

HỌC VIỆN KỸ THUẬT MẬT MÃ

KHOA: CÔNG NGHỆ THÔNG TIN

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**🙣🙣🕮🙡🙡**



**THỰC TẬP CƠ SỞ**

**ĐỀ TÀI**

**NGHIÊN CỨU, XÂY DỰNG**

**HỆ THỐNG HỖ TRỢ CHĂM SÓC BỆNH NHÂN**

**SỬ DỤNG ARDUINO MEGA 2560**

**Giảng viên HD: Ths.Lê Thị Hồng Vân**

**Sinh viên thực hiện: Nhóm 15**

**Nguyễn Thị An CT040101**

**Đỗ Thị Ngọc Anh CT040104**

**Hoàng Nghĩa Thái CT040142**

Hà Nội, ngày 8 tháng 2 năm 2023

**MỤC LỤC**

[MỤC LỤC 2](#_heading=h.30j0zll)

[DANH MỤC HÌNH ẢNH 4](#_heading=h.1fob9te)

[DANH MỤC BẢNG 5](#_heading=h.3znysh7)

[LỜI MỞ ĐẦU 6](#_heading=h.2et92p0)

[CHƯƠNG 1: CƠ SỞ LÝ THUYẾT 8](#_heading=h.tyjcwt)

[1.1. Phát biểu bài toán 8](#_heading=h.p6jot59m8h3m)

[1.2. Tổng quan về Arduino Mega 2560 8](#_heading=h.3dy6vkm)

[1.2.1. Giới thiệu về Arduino Mega 2560 8](#_heading=h.1t3h5sf)

[1.2.2. Ưu điểm và ứng dụng của Arduino Mega 2560 15](#_heading=h.26in1rg)

[1.3. Tổng quan về ESP32 16](#_heading=h.tkxoq7ar15cg)

[1.3.1. Giới thiệu về ESP32 16](#_heading=h.mcp0yeb13avz)

[1.3.2. Ưu điểm và ứng dụng của ESP32 22](#_heading=h.bo2yxf9hd5a5)

[1.4. Tổng quan về giao thức I2C 22](#_heading=h.1ksv4uv)

[1.4.1. Giới thiệu về giao thức I2C 22](#_heading=h.gjdgxs)

[1.4.2. Đặc điểm của giao thức I2C 23](#_heading=h.44sinio)

[1.4.3. Cách thức hoạt động của I2C 23](#_heading=h.2jxsxqh)

[1.5. Tổng quan về giao thức UART 26](#_heading=h.2xcytpi)

[1.5.1. Giới thiệu về giao thức UART 26](#_heading=h.1ci93xb)

[1.5.2. Đặc điểm của UART 27](#_heading=h.3whwml4)

[1.5.3. Cách thức hoạt động của giao thức UART 27](#_heading=h.qsh70q)

[1.6. Giới thiệu về Arduino IDE 28](#_heading=h.3ceuz6yzwe3v)

[1.6.1. Tổng quan về lập trình Arduino 28](#_heading=h.1pxezwc)

[1.6.2. Tổng quan về Arduino IDE 30](#_heading=h.maswbuoudg43)

[1.7. Giới thiệu về Firebase 31](#_heading=h.sqfpf8dkle45)

[1.7.1 Tổng quan về Firebase 31](#_heading=h.wyjycrm607w9)

[1.7.2 Lịch sử phát triển của Firebase 31](#_heading=h.iaigvpepozdm)

[1.7.3. Những service nổi bật 32](#_heading=h.hq5wnvxh2p5k)

[1.8. Giới thiệu về Android Studio 34](#_heading=h.nf4u3myd8d5o)

[1.8.1. Tổng quan về hệ điều hành Android 34](#_heading=h.s5mkvhamkyxz)

[1.8.2. Tổng quan về Android Studio 34](#_heading=h.njlcfwxi862n)

[1.8.3. Các tính năng của Android Studio 35](#_heading=h.57w21v24eny7)

[CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 37](#_heading=h.23ckvvd)

[2.1. Sơ đồ khối chức năng 37](#_heading=h.5o4iqssqq7xi)

[2.2. Thiết kế phần cứng 38](#_heading=h.qdlpiq4omowm)

[2.2.1. Lựa chọn linh kiện thiết kế mô hình 38](#_heading=h.1r8jnglydd5e)

[2.2.2. Sơ đồ đấu nối 41](#_heading=h.st64tk3f917j)

[2.2.3. Nguyên lý hoạt động 41](#_heading=h.u1iylexpui6p)

[2.2.4. Thi công phần cứng 42](#_heading=h.n2kndowhhtaz)

[2.3. Thiết kế phần mềm 43](#_heading=h.3fwokq0)

[2.3.1. Lưu đồ thuật toán 43](#_heading=h.1v1yuxt)

[2.3.2. Các biểu đồ usecase và tuần tự 44](#_heading=h.2u6wntf)

[CHƯƠNG 3: THỰC NGHIỆM VÀ ĐÁNH GIÁ HỆ THỐNG 45](#_heading=h.28h4qwu)

[3.1. Mô hình thiết bị y tế chăm sóc bệnh nhân 45](#_heading=h.pgohfbu6ut0s)

[3.1.1. Các chức năng của thiết bị 45](#_heading=h.9nvz02vb49g7)

[3.1.2. Kết quả thực nghiệm đo các chỉ số của bệnh nhân 46](#_heading=h.mu1700v2kc6y)

[3.2. Giao diện ứng dụng trên điện thoại 48](#_heading=h.2lwamvv)

[3.3. Đánh giá hệ thống 48](#_heading=h.kzbjpxt4wou5)

[KẾT LUẬN 49](#_heading=h.ltpzryoxxqfh)

[TÀI LIỆU THAM KHẢO 50](#_heading=h.3l18frh)

[PHỤ LỤC (Phần code nhúng và lập trình phần mềm) 51](#_heading=h.206ipza)

# DANH MỤC HÌNH ẢNH

[Hình 1-1: Arduino Mega 2560 9](#_heading=h.4d34og8)

[Hình 1-2: Các thành phần chức năng của board Arduino Mega 10](#_heading=h.17dp8vu)

[Hình 1-3: Sơ đồ chân của Arduino Mega 2560 12](#_heading=h.3rdcrjn)

[Hình 1-4: Arduino mở rộng cánh cửa với IoT](about:blank) 15

[Hình 1-5: Sơ đồ chân ESP32 DEVKIT V1 – DOIT phiên bản 30 GPIO 18](#_heading=h.35nkun2)

[Hình 1-6: Đường truyền tín hiệu của I2C 23](#_heading=h.z337ya)

[Hình 1-7: Khung truyền I2C 24](#_heading=h.3j2qqm3)

[Hình 1-8: Khối bit địa chỉ 24](#_heading=h.1y810tw)

[Hình 1-9: Quá trình truyền nhận dữ liệu 25](#_heading=h.4i7ojhp)

[Hình 1-10: Kết nối UART 27](#_heading=h.3as4poj)

[Hình 1-11: Giao diện chính của phần mềm Arduino IDE 31](#_heading=h.49x2ik5)

[Hình 1-12: Firebase là nền tảng phát triển ứng dụng đa năng của di động và website 32](#_heading=h.2p2csry)

[Hình 1-13: Tạo một dự án với Firebase 33](#_heading=h.147n2zr)

[Hình 1-14: Giao diện màn hình đầu của Android Studio 35](#_heading=h.3o7alnk)

[Hình 2-1: Sơ đồ khối chức năng 36](#_heading=h.ihv636)

[Hình 2-2: Module cảm biến nhịp tim MAX30100 38](#_heading=h.32hioqz)

[Hình 2-3: Màn hình LCD 1602 39](#_heading=h.1hmsyys)

[Hình 2-4: Các chân của LCD](about:blank) 39

[Hình 2-5: Sơ đồ đấu nối 40](#_heading=h.41mghml)

[Hình 2-6: Sơ đồ phần cứng 41](#_heading=h.oiq3zpnachtw)

[Hình 2-7: Lưu đồ thuật toán 42](#_heading=h.4f1mdlm)

[Hình 2-8: Sơ đồ usecase đo các chỉ số 43](#_heading=h.19c6y18)

[Hình 2-9: Biểu đồ tuần tự đo các chỉ số 43](#_heading=h.3tbugp1)

[Hình 3-1: Khi chưa đo 44](#_heading=h.nmf14n)

[Hình 3-2: Khi đo sẽ hiển thị kết quả 45](#_heading=h.37m2jsg)

[Hình 3-3: Giao diện ứng dụng khi đo 47](#_heading=h.111kx3o)

# DANH MỤC BẢNG

[Bảng 1-1: Thông số của Arduino Mega 2560 9](#_heading=h.2s8eyo1)

[Bảng 1-2: Thông số kỹ thuật của ESP32 DEVKIT V1-DOIT 17](#_heading=h.lnxbz9)

[Bảng 1-3: Đặc điểm của UART 27](#_heading=h.2bn6wsx)

[Bảng 3-1: Kết quả đo bệnh nhân Thái 46](#_heading=h.1mrcu09)

[Bảng 3-2: Nhịp tim bình thường ở từng độ tuổi 47](#_heading=h.46r0co2)

# LỜI MỞ ĐẦU

Ngày nay, sự bùng nổ của các ngành khoa học kỹ thuật, đặc biệt là ngành công nghệ thông tin đã làm cho đời sống con người ngày càng phát triển và hoàn thiện. Công nghệ thông tin (CNTT) đang dần chứng tỏ tầm ảnh hưởng rất lớn đến mọi mặt của đời sống xã hội. Đối với hoạt động của ngành y tế, có thể thấy rằng, CNTT ngày càng đóng vai trò quan trọng, không chỉ “bà đỡ” cho quá trình cải cách hành chính trong công tác quản lý, điều hành của cơ quan quản lý mà còn “đỡ đầu” cho việc triển khai và ứng dụng thành công các kỹ thuật cao trong công tác khám chữa bệnh. Việc nghiên cứu và phát triển các thiết bị y tế thông minh ứng dụng CNTT càng được con người chú trọng.

Cùng với đó, thiết bị Arduino đang được rất nhiều sinh viên lựa chọn để sử dụng trong dự án của mình bởi tính linh hoạt và chi phí học tập sử dụng thấp hơn nhiều so với các linh kiện khác. Ưu điểm của Arduino với người lập trình là ngôn ngữ dễ học, các ngoại vi trên bo mạch đều đã được chuẩn hóa nên không cần hiểu biết quá sâu về điện tử vẫn có thể dễ dàng lập trình được rất nhiều ứng dụng thú vị.

Ngoài ra, một thiết bị thông minh, tự động hóa không thể thiếu kết nối với Internet. Đây là lúc chúng ta cần tới ESP32. Linh kiện này giúp chúng ta xây dựng một hệ thống hoàn chỉnh với kết nối Wifi, kết nối Bluetooth, ADC độ phân giải cao, kết nối nối tiếp và nhiều tính năng khác.

Kết hợp những điều trên, với mong muốn phát triển một hệ thống hỗ trợ có tính ứng dụng cao trong y tế, nhóm chúng em quyết định chọn đề tài **Nghiên cứu, xây dựng thiết bị hỗ trợ chăm sóc bệnh nhân sử dụng Arduino Mega 2560** để làm báo cáo Thực tập cơ sở chuyên ngành.

**Mục tiêu đặt ra khi hoàn thiện đề tài:**

* Hiểu rõ đặc điểm, tính năng và cấu trúc của hệ thống chăm sóc bệnh nhân.
* Thiết kế thành công và chạy ổn định hệ thống.
* Kết nối điều khiển hệ thống thông qua Internet.

**Phạm vi nghiên cứu:**

* **Linh kiện, thiết bị sử dụng:**
* Kit Arduino Mega 2560
* Kit ESP32 DEVKIT V1-DOIT
* Module cảm biến nhịp tim GY-Max30100
* Led RGB hiển thị tình trạng (xanh-an toàn, vàng-cẩn thận, đỏ-nguy hiểm)
* Màn hình LCD 16x02
* **Môi trường phát triển:**
* Lập trình phần nhúng trên Arduino IDE
* Lập trình ứng dụng trên Android Studio
* **Ý nghĩa thực tiễn:**

Thiết bị tự động đo các chỉ số của bệnh nhân (nhịp tim, nhiệt độ), hiển thị các thông số đo được lên màn hình LCD và giao diện app, hiển thị tình trạng bệnh nhân theo màu led. Thiết bị này sẽ giúp nhân viên y tế, bệnh nhân cũng như người nhà luôn nắm rõ tình trạng của bệnh nhân một cách nhanh chóng, dễ dàng và kịp thời phát hiện bất thường.

**Nội dung báo cáo này gồm các phần:**

* Chương 1: Cơ sở lý thuyết
* Chương 2: Phân tích và thiết kế hệ thống
* Chương 3: Thực nghiệm và đánh giá hệ thống

Trong quá trình thực hiện đề tài, không thể tránh khỏi những sai sót, nhóm thực hiện mong rằng sẽ nhận được sự góp ý từ phía thầy để chúng em có thể hoàn thiện sản phẩm của mình hơn nữa.

Nhóm em xin chân thành cảm ơn!

# CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

## 1.1. Phát biểu bài toán

Hiện nay, khoa học ngày càng phát triển, con người đã có cũng như đang có nhiều bước tiến xa trong nhiều lĩnh vực, thực hiện được những việc mà trước đây tưởng chừng như không thể. Nhìn chung, tất cả cả các nỗ lực đó đều để phục vụ cho nhu cầu, đời sống con người ngày càng đầy đủ, tiện nghi hơn.

Theo WHO, bệnh tim mạch đang là nguyên nhân tử vong hàng đầu ở người trên toàn thế giới và chiếm nhiều nhất ở các nước đang phát triển. Mỗi năm, người chết do bệnh tim và đột quỵ nhiều hơn cả ung thư, lao, sốt rét và HIV cộng lại. Ở nước ta hiện nay, bệnh tim mạch là bệnh có nguy cơ tử vong cao. Từ năm 2000 thì bệnh tim mạch là bệnh gây tử vong hàng đầu.

Nhu cầu về một thiết bị đo nhịp tim và nhiệt độ cơ thể chính xác và giá cả phải chăng mà vẫn tạo ra cho chúng ta sự thoải mái, giải quyết được sự quá tải của đội ngũ y bác sĩ, hay theo dõi người bị bệnh trong phòng cách ly mà không cần phải tiếp xúc trực tiếp với người bệnh là điều tất yếu. Thiết bị này còn đặc biệt cần thiết đối với các bệnh nhân có bệnh lý về tim mạch. Tuy nhiên với các hạn chế như trình độ, thời gian thực hiện đề tài, yếu tố về công nghệ, đề tài sẽ chỉ tập chung vào giá trị nhịp tim, nhiệt độ cơ thể để nghiên cứu và xử lý. Thiết bị được thiết kế nhỏ nhẹ và linh động để phục vụ nhu cầu của bệnh nhân mọi lúc mọi nơi và đặc biệt thân thiện, dễ dàng sử dụng.

Nghiên cứu này hướng đến một sản phẩm mang tính ứng dụng cao, xây dựng hệ thống cho phép giám sát, theo dõi sức khỏe người dùng một cách dễ dàng và tiện dụng. Cụ thể là **Nghiên cứu, xây dựng hệ thống chăm sóc bệnh nhân sử dụng Arduino Mega 2560.**

## 1.2. Tổng quan về Arduino Mega 2560

### 1.2.1. Giới thiệu về Arduino Mega 2560

#### *1.2.1.1. Sự ra đời của Arduino Mega 2560*

Arduino Mega 2560 là phiên bản nâng cấp của Arduino Mega hay còn gọi là Arduino Mega 1280. Sự khác biệt lớn nhất với Arduino Mega 1280 chính là chip nhân.

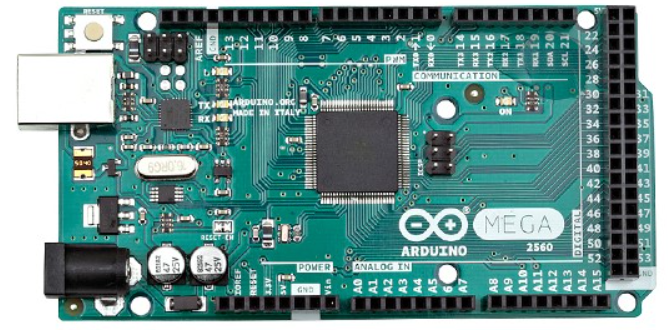
Ở Arduino Mega 1280 sử dụng:

* Chip ATmega1280.
* Flash memory 128KB.
* SRAM 8 KB.
* EEPROM 4 KB.

Arduino Mega 2560 phiên bản hiện đang được sử dụng rộng rãi và ứng dụng nhiều hơn.

* Chip ATmega2560.
* Bộ nhớ flash memory 256 KB.
* SRAM 8KB
* EEPROM 4 KB.

Giúp cho nhà phát triển khả năng viết những chương trình phức tạp hơn và điều khiển các thiết bị lớn hơn như máy in 3D, điều khiển robot.



