TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
KHOA ĐIỆN-ĐIỆN TỬ
BỘ MÔN KỸ THUẬT ĐIỆN TỬ
http://dee.utc.edu.vn/

BÁO CÁO MÔN HỌC ĐỒ ÁN THIẾT KẾ

Đề tài : Thiết kế và thi công thiết bị theo dõi nhịp tim và nồng độ Oxy trong máu sử dụng ESP 32 kết nối với Mobile app

HỌ VÀ TÊN: Nguyễn Quang Minh MSSV: 201404024
Nguyễn Công Hoan MSSV: 191412043

LỚP: Điện tử và THCN1

HÀ NỘI, 2024

LỜI MỞ ĐẦU

Ngày nay kỹ thuật vi điều khiển đã trở nên quen thuộc trong các ngành kỹ thuật và trong dân dụng. 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, tôi đã thiết kế, chế tạo "**Thiết bị theo dõi nhịp tim và nồng độ Oxy trong máu sử dụng vi điều khiển esp32**"

Do thời gian, kiến thức và kinh nghiệm của tôi còn có hạn nên sẽ không thể tránh khỏi những sai sót. Tôi 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.

MỤC LỤC

THUẬT NGỮ VIẾT TẮT

STT	Từ viết tắt	Từ tiếng Anh	Nghĩa tiếng Việt
1	RFID	Radio Frequency Identification	Nhận dạng qua tần số vô tuyến
2	UART	Universal Asynchronous Receiver / Transmitter	Truyền dữ liệu nối tiếp bất đồng bộ
3	SPI	Serial Peripheral Interface	Chuẩn truyền thông nối tiếp
4	I2C	Inter - Intergrated Circuit	Giao tiếp giữa các IC
5	SCK	Serial Clock	Chân giữ xung nhịp trong giao tiếp SPI
6	MISO	Master Input Slave Output	Chân mang dữ liệu từ các thiết bị SPI về vi điều khiển
7	MOSI	Master Output Slave Input	Chân mang dữ liệu từ vi điều khiển đến các thiết bị SPI
8	SDA	Serial Data Line	Dây truyền dữ liệu
9	IoT	Internet of Things	Mạng lưới vạn vật kết nối Internet
10	LCD	Liquid Crystal Display	Màn hình tinh thể lỏng
11	UID	Unique Identifier	Mã định danh duy nhất
12	VDC	Volt Direct Current	Dòng điện một chiều
13	URLs	Uniform Resource Locators	Địa chỉ tài nguyên
14	SQL	Structured Query Language	Hệ cơ sở dữ liệu
15	GPIO	General Purpose Input / Output	Đầu vào/đầu ra thông dụng
16	GND	Ground	Đất, Mặt đất

DANH MỤC BẢNG BIỂU

DANH MỤC HÌNH ẢNH

CHƯƠNG I : TỔNG QUAN VỀ IOT VÀ MỤC TIÊU CỦA ĐỒ ÁN

1.1 Tổng quan về một dự án IOT

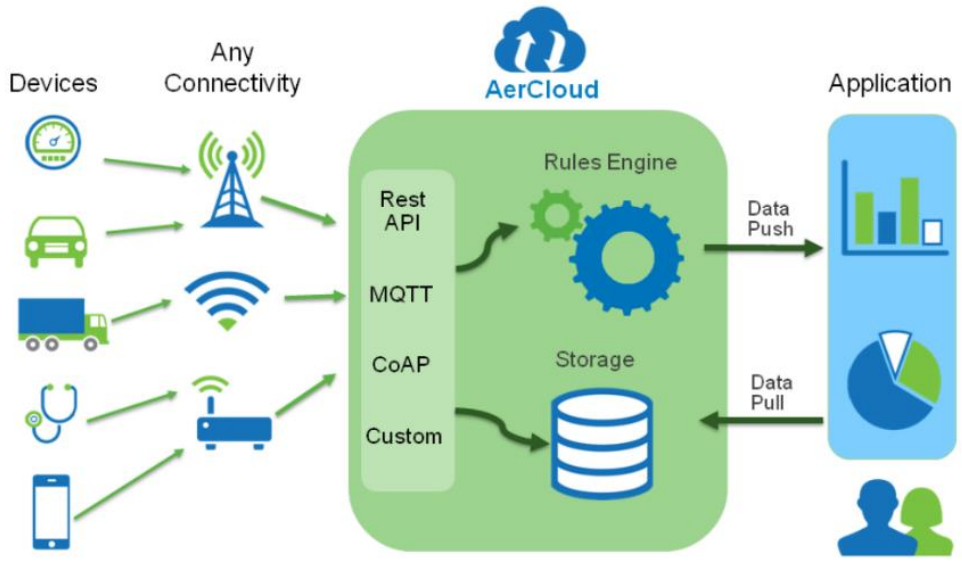
Internet vạn vật IoT là gì ?

IoT là viết tắt của **Internet of Things**, hay còn gọi là **Internet vạn vật**. Đây là một mạng lưới khổng lồ gồm các thiết bị vật lý được nhúng với các cảm biến, phần mềm, bộ truyền động và kết nối mạng cho phép chúng thu thập và trao đổi dữ liệu với nhau qua Internet. Các thiết bị IoT có thể bao gồm bất kỳ thứ gì từ các thiết bị gia dụng thông minh như tủ lạnh, máy giặt đến các thiết bị công nghiệp như cảm biến, máy móc, robot.

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.

Internet vạn vật tích hợp “vạn vật” với Internet mỗi ngày. Các kỹ sư máy tính đã và đang thêm các cảm biến và bộ xử lý vào các vật dụng hàng ngày kể từ những năm 90. Tuy nhiên, tiến độ ban đầu rất chậm vì các con chip còn to và cồng kềnh. Loại chip máy tính công suất thấp gọi là thẻ tag RFID, lần đầu tiên được sử dụng để theo dõi các thiết bị đắt đỏ. Khi kích cỡ của các thiết bị điện toán dần nhỏ lại, những con chip này cũng trở nên nhỏ hơn, nhanh hơn và thông minh hơn theo thời gian.

Cấu trúc của một dự án IoT:



Hình 1.1 Cấu trúc cơ bản của dự án IoT

Một hệ thống IoT thường bao gồm các thành phần sau:

+Thiết bị cảm biến: Là những thiết bị thu thập thông tin và gửi dữ liệu đến mạng lưới IoT. Chúng có thể là các cảm biến nhiệt độ, độ ẩm, ánh sáng, hình ảnh, âm thanh, độ rung, v.v.

+Thiết bị **Gateway:** Là thiết bị trung gian giữa các thiết bị cảm biến và mạng lưới IoT. Thiết bị này có chức năng thu thập và chuyển tiếp dữ liệu từ các thiết bị cảm biến lên mạng lưới IoT.

+Mạng lưới IoT : Là một mạng lưới kết nối các thiết bị thông minh với nhau và với internet. Các thiết bị có thể giao tiếp với nhau để thực hiện các tác vụ tự động.

+Trung tâm điều khiển: Là nơi tổng hợp, xử lý và quản lý dữ liệu từ các thiết bị cảm biến và điều khiển các thiết bị thông minh trong mạng lưới IoT. Nó có thể là một phần mềm hoặc phần cứng.

+Ứng dụng: Là các ứng dụng được phát triển để sử dụng các dữ liệu và thông tin thu thập được từ mạng lưới IoT. Các ứng dụng này có thể được sử dụng để giám sát và điều khiển các thiết bị trong mạng lưới, hoặc để phân tích và xử lý dữ liệu để cung cấp các thông tin hữu ích cho người dùng.

Một số dự án IoT em đã từng làm:

- Bãi đỗ ô tô thông minh: sử dụng cảm biến khoảng cách để phát hiện xe vào điểm đỗ hiển thị lên lcd, app blynk những vị trí trống và đã có ô tô sử dụng 2 vi điều khiển là Arduino UNO(điều khiển), ESP8266(gửi dữ liệu).
- Máy CNC: sử dụng Motor được điều khiển bởi Arduino UNO vẽ các hình vẽ mong muốn lên tấm mica.
- Máy đo nhiệt độ, độ ẩm : sử dụng cảm biến DHT11 để đo nhiệt độ, độ ẩm hiển thị lên màn hình Oled 0.96inch và app Blynk thông qua vi điều khiển ESP32.

Các lỗi thường gặp khi làm dự án IoT:

- Copy không chọn lọc của những người đã từng làm dự án trước đó có thể code sẽ không chạy, điều này khiến sinh viên thiếu sự sáng tạo, trì trệ trong suy nghĩ, nên tham khảo có chọn lọc và nâng cấp dự án của những người đã từng làm trước đó không nên lấy y hệt của người đã từng làm.
- Đấu nối phần cứng sai dẫn đến hỏng các linh kiện, để khắc phục cần có kiến thức điện tử, mạch điện để không dẫn đến hỏng mạch, cháy nổ ảnh hưởng đến tính mạng người làm.
- Không biết cách sửa lỗi code do đi copy không có tư duy học tập bài bản, chỉ có thể học lại các kiến thức căn bản đến chuyên sâu để khắc phục lỗi này.

1.2 Lý do thực hiện đồ án này. Vấn đề cần làm, tại sao phải làm?

- Với nhu cầu học tập, tìm hiểu của bản thân và cũng là góp phần sáng tạo của mình cho việc giảng dạy và học tập ở Trường Đại học Giao thông vận tải em quyết định làm dự án : Thiết bị theo dõi nhịp tim và nồng độ Oxy trong máu.
- 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.
- Khả năng hỗ trợ lập trình cho vi điều khiển Esp32 được phổ biến trong lĩnh vực lập trình nhúng do đó có nguồn tài liệu tham khảo vô cùng phong phú dễ tiếp cận, cộng đồng hỗ trợ đông đảo.

- Việc thực hiện một dự án về thiết bị theo dõi nhịp tim và nồng độ Oxy trong máu cung cấp một cơ hội để áp dụng kiến thức vào thực tế. Điều này có thể giúp sinh viên hiểu rõ hơn về cách áp dụng các nguyên lý kỹ thuật để giải quyết các vấn đề thực tế và cải thiện cuộc sống hàng ngày.

- Sinh viên có thể phát triển kỹ năng trong việc thiết kế, lập trình và xây dựng các hệ thống điện tử thông qua việc thực hiện dự án này. Điều này bao gồm việc làm việc với các linh kiện điện tử, vi điều khiển, cảm biến, và việc lập trình phần mềm điều khiển.

- 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.

Hình 1.2 Một số loại máy đo nhịp tim và nồng độ Oxy trong máu hiện có trên thị trường

1.3 Mục đích thực hiện đồ án này.

- Khi 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 và thực hiện đề tài này em đã mong muốn tạo ra một thiết bị sử dụng một cách dễ dàng, tiếp cận được với đại đa số người dùng. Sản phẩm đơn giản và sử dụng khá ít linh kiện sẽ giúp thực hiện dễ dàng và nhanh chóng.

- 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.

