

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

ĐỒ ÁN TỐT NGHIỆP

Xây dựng bộ firmware tiêu chuẩn và các công cụ hỗ trợ SIOT Platform cho hệ thống nhúng

Đồng Quang Linh

Linh.dq162385@sis.hust.edu.vn

Ngành Công nghệ thông tin

Giảng viên hướng dẫn: ThS. Nguyễn Đức Tiến

Chữ ký của GVHD

Bộ môn: Kỹ thuật máy tính

Viện: Công nghệ Thông tin và Truyền thông

HÀ NỘI, 01/2021

PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

1. Thông tin về sinh viên

Họ và tên sinh viên: Đồng Quang Linh

MSSV: 20162385

Điện thoại liên lạc: 0976268575

Email: linh.dq162385@sis.hust.edu.vn

Lóp: CNTT 1.1-K61

Hệ đào tạo: Chính quy

Thời gian làm ĐATN: Từ ngày 07/09/2020 đến ngày 08/01/2021

2. Mục đích nội dung của ĐATN

Xây dựng nền tảng, chuẩn hóa các thiết bị phần cứng trong SIot Platform

3. Các nhiệm vụ cụ thể của ĐATN

- Tìm hiểu và nghiên cứu các công nghệ kỹ thuật liên quan
- Xây dựng các bài toán liên quan đến kết nối Internet trên vi điều khiển
- Kết nối vi điều khiển với server của SIot Platform
- Gửi dữ liệu sensor lên server, cập nhật dữ liệu.
- Update firmware từ xa và bài toán liên quan đến nạp code tự động

4. Lời cam đoan của sinh viên

Tôi – *Đổng Quang Linh* - cam kết ĐATN là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của *Thạc sĩ Nguyễn Đức Tiến*. 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 01 năm 2021

Tác giả ĐATN

Đồng Quang Linh

5. Xác nhận của giáo viên hướng dẫn về mức độ hoàn thành của ĐATN và cho phép bảo vệ

Hà Nội, ngày tháng 01 năm 2021

Giáo viên hướng dẫn

ThS. Nguyễn Đức Tiến

TÓM TẮT NỘI DUNG ĐỒ ÁN

Trong những năm gần đây, cụm từ về “Cuộc cách mạng 4.0”, “Internet of things”, “Big Data”, ... xuất hiện thường xuyên và liên tục trên các phương tiện truyền thông và cũng đang len lỏi sâu rộng vào trong cuộc sống. Và “Data” là mấu chốt của mọi vấn đề. Trước thực tế đó, SIOT Platform ra đời với mục tiêu hợp chuẩn các thiết bị nhúng có thể dễ dàng tích hợp vào hệ thống, lưu trữ tập trung thông tin và có thể hiệu chỉnh nâng cấp dễ dàng. Mục tiêu trước mắt có thể áp dụng chúng vào trong các bài giảng của Viện, cũng áp dụng ở quy mô số lượng nhỏ thiết bị, để có thể là tiền đề cho việc nhân rộng với số lượng lớn.

Đồ án được chia thành những chương chính như sau:

Chương 1: Giới thiệu đề tài

Chương 2: Phân tích thiết kế

Chương 3: Phát triển mã nguồn siot core

Chương 4: Kết nối, cập nhật dữ liệu siot server

Chương 5: Kết quả đạt được

Trong mỗi chương sẽ trình bày chi tiết từng phần nhỏ hơn và các bước triển khai.

Người thực hiện đề tài

Đông Quang Linh

LỜI NÓI ĐẦU

Với tôi, Bách Khoa giống như một cuộc tình đầy trắc trở, lúc yêu lúc ghét. Ngay từ những năm đi học cấp một, cấp hai những chương trình về khoa học, về Robot đã dẫn tôi đến đam mê về công nghệ và cái tên Bách Khoa xuất hiện trong đầu từ ngày đó. Thời gian trôi qua có những lúc tình cảm lung lay, nhưng rồi cuối cùng tôi cũng đến Bách Khoa và Kỹ thuật máy tính là chuyên ngành mà tôi chọn.

Đến với đại học với bao lạ lẫm, chính các thầy và các bạn trong giảng đường đã khiến tôi quen hơn và hòa nhập với môi trường. Tôi xin gửi lời cảm ơn đến thầy Nguyễn Đức Tiến, có lẽ thầy là người đưa tôi đến với những kiến thức về “IT” đầu tiên với những bài giảng về nhập môn CNTT, về đạo đức máy tính. Chính thầy là người định hướng và xây dựng trong tôi rất nhiều ý tưởng về các bài toán, công nghệ và học được nhiều bài học không chỉ về chuyên môn mà còn là bài học về cuộc sống.

Tôi xin gửi lời cảm ơn đến các thầy, cô trong bộ môn Kỹ Thuật Máy Tính, thầy cô trong trường, với những bài giảng sâu sắc, sự tận tâm trong công việc. Khi làm việc với các thầy cô, tôi cảm nhận sự gần gũi, ân cần, không hề có chút sự xa cách.

Tôi xin gửi lời cảm ơn đến gia đình, người thân và bạn bè, những người luôn động viên, sát cánh cùng tôi. Luôn là chỗ dựa cho những lúc mệt nhọc, chán nản, và cũng là động lực để tôi luôn cố gắng.

Cảm ơn tất cả mọi người. Cảm ơn HUST, cảm ơn SOICT và cảm ơn DCE!

MỤC LỤC

PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP.....	i
TÓM TẮT NỘI DUNG ĐỒ ÁN	ii
LỜI NÓI ĐẦU	iii
MỤC LỤC	iv
DANH MỤC HÌNH VẼ	vi
DANH MỤC BẢNG	viii
DANH MỤC CÁC TỪ VIẾT TẮT.....	ix
Chương 1 GIỚI THIỆU VỀ ĐỀ TÀI	1
1.1 Đặt vấn đề	1
1.2 Ý tưởng phát triển sản phẩm	1
1.3 Một số thiết bị phần cứng sử dụng.....	3
1.3.1 Vi điều khiển ESP8266-12E.....	3
1.3.2 Cảm biến nhiệt độ, độ ẩm DHT22.....	4
1.3.3 Modul cảm biến bụi laser SDS011	5
1.4 Giao tiếp kết nối với ngoại vi, cảm biến	6
1.4.1 Giao tiếp UART.....	6
1.4.2 Giao tiếp I2C.....	7
1.4.3 Giao tiếp SPI.....	8
1.5 Công nghệ và công cụ sử dụng	9
Chương 2 PHÂN TÍCH THIẾT KẾ.....	10
2.1 Mô hình chung của hệ thống	10
2.2 Xác định tính năng chính của các thành phần trong SIoT Platform	11
2.3 Mô hình thử nghiệm thiết kế và triển khai.....	12
2.4 Thiết kế giải pháp kết nối.....	12
2.4.1 Đặt vấn đề	12
2.4.2 Mô hình kết nối.....	14

2.5 Kết nối VDK server của Siot Platform	14
2.6 Cập nhật phần mềm từ xa.....	16
2.7 Thiết kế giao diện và đóng gói SIOTCore	17
2.8 Thiết kế kit ứng dụng nền tảng SIOTPlatform	18
2.9 Công cụ nạp code tự động.....	19
Chương 3 PHÁT TRIỂN MÃ NGUỒN SIOT CORE	22
3.1 Cài đặt Siot Core cho thiết bị nhúng.	22
3.2 Khai báo, định danh thiết bị	23
3.3 Kết nối Siot Server	24
3.4 Đọc giá trị cảm biến, ngoại vi	27
3.5 Công cụ debug.....	27
Chương 4 KẾT NỐI, CẬP NHẬT DỮ LIỆU SIOT SERVER	29
4.1 Tạo thiết bị mới trên Siot server	29
4.2 Cập nhật dữ liệu	31
Chương 5 KẾT QUẢ ĐẠT ĐƯỢC.....	32
5.1 Cập nhật firmware thành công	32
5.2 Cập nhật dữ liệu lên SIOT Platform	32
5.3 Thiết kế phần cứng.....	33
5.4 Định hướng phát triển đề tài	35
5.5 Kết luận đề tài	35
TÀI LIỆU THAM KHẢO	37

DANH MỤC HÌNH VẼ

Hình 1.3.1 Sơ đồ chân Esp8266-12e.....	3
Hình 1.3.2 Cảm biến nhiệt độ, độ ẩm DHT22	5
Hình 1.3.3 Cảm biến bụi SDS011	5
Hình 1.4.1 Sơ đồ kết nối VDK với ngoại vi theo chuẩn UART	7
Hình 1.4.2 Sơ đồ BUS I2c.....	8
Hình 1.4.3 Giao tiếp SPI giữa VDK và các thiết bị khác	9
Hình 2.1.1 Mô hình chung của hệ thống.....	10
Hình 2.2.1 Tính năng của các thành phần Siot Platform	11
Hình 2.4.1 Mô hình chung của hệ thống.....	13
Hình 2.4.2 Sơ đồ kết nối internet.	14
Hình 2.5.1 Mô hình kết nối VDK với Server.....	15
Hình 2.6.1 Mô hình cập nhật firmware OTA cho thiết bị.....	16
Hình 2.6.2 Cách ghi lưu firmware trên bộ nhớ Flash	17
Hình 2.8.1 Khởi nguồn và ESP8266	19
Hình 2.9.1 Mô hình hoạt động nạp code.....	20
Hình 2.9.2 Phần mềm nạp code	21
Hình 3.1.1 Cấu trúc mẫu khi dùng Siot core.....	22
Hình 3.2.1 Mô hình tạo mã định danh cho thiết bị	23
Hình 3.2.2 Hàm lấy mã định danh cho thiết bị	24
Hình 3.3.1 Define các giá trị đường dẫn API	25
Hình 3.3.2 Hàm cập nhật firmware.....	26
Hình 3.4.1 Class sensor chung	27
Hình 3.5.1 Hàm log chuỗi ra màn hình Oled	28
Hình 3.5.2 Hiển thị màn hình debug khi cập nhật firmware.....	28
Hình 4.1.1 Giao diện tạo thiết bị mới trên siot core.....	29
Hình 4.1.2 Các thông tin cảm biến của thiết bị.....	30
Hình 4.1.3 Thông tin về API của thiết bị	30
Hình 4.2.1 Các thông tin trong request tới Siot server.....	31
Hình 5.1.1 Hiển thị quá trình cập nhật firmware	32
Hình 5.2.1 Đồ thị dữ liệu độ ẩm theo thời gian	32

Hình 5.2.2 Bảng dữ liệu theo ngày	33
Hình 5.2.3 Đồ thị phân bố các giá trị về nhiệt độ	33
Hình 5.3.1 Sơ đồ schematic Siot board.....	34
Hình 5.3.2 Mô phỏng 3D Siot Board	35

DANH MỤC BẢNG

Bảng 2.5-1 Bảng các HTTP_CODE hay gặp.....	15
Bảng 3.2-1 Các trường thông tin của thiết bị.....	23
Bảng 3.3-1 Các trường thông tin của 1 request	24
Bảng 3.3-2 Giá trị mẫu 1 request	26

DANH MỤC CÁC TỪ VIẾT TẮT

VĐK	Vi điều khiển
SSID	Service Set Identifier
IOT	Internet of Things
STA	Station
AP	Access Point
MOSI	Master Out – Slave In
MISO	Master In – Slave Out
SPI	Serial Peripheral Interface
I2C	Inter-Integrated Circuit
UART	Universal Asynchronous Receiver / Transmitter

Chương 1 GIỚI THIỆU VỀ ĐỀ TÀI

1.1 Đặt vấn đề

Trong bối cảnh cuộc cách mạng công nghiệp 4.0, các thiết bị thông minh được sản xuất và ra đời rất đa dạng, phong phú về chủng loại mẫu mã, đáp ứng hầu hết các nhu cầu của cuộc sống: học tập, y tế, việc nhà, giao thông,... Tuy nhiên còn rất nhiều hạn chế và thiếu sót.

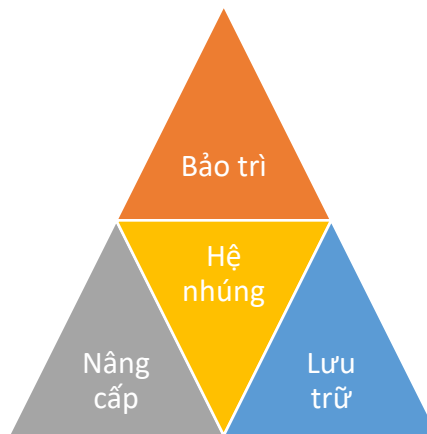
Các thiết bị được phát triển một cách đơn lẻ sử dụng công nghệ khác nhau dẫn đến dữ liệu bị phân mảnh, không được lưu trữ tập trung, khi cần dữ liệu để phát triển nghiên cứu sẽ gặp khó khăn. Mặt khác, các thiết bị hiện chỉ để thu thập các dữ liệu chuyên biệt, như vậy sẽ gây lãng phí các thiết bị phần cứng, trong khi có thể tích hợp nhiều loại cảm biến lên chúng.

Ngoài ra, việc phát triển một cách đơn lẻ không theo một quy chuẩn chung sẽ khó trong việc quản lí thiết bị trong các vấn đề về bảo trì, bảo dưỡng cũng như nâng cấp về sau. Trong khi đó với mỗi thiết bị thì điều đó là cần thiết và gần như bắt buộc. Việc quản lí còn liên quan đến vấn đề bảo mật và an toàn, các thông tin quan trọng nếu không được quản lí có thể sẽ gây hại khi bị sử dụng với mục đích không đúng với ý định ban đầu.

