Contents

[CHƯƠNG 1: GIỚI THIỆU 5](#_Toc104151399)

[1.1. Tổng quan về đề tài: 5](#_Toc104151400)

[1.2. Lý do chọn đề tài: 5](#_Toc104151401)

[1.3. Nội dung nghiên cứu: 5](#_Toc104151402)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 6](#_Toc104151403)

[2.1. IOT Gateway: 6](#_Toc104151404)

[2.1.2. Tổng quan về IOI Gateway: 6](#_Toc104151405)

[2.1.3. Các chức năng của IOT Gateway: 6](#_Toc104151406)

[2.1.3. Lợi ích khi sử dụng IOT Gateway: 7](#_Toc104151407)

[2.2. Giới thiệu phần cứng: 8](#_Toc104151408)

[**2.2.1.** **Vi điều khiển STM32F103C8T6:** 8](#_Toc104151409)

[2.2.2. Module Wifi ESP8266 NodeMCU: 12](#_Toc104151410)

[**2.2.3.** **Module Bluetooth HC05:** 15](#_Toc104151411)

[**2.2.4.** **Module Sub-GHz CC1101:** 19](#_Toc104151412)

[2.3. Giới thiệu phần mềm: 26](#_Toc104151413)

[2.4. Mạng Mesh Wifi: 27](#_Toc104151414)

[2.4.1. Tổng quan về mạng mesh wifi: 27](#_Toc104151415)

[2.4.2. Cấu trúc mạng: 28](#_Toc104151416)

[2.5. Tổng quan về mạng Internet, giao thức HTTP và mô hình MVC: 29](#_Toc104151417)

[2.5.1. Mạng Internet 29](#_Toc104151418)

[2.5.2. Giao thức HTTP 29](#_Toc104151419)

[2.5.3. Ngôn ngữ lập trình PHP: 32](#_Toc104151420)

[2.5.4. Mô hình MCV: 34](#_Toc104151421)

[2.6. Quản lý cơ sở dữ liệu với phpMyAdmin: 36](#_Toc104151422)

[2.6.1. Tổng quan về cơ sở dữ liệu và hệ quản trị cơ sở dữ liệu: 36](#_Toc104151423)

[2.6.2. PHPMyAdmin: 36](#_Toc104151424)

[CHƯƠNG 3: THIẾT KẾ VÀ THỰC HIỆN PHẦN CỨNG 38](#_Toc104151425)

[3.1. Sơ đồ tổng quát hệ thống: 38](#_Toc104151426)

[3.2. Sơ đồ mạch: 39](#_Toc104151427)

[3.3. Thiết kế PCP: 42](#_Toc104151428)

[3.3.1. Thiết kế PCB bằng phần mềm Orcad: 42](#_Toc104151429)

[3.3.2. Kết quả layout: 43](#_Toc104151430)

[CHƯƠNG 4: THIẾT KẾ VÀ THỰC HIỆN PHẦN MỀM 44](#_Toc104151431)

[4.1. Ý tưởng: 44](#_Toc104151432)

[4.1.1. Nhận dữ liệu các thiết bị: 44](#_Toc104151433)

[4.1.2. Kiểm tra kết nối với thiết bị: 45](#_Toc104151434)

[4.1.3. Tín hiệu điều khiểu gửi từ ứng dụng Android: 45](#_Toc104151435)

[4.2. Tạo cơ sở dữ liệu trên PHPMyAdmin: 46](#_Toc104151436)

[4.2.1. Cấu trúc cơ sở dữ liệu: 46](#_Toc104151437)

[4.2.2. Mã php: 47](#_Toc104151438)

[4.3. Lưu đồ giải thuật: 51](#_Toc104151439)

[4.3.1. ESP8266: 51](#_Toc104151440)

[4.3.2. STM32F103C8T6: 55](#_Toc104151441)

[4.4. Cấu hình hệ thống: 57](#_Toc104151442)

[4.4.1. Mạng Mesh Wifi: 57](#_Toc104151443)

[4.4.2. Cấu hình cho module Bluetooth: 58](#_Toc104151444)

[4.5. Ứng dụng IoTGateway: 60](#_Toc104151445)

[4.5.1. Tổng quan: 60](#_Toc104151446)

[4.5.2. Chức năng: 60](#_Toc104151447)

[4.5.3. Giao diện: 60](#_Toc104151448)

[CHƯƠNG 5: KẾT QUẢ THỰC HIỆN 65](#_Toc104151449)

[*Hình 2.1: Mô hình kết nối của IoT Gateway.* 7](#_Toc104398990)

[*Hình 2.2: Cấu trúc của vi xử lý ARM Cortex M3.* 9](#_Toc104398991)

[*Hình 2.3: Kiến trúc của STM32 nhánh Performance và Access.* 10](#_Toc104398992)

[*Hình 2.4: Vi điều khiển STM32F103C8T6* 11](#_Toc104398993)

[*Hình 2.5: Sơ đồ chức năng các chân KIT STM32F103C8T6* 12](#_Toc104398994)

[*Hình 2.6: Sơ đồ kết nối mạng Wifi.* 13](#_Toc104398995)

[*Hình 2.7: NodeMCU ESP8266.* 14](#_Toc104398996)

[*Hình 2.8: Tổng quan về giao thức Bluetooth.* 15](#_Toc104398997)

[*Hình 2.9: Module Bluetooth HC05.* 16](#_Toc104398998)

[*Hình 2.10: Mô hình hoạt động của php.* 32](#_Toc104398999)

[*Hình 2.11: Sơ đồ tương tác giữa CLIENT, SERVER và DATABASE* 35](#_Toc104399000)

[*Hình 3.1: Sơ đồ tổng quát hệ thống.* 36](#_Toc104399001)

[*Hình 3.2: Sơ đồ mạch.* 37](#_Toc104399002)

[*Hình 3.3: Layout PCB mạch IOT Gateway.* 40](#_Toc104399003)

[*Hình 3.4: Kết quả layout PCB lớp TOP.* 41](#_Toc104399004)

[*Hình 3.5: Kết quả layout PCB lớp BOTTOM.* 41](#_Toc104399005)

[*Hình 4.1: Cấu trúc cơ sở dữ liệu.* 43](#_Toc104399006)

[*Hình 4.2: Cấu trúc bảng trong cơ sở dữ liệu.* 44](#_Toc104399007)

[*Hình 4.3: File PHP trong server.* 44](#_Toc104399008)

[*Hình 4.4: File .JSON trong thư mục WriteDataJson.* 45](#_Toc104399009)

[*Hình 4.5: JSON trả về trong file WIFI1.json* 46](#_Toc104399010)

[*Hình 4.6 : File .JSON trong thư mục control.* 46](#_Toc104399011)

[*Hình 4.7: JSON trả về trong file control.json* 46](#_Toc104399012)

[*Hình 4.8: File .JSON trong thư mục connect.* 47](#_Toc104399013)

[*Hình 4.9: JSON trả về trong file DVconnect.json* 47](#_Toc104399014)

[*Hình 4.10: File PHP trong thư mục AndroidGetdata.* 48](#_Toc104399015)

[*Hình 4.11: Lưu đồ giải thuật tổng quát của ESP8266.* 50](#_Toc104399016)

[*Hình 4.12: Lưu đồ giải thuật xử lý tín hiệu điều khiển.* 51](#_Toc104399017)

[*Hình 4.13: Flowchart chương trình yêu cầu dữ liệu từ các thiết bị Wifi.* 53](#_Toc104399018)

[*Hình 4.14: Lưu đồ giải thuật xử lý dữ liệu từ STM32F103C8T6.* 54](#_Toc104399019)

[*Hình 4.15: Lưu đồ giải thuật tổng quát của STM32F103C8T6.* 55](#_Toc104399020)

[*Hình 4.16: Lưu đồ giải thuật xử lý tín hiệu điều khiển trên STM32F103C8T6.* 56](#_Toc104399021)

[*Hình 4.17: Lưu đồ giải thuật xử lý dữ liệu từ thiết bị trên STM32F103C8T6.* 57](#_Toc104399022)

[*Hình 4.18: Lưu đồ giải thuật yêu cầu dưc liệu từ các thiết bị Sub-GHz.* 58](#_Toc104399023)

[*Hình 4.19: Kết nối mạng mesh các nodeMCU ESP8266.* 59](#_Toc104399024)

[*Hình 4.20: Kết quả cấu hình cho module Bluetooth HC05 – Cổng 1.* 60](#_Toc104399025)

[*Hình 4.21: Giao diện chính - ứng dụng IoTGateway.* 62](#_Toc104399026)

[*Hình 4.22: Giao diện hiển thị dữ liệu - ứng dụng IoTGateway.* 64](#_Toc104399027)

[*Hình 4.23: Giao diện lịch sử.* 65](#_Toc104399028)

# GIỚI THIỆU

1. Tổng quan về đề tài:

Trong đề tài này, ta đi thiết kế một bộ IoT Gateway có thể nhận dữ liệu từ nhiều nguồn thiết bị khác nhau sử dụng một số giao thức như Bluetooth, Wifi, Sub GHz… Người dùng có thể quản lý và điều khiển các thiết bị thông qua Gateway bằng phần mềm điện thoại hoặc website.

1. Lý do chọn đề tài:

Trong những năm gần đây, công nghệ phát triển vượt bậc, nhu cầu được kết nối ở mọi nơi, mọi lĩnh vực của con người cao hơn bao giờ hết. Từ đó khái niệm Internet of Things và những ứng dụng của nó đã được hình thành và ngày càng phát triển phổ biến. Những mô hình IoT được xây dựng ở nhiều nơi như smart home, smart city, smart manufacturing,… đã và đang mang lại nhiều lợi ích cho con người.

Tuy nhiên, với lượng thiết bị trong hệ thống ngày càng nhiều thì lượng thông tin cần truyền tải lên hệ thống cũng tăng lên. Một vấn đề nữa là dữ liệu từ các thiết bị khác nhau sẽ có cấu trúc, kiểu dữ liệu khác nhau. Những vấn đề đó tạo ra gánh nặng cho hệ thống xử lý trung tâm. Vì vậy, ta cần một thiết bị giúp thu thập, lọc dữ liệu thô và đông nhất kiểu dữ liệu trước khi gửi nó lên hệ thống xử lý trung tâm qua Internet.

1. Nội dung nghiên cứu:

Nội dung cần nghiên cứu trong đề tài này:

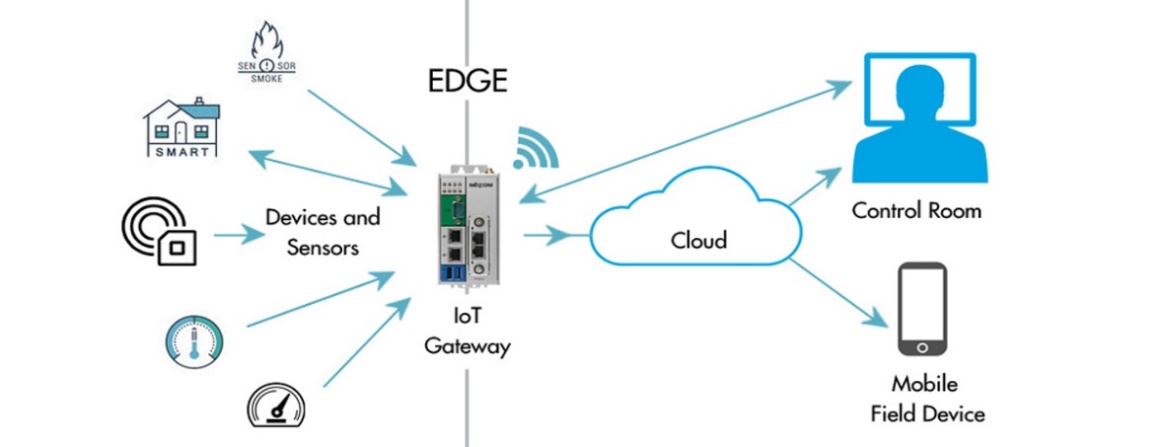
* Tìm hiểu IoT Gateway
* Thiết kế Schematic IoT Gateway
* Thiết kế Firmware
* Thiết kế software

# CƠ SỞ LÝ THUYẾT

## IOT Gateway:

### Tổng quan về IOI Gateway:

IoT Gateway là một thiết bị đóng vai trò là điểm kết nối giữa đám mây và các thiết bị điều khiển, cảm biến và các thiết bị thông minh, … Tất cả dữ liệu di chuyển lên đám mây hoặc ngược lại sẽ đi qua gateway này.



*Hình 2.1: Mô hình kết nối của IoT Gateway.*

IoT gateway thông thường sẽ là một thiết bị phần cứng chuyên dụng, có thiết kế linh hoạt, chịu được môi trường khắc nghiệt, hỗ trợ các chuẩn kết nối cục bộ như: LAN, WiFi, 3G, Zigbee, Z-wave, RF. Hoặc nó cũng có thể là một máy chủ thông thường nếu không gian lắp đặt cho phép, được cài đặt phần mềm gateway và tích hợp các cổng kết nối, wireless phù hợp.

### Các chức năng của IOT Gateway:

**Một IoT Gateway đa năng có thể thực hiện bất kỳ nhiệm vụ nào sau đây:**

* Tạo điều kiện giao tiếp với các thiết bị cũ hoặc không có kết nối internet.
* Bộ nhớ đệm dữ liệu, caching và media streaming.
* Xử lý trước dữ liệu (data preprocessing), làm sạch, lọc và tối ưu hóa.
* Tổng hợp dữ liệu thô.
* Liên lạc Device-to-Device / Machine-to-Machine.
* Tính năng kết nối mạng và lưu trữ dữ liệu trực tiếp.
* Trực quan hóa dữ liệu và phân tích dữ liệu cơ bản thông qua các ứng dụng IoT Gateway.
* Tính năng lịch sử dữ liệu ngắn hạn.
* Bảo mật – quản lý truy cập người dùng và các tính năng bảo mật mạng.
* Quản lý cấu hình thiết bị.
* Chẩn đoán hệ thống.

### Lợi ích khi sử dụng IOT Gateway:

Từ các chức năng trên, ta nhận thấy được lợi ích của việc sử dụng IoT Gateway:

1. ***Lọc dữ liệu trong giai đoạn tiền xử lý để cải thiện thời gian phản hồi***

Nếu tất cả dữ liệu này được đẩy thẳng lên hệ thống (bao gồm cả dữ liệu rác và dữ liệu không cần thiết), hoạt động phân tích dữ liệu diễn ra sau đó sẽ bị chậm hơn theo yêu cầu. IoT Gateway cung cấp các khả năng điều khiển bằng thuật toán xử lý dữ liệu tại biên, được gọi là điện toán biên. Cổng IoT có thể xử lý trước và lọc ra dữ liệu không cần thiết ở điểm cuối để có ít dữ liệu được chuyển tiếp sang giai đoạn tiếp theo.

1. ***Hỗ trợ bảo mật cho dữ liệu IoT***

IoT Gateway có thể là một giải pháp để tăng tính bảo mật cho thông tin của người dùng. Với việc tất cả các thiết bị IoT phải được kết nối với mạng lưới thông qua cổng IoT, sự tiếp xúc trực tiếp từ các thiết bị với hệ thống và người dùng đã được hạn chế tối đa. IoT gateway trong trường hợp này còn đóng vai trò người gác cổng mẫn cán, giúp bảo vệ toàn bộ hệ sinh thái IoT.

1. ***Cải thiện hiệu quả năng lượng trên các thiết bị IoT***

Những cổng IoT được bố trí gần với các thiết bị này được kết nối sẵn với mạng lưới thông tin có thể đóng vai trò là cầu nối để rút ngắn phạm vi kết nối của các tín hiệu và quản lý tất cả các giao tiếp cảm biến. Với phương pháp này, các cảm biến chỉ cần truyền tải dữ liệu trong phạm vi ngắn nên tiết kiệm năng lượng và kéo dài tuổi thọ hơn.

1. ***Xử lý các thông tin hỗn hợp từ các thiết bị IoT về một giao thức chuẩn duy nhất***

Các cảm biến thu thập dữ liệu từ môi trường bên ngoài. Dữ liệu được thu về có nhiều nguồn khác nhau (chuyển động, ánh sáng, âm thanh….). Các nội dung của dữ liệu này sẽ được mã hóa về dạng dữ liệu chung của hệ thống để có thể giao tiếp với nhau một cách có hiệu quả.

## Giới thiệu phần cứng:

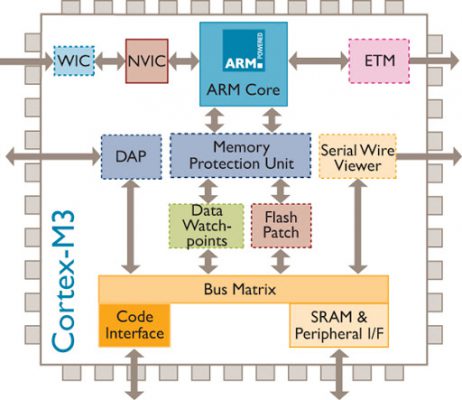
* + 1. **Vi điều khiển STM32F103C8T6:**

#### Giới thiệu bộ vi xử lý ARM – Cortex M3:

Nền tảng ARM Cortex đang được quan tâm nhiều hiện nay bởi vì các thiết bị ARM được bắt gặp ở mọi nơi. Các sản phẩm có chứa các thiết bị ARM có thể kể từ các hệ thống nhúng vi điều khiển nhỏ đến điện thoại di động, cho đến các máy chủ lớn hơn chạy Linux. ARM cũng sẽ sớm góp mặt nhiều hơn trong trung tâm dữ liệu. Đây là những lý do đủ thuyết phục để làm quen với công nghệ ARM.

Dòng Cortex gồm 3 nhánh: dòng A dành cho các ứng dụng cao cấp, dòng R dành cho các ứng dụng thời gian thực và dòng M dành cho các ứng dụng điều khiển và chi phí thấp.

Bộ vi xử lý ARM Cortex-M3 là bộ vi xử lý ARM đầu tiên dựa trên kiến trúc ARMv7-M và được thiết kế để đạt được hiệu suất cao trong các ứng dụng nhúng cần tiết kiệm năng lượng và chi phí, chẳng hạn như các vi điều khiển, hệ thống cơ ô tô, hệ thống kiểm soát công nghiệp và hệ thống mạng không dây. Thêm vào đó là việc lập trình được đơn giản hóa đáng kể giúp kiến trúc ARM trở thành một lựa chọn tốt cho ngay cả những ứng dụng đơn giản nhất.

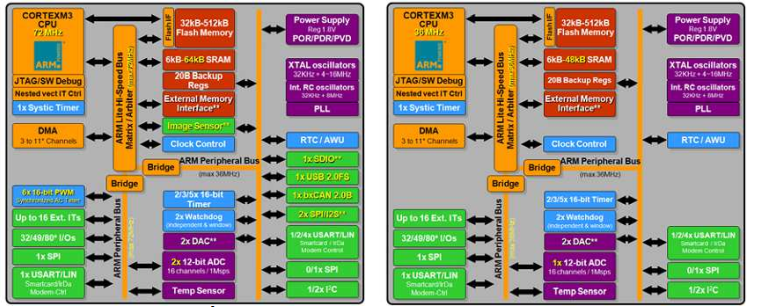


*Hình 2.2: Cấu trúc của vi xử lý ARM Cortex M3.*

#### Giới thiệu về dòng vi điều khiển STM32:

ST đã đưa ra thị trường 4 dòng vi điều khiển dựa trên ARM7 và ARM9, nhưng mà STM32 là một bước tiến quan trọng trên đường cong chi phí và hiệu suất (price/performance). STM32 là sự thách thức thật sự với các vi điều khiển 8 và 16-bit truyền thống.

STM32 đầu tiên gồm 14 biến thể khác nhau, được phân thành hai nhóm: dòng Performance có tần số hoạt động của CPU lên tới 72Mhz và dòng Access có tần số hoạt động lên tới 36Mhz. Các biến thể STM32 trong hai nhóm này tương thích hoàn toàn về cách bố trí chân (pin) và phần mềm, đồng thời kích thước bộ nhớ FLASH ROM có thể lên tới 128K và 20K SRAM.

