TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

——————— * ———————

ĐỒ ÁN

TỐT NGHIỆP ĐẠI HỌC

NGÀNH KĨ THUẬT MÁY TÍNH

PHÁT TRIỂN HỆ QUAN TRẮC DỮ LIỆU CHẤT LƯỢNG KHÔNG KHÍ

Sinh viên thực hiện: **Nguyễn Văn Diện**

Lớp: CNTT 1-1 – K60

Giáo viên hướng dẫn: **TS**. **Ngô Lam Trung**

HÀ NỘI 6-2020

PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

1. Thông tin về sinh viên

Họ và tên sinh viên: Nguyễn Văn Diện
Điện thoại liên lạc: 0355813217 Email: dienvx1997bn@gmail.com
Lớp: CNTT 1-1 K60 Hệ đào tạo: Chính quy

2. Mục đích nội dung của ĐATN

Xây dựng firmware, thiết kế thiết bị giám sát và thu thập dữ liệu chỉ số chất lượng không khí (độ bụi, nhiệt độ, độ ẩm), lưu trữ dữ liệu trên server và hiển thị lên website.

3. Các nhiệm vụ cụ thể của ĐATN

- Tìm hiểu và nghiên cứu các công nghệ đo chỉ số chất lượng không khí
- Thiết kế phần cứng
- Phát triển firmware cho thiết bị
- Xây dựng website hiển thị dữ liệu
- Chế tạo prototype thử nghiệm

4. Lời cam đoan của sinh viên:

Nguyễn Văn Diện - cam kết ĐATN là công trình nghiên cứu của bản thân dưới sự hướng dẫn của *Tiến sĩ Ngô Lam Trung*. Các kết quả nêu trong ĐATN là trung thực, không phải là sao chép toàn văn của bất kỳ công trình nào khác.

Hà Nội, ngày tháng năm

Tác giả ĐATN

Nguyễn Văn Diện

5. Xác nhận của giáo viên hướng dẫn về mức độ hoàn thành của đồ án tốt nghiệp và cho phép bảo vệ:

Hà Nội, ngày tháng năm

Giáo viên hướng dẫn

TS. Ngô Lam Trung

LỜI CẢM ƠN

Xin chân thành cảm ơn!

Người thực hiện đề tài

Nguyễn Văn Diện

TÓM TẮT NỘI DUNG ĐỒ ÁN

Thời gian qua, với thực trạng chất lượng không khí ở nước ta đặc biệt là ở Hà Nội luôn trong tình trạng cảnh báo thậm chí là nguy hiểm tới sức khỏe con người. Với mong muốn xây dựng một hệ thống quan trắc chất lượng không khí nhằm mục đích giám sát và cảnh báo về tình trạng chất lượng không khí, em quyết định chọn đề tài "Phát triển hệ quan trắc dữ liệu chất lượng không khí". Trong toàn bộ đồ án này, tập trung vào xây dựng thiết bị thu thập dữ liệu và đưa ra chỉ số đánh giá chất lượng không khí AQI và cung cấp giao diện theo dõi theo từng thời điểm đo trong ngày.

Dưới đây là đề mục các chương sẽ trình bày trong đồ án và sẽ bám sát vào nó để triển khai. Đồ án chia làm 6 chương chính:

Chương 1. Giới thiệu đề tài
Chương 2. Thiết kế phần cứng
Chương 3. Phát triển firmware cho thiết bị
Chương 4. Xây dựng website
Chương 5: Kết quả đạt được
Chương 6: Kết luận và đánh giá

Trong mỗi chương sẽ trình bày thành các đề mục nhỏ nhằm phân tích cụ thể và nêu ra các bước cần triển khai cho từng nhiệm vụ.

MỤC LỤC

DANH MỤC BẢNG

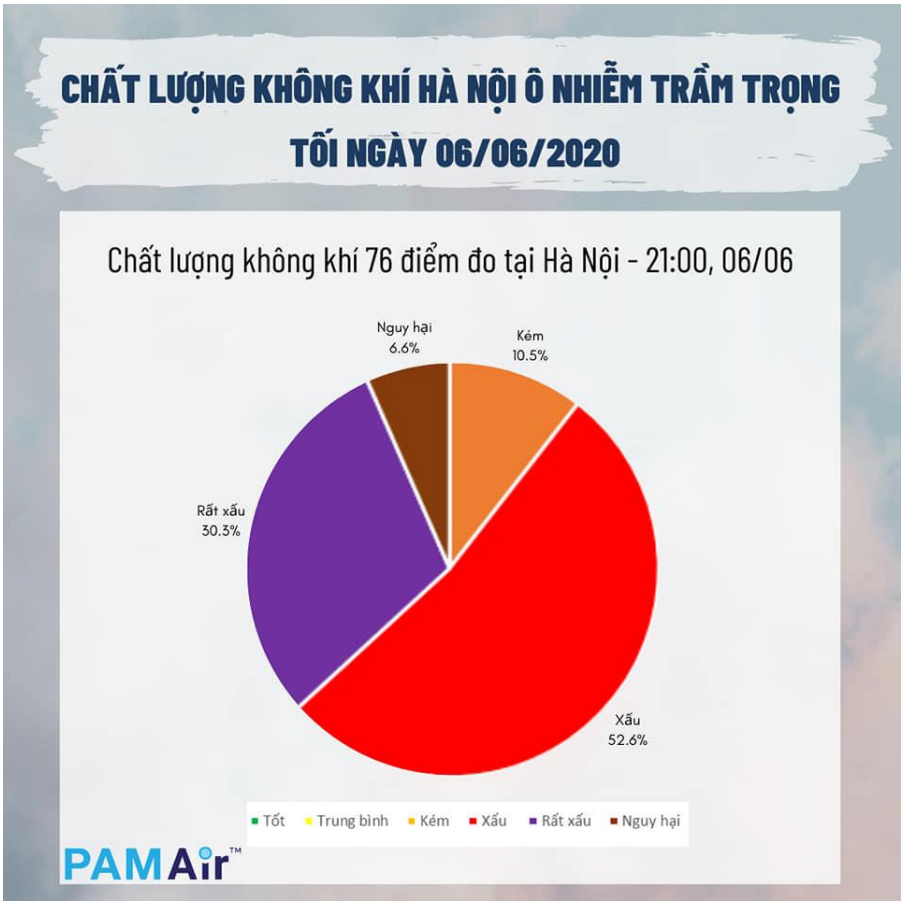
DANH MỤC HÌNH VẼ

DANH MỤC CÁC TỪ VIẾT TẮT

HTTP	Hyper Text Transfer Protocol
GPS	Global Positioning System
MQTT	Message Queuing Telemetry Transport
AQI	Air Quality Index
MCU	Micro Controller Unit
ĐATN	Đồ án tốt nghiệp
MSB	most significant bit
SPIFFS	SPI Flash File System

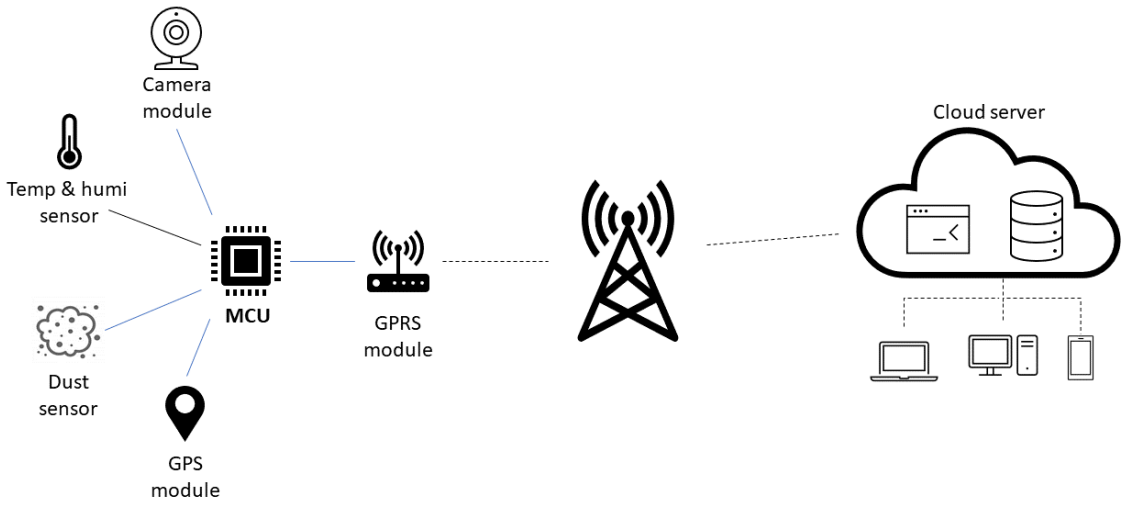
CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

1.1. Đặt vấn đề



1.2. Mô tả tổng quan hệ thống

Hệ thống quan trắc chất lượng không khí gồm ba thành phần chính: hệ thống đo và thu thập dữ liệu không khí tại hiện trường, hệ thống truyền dẫn dữ liệu và hệ thống theo dõi giám sát. Mô hình tổng quan của hệ thống được mô tả như hình dưới đây:



Hình 1 Sơ đồ tổng quát của hệ thống

Phần thiết bị có tính năng thu thập dữ liệu môi trường như nhiệt độ, độ ẩm, độ bụi. Camera chụp ảnh vừa có tính năng theo dõi vừa để có cái nhìn đánh giá về không gian xung quanh thiết bị. Ví dụ như không gian rộng hay hẹp, có gần các công trình đang xây dựng hay không, hay theo dõi sự thay đổi của môi trường không khí tại một địa điểm cố định nào đó nhằm góp phần xác minh tính xác thực của dữ liệu thu được. Với việc sử dụng kết hợp định vị vệ tinh GPS và sử dụng giao tiếp truyền dẫn dữ liệu thông qua mạng GPRS sẽ cho phép thiết bị chỉ cần cấp nguồn là có thể hoạt động và gửi thông tin về CLKK tại điểm đo đó lên server,

Việc trao đổi dữ liệu giữa các thiết bị với server có được thực hiện thông qua hai giao thức truyền tải dữ liệu thông dụng hiện nay là MQTT và HTTP. Sử dụng giao thức MQTT rất phù hợp để trao đổi dữ liệu và lệnh điều khiển giữa thiết bị và server, kết hợp sử dụng giao thức HTTP để truyền file ảnh để lưu trữ trên cloud.

Phía server có nhiệm vụ lưu trữ dữ liệu của các cảm biến thu được và hình ảnh từ thiết bị truyền lên. Hệ thống bao gồm website hiển thị dữ liệu dạng đồ thị giúp so sánh đánh giá về chất lượng không khí giữa các thời điểm đo.

1.3. Mục tiêu và nhiệm vụ của đề tài

Đồ án tập trung vào phát triển thiết bị thu thập dữ liệu về chất lượng không khí, bao gồm:

- ✓ Phát triển phần cứng thiết bị:
  - Thiết kế khối nguồn
  - Thiết kế các khối đọc mức pin
  - Thiết kế các khối giao tiếp cảm biến
- ✓ Phát triển phần mềm cho thiết bị:
  - Phát triển chức năng thu thập dữ liệu từ các sensor

- Phát triển trao đổi dữ liệu thiết bị và server qua giao thức MQTT và HTTP
- Phát triển tính năng update firmware từ xa

✓ Phát triển server:

- Phát triển website hiển thị dữ liệu
- Lưu trữ dữ liệu vào CSDL SQL

1.4. Giải pháp

1.4.1. Giải pháp phần cứng

Thiết bị được xây dựng trên module ESP32 camera với một camera OV2604 độ phân giải 2 megapixels (1632×1232 pixels) kích thước nhỏ gọn đầy đủ tính năng và có một bộ xử lý ảnh bên trong. Trên module còn có khe cắm thẻ nhớ microSD để mở rộng không gian lưu trữ dữ liệu.

