

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**  
**KHOA ĐIỆN – ĐIỆN TỬ**  
**BỘ MÔN KỸ THUẬT ĐIỆN TỬ**



**ĐỒ ÁN TỐT NGHIỆP**

**Thiết kế, chế tạo hệ thống giám sát nhịp  
tim và nồng độ Oxy trong máu sử dụng vi  
điều khiển ESP32**

**Giảng viên hướng dẫn : ThS. Vũ Ngọc Quý**  
**Sinh viên thực hiện : Nguyễn Quang Minh**  
**MSSV : 201404024**  
**Lớp : Điện tử và tin học công nghiệp 1**  
**Khoá : 61**

**Hà Nội, tháng 1 năm 2025**

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**  
**KHOA ĐIỆN – ĐIỆN TỬ**  
**BỘ MÔN KỸ THUẬT ĐIỆN TỬ**



**ĐỒ ÁN TỐT NGHIỆP**

**Thiết kế, chế tạo hệ thống giám sát nhịp  
tim và nồng độ Oxy trong máu sử dụng vi  
điều khiển ESP32**

**Giảng viên hướng dẫn : ThS. Vũ Ngọc Quý**  
**Sinh viên thực hiện : Nguyễn Quang Minh**  
**MSSV : 201404024**  
**Lớp : Điện tử và tin học công nghiệp 1**  
**Khoá : 61**

**Hà Nội, tháng 1 năm 2025**

**BẢNG DUYỆT ĐỒ ÁN VÀ XÁC NHẬN  
CHO SINH VIÊN NỘP ĐỒ ÁN TỐT NGHIỆP**

**Họ và tên sinh viên:** Nguyễn Quang Minh      **Mã SV:** 201404024

**Tên đề tài:** Thiết kế, chế tạo hệ thống giám sát nhịp tim và nồng độ Oxy trong máu  
sử dụng vi điều khiển ESP32

STT	Hạng mục	Nội dung	Đánh giá	
			ĐẠT	CHƯA ĐẠT
1	Bố cục đồ án	Bố trí trang bìa, trang lót, đúng mẫu quy định		
		Có lời mở đầu, lời cảm ơn		
		Mục lục chính xác, đúng mẫu		
		Danh mục bảng biểu, hình vẽ đầy đủ, chính xác		
		Danh mục cụm từ viết tắt đầy đủ, chính xác		
		Bố cục các chương đúng mẫu, có kết luận của mỗi chương		
		Phụ lục trình bày hợp lý (nếu có)		
		Danh mục tài liệu tham khảo đủ		
2	Nội dung đồ án	Nội dung của đồ án đã được GVHD duyệt, và kết luận: <input type="checkbox"/> Đồng ý cho SV nộp đồ án cho Bộ môn để bảo vệ <input type="checkbox"/> Không đồng ý cho SV nộp đồ án.		
3	Điểm hướng dẫn đồ án tốt nghiệp:			

*Hà Nội, ngày 14 tháng 1 năm 2025*  
**Giáo viên hướng dẫn**  
(Ký và ghi rõ họ tên)

## LỜI MỞ ĐẦU

Trong thời đại công nghệ ngày nay, việc truyền nhận thông tin, giao tiếp giữa các thiết bị điện tử ngày càng phổ biến và chiếm một ưu tiên lớn để phát triển. Cách đây một vài năm mọi người đã nói về Internet of Things sẽ thay đổi thế giới như thế nào. Nhưng tầm nhìn về việc kết nối hàng tỷ thiết bị có những thử thách nhất định đặc biệt là ở phương thức truyền dẫn.

Hiện nay, công nghệ kỹ thuật ngày càng phát triển. Các máy móc được tự động hóa đáp ứng nhu cầu con người đem lại hiệu quả cao trong nhiều lĩnh vực như công nghiệp, nông nghiệp, dịch vụ,... Các bộ vi điều khiển có khả năng xử lý nhiều hoạt động phức tạp mà chỉ cần một chip vi mạch nhỏ, nó đã dần thay thế các tủ điều khiển lớn và phức tạp bằng những mạch điện gọn nhẹ, dễ dàng thao tác sử dụng.

Vi điều khiển không những góp phần vào kỹ thuật điều khiển mà còn góp phần to lớn vào việc phát triển thông tin. Chính vì các lý do trên, việc tìm hiểu, khảo sát vi điều khiển là điều mà các sinh viên chuyên ngành kỹ thuật điện tử phải hết sức quan tâm. Đó chính là một nhu cầu cần thiết và cấp bách đối với mỗi sinh viên, đề tài này được thực hiện chính là đáp ứng nhu cầu đó.

Để góp phần đáp ứng nhu cầu trên và đóng góp thêm giải pháp thay thế các tủ điều khiển lớn và phức tạp, sau một thời gian dưới sự giảng dạy của các thầy cô trường Đại học Giao Thông Vận Tải và các bạn cùng khoa, em đã thiết kế, chế tạo "hệ thống giám sát nhịp tim và nồng độ Oxy trong máu sử dụng vi điều khiển ESP32".

Hệ thống được xây dựng dựa trên phần cứng và phần mềm, 2 phần này được xây dựng riêng biệt nhưng khi kết nối lại, hỗ trợ lẫn nhau tạo nên hệ thống hoàn chỉnh. Trong suốt quá trình thực hiện đồ án em thực hiện chế tạo và thử nghiệm thành công hệ thống giám sát nhịp tim và nồng độ Oxy trong máu sử dụng vi điều khiển ESP32 có thể đọc giá trị từ cảm biến hiển thị lên màn hình của thiết bị, sau đó sẽ gửi giá trị nhận được lên App Mobile thông qua WiFi, server MQTT, xây dựng hệ thống theo dõi lịch sử đo của từng thiết bị trên App Mobile.

Do thời gian, kiến thức và kinh nghiệm của em còn có hạn nên sẽ không thể tránh khỏi những sai sót. Em rất mong được sự giúp đỡ và tham khảo ý kiến của thầy cô và các bạn nhằm đóng góp phát triển thêm đề tài.

## LỜI CẢM ƠN

Lời đầu tiên em xin chân thành cảm ơn thầy Vũ Ngọc Quý cùng tất cả các thầy cô khoa Điện – Điện Tử đã truyền đạt những kiến thức nền tảng, hữu ích cho chúng em, tận tình hướng dẫn và giúp đỡ cho chúng em trong suốt quá trình 4.5 năm trên giảng đường Đại học và hoàn thành đồ án tốt nghiệp.

Do kiến thức còn hạn hẹp nên trong quá trình thực hiện đề tài em không thể tránh khỏi những sai sót kính mong quý thầy cô trong hội đồng chỉ dẫn và giúp đỡ. Em xin gửi lời cảm ơn đến thầy Vũ Ngọc Quý đã trực tiếp hướng dẫn, góp ý, chia sẻ nhiều kinh nghiệm quý báu, tận tình giúp đỡ và tạo điều kiện để em hoàn thành tốt đề tài.

Cảm ơn sự giúp đỡ của quý thầy cô trong khoa Điện-Điện Tử đã tạo điều kiện cho chúng em về thời gian, môi trường cũng như kiến thức để hoàn thành đề tài. Ngoài ra, em xin cảm ơn đến bạn Trần Minh Đức là một sinh viên Đại học Bách Khoa Hà Nội đã giúp đỡ em trong việc thiết kế vỏ sản phẩm. Em chắc rằng quyển báo cáo luận văn tốt nghiệp này không tránh khỏi những thiếu sót cũng như những sai sót trong suốt thời gian hoàn thành đề tài này, em rất cảm ơn nếu nhận được những ý kiến đóng góp của khoa, giảng viên và cũng tất cả các bạn đọc để em có thể hoàn thiện hơn thực hiện tốt hơn trong công việc sau này.

Cảm ơn toàn thể quý thầy cô, cán bộ công nhân viên chức trường Đại học Giao thông Vận tải đã tạo điều kiện tốt nhất để chúng em tiếp thu những kiến thức quý báu và thực hiện đề tài này. Em cũng xin gửi lời cảm ơn đến các bạn lớp Kỹ Thuật Điện Tử và Tin Học Công Nghiệp khóa K61 đã trao đổi chia sẻ kiến thức cũng như những kinh nghiệm quý báu trong thời gian thực hiện đề tài.

Em xin chân thành cảm ơn!

## MỤC LỤC

<b>LỜI MỞ ĐẦU .....</b>	<b>ii</b>
<b>LỜI CẢM ƠN .....</b>	<b>iii</b>
<b>MỤC LỤC .....</b>	<b>iv</b>
<b>DANH MỤC CÁC BẢNG BIỂU .....</b>	<b>vi</b>
<b>DANH MỤC CÁC HÌNH VẼ.....</b>	<b>vi</b>
<b>DANH MỤC CÁC CỤM TỪ VIẾT TẮT .....</b>	<b>viii</b>
<b>CHƯƠNG 1: TỔNG QUAN VỀ IoT VÀ MỤC TIÊU CỦA ĐỒ ÁN .....</b>	<b>1</b>
1.1. Tổng quan về đề tài.....	1
1.2. Giới thiệu một số sản phẩm trên thị trường .....	2
1.3. Mục tiêu đề tài .....	3
1.4. Nội dung thực hiện .....	3
1.5. Giới hạn đề tài.....	4
1.6. Kết luận chương I .....	4
<b>CHƯƠNG 2: CƠ SỞ LÝ THUYẾT .....</b>	<b>5</b>
2.1. Thiết kế tổng quan .....	5
2.1.1. Sơ đồ khối phần cứng của thiết bị .....	5
2.1.2. Sơ đồ thuật toán của thiết bị .....	6
2.1.3. Quy trình hoạt động của hệ thống theo dõi nhịp tim và nồng độ Oxy trong máu. ....	6
2.2. Phần cứng sử dụng trong hệ thống .....	6
2.2.1. Khối vi điều khiển Module WiFi ESP32 .....	6
2.2.2. Khối cảm biến(MAX30102) .....	12
2.2.3. Khối hiển thị - màn hình Oled.....	14
2.2.4. Khối nguồn .....	15
2.2.5. Khối Mobile App.....	16
2.3. Giới thiệu các phần mềm sử dụng trong hệ thống.....	18
2.3.1. Giới thiệu về Visual Studio Code.....	18
2.3.2. Giới thiệu về Android Studio .....	19
2.3.3. Giới thiệu về Arduino IDE .....	20
2.4 Kết luận chương 2.....	20
<b>CHƯƠNG 3: THIẾT KẾ HỆ THỐNG GIÁM SÁT NHỊP TIM VÀ NỒNG ĐỘ OXY TRONG MÁU.....</b>	<b>21</b>
3.1. Thiết kế phần cứng .....	21
3.1.1. Thiết kế sơ đồ đấu nối của hệ thống.....	21

3.1.2. Sơ đồ nguyên lý của hệ thống .....	22
3.1.3. Thiết kế sơ đồ mạch PCB của hệ thống .....	22
3.2. Thi công lắp đặt hệ thống .....	24
3.3. Sản phẩm sau khi hoàn thiện .....	25
3.4 Thiết kế phần mềm .....	27
3.4.1. Lưu đồ thuật toán của Web đăng nhập WiFi cho thiết bị .....	27
3.4.2. Thiết kế giao diện Web đăng nhập WiFi.....	28
3.4.3. Thiết kế lưu đồ thuật toán của APP.....	30
3.4.4. Thiết kế giao diện App .....	31
3.5. Kết luận chương 3.....	33
<b>CHƯƠNG 4: THỬ NGHIỆM VÀ ĐÁNH GIÁ HỆ THỐNG .....</b>	<b>34</b>
4.1. Thử nghiệm hệ thống.....	34
4.1.1. Đi vào thử nghiệm .....	34
4.1.2. So sánh với sản phẩm hiện có trên thị trường .....	37
4.1.3. Hoạt động với nhiều thiết bị để tạo thành hệ thống .....	38
4.1.4. Thử nghiệm trên người với lúc hoạt động bình thường và lúc hoạt động mạnh .....	39
4.1.5. Thử nghiệm trên nhiều độ tuổi khác nhau.....	39
4.2. Đánh giá kết quả nhận được .....	40
4.3. Kết luận chương 4.....	41
<b>CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN ĐỀ TÀI.....</b>	<b>42</b>
5.1 Kết quả đạt được .....	42
5.2. Hạn chế .....	42
5.3 Định hướng phát triển.....	42
<b>PHỤ LỤC.....</b>	<b>43</b>
PHỤ LỤC 1: Code điều khiển hệ thống .....	43
PHỤ LỤC 2: Code App Giao diện tổng quan .....	54
<b>Tài liệu tham khảo .....</b>	<b>59</b>

## DANH MỤC CÁC BẢNG BIỂU

Bảng 2. 1 Định nghĩa các chân module ESP32-WROOM-32.....	9
Bảng 4. 1. Bảng so sánh nhịp tim giữa các độ tuổi.....	40

## DANH MỤC CÁC HÌNH VẼ

Hình 1. 1. Một số thiết bị đeo tay theo dõi sức khỏe trên thị trường .....	2
Hình 1. 2. Một số thiết bị đo nhịp tim và nồng độ Oxy trong máu đang có trên thị trường .....	2
Hình 2. 1. Sơ đồ khối phần cứng của thiết bị.....	5
Hình 2. 2. Sơ đồ thuật toán của thiết bị.....	6
Hình 2. 3. Sơ đồ hoạt động của hệ thống .....	6
Hình 2. 4. Module ESP32-WROOM-32 .....	7
Hình 2. 5. Kiến trúc của ESP32 .....	7
Hình 2. 6. Sơ đồ chức năng các chân của ESP32 -WROOM-32 .....	8
Hình 2. 7. Sơ đồ bố trí chân của module ESP32-WROOM-32 .....	9
Hình 2. 8. Cấu trúc và địa chỉ bộ nhớ của ESP32 .....	10
Hình 2. 9. Sơ đồ nguyên lý module ESP32-WROOM-32.....	11
Hình 2. 10. Sơ đồ nguyên lý thiết bị ngoại vi module ESP32-WROOM-32 .....	11
Hình 2. 11. Module Max30102 .....	12
Hình 2. 12. Sơ đồ chân Module Max30102 .....	13
Hình 2. 13. Mô tả kết nối I2C 1M-1S .....	13
Hình 2. 14. Mô tả đường truyền dữ liệu của giao tiếp I2C .....	14
Hình 2. 15. Màn hình Oled 0.96inch giao tiếp I2C.....	15
Hình 2. 16. Mạch nguồn UPS 5V kết hợp khay pin .....	15
Hình 2. 17. Pin 18650 .....	16
Hình 2. 18. Tổng quan về Mobile App .....	17
Hình 2. 19. Mô tả cách hoạt động của MQTT .....	17
Hình 2. 20. Phần mềm Visual Studio Code .....	18
Hình 2. 21. Cấu trúc của file và thành phần Project .....	19
Hình 2. 22. Phần mềm lập trình Arduino IDE .....	20
Hình 3. 1. Sơ đồ đấu nối của thiết bị.....	21
Hình 3. 2. Sơ đồ nguyên lý của thiết bị.....	22
Hình 3. 3. Sơ đồ mạch PCB .....	22
Hình 3. 4. 3D mặt trước của PCB .....	23
Hình 3. 5. 3D mặt sau của PCB .....	23
Hình 3. 6. Mặt trước của mạch sau khi gia công .....	24
Hình 3. 7. Mặt sau của mạch sau khi gia công.....	24
Hình 3. 8. Mạch sau được hàn linh kiện .....	25
Hình 3. 9. Mặt trước của sản phẩm khi hoàn thiện .....	25
Hình 3. 10. Mặt sau của sản phẩm sau khi hoàn thiện.....	26
Hình 3. 11. Mặt bên của sản phẩm sau khi hoàn thiện .....	26
Hình 3. 12. Phần chân cắm sạc của sản phẩm.....	26
Hình 3. 13. Lưu đồ thuật toán của Web đăng nhập WiFi cho thiết bị .....	27
Hình 3. 14. Giao diện tổng quát Web đăng nhập WiFi.....	28



Hình 3. 15. Giao diện Configure WiFi.....	29
Hình 3. 16. Giao diện Info .....	29
Hình 3. 17. Giao diện UPDATE .....	30
Hình 3. 18. Lưu đồ thuật toán của App.....	30
Hình 3. 19. Giao diện theo dõi tổng quát tất cả các thiết bị hiện có .....	31
Hình 3. 20. Giao diện theo dõi chi tiết của từng thiết bị .....	32
Hình 3. 21: Xóa 1 dữ liệu lịch sử đo .....	33
Hình 3. 22: Reset toàn bộ thông tin .....	33
Hình 4. 1. Thiết bị sau khi được cấp nguồn .....	34
Hình 4. 2. Đăng nhập WiFi cho thiết bị .....	35
Hình 4. 3. Dữ liệu đo hiển thị trên màn hình Oled 0.96 inch.....	36
Hình 4. 4. Dữ liệu đo hiển thị trên Mobile App.....	36
Hình 4. 5. Lưu và xóa lịch sử trên App.....	37
Hình 4. 6. So với sản phẩm trên thị trường.....	37
Hình 4. 7. Hoạt động với nhiều thiết bị để tạo thành hệ thống .....	38
Hình 4. 8. So sánh nhịp tim giữa lúc hoạt động bình thường và hoạt động mạnh....	39
Hình 4. 9. So sánh nhịp tim giữa các độ tuổi .....	40

## DANH MỤC CÁC CỤM TỪ VIẾT TẮT

STT	Từ viết tắt	Từ tiếng Anh	Nghĩa tiếng Việt
1	AI	Artificial intelligence	Trí tuệ nhân tạo
2	CPU	Central Processing Unit	Bộ xử lý trung tâm
3	IoT	Internet of Things	Vạn vật kết nối
4	UID	Unique Identifier	Mã định danh duy nhất
5	UART	Universal Asynchronous reveiver - transmitter	Truyền thông nối tiếp không đồng bộ
6	I2C	Inter - Intergrated Circuit	Chuẩn giao tiếp I2C
7	SCK/SCL	Serial Clock	Chân giữ xung nhịp trong giao tiếp I2C
8	SDA	Serial Data Line	Dây truyền dữ liệu
9	LCD	Liquid Crystal Display	Màn hình tinh thể lỏng
10	OLED	Organic Light Emitting Diodes	Màn hình OLED
11	VDC	Volt Direct Current	Dòng điện một chiều
12	SQL	Structured Query Language	Hệ cơ sở dữ liệu
13	GPIO	General Purpose Input / Output	Đầu vào/đầu ra
14	GND	Ground	Đất
15	Framework	Framework	Khung phần mềm
16	APP	Application	Ứng dụng
17	Mobile	Mobile	Thiết bị di động
18	MQTT	Message Queuing Telemetry Transport	Giao thức truyền thông điện
19	Doul core	Doul core	Lõi kép
20	Wifi	Wireless Fidelity	Sóng vô tuyến để truyền tín hiệu

# CHƯƠNG 1: TỔNG QUAN VỀ IoT VÀ MỤC TIÊU CỦA ĐỒ ÁN

## 1.1. Tổng quan về đề tài

Công nghệ Internet of Things (IOT) nói chung và công nghệ cảm biến không dây (Wireless Sensor) nói riêng được tích hợp từ các kỹ thuật điện tử, tin học và viễn thông tiên tiến vào trong mục đích nghiên cứu, giải trí, sản xuất, kinh doanh, v.v..., phạm vi này ngày càng được mở rộng, để tạo ra các ứng dụng đáp ứng cho các nhu cầu trên các lĩnh vực khác nhau.

Thuật ngữ IoT hay Internet vạn vật đề cập đến mạng lưới tập hợp các thiết bị thông minh và công nghệ tạo điều kiện thuận lợi cho hoạt động giao tiếp giữa thiết bị và đám mây cũng như giữa các thiết bị với nhau. Nhờ sự ra đời của chip máy tính giá rẻ và công nghệ viễn thông băng thông cao, ngày nay, chúng ta có hàng tỷ thiết bị được kết nối với internet. Điều này nghĩa là các thiết bị hàng ngày như bàn chải đánh răng, máy hút bụi, ô tô và máy móc có thể sử dụng cảm biến để thu thập dữ liệu và phản hồi lại người dùng một cách thông minh.

Hiện nay, mặc dù khái niệm IOT và công nghệ cảm biến không dây đã trở nên khá quen thuộc và được ứng dụng nhiều trong các lĩnh vực của đời sống con người, đặc biệt ở các nước phát triển có nền khoa học công nghệ tiên tiến. Tuy nhiên, những công nghệ này chưa được áp dụng một cách rộng rãi ở nước ta, do những điều kiện về kỹ thuật, kinh tế, nhu cầu sử dụng. Song nó vẫn hứa hẹn là một đích đến tiêu biểu cho các nhà nghiên cứu, cho những mục đích phát triển đầy tiềm năng.

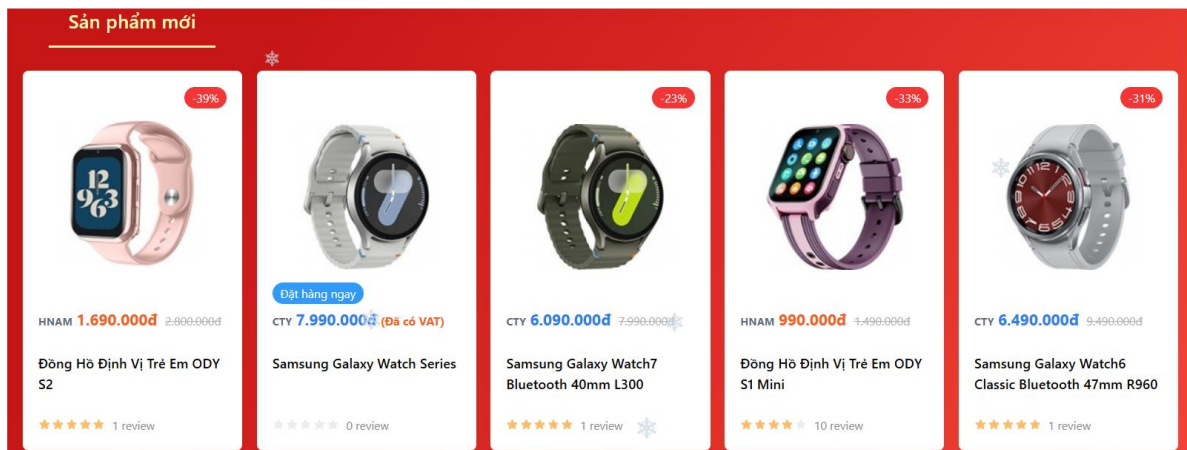
Nhu cầu về việc chăm sóc sức khỏe ngày càng được chú trọng vì thế mà các thiết bị chăm sóc sức khỏe ngày càng được phát triển trong số đó có thiết bị theo dõi nhịp tim và nồng độ Oxy trong máu. Trên thị trường hiện nay giá của các thiết bị chăm sóc sức khỏe không hề rẻ và thường phải nhập khẩu từ nước ngoài trong đó có máy đo nhịp tim và nồng độ Oxy trong máu.