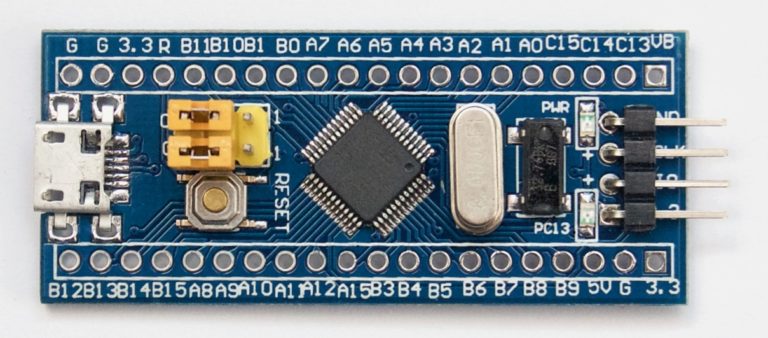


*Hình 2.3: Kiến trúc của STM32 nhánh Performance và Access.*

Các dòng STM32 được ST tích hợp thêm nhiều ngoại vi thích hợp cho các ứng dụng điều khiển đa dụng. Thành phần chính của STM32 là nhân Cortex M3, dùng I-Bus và D-Bus để kết nối với FLASH cũng như các ngoại vi. Ngoài ra thành phần quan trọng khác là DMA. Các ngoại vi được chia làm 2 nhóm kết nối đến hai giao diện khác nhau AHB-APB1 và AHB-APB2( có tốc độ tối đa lớn hơn AHB-APB1)

#### STM32F103:

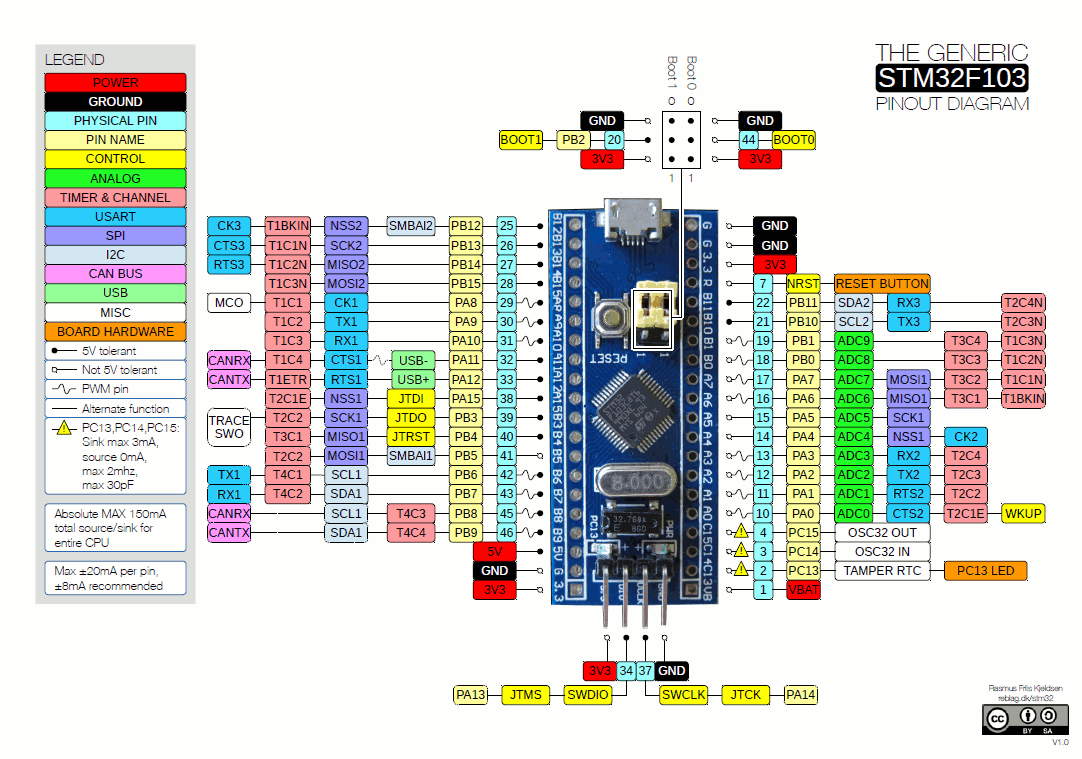
STM32 là một trong những dòng chip phổ biến của ST với nhiều họ thông dụng như F0, F1, F2, F3, F4….. STM32F103 thuộc họ F1 với lõi là ARM Cortex M3. STM32F103 là vi điều khiển 32 bit, tốc độ tối đa là 72Mhz. Giá thành cũng khá rẻ so với các loại vi điều khiển có chức năng tương tự. Mạch nạp cũng như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



*Hình 2.4: Vi điều khiển STM32F103C8T6*

Một số ứng dụng chính: 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…

* **Thông số kỹ thuật:**
* MCU: STM32F103C8T6.
* Core: ARM 32 Cortex-M3 CPU.
* Tần số: 72MHz.
* Bộ nhớ Flash: 64Kb.
* SRAM 20Kb.
* Điện áp I/O: 2.0~3.6 VDC.
* Thạch anh: 4~16MHz.
* Cổng MiniUSB dùng để cấp nguồn và giao tiếp.
* Kích thước: 5.3 x 2.2cm.
* **Các kết nối ngoại vi của STM32F103:**
* 4 × Cổng GPIO 16-bit (đa số chịu được điện áp 5-volt).
* 3 × USART (Universal Synchronous/Asynchronous Receiver/Transmitter).
* 2 × bộ điều khiển I2C.
* 2 × bộ điều khiển SPI.
* 2 × ADC (Analog Digital Converter).
* 2 × DMA (Direct Memory Address).
* 4 × timer (bộ định thời).
* 2 × đồng hồ báo hiệu (watchdog timers).
* 1 × bộ điều khiển USB.
* 1 × bộ điều khiển CAN.
* 1 × bộ phát CRC.

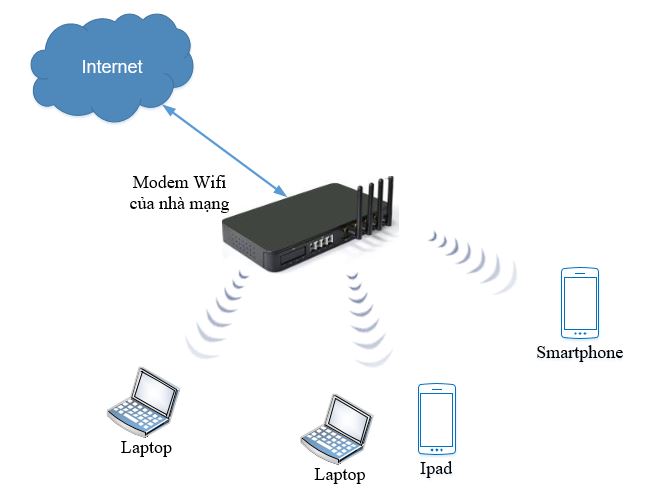


*Hình 2.5: Sơ đồ chức năng các chân KIT STM32F103C8T6*

### Module Wifi ESP8266 NodeMCU:

#### Tổng quan về giao thức Wifi:

Wifi là từ viết tắt của Wireless Fidelity, nó là một hệ thống hoạt động dựa trên sóng vô tuyến không dây, hay còn được gọi là mạng IEEE 802.11. Wifi cho phép truy cập mạng internet ở một khoảng cách xác định mà không cần kết nối vật lý.



*Hình 2.6: Sơ đồ kết nối mạng Wifi.*

Sóng Wifi ngày nay được sử dụng rộng rãi phục vụ đời sống con người, đặc điểm của sóng Wifi là thu phát ở tần số từ 2.4 GHz đến 5 GHz cao hơn so với sóng vô tuyến truyền hình, sóng điện thoại và radio nên khá an toàn trong vấn đề bảo toàn thông tin khi truyền và nhận dữ liệu.

* **Ưu và nhược điểm của Wifi:**
* **Ưu điểm:**

Ưu điểm của kết nối Wifi là tính tiện dụng, và đơn giản gọn nhẹ so với kết nối trực tiếp bằng cable truyền thống qua cổng RJ45. Người sử dụng có thể truy cập ở bất cứ vị trí nào trong vùng bán kính phủ sóng mà tại đó Router Wifi làm trung tâm.

Ưu điểm thứ hai của mạng sử dụng Wifi là dễ dàng sửa đổi và nâng cấp, người sử dụng có thể tăng băng thông truy cập, tăng số lượng người sử dụng mà không cần nâng cấp thêm Router hay dây cắm như các kết nối bằng dây vật lý. Tính thuận tiện: người truy cập có thể duy trì kết nối kể cả khi đang di chuyển, một ví dụ cụ thể là các Router Wifi lắp đặt trên các xe khách đường dài. Bên cạnh đó, tính bảo mật của mạng Wifi tương đối cao.

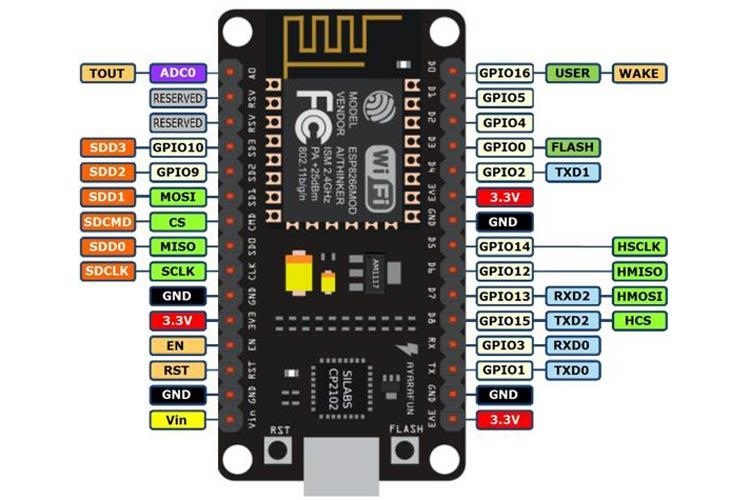
* **Nhược điểm:**

Bên cạnh những ưu điểm, mạng Wifi cũng tồn tại nhiều nhược điểm chưa thể khắc phục như: phạm vi kết nối của mạng Wifi với thiết bị có giới hạn, đi càng xa router kết nối càng yếu dần đi. Giải pháp cho vấn đề này là trang bị thêm các Repeater hoặc Access point. Tuy nhiên, gặp nhiều khó khăn do giá thành cao.

Nhược điểm tiếp theo của mạng Wifi là về vấn đề băng thông, càng nhiều người kết nối vào mạng thì tốc độ truy cập giảm rõ rệt.

#### Module ESP8266 NodeMCU:

NodeMCU được phát triển dựa trên Chip WiFi ESP8266EX bên trong Module  
ESP-12E dễ dàng kết nối WiFi với một vài thao tác.Board còn tích hợp IC CP2102,  
giúp dễ dàng giao tiếp với máy tính thông qua Micro USB để thao tác với board. Và  
có sẵn nút nhấn, led để tiện qua quá trình học, nghiên cứu. Ở đồ án này chúng em sử  
dụng phiên bản ESP8266 NodeMCU Lua CP2102.



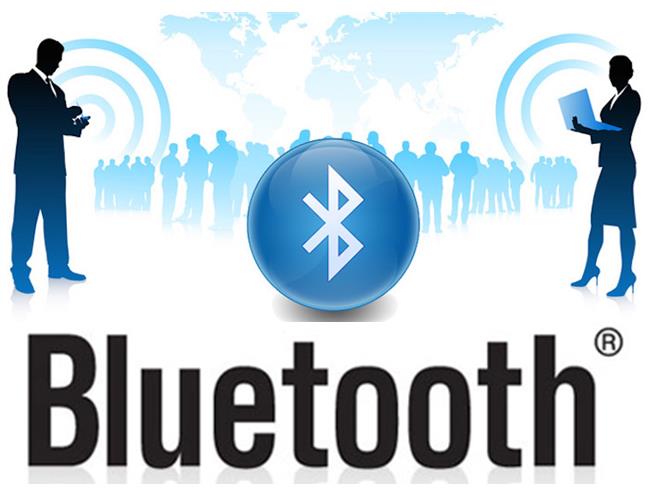
*Hình 2.7: NodeMCU ESP8266.*

* **Thông số kỹ thuật**:
* IC chính: ESP8266
* Phiên bản firmware: NodeMCU Lua
* Chip nạp và giao tiếp UART: CP2102.
* GPIO tương thích hoàn toàn với firmware Node MCU.
* Cấp nguồn: 5VDC Micro USB hoặc Vin.
* GPIO giao tiếp mức 3.3VDC
* Tích hợp Led báo trạng thái, nút Reset, Flash.
* Tương thích hoàn toàn với trình biên dịch Arduino IDE.
* Kích thước: 25 x 50 mm
  + 1. **Module Bluetooth HC05:**

#### Tổng quan về Bluetooth:

Bluetooth là công nghệ kết nối không dây nhằm trao đổi dữ liệu ở cự ly gần giữa các thiết bị điện tử. Nhờ công nghệ này, các dữ liệu sẽ được truyền tải thông qua sự kết nối giữa các thiết bị di động và cố định mà không cần dùng tới hệ thống dây nhợ lằng nhằng như kiểu vật lý trước đây. Trên thực tế, trong điều kiện hoàn hảo để kết nối thì Bluetooth có tầm hoạt động hiệu quả nhất trong khoảng cách 10m đổ lại.

Bluetooth sử dụng sóng Radio tần số 2.4GHz. Mặc dù đó cũng là tần số chung với công nghệ Wifi nhưng nó lại không hề có sự xung đột do bước sóng và phạm vi sử dụng của Bluetooth ngắn hơn rất nhiều. Những hãng muốn tích hợp Bluetooth vào trong sản phẩm của mình thì phải tuân theo các tiêu chuẩn nhất định. Thông qua những tiêu chuẩn kỹ thuật này mà các thiết bị có thể nhận ra và tương tác với nhau khi được kết nối Bluetooth.

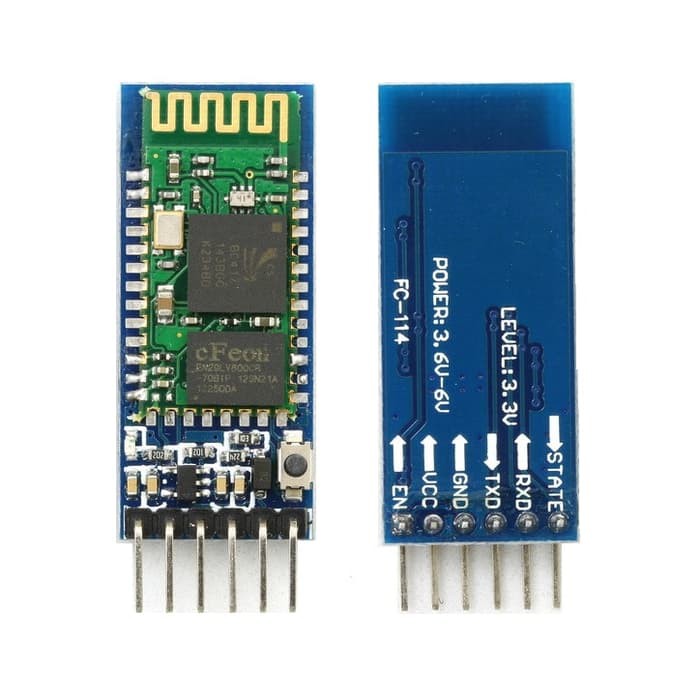


*Hình 2.8: Tổng quan về giao thức Bluetooth.*

* **Những ưu và nhược điểm của công nghệ Bluetooth:**
* **Ưu điểm:**
* Không cần dùng dây cáp nên rất gọn gàng.
* Không làm ảnh hưởng sức khỏe con người.
* Khả năng bảo mật cực cao. Do chỉ là mạng nội bộ và kín nên rất khó để bị nghe lén, ăn cắp thông tin.
* Các thiết bị có thể kết nối với nhau trong vòng 10m (ở điều kiện hoàn hảo) mà không cần tiếp xúc trực diện (hiện nay có loại Bluetooth kết nối lên đến 100m).
* Người dùng có thể dùng tai nghe Bluetooth để nghe điện thoại trong lúc lái xe.
* Chi phí thấp.
* Tốn ít năng lượng.
* Không gây nhiễu cho các thiết bị không dây khác.
* Khả năng tương thích cực cao giữa các thiết bị.
* **Nhược điểm:**
* Tốc độ thấp (tối đa khoảng 720 kbps).
* Kết nối có khi bị yếu nếu có nhiều vật cản.
* Thời gian thiết lập lâu

#### Module thu phát Bluetooth HC-05:

Module thu phát Bluetooth HC-05 dùng để thiết lập kết nối Serial giữa 2 thiết bị bằng sóng bluetooth. Điểm đặc biệt của module bluetooth HC-05 là module có thể hoạt động được ở 2 chế độ: MASTER hoặc SLAVE.



*Hình 2.9: Module Bluetooth HC05.*

* **Thông số kỹ thuật:**
* Điện áp hoạt động: 3.3 - 5V.
* Dòng điện khi hoạt động:
* Khi kết nối: 30mA.
* Khi truyền/nhận: 8 mA.
* Baud Rate: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.
* Băng tần ISM 2,4 GHz.
* Bluetooth protocol: Bluetooth Specification V2.0 + EDRo
* Điều chế: GFSK.
* Công suất phát: 4 dBm.
* Độ nhạy: -84 dBm ở 0,1% BER.
* Tốc độ: 1 Mbps.
* Bảo mật: xác thực và mã hóa.
* Kích thước: 26.9mm x 13mm.
* Thiết lập mặc định:
* Baudrate: 38400
* Pairing code: 1234.
* **Chế độ hoạt động:**

HC-05 có hai chế độ hoạt động là Command Mode và Data Mode. Chân KEY dùng để chuyển đổi qua lại giữa hai chế độ này.

* Ở chế độ Command Mode ta có thể giao tiếp với module thông qua cổng serial trên module bằng tập lệnh AT.
* Ở chế độ Data Mode module có thể truyền nhận dữ liệu với các module bluetooth khác.

Nếu đưa chân KEY lên mức logic cao trước khi cấp nguồn module sẽ đưa vào chế độ Command Mode với baudrate mặc định 38400. Chế độ này khá hữu ích khi bạn không biết baudrate trong module được thiết lập ở tốc độ bao nhiêu. Khi chuyển sang chế độ này đèn led trên module sẽ nháy chậm (khoảng 2s) và ngược lại khi chân KEY nối với mức logic thấp trước khi cấp nguồn module sẽ hoạt động chế độ Data Mode.

Nếu module đang hoạt động ở chế Data Mode để có thể đưa module vào hoạt động ở chế độ Command Mode bạn đưa chân KEY lên mức cao. Lúc này module sẽ vào chế độ Command Mode nhưng với tốc độ Baud Rate được bạn thiết lập lần cuối cùng. Chú ý nếu module chưa thiết lập lại lần nào thì mặc định của nó như sau:

* Baud rate: 9600, data 8 bits, stop bits 1, parity: none, handshake: none
* Passkey: 1234
* Device Name: HC-05
* **Sử dụng lệnh AT cho module HC05:**
* **Command Mode**:

|  |  |
| --- | --- |
| Lệnh | Chức năng |
| AT | Kiểm tra module có ở chế độ command không? |
| AT+VERSION? | Trả về firmware hiện tại của module |
| T+UART=9600,0,0 | Thiết lập baudrate 9600,1 bit stop, no parity |

* **MASTER:**

|  |  |
| --- | --- |
| Lệnh | Chức năng |
| AT+RMAAD | Ngắt kết nối với các thiết bị đã ghép |
| AT+RESET | Reset lại thiết bị |
| AT+CMODE=0 | Cho phép kết nối với bất kì địa chỉ nào |
| AT+INQM=0,2,5 | Dừng tìm khi đã tìm được 2 thiết bị hoặc sau 5s |
| AT+PSWD=1234 | Set Pin cho thiết bị |
| AT+INQ | Bắt đầu tìm kiếm thiết bị để ghép nối |
| AT+PAIR=<address>,<timeout> | Đặt timeout(s) khi kết nối với 1 địa chỉ slave |
| AT+LINK=<address> | AT+LINK=<address> |
| AT + NAME = <name> | Đặt tên cho module |
| AT+ROLE=0 | Đặt là chế độ SLAVE |

* **SLAVE:**

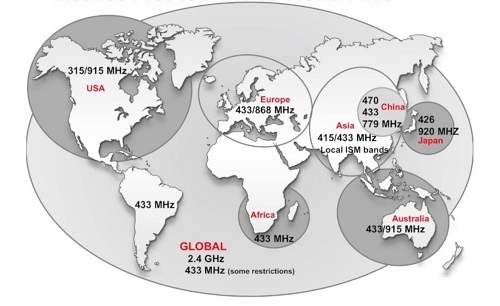
|  |  |
| --- | --- |
| Lệnh | Chức năng |
| AT+ORGL | Reset lại cài đặt mặc định |
| AT+RMAAD | Xóa mọi thiết bị đã ghép nối |
| AT+ADDR | Hiển thị địa chỉ của SLAVE |
| AT + NAME = <name> | Đặt tên cho module |
| AT+PSWD=1234 | Set Pin cho thiết bị |
| AT+ROLE=1 | Đặt là module ở master |

* + 1. **Module Sub-GHz CC1101:**

#### Công nghệ Sub-GHz:

Công nghệ Sub - GHz hoạt động ở dải tần nhỏ hơn 1 GHz. Các quốc gia trên thế giới chủ yếu sử dụng dải tần miễn phí cấp phép 433 Mhz. Tuy nhiên có một số dải tần khác được sử dụng ở một số khu vực:

* Bắc Mỹ và Úc là 915 Mhz
* Châu Âu là 868 Mhz
* Trung Quốc là 470 Mhz và 779 Mhz
* Nhật Bản là 426 Mhz và 920 Mhz.



