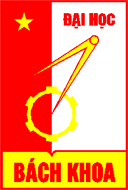
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

**VIỆN ĐIỆN TỬ - VIỄN THÔNG**



BÁO CÁO BÀI TẬP LỚN

**HỆ THỐNG NHÚNG VÀ THIẾT KẾ GIAO TIẾP NHÚNG**

**Đề tài:**

**THIẾT KẾ VÀ LẬP TRÌNH MODULE TRUYỀN VÀ NHẬN BẢN TIN THEO CHUẨN UART**

|  |  |  |
| --- | --- | --- |
| Giảng viên hướng dẫn: | **TS. Đào Việt Hùng** | |
| Học kỳ: | 20221 | |
| Mã học phần: | ET4361 | |
| Mã lớp: | 133383 | |
| Nhóm: | X | |
| **Phan Hoàng Trung** | 20186318 | Điện tử 9 – K63 |
| **Nguyễn Mạnh Cường** | 20182401 | Điện tử 11 – K63 |
| **Lê Huy Hiếu** | 20182512 | Điện tử 11 – K63 |

*Hà Nội, 7/2022*

**Lời cảm ơn**

Để có thể hoàn thành môn học “Hệ thống nhúng và thiết kế giao tiếp nhúng” này, nhóm xin gửi lời cảm ơn sâu sắc đến thầy TS. Đào Việt Hùng, trong suốt quá trình thực hiện, bọn em luôn nhận được sự giúp đỡ, chỉ dạy, định hướng đi một cách khoa học, tận tình của thầy.

Trong quá trình thực hiện, chúng em đã cố gắng nỗ lực, tuy nhiên do thời gian gấp rút và trình độ nghiên cứu còn hạn chế nên vẫn cần sự đánh giá, nhận xét của thầy để đề tài được đầy đủ và hoàn chỉnh hơn.

Em xin chân thành cảm ơn.

**Tóm tắt nội dung yêu cầu**

*Thiết kế và lập trình mô-đun truyền và nhận bản tin theo chuẩn UART, yêu cầu:*

* Sử dụng vi điều khiển, lập trình bằng ngôn ngữ C hoặc tương đương, không giới hạn hãng và loại chip sử dụng.
* Mạch điện hàn trên PCB tự thiết kế hoặc sử dụng kit có bán sẵn, tuyệt đối không sử dụng breadboard và cắm dây.
* Gửi/nhận được bản tin “Hello World 2022“ và có thể thay đổi bản tin dễ dàng về nội dung và độ dài (đến 100 ký tự) khi được yêu cầu.
* Có thể thay đổi được tốc độ truyền thông(b/s) một cách linh hoạt. Tốc độ truyền và nhận được định nghĩa tại đầu mỗi chương trình bằng lệnh #define.
* Có thể đổi chân IC tuỳ ý bằng cách thay đổi code. Các chân chức năng được định nghĩa tại đầu mỗi chương trình bằng lệnh #define.

*Ngoài ra, nhóm thiết kế thêm:*

* Truyền và nhận nhiều trường bản tin: Nhiệt độ, độ ẩm,...

**PHÂN CÔNG CÔNG VIỆC**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **CÔNG VIỆC** | **THỰC HIỆN BỞI** | **TIẾN ĐỘ** |
| 1 | Tìm hiểu lý thuyết về UART, STM32 | Trung, Hiếu, Cường | 100% |
| 2 | Tìm hiểu cách cài đặt các phần mềm phục vụ lập trình: KeilC, CubeMX, Hercules | Cường | 100% |
| 3 | Cấu hình các chân trên CubeMX | Trung | 100% |
| 4 | Khai báo thư viện, define các chân, hàm khởi tạo UART, hàm truyền/nhận 1 byte | Hiếu | 100% |
| 5 | Thực hiện viết hàm truyền chuỗi dữ liệu | Trung | 100% |
| 6 | Thực hiện viết hàm truyền nhiều trường bản tin | Cường | 100% |
| 7 | Triển khai phần cứng, hoàn thiện mạch cứng | Hiếu + Cường | 100% |
| 8 | Báo cáo + slide | Trung | 100% |

**MỤC LỤC**

[CHƯƠNG 1. LÝ THUYẾT VỀ UART 1](#_Toc111645503)

[1.1 Giao thức UART là gì? 1](#_Toc111645504)

[1.2 Cách hoạt động của giao thức UART 1](#_Toc111645505)

[1.3 Các bước truyền UART 2](#_Toc111645506)

[1.4 Ưu điểm và nhược điểm của UART 4](#_Toc111645507)

[CHƯƠNG 2. LÝ THUYẾT VỀ STM32 4](#_Toc111645508)

[2.1 STM32 là gì? 5](#_Toc111645509)

[2.2 Cấu hình chi tiết của Kit Phát Triển STM32F103C8T6 BluePill 5](#_Toc111645510)

[CHƯƠNG 3. TRIỂN KHAI HỆ THỐNG 7](#_Toc111645511)