Trên cơ sở tìm hiểu về IoT nhằm phát triển hệ thống giám sát nhịp tim và nồng độ Oxy trong máu để có thể ứng dụng IoT trong y tế. Hệ thống sử dụng cảm biến đọc giá trị nhịp tim và nồng độ Oxy trong máu hiển thị lên màn hình của thiết bị và gửi giá trị lên server MQTT thông qua kết nối WiFi sau đó sử dụng App Mobile để lấy giá trị trên server MQTT hiển thị lên màn hình điện thoại. Qua đó, giải quyết được vấn đề theo dõi sức khỏe từ xa cho bệnh nhân.

## 1.2. Giới thiệu một số sản phẩm trên thị trường

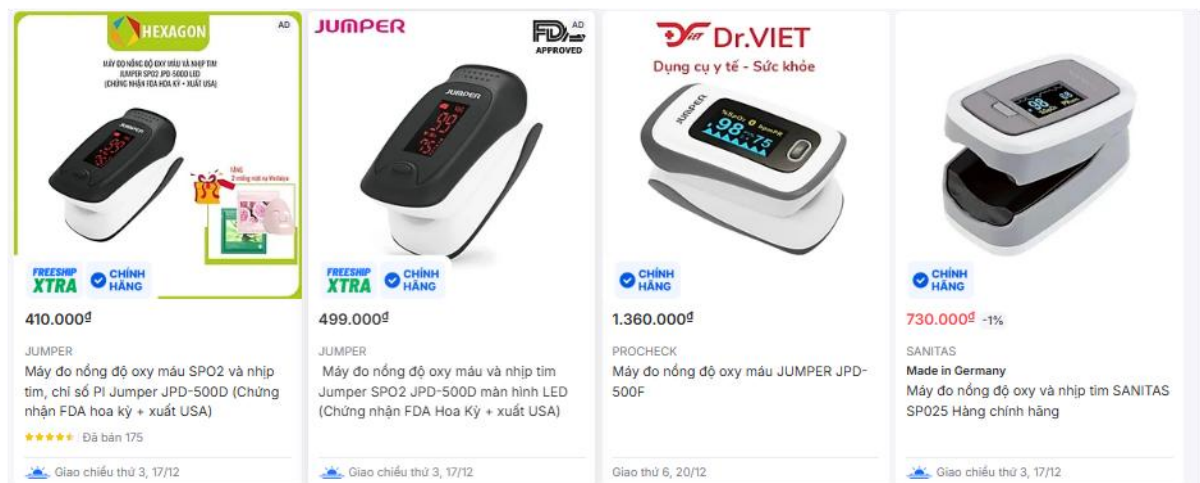
Hiện nay, do sự phát triển của khoa học công nghệ nhu cầu chăm sóc sức khỏe con người được chú trọng nhiều hơn. Nhằm đáp ứng được nhu cầu thực tế thì trên thị trường hiện nay đã có rất nhiều sản phẩm theo dõi sức khỏe như máy đo huyết áp, thiết bị theo dõi giấc ngủ, máy đo nhịp tim và nồng độ Oxy trong máu. Đa số các thiết bị này đã được tích hợp vào trong đồng hồ thông minh và cũng sử dụng công nghệ IoT để truyền dữ liệu.

Tuy nhiên giá thành của 1 chiếc đồng hồ thông minh không hề rẻ do vậy khá khó để ai cũng có điều kiện mua để sử dụng.



Hình 1. 1. Một số thiết bị đeo tay theo dõi sức khỏe trên thị trường

Ngoài ra, trên thị trường cũng có nhiều sản phẩm có chức năng đo nhịp tim và nồng độ Oxy trong máu nhưng giá thành cũng không hề rẻ và những sản phẩm này vẫn đơn thuần là thiết bị đo bình thường không phải là thiết bị IoT.



Hình 1. 2. Một số thiết bị đo nhịp tim và nồng độ Oxy trong máu đang có trên thị trường

### 1.3. Mục tiêu đề tài

- Tiếp cận công nghệ theo dõi sức khỏe và hệ thống theo dõi sức khỏe sử dụng các cảm biến và ngoại vi giao tiếp của vi điều khiển.
- Xây dựng hệ thống App theo dõi nhịp tim và nồng độ Oxy trong máu từ xa.
- Mạch được thực hiện nhỏ gọn dễ dùng với việc sử dụng Esp32 kết hợp với các module nguồn, cảm biến, hiển thị nhằm đáp ứng cho người sử dụng sự tiện lợi, dễ mang, dễ dùng.
- Xây dựng hệ thống lưu trữ dữ liệu trực tiếp của bộ nhớ điện thoại.
- Chế tạo thành công hệ thống theo dõi nhịp tim và nồng độ Oxy trong máu.
- Hiện nay, nhu cầu về việc chăm sóc sức khỏe ngày càng được chú trọng vì thế mà các thiết bị chăm sóc sức khỏe ngày càng được phát triển trong số đó có thiết bị theo dõi nhịp tim và nồng độ Oxy trong máu. Trên thị trường hiện nay giá của các thiết bị chăm sóc sức khỏe không hề rẻ và thường phải nhập khẩu từ nước ngoài trong đó có máy đo nhịp tim và nồng độ Oxy trong máu.
- Mở rộng nghiên cứu đề tài, thử nghiệm các công nghệ mới hơn trên ý tưởng đã có.

### 1.4. Nội dung thực hiện

- Thiết kế, thi công hệ thống, phần cứng, phần điện (cơ khí, linh kiện).
- Lắp ráp các khối điều khiển vào hệ thống.
- Lập trình đọc dữ liệu với Module Wifi ESP32, Module cảm biến nhịp tim MAX30102.
- Thiết kế App theo dõi từ xa thông qua WiFi với giao thức truyền thông MQTT.
- Giao diện App theo dõi từ xa bao gồm:
  - Giao diện quản trị:
    - Cho phép đổi tên người bệnh ứng với mỗi thiết bị
    - Biểu đồ theo dõi nhịp tim và nồng độ Oxy trong máu và hiển thị giá trị lên màn hình.
  - Giao diện nhật ký:
    - Cho phép lưu trữ lịch sử đo lên Listview, đồng thời cũng cho phép xóa nếu không cần đến dữ liệu cần thiết.
    - Cho phép theo dõi nhiều thiết bị một lúc với cùng một App.
- Chạy thử nghiệm hệ thống.
- Đánh giá kết quả thực hiện mô hình.

### 1.5. Giới hạn đề tài.

- Tổng quan về hệ thống theo dõi nhịp tim và nồng độ Oxy trong máu.
- Thiết kế và chế tạo hệ thống theo dõi nhịp tim và nồng độ Oxy trong máu.
- Sử dụng App Mobile để dễ dàng thiết lập và quản lý từ xa.

### 1.6. Kết luận chương I

Công nghệ IoT đã tạo ra những tiến bộ đáng kể trong cuộc sống hằng ngày và các potential thay đổi cách chúng ta quản lý và tương tác với thế giới xung quanh. Với sự phát triển đa dạng và giảm chi phí liên tục. Công nghệ IoT đang trở thành một giải pháp ngày càng được chú ý và được tích hợp rộng rãi trong nhiều lĩnh vực của đời sống đặc biệt là trong phát triển hệ thống y tế.

Phạm vi ứng dụng của công nghệ IoT là rất lớn, và sự kết hợp với các công nghệ khác làm tăng tính linh hoạt và khả năng tương tác. Sự tích hợp này không chỉ giúp tối ưu hóa quy trình quản lý mà còn mang lại những cơ hội sáng tạo mới. Điều này làm cho IoT trở thành một lựa chọn hấp dẫn cho nhiều ứng dụng hiện tại và có thể mở ra những triển vọng hứa hẹn trong tương lai.

Thiết bị hướng tới sự nhỏ gọn, giá thành rẻ hơn trên thị trường, giá thành và chi phí tự sản xuất hiện tại chỉ khoảng 300-400 nghìn đồng rẻ hơn khá nhiều so với thị trường và còn có thể theo dõi từ xa cho bác sĩ hoặc người nhà bệnh nhân theo dõi sức khỏe người bệnh.

Ngoài ra, dự án này còn có mục tiêu: Nâng cao kiến thức về các linh kiện điện tử và lập trình vi điều khiển, rèn luyện kỹ năng thiết kế, chế tạo và thử nghiệm thiết bị điện tử, phát triển khả năng sáng tạo và giải quyết vấn đề. Thiết bị có thể hoạt động độc lập hoặc kết nối với mạng Wi-Fi để gửi dữ liệu đến điện thoại thông minh của người dùng.

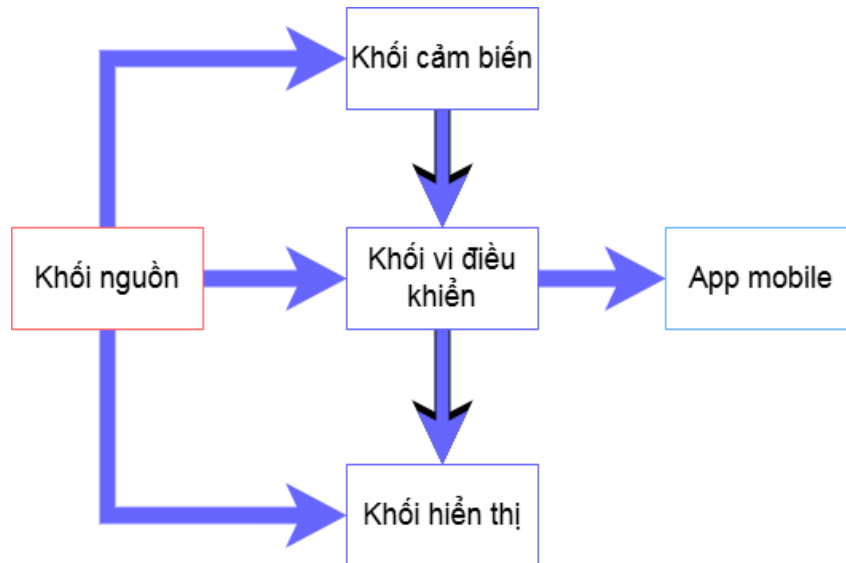
Tổng kết lại chương I em đã thực hiện tìm hiểu về sản phẩm theo dõi nhịp tim và nồng độ Oxy trong máu nêu ra được cấu trúc thực hiện của dự án . So sánh với các sản phẩm đang có và tìm hiểu được các báo cáo đề án nghiên cứu đã công bố, từ đó nêu ra được mục tiêu của đề án này. Dự án này thiết kế và chế tạo một thiết bị theo dõi nhịp tim và nồng độ Oxy trong máu sử dụng các linh kiện ESP32, màn hình OLED, cảm biến nhịp tim Max30102. Dự án thiết bị này có ý nghĩa thực tiễn cao, ứng dụng được trong lĩnh vực sức khỏe, y tế. Dự án giúp nâng cao kiến thức và kỹ năng cho người thực hiện, đồng thời thúc đẩy khả năng sáng tạo và giải quyết vấn đề.

## CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

### 2.1. Thiết kế tổng quan

Mục tiêu của đề án này là thiết kế và phát triển được mô hình “Hệ thống giám sát nhịp tim và nồng độ Oxy trong máu sử dụng ESP32 và cảm biến Max30102”.

#### 2.1.1. Sơ đồ khối phần cứng của thiết bị

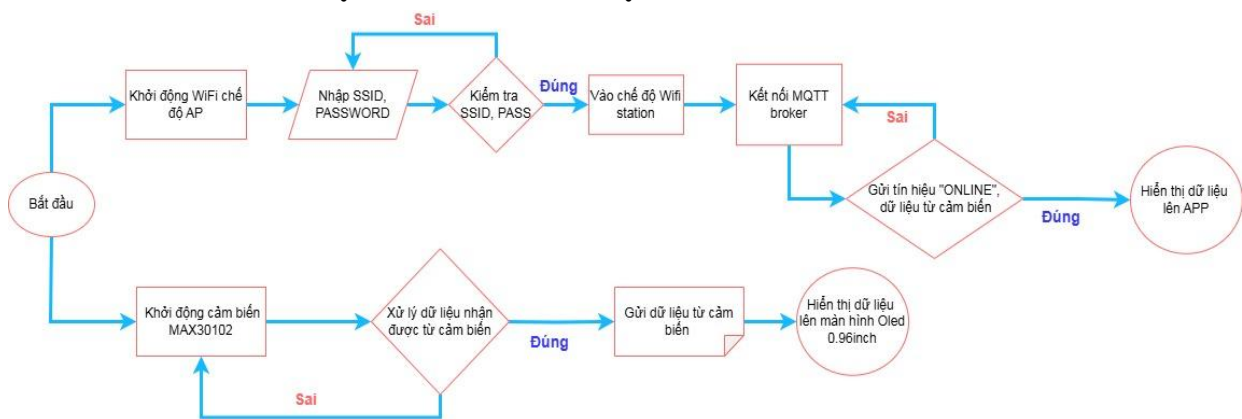


Hình 2. 1. Sơ đồ khối phần cứng của thiết bị

Trong đề tài này, em đã thiết kế hệ thống giám sát nhịp tim và nồng độ Oxy trong máu sử dụng các khối chức năng sau:

- **Khối nguồn(nguồn pin 18650):** Sử dụng module ổn áp đầu ra 5V – 1A để cung cấp cho vi điều khiển và các ngoại vi. Đảm bảo rằng mọi thiết bị trong hệ thống có nguồn cung cấp đầy đủ để hoạt động.
- **Khối cảm biến(MAX30102):** Sử dụng cảm biến max30102 để đọc giá trị nhận về vi điều khiển.
- **Khối vi điều khiển(ESP32):** Sử dụng vi điều khiển ESP32 để xử lý dữ liệu nhận về cảm biến và đọc lên màn hình hiển thị và gửi dữ liệu lên MQTT sau đó APP sẽ subscribe đến đúng channel để đọc đúng giá trị nhận được về.
- **Khối hiển thị(OLED 0.96):** Sử dụng màn hình OLED 0.96 để hiển thị dữ liệu từ vi điều khiển gửi lên. Giúp người dùng dễ dàng theo dõi nhịp tim và nồng độ Oxy trong máu.
- **App mobile:** Giúp bác sĩ có thể theo dõi nhịp tim và nồng độ Oxy tron máu của bệnh nhân thông qua WiFi và phương thức giao tiếp MQTT.

### 2.1.2. Sơ đồ thuật toán của thiết bị

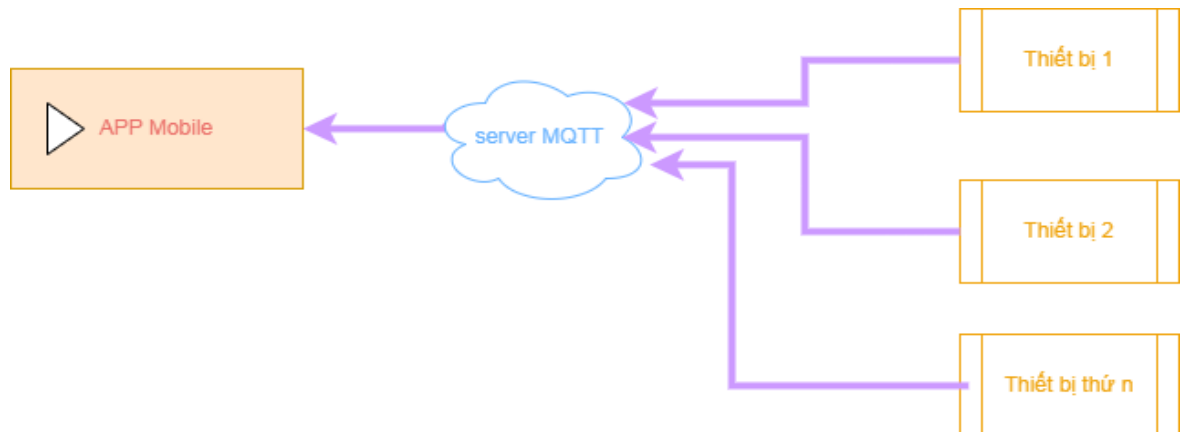


Hình 2. 2. Sơ đồ thuật toán của thiết bị

### 2.1.3. Quy trình hoạt động của hệ thống theo dõi nhịp tim và nồng độ Oxy trong máu.

Khi bệnh nhân thực hiện việc đo nhịp tim và nồng độ Oxy trong máu bằng cách chạm ngón tay vào cảm biến MAX30102, dữ liệu đọc được sẽ trả về ESP32 để xử lý, những dữ liệu này sẽ được ESP32 xử lý và hiển thị lên màn hình OLED 0.96 cùng lúc đó những dữ liệu nhận được này cũng được gửi lên server MQTT thông qua kết nối WiFi.

Thông qua WiFi, các thiết bị sẽ thực hiện gửi dữ liệu lên MQTT để App mobile subscribe đến đúng channel tương ứng của từng thiết bị để theo dõi nhiều thiết bị một lúc từ xa.



Hình 2. 3. Sơ đồ hoạt động của hệ thống

## 2.2. Phần cứng sử dụng trong hệ thống

Xây dựng phần cứng của từng thiết bị dựa trên sơ đồ khối bên trên.

### 2.2.1. Khối vi điều khiển Module WiFi ESP32

Với yêu cầu của đề tài đặt ra là điều khiển được các thiết bị điện, qua quá trình nghiên cứu, tìm hiểu tài liệu em đã chọn giải pháp tối ưu cho khối xử lý trung tâm là ESP32 được phát triển bởi Espressif Systems. Là một vi điều khiển 32bit,



đáp ứng đầy đủ các yêu cầu ban đầu đặt ra, ngoài ra còn có thể mở rộng thêm một số chức năng khác khi cải tiến hệ thống.

### 2.2.1.1. Tổng quan ESP32

ESP32 là một chip được tích hợp công nghệ Wifi và Bluetooth với công nghệ tiêu thụ năng lượng cực thấp. Các dòng chip ESP32 bao gồm ESP32-D0WDQ6, ESP32-D0WD, ESP32-D2WD và ESP32-S0WD.

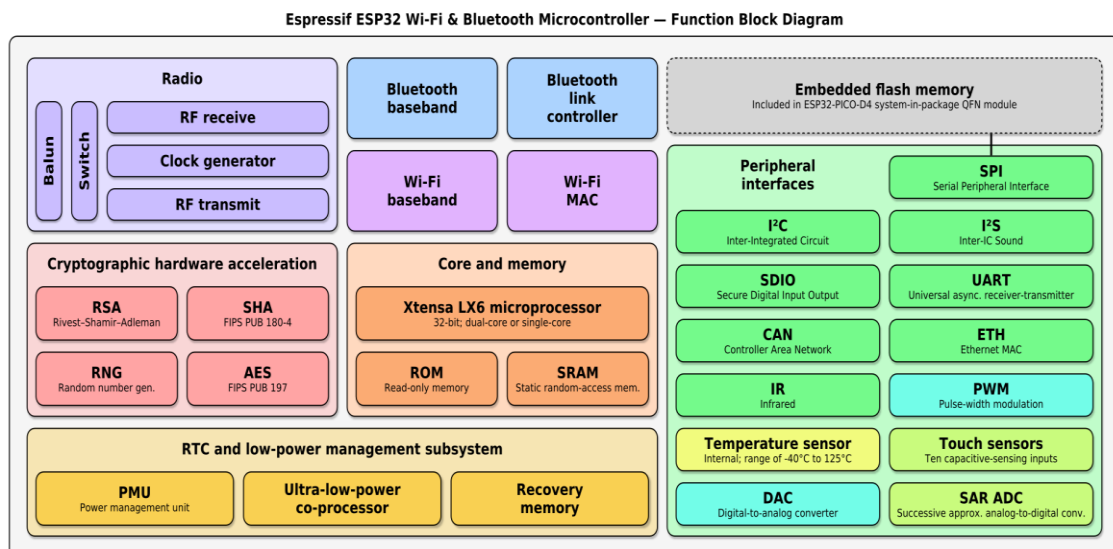
Phiên bản ESP32-WROOM-32 là một module vi điều khiển (MCU) Wifi (Wireless Fidelity) – BT (Bluetooth) – BLE (Bluetooth Low Energy) phổ biến và mạnh mẽ phục vụ cho nhiều ứng dụng khác nhau từ những ứng dụng đơn giản như điều khiển thiết bị, đọc giá trị cảm biến đến những nhiệm vụ phức tạp như mã hóa giọng nói, phát nhạc trực tuyến, giải mã MP3,...



Hình 2. 4. Module ESP32-WROOM-32

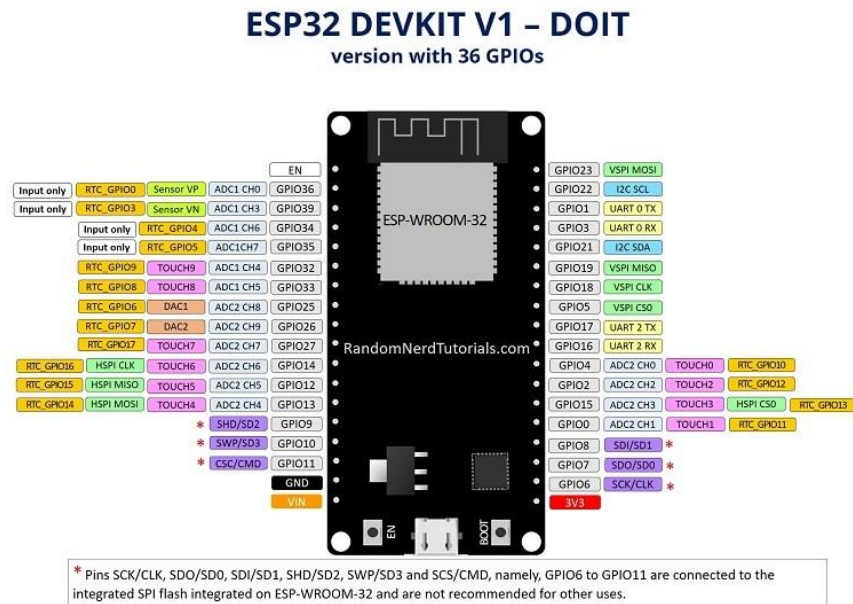
Module này được xây dựng với lõi nhân là chip ESP32-D0WDQ6, chip được thiết kế để có thể mở rộng. Có 2 lõi CPU (Central Processing Unit) có thể kiểm soát riêng biệt và tần số xung đồng hồ dao động từ 80MHz đến 240MHz.

