BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
KHOA ĐÀO TẠO CHẤT LƯỢNG CAO

ĐỒ ÁN MÔN HỌC 2

NGHIÊN CỨU HỆ THỐNG BLUETOOTH VÀ LORA

SVTH :

VÕ ĐỨC THUẬN 21161085

NGUYỄN ĐÌNH HIẾU 21161049

Khóa : 2021

Ngành : CNKT Điện tử -Viễn thông

GVHD: ThS. NGUYỄN NGÔ LÂM

TP. HỒ CHÍ MINH – THÁNG 1/ NĂM 2025

ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM
KHOA ĐÀO TẠO CHẤT LƯỢNG CAO
www.fhq.hcmute.edu.vn

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

Tp. Hồ Chí Minh, ngày 6 tháng 01 năm 2025

NHIỆM VỤ ĐỒ ÁN MÔN HỌC

Họ và tên sinh viên: Võ Đức Thuận 21161085
Nguyễn Đình Hiếu 21161049
Ngành:Công Nghệ Kỹ Thuật Điện tử - Viễn thông Lớp: 21161CLVT2A
Giảng viên hướng dẫn: ThS. Nguyễn Ngô Lâm
Ngày nhận đề tài: Ngày nộp đề tài:

1. Tên đề tài: Nghiên cứu hệ thống Bluetooth và lora
2. Các số liệu, tài liệu ban đầu:
Kiến thức cơ bản về các môn Mạch điện, Điện tử cơ bản, Vi xử lý, IOT.
3. Nội dung thực hiện đề tài:
   - Thiết kế hệ thống
   - Vẽ sơ đồ nguyên lý
   - Lập trình cho hệ thống
   - Chỉnh sửa và kiểm tra mạch
   - Viết báo cáo
4. Sản phẩm:
Hệ thống Bluetooth và Lora

GIẢNG VIÊN HƯỚNG DẪN

ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM
KHOA ĐÀO TẠO CHẤT LƯỢNG CAO
www.fhq.hcmute.edu.vn

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

PHIẾU NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

Họ và tên Sinh viên: Võ Đức Thuận 21161085
Nguyễn Đình Hiếu 21161049
Ngành:Công Nghệ Kỹ Thuật Điện tử - Viễn thông
Tên đề tài: Nghiên cứu hệ thống Bluetooth và Lora
Họ và tên Giáo viên hướng dẫn: ThS. Nguyễn Ngô Lâm

NHẬN XÉT

1. Về nội dung đề tài & khối lượng thực hiện:

.....
.....
.....
.....

2. Ưu điểm:

.....
.....
.....
.....

3. Khuyết điểm:

.....
.....
.....

4. Đề nghị cho bảo vệ hay không?

.....

5. Đánh giá loại:

.....

6. Điểm:…….(Bằng chữ:.....)

.....

Tp. *Hồ Chí Minh, ngày 06 tháng 01 năm 2025*

Giáo viên hướng dẫn

ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM
KHOA ĐÀO TẠO CHẤT LƯỢNG CAO
www.fhq.hcmute.edu.vn

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

PHIẾU NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN

Họ và tên Sinh viên: Võ Đức Thuận 21161085
Nguyễn Đình Hiếu 21161049
Ngành:Công Nghệ Kỹ Thuật Điện tử - Viễn thông
Tên đề tài: Nghiên cứu hệ thống Bluetooth và Lora
Họ và tên Giáo viên phản biện: .....
.....

NHẬN XÉT

1. Về nội dung đề tài & khối lượng thực hiện:

.....
.....
.....
.....

2. Ưu điểm:

.....
.....
.....
.....

3. Khuyết điểm:

.....
.....

4. Đề nghị cho bảo vệ hay không?

.....

5. Đánh giá loại:

.....

6. Điểm:…….(Bằng chữ:.....)

.....

Tp. *Hồ Chí Minh, ngày 06 tháng 01 năm 2025*

Giáo viên phản biện

LỜI CẢM ƠN

Để hoàn thành báo cáo đồ án môn học 2 chuyên ngành Công nghệ Kỹ thuật Điện tử - Viễn thông trước hết em xin gửi đến quý Thầy/Cô trong khoa Đào tạo Chất lượng cao, trường Đại học Sư Phạm Kỹ Thuật Thành Phố Hồ Chí Minh lời cảm ơn chân thành. Đặc biệt, thầy **Nguyễn Ngô Lâm** đã tận tình hướng dẫn, giúp đỡ và tạo điều kiện thuận lợi cho em trong suốt quá trình thực hiện đồ án. Em xin gửi đến thầy lời cảm ơn chân thành và sâu sắc nhất.

Đồng thời, em cũng xin cảm ơn đến các bạn bè đã hỗ trợ, đóng góp ý kiến cũng như chia sẻ kinh nghiệm để em hoàn thành tốt đề tài.

Mặc dù đã cố gắng hết sức, nhưng do lượng kiến thức còn eo hẹp nên không tránh khỏi những thiếu sót. Do vậy, em rất mong nhận được sự góp ý quý báu của Thầy/Cô để có thể hoàn thiện và tốt hơn nữa cũng như tích lũy kinh nghiệm để hoàn thành tốt báo cáo đồ án tốt nghiệp sau này.

Sau cùng, em kính chúc quý thầy cô thật dồi dào sức khỏe, luôn tràn đầy nhiệt huyết cùng với thành công trong sự nghiệp cao quý.

Em xin chân thành cảm ơn!

MỤC LỤC

PHẦN I: HỆ THỐNG BLUETOOTH

CHƯƠNG 1: TỔNG QUAN VỀ HỆ THỐNG

1.1 GIỚI THIỆU

Trong thời đại công nghệ số, kết nối không dây đang trở thành một phần không thể thiếu trong cuộc sống hiện đại. Hệ thống Bluetooth, với khả năng truyền dữ liệu không dây trong phạm vi ngắn, đã đóng góp quan trọng vào việc tối ưu hóa sự tiện lợi và hiệu quả của các thiết bị thông minh. Công nghệ này không chỉ giúp kết nối các thiết bị điện tử mà còn mở ra những ứng dụng đa dạng trong các lĩnh vực như IoT, y tế, ô tô, và giải trí.

Bluetooth hoạt động dựa trên sóng radio tần số 2.4 GHz, cho phép truyền dữ liệu nhanh chóng và ổn định giữa các thiết bị. Một hệ thống Bluetooth điển hình bao gồm các thành phần chính như bộ phát, bộ thu, và giao thức truyền thông, đảm bảo tính tương thích và bảo mật cao. Với sự phát triển của các phiên bản mới như Bluetooth 5.0 và 5.3, công nghệ này đã cải thiện đáng kể về tốc độ, phạm vi hoạt động, và khả năng tiêu thụ năng lượng thấp, phù hợp với các thiết bị di động và cảm biến thông minh.

Trong bối cảnh hiện nay, việc sử dụng Bluetooth không chỉ giúp nâng cao trải nghiệm người dùng mà còn góp phần vào xu hướng xây dựng các hệ thống thông minh và bền vững. Khả năng kết nối đơn giản, linh hoạt, cùng với chi phí triển khai thấp đã biến Bluetooth trở thành một lựa chọn phổ biến trong việc phát triển các giải pháp công nghệ tiên tiến.

1.2. MỤC TIÊU NGHIÊN CỨU:

Mục tiêu của đề tài này là thiết kế và xây dựng một hệ thống truyền dữ liệu qua Bluetooth với các chức năng sau:

-Kết nối thiết bị: Thiết lập kết nối không dây ổn định giữa các thiết bị thông qua công nghệ Bluetooth.

-Truyền dữ liệu: Hệ thống có khả năng gửi và nhận dữ liệu nhanh chóng và chính xác trong phạm vi hoạt động của Bluetooth.

-Hiển thị thông tin: Cung cấp giao diện hiển thị dữ liệu nhận được hoặc trạng thái kết nối trên màn hình LCD hoặc thiết bị tương ứng.

-Điều khiển thủ công: Cho phép người dùng khởi tạo hoặc ngắt kết nối Bluetooth thông qua các nút nhấn hoặc giao diện điều khiển.

Mục tiêu cuối cùng là tạo ra một hệ thống truyền Bluetooth hoạt động ổn định, dễ sử dụng, và có khả năng ứng dụng thực tế trong các giải pháp IoT và thiết bị thông minh.

1.3. PHẠM VI NGHIÊN CỨU:

Phạm vi của đề tài bao gồm:

- Thiết kế mạch điện:
  Thiết kế và xây dựng mạch điện bao gồm module Bluetooth (như HC-05 hoặc BLE), vi điều khiển, nguồn cấp, và các thiết bị ngoại vi như nút nhấn, màn hình LCD hoặc LED hiển thị.
- Phát triển phần mềm:
  Lập trình vi điều khiển (ESP 32) để:
  - Thiết lập kết nối Bluetooth giữa các thiết bị.
  - Truyền và nhận dữ liệu không dây qua Bluetooth.
  - Hiển thị trạng thái và thông tin dữ liệu trên màn hình LCD.
- Lắp ráp và thử nghiệm:
  Lắp ráp các linh kiện, kiểm tra và hiệu chỉnh hệ thống để đảm bảo kết nối Bluetooth ổn định, dữ liệu truyền nhận chính xác và hệ thống hoạt động đáng tin cậy trong các điều kiện thực tế.
- Tài liệu hóa:
  Ghi chép chi tiết về quá trình thiết kế, lập trình, và thử nghiệm hệ thống, bao gồm sơ đồ mạch điện, mã nguồn chương trình, và các kết quả thử nghiệm.

Phạm vi này đảm bảo hệ thống truyền Bluetooth được thiết kế toàn diện, dễ triển khai, và sẵn sàng cho các ứng dụng thực tế trong IoT hoặc các thiết bị thông minh.

1.4. CẤU TRÚC HỆ THỐNG:

Hệ thống truyền dữ liệu qua Bluetooth bao gồm các khối chính sau:

- **Khối nguồn**:
  Cung cấp năng lượng cho toàn bộ hệ thống, bao gồm module Bluetooth, vi điều khiển, và các thiết bị ngoại vi. Khối nguồn thường sử dụng pin hoặc bộ chuyển đổi điện áp, đảm bảo cung cấp điện áp ổn định và phù hợp với các linh kiện.
- **Khối điều khiển trung tâm (Vi điều khiển)**:
  Vi điều khiển (như Arduino, ESP32 hoặc STM32) đóng vai trò nhận, xử lý dữ liệu và điều khiển toàn bộ hệ thống. Vi điều khiển được lập trình để giao tiếp với module Bluetooth, xử lý dữ liệu đầu vào và điều khiển các thiết bị ngoại vi như màn hình LCD hoặc bàn phím ma trận.
- **Khối nhập dữ liệu**:
  Bao gồm các thiết bị đầu vào như nút nhấn, bàn phím hoặc cảm biến, cho phép

người dùng hoặc hệ thống gửi dữ liệu đến vi điều khiển. Dữ liệu này có thể là các lệnh điều khiển, thông số cấu hình, hoặc tín hiệu từ các cảm biến.

- **Khối truyền dữ liệu (Module Bluetooth)**:
  Module Bluetooth (ESP32) đảm nhiệm việc truyền và nhận dữ liệu không dây giữa các thiết bị.BLE được tích hợp trong ESP 32 điều khiển trực tiếp nên không cần dùng đến giao thức UART.
- **Khối hiển thị (LCD hoặc LED)**:
  Màn hình LCD (Liquid Crystal Display) hoặc LED hiển thị được sử dụng để cung cấp thông tin trực quan về trạng thái kết nối Bluetooth, dữ liệu nhận được, và các thông báo hệ thống.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

Trong chương này, chúng ta sẽ đi sâu vào tìm hiểu các thành phần chính của hệ thống Bluetooth và các lý thuyết liên quan đến hoạt động của chúng. Đây là nền tảng quan trọng để hiểu rõ cách thức hoạt động và cách thiết kế hệ thống.

2.1 TỔNG QUAN VỀ CÔNG NGHỆ BLUETOOTH:

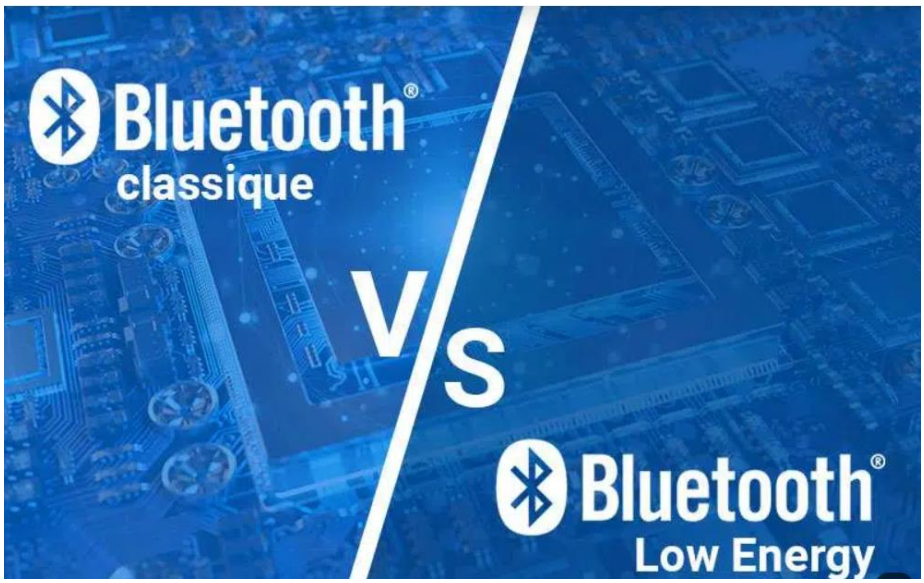
Kết nối máy tính cá nhân và thiết bị di động với các thiết bị ngoại vi như tai nghe không dây, bàn phím, chuột và bộ điều khiển chơi game. Các thiết bị Bluetooth giao tiếp thông qua các tín hiệu vô tuyến tầm ngắn trên dải tần 2,4 GHz.

Dải tần 2,4 GHz cung cấp nhiều kênh mà thiết bị Bluetooth có thể tận dụng để giao tiếp. Các thiết bị kết hợp nhảy giữa các kênh này trong chế độ khóa, liên tục tìm kiếm nhiễu nhất và chất lượng tín hiệu tốt nhất. Quá trình này, được gọi là hopping tần số, giúp các thiết bị Bluetooth mang lại độ trễ thấp, hiệu suất nhất quán.

Lần đầu tiên sử dụng thiết bị Bluetooth với thiết bị di động hoặc máy tính của bạn cần một quy trình cấu hình ban đầu được gọi là ghép nối, nơi mỗi thiết bị trao đổi các thông tin cần thiết như ID thiết bị và khóa bảo mật. Người dùng thường cần nhập mã PIN hoặc mã khóa trên một hoặc cả hai thiết bị để xác thực quá trình ghép nối. Sau quá trình ban đầu, thông tin kết nối quan trọng sẽ được lưu để hợp lý hóa việc kết nối trong tương lai.

Hiện tại, công nghệ Bluetooth được cung cấp trong hai phiên bản khác nhau, Bluetooth Classic và Bluetooth Năng lượng thấp (LE). Bluetooth Năng lượng thấp được tối ưu hóa để truyền thông liên tục giúp duy trì tuổi thọ pin, trong khi

Bluetooth Classic được sử dụng cho các ứng dụng yêu cầu truyền dữ liệu thường xuyên và liên tục hơn. Bluetooth Classic có tới 79 kênh 2,4 GHz tùy ý sử dụng, trong khi Bluetooth năng lượng thấp có đến 40 kênh để kết nối giữa các kênh.



2.2 CÁC KỸ THUẬT TRONG CÔNG NGHỆ BLUETOOTH

a) Frequency Hopping Spread Spectrum (FHSS)

- Tần số nhảy (Frequency Hopping): Bluetooth sử dụng kỹ thuật nhảy tần để truyền dữ liệu, giúp giảm nhiễu và tối ưu hóa việc chia sẻ phổ tần số 2.4GHz với các công nghệ khác như Wi-Fi và lò vi sóng. Bluetooth Classic nhảy qua 79 kênh, còn Bluetooth LE sử dụng 40 kênh.

- Lợi ích: Giảm thiểu nhiễu tín hiệu và cải thiện độ tin cậy của kết nối.

b) Bluetooth Classic (BR/EDR)

-Basic Rate (BR): Tốc độ truyền dữ liệu cơ bản với tốc độ 1 Mbps.
- Enhanced Data Rate (EDR): Tốc độ truyền dữ liệu cao hơn với tốc độ 2 hoặc 3 Mbps.

