

**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**ĐỒ ÁN TỐT NGHIỆP**  
**Thiết kế chế tạo hệ đo và giám sát nồng độ**  
**bụi mịn PM10, PM2.5**

**PHAN NHẬT TÂN**

Tan.pn153310@sis.hust.edu.vn

**Ngành Kỹ thuật Điều khiển và Tự động hóa**  
**Chuyên ngành Kỹ thuật đo và Tin học công nghiệp**

**Giảng viên hướng dẫn:** PGS. TS. Bùi Đăng Thành

Chữ ký của GVHD

**Bộ môn:**

Kỹ thuật đo và Tin học công nghiệp

**Viện:**

Điện

**HÀ NỘI, 1/2020**



--\*\*\*--

## NHIỆM VỤ THIẾT KẾ TỐT NGHIỆP

Họ và tên: Phan Nhật Tân

Mã số sinh viên: 20153310

Khóa: 60

Viện: Điện

Ngành: Kỹ thuật đo và tin học công nghiệp

### 1. Đầu đề thiết kế/Tên đề tài

#### Thiết kế chế tạo hệ đo và giám sát nồng độ bụi mịn PM10, PM2.5

### 2. Các số liệu ban đầu

- Thiết kế phần cứng hệ cho phép đo được các thông số về :
  - Nồng độ bụi mịn PM10, PM2.5 : 0-999.9  $\mu\text{g}/\text{m}^3$
  - Độ ẩm không khí : 0 – 100%
  - Nhiệt độ không khí : 0 – 50 °C
- Phần cứng thiết kế dựa trên vi điều khiển
- Có khả năng hiển thị kết quả trên Web

### 3. Các nội dung tính toán, thiết kế

- Tìm hiểu về bụi và các nguyên nhân gây ra bụi và các phương pháp đo nồng độ bụi
- Tìm hiểu ảnh hưởng của bụi đối với sức khỏe con người
- Thiết kế thiết bị đo có tích hợp vi điều khiển
- Thiết kế một Web server có chức năng nhận và lưu trữ dữ liệu từ các thiết bị đo
- Thiết kế giao diện người dùng hiển thị các thông số đo

4. Cán bộ hướng dẫn : PGS.TS. Bùi Đăng Thành

5. Ngày giao nhiệm vụ thiết kế : Ngày 1 tháng 9 năm 2019

6. Ngày hoàn thành nhiệm vụ : Ngày 05 tháng 1 năm 2020



*Ngày ..... tháng 01 năm 2020*

**CHỦ NHIỆM BỘ MÔN**

*(Ký, ghi rõ họ tên)*

**CÁN BỘ HƯỚNG DẪN**

*(Ký, ghi rõ họ tên)*

**SINH VIÊN THỰC HIỆN**

*(Ký, ghi rõ họ tên)*



## LỜI CẢM ƠN

Đầu tiên em xin gửi lời cảm ơn chân thành tới PGS.TS. Bùi Đăng Thành, người đã hướng dẫn em tận tình trong suốt quá trình học tập cũng như thực hiện đồ án này. Em cũng xin cảm ơn các thầy cô đang công tác và giảng dạy tại trường Đại học Bách Khoa Hà Nội, các thầy cô trong viện Điện tử chung và viện Điều khiển & Tự động hóa nói riêng đã tạo cho chúng em một môi trường học tập và nghiên cứu năng động, giúp em trang bị những kiến thức cần thiết trong suốt thời gian ngồi trên ghế nhà trường. Em xin gửi lời biết ơn sâu sắc tới gia đình và bạn bè đã luôn tạo điều kiện, quan tâm, giúp đỡ và động viên em trong suốt gần năm năm học. Do thời gian và kiến thức có hạn nên không thể tránh khỏi những thiếu sót nhất định. Em rất mong nhận được những ý kiến đóng góp quý báu của thầy cô. Thời điểm năm mới sắp đến, em xin gửi lời chúc sức khỏe, hạnh phúc tới thầy cô, gia đình và bạn bè.

Em xin chân thành cảm ơn!

## TÓM TẮT NỘI DUNG ĐỒ ÁN

Trong đồ án tốt nghiệp này, em tập trung thiết kế và chế tạo một hệ đo và giám sát bụi mịn PM10, PM2.5. Hiện nay tình trạng ô nhiễm không khí ở Việt Nam nói chung và Hà Nội nói riêng đang luôn ở tình trạng báo động. Tuy nhiên việc theo dõi, giám sát nồng độ bụi chưa thực sự đáp ứng được do sự thiếu hụt về số lượng các hệ thống quan trắc cũng như sự xuống cấp về chất lượng của các hệ thống này. Các trạm quan trắc thường được sử dụng trong thời gian dài, có thể là cả thập kỷ, hầu hết đây là các trạm quan trắc lớn, và rất khó để di chuyển đến một vị trí khác. Chính vì vậy cần thiết kế và chế tạo một hệ đo và giám sát nồng độ bụi trong không khí theo thời gian thực, sử dụng các công nghệ cảm biến mới có kích thước nhỏ gọn, với chi phí chế tạo, lắp đặt và vận hành thấp. Kết quả đo đạc có thể được sử dụng để tính toán cũng như đánh giá chất lượng không khí từ đó có thể đưa ra các khuyến cáo. Ở trong đồ án này em đã nghiên cứu thiết kế thành công một hệ đo và giám sát nồng độ bụi PM10, PM2.5 sử dụng cảm biến đo nồng độ bụi PM10, PM2.5 là SDS011, tích hợp cảm biến DHT22 đo nhiệt độ và độ ẩm không khí. Các dữ liệu được xử lý trên vi điều khiển STM32F103C8T6, hiển thị trên màn hình LCD và truyền thông với Webserver thông qua giao tiếp không dây GPRS của module SIM808. Một giao diện Web được thiết kế giúp hiển thị các kết quả đo một cách trực quan, thân thiện người dùng.

Sinh viên thực hiện



## MỤC LỤC

<b>LỜI NÓI ĐẦU .....</b>	<b>1</b>
<b>CHƯƠNG 1. GIỚI THIỆU CHUNG .....</b>	<b>2</b>
1.1    Bụi và tác hại của bụi .....	2
1.1.1    Khái niệm về bụi .....	2
1.1.2    Các tác nhân gây ra bụi .....	3
1.1.3    Các tiêu chuẩn đánh giá nồng độ bụi .....	6
1.1.4    Ảnh hưởng của bụi tới môi trường và sức khỏe con người .....	7
1.2    Tình hình ô nhiễm bụi ở Việt Nam và trên thế giới.....	8
1.2.1    Tình hình ô nhiễm bụi trên thế giới .....	8
1.2.2    Tình hình ô nhiễm không khí ở Việt Nam .....	9
<b>CHƯƠNG 2. THIẾT KẾ HỆ ĐO VÀ GIÁM SÁT NỒNG ĐỘ BỤI.....</b>	<b>11</b>
2.1    Cấu trúc hệ thống .....	11
2.2    Yêu cầu thiết kế .....	13
<b>CHƯƠNG 3. THIẾT KẾ THIẾT BỊ ĐO VÀ GIÁM SÁT NỒNG ĐỘ BỤI.</b>	<b>14</b>
3.1    Các phương pháp đo nồng độ bụi .....	14
3.1.1    Phương pháp cân trọng lượng .....	14
3.1.2    Phương pháp suy giảm Beta .....	14
3.1.3    Phương pháp thác va chạm .....	15
3.1.4    Phương pháp tán xạ laser .....	16
3.2    Yêu cầu thiết kế cho thiết bị.....	19
3.3    Sơ đồ khối thiết bị .....	19
3.4    Thiết kế phần cứng.....	20
3.4.1    Vi điều khiển STM32F103C8T6 .....	20
3.4.2    Cảm biến đo nồng độ bụi SDS011.....	22
3.4.3    Cảm biến đo nhiệt độ độ ẩm DHT22 .....	24
3.4.4    Màn hình LCD NOKIA N5110 .....	25
3.4.1    Khối SIM808.....	27
3.4.2    Khối nguồn.....	27
3.4.3    Mạch nguyên lý và mạch in .....	30
3.5    Thiết kế phần mềm.....	37
3.5.1    Lưu đồ thuật toán .....	37
3.5.2    Giao tiếp với cảm biến SDS011 .....	37

3.5.3	Giao tiếp với cảm biến DHT22.....	38
3.5.4	Giao tiếp với màn hình LCD NOKIA N5110 .....	40
3.5.5	Giao tiếp với khối SIM808 .....	44
<b>CHƯƠNG 4. THIẾT KẾ WEB SERVER VÀ GIAO DIỆN NGƯỜI DÙNG</b>		<b>46</b>
4.1	Web Server.....	46
4.2	Giới thiệu về REST Server .....	46
4.2.1	API .....	46
4.2.2	Chuỗi Json.....	47
4.2.3	Giới thiệu về RESTful API.....	47
4.2.4	Cấu trúc của REST server.....	48
4.3	Giới thiệu về Node.js, Express, ReactJS.....	48
4.3.1	Node.js .....	49
4.3.2	Express.....	49
4.3.3	ReactJS.....	50
4.4	Thiết kế các RESTful API .....	51
4.4.1	RESTful API cho giao diện người dùng.....	51
4.4.2	RESTful API cho các Sensor Node (RESTful client) .....	53
4.5	Triển khai giao diện người dùng.....	53
<b>CHƯƠNG 5. KẾT QUẢ VÀ ĐÁNH GIÁ HỆ THỐNG</b>		<b>55</b>
5.1	Kết quả đạt được .....	55
5.1.1	Mạch phần cứng thiết bị đo nồng độ bụi .....	55
5.1.2	Web server .....	55
5.2	Thử nghiệm và đánh giá .....	56
5.2.1	Thử nghiệm .....	56
5.2.2	Đánh giá và thảo luận .....	57
<b>KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN ĐỀ TÀI</b>		<b>58</b>
<b>TÀI LIỆU THAM KHẢO</b>		<b>59</b>
<b>PHỤ LỤC</b>		<b>60</b>

## DANH MỤC HÌNH ẢNH

Hình 1.1: Kích thước các loại hạt bụi .....	2
Hình 1.2: Đường phố Hà Nội với lượng phương tiện giao thông đông đúc .....	3
Hình 1.3: Hoạt động đốt rơm rạ ở các vùng ven ngoại ô Hà Nội .....	3
Hình 1.4: Các cột khói bụi từ các nhà máy công nghiệp .....	4
Hình 1.5: Hiện tượng nghịch nhiệt.....	4
Hình 1.6: Một núi lửa phun trào tạo nên cột khói khổng lồ.....	5
Hình 1.7: Biểu đồ phân bố sự thải bụi PM2.5 của các hoạt động.....	5
Hình 1.8: Đường xâm nhập của bụi mịn vào cơ thể con người .....	7
Hình 1.9: Bản đồ ô nhiễm bụi toàn thế giới .....	8
Hình 1.10: Nồng độ PM2.5 được mô hình hóa cho năm 2015 .....	9
Hình 1.11: Ảnh chụp Hà Nội từ cầu Vĩnh Tuy ngày 15/12/2019 .....	10
Hình 2.1: Cấu trúc hệ thống đo và giám sát nồng độ bụi.....	11
Hình 2.2: Sơ đồ trình tự hoạt động của hệ thống .....	12
Hình 2.3: Context Diagram .....	12
Hình 3.1: Phương pháp cân trọng lượng .....	14
Hình 3.2: Phương pháp suy giảm Beta .....	15
Hình 3.3: Phương pháp thác va chạm .....	16
Hình 3.4: Phương pháp tán xạ laser .....	17
Hình 3.5: Dữ liệu thô từ cảm biến theo điện áp .....	17
Hình 3.6: Cảm biến đọc giá trị LPO .....	18
Hình 3.7: Mối liên hệ giữa LPO và kích thước hạt.....	18
Hình 3.8: Sơ đồ khái thiết bị .....	19
Hình 3.9: STM32F103C8T6 .....	20
Hình 3.10: Sơ đồ chân của STM32F103C8T6.....	21
Hình 3.11: Cảm biến SDS011 .....	22
Hình 3.12: Bên trong cảm biến SDS011 .....	23
Hình 3.13: Mô tả dòng khí trong cảm biến SDS011.....	23
Hình 3.14: Cấu trúc cảm biến nhiệt độ, độ ẩm DHT22 .....	24
Hình 3.15: Màn hình LCD NOKIA N5110 .....	25
Hình 3.16: Sơ đồ chân của LCD N5110 .....	26
Hình 3.17: SIM808.....	27
Hình 3.18: Adapter 12V, 2A .....	28
Hình 3.20: LM2576.....	28
Hình 3.20: AMS1117-3.3.....	29
Hình 3.21: Diode .....	29
Hình 3.22: Đầu vào nguồn adapter .....	30

Hình 3.23: Mạch nguồn 5VDC .....	30
Hình 3.24: Mạch nguồn 3.3VDC.....	30
Hình 3.25: Mạch nguồn 4.3VDC.....	31
Hình 3.27: Khối Reset.....	31
Hình 3.27: Khối tạo dao động.....	31
Hình 3.28: Khối nạp chương trình .....	31
Hình 3.29: 2 chân Bootloader .....	32
Hình 3.30: Kết nối chân của DHT22 .....	32
Hình 3.31: Kết nối chân của SDS011 .....	33
Hình 3.32: Kết nối chân của LCDN5110 .....	33
Hình 3.33: Mạch SIM808 .....	33
Hình 3.34: Simcard .....	34
Hình 3.35: Amtena.....	34
Hình 3.36: Led báo SIM808 .....	34
Hình 3.37: Layer tổng quan .....	35
Hình 3.38: Top layer .....	35
Hình 3.39: Bottom layer .....	36
Hình 3.40: Mô hình 3D .....	36
Hình 3.41: Sơ đồ khối chương trình chính .....	37
Hình 3.42: Tín hiệu Start trong giao tiếp với cảm biến DHT22 .....	39
Hình 3.43: Bit dữ liệu 0 của DHT22 .....	40
Hình 3.44: Bit dữ liệu 1 của DHT22 .....	40
Hình 3.45: Chế độ truyền 1 byte.....	41
Hình 3.46: Chế độ truyền nhiều byte .....	41
Hình 3.47: Giản đồ xung chân RST.....	41
Hình 3.48: Các bit trong chân DC .....	42
Hình 3.49: Cấu hình cài đặt hiển thị .....	42
Hình 3.50: Cấu hình Set địa chỉ dòng .....	42
Hình 3.51: Giá trị các bit set địa chỉ dòng .....	43
Hình 3.52: Cấu hình set địa chỉ cột.....	43
Hình 3.53: Cấu hình set điện áp hoạt động .....	43
Hình 3.54: Các bước khởi tạo LCD N5110 .....	44
Hình 4.1: Cách thức API hoạt động.....	47
Hình 4.2: Hệ thống giám sát chất lượng không khí dựa trên RESTful .....	48
Hình 4.3: Nhận dữ liệu hiện tại của một vị trí theo ID .....	51
Hình 4.4: Nhận dữ liệu của một ID trong một khoảng thời gian.....	52
Hình 4.5: Nhận tất cả dữ liệu của một vị trí theo ID .....	53

Hình 4.6: Dữ liệu được gửi từ Sensor Node theo ID .....	53
Hình 4.7: Giao diện hiển thị.....	54
Hình 5.1: Mạch phần cứng thiết bị đo.....	55
Hình 5.2: Giao diện hiển thị Web .....	56
Hình 5.3: Kết quả đo tại tòa nhà HiTech .....	56
Hình 5.4: Diễn biến nồng độ bụi trên ứng dụng PAM AIR.....	57

## **DANH MỤC BẢNG BIỂU**

Bảng 1.1: Giá trị tối hạn của nồng độ bụi PM10 và PM2.5 ở Việt Nam ( $\mu\text{g}/\text{m}^3$ ) .	6
Bảng 1.2: Bảng quy đổi AQI .....	6
Bảng 3.1: Thông số kỹ thuật của vi điều khiển STM32F103C8T6.....	21
Bảng 3.2: Thông số kỹ thuật cảm biến SDS011 .....	22
Bảng 3.3: Chân cảm biến SDS011.....	24
Bảng 3.4: Thông số kỹ thuật của cảm biến DHT22 .....	25
Bảng 3.5: Thông số về điện của cảm biến DHT22.....	25
Bảng 3.6: Mô tả sơ đồ chân DHT22 .....	25
Bảng 3.7: Thông số kỹ thuật màn hình LCD NOKIA N5110 .....	26
Bảng 3.8: Chức năng các chân của LCD N5110 .....	26
Bảng 3.9: Thông số kỹ thuật của SIM808 .....	27
Bảng 3.10: Thông số kỹ thuật cơ bản của LM2576.....	28
Bảng 3.11: Sơ đồ chân LM2576 .....	28
Bảng 3.12: Thông số kỹ thuật cơ bản của AMS117 .....	29
Bảng 3.13: Sơ đồ chân AMS117 .....	29
Bảng 3.14: Kết nối chân mạch nạp và thiết bị .....	32
Bảng 3.15: Chuỗi giá trị truyền thông UART cảm biến SDS011 .....	38
Bảng 3.16: Ý nghĩa các bit D, E .....	42

## **DANH MỤC VIẾT TẮT**

<b>Tên viết tắt</b>	<b>Tên tiếng Anh</b>	<b>Tên tiếng Việt</b>
PM	Particulate Matter	Hạt vật chất
AQI	Air Quality Index	Chỉ số chất lượng không khí
MCU	Micro controller unit	Vi điều khiển trung tâm
LCD	Liquid Crystal Display	Màn hình tinh thể lỏng
UI	User interface	Giao diện người dùng
API	Application Programming Interface	Giao diện lập trình ứng dụng



## LỜI NÓI ĐẦU

Hiện nay ô nhiễm bụi và những tác động của nó đến môi trường cũng như sức khỏe con người đang là một mối lo ngại lớn ở các nước đang phát triển như Việt Nam. Tại những thành phố lớn như Hà Nội, Hồ Chí Minh và những thành phố công nghiệp của nước ta mức độ ô nhiễm những năm gần đây luôn đạt mức đáng báo động. Số ca mắc các bệnh liên quan đến ô nhiễm bụi như các bệnh về đường hô hấp, tai mũi họng ngày càng nhiều. Tuy nhiên việc theo dõi, giám sát nồng độ bụi chưa thực sự đáp ứng được với tình hình hiện nay, do sự thiếu hụt về số lượng các hệ thống quan trắc và sự xuống cấp về chất lượng của các hệ thống này.

Chi phí là một rào cản rất lớn trong việc mở rộng giám sát nồng độ bụi trong không khí, giá cho một trạm giám sát nồng độ bụi chuyên dụng có thể lên đến hàng chục nghìn đô la. Các trạm quan trắc thường được sử dụng trong thời gian dài, có thể là cả thập kỷ, hầu hết đây là các trạm quan trắc lớn, và rất khó để di chuyển đến một vị trí khác. Điều này gây khó khăn cho việc thu thập dữ liệu ở quy mô lớn. Chỉ với một số lượng nhỏ các trạm giám sát được lắp đặt thì lượng dữ liệu thu thập được sẽ bị giới hạn.

Sự xuất hiện của các cảm biến chi phí thấp hơn, kết hợp với “Internet of Thing” (IoT), có thể cho phép chúng ta thay đổi cách theo dõi nồng độ bụi trong khí bằng cách sử dụng những thiết bị giám sát nhỏ, có thể kết nối với nhau thông qua giao tiếp không dây. Với giải pháp này chúng ta có thể tạo ra một hệ thống giám sát, bán cố định hoặc di động, dễ cài đặt và có thể sẵn sàng để bắt đầu thu thập dữ liệu một cách nhanh chóng.

Với định hướng đó, em quyết định thực hiện đề tài: “*Thiết kế chế tạo hệ đo và giám sát nồng độ bụi mịn PM10, PM2.5*”, đặt ra mục tiêu thiết kế một hệ thống có thể đo và giám sát nồng độ bụi mịn PM10, PM2.5 trong không khí theo thời gian thực, sử dụng các công nghệ cảm biến mới có kích thước nhỏ gọn, với chi phí chế tạo, lắp đặt và vận hành thấp. Kết quả đo đạc có thể được sử dụng để tính toán cũng như đánh giá chất lượng không khí.

Đồ án tốt nghiệp này được trình bày gồm các phần chính sau đây:

- *Chương 1: Giới thiệu chung*
- *Chương 2: Thiết kế hệ đo và giám sát nồng độ bụi*
- *Chương 3: Thiết kế thiết bị đo và giám sát nồng độ bụi*
- *Chương 4: Thiết kế Webserver và giao diện người dùng*
- *Chương 5: Kết quả và đánh giá hệ thống*
- *Kết luận và hướng phát triển*

Do kiến thức còn hạn chế nên trong quá trình nghiên cứu, thiết kế không thể tránh khỏi những thiếu sót. Em hy vọng sẽ nhận được ý kiến đóng góp và phản biện từ hội đồng chuyên môn để có thể hoàn thiện thiết bị hơn trong tương lai.

Em xin chân thành cảm ơn!

# CHƯƠNG 1. GIỚI THIỆU CHUNG

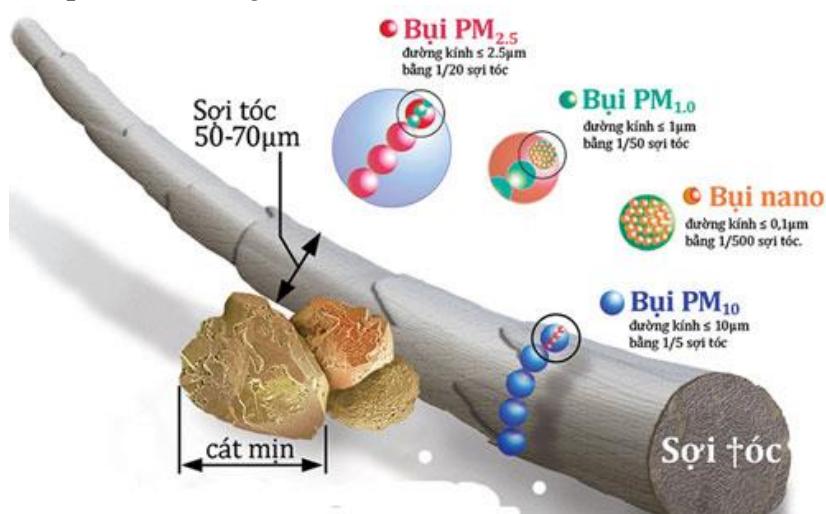
## 1.1 Bụi và tác hại của bụi

### 1.1.1 Khái niệm về bụi

Bụi là một hỗn hợp phức tạp chứa các hạt vô cơ và hữu cơ ở dạng lỏng hoặc rắn bay lơ lửng trong không khí; bao gồm sulfate, nitrat, amoniac, natri clorua, cacbon đen, bụi khoáng và nước. Bụi hay hợp chất có trong bụi được gọi chung là Particulate Matter - ký hiệu PM.

Bụi có thể phân thành bốn loại dựa trên kích thước hạt:

- PM10: là các hạt mịn, rất mịn và siêu mịn có đường kính nhỏ hơn hoặc bằng  $10\mu\text{m}$ . Nguồn phát thải PM10 có thể là bụi đường, khói từ các nhà máy công nghiệp...
- PM2.5: là các hạt mịn có đường kính nhỏ hơn hoặc bằng  $2.5\mu\text{m}$  được tìm thấy trong khói và sương mù. Nguồn phát thải PM2.5 có thể là khói từ các vụ cháy rừng, khí thải từ các nhà máy điện, các ngành công nghiệp và ô tô, khí thải từ động cơ diesel của các phương tiện giao thông.
- PM1: là các hạt rất mịn (nguy hiểm nhất đối với sức khỏe) có đường kính nhỏ hơn hoặc bằng  $1\mu\text{m}$ . Chúng hầu như bị loại bỏ bởi khói không khí khi có mưa. Tuy nhiên chúng vẫn có thời gian tích lũy khá dài trong khí quyển.
- PM0.1: là các hạt siêu mịn có đường kính nhỏ hơn  $0.1\mu\text{m}$ , còn được gọi là "hạt nano". Thời gian cư trú của hạt này trong khí quyển rất ngắn, dao động từ vài phút đến vài giờ.



Hình 1.1: Kích thước các loại hạt bụi

PM2.5 và PM1 có thể đi sâu vào phần sâu nhất (phế nang) của phổi, nơi trao đổi khí xảy ra giữa không khí và máu. Đây là những hạt nguy hiểm nhất vì phần phế nang của phổi không có phương tiện hiệu quả để loại bỏ chúng và nếu các hạt tan trong nước, chúng có thể đi vào máu trong vòng vài phút. Nếu chúng không hòa tan trong nước, chúng vẫn tồn tại trong phổi phế nang trong một thời gian dài. Các nguyên tố hòa tan có thể là PAHs (hydrocacbon thơm đa vòng) hoặc dư lượng của benzen được phân loại là chất gây ung thư.

