

ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN – ĐIỆN TỬ
BỘ MÔN ĐIỆN TỬ

-----o0o-----



ĐỒ ÁN MÔN HỌC

HỆ THỐNG IOT THEO DÕI CHỈ SỐ SINH TỒN CỦA BỆNH NHÂN
THÔNG QUA BLUETOOTH LOW ENERGY VÀ WIFI

GVHD: ThS. Trần Hoàng Quân

Sinh viên thực hiện	Mã số sinh viên
Nguyễn Đức Thắng	2213198
Lê Trường Doanh	2210478

TP. HỒ CHÍ MINH, THÁNG 5 NĂM 2025

LỜI CẢM ƠN

Trước tiên, xin gửi lời cảm ơn chân thành quý thầy cô khoa Điện – Điện Tử trường Đại Học Bách Khoa – Đại Học Quốc Gia HCM đã tận tình giảng dạy và truyền đạt những kiến thức nền tảng bổ ích trong suốt quá trình học tập.

Đặt biệt, xin chân thành cảm ơn thầy Trần Hoàng Quân - người trực tiếp chỉ bảo, hướng dẫn chúng em trong quá trình thực hiện đồ án này.

Mặc dù đã có những đầu tư nhất định trong quá trình thực hiện đồ án song cũng khó tránh khỏi những sai sót, chúng em kính mong nhận được ý kiến đóng góp của quý thầy cô để được hoàn thiện hơn trong các lần sau.

Chúng em xin chân thành cảm ơn!

Tp. Hồ Chí Minh, ngày 29 tháng 5 năm 2025.

Sinh viên

TÓM TẮT ĐỒ ÁN

Đồ án này trình bày về Hệ thống IoT được phát triển nhằm giám sát các chỉ số sinh tồn (nhịp tim và SpO2) của bệnh nhân từ xa thông qua công nghệ Bluetooth Low Energy (BLE) và Wi-Fi. Thiết bị đeo sử dụng cảm biến MAX30100 để đo dữ liệu và truyền về HUB trung tâm qua module JDY-24M. HUB tích hợp ESP8266 NodeMCU xử lý dữ liệu và cập nhật real-time lên Firebase Realtime Database. Kết quả thử nghiệm cho thấy hệ thống hoạt động ổn định với thời gian truyền dữ liệu mỗi 1 giây, đồng thời đảm bảo độ chính xác và khả năng mở rộng cho các ứng dụng y tế thông minh.

MỤC LỤC

1. GIỚI THIỆU	1
1.1 Tổng quan	1
1.2 Nhiệm vụ đề tài.....	1
1.3 Phân chia công việc trong nhóm.....	1
2. LÝ THUYẾT	2
2.1 Vi điều khiển ATmega328P	2
2.2 Kit ESP8266 NodeMCU.....	3
2.3 Module Bluetooth 5.0 JDY-24M.....	4
2.4 Màn Hình LCD 16*2 Và Mạch Chuyển Đổi Giao Tiếp I2C	5
2.6 Firebase	6
2.6 Giao Thức UART	6
2.7 Giao Thức I2C	7
3 THIẾT KẾ VÀ THỰC HIỆN PHẦN CỨNG.....	9
4 THIẾT KẾ VÀ THỰC HIỆN PHẦN MỀM	15
4.1 Thiết Bị Đeo Của Bệnh Nhân.....	15
4.2 HUB trung tâm	16
4.3 Thiết Bị hiển thị lên LCD	18
5 Kết nối module Bluetooth thông qua AT Command	19
5.1 Mục tiêu kết nối.....	19
5.2 Quy trình thiết lập kết nối thông qua tập lệnh AT Command	19
6 Cấu hình Firebase	22
7 KẾT QUẢ THỰC HIỆN	23
7.1 Thiết bị đeo	23
7.2 HUB và Firebase	25
7.3 Hiển thị LCD	26

8	KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.....	28
8.1	Kết luận.....	28
8.2	Hướng phát triển.....	28
9	TÀI LIỆU THAM KHẢO.....	29
10	PHỤ LỤC	30

DANH SÁCH HÌNH MINH HỌA

Hình 2-1 Sơ đồ IO và chức năng của ATmega328P.....	2
Hình 2-2: Sơ đồ IO và chức năng của kit ESP8266 NodeMCU	3
Hình 2-3 Module Bluetooth 5.0 JDY-24M và sơ đồ IO.....	4
Hình 2-4: Sơ đồ IO của LCD 16*2.....	5
Hình 2-5 Mạch chuyển đổi giao tiếp I2C	5
Hình 2-6 Nền tảng Realtime Database của Firebase	6
Hình 2-7 Giao Thức UART.....	7
Hình 2-8 Giao Thức I2C.....	8
Hình 3-1 Sơ đồ khối tổng quát	11
Hình 3-2 Sơ đồ khối của thiết bị đeo cho bệnh nhân	12
Hình 3-3 Khối HUB Trung Tâm	13
Hình 3-4 Khối thiết bị hiển thị	13
Hình 4-1 Lưu Đồ giải thuật khối 1	15
Hình 4-2 Lưu đồ giải thuật khối 2	17
Hình 4-3 Lưu đồ giải thuật khối 3	18
Hình 4-4 Cấu hình Module Slave trên cửa sổ Serial Monitor.....	20
Hình 4-5 Cấu hình Module Master trên cửa sổ Serial Monitor.....	21
Hình 4-6 Cấu hình Firebase.....	22
Hình 7-1 Phần cứng thiết bị đeo cho bệnh nhân	24
Hình 7-2 kết quả gửi chỉ số sinh tồn cho HUB trung tâm.....	24
Hình 7-3 Phần cứng thiết bị HUB trung tâm.....	25
Hình 7-4 kết quả gửi chỉ số sinh tồn của bệnh nhân lên Firebase	26
Hình 7-5 Phần cứng của thiết bị hiển thị và kết quả hiển thị lên LCD	27

DANH SÁCH BẢNG SỐ LIỆU

Bảng 1.1 Bảng phân chia công việc	1
Bảng 7.1 Sơ đồ kết nối Module Bluetooth 5.0 JDY-24M ở HUB trung tâm.....	25
Bảng 7.2 Sơ đồ kết nối LCD 16*2 và mạch chuyển đổi giao tiếp I2C	26

1. GIỚI THIỆU

1.1 Tổng quan

Hệ thống IoT theo dõi chỉ số sinh tồn của bệnh nhân được thiết kế nhằm giám sát các thông số sức khỏe quan trọng như nhịp tim và nồng độ Oxide trong máu SpO2 từ xa thông qua công nghệ Bluetooth Low Energy (BLE) và Wi-fi. Hệ thống hướng đến mục tiêu cung cấp giải pháp y tế thông minh, giúp theo dõi liên tục và kịp thời phát hiện các bất thường về sức khỏe, đặc biệt hữu ích trong các bệnh viện hoặc cho bệnh nhân tự chăm sóc tại nhà. Ứng dụng IoT trong lĩnh vực này không chỉ nâng cao hiệu quả chẩn đoán mà còn giảm thiểu rủi ro do chậm trễ thông tin.

1.2 Nhiệm vụ đề tài

Nội dung 1: Thiết kế thiết bị đeo trên người bệnh

- Sử dụng cảm biến MAX30100 để đo nhịp tim và SpO2.
- Tích hợp module Bluetooth JDY-24M để truyền dữ liệu đến HUB trung tâm.

Nội dung 2: Phát triển HUB trung tâm

- Kết nối dữ liệu từ thiết bị đeo qua BLE và truyền lên đám mây (Firebase Realtime Database) thông qua Wi-fi bằng ESP8266 NodeMCU.

Nội dung 3: Xây dựng hệ thống lưu trữ và hiển thị

- Lưu trữ dữ liệu trên Firebase Realtime Database.
- Xây dựng thiết bị hiển thị thông số trên màn hình LCD.

1.3 Phân chia công việc trong nhóm

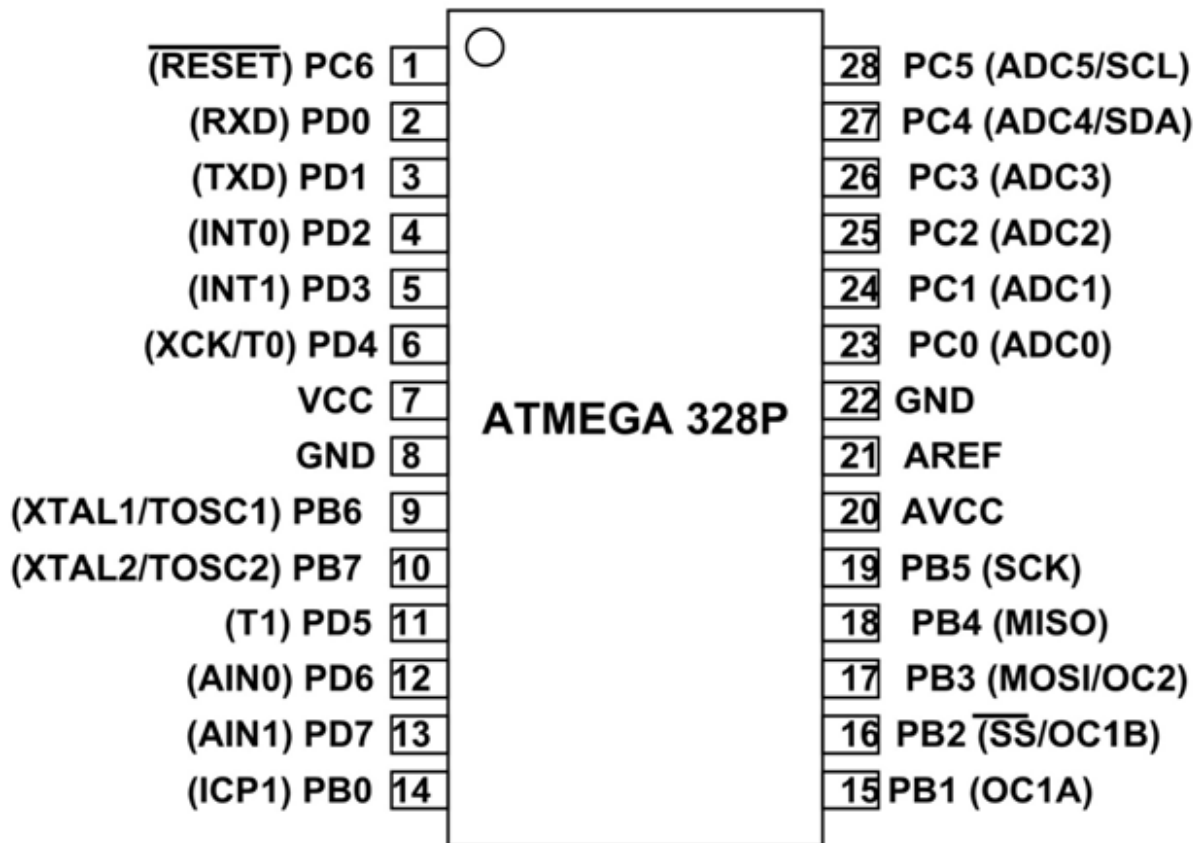
Bảng 1.1 Bảng phân chia công việc

Công việc	Người chịu trách nhiệm
Thiết kế code cho thiết bị đeo, kết nối module Bluetooth	Trường Doanh
Thiết kế code cho HUB trung tâm và thiết bị hiển thị lên LCD	Đức Thắng

2. LÝ THUYẾT

2.1 Vi điều khiển ATmega328P

ATmega328P là một bộ vi điều khiển tiên tiến và nhiều tính năng. Nó là một trong những vi điều khiển nổi tiếng của Atmel vì nó được sử dụng trong bo mạch arduino UNO. Nó là một bộ vi điều khiển thuộc họ vi điều khiển megaMVR của Atmel (Cuối năm 2016, Atmel được Microchip Technology Inc mua lại). Các vi điều khiển được sản xuất trong họ megaMVR được thiết kế để xử lý các bộ nhớ chương trình lớn



Hình 2-1 Sơ đồ IO và chức năng của ATmega328P

và mỗi vi điều khiển trong họ này chứa lượng ROM, RAM, các chân I / O và các tính năng khác nhau và được sản xuất với các chân đầu ra khác nhau, từ 8 chân đến hàng trăm chân.

