Giờ"])
    df_copy = df_copy.round(0).astype(int)
    df_copy['Ngày giờ'] = pd.to_datetime(df['Ngày giờ'])
    # Sắp xếp theo thời gian nếu cần
    df_copy.sort_values(by="Ngày giờ", inplace=True)
    # Loại bỏ các dòng trùng nhau theo 'Ngày giờ'
    df_copy = df_copy.drop_duplicates(subset=["Ngày giờ"], keep="first")
    # Đặt lại thứ tự cột
    cols = ['Ngày giờ'] + [col for col in df_copy.columns if col != 'Ngày giờ']
    df_copy = df_copy[cols]
    # Lưu file Excel
    df_copy.to_excel("du_lieu_goc.xlsx", index=False)
    return np.array(input_sequences), np.array(output_sequences), scaler_aqi
# Đánh giá
def evaluate_model(model, test_loader):
    model.eval()
    predictions, actuals = [], []
    with torch.no_grad():
        for x, y in test_loader:
            y_pred = model(x)
            predictions.extend(y_pred.numpy())
            actuals.extend(y.numpy())
    predictions = np.array(predictions)
    actuals = np.array(actuals)
    mse = mean_squared_error(actuals, predictions)
    mae = mean_absolute_error(actuals, predictions)
```

```
    print(f"\nĐánh giá mô hình:\nMSE: {mse:.4f}, MAE: {mae:.4f}")
    return mse, mae
# Huấn luyện
def train(model, train_loader, test_loader, optimizer, criterion, epochs=40):
    log_file = "training_log.csv"
    if not os.path.exists(log_file):
        with open(log_file, 'w', newline='') as f:
            writer = csv.writer(f)
            writer.writerow(["Epoch", "Loss", "MAE", "MSE"])
    mae_list, mse_list = [], []
    for epoch in range(epochs):
        model.train()
        total_loss = 0
        for x, y in train_loader:
            optimizer.zero_grad()
            y_pred = model(x)
            loss = criterion(y_pred, y)
            loss.backward()
            optimizer.step()
            total_loss += loss.item()
        mse, mae = evaluate_model(model, test_loader)
        mae_list.append(mae)
        mse_list.append(mse)
        with open(log_file, 'a', newline='') as f:
            writer = csv.writer(f)
            writer.writerow([epoch+1, total_loss / len(train_loader), mae, mse])
        print(f'Epoch {epoch+1}/{epochs} - Loss: {total_loss/len(train_loader):.4f}")
    plt.figure()
    plt.plot(mae_list, label="MAE")
    plt.plot(mse_list, label="MSE")
    plt.xlabel("Epoch")
    plt.ylabel("Error")
    plt.legend()
    plt.savefig("metrics.png")
    plt.close()
# Dự đoán và phân loại
def predict_sequence(model, input_seq):
    model.eval()
    input_tensor = torch.tensor(input_seq, dtype=torch.float32).unsqueeze(0)
```

```
    with torch.no_grad():
        prediction = model(input_tensor).view(-1).numpy()
    mean_val = prediction.mean()
    label = classify_aqi_level(mean_val)
    return prediction, label
# Dự đoán trên tập kiểm tra và so sánh với giá trị thực tế
def predict_on_test_set(model, test_loader, scaler):
    model.eval()
    predictions, actuals = [], []

    with torch.no_grad():
        for x, y in test_loader:
            y_pred = model(x)
            predictions.extend(y_pred.numpy())
            actuals.extend(y.numpy())
    # Đưa từ dạng chuẩn hóa về giá trị AQI gốc
    predictions = scaler.inverse_transform(np.array(predictions))
    actuals = scaler.inverse_transform(np.array(actuals))
    return predictions, actuals
def plot_prediction_vs_actual(predictions, actuals, day_index=0,
output_path="comparison_prediction_actual.png"):
    plt.figure(figsize=(10, 5))
    plt.plot(actuals[day_index], label='Giá trị thực tế', marker='o')
    plt.plot(predictions[day_index], label='Giá trị dự báo', marker='x')
    plt.title(f"So sánh AQI ngày test thứ {day_index+1}")
    plt.xlabel("Giờ trong ngày")
    plt.ylabel("Chỉ số AQI")
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.savefig(output_path)
    plt.show()
    print(f"Biểu đồ đã lưu vào {output_path}")
# MAIN
if __name__ == "__main__":
    set_seed(42)
    X, y, scaler_aqi = read_data_from_mysql_sequence()
    dataset = SequenceAQIDataset(X, y)
```

