TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN ĐIỆN

BỘ MÔN KỸ THUẬT ĐO VÀ TIN HỌC CÔNG NGHIỆP

.....***.....

ĐỒ ÁN TỐT NGHIỆP

(HỆ ĐẠI HỌC CHÍNH QUY)

Đề tài:

ỨNG DỤNG MẠNG CẢM BIẾN BLUETOOTH LOW ENERGY GIÁM SÁT THIẾT BỊ TRONG NHÀ

Sinh viên thực hiện : Ngô Vũ Trường Giang

MSSV : 20131098

Lớp : KTĐK&TĐH 04 - K58

Giảng viên hướng dẫn : PGS. TSKH. Trần Hoài Linh

Hà Nội – 2018

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

NHIỆM VỤ THIẾT KẾ TỐT NGHIỆP

Họ và tên: *Ngô Vũ Trường Giang* Mã số sinh viên: *20131098*

Khóa: *K58*

Viện: ***Điện***

Ngành: ***Kỹ thuật đo và tin học công nghiệp***

1. Đầu đề thiết kế/Tên đề tài

Ứng dụng mạng cảm biến Bluetooth Low Energy giám sát thiết bị trong nhà

2. Các số liệu ban đầu

- Các thiết bị truyền thông qua mạng Bluetooth Low Energy.
- Điều khiển thông qua Smart phone và Web.

3. Các nội dung tính toán, thiết kế

- Lập trình điều khiển các thiết bị trong nhà qua mạng Bluetooth Low Energy.
- Thiết kế mạch điều khiển, mạch công suất.
- Xây dựng Web điều khiển và ứng dụng điện thoại.

4. Cán bộ hướng dẫn: *PGS. TSKH. Trần Hoài Linh*

5. Ngày giao nhiệm vụ thiết kế:

6. Ngày hoàn thành nhiệm vụ:

Ngày..... tháng..... năm.....

TRƯỞNG BỘ MÔN
(Ký, ghi rõ họ tên)

CÁN BỘ HƯỚNG DẪN
(Ký, ghi rõ họ tên)

SINH VIÊN THỰC HIỆN
(Ký, ghi rõ họ tên)

LỜI CAM ĐOAN

Em xin cam đoan bản đồ án tốt nghiệp: "**ỨNG DỤNG MẠNG CẢM BIẾN BLUETOOTH LOW ENERGY GIÁM SÁT THIẾT BỊ TRONG NHÀ**" do em tự thiết kế dưới sự hướng dẫn của thầy giáo PGS. TSKH. Trần Hoài Linh. Các số liệu và kết quả hoàn toàn đúng với thực tế. Để hoàn thành đồ án này em sử dụng những tài liệu ghi trong danh mục tài liệu tham khảo và không sao chép hay sử dụng bất kì tài liệu nào khác. Nếu phát hiện có sự sao chép em xin hoàn toàn chịu trách nhiệm.

Sinh viên thực hiện.

(Kí và ghi rõ họ tên)

Ngô Vũ Trường Giang

MỤC LỤC

CÁC THUẬT NGỮ VIẾT TẮT

BLE	Bluetooth Low Energy
BR	Bluetooth Basic Rate
SDK	Software Development Kit
GFSK	Gaussian Frequency Shift Keying
ISM	Industrial, Scientific and Medical
RF	Radio frequency
NFC	Near-Field Communications
PDU	Protocol Data Unit
SDU	Service Data Unit
MTU	Maximum Transmission Unit
SIG	Special Interest Group
ATT	Attribute Protocol
MIC	Message Integrity Check
CRC	Cyclic Redundancy Check
NESN	Next Expected Sequence Number
SN	Sequence Number
RSSI	Received Signal Strength Indication
UUID	Universal Unique Identifier
API	Application Programming Interface
MQTT	Message Queueing Telemetry Transport
SoC	System on Chip
UART	Universal Asynchronous Receiver Transmitter
IEEE	Institute of Electrical and Electronics Engineers
AES-CCM	Advanced Encryption Standard – Counter with CBC MAC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

LỜI NÓI ĐẦU

Ngày nay, khoa học kĩ thuật phát triển nhanh một cách chóng mặt, robot ngày càng hoàn thiện cùng với sự ra đời của trí thông minh nhân tạo đã có thể thay thế con người trong rất nhiều công việc đòi hỏi sự phức tạp trong cuộc sống. Chúng ta có thể nhìn thấy được một viễn cảnh mọi đồ vật xung quanh con người đều sẽ được kết nối Internet, đó chính là lý do hiện nay đang bùng nổ một xu thế mới mang tên IoT.

Thời đại con người có thể tối giản hóa mọi hoạt động chân tay không còn xa nữa, giờ đây việc điều khiển các thiết bị trong gia đình thông qua các thiết bị cầm tay như smart phone, tablets, laptop… không còn là điều gì đó quá phức tạp. Tuy nhiên để một hệ thống điều khiển không dây hoạt động được lâu dài và ổn định ngay cả khi không có Internet thì chúng ta cần giải quyết thêm các vấn đề về năng lượng cũng như các phương án điều khiển khi offline. Chính vì vậy, em quyết định lựa chọn đề tài "*Ứng dụng mạng cảm biến Bluetooth Low Energy giám sát thiết bị trong nhà*" khi mà hầu hết các mạng không dây khác đều không thể tương thích với điện thoại hay laptop thì Bluetooth lại là một giải pháp vô cùng hợp lý khi mà nó được trang bị lên hầu hết mọi thiết bị di động hiện nay.

Trong quá trình thực hiện đề tài này, em dù đã cố gắng tìm hiểu về mạng Bluetooth Low Energy và việc sử dụng nó để tạo thành mạng điều khiển, tuy nhiên do thời gian tìm hiểu còn hạn chế cũng như các công nghệ hiện nay còn khá mới mẻ, cộng đồng và tài liệu còn ít, cùng với khả năng có hạn của bản thân nên đề tài vẫn còn nhiều sai sót và chưa hoàn hảo như dự kiến. Em rất mong nhận được ý kiến góp ý của các quý thầy cô để đề tài có thể mở rộng và hoàn thiện hơn về sau này.

Cuối cùng, em muốn bày tỏ lòng cảm ơn và sự kính trọng đến thầy Trần Hoài Linh, người đã luôn theo sát em trong những tháng vừa qua, không chỉ dạy em những kiến thức chuyên môn quý báu mà còn cho em những lời khuyên và bài học vô giá trong cuộc sống.

Năm năm học Bách Khoa, chặng đường này cuối cùng cũng kết thúc. Xin cảm ơn.

Sinh viên

Ngô Vũ Trường Giang

CHƯƠNG 1: GIỚI THIỆU CHUNG

1.1. ĐẶT VẤN ĐỀ

Xã hội hiện nay ngày càng phát triển, vì đó là xu hướng tất yếu, xã hội không phát triển thì con người chững lại, con người chững lại thì sẽ bị đào thải. Mà xã hội muốn phát triển thì khoa học kĩ thuật phải phát triển trước tiên. Vì vậy mà qua các cuộc cách mạng về công nghiệp 1.0, 2.0 rồi 3.0 cuối cùng Cách mạng công nghiệp 4.0 cũng bùng nổ, đánh dấu tốc độ đột phá "không có tiền lệ lịch sử", cuộc cách mạng về kỹ thuật số không dây. Những cốt lõi của kỹ thuật số sẽ là Trí tuệ nhân tạo (AI), Vạn vật kết nối - Internet of Things (IoT) và Dữ liệu lớn (Big Data). Vì vậy nhu cầu thiết yếu hàng ngày của con người về các thiết bị chăm sóc sức khỏe thông minh, ngôi nhà thông minh cũng sẽ dần trở thành xu hướng trong thời gian không xa nữa. Để có thể kết hợp tốt được cả hai nhu cầu đó thành một khối đồng bộ duy nhất là điều không hề dễ. Trong bối cảnh đó, công nghệ không dây Bluetooth nổi lên như một giải pháp phù hợp và Bluetooth Low Energy (BLE) được sinh ra với sứ mệnh phục vụ riêng cho các ứng dụng IoT cần tiêu thụ ít năng lượng mà vẫn tương tác tốt với những gì Bluetooth Classic để lại.

1.2. MỤC TIÊU NGHIÊN CỨU CỦA ĐỒ ÁN

- Tìm hiểu về công nghệ BLE.
- Lập trình ứng dụng sử dụng công nghệ BLE điều khiển thiết bị.
- Tạo được mạng cảm biến BLE có thể áp dụng được thực tế và điều khiển được bằng điện thoại.

1.3. KẾ HOẠCH THỰC HIỆN

- Tìm hiểu lý thuyết về BLE.
- Tìm hiểu sử dụng gói SDK của nhà sản xuất.
- Lập trình được chip sử dụng BLE.
- Thiết lập được mạng cảm biến chứa các thiết bị BLE.
- Lập trình điều khiển được bằng điện thoại.

1.4. Ý NGHĨA ĐỀ TÀI

Áp dụng được vào điều khiển các thiết bị trong nhà thông qua điện thoại hoặc máy tính. Xa hơn có thể đồng bộ với các thiết bị theo dõi sức khỏe có sẵn trên thị trường để giám sát tốt hơn, mang mọi thứ trở nên dễ dàng để tiếp cận.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VỀ BLE VÀ GIAO THỨC MQTT

2.1. TỔNG QUAN VỀ BLE

2.1.1. Công nghệ Bluetooth là gì?

Công nghệ không dây Bluetooth là một hệ thống truyền thông dải ngắn với mục đích thay thế các sợi cáp cầm tay hoặc các thiết bị điện cố định. Những đặc điểm chính của công nghệ Bluetooth là kết nối mạnh, năng lượng tiêu thụ thấp và giá thành rẻ. Có 2 mô hình hệ thống Bluetooth là Basic Rate (BR) và Low Energy(LE).

2.1.2. Đặc tính của BLE

Hệ thống LE bao gồm các đặc tính được thiết kế để cho phép các sản phẩm có được mức tiêu thụ năng lượng thấp hơn, ít phức tạp hơn và giá thành rẻ hơn hệ thống BR. Hệ thống LE cũng được thiết kế dùng trong các trường hợp và ứng dụng cần tốc độ truyền dữ liệu thấp và sử dụng ít năng lượng, cho phép các thiết bị hoạt động từ vài tháng hoặc vài năm chỉ bằng một viên pin nhỏ mà không cần sạc lại hoặc thay thế. Hiện nay có rất nhiều ứng dụng điển hình của BLE như theo dõi sức khỏe, beacon, nhà thông minh, an ninh, giải trí, cảm biến… BLE được phát triển từ nền Bluetooth cho nên nó tương thích với hầu hết các thiết bị hiện này như Smart phone, tablets và PCs.

2.1.3. Special Interest Group Bluetooth (SIG) là gì?

Đây là một tổ chức cá nhân phi lợi nhuận. SIG không làm, sản xuất hay bán các sản phẩm Bluetooth. Nhiệm vụ chính của SIG là:

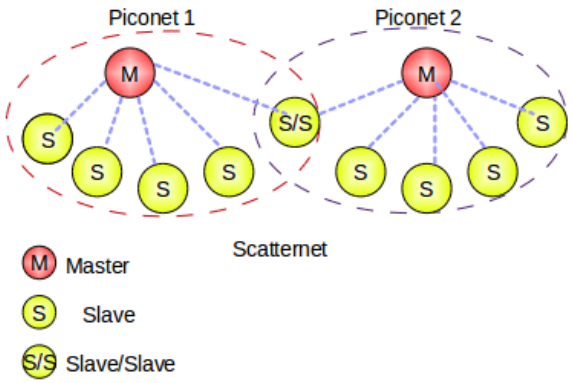
- Phát hành các đặc điểm kĩ thuật của Bluetooth.
- Người quản lý các chương trình tiêu chuẩn của Bluetooth.
- Truyền bá công nghệ không dây Bluetooth.

2.1.4. So sánh Bluetooth Classic (BR) và BLE

Công nghệ BLE được ra đời bắt đầu từ Bluetooth 4.0. Kể từ trước Bluetooth 4.0 thì các phiên bản Bluetooth là Bluetooth Classic.

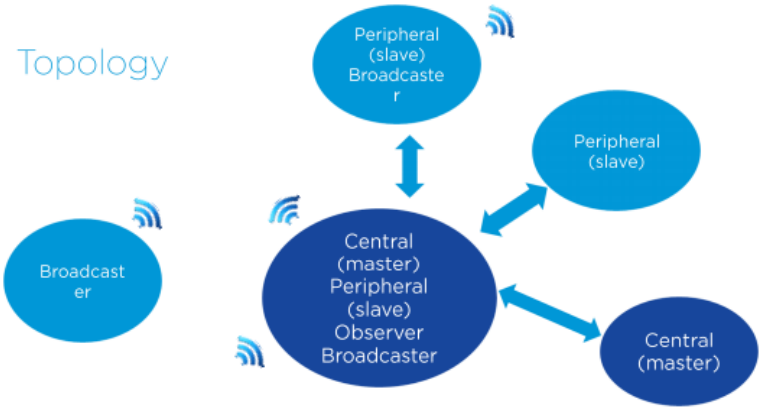
- **Kênh vật lý**: Bluetooth Classic và BLE đều làm việc ở tần số 2.4Ghz, trong đó BR có 80 kênh vật lý (0-79) để truyền dữ liệu nằm trong dải 2400-2483.5Mhz. BLE có 40 kênh vật lý (0-39) nằm trong dải 2400-2483.5 Mhz trong đó 3 kênh 37,38,39 là 3 kênh quảng cáo.

- **Năng lượng tiêu thụ:** BLE tiêu thụ năng lượng ít hơn hẳn so với BR (~1-50% BR), đây cũng là một đặc tính quan trọng của BLE, người dùng có thể sử dụng 1 viên pin nhỏ cho thiết bị BLE hoạt động trong nhiều tháng đến vài năm.
- **Cấu trúc mạng:** BR hỗ trợ Piconet, nó có cấu trúc mạng sao, ngoài ra nó còn hỗ trợ Scatternet (1 nhóm các Piconet). Trong chế độ BR, thiết bị sẽ chỉ hoạt động ở 1 vai trò (hoặc là Central hoặc là Peripheral) và để gửi dữ liệu giữa 2 thiết bị thì bắt buộc 1 kết nối phải được thành lập.



Hình 1: Mô hình Piconet và Scatternet

Trong LE hỗ trợ "dual mode" tức là một thiết bị có thể đóng 2 vai trò Central và Peripheral đồng thời, ngoài ra LE còn rất linh hoạt trong việc phát triển ứng dụng. Nó có thể tạo kết nối hoặc không cần tạo kết nối giữa 2 thiết bị vẫn có thể gửi được dữ liệu (beacon). Trong tương lai sắp tới, LE sẽ hỗ trợ mạng Mesh – SIG hiện đang làm việc để cung cấp một chuẩn Mesh cho BLE. Và rất nhiều ứng dụng sẽ được hưởng lợi mà công nghệ Mesh mang lại.

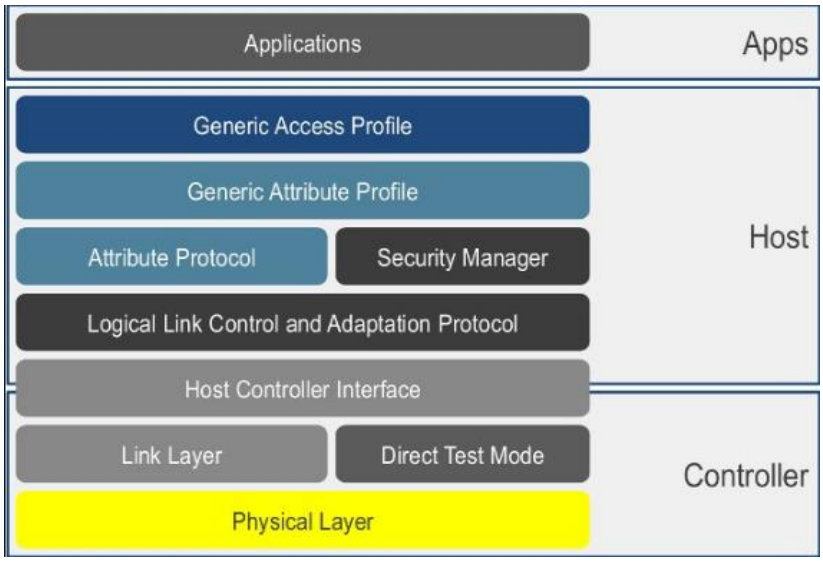


Hình 2: Mô hình mạng BLE

- **Tốc độ truyền dữ liệu**: Tốc độ truyền dữ liệu của BR (2-3Mbps) cao hơn so với BLE (tối đa là 2Mbps).

- **Thiết bị Node**: đối với BR thì 1 thiết bị Master có thể kết nối với tối đa 7 thiết bị Slave trong khi con số đó của BLE cao hơn rất nhiều.
- **Khoảng cách truyền**: khoảng cách truyền và đâm xuyên của BLE cũng tốt hơn và ngày càng cải thiện ở các phiên bản Bluetooth mới, đặc biệt mới đây là Bluetooth 5.0 với khoảng cách truyền lên tới hơn 100 m.

2.2. CẤU TRÚC CỦA MỘT HỆ THỐNG BLE



Hình 3: Cấu trúc hệ thống BLE

2.2.1. Lớp vật lý

Lớp vật lý là lớp thấp nhất trong Protocol Stack, nó đảm nhận trách nhiệm gửi và nhận dữ liệu qua không khí.

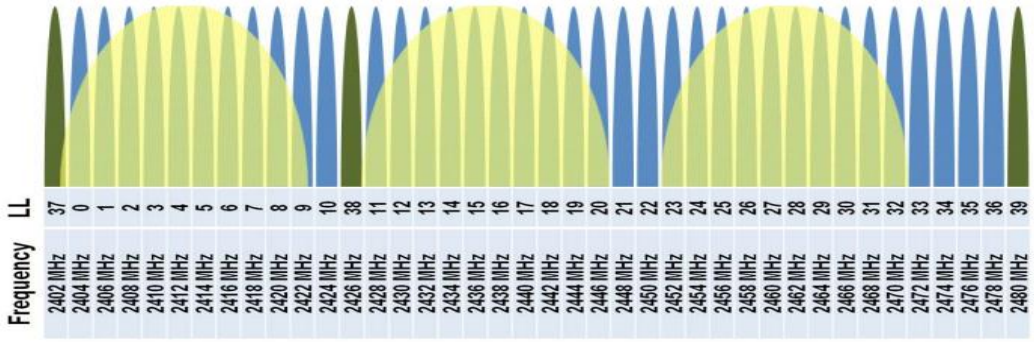
a) Dải tần số truyền

BLE radio hoạt động ở dải tần 2.4Ghz ISM (ISM stands for Industrial, Scientific and Medical) đây là dải tần sử dụng toàn cầu không giấy phép và được dùng cho một vài loại thiết bị khác nữa như remote điều khiển đồ chơi, điện thoại di động, NFC, Wireless LAN … Vì băng tần này được chia sẻ bởi nhiều thiết bị nên có thể có nhiều thiết bị hoạt động độc lập sử dụng cùng một kênh RF giống nhau dẫn đến việc xung đột kênh vật lý. Để giảm thiểu điều này, việc thay đổi kênh liên tục cho các lần trao đổi dữ liệu tiếp theo được sử dụng, kênh mới sẽ được xác định trước để thiết bị biết. Đây được biết đến là phương pháp *frequency hopping*. BLE sử dụng kĩ thuật này để giảm thiểu ảnh hưởng.

Dải tần số hoạt động được chia thành 40 kênh, mỗi kênh cách nhau 2MHz. Số kênh bắt đầu đếm từ 0 – 39 với tần số khởi điểm là 2402MHz tới 24835MHz.

$$f(x) = 2.402 + k * 2\ MHz,\ k = 0, \dots, 39$$

trong đó từ kênh 0-37 được sử dụng để truyn dữ liệu, riêng 3 kênh cuối 37, 38, 39 được dùng là kênh quảng cáo, thiết lập kết nối và gửi các bản tin Broadcast.



Hình 4: Tần số kênh BLE

BLE sử dụng một kĩ thuật được gọi là *frequency hopping spread spectrum* để có thể nhảy giữa 37 kênh truyền dữ liệu, theo đó khi kết nối được thành lập và thiết bị bắt đầu truyền dữ liệu, kênh truyền dữ liệu cho lần kế tiếp sẽ được tính theo công thức:

$$channel = (curr\_channel + hop) \% 37$$

Giá trị "hop" sẽ được truyền đạt mỗi khi kết nối được thành lập. Hop sẽ được tăng và radom giá trị từ 5 → 16. Kênh truyền dữ liệu tiếp theo sẽ khác với kênh hiện tại, và nó được thay đổi liên tục. Thuật toán đơn giản này giúp giảm thiểu tối đa ảnh hưởng từ các hiện diện trong dải tần 2.4GHz, đặc biệt là Wifi và các Classic Bluetooth.

b) Năng lượng đầu ra

BLE v4.0, v4.1, and v4.2 định nghĩa năng lượng đầu ra trong dải từ 0.01 mW (-20 dBm) tới 10 mW (+10 dBm). Việc sử dụng năng lượng đầu ra cao trong trường hợp khoảng truyền ngắn có thể đối mặt với tình trạng thiết bị nhận bị bão hòa hoặc dẫn đến kết nối thất bại. Do đó cần tránh các trường hợp như vậy hoặc sử dụng thêm các bộ chuyển đổi để chuyển đổi giữa các mức năng lượng đầu ra.

Việc điều khiển năng lượng đầu ra cho thiết bị có thể tối ưu mức năng lượng tiêu thụ và làm giảm ảnh hưởng tới các thiết bị khác.

c) Khoảng cách truyền

Dựa trên mức năng lượng đầu ra ở phần trên, thiết bị BLE hỗ trợ dải lên tới từ 30-100 m.

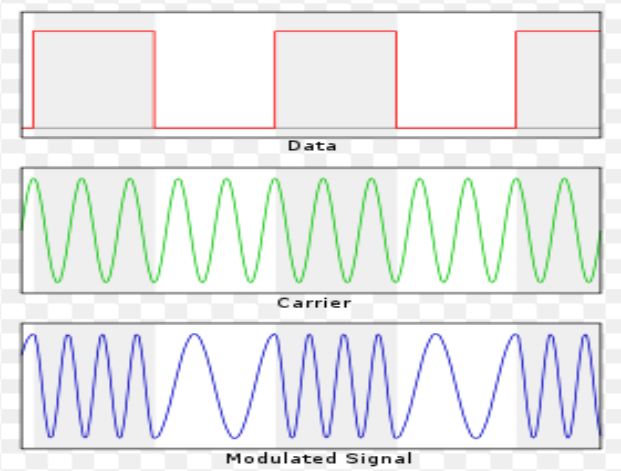
d) Tốc độ truyền dữ liệu

Tốc độ dữ liệu cho BLE được mở rộng từ 1Mbs tới 4 sự lựa chọn: 125Kbs, 500Kbs, 1Mbs và tối đa lên tới 2Mbs.

e) Bộ điều chế tín hiệu

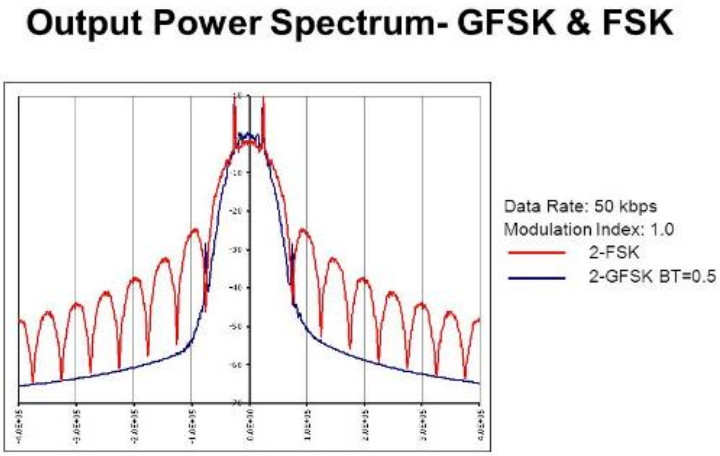
Modulation (bộ điều chế tín hiệu) là một quá trình trộn một tín hiệu với một tín hiệu khác. Tín hiệu chứa thông tin sẽ được gọi là *modulating signal*. Nó sẽ được trộn với một tín hiệu tần số cao được gọi là *carrier signal (sóng mang).* Nói chung, tần số của *carrier signal* phải cao hơn rất nhiều so với *modulating signals*. Ở đây tần số 2.4GHz sẽ là *carrier signal* và *modulating signal* sẽ khoảng MHz. BLE sử dụng kĩ thuật điều chế Gaussian Frequency Shift Keying (GFSK).

Kỹ thuật FSK vận chuyển thông tin bằng cách thay đổi tần số sóng mang để đại diện cho mức 0 và 1. Nhị phân 1 được đại diện bằng cách tăng tần số sóng mang và nhị phân 0 được đại điện bằng cách giảm tần số sóng mang.



Hình 5: Điều chế tần số bằng phương pháp FSK

GFSK sử dụng bộ lọc Gaussian cho tín hiệu trước khi điều chế bằng FSK. Bộ lọc Gaussian sẽ lọc trơn các cạnh nhọn của xung tần số do vậy có thể tránh được các tần số cao tại thời điểm switching.



Hình 6: So sánh hai phương pháp GFSK & FSK

2.2.2. Lớp Link Layer

Lớp Link Layer đảm nhận vai trò điều khiển, đàm phán và thành lập kết nối, chấm dứt kết nối, lựa chọn tần số để truyền dữ liệu, hỗ trợ các cấu trúc liên kết khác nhau và hỗ trợ các cách khác nhau để trao đổi dữ liệu. Vị trí của Link Layer ở trên lớp vật lý và cung cấp dịch vụ tới L2CAP Layer.