#### • Kiến trúc



Hình 2. 5. Kiến trúc của ESP32

### 2.2.1.2 Sơ đồ chân của module ESP32-WROOM-32

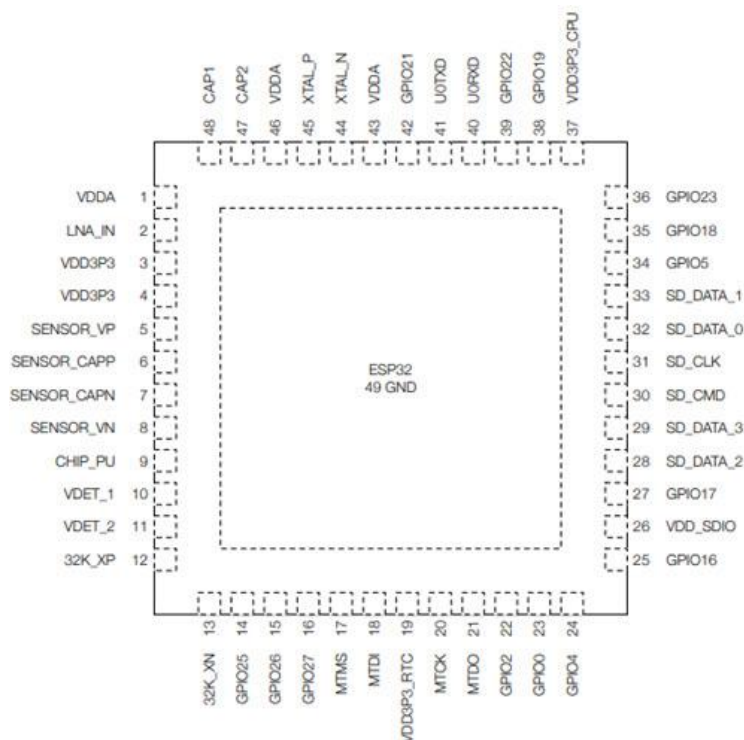


Hình 2. 6. Sơ đồ chức năng các chân của ESP32 -WROOM-32

TÊN	SỐ	LOẠI	CHỨC NĂNG
GND	1	P(Nguồn)	Ground
3.3V	2	P	Chân cấp nguồn cung cấp 3.3V
EN	3	I(input)	Cho phép modul hoạt động ở mức cao
SENSOR_VP	4	I	GPIO36, ADC1_CH0, RTC_GPIO0
SENSOR_VN	5	I	GPIO39, ADC1_CH3, RTC_GPIO3
IO_34	6	I	GPIO34, ADC1_CH6, RTC_GPIO4
IO_35	7	I/O(input, output)	GPIO35, ADC1_CH7, RTC_GPIO5
IO_32	8	I/O	GPIO32, XTAL_32K_P, ADC1_CH4, TOUCH9, RTC_GPIO9
IO_33	9	I/O	GPIO33, XTAL_33K_N, ADC1_CH5, TOUCH8, RTC_GPIO8
IO_25	10	I/O	GPIO25, DAC_1, ADC2_CH8, RTC_GPIO6
IO_26	11	I/O	GPIO26, DAC_2, ADC2_CH9, RTC_GPIO7
IO_27	12	I/O	GPIO27, ADC2_CH7, TOUCH7 RTC_GPIO17
IO_14	13	I/O	GPIO14, ADC2_CH6, TOUCH6, RTC_GPIO16
IO_12	14	I/O	GPIO12, ADC2_CH5, TOUCH5, RTC_GPIO15
GND	15	P	Ground
IO_13	16	I/O	GPIO13, ADC2_CH4, TOUCH4, RTC_GPIO14
SHD/SD2	17	I/O	GPIO9, SD_DATA2, U1RXD
SWP/SD3	18	I/O	GPIO10, SD_DATA3, U1TXD

SCS/CMD	19	I/O	GPIO11, SD_CMD, U1RTS
SCK/CLK	20	I/O	GPIO6, SD_CLK, U1CTS
SDO/SD0	21	I/O	GPIO7, SD_DATA0, U2RTS
SDI/SD1	22	I/O	GPIO8, SD_DATA1, U2CTS
IO_15	23	I/O	GPIO15, ADC2_CH3, TOUCH3, RTC_GPIO13
IO_2	24	I/O	GPIO2, ADC2_CH2, TOUCH2, RTC_GPIO12
IO_0	25	I/O	GPIO0, ADC2_CH1, TOUCH1, RTC_GPIO11, CLK_OUT1
IO_4	26	I/O	GPIO4, ADC2_CH0, TOUCH0, RTC_GPIO10
IO_16	27	I/O	GPIO16, HS1_DATA4, U2RXD
IO_17	28	I/O	GPIO17, HS1_DATA5, U2TXD
IO_5	29	I/O	GPIO5, HS1_DATA6
IO_18	30	I/O	GPIO18, HS1_DATA7
IO_19	31	I/O	GPIO19, U0CTS
NC	32	-	-
IO_21	33	I/O	GPIO21, VSPIHD
RXD0	34	I/O	GPIO3, U0RXD, CLK_OUT2
TXD0	35	I/O	GPIO1, U0TXD, CLK_OUT3
IO_22	36	I/O	GPIO22, VSPIWP, U0RTS
IO_23	37	I/O	GPIO23, VSPID
GND	38	I/O	Ground

Bảng 2. 1 Định nghĩa các chân module ESP32-WOOM-32

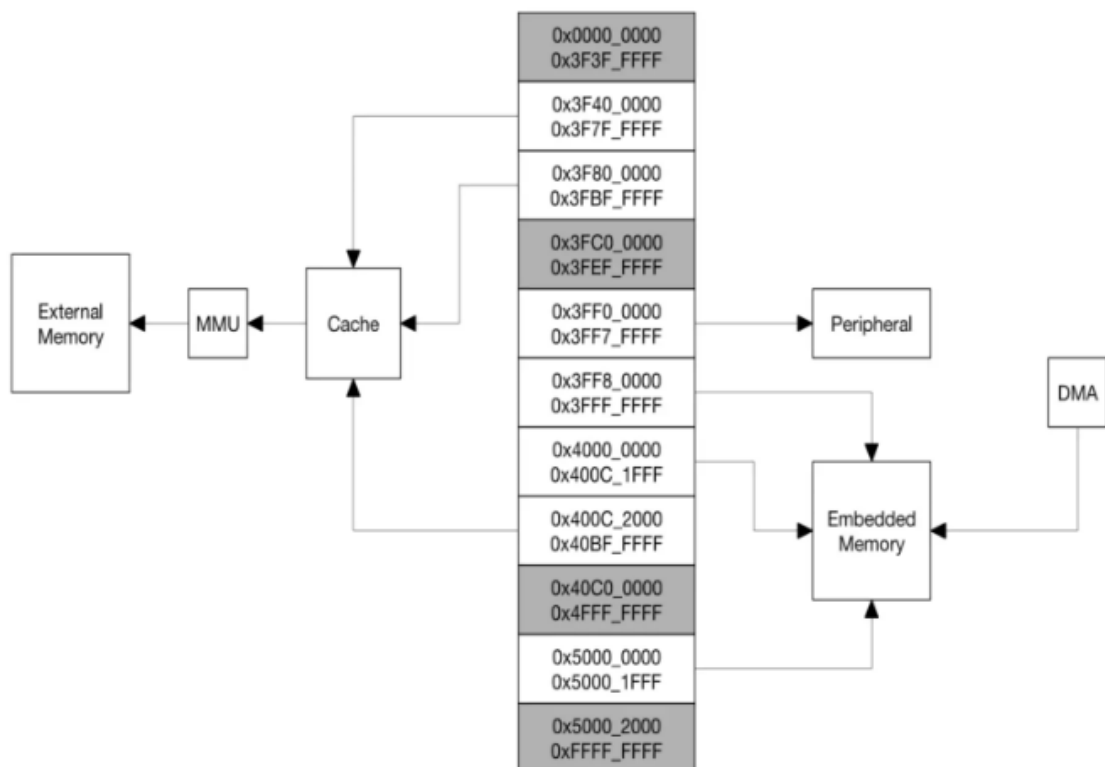


Hình 2. 7. Sơ đồ bố trí chân của module ESP32-WOOM-32

### 2.2.1.3. Chức năng tích hợp trong ESP32

#### a. Chức năng tích hợp trong ESP32

- **CPU:** Chip ESP32 là bộ xử lý lõi kép (Dual core) của vi điều khiển Xtensa® 32bit LX6 với các đặc trưng
  - Hỗ trợ xung nhịp lên đến 240Mhz.
  - Hỗ trợ DSP như bộ nhân 32bit, bộ chia 32bit, MAC 40bit.
  - Hỗ trợ 32 vector interrupt.
- **Bộ nhớ nội (Internal Memory) bao gồm:**
  - 448KB ROM cho việc khởi động và các chức năng cốt lõi.
  - 520KB SRAM trên chip cho dữ liệu.
  - 8KB SRAM trong RTC (Real Time Control): được gọi là bộ nhớ RTC FAST.
  - 8KB SRAM trong RTC: được gọi là bộ nhớ RTC LOW.
  - 1Kbit eFuse: 256bit sử dụng cho hệ thống, 768bit chuyển đổi cho ứng dụng.
- **External Flash (bộ nhớ flash ngoài) và SRAM:** ESP32 hỗ trợ nhiều external QSPI flash và SRAM, có thể truy cập ở tốc độ cao. SRAM được hỗ trợ lên đến 8MB, đọc và ghi 8bit, 16bit, 32bit.
- **Tổ chức bộ nhớ**



Hình 2. 8. Cấu trúc và địa chỉ bộ nhớ của ESP32

## b. WiFi

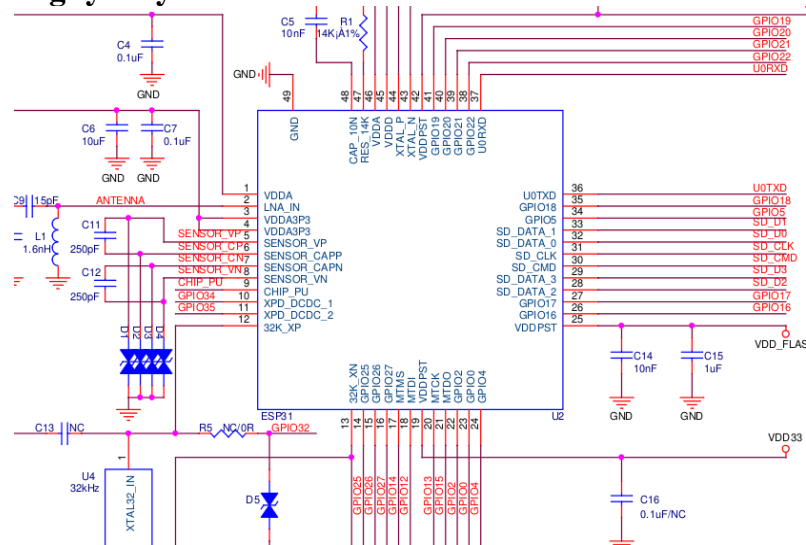
ESP32 thực hiện giao thức TCP/IP và Wifi 802.11 b/g/n, quản lý năng lượng được xử lý để giảm thiểu thời gian thực hiện các tác vụ. Các thư viện của Wifi cung cấp để định cấu hình và giám sát chức năng kết nối mạng ESP32, nó cấu hình cho:

- Chế độ trạm (hay chế độ STA hoặc chế độ Wifi client): ESP32 kết nối với một điểm truy cập.
- Chế độ AP (hay chế độ Soft-AP hoặc chế độ điểm truy cập): các trạm kết nối với ESP32.
- Chế độ kết hợp AP-STA (ESP32 đồng thời là điểm truy cập và là trạm được kết nối với các điểm truy cập khác).
- Các chế độ bảo mật khác nhau cho các chế độ trên (WPA, WPA2, WEP, ...).
- Quét các điểm truy cập (chủ động và thụ động).
- Chế độ giám sát các gói Wifi của tiêu chuẩn IEEE802.11.

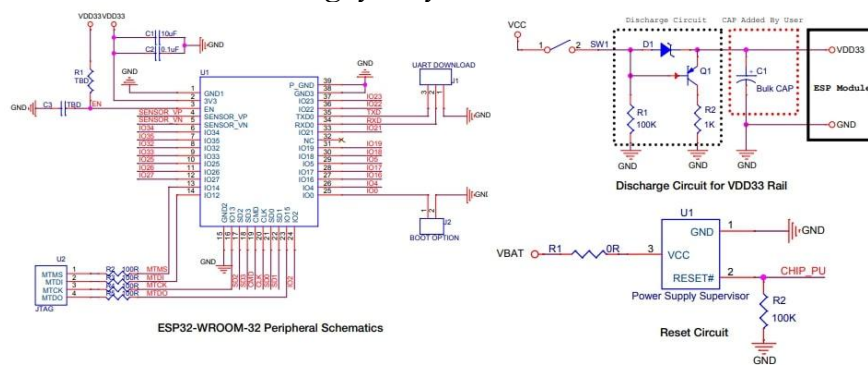
## c. Bluetooth

ESP32 tích hợp bộ điều khiển liên kết Bluetooth, thực hiện các giao thức như điều chế, giải điều chế, xử lý gói, xử lý luồng bit,...

### 2.2.1.4. Sơ đồ nguyên lý module ESP32-WOOM-32



Hình 2. 9. Sơ đồ nguyên lý module ESP32-WOOM-32



Hình 2. 10. Sơ đồ nguyên lý thiết bị ngoại vi module ESP32-WOOM-32

### 2.2.2. Khối cảm biến(MAX30102)

#### 2.2.2.1. Giới thiệu về công nghệ Photoplethysmography(PPG)

Công nghệ **Photoplethysmography** viết tắt là **PPG**. Đây là một kỹ thuật quang học được sử dụng để đo các thay đổi trong thể tích máu bên dưới da.

**Photoplethysmography (PPG)** hoạt động bằng cách sử dụng:

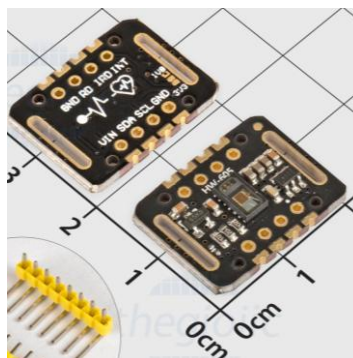
1. **Đèn LED** để phát ra ánh sáng (thường là ánh sáng đỏ và hồng ngoại).
2. **Bộ tách sóng quang** để phát hiện ánh sáng phản xạ hoặc truyền qua mô, từ đó xác định những thay đổi trong lưu lượng máu.
3. **Các thiết bị điện tử xử lý tín hiệu**, giúp lọc nhiễu và đo lường chính xác tín hiệu sinh học.

Công nghệ này thường được tích hợp trong các cảm biến như MAX30102, MAX30100, và các thiết bị đeo thông minh (smartwatches, fitness trackers) để đo nhịp tim và mức oxy trong máu ( $SpO_2$ ).

#### 2.2.2.2. Tổng quan về cảm biến Max30102

Cảm biến nhịp tim và oxy trong máu Max30102 là một mô-đun đo nhịp tim và oxy trong máy tích hợp.

- **Đèn LED bên trong:** Phát ra ánh sáng với bước sóng cụ thể (thường là ánh sáng đỏ và hồng ngoại), giúp thâm nhập vào mô da để đo các tín hiệu sinh học.
- **Bộ tách sóng quang:** Cảm nhận ánh sáng phản xạ hoặc truyền qua mô để phát hiện các thay đổi về cường độ ánh sáng, tương ứng với các thông số sinh học như nhịp tim hoặc mức oxy trong máu.
- **Các bộ phận quang học:** Tối ưu hóa đường truyền ánh sáng và giảm thiểu mất mát ánh sáng, đồng thời cải thiện độ chính xác của việc đo lường.
- **Thiết bị điện tử tiếng ồn thấp:** Tăng cường độ nhạy và độ chính xác bằng cách giảm nhiễu tín hiệu trong quá trình xử lý dữ liệu.
- **Khả năng loại bỏ ánh sáng xung quanh:** Loại bỏ ảnh hưởng của ánh sáng môi trường bên ngoài.



Hình 2. 11. Module Max30102

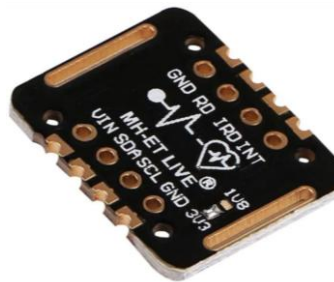


### 2.2.2.3. Thông số kỹ thuật của Max30102

- Bước sóng cực đại LED: 660nm / 880nm
- Điện áp nguồn LED: 3,3 ~ 5VDC
- Loại tín hiệu phát hiện: tín hiệu phản xạ ánh sáng (PPG)
- Giao tiếp: I2C
- Điện áp giao tiếp: 1,8 ~ 3,3V ~ 5V (tùy chọn)

### 2.2.2.4. Sơ đồ chân của Max30102

- VIN: thiết bị đầu cuối nguồn chính 1,8-5V
- Đệm 3 bit: Chọn mức độ kéo lên của bus, tùy thuộc vào điện áp chính của chân, chọn 1.8v hoặc 3.3V
- SCL: kết nối với bus I2C;
- SDA: dữ liệu được kết nối với bus I2C;
- INT: Chân ngắt của chip MAX30102
- RD: thường không được kết nối
- IRD: nối đất IR LED của chip MAX30102 thường không được kết nối
- GND: Dây nối đất

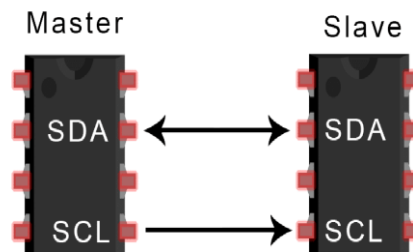


Hình 2. 12. Sơ đồ chân Module Max30102

### 2.2.2.5. Chuẩn giao tiếp I2C

#### a. Giới thiệu giao tiếp I2C

I2C kết hợp các tính năng tốt nhất của SPI và UART. Với I2C, bạn có thể kết nối nhiều slave với một master duy nhất (như SPI) và bạn có thể có nhiều master điều khiển một hoặc nhiều slave.

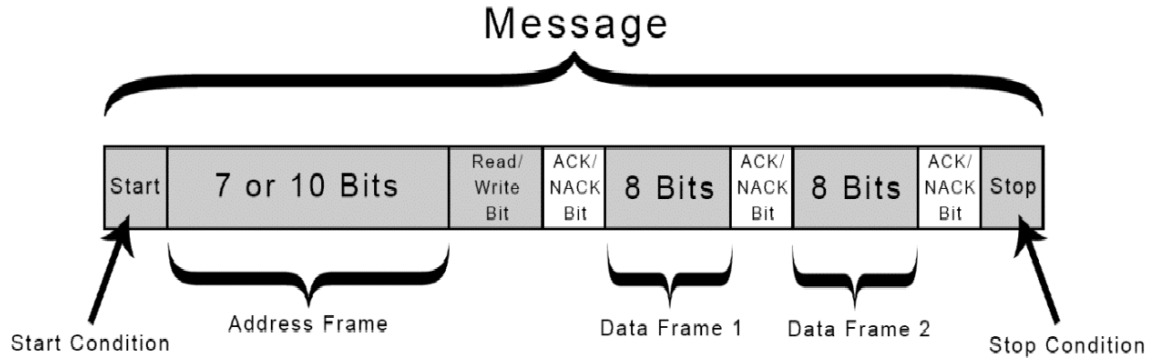


Hình 2. 13. Mô tả kết nối I2C 1M-1S

- SDA (Serial Data) - đường truyền cho master và slave để gửi và nhận dữ liệu.
- SCL (Serial Clock) - đường mang tín hiệu xung nhịp.

## b. Cách hoạt động của giao tiếp I2C

Với I2C, dữ liệu được truyền trong các tin nhắn. Tin nhắn được chia thành các khung dữ liệu. Mỗi tin nhắn có một khung địa chỉ chứa địa chỉ nhị phân của địa chỉ slave và một hoặc nhiều khung dữ liệu chứa dữ liệu đang được truyền. Thông điệp cũng bao gồm điều kiện khởi động và điều kiện dừng, các bit đọc / ghi và các bit ACK / NACK giữa mỗi khung dữ liệu:



Hình 2. 14. Mô tả đường truyền dữ liệu của giao tiếp I2C

- Điều kiện khởi động: Đường SDA chuyển từ mức điện áp cao xuống mức điện áp thấp trước khi đường SCL chuyển từ mức cao xuống mức thấp.
- Điều kiện dừng: Đường SDA chuyển từ mức điện áp thấp sang mức điện áp cao sau khi đường SCL chuyển từ mức thấp lên mức cao.
- Khung địa chỉ: Một chuỗi 7 hoặc 10 bit duy nhất cho mỗi slave để xác định slave khi master muốn giao tiếp với nó.
- Bit Đọc / Ghi: Một bit duy nhất chỉ định master đang gửi dữ liệu đến slave (mức điện áp thấp) hay yêu cầu dữ liệu từ nó (mức điện áp cao).
- Bit ACK / NACK: Mỗi khung trong một tin nhắn được theo sau bởi một bit xác nhận / không xác nhận. Nếu một khung địa chỉ hoặc khung dữ liệu được nhận thành công, một bit ACK sẽ được trả lại cho thiết bị gửi từ thiết bị nhận.

### 2.2.3. Khởi hiển thị - màn hình Oled

#### 2.2.3.1. Tổng quan về màn hình OLED

Màn hình Oled 0.96 inch giao tiếp I2C cho khả năng hiển thị đẹp, sang trọng, rõ nét vào ban ngày và khả năng tiết kiệm năng lượng tối đa với mức chi phí phù hợp, màn hình sử dụng giao tiếp I2C cho chất lượng đường truyền ổn định và rất dễ giao tiếp chỉ với 2 chân GPIO.