Hình 2 Module ESP32-Camera

Các cảm biến được sử dụng trong đồ án bao gồm:

- Cảm biến nhiệt độ độ ẩm: SHT10 để đo nhiệt độ, độ ẩm trong không khí
- Cảm biến đo bụi laser: SDS011 để đo nồng độ hạt bụi mịn trong không khí
- Module định vị: GPS Ne0-6M xác định vị trí sử dụng vệ tinh GPS của Mỹ
- Module GSM/GPRS: SIM800 để giao tiếp với server mạng qua internet

1.4.2. Giải pháp phần mềm

Do hệ thống bao gồm nhiều công việc như đọc giá trị từ các cảm biến, định vị GPS, giao tiếp mạng. Quyết định lựa chọn hệ điều hành FreeRTOS để quản lý công việc hợp lý. FreeRTOS là một hệ điều hành thời gian thực miễn phí giúp lập trình hệ thống phức tạp dễ dàng hơn thông qua việc chia chương trình thành các tác vụ (task) và nó giải quyết việc điều phối các task này, lập lịch và phân mức ưu tiên cho task và nhận các thôn điệp gửi đi từ task. Các chức năng của chương trình sẽ được thực hiện trên các tác vụ khác nhau và có thể hoạt động "đồng thời" với nhau.

Giao thức MQTT là giao thức phổ biến trong thời đại Iot ngày nay. Đây là một giao thức gọn nhẹ, được thiết kế để kết nối các thiết bị mà có mạng băng thông thấp rất phù hợp với hệ thống giám sát này. Sử dụng HTTP để có thể gửi được file ảnh lên server một cách đơn giản và dễ dàng.

1.5. Công cụ và môi trường phát triển

1.5.1. Công cụ dùng để lập trình ESP32

Trong đồ án sử dụng Arduino làm nền tảng lập trình bởi vì các lý do sau:

- Miễn phí và cộng đồng phát triển mạnh
- Hỗ trợ nhiều thư viện đọc và giao tiếp sensor
- Hỗ trợ FreeRtos

Môi trường lập trình trên hệ điều hành Windown 10

1.5.2. Các công cụ khác

Sử dụng tiện ích mở rộng Vmicro cho phép biên dịch và nạp code Arduino trên Visual Studio để code thuận tiện hơn.

Sử dụng phần mềm thiết kế mạch EasyEDA. Đây là phần mềm miễn phí, đơn giản, có thể sử dụng trực tiếp trên website đáp ứng đầy đủ yêu cầu cơ bản cho việc thiết kế mạch điện tử.

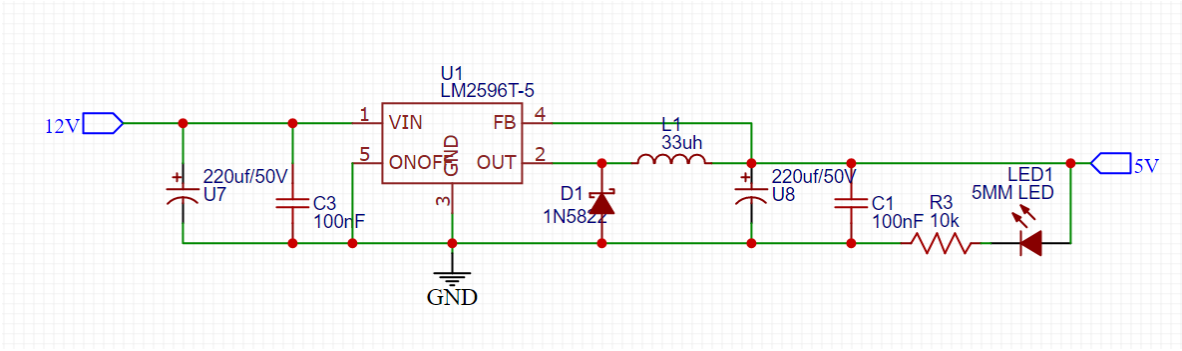
CHƯƠNG 2. THIẾT KẾ PHẦN CỨNG

Chương này sẽ mô tả chi tiết việc thiết kế phần cứng cho thiết bị. Phần cứng của thiết bị sẽ được xây dựng dựa trên module ESP32-camera. Và bao gồm các chức năng: giao tiếp với các sensor nhiệt độ độ ẩm, sensor độ bụi, module GPS, module SIM, bao gồm cả khối nguồn và khối đọc mức pin.
Sau đây là phần trình bày chi tiết các khối của thiết bị.

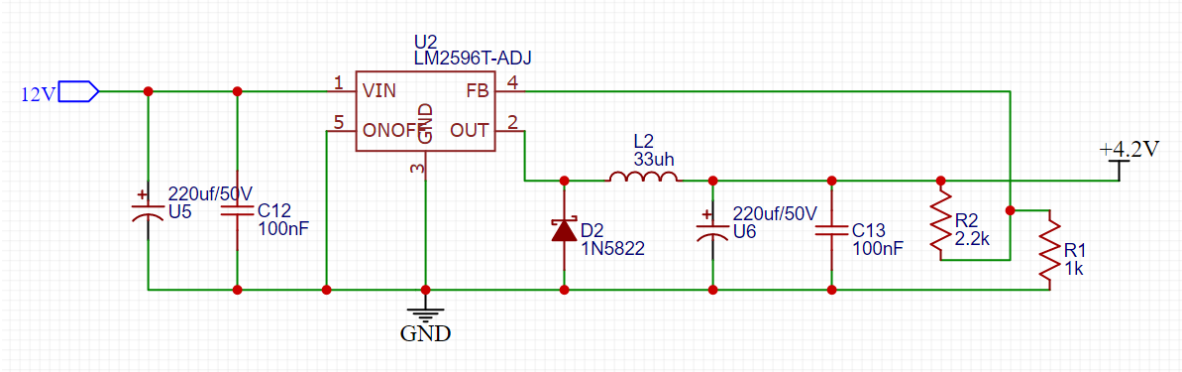
2.1. Thiết kế khối nguồn

Nguồn vào cho thiết bị là nguồn 12VDC có thể được cấp bằng ắc-quy hoặc trực tiếp từ adapter. Module SIM800L hoạt động với nguồn cấp 4,2V-1A. Module ESP32 camera hoạt động với nguồn vào mức 5V nhưng giao tiếp mức điện áp 3,3V, trên module ESP32 camera đã có sẵn module hạ áp 5V xuống 3,3V có thể tận dụng để cấp nguồn cấp trực tiếp cho sensor.

Vậy tổng kết lại cần thiết kế 2 khối nguồn: Khối hạp áp 12V → 4,2V và khối hạ áp 12V→ 5V. Ở đây lựa chọn IC hạ áp LM2596 với khả năng hạ áp từ 35VDC và cho tải tối đa 3A. Có 2 loại IC là LM2596 5.0 cho đầu ra ổn định 5VDC và IC LM2596ADJ có thể tùy chỉnh điện áp đầu ra thông qua biến trở.



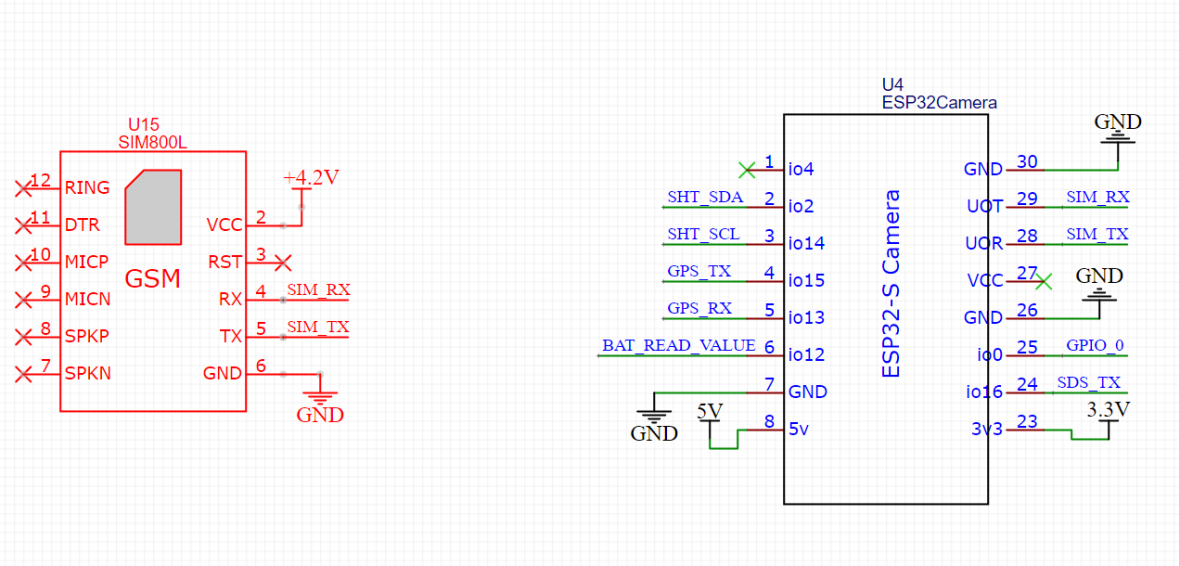
Hình 3 Sơ đồ nguyên lý khối nguồn 5V



Hình 4 Sơ đồ nguyên lý khối nguồn 4.2V

2.2. Thiết kế khối MCU

Module ESP32 camera cho phép cấu hình các chân I/O ở các chế độ hoạt động khác nhau (UART, I2C, PWM, ADC). Ở đây sử dụng 3 cổng UART để giao tiếp với module SIM, cảm biến bụi và module GPS. Sử dụng 2 chân I/O để giao tiếp I2C với sensor nhiệt độ độ ẩm. Sử dụng thêm 1 chân ADC để đọc giá trị mức pin.



Hình 5 Sơ đồ nguyên lý khối MCU và SIM

Các chân GPIO sử dụng

Tên ngoại vi	Chuẩn giao tiếp	Chân GPIO sử dụng
Module SIM800L	UART	TX – GPIO 1, RX – GPIO 3
Module GPS	UART	TX – GPIO 15, RX – GPIO 13
Sensor SHT10	I2C	SDA – GPIO 2, SCL – GPIO 14
SDS011	UART	TX – GPIO 16
Khối đọc pin	ADC	GPIO 12

Bảng 1 Sơ đồ cấu hình chân GPIO của module ESP32 trên thiết bị

Module ESP32 camera

Một số thông tin của module ESP32-camera:

- Hỗ trợ wifi 802.11b/g/n, Bluetooth Low Energy
- Vi sử lý 32-bit Dual-core Xtensa LX6, tốc độ 160-240MHz
- Bộ nhớ trong 520 KB SRAM, bộ nhớ ram mở rộng 4M PSRAM

- Hỗ trợ UART/SPI/I2C/PWM/ADC/DAC
- Hộ trợ camera OV2640 và OV7670, có đèn flash
- Hỗ trợ upload ảnh qua WiFI
- Hỗ trợ TF card
- Hỗ trợ nhiều chế độ sleep modes
- Hỗ trợ FreeRTOS
- Hoạt động các chế độ Station mode: STA/AP/STA+AP
- 10 chân GPIOs khả dụng (các chân còn lại đã được sử dụng để giao tiếp với camera)
- Có đèn flash hỗ trợ (kết nối với chân GPIO 4)

Trên module ESP32 camera sử dụng camera OV2640 với một số thông số như sau:

