ĐẠI HỌC BÁCH KHOA HÀ NỘI

Trường Điện – Điện tử

BÁO CÁO ĐỒ ÁN THIẾT KẾ II

Đề tài: Thiết kế mạch đo nhịp tim và nồng độ oxy trong máu

GV hướng dẫn: ThS. Vũ Sinh Thượng

SV thực hiện: Nguyễn Quang Minh

MSSV: 20214010

Lớp: Điện tử 11 – K66

Học kì: 20233

Hà Nội, 08/2023

MỤC LỤC

DANH MỤC HÌNH ẢNH

A. LỜI NÓI ĐẦU

Trong bối cảnh y tế hiện đại, việc theo dõi các chỉ số sinh học như nhịp tim và nồng độ oxy trong máu ngày càng trở nên quan trọng. Các chỉ số này không chỉ giúp phát hiện sớm các bệnh lý về tim mạch và hô hấp mà còn hỗ trợ việc theo dõi sức khỏe tổng thể của người bệnh. Với sự tiến bộ của công nghệ, thiết bị đo lường y tế đã trở nên nhỏ gọn, chính xác và dễ tiếp cận hơn, tạo điều kiện thuận lợi cho việc theo dõi sức khỏe tại nhà.

Đề tài "Thiết kế mạch đo nhịp tim và nồng độ oxy trong máu" được chọn với mục tiêu phát triển một thiết bị đo lường hiệu quả, chi phí thấp và dễ sử dụng. Trong báo cáo này, em sẽ trình bày chi tiết quá trình nghiên cứu, thiết kế và phát triển sản phẩm. Nội dung bao gồm các giai đoạn sau:

1. **Nghiên cứu tổng quan:** Khảo sát và nghiên cứu các công nghệ hiện có trong việc đo nhịp tim và nồng độ oxy trong máu. Điều này giúp em hiểu rõ hơn về yêu cầu kỹ thuật và các giải pháp khả thi.
2. **Thiết kế hệ thống:** Dựa trên kết quả nghiên cứu, em đã thiết kế một mạch đo với các thành phần chủ yếu như cảm biến, vi điều khiển, và giao diện hiển thị. Quá trình thiết kế này cũng bao gồm việc lựa chọn các linh kiện phù hợp, tính toán mạch điện và thiết kế PCB.
3. **Phát triển và thử nghiệm:** Sau khi hoàn thành thiết kế, em tiến hành lắp ráp và thử nghiệm thiết bị. Quá trình thử nghiệm bao gồm kiểm tra tính chính xác, độ nhạy và độ ổn định của thiết bị trong các điều kiện khác nhau. Kết quả thử nghiệm được ghi nhận và phân tích để đưa ra các điều chỉnh cần thiết.
4. **Kết luận và hướng phát triển:** Tổng kết những kết quả đạt được và đề xuất hướng phát triển trong tương lai.

Báo cáo này sẽ trình bày chi tiết các giai đoạn trên, từ việc nghiên cứu, thiết kế đến phát triển và thử nghiệm sản phẩm. Em hy vọng rằng, với những nỗ lực và kiến thức đã tích lũy được, sản phẩm cuối cùng sẽ đáp ứng được các tiêu chí về hiệu quả, chi phí và tiện ích, góp phần nâng cao chất lượng chăm sóc sức khỏe.

Em xin chân thành cảm ơn sự hỗ trợ và góp ý từ thầy Vũ Sinh Thượng và các bạn đồng nghiệp trong suốt quá trình thực hiện đồ án này.

B. NỘI DUNG

1. NGHIÊN CỨU TỔNG QUAN

1.1. Module cảm biến MAX30102

Em lựa chọn module cảm biến MAX30102 cho đồ án này bởi trên thị trường hiện nay MAX30102 đáp ứng được rất nhiều tiêu chí để thực hiện sản phẩm có độ chính xác cao, giá thành rẻ. Ngoài ra module còn được thiết kế tích hợp sẵn LED, photodectector và mạch khuếch đại tín hiệu, giúp đơn giản hoá việc thiết kế mạch, hơn nữa module tiêu thụ năng lượng rất thấp và có thể giao tiếp dễ dàng với vi điều khiển qua giao thức I2C.

1.1.1.Tổng quan

Module này tích hợp một IC cao cấp MAX30102 (phiên bản nâng cấp của MAX30100) từ **Analog Devices**, được sử dụng để đo nồng độ oxy (SpO2) và nhịp tim (HR). Nó kết hợp hai đèn LED, bộ tách sóng quang, và quá trình xử lý tín hiệu Analog với độ nhiễu thấp để phát hiện các tín hiệu SpO2 và HR.

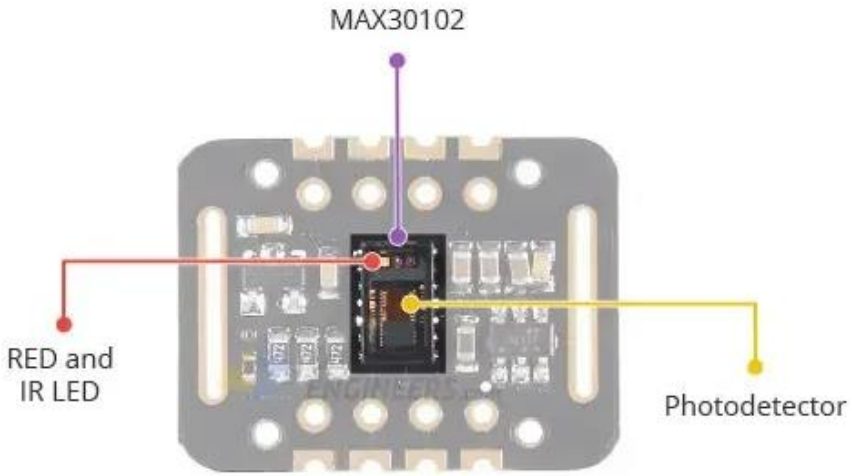

Figure 1: Module MAX30102

Mặt sau của module có hai đèn LED, bao gồm một đèn LED đỏ và một đèn LED hồng ngoại. Phía bên kia là bộ tách sóng quang cực kỳ nhạy. Cách hoạt động là cảm biến sẽ chiếu ánh sáng từ một đèn LED duy nhất vào cơ thể, sau đó ánh sáng phản xạ từ module và dựa trên thông tin thu được có thể đo nồng độ oxy trong máu và nhịp tim.

1.1.2.Giao tiếp I2C

Module sử dụng giao tiếp I2C, thông qua hai dây truyền dữ liệu SDA và SCL để giao tiếp với Vi xử lí ESP32. Địa chỉ I2C mặc định là: 0xAE $_{HEX}$ (cho thao tác ghi) và 0xAF $_{HEX}$ (cho thao tác đọc).

1.1.3.Thông số kỹ thuật

Nguồn cấp	3.3V đến 5V
Dòng điện	~600µA (trong lúc đo)
	~0.7µA (trong lúc ở chế độ chờ)
Bước sóng LED đỏ (RED LED)	660nm
Bước sóng LED hồng ngoại (IR LED)	880nm
Nhiệt độ hoạt động	-40˚C tới +85˚C
Sai số nhiệt độ	±1˚C

1.1.4.Nguyên lý hoạt động

Cảm biến nhịp tim và oxy trong máu MAX30102 hoạt động dựa trên nguyên lý phát hiện ánh sáng hấp thụ bởi máu trong các mô và mạch máu. MAX30102 sử dụng công nghệ phản xạ ánh sáng hồng ngoại (IR) và ánh sáng đỏ để đo lường nồng độ oxy (SpO2) và nhịp tim (HR) của người dùng.

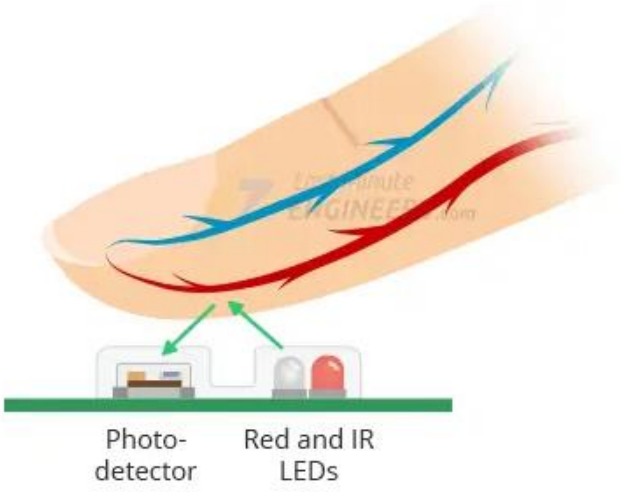

Figure 2: Nguyên lý hoạt động module MAX30102

Cảm biến được tích hợp các đèn LED hồng ngoại và đèn LED đỏ, cùng với một bộ lọc quang để tạo ra ánh sáng phù hợp để xuyên qua da và mô màu đỏ. Đèn LED hồng ngoại thẩm thấu sâu vào da, trong khi đèn LED đỏ thẩm thấu xa hơn. Khi ánh sáng được chiếu qua da, nó sẽ gặp phản xạ từ máu chảy trong các mạch máu dưới da.

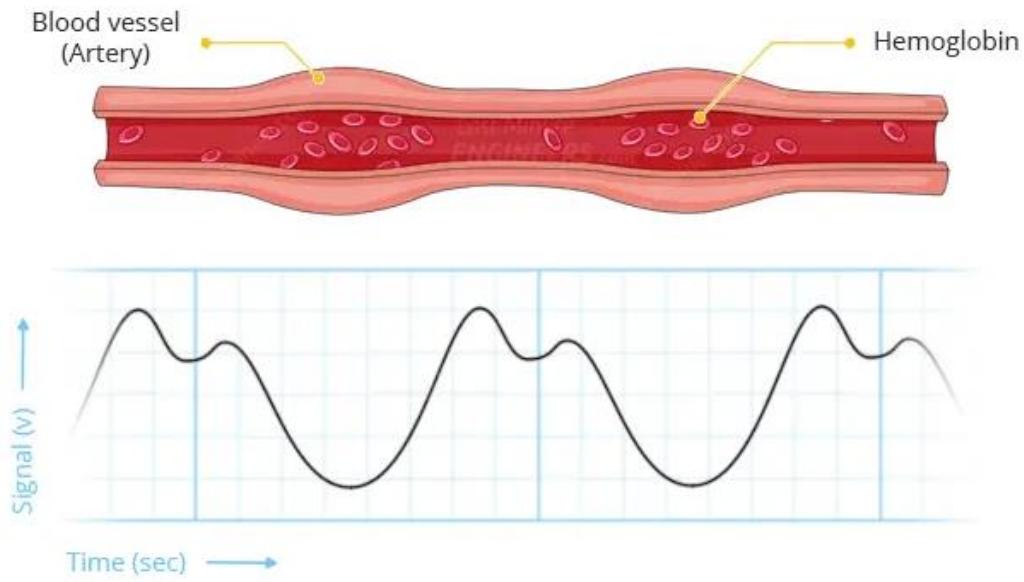

Figure 3: Tín hiệu module thu được khi đo

Cảm biến sử dụng photodiodes để đo lượng ánh sáng phản xạ từ máu. Các photodiodes nhận tín hiệu ánh sáng và chuyển đổi chúng thành dữ liệu Analog. Sau đó, dữ liệu này được khuếch đại và chuyển đổi thành dữ liệu kỹ thuật số bằng một bộ chuyển đổi analog-số (ADC).