- Ứng dụng: Thường được sử dụng trong các ứng dụng truyền tải âm thanh không dây (loa, tai nghe, hệ thống giải trí trong ô tô).

c) Bluetooth Low Energy (LE)

- Tối ưu hóa năng lượng: Bluetooth LE được thiết kế để tiêu thụ năng lượng rất thấp, thích hợp cho các thiết bị cần hoạt động lâu dài với nguồn pin hạn chế như đồng hồ thông minh, thiết bị y tế, cảm biến IoT.
- Chế độ ngủ: Thiết bị có thể vào chế độ ngủ khi không truyền dữ liệu để tiết kiệm pin.
- Topologies: Bluetooth LE hỗ trợ nhiều cấu trúc liên kết giao tiếp như point-to-point, broadcast, và mesh network (mạng lưới), thích hợp cho các hệ thống điều khiển trong nhà, đèn thông minh, v.v.

d) Mesh Networking

- Mạng lưới (Mesh): Bluetooth LE Mesh cho phép nhiều thiết bị giao tiếp với nhau trong một mạng lưới lớn. Mỗi thiết bị đóng vai trò là một "nút" và có thể chuyển tiếp thông tin đến các thiết bị khác trong mạng, giúp tạo ra hệ thống mạng lớn và tin cậy.
- Ứng dụng: Nhà thông minh, hệ thống điều khiển đèn, hệ thống tự động trong tòa nhà.

e) Adaptive Frequency Hopping (AFH)

- Tần số nhảy thích ứng: Bluetooth có khả năng tự động điều chỉnh tần số để tránh nhiễu từ các thiết bị khác trong dải tần 2.4GHz. Kỹ thuật này giúp tăng độ tin cậy cho kết nối Bluetooth, đặc biệt khi có nhiều nguồn nhiễu trong môi trường.
- Lợi ích: Cải thiện hiệu suất và độ tin cậy trong môi trường tần số đông đúc.

f) Data Encryption and Authentication

- Mã hóa dữ liệu: Bluetooth sử dụng các phương thức mã hóa để bảo vệ thông tin khỏi sự truy cập trái phép. AES-128 là một thuật toán mã hóa thường được sử dụng trong Bluetooth LE.
- Xác thực: Bluetooth cung cấp các phương thức xác thực nhằm đảm bảo rằng các thiết bị được kết nối là hợp lệ và tránh các cuộc tấn công trung gian (Man-in-the-Middle).

g) Bluetooth Dual Mode

- Chế độ kép: Nhiều thiết bị hiện nay hỗ trợ cả Bluetooth Classic và Bluetooth LE, cho phép chúng truyền tải dữ liệu ở tốc độ cao với Bluetooth Classic và tiết kiệm năng lượng với Bluetooth LE khi cần thiết.
- Ứng dụng: Điện thoại thông minh, máy tính bảng, các thiết bị thông minh.

h) Audio Over Bluetooth (A2DP)

- A2DP (Advanced Audio Distribution Profile): Là giao thức truyền tải âm thanh qua Bluetooth, cho phép truyền phát âm thanh stereo từ thiết bị nguồn (như điện thoại di động) đến các thiết bị nhận (như tai nghe hoặc loa).
- Ứng dụng: Truyền tải âm thanh không dây trong các hệ thống âm thanh di động và gia đình.

i) Bluetooth Audio Codec (SBC, aptX, LDAC)

- SBC (Subband Codec): Codec âm thanh tiêu chuẩn trong Bluetooth.
- aptX, aptX HD, LDAC: Là các codec tiên tiến cho phép truyền tải âm thanh chất lượng cao qua Bluetooth, giúp giảm độ trễ và tăng cường chất lượng âm thanh.

j) Bluetooth Direction Finding

- Định vị hướng: Bluetooth LE hỗ trợ các tính năng định vị giúp xác định hướng của một thiết bị Bluetooth khác, phù hợp cho các ứng dụng theo dõi vị trí và tìm kiếm thiết bị.
- Ứng dụng: Theo dõi thiết bị, dẫn đường trong các không gian nội thất như tòa nhà lớn.

k) Bluetooth Beacons

- Beacon: Bluetooth Beacons là các thiết bị nhỏ sử dụng Bluetooth LE để phát sóng tín hiệu một cách định kỳ. Các thiết bị di động trong phạm vi có thể nhận diện và phản hồi lại các beacon.
- Ứng dụng: Marketing tại chỗ, hướng dẫn trong tòa nhà, định vị trong siêu thị.

2.3 CÁC CHUẨN TRONG CÔNG NGHỆ BLUETOOTH:

- Bluetooth 1.0: Đây là phiên bản Bluetooth đầu tiên với tốc độ 1Mpbs, nhưng độ tương thích chưa cao.

-Bluetooth 1.1: Phiên bản này sửa được những lỗi cơ bản được phát hiện ở dòng Bluetooth đời đầu. Tuy nhiên, tốc độ kết nối không được cải thiện.

-Bluetooth 1.2: Dòng Bluetooth này phát triển phụ thuộc vào băng tần 2.4Ghz, có tốc độ kết nối và truyền tải dữ liệu nhanh hơn so với phiên bản Bluetooth trước đó.

-Bluetooth 2.0 + ERD: Được công bố vào năm 2007, dòng Bluetooth mới này có nhiều cải thiện mới, tiêu biểu là kết nối ổn định và tốc độ truyền tải dữ liệu qua thiết bị khác nhanh hơn.

-Bluetooth 2.1 + ERD: Chứa toàn bộ ưu điểm của phiên bản 2.0, đồng thời có hiệu năng cao hơn và năng lượng tiêu hao ít hơn. Đồng thời, tích hợp thêm cơ chế kết nối phạm vi nhỏ.

-Bluetooth 3.0 + HS: Được ra mắt vào tháng 04 năm 2009. Và có tốc độ kết nối cao tương đương chuẩn WiFi thế hệ đầu tiên đạt mức 24Mbps, phụ thuộc vào công nghệ High Speed. Tuy vậy, nhưng tầm phủ sóng chỉ vẫn còn khá hẹp (10 mét).

-Bluetooth 4.0: Ra mắt tháng 6 năm 2010, phiên bản 4.0 là sự kết hợp của phiên bản Bluetooth 2.1 và 3.0. Ưu điểm nổi bật là tốc độ truyền tải dữ liệu nhanh hơn đồng thời giúp tiết kiệm năng lượng hiệu quả.

-Bluetooth 4.1: Đây là bản nâng cấp của Bluetooth 4.0 vào năm 2014, nhằm giúp cải thiện tình trạng nhiễu loạn dữ liệu với mạng 4G. Làm cho hiệu suất kết nối, truyền tải giữa các thiết bị được tối ưu hơn.

-Bluetooth 4.2: Cùng được công bố vào năm 2014 với Bluetooth 4.1 nhưng phiên bản 4.2 có tốc độ được cải thiện nhanh gấp 2.5 lần. Đồng thời có thể giảm năng lượng tiêu thụ, khắc phục lỗi kết nối, tăng tính bảo mật và hỗ trợ chia sẻ kết nối mạng internet nhờ vào giao thức IPv6.

-Bluetooth 5.0: Đây là phiên bản Bluetooth mới nhất được SIG trình làng vào 2016. Phiên bản Bluetooth mới này có nhiều cải thiện vượt bậc về tầm phủ

sóng (rộng gấp 4 lần), tốc độ kết nối nhanh và khả năng tiết kiệm năng lượng hơn so với bản 4.0.

-Bluetooth 5.1: Nâng cấp thêm về khả năng dò hướng AoA (Angle of Arrival) và AoD (Angle of Departure) để xác định vị trí của thiết bị kết nối chính xác hơn. Ngoài ra phiên bản 5.1 còn có khả năng kết nối không cần gói dữ liệu hỗ trợ đồng bộ đơn giản và tiết kiệm điện nhiều gấp 2.5 lần so với 4.0.

-Bluetooth 5.2: Được SIG trình làng vào năm 2020, với chức năng là giao thức thuộc tính nâng cao (EATT) làm giảm độ trễ và tăng độ mã hóa trong kết nối, tính năng kiểm soát LEPC giúp kiểm soát nguồn và ổn định chất lượng tín hiệu, giảm tỉ lệ nhiễu tín hiệu và tính năng ISOC có thể truyền dữ liệu hai chiều với nhiều thiết bị cùng lúc.

2.4 CÁC MODULETRONG CÔNG NGHỆ BLUETOOTH:

-Bluetooth Radio Module (Radio Layer)

Chức năng: Đây là lớp vật lý chịu trách nhiệm truyền và nhận tín hiệu radio. Bluetooth hoạt động trên dải tần ISM 2.4 GHz (công nghiệp, khoa học, và y tế) và sử dụng kỹ thuật nhảy tần số (FHSS) để giảm nhiễu.

- Baseband and Link Control Module

Chức năng: Quản lý việc truy cập vào kênh vật lý, điều khiển liên kết, đồng bộ dữ liệu giữa các thiết bị. Module này đảm bảo rằng các thiết bị Bluetooth có thể giao tiếp với nhau một cách hiệu quả.

-L2CAP (Logical Link Control and Adaptation Protocol)

Chức năng: L2CAP là một giao thức quản lý gói dữ liệu, phân chia và ghép nối dữ liệu để đảm bảo rằng thông tin có thể được truyền chính xác và theo trình tự qua Bluetooth.

-HCI (Host Controller Interface)

Chức năng: Giao diện giữa bộ điều khiển Bluetooth và hệ thống xử lý (thường là CPU của thiết bị. HCI là cầu nối giữa phần cứng Bluetooth và phần mềm chạy trên thiết bị.

- SDP (Service Discovery Protocol)

Chức năng: SDP cho phép các thiết bị Bluetooth khám phá dịch vụ mà các thiết bị khác hỗ trợ. Đây là bước đầu tiên trong quá trình thiết lập một kết nối giữa các thiết bị Bluetooth.

-RFCOMM (Radio Frequency Communication)

Chức năng: RFCOMM là giao thức mô phỏng các cổng nối tiếp (serial port) trên Bluetooth, cho phép các thiết bị truyền dữ liệu song song như cổng COM trên máy tính.

-ATT (Attribute Protocol)

Chức năng: Đây là giao thức được sử dụng trong Bluetooth LE để trao đổi dữ liệu giữa máy chủ và máy khách dưới dạng thuộc tính.

-GATT (Generic Attribute Profile)

Chức năng: GATT định nghĩa cách các thiết bị truyền và nhận các gói dữ liệu ngắn trong Bluetooth LE, dựa trên ATT.

-A2DP (Advanced Audio Distribution Profile)

Chức năng: A2DP là một profile Bluetooth dành cho việc truyền tải âm thanh stereo từ thiết bị phát (như điện thoại) đến thiết bị nhận (như tai nghe hoặc loa).

- AVRCP (Audio/Video Remote Control Profile)

Chức năng: AVRCP cung cấp giao thức để điều khiển các thiết bị audio/video từ xa qua Bluetooth.

-HID (Human Interface Device Profile)

Chức năng: HID là profile Bluetooth dành cho việc kết nối các thiết bị giao diện người dùng, như bàn phím, chuột, và gamepad.

-BLE Mesh Profile

Chức năng: BLE Mesh cho phép các thiết bị Bluetooth LE kết nối và giao tiếp trong một mạng lưới (mesh) lớn.

2.5 CÁC ỨNG DỤNG CỦA BLUETOOTH:

-Cho phép các thiết bị kết nối được với nhau để trao đổi thông tin giữa chúng. Ví dụ như điện thoai di động với nhau, máy tính bảng, máy tính xách tay, máy in,…

-Giao tiếp và điều khiển giữa một thiết bị di dộng với tai nghe không dây như tai nghe bluetooth.

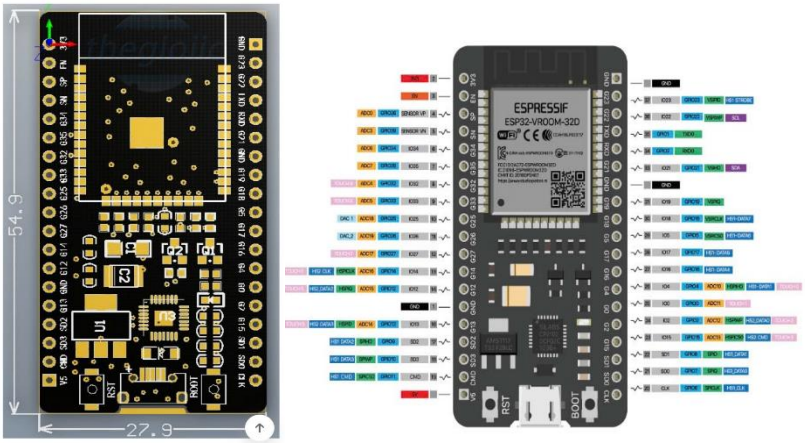
-Trở thành đường truyền kết nối không dây giữa các thiết bị vào – ra của máy tính như: chuột, bàn phím không dây.

-Ứng dụng trong các thiết bị điều khiển từ xa như bộ điều khiển đồ chơi, điều khiển tivi, điều hòa,…

-Kết nối internet cho máy tính bằng cách biến smartphone làm modem.

2.6 SƠ LƯỢC PHẦN CỨNG HỆ THỐNG BLUETOOTH:

a. ESP32 WROOM (38 chân):



-Các ngoại vi của ESP32 bao gồm:

- 18 kênh Bộ chuyển đổi Tín hiệu Tương tự sang Số (ADC)
- 3 cổng giao tiếp SPI
- 3 cổng giao tiếp UART
- 2 cổng giao tiếp I2C
- 16 kênh đầu ra PWM
- 2 Bộ chuyển đổi Tín hiệu Số sang Tương tự (DAC)
- 2 cổng giao tiếp I2S
- 10 chân GPIO cảm ứng điện dung

Chức năng các chân:

Chân	Tín hiệu	Chức năng
GND	GND	Nối đất, thường được sử dụng để hoàn thiện mạch
G23	GPIO23	GPIO, có thể sử dụng cho giao tiếp PWM
G22	GPIO22	GPIO, có thể sử dụng cho giao tiếp PWM
G21	GPIO21	GPIO, có thể sử dụng cho giao tiếp PWM
G19	GPIO19	GPIO, thường được sử dụng cho giao tiếp SPI
G18	GPIO18	GPIO, thường được sử dụng cho giao tiếp SPI
G17	GPIO17	GPIO, có thể sử dụng cho giao tiếp PWM
G16	GPIO16	GPIO, có thể sử dụng cho giao tiếp PWM
G15	GPIO15	GPIO, có thể sử dụng cho giao tiếp PWM
G14	GPIO14	GPIO, thường được sử dụng cho giao tiếp SPI
G13	GPIO13	GPIO, có thể sử dụng cho giao tiếp PWM
G12	GPIO12	GPIO, thường được sử dụng để đầu vào cho đột lập trình
G2	GPIO2	GPIO, thường được sử dụng để điều khiển LED tích hợp trên board
G1	GPIO1	TXD0, chân truyền dữ liệu UART 0
G3	GPIO3	RXD0, chân nhận dữ liệu UART 0
G5	GPIO5	GPIO, có thể sử dụng cho giao tiếp PWM
G16	GPIO16	GPIO, có thể sử dụng cho giao tiếp PWM
G8	GPIO8	GPIO, có thể sử dụng cho giao tiếp PWM
G7	GPIO7	GPIO, có thể sử dụng cho giao tiếp PWM
G6	GPIO6	GPIO, có thể sử dụng cho giao tiếp PWM

G10	GPIO10	GPIO, có thể sử dụng cho giao tiếp PWM
G11	GPIO11	GPIO, có thể sử dụng cho giao tiếp PWM
G9	GPIO9	GPIO, có thể sử dụng cho giao tiếp PWM
3V3	VCC	Cung cấp nguồn 3.3V cho board
VIN	VCC	Cung cấp nguồn cho board (thường là 5V)
EN	EN	Nút Reset, dùng để khởi động lại board
BOOT	BOOT	Nút Boot, dùng để vào chế độ lập trình

b. MÀN HÌNH LCD:

- Màn hình tinh thể lỏng LCD (Liquid Crystal Display) được cấu tạo nên bởi các tế bào (các điểm ảnh) chứa tinh thể lỏng với khả năng thay đổi tính phân cực của ánh sáng và thay đổi cường độ ánh sáng truyền qua khi kết hợp với các loại kính lọc phân cực.

- Nói một cách dễ hiểu hơn thì LCD chính là công nghệ dùng đèn nền để tạo ánh sáng chứ không tự phát sáng được.