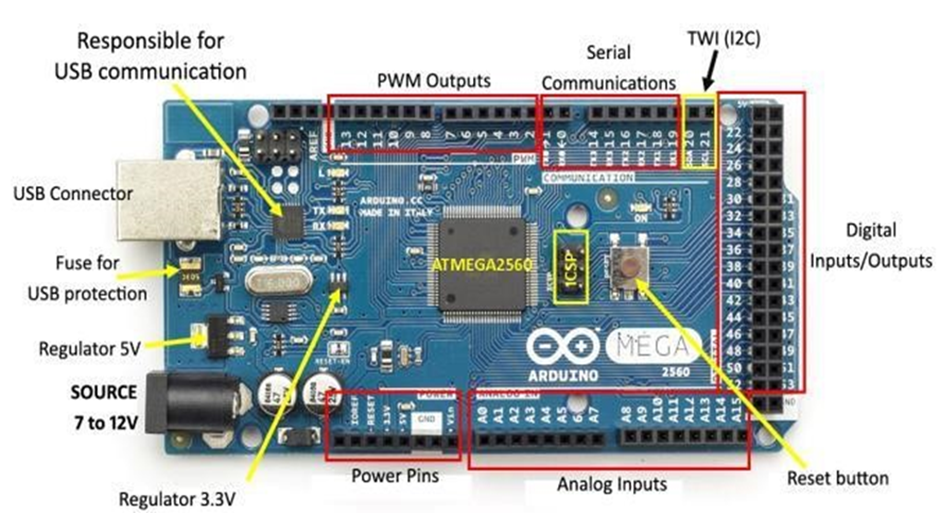
Hình 1-1: Arduino Mega 2560

#### *1.2.1.2. Đặc điểm của Arduino Mega 2560*

Bảng 1-1: Thông số của Arduino Mega 2560

| Vi điều khiển | AVR ATmega 2560 (8bit) |
| --- | --- |
| Nguồn cung cấp | 7-12V (Cho V-in hoặc jack 5.5 DC) hay 5v từ cổng USB |
| Số chân I/O số | 54 |
| Số chân I/O tương tự | 16 |
| Xung clock | 16 MHz ( nhà sản xuất cài đặt là 1MHz) |
| Bộ nhớ flash | 128 KB |
| SRAM | 8 KB |
| Giao tiếp | USB (Lập trình với ATmega8), ICSP (lập trình), GPIO, I2C và UART |
| Bộ Timer | 2 (8bit) + 4 (16bit) = 6 Timer |
| PWM | 12 (2-16 bit) |
| ADC | 16 (10 bit) |
| USART | 4 |
| Ngắt thay đổi chân | 24 |

#### *1.2.1.3. Cấu trúc của Arduino Mega 2560*



Hình 1-2: Các thành phần chức năng của board Arduino Mega

Cấu tạo chính của Arduino Mega 2560 bao gồm các phần sau:

● Cổng USB: đây là loại cổng giao tiếp để ta upload code từ PC lên vi điều khiển. Đồng thời nó cũng là giao tiếp serial để truyền dữ liệu giữa vi điều khiển và máy tính.

● Jack nguồn: để chạy Arduino thì có thể lấy nguồn 5v từ cổng USB ở trên, nhưng không phải lúc nào cũng cắm với máy tính được. Lúc đó ta cần nguồn DC từ 9v đến 12v cắm vào jack nguồn.

● Có 54 chân ra/vào số đánh số thứ tự từ 0 đến 53, ngoài ra có 3 chân nối đất (GND) và một chân điện áp tham chiếu (AREF).

● Vi điều khiển AVR: đây là bộ xử lý trung tâm của bo mạch. Với mỗi mẫu Arduino khác nhau thì chip khác nhau. Ở con Arduino Mega 2560 này thì sử dụng ATMega2560.

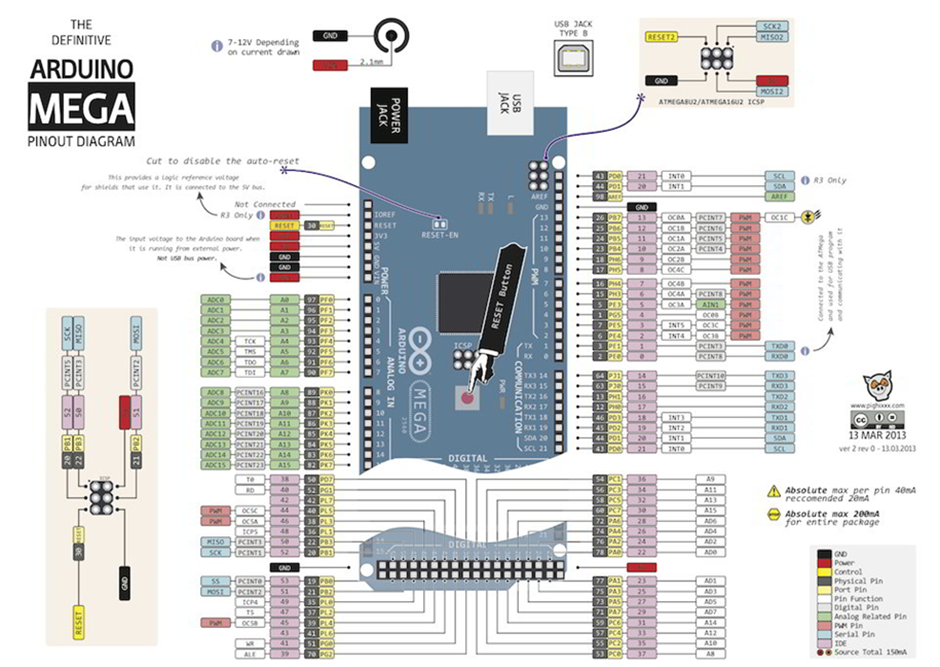
● Có một nút nhấn reset và một đầu ICSP.

Atmega2560:

● Các Mega 2560 có 16 chân vào tương tự, mỗi ngõ vào tương tự điều có độ phân giải 10bit (tức là 1024 giá trị khác nhau). Theo mặc định đo từ 0 đến 5 volts, mặc dù là nó có thể thay đổi phần trên của phạm vi bằng cách sử dụng chân Aref và analogReference chức năng.

● Các Atmega2560 có 256 KB bộ nhớ flash để lưu trữ mã (trong đó có 8KB được sử dụng cho bộ nạp khởi động), 8 KB SRAM và 4 KB EEPROM.

#### *1.2.1.4. Mô tả các chân của Arduino Mega 2560*



Hình 1-3: Sơ đồ chân của Arduino Mega 2560

**Chân điều khiển:**

RESET: Arduino Mega Mega 2560 có sẵn mạch reset với nút ấn để thiết lập lại hệ thống và chân này có thể được sử dụng khi kết nối các thiết bị khác để thiết lập lại bộ điều khiển.

XTAL1, XTAL2: Thạch anh(16Mhz) được kết nối với xung clock cung cấp cho bộ điều khiển.

AREF: Chân này được dùng khi sử dụng ADC để chuyển đổi tín hiệu với điện áp tham chiếu bên ngoài mà không muốn sử dụng điện áp tham chiếu nội bộ 1.1V hoặc 5V.

**Các chân Digital (70):**

Chân số: Từ 0-53 (số) và 0-15 (tương tự) có thể được sử dụng làm đầu vào hoặc đầu ra cho thiết bị được thiết lập bằng các hàm Mode (), digtalWrite (), digitalRead ().

Ứng dụng:

Thiết bị đầu ra: Relay, LED, buzzer, LCD và các thiết bị khác.

Thiết bị đầu vào: Nút ấn, cảm biến siêu âm, cần điều khiển và các thiết bị khác

**Chân tương tự (16):**

Từ 0-15 (analog) có thể được sử dụng như chân đầu vào tương tự cho bộ ADC, nếu không sử dụng nó hoạt động như chân digital bình thường. Nó được thiết lập bởi các hàm pinMode () khai báo chân, analogRead () để đọc trạng thái chân và nhận giá trị kỹ thuật số cho tín hiệu analog. Lưu ý phải cẩn thận để lựa chọn điện áp tham chiếu bên trong hoặc bên ngoài và chân Aref.

Ứng dụng :

Thiết bị đầu vào: Cảm biến nhiệt độ, cảm biến (như ldr, irled và độ ẩm) và các thiết bị khác

**Chân có Chức năng thay thế:**

***Chân SPI:*** Chân 22-SS, 23\_SCK, 24-MOSI, 25-MISO

Các chân này được sử dụng cho giao tiếp nối tiếp với giao thức SPI để liên lạc giữa 2 thiết bị trở lên. SPI cho phép bit phải được thiết lập để bắt đầu giao tiếp với các thiết bị khác.

Ứng dụng: Lập trình điều khiển AVR, giao tiếp với những người khác ngoại vi như LCD và thẻ SD.

***Chân I2C:***

Chân 20 cho SDA và 21 cho SCK (Tốc độ 400khz) để cho phép liên lạc hai dây với các thiết bị khác. Hàm được sử dụng là wire.begin () để bắt đầu chuyển đổi I2C, với wire.Read () để đọc dữ liệu i2c và wire.Write () để ghi dữ liệu i2c.

Ứng dụng:

Thiết bị đầu ra: LCD và liên lạc giữa nhiều thiết bị với hai dây.

Thiết bị đầu vào: RTC và các thiết bị khác.

***PWM chân :***

Chân 2-13 có thể được sử dụng như đầu ra PWM với hàm analogWrite () để ghi giá trị pwm từ 0-255.

Ứng dụng:

Thiết bị đầu ra: Điều khiển tốc độ của động cơ, ánh sáng mờ, pid cho hệ thống điều khiển hiệu quả.

***Chân UART:***

Chân 0 - RXD0, chân 1 - TXD0

Chân 19 - RXD1, chân 18 - TXD1

Chân 17 - RXD2, chân 16 - TXD2

Chân 15 - RXD3, chân 14 - TXD3

Chân này được sử dụng cho giao tiếp nối tiếp giữa bo mạch với máy tính hoặc hệ thống khác để chia sẻ và ghi dữ liệu. Nó được sử dụng với hàm serialBegin () để cài đặt tốc độ truyền và bắt đầu truyền thông với serial.Println () để in mảng ký tự (mảng char) ra thiết bị khác.

Ứng dụng:

Bộ điều khiển truyền thông và máy tính

***Chân ngắt :***

Chân digital: 0, 22, 23, 24, 25, 10, 11, 12, 13, 15, 14

Chân analog: 6, 7, 8, 9, 10, 11, 12, 13, 14, 15

Chân này được sử dụng để ngắt. Để kích hoạt chân ngắt phải cài đặt bật ngắt toàn cục.

Ứng dụng :

Bộ mã hóa vòng quay, nút bấm dựa trên ngắt và các nút khác.

**Chân ngắt phần cứng:**

Chân 18 - 21,2,3 ngắt phần cứng được sử dụng cho các ứng dụng ngắt. Ngắt phần cứng phải được bật với tính năng ngắt toàn cục để ngắt quãng từ các thiết bị khác.

Ứng dụng:

Nhấn nút cho chương trình ISR, đánh thức bộ điều khiển bằng thiết bị bên ngoài như cảm biến siêu âm và các thiết bị khác.

Một số lưu ý khi sử dụng Arduino Mega 2560:

● Khi bắt đầu sử dụng Arduino Mega 2560, bạn nên chú ý lựa chọn lại board. Bằng cách vào Tool → Board → Arduino Mega 2560. Nhằm tránh trước đó bạn đã sử dụng loại Arduino khác cổng vẫn còn nhận là board cũ nên khi build bạn sẽ gặp lỗi.

● Khi sử dụng chân RX, TX của Arduino, các bạn nên nhớ rút dây cắm tại 2 chân này ra rồi hãy bắt đầu upload. Sau đó hãy cắm lại bình thường và sử dụng để tránh gặp phải lỗi.

● Không được phép cắm trực tiếp chân GND vào chân nguồn 5V, có thể dẫn tới hỏng mạch.

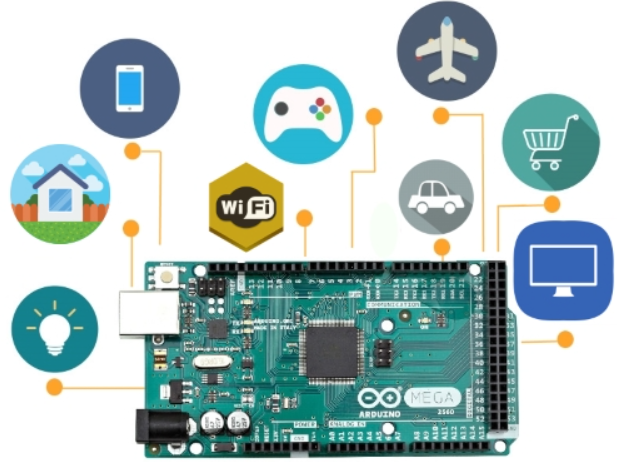
### 1.2.2. Ưu điểm và ứng dụng của Arduino Mega 2560

Ưu điểm của Arduino Mega 2560:

Arduino Mega 2560 được chọn sử dụng nhiều trong các dự án lớn nhỏ bởi đây là bo mạch được tích hợp nhiều tính năng nổi bật. Tính năng đầu tiên là thiết kế hệ thống I / O lớn với 16 bộ chuyển đổi tương tự và 54 bộ chuyển đổi digital hỗ trợ UART và các chế độ giao tiếp khác. Thứ hai, Arduino Mega 2560 có sẵn RTC và các tính năng khác như bộ so sánh, timer, ngắt để điều khiển hoạt động, tiết kiệm điện năng và tốc độ nhanh hơn với xung thạch anh16 Mhz.

Các tính năng khác bao gồm hỗ trợ JTAG để lập trình, gỡ lỗi và xử lý sự cố. Với bộ nhớ FLASH lớn và SRAM, bo này có thể xử lý chương trình hệ thống lớn một cách dễ dàng. Nó cũng tương thích với các loại bo mạch khác nhau như tín hiệu mức cao (5V) hoặc tín hiệu mức thấp (3.3V) với chân nạp I / O.

Brownout và watchdog giúp hệ thống đáng tin cậy và mạnh mẽ hơn. Nó hỗ trợ ICSP cũng như lập trình vi điều khiển USB với PC.

Arduino Mega 2560 là một sự thay thế của Arduino Mega cũ. Nó thường được sử dụng cho các dự án rất phức tạp.

Ứng dụng của Arduino Mega 2560:

Sở hữu tính năng mạnh mẽ, bộ nhớ flash rộng, số chân nhiều lên đến 54 chân digital và được hỗ trợ bởi số lượng shield lớn, Arduino Mega 2560 được đưa vào ứng dụng trong các dự án lớn như xử lý thông tin nhiều luồng, điều khiển động cơ, cánh tay robot, máy gắp thú bông, điều khiển từ xa, ma trận LED,... Ngoài ra, board Arduino Mega 2560 cải tiến mới còn giúp cho các kỹ thuật viên phát triển khả năng viết nhiều chương trình lập trình phức tạp hơn để điều khiển các thiết bị lớn hơn như máy in 3D, điều khiển robot.và mở rộng hơn nữa với Internet kết nối vạn vật.

## 1.3. Tổng quan về ESP32

### 1.3.1. Giới thiệu về ESP32

#### *1.3.1.1. Sự ra đời của ESP32*

ESP32 là một vi điều khiển giá rẻ, năng lượng thấp có hỗ trợ WiFi và dual-mode Bluetooth (tạm dịch: Bluetooth chế độ kép). Dòng ESP32 sử dụng bộ vi xử lý Tensilica Xtensa LX6 ở cả hai biến thể lõi kép và lõi đơn, và bao gồm các công tắc anten tích hợp, RF balun, bộ khuếch đại công suất, bộ khuếch đại thu nhiễu thấp, bộ lọc và module quản lý năng lượng.

ESP32 được chế tạo và phát triển bởi Espressif Systems, một công ty Trung Quốc có trụ sở tại Thượng Hải, và được sản xuất bởi TSMC bằng cách sử dụng công nghệ 40 nm. ESP32 là sản phẩm kế thừa từ vi điều khiển ESP8266.