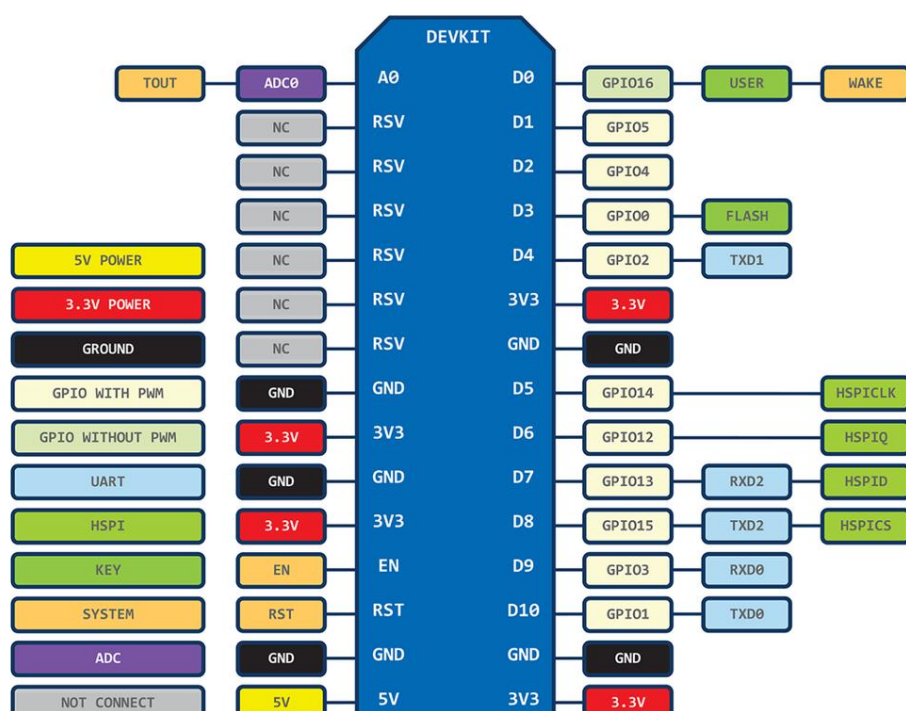
Mạch bên trong của ATmega328P được thiết kế với tính năng tiêu thụ dòng điện thấp. Con chip này chứa 32 kilobyte bộ nhớ flash trong, 1 kilobyte EEPROM và 2 kilobyte SRAM. EEPROM và bộ nhớ flash là bộ nhớ lưu thông tin và thông tin đó vẫn

thoát ra mỗi khi nguồn điện bị ngắt nhưng SRAM là bộ nhớ chỉ lưu thông tin cho đến khi có điện và khi ngắt nguồn điện tất cả thông tin được lưu trong SRAM sẽ bị xóa.

2.2 Kit ESP8266 NodeMCU

ESP8266 NodeMCU là 1 kit wifi phát triển mạnh mẽ chuyên dành cho các dự án IoT, được xây dựng dựa trên nền chip wifi SoC ESP8266 với thiết kế dễ sử dụng và đặc biệt có thể sử dụng trực tiếp trình biên dịch Arduino IDE để lập trình và nạp code. Với khả năng kết nối wifi tích hợp sẵn, kit có thể dễ dàng giao tiếp với internet mà không cần bất kỳ phần cứng nào khác.

NodeMCU ESP8266 sử dụng vi điều khiển ESP8266 với CPU 32-bit Tensilica Xtensa LX106 RISC, có tốc độ xung nhịp 80 MHz (có thể tăng lên 160 MHz). Bộ nhớ gồm 128 KB RAM và 4 MB Flash, giúp lưu trữ chương trình và dữ liệu hiệu quả. Nó hỗ trợ Wi-Fi chuẩn 802.11 b/g/n, có thể hoạt động ở chế độ AP, STA hoặc AP+STA. Điện áp hoạt động 3.3V, với khả năng giao tiếp UART, SPI, I2C. Ngoài ra, NodeMCU có 16 chân GPIO, giúp kết nối linh hoạt với các thiết bị ngoại vi. Với những đặc điểm này, NodeMCU ESP8266 trở thành một lựa chọn lý tưởng cho các dự án IoT cần kết nối mạng và xử lý dữ liệu nhanh chóng.

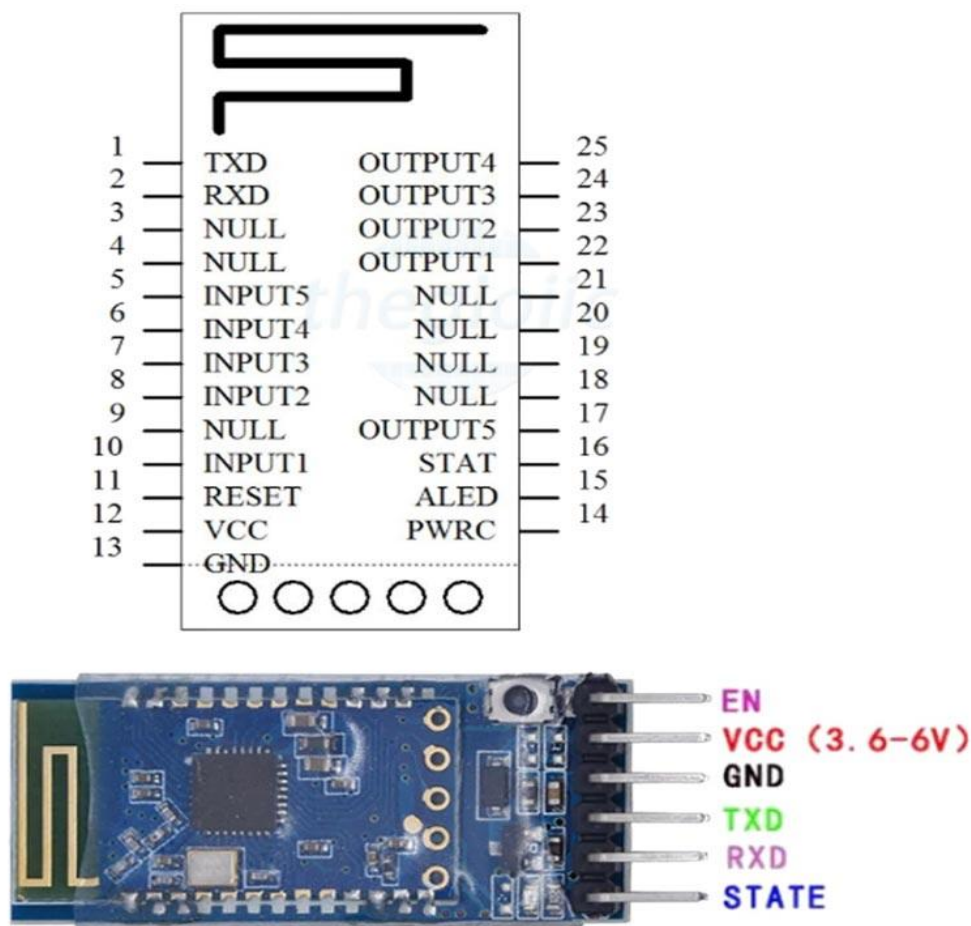


Hình 2-2: Sơ đồ IO và chức năng của kit ESP8266 NodeMCU

2.3 Module Bluetooth 5.0 JDY-24M

Module Bluetooth 5.0 JDY-24M là một mạch thu phát Bluetooth tiên tiến, hỗ trợ các chế độ Master-Slave, BLE, iBeacon, và MESH Network. Với khả năng truyền nhận theo mạng MESH, module này có thể kết nối lên đến 65.280 thiết bị, giúp mở rộng phạm vi ứng dụng trong các hệ thống IoT.

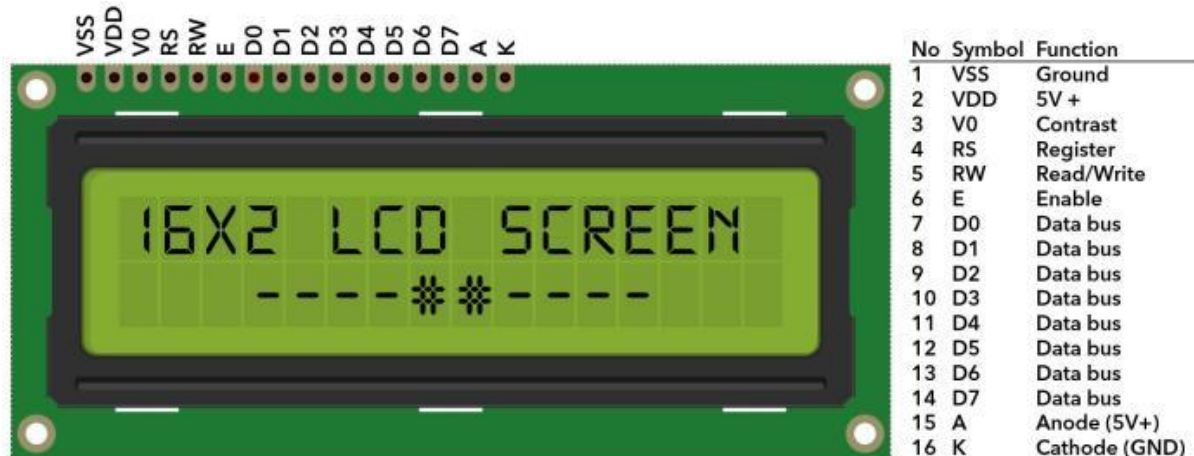
JDY-24M hoạt động với điện áp 3.3~5VDC, giao tiếp UART với chuẩn kết nối Bluetooth 5.0, tần số sóng 2.4Ghz. tương thích với hệ điều hành Windows / Android / IOS. Khoảng cách truyền nhận tối đa lên tới 80m



Hình 2-3 Module Bluetooth 5.0 JDY-24M và sơ đồ IO

2.4 Màn Hình LCD 16*2 Và Mạch Chuyển Đổi Giao Tiếp I2C

Màn hình text LCD16x2 sử dụng driver HD44780, có khả năng hiển thị 2 dòng với mỗi dòng 16 ký tự, màn hình có độ bền cao, rất phổ biến, nhiều code mẫu và dễ sử dụng thích hợp cho những người mới học và làm dự án. màn hình LCD 16×2 hoạt động với điện áp 5V, có 16 chân kết nối, trong đó bao gồm các chân điều khiển, dữ liệu, nguồn và điều chỉnh độ tương phản. Nó hỗ trợ hiển thị ký tự ASCII, có thể tùy



Hình 2-4: Sơ đồ IO của LCD 16*2

chỉnh độ sáng thông qua điện trở biến thiên. LCD 16×2 có thể giao tiếp với vi điều khiển thông qua giao thức song song, giúp truyền dữ liệu nhanh chóng.

Mạch chuyển đổi giao tiếp I2C cho LCD là một module giúp đơn giản hóa việc kết nối màn hình LCD (thường là LCD 16x2 hoặc 20x4) với vi điều khiển thông qua giao tiếp I2C. Thay vì sử dụng tối thiểu **6 chân** để kết nối với LCD (RS, EN, D7, D6, D5, D4), module này chỉ cần 2 chân (SDA và SCL), giúp tiết kiệm tài nguyên phần cứng và giảm độ phức tạp trong lập trình.