1.1.5.Sơ đồ chân module MAX30102

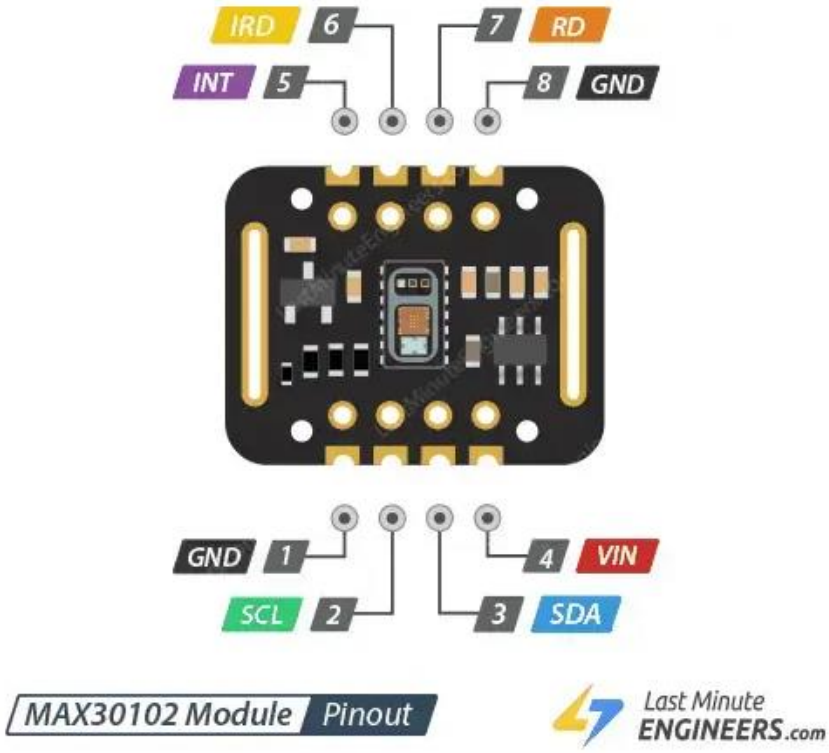

Figure 4: Sơ đồ chân module MAX30102

- **VIN:** Chân dương (3.3V hoặc 5V)

- **SCL:** Chân xung clock
- **SDA:** Chân dữ liệu
- **INT:** Chân ngắt (interrupt), được sử dụng để kích hoạt ngắt trong các sự kiện
- **IRD:** Chân dữ liệu hồng ngoại, đây là chân dữ liệu đầu ra hồng ngoại từ cảm biến
- **RD:** Chân dữ liệu LED đỏ, dùng để đo nồng độ oxy (SpO2) và nhịp tim (HR)
- **GND:** Chân nối đất

1.1.6.Ưu nhược điểm của module MAX30102

a. Ưu điểm:

- **Độ chính xác cao:** Cảm biến MAX30102 có khả năng đo lường chính xác nhịp tim và nồng độ oxy trong máu nhờ vào công nghệ quang học tiên tiến và các thuật toán xử lý tín hiệu.
- **Tính tích hợp cao:** Cảm biến này tích hợp đầy đủ các thành phần cần thiết, bao gồm LED, photodetector, và mạch khuếch đại tín hiệu, giúp giảm kích thước và đơn giản hóa thiết kế hệ thống.
- **Tiêu thụ năng lượng thấp:** MAX30102 được thiết kế để tiêu thụ năng lượng tối thiểu, phù hợp cho các ứng dụng di động và thiết bị đeo.
- **Giao tiếp I2C:** Cảm biến hỗ trợ giao thức I2C, giúp dễ dàng tích hợp với các vi điều khiển và hệ thống nhúng khác.
- **Khả năng loại bỏ nhiễu:** Với các bộ lọc số tích hợp, MAX30102 có khả năng loại bỏ nhiễu từ môi trường và các yếu tố ngoại vi, đảm bảo tín hiệu đo được ổn định và đáng tin cậy.

b. Nhược điểm:

- **Nhạy cảm với vị trí và áp lực:** Độ chính xác của cảm biến phụ thuộc nhiều vào vị trí đặt và áp lực của cảm biến lên da. Điều này có thể gây ra sai số nếu không được đặt đúng cách.
- **Giới hạn trong đo lường:** MAX30102 có thể gặp khó khăn khi đo lường trong các tình huống có ánh sáng môi trường mạnh hoặc khi người dùng có làn da dày hoặc sẫm màu, ảnh hưởng đến khả năng đọc tín hiệu.
- **Phụ thuộc vào thuật toán:** Độ chính xác của các phép đo cũng phụ thuộc nhiều vào các thuật toán xử lý tín hiệu và lọc, đòi hỏi sự tinh chỉnh và tối ưu hóa trong phần mềm.
- **Giá thành:** Mặc dù không phải là cảm biến đắt nhất, nhưng so với một số lựa chọn khác, MAX30102 có thể có giá thành cao hơn, đặc biệt khi cần tích hợp vào các sản phẩm giá rẻ.

1.2. LCD OLED 128x64 SSD1306

1.2.1.Tổng quan

Màn hình OLED 0.96 SSD1306 là một loại màn hình hiển thị sử dụng công nghệ Organic Light Emitting Diode (OLED). Kích thước màn hình thường là 0.96 inch, đây là kích thước phổ biến được sử dụng trong nhiều ứng dụng nhỏ gọn.

OLED 0.96 có độ phân giải là 128×64 pixel, tức là có 128 điểm ảnh trên chiều ngang và 64 điểm ảnh trên chiều dọc. Với độ phân giải này, màn hình có thể hiển thị các đồ họa, ký tự và biểu đồ đơn giản.

Figure 5: Màn hình OLED SSD1306

Một trong những đặc điểm nổi bật của màn hình OLED là khả năng tự phát sáng của các pixel. Mỗi pixel trên màn hình OLED có thể tự phát ra ánh sáng mà không cần đèn nền phụ trợ như các loại màn hình khác. Điều này cho phép màn hình OLED có độ tương phản cao, màu sắc tươi sáng và góc nhìn rộng.

Màn hình OLED 0.96 thường được kết nối với Arduino thông qua **giao tiếp I2C** hoặc **SPI**. Có sẵn các thư viện hỗ trợ để điều khiển màn hình OLED trên Arduino IDE.

1.2.2.Bản đồ bộ nhớ OLED (OLED Memory Map)

Để có thể điều khiển được màn hình OLED, việc hiểu bản đồ bộ nhớ là rất quan trọng.

Dù kích thước của màn hình OLED là bao nhiêu, bộ điều khiển SSD1306 bao gồm một RAM Dữ liệu Hiển thị Đồ họa (GDDRAM) có dung lượng 1KB để lưu trữ các mẫu bit sẽ được hiển thị trên màn hình. Vùng nhớ 1KB này được chia thành 8 trang (từ 0 đến 7). Mỗi trang có 128 cột/đoạn (khối 0 đến 127). Và, mỗi cột có thể lưu trữ 8 bit dữ liệu (từ 0 đến 7).

Tổng cộng:

8 trang x 128 cột x 8 bit = 8192 bit = 1024 byte = 1KB bộ nhớ

Toàn bộ bộ nhớ 1KB này, bao gồm các trang, cột và dữ liệu, được đánh dấu như sau:

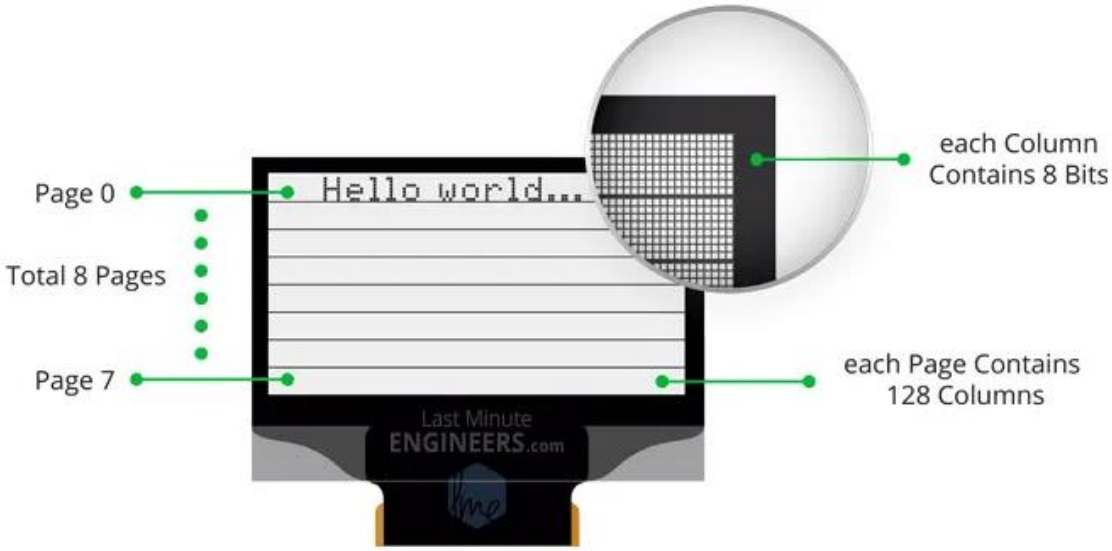

Figure 6: OLED Memory Map

Mỗi bit đại diện cho một pixel OLED duy nhất trên màn hình và có thể được chương trình BẬT hoặc TẮT.