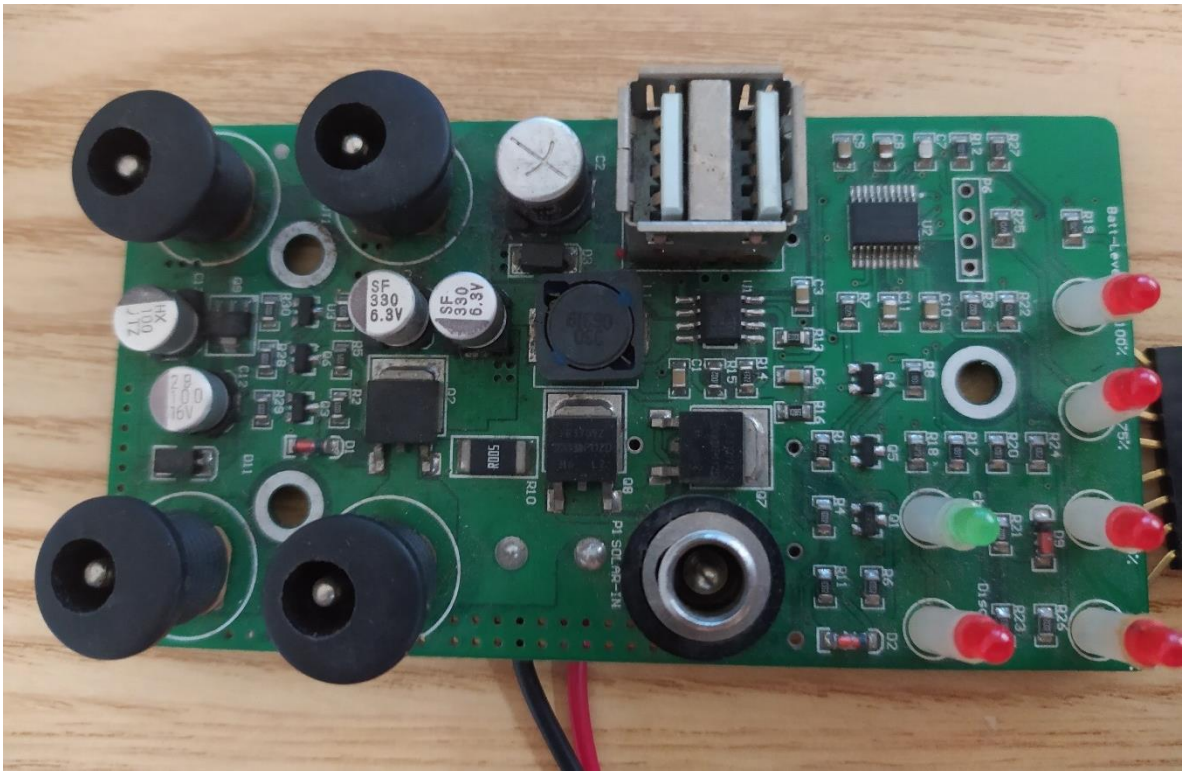
- Độ phân giải 2 megapixels (1632×1232 pixels)
- Điện áp hoạt động 3.3V
- Hỗ trợ kích thước ảnh UXGA, SXGA, VGA, QVGA, QQVGA, CIF, QCIF
- Hỗ trợ tùy chỉnh độ tương phản, độ sáng, xoay ảnh, chất lượng hình ảnh, độ bão hòa, màu sắc,…
- Hỗ trợ đầu ra JPEG, GRAYSCALE, RGB565, YUV422

2.3. Thiết kế khối đọc mức pin

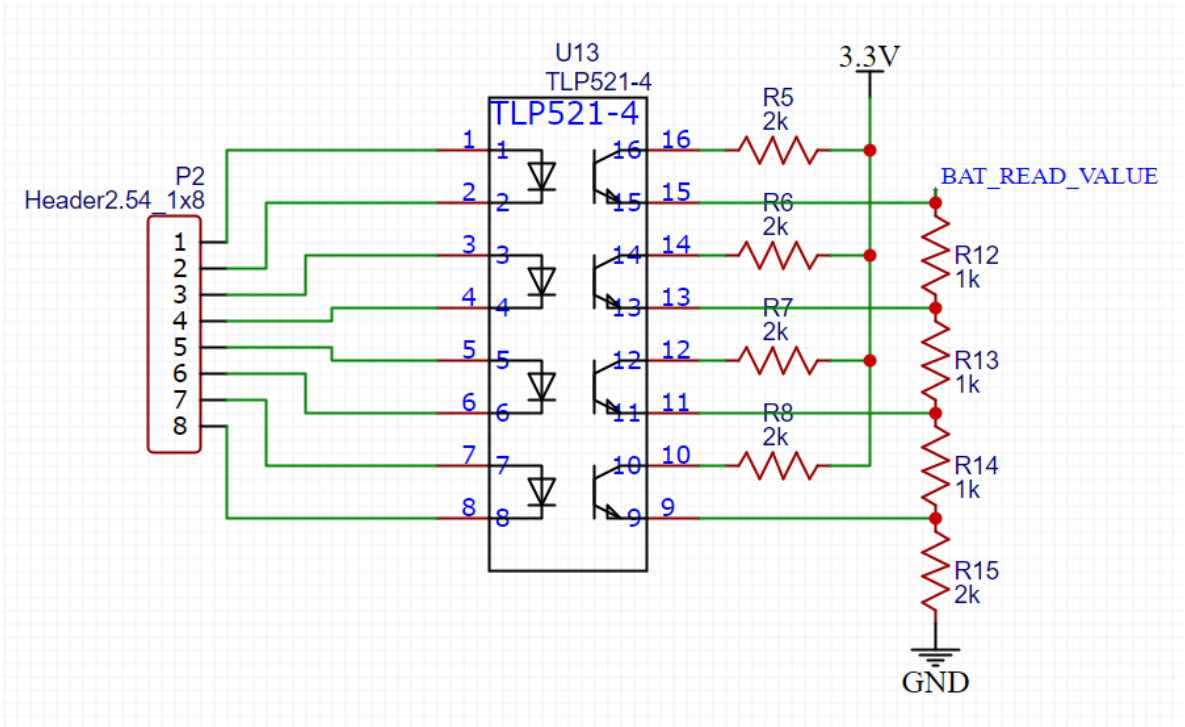
Mạch sạc pin ắc quy không có hỗ trợ đầu ra tín hiệu, chỉ có 6 led chỉ thị bao gồm led nguồn, led báo sạc pin và 4 led hiển thị mức phần trăm pin 100 – 75 – 50 – 25 %.

Với mục đích theo dõi mức pin để đưa ra cảnh báo hoạt động cho thiết bị, thiết bị cần thêm một khối đọc pin dựa vào 4 led báo mức pin này. Việc thiết bị có đang sạc hay không sạc có thể suy đoán từ mức pin thu được không cần sử dụng thêm chân giao tiếp

Thông thường, với 4 led hiển thị có thể nối trực tiếp vào 4 chân digital input. Nhưng do phần cứng bị giới hạn số chân GPIO khả dụng nên cần xây dựng 1 khối đọc pin đầu vào là 4 tín hiệu số digital và đầu ra có thể phân biệt được các mức trạng thái. Qua nghiên cứu và tìm hiểu thì việc lựa chọn thiết kế bộ chuyển đổi DAC là phù hợp nhất.



Hình 6 Mạch sạc pin ắc quy dùng năng lượng mặt trời



Hình 7 Nguyên lý khối đọc pin

Nguyên lý hoạt động:

Đây là bộ Digital-Analog-Converter (DAC) bằng cách sử dụng điện trở R2R sẽ tạo thành các cầu phân áp cho đầu ra khả dụng đọc bằng bộ ADC trong khối MCU. Đầu vào tín hiệu từ pin sẽ được đưa qua khối cách ly quang sử dụng IC TLP521-4 nhằm tránh gây ảnh hưởng từ nguồn pin trực tiếp tới chân giao tiếp của khối điều khiển. Với 4 tín hiệu từ 4 led hiển thị sẽ có các mức tín hiệu đầu vào 1 1 1 1; 1 1 1 0; 1 1 0 0; 1 0 0 0; 0 0 0 0 (với 1 là trạng thái led sáng và có tín hiệu, 0 là trạng thái led tắt không có tín hiệu). Qua tính toán, với các khả năng của tín hiệu đầu vào này thì đầu ra sẽ tương ứng kết quả như hình dưới

Tín hiệu	V_out
0 0 0 0	
1 0 0 0	VCC/16
1 1 0 0	
1 1 1 0	
1 1 1 1	

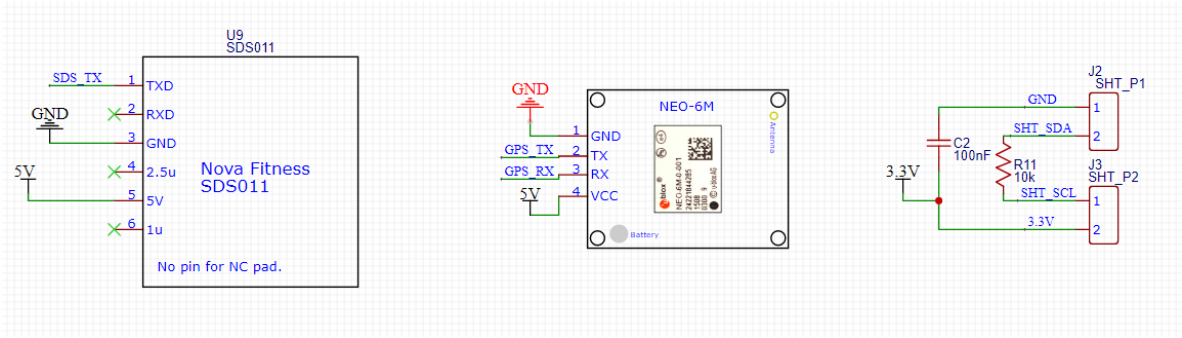
Hình 8 Kết quả đầu ra với các tín hiệu đầu vào khối đọc pin

2.4. Thiết kế khối cảm biến

Cảm biến nhiệt độ độ ẩm SHT10 hoạt động ở mức điện áp 3.3V, giao tiếp I2C

Cảm biến đo độ bụi SDS011 dạng module, sử dụng nguồn 5V và giao tiếp UART

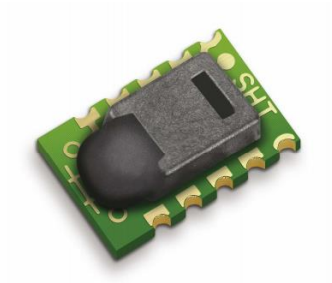
Module GPS hoạt động ở mức điện áp 5V, giao tiếp UART



Hình 9 Sơ đồ nguyên lý khối cảm biến

2.4.1.Cảm biến nhiệt độ độ ẩm

Có nhiều loại cảm biến nhiệt độ và độ ẩm trên thị trường. Nhưng cần cân đối giữa giá thành với chất lượng sản phẩm. Trong đồ án này sử dụng cảm biến SHT10 của hãng SENSIRION

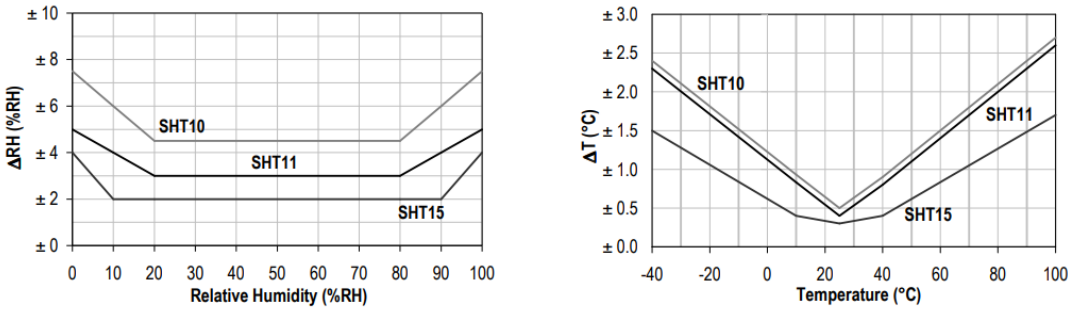


Hình 10 Cảm biến nhiệt độ, độ ẩm SHT10

Thông số kỹ thuật của SHT10:

Điện áp hoạt động	3.3V – 5V DC
Giao tiếp	chung giao tiếp I2C
Khoảng đo	Độ ẩm từ 0-100%RH, nhiệt độ từ -40~123 °C
°Chính xác	±4%RH, ±0.5 °C chi tiết Hình 11
Độ nhạy	0.05% RH, 0.01 °C
Độ phân giải	8-12-14 bit (12 bit mặc định với độ ẩm và 14 bit với nhiệt độ)

Bảng 2 Thông số kỹ thuật của SHT10



Hình 11 Sai số của sensor SHT10

2.4.2. Cảm biến độ bụi

Cảm biến độ bụi SDS011 là cảm biến đo nồng độ các hạt bụi có trong không khí bằng công nghệ laser. Cảm biến có thể phát hiện nồng độ các hạt có kích thước từ 0,3 đến 10 µm. Trên cảm biến có quạt hút giúp lưu không không khí và một vi xử lý 8bit cho đầu ra PM2.5 và PM10 trực tiếp dạng xung PWM hoặc gửi liên tục bản tin qua cổng serial ở tần số 1Hz.