Hình 2-5 Mạch chuyển đổi giao tiếp I2C

Về thông số kỹ thuật, mạch chuyển đổi I2C hoạt động với điện áp 2.5V - 6V DC, hỗ trợ các loại LCD sử dụng driver HD44780 (LCD 1602, LCD 2004, ...) và có địa chỉ I2C mặc định 0x27 (có thể điều chỉnh bằng ngắn mạch chân A0/A1/A2). Ngoài ra, module còn tích hợp biến trở xoay để điều chỉnh độ tương phản của LCD và jump chốt để bật/tắt đèn nền.

2.6 Firebase

Firebase là một nền tảng phát triển ứng dụng do Google cung cấp, giúp các lập trình viên xây dựng và quản lý ứng dụng web và di động một cách dễ dàng. Firebase hoạt động theo mô hình Backend-as-a-Service (BaaS), nghĩa là nó cung cấp các dịch vụ backend mà không cần lập trình viên phải tự xây dựng hệ thống máy chủ.

Một trong những tính năng quan trọng của Firebase là Realtime Database, cho phép lưu trữ và đồng bộ dữ liệu theo thời gian thực giữa các thiết bị. Ngoài ra, Firebase còn hỗ trợ Authentication, giúp xác thực người dùng bằng nhiều phương thức như Google, Facebook, Email, Số điện thoại, v.v. Firebase cũng cung cấp Cloud Firestore, một cơ sở dữ liệu NoSQL mạnh mẽ, giúp quản lý dữ liệu linh hoạt hơn.

Firebase được sử dụng rộng rãi trong các ứng dụng IoT, trò chơi, ứng dụng thương mại điện tử, và hệ thống quản lý dữ liệu nhờ khả năng mở rộng tốt và tích hợp dễ dàng với nhiều nền tảng.



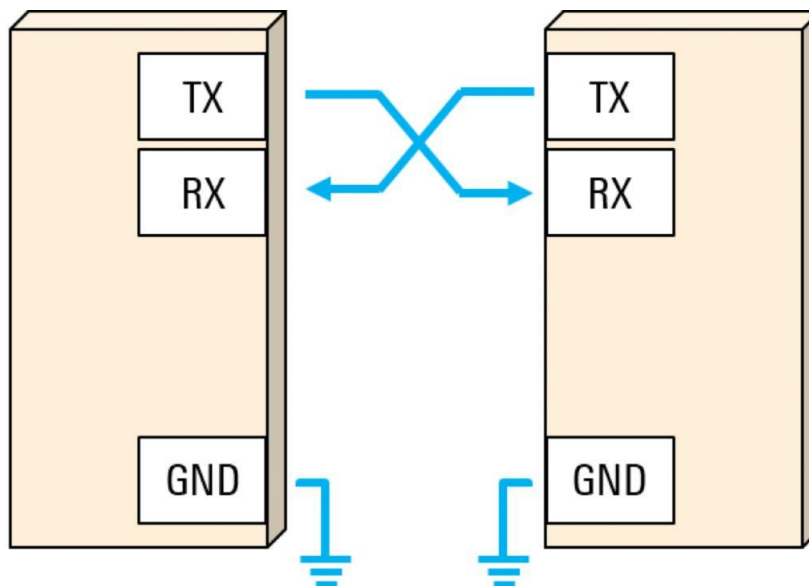
Hình 2-6 Nền tảng Realtime Database của Firebase

2.6 Giao Thức UART

UART, hay Universal Asynchronous Receiver-Transmitter, là một giao thức truyền thông đơn giản và được sử dụng rộng rãi, hoạt động không đồng bộ. Nó đạt được truyền dữ liệu bằng cách sử dụng các bit bắt đầu và dừng để đánh dấu sự bắt đầu và kết thúc của mỗi gói dữ liệu, đảm bảo đồng bộ hóa phù hợp mà không cần đường xung nhịp dùng chung.

Thiết bị truyền chuẩn bị dữ liệu để gửi đi bằng cách chuyển đổi dữ liệu thành luồng bit nối tiếp. Một bit bắt đầu (thường là tín hiệu logic thấp) được truyền đi để báo hiệu bắt đầu khung dữ liệu. Các bit dữ liệu được truyền tuần tự, từng bit một, thường từ bit ít quan trọng nhất (LSB) đến bit quan trọng nhất (MSB). Nếu kiểm tra chẵn lẻ được bật, một bit chẵn lẻ sẽ được truyền đi để đảm bảo tính toàn vẹn của dữ liệu. Một hoặc nhiều bit dừng (thường là tín hiệu logic cao) được truyền đi để chỉ ra sự kết thúc của khung dữ liệu. Sau bit dừng, đường truyền thông tin trở về trạng thái nhàn rỗi (thường là tín hiệu logic cao). Thiết bị nhận phát hiện bit bắt đầu, lấy mẫu các bit dữ liệu đến ở tốc độ truyền đã thỏa thuận và tái tạo lại dữ liệu gốc. Dữ liệu nhận được sau đó sẽ được xử lý **bởi thiết bị nhận**.

2.7 Giao Thức I2C



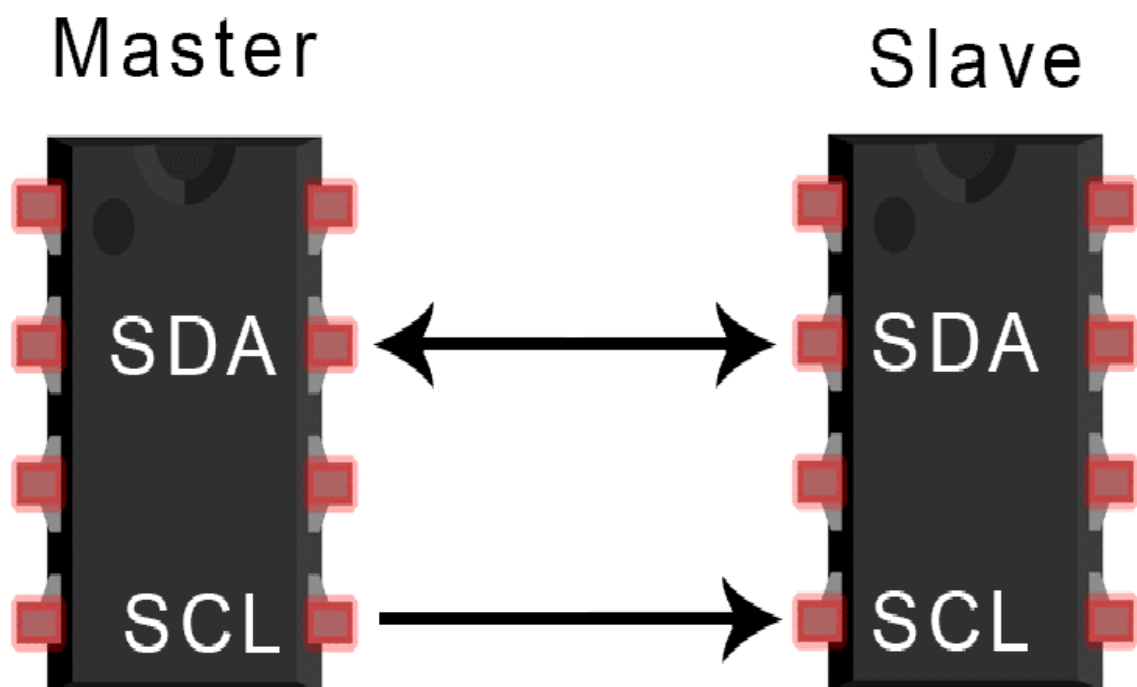
Hình 2-7 Giao Thức UART

Giống như giao tiếp UART, I2C chỉ sử dụng hai dây để truyền dữ liệu giữa các thiết bị:

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

I2C là một giao thức truyền thông nối tiếp, vì vậy dữ liệu được truyền từng bit dọc theo một đường duy nhất (đường SDA).

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. Điều này thực sự hữu ích khi bạn muốn có nhiều hơn một vi điều khiển ghi dữ liệu vào một thẻ nhớ duy nhất hoặc hiển thị văn bản trên một màn hình LCD.



Hình 2-8 Giao Thức I2C

3 THIẾT KẾ VÀ THỰC HIỆN PHẦN CỨNG

3.1 Yêu cầu thiết kế

- **Yêu cầu chung**

Hệ thống đọc dữ liệu nhịp tim từ cảm biến MAX30100, truyền dữ liệu qua Bluetooth, lưu trữ trên Firebase Realtime Database, và hiển thị thông tin lên màn hình LCD 16×2.

Đảm bảo độ trễ thấp trong quá trình truyền dữ liệu và hiển thị.

Các module giao tiếp với nhau theo các chuẩn I2C, UART, WiFi.

Hệ thống phải hoạt động ổn định, tiêu thụ năng lượng tối ưu, và có khả năng mở rộng.

- **Yêu cầu từng khối**

- Khối 1: Đọc dữ liệu từ cảm biến và truyền qua Bluetooth

- Cảm biến MAX30100 giao tiếp với vi xử lý qua I2C.
- Vi xử lý xử lý dữ liệu nhận được và truyền tới module Bluetooth.
- Module Bluetooth được cấu hình ở chế độ Master, thực hiện gửi dữ liệu đến khối 2.
- Chu kỳ đọc dữ liệu: 1 giây/lần để đảm bảo cập nhật thông tin liên tục.

- Khối 2: Nhận dữ liệu và gửi lên Firebase

- Module Bluetooth ở chế độ Slave, nhận dữ liệu từ Module Bluetooth master.
- ESP giao tiếp với module Bluetooth qua UART, đọc dữ liệu và gửi lên Firebase Realtime Database.
- Dữ liệu được cập nhật liên tục mỗi giây để đảm bảo theo dõi nhịp tim trong thời gian thực.

- Khối 3: Hiển thị dữ liệu từ Firebase lên LCD

- ESP kết nối với Firebase, đọc dữ liệu nhịp tim đã lưu trữ.
- Dữ liệu được hiển thị lên màn hình LCD 16×2
- Cập nhật dữ liệu liên tục mỗi giây, đảm bảo thông tin nhịp tim luôn hiển thị chính xác.

3.2 Phân Tích Thiết Kế

Phương án 1: sử dụng một cảm biến MAX30100, hai chip ATmega328P, hai module Bluetooth HC-05, một module Wi-Fi ESP8266-01S, một ESP8266 và sáu LED 7 đoạn. Trong phương án này, khối 1 sử dụng cảm biến MAX30100 giao tiếp I2C với chip ATmega328P và module Bluetooth HC-05. Khối 2 sử dụng module Bluetooth HC-05 giao tiếp UART với chip ATmega328P và module Wi-Fi ESP8266-01S. Khối 3 sử dụng ESP8266 và sáu LED 7 đoạn để hiển thị dữ liệu.