Hình 2. 15. Màn hình Oled 0.96inch giao tiếp I2C

### 2.2.3.2. Thông số kỹ thuật OLED

- Điện áp sử dụng: 2.2-5.5VDC
- Công suất tiêu thụ: 0.04w
- Góc hiển thị: lớn hơn 160 độ
- Số điểm hiển thị: 128x64 điểm.
- Giao tiếp: I2C
- Driver: SSD1306

### 2.2.4. Khối nguồn

Sử dụng bộ Module ổn định điện áp 5V-1A từ 2 pin 18650 (3.7V) để cung cấp nguồn cho mỗi thiết bị.

#### Mô tả chức năng:

- Chức năng UPS: Khi nguồn điện được bật nguồn, nguồn điện trực tiếp cung cấp năng lượng cho tải qua mạch tăng áp và đồng thời sạc pin lithium. Khi nguồn điện bị cắt, đường dẫn của pin sẽ tự động được kích hoạt để cung cấp năng lượng cho tải thông qua mạch tăng áp.
- Mạch tăng cường: công suất đỉnh lên đến 20W, công suất định mức lên đến 15W, mạch tăng cường BOST, tích hợp MOS dòng cao 12A, hiệu suất lên đến 96%, công suất tiêu thụ tĩnh mô-đun 25mA, đầu ra 5V
- Bảo vệ pin Lithium: sạc quá mức, xả quá mức, quá nhiệt, bảo vệ quá dòng
- Kích thước mô-đun: 89 \* 42 \* 22,4mm (hộp pin và mặt sau)



Hình 2. 16. Mạch nguồn UPS 5V kết hợp khay pin



Hình 2. 17. Pin 18650

### Thông tin chi tiết của Pin18650

- Pin Sạc lithium-ion 3,7v Chính Hãng dung lượng định danh 4200mAh.
- Sử dụng cho máy khoan, xe ,tông đơ , đèn pin ,pin sạc dự phòng..... .các thiết bị điện áp 4v.
- Pin cho phép sạc lại đến 600 lần , tương đương 2 - 3 năm mà không làm giảm chất lượng pin.
- Thông tin chi tiết sản phẩm:
- Loại pin: lithium ion (li-ion)
- Dòng xả : 8A ~10A
- Kích thước khoảng : 18mmx65mm/ viên
- Điện thế : 3.7V, khi sạc đầy có thể đạt đến 4.2v
- Dung lượng định danh: 4200mAh
- Dung lượng thực tế : 1800~ 2000mAh
- Nội trở : 16- 18Ω
- Kích thước: 18mm x 65mm

### 2.2.5. Khối Mobile App

#### 2.2.5.1. Mobile App

Mobile app là một chương trình ứng dụng dành riêng cho thiết bị di động như tablet hay smartphone cho phép người dùng có thể sử dụng để truy cập vào nội dung mà bạn mong muốn ngay trên thiết bị điện thoại di động đó. Những nhà lập trình mobile app sẽ biến nó trở thành một không gian lớn được thu nhỏ để người dùng có thể giải trí, xem tin tức, mua sắm,...

Trong IoT các ứng dụng được phát triển phù hợp cho mục đích theo dõi giám sát và điều khiển từ xa các thiết bị. Ngoài ra, các App phục vụ trong IoT cũng đã phát triển lưu trữ dữ liệu ngày càng hoàn thiện với sự phát triển của các nền tảng đi

kèm. Hiện nay, đang được phát triển rất nhiều trên hệ thống nhà thông minh, công nghiệp, y tế,....



Hình 2. 18. Tổng quan về Mobile App

Sự kết hợp của các công nghệ phần cứng và phần mềm, bao gồm giao thức truyền thông, dịch vụ đám mây, cơ sở dữ liệu, phụ trợ, giao diện người dùng và công nghệ máy học, là cần thiết để phát triển ứng dụng IoT. Nhu cầu và khả năng cá nhân của ứng dụng IoT sẽ xác định ngăn xếp công nghệ để sử dụng.

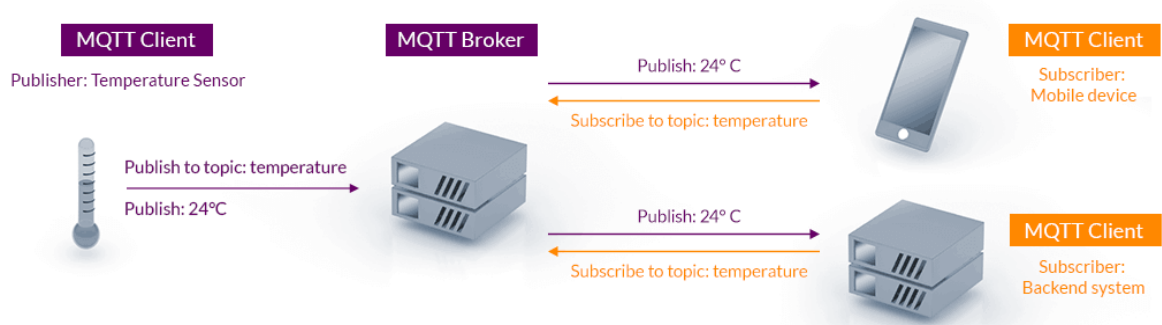
#### 2.2.5.2. Giới thiệu về giao thức mạng MQTT

Trong đề tài này, em sử dụng MQTT để cập nhật dữ liệu từ các thiết bị lên Mobile App để theo dõi nhịp tim và nồng độ Oxy trong máu từ xa thông qua kết nối WiFi.

##### a. MQTT

MQTT (Message Queuing Telemetry Transport) là một giao thức nhắn tin tiêu chuẩn OASIS cho Internet of Things (IoT). Nó được thiết kế như một phương tiện truyền tải tin nhắn publish/subscribe (xuất bản/đăng ký) cực kỳ nhẹ, lý tưởng để kết nối các thiết bị từ xa với băng thông mạng thấp. MQTT ngày nay được sử dụng trong nhiều ngành công nghiệp, chẳng hạn như ô tô, sản xuất, viễn thông, dầu khí,...

##### b. Cách MQTT hoạt động



Hình 2. 19. Mô tả cách hoạt động của MQTT

Một phiên MQTT được chia thành bốn giai đoạn: kết nối, xác thực, giao tiếp và kết thúc. Client (máy khách) bắt đầu bằng cách tạo kết nối Transmission Control Protocol/Internet Protocol (TCP/IP) tới broker bằng cách sử dụng cổng tiêu chuẩn hoặc cổng tùy chỉnh được xác định bởi các nhà phát triển broker.

Các cổng tiêu chuẩn là 1883 cho giao tiếp không mã hóa và 8883 cho giao tiếp được mã hóa – sử dụng Lớp cổng bảo mật (SSL) / Bảo mật lớp truyền tải (TLS). Trong quá trình giao tiếp SSL/TLS, máy khách cần kiểm chứng và xác thực máy chủ. Máy khách cũng có thể cung cấp tính xác thực máy khách cho broker trong quá trình giao tiếp. Broker có thể sử dụng điều này để xác thực máy khách. Mặc dù không phải là một phần cụ thể của đặc trưng MQTT, nhưng các broker đã trở thành thông lệ để hỗ trợ xác thực máy khách bằng SSL/TLS phía máy khách.

**Trong đó:**

- Broker: là một thành phần trung gian (môi giới)
- SSL/TLS (Secure Sockets Layer/Transport Layer Security) là kỹ thuật mã hóa truyền tin trên internet.

MQTT được gọi là một giao thức nhẹ vì tất cả các thông điệp của nó chỉ có một mã nhỏ. Mỗi thông báo bao gồm một tiêu đề cố định – 2 byte – một tiêu đề biến tùy chọn, dung lượng thông điệp được giới hạn ở 256 megabyte (MB) và mức chất lượng dịch vụ ( QoS ).

## **2.3. Giới thiệu các phần mềm sử dụng trong hệ thống**

### **2.3.1. Giới thiệu về Visual Studio Code**

#### **2.3.1.1. Visual code**

Visual Studio Code là công cụ soạn thảo mã nguồn do Microsoft phát triển. Nó được giới thiệu lần đầu tiên vào năm 2015 và chính thức được phát hành vào năm 2016.

Có thể thấy Visual Studio Code là sự kết hợp rất hoàn hảo giữa IDE VÀ Code Editor. Nó hỗ trợ cho người dùng rất nhiều tiện ích như: đổi theme, hỗ trợ Git, cải tiến mã nguồn, ...



*Hình 2. 20. Phần mềm Visual Studio Code*

## 2.3.2. Giới thiệu về Android Studio

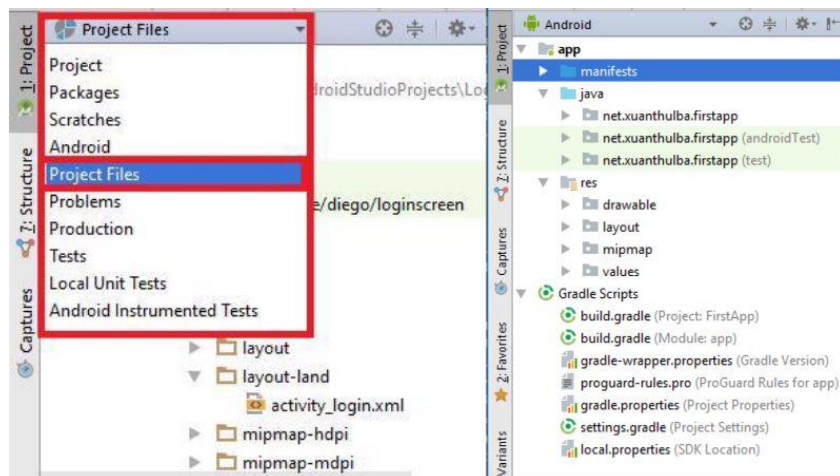
### 2.3.2.1. Android Studio

Android Studio là một môi trường tích hợp phát triển (Integrated Development Environment – IDE) được phát triển bởi Google dành cho việc phát triển ứng dụng trên nền tảng Android. Nó cung cấp một loạt các công cụ và tính năng để giúp nhà phát triển xây dựng và triển khai ứng dụng Android một cách dễ dàng.

### 2.3.2.2 Cấu trúc của file và thành phần Project

Trong Android Studio, cấu trúc của một dự án bao gồm các thành phần chính sau:

- **Project:** Đây là thư mục gốc của dự án và chứa tất cả các file và thư mục liên quan đến dự án. Trong thư mục này, bạn sẽ tìm thấy các file cấu hình, mã nguồn và tài nguyên của dự án.
- **Module:** Một dự án Android có thể bao gồm nhiều module, mỗi module đại diện cho một thành phần riêng biệt trong dự án như ứng dụng chính, thư viện, module kiểm thử và module khác. Mỗi module có thể có các tệp và thư mục riêng của nó.
- **Android:** Đây là một bộ lọc mặc định trong Android Studio và được sử dụng để nhóm các tệp liên quan đến phát triển ứng dụng Android. Bộ lọc này hiển thị các thành phần như mã nguồn Java, tài nguyên (như hình ảnh, bố cục, tệp xml), các tệp Manifest và các tệp cấu hình khác của ứng dụng.
- **Scratches:** Thư mục này chứa các file scratch (tạm thời) được sử dụng để viết và thử nghiệm mã nguồn tạm thời hoặc các đoạn mã ngắn.



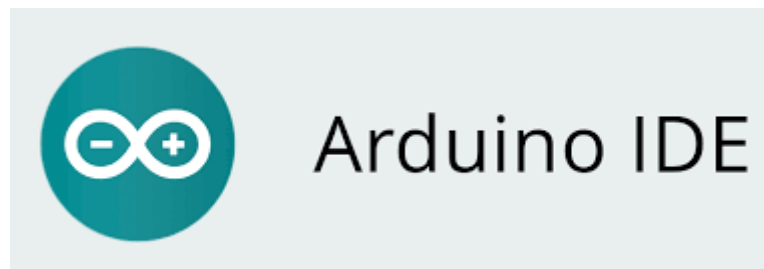
Hình 2. 21. Cấu trúc của file và thành phần Project

Cấu trúc của một dự án Android có thể thay đổi tùy thuộc vào cách bạn tổ chức dự án của mình và các tùy chọn cấu hình. Bạn có thể tùy chỉnh cấu trúc dự án và hiển thị các thành phần khác nhau bằng cách sử dụng các bộ lọc và tùy chọn.

### 2.3.3. Giới thiệu về Arduino IDE

#### a. Phần mềm Arduino IDE

- Arduino IDE là một phần mềm mã nguồn mở chủ yếu được sử dụng để viết và biên dịch mã vào module Arduino.
- Mã chính, còn được gọi là sketch, được tạo trên nền tảng IDE sẽ tạo ra một file Hex, sau đó được chuyển và tải lên trong bộ điều khiển trên bo.
- Môi trường IDE chủ yếu chứa hai phần cơ bản: Trình chỉnh sửa và Trình biên dịch, phần đầu sử dụng để viết mã được yêu cầu và phần sau được sử dụng để biên dịch và tải mã lên module Arduino.
- Môi trường này hỗ trợ cả ngôn ngữ C và C ++.



Hình 2. 22. Phần mềm lập trình Arduino IDE

#### b. Cách Arduino IDE hoạt động

Khi người dùng viết mã và biên dịch, IDE sẽ tạo file Hex cho mã. File Hex là các file thập phân Hexa được Arduino hiểu và sau đó được gửi đến bo mạch bằng cáp USB. Mỗi bo Arduino đều được tích hợp một bộ vi điều khiển, bộ vi điều khiển sẽ nhận file hex và chạy theo mã được viết.

### 2.4 Kết luận chương 2

Sau khi đã xác định được hướng thiết kế phù hợp, đồng thời hiểu rõ nguyên lý hoạt động và cơ sở lý thuyết của hệ thống đo nhịp tim và nồng độ Oxy trong máu, em đã tiến hành nghiên cứu, lựa chọn các bộ phận, chi tiết phần cứng và phần mềm cần thiết để xây dựng, thiết kế đề tài một cách chi tiết và chính xác. Các thành phần cấu tạo của hệ thống được xác định dựa trên yêu cầu kỹ thuật, khả năng đáp ứng thực tế cũng như tính khả thi trong việc thi công.

Song song với đó, em đã chuẩn bị đầy đủ các thành phần phần cứng cần thiết, đảm bảo chất lượng và tính tương thích cao giữa các linh kiện. Phần mềm cũng được thiết kế và thử nghiệm trước, nhằm đảm bảo độ chính xác, tính ổn định, và khả năng mở rộng khi triển khai thực tế.



## CHƯƠNG 3: THIẾT KẾ HỆ THỐNG GIÁM SÁT NHỊP TIM VÀ NỒNG ĐỘ OXY TRONG MÁU

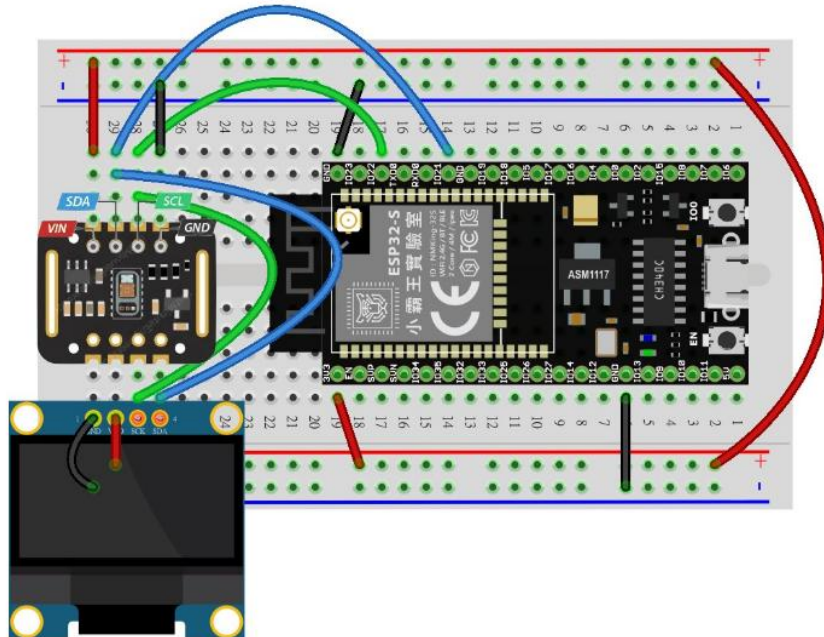
### 3.1. Thiết kế phần cứng

#### 3.1.1. Thiết kế sơ đồ đấu nối của hệ thống

Trên cơ sở sơ đồ khối tổng thể và thực tế khảo sát các chủng loại thiết bị hiện có trên thị trường, các thiết bị được lựa chọn đảm bảo các tiêu chí đặt ra cho hệ thống về giá thành, tính mở và tính đơn giản.

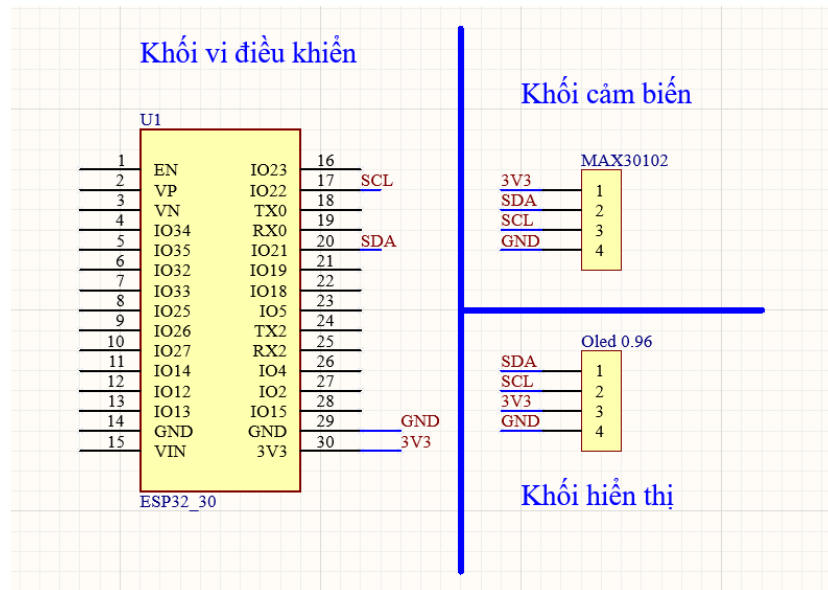
Từ đó, các thiết bị được kết nối dựa trên sơ đồ chân theo thiết kế ở dưới. Dòng thiết bị vi điều khiển ESP32 được chọn làm phần lõi cho hệ thống. Đây là dòng thiết bị có đặc trưng mở, dễ sử dụng và khá phổ biến trên thị trường, phù hợp với các thiết kế nhỏ gọn, linh hoạt để nghiên cứu phát triển các ý tưởng ban đầu.

Vi điều khiển ESP32 được cắm trực tiếp với máy tính để nạp code điều khiển nhận dữ liệu từ cảm biến Max30102 về để xử lý dữ liệu nhận được. Cảm biến Max30102 được lựa chọn vì giá thành rẻ, cho kết quả đo tương đối chính xác, có độ tích hợp cao, sử dụng giao tiếp I2C phổ biến trong công nghiệp, điều khiển ngõ ra có đệm với số lượng thành phần linh kiện bên ngoài tối thiểu, có thể lập trình điều khiển các chân tín hiệu vào/ra một cách dễ dàng. Module Oled I2C được lựa chọn kết nối thông qua bộ thích ứng chuẩn giao diện I2C (I2C Adapter) để đơn giản hóa và tiết kiệm chân truyền thông tin giữa vi điều khiển và Oled.



