IOT CHALLENGE FPT SOFTWARE 2024

BÁO CÁO CUỘC THI

SMART GARBAGE MONITORING

NHÓM E401

Mentor: Nguyễn Duy Linh

Thành viên: Nguyễn Trung Kiên
Hà Ngọc Sơn
Nguyễn Đức Quốc
Võ Anh Khôi
Lê Đình Hiếu

HÀ NỘI, 7/2024

Lời cảm ơn

Chúng em xin cảm ơn ban tổ chức cuộc thi IoT Challenge – FPT Software, mentor Duy Linh đã dẫn dắt và tạo sân chơi cho chúng em được học tập và làm quen với nhiều công nghệ mới. Đồng thời chúng em cảm ơn các thành viên của MANDevices Lab đã hỗ trợ chúng em trong suốt quá trình phát triển sản phẩm.

Tóm tắt nội dung đề tài

Trong đề tài này, chúng em thực hiện thiết kế thùng rác tự động và mạng quản lý thùng rác tự động bằng các cảm biến, động cơ, giao thức mạng Zigbee, HTTP, … Phần mềm quản lý được xây dựng từ cơ sở dữ liệu Firebase Realtime data base. Mục tiêu phục vụ quản lý rác cho khu đô thị thông minh, chung cư và đặc biệt hướng đến thu thập, dọn dẹp rác thải tự động trong tương lai.

Báo cáo được cập nhật thêm trong những ngày phát triển tiếp theo tại link:

Cap_nhat_Bao_cao

MỤC LỤC

TỔNG QUAN ĐỀ TÀI

Giới thiệu chung

Trong những năm gần đây, tốc độ đô thị hóa nhanh chóng và sự gia tăng dân số đã tạo ra một lượng rác thải sinh hoạt khổng lồ. Các thành phố lớn và khu đô thị đang phải đối mặt với những thách thức nghiêm trọng trong việc quản lý và xử lý rác thải. Lượng rác thải sinh hoạt tăng nhanh không chỉ gây áp lực lớn lên hệ thống thu gom và xử lý rác mà còn ảnh hưởng tiêu cực đến môi trường sống, sức khỏe cộng đồng, và mỹ quan đô thị.

Rác thải không được quản lý tốt có thể gây ra ô nhiễm không khí, đất, và nước, dẫn đến nhiều hệ lụy nghiêm trọng. Các bãi rác lộ thiên và không được quản lý đúng cách có thể là nguồn gốc của các bệnh dịch và ảnh hưởng đến hệ sinh thái xung quanh. Bên cạnh đó, việc xử lý rác thải truyền thống bằng cách chôn lấp hoặc đốt cháy không chỉ gây lãng phí tài nguyên mà còn tạo ra khí thải độc hại, góp phần làm gia tăng hiệu ứng nhà kính và biến đổi khí hậu.

Để giải quyết vấn đề này, tự động hóa trong quản lý rác thải đô thị là một giải pháp tiên tiến và hiệu quả. Tự động hóa không chỉ giúp giảm thiểu sự phụ thuộc vào sức lao động con người mà còn cải thiện hiệu suất thu gom và xử lý rác thải. Việc áp dụng các công nghệ tiên tiến như cảm biến, động cơ tự động, và các giao thức mạng thông minh giúp tăng cường khả năng giám sát và quản lý rác thải một cách hiệu quả và chính xác.

Đề tài "Smart Garbage Mobitoring" ra đời với mục tiêu phục vụ quản lý rác thải cho các khu đô thị thông minh và chung cư. Hệ thống được thiết kế nhằm thu thập và quản lý dữ liệu rác thải một cách tự động, giúp giảm thiểu sự can thiệp của con người và nâng cao hiệu suất quản lý.

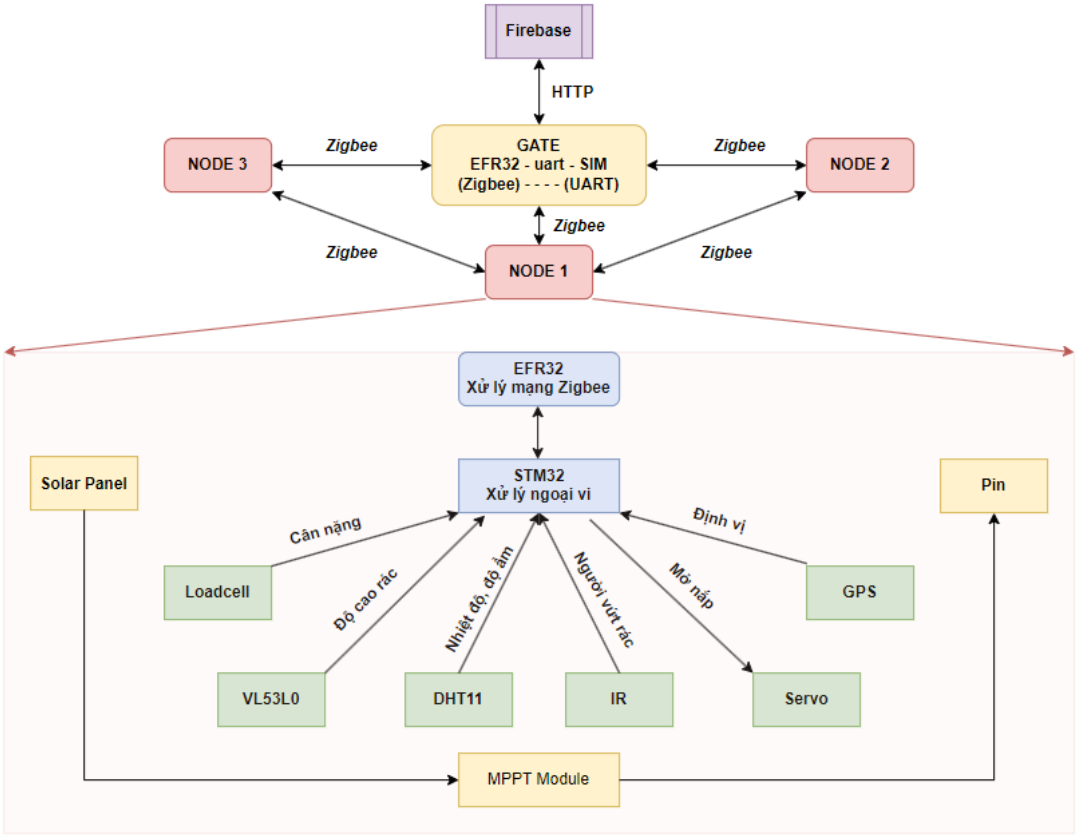
Các mục tiêu của đề tài

Việc triển khai hệ thống thùng rác tự động và mạng quản lý rác thải không chỉ mang lại lợi ích về mặt quản lý mà còn góp phần nâng cao chất lượng cuộc sống của người dân. Hệ thống giúp giảm thiểu tình trạng ô nhiễm, bảo vệ môi trường, và tạo ra một môi trường sống xanh, sạch, và đẹp. Bên cạnh đó, việc tự động hóa quy trình thu gom và xử lý rác thải giúp giảm thiểu chi phí vận hành và bảo trì, tạo ra một mô hình quản lý rác thải bền vững và hiệu quả.

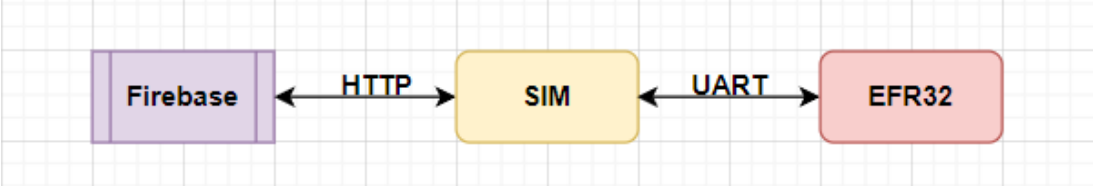
Trong tương lai, hệ thống này có thể được mở rộng và áp dụng rộng rãi tại các khu đô thị thông minh, khu chung cư, và các khu công nghiệp, góp phần vào việc xây dựng một môi trường sống hiện đại và bền vững. Việc thu thập và phân tích dữ liệu rác thải còn giúp đưa ra các chính sách quản lý phù hợp và hiệu quả, hỗ trợ nghiên cứu và phát triển các công nghệ mới trong lĩnh vực quản lý rác thải.

- Thiết kế thùng rác tự động: Thùng rác được trang bị các cảm biến để thu thập dữ liệu, động cơ tự động để mở và đóng nắp, và các module không dây để truyền dữ liệu.
- Phát triển phần mềm quản lý: Sử dụng Firebase Realtime Database để lưu trữ và quản lý dữ liệu, cung cấp thông tin theo thời gian thực cho người quản lý.

CÔNG NGHỆ VÀ THIẾT BỊ



GATE



Nhận dữ liệu từ Node

Gate hoạt động dựa trên các event của Zigbee được khởi tạo bởi người dùng và được xử bởi các hàm xử lí tương ứng khi các event được gọi đến. Khi nhận bản tin từ Node, hàm xử lí bản tin sẽ được gọi đến và xử lí bản tin phục vụ cho quá trình tiếp theo là giao tiếp với server.

Giao tiếp với Sever

a. <u>Lựa chọn Giao thức truyền thông</u>

Với yêu cầu gửi dữ liệu từ Gate lên Sever với mục tiêu gửi dữ liệu của nhiều NODE lên sever một số giao thức IoT có thể sử dụng:

MQTT (Message Queuing Telemetry Transport):

- Truyền thông tin dạng publish/subcribe cho phép tạo luồng dữ liệu đơn giản giữa các thiết bị khác nhau. Dựa trên giao thức TCP/IP.
- Giao thức đơn giản, nhẹ, tiết kiệm bằng thông nhưng vẫn đảm bảo độ ổn định và tin cậy trong việc truyền tải

- MQTT có khả năng thích ứng rộng rãi, có tiêu chuẩn phù hợp IoT trong công nghiệp nhưng không hỗ trợ định dạng dữ liệu phức tạp. Thường dùng kết hợp với Cloud để truyền dữ liệu từ thiết bị đến Cloud hoặc truyền lệnh điều khiển từ Cloud đến thiết bị.

HTTP (Hypertext Transfer Protocol):

- Là giao thức truyền tải thông tin giữa các thiết bị IoT và sever thông qua giao thức truyền tải dữ liệu trên mạng Internet.
- Thường dùng để truyền tải dữ liệu, truy cập các web hoặc các API tuy nhiên mức tiêu thụ năng lượng, tài nguyên lớn nên sử dụng khi dữ liệu cần truyền lớn.

CoAP (Constrained Application Protocol):

- Là giao thức lớp ứng dụng, thiết kế đặc biệt dựa trên HTTP để giải quyết những hạn chế của HTTP trong hệ thống IoT mà vẫn đảm bảo tính tin cậy của dữ liệu.
- Có khả năng hoạt động trong điều kiện mạng kém ổn định, giảm tiêu hao năng lượng và tài nguyên phần cứng.
- Phù hợp với các hệ thống theo dõi năng lượng, tự động hóa tòa nhà.

AMQP (Advanced Message Queuing Protocol):

- Giao thức lớp ứng dụng tiêu chuẩn mở, xếp hàng tin nhắn nâng cao.
- Các chức năng chính: Nhận và đặt tin nhắn vào hàng đợi, Lưu trữ tin nhắn, Thiết lập mối quan hệ giữa các thành phần dữ liệu.
- Mức độ bảo mật và độ tin cậy rất cao, phù hợp với những hệ thống cần đảm bảo tuyệt đối bảo mật. Tuy nhiên việc tốn kém tài nguyên phần cứng không phù hợp với các thiết bị IoT có bộ nhớ hạn chế.

Websocket:

- Truyền tải dữ liệu 2 chiều giữa các thiết bị IoT và server/cloud thông qua một kết nối TCP.
- Sử dụng kết nối định kì để truyền tải dữ liệu giữa các thiết bị IoT và server, đảm bảo tính liên tục và ổn định của dữ liệu được truyền tải.

DDS (Data Distribution Service):

- Tương tự MQTT, DDS cũng hoạt động với mô hình publish/subcribe, có khả năng mở rộng và cho truyền thông chất lượng cao.
- Là tiêu chuẩn IoT trung gian quốc tế mở đầu tiên.
- Có khả năng triển khai trên các thiết bị rất nhỏ như hệ thống nhúng và đảm bảo tính thời gian thực.
- DDS có một ưu điểm hơn MQTT đó là khả năng tương tác dữ liệu độc lập phần cứng và phần mềm.