#### *1.3.1.2. Đặc điểm của ESP32*

* ESP32 là lõi kép, điều này có nghĩa là nó có 2 bộ vi xử lý.
* Nó được tích hợp Wifi và bluetooth.
* Nó chạy các chương trình 32 bit.
* Xung nhịp (clock frequency) có thể lên đến 240MHz và nó có RAM 512 kB.
* Loại board này có 30 hoặc 36 chân, mỗi hàng có 15 chân.
* Nó cũng có sẵn nhiều loại ngoại vi (peripheral), như: cảm ứng điện dung (capacitive touch), ADC, DAC, UART, SPI, I2C và nhiều hơn nữa.
* Nó được tích hợp với cảm biến hiệu ứng Hall (Hall effect sensor) và cảm biến nhiệt độ

Bảng 1-2: Thông số kỹ thuật của ESP32 DEVKIT V1-DOIT

| Số lượng lõi | 2 (dual core) |
| --- | --- |
| Wi-Fi | 2,4 GHz lên đến 150 Mbits / s |
| Bluetooth | BLE (Bluetooth Low Energy) và legacy Bluetooth |
| Kiến trúc | 32 bit |
| Clock frequency | Lên đến 240 MHz |
| RAM | 512 KB |
| Số chân (Pin) | 30 hoặc 36 (phụ thuộc vào model) |
| Peripheral | Cảm ứng điện dung, ADC (Analog to Digital Converter), DAC (Digital to Analog Converter), I2C (Inter-Integrated Circuit), UART (universal asynchronous receiver/transmitter), CAN 2.0 (Controller Area Network), SPI (Serial Peripheral Interface), I2S (Integrated Inter-IC Sound), RMII (Reduced Media-Independent Interface), PWM (Pulse Width Modulation), v.v. |

#### *1.3.1.3. Cấu trúc của ESP32 DEVKIT V1-DOIT*

ESP32 có nhiều chân GPIO cùng với nhiều chức năng hơn so với ESP826.

Với ESP32, ta có thể quyết định chọn chân nào là UART, I2C hoặc SPI –  chỉ cần cài đặt chân đó trên code. Điều này có được là do chức năng ghép kênh của chip ESP32, cho phép gán nhiều chức năng cho cùng một chân (pin). Nếu không cài đặt trên code, các chân sẽ được sử dụng theo mặc định – như mô tả trong hình bên dưới (vị trí chân có thể thay đổi tùy thuộc vào nhà sản xuất).

Description: Graphical user interface, application

Description automatically generated

Hình 1-5: Sơ đồ chân ESP32 DEVKIT V1 – DOIT phiên bản 30 GPIO

#### *1.3.1.4. Mô tả các chân ESP32*

* **Chân Input Only**

GPIO từ 34 đến 39 là GPI – chân chỉ đầu vào. Các chân này không có điện trở kéo lên hoặc kéo xuống bên trong. Chúng không thể được sử dụng làm đầu ra, vì vậy chỉ sử dụng các chân này làm đầu vào:

* GPIO 34
* GPIO 35
* GPIO 36
* GPIO 39
* **Chân tích hợp Flash trên ESP32**

GPIO 6 đến GPIO 11 dùng để kết nối Flash SPI, không khuyến khích sử dụng trong các ứng dụng khác

* GPIO 6 (SCK/CLK)
* GPIO 7 (SDO/SD0)
* GPIO 8 (SDI/SD1)
* GPIO 9 (SHD/SD2)
* GPIO 10 (SWP/SD3)
* GPIO 11 (CSC/CMD)
* **Chân cảm biến điện dung**

Các chân ESP32 này có chức năng như 1 nút nhấn cảm ứng, có thể phát hiện sự thay đổi về điện áp cảm ứng trên chân.

Các cảm biến cảm ứng bên trong đó được kết nối với các GPIO sau:

* T0 (GPIO 4)
* T1 (GPIO 0)
* T2 (GPIO 2)
* T3 (GPIO 15)
* T4 (GPIO 13)
* T5 (GPIO 12)
* T6 (GPIO 14)
* T7 (GPIO 27)
* T8 (GPIO 33)
* T9 (GPIO 32)
* **Analog to Digital Converter (ADC)**

ESP32 có các kênh đầu vào ADC 18 x 12 bit (trong khi ESP8266 chỉ có ADC 1x 10 bit). Đây là các GPIO có thể được sử dụng làm ADC và các kênh tương ứng:

* ADC1\_CH0 (GPIO 36)
* ADC1\_CH1 (GPIO 37)
* ADC1\_CH2 (GPIO 38)
* ADC1\_CH3 (GPIO 39)
* ADC1\_CH4 (GPIO 32)
* ADC1\_CH5 (GPIO 33)
* ADC1\_CH6 (GPIO 34)
* ADC1\_CH7 (GPIO 35)
* ADC2\_CH0 (GPIO 4)
* ADC2\_CH1 (GPIO 0)
* ADC2\_CH2 (GPIO 2)
* ADC2\_CH3 (GPIO 15)
* ADC2\_CH4 (GPIO 13)
* ADC2\_CH5 (GPIO 12)
* ADC2\_CH6 (GPIO 14)
* ADC2\_CH7 (GPIO 27)
* ADC2\_CH8 (GPIO 25)
* ADC2\_CH9 (GPIO 26)

Các kênh đầu vào ADC có độ phân giải 12 bit. Điều này có nghĩa là ta có thể nhận được các số đọc tương tự từ 0 đến 4095, trong đó 0 tương ứng với 0V và 4095 đến 3,3V. Ta cũng có thể lập trình độ phân giải của các kênh của mình trên code.

* **Digital to Analog Converter (DAC)**

Có các kênh DAC 2 x 8 bit trên ESP32 để chuyển đổi tín hiệu kỹ thuật số thành đầu ra tín hiệu điện áp tương tự. Các kênh này chỉ có độ phân giải 8 bit, nghĩa là có giá trị từ 0 – 255 tương ứng với 0 – 3.3V

Đây là các kênh DAC:

* DAC1 (GPIO 25)
* DAC2 (GPIO 26)
* **Các chân thời gian thực RTC**

Các chân này có tác dụng đánh thức ESP32 khi trong chế độ Low Power Mode. Sử dụng như 1 chân ngắt ngoài.

Các chân RTC:

* RTC\_GPIO0 (GPIO 36)
* RTC\_GPIO3 (GPIO 39)
* RTC\_GPIO4 (GPIO 34)
* RTC\_GPIO5 (GPIO 35)
* RTC\_GPIO6 (GPIO 25)
* RTC\_GPIO7 (GPIO 26)
* RTC\_GPIO8 (GPIO 33)
* RTC\_GPIO9 (GPIO 32)
* RTC\_GPIO10 (GPIO4)
* RTC\_GPIO11 (GPIO0)
* RTC\_GPIO12 (GPIO2)
* RTC\_GPIO13 (GPIO15)
* RTC\_GPIO14 (GPIO 13)
* RTC\_GPIO15 (GPIO 12)
* RTC\_GPIO16 (GPIO 14)
* RTC\_GPIO17 (GPIO 27)
* **Chân PWM**

ESP32 LED PWM có 16 kênh độc lập có thể được định cấu hình để tạo tín hiệu PWM với các thuộc tính khác nhau. Tất cả các chân có thể hoạt động như đầu ra đều có thể được sử dụng làm chân PWM (GPIO từ 34 đến 39 không thể tạo PWM).

Để xuất PWM, ta cần xác định các thông số này trong code:

* Frequency – tần số
* Duty cycle
* Kênh PWM
* Chân GPIO nơi ta muốn xuất tín hiệu
* **Chân I2C**

ESP32 có hai kênh I2C và bất kỳ chân nào cũng có thể được đặt làm SDA hoặc SCL. Khi sử dụng ESP32 với Arduino IDE, các chân I2C mặc định là:

* GPIO 21 (SDA)
* GPIO 22 (SCL)
* **Chân Ngắt Ngoài**

Tất cả các chân ESP32 đều có thể sử dụng ngắt ngoài.

### 1.3.2. Ưu điểm và ứng dụng của ESP32

Có thể nói ESP32 là sự nâng cấp hoàn hảo của ESP8266, với ESP8266 phù hợp với các dự án nhỏ và tiết kiệm chi phí. ESP32 lại phù hợp với các dự án phức tạp hơn, tốc độ xử lý cao hơn và tích hợp nhiều ngoại vi mạnh mẽ hơn.

ESP8266 là 17 chân GPIO, ADC độ phân giải 10 bit, 8 kênh PWM mềm trong khi đó ESP 32 có tới 30/36 chân GPIO, 18 kênh ADC độ phân giải 12-bit, 16 kênh PWM mềm, Touch Sensor, Hall Effect Sensor, Ethernet MAC Interface, Cảm biến nhiệt độ được tích hợp sẵn.

Về bộ nhớ ESP32 có thêm 4MB External Flash và 520 KB SRAM (static random access memory) trong đó 8KB RAM RTC tốc độ cao – 8KB RAM RTC tốc độ thấp (dùng ở chế độ DeepSleep).

ESP32 hỗ trợ Bluetooth 4.2 và BLE (Bluetooth Low Energy). Việc hỗ trợ cả bluetooth khiến ESP32 có thể tương tác với các thiết bị như là bàn phím, chuột, điện thoại khi mà không có wifi.

Ultra Low Power giải quyết vấn đề năng lượng cho ESP bởi vì sử dụng Wi-Fi sẽ rất ngốn điện đặc biệt khi chúng ta sử dụng pin phải tính toán rất kỹ.

Ngoài ra ESP32 đang được rất nhiều các công ty trong và ngoài nước ưa chuộng, vậy nên làm chủ được dòng chip này sẽ giúp bạn kiếm được thu nhập tốt hơn là điều hiển nhiên.

## 1.4. Tổng quan về giao thức I2C

### 1.4.1. Giới thiệu về giao thức I2C

I2C là tên viết tắt của cụm từ tiếng anh “Inter-Integrated Circuit”. Nó là một giao thức giao tiếp được phát triển bởi Philips Semiconductors để truyền dữ liệu giữa một bộ xử lý trung tâm với nhiều IC trên cùng một board mạch chỉ sử dụng hai đường truyền tín hiệu.

Do tính đơn giản của nó nên loại giao thức này được sử dụng rộng rãi cho giao tiếp giữa vi điều khiển và mảng cảm biến, các thiết bị hiển thị, thiết bị IoT, EEPROMs, v.v …

Đây là một loại giao thức giao tiếp nối tiếp đồng bộ. Nó có nghĩa là các bit dữ liệu được truyền từng bit một theo các khoảng thời gian đều đặn được thiết lập bởi một tín hiệu đồng hồ tham chiếu.

### 1.4.2. Đặc điểm của giao thức I2C

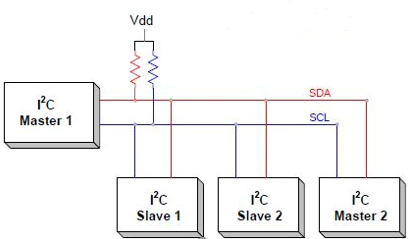
Sau đây là một số đặc điểm quan trọng của giao thức giao tiếp I2C:

* Chỉ cần có 2 đường bus (dây) chung để điều khiển bất kỳ thiết bị/IC nào trên mạng I2C
* Không cần thỏa thuận trước về tốc độ truyền dữ liệu như trong giao tiếp UART. Vì vậy tốc độ truyền dữ liệu có thể được điều chỉnh bất cứ khi nào cần thiết
* Cơ chế đơn giản để xác thực dữ liệu được truyền
* Sử dụng hệ thống địa chỉ 7 bit để xác định một thiết bị/IC cụ thể trên bus I2C
* Các mạng I2C dễ dàng mở rộng. Các thiết bị mới có thể được kết nối đơn giản với hai đường bus chung I2C

### 1.4.3. Cách thức hoạt động của I2C

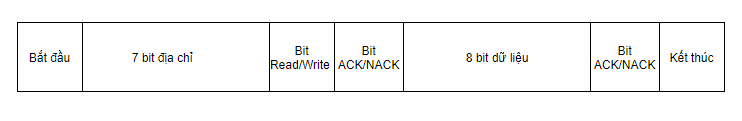
Cấu tạo:

* I2C sử dụng 2 đường truyền tín hiệu:
* SCL – Serial Clock Line: Tạo xung nhịp đồng hồ do Master phát đi
* SDA – Serial Data Line: Đường truyền nhận dữ liệu



Hình 1-6: Đường truyền tín hiệu của I2C

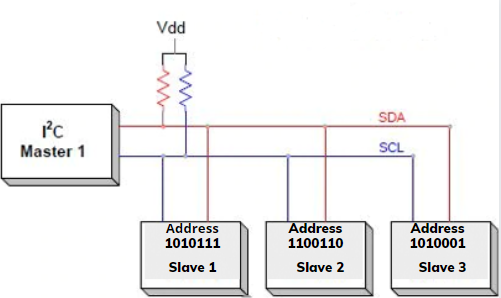
* Giao tiếp I2C bao gồm quá trình truyền nhận dữ liệu giữa các thiết bị chủ-tớ hay Master-Slave
* Thiết bị Master là một VĐK, nó có nhiệm vụ điều khiển đường tín hiệu SCL và gửi nhận dữ liệu hay lệnh thông qua đường SDA đến các thiết bị khác
* Các thiết bị nhận các dữ liệu lệnh và tín hiệu từ thiết bị Master được gọi là các thiết bị Slave. Các thiết bị Slave thường là các IC hoặc thậm chí là VĐK
* Master hay Slave được kết nối với nhau như hình trên. Hai đường bus SCL và SDA đều hoạt động ở chế độ Open Drain, nghĩa là bất cứ thiết bị nào kết nối với mạng I2C này cũng chỉ có thể kéo 2 đường bus này xuống mức thấp (LOW), nhưng lại không thể kéo được lên mức cao. Vì để tránh trường hợp bus vừa bị một thiết bị kéo lên mức cao vừa bị một thiết bị khác kéo xuống mức thấp gây hiện tượng ngắn mạch. Do đó, cần có một điện trở (từ 1 - 4,7kΩ) để giữ mặc định ở mức cao.



Hình 1-7: Khung truyền I2C

o  *Khối bit địa chỉ :*

Thông thường quá trình truyền nhận sẽ diễn ra với rất nhiều thiết bị, IC với nhau. Do đó để phân biệt các thiết bị này, chúng sẽ được gắn 1 địa chỉ vật lý 7 bit cố định.



Hình 1-8: Khối bit địa chỉ

o  *Bit Read/Write:*

Bit này dùng để xác định quá trình là truyền hay nhận dữ liệu từ thiết bị Master. Nếu Master gửi dữ liệu đi thì ứng với bit này bằng ‘0’, và ngược lại, nhận dữ liệu khi bit này bằng ‘1’.

o  *Bit ACK/NACK:*

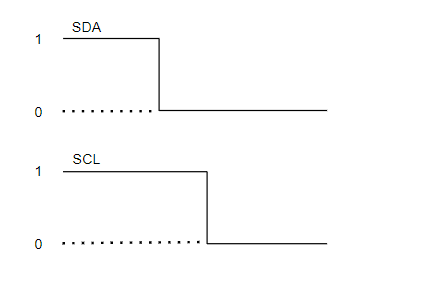
Viết tắt của Acknowledged / Not Acknowledged. Dùng để so sánh bit địa chỉ vật lý của thiết bị so với địa chỉ được gửi tới. Nếu trùng thì Slave sẽ được đặt bằng ‘0’ và ngược lại, nếu không thì mặc định bằng ‘1’.

o  *Khối bit dữ liệu:*

Gồm 8 bit và được thiết lập bởi thiết bị gửi truyền đến thiết bị nhân. Sau khi các bit này được gửi đi, lập tức 1 bit ACK/NACK được gửi ngay theo sau để xác nhận rằng thiết bị nhận đã nhận được dữ liệu thành công hay chưa. Nếu nhận thành công thì bit ACK/NACK được set bằng ‘0’ và ngược lại.

***Quá trình truyền nhận dữ liệu:***

* Bắt đầu: Thiết bị Master sẽ gửi đi 1 xung Start bằng cách kéo lần lượt các đường SDA, SCL từ mức 1 xuống 0.



Hình 1-9: Quá trình truyền nhận dữ liệu

* Tiếp theo đó, Master gửi đi 7 bit địa chỉ tới Slave muốn giao tiếp cùng với bit Read/Write.
* Slave sẽ so sánh địa chỉ vật lý với địa chỉ vừa được gửi tới. Nếu trùng khớp, Slave sẽ xác nhận bằng cách kéo đường SDA xuống 0 và set bit ACK/NACK bằng ‘0’. Nếu không trùng khớp thì SDA và bit ACK/NACK đều mặc định bằng ‘1’.
* Thiết bị Master sẽ gửi hoặc nhận khung bit dữ liệu. Nếu Master gửi đến Slave thì bit Read/Write ở mức 0. Ngược lại nếu nhận thì bit này ở mức 1.
* Nếu như khung dữ liệu đã được truyền đi thành công, bit ACK/NACK được set thành mức 0 để báo hiệu cho Master tiếp tục.
* Sau khi tất cả dữ liệu đã được gửi đến Slave thành công, Master sẽ phát 1 tín hiệu Stop để báo cho các Slave biết quá trình truyền đã kết thúc bằng các chuyển lần lượt SCL, SDA từ mức 0 lên mức 1.

***Các chế độ hoạt động của I2C:***

* Chế độ chuẩn (standard mode) với tốc độ 100 kBit/s.
* Chế độ tốc độ thấp (low speed mode) với tốc độ 10 kBit/s.

Ngoài ra, khác với giao tiếp SPI chỉ có thể có 1 Master, giao tiếp I2C cho phép chế độ truyền nhận dữ liệu giữa nhiều thiết bị Master khác nhau với thiết bị Slave. Tuy nhiên quá trình này có hơi phức tạp vì thiết bị Slave có thể nhận 1 lúc nhiều khung dữ liệu từ các thiết bị Master khác nhau, điều đó đôi khi dẫn đến xung đột hoặc sai sót dữ liệu nhận được.