[3.1 Cấu hình chân Vi điều khiển trên CUBE MX 7](#_Toc111645512)

[3.2 Download và Cài đặt STM32 CubeMX 7](#_Toc111645513)

[3.3 Thực hiện code trên KeilC v5 8](#_Toc111645514)

[CHƯƠNG 4. TRIỂN KHAI PHẦN CỨNG 12](#_Toc111645515)

[4.1 Phần cứng 12](#_Toc111645516)

[4.2 Thực nghiệm 12](#_Toc111645517)

[4.2.1 Gửi và nhận 1 kí tự 12](#_Toc111645518)

[4.2.2 Gửi và nhận 1 chuỗi 13](#_Toc111645519)

[4.2.3 Gửi và nhận 1 trường dữ liệu 14](#_Toc111645520)

[4.2.4 Thay đổi các chân và tốc độ baudrate 16](#_Toc111645521)

[TÀI LIỆU THAM KHẢO 18](#_Toc111645522)

**DANH MỤC HÌNH VẼ**

[Hình 1.1 UART Communication 1](#_Toc111645523)

[Hình 1.2 Định dạng gói tin UART 2](#_Toc111645524)

[Hình 1.3 Truyền nhận dữ liệu song song từ bus dữ liệu 3](#_Toc111645525)

[Hình 1.4 Truyền bit start, chẵn lẻ, bit dừng 3](#_Toc111645526)

[Hình 1.5 Gửi gói thông tin 3](#_Toc111645527)

[Hình 1.6 Loại bỏ bit start, chẵn lẻ và stop 3](#_Toc111645528)

[Hình 1.7 Chuyển đổi dữ liệu 4](#_Toc111645529)

[Hình 2.1 Cấu hình STM32F103C8T6 5](#_Toc111645530)

[Hình 3.1 Cấu hình cho Vi điều khiển 7](#_Toc111645531)

[Hình 3.2 Cấu hình chân trên CUBE MX 8](#_Toc111645532)

[Hình 3.3 Generate code sang KeilC v5 8](#_Toc111645533)

[Hình 3.4 Khai báo thư viện và định nghĩa chân 9](#_Toc111645534)

[Hình 3.5 Khởi tạo UART 9](#_Toc111645535)

[Hình 3.6 Hàm truyền 1 byte 9](#_Toc111645536)

[Hình 3.7 Hàm nhận 1 byte 10](#_Toc111645537)

[Hình 3.8 Truyền chuỗi dữ liệu 10](#_Toc111645538)

[Hình 3.9 Hàm truyền chuỗi dữ liệu 11](#_Toc111645539)

[Hình 4.1 Sơ đồ kết nối 12](#_Toc111645540)

[Hình 4.2 Mạch cứng 12](#_Toc111645541)

[Hình 4.3 Kết quả mô phỏng truyền 1 ký tự 13](#_Toc111645542)

[Hình 4.4 Kết quả thực nghiệm 13](#_Toc111645543)

[Hình 4.5 Kết quả mô phỏng 13](#_Toc111645544)

[Hình 4.6 Kết quả thực nghiệm 14](#_Toc111645545)

[Hình 4.7 Kết quả mô phỏng 14](#_Toc111645546)

[Hình 4.8 Kết quả thực nghiệm 15](#_Toc111645547)

[Hình 4.9 Kết quả thực nghiệm 15](#_Toc111645548)

[Hình 4.10 Kết quả thực nghiệm 15](#_Toc111645549)

[Hình 4.11 Mạch khi thay đổi các chân 16](#_Toc111645550)

[Hình 4.12 Truyền 1 ký tự khi thay đổi 16](#_Toc111645551)

[Hình 4.13 Truyền 1 chuỗi khi thay đổi 16](#_Toc111645552)

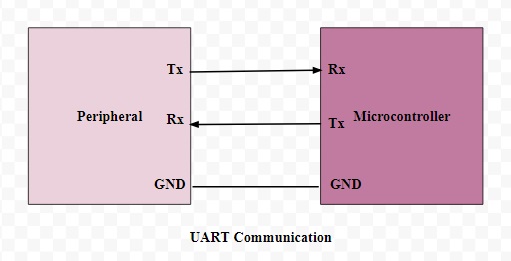
[Hình 4.14 Truyền 1 trường bản tin khi thay đổi 17](file:///C:\Users\Admin\Desktop\NHÚNG.docx#_Toc111645553)

# LÝ THUYẾT VỀ UART

## Giao thức UART là gì?

UART tiếng anh là Universal Asynchronous Reciver/Transmister  một chuẩn giao tiếp không đồng bộ cho MCU và các thiết bị ngoại vi.

Chuẩn UART là chuẩn giao tiếp điểm và điểm, nghĩa là trong mạng chỉ có hai thiết bị đóng vai trò là transmister hoặc reciver.