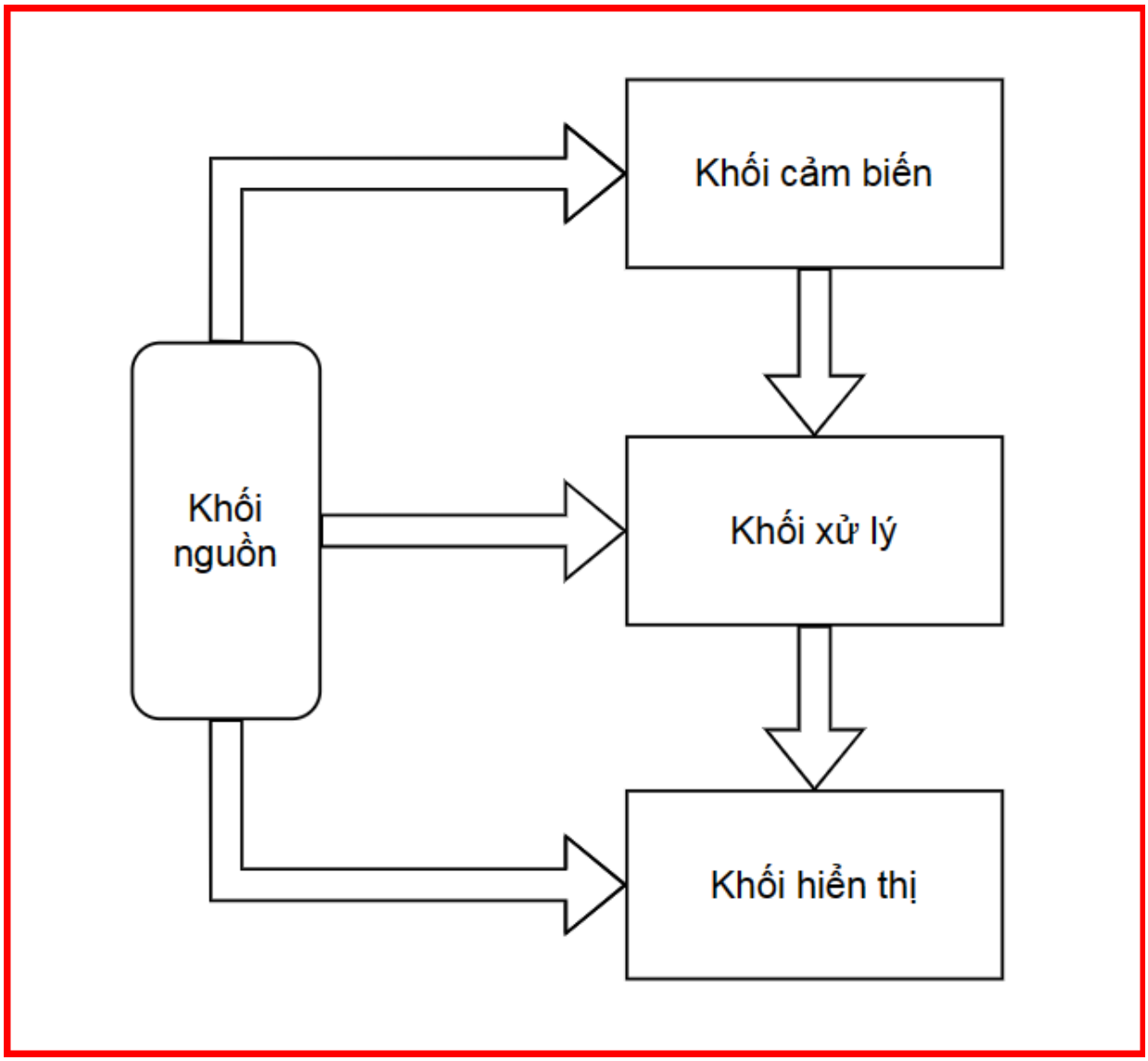
- Ngoài ra, dự án này còn có mục tiêu: Nâng cao kiến thức về các linh kiện điện tử và lập trình vi điều khiển, rèn luyện kỹ năng thiết kế, chế tạo và thử nghiệm thiết bị điện tử, phát triển khả năng sáng tạo và giải quyết vấn đề. Thiết bị có thể hoạt động độc lập hoặc kết nối với mạng Wi-Fi để gửi dữ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

1.4. Kết luận chương 1

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 đồ án nghiên cứu đã công bố, từ đó nêu ra được mục tiêu của đồ 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 II. THIẾT KẾ VÀ THI CÔNG MẠCH THEO DÕI NHỊP TIM VÀ NỒNG ĐỘ OXY TRONG MÁU

2.1Thiết kế tổng quát



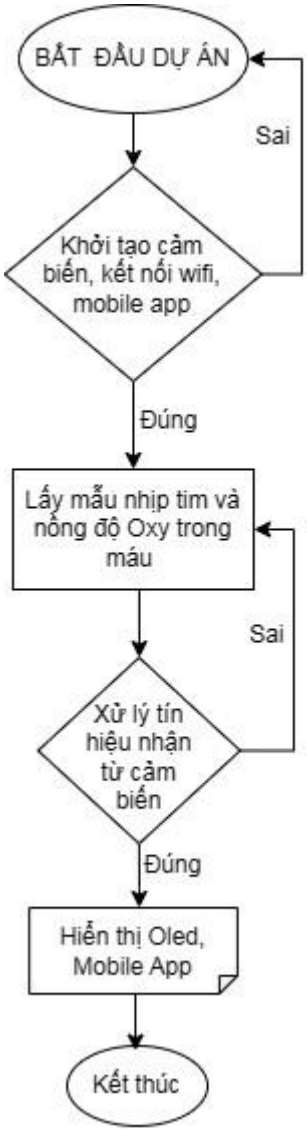
Hình 2.1 Sơ đồ khối của phần cứng điện tử

• Khối nguồn: Cung cấp năng lượng cho toàn bộ mô hình hoạt động.

• Khối hiển thị: Hiển thị số lượng sản phẩm đã đếm được.

• Khối cảm biến: Gồm các cảm biến hồng ngoại dùng để phát hiện vật, các cảm biến này sẽ được đặt ở các độ cao khác nhau tùy vào chiều cao muốn phân loại

• Khối xử lý trung tâm: Thu thập dữ liệu từ cảm biến, tính toán và đưa ra quyết định điều khiển khối hiển thị và khối chấp hành.



Hình 2.2 Sơ đồ thuật toán của thiết bị

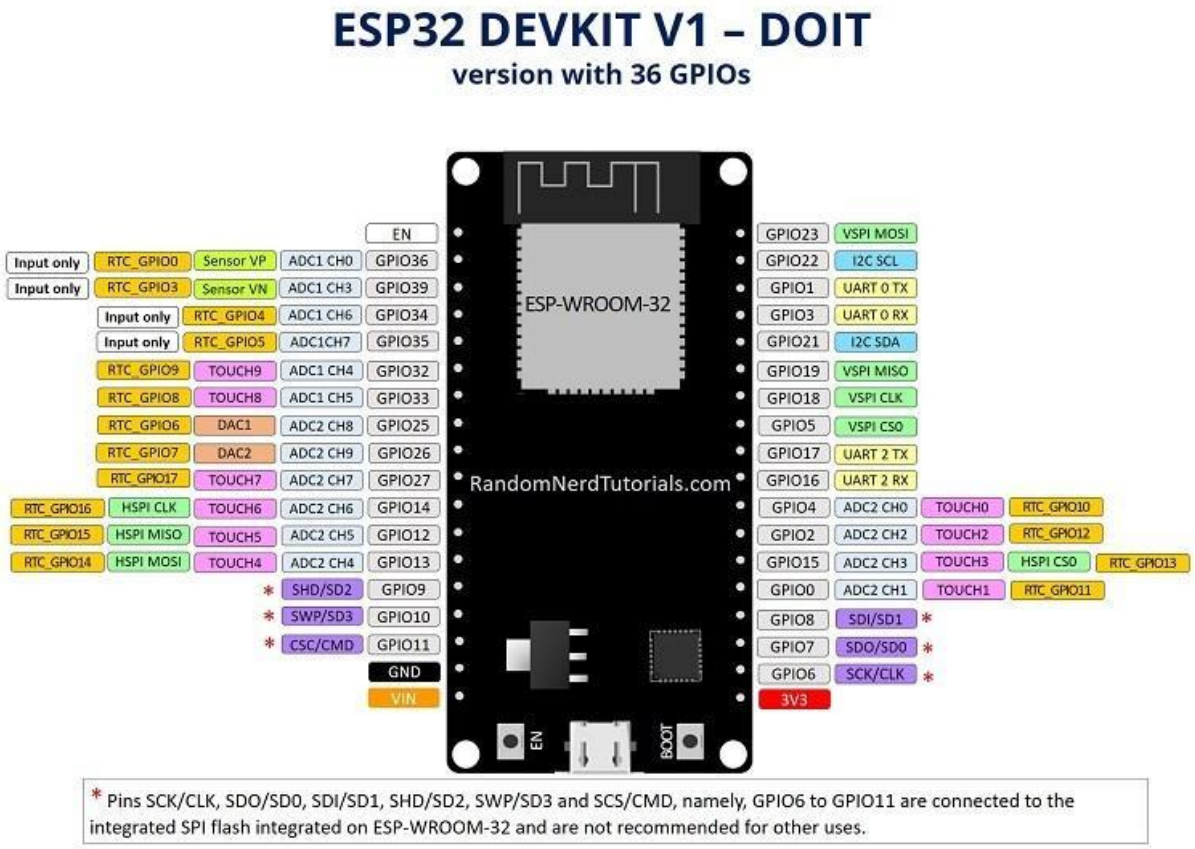
Bảng 2.1 Linh kiện sử dụng trong thiết này.

Linh kiện	Số lượng	Hãng bán
ESP 32	1	Espressif Systems
MAX30102	1	DigiKey
Oled 0.96 inch	1	Vishay
Mạch sạc TP4056	1	Billiton Ltd.
Pin16850	1	Panasonic

2.2 Thiết kế chi tiết

Dựa vào phần 2.1 đã làm rõ cho mọi người xem về sơ đồ khối, sơ đồ thuật toán của đồ án chúng ta hãy cùng làm rõ chi tiết từng khối và từng linh kiện :

+ **Khối vi xử lý:** Sử dụng vi điều khiển ESP32 DevKit V1



Hình 2.3 Sơ đồ chân của vi điều khiển Esp32

Dưới đây là các thông số cơ bản của ESP32 DevKit V1:

Microcontroller: ESP32-D0WDQ6

WiFi: Chuẩn IEEE 802.11 b/g/n 2.4 GHz (tương thích với WEP, WPA/WPA2)

Bluetooth: Bluetooth v4.2 và BLE (Bluetooth Low Energy)

Giao diện kết nối:

UART, SPI, I2C, I2S, CAN, Ethernet, SDIO, PWM, GPIO, ADC (Analog to Digital Converter)

Điện áp hoạt động: 3.3V

Nguồn cung cấp: Có thể hoạt động từ USB hoặc pin ngoại vi

Kích thước board: Thường là 54mm x 27mm

GPIO Pin	Input	Output	Chức năng đặc biệt
0	pulled up	OK	outputs PWM signal at boot
1	TX pin	OK	debug output at boot
2	OK	OK	connected to on-board LED
3	OK	RX pin	HIGH at boot
4	OK	OK	
5	OK	OK	outputs PWM signal at boot
6	X	X	connected to the integrated SPI flash
7	X	X	connected to the integrated SPI flash
8	X	X	connected to the integrated SPI flash
9	X	X	connected to the integrated SPI flash
10	X	X	connected to the integrated SPI flash
11	X	X	connected to the integrated SPI flash
12	OK	OK	boot fail if pulled high
13	OK	OK	
14	OK	OK	outputs PWM signal at boot
15	OK	OK	outputs PWM signal at boot
16	OK	OK	
17	OK	OK	
18	OK	OK	
19	OK	OK	
21	OK	OK	
22	OK	OK	
23	OK	OK	
25	OK	OK	
26	OK	OK	
27	OK	OK	
32	OK	OK	
33	OK	OK	
34	OK		input only
35	OK		input only
36	OK		input only
39	OK		input only

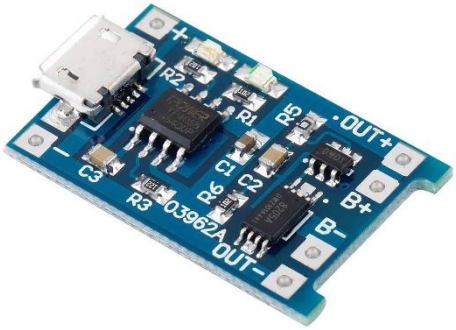
Bảng 2.2 Bảng tra cứu khả năng sử dụng của các chân

+ **Khối nguồn:** Sử dụng nguồn Pin 5V từ pin16850

Hình 2.4 Pin16850

♦ 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

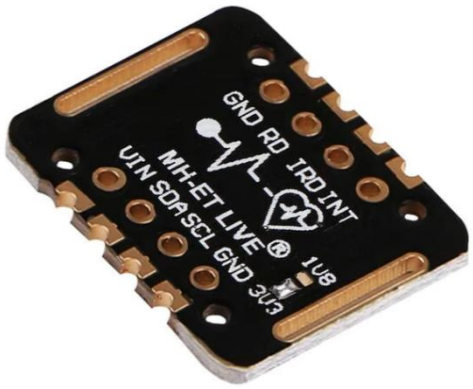
Kết hợp với mạch sạc pin TP4056 có IC bảo vệ



Hình 2.5Mạch sạc TP4056

- Điện áp vào: 5VDC
- Cổng Sạc Micro Type C
- Điện Áp Sạc: 4.2V
- Dòng sạc tối đa: 1A
- Bảo vệ xả đến điện áp: 2.5V
- Bảo Vệ Xả Quá Dòng: 3A
- Kích thước: 26x17MM

+ **Khối cảm biến:** Sử dụng cảm biến nhịp tim Max30102



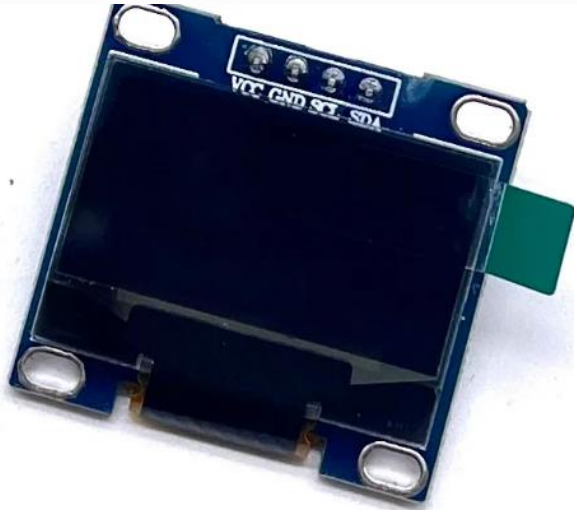
Hình 2.6 Cảm biến nhịp tim và nồng độ Oxy trong máu Max30102

Cảm biến sử dụng phương pháp đo quang phổ biến hiện nay với thiết kế và chất liệu mắt đo chuyên biệt từ chính hãng Maxim, cho độ chính xác và độ bền cao. Cảm biến sử dụng giao tiếp I2C với bộ thư viện có sẵn, rất dễ sử dụng.