Để tránh điều đó, khi làm việc ở chế độ này, mỗi thiết bị Master cần phát hiện xem đường SDA đang ở trạng thái nào.

Nếu SDA ở mức 0, nghĩa là đang có 1 thiết bị Master khác đang có quyền điều khiển và phải chờ đến khi truyền xong.

Ngược lại nếu SDA ở mức 1, nghĩa là đường truyền SDA đã an toàn và có sử dụng .

## 1.5. Tổng quan về giao thức UART

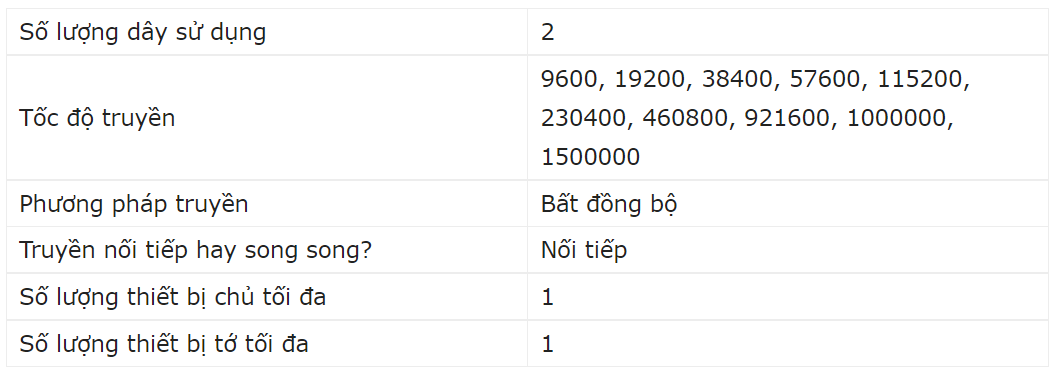
### 1.5.1. Giới thiệu về giao thức UART

UART là một giao thức truyền thông phần cứng sử dụng giao tiếp nối tiếp không đồng bộ với tốc độ có thể định cấu hình. Không đồng bộ có nghĩa là không có tín hiệu đồng hồ để đồng bộ hóa các bit đầu ra từ thiết bị truyền đi đến bên nhận.

### 1.5.2. Đặc điểm của UART

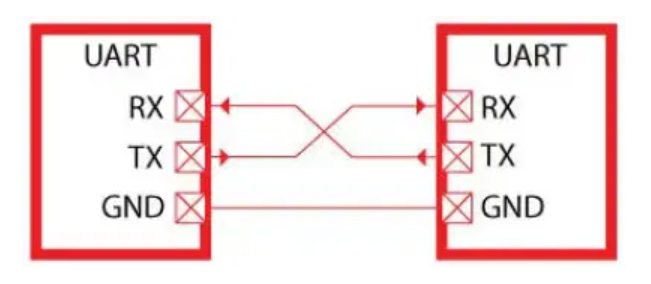
Cả 2 UART cũng phải được cấu hình để truyền và nhận cùng một cấu trúc gói dữ liệu.

Bảng 1-3: Đặc điểm của UART

****

### 1.5.3. Cách thức hoạt động của giao thức UART

Trong giao tiếp UART, hai UART giao tiếp trực tiếp với nhau. UART truyền chuyển đổi dữ liệu song song từ một thiết bị điều khiển như CPU thành dạng nối tiếp, truyền nó nối tiếp đến UART nhận, sau đó chuyển đổi dữ liệu nối tiếp trở lại thành dữ liệu song song cho thiết bị nhận.



Hình 1-10: Kết nối UART

Hai đường dây mà mỗi thiết bị UART sử dụng để truyền dữ liệu đó là:

* Transmitter (Tx)
* Receiver (Rx)

UART truyền dữ liệu không đồng bộ, có nghĩa là không có tín hiệu đồng hồ để đồng bộ hóa đầu ra của các bit từ UART truyền đến việc lấy mẫu các bit bởi UART nhận. Thay vì tín hiệu đồng hồ, UART truyền thêm các bit start và stop vào gói dữ liệu được chuyển. Các bit này xác định điểm bắt đầu và điểm kết thúc của gói dữ liệu để UART nhận biết khi nào bắt đầu đọc các bit.

Khi UART nhận phát hiện một bit start, nó bắt đầu đọc các bit đến ở một tần số cụ thể được gọi là tốc độ truyền (baud rate). Tốc độ truyền là thước đo tốc độ truyền dữ liệu, được biểu thị bằng bit trên giây (bps – bit per second). Cả hai UART đều phải hoạt động ở cùng một tốc độ truyền. Tốc độ truyền giữa UART truyền và nhận chỉ có thể chênh lệch khoảng 10% trước khi thời gian của các bit bị lệch quá xa.

## 1.6. Giới thiệu về Arduino IDE

### 1.6.1. Tổng quan về lập trình Arduino

Có thể mọi người đã quen lập trình trên PC, với những ngôn ngữ như C, C++, C#, Java, Python, Ruby...

Nhưng có biết rằng phần mềm trên PC chỉ chiếm khoảng 10% sản lượng phần mềm trên thị trường. 90% còn lại là code điều khiển tivi, máy giặt, điều hòa, tủ lạnh... tóm lại là tất cả các thiết bị điện tử xung quanh ta. Lập trình theo hướng này được gọi là embedded computing, hay physical computing, tức là lập trình để con người tương tác với các thiết bị thực.

Để người thiết kế có thể nhanh chóng đưa ra được mẫu thể hiện ý tưởng của mình, rất cần phải có những platform để dễ dàng prototyping. Và một trong những platform đang được sử dụng rất nhiều trong prototyping là Arduino.

Arduino là một bo mạch xử lý được dùng để lập trình tương tác với các thiết bị phần cứng như cảm biến, động cơ,... Điểm hấp dẫn ở Arduino là ngôn ngữ cực kì dễ học (giống C/C++), các ngoại vi trên bo mạch đều đã được chuẩn hóa, nên không cần biết nhiều về điện tử, cũng có thể lập trình được những ứng dụng thú vị. Thêm nữa, vì Arduino là một platform đã được chuẩn hóa, nên đã có rất nhiều các bo mạch mở rộng (gọi là shield) để cắm chồng lên bo mạch Arduino, có thể hình dung như là "library" của các ngôn ngữ lập trình. Ví dụ, muốn kết nối Internet thì có Ethernet shield, muốn điều khiển động cơ thì có Motor shield, muốn kết nối nhận tin nhắn thì có GSM shield,... Rất đơn giản, và chỉ phải tập trung vào việc "lắp ghép" các thành phần này và sáng tạo ra các ứng dụng cần thiết.

Có thể kể ở đây một số ứng dụng hay của Arduino:

Robot: Arduino được dùng để làm bộ xử lý trung tâm của rất nhiều loại robot. Đó là nhờ vào khả năng đọc các thiết bị cảm biến, điều khiển động cơ,... của Arduino.

· Game tương tác: chúng ta có thể dùng Arduino để tương tác với Joystick, màn hình,... để chơi các trò như Tetris, phá gạch, Mario…; máy bay không người lái. Và nhiều ví dụ khác nữa,...

Như vậy, tuy là một bo mạch nhỏ như thế, nhưng Arduino có thể dùng vào rất nhiều ứng dụng thú vị khác nhau. Vậy để phát triển ứng dụng dựa trên Arduino, cần những gì?

Rất đơn giản, chỉ cần IDE phát, một dây kết nối USB loại A-B, và một bo mạch Arduino là có thể bắt đầu.

Ngôn ngữ lập trình của Arduino chính là C/C++, nhưng so với lập trình lập trình trực tiếp với vi điều khiển, lập trình với Arduino đơn giản hơn nhiều vì chỉ phải giao tiếp với phần cứng thông qua các thư viện, có thể xem như các lớp C++ wrapper lên các giao tiếp với phần cứng. Trên website, có khá nhiều các library viết sẵn để điều khiển ngoại vi: LCD, sensor, motor... việc cần làm là kết hợp chúng với nhau để tạo ứng dụng cho riêng mình.

Một chương trình arduino được gọi là sketch, sẽ được upload lên bo mạch Arduino qua cổng USB.

Phân tích chương trình: có 2 method quan trọng nhất là setup() và loop().

setup(): làm nhiệm vụ khởi tạo model cho các ngoại vi của Arduino. Hàm này sẽ được chạy một lần khi bo mạch Arduino được reset. Ở chương trình này, setup() chỉ làm nhiệm vụ đặt các chân 4,5,6,7 của Arduino sang mode output.

loop(): là chương trình chính của Arduino. Đoạn code trong loop() sẽ được Arduino chạy vô hạn. Trong chương trình này, có hàm digitalWrite() để đặt các chân (pin) ở mức điện áp cao (HIGH) hay thấp (LOW). Hàm tiếp theo là delay(), nhận đối số là một số nguyên, thể hiện số mili giây ta muốn chương trình tạm ngưng.

### 1.6.2. Tổng quan về Arduino IDE

Arduino IDE là phần mềm dùng để lập trình cho Arduino. Môi trường lập trình Arduino IDE có thể chạy trên ba nền tảng phổ biến nhất hiện nay là Windows, Macintosh os và Linux. Do có tính chất nguồn mở nên môi trường lập trình này hoàn toàn miễn phí và có thể mở rộng thêm bởi người dùng có kinh nghiệm.

Nó bao gồm một trình soạn thảo mã với các tính năng như làm nổi bật cú pháp, khớp dấu ngoặc khối chương trình, thụt đầu dòng tự động và cũng có khả năng biên dịch và tải lên các chương trình vào board mạch với một nhấp chuột duy nhất. Một chương trình hoặc mã viết cho Arduino được gọi là "sketch". Chương trình Arduino được viết bằng C hoặc C++. Arduino IDE đi kèm với một thư viện phần mềm được gọi là "Wiring" từ dự án lắp ráp ban đầu, cho hoạt động đầu vào/đầu ra phổ biến trở nên dễ dàng hơn nhiều.

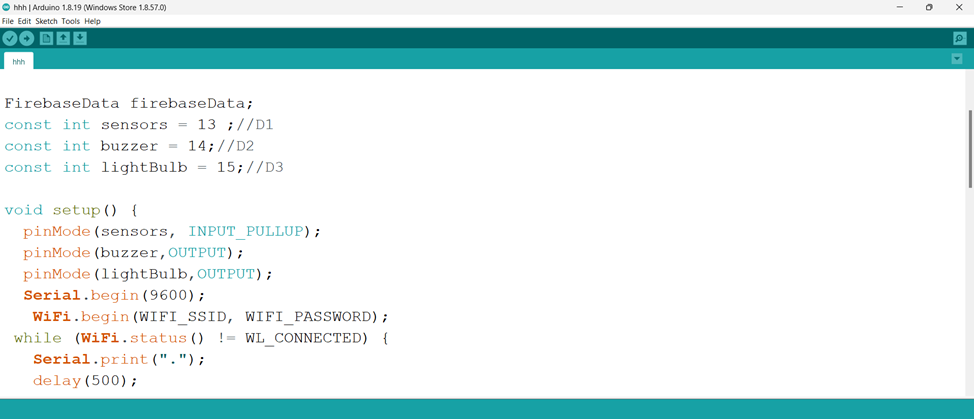
Người sử dụng chỉ cần định nghĩa hai hàm để thực hiện một chương trình điều hành theo chu kỳ. Khi các chúng ta bật điện bảng mạch Arduino, reset hay nạp chương trình mới, hàm setup() sẽ được gọi đến đầu tiên. Sau khi xử lý xong hàm setup(), Arduino sẽ nhảy đến hàm loop() và lặp vô hạn hàm này cho đến khi tắt điện board mạch Arduino. Dưới đây là giao diện của phần mềm. Ngôn ngữ lập trình có thể được mở rộng thông qua các thư viện C++. Và do ngôn ngữ lập trình này dựa trên nền tảng ngôn ngữ c của AVR nẽn người dùng hoàn toàn có thể nhúng thêm code viết bằng AVR vào chương trình nếu muốn.

Hiện tại, Arduino IDE có thể download từ trang chủ http://arduino.cc/ bao gồm các phiên bản sau:

● Arduino 1.0.5

● Arduino.5.5 BETA (Hỗ trợ cho 2 board Arduino mới nhất là: Arduino Yun và Arduino Due).

● Arduino IDE cho Intel Galileo.



Hình 1-11: Giao diện chính của phần mềm Arduino IDE

## 1.7. Giới thiệu về Firebase

### 1.7.1 Tổng quan về Firebase

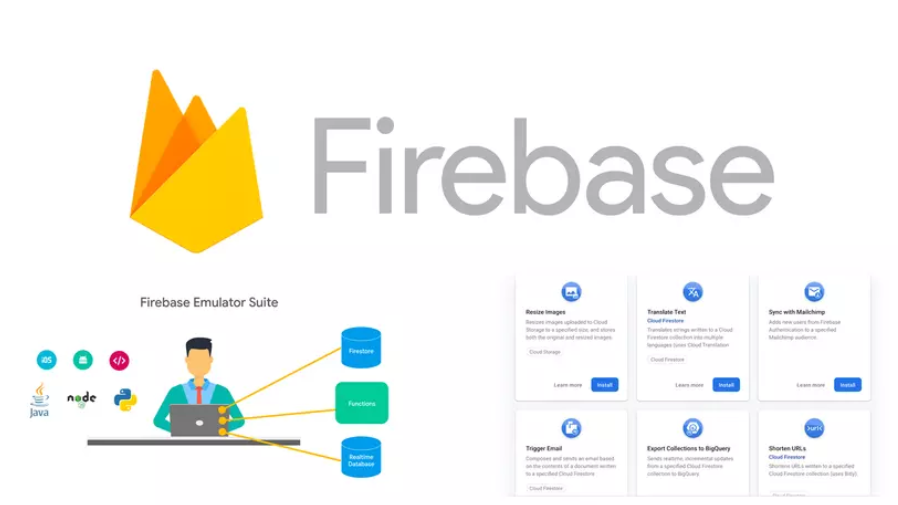
Firebase là một nền tảng sở hữu bởi Google giúp chúng ta phát triển các ứng dụng di động và web. Họ cung cấp rất nhiều công cụ và dịch vụ tiện ích để phát triển ứng dụng nên một ứng dụng chất lượng cụ thể là các dịch vụ cơ sở dữ liệu hoạt động trên nền tảng đám mây với hệ thống máy chủ cực kỳ mạnh mẽ của Google. Chức năng chính của firebase là giúp người dùng lập trình ứng dụng, phần mềm trên các nền tảng web, di động bằng cách đơn giản hóa các thao tác với cơ sở dữ liệu.

### 1.7.2 Lịch sử phát triển của Firebase

Firebase phát triển từ Envolve, một công ty khởi nghiệp trước đó do James Tamplin và Andrew Lee thành lập vào năm 2011. Họ thành lập Firebase như một công ty vào tháng 9 - 2011. Đến tháng 4 năm 2012 Firebase đã lần đầu tiên được ra mắt công chúng.

Sản phẩm đầu tiên của Firebase là Cơ sở dữ liệu thời gian thực (Firebase realtime database), một API đồng bộ hóa dữ liệu ứng dụng trên các thiết bị iOS, Android và Web, đồng thời lưu trữ trên đám mây của Firebase. Sản phẩm hỗ trợ các nhà phát triển phần mềm trong việc xây dựng các ứng dụng cộng tác, theo thời gian thực.

Vào tháng 10 năm 2014, Firebase đã được Google mua lại. Từ đó đến nay, Firebase đã ra mắt thêm nhiều tính năng mới và được nhiều nhà phát triển ưa thích sử dụng trong các dự án của mình.



Hình 1-12: Firebase là nền tảng phát triển ứng dụng đa năng của di động và website

### 1.7.3. Những service nổi bật

#### 1.7.3.1. Real-time Database

Firebase realtime database là một cơ sở dữ liệu thời gian thực, NoSQL được lưu trữ đám mây cho phép bạn lưu trữ và đồng bộ dữ liệu. Dữ liệu được lưu trữ dưới dạng cây Json, và được đồng bộ theo thời gian thực đối với mọi kết nối.

Khi bạn xây dựng những ứng dụng đa nền tảng như Android, IOS và Web App, tất cả các client của bạn sẽ kết nối trên cùng một cơ sở dữ liệu Firebase và tự động cập nhật dữ liệu mới nhất khi có sự thay đổi.



Hình 1-13: Tạo một dự án với Firebase

#### 1.7.3.2. Authentication

Firebase Authentication là chức năng xác thực người dùng.Hiểu một cách đơn giản, app của bạn cần phải đăng nhập/ đăng ký tài khoản để sử dụng, Firebase cung cấp cho chúng ta chức năng xác thực người dùng bằng email, số điện thoại, hay tài khoản Facebook, Google,...