Hình 1.1 UART Communication

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

* Transmitter (Tx)
* Receiver (Rx)

Các hệ thống nhúng, vi điều khiển và máy tính hầu hết sử dụng UART như một dạng giao thức giao tiếp phần cứng giữa thiết bị và thiết bị. Trong số các giao thức truyền thông hiện có, UART chỉ sử dụng hai dây cho bên truyền và bên nhận.

## Cách hoạt động của giao thức UART

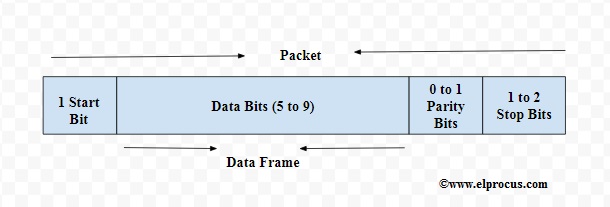
UART là giao thức truyền thông không đồng bộ, nghĩa là không có xung Clock, các thiết bị có thể hiểu được nhau nếu các Setting giống nhau

UART là truyền thông song công(Full duplex) nghĩa là tại một thời điểm có thể truyền và nhận đồng thời.

Trong đó quan trọng nhất là Baund rate (tốc độ Baund)  là khoảng thời gian dành cho 1 bit được truyền. Phải được cài đặt giống nhau ở gửi và nhận.

Sau đó là định dạng gói tin.

Định dạng gói tin như sau:



Hình 1.2 Định dạng gói tin UART

**Start – Bit**

Start-bit còn được gọi là bit đồng bộ hóa được đặt trước dữ liệu thực tế. Nói chung, một đường truyền dữ liệu không hoạt động được điều khiển ở mức điện áp cao. Để bắt đầu truyền dữ liệu, truyền UART kéo đường dữ liệu từ mức điện áp cao (1) xuống mức điện áp thấp (0). UART thu được thông báo sự chuyển đổi này từ mức cao sang mức thấp qua đường dữ liệu cũng như bắt đầu hiểu dữ liệu thực. Nói chung, chỉ có một start-bit.

**Stop – Bit**

Bit dừng được đặt ở phần cuối của gói dữ liệu. Thông thường, bit này dài 2 bit nhưng thường chỉ sử dụng 1 bit. Để dừng sóng, UART giữ đường dữ liệu ở mức điện áp cao.

**Partity Bit**

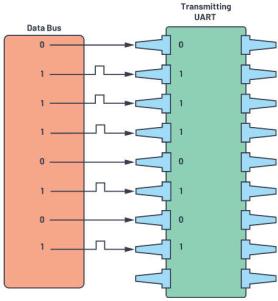
Bit chẵn lẻ cho phép người nhận đảm bảo liệu dữ liệu được thu thập có đúng hay không. Đây là một hệ thống kiểm tra lỗi cấp thấp & bit chẵn lẻ có sẵn trong hai phạm vi như Chẵn lẻ – chẵn lẻ cũng như Chẵn lẻ – lẻ. Trên thực tế, bit này không được sử dụng rộng rãi nên không bắt buộc.

**Data frame**

Các bit dữ liệu bao gồm dữ liệu thực được truyền từ người gửi đến người nhận. Độ dài khung dữ liệu có thể nằm trong khoảng 5 & 8. Nếu bit chẵn lẻ không được sử dụng thì chiều dài khung dữ liệu có thể dài 9 bit. Nói chung, LSB của dữ liệu được truyền trước tiên sau đó nó rất hữu ích cho việc truyền.

## Các bước truyền UART

1. UART truyền nhận dữ liệu song song từ bus dữ liệu.



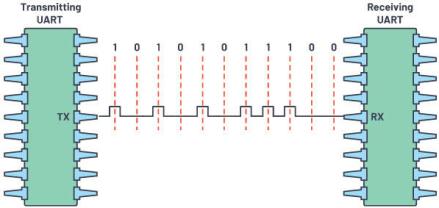
Hình 1.3 Truyền nhận dữ liệu song song từ bus dữ liệu

2. UART truyền thêm bit start, bit chẵn lẻ và bit dừng vào khung dữ liệu.



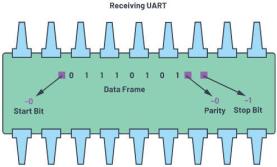
Hình 1.4 Truyền bit start, chẵn lẻ, bit dừng

3. Toàn bộ gói được gửi nối tiếp từ UART truyền đến UART nhận. UART nhận lấy mẫu đường dữ liệu ở tốc độ truyền được định cấu hình trước.