* + - **Đặc điểm:**

Sub - GHz có thể dễ dàng đạt được vài trăm mét trong nhà và vài km (dặm) ngoài trời, tùy thuộc vào điều kiện, có tần số thấp, bước sóng dài nên dễ dàng lan truyền qua các vật cản như tường, cây cối, tòa nhà trong các đô thị và trong nhà máy. Bước sóng dài của Sub – GHz làm cho ít suy hao đường truyền nên truyền đi được xa hơn.

Sub-GHz sử dụng băng tần hẹp và tốc độ truyền thấp nên tiêu thụ ít năng lượng. Sub-GHz cần tín hiệu năng lượng thấp từ bộ truyền cho cùng một tín hiệu ở bộ nhận nên Sub-GHz phù hợp cho các thiết bị cảm biến IoT dùng pin. Một viên pin có thể cấp năng lượng cho cảm biến Sub-GHz lên tới 10-20 năm. Sub-GHz hoạt động ở dải tần thấp hơn và có rất ít thiết bị wireless hoạt động ở dải tần này nên sẽ chống nhiễu tốt hơn.

* **Ứng dụng:**

Cảm biến wireless Sub-GHz bao gồm 2 phần chính: phần cảm biến và phần truyền wireless:

* Phần cảm biến có thể là module cảm biến hoặc các cảm biến hoàn chỉnh. Phần truyền wireless bao gồm mạch xử lý tín hiệu và anten truyền.
* Phần truyền wireless thường sử dụng pin để cấp năng lượng cho phần truyền và cho cả phần cảm biến.

Bộ thu wireless Sub - 1GHz gồm ăng ten thu, bộ xử lý tín hiệu và cổng xuất dữ liệu . Mỗi bộ thu wireless có thể kết nối dạng sao với nhiều bộ cảm biến wireless trong vùng phủ sóng của bộ thu. Vùng phủ sóng của bộ thu có thể lên tới vài km khi 2 ăng ten thu và phát nhìn thấy nhau. Khoảng cách truyền sẽ nhỏ hơn nếu có vật cản giữa cảm biến phát và bộ thu wireless. Dữ liệu từ bộ thu wireless có thể kết nối có dây đến PLC, HMI, SCADA, IoT gateway để phục vụ cho việc giám sát, phân tích dữ liệu và lập báo cáo.

#### Module CC1101:

#### Tổng quan:

CC1101 - được phát triển bởi tạp đoàn **Texas Instruments** (thường được viết tắt là TI) - là bộ thu phát Sub -1GHz giá rẻ dành cho các ứng dụng không dây yêu cầu rất ít năng lượng. Mạch được thiết kế để hoạt động trong các dải tần là 315, 433, 868 và 915 MHz và cho phép người dùng truyền nhận dữ liệu trong khoảng cách 500m, tốc độ truyền lên đến 600kbps.

**/**

*Hình 2.4: Module CC1101*

* **Thông số kỹ thuật:**
* Điện áp hoạt động: 1.8V ~ 3.6V
* Độ nhạy của máy thu: -110dBm với tốc độ truyền 1200
* Công suất phát tối đa: 10mW (+ 10dBm)
* Tốc độ truyền tối đa: 500kbps
* Băng tần 315Mhz, 433Mhz, 868Mhz, 915Mhz
* Tốc độ truyền 500kbps
* **Mục đích:**
* Truyền tín hiệu từ nơi phát đến nơi thu.
* Khoảng cách xa, chính xác, ít sai số.

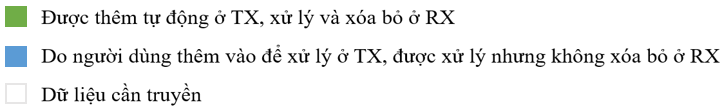
#### Truyền nhận dữ liệu giữa 2 module CC1101:

##### Xử lý dữ liệu ở máy phát:

Một gói dữ liệu truyền đi có dạng:

* Phần mở đầu
* Đồng bộ ký tự
* Chiều dài gói data tùy chọn
* Địa chỉ gói data tùy chọn
* Data
* 2 byte CRC tùy chọn

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Phần mở đầu  1010…1010  (8 x n bit) | Đồng bộ ký tự  (16/32 bits) | Chiều dài  (8 bits) | Địa chỉ  (8 bits) | Data | CRC  (16 bits) |



Số byte mở đầu được lập trình với giá trị MDMCFG1.NUM\_PREAMBLE.

#define CC1101\_MDMCFG1 0x13 // Set 3 byte phần mở đầu

Đồng bộ ký tự là một giá trị hai byte được đặt trong thanh ghi SYNC1 và SYNC0.

#define CC1101\_SYNC1 0x04 // Sync word, high INT8U

#define CC1101\_SYNC0 0x05 // Sync word, low INT8U

Byte chiều dài gói tin phải được ghi vào TX FIFO. Giá trị byte độ dài bằng chiều dài của gói dữ liệu (bao gồm cả byte địa chỉ tùy chọn).

Khi bật TX, bộ điều chế sẽ bắt đầu truyền phần mở đầu. Khi phần mở đầu truyền xong, bộ điều chế sẽ truyền tiếp byte đồng bộ ký tự và sau đó là dữ liệu từ TX FIFO nếu có sẵn dữ liệu. Nếu TX FIFO trống, bộ điều chế sẽ tiếp tục gửi các byte mở đầu cho đến khi byte đầu tiên được ghi vào TX FIFO. Sau đó, bộ điều biến sẽ gửi từ đồng bộ và sau đó là các byte dữ liệu.

Dữ liệu phía sau byte đồng bộ ký tự được xáo trộn và mã hóa bởi Interleaver và FEC trước khi được điều chế.

##### Xử lý dữ liệu ở máy thu:

Trong chế độ nhận, bộ giải điều chế và xử lý gói sẽ tìm kiếm các byte mở đầu và đồng bộ ký tự hợp lệ. Nếu tìm được sẽ bắt đầu nhận byte dữ liệu đầu tiên. Bộ xen kẽ sẽ khử xáo trộn các bit trước khi các quá trình xử lý khác được thực hiện.

Bộ giải mã FEC sẽ bắt đầu giải mã byte dữ liệu đầu tiên và bộ xử lý gói sẽ lưu trữ giá trị này dưới dạng độ dài gói. Tiếp theo là byte địa chỉ, bộ xử lý gói kiểm tra địa chỉ và chỉ tiếp tục nhận nếu địa chỉ trùng khớp.

Khi kết thúc tải trọng, trình xử lý gói sẽ tùy chọn ghi thêm hai byte trạng thái gói chứa trạng thái CRC, chỉ báo chất lượng liên kết và giá trị RSSI.

##### Định dạng điều chế tín hiệu:

CC1101 hỗ trợ các định dạng điều chế 2-FSK, GFSK và MSK.

CC1101 cũng có khả năng sử dụng bộ lọc Gaussian để định hình 2-FSK (GFSK). Tính năng định hình phổ này cải thiện công suất kênh lân cận (ACP) và băng thông bị chiếm dụng. Trong các hệ thống 2-FSK phù hợp với sự thay đổi tần số đột ngột, phổ tần vốn rộng. Bằng cách làm cho sự dịch chuyển tần số trở nên 'mềm mại hơn', phổ có thể được thu hẹp đáng kể. Do đó, tốc độ dữ liệu cao hơn có thể được truyền trong cùng một băng thông bằng cách sử dụng GFSK.

Định dạng điều chế được đặt trong thanh ghi MDMCFG2.MOD\_FORMAT.

#define CC1101\_MDMCFG2 0x12 // Định dạng điều chế GFSK

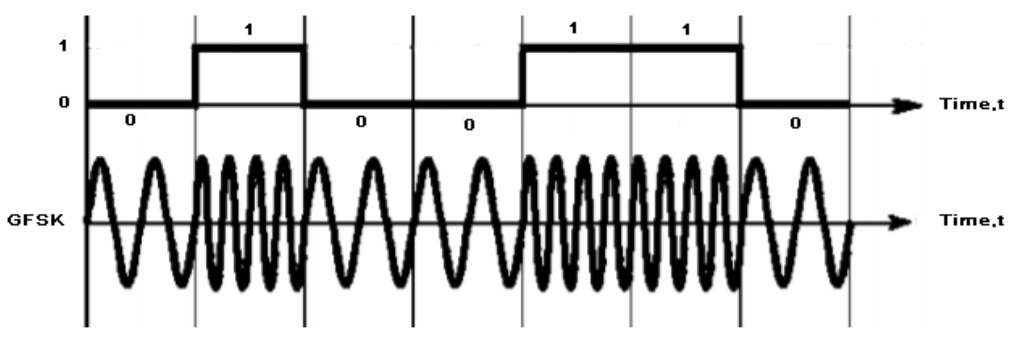
Độ lệch tần số được lập trình với các giá trị DEVIATION\_M và DEVIATION\_E trong thanh ghi DEVIATN.

#define CC1101\_DEVIATN 0x15 // DEVIATION\_M = 5

// DEVIATION\_E = 1

* **Kỹ thuật điều chế GFSK:**

Điều chế FSK ấn định các tần số khác nhau cho sóng mang tùy thuộc vào bit được truyền đi. Do đó, khi một bit 0 được truyền đi, sóng mang giả định một tần số tương ứng với một bit 0 trong thời gian của một bit. Khi bit 1 được truyền đi, tần số sóng mang được thay đổi thành giá trị tương ứng với bit 1 và tương tự, vẫn giữ nguyên tần số này trong suốt thời gian của một bit, như thể hiện trong hình sau:



*Hình 2.5: Điều chế FSK*

Trong điều chế Gaussian FSK (GFSK) dữ liệu được mã hóa dưới dạng các biến thể của tần số trong sóng mang theo cách tương tự như FSK. Do đó, bộ điều chế được sử dụng có thể giống bộ điều chế được sử dụng cho điều chế FSK. Tuy nhiên, các xung đi qua một bộ lọc Gaussian trước khi đi vào bộ điều biến xung để giảm độ rộng phổ của cùng một bộ lọc Gaussian.



*Hình 2.6 : Gaussian Filter*

Bộ lọc này có ưu điểm là giảm công suất dải biên, giảm nhiễu với các kênh lân cận.

Độ lệch tần số được tính theo công thức:

*= DEVIATION\_M)\**

Như vậy, giá trị của độ lệch là ± 5,157kHz, với tần số tinh thể là 26,0 MHz.

Bit 0: - deviation

Bit 1: + deviation

##### Sửa lỗi:

Module CC1101 hỗ trợ tích hợp sửa lỗi chuyển tiếp và xen kẽ.

* **Sửa lỗi chuyển tiếp (FEC):**

Để bật tùy chọn này, phải đặt thanh ghi MDMCFG1.FEC\_EN = 1.

Sửa lỗi chuyển tiếp (FEC) là một quá trình trong đó kết quả của các thuật toán được gửi dưới dạng thông tin bổ sung cùng với dữ liệu từ đầu truyền. Bằng cách lặp lại các thuật toán tương tự ở đầu xa, máy thu có khả năng phát hiện lỗi ở mức bit đơn và sửa chúng (lỗi có thể sửa được) mà không cần phải truyền lại dữ liệu.

Lược đồ FEC được áp dụng cho CC1101 là mã hóa tích chập. Bộ mã tích chập là mã tỷ lệ 1/2 với độ dài giới hạn là m = 4. Mã hóa một bit đầu vào và tạo ra hai bit đầu ra. Do đó, tốc độ dữ liệu hiệu quả giảm đi một nửa. Để truyền ở cùng một dải dữ liệu hiệu quả khi sử dụng FEC, cần phải sử dụng băng thông gấp đôi. Điều này sẽ yêu cầu băng thông máy thu cao hơn, và do đó làm giảm độ nhạy.

FEC được sử dụng trên trường dữ liệu và từ CRC để giảm tỷ lệ lỗi bit tổng khi hoạt động gần giới hạn độ nhạy. Dữ liệu bổ xung được thêm vào dữ liệu được truyền theo cách mà người nhận có thể khôi phục dữ liệu ban đầu khi có một số lỗi bit. Việc sử dụng FEC cho phép tiếp nhận chính xác hơn ở SNR thấp hơn, do đó mở rộng phạm vi truyền thông nếu băng thông máy thu không đổi.Ngoài ra, đối với SNR nhất định, việc sử dụng FEC làm giảm tỷ lệ lỗi gói (PER). Vì tỷ lệ lỗi gói (PER) có liên quan đến BER bởi:

PER = (1-(1-BER))\*packet\_length

Cuối cùng, trong môi trường vô tuyến ISM thực tế, các hiện tượng nhất thời và thay đổi theo thời gian sẽ tạo ra các lỗi không thường xuyên ngay cả trong các điều kiện thu sóng tốt. FEC sẽ che giấu các lỗi như vậy và kết hợp với việc xen kẽ các dữ liệu được mã hóa, thậm chí sửa các lỗi tiếp nhận bị lỗi trong thời gian tương đối dài (lỗi liên tục).

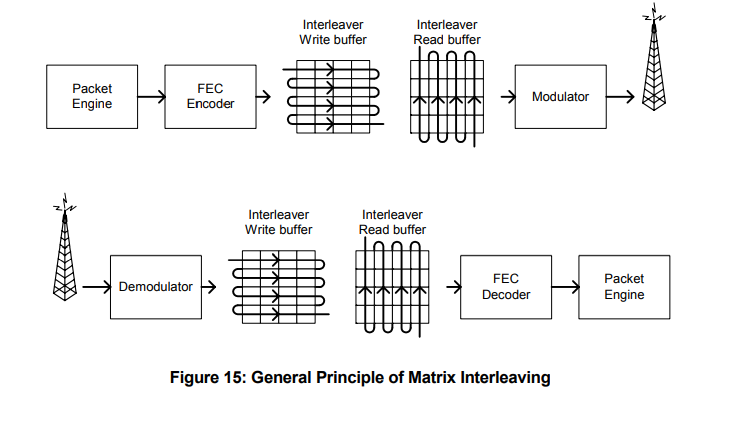
* **Sửa lỗi xen kẽ:**

Dữ liệu nhận được qua các kênh vô tuyến sẽ thường gặp lỗi liên tục do nhiễu và cường độ tín hiệu thay đổi theo thời gian. Để giảm các lỗi kéo dài nhiều bit, phép xen kẽ được sử dụng khi FEC được kích hoạt. Sau khi loại bỏ xen kẽ, một khoảng lỗi liên tục trong luồng đã nhận sẽ trở thành các lỗi đơn lẻ được phân tán.

Trong bộ truyền, các bit dữ liệu từ bộ mã chập tốc độ ½ được ghi vào các hàng của ma trận, trong khi chuỗi bit cần truyền được đọc từ các cột của ma trận. Ngược lại, trong máy thu, các ký hiệu nhận được được ghi vào các cột của ma trận, trong khi dữ liệu được truyền vào bộ giải mã tích hợp được đọc từ các hàng của ma trận.

Bộ đệm xen kẽ và loại bỏ xen kẽ trên CC1101 là ma trận 4 x 4.

Sơ đồ:



Hình 2.7 : Nguyên tắc chung của xen kẽ ma trận

Khi sửa lỗi chuyển tiếp và xen kẽ được sử dụng, cần ít nhất một byte bổ sung. Lượng dữ liệu được truyền phải là bội số của kích thước của bộ đệm xen kẽ (hai byte). Do đó, phần cứng điều khiển gói tự động chèn thêm một hoặc hai byte vào cuối gói, để tổng độ dài của dữ liệu được xen kẽ là một số chẵn. Những byte thừa này sẽ bị loại bỏ trước khi gói nhận được đi vào RX FIFO. Khi sử dụng FEC và xen kẽ, tải trọng dữ liệu tối thiểu là 2 byte.

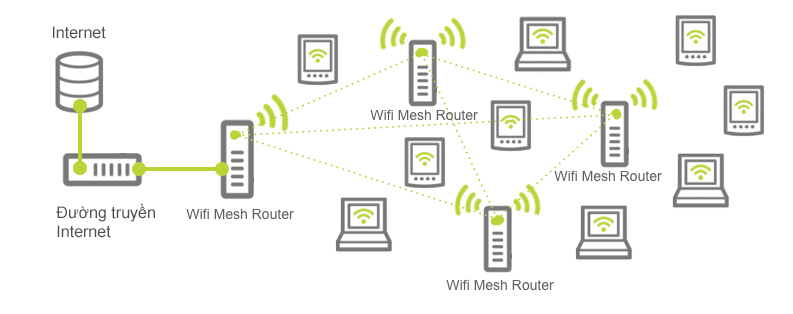
## Giới thiệu phần mềm:

## Mạng Mesh Wifi:

### **Tổng quan về mạng mesh wifi:**

Trong quá trình phát triển IoT thường đòi hỏi việc tăng số lượng node kết nối với  
Internet. Các kết nối thông thường dùng cơ chế Client - Server. Nhược điểm lớn  
nhất là số lượng node có thể trực tiếp kết nối tới router/server bị giới hạn về cả số  
lượng và hiệu suất.

Để khắc phục điểm này thì giao thức MESH đã ra đời. Trong giao thức này node  
có thể tạo ra mạng để chuyển tiếp gói tin cho nhau, nhờ đó mà một số lượng lớn node  
có thể kết nối với Server/Internet mà không cần phải cải tiến, nâng cấp router/server.



*Mô hình mạng Mesh Wifi .*

Wifi mesh về cơ bản là một hệ thống mạng Wifi diện rộng, là tập hợp bởi các thiết bị phát sóng Wifi (có thể là bộ phát sóng access point hay bộ định tuyến router) hoạt động chung với nhau một cách thống nhất. Tín hiệu của Wifi mesh có thể trải trong quy mô rộng lớn, khi mà bộ phát sóng hay bộ kích sóng Wifi không đáp ứng được.

* **Một số ưu điểm của hệ thống Mesh:**
* Vị trí lắp đặt các AP linh động và dễ dàng
* Khả năng chịu tải tốt, ổn định, kết nối liền mạch nhờ chỉ sử dụng một SSID chung.
* Dễ dàng mở rộng hệ thống, phù hợp cho việc sử dụng các thiết bị dạng “smarthome”.
* Giảm thiểu chi phí triển khai: các AP kết nối với nhau thông qua giao diện không dây nên không phải kéo dây mạng trực tiếp tới các AP.

### **Cấu trúc mạng**:

**Các node:** sẽ kết nối trực tiếp tới router được gọi là root node, các node khác thì được gọi là non-root node.

**Online-Mesh:** Khi router kết nối với internet thì ta có thể dùng IOT App để điều  
khiển từ xa ở bất kỳ đâu

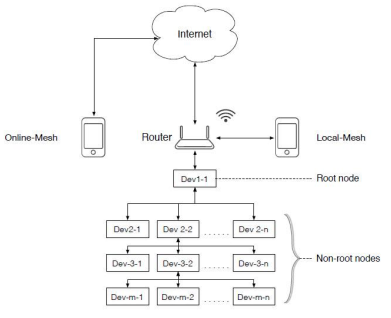
**Local-Mesh:** Bạn chỉ có thể điều khiển local device trong mạng thông qua router.

**Root Node:**

* Nhận và gửi gói tin
* Chuyển tiếp gói tin từ server, ứng dụng mobile và các node con của nó

**None-root Node:**

* Non-leaf node: Nhận và gửi gói tin, chuyển tiếp gói tin từ node cha và các node con khác
* Leaf node: Chỉ được nhận và gửi gói tin, không có chức năng chuyển tiếp.