#### 1.7.3.3. Firebase Cloud Storage

Firebase Cloud Storage là một không gian lưu trữ dữ liệu, nó giống như một chiếc ổ cứng. Bạn có thể upload và download các loại file bạn muốn. Đó có thể là một file ảnh, hay file văn bản, .zip, …

#### 1.7.3.4. Firebase Cloud Function

Cloud Functions Firebase cho phép chúng ta viết những câu truy vấn database lưu trữ trên cloud. Code của bạn được lưu trữ trên cloud của Google và chạy trong một môi trường bảo mật, được quản lý. Bạn không cần quan tâm đến vấn đề mở rộng các máy chủ.Với firebase, khi bạn muốn lấy dữ liệu bạn cần phải viết các câu truy vấn trực tiếp từ client.

#### 1.7.3.5. Firebase Analytics

Firebase Analytics là tính năng giúp bạn phân tích hành vi của người sử dụng trên ứng dụng của bạn. Cuối cùng nó sẽ đưa ra lời khuyên về lộ trình xây dựng ứng dụng.

## 1.8. Giới thiệu về Android Studio

### 1.8.1. Tổng quan về hệ điều hành Android

Androidlà hệ điều hành mã nguồn mở, dựa trên Linux Kernel, dành cho các thiết bị di động nói chung (điện thoại, máy tính bảng, đồng hồ thông minh, máy nghe nhạc,…).

Có nghĩa là Android không chỉ giới hạn trong phạm vi một hệ điều hành cho điện thoại! Nó có thể được nhà sản xuất cài đặt lên đồng hồ, máy nghe nhạc, thiết bị định vị GPS, thậm chí là ô tô (các thiết bị Android Auto).

Android cũng không phải là một thiết bị hay sản phẩm cụ thể, nó là một hệ điều hành dựa trên Linux, nguồn mở, linh hoạt.

Hiện Android là một thương hiệu của Google. Có khả năng tùy biến rất cao và có thể chạy trên nhiều thiết bị, nhiều kiến trúc vi xử lý (ARM / x86).

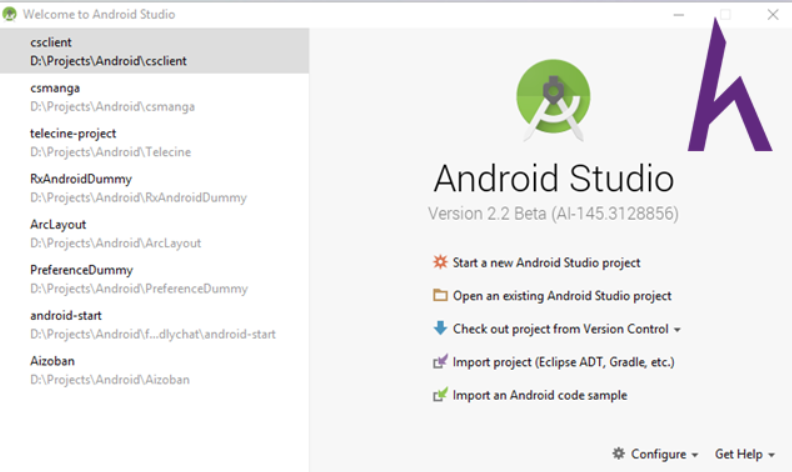
### 1.8.2. Tổng quan về Android Studio

Tháng 5 năm 2013, Google công bố Android Studio, một môi trường phát triển ứng dụng tích hợp (IDE) dành riêng cho Android, mã nguồn mở, dựa trên IDE Java IntelliJ của hãng JetBrains (đối thủ với Eclipse và Netbeans, vốn khá quen thuộc với dân lập trình Java).

Android Studio chạy trên Windows, Mac và Linux, nhằm thay thế cho Eclipse Android Development Tool (ADT) vốn được sử dụng làm IDE chính trong các năm trước đó.

Android Studio được đóng gói với một bộ code editor, debugger, các công cụ performance tool và một hệ thống build/deploy (trong đó có trình giả lập emulator để giả lập môi trường của thiết bị điện thoại hoặc tablet trên máy tính) cho phép các lập trình viên có thể nhanh chóng phát triển các ứng dụng từ đơn giản tới phức tạp.

Việc xây dựng một ứng dụng mobile (hoặc tablet) bao gồm rất nhiều các công đoạn khác nhau. Đầu tiên chúng ta sẽ viết code ứng dụng sử dụng máy tính cá nhân hoặc laptop. Sau đó chúng ta cần build ứng dụng để tạo file cài đặt. Sau khi build xong thì chúng ta cần copy file cài đặt này vào thiết bị mobile (hoặc table) để tiến hành cài đặt ứng dụng và chạy kiểm thử (testing). Bạn thử tưởng tượng nếu với mỗi lần viết một dòng code bạn lại phải build ứng dụng, cài đặt trên điện thoại hoặc tablet và sau đó chạy thử thì sẽ vô cùng tốn thời gian và công sức. Android Studio được phát triển để giải quyết các vấn đề này. Với Android Studio tất cả các công đoạn trên được thực hiện trên cùng một máy tính và các quy trình được tinh gọn tới mức tối giản nhất.



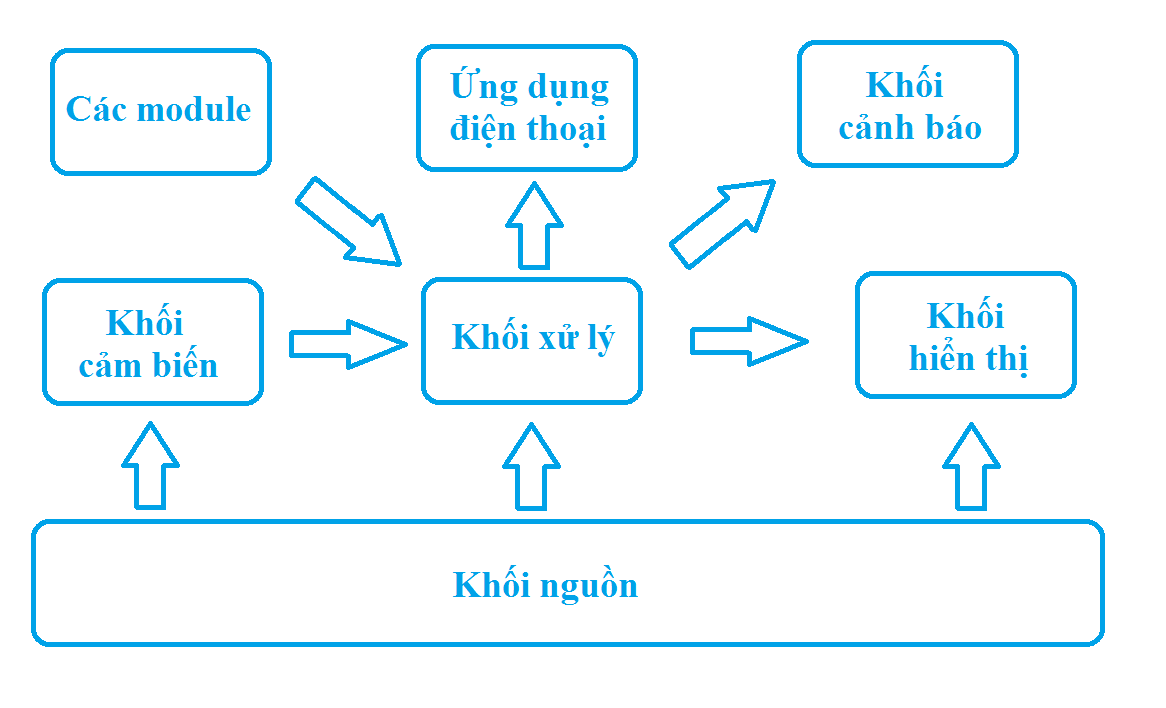
Hình 1-14: Giao diện màn hình đầu của Android Studio

### 1.8.3. Các tính năng của Android Studio

* Bộ công cụ build ứng dụng dựa trên Gradle (thay vì Maven).
* Chức năng dò và sửa lỗi nhanh, hướng Android.
* Công cụ chỉnh sửa màn hình dạng kéo thả tiện lợi.
* Các wizard tích hợp nhằm giúp lập trình viên tạo ứng dụng từ mẫu có sẵn.
* Tích hợp Google Cloud Platform, dễ dàng tích hợp với Google Cloud Messaging và App Engine của Google.

# CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

## 2.1. Sơ đồ khối chức năng



Hình 2-1: Sơ đồ khối chức năng

* Khối nguồn: cung cấp nguồn điện cho các khối khác hoạt động với nguồn điện cung cấp là 5V.
* Khối hiển thị: hiển thị các thông số nhịp tim, nhiệt độ của người sử dụng lên màn hình LCD.
* Khối cảm biến: gồm có cảm biến MAX30100.
* Khối xử lý: Arduino Mega 2560 tiếp nhận và thực thi các lệnh điều khiển từ khối cảm biến đồng thời điều khiển khối hiển thị.
* Khối truyền tải: ESP32 kết nối với Arduino đưa dữ liệu thời gian thực lên nền tảng firebase sau đó hiển thị lên giao diện app giúp người dùng dễ dàng đọc được các thông số chỉ trên một thiết bị di động.

## 2.2. Thiết kế phần cứng

### 2.2.1. Lựa chọn linh kiện thiết kế mô hình

#### *2.2.1.1. Kit Arduino Mega 2560*

Sở hữu tính năng mạnh mẽ, bộ nhớ flash rộng, số chân nhiều lên đến 54 chân digital và được hỗ trợ bởi số lượng shield lớn, Arduino Mega 2560 được đưa vào ứng dụng trong các dự án lớn như xử lý thông tin nhiều luồng, điều khiển động cơ, cánh tay robot, máy gắp thú bông, điều khiển từ xa, ma trận LED,... Ngoài ra, board Arduino Mega 2560 cải tiến mới còn giúp cho các kỹ thuật viên phát triển khả năng viết nhiều chương trình lập trình phức tạp hơn để điều khiển các thiết bị lớn hơn như máy in 3D, điều khiển robot và mở rộng hơn nữa với Internet kết nối vạn vật.

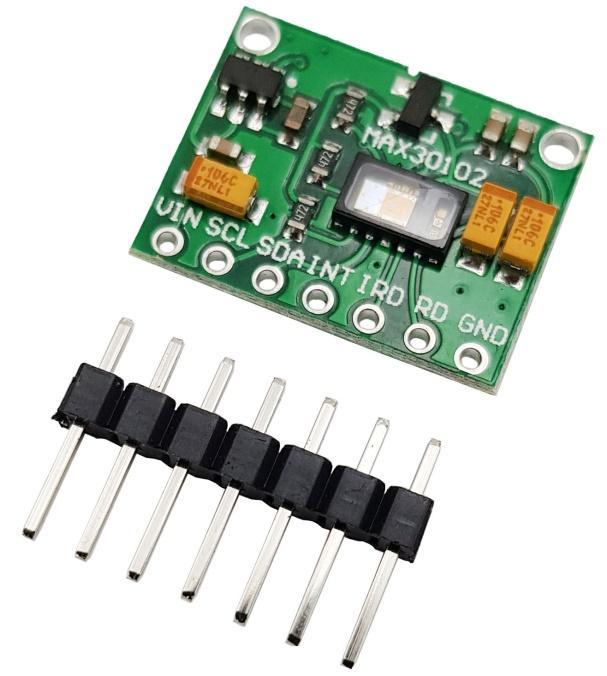
#### *2.2.1.2. Kit ESP32 DEVKIT V1-DOIT*

Là phiên bản cải tiến của ESP8266, ESP32 hỗ trợ thêm truyền nhận Bluetooth, RAM nhiều hơn. Tốc độ xử lý nhanh hơn, số chân GPIO nhiều hơn, nhiều cổng giao tiếp hơn, nhiều chân PWM hơn, nhiều chân ADC hơn, tích hợp cả 3 loại cảm biến (nhiệt độ, hall, touch sensor...). ESP32 hiện đang đứng đầu về cấu hình cũng như tốc độ xử lý.

#### *2.2.1.3. Module cảm biến nhịp tim MAX30100*

Cảm biến nhịp tim MAX30100 là một mô-đun đo nhịp tim tích hợp. Nó bao gồm đèn led bên trong, bộ tách sóng quang, các bộ phận quang học và các thiết bị điện tử có tiếng ồn thấp với khả năng loại bỏ ánh sáng xung quanh. Nó hoạt động trên một nguồn cung cấp điện 1,8V và một nguồn cấp điện 5V riêng biệt cho các đèn led bên trong.

Giao tiếp thông qua giao diện tương thích I2C tiêu chuẩn và có thể được tắt thông qua phần mềm với chế độ chờ bằng không. Nó cho phép các thanh ray nguồn vẫn được cấp nguồn ở mọi thời điểm.



Hình 2-2: Module cảm biến nhịp tim MAX30100

Thông số kỹ thuật

* IC chính: MAX30100
* Đo được nhịp tim.
* Điện áp sử dụng 2,5V ~ 5,5VDC
* Nhỏ gọn, siêu tiết kiệm năng lượng, thích hợp cho các thiết bị đo nhỏ gọn, Wearable Devices.
* Giao tiếp: I2C, mức tín hiệu TTl.
* Kích thước: 1,9cm x 1,4cm x 0,3cm

Cảm biến MAX30100 được sử dụng để đo nhịp tim thích hợp cho nhiều ứng dụng liên quan đến y sinh, 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 sẵn có trên Arduino rất dễ sử dụng.

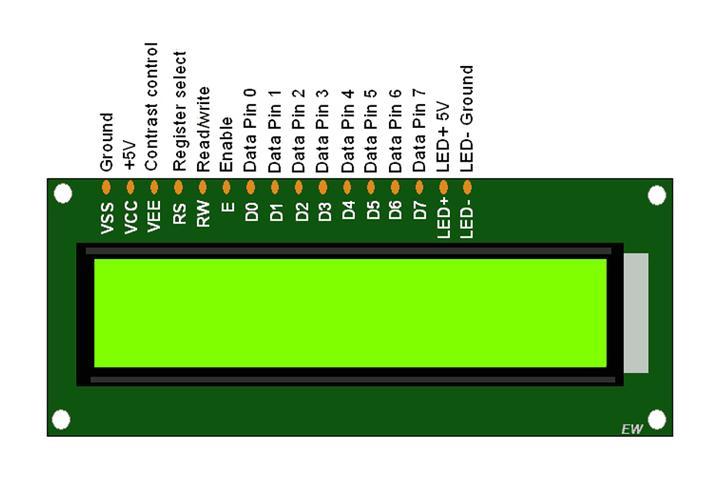
Cảm biến nhịp tim MAX30100 được tích hợp 16-bit sigma delta ADC và bộ xử lý tín hiệu tương tự với độ nhiễu thấp giúp cảm biến hoạt động chính xác và ổn định cao. Đồng thời được thiết kế nhỏ gọn có thể sử dụng làm thiết bị đeo tay và dễ dàng giao tiếp với các MCU, Arduino và Raspberry Pi.

#### *2.2.1.4. Màn hình LCD 1602*



Hình 2-3: Màn hình LCD 1602

Màn hình LCD1602 sử dụng driver HD44780, có khả năng hiển thị 2 dòng với mỗi dòng 16 ký tự. Màn hình LCD 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.

THÔNG SỐ KỸ THUẬT

* Điện áp hoạt động: 5V
* Kích thước: 8 x 3,6 x 0,8 cm
* Màu nền: xanh
* Màu chữ: đen
* Khoảng cách giữa 2 chân kết nối: 0,1 inch
* Tiện dụng khi kết nối với Breadboard

Ưu điểm của LCD1602:

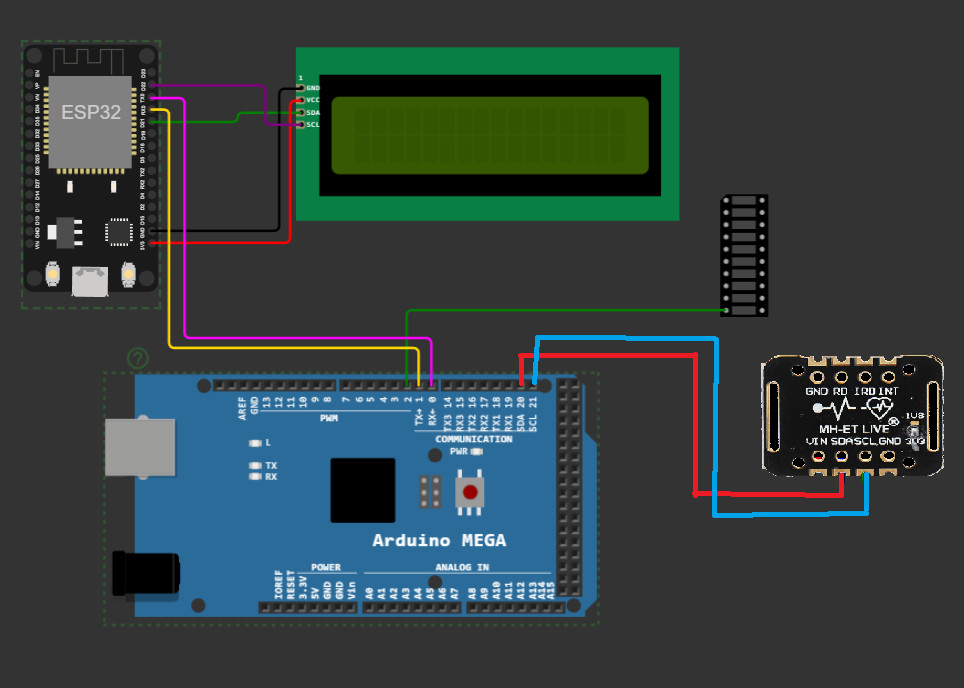
* Tên các chân được ghi ở mặt sau của màn hình LCD hỗ trợ việc kết nối, đi dây điện.
* Có đèn led nền, có thể dùng biến trở hoặc PWM điều chỉnh độ sáng để sử dụng ít điện năng hơn.
* Có thể được điều khiển với 6 dây tín hiệu.
* Có bộ ký tự được xây dựng hỗ trợ tiếng Anh và tiếng Nhật.