Thông số kỹ thuật:

- IC chính: MAX30102.
- Đo được nhịp tim và nồng độ Oxy trong máu.
- Điện áp sử dụng: 3.3~5VDC.
- Tiêu thụ tối đa 6 mA
- Giao tiếp: I2C, mức tín hiệu TTL.
- Kích thước: 20.6 x 15.5mm

+ **Khối hiển thị:** Sử dụng màn hình Oled 0.96inch



Hình 2.7 Màn hình Oled 0.96inch

Màn hình Oled 0.96 inch giao tiếp I2C cho khả năng hiển thị đẹp, 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.

Thông tin kỹ thuật:

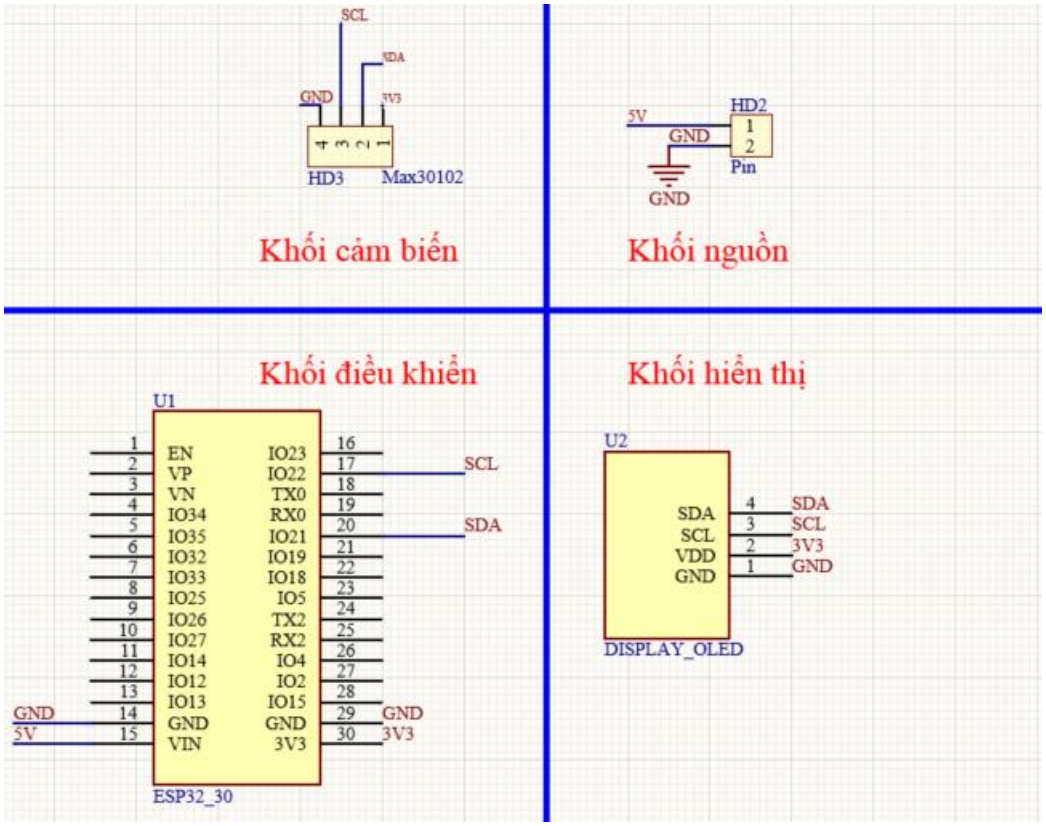
- Điện áp sử dụng: 2.2~5.5VDC.
- Công suất tiêu thụ: 0.04w
- Góc hiển thị: lớn hơn 160 độ
- Số điểm hiển thị: 128x64 điểm.
- Độ rộng màn hình: 0.96 inch
- Màu hiển thị: Trắng / Xanh Dương.
- Giao tiếp: I2C
- Driver: SSD1306

VCC	2.2~5.5VDC
GND	0VDC
SCL/SCK	xung Clock
SDA	dữ liệu vào Data in

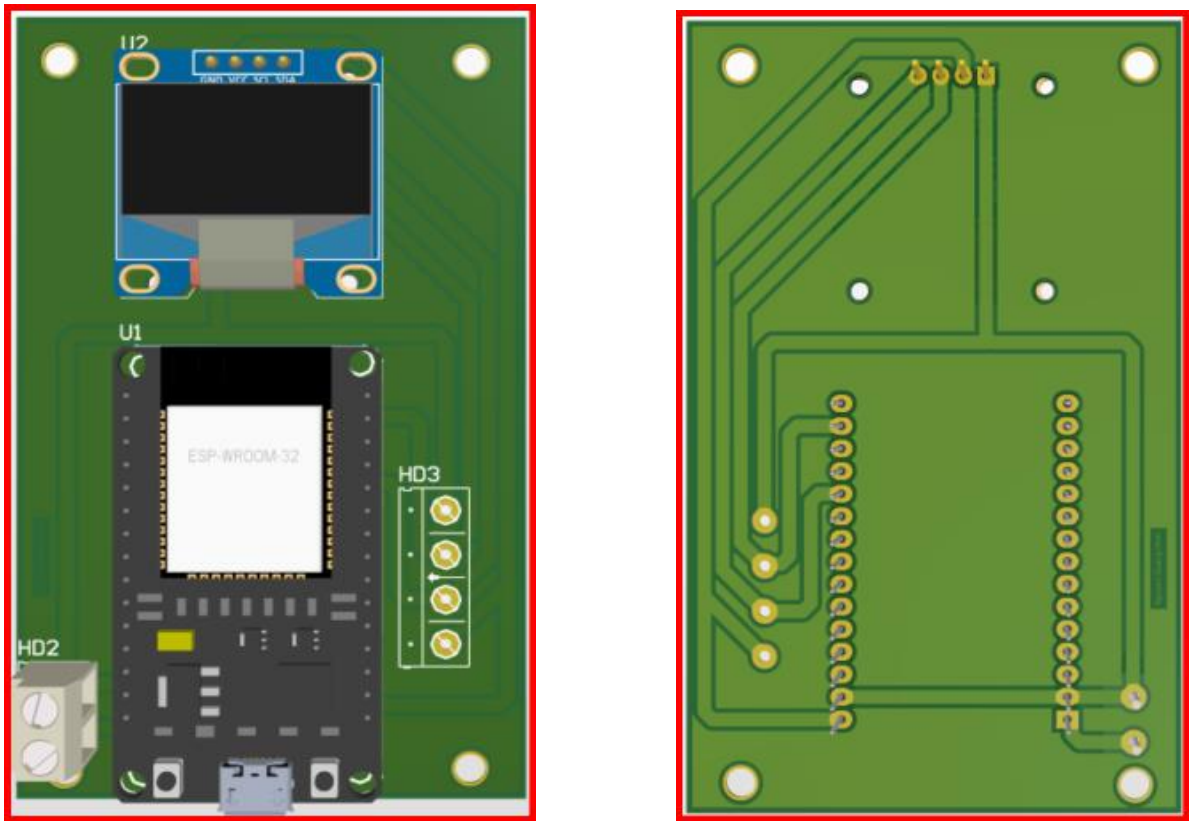
Bảng 2.3 Sơ đồ chân của màn hình Oled 0.96

2.3Thi công mạch/Thực hiện chương trình

+) Bước 1:Thiết kế mạch, đặt mạch in



Hình 2.8 Sơ đồ nguyên lý

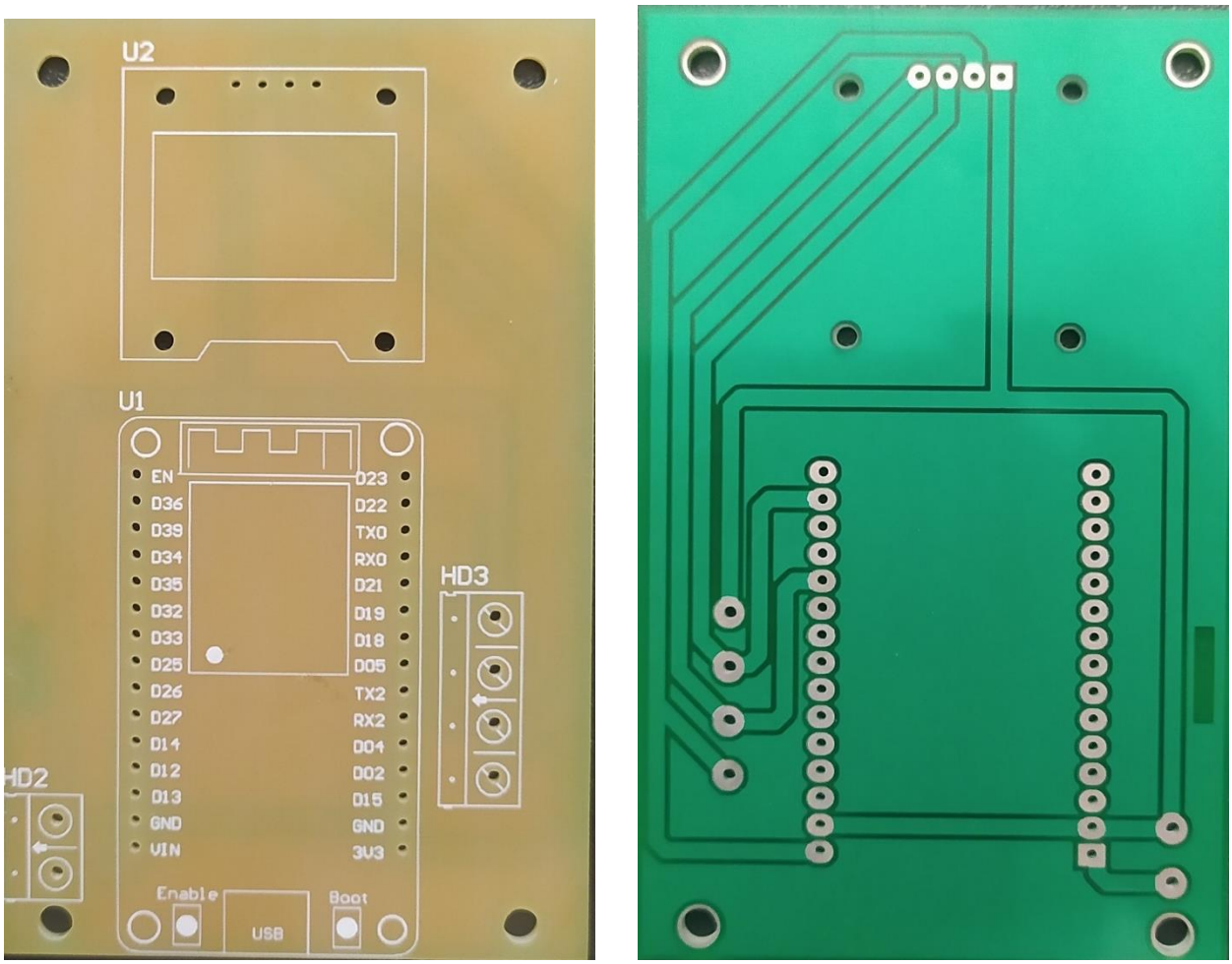


(a) Mặt trước (b) Mặt sau

Hình 2.9 Hình ảnh mô phỏng 3d mạch in

Sử dụng phần mềm Altium để vẽ mạch và sắp xếp các linh kiện hợp lí để khi đặt mạch in sẽ dễ dàng tìm được nơi nhận đặt mạch ‘

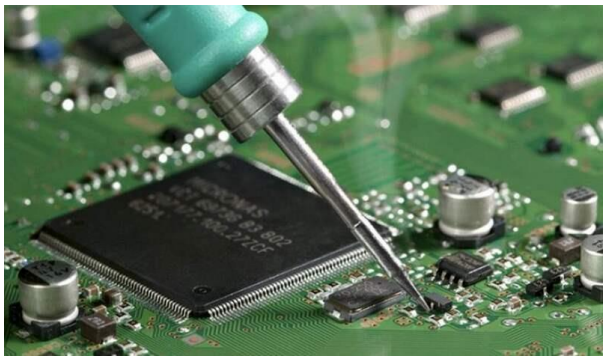
Liên hệ đặt mạch in tại: Điện tử Hatakey Việt Nam.Đây là nhà cung cấp mạch in độc quyền của **SUNBRIGHT PCB TECHNOLOGY CO.,LTD** tại Việt Nam. Các sản phẩm mạch in PCB cung cấp có xuất xứ rõ ràng, chất lượng tốt đạt tiêu chuẩn quốc tế với giá cả cạnh tranh so với các sản phẩm cùng loại trên thị trường. Đặc biệt, có thể đáp ứng được tất cả các đơn hàng từ làm mạch in mẫu đến các đơn hàng sản xuất hàng loạt với số lượng lớn một cách nhanh chóng, chuyên nghiệp.



