BỘ CÔNG THƯƠNG

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THÀNH PHỐ HỒ CHÍ MINH

KHOA CÔNG NGHỆ ĐIỆN TỬ

NGUYỄN TRỌNG HIẾU

TRẦN BẢO MINH LONG

ĐÀO PHAN GIA HUY

GIAO TIẾP ĐIỀU KHIỂN MODULE BLUETOOTH HC-05 VỚI PIC16F887 BẰNG UART

Chuyên ngành: Kỹ thuật máy tính

Mã chuyên ngành: 7480108

THÀNH PHỐ HỒ CHÍ MINH NĂM 2024

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

MỞ ĐẦU

1. Đặt vấn đề

Bối cảnh

Trong thời đại IoT phát triển mạnh mẽ như hiện nay, việc kết nối không dây giữa các thiết bị điện tử để phục vụ đời sống ngày càng trở nên phổ biến. Bluetooth dần trở thành phương tiện truyền thông tiện lợi, ổn định và phù hợp với nhiều ứng dụng và người dùng trong đời sống, đặc biệt là trong tự động hóa và hệ thống nhúng. Bluetooth HC-05 là module khá dễ sử dụng và thường được tích hợp với các vi điều khiển như PIC16F887, việc tích hợp HC-05 vào hệ thống điều khiển và giao tiếp UART cho phép truyền dữ liệu giữa các thiết bị, mở ra nhiều ứng dụng thực tiễn trong tự động hóa, điều khiển từ xa, và các thiết bị IoT.

Lý do chọn đề tài

Đề tài này của nhóm giúp hiểu rõ hơn về giao thức UART và cách tích hợp Bluetooth vào các hệ thống ứng dụng thực tế. Nắm vững cách giao tiếp giữa UART và Bluetooth HC-05 mở ra nhiều ứng dụng, từ điều khiển thiết bị từ xa đến thu thập dữ liệu, đáp ứng nhu cầu phát triển hệ thống nhúng hiện đại, thông minh và tiện lợi, phù hợp với xu thế phát triển công nghệ.

Tính cấp thiết

Nhu cầu kiểm soát và giám sát từ xa qua smartphone hoặc các thiết bị thông minh ngày càng cao, thúc đẩy việc nghiên cứu các giải pháp giao tiếp không dây. Việc sử dụng HC-05 trong giao tiếp với PIC16F887 thông qua UART là cơ sở để xây dựng các hệ thống nhúng hiện đại, phục vụ nhiều lĩnh vực như y tế, nông nghiệp thông minh, và nhà thông minh. Cùng với chi phí thực hiện không quá cao, với các thành phần phần cứng dễ dàng tiếp cận, phù hợp với điều kiện học tập và nghiên cứu tại trường

2. Mục tiêu của đề tài

- Nghiên cứu nguyên lý hoạt động của module HC-05 và giao tiếp UART trên PIC16F887.

- Lập trình và kiểm tra khả năng truyền/nhận dữ liệu qua UART giữa HC-05 và PIC16F887.

- Xây dựng ứng dụng thử nghiệm, chẳng hạn như điều khiển từ xa hoặc hiển thị dữ liệu trên máy tính hoặc smartphone.

- Đánh giá hiệu suất hệ thống và khả năng mở rộng trong các ứng dụng thực tế.

3. Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu

- Vi điều khiển PIC16F887: trung tâm điều khiển, thực hiện các lệnh và xử lý dữ liệu.

Module Bluetooth HC-05: Là thiết bị giao tiếp không dây, có chức năng kết nối các thiết bị với nhau qua sóng Bluetooth.

- Giao thức UART: Là giao thức truyền thông nối tiếp, được sử dụng để trao đổi dữ liệu giữa vi điều khiển và module Bluetooth.

- Phần mềm: Các phần mềm lập trình vi điều khiển (ví dụ: MPLAB X IDE), phần mềm giao tiếp với module Bluetooth (ví dụ: Tera Term), và có thể có thêm các phần mềm khác để xử lý dữ liệu (nếu cần).

Phạm vi nghiên cứu

- Giới hạn trong phạm vi giao tiếp giữa PIC16F887 và module HC-05 qua UART, với không gian ứng dụng nhỏ gọn phù hợp cho các hệ thống nhúng cỡ nhỏ.

- Phạm vi ảnh hưởng chủ yếu là trong môi trường phòng thí nghiệm.

4. Ý nghĩa thực tiễn của đề tài

Đề tài có tiềm năng ứng dụng cao trong các hệ thống IoT, tự động hóa và điều khiển từ xa. Module Bluetooth HC-05 tích hợp dễ dàng với các vi điều khiển như PIC16F887, phù hợp để xây dựng các thiết bị điều khiển không dây, thu thập dữ liệu hoặc giám sát.

Đề tài giải quyết nhu cầu truyền dữ liệu không dây một cách đơn giản và hiệu quả, giúp các kỹ sư và sinh viên hiểu sâu hơn về giao thức UART và cách ứng dụng Bluetooth

trong thực tế. Đối tượng hưởng lợi bao gồm những người nghiên cứu và phát triển hệ thống nhúng, giám sát thiết bị từ xa và điều khiển thiết bị qua mạng Bluetooth.

Đề tài có tính hiệu quả cao nhờ khả năng truyền dữ liệu ổn định, dễ thiết lập và chi phí thấp. Việc tích hợp Bluetooth giúp giảm thiểu dây nối, tăng tính linh hoạt và tiện lợi, góp phần vào sự phát triển của các hệ thống nhúng thông minh và hiện đại.

CHƯƠNG 1 TỔNG QUAN VỀ ĐỀ TÀI

1.1 Bối cảnh của đề tài

Hiện nay, việc nghiên cứu và ứng dụng module Bluetooth HC-05 trong giao tiếp không dây đã được nhiều người thực hiện, đặc biệt trong các dự án DIY, học tập và các ứng dụng IoT nhỏ. Nhiều tài liệu, hướng dẫn và dự án mẫu đã chỉ ra cách kết nối cơ bản giữa vi điều khiển (như Arduino, PIC) và module HC-05 để truyền dữ liệu không dây qua giao thức UART. Đã có nhiều dự án sử dụng HC-05 trong điều khiển từ xa, giám sát cảm biến và hệ thống nhúng.

Đề tài này sẽ tập trung cải thiện các yếu tố này bằng cách nghiên cứu sâu hơn về giao thức UART và tối ưu hóa kết nối Bluetooth HC-05 góp phần giải quyết các vấn đề mà các nghiên cứu trước còn tồn động.

1.2 Một số nghiên cứu, dự án khác

1.2.1 Nghiên cứu thứ nhất

Nghiên cứu thứ nhất trình bày về một hệ thống tự động bật/tắt đèn xi-nhan cho phương tiện dựa trên tọa độ GPS và ứng dụng Android [1]. Hệ thống được thiết kế nhằm giảm thiểu tai nạn giao thông do người lái xe quên bật đèn xi-nhan khi chuyển hướng. Hệ thống gồm hai phần chính:

- Smartphone: Dùng để theo dõi tuyến đường thông qua GPS và gửi tín hiệu điều khiển qua Bluetooth.
- Thiết bị trên xe (On-board device): Tích hợp vi điều khiển PIC16F877A, nhận tín hiệu từ smartphone và điều khiển đèn xi-nhan.

Hệ thống hoạt động dựa trên ứng dụng Android, sử dụng Google Map để xác định lộ trình. Khi đến các điểm rẽ, tín hiệu điều khiển sẽ được gửi qua module Bluetooth HC-05 để bật đèn xi-nhan, sau đó tự động tắt sau một khoảng thời gian.

1.2.2 Nghiên cứu thứ hai

Nghiên cứu thứ hai tập trung vào việc tích hợp module Bluetooth HC-05 với phần mềm Minicom trên hệ điều hành Linux thông qua UART [2]. Nghiên cứu trình bày:

- HC-05 là một module Bluetooth giao tiếp nối tiếp, hoạt động trên công nghệ không dây hiện đại. Nó hỗ trợ giao tiếp qua giao thức UART, cho phép truyền dữ liệu tốc độ cao với chất lượng tín hiệu cải thiện.
- Minicom là một chương trình điều khiển modem và mô phỏng terminal dựa trên văn bản dành cho hệ điều hành giống Unix-like.
- Bài viết hướng dẫn cách thiết lập kết nối giữa module HC-05 và máy tính thông qua bộ chuyển đổi USB qua cổng COM, cấu hình Minicom để giao tiếp với HC-05 và sử dụng lệnh AT để điều khiển module.

Công trình này đã minh họa giao tiếp không dây bằng module Bluetooth HC-05 với UART. Module này mang lại nhiều lợi ích như bảo mật, tiêu thụ năng lượng thấp, và khả năng tương thích cao. Việc sử dụng Linux và Minicom giúp đơn giản hóa quá trình điều khiển và cấu hình.