1.2.3.Thông số kỹ thuật

Công nghệ màn hình	OLED (Organic LED)
Giao tiếp MCU	I2C / SPI
Kích thước	0.96 inch
Độ phân giải	128×64 pixels
Điện áp hoạt động	3.3V – 5V
Dòng điện hoạt động	20mA tối đa
Góc nhìn	160°
Số ký tự một hàng	21 ký tự
Số cột ký tự	7 cột

1.2.4.Sơ đồ chân màn hình OLED 0.96 SSD1306

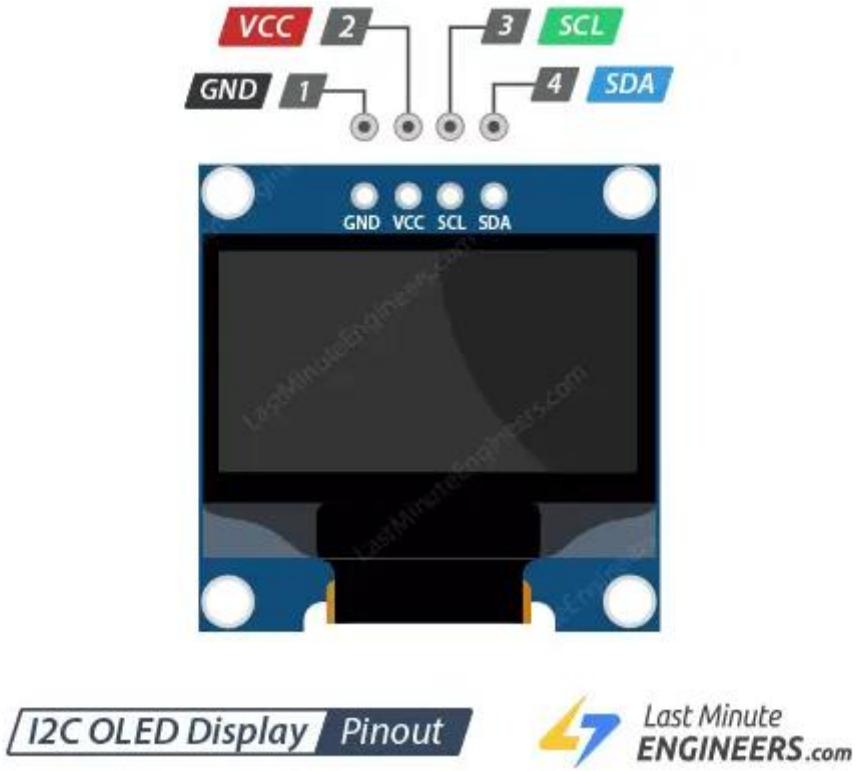

Figure 7: Sơ đồ chân màn hình OLED 0.96 I2C

- **VCC:** là nguồn cấp cho màn hình có thể là 3.3V hoặc 5V
- **GND:** chân nối đất
- **SCL:** chân xung clock
- **SDA:** chân dữ liệu nối tiếp

1.2.5.Ưu nhược điểm của màn hình OLED 0.96 SSD1306

a. <u>Ưu điểm:</u>

- **Tiêu thụ điện năng thấp**: OLED không cần đèn nền như các màn hình LCD truyền thống, vì mỗi điểm ảnh tự phát sáng. Điều này giúp tiết kiệm năng lượng, đặc biệt là khi hiển thị các hình ảnh có nhiều điểm đen (đã tắt điểm ảnh).
- **Độ tương phản cao**: OLED có khả năng hiển thị màu đen sâu hơn nhiều so với LCD, do các điểm ảnh có thể hoàn toàn tắt. Điều này mang lại độ tương phản cao và hình ảnh rõ nét.
- **Góc nhìn rộng**: OLED cung cấp góc nhìn rộng, giúp hình ảnh không bị thay đổi màu sắc hoặc chất lượng khi nhìn từ các góc khác nhau.
- **Kích thước nhỏ gọn và nhẹ**: Màn hình 0.96 inch rất nhỏ gọn, thích hợp cho các dự án cần tiết kiệm không gian hoặc thiết bị di động.

- **Độ phân giải đủ tốt**: Độ phân giải 128x64 pixel đủ để hiển thị văn bản đơn giản, hình ảnh đồ họa cơ bản và biểu đồ.
- **Giao tiếp đơn giản**: SSD1306 hỗ trợ giao tiếp I2C và SPI, dễ dàng kết nối với vi điều khiển và các hệ thống nhúng khác.

b. Nhược điểm

- **Kích thước màn hình nhỏ**: Với kích thước chỉ 0.96 inch, màn hình không phù hợp cho các ứng dụng cần hiển thị nhiều thông tin hoặc chi tiết.
- **Hạn chế màu sắc**: Hầu hết các màn hình OLED SSD1306 chỉ hỗ trợ hiển thị đơn sắc (thường là trắng trên nền đen), hạn chế khả năng hiển thị các hình ảnh màu hoặc giao diện người dùng phức tạp.
- **Tuổi thọ hạn chế**: OLED có thể suy giảm độ sáng và chất lượng theo thời gian do hiện tượng "burn-in" (bóng mờ trên màn hình) khi hiển thị các hình ảnh tĩnh trong thời gian dài.
- **Chi phí cao hơn**: So với các màn hình LCD hoặc TFT thông thường, OLED có thể đắt hơn một chút, đặc biệt là với các phiên bản có màu hoặc kích thước lớn hơn.

1.3. Vi điều khiển ESP32

Vi điều khiển ESP32 được em lựa chọn để làm đồ án này bởi nó có những ưu điểm vượt trội hơn rất nhiều so với vi điều khiển Arduino Nano, từ những kết nối không dây như Wifi và Bluetooth để phục vụ những nâng cấp sau này, hiệu năng và tài nguyên mạnh mẽ, hỗ trợ nhiều tính năng và tiêu thụ năng lượng hiệu quả là những tiêu chí hàng đầu để có thể phát triển một sản phẩm dễ tiếp cận và tiết kiệm chi phí hoạt động.

1.3.1.Tổng quan

ESP32 là một bộ vi điều khiển thuộc danh mục vi điều khiển trên chip công suất thấp và tiết kiệm chi phí. Hầu hết tất cả các biến thể ESP32 đều tích hợp Bluetooth và Wi-Fi chế độ kép, làm cho nó có tính linh hoạt cao, mạnh mẽ và đáng tin cậy cho nhiều ứng dụng.

Nó là sự kế thừa của vi điều khiển NodeMCU ESP8266 phổ biến và cung cấp hiệu suất và tính năng tốt hơn. Bộ vi điều khiển ESP32 được sản xuất bởi Espressif Systems và được sử dụng rộng rãi trong nhiều ứng dụng khác nhau như IoT, robot và tự động hóa.

ESP32 cũng được thiết kế để tiêu thụ điện năng thấp, lý tưởng cho các ứng dụng chạy bằng pin. Nó có hệ thống quản lý năng lượng cho phép nó hoạt động ở chế độ ngủ và chỉ thức dậy khi cần thiết, điều này có thể kéo dài tuổi thọ pin rất nhiều.

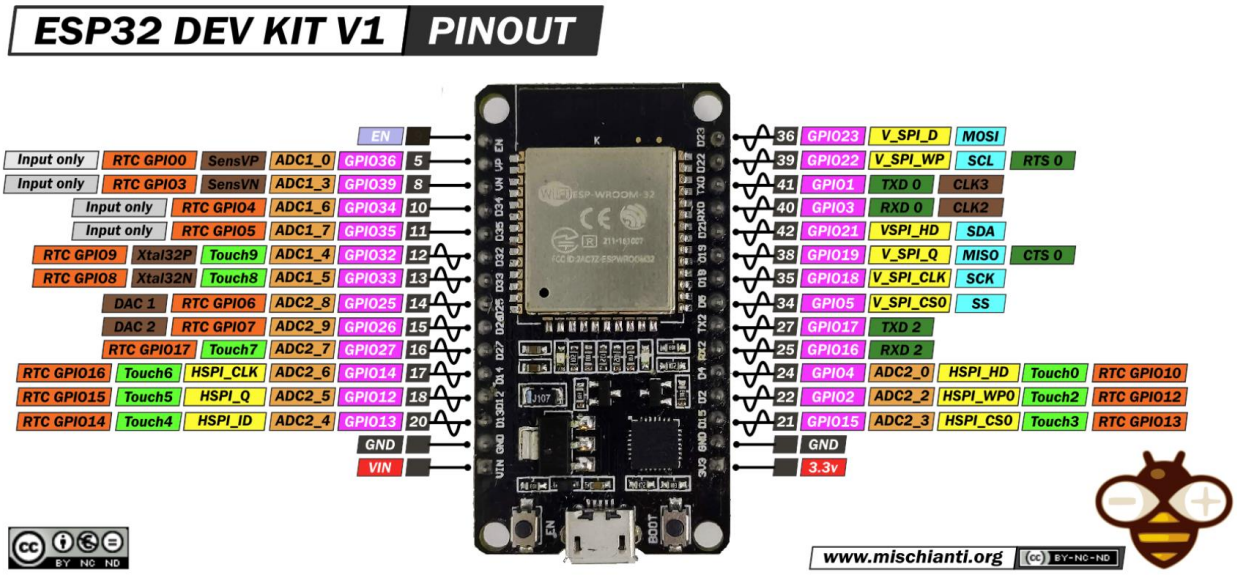

Figure 8: Vi điều khiển ESP32

1.3.2.Thông số kỹ thuật

- 30 chân GPIOs
- Vi xử lí Xtensa 32bit lõi kép
- 4MB bộ nhớ flash
- 50KB RAM

Giao thức không dây	Wireless 802.11 b / g / n standard
Phiên bản Bluetooth	BLE 4.0
Tần số hoạt động	2.4 GHz - 2.5 GHz
Điện áp hoạt động	2.7 – 3.6V
Nhiệt độ hoạt động	-40˚C tới +85˚C

1.3.3.Ưu nhược điểm của ESP32

a. Ưu điểm:

- **Kết nối không dây tích hợp**: ESP32 có cả Wi-Fi và Bluetooth (BLE và Classic) tích hợp, giúp dễ dàng kết nối và truyền dữ liệu qua các mạng không dây. Điều này làm cho ESP32 lý tưởng cho đồ án bởi có thể tích hợp thêm tính năng IoT sau này.
- **Hiệu năng cao**: Được trang bị bộ vi xử lý Xtensa lõi kép 32-bit với xung nhịp lên đến 240 MHz, ESP32 cung cấp hiệu năng mạnh mẽ để xử lý các tác

vụ phức tạp. Nó cũng có bộ nhớ RAM lớn hơn so với nhiều vi điều khiển khác trong cùng phân khúc.