#### *2.2.1.5. Led RGB*

* Led là hệ thống chiếu sáng được hình thành từ nguồn ánh sáng bán dẫn để tạo ra ánh sáng.
* Led RGB là sự phối hợp về cường độ ánh sáng của 3 nguồn ánh sáng cơ bản là: Màu đỏ (Red), màu xanh lá (Green) và màu xanh dương (Blue).
* Dựa vào sự phối hợp trên có thể tạo ra được Led có khả năng đổi màu bằng cách ghép 3 ánh sáng nêu trên lại với nhau.
* Led RGB cung cấp phương tiện kiểm soát và thay đổi màu sắc của ánh sáng.

### 2.2.2. Sơ đồ đấu nối

* + Màn hình LCD và cảm biến Max30100 kết nối với Arduino Mega qua 2 cổng SCL và SDA của giao thức I2C.
  + Led hiển thị tình trạng sử dụng 1 chân digital của Arduino Mega.
  + ESP32 sử dụng 1 cổng UART để kết nối với mega2560.



Hình 2-5: Sơ đồ đấu nối

### 2.2.3. Nguyên lý hoạt động

- Kit Arduino Mega 2560 là bộ xử lý chính , được nạp code để xử lý các dữ liệu được gửi về từ các cảm biến và module.

- Cảm biến nhịp tim MAX30100 sử dụng để đo nhịp tim truyền tín hiệu về kit Arduino qua SDA & SCL.

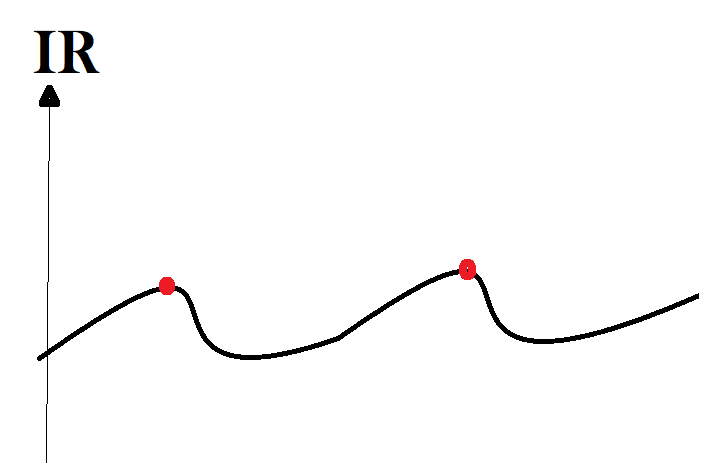
- Màn LCD hiển thị các thông số, thông tin cần thiết , kết nối với mạch chuyển giao diện HW-061.

- Mạch chuyển giao diện HW-061 là mạch gắn kết màn hình LCD (thông qua 16 chân kết nối) với kit Arduino (SCL & SDA).

- LED RGB hiển thị tình trạng sức khỏe của người bệnh (người thân có thể quan sát từ xa mà không cần lại gần qua ứng dụng) với các trường hợp: Đỏ (nguy hiểm), Vàng (cần quan tâm), Xanh (đang ổn). Kết nối với kit qua 1 dây tín hiệu digital.

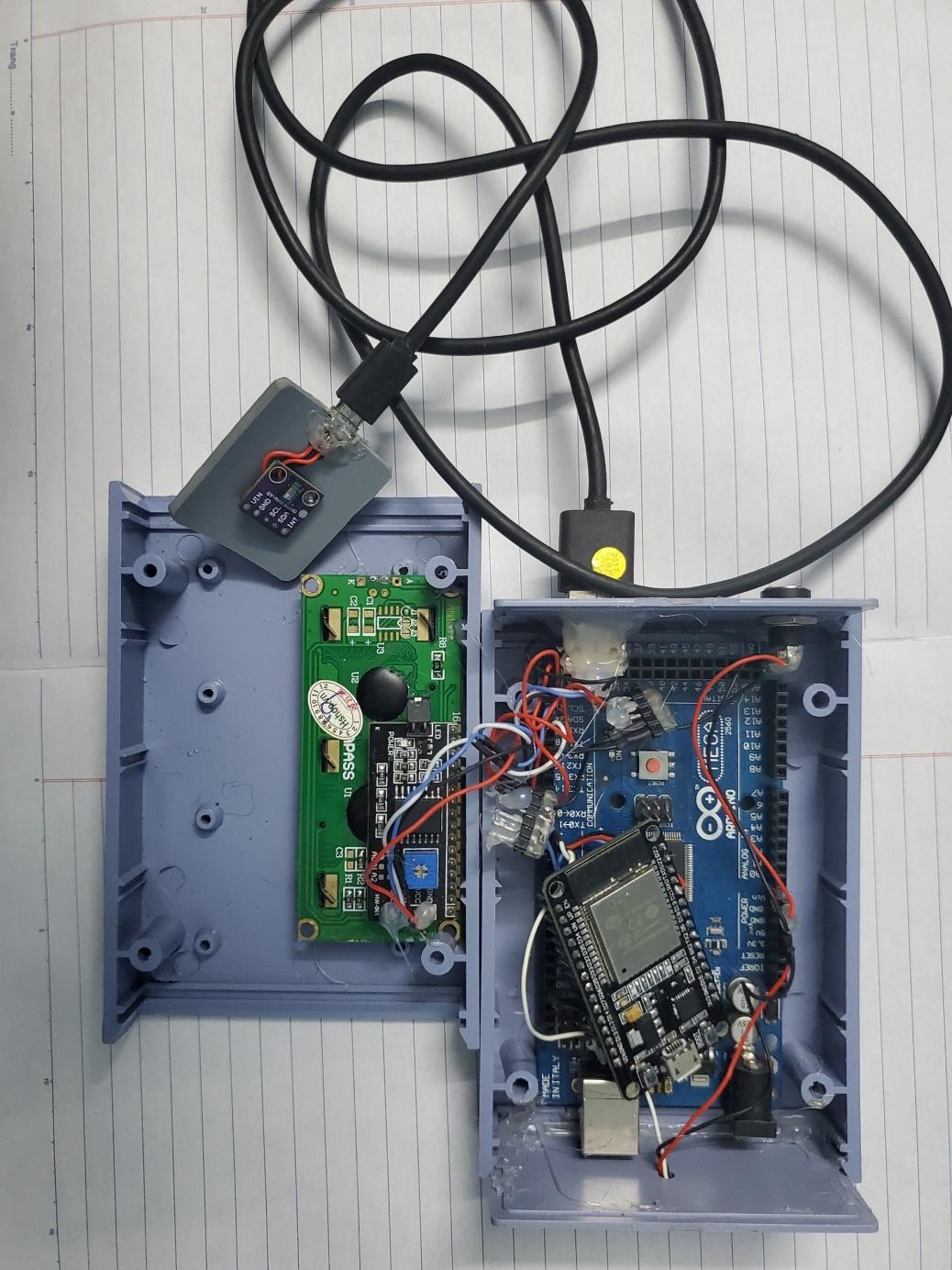
\*Nguyên lý hoạt động của cảm biến đo nhịp tim:

Thực chất cảm biến đo nhịp tim chính là một cảm biến dao động có độ nhạy rất cao, khi tim đập nó sẽ tạo một lực đẩy rất nhỏ tác động lên vành của mạch máu từ đó lớp biểu bì cũng sẽ dao động theo, khi đó điện trở (IR) trong cảm biến sẽ tăng lên (tương tự như những cảm biến nhiệt độ), người ta cũng sẽ tích hợp một con trở Shunt để đo dòng từ đó trả được giá trị về. Từ một hàm nhận biết nhịp tim trong thư viện Max30102 sẽ cho chúng ta biết đó có phải là nhịp tim hay không? sau khi lấy được 2 giá trị gần nhất là nhịp tim thì nó sẽ tính toán xem khoảng thời gian giữa 2 nhịp là bao nhiêu và lấy 60 giây chia cho khoảng thời gian đó thì chúng ta sẽ được nhịp tim trung bình.



### 2.2.4. Thi công phần cứng

Lắp đặt, nối các linh kiện lại với nhau theo sơ đồ đấu nối ở trên



Hình 2-6: Sơ đồ phần cứng

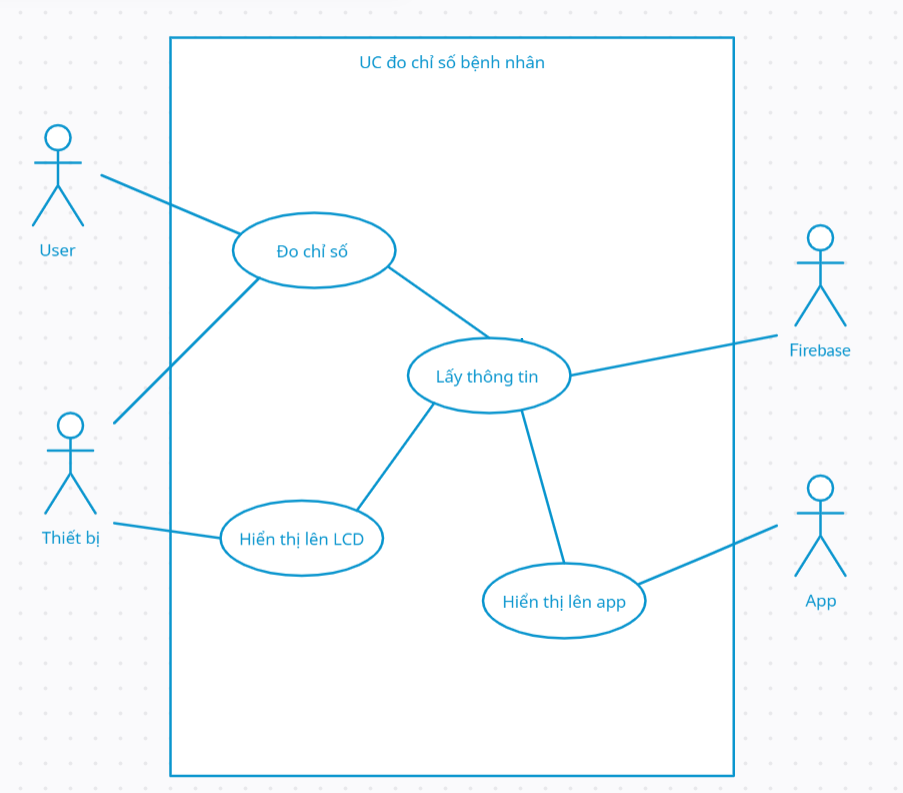
## 2.3. Thiết kế phần mềm

### 2.3.1. Lưu đồ thuật toán

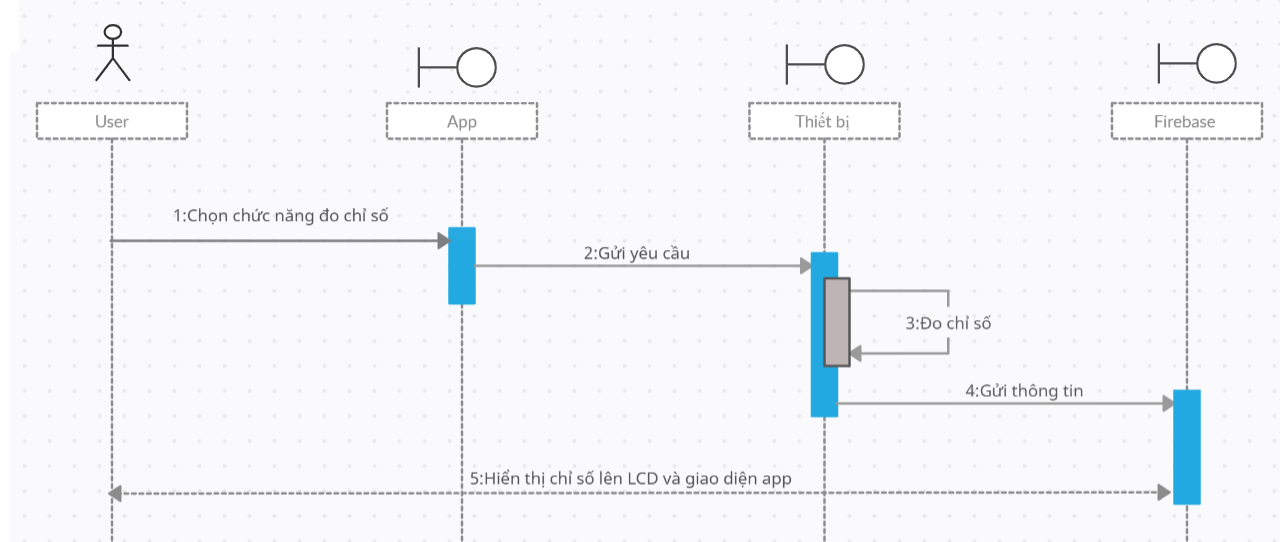


Hình 2-7: Lưu đồ thuật toán

### 2.3.2. Các biểu đồ usecase và tuần tự



Hình 2-8: Sơ đồ usecase đo các chỉ số



Hình 2-9: Biểu đồ tuần tự đo các chỉ số

# CHƯƠNG 3: THỰC NGHIỆM VÀ ĐÁNH GIÁ HỆ THỐNG

## 3.1. Mô hình thiết bị y tế chăm sóc bệnh nhân

### 3.1.1. Các chức năng của thiết bị

Đo nhịp tim và nhiệt độ của bệnh nhân. Các chỉ số nhịp tim và nhiệt độ cơ thể được hiển thị đồng thời lên màn hình LCD và giao diện app trên thiết bị di động. Từ đó giúp người dùng theo dõi được tình trạng sức khỏe của bản thân/người thân, các bác sĩ quản lý được sức khoẻ bệnh nhân từ xa để từ đó đưa ra những phác đồ điều trị thích hợp.



Hình 3-1: Khi chưa đo



Hình 3-2: Khi đo sẽ hiển thị kết quả

### 3.1.2. Kết quả thực nghiệm đo các chỉ số của bệnh nhân

Bảng 3-1: Kết quả đo bệnh nhân Thái

| Lần đo | Nhịp tim (BPM) | Nhiệt độ (°C) |
| --- | --- | --- |
| Lần 1 | 67 | 37,1 |
| Lần 2 | 85 | 36,5 |
| Lần 3 | 77 | 36,7 |
| Lần 4 | 59 | 36,8 |
| Lần 5 | 65 | 36,5 |

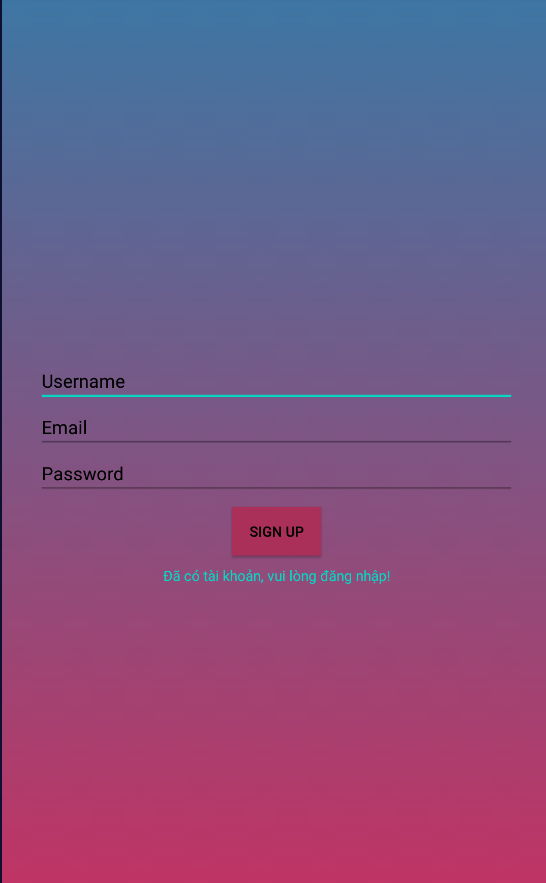
Nhịp tim không bình thường xuất hiện khi tần số nhịp tim quá nhanh hoặc quá chậm. Nếu nhịp tim trên 100 nhịp trên một phút được coi là nhịp tim nhanh. Còn nhịp tim dưới 50 nhịp trên một phút được coi là nhịp tim chậm. Nhịp tim không bình thường sẽ dẫn đến bệnh rối loạn nhịp tim. Ở người lớn, nhịp tim bình thường khi nghỉ ngơi rơi vào khoảng 60-100 nhịp trên một phút. Tuy nhiên theo nhiều chuyên gia Tim mạch, nhịp tim lý tưởng ở người trưởng thành khỏe mạnh là từ 60 - 80 nhịp/phút.