```
    train_size = int(0.8 * len(dataset))
    test_size = len(dataset) - train_size
    train_dataset, test_dataset = torch.utils.data.random_split(dataset, [train_size, test_size])
    train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
    test_loader = DataLoader(test_dataset, batch_size=32, shuffle=False)

    model = AirQualityLSTM()
    optimizer = optim.Adam(model.parameters(), lr=0.001)
    criterion = nn.MSELoss()

    print("Bắt đầu huấn luyện...")
    train(model, train_loader, test_loader, optimizer, criterion, epochs=30)

    torch.save(model.state_dict(), "air_quality_forecast_24h.pth")
    print("Đã lưu mô hình.")

    latest_input = X[-1]
    prediction_scaled, label = predict_sequence(model, latest_input)
        prediction_original = scaler_aqi.inverse_transform(prediction_scaled.reshape(-1,
1)).flatten()

    now = datetime.now().replace(minute=0, second=0, microsecond=0)
    forecast_start = (now + timedelta(days=1)).replace(hour=0)

    times = []
    aqi_values = []
    hourly_labels = []

    print("\nDự báo AQI 24 giờ tiếp theo:")
    for i, val in enumerate(prediction_original):
        forecast_time = forecast_start + timedelta(hours=i)
        label_hour = classify_aqi_level(val)
          print(f"{forecast_time.strftime('%Y-%m-%d %H:%M')} - AQI: {val:.2f} - Cảnh báo:
{label_hour}")
        times.append(forecast_time)
        aqi_values.append(round(val))
        hourly_labels.append(label_hour)

    print(f"\nCảnh báo chất lượng không khí tổng thể ngày mai: {label}")
```

```
# Lưu Excel
df_result = pd.DataFrame({
    "Thời gian": times,
    "AQI Dự đoán": aqi_values,
    "Cảnh báo": hourly_labels
})

output_file_excel = "du_doan_24h_ngay_mai.xlsx"
with pd.ExcelWriter(output_file_excel, engine='openpyxl') as writer:
    df_result.to_excel(writer, sheet_name='Dự báo', index=False)

    workbook = writer.book
    worksheet = writer.sheets['Dự báo']
    font = Font(name='Times New Roman', size=13)
    for row in worksheet.iter_rows():
        for cell in row:
            cell.font = font
print(f"\nĐã lưu kết quả dự báo 24h vào '{output_file_excel}'")

print("\n== Đánh giá cuối cùng trên tập kiểm tra (test set) ==")
final_mse, final_mae = evaluate_model(model, test_loader)
print(f"MSE Test Set: {final_mse:.4f}, MAE Test Set: {final_mae:.4f}")

# Dự đoán & vẽ biểu đồ so sánh cho 1 ngày bất kỳ (ở đây là ngày đầu tiên trong test set)
preds, trues = predict_on_test_set(model, test_loader, scaler_aqi)
plot_prediction_vs_actual(preds, trues, day_index=0)
```

2. Chương trình tạo Webapp

```
import pandas as pd
from flask import Flask, render_template, request, jsonify
import webbrowser
import threading
import os

app = Flask(__name__)
# Định nghĩa hàm đọc dữ liệu cụ thể
def read_excel_data():
    file2 = os.path.abspath("D:/DATN/du_lieu_goc.xlsx")
    df = pd.read_excel(file2, engine="openpyxl")
```

```
    df.fillna("", inplace=True)
    df.drop_duplicates(inplace=True)
    return df
# Hàm load cả 2 file (cho trang chính)
def load_data():
    file1 = os.path.abspath("D:/DATN/du_doan_24h_ngay_mai.xlsx")
    file2 = os.path.abspath("D:/DATN/du_lieu_goc.xlsx")
    df1 = pd.read_excel(file1, engine="openpyxl")
    df2 = pd.read_excel(file2, engine="openpyxl")
    df1.fillna("", inplace=True)
    df2.fillna("", inplace=True)
    df2.drop_duplicates(inplace=True)
    return df1, df2
@app.route('/')
def home():
    df1, df2 = load_data()
    return render_template(
        "index.html",
        data=df1.to_dict(orient="records"),
        du_doan=df1.to_dict(orient="records"),
        lich_su=df2.to_dict(orient="records")
    )
@app.route('/lich_su')
def lich_su():
    start = request.args.get("start")
    end = request.args.get("end")
    df = read_excel_data(start, end)
    return jsonify(df.to_dict(orient="records"))
def read_excel_data(start_date=None, end_date=None):
    file2 = os.path.abspath("D:/DATN/du_lieu_goc.xlsx")
    df = pd.read_excel(file2, engine="openpyxl")
    df.fillna("", inplace=True)
    df.drop_duplicates(inplace=True)
    return df
def open_browser():
    webbrowser.open("http://127.0.0.1:5000/")
if __name__ == "__main__":
    threading.Thread(target=open_browser).start()
    app.run(debug=True, use_reloader=False)
```

3. Chương trình đọc dữ liệu thu thập được trên arduino bằng raspberry