### 1.1.2 Các tác nhân gây ra bụi

Có nhiều nguyên nhân gây ra bụi khác nhau, tiêu biểu là:

a) Khói bụi từ các phương tiện giao thông



Hình 1.2: Đường phố Hà Nội với lượng phương tiện giao thông đông đúc

Khói bụi từ các phương tiện giao thông trên đường là nguyên nhân đầu tiên gây ra bụi mịn. Các loại xe cộ xả ra ngoài môi trường các hạt sooty và oxit nitơ... góp phần làm ô nhiễm không khí. Tình trạng ô nhiễm bụi mịn trong thành phố cao hơn vùng ngoại ô phần lớn là do lượng phương tiện đi lại lớn hơn rất nhiều.

b) Hoạt động đốt rơm rạ, đốt rác, dùng than củi



Hình 1.3: Hoạt động đốt rơm rạ ở các vùng ven ngoại ô Hà Nội

Khi đốt rơm rạ, đốt rác vô tổ chức, con người cũng đang giải phóng ra môi trường những loại khí độc hại, các hạt bụi mịn sinh ra từ những đống tro tàn này sẽ được gió khuếch tán trong không khí. Đây là nguyên nhân xuất phát từ sự thiếu hiếu biết và ý thức kém của người dân.

c) Khói bụi từ nhà máy, khu công nghiệp, công trình



Hình 1.4: Các cột khói bụi từ các nhà máy công nghiệp

Thành phố càng phát triển, các công trình xây dựng, các khu công nghiệp, nhà máy càng mọc lên không ngừng. Các loại vật liệu như xi măng, đất, cát, phế liệu, khí thải... đều góp phần không nhỏ làm tăng tỷ lệ bụi, gây ô nhiễm trong thành phố.

d) *Hiện tượng nghịch nhiệt, mù quang hóa*



Hình 1.5: Hiện tượng nghịch nhiệt

Khi thời tiết xuất hiện hình thái nghịch nhiệt bức xạ, hiện tượng đảo nhiệt sẽ được tăng cường, cùng với độ ẩm trong không khí sẽ khiến cho các chất bụi bẩn, ô nhiễm trong không khí bị tích tụ và ngưng kết lại tạo thành hiện tượng mù quang hóa. Hiện tượng này càng khiến cho chất lượng không khí trở nên nghiêm trọng hơn và ảnh hưởng xấu đến sức khỏe người dân.

e) *Các hiện tượng tự nhiên khác*

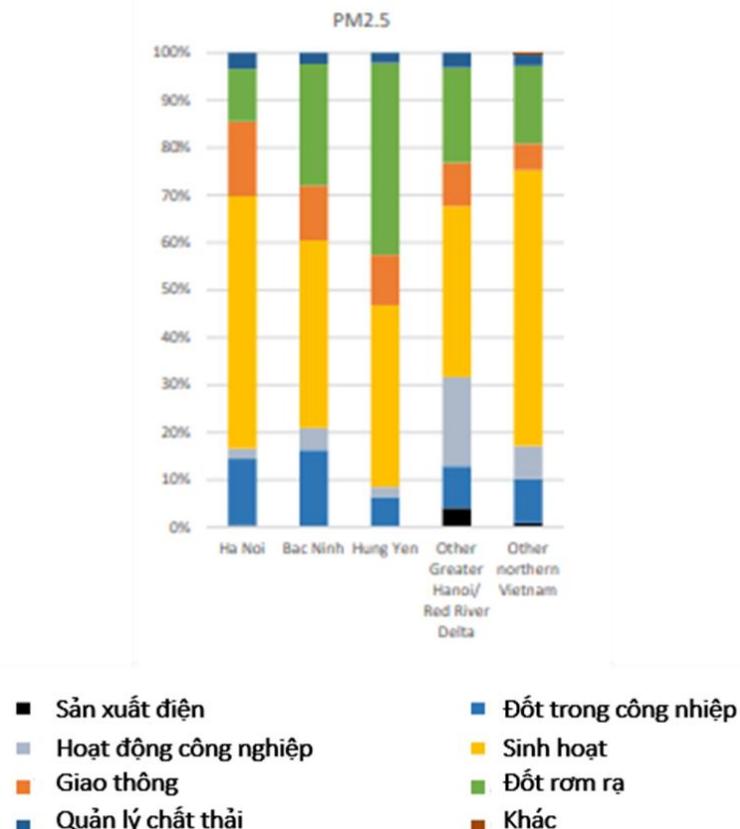


*Hình 1.6: Một núi lửa phun trào tạo nên cột khói khổng lồ*

Các hiện tượng tự nhiên khác cũng góp phần tạo nên lượng bụi mịn trong không khí bao gồm sự hoạt động của núi lửa, giông, gió, lốc xoáy, bão bụi, cháy rừng... Bụi mịn cũng có thể đến từ những hạt phấn hoa hay những bào tử thực vật trong quá trình sinh trưởng.

Ở Việt Nam, tác nhân chủ yếu gây ra bụi thường là khí thải từ giao thông, sinh hoạt hằng ngày và hoạt động công nghiệp.

Hình dưới đây mô tả sự đóng góp tương đối của các hoạt động khác nhau vào PM2.5 trong năm khu vực miền bắc Việt Nam, ước tính năm 2015. [1]



*Hình 1.7: Biểu đồ phân bố sự thải bụi PM2.5 của các hoạt động*

### 1.1.3 Các tiêu chuẩn đánh giá nồng độ bụi

#### a) Tiêu chuẩn Việt Nam:

Theo quy chuẩn Việt Nam thông số về bụi PM10, bụi PM2,5 với các giá trị giới hạn được ghi trong bảng 1.1. (QCVN 06: 2009/BTNMT) [2]

Bảng 1.1: Giá trị tối hạn của nồng độ bụi PM10 và PM2.5 ở Việt Nam ( $\mu\text{g}/\text{m}^3$ )

TT	Thông số	Trung bình 1 giờ	Trung bình 8 giờ	Trung bình 24 giờ	Trung bình năm
1	Bụi PM10	-	-	150	50
2	Bụi PM2,5	-	-	50	25

#### b) Tiêu chuẩn của Hoa Kỳ:

Ở Hoa Kỳ ngoài bảng giới hạn các thông số, còn có bảng chỉ số chất lượng không khí (air quality index AQI) dùng để đánh giá các mức độ ô nhiễm không khí xung quanh. [3]

Công thức tính toán AQI:

$$I = \frac{I_{high} - I_{low}}{C_{high} - C_{low}} (C - C_{low}) + I_{low} \quad PT 1.1$$

Trong đó:

**I** là chỉ số chất lượng không khí

**C** là nồng độ chất ô nhiễm

**C\_low, C\_high** là nồng độ chất ô nhiễm ở 2 đầu của khoảng đang xét

**I\_low, I\_high** là chỉ số chất lượng không khí ở 2 đầu của khoảng đang xét

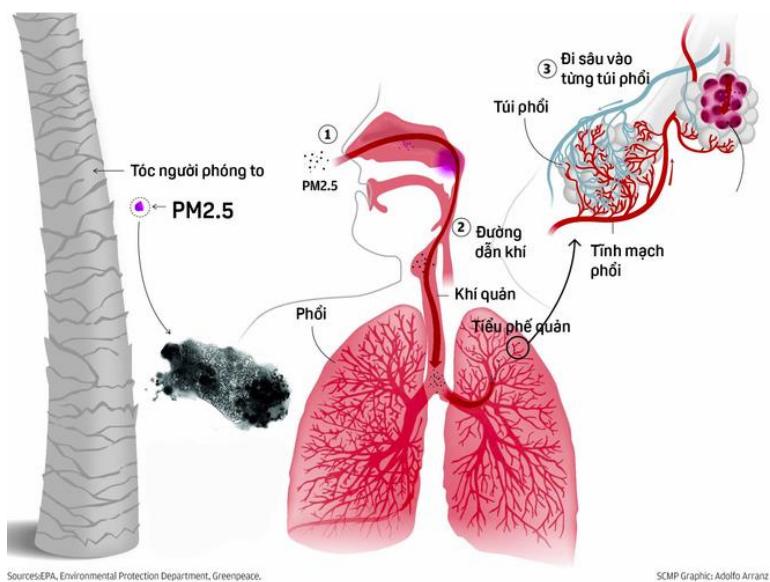
Các thông số C\_low, C\_high, I\_low, I\_high được lấy dựa vào bảng quy đổi

Bảng 1.2: Bảng quy đổi AQI

PM2.5 ( $\mu\text{g}/\text{m}^3$ )	PM10 ( $\mu\text{g}/\text{m}^3$ )	AQI	Mức độ ô nhiễm
$C_{low} - C_{high}$	$C_{low} - C_{high}$	$I_{low} - I_{high}$	
0.0-12.0(24 tiếng)	0-54 (24 tiếng)	0-50	Tốt
12.1-35.4 (24 tiếng)	55-154 (24 tiếng)	51-100	Vừa phải
35.5-55.4 (24 tiếng)	155-254 (24 tiếng)	101-150	Không tốt cho nhóm người nhạy cảm
55.5-150.4 (24 tiếng)	255-354 (24 tiếng)	151-200	Không tốt
150.5-250.4 (24 tiếng)	355-424 (24 tiếng)	201-300	Rất không tốt
250.5-350.4 (24 tiếng)	425-504 (24 tiếng)	301-400	Nguy hiểm
350.5-500.4 (24 tiếng)	505-604 (24 tiếng)	401-500	Cực nguy hiểm

#### 1.1.4 Ảnh hưởng của bụi tới môi trường và sức khỏe con người

Theo nghiên cứu của Tổ chức Y tế thế giới WHO và Cơ quan nghiên cứu ung thư quốc tế IARC đã cho thấy mối tương quan tỷ lệ thuận giữa mức độ ô nhiễm bụi không khí với tỷ lệ người mắc ung thư. Cụ thể là mật độ PM10 trong không khí tăng lên  $10 \mu\text{g}/\text{m}^3$  thì tỷ lệ ung thư tăng 22%, và mật độ PM2.5 tăng thêm  $10 \mu\text{g}/\text{m}^3$  thì tỷ lệ ung thư phổi tăng đến 36%.



Hình 1.8: Đường xâm nhập của bụi mịn vào cơ thể con người

PM2.5 và PM10 đi vào đường hô hấp khi con người hít thở, nhưng mức độ xâm nhập khác nhau tùy theo kích thước hạt bụi. Trong khi PM10 đi vào cơ thể qua đường dẫn khí và tích tụ trên phổi, thì PM2.5 đặc biệt nguy hiểm hơn vì chúng bé đến mức có thể luồn lách vào các túi phổi, tinh mạch phổi và xâm nhập vào hệ tuần hoàn máu.

PM2.5 và PM10 tích tụ lâu ngày sẽ làm tăng nguy cơ phát bệnh ở hệ hô hấp, hệ tim mạch, hệ tuần hoàn và cả hệ sinh sản của con người, các chuyên gia đã đưa ra những cảnh báo sau:

- PM2.5 là nguyên nhân gây nhiễm độc máu, máu khó đông khiến hệ tuần hoàn bị ảnh hưởng, làm suy nhược hệ thần kinh điều khiển hoạt động của cơ tim gây ra các bệnh tim mạch.
- Những hạt bụi mịn xâm nhập vào cơ thể, làm giảm chức năng của phổi, viêm phế quản mãn tính, gây nên bệnh hen suyễn và ung thư phổi. Đồng thời, khiến tình trạng bệnh trở nặng hơn và có thể tử vong.
- Chúng là nguyên nhân gây nhiễm độc máu nhau thai, khiến thai nhi chậm phát triển. Trẻ sinh ra bị ít cân, nhiều khả năng bị suy nhược thần kinh và tự kỷ.
- Ngoài ra, các chuyên gia của Cơ quan bảo vệ Môi sinh Mỹ (EPA) nhận định, hạt PM2.5 chứa nhiều kim loại nặng có khả năng gây ung thư, hoặc tác động đến DNA và gây ra đột biến gen.

- Uớc tính có đến 4,3 triệu người chết mỗi năm do các bệnh liên quan đến ô nhiễm bụi mịn PM2.5 và PM10.

Nhóm đối tượng nhạy cảm và chịu ảnh hưởng nhiều nhất của ô nhiễm bụi mịn PM2.5 và PM10 đó là trẻ em, người già, phụ nữ có thai, những người có bệnh tim hoặc các vấn đề về hô hấp. Trẻ nhỏ sống ở những nơi ô nhiễm không khí nặng khó phát triển chiều cao toàn diện và có nguy cơ mắc bệnh hô hấp cao hơn từ 19 - 25% so với bình thường. [4]

## 1.2 Tình hình ô nhiễm bụi ở Việt Nam và trên thế giới

### 1.2.1 Tình hình ô nhiễm bụi trên thế giới

Theo thông tin từ Tổ chức Y tế thế giới WHO, ô nhiễm bụi gây ra cái chết sớm cho khoảng 4,2 triệu người trên thế giới vào năm 2016. Trong đó, 91% tỉ lệ thuộc về các nước nghèo và đông dân ở Đông Nam Á và Tây Thái Bình Dương.

Bob O'Keefe, Phó Chủ tịch WHO chia sẻ: "Ô nhiễm bụi thực sự là một cú sốc lớn cho toàn cầu. Vấn nạn này khiến những người mắc bệnh hô hấp thêm khó thở, trẻ con và người già phải vào viện, bỏ học, bỏ việc và gây ra những cái chết sớm cho con người".

Health Effects Institute (HEI) vừa đưa ra phát hiện mới nhất trong báo cáo thường niên 2018, dựa trên dữ liệu vệ tinh và được quy chiếu với các tiêu chuẩn trong Hướng dẫn đánh giá chất lượng không khí của WHO.

HEI cho biết, hơn 95% dân số thế giới đang phải hít thở bầu không khí ô nhiễm và có đến 60% người sống ở những khu vực không đáp ứng được tiêu chuẩn cơ bản nhất của WHO. Theo đó, ô nhiễm môi trường không khí là nguyên nhân gây tử vong cao thứ tư thế giới, chỉ đứng sau cao huyết áp, suy dinh dưỡng và hút thuốc lá.

Hình dưới đây được trích xuất từ trang web waqi.info theo dõi mức độ ô nhiễm trực tuyến trên toàn thế giới. [5]



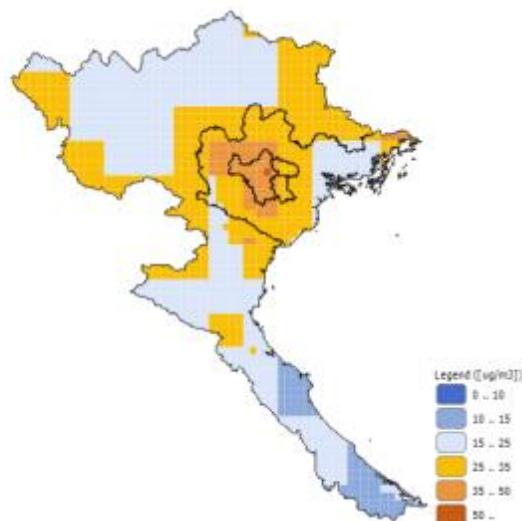
Hình 1.9: Bản đồ ô nhiễm bụi toàn thế giới

Trung Quốc và Ấn Độ là hai quốc gia đứng đầu danh sách ô nhiễm môi trường, chiếm 50% số ca tử vong do ô nhiễm không khí trên toàn cầu. Riêng tại Trung Quốc đã ghi nhận 1,1 triệu người chết vì ô nhiễm không khí trong năm 2016.

Mặc cho nhận thức về môi trường sống và ô nhiễm không khí ngày càng được cải thiện tại các đô thị lớn, tình hình vẫn ngày càng trầm trọng hơn khi 2/3 thế giới đang phải hứng chịu nạn ô nhiễm khủng khiếp với chỉ số hạt bụi PM2.5 cao trên mức  $35 \mu\text{g}/\text{m}^3$  khí, chủ yếu tại Châu Á, Trung Đông và Châu Phi. Nguyên nhân là do dân số những khu vực này tăng quá nhanh, khiến các nỗ lực giảm thiểu ô nhiễm không khí chỉ như muối bỏ bể.

### 1.2.2 Tình hình ô nhiễm không khí ở Việt Nam

Hình 1.10 là mô hình hóa nồng độ bụi PM2.5 trong năm 2015[2]



Hình 1.10: Nồng độ PM2.5 được mô hình hóa cho năm 2015

Theo báo cáo thường niên The Environmental Performance Index (EPI) của Mỹ thực hiện, Việt Nam hiện đang đứng trong top 10 các nước ô nhiễm không khí ở Châu Á. Đáng lưu ý, tổng lượng bụi ở Hà Nội và TP Hồ Chí Minh đang liên tục tăng cao khiến chỉ số chất lượng không khí (AQI) luôn ở mức báo động.

Năm 2016, GreenID công bố báo cáo Sơ lược tình trạng môi trường Hà Nội và TP.HCM:

- Hà Nội: chỉ số AQI trung bình là 121, nồng độ bụi PM2.5 là  $50.5 \mu\text{g}/\text{m}^3$  gấp đôi quy chuẩn quốc gia ( $25 \mu\text{g}/\text{m}^3$ ) và gấp năm lần khuyến nghị từ WHO ( $10 \mu\text{g}/\text{m}^3$ ).
- TP.HCM: chỉ số AQI trung bình là 86, nồng độ bụi PM2.5 là  $28.3 \mu\text{g}/\text{m}^3$  cao hơn so với quy chuẩn quốc gia và gấp ba lần khuyến nghị từ WHO.

Nồng độ bụi trung bình trong không khí ở Hà Nội và TP.HCM vượt mức cho phép từ hai đến ba lần và có xu hướng duy trì ở ngưỡng cao. Nguồn sinh ra bụi ô nhiễm ở các đô thị lớn hầu hết là từ khí thải giao thông, công trình xây dựng, đường sá và nhà máy công nghiệp. Hà Nội chỉ đứng sau New Delhi, Ấn Độ ( $124 \mu\text{g}/\text{m}^3$ ), nơi ô nhiễm không khí nặng nhất nhì thế giới.



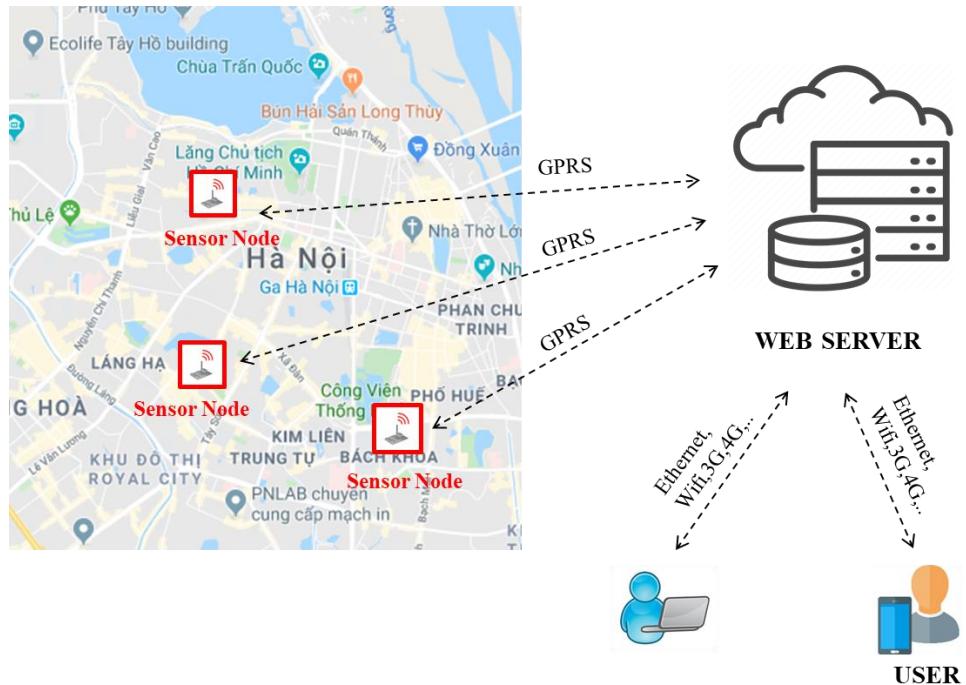
*Hình 1.11: Ảnh chụp Hà Nội từ cầu Vĩnh Tuy ngày 15/12/2019*

Đối với vùng nông thôn, nhìn chung chất lượng môi trường không khí còn khá tốt. Môi trường chủ yếu bị tác động cục bộ bởi các hoạt động sản xuất làng nghề, xây dựng, đốt rơm rạ, đốt rác thải, đun nấu, v.v.

## CHƯƠNG 2. THIẾT KẾ HỆ ĐO VÀ GIÁM SÁT NỒNG ĐỘ BỤI

### 2.1 Cấu trúc hệ thống

Sơ đồ mô tả hệ thống đo và giám sát nồng độ bụi PM10, PM2.5 thể hiện trên Hình 2.1.



Hình 2.1: Cấu trúc hệ thống đo và giám sát nồng độ bụi

Hệ thống được xây dựng với các thành phần bao gồm:

- Sensor Node
- Webserver (Webserver)

#### Sensor Node

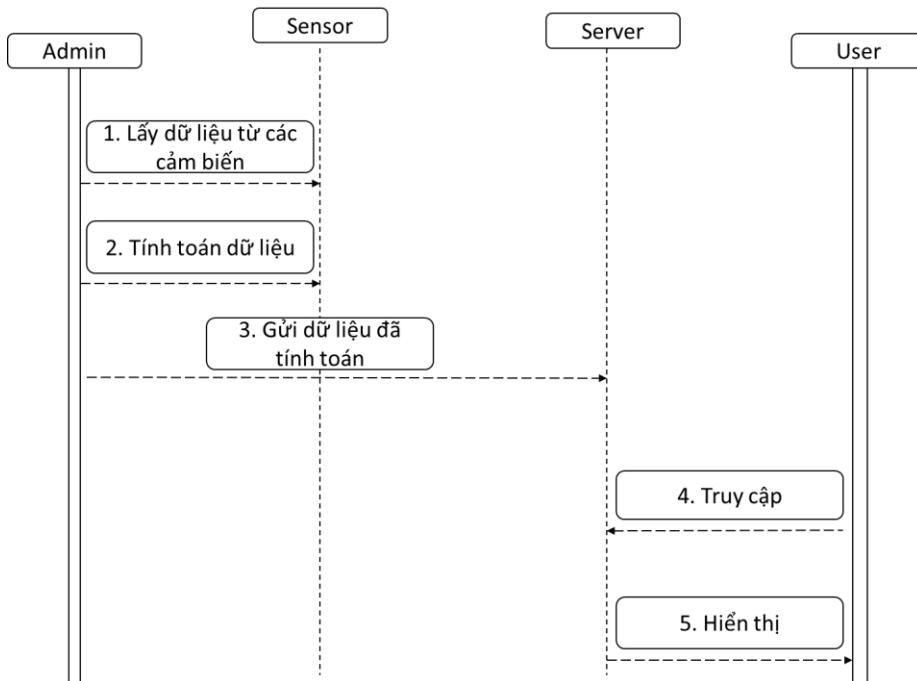
Sensor Node là các thiết bị đo nồng độ bụi PM10, PM2.5, ngoài ra chúng cũng được tích hợp thêm cảm biến nhiệt độ và độ ẩm không khí. Thông tin về các thông số này sẽ được hiển thị trên LCD và truyền qua mạng không dây GPRS tới Webserver. Mô tả chi tiết về thiết kế thiết bị đo và giám sát nồng độ bụi được trình bày trong Chương 3.

#### Web Server

Việc ứng dụng Web Server trong hệ thống này mang lại khả năng lưu trữ dữ liệu trực tuyến cũng như thực hiện công việc giám sát hệ thống từ xa. Với khả năng tương thích của giao diện Web, chỉ với một thiết bị có hỗ trợ trình duyệt Web và được kết nối Internet (như Smart Phone, Tablet, laptop...) người dùng có thể theo dõi toàn bộ hệ thống. Phát triển phần mềm trên Web Server cho phép thu thập, xử lý số liệu và hiển thị thông tin sẽ được trình bày trong Chương 4. Người dùng (User) có thể truy cập vào Web server để nhận các thông tin về nồng độ bụi.

## Sự tương tác giữa hệ thống với các thành phần

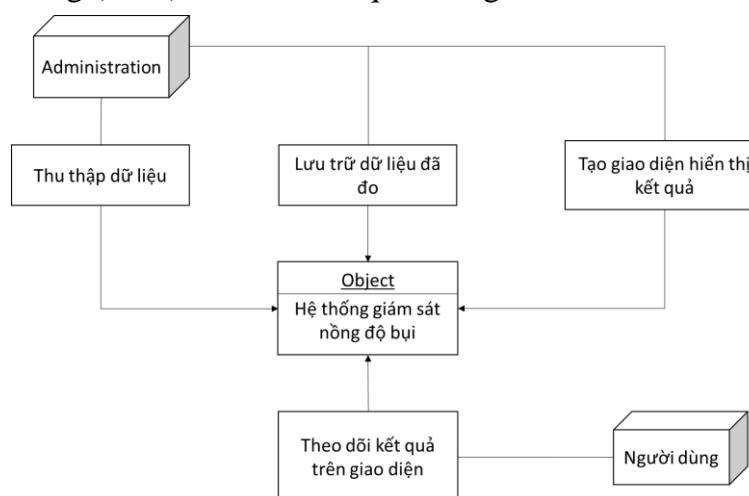
Hình 2.2 mô tả sự tương tác theo trình tự thời gian giữa các thành phần trong hệ thống. Người quản lý (Admin) thu thập dữ liệu từ cảm biến, sau đó sẽ thực hiện xử lý dữ liệu và gửi dữ liệu đã tính toán đến server. Thực chất việc này được giao cho vi điều khiển thực hiện. Người dùng sẽ truy cập vào server thông qua giao diện hiển thị để xem và lấy dữ liệu.



Hình 2.2: Sơ đồ trình tự hoạt động của hệ thống

Hình 2.3 mô tả tổng quan về môi trường mà hệ thống sẽ vận hành, sự tương tác giữa hệ thống và người quản lý, người sử dụng.

- Người quản lý (Admin) thu thập dữ liệu từ cảm biến, lưu trữ dữ liệu và tạo giao diện hiển thị.
- Người dùng (User) theo dõi kết quả trên giao diện đã được tạo.