Có thể tóm gọn chức năng đảm nhận của Link Layer như sau:

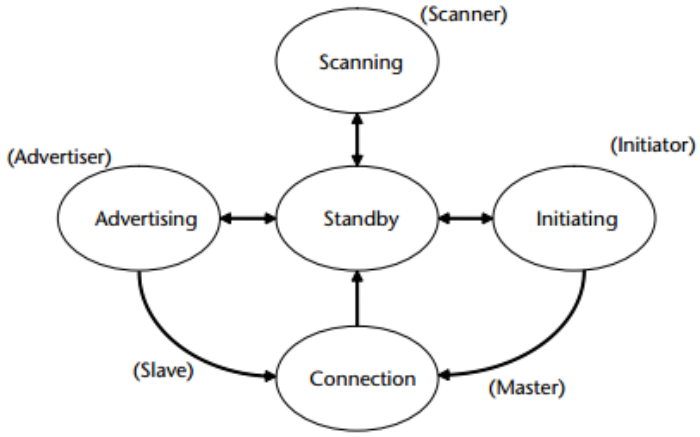
- Quản lý trạng thái của thiết bị.
- Định dạng cấu trúc của gói tin quảng cáo và gói tin dữ liệu.
- Các hoạt động như: Quảng cáo, quét tìm kiếm thiết bị, tạo kết nối.
- Bảo mật

a) Trạng thái của Link Layer

Link Layer quản lý các trạng thái của thiết bị. Có 5 trạng thái chính:

- Standby State (trạng thái chờ)
- Advertising State (trạng thái quảng cáo)
- Scanning State (trạng thái tìm kiếm)
- Initiating State (trạng thái khởi tạo)
- Connection State (trạng thái kết nối)

Lớp Link Layer chỉ cho phép một trạng thái được kích hoạt ở tại một thời điểm.



Hình 7: Trạng thái Link Layer

i) Standby State

Là trạng thái mặc định của Link layer. Trong trạng thái này, sẽ không truyền hay nhận bất kì bản tin nào. Standby State có thể được kích hoạt từ bất kì một trạng thái nào khác.

ii) Advertising State (vai trò Advertiser)

Trong trạng thái Advertising, Link layer sẽ truyền các gói tin quảng cáo. Nó cũng có thể lắng nghe tới các thiết bị khác để phản hồi tới chúng. Trạng thái này có thể kích hoạt từ trạng thái Standby khi Link layer quyết định bắt đầu quảng cáo. Khi đó Link layer được biết đến như Advertiser (thiết bị quảng cáo).

iii) Scanning State (vai trò Scanner)

Trong trạng thái Scanning, Link layer sẽ lắng nghe các gói bản tin từ các Advertiser và có thể yêu cầu Advertiser cung cấp thêm thông tin. Trạng thái này có thể được kích hoạt từ trạng thái Standby khi Link layer quyết định bắt đầu tìm kiếm các thiết bị quảng cáo. Khi đó Link layer được biết đến như Scanner (thiết bị tìm kiếm).

iv) Initiating State (vai trò Initiator)

Khi một Scanner gửi một bản tin tới một Advertiser để yêu cầu khởi tạo một kết nối, khi đó Scanner được biết đến với vai trò Initiator (thiết bị khởi tạo).

v) Connection State (vai trò Master hoặc Slave)

Trong một kết nối, có 2 vai trò được biết đến là Master(chủ) và Slaver(tớ). Nếu thiết bị được khởi tạo từ trạng thái initiating sẽ đóng vai trò như một Master, nếu thiết bị được khởi tạo từ trạng thái advertising sẽ đóng vai trò như một Slaver.

b) Địa chỉ thiết bị

Một thiết bị BLE có thể có một Public Address hoặc một Random Address hoặc cả hai. Một trong hai địa chỉ trên là điều kiện tối thiểu để phân biệt giữa các thiết bị. Cả 2 loại địa chỉ đều có độ dài là 48 bit.

i) Địa chỉ công khai (Public Device Address)

Là 48 bit địa chỉ duy nhất được đăng kí với tổ chức IEEE Registration Authority và sẽ không bao giờ thay đổi trong vòng đời của thiết bị.

Bao gồm 2 phần:

- 24 bit ID công ty được ấn định bởi IEEE ủy quyền. Đây được gọi là Organizationally Unique Identifer và khác nhau giữa các công ty
- 24 bit duy nhất được ấn định bởi công ty tới mỗi Controller. Đây là chuỗi khác nhau giữa các Controller được sản xuất bởi công ty.

ii) Địa chỉ ngẫu nhiên (Random Address)

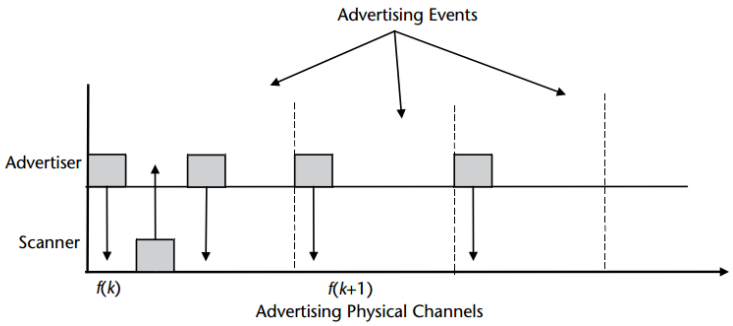
Đây là một đặc tính riêng tư của BLE nơi thiết bị có thể ẩn đi địa chỉ thật sự và dùng 1 số random để thay thế. Điều này giúp đảm bảo thiết bị không bị theo dõi. Được mở rộng thành 3 loại:

- **Static address**: Được sử dụng bất cứ khi nào nhà sản xuất không muốn đăng kí với IEEE. Nó là 1 số random đơn giản được tạo ra bất cứ khi nào thiết bị khởi động lại và sẽ giữ nguyên khi đang chạy cũng như không thể thay đổi nếu đang hoạt động.
- **Non-resolvable private address**: Thông thường sẽ không được sử dụng, nó cũng là một số random được đại diện là địa chỉ tạm thời trong một khoảng thời gian nhất định.
- **Resolvable private address**: Chúng được tạo ra từ một IRK (Identity Resolving Key) và một số random được trao đổi giữa 2 thiết bị trong quá trình ghép cặp được nói tới trong lớp SM. Chúng có thể thay đổi thường xuyên để tránh thiết bị bị nhận dạng và theo dõi bởi một máy quét vô danh. Chỉ thiết bị được phân phối cho IRK mới có khả năng nhận ra được thiết bị nguồn cấp.

c) Hoạt động của Link Layer

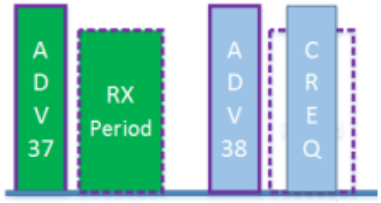
i) Advertising

Sự kiện được dùng cho việc truyền trên các kênh vật lý quảng cáo. Ở thời điểm bắt đầu mỗi sự kiện quảng cáo, Advertiser gửi một gói bản tin quảng cáo. Scanner nhận được bản tin này và dựa vào kiểu của gói tin quảng cáo, nó có thể gửi lại một scan_request cho Advertiser. Advertiser sẽ phản hồi lại scan_request ngay trong sự kiện quảng cáo đó. Sau khi sự kiện quảng cáo kết thúc, Advertiser sẽ sử dụng kênh quảng cáo kế tiếp để phát đi bản tin quảng cáo. Điều này cho thấy Advertiser sẽ quảng cáo trên tất cả 3 kênh theo thứ tự 37->38->39->37->38…



Hình 8: Sự kiện quảng cáo

Sau khi gửi đi bản tin quảng cáo Advertiser sẽ có 1 khoảng thời gian nhất định ngay sau đó để lắng nghe xem có bản tin yêu cầu nào (scan_request hoặc connect_request) của Scanner phản hồi lại không. Nếu Advertiser không nhận được gì nó sẽ chuyển qua kênh tiếp theo để quảng cáo.

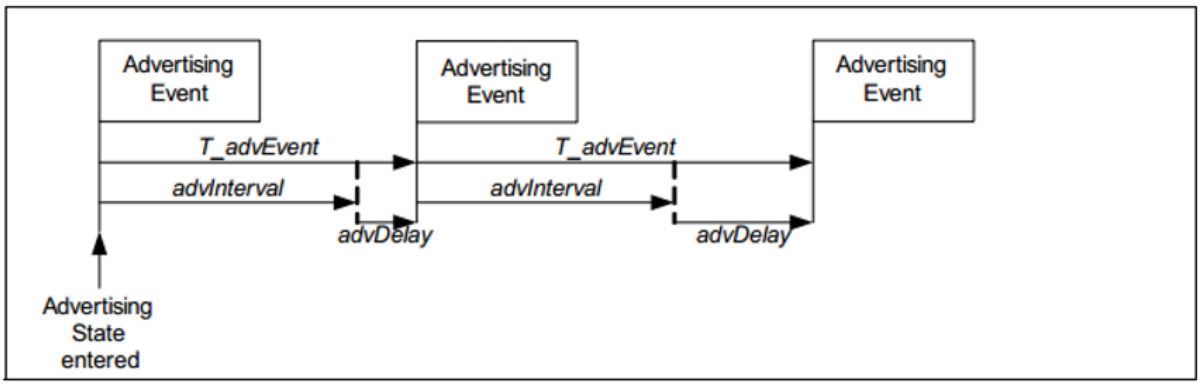


Hình 9: Quá trình quảng cáo

Nếu Advertiser nhận được một connect_request thì nó sẽ ngay lập tức ngừng việc quảng cáo trên các kênh tiếp theo và sau đó kết nối được thành lập.

Nếu Advertiser nhận được một scan_request nó sẽ phản hồi lại Scanner một response_request. Nếu Scanner nhận được response_scan, nó sẽ gửi một connect_request tới Advertiser và sau đó kết nối được thành lập.

Giá trị của chu kì quảng cáo (advInterval) nằm trong khoảng từ 20 ms to 10.24 seconds và advDelay là từ 0-10ms được tạo ra bởi Link layer cho mỗi sự kiện quảng cáo. Nếu đặt chu kì quảng cáo cao, nó có thể làm lâu hơn cho việc thành lập kết nối, trong khi để chu kì quảng cáo thấp sẽ dẫn tới việc thành lập kết nối nhanh hơn. Nó cũng dẫn tới việc phải gửi các gói tin quảng cáo nhiều hơn và dẫn tới tiêu tốn nhiều năng lượng hơn.



Hình 10: Chu kì quảng cáo

Sau khi phát bản tin quảng cáo, Advertiser sẽ ngủ và chờ sự kiện Advertising tiếp theo. Các kiểu gói quảng cáo:

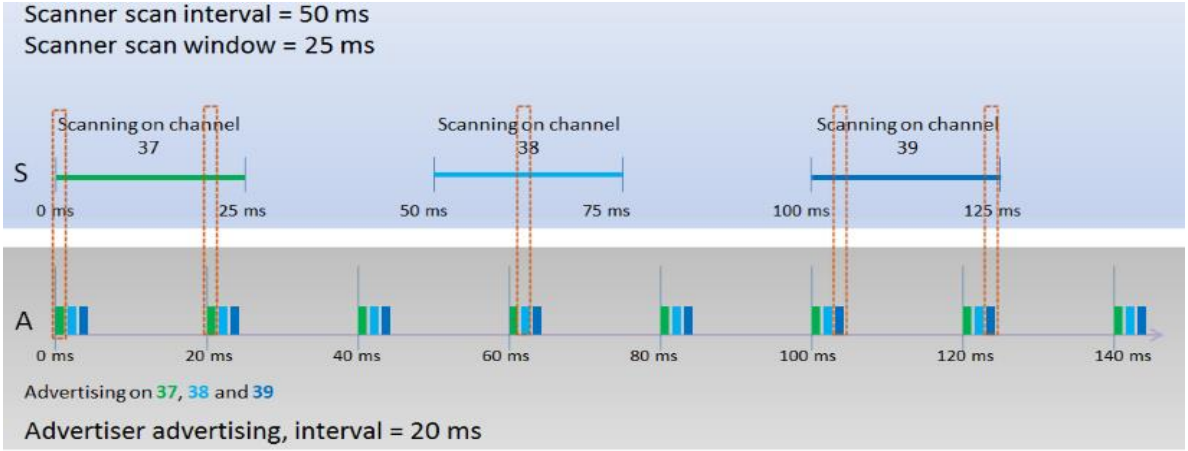
- Connectability
  - Connectable: Scanner có thể khởi tạo kết nối dựa trên gói tin quảng cáo nhận được.
  - Non-connectable: Scanner không thể khởi tạo kết nối.
- Scannability
  - Scannable: Scanner có thể phát hành một scan_request dựa trên gói tin quảng cáo nhận được.

- Non-scannable: Scanner không có thể phát hành một scan_request.

- Directability
  - Directed: Gói dữ liệu chỉ chứa dữ liệu của Advertiser và địa chỉ của các Scanner cụ thể (white list), không cho phép thêm vào các dữ liệu người dùng (ví dụ người dùng muốn thêm vào các dữ liệu như nhiệt độ, độ ẩm …).
  - Undirected: Gói dữ liệu không hướng đến một Scanner nào cụ thể. Nó cho phép người dùng thêm vào các dữ liệu muốn thêm để phát đi cùng quảng cáo.

ii) Scanning

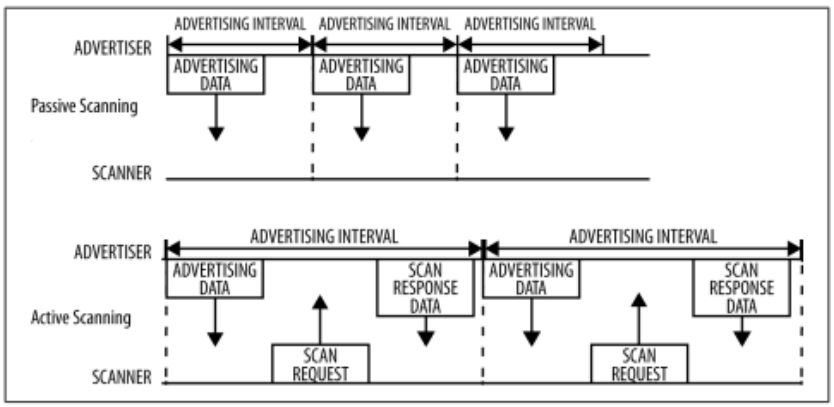
Thiết bị Scanner sẽ lắng nghe các gói tin quảng cáo trên các kênh tương ứng 37,38,39. Sẽ không có cách nào cho Scanner biết được Advertiser đang quảng cáo trên kênh nào, vì vậy chúng sẽ phải vô tình gặp được nhau.



Hình 11: Quá trình phát hiện bản tin quảng cáo

Có 2 dạng scanning chính là:

- **Passive scanning**: Scanner sẽ lắng nghe các gói tin quảng cáo và sẽ không bao giờ phản hồi lại Advertiser.
- **Active scanning:** Scanner phát ra một gói scan_request sau khi nhận được gói tin quảng cáo. Sau đó Advertiser sẽ nhận được và phản hồi lại 1 gói scan_response. Gói gửi thêm này sẽ tăng hiệu quả gửi dữ liệu tới cho Scanner.

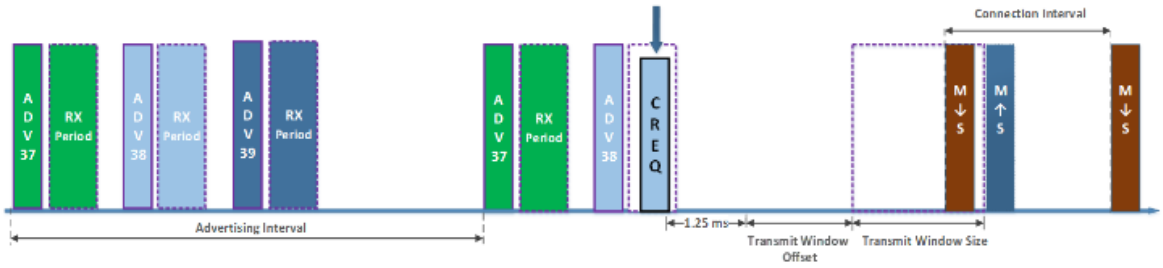


Hình 12: Các dạng scan bản tin quảng cáo

iii) Connection

Để thành lập 1 kết nối, Scanner cần scanning để tìm kiếm các Advertiser để gửi connect_request. Gói bản tin quảng cáo sẽ chứa địa chỉ Advertiser, khi một Advertiser thích hợp được tìm ra, Scanner sẽ gửi một connect_request tới Advertiser, và kết nối được thành lập. Gói connect_request sẽ chứa bước nhảy tần số "hop", cái sẽ xác định kênh để truyền dữ liệu cho cả Scanner và Advertiser trong lần tiếp theo.

Khi một Scanner gửi bản tin connect_request (đang đóng vai trò Iniator) nó sẽ chuyển sang vai trò Master, Advertiser sau khi nhận bản tin connect_request và qua một khâu kiểm tra gói tin này nó sẽ chuyển sang vai trò Slave.

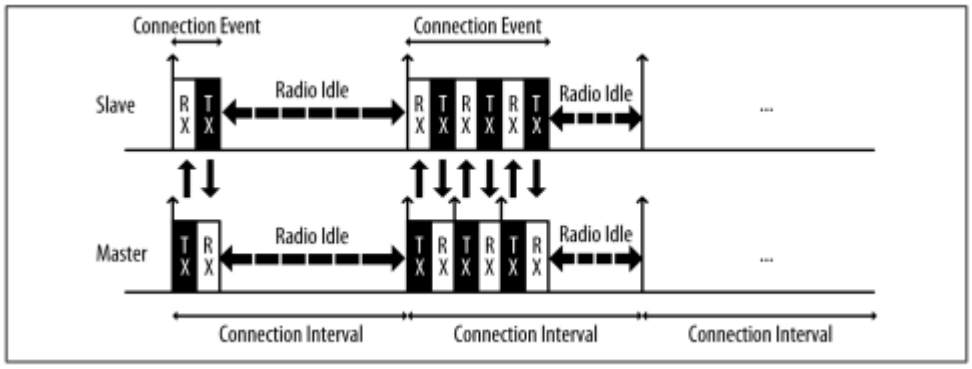


Hình 13: Quá trình thành lập kết nối

Trong bản tin connect_request sẽ chứa 3 thông tin để xác định cửa sổ truyền (transmit window). Cửa sổ truyền sẽ được bắt đầu ngay sau khoảng thời gian transmitWindowDelay + transmitWindowOffset tính từ thời điểm kết thúc bản tin connect_request.

- **transmitWindowDelay**: 1.25ms.
- **transmitWindowOffset** = n*1.25ms nằm trong khoảng từ 0ms tới Connection Interval.
- **transmitWindowSize** = m*1.25ms nằm trong khoảng từ 1.25ms tới 10ms và (Connection Interval -1.25ms).

Connection là cách đơn giản để trao đổi dữ liệu giữa Slaver và Master, mỗi lần trao đổi sẽ được gọi là *connection event*.



Hình 14: Trao đổi dữ liệu

Các thông số quan trọng:

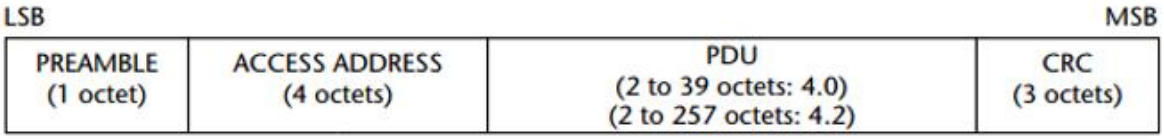
- **Connection interval**: Thời gian giữa 2 lần tạo ra connection event liên tiếp, nằm trong khoảng từ 7.5ms đến 4s.
- **Slave latency:** Số sự kiện connection events mà Slave có thể bỏ qua để tiếp tục ngủ đông thay vì thức dậy trao đổi dữ liệu với Master.
- **Connection supervision timeout:** Khoảng thời gian tối đa giữa 2 lần nhận được gói dữ liệu hợp lệ trước khi kết nối bị mất.

iv) White Lists

Đây là một đặc tính quan trọng của BLE, white lists là một chuỗi đơn giản của địa chỉ Bluetooth Device Addresses. Các Scanner có thể dùng white lists để giới hạn số lượng thiết bị để tìm kiếm hoặc những thiết bị nó có thể kết nối. Các Advertiser có thể dùng white lists để chỉ định các Scanner có thể kết nối tới.

d) Giao thức của Link Layer (Protocol)

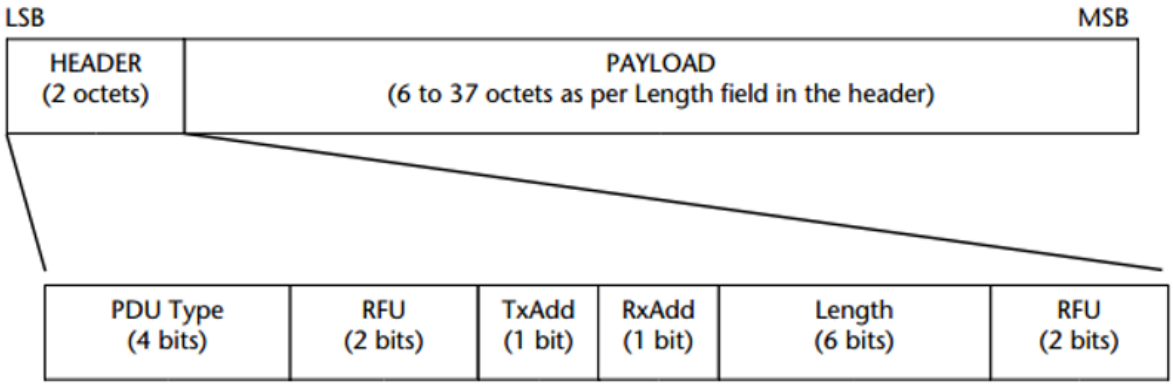
Cấu trúc bản tin của lớp Link Layer dùng chung cho cả kênh quảng cáo và kênh truyền dữ liệu.



Hình 15: Cấu trúc bản tin lớp Link Layer

- **Preamble:** 8 bit gồm các bit 0 và 1 đan xen nhau cho BLE 1Mbs và 16 bit cho BLE 2Mbs:
  - 10101010 hoặc 01010101: dành cho BLE 1Mbs
  - 1010101010101010 hoặc 0101010101010101: dành cho BLE 2Mbs
- **Access Address:** dùng cho tất cả các kênh quảng cáo sẽ là 0x8E89BED6. Link Layer sẽ tạo một giá trị Access Address mới mỗi khi một chu kì quảng cáo mới được bắt đầu.
- **PDU (**Protocol Data Unit): chứa dữ liệu. Có 2 loại PDU:
  - Advertising Channel PDUs: PDU dùng cho kênh quảng cáo
  - Data Channel PDUs: PDU dùng cho kênh truyền dữ liệu

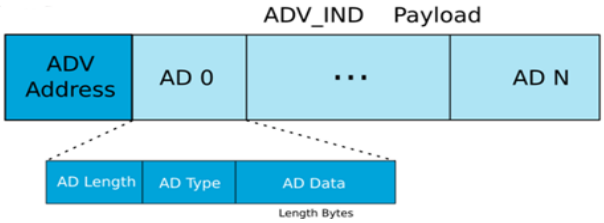
i) Advertising Channel PDUs



Hình 16: Cấu trúc bản tin quảng cáo của lớp Link Layer

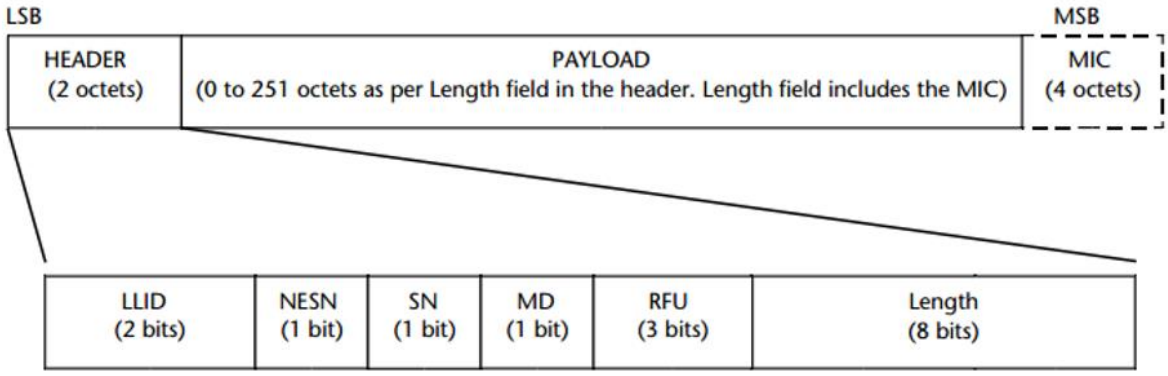
- **PDU Type:** Được chia ra thành 3 dạng:
  - Advertising PDUs
    - **ADV_IND**: bản tin quảng cáo của Peripheral cho phép Central gửi yêu cầu kết nối.
    - **ADV_DIRECT_IND**: bản tin quảng cáo của Peripheral chỉ chấp nhận yêu cầu kết nối từ 1 Central cụ thể (white list).
    - **ADV_NONCONN_IND**: là bản tin quảng cáo của Peripheral, không chấp nhận bất kì yêu cầu kết nối nào.
    - **ADV_SCAN_IND**: là bản tin quảng cáo Peripheral, chấp nhận bản tin scan_request, không chấp nhận bất kì yêu cầu kết nối nào.
  - Scanning PDUs
    - **SCAN_REQ**: bản tin scan_request từ Central để yêu cầu thêm dữ liệu nhận dạng từ Peripheral.
    - **SCAN_RESP**: bản tin trả lời của Peripheral cho scan_request.

  - Initiating PDUs
    - **CONN_REQ**: yêu cầu kết nối gửi từ Central đến một Peripheral sau khi đã nhận đủ dữ liệu nhận dạng.
- **RFU:** 4 bit dự trữ cho tương lai, không dùng tới.
- **TxAdd:** 1 bit xác định địa chỉ MAC của thiết bị BLE là Public (RxAdd/TxAdd = 0) và Random (RxAdd/TxAdd = 1).
- **RxAdd:** xác định Advertising Packet này có chức năng Beacon hay không.
- **Length:** 6 bit xác định độ dài dữ liệu nằm trong phần payload của gói.
- **Payload**: Phần Payload của Advertising Packet chứa các thông tin nhận dạng của thiết bị BLE Peripheral theo cấu trúc như sau:
  - 6 byte đầu tiên chứa địa chỉ của thiết bị phát ra quảng cáo này
  - 31 byte tiếp theo sẽ được gộp theo nhóm để mô tả 1 loại dữ liệu nhận dạng có cấu trúc: **[Chiều dài dữ liệu (1byte)] [Loại thông tin nhận dạng(1byte)] [Dữ liệu của thông tin nhận dạng]**



Hình 17: Cấu trúc dữ liệu chèn thêm

ii) Data Channel PDUs



Hình 18: Cấu trúc bản tin dữ liệu của lớp Link Layer

Bao gồm 2 loại:

- **LL data PDU**: dữ liệu truyền là data.
- **LL Control PDU**: dữ liệu truyền là các lệnh quy định sẵn.