Bảng 3-2: Nhịp tim bình thường ở từng độ tuổi

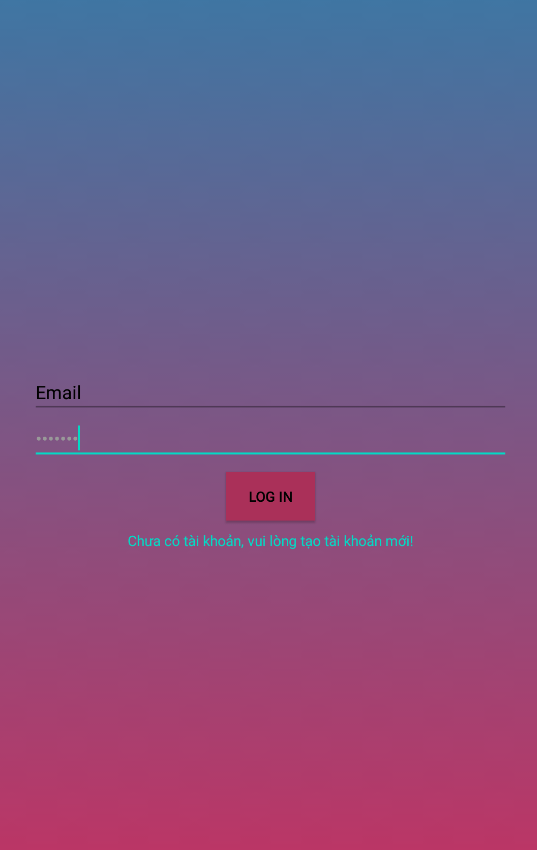
| TUỔI | NAM | NỮ |
| --- | --- | --- |
| < 1 | 102-155 | 104 – 156 |
| 1 | 95-137 | 95- 139 |
| 2-3 | 85-124 | 88 – 125 |
| 4-5 | 74-112 | 76-117 |
| 6-8 | 66-105 | 69-106 |
| 9-11 | 61-97 | 66-103 |
| 12-15 | 57-97 | 60-99 |
| 16-19 | 52-92 | 58-99 |
| 20-39 | 52-89 | 57-95 |
| 40-59 | 52-90 | 56-92 |
| 60-79 | 50-91 | 56-92 |
| > 80 | 51-94 | 56-93 |

## 3.2. Giao diện ứng dụng trên điện thoại

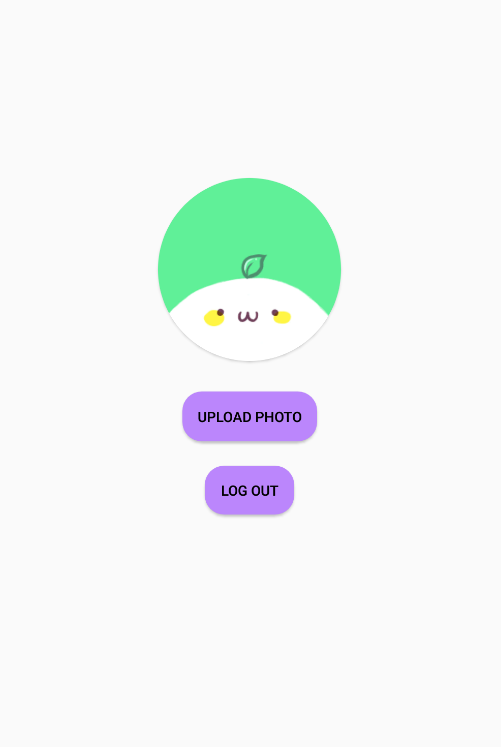
Giao diện đăng ký tài khoản với email: tài khoản email yêu cầu phải đúng định dạng, password có tối thiểu 6 ký tự. Nếu một trong các yêu cầu trên không được nhập đúng sẽ có thông báo đăng ký lỗi được hiển thị.



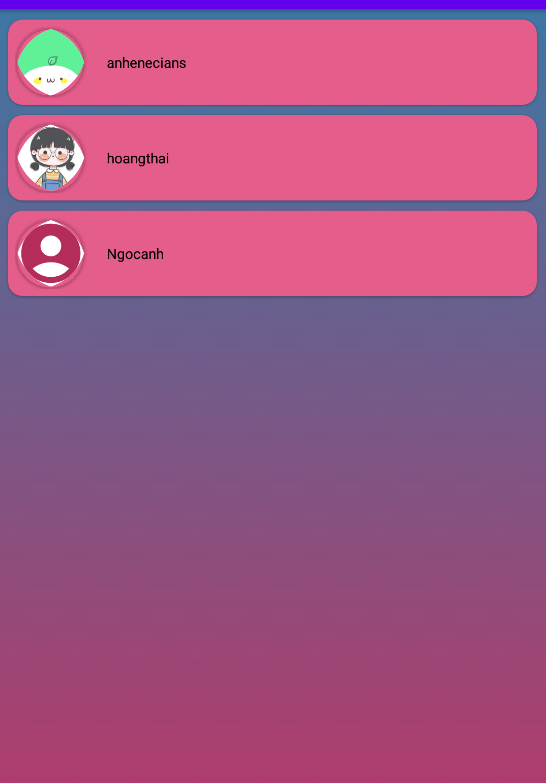
Giao diện đăng nhập



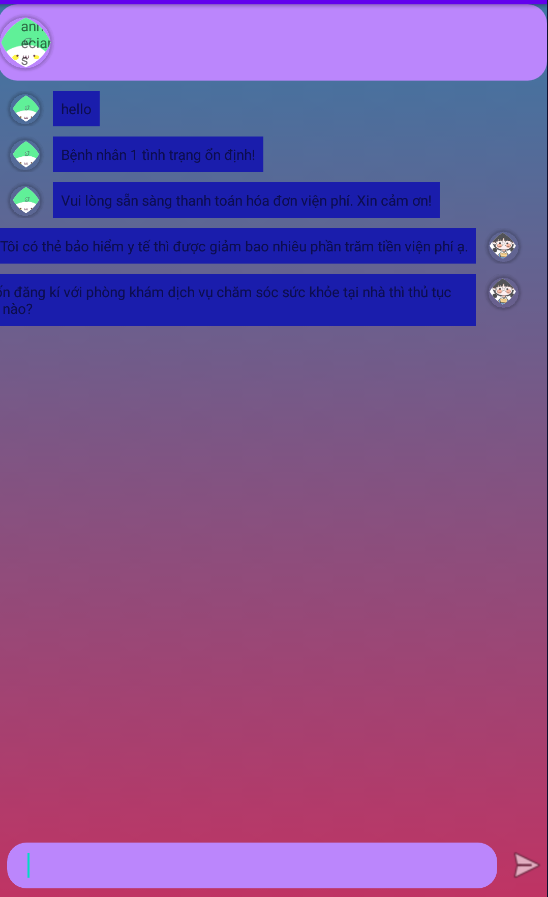
Chức năng đặt ảnh đại diện và nút LogOut



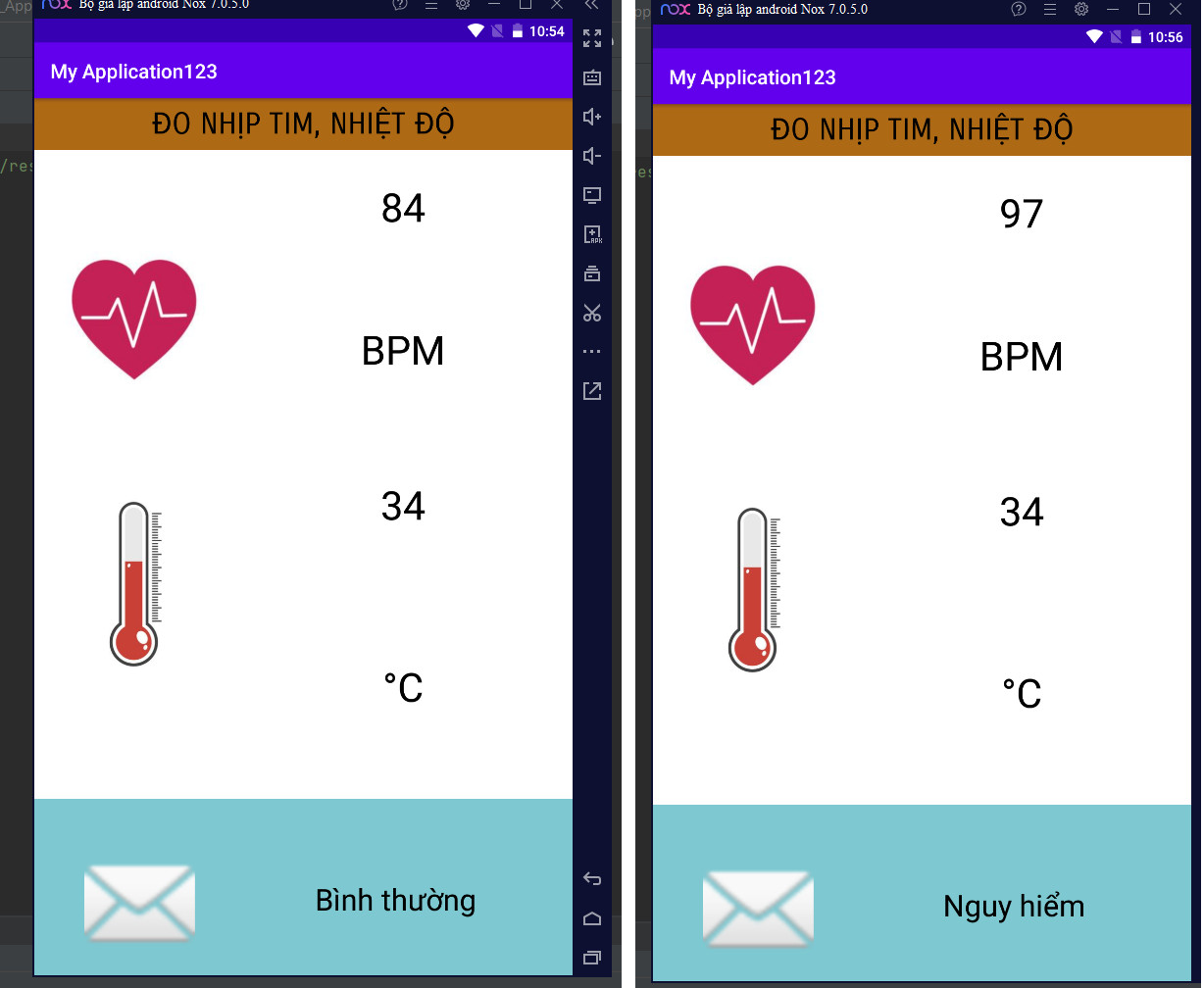
Hiển thị các tài khoản có trong hệ thống để tiến hành chat



Giao diện chat giữa hai tài khoản



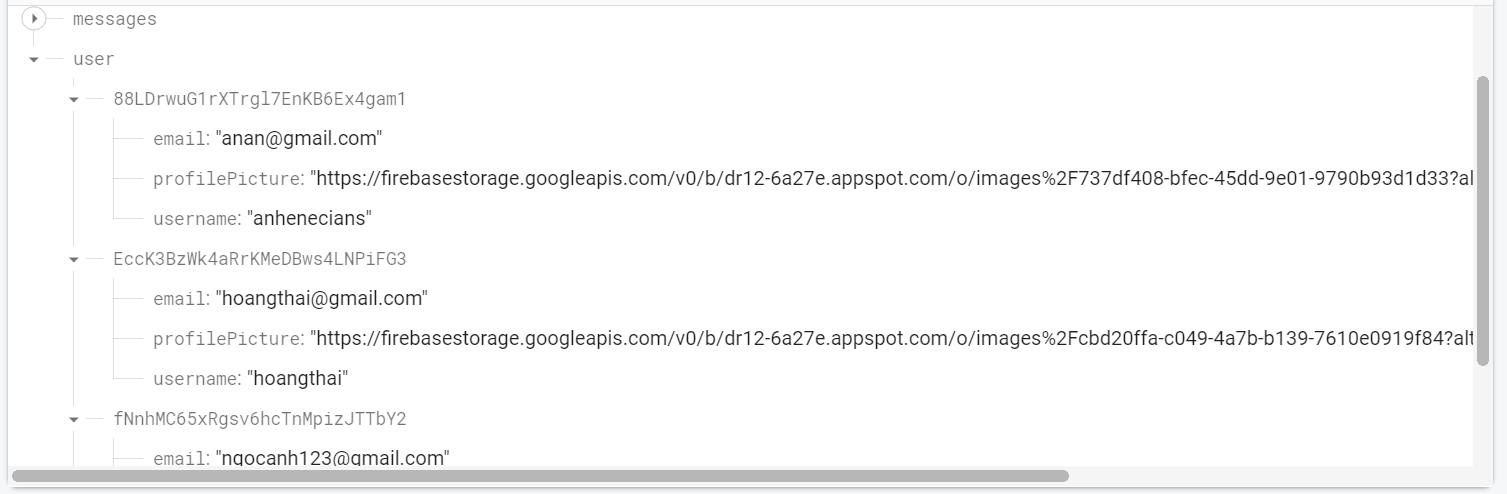
Hiển thị nhịp tim, nhiệt độ và tình trạng của bệnh nhân



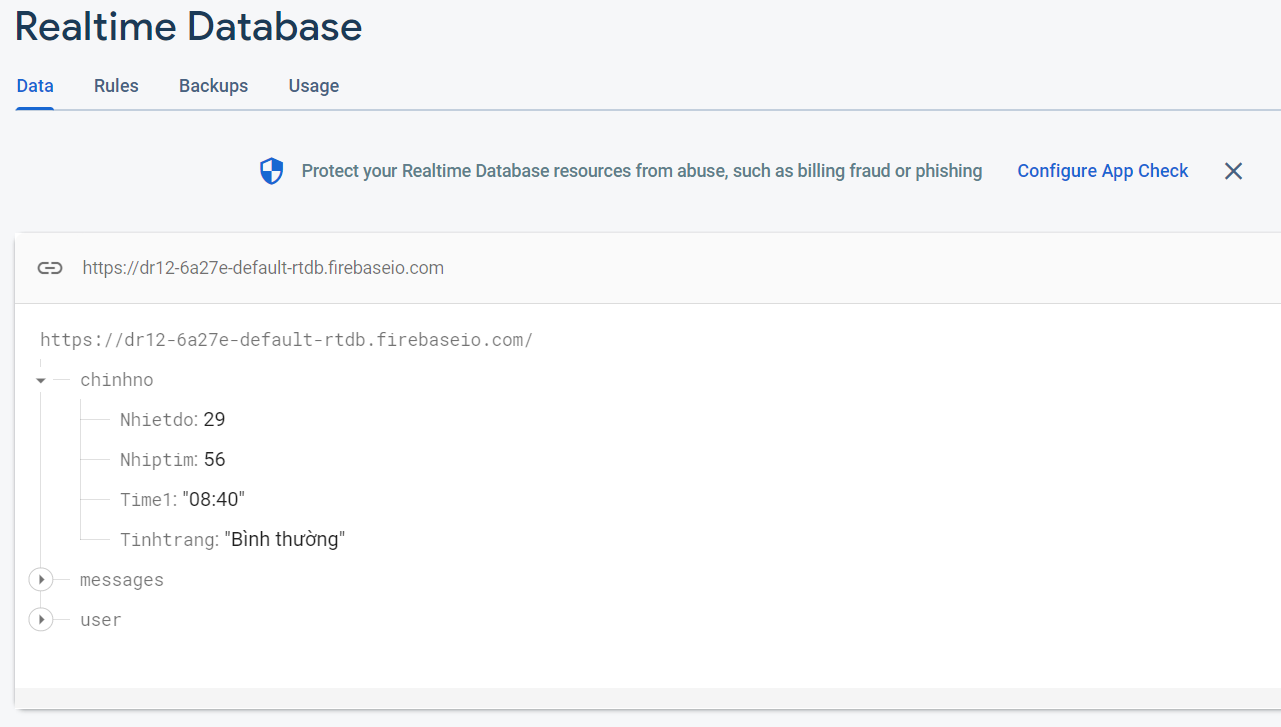
Hình 3-3: Giao diện ứng dụng khi đo

Dữ liệu trên firebase:

Danh sách các tài khoản đã thực hiện đăng ký thành công được đẩy lên firebase với node user: chủ sở hữu firebase có thể thực hiện sửa và xóa thông tin các tài khoản này.