Với những yêu cầu đó, ý tưởng về SIot Platform được ra đời, nhằm mục đích có thể tích hợp dễ dàng vào tất cả thiết bị, để chúng có thể trở thành 1 node trong mạng Internet of Things (IOT).

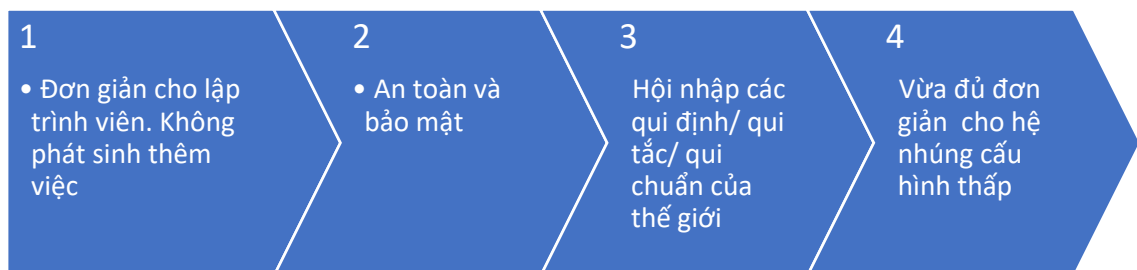
1.2 Ý tưởng phát triển sản phẩm

Với những khó khăn và thách thức, cũng như nhìn thấy rất nhiều các sản phẩm nhúng hay nhưng khó triển khai thực địa, nhu cầu áp dụng các tiêu chuẩn trong xây dựng hệ nhúng trở nên cấp thiết. Các tiêu chuẩn về các tính năng cần có, bắt buộc phải có... sẽ giúp các thiết bị nhúng dễ hoà nhập và đáp ứng nhu cầu của khách hàng của người dùng, về cả tính năng và sự tiện lợi.



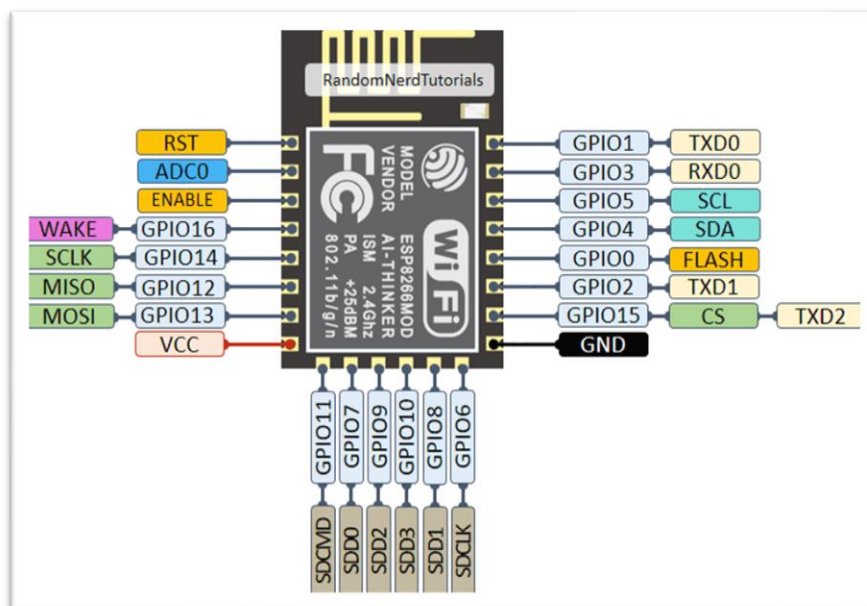
Đề tài này sẽ hi vọng sẽ xây dựng một nền tảng nhằm giúp cho lập trình viên hệ nhúng có được các tiện ích tốt nhất về 3 yếu tố: **bảo trì, nâng cấp và lưu trữ.**

Hơn thế nữa, nền tảng đó cần phải đáp ứng các tiêu chí:



1.3 Một số thiết bị phần cứng sử dụng

1.3.1 Vi điều khiển ESP8266-12E



Hình 1.3.1 Sơ đồ chân Esp8266-12e

Thu phát wifi ESP8266 12E là module wifi giá rẻ và được đánh giá rất cao cho các ứng dụng liên quan đến Internet và Wifi cũng như các ứng dụng truyền nhận sử dụng thay thế cho các module RF khác.

ESP8266 là một chip tích hợp cao, được thiết kế cho nhu cầu của một thế giới kết nối mới, thế giới Internet of thing (IOT). Nó cung cấp một giải pháp kết nối mạng Wi-Fi đầy đủ và khép kín, cho phép nó có thể lưu trữ các ứng dụng hoặc để giảm tải tất cả các chức năng kết nối mạng Wi-Fi từ một bộ xử lý ứng dụng.

Mạch thu phát wifi ESP8266 12E có khả năng xử lý và lưu trữ mạnh mẽ cho phép nó được tích hợp với các bộ cảm biến, vi điều khiển và các thiết bị ứng dụng cụ thể khác thông qua GPIOs với một chi phí tối thiểu và một PCB tối thiểu.

ESP8266 12E có kích thước nhỏ gọn, ra chân đầy đủ của IC ESP8266, mạch được thiết kế và gia công chất lượng tốt với vỏ bọc kim loại chống nhiễu và anten Wifi PCB tích hợp cho khoảng các truyền xa và ổn định.

Thông số kỹ thuật:

- IC chính: Wifi SoC ESP8266
- Điện áp sử dụng: 3.0V~3.6V (Optimal 3.3V)
- Working current: $\approx 70\text{mA}$ (170mA MAX), standby $< 200\mu\text{A}$
- Dòng tiêu thụ 10 μA , dòng điện năng chờ $< 5 \mu\text{A}$

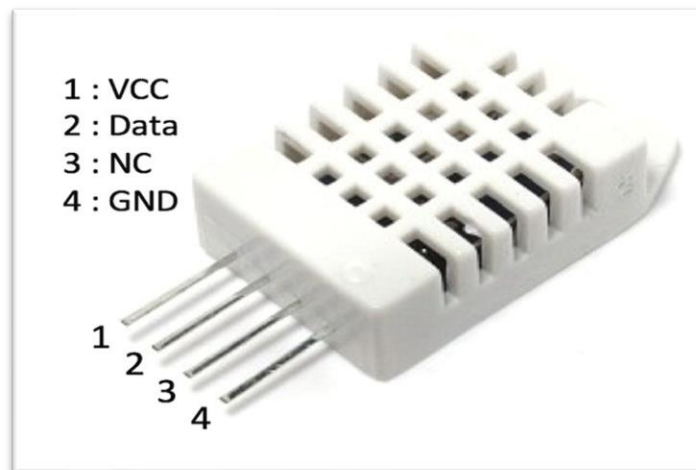
- 30 chân (10 GPIO, every GPIO can be PWM, I2C, 1-wire)
- MCU Frequency: 80-160 MHz, 32-bit micro MCU
- SRAM size: 36 KB
- ROM size: 4 MB (SPI External Flash)
- Antena on PCB
- Transmission data rate: 110-460800bps
- 10bit precision ADC pinout on board (0~1V)
- WiFi @ 2.4 GHz, supports WPA / WPA2 security mode
- Wi-Fi Connectivity (802.11 b/ g/ n)
- Support UART/GPIO data communication interface
- Support STA/AP/STA+AP 3 working modes
- Built-in TCP/IP protocol stack, maximum 5 clients
- Nhiệt độ làm việc: -40°C ~ +125°C
- Kích thước: 24 x 16 x 3mm
- Trọng lượng: 4g

1.3.2 Cảm biến nhiệt độ, độ ẩm DHT22

Cảm biến độ ẩm và nhiệt độ DHT22 là cảm biến rất thông dụng hiện nay được sản xuất bởi công ty Aosong Electronics, chi phí rẻ và rất dễ lấy dữ liệu thông qua giao tiếp 1 wire (giao tiếp digital 1 dây truyền dữ liệu duy nhất), so với DHT11 là phiên bản rẻ hơn thì DHT22 có độ chính xác cao hơn, khoảng đo rộng hơn DHT11 rất nhiều.

Thông số kỹ thuật:

- Nguồn cung cấp 3.3-6V DC
- Tín hiệu đầu ra tín hiệu kỹ thuật số thông qua một bus
- Phần tử cảm biến Tụ điện polyme
- Độ ẩm phạm vi hoạt động 0-100% RH; nhiệt độ -40 ~ 80 độ C
- Độ ẩm chính xác $\pm 2\%$ RH (Tối đa $\pm 5\%$ RH); nhiệt độ $< \pm 0,5$ độ C
- Độ phân giải hoặc độ nhạy 0,1% RH; nhiệt độ 0,1 độ C
- Độ ẩm lặp lại $\pm 1\%$ RH; nhiệt độ $\pm 0,2$ độ C
- Độ ẩm trễ $\pm 0,3\%$ RH
- Ổn định dài hạn $\pm 0,5\%$ RH / năm
- Thời gian cảm biến Trung bình: 2 giây



Hình 1.3.2 Cảm biến nhiệt độ, độ ẩm DHT22

1.3.3 Modul cảm biến bụi laser SDS011



Hình 1.3.3 Cảm biến bụi SDS011

SDS011 là loại cảm biến đo chất lượng không khí phổ biến được phát triển bởi Nova Fitness. Cảm biến bụi quang học này có thể đo chất lượng không khí rất chính xác bằng cách đo các hạt không khí hoặc bụi dựa trên phát hiện laser, có thể nhận được nồng độ hạt trong khoảng từ 0,3 đến 10 μ m trong không khí, nó kết nối với đầu ra kỹ thuật số và quạt tích hợp là ổn định và tin cậy.

Thông số kỹ thuật:

- Điện áp: 4.7 – 5.3V DC
- Công suất tiêu thụ: 70mA \pm 10mA (khi hoạt động), <4mA (chế độ ngủ của laser và quạt)
- Phạm vi đo: 0.0-999.9 g/m³

- Đầu ra PM2.5 và PM10
- Tuổi thọ của Laser: lên tới 8000 giờ (hoạt động liên tục)
- Dòng tối đa: 100mA
- Phạm vi nhiệt độ hoạt động: -20 ~ 500C
- Kích thước: 71x70x23 mm

Ngoài các loại cảm biến nêu trên, có thể sử dụng thêm bất kỳ một loại cảm biến hay thiết bị phần cứng khác (có hỗ trợ), có thể là một VDK khác thông qua các chuẩn truyền thông kết nối.

1.4 Giao tiếp kết nối với ngoại vi, cảm biến

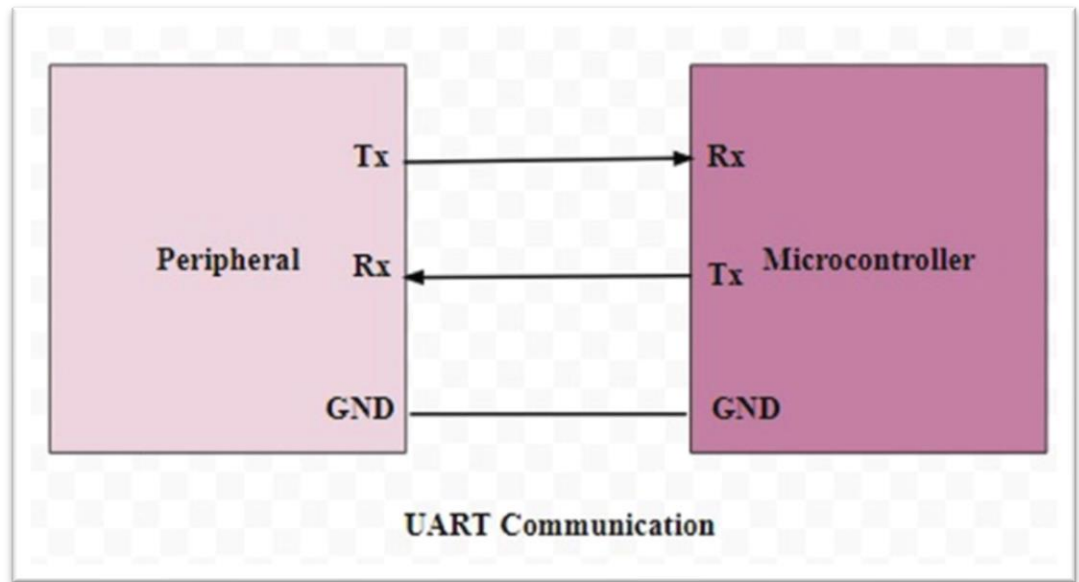
Số lượng ngoại vi đa dạng và rất nhiều chủng loại khác nhau, mỗi ngoại vi và sensor có những chuẩn giao tiếp khác nhau, do nhà sản xuất quy định trong datasheet và tài liệu sử dụng. Vì vậy cần phải hợp chuẩn Siot core để có thể dễ dàng sử dụng các thiết bị ngoại vi và cảm biến.

VDK ESP8226 hỗ trợ các chuẩn giao tiếp truyền thông phổ biến như: UART, I2C, SPI, tín hiệu Analog, tín hiệu Digital.