- Cấu tạo của màn hình LCD gồm 6 lớp xếp chồng lên nhau:

(1) Kính lọc phân cực thẳng đứng có tác dụng lọc ánh sáng tự nhiên khi vào

(2) Lớp kính có điện cực ITO

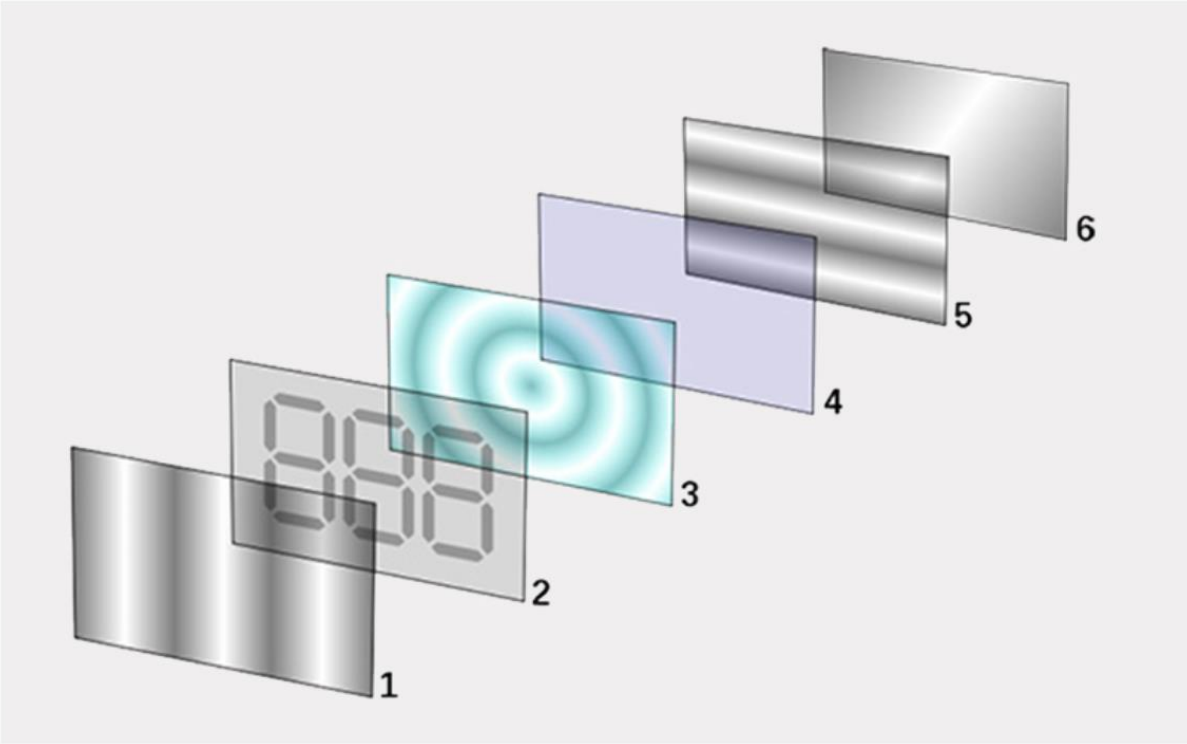
(3) Lớp tinh thể lỏng

(4) Lớp kính có điện cực ITO chung

(5) Kính lọc phân cực nằm ngang

(6) Gương phản xạ, tác dụng phản xạ lại ánh sáng với người quan sát

-Vốn dĩ màn hình LCD hiển thị màu sắc nhờ vào những điểm ảnh chứa tinh thể lỏng có thể thay đổi màu sắc và cường độ ánh sáng.



Màn hình LCD gồm 6 lớp xếp chồng lên nhau

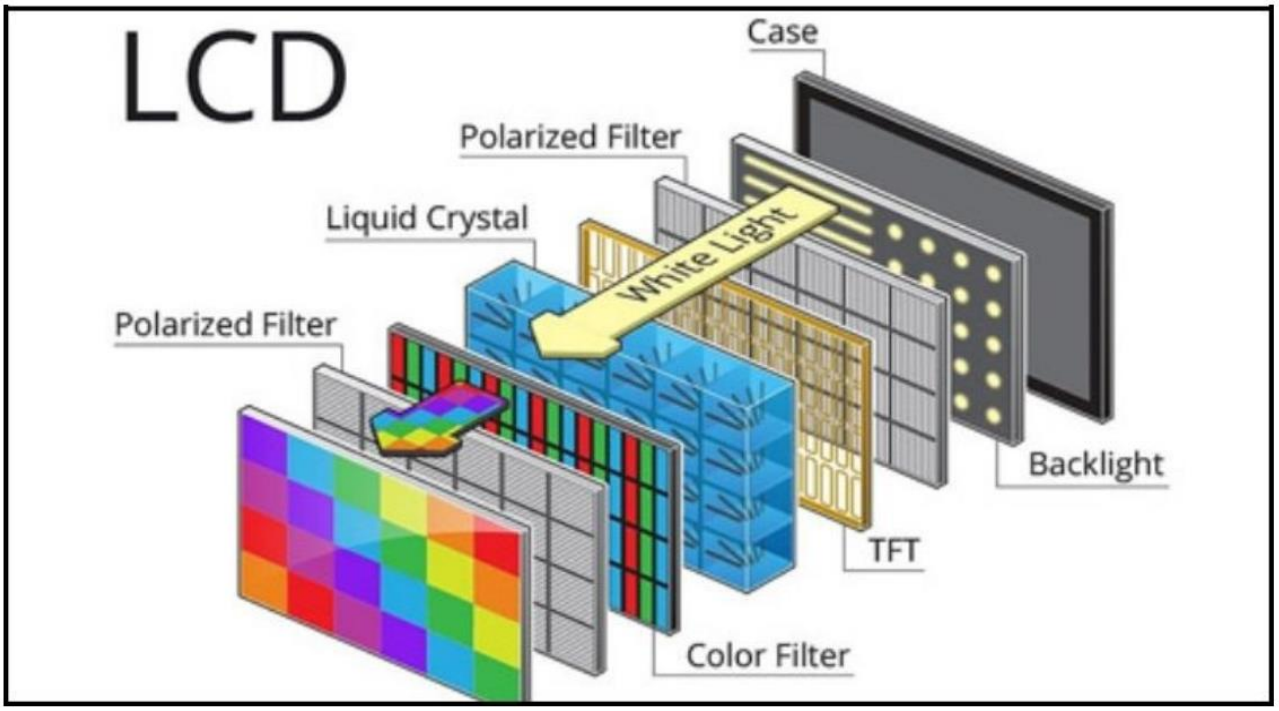
- **Nguyên lý hoạt động của LCD:** LCD (Liquid Crystal Display) là loại màn hình sử dụng các tinh thể lỏng để hiển thị hình ảnh. Màn hình LCD bao gồm các thành phần chính:

-Tấm kính phân cực: Điều chỉnh ánh sáng đi qua.

-Tinh thể lỏng: Thay đổi hướng ánh sáng khi có điện áp tác động.

-Đèn nền (Backlight): Cung cấp ánh sáng cho màn hình.

-Điện cực điều khiển: Tạo ra điện áp tác động lên tinh thể lỏng để điều chỉnh ánh sáng.



Cách thức hoạt động của màn hình LCD

c. BÀN PHÍM HEX:

-Cấu tạo của bàn phím Hex: Bàn phím ma trận Hex là một loại bàn phím được cấu tạo từ các nút nhấn được tổ chức dưới dạng ma trận hàng và cột. Đây là một thiết bị đầu vào phổ biến trong các ứng dụng vi điều khiển.

-Cấu trúc cơ bản:
Bàn phím Hex bao gồm:

+Các hàng và cột: Các nút nhấn được kết nối giữa các hàng và cột của mạch, tạo nên cấu trúc ma trận.

+Công tắc nút nhấn (Switches): Mỗi nút nhấn trong bàn phím là một công tắc, đóng mạch khi được nhấn.

+Dây dẫn: Kết nối các hàng và cột với vi điều khiển để truyền tín hiệu.

+Sơ đồ cấu tạo của bàn phím Hex (4x4):
Một bàn phím Hex phổ biến thường có cấu trúc 4 hàng và 4 cột (4x4), cung cấp

tổng cộng 16 phím. Các hàng và cột được kết nối với vi điều khiển thông qua các chân GPIO.

-Cách thức hoạt động:

+Vi điều khiển gửi tín hiệu quét qua các cột (hoặc hàng) và kiểm tra tín hiệu phản hồi từ các hàng (hoặc cột).

+Khi một nút nhấn được nhấn, nó tạo ra một kết nối giữa một hàng và một cột, cho phép dòng điện chạy qua.

+Vi điều khiển phát hiện vị trí nút nhấn dựa trên hàng và cột tương ứng.

-Quy trình quét ma trận:

+Vi điều khiển kích hoạt lần lượt từng cột (hoặc hàng) bằng cách đặt tín hiệu logic thấp (LOW).

+Các hàng còn lại được đọc tín hiệu. Nếu một hàng nhận được tín hiệu, điều đó cho biết nút nhấn tại giao điểm của hàng và cột đó đã được nhấn.

+Quá trình lặp lại cho đến khi tất cả các cột được quét.

CHƯƠNG 3:THIẾT KẾ VÀ XÂY DỰNG HỆ THỐNG

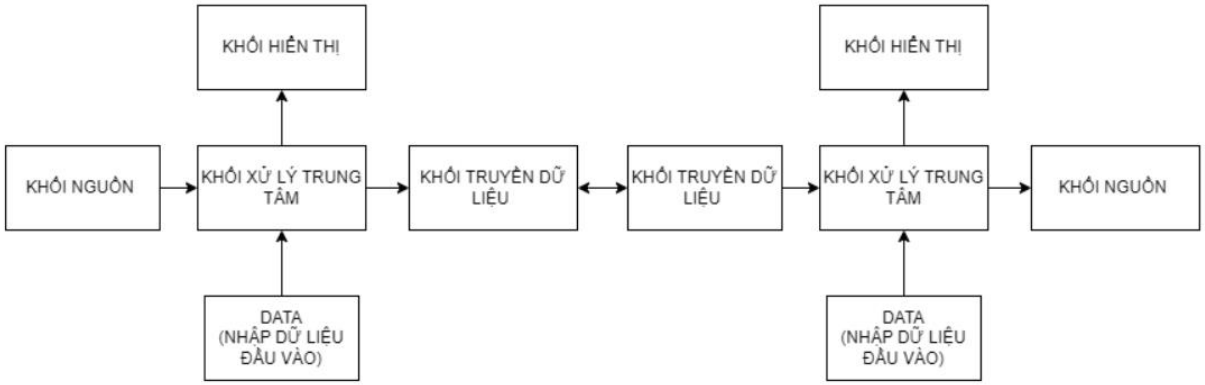
3.1 YÊU CẦU VÀ SƠ ĐỒ KHỐI HỆ THỐNG

a. Yêu cầu của hệ thống

Hệ thống có các chức năng sau:

- Truyền và nhận dữ liệu không dây: Đảm bảo kết nối không dây ổn định để truyền dữ liệu giữa các thiết bị Bluetooth.
- Kết nối và điều khiển thiết bị: Ghép nối nhanh chóng và điều khiển các thiết bị LCD, và nút nhấn qua Bluetooth.
- Hiển thị trạng thái hệ thống: Hiển thị thông tin kết nối, dữ liệu nhận được, hoặc trạng thái hoạt động của hệ thống trên màn hình LCD hoặc ứng dụng.
- Cấu hình thông số: Cho phép người dùng truyền dữ liệu số Hex từ hệ thống này sang hệ thống khác.
- Phát hiện và xử lý sự cố kết nối: Cảnh báo khi mất kết nối hoặc khi hệ thống gặp lỗi.

b. Sơ đồ khối và chức năng mỗi khối



Sơ đồ khối của hệ thống

Chức năng từng khối:

- **Khối nguồn**:Cung cấp năng lượng ổn định cho toàn bộ hệ thống, bao gồm vi điều khiển, cảm biến, và các module Bluetooth.

- **Khối xử lý trung tâm:**

+Thu nhận dữ liệu từ các cảm biến hoặc thiết bị đầu vào.

+Xử lý và phân tích dữ liệu nhận được để đưa ra lệnh điều khiển phù hợp.

+Quản lý và điều phối các tín hiệu truyền nhận giữa các khối.

- **Khối nhập dữ liệu (Data):**Thu thập dữ liệu đầu vào từ các cảm biến hoặc thiết bị điều khiển (như nút nhấn, tín hiệu từ các cảm biến, hoặc thiết bị ngoại vi).

- **Khối truyền dữ liệu:**

+Đảm nhiệm việc truyền dữ liệu không dây giữa các thiết bị qua kết nối Bluetooth.

+Đảm bảo dữ liệu được truyền đi và nhận lại chính xác, không bị mất gói.

- **Khối hiển thị:**

+Hiển thị thông tin hệ thống, như trạng thái kết nối, dữ liệu nhận được, hoặc các thông số cấu hình.

+Cung cấp giao diện trực quan để người dùng dễ dàng theo dõi hoạt động của hệ thống.

c. Hoạt động hệ thống

Khi hệ thống được cấp nguồn hệ thống sẽ hoạt động theo trình tự như sau:

- Khởi động hệ thống (Khối nguồn):

+Hệ thống bắt đầu hoạt động khi khối nguồn cung cấp năng lượng ổn định cho các khối trong hệ thống.

+Các thiết bị như vi điều khiển, cảm biến, module Bluetooth và màn hình hiển thị được cấp nguồn và sẵn sàng hoạt động.

- Thu thập dữ liệu (Khối nhập dữ liệu):

+_Dữ liệu đầu vào được thu thập từ các cảm biến hoặc thiết bị điều khiển, như tín hiệu từ nút nhấn, giá trị cảm biến nhiệt độ, độ ẩm, hoặc các tín hiệu tương tự.

+Dữ liệu này được gửi đến khối xử lý trung tâm để xử lý.

- Xử lý dữ liệu (Khối xử lý trung tâm):

+Vi điều khiển trong khối xử lý trung tâm tiếp nhận dữ liệu từ khối nhập dữ liệu.

+Dữ liệu được phân tích, xử lý để đưa ra quyết định điều khiển hoặc truyền thông tin đến các thiết bị khác.

+Nếu cần truyền dữ liệu đến thiết bị khác qua Bluetooth, khối xử lý trung tâm gửi dữ liệu đến khối truyền dữ liệu.

-Truyền dữ liệu (Khối truyền dữ liệu):

+Dữ liệu đã xử lý được truyền qua khối truyền dữ liệu, sử dụng module Bluetooth để gửi thông tin đến thiết bị đầu cuối khác.

+Đồng thời, khối truyền dữ liệu cũng nhận thông tin từ các thiết bị khác gửi về và chuyển lại cho khối xử lý trung tâm để xử lý.

- Hiển thị thông tin (Khối hiển thị):

+Trạng thái của hệ thống, dữ liệu thu thập được, hoặc kết quả xử lý sẽ được hiển thị trên màn hình LCD hoặc thiết bị đầu cuối.

+Người dùng có thể theo dõi trạng thái hệ thống, kiểm tra kết nối Bluetooth, hoặc quan sát các giá trị dữ liệu theo thời gian thực.

- Phản hồi và điều khiển:

+Dựa trên dữ liệu nhận được hoặc tín hiệu từ người dùng, hệ thống có thể gửi lệnh điều khiển đến các thiết bị đầu cuối khác (như bật/tắt đèn, điều khiển động cơ).

+Hệ thống liên tục kiểm tra trạng thái kết nối Bluetooth để đảm bảo hoạt động ổn định và phản hồi kịp thời khi có sự cố.

- Lặp lại quy trình:Hệ thống hoạt động theo chu trình liên tục, từ thu thập dữ liệu, xử lý, truyền nhận và hiển thị, đảm bảo hệ thống Bluetooth vận hành hiệu quả và ổn định.

3.2 THIẾT KẾ MẠCH ĐIỆN

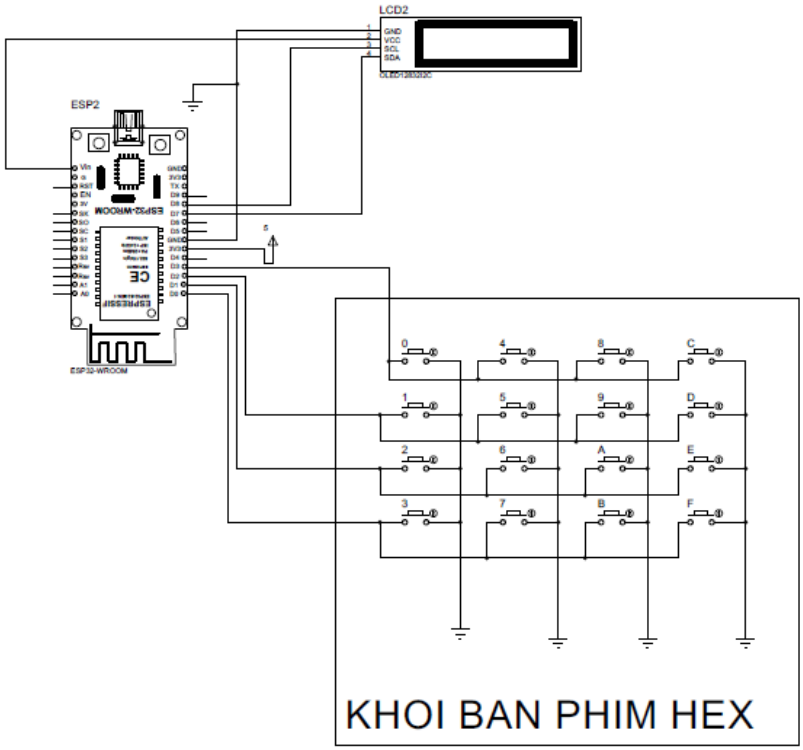
a. Sơ đồ mạch điện:

Sơ đồ mạch điện là bản vẽ chi tiết các kết nối giữa các thành phần của hệ thống. Sơ đồ mạch điện của hệ thống Bluetooth bao gồm các thành phần chính sau:

-Các nút nhấn số HEX: Dùng để nhập dữ liệu.

-Vi điều khiển (ESP 32): Trung tâm điều khiển hệ thống.

-Màn hình LCD: Hiển thị dữ liệu đã nhập.



Sơ đồ mạch của một khối thu và phát bluetooth

b. Các kết nối của sơ đồ mạch điện:

-Các nút nhấn số HEX: được tạo từ các nút nhấn đơn chạy từ 1 đến F theo số HEX được kết nối với các chân D0 tới D3 của ESP 32 .

-LCD: Kết nối các chân của LCD (RS, EN, D4, D5, D6, D7) với các chân số của ESP 32 . Kết nối nguồn và GND của LCD với nguồn 5V và GND của ESP 32.