*Cấu trúc mạng mesh wifi.*

## Tổng quan về mạng Internet, giao thức HTTP và mô hình MVC:

### Mạng Internet

Internet là một hệ thống thông tin toàn cầu có thể được truy cập công cộng gồm các mạng máy tính được liên kết với nhau. Hệ thống này truyền thông tin theo kiểu nối chuyển gói dữ liệu (packet switching) dựa trên một giao thức liên mạng đã được chuẩn hóa (giao thức IP). Hệ thống này bao gồm hàng ngàn mạng máy tính nhỏ hơn của các doanh nghiệp, của các viện nghiên cứu và các trường đại học, của người dùng cá nhân, và các chính phủ trên toàn cầu. Chúng cung cấp một khối lượng thông tin và dịch vụ khổng lồ trên internet.

* **Hoạt động của mạng Internet:**

Các Giao Thức (Protocols):là tập các luật mà các máy tính phải tuân theo khi giao tiếp trên Internet.

* Đảm bảo việc truyền nhận dữ liệu. TCP là giao thức hướng kết nối (connection-oriented protocol).
* Internet Protocol (IP):điều chỉnh đường đi của những gói dữ liệu đường truyền nhận trên Internet.

TCP/IPđược dùng làm giao thức chuẩn khi giao tiếp trên Internet.

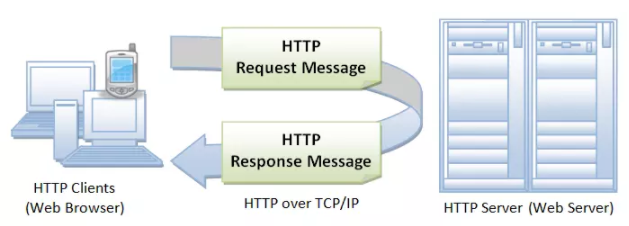
* Địa Chỉ IP (IP Address): Internet là một mạng kết nối rộng lớn giữa các máy tính. Để xác định một máy tính trên mạng này, người ta dùng một con số gọi là địa chỉ IP. Địa chỉ IP gồm một tập 4 số nhỏ hơn 255 và được ngăn cách bởi các dấu ‘.’

Transmission Control Protocol (TCP): thiết lập kết nối giữa hai máy tính để truyền tải dữ liệu, chia dữ liệu thành những gói nhỏ và

### Giao thức HTTP

#### Tổng quan về giao thức HTTP:

Hyper Text Transfer Protocol (**HTTP)** là một trong các giao thức chuẩn về mạng Internet, được dùng để liên hệ thông tin giữa Máy cung cấp dịch vụ (Web server) và Máy sử dụng dịch vụ (Web client), là giao thức Client/Server dùng cho World Wide Web – WWW.

**

*Sơ đồ hoạt động của HTTP*

**HTTP**là một **stateless protocol**. Hay nói cách khác, request hiện tại không biết những gì đã hoàn thành trong request trước đó.

HTTP cho phép tạo các yêu cầu gửi và nhận các kiểu dữ liệu, do đó cho phép xây dựng hệ thống độc lập với dữ liệu được chuyển giao.

Giao tiếp giữa máy khách và máy chủ là một hướng kết nối và không xác định rõ ràng QoS.

#### Các thành phần của HTTP:

* ***HTTP – Requests*:**
* **HTTP Request Method:** Là phương thức để chỉ ra hành động mong muốn được thực hiện trên tài nguyên đã xác định. **1 số HTTP Request method thường dùng:** GET, POST, HEAD, PUT.
* Cấu trúc của một HTTP Request:
* Một R*equest-line = Phương thức + URI–Request + Phiên bản HTTP.*  Giao thức HTTP định nghĩa một tập các giao thức GET, POST, HEAD, PUT... Client có thể sử dụng một trong các phương thức đó để gửi request lên server.
* Khi request đến server, server thực hiện một trong 3 hành động sau:
* Server phân tích request nhận được, maps yêu cầu với tập tin trong tập tài liệu của server, và trả lại tập tin yêu cầu cho client.
* Server phân tích request nhận được, maps yêu cầu vào một chương trình trên server, thực thi chương trình và trả lại kết quả của chương trình đó.
* Request từ client không thể đáp ứng, server trả lại thông báo lỗi.
* ***HTTP – Responses:***
* Cấu trúc của một HTTP response:
* Một *Status-line = Phiên bản HTTP + Mã trạng thái + Trạng thái*
* **Mã trạng thái**: Thông báo về kết quả khi nhận được yêu cầu và xử lý bên server cho client. **Các kiểu mã trạng thái:**
* 1xx: Thông tin (100 -> 101)
* 2xx: Thành công (200 -> 206)
* 3xx: Sự điều hướng lại (300 -> 307)
* 4xx: Lỗi phía Client (400 -> 417)
* 5xx: Lỗi phía Server (500 -> 505)
* Ví dụ khi dùng trình duyệt truy cập Web, sẽ thường gặp các thông báo lỗi khác nhau như sau:
* Lỗi 404 hay Http 404: là lỗi không tồn tại địa chỉ đang truy cập.
* Lỗi 401: truy cập vào nơi yêu cầu xác thực, nhưng không vượt qua được sẽ có lỗi này.
* Lỗi 500: lỗi này thường do Web server truy cập bị lỗi nên không thể truy cập vào được.
* Ngoài ra Http 200 tức là bạn truy cập thành công.

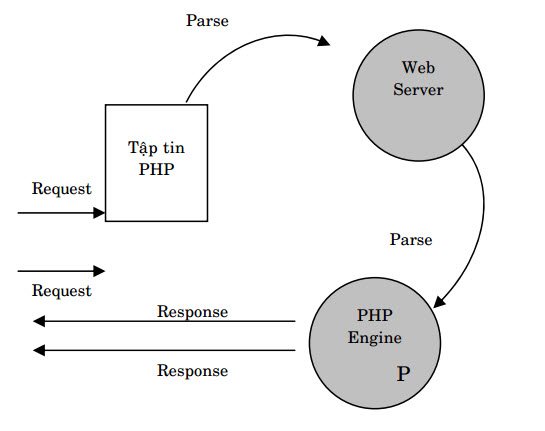
#### Đánh giá HTTP:

* **Ưu điểm:**
* Định địa chỉ dễ dàng: HTTP chỉ định địa chỉ IP với các tên dễ nhận biết, cho phép chúng dễ dàng được xác định trên world wide web.
* Khả năng xử lý lượng lớn dữ liệu.
* Tính linh hoạt: HTTP có khả năng tải xuống các tiện ích mở rộng hoặc bổ trợ khi được yêu cầu, ví dụ như trình phát Flash.
* **Nhược điểm:**
* Tính toàn vẹn của dữ liệu: Không có bất kỳ phương pháp mã hóa nào được sử dụng trong HTTP. Điều này làm cho HTTP không an toàn và dễ gặp các vấn đề về dữ liệu.
* Quyền riêng tư của dữ liệu: Mức độ bảo mật thấp khiến cho quyền riêng tư của dữ liệu không đảm bảo.
* Tiêu thụ nhiều điện năng: HTTP sử dụng nhiều tài nguyên hệ thống hơn, dẫn đến tiêu thụ nhiều điện năng hơn.

### Ngôn ngữ lập trình PHP:

#### Tổng quan:

PHP - viết tắt hồi quy của "Hypertext Preprocessor", là một ngôn ngữ lập trình kịch bản được chạy ở phía server nhằm sinh ra mã html trên client. PHP đã trải qua rất nhiều phiên bản và được tối ưu hóa cho các ứng dụng web, với cách viết mã rõ ràng, tốc độ nhanh, dễ học nên PHP đã trở thành một ngôn ngữ lập trình web rất phổ biến và được ưa chuộng.



*Hình 2.10: Mô hình hoạt động của php.*

 PHP chạy trên môi trường Webserver và lưu trữ dữ liệu thông qua hệ quản trị cơ sở dữ liệu nên PHP thường đi kèm với Apache, MySQL và hệ điều hành Linux (LAMP).

#### Phương thức $\_get và $\_post trong PHP:

* **Phương thức GET**: gửi thông tin người dùng được mã hóa vào yêu cầu trang. Trang này và thông tin được mã hóa được ngăn cách bởi ký tự *“?”*.

Ví dụ: http://www.test.com/index.htm?name1=value1&name2=value2

* Phương thức GET tạo ra một chuỗi dài xuất hiện trên nhật ký máy chủ, trong phần Location của trình duyệt: box.
* Phương thức GET có thể gửi được tối đa chỉ 1024 ký tự.
* Không được sử dụng phương thức GET khi bạn gửi mật khẩu hoặc các thông tin nhạy khác đến máy chủ.
* Phương thức GET không thể được sử dụng để gửi dữ liệu nhị phân đến máy chủ, ví dụ như hình ảnh hoặc tài liệu word.
* Dữ liệu được gửi bằng phương thức GET có thể được truy cập bằng cách sử dụng biến môi trường QUERY\_STRING.
* PHP cung cấp mảng liên kết $\_GET để truy cập tất cả thông tin đã được gửi bằng phương thức GET.
* **Phương thức POST**: giúp chuyển đổi thông tin qua HTTP headers. Thông tin được mã hóa tương tự như khi sử dụng phương thức GET và được đặt trong header có tên là QUERY\_STRING.
* Phương thức POST không giới hạn kích thước dữ liệu được gửi.
* Phương thức POST có thể được sử dụng để gửi bảng mã ASCII cũng như dữ liệu nhị phân.
* Dữ liệu được gửi bởi phước thức POST sẽ đi qua HTTP header, do đó, tính bảo mật phụ thuộc vào giao thức HTTP. Bằng cách sử dụng Secure HTTP (phần mở rộng cho Giao thức truyền siêu văn bản), bạn có thể an tâm bởi vì mọi thông tin của mình đều được bảo mật.
* PHP cung cấp mảng kết hợp $\_POST để truy cập tất cả thông tin được gửi bằng phương thức POST.
* Nhìn chung, cả $\_POST và $\_GET trong PHP đều được sử dụng để gửi dữ liệu lên server. Tuy nhiên, giữa chúng vẫn tồn tại một số khác biệt:
* *Xét về tính bảo mật*: Phương thức POST có tính bảo mật cao hơn, bởi nó không hiển thị dữ liệu trên thanh URL
* *Xét về giới hạn ký tự:*
* POST không giới hạn kích thước dữ liệu
* GET lại có giới hạn độ dài ký tự (tối đa 1024 ký tự).
* *Xét về mức độ hỗ trợ kiểu dữ liệu:*
* POST hỗ trợ khá nhiều kiểu dữ liệu như String (chuỗi), Numeric (số), Binary (nhị phân), …
* GET chỉ hỗ trợ một kiểu dữ liệu là kiểu String.
* *Xét về tốc độ xử lý*: Phương thức POST xử lý dữ liệu mất nhiều thời gian hơn là phương thức GET

## Quản lý cơ sở dữ liệu với phpMyAdmin:

### Tổng quan về cơ sở dữ liệu và hệ quản trị cơ sở dữ liệu:

#### Cơ sở dữ liệu:

Cơ sở dữ liệu (Database): là một hệ thống các thông tin có cấu trúc, được lưu trữ trên các thiết bị lưu trữ nhằm thỏa mãn yêu cầu khai thác thông tin đồng thời của nhiều người sử dụng hay nhiều chương trình ứng dụng chạy cùng một lúc với những mục đích khác nhau.

#### Hệ quản trị cơ sở dữ liệu:

Mysql làmột hệ thống quản trị cơ sở dữ liệu mã nguồn mở (gọi tắt là RDBMS) hoạt động theo mô hình client-server. Mysql được sử dụng rất rộng rãi trên thế giới, đặc biệt là các ứng dụng website và nó thường đi kết hợp với ngôn ngữ lập trình PHP để xây dựng các ứng dụng website. Các hệ thống web ưa chuộng MYSQL là tại vì tốc độ xử lý của nó cao, tính dễ sử dụng và thương thích với các hệ điều hành thông dụng hiện nay như Linux, Window,…

### ****Phần**** mềm ****PHPMyAdmin:****

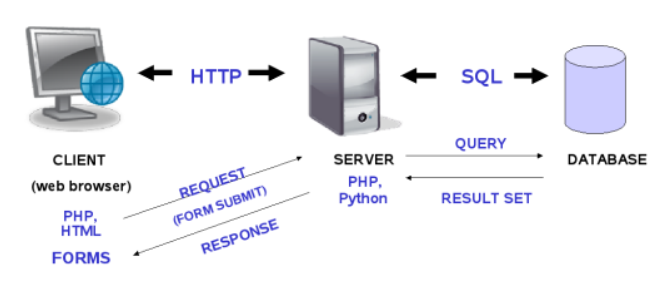
PHPMyAdmin là một công cụ nguồn mở được viết bằng ngôn ngữ PHP để giúp người dùng quản lý cơ sở dữ liệu của MySQL thông qua một trình duyệt web. Đây là công cụ quản trị MySQL phổ biến nhất được sử dụng bởi hàng triệu người dùng trên toàn thế giới, đặc biệt là các nhà quản trị cơ sở dữ liệu hay database administrator.

Thay vì sử dụng giao diện cửa sổ dòng lệnh (command line interface), với trình duyệt web của mình, thông qua giao diện người dùng (user interface), phpMyAdmin có thể thực hiện nhiều tác vụ như tạo, cập nhật, sửa đổi hoặc xóa bỏ cơ sở dữ liệu, bảng hoặc bản ghi; thực hiện báo cáo SQL; phân quyền và quản lý người dùng,…

Một số tính năng chung thường được sử dụng trên phpMyAdmin:

* Quản lý user(người dùng): thêm, xóa, sửa(phân quyền).
* Quản lý cơ sở dữ liệu: tạo mới, xóa, sửa, thêm bảng, hàng, trường, tìm kiếm đối tượng.
* Nhập xuất dữ liệu(Import/Export): hỗ trợ các định dạng SQL, XML và CSV.
* Thực hiện các truy vấn MySQL, giám sát quá trình và theo dõi.
* Sao lưu và khôi phục(Backup/Restore): Thao tác thủ công.

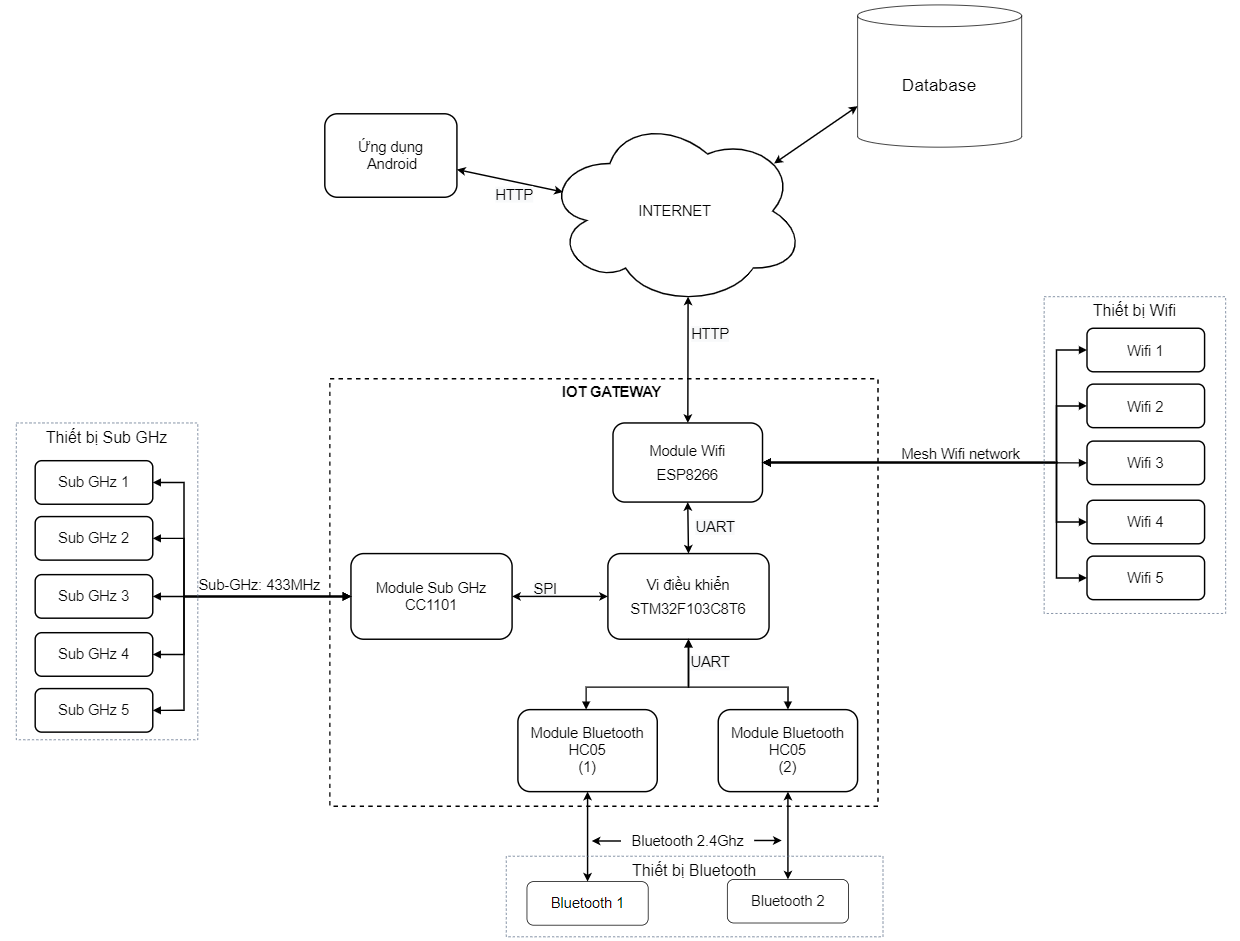
Việc kết hợp cơ sở dữ liệu với ứng dụng web, dữ liệu xử lý do PHP sẽ tương tác với cơ sở dữ liệu như lấy dữ liệu.