Dưới đây là 3 loại giao tiếp được sử dụng trong Siot core.

1.4.1 Giao tiếp UART

UART là từ viết tắt của cụm từ tiếng anh Universal Asynchronous Receiver – Transmitter. UART là một mạch tích hợp được sử dụng trong việc truyền dẫn dữ liệu nối tiếp giữa VDK và các thiết bị ngoại vi khác (có hỗ trợ). Trên ESP8266 có 2 chân TX và RX lần lượt ứng với GPIO1 và GPIO3. Ngoài ra VDK còn hỗ trợ software uart nghĩa là hoàn toàn có thể cấu hình 2 chân khác đóng vai trò như Tx, Rx.



Hình 1.4.1 Sơ đồ kết nối VDK với ngoại vi theo chuẩn UART

Trong giao tiếp UART có các thông số chính:

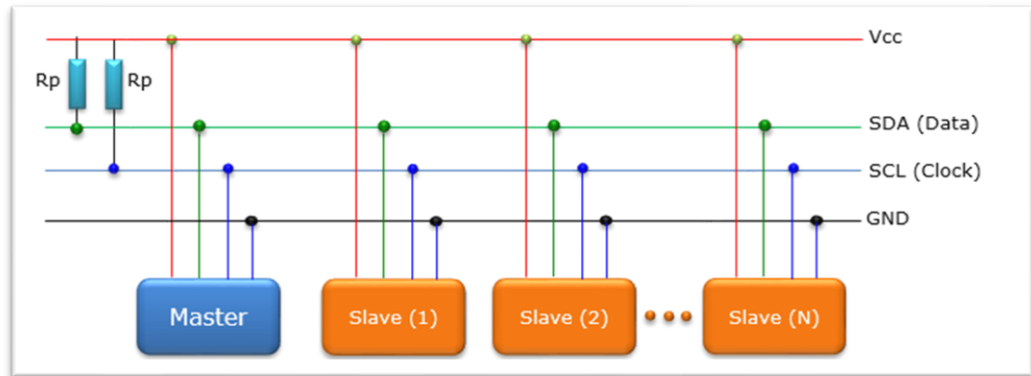
- Baud rate (tốc độ baud): khoảng thời gian để 1bit được truyền đi. Phải được cài đặt giống nhau ở cả phần gửi và nhận.
- Frame (khung truyền): khung truyền quy định về mỗi lần truyền bao nhiêu bit.
- Start bit: là bit đầu tiên được truyền trong 1 Frame. Báo hiệu cho thiết bị nhận có một gói dữ liệu sắp đc truyền đến. Đây là bit bắt buộc.
- Data: dữ liệu cần truyền. Bit có trọng số nhỏ nhất LSB được truyền trước sau đó đến bit MSB.
- Parity bit: kiểm tra dữ liệu truyền có đúng không.
- Stop bit: là 1 hoặc các bit báo cho thiết bị rằng các bit đã được gửi xong. Thiết bị nhận sẽ tiến hành kiểm tra khung truyền nhằm đảm bảo tính đúng đắn của dữ liệu. Đây là bit bắt buộc.

1.4.2 Giao tiếp I2C

I2C là tên viết tắt của cụm từ tiếng anh “Inter-Integrated Circuit”. Nó là một giao thức giao tiếp được phát triển bởi Philips Stôiiiconductors để truyền dữ liệu giữa một bộ xử lý trung tâm với nhiều IC trên cùng một board mạch chỉ sử dụng hai đường truyền tín hiệu. Trên VDK sử dụng 2 đường SDA và SCL lần lượt là GPIO5 và GPIO4

Do tính đơn giản của nó nên loại giao thức này được sử dụng rộng rãi cho giao tiếp giữa vi điều khiển và mảng cảm biến, các thiết bị hiển thị, thiết bị IoT, EEPROMs, v.v ...

Đây là một loại giao thức giao tiếp nối tiếp đồng bộ: các bit dữ liệu được truyền từng bit một theo các khoảng thời gian đều đặn được thiết lập bởi một tín hiệu đồng hồ tham chiếu.



Hình 1.4.2 Sơ đồ BUS I2C

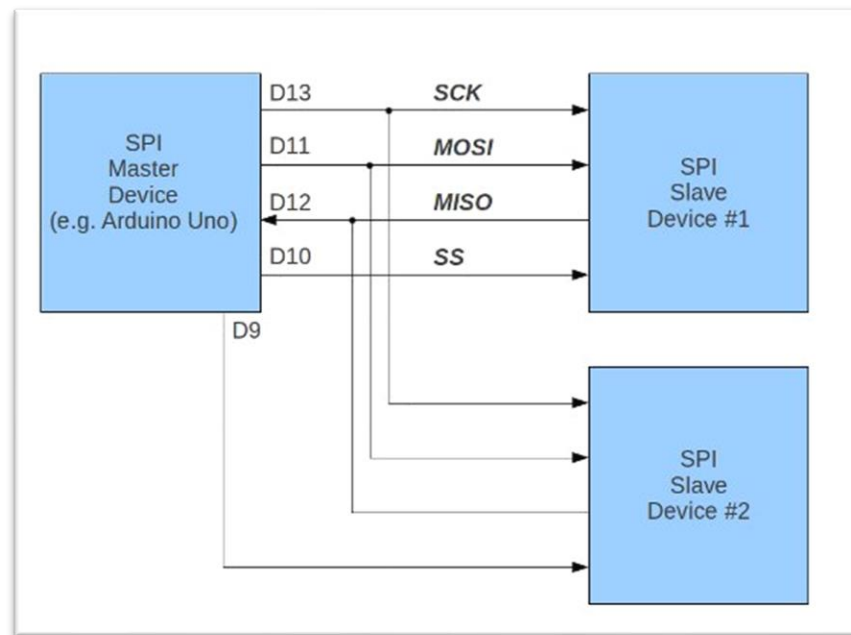
Theo sơ đồ trên có thể thấy, có thể kết nối nhiều thiết bị, cảm biến trên cùng một đường bus I2C, khi truyền chỉ cần phân biệt bằng địa chỉ của chúng.

1.4.3 Giao tiếp SPI

SPI (Serial Peripheral Bus) là một chuẩn truyền thông nối tiếp tốc độ cao do hãng Motorola đề xuất. Đây là kiểu truyền thông Master-Slave, trong đó có 1 chip Master điều phối quá trình truyền thông và các chip Slaves được điều khiển bởi Master vì thế truyền thông chỉ xảy ra giữa Master và Slave. SPI là một cách truyền song công (full duplex) nghĩa là tại cùng một thời điểm quá trình truyền và nhận có thể xảy ra đồng thời. Các chân có chức năng như sau

- **MISO** - Mang các dữ liệu từ các thiết bị SPI về VDK
- **MOSI** - Mang các dữ liệu từ VDK đến các thiết bị SPI
- **SS** - Chọn thiết bị SPI cần làm việc
- **SCK** - Dòng đồng bộ

Cũng giống như giao tiếp I2C, có thể dùng 1 VDK ESP8266 đóng vai trò là master kết nối với nhiều thiết bị (Slave) khác nhau. Tại một thời điểm, chỉ cho phép 1 thiết bị Slave hoạt động bằng cách kéo chân SS (SDA) xuống 0.



Hình 1.4.3 Giao tiếp SPI giữa VDK và các thiết bị khác

Ngoài những chuẩn giao tiếp trên, hoàn toàn người sử dụng có thể thêm các chuẩn khác bằng cách kế thừa các class mà siot core đã có sẵn.

1.5 Công nghệ và công cụ sử dụng

Để xây dựng và hoàn thiện đồ án, có sử dụng một số công nghệ và công cụ phát triển thích hợp cho từng yêu cầu và giai đoạn khi thực hiện.

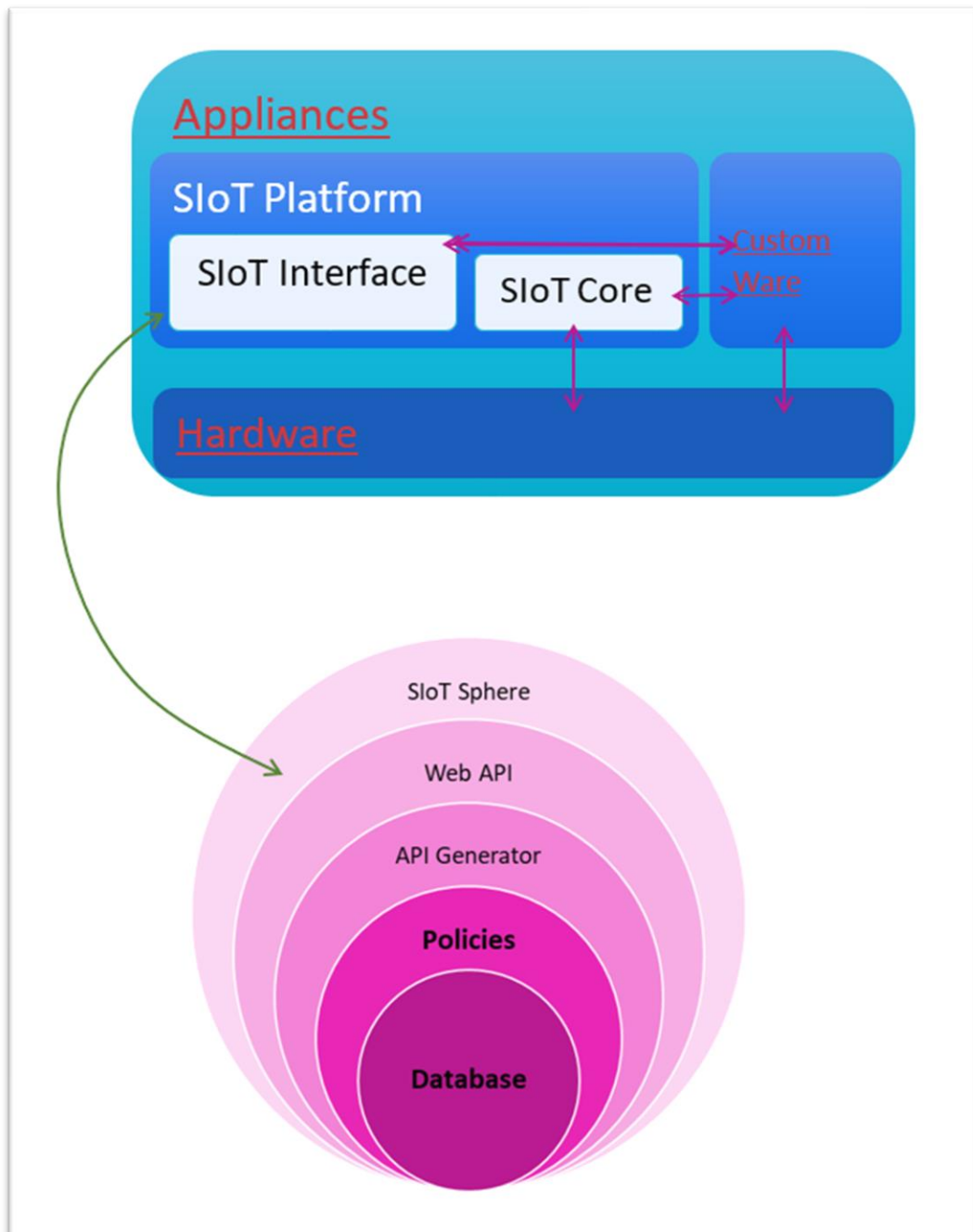
Lập trình phần core, sử dụng Arduino là nền tảng lập trình chính. Arduino là một trong những nền tảng lập trình rất phổ biến trong phát triển firmware hiện nay, cộng đồng phát triển tích cực, dễ tiếp cận. Ngôn ngữ lập trình chính là C/C++.

Trong quá trình phát triển, để kiểm tra các chức năng của từng module, có sử dụng Nodejs để tạo một server nhỏ, đảm bảo đủ các tính năng để test các chức năng, sử dụng IDE Visual Code.

Ngoài ra, có sử dụng Git và Github để quản lý toàn bộ mã nguồn của dự án.