- **Đa dạng giao diện ngoại vi**: ESP32 hỗ trợ nhiều giao thức giao tiếp như UART, SPI, I2C, I2S, ADC, DAC, PWM, và hơn thế nữa. Số lượng chân GPIO phong phú và đa chức năng giúp nó dễ dàng kết nối với nhiều loại thiết bị ngoại vi.
- **Tiêu thụ năng lượng linh hoạt**: ESP32 có các chế độ tiết kiệm năng lượng, bao gồm chế độ ngủ sâu (deep sleep) và chế độ ngủ (light sleep), giúp tối ưu hóa tiêu thụ điện năng cho các ứng dụng di động và các thiết bị chạy bằng pin.

b. Nhược điểm:

- **Độ phức tạp**: So với các vi điều khiển đơn giản như Arduino Uno hoặc Nano, ESP32 có kiến trúc phức tạp hơn, yêu cầu kiến thức về lập trình và phần cứng cao hơn để tận dụng tối đa các tính năng của nó.
- **Tiêu thụ năng lượng ở chế độ hoạt động cao**: Mặc dù có các chế độ tiết kiệm năng lượng, khi hoạt động ở chế độ hiệu suất cao, ESP32 tiêu thụ nhiều điện năng hơn so với các vi điều khiển đơn giản hơn.
- **Chi phí cao hơn**: ESP32 thường có giá thành cao hơn so với các vi điều khiển cơ bản khác, đặc biệt khi so sánh với các vi điều khiển không có kết nối không dây.

1.4. Giao thức kết nối I2C

Giao thức I2C được sử dụng trong hệ thống để kết nối vi xử lí ESP32 với cảm biến nhịp tim và nồng độ oxy trong máu MAX30102 và màn hình LCD OLED SSD1306 để chúng có thể truyền nhận dữ liệu trong suốt quá trình hoạt động.

1.4.1.Giới thiệu về chuẩn giao tiếp I2C

Giao thức I2C (Inter-Integrated Circuit) là một giao thức truyền thông nối tiếp được phát triển bởi Philips (hiện tại là NXP Semiconductors) vào những năm 1980. Nó cho phép truyền dữ liệu giữa các thiết bị điện tử như vi điều khiển, cảm biến,… thông qua hai dây SDA và SCL.

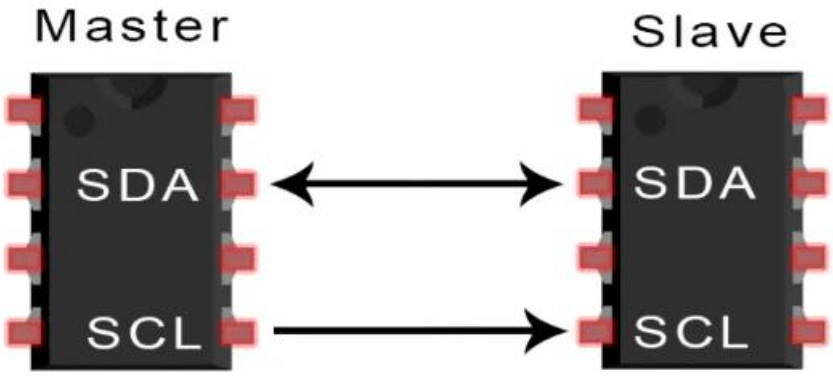

Figure 9: Mô hình giao tiếp I2C

Chuẩn giao tiếp I2C được xây dựng trên nguyên tắc Master-Slave, trong đó một Master điều khiển quá trình truyền và nhận dữ liệu từ các Slave. Một Master có thể kết nối với nhiều Slave, và mỗi Slave được xác định bằng một địa chỉ duy nhất.

SDA và SCL của **giao thức I2C** sử dụng cơ chế truyền thông đồng bộ, trong đó tín hiệu (SCL) được Master điều khiển. Dữ liệu (bit) được truyền từng bit một qua đường SDA.

1.4.2.Cách giao thức I2C hoạt động

Với I2C, dữ liệu được truyền dưới dạng tin nhắn và được chia thành các khung dữ liệu. Mỗi tin nhắn bao gồm một khung địa chỉ chứa địa chỉ nhị phân của thiết bị Slave và một hoặc nhiều khung dữ liệu chứa dữ liệu được truyền. Tin nhắn cũng bao gồm các điều kiện bắt đầu và dừng, các bit đọc/ghi và các bit ACK/NACK giữa các khung dữ liệu:

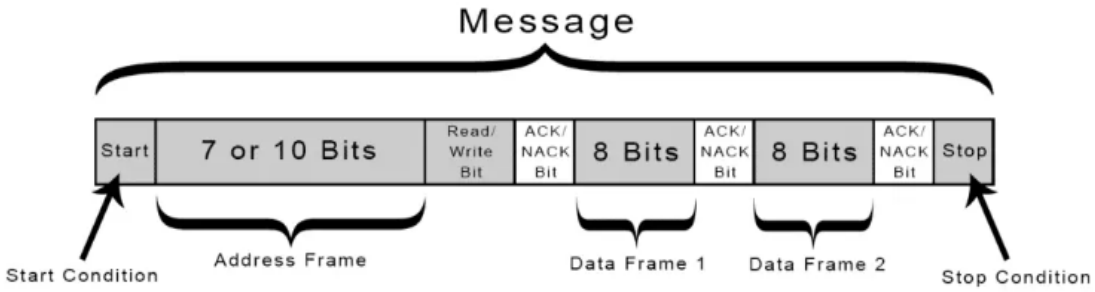

Figure 10: Khung tin nhắn I2C

Điều kiện bắt đầu: Dây SDA chuyển từ mức điện áp cao xuống mức điện áp thấp trước khi dây SCL chuyển từ mức cao xuống mức thấp.

Điều kiện dừng: Dây SDA chuyển từ mức điện áp thấp lên mức điện áp cao sau khi dây SCL chuyển từ mức thấp lên mức cao.

Khung địa chỉ: Đây là một chuỗi duy nhất gồm 7 hoặc 10 bit để xác định địa chỉ của thiết bị Slave mà Master muốn truyền thông.

Read/Write Bit: Đây là một bit đơn xác định liệu Master đang gửi dữ liệu đến Slave (mức điện áp thấp) hay yêu cầu dữ liệu từ Slave (mức điện áp cao).

ACK/NACK Bit: Mỗi khung dữ liệu trong một tin nhắn được theo sau bởi một bit ACK/NACK. Nếu khung địa chỉ hoặc khung dữ liệu được Slave nhận thành công, một bit ACK sẽ được trả lại cho Master từ Slave để xác nhận.

Giao tiếp I2C Arduino sử dụng các tín hiệu điện áp logic để đồng bộ hoạt động giữa các thiết bị trong mạng I2C và cung cấp một phương pháp đơn giản và hiệu quả để truyền thông giữa chủ (Master) và các thiết bị phụ (Slave).

1.4.3.Địa chỉ

Giao thức I2C không sử dụng các dòng chọn thiết bị như giao tiếp SPI, do đó nó cần một phương pháp khác để cho thiết bị nhận (Slave) biết rằng dữ liệu đang được gửi tới nó mà không phải là một thiết bị khác (Slave). I2C thực hiện điều này bằng cách sử dụng khung địa chỉ. Khung địa chỉ luôn là khung đầu tiên sau bit bắt đầu trong mỗi tin nhắn I2C.

Trong **giao tiếp I2C Arduino**, thiết bị chủ (Master) gửi địa chỉ của thiết bị nhận (Slave) mà nó muốn truyền thông tin tới. Tất cả các thiết bị nhận kết nối với thiết bị chủ sau đó so sánh địa chỉ được gửi từ thiết bị chủ với địa chỉ của chính nó. Nếu địa chỉ khớp, thiết bị nhận (Slave) sẽ gửi bit ACK (mức điện áp thấp) trở lại cho thiết bị chủ. Nếu địa chỉ không khớp, thiết bị nhận sẽ không thực hiện bất kỳ hành động nào và đường truyền dữ liệu (SDA) sẽ tiếp tục ở mức điện áp cao.

1.4.4.Đọc/Ghi Bit

Khi thiết bị chủ (Master) muốn đọc dữ liệu từ thiết bị Slave, nó sẽ gửi các bit yêu cầu đọc (Read) liên tiếp qua đường SDA. Sau mỗi bit yêu cầu đọc, thiết bị Slave sẽ gửi một bit dữ liệu trong khung dữ liệu và thiết bị Master sẽ gửi một bit ACK để xác nhận việc nhận dữ liệu.

Quá trình gửi và nhận các bit dữ liệu trong I2C được điều khiển bởi tín hiệu SCL được điều chỉnh bởi thiết bị Master. Các bit dữ liệu được truyền và nhận theo trình tự từ bit MSB (Most Significant Bit) đến bit LSB (Least Significant Bit).

1.4.5.Khung dữ liệu

Mỗi khung dữ liệu bao gồm 8 bit dữ liệu và một bit ACK/NACK (ACKnowledge/Not ACKnowledge) để xác nhận việc nhận dữ liệu. Bit ACK/NACK được gửi bởi thiết bị nhận sau khi nhận mỗi khung dữ liệu. Nếu thiết bị nhận nhận dữ liệu đúng, nó sẽ gửi bit ACK trở lại cho thiết bị truyền. Ngược lại, nếu có lỗi xảy ra, nó sẽ gửi bit NACK để yêu cầu truyền lại dữ liệu.

Các khung dữ liệu được gửi từ thiết bị Master đến thiết bị Slave hoặc ngược lại. Mỗi khung dữ liệu được gắn kết sau khung địa chỉ, xác định thiết bị Slave mà thiết bị Master muốn truyền dữ liệu tới.

Sau khi các khung dữ liệu được truyền hoặc nhận, một điều kiện dừng được tạo ra bằng cách thiết bị Master hoặc Slave chuyển đường dữ liệu (SDA) từ mức điện áp cao lên mức điện áp thấp trước khi SCL từ mức thấp lên mức cao.