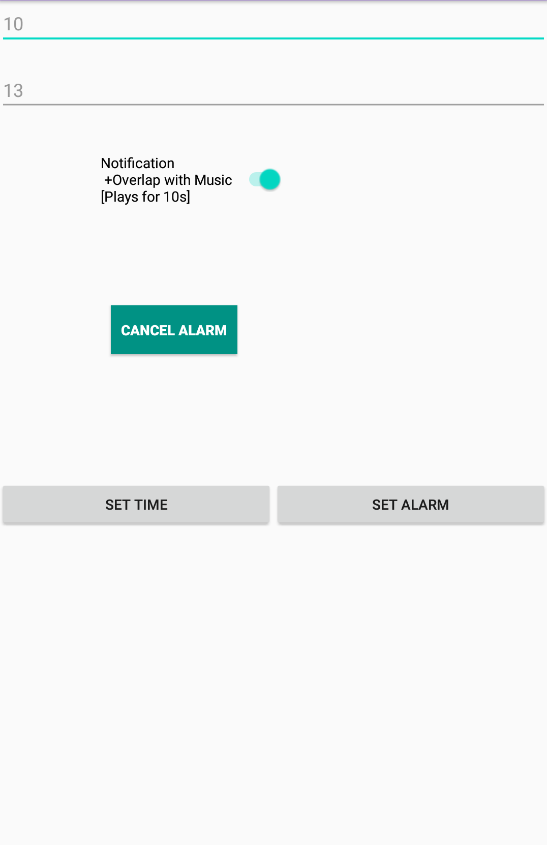
Tương tự, các thông số đo được Esp32 đẩy lên firebase với node chinhno, các tin nhắn trong cuộc trò chuyện giữa các người dùng được đẩy vào node messages.



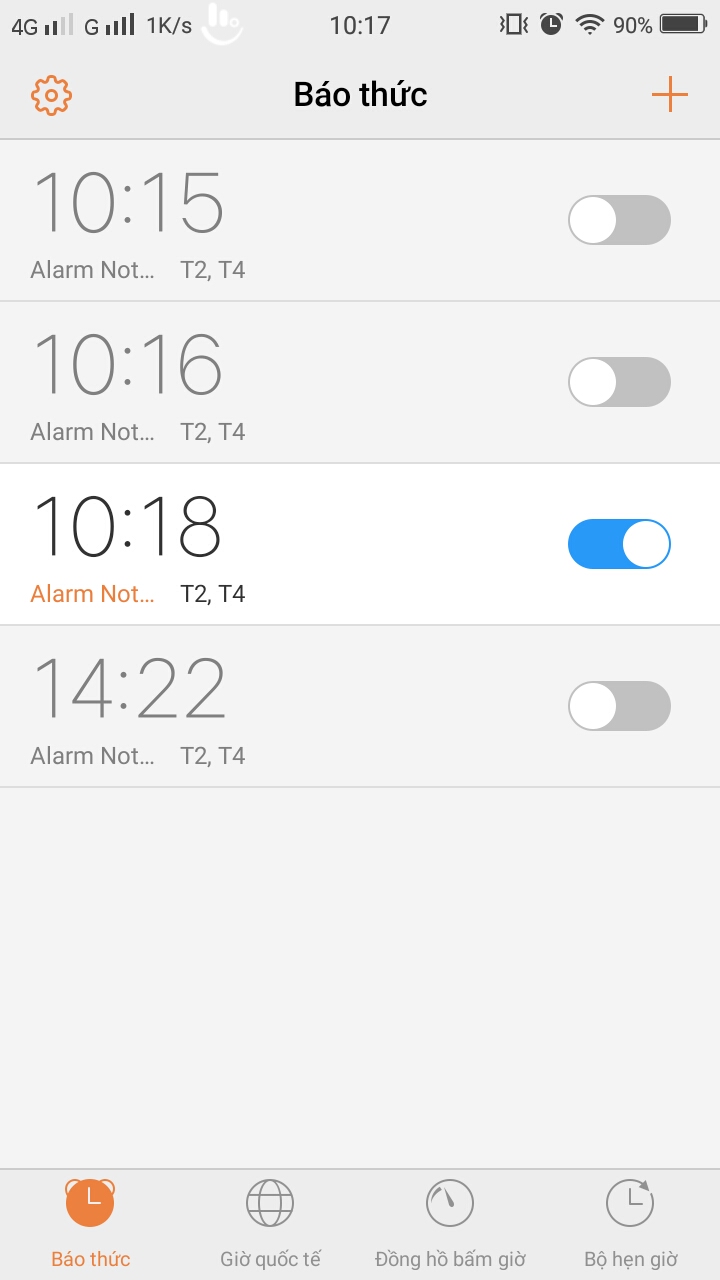
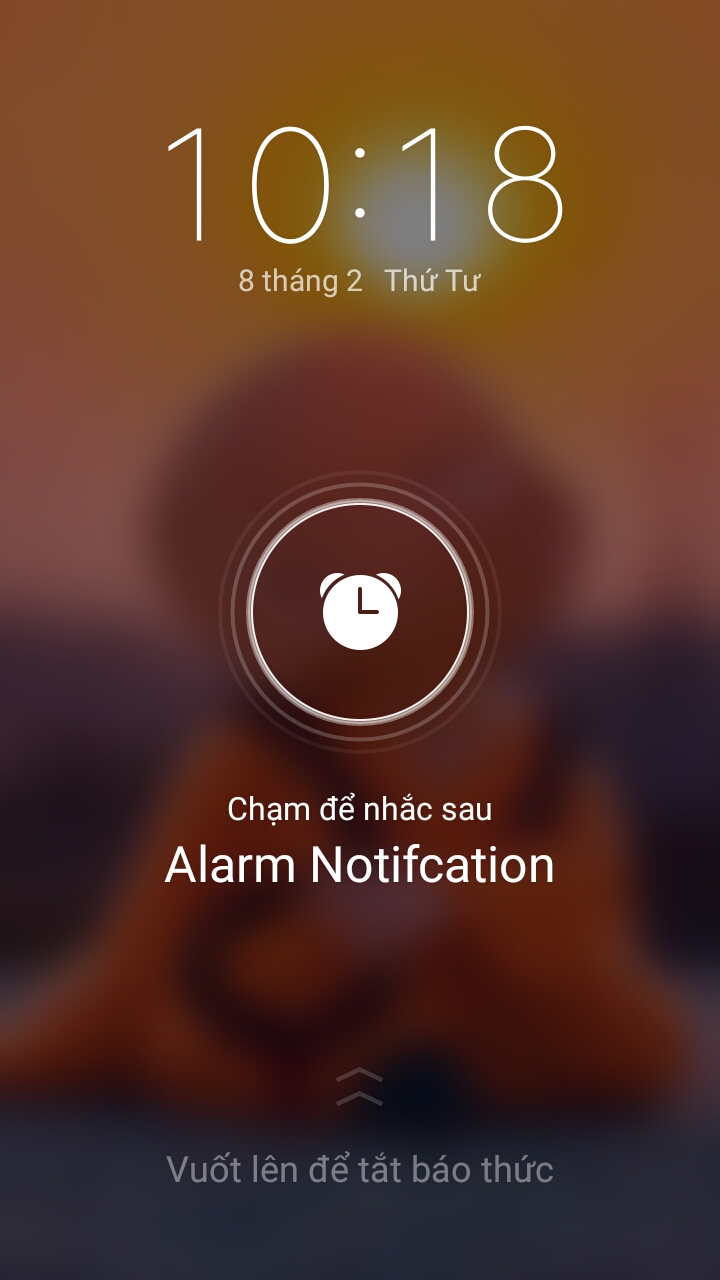
Giao diện xem lại lịch sử đo: lịch sử đo được người dùng tiến hành nhập thủ công, có hai fragment là xem lại lịch sử đo của ngày hôm nay và xem lại toàn bộ lịch sử đo.



Cài hẹn giờ uống thuốc trên thiết bị di động: người dùng có thể cài đặt nhiều báo thức.



Báo thức trên điện thoại



## 3.3. Đánh giá hệ thống

**Ưu điểm của hệ thống**

* Hiểu rõ đặc điểm, tính năng và cấu trúc của hệ thống chăm sóc bệnh nhân.
* Đo và hiển thị được kết quả như mong đợi
* Kết nối thành công với ứng dụng và Internet

**Nhược điểm của hệ thống**

* Giao diện ứng dụng còn đơn giản
* Hạn chế về chức năng của thiết bị
* Độ ổn định của thiết bị không cao
* Chưa tự động lưu được các kết quả đo

# KẾT LUẬN

☞ Hướng phát triển

* Phát triển thêm tính năng đăng ký và đăng nhập để đảm bảo bảo mật thông tin từng người dùng.
* Tạo cơ sở dữ liệu cho phép lưu trữ các thông số của từng bệnh nhân.
* Tích hợp thêm các chức năng đo nồng độ oxy, đo huyết áp,... giúp mở rộng thêm hệ thống chăm sóc bệnh nhân.
* Nghiên cứu phát triển tính năng thông báo khẩn tới người nhà qua thiết bị di động khi nhận thấy bệnh nhân có dấu hiệu nhịp tim và nhiệt độ không ổn định.

# TÀI LIỆU THAM KHẢO

[1] Tổng quan về sơ đồ chân ESP32 và ngoại vi, <https://khuenguyencreator.com/tong-quan-ve-so-do-chan-esp32-va-ngoai-vi/>

[2] MAX30102, cảm biến nhịp tim, oxi hiển thị LCD 1602, Arduino Uno R3,

<https://huynhnhattung.com/max30102-cam-bien-nhip-tim-oxy-hien-thi-lcd1602-arduino-uno-r3/>

[3] Tìm hiểu cấu tạo và cách dùng của mạch Arduino mega 2560,

<https://maysanxuattudong.com/mach-arduino-mega-2560/>

[4] Using Android Studio, <https://github.com/codepath/android_guides/wiki/Using-Android-Studio>

[5] Giao tiếp với Realtime Database Firebase sử dụng ESP32 và App, <https://khuenguyencreator.com/giao-tiep-voi-realtime-database-firebase-su-dung-esp32-va-app/>

[6] Chuẩn giao tiếp I2C là gì?,

<https://dientutuonglai.com/chuan-giao-tiep-i2c-la-gi.html>

[7] Chuẩn giao tiếp UART là gì?, <https://kysungheo.com/chuan-giao-tiep-uart/>

# PHỤ LỤC (Phần code nhúng và lập trình phần mềm)

## Code nạp vào Arduino Mega

#include <Wire.h>

#include <Arduino\_FreeRTOS.h>

#include "MAX30105.h"

#include "heartRate.h"

#include <LiquidCrystal\_I2C.h>

#include <FastLED.h>

#define LED\_PIN 2 //chân điều khiển led

#define NUM\_LEDS 8

CRGB leds[NUM\_LEDS];

LiquidCrystal\_I2C lcd(0x27,16,2);

MAX30105 particleSensor;

//const byte RATE\_SIZE = 4; //Increase this for more averaging. 4 is good.

//byte rates[RATE\_SIZE]; //Array of heart rates

//byte rateSpot = 0;

long lastBeat = 0; //Time at which the last beat occurred

float beatsPerMinute;

//int beatAvg;

void setup() {

Serial.begin(9600);

FastLED.addLeds<WS2812, LED\_PIN, GRB>(leds, NUM\_LEDS);

FastLED.setMaxPowerInVoltsAndMilliamps(5, 500);

FastLED.clear();

FastLED.show();

lcd.init();

lcd.clear();

lcd.backlight();

// Initialize sensor

if (!particleSensor.begin(Wire, I2C\_SPEED\_FAST)) {

Serial.println("chua ket noi dc cam bien ");

while (1);

}

Serial.println("dat ngon tay len cam bien");

particleSensor.setup();

particleSensor.setPulseAmplitudeRed(0x0A);

particleSensor.setPulseAmplitudeGreen(0);

xTaskCreate(

TaskRead

, "Read" // A name just for humans

, 1024 // This stack size can be checked & adjusted by reading the Stack Highwater

, NULL

, 1 // Priority, with 3 (configMAX\_PRIORITIES - 1) being the highest, and 0 being the lowest.

, NULL );

}

void TaskRead(void \*pvParameters) // This is a task.

{

(void) pvParameters;

pinMode(3,OUTPUT);

for (;;)

{

long irValue = particleSensor.getIR();

//Serial.println(irValue);

if (checkForBeat(irValue) == true && irValue >50000) {

long delta = millis() - lastBeat;

lastBeat = millis();

//delay(100);

beatsPerMinute = 60 / (delta / 1000.0);

lcd.setCursor(0,0);

lcd.print("Nhip tim: ");

lcd.setCursor(11,0);

//Serial.println(beatsPerMinute);

if(beatsPerMinute<99){

lcd.print((int)beatsPerMinute);

}

lcd.setCursor(0,1);

lcd.print("Nhiet do: ");

lcd.setCursor(11,1);

lcd.print(particleSensor.readTemperature());

if(beatsPerMinute>50)

Serial.println((int )beatsPerMinute);

Serial.print((int)particleSensor.readTemperature());

if(beatsPerMinute>50&&beatsPerMinute<85){

for(int i=0;i<8;i++){

leds[i]=CRGB(0,255,0);

FastLED.setBrightness(100);

FastLED.show();

digitalWrite(3,LOW);

//delay(100);

}

}

else{

for(int i=0;i<8;i++){

leds[i]=CRGB(255,0,0);

FastLED.setBrightness(100);

FastLED.show();

digitalWrite(3,HIGH);

}

}

}

if (irValue < 50000){

lcd.clear();

lcd.setCursor(0,0);

//Serial.print(" No finger?");

lcd.print("no finger");

delay(100);irValue = 60000;

}

//Serial.println(70);

}

}

void loop() {

/\*long irValue = particleSensor.getIR();

if (checkForBeat(irValue) == true) {

long delta = millis() - lastBeat;

lastBeat = millis();

beatsPerMinute = 60 / (delta / 1000.0);

}

lcd.setCursor(0,0);

lcd.print("Nhip tim: ");

lcd.setCursor(11,0);

if(beatsPerMinute<99){

lcd.print((int)beatsPerMinute);

}

//lcd.clear();

Serial.println(beatsPerMinute);

//lcd.setCursor(0,1);

//lcd.print("Nhiet do: ");

//lcd.setCursor(11,1);

//lcd.print(particleSensor.readTemperature());

if(beatsPerMinute>50&&beatsPerMinute<85){

for(int i=0;i<8;i++){

leds[i]=CRGB(0,255,0);

FastLED.setBrightness(100);

FastLED.show();

//delay(100);

}

}

else{

for(int i=0;i<8;i++){

leds[i]=CRGB(255,0,0);

FastLED.setBrightness(100);

FastLED.show();

}

}

if (irValue < 50000){

lcd.clear();

delay(1000);

lcd.clear();

lcd.setCursor(0,0);

Serial.print(" No finger?");

lcd.print("no finger");

delay(1000);

}

Serial.println();\*/

}

## Code nạp vào ESP32

#include <ArduinoJson.h>

#include <ArduinoJson.hpp>

#include <WiFi.h>

#include <FirebaseESP32.h>

#include <ArduinoJson.h>

#define WIFI\_SSID "302"

#define WIFI\_PASSWORD "23456789"

#define RXp2 16

#define TXp2 17

FirebaseData firebaseData;

String path = "/";

FirebaseJson json;

#define FIREBASE\_HOST "aatqka-default-rtdb.firebaseio.com"

#define FIREBASE\_AUTH "zIW9tKDUXkU0HMUhdD1Bwwo1nk3HtqO4TtY5T39m"

void setup() {

Serial.begin(115200);

Serial2.begin(9600, SERIAL\_8N1, RXp2, TXp2);

// put your setup code here, to run once:

WiFi.begin(WIFI\_SSID, WIFI\_PASSWORD);

while(WiFi.status()!=WL\_CONNECTED){

delay(500);

Serial.print(".");

}

Firebase.begin(FIREBASE\_HOST, FIREBASE\_AUTH);

Firebase.reconnectWiFi(true);

if(!Firebase.beginStream(firebaseData,path)){

Serial.println("REASON: "+firebaseData.errorReason());

Serial.println();

}

Serial.print("Connected with IP:");

Serial.println(WiFi.localIP());

Serial.println();

Firebase.setReadTimeout(firebaseData,1000\*60);

Firebase.setwriteSizeLimit(firebaseData,"tiny");

//int nhip = Serial2.readString().toInt();

//Firebase.setInt(firebaseData,"/nhiptimofthai/nhiptim",nhip);

}

void loop() {

String My\_S = Serial2.readString();

int nhiet = My\_S.substring(0,2).toInt();

int nhip = My\_S.substring(2,4).toInt();

// put your main code here, to run repeatedly:

//int nhip =Serial2.readString().toInt();

//

if(nhip!=0){

Firebase.setInt(firebaseData,"/chinhno/Nhiptim",nhip);

Firebase.setInt(firebaseData,"/chinhno/Nhietdo",nhiet);

//delay(1000);

}

}