Nguyên tắc hoạt động của cảm biến:

Cảm biến độ bụi SDS011 sử dụng công nghệ laser tán xạ ánh sáng. Sự tán xạ ánh sáng xảy ra khi các hạt đi qua khu vực phát hiện, tại khu vực này ánh sáng được chuyển thành tín hiệu điện và qua bộ khuếch đại để xử lý. Số lượng và đường kính các hạt được tính toán bằng cách phân tích dạng sóng của tín hiệu thu được vì dạng sóng tín hiệu và đường kính hạt có mối quan hệ nhất định.

Tuổi thọ của thiết bị:

Cảm biến laser bên trong thiết bị có độ bền cao, thời gian hoạt động liên tục tới 8000 giờ.

Hình 12 Cảm biến bụi SDS011

Thông số kỹ thuật của SDS011

Điện áp hoạt động	5V DC
Giá trị đo	PM2.5, PM10
Khoảng đo	0.0 – 999.9 µg /m3
Dòng tiêu thụ	70mA±10mA
Chu kỳ gửi gói tin qua serial	1Hz
Baud Rate	9600
Độ phân giải tối thiểu của hạt	0.3 µm
Sai số tương đối	Tối đa ±15% và ±10µg/m3 (ở 25℃, 50%RH)

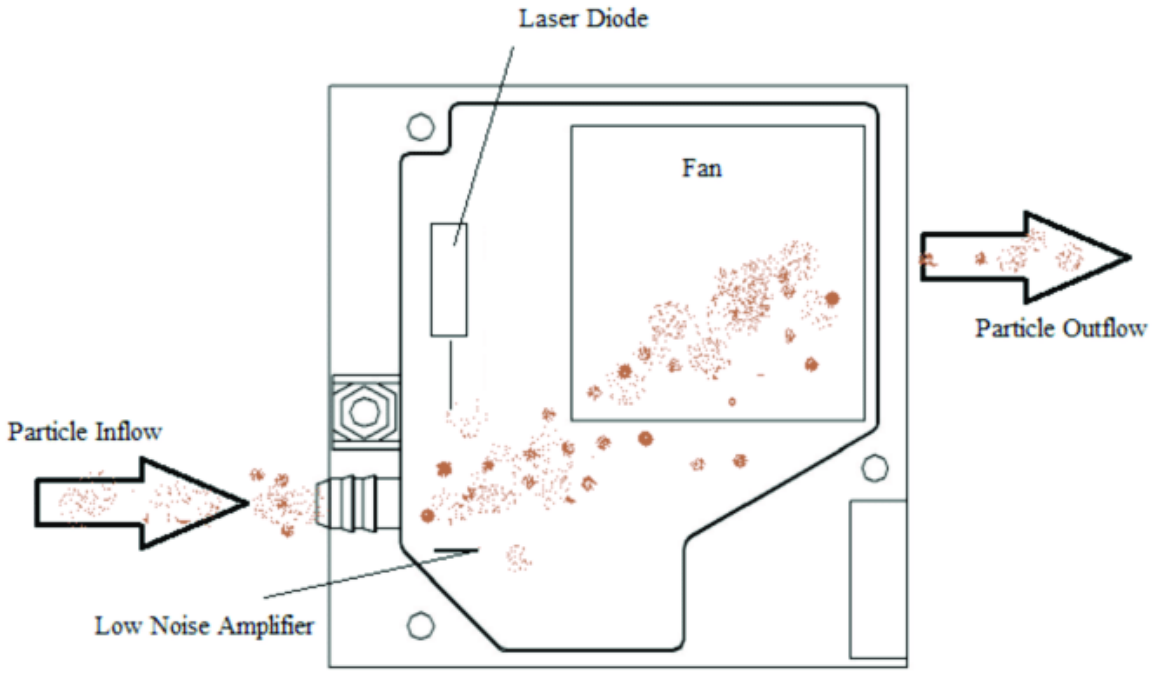
Bảng 3 Thông số kỹ thuật của cảm biến SDS011

Sơ đồ cổng giao tiếp sensor

Thứ tự	Tên	Nội dung
1	NC	Không kết nối

2	1μm	PM2.5:0-999μg/m³; PWM Output
3	5V	Nguồn vào 5V
4	2.5μm	PM10: 0-999 μg/m³;PWM Output
5	GND	Chân nối đất
6	R	Chân RX của UART（TTL）3.3V
6	T	Chân TX của UART（TTL）3.3V

Bảng 4 Sơ đồ cổng giao tiếp cảm biến SDS011



Hình 13 Hướng đặt sensor SDS011

2.5. Khối giao tiếp module GPS và GSM/GPRS

2.5.1. Module định vị GPS

Sử dụng module GPS NEO-6M là module định vị toàn cầu sử dụng hệ thống định vị vệ tinh GPS của Mỹ. Module GPS NEO-6M có khả năng theo dõi tới 22 vệ tinh trên 50 kênh và xác định vị trí được tại mọi nơi trên thế giới. Ưu điểm của module này là chi phí thấp, dễ dàng giao tiếp với các loại vi điều khiển (qua giao tiếp UART TTL). Với công suất hoạt động thấp (dòng hoạt động tối đa 50mA) sẽ rất phù hợp cho các ứng dụng dùng pin.

GPS sử dụng board điều khiển kết nối của hãng U-BLOX đến từ Thụy Sĩ có rất nhiều năm kinh nghiệm trong lĩnh vực sản xuất module định vị toàn cầu. Việc cấu hình thông số kết

nối GPS, thời gian nhấp nháy LED, mức năng lượng hoạt động... được thiết lập thông qua phần mềm u-Center

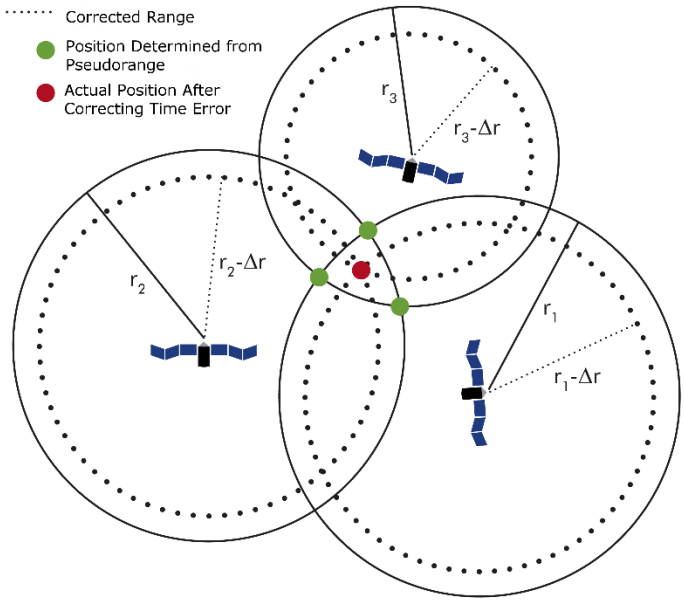
Một vài thông số kỹ thuật của GPS NEO-6M

Băng tần hoạt động	GPS L1(1575.42Mhz)
Độ chính xác theo chiều ngang	2.5m
Thời gian cập nhật	1HZ (tối đa 5Hz)
Dòng tiêu thụ bình thường	50mA
Dòng tiêu thụ chế độ tiết kiệm năng lượng	30mA
Baud Rate	4800-230400 (default 9600)

Bảng 5 Một vài thông số kỹ thuật của module GPS NEO-6M

Cách hoạt động của module GPS NEO-6M:

Các vệ tịnh GPS bay vòng quanh Trái đất hai lần một ngày trên một quỹ đạo cố định. Mỗi vệ tinh truyền một tín hiệu và thông số quỹ đạo riêng của nó cho phép các thiết bị GPS giải mã và tính toán vị trí chính xác của vệ tinh. Thiết bị thu GPS sẽ tính toán ra vị trí chính xác của nó trên Trái đất thông qua cách đo khoảng cách đến từng vệ tinh theo thời gian nhận được mỗi tín hiệu.



Hình 14 Cách hoạt động GPS

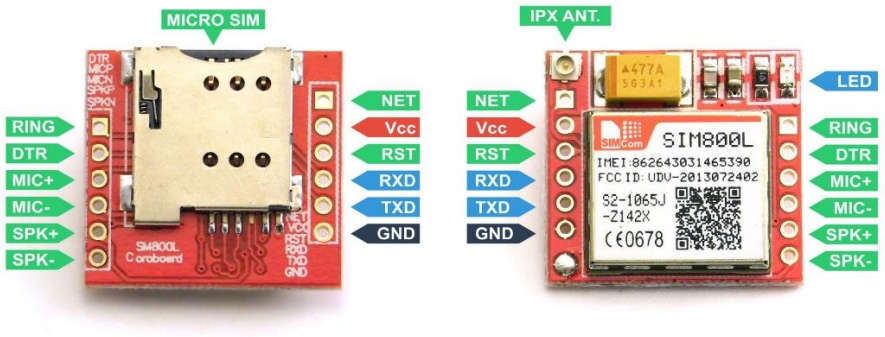
Thiết bị thu GPS tính toán khoảng cách đến mỗi vệ tinh là bao xa thông qua việc tính xem mất bao lâu để các tín hiệu từ vệ tinh này truyền đến. Một khi có ít nhất thông tin khoảng cách với 3 vệ tinh và vị trí của 3 vệ tinh này thì có thể suy ra được vị trí của thiết bị thu GPS trên trái đất. Và với từ 4

vệ tinh trở lên sẽ xác định được vị trí 3 chiều (vĩ độ, kinh độ và độ cao). Thông thường module GPS NEO-6M sẽ theo dõi 8 vệ tinh trở lên để cho kết quả chính xác nhất.

Để module có thể định vị được GPS tốt nhất cần đem ra ngoài trời trong khoảng 3 phút. Khi nào đèn LED trên mạch nháy sáng thì tức là lúc đó đã định vị thành công.

2.5.2. Module GSM/GPRS

Moudle SIM800L là một module di động thu nhỏ cho phép truyền nhận GPRS, gửi và nhận SMS cũng như thực hiện và nhận cuộc gọi thoại. Module này sử dụng bộ tập lệnh AT nên dễ dàng giao tiếp với vi điều khiển (chuẩn UART). Trên module có sẵn khe sim hỗ trợ MICROSIM thông dụng.