*Hình 2.11: Sơ đồ tương tác giữa CLIENT, SERVER và DATABASE*

# THIẾT KẾ VÀ THỰC HIỆN PHẦN CỨNG

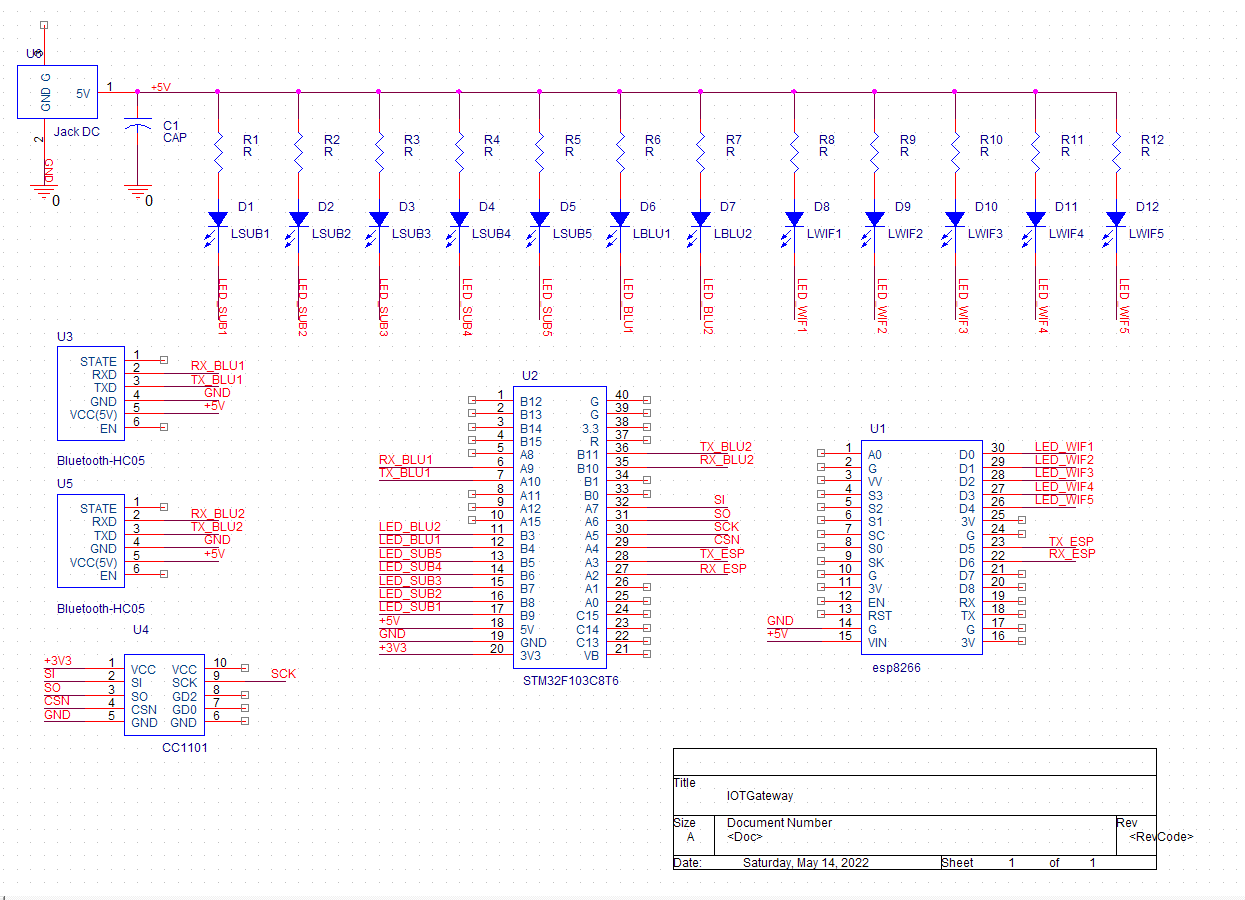
## Sơ đồ tổng quát hệ thống:



*Hình 3.1: Sơ đồ tổng quát hệ thống.*

* **IOT Gateway có thể kết nối với tối đa với:**
* 5 thiết bị có kết nối Wifi thông qua mạng Mesh Wifi.
* 5 thiết bị sử dụng giao thức Sub GHz ở tần số 433Mhz.
* 2 thiết bị Sử dụng giao thức Bluetooth 2.4GHz.
* **Hệ thống bao gồm:**
* Vi điều khiển STM32F103C8T6:
* Nhận dữ liệu từ các module HC05 và CC1101
* Xử lý dữ liệu và gửi cho Module Wifi ESP8266 qua giao tiếp UART.
* Nhận tín hiệu điều khiển được gửi từ ESP8266.
* Xử lý và phân luồng cho tín hiệu điều khiển.
* Kiểm tra kết nối với các thiết bị SubGHz và Bluetooth.
* Module Wifi ESP8266:
* Nhận dữ liệu được gửi từ vi điều khiển STM32F103C8T6 các thiết bị Wifi.
* Xử lý dữ liệu và ghi vào database thông qua Internet.
* Nhận tín hiệu điều khiển từ ứng dụng Androi.
* Xử lý và phân luồng cho tín hiệu điều khiển.
* Kiểm tra kết nối với các thiết bị Wifi.
* Module Sub GHz CC1101:
* Truyền nhận dữ liệu với các thiết bị Sub GHz khác.
* CC1101 giao tiếp với vi điều khiển STM32F103C8T6 qua giao tiếp SPI.
* Module Bluetooth Sub GHz:
* Truyền nhận dữ liệu với thiết bị Bluetooth.
* HC05 giao tiếp với vi điều khiển STM32F103C8T6 qua giao tiếp UART.

## Sơ đồ mạch:



*Hình 3.2: Sơ đồ mạch.*

* **Danh sách linh kiện sử dụng:**

|  |  |
| --- | --- |
| Tên | Số lượng |
| STM32F103C8T6 | 1 |
| ESP8266 | 2 |
| HC05 | 2 |
| CC1101 | 1 |
| Led RGB | 12 |
| Điện trở 330Ohm | 12 |
| Tụ điện 1000uF, 16v | 1 |
| Jack DC | 1 |
| Adapter 5V,2A | 1 |

* Cả ESP8266 và HC05 đều giao tiếp với STM32F103C8T6 thông qua giao tiếp UART. Trong đó:
* HC05- Cổng 1 kết nối với UART1 của STM32F103C8T6.
* HC05- Cổng 2 kết nối với UART3 của STM32F103C8T6.
* ESP8266 (D2-RX và D3-TX) kết nối với UART2 của STM32F103C8T6.
* STM32F103C8T6 giao tiếp với CC1101 qua SPI.
* **STM32F103C8T6 - HC-05:**
* **Module 1:**

|  |  |
| --- | --- |
| **HC 05 - 1** | **STM32F103C8T6** |
| VCC | 5V |
| GND | GND |
| RXD | PA9 (UART1-Tx) |
| TXD | PA10 (UART1-Rx) |

* **Module 2:**

|  |  |
| --- | --- |
| **HC 05 - 2** | **STM32F103C8T6** |
| VCC | 5V |
| GND | GND |
| RXD | PB10 (UART3-Tx) |
| TXD | PB11 (UART3-Rx) |

* **STM32F103C8T6 - ESP8266:**

|  |  |
| --- | --- |
| **ESP8266** | **STM32F103C8T6** |
| VCC | 5V |
| GND | GND |
| D3 | PA3 (UART2-Tx) |
| D2 | PA2 (UART2-Rx) |

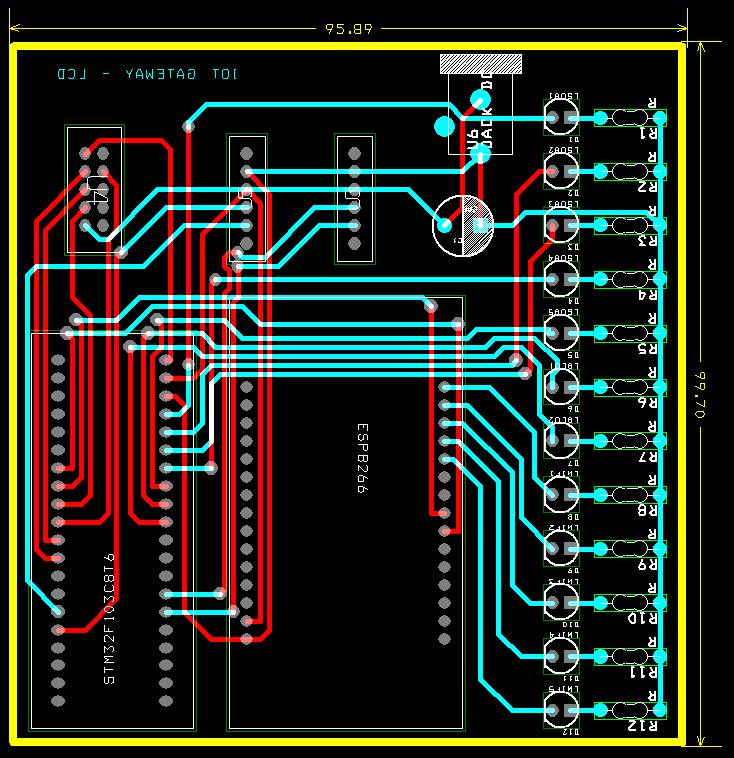
* **STM32F103C8T6 – CC1101:0**

|  |  |
| --- | --- |
| **CC1101** | **STM32F103C8T6** |
| VCC | 3V3 |
| GND | GND |
| CSN | PA4 |
| SCK | PA5 |
| MISO | PA6 |
| MOSI | PA7 |

* Nguồn cung cấp cho hệ thống được lấy từ Adapter 5V – 2A thông qua Jack DC.
* Các Led có tác dụng chỉ báo cho kết nối của thiết bị vào IOT gateway. Trong đó:
* Led D1 - Led D5 chỉ báo cho kết nối của thiết bị Sub GHz.
* Led D6, Led D7 chỉ báo cho kết nối của thiết bị Bluetooth.
* Led D8 - Led D12 chỉ báo cho kết nối của thiết bị Wifi.

## Thiết kế PCP:

### Thiết kế PCB bằng phần mềm Orcad:

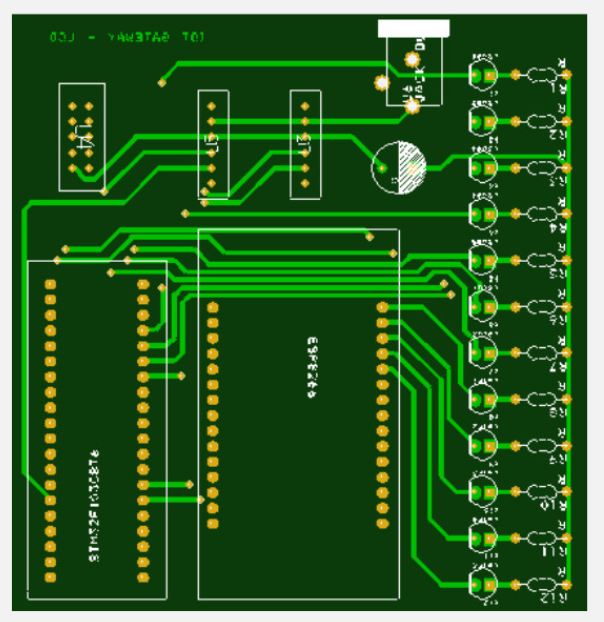


*Hình 3.3: Layout PCB mạch IOT Gateway.*

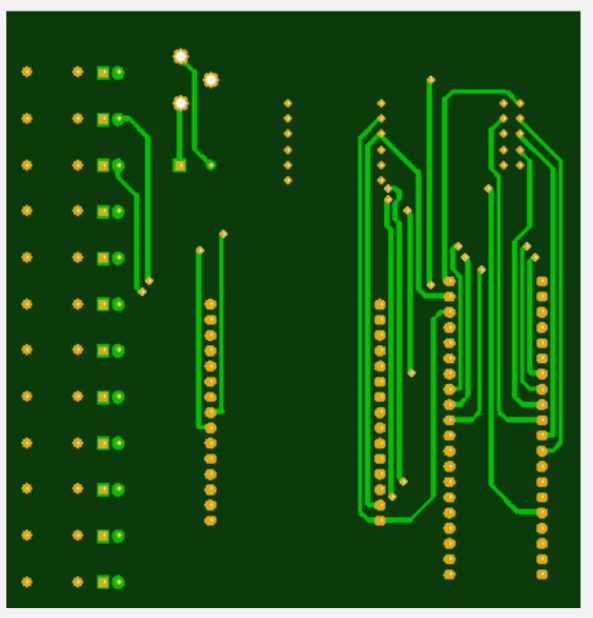
PCB được thiết kế với 2 lớp TOP và BOTTOM.

Kích thước: 95.89 x 99.70 mm.

### Kết quả layout:



*Hình 3.4: Kết quả layout PCB lớp TOP.*



*Hình 3.5: Kết quả layout PCB lớp BOTTOM.*

# THIẾT KẾ VÀ THỰC HIỆN PHẦN MỀM

## Ý tưởng:

### Nhận dữ liệu các thiết bị:

#### Dữ liệu từ các thiết bị Sub-GHz và Blutooth:

Việc nhận dữ liệu từ các thiết bị Sub-GHz và Bluetooth được thực hiện bởi vi điều khiển STM32F103C8T6 (tạm gọi là STM32F1).

* *Bước 1:* STM32F1 gửi lệnh yêu cầu dữ liệu và chờ thiết bị phản hồi.( Cứ mỗi 60s, STM32F1 sẽ gửi yêu cầu một lần.)
* *Bước 2:*
* Nếu nhận được dữ liệu: STM32F1 xử lý dữ liệu nhận được và gửi cho ESP8266 thông qua UART và cuối cùng được lưu trữ trong Database.
* Nếu không nhận được dữ liệu: quay lại bước 1.

#### Dữ liệu từ các thiết bị Wifi:

Việc nhận dữ liệu từ các thiết bị Wifi cũng giống như việc nhận dữ liệu của các thiết bị Sub-GHz nhưng được thực hiện trên chính Module ESP8266.

*Bước 1:* ESP8266 gửi lệnh yêu cầu dữ liệu và chờ thiết bị phản hồi.( Cứ mỗi 30s, ESP8266 sẽ gửi yêu cầu một lần.)

*Bước 2:*

* Nếu nhận được dữ liệu: ESP8266 xử lý dữ liệu nhận được và được lưu trữ trong Database.
* Nếu không nhận được dữ liệu: quay lại bước 1.

### Kiểm tra kết nối với thiết bị:

#### Các thiết bị Bluetooth và Sub-GHz:

Chỉ báo kết nối với các thiết bị Bluetooth và Sub-GHz được STM32F1 thực hiện dựa vào việc có nhận được dữ liệu từ thiết bị đó hay không. Nghĩa là khi nhận được dữ liệu từ một thiết bị, ta có thể nói thiết bị đó đã được kết nối với IOT Gateway.

Sau mỗi lần gửi yêu cầu dữ liệu đến thiết bị, STM32F1 sẽ chờ trong 1s. Nếu trong khoảng thời gian đó, thiết bị gửi dữ liệu phản hồi thì ta biết được thiết bị đó đã kết nối với IOT Gateway.

Sau khi phát hiện thấy sự thay đổi trong kết nối với các thiết bị Bluetooth và Sub-GHz , STM32F1 sẽ gửi các trạng thái kết nối mới qua ESP8266 và cuối cùng là cập nhật lên server qua Internet.

#### Các thiết bị Wifi:

Đối với các thiết bị Wifi kết nối thông qua mạng Mesh Wifi, ta có thể giám sát qua ChipID. Khi có sự thay đổi trong kết nối, chương trình sẽ tự động cập nhật list ChipID đang được kết nối với IOT Gateway. Khi có thay đổi trong kết nối với các thiết bị Wifi, ESP8266 sẽ cập nhật lên server qua Internet.

### Tín hiệu điều khiểu gửi từ ứng dụng Android:

Tín hiệu điều khiển được gửi từ ứng dụng Android thông qua server tới ESP8266. Khi nhận được tín hiệu điều khiển, ESP8266 sẽ kiểm tra đó là tín hiệu điều khiển thiết bị Wifi, Bluetooth hay Sub-GHz.

* Nếu là tín hiệu điều khiển thiết bị Wifi: ESP8266 trực tiếp gửi tới thiết bị Wifi đó.
* Nếu là tín hiệu điều khiển thiết bị Bluetooth hay Sub-GHz: ESP8266 sẽ gửi cho STM32F1 thông qua UART. Từ đây, STM32F1 sẽ kiểm tra và gửi cho thiết bị tương ứng.

Khi nhận được tín hiệu điều khiển, thiết bị sẽ thực hiện nó và cập dữ liệu lại cho IOT Gateway. Và cuối cùng dữ liệu lại được lưu trữ vào Database.

## Tạo cơ sở dữ liệu trên PHPMyAdmin:

### Cấu trúc cơ sở dữ liệu:

Tạo một tài khoản trên PHPMyAdmin, tạo một cơ sở dữ liệu có cấu trúc như sau:



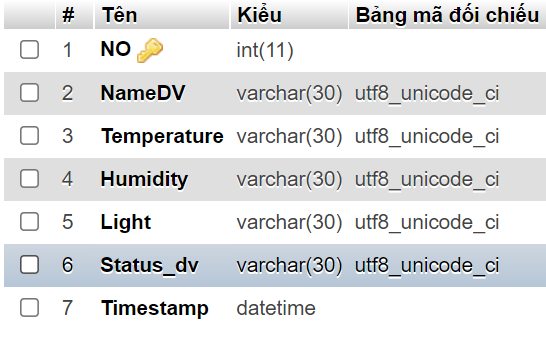
*Hình 4.1: Cấu trúc cơ sở dữ liệu.*

Cơ sở dữ liệu có tên là: [highalln\_iotgateway\_db](https://hkg108.hawkhost.com:2083/cpsess6910027392/3rdparty/phpMyAdmin/db_structure.php?server=1&db=highalln_iotgateway_db).

Bao gồm các table: BLU1, BLU2, SUB1, SUB2, SUB3, SUB4, SUB5, WIFI1, WIFI2, WIFI3, WIFI4, WIFI5. Mỗi bảng được dùng để lưu trữ cho một thiết bị tương ứng .

Mỗi table đều có các cột giống nhau:

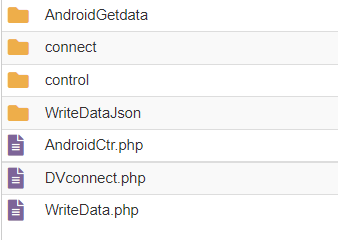
* NO: số thứ tự.
* NameDV: Tên thiết bị.
* Status\_dv: trạng thái thiết bị.
* Temperature: Giá trị nhiệt độ.
* Humidity: Giá trị độ ẩm.
* Light: Giá trị ánh sáng.
* timestamp: thời gian ghi dữ liệu(được lấy theo giờ của TP. Hồ Chí Minh).



*Hình 4.2: Cấu trúc bảng trong cơ sở dữ liệu.*

### Mã php:

Để tương tác với Database, ta sử dụng các file PHP được lưu trong server như sau:



*Hình 4.3: File PHP trong server.*

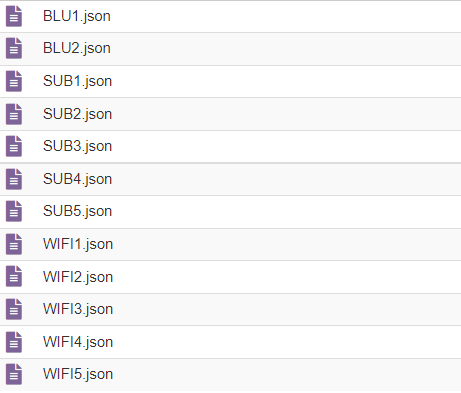
#### Chương trình ghi dữ liệu vào Database – *WriteData.php*:

Khi cần ghi dữ liệu vào Database, ESP8266 sẽ truy cập vào file *“WriteData.php”* bằng giao thức HTTP theo đường link:

[*https://luanvanlogistic.highallnight.com/AppIoTgateway/WriteData.php*](https://luanvanlogistic.highallnight.com/AppIoTgateway/WriteData.php)

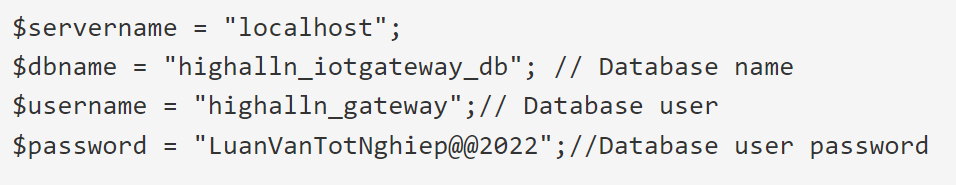
File này có nhiệm vụ:

* Truy cập Database.
* Nhận dữ liệu được gửi từ ESP8266.
* Ghi dữ liệu đó vào Database theo đúng bảng tương ứng với thiết bị.
* Tạo ra một JSON chứa dữ liệu trong file .*json* tương ứng với thiết bị trong thư mục *“*[*WriteDataJson*](#WriteDataJson_folder)*”.*



*Hình 4.4: File .JSON trong thư mục WriteDataJson.*

Đầu tiên chương trình đăng nhập vào database theo thông tin đăng nhập như sau:



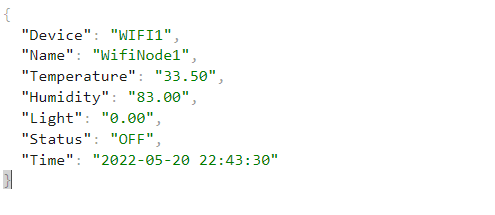
Bằng các truy cập đúng tên bảng và tên cột, ta có thể ghi dữ liệu vào database đúng theo từng loại dữ liệu.



Trong đó:

* *NameDV, Status\_dv, Temperature, Humidity, Light, Timestamp:* là tên của các cột.
* *$Name, $Stt, $ND, $DA, $AS, $timest:* là tên các biến lưu dữ liệu tương ứng.