```
import serial
import time

# Cấu hình cổng Serial (điều chỉnh cho đúng cổng Arduino của bạn)
serial_port = '/dev/ttyUSB0'  # Thay đổi nếu Arduino dùng cổng khác
baud_rate = 9600              # Phải trùng baudrate của Arduino

# Khởi tạo kết nối Serial
ser = serial.Serial(serial_port, baud_rate, timeout=1)

def read_serial_data():
    while True:
        if ser.in_waiting > 0:
            line = ser.readline().decode('utf-8').strip()
            if line:
                # Tách chuỗi thành các cặp key-value
                data_pairs = line.split(', ')
                data = {}
                for pair in data_pairs:
                    if ': ' in pair:
                        key, value = pair.split(': ')
                        try:
                            data[key] = float(value)
                        except ValueError:
                            data[key] = value

                # Lấy dữ liệu và in ra
                aqi = data.get('VN_AQI giờ', 'N/A')
                temp = data.get('Nhiệt độ', 'N/A')
                humi = data.get('Độ ẩm', 'N/A')
                co = data.get('CO', 'N/A')
                no2 = data.get('NO2', 'N/A')
                pm10 = data.get('PM-10', 'N/A')
                pm25 = data.get('PM-2.5', 'N/A')

                print(f"VN_AQI giờ: {aqi}, Nhiệt độ: {temp}, Độ ẩm: {humi}, CO: {co}, NO2: {no2}, PM-10: {pm10}, PM-2.5: {pm25}")
```

```
        time.sleep(3600)  # Đọc mỗi giờ

try:
    print(f"Đã kết nối {serial_port} ở tốc độ {baud_rate} baud. Đang đọc dữ liệu...")
    read_serial_data()
except KeyboardInterrupt:
    print("\nDừng bởi người dùng.")
    print(f"Lỗi kết nối Serial: {e}")
finally:
    ser.close()
    print("Đã đóng kết nối Serial.")

except serial.SerialException as e:
```

4. Chương trình đưa dữ liệu đọc được từ raspberry lên SQL

```
import serial
import time

# Cấu hình cổng Serial (điều chỉnh cho đúng cổng Arduino của bạn)
serial_port = '/dev/ttyUSB0'  # Thay đổi nếu Arduino dùng cổng khác
baud_rate = 9600              # Phải trùng baudrate của Arduino

# Khởi tạo kết nối Serial
ser = serial.Serial(serial_port, baud_rate, timeout=1)

def read_serial_data():
    while True:
        if ser.in_waiting > 0:
            line = ser.readline().decode('utf-8').strip()
            if line:
                # Tách chuỗi thành các cặp key-value
                data_pairs = line.split(', ')
                data = {}
                for pair in data_pairs:
                    if ': ' in pair:
                        key, value = pair.split(': ')
                        try:
                            data[key] = float(value)
                        except ValueError:
```

```
                data[key] = value

        # Lấy dữ liệu và in ra
        aqi = data.get('VN_AQI giờ', 'N/A')
        temp = data.get('Nhiệt độ', 'N/A')
        humi = data.get('Độ ẩm', 'N/A')
        co = data.get('CO', 'N/A')
        no2 = data.get('NO2', 'N/A')
        pm10 = data.get('PM-10', 'N/A')
        pm25 = data.get('PM-2.5', 'N/A')

          print(f"VN_AQI giờ: {aqi}, Nhiệt độ: {temp}, Độ ẩm: {humi}, CO: {co}, NO2:
{no2}, PM-10: {pm10}, PM-2.5: {pm25}")

    time.sleep(3600)  # Đọc mỗi giờ

try:
  print(f"Đã kết nối {serial_port} ở tốc độ {baud_rate} baud. Đang đọc dữ liệu...")
  read_serial_data()

except serial.SerialException as e:
  print(f"Lỗi kết nối Serial: {e}")

except KeyboardInterrupt:
  print("\nDừng bởi người dùng.")

finally:
  ser.close()
  print("Đã đóng kết nối Serial.")
```