Hình 15 Module SIM800L

Thông số kĩ thuật:

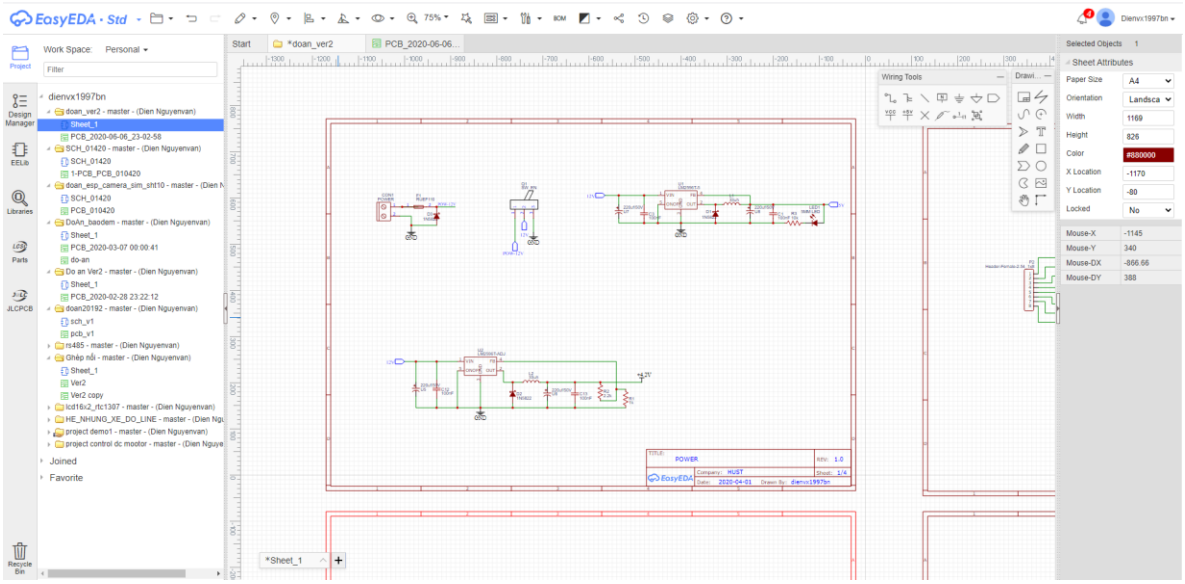
Điện áp hoạt động	3.7 - 4.2V
Dòng tiêu thụ khi ở chế độ chờ	10 mA
Dòng khi hoạt động	100 mA đến 1A
Dòng tiêu thụ bình thường	50mA
Khe cắm SIM	MICROSIM
Băng tần GSM	GSM850MHz, EGSM900MHz, DSC1800Mhz, PCS1900MHz
Chuẩn giao tiếp	UART

Một vài lưu ý khi sử dụng:

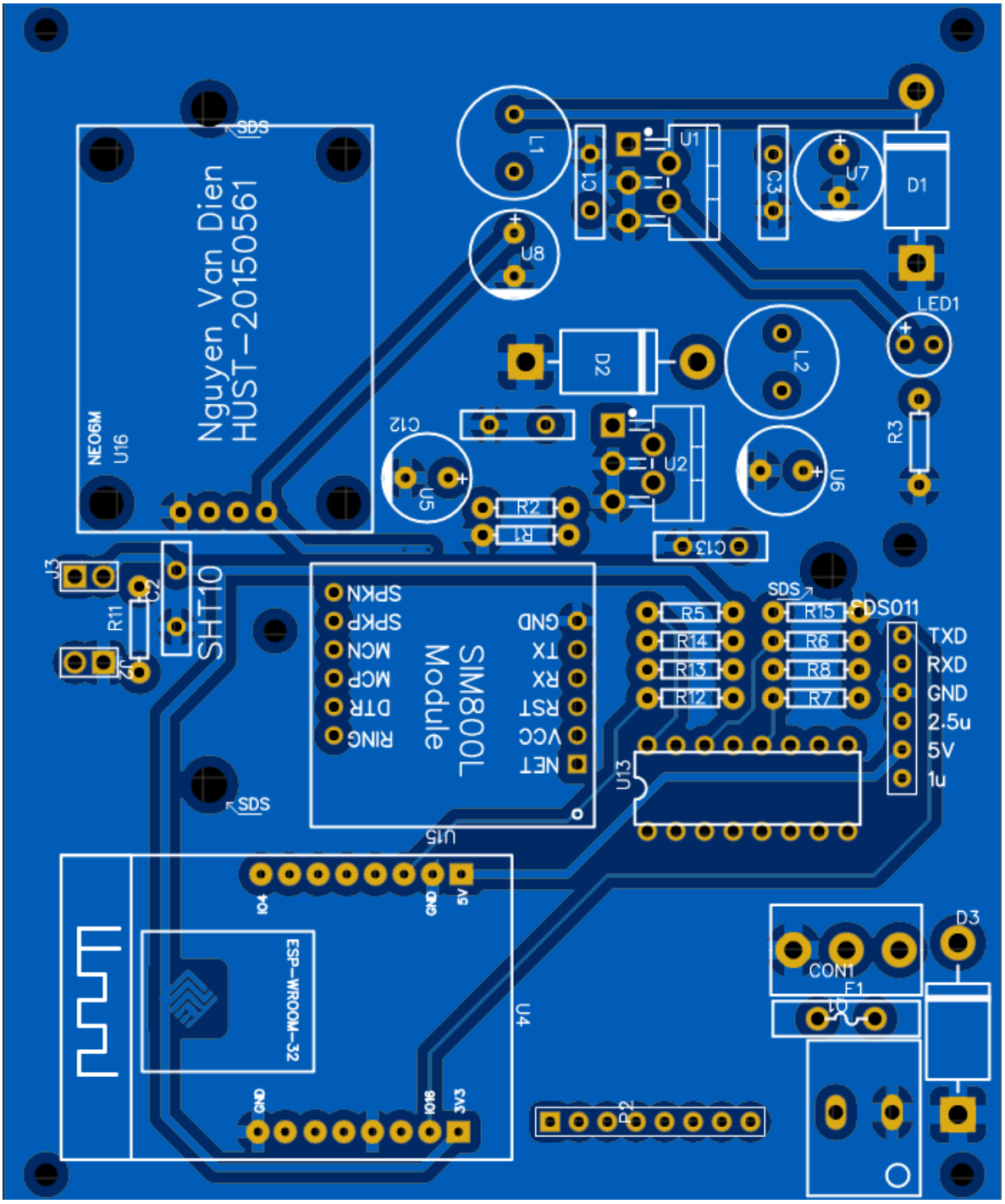
Cần cấp nguồn 4.2VDC cho mạch và dòng tải tối thiểu 1A và tối đa 2A để đảm bảo cho module hoạt động ổn định

2.6. Thực hiện vẽ mạch in

Thực hiện vẽ mạch trực tiếp trên website easyeda.com



Hình 16 Giao diện phần mềm EasyEDA



Hình 17 Kết quả thu được khi sử dụng phần mềm EasyEDA

CHƯƠNG 3. PHÁT TRIỂN FIRMWARE CHO THIẾT BỊ

3.1. Tìm hiểu về hệ điều hành FreeRTOS

3.1.1. Giới thiệu

Một hệ thống thời gian thực (real-time system) là hệ thống điều khiển một vật thể vật lý với một tốc độ phù hợp với sự tiến triển của tiến trình. Ví dụ như hệ thống hiển thị thời gian chính xác, hệ thống làm việc theo chu kỳ. Hệ thống thời gian thực có sự gò bó về thời gian hoạt động không chỉ đưa ra kết quả chính xác mà nó phải thực hiện để đưa ra kết quả đó trong một khoảng thời gian rất ngắn. "Thời gian thực" ở đây phải đảm bảo các yếu tố: đáp ứng nhanh, các tác vụ được xác định bởi deadline (là thời gian tối đa để tác vụ hoàn thành việc tính toán).

FreeRTOS là một hệ điều hành nhúng thời gian thực (Real Time Operating System) mã nguồn mở được phát triển bởi Real Time Engineers Ltd, sáng lập và sở hữu bởi Richard Barry. FreeRTOS được thiết kế phù hợp cho nhiều hệ nhúng nhỏ gọn vì nó chỉ triển khai rất ít các chức năng như: cơ chế quản lý bộ nhớ và tác vụ cơ bản, các hàm API quan trọng cho cơ chế đồng bộ. Nó không cung cấp sẵn các giao tiếp mạng, drivers, hay hệ thống quản lý tệp (file system) như những hệ điều hành nhúng cao cấp khác. Tuy vậy, FreeRTOS có nhiều ưu điểm, hỗ trợ nhiều kiến trúc vi điều khiển khác nhau, kích thước nhỏ gọn (4.3 Kbytes sau khi biên dịch trên ARM7), được viết bằng ngôn ngữ C và có thể sử dụng, phát triển với nhiều trình biên dịch C khác nhau (GCC, OpenWatcom, Keil, IAR, Eclipse, …), cho phép không giới hạn các tác vụ chạy đồng thời, không hạn chế quyền ưu tiên thực thi, khả năng khai thác phần cứng. Ngoài ra, nó cũng cho phép triển khai các cơ chế điều độ giữa các tiến trình như: queues, counting semaphore, mutexes.

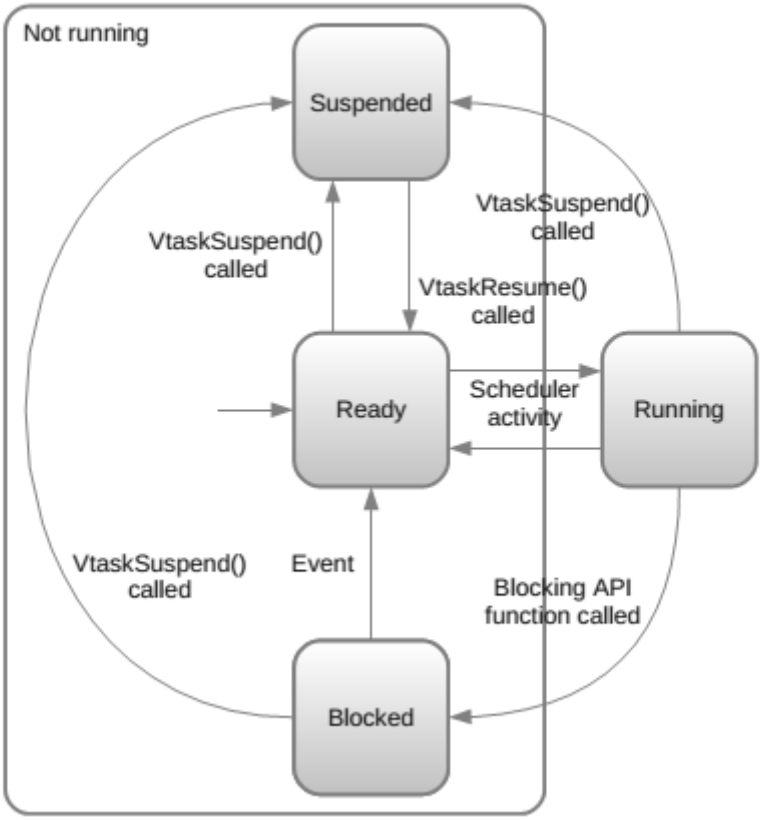
3.1.2. Quản lý tác vụ trong FreeRTOS

FreeRTOS cho phép một số lượng không giới hạn các tác vụ miễn là tài nguyên phần cứng và bộ nhớ vẫn còn đáp ứng được. Một tác vụ trong FreeRTOS được định nghĩa bởi một hàm C đơn giản, với đầu vào là một tham số void* và trả về kiểu void. FreeRTOS cung cấp các API có sẵn để quản lý tác vụ như là tạo tác vụ (vTaskCreate()), huỷ tác vụ (vTaskDelete()), quản lý mức ưu tiên của tác vụ (uxTaskResumeFreeRTOSomlISR())

Chu trình sống của một tác vụ là quá trình kể từ khi nó được tạo ra cho đến khi nó bị huỷ bỏ. Các tác vụ có thể rơi vào một trong hai trạng thái: "running" hoặc "not running". Có một vài lý do để các tác vụ còn lại không ở chế độ running. Trạng thái "not running" được diễn tả ở hình 21. Một tác vụ có thể bị ngăn không cho chạy có thể vì độ ưu tiên của nó thấp hơn các tác vụ khác, hoặc bị delay, hoặc là đang đợi event nào xảy ra. Khi một tác vụ có thể "running" nhưng nó đang đợi hệ điều hành cấp phát tài nguyên bộ vi xử lý, trạng thái đó được gọi là "ready". Điều này cũng có thể xảy ra khi một tác vụ đủ điều kiện để chuyển sang "running" nhưng lại có tác vụ khác có độ ưu tiên cao hơn đang chạy. Khi một tác vụ bị delay hoặc đang đợi tác vụ khác (đồng bộ hóa qua semaphores hoặc mutex) thì tác vụ đó được gọi là

"blocked". Và cuối cùng, khi một lời gọi tới vTaskSuspend(), vTaskResume() hoặc xTaskResumeFreeRTOSomISR() sẽ làm tác vụ chạy và thoát khỏi trạng thái "Suspend".