1.2.3 Nghiên cứu thứ ba

Nghiên cứu trình bày về một hệ thống điều khiển robot giám xát từ xa sử dụng giao diện Android hoặc GUI, tích hợp module Bluetooth và camera quan sát ban đêm [3]

Hệ thống được thiết kế để hỗ trợ giám sát từ xa, hoạt động trong các khu vực nguy hiểm hoặc khó tiếp cận đối với con người. Robot được điều khiển thông qua:

- Thiết bị Android hoặc máy tính: Gửi lệnh điều khiển qua Bluetooth.
- Camera quan sát ban đêm: Gửi hình ảnh thời gian thực đến thiết bị di động.

Thành phần chính:

- Vi điều khiển PIC16F877A: Xử lý lệnh nhận được từ Bluetooth và điều khiển động cơ robot.

- Module Bluetooth HC-05: Kết nối không dây với thiết bị điều khiển.
- Camera quan sát ban đêm: Kết nối qua WiFi để truyền hình ảnh và âm thanh.

Hệ thống được cho là sử dụng để giám sát ban đêm, trong các khu vực chiến tranh, hoặc môi trường nguy hiểm như rừng rậm.

1.3 Những khó khăn, vấn đề cần giải quyết

Từ những nghiên cứu trên nhóm đã rút ra được một số thứ như sau:

1.3.1 Khó khăn

Việc đồng bộ hóa giữa ứng dụng Android và vi điều khiển qua Bluetooth đôi khi bị chậm do độ trễ hoặc mất kết nối. Bluetooth có phạm vi giới hạn (khoảng 10m) và dễ bị mất kết nối khi có vật cản hoặc tín hiệu nhiễu.

UART yêu cầu tốc độ baud của hai thiết bị phải khớp nhau hoàn toàn, nếu không sẽ dẫn đến lỗi giao tiếp nên người dùng phải thiết lập các tham số như baud rate, chế độ Master/Slave bằng lệnh AT, điều này gây khó khăn cho người không quen với kỹ thuật.

Hệ thống yêu cầu tích hợp các thiết bị như module Bluetooth, vi điều khiển, và ứng dụng, dẫn đến chi phí cao khi áp dụng cho số lượng lớn phương tiện.

Các lỗi như lỗi khung (framing error) hoặc tràn dữ liệu (overrun error) thường xảy ra trong hệ thống có nhiều thiết bị kết nối.

1.3.2 Vấn đề cần giải quyết

Tăng độ ổn định kết nối Bluetooth bằng cách tích hợp các phương pháp xử lý nhiễu tín hiệu

Phát triển phần mềm tự động cấu hình các tham số cho module HC-05 để giảm sự phụ thuộc vào người dùng

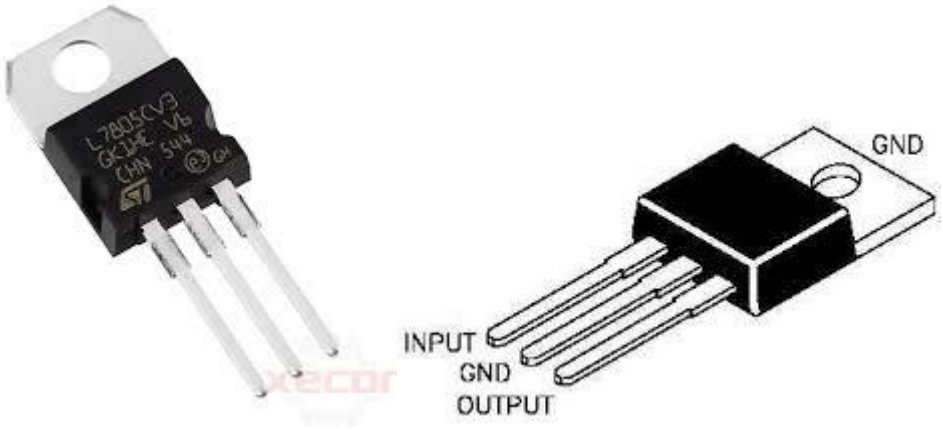
Thiết kế ứng dụng với mã nguồn tối ưu, giảm mức tiêu thụ năng lượng

Giảm chi phí hệ thống bằng cách sử dụng các module giá rẻ hoặc tích hợp tính năng vào hệ thống có sẵn

CHƯƠNG 2 CƠ SỞ LÝ THUYẾT

2.1 Tổng quan về LM7805

LM7805 thường được biết đến là một loại IC ổn áp dùng để tạo điện áp đầu ra +5V thường được tin cậy để sử dụng trong các mạch điện tử phục vụ cho mục đích thương mại



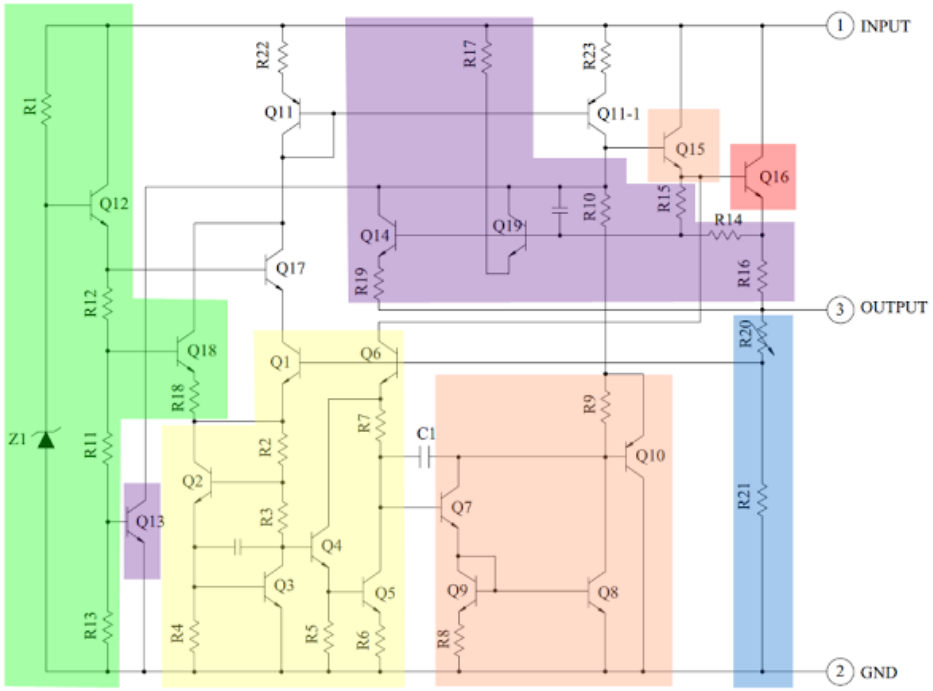
Hình 2.1.1 Hình ảnh minh họa cho LM7805

Bảng 2.1.1 Các chân của LM7805 [4]

Chân số	Tên chân	Chức năng
1	Điện áp vào (V_{in})	Nhận điện áp vào từ 7V đến 20V
2	Chân nối đất (GND)	Nối xuống mass
3	Điện áp ra (V_{out})	Cung cấp điện áp ra 4.85V đến 5.15V

Nguyên lý hoạt động:

Bộ ổn áp hoạt động dựa trên tham chiếu khoảng cấm (bandgap reference), bộ phận cung cấp điện áp ổn định và chính xác ngay cả khi nhiệt độ hay điện áp đầu vào thay đổi. Bandgap lấy tín hiệu từ điện áp đầu ra đã được giảm bởi bộ chia điện áp (vùng màu xanh dương) để so sánh và tạo tín hiệu lỗi nếu điện áp quá cao hoặc quá thấp. Tín hiệu lỗi này được khuếch đại bởi bộ khuếch đại lỗi (vùng màu cam), sau đó điều khiển bóng bán dẫn đầu ra thông qua trình điều khiển ở đầu Q15 để duy trì điện áp ổn định, hoàn thiện vòng phản hồi âm. Bộ chia điện áp có thể điều chỉnh linh hoạt để tạo ra các mức điện áp đầu ra khác nhau như 5V, 12V, hoặc 24V bằng cách thay đổi giá trị điện trở.

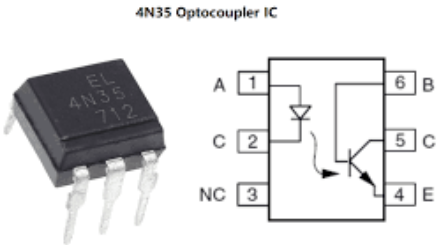


Hình 2.1.2 Hình ảnh minh họa cấu trúc bên trong của LM7805 [5]

Mạch còn được trang bị các cơ chế bảo vệ như chống quá dòng, quá nhiệt, và điện áp đầu vào quá mức nhờ các bóng bán dẫn bảo vệ. Ngoài ra, mạch khởi động (vùng màu xanh lá cây) đảm bảo bandgap nhận đủ dòng khởi động để hoạt động, tránh tình trạng bị kẹt ở trạng thái tắt. Tất cả phối hợp để đảm bảo bộ ổn áp hoạt động ổn định và luôn cho ra giá trị điện áp kì vọng.