Chương 2 PHÂN TÍCH THIẾT KẾ

2.1 Mô hình chung của hệ thống



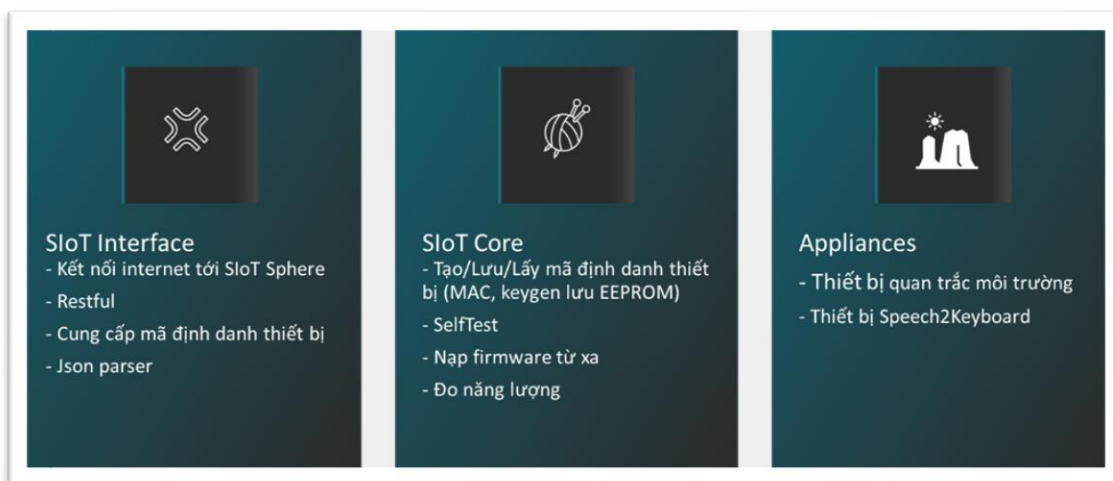
Hình 2.1.1 Mô hình chung của hệ thống

- Các thiết bị nhúng được tích hợp bộ thư viện firmware SIoT Platform để chúng có thể kết nối với hệ thống đám mây SIoT Sphere nhanh chóng.
- **SIoT Platform** bao gồm 2 thành phần
 - SIoT Interface: các hàm giao diện để triệu gọi các dịch vụ từ đám mây.
 - SIoT Core: các dịch vụ hạ tầng cơ bản cho thiết bị để khởi động, bảo trì, và kết thúc hoạt động.
- **SIoT Sphere** dịch vụ đám mây để lưu trữ tập trung mọi số liệu từ các thiết bị nhúng. Bao gồm:
 - Database: dữ liệu tập trung
 - Policies: quy chuẩn về các trường dữ liệu bắt buộc, ngầm định, cú pháp json...
 - API Generator: sinh các WebAPI dựa trên các trường dữ liệu do lập trình viên khai báo.
 - WebAPI: giao diện Restful kết nối với các thiết bị.

Đề tài này chỉ liên quan tới SIoT Platform và được hợp chuẩn để kết nối với WebAPI của SIoT Sphere.

2.2 Xác định tính năng chính của các thành phần trong SIoT Platform

SIoTCore cung cấp giao diện SelfTest vì đây là cơ sở để kiểm thử và xuất xưởng sản phẩm hoàn thiện, cũng như là công cụ giúp nhân viên kỹ thuật kiểm tra nhanh tình trạng sản phẩm.



Hình 2.2.1 Tính năng của các thành phần SIoT Platform

SIoT Core cũng có tính năng hết sức quan trọng là Nạp firmware từ xa. Với tốc độ phát triển vũ bão của các tính năng và độ phức tạp của các đoạn mã, việc một sản phẩm lỗi thời ngay từ khi xuất xưởng đã trở thành điều khá phổ biến. Vì vậy SIOT Core nhất thiết phải hỗ trợ giải pháp nạp firmware từ xa linh hoạt và phổ quát cho mọi trường hợp.

SIOT Interface cần phải phù hợp với các WebAPI URL do hệ thống SIOT Sphere khởi tạo ra. Và để cấu trúc thông tin nhỏ gọn, phù hợp với hệ nhúng, cấu trúc dữ liệu json sẽ được dùng. Không dùng XML.

2.3 Mô hình thử nghiệm thiết kế và triển khai

Theo thiết kế, các thành phần cơ bản của mô hình cần có như sau:

- Vi điều khiển dùng ESP8266 tối thiểu 1
- Router wifi hỗ trợ băng tần 2.4Ghz
- Webserver và Database: hiện tại đang sử dụng **Siot Sphere**. Ngoài ra trong quá trình phát triển có thể xây dựng sẵn một server test để tiện trong quá trình kiểm thử và sửa lỗi. Trong trường hợp đó, Docker là công nghệ được chọn, dễ triển khai, có tính đóng gói và dễ dàng deploy

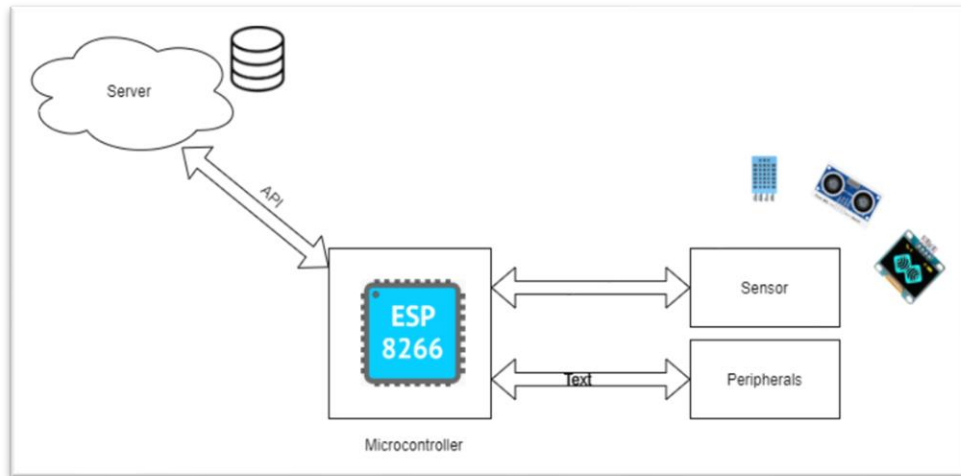
Tùy vào từng điều kiện thi triển khai, nếu trường hợp không có hỗ trợ Wifi thì các dữ liệu thu thập được đều được lưu trữ lại trong bộ nhớ trong, khi có kết nối dữ liệu sẽ được cập nhật lên cơ sở dữ liệu.

2.4 Thiết kế giải pháp kết nối

2.4.1 Đặt vấn đề

Với mỗi thiết bị, để có thể chia sẻ dữ liệu trong mạng IOT đều cần phải kết nối vào Internet. Với Siot core sử dụng VDK Esp8266 có hỗ trợ chuẩn kết nối Wifi ở băng tần 2,4GHz, vì vậy việc sử dụng thuận tiện và dễ dàng hơn. Tuy nhiên các yếu tố ảnh hưởng đến tốc độ đường truyền, smart config vẫn cần phải được xử lý.

Với những thiết bị được đặt ở vị trí cố định, cần phải đảm bảo có cơ chế tự động kết nối lại khi có các sự cố bất ngờ xảy ra như mất điện, mất mạng. Ngoài ra, với đường truyền kém, hoặc không ổn định cũng cần phải có cơ chế để thay đổi, lựa chọn đường truyền tốt hơn.



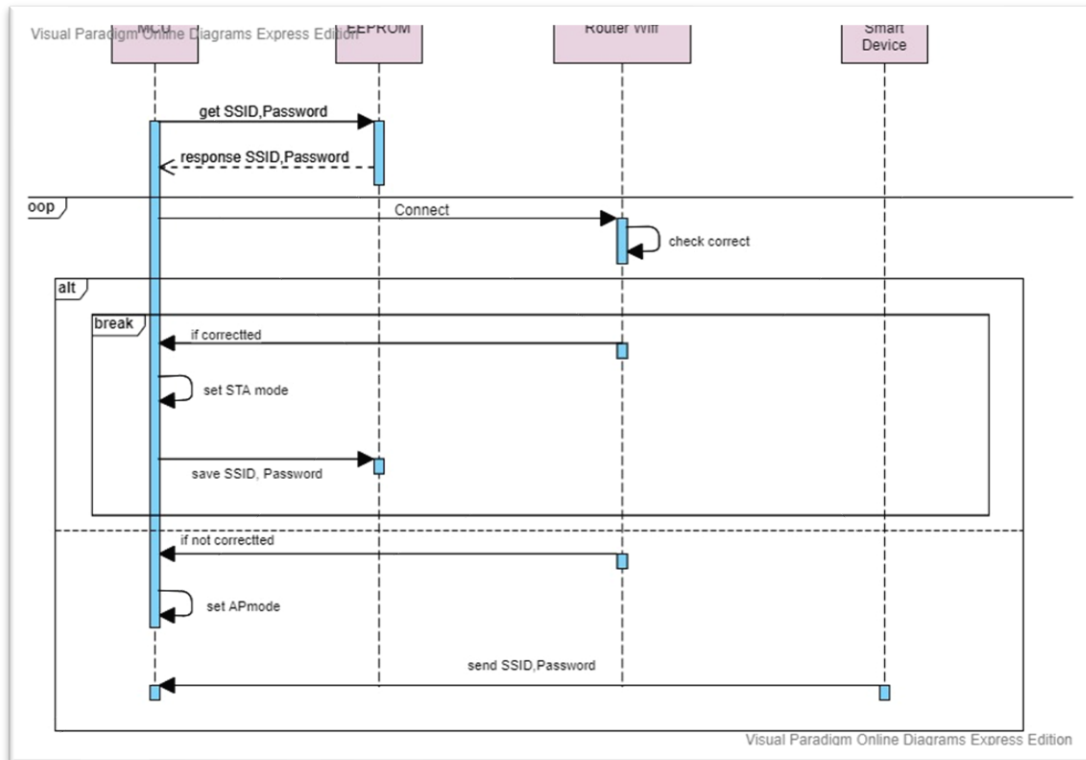
Hình 2.4.1 Mô hình chung của hệ thống

Mô hình trên gồm 3 phần chính:

- Khối core: Chứa vi điều khiển chính của mạch, dùng vi điều khiển ESP8266 – là một dòng vi mạch Wifi giá rẻ sản xuất bởi Espressif.
- Khối ngoại vi, sensor: Khối này là các thiết bị, các loại cảm biến có thể giao tiếp với VĐK, vi điều khiển có thể đọc được giá trị từ chúng qua các chuẩn giao tiếp thông dụng như I2c, UART, SPI, USB, Analog, ...
- Khối server: Khối này sẽ giao tiếp với VĐK thông qua các hàm API, dữ liệu từ VĐK có thể đưa lên qua các phương thức GET, POST, UPDATE,...Dữ liệu như nhiệt độ, độ ẩm sẽ được lưu trữ cho tùy mục đích sử dụng.

Ưu điểm của mô hình trên có thể dễ dàng nâng cấp mở rộng. Số lượng cảm biến, ngoại vi có thể thay đổi tùy ý, vì thế có thể đa dạng dữ liệu được thu thập. Ngoài ra, VĐK có thể dễ dàng đưa dữ liệu ra nhiều khối dữ liệu, server khác nhau, chỉ cần có API do phía server cung cấp.

2.4.2 Mô hình kết nối



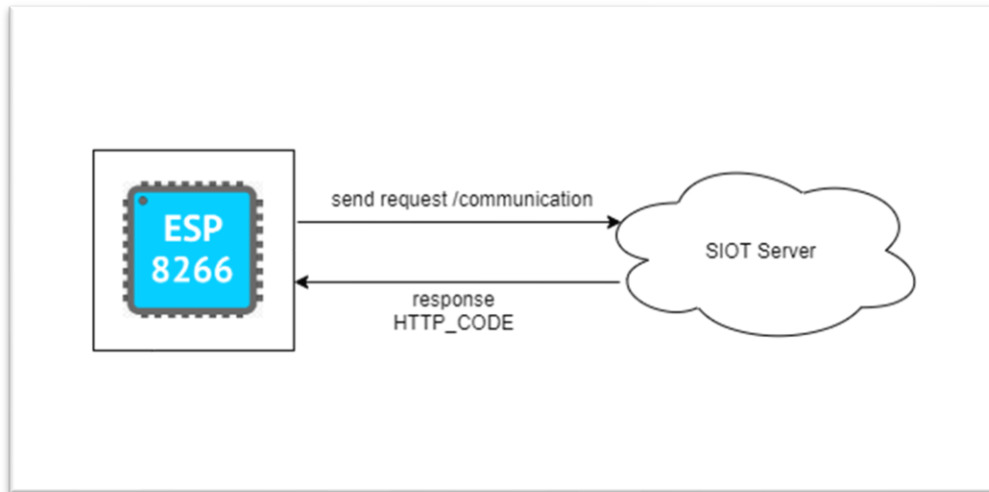
Hình 2.4.2 Sơ đồ kết nối internet.

Mô hình trên gồm 4 đối tượng chính: VDK, Bộ nhớ, Router wifi, Smart device (Laptop, Smart Phone). Luồng hoạt động của mô hình như sau:

1. Khi VDK khởi động chương trình để kết nối vào Internet, nó kiểm tra trong bộ nhớ (Flash/EEPROM) thông tin về SSID và Password của một mạng wifi đã tồn tại.
2. Với thông tin lấy được, nếu kết nối thành công, VDK điều khiển sẽ log ra tên Wifi mà nó kết nối.
3. Nếu bước 2 không thành công, VDK sẽ về chế độ STA mode, nó sẽ đóng vai trò là webbase-server cho phép các thiết bị khác kết nối vào, và đưa thông tin về SSID và Password để VDK có thể kết nối với 1 mạng phù hợp.
4. Quá trình lặp lại cho đến khi kết nối thành công. Thông tin sẽ được cập nhật lại bộ nhớ. Vì vậy, đảm bảo được tính năng cần tự động kết nối lại (khi có sự cố xảy ra).

2.5 Kết nối VDK server của Siot Platform

Sau khi VDK kết nối được với Internet, VDK sẽ được kết nối với Siot server để thực hiện các việc quan trọng như: lấy mã định danh, cập nhật các giá trị của sensor lên hệ thống, cập nhật phần mềm,... Dưới đây là mô hình của bài toán.



Hình 2.5.1 Mô hình kết nối VDK với Server

VDK gửi một request /communication để kiểm tra có giao tiếp được với server hay không. Dựa vào mã http status code trong phần response, có thể kiểm tra được đã kết nối thành công hay không, nếu có lỗi xảy ra, dựa vào mã code nhận được đều có thể debug được đã có lỗi gì. Dưới đây là bảng một số mã HTTP_STATUS_CODE hay gặp.