Điều quan trọng cần nhấn mạnh ở đây là khi một tác vụ ra khỏi trạng thái "Running" bởi chính nó (delay, suspend hoặc chờ đợi một sự kiện), chỉ có bộ lập lịch (scheduler) mới có thể chuyển trạng thái của nó trở lại về "running". Khi một tác vụ muốn trở về trạng thái "running" thì nó phải trở về trạng thái "ready" trước. Sau đó bộ lập lịch có thể chọn một trong các tác vụ ở trạng thái "ready" để chạy.



Hình 18 Chu trình sống của một tác vụ

Tạo và xóa một tác vụ:

Một tác vụ được định nghĩa bởi một hàm C đơn giản với tham số là một con trỏ void* và không cần giá trị trả về: *void MyTaskFunction(void *pvParameters);*
Thông thường đoạn chương trình tạo task đó có một vòng lặp vô hạn hoặc là có một lời gọi vTastDetroy(NULL) trước khi kết thúc.
Có thể tạo một task bằng hàm vTaskCreate(). Các tham số như sau:

- pvTaskCode: con trỏ tới hàm task
- pcName: Tên, không có ý nghĩa đối với hdh FreeRTOS và chỉ có mục đích là phục vụ debug

- usStackDepth: kích thước của stack tính bằng WORD. Kích thước thực sự của stack phục thuộc vào trình điều khiển thiết bị. Nếu stack là 32 bit (4byte) và usStackDepth là 100 thì kích thước stack sẽ là 4*100 = 400 byte.
- pvParameters: con trỏ tới các tham số đã được chứa trong stack, Cách tốt nhất là tạo một cấu trúc dữ liệu riêng, điền các tham số vào cấu trúc dữ liệu đó và truyền tham số là con trỏ tới cấu trúc đó.
- uxPriority: Độ ưu tiên cho task, một số giữa 0 tới MAX_PRIORITIES – 1
- pxCreatedTask: con trỏ tới một định danh cho phép xử lý task, nếu task không có xử lý sau này thì có thể để tham số này là NULL

3.2. Thu thập dữ liệu từ các cảm biến

Tác vụ thu thập dữ liệu từ cảm biến được khởi tạo ngay khi ESP khởi động xong, có nhiệm vụ định kỳ thu thập dữ liệu từ các cảm biến nói trên. Chu kỳ đọc dữ liệu cảm biến là một tham số có thể cấu hình thay đổi theo mong muốn, hiện tại lựa chọn đọc giá trị cảm biến sau mỗi chu kỳ 5 giây. Tất cả dữ liệu thu được từ cảm biến được lưu vào các biến dữ liệu. Khi ESP gửi dữ liệu lên server chính là gửi giá trị của các biến này.

3.2.1. Đọc nhiệt độ độ ẩm từ cảm biến SHT10

Task readSHT

Mục đích sử dụng:

Task này được sử dụng để đọc giá trị cảm biến nhiệt độ độ ẩm, với chu kỳ 5 giây, các giá trị được lưu vào một mảng buffer FIFO chứa tối đa 10 phần tử và sẽ lấy giá trị trung bình các kết quả đo trong mảng để gửi dữ liệu lên server

Quy định bản tin

Bản tin bao gồm 3 bit địa chỉ (bắt buộc là 000) và năm bit mã lệnh được mô tả như sau

Nội dung	Mã lệnh
Lệnh dự phòng	0000x
Đo nhiệt độ	00011
Đo độ ẩm	00101
Đọc thanh ghi trạng thái	00111
Ghi vào thanh ghi trạng thái	00110
Lệnh dự phòng	0101x – 1110x
Khởi động lại (Soft reset), hủy kết nối, xóa thanh ghi trạng thái về mặc định, cần đợi ít nhất 11ms trước khi gửi lệnh tiếp theo	11110

Bảng 6 Quy định bản tin SHT10

Để khởi tạo kết nối với sensor cần hạ chân DATA xuống mức thấp trong khoảng giữa hai mức cao liên tiếp của xung SCK.

2 bytes dữ liệu và 1 byte checksum (CRC-8) (tùy chọn) sẽ được phản hồi lại cho MCU. Tất cả giá trị lưu dạng MSB, nếu là kết quả đo 8 bit thì byte đầu tiên không được sử dụng.

3.2.2. Đọc độ bụi từ cảm biến SDS011

Task readSDS

Mục đích sử dụng:

Task này được sử dụng để đọc giá trị cảm biến độ bụi, với chu kỳ 5giây, các giá trị được lưu vào một mảng buffer FIFO chứa tối đa 10 phần tử và sẽ lấy giá trị trung bình các kết quả đo trong mảng để gửi dữ liệu lên server

Quy định bản tin:

Trong đồ án này sử dụng giao tiếp UART cho độ chính xác của dữ liệu tốt hơn so với đọc PWM. Giao thức UART cấu hình như sau:

- Bit rate : 9600
- Data bit : 8
- Parity bit : NO
- Stop bit : 1
- Data Packet frequency: 1Hz

Cấu trúc bản tin nhận được được mô tả trong bảng sau:

Số thứ tự byte	Nội dung quy định	Nội dung
0	Header bản tin	AA
1	Mã lệnh	C0
2	DATA 1	PM2.5 low byte
3	DATA 2	PM2.5 high byte
4	DATA 3	PM10 low byte
5	DATA 4	PM10 high byte
6	DATA 5	ID byte 1
7	DATA 6	ID byte 2
8	Check sum	Check sum
9	Ký tự cuối bản tin	AB

Check-sum: Check-sum=DATA1+DATA2+...+DATA6

Giá trị đo PM2.5 (µg /m3) = ((PM2.5 high byte *256) + PM2.5 low byte)/10

Giá trị đo PM10 (µg /m3) = ((PM10 high byte*256) + PM10 low byte)/10

3.3. Tác vụ đọc dữ liệu GPS

Mục đích sử dụng:

Task này được sử dụng để lắng nghe các bản tin từ cổng serial mà module GPS trả về. Do module GPS NEO-6M liên tục gửi các bản tin qua cổng serial nên nếu chỉ thực hiện tuần tự việc đọc GPS rồi mới thực hiện công việc tiếp theo có thể làm block chương trình do trong serial buffer luôn có dữ liệu.

Các gói tin từ module GPS trả về theo định dạng NMEA.

NMEA is viết tắt cuả National Marine Electronics Association nghĩa là Hiệp hội Điện tử Hàng hải Quốc gia. NMEA đã xuất hiện trước khi GPS được phát minh. Mục đích của NMEA là cung cấp cho người dùng thiết bị khả năng trộn và kết hợp phần cứng và phần mềm. Dữ liệu GPS được định dạng NMEA cũng giúp các nhà phát triển phần mềm dễ dàng viết phần mềm cho nhiều loại máy thu GPS hơn thay vì phải viết giao diện tùy chỉnh cho mỗi máy thu GPS. Ngày nay trong thế giới của GPS, NMEA là định dạng dữ liệu tiêu chuẩn được hỗ trợ bởi tất cả các nhà sản xuất GPS, giống như ASCII là tiêu chuẩn cho các ký tự máy tính kỹ thuật số trong thế giới máy tính.

Ví dụ bản tin GPRMC:

$GPRMC, 030742.00, A, 2232.73830,N, 11404.58520, E, 0.356, , 070314, , ,A*77.

030742.00	Giờ UTC. 3 giờ 7 phút 42 giây
A	A - Trạng thái dữ liệu đúng, V - dữ liệu lỗi
2232.73830	Giá trị vĩ độ: 22 độ 32.73830 phút
N	N - vĩ độ Bắc, S – vĩ độ Nam
11404.58520	Giá trị kinh độ: 114 độ 4.58520 phút
E	E – kinh độ Đông, W – kinh độ Nam
0.356	Tốc độ quay mặt đất. 1 knot = 0.514444 m/s
070314	Ngày theo giờ UTC; ngày 7 tháng 3 năm 2014
A	Chế độ hoạt động, A – tự động, D – DGPS, E - DR
77	Checksum; Giá trị được tính bằng phép XOR tất cả byte giữa 2 ký tự $ và *

3.4. Tác vụ chụp ảnh lưu vào bộ nhớ flash

Mục đích sử dụng:

Thiết bị sẽ định kỳ thực hiện chụp ảnh thông qua camera và lưu trữ dữ liệu vào bộ nhớ flash, sau đó sẽ đọc lại dữ liệu từ flash để gửi hình ảnh lên server

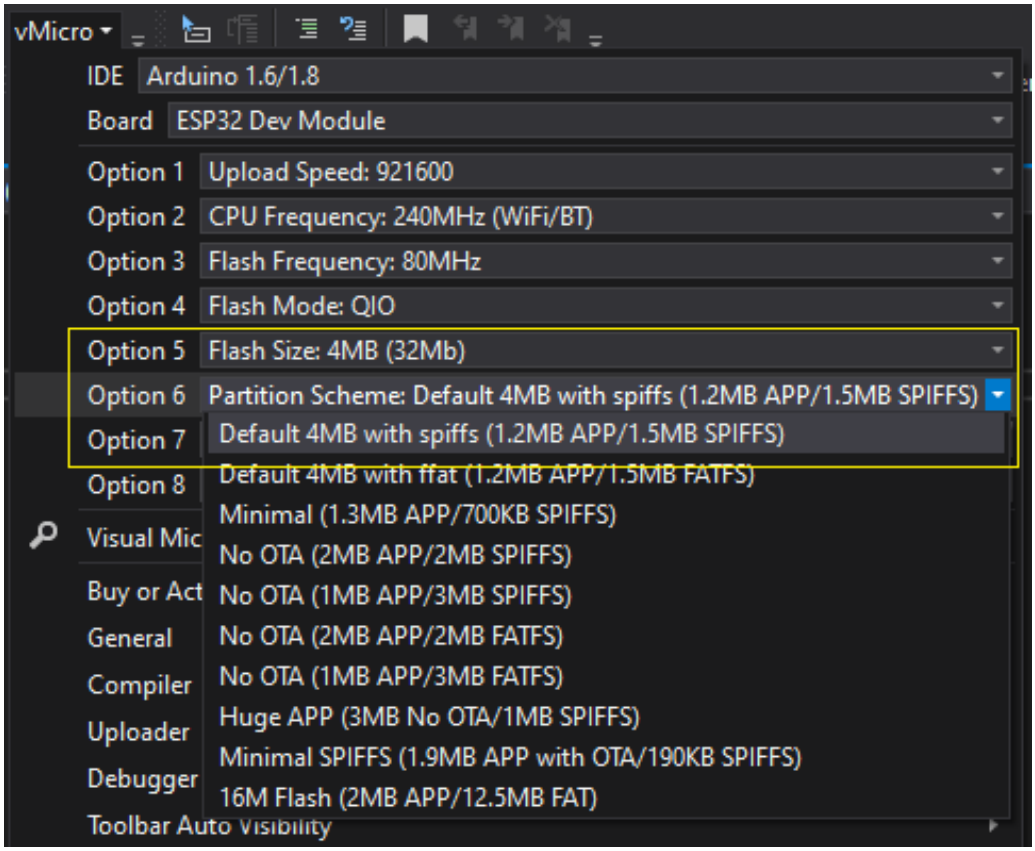
3.5. Tác vụ update firmware qua OTA

Câu lệnh điều khiển:

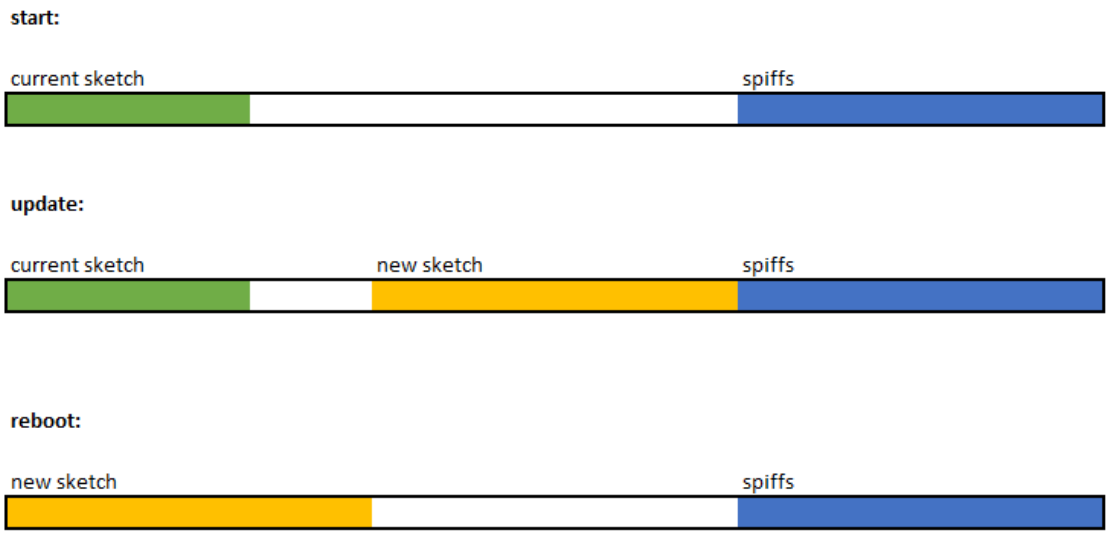
Khi nhận lệnh update firmware từ phía server, thiết bị sẽ tự động tải file firmware đã được biên dịch theo đường dẫn được cấu hình trong bản tin. File firmware này sẽ được lưu vào bộ nhớ flash.

Thư viện "Update.h" trên Arduino hỗ trợ việc sử dụng file firmware đã được biên dịch lưu trong bộ nhớ SPIFFS để ghi vào vùng nhớ trống còn lại giữa vùng nhớ chương trình và vùng nhớ SPIFFS. Sau khi thực hiện lệnh reset, bootloader trên ESP32 sẽ thực hiện kiểm tra và thực hiện việc ghi đè firmware mới vào vùng nhớ mặc định của firmware (ghi đèn lên firmware cũ). Sau khi hoàn tất các bước như vậy thì việc update firmware đã thành công.

Lưu ý: cần cấu hình vùng nhớ Flash có chế độ OTA



Hình 19 Lưu ý cấu hình flash có OTA



Hình 20 Mô tả vùng nhớ flash khi thực hiện quá trình update firmware

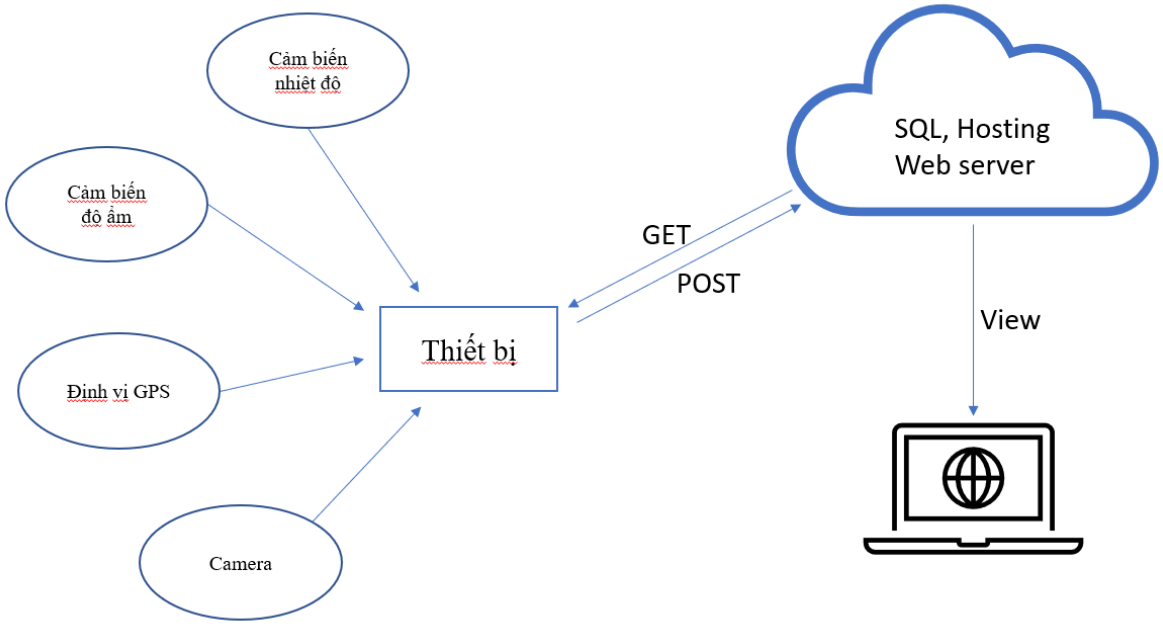
3.6. Tác vụ kết nối và trao đổi dữ liệu với Server

Mục đích sử dụng:

Để tạo kết nối, duy trì kết nối cũng như là nhận lệnh điều khiển từ phía server

CHƯƠNG 4. Xây dựng website

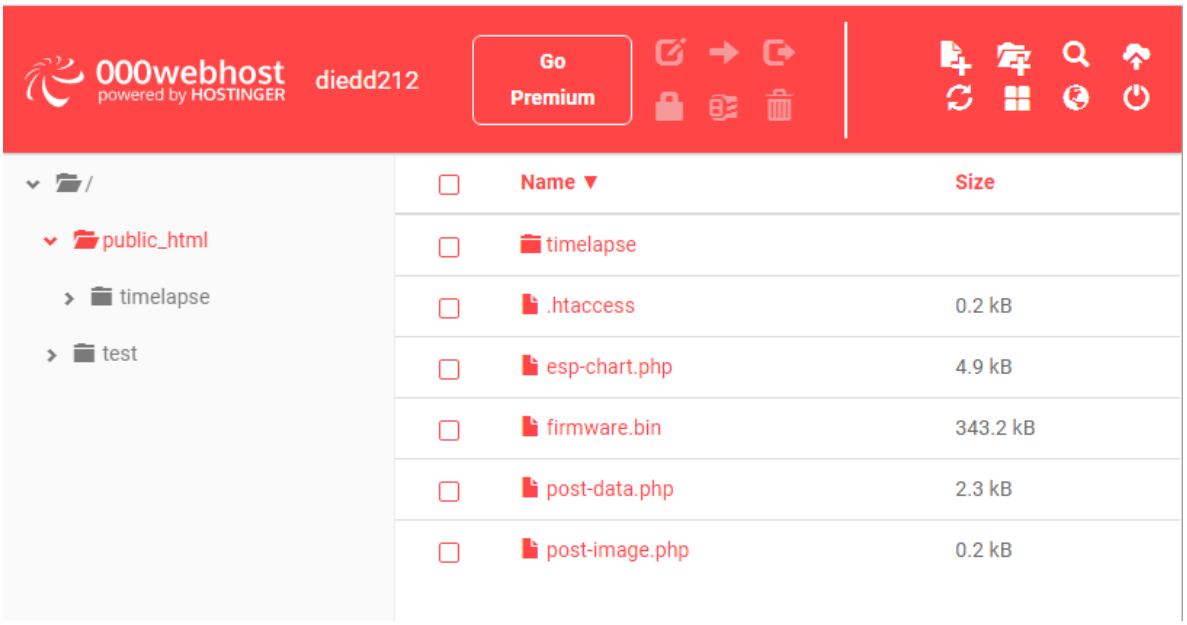
4.1. Mô hình tổng quan



Hình 21 Mô hình tổng quan website

Thiết bị thu thập dữ liệu từ cảm biến và camera gửi dữ liệu lên server thông qua các bản tin HTTP Request. Hệ thống sẽ hiển thị dữ liệu nhận được trên website nhằm mục đích theo dõi sự thay đổi của chất lượng không khí, tính toán hệ số AQI (pm25) nhằm đưa ra mức cảnh báo về chất lượng không khí. Việc tính toán và công bố chỉ số chất lượng không khí theo tiêu chuẩn VN_AQI (Số 1459/QĐ-TCMT).

4.2. Triển khai



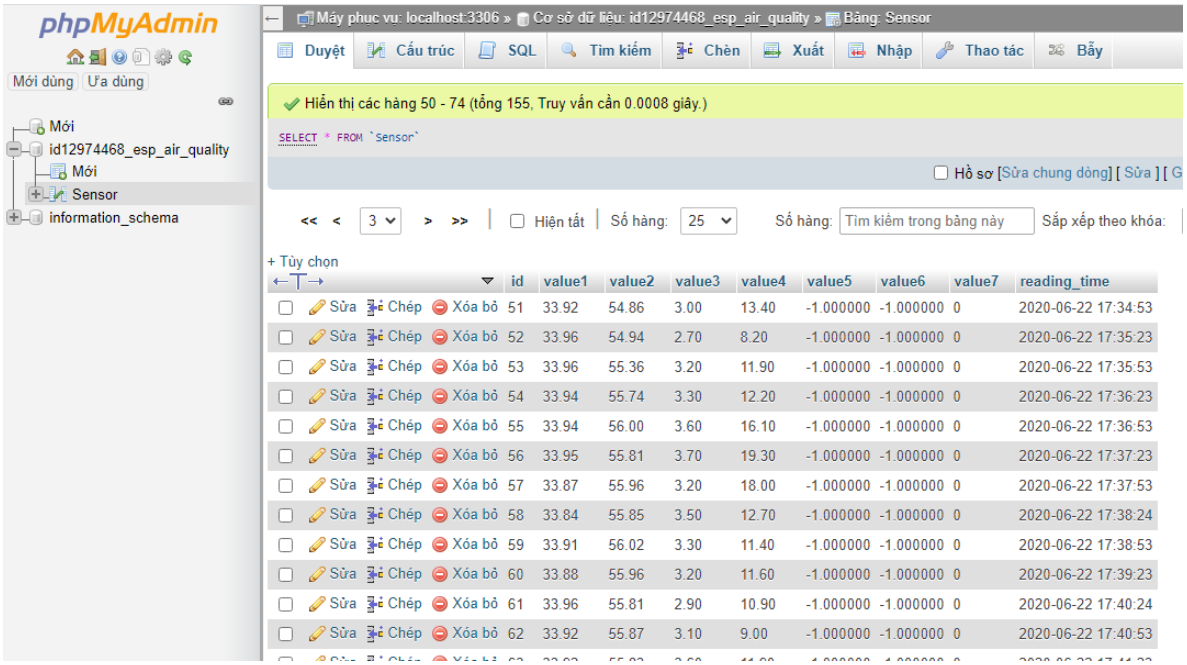
Hình 22 Các file mã nguồn phía server

Trong đó bao gồm:

post-image.php: Thực hiện lưu hình ảnh từ thiết bị gửi lên vào thư mục timelapse

post-data.php: Thực hiện lưu dữ liệu từ thiết bị gửi lên vào CSDL

esp-chart.php: Hiển thị dữ liệu từ CSDL lên website, tính toán giá trị AQI và đưa ra mức cảnh báo.