Sau khi ghi dữ liệu vào Database thành công, chương trình tiếp tục tạo ra 1 JSON trong file *<tên thiết bị>.json* tương ứng với thiết bị trong thư mục *“WriteDataJson”.* Dữ liệu trong JSON này có dạng:



*Hình 4.5: JSON trả về trong file WIFI1.json*

#### Chương trình tạo tín hiệu điều khiển từ Android – *AndroidCtr.php*:

Khi người dùng nhấn lệnh điều khiển từ ứng dụng Android, chương trình trên ứng dụng truy cập vào file *“AndroidCtr.php”* bằng giao thức HTTP theo đường link:

[*https://luanvanlogistic.highallnight.com/AppIoTgateway/AndroidCtr.php*](https://luanvanlogistic.highallnight.com/AppIoTgateway/AndroidCtr.php)

File này có nhiệu vụ:

* Nhận dữ liệu của điều khiển được gửi từ ứng dụng.
* Tạo ra 1 JSON chứa dữ liệu điều khiển vào file *“control.json”* trong thư mục *“*[*control*](#control_folder)*”.*



*Hình 4.6 : File .JSON trong thư mục control.*

Ví dụ: Người dùng muốn bật thiết bị Wifi 1, JSON được tạo ra trong file *“control.json”* có dạng:



*Hình 4.7: JSON trả về trong file control.json*

#### Chương trình POST trạng thái kết nối của thiết bị - *DVconnect.php*:

Khi có sự thay đổi trong trạng thái kết nối với các thiết bị trong hệ thống, ESP8266 sẽ truy cập vào file *“DVconnect.php”* theo đường link:

[*https://luanvanlogistic.highallnight.com/AppIoTgateway/DVconnect.php*](https://luanvanlogistic.highallnight.com/AppIoTgateway/DVconnect.php)

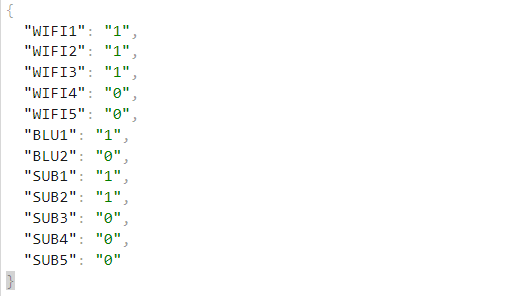
File này có nhiệm vụ:

* Nhận dữ liệu trạng thái kết nối với thiết bị trong hệ thống được gửi từ ESP8266.
* Tạo ra 1 JSON chứa trạng thái kết nối với các thiết bị trong hệ thống vào file *“DVconnect.json”* được lưu trong thư mục “[*connect*](#connect_folder)*”.*



*Hình 4.8: File .JSON trong thư mục connect.*

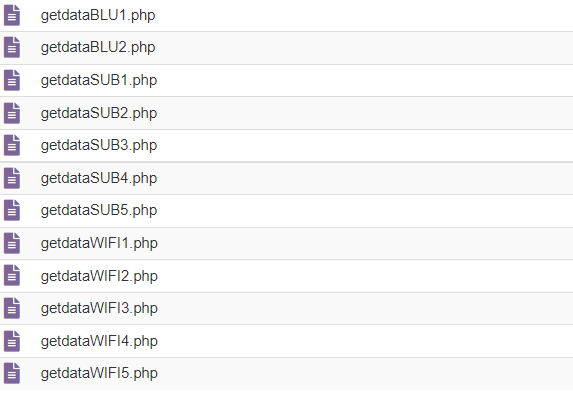
JSON được tạo ra trong file *“DVconnect.json”* có dạng:



*Hình 4.9: JSON trả về trong file DVconnect.json*

#### Chương trình đọc dữ liệu từ Database – *getdata<tên thiết bị>.php*:

Các file này được lưu trong thư mục *“*[*AndroidGetdata*](#AndroidGetdata_folder)*”*.



*Hình 4.10: File PHP trong thư mục AndroidGetdata.*

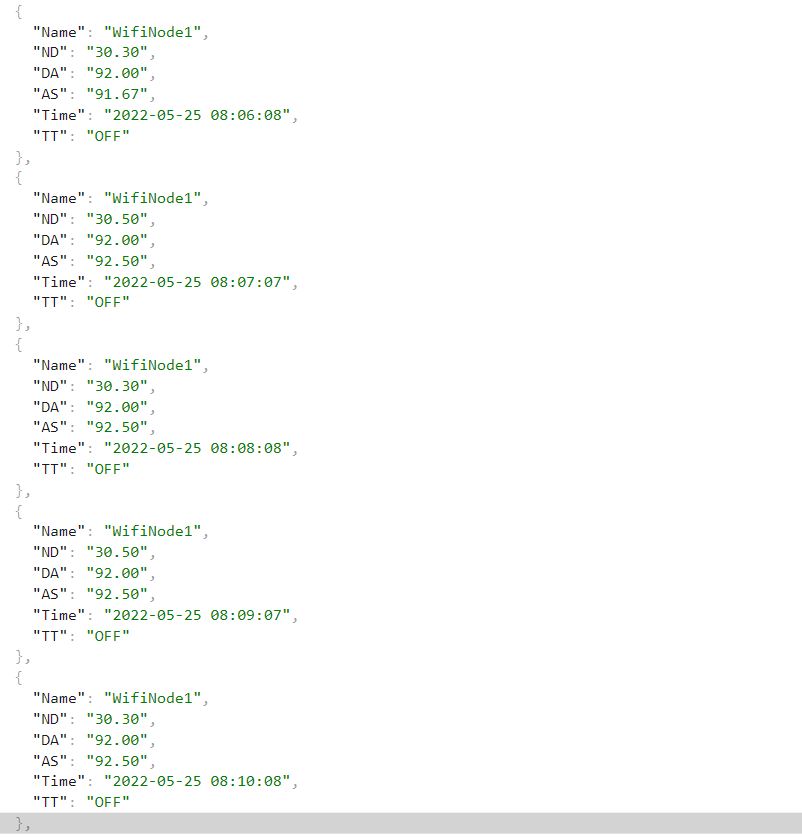
Khi ứng dụng Android cần đọc dữ liệu của thiết bị được lưu trong Database, chương trình trên ứng dụng sẽ truy cập vào file tương ứng. Nhiệm vụ của các file này:

* Truy cập vào bảng tương ứng trong Database.
* Đọc tất cả dữ liệu của bảng.
* Trả về 1 JSONArray với mỗi JSONObject chứa dữ liệu của 1 hàng.

Ví dụ: Khi người dùng muốn xem dữ liệu của thiết bị Wifi 1, chương trình trên ứng dụng sẽ truy cập vào file *“getdataWIFI1.php”* theo đường link:

[*http://luanvanlogistic.highallnight.com/AppIoTgateway/AndroidGetdata/getdataWIFI1.php*](http://luanvanlogistic.highallnight.com/AppIoTgateway/AndroidGetdata/getdataWIFI1.php)

Khi chạy chương trình trong file *getdataWIFI1.php”* ta thu được kết quả:

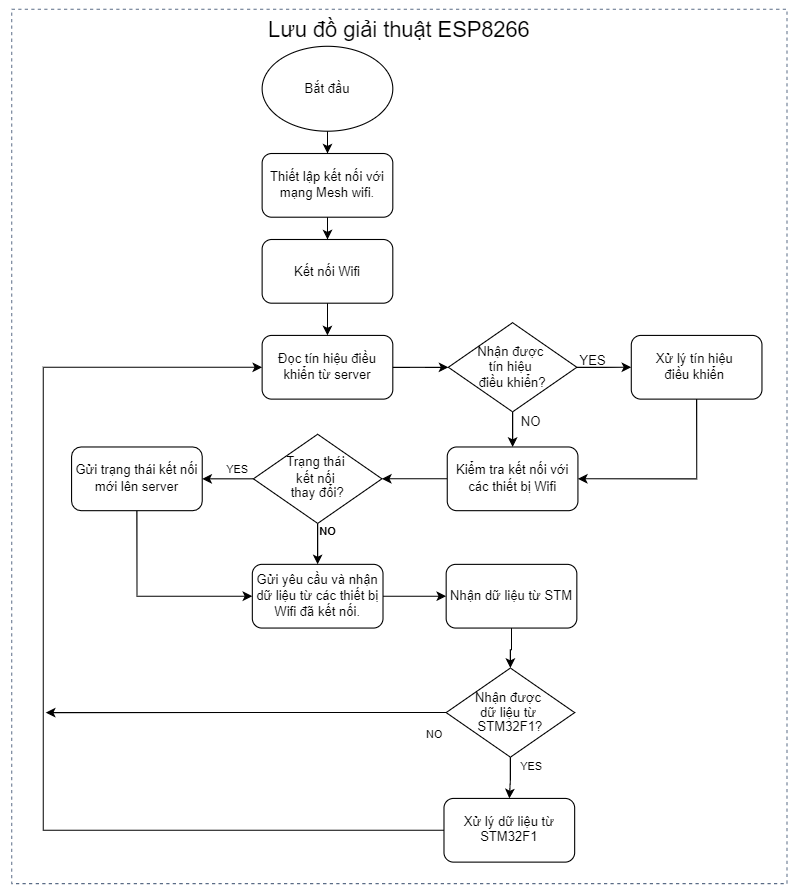


*Dữ liệu trả về của file getdataWIFI1.php*

## Lưu đồ giải thuật:

### ESP8266:

1. **Lưu đồ giải thuật tổng quát:**



*Hình 4.11: Lưu đồ giải thuật tổng quát của ESP8266.*

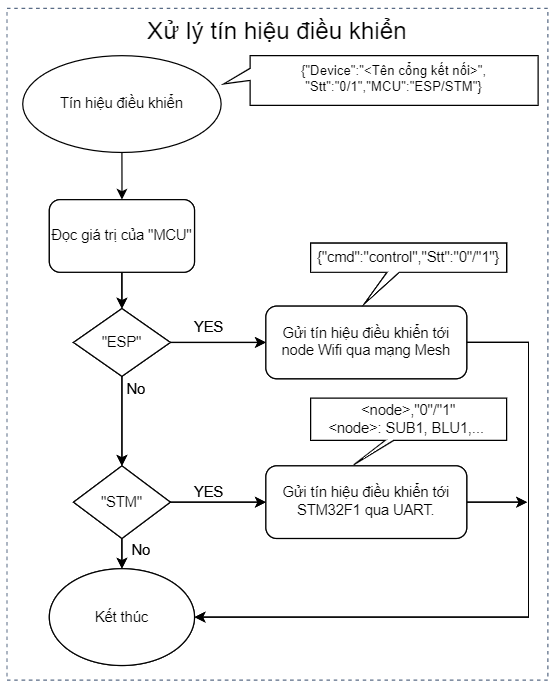
1. **Chương trình đọc tín hiệu điều khiển từ Server:**

Để đọc tín hiệu điều khiển được gửi từ ứng dụng Android, chương trình trên ESP8266 truy cập vào file *“control.json”* được lưu trong thư mục *“*[*control*](#control_folder)*”* theo đường link sau: [*https://luanvanlogistic.highallnight.com/AppIoTgateway/control/control.json*](https://luanvanlogistic.highallnight.com/AppIoTgateway/control/control.json)

Sau khi truy cập vào file *“control.json”* chương trình tiến hành đọc nội dung của nó. Kết quả trả về có dạng: *{"Device":"WIFI1","Stt":"1","MCU":"ESP"}.* Trong đó:

* *"Device"*: là thiết bị được điều khiển.
* *"Stt"*: là trạng thái mong muốn.
* *"MCU"*: là vi điều khiển thực hiện gửi lệnh điều khiển.

#### Chương trình xử lý tín hiệu điều khiển:



*Hình 4.12: Lưu đồ giải thuật xử lý tín hiệu điều khiển.*

Sau khi nhận được tín hiệu điều khiển từ Server, ESP8266 kiểm tra giá trị của key *“MCU”.* Nếu giá trị đó là *“ESP”* có nghĩa là thiết bị đang được điều khiển là Wifi, ESP8266 sẽ trực tiếp gửi lệnh điều khiển tới thiết bị Wifi đó. Nếu giá trị của *“MCU”* là *“STM”* thì thiết bị được điều khiển là Bluetooth hoặc Sub-GHz, ESP8266 sẽ gửi tín hiệu điều khiển tới STM32F1 thông qua UART để tiếp tục xử lý. Dựa vào giá trị của *“MCU”* ta có thể phân luồn cho tín hiệu điều khiển một cách dễ dàng.

Đồi với tín hiệu điều khiển các thiết bị Wifi: ESP8266 trực tiếp gửi tín hiệu điều khiển tới thiết bị thông qua mạng Mesh Wifi. Lệnh *mesh.sendSingle(<ChipID>,msg)* được sử dụng để gửi tín hiệu điều khiển cho thiết bị. Trong đó:

* *ChipID*: địa chỉ ID của thiết bị.
* *msg* được định dạng theo cấu trúc: *“{“cmd”:“control” ,”Stt”:”0/1”}”*

#### Chương trình kiểm tra kết nối vơi các thiết bị Wifi:

Mạng Mesh Wifi có chắc năng kiểm soát các thiết bị đang kết nối trong mạng theo ChipID của từng thiết bị. Khi có 1 thiết bị kết nối mới hoặc ngắt kết nối với mạng, chương trình sẽ tự động gọi hàm *changedConnectionCallback(),* hàm này có chức năng trả về 1 list chứa các ChipID của các thiết bị đang kết nối. Bằng cách kiểm tra các ChipID vừa được trả về so với các ChipID trước đó, ta có thể biết được thiết bị nào vừa kết nối hoặc ngắt kết nối với IOT Gateway.

Khi phát hiện có sự thay đổi này, chương trình sẽ gửi trạng thái kết nối mới của các thiết bị Wifi lên Server. Chương trình sẽ POST các giá trị trạng thái kết nối của thiết bị vào file *“DVconnect.php”* bằng giao thức HTTPtheo đường link:

[*https://luanvanlogistic.highallnight.com/AppIoTgateway/DVconnect.php*](https://luanvanlogistic.highallnight.com/AppIoTgateway/DVconnect.php)

File này có nhiệm vụ nhận các giá trị trạng thái kết nối của thiết bị (0 hoặc 1) được POST từ ESP8266, sau đó tạo ra 1 JSON chứa các trạng thái kết nối đó vào trong file *“DVconnect.json”*  được lưu trong thư mục *“connect”.* Dữ liệu có dạng như [Hình..](#DVconnect_json).

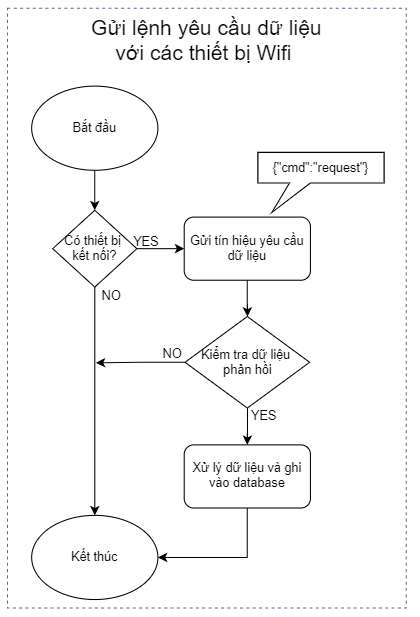
Ứng với mỗi *Key* là trạng thái kết nối của một thiết bị.

#### Chương trình yêu cầu dữ liệu từ các thiết bị Wifi:

Nếu IOT Gateway chưa kết nối với bất kỳ thiết bị Wifi nào thì bỏ qua bước này.

Mỗi một phút, ESP8266 sẽ gửi một lệnh request: *“{“cmd”:”request”}*  bằng hàm *mesh.sendBroadcast().* Hàm này sẽ gửi lệnh request tới toàn bộ các thiết bị Wifi đã kết nối trong mạng mesh.

Khi nhận được lệnh request, các thiết bị sẽ gửi dữ liệu phản hồi về IOT Gateway. Dữ liệu này bao gồm: Tên thiết bị, giá trị của nhiệt độ, độ ẩm, cường độ ánh sáng và trạng thái thiết bị.



*Hình 4.13: Flowchart chương trình yêu cầu dữ liệu từ các thiết bị Wifi.*

Khi đã nhận được dữ liệu từ các thiết bị Wifi, chương trình sẽ so sánh với dữ liệu nhận được trước đó của thiết bị. Nếu nhận thấy có sự khác biệt giữa dữ liệu cũ và dữ liệu mới, chương trình sẽ ghi dữ liệu đó vào Database (như đã trình bày ở [4.2.2.a](#_Chương_trình_ghi)).

#### Chương trình xử lý dữ liệu từ STM32F103C8T6:

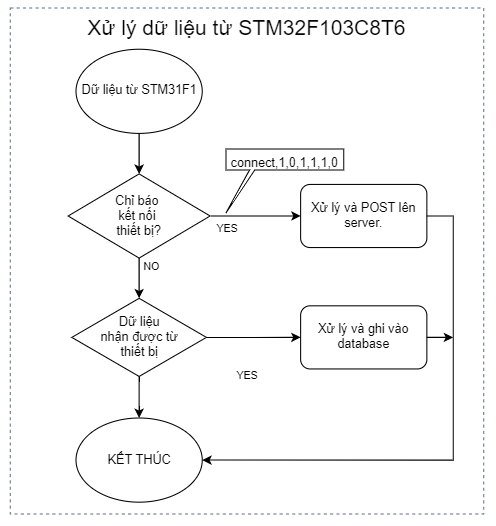
Dữ liệu được gửi từ FTM32F1 có 2 dạng:

* Trạng thái kết nối với các thiết bị Bluetooth và SubGHz.

VD: *“connect,1,1,1,0,0,1,0”*

* Dữ liệu nhận được từ thiết bị Bluetooth hoặc SubGHz.

VD: *“SUB1,SubNode1,32.60,72.00,0.00,OFF”*



*Hình 4.14: Lưu đồ giải thuật xử lý dữ liệu từ STM32F103C8T6.*

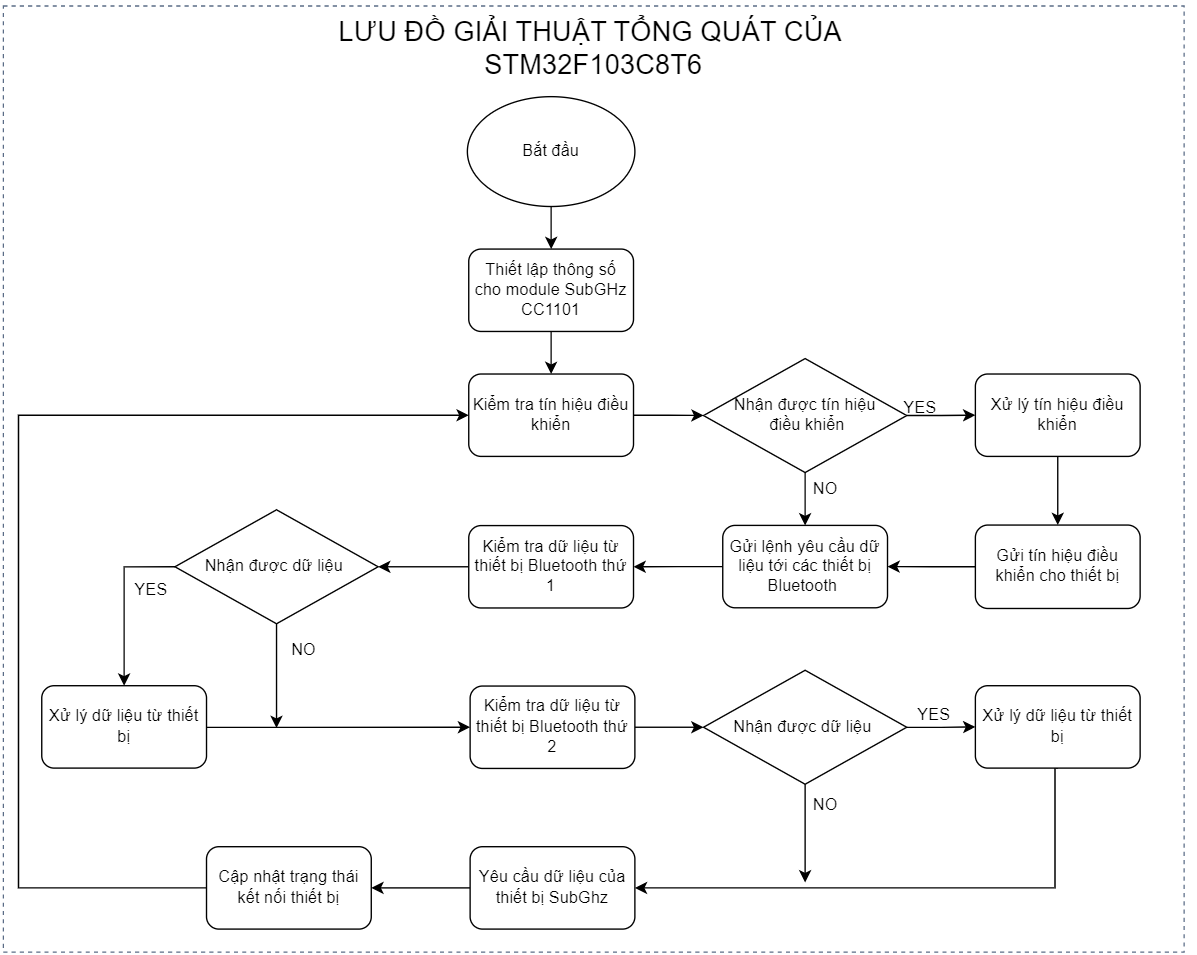
Để phân biệt giữa dữ liệu từ thiết bị và trạng thái kết nối, ta kiểm tra đoạn text đầu tiền trước dấu phẩy *(,)*. Nếu là *“connect”* thì đó là dữ liệu chứa trạng thái kết nối với các thiết bị Bluetooth và Sub-GHz. Ngược lại, nếu khác *“connect”*, thì đó là dữ liệu được gửi từ thiết bị.