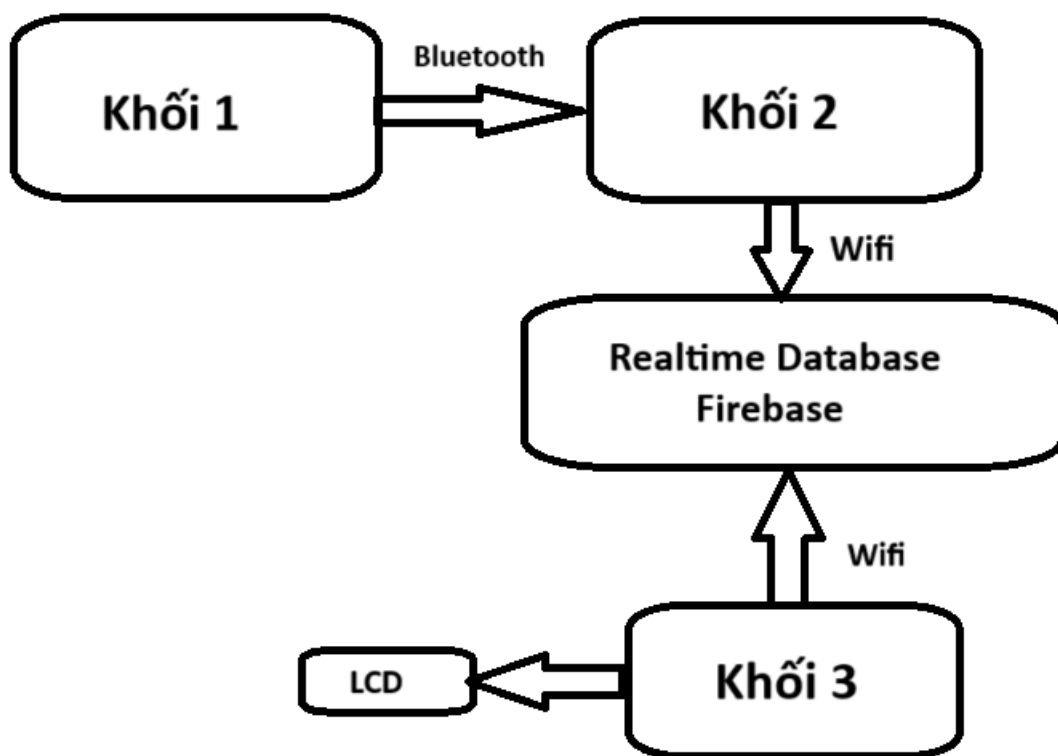
Phương án 1 sử dụng ATmega328P, một vi điều khiển phổ biến, dễ lập trình và có nhiều tài liệu hỗ trợ. Module Bluetooth HC-05 hoạt động ở chế độ Master-Slave, phù hợp với truyền dữ liệu đơn giản. Module Wi-Fi ESP8266-01S hỗ trợ kết nối với Firebase và giúp hệ thống hoạt động ổn định. LED 7 đoạn có thiết kế đơn giản, dễ hiển thị số liệu. Nhược điểm của phương án này là HC-05 chỉ hỗ trợ Bluetooth 2.0, tốc độ truyền dữ liệu thấp hơn JDY-24M. ESP8266-01S có ít GPIO, làm hạn chế khả năng mở rộng hệ thống. LED 7 đoạn chỉ hiển thị số, không thể hiển thị chữ như LCD 16x2. ATmega328P có hiệu suất xử lý thấp hơn ESP8266, có thể gây hạn chế khi xử lý dữ liệu nhanh.

Phương án 2: Sử dụng một cảm biến MAX30100, một chip ATMEGA328P, hai ESP8266 NodeMCU, hai Module Bluetooth 5.0 JDY-24M, một LCD 16*2 và một mạch chuyển đổi giao tiếp I2C cho LCD. Khối 1 gồm cảm biến MAX30100 giao tiếp I2C với vi điều khiển ATmega328P và module Bluetooth JDY-24M. ATmega328P có nhiệm vụ đọc dữ liệu từ cảm biến và truyền qua Bluetooth đến khối 2. Khối 2 sử dụng ESP8266 NodeMCU giao tiếp với module Bluetooth JDY-24M ở chế độ Slave để nhận dữ liệu từ khối 1. Sau đó, ESP8266 gửi dữ liệu lên Firebase Realtime Database mỗi giây để lưu trữ và đồng bộ. Khối 3 gồm ESP8266 NodeMCU giao tiếp I2C với màn hình LCD 16*2 thông qua module chuyển đổi I2C. ESP8266 sẽ đọc dữ liệu từ Firebase và hiển thị lên LCD

Phương án này có ưu điểm là sử dụng Bluetooth 5.0 JDY-24M, giúp tốc độ truyền dữ liệu nhanh hơn và khoảng cách kết nối xa hơn so với các module Bluetooth 2.0 như HC-05. ESP8266 NodeMCU có khả năng xử lý mạnh mẽ, hỗ trợ kết nối Wi-Fi

và tương thích tốt với Firebase, giúp lưu trữ và đồng bộ dữ liệu theo thời gian thực. Màn hình LCD 16×2 hiển thị thông tin rõ ràng, có thể hiển thị cả chữ và số, giúp người dùng dễ dàng theo dõi dữ liệu, với mạch chuyển đổi I2C cho LCD ta có thể giảm dây kết nối với vi xử lý làm cho thiết kế gọn gàng hơn. Ngoài ra, hệ thống có khả năng mở rộng, có thể tích hợp thêm cảm biến hoặc tính năng hiển thị khác. Tuy nhiên, nhược điểm của phương án này là ESP8266 tiêu thụ điện năng cao hơn ATmega328P, có thể cần quản lý nguồn tốt nếu sử dụng pin. LCD 16×2 cần module chuyển đổi I2C để tiết kiệm chân GPIO. Dù vậy, với những ưu điểm vượt trội, phương án này vẫn là lựa chọn tối ưu cho dự án.

3.3 Sơ Đồ Khối



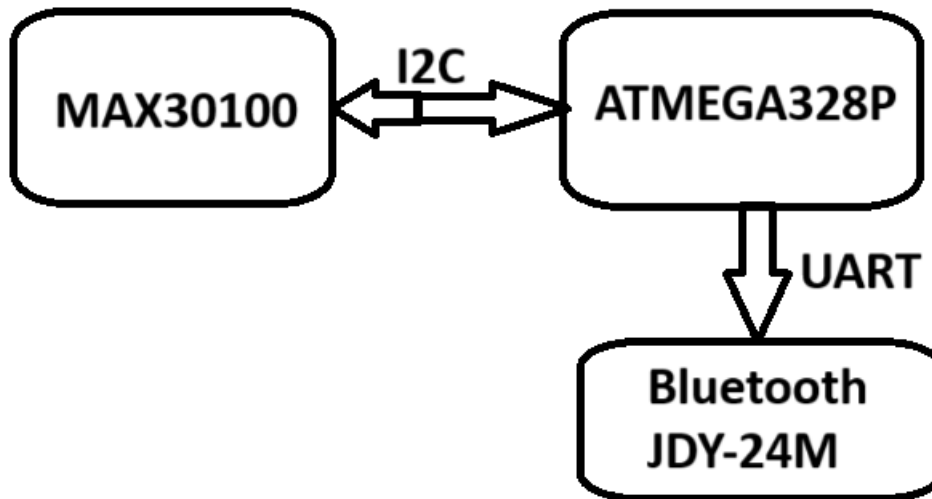
Hình 3-1 Sơ đồ khối tổng quát

3.3.1 Sơ đồ khối tổng quát

Khối 1 (thiết bị đeo của bệnh nhân) sẽ gửi dữ liệu đọc được từ cảm biến MAX30100 thông qua module Bluetooth Master tới module Slave của khối 2 (Hub

trung tâm). Sau đó Esp8266 NodeMCU của khối 2 sẽ gửi dữ liệu vừa nhận được lên Realtime Database của nền tảng Firebase. Khối 3 (thiết bị hiển thị) Esp8266 sẽ đọc dữ liệu ở Realtime Database được gửi lên từ khối 2, sau đó hiển thị lên LCD.

3.3.2 Sơ đồ khối chi tiết

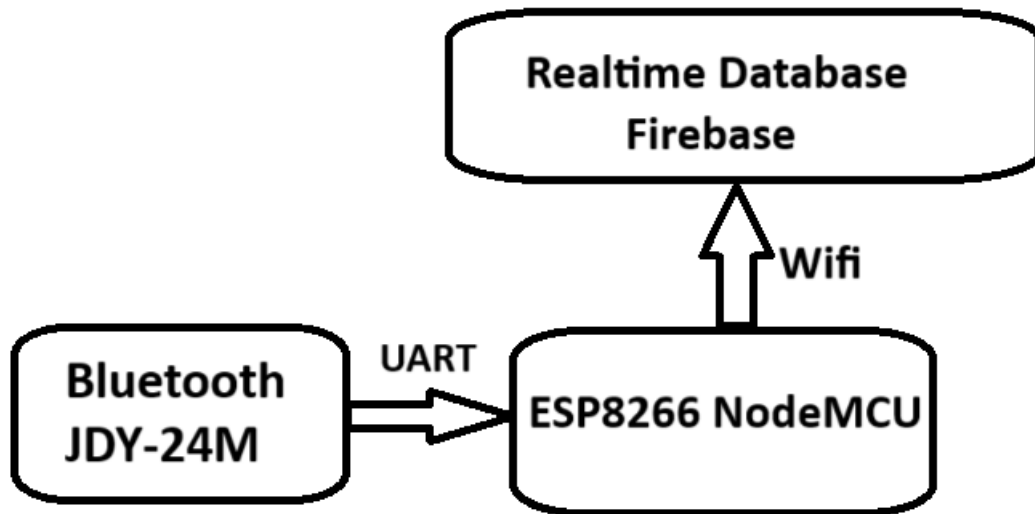


Hình 3-2 Sơ đồ khối của thiết bị đeo cho bệnh nhân

- **Khối 1: Thiết bị đeo cho bệnh nhân**

Khối 1 sử dụng cảm biến MAX30100 giao tiếp I2C với ATmega328P để đo nhịp tim. Sau khi xử lý dữ liệu, ATmega328P truyền thông tin đến module Bluetooth JDY-24M qua UART, module này được cấu hình ở chế độ Master để gửi dữ liệu đến khối 2.

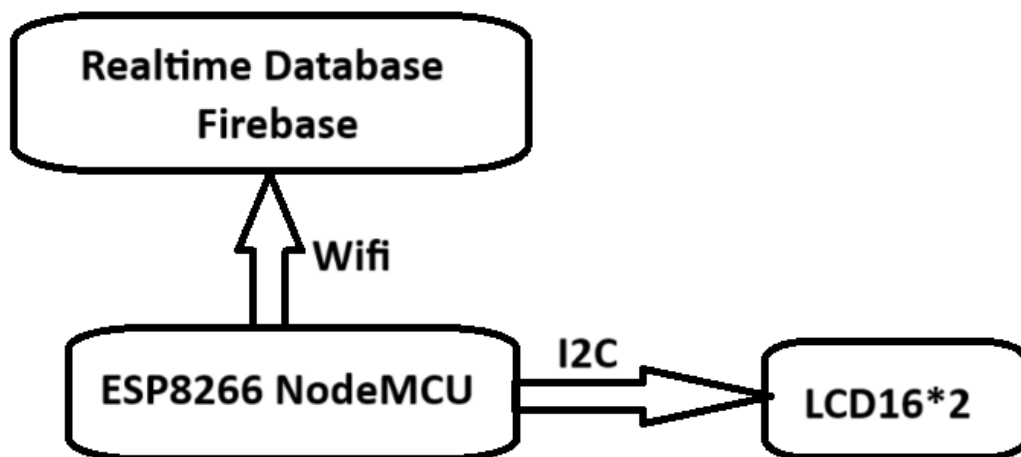
- **Khối 2: HUB Trung Tâm**



Hình 3-3 Khối HUB Trung Tâm

Khối 2 gồm ESP8266 NodeMCU giao tiếp với module Bluetooth JDY-24M ở chế độ Slave để nhận dữ liệu từ khối 1. ESP8266 sau đó kết nối Wi-Fi và gửi dữ liệu lên Firebase Realtime Database mỗi giây để lưu trữ và đồng bộ.