Hình 2.3: Context Diagram

## 2.2 Yêu cầu thiết kế

Một hệ đo và giám sát nồng độ bụi trên thực tế cần đáp ứng những yêu cầu cơ bản sau:

- Là một hệ thống đo, giám sát vận hành ổn định.
- Hiển thị kết quả đo trên màn hình LCD và giao diện Web.
- Truyền thông kết quả lên Webserver thông qua kết nối không dây để hiện thị và lưu trữ dữ liệu.
- Đơn giản, tiện dụng cho người sử dụng.

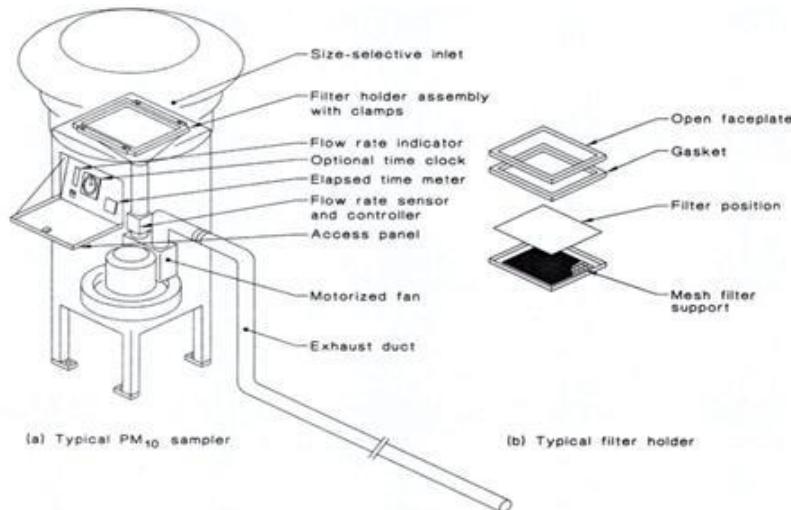
Một hệ thống đo, giám sát vận hành ổn định nghĩa là một hệ thống vận hành liên tục trong thời gian dài. Kết quả đo sẽ được hiển thị trực tiếp trên thiết bị đo thông qua màn hình LCD, đồng thời cũng được hiển thị trên giao diện Web. Việc này giúp cho người giám sát thuận lợi trong việc theo dõi, lưu trữ, báo cáo số liệu. Dữ liệu của hệ thống được truyền thông qua giao thức kết nối không dây. Các thiết bị cũng như giao diện người dùng cần đơn giản, thân thiện với người sử dụng.

Ngoài ra chi phí đầu tư cho hệ thống cần giảm thiểu tối đa. Chi tiết về các thành phần trong hệ thống sẽ được trình bày ở các chương tiếp theo trong đồ án.

## CHƯƠNG 3. THIẾT KẾ THIẾT BỊ ĐO VÀ GIÁM SÁT NỒNG ĐỘ BỤI

### 3.1 Các phương pháp đo nồng độ bụi

#### 3.1.1 Phương pháp cân trọng lượng



Hình 3.1: Phương pháp cân trọng lượng

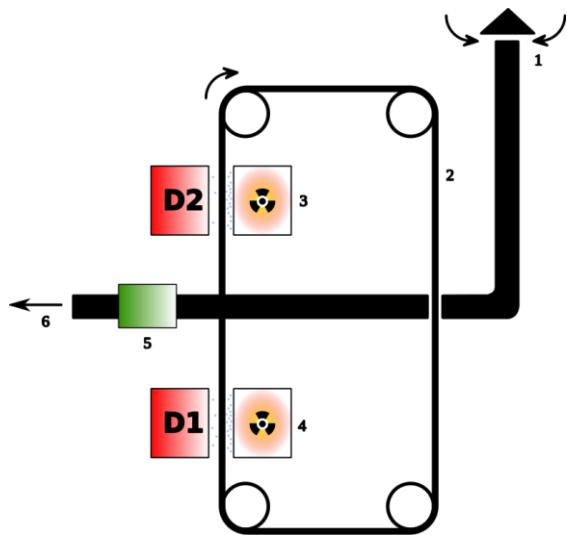
Phương pháp này hiểu đơn giản là không khí sẽ được lấy mẫu vào thiết bị và đi qua các miếng lọc. Các hạt có kích thước khác nhau sẽ được giữ lại ở các miếng lọc khác nhau. Sau đó người ta sẽ phân tích trọng lượng để xác định lượng hạt bụi lǎng đọng trong một thời gian nhất định (thường là 24 giờ).

- **Ưu điểm:**
  - Đơn giản, dễ sử dụng
  - Độ chính xác cao
- **Nhược điểm:**
  - Kích thước thiết bị đo lớn
  - Phải thực hiện các phép đo thủ công hàng ngày
  - Kết quả đo chỉ là  $\text{mg}/\text{m}^3$

#### 3.1.2 Phương pháp suy giảm Beta

Thiết bị giám sát suy giảm Beta (Beta Attenuation Monitors), hay còn gọi là BAM. Đây là thiết bị chuyên dụng, và cũng là thiết bị chính thức duy nhất được sử dụng để theo dõi chất lượng không khí ở Trung Quốc, Mỹ và hầu hết các nước thế giới.

Nguyên tắc làm việc BAM: Nó đo lường sự giảm số lượng hạt Beta (electron) truyền qua một lớp bụi mỏng (PM). Khi độ dày của lớp PM tăng lên, số lượng các hạt beta có thể đi qua càng thấp.



Hình 3.2: Phương pháp suy giảm Beta

Ký hiệu:

1 - Đầu vào không khí; 2 - Băng lấy mẫu; 3, 4 - Nguồn bức xạ beta;

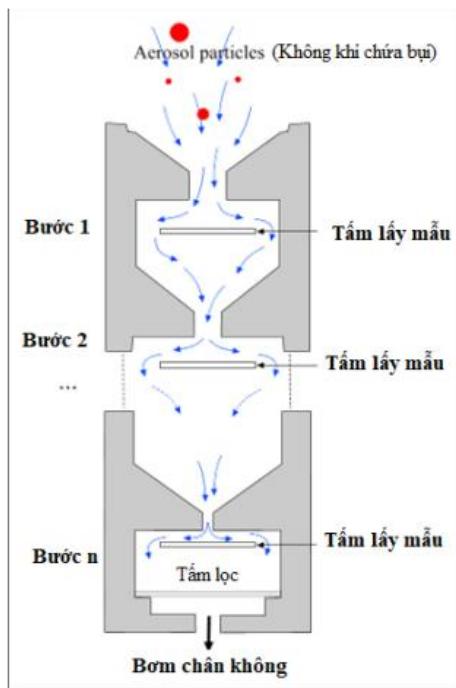
D1, D2 – Đầu dò bức xạ beta; 5 - Máy bơm không khí; 6 - Khí thải.

- Ưu điểm:
  - Độ chính xác cao
- Nhược điểm:
  - Giá thành cao
  - Nguy hiểm tới người sử dụng
  - Cần thời gian lấy mẫu

### 3.1.3 Phương pháp thác và chạm

Một phương pháp đo bụi khác cũng được sử dụng khá phổ biến là Thác và chạm (Cascade impactor). Thiết bị hoạt động dựa trên quan tính để tách các hạt bụi ra khỏi dòng khí. Dòng khí chứa bụi được bơm qua một thiết bị với nhiều tầng lọc và các vòi phun có kích thước khác nhau tương ứng với mỗi tầng. Khi không khí được tăng tốc thông qua nhiều vòi phun hẹp, các hạt nhỏ vẫn còn trong dòng chảy, trong khi các hạt lớn có quan tính đủ lớn sẽ bị giữ lại trên đĩa thu thập mẫu tương ứng.

Tổng nồng độ khối lượng của các hạt được đo bằng cách sử dụng một lớp lọc phủ Teflon và một cân điện tử độ nhạy cao để đo khối lượng của bộ lọc. Đôi với mục đích quan trắc, yêu cầu tần số lấy mẫu cao (1-10 s), trong khi các điểm lấy mẫu hàng loạt sử 14 dụng lượng mẫu lớn (high-volume samplers HVS) được chứa và bảo vệ trong bình lấy mẫu trong một thời gian dài để tính toán nồng độ trung bình và thực hiện phân tích thành phần hóa học.

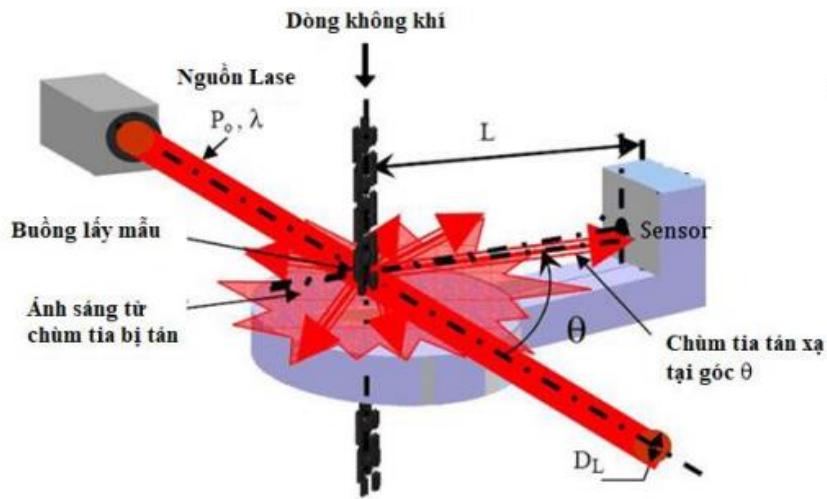


Hình 3.3: Phương pháp thác và chạm

- **Ưu điểm:**
  - Nhỏ gọn
  - Có thể đo được hạt có kích thước nhỏ
- **Nhược điểm:**
  - Giá thành cao

#### 3.1.4 Phương pháp tán xạ laser

Nguyên tắc của phương pháp này như sau: Khi một chùm laser đi qua không khí không chứa bụi, ánh sáng từ chùm tia không bị tán xạ. Khi trong không khí có bụi, ánh sáng từ chùm tia sẽ bị tán xạ ra xung quanh. Ánh sáng tán xạ được thu nhận bởi đầu thu và chuyển thành tín hiệu điện và các tín hiệu này sẽ được khuếch đại và xử lý. Số lượng và đường kính của các hạt có thể thu được bằng cách phân tích tín hiệu bởi vì dạng sóng tín hiệu có quan hệ nhất định với đường kính hạt. Cảm biến đo bụi như vậy sử dụng nguồn phát quang hồng ngoại gần (diode laser). Đầu thu là một diode quang kiểu thác với bộ khuếch đại. Nguồn phát hồng ngoại được sử dụng trong trường hợp này để tránh nhiễu với ánh sáng ban ngày vào buồng.

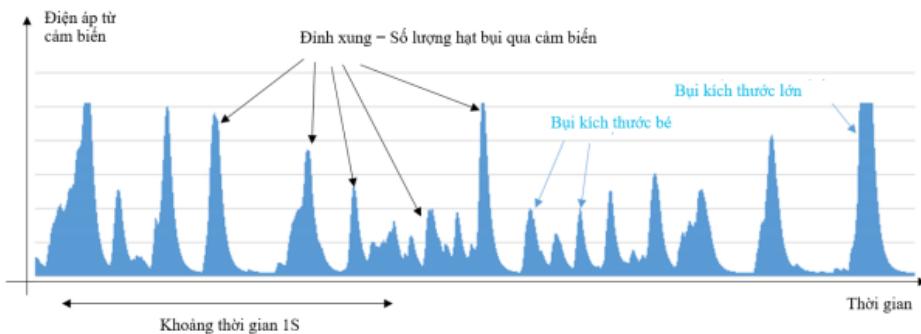


Hình 3.4: Phương pháp tán xạ laser

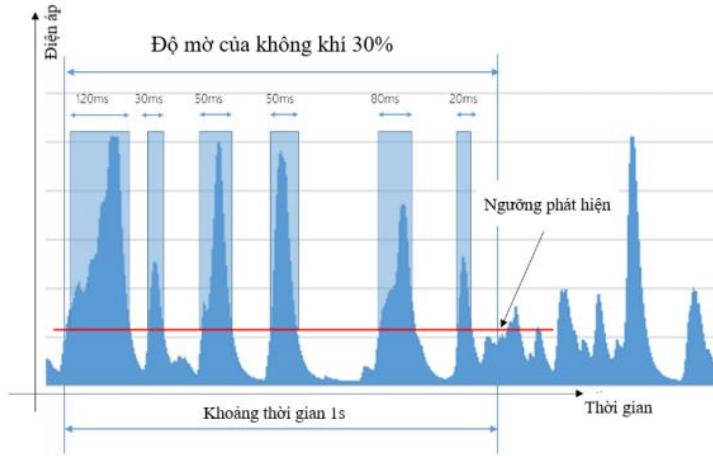
Mật độ bụi phụ thuộc chủ yếu vào lưu lượng không khí. Nguồn laser, cảm biến và ống kính chuẩn trực được đặt ở vị trí bên trên để ngăn chặn sự lảng đọng của bụi lên bề mặt khi dòng khí không lưu thông qua buồng đo. Góc tối ưu giữa nguồn Laser và cảm biến được tối ưu từ kết quả thực nghiệm. Mỗi hạt đi qua chùm tia laser khuếch tán một phần của chùm tia này tới cảm biến và lưu lượng không khí là không đổi, độ rộng xung của tín hiệu từ cảm biến được sử dụng để phân loại các hạt theo kích thước.

Cảm biến sẽ đếm số lượng các hạt bụi dựa trên thời gian các hạt bụi đi qua cảm biến. Đầu tiên, cảm biến áp dụng một dải lọc để loại bỏ các hạt rất nhỏ hoặc nhiễu, được biểu diễn bằng đường màu đỏ. Sau đó, nó đếm tổng thời gian của bất kỳ hạt nào được nhìn thấy độc lập với cường độ (hoặc kích thước hạt). Lượng thời gian này, còn được gọi là "Low Pulse Occupancy" (LPO: Thời gian xung thấp), có thể được coi là "độ mờ" của không khí lưu thông qua cảm biến. Để đo LPO cho các kích thước hạt khác nhau, cảm biến cung cấp đầu vào biến cho phép điều chỉnh bộ lọc bằng thông.

Biểu đồ dưới đây mô tả cách cảm biến "tính toán" chất lượng không khí:

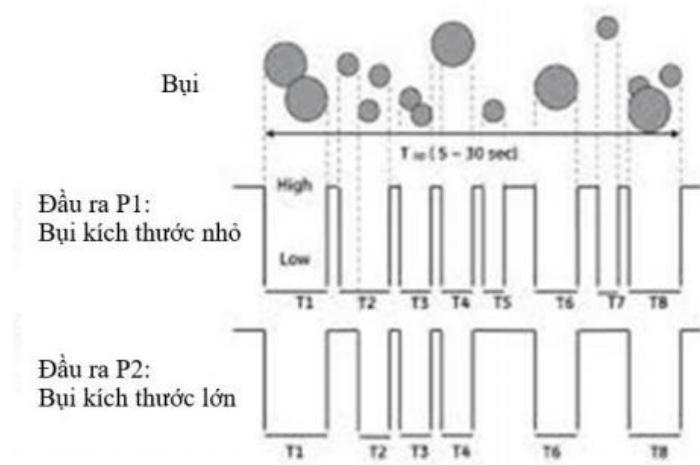


Hình 3.5: Dữ liệu thô từ cảm biến theo điện áp



Hình 3.6: Cảm biến đọc giá trị LPO

Với nguyên lý hoạt động như vậy, cảm biến có thể xác định được nồng độ bụi trong không khí và kích thước của bụi thông qua việc phân tích tín hiệu phản hồi từ đầu dò quang. Việc đếm số lượng hạt được thực hiện thông qua các bộ lọc tín hiệu bộ lọc thông thấp và bộ lọc thông cao tương ứng với đó là hai đầu ra P1 và đầu ra P2. [6]



Hình 3.7: Mối liên hệ giữa LPO và kích thước hạt

Các tín hiệu này sẽ được đưa vào vi xử lý có trên cảm biến để xử lý và đưa ra ngoài theo các chuẩn giao tiếp khác nhau.

- **Ưu điểm:**
  - Kích thước nhỏ
  - Giá thành thấp
  - Dễ sử dụng
  - Thuận tiện cho việc truyền thông với các thiết bị khác
- **Nhược điểm:**
  - Độ chính xác chưa cao, phụ thuộc vào yếu tố môi trường như nhiệt độ và độ ẩm không khí

Trong đồ án này, em sử dụng cảm biến SDS011 dựa trên phương pháp tán xạ laser. Đây là cảm biến cho phép đo online, có thể đo được nồng độ bụi PM10 và PM2.5 với dải đo từ 0 đến 999.9  $\mu\text{g}/\text{m}^3$ .

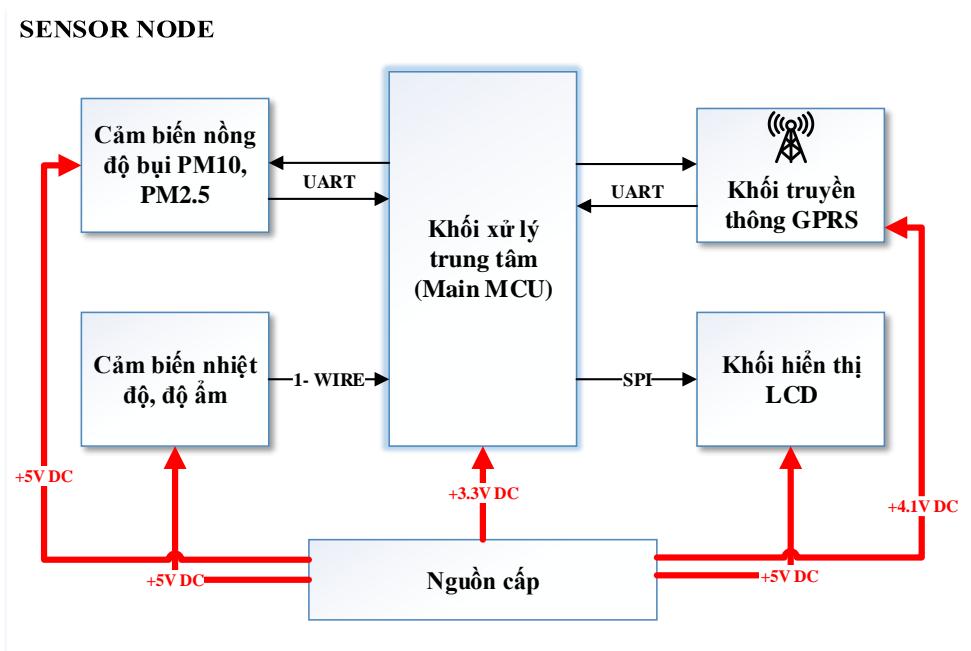
### 3.2 Yêu cầu thiết kế cho thiết bị

Thiết bị được thiết kế phải đáp ứng các yêu cầu sau:

- Đo được nồng độ bụi PM10 và PM2.5 từ 0.0 đến 999.9  $\mu\text{g}/\text{m}^3$
- Đo được nhiệt độ và độ ẩm môi trường xung quanh vị trí đặt thiết bị
- Có khả năng thay đổi linh hoạt các thuật toán dùng vi điều khiển
- Có khả năng truyền dữ liệu và lưu trữ trên Webserver thông qua truyền thông không dây GPRS.

### 3.3 Sơ đồ khái niệm thiết bị

Sơ đồ khái niệm của thiết bị Sensor Node được mô tả ở hình 3.8



Hình 3.8: Sơ đồ khái niệm thiết bị

Thiết bị được thiết kế gồm các khái niệm chính sau:

- **Khái niệm trung tâm:**

Khái niệm này đóng vai trò quan trọng nhất trong thiết bị, chịu trách nhiệm chính trong việc xử lý các tác vụ về truyền thông cũng như giao tiếp với cảm biến để lấy các dữ liệu đo được.

- **Khái niệm truyền thông GPRS**

Khái niệm truyền thông có chức năng cung cấp kết nối mạng từ thiết bị thu thập dữ liệu cục bộ tới các cấp trên, cụ thể là cho phép kết nối để truyền dẫn dữ liệu từ thiết bị tới Webserver.

- **Khái niệm cảm biến đo nồng độ bụi PM10, PM2.5**

Khái niệm này có chức năng thu thập, chuyển đổi nồng độ bụi PM10, PM2.5 thành tín hiệu số và truyền về khái niệm trung tâm.

- **Khái niệm cảm biến độ ẩm, nhiệt độ không khí**

Chức năng khái niệm này là thu thập, chuyển đổi các tín hiệu nhiệt độ, độ ẩm trong không khí thành tín hiệu số và truyền về khái niệm trung tâm.

- *Khối hiển thị LCD*

Khối này được thiết kế để chỉ thị các kết quả đo được từ cảm biến lên màn hình được đặt trên thiết bị.

- *Khối nguồn cung cấp*

Khối này có chức năng là đảm bảo cung cấp nguồn điện áp cho các khối chức năng khác hoạt động.

### 3.4 Thiết kế phần cứng

Để thực hiện chức năng của các khối đã nêu trong hình, mạch phần cứng của thiết bị được thiết kế gồm các thành phần:

- Vi điều khiển STM32F103C8T6 đảm nhận chức năng của khối “*Bộ xử lý trung tâm*”.
- SDS011 đảm nhận chức năng của khối “*Cảm biến đo nồng độ bụi PM10 và PM2.5*”
- DHT22 đảm nhận chức năng của khối “*Cảm biến nhiệt độ, độ ẩm không khí*”.
- SIM808 đảm nhận chức năng của khối “*Truyền thông GPRS*”.
- LCD NOKIA N5110 đảm nhận chức năng của khối: “*Màn hình hiển thị LCD*”
- IC LM2576 5V, IC ASM1117 3.3V đảm nhận chức năng của khối “*Nguồn cung cấp*”.

Chi tiết của các thành phần và sơ đồ nguyên lý ghép nối trong mạch được trình bày ngay dưới đây.

#### 3.4.1 Vi điều khiển STM32F103C8T6

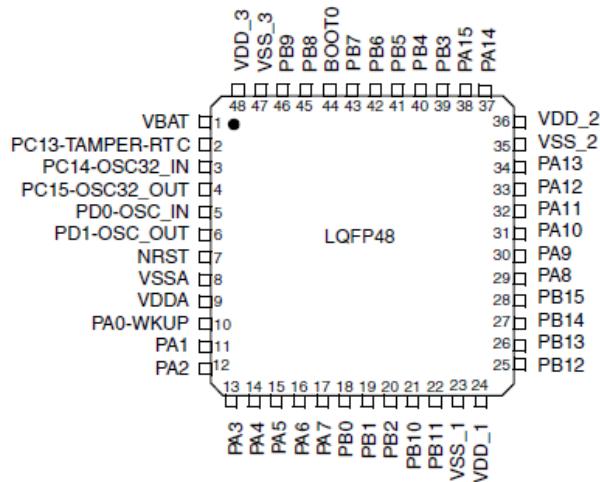
STM32 là một trong những dòng chip phổ biến của ST với nhiều họ thông dụng như F0,F1,F2,F3,F4..... STM32F103 thuộc họ F1 với lõi là ARM COTEX M3. STM32F103 là vi điều khiển 32 bit, tốc độ tối đa là 72Mhz. Giá thành cũng khá rẻ so với các loại vi điều khiển có chức năng tương tự. Mạch nạp cũng như công cụ lập trình khá đa dạng và dễ sử dụng. [7]



Hình 3.9: STM32F103C8T6

Bảng 3.1: Thông số kỹ thuật của vi điều khiển STM32F103C8T6

Thông số	Điều kiện	Giá trị	Đơn vị
Điện áp hoạt động		2.0 - 3.6	V
Dòng tiêu thụ	Xung nhịp 72 MHz	< 70	mA
Lõi xử lý		32bit, Arm Cortex-M3	
Xung nhịp tối đa		72	MHz
Giao tiếp I2C	Có		
Giao tiếp SPI	Có		
Giao tiếp UART	Có		
Bộ nhớ Flash	256		Kb
Bộ nhớ SRAM	64		Kb
Timer	8 bộ, 16bit		
Nhiệt độ hoạt động	-40 ~ 85		°C



Hình 3.10: Sơ đồ chân của STM32F103C8T6

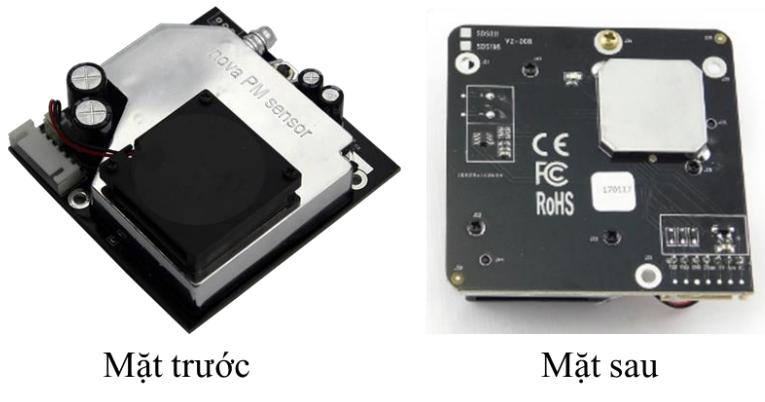
Một số ứng dụng chính của vi điều khiển: dùng cho driver để điều khiển ứng dụng, điều khiển ứng dụng thông thường, thiết bị cầm tay, máy tính và thiết bị ngoại vi chơi game, GPS cơ bản, các ứng dụng trong công nghiệp, thiết bị lập trình PLC, biến tần, máy in, máy quét, hệ thống cảnh báo, thiết bị liên lạc nội bộ...