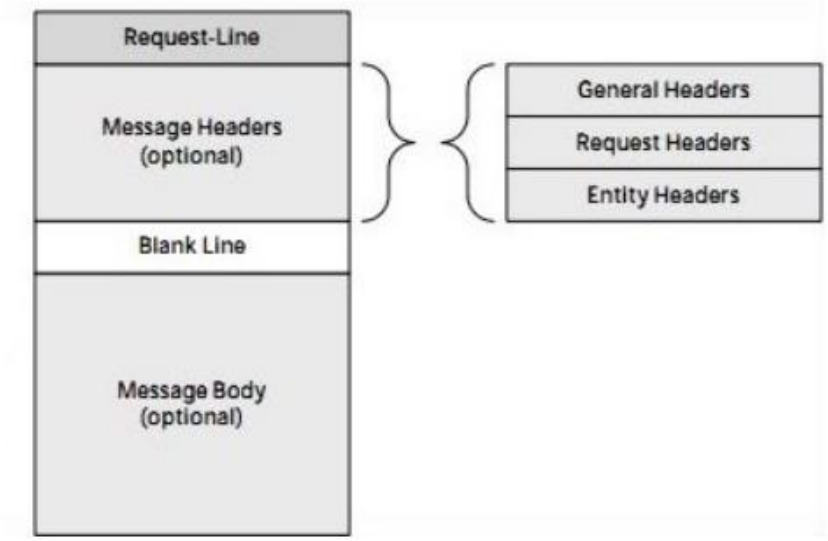
⇨ Mỗi giao thức truyền tin đều có những ưu điểm và nhược điểm khác nhau. Nhưng với các tiêu chí đặt ra để chọn lựa giao thức là: Hiệu suất, kết nối, tốc độ truyền tin, bản tin truyền tải và độ phù hợp với công nghệ truyền thông 4G. Với các tiêu chí trên, nhóm em quyết định chọn giao thức HTTP là giao thức truyền tin được sử dụng.

b. Tổng quan HTTP

Nguyên lý hoạt động:

HTTP (Hypertext Transfer Protocol) là một giao thức truyền tải siêu văn bản được sử dụng rộng rãi trên World Wide Web. HTTP hoạt động theo mô hình client-server, trong đó client (thường là trình duyệt web hoặc ứng dụng) gửi yêu cầu (request) đến máy chủ (server). Máy chủ nhận yêu cầu, xử lý và gửi lại phản hồi (response) chứa tài nguyên yêu cầu (như trang HTML, hình ảnh, video, dữ liệu JSON, v.v.). Mỗi giao dịch HTTP bao gồm một yêu cầu từ client và một phản hồi từ server.

Cấu trúc và yêu cầu phản hồi của HTTP:



(1) Cấu trúc của một yêu cầu HTTP

- Dòng yêu cầu (Request Line): Chứa phương thức HTTP (như GET, POST), URL của tài nguyên được yêu cầu, và phiên bản HTTP (ví dụ: HTTP/1.1).
- Tiêu đề yêu cầu (Request Headers): Là các cặp khóa-giá trị cung cấp thông tin bổ sung về yêu cầu, chẳng hạn như loại nội dung mà client có thể xử lý (Accept), thông tin về trình duyệt (User-Agent), và thông tin xác thực (Authorization).
- Thân yêu cầu (Request Body): Nếu có, chứa dữ liệu cần gửi tới server, thường được sử dụng trong các yêu cầu POST hoặc PUT để truyền dữ liệu từ client đến server.

(2) Cấu trúc của một phản hồi HTTP

- Dòng trạng thái (Status Line): Bao gồm phiên bản HTTP, mã trạng thái (như 200, 404), và thông điệp trạng thái tương ứng (như OK, Not Found).
- Tiêu đề phản hồi (Response Headers): Là các cặp khóa-giá trị cung cấp thông tin về phản hồi, như loại nội dung (Content-Type), kích thước nội dung (ContentLength), và thông tin về máy chủ (Server).
- Thân phản hồi (Response Body): Chứa dữ liệu thực tế được server trả về, có thể là tài liệu HTML, hình ảnh, hoặc dữ liệu JSON.

c. **Module truyền thông: SIM 7680C**

Khối 4G/LTE có nhiệm vụ kết nối thiết bị với server thông qua công nghệ 4G để trao đổi dữ liệu. Trong thị trường hiện nay có rất nhiều module có thể sử dụng 4G truyền dữ liệu với nhiều loại ứng dụng khác nhau. Trong đồ án lần này em chọn module 4G SIMCOM A7680C của hãng SIMCOM và TDMAKER.

Thông số kỹ thuật:

- Điện áp hoạt động : 3.7 – 4 V
- Băng tần: LTE-CAT 1 10Mbps, LTE-TDD: B34/B38/B39/B40/B41, LTE-FDD: B1/B3/B5/B8
- Hỗ trợ nhiều chế độ hoạt động khác nhau
- Hỗ trợ giao thức : TCP/IP, IPV4/IPV6, MultiPDP, FTP, HTTP, DNS, RNDIS, ECM, PPP, TLS, LBS, TTS, MQTT . . . .
- Hỗ trợ gửi SMS và gọi điện thông qua mạng 4G.
- Hỗ trợ sử dụng tập lệnh AT để giao tiếp với các MCU
- Ngoại vi : Sim Card, GPIO, UART, ADC
- Giao tiếp UART baudrate từ 9600-115200, mặc định là 115200

-

d. **Cơ sở dữ liệu Firebase và quy trình đẩy bản tin lên bằng HTTP**

Với mục đích dễ dàng triển khai các ứng dụng điều khiển qua web và ứng dụng di động do nhóm chưa có nhiều kinh nghiệm trong việc phát triển các app điều khiển vì vậy Firebase được đề xuất sử dụng với một số ưu điểm sau:

- Firebase cung cấp rất nhiều tính năng và công cụ giúp phát triển ứng dụng nhanh chóng nhờ đó hệ thống có tính thời gian thực cập nhật dữ liệu ngay khi nó được thay đổi mà không cần nhiều kinh nghiệm làm việc back-end cũng có thể dễ dàng mở rộng hệ thống.
- Firebase hoạt động dưới sự quản lý và trên cơ sở hạ tầng của Google nên đảm bảo được sự tin cậy về tính toàn vẹn và bảo mật của dữ liệu. -

Firebase cung cấp các tính năng miễn phí như cơ sở dữ liệu realtime, lưu trữ và đăng nhập và nhiều gói dịch vụ tiện ích có trả phí.

- Cộng đồng sử dụng lớn, nhiều tài liệu và video hướng dẫn, dễ tiếp cận.
- Hỗ trợ nhiều ngôn ngữ lập trình cho iOS, Android, Web, …
- Cung cấp các công nghệ AI giúp cải thiện tính năng hệ thống IoT, xử lý ảnh, ngôn ngữ tự nhiện, phân tích dữ liệu giúp người dùng hiểu rõ hơn về thói quen, hành vi, …

Các bước thực hiện HTTP giữa module SIM và Firebase:

1. Khởi tạo Module Sim:

Gửi lệnh "AT" để kiểm tra kết nối và nhận phản hồi "OK".

2. Kiểm tra đăng kí mạng

- Gửi lệnh "AT+CPIN?" để kiểm tra trạng thái SIM.
- Gửi lệnh "AT+CREG?" để kiểm tra trạng thái đăng ký mạng.

3. Thiết lập mạng

Gửi lệnh "AT+NETOPEN" để thực hiện khởi tạo mạng.

4. Khởi tạo HTTP

- Gửi lệnh "AT+HTTPINIT" để khởi tạo dịch vụ HTTP.
- Gửi lệnh "AT+HTTPPARA="CID",1" để chỉ định kênh kết nối.

5. Thiết lập HTTP

- Gửi lệnh "AT+HTTPPARA="URL","<firebase-url>"" để thiết lập URL.
- Gửi lệnh "AT+HTTPPARA="CONTENT","application/json"" để thiết lập loại nội dung.

6. Gửi dữ liệu HTTP

- Gửi lệnh "AT+HTTPDATA=<data-length>,<timeout>" để chuẩn bị gửi dữ liệu.
- Gửi dữ liệu JSON.

7. Xử lý phản hồi

Chờ phản hồi từ server và xử lý kết quả.

8. Kết thúc HTTP

Gửi lệnh "AT+HTTPTERM" để kết thúc dịch vụ HTTP.

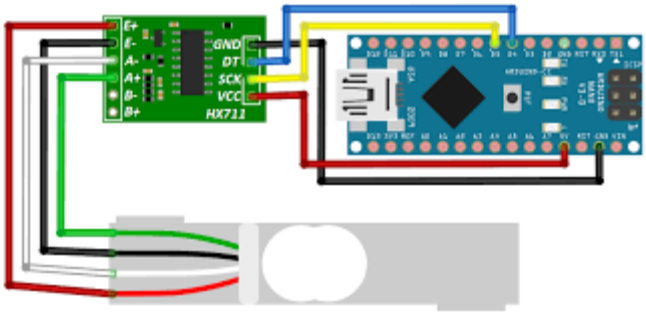
NODE

Lựa chọn phần cứng

a. Khối xử lý:

- EFR32MG24: Xử lý Mạng Zigbee.
- STM32F4: Xử lý ngoại vi.
  ⇨ Giao tiếp thông qua UART

b. Loadcell + HX711: Đo cân nặng rác trong thùng



Load cell là một cảm biến lực, được sử dụng để đo trọng lượng hoặc lực bằng cách chuyển đổi nó thành tín hiệu điện. **HX711** là một mô-đun khuếch đại tín hiệu chuyên dụng để đọc dữ liệu từ load cell, với độ chính xác cao và dễ dàng sử dụng trong các ứng dụng đo lường.

Cấu Tạo Load Cell

Load cell thường được làm từ kim loại và có một hoặc nhiều cảm biến lực bên trong. Các thành phần chính bao gồm:

- Thân load cell: Thường là kim loại có độ bền cao, được thiết kế để biến dạng dưới tác dụng của lực.
- Strain gauges (cảm biến biến dạng): Được gắn trên thân load cell. Khi load cell bị biến dạng do lực, các strain gauge này cũng bị biến dạng, làm thay đổi điện trở của chúng.
- Wheatstone bridge: Một mạch điện cầu gồm bốn strain gauge. Khi lực tác động lên load cell, sự thay đổi điện trở trong strain gauge làm mất cân bằng cầu, tạo ra một tín hiệu điện tỷ lệ với lực tác động.

Nguyên Lý Hoạt Động

- Biến dạng**:** Khi lực tác động lên load cell, thân kim loại của nó bị biến dạng.
- Thay đổi điện trở**:** Sự biến dạng này làm thay đổi điện trở của các strain gauge.
- Mất cân bằng cầu Wheatstone: Sự thay đổi điện trở dẫn đến mất cân bằng trong cầu Wheatstone, tạo ra một tín hiệu điện nhỏ.
- Khuếch đại tín hiệu: Tín hiệu điện nhỏ này cần được khuếch đại để có thể đọc được và xử lý.

HX711 là một bộ chuyển đổi tín hiệu từ load cell, với các đặc điểm chính:

- Khuếch đại tín hiệu: HX711 có hai kênh khuếch đại với độ lợi cố định 128 và 64, cho phép xử lý các tín hiệu rất nhỏ từ load cell.
- Chuyển đổi ADC: Tích hợp bộ chuyển đổi tương tự-số (ADC) với độ phân giải 24-bit, cho độ chính xác cao.
- Giao tiếp: HX711 giao tiếp với vi điều khiển thông qua giao thức hai dây (clock và data).

c. VL53L0: Đo độ cao rác trong thùng

Cấu tạo:

- **Module cảm biến**: Bao gồm cảm biến chính với các bộ phận phát và thu tia laser.

- **Bộ phát laser (VCSEL)**: Một diode laser phát ra các xung ánh sáng tia hồng ngoại.
- **Bộ thu ánh sáng (SPAD array)**: Một mảng các cảm biến thu nhận ánh sáng phản xạ trở lại từ bề mặt đối tượng.
- **Bộ xử lý tín hiệu (ASIC)**: Bộ xử lý tích hợp để tính toán thời gian bay (Time-of-Flight) của tia laser.
- **Giao tiếp I2C**: Cho phép kết nối và giao tiếp với vi điều khiển hoặc các thiết bị khác.

Nguyên lý hoạt động:

- **Phát tia laser**: Cảm biến phát ra một xung ánh sáng laser hồng ngoại từ bộ phát laser.
- **Phản xạ ánh sáng**: Xung ánh sáng này di chuyển đến bề mặt đối tượng và phản xạ trở lại.
- **Thu nhận tín hiệu**: Bộ thu ánh sáng (SPAD array) nhận lại ánh sáng phản xạ.
- **Tính toán thời gian**: Bộ xử lý tín hiệu (ASIC) tính toán thời gian mà tia laser đi từ cảm biến đến bề mặt đối tượng và trở lại (Time-of-Flight).
- **Tính khoảng cách**: Dựa trên thời gian bay của tia laser, cảm biến tính toán khoảng cách từ cảm biến đến bề mặt đối tượng.