Hình 3. 1. Sơ đồ đấu nối của thiết bị

### 3.1.2. Sơ đồ nguyên lý của hệ thống



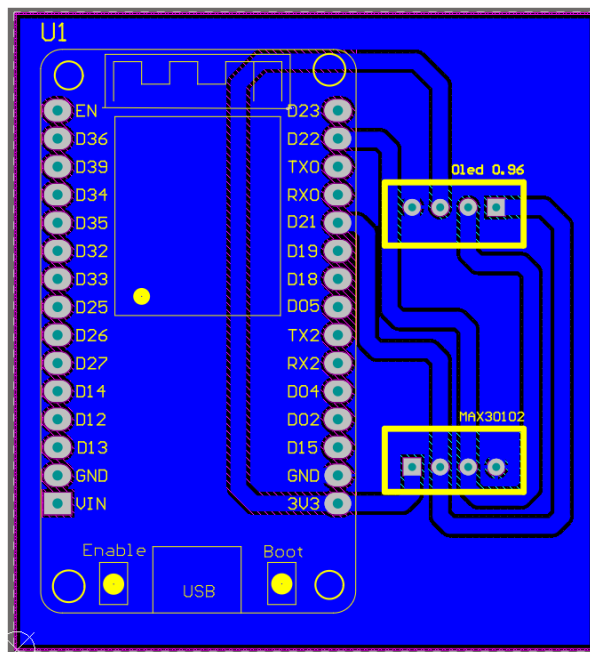
Hình 3. 2. Sơ đồ nguyên lý của thiết bị

### 3.1.3. Thiết kế sơ đồ mạch PCB của hệ thống

Hệ thống giám sát nhịp tim và nồng độ oxy trong máu là một ứng dụng quan trọng trong lĩnh vực y tế, giúp theo dõi sức khỏe người dùng một cách hiệu quả và liên tục. Hệ thống được thiết kế với sự tham khảo kỹ lưỡng từ sơ đồ nguyên lý của các nhà sản xuất Kit, đồng thời cải tiến để phù hợp với mục tiêu nghiên cứu của em. Kết quả là một mô hình cơ bản được xây dựng như sau:

- Sơ đồ mạch PCB**

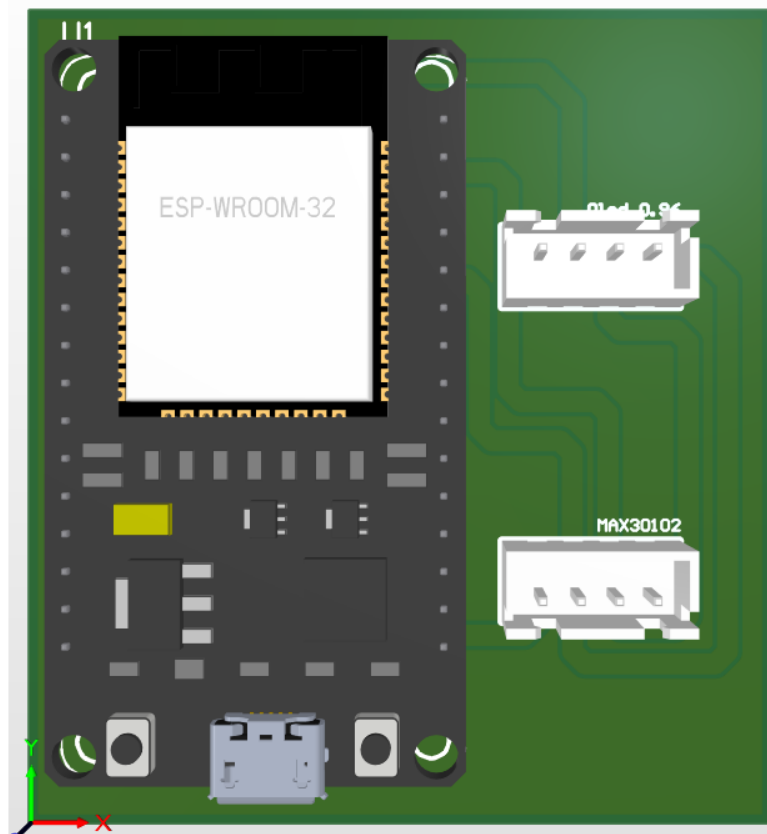
- a. Sơ đồ mạch PCB**



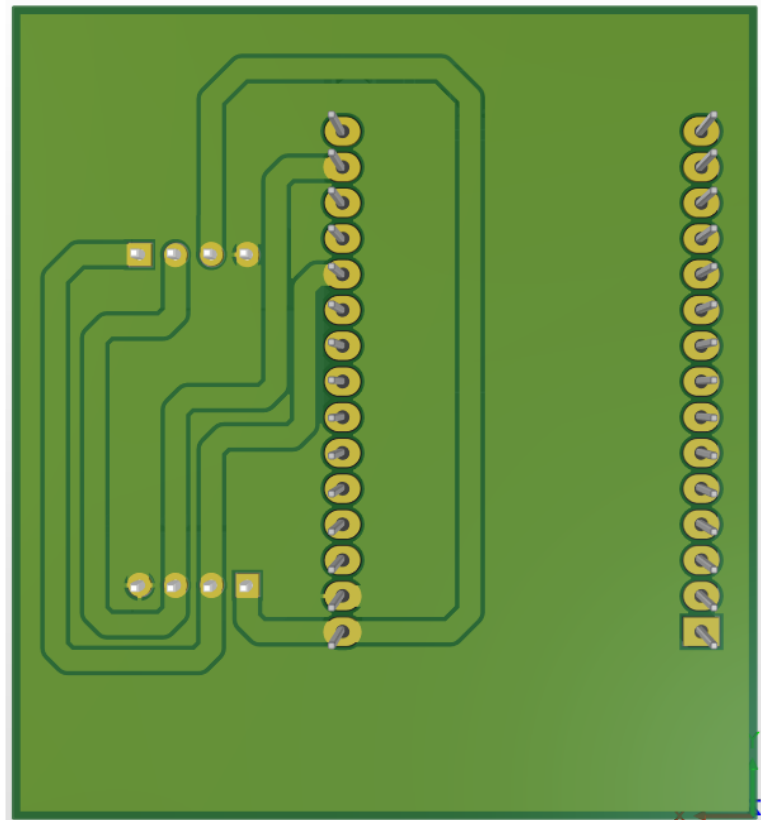
Hình 3. 3. Sơ đồ mạch PCB



**b. Sơ đồ 3d của mạch PCB sau khi lắp đầy đủ linh kiện**



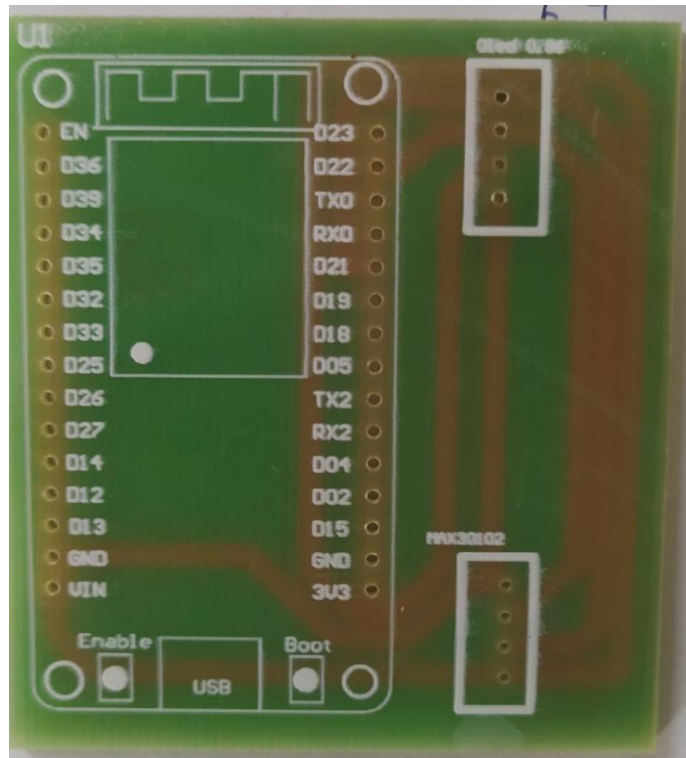
*Hình 3. 4. 3D mặt trước của PCB*



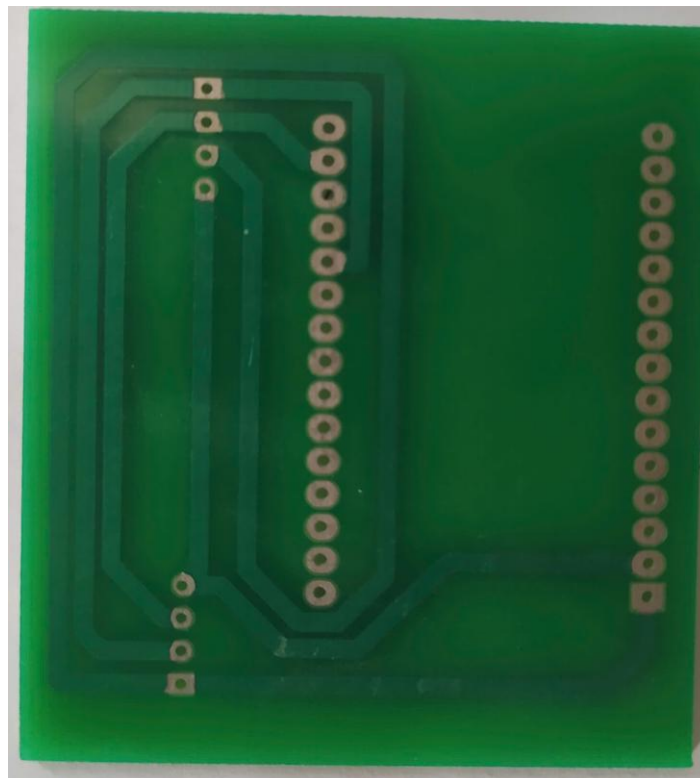
*Hình 3. 5. 3D mặt sau của PCB*

### 3.2. Thi công lắp đặt hệ thống

Sau khi thiết kế mạch PCB bằng phần mềm Altium Designer và đặt mạch gia công ta thu được mạch in như sau:

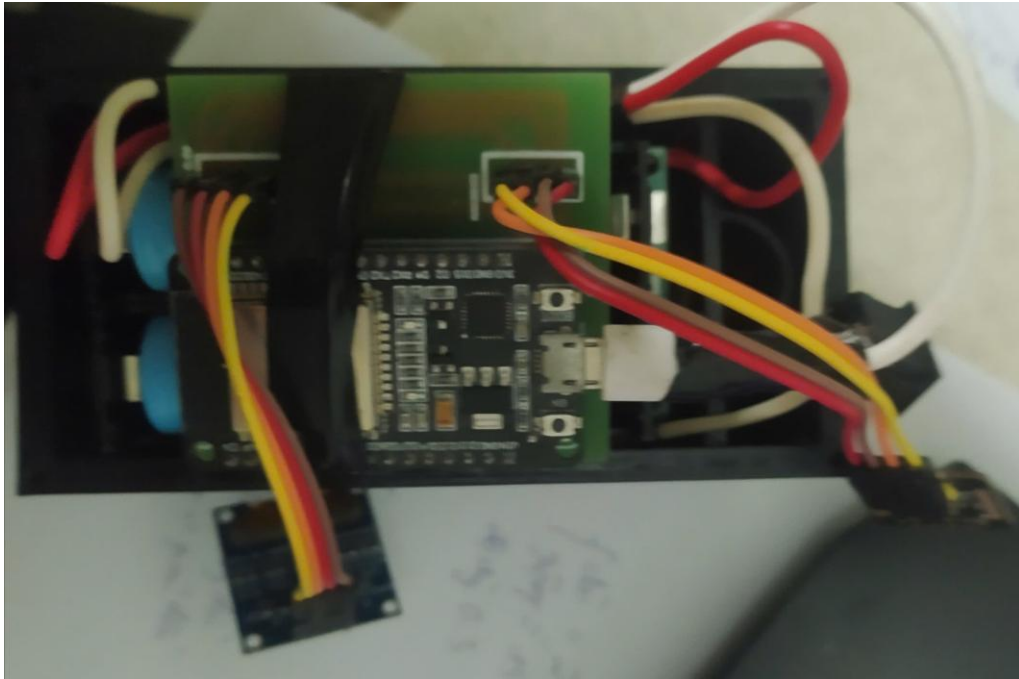


Hình 3. 6. Mặt trước của mạch sau khi gia công



Hình 3. 7. Mặt sau của mạch sau khi gia công

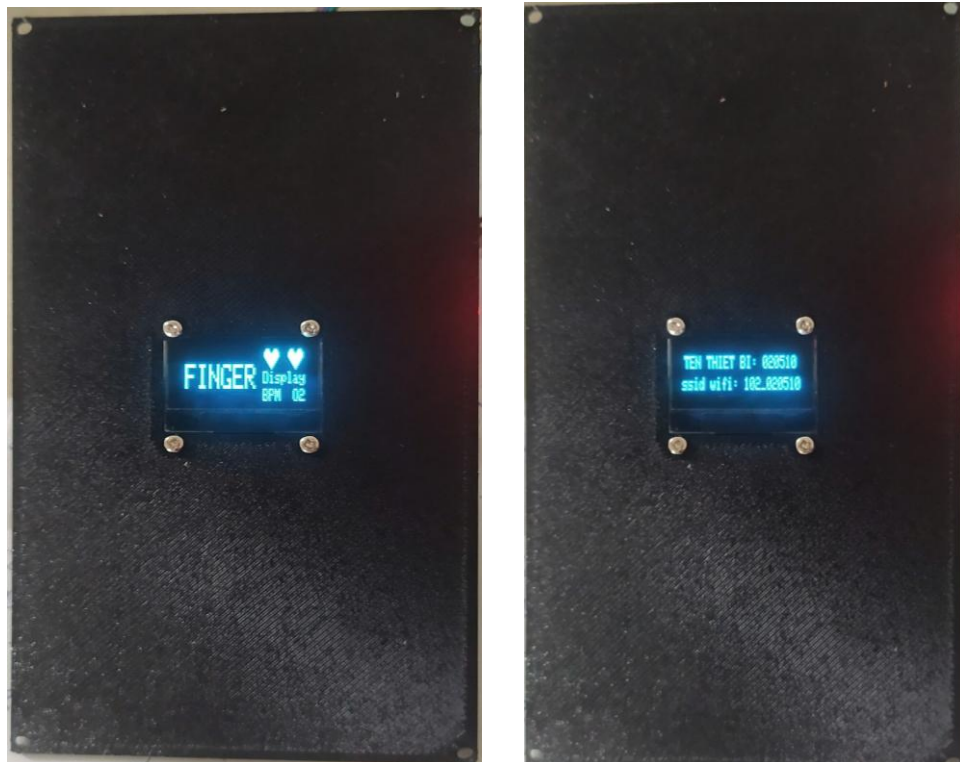
Sau khi hàn linh kiện vào mạch PCB ta thu được mạch điều khiển cho hệ thống có hình dạng như sau:



*Hình 3. 8. Mạch sau được hàn linh kiện*

### 3.3. Sản phẩm sau khi hoàn thiện

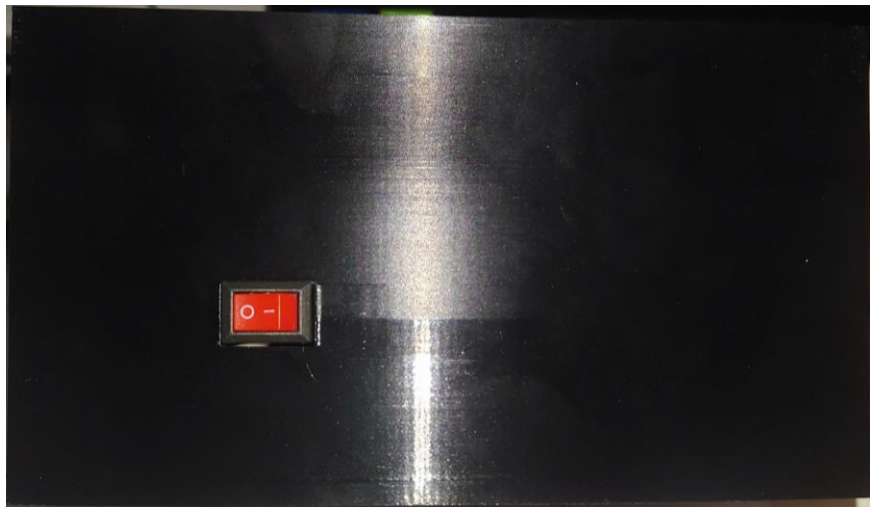
Sau khi hoàn thành chuẩn bị các thành phần trên ta tiến hành lắp ráp thiết bị Hệ thống sau khi đã được lắp ráp xong:



*Hình 3. 9. Mặt trước của sản phẩm khi hoàn thiện*



*Hình 3. 10. Mặt sau của sản phẩm sau khi hoàn thiện*



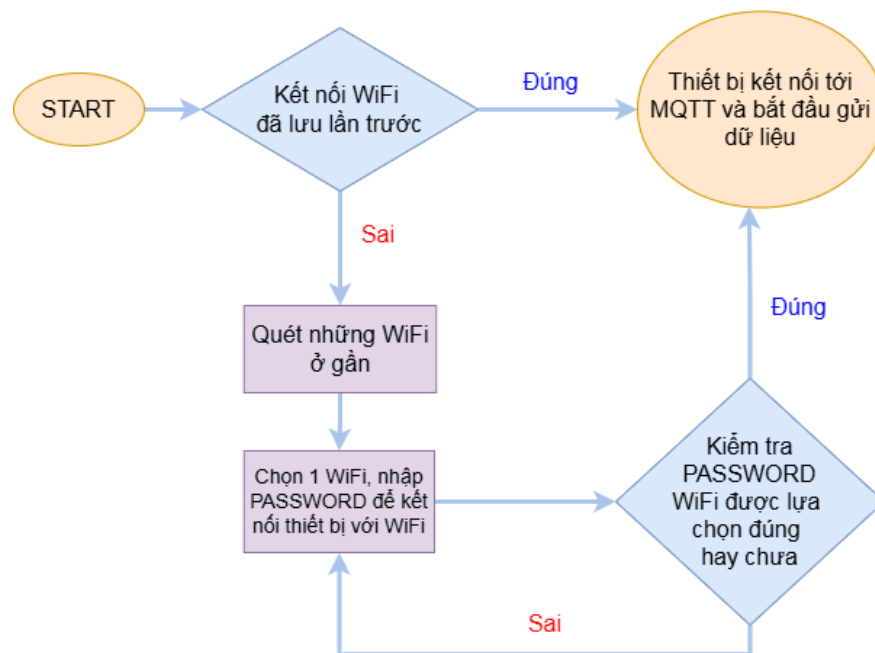
*Hình 3. 11. Mặt bên của sản phẩm sau khi hoàn thiện*



*Hình 3. 12. Phần chân cắm sạc của sản phẩm*

### 3.4 Thiết kế phần mềm

#### 3.4.1. Lưu đồ thuật toán của Web đăng nhập WiFi cho thiết bị



Hình 3. 13. Lưu đồ thuật toán của Web đăng nhập WiFi cho thiết bị

Lưu đồ thuật toán này mô tả quy trình hoạt động của thiết bị khi đăng nhập WiFi cho thiết bị qua Web. Dưới đây là giải thích từng bước của lưu đồ:

- **START:** Bắt đầu khởi chạy thiết bị và thiết bị sẽ vào chế độ AP để đăng nhập WiFi cho thiết bị.
- **Kết nối WiFi đã lưu từ lần trước:**
  - Thiết bị sẽ cố gắng kết nối tới WiFi đã lưu từ lần đăng nhập trước đó.
  - **Đúng:** Nếu như đã kết nối được với WiFi lần trước đó thì thiết bị sẽ kết nối tới MQTT và gửi dữ liệu lên MQTT.
  - **Sai:** tới bước tiếp theo.
- **Quét những WiFi ở gần:**
  - Thiết bị sẽ quét ra những WiFi đang ở gần để người dùng có thể đăng nhập WiFi cho thiết bị/
- **Chọn 1 WiFi, nhập PASSWORD để kết nối thiết bị với WiFi**
- **Kiểm tra PASSWORD WiFi được lựa chọn đúng hay chưa:**
  - **Đúng:** Nếu như đã kết nối được với WiFi lần trước đó thì thiết bị sẽ kết nối tới MQTT và gửi dữ liệu lên MQTT.
  - **Sai:** Quay lại chọn lại 1 WiFi, và nhập lại PASSWORD WiFi để đăng nhập WiFi cho thiết bị.



### 3.4.2. Thiết kế giao diện Web đăng nhập WiFi

#### a. Giao diện tổng quan



Hình 3. 14. Giao diện tổng quát Web đăng nhập WiFi

- **Nút nhấn Configure WiFi**

Khi nhấn nút này, giao diện sẽ chuyển sang chế độ quét các mạng WiFi xung quanh mà thiết bị ESP32 có thể phát hiện được. Danh sách các mạng WiFi sẽ được hiển thị, cho phép người dùng lựa chọn và nhập thông tin đăng nhập (SSID và mật khẩu) để kết nối thiết bị với mạng WiFi mong muốn.

- **Nút nhấn Info**

Nút này đưa người dùng đến giao diện hiển thị thông tin chi tiết về trạng thái hiện tại của chip ESP32. Các thông tin có thể bao gồm địa chỉ IP, cường độ tín hiệu WiFi (RSSI), trạng thái kết nối mạng, phiên bản firmware đang sử dụng, dung lượng bộ nhớ khả dụng, và các thông số hệ thống khác.

- **Nút nhấn Exit**

Khi người dùng nhấn vào nút này, giao diện sẽ đóng lại và người dùng sẽ không thực hiện cấu hình WiFi cho thiết bị.

- **Nút nhấn Update**

Nhấn nút này sẽ chuyển giao diện sang chế độ cập nhật firmware qua WiFi. Thiết bị ESP32 sẽ phát sóng dưới dạng điểm truy cập (Access Point - AP), cho phép người dùng kết nối và tải lên tệp firmware mới để cập nhật hệ thống.

## b. Giao diện Configure WiFi

Hình 3. 15. Giao diện Configure WiFi

Ở giao diện này, thiết bị sẽ quét ra những WiFi xung quanh mà ESP32 có thể phát hiện được. Danh sách các mạng WiFi sẽ được hiển thị lên màn hình. Người dùng sẽ lựa chọn WiFi mà người dùng biết PASSWORD và đăng nhập cho thiết bị. Nếu đúng thiết bị sẽ kết nối tới WiFi và kết nối tới server MQTT để gửi dữ liệu từ cảm biến lên MQTT. Nếu sai sẽ quay lại giao diện đăng nhập WiFi. Giao diện này thường được sử dụng để cấu hình kết nối mạng cho các thiết bị IoT hiện có.

## c. Giao diện Info

Hình 3. 16. Giao diện Info

Giao diện Info hiển thị thông tin chi tiết của ESP32 đang sử dụng với các thông số quan trọng như Chip ID, Flash size, PSRAM Size,...Đặc biệt là hiển thị bộ nhớ đang sử dụng trong ESP32 đang dùng là bao nhiêu. Ngoài ra, còn tận dụng cảm biến Hall bên trong chip ESP32 để lấy giá trị nhiệt độ và hiển thị một số thông tin chi tiết của WiFi đang phát ra từ ESP32 ở chế độ AP.

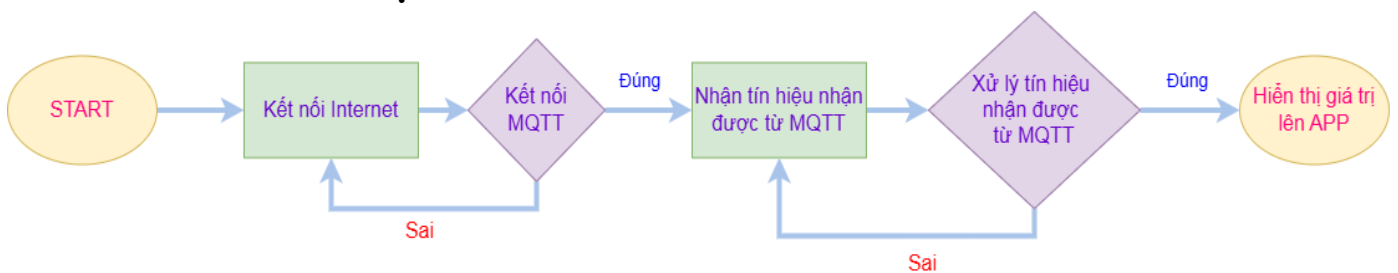
#### d. Giao diện UPDATE



Hình 3. 17. Giao diện UPDATE

Với giao diện Update người dùng sẽ chọn file .bin tương ứng với phiên bản cập nhật tiếp theo của sản phẩm để update firmware cho thiết bị thông qua WiFi của ESP32 ở chế độ AP. Trong những sản phẩm IoT hiện nay phần cập nhật Firmware đã được tích hợp khá phổ biến và quan trọng của các sản phẩm IoT.