2.2 Tổng quan về mạch bảo vệ trong giao tiếp UART sử dụng Opto-isolator 4N35

Opto-isolator 4N35 hay còn biết đến là bộ cách ly quang 4N35 thường được dùng để điều khiển sự truyền tải tín hiệu giữa hai mạch điện dựa trên sự chênh lệch điện áp thông qua ánh sáng



Hình 2.2.1 Ảnh minh họa cho Opto-isolator 4N35

Bảng 2.2.1 Các chân của Opto-isolator 4N35 [6]

Chân số	Loại chân	Mô tả
1	A	Anode
2	C	Cathode
3	NC	No connect
4	E	Emitter
5	C	Collector
6	B	Base

Nguyên lý hoạt động

Khi LED phát sáng, vùng base của transistor cảm quang tiếp nhận ánh sáng và giảm mức điện trở thuần tương đương của transistor, làm dòng qua transistor tăng. Khi đó nếu cường độ sáng đủ mạnh, thì transistor cảm quang đạt trạng thái bão hòa. Photocoupler thực hiện truyền tín hiệu Tx từ PIC16F887 hoặc từ CP2102 mini sang Rx của opto còn lại.

2.3 Tổng quan về PIC16F887-I/P

PIC16F887 là một vi điều khiển 8-bit thuộc dòng PIC16F của hãng Microchip Technology, được biết đến với sự phổ biến và ứng dụng rộng rãi trong lĩnh vực nhúng. Đây là một lựa chọn lý tưởng cho cả người mới học và các dự án thực tế nhờ tính năng phong phú, giá thành hợp lý và dễ lập trình.

Hình 2.3.1 Hình ảnh minh họa cho IC PIC16F887-I/P

Với đề tài "Giao tiếp UART giữa CP2102 mini và module Bluetooth HC-05 thông qua PIC16F887 I/P", 2 chân RC6 và RC7 của IC PIC16F887-I/P sẽ được sử dụng để phục vụ cho mục đích giao tiếp UART cứng giữa PIC16F887 với module Bluetooth HC-05 và 2 chân RA0 và RA2 làm giao tiếp UART mềm giữa PIC16F887 với CP2102 mini.

Các tính năng nổi trội của IC PIC16F887-I/P gồm:

- Watchdog Timer (WDT): Bảo vệ hệ thống trong trường hợp lỗi phần mềm.
- Power-Saving Mode: Giảm tiêu thụ năng lượng khi không hoạt động.
- Brown-out Reset: Đặt lại hệ thống khi điện áp giảm xuống mức thấp.

Ngoài ra, Điện áp hoạt động của IC PIC16F887-I/P chỉ từ 2.0V đến 5.5V, phù hợp với các ứng dụng sử dụng nguồn thấp.

2.4 Tổng quan về module Bluetooth HC-05

Module Bluetooth HC-05 là một thiết bị nhỏ gọn, có khả năng kết nối không dây giữa các thiết bị điện tử khác nhau như điện thoại, máy tính bảng, hoặc các module khác. Nó hoạt động dựa trên công nghệ Bluetooth, cho phép truyền dữ liệu không dây ở khoảng cách ngắn (thường khoảng 10m). HC-05 rất dễ sử dụng và được tích hợp sẵn các giao thức truyền thông cần thiết, giúp cho việc kết nối và truyền dữ liệu trở nên đơn giản hơn.

Hình 2.4.1 Hình ảnh minh hoạ cho module Bluetooth HC-05

Bảng 2.4.1 Các chân của HC-05 [7]

Chân số	Kí hiệu	Mô tả
1	STATE	Chân thông báo trạng thái kết nối của HC-05
2	RXD	Chân nhận dữ liệu (Recieve)
3	TXD	Chân truyền dữ liệu (Transmit)
4	GND	Chân nối đất (mass)
5	VCC	Chân cấp nguồn (nguồn 3.3V)
6	EN	Chân enable

Nguyên lý hoạt động

Kết nối VCC vào nguồn 3.3V, GND xuống mass, chân Rx của HC-05 với Tx (RC6) của PIC16F887-I/P, chân Tx của HC-05 với Rx (RC7) của PIC16F887-I/P. Sau đó kết nối với địa chỉ Bluetooth HC-05 trên điện thoại để thực hiện truyền nhận dữ liệu với CP2102 mini

2.5 Tổng quan về CP2102 mini

Module CP2102 mini là một mạch chuyển đổi USB sang UART TTL và ngược lại. Nó sử dụng chip CP2102 của Silabs, một chip chuyên dụng cho việc chuyển đổi giao tiếp này. Nhờ module CP2102 mini, bạn có thể dễ dàng kết nối một thiết bị sử dụng giao tiếp UART (như Arduino, các bo mạch vi điều khiển khác) với máy tính thông qua cổng USB. Điều này giúp việc lập trình, gỡ lỗi và truyền dữ liệu trở nên thuận tiện hơn rất nhiều.

Hình 2.5.1 Hình ảnh minh họa cho CP2102 mini

Bảng 2.5.1 Sơ đồ chân của CP2102 [8]

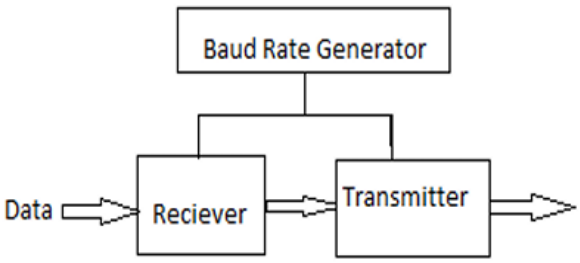
Chân số	Ký hiệu	Mô tả
1	3V3	Chân tạo nguồn 3.3V
2	TXD	Chân truyền dữ liệu
3	RXD	Chân nhận dữ liệu
4	GND	Chân nối đất
5	+5V	Chân nối nguồn 5V

Nguyên lí hoạt động

Kết nối chân Tx của CP2102 mini với Rx mềm (RA0) của PIC16F887-I/P thông qua mạch bảo vệ UART dùng opto, Rx của CP2102 mini với Tx mềm (RA2) của PIC16F887-I/P thông qua mạch bảo vệ UART, kết nối cổng USB của CP2102 mini với máy tính để giao tiếp truyền nhận dữ liệu với module bluetooth HC-05.

2.6 Nguyên lý của giao thức truyền nhận UART

UART (Universal Asynchronous Receive/Transmit) là một mạch tích hợp đóng vai trò quan trọng trong truyền thông nối tiếp. Truyền thông nối tiếp làm giảm nhiễu tín hiệu, giúp việc truyền và nhận dữ liệu trở nên tốt hơn. UART bao gồm một bộ chuyển đổi từ song song sang nối tiếp để truyền dữ liệu từ máy tính và một bộ chuyển đổi từ nối tiếp sang song song để nhận dữ liệu qua đường nối tiếp.



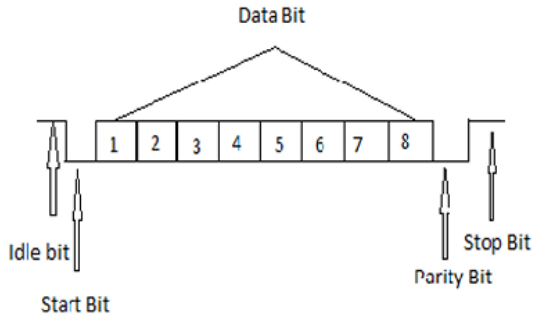
Hình 2.6.1 Các module của UART [9]

Mô-đun giao tiếp nối tiếp UART được chia thành ba mô-đun con:

- Bộ tạo tốc độ baud (Baud Rate Generator): Tạo tín hiệu xung nhịp cục bộ cao hơn tốc độ baud để kiểm soát quá trình nhận và truyền của UART

- Mô-đun nhận (Receiver): Dùng để nhận tín hiệu nối tiếp tại chân RXD và chuyển đổi chúng thành dữ liệu song song
- Mô-đun truyền (Transmitter): Chuyển đổi các byte thành các bit nối tiếp theo định dạng cơ bản và truyền các bit đó qua chân TXD

Trong giao thức UART, dữ liệu được tổ chức thành các khung truyền, bao gồm: bit bắt đầu (start bit), bit dữ liệu (data bit), bit chẵn lẻ (parity bit), bit dừng (stop bit), và trạng thái nghỉ (idle state).