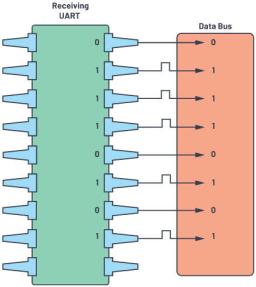
Hình 1.5 Gửi gói thông tin

4. UART nhận loại bỏ bit start, bit chẵn lẻ và bit stop khỏi khung dữ liệu.



Hình 1.6 Loại bỏ bit start, chẵn lẻ và stop

5. UART nhận chuyển đổi dữ liệu nối tiếp trở lại thành song song và chuyển nó đến bus dữ liệu ở đầu nhận.



Hình 1.7 Chuyển đổi dữ liệu

## Ưu điểm và nhược điểm của UART

Không có giao thức truyền thông nào là hoàn hảo, nhưng UART thực hiện khá tốt những gì chúng làm. Dưới đây là một số ưu và nhược điểm để giúp bạn quyết định xem chúng có phù hợp với nhu cầu của bạn hay không.

Ưu điểm

* Chỉ sử dụng hai dây để truyền dữ liệu
* Không cần tín hiệu đồng hồ
* Có một bit chẵn lẻ để cho phép kiểm tra lỗi
* Cấu trúc của gói dữ liệu có thể được thay đổi miễn là cả hai bên được thiết lập cho nó
* Phương pháp truyền đơn giản, giá thành thấp

Nhược điểm

* Kích thước của khung dữ liệu được giới hạn tối đa là 9 bit
* Không phù hợp với các hệ thống đòi hỏi nhiều thiết bị chủ và tớ
* Tốc độ truyền của mỗi UART phải nằm trong khoảng 10%

# LÝ THUYẾT VỀ STM32

## STM32 là gì?

STM32 là dòng chip 32bit của hãng [**STMicrochip**](https://www.st.com/content/st_com/en.html)sử dụng công nghệ lõi **ARM Cortex**mạnh mẽ, hiệu năng tốt nhưng vẫn giữ được giá thành rẻ. Phù hợp với đa số các công ty hiện nay

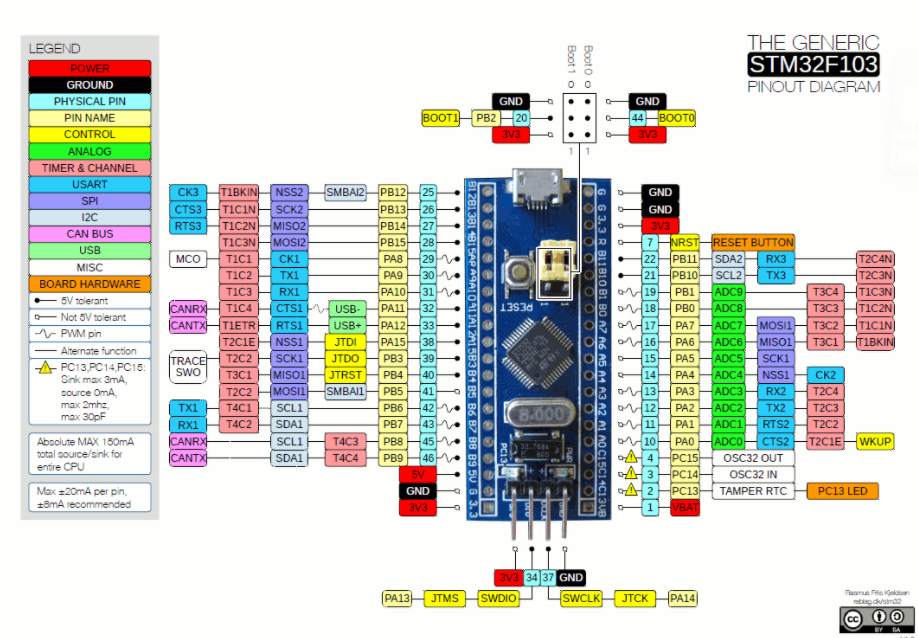
Các lý do nên chọn STM32 đó là:

* Tốc độ xử lý cao, ngoại vi hỗ trợ rất nhiều, dòng chip phân khúc thâp là STM32F0x cũng có thể hoạt động lên tới 48Mhz, 64kB Flash, 16kB RAM, 8 bộ Timer 16 bit, 1 bộ Timer 32 bit, 10 bộ ADC 12 bit, 8 bộ USART, 2 bộ SPI, 2 bộ I2C.
* Giá thành rẻ nhưng hiệu quả đem lại lớn.
* Học lập trình STM32 rất dễ dàng do cộng đồng hỗ trợ nhiều.
* Dễ xin việc do các công ty vừa và nhỏ sử dụng STM32 trong các ứng dụng rất nhiều
* Công cụ lập trình đều Free và đầy đủ tài liệu hỗ trợ

Nếu đem STM32 ra so sánh với các dòng chip khác sẽ vẫn có nhiều khuyết điểm. Thế nhưng mặt bằng chung STM32 vẫn là lựa chọn tối ưu khi học lập trình.

## Cấu hình chi tiết của Kit Phát Triển STM32F103C8T6 BluePill

KIT Blue Pill sử dụng chip STM32F103c8t6 là một dòng chip phổ thông nhất của STM, có đầy đủ các ngoại vi cơ bản, bộ nhớ vừa đủ và giá thành rẻ, rất phù hợp với những người mới bắt đầu.