Phần mềm lập trình: có khá nhiều trình biên dịch cho STM32 như IAR Embedded Workbench, Keil C... Ở đồ án này em sử dụng Keil C V5.

Thư viện lập trình: có nhiều loại thư viện lập trình cho STM32 như: STM32snippets, STM32Cube LL, STM32Cube HAL, Standard Peripheral Libraries, Mbed core. Mỗi thư viện đều có ưu và khuyết điểm riêng, ở đồ án này em sử dụng Standard Peripheral Libraries vì nó ra đời khá lâu và khá thông dụng, hỗ trợ nhiều ngoại vi và cũng dễ hiểu rõ bản chất của lập trình.

Mạch nạp: có khá nhiều loại mạch nạp như: ULINK, J-LINK, CMSIS-DAP, STLINK... Em sử dụng Stlink vì giá thành khá rẻ và khả năng debug lõi tốt.

### 3.4.2 Cảm biến đo nồng độ bụi SDS011



Hình 3.11: Cảm biến SDS011

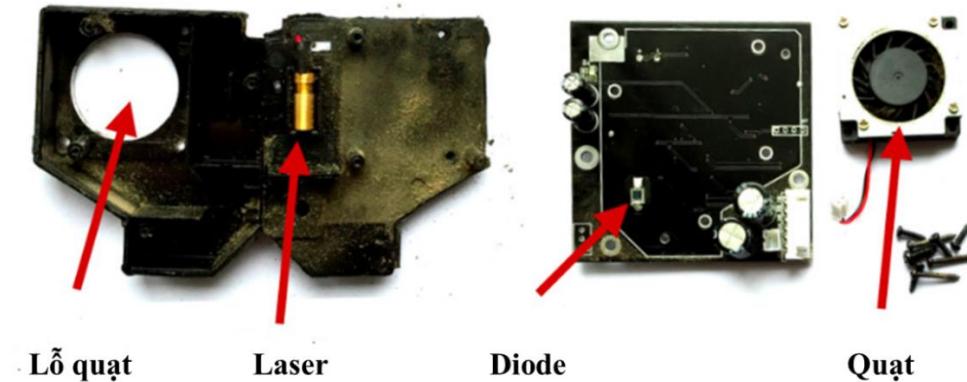
Cảm biến SDS011 là cảm đo nồng độ bụi PM10, PM2.5 sử dụng công nghệ tán xạ laser được phát triển bởi Inovafit - một công ty con của trường đại học Té Nam (Sơn Đông). Đây có thể coi là cảm biến đo bụi có độ chính xác cao trong các dòng cảm biến đo bụi kích thước nhỏ. Trong khi các cảm biến khác có xu hướng tập trung vào thu hẹp kích thước cảm biến, SDS011 lựa chọn giải pháp cân bằng giữa hiệu năng và kích thước trang bị thêm một quạt hút để tạo dòng khí đối lưu cho cảm biến, khác với các dòng cảm biến tương tự khác như Shinyei PPD24NS (một cảm biến từ Nhật Bản) sử dụng nhiệt điện trở để tạo dòng đối lưu tự nhiên.

Bảng 3.2 mô tả thông số kỹ thuật của cảm biến SDS011 [8]

Bảng 3.2: Thông số kỹ thuật cảm biến SDS011

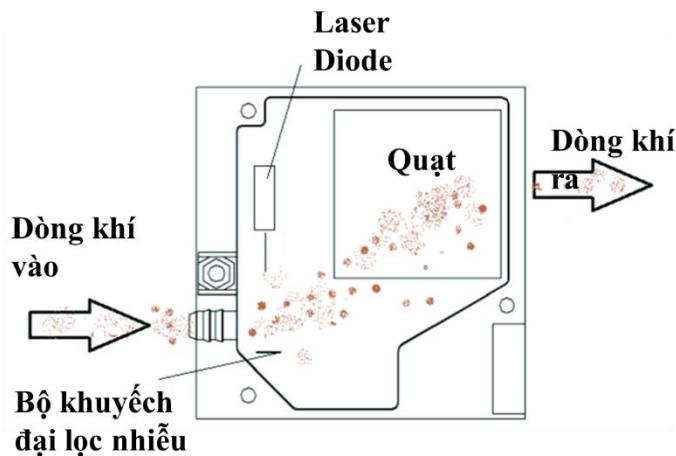
STT	Đối tượng	Thông số	Điều kiện
1	Thông số đo	PM2.5, PM10	
2	Khoảng đo	0.0-999.9 µg/m <sup>3</sup>	
3	Nguồn cung cấp	5V	
4	Dòng làm việc trung bình	70mA ± 10mA	
5	Dòng làm việc tối đa	220mA	
6	Dòng ngủ	<4mA	Laser và quạt ngủ
7	Khoảng nhiệt độ làm việc	-20 đến 50°C	
8	Thời gian đáp ứng	1 giây	
9	Tần số dữ liệu đầu ra	1 lần/giây	
10	Độ phân giải thông số bụi	<0.3 µm	
11	Sai số	Tối đa ±15% và ±10µg/m <sup>3</sup>	25°C, 50%RH
12	Kích thước sản phẩm	71x20x23mm	

Hình 3.12 mô tả các thành phần bên trong của cảm biến.



Hình 3.12: Bên trong cảm biến SDS011

Hình 3.13 mô tả dòng khí di chuyển bên trong cảm biến. Quạt được sử dụng trong cảm biến giúp dòng khí được lưu thông.



Hình 3.13: Mô tả dòng khí trong cảm biến SDS011

#### Nguyên lý hoạt động của cảm biến

Cảm biến sử dụng nguyên lý tán xạ laser: khi chùm laser qua các vị trí phát hiện hạt sẽ tạo ra sự tán xạ ánh sáng yếu, sóng ánh sáng tán xạ và hạt trong một hướng cụ thể có liên quan đến đường kính của sự tập trung số lượng hạt, qua thống kê kích thước phân loại dạng sóng khác nhau và công thức chuyển đổi có thể có được các hạt có kích thước khác nhau trong thời gian thực, phương pháp điều chỉnh để có được nồng độ khối lượng thống nhất với các đơn vị chính thức.

Kết quả về nồng độ bụi được tính toán bởi bộ xử lý 8 bit tích hợp trên cảm biến và đưa ra các chân kết nối.

Bảng 3.3 mô tả các chân kết nối của cảm biến.

Bảng 3.3: Chân cảm biến SDS011

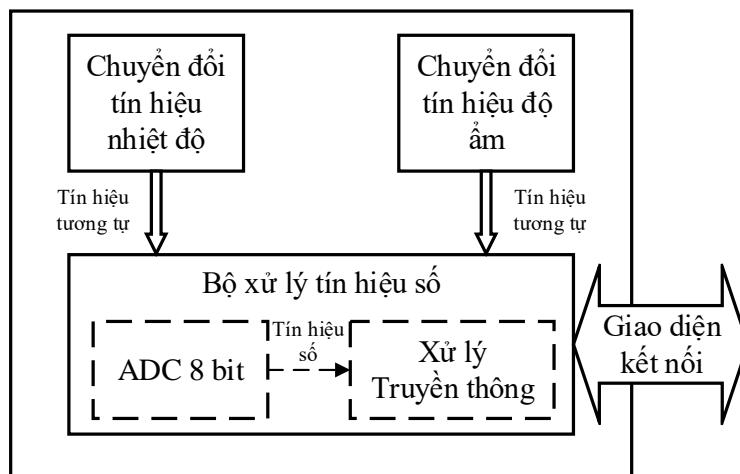
STT	Tên chân	Mô tả
1	CTL	Chân điều khiển ( Không sử dụng )
2	1µm	PM2.5 0-999 µg/m <sup>3</sup> , PWM Output
3	5V	Đầu vào 5V
4	2.5µm	PM10 0-999 µg/m <sup>3</sup> , PWM Output
5	GND	Đất
6	RX	Chân RX USART(TTL)
7	TX	Chân TX USART(TTL)

- Chân 1 là chân điều khiển backup của cảm biến (Không sử dụng)
- Chân 2 và 4 xuất đầu ra độ rộng xung PWM tỉ lệ với giá trị đo nồng độ khói lượng bụi PM2.5 và PM10 tương ứng.
- Chân 3 là chân cấp nguồn đầu vào 5VDC
- Chân 5 là chân GND
- Chân 6,7 là hai chân truyền dữ liệu theo UART (TTL) của cảm biến.

### 3.4.3 Cảm biến đo nhiệt độ độ ẩm DHT22

DHT22 là một loại cảm biến thông minh hai trong một, vừa cho phép đo được nhiệt độ, vừa cho phép đo được độ ẩm môi trường. Nhờ sử dụng công nghệ cảm biến mới, DHT22 đảm bảo mang lại độ tin cậy cao và có thể hoạt động ổn định trong thời gian dài. Cảm biến được tích hợp bên trong bao gồm:

- Bộ chuyển đổi tín hiệu độ ẩm theo nguyên lý trở kháng (resistive-type).
- Bộ chuyển đổi tín hiệu nhiệt độ sử dụng nhiệt điện trở bán dẫn loại NTC (điện trở giảm khi nhiệt độ tăng và ngược lại).
- Bộ xử lý số 8bit để số hóa kết quả đo và giao tiếp với khối xử lý trung tâm (host MCU bên ngoài).



Hình 3.14: Cấu trúc cảm biến nhiệt độ, độ ẩm DHT22

Các thông số hiệu chuẩn (Calibration) của DHT22 được lưu trữ trong vùng nhớ OTP và được bộ xử lý số sử dụng để xử lý kết quả trong suốt quá trình thực hiện phép đo. Việc cảm biến có kích thước nhỏ gọn và sử dụng chuẩn giao tiếp một dây giúp tích hợp dễ dàng và tiết kiệm tài nguyên cho khối xử lý trung tâm

(MCU). Ngoài ra, theo nhà sản xuất, nó còn có khả năng trao đổi dữ liệu trong khoảng cách lên tới 20m [9]

Bảng 3.4: Thông số kỹ thuật của cảm biến DHT22

Thông số	Giá trị	Đơn vị
Nguồn cung cấp	5.5	V
Tần số lấy mẫu tối đa	2	s
Dải đo nhiệt độ	-40 đến 80	°C
Sai số nhiệt độ	±0.5	°C
Dải đo độ ẩm	0-100	%
Sai số độ ẩm	2 đến 5	%
Kích thước	27 x 59 x 13.5	mm

Bảng 3.5: Thông số về điện của cảm biến DHT22

Thông số	Điều kiện	Tối thiểu	Trung bình	Tối đa	Đơn vị
Nguồn cung cấp	Một chiều (DC)	3	5	5.5	V
Dòng tiêu thụ	Chế độ đo	0.5		2.5	mA
	Chế độ chờ	100		150	uA

Về phần kết nối, DHT22 có giao diện kết nối gồm 4 chân trong đó một chân không được sử dụng, ba chân còn lại là các chân dùng để cấp nguồn và giao tiếp với khối xử lý trung tâm.

Bảng 3.6: Mô tả sơ đồ chân DHT22

Chân	Kí hiệu	Chú thích	
1	VDD	Nguồn cấp	
2	DATA	Chân dữ liệu	
3	NC	Không kết nối	
4	GND	Chân đất	

### 3.4.4 Màn hình LCD NOKIA N5110

LCD Nokia 5110 là màn hình LCD Graphic đơn giản với nhiều ứng dụng khác nhau. Nguồn gốc màn hình này thực chất là từ các thế hệ điện thoại di động trước đây. Nó đã được gắn trên một bộ PCB dễ dàng hàn gắn với các board khác.



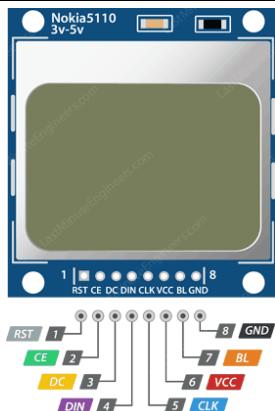
Hình 3.15: Màn hình LCD NOKIA N5110

LCD Nokia 5110 sử dụng vi điều khiển PCD8544, được sử dụng cùng với Nokia 3310 LCD trước đây. PCD8544 là dạng low power CMOS controller/driver, được thiết kế với chế độ hiển thị màn hình graphic là 84 cột và 48 hàng. Tất cả các chức năng cơ bản đã được tích hợp sẵn trên chip, từ đó cho ta hiệu quả về một thiết bị ngoại vi chiếm ít nguồn tiêu thụ. PCD8544 giao tiếp với vi điều khiển thông qua một loạt các chân bus đã được cung cấp sẵn.

Các thông số kỹ thuật cơ bản của LCD N5110 được mô tả dưới bảng 3.7.

Bảng 3.7: Thông số kỹ thuật màn hình LCD NOKIA N5110

Thông số	Giá trị	Đơn vị
Điện áp hoạt động	+5	VDC
Dòng tiêu thụ	6	mA
Độ phân giải	84×48	pixels
Nhiệt độ hoạt động	-25 đến 70	°C



Hình 3.16: Sơ đồ chân của LCD N5110

Chức năng của từng chân được mô tả ở bảng 3.8 [10]

Bảng 3.8: Chức năng các chân của LCD N5110

Chân	Tên	Ký hiệu	Chức năng
1	Reset	RST	Tín hiệu Reset, giúp khởi động, reset màn hình
2	Chip Enable	CE	Tín hiệu Chip Select, giúp đưa ra tín hiệu chọn chip hoạt động ở mức tích cực thấp
3	Data/Command	DC	Tín hiệu Data/Command giúp ta chọn chế độ đưa lệnh vào PCD8544 là lệnh điều khiển, hay dữ liệu hiển thị
4	Serial Input	DIN	Dữ liệu đầu vào có thể là lệnh điều khiển hoặc dữ liệu hiển thị màn hình
5	Clock	CLK	Xung Clock điều khiển
6	Power	Vcc	Nguồn nuôi
7	Back Light	BL	Đát cho đèn LED Backlight
8	Ground	GND	Đát cho nguồn nuôi

### 3.4.1 Khối SIM808

SIM808 là bản nâng cấp của SIM908, hỗ trợ GSM/GPRS với 4 băng tần 850/900/1800/1900MHz và công nghệ định vị vệ tinh GPS. Ngoài 2 chức năng chính GSM/GPS, Sim808 còn có thể hỗ trợ thêm tính năng Bluetooth.

Sim808 giao tiếp với các thiết bị khác thông qua giao tiếp UART và tập lệnh AT Command.



Hình 3.17: SIM808

Bảng 3.9 trình bày một số thông số kỹ thuật cơ bản của SIM808. [11]

Bảng 3.9: Thông số kỹ thuật của SIM808

Thông số	Điều kiện	Giá trị	Đơn vị
Điện áp hoạt động		3.4 - 4.4	VDC
Hỗ trợ		GPRS/GPS/GSM/Bluetooth	
Dòng tiêu thụ	Trung bình	100	mA
	Sleep	1	mA
	Cực đại	2 (trong 1 ms)	A
Dải băng tần		850/900/1800/1900	Mhz
Mã hóa CS		1,2,3,4	
Hỗ trợ GPRS		12/10	
Kích thước		24x24x2.6	mm
Nhiệt độ hoạt động		-40~85	°C

### 3.4.2 Khối nguồn

Cảm biến SDS011, cảm biến DHT22, màn hình LCD N5110 yêu cầu điện áp cung cấp 5V DC. Vi điều khiển STM32F103C8T6 yêu cầu điện áp 3.3V DC và SIM808 là 3,4-4,4V DC. Điện áp SIM808 sử dụng là từ 3,4-4,4VDC nên ta sẽ dùng một diode để giảm điện áp từ 5VDC xuống khoảng 4,3VDC cho SIM808 hoạt động. Vì vậy ta cần thiết kế một khối nguồn có chức năng cấp 3 mức điện áp điện áp 5V, 4,3V 3.3V DC cung cấp cho các linh kiện từ nguồn điện 220VAC.

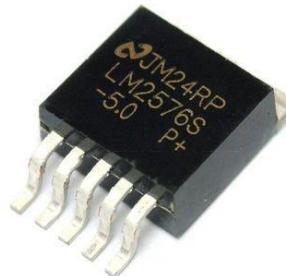
Trước hết ta cần chuyển từ nguồn xoay chiều sang một chiều. Để thực hiện việc này em sử dụng Adapter chuyển từ điện áp xoay chiều 220V, 50Hz sang điện áp 1 chiều 12V, 2A. Mô tả về Adapter này thể hiện trên hình 3.19.



Hình 3.18: Adapter 12V, 2A

Sau khi đã có nguồn 12VDC ta thực hiện chuyển đổi sang nguồn 5V, 4.3V và 3.3V DC.

Để chuyển từ 12V DC xuống 5V DC, em sử dụng IC LM2576.



Hình 3.19: LM2576

Đây là một loại IC nguồn tuyến tính cho phép chuyển đổi điện áp cung cấp từ bên ngoài thành điện áp 5VDC để nuôi các phần tử khác trong mạch. Dưới đây là một vài các thông số cơ bản cũng như sở đồ nguyên lý của mạch nguồn sử dụng LM2576. [12]

Bảng 3.10: Thông số kỹ thuật cơ bản của LM2576

Thông số	Giá trị	Đơn vị
Điện áp vào (VIN)	7 to 40	V
Điện áp đầu ra	4.8 to 5.2	V
Dòng ra tối đa	3	A
Nhiệt độ hoạt động	-40 to 125	°C

Bảng 3.11: Sơ đồ chân LM2576

Chân	Kí hiệu	Chú thích	
1	V <sub>IN</sub>	Chân điện áp vào	
2	V <sub>OUT</sub>	Chân điện áp ra	
3	GND	Đất	
4	Feedback	Chân phản hồi	
5	On/OFF	Chân bật tắt	

Để chuyển đổi từ nguồn 5V DC sang 3.3V DC em sử dụng IC ổn áp AMS1117- 3.3.



Hình 3.20: AMS1117-3.3

Đây là một loại IC nguồn tuyến tính cho phép chuyển đổi điện áp cung cấp từ bên ngoài thành điện áp 3.3V để nuôi các phần tử khác trong mạch. Dưới đây là một vài các thông số cơ bản cũng như sở đồ nguyên lý của mạch nguồn sử dụng AMS117.

Bảng 3.12: Thông số kỹ thuật cơ bản của AMS117

Thông số	Điều kiện	Giá trị	Đơn vị
Điện áp vào ( $V_{IN}$ )		3.3 to 15	V
Điện áp đầu ra	$V_{IN} = 4.8V$	3.2 to 3.4	V
Dòng ra tối đa		1	A
Line Regulation (Max)	$1.5V \leq (V_{IN} - V_{OUT}) \leq 12V$	0.2	%
Load Regulation	$V_{IN} = 4.75V, 0 \leq I_{OUT} \leq 0.8A$	0.4	%
Nhiệt độ hoạt động		-40 to 125	°C

Bảng 3.13: Sơ đồ chân AMS117

Chân	Kí hiệu	Chú thích	
1	$V_{OUT}$	Chân cung cấp điện áp ra	
2	GND	Đất	
3	$V_{IN}$	Chân điện áp vào	

Để tạo nguồn 4,3VDC cấp cho SIM808, em sử dụng một Diode thường nối vào nguồn 5VDC. Diode có tác dụng làm giảm điện áp đi qua nó khoảng 0.7V. Như vậy ta sẽ có nguồn 3,4-4,4VDC cấp cho SIM808.

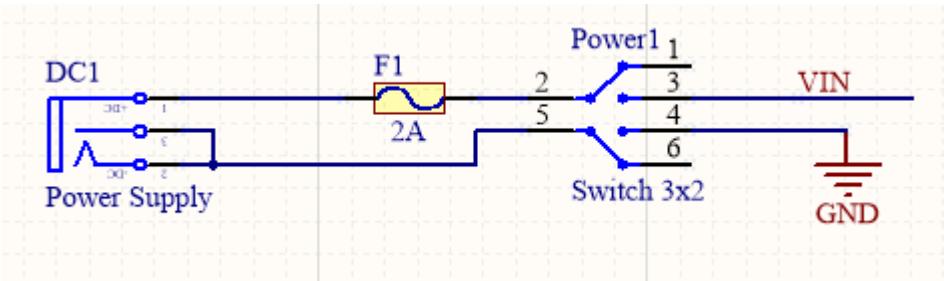


Hình 3.21: Diode

### 3.4.3 Mạch nguyên lý và mạch in

#### 3.4.3.1. Khối nguồn

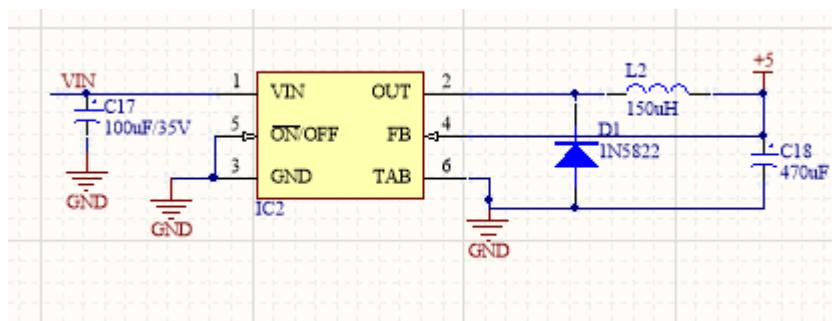
a) Đầu vào adapter 12VDC, 2A



Hình 3.22: Đầu vào nguồn adapter

Nguồn adapter 12VDC, 2A được đưa vào mạch thông qua Jack DC-DC. Cầu chì F1 giúp bảo vệ mạch và nút nhấn giữ dùng để bật/tắt thiết bị.

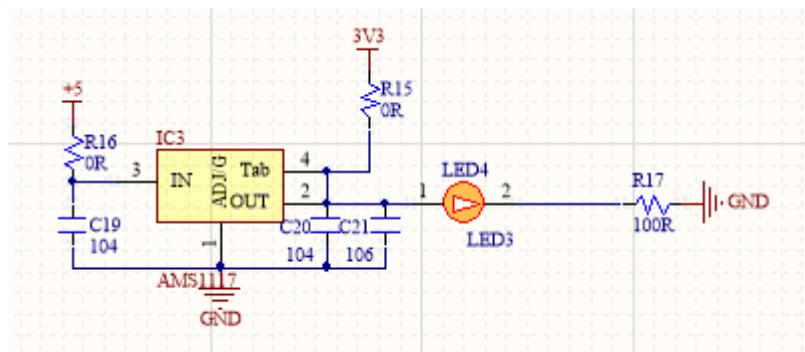
b) Mạch sử dụng LM2576 tạo nguồn 5VDC



Hình 3.23: Mạch nguồn 5VDC

Trong hình 3.23, nguồn đầu vào được cung cấp tới 2 chân VIN và GND. Tụ C17 dùng để duy trì sự ổn định đầu vào. Cuộn cảm L2 làm phẳng dạng song dòng điện đầu ra, làm dòng ra ổn định và tránh sụt dòng trên tải. Tụ C18 giúp ổn định điện áp đầu ra. Diode giúp tránh dòng ngược.

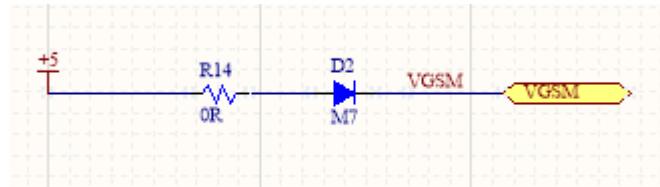
c) Mạch sử dụng AMS 1117 tạo nguồn 3,3VDC



Hình 3.24: Mạch nguồn 3.3VDC

Nguồn đầu vào của AMS1117 là nguồn 5VDC. Các tụ C19, C20, C21 giúp ổn định đầu vào và đầu ra. LED4 là đèn báo chỉ thị nguồn. Các trở R15, R16 có giá trị 0Ω dùng để kiểm tra mạch khi gắp sục.

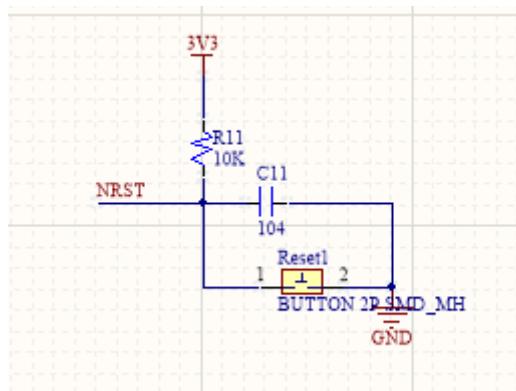
d) Mạch tạo nguồn 4,3VDC cho SIM808



Hình 3.25: Mạch nguồn 4.3VDC

Diode D2 giúp giảm điện áp từ 5VDC xuống 4,3VDC cung cấp cho SIM808, R14 có giá trị 0Ω để kiểm tra mạch.

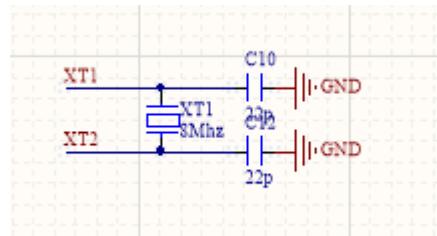
#### 3.4.3.2. Khởi Reset



Hình 3.26: Khởi Reset

Khởi Reser làm nhiệm vụ khởi động lại vi điều khiển mà không phải ngắt nguồn cung cấp.