Hình 2.6.2 Khung truyền của UART [9]

Khi một từ (hay còn gọi là 1 ký hiệu) được đưa vào UART để truyền, bit bắt đầu được thêm vào đầu mỗi từ dữ liệu cần truyền. Bit bắt đầu này báo hiệu cho bộ thu rằng một từ dữ liệu sắp được gửi đi và đồng bộ hóa xung nhịp trong bộ thu với bộ phát. Sau bit bắt đầu, các bit dữ liệu được gửi từng bit một, với bit ít quan trọng nhất (LSB) được gửi trước.

Khi toàn bộ dữ liệu đã được gửi đi, bộ phát có thể thêm một bit chẵn lẻ (parity bit), bit này được dùng để kiểm tra lỗi cơ bản. Sau đó, một hoặc nhiều bit dừng được gửi bởi bộ phát để đánh dấu kết thúc của khung dữ liệu. Nếu dữ liệu nhận được không đúng định dạng, UART có thể sẽ báo lỗi, còn nếu một byte mới đến trước khi byte trước đó được đọc, UART sẽ báo lỗi tràn.

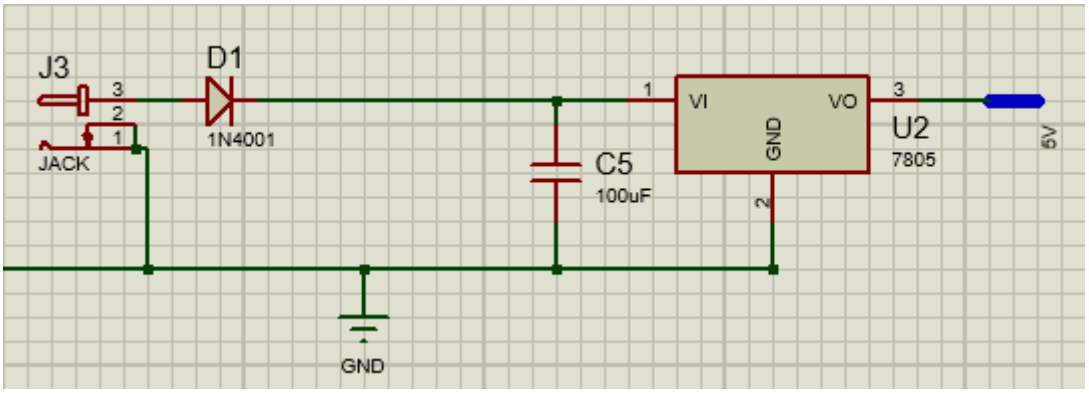
2.7 Tổng quan về mạch ổn áp (Voltage regulator)

Mạch ổn áp là một dạng mạch có chức năng điều chỉnh điện áp về một giá trị nhất định và giá trị này sẽ không bị làm thay đổi bởi giá trị của điện áp đầu vào được cấp vào mạch. Nhiệm vụ chính của mạch ổn áp là cung cấp giá trị điện áp đầu ra không đổi để đảm bảo sự hoạt động ổn định của các thiết bị, linh kiện điện tử trong mạch.

Với đề tài “Giao tiếp UART giữa CP2102 mini và module Bluetooth HC-05 thông qua PIC16F887 I/P”, nhóm sẽ áp dụng mạch ổn áp sử dụng IC LM7805 để tạo ra giá trị điện áp đầu ra 5V từ nguồn điện áp vào 9V-1A.

Nguyên lí hoạt động của mạch ổn áp dùng LM7805

Nguyên lý hoạt động của mạch ổn áp sử dụng LM7805 dựa trên khả năng điều chỉnh và duy trì điện áp đầu ra ổn định ở mức 5V DC. Điện áp đầu vào (Vin) được cấp vào chân Input (chân 1) của IC, yêu cầu phải cao hơn điện áp đầu ra ít nhất 2V (thường tối thiểu là 7V) để IC hoạt động hiệu quả. Tại đầu vào, một tụ lọc (C1) được sử dụng để giảm nhiễu và làm phẳng tín hiệu từ nguồn. Sau đó LM7805 xử lý và điều chỉnh dòng điện để cung cấp điện áp đầu ra ổn định tại chân Output (chân 3). Nếu điện áp đầu vào chênh lệch quá cao so với đầu ra, LM7805 sẽ chuyển phần năng lượng dư thành nhiệt tỏa ra ở miếng kim loại trên đầu. Cơ chế này giúp mạch duy trì đầu ra 5V ổn định, phù hợp cho các thiết bị điện tử sử dụng nguồn DC.



Hình 2.7.1 Hình ảnh sơ đồ nguyên lý của mạch ổn áp sử dụng IC LM7805

2.8 Phương pháp đo đạt

Sử dụng oscilloscope để đo và kiểm tra tính hiệu truyền đi từ UART

CHƯƠNG 3 PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

3.1 Thông số cho mạch ổn áp (Voltage regulator)

Bảng 3.1.1 Các linh kiện của mạch ổn áp

Tên linh kiện	Số lượng
Jack cắm 3.5	1
Diode 1N4001	1
Tụ điện 100uF	1
IC LM7805	1

Trong đó:

- Jack cắm 3.5 dùng để kết nối nguồn điện 9V-1A từ Adapter
- Vì nguồn cấp có dòng 1A nên nhóm chọn diode 1N4001 vừa phù hợp với hạn mức 1A của diode, vừa có chức năng bảo vệ mạch điện khỏi trường hợp mạch bị cấp ngược dòng
- Tụ điện 100uF dùng để chống nhiễu điện

3.2 Thông số cho mạch dùng opto

Mỗi optocoupler có một LED bên trong. Dòng điện định mức qua LED (I_{LED}) và điện áp rơi trên LED (V_f) được lấy từ datasheet của 4N35:

I_{LED}=10−20mA (giả sử chọn I_{LED}=15mA).

V_f=1.2V (điện áp LED khi sáng).

Điện áp cần giảm trên điện trở (R7 hoặc R8) được tính bằng:

$V_R=V_{cc}-V_f$

Thay giá trị:

V_R=5V−1.2V=3.8V

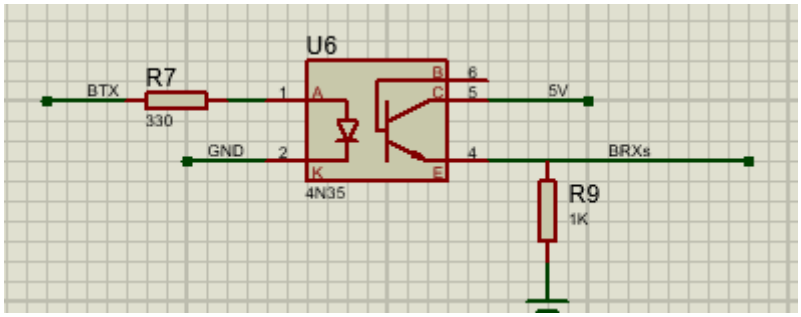
Giá trị điện trở được tính theo định luật Ohm:

$$R = \frac{V_R}{I_{LED}}$$

Thay I_{LED}=15mA:

$R = \frac{3.8V}{0.015A} \approx 253\ \Omega$

Chọn giá trị gần nhất là 330Ω, để hạn chế dòng LED nhỏ hơn.



Hình 3.2.1 Ảnh minh họa của opto 1

Ngoài ra, Transistor bên trong optocoupler hoạt động như một công tắc. Điện trở tải (R9 hoặc R10) xác định dòng điện qua collector (I_C). Thông thường, I_C lớn hơn khoảng 10-20 lần I_{LED} nhờ hệ số khuếch đại dòng DC (β) của transistor.