Dải đo:

- Khoảng cách: Từ 30 mm đến 2 mét.
- Độ chính xác: Dải đo với độ chính xác lên đến ±3%.
- Độ phân giải: Có khả năng đo với độ phân giải cao, cho phép phát hiện các đối tượng nhỏ và xác định khoảng cách một cách chính xác.

d. Cảm biến hồng ngoại: Phát hiện người vứt rác

Cấu tạo:

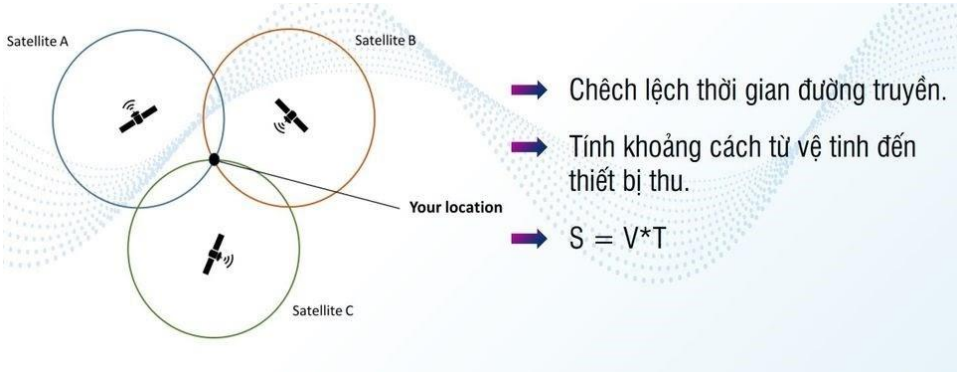
- Bộ phát hồng ngoại (LED hồng ngoại): Phát ra ánh sáng hồng ngoại, không thể nhìn thấy bằng mắt thường.
- Bộ thu hồng ngoại (Photodiode hoặc Phototransistor): Nhận ánh sáng hồng ngoại phản xạ trở lại từ bề mặt đối tượng.
- Mạch xử lý: Khuếch đại và xử lý tín hiệu từ bộ thu để xác định khoảng cách hoặc phát hiện đối tượng.

Nguyên lý hoạt động:

- Phát tia hồng ngoại: Bộ phát hồng ngoại phát ra ánh sáng hồng ngoại hướng về phía đối tượng cần đo.
- Phản xạ ánh sáng: Ánh sáng hồng ngoại phát ra sẽ phản xạ lại khi gặp bề mặt của đối tượng.
- Thu nhận ánh sáng: Bộ thu hồng ngoại sẽ nhận ánh sáng hồng ngoại phản xạ trở lại.
- Xử lý tín hiệu: Mạch xử lý sẽ xử lý tín hiệu từ bộ thu để xác định khoảng cách hoặc phát hiện sự có mặt của đối tượng.

e. Module GPS NEO 6M định vị thùng rác

Module GPS NEO-6M hoạt động dựa trên việc nhận tín hiệu từ nhiều vệ tinh trong hệ thống GPS. Dựa vào thời gian tín hiệu truyền từ vệ tinh đến module, thiết bị sẽ tính toán khoảng cách từ nó đến các vệ tinh đó và từ đó xác định vị trí chính xác của nó trên bề mặt Trái đất thông qua phương pháp ba giác.



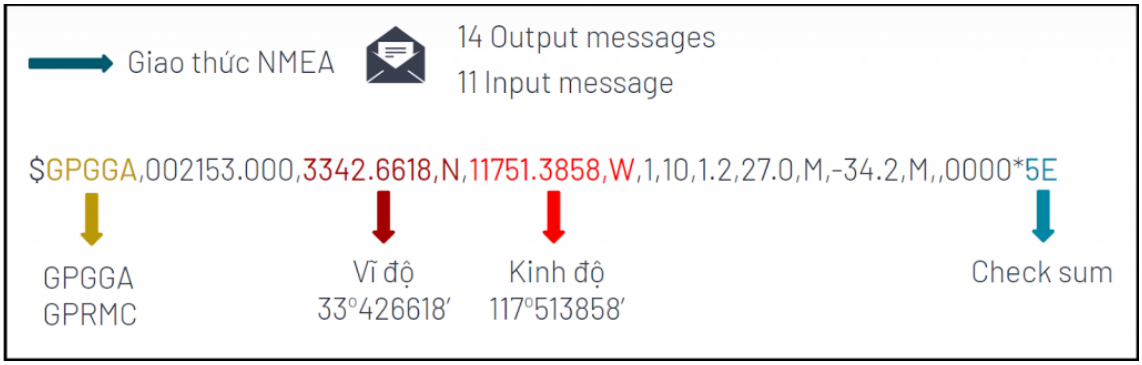
Trình tự thu thập dữ liệu:

1. Thu tín hiệu từ vệ tinh: Antenna GPS thu tín hiệu từ ít nhất 4 vệ tinh để xác định vị trí.
2. Giải mã tín hiệu: Module giải mã tín hiệu thu được để lấy dữ liệu về thời gian và khoảng cách từ vệ tinh.
3. Tính toán vị trí: Sử dụng phương pháp ba giác để tính toán vị trí chính xác dựa trên dữ liệu thu được.
4. Truyền dữ liệu: Gửi dữ liệu vị trí dưới dạng bản tin NMEA thông qua giao tiếp UART đến vi điều khiển hoặc máy tính.

Thành phần bản tin:

Module GPS NEO-6M truyền dữ liệu vị trí dưới dạng bản tin NMEA (National Marine Electronics Association):

- **GGA:** Thông tin vị trí và độ chính xác.
- **RMC:** Thông tin vị trí và thời gian.
- **GSV:** Thông tin về các vệ tinh đang theo dõi.
- **GSA:** Thông tin về trạng thái định vị.



Xử lý bản tin:

1. Thiết lập giao tiếp UART: Kết nối module GPS với vi điều khiển hoặc máy tính qua giao tiếp UART.
2. Nhận dữ liệu từ UART: Đọc dữ liệu từ UART và lưu trữ trong bộ đệm.
3. Xác định bản tin NMEA: Tìm ký tự $ để xác định bắt đầu của một bản tin NMEA và ký tự \n để xác định kết thúc bản tin.
4. Tách chuỗi dữ liệu: Sử dụng dấu phẩy (,) để tách các trường dữ liệu trong bản tin NMEA.
5. Phân tích dữ liệu: Giải mã các trường dữ liệu để lấy thông tin vị trí, thời gian, và các thông số khác.

f. Động cơ Servo SG90: Mở nắp tự động

Cấu Tạo: Động cơ servo SG90 bao gồm một động cơ DC, hộp số, mạch điều khiển, biến trở và vỏ bảo vệ.

- Vỏ Ngoài (Case):Vỏ ngoài của động cơ làm từ nhựa, bảo vệ các bộ phận bên trong khỏi bụi bẩn và va đập.
- Động Cơ DC:Một động cơ DC nhỏ gọn được sử dụng để tạo ra chuyển động quay.
- Hộp Số (Gearbox):Hộp số giảm tốc gồm nhiều bánh răng giúp giảm tốc độ quay của động cơ và tăng mô-men xoắn, đồng thời chuyển động quay của động cơ thành chuyển động quay chính xác của trục ra.
- Mạch Điều Khiển (Control Circuit):Mạch điều khiển bên trong động cơ servo xử lý tín hiệu điều khiển từ bên ngoài và điều chỉnh vị trí của động cơ theo tín hiệu này.
- Biến Trở (Potentiometer):Một biến trở gắn trên trục ra của động cơ để theo dõi vị trí hiện tại của trục. Thông tin vị trí này được gửi về mạch điều khiển để điều chỉnh và duy trì vị trí chính xác.
- Trục Ra (Output Shaft):Trục ra được kết nối với hộp số và là phần mà người dùng có thể gắn các thành phần khác để điều khiển.

Nguyên lý hoạt động: Động cơ hoạt động dựa trên nguyên lý điều khiển vị trí thông qua tín hiệu PWM, mạch điều khiển so sánh vị trí hiện tại và vị trí mong muốn, điều chỉnh động cơ để đạt được vị trí mong muốn. Các hoạt động chính của động cơ như sau:

- Nhận Tín Hiệu PWM: Động cơ nhận tín hiệu điều khiển PWM từ một vi điều khiển hoặc thiết bị điều khiển bên ngoài. Tín hiệu PWM có chu kỳ (period) khoảng 20 ms, với độ rộng xung (pulse width) từ 1 ms đến 2 ms.
- Xác Định Vị Trí Mong Muốn:Độ rộng xung của tín hiệu PWM xác định vị trí mong muốn của trục động cơ. Thông thường, xung 1 ms tương ứng với góc 0 độ, xung 1.5 ms tương ứng với góc 90 độ, và xung 2 ms tương ứng với góc 180 độ.
- So Sánh Vị Trí:Mạch điều khiển so sánh vị trí hiện tại của trục (được đo bằng biến trở) với vị trí mong muốn (được xác định bởi tín hiệu PWM).

- Điều Chỉnh Vị Trí:Nếu vị trí hiện tại không khớp với vị trí mong muốn, mạch điều khiển sẽ điều chỉnh dòng điện tới động cơ DC để quay trục đến vị trí mong muốn.
- Phản Hồi Liên Tục:Quá trình so sánh và điều chỉnh này diễn ra liên tục để duy trì vị trí chính xác của trục động cơ theo tín hiệu PWM.

g. Relay: Bật đèn quảng cáo

Cấu tạo của Relay:

- Cuộn dây (Coil): Là phần chứa một dây dẫn điện được cuộn quanh một lõi từ. Khi dòng điện được điều khiển qua cuộn dây này, nó tạo ra một từ trường từ cuộn dây, làm cho các bộ phận cơ khí bên trong relay chuyển động.
- Các tiếp điểm (Contacts): Relay có thể có nhiều tiếp điểm, bao gồm:
    - Tiếp điểm Normally Open (NO): Khi relay không được kích hoạt (không có dòng điện điều khiển), tiếp điểm này mở ra và ngắt mạch.
    - Tiếp điểm Normally Closed (NC): Khi relay không được kích hoạt, tiếp điểm này đóng lại và hoàn thành mạch.
- Cơ cấu chuyển động: Bao gồm các cơ cấu cơ khí như bộ đếm, cơ cấu nhấn, hoặc bộ cơ cấu khác nhằm mở hoặc đóng tiếp điểm khi relay được kích hoạt.

Nguyên lý hoạt động của Relay:

- Khi dòng điện được cấp vào cuộn dây của relay (coil), nó tạo ra một từ trường từ, làm cho các bộ phận cơ khí bên trong relay chuyển động.
- Sự chuyển động này làm thay đổi trạng thái của các tiếp điểm (NO hoặc NC). Chẳng hạn, nếu relay có tiếp điểm NO, khi cuộn dây được kích hoạt, tiếp điểm NO sẽ đóng mạch (kết nối các đầu vào và đầu ra của relay lại với nhau), cho phép dòng điện chạy qua mạch được kết nối.
- Ngược lại, nếu relay có tiếp điểm NC, khi cuộn dây được kích hoạt, tiếp điểm NC sẽ mở ra, ngắt mạch (cắt dòng điện chạy qua mạch được kết nối).

h. Si7021:Đo nhiệt độ, độ ẩm

Cấu tạo:

- Cảm biến nhiệt độ và độ ẩm: Si7021 tích hợp sẵn trong một module IC nhỏ gọn.
- Giao tiếp I2C: Cảm biến có thể truyền dữ liệu nhiệt độ và độ ẩm kỹ thuật số thông qua giao tiếp này.
- Bộ xử lý và bộ nhớ nội: Đảm bảo tính toàn vẹn và độ tin cậy của dữ liệu đo.

Nguyên lý hoạt động:

- Đo nhiệt độ: Sử dụng sự thay đổi của điện trở hoặc điện dung trong vùng cảm biến nhiệt độ khi nhiệt độ thay đổi.
- Đo độ ẩm: Dựa trên sự thay đổi của điện trở hoặc điện dung của vật liệu hấp thụ nước trong không khí xung quanh.

Cảm biến Si7021 của Silicon Labs cung cấp dải đo tín hiệu đầu ra như sau:

1. Nhiệt độ:
    - Dải đo: từ -40°C đến +125°C.
    - Độ chính xác: ±0.4°C (ở khoảng nhiệt độ từ -10°C đến +85°C).
2. Độ ẩm:
    - Dải đo: từ 0% đến 100% độ ẩm tương đối.
    - Độ chính xác: ±3% (ở khoảng độ ẩm từ 20% đến 80% RH).

i. Cảm biến H2S/NH3: Đo nồng độ khí gây mùi

Do giá thành cảm biến lớn, khó mua nên tạm thời chúng em đề xuất sử một một cảm biến khí tương tự khác là MQ-2. Các thành phần khí mà cảm biến này H2, LPG, Co, CH4, smoke,... Tuy nhiên các cảm biến trong series các cảm biến MQ đều có cùng cách thức thu thập và xử lý tín hiệu nên hoàn toàn có thể demo sản phần MQ-2.