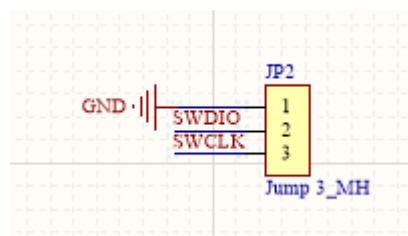
#### 3.4.3.3. Khởi tạo dao động



Hình 3.27: Khởi tạo dao động

Thạch anh 8MHz cung cấp Clock ngoài cho hoạt động của vi điều khiển.

#### 3.4.3.4. Khởi nạp chương trình



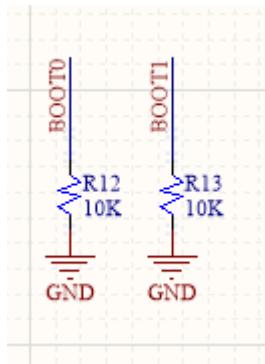
Hình 3.28: Khởi nạp chương trình

Khối này dùng để nạp chương trình cho vi điều khiển thông qua mạch nạp STLINK V2. Sơ đồ kết nối chân giữa thiết bị và mạch nạp STLINK V2 được mô tả dưới bảng 3.14

Bảng 3.14: Kết nối chân mạch nạp và thiết bị

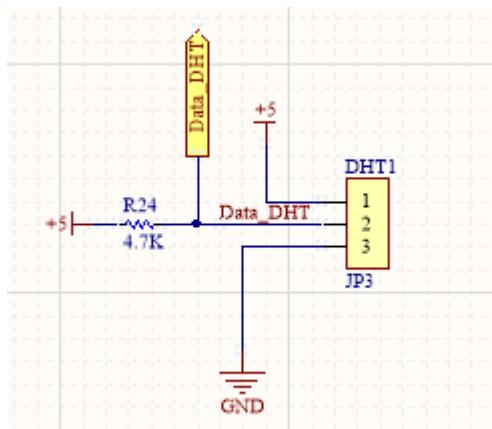
STLINK V2	STM32F103C8T6
GND	GND
SWDIO	DIO
SWCLK	CLK

Vì mạch này không sử dụng BootLoader nên hai chân Boot0 và Boot1 được kéo xuống 0 thông qua điện trở kéo nguồn.



Hình 3.29: 2 chân Bootloader

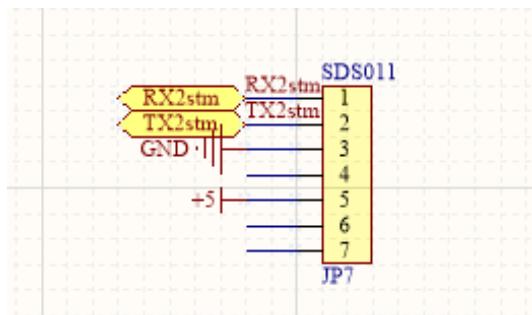
#### 3.4.3.5. Khối giao tiếp cảm biến DHT22



Hình 3.30: Kết nối chân của DHT22

Nguồn 5VDC được cấp cho cảm biến hoạt động thông qua hai chân VDD và GND. Điện trở treo 4.7k được sử dụng để kéo đường DATA lên mức cao sẽ đảm bảo loại trừ trạng thái không xác định. Khi thực hiện cấp nguồn cho cảm biến hoạt động cần đảm bảo không gửi bất cứ dữ liệu gì cho cảm biến trong giây đầu tiên để tránh trạng thái hoạt động mất ổn định của cảm biến về sau.

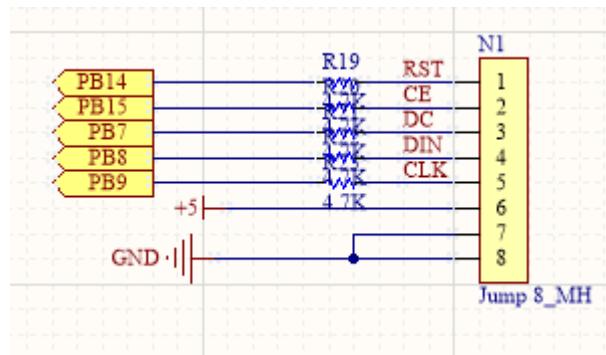
#### 3.4.3.6. Khối giao tiếp cảm biến SDS011



Hình 3.31: Kết nối chân của SDS011

Cảm biến SDS011 giao tiếp với vi điều khiển bằng chuẩn giao tiếp USART thông qua 2 chân TX, RX. Nguồn 5VDC được cấp cho cảm biến thông qua 2 chân GND và +5.

#### **3.4.3.7. Khối giao tiếp LCD N5110**



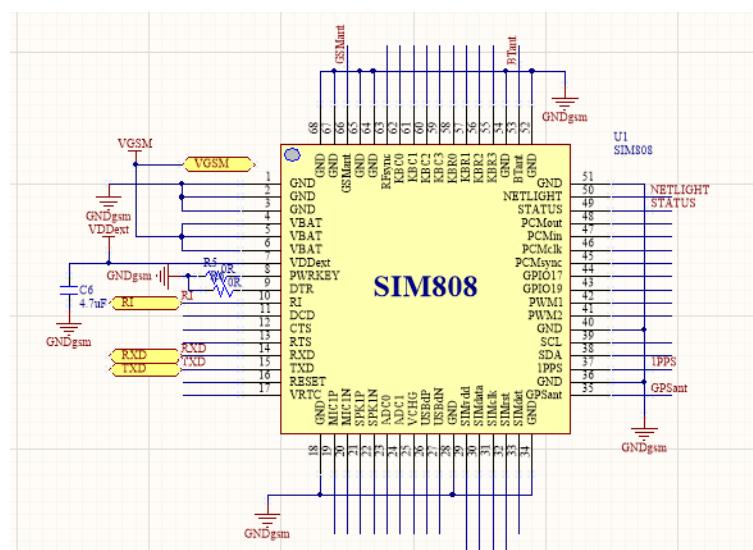
Hình 3.32: Kết nối chân của LCDN5110

Màn hình LCD N5110 kết nối với vi điều khiển thông qua 8 chân. Các trổ 4.7k được sử dụng để treo các chân điều khiển lên mức cao tránh tình trạng không xác định. Nguồn cấp 5VDC cho LCD thông qua 2 chân +5 và GND

### 3.4.3.8. Khởi SIM808

Khối SIM 808 gồm các phần:

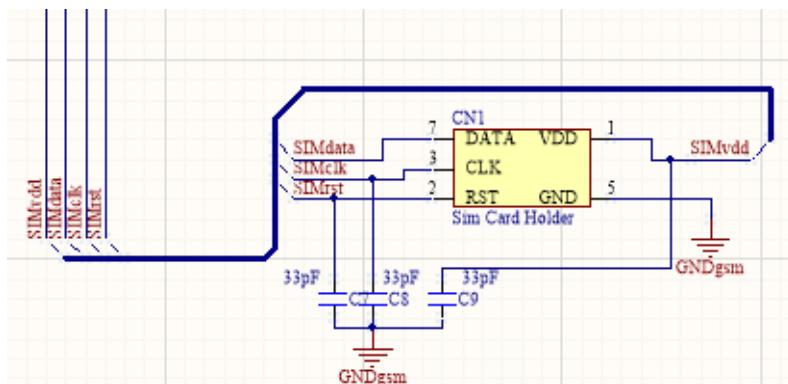
a) SIM808



Hình 3.33: Mach SIM808

SIM808 được cấp nguồn 4,3VDC từ 2 chân GNDgsm và VGSM. 2 chân điều khiển bật/tắt là PWRKEY và DTR được kéo xuống đất để SIM808 luôn hoạt động

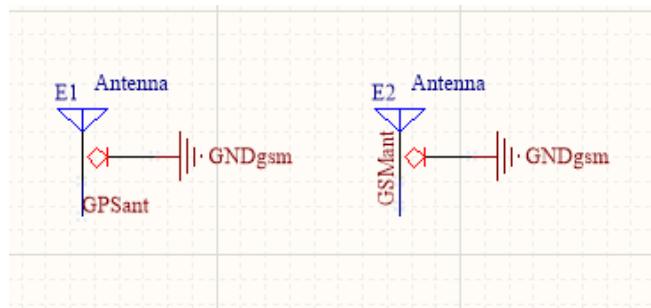
b) Simcard



Hình 3.34: Simcard

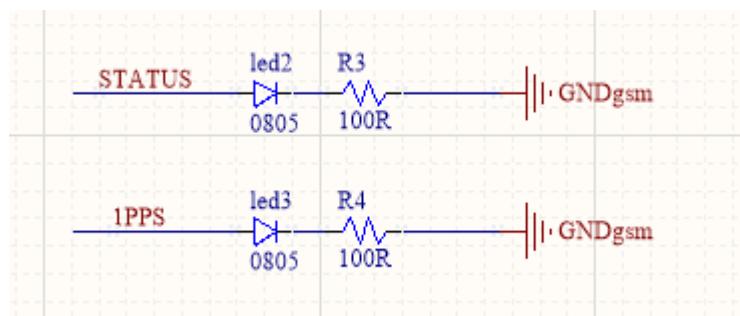
Simcard kết nối với SIM808 thông qua 4 chân SIMvdd, SIMdata, SIMclk, SIMrst.

c) Antena



Hình 3.35: Amntena

d) Led báo của SIM808



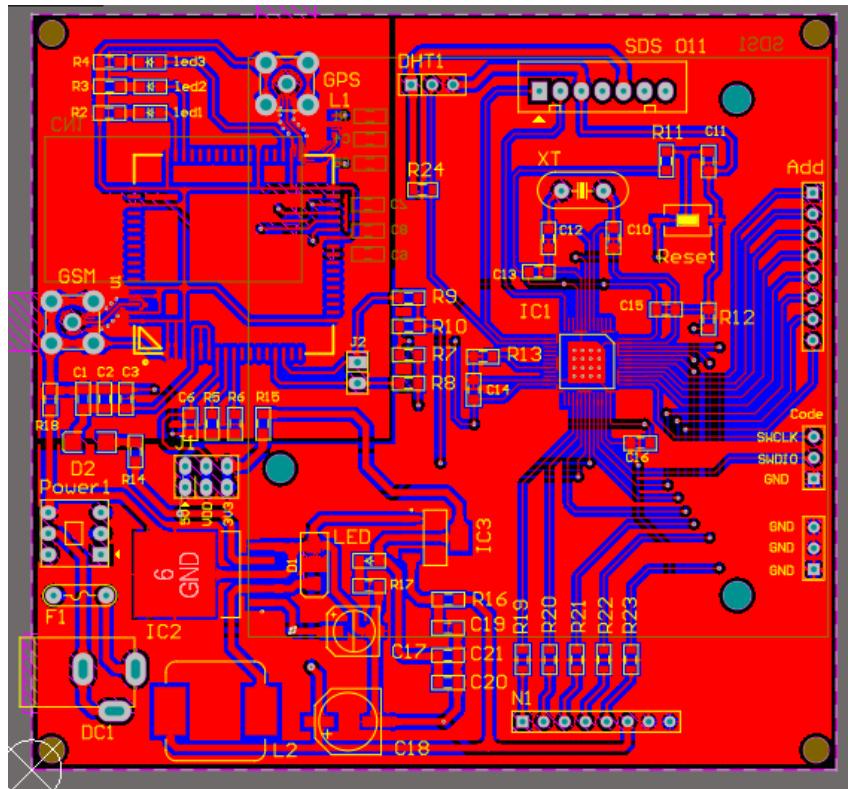
Hình 3.36: Led báo SIM808

Led2 có tác dụng báo nguồn của SIM808, led3 có tác dụng báo các trạng thái hoạt động của SIM808.

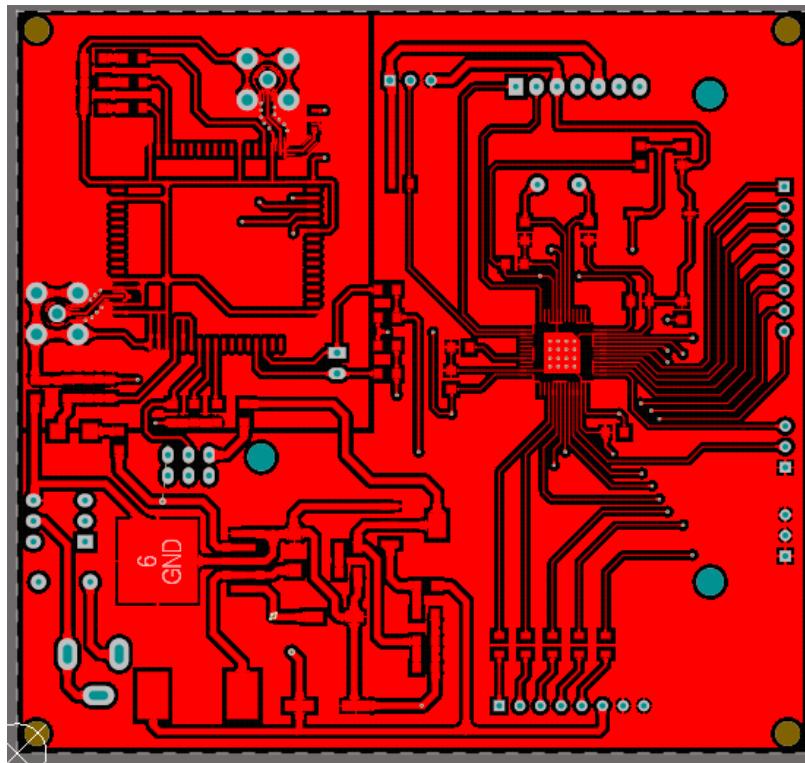
### 3.4.3.9. Mạch in

Mạch in được thiết kế trên phần mềm Altium 18, dưới đây là một số hình ảnh thiết kế:

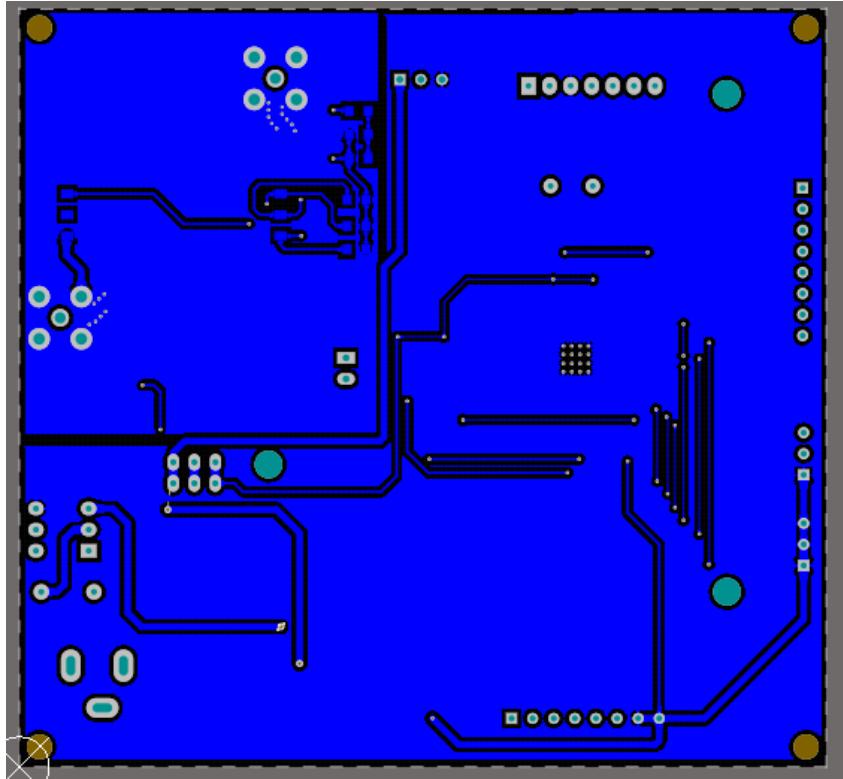
Mạch có kích thước: 92mmx96mm, là mạch PCB 2 lớp.



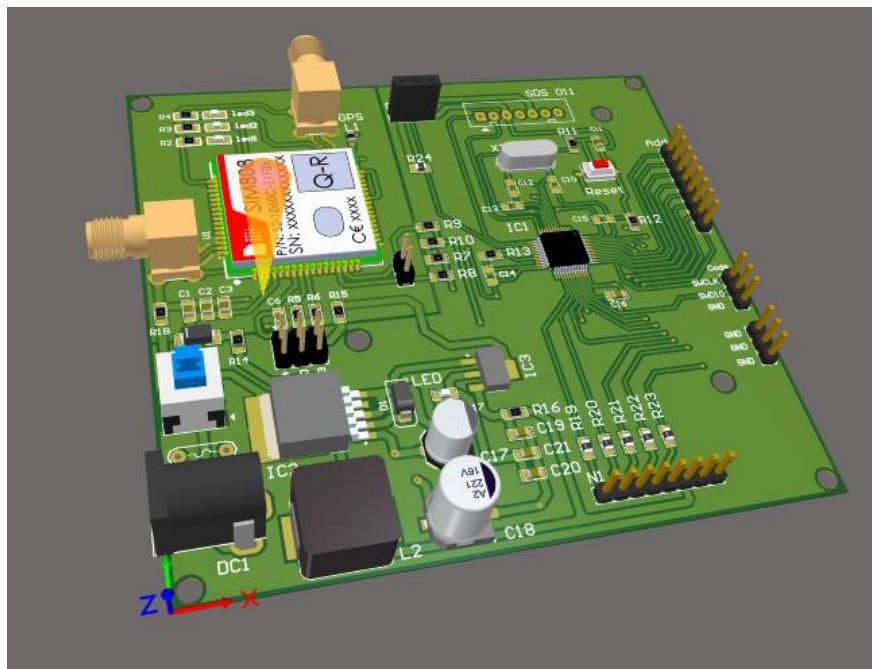
Hình 3.37: Layer tổng quan



Hình 3.38: Top layer



Hình 3.39: Bottom layer

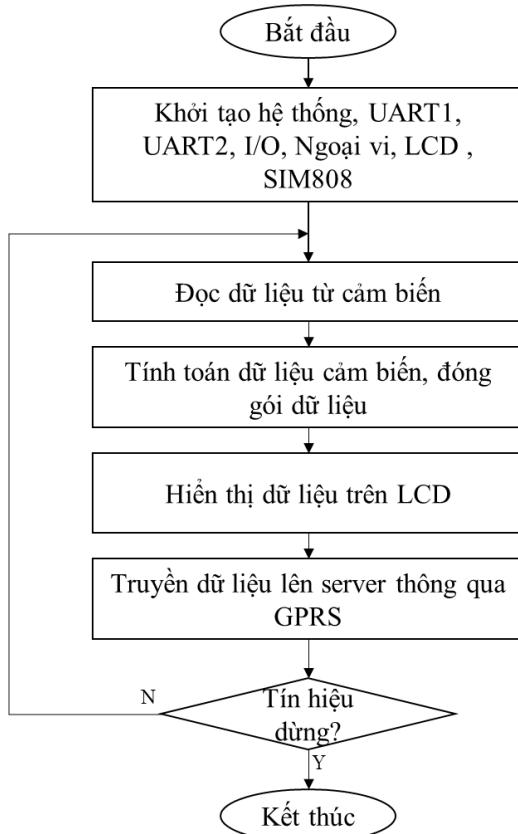


Hình 3.40: Mô hình 3D

### 3.5 Thiết kế phần mềm

#### 3.5.1 Lưu đồ thuật toán

Trình tự các bước thực hiện của chương trình phần mềm trên vi xử lý được thể hiện trên hình 3.41.



Hình 3.41: Sơ đồ khái niệm chương trình chính

Trình tự hoạt động:

Khi mạch được cấp nguồn hoạt động, vi xử lý sẽ bắt đầu quá trình cấu hình (quá trình khởi tạo) bao gồm việc khởi tạo các biến toàn cục, Clock, cấu hình ngắt, UART1, UART2, GPIO và đồng thời khởi tạo SIM808 và màn hình LCD. Sau khi khởi tạo xong vi xử lý sẽ kết nối tới các cảm biến và tiến hành đọc dữ liệu. Dữ liệu sau khi được đọc về sẽ được vi xử lý tính toán và đóng gói lại. Sau đó vi xử lý sẽ truyền dữ liệu lên LCD để hiển thị và truyền lên Server thông qua kết nối GPRS. Nếu có tín hiệu dừng, vi xử lý sẽ kết thúc hoạt động. Nếu không có tín hiệu dừng, chương trình sẽ quay lại bước kết nối và đọc các giá trị từ cảm biến.

#### 3.5.2 Giao tiếp với cảm biến SDS011

Cảm biến SDS011 giao tiếp với vi điều khiển thông qua chuẩn giao tiếp UART.

Giao thức truyền thông UART trên cảm biến:

- Bit rate : 9600
- Data bit : 8

- Parity bit : NO
- Stop bit : 1

Bảng 3.15: Chuỗi giá trị truyền thông UART cảm biến SDS011

Thứ tự các byte được truyền	Tên	Giá trị
0	Message header	AA
1	Commander NO	C0
2	DATA1	PM2.5 low byte
3	DATA2	PM2.5 high byte
4	DATA3	PM10 low byte
5	DATA4	PM10 high byte
6	DATA5	Reserved
7	DATA6	Reserved
8	Checksum	Checksum
9	Message tail	AB

Khi cảm biến truyền giá trị đo được được gửi lên theo chuẩn giao tiếp UART, cứ sau 1 giây nó sẽ truyền một lúc 10 byte dữ liệu liên tiếp.

Trong đó: 2 byte đầu là AA, C0 (mã Hexa), 2 byte tiếp theo để gửi giá trị PM2.5, 2 byte tiếp theo nữa gửi giá trị PM10, byte thứ 9 là giá trị checksum và byte kết thúc thứ 10 có giá trị AB, nhằm mục đích nâng cao độ tin cậy của giá trị đo gửi lại từ cảm biến.

Giá trị tính toán cụ thể như sau:

Checksum:

$$\text{Checksum} = \text{DATA1} + \text{DATA2} + \text{DATA3} + \text{DATA4} + \text{DATA5} + \text{DATA6}$$

PM2.5 value:

$$PM\ 2.5 = \frac{PM\ 2.5\ High\ byte * 256 + PM\ 2.5\ Low\ byte}{10} (\mu g / m^3) \quad PT\ 3.1$$

PM10 value:

$$PM\ 10 = \frac{PM\ 10\ High\ byte * 256 + PM\ 10\ Low\ byte}{10} (\mu g / m^3) \quad PT\ 3.2$$

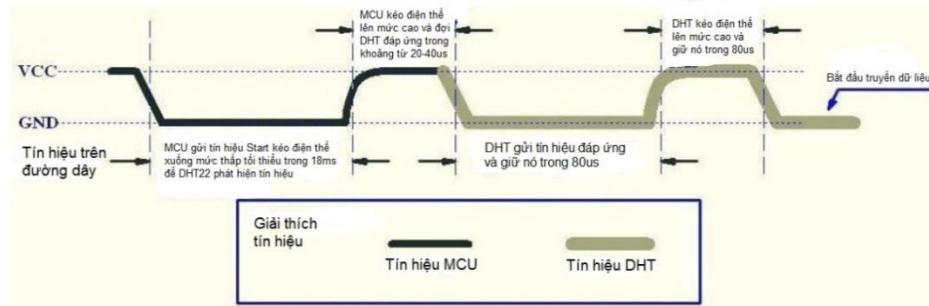
### 3.5.3 Giao tiếp với cảm biến DHT22

DHT22 là cảm biến sử dụng giao tiếp 1 dây, mỗi khi muốn đọc dữ liệu vi xử lý gửi tín hiệu yêu cầu đọc và đợi dữ liệu gửi về từ cảm biến, ở đây chân data của DHT22 được nối với chân PA4 của vi xử lý.

Trình tự đọc dữ liệu từ cảm biến gồm hai bước:

- Gửi tín hiệu muốn đo (Start) tới DHT22, sau đó DHT22 xác nhận lại.
- Khi đã giao tiếp được với DHT22, Cảm biến sẽ gửi lại 5 byte dữ liệu và nhiệt độ đo được.

## Bước 1: Gửi tín hiệu Start:



Hình 3.42: Tín hiệu Start trong giao tiếp với cảm biến DHT22

- Vì xử lý thiết lập chân DATA là Output, kéo chân DATA xuống 0 trong khoảng thời gian >18ms. Ở đây chọn thời gian 25ms. Khi đó DHT22 sẽ hiểu vi xử lý muốn đo giá trị nhiệt độ và độ ẩm.
- Vì xử lý đưa chân DATA lên 1, sau đó thiết lập lại là chân đầu vào.
- Sau khoảng 20-40us, DHT22 sẽ kéo chân DATA xuống thấp. Nếu >40us mà chân DATA ko được kéo xuống thấp nghĩa là ko giao tiếp được với DHT22.
- Chân DATA sẽ ở mức thấp 80us sau đó nó được DHT22 kéo lên cao trong 80us. Bằng việc giám sát chân DATA, Vi xử lý có thể biết được có giao tiếp được với DHT22 không. Nếu tín hiệu đo được DHT22 lên cao, khi đó hoàn thiện quá trình giao tiếp của Vi xử lý với DHT22.

## Bước 2: Đọc giá trị trên DHT22

- DHT22 sẽ trả giá trị nhiệt độ và độ ẩm về dưới dạng 5 byte. Trong đó:
  - Byte 1: giá trị byte cao của độ ẩm
  - Byte 2: giá trị byte thấp của độ ẩm
  - Byte 3: giá trị byte cao của nhiệt độ
  - Byte 4: giá trị byte thấp của nhiệt độ
  - Byte 5: kiểm tra tổng.