Bảng 2.5-1 Bảng các HTTP_CODE hay gặp

Code	Trạng thái
1xx	Khi nhận được những mã như vậy tức là request đã được server tiếp nhận và quá trình xử lý request đang được tiếp tục.
200	Request đã được tiếp nhận và xử lý thành công. Các Response thực tế trả về sẽ phụ thuộc vào phương thức HTTP của Request.
3xx	Mã trạng thái này cho biết client cần có thêm action để hoàn thành request.
400	Bad Request: Server không thể xử lý hoặc sẽ không xử lý các Request lỗi của phía client.
403	Forbidden: Request là hợp lệ nhưng server từ chối đáp ứng nó.
404	Not Found: Các tài nguyên hiện tại không được tìm thấy nhưng có thể có trong tương lai. Các request tiếp theo của Client được chấp nhận.
5xx	Server Error: Server thất bại với việc thực hiện một request nhìn như có vẻ khả thi.

Khi kết nối thành công, các tác vụ khác với server mới diễn ra được, vì vậy cần kiểm tra kết nối trước mỗi lần sử dụng tài nguyên hay những tác vụ có liên quan đến Siot server.

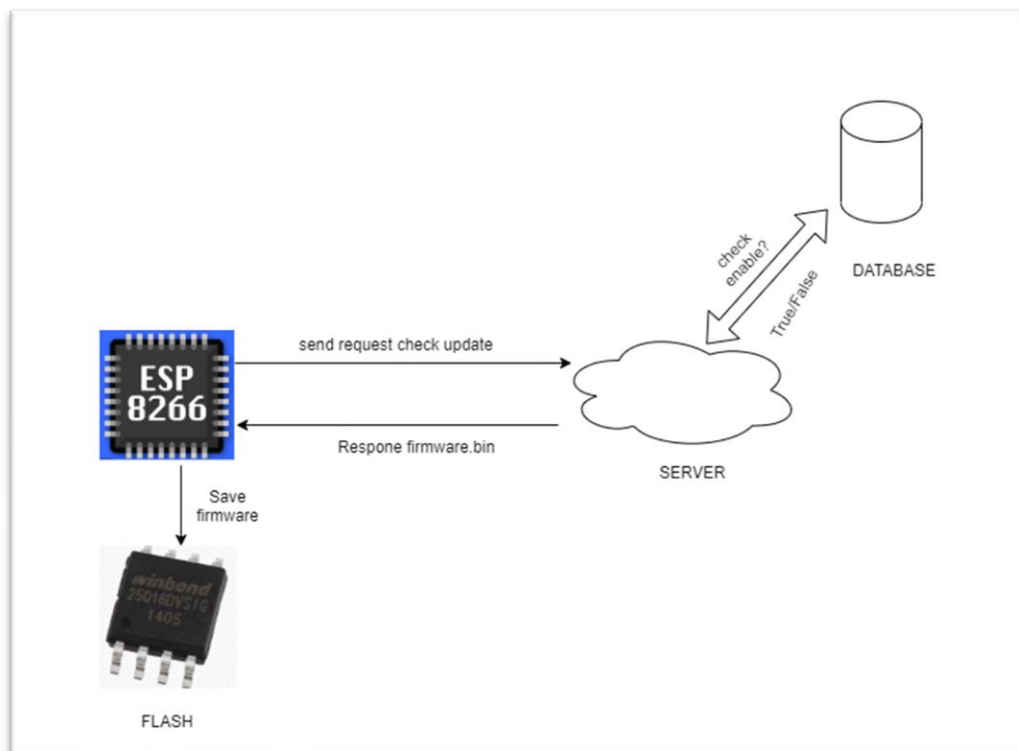
2.6 Cập nhật phần mềm từ xa

Cập nhật firmware từ xa là bài toán bắt buộc và cần thiết với mọi thiết bị điện tử. Nó cho phép nâng cấp, sửa lỗi cho thiết bị gặp phải trong quá trình triển khai lắp đặt ra thực tế, những sai sót đó có thể không gặp phải trong quá trình phát triển.

Ngoài ra, với việc cập nhật firmware mới, đem lại trải nghiệm mới cho người sử dụng, đổi mới thiết bị, khiến cho nó tương thích với những sự phát triển công nghệ quanh nó, và thiết bị không bị lạc hậu.

Đối với Siot core, VDK ESP8266-12E hỗ trợ việc cập nhật từ xa OTA (Over the air). Cập nhật firmware OTA là tiến trình tải firmware mới vào ESP module thay vì sử dụng cổng Serial. Tính năng này thực sự rất hữu dụng trong nhiều trường hợp giới hạn về kết nối vật lý đến ESP Module. Trong bài toán này, Siot core sử dụng cập nhật OTA bằng phương pháp http server.

Mô hình bài toán:



Hình 2.6.1 Mô hình cập nhật firmware OTA cho thiết bị

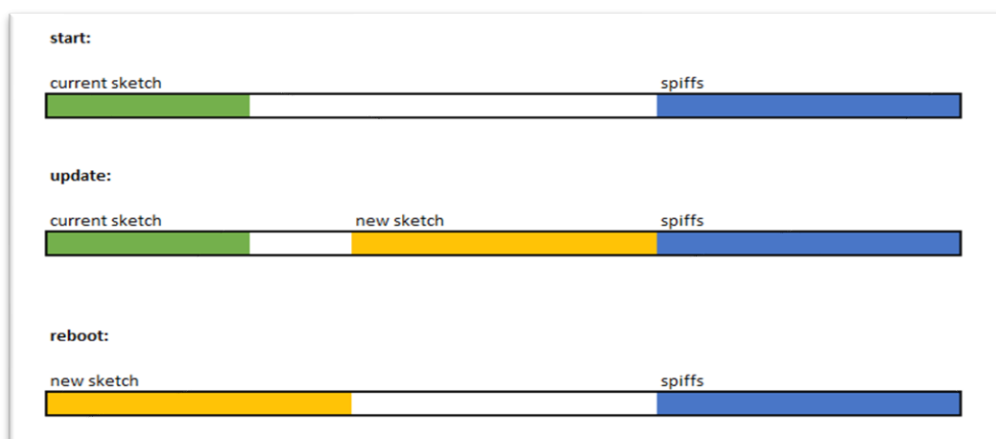
Theo chu kì đã được cài đặt, trung bình 24h VDK gửi một request đến server kiểm tra xem thiết bị có cần update phần mềm mới hay không. Việc thiết bị có được update hay không phụ thuộc vào nhiều yếu tố, vì vậy cùng một thiết bị có thể có cái được cần cập nhật, có cái không cần. Để quản lí trong CSDL luôn có cơ

“ENABLE” để báo thiết bị nào được cập nhật. Việc kiểm tra được thực hiện khi có tín hiệu request từ thiết bị.

Trong trường hợp thiết bị được phép cập nhật, server sẽ trả về file firmware trong response. VDK nhận được firmware sẽ lưu lại trong bộ nhớ Flash. Việc update cần một số điều kiện về bộ nhớ Flash có đủ để chứa firmware mới hay không.

Khi cập nhật thành công, thông tin về firmware mới, cũng như trạng thái cũng được cập nhật trên CSDL. Nếu có lỗi xảy ra, có thể do đường truyền, server hoặc bộ nhớ thì đều được ghi log để người sử dụng phát hiện và kiểm tra.

Bản cập nhật được ghi vào bộ nhớ Flash như hình. Trong lần khởi động tiếp theo, VDK sẽ chép firmware mới đè lên firmware cũ. Sketch mới được khởi động.



Hình 2.6.2 Cách ghi lưu firmware trên bộ nhớ Flash

2.7 Thiết kế giao diện và đóng gói SIOTCore

Bộ thư viện được thiết kế gồm các class như sau:

- Board: chứa các thông tin liên quan đến Board mạch, các thông tin liên quan đến VDK, ID, version firmware, tình trạng kích hoạt,...
- Common: chứa các hàm chung, thông tin chung mà các class khác đều sử dụng như: write/read eeprom,...
- ConnectInternet: cung cấp các hàm, các thuộc tính phục vụ việc kết nối vào mạng Wifi.
- ConnectServer: các hàm để truyền nhận, thông tin từ một server thông qua các đường dẫn URL, các API mà phía server đang quy định.
- Debug: cung cấp các phương thức để hiển thị, log thuận tiện cho phát triển, có hai cách log là: đường UART qua serial, hoặc hiển thị trên Oled
- Sensor: class chung cho việc kết nối và đọc và lưu trữ giá trị từ các cảm biến.

- SiotCore: là class triệu gọi tất cả các class trên, để đóng gói, khi sử dụng chỉ cần khởi tạo một đối tượng SiotCore là hoàn toàn có thể sử dụng Platform với các chức năng.

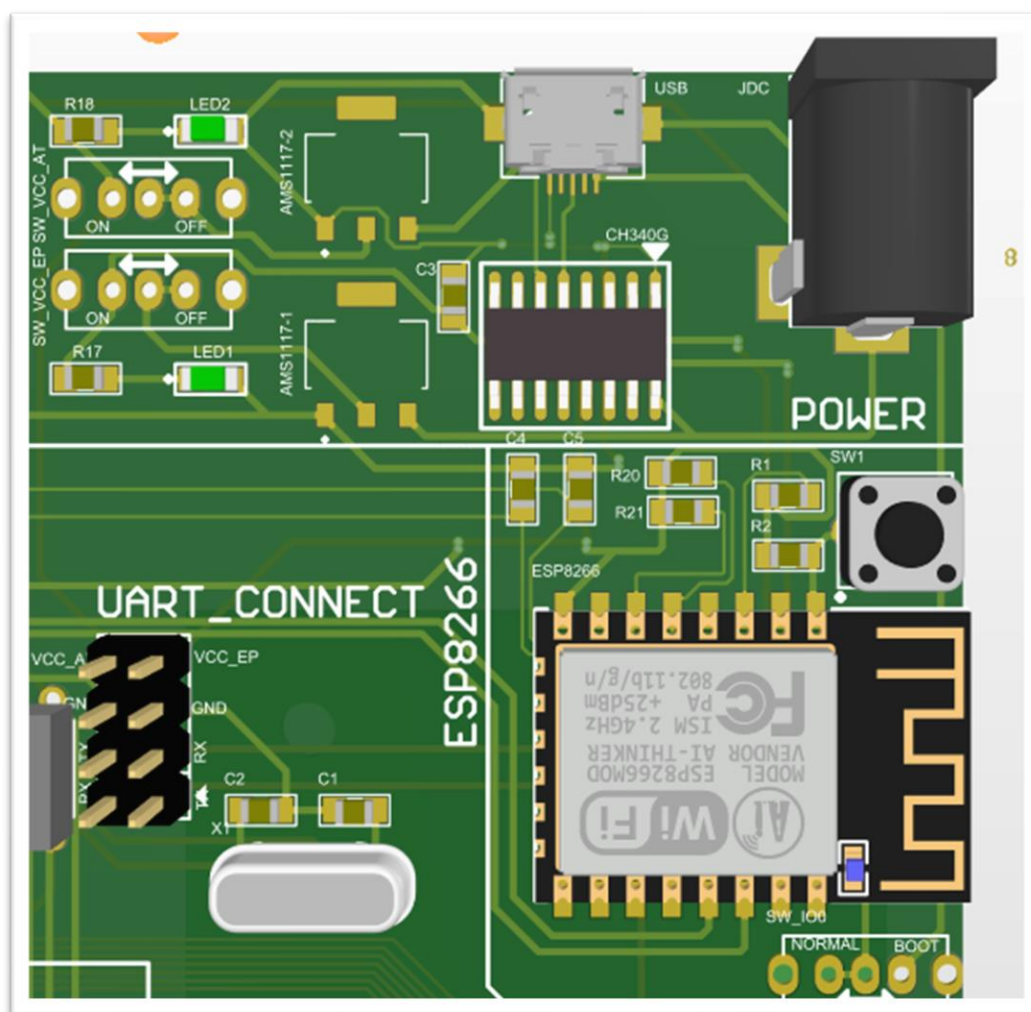
Các sử dụng từng class và chức năng của từng hàm và thuộc tính đều được giải thích và tổng hợp thành document để dễ dàng cho người sử dụng.

Chương trình mẫu “Hello World” sẽ được trình bày trong phần sau, sẽ giải thích cách tích hợp SiotCore vào chương trình các nhân, các yêu cầu tối thiểu để có thể sử dụng.

2.8 Thiết kế kit ứng dụng nền tảng SIOTPlatform

Dựa trên các phân tích ở trên, kit lập trình ứng dụng được xây dựng trên nền tảng SIOTPlatform. Với VDK chính là ES8266-12e, các loại cảm biến hỗ trợ chính là cảm biến nhiệt độ, độ ẩm, cảm biến khoảng cách, cảm biến bụi. Ngoài ra Kit có thêm VDK Atmega 328p (chip dùng cho các board Arduino) có thể thực hành và phát triển giống như trên các mạch Arduino, tiện lợi cho phát triển và học tập.