Cấu tạo:

- **Oxide Thiếc (SnO2)**: Phần tử nhạy cảm của MQ-2 là một lớp oxide thiếc (SnO2), một chất bán dẫn có điện trở thay đổi khi tiếp xúc với các loại khí khác nhau. Khi gặp các khí dễ cháy như H2, LPG, methane, CO, và khói, điện trở của lớp oxide thiếc sẽ giảm.
- **Lớp Sưởi (Heater Coil)**:Cảm biến MQ-2 có một lớp sưởi tích hợp, thường được làm từ dây nicrôm, để duy trì nhiệt độ cao cần thiết cho phần tử nhạy cảm hoạt động hiệu quả. Nhiệt độ này giúp tăng cường phản ứng của oxide thiếc với các khí cần phát hiện.
- **Lưới Kim Loại (Metal Mesh)**: Cảm biến được bảo vệ bởi một lớp lưới kim loại không gỉ, giúp lọc bụi và các hạt lớn khác, đồng thời bảo vệ lớp sưởi và phần tử nhạy cảm khỏi các tác động cơ học và hóa học từ môi trường bên ngoài.

Nguyên lý hoạt động:

Thành phần điện trở bán dẫn được nối đến một điện trở không đổi tạo thành bộ chia điện áp. Khi nồng độ khí tăng lên làm thay đổi điện trở của điện trở bán dẫn, dẫn đến sự thay đổi điện áp ở đầu ra.

- EFR32MG24: Xử lý Mạng Zigbee
- STM32F4: Xử lý ngoại vi
    ⇨ Giao tiếp thông qua UART

Giao tiếp USART giữa STM32 và EFR32

Khung bản tin:

ID(Start)	Temp	Humi	Mass	Lot	Lat	Height	UV	Stop

Byte start Data frame Byte stop

Bản tin được truyền từ STM32 đến EFR32 sẽ được xử lí và truyền vào trường data quản lý bởi 1 struct để dễ dàng quản lí các thông số cảm biến thu thập được.

Giao thức truyền thông không dây

Personal Area Network Identifiers	PANID
Extended Personal Area Network Identifiers	EPANID
Zigbee Coordinator	ZC
Endpoint	EP
Zigbee Router	ZR
Zigbee End Device	ZED
Zigbee 3.0	Z3

Sử dụng mạng không dây Zigbee cho hệ thống để giao tiếp giữa các node

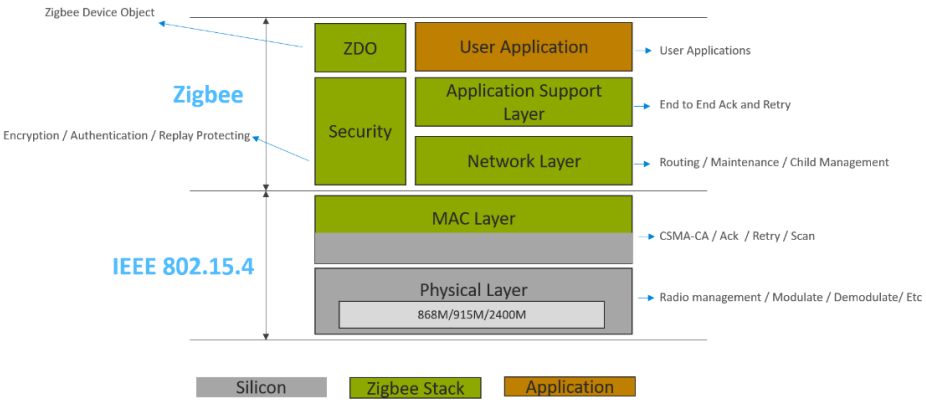
a) Giới thiệu về mạng Zigbee

- Là một chuẩn truyền thông không dây theo tiêu chuẩn 802.15.4 được hỗ trợ bởi tổ chức Zigbee Alliance.
- Là sóng RF có 3 dải tần (868MHz -1 kênh, 915MHz - 10 kênh, 2.4GHz - 16 kênh). Phần lớn Zigbee hoạt động ở dải tần 2.4 GHz.
- Chuẩn Zigbee mới nhất: Zigbee 3.0
- Mô hình mạng: Mạng Mesh

b) Đặc điểm của mạng Zigbee

- Công suất tiêu thụ thấp
- Tốc độ thấp: 250Kbps
- Độ tin cậy cao (với việc sử dụng mô hình mạng Mesh)
- Có thể sử dụng toàn cầu (do là chuẩn mở và dùng băng tần 2.4GHz)
- Ưu điểm
  - Tiêu thụ ít năng lượng
  - Hỗ trợ lượng node lớn (lên đến 65535 node)
  - Tính tương thích cao
  - Giá rẻ
- Nhược điểm
  - Vì sử dụng băng tần phổ thông 2.4Hz nên dễ bị nhiễu.
  - So với Bluetooth Mesh/Wifi thì không điều khiển trực tiếp với Smartphone mà vẫn phải qua Gateway.

c) Các khái niệm và thành phần trong mạng

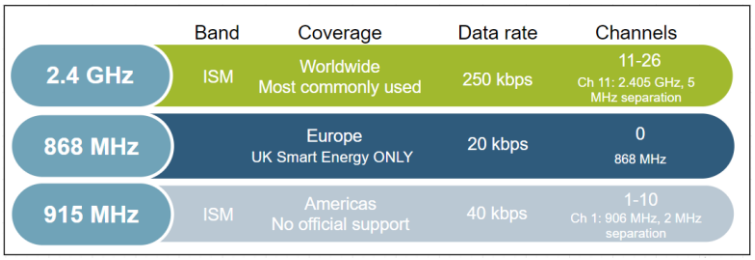


Hình 2.1 Kiến trúc mạng Zigbee

Bao gồm 5 tầng:

- **Physical Layer**: Tầng này thực việc việc mã hóa và giải mã tín hiệu khi truyền và nhận tín hiệu tương ứng. Tần số, tốc độ dữ liệu và số lượng kênh của tang này được liệt kê ở trên mà đã trình bày.
- **MAC Layer**: tầng MAC có nhiệm vụ quản lý truy cập vào mạng, đảm bảo rằng dữ liệu được truyền đi một cách đáng tin cậy và tránh xung đột. Giao thức CSMA giúp kiểm tra xem môi trường truyền dẫn có sẵn để truyền dữ liệu hay không, tránh va chạm dữ liệu trong quá trình truyền. Ngoài ra, tầng MAC cũng có nhiệm vụ truyền các khung báo hiệu (beacon frames) để đồng bộ hóa các thiết bị trong mạng cảm biến không dây, đặc biệt là khi chúng cần biết thời gian hoặc thông tin địa lý để thực hiện giao tiếp.
- **Network Layer**: chịu trách nhiệm quản lý và điều phối các hoạt động mạng. Điều này bao gồm việc xây dựng và duy trì kết nối giữa các thiết bị đầu cuối và mạng, định tuyến dữ liệu từ nguồn tới đích, cấu hình thiết bị để chúng có thể tham gia mạng, và các nhiệm vụ liên quan khác. Tầng Mạng cũng đảm bảo rằng thông tin được định tuyến một cách hiệu quả và đáng tin cậy trong mạng cảm biến, giúp cho dữ liệu có thể chuyển đổi giữa các thiết bị trong mạng một cách hiệu suất và đảm bảo tính ổn định của mạng.
- **Application Support Sub-Layer**: đóng vai trò quan trọng trong việc đảm bảo rằng các đối tượng thiết bị và ứng dụng có thể tương tác và giao tiếp một cách hiệu quả. Nó giúp kết nối các thiết bị dựa trên khả năng và dịch vụ mà chúng cung cấp, đồng thời đáp ứng các yêu cầu của ứng dụng. Điều này đảm bảo rằng các thiết bị và ứng dụng có thể làm việc cùng nhau trong mạng Zigbee để quản lý và chuyển đổi dữ liệu một cách hiệu quả.
- **Application Framework**: cung cấp hai loại dịch vụ dữ liệu: dịch vụ key value pair (cặp khóa-giá trị) và dịch vụ thông điệp chung (generic message). Thông điệp chung là một cấu trúc được định nghĩa bởi nhà phát triển, trong khi cặp khóa-giá trị được sử dụng để lấy các thuộc tính trong các đối tượng ứng dụng. ZDO (Zigbee Device Object) cung cấp một giao diện giữa các đối tượng ứng dụng và tầng APS trong các thiết bị Zigbee. Nó chịu trách nhiệm phát hiện, khởi tạo và kết nối các thiết bị khác vào mạng.

d) Các khái niệm và thành phần trong mạng



Các kênh trong mạng Zigbee

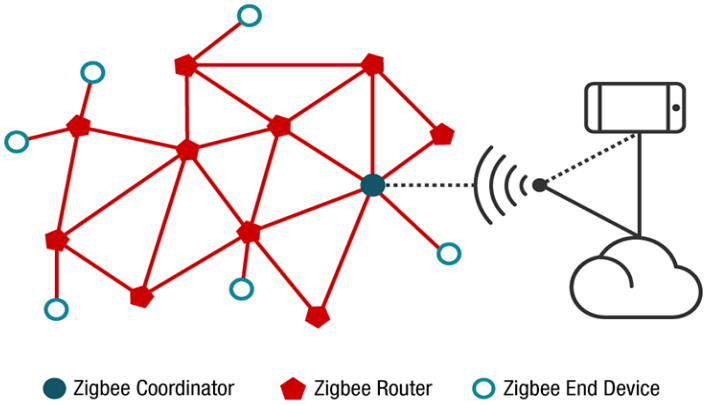
Hoạt động ở 3 dải tần:

- Dải tần 868MHz có 1 kênh (kênh 0) tốc độ 20kbps, sử dụng ở châu âu
- Dải tần 915 MHz có 10 kênh (từ kênh 1-10) tốc độ 40 Kbps, sử dụng ở Mỹ
- Dải tần 2.4 GHz có 16 kênh (từ kênh 11-26) tốc độ 250 Kbps, sử dụng trên toàn thế giới

Trong mạng Zigbee, có 3 loại node

- Co-ordinator
    - Nó hình thành một mạng: Người điều phối chịu trách nhiệm tạo và quản lý mạng. Nó cho phép thiết bị tham gia vào mạng và xác định các tham số của mạng.
    - Nó thiết lập kênh mà mạng sẽ hoạt động trên đó: Nó chọn kênh cụ thể trong dải tần số mà mạng sẽ hoạt động. Điều này quan trọng để tránh xung đột từ các thiết bị không dây khác.
    - Nó thiết lập PAN ID mở rộng và PAN ID cho mạng.
    - Nó quyết định về cấu hình của ngăn xếp (stack profile) để sử dụng
    - Nó chịu trách nhiệm quản lý bảo mật trong mạng, bao gồm quản lý khóa và xác thực, để đảm bảo giao tiếp an toàn.
    - Nó hoạt động như một bộ định tuyến cho định tuyến lưới (mesh routing): Trong mạng lưới, coordinator cũng có thể hoạt động như một bộ định tuyến, chuyển tiếp dữ liệu giữa các thiết bị để đảm bảo nó đến được đích một cách hiệu quả.
- Router
    - Tìm và tham gia vào đúng mạng.
    - Thiết bị có thể tham gia vào quá trình định tuyến dữ liệu trong mạng. Điều này bao gồm việc tìm ra các đường tuyến hiệu quả để chuyển tiếp dữ liệu từ nguồn tới đích và duy trì các đường tuyến này khi cần thiết.
    - Cho phép các thiết bị khác tham gia vào mạng (nếu cho phép tham gia): Thiết bị có thể chấp nhận và quản lý việc tham gia của các thiết bị mới vào mạng.
- End device, bao gồm non-sleepy end device và sleep end device
    - Tìm và tham gia vào đúng mạng.
    - Tìm kiếm cha mẹ mới nếu kết nối với cha mẹ cũ bị mất (NWK rejoin): Khi một thiết bị con mất kết nối với cha mẹ hiện tại (parent

device) do nguyên nhân nào đó, nó cần tìm một cha mẹ mới để kết nối. Quá trình này được gọi là NWK (Network) Rejoin, và nó giúp đảm bảo rằng thiết bị con vẫn có thể tham gia vào mạng và gửi/receive dữ liệu một cách liên tục.



Các loại node trong mạng Zigbee

(1) PanID

Viết tắt của từ ZigBee Personal Area Network Identifiers. Là một số 16 bit được sinh ra bởi Zigbee Coordinator(ZC) khi thành lập mạng hoặc được có thể được pre-configured. Dùng để phân biệt mạng này với mạng kia trên cùng một kênh.