#### 3.4.3. Lưu đồ thuật toán của APP



Hình 3. 18. Lưu đồ thuật toán của App

Lưu đồ thuật toán này mô tả quy trình hoạt động của một hệ thống sử dụng MQTT để nhận và xử lý dữ liệu. Dưới đây là giải thích từng bước của lưu đồ:

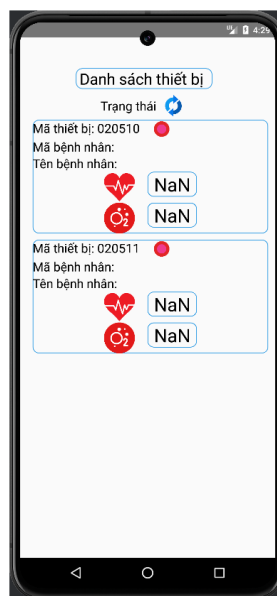
- **START**
  - Đây là điểm bắt đầu của hệ thống. Hệ thống sẽ khởi động và bắt đầu quy trình.
- **Kết nối Internet**
  - Hệ thống cố gắng kết nối với mạng Internet.
  - **Sai (Không kết nối được):** Nếu không kết nối được Internet, hệ thống quay lại bước này và thử kết nối lại.





- **Đúng (Kết nối thành công):** Tiếp tục đến bước tiếp theo.
- **Kết nối MQTT**
  - Sau khi có Internet, hệ thống cố gắng kết nối với server MQTT.
  - **Sai (Không kết nối được):** Nếu không kết nối được với MQTT, hệ thống quay lại bước này và thử kết nối lại.
  - **Đúng (Kết nối thành công):** Tiếp tục đến bước tiếp theo.
- **Nhận tín hiệu nhận được từ MQTT**
  - Hệ thống chờ nhận tín hiệu từ server MQTT.
  - **Sai (Không nhận được tín hiệu):** Nếu không nhận được tín hiệu, hệ thống quay lại bước này để tiếp tục chờ.
  - **Đúng (Nhận được tín hiệu):** Tiếp tục đến bước tiếp theo.
- **Xử lý tín hiệu nhận được từ MQTT**
  - Hệ thống phân tích và xử lý dữ liệu nhận được từ MQTT.
  - **Sai (Xử lý thất bại):** Nếu xảy ra lỗi trong quá trình xử lý, hệ thống quay lại bước này và thử xử lý lại.
  - **Đúng (Xử lý thành công):** Tiếp tục đến bước cuối cùng.
- **Hiển thị giá trị lên APP**
  - Sau khi xử lý thành công, giá trị sẽ được hiển thị trên ứng dụng di động hoặc giao diện người dùng.
  - Đây là điểm kết thúc của quy trình.

### 3.4.4. Thiết kế giao diện App



#### a. Giao diện theo dõi tổng quát tất cả các thiết bị hiện có



Hình 3. 19. Giao diện theo dõi tổng quát tất cả các thiết bị hiện có

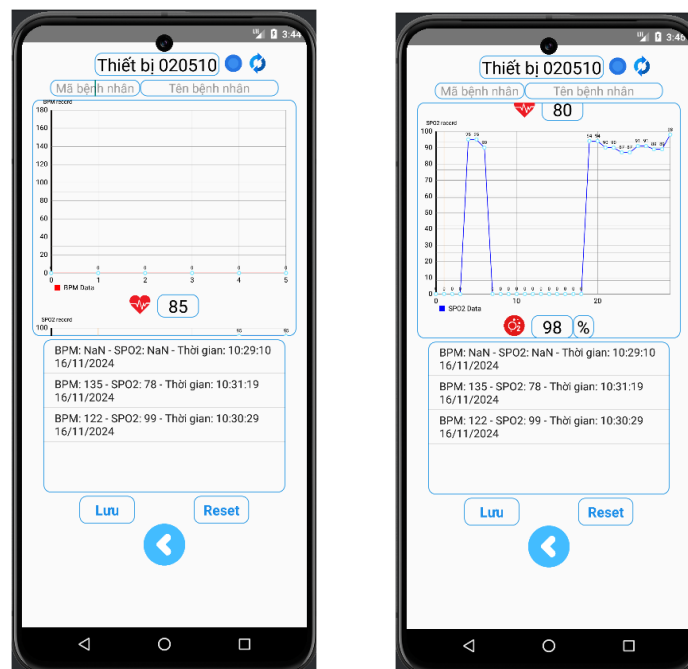
Ở giao diện này, người dùng sẽ theo dõi được là điện thoại có kết nối tới MQTT hay không. Nếu hiển thị  lúc này điện thoại đã kết nối tới server MQTT, còn nếu hiển thị  lúc này điện thoại chưa kết nối được tới MQTT thì sẽ có 2 trường hợp xảy ra với trường hợp thứ nhất là server MQTT bị mất, lúc này cần báo lại cho người sản xuất để sửa lại server MQTT, còn trường hợp thứ 2 là do điện thoại chưa kết nối Internet, lúc này người dùng cần kiểm tra lại trạng thái Internet của điện thoại.

Nếu điện thoại đã kết nối Internet và đã kết nối được với MQTT thì lúc này người dùng có thể theo dõi được những thiết bị nào đã kết nối được WiFi và MQTT để App có thể lấy dữ liệu từ MQTT về và hiển thị lên màn hình App.

Với những thiết bị hiển thị trạng thái  là những thiết bị chưa được sử dụng hoặc đang sử dụng nhưng chưa kết nối được tới WiFi hoặc chưa kết nối được tới MQTT. Còn nếu hiển thị trạng thái:  thì thiết bị đã được kết nối tới MQTT và lúc này những dữ liệu về nhịp tim và nồng độ Oxy trong máu sẽ được hiển thị lên màn hình App.

Với mỗi thiết bị sẽ có một mã cứng ID tương ứng để phân biệt với nhau và kiểm soát hệ thống thuận tiện, nhanh chóng và chính xác hơn với chỉ một App có thể kiểm soát nhiều thiết bị cùng một lúc. Nếu giá trị nhịp tim và nồng độ Oxy trong máu hiển thị là NaN thì có nghĩa là thiết bị đang chưa được sử dụng hoặc đã được sử dụng nhưng chưa kết nối tới WiFi và MQTT.

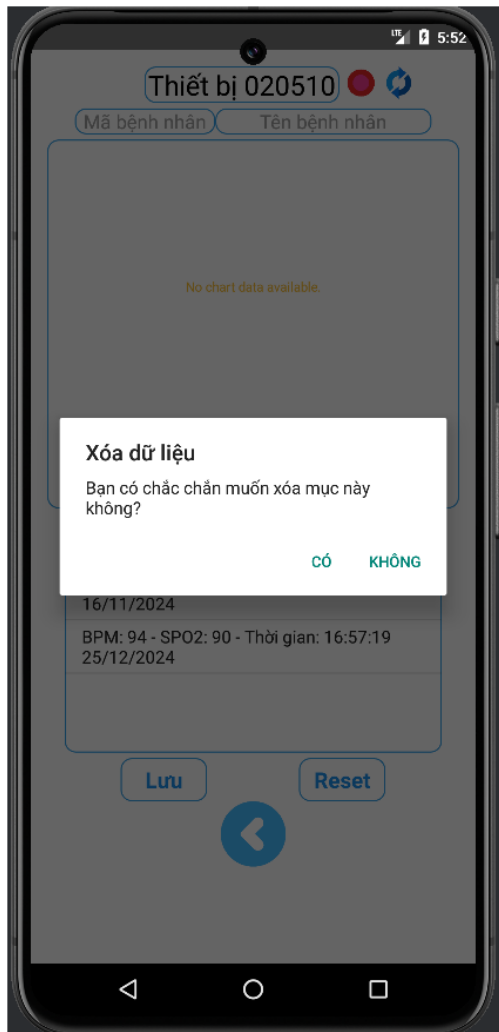
### b. Giao diện chi tiết của từng thiết bị



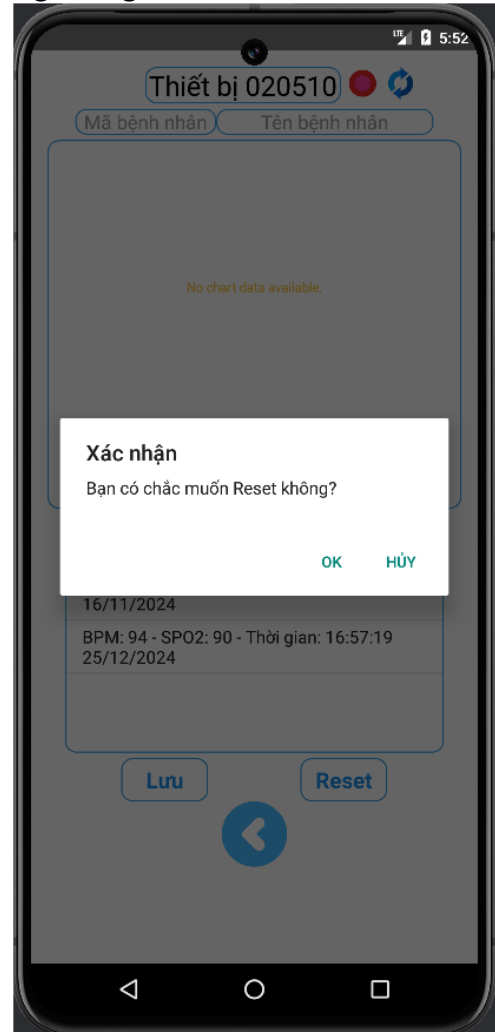
Hình 3. 20. Giao diện theo dõi chi tiết của từng thiết bị

Khi nhấn vào một thiết bị cụ thể trên giao diện chính, ứng dụng sẽ chuyển sang giao diện chi tiết của thiết bị đó. Giao diện này cung cấp thông tin chi tiết về trạng thái hoạt động và kết nối MQTT của thiết bị, bao gồm các chỉ số theo dõi sức khỏe như BPM (nhịp tim), SpO2 (nồng độ oxy trong máu), cùng với thời gian lấy mẫu dữ liệu gần nhất. Các biểu đồ trực quan được tích hợp để hiển thị sự thay đổi của các chỉ số theo thời gian, giúp người dùng dễ dàng nhận biết xu hướng sức khỏe của bệnh nhân.

Ngoài ra, còn lưu lịch sử đo theo từng thời điểm tùy ý người dùng muốn lưu lúc nào, và hỗ trợ xóa lịch sử, reset toàn bộ thông tin người bệnh đã lưu trên thiết bị.



Hình 3. 21: Xóa 1 dữ liệu lịch sử đo



Hình 3. 22: Reset toàn bộ thông tin

### 3.5. Kết luận chương 3

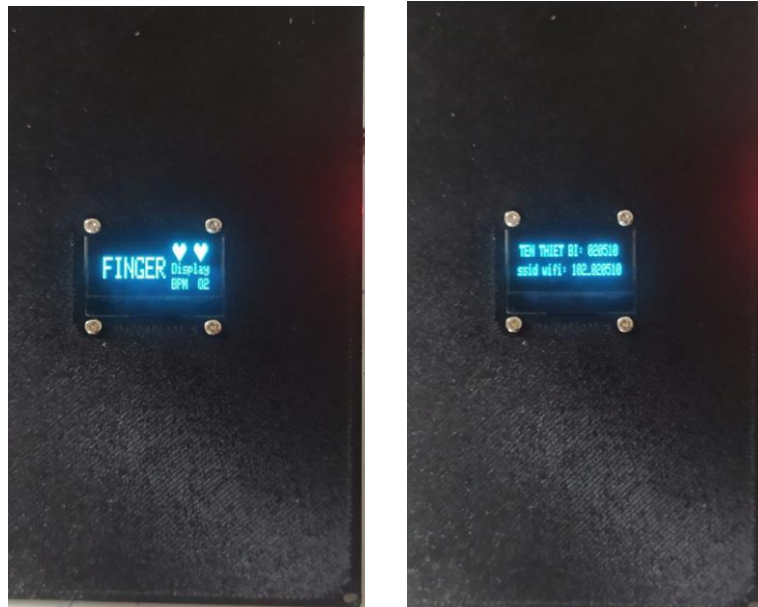
Trong quá trình thiết kế và phát triển hệ thống, em đã vận dụng và tích hợp hiệu quả các kiến thức được học từ nhiều lĩnh vực, bao gồm điện tử, lập trình, cơ khí, và tự động hóa. Việc này không chỉ giúp đảm bảo tính ổn định và độ tin cậy cao cho hệ thống mà còn tối ưu hóa hiệu suất và khả năng ứng dụng trong thực tiễn. Sự kết hợp chặt chẽ giữa các nguyên lý điện tử, phần mềm, và cơ học đã tạo ra một sản phẩm có khả năng vận hành mượt mà, linh hoạt và đáng tin cậy trong nhiều điều kiện khác nhau.

## CHƯƠNG 4: THỬ NGHIỆM VÀ ĐÁNH GIÁ HỆ THỐNG

### 4.1. Thử nghiệm hệ thống

#### 4.1.1. Đi vào thử nghiệm

- Hệ thống sau khi được cấp nguồn



Hình 4. 1. Thiết bị sau khi được cấp nguồn

- Tiến hành đăng nhập WiFi cho thiết bị:

Đăng nhập WiFi là bước quan trọng để đảm bảo thiết bị có thể kết nối với mạng không dây và thực hiện việc truyền tải dữ liệu từ cảm biến đến server MQTT.

Quy trình này được thiết kế linh hoạt để phù hợp với nhiều trường hợp, kể cả khi thiết bị chưa được cấu hình mạng hoặc không thể kết nối với WiFi đã lưu trước đó. Cụ thể:

- Trường hợp kết nối WiFi lần đầu hoặc không có kết nối WiFi đã lưu:

Khi thiết bị khởi động mà không tìm thấy kết nối WiFi đã lưu trong bộ nhớ, hệ thống sẽ tự động chuyển sang chế độ Access Point (AP).

Ở chế độ AP, thiết bị sẽ phát ra một mạng WiFi local riêng, với SSID (tên mạng) mặc định để người dùng dễ dàng nhận diện, ví dụ: "Device\_XXXX" (trong đó XXXX là mã định danh của thiết bị).

Người dùng sử dụng điện thoại hoặc máy tính để kết nối vào mạng WiFi do thiết bị phát ra, sau đó truy cập vào giao diện cài đặt qua trình duyệt web hoặc ứng dụng Mobile App.

Giao diện này cho phép người dùng nhập thông tin SSID và mật khẩu của mạng WiFi mà thiết bị cần kết nối.

### ○ Quá trình quét và kết nối WiFi:

Sau khi nhận được thông tin đăng nhập WiFi từ người dùng, thiết bị sẽ thực hiện quét các mạng WiFi xung quanh để tìm mạng phù hợp.

Nếu mạng được chỉ định có tín hiệu và cho phép kết nối, thiết bị sẽ bắt đầu quá trình đăng nhập vào mạng WiFi đó.

Thiết bị sẽ lưu thông tin SSID và mật khẩu vào bộ nhớ không thay đổi (non-volatile memory) để sử dụng trong các lần khởi động sau, tránh việc người dùng phải nhập lại thông tin.

### ○ Xử lý trường hợp nhập đúng hoặc sai thông tin WiFi:

#### - Nhập đúng thông tin WiFi:

Khi thiết bị kết nối thành công với mạng WiFi, nó sẽ tự động khởi động các dịch vụ liên quan đến việc truyền tải dữ liệu.

Ứng dụng Mobile App sẽ lấy dữ liệu từ server MQTT, xử lý và hiển thị thông tin đo đạc cho người dùng.

#### - Nhập sai thông tin WiFi:

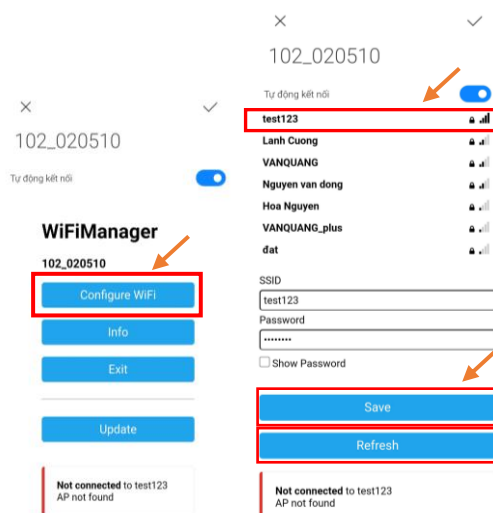
Nếu người dùng nhập sai SSID hoặc mật khẩu WiFi, thiết bị sẽ phát hiện lỗi và quay lại chế độ AP để yêu cầu người dùng nhập lại thông tin.

### ○ Hoạt động của thiết bị khi không có kết nối WiFi:

Trong trường hợp thiết bị không thể kết nối với WiFi (do lỗi mạng, nhập sai thông tin, hoặc khu vực không có sóng WiFi), hệ thống vẫn tiếp tục hoạt động bình thường.

Các cảm biến trên thiết bị vẫn thực hiện đo lường dữ liệu và hiển thị kết quả lên màn hình thiết bị cục bộ.

**Thiết bị sẽ vẫn hoạt động bình thường ngay cả khi không có kết nối WiFi chỉ là sẽ không gửi được dữ liệu lên MQTT thôi vì không có mạng.**



Hình 4. 2. Đăng nhập WiFi cho thiết bị

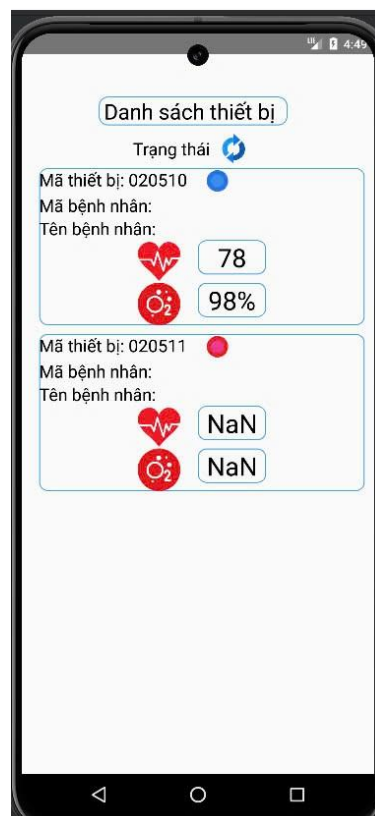
- **Tiến hành đo và gửi dữ liệu nhận được lên MQTT:**

Ngay cả khi thiết bị không có kết nối WiFi thiết bị vẫn đo và hiển thị giá trị cảm biến lên màn hình Oled 0.96inch.



Hình 4. 3. Dữ liệu đo hiển thị trên màn hình Oled 0.96 inch

Ngay khi có kết nối WiFi và MQTT thiết bị sẽ trực tiếp gửi dữ liệu từ cảm biến lên Mobile App.

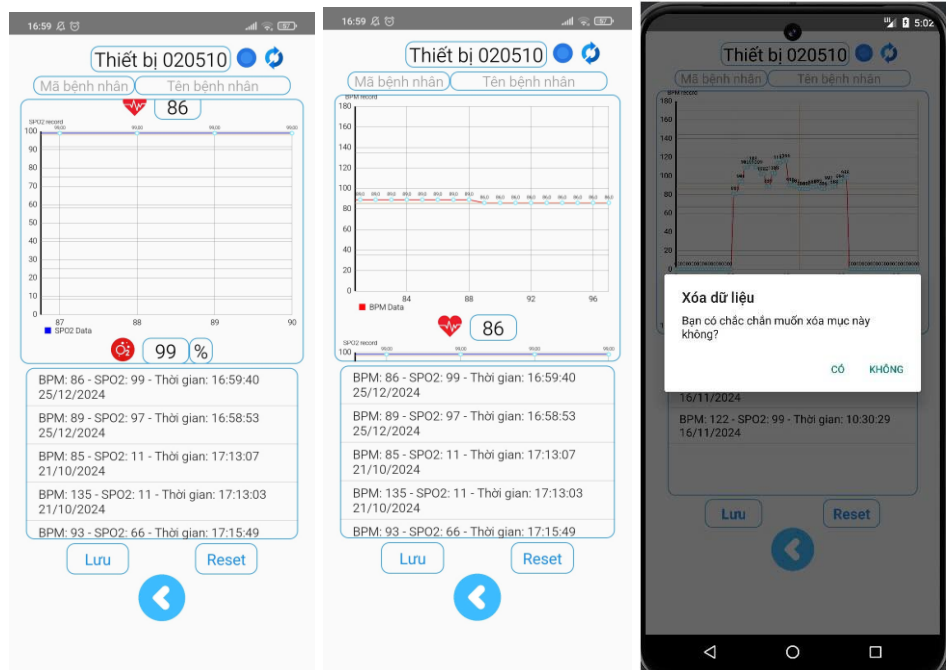


Hình 4. 4. Dữ liệu đo hiển thị trên Mobile App



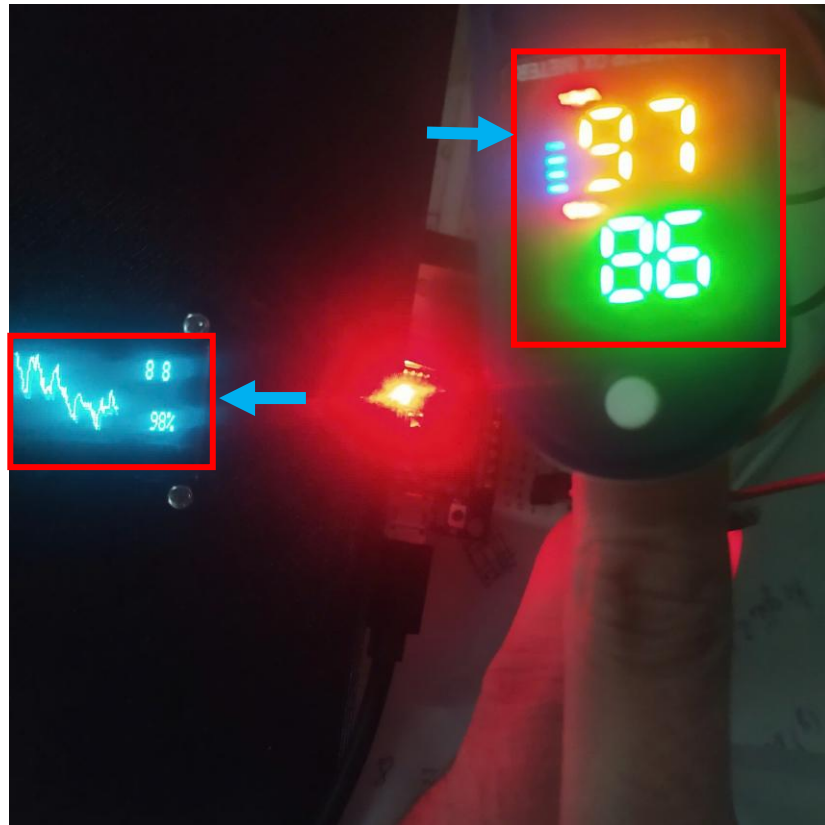
- **Tiến hành lưu giá trị tại một thời điểm bất kỳ:**

Giá trị nhận được sẽ được lưu lại tại một thời điểm bất kỳ nếu người dùng mong muốn và có thể xóa những dữ liệu không muốn lưu



Hình 4. 5. Lưu và xóa lịch sử trên App

#### 4.1.2 So sánh với sản phẩm hiện có trên thị trường



Hình 4. 6. So với sản phẩm trên thị trường

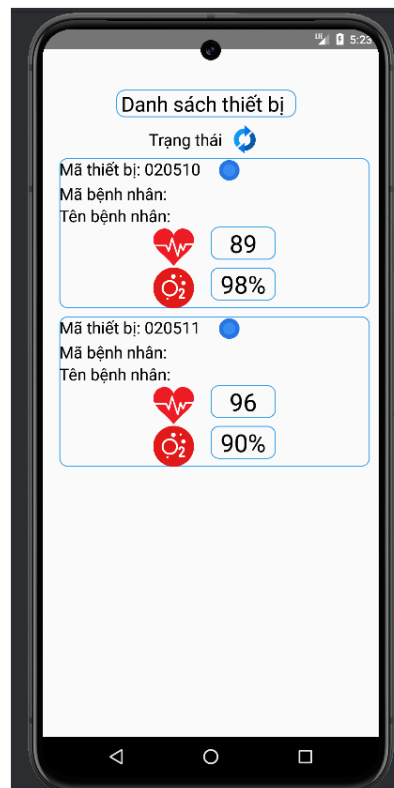
- **Giá trị SPO2:**

- Thiết bị trên thị trường hiển thị giá trị SPO2 là 97%, trong khi sản phẩm của em cũng đo được giá trị 97%.
- Điều này cho thấy thiết bị do em thực hiện có khả năng đo lường độ bão hòa oxy trong máu với độ chính xác rất cao, gần như không có sai số so với sản phẩm hiện có trên thị trường.