(a) Mặt trước (b) Mặt sau

Hình 2.10 Hình ảnh mạch in sau khi đặt gia công

+) Bước 2: Hàn



Hình 2.11 Hàn mạch

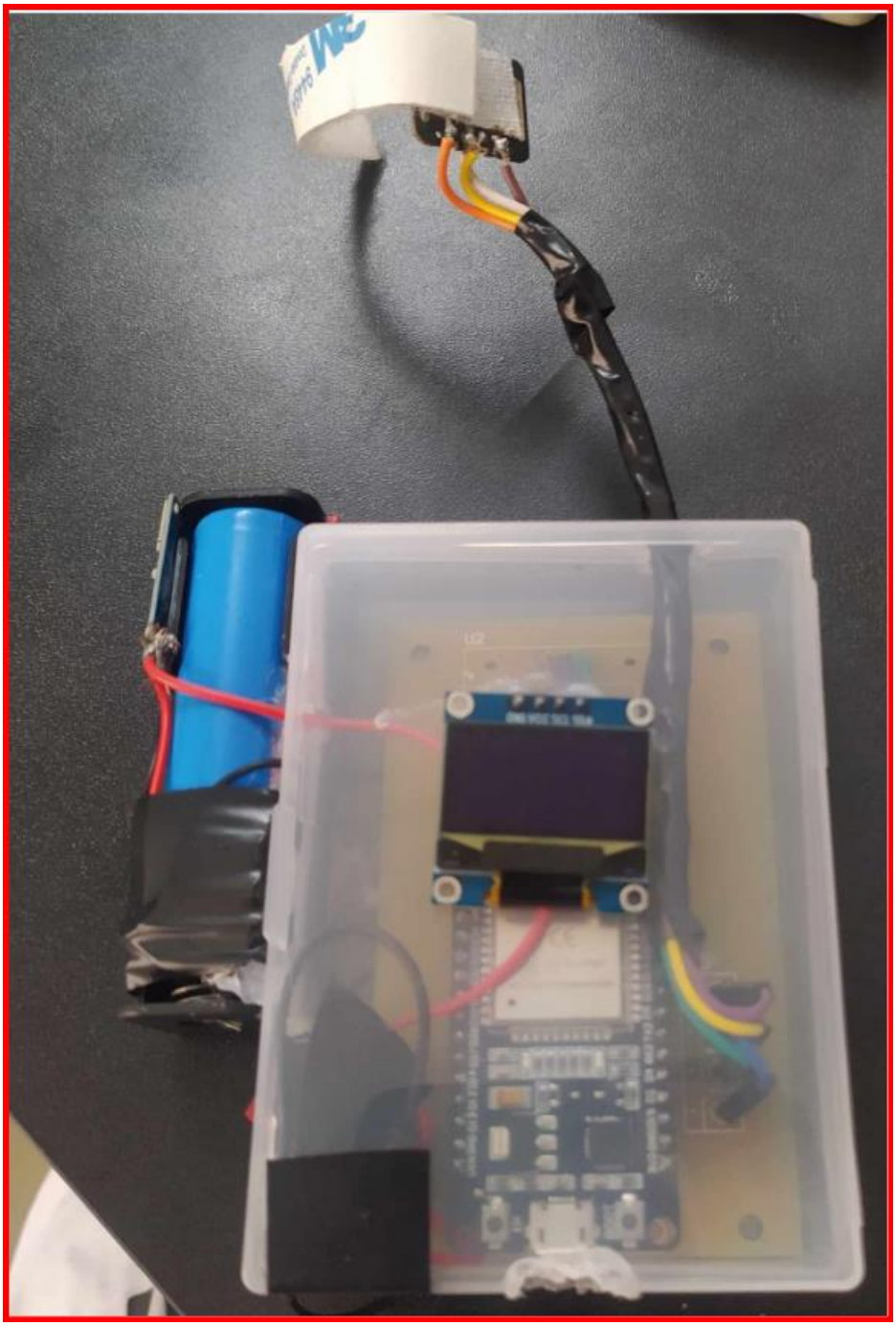
Khi Hàn chúng ta cần chú ý :
- Hàn nơi thoáng khí, cần có 1 quạt hút hơi - khói hàn ra ngoài, tránh để người hàn hít - ngửi trực tiếp với khói hàn.
- Khi hàn nên đeo kính, đi găng tay để tránh tiếp xúc trực tiếp với linh kiện, hóa chất.
- Chú ý để tránh tiếp xúc với mũi hàn, đầu mỏ hàn gây bỏng.
- Cần sử dụng kính lúp, kính phóng đại khi hàn, làm việc với các loại board mạch cỡ nhỏ, linh kiện nhỏ và phải đầy đủ ánh sáng tránh bị tật về mắt.

+) Bước 3: Thiết kế vỏ hộp và đặt mạch vào hộp

Hình 2.12 Vỏ hộp

Hộp đựng mạch làm từ nhựa, phù hợp với kích thước mạch điện tử nhỏ tiêu chuẩn size 88mm x 63mm. Sản phẩm giúp bảo vệ mạch điện tử một cách tốt nhất.

2.4Sản phẩm hoàn thiện

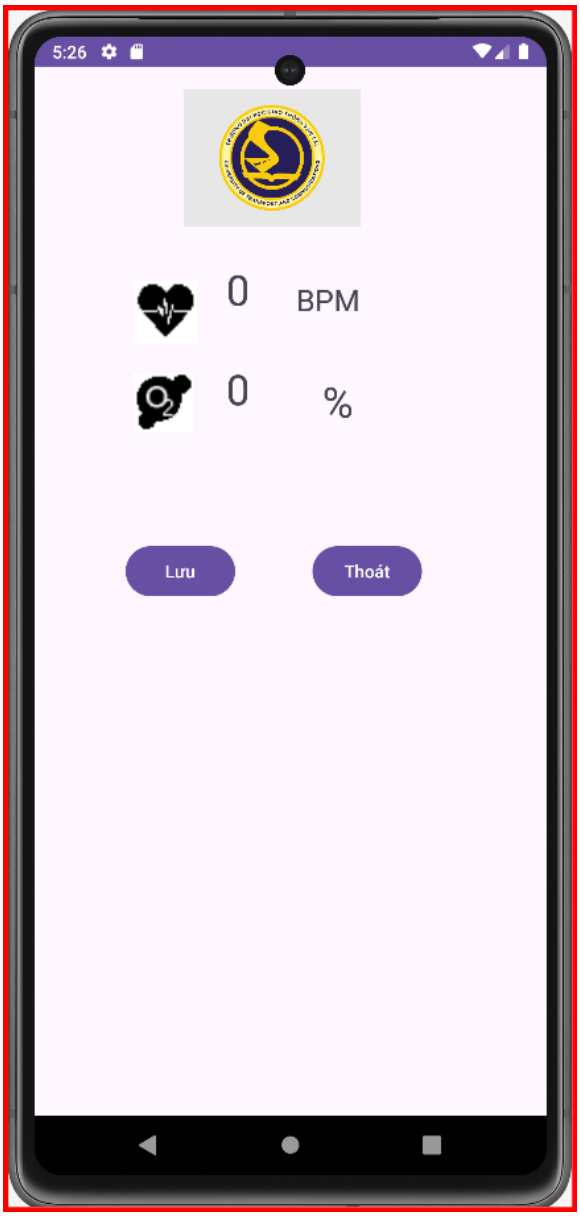


Hình 2.13 Hình ảnh sản phẩm hoàn thiện

Thông tin kỹ thuật:

- Điện áp sử dụng: 3.5~5.5VDC.
- Công suất tiêu thụ: 0.36w
- Kích thước: 100mm x 70mm
- Phạm vi nhiệt độ: 0 °C đến + 50 °C

Kết hợp với phần mềm tự thiết kế bằng Android Studio



Hình 2.14 Hình ảnh App hiển thị dữ liệu đo

Ứng dụng chạy trên Hệ điều hành Android, có các tính năng: hiển thị dữ liệu đo, lưu lại lịch sử đo tiện cho người dung theo dõi sức khỏe theo từng thời điểm. Thiết kế đơn giản dễ sử dụng phù hợp với mọi lứa tuổi.

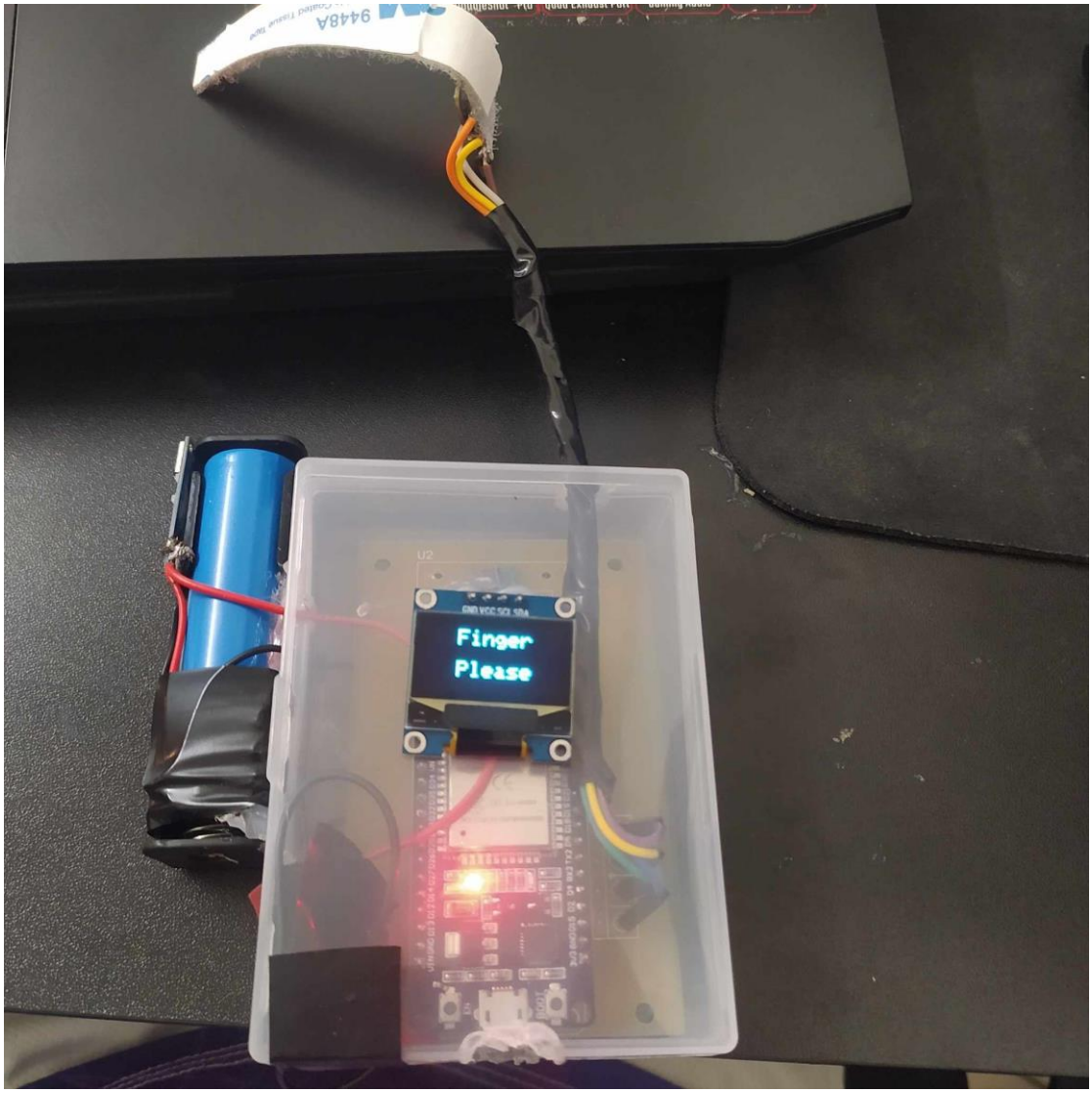
2.5 Kết luận chương 2

Qua chương 2, em đã biết cách tạo 1 sản phẩm IoT đơn giản ở mức đồ án môn học, tự tìm hiểu và áp dụng các môn đã học vào trong một thiết kế hoàn chỉnh.Biết cách Thiết kế và lập trình các mạch điện tử hoặc bo mạch in để thu thập dữ liệu từ cảm biến và điều khiển thiết bị.Kiểm tra và điều chỉnh phần cứng để đảm bảo hoạt động ổn định và hiệu quả. Lập trình ứng dụng hoặc firmware để thu thập, xử lý, và truyền dữ liệu từ thiết bị.Phát triển giao diện người dùng. Kết nối phần cứng và phần mềm lại với nhau và kiểm tra tính tương thích. Quá trình này có thể phức tạp và đòi hỏi sự cẩn thận và kiên nhẫn.