Hai mạng Zigbee trên 2 kênh khác nhau có thể trùng PANID có giá trị từ 0x0000 to 0xFFFE. Khi thành lập mạng nếu tham số PANID để 0xFFFF thì Zigbee Stack sẽ random một PANID bất kỳ không trùng với PANID mạng khác. Thường để mặc định là 0xFFFF. Khi Join vào một mạng, để tham số PANID là 0xFFFF thì node đó muốn thông báo rằng nó muốn tham gia vào bất kỳ mạng nào không cần quan tâm tới PANID.

Nếu tình cờ chọn 1 PANID đã được sử dụng bởi 1 mạng khác hoặc nếu chọn 1 PANID ngẫu nhiên không xung đột với bất kỳ mạng nào nhưng sau đó 1 mạng khác phát triển chồng lên mạng chúng ta đang sử dụng, trên thực tế, ngăn xếp có thể phát hiện xung đột như vậy và có thể tự động cập nhật PAN ID của nó và thông báo cho tất cả các nút trong mạng chuyển sang PAN ID mới để mỗi mỗi nút có thể tiếp tục giao tiếp với các nút trong mạng ban đầu của nó và loại trừ bất kỳ thứ gì trên mạng.

(2) EPanID

Viết tắt của từ Extended PAN ID. Có địa địa chỉ 64 bit, trong đó, 24 bit đầu thường được gọi là OUI (Organizationally Unique Identifier) - một số duy nhất gồm 24 bit được cấp bởi IEEE (Institute of Electrical and Electronics Engineers) cho các công ty sản xuất sản phẩm, những bit sau là Random.

Được sinh ra lúc ZC thành lập mạng, thường lấy trùng với địa chỉ MAC của ZC. Hầu hết các trường hợp gửi bản tin trong mạng Zigbee chỉ dùng địa chỉ PANID bởi vì nó ngắn và đơn giản.

Được lưu trong bộ nhớ non-volatile khi node join vào mạng, và sẽ không thay đổi cho đến khi node dời mạng

Sử dụng EPANID:

- Khi update lại mạng: ví dụ khi thay đổi kênh hoặc thay đổi PANID vì xung đột thì dùng địa chỉ EPANID bởi tính duy nhất và không thay đổi của nó.
- Khi xảy ra xung đột PAN ID và chúng ta muốn thông báo cho tất cả các thiết bị trong mạng của mình di chuyển theo cách mà chúng ta phân biệt với mạng xung đột là tất cả các thiết bị trong mạng của bạn đều chia sẻ cùng 1 EPANID.

Mỗi mạng Zigbee đều có cơ chế bảo mật mà các mạng khác không thể đọc được bản tin. Hai mạng Zigbee sẽ không thể nói chuyện với nhau khi một trong hai tham số Channel và PANID khác nhau. Ngoài ra EPAN cũng để phân biệt giữa hai mạng trong một số trường hợp (join mạng, update mạng).

(3) Address

Bên cạnh các tiêu chí trên toàn mạng, một nút được phân biệt với một nút khác bởi địa chỉ của nó. Một nút có một địa chỉ ngắn và một địa chỉ dài. Địa chỉ dài là địa chỉ MAC được gán bởi IEEE, còn được gọi là EUI-64. Đây là một địa chỉ 64 bit duy nhất trên toàn cầu, có nghĩa là không có hai bộ địa chỉ EUI-64 dựa trên IEEE nào trên thế giới sẽ bao giờ giống nhau. Điều này thường được gán khi sản xuất. Chúng được cài đặt khi chip ra khỏi nhà máy sản xuất trước khi đến với người dùng, và chúng sẽ không bao giờ thay đổi. Tuy nhiên, vì 64 bit là một lượng dữ liệu lớn, địa chỉ dài này thường không được gửi qua môi trường truyền.

Hầu hết thời gian, địa chỉ ngắn hơn nhiều, gồm 16 bit, được sử dụng qua môi trường truyền. Đây được gọi là Node ID và là duy nhất trong mạng. Nó được gán khi nút gia nhập vào mạng và nó được giả định là duy nhất trong mạng đó. Có thể có hai mạng, mỗi mạng có một nút có cùng Node ID, nhưng vì chúng nằm trong các PAN khác nhau, điều này không quan trọng. Lưu ý rằng hai nút có thể chọn cùng một Node ID ngẫu nhiên khi gia nhập vào mạng. Nếu điều đó xảy ra, tương tự như sơ đồ PAN ID, có một phương pháp để giải quyết xung đột. Sử dụng thông tin EUI-64 như một phương án dự phòng, các nút có thể đồng ý trên các địa chỉ mới khi họ nhận ra xung đột. Do đó, các nút có thể thay đổi địa chỉ tại thời điểm chạy nếu cần thiết, dựa trên xung đột.

(4) Endpoint

Là một số 8 bit có giá trị từ 1-255, để xác định các Application chạy trên một node. Một node có thể có nhiều Application (nhiều EP). Vì thế -Một node có thể support nhiều Application Profiles: Home Automation, Commercial Building Automation,…

Broadcast endpoint (0xFF); khi gửi một data request tới EP này thì tất cả các EP trong node đó có cùng profile ID cũng sẽ nhận được bản tin này.

(5) Profile

Profile là một tập hợp các tiêu chuẩn được định nghĩa để hỗ trợ các thiết bị Zigbee thực hiện một số nhiệm vụ cụ thể. Ví dụ, Profile Home Automation sử dụng để điều khiển các thiết bị trong gia đình như đèn, máy lạnh, thiết bị an ninh. Hay Profile Home Controls Light (HCL), nó xác định một số thiết bị và chức năng cần thiết hoặc hữu ích để điều khiển hệ thống chiếu sáng trong nhà, chẳng

hạn như công tắc, bộ điều chỉnh độ sáng, cảm biến chiếm chỗ và bộ điều khiển tải (điều khiển nguồn sáng).

Profile ID	Profile Name
0101	Industrial Plant Monitoring (IPM)
0104	Home Automation (HA)
0105	Commercial Building Automation (CBA)
0107	Telecom Applications (TA)
0108	Personal Home & Hospital Care (PHHC)
0109	Advanced Metering Initiative (AMI)

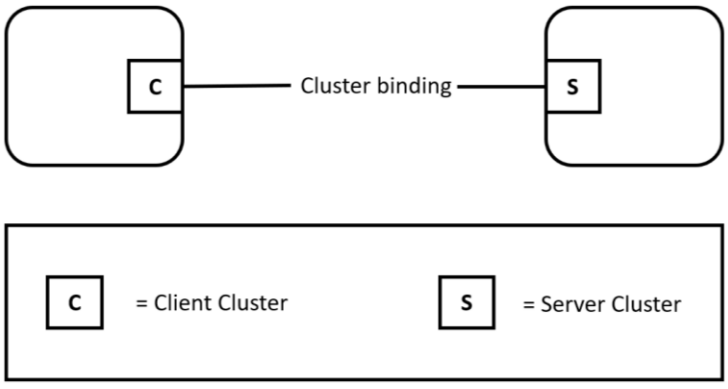
Hình 2.4 Ví dụ về Profile

Giống như một nhóm các thiết bị hoặc ứng dụng trong một ngành cụ thể. Profile IDs là một số 16-bit và dải từ 0x0000 tới 0x7FFF cho public profiles và từ 0xBF00 tới 0xFFFF cho manufacturer specific profiles.

(6) Cluster

Trong mỗi Endpoint, chúng ta có thể cấu hình một vài cluster. Cluster là một nhóm các command và attribute được sử dụng để điều khiển và truyền tải dữ liệu giữa các thiết bị Zigbee. Mỗi Cluster chứa một số lượng các command và attribute khác nhau tùy thuộc vào mục đích sử dụng của nó.

Giả sử ta có the On/Off cluster. Cái cluster này dùng để điều khiển trạng thái on/off của thiết bị. Nó định nghĩa attribute như là trạng thái On/Off và command như là On/Off/Toggle. Những thiết mà được hỗ trợ On/Off cluster có thể giao tiếp và điều khiển trạng thái bằng các sử dụng các lệnh. Zigbee Cluster thực chất là 1 mô hình truyền thông.



Hình 2.5 Mô hình Zigbee Cluster

Nó dựa trên chế độ Client/Server và được sử dụng để mô tả giao thức ứng dụng giữa hai thiết bị. Mỗi 1 cluster có một Cluster ID được xác định trong Zigbee Cluster Library (ZCL). Một Cluster có thể xác định một số Attribute và Command. Trong quá trình binding, 2 thiết bị được kết nối với nhau nếu cả 2 thiết bị có cùng Cluster ID, nhưng phải có 1 thiết bị là Client Cluster và thiết bị còn lại là Server Cluster.

(7) Attribute

Attribue (Thuộc tính) là các phần dữ liệu đại diện cho các đặc điểm hoặc trạng thái cụ thể của một thiết bị, ví dụ như trạng thái On/Off, nhiệt độ, độ sáng hoặc bất kỳ thông tin quan trọng nào khác.

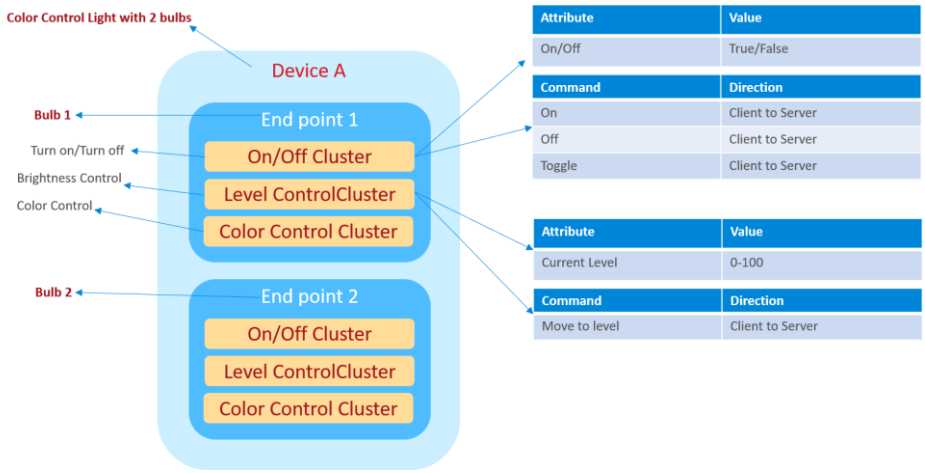
Trong ví dụ hệ thống điều khiển bóng đèn, thuộc tính "On/Off " được sử dụng để cho biết liệu thiết bị có đang bật hay tắt.

- Phía máy chủ (Server): thiết bị chứa các thuộc tính này, nó có trách nhiệm duy trì và cung cấp quyền truy cập đến các thuộc tính này. Trong ví dụ này, "đèn" là phía máy chủ vì nó chứa thuộc tính "On /Off ". Các thiết bị khác có thể đọc hoặc thay đổi giá trị của thuộc tính này bằng cách gửi các lệnh hoặc yêu cầu tương ứng.
- Phía khách (Client): có trách nhiệm tương tác với các thuộc tính trên phía máy chủ. Trong ví dụ của bạn, "công tắc" là phía khách hàng vì nó gửi các thông điệp hoặc lệnh đến "đèn" (phía máy chủ) để điều khiển trạng thái của nó. Ví dụ, công tắc gửi các lệnh On/Off để thay đổi giá trị của thuộc tính "On /Off" trên đèn.

	Server	Client
Attributes	On/Off	None
Received Commands	On Off Toggle	None
Generated Commands	None	On Off Toggle

Hình 2.6 Các thành phần của Attribue trong hệ thống điều khiển bóng đèn

Ví dụ về Cluster để chúng ta có thể hiểu rõ hơn về chúng. Giả sử chúng ta cần triển khai hệ thống chiếu sáng với 2 bóng đèn.



Hình 2.7 Attribute trong hệ thống điều khiển bóng đèn

Chúng ta có 2 EP trong thiết bị A, mỗi EP đại diện cho 1 bóng đèn. Đối với chức năng cơ bản như On/Off, chúng ta có thể sử dụng “On/Off” cluster. Đèn là

phía máy chủ (Server) và Switch là phía máy khách (Client). Có một Attribute "On/Off" được xác định ở phía máy chủ, cho biết đèn bật hay tắt. Ngoài ra còn có các Command như "On", "Off", "Toggle" được xác định và phải được gửi từ phía máy khách đến phía máy chủ. Nếu chúng ta muốn có nhiều chức năng hơn, chẳng hạn như chúng ta cần hỗ trợ kiểm soát độ sáng của đen. Chúng ta có thể sử dụng “Level Control” Cluster. Trong Cluster này, có một Attribute "Current Level" được xác định ở phía máy chủ, biểu thị độ sáng của bóng đèn. Ngoài ra còn có các Command như "Move to Level" được xác định và phải được gửi từ phía máy khách đến phía máy chủ. Và nếu chúng ta cần nhiều chức năng hơn nữa, chẳng hạn như cần hỗ trợ điều khiển màu, chúng ta có thể sử dụng “Color Control” Cluster.