1.4.6.Ưu và nhược điểm

a. <u>Ưu điểm:</u>

- Giao tiếp I2C chỉ sử dụng hai dây, SDA và SCL, giúp tiết kiệm số lượng chân kết nối.
- I2C cho phép kết nối nhiều thiết bị trong một mạng duy nhất, bằng cách sử dụng các địa chỉ duy nhất cho từng thiết bị.
- Giao thức I2C hỗ trợ nhiều tốc độ truyền thông, cho phép điều chỉnh tốc độ truyền dữ liệu phù hợp với yêu cầu của ứng dụng.
- I2C sử dụng các quy tắc truyền thông điều khiển và kiểm soát xung đột, giúp đảm bảo tính tin cậy và chính xác của dữ liệu truyền qua mạng.

b. <u>Nhược điểm:</u>

- So với các giao thức truyền thông khác như SPI hay UART, I2C có tốc độ truyền thông thấp hơn, điều này có thể ảnh hưởng đến hiệu suất chuyển dữ liệu trong các ứng dụng yêu cầu tốc độ cao.

- Do sử dụng dây truyền thông, khoảng cách truyền thông trong **giao tiếp I2C arduino** có hạn chế. Trong các ứng dụng yêu cầu kết nối xa hoặc chịu ảnh hưởng nhiễu, cần phải xem xét các giải pháp bù nhiễu và tăng cường tín hiệu.

- **Chuẩn giao tiếp I2C** yêu cầu việc triển khai đúng các tín hiệu điều khiển và quy tắc truyền thông, điều này có thể làm tăng phức tạp trong việc thiết kế phần cứng của hệ thống.

2. THIẾT KẾ HỆ THỐNG

2.1. Sơ đồ trình tự (Sequence Diagram)

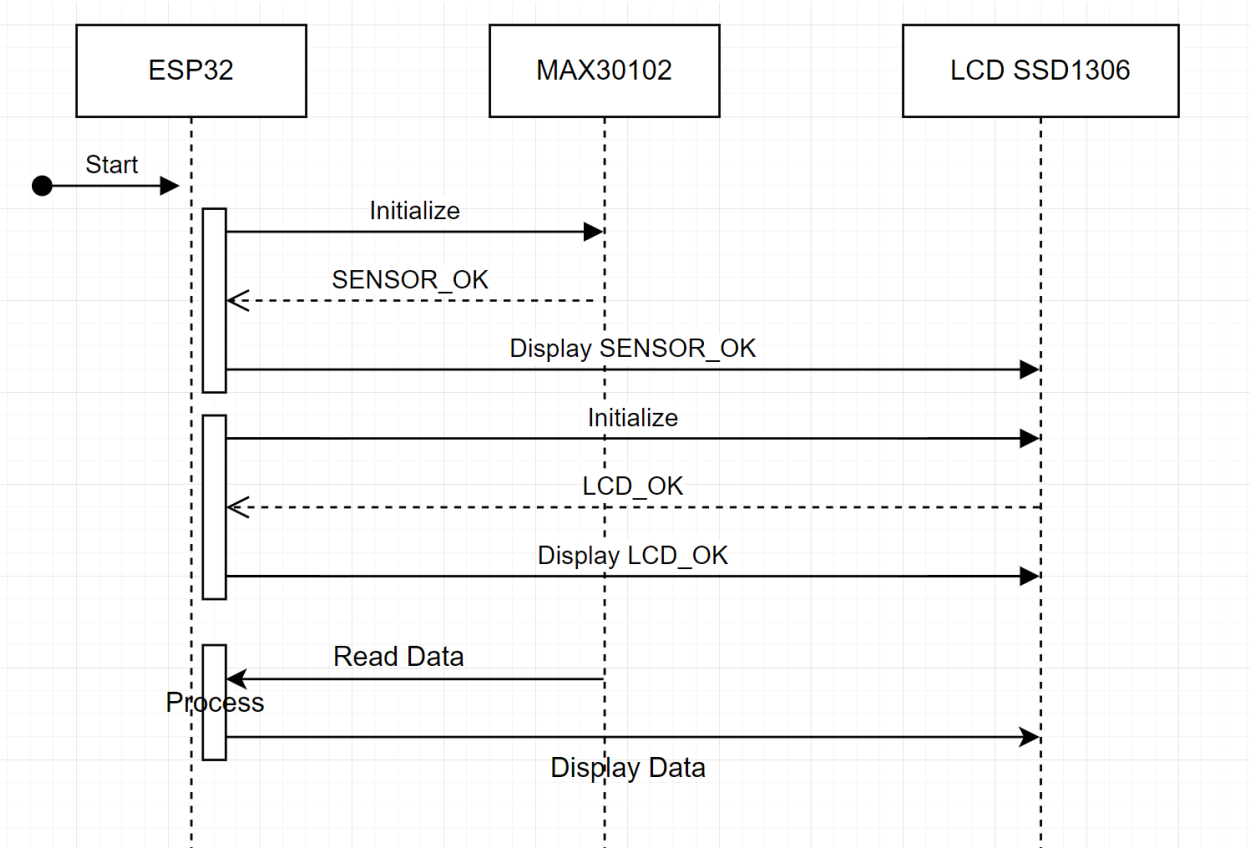

Figure 11: Sơ đồ trình tự

Trình tự hoạt động của mạch bắt đầu từ lúc khởi động sẽ bao gồm ba khối chính thực hiện theo tuần tự: Khối khởi động cảm biến MAX30102, khối khởi động màn hình LCD SSD1306, và cuối cùng là khối đọc và xử lí dữ liệu. Khi khởi động mạch sẽ tự động kiểm tra hệ thống xem có phát sinh lỗi nào hay không, nếu xảy ra lỗi sẽ có còi buzzer báo hiệu và người dùng sẽ phải kiểm tra lại cách đi dây rồi mới có thể khởi động lại.

2.1.1. Khởi động cảm biến MAX30102 (Initialize MAX30102)

Khi người dùng khởi động hệ thống, ESP32 sẽ bắt đầu khởi động cảm biến MAX30102, nếu:

- **Thành công:** Sẽ hiển thị tin nhắn "SENSOR OK" lên trên màn hình và beep buzzer 2 tiếng liên tiếp để báo hiệu thành công.
- **Thất bại:** Sẽ hiển thị tin nhắn "SENSOR ERROR" lên trên màn hình và beep buzzer 4 tiếng liên tiếp để báo hiệu thất bại.

2.1.2. Khởi động màn hình LCD SSD1306 (Initialize SSD1306)

Khi người dùng khởi động hệ thống, ESP32 sẽ bắt đầu khởi động màn hình LCD SSD1306, nếu:

- **Thành công:** Sẽ hiển thị tin nhắn "LCD OK" lên trên màn hình và beep buzzer 2 tiếng liên tiếp để báo hiệu thành công.
- **Thất bại**: Beep buzzer 4 tiếng liên tiếp để báo hiệu thất bại.

2.1.3. Đọc dữ liệu và xử lý

Sau khi thành công khởi động hệ thống, ESP32 sẽ bước vào phần hoạt động chính là đo lường số liệu từ cảm biến MAX30102 và xử lí số liệu, sau đó hiển thị kết quả xử lý được ra màn hình LCD SSD1306. Nếu trong quá trình đọc dữ liệu người dùng bỏ tay ra khỏi cảm biến hệ thống sẽ ngắt và hiển thị thông báo đặt lại tay để tiếp tục quá trình đo đạc.

2.2. *Sơ đồ hoạt động (Activity Diagram)*

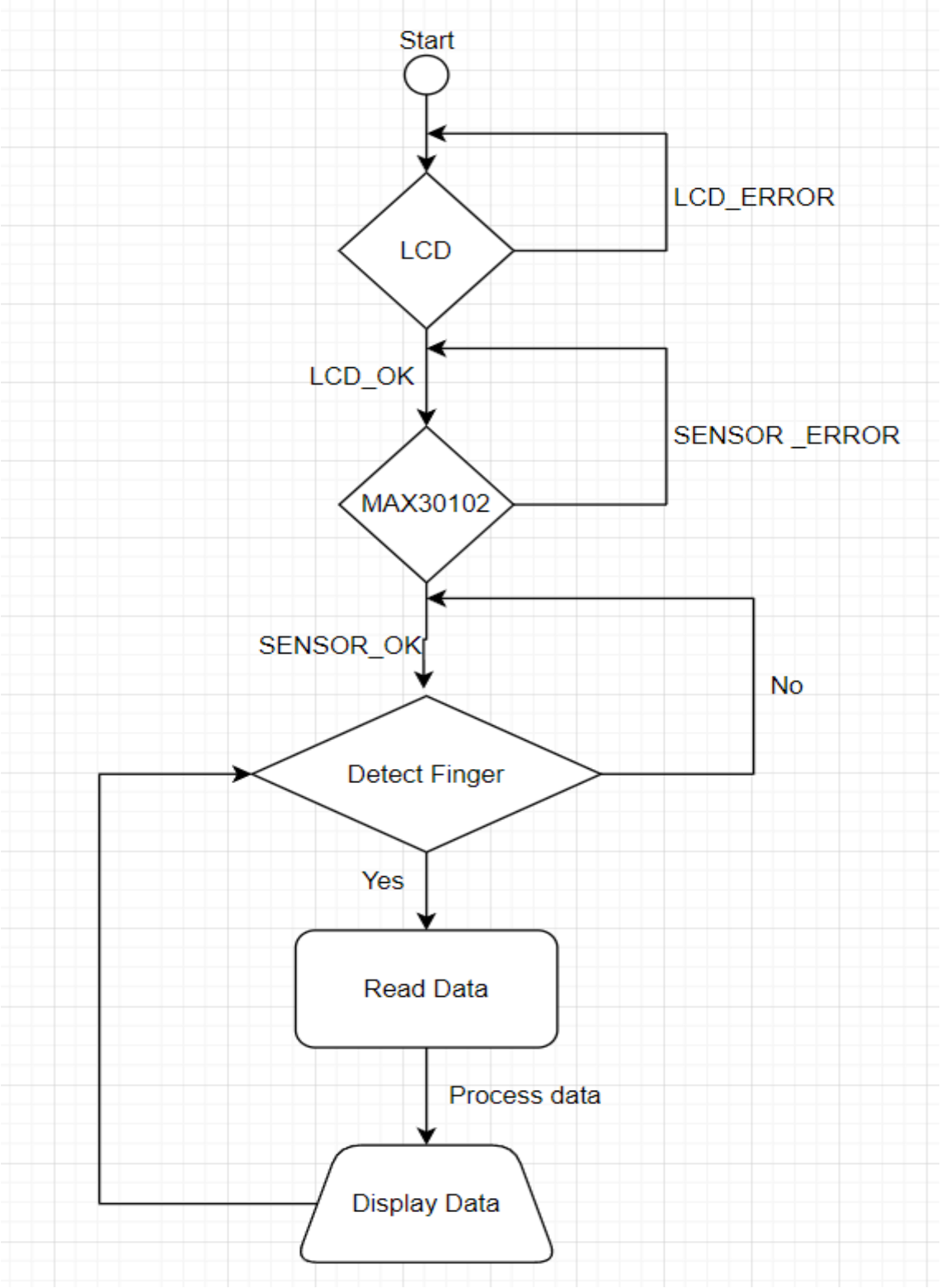

Figure 12: Sơ đồ hoạt động

Khi bắt đầu khởi động, hệ thống bước vào quá trình kiểm tra kết nối với màn hình LCD SSD1306 và cảm biến MAX30102. Nếu ở trong bước kiểm tra này có phát hiện ra lỗi kết nối với ngoại vi thì hệ thống sẽ dừng không chạy tiếp nữa, yêu cầu người dùng phải kiểm tra lại dây kết nối và khởi động lại.