Chương III. THỬ NGHIỆM THIẾT BỊ THEO DÕI NHỊP TIM, NỒNG ĐỘ OXY TRONG MÁU VÀ KẾT QUẢ

3.1 Kịch bản thử nghiệm

Nội dung, tình huống thử nghiệm



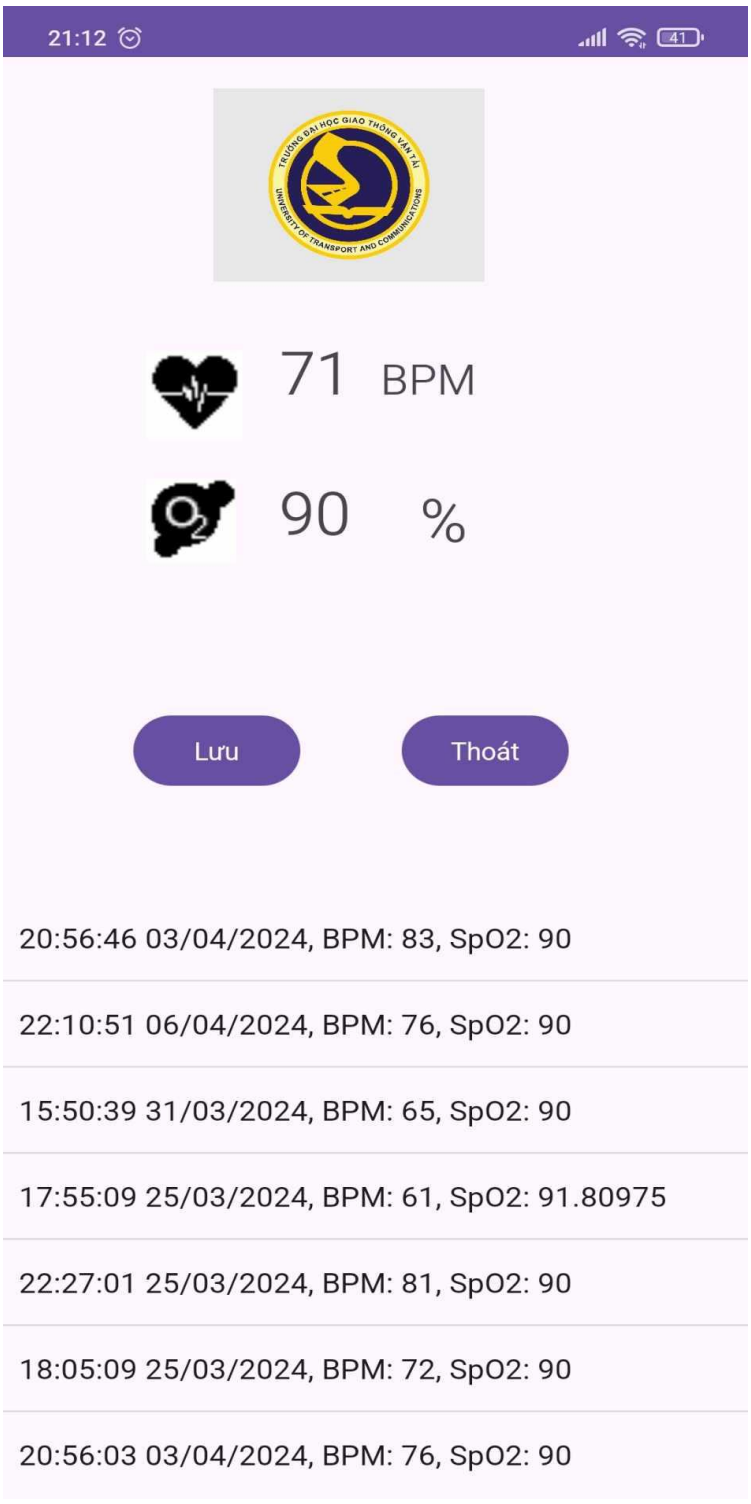
Hình 3.1 Khởi động hệ thống

Khi cần đo nhịp tim và nồng độ Oxy trong máu người sử dụng bật công tắc phía bên trái để khởi động hệ thống. Đợi đến khi màn hình hiển thị “Finger please” là hệ thống đã khởi động hoàn tất.

Lúc này, người sử dụng chỉ cần cho đặt thiết bị vào bàn tay trái và đặt cảm biến vào ngón tay trỏ để thiết bị thực hiện việc lấy mẫu và tính toán hiển thị giá trị lên màn hình Oled và App mobile tự thiết kế.

Hình 3.2 Hiển thị giá trị đo nhịp tim lên màn hình

Người dùng muốn lưu lại giá trị đo vào thời điểm nào đó chỉ cần ấn nút lưu là sẽ lưu lại giá trị đo tại thời điểm đó lên App mobile



Hình 3.3 Hiển thị App mobile và lịch sử đo

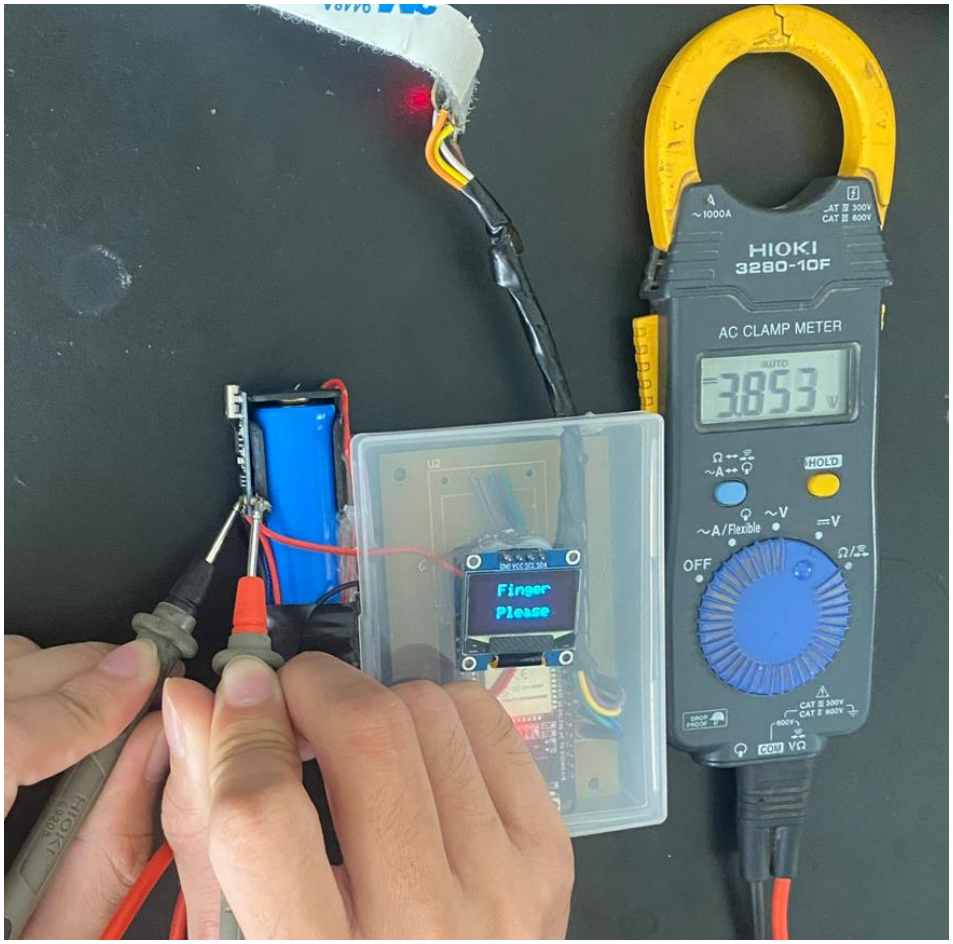
Một số lỗi người dùng có thể gặp phải như:

Thiết bị bị treo không vào chế độ đo, kiểm tra lại vi điều khiển ESP32, cảm biến, màn hình Oled xem đã hoạt động chưa, nếu như chưa hoạt động cần kiểm tra lại các kết nối hoặc gửi lại cho người sản xuất để kiểm tra lại các kết nối phần cứng.

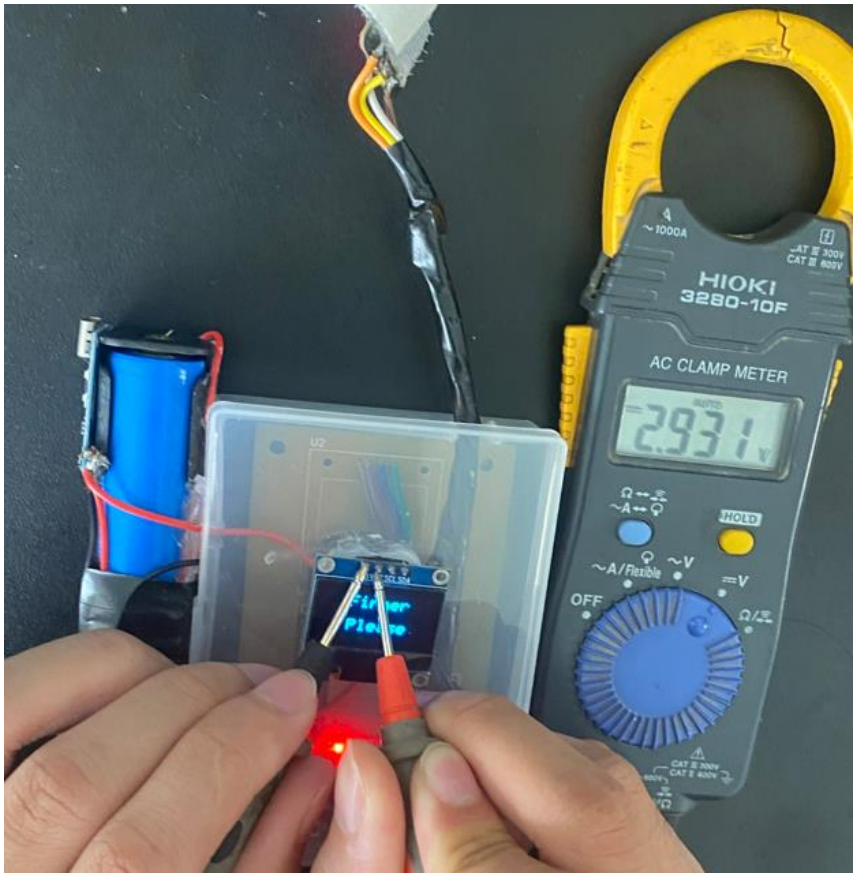
Khi vi điều khiển, cảm biến, màn hình Oled đã hoạt động thì mà thiết bị vẫn chưa vào chế độ đo thì người dùng cần kiểm tra lại kết nối WiFi đã ổn định chưa. Khi đã kết nối WiFi thành công mà thiết bị vẫn chưa vào chế độ đo người sử dụng nên gửi lại cho người sản xuất để kiểm tra lại thiết bị.