(8) Binding

Là kỹ thuật để liên kết EP của node này với một hoặc nhiều EP của một node khác. Thông tin được trao đổi dưới dạng Cluster, để liên kết hai ứng dụng, chúng phải có các Cluster tương thích. Ví dụ: đối với hai ứng dụng ở các node khác nhau để kiểm soát nhiệt độ, một ứng dụng phải có khả năng tạo Cluster đầu ra liên quan đến nhiệt độ và ứng dụng kia phải có khả năng sử dụng nó làm cụm đầu vào.

Binding được lưu trữ trong 1 Binding table. Binding Table là 1 bảng lưu các EP, Address , Cluster IDs mà nó đã bind. Khi 1 EP trong node gửi dữ liệu ở chế độ “indirect mode”, thì nó tự động tra bảng Binding table để gửi tới tất cả các Endpoint mà nó binding tới.

Binding table chỉ được lưu trên Client EP (ví dụ công tắc bind tới bóng đèn thì binding table được lưu trên công tắc) và ở trên Coordinator.

Bảng 2.1 Binding Table

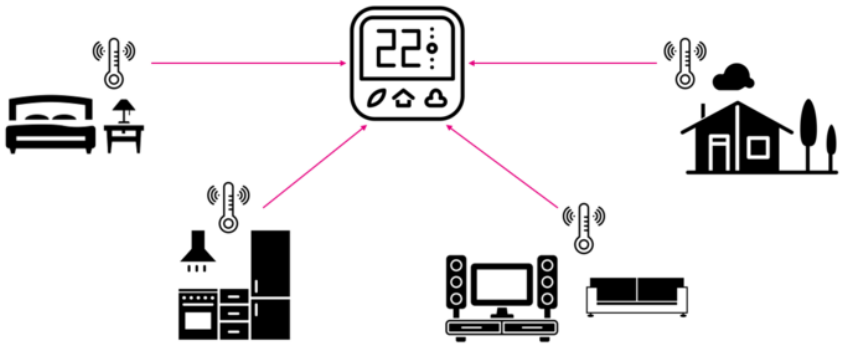
Src EP	**Destination Addr**	**Addr/Grp**	**Dst EP**	**Cluster ID**
5	0x1234	A	12	0x0006
6	0x796F	A	240	0x0006
5	0x9999	G	--	0x0006
5	0x5678	A	44	0x0006

Quá trình Binding**:** 2 node muốn binding với nhau cùng gửi tín hiệu binding request tới ZC. Trong bản tin chứa thông tin về EP in/out Cluster. Sau đó nó sẽ kiểm tra 2 thiết bị có tương thích hay không: cùng Cluster, 1 bên là Client, 1 bên là Server. Nếu tương thích nó sẽ lưu thông tin trên binding table.

Unbind: node gửi unbind request, sau đó nó sẽ xóa địa chỉ của node đã binding trên binding table.

(9) Reporting

Reporting là một khía cạnh quan trọng trong việc trao đổi dữ liệu. Ngoài việc gửi các lệnh (On, Off, Toggle), một số thiết bị muốn gửi trạng thái hoặc dữ liệu đo lường của họ. Ví dụ, một cảm biến DHT có thể muốn gửi định kỳ các giá trị nhiệt độ và độ ẩm đến Coordinator.



Hình 2.8 Ví dụ về Reporting

Có hai cách để gửi Reporting Message:

- Active Report: Các Report được cấu hình dựa trên một tham số gọi là ngưỡng (Threshold). Nếu giá trị thuộc tính đã chọn thay đổi vượt qua ngưỡng được định trước, nó sẽ gửi một thông điệp Report đến node được Binding. Report này được gửi tự động khi có sự thay đổi phù hợp hoặc có thể được yêu cầu thông qua giao diện dòng lệnh (CLI) bằng một node khác.
- Active Report: liên quan đến việc gửi một message chứa các giá trị của các thuộc tính đã chọn. Thông điệp này được nạp vào bộ đệm lệnh sau đó được gửi bằng cơ chế địa chỉ hóa. Để kích hoạt Active Report, chức năng "function" được sử dụng.

e) Thành lập mạng Zigbee

Mạng được thành lập bởi ZC và trải qua các bước:

Bước 1: Đặt địa chỉ EPID và Coordinator.

Đặt giá trị EPID 64bit, nếu giá trị này = 0 thì EPID = MAC address của ZC. Đặt giá trị Network Address của ZC = 0x0000

Bước 2: Chọn kênh

Thực hiện Energy Scan để tìm ra kênh ít nhiễu nhất, ít các thiết bị mạng hoạt động nhất. Quét 16 kênh hết 8s. Nếu muốn chọn kênh cụ thể để tham số apsChannelMask từ 11-26

Bước 3: Chọn PAN ID cho mạng

Khi chọn được kênh ở bước 2. ZC sẽ chọn địa chỉ PANID. Để làm được việc đó nó thực hiện Active scan (trên kênh được chọn nó gửi Beacon request để lắng nghe và xác định địa chỉ PANID của các mạng khác). Sau đó chọn random một PANID không trùng với các PANID của mạng khác trên kênh đó.

nwkPanID = 0xFFFF thì mạng sẽ tự chọn random, Ta có thể chọn trước 1 giá trị từ 0x0000 tới 0x3FFF

Bước 4: Nhận yêu cầu tham gia từ các thiết bị khác

Giờ ZC đã sẵn sàng để nhận lệnh Join request từ node khác (ZR, ZED).

f) Quá trình tham gia mạng Zigbee của các node

Bước 1: Khởi tạo quá trình gia nhập. Khi một thiết bị (End Device) muốn gia nhập vào mạng Zigbee, nó bắt đầu quá trình bằng cách thực hiện "Active Scan" thông qua việc gửi "Beacon Request" đến tất cả các mạng trên một kênh cụ thể. Điều này nhằm mục đích tìm kiếm một mạng có khả năng kết nối.

Bước 2: Phản hồi từ mạng: Các ZR/ZC hiện đang cho phép gia nhập (Permit Joining) sẽ phản hồi lại "Beacon Request" bằng "Beacon Response". Phản hồi này bao gồm thông tin quan trọng như PANID, EPANID, kênh truyền (channel), và thông tin về không gian trống cho thiết bị mới gia nhập.

Bước 3: Lựa chọn mạng: Chọn Mạng Phù Hợp: Thiết bị sẽ chọn mạng dựa trên PANID. Trong trường hợp thiết bị đã được cài đặt EPANID từ trước, nó sẽ kiểm tra EPANID để tìm kiếm sự trùng khớp. Nếu không có EPANID được cài đặt, thiết bị sẽ chọn "parent" gần nhất dựa trên thuật toán của Zigbee để thực hiện kết nối.

Bước 4: Gửi Yêu Cầu Gia Nhập. Sau khi đã chọn được mạng và parent phù hợp, thiết bị sẽ gửi "Join Request" đến parent đã chọn.

Bước 5: Quá trình xác thực và trao đổi key. Trong giai đoạn này, quá trình xác thực được thực hiện và key bảo mật được trao đổi giữa thiết bị và mạng để đảm bảo kết nối an toàn.

Bước 6: Gia nhập vào bất kỳ mạng nào. Trường hợp đặc biệt, nếu nwkPANID được cài đặt là 0xFFFF, thiết bị có khả năng gia nhập vào bất kỳ mạng Zigbee nào có sẵn.

Với ZED khi bị mất kết nối với parent node: Gửi beacon request tìm parent mới → Nhận response từ ZC/ZR → chọn một parent thích hợp → Thông báo tới mạng (Device annouce) mình đã thay đổi (nhận PANID mới).

Với ZR khi bị mất điện: Silent Rejoin. Không cần gửi bất kỳ request gì, vẫn cứ hoạt động như bình thường với PANID, EPANID, short addr, network key như cũ.

g)Quá trình rời mạng Zigbee của các node

Một node muốn dời khỏi mạng sẽ gửi leaving request tới tất cả các thiết bị trong mạng. Tất cả các thiết bị đó sẽ xóa tất cả thông tin về thiết bị dời mạng.

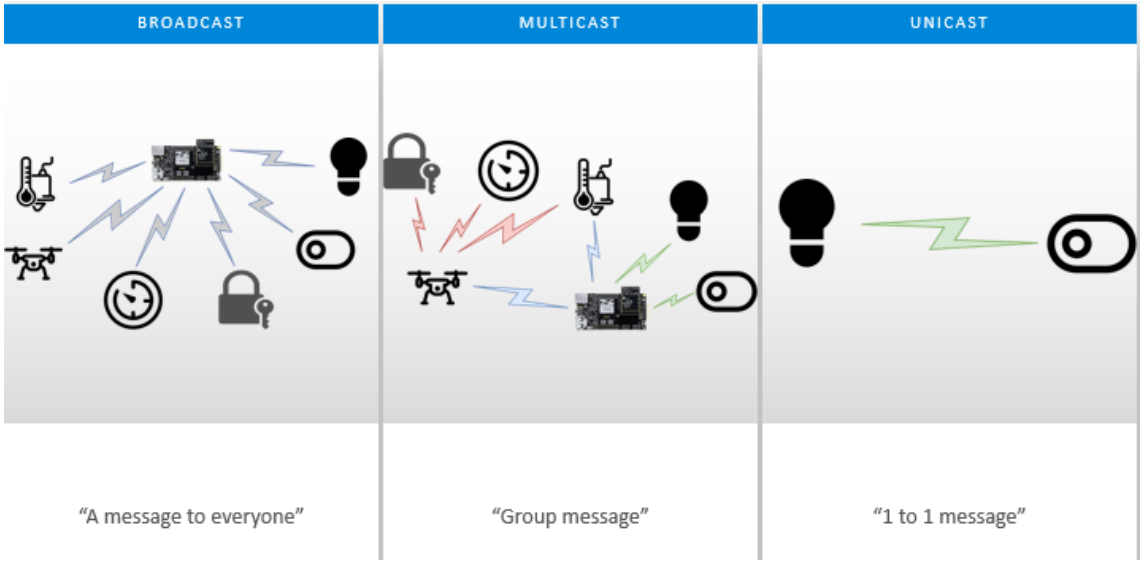
Khi một Node bị dời mạng do mất nguồn thì nó sẽ tự kết nối trở lại khi có nguồn.

Khi một node bị dời mạng do mất nguồn hoặc do sóng radio không tốt làm cho các thiết bị khác không thể giao tiếp với nó. Các thiết bị khác có thể nhận biết việc dời mạng này bằng bằng việc không nhận được bản tin ACK.

h) Các loại bản tin trong zigbee

-Có ba loại tin nhắn chính, bao gồm: Broadcast (Phát sóng), Multicast (Phát đa điểm) và Unicast (Phát đơn điểm).

A. Unicast: Truyền tin nhắn đến một thiết bị duy nhất trong mạng.
B. Multicast: Truyền tin nhắn đến mọi thiết bị trong một mạng cá nhân (PAN) thuộc về một nhóm multicast được định nghĩa động.
C. Broadcast: Truyền tin nhắn đến mọi thiết bị trong một mạng cá nhân (PAN) thuộc về một trong số ít nhóm phát sóng được định nghĩa tĩnh, ví dụ như tất cả các bộ định tuyến hoặc tất cả các nút.



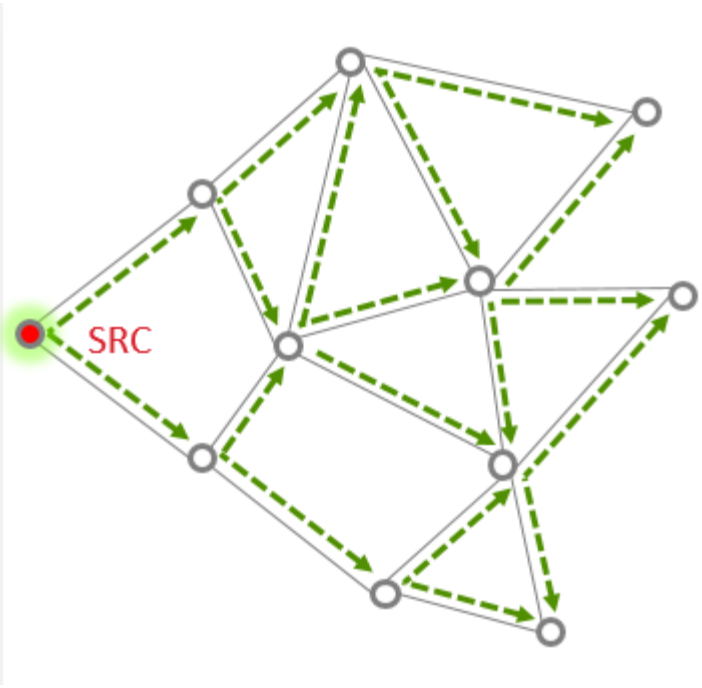
1) Unicast

Unicast là một tin nhắn được gửi từ một nút đến một nút khác. Ở mức cơ bản nhất, unicast chỉ là một tin nhắn được gửi giữa hai nút liền kề.