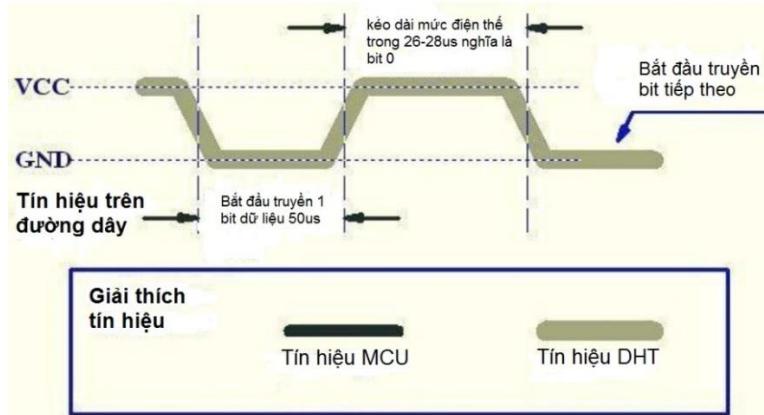
=> Nếu Byte 5 = (8 bit) (Byte1 + Byte2 + Byte3 + Byte4) thì giá trị độ ẩm và nhiệt độ là chính xác, nếu sai thì kết quả đo không có nghĩa.

Kết quả đo nhiệt độ độ ẩm được tính theo công thức:

- Độ ẩm = (Byte 1 \* 256 + Byte 2)/10 (đơn vị %)
- Nhiệt độ = (Byte 3\*256 + Byte 4)/10 (đơn vị °C)

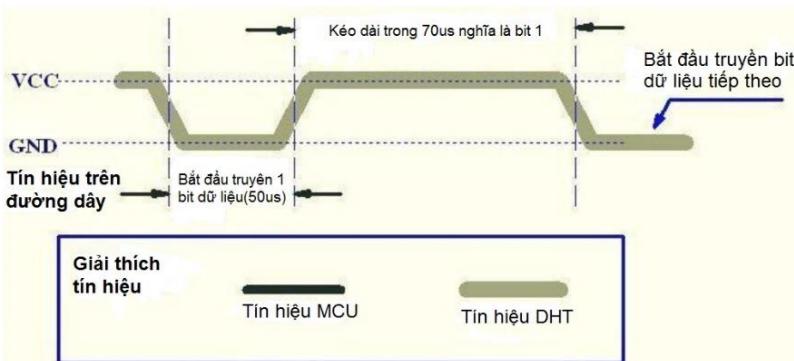
Sau khi giao tiếp được với DHT22, DHT22 sẽ gửi liên tiếp 40 bit 0 hoặc 1 về Vi xử lý, tương ứng chia thành 5 byte kết quả của nhiệt độ và độ ẩm.

Bit 0:



Hình 3.43: Bit dữ liệu 0 của DHT22

Bit 1:



Hình 3.44: Bit dữ liệu 1 của DHT22

Sau khi tín hiệu được đưa về 0, ta đợi chân Data của vi xử lý được DHT22 kéo lên 1. Nếu chân DATA là 1 trong khoảng 26-28 us thì là bit 0, còn nếu tồn tại 70us là bit 1. Do đó trong lập trình ta bắt sùn lên của chân DATA, sau đó delay 50us. Nếu giá trị đo được là 0 thì ta đọc được bit 0, nếu giá trị đo được là 1 thì giá trị đo được là 1. Cứ như thế ta đọc các bit tiếp theo.

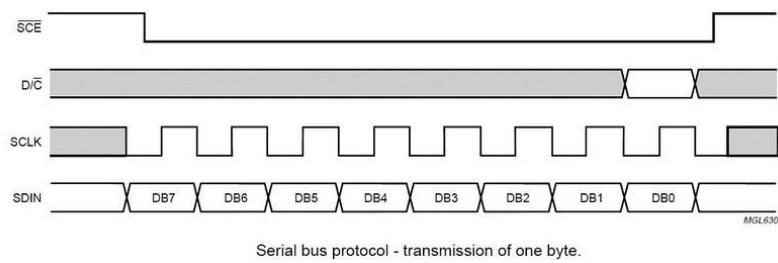
### 3.5.4 Giao tiếp với màn hình LCD NOKIA N5110

Việc gửi dữ liệu từ vi điều khiển đến LCD NOKIA 5110 được chia làm chế độ. Chế độ gửi lệnh điều khiển LCD và chế độ gửi dữ liệu hiển thị ra LCD. Chân DC được sử dụng để chọn 1 trong 2 chế độ này.

- Nếu DC=0, dữ liệu gửi đến LCD là lệnh dùng để điều khiển hoạt động của LCD (dữ liệu này không được hiển thị ra màn hình).
- Nếu DC=1, dữ liệu gửi đến LCD là dữ liệu hiển thị ra màn hình LCD.

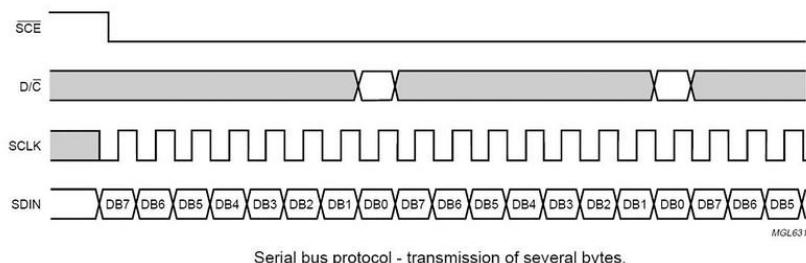
Có 2 cách để gửi dữ liệu đến LCD.

Cách thứ nhất: Gửi mỗi lần 1 byte đến LCD



Hình 3.45: Chế độ truyền 1 byte

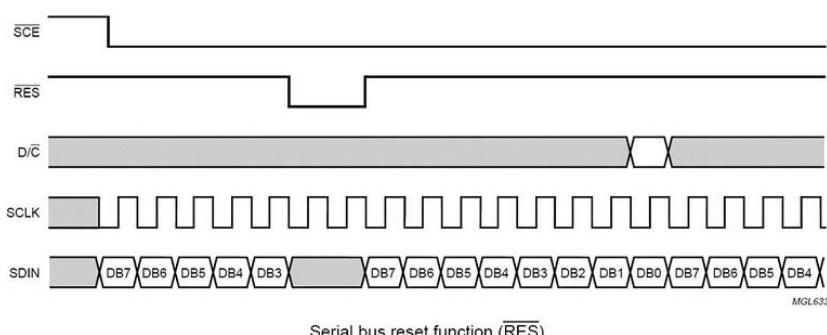
Cách thứ hai: Gửi nhiều byte liên tiếp nhau



Hình 3.46: Chế độ truyền nhiều byte

Khi chân CE ở mức cao (CE=1), bất kỳ một sự thay đổi tín hiệu nào trên chân CLK cũng không ảnh hưởng đến LCD. Người dùng chỉ có thể gửi dữ liệu đến LCD khi chân CE ở mức thấp(CE=0).

Sau mỗi chu kì của xung clock (xung cạnh lên) thì 1 bit dữ liệu được dịch vào LCD trên chân DIN. Chân CE sẽ được giữ ở mức thấp(CE=0) cho tới khi việc gửi dữ liệu hoàn tất. Tín hiệu reset LCD được tạo ra khi chân RST được kéo xuống mức thấp (RST=0). Khi đang truyền 8 bit dữ liệu (1 byte), nếu có tín hiệu reset LCD thì quá trình truyền sẽ bị hủy. Cho đến khi chân RST ở mức cao (RST=1), trong chu kì xung clock tiếp theo, quá trình truyền dữ liệu (của byte vừa bị hủy) sẽ được thực hiện lại.



Hình 3.47: Giản đồ xung chân RST

Các lệnh điều khiển LCD NOKIA 5110.

1. Lệnh **Function set**: Set các chế độ hoạt động cho LCD

D/C	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	1	0	0	PD	V	H

Hình 3.48: Các bit trong chân DC

- PD: là bit chọn chế độ hoạt động.
- PD=0: kích hoạt LCD hoạt động.
- PD=1: chế độ "power down".
- V: là bit chọn chiều tăng giá trị địa chỉ của bộ nhớ (DDRAM) của LCD.
- V=0: Giá trị của địa chỉ tăng theo chiều ngang.
- V=1: Giá trị của địa chỉ tăng theo chiều dọc.
- H: H=0: cho phép sử dụng các lệnh cơ bản.
- H=1: cho phép sử dụng thêm 1 số lệnh bổ sung.

Một số lệnh trong chế độ cho phép sử dụng các lệnh cơ bản (H=0).

## 2. Lệnh: Cài đặt hiển thị (Display Control).

D/C	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	1	D	0	E

Hình 3.49: Cấu hình cài đặt hiển thị

Ý nghĩa của các bit D, E:

Bảng 3.16: Ý nghĩa các bit D, E

D	E	Ý nghĩa
0	0	Không hiển thị
0	1	Hiển thị tất cả các điểm ảnh của màn hình LCD
1	0	Chế độ hiển thị thông thường
1	1	Chế độ hiển thị ngược (nền đen, chữ trắng)

## 3. Lệnh Set địa chỉ dòng (Set Y address).

D/C	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	1	0	0	0	Y2	Y1	Y0

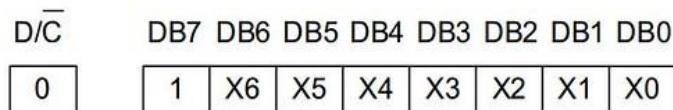
Hình 3.50: Cấu hình Set địa chỉ dòng

LCD Nokia 5110 có thể hiển thị 6 dòng (0 ->5), vì vậy chỉ cần 3 bit Y2, Y1, Y0, để chứa giá trị này

Y2	Y1	Y0	Position of Y-Address
0	0	0	Bank 0
0	0	1	Bank 1
0	1	0	Bank 2
0	1	1	Bank 3
1	0	0	Bank 4
1	0	1	Bank 5

Hình 3.51: Giá trị các bit set địa chỉ dòng

#### 4. Lệnh Set địa chỉ cột (Set X address).

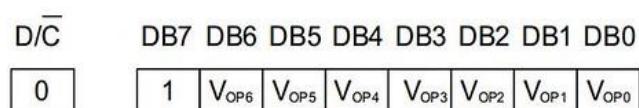


Hình 3.52: Cấu hình set địa chỉ cột

LCD Nokia 5110 có 84 cột (0 ->83), nên cần 7 bít (X0->X6 ) để chứa giá trị cột.

Một số lệnh trong chế độ cho phép sử dụng các lệnh bổ sung (H=1).

#### 5. Lệnh Set điện áp hoạt động cho LCD. (Set Vop)



Hình 3.53: Cấu hình set điện áp hoạt động

Người dùng có thể set các bit từ Vop6 ->Vop0 để chọn điện áp hoạt động cho LCD, dựa vào công thức sau.

$$V_{LCD} = a + (Vop6 ->Vop0) * b \quad PT\ 3.3$$

với giá trị của a, b là: a=3.06, b=0.06

Ví dụ: Để chọn điện áp hoạt động của LCD là 5V, ta làm như sau:

Theo công thức ta sẽ có:

$$5 = 3.06 + (Vop6 ->Vop0) * 0.06 \quad PT\ 3.4$$

từ đó suy ra: (Vop6 ->Vop0) = 32,33.

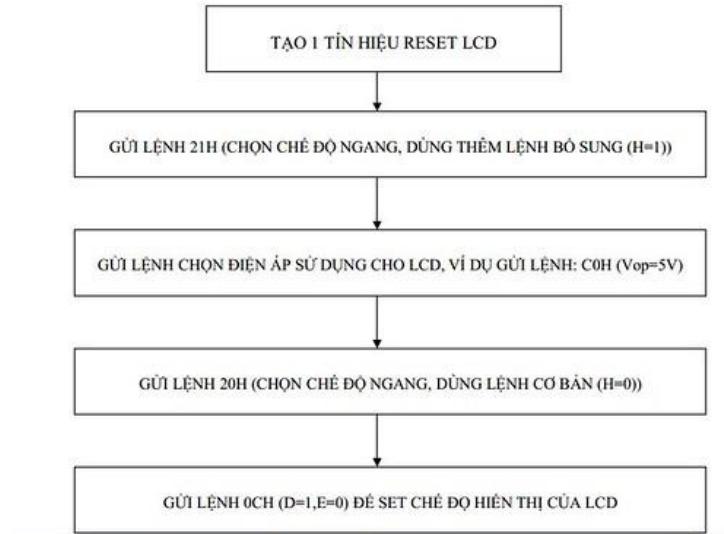
Ta sẽ chọn số nguyên là 32 hay bằng 20H=0100000B. Vậy, giá trị đưa vào LCD để set giá trị điện áp cho LCD là: 11000000B hay C0h.

Ngoài ra còn 1 số lệnh khác trong chế độ (H=1), các bạn có thể tham khảo thêm trên datasheet của LCD Nokia 5110.

#### Cài đặt (khởi tạo) cho LCD

Thường khi giao tiếp LCD với vi điều khiển, chúng ta có 1 hàm gọi là hàm "khởi tạo" cho LCD, và chúng ta gọi hàm này trước khi muốn hiển thị cái gì đó

ra LCD. Sau đây mình xin trình bày thứ tự các bước khởi tạo LCD Nokia 5110 một cách thông thường.



Hình 3.54: Các bước khởi tạo LCD N5110

Sau khi khởi tạo cho LCD thì chúng ta có thể làm việc với LCD.

### 3.5.5 Giao tiếp với khôi SIM808

Khôi SIM808 giao tiếp với vi điều khiển thông qua giao tiếp UART

Các tập lệnh AT để điều khiển SIM808 sẽ được gửi từ vi điều khiển tới SIM808 thông qua 2 chân TX, RX của UART1

Tập lệnh AT command là tập lệnh chuẩn được hỗ trợ bởi hầu hết các thiết bị di động như: điện thoại di động, GSM modem (SIM808 là một dạng GSM modem) có hỗ trợ gửi và nhận tin nhắn dưới dạng SMS và điều khiển các cuộc gọi.

AT là chữ viết tắt của ATtention. Tất cả các dòng lệnh bắt đầu với “AT” hoặc “at”. Đó là lý do tại sao các lệnh này được gọi là lệnh AT. Các khởi đầu “AT” là tiền tố thông báo cho modem về sự bắt đầu của một dòng lệnh. Các modem GSM/GPRS và các điện thoại di động còn được hỗ trợ bởi một bộ lệnh AT đặc biệt đối với công nghệ GSM.

Việc khởi tạo, kết nối GPRS và gửi dữ liệu lên Webserver được thực hiện qua các tập lệnh AT command dưới đây:

#### Bước 1: Khởi tạo SIM808

- Cài đặt tốc độ truyền có giá trị 115200:  
AT+IPR=115200
- Lưu việc cài đặt vào bộ nhớ:  
AT&W

#### Bước 2: Mở kết nối và cấu hình GPRS

AT+SAPBR=3,1,"Contype","GPRS"  
AT+SAPBR=3,1,"APN","CMNET"

AT+SAPBR=1,1

AT+SAPBR=2,1

Các câu lệnh này điều khiển SIM808 cấu hình GPRS để sẵn sàng kết nối với Server.

#### Bước 3: Kết nối tới Web Server

- Khởi tạo dịch vụ HTTP:  
AT+HTTPINIT
- Thiết lập các thông số HTTP:  
AT+HTTPPARA="CID",1  
AT+HTTPPARA="URL\",<http://110.76.86.170:9000/data>  
AT+HTTPPARA="CONTENT","application/json\".

Các tham số có thể được đặt bằng lệnh AT + HTTPPARA là: CID, URL, máy chủ proxy, cổng của máy chủ proxy HTTP, dạng dữ liệu gửi.

Ở đây:

- 110.76.86.170: chính là địa chỉ IP của Server
- 9000: là cổng server mở để nhận dữ liệu.
- Data: là file chứa dữ liệu gửi.
- Json: là dạng dữ liệu gửi.

#### Bước 4: Gửi dữ liệu lên Web Server

- Bắt đầu việc gửi dữ liệu lên Web server:  
AT+HTTPDATA=<size><time>

Với:

- Size: là kích thước của dữ liệu
- Time: là thời gian tối đa chờ gửi dữ liệu
- Gửi chuỗi json lên Webserver chứa các thông số của thiết bị:  
{ "deviceId": a, "timeStamp": b, "temperature": temp\_value, "humidity": hum\_value, "pm25": pm2.5\_value, "pm10": pm10\_value, "act1State": g, "act2State": h }

Phần nội dung của chuỗi json được gửi lên em sẽ trình bày trong chương 4.

- Lựa chọn phương thức gửi:  
AT+HTTPACTION=<method>

Với mỗi giá trị method khác nhau thì sẽ thực hiện các phương thức khác nhau:

- 0: GET: Nhận dữ liệu
- 1: POST: Gửi dữ liệu
- 3: DELETE: Xóa dữ liệu

Ở đây em chọn method là 1 nghĩa là chọn phương thức gửi dữ liệu.

## CHƯƠNG 4. THIẾT KẾ WEB SERVER VÀ GIAO DIỆN NGƯỜI DÙNG

### 4.1 Web Server

Web server dịch ra tiếng Việt nghĩa là máy chủ. Web server là máy tính lớn được kết nối với tập hợp mạng máy tính mở rộng. Đây là một dạng máy chủ trên internet mỗi máy chủ là một IP khác nhau và có thể đọc các ngôn ngữ như file \*.htm và \*.html... Tóm lại máy chủ là kho để chứa toàn bộ dữ liệu hoạt động trên internet mà nó được giao quyền quản lý.

Web server phải là một máy tính có dung lượng lớn, tốc độ rất cao để có thể lưu trữ vận hành tốt một kho dữ liệu trên internet. Nó sẽ điều hành trọn chu cho một hệ thống máy tính hoạt động trên internet, thông qua các cổng giao tiếp riêng biệt của mỗi máy chủ. Các web server này phải đảm bảo hoạt động liên tục không ngừng nghỉ để duy trì cung cấp dữ liệu cho mạng lưới máy tính của mình. Để hiểu hơn web server chính là máy chủ, được thiết kế với các siêu tính năng dùng để chứa các dữ liệu cho một phần mạng lưới máy tính trên internet. Tất cả những hoạt động dịch vụ trên internet nào đều phải có máy chủ này mới hoạt động được.

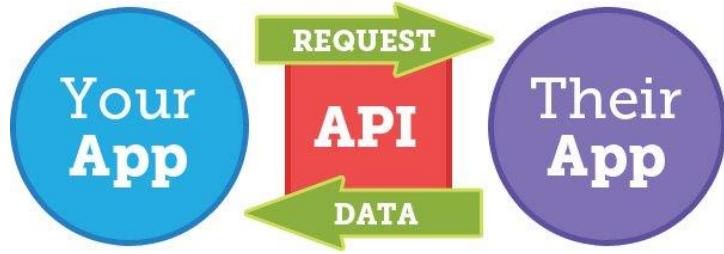
- Web server có thể xử lý dữ liệu và cung cấp thông tin đến máy khách thông qua các máy tính cá nhân trên môi trường Internet qua giao thức HTTP, giao thức được thiết kế để gửi các file đến trình duyệt Web, và các giao thức khác. (Ví dụ: khi các bạn truy cập vào trang web vinahost.vn máy chủ sẽ cung cấp đến các bạn tất cả dữ liệu về trang web đó thông qua lệnh giao tiếp)
- Máy tính nào cũng có thể là một máy chủ nếu cài đặt lên nó một chương trình phần mềm Server Software và sau đó kết nối vào Internet.
- Phần mềm Web Server Software cũng giống như các phần mềm khác, nó dùng để cài đặt và chạy trên bất kỳ máy tính nào đáp ứng đủ yêu cầu về bộ nhớ. Nhờ có chương trình này mà người sử dụng có thể truy cập đến các thông tin của trang Web từ một máy tính khác ở trên mạng
- Khi là SEO chúng ta thường gặp các máy chủ nhỏ, máy chủ ảo và thông thường chúng ta hay thuê máy chủ dạng VPS hay Hosting để lưu giữ liệu trang web của mình.

### 4.2 Giới thiệu về REST Server

#### 4.2.1 API

API là viết tắt của Application Programming Interface (Giao diện lập trình ứng dụng), một phần mềm trung gian cho phép hai ứng dụng nói chuyện với nhau. Mỗi khi sử dụng một ứng dụng như Facebook, gửi tin nhắn tức thì hoặc kiểm tra thời tiết trên điện thoại, có nghĩa là ta đang sử dụng API.

Khi sử dụng một ứng dụng trên điện thoại di động, ứng dụng kết nối Internet và gửi dữ liệu tới máy chủ. Sau đó, máy chủ lấy ra dữ liệu đó, diễn giải nó, thực hiện các hành động cần thiết và gửi nó trở lại điện thoại. Ứng dụng sau đó sẽ diễn giải dữ liệu đó và trình bày thông tin ta muốn theo cách có thể đọc được.



Hình 4.1: Cách thức API hoạt động

Ví dụ, trong một nhà hàng, khách hàng gọi món ăn và nhà bếp là một phần “hệ thống” chuẩn bị thực đơn. Cái còn thiếu ở đây là liên kết quan trọng để truyền đạt yêu cầu của khách hàng tới nhà bếp và chuyển đồ ăn tới bàn cho họ. Liên kết quan trọng này chính là bồi bàn hoặc API. Người bồi bàn - API là người đưa tin, sẽ nhận yêu cầu gọi món và truyền đạt lại tới nhà bếp - hệ thống. Sau khi thức ăn đã sẵn sàng, bồi bàn sẽ chuyển nó tới khách hàng.

#### 4.2.2 Chuỗi Json

JSON là chữ viết tắt của Javascript Object Notation, đây là một dạng dữ liệu tuân theo một quy luật nhất định mà hầu hết các ngôn ngữ lập trình hiện nay đều có thể đọc được, ta có thể sử dụng lưu nó vào một file, một record trong CSDL rất dễ dàng. JSON có định dạng đơn giản, dễ dàng sử dụng và truy vấn hơn XML rất nhiều nên tính ứng dụng của nó hiện nay rất là phổ biến.

Cú pháp của JSON rất đơn giản là mỗi thông tin dữ liệu sẽ có 2 phần đó là key và value, điều này tương ứng trong CSDL là tên field và giá trị của nó ở một record nào đó.

Chuỗi JSON được bao lại bởi dấu ngoặc nhọn {}

Các key, value của JSON bắt buộc phải đặt trong dấu nháy kép " ".

Nếu có nhiều dữ liệu (nhiều cặp key => value) thì ta dùng dấu phẩy (,) để ngăn cách

Trong đồ án này, chuỗi dữ liệu được gửi lên từ Sensor Node đến Server có cấu trúc là mỗi chuỗi JSON. Nội dung chuỗi dữ liệu em sẽ trình bày trong phần triển khai thiết kế Web server.

#### 4.2.3 Giới thiệu về RESTful API

REST (REpresentational State Transfer) là một dạng chuyển đổi cấu trúc dữ liệu, một kiểu kiến trúc để viết API. Nó sử dụng phương thức HTTP đơn giản để tạo cho giao tiếp C. Vì vậy, thay vì sử dụng một URL cho việc xử lý một số thông tin người dùng, REST gửi một yêu cầu HTTP như GET, POST, DELETE, vv đến một URL để xử lý dữ liệu.

API RESTful là một tiêu chuẩn dùng trong việc thiết kế các API cho các ứng dụng web để quản lý các resource. RESTful là một trong những kiểu thiết kế API được sử dụng phổ biến ngày nay để cho các ứng dụng (web, mobile...) khác nhau giao tiếp với nhau.

Chức năng quan trọng nhất của REST là quy định cách sử dụng các phương pháp HTTP (như GET, POST, PUT, DELETE...) và cách định dạng các URL

cho ứng dụng web để quản các resource. RESTful không quy định logic code ứng dụng và không giới hạn bởi ngôn ngữ lập trình ứng dụng, bất kỳ ngôn ngữ hoặc framework nào cũng có thể sử dụng để thiết kế một RESTful API.

#### 4.2.4 Cấu trúc của REST server

Trên cơ sở cấu trúc của hệ thống, em sẽ xây dựng một REST server bao gồm 3 phần:

- *Máy chủ REST:*

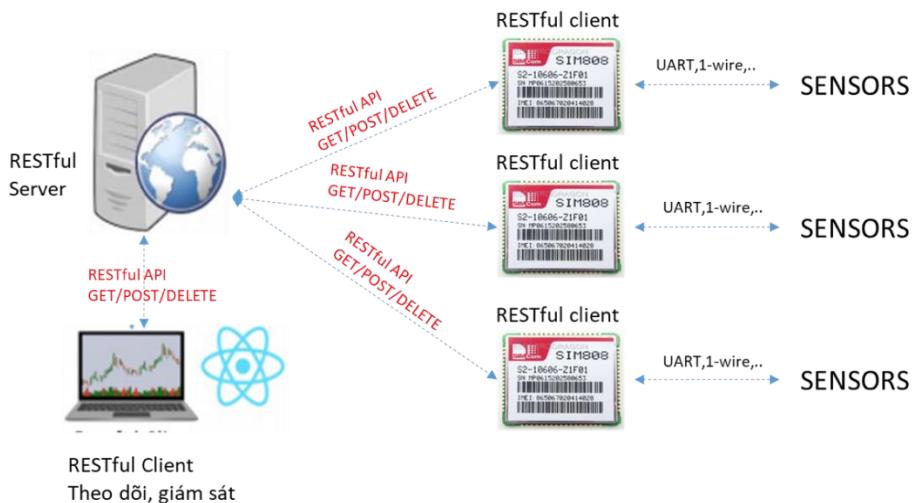
Máy chủ REST là thành phần quan trọng nhất của hệ thống. Về mặt chức năng, nó chịu trách nhiệm xử lý và lưu trữ dữ liệu của toàn bộ hệ thống. Bằng cách cung cấp RESTful API, nó cho phép các thành phần khác của hệ thống có thể dễ dàng tương tác và chia sẻ dữ liệu mà không cần quan tâm đến sự khác biệt về kiến trúc phần cứng cũng như hệ điều hành.