-ESP 32: Vi điều khiển chính của hệ thống Bluetooth.

3.3 THIẾT KẾ HỆ THỐNG PHẦN CỨNG

Thiết kế phần cứng của hệ thống bluetooth là một bước quan trọng trong việc đảm bảo hệ thống hoạt động hiệu quả và ổn định. Phần này sẽ trình bày chi tiết về các thành phần phần cứng, cách kết nối và nguyên lý hoạt động của từng khối trong hệ thống.

a. Tổng Quan Về Thiết Kế Hệ Thống Phần Cứng

Hệ thống phần cứng bao gồm các thành phần chính sau:

-Vi điều khiển ESP 32: Trung tâm điều khiển hệ thống.

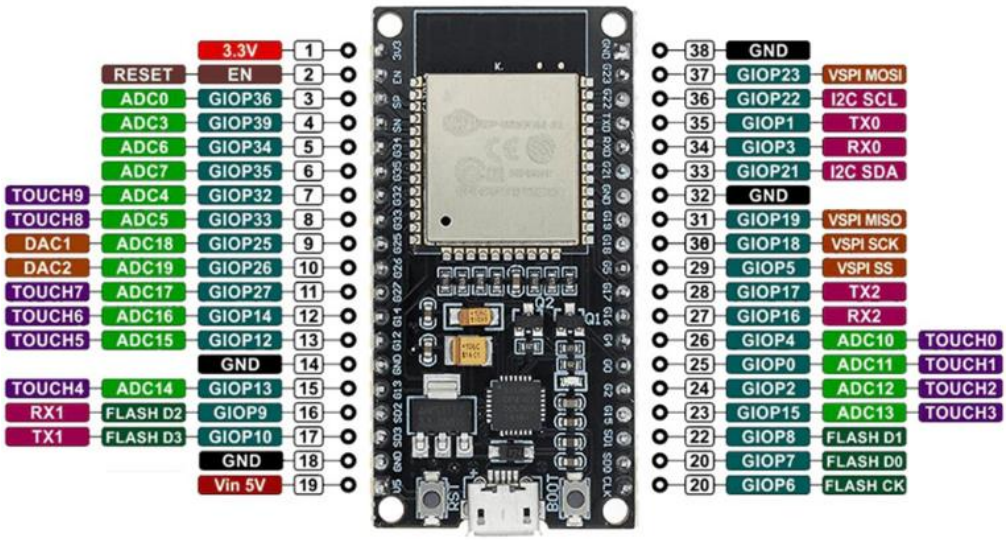
-Màn hình LCD: Hiển thị thông tin hệ thống.

-Các nút nhấn phím HEX: Nhập dữ liệu cần truyền.

b. Vi Điều Khiển ESP 32

ESP 32 là một vi điều khiển phổ biến và dễ sử dụng, nó có tích hợp truyền Bluetooth trong vi điều khiển không cần phải thêm module bluetooth nên em chọn làm trung tâm điều khiển của hệ thống. Các thông số kỹ thuật chính sau:

- Điện áp hoạt động: 5V
- Số chân digital I/O: 34
- Số chân analog: 18 (ADC), 2 (DAC)
- Tần số xung nhịp: 160 MHz hoặc 240 MHz



Sơ đồ chân ESP 32

c.Màn Hình LCD

Màn hình LCD được sử dụng để hiển thị các thông số như cường độ ánh sáng, trạng thái đèn và trạng thái màn che. LCD 16x2 được kết nối với ESP32 qua giao thức I2C để tiết kiệm chân kết nối.

kết nối màn hình LCD với Arduino:

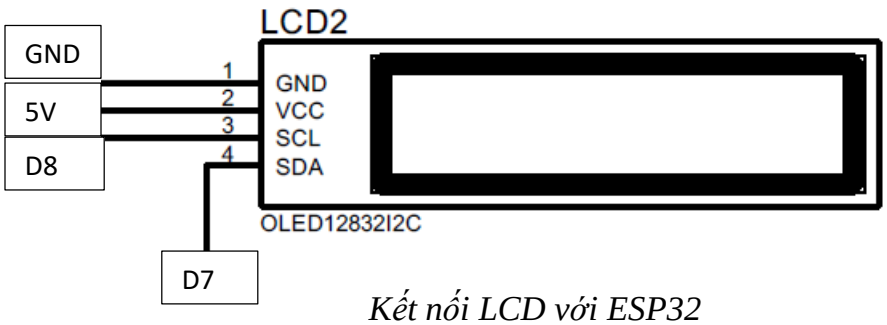
-Kết nối chân VCC của LCD với nguồn 5V.

- Kết nối chân GND của LCD với chân GND của ESP32.

- Kết nối chân SCL của LCD với chân D8 của ESP32.

- Kết nối chân SDA của LCD với chân D7 cảu ESP32.

Sơ đồ kết nối:



Kết nối LCD với ESP32

d. Các nút nhấn số HEX

Nút nhấn được sử dụng để nhập dữ liệu muốn truyền đạt , Em dùng bàn phím số HEX để thực hiện chức năn cho hệ thống.

Cách kết nối nút nhấn với ESP 32:

-Các nút 0,4,8,C nối với chân D3 của ESP32.

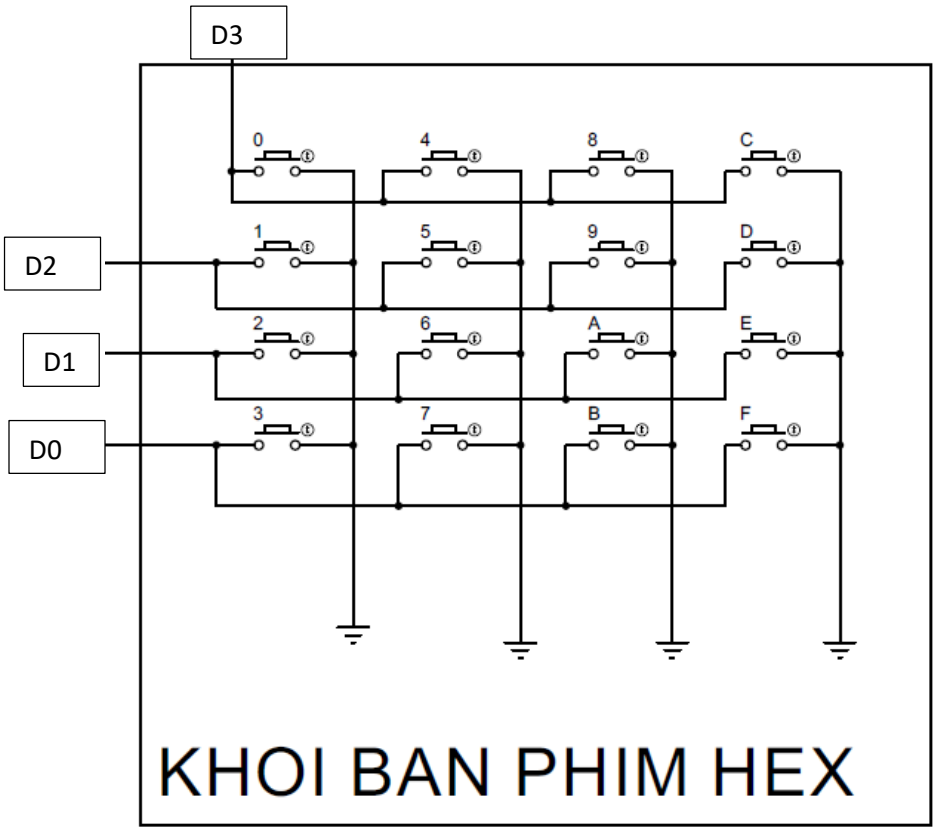
- Các nút 1,5,9,D nối với chân D2 của ESP32.

- Các nút 2,6,A,E nối với chân D1 của ESP32.

-Các nút 3,7,B,F nối với chân D0 của ESP32.

- Chân còn lại của nút nhấn nối với GND

Sơ đồ kết nối:

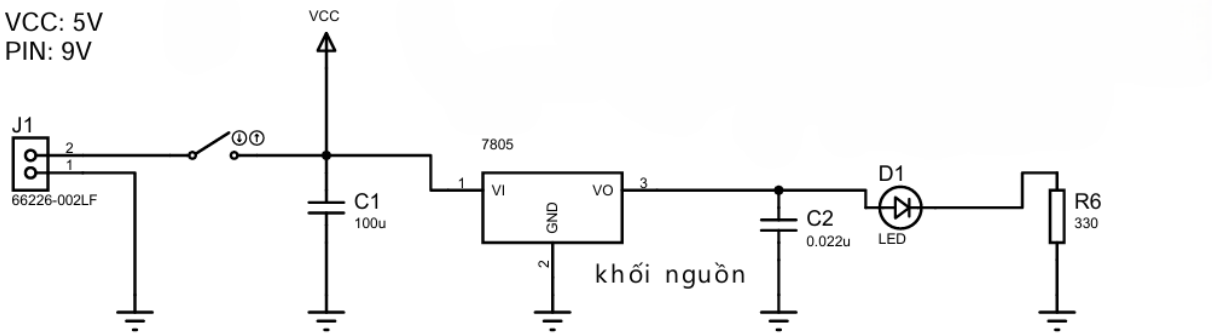


Kết nối nút nhấn với ESP32

e. Khối nguồn

Nguồn điện cung cấp năng lượng cho toàn bộ hệ thống. Em sử dụng nguồn 5V DC để đảm bảo an toàn và phù hợp với các thành phần của hệ thống.

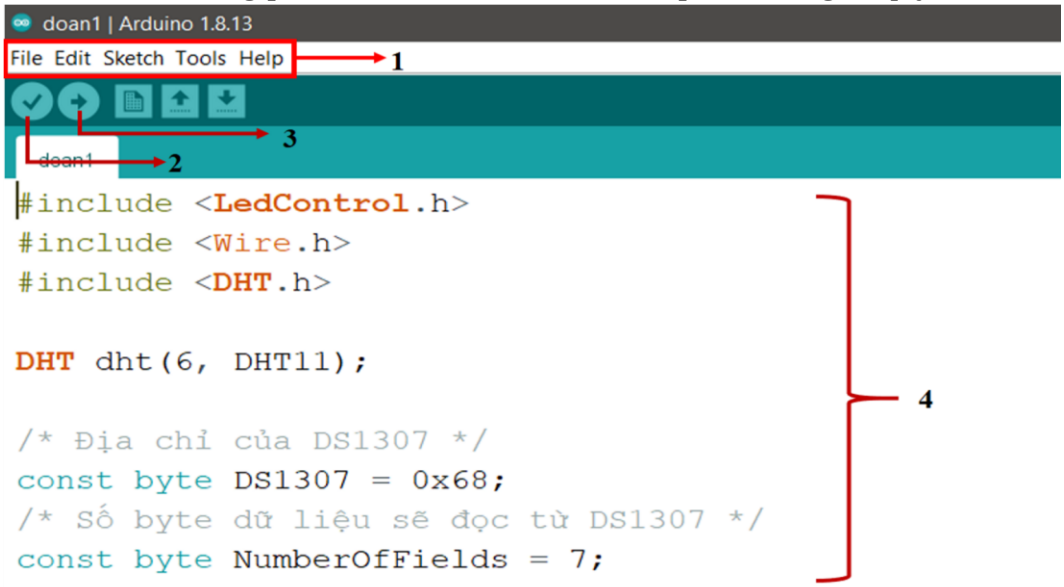
Sơ đồ kết nối :



Sơ đồ khối nguồn

3.4 CHỨC NĂNG VÀ HOẠT ĐỘNG CỦA PHẦN MỀM

Trong đề tài em sử dụng phần mềm Arduino IDE để lập trình và giải quyết các vấn đề.



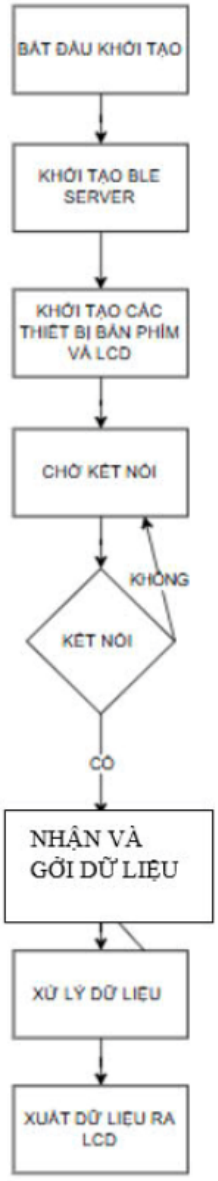
Giao diện phần mềm Arduino IDE

Giao diện của phần mềm Arduino IDE có nhiều phần, tuy nhiên chúng ta chú ý đến những phần quan trọng như được nêu ra trong hình trên. Chức năng của từng phần như sau:

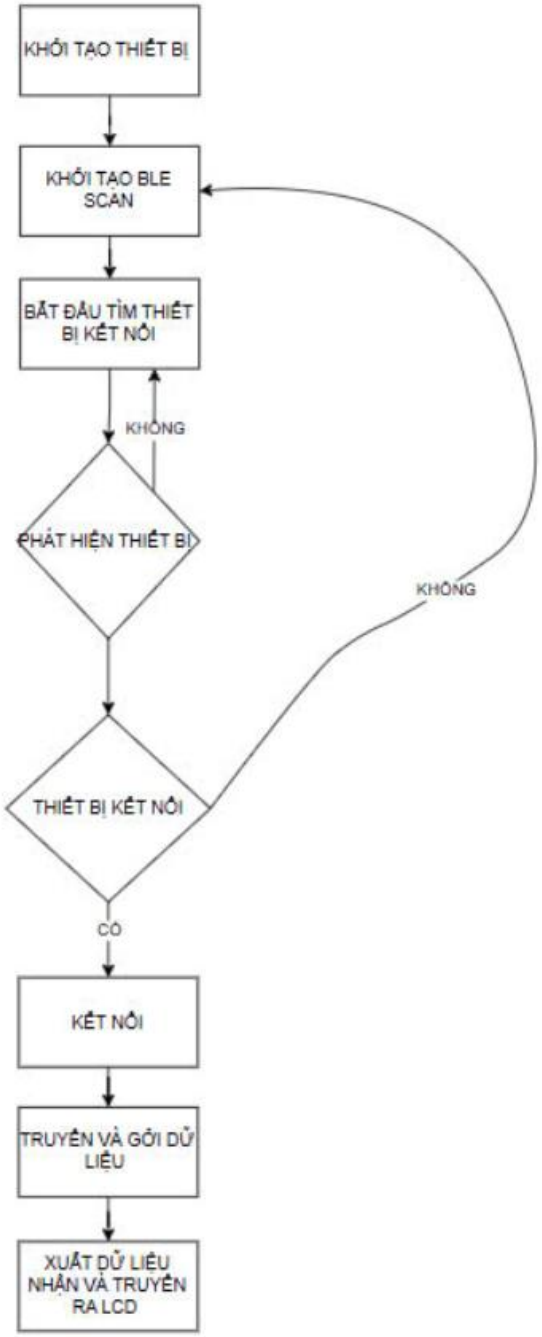
- 1: Menu lệnh: Dùng để thêm thư viện, lưu, tạo project mới,….
- 2: Nút kiểm tra chương trình (built): Dùng để kiểm tra xem chương trình được viết có lỗi không. Nếu chương trình bị lỗi thì phần mềm Arduino IDE sẽ hiển thị thông tin lỗi ở vùng thông báo thông tin.
- 3: Nút nạp chương trình xuống board Arduino: Dùng để nạp chương trình được viết xuống mạch Arduino. Trong quá trình nạp, chương trình sẽ được kiểm tra lỗi trước sau đó mới thực hiện nạp xuống mạch Arduino.
- 4: Vùng lập trình: Vùng này để người lập trình thực hiện việc lập trình cho chương trình của mình.

3.5 LƯU ĐỒ

Lưu Đồ Hoạt Động Khối Xử Lý Trung Tâm



Lưu đồ hoạt động khối xử lý trung tâm esp-esp



Lưu đồ hoạt động khối xử lý trung tâm esp-mobile

Mô tả chi tiết lưu đồ

- **Khởi tạo thiết bị**:Bắt đầu quá trình, hệ thống tiến hành khởi tạo thiết bị cần thiết để hoạt động.

- **Khởi tạo BLE Scan**:Kích hoạt chức năng quét BLE (Bluetooth Low Energy) để tìm kiếm các thiết bị xung quanh.

- **Bắt đầu tìm thiết bị kết nối**:Hệ thống bắt đầu quét để tìm kiếm các thiết bị có thể kết nối qua BLE.

- **Phát hiện thiết bị**:

  +Nếu phát hiện được thiết bị, chuyển sang bước kiểm tra kết nối.

  +Nếu không phát hiện được thiết bị, quay lại bước quét để tiếp tục tìm kiếm.

- **Thiết bị kết nối**:

  +Nếu thiết bị được phát hiện sẵn sàng kết nối, tiến hành kết nối.

  +Nếu không, quay lại bước quét để tìm kiếm thiết bị khác.

- **Kết nối**:Thực hiện kết nối giữa hệ thống và thiết bị BLE.

- **Nhận và gửi dữ liệu**:Sau khi kết nối, hệ thống thực hiện trao đổi dữ liệu với thiết bị BLE.