Hình 2.1 Cấu hình STM32F103C8T6

**Thông số kỹ thuật:**

* Vi điều khiển: STM32F103C8T6.
* Điện áp cấp 5VDC qua cổng Micro USB sẽ được chuyển đổi thành 3.3VDC qua IC nguồn và cấp cho Vi điều khiển chính.
* Tích hợp sẵn thạch anh 8Mhz.
* Tích hợp sẵn thạnh anh 32Khz cho các ứng dụng RTC.
* Ra chân đầy đủ tất cả các GPIO và giao tiếp: CAN, I2C, SPI, UART, USB,…
* Tích hợp Led trạng thái nguồn, Led PC13, Nút Reset.
* Kích thước: 53.34 x 15.24mm
* Sử dụng với các mạch nạp:
  + ST-Link Mini
  + J-link
  + USB TO COM
* Kết nối chân khi nạp bằng ST-Link Mini
* Nạp theo chuẩn SWD
  + TCK — SWCLK
  + TMS — SWDIO
  + GND — GND
  + 3.3V — 3.3V

# TRIỂN KHAI HỆ THỐNG

## Cấu hình chân Vi điều khiển trên CUBE MX

Phần mềm CubeMX là một công cụ của ST giúp người lập trình có thể sinh code C thông qua giao diện tương tác rất dễ sử dụng.

Các tính năng chủ yếu như:

* Tự động giải quyết các xung đột về Pin out.
* Hỗ trợ tất cả các dòng chip STM32
* Cầu hình Clock thông qua Clock Tree rất dễ sử dụng
* Ước tính năng lượng tiêu thụ
* Hỗ trợ sinh code cho các IDE phổ biến như Keil C, IAR, GCC

## Download và Cài đặt STM32 CubeMX

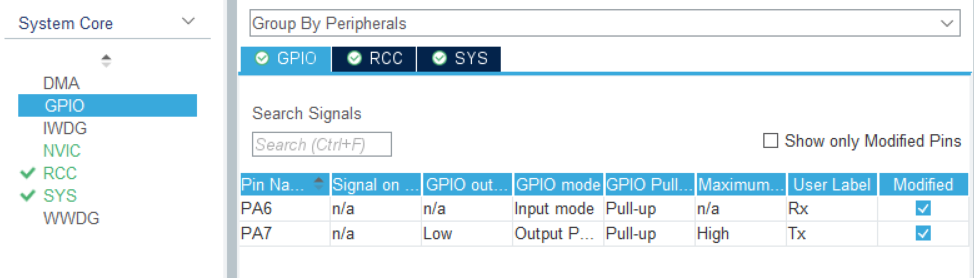
CubeMx là 1 tools miễn phí của ST các bạn vào link Sau để download:

Các Bạn download tại link:

<https://www.st.com/en/development-tools/stm32cubemx.html>

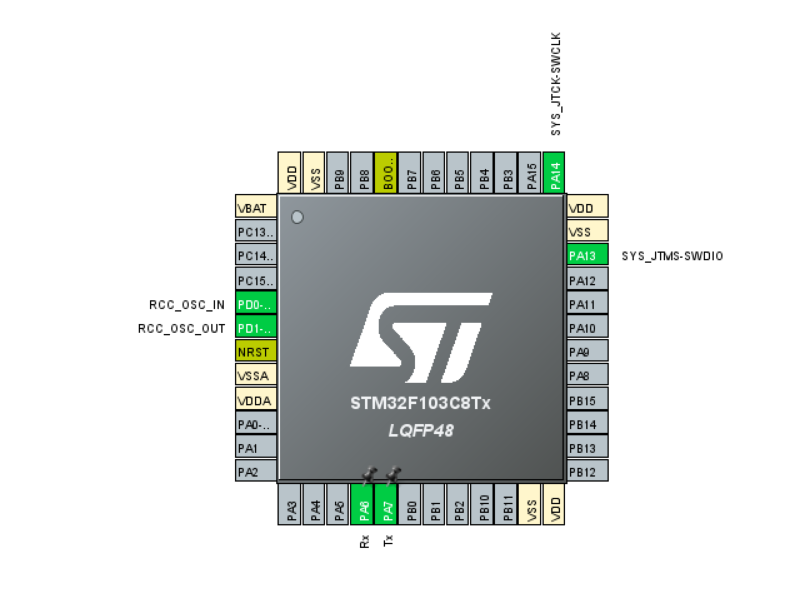
***Thực hiện cấu hình cho Vi điều khiển:***

* System core:



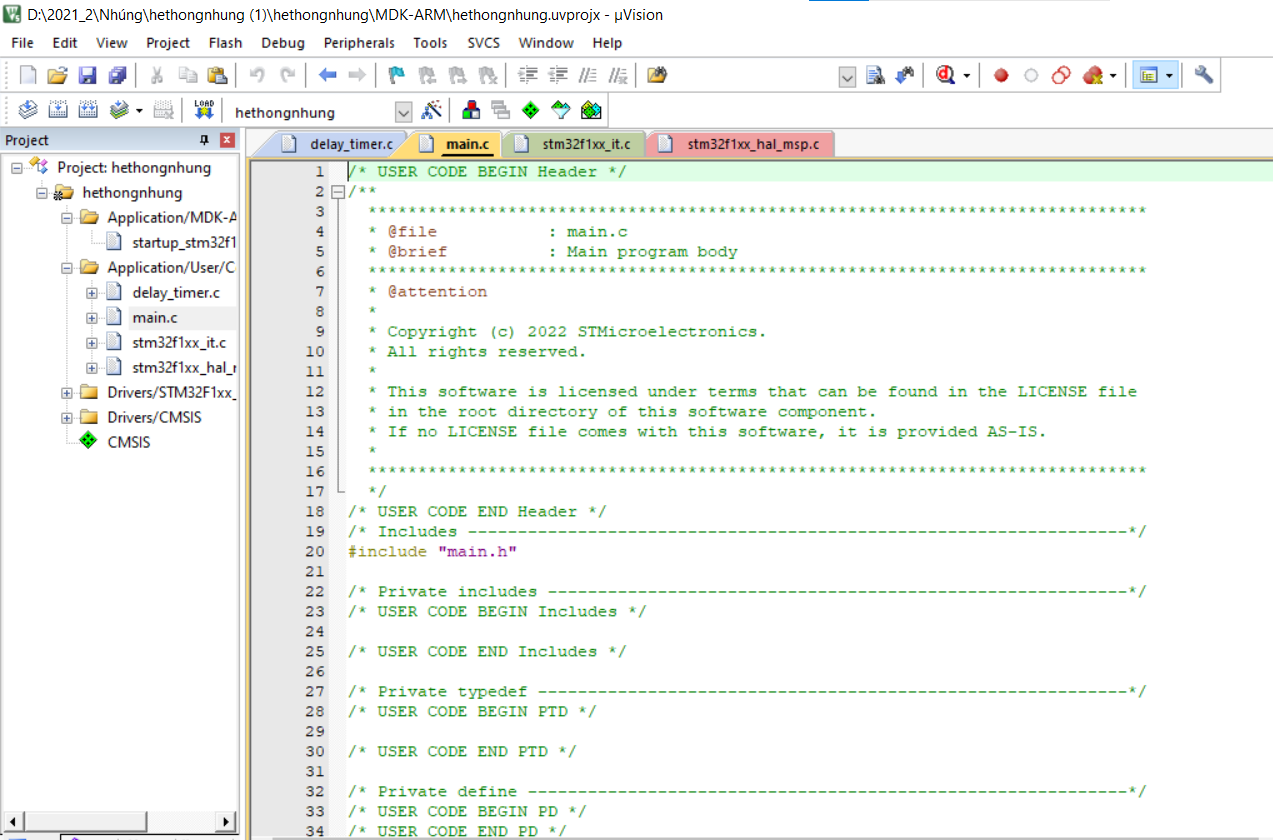
Hình 3.1 Cấu hình cho Vi điều khiển

* Sử dụng chân PA5 và PA7 tương ứng là chân Rx và Tx



Hình 3.2 Cấu hình chân trên CUBE MX

* Sau khi cấu hình xong, thực hiện Generate code sang phần mềm KeilC



Hình 3.3 Generate code sang KeilC v5

## Thực hiện code trên KeilC v5

***Khai báo các thư viện và định nghĩa các chân***

Text

Description automatically generated

Hình 3.4 Khai báo thư viện và định nghĩa chân

* “main.h”: phần viết code chính
* “delay\_timer.h”: chứa các hàm delay theo đơn vi millisecond và microsecond sử dụng bộ Timer có sẵn trong vi điều khiển
* “stdio.h” và “string.h”: thư viện chuẩn của C
* Baudrate: tốc độ baudrate sử dụng trong UART
* Rx: chân nhận dữ liệu
* Tx: chân truyền dữ liệu

Lưu ý: 2 chân tx, rx sử dụng trên PORT A

Có thể thay đổi chân rx, tx và Baudrate ở đây

***Tạo các hàm chính***

* Hàm khởi tạo Uart

Text

Description automatically generated

Hình 3.5 Khởi tạo UART

* Truyền vào tốc độ baudrate
* Tính thời gian 1 chu kỳ
* Đặt các chân Tx, Rx ở mức 1

***Hàm truyền 1 byte***

Text

Description automatically generated

Hình 3.6 Hàm truyền 1 byte

* Truyền vào hàm một byte dữ liệu cần gửi
* Đặt chân Tx xuống mức 0
* Delay 1 chu kỳ
* Gửi từng bit dữ liệu mỗi lần cách nhau 1 chu kỳ
* Sau khi gửi 8 bit dữ liệu, kéo chân Tx lên mức 1 và Delay 1 chu kỳ
* Kết thúc khung truyền byte dữ liệu