- *Giao diện người dùng:*

Giao diện người dùng (User Interface) được thiết kế để giúp người dùng dễ dàng tương tác với máy chủ REST. Dữ liệu của từng khu vực sẽ được hiển thị trực quan dưới các hình thức rõ ràng nhất. Người dùng cũng có thể vận hành và quản lý toàn bộ hệ thống thông qua giao diện người dùng này.

- *Máy khách REST:*

Các máy khách RESTful, cũng có thể được gọi là các RESTful client, được đặt ở các cấp trường. Trong đồ án này nó chính là các Sensor Node. Nó được tích hợp với các cảm biến cho phép thu thập dữ liệu và gửi dữ liệu lên RESTful Server



Hình 4.2: Hệ thống giám sát chất lượng không khí dựa trên RESTful

### 4.3 Giới thiệu về Node.js, Express, ReactJS

Một web server được xây dựng sẽ bao gồm 2 thành phần:

- BackEnd (REST server) có nhiệm vụ nhận dữ liệu từ các Sensor node cũng như thực hiện các yêu cầu của người sử dụng.
- FrontEnd (giao diện web) có nhiệm vụ hiển thị hệ thống đo và giám sát cũng như nhận các yêu cầu của người sử dụng.

Ở trong đồ án này, em sử dụng Nodejs và Express để xây dựng phần BackEnd và Reactjs để xây dựng FrontEnd.

#### 4.3.1 Node.js

Node (hoặc chính thức hơn là Node.js) là một môi trường thời gian chạy đa nền tảng, mã nguồn mở cho phép các nhà phát triển tạo ra tất cả các loại công cụ và ứng dụng phía máy chủ trong JavaScript. Thời gian chạy được thiết kế để sử dụng bên ngoài bối cảnh trình duyệt (nghĩa là chạy trực tiếp trên hệ điều hành máy tính hoặc máy chủ). Như vậy, môi trường bỏ qua các API JavaScript dành riêng cho trình duyệt và thêm hỗ trợ cho các API hệ điều hành truyền thống hơn bao gồm HTTP và các thư viện hệ thống tệp.

Từ quan điểm phát triển máy chủ web, Node có những ưu điểm sau:

- Node được thiết kế để tối ưu hóa thông lượng và khả năng mở rộng trong các ứng dụng web và là một giải pháp tốt cho nhiều vấn đề phát triển web phổ biến (ví dụ: các ứng dụng web thời gian thực).
- Mã code được viết bằng JavaScript đơn giản, nghĩa là sử dụng ít thời gian hơn để xử lý thay đổi các ngôn ngữ lập trình khi viết bằng cả mã phía máy khách và mã phía máy chủ.
- JavaScript là ngôn ngữ lập trình tương đối mới và được hưởng lợi từ các cải tiến trong thiết kế ngôn ngữ khi so sánh với các ngôn ngữ máy chủ web truyền thống khác (ví dụ: Python, PHP, v.v.) Nhiều ngôn ngữ mới và phổ biến khác biên dịch / chuyển đổi thành JavaScript để bạn cũng có thể sử dụng TypeScript, CoffeeScript, ClojureScript, Scala, LiveScript, v.v ...
- Node.js có thể tùy biến. Nó có sẵn trên Microsoft Windows, macOS, Linux, Solaris, FreeBSD, OpenBSD, WebOS và hệ điều hành NonStop. Hơn nữa, nó được hỗ trợ tốt bởi nhiều nhà cung cấp dịch vụ lưu trữ web, thường cung cấp cơ sở hạ tầng và tài liệu cụ thể để lưu trữ các trang web Node.
- Nó có một hệ sinh thái và cộng đồng nhà phát triển bên thứ ba rất tích cực, với rất nhiều người sẵn sàng giúp đỡ.

#### 4.3.2 Express

Express là khung web Node phổ biến nhất và là thư viện cơ bản cho một số khung web Node phổ biến khác. Nó cung cấp các cơ chế để:

- Viết trình xử lý cho các yêu cầu có các phương thức HTTP khác nhau tại các đường dẫn URL khác nhau.
- Tích hợp với các công cụ hiển thị chế độ xem của Wap để tạo phản hồi bằng cách chèn dữ liệu vào các mẫu.
- Đặt các cài đặt ứng dụng web phổ biến như cổng để sử dụng để kết nối và vị trí của các mẫu được sử dụng để hiển thị phản hồi.
- Thêm xử lý yêu cầu bổ sung, trung gian trực tuyến, bắt cứ lúc nào trong đường dẫn xử lý yêu cầu. Mặc dù bản thân Express khá tối giản, các nhà phát triển đã tạo ra các gói phần mềm trung gian tương thích để giải quyết hầu hết mọi vấn đề phát triển web. Có thư viện để làm việc với cookie,

phiên, đăng nhập người dùng, tham số URL, dữ liệu POST, tiêu đề bảo mật và nhiều hơn nữa. Có thể tìm thấy danh sách các gói phần mềm trung gian được duy trì bởi nhóm Express tại Express Middleware (cùng với danh sách một số gói bên thứ 3 phổ biến).

### 4.3.3 ReactJS

React JS là một thư viện Javascript, mã nguồn mở, được sử dụng để xây dựng giao diện người dùng (UI) dành riêng cho các ứng dụng một trang. Nó là một trong những thư viện Javascript phổ biến nhất được sử dụng để xây dựng ứng dụng front-end ngay bây giờ. Các nhà phát triển Facebook đã tạo ra ReactJS vào năm 2011 và được sử dụng đầu tiên trong ứng dụng Facebook và cho đến ngày hôm nay, nó có một cộng đồng lớn hỗ trợ nó và rất nhiều tài nguyên để tìm hiểu nó.

Javascript là ngôn ngữ lập trình được sử dụng để tạo các yếu tố động trên trang web hoặc ứng dụng web. Hầu như tất cả các trình duyệt hiện đại đều hỗ trợ Javascript. Vào bất kỳ trang web nào, nó có rất nhiều tính năng được viết bằng Javascript, như xác thực mẫu, hoạt hình nâng cao, cửa sổ bật lên, v.v. Javascript được sử dụng để xây dựng các ứng dụng web và trang với kịch bản phía máy khách.

Thư viện JS là một mã Javascript được viết sẵn giúp cho việc phát triển ứng dụng dễ dàng và nhanh hơn. Nó bao gồm các thành phần, chức năng và các mẫu có sẵn, có thể được sử dụng để thực hiện các nhiệm vụ lập trình cụ thể. Trong nhiều trường hợp, nó cũng giúp giữ cho các ứng dụng an toàn hơn và hiệu quả hơn. Các khung Javascript được sử dụng để giảm số lượng mã hóa và giảm chi phí phát triển ứng dụng, vì các thư viện chủ yếu là nguồn mở và miễn phí.

#### Các tính năng của ReactJS

- DOM ảo. Nếu sử dụng Javascript thuần túy, đối tượng DOM sẽ kết xuất lại mỗi khi có bất kỳ thay đổi nào được thực hiện trong các phần tử HTML. Nó tốt nếu bạn có một trang web tĩnh, nơi không có quá nhiều xảy ra; hiệu suất là an toàn. Nhưng trong trường hợp các ứng dụng web động có nhiều yếu tố tương tác với người dùng, thì nó không hoạt động tốt như vậy. Hiệu suất của ứng dụng đi xuống đáng kể. Những người tạo ra ReactJS đã quyết định xử lý vấn đề này và họ đã tạo ra một DOM ảo. Khi các thay đổi được thực hiện trong DOM, ReactJS tạo một bản sao, được gọi là Virtual DOM. Bản sao này được so sánh với 8 DOM bình thường và chỉ có phần tử khác biệt được hiển thị lại. Nó tốn ít năng lực tính toán và thời gian tải, đó là lý do tại sao nó cải tiến tốt.
- JSX. Trong ReactJS, JSX được sử dụng để tạo mẫu thay vì HTML. JSX là một loại phần mở rộng của Javascript và nó cho phép chúng ta sử dụng các thẻ HTML bên trong mã Javascript. Nó được sử dụng để tạo các mẫu trong ReactJS. ReactJS là một công nghệ rất phổ biến được sử dụng không chỉ để tạo ra ý tưởng khởi nghiệp của bạn, mà cả các công ty lớn cũng sử dụng công nghệ này cho các dự án lớn.

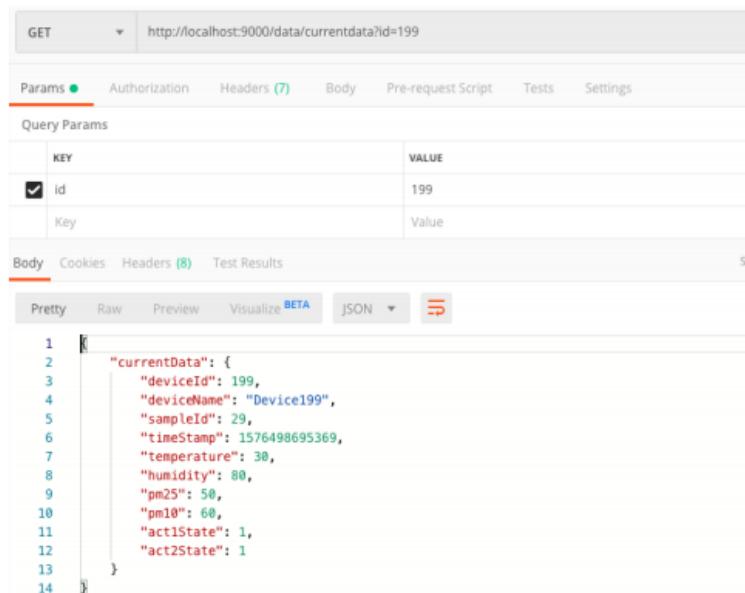
Hiện tại, có nhiều ứng dụng phổ biến được tạo bằng ReactJS:

- Facebook
- Instagram
- Netflix
- New York Times
- Dropbox
- Yahoo!

#### 4.4 Thiết kế các RESTful API

Như đã đề cập, máy chủ REST đóng vai trò quan trọng nhất trong hệ thống. Về cơ bản, nó cung cấp dịch vụ API cho các thành phần khác. Nói cách khác, các Sensor Node và giao diện người dùng web giao tiếp và chia sẻ dữ liệu với máy chủ REST bằng cách sử dụng RESTful API. Thật vậy, để tạo thuận lợi cho việc giao tiếp giữa các thành phần, các RESTful API phải được thiết kế rõ ràng dễ sử dụng. Do đó, em đã chia các API thành hai nhóm là RESTful API cho giao diện người dùng và RESTful API cho các thiết bị Sensor node.

##### 4.4.1 RESTful API cho giao diện người dùng



Hình 4.3: Nhận dữ liệu hiện tại của một vị trí theo ID

API trong Hình 4.3 lấy id làm tham số truy vấn, id này được đặt duy nhất cho một thiết bị ở vị trí tương ứng. Phản hồi từ máy chủ được hình thành theo một chuỗi json, trong đó có:

- deviceId: là số nhận dạng của thiết bị, nó là duy nhất trong số tất cả các thiết bị Sensor Node
- deviceName: là tên của thiết bị, được gắn vs Id của thiết bị
- sampleId: mỗi thiết bị được xác định bởi deviceId, mỗi mẫu được cung cấp bởi một Sensor Node cũng được nhận dạng bằng một số gọi là số nhận dạng mẫu (sampleId)

- Timestamp: chứa thông tin về thời gian, nó chỉ định khi dữ liệu được ghi lại
- Nhiệt độ, độ ẩm, pm25 và pm10: là các thông số môi trường được thu thập bởi Sensor Node.
- act1State và act2State: cho biết trạng thái của các bộ truyền động được tích hợp trong Sensor Node. Giá trị 1 có nghĩa là bộ truyền động đang hoạt động.

The screenshot shows a Postman interface with a GET request to `http://localhost:9000/data/chart?id=100&type=0&start=1575728492010&stop=1575728506715`. The 'Params' tab is selected, displaying four parameters: id (100), type (0), start (1575728492010), and stop (1575728506715). Below the table, there are tabs for Body, Cookies, Headers, and Test Results. The Body tab is selected, showing a JSON response with the following content:

```

1  {
2     "extractedData": {
3         "labels": [
4             1575728492010,
5             1575728496188,
6             1575728500199,
7             1575728506715
8         ],
9         "values": [
10            30,
11            35,
12            34,
13            33
14        ]
15    }
16 }

```

Hình 4.4: Nhận dữ liệu của một ID trong một khoảng thời gian

API trong Hình 4.4 trả về dữ liệu của một id (bao gồm cả thời gian được ghi lại) của thiết bị trong một khoảng thời gian được chỉ định dưới dạng chuỗi json. API này lấy ra một cách đơn giản:

- id: là số nhận dạng của thiết bị
- type: là loại tham số, giá trị của phạm vi loại phạm vi từ 0 đến 3 tương ứng với nhiệt độ, độ ẩm, nồng độ pm2,5 và nồng độ pm10 tương ứng.
- start và stop: biểu thị cho thời gian bắt đầu và thời gian dừng của giai đoạn được lấy.

API trong Hình 4.5 được sử dụng để nhận tất cả dữ liệu được lưu trữ trong máy chủ REST. Khi yêu cầu được gửi đến máy chủ, nó sẽ ngay lập tức phản hồi một chuỗi json bao gồm tất cả dữ liệu có trong nó.

```

1  [
2   "allData": [
3     {
4       "deviceId": 199,
5       "deviceName": "Device199",
6       "sampleId": 0,
7       "timeStamp": 1576573144020,
8       "temperature": 30,
9       "humidity": 80,
10      "pm25": 50,
11      "pm10": 60,
12      "act1State": 1,
13      "act2State": 1
14    },
15    {
16      "deviceId": 199,
17      "deviceName": "Device199",
18      "sampleId": 1,
19      "timeStamp": 1576573146191,
...

```

Hình 4.5: Nhận tất cả dữ liệu của một vị trí theo ID

#### 4.4.2 RESTful API cho các Sensor Node (RESTful client)

Chỉ có một API được thiết kế cho Sensor Node. Nó cho phép các Sensor Node gửi dữ liệu được thu thập lên máy chủ RESTful theo thời gian thực. API này dựa trên phương thức POST trong khi các API trên dựa trên phương thức GET. Như trong Hình 4.6, phần thân của yêu cầu bao gồm:

- deviceId: chỉ định số nhận dạng của thiết bị
- timestamp: chỉ định thời gian ghi
- nhiệt độ, độ ẩm, pm25 và pm10: là các tham số được thu thập bởi các cảm biến
- act1State và act2State: là trạng thái của bộ truyền động

```

POST http://110.76.86.170:9000/data/ Send
Params Authorization Headers (1) Body Pre-request Script Tests Settings C
none form-data x-www-form-urlencoded raw binary GraphQL BETA JSON
1  [
2   "deviceId": 200,
3   "timeStamp": 231095,
4   "temperature": 22,
5   "humidity": 84,
6   "pm25": 101,
7   "pm10": 162,
8   "act1State": 1,
9   "act2State": 1
10  ]

```

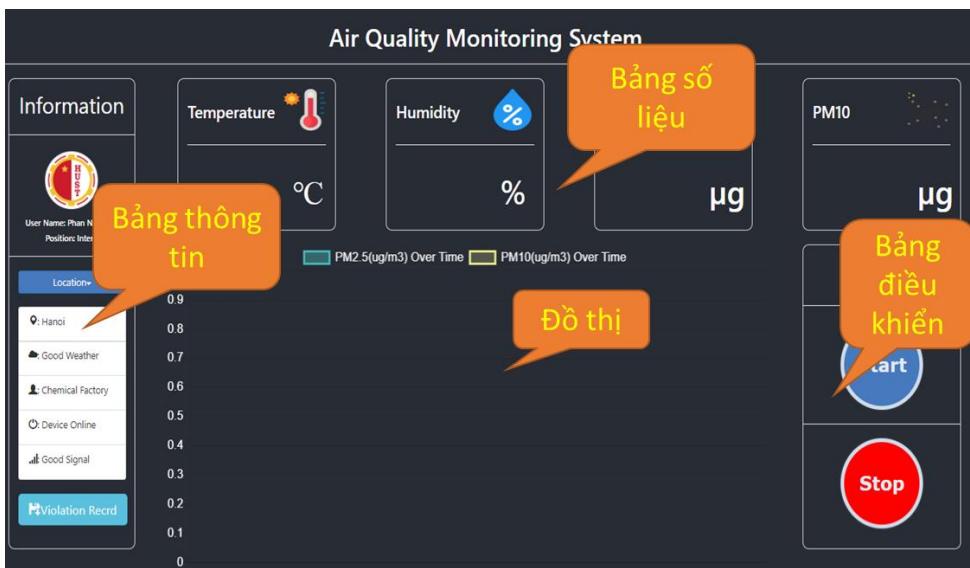
Hình 4.6: Dữ liệu được gửi từ Sensor Node theo ID

Chuỗi dữ liệu này chính là chuỗi dữ liệu được gửi từ SIM808 mà em đã trình bày ở cuối Chương 3.

#### 4.5 Triển khai giao diện người dùng

Để thiết kế giao diện người dùng, ReactJS được sử dụng. Khung này tạo điều kiện cho quá trình thiết kế bằng cách cung cấp các thành phần dựa trên cơ chế. Trên cơ sở của cơ chế này, giao diện người dùng web được nhắm mục tiêu chia thành các thành phần chính. Các thành phần là bảng dữ liệu, biểu đồ, bảng điều khiển

và bảng thông tin. Cấu trúc của giao diện người dùng được minh họa trong Hình 4.7.



Hình 4.7: Giao diện hiển thị

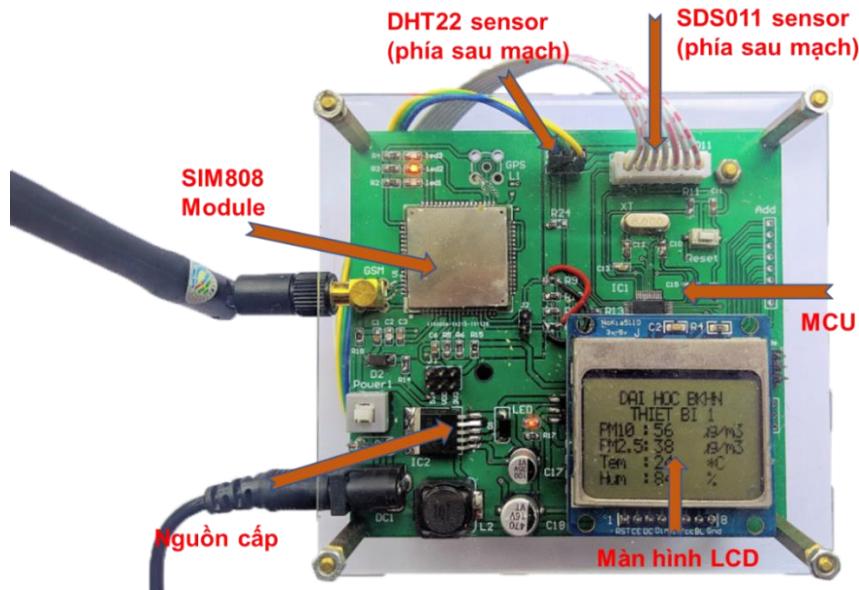
- Thành phần bảng thông tin đặt ở bên phải của giao diện cung cấp các thông tin về thiết bị đang được hiển thị kết quả.
- Thành phần bảng số liệu được đặt ở trên cùng của giao diện người dùng, được sử dụng để chỉ ra các giá trị của các tham số được thu thập bởi Sensor Node. Hiện tại, nó chỉ hiển thị bốn thông số bao gồm nhiệt độ, độ ẩm, nồng độ. Trên thực tế, Sensor Node có thể được tích hợp với nhiều loại cảm biến, vì vậy giao diện người dùng có thể được sửa đổi theo yêu cầu thỏa mãn hiển thị tất cả các tham số có sẵn.
- Thành phần đồ thị được thiết kế để cung cấp cho người dùng một công cụ mạnh mẽ để quan sát sự thay đổi của nồng độ bụi
- Thành phần bảng điều khiển được đặt ở bên phải của giao diện người dùng, nó có hai nút cho phép người dùng bắt đầu hoặc dừng thu thập mẫu không khí từ xa (Chức năng này hiện tại chưa được áp dụng trong đồ án).

## CHƯƠNG 5. KẾT QUẢ VÀ ĐÁNH GIÁ HỆ THỐNG

### 5.1 Kết quả đạt được

#### 5.1.1 Mạch phần cứng thiết bị đo nồng độ bụi

Mạch phần cứng được thể hiện ở hình dưới đây:

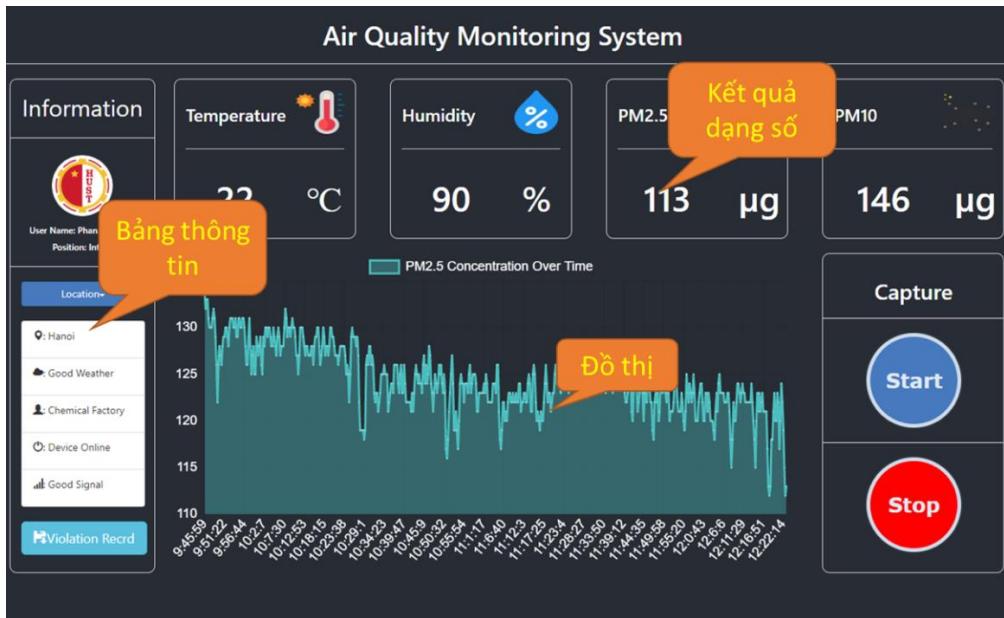


Hình 5.1: Mạch phần cứng thiết bị đo

Mạch đã được thử nghiệm đo nồng độ bụi PM10, PM2.5 trên thành phố Hà Nội. Mạch đã chạy ổn định trong thời gian dài, đúng theo những thông số thiết kế. Ngoài các số liệu về bụi, mạch cũng cho phép đo các thông số khác là nhiệt độ và độ ẩm môi trường xung quanh. Các số liệu được hiển thị trên LCD trên mạch cũng như gửi lên Webserver thông qua truyền thông không dây GPRS.

#### 5.1.2 Web server

Web server đã được thiết kế để nhận và lưu trữ dữ liệu đo từ các Sensor Node, cho phép người dùng truy cập giao diện người dùng Web. Người dùng có thể trực tiếp truy cập vào trang web [airqualitybka.xyz](http://airqualitybka.xyz) để theo dõi cũng như giám sát các dữ liệu từ các thiết bị Sensor Node. Web Server hoạt động ổn định, không xảy ra việc mất mát hay hiển thị sai dữ liệu.



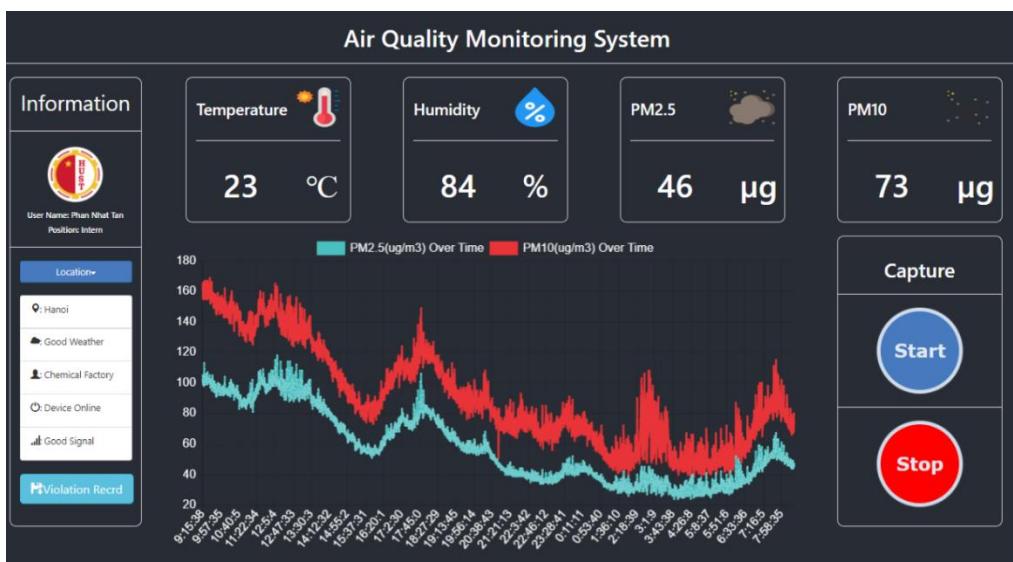
Hình 5.2: Giao diện hiển thị Web

## 5.2 Thử nghiệm và đánh giá

### 5.2.1 Thử nghiệm

*Kịch bản thử nghiệm:*

Thiết bị Sensor Node đã được đặt tòa nhà HiTech - Trường đại học Bách Khoa Hà Nội từ 9h sáng ngày 02/01/2020 đến 9h sáng ngày 03/01/2020.