- **Xuất dữ liệu nhận và truyền ra LCD**:Dữ liệu nhận được từ thiết bị BLE được xử lý và hiển thị lên màn hình LCD.

3.6. CODE CHÍNH CHẠY CHƯƠNG TRÌNH:

Trong hệ thống này, em đã xây dựng ba chương trình phục vụ cho ba mục đích khác nhau:

+Một chương trình truyền dữ liệu không liên tục giữa hai thiết bị ESP32.

+Một chương trình truyền dữ liệu liên tục giữa hai thiết bị ESP32.

+Một chương trình truyền dữ liệu giữa thiết bị ESP32 và các thiết bị điện thoại di động.

Các chương trình này được thiết kế để đảm bảo tính linh hoạt và đáp ứng các yêu cầu giao tiếp dữ liệu trong các tình huống khác nhau

- Chương trình truyền dữ liệu không liên tục giữa hai thiết bị ESP32.

+Scanner:

```
// THƯ VIỆN
#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEScan.h>
#include <BLEAdvertisedDevice.h>
#include <BLEClient.h>
#include <Keypad.h>
#include <LiquidCrystal_I2C.h>  // thư viện LCD

// BIẾN GLOBAL
int scanTime = 5;                                   // Thời gian quét BLE (giây)
BLEScan* pBLEScan;                                  // Con trỏ để quản lí việc quét
BLE
BLEClient* pClient;                                 // Con trỏ đến client BLE để kết
nối đến server BLE
bool doConnect = false;                             // Biến để xác định có thực hiện
kết nối hay không
bool connected = false;                             // Biến để xác định trạng thái
đã kết nối hay chưa
BLEAdvertisedDevice* myDevice;                      // Con trỏ đến đối tượng
BLEAdvertisedDevice đại diện cho thiết bị đã tìm thấy
BLERemoteCharacteristic* pRemoteCharacteristic;  // Con trỏ đến đặc tính của
server BLE
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Định nghĩa layout của bàn phím
const byte ROWS = 4;  // Số hàng
const byte COLS = 4;  // Số cột
char keys[ROWS][COLS] = {
  { '1', '2', '3', 'A' },
  { '4', '5', '6', 'B' },
  { '7', '8', '9', 'C' },
  { '*', '0', '#', 'D' }
};

// trên esp32 từ chân 1-8
byte rowPins[ROWS] = { 18, 23, 12, 13 };  // Các chân kết nối với các hàng
byte colPins[COLS] = { 14, 15, 16, 17 };  // Các chân kết nối với các cột
```

```
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

String inputData = "";     // Chuỗi lưu dữ liệu nhập từ bàn phím
String receivedData = "";  // Biến lưu dữ liệu nhận từ server

// THÔNG TIN VỀ SERVER
const char* targetName = "BLE_SERVER";            // tên
const char* targetAddress = "c8:2e:18:25:e4:62";  // địa chỉ
BLEUUID serviceUUID("4fafc201-1fb5-459e-8fcc-c5c9c331914b");
BLEUUID charUUID("beb5483e-36e1-4688-b7f5-ea07361b26a8");

class MyAdvertisedDeviceCallbacks : public BLEAdvertisedDeviceCallbacks {
  void onResult(BLEAdvertisedDevice advertisedDevice) {
    Serial.printf("Advertised Device: %s \n",
advertisedDevice.toString().c_str());

    if (advertisedDevice.getName() == targetName ||
advertisedDevice.getAddress().toString() == targetAddress) {
      Serial.println("Target device found. Stopping scan...");
      pBLEScan->stop();
      myDevice = new BLEAdvertisedDevice(advertisedDevice);
      doConnect = true;
    }
  }
};

// Hàm callback khi nhận thông báo từ server BLE
void notifyCallback(BLERemoteCharacteristic* pRemoteCharacteristic, uint8_t*
pData, size_t length, bool isNotify) {
  receivedData = "";  // Clear dữ liệu cũ trước khi ghi dữ liệu mới
  for (size_t i = 0; i < length; i++) {
    receivedData += (char)pData[i];
  }

  lcd.setCursor(0, 1);
  lcd.print("                ");
  lcd.setCursor(0, 1);
  lcd.print("rc:" + receivedData);

  Serial.print("Received data: ");
  Serial.println(receivedData);
}

// Kết nối tới server BLE và đăng ký callback
void connectToServer() {
```

```
  Serial.println("Connecting to server...");
  pClient = BLEDevice::createClient();
  if (pClient->connect(myDevice)) {
    Serial.println("Connected to server.");
    connected = true;

    // Tìm server service "4fafc201-1fb5-459e-8fcc-c5c9c331914b"
    BLERemoteService* pRemoteService = pClient->getService(serviceUUID);
    if (pRemoteService == nullptr) {
      Serial.println("Failed to find the service UUID. Disconnecting...");
      pClient->disconnect();
      connected = false;
      return;
    }

    // Tìm server charUUID "beb5483e-36e1-4688-b7f5-ea07361b26a8"
    pRemoteCharacteristic = pRemoteService->getCharacteristic(charUUID);
    if (pRemoteCharacteristic == nullptr) {
      Serial.println("Failed to find the characteristic UUID. Disconnecting...");
      pClient->disconnect();
      connected = false;
      return;
    }

    // Đăng ký hàm callback để nhận thông báo từ server
    if (pRemoteCharacteristic->canNotify()) {
      pRemoteCharacteristic->registerForNotify(notifyCallback);
      Serial.println("Registered for notifications.");
    }
  } else {
    Serial.println("Failed to connect to the server.");
  }
}

void setup() {
  Serial.begin(9600);
  Serial.println("Starting BLE scan...");

  lcd.init();       // khởi tạo LCD
  lcd.backlight();  // bật đèn nền lcd
  lcd.setCursor(0, 0);
  lcd.print("sd:");
  lcd.setCursor(0, 1);
  lcd.print("rc:");
```

```
  BLEDevice::init("");
  pBLEScan = BLEDevice::getScan();
  pBLEScan->setAdvertisedDeviceCallbacks(new MyAdvertisedDeviceCallbacks());
  pBLEScan->setActiveScan(true);
  pBLEScan->setInterval(100);
  pBLEScan->setWindow(99);

  keypad.setDebounceTime(50);  // Tăng thời gian debounce lên 50ms
}

void loop() {
  if (connected) {
    char key = keypad.getKey();
    if (key) {
      Serial.print("Key pressed: ");
      Serial.println(key);

      if (key == '#') {
        lcd.setCursor(0, 0);
        lcd.print("                ");
        if (pRemoteCharacteristic->canWrite() && inputData.length() > 0) {
          pRemoteCharacteristic->writeValue((uint8_t*)inputData.c_str(),
inputData.length(), false);
          Serial.print("Data sent to server: ");
          Serial.println(inputData);
          lcd.setCursor(0, 0);
          lcd.print("sd:" + inputData);
          inputData = "";
        }
      } else {
        inputData += key;
      }
    }
    delay(100);
    return;
  }

  if (doConnect) {
    connectToServer();
    doConnect = false;
  } else {
    BLEScanResults* foundDevices = pBLEScan->start(scanTime, false);
    Serial.print("Devices found: ");
    Serial.println(foundDevices->getCount());
    Serial.println("Scan done!");
```

```
    pBLEScan->clearResults();
    delay(2000);
  }
}
```

+Server

```
#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEServer.h>
#include <Keypad.h>
#include <LiquidCrystal_I2C.h>  // thư viện LCD

#define SERVICE_UUID "4fafc201-1fb5-459e-8fcc-c5c9c331914b"
#define CHARACTERISTIC_UUID "beb5483e-36e1-4688-b7f5-ea07361b26a8"

// Cấu hình bàn phím 4x4
const byte ROWS = 4;  // 4 hàng
const byte COLS = 4;  // 4 cột
char keys[ROWS][COLS] = {
  { '1', '2', '3', 'A' },
  { '4', '5', '6', 'B' },
  { '7', '8', '9', 'C' },
  { '*', '0', '#', 'D' }
};

byte rowPins[ROWS] = { 18, 23, 12, 13 };  // Kết nối đến các chân của hàng
byte colPins[COLS] = { 14, 15, 16, 17 };  // Kết nối đến các chân của cột

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

LiquidCrystal_I2C lcd(0x27, 16, 2);
String inputData = "";  // Biến để lưu dữ liệu từ các phím nhấn

BLECharacteristic *pCharacteristic;

// Tạo một lớp để xử lý việc nhận dữ liệu từ client
class MyCallbacks : public BLECharacteristicCallbacks {
  void onWrite(BLECharacteristic *pCharacteristic) {
    String value = pCharacteristic->getValue();  // Sử dụng String thay vì
std::string
    if (value.length() > 0) {
      lcd.setCursor(0, 1);
      lcd.print("                ");
      lcd.setCursor(0, 1);
```

```
      lcd.print("rc:" + value);
      Serial.print("Received Value: ");
      Serial.println(value);  // Hiển thị dữ liệu nhận được từ client
    }
  }
};

void setup() {
  Serial.begin(9600);
  Serial.println("Starting BLE work!");

  lcd.init();       // khởi tạo LCD
  lcd.backlight();  // bật đèn nền lcd
  lcd.setCursor(0, 0);
  lcd.print("sd:");
  lcd.setCursor(0, 1);
  lcd.print("rc:");

  BLEDevice::init("BLE_SERVER");
  BLEServer *pServer = BLEDevice::createServer();
  BLEService *pService = pServer->createService(SERVICE_UUID);
  pCharacteristic = pService->createCharacteristic(
    CHARACTERISTIC_UUID,
    BLECharacteristic::PROPERTY_READ | BLECharacteristic::PROPERTY_WRITE |
BLECharacteristic::PROPERTY_NOTIFY);

  pCharacteristic->setValue("Hello World says Neil");
  pCharacteristic->setCallbacks(new MyCallbacks());  // Đăng ký callback
  pService->start();

  BLEAdvertising *pAdvertising = BLEDevice::getAdvertising();
  pAdvertising->addServiceUUID(SERVICE_UUID);
  pAdvertising->setScanResponse(true);
  pAdvertising->setMinPreferred(0x06);
  pAdvertising->setMinPreferred(0x12);
  BLEDevice::startAdvertising();

  keypad.setDebounceTime(50);  // Tăng thời gian debounce lên 50ms

  Serial.println("Characteristic defined! Now you can read it in your phone!");
}

void loop() {
  char key = keypad.getKey();  // Lấy phím được nhấn từ bàn phím 4x4
```

```
  if (key) {
    Serial.print("Key pressed: ");
    Serial.println(key);

    if (key == '#') {
      lcd.setCursor(0, 0);
      lcd.print("                ");
      // Gửi dữ liệu khi nhấn phím '#'
      if (inputData.length() > 0) {
        pCharacteristic->setValue(inputData.c_str());  // Cập nhật giá trị mới
cho characteristic
        pCharacteristic->notify();                     // Gửi thông báo tới
scanner

        Serial.print("Data sent to scanner: ");
        Serial.println(inputData);

        lcd.setCursor(0, 0);
        lcd.print("sd:" + inputData);

        inputData = "";  // Xóa chuỗi sau khi gửi
      }
    } else {
      // Thêm ký tự vào inputData nếu không phải là phím '#'
      inputData += key;
    }
  }

  delay(100);  // Giảm tải cho vi xử lý
}
```

-Chương trình truyền dữ liệu liên tục giữa hai thiết bị ESP32.

+Scanner:

```
// THƯ VIỆN
#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEScan.h>
#include <BLEAdvertisedDevice.h>
#include <BLEClient.h>
#include <Keypad.h>
#include <LiquidCrystal_I2C.h>  // thư viện LCD

// BIẾN GLOBAL
```

```
int scanTime = 5;                                  // Thời gian quét BLE (giây)
BLEScan* pBLEScan;                                 // Con trỏ để quản lí việc quét
BLE
BLEClient* pClient;                                // Con trỏ đến client BLE để kết
nối đến server BLE
bool doConnect = false;                            // Biến để xác định có thực hiện
kết nối hay không
bool connected = false;                            // Biến để xác định trạng thái
đã kết nối hay chưa
BLEAdvertisedDevice* myDevice;                     // Con trỏ đến đối tượng
BLEAdvertisedDevice đại diện cho thiết bị đã tìm thấy
BLERemoteCharacteristic* pRemoteCharacteristic;  // Con trỏ đến đặc tính của
server BLE
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Định nghĩa layout của bàn phím
const byte ROWS = 4;  // Số hàng
const byte COLS = 4;  // Số cột
char keys[ROWS][COLS] = {
  { '1', '2', '3', 'A' },
  { '4', '5', '6', 'B' },
  { '7', '8', '9', 'C' },
  { '*', '0', '#', 'D' }
};

// trên esp32 từ chân 1-8
byte rowPins[ROWS] = { 18, 23, 12, 13 };  // Các chân kết nối với các hàng
byte colPins[COLS] = { 14, 15, 16, 17 };  // Các chân kết nối với các cột

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

String inputData = "";     // Chuỗi lưu dữ liệu nhập từ bàn phím
String receivedData = "";  // Biến lưu dữ liệu nhận từ server

// THÔNG TIN VỀ SERVER
const char* targetName = "BLE_SERVER";            // tên
const char* targetAddress = "c8:2e:18:25:e4:62";  // địa chỉ
BLEUUID serviceUUID("4fafc201-1fb5-459e-8fcc-c5c9c331914b");
BLEUUID charUUID("beb5483e-36e1-4688-b7f5-ea07361b26a8");

class MyAdvertisedDeviceCallbacks : public BLEAdvertisedDeviceCallbacks {
  void onResult(BLEAdvertisedDevice advertisedDevice) {
    Serial.printf("Advertised Device: %s \n",
advertisedDevice.toString().c_str());
```

```
    if (advertisedDevice.getName() == targetName ||
advertisedDevice.getAddress().toString() == targetAddress) {
      Serial.println("Target device found. Stopping scan...");
      pBLEScan->stop();
      myDevice = new BLEAdvertisedDevice(advertisedDevice);
      doConnect = true;
    }
  }
};

// Hàm callback khi nhận thông báo từ server BLE
void notifyCallback(BLERemoteCharacteristic* pRemoteCharacteristic, uint8_t*
pData, size_t length, bool isNotify) {
  receivedData = "";  // Clear dữ liệu cũ trước khi ghi dữ liệu mới
  for (size_t i = 0; i < length; i++) {
    receivedData += (char)pData[i];
  }

  lcd.setCursor(0, 1);
  lcd.print("                ");
  lcd.setCursor(0, 1);
  lcd.print("rc:" + receivedData);

  Serial.print("Received data: ");
  Serial.println(receivedData);
}

// Kết nối tới server BLE và đăng ký callback
void connectToServer() {
  Serial.println("Connecting to server...");
  pClient = BLEDevice::createClient();
  if (pClient->connect(myDevice)) {
    Serial.println("Connected to server.");
    connected = true;

    // Tìm server service "4fafc201-1fb5-459e-8fcc-c5c9c331914b"
    BLERemoteService* pRemoteService = pClient->getService(serviceUUID);
    if (pRemoteService == nullptr) {
      Serial.println("Failed to find the service UUID. Disconnecting...");
      pClient->disconnect();
      connected = false;
      return;
    }

    // Tìm server charUUID "beb5483e-36e1-4688-b7f5-ea07361b26a8"
```

```
    pRemoteCharacteristic = pRemoteService->getCharacteristic(charUUID);
    if (pRemoteCharacteristic == nullptr) {
      Serial.println("Failed to find the characteristic UUID. Disconnecting...");
      pClient->disconnect();
      connected = false;
      return;
    }

    // Đăng ký hàm callback để nhận thông báo từ server
    if (pRemoteCharacteristic->canNotify()) {
      pRemoteCharacteristic->registerForNotify(notifyCallback);
      Serial.println("Registered for notifications.");
    }
  } else {
    Serial.println("Failed to connect to the server.");
  }
}

void setup() {
  Serial.begin(115200);
  Serial.println("Starting BLE scan...");

  lcd.init();       // khởi tạo LCD
  lcd.backlight();  // bật đèn nền lcd
  lcd.setCursor(0, 0);
  lcd.print("sd:");
  lcd.setCursor(0, 1);
  lcd.print("rc:");

  BLEDevice::init("");
  pBLEScan = BLEDevice::getScan();
  pBLEScan->setAdvertisedDeviceCallbacks(new MyAdvertisedDeviceCallbacks());
  pBLEScan->setActiveScan(true);
  pBLEScan->setInterval(100);
  pBLEScan->setWindow(99);

  keypad.setDebounceTime(50);  // Tăng thời gian debounce lên 50ms
}

void loop() {
  if (connected) {
    char key = keypad.getKey();
    if (key) {
      Serial.print("Key pressed: ");
      Serial.println(key);
```

```
      // Thêm ký tự vừa nhập vào chuỗi dữ liệu
      inputData += key;

      // Nếu chuỗi vượt quá 13 ký tự (16 ký tự trừ "sd:" = 13), reset nội dung
      if (inputData.length() > 13) {
        inputData = "";  // Reset chuỗi nội dung, giữ lại tiền tố sd:
      }

      // Gửi dữ liệu liên tục mỗi khi có phím được nhấn
      if (pRemoteCharacteristic->canWrite()) {
        pRemoteCharacteristic->writeValue((uint8_t*)inputData.c_str(),
inputData.length(), false);
        Serial.print("Data sent to server: ");
        Serial.println(inputData);

        // Hiển thị chuỗi dữ liệu trên LCD
        lcd.setCursor(0, 0);
        lcd.print("                ");
        lcd.setCursor(0, 0);
        lcd.print("sd:" + inputData);
      }
    }
    delay(100);  // Giảm tần số quét để tránh xung đột
    return;
  }

  if (doConnect) {
    connectToServer();
    doConnect = false;
  } else {
    BLEScanResults* foundDevices = pBLEScan->start(scanTime, false);
    Serial.print("Devices found: ");
    Serial.println(foundDevices->getCount());
    Serial.println("Scan done!");
    pBLEScan->clearResults();
    delay(2000);
  }
}
```