Giả sử:

I_C = 5mA

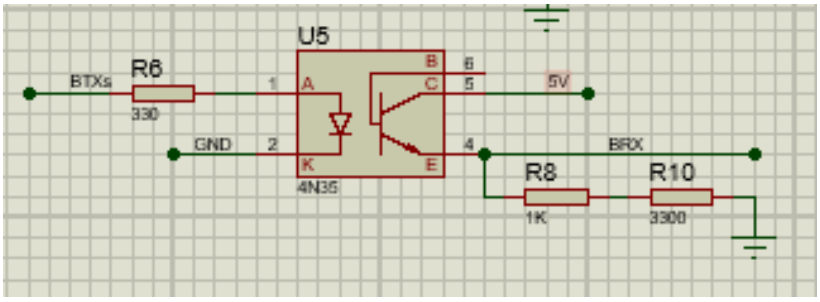
$V_{CE_{sat}}$= 0.2V (khi bão hòa)

Ta được:

$R = \frac{4.8V}{0.005A}$=960 Ω

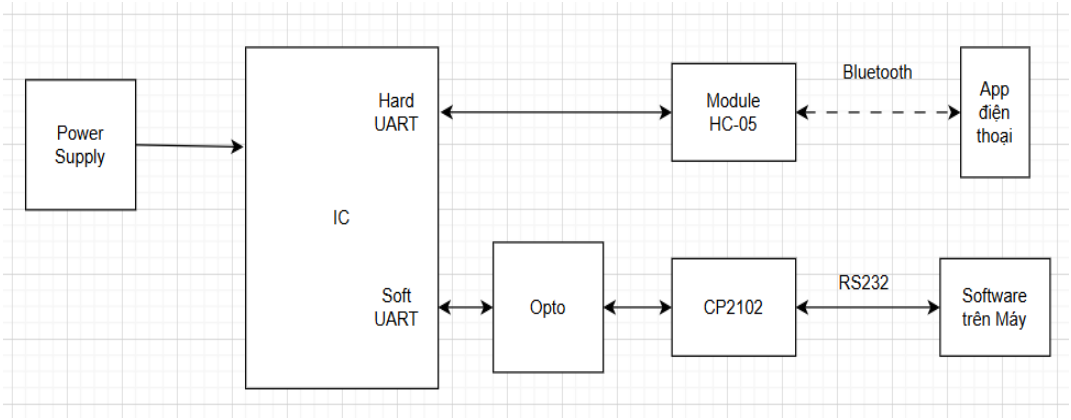
Vậy ta chọn R8, R9 xấp xỉ 1k Ω

Còn với R10, việc chọn R10=3.3K Ω nhằm mục đích tạo ra dòng I_C nhỏ hơn để BRX có thể nhận dữ liệu đúng hơn.



Hình 3.2.2 Ảnh minh họa cho opto 2

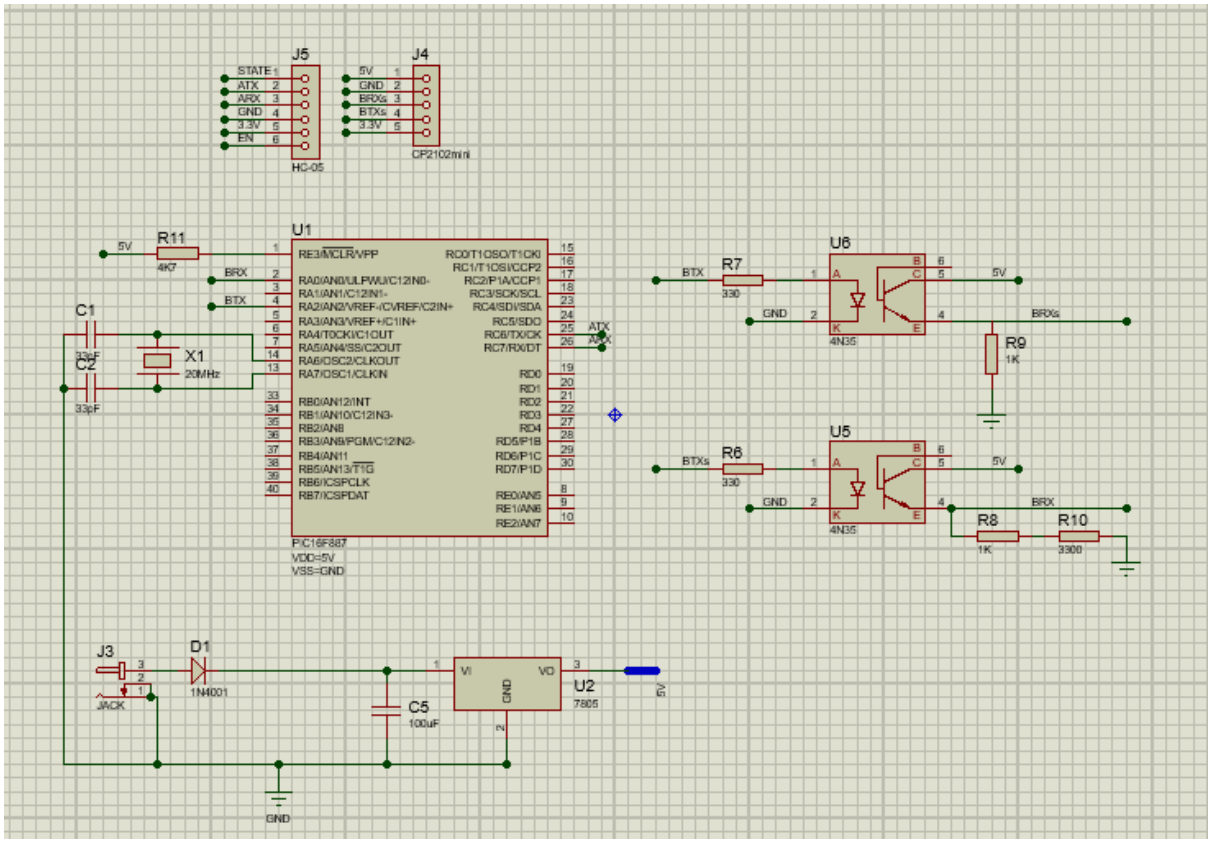
3.3 Thiết kế bo mạch và mô hình pcb



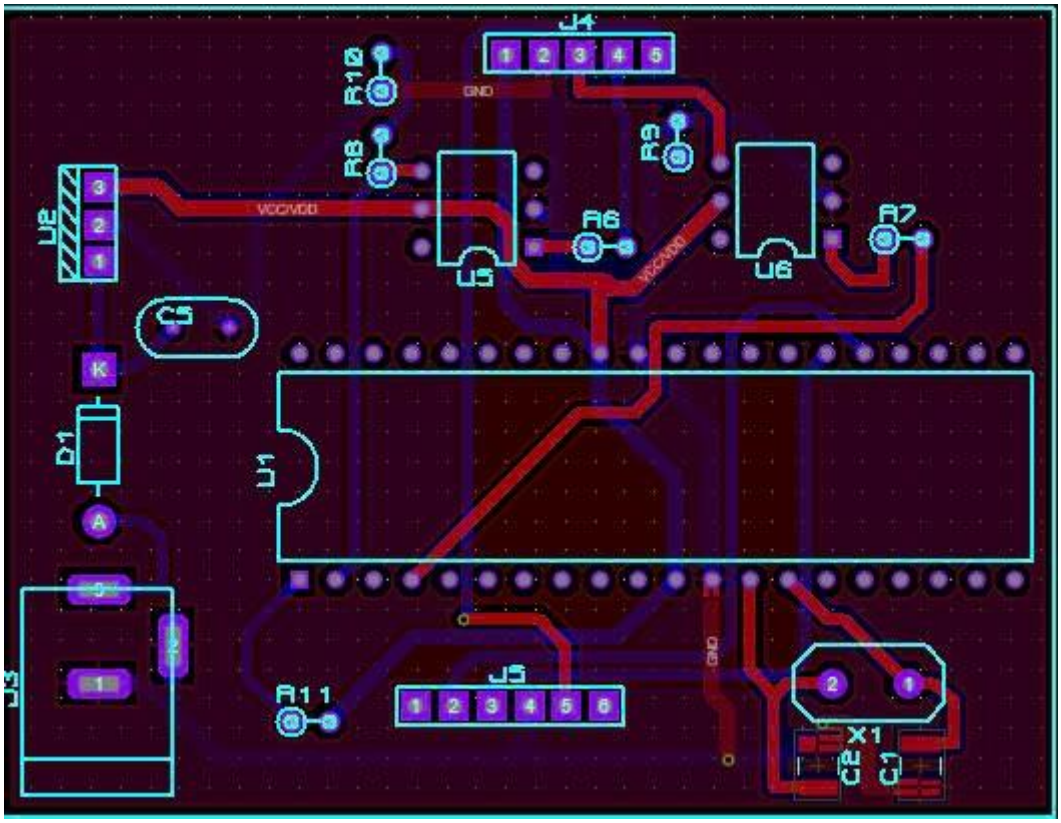
Hình 3.3.1 Sơ đồ khối của bo mạch

Trong đó sơ đồ khối gồm các khối chính sau:

- Khối IC (bao gồm thạch anh, IC, …)
- Khối ổn áp (nguồn)
- Khối opto (nối giữa sotf uart và Cp2102 để giao tiếp COM với giao diện trên máy tính)
- Khối HC-05 (nối trực tiếp với UART của pic16f887 để kết nối bluetooth với app điện thoại)