Thường thì người dùng sẽ gặp trường hợp 2 nhiều hơn do kết nối WiFi không ổn định sẽ dẫn đến lỗi này

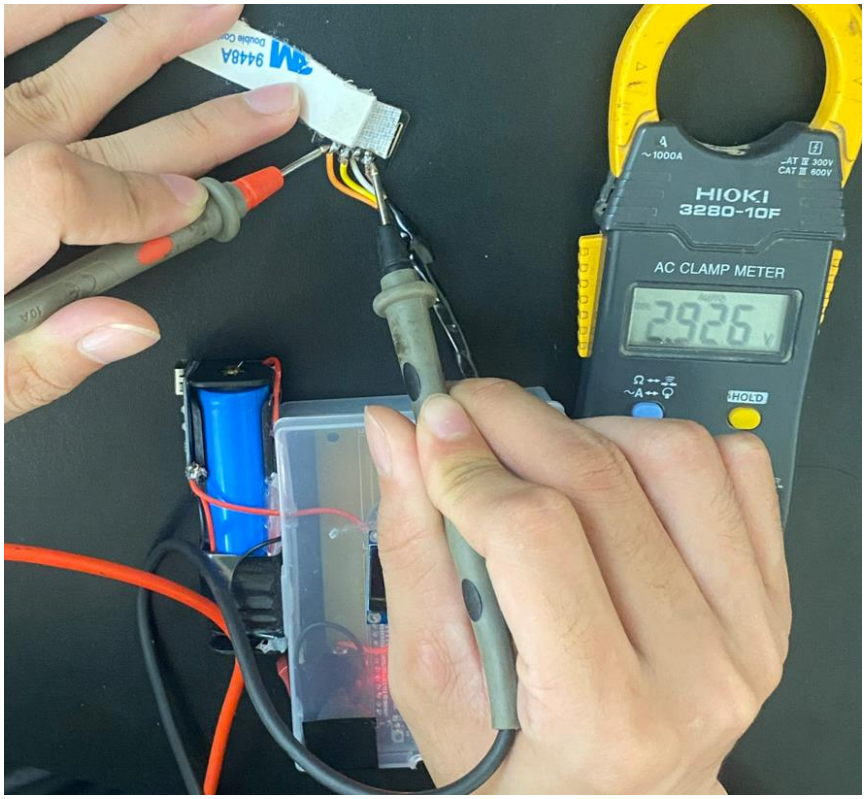
3.2 Kiểm tra độ ổn định của mạch



Hình 3.4 Điện áp nguồn pin cấp cho mạch



Hình 3.5 Điện áp sử dụng cho Oled



Hình 3.6 Điện áp vào cảm biến

3.3 Kết quả thử nghiệm

Thông qua thử nghiệm thực tế em thấy sản phẩm hoạt động đúng với yêu cầu đặt ra ở các mục 1.2, 1.3 .

Thời gian hoàn thành sản phẩm đạt yêu cầu mong muốn để thử nghiệm sản phẩm trong thời gian trước khi báo cáo để có cái nhìn, đánh giá tổng quan nhất về sản phẩm.

Lúc đầu,ý tưởng tạo ra sản phẩm nhỏ gọn tiện dụng dựa trên sự linh hoạt của vi điều khiển ESP32. Điều khó khăn trong việc làm dự án này là phải tìm được cách gửi dữ liệu lên Mobile App mình tự thiết kế.

Sau nhiều lần lựa chọn các phương án như sử dụng Web, Blynk app, app tự thiết kế sử dụng phần mềm Android Studio thì chúng em chọn phần cách tự viết App hiển thị sử dụng phần mềm Android Studio để thiết kế do có những ưu điểm như theo dõi từ xa mà không cần vào mạng nội bộ như cách sử dụng Web do em liên kết ứng dụng của mình với một cơ sở dữ liệu thời gian thực là Firebase – đây là một cơ sở dữ liệu có hệ thống cơ sở dữ liệu cực kì mạnh do Google phát hành và hoàn tiền miễn phí với đầy đủ các tính năng sử dụng cho việc theo dõi theo thời gian thực. Ưu điểm hơn Blynk app vì mình có thể tự viết thêm các chức năng mà không cần phụ thuộc vào bên thứ 3 thoải mái sáng tạo các chức năng và đặc biệt không cần mất phí để sử dụng những tính năng mà mình muốn.

Do vậy, mà chúng em chọn cách là sử dụng App mobile tự thiết kế để chủ động với các chức năng mà mình muốn và cũng là một cách tự học thêm cho mình một kiến thức cần thiết cho việc làm embedded sau này.

Các sản phẩm trên thị trường trên hiện nay chỉ có thể theo dõi trực tiếp trên thiết bị đo, không có chức năng theo dõi từ xa như thiết bị của chúng em và giá thành cũng cao hơn thiết bị của chúng em rất nhiều. Một số đồ án tương tự em tham khảo được họ cũng chỉ theo dõi bằng Web nhưng chỉ theo dõi được trong mạng nội bộ, một số khác thì sử dụng Blynk app thì rất khó tiếp cận được với người dùng vì sản phẩm hướng tới sự tiện dụng cho người sử dụng. Với sản phẩm của chúng em thì tiếp cận rất dễ với người sử dụng, hoạt động dễ dàng vì hoạt động trên nền tảng của hệ điều hành Android nếu như phát triển sản phẩm thì hoàn toàn có thể đẩy lên GOOGLE PLAY STORE với việc cung cấp cho mỗi người 1 tài khoản tương ứng với 1 id của thiết bị để theo dõi nhịp tim của bệnh nhân từ xa cho bác sĩ và người thân của bệnh nhân.

3.4 Đánh giá kết quả đạt được

Sau một thời gian thử nghiệm sản phẩm trong hơn 3 tuần, chúng em đã nhận thấy rằng thiết bị hoạt động ổn định và các giá trị đo khá chuẩn xác. Tuy nhiên, khi áp dụng sản phẩm trong điều kiện thực tế, chúng em nhận thấy rằng thiết kế của nó chưa đủ chắc chắn.

Vấn đề đầu tiên là khả năng chịu lực của sản phẩm không được đảm bảo đầy đủ. Trong một số trường hợp, sản phẩm có thể gặp phải áp lực vượt quá giới hạn, dẫn đến hỏng hóc hoặc mất tính hiệu.

Ngoài ra, chúng 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, ảnh hưởng đến hiệu suất và độ chính xác của nó.

Do đó, để cải thiện sản phẩm và đảm bảo hiệu suất tốt nhất khi sử dụng trong điều kiện thực tế, chúng tôi cần phải tối ưu hóa thiết kế để cải thiện khả năng chịu lực và chống nước, đồng thời tăng cường bảo vệ chống va đập. Điều này sẽ giúp sản phẩm trở nên đáng tin cậy và đáp ứng được các yêu cầu của người sử dụng.

3.5 Kết luận chương 3

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.

Việc sử dụng vi điều khiển ESP32, cảm biến Max30102 đã mang lại nhiều lợi ích trong việc nhanh chóng và chính xác hóa quá trình đo, nhưng cũng tiết lộ một số thách thức cần được giải quyết. Để đảm bảo tính ổn định và an toàn, việc tiếp tục nghiên cứu và cải tiến là cần thiết.

Qua đây, có thể thấy rằng làm ra một sản phẩm hoàn chỉnh có thể áp dụng thực tiễn không phải là điều dễ dàng gì.

KẾT LUẬT CHUNG VÀ HƯỚNG PHÁT TRIỂN

KẾT LUẬT CHUNG

• Ưu điểm:

- Hoạt động ổn định: Hệ thống hoạt động ổn định, đảm bảo tính chính xác và đáng tin cậy trong việc ghi lại nhịp tim và nồng độ Oxy trong máu.

- Thuận tiện: Sự thuận tiện của hệ thống là một trong những ưu điểm lớn nhất. Người dùng chỉ cần đưa tay vào gần vùng cảm biến mà không cần phải dùng bất kỳ thiết bị nào khác.

- Dễ sử dụng: Giao diện người dùng được thiết kế một cách đơn giản và trực quan. Người điểm dùng đều có thể sử dụng hệ thống một cách dễ dàng ngay từ lần đầu tiên

• Nhược điểm:

- Độ tin cậy của thiết bị:Thiết bị chưa được kiểm chứng của bất kì cơ quan tổ chức nào. Điều này đặt ra thách thức đối với tính tin cậy của thiết bị.

- Phụ thuộc vào công nghệ: Hệ thống phụ thuộc nhiều vào công nghệ gửi dữ liệu qua Clound của bên thứ 3 là GOOGLE, 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: Cần đảm bảo rằng hệ thống có các biện pháp bảo mật đủ mạnh để ngăn chặn việc lạm dụng hoặc giả mạo vào hệ thống, đặc biệt là trong các môi trường yêu cầu mức độ bảo mật cao.

ĐỊNH HƯỚNG/HƯỚNG PHÁT TRIỂN

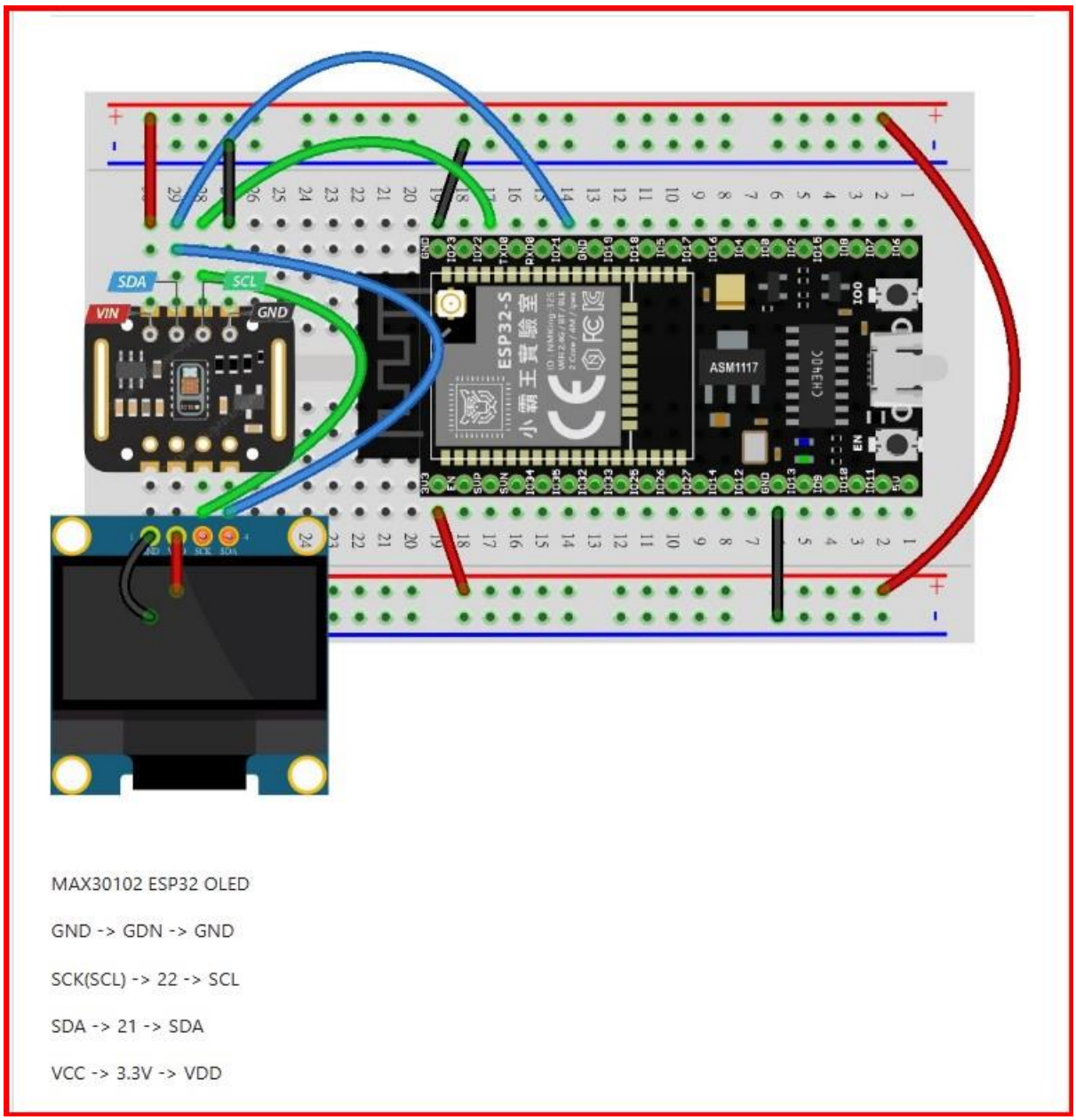
- Phát triển thêm các tính năng như lúc người dùng không sử dụng thì màn hình sẽ hiển thị thêm ngày giờ.
- Liên kết với các ứng dụng khác để có thể tận dụng tối đa tính năng
- Kết hợp thêm các cảm biến: Nhiệt độ, độ ẩm nhằm tận dụng các chân còn lại của vi điều khiển đồng thời tăng khả năng giám sát 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.
- Nâng cấp mẫu mã sản phẩm để phù hợp với thi yếu người dùng không chỉ tốt về mặt chức năng mà còn đẹp ở vẻ bề ngoài.