+Server:

```
#include <BLEDevice.h>

#include <BLEUtils.h>
```

```
#include <BLEServer.h>
#include <Keypad.h>
#include <LiquidCrystal_I2C.h>  // thư viện LCD

#define SERVICE_UUID "4fafc201-1fb5-459e-8fcc-c5c9c331914b"
#define CHARACTERISTIC_UUID "beb5483e-36e1-4688-b7f5-ea07361b26a8"

// Cấu hình bàn phím 4x4
const byte ROWS = 4;  // 4 hàng
const byte COLS = 4;  // 4 cột
char keys[ROWS][COLS] = {
  { '1', '2', '3', 'A' },
  { '4', '5', '6', 'B' },
  { '7', '8', '9', 'C' },
  { '*', '0', '#', 'D' }
};

byte rowPins[ROWS] = { 18, 23, 12, 13 };  // Kết nối đến các chân của hàng
byte colPins[COLS] = { 14, 15, 16, 17 };  // Kết nối đến các chân của cột

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

LiquidCrystal_I2C lcd(0x27, 16, 2);
String inputData = "";  // Biến để lưu dữ liệu từ các phím nhấn

BLECharacteristic *pCharacteristic;

// Tạo một lớp để xử lý việc nhận dữ liệu từ client
class MyCallbacks : public BLECharacteristicCallbacks {
  void onWrite(BLECharacteristic *pCharacteristic) {
    String value = pCharacteristic->getValue();  // Sử dụng String thay vì
std::string
    if (value.length() > 0) {
      lcd.setCursor(0, 1);
      lcd.print("                ");
      lcd.setCursor(0, 1);
      lcd.print("rc:" + value);
      Serial.print("Received Value: ");
      Serial.println(value);  // Hiển thị dữ liệu nhận được từ client
    }
  }
};

void setup() {
  Serial.begin(115200);
```

```
  Serial.println("Starting BLE work!");

  lcd.init();       // khởi tạo LCD
  lcd.backlight();  // bật đèn nền lcd
  lcd.setCursor(0, 0);
  lcd.print("sd:");
  lcd.setCursor(0, 1);
  lcd.print("rc:");

  BLEDevice::init("BLE_SERVER");
  BLEServer *pServer = BLEDevice::createServer();
  BLEService *pService = pServer->createService(SERVICE_UUID);
  pCharacteristic = pService->createCharacteristic(
    CHARACTERISTIC_UUID,
    BLECharacteristic::PROPERTY_READ | BLECharacteristic::PROPERTY_WRITE |
BLECharacteristic::PROPERTY_NOTIFY);

  pCharacteristic->setValue("Hello World says Neil");
  pCharacteristic->setCallbacks(new MyCallbacks());  // Đăng ký callback
  pService->start();

  BLEAdvertising *pAdvertising = BLEDevice::getAdvertising();
  pAdvertising->addServiceUUID(SERVICE_UUID);
  pAdvertising->setScanResponse(true);
  pAdvertising->setMinPreferred(0x06);
  pAdvertising->setMinPreferred(0x12);
  BLEDevice::startAdvertising();

  keypad.setDebounceTime(50);  // Tăng thời gian debounce lên 50ms

  Serial.println("Characteristic defined! Now you can read it in your phone!");
}

void loop() {
  char key = keypad.getKey();  // Lấy phím được nhấn từ bàn phím 4x4

  if (key) {
    Serial.print("Key pressed: ");
    Serial.println(key);

    // Thêm ký tự vừa nhập vào chuỗi dữ liệu
    inputData += key;

    // Nếu chuỗi vượt quá 13 ký tự (16 ký tự trừ "sd:" = 13), reset nội dung
    if (inputData.length() > 13) {
```

```
      inputData = "";  // Reset chuỗi nội dung, giữ lại tiền tố sd:
    }

    // Gửi dữ liệu liên tục mỗi khi có phím được nhấn
    pCharacteristic->setValue(inputData.c_str());  // Cập nhật giá trị mới cho
characteristic
    pCharacteristic->notify();                     // Gửi thông báo tới scanner

    Serial.print("Data sent to scanner: ");
    Serial.println(inputData);

    // Hiển thị chuỗi dữ liệu trên LCD
    lcd.setCursor(0, 0);
    lcd.print("                ");
    lcd.setCursor(0, 0);
    lcd.print("sd:" + inputData);
  }

  delay(100);  // Giảm tải cho vi xử lý
}
```

-Chương trình truyền dữ liệu giữa thiết bị ESP32 và các thiết bị điện thoại di động.

```
// KHAI BÁO THƯ VIỆN

#include "BluetoothSerial.h"
#include <LiquidCrystal_I2C.h>  // thư viện LCD
#include <Keypad.h>             // thư viện key_4x4

// KHAI BÁO BIẾN
BluetoothSerial BT;
LiquidCrystal_I2C lcd(0x27, 16, 2);
// #define LED_PIN 18  // khai báo chân cho con led
const byte ROWS = 4;  // Số hàng
const byte COLS = 4;  // Số cột

// Định nghĩa layout của bàn phím
char keys[ROWS][COLS] = {
  { '1', '2', '3', 'A' },
```

```
  { '4', '5', '6', 'B' },
  { '7', '8', '9', 'C' },
  { '*', '0', '#', 'D' }
};

// trên esp32 từ chân 1-8
// cột: 1-4 {14, 15, 16, 17}
// hàng: 5-8 {18, 19, 23, 5}
byte rowPins[ROWS] = { 18, 19, 23, 5 };   // Các chân kết nối với các hàng
byte colPins[COLS] = { 14, 15, 16, 17 };  // Các chân kết nối với các cột

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

String inputString = "";  // Chuỗi để lưu ký tự

// HÀM KHỞI TẠO
void setup() {
  Serial.begin(115200);
  // pinMode(LED_PIN, OUTPUT);  // khởi tạo led
  BT.begin("Esp32");
  Serial.println("The device started, now you can pair it with bluetooth!");
  lcd.init();       // khởi tạo LCD
  lcd.backlight();  // bật đèn nền lcd
}

// VÒNG LẶP
void loop() {
  // key 4x4
  char key = keypad.getKey();
  if (key) {
    // Thêm ký tự nhấn vào chuỗi
    inputString += key;

    // Nếu chuỗi dài hơn 32 ký tự, xóa và bắt đầu lại
    if (inputString.length() > 32) {
      inputString = "";
      lcd.clear();
    }

    // Hiển thị chuỗi lên LCD
    lcd.clear();
    if (inputString.length() <= 16) {
      // Nếu chuỗi ngắn hơn hoặc bằng 16 ký tự, hiển thị trên dòng đầu tiên
      lcd.setCursor(0, 0);
      lcd.print(inputString);
```

```
  } else {
    // Nếu chuỗi dài hơn 16 ký tự, chia thành 2 phần
    String line1 = inputString.substring(0, 16);   // 16 ký tự đầu
    String line2 = inputString.substring(16, 32);  // 16 ký tự tiếp theo
    lcd.setCursor(0, 0);
    lcd.print(line1);
    lcd.setCursor(0, 1);
    lcd.print(line2);
  }

  // In chuỗi ra Serial Monitor (tuỳ chọn)
  Serial.println(inputString);
  BT.println(inputString);
}

// Đọc và xử lý dữ liệu từ Serial
if (Serial.available()) {
  BT.write(Serial.read());
}

// Đọc và xử lý dữ liệu từ Bluetooth
if (BT.available()) {
  String btData = "";  // Biến để lưu chuỗi dữ liệu từ Bluetooth
  while (BT.available()) {
    char data1 = BT.read();  // Đọc từng ký tự từ Bluetooth

    // Kiểm tra xem ký tự có phải là ký tự in được không (ASCII từ 32 đến 126)
    if (data1 >= 32 && data1 <= 126) {
      btData += data1;  // Thêm ký tự hợp lệ vào chuỗi
    }
  }

  lcd.clear();  // Xóa toàn bộ LCD trước khi in dữ liệu mới

  // Kiểm tra nếu chuỗi dài hơn 16 ký tự
  if (btData.length() > 16) {
    // In 16 ký tự đầu lên hàng 0
    lcd.setCursor(0, 0);
    lcd.print(btData.substring(0, 16));

    // In phần còn lại xuống hàng 1
    lcd.setCursor(0, 1);
    lcd.print(btData.substring(16));
  } else {
    // Nếu chuỗi ngắn hơn hoặc bằng 16 ký tự, in lên hàng 0
```

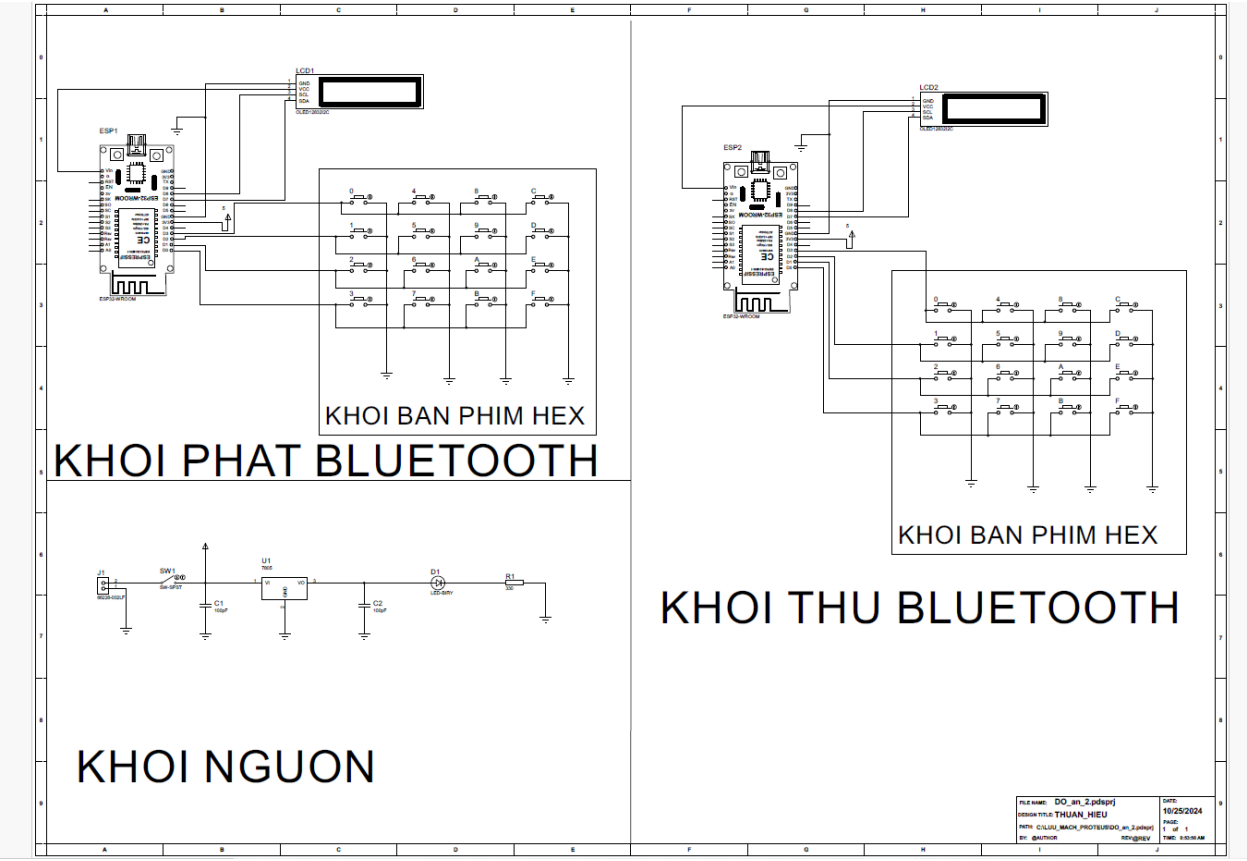
```
      lcd.setCursor(0, 0);
      lcd.print(btData);
    }

    Serial.println(btData);  // In dữ liệu lên Serial để kiểm tra
  }

  delay(20);  // Tránh quá tải bộ xử lý
}
```

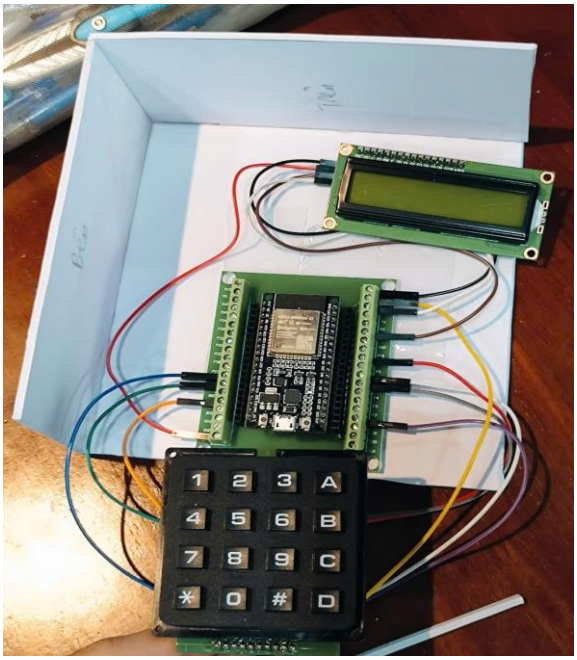
CHƯƠNG 4: KẾT QUẢ THỰC HIỆN

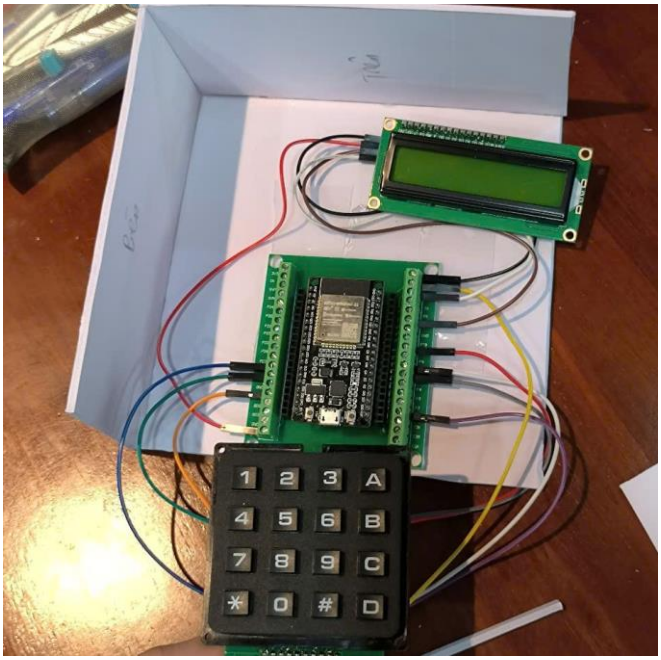
4.1. SƠ ĐỒ MẠCH HOÀN CHỈNH:



4.2. KẾT QUẢ HOẠT ĐỘNG CỦA HỆ THỐNG

Sau khi kiểm tra hệ thống hoạt động ổn định. Em tiến hành kết nối các module và linh kiện lại với nhau và thu được sản phẩm.





Mô hình sản phẩm

Hoạt động của sản phẩm:

Khi nhấn nút nhập dữ liệu cần truyền ở bên thiết bị phát thì thiết bị thu nhận được và hiện thị trên màn hình LCD và cũng đồng thời hiện thị trên LCD của thiết bị phát và ngược lại.