- **Khối 3: Thiết Bị Hiển Thị**

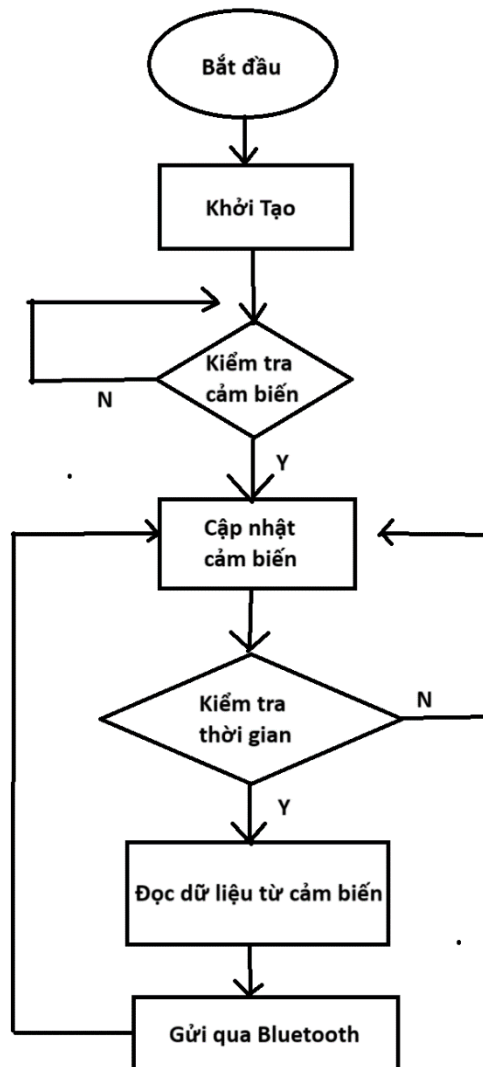


Hình 3-4 Khối thiết bị hiển thị

Khối 3 sử dụng ESP8266 NodeMCU giao tiếp I2C với màn hình LCD 16*2 thông qua module chuyển đổi I2C. ESP8266 đọc dữ liệu từ Firebase và hiển thị lên LCD, giúp người dùng theo dõi nhịp tim theo thời gian thực.

4 THIẾT KẾ VÀ THỰC HIỆN PHẦN MỀM

4.1 Thiết Bị Đo Của Bệnh Nhân



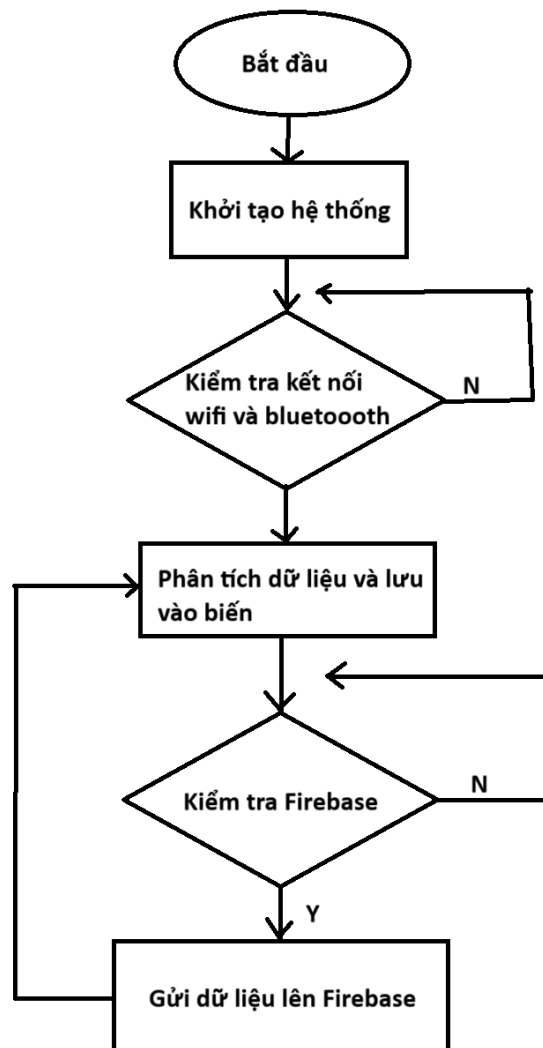
Hình 4-1 Lưu Đồ giải thuật khối 1

Khối 1 gồm cảm biến MAX30100, vi điều khiển ATmega328P và module Bluetooth JDY-24M. ATmega328P đọc dữ liệu từ MAX30100 qua giao tiếp I2C, xử lý dữ liệu, loại bỏ nhiễu và truyền thông tin đến module JDY-24M qua UART. JDY-24M được cấu hình ở chế độ Master để gửi dữ liệu đến khối 2 theo chu kỳ 1 giây. Sử dụng thư viện MAX30100_PulseOximeter để đọc dữ liệu. Truyền dữ liệu định dạng "temp: [nhịp_tim], spo2: [SpO2]" qua Bluetooth mỗi 1 giây.

Giải thích: Đầu tiên, hệ thống khởi động, cấu hình các giao thức giao tiếp như I2C với cảm biến, UART với Bluetooth, và kiểm tra nếu cảm biến hoạt động đúng cách. Trong vòng lặp chính, dữ liệu nhịp tim được cập nhật liên tục, nhưng chỉ gửi đi khi thời gian giữa hai lần truyền vượt quá 1 giây. Khi đủ điều kiện, chương trình đọc giá trị nhịp tim và nồng độ SpO2, chuyển đổi sang dạng chuỗi, gửi dữ liệu qua Bluetooth. Quá trình này lặp lại liên tục, đảm bảo hệ thống hoạt động ổn định và phản hồi theo thời gian thực.

4.2 HUB trung tâm

Khối 2 gồm ESP8266 NodeMCU giao tiếp với module Bluetooth JDY-24M ở chế độ Slave để nhận dữ liệu từ khối 1. ESP8266 xử lý dữ liệu nhận được và kết nối Wi-Fi để gửi thông tin lên Firebase Realtime Database mỗi giây. Dữ liệu trước khi gửi lên Firebase phải được định dạng phù hợp để đảm bảo lưu trữ chính xác.

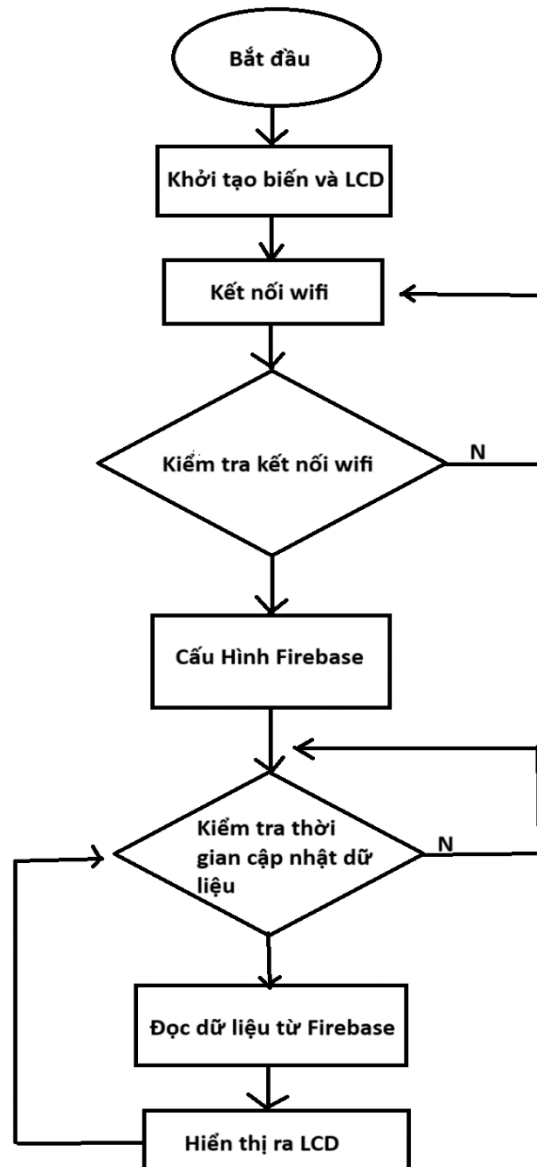


Hình 4-2 Lưu đồ giải thuật khối 2

Giải thích: Đầu tiên, hệ thống khởi động, thiết lập kết nối Wi-Fi và Firebase, sau đó bắt đầu quá trình chờ dữ liệu từ Bluetooth. Trong vòng lặp chính, chương trình kiểm tra xem kết nối Bluetooth có bị gián đoạn hay không. Nếu có dữ liệu mới, chương trình đọc chuỗi dữ liệu, tách thông tin về nhịp tim và nồng độ SpO2, rồi cập nhật trạng thái Bluetooth. Khi đủ điều kiện về thời gian và Firebase sẵn sàng, hệ thống sẽ gửi dữ liệu lên cơ sở dữ liệu, đồng thời hiển thị trạng thái gửi hoặc báo lỗi trên Serial Monitor. Quá trình này được lặp lại liên tục để đảm bảo hệ thống hoạt động ổn định và cập nhật dữ liệu theo thời gian thực.

4.3 Thiết Bị hiển thị lên LCD

Khối 3 gồm ESP8266 NodeMCU giao tiếp I2C với màn hình LCD 16×2 thông qua module chuyển đổi I2C. ESP8266 kết nối với Firebase để đọc dữ liệu nhịp tim đã lưu trữ, xử lý dữ liệu nếu cần, và hiển thị lên LCD. Việc hiển thị dữ liệu được cập nhật mỗi giây, đảm bảo đồng bộ với Firebase



Hình 4-3 Lưu đồ giải thuật khối 3

Ban đầu, hệ thống khởi tạo các thư viện cần thiết, thiết lập giao tiếp với Firebase, Wi-Fi và LCD. Nếu kết nối Wi-Fi thành công, chương trình tiến hành vòng lặp chính,

kiểm tra thời gian cập nhật dữ liệu. Mỗi giây, nó gửi yêu cầu đến Firebase để lấy dữ liệu nhịp tim và SpO2. Giá trị sẽ được hiển thị lên LCD. Sau khi cập nhật LCD, chương trình quay lại vòng lặp chính và tiếp tục lấy dữ liệu mới từ Firebase theo chu kỳ đã định. Quá trình này lặp lại liên tục, đảm bảo hệ thống hoạt động ổn định và hiển thị thông tin theo thời gian thực.

4.4 Kết nối module Bluetooth thông qua AT Command

4.4.1 Mục tiêu kết nối

Thiết lập kết nối giữa 2 module Bluetooth JDY-24M, trong đó một Module hoạt động ở chế độ Master, một module hoạt động ở chế độ Slave, nhằm thực hiện truyền và nhận dữ liệu thông qua Bluetooth ở khoảng cách gần

4.4.2 Quy trình thiết lập kết nối thông qua tập lệnh AT Command

Bước 1: Nạp chương trình cho Arduino Uno để vào chế độ AT Command cấu hình cho 2 Module Bluetooth JDY-24M.

```
#include <SoftwareSerial.h>
SoftwareSerial BTserial(10, 11);
void setup() {
    Serial.begin(9600);
    BTserial.begin(9600);
    Serial.println("Start AT Command mode test");
}
void loop() {
    if (Serial.available()) {
        char c = Serial.read();
        BTserial.write(c);
    }
    if (BTserial.available()) {
        char c = BTserial.read();
        Serial.write(c);
    }
}
```

```
}

```

Sau khi nạp chương trình xong, kết nối Module Bluetooth với Arduino Uno và bắt đầu cấu hình cho Module Bluetooth bằng tập lệnh AT Command bằng cửa sổ Serial Monitor trên IDE Arduino.