Trên kit còn có sẵn các chân socket, GPIO người dùng hoàn toàn có thể kết nối, cắm thêm nhiều loại cảm biến khác để phục vụ cho việc phát triển ứng dụng



Hình 2.8.1 Khối nguồn và ESP8266

Ở trên, gồm khối nguồn và khối ESP8266, điện áp đầu vào là 5V, trên mạch có 2 ic nguồn ASM1117 có tác dụng giảm áp từ 5v xuống 3,3v – mức điện áp thích hợp ESP8266 hoạt động. Với việc dùng 2 ic nguồn có tác dụng chia tải, ESP dùng một nguồn riêng, các khối cảm biến sẽ dùng nguồn riêng.

Trên board có sử dụng ic Ch340c chuyển từ USB to TTL, tiện lợi cho người lập trình, chỉ cần cắm cap Micro USB là có thể giao tiếp giữa bo mạch và máy tính.

Ngoài ra board có khối connect theo chuẩn UART, cho phép ESP có thể giao tiếp với các ngoại vi khác một cách dễ dàng.

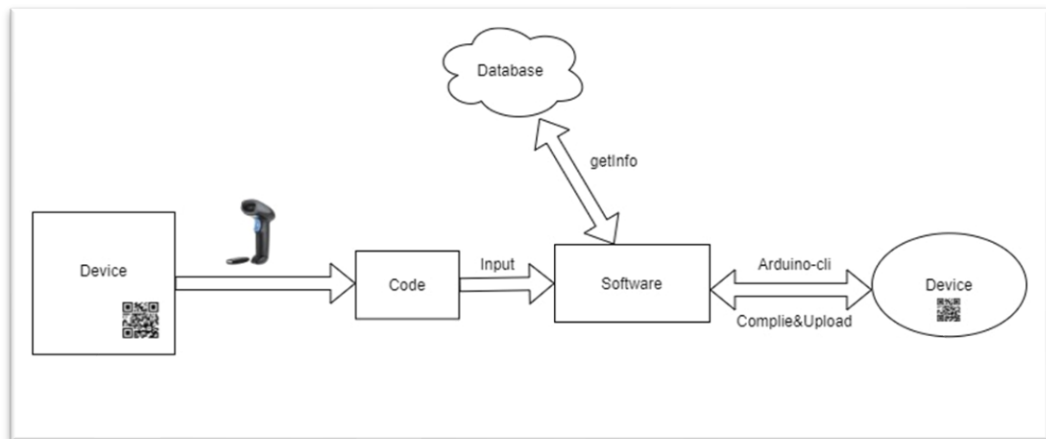
2.9 Công cụ nạp code tự động

Hiện nay, với số lượng sản phẩm quy mô công nghiệp việc nạp firmware cho số lượng hàng loạt luôn gây ra rất nhiều khó khăn. Thực tế trong các công ty, công việc này thường cần phải mất từ 2-3 người với nhiều công đoạn, gây mất

nhieu thời gian. Ngoài ra, quá trình trên còn rất dễ để xảy ra những sai sót từ phía người thực hiện.

Từ những bất cập ở trên, dưới đây là quy trình chung cho việc nạp firmware lần đầu cho những thiết bị được sản xuất được đề xuất và đã được áp dụng ở công ty, hoàn toàn có thể áp dụng cho Siot core trong tương lai.

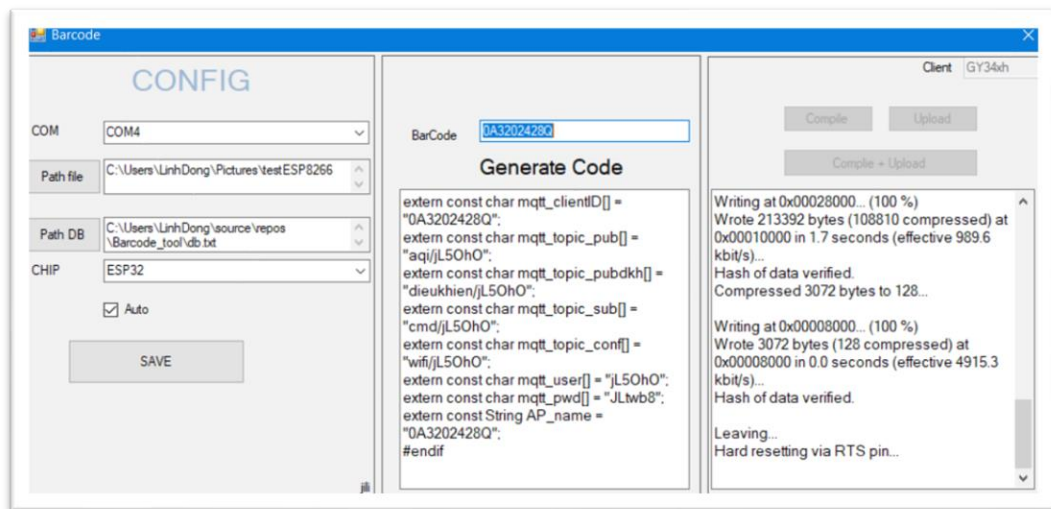
Với mỗi thiết bị được sản xuất, đều được đặt gắn với một mã QR code, chứa thông tin mã định danh của sản phẩm. Từ mã định danh ở trên, mà có thể tra cứu tất cả thông tin liên quan đến sản phẩm.



Hình 2.9.1 Mô hình hoạt động nạp code

Sử dụng máy đọc barcode, từ đó lấy được mã sản phẩm, máy barcode được kết nối với máy tính qua giao thức USB, từ đó sẽ lấy được mã định danh thiết bị.

Xây dựng phần mềm với input là mã code vừa nhận, sử dụng các hàm API để lấy thông tin từ CSDL của công ty. Tất cả những thông tin trên sẽ được lưu trên một file Device_info.h, để chương trình firmware có thể sử dụng. Trong phần mềm ở trên, đã tích hợp Arduino-cli, cho phép sử dụng các câu lệnh để complie và Upload firmware cho thiết bị, giống như việc sử dụng Arduino IDE.



Hình 2.9.2 Phần mềm nạp code

Tất cả các bước ở trên đã được tự động hóa, việc của người nạp code chỉ cần cầm máy barcode quét mã QR, vì thế sẽ giảm được tối đa sai sót về phía con người.

Chương 3 PHÁT TRIỂN MÃ NGUỒN SIOT CORE

3.1 Cài đặt Siot Core cho thiết bị nhúng.

Siot Core cung cấp tất cả các tính năng đã nêu ở trên, hiện tại hỗ trợ các dòng loại VDK ESP8266 series, dễ dàng tích hợp và sử dụng. Dưới đây là cách sử dụng.

Tải bộ SDK ở link sau: https://github.com/LinhIThust/siotcore_sdk

Giải nén file hiện tại, bộ SDK gồm 1 folder src và 1 file Main.ino là file chương trình template. Để sử dụng, người dùng chỉ cần sửa lại code trong file Main.ino theo đúng yêu cầu và mục đích sử dụng.

Cấu trúc file Main.ino

```
//TODO: include siotcore sdk and declare a object to use
#include "src/SiotCore.h"
SiotCore core;
//TODO: END

#define PIN_IN 14
void setup()
{
    Serial.begin(115200);
    pinMode(LED_BUILTIN, OUTPUT);
    //TODO: init siot core
    core.init();
    //FIXME:
    core.updateFirmware("1.0");
    pinMode(PIN_IN, INPUT);
}
void loop()
{
}
```

Hình 3.1.1 Cấu trúc mẫu khi dùng Siot core

Chương trình gồm 2 phần là “**setup**” và “**loop**”, “**setup**” là phần chạy một lần duy nhất khi VDK hoạt động, “**loop**” là phần code sẽ được chạy lặp lại cho đến khi VDK không hoạt động.

Trong file Main.ino có những từ khóa như “**TODO**”, “**FIXME**” với những ý nghĩa khác nhau. Với “**TODO**” là những câu lệnh bắt buộc cần phải khai báo và sử dụng, người dùng bắt buộc phải có để có thể sử dụng Siot core. Với “**FIXME**”, người dùng có thể sử dụng như khai báo mẫu, hoặc có thể sửa đổi để phù hợp với mục đích sử dụng, tuy nhiên cần tuân theo định dạng như trong SDK để chương trình có thể hoạt động đúng và tối ưu nhất.

Ngoài cách trên, người dùng có thể tách riêng biệt phần SDK ra bằng cách đưa thư mục /src vào thư mục Documents/Arduino/libraries (trên Windows), trong file Main.ino cần include “SiotCore.h” là có thể sử dụng như bình thường.

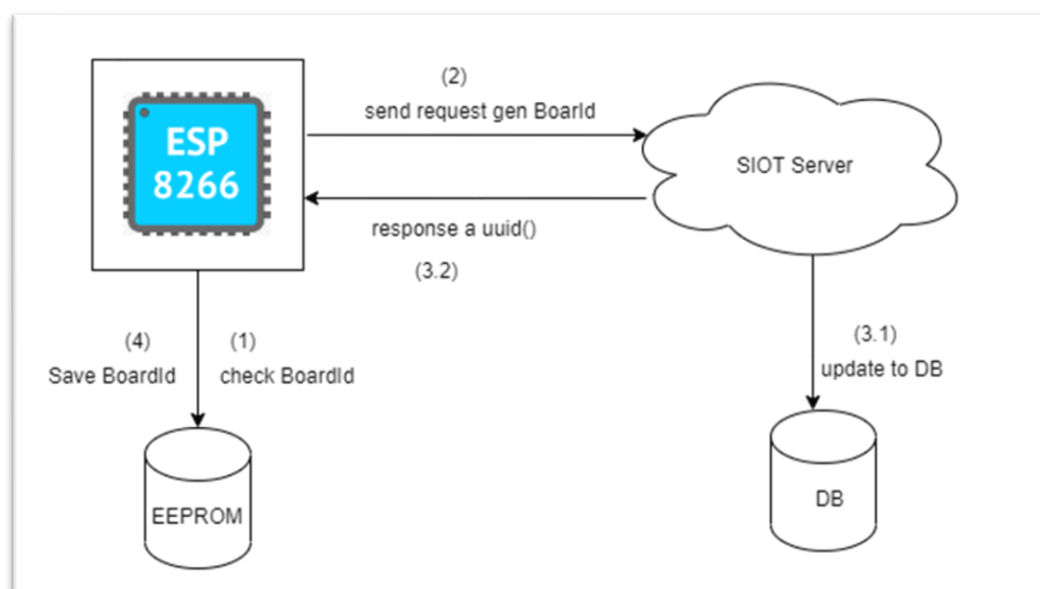
3.2 Khai báo, định danh thiết bị

Với mỗi thiết bị nhúng cần có một mã định danh để phân biệt và quản lí, định danh một lớp với các trường thông tin như sau

Bảng 3.2-1 Các trường thông tin của thiết bị

Trường	Kiểu dữ liệu	Ý nghĩa
chipId	Int	Mã định danh VDK chính trên thiết bị
boardId	String	Mã định danh thiết bị
currentFirmware	String	Phiên bản firmware hiện tại thiết bị
isActive	Bool	Kiểm tra thiết bị đã được kích hoạt hay không, phục vụ bài toán bảo hành thiết bị

Trong đó, “boardId” đóng vai trò là khóa chính. Cơ chế sinh mã định danh thiết bị như sau:



Hình 3.2.1 Mô hình tạo mã định danh cho thiết bị

Khi cần kiểm tra thông tin về mã thiết bị, VDK sẽ đọc từ ô nhớ trong EEPROM đã được quy định về địa chỉ. Nếu thông tin đọc được là trống hoặc không đúng định danh như mong muốn. VDK sẽ coi thiết bị chưa được định danh, VDK gửi một request đến SIOT Platform để “xin” một mã định danh. Khi nhận được yêu cầu, Server sẽ xử lí, bằng cơ chế sinh mã đã được định sẵn, mã thiết bị được tạo ra sẽ được cập nhật trong cơ sở dữ liệu, đồng thời gửi lại mã đó cho VDK. Sau khi nhận được mã định danh, VDK sẽ lưu lại thông tin trong EEPROM phục vụ cho các tiện ích mà SIOT Server cung cấp.