Đối với dữ liệu là trạng thái kết nối với các thiết bị, ESP8266 sẽ xử lý để tách các giá trị trạng thái kết nối, sau đó POST các trạng thái đó lên Server giống như các trạng thái kết nối với thiết bị Wifi.

Đối với dữ liệu từ các thiết bị, ESP8266 sẽ xử lý để lấy ra tên thiết bị, các giá trị của nhiệt độ, độ ẩm, cường độ ánh sáng và trạng thái của thiết bị. Sau đó, dữ liệu sẽ được ghi vào Database giống như với dữ liệu từ thiết bị Wifi.

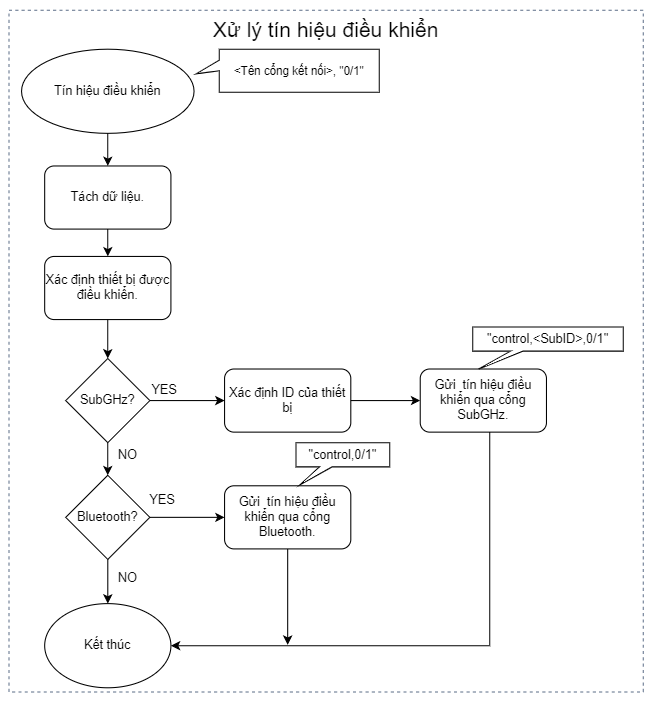
### STM32F103C8T6:

#### Lưu đồ giải thuật tổng quát:

****

*Hình 4.15: Lưu đồ giải thuật tổng quát của STM32F103C8T6.*

#### Chương trình xử lý tín hiệu điều khiển:



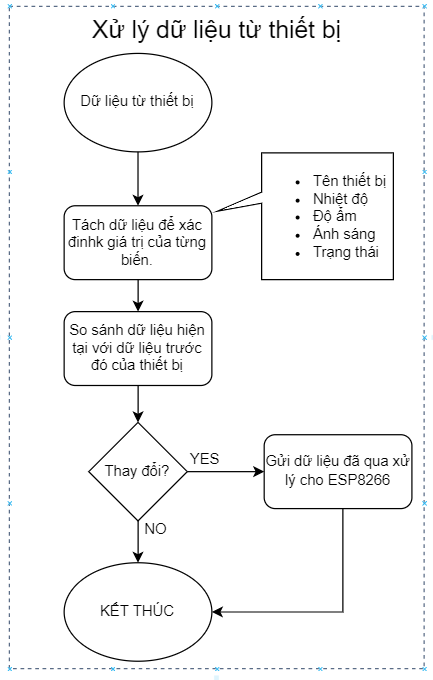
*Hình 4.16: Lưu đồ giải thuật xử lý tín hiệu điều khiển trên STM32F103C8T6.*

Khi nhận được tín hiệu điều khiển được gửi từ ESP8266, STM32F1 sẽ tiến hành tách dữ liệu, từ đó xác định được thiết bị được điều khiển và trạng thái điều khiển. Sau đó, chương trình sẽ gửi tín hiệu điều khiển tới thiết bị.

Nếu thiết bị là Bluetooth 1 hoặc Bluetooth 2, tín hiệu điều khiển sẽ được gửi thông qua module HC05 số 1 hoặc số 2 tương ứng với thiết bị đó. Dữ liệu được gửi đi được định dạng theo cấu trúc: *“control,0/1”.*

Nếu thiết bị là SubGHz, chương trình sẽ gửi tín hiệu điều khiển qua module CC1101. Dữ liệu gửi đi được định dạng theo cấu trúc: *“control,<SubID>,0/1”*.

#### Chương trình xử lý dữ liệu từ thiết bị trên STM32F103C8T6:

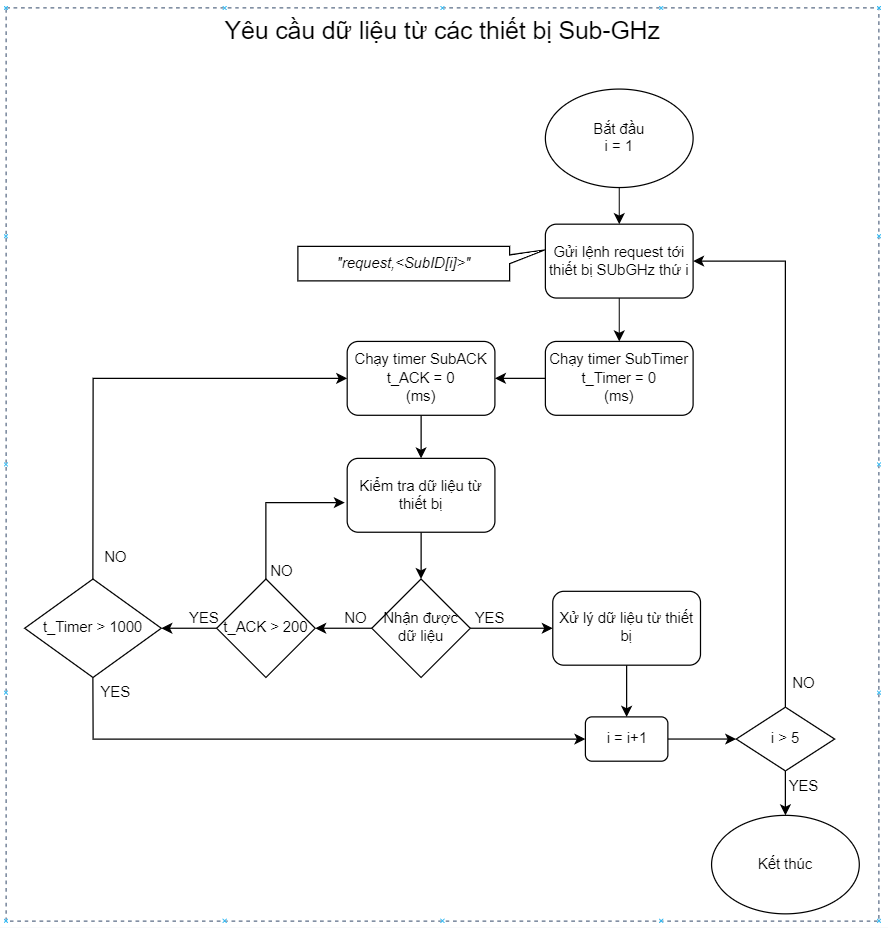


*Hình 4.17: Lưu đồ giải thuật xử lý dữ liệu từ thiết bị trên STM32F103C8T6.*

Khi nhận được dữ liệu từ thiết bị, STM32F1 tiến hành xử lý dữ liệu để lấy ra các giá trị của *tên thiết bị, nhiệt độ, độ ẩm, cường độ ánh sáng* và  *trạng thái của thiết bị.* Khi đã lấy được các dữ liệu đó, chương trình sẽ so sánh với dữ liệu nhận được trước đó của thiết bị. Nếu nhận thấy có sự khác biệt giữa dữ liệu cũ và dữ liệu mới, chương trình sẽ gửi dữ liệu qua ESP8266 theo cấu trúc: *“<tên cổng>,<tên thiết bị>,<giá trị nhiệt độ>,<giá trị độ ẩm,<cường độ ánh sáng>,<trạng thái>”.* Cuối cùng, dữ liệu cũ của thiết bị được thay bằng dữ liệu mới.

#### Chương trình yêu cầu dữ liệu từ các thiết bị SubGHz:

Hệ thống được thiết kế để nhận dữ liệu của tối đa 5 thiết bị Sub-GHz. Để phân biệt giữa các thiết bị Sub-GHz với nhau, ta phải gắn ID cho từng thiết bị.



*Hình 4.18: Lưu đồ giải thuật yêu cầu dưc liệu từ các thiết bị Sub-GHz.*

Để yêu cầu dữ liệu từ các thiết bị Sub-GHz, chương trình sẽ lần lược gửi lệnh request với cấu trúc: “request,<SubID[i]>”, với SubID[i] là ID của thiết bị thứ i. Nhờ vào ID này, các thiết bị có thể biết được IOT Gateway đang tương tác với nó.

Khi gửi lệnh request, chương trình sẽ chạy đồng thời 2 timer:

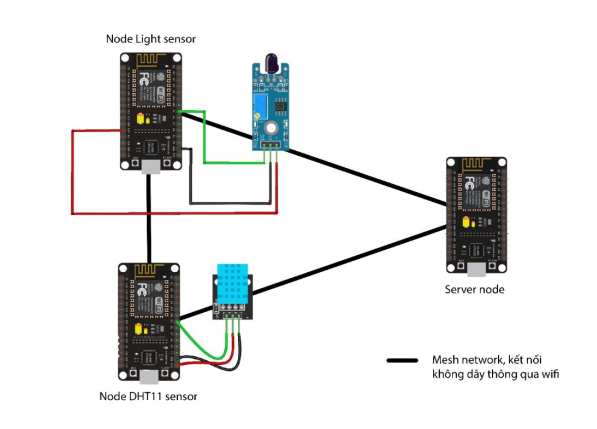
* SubACK: sau khi gửi lệnh request đầu tiên, nếu như k nhận được dữ liệu phản hồi của thiết bị, mỗi 200ms chương trình sẽ gửi lại lệnh request.
* SubTimer: khi Timer này lớn hơn 1000ms mà vẫn chưa nhận được dữ liệu phản hồi từ thiết bị, có thể kết luận là thiết bị Sub-GHz đó chưa được kết nối với IOT Gateway. Chương trình sẽ gửi lệnh request tới thiết bị Sub-GHz tiếp theo.

Qua chương trình con này ta cũng có thể kiểm tra trạng thái kết nối giữa IOT Gateway với các thiết bị Sub-GHz.

## Cấu hình hệ thống:

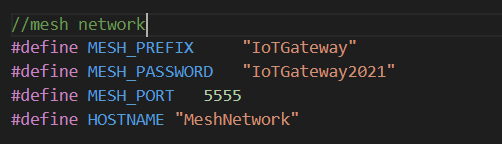
### Mạng Mesh Wifi:

Cổng Wifi hoạt động dựa trên hệ thống mạng mesh. Leafnode sẽ gửi dữ liệu thu thập từ cảm biến đến Server node thông qua cơ chế mạng mesh, được tạo nên nhờ sử dụng thư viện PainlessMesh của arduino.



*Hình 4.19: Kết nối mạng mesh các nodeMCU ESP8266.*

Tạo một mạng mesh có tên là *“IoTGateway”* có mật khẩu là *“IoTGateway2021”.*

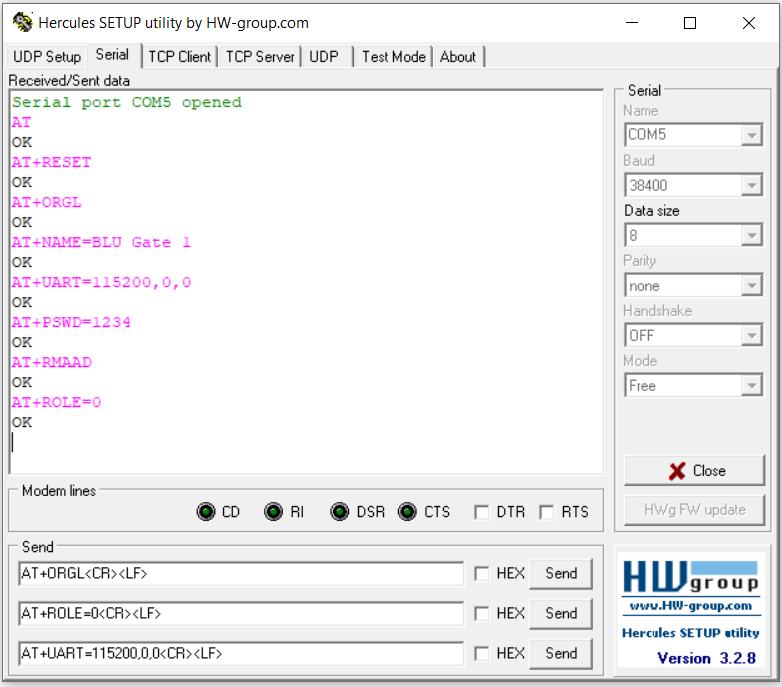


Các node trong hệ thống sau khi kết nối vào mạng mesh có thể gửi dữ liệu về server theo dạng chuỗi Json.



### Cấu hình cho module Bluetooth:

* **Cấu hình cho module Bluetooth HC05:** Sử dụng phần mềm Hercules và tập lệnh AT(AT command) để cấu hình cho module HC05.
* AT Kiểm tra module có ở chế độ command không?
* AT+RESET Reset thiết bị
* AT+ORGL Đặt lại cài đặt mặc định
* AT+NAME=BLU Gate 1 Đặt tên cho module là “BLU Gate”
* AT+UART=115200,0,0 Thiết lập baudrate 115200,0 bit stop, 0 parity
* AT+PSWD=1234 Đặt mật khẩu cho module là “1234”
* AT+RMAAD Xóa mọi thiết bị đã ghép nối
* AT+ROLE=0 Đặt là chế độ SLAVE
* **Kết quả thực hiện:**



*Hình 4.20: Kết quả cấu hình cho module Bluetooth HC05 – Cổng 1.*

* Làm tượng tự cho cổng 2.

## Ứng dụng IoTGateway:

### Tổng quan:

Ứng dụng được viết trên phần mềm Android Studio. Ứng dụng hổ trợ cho các dòng điện thoại sử dụng hệ điều hành Android từ 7.0 trở lên.

Ứng dụng có thể kết nối Internet, và đọc dữ liệu từ Database được lưu trữ trong phần mềm phpMyAdmin. Dữ liệu được xử lý và phục vụ cho các chức năng khác nhau.

### Chức năng:

* Xem trạng thái kết nối với các thiết bị.
* Xem dữ liệu hiện tại của thiết bị.
* Xem lịch sử dữ liệu.
* Hiển thị dữ liệu theo dạng đồ thị.
* Tạo tín hiệu điều khiển thiết bị (ON/OFF).

### Giao diện:

#### Giao diện chính:



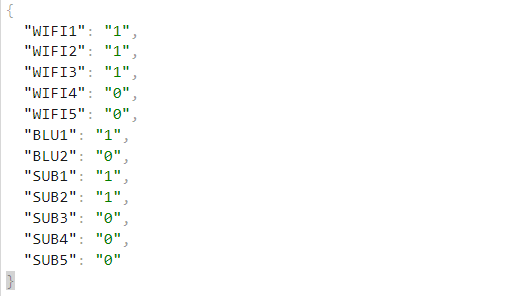
*Hình 4.21: Giao diện chính - ứng dụng IoTGateway.*

* Giao diện chính hiển thị số lượng thiết bị đã kết nối vào IOT Gateway.
* Nút nhấn sẽ chuyển sang màu xanh khi thiết bị tương ứng kết nối với IOT Gateway.
* **Hoạt động:**

Chương trình truy cập vào file *“DVconnect.json”* theo đường dẫn:

[*http://luanvanlogistic.highallnight.com/AppIoTgateway/connect/DVconnect.json*](http://luanvanlogistic.highallnight.com/AppIoTgateway/connect/DVconnect.json)

Kết quả trả về có dạng như [Hình 4......](#DVconnect_json)



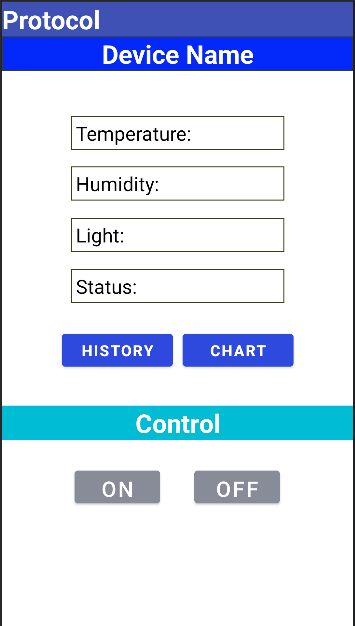
Từ kết quả trên, chương trình xác thể xác định được chính xác thiết bị nào đang kết nối và tổng số các thiết bị đang kết nối với IOT Gateway bằng cách kiểm tra các giá trị bằng 1.

Sau khi xác định, chương trình sẽ hiển thị số lượng thiết bị đang kết nối lên khung *“Connected devices:”* và enable các nút nhấn tương ứng với thiết bị đang kết nối (Nút nhấn chuyển qua màu xanh và có thể hoạt động).

Nhấn vào nút tương ứng để chuyển sang giao hiện hiển thị dữ liệu của thiết bị đó.

#### Giao diện hiển thị:

* Giao diện hiển thị dữ liệu của thiết bị bao gồm nhiệt độ, độ ẩm, ánh sáng và trạng thái của thiết bị(ON/OFF)
* Hai nút nhấn *HISTORY* và *CHART* để mở lịch sử và biểu đồ của dữ liệu.
* Trong phần Control có 2 nút nhấn *ON* và *OFF* được dùng để tạo tín hiệu điều khiển thiết bị qua Internet.



*Hình 4.22: Giao diện hiển thị dữ liệu - ứng dụng IoTGateway.*

* **Hoạt động:**
* ***Hiển thị dữ liệu:***

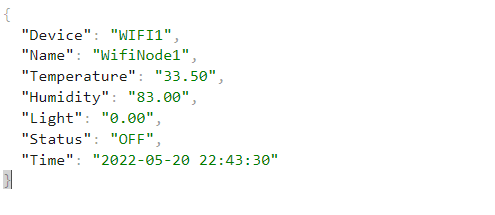
Chương trình truy cập vào file đọc dữ liệu tương ứng với thiết bị muốn xem dữ liệu.

Ví dụ: để xem dữ liệu của thiết bị Wifi thứ nhất, chương trình sẽ đọc kết quả của file *“WIFI1.json”* được lưu trong thư mục *“WriteDataJson”*. Như đã giải thích ở phần [4.2.2.a](#WriteDataJson) file này sẽ được cập nhật một JSON mới chứa dữ liệu mới nhất của thiết bị. Bằng cách được JSON trong file này, ta có thể cập nhật được dữ liệu mới mất của thiết bị về ứng dụng Android.

Link truy cập file *“WIFI1.json”:*

[*http://luanvanlogistic.highallnight.com/AppIoTgateway/WriteDataJson/WIFI1.json*](http://luanvanlogistic.highallnight.com/AppIoTgateway/WriteDataJson/WIFI1.json)

Kết quả trả về khi truy cập file *“WIFI1.json”* có dạng như [Hình 4...](#WriteDataJson)



Chương trình sẽ xử lý JSON này để lấy ra các giá trị tên thiết bị, nhiệt độ, độ ẩm, cường độ ánh sáng và trạng thái thiết bị để hiển thị ra màn hình.