Nếu thành công, hệ thống sẽ bắt đầu quá trình đo đạc khi người dùng đặt ngón tay lên mặt trên của cảm biến, dữ liệu sẽ được lấy mẫu trong khoảng vài giây rồi được cho vào thuật toán tính toán Nhịp tim (Heart Rate) và Nồng độ oxy trong máu (Spo2), và sau đó sẽ được hiển thị lên màn hình LCD SSD1306 cho người dùng.

2.3. Sơ đồ đi dây

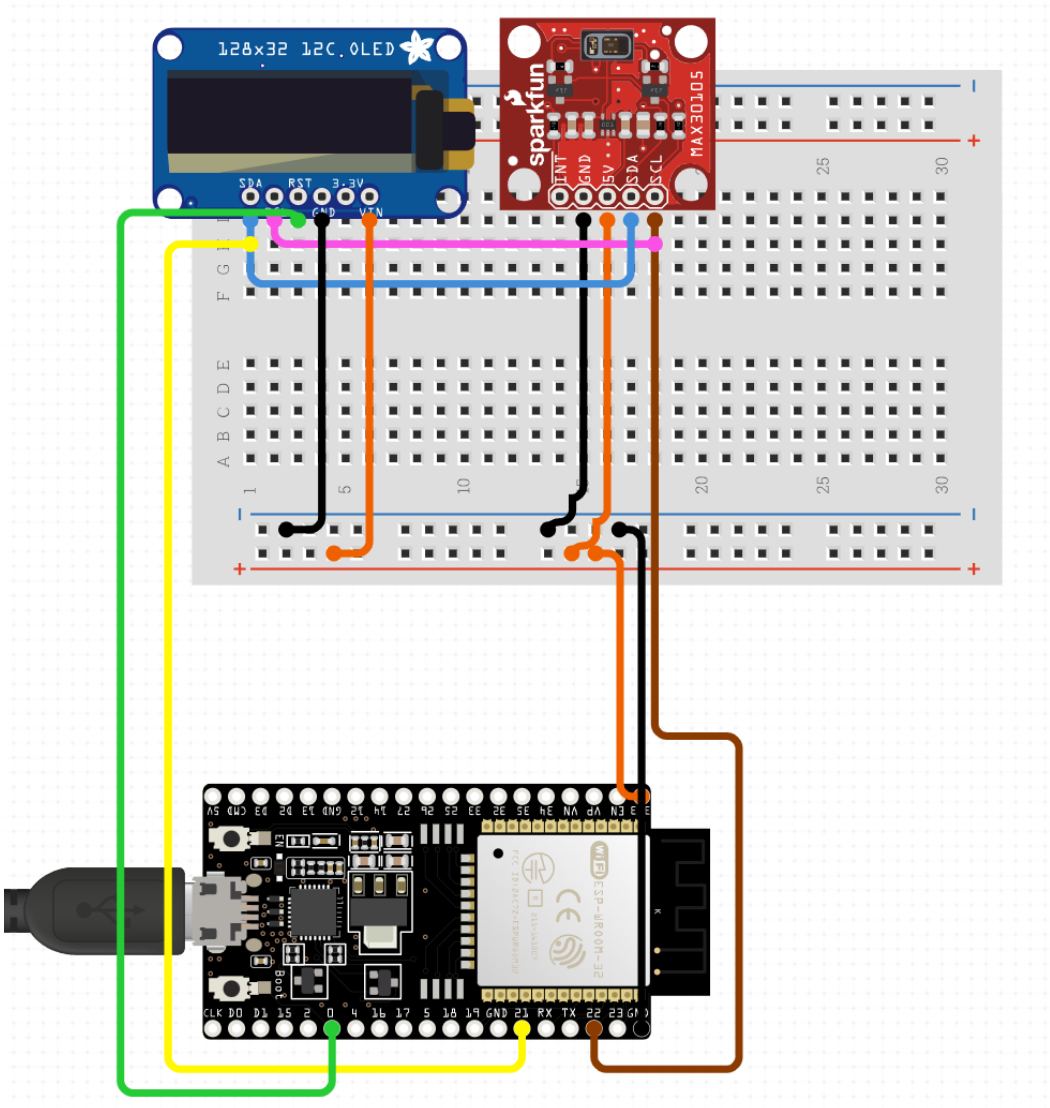

Figure 13: Sơ đồ đi dây

Vì điện áp hoạt động của cảm biến MAX30102 và LCD SSD1306 đều trong khoảng 3.3-5V nên em quyết định sử dụng nguồn 3.3V trực tiếp từ ESP32.

Cả 2 thiết bị đều sử dụng giao thức I2C để giao tiếp với ESP32 và chúng phân biệt nhau bằng địa chỉ 7-10bit trong code nên dây SDA và SCL của cả 2 có thể đấu chung và nối vào chân GPIO 21 (SDA) và GPIO 22 (SCL) của ESP32.

3. PHÁT TRIỂN VÀ THỬ NGHIỆM

3.1. Thiết lập môi trường phát triển

Các công cụ:

- Code Editor: Visual Studio Code
- Công cụ biên dịch: PlatformIO, gcc
- Ngôn ngữ lập trình: C/C++
- Quản lí mã nguồn: Github
- Mã nguồn: https://github.com/moenguyenx/HealthMonitor_ESP32.git
- Phần mềm thiết kế PCB: KiCad, Altium.

Framework và thư viện sử dụng:

- Framework: Arduino
- Adafruit SSD1306
- Adafruit GFX Library
- Sparkfun MAX3010x Pulse and Proximity Sensor Library

3.2. Phát triển Code

Do lựa chọn sử dụng framework Arduino và code bằng ngôn ngữ C++ nên em quyết định tiếp cận vấn đề bằng OOP (Lập trình hướng đối tượng), chia thành 2 khối chính là khối hiển thị và khối cảm biến. Khối hiển thị sẽ đảm nhiệm hiển thị các thông tin cần thiết lên màn hình OLED SSD1306, khối cảm biến sẽ đảm nhiệm việc đo lường dữ liệu từ cảm biến MAX30102 và thực hiện thuật toán tính toán nhịp tim và nồng độ oxy trong máu.

Việc chia nhỏ code thành các khối sẽ giúp cho việc Debug và phát triển code rộng hơn sau này dễ dàng hơn.

3.2.1. Khối hiển thị:

Khối hiển thị được định nghĩa dưới dạng class, khi vào chương trình chính sẽ khai báo một đối tượng LCD mới, đối tượng này có thể sử dụng tất cả các hàm được định nghĩa. Các hàm sẽ có những tham số để người dùng có thể truyền vào và hiển thị lên màn hình.

Mục đích là để gói gọn những bước code để có thể hiển thị thông tin lên trên màn hình, sắp xếp code không còn lộn xộn như việc sử dụng thư viện Adafruit.

```
class LCD
{
public:
    LCD();
    lcdStatus setupDisplay();
    void displayMessage(const char* message);
    void displayPercentage(float percentage);
    void displayReadings(int heartRate, int spo2);
    void displayHeartBeat(int heartRate, int spo2);
    void displayFingerMessage();
    void clear();
private
    Adafruit_SSD1306 display;
};
```

3.2.2.Khối cảm biến:

Khối cảm biến được định nghĩa dưới dạng class, khi vào chương trình chính sẽ khai báo một đối tượng cảm biến mới, đối tượng này có thể sử dụng tất cả các hàm được định nghĩa. Các hàm sẽ có những tham số để người dùng có thể truyền vào để có thể tính toán và xử lí số liệu.

```
class HeartRateSensor {
public:
    HeartRateSensor();
    sensorStatus begin();
    void readSamples(int32_t bufferLength, LCD &lcd);
    void continuousSampling();
    void calculate(int32_t bufferLength, int32_t* spo2, int8_t* validSPO2, int32_t*
heartRate, int8_t* validHeartRate);
    long readIR();
private:
    MAX30105 particleSensor;
    #if defined(__AVR_ATmega328P__) || defined(__AVR_ATmega168__)
    uint16_t irBuffer[100];
    uint16_t redBuffer[100];
    #else
    uint32_t irBuffer[100];     // IR Buffer for ESP32
    uint32_t redBuffer[100];    // Red Buffer for ESP32
    #endif };
```

3.2.3.Hàm hoạt động chính:

```
#include <Wire.h>
#include <Arduino.h>
#include "lcd.h"
#include "sensor.h"
#include "buzzer.h"

LCD             lcd;
HeartRateSensor sensor;

int32_t         bufferLength = 100;
int32_t         spo2;
int8_t          validSPO2;
int32_t         heartRate;
int8_t          validHeartRate;

void setup() {
    Serial.begin(115200);
    Wire.begin();
    if(lcd.setupDisplay() == LCD_OK)
    {
        lcd.displayMessage("Display OK");
    }
    delay(1000);
    lcd.displayMessage("System  OK");
    beepBuzzer(OK_BEEP);
    delay(1000);
    if(sensor.begin() == SENSOR_OK) {
        lcd.displayMessage("Sensor  OK");
    } else {
        lcd.displayMessage("Sensor  ERROR");
        delay(1000);
        lcd.displayMessage("");
        lcd.displayMessage("Startup ERROR");
        lcd.displayMessage("Check Wiring!");
        beepBuzzer(ERR_BEEP);
        for(;;);
    }
    delay(1000); }
```

```
void loop()
{
    lcd.clear();
    long irValue = sensor.readIR();
    if (irValue > 7000)
    {
        tone(BUZZER_PIN, 1000, BEEP_DURATION);
        sensor.readSamples(bufferLength, lcd);
        sensor.calculate(bufferLength, &spo2, &validSPO2, &heartRate,
&validHeartRate);
        while(1)
        {
            irValue = sensor.readIR();
            if (irValue <= 7000) break;
            sensor.continuousSampling();

            lcd.displayHeartBeat(heartRate, spo2);
            lcd.displayReadings(heartRate, spo2);
            tone(BUZZER_PIN, 1000, BEEP_DURATION);
            sensor.calculate(bufferLength, &spo2, &validSPO2, &heartRate,
&validHeartRate);
        }
    }
    else
    {
        lcd.displayFingerMessage();
    }
}
```

Trong phần chính, code sẽ chạy hàm setup() trước để có thể bắt đầu kết nối với các thiết bị ngoại vị, nếu không thành công sẽ dừng mọi hoạt động và yêu cầu người dùng kiểm tra lại kết nối. Nếu thành công sẽ chuyển sang hàm loop() để tiến hành quá trình đo đạc, khi cảm biến phát hiện có tay người dùng đặt lên sẽ bắt đầu lấy mẫu tín hiệu và đưa ra kết quả đo đạc được lên màn hình cho người dùng.