Tuy nhiên, chúng có thể phức tạp hơn nhiều, được định tuyến qua 1 hoặc nhiều nút bổ sung trong mạng.

Tin nhắn unicast có thể được gửi bằng API emberAfSendUnicast()

2) Broadcast



Trong Broadcast, nút nguồn gửi tin nhắn đến tất cả các nút trong phạm vi một bước nhảy của nó và các nút này lặp lại tin nhắn đến tất cả các nút trong phạm vi một bước nhảy của chúng cho đến khi tin nhắn đến được tất cả các nút trong một bán kính nhất định. Trong một mạng lớn, việc mong đợi tất cả các thiết bị nhận được tin nhắn broadcast gửi một xác nhận trở lại cho nguồn gốc tin nhắn sẽ khó khăn và không cần thiết. Thay vào đó, một tin nhắn phát lại là một chỉ báo rằng một thiết bị lân cận đã nhận và phát lại tin nhắn broadcast thành công. Nút nguồn xác minh liệu tất cả các thiết bị lân cận đã phát lại tin nhắn thành công chưa. Điều này được gọi là cơ chế xác nhận thụ động. Mỗi lần một tin nhắn broadcast được lặp lại, trường bán kính được giảm xuống và các tin nhắn broadcast với trường bán kính bằng 0 sẽ không được lặp lại nữa. Các tin nhắn broadcast được làm trễ ngẫu nhiên để giảm khả năng va chạm.

Các bộ định tuyến duy trì hồ sơ của tất cả các tin nhắn mà chúng phát lại trong một bảng gọi là bảng giao dịch broadcast (BTT).

ĐỊA CHỈ BROADCAST NHÓM ĐÍCH

0xFFFF	Tất cả các thiết bị trong PAN
0xFFFD	Tất cả các thiết bị không ngủ
0xFFFC	Tất cả các thiết bị có khả năng định tuyến

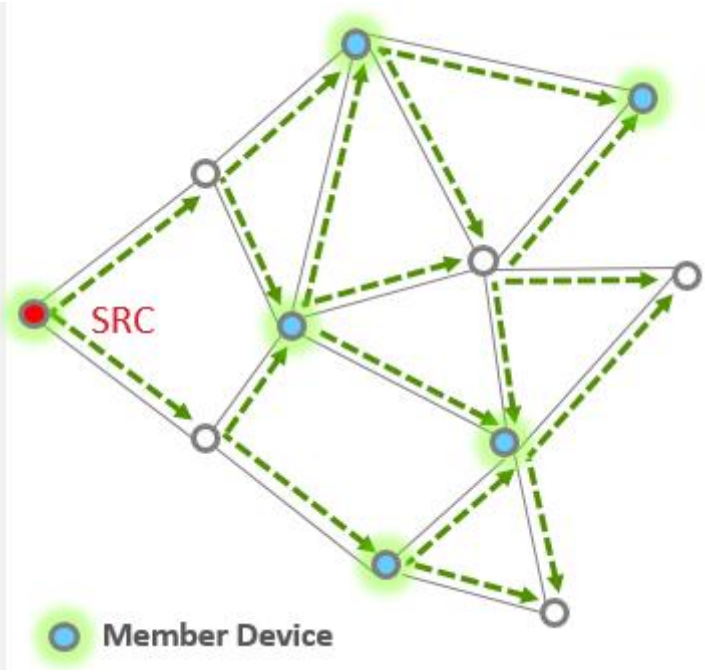
Hồ sơ này được gọi là bản ghi giao dịch broadcast (BTR) và chứa số thứ tự và địa chỉ nguồn của khung broadcast, giúp trong việc phát lại các tin nhắn broadcast. Do đó, trong một khoảng thời gian nhất định, số lượng broadcast bị giới hạn bởi kích thước bảng giao dịch broadcast. Trong EmberZnet, kích thước bảng broadcast có thể được cấu hình trong Zigbee Pro Stack Component.

Mặc dù tin nhắn broadcast nghe có vẻ đơn giản, nhưng nó có thể làm tăng đáng kể lưu lượng trong mạng, gây tắc nghẽn và giảm hiệu suất. Chúng nên được sử dụng một cách tiết kiệm và cẩn thận.

Các tùy chọn broadcast ở mức mạng (địa chỉ broadcast) tồn tại để gửi tin nhắn đến chỉ các bộ định tuyến, đến tất cả các nút không ngủ bao gồm các thiết bị cuối, hoặc cũng để gửi đến các thiết bị cuối đang ngủ. Các thiết bị cuối gửi unicast tin nhắn broadcast của chúng đến nút cha của chúng, nút này sau đó truyền bá tin nhắn qua mạng thay mặt chúng.

Tin nhắn broadcast có thể được gửi bằng API emberAfSendBroadcast()

3) Multicast



Trong multicast, tin nhắn được gửi đến một nhóm các thiết bị trong cùng một mạng, đây là một loại broadcast có lọc giới hạn. Hình dưới đây minh họa rằng chỉ các thiết bị thành viên mới có thể nhận tin nhắn multicast ở lớp ứng dụng.

Nút nguồn không cần phải là thành viên của một nhóm multicast để có thể sử dụng multicast để liên lạc với các thành viên. Các tin nhắn multicast có thể được khởi tạo bởi các thiết bị cuối nhưng không được gửi đến các thiết bị mà macRxOnWhenIdle bằng FALSE.

Mỗi nhóm được xác định bởi một ID nhóm multicast 16-bit. Các thiết bị trong cùng một nhóm được gọi là các thành viên nhóm. Một thiết bị có thể là thành viên của nhiều nhóm multicast. Mỗi thiết bị giữ danh sách các nhóm multicast của mình trong một bảng gọi là bảng nhóm/multicast.

Các nhóm thường được quản lý qua ZCL Groups cluster (xem các plugin Groups client & server và Zigbee cluster library Specification để biết thêm thông tin). Bảng trên cho thấy các lệnh được hỗ trợ bởi Group cluster.

Tin nhắn multicast có thể được gửi bằng API emberAfSendMulticast()

SOFTWARE

Giới thiệu

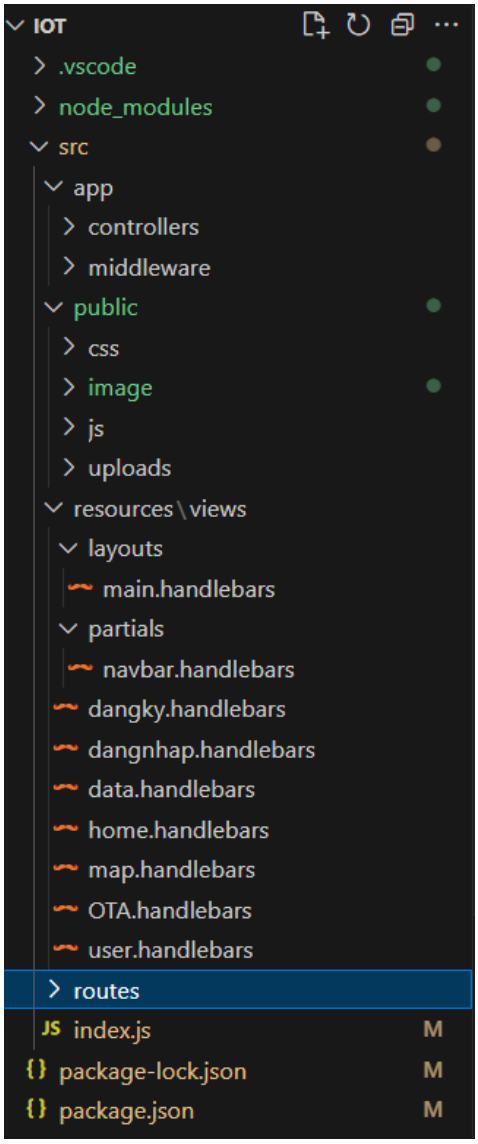
Mục tiêu xây dựng trang web bao gồm :

- Xử lý yêu cầu HTTP và routing sử dụng Express.
- Quản lý phiên làm việc (session) của người dùng sử dụng Express Session.
- Sử dụng Express Handlebars để quản lý giao diện người dùng.
- Tích hợp Firebase để quản lý cơ sở dữ liệu và xác thực người dùng.
- Sử dụng npm để quản lý các gói và dependencies.

Công nghệ sử dụng

- NodeJS: là nền tảng runtime cho JavaScript, cho phép thực thi mã JavaScript ở phía server. Phiên bản sử dụng là 20.12.1. Ưu điểm của NodeJS:
    + Hiệu suất cao
    + Khả năng mở rộng tốt: dễ dàng mở rộng thêm bằng các module hay gói npm
    + Cộng đồng lớn với nhiều thư viện và framework có sẵn
    + Dễ dàng tích hợp với các công cụ khác như cơ sở dữ liệu (SQL sever, MongoDB hay Firebase)
- NPM: là trình quản lý gói cho Node.js, cho phép cài đặt và quản lý các thư viện và module. Phiên bản sử dụng là 10.5.0.
- Express: là một framework web nhẹ và mạnh mẽ cho Node.js, giúp xây dựng các ứng dụng web và API. Phiên bản sử dụng là 4.19.2. Ưu điểm:
    + Khung web nhẹ, linh hoạt
    + Middleware mạnh mẽ
    + Hệ thống route đơn giản
    + Hỗ trợ templating engine như handlebars
- Express Handlebars: là công cụ templating cho Express, giúp tạo ra các trang HTML động. Phiên bản sử dụng là 7.1.2. Ưu điểm:
    + Ngôn ngữ templating đơn giản
    + Cho phép sử dụng lại các thành phần đã viết : cho phép tạo các partials và helpers, giúp tái sử dụng các phần của giao diện và giữ cho mã nguồn gọn gàng và dễ quản lý. Tuy nhiên cũng không được mạnh mẽ như các Framework chuyên về Front end khác như React
- Firebase: là nền tảng phát triển ứng dụng của Google, cung cấp các dịch vụ như cơ sở dữ liệu, xác thực, và lưu trữ. Phiên bản sử dụng là 10.11.0.
    + Được sử dụng để lưu data của các thùng rác và thông tin đăng nhập của các người dùng.
    + Các công cụ sử dụng trong firebase là Firestore database và realtime database
- Express Session là middleware cho Express, giúp quản lý phiên làm việc của người dùng. Phiên bản sử dụng là 1.18.0.

Cấu trúc thư mục



- Bao gồm thư mục src (source) dùng để quản lý mã nguồn viết trang web, thư mục node_modules để quản lý các gói thư viện được sử dụng và 2 file json có vai trò quan trọng trong việc quản lý các dependencies và thông tin dự án.
- Thư mục src: bao gồm các thư mục con khác :
    + app: lưu các file controller để tạo các hàm điều khiển cho từng trang, file middleware chứa các hàm để kiểm tra hoặc xử lý giữa 2 thao tác nếu cần thiết
    + public: dùng để lưu css, các image và các file khác
    + resource/views: nơi để lưu các file handlebars là các file HTML để tạo FE
    + routes: nơi cấu hình các route của trang web
    + File index.js là trang chính để có thể bắt đầu chạy sever
- 2 file json:
    + package-log.json: à tệp tự động được tạo ra khi bạn cài đặt các dependencies bằng npm (ví dụ: npm install). Tệp này lưu lại chính

xác phiên bản của từng gói đã được cài đặt, bao gồm cả các gói con (sub-dependencies). Điều này giúp đảm bảo rằng môi trường phát triển của bạn và của các cộng tác viên trên dự án là nhất quán.

+ package.json: là tệp cấu hình chính cho dự án Node.js của bạn. Nó chứa các thông tin về dự án như tên, phiên bản, tác giả, và quan trọng nhất là danh sách các dependencies mà dự án cần.