Code mẫu:

```
String ConnectServer::getBoardId(){
    String boardID = "";
    int status = client.get(URI_GET_BOARD_ID, &boardID);
    if(status == STATUS_CODE_OKE){
        return boardID;
    }
    return "";
}
```

Hình 3.2.2 Hàm lấy mã định danh cho thiết bị

3.3 Kết nối Siot Server

Siot server được sử dụng Nodejs tạo nên Web Server. Cấu trúc chung của 1 request sẽ gồm các thông tin như sau

Bảng 3.3-1 Các trường thông tin của 1 request

Trường	Kiểu dữ liệu	Ý nghĩa
baseUrl	String	Đường dẫn chung của tất cả API
Uri	String	Đường dẫn riêng của từng API
Method	String	Kiểu request: POST, GET, UPDATE, DELETE, CREATE
authentication	String	Key để xác thực bảo mật
Content-type	String	Kiểu dữ liệu trong request trong SiotServer là application/json
Value	String	Dạng json. Dữ liệu raw đẩy lên server

Với mỗi phiên làm việc, đầu VDK sẽ gửi 1 request để kiểm tra đã kết nối được với SIOT server không? Nếu thành công việc gửi dữ liệu và các thao tác khác sẽ được diễn ra. Với mỗi request thành công, phía client là VDK sẽ nhận được mã HTTP_CODE là 200 và giá trị cần. Ví dụ dưới đây chỉ ra hàm kết nối với Siot core, với mục đích gửi 1 giá trị random lên server. Với các giá trị đã được

#define trước theo định nghĩa ở bảng trên. Khi sử dụng, chỉ cần define lại các giá trị trên theo giá trị riêng của từng thiết bị.

```
#define BASE_URL= "soict-core-01.herokuapp.com";
#define URI_CHECK_COMMUNICATION= "/communication";
#define URI_GET_VERSION= "/version";
#define URI_GET_BOARD_ID= "/boardId";
#define URI_GET_FIRMWARE= "/updateFirmware" ;
#define URI_POST_VALUE = "/sensor"
```

Hình 3.3.1 Define các giá trị đường dẫn API

Người dùng có thể định nghĩa lại BASE_URL và các đường dẫn API trong file Define_Info.h tùy thuộc vào API mà người dùng sử dụng.

```
// configure traged server and url
http.begin(client, URL); //HTTP
http.addHeader("Content-Type", "application/json");
http.addHeader("Authorization", KEY);

// start connection and send HTTP header and body
int httpCode = http.POST("{\"value\":\""+String(random())+"\"}");

// httpCode will be negative on error
if (httpCode > 0) {
    // HTTP header has been send and Server response header has been
    handled
    // file found at server
    if (httpCode == HTTP_CODE_OK) {
        const String& payload = http.getString();    }
    } else {
        //Log fail detail
    }

    http.end();
```

Bảng 3.3-2 Giá trị mẫu 1 request

Trường	Giá trị
URL	"http://siot.techlinkvn.com/api/devices/test-v6s6s/attributes/temperature-jdhv1"
Content-Type	"application/json"
Authorization	"Bear ereyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI1YWU5MmZhNC01NGI4LTRkNDMtODE5My1jYjg3YTBiNzg0ZmMiLCJpYXQiOiJlE2MDgxMzMwNDV9.pWXUXsHNcClcJo42ZhEyE75kyq_VKVuaES9Ft8lWmvg"
Method	POST
Value	"{\"value\":\""+String(random())+"\"}"

Với bài toán cập nhật firmware, cũng giống như những tham số ở trên, người dùng chỉ cần thêm version firmware hiện tại, thông số này có thể lấy được từ EEPROM đã được giới thiệu ở trên

```
int ConnectServer::updateFirmware(String version){
    ESPhttpUpdate.onEnd(update_finished);
    ESPhttpUpdate.onProgress(update_progress);
    ESPhttpUpdate.onError(update_error);
    Debug::LOG_TO_SCREEN(0,0,"Start upload firmware.....");
    t_httpUpdate_return ret = ESPhttpUpdate.update(URI_GET_FIRMWARE, version);
    switch (ret) {
        case HTTP_UPDATE_FAILED:
            Debug::LOG_TO_SCREEN(0,0,"HTTP_UPDATE_FAILED Error");
            break;
        case HTTP_UPDATE_NO_UPDATES:
            Debug::LOG_TO_SCREEN(0,0,"HTTP_UPDATE_NO_UPDATES");
            break;
        case HTTP_UPDATE_OK:
            Debug::LOG_TO_SCREEN(0,0,"HTTP_UPDATE_OK");
            break;
    }
}
```

Hình 3.3.2 Hàm cập nhật firmware

Trong hàm trên có sử dụng các hàm callback:

- onEnd- được gọi khi kết thúc quá trình cập nhật. Nếu thành công người dùng sẽ được báo hoàn thành đồng thời cập nhật lại các giá trị lên database, còn lại sẽ chỉ rõ lỗi gặp phải.
- onProgress() trong quá trình cập nhật. Trong phần này log ra tiến trình dưới dạng % tiến độ, đã tải được bao nhiêu trong toàn bộ file.
- onError() được gọi khi có lỗi xảy ra trong quá trình cập nhật.

3.4 Đọc giá trị cảm biến, ngoại vi

Trong bộ SDK của Siot core xây dựng một class dành chung cho tất cả các loại cảm biến có cấu trúc như sau.

```
class Sensor {  
private:  
public:  
    __sensor __sensor;  
    virtual void setup()=0;  
    virtual int read()=0;  
    virtual void calibrate() = 0;  
    virtual void read_calib() = 0;  
    float read_value(int _add);  
    int write_value(float _value, int _add);  
};
```

Hình 3.4.1 Class sensor chung

Trong đó có định nghĩa một struct __sensor được khai báo như sau

struct __sensor {
int sensor_status;
float *sensor_value;
}

Các hàm đều dùng từ khóa “virtual” để cho người dùng có thể kế thừa một cách dễ dàng, tùy theo mục đích sử dụng có thể viết lại các hàm trên.

Ngoài ra, cần định nghĩa các chân của sensor đã được kết nối vào những GPIO nào của VDK.

Siot core cung cấp các hàm cho việc calib dữ liệu, vì chất lượng các cảm biến là khác nhau, phụ thuộc nhiều vào các yếu tố ngoại cảnh dễ đến sai số là điều khó tránh, vì vậy để có thể có kết quả tốt nhất, cần có sự can thiệp và điều chỉnh cho hợp lí.

3.5 Công cụ debug

Debug là việc không thể thiếu khi phát triển một ứng dụng, giúp lập trình viên có thể tìm và sửa những lỗi trong quá trình phát triển. Trong Siot core có xây dựng một class để Debug. Có hai hình thức Debug chính được dùng: Log ra Serial (cổng COM) hoặc ra màn hình (sử dụng modul Oled 1306) cũng là hai phương thức hay dùng.

Dưới đây là ví dụ về hàm log một chuỗi bất kì ra màn hình

```

void Debug::LOG_TO_SCREEN(int x,int y,String string){
    display.init();
    display.displayOn();
    if(x >128) x=0;
    if(y>64) y=0;
    display.clear();
    display.setTextAlignment(TEXT_ALIGN_LEFT);
    display.drawStringMaxWidth(x,y,MAX_WIDTH,string);
    display.display();
    delay(300);
}

```

Hình 3.5.1 Hàm log chuỗi ra màn hình Oled

Để sử dụng, người dùng cần gọi hàm và truyền tham số gồm:

- Int x: giá trị vị trí điểm ảnh bắt đầu theo trục x
- Int y: giá trị vị trí điểm ảnh bắt đầu theo trục y
- String string: giá trị chuỗi cần hiển thị

Ngoài ra, người dùng có thể chỉnh lại font chữ hoặc cỡ chữ theo tùy mục đích sử dụng.

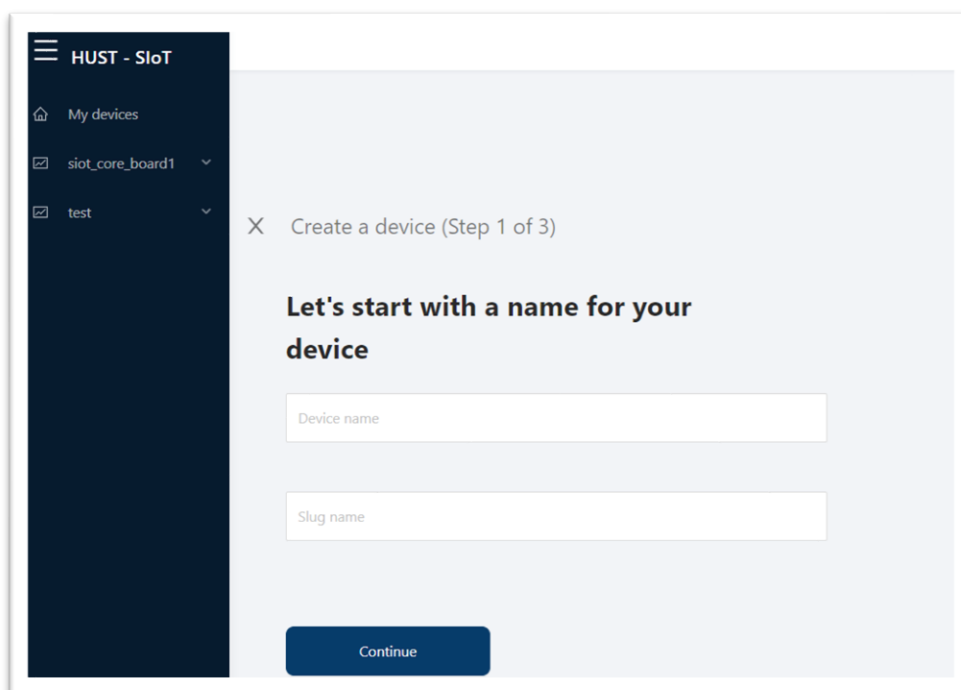


Hình 3.5.2 Hiển thị màn hình debug khi cập nhật firmware

Chương 4 KẾT NỐI, CẬP NHẬT DỮ LIỆU SIOT SERVER

4.1 Tạo thiết bị mới trên Siot server

Để có thể đẩy dữ liệu cập nhật lên server trước hết cần tạo một thiết bị mới trên cơ sở dữ liệu. Để có thể thêm thiết bị cần truy cập vào đường dẫn: <http://iot.techlinkvn.com:90/> cần tạo tài khoản nếu chưa có, chọn vào mục “Add device”

The screenshot shows a web application interface for 'HUST - SIoT'. On the left is a dark sidebar with a menu containing 'My devices', 'siot_core_board1', and 'test'. The main content area is titled 'Create a device (Step 1 of 3)'. It features the heading 'Let's start with a name for your device' and two input fields: 'Device name' and 'Slug name'. A blue 'Continue' button is positioned at the bottom right of the form.

Hình 4.1.1 Giao diện tạo thiết bị mới trên siot core

Điền các thông tin cần thiết, sau đó tạo các trường thông tin cho các dữ liệu cảm biến cần thiết của thiết bị.

Hình 4.1.2 Các thông tin cảm biến của thiết bị

Theo ví dụ ở trên, thiết bị trên gồm 2 giá trị gồm:

- Giá trị về nhiệt độ, kiểu số thực, đơn vị là độ C.
- Giá trị về độ ẩm, kiểu số nguyên, đơn vị là %.

Sau khi hoàn thành các bước, các thông tin về API, authorization được hiển thị.

Hình 4.1.3 Thông tin về API của thiết bị

Vì vậy chỉ cần sửa lại #define theo như bảng 3.3-2 như ở trên.

4.2 Cập nhật dữ liệu

Sau khi đã tạo được thiết bị mới ở trên. Người dùng đã được cung cấp thông tin cần thiết có thể giao tiếp với Siot server. Trong siot core đã xây dựng sẵn hàm để có thể đẩy dữ liệu

```
HttpClient http;  
http.begin(client, URL);  
http.addHeader("Content-Type", "application/json");  
http.addHeader("Authorization", KEY);  
int httpCode = http.POST("{\"value\":\""+String(getValue())+"\"}");
```

Hình 4.2.1 Các thông tin trong request tới Siot server

Trong đó nội dung trong http.POST() là một giá trị định dạng json. Ở ví dụ ở trên, giá trị ở trong value tùy thuộc vào giá trị mà người dùng muốn truyền: có thể là nhiệt độ, độ ẩm, bụi, version, ... tùy thuộc vào URL- giá trị trường mà người dùng muốn cập nhật lên server.

Ngoài ra, có thể cấu hình tần suất cập nhật dữ liệu, trong siot core mặc định là 5p/ 1 lần gửi dữ liệu.