* ***Điều khiển:***

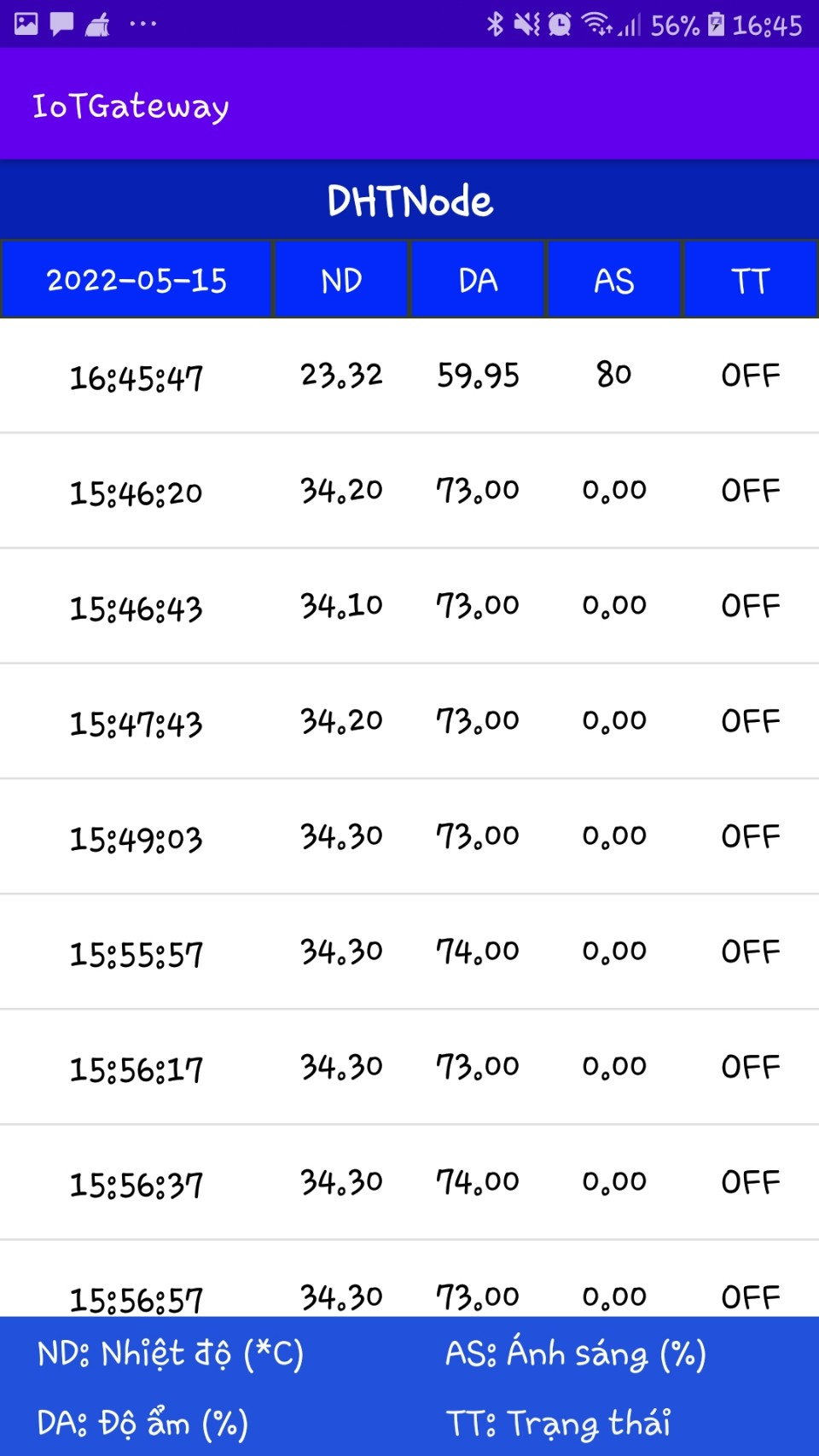
Khi nút ON/OFF được nhấn, chương trình sẽ truy cập vào file *“AndroidCtr.php”*theo link: <http://luanvanlogistic.highallnight.com/AppIoTgateway/AndroidCtr.php>

File *“AndroidCtr.php”*có nhiệm vụ tạo ra một JSON được lưu trong file “*control.json”* với nội dung: *{"Device":<"Tên">,"Stt":"0/1","MCU":"STM/ESP"}*

Trong đó:

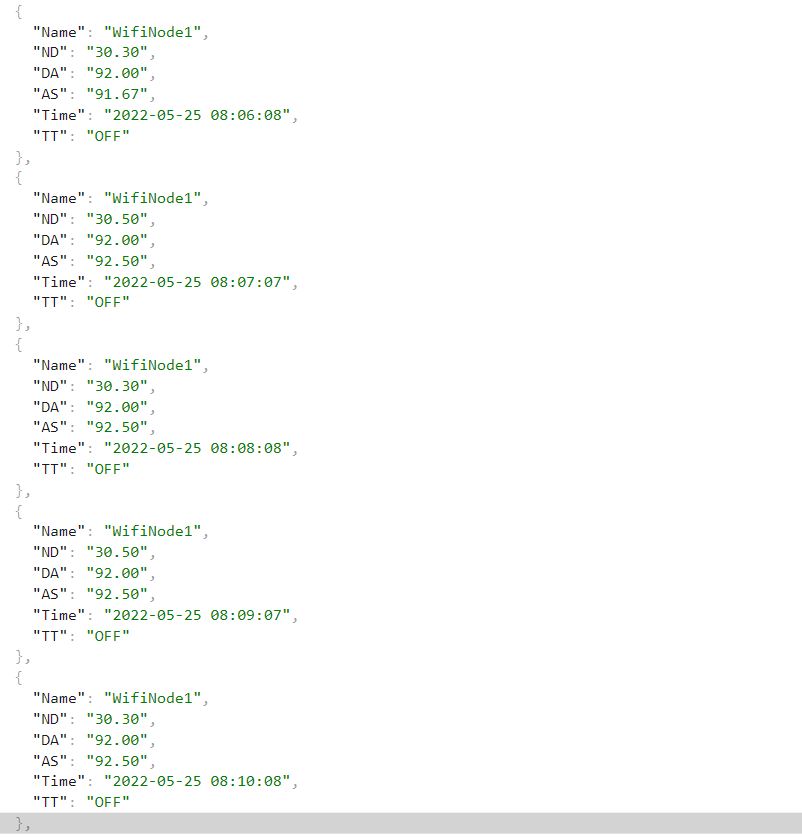
* Device: Tên của cổng đang chọn(WIF1, WIF2, WIF3, WIF4, WIF5, BLU1, BLU2, SUB1, SUB2, SUB3, SUB4, SUB5).
* Stt: trạng thái muốn điều khiển thiết bị(0 hoặc 1).
* MCU: có giá trị là STM hoặc ESP. Giúp IOT Gateway phân luồng tín hiệu điều khiển dễ dàng hơn.
* STM với tín hiệu điều khiển của thiết bị Bluetooth và Sub-GHz.
* ESP với tín hiệu điều khiển của thiết bị Wifi.

#### Giao diện lịch sử:



*Hình 4.23: Giao diện lịch sử.*

Để coi lịch sử dữ liệu của thiết, chương trình sẽ truy cập vào file getdata tương ứng. Như đã trình bày ở phần [4.2.2.d](#_Chương_trình_đọc), dữ liệu được lấy ra từ bảng tương ứng với thiết bị được chọn được lưu trong Database.



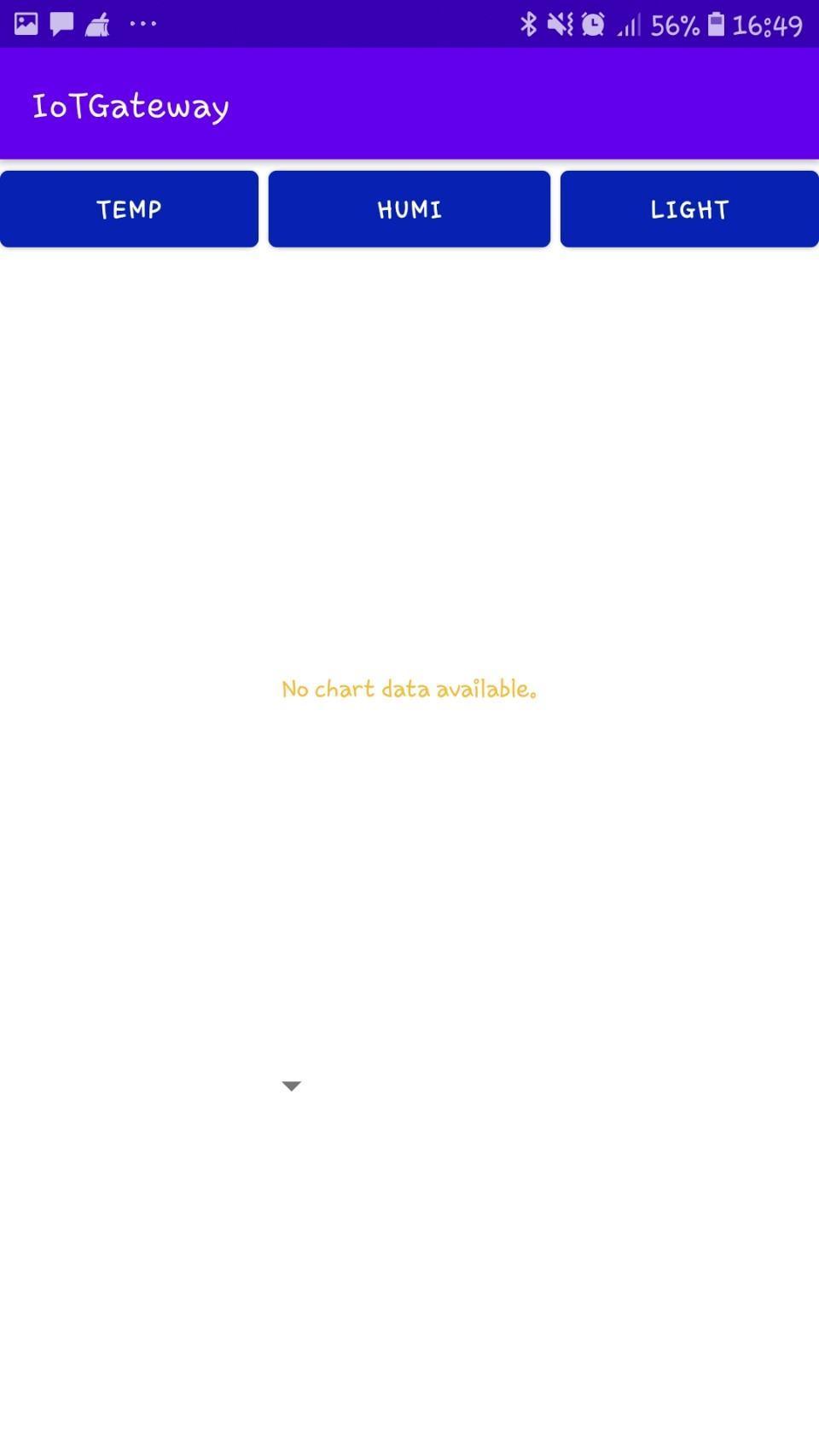
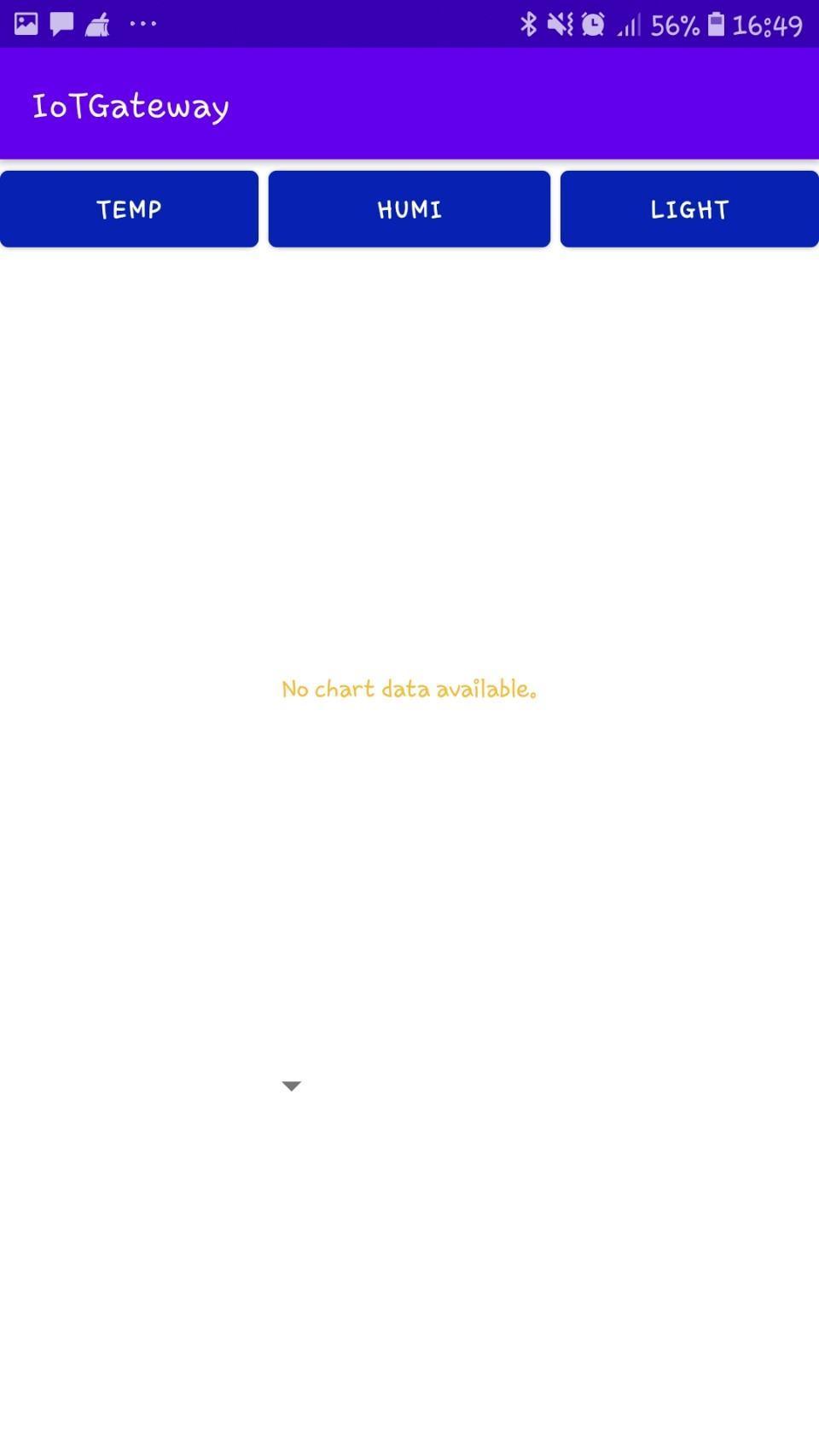
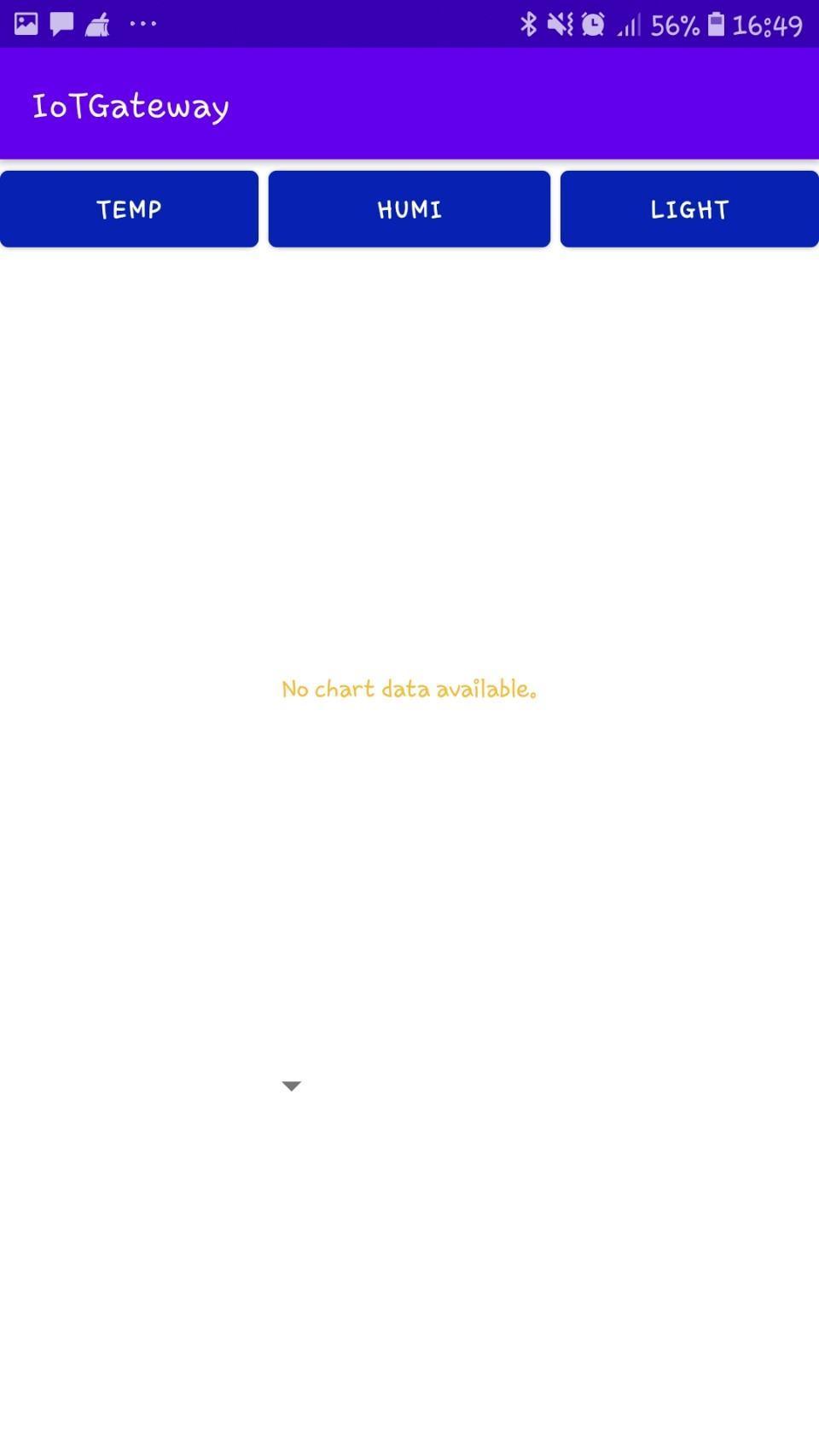
Dữ liệu được xử lý để hiển thị ra theo dạng ListView theo từng ngày. Theo mặc định, dữ liệu hiển thị lên là dữ liệu của ngày gần nhất. Tất cả các ngày có dữ liệu đều được liệt kê trong Spinner. Ta có thể chọn ngày khác để xem dữ liệu của ngày đó.

Theo lý thuyết, cứ 60s sẽ yêu cầu dữ liệu một lần, vậy trong ListView sẽ có khoảng 1140 dòng nên sẽ rất khó để có thể giám sát thiết bị. Vì vậy, ứng dụng cung cấp thêm giao diện Chart để có thể giám sát dữ liệu dễ dàng hơn.

#### Giao diện biểu đồ:

Giao diện này giúp cho người dùng có thể xem dữ liệu của thiết bị một cách trực quan hơn thông qua biểu đồ đường.

Biểu đồ đường được dựng lên theo từng loại dữ liệu riêng biệt: nhiệt độ, độ ẩm và cường độ ánh sáng. Bằng cách nhấn vào các nút ta có thể xem được các biểu đồ của dữ liệu tương ứng với nút đó: TEPM – biểu đồ nhiệt độ (mặc định), HUMI – biểu đồ độ ẩm, LIGHT – Biểu đồ cường độ ánh sáng.



Trục hoành của biểu đồ được biểu diễn theo thời gian của của 1 ngày từ 0-23h. Trục tung là giá trị của dữ liệu được chọn. Với mỗi loại dữ liệu khác nhau thì biên độ của trục tung cũng thay đổi theo: Nhiệt độ (0-50 \*C), Độ ẩm(0-100 %), cường độ ánh sáng(1-500 lx). Đường biểu diễn có màu sắc thay đổi theo loại dữ liệu: Nhiệt độ- màu đỏ, Độ ẩm- màu xanh dương, Cường độ ánh sáng- màu vàng.

Spinner liệt kê lất cả các ngày có dữ liệu trong Database. Dữ liệu được lấy theo ngày được chọn trong Spinner, mặc định là ngày gần nhất.

# KẾT QUẢ THỰC HIỆN