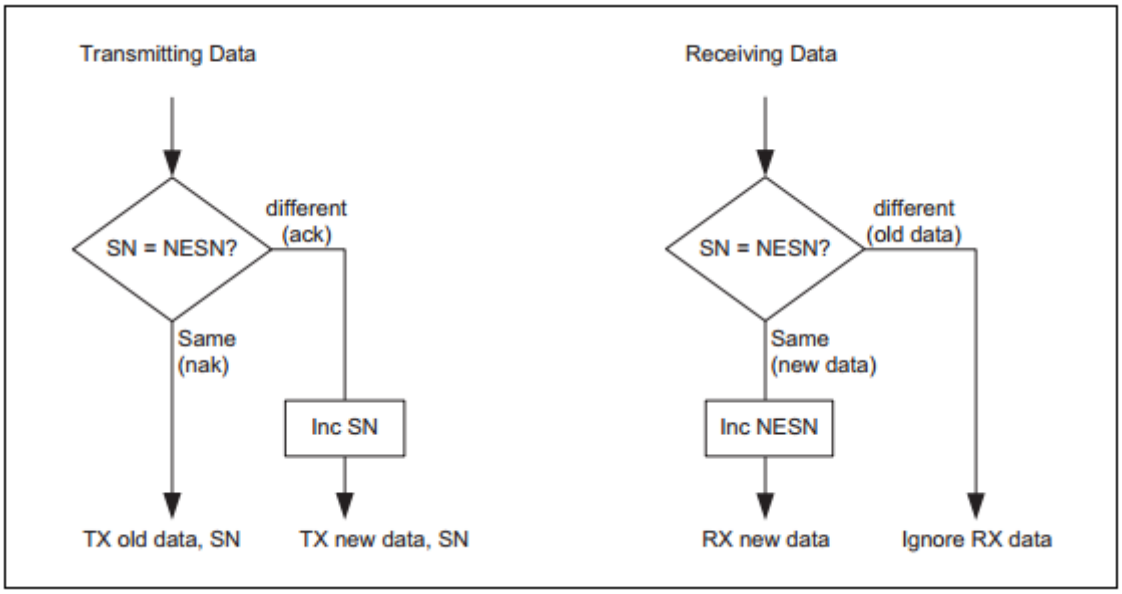
- **LLID:** Chỉ thị kiểu của link layer PDU.
  - **00**: không sử dụng giá trị này.
  - **01**: LL data PDU nếu LLID set 01b và Length set 00000000b, nghĩa là đây là 1 empty PDU. Khi lớp Link layer của Master gửi empty PDU cho Link layer của Slave nghĩa là Master cho phép Slave có thể phản hồi với bất kì bản tin PDU nào kể cả 1 empty PDU.
  - **10**: LL data PDU nghĩa là Master không cho phép Slave gửi 1 empty PDU.
  - **11**: LL Control PDU được dùng để điều khiển và đàm phán kết nối giữa 2 lớp Link layer. Có rất nhiều kiểu bản tin được định nghĩa cho Link layer để trao đổi thông tin điều khiển với các lớp Link layer của các thiết bị khác ví dụ như:
    - Thủ tục cập nhật kết nối
    - Thủ tục cập nhật bản đồ kênh
    - Thủ tục bắt đầu sự mã hóa
    - Thủ tục tạm dừng mã hóa
    - Thủ tục trao đổi đặc tính
    - Thủ tục trao đổi version
    - Thủ tục kết thúc kết nối
    - Loại bỏ PDU
    - Cập nhật thông số kết nối
    - Thủ tục LE Ping
    - Thủ tục cập nhật độ dài dữ liệu
    - Thủ tục từ chối
- **MD**: Chỉ thị là Slave hoặc Master còn dữ liệu để gửi. Nếu bit này được đặt thì sau khi gửi gói hàng này, Master sẽ tiếp tục gửi, Slave sẽ tiếp tục nghe hoặc ngược lại. Nếu gói hàng thứ 2 không được nhận bởi Slave thì Master sẽ đóng kết nối và ngược lại. Nếu gói tin thứ 2 được nhận nhưng CRC không hợp lệ thì kết nối cũng sẽ bị đóng lại.
- **NESN/SN** (bit Acknowledgment)

  Hai bit SN và NESN được dùng để 2 thiết bị yêu cầu gửi lại bản tin nếu thiết bị nhận không nhận được bản tin đúng từ thiết bị gửi. Khi bắt đầu vào trạng thái Connection, SN= NESN=0. Bên gửi: Gửi bản tin, kèm 2 bit SN, NESN = 0. Bên nhận: kiểm tra 2 bit SN và NESN:
  - **SN = NESN**: đây là gói bản tin mới, kiểm tra gói bản tin, nếu gói bản tin đúng, tăng NESN lên 1 đơn vị, và gửi lại bản tin phản hồi với 2 bit SN, NESN mới được cập nhật. Nếu gói bản tin sai, giữ nguyên NESN, gửi lại bản tin phản hồi với 2 bit SN, NESN để báo hiệu bên gửi cần gửi lại bản tin vừa nhận

- **SN != NESN**: đây là 1 gói bản tin gửi lại, kiểm tra gói bản tin. Thực hiện các bước y như trên

Bên gửi: Sau khi gửi xong 1 gói bản tin sẽ chờ gói phản hồi (ACK). Sau khi nhận được gói phản hồi. Kiểm tra 2 bit SN và NESN.

- **SN = NESN**: bản tin vừa gửi bị sai, giữ nguyên giá trị của SN, gửi lại bản tin được yêu cầu cho bên nhận.
- **SN != NESN**: bản tin vừa gửi đã đúng, tăng SN lên 1 đơn vị, gửi bản tin mới



Hình 19: Mô tả kiểm tra ACK

- **MIC** (Message Integrity Check): Được tính theo Payload của Data Chanel PDU và byte đầu tiên của Header và các bit NESN, SN, MD. Sẽ chỉ được thêm vào khi một kết nối Link Layer đã được mã hóa (pairing/bonding) và độ dài trường dữ liệu (Payload) phải khác 0.
- **Length**: Độ dài trường chỉ thị size của Payload + MIC.
- **CRC:** CRC sẽ được tính toán trên trường PDU. Nếu PDU đã được mã hóa (pairing/bonding), thì CRC sẽ được tính toán sau khi việc mã hóa PDU hoàn thành. CRC là 1 số 24 bit nhị phân. Cách tính CRC bằng cách chuyển PDU (Payload + MIC) về dạng bit rồi chia cho giá trị CRC Init được gửi trong bản tin Connect_request có dạng như bên dưới:

$$x^{24} + x^{10} + x^9 + x^6 + x^4 + x^3 + x + 1$$

Phần dư chính là CRC.

e) Bảo mật

Lớp Link Layer cung cấp quá trình bảo mật và quá trình xác nhận sử dụng Counter với chế độ CCM, CCM được thực thi bởi một thuật toán được định nghĩa trong IETF RFC 3610 (http://www.ietf.org/rfc/rfc3610.txt) kết hợp với khối thuật toán AES-128 được định nghĩa trong NIST Publication FIPS-197 (http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf).

Khi một kết nối Link Layer được mã hóa (pairing/bonding) thì Data Channels PDU với Length khác 0 cũng sẽ được mã hóa và xác nhận. Việc xác nhận chính là thêm vào 1 trường MIC tới Payload. Việc mã hóa sẽ được áp dụng cho trường Payload và MIC.

2.2.3. Lớp HCI

Lớp HCI (Host controller interface) cung cấp một phương thức chuẩn cho việc giao tiếp giữa tầng cao và tầng thấp của BLE. Các tầng cao thuộc về Host, các tầng thấp thuộc về Controller.

a) Lớp truyền của HCI

HCI định nghĩa 4 lớp truyền có thể được dùng là:

- UART Transport Layer.
- USB Transport Layer.
- Secure Digital (SD) Transport Layer.
- Three-Wire UART Transport Layer.

Thông thường thì UART Transport Layer được dùng thường xuyên nhất.

b) Giao thức của HCI

Có 4 kiểu gói tin được sử dụng trong giao tiếp Host và Controller là:

- **HCI Command Packet**: Chỉ được dùng để gửi lệnh yêu cầu từ Host tới Controller.
- **HCI Asynchronous (ACL) Data Packet**: Dùng để trao đổi dữ liệu theo 2 chiều giữa Host và Controller. Chỉ có thể trao đổi gói dữ liệu này khi có một kết nối được thành lập.
- **HCI Synchronous (SCO/eSCO) Data Packet**: Được dùng để trao đổi dữ liệu đồng thời giữa Host và Controller.
- **HCI Event Packet**: Chỉ được dùng để thông báo cho Host biết sự kiện nào vừa xảy ra.

Sẽ có 1 gói bản tin chỉ thị được gửi ngay trước khi gửi 1 trong 4 gói trên để báo trước kiểu gói tin HCI sắp được gửi.

Bảng 1 : Các kiểu gói tin của lớp HCI

HCI packet type	HCI packet indicator
HCI Command Packet	0x01
HCI ACL Data Packet	0x02
HCI Synchronous Data Packet	0x03
HCI Event Packet	0x04

2.2.4. Lớp L2CAP

Có 2 chức năng chính:

- Như một giao thức dồn kênh, từ nhiều giao thức ở các lớp trên gói gọn thành định dạng chuẩn của các gói dữ liệu BLE.
- Thực hiện phân mảnh và tái kết hợp dữ liệu, nhìn chung sẽ lấy các gói dữ liệu lớn từ các lớp trên (SM và ATT), phân mảnh thành các gói nhỏ tối thiểu 27 bytes để truyền đi. Về phần nhận, nó nhận các gói hàng nhỏ và đóng gói thành một bản tin duy nhất (B-frame) và truyền lên các lớp bên trên của Host.

a) Chế độ làm việc của L2CAP

Có 6 chế độ làm việc nhưng BLE chỉ làm việc trên 1 chế độ gọi là Basic L2CAP Mode, đây cũng là chế độ mặc định nếu như các chế độ khác không được yêu cầu.

b) Kiểu kênh của L2CAP

Có 3 kiểu channel:

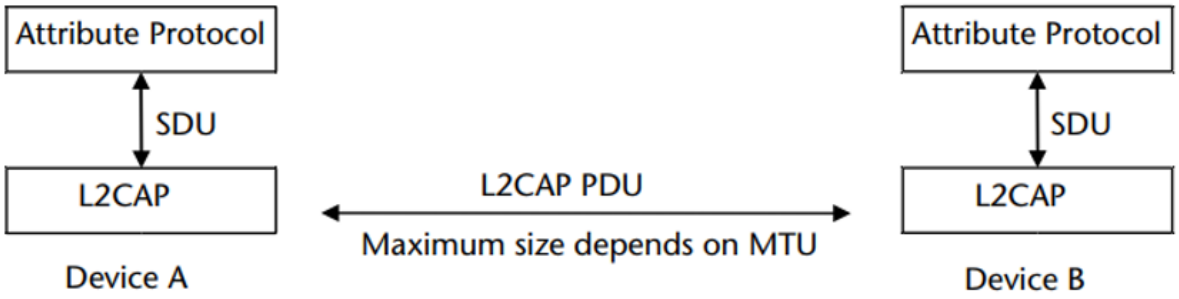
- **Connection-oriented**: Kênh gửi được 2 chiều giao tiếp giữa 2 thiết bị.
- **Connectionless data**: Kênh gửi broadcast, 1 thiết bị cho nhiều thiết bị nhưng chỉ 1 chiều.
- **L2CAP Signaling**: Kênh chuyên dùng để gửi lệnh điều khiển.

c) L2CAP SDU

Chứa các khối dữ liệu nguyên gốc từ các lớp trên, sau khi truyền xuống lớp L2CAP nó sẽ được chia nhỏ ra thành một hoặc nhiều PDU trước khi được truyền đi xuống các lớp dưới.

d) L2CAP PDUs

Là phần dữ liệu chia nhỏ từ SDU được thêm vào thông tin Header để thành một gói đúng cấu trúc cung cấp tới các lớp dưới của các thiết bị đích.



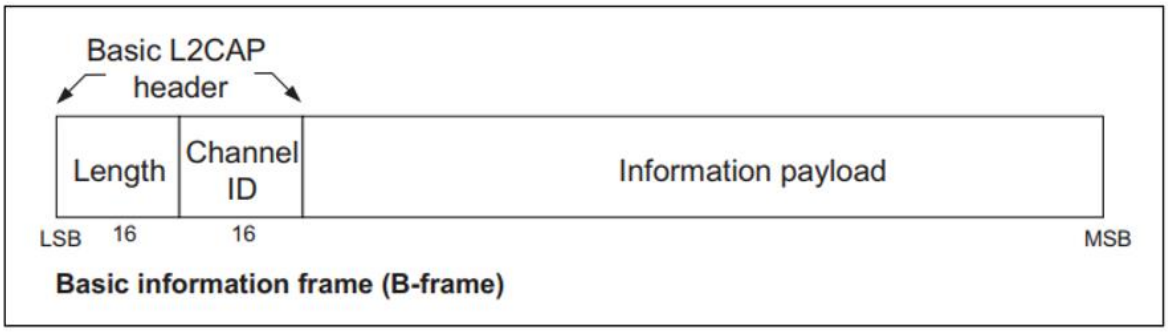
Hình 20: Quá trình truyền dữ liệu ở lớp L2CAP

Có 5 loại gói bản tin dùng để truyền và nhận trên lớp L2CAP bao gồm:

- **B-frame** (Basic Frame): được dùng trong chế độ Basic L2CAP.
- **I-frames** (Information Frame).
- **S-frame** (Supervisory Frame).
- **C-frame** (Control Frame): được dùng trên kênh L2CAP Signaling.
- **G-frame** (Group Frame): được dùng trên kênh connectionless L2CAP.

i) B-frame (Basic Frame)

Trong chế độ Basic L2CAP sử dụng gói dữ liệu có format B-frame để chứa dữ liệu nhận được từ các lớp trên truyền xuống hoặc chứa gói tin sau khi tổng hợp để truyền lên các lớp trên:



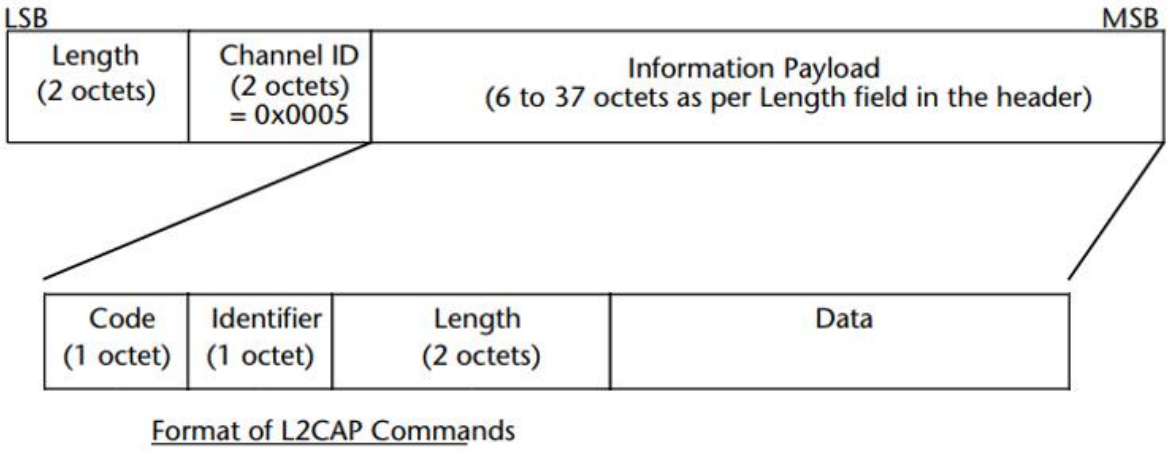
Hình 21: Cấu trúc gói tin B-frame

trong đó:

- **Length** (2 byte): chỉ thị thông tin độ dài của Information Payload.
- **Channel ID** (2 byte): là một kênh chỉ thị đích đến của gói hàng
  - 0x0004: ATT protocol
  - 0x0006: SM protocol
- **Information payload** (0-65635 byte): chứa khối dữ liệu chia nhỏ từ SDU, thông thường chọn là 23 byte.

ii) C-frame (Control Frame)

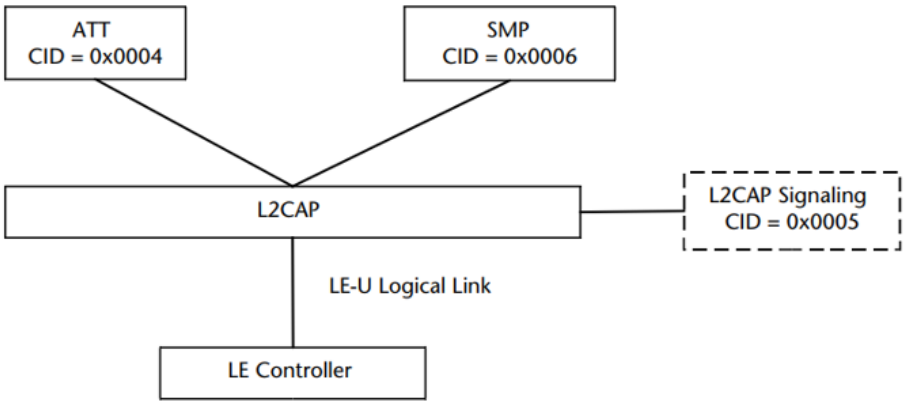
Được BLE sử dụng để gửi lệnh command trên kênh Signaling có Chanel ID là 0x0005.



Hình 22: Cấu trúc gói tin C-frame

- **Code**: 1 byte để nhận dạng kiểu câu lệnh command. Có rất nhiều mã nhận dạng nhưng trong BLE chỉ sử dụng 6 Signaling command là:
  - **LE Credit Based Connection Request:** Tạo và cấu hình một kênh L2CAP giữa 2 thiết bị, như vậy cần gửi đi lệnh này trước tiên để có thể tạo ra các kênh truyền dữ liệu ở lớp L2CAP.
  - **LE Credit-Based Connection Response:** Gói phản hồi lại LE Credit-based connection request.
  - **LE Flow control credit:** Một thiết bị sẽ gửi bản tin này nếu nó có khả năng nhận thêm BLE_frames.
  - **Command Reject:** Được gửi như 1 phản hồi nếu command code không có Identifier hoặc Length không đúng. Nó có chứa lý do chỉ thị tại sao gói bị từ chối.
  - **Connection Parameter Update Request:** Lệnh gửi bởi BLE Slave để yêu cầu Master cài đặt 1 sự lựa chọn thông số mới cho kết nối. Nếu Master đồng ý với yêu cầu, nó sẽ sử dụng thủ tục Link layer connection update (LLID = 11b) đã được giải thích ở mục Link Layer.
  - **Connection Parameter Update Response:** Gói này gửi từ BLE Master tới Slave để phản hồi yêu cầu cập nhật dữ liệu từ Slave. Nếu Master chấp nhận nó sẽ gửi thông số cập nhật kết nối tới Controller sử dụng command HCI_LE_Connection_Update để Controller bắt đầu tiến trình ở lớp Link layer để cập nhật kết nối.

- **Identifier:** giá trị này được đặt bởi thiết bị yêu cầu, thiết bị nhận sẽ gửi lại đúng giá trị này trong bản tin phản hồi. Một giá trị khác nhau được sử dụng cho mỗi command gửi thành công và khác 0. Giá trị này sẽ được tái sử dụng khi mọi giá trị đều đã được dùng.
- **Length**: chỉ thị kích cỡ trường data của command.
- **Data**: 0 hoặc nhiều hơn, trường Code quyết định trường data.



Hình 23: Các kênh trên lớp L2CAP

Sau đó chúng sẽ được liên kết với 1 LE-U Logical Link duy nhất. Chú ý chỉ khi LE-U link được cài đặt xong thì các kênh Chanel ID mới có sẵn để các lớp trên có thể trao đổi dữ liệu với L2CAP. Ngay khi LE-U Logic Link được cài đặt xong, thì các kênh 0x0004, 0x0005, 0x0006 sẽ có sẵn, các lớp cao có thể gửi dữ liệu tới L2CAP để truyền tới thiết bị khác ngay trên các kênh này.

2.2.5. Lớp ATT

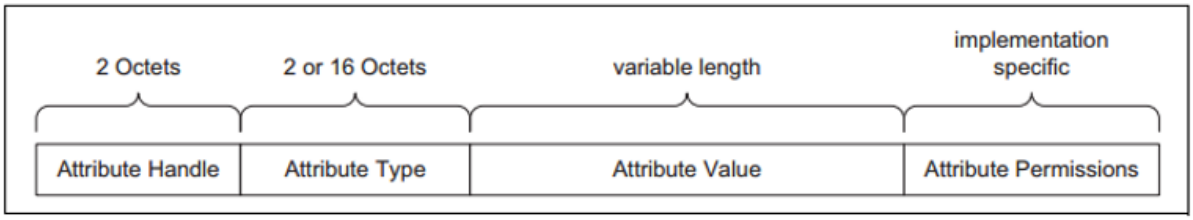
Lớp ATT cung cấp cơ chế cho việc tìm kiếm các thuộc tính, đọc và ghi các thuộc tính. Nó nằm bên trên lớp L2CAP và sử dụng cơ cấu chuyên chở của L2CAP để truyền dữ liệu. Mặt khác ATT cung cấp dịch vụ đến lớp GAP. ATT protocol làm việc theo mô hình Server - Client.

- **Attribute Server**: đưa ra các thuộc tính và các giá trị liên kết với nó.
- **Attribute Client**: tìm kiếm, đọc ghi các thuộc tính trên Server.

a) Thuộc tính (attribute)

Thuộc tính đặc chưng cho dữ liệu. Nó có thể là dữ liệu về bất cứ lĩnh vực gì. Thuộc tính có các đặc trưng quan trọng sau:

- Attribute Type.
- Attribute Handle.
- Attribute Permissions.
- Attribute Value.



Hình 24: Cấu trình Attribute

- Attribute Type: Được nhận diện bởi một UUID (Universally Unique Identifier). Là 1 số gồm 128 bit duy nhất được định nghĩa bởi SIG. Thông thường hay sử dụng kiểu rút gọn gồm 16bit UUID được ấn định bởi SIG. 128 bit value có thể được suy ra từ 16 bit value bằng cách:

  *128-bit UUID = 16-bit UUID * 296 + Bluetooth_Base_UUID*

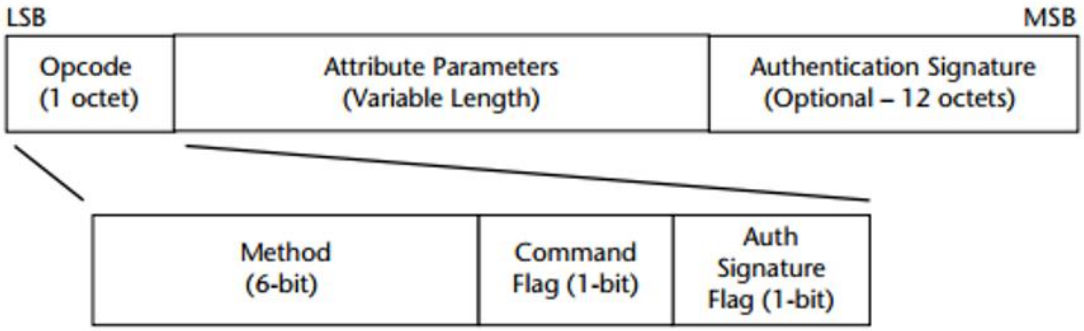
  Điều này cũng có được bằng cách thay giá trị hexa của 16 bit UUID vào *0000xxxx-0000-1000-8000-00805F9B34FB.*
- Attribute Handle (2 byte): Tất cả Attribute trên Server đều được đặc chưng bởi 1 số duy nhất. Ta hiểu đơn giản nó giống như một con trỏ C trỏ đến vị trí của Attribute. Giá trị tối đa là 0xFFFF.
- Attribute Permissions: Mỗi một Attribute sẽ có một quyền hạn được liên kết với nó để xác định mức độ truy cập vào giá trị của Attribute này. Có 4 quyền hạn như sau:
  - Access Permissions:
    - Có thể đọc
    - Có thể ghi
    - Có thể đọc và ghi
  - Encryption Permissions: Có thể yêu cầu (hoặc không) việc kết nối phải được mã hóa để đọc hoặc ghi Attribute.
  - Authentication Permissions: Có thể yêu cầu phải có (hoặc không) mã xác nhận để truy cập Attribute.
  - Authorization Permissions. Có thể yêu cầu (hoặc không) thẩm quyền để truy cập Attribute.
- Attribute Value: Một giá trị Attribute là một chuỗi các byte chứa giá trị thực tế của Attribute. Độ dài của Attribute có thể là:
  - Độ dài được cố định trước: có thể là 1 byte, 2 byte hoặc 4 byte.
  - Độ dài tùy biến: có thể là một chuỗi kí tự có thể thay đổi độ dài.

b) Attribute PDU Protocol

Có 6 loại gói PDU dùng để trao đổi dữ liệu ATT (các Attribute) theo từng mục đích:

- **Request**: PDU này được gửi yêu cầu từ Client lên Server và đòi hỏi phản hồi.
- **Response**: PDU này được Server gửi để phản hồi lại yêu cầu của Client.
- **Command**: Gửi từ Client cho Server. Không đòi phản hồi lại.
- **Notification**: Gửi bởi Server cho Client, không đòi hỏi phản hồi.
- **Indication**: Gửi từ Server cho Client và đòi hỏi phải có 1 sự xác nhận.
- **Confirmation**: PDU này được gửi từ client để xác nhận lại Indication.

Từ 6 loại gói PDU kể trên có chứa rất nhiều bản tin cho các mục đích riêng của Client và Server. PDU Format được thể hiện trên hình sau:



Hình 25: Cấu trúc gói tin của lớp ATT

- **Opcode**: giá trị xác định kiểu và ý nghĩa của Attribute Parameters.
- **Command Flag**: chỉ thị đây là 1 lệnh.
- **Attribute Parameters:** chứa dữ liệu phụ thuộc vào phương thức Method. Có độ dài trường từ 0 đến (ATT_MTU – X) trong đó ATT_MTU là giá trị max của PDU được quy định bởi các lớp tầng trên, X = 1 nếu **Auth Signature Flag** = 0, X = 13 nếu **Auth Signature Flag** = 1.
- **Authentication Signature**: có độ dài 0 nếu Auth Signature Flag = 0, có độ dài 12 byte nếu Auth Signature Flag = 1. Giá trị của trường này sẽ được tính toán ở lớp SM.