Để hiểu rõ hơn hoạt động của sản phẩm em có đính kèm video clip được đính kèm file sau:

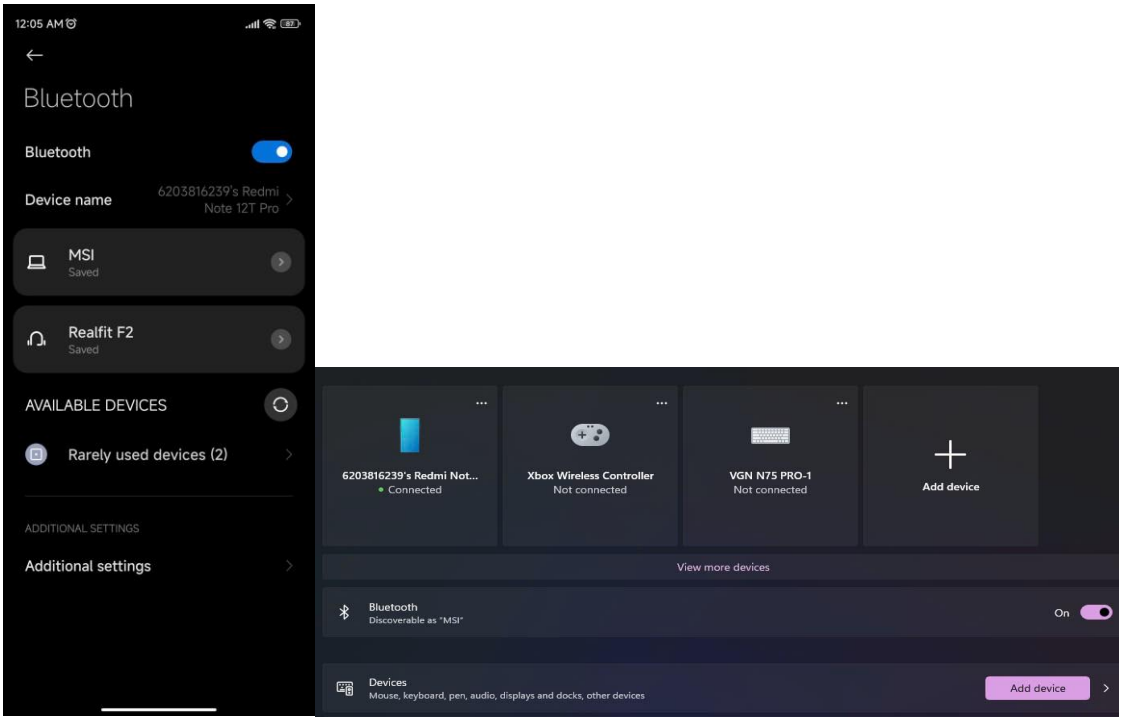
+ HỆ THỐNG BLUETOOTH TRUYỀN DỮ LIỆU LIÊN TỤC:
https://www.youtube.com/watch?v=Dkg3njfxeOs

+ HỆ THỐNG BLUETOOTH TRUYỀN DỮ LIỆU NGẮT QUẢN:

https://www.youtube.com/watch?v=p1s3aFyQwP4

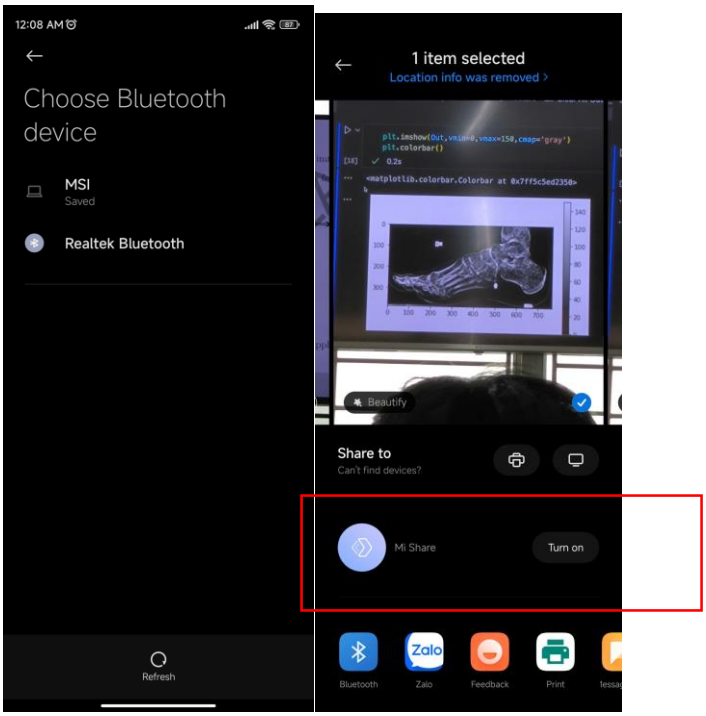
4.3. CÁCH KẾT NỐI ĐIỆN THOẠI VỚI MÁY TÍNH

B1: Bật Bluetooth của điện thoại và máy tính để bắt đầu kết nối.



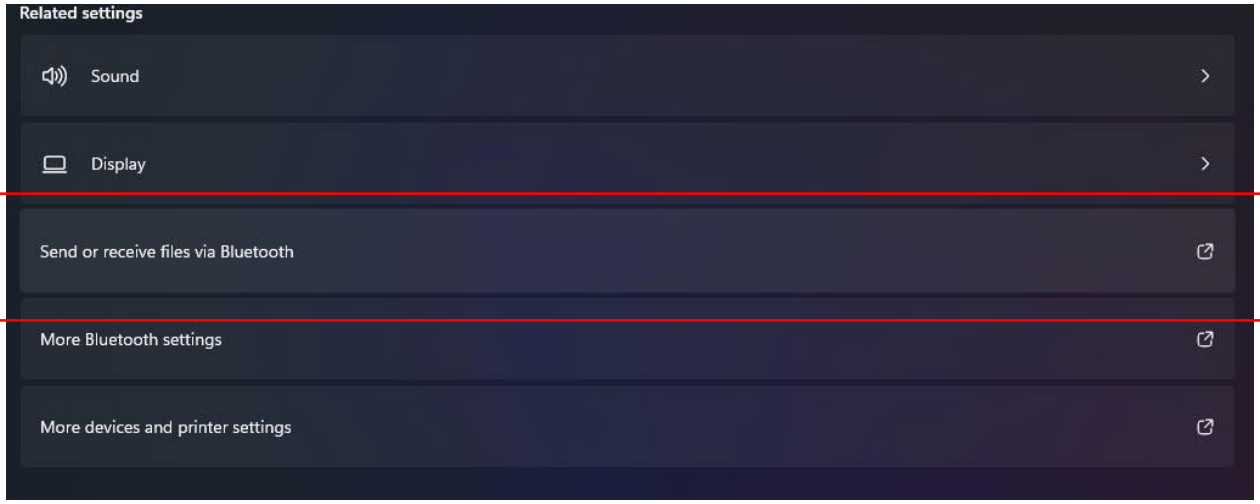
Sau khi bật Bluetooth lên thì sẽ hiện device name thiết bị mà mình muốn kết nối sau đó ta bấm vào thiết bị đó một trong hai thiết bị sẽ gửi mã kết nối tới thiết bị còn lại để kết nối sau đó ta bấm kết nối chờ trong vài giây khi hiện lên chữ connected thì đã kết nối thành công

B2: Gửi file từ điện thoại lên máy tính.

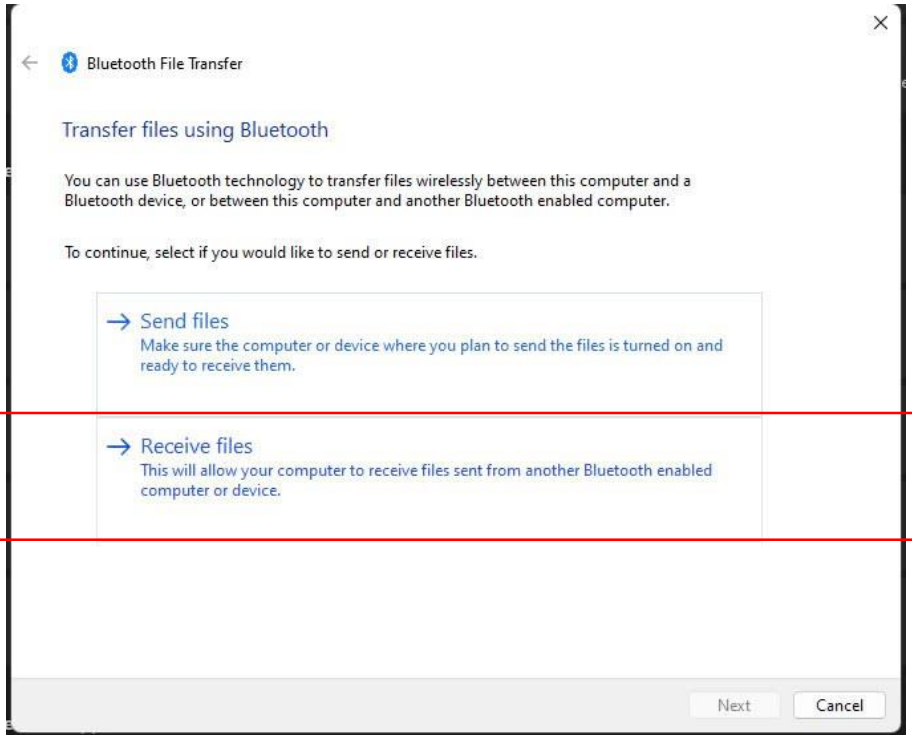


Để gửi file từ điện thoại đến máy tính thì ta chọn file mà ta muốn gửi đi bấm vào phần share to nhấn vào thiết mà mình muốn gửi.

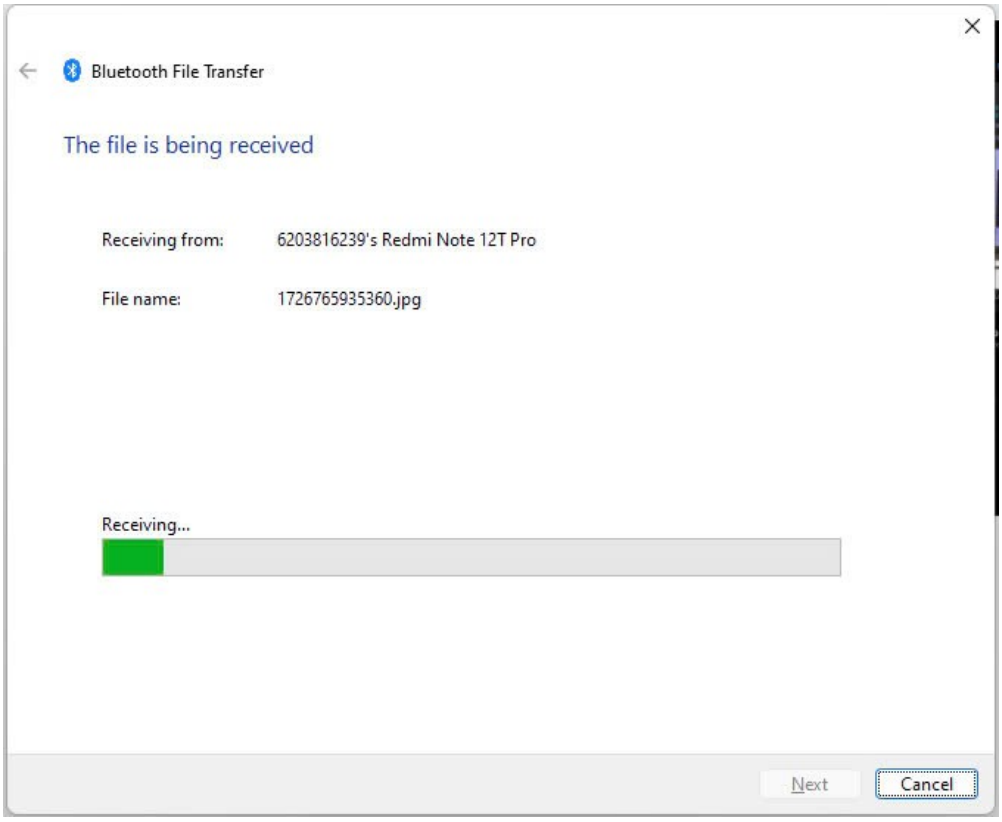
B3: Nhận file từ điện thoại trên máy tính.



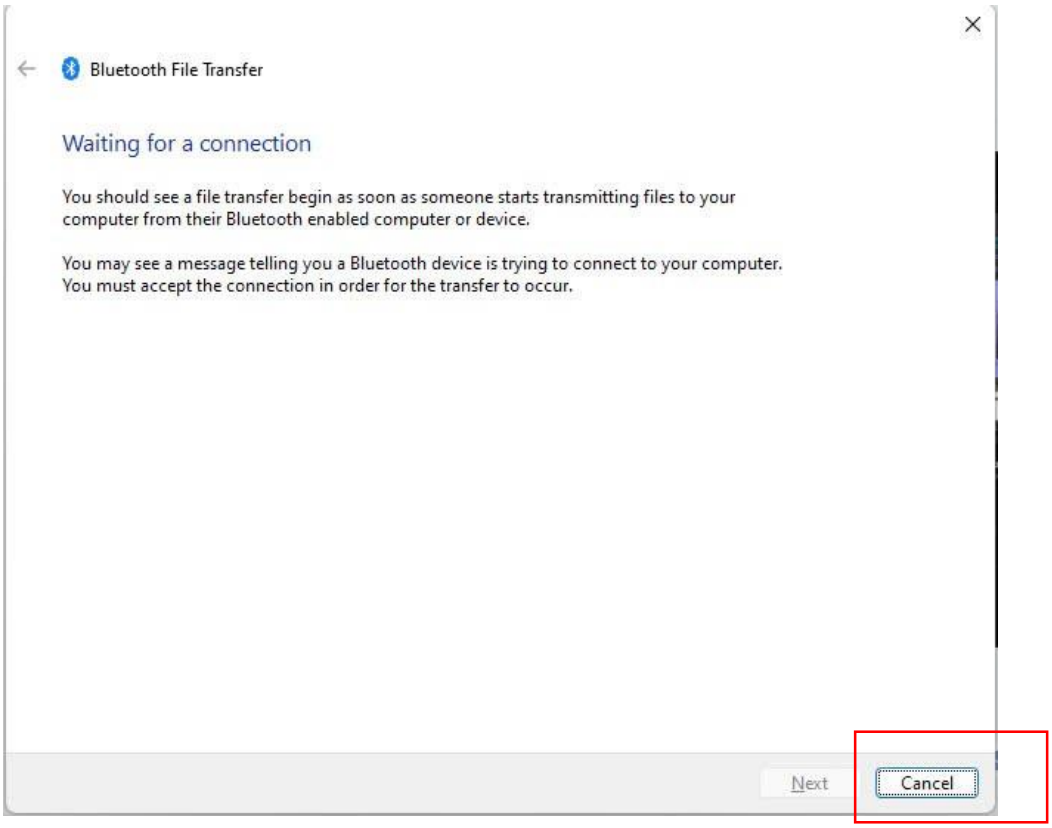
Ở máy tính ta vào phần setting→ send or receive file via Bluetooth



Tiếp tục ta bấm receive files hiển thị trên màn hình.



Sau khi bấm receivefile file bắt đầu được truyền từ điện thoại sang.



Khi file đã chạy xong ta nhấn cancel để kết thúc và file đã được nhận trên máy tính.

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 KẾT LUẬN

Trong đồ án này, em đã nghiên cứu và thiết kế một hệ thống truyền dữ liệu không dây sử dụng công nghệ Bluetooth Low Energy (BLE) trên vi điều khiển ESP32. Hệ thống bao gồm ba chương trình phục vụ các mục đích khác nhau: truyền dữ liệu không liên tục giữa hai thiết bị ESP32, truyền dữ liệu liên tục giữa hai thiết bị ESP32, và truyền dữ liệu giữa ESP32 với các thiết bị điện thoại di động. Qua quá trình nghiên cứu, thiết kế và thực hiện, em đã đạt được những kết quả sau:

-Hiểu rõ về các thành phần và nguyên lý hoạt động:

+ESP32: Em đã tìm hiểu và ứng dụng vi điều khiển ESP32 để triển khai các chức năng BLE, bao gồm phát hiện, kết nối, và trao đổi dữ liệu. ESP32 đã chứng minh là một nền tảng mạnh mẽ và phù hợp cho các ứng dụng IoT.

+Bluetooth Low Energy (BLE): Em đã nắm rõ nguyên lý hoạt động của BLE, cách quét thiết bị, kết nối, và truyền nhận dữ liệu giữa các thiết bị.

+Giao tiếp giữa thiết bị: Hệ thống đã được thiết kế để xử lý cả hai chế độ truyền dữ liệu liên tục và không liên tục, đảm bảo tính linh hoạt trong các ứng dụng thực tế.

+Kết nối với điện thoại: Em đã triển khai thành công việc truyền dữ liệu giữa ESP32 và các thiết bị di động, giúp hệ thống dễ dàng tương tác với người dùng thông qua ứng dụng.

-Thiết kế và xây dựng hệ thống hoạt động ổn định:

+Hệ thống đảm bảo kết nối ổn định, khả năng truyền nhận dữ liệu nhanh chóng và đáng tin cậy.

+Các chương trình được thiết kế rõ ràng, tối ưu, phù hợp với từng mục đích sử dụng.

+Kết nối BLE hoạt động hiệu quả, đảm bảo tốc độ và độ tin cậy trong quá trình trao đổi dữ liệu.

-Thành công trong việc triển khai và thử nghiệm hệ thống:

+Các chương trình đã được triển khai và thử nghiệm thành công, hoạt động đúng như yêu cầu đề ra.

+Hệ thống đáp ứng tốt các tiêu chí về hiệu suất, độ ổn định, và khả năng mở rộng.

Nhìn chung, dự án đã hoàn thành mục tiêu ban đầu, tạo ra một hệ thống truyền dữ liệu Bluetooth hiệu quả, linh hoạt và dễ sử dụng. Hệ thống này có tiềm năng ứng dụng rộng rãi trong các lĩnh vực như tự động hóa, giám sát từ xa, và các ứng dụng IoT thông minh.

5.2 MỘT SỐ HẠN CHẾ

Dù hệ thống truyền dữ liệu Bluetooth đáp ứng các yêu cầu ban đầu, vẫn còn một số hạn chế cần được xem xét và cải thiện trong các phiên bản tiếp theo. Dưới đây là một số hạn chế cụ thể của hệ thống mà em nhận thấy được:

-Phạm vi kết nối hạn chế: Phạm vi hoạt động của BLE chỉ giới hạn trong khoảng 10-15 mét, gây khó khăn trong các ứng dụng yêu cầu khoảng cách xa.