3.3. *Phát triển PCB*

3.3.1.Khối nguồn:

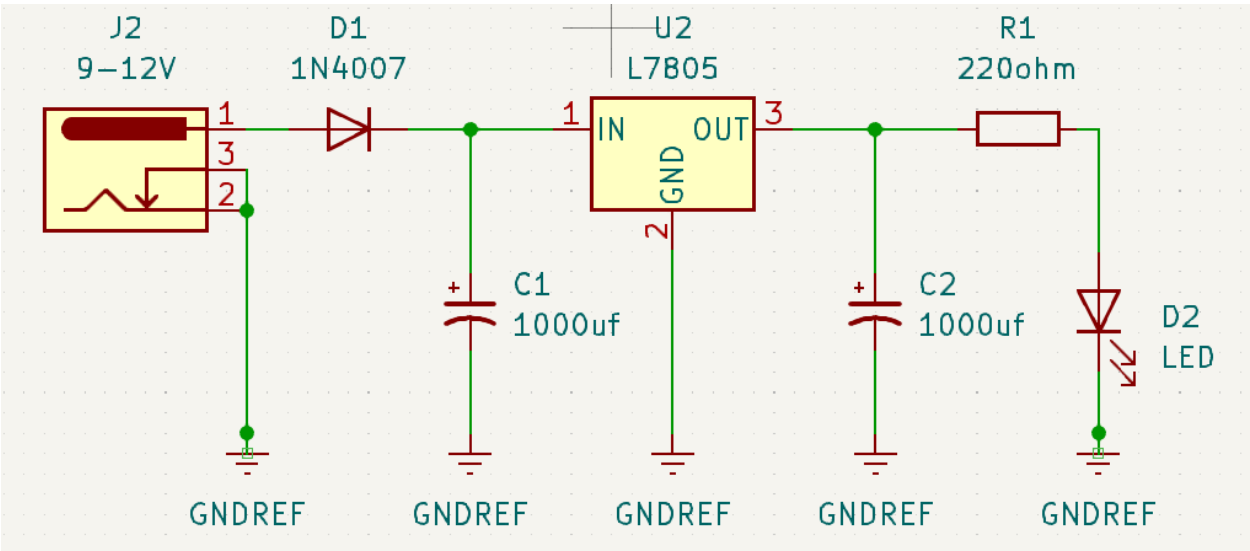

Figure 14: Sơ đồ khối nguồn

Khối nguồn được thiết kế để có thể sử dụng trực tiếp nguồn DC 9-12V, hạ áp xuống 5V bằng IC nguồn L7805 và lọc tín hiệu điện bằng 2 tụ hoá C1, C2 1000uf để có thể cung cấp nguồn cho ESP32, module cảm biến MAX30102 và màn hình OLED SSD1306.

3.3.2.Khối cảm biến và hiển thị:

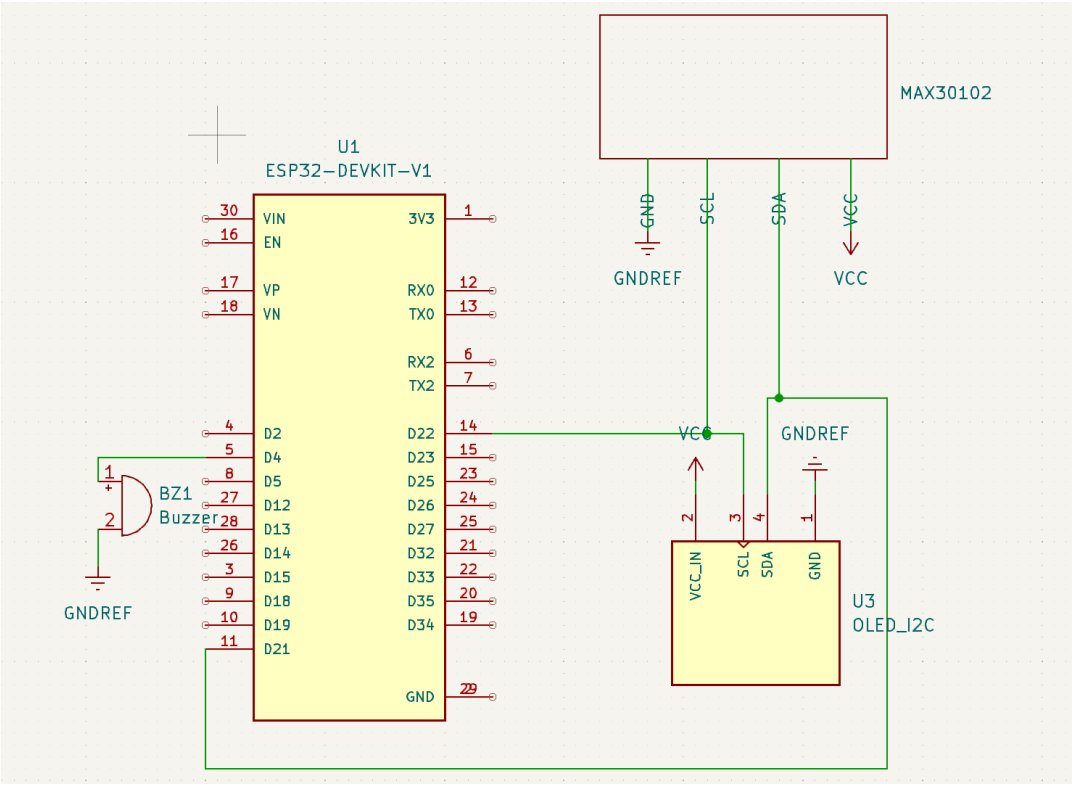

Figure 15: Sơ đồ khối cảm biến và hiển thị

Trong khối này sẽ bao gồm kết nối của vi xử lí ESP32, module cảm biến MAX30102, màn hình OLED SSD1306 và còi buzzer.

Còi Buzzer được nối vào chân D5 của vi xử lí ESP32 để điều khiển kêu báo hiệu.

Màn hình OLED SSD1306 và module cảm biến MAX30102 đều sử dụng giao thức I2C để có thể truyền nhận dữ liệu từ vi xử lí ESP32 nên chúng đều đi chung chân SDA vào chân D21 và chân SCL vào chân D22 của ESP32, việc phân biệt giữa hai thiết bị này với nhau sẽ được xử lí bởi địa chỉ I2C mà chúng kết nối với vi xử lí.

3.3.3.Vẽ mạch PCB:

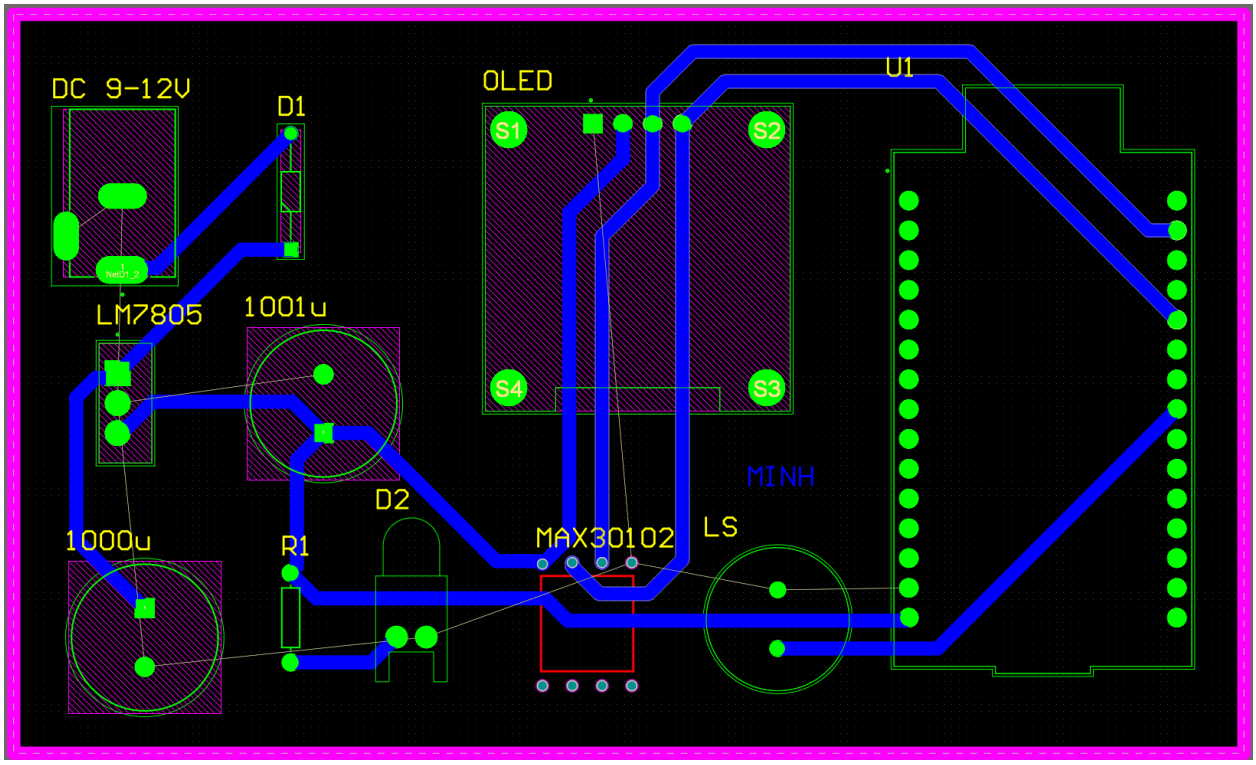

Figure 16: Sơ đồ đi dây PCB

Sau khi đã hoàn thành thiết kế Schematic, em tiến hành vẽ mạch in PCB trên Altium. Em quyết định làm mạch in một lớp để có thể tiết kiệm chi phí sản phẩm. Các đường mạch được đi chuẩn theo góc tù để có thể giảm thiểu trở kháng.