***Hàm nhận 1 byte***

Text

Description automatically generated

Hình 3.7 Hàm nhận 1 byte

* Delay 1.5 chu kỳ
* Nhận từng bit dữ liệu mỗi lần cách nhau 1 chu kỳ
* Sau khi nhận 8 bit dữ liệu nhận thêm 1 bit stop
* Kiểm tra bit stop nếu là 0 trả về byte vừa nhận, nếu là 1 huỷ bỏ byte lỗi
* Kết thúc khung nhận 1 byte dữ liệu
* Hàm truyền chuỗi dữ liệu

Text

Description automatically generated

Hình 3.8 Truyền chuỗi dữ liệu

* Tạo 1 biến kiểm tra kết thúc chuỗi
* Truyền từng byte dữ liệu dung hàm Write\_byte
* Sau khi phát hiện kí tự cuối là “Enter” thì kết thúc truyền
* Hàm truyền chuỗi dữ liệu

Graphical user interface, text

Description automatically generated

Hình 3.9 Hàm truyền chuỗi dữ liệu

* Tạo 1 biến kiểm tra kết thúc chuỗi (check)
* Xoá dữ liệu trong biến nhận ‘dataRx’ và các trường nhiệt độ, độ ẩm, ánh sáng
* Nhận lần lượt từng byte dữ liệu vào ‘dataRx’ kết thúc khi kí tự cuối là “Enter” hoặc quá 100 kí tự.
* Kiểm tra chuỗi xem có phải chuỗi thường hay kiểu trường dữ liệu.
* Nếu kiểu thường thì gửi lại dữ liệu qua hàm Send().
* Nếu kiểu trường dữ liệu thì thực hiện tách dữ liệu theo từng trường rồi gửi lại qua hàm Send().

# TRIỂN KHAI PHẦN CỨNG

## Phần cứng

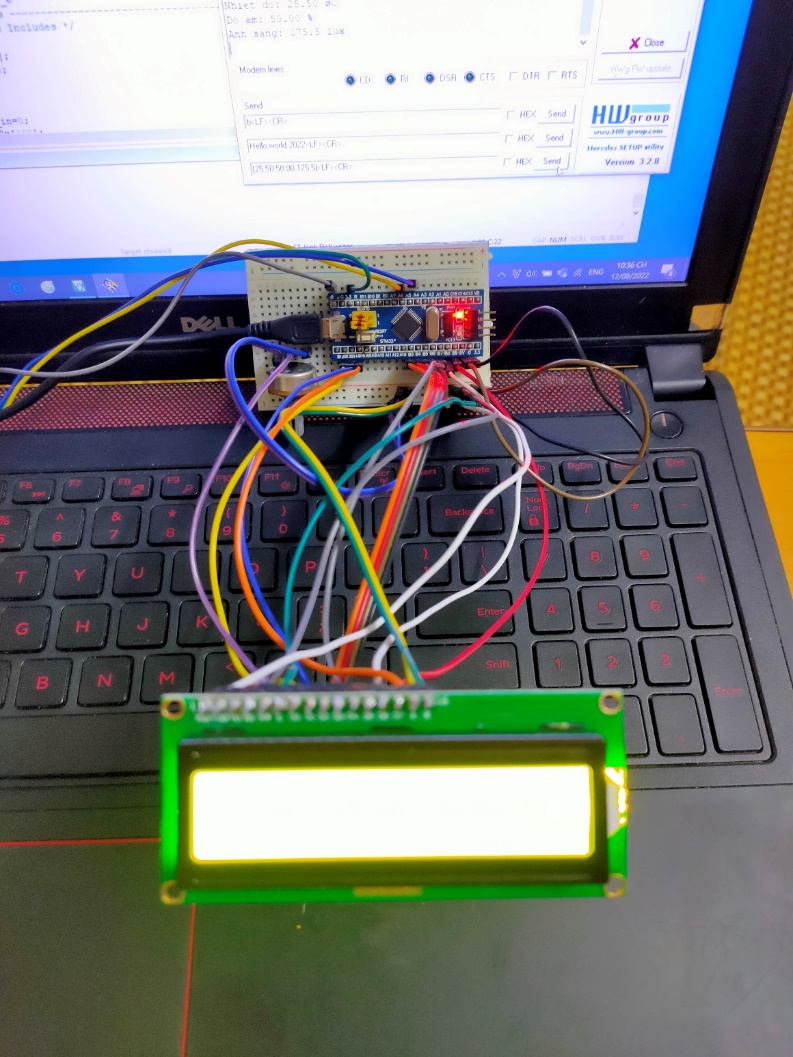
*Sơ đồ kết nối*

A close-up of a circuit board

Description automatically generated with low confidence

Hình 4.1 Sơ đồ kết nối

*Mạch cứng đã triển khai*



Hình 4.2 Mạch cứng

## Thực nghiệm

### Gửi và nhận 1 kí tự

Gửi 1 kí tự từ máy tính thông qua cổng USB đến vi điều khiển. Sau đó vi điều khiển gửi lại đúng kí tự đó lại cho máy tính.