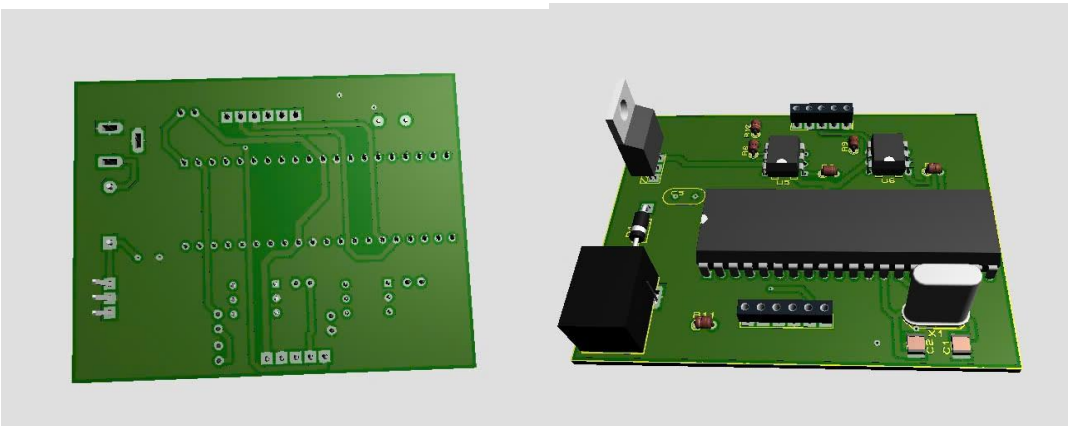
Hình 3.3.2 Sơ đồ nguyên lý của mạch



Hình 3.3.3 Sơ đồ mạch pcb

Trong đó:

- Bo mạch pcb dài 71 và rộng 54.5 (mm)
- Độ dày pcb là 1.6 (mm)
- Độ dày đồng là 1 (Oz)
- Kiểu mạ: mạ thiết
- Độ rộng đường mạch nhỏ nhất: 3.5 (mm)
- Kích thước lỗ via nhỏ nhất: 0.2 (mm)



Hình 3.3.4 Sơ đồ 3d mạch pcb

3.4 Thiết kế các giải thuật cho code

3.4.1 Firmware

UART mềm:

Dựa vào nguyên lý khung truyền của UART kết hợp với tốc độ truyền dữ liệu (Baudrate), ta có thể tạo ra một UART mềm (Soft UART) cho vi xử lý/vi điều khiển với các chân truyền (Tx) và nhận (Rx) theo ý ta mong muốn. Tốc độ truyền của 1 bit sẽ phụ thuộc vào cách ta chọn baudrate, baudrate càng lớn tốc độ truyền càng cao [10].

Ví dụ: Ta chọn baudrate là 4800 thì tốc độ truyền của 1 bit sẽ được tính như sau:

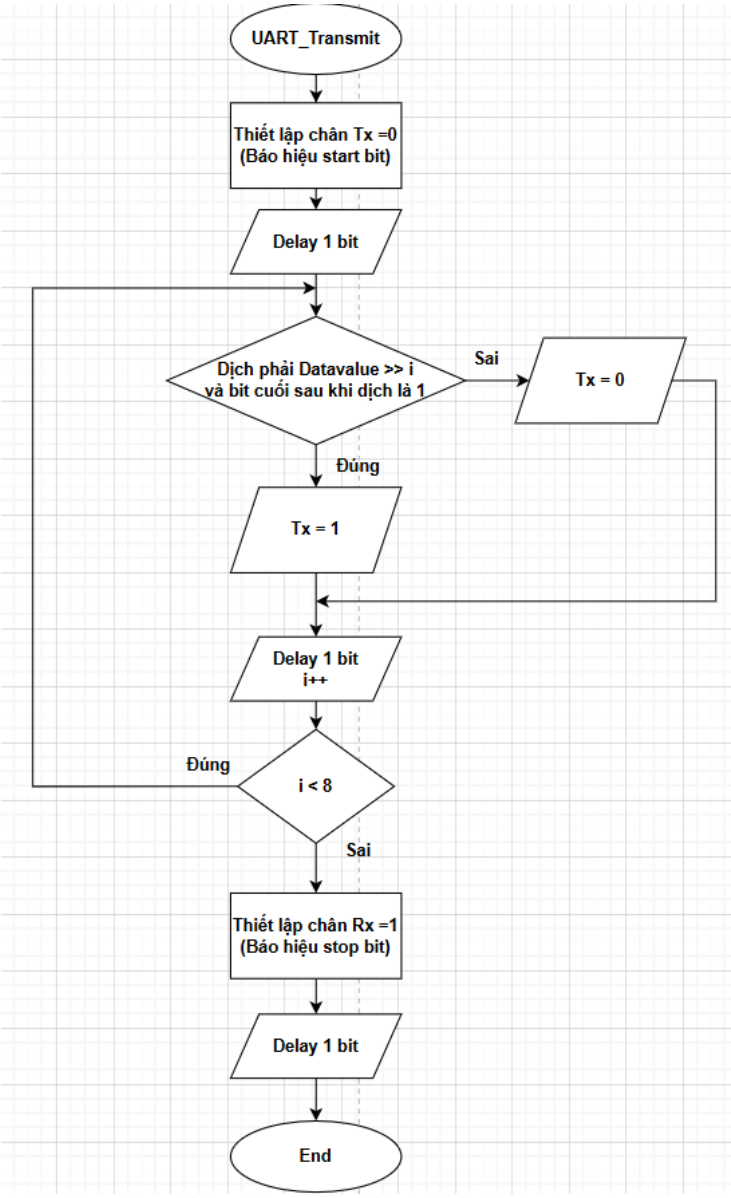
$$T_{bit} = \frac{1}{baudrate} = \frac{1}{4800} = 0.0002083\ s/bit = 208.3\ \mu s/bit$$

Trong đó: T_{bit} là thời gian truyền của 1 bit

Giả sử ta muốn truyền một ký tự "a" là "0110 0001" thì đầu tiên ta sẽ gán biến Datavalue bằng "0110 0001", sau đó cho chân Tx xuống mức thấp và delay khoảng thời gian là 1 bit như đã tính trên để báo hiệu bit bắt đầu. Ta sẽ bắt đầu tách dữ liệu gửi bằng cách cho một vòng lặp for i từ 0 đến 7 (8 bits) để xét từng bit của Datavalue. Ta dịch phải biến Datavalue theo i đơn vị và xét nếu số cuối cùng bên phải có bằng 1 hay không, nếu có thì chân Tx sẽ tích cực mức cao, còn ngược lại Tx sẽ tích cực mức thấp, sau đó delay 1 bit và khi đã hết 8 bits Tx sẽ lên mức cao để báo hiệu bit dừng.

Bảng 3.4.1 Phân tích tách từng bit cho các lần lặp i

Vòng lặp i	**Dịch phải i**	**Số cuối cùng bên phải**	**TX_PIN**
0	0110 0001	1	1 (High)
1	0011 0000	0	0 (Low)
2	0001 1000	0	0 (Low)
3	0000 1100	0	0 (Low)
4	0000 0110	0	0 (Low)
5	0000 0011	1	1 (High)
6	0000 0001	1	1 (High)
7	0000 0000	0	0 (Low)