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://mecsu.vn/ho-tro-ky-thuat/gioi-thieu-ve-bo-mach-phat-trien-esp32.KyJ, truy cập ngày 24/2/2024, Giới thiệu về bo mạch phát triển ESP32.
3. https://linhkien888.vn/huong-dan-lap-trinh-esp32-voi-arduino-ide, truy cập ngày 24/2/2024, Hướng dẫn lập trình ESP32
4. https://khuenguyencreator.com/tong-quan-ve-so-do-chan-esp32-va-ngoai-vi/, truy cập ngày 24/2/2024, Tổng quan về sơ đồ chân ESP32 và ngoại vi.
5. https://randomnerdtutorials.com/esp32-firebase-realtime-database/, truy cập ngày 5/3/2024**,** ESP32: Getting Started with Firebase (Realtime Database).
6. https://randomnerdtutorials.com/esp32-client-server-wi-fi/, truy cập ngày 5/3/2024**,** hướng dẫn kết nối WiFi cho ESP32.
7. https://www.datasheethub.com/ssd1306-128x64-mono-0-96-inch-i2c-oled-display/, truy cập ngày 5/3/2024, datasheet Oled 0.96 inch.
8. https://randomnerdtutorials.com/esp32-ssd1306-oled-display-arduino-ide/, truy cập ngày 10/3/2024**,** hướng dẫn kết nối màn hình Oled với ESP32.
9. https://developer.android.com/studio/write/firebase?hl=vi, truy cập ngày 10/3/2024**,** kết nối FireBase với Android Studio.
10. https://microcontrollerslab.com/esp32-heart-rate-pulse-oximeter-max30102/, truy cập ngày 15/3/2024**,** MAX30102 Pulse Oximeter and Heart Rate Sensor with ESP32.
11. https://github.com/Probots-Techno-Solutions/Wifi-Oximiter-using-MAX30102-and-ESP32/blob/main/README.md, truy cập ngày 22/3/2024, Wifi Oximiter using MAX30102 and ESP32.10.
12. Datasheet ESP32 series, ESPRESSIF SYSTEMS, 2016.
13. Datasheet Max30102, maxim integrated products, 2021.
14. Datasheet Pin18650, Tenergy Corporation, 2009.
15. Datasheet TP4056, Shenzhen TPOWER Semiconductor, 2010.

Phụ lục

1. Phụ lục 1: sơ đồ mạch chi tiết



2. Phụ lục 2: Code ESP32

```
#include <Arduino.h>

#include <Adafruit_GFX.h>    //OLED libraries

#include <Adafruit_SSD1306.h> //OLED libraries
```

```
#include "MAX30105.h"           //MAX3010x library
#include "heartRate.h"          //Heart rate calculating algorithm
#include "ESP32Servo.h"

#if defined(ESP32)
  #include <WiFi.h>
#elif defined(ESP8266)
  #include <ESP8266WiFi.h>
#endif
#include <Firebase_ESP_Client.h>

//Provide the token generation process info.
#include "addons/TokenHelper.h"
//Provide the RTDB payload printing info and other helper functions.
#include "addons/RTDBHelper.h"

// Insert your network credentials
#define WIFI_SSID "P502"
#define WIFI_PASSWORD "20032002@"

// Insert Firebase project API Key
#define API_KEY "AIzaSyBME-Ez7_5SugKa1xcdo6VHPdKi_VNVrzc"
```

```
// Insert RTDB URLefine the RTDB URL */

#define DATABASE_URL "https://esp32-firebase-demo-8be1d-default-
rtdb.asia-southeast1.firebasedatabase.app/"


//Define Firebase Data object

FirebaseData fbdo;


FirebaseAuth auth;

FirebaseConfig config;


unsigned long sendDataPrevMillis = 0;

int count = 0;

bool signupOK = false;


MAX30105 particleSensor;


const byte RATE_SIZE = 4;

byte rates[RATE_SIZE];

byte rateSpot = 0;

long lastBeat = 0;

float beatsPerMinute;

int beatAvg;


// Timer variables (send new readings every three minutes)
```

```
unsigned long timerDelay = 180000;


double avered = 0;

double aveir = 0;

double sumirrms = 0;

double sumredrms = 0;

double SpO2 = 0;

double ESpO2 = 90.0;

double FSpO2 = 0.7; //filter factor for estimated SpO2

double frate = 0.95; //low pass filter for IR/red LED value to eliminate AC component

int i = 0;

int Num = 30;

#define FINGER_ON 7000

#define MINIMUM_SPO2 90.0


//OLED

#define SCREEN_WIDTH 128

#define SCREEN_HEIGHT 64

#define OLED_RESET    -1

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET); //Declaring the display name (display)
```

```
static const unsigned char PROGMEM logo2_bmp[] =

{ 0x03, 0xC0, 0xF0, 0x06, 0x71, 0x8C, 0x0C, 0x1B, 0x06, 0x18, 0x0E, 0x02,
0x10, 0x0C, 0x03, 0x10,

  0x04, 0x01, 0x10, 0x04, 0x01, 0x10, 0x40, 0x01, 0x10, 0x40, 0x01, 0x10,
0xC0, 0x03, 0x08, 0x88,

  0x02, 0x08, 0xB8, 0x04, 0xFF, 0x37, 0x08, 0x01, 0x30, 0x18, 0x01, 0x90,
0x30, 0x00, 0xC0, 0x60,

  0x00, 0x60, 0xC0, 0x00, 0x31, 0x80, 0x00, 0x1B, 0x00, 0x00, 0x0E, 0x00,
0x00, 0x04, 0x00,

};


static const unsigned char PROGMEM logo3_bmp[] =

{ 0x01, 0xF0, 0x0F, 0x80, 0x06, 0x1C, 0x38, 0x60, 0x18, 0x06, 0x60, 0x18,
0x10, 0x01, 0x80, 0x08,

  0x20, 0x01, 0x80, 0x04, 0x40, 0x00, 0x00, 0x02, 0x40, 0x00, 0x00, 0x02,
0xC0, 0x00, 0x08, 0x03,

  0x80, 0x00, 0x08, 0x01, 0x80, 0x00, 0x18, 0x01, 0x80, 0x00, 0x1C, 0x01,
0x80, 0x00, 0x14, 0x00,

  0x80, 0x00, 0x14, 0x00, 0x80, 0x00, 0x14, 0x00, 0x40, 0x10, 0x12, 0x00,
0x40, 0x10, 0x12, 0x00,

  0x7E, 0x1F, 0x23, 0xFE, 0x03, 0x31, 0xA0, 0x04, 0x01, 0xA0, 0xA0, 0x0C,
0x00, 0xA0, 0xA0, 0x08,

  0x00, 0x60, 0xE0, 0x10, 0x00, 0x20, 0x60, 0x20, 0x06, 0x00, 0x40, 0x60,
0x03, 0x00, 0x40, 0xC0,

  0x01, 0x80, 0x01, 0x80, 0x00, 0xC0, 0x03, 0x00, 0x00, 0x60, 0x06, 0x00,
0x00, 0x30, 0x0C, 0x00,

  0x00, 0x08, 0x10, 0x00, 0x00, 0x06, 0x60, 0x00, 0x00, 0x03, 0xC0, 0x00,
0x00, 0x01, 0x80, 0x00

};
```

```
static const unsigned char PROGMEM O2_bmp[] = {

  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xf0, 0x00, 0x3f, 0xc3, 0xf8, 0x00,
0xff, 0xf3, 0xfc,

  0x03, 0xff, 0xff, 0xfe, 0x07, 0xff, 0xff, 0xfe, 0x0f, 0xff, 0xff, 0xfe, 0x0f, 0xff,
0xff, 0x7e,

  0x1f, 0x80, 0xff, 0xfc, 0x1f, 0x00, 0x7f, 0xb8, 0x3e, 0x3e, 0x3f, 0xb0, 0x3e,
0x3f, 0x3f, 0xc0,

  0x3e, 0x3f, 0x1f, 0xc0, 0x3e, 0x3f, 0x1f, 0xc0, 0x3e, 0x3f, 0x1f, 0xc0, 0x3e,
0x3e, 0x2f, 0xc0,

  0x3e, 0x3f, 0x0f, 0x80, 0x1f, 0x1c, 0x2f, 0x80, 0x1f, 0x80, 0xcf, 0x80, 0x1f,
0xe3, 0x9f, 0x00,

  0x0f, 0xff, 0x3f, 0x00, 0x07, 0xfe, 0xfe, 0x00, 0x0b, 0xfe, 0x0c, 0x00, 0x1d,
0xff, 0xf8, 0x00,

  0x1e, 0xff, 0xe0, 0x00, 0x1f, 0xff, 0x00, 0x00, 0x1f, 0xf0, 0x00, 0x00, 0x1f,
0xe0, 0x00, 0x00,

  0x0f, 0xe0, 0x00, 0x00, 0x07, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00

};


int Tonepin = 4;


// Initialize WiFi

void initWiFi() {

  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

  Serial.print("Connecting to WiFi... ");

  while (WiFi.status() != WL_CONNECTED) {
```

```
    Serial.print('.');
    delay(1000);
  }
  Serial.println(WiFi.localIP());
  Serial.println();
}

void sendFloat(String path, float value){
  if (Firebase.RTDB.setFloat(&fbdo, path.c_str(), value)){
    Serial.print("Writing value: ");
    Serial.print (value);
    Serial.print(" on the following path: ");
    Serial.println(path);
    Serial.println("PASSED");
    Serial.println("PATH: " + fbdo.dataPath());
    Serial.println("TYPE: " + fbdo.dataType());
  }
  else {
    Serial.println("FAILED");
    Serial.println("REASON: " + fbdo.errorReason());
  }
}
```

```
void setup() {
  Serial.begin(115200);
  Serial.println("System Start");
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C); //Start the OLED display
  display.display();
  delay(3000);

  if (!particleSensor.begin(Wire, I2C_SPEED_FAST)) //Use default I2C port, 400kHz speed
  {
    Serial.println("MAX30102");
    while (1);
  }

  byte ledBrightness = 0x7F;
  byte sampleAverage = 4; //Options: 1, 2, 4, 8, 16, 32
  byte ledMode = 2;
  int sampleRate = 800; //Options: 50, 100, 200, 400, 800, 1000, 1600, 3200
  int pulseWidth = 215; //Options: 69, 118, 215, 411
  int adcRange = 16384; //Options: 2048, 4096, 8192, 16384
  // Set up the wanted parameters
  particleSensor.setup(ledBrightness, sampleAverage, ledMode, sampleRate, pulseWidth, adcRange); //Configure sensor with these settings
  particleSensor.enableDIETEMPRDY();
```

```
  particleSensor.setPulseAmplitudeRed(0x0A); //Turn Red LED to low to
indicate sensor is running

  particleSensor.setPulseAmplitudeGreen(0); //Turn off Green LED


  initWiFi();


  /* Assign the api key (required) */

  config.api_key = API_KEY;


  /* Assign the RTDB URL (required) */

  config.database_url = DATABASE_URL;


  /* Sign up */

  if (Firebase.signUp(&config, &auth, "", "")){

    Serial.println("ok");

    signupOK = true;

  }

  else{

    Serial.printf("%s\n", config.signer.signupError.message.c_str());

  }


  /* Assign the callback function for the long running token generation task */
```

```
  config.token_status_callback = tokenStatusCallback; //see
addons/TokenHelper.h


  Firebase.begin(&config, &auth);

  Firebase.reconnectWiFi(true);


}


void loop() {

  long irValue = particleSensor.getIR();    //Reading the IR value it will permit
us to know if there's a finger on the sensor or not


  if (irValue > FINGER_ON ) {

     if (checkForBeat(irValue) == true) {

       display.clearDisplay();

       display.drawBitmap(0, 0, logo3_bmp, 32, 32, WHITE);

       display.setTextSize(2);

       display.setTextColor(WHITE);

       display.setCursor(42, 10);

       //display.print(beatAvg); display.println(" BPM");

       display.drawBitmap(0, 35, O2_bmp, 32, 32, WHITE);

       display.setCursor(42, 40);


       if (beatAvg > 30) display.print(String(ESpO2) + "%");
```

```
  else display.print("---- %" );
  display.display();
  tone(Tonepin, 1000);
  delay(10);
  noTone(Tonepin);
  //Serial.print("Bpm="); Serial.println(beatAvg);
  long delta = millis() - lastBeat;
  lastBeat = millis();
  beatsPerMinute = 60 / (delta / 1000.0);
  if (beatsPerMinute < 255 && beatsPerMinute > 20) {
    rates[rateSpot++] = (byte)beatsPerMinute;
    rateSpot %= RATE_SIZE;
    beatAvg = 0;
    for (byte x = 0 ; x < RATE_SIZE ; x++) beatAvg += rates[x];
    beatAvg /= RATE_SIZE;
  }
}

uint32_t ir, red ;
double fred, fir;
particleSensor.check(); //Check the sensor, read up to 3 samples
if (particleSensor.available()) {
  i++;
```

```
        ir = particleSensor.getFIFOIR();

        red = particleSensor.getFIFORed();

        //Serial.println("red=" + String(red) + ",IR=" + String(ir) + ",i=" +
String(i));

        fir = (double)ir;

        fred = (double)red;

        aveir = aveir * frate + (double)ir * (1.0 - frate); //average IR level by low
pass filter

        avered = avered * frate + (double)red * (1.0 - frate);//average red level by
low pass filter

        sumirrms += (fir - aveir) * (fir - aveir);//square sum of alternate component
of IR level

        sumredrms += (fred - avered) * (fred - avered); //square sum of alternate
component of red level

        if ((i % Num) == 0) {

          double R = (sqrt(sumirrms) / aveir) / (sqrt(sumredrms) / avered);

          SpO2 = -23.3 * (R - 0.4) + 100;

          ESpO2 = FSpO2 * ESpO2 + (1.0 - FSpO2) * SpO2;//low pass filter

          if (ESpO2 <= MINIMUM_SPO2) ESpO2 = MINIMUM_SPO2;
//indicator for finger detached

          if (ESpO2 > 100) ESpO2 = 99.9;

          //Serial.print(",SPO2="); Serial.println(ESpO2);

          sumredrms = 0.0; sumirrms = 0.0; SpO2 = 0;

          i = 0;

        }
```

```
      particleSensor.nextSample(); //We're finished with this sample so move to
next sample

    }


    Serial.print("Bpm:" + String(beatAvg));


    if (beatAvg > 30)  Serial.println(",SPO2:" + String(ESpO2));

    else Serial.println(",SPO2:" + String(ESpO2));


    display.clearDisplay();

    display.drawBitmap(5, 5, logo2_bmp, 24, 21, WHITE);

    display.setTextSize(2);

    display.setTextColor(WHITE);

    display.setCursor(42, 10);

    display.print(beatAvg); display.println(" BPM");

    display.drawBitmap(0, 35, O2_bmp, 32, 32, WHITE);

    display.setCursor(42, 40);


    if (beatAvg > 30) display.print(String(ESpO2) + "%");

    else display.print("90.00%" );

    display.display();

    if (Firebase.ready() && (millis() - sendDataPrevMillis > 7000 ||
sendDataPrevMillis == 0)){

      sendDataPrevMillis = millis();
```

```
      if (Firebase.RTDB.setFloat(&fbdo, "/BPM", beatAvg)) {
        Serial.println("Sent BPM data to Firebase");
      } else {
        Serial.println("Failed to send BPM data to Firebase");
      }

      // Gửi dữ liệu SpO2 lên Firebase
      if (Firebase.RTDB.setFloat(&fbdo, "/SpO2", ESpO2)) {
        Serial.println("Sent SpO2 data to Firebase");
      } else {
        Serial.println("Failed to send SpO2 data to Firebase");
      }
    }
  }
//"Finger Please"
else {
  for (byte rx = 0 ; rx < RATE_SIZE ; rx++) rates[rx] = 0;
  beatAvg = 0; rateSpot = 0; lastBeat = 0;
  avered = 0; aveir = 0; sumirrms = 0; sumredrms = 0;
  SpO2 = 0; ESpO2 = 0;
  //Finger Please
  display.clearDisplay();
  display.setTextSize(2);
```

```
    display.setTextColor(WHITE);
    display.setCursor(30, 5);
    display.println("Finger");
    display.setCursor(30, 35);
    display.println("Please");
    display.display();
    noTone(Tonepin);
    if (Firebase.ready() && (millis() - sendDataPrevMillis > 1000 ||
sendDataPrevMillis == 0)){
        sendDataPrevMillis = millis();
        if (Firebase.RTDB.setFloat(&fbdo, "/BPM", beatAvg)) {
          Serial.println("Sent BPM data to Firebase");
        } else {
          Serial.println("Failed to send BPM data to Firebase");
        }
        // Gửi dữ liệu SpO2 lên Firebase
        if (Firebase.RTDB.setFloat(&fbdo, "/SpO2", ESpO2)) {
          Serial.println("Sent SpO2 data to Firebase");
        } else {
          Serial.println("Failed to send SpO2 data to Firebase");
        }
      }
  }
}
```