2.2.6. Lớp SM

Định nghĩa thủ tục cho việc ghép cặp, sự xác nhận, sự mã hóa giữa các thiết bị BLE. Điều này cần thiết mỗi khi một sự kết nối Link layer được thành lập hoặc một sự bảo mật cho kiểu kết nối đặc biệt. SM được đặt trên L2CAP trong kiến trúc của BLE, nó sử dụng các dịch vụ của L2CAP để thực hiện các thủ tục khác nhau.

a) Ghép cặp (Pairing/Bonding)

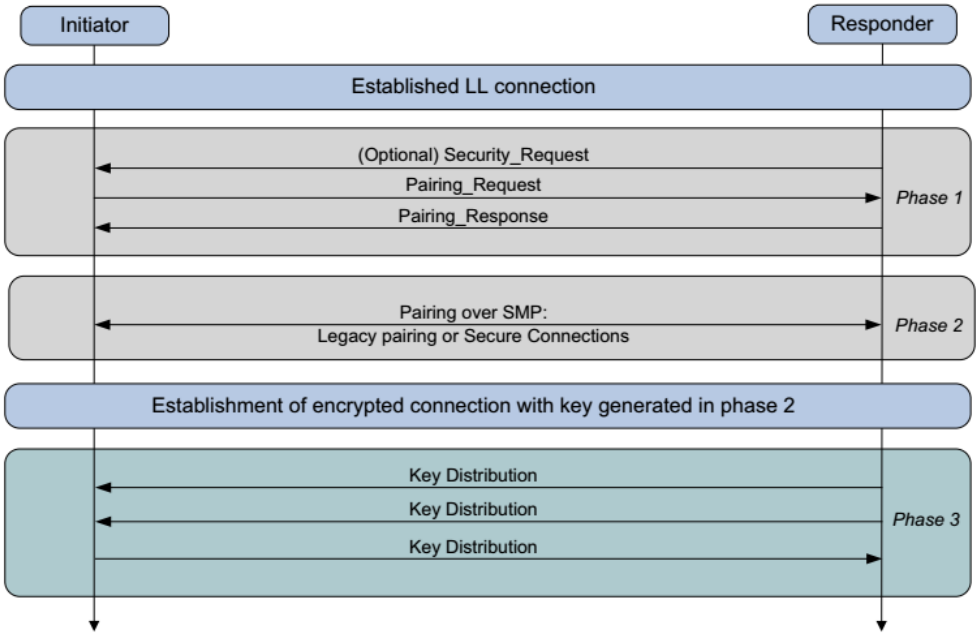
Pairing hoặc Bonding được sử dụng để thành lập các chìa khóa dùng để mã hóa kết nối Link Layer và các gói PDUs. Trong đó:

- Pairing: Hai thiết bị tiến hành kết nối và tạo ra các chìa khóa để mã hóa kết nối đấy. Sau khi hủy kết nối, không thông tin nào được lưu lại. Để kết nối lại với thiết bị trước đó, chúng phải thực hiện lại toàn bộ quá trình.
- Bonding: Hai thiết bị tiến hành kết nối và tạo ra các chìa khóa để mã hóa kết nối đấy. Sau đó chúng thực hiện thêm giai đoạn phân phối các chìa khóa giữa 2 thiết bị để lưu trữ. Sau khi ngắt kết nối, các chìa khóa được trao đổi vẫn được lưu trữ và sẽ được sử dụng lại khi có sự kết nối giữa 2 thiết bị đã từng ghép cặp. Nhờ vậy chúng không cần lại thực hiện lại quá trình pairing.

b) Tiến trình

Nhìn chung sẽ có 3 giai đoạn

- Giai đoạn 1: trao đổi đặc tính ghép nối.
- Giai đoạn 2: lựa chọn 1 trong 2 chế độ dưới để thực hiện
  - **LE legacy pairing**: Tạo Short Term Key
  - **LE secure connections**: Tạo Long Term Key
- Giai đoạn 3: phân phối các chìa khóa đặc biệt (chỉ Bonding mới thực hiện giai đoạn này).



Hình 26: Các giai đoạn diễn trên lớp SM

i) Giai đoạn 1: Trao đổi đặc tính ghép nối

Giai đoạn 1 dùng để trao đổi đặc tính ghép nối bao gồm những thứ như khả năng của IO, yêu cầu cho MITM (Man-In-The-Middle) protection, yêu cầu Bonding… thông tin trao đổi giữa 2 thiết bị thông qua gói bản tin Pairing Request và Pairing Response. Dựa vào khả năng IO để lựa chọn phương thức tạo chìa khóa cho giai đoạn 2. Có 4 phương thức được hỗ trợ, đó là:

- **Just Works**: được dùng khi 1 thiết bị không có cả hai khả năng nhập pass và hiển thị pass ví dụ như ống nghe điện thoại.
- **Numeric Comparison**: 2 thiết bị sẽ tính toán 6 số pass, người dùng sẽ kiểm tra xem 6 số này trên 2 thiết bị có trùng nhau không để xác nhận kết nối.
- **Passkey Entry**: phương thức này được dùng khi 1 thiết bị có thể nhập 6 số pass và 1 thiết bị có thể hiện thị, khi đó thiết bị thứ nhất sẽ hiển thị 6 số pass, người dùng phải nhập vào 6 số đó trên thiết bị thứ hai để kết nối 2 thiết bị.
- **Out of Band(OOB)**: người dùng có thể chạm 2 thiết bị với nhau để truyền 6 số pass ví dụ như NFC hay mã QR.

ii) Giai đoạn 2: Tạo chìa khóa và mã hóa kết nối

Giai đoạn 2 có 2 chế độ như đã nói ở trên và sẽ chỉ có 1 chế độ được lựa chọn và hoạt động.

Trong chế độ **LE legacy pairing**, một chìa khóa tạm thời TK (chính là 128 bit nhị phân của 6 số decimal trao đổi trong giai đoạn 1) được tạo ra nhờ 1 trong 4 phương thức kể trên (just work, numeric comparison, passkey entry, OOB), TK này sẽ được dùng tạo ra STK (Short Term Key) để mã hóa kết nối. Như vậy trong chế độ này sẽ tạo ra 2 chìa khóa là TK và STK để mã hóa kết nối. STK sau đó sẽ được chuyển xuống lớp Link Layer để tạo session key cho việc mã hóa các gói dữ liệu PDUs ở lớp này.

Trong chế độ **LE secure connection pairing** chỉ tạo ra và sử dụng LTK (Long Term Key) để mã hóa kết nối. Sau đó LTK này cũng được dùng ở lớp Link Layer cho việc mã hóa các gói dữ liệu PDUs.

iii) Giai đoạn 3: phân phối các chìa khóa đặc biệt (Bonding)

Giai đoạn này không bắt buộc và nó chỉ được thực hiện khi ở giai đoạn 1 yêu cầu Bonding, giai đoạn 3 cũng chỉ thực hiện trên một kết nối đã được mã hóa nhờ giai đoạn 2, trong giai đoạn này, Master và Slave phân phối các chìa khóa đặc biệt cho nhau. Điều này làm tăng sự tin cậy kết nối giữa 2 thiết bị, cho phép nhanh chóng bảo mật lại kết nối mà không cần thực hiện lại tiến trình pairing (hoặc bonding) trong các lần kết nối tiếp theo. Nếu giai đoạn 2 sử dụng chế độ **LE legacy pairing**, các key vận chuyển bao gồm:

- **Long Term Key (LTK):** 128 bit chìa khóa, chìa khóa này sẽ được tạo ra và phân phối xuống lớp Link Layer để sử dụng cho các lần kết nối sau thay cho STK

- **Encrypted Diversifier (EDIV):** 16 bit dùng để tạo ra và nhận dạng LTK được phân phối
- **Random Number**: 64 bit giá trị dùng để tạo ra và nhận dạng LTK được phân phối
- **Identity Resolution Key (IRK)**: 128bit chìa khóa dùng để tạo ra và giải quyết địa chỉ random của thiết bị. Chỉ các thiết bị nào được phân phối cho IRK mới có thể nhận biết được địa chỉ random của thiết bị phân phối IRK. Điều này giúp bảo vệ thiết bị khỏi các đối tượng nghe lén.
- **Public Device Address or Static Random Address**: địa chỉ của thiết bị.
- **Connection Signature Resolving Key (CSRK)**: 128bit chìa khóa dùng để kí và xác nhận chữ kí dữ liệu dùng ở lớp ATT.

Để tối ưu năng lượng, thiết bị slave sẽ không lưu trữ LTK, EDIV, Random Number mà thiết bị master sẽ làm điều này. Mỗi khi muốn khởi tạo lại 1 kết nối bảo mật với thiết bị từng ghép nối, slave sẽ yêu cầu master gửi EDIV và Random Number để tạo lại LTK.

Nếu giai đoạn 2 sử dụng **LE Secure Connections** thì danh sách bao gồm:

- IRK
- CSRK

Kết nối sẽ được mã hóa hoặc mã hóa lại sử dụng LTK tạo ra từ giai đoạn 2, EDIV và Random được đặt bằng 0.

2.2.7. Lớp GATT

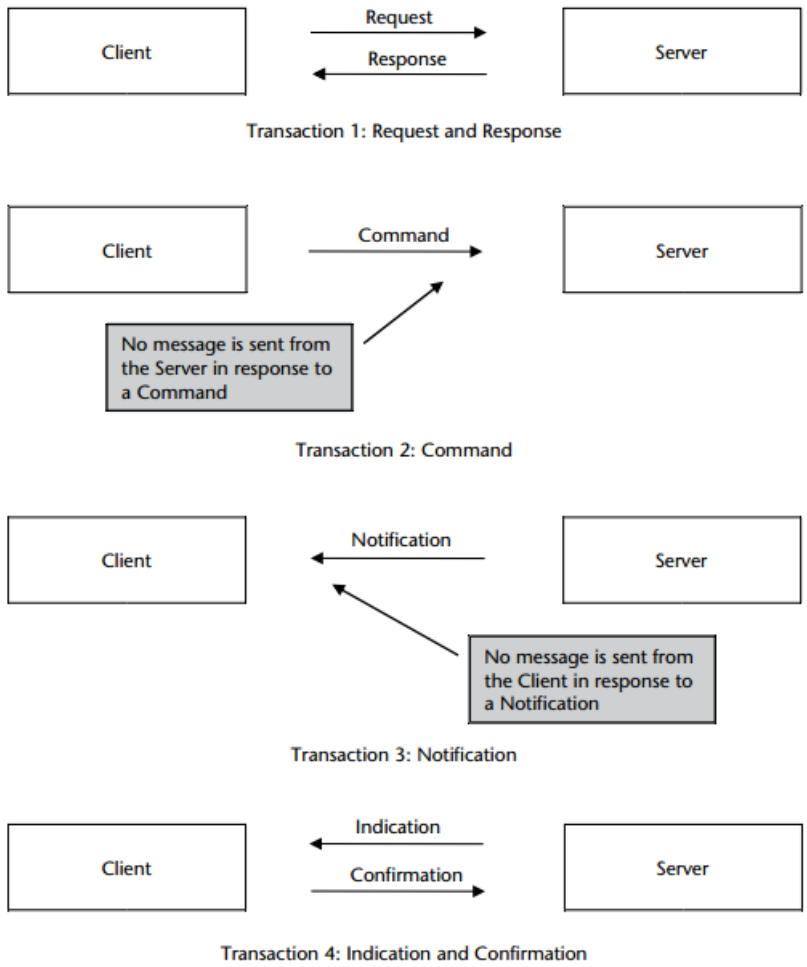
Sử dụng ATT protocol để truyền dữ liệu theo các chuẩn PDU của command, request, indications, notifications và confimations. Giống như ATT, GATT định nghĩa 2 vai trò:

- **Client**: Client khởi tạo giao dịch tới Server và có thể nhận phản hồi từ Server.
- **Server**: Nhận lệnh và yêu cầu từ Client và gửi lại các phản hồi, chỉ thị và thông báo tới Client.

Một thiết bị có thể vừa là Client và Server trong cùng một thời điểm.

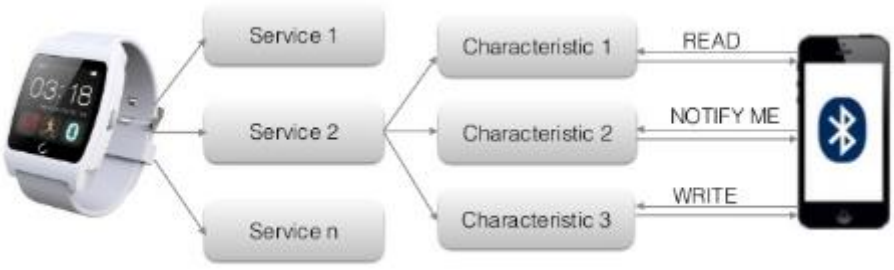
Trong khối GATT, cấu trúc dữ liệu được tổ chức theo mô hình **Service Declaration – Characteristic Declaration – Attribute (Characteristic Value Declaration)**, trong đó Service, Characteristic, Attribute đều là các dữ liệu trong khối ATT, cho nên chúng đều có cấu trúc chuẩn của ATT. Nghĩa là chúng có 4 thành phần UUID riêng, có handle, value, permission riêng.

Một service có thể bao gồm nhiều Characteristic, 1 Characteristic có thể bao gồm nhiều Attribute. BLE profile chính là tập hợp của các Service để thực hiện 1 chức năng cụ thể của 1 thiết bị nào đó. Có thể hiểu BLE Profile là một chức năng cao cấp hơn của GATT Service.



Hình 27: Các hoạt động trên lớp GATT

Ví dụ: Service đo nhiệt độ thì bao gồm 2 loại dữ liệu là nhiệt độ và thời gian, Service đo glucose thì cần 2 dữ liệu là giá trị glucose và giá trị thời gian, và 1 thiết bị sức khỏe BLE (Health Device Profile) thì cần cả 2 service nhiệt độ và service glucose để thực hiện chức năng của 1 thiết bị theo dõi sức khỏe do nó cần cả chức năng đo nhiệt độ và chức năng đo glucose theo thời gian.



Hình 28: Mô tả cấu trúc dữ liệu GATT

a) Service Declaration

- **Handle**: 0xXXXX, giá trị này là duy nhất và không bị trùng nhau.
- **Attribute Type**: 0x2800 – UUID cho dịch vụ chính hoặc 0x2801 cho dịch vụ phụ thứ 2.
- **Attribute Value**: 16 bit UUID hoặc 128 bit UUID cho dịch vụ đã được SIG quy định sẵn, dựa vào gía trị này ta sẽ biết dịch vụ này là dịch vụ về cái gì.
- **Attribute Permission**: chỉ đọc, không cần xác nhận, không cần ủy quyền.

Bảng 2: Cấu trúc của Service Declation

Attribute Handle	Attribute Type	Attribute Value	Attribute Permission
0xNNNN	0x2800	16 bit UUID or 128 bit UUID cho Service	Chỉ đọc, không cần xác nhận, không cần ủy quyền.

b) Characteristic Declaration

- **Handle**: 0xXXXX, giá trị này là duy nhất và không bị trùng nhau.
- **Attribute Type**: UUID của Characteristic là 0x2803.
- **Attribute Value:** Bao gồm Handle, UUID của một Attribute khác chứa dữ liệu của Characteristic này và Charecteristic Properties mô tả cách sử dụng của Attribute đấy.
- **Attribute Permission**: chỉ đọc, không cần xác nhận, không cần ủy quyền.

Bảng 3: Cấu trúc của Characteristic Declaration

Attribute Handle	Attribute Types	Attribute Value			Attribute Permission
0xNNNN	0x2803	Characteristic Properties	Characteristic Value Attribute Handle	Characteristic UUID	Chỉ đọc, không cần xác nhận, không cần ủy quyền.

c) Characteristic Value Declaration

Characteristic Value Declaration là các attribute chứa giá trị của các Characteristic Declaration

- **Handle:** giá trị Handle nằm trong Characteristic Declaration.
- **Attribute Type:** giá trị UUID nằm trong Characteristic Declaration.

- **Attribute Value:** Chứa giá trị của Characteristic.
- **Attribute Permission:** được chỉ định bởi các lớp bên trên.

Bảng 4: Cấu trúc của Characteristic Value Declaration

Attribute Handle	Attribute Type	Attribute Value	Attribute Permissions
0xNNNN	16 bit UUID hoặc 128 bit UUID cho Characteristic UUID	Characteristic Value	Phụ thuộc vào các lớp tầng trên

d) Characteristic Descriptor Declarations

Được dùng để chứa các thông tin liên quan về Characteristic Value (nằm trong **Characteristic Value Declaration**) bao gồm 6 loại:

- Characteristic Extended Properties (0x2900)
- Characteristic User Description (0x2901)
- Client Characteristic Configuration (0x2902)
- Server Characteristic Configuration (0x2903)
- Characteristic Presentation Format (0x2904)
- Characteristic Aggregate Format (0x2905)

trong đó chúng ta sẽ chỉ tập trung vào **Client Characteristic Configuration (CCCD).** Cấu trúc khung của attribute này như sau:

Bảng 5: Cấu trúc của Characteristic Descriptor Declarations

Attribute Handle	Attribute Type	Attribute Value	Attribute Permissions
0xNNNN	0x2902	Characteristic Configuration Bits	Có thể đọc không cần xác nhận không cần ủy quyền. Có thể ghi cùng với sự xác nhận và ủy quyền được định nghĩa bởi các lớp trên.

Như chúng ta biết, khi Client muốn đọc 1 giá trị Attribute của Server, để Server có thể gửi lại giá trị này cho Client (ví dụ giá trị về nhiệt độ …), Server sẽ sử dụng gói PDU Indication hoặc Notification. Indication đòi hỏi một ACK gửi lại trong khi đó Notification thì không cần Client phản hồi lại ACK. Và như vậy để đơn giản, chúng ta có xu hướng sử dụng Notification. Chỉ khi Notification được cho phép (enable) bởi Client thì Server mới có thể gửi dữ liệu Attribute cho Client. Các giá trị của Attribute:

- 0x0000: giá trị mặc định, không cho phép Indification và Notification
- 0x0001: Notification được cho phép
- 0x0002: Indification được cho phép

2.2.8. Lớp GAP

GAP định nghĩa những chức năng cơ sở chung tới tất cả thiết bị Bluetooth. Điều này bao gồm các chế độ thiết bị và những tiến trình chung liên quan tới tìm kiếm thiết bị trong phạm vi lân cận, kết nối tới thiết bị và bảo mật. GAP là bắt buộc để thực thi tới tất cả thiết bị hỗ trợ Bluetooth.

a) Vai trò

GAP định nghĩa 4 vai trò cho thiết bị BLE:

i) Vai trò Broadcaster:

Thiết bị làm việc trong chế độ Broadcaster sẽ chỉ phát ra các bản tin quảng cáo mà không tạo kết nối với các thiết bị khác.

ii) Vai trò Observer:

Thiết bị hoạt động trong chế độ Observer sẽ chỉ tìm kiếm các bản tin quảng cáo mà không tạo kết nối.

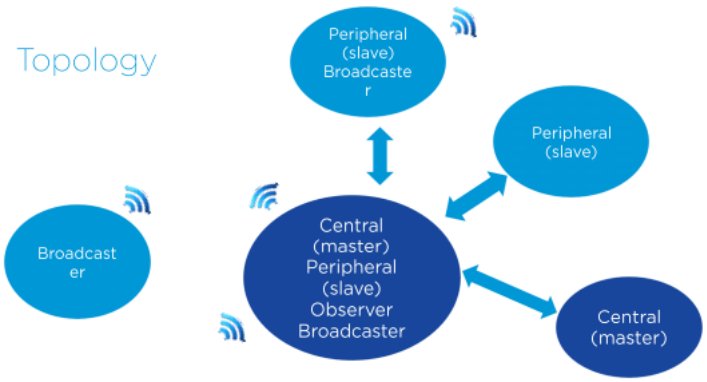
iii) Vai trò Peripheral:

Thiết bị làm việc trong chế độ Peripheral sẽ phát ra các bản tin quảng cáo và chấp nhận các yêu cầu kết nối đến từ các thiết bị khác. Một thiết bị Pheripheral có thể kết nối tới 20 thiết bị cùng lúc.

iv) Vai trò Central:

Thiết bị làm việc trong chế độ Central sẽ tìm kiếm các bản tin quảng cáo và phát ra các yêu cầu kết nối tới thiết bị khác. Một Central có thể kết nối tới 20 thiết bị cùng lúc.

Một thiết bị có thể hoạt động nhiều vai trò cùng lúc.



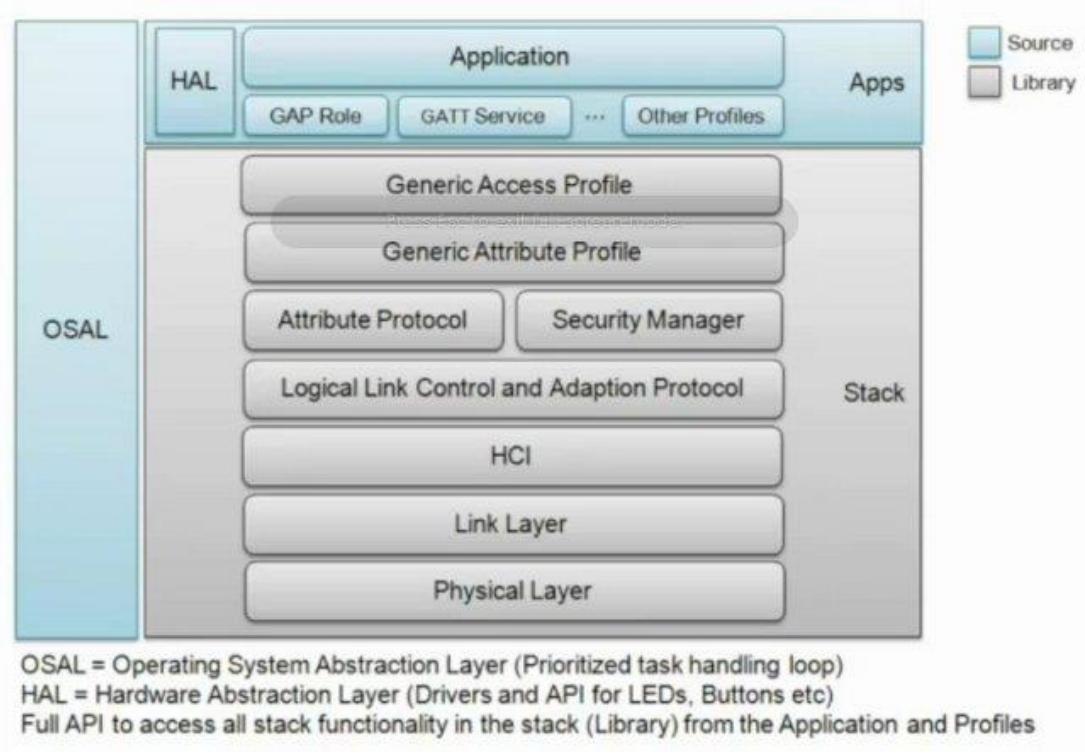
Hình 29: Cấu trúc mạng BLE

b) Chế độ hoạt động và các tiến trình

- Chế độ Broadcast và tiến trình Observation.
- Chế độ Discovery và tiến trình Discovery.
- Chế độ Connection và tiến trình Connection.
- Chế độ Bonding và tiến trình Bonding.
- Chế độ Advertising chu kì và tiến trình Advertising.

2.2.9. Gói BLE Stack

BLE stack là tên gọi chung cho các thư viện code được cung cấp bởi nhà sản xuất chip BLE như Nodric, TI, Silabs…



Hình 30: Cấu trúc của BLE Stack

Như trên mô hình trên chúng ta có thể thấy khối BLE Stack rất nhiều các khối nhỏ, mỗi khối này lại thực hiện một chức năng riêng biệt trong chuẩn BLE. Do đó BLE Stack giúp ta thực hiện toàn bộ các chức năng của các khối nằm trong nó. Việc của chúng ta là nạp thư việc stack này vào chip và sử dụng các hàm API được hãng cung cấp sẵn. Trong thư viện này, các nhà sản xuất phần cứng đã đảm bảo quản lý tất cả các lớp phần cứng cho truyền nhận BLE, quản lý dữ liệu … một cách ổn định.

Do đó nếu chúng ta không nạp thư viện stack vào chip thì sẽ không thể ra lệnh để chip hoạt động được.

2.3. GIAO THỨC MQTT

2.3.1. MQTT là gì?

Message Queuing Telemetry Transport (MQTT) là giao thức gửi nhận tin nhắn (message) theo mô hình Publish/Subscribe, dựa trên một Broker (điểm trung gian). Được sử dụng cho các thiết bị Internet of Things với băng thông thấp, độ tin cậy cao và khả năng được sử dụng trong mạng lưới không ổn định. Bởi vậy MQTT phù hợp cho các ứng dụng M2M (Machine to Machine), WSN (Wireless Sensor Networks) và IoT (Internet of things).