- **Module Slave:**

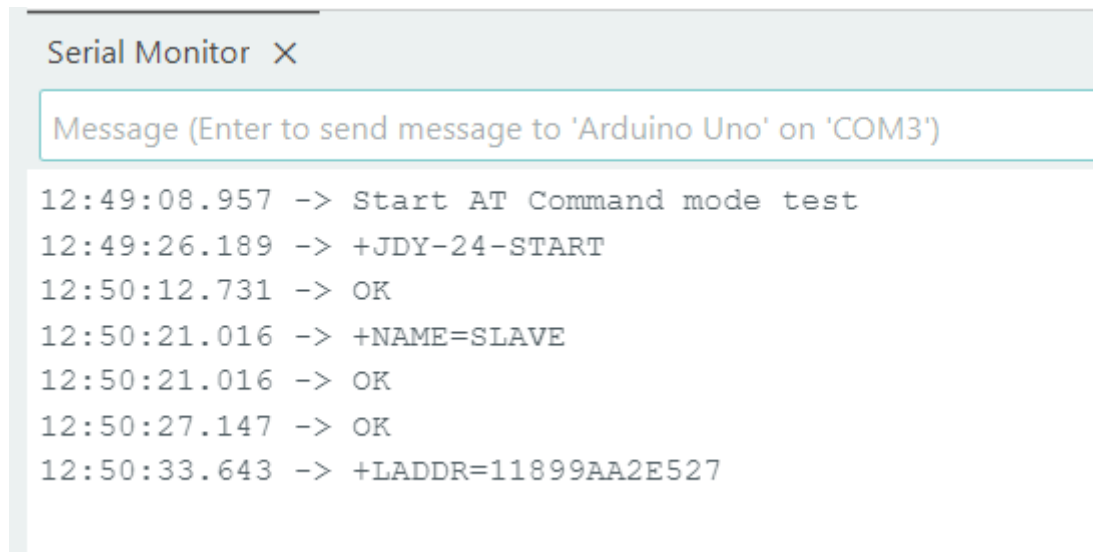
Baud rate: 9600

Sử dụng lệnh **AT** để kiểm tra hoạt động của Module. Phản hồi: **OK**, Module hoạt động bình thường.

Sử dụng lệnh **AT+NAMESLAVE** để đổi tên của Module thành **SLAVE**. Phản hồi: **OK**, đã đổi tên thành công.

Sử dụng lệnh **AT+ROLE0** để chỉnh vai trò của Module là Slave. Phản hồi: **OK**, đã thiết lập vai trò thành công.

Sử dụng lệnh **AT+LADDR** để xem địa chỉ MAC của Module Slave. Phản hồi: **LADDR=11899AA2E527**, địa chỉ MAC của Module Slave là:



```

Serial Monitor X
Message (Enter to send message to 'Arduino Uno' on 'COM3')
12:49:08.957 -> Start AT Command mode test
12:49:26.189 -> +JDY-24-START
12:50:12.731 -> OK
12:50:21.016 -> +NAME=SLAVE
12:50:21.016 -> OK
12:50:27.147 -> OK
12:50:33.643 -> +LADDR=11899AA2E527
  
```

Hình 4-4 Cấu hình Module Slave trên cửa sổ Serial Monitor

11899A2E527.

- **Module Master**

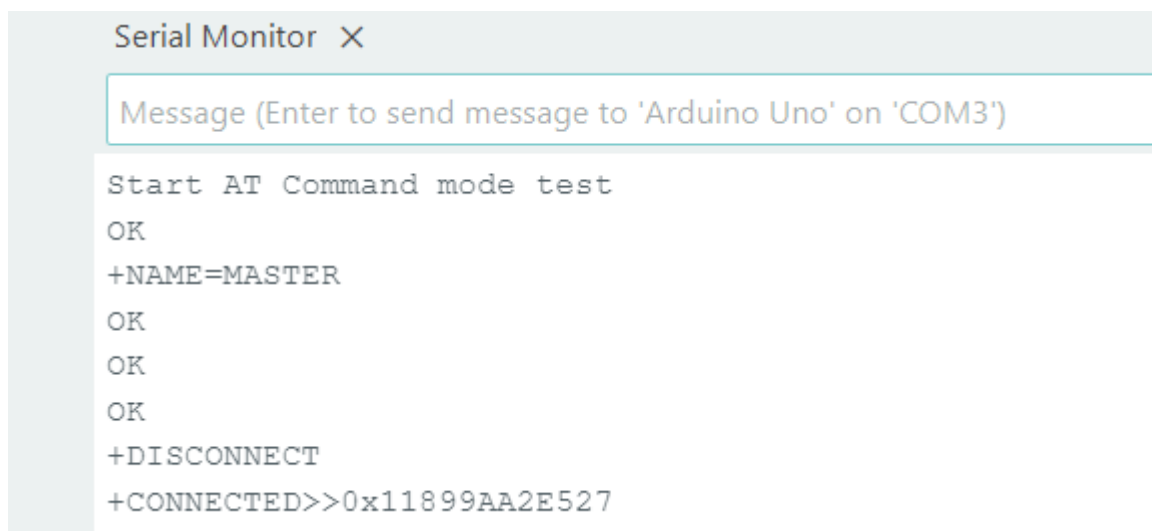
Baud rate: 9600

Sử dụng lệnh **AT** để kiểm tra hoạt động của Module. Phản hồi: **OK**, Module hoạt động bình thường.

Sử dụng lệnh **AT+NAMEMASTER** để đổi tên của Module thành **MASTER**. Phản hồi: **OK**, đã đổi tên thành công.

Sử dụng lệnh **AT+ROLE1** để chỉnh vai trò của Module là Master. Phản hồi: **OK**, đã thiết lập vai trò thành công.

Sử dụng lệnh **AT+BAND11899AA2E527** để kết nối Module Master với Module Slave. Phản hồi: **CONENECTED>>0x11899AA2E527**, đã kết nối thành công 2 Module với

The image shows a screenshot of a 'Serial Monitor' window. At the top, there is a title bar that says 'Serial Monitor' followed by a close button icon. Below the title bar is a text input field with the placeholder text 'Message (Enter to send message to 'Arduino Uno' on 'COM3')'. The main area of the window displays the output of AT commands. The text shown is: 'Start AT Command mode test', 'OK', '+NAME=MASTER', 'OK', 'OK', 'OK', '+DISCONNECT', and '+CONENECTED>>0x11899AA2E527'.

```
Serial Monitor X
Message (Enter to send message to 'Arduino Uno' on 'COM3')
Start AT Command mode test
OK
+NAME=MASTER
OK
OK
OK
+DISCONNECT
+CONENECTED>>0x11899AA2E527
```

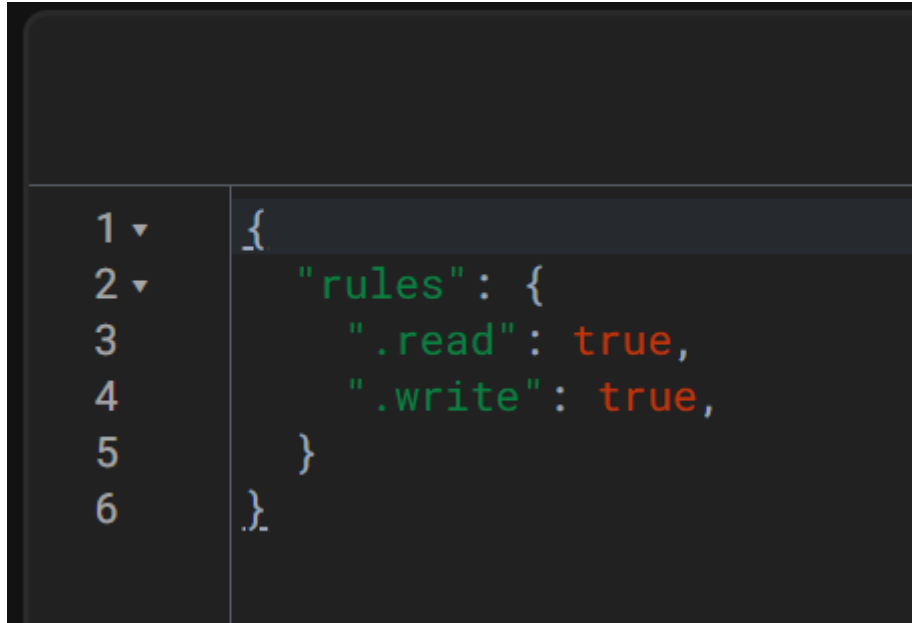
Hình 4-5 Cấu hình Module Master trên cửa sổ Serial Monitor

nhau.

Khi hai Module kết nối thành công với nhau, chúng sẽ luôn sáng đèn chứ không chớp đèn nhấp nháy như lúc chưa kết nối. Lúc này, ta có thể truyền dữ liệu từ Module Master tới Module Slave thông qua Bluetooth.

4.5 Cấu hình Firebase

- Rules Database cho phép đọc/ghi tự do để phát triển.

A screenshot of a code editor with a dark background. On the left, there is a line number column with numbers 1 through 6. To the right of the numbers is a JSON object representing Firebase database rules. The code is: {, "rules": {, ".read": true,, ".write": true,, }, }.

```
1 {
2   "rules": {
3     ".read": true,
4     ".write": true,
5   }
6 }
```

Hình 4-6 Cấu hình Firebase

5 KẾT QUẢ THỰC HIỆN

5.1 Thiết bị đo

- Thiết Bị sử dụng: Nguồn adapter 5V-1A để cấp nguồn cho hệ thống, 1 chip ATmega328P làm vi điều khiển chính, 1 cảm biến MAX30100 để đo chỉ số nhịp tim và SpO2 của bệnh nhân, 1 Module Bluetooth 5.0 JDY-24M để gửi dữ liệu qua HUB trung tâm.
- Kết nối thiết bị:
 - Max30100:

Vin	Vcc
Gnd	Gnd
SDA	Pin 27 ATmega328P
SCL	Pin 28 ATmega328P

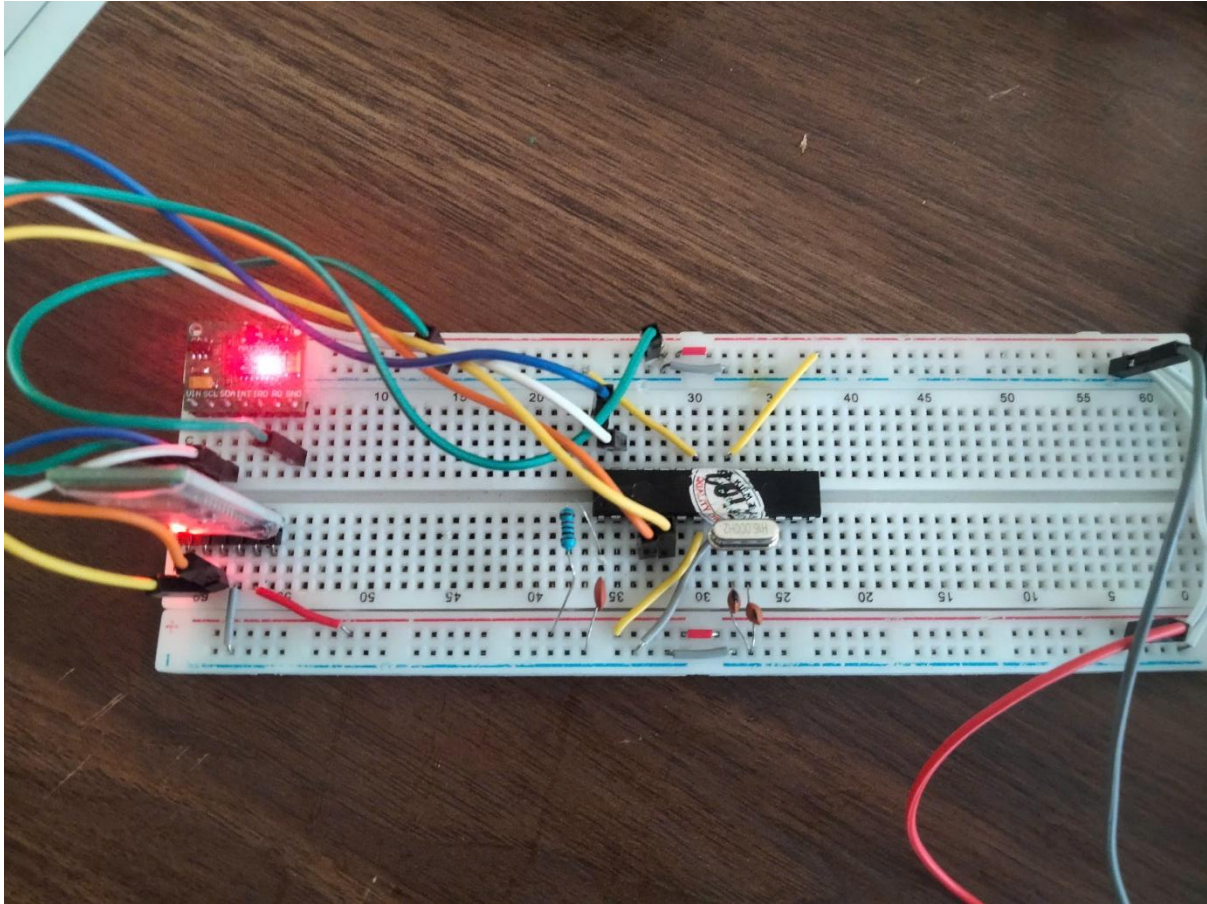
Bảng 2: sơ đồ kết nối Max30100 ở thiết bị đo