3. Phụ lục 3: code App mobile sử dụng Android studio

```
package com.example.max30102firebase;

import android.content.DialogInterface;
import android.content.SharedPreferences;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import com.example.max30102firebase.R;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Collections;
import java.util.Comparator;
import java.util.HashSet;
import java.util.Locale;
import java.util.Set;

public class MainActivity extends AppCompatActivity {
    // Khai báo biến giao diện
    TextView txtBPM, txtSPO2;
    Button btnLuu, btnThoat;
    ListView lv;
    ArrayList<String> mylist;
    ArrayAdapter<String> myadapter; // Sửa từ ArrayList thành ArrayAdapter
    SQLiteDatabase mydatabase;
    FirebaseDatabase database;
    private DatabaseReference mdatabase;
```

```
    private DatabaseReference reff;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        txtBPM = findViewById(R.id.txtBPM);
        txtSPO2 = findViewById(R.id.txtSPO2);
        btnLuu = findViewById(R.id.btnLuu);
        btnThoat = findViewById(R.id.btnThoat);
        lv = findViewById(R.id.lv);
        // Trong phương thức onClick của nút "Lưu"
        String currentTime = getCurrentTime();
        // Phương thức để lấy thời gian hiện tại theo định dạng chuẩn
        mdatabase = FirebaseDatabase.getInstance().getReference();
        mylist = new ArrayList<>();
        myadapter = new ArrayAdapter<>(this, android.R.layout.simple_list_item_1,
mylist);
        lv.setAdapter(myadapter);
        mydatabase = openOrCreateDatabase("ql.db", MODE_PRIVATE, null);
        // tạo table để chứa dữ liệu
        try{
            String sql = "CREATE TABLE tbl(time Text primary key, bpm TEXT,
spo2 TEXT )";
            mydatabase.execSQL(sql);
        }
        catch (Exception e){
            Log.e("Error", "Table đã tồn tại");
        }

        // Thêm lắng nghe để lấy dữ liệu từ Firebase cho nút "BPM"
        DatabaseReference bpmRef =
FirebaseDatabase.getInstance().getReference().child("BPM");
        bpmRef.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                if (dataSnapshot.exists()) {
                    String bpmValue = dataSnapshot.getValue().toString();
                    txtBPM.setText(bpmValue);
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {
                Log.e("Firebase", "Lỗi: " + databaseError.getMessage());
            }
        });

        // Thêm lắng nghe để lấy dữ liệu từ Firebase cho nút "SPO2"
        DatabaseReference spo2Ref =
```

```
FirebaseDatabase.getInstance().getReference().child("SpO2");
        spo2Ref.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                if (dataSnapshot.exists()) {
                    String spo2Value = dataSnapshot.getValue().toString();
                    txtSPO2.setText(spo2Value);
                }
            }
            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {
                Log.e("Firebase", "Lỗi: " + databaseError.getMessage());
            }
        });
        //
        btnThoat.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Hiển thị hộp thoại hỏi người dùng
                AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
                builder.setTitle("Xác nhận thoát");
                builder.setMessage("Bạn có muốn thoát khỏi ứng dụng không?");

                // Thêm nút "Có"
                builder.setNegativeButton("Có", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        // Kết thúc (finish) hoạt động hiện tại (Activity) để thoát khỏi ứng dụng
                        finish();
                    }
                });

                // Thêm nút "Không"
                builder.setPositiveButton("Không", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        // Đóng hộp thoại
                        dialog.dismiss();
                    }
                });

                // Hiển thị hộp thoại
                builder.show();
            }
        });

        // lưu dữ liệu
        btnLuu.setOnClickListener(new View.OnClickListener() {
            @Override
```

```
        public void onClick(View v) {
            // Hiển thị hộp thoại hỏi người dùng có muốn lưu không
            AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
            builder.setTitle("Xác nhận lưu");
            builder.setMessage("Bạn có muốn lưu dữ liệu không?");

            // Thêm nút "Không"
            builder.setPositiveButton("Không", new
DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    // Đóng hộp thoại
                    dialog.dismiss();
                }
            });

            // Thêm nút "Có"
            builder.setNegativeButton("Có", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    saveData();
                }
            });
        // Hiển thị hộp thoại
            builder.show();

        }
    });
    lv.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener()
{
        @Override
        public boolean onItemLongClick(AdapterView<?> parent, View view, int
position, long id) {
            // Hiển thị hộp thoại hỏi người dùng
            AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
            builder.setTitle("Xác nhận xóa");
            builder.setMessage("Bạn có muốn xóa phần tử này không?");

            // Thêm nút "Có"
            builder.setNegativeButton("Có", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    // Xóa phần tử khỏi danh sách
                    String removedItem = mylist.remove(position);

                    // Cập nhật ListView
                    myadapter.notifyDataSetChanged();

                    // Thêm code xóa dữ liệu khỏi cơ sở dữ liệu SQLite nếu cần
                }
```

```
            });

            // Thêm nút "Không"
            builder.setPositiveButton("Không", new
DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    // Đóng hộp thoại
                    dialog.dismiss();
                }
            });

            // Hiển thị hộp thoại
            builder.show();
            return true;
        }

    });

    restoreListState();

  }
  // tạo biến thời gian
  private String getCurrentTime() {
    Calendar calendar = Calendar.getInstance();
    SimpleDateFormat dateFormat = new SimpleDateFormat("HH:mm:ss
dd/MM/yyyy", Locale.getDefault());
    return dateFormat.format(calendar.getTime());
  }
  // tạo dữ liệu lưu
  private void saveData() {
    String time = getCurrentTime();
    String bpmValue = txtBPM.getText().toString();
    String spo2Value = txtSPO2.getText().toString();
    // Thêm dữ liệu vào cơ sở dữ liệu SQLite
    String insertQuery = "INSERT INTO tbl (time, bpm, spo2) VALUES ('" +
time + "', '" + bpmValue + "', '" + spo2Value + "')";
    mydatabase.execSQL(insertQuery);

    // Thêm dữ liệu vào ArrayList để hiển thị trong ListView
    String newItem = "" + time + ", BPM: " + bpmValue + ", SpO2: " +
spo2Value;
    mylist.add(newItem);
    // Sắp xếp ArrayList theo thời gian (tăng dần)
    Collections.sort(mylist, new Comparator<String>() {
      DateFormat f = new SimpleDateFormat("HH:mm:ss dd/MM/yyyy");

      @Override
      public int compare(String o1, String o2) {
        try {
```

```
                // Đảo ngược kết quả so sánh để sắp xếp theo thời gian mới nhất lên
trước
                return f.parse(o2.substring(0, 19)).compareTo(f.parse(o1.substring(0,
19)));
            } catch (ParseException e) {
                throw new IllegalArgumentException(e);
            }
        }
    });
    // Cập nhật ArrayAdapter
    myadapter.notifyDataSetChanged();
}
// Phương thức lưu trạng thái danh sách vào SharedPreferences
private void saveListState() {
    SharedPreferences sharedPreferences = getSharedPreferences("MyPrefs",
MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPreferences.edit();
    Set<String> set = new HashSet<>(mylist);
    editor.putStringSet("list", set);
    editor.apply();
}
// Phương thức khôi phục dữ liệu từ SharedPreferences
private void restoreListState() {
    SharedPreferences sharedPreferences = getSharedPreferences("MyPrefs",
MODE_PRIVATE);
    Set<String> set = sharedPreferences.getStringSet("list", null);
    if (set != null) {
        mylist.clear();
        mylist.addAll(set);
        myadapter.notifyDataSetChanged();
    }
}
// Trước khi thoát khỏi ứng dụng
@Override
protected void onPause() {
    super.onPause();
    saveListState();
}
}
```