Các bước cấu hình sever và các route

- Bước 1: tại file index.js bên ngoài cấu hình sever bằng handlebars với session

```
app1.use(session({
  secret: 'secret_key',
  resave: false,
  saveUninitialized: true
}));

// cau hinh path
app1.use(express.static(path.join(__dirname,'public')));

// Cấu hình handlebars
app1.engine('handlebars', engine());
app1.set('view engine', 'handlebars');
app1.set('views',path.join(__dirname,'resources/views/'));
```

- Bước 2: Sau đó gọi hàm route(app1) được export từ routes/index.js trong đó route(app1) là hàm để cấu hình các route đầu tiên

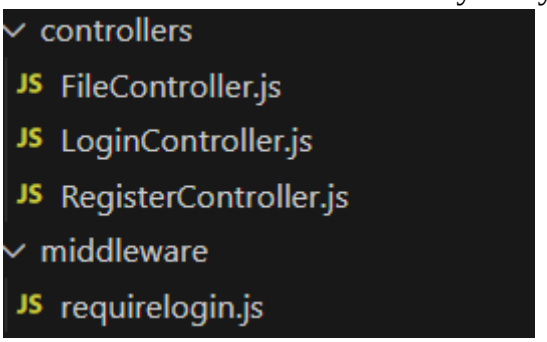
```
const LoginRouter = require('./login');
const RegisterRouter = require('./register');
const FileRouter = require('./file')
function route(app1){

    app1.get('/',(req ,res) =>{
      res.render('home');
    });
    app1.use('/login', LoginRouter);
    app1.use('/register',RegisterRouter);
    app1.use('/file',FileRouter)
}
```

- Bước 3: đối với mỗi route sâu hơn trong mỗi file sẽ được cấu hình trong cái file js chỉ riêng ra cho từng route

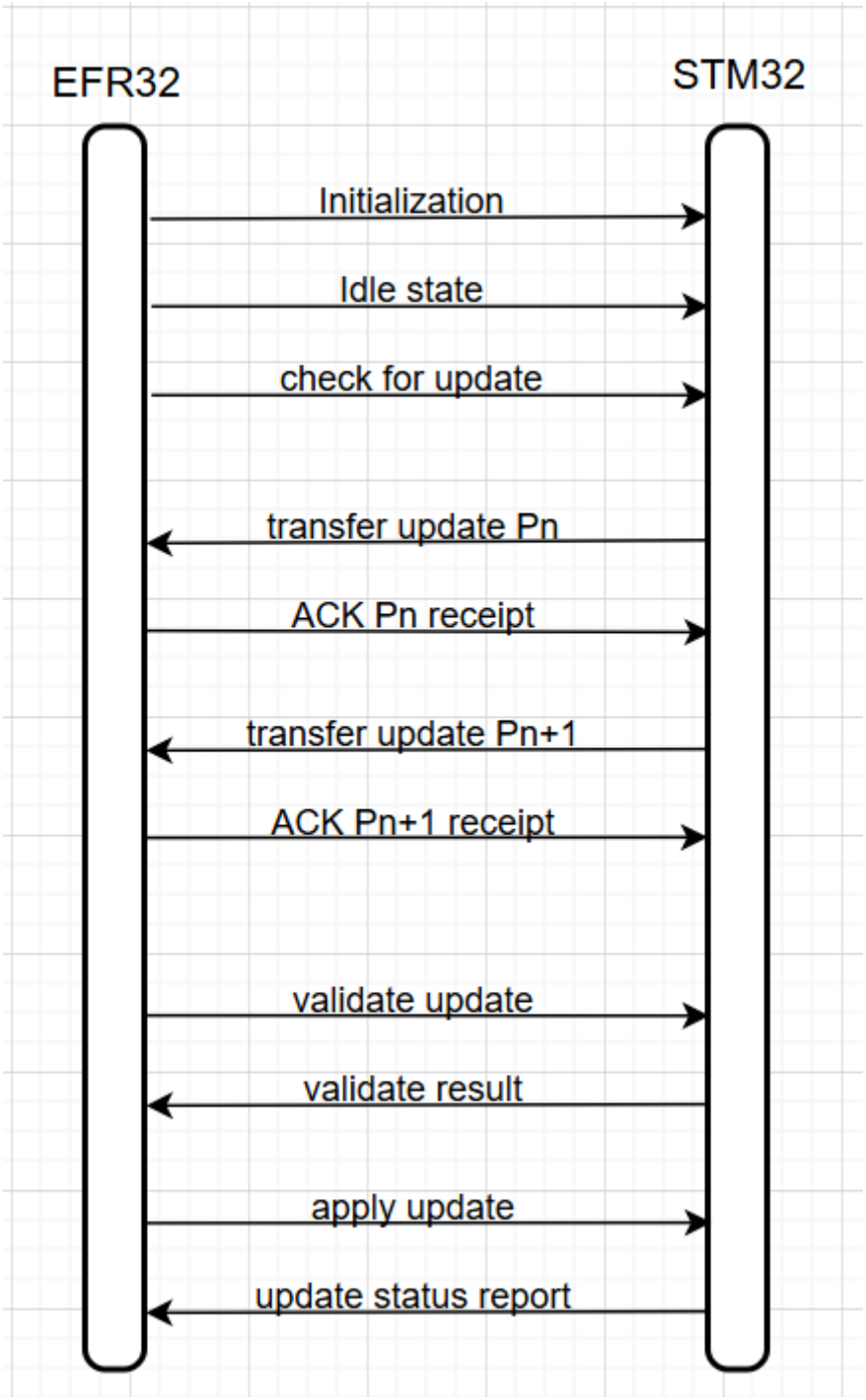
```
const FileController = require('../app/controllers/FileController');

router.get('/',FileController.getdata);
router.post('/',FileController.uploadMiddleware, FileController.pushfile);
```

```
const loginController = require('../app/controllers/LoginController');
const requireLogin = require('../app/middleware/requirelogin');


router.get('/user',requireLogin,loginController.renderuser);
router.get('/admin',requireLogin,loginController.renderadmin);
router.get('/getFirebaseData', loginController.getFirebaseData);
router.post('/updateLights', loginController.updateLights);
router.get('/',loginController.index);
router.post('/',loginController.check);
const registerController = require('../app/controllers/RegisterController');

router.get('/',registerController.index);
router.post('/',registerController.check)
```

- Bước 4: các hàm được dùng trong từ route sẽ được viết trong các file controller và middleware để xử lý các yêu cầu riêng biệt



Quá trình flash firmware từ EFR32 sang STM32



- Bước 1: sau khi khởi tạo hệ thống và EFR32 nhận được bản cập nhật lưu trữ của mình, EFR32 chia bản cập nhật thành các gói để và lần lượt truyền gia qiao tiếp USART với STM32
- Bước 2: Sau khi nhận đủ dữ liệu, STM32 kiểm tra dữ liệu có toàn vẹn không,nếu hợp lệ sẽ tiến hành cập nhật vào bộ nhớ flash
- Bước 3: khởi động lại STM32 và chạy thử firmware mới

Ở trong bootloader của STM32 sẽ nhận được như

Hàm xử lí lựa chọn nhảy qua vùng Application hoặc tiến hành OTA :

```
int main()
{
```

```
    clock_config();
    log_init();
    if(systick_init() < 0)
        loges("systick config error");

    registerAppMainCallBack(processAppMain, processAppOta);

    uint32_t timerTick = 0;
    uint32_t rstReason = 0;
    rstReason = get_rstReason();
    logi("rstReason : %d", rstReason);

    while (1)
    {
        if((uint32_t)rstReason & RCC_RESET_FLAG_PIN)
        {
            if((uint32_t)rstReason & RCC_RESET_FLAG_POR)
            {
                if(pAppMainCb[APP_MAIN] != NULL)
                    pAppMainCb[APP_MAIN]();
            }
            else
            {
                if(pAppMainCb[APP_OTA] != NULL)
                    pAppMainCb[APP_OTA]();
            }

        }
        if((uint32_t)(getSysTicks() - timerTick) == 3000)
        {
            logi("Timer Event !!!");
            timerTick = getSysTicks();
        }

    }

}
```

Chương trình sẽ đọc nguyên nhân reset của chip, nếu :

- Nguyên nhân Reset là Reset cứng, tức là reset do mất nguồn điện thì chương trình sẽ nhảy đến processAppMain để xử lí.
- Nguyên nhân Reset là reset mềm, do chip ESP8266 kích xung reset vào chân RST của STM32 thì chương trình sẽ nhảy đến processAppOta để xử lí.

Hàm xử lí chương trình processAppMain nhảy Application :

```
int processAppMain()
{
```

```
    logw("check appmain");
    appMainAddr = START_ADDRESS; // Gán địa chỉ bắt đầu của vùng
application main

    /* Tắt hết ngoại vi và clear cờ ngắt, ngắt pending */
    disable_peripheral();
    disable_interrupt();

    /*
        * (volatile uint32_t*)(appMainAddr + 4) là trỏ đến vùng nhớ của
Reset Handler của vùng app Main
        * *(volatile uint32_t*)(appMainAddr + 4) là lấy địa chỉ bắt đầu
của Reset Handler của vùng app Main
        * Cuối cùng là gán cho biến appMainRstHdlAddr lưu địa chỉ bắt đầu
của Reset Handler
        * Địa chỉ của Reset Handler pointer (0x08002004) khác với địa chỉ
bắt đầu của hàm Reset Handler (do Linker lúc conpile quy định)
    */

    appMainRstHdlAddr = *((volatile uint32_t*)(appMainAddr + 4));

    /* Ép kiểu về con trỏ để gán cho resetHandler pointer */
    void(*resetHandler)(void) = (void*)appMainRstHdlAddr;

    /* Set Main Stack Pointer vào địa chỉ đầu tiên của vùng App Main */
    __set_MSP(*((volatile uint32_t*)appMainAddr));

    /* Reset các giá trị của Systick */
    SysTick->CTRL = 0;
    SysTick->LOAD = 0;
    SysTick->VAL = 0;

    /* Cuối cùng gọi Reset Handler để chạy hàm main() của App Main */
    resetHandler();


    while (1)
    {
        logws("check while1 appmain");
        delay(1000);
        /* code */
    }

}
```

Để nhảy đến thực thi chương trình ở vùng khác trong bộ nhớ thì cần phải thực hiện tuần tự các bước như sau :

- Bước 1 : Chọn địa chỉ bắt đầu cho vùng Application.

- Bước 2: Tắt hết ngoại vi đang chạy và xóa hết các cờ ngắt, các ngắt đang pending.
- Bước 3: Chọn địa chỉ Reset Handler của Application bằng các cộng thêm 4 vào địa chỉ bắt đầu của vùng.
- Bước 4: Khởi tạo Main Stack Pointer vào địa chỉ đầu tiên của Application.
- Bước 5: Reset các giá trị của Systick về 0.
- Bước 6: Gọi Reset Handler đã chọn ở bước 3 để nhảy đến chương trình Application.

Hàm xử lí chương trình processAppOta để tiến hành OTA :

```
int processAppOta()
{
    logw("check appota");
    uart_init();
    uart_register_callback(onDataProcess);
    writeAddress = START_ADDRESS;
    pageData = malloc(PAGE_SIZE);

    uint32_t otaTick = 0;

    while (1)
    {
        processUartEvents();

        if((uint32_t)(getSysTicks() - otaTick) == 10000)
        {
            logi("OTA mode !!!");
            otaTick = getSysTicks();
        }
    }

}
```

Hàm này sẽ tiến này thiết lập các giao thức Uart được quy định ở phần sau, và gọi ra các hàm xử lí tương ứng với các gói giao thức gửi đến từ khối truyền thông.

```
static void onDataProcess(uart_data_t* data)
{
    switch (data->type)
    {
    case START_UPDATE_REQ:
        processStartUpdate(data);
        return;
    case SEND_DATA_REQ:
        processRecvBlockData(data);
        return;
```

```
    case STOP_UPDATE_REQ:
        processStopUpdate(data);
        return;
    case CHECK_BLOCK_MASK:
        processFlashData(data);
        return;
    default:
        uart_sendBuffer("Unknown", 7);
        return;
    }
}
```

Các gói giao thức như là gói yêu cầu bắt đầu quá trình Update, Gói nhận Data, Gói yêu cầu dừng Update, Gói yêu cầu check các block dữ liệu bị lỗi.

KẾT LUẬN

Kết luận

Nhóm đã thực hiện được mục tiêu cơ bản đề ra đó là thùng rác tự động có các cảm biến thu thập dữ liệu và có phần mềm cho người quản lý. Đã sử dụng được Zigbee và HTTP trong việc truyền nhận dữ liệu.

Tuy nhiên do các thành viên có chương trình học nặng và nhiều công việc trong lab nghiên cứu vì vậy thời gian đầu tư cho đề tài chưa cao và đều đặn vì thế các nội dung mở rộng về OTA và AI detect chưa được hoàn thiện và trình bày trong báo cáo này, thời gian viết báo cáo không nhiều nên báo cáo còn sơ sài chưa nêu bật được hết các nội dung thực hiện. Nhóm em đang tập trung cao độ trong những ngày cuối nên sẽ cố gắng có thêm update mới và hoàn thiện hơn để trình bày trong ngày Chung kết.

Hướng phát triển trong tương lai

- Hoàn thiện tính năng gỡ lỗi, cập nhật chương trình từ xa FOTA
- Kết hợp với các robot tự hành, nhặt rác, thu rác tự động, tối ưu quãng đường, chu kì thu thập rác thải, giảm áp lực giao thông và tiêu thụ năng lượng
- Kết hợp kịch bản cho người dùng như: yêu cầu thu rác gấp, tích điểm, ….
- Xây dựng mô hình AI nhẹ, triển khai trên chip để có thể phân loại được rác tái chế, hữu cơ, vô cơ, kim loại vào các ngăn
- Tối ưu chu kì hoạt động với các kịch bản kiểm soát năng lượng