- Module Bluetooth 5.0 JDY-24M

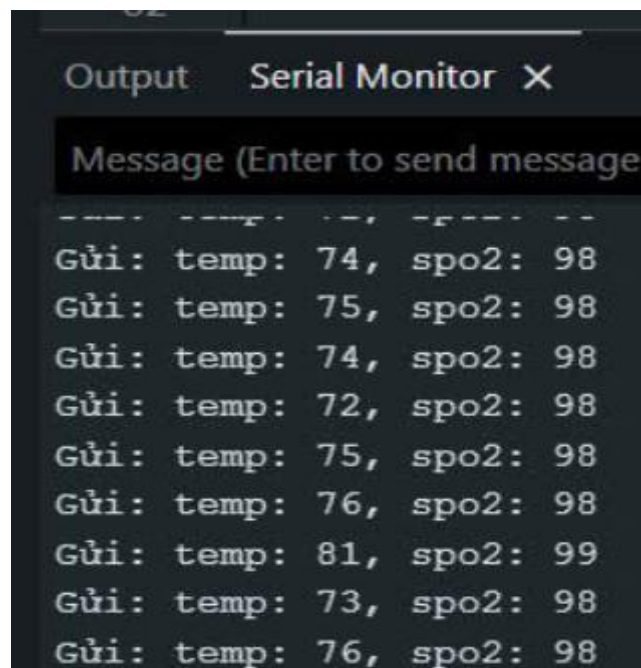
Vin	Vcc
Gnd	Gnd
Tx	Pin 4 ATmega328P
Rx	Pin 5 ATmega328P

Bảng 3: Sơ đồ kết nối Module Bluetooth 5.0 JDY-24M ở thiết bị đeo

- Kết quả:



Hình 5-1 Phần cứng thiết bị đeo cho bệnh nhân



Hình 5-2 kết quả gửi chỉ số sinh tồn cho HUB trung tâm

5.2 HUB và Firebase

- Thiết bị sử dụng:

Nguồn Adapter 5V-1A để cấp nguồn cho HUB trung tâm, 1 ESP8266 NodeMCU làm vi điều khiển chính cho HUB, 1 Module Bluetooth 5.0 JDY-24M để nhận dữ liệu từ thiết bị đeo cho bệnh nhân.

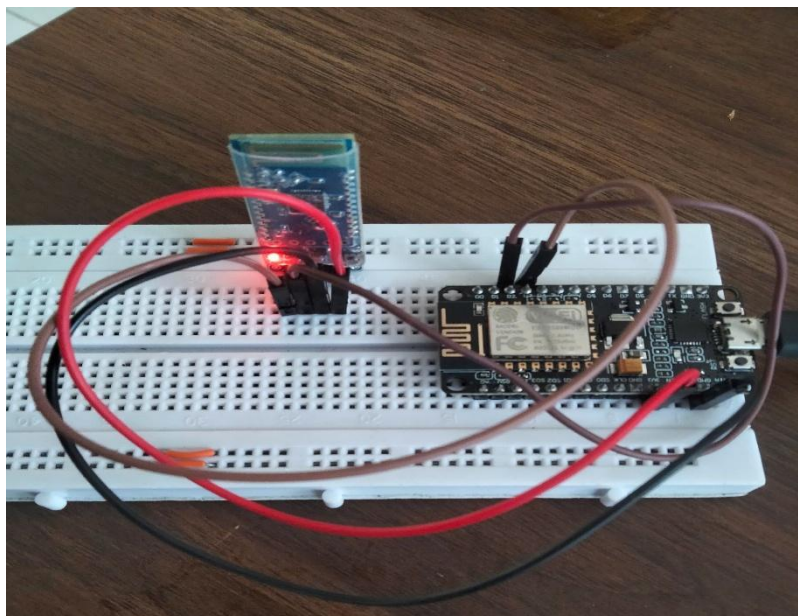
- Kết nối thiết bị:

- Module Bluetooth 5.0 JDY-24M

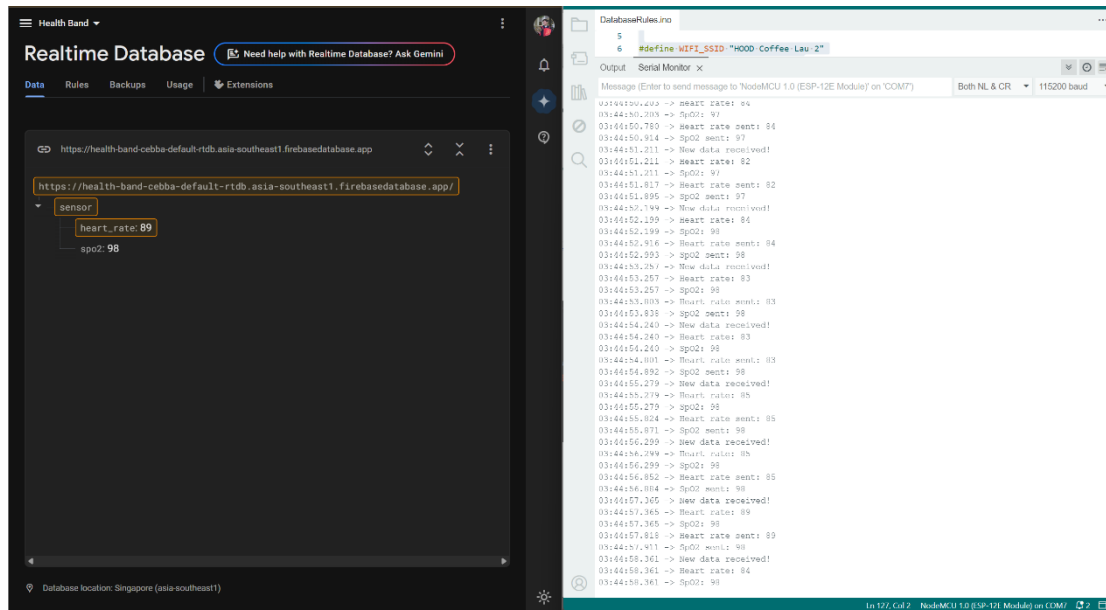
Bảng 5.1 Sơ đồ kết nối Module Bluetooth 5.0 JDY-24M ở HUB trung tâm

Vin	Vcc
Gnd	Gnd
Tx	D2 ESP8266 NodeMCU
Rx	D3 ESP8266 NodeMCU

- Kết quả



Hình 5-3 Phân cứng thiết bị HUB trung tâm



Hình 5-4 kết quả gửi chỉ số sinh tồn của bệnh nhân lên Firebase

5.3 Hiện thị LCD

- Thiết bị sử dụng:
Một ESP8266 NodeMCU là vi điều khiển chính, một mạch chuyển đổi I2C kết nối với một LCD 16*2 để hiện thị chỉ số của bệnh nhân.
- Kết nối
 - LCD16*2 kết nối mạch chuyển đổi giao tiếp I2C

Bảng 5.2 Sơ đồ kết nối LCD 16*2 và mạch chuyển đổi giao tiếp I2C

Vin	3V3 ESP8266 NodeMCU
Gnd	GND ESP8266 NodeMCU
SCL	D1 ESP8266 NodeMCU
SDA	D2 ESP8266 NodeMCU

- Kết Quả



Hình 5-5 Phần cứng của thiết bị hiển thị và kết quả hiển thị lên LCD

6 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1 Kết luận

Sau quá trình nghiên cứu và thực hiện đề tài “Hệ thống IoT theo dõi chỉ số sinh tồn của bệnh nhân thông qua Bluetooth Low Energy và Wifi”, nhóm chúng em đã xây dựng thành công hệ thống theo dõi chỉ số sinh tồn của bệnh nhân đảm bảo quá trình thu thập, xử lý, truyền tải và hiển thị dữ liệu theo thời gian thực. Dữ liệu được đo, xử lý và truyền đi một cách ổn định, với khả năng đồng bộ liên tục giúp theo dõi sức khỏe hiệu quả. Hệ thống hoạt động với độ chính xác cao, tốc độ truyền dữ liệu nhanh và độ trễ thấp, đảm bảo khả năng giám sát liên tục. Kết quả thử nghiệm cho thấy hệ thống có tính linh hoạt và phù hợp với các ứng dụng giám sát sức khỏe từ xa.

6.2 Hướng phát triển

- Bổ sung cảm biến nhiệt độ, huyết áp.
- Tối ưu năng lượng cho thiết bị đeo (sử dụng pin Li-Po).
- Triển khai ứng dụng di động để cảnh báo và theo dõi.
- Phát triển HUB trung tâm có thể sử dụng cho nhiều thiết bị đeo cùng lúc

7 TÀI LIỆU THAM KHẢO

- [1] Arduino Kit, “ESP8266 là gì? Hướng dẫn lập trình ESP8266 bằng Arduino IDE,” *ArduinoKit.vn*. <https://arduinookit.vn/esp8266-la-gi-huong-dan-lap-trinh-esp8266-bang-arduino-ide/>.
- [2] Nhà xuất bản Xây dựng, Internet vạn vật, Hà Nội, Việt Nam, 2025
- [3] Atmel Corporation, "ATmega32 datasheet," 2012. [Online]. Available: <https://www.microchip.com/wwwproducts/en/ATmega32>
- [4] RF-star, “Giao thức truyền thông nối tiếp chung: UART, SPI, I2C,” *RFstariot.com*. https://vi.rfstariot.com/blog/generic-serial-communication-protocols-uart-spi-i2c_b64
- [5] Mecsuvn, “Tìm hiểu MAX30100 - Cảm biến nhịp tim,” *Mecsuvn*. <https://mecsuvn/ho-tro-ky-thuat/tim-hieu-max30100-cam-bien-nhip-tim.Qvn>.

8 PHỤ LỤC

PHỤ LỤC I.