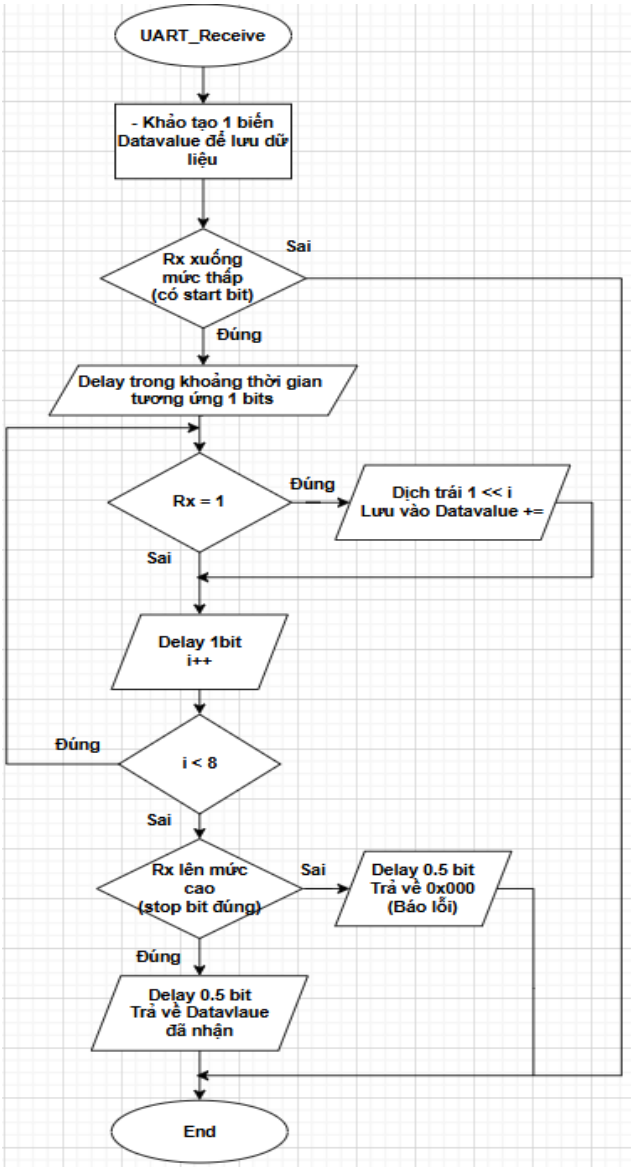


Hình 3.4.1 Lưu đồ giải thuật cho quá trình gửi của UART mềm

Còn nếu ta muốn nhận một ký tự "a" là "0110 0001", đầu tiên ta tạo một biến Datavalue và gán cho nó giá trị 0, sau đó ta sẽ chờ cho đến khi chân Rx xuống mức thấp (có bit bắt đầu), delay 1 bit cho bit bắt đầu. Ta xét, nếu Rx mức cao ta sẽ dịch trái số 1 sang i đơn vị, lưu giá trị dịch đó vào biến Datavalue mới và cộng nhị phân nó với biến Datavalue cũ, cho vào vòng lặp for i 8 lần (gộp 8 bits lại) delay 1 bit trong mỗi lần lặp. Sau 8 bits nếu chân Rx tích cực cao (tức có bit dừng) sẽ trả về Datavalue đã được gộp, còn ngược lại sẽ báo lỗi.

Bảng 3.4.2 Phân tích gộp từng bit cho các lần lặp i

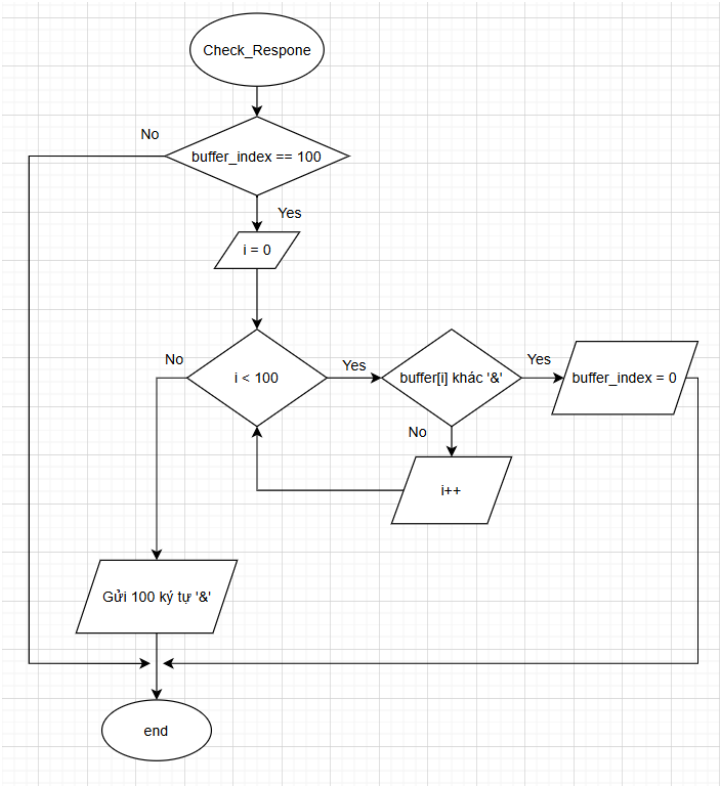
Vòng lặp i	**RX_PIN**	**Dịch trái i và cộng nhị phân**
0	1	0000 0000 0000 0001 0000 0001
1	0	0000 0001
2	0	0000 0001
3	0	0000 0001
4	0	0000 0001
5	1	0000 0001 0010 0000 0010 0001
6	1	0100 0000 0010 0001 0110 0001
7	0	0110 0001



Hình 3.4.2 Lưu đồ giải thuật cho quá trình nhận của UART mềm

Xử lý trạng thái test (khi nhận 100 ký tự "&"):

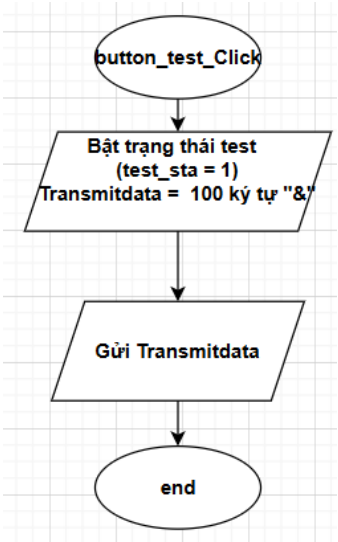
Tạo một dãy toàn cục rỗng buffer với 100 ký tự tối đa và một biến toàn cục buffer_index bằng 0 để xét số ký tự trong dãy buffer biến này sẽ tăng lên mỗi khi nhận được một giá trị từ UART mềm, nếu giá trị buffer_index bằng với 100 thì ta sẽ bắt đầu xét từng vị trí trong index xem các giá trị trong dãy buffer có phải là ký tự "&" không, nếu đúng thì ta sẽ gửi ngược về 100 ký tự "&"



Hình 3.4.3 Lưu đồ giải thuật cho quá trình test

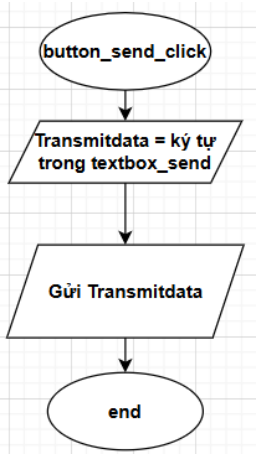
3.4.2 Software

Nguyên lý test của software rất đơn giản, khi ta nhấn nút Test thì chương trình sẽ cho biến báo trạng thái test test_sta lên mức True, sau đó gán biến Transmitdata với 100 ký tự "&" liên tiếp và gửi biến Transmitdata đi



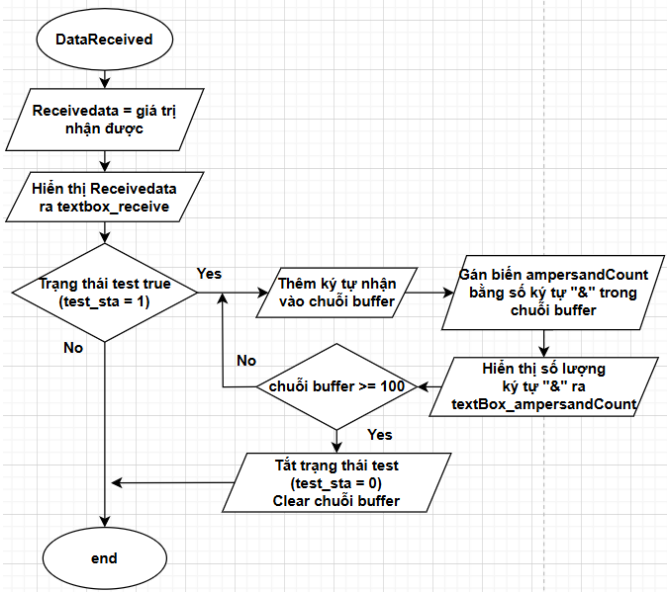
Hình 3.4.3 Lưu đồ giải thuật của nút Test

Tương tự vậy với nút Send cũng rất đơn giản, khi nhấn send thì chương trình sẽ lưu các ký tự trong ô textbox_send vào biến Transmitdata và gửi biến đó đi