-Độ ổn định trong môi trường phức tạp: Tín hiệu BLE dễ bị ảnh hưởng bởi nhiễu từ các thiết bị không dây khác hoặc vật cản lớn.

-Hiệu suất năng lượng chưa tối ưu: Truyền dữ liệu liên tục có thể làm tiêu hao năng lượng nhanh, ảnh hưởng đến thời gian hoạt động khi dùng pin.

-Tốc độ truyền dữ liệu: BLE phù hợp với dữ liệu nhỏ, nhưng không đáp ứng tốt khi cần truyền lượng lớn dữ liệu hoặc dữ liệu thời gian thực với tốc độ cao.

-Khả năng mở rộng hạn chế: Hệ thống chưa được tối ưu để kết nối với nhiều thiết bị cùng lúc, gây khó khăn khi mở rộng quy mô.

-Tính tương thích: Một số thiết bị di động có thể không hỗ trợ đầy đủ các tính năng BLE, làm giảm khả năng kết nối và hiệu quả sử dụng.

-Phản hồi chậm trong điều kiện phức tạp: Hệ thống có thể phản ứng chậm khi tín hiệu kết nối không ổn định hoặc khi môi trường thay đổi đột ngột.

-Độ phức tạp trong cấu hình: Việc thiết lập hệ thống yêu cầu hiểu biết về BLE và ESP32, gây khó khăn cho người dùng không chuyên.

5.3 HƯỚNG PHÁT TRIỂN

Mặc dù hệ thống truyền dữ liệu Bluetooth hiện tại đã hoạt động hiệu quả, em nhận thấy vẫn còn nhiều cơ hội để cải tiến và mở rộng. Dưới đây là một số hướng phát triển tiềm năng cho hệ thống mà em sẽ định hướng tới:

-Tích hợp đa dạng cảm biến:Bổ sung các cảm biến như nhiệt độ, độ ẩm, khí gas, hoặc cảm biến chuyển động để mở rộng khả năng ứng dụng của hệ thống trong các lĩnh vực như giám sát môi trường, an ninh, hoặc tự động hóa nhà thông minh.

-Kết nối mạng và điều khiển từ xa:Tích hợp Wi-Fi hoặc các giao thức không dây khác (như Zigbee) để mở rộng phạm vi kết nối, cho phép hệ thống được điều khiển và giám sát từ xa thông qua ứng dụng di động hoặc giao diện web.

-Cải tiến giao thức truyền dữ liệu:Nghiên cứu và áp dụng các giao thức truyền dữ liệu tiên tiến hơn như MQTT hoặc WebSocket để nâng cao tốc độ, độ ổn định và khả năng bảo mật của hệ thống.

-Phát triển ứng dụng đa nền tảng:Xây dựng ứng dụng điều khiển trên các nền tảng phổ biến (Android, iOS) với giao diện thân thiện, giúp người dùng dễ dàng quản lý và tùy chỉnh hệ thống.

-Nâng cao thuật toán điều khiển:Áp dụng các thuật toán thông minh như học máy (machine learning) để hệ thống tự động tối ưu hóa quá trình kết nối, truyền dữ liệu và phản hồi theo điều kiện thực tế.

-Tăng cường bảo mật:Tích hợp các cơ chế bảo mật mạnh mẽ hơn như mã hóa dữ liệu AES hoặc giao thức bảo mật TLS để bảo vệ hệ thống khỏi các mối đe dọa an ninh mạng.

-Tối ưu hóa năng lượng:Sử dụng các chế độ tiết kiệm năng lượng của ESP32 và tối ưu hóa thời gian hoạt động để tăng tuổi thọ của pin, giảm tiêu thụ năng lượng trong các ứng dụng di động.

-Mở rộng khả năng tích hợp:Phát triển khả năng tương thích và tích hợp với các nền tảng IoT lớn như Blynk, Firebase, hoặc Google Cloud IoT để mở rộng phạm vi ứng dụng của hệ thống trong các dự án IoT quy mô lớn.

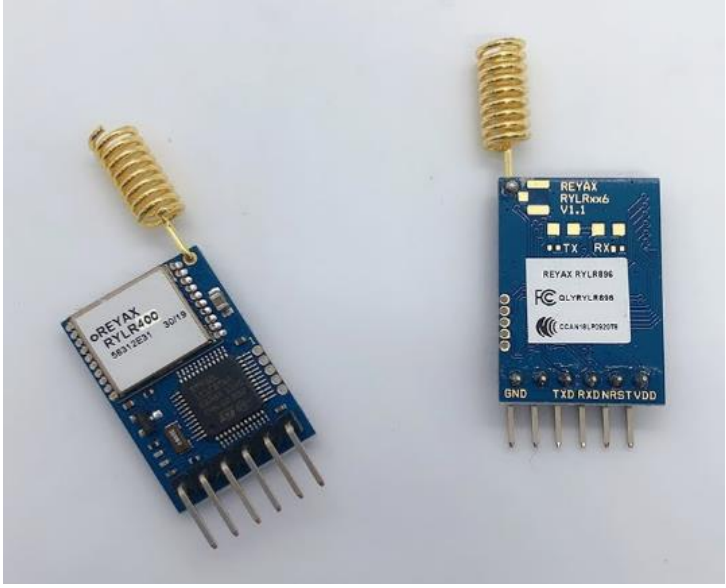
-Ứng dụng trong công nghiệp và y tế:Nghiên cứu và phát triển hệ thống để ứng dụng trong các lĩnh vực đặc thù như giám sát sức khỏe từ xa, quản lý kho thông minh, hoặc hệ thống tự động hóa công nghiệp.

PHẦN II: CÔNG NGHỆ LORA

1. Tổngquan:

LoRa (Long Range) là một công nghệ truyền thông không dây tầm xa, sử dụng phổ tần số thấp để truyền dữ liệu với tốc độ thấp, nhưng hiệu quả trên khoảng cách rất xa. Nó chủ yếu được sử dụng trong các ứng dụng IoT (Internet of Things) để kết nối các thiết bị từ xa mà tiêu thụ ít năng lượng.

LoRa điều chế RF cho mạng diện rộng công suất thấp (LPWAN) có khả năng truyền dữ liệu lên đến 5km ở khu vực đô thị và 10-15km ở khu vực nông thôn. Đặc điểm của công nghệ Lora là yêu cầu điện năng cực thấp, cho phép tạo ra các thiết bị hoạt động bằng pin với thời gian lên tới 10 năm.



Một số thiết bị Lora phổ biến

2. Kỹ thuật điều chế vô tuyến trong LORA:

Sử dụng một kỹ thuật điều chế trải phổ độc quyền có nguồn gốc từ công nghệ Chirp Spread Spectrum (CSS), LoRa cung cấp sự cân bằng giữa độ nhạy thu với tốc độ dữ liệu, hoạt động trong kênh băng thông cố định 125 KHz hoặc 500 KHz (đối với kênh uplink), và 500 KHz (cho các kênh downlink). Ngoài ra, LoRa sử dụng các hệ số lan truyền trực giao, cho phép duy trì tuổi thọ pin của các nút cuối bằng cách tối ưu hóa mức công suất và tốc độ dữ liệu.

Cách hoạt động của Chirp Spread Spectrem (CSS) trong LORA:

-Nguyên lý điều chế:

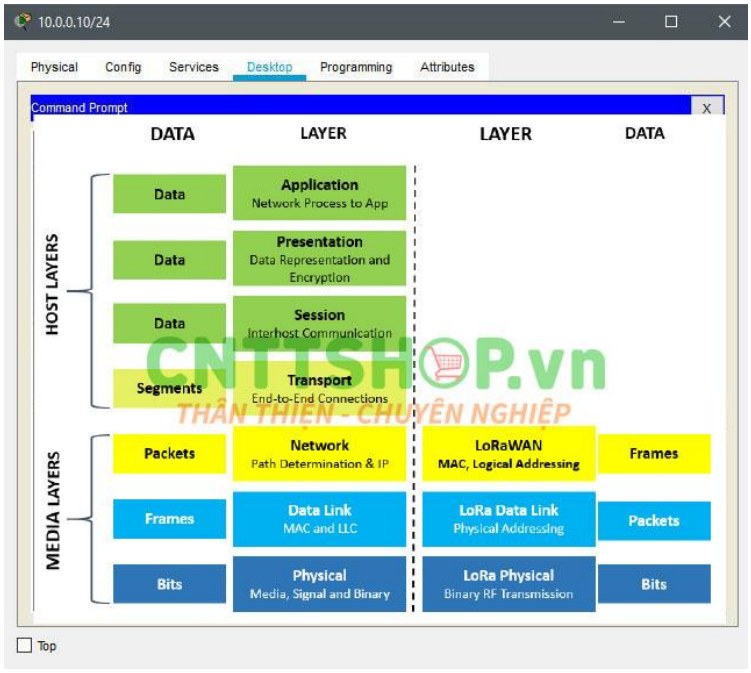
+Chirp: Trong CSS, dữ liệu được mã hóa vào các tín hiệu thay đổi tần số theo thời gian. Sự thay đổi tần số này được gọi là một "chirp". Một chirp bắt đầu từ tần số thấp và tăng lên tần số cao (up-chirp) hoặc bắt đầu từ tần số cao và giảm xuống tần số thấp (down-chirp).

+Spread (Phát tán): Sự thay đổi tần số này phát tán tín hiệu ra một băng tần rộng hơn, giúp tín hiệu trở nên mạnh mẽ hơn trong môi trường nhiễu.

+Mã hóa ký tự:Mỗi ký tự trong CSS được đại diện bởi một tín hiệu chirp. Thời gian và tần số của các chirp này được sử dụng để mã hóa dữ liệu.
+Băng tần tần số:LoRa hoạt động ở các băng tần dưới 1 GHz (ví dụ: 868 MHz ở châu Âu, 915 MHz ở Mỹ). Băng tần của các chirp có thể được điều chỉnh dựa trên khoảng cách giao tiếp và tốc độ dữ liệu yêu cầu.
+Tốc độ dữ liệu và khoảng cách:LoRa hỗ trợ nhiều tốc độ dữ liệu bằng cách điều chỉnh thời gian và băng tần của các chirp. Tốc độ dữ liệu thấp hơn tương ứng với

khoảng cách dài hơn và độ nhạy tốt hơn, trong khi tốc độ dữ liệu cao hơn cung cấp tốc độ giao tiếp nhanh hơn nhưng khoảng cách ngắn hơn.
+Sửa lỗi:LoRa tích hợp sửa lỗi trước (FEC) để nâng cao độ tin cậy. Dữ liệu được mã hóa với sự dư thừa để sửa lỗi có thể xảy ra trong quá trình truyền.

LoRa nằm hoàn toàn ở lớp vật lý trong mô hình OSI, tuy nhiên thay vì đi cáp, không khí được sử dụng như một phương tiện để vận chuyển sóng vô tuyến từ một bộ phát RF trong thiết bị IoT đến bộ thu RF trong gateway và ngược lại.



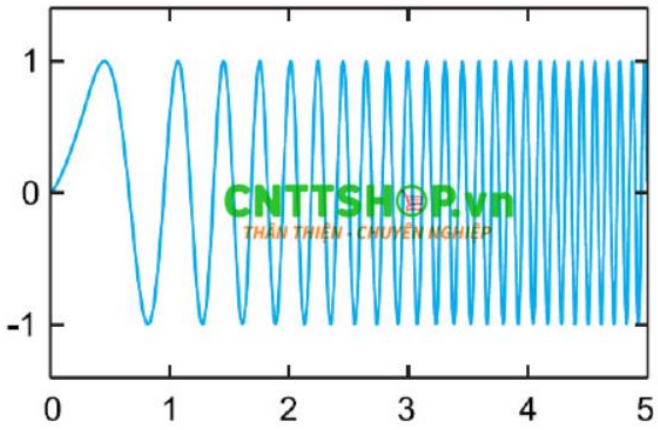
*cấu trúc của **LoRa và LoRaWAN**, thể hiện cách các lớp mạng này hoạt động dựa trên mô hình OSI.*

+ **LoRaWAN (Lớp Ứng dụng và Liên kết Dữ liệu)**: Chịu trách nhiệm về việc định địa chỉ logic và quản lý truy cập phương tiện truyền thông. LoRaWAN là giao thức lớp MAC (Medium Access Control) dành cho LoRa, quản lý việc truy cập mạng và địa chỉ của các thiết bị.

+ **LoRa Data Link (Lớp Liên kết Dữ liệu)**: Tương tự như lớp liên kết dữ liệu của mô hình OSI, lớp này chịu trách nhiệm về địa chỉ hóa vật lý và đảm bảo truyền tải dữ liệu thành công qua kết nối vật lý.

+ **LoRa Physical (Lớp Vật lý)**: Lớp này điều khiển việc truyền dữ liệu dưới dạng tín hiệu vô tuyến và xử lý mã hóa tín hiệu ở dạng sóng RF (Radio Frequency)

LoRa Chirp Spread Spectrum (CSS) của Semtech cung cấp giải pháp thay thế cho công nghệ điều chế truyền thống DSSS với chi phí và công suất thấp những vẫn rất mạnh mẽ. Trong điều chế LoRa, sự trải rộng phổ của tín hiệu đạt được bằng cách tạo ra một tín hiệu chirp thay đổi liên tục về tần số.



3. Các Module của LORA:

Các **module LoRa** là phần cứng chịu trách nhiệm truyền và nhận dữ liệu sử dụng công nghệ **LoRa (Long Range)**. Các module này thường được tích hợp trong các thiết bị IoT (Internet of Things) để cung cấp khả năng giao tiếp tầm xa với công suất thấp. Dưới đây là một số module LoRa phổ biến và các đặc điểm của chúng:

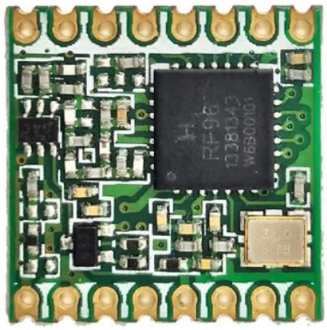
SX1276

- **Nhà sản xuất**: Semtech.
- **Băng tần hỗ trợ**: 137 MHz – 1020 MHz (thường sử dụng trong băng tần 868 MHz và 915 MHz).
- **Ứng dụng**: Module này rất phổ biến cho các dự án IoT yêu cầu khả năng truyền dữ liệu tầm xa và công suất thấp. Nó là nền tảng cho nhiều module LoRa thương mại khác.

Module SX1276

RFM95

- **Nhà sản xuất**: HopeRF.
- **Băng tần hỗ trợ**:RFM95: 868 MHz (Châu Âu) hoặc 915 MHz (Bắc Mỹ).
- **Tính năng**: Module này sử dụng chip SX1276 của Semtech và hỗ trợ giao tiếp LoRa với khoảng cách lên đến vài km, tùy vào môi trường.
- **Ứng dụng**: Phổ biến trong các ứng dụng tầm xa, như các hệ thống cảm biến, đo lường không dây và các dự án IoT.



Module RFM 95

RA-02

- **Nhà sản xuất**: Ai-Thinker.
- **Băng tần hỗ trợ**: 433 MHz.
- **Tính năng**: Sử dụng chip SX1278 của Semtech. Module này hỗ trợ LoRa với tầm phủ sóng lên tới 10 km trong môi trường mở.
- **Ứng dụng**: Phổ biến trong các dự án DIY, Arduino, và các ứng dụng IoT yêu cầu truyền dữ liệu tầm xa.

Module RA- 02

TÀI LIỆU THAM KHẢO

[1] Getting Started with Bluetooth Low Energy" by Kevin Townsend, Carles Cufi, Akiba, and Robert Davidson

[2] HC-05 Bluetooth Module Datasheet: https://components101.com/sites/default/files/component_datasheet/HC-05%20Datasheet.pdf

[3] Bluetooth 5.2 Wireless Module Datasheet: https://www.murata.com/products/productdata/8819747618846/TYPE2EG.pdf

[4] "Bluetooth Essentials for Programmers" by Albert S. Huang and Larry Rudolph

[5] "LoRa and IoT Networks: From Theory to Practice" by Ibrahiem M. M. El Emary and Anna Shakhatreh

[6] "Internet of Things: LoRaWAN and LPWAN Technologies" by Marc Pous and Marco Zennaro

[7] RN2483 LoRa Technology Transceiver Module Datasheet: https://www.mouser.com/datasheet/2/268/50002346B-947485.pdf

[8] LoRa-E5 Module Datasheet: https://www.mouser.com/datasheet/2/268/50002346B-947485.pdf

[9] SX1278 LoRa Transceiver Datasheet: https://www.semtech.com/products/wireless-rf/lora-connect/sx1278

[10] Button Datasheet: https://wiki-content.arduino.cc/documents/datasheets/Button.pdf

PHỤ LỤC

- **CODE TOÀN BỘ HỆ THỐNG:**

  https://github.com/hieuchuai123/DO_AN_2_CODE

- **VIDEO CHẠY THỬ SẢN PHẨM:**

  https://www.youtube.com/watch?v=Dkg3njfxeOs

  https://www.youtube.com/watch?v=p1s3aFyQwP4&t=4s