3.4. *Thử nghiệm*

Sau khi đã hoàn thành công đoạn thiết kế và code, em tiến hành nạp code vào ESP32 và đi dây theo như đúng sơ đồ thiết kế trên board trắng để chạy thử.

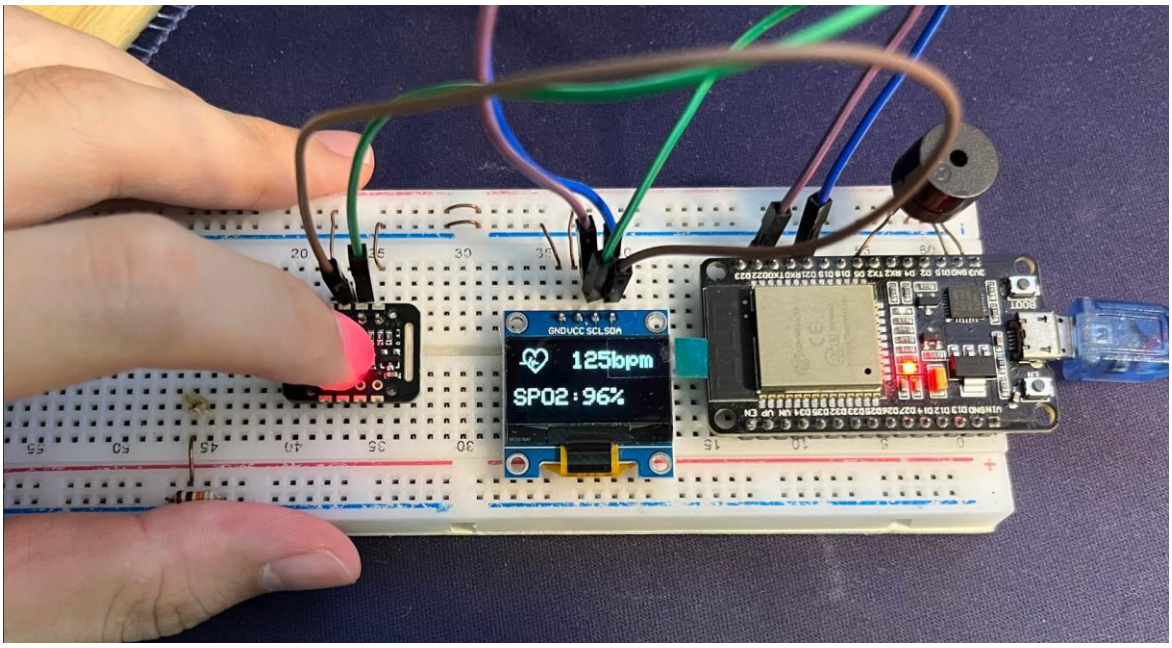

Figure 17: Hình ảnh mạch thực tế

Kết quả mạch chạy ổn định và đúng so với thiết kế ban đầu. Thuật toán đo lường nhịp tim và nồng độ oxy trong máu sẽ mất khoảng 15 đến 30 giây để có thể đưa ra kết quả đo ổn định cho người dùng.

Sau khi đã kiểm thử mạch chạy ổn định, em tiến hành đặt mạch PCB như đã thiết kế ở phần 3.3.

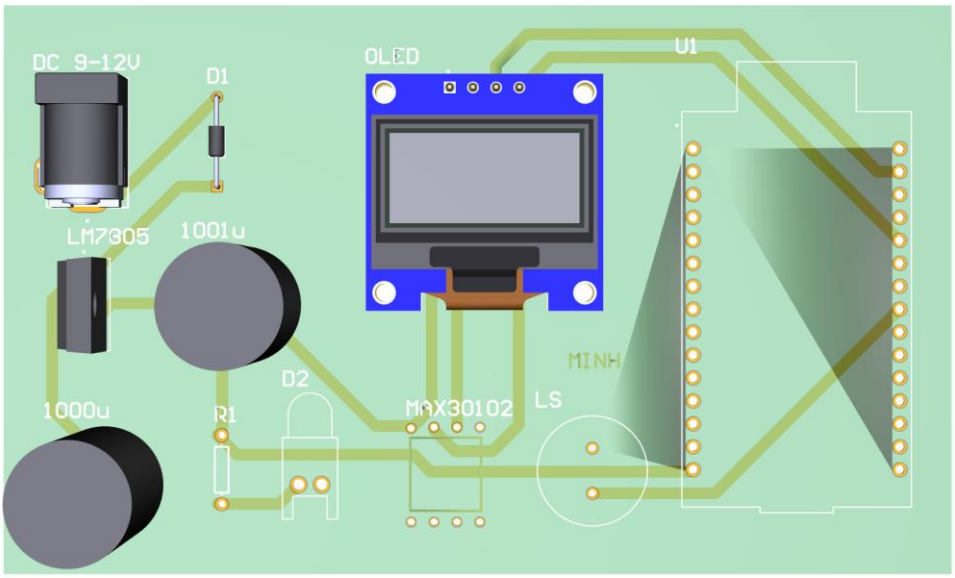

Figure 18: Hình ảnh mạch PCB

4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

4.1. Kết luận

Qua quá trình nghiên cứu và phát triển đề tài "Thiết kế mạch đo nhịp tim và nồng độ oxy trong máu", em đã học được nhiều kiến thức và kỹ năng quan trọng:

- **Kiến thức về lập trình nhúng:**
  - Hiểu rõ hơn về cách hoạt động của một chương trình trong hệ thống nhúng.
  - Nắm vững cách tối ưu hoá code sao cho có thể tái sử dụng lại và tiết kiệm tài nguyên phần cứng.
- **Hiểu biết về các cảm biến sinh học:**
  - Nắm vững nguyên lý hoạt động của các cảm biến đo nhịp tim và nồng độ oxy trong máu như cảm biến quang học, cảm biến xung điện, v.v.
  - Biết cách lựa chọn và tích hợp cảm biến vào mạch điện tử.
- **Xử lí tín hiệu:**
  - Kỹ thuật xử lý tín hiệu số để lọc nhiễu và phân tích dữ liệu từ các cảm biến.
  - Áp dụng các thuật toán phân tích tín hiệu để xác định chính xác nhịp tim và nồng độ oxy trong máu.
- **Thiết kế mạch điện tử:**
  - Kỹ năng thiết kế và xây dựng mạch điện tử bao gồm việc chọn linh kiện, tạo sơ đồ mạch và thiết kế PCB.
  - Kiểm tra và khắc phục sự cố của mạch điện tử để đảm bảo hệ thống hoạt động ổn định.
- **Quản lý và triển khai dự án:**
  - Kỹ năng lập kế hoạch và quản lý dự án, phân chia công việc, theo dõi tiến độ và xử lý các vấn đề phát sinh.

4.2. Hướng phát triển tối ưu

- **Cải thiện hiệu suất và Tối ưu hoá:**
  - Nâng cấp thuật toán để có thể đưa ra kết quả đo chính xác hơn.
  - Tối ưu hóa mã nguồn để tiết kiệm năng lượng hơn.
  - Tối ưu hoá mạch PCB nhỏ gọn và giảm thiểu nhiễu tín hiệu.
- **Mở rộng tính năng:**
  - Phát triển thêm các tính năng mới như cảnh báo nhịp tim quá cao, các cảnh báo khi phát hiện chỉ số sinh học trong cơ thể có dấu hiệu bất thường.
  - Kết nối không dây tới các web server để lưu lại thông tin chỉ số sinh học trong một thời gian dài để có thể phân tích sâu hơn về sức khoẻ.
- **Nâng cao trải nghiệm người dùng:**
  - Thiết kế lại PCB nhỏ dạng đồng đồ để người dùng có thể đeo và theo dõi các chỉ số thường xuyên.
  - Tích hợp AI phân tích sức khoẻ người dùng.

C. LỜI KẾT

Sau quá trình nghiên cứu và phát triển, đề tài "Thiết kế mạch đo nhịp tim và nồng độ oxy trong máu" đã đạt được những kết quả khả quan. Em đã thiết kế và chế tạo thành công một thiết bị có khả năng đo lường hai chỉ số quan trọng này với độ chính xác cao và dễ sử dụng. Thiết bị không chỉ đáp ứng được các yêu cầu kỹ thuật đặt ra ban đầu mà còn mở ra hướng phát triển mới cho các ứng dụng theo dõi sức khỏe cá nhân.

Quá trình thực hiện đồ án đã giúp em tích lũy được nhiều kiến thức và kỹ năng quý báu, từ việc nắm bắt công nghệ cảm biến, thiết kế mạch điện tử, đến phân tích và xử lý tín hiệu. Đặc biệt, em đã có cơ hội áp dụng các kiến thức lý thuyết vào thực hành, giải quyết những vấn đề phát sinh và học hỏi từ những thất bại trong quá trình phát triển sản phẩm.

Tuy nhiên, do giới hạn về thời gian và nguồn lực, thiết bị vẫn còn một số hạn chế cần được khắc phục trong tương lai. Em nhận thấy rằng cần có sự cải tiến về thiết kế mạch để giảm thiểu nhiễu tín hiệu, cũng như tối ưu phần mềm xử lý để tăng cường độ chính xác và tính ổn định của thiết bị.

Em xin gửi lời cảm ơn chân thành đến thầy Vũ Sinh Thượng và các bạn đồng nghiệp đã hỗ trợ và đóng góp ý kiến trong suốt quá trình thực hiện đồ án.

Hy vọng rằng, với những kết quả đạt được từ đề tài này, em có thể đóng góp một phần nhỏ vào sự phát triển của lĩnh vực công nghệ y tế và giúp cải thiện chất lượng chăm sóc sức khỏe cho cộng đồng.

Xin chân thành cảm ơn!

TÀI LIỆU THAM KHẢO

[1] DOIT ESP32 DEVKIT V1

[2] MAX30102 Pulse Oximeter and Heart Rate Sensor

[3] SSD1306 OLED Display