MÃ NGUỒN VÀ CHÚ THÍCH CỦA THIẾT BỊ ĐEO CHO BỆNH NHÂN

```
#include <Wire.h>
#include "MAX30100_PulseOximeter.h"
#include <SoftwareSerial.h>
// Khởi tạo phần mềm Serial cho giao tiếp với JDY-24M (Master)
SoftwareSerial BTSerial(2, 3);
PulseOximeter pox; // Khởi tạo đối tượng đo nhịp tim và SpO2
// Biến lưu thời gian gửi dữ liệu
unsigned long lastSendTime = 0;
// Thời gian giữa hai lần gửi dữ liệu, đặt là 1 giây
unsigned long sendInterval = 1000;
// Hàm callback được gọi khi phát hiện nhịp tim mới
void onBeatDetected() {
}
void setup() {
    // Khởi động Serial Monitor để kiểm tra dữ liệu
    Serial.begin(115200);
    BTSerial.begin(9600); // Thiết lập baud rate cho module Bluetooth JDY-24M
    // Khởi tạo cảm biến MAX30100
    if (!pox.begin()) {
        for (;;)
    }
    // Cấu hình dòng LED IR để tăng độ ổn định
    pox.setIRLedCurrent(MAX30100_LED_CURR_7_6MA);
    // Đăng ký hàm callback để xử lý sự kiện phát hiện nhịp tim
    pox.setOnBeatDetectedCallback(onBeatDetected);
    Serial.println("Module Master đang hoạt động...");
}
void loop() {
    // Cập nhật dữ liệu từ cảm biến MAX30100
    pox.update();
    // Kiểm tra nếu đủ thời gian để gửi dữ liệu (1 giây)
    if (millis() - lastSendTime >= sendInterval) {
        // Đọc giá trị nhịp tim và SpO2 từ cảm biến
        float heartRate = pox.getHeartRate();
        float spO2 = pox.getSpO2();
        // Chuyển đổi giá trị nhịp tim và SpO2 sang số nguyên
        int heartRateInt = (int)heartRate;
        int spO2Int = (int)spO2;
        // Định dạng dữ liệu trước khi gửi
        String dataToSend = "temp: " + String(heartRateInt) + ", spo2: " + String(spO2Int);
        // Gửi dữ liệu qua Bluetooth đến JDY-24M
        BTSerial.println(dataToSend);
    }
}
```

```
// Hiển thị dữ liệu lên Serial Monitor để kiểm tra
Serial.print("Gửi: ");
Serial.println(dataToSend);
// Cập nhật lại thời điểm gửi dữ liệu
lastSendTime = millis();
}
}
```


PHỤ LỤC II.

MÃ NGUỒN VÀ CHÚ THÍCH CỦA HUB TRUNG TÂM

```
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>
#include <SoftwareSerial.h>
// Cấu hình thông tin Wi-Fi
#define WIFI_SSID "OPPO A53"
#define WIFI_PASSWORD "00000000"
// Cấu hình thông tin Firebase
#define API_KEY "AIzaSyDbn7kJuDQq7QK-8hXG9C66s2vQZJNq_3Y"
#define DATABASE_URL "health-band-cebba-default-rtdb.asia-southeast1.firebaseio.com"
#define FIREBASE_AUTH "YMvSSQgZLUKJeXap6CUac1Rf9ssFX3CfyQoYWeNL"
// Định nghĩa chân RX và TX cho giao tiếp Bluetooth
#define BT_RX D2
#define BT_TX D3
// Khởi tạo giao tiếp Bluetooth
SoftwareSerial bluetooth(BT_RX, BT_TX);
// Khởi tạo Firebase
FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;
// Biến lưu thời gian gửi dữ liệu
unsigned long sendDataPrevMillis = 0;
unsigned long lastDataTime = 0;
bool bluetoothConnected = false;
const unsigned long BLUETOOTH_TIMEOUT = 30000; // Thời gian timeout cho Bluetooth
// Biến lưu dữ liệu nhịp tim và SpO2
int heartRate = 0;
int spo2 = 0;
bool newData = false;
void setup() {
    // Khởi động Serial Monitor
    Serial.begin(115200);
    // Khởi động giao tiếp Bluetooth
    bluetooth.begin(9600);
    // Kết nối Wi-Fi
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
    Serial.print("Connecting to WiFi");
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("\nConnected!");
    // Cấu hình Firebase
    config.api_key = API_KEY;
```



```
config.database_url = DATABASE_URL;
config.signer.tokens.legacy_token = FIREBASE_AUTH;
Firebase.begin(&config, &auth);

Serial.println("Waiting for Bluetooth connection...");
}
void loop() {
    // Đọc dữ liệu từ Bluetooth
    readBluetoothData();
    // Kiểm tra trạng thái kết nối Bluetooth
    checkBluetoothConnection();
    // Gửi dữ liệu lên Firebase khi có dữ liệu mới và thời gian gửi hợp lệ
    if (Firebase.ready() && (millis() - sendDataPrevMillis > 1000) && newData) {
        sendDataPrevMillis = millis();
        // Gửi dữ liệu nhịp tim lên Firebase
        if (Firebase.setInt(fbdo, "/sensor/heart_rate", heartRate)) {
            Serial.print("Heart rate sent: ");
            Serial.println(heartRate);
        } else {
            Serial.println("Heart rate error: " + fbdo.errorReason());
        }
    }
    // Gửi dữ liệu SpO2 lên Firebase
    if (Firebase.setInt(fbdo, "/sensor/spo2", spo2)) {
        Serial.print("SpO2 sent: ");
        Serial.println(spo2);
    } else {
        Serial.println("SpO2 error: " + fbdo.errorReason());
    }
    newData = false; // Đặt lại trạng thái dữ liệu
}
// Kiểm tra trạng thái kết nối Bluetooth
void checkBluetoothConnection() {
    if (bluetoothConnected && (millis() - lastDataTime > BLUETOOTH_TIMEOUT)) {
        bluetoothConnected = false;
        Serial.println("Bluetooth disconnected!");
    }
}
// Đọc dữ liệu từ Bluetooth
void readBluetoothData() {
    static String buffer = "";
    while (bluetooth.available()) {
        char c = bluetooth.read();
        if (c == '\n') {
            parseSensorData(buffer);
            buffer = "";
        } else {
            buffer += c;
        }
    }
}
```

```
    }  
  }  
}  
// Xử lý chuỗi dữ liệu nhận được từ Bluetooth  
void parseSensorData(String data) {  
  int hrIndex = data.indexOf("temp: ");  
  int spo2Index = data.indexOf("spo2: ");  
  if (hrIndex != -1 && spo2Index != -1) {  
    lastDataTime = millis(); // Cập nhật thời gian nhận dữ liệu  
    // Trích xuất giá trị nhịp tim từ chuỗi dữ liệu  
    String hrStr = data.substring(hrIndex + 6, data.indexOf(',', hrIndex));  
    heartRate = hrStr.toInt();  
    // Trích xuất giá trị SpO2 từ chuỗi dữ liệu  
    String spo2Str = data.substring(spo2Index + 6);  
    spo2 = spo2Str.toInt();  
    if (!bluetoothConnected) {  
      bluetoothConnected = true;  
      Serial.println("Bluetooth connected!");  
    }  
    newData = true; // Đánh dấu dữ liệu mới đã nhận  
    Serial.println("New data received!");  
    Serial.println("Heart rate: " + String(heartRate));  
    Serial.println("SpO2: " + String(spo2));  
  }  
}
```

PHỤ LỤC III. MÃ NGUỒN VÀ CHÚ THÍCH CỦA THIẾT BỊ HIỂN THỊ

```
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
// Cấu hình thông tin Wi-Fi
#define WIFI_SSID "OPPO A53"
#define WIFI_PASSWORD "00000000"
// Cấu hình thông tin Firebase
#define API_KEY "AIzaSyDbn7kJuDQq7QK-8hXG9C66s2vQZJNq_3Y"
#define DATABASE_URL "health-band-cebba-default-rtdb.asia-southeast1.firebaseio.com"
#define FIREBASE_AUTH "YMvSSQgZLukJeXap6CUac1Rf9ssFX3CfyQoYWeNL"
// Khởi tạo màn hình LCD 16x2 sử dụng giao tiếp I2C
LiquidCrystal_I2C lcd(0x27, 16, 2);
// Khởi tạo Firebase
FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;
// Biến lưu thời gian cập nhật dữ liệu từ Firebase
unsigned long lastUpdateTime = 0;
const unsigned long UPDATE_INTERVAL = 1000; // Chu kỳ cập nhật dữ liệu (1 giây)
void setup() {
    // Khởi động Serial Monitor để kiểm tra quá trình kết nối
    Serial.begin(115200);
    // Khởi tạo và cấu hình màn hình LCD
    lcd.init();
    lcd.backlight();
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Initializing...");
    // Kết nối Wi-Fi
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
    Serial.print("Connecting to WiFi");
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("\nConnected!");
    // Cấu hình Firebase
    config.api_key = API_KEY;
    config.database_url = DATABASE_URL;
    config.signer.tokens.legacy_token = FIREBASE_AUTH;
    Firebase.begin(&config, &auth);
    // Xác nhận Firebase sẵn sàng trên màn hình LCD
    lcd.clear();
    lcd.print("Firebase Ready!");
```

```
    delay(1000);
    lcd.clear();
}

void loop() {
    // Kiểm tra nếu đã đủ thời gian cập nhật dữ liệu từ Firebase
    if (millis() - lastUpdateTime > UPDATE_INTERVAL) {
        lastUpdateTime = millis();
        // Đọc dữ liệu nhịp tim và SpO2 từ Firebase
        int heartRate = getFirebaseData("/sensor/heart_rate");
        int spo2 = getFirebaseData("/sensor/spo2");
        // Cập nhật dữ liệu hiển thị lên màn hình LCD
        updateLCD(heartRate, spo2);
    }
}

// Hàm lấy dữ liệu từ Firebase
int getFirebaseData(const String &path) {
    if (Firebase.getInt(fbdo, path)) {
        return fbdo.intData(); // Trả về giá trị dữ liệu nếu lấy thành công
    } else {
        Serial.println("Error reading " + path + ": " + fbdo.errorReason()); // Báo lỗi nếu đọc thất bại
        return -999; // Trả về giá trị lỗi nếu không nhận được dữ liệu
    }
}

// Hàm cập nhật dữ liệu lên màn hình LCD
void updateLCD(int hr, int spo2) {
    // Hiển thị nhịp tim
    lcd.setCursor(0, 0);
    lcd.print("Nhịp Tim:   ");
    lcd.setCursor(9, 0);
    if (hr != -999) {
        lcd.print(hr);
        lcd.print(" bpm");
    } else {
        lcd.print("--- "); // Nếu dữ liệu bị lỗi, hiển thị "---"
    }
    // Hiển thị SpO2
    lcd.setCursor(0, 1);
    lcd.print("SpO2:   ");
    lcd.setCursor(6, 1);
    if (spo2 != -999) {
        lcd.print(spo2);
        lcd.print("%");
    } else {
        lcd.print("--- "); // Nếu dữ liệu bị lỗi, hiển thị "---"
    }
}
}
```

PHỤ LỤC IV.

BẢNG GIÁ LINH KIỆN DÙNG TRONG ĐỀ TÀI

STT	SL	TÊN LINH KIỆN	GIÁ	NHÀ CUNG CẤP
1	2	Module Bluetooth BLE MESH JDY-24M	65.000	Hshop
2	1	Cảm biến nhịp tim và oxy trong máu MAX30100	32.000	Hshop
3	1	Màn hình LCD 16x2	24.000	Thegioiic
4	2	ESP8266 NodeMCU Lua CP2102	73.000	Thegioiic
5	1	ATMega328P	75.000	Hshop
6	1	Thạch anh 16MHz	1.700	Thegioiic
7	2	Tụ gốm tròn 22pF	100	Thegioiic
8	1	Tụ gốm tròn 100nF	100	Thegioiic
9	2	Breadboard	16.000	Thegioiic
10	1	Điện trở 10k Ohm	56	Thegioiic
11	2	Adapter 5V	22.000	Thegioiic
12	1	Dây nối các loại	20.000	Thegioiic
		TỔNG	504.956	