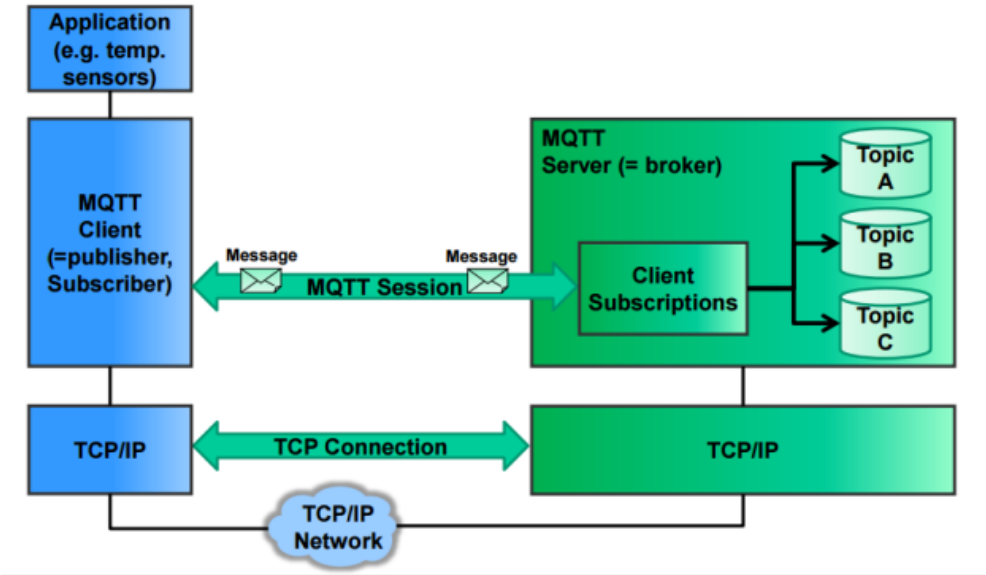
2.3.2. Đặc điểm của MQTT

- Truyền tin nhắn theo mô hình Publish/Subcibe cung cấp việc truyền tin nhắn từ một điểm đến nhiều điểm.
- Truyền tin nhắn không quan tâm đến nội dung được truyền.
- Sử dụng TCP/IP là giao thức nền. Nó mất rất ít bytes cho việc kết nối với Server và sự kết nối có thể giữ trạng thái mở xuyên suốt. Ưu điểm là giao tiếp sẽ mất ít dữ liệu và thời gian hơn HTTP Protocol.
- MQTT kiểm soát các gói tin Header được giữ sao cho càng nhỏ càng tốt. Mỗi gói tin điều khiển MQTT bao gồm ba phần, phần Header cố định 2 byte, phần Header thay đổi chứa thông tin nhận dạng của gói tin và Payload có dung lượng lên tới 256Mb.
- Thông báo ngắt kết nối bất thường của MQTT Client. Khi Client bất ngờ bị ngắt kết nối, bộ đếm thời gian giữ ở phía Server sẽ phát hiện ra rằng Client đã không gửi bất kỳ tin nhắn hay PINGREQ còn sống nào. Lúc đó, Server ngay lập tức publish tin nhắn Will vào Topic Will đã được chỉ định bởi Client.
- Có 3 cấp độ chất lượng truyền tin:
  - QoS0: chỉ truyền một lần, không quan tâm về việc mất hoặc lặp nội dung truyền, việc truyền tin phụ thuộc hoàn vào độ tin cậy của TCP/IP.
  - QoS1: truyền ít nhất một lần, các gói tin đươc đảm bảo nhận nhưng có thể xảy ra lặp.

  - QoS2: chính xác chỉ một lần, đảm bảo gói tin đến nơi đúng một lần tuy nhiên thời gian thực hiện dài hơn.
- Là giao thức đơn giản, tiêu thụ ít năng lượng, sử dụng được cho mạng có dây hoặc không dây thiếu sự ổn định.

2.3.3. Các thành phần của MQTT

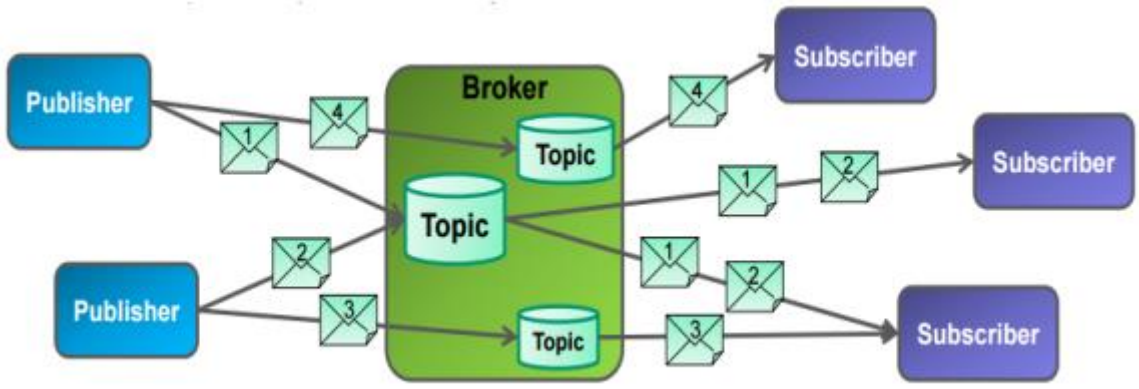
Các thành phần chính của MQTT là Client, Server (Broker), phiên (Session), Subscriptions và các Topic.



Hình 31: Các thành phần của MQTT

- **MQTT Client**: MQTT Client đăng kí (Subscribe) tới Topic để nhận dữ liệu từ Topic đấy và đẩy (Publish) dữ liệu lên Topic đã đăng kí.
- **MQTT Server** (Broker): Server chạy các topic để nhận các đăng kí topic từ Client, từ đó có thể nhận dữ liệu mà Client đẩy lên Topic hoặc gửi dữ liệu từ Topic tới các Client đã đăng kí Topic đấy.
- **Topic**: là nơi đợi dữ liệu được đẩy lên từ Client và gửi dữ liệu đến Client đã đăng kí nó.
- **Session**: dùng để xác định các tệp truyền từ Client đến Topic. Tất cả các liên kết giữa Client và Server là một phần của phiên.
- **Subscription**: Một Subscription "gắn" 1 Client đến 1 Topic.
- **Message**: Là đơn vị dữ liệu được vận chuyển giữa MQTT Client và MQTT Server.

2.3.4. Hoạt động của MQTT

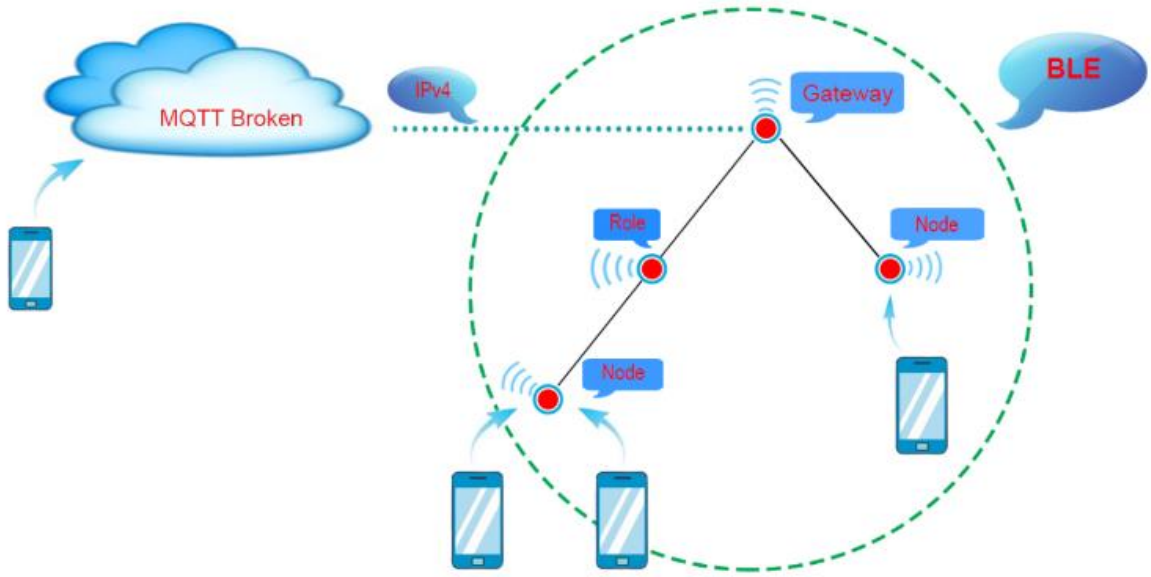


Hình 32: Mô hình miêu tả sự hoạt động của MQTT

Các Client sẽ đăng kí nhận dữ liệu từ các Topic. Một Client có thể đăng kí nhiều Topic và 1 Topic có thể nhận đăng kí từ nhiều Client. Khi Client gửi tin nhắn chứa dữ liệu lên Topic thì được gọi là Public. Đăng kí nhận dữ liệu từ 1 Topic của Broker thì gọi là Subscriber. Khi 1 Client gửi tin nhắn lên 1 Topic mà nó đăng kí, thì Broker sẽ gửi tin nhắn đó cho tất cả các Client cùng đăng kí Topic đấy.

CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

3.1. PHÂN TÍCH HỆ THỐNG



Hình 33: Mô hình hệ thống mạng cảm biến BLE

Một mô hình hệ thống đầy đủ sẽ bao gồm 1 MQTT Broken (Ubidots), 1 Gateway có vai trò thu thập toàn bộ thông tin của các Node để gửi bản tin public lên MQTT Broken thông qua 1 module Wifi, các Node nhận dữ liệu và gửi dữ liệu về Gateway, các Role đóng vai trò trung gian truyền dữ liệu giữa các Node và Gateway.

Trong đó trên các mạch Gateway, Node, Role đều có mạch công suất điều khiển các Relay để bật tắt các thiết bị trong nhà theo 3 cách: qua nút bấm trên mỗi mạch và qua điện thoại hoặc Web. Mô hình có một ưu điểm đó là có thể điều khiển các Relay cả khi online hay offline.

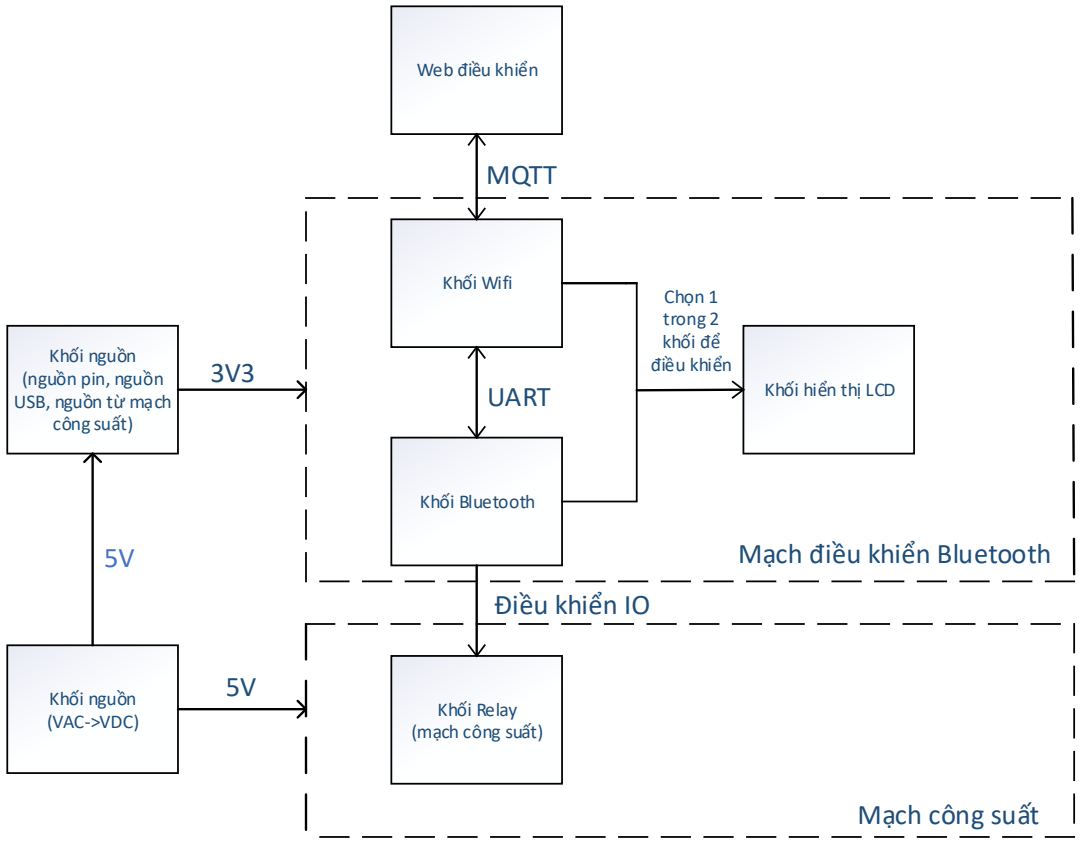
Ở chế độ online, có Wifi, có thể điều khiển Relay trực tiếp trên giao diện web của Ubidot hoặc qua phần mềm điện thoại (rất nhiều phần mềm MQTT trên cả Android và IOS) hoặc điều khiển trực tiếp qua nút bấm trên các mạch Gateway, Node, Role.

Ở chế độ offline, không có Wifi, có thể điều khiển Relay qua phần mềm điện thoại bằng Bluetooth (nRF UART có trên Android và IOS) hoặc điều khiển trực tiếp qua nút bấm trên mạch.

Một điểm đặc biệt nữa là từ một Node bất kì có thể điều khiển trạng thái Relay trên các Node khác trong mạng BLE.

3.2. THIẾT KẾ PHẦN CỨNG

Sơ đồ khối tổng quan



Hình 34 : Mô hình mạch tổng quan

Từ sơ đồ khối ta có thể thấy toàn bộ mô hình bao gồm 2 mạch riêng rẽ. mạch điều khiển bao gồm 3 khối chính: khối Bluetooth, khối Wifi và khối hiển thị LCD. Mạch công suất để điều khiển trạng thái đóng mở của các Relay.

- Khối Wifi sử dụng module Wifi ESP8266.
- Khối Bluetooth sử dụng module NRF52832 của Nordic.
- Khối hiển thị sử dụng LCD 16x2.
- Khối chuyển đổi 5V-3V3 sử dụng IC AMS1117.
- Khối chuyển đổi AC-DC sử dụng module HLK-PM01.

Sau đây chúng ta sẽ đi sâu vào giới thiệu, phân tích và lựa chọn từng linh kiện có trong mạch.

3.2.1. Giới thiệu chip BLE NRF52832

Chip BLE NRF52832 của hãng Nordic là một Wireless Microcontroller System-on-Chip (SoC) vô cùng mạnh mẽ với lõi ARM Cortex-M4 32-bit 64MHz được thiết kế nhỏ gọn giá thành thấp, dễ dàng tích hợp trực tiếp lên các mạch ứng dụng thực tế để thực hiện chức năng BLE kết nối với các thiết bị smart phone.

Thông số kĩ thuật chính:

- Lõi ARM Cotex -M4 32-bit
- Tốc độ clock lên đến 64MHz
- 512 kB bộ nhớ Flash / 64 kB RAM
- 8 kênh ADC 12 bit (không sử dụng)
- 32 chân I/O (trong đề tài chỉ sử dụng 8 chân IO)
- 5 timer 32-bit, Watchdog Timer (không sử dụng)
- 3 bộ SPI tích hợp EasyDMA (không sử dụng)
- 2 bộ I2C (không sử dụng)
- 1 UART tích hợp EasyDMA (sử dụng trong đề tài để giao tiếp với ESP8266)
- Hỗ trợ PPI (không sử dụng)
- 3 bộ đếm real-time (không sử dụng)
- 3 module PWM cung cấp tới 12 kênh PWM (không sử dụng)
- Khối mã hóa AES tích hợp EasyDMA
- Hỗ trợ nhiều giao thức BLE / ANT / 2.4Ghz SoC
- Năng lượng thấp
  - Dòng đỉnh trong chế độ TX (0 dBm) là 5.3mA
  - Dòng đỉnh trong chế độ RX là 5.4mA
  - 0.3µA, 3V trong chế độ System OFF
  - 0.7µA, 3V trong chế độ System OFF với bộ nhớ RAM 64kB
  - 1.9µA, 3V trong chế độ System ON và RTC
- Bộ thu phát RF:
  - Độ nhạy nhận tốt -96dBm
  - Hỗ trợ tốc độ truyền 125Kbps, 500Kbps, 1Mbps, 2 Mbps.
  - Output Power từ -20 tới 4dBm

Các công cụ hỗ trợ:

- Segger Embedded Studio
- Keil MDK-ARM (sử dụng để lập trình trong đề tài)
- GCC
- IAR Embedded Workbench
- Development kit for nRF52832 SoC
- Hỗ trợ đầy đủ phần mềm test riêng cho từng ứng dụng trên PC và điện thoại
- Hỗ trợ nRF52 SDK (Software Development Kit)

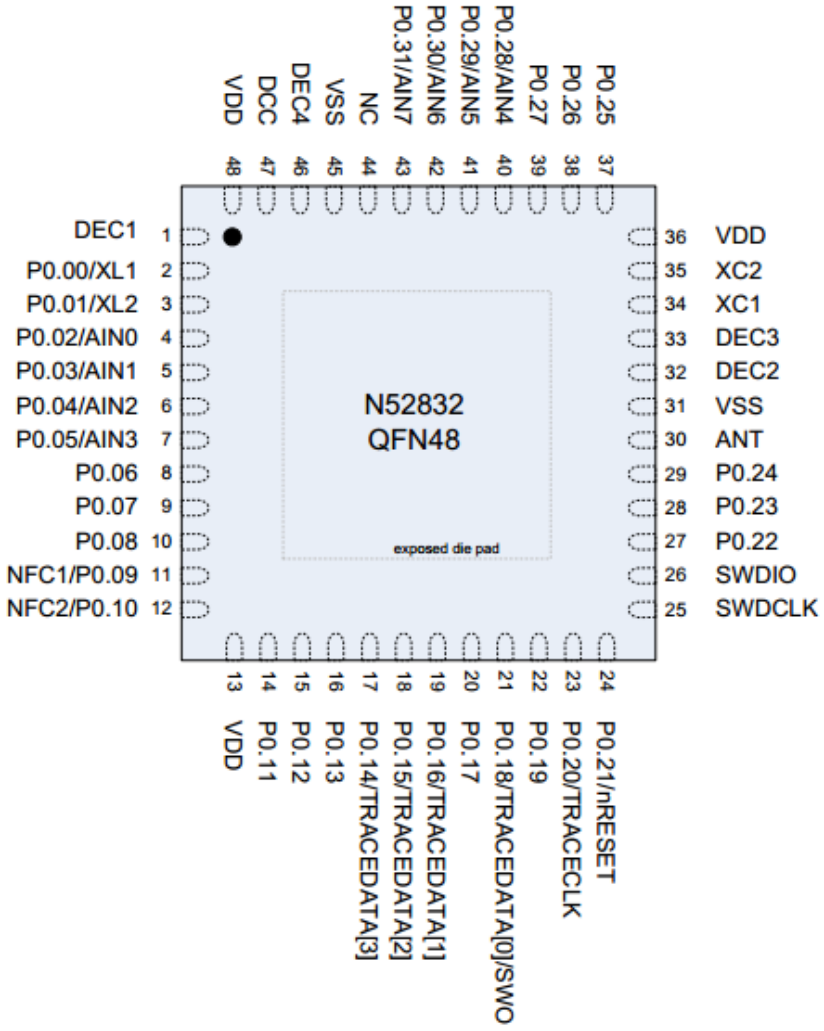
Ứng dụng:

- Internet of Things (nhà thông minh, mạng cảm biến, đô thị tự động, mạng khu vực cá nhân…)
- Cảm biến sức khỏe và thiết bị điều khiển, thiết bị y tế, đồng hồ
- Khóa cửa thông minh
- Các thiết bị điều khiển chơi game
- Beacons
- Các thiết bị phụ trợ PCs (chuột, bàn phím...)

Tài liệu chi tiết tham khảo tại:

http://infocenter.nordicsemi.com/pdf/nRF52832_PS_v1.4.pdf

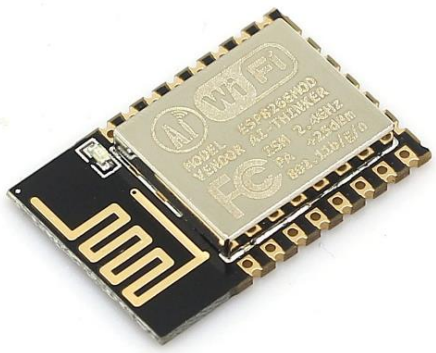
Sơ đồ chân:



Hình 35: Sơ đồ chân chip NRF52832

3.2.2. Giới thiệu Module Wifi ESP8266

ESP8266 là dòng chip tích hợp Wi-Fi 2.4Ghz có thể lập trình được, rẻ tiền được sản xuất bởi một công ty bán dẫn Trung Quốc: Espressif Systems. Hiện nay tất cả các dòng chip ESP8266 trên thị trường đều mang nhãn ESP8266EX, là phiên bản nâng cấp của ESP8266.

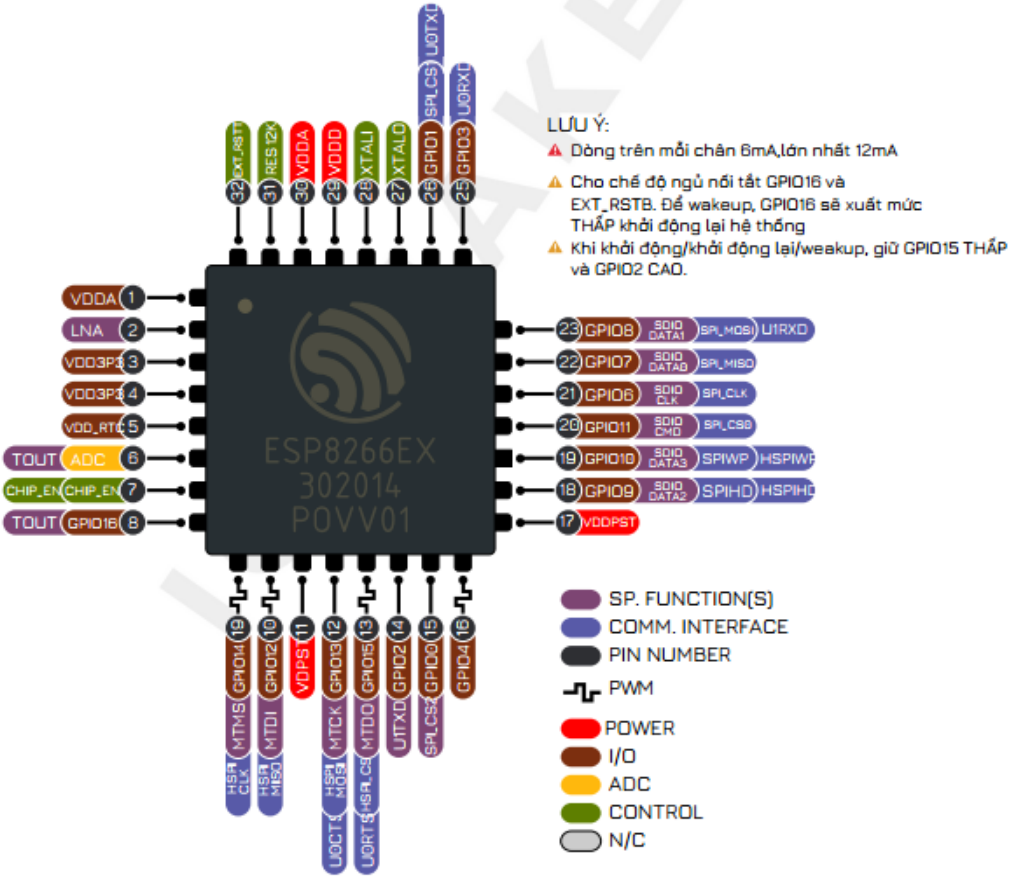


Hình 36: Hình ảnh thực tế module ESP8266

Thông số phần cứng:

- Điện áp hoạt động 3V~3.6V.
- Dòng tiêu thụ 80mA.
- Nhiệt độ hoạt động: -40~125^{o}C.
- 32-bit RISC CPU: Tensilica Xtensa LX106 chạy ở xung nhịp 80 MHz.
- Hổ trợ Flash ngoài từ 512KiB đến 4MiB.
- 64KBytes RAM thực thi lệnh.
- 96KBytes RAM dữ liệu.
- 64KBytes boot ROM.
- Chuẩn wifi EEE 802.11 b/g/n, Wi-Fi 2.4 GHz.
    - Tích hợp TR switch, balun, LNA, khuếch đại công suất và matching network.
    - Hổ trợ WEP, WPA/WPA2, Open network.
- Tích hợp giao thức TCP/IP.
- Hổ trợ nhiều loại anten.
- 16 chân GPIO.
- Hỗ trợ SDIO 2.0, UART, SPI, I²C, PWM, I²S với DMA (chỉ sử dụng UART trong đề tài).
- 1 ADC 10-bit.

Sơ đồ chân:



Hình 37: Sơ đồ chân và chức năng của chip ESP8266

SDK hỗ trợ chính thức từ hãng: Espressif hiện đã hỗ trợ 3 nền tảng SDK (Software Development Kit - Gói phát triển phần mềm) độc lập, là: NONOS SDK, RTOS SDK và Arduino. Cả 3 đều có những ưu điểm riêng phù hợp với từng ứng dụng nhất định, và sử dụng chung nhiều các hàm điều khiển phần cứng. Trong dự án này chúng ta sẽ sử dụng Arduino bởi tính dễ sử dụng, kiến trúc phần mềm tốt và tận dụng được nhiều thư viện.

Tài liệu chi tiết tham khảo tại:

http://espressif.com/en/support/download/documents?keys=&field_type_tid%5B%5D=14

3.2.3. Thiết kế mạch công suất

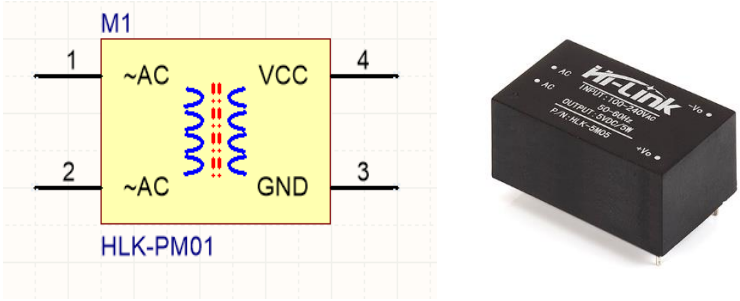
a) Khối nguồn

Yêu cầu kĩ thuật

- Chuyển từ 220VAC về 5VDC cấp nguồn cho khối đóng ngắt và khối mạch điều khiển.
- Đầu ra DC ổn định, cho dòng cao.

- Cách ly tốt giữa nguồn 220VAC và nguồi một chiều cung cấp cho khối đóng cắt và khối điều khiển.
- Cung cấp đủ công suất cho khối đóng cắt và khối điều khiển.

Để chuyển từ 220VAC về 5VDC, em lựa chọn module nguồn AC-DC 5V3W HLK-PM01 chuyển đổi nguồn AC sang DC cách ly, chống nhiễu cao. Dễ thiết kế do yêu cầu ít thành phần ngoài.