Hình 23 CSDL lưu trên server

Công thức tính AQI giờ theo thông số PM25, PM10 được tính như sau:

$$AQI_x = \frac{I_{i+1} - I_i}{BP_{i+1} - BP_i}\left(Nowcast_x - BP_i\right) + I_i$$

Tính giá trị Nowcast đối với thông số PM2.5 và PM10

Gọi $c_1, c_2, \ldots c_{12}$ là 12 giá trị quan trắc trung bình 1 giờ (với c_1 là giá trị quan trắc trung bình 1 giờ hiện tại, c_{12} là giá trị quan trắc trung bình 1 giờ cách 12 giờ so với hiện tại).

Tính giá trị trọng số: $w^* = \frac{c_{min}}{c_{max}}$

Trong đó *Cmin* là giá trị nhỏ nhất trong số 12 giá trị trung bình 1 giờ

Cmax là giá trị lớn nhất trong số 12 giá trị trung bình 1 giờ

Nếu $w^* \leq \frac{1}{2}$ Thì lấy $w = \frac{1}{2}$

Nếu $w^* > \frac{1}{2}$ Thì lấy w=w*

Trong trường hợp $w > \frac{1}{2}$ thì giá trị $Nowcast = \frac{\sum_{i=1}^{12} w^{i-1} c_i}{\sum_{i=1}^{12} w^{i-1}}$

Trong trường hợp $w = \frac{1}{2}$ thì $Nowcast = \frac{1}{2}c_1 + \left(\frac{1}{2}\right)^2 c_2 + \cdots + \left(\frac{1}{2}\right)^{12} c_{12}$

Chú ý:

- Nếu có ít nhất 2 trong 3 giá trị c_1, c_2, c_3 có dữ liệu thì mới tính được giá trị Nowcast, ngược lại coi như "không có dữ liệu" (không tính được giá trị Nowcast).

- Nếu c_i không có giá trị thì lấy $w^{i-1} = 0$

i	**Ii**	**PM_{10}**	**$PM_{2.5}$**
1	0	0	0
2	50	50	25
3	100	150	50
4	150	250	80
5	200	350	150

6	300	420	250
7	400	500	350
8	500	≥600	≥500

Bảng 7 Giá trị BPi quy định đối với từng thông số (Đơn vị: µg/m3)

Khoảng giá trị AQI	**Chất lượng không khí**	**Màu sắc**	**Mã màu RBG**
0 - 50	Tốt	Xanh	0;228;0
51 - 100	Trung bình	Vàng	255;255;0
101 - 150	Kém	Da cam	255;126;0
151 - 200	Xấu	Đỏ	255;0;0
201 - 300	Rất xấu	Tím	143;63;151
301-500	Nguy hại	Nâu	126;0;35

Bảng 8 Khoảng giá trị AQI và đánh giá chất lượng không khí

Tính toán mẫu

Tính giá trị Nowcast

Giả sử có bảng số liệu quan trắc $PM_{2.5}$ như sau:

09:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00	18:00	19:00	20:00
26,9	24,7	20,5	23,5	19,5	16,5	19,0	16,5	20,3	22,4	19,6	20,6

Tính giá trị $w* = \frac{C_{min}}{C_{max}} = \frac{16,5}{26,9} = 0,61$

Do w*=0,61 > 0,5 vì vậy lấy w=w*=0,61

$$Nowcast = \frac{0,61^0 \times 20,6 + 0,61^1 \times 19,6 + 0,61^2 \times 22,4 + ... + 0,61^{11} \times 26,9}{0,61^0 + 0,61^1 + 0,61^2 + ... + 0,61^{11}} = 20,3\left(\mu g / m^3\right)$$

b. Tính giá trị AQI giờ

Giả sử có bảng số liệu quan trắc trung bình 1 giờ như sau:

O3 *(µg/m^3)*	NO2 *(µg/m^3)*	Nowcast (PM2.5) *(µg/m^3)*
136,1	118,7	20,3

Tính toán các giá trị AQI thông số như sau:

$$AQI_{O_3} = \frac{50-0}{160-0}(136,1-0)+0 = 43$$

$$AQI_{NO_2} = \frac{100-50}{200-100}(118,7-100)+50 = 60$$

$$AQI_{PM_{2,5}} = \frac{50-0}{25-0}(20,3-0)+0 = 41$$

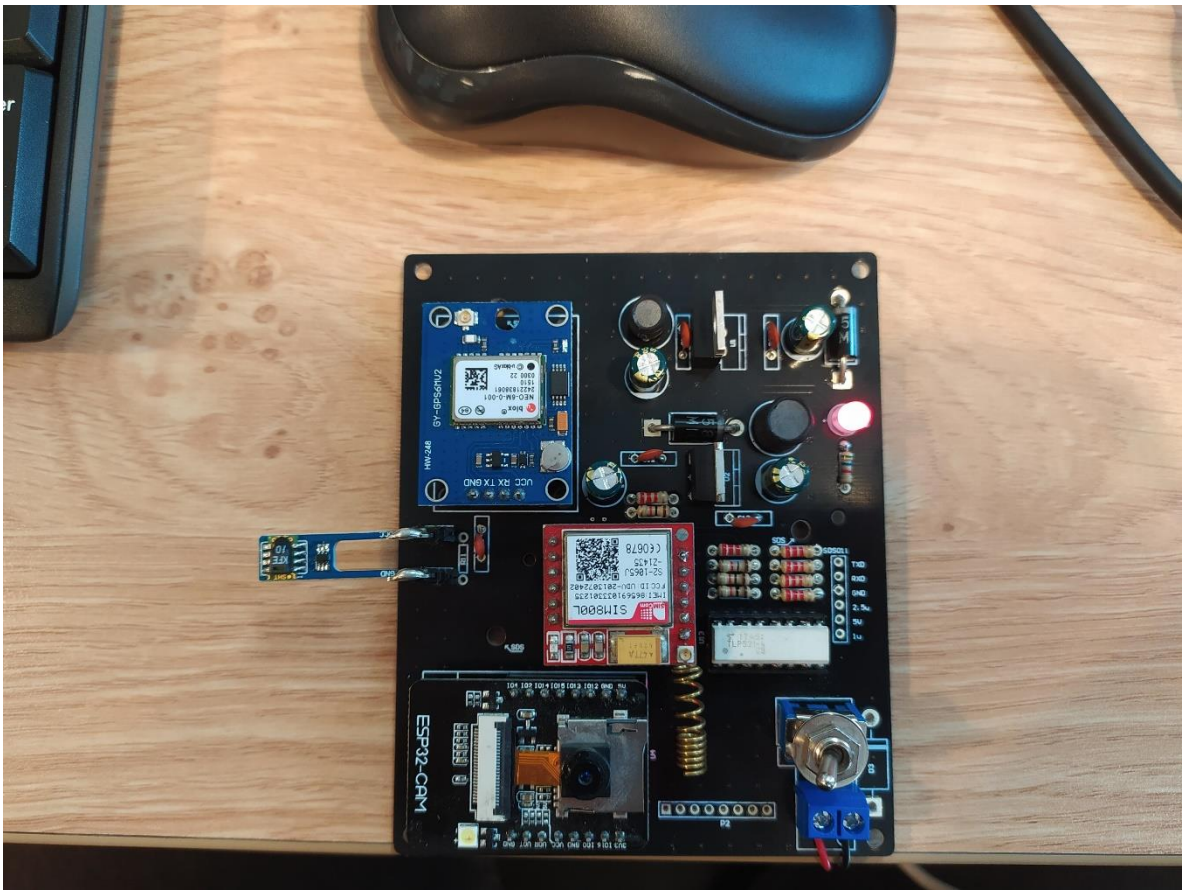
Như vậy giá trị AQIh = 60.

4.3. Kiểm thử

CHƯƠNG 5. KẾT QUẢ ĐẠT ĐƯỢC

5.3. Kết quả thiết kế phần cứng thiết bị

Dưới đây là hình ảnh thực tế của thiết bị:



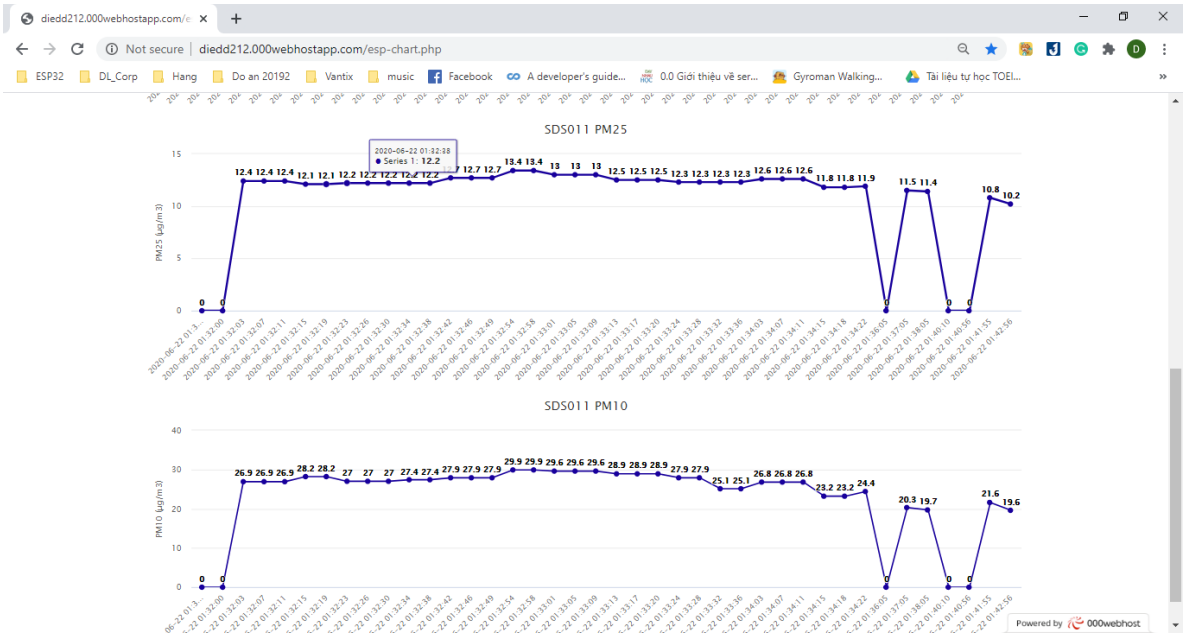
Hình 24 Thiết bị thực tế (chụp lại)

Chú thích:

1. Khối nguồn, cắm adapter 12V
2. Khối hiển thị, sử dụng màn hình LCD 16x02
3. Khối điều khiển: Sử dụng 4 relay 12V/3A
4. Module ESP32
5. Khối cảm biến: gồm 2 cổng 1-wire, 2 cổng ADC, 1 cổng I2C
6. Nút bấm để khởi động config mode và đèn báo

5.4. Kết quả hiển thị trên web

Link: http://diedd212.000webhostapp.com/esp-chart.php



Hình 25 Website hiển thị (chụp lại)

CHƯƠNG 6. KẾT LUẬN VÀ ĐÁNH GIÁ

5.1. Kết luận

5.2. Định hướng phát triển

CHƯƠNG 7. TÀI LIỆU THAM KHẢO

[1]. Embedded247, "Tìm hiểu hệ điều hành nhúng FreeRTOS", 2014. https://sites.google.com/site/embedded247/embedded_system/tim-hieu-he-dieu-hanh-nhung-freertos

[2]. Embedded247, "Tìm hiểu về FreeRTOS (2)", 2014.

https://sites.google.com/site/embedded247/embedded_system/tim-hieu-ve-freertos-2

[3] R2R-Digital -Analog-Converter-DAC

https://www.instructables.com/id/R2R-Digital-Analog-Converter-DAC/