- **Giá trị BPM (nhịp tim):**

- Thiết bị trên thị trường hiển thị giá trị BPM là 86, trong khi sản phẩm của em đo được giá trị BPM là 88.
- Nguyên nhân dẫn đến sự chênh lệch này có thể do tốc độ cập nhật dữ liệu của cảm biến hoặc cách xử lý tín hiệu của thiết bị. Tuy nhiên, khoảng sai số này được đánh giá là không đáng kể và không ảnh hưởng nhiều đến khả năng theo dõi tình trạng sức khỏe của người dùng.
- Điều này chứng minh rằng sản phẩm của em có thể cung cấp các giá trị BPM tương đối chính xác và phù hợp để sử dụng trong các ứng dụng theo dõi sức khỏe.

#### 4.1.3. Hoạt động với nhiều thiết bị để tạo thành hệ thống



Hình 4. 7. Hoạt động với nhiều thiết bị để tạo thành hệ thống

Hệ thống hoạt động ổn định với nhiều thiết bị được theo dõi qua chung một App Mobile. Hệ thống đa thiết bị này không chỉ đáp ứng nhu cầu theo dõi sức khỏe

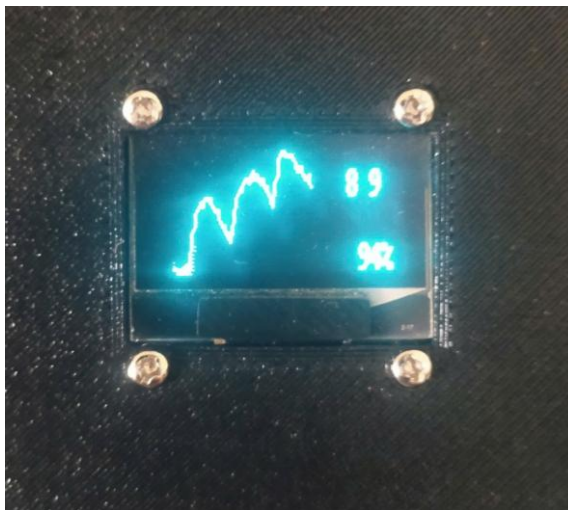


cá nhân mà còn mở ra khả năng ứng dụng rộng rãi trong các môi trường y tế chuyên nghiệp. Các thông số được cập nhật liên tục, đảm bảo người giám sát có thể theo dõi tình trạng sức khỏe của bệnh nhân một cách chính xác và kịp thời. Cung cấp thông tin chi tiết hơn về đối tượng sử dụng thiết bị, giúp các bác sĩ hoặc người giám sát nhanh chóng nhận biết người dùng cụ thể.

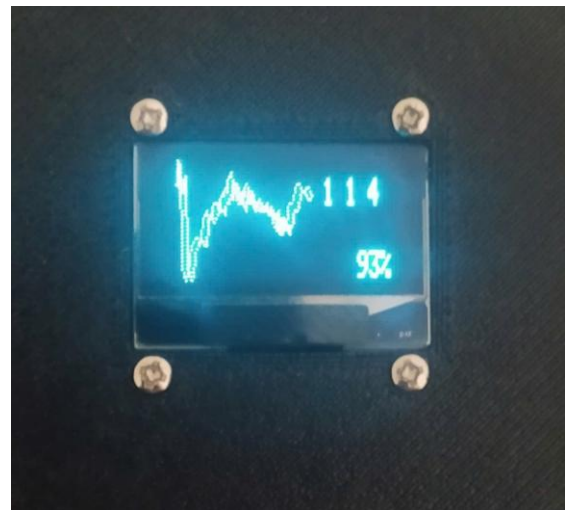
#### 4.1.4. Thử nghiệm trên người với lúc hoạt động bình thường và lúc hoạt động mạnh

Ở người trưởng thành không có các bệnh liên quan đến tim mạch nhịp tim thường dao động từ 60 – 100 lần/phút. Nếu tình trạng nhịp tim lên hơn 100 trong thời gian ngắn chẳng hạn như khi đang chơi thể thao hay khi tinh thần căng thẳng thì thường không nguy hiểm.

Do vậy, em đã có thử nghiệm thực tế với thiết bị của mình với việc so sánh nhịp tim người trưởng thành lúc hoạt động bình thường và so với lúc hoạt động mạnh.



(a) Lúc hoạt động bình thường



(b) Lúc hoạt động mạnh

*Hình 4. 8. So sánh nhịp tim giữa lúc hoạt động bình thường và hoạt động mạnh*

Qua đây có thể thấy thiết bị hoạt động khá chính xác khi thử nghiệm trên người trưởng thành với hai trạng thái khác nhau là hoạt động bình thường và hoạt động mạnh dựa trên thực tế khoa học.

#### 4.1.5. Thử nghiệm trên nhiều độ tuổi khác nhau

Với mỗi độ tuổi khác nhau sẽ có sự khác biệt đáng kể về nhịp tim và mức độ bão hòa oxy trong máu ( $SpO_2$ ). Dưới đây là bảng so sánh nhịp tim giữa các độ tuổi khác nhau em tham khảo được qua các trang báo y tế uy tín.

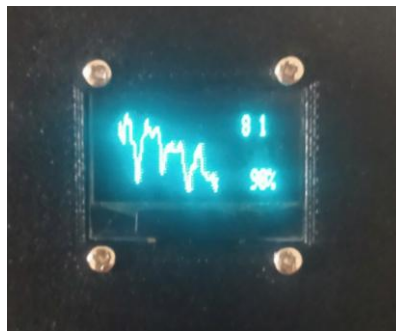
Độ tuổi	Tiêu chuẩn nhịp tim (nhịp/phút)
Trẻ sơ sinh	120 – 160
Trẻ từ 1-12 tháng	80 – 140
Trẻ từ 1 đến 2 tuổi	80 – 130
Trẻ từ 2 đến 6 tuổi	75- 120
Trẻ từ 7 đến 12 tuổi	75 - 110
Người lớn từ 18 đến 50 tuổi	60-100
Từ 50 tuổi trở lên	50 - 80

*Bảng 4. 1. Bảng so sánh nhịp tim giữa các độ tuổi*

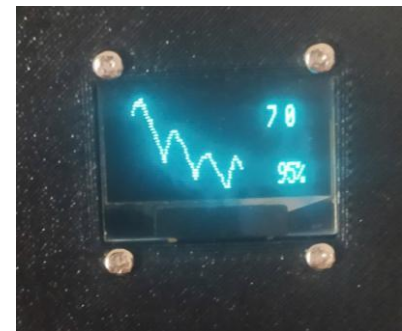
Ở đề tài lần này em lựa chọn 3 nhóm đối tượng để thử nghiệm chính trong đó nhóm đối tượng đầu tiên là nhóm trẻ từ 2 đến 6 tuổi, nhóm đối tượng thứ hai là nhóm người trưởng thành từ 18 đến 50 tuổi. Nhóm cuối cùng là nhóm từ 50 tuổi trở lên để tiến hành thử nghiệm thiết bị.



(a) Nhóm trẻ nhỏ



(b) Nhóm người lớn



(c) Nhóm từ 50 tuổi trở lên

*Hình 4. 9. So sánh nhịp tim giữa các độ tuổi*

Qua các kết quả này, có thể thấy thiết bị đo nhịp tim và SpO2 hoạt động chính xác và ổn định trên nhiều nhóm đối tượng với các thông số khác biệt dựa trên đặc điểm sinh lý của từng độ tuổi. Đây là một yếu tố quan trọng, cho thấy tính ứng dụng cao của thiết bị trong việc theo dõi sức khỏe của người dùng ở các giai đoạn khác nhau trong cuộc đời.

## 4.2. Đánh giá kết quả nhận được

### • Ưu điểm:

- Thuận tiện: Sự thuận tiện của hệ thống là một trong những ưu điểm lớn nhất. Khi mà có thể theo dõi từ xa thông qua Internet với sự phát triển của các ứng dụng của IoT.
- Dễ sử dụng: Giao diện người dùng được thiết kế một cách đơn giản và trực quan.
- Hoạt động ổn định: Hệ thống hoạt động ổn định, đảm bảo tính chính xác và đáng tin cậy trong việc ghi nhận sự hiện diện của sinh viên.

- Nhanh chóng: Việc sử dụng cảm biến Max30102 để đọc giá trị nhịp tim và nồng độ Oxy trong máu trở nên dễ dàng thông qua chuẩn giao tiếp I2C.
- **Nhược điểm:**
  - Phụ thuộc vào công nghệ: Hệ thống phụ thuộc nhiều vào công nghệ MQTT, điều này đồng nghĩa với việc cần có một hạ tầng kỹ thuật và thiết bị vững chắc. Nếu có sự cố với hạ tầng kỹ thuật, hệ thống có thể gặp khó khăn trong việc hoạt động.
  - Bảo mật: Vì hiện tại em đang dùng server MQTT public nên rủi ro bảo mật kênh đang sử dụng khá cao
  - Ngoài ra, em cũng nhận thấy rằng thiết kế của sản phẩm chưa đảm bảo khả năng chống nước và chống va đập đúng mức. Nếu sản phẩm rơi vào nước, có khả năng cao rằng nó sẽ bị hỏng.

### 4.3. Kết luận chương 4

Sau khi tập trung vào việc thử nghiệm và đánh giá thiết bị theo dõi nhịp tim và nồng độ Oxy trong máu. Từ các thử nghiệm đã tiến hành, ta đã có cái nhìn rõ hơn về hiệu suất và hạn chế của thiết bị trong môi trường thực tế.

Bên cạnh đó, trong quá trình thiết kế, em đã chú trọng đặc biệt đến tính khả dụng và tính thân thiện với người dùng. Giao diện hệ thống được thiết kế sao cho trực quan, dễ sử dụng, phù hợp với đối tượng người dùng cuối. Đồng thời, em cũng quan tâm đến tính thẩm mỹ, độ bền và khả năng mở rộng của hệ thống để bảo đảm rằng sản phẩm có thể được ứng dụng lâu dài, phù hợp với các yêu cầu trong thực tiễn.

Qua đây, có thể thấy rằng việc làm ra một sản phẩm hoàn chỉnh có thể ứng dụng thực tiễn không phải là điều dễ dàng. Đây không chỉ là một quá trình sáng tạo và đổi mới mà còn là cơ hội để áp dụng kiến thức, rèn luyện kỹ năng và mở rộng hiểu biết về lĩnh vực thiết kế và sản xuất thiết bị. Kết quả cuối cùng không chỉ là một sản phẩm, mà còn là hành trang kinh nghiệm và sự tự tin để tiếp tục phát triển các giải pháp công nghệ hữu ích trong tương lai.

## CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN ĐỀ TÀI

### 5.1 Kết quả đạt được

Qua quá trình thực hiện đồ, em đã đạt được một số kết quả như sau:

- Xây dựng được sản phẩm cuối cùng là hệ thống giám sát nhịp tim và nồng độ Oxy trong máu đáp ứng được các chức năng cơ bản của một thiết bị IoT cần có.
- Hiện thị đầy đủ phần giao diện người dùng thuận tiện và nhanh chóng.
- Hiểu biết thêm về ứng dụng web công nghệ và App Mobile vào trong một sản phẩm IoT cơ bản.
- Hiểu biết về ngôn ngữ lập trình C/C++, Java trong lập trình nhúng, IoT cho các thiết bị hiện đại cơ bản.
- Hiểu thêm một số ngôn ngữ như HTML5, CSS3, JQUERY, AJAX, ...
- Nâng cao khả năng lập trình, cũng như được biết thêm một số công nghệ mới.

### 5.2. Hạn chế

- Do lượng kiến thức khá lớn nên việc tìm hiểu triển khai tính năng vẫn còn nhiều khó khăn, thiếu sót.
- Code còn dài và chưa được tối ưu hóa
- Giao diện còn khá đơn giản.
- Các chức năng của website còn hạn chế, một số chức năng vẫn còn gặp lỗi.
- Một vài thành phần có thời gian xử lý lâu chưa được tối ưu và tính bảo mật cũng chưa đảm bảo nhiều.

### 5.3 Định hướng phát triển

- Phát triển thêm các tính năng như lúc người dùng không sử dụng thì màn hình sẽ hiển thị thêm ngày giờ.
- Kết hợp thêm các cảm biến: Nhiệt độ, độ ẩm nhằm tận dụng các chân còn lại của vi điều khiển đồng thời tăng khả năng giám sát của mô hình.
- Sử dụng AI để phân tích và đánh giá dữ liệu nhịp tim và nồng độ Oxy trong máu gửi về App Mobile, từ đó đưa ra dự đoán về sức khỏe người dùng.
- Phát triển ứng dụng di động: Nếu sản phẩm được ứng dụng rộng rãi cần cấp tài khoản cho người dùng tương ứng với 1 id máy để theo dõi được chuẩn xác.

## PHỤ LỤC

### PHỤ LỤC 1: Code điều khiển hệ thống

```
#if CONFIG_FREERTOS_UNICORE
#define ARDUINO_RUNNING_CORE 0
#else
#define ARDUINO_RUNNING_CORE 1
#endif
#define ESP_DRD_USE_SPIFFS true
#include <WiFi.h>
#include <FS.h>
#include <SPIFFS.h>
#include <WiFiManager.h>
#include <ArduinoJson.h>
#include "ssd1306h.h"
#include "MAX30102.h"
#include "Pulse.h"
#include <PubSubClient.h>
const char* mqttServer = "mqtt.eclipseprojects.io";/"broker.hivemq.com";
const int mqttPort = 1883;
const char *topic_publish = "esp32_Max30102_020511";
const char *ssid_sta = "102_020511";
const char *pass_sta = "1234567890";
const char *id_thietbi = "TEN THIET BI: 020511";
const char *id_staWifi = "ssid wifi: 102_020511";
// WiFi and MQTT clients
WiFiClient wifiClient;
PubSubClient mqttClient(wifiClient);
#define JSON_CONFIG_FILE "/test_config.json"
#define ESP_DRD_USE_SPIFFS true
bool shouldSaveConfig = false;
WiFiManager wm;
char ssid[32];
char pass[64];
// Routines to clear and set bits
#ifndef cbi
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#endif
#ifndef sbi
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))
#endif
SSD1306 oled;
MAX30102 sensor;
Pulse pulseIR;
Pulse pulseRed;
MAFilter bpm;
```

```

TaskHandle_t TaskHandle_WiFi;
TaskHandle_t TaskHandle_MQTT;
////////////////////////////////////
#define AVERAGE_INTERVAL 10000 // Thời gian tính trung bình: 10 giây
unsigned long lastAverageTime = 0;
int sumBPM = 0;    // Tổng BPM
int sumSPO2 = 0;   // Tổng SPO2
int countSamples = 0; // Đếm số lượng mẫu
int averagedBPM = 0; // BPM trung bình
int averagedSPO2 = 0; // SPO2 trung bình
////////////////////////////////////
#define OPTIONS 7
// Vẽ hình trái tim
static const uint8_t heart_bits[] PROGMEM = { 0x00, 0x00, 0x38, 0x38, 0x7c,
0x7c, 0xfe, 0xfe, 0xfe, 0xff,
                                0xfe, 0xff, 0xfc, 0x7f, 0xf8, 0x3f, 0xf0, 0x1f, 0xe0, 0x0f,
                                0xc0, 0x07, 0x80, 0x03, 0x00, 0x01, 0x00, 0x00, 0x00,
0x00,
                                0x00, 0x00 };
//Bảng spo2 có giá trị gần đúng -45.060*ratioAverage* ratioAverage + 30.354
*ratioAverage + 94.845 ;
const uint8_t spo2_table[184] PROGMEM =
    { 95, 95, 95, 96, 96, 96, 97, 97, 97, 97, 97, 98, 98, 98, 98, 98, 99, 99, 99, 99,
      99, 99, 99, 99, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100,
100, 100, 100, 100,
      100, 100, 100, 100, 99, 99, 99, 99, 99, 99, 99, 99, 98, 98, 98, 98, 98, 97,
97,
      97, 97, 96, 96, 96, 96, 95, 95, 95, 94, 94, 94, 93, 93, 93, 92, 92, 92, 91, 91,
      90, 90, 89, 89, 89, 88, 88, 87, 87, 86, 86, 85, 85, 84, 84, 83, 82, 82, 81, 81,
      80, 80, 79, 78, 78, 77, 76, 76, 75, 74, 74, 73, 72, 72, 71, 70, 69, 69, 68, 67,
      66, 66, 65, 64, 63, 62, 62, 61, 60, 59, 58, 57, 56, 56, 55, 54, 53, 52, 51, 50,
      49, 48, 47, 46, 45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33, 31, 30, 29,
      28, 27, 26, 25, 23, 22, 21, 20, 19, 17, 16, 15, 14, 12, 11, 10, 9, 7, 6, 5,
      3, 2, 1 } ;
void print_digit(int x, int y, long val, char c=' ', uint8_t field = 3, const int BIG = 2)
{
    uint8_t ff = field;
    do {
        char ch = (val!=0) ? val%10+'0': c;
        oled.drawChar( x+BIG*(ff-1)*6, y, ch, BIG);
        val = val/10;
        --ff;
    } while (ff>0);
}

```

```

/*
 * Ghi, chia tỉ lệ và vẽ tần số xung nhịp PPG
 */
const uint8_t MAXWAVE = 72;

class Waveform {
public:
    Waveform(void) { wavep = 0;}

    void record(int waveval) {
        waveval = waveval/8;
        waveval += 128;
        waveval = waveval<0? 0 : waveval;
        waveform[wavep] = (uint8_t) (waveval>255)?255:waveval;
        wavep = (wavep+1) % MAXWAVE;
    }

    void scale() {
        uint8_t maxw = 0;
        uint8_t minw = 255;
        for (int i=0; i<MAXWAVE; i++) {
            maxw = waveform[i]>maxw?waveform[i]:maxw;
            minw = waveform[i]<minw?waveform[i]:minw;
        }
        uint8_t scale8 = (maxw-minw)/4 + 1;
        uint8_t index = wavep;
        for (int i=0; i<MAXWAVE; i++) {
            disp_wave[i] = 31-((uint16_t)(waveform[index]-minw)*8)/scale8;
            index = (index + 1) % MAXWAVE;
        }
    }

    void draw(uint8_t X) {
        for (int i=0; i<MAXWAVE; i++) {
            uint8_t y = disp_wave[i];
            oled.drawPixel(X+i, y);
            if (i<MAXWAVE-1) {
                uint8_t nexty = disp_wave[i+1];
                if (nexty>y) {
                    for (uint8_t iy = y+1; iy<nexty; ++iy)
                        oled.drawPixel(X+i, iy);
                }
                else if (nexty<y) {
                    for (uint8_t iy = nexty+1; iy<y; ++iy)
                        oled.drawPixel(X+i, iy);
                }
            }
        }
    }
}

```

```

    }
    }
}
private:
    uint8_t waveform[MAXWAVE];
    uint8_t disp_wave[MAXWAVE];
    uint8_t wavep = 0;

} wave;

int beatAvg;
int SPO2, SPO2f;

bool filter_for_graph = false;
bool draw_Red = false;

uint8_t istate = 0;
uint8_t sleep_counter = 0;

void draw_oled(int msg) {
    oled.firstPage();
    do{
        switch(msg){
            case 0: oled.drawStr(10,0,F("Device error"),1);
                    break;
            case 1: oled.drawStr(6,10,F("FINGER"),2);
                    oled.drawXBMP(84,0,16,16,heart_bits);
                    oled.drawXBMP(108,0,16,16,heart_bits);
                    oled.drawStr(84,14,F("Display"),1);
                    oled.drawStr(84,24,F("BPM O2"),1);
                    break;
            case 2:
                    // print_digit(86,0,beatAvg);
                    print_digit(86, 5, averagedBPM, ' ', 3, 2); // Kích thước chữ lớn hơn
                    wave.draw(8);
                    //print_digit(98,16,SPO2f,' ',3,1);
                    //oled.drawChar(116,16,'%');
                    print_digit(98,24, averagedSPO2,' ',3,1);
                    oled.drawChar(116,24,'% ',1,1);
                    break;
            case 3: oled.drawStr(2,6,F(id_thietbi),1);
                    oled.drawStr(0,18,F(id_staWifi),1);
                    break;
        }
    } while (oled.nextPage());
}