Hình 38: Module nguồn chuyển đổi xoay chiều sang một chiều

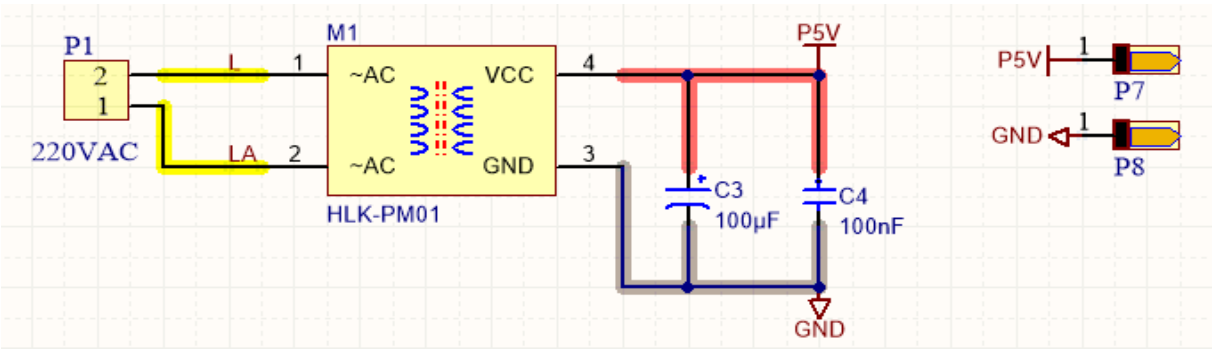
Sơ đồ chân HLK-PM01:

- 2 chân AC đầu vào nguồn xoay chiều
- VCC: output 5V
- GND: chân nối đất

Thông số kĩ thuật:

- Dải điện áp đầu vào: 90~264VAC
- Dòng vào tối đa: <= 200mA
- Điện áp đầu ra: 5VDC
- Dòng ra tối đa trong thời gian dài: >= 600mA
- Công suất: 3W
- Nhiệt độ hoạt động: -20~60^{o} C, độ ẩm 5~95%

Sơ đồ nguyên lý :



Hình 39: Sơ đồ nguyên lý mạch nguồn công suất

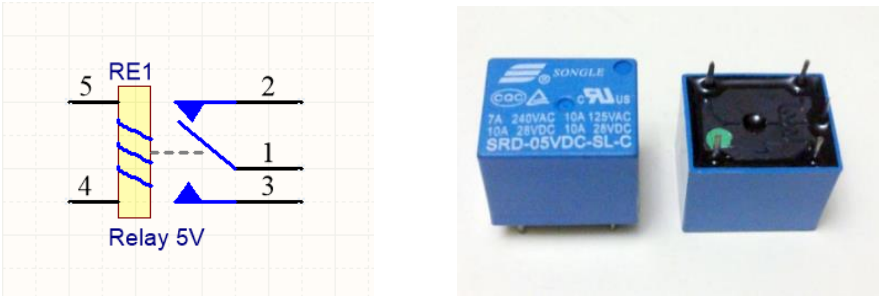
Tụ C3 có chức năng giảm độ gợn sóng, là phẳng điện áp tăng chất lượng đầu ra, tụ C4 chống nhiễu. 2 jump P7, P8 cấp nguồn cho khối điều khiển.

b) Khối đóng cắt

Yêu cầu thiết kế:

- Đóng cắt được thiết bị sử dụng nguồn 220VAC
- Được điều khiển bởi khối VDK
- Độ bền cao, cách ly được khối nguồn 220VAC, đáp ứng đóng cách nhanh

Để đóng cắt thiết bị điện gia dụng, em lựa chọn Relay 5V10A 5 Chân SRD-05VDC-SL-C.



Hình 40: Thiết bị đóng cắt Relay

Sơ đồ chân:

- Chân 4,5 là hai chân đầu vào cuộn hút được nối với nguồn 5V.
- Chân 2 là tiếp điểm thường đóng (NC).
- Chân 3 là tiếp điểm thường mở (NO).
- Chân 1 là chân chung (COM).

Thông số kĩ thuật:

- Relay 1 kênh.
- Điện áp điều khiển cuộn hút: 3~48VDC.
- Dòng chịu đựng 10A.
- Áp chịu đựng 240VAC.

Để kết hợp với Relay đóng cắt và VDK, em lựa chọn transistor NPN MMBT4401.

Hình 41: Transistor NPN

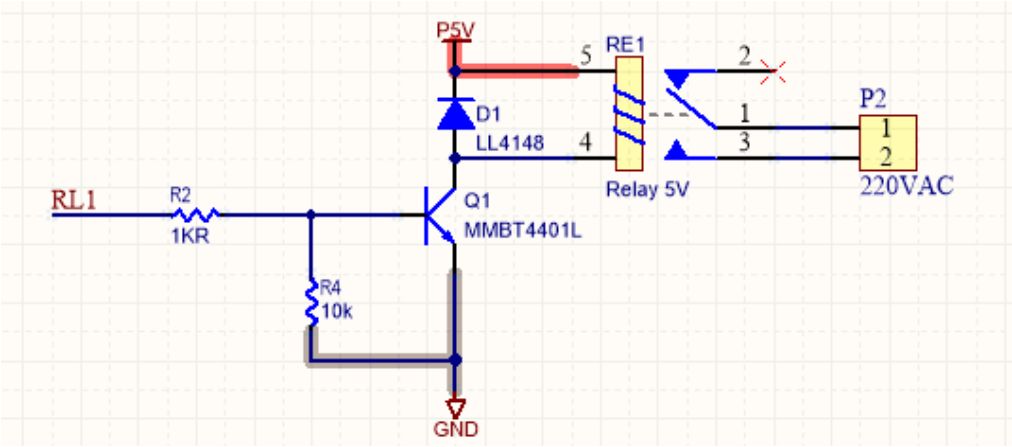
Sơ đồ chân:

- Chân 1: chân Base.
- Chân 2: chân Emitter.
- Chân 3: chân Collector.

Thông số kĩ thuật:

- Nhiệt độ hoạt động -55~150^{o}C.
- Điện áp định mức 60V.
- Dòng định mức 500mA.
- Loại transistor thuận NPN.

Sơ đồ nguyên lý:



Hình 42: Sơ đồ nguyên lý mạch điều khiển đóng cắt Relay

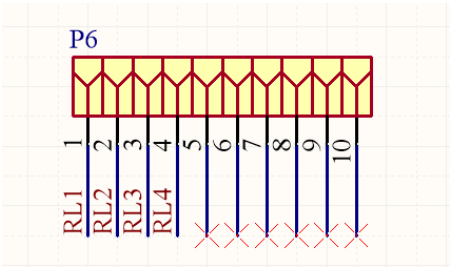
Trở R2 để hạn dòng đi vào cực B của transistor, trở R4 để ghim điện áp cực B xuống đất, diode D1 để khi đóng cắt xuất hiện dòng dội ngược về từ cuộn hút sẽ đi qua D1 và suy giảm hết tránh việc bị chết transistor. Hai chân 1 và 3 được nối với một chân thiết bị điện và 1 chân nguồn.

Khi RL1 xuất ra mức 1 sẽ xuất hiện dòng đi vào cực B của transistor sẽ gây mở transistor làm 1 đầu cuộn hút thông xuống đất làm xuất hiện dòng điện qua cuộn hút và sinh ra từ trường như một nam châm điện, từ trường sinh ra sẽ hút lõi thép động 1 từ vị trí 2 xuống vị trí 3 cung cấp nguồn cho thiết bị điện hoạt động.

Khi RL1 xuất ra mức 0 hoặc bị thả trôi, cực B của transistor được ghim xuống đất và ở trạng thái đóng.

c) Khối kết nối

Gồm Jump để kết nối mạch công suất với mạch điều khiển và cung cấp nguồn 5VDC cho khối điều khiển.



Hình 43: Schematic Jump

3.2.4. Thiết kế mạch Gateway

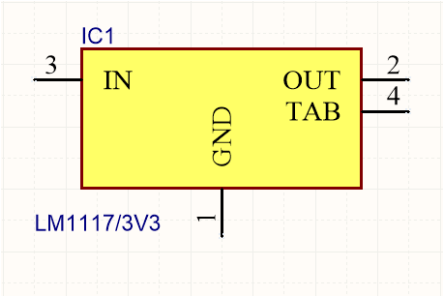
Mạch Node và Gateway được thiết kế để có thể sử dụng chung một bản vẽ, đối với mạch Gateway chỉ cần thêm thành phần Wifi so với mạch Node và Role. Khi hàn mạch Gateway chỉ cần hàn thêm module Wifi so với mạch Node và Role.

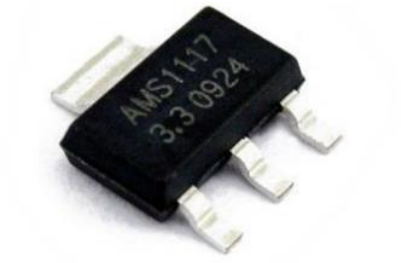
a) Khối nguồn

Yêu cầu kĩ thuật:

- Cấp đủ nguồn 3.3V cho khối VDK, Wifi và khối hiển thị.
- Chất lượng nguồn tốt, chống nhiễu.
- Thiết kế đơn giản.

Để chuyển đổi nguồn từ 5V về 3.3V em lựa chọn IC AMS1117-3.3V:





Hình 44: IC chuyển đổi nguồn áp AMS1117

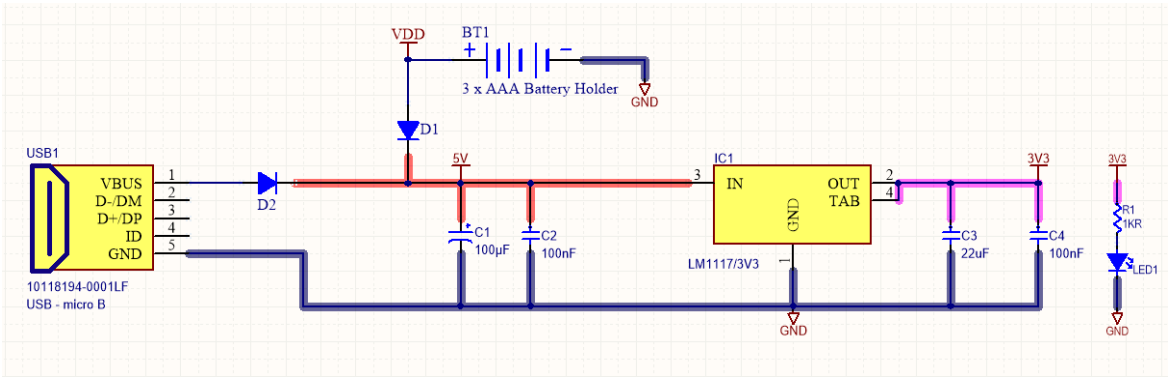
Sơ đồ chân:

- Chân 1: chân nối xuống đất.
- Chân 3: nguồn đầu vào.
- Chân 2,4: nguồn đầu ra.

Thông số kĩ thuật:

- Điện áp đầu vào trong dải: 4.8V ~ 10.3V.
- Điện áp đầu ra: 3.3V.
- Dòng đầu ra: 1A.
- Nhiệt độ hoạt động trong dải: -20~125^{o}C

Sơ đồ nguyên lý:



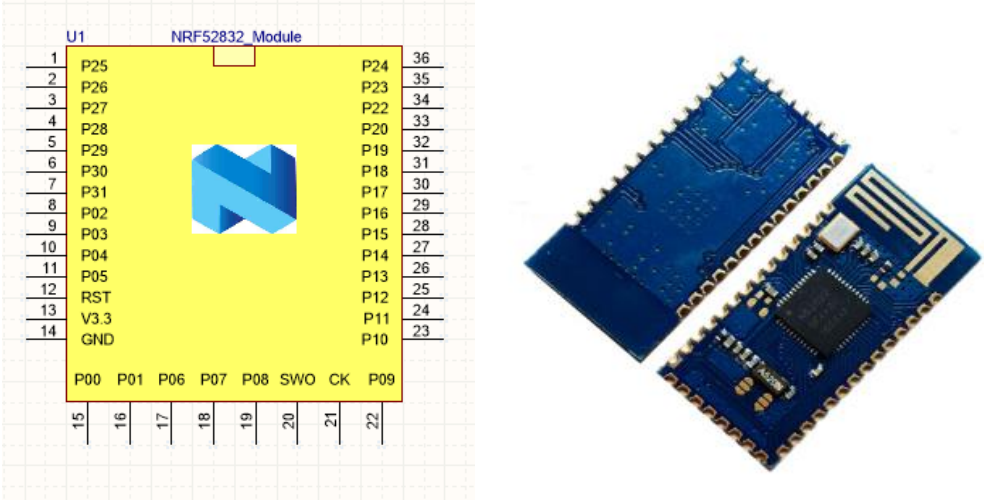
Hình 45: Sơ đồ nguyên lý mạch nguồn Bluetooth

Đầu vào 5V bao gồm 3 nguồn lựa chọn: nguồn cấp từ micro USB, nguồn từ pin, và nguồn từ mạch công suất. Diode D1 và D2 chống áp ngược chiều và tự động lựa chọn nguồn cấp cho IC AMS 1117. Tụ C1 và C2 để giảm gợn sóng là phẳng nguồn vào và chống nhiễu. Tụ C3, C4 có chức năng tương tự. LED1 là led để báo hiệu nguồn.

Khi điện áp nguồn đầu vào nào lớn hơn nó sẽ làm đóng Diode của nguồn kia và loại bỏ nguồn đó ra khỏi mạch. Do đó sẽ chỉ có một nguồn cấp vào cho AMS1117.

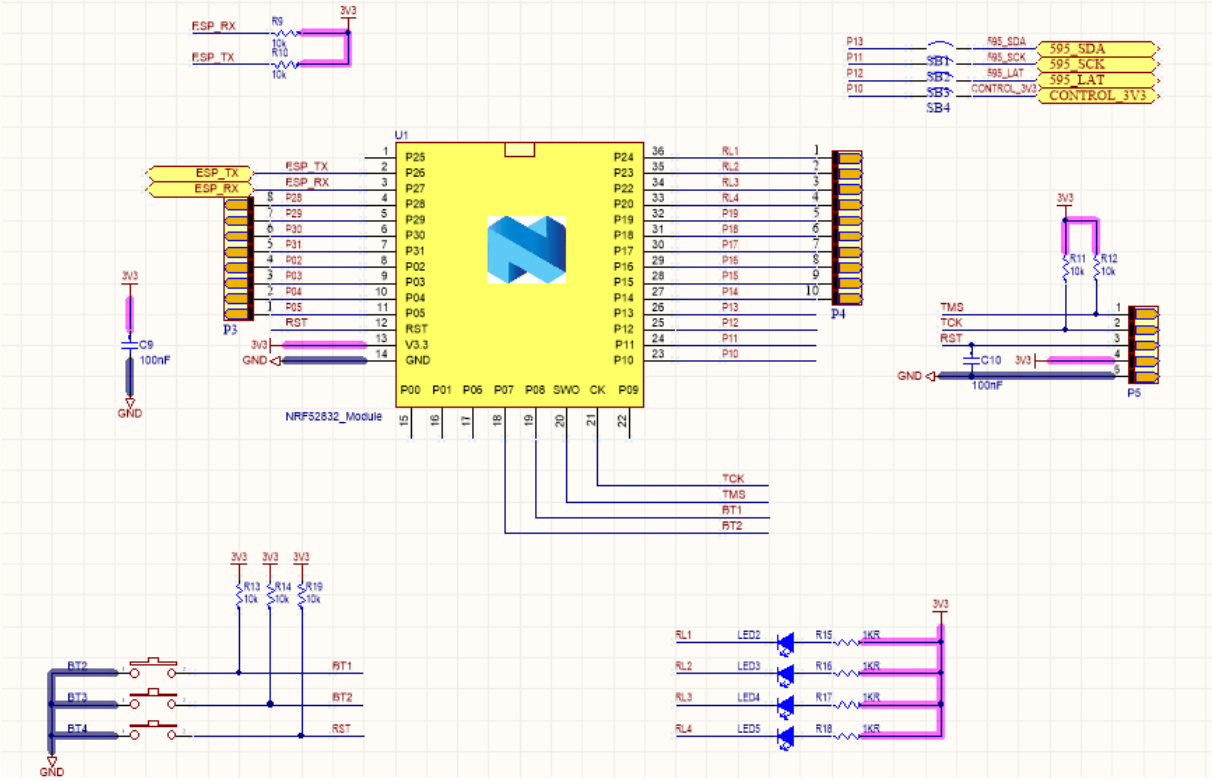
b) Khối Bluetooth

Lựa chọn module NRF52832 của Nordic, đây là dòng chip SoC cho phép điều khiển mạch công suất thông qua các chân IO.



Hình 46: Module Bluetooth NRF52832

Sơ đồ nguyên lý:

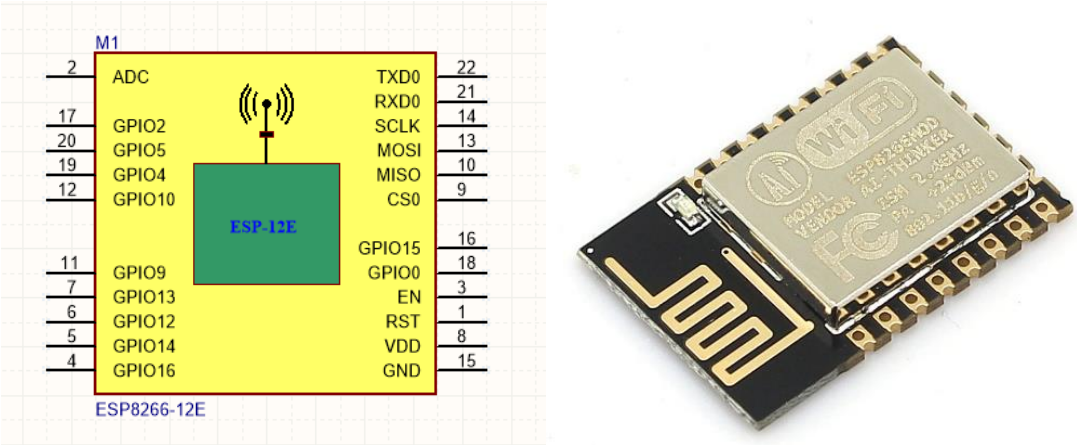


Hình 47: Sơ đồ nguyên lý mạch điều khiển bằng Bluetooth

Sẽ có 4 led báo trạng thái của 4 Relay.

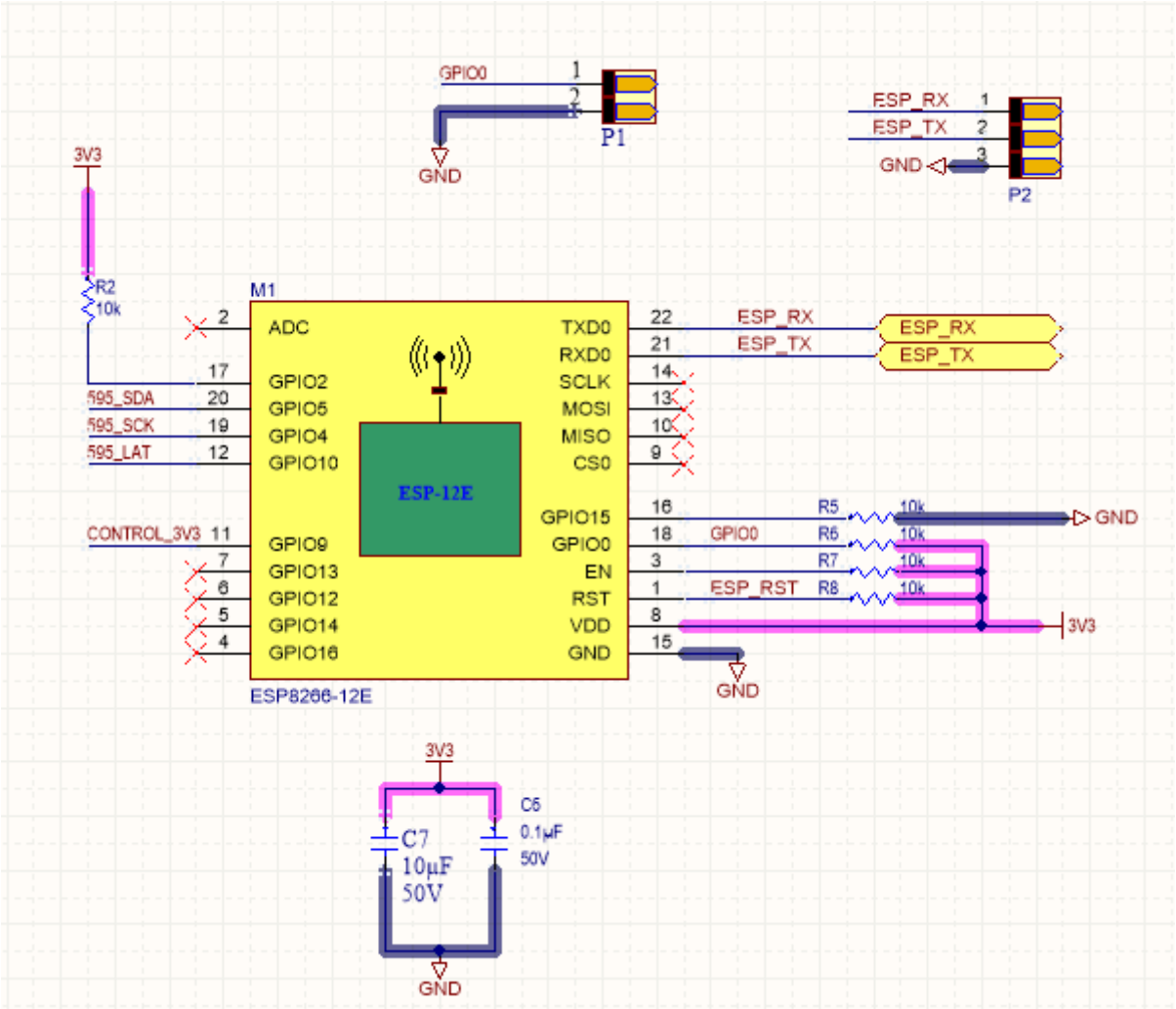
c) Khối Wifi

Lựa chọn module ESP8266 vốn đang khá phổ biến và dễ sử dụng.



Hình 48: Module Wifi ESP8266

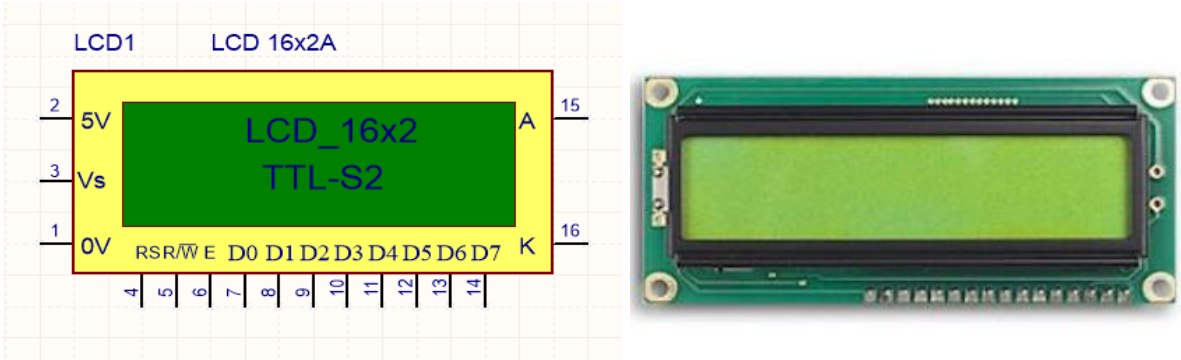
Sơ đồ nguyên lý:



Hình 49: Sơ đồ nguyên lý mạch Wifi

d) Khối hiển thị

Lựa chọn màn hình LCD 16x2 đủ khả năng đáp ứng các hoạt động cơ bản của mạch. Ngoài ra sử dụng thêm IC mở rộng chân 74HC595 để tiết kiệm chân điều khiển cho VDK.



Hình 50: Màn hình hiển thị LCD 16x2

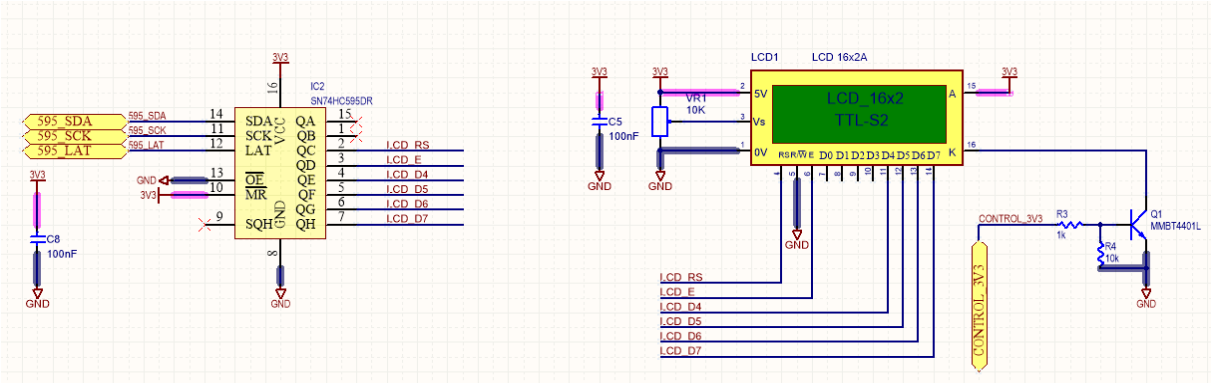
Sơ đồ chân:

- **0V:** chân kéo xuống đất
- **5V:** chân nguồn
- **A, K:** 2 cực của đèn led màn hình
- **Vs:** chân biến trở điều chỉnh độ sang màn hình
- **RS:** ghi lệnh hoặc ghi dữ liệu
- **R/W:** chọn chế độ đọc hoặc ghi
- **E:** chân cho phép
- **D0-D7:** 8 đường của bus dữ liệu dùng giao tiếp với LCD

Thông số kĩ thuật.

- Điện áp hoạt động: 3.3V
- Dòng tiêu thụ: 20ma

Sơ đồ nguyên lý.