A picture containing shape

Description automatically generated

Hình 4.3 Kết quả mô phỏng truyền 1 ký tự

Lưu ý:

* Kí tự màu hồng là máy tính gửi cho vi điều khiển
* Kí tự màu đen là máy tính nhận từ vi điều khiển

***Kết quả hiển thị***



Hình 4.4 Kết quả thực nghiệm

### Gửi và nhận 1 chuỗi

Graphical user interface, application

Description automatically generated

Hình 4.5 Kết quả mô phỏng

Lưu ý:

* Kí tự màu hồng là máy tính gửi cho vi điều khiển
* Kí tự màu đen là máy tính nhận từ vi điều khiển
* Chuỗi kết thúc khi phát hiện kí tự ‘Enter’ (<CR>)

Kết quả thực hiện:

A picture containing text, monitor, indoor, electronics

Description automatically generated

Hình 4.6 Kết quả thực nghiệm

### Gửi và nhận 1 trường dữ liệu

Graphical user interface

Description automatically generated with low confidence

Hình 4.7 Kết quả mô phỏng

Lưu ý:

* Kí tự màu hồng là máy tính gửi cho vi điều khiển
* Kí tự màu đen là máy tính nhận từ vi điều khiển
* Chuỗi kết thúc khi phát hiện kí tự ‘Enter’ (<CR>)

***Kết quả thực hiện***

A picture containing text, monitor, electronics

Description automatically generated

Hình 4.8 Kết quả thực nghiệm

A picture containing text, monitor, indoor, electronics

Description automatically generated

Hình 4.9 Kết quả thực nghiệm

A picture containing text, monitor, electronics, close

Description automatically generated

Hình 4.10 Kết quả thực nghiệm

### Thay đổi các chân và tốc độ baudrate

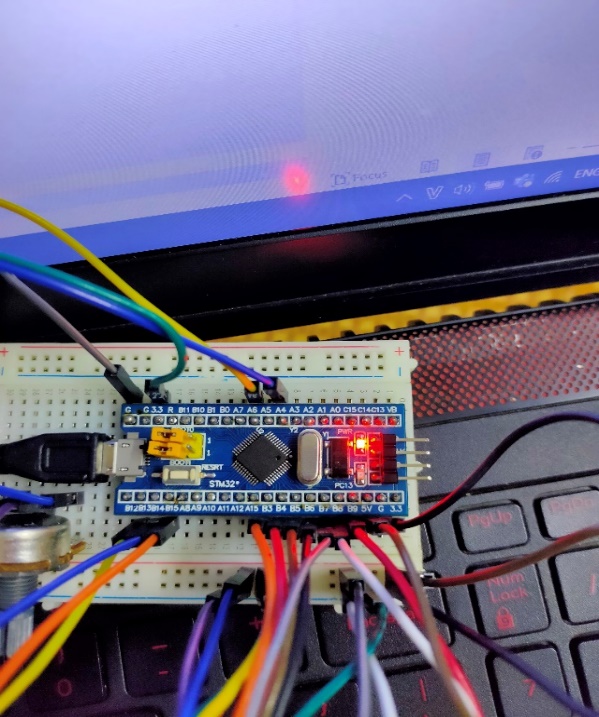
* Thay đổi chân 5 và chân 7 sang chân 4 và chân 6
* Tốc độ Baudrate từ 9600->14400

Text

Description automatically generated with medium confidence Graphical user interface, text

Description automatically generated

Mạch cứng cũng sẽ thay đổi các chân cắm:



Hình 4.11 Mạch khi thay đổi các chân

**KẾT QUẢ**

*Truyền và nhận 1 ký tự*



Hình 4.12 Truyền 1 ký tự khi thay đổi

*Truyền và nhận chuỗi ký tự*

A picture containing text, monitor, indoor, electronics

Description automatically generated

Hình 4.13 Truyền 1 chuỗi khi thay đổi

*Truyền và nhận nhiều trường bản tin*

A picture containing text, monitor, electronics

Description automatically generatedA picture containing text, monitor, indoor, electronics

Description automatically generatedA picture containing text, monitor, electronics, close

Description automatically generated

Hình . Truyền 1 trường bản tin khi thay đổi

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | [UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter | Analog Devices](https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html) |
| [2] | [Bài 1 : Giới Thiệu Về Stm32 Là Gì ? Tác Dụng Của Nó Như Thế Nào? (hit.edu.vn)](https://hit.edu.vn/stm32-la-gi/) |
| [3] | [Bài 11: Lập trình STM32 với Giao thức UART trên Cube MX (khuenguyencreator.com)](https://khuenguyencreator.com/lap-trinh-stm32-voi-giao-thuc-uart/) |

[4] [STM32F1: UART với STM32 (deviot.vn)](https://deviot.vn/tutorials/stm32f1.23165131/uart-voi-stm32.99038892)