```



```
void setup(void) {
    Serial.begin(115200); // Khởi động Serial với tốc độ 115200 bps
    oled.init();
    oled.fill(0x00);
    draw_oled(3);
    delay(3000);
    if (!sensor.begin()) {
        draw_oled(0);
        while (1);
    }
    sensor.setup();

    xTaskCreate(
        Task_initWiFi,
        "WiFi Task", // Name for humans
        8192, // Stack size in bytes
        NULL,
        1, // Priority
        &TaskHandle_WiFi
    );

    xTaskCreate(
        Task_connectMqtt,
        "mqtt task", // Name for humans
        2048, // Stack size in bytes
        NULL,
        1, // Priority
        &TaskHandle_MQTT
    );

}

long lastBeat = 0;
long displaytime = 0;

unsigned long previousMillis_WiFi = 0;

void loop() {
    // Thời gian hiện tại
    long now = millis();
    // Đo và hiển thị dữ liệu cảm biến
    sensor.check();
    if (!sensor.available()) return;

    uint32_t irValue = sensor.getIR();
```

```

uint32_t redValue = sensor.getRed();
sensor.nextSample();

// Kiểm tra nếu không phát hiện tay
if (irValue < 5000) {
    draw_oled(sleep_counter <= 50 ? 1 : 3);
    delay(200);
    ++sleep_counter;
    if (sleep_counter > 100) {
        sleep_counter = 0;
    }

    // Nếu không phát hiện tay, trả giá trị BPM và SPO2 về 0 và gửi nếu có kết nối
    // WiFi và MQTT
    if (mqttClient.connected() && WiFi.status() == WL_CONNECTED && now
- previousMillis_WiFi >= 5000) {
        String payload = String("{\"BPM\":0,\"SPO2\":0}");
        mqttClient.publish(topic_publish, payload.c_str());
        Serial.println("Data sent: " + payload);
        previousMillis_WiFi = now;
    }

    return; // Kết thúc hàm loop, không thực hiện phần còn lại
} else {
    sleep_counter = 0;
    int16_t IR_signal, Red_signal;
    bool beatRed, beatIR;

    if (!filter_for_graph) {
        IR_signal = pulseIR.dc_filter(irValue);
        Red_signal = pulseRed.dc_filter(redValue);
        beatRed = pulseRed.isBeat(pulseRed.ma_filter(Red_signal));
        beatIR = pulseIR.isBeat(pulseIR.ma_filter(IR_signal));
    } else {
        IR_signal = pulseIR.ma_filter(pulseIR.dc_filter(irValue));
        Red_signal = pulseRed.ma_filter(pulseRed.dc_filter(redValue));
        beatRed = pulseRed.isBeat(Red_signal);
        beatIR = pulseIR.isBeat(IR_signal);
    }

    wave.record(draw_Red ? -Red_signal : -IR_signal);

    if (draw_Red ? beatRed : beatIR) {
        long btpm = 60000 / (now - lastBeat);
        if (btpm > 0 && btpm < 200) beatAvg = bpm.filter((int16_t)btpm);
        lastBeat = now;
    }
}

```

```

// Tính tỉ lệ SPO2
long numerator = (pulseRed.avgAC() * pulseIR.avgDC()) / 256;
long denominator = (pulseRed.avgDC() * pulseIR.avgAC()) / 256;
int RX100 = (denominator > 0) ? (numerator * 100) / denominator : 999;

// Công thức
SPO2f = (10400 - RX100 * 17 + 50) / 100;

// from table
if ((RX100 >= 0) && (RX100 < 184)) {
    SPO2 = pgm_read_byte_near(&spo2_table[RX100]);
}

// Lưu dữ liệu BPM và SPO2 để tính trung bình
sumBPM += beatAvg;
sumSPO2 += SPO2;
countSamples++;

Serial.print("BPM: ");
Serial.println(btpm);
Serial.print("SPO2: ");
Serial.println(SPO2);
}

// Cập nhật giá trị trung bình mỗi AVERAGE_INTERVAL giây
if (now - lastAverageTime >= AVERAGE_INTERVAL) {
    if (countSamples > 0) {
        averagedBPM = sumBPM / countSamples;
        averagedSPO2 = sumSPO2 / countSamples;
        if(averagedBPM > 20 && averagedBPM < 50){
            averagedBPM = 50;
        }
        if( averagedSPO2 > 10 && averagedSPO2 < 85){
            averagedSPO2 = 90;
        }
    }

    // Hiển thị giá trị trung bình
    Serial.print("Averaged BPM: ");
    Serial.println(averagedBPM);
    Serial.print("Averaged SPO2: ");
    Serial.println(averagedSPO2);
    // Reset bộ đếm
    sumBPM = 0;
    sumSPO2 = 0;
    countSamples = 0;
}

```

```

        lastAverageTime = now;
    }

}

if (now - displaytime > 50) {
    displaytime = now;
    wave.scale();
    draw_oled(2);
}

// Gửi dữ liệu lên MQTT nếu có kết nối và mỗi 1000 ms
if (mqttClient.connected() && WiFi.status() == WL_CONNECTED && now
- previousMillis_WiFi >= 5000) {
    String payload = String("{\"BPM\":") + String(averagedBPM) +
String(",\"SPO2\":") + String(averagedSPO2) + String("}");
    mqttClient.publish(topic_publish , payload.c_str());
    Serial.println("Data sent: " + payload);
    previousMillis_WiFi = now;
}
}
}

// task connect mqtt
void Task_connectMqtt(void *pvParameters)
{
    mqttClient.setServer(mqttServer, mqttPort);
    for(;;)
    {
        if (!mqttClient.connected()) {
            reconnect_mqtt();
        }
        mqttClient.loop();
        vTaskDelay(1000/portTICK_PERIOD_MS);
    }
}

// reconnect mqtt
void reconnect_mqtt() {
    Serial.println("Connecting to MQTT Broker...");
    while (!mqttClient.connected()) {
        Serial.println("Attempting MQTT connection...");
        String clientID = "ESP32Client-";
        clientID += String(random(0xffff), HEX);
        if (mqttClient.connect(clientID.c_str())) {
            Serial.println("Connected to MQTT Broker");
        } else {

```

```

        Serial.print("Failed, rc=");
        Serial.print(mqttClient.state());
        Serial.println(" try again in 2 seconds");
        vTaskDelay(2000/portTICK_PERIOD_MS);
    }
}
}

// ket noi wifi
void saveConfigFile() {
    Serial.println(F("Saving configuration..."));
    StaticJsonDocument<512> json;

    File configFile = SPIFFS.open(JSON_CONFIG_FILE, "w");
    if (!configFile) {
        Serial.println("Failed to open config file for writing");
    }

    serializeJsonPretty(json, Serial);
    if (serializeJson(json, configFile) == 0) {
        Serial.println(F("Failed to write to file"));
    }
    configFile.close();
}

bool loadConfigFile() {
    Serial.println("Mounting File System...");
    if (SPIFFS.begin(false) || SPIFFS.begin(true)) {
        Serial.println("Mounted file system");
        if (SPIFFS.exists(JSON_CONFIG_FILE)) {
            Serial.println("Reading config file");
            File configFile = SPIFFS.open(JSON_CONFIG_FILE, "r");
            if (configFile) {
                Serial.println("Opened configuration file");
                StaticJsonDocument<512> json;
                DeserializationError error = deserializeJson(json, configFile);
                serializeJsonPretty(json, Serial);
                if (!error) {
                    Serial.println("Parsing JSON");
                    const char* savedSsid = json["ssid"];
                    const char* savedPass = json["pass"];
                    if (savedSsid && savedPass) {
                        strncpy(ssid, savedSsid, sizeof(ssid));
                        strncpy(pass, savedPass, sizeof(pass));
                    }
                    return true;
                } else {

```

```

        Serial.println("Failed to load JSON config");
    }
}
} else {
    Serial.println("Failed to mount FS");
}
return false;
}
void saveConfigCallback() {
    Serial.println("Should save config");
    shouldSaveConfig = true;
}
void configModeCallback(WiFiManager *myWiFiManager) {
    Serial.println("Entered Configuration Mode");
    Serial.print("Config SSID: ");
    Serial.println(myWiFiManager->getConfigPortalSSID());
    Serial.print("Config IP Address: ");
    Serial.println(WiFi.softAPIP());
}
// Task inti WiFi
void Task_initWiFi(void *pvParameters){
    bool forceConfig = false;
    bool spiffsSetup = loadConfigFile();
    if (!spiffsSetup) {
        Serial.println(F("Forcing config mode as there is no saved config"));
        forceConfig = true;
    }
    // sau khi vao che do STA thi se dung task mqtt
    vTaskSuspend(TaskHandle_MQTT);
    WiFi.mode(WIFI_STA);
    // Uncomment to clear saved WiFi settings
    //wm.resetSettings();

    wm.setSaveConfigCallback(saveConfigCallback);
    wm.setAPCallback(configModeCallback);
    if (forceConfig) {
        if (!wm.startConfigPortal(ssid_sta, pass_sta)) {
            Serial.println("Failed to connect and hit timeout");
        }
    } else {
        if (!wm.autoConnect(ssid_sta, pass_sta)) {
            Serial.println("Failed to connect and hit timeout");
        }
    }
}

```

```

Serial.println("");
Serial.println("WiFi connected");
Serial.print("IP address: ");
Serial.println(WiFi.localIP());

strncpy(ssid, wm.getWiFiSSID().c_str(), sizeof(ssid));
strncpy(pass, wm.getWiFiPass().c_str(), sizeof(pass));
Serial.print("SSID: ");
Serial.println(ssid);
Serial.print("Password: ");
Serial.println(pass);

if (shouldSaveConfig) {
    saveConfigFile();
}

bool cnt = true;
// // ket noi WiFi thanh cong se mo lai task mqtt
if(WiFi.status() == WL_CONNECTED)
{
    vTaskResume(TaskHandle_MQTT);
}
for(;;)
{
    if(WiFi.status() != WL_CONNECTED && cnt == true)
    {
        reconnectWiFi();
    }
    vTaskDelay(1000 / portTICK_PERIOD_MS);
}
}
// reconnect WiFi
void reconnectWiFi(){
    Serial.println("WiFi lost connection. Attempting to reconnect...");
    WiFi.begin(ssid, pass);
    vTaskDelay(3000 / portTICK_PERIOD_MS);
    while (WiFi.status() != WL_CONNECTED) {
        vTaskDelay(1000 / portTICK_PERIOD_MS);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("Reconnected to WiFi");
    Serial.print("IP address: ");
    Serial.println(WiFi.localIP());
}

```

## PHỤ LỤC 2: Code App Giao diện tổng quan

```

public class MainActivity extends AppCompatActivity {
    // thiết bị 020510
    private TextView txtBPM_020510, txtSPO2_020510;
    private TextView txtMaNguoiBenh_1, txtTenNguoiBenh_1;
    private ImageView imgAppStatus, imgStatus_020510;
    private RelativeLayout rlt020510;
    // thiết bị 020511
    private TextView txtBPM_020511, txtSPO2_020511;
    private TextView txtMaNguoiBenh_2, txtTenNguoiBenh_2;
    private ImageView imgStatus_020511;
    private RelativeLayout rlt020511;
    // khai báo kết nối MQTT
    MqttAndroidClient mqttAndroidClient;
    private static final String Tag = "";
    String clientId = MqttClient.generateClientId();
    MqttAndroidClient client;
    // handle kiểm tra thiết bị có kết nối tới mqtt hay không
    private Handler handler_mqtt;
    private Runnable checkConnectionRunnable_020510;
    private Runnable checkConnectionRunnable_020511;
    // lưu giá trị mã người bệnh và tên người bệnh
    private SharedPreferences sharedPreferences;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        AnhXa();
        // tạo handler để biết thiết bị kết nối với mqtt hay không
        handler_mqtt = new Handler();
        //kết nối MQTT
        MQTT();
        imgAppStatus.setOnClickListener(v -> MQTT());
        private void AnhXa() {
            // Ánh xạ thiết bị 1
            txtBPM_020510 = findViewById(R.id.txtBPM_020510);
            txtSPO2_020510 = findViewById(R.id.txtSPO2_020510);
            txtMaNguoiBenh_1 = findViewById(R.id.txtMaNguoiBenh_1);
            txtTenNguoiBenh_1 = findViewById(R.id.txtTenNguoiBenh_1);
            imgAppStatus = findViewById(R.id.imgAppStatus);
            imgStatus_020510 = findViewById(R.id.imgStatus_020510);
            rlt020510 = findViewById(R.id.rlt020510);

            // Ánh xạ thiết bị 2
            txtBPM_020511 = findViewById(R.id.txtBPM_020511);
            txtSPO2_020511 = findViewById(R.id.txtSPO2_020511);

```



```

        txtMaNguoiBenh_2 = findViewById(R.id.txtMaNguoiBenh_2);
        txtTenNguoiBenh_2 = findViewById(R.id.txtTenNguoiBenh_2);
        imgStatus_020511 = findViewById(R.id.imgStatus_020511);
        rlt020511 = findViewById(R.id.rlt020511);
    }
    // kết nối mqtt
    public void MQTT() {
        String clientID = MqttClient.generateClientId();
        final MqttAndroidClient client = new
MqttAndroidClient(this.getApplicationContext(),
        "tcp://mqtt.eclipseprojects.io:1883",
clientID);/"tcp://broker.hivemq.com:1883"
        MqttConnectOptions options = new MqttConnectOptions();
        options.setMqttVersion(MqttConnectOptions.MQTT_VERSION_3_1);
        options.setCleanSession(false);
        // Remove these lines if not required
        // options.setUsername("");
        // options.setPassword("").toCharArray());
        try {
            final IMqttToken token = client.connect(options);
            token.setActionCallback(new IMqttActionListener() {
                @Override
                public void onSuccess(IMqttToken asyncActionToken) {
                    Toast.makeText(MainActivity.this, "Connected MQTT server",
Toast.LENGTH_SHORT).show();
                    imgAppStatus.setBackgroundResource(R.drawable.connect);
                    SUB(client, "esp32_Max30102_020510"); // thiết bị 020510
                    SUB(client, "esp32_Max30102_020511"); // thiết bị 020511
                    client.setCallback(new MqttCallback() {
                        @Override
                        public void connectionLost(Throwable cause) {
                            Toast.makeText(MainActivity.this, "Mat ket noi Server",
Toast.LENGTH_SHORT).show();
                            imgAppStatus.setBackgroundResource(R.drawable.disconnect);
                        }

                        @Override
                        public void messageArrived(String topic, MqttMessage message)
throws Exception {
                            // lay du lieu BPM và SPO2 của thiết bị 020510 từ MQTT về
                            if(topic.equals("esp32_Max30102_020510"))
                            {
                                try {
                                    JSONObject jsonObject = new
JSONObject(message.toString());
                                    int bpm_020510 = jsonObject.getInt("BPM");

```

```

        int spo2_020510 = jsonObject.getInt("SPO2");

        // Cập nhật giá trị BPM và SpO2 lên TextView
        runOnUiThread() -> {
            txtBPM_020510.setText(String.valueOf(bpm_020510));
            txtSPO2_020510.setText(String.valueOf(spo2_020510 +
"%"));

            // Cập nhật trạng thái thiết bị 020510 khi có dữ liệu

imgStatus_020510.setBackgroundResource(R.drawable.point_connect);
        });
        // Reset thời gian chờ 5 giây mỗi khi nhận được dữ liệu
        resetConnectionTimeout_020510();
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
if(topic.equals("esp32_Max30102_020511"))
{
    try {
        JSONObject jsonObject = new
JSONObject(message.toString());
        int bpm_020511 = jsonObject.getInt("BPM");
        int spo2_020511 = jsonObject.getInt("SPO2");

        // Cập nhật giá trị BPM và SpO2 lên TextView
        runOnUiThread() -> {
            txtBPM_020511.setText(String.valueOf(bpm_020511));
            txtSPO2_020511.setText(String.valueOf(spo2_020511 +
"%"));

            // Cập nhật trạng thái thiết bị 020510 khi có dữ liệu

imgStatus_020511.setBackgroundResource(R.drawable.point_connect);
        });

        // Reset thời gian chờ 5 giây mỗi khi nhận được dữ liệu
        resetConnectionTimeout_020511();
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
}
@Override
public void deliveryComplete(IMqttDeliveryToken token) {
}
});

```

```

    }
    @Override
    public void onFailure(IMqttToken asyncActionToken, Throwable
exception) {
//        Toast.makeText(MainActivity.this, "that bai",
Toast.LENGTH_SHORT).show();
        Log.d(Tag, "onFailure");
        imgAppStatus.setBackgroundResource(R.drawable.disconnect);
    }
});
} catch (MqttException e) {
    e.printStackTrace();
}
} // end MQTT
// kiểm tra kết nối thiết bị 020510
private void resetConnectionTimeout_020510() {
    if (handler_mqtt != null && checkConnectionRunnable_020510 != null) {
        handler_mqtt.removeCallbacks(checkConnectionRunnable_020510);
    }

    // Tạo một Runnable mới để kiểm tra kết nối sau 5 giây
    checkConnectionRunnable_020510 = () -> {
        //set img point về disconnect
        imgStatus_020510.setBackgroundResource(R.drawable.point_disconnect);
        // Đặt lại giá trị của txtBPM_020510 và txtSPO2_020510 về "NaN"
        runOnUiThread() -> {
            txtBPM_020510.setText("NaN");
            txtSPO2_020510.setText("NaN");
        });
    };

    // Đặt lại thời gian chờ 15 giây
    handler_mqtt.postDelayed(checkConnectionRunnable_020510, 7000);
}
// kiểm tra kết nối thiết bị 020510 //

// kiểm tra kết nối thiết bị 020511
private void resetConnectionTimeout_020511() {
    if (handler_mqtt != null && checkConnectionRunnable_020511 != null) {
        handler_mqtt.removeCallbacks(checkConnectionRunnable_020511);
    }

    // Tạo một Runnable mới để kiểm tra kết nối sau 15 giây
    checkConnectionRunnable_020511 = () -> {
        //set img point về disconnect
        imgStatus_020511.setBackgroundResource(R.drawable.point_disconnect);
    };
}

```

```
// Đặt lại giá trị của txtBPM_020511 và txtSPO2_020511 về "NaN"
runOnUiThread() -> {
    txtBPM_020511.setText("NaN");
    txtSPO2_020511.setText("NaN");
});

// Đặt lại thời gian chờ 15 giây
handler_mqtt.postDelayed(checkConnectionRunnable_020511, 7000);
}
// kiểm tra kết nối thiết bị 020511 //

// cập nhật mã và tên bệnh nhân
private void MAandTenBN() {
    Intent intent = getIntent();

    String maNguoiBenh_1 = intent.getStringExtra("maNguoiBenh_1");
    String tenNguoiBenh_1 = intent.getStringExtra("tenNguoiBenh_1");

    String maNguoiBenh_2 = intent.getStringExtra("maNguoiBenh_2");
    String tenNguoiBenh_2 = intent.getStringExtra("tenNguoiBenh_2");

    // Hiển thị dữ liệu lên TextView
    txtMaNguoiBenh_1.setText(maNguoiBenh_1);
    txtTenNguoiBenh_1.setText(tenNguoiBenh_1);

    txtMaNguoiBenh_2.setText(maNguoiBenh_2);
    txtTenNguoiBenh_2.setText(tenNguoiBenh_2);

    // Tạo hoặc truy cập SharedPreferences
    sharedPreferences = getSharedPreferences("PatientData", MODE_PRIVATE);

    // Gọi hàm để tải dữ liệu từ SharedPreferences
    loadPatientData();

    if (maNguoiBenh_1 != null && tenNguoiBenh_1 != null) {
        // Cập nhật TextView với dữ liệu mới
        txtMaNguoiBenh_1.setText(maNguoiBenh_1);
        txtTenNguoiBenh_1.setText(tenNguoiBenh_1);

        // Gọi hàm để lưu dữ liệu vào SharedPreferences
        savePatientData_1(maNguoiBenh_1, tenNguoiBenh_1);
    }
}
```

## Tài liệu tham khảo

1. TS. Bùi Văn Minh, TS. Dương Thanh Long, KS. Phạm Quang Huy, Lập Trình Điều Khiển Xa Với ESP8266-ESP32 Và Arduino, NXB Thanh Niên, 2019.
2. <https://mecsuvn/ho-tro-ky-thuat/gioi-thieu-ve-bo-mach-phat-trien-esp32.KyJ>, Giới thiệu về bo mạch phát triển ESP32.
3. <https://khuenguyencreator.com/tong-quan-ve-so-do-chan-esp32-va-ngoai-vi/>, Tổng quan về sơ đồ chân ESP32 và ngoại vi.
4. <https://randomnerdtutorials.com/esp32-client-server-wi-fi/>, hướng dẫn kết nối WiFi cho ESP32.
5. <https://www.datasheethub.com/ssd1306-128x64-mono-0-96-inch-i2c-oled-display/>, datasheet Oled 0.96 inch.
6. <https://randomnerdtutorials.com/esp32-ssd1306-oled-display-arduino-ide/>, hướng dẫn kết nối màn hình Oled với ESP32.
7. <https://microcontrollerslab.com/esp32-heart-rate-pulse-oximeter-max30102/>, MAX30102 Pulse Oximeter and Heart Rate Sensor with ESP32.
8. <https://github.com/Probots-Techno-Solutions/Wifi-Oximeter-using-MAX30102-and-ESP32/blob/main/README.md>, Wifi Oximeter using MAX30102 and ESP32.10.
9. <https://www.hivemq.com/mqtt/public-mqtt-broker/>, hướng dẫn kết nối ESP với server MQTT HiveMQ
10. <https://randomnerdtutorials.com/esp32-http-get-post-arduino/>, ESP32 HTTP GET and HTTP POST with Arduino IDE
11. <https://www.instructables.com/How-to-Build-a-DIY-WiFi-Smart-Oximeter-Using-MAX30/>,
12. <https://randomnerdtutorials.com/getting-started-with-esp32/>, Getting Started with the ESP32 Development Board
13. <https://www.electronicwings.com/esp32/introduction-to-esp32>, Introduction to ESP32
14. <https://randomnerdtutorials.com/esp32-websocket-server-arduino/>, ESP32 WebSocket Server: Control Outputs (Arduino IDE)
15. <https://randomnerdtutorials.com/esp32-mqtt-publish-subscribe-arduino-ide/>, ESP32 MQTT – Publish and Subscribe with Arduino IDE
16. <https://randomnerdtutorials.com/cloud-mqtt-mosquitto-broker-access-anywhere-digital-ocean/>, Run Your Cloud MQTT Mosquitto Broker
17. Datasheet ESP32 series, ESPRESSIF SYSTEMS, 2016.
18. Datasheet Max30102, maxim integrated products, 2021.
19. Datasheet Pin18650, Tenenergy Corporation, 2009.
20. Datasheet TP4056, Shenzhen TPOWER Semiconductor, 2010
21. <https://microlife.com.vn/nhip-tim-binh-thuong-cua-nguoi-truong-thanh-la-bao-nhieu/>, chỉ tiết về nhịp tim trong nghiên cứu y tế