Hình 51: Mạch ghép nối màn hình hiển thị LCD

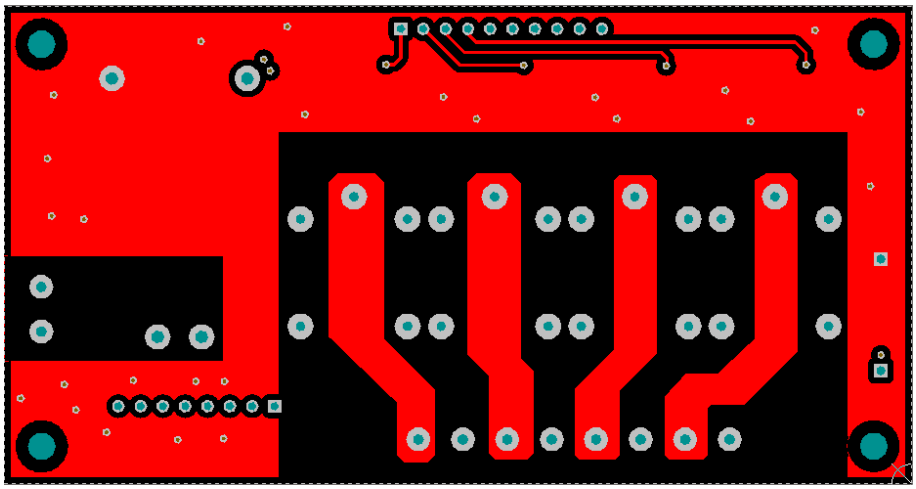
Sử dụng 3 chân đầu vào của 74HC595 để điều khiển LCD theo chế độ hiển thị 4 bit (4 chân từ D4-D7). Sử dụng 1 chân điều khiển để bật tắt LCD theo ý muốn.

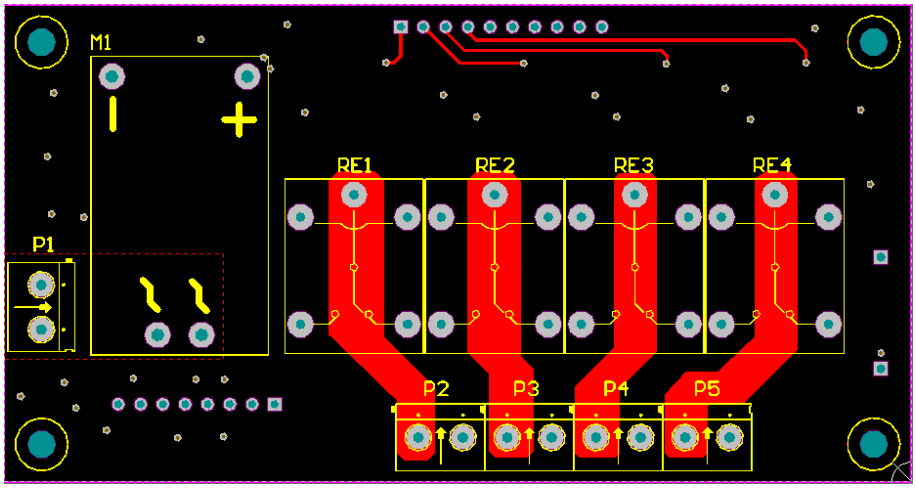
3.2.5. Thiết kế mạch Node, Role

Mạch được thiết kế tận dụng để hoạt động được ở 3 vai trò Gateway, Role và Node. Mạch Node và Role chỉ cần bỏ qua (không hàn) khối module Wifi.

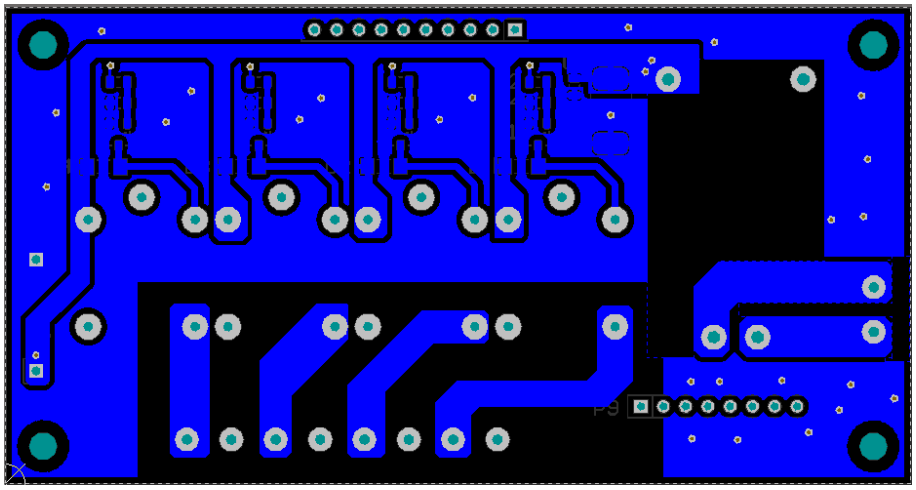
3.2.6. Giới thiệu mạch in

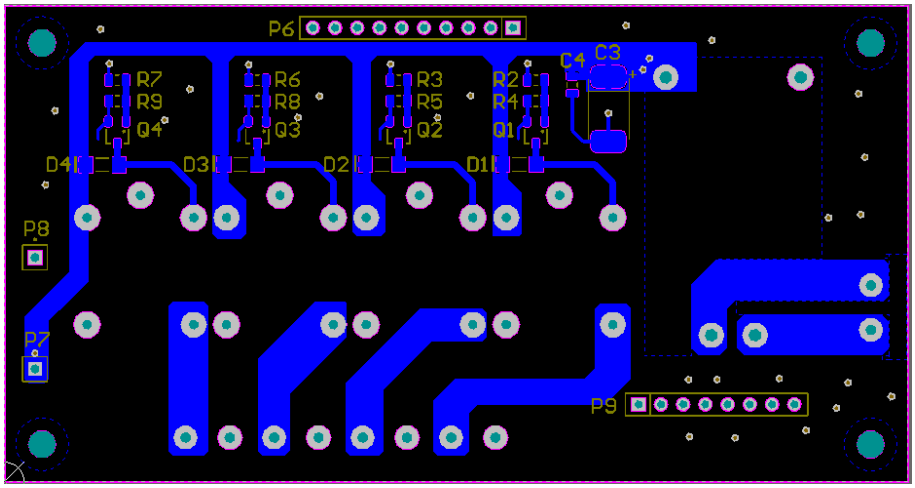
a) Mạch công suất





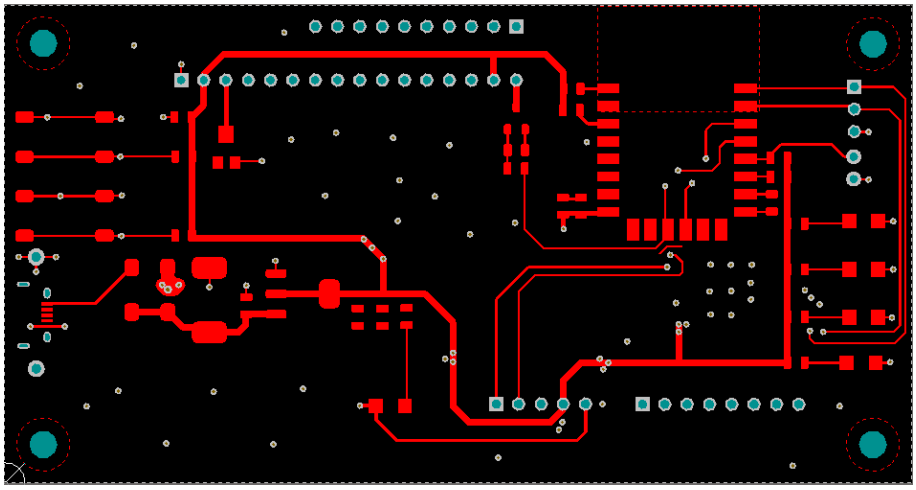
Hình 52: Hình ảnh Top Layer mạch điều khiển Relay

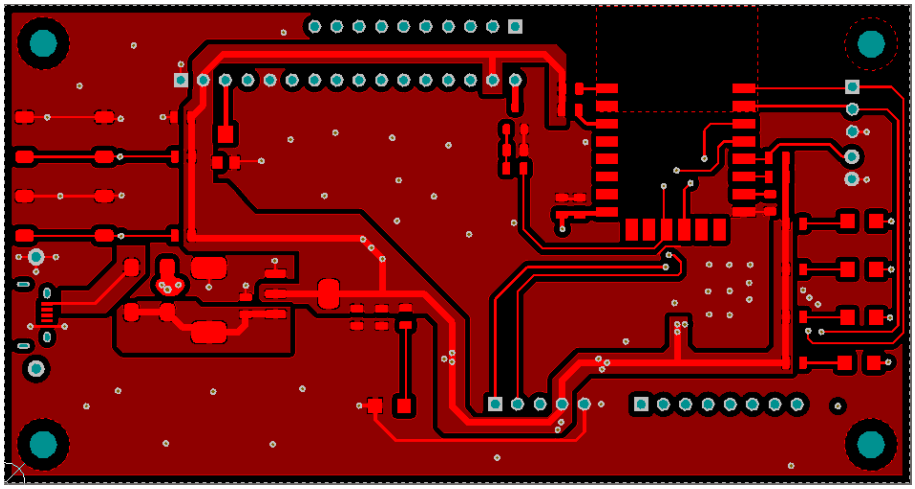


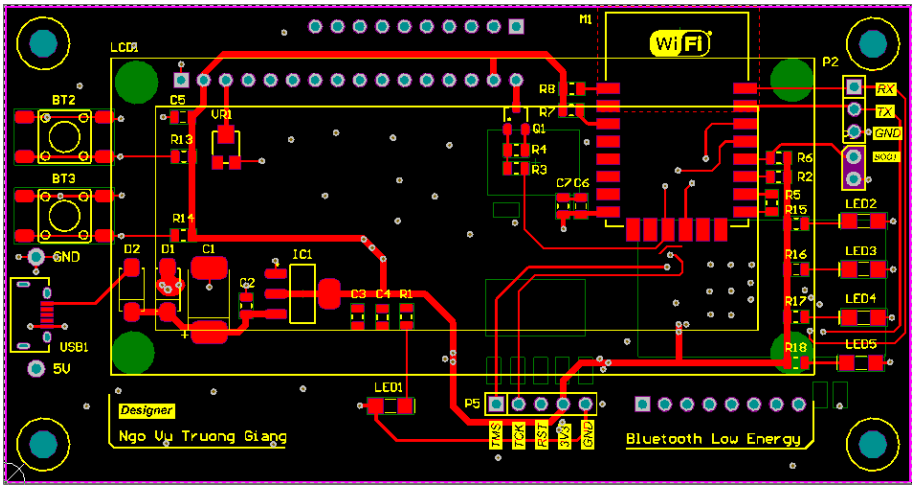


Hình 53: Hình ảnh Bottom Layer mạch điều khiển Relay

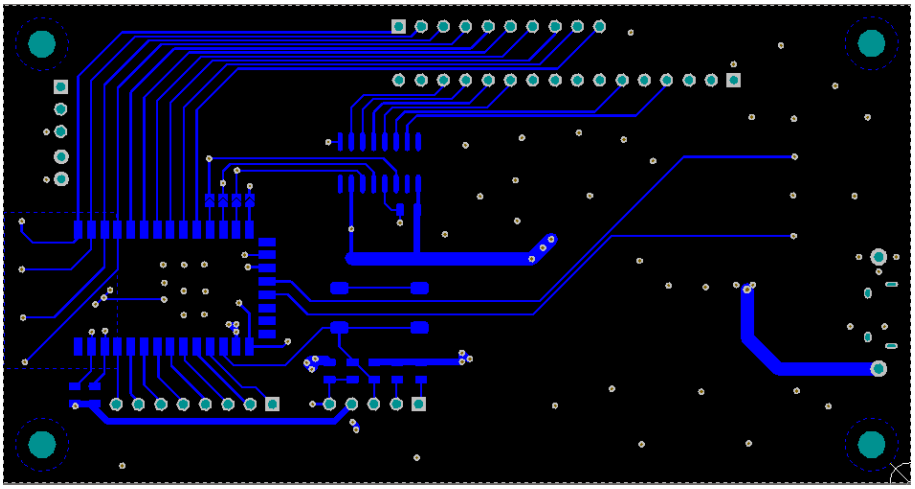
b) Mạch Node & Gateway.

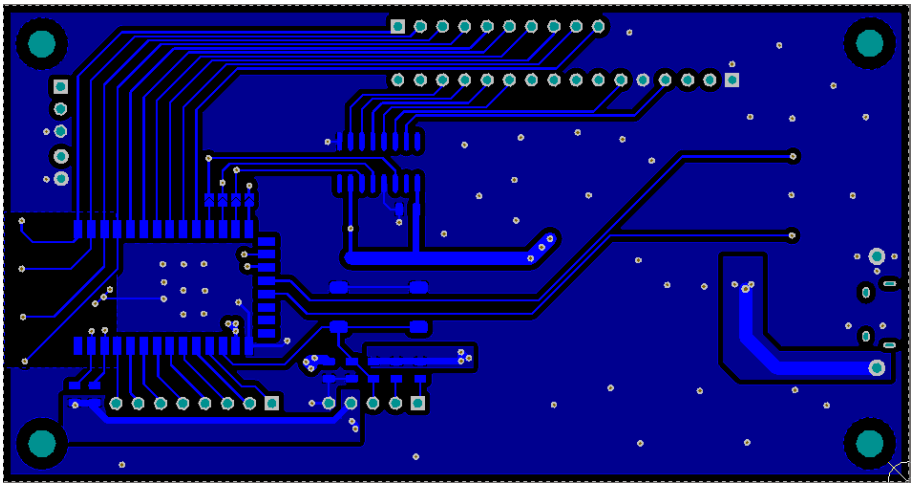


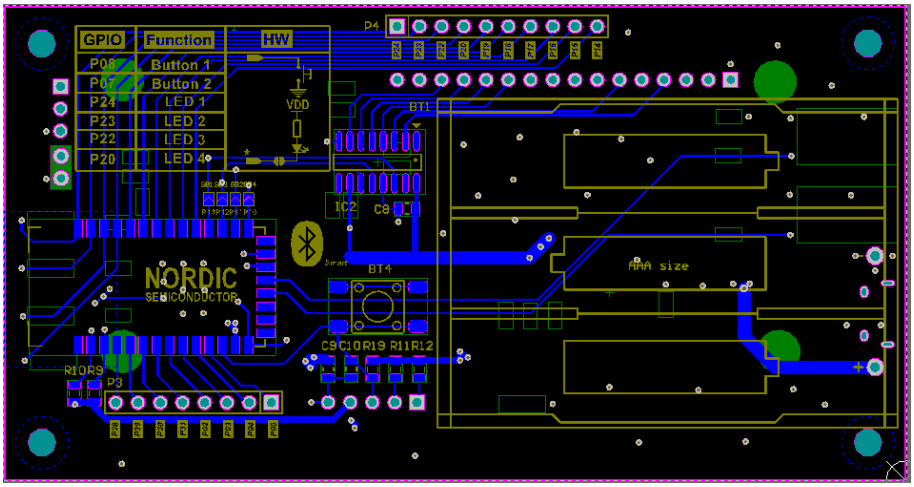




Hình 54: Hình ảnh Top Layer mạch điều khiển bằng Bluetooth







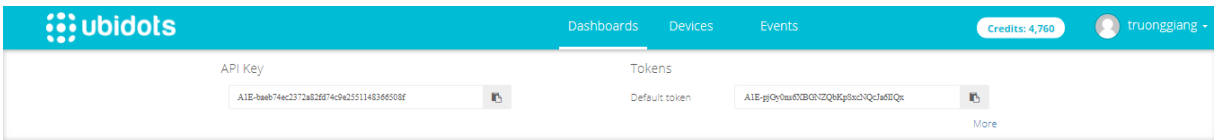
Hình 55: Hình ảnh Bottom Layer mạch điều khiển bằng Bluetooth

3.3. THIẾT KẾ PHẦN MỀM

3.3.1. Giới thiệu về Ubidots

Ubidots là một nền tảng nổi tiếng được xây dựng cho các ứng dụng Internet of Things. Nền tảng này cho phép tiếp nhận dữ liệu được gửi lên từ các thiết bị phần cứng như AVR, Arduino, STM32 … Ngoài khả năng giám sát, Ubidot còn cho phép điều khiển các thiết bị ngay từ trên giao diện của nó. Ubidots hỗ trợ rất nhiều dạng hiển thị mô phỏng như nút bấm, bảng, cột, bản đồ, kim số … Giúp cho mọi thứ trở nên trực quan gần gũi và dễ dàng cho người dùng. Hiện nay, Ubidot có tài khoản miễn phí và tài khoản tính phí. Tài khoản miễn phí tuy có nhiều hạn chế hơn nhưng cũng phần nào đáp ứng được các nhu cầu cơ bản về IOT.

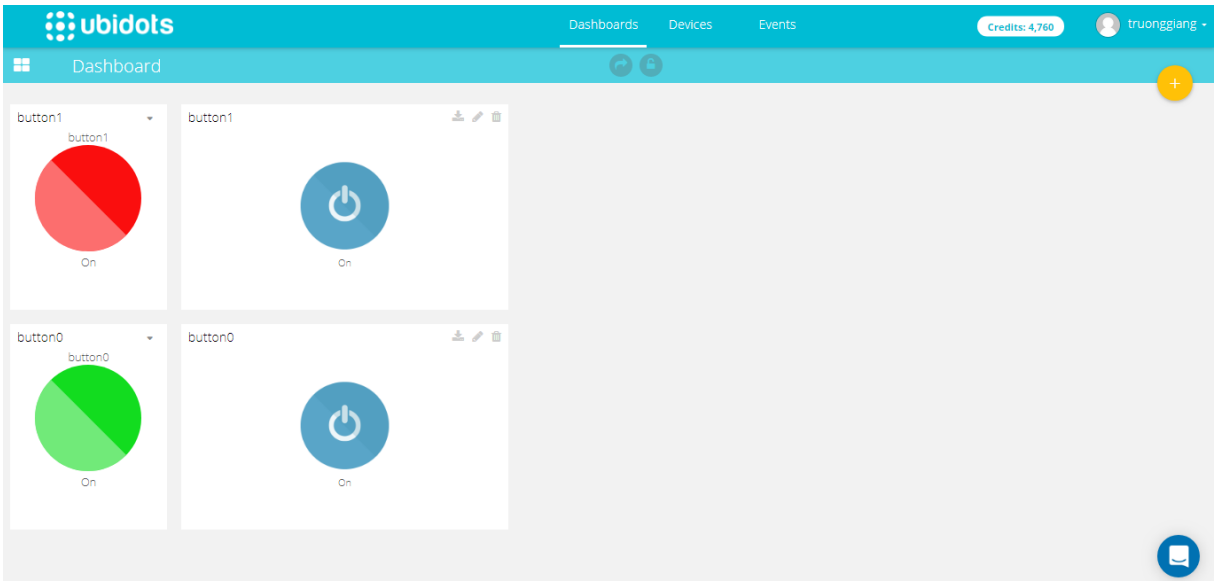
Để bắt đầu làm việc với Ubidots, chúng ta cần đăng kí một tài khoản miễn phí. Sau đó, mỗi tài khoản sẽ được cấp một API Key và Default Token duy nhất.



Hình 56: Tài khoản Ubidots

Ubitdots được sử dụng như một MQTT Broker với địa chỉ: things.ubidots.com qua cổng 1883, sử dụng tên đăng nhập là mã Default Token, Password để trống. Sau khi kết nối thành công đã có thể Subscribe và Publish đến các MQTT topic.

Giao diện Ubidot sau khi thiết lập:



Hình 57: Giao diện Web điều khiển

a) Publish các giá trị cho một biến

Publish tin nhắn vào topic có dạng:

- */v1.6/devices/{LABEL_DEVICE}/*

trong đó *{LABEL_DEVICE}* là tên thiết bị mà người dùng đặt.

Tin nhắn gửi đến viết dưới dạng JSON: *"LABEL_VARIABLE":value*

Một *LABEL_DEVICE* có thể có nhiều *LABEL_*VARIABLE.

Ví dụ: `publish("/v1.6/devices/environment","{"temperature": 50,"humidity":50}")`

b) Subscribe topic của một biến

Một MQTT Client có thể subscribe tới topic của một biến để nhận giá trị của biến đấy. Tin nhắn subscribe có dạng:

- */v1.6/devices/{LABEL_DEVICE}/{LABEL_VARIABLE}/lv*

Dữ liệu trả về là giá trị cuối cùng của biến.

Ví dụ: `subscribe("/v1.6/devices/environment/temperature/lv")`

3.3.2. Chương trình chính

a) Xây dựng platform

Do mạch sử dụng BLE là mạch tự vẽ thủ công, không nằm trong số các Development Kit do Nordic sản xuất nên khi sử dụng SDK sẵn của hãng chúng ta cần xây dựng lại platform riêng cho đúng với mạch sử dụng của chúng ta.

Để làm được điều này chúng ta cần sửa lại file *pca10040.h* trong project sử dụng

```
// LEDs definitions for PCA10040
```

```
#define LEDS_NUMBER 4
#define LED_START 20
#define LED_1 20
#define LED_2 22
#define LED_3 23
#define LED_4 24
#define LED_STOP 24
#define LEDS_ACTIVE_STATE 0
```

Định nghĩa các chân LED sẽ sử dụng, trong mạch chúng ta sử dụng 4 led tương ứng chân như trên.

```
// BUTTONs definitions for PCA10040
#define BUTTONS_NUMBER 4
#define BUTTON_START 10
#define BUTTON_1 10
#define BUTTON_2 11
#define BUTTON_3 12
#define BUTTON_4 13
#define BUTTON_STOP 13
#define BUTTON_PULL NRF_GPIO_PIN_PULLUP
```

Định nghĩa chân các nút bấm.

```
#define RX_PIN_NUMBER 27
#define TX_PIN_NUMBER 26
```

Định nghĩa chân UART.

Ngoài ta chúng ta cần định nghĩa các thông số trong file *sdk_config.h* để cho phép các ngoại vi sẽ sử dụng được hoạt động.

```
#ifndef BLE_DB_DISCOVERY_ENABLED
#define BLE_DB_DISCOVERY_ENABLED 1
#endif
```

Bật chế độ tìm kiếm thiết bị.

```
#ifndef NRF_BLE_GATT_ENABLED
#define NRF_BLE_GATT_ENABLED 1
#endif
```

Bật GATT module.

```
#ifndef BLE_NUS_C_ENABLED
#define BLE_NUS_C_ENABLED 1
#endif
```

Bật dịch vụ sẽ sử dụng trong project. Chúng ta sẽ sử dụng dịch vụ NUS, đây là dịch vụ Nodric UART Service cho phép truyền UART qua Bluetooth.

```
#ifndef GPIOTE_ENABLED
#define GPIOTE_ENABLED 1
```

```
#endif
```

Cho phép sử dụng ngắt của các chân IO.

```
#ifndef UART_ENABLED
#define UART_ENABLED 1
#endif
```

Cho phép ngoại vi UART hoạt động.

```
#ifndef UART0_ENABLED
#define UART0_ENABLED 1
#endif
```

Cho phép UART0 hoạt động.

```
#ifndef NRF_SDH_BLE_PERIPHERAL_LINK_COUNT
#define NRF_SDH_BLE_PERIPHERAL_LINK_COUNT 0
#endif

#ifndef NRF_SDH_BLE_CENTRAL_LINK_COUNT
#define NRF_SDH_BLE_CENTRAL_LINK_COUNT 3
#endif

#ifndef NRF_SDH_BLE_TOTAL_LINK_COUNT
#define NRF_SDH_BLE_TOTAL_LINK_COUNT 3
#endif
```

Định nghĩa số kết nối được xảy ra đồng thời.

b) Gói dữ liệu truyền giữa các thiết bị

Để đơn giản trong việc điều khiển giữa các thiết bị, em quy định một bản tin dữ liệu khá đơn giản để truyền giữa module Wifi và Gateway.

Bảng 6: Cấu trúc dữ liệu truyền giữa module Wifi và Gateway

Tên Node	Tên relay trên Node	Trạng thái relay	Kí tự kết thúc
Từ 0 → n	Từ 0 → n	0: mở	'@'
		1: đóng	

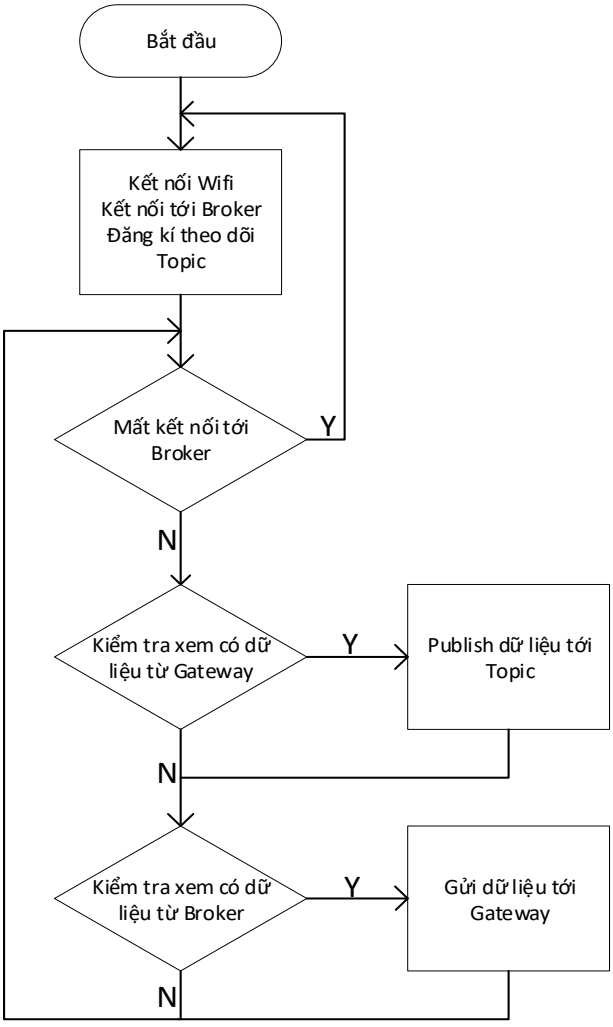
Giữa module Gateway và các Node:

Bảng 7: Cấu trúc dữ liệu truyền giữa Gateway và Node

Thiết bị gửi	Tên Node	Tên relay	Trạng thái	Hành động
0: điện thoại	Từ 0 → n	Từ 0 → n	0: mở	0: chỉ để xử lý không gửi đi
1: Web			1: đóng	1: tiếp tục gửi tới các Node và Gateway khác
2: Nút bấm				

c) Chương trình Wifi

Sơ đồ thuật toán:



Hình 58: Sơ đồ thuật toán hoạt động của module Wifi

Mô tả thuật toán:

- **B1**: Khởi tạo bằng cách nháy led để báo hiệu chương trình bắt đầu hoạt động, kết nối vào mạng Wifi có sẵn, ta đăng ký các thông tin kết nối: clientID, username, password, hàm mqtt_event xử lý sự kiện tin nhắn MQTT (hàm sẽ được gọi mỗi khi broker gửi cho node một sự kiện MQTT).
- **B2**: Sau khi đăng kí thành công, liên tục kiểm tra trạng thái kết nối của Wifi tới Broker, nếu mất kết nối lập tức kết nối lại.
- **B3**: Liên tục kiểm tra chân UART xem có dữ liệu truyền tới từ Gateway không, nếu có sẽ gửi tin nhắn Publish tương ứng tới Broker để cập nhật trạng thái hiện tại của các Relay ở các mạch Node.
- **B4**: Nếu có tin nhắn gửi tới từ Broker báo hiệu có sự thay đổi trạng thái Relay từ trên Web, sẽ gửi tin nhắn tương ứng qua UART tới Gateway để Gateway gửi tin nhắn điều khiển tới các mạch Node.