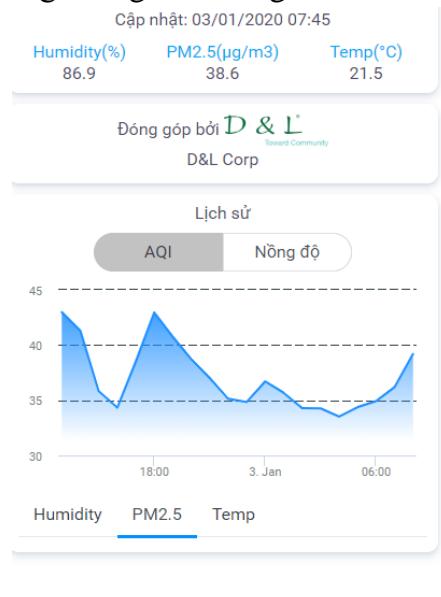


Hình 5.3: Kết quả đo tại tòa nhà HiTech

*Kết quả cho thấy:*

- Diễn biến nồng độ bụi thay đổi theo khoảng thời gian trong ngày
- Nồng độ bụi tăng cao vào các giờ cao điểm do lượng phương tiện giao thông tăng lên và giảm xuống thấp vào thời gian ban đêm khi lượng phương tiện giao thông ít đi.

- Đồ thị diễn biến nồng độ bụi khá tương đồng so với dữ liệu từ ứng dụng PAM AIR với trạm quan trắc đặt tại đường Bà Triệu, Hai Bà Trung, Hà Nội đo cùng thời gian thử nghiệm.



Hình 5.4: Diễn biến nồng độ bụi trên ứng dụng PAM AIR

### 5.2.2 Đánh giá và thảo luận

Dựa trên những kết quả đo thử nghiệm, bước đầu có thể đánh giá:

- Hệ thống hoạt động ổn định trong thời gian dài mà không xảy ra lỗi.
- Các thông số đo được cập nhật tới giao diện giám sát đầy đủ, không xảy ra tình trạng mất số liệu.
- So sánh kết quả đo với các trạm quan trắc trong thành phố Hà Nội cho kết quả tương đồng, sai lệch không nhiều.

Bên cạnh những ưu điểm, hệ thống vẫn còn tồn tại một số nhược điểm như:

- Các Sensor Node vẫn sử dụng nguồn điện lướt, chưa thể mang trực tiếp ra ngoài
- Giao diện hiển thị còn đơn giản chưa thể hiện rõ được các mức độ ô nhiễm tại các điểm đo
- Mới chỉ đo được các thông số về PM2.5, PM10, nhiệt độ, độ ẩm không khí, cần tích hợp thêm các thông số môi trường khác để tạo thành một trạm quan trắc hoàn chỉnh.

## KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN ĐỀ TÀI

Với đề tài “*Thiết kế chế tạo hệ đo và giám sát nồng độ bụi mịn PM10, PM2.5*”, em đã thực hiện được những công việc sau:

- Đã tìm hiểu được ảnh hưởng của bụi đối với con người và môi trường và các phương pháp đo nồng độ bụi.
- Thiết kế và thực hiện thành công phần cứng thiết bị đo nồng độ bụi có tích hợp truyền thông không dây. Thiết bị này hoạt động dựa trên vi điều khiển STM32 kết hợp với phần tử đo nồng độ bụi là cảm biến SDS011.
- Lập trình được phần mềm cho vi điều khiển với các chức năng, đọc số liệu từ các cảm biến đo (đo nồng độ bụi, đo nhiệt độ, độ ẩm), xử lý các số liệu đo, thực hiện thuật toán truyền tin lên Web giám sát.
- Lập trình được phần mềm thu thập và giám sát trên Web. Phần mềm này được phát triển dựa trên ngôn ngữ JavaScript hiển thị kết quả đo dưới dạng số, đồ thị, có thể hiển thị số liệu của nhiều thiết bị đo cùng lúc.

Bước đầu cho thấy hệ thống hoạt động tương đối ổn định, thực hiện đúng các chức năng theo yêu cầu đặt ra. Nếu điều kiện cho phép, trong thời gian tới, em mong muốn sẽ tiếp tục phát triển thêm các phần sau:

- Tích hợp thêm các cảm biến đo nồng độ các khí khác như CO, NO, NO2 để giám sát chất lượng không khí một cách đầy đủ hơn
- Nghiên cứu, cải tiến các thiết bị với khả năng tự chủ năng lượng.
- Thêm các tính năng trên giao diện Web như bản đồ các thiết bị, các mức cảnh báo đối với người dùng.

Trên cơ sở những kết quả đã đạt được trong khuôn khổ đề án này, em rất hy vọng trong tương lai, hệ thống có thể được tiếp tục cải tiến, hoàn thiện, và ứng dụng vào thực tiễn để góp phần cải thiện tình trạng ô nhiễm không khí như hiện nay.

## TÀI LIỆU THAM KHẢO

- [1] Z. K. A. H. T. Markus Amann, "Future air quality in Ha Noi," in *VAST-IIASA study 2018*, 2018.
- [2] QCVN 06: 2009/BTNMT.
- [3] "Trang web của văn phòng bảo vệ môi trường Hoa Kỳ chứa các tiêu chuẩn và hướng dẫn đo đặc các thông số môi trường," [Online]. Available: <https://www.epa.gov/>.
- [4] "nosewash," [Online]. Available: <https://nosewash.rohto.com.vn/cam-nang-mui-xoang/tin-tuc-xa-hoi/bui-min-pm2.5-va-pm10-sat-thu-vo-hinh-cua-loai-nguo-1.html>.
- [5] waqi. [Online]. Available: [waqi.info](http://waqi.info).
- [6] N. V. Long, "Nghiên cứu thiết kế hệ thống đo ô nhiễm không khí," Hà Nội, 2018.
- [7] STMicroelectronic, Reference Manual of STM32F103xx.
- [8] Inovafitness, Datasheet of SDS011 sensor.
- [9] Aosong, Datasheet DHT22.
- [10] Nokia, Datasheet LCD N5110.
- [11] SIMCOM, Datasheet of SIM808.
- [12] T. Instruments, LM2576 Datasheet.
- [13] A. M. Systems, AMS 1117 Datasheet.
- [14] Nosewash, "Bụi mịn PM2.5 và PM10 - Sát thủ vô hình của loài người".
- [15] B. G. R. M. O. R. H. Michaël Canu, "Understanding the Shinyei PPD24NS low-cost dust sensor," in *IEEE International Conference on Environmental Engineering*, At Milan, Italy, 2018.

## PHỤ LỤC

### A1. Code trên vi điều khiển

```
#include "stm32f10x.h"
#include "stm32f10x_rcc.h"
#include "stm32f10x_gpio.h"
#include "stm32f10x_usart.h"
#include "misc.h"
#include "stdio.h"
#include "string.h"
#include "stdlib.h"
#include "stm32f10x_tim.h"
#include "stm32f10x_flash.h"

typedef struct dht_sensor dht_sensor;
typedef struct sds_sensor sds_sensor;
typedef struct dusty_sensor dusty_sensor;

struct dht_sensor{
    float temperature;
    float humi;
};

struct sds_sensor{
    uint16_t pm25_value ;
    uint16_t pm10_value;
};

struct dusty_sensor{
    dht_sensor p_dht_ss;
    sds_sensor p_sds_ss;
};

volatile uint32_t usTicks;
volatile uint8_t sds_timeout;

uint8_t DHT_GetTemHumi (dht_sensor* p_sensor);

unsigned char N5110_buffer[X_max][Rows];
volatile uint16_t data = 0;
volatile uint8_t sds10_data[10] = {0};
volatile uint8_t counter_byte = 0;
volatile uint8_t received_flag = 0;
dusty_sensor p_dusty_sensor;

static void delay_ms(uint32_t ms);
```

```

static void delay_us(uint32_t us);
static void integer_to_string(uint32_t x, uint8_t* s);

void update_data(dusty_sensor* p_ss){

    char s1[10],s2[20],s3[10],s4[10];

    integer_to_string((uint32_t)p_ss->p_dht_ss.temperature,&s1[0]);
    integer_to_string((uint32_t)p_ss->p_dht_ss.humi,&s2[0]);
    integer_to_string(p_ss->p_sds_ss.pm10_value,&s3[0]);
    integer_to_string(p_ss->p_sds_ss.pm25_value,&s4[0]);

    sim808_usart_send_string("AT+HTTPDATA=200,10000");
    sim808_usart_send_string("\r\n");
    delay_ms(1000);
    sim808_usart_send_string("{  \"deviceId\": 199,  \"timeStamp\":b,
\"temperature\": ");
        sim808_usart_send_string(s1);
        sim808_usart_send_string(", \"humidity\": ");
        sim808_usart_send_string(s2);
        sim808_usart_send_string(", \"pm25\": ");
        sim808_usart_send_string(s4);
        sim808_usart_send_string(", \"pm10\":\"");
        sim808_usart_send_string(s3);

    sim808_usart_send_string(", \"act1State\": 1, \"act2State\": 1 }");
        sim808_usart_send_string("\r\n");
        delay_ms(12000);
        sim808_usart_send_string("AT+HTTPACTION=1");
        sim808_usart_send_string("\r\n");
        delay_ms(3000);
    }

void sds_update_data(sds_sensor* p_sensor)
{
    sim808_send_data_to_server(p_sensor->pm10_value,p_sensor->pm25_value);
}

void SDS_init(void)
{
    SDS_init_rcc();
    SDS_init_gpio();
    SDS_init_usart();
    SDS_init_interrupt();
}

void sim808_init(void)

```

```

{
    sim808_init_rcc();
    sim808_init_gpio();
    sim808_init_usart();
    sim808_init_interrupt();
    delay_ms(100);
    sim808_usart_send_string("AT\r\n");
                                // check for module's status
    delay_ms(20);
    sim808_usart_send_string("ATE0\r\n");
                                // disable echo mode
    delay_ms(20);
    sim808_usart_send_string("AT+CMGF=1\r\n");
    delay_ms(20);
    sim808_usart_send_string("AT+IPR=115200\r\n");           //
sim808_BaudRate = 115200
    delay_ms(20);
    sim808_usart_send_string("AT+CNMI=2,2,0,0\r\n");
    delay_ms(20);
    sim808_usart_send_string("AT&W\r\n");                  // save data
    delay_ms(200);
    sim808_usart_send_string("AT+SAPBR=3,1,\"Contype\",\"GPRS\"");
    sim808_usart_send_string("\r\n");
    delay_ms(200);
    sim808_usart_send_string("AT+SAPBR=3,1,\"APN\",\"CMNET\"");
    sim808_usart_send_string("\r\n");
    delay_ms(1000);
    sim808_usart_send_string("AT+SAPBR=1,1");
    sim808_usart_send_string("\r\n");
    delay_ms(200);
    sim808_usart_send_string("AT+SAPBR=2,1");
    sim808_usart_send_string("\r\n");
    delay_ms(200);
    sim808_usart_send_string("AT+HTTPINIT");
    sim808_usart_send_string("\r\n");
    delay_ms(200);
    sim808_usart_send_string("AT+HTTPPARA=\"CID\",1");
    sim808_usart_send_string("\r\n");
    delay_ms(200);
    sim808_usart_send_string("AT+HTTPPARA=\"URL\",\"http://110.76.86.17
0:9000/data\"");
    sim808_usart_send_string("\r\n");
    delay_ms(200);
    sim808_usart_send_string("AT+HTTPPARA=\"CONTENT\",\"application/json\"");
    sim808_usart_send_string("\r\n");
    delay_ms(200);
}

```

```

void lcd_display(void){
    N5110_clear_screen(WHITE);
    N5110_print_string(12, 0,"DAI HOC BKHN", WHITE);
    N5110_print_string(18, 1,"THIET BI 1", WHITE);
    N5110_print_string(0, 2,"PM10 :", WHITE);
    N5110_print_string(0, 3,"PM2.5:", WHITE);
    N5110_print_string(0, 4,"Tem :", WHITE);
    N5110_print_string(0, 5,"Hum :", WHITE);
    N5110_print_string(59, 2,"ug/m3", WHITE);
    N5110_print_string(59, 3,"ug/m3", WHITE);
    N5110_print_string(63, 4,"*C", WHITE);
    N5110_print_string(63, 5,"%", WHITE);
    print_int(32, 2,p_dusty_sensor.p_sds_ss.pm10_value, WHITE);
    print_int(32, 3,p_dusty_sensor.p_sds_ss.pm25_value, WHITE);
    print_int(32, 4,(int32_t)p_dusty_sensor.p_dht_ss.temperature,
WHITE);
    print_int(32, 5,(int32_t)p_dusty_sensor.p_dht_ss.humi,
WHITE);
}

int main()
{
    SystemInit();
    delayInit();
    DHT22_init();
    SDS_init();
    sim808_init();
    N5110_init();
    N5110_clear_screen(WHITE);

while(1)
{
    while(DHT_GetTemHumi(&p_dusty_sensor.p_dht_ss) == DHT_ERROR
);

    if(received_flag)
    {
        p_dusty_sensor.p_sds_ss.pm25_value =
(uint16_t)((sds10_data[3] *256)+sds10_data[2])/10;
        p_dusty_sensor.p_sds_ss.pm10_value =
(uint16_t)((sds10_data[5] *256)+sds10_data[4])/10;
        received_flag = 0;
        update_data(&p_dusty_sensor);
        lcd_display();
    }
}

```

```

        }
    }

void USART2_IRQHandler(void) {
    sds10_data[counter_byte] = USART_ReceiveData(SDS_USART_DEVICE );
    if(counter_byte == 9){
        if((sds10_data[0] == 0xAA) && (sds10_data[1] == 0xC0) && (sds10_data[9] == 0xAB) && (sds10_data[8] ==(uint8_t)(sds10_data[2]+sds10_data[3]+sds10_data[4]+sds10_data[5]+sds10_data[6]+sds10_data[7]))){
            received_flag = 1;
            sds_timeout=0;
        }
        else
        {
            if (sds_timeout++ == 10)
        {
            //
        }
        counter_byte = 0;
    }
    else{
        counter_byte++;
    }
    USART_ClearFlag (SDS_USART_DEVICE, USART_IT_RXNE);
}

```

```

void USART1_IRQHandler(void) {
    data = USART_ReceiveData(SIM808_USART_DEVICE);
    USART_ClearFlag(SIM808_USART_DEVICE, USART_IT_RXNE);
}

```

## A2. Code Webserver phần BackEnd

```
const express = require('express');
```

```
const router = express.Router();
```

```
var currentPara = {
    id: 10,
    temperature: 30,
    humidity: 80,
    pm25: 50,
    pm10: 60
};
```

```
var act = {
    id: 0,
    act1State: 0,
```

```

        act2State: 0
    }

var newPara = {
    deviceId: 11,
    deviceName: 'DeviceX',
    sampleId: 1,
    timeStamp: 0,
    temperature: 30,
    humidity: 80,
    pm25: 50,
    pm10: 80,
    act1State: 0,
    act2State: 0
};

var timestamp = new Date().getTime()
var date = new Date(timestamp)
console.log(timestamp)
console.log(date.getHours().toString() + ":" + date.getMinutes().toString())

var chartData = {
    labels: [1, 2, 3, 4, 5, 6, 7],
    values: [15, 20, 26, 20, 25, 29, 28]
};
var allData = []
//allData.push(currentPara)
//allData.push(newPara)

// http://localhost:9000/data/currentdata?id=109
// return the current values of all parameter at an location specified by
// "id"
router.get('/currentdata', (req, res, next)=>{
    var id = req.query.id;
    var currentDate = {};
    allData.forEach(function(item, i){
        if(id == item.deviceId){
            currentDate= item
        }
    });
    //console.log(newestValue)
    res.status(200).json({
        currentDate
    });
});

});
```

```

//http://localhost:9000/data/chart?id=100&type=0&start=1575728492010&stop
=1575728506715
// Return the values of a parameter (specified by "type") in a specific d
uration
router.get('/chart', (req, res, next)=>
  var id = req.query.id;
  var typeOfData = req.query.type; // type = 0/1/2/3 <=> tem / hum/ pm2
5/ pm10
  var timeStart = req.query.start;
  var timeEnd   = req.query.stop;
  var extractedData = {
    labels: [],
    values: []
  };
  allData.forEach(function(item, i){
    if(id == item.deviceId && timeStart <= item.timeStamp && item.time
Stamp <= timeEnd){
      extractedData.labels.push(item.timeStamp);
      if (typeOfData == 0) { extractedData.values.push(item.tempera
ture) }
      if (typeOfData == 1) { extractedData.values.push(item.humidit
y) }
      if (typeOfData == 2) { extractedData.values.push(item.pm25) }

      if (typeOfData == 3) { extractedData.values.push(item.pm10) }
    }
  });
  //console.log(newestValue)
  res.status(200).json({
    extractedData
  });
});

// Return all values of a parameter and recorded time (specified by "type
") in a specific duration
router.get('/chart/all', (req, res, next)=>
  var id = req.query.id;
  var typeOfData = req.query.type; // type = 0/1/2/3 <=> tem / hum/ pm2
5/ pm10
  var extractedData = {
    labels: [],
    values: []
  };
  allData.forEach(function(item, i){

    var date = new Date(item.timeStamp)
    extractedData.labels.push(date.getHours().toString()+ ":" + date.
getMinutes().toString()+ ":" + date.getSeconds().toString());

```

```

        if (typeOfData == 0) { extractedData.values.push(item.temperature)
    } }
        if (typeOfData == 1) { extractedData.values.push(item.humidity) }
        if (typeOfData == 2) { extractedData.values.push(item.pm25) }
        if (typeOfData == 3) { extractedData.values.push(item.pm10) }

    });
    //console.log(newestValue)
    res.status(200).json({
        extractedData
    });
});

router.get('/chartData', (req, res, next)=>{
    chartData.values[0] = chartData.values[0] + 0.2;
    chartData.values[3] = chartData.values[3] + 0.2;
    res.status(200).json({
        chartData
    });
});

// return whole data
router.get('/all', (req, res, next)=>{
    res.status(200).json({
        allData
    });
});

router.post('/', (req, res, next)=>{
    const endDeviceData = {
        deviceId: req.body.deviceId,
        deviceName: 'Device' + req.body.deviceId.toString(),
        sampleId: allData.length,
        timeStamp: new Date().getTime(),
        temperature: req.body.temperature,
        humidity: req.body.humidity,
        pm25: req.body.pm25,
        pm10: req.body.pm10,
        act1State: req.body.act1State,
        act2State: req.body.act2State
    };
    allData.push(endDeviceData)
    //console.log('temperature: ', allData)
    console.log('JSON from End Device', req.body)
    //console.log('Parsed Data', allData)
    /*
    if (req.body.deviceId == act.id) {

```

```

        res.status(201).json({
            message: 'Successfully',
            act1State: act.act1State,
            act2State: act.act2State,
        });
    }
}

res.status(201).json({
    message: 'Successfully',
});

});

router.get('/control', (req, res, next)=>{
    // temporary use
    act.id = req.query.id;
    act.act1State = req.query.bt1State;
    act.act2State = req.query.bt2State;
    console.log(act)
    res.status(201).json({
        message: 'Control Request have been sent'
    });
});

module.exports = router;

```

### A3. Code Webserver phần FrontEnd

```

import React, { Component } from 'react';
import './App.css';
import Header from './components/Header';
import Product from './components/Product';
import DataBoard from './components/DataBoard';
import ControlBoard from './components/ControlBoard';
import Chart from './components/Chart';
import LineChart from './components/LineChart';
import ChartComponent from 'react-chartjs-2';
import Information from './components/Information';

class App extends Component {
    constructor(props) {
        super();
        this.state = {
            currentPara: {},
            chartData: {}
        }
    }
}

```

```

componentWillMount() {
    //const labels = ['January', 'February', 'March', 'April', 'May',
    'June', 'July'];
    //const data = [15, 20, 26, 20, 25, 29, 28];
    //this.getChartData(labels, data);
}
async componentDidMount() {

    //const urlCurrentData = "http://localhost:9000/data/currentdata"
    ;
    //const urlChartData = "http://localhost:9000/data/chart/all?id=
    =199&type=2";
    const urlCurrentData = "http://localhost:9000/data/currentdata?id=
    =199";
    const urlChartData1 = "http://localhost:9000/data/chart/all?id=
    =199&type=2";
    const urlChartData2 = "http://localhost:9000/data/chart/all?id=
    =199&type=3";
    setInterval(async () => {
        const response = await fetch(urlCurrentData);
        const data = await response.json();
        //console.log('data', data.currentPara)
        this.setState({ currentPara: data.currentData });
        //console.log('state', this.state.currentPara)
    }, 2000);

    setInterval(async () => {
        const response1 = await fetch(urlChartData1);
        const response2 = await fetch(urlChartData2);
        const xdata1 = await response1.json();
        const xdata2 = await response2.json();
        console.log(xdata1);
        console.log(xdata2);
        //console.log('data', data.chartData);
        //this.getChartData(xdata.chartData.labels, xdata.chartData.v
        alues);
        this.getChartData(xdata1.extractedData.labels, xdata1.extract
        edData.values,xdata2.extractedData.values);
    }, 2000);
}

getChartData(setlabel, setdata1, setdata2) {
    this.setState({
        chartData: {
            labels: setlabel,
            datasets: [
                {
                    label: 'PM2.5 ',
                    fill: true,

```

```

        lineTension: 0.3,
        backgroundColor: 'rgba(0,178,191,0.5)',
        borderColor: 'rgba(0,178,191,1)',
        borderCapStyle: 'butt',
        borderDash: [],
        borderDashOffset: 0.0,
        borderJoinStyle: 'miter',
        pointBorderColor: 'rgba(75,192,192,1)',
        pointBackgroundColor: '#fff',
        pointBorderWidth: 2,
        pointHoverRadius: 5,
        pointHoverBackgroundColor: 'rgba(220,220,0,0.2)',
        pointHoverBorderColor: 'rgba(220,220,220,1)',
        pointHoverBorderWidth: 2,
        pointRadius: 1,
        pointHitRadius: 10,
        data: setdata1
    },
    {label: 'PM10 ',
        fill: true,
        lineTension: 0.3,
        backgroundColor: 'rgba(223,53,57,0.5)',
        borderColor: 'rgba(223,53,57,1)',
        borderCapStyle: 'butt',
        borderDash: [],
        borderDashOffset: 0.0,
        borderJoinStyle: 'miter',
        pointBorderColor: 'rgba(75,192,192,1)',
        pointBackgroundColor: '#fff',
        pointBorderWidth: 2,
        pointHoverRadius: 5,
        pointHoverBackgroundColor: 'rgba(13,53,1,1)',
        pointHoverBorderColor: 'rgba(220,220,220,1)',
        pointHoverBorderWidth: 4,
        pointRadius: 1,
        pointHitRadius: 10,
        data: setdata2
    }
]

}
});

}

render() {
    return (

```

```

<div>
    <div className="row row-style">
        <div className="row">
            <h1 className="text-center white-text"> Air Quality Monitoring System </h1>
            <hr />
        </div>

        <div className="row">
            <div className="col-xs-2 col-sm-2 col-md-2 col-lg-2">
                <Information />
            </div>
            <div className="col-xs-10 col-sm-10 col-md-10 col-lg-10">
                <div className="row">
                    <div className="col-xs-3 col-sm-3 col-md-3 col-lg-3">
                        <DataBoard boardname="Temperature" iconsr
c="icon/temperature.png" type="#451;" value={this.state.currentPara.tempera
ture}/>
                    </div>
                    <div className="col-xs-3 col-sm-3 col-md-3 col-lg-3">
                        <DataBoard boardname="Humidity" iconsr
c="icon/humidity.png" type="#37;" value={this.state.currentPara.humidit
y}/>
                    </div>
                    <div className="col-xs-3 col-sm-3 col-md-3 col-lg-3">
                        <DataBoard boardname="PM2.5" iconsr
c="ico
n/dust.png" type="#181;g" value={this.state.currentPara.pm25}/>
                    </div>
                    <div className="col-xs-3 col-sm-3 col-md-3 col-lg-3">
                        <DataBoard boardname="PM10" iconsr
c="icon
/dustpm10.png" type="#181;g" value={this.state.currentPara.pm10} />
                    </div>
                </div>
                <div className="row">
                    <div className="col-xs-9 col-sm-9 col-md-9 col-lg-9">
                        <LineChart chartData={this.state.chartDat
a} />
                    </div>
                </div>
            </div>
        </div>
    </div>

```

```
        <div className="row">
          <div className="col-xs-9 col-sm-9 col-md-
9 col-lg-9">
            <ControlBoard c={this.state.chartData} />
          </div>
        </div>
      </div>
    </div>
  );
}

}

export default App;
```