Chương 5 KẾT QUẢ ĐẠT ĐƯỢC

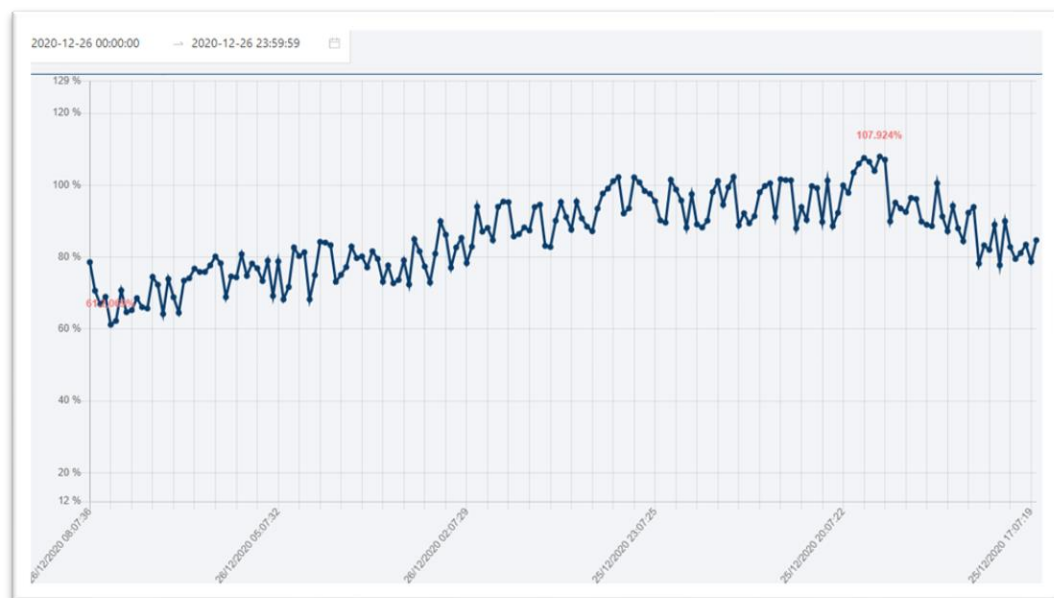
5.1 Cập nhật firmware thành công



Hình 5.1.1 Hiện thị quá trình cập nhật firmware

5.2 Cập nhật dữ liệu lên SIOT Platform

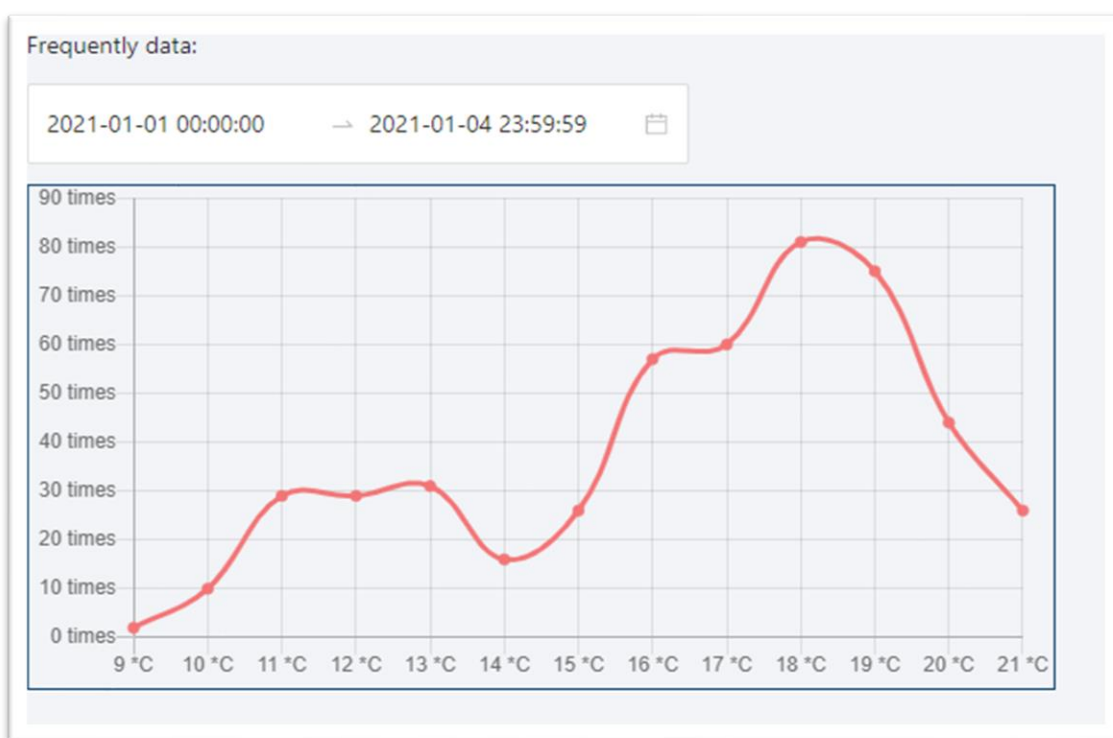
Kết nối thành công với Siot server. Dữ liệu về nhiệt độ, độ ẩm sử dụng cảm biến DHT22, được VDK đọc và update lên Siot Server, với tần suất 5 phút/lần.



Hình 5.2.1 Đồ thị dữ liệu độ ẩm theo thời gian

Name: Humidity Created at: 19/12/2020 14:14:21 Max value: 107.924 % Min value: 31.0135 % Last active: 26/12/2020 08:07:36		
2020-12-26 00:00:00 → 2020-12-26 23:59:59		
Total: 182 in 1815 total records		
	Value (%)	Created date
1	78.5939	26/12/2020 08:07:36
2	70.6499	26/12/2020 08:02:36
3	66.8663	26/12/2020 07:57:35
4	66.9573	26/12/2020 07:52:35

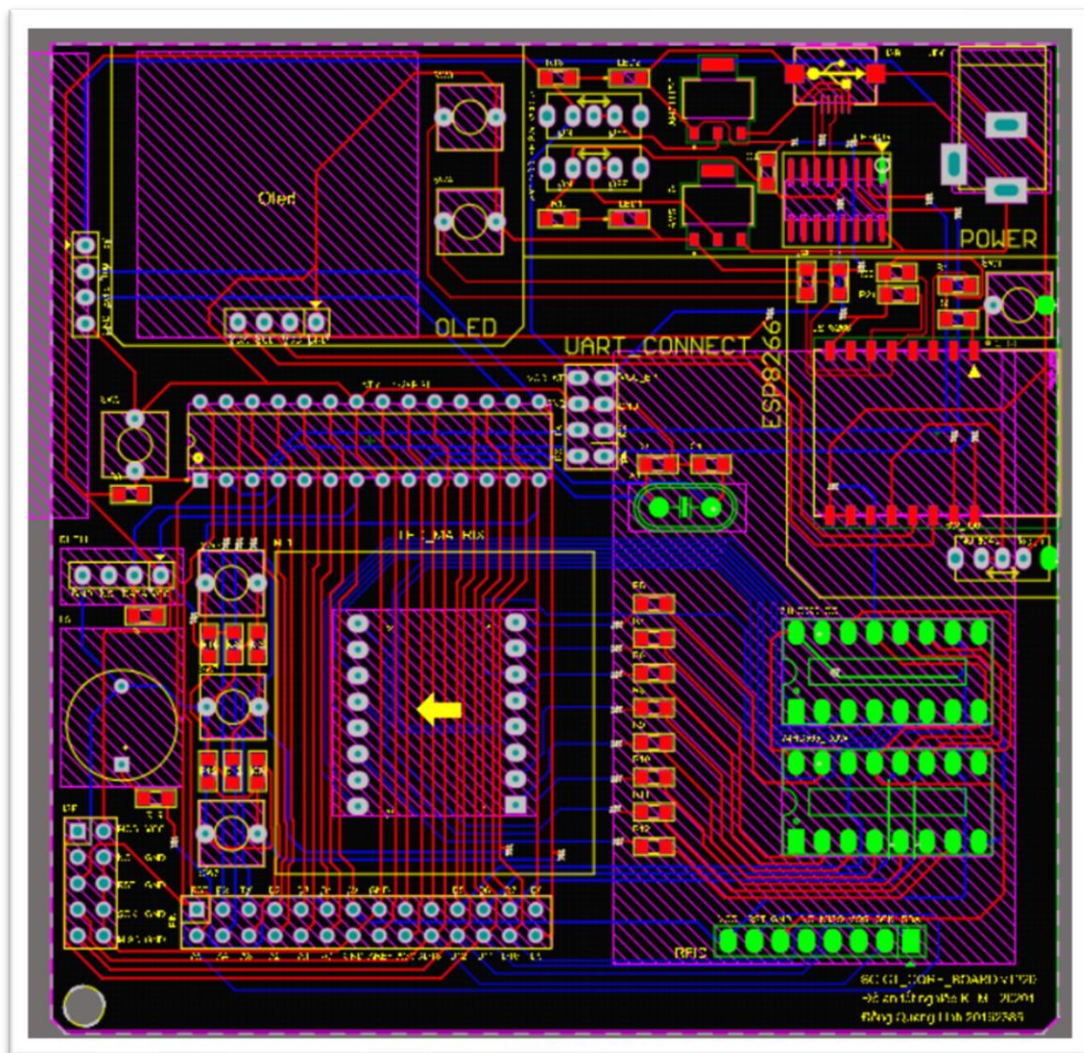
Hình 5.2.2 Bảng dữ liệu theo ngày



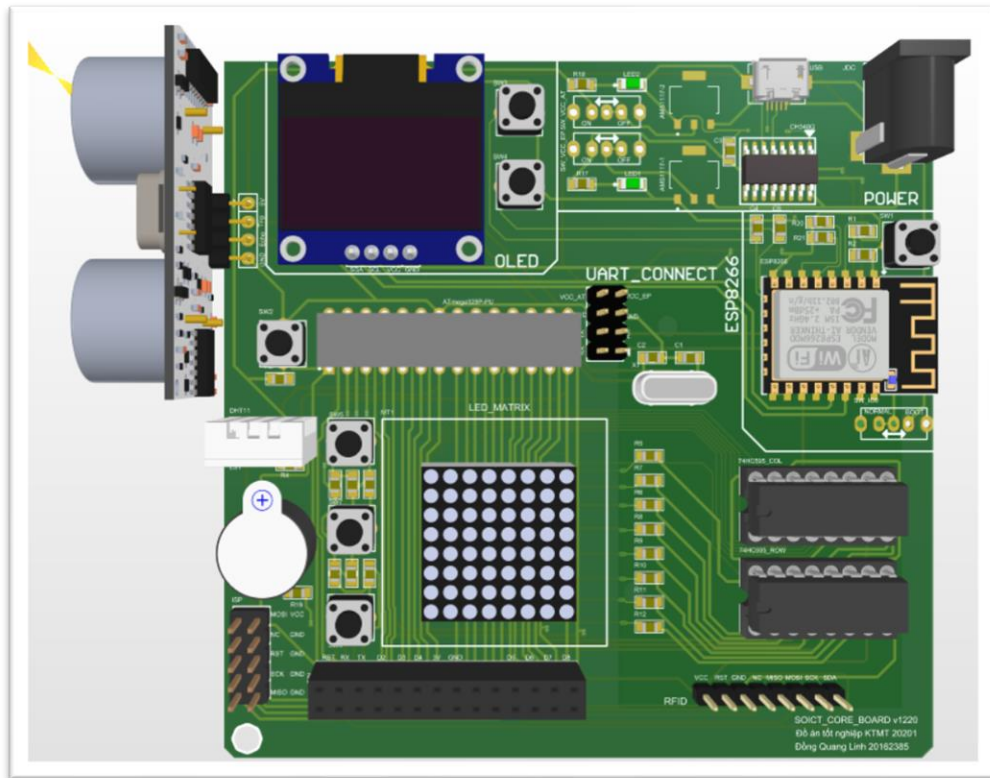
Hình 5.2.3 Đồ thị phân bố các giá trị về nhiệt độ

5.3 Thiết kế phần cứng

Phần cứng board được thiết kế sử dụng phần mềm Altium, với kit lập trình trên, có thể sử dụng và thực hành với hầu hết các bài học về lập trình nhúng và ứng dụng phổ biến hiện nay



Hình 5.3.1 Sơ đồ schematic Siot board



Hình 5.3.2 Mô phỏng 3D Siot Board

5.4 Định hướng phát triển đề tài

Khi nhận đề tài, tôi muốn đề tài sẽ được áp dụng vào thực tế, giúp mọi đối tượng yêu thích về lập trình nhúng có thể tiếp cận và sử dụng. Trước hết, đề tài sẽ được áp dụng tại viện CNTT&TT của trường, có thể áp dụng vào các tiết thực hành môn học. Với từng môn học thì cần thực hiện các chức năng trong phạm vi, từ đó có thể tạo thành một series ứng dụng, từ đơn giản đến nâng cao.

Trong tương lai, tôi muốn hướng đến đối tượng là các em học sinh cấp 2, cấp 3. Có thể dùng Siot core như một công cụ, để lập trình, áp dụng những bài toán, thuật toán khô khan thành những hiệu ứng, sự thể hiện trên phần cứng bằng màn hình, hay đèn led, tạo hứng thú hơn cho các em học tập.

Ngoài ra, trong các phiên bản tiếp theo của Siot core sẽ hỗ trợ nhiều dòng VDK hơn, hỗ trợ nhiều chuẩn truyền thông hơn như Zingbee, 4G, BLE, ...

5.5 Kết luận đề tài

Đề tài đã được hoàn thành và giải quyết được những vấn đề mà thầy giáo hướng dẫn đã giao, ưu điểm:

- Thực hiện giao tiếp, cập nhật dữ liệu thành công với Siot Server. Chức năng cập nhật firmware từ xa được thực hiện và hoàn thành.

- Chuẩn hóa code, dễ dàng cho người sử dụng.
- Có thể extend rộng hơn, bằng cách kết thừa từ những class có sẵn, người dùng có thể tự tạo các custom function từ code hiện tại.
- Kit lập trình được thiết kế riêng, phù hợp với các chức năng hiện tại.

Bên cạnh những nhiệm vụ đã được hoàn thành, đề tài còn một số vấn đề tồn tại thiếu sót:

- Hiện chỉ hỗ trợ các dòng VDK ESP8266 như 12E, 12F, 07 nên các loại VDK khác đều không sử dụng được
- Hỗ trợ Wifi ở băng tần 2,4Ghz chưa hỗ trợ 5Ghz, ngoài ra không hỗ trợ 4G vì vậy khi lắp đặt ở những nơi không có Wifi thì không thể thu thập được dữ liệu.
- Các vấn đề về pin và năng lượng chưa được tính toán và tối ưu trong phiên bản này.

TÀI LIỆU THAM KHẢO

- ESP-IDF Programming Guide:
<https://docs.espressif.com/projects/esp-idf/en/latest/esp32/index.html#>
- Node.js v15.5.0 Documentation:
<https://nodejs.org/dist/latest-v15.x/docs/api/>
- Các chuẩn giao tiếp truyền thông: <http://dammedientu.vn/>
- DHT22 datasheet:
<https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf>
- SDS011 Manual:
<https://cdn-reichelt.de/documents/datenblatt/X200/SDS011-DATASHEET.pdf>