Câu lệnh kết nối đến Broker:

```
client.connect("mqtt://things.ubidots.com",
"A1E-pjGy0ns6XBGNZQbKpSxcNQcJa6lIQx", "")
```

Câu lệnh Publish giá trị tới Topic:

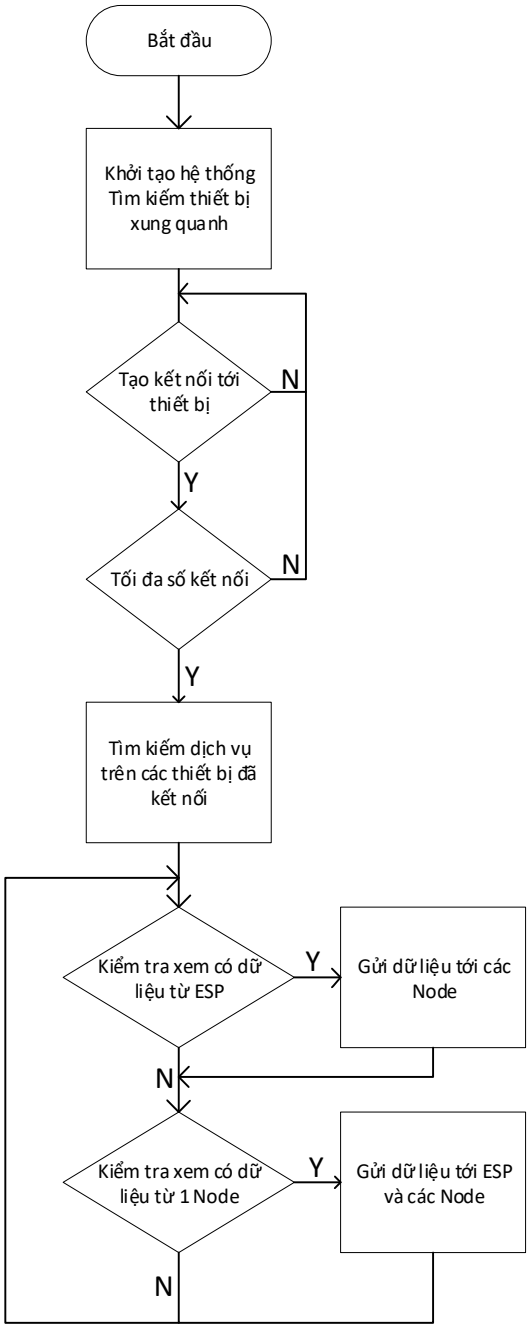
```
client.publish("/v1.6/devices/esp8266", "{\"button0\": 0}");
client.publish("/v1.6/devices/esp8266", "{\"button0\": 1}");
client.publish("/v1.6/devices/esp8266", "{\"button1\": 0}");
client.publish("/v1.6/devices/esp8266", "{\"button1\": 1}");
```

Câu lệnh Subscribe đến Topic:

```
client.subscribe("/v1.6/devices/esp8266/button0/lv");
client.subscribe("/v1.6/devices/esp8266/button1/lv");
```

d) Chương trình Gateway

Sơ đồ thuật toán



Hình 59: Sơ đồ thuật toán mạch Gateway

Mô tả thuật toán:

- **B1**: Tìm kiếm các Node xung quanh để kết nối. Hàm bắt đầu tìm kiếm:

```
static void scan_start(void)
{
ret_code_t ret;
```

```
ret = sd_ble_gap_scan_start(&m_scan_params);
APP_ERROR_CHECK(ret);
ret = bsp_indication_set(BSP_INDICATE_SCANNING);
APP_ERROR_CHECK(ret);
}
```

trong đó cần thiết lập các thông số cơ bản cho quá trình scan

```
#define SCAN_INTERVAL 0x00A0   // chu kì scan
#define SCAN_WINDOW 0x0050     // cửa sổ scan
#define SCAN_TIMEOUT 0x0000    // thời gian timeout khi scan
```

- **B2**: Khi nhận được bản tin quảng cáo, chúng ta sẽ tiến hành phân tích để tìm kiếm UUID của dịch vụ, nếu đúng với UUID cần tìm kiếm sẽ tiến hành kết nối 2 thiết bị. Sự kiện nhận được bản tin quảng cáo như sau:
  - *BLE_GAP_EVT_ADV_REPORT*: sự kiện phân tích xong bản quảng cáo phát ra từ các Node
  - Tìm kiếm dịch vụ UUID sử dụng hàm:

```
Static bool is_uuid_present(ble_uuid_t const * p_target_uuid,
ble_gap_evt_adv_report_t const * p_adv_report)
```

trong đó 2 thông số truyền vào là UUID cần tìm kiếm và gói bản tin quảng cáo nhận được, nếu đúng UUID cần tìm kiếm sẽ trả về true.

Để có thể thành lập kết nối chúng ta cần truyền cho thiết bị Node các thông số của kết nối như sau:

```
static ble_gap_conn_params_t const m_connection_param =
{
(uint16_t) MIN_CONNECTION_INTERVAL, // Chu kì kết nối tối thiểu
(uint16_t) MAX_CONNECTION_INTERVAL, // Chu kì kết nối tối đa
(uint16_t) SLAVE_LATENCY,  // Số sự kiện kết nối có thể bỏ qua
(uint16_t) SUPERVISION_TIMEOUT    // Thời gian timeout
};
```

Khi một kết nối được thành lập các sự kiện có thể xảy ra là:

BLE_GAP_EVT_CONNECTED: sự kiện kết nối thành công đến một Node.

BLE_GAP_EVT_TIMEOUT: sự kiện quá thời gian xử lý.

Trong sự kiện *BLE_GAP_EVT_CONNECTED* chúng ta kiểm tra số kết nối đến các thiết bị Node đã đủ với định nghĩa chưa, nếu chưa tiếp tục scan để thành lập các kết nối khác đồng thời sẽ tìm kiếm dịch vụ trên các Node vừa kết nối bằng cách sử dụng hàm:

```
ble_db_discovery_start(&m_db_disc[p_gap_evt->conn_handle],
p_ble_evt->evt.gap_evt.conn_handle);
```

Các sự kiện có thể nhận được là:

- *BLE_NUS_C_EVT_DISCOVERY_COMPLETE*: tìm kiếm dịch vụ thành công

- *BLE_NUS_C_EVT_DISCONNECTED*: dịch vụ đã bị ngắt kết nối

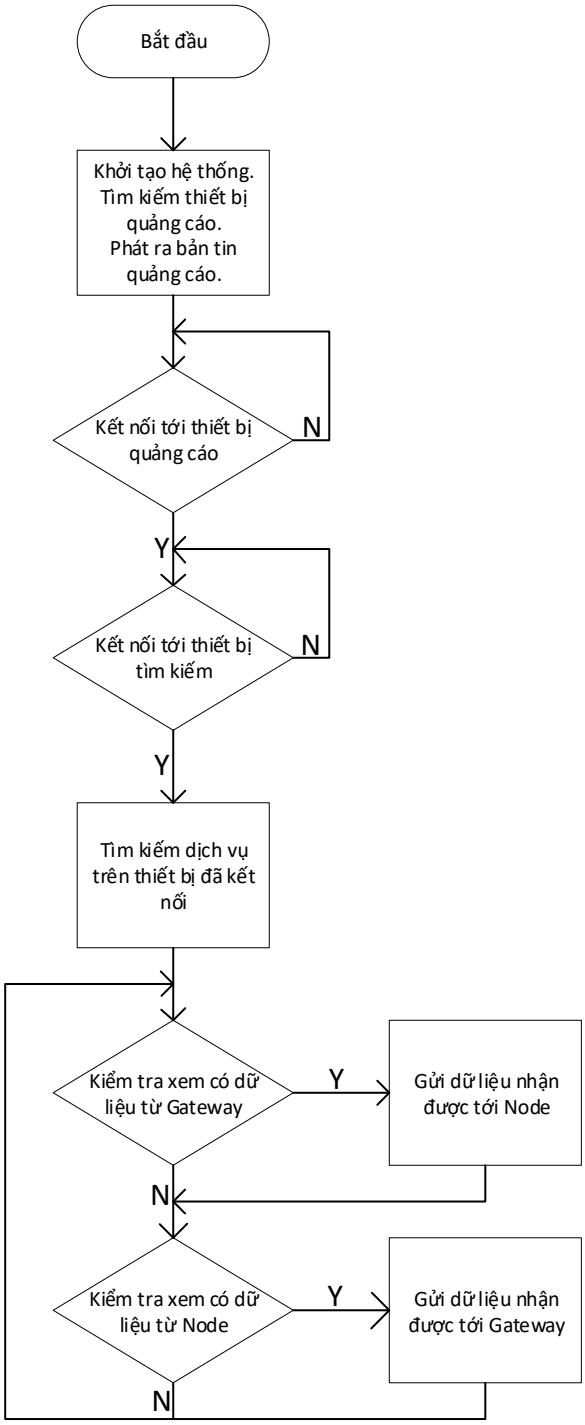
Sau khi tìm kiếm dịch vụ thành công, chúng ta cần cho phép Notification để các thiết bị có thể truyền nhận dữ liệu cho nhau. Sử dụng câu lệnh sau

```
ble_nus_c_tx_notif_enable(p_ble_nus_c);
```

- **B3**: Kiểm tra xem có dữ liệu nhận được từ các Node hoặc từ module Wifi không.
  - Nếu có dữ liệu nhận được từ các Node, chương trình sẽ xảy ra sự kiện là *BLE_NUS_C_EVT_NUS_TX_EVT*: chỉ thị Gateway nhận được dữ liệu từ Node. Trong sự kiện này, ta xử lý dữ liệu nhận được bằng cách gửi lệnh tương ứng tới module Wifi và các Node còn lại.
  - Nếu có dữ liệu nhận được từ module Wifi, chương trình sẽ nhảy vào sự kiện. *APP_UART_DATA_READY*: chỉ thị dữ liệu UART đã được nhận. Trong sự kiện này, ta xử lý dữ liệu bằng cách gửi lệnh tương ứng tới tất cả các Node.

e) Chương trình Role

Sơ đồ thuật toán:



Hình 60: Sơ đồ thuật toán mạch Role

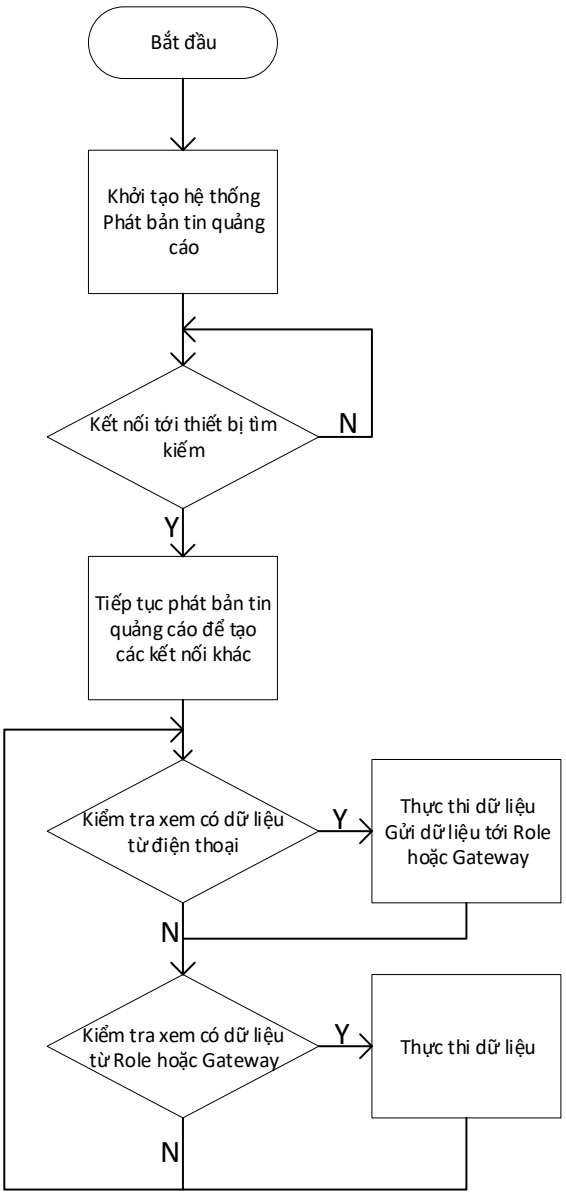
Mô tả thuật toán:

- **B1**: Tìm kiếm tới các Node đã được quy định sẵn, tránh kết nối nhầm tới Node khác, đồng thời phát ra bản tin quảng cáo để kết nối tới Gateway.

- **B2**: Kiểm tra xem có nhận được dữ liệu từ Gateway hoặc Node không để có hành động tương ứng.
  - Nếu nhận dữ liệu từ Gateway thì gửi dữ liệu đó tới các Node đã liên kết.
  - Nếu nhận dữ liệu từ Node thì gửi dữ liệu đó tới Gateway.

f) Chương trình Node

Sơ đồ thuật toán:



Hình 61: Sơ đồ thuật toán mạch Node

Mô tả thuật toán:

- **B1**: Phát ra bản tin quảng cáo ra xung quanh để tìm kiếm thiết bị có thể kết nối. Khởi tạo bản tin quảng cáo sử dụng hàm sau:

```
static void advertising_init(void)
{
uint32_t err_code;
ble_advertising_init_t init;
memset(&init, 0, sizeof(init));
init.advdata.name_type = BLE_ADVDATA_FULL_NAME;
init.advdata.include_appearance = false;
init.advdata.flags = BLE_GAP_ADV_FLAGS_LE_ONLY_LIMITED_DISC_MODE;
init.srdata.uuids_complete.uuid_cnt = sizeof(m_adv_uuids) /
sizeof(m_adv_uuids[0]);
init.srdata.uuids_complete.p_uuids = m_adv_uuids;
init.config.ble_adv_fast_enabled = true;
init.config.ble_adv_fast_interval = APP_ADV_INTERVAL;
init.config.ble_adv_fast_timeout = APP_ADV_TIMEOUT_IN_SECONDS;
init.evt_handler = on_adv_evt;
err_code = ble_advertising_init(&m_advertising, &init);
APP_ERROR_CHECK(err_code);
ble_advertising_conn_cfg_tag_set(&m_advertising,
APP_BLE_CONN_CFG_TAG);
}
```

Trong bản tin quảng cáo này đã được thêm vào tên phát ra quảng cáo và UUID của dịch vụ, thiết bị scan sẽ dựa và 2 thông số này để thành lập kết nối.

Để bắt đầu quảng cáo sử dụng hàm sau:

```
ble_advertising_start(&m_advertising, BLE_ADV_MODE_FAST);
```

Các sự kiện có thể xảy ra là:

- *BLE_ADV_EVT_FAST*: chế độ quảng cáo nhanh được bắt đầu
- *BLE_ADV_EVT_IDLE*: không có sự kết nối nào diễn ra, sự kiện này xảy ra sau một thời gian timeout khi không có thiết bị scan nào khởi tạo kết nối, thiết bị sẽ đi vào chế độ ngủ để tiết kiệm năng lượng.

- **B2**: Thành lập kết nối. Sau khi nhận được bản tin yêu cầu kết nối từ Central, các sự kiện sau có thể xảy ra:
  - *BLE_GAP_EVT_CONNECTED*: kết nối được thành lập
  - *BLE_GAP_EVT_DISCONNECTED*: mất kết nối

Trong sự kiện *BLE_GAP_EVT_CONNECTED*, chúng ta sẽ kiểm tra xem đủ số lượng kết nối như định nghĩa chưa nếu chưa đủ sẽ tiếp tục phát ra bản tin quảng cáo để thành lập các kết nối khác.

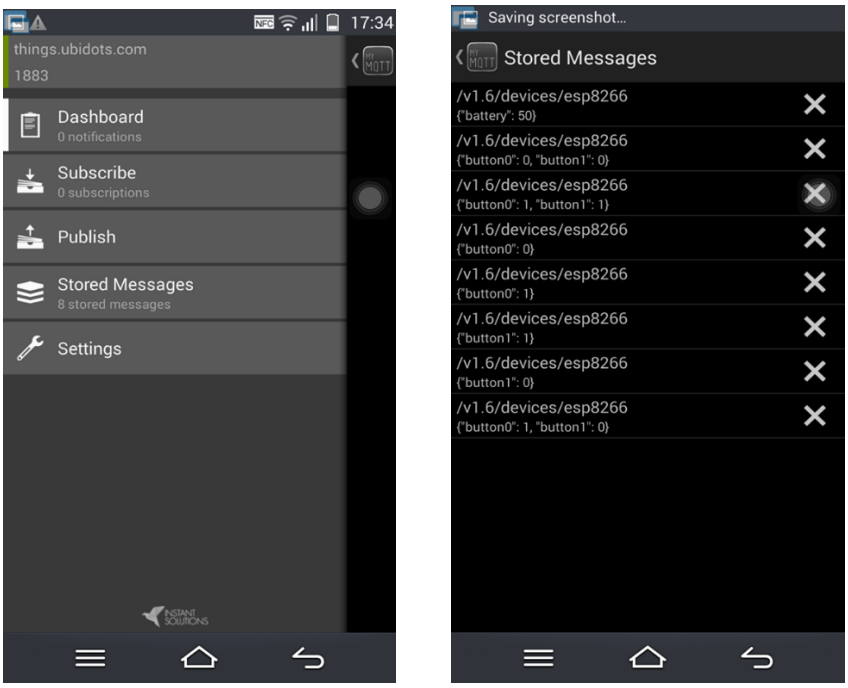
- **B3**: Kiểm tra dữ liệu nhận được. Khi có dữ liệu nhận được sẽ xảy ra sự kiện:

BLE_NUS_EVT_RX_DATA: nhận được dữ liệu từ Bluetooth.

Nếu dữ liệu nhận được từ điện thoại, xử lý dữ liệu bằng cách gửi mã lệnh tương ứng tới Role hoặc Gateway. Nếu dữ liệu nhận được từ Role hoặc Gateway, thực thi lệnh nhận được.

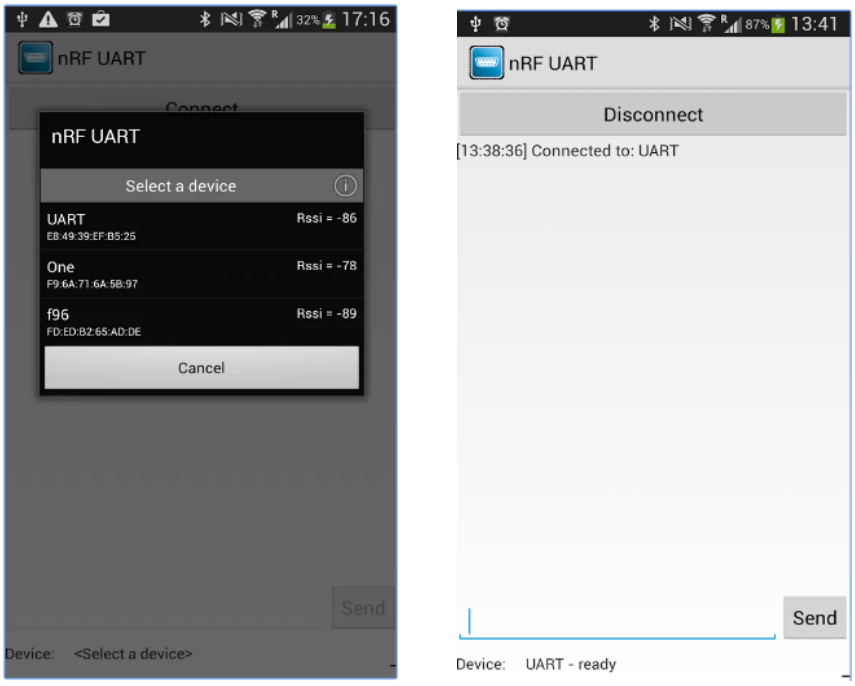
g) Phần mềm điện thoại

Giao diện MQTT Client



Hình 62 Giao diện phần mềm MQTT

Giao diện nRF UART



Hình 63: Giao diện phần mềm Bluetooth

CHƯƠNG 4: KẾT QUẢ ĐẠT ĐƯỢC

4.1. KẾT QUẢ ĐẠT ĐƯỢC

Sau thời gian tìm hiểu và thực hiện đề tài, em đạt được một số kết quả nhất định như sau:

- Xây dựng được mạng truyền thông giữa các thiết bị BLE. Tiến hành chạy thực tế cho thấy các thiết bị chạy ổn định và chính xác. Khoảng cách truyền đo được giữa 2 thiết bị bị chắn bởi 2 bức tường kín rơi vào khoảng gần 15m. Ở không gian lý tưởng ít vật cản phạm vi kết nối lên tới 30m bán kính.
- Điều khiển thiết bị thông qua Web và điện thoại trong chế độ có Wifi và không có Wifi. Tốc độ đáp ứng nhanh và chính xác.
- Tiến hành thiết kế và lắp ráp mạch, mạch chạy ổn định như mong muốn.

Các bước tiến hành chạy kiểm tra mạch:

- Cấp nguồn cho các mạch Node và Role, các mạch sẽ tự động tạo kết nối và báo hiệu ra chuỗi đèn led, mỗi module Bluetooth sẽ có 2 đèn led báo hiệu, LED1 báo hiệu quá trình Advertising hoặc Scanning, LED2 báo hiệu thành lập kết nối. Ví dụ trong quá trình Advertising hoặc Scanning thì LED1 sẽ nhấp nháy liên tục, khi một kết nối được thành lập thì LED2 sẽ sáng, khi số kết nối bằng 0 LED2 sẽ tắt đồng thời LED1 tiếp tục nhấp nháy báo hiệu mạch đang Advertising hoặc Scanning. Nếu số kết nối chưa đủ, LED2 sáng và LED1 tiếp tục nhấp nháy báo hiệu mạch vẫn đang tìm kiếm thực hiện các kết nối tiếp theo.
- Cấp nguồn cho mạch Gateway, đèn led gắn trên module Wifi sẽ nhấp nháy 3 lần liên tiếp báo hiệu Wifi đang được khởi tạo và kết nối với Access Point. Khi led ngừng nhấp nháy sẽ báo hiệu kết nối Web thành công, sẵn sàng nhận lệnh điều khiển từ Web. Module Bluetooth trên mạch Gateway cũng được chỉ thị giống như trên các mạch Node thông qua LED1 và LED2, khác với đèn led trên module Wifi. Khi LED1 ngưng nhấp nháy và LED2 giữ trạng thái sáng báo hiệu số kết nối đã thành lập đủ, sẵn sàng thực hiện các chức năng của mạch.
- Khi tiến hành điều khiển mạch từ điện thoại hoặc Web, quá trình cập nhật trạng thái các Relay lên Web phụ thuộc vào tốc độ đường truyền internet của module Wifi. Thông thường rơi vào khoảng 0-1s.
- Mạch sẽ chỉ nhận các kí tự điều khiển đã được quy định sẵn, trong trường hợp nhận được các kí tự điều khiển sai, mạch sẽ tự động gửi lại chuỗi “unknown” cho thiết bị điều khiển.
- Ở chế độ online hoàn toàn có thể điều khiển thông qua 2 phần mềm sử dụng Wifi và Bluetooth, khi mất kết nối internet, module Wifi không hoạt động, mạch bị tách khỏi

Web Server và vẫn có thể tiếp tục điều khiển bằng điện thoại nhờ phần mềm sử dụng Bluetooth.

Dưới đây là một số hình ảnh mô tả kết quả em đã đạt được.

Hình 64: Mô tả kết quả đạt được

4.2. HẠN CHẾ

Trong quá trình thực nghiệm, đề tài vẫn còn một số hạn chế như sau:

- Số kết nối vẫn chưa đạt được tối đa cho mỗi thiết bị.
- Chưa bổ sung mạch khuếch đại cho module BLE để tăng khoảng cách truyền.
- Chưa thiết kế lại giao diện phần mềm riêng trên điện thoại.
- Giao diện Web Server vẫn đơn giản chưa có tính chuyên nghiệp.

4.3. HƯỚNG PHÁT TRIỂN

Bên cạnh một số kết quả đạt được, vẫn còn khá nhiều hạn chế và cần cải thiện bổ sung thêm nhiều tính năng có thể kể tới như:

- Cải thiện tối ưu chất lượng phần cứng.
- Hoàn thiện phần mềm điện thoại.
- Xây dựng được một Server quản lý riêng.
- Nâng cấp số lượng kết nối đồng thời của một thiết bị để mở rộng mạng cảm biến.
- Tăng tính ổn định của mỗi thiết bị.
- Sử dụng thêm tính năng truyền tốc độ cao để truyền hình ảnh hoặc video phục vụ thêm các Module Camera.
- Đồng bộ các thiết bị sử dụng Bluetooth hiện nay (như loa Bluetooth, các thiết bị theo dõi sức khỏe, Beacon) để đồng bộ vào mạng gia đình dễ dàng quản lý theo dõi.

Tài liệu tham khảo

[1] Core_Bluetooth v5.0

[2] Getting Started with Bluetooth Low Energy

[3] Inside Bluetooth Low Energy, Second Edition

[4] nRF52832_PS_v1.4

[5] NordicSemi_eBook_Bluetooth_Low_Energy_development

[6] iot-starter

[7] https://devzone.nordicsemi.com/blogs/

[8] http://infocenter.nordicsemi.com/index.jsp

[9] https://esp8266.vn/

Ngoài ra còn rất nhiều nguồn từ các trang Web trực tuyến.