Hình 3.4.4 Lưu đồ giải thuật của nút Send

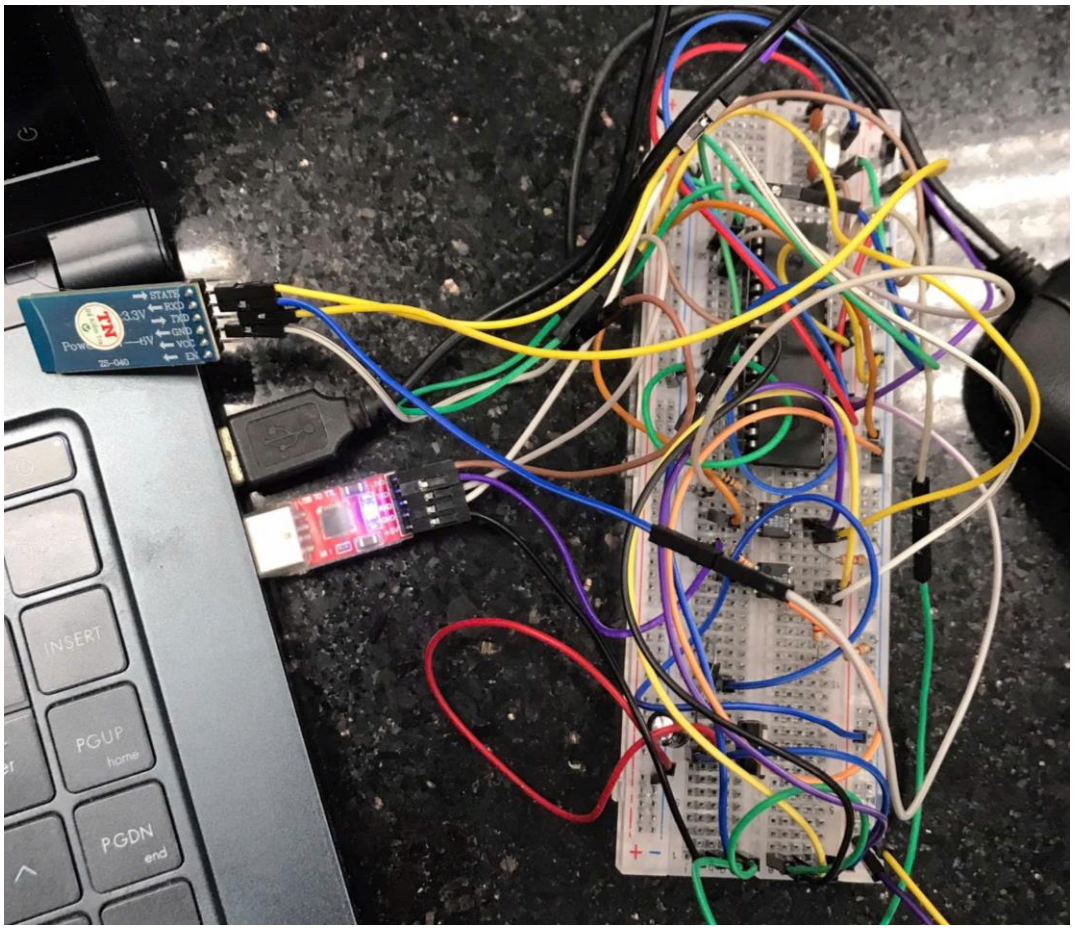
Còn đối với việc xử lý khi nhận data sẽ phức tạp hơn một chút, đầu tiên ta gán dữ liệu nhận được cho biến Receivedata, sau đó ta sẽ chuyển đổi Receivedata thành chuỗi và hiển thị nó ra textbox_receive. Nhưng nếu biến báo trạng thái test test_sta là True thì chương trình sẽ thêm các ký tự nhận vào 1 chuỗi buffer, tạo 1 biến để đếm là apersandCount và bắt đầu đếm số ký tự "&" có trong chuỗi buffer trên, chương trình sẽ đếm cho đến khi chuỗi buffer lớn hơn hoặc bằng 100 ký tự thì sẽ clear chuỗi buffer và tắt biến test (cho test_sta = false). Trong quá trình đếm chương trình sẽ hiển thị số ký tự "&" đếm được trong chuỗi ra textbox



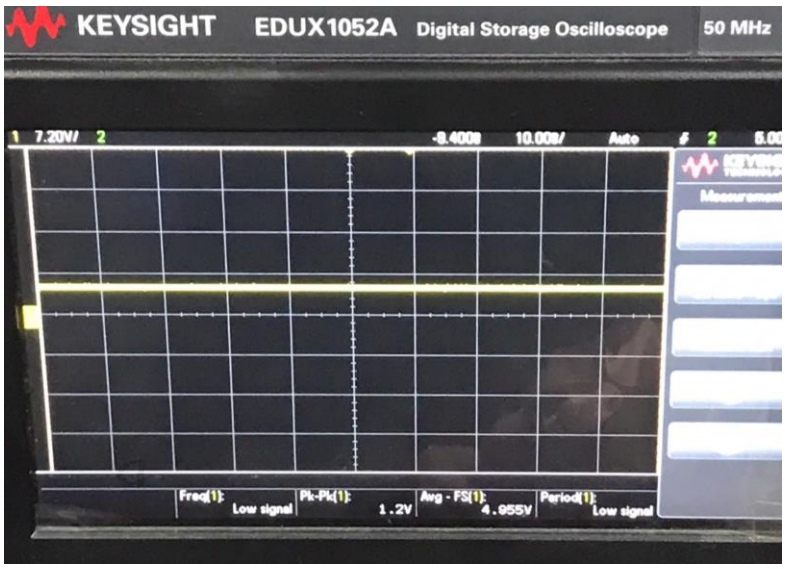
Hình 3.4.5 Lưu đồ giải thuật khi nhận dữ liệu

CHƯƠNG 4 THỰC NGHIỆM VÀ ĐÁNH GIÁ KẾT QUẢ

4.1 Hình ảnh thực nghiệm, đo đạc



Hình 4.1.1 Mạch trên breadboard



Hình 4.1.2 Nguồn ổn áp 5V đo trên oscilloscope

4.2 Đánh giá kết quả

Mạch ổn áp 5V: Do dòng điện là dòng DC (1 chiều) nên tín hiệu đo được là dạng tín hiệu thẳng (xem hình 4.1.2). Ổn áp 5V cho ra dòng điện ít nhiễu vì tín hiệu đo được thông qua oscillo cho ra dạng xung thẳng và ít tạo ra xung nhọn làm biến dạng xung

KẾT LUẬN VÀ KIẾN NGHỊ

Kết luận

Nhóm đã thành công trong việc thiết lập giao tiếp giữa module bluetooth HC-05 và PIC16F887 thông qua UART. Qua quá trình nghiên cứu và thực hiện đã giúp nhóm hiểu rõ hơn về nguyên lý hoạt động của giao tiếp UART cũng như cách hoạt động của các linh kiện, các mạch, trong đó mới mẻ nhất là mạch opto. Nhóm cũng đã triển khai hệ thống một cách có thể nói là thành công.

Tuy nhiên, trong quá trình thực hiện, vẫn tồn tại một số hạn chế như khả năng truyền dữ liệu bị ảnh hưởng bởi khoảng cách và môi trường truyền sóng, cùng với những khó khăn trong việc tối ưu hóa mã nguồn và giao tiếp đồng thời nhiều thiết bị.

Kiến nghị

Cho các nghiên cứu có liên quan đến đề tài sau này, nhóm có một vài lưu ý sau:

- Tối ưu hóa mã nguồn, giảm RAM sử dụng do PIC16F887 có hạn, nên dùng EEPROM hoặc Flash (Program Memory) nếu có thể.
- Nên kết hợp module Wi-Fi (như ESP8266 hoặc ESP32) để xây dựng hệ thống giao tiếp qua internet, giúp điều khiển từ xa từ bất kỳ đâu.
- Tích hợp thêm các cảm biến và thiết bị đầu ra như relay, màn hình hiển thị để tăng tính ứng dụng.
- Sử dụng các module Bluetooth tiên tiến hơn (như HC-06 hoặc BLE) để cải thiện tốc độ và hiệu quả giao tiếp.

TÀI LIỆU THAM KHẢO

[1] A. Senthilkumar, G. Kaviya, K. Manojkumar và S. Indrajith, "Automatic Vehicle Flasher Control Using Google Map," *International Journal of Research in Engineering, Science and Management,* pp. 367-371, May 2020.

[2] S. H. Jayantilal, "Interfacing of AT Command based HC-05 Serial Bluetooth Module with Minicom in Linux," *IJSRD - International Journal for Scientific Research & Development,* pp. 329-332, 2014.

[3] N. K. Thomas, R. S, ShyamMohanan.C, S. Rajan, D. S. Koovackal và S. A, "Android/GUI Controlled Bluetooth Spy Robot (SPY-BOT)," *Nirmal K Thomas.et.al.Int. Journal of Engineering Research and Application,* pp. 22-24, tháng 5 năm 2017.

[4] "alldatasheet.com," FCI, [Trực tuyến]. Available: https://www.alldatasheet.com/html-pdf/139342/FCI/LM7805/112/2/LM7805.html. [Đã truy cập 17 November 2024].

[5] K. Shirriff, "righto.com," [Trực tuyến]. Available: https://www.righto.com/2014/09/reverse-engineering-counterfeit-7805.html. [Đã truy cập 17 tháng 11 năm 2024].

[6] "alldatasheet.com," TOSHIBA, [Trực tuyến]. Available: https://www.alldatasheet.com/html-pdf/30832/TOSHIBA/4N35/243/1/4N35.html. [Đã truy cập 20 tháng 10 năm 2024].

[7] "components101.com," Itead Studio, [Trực tuyến]. Available: https://components101.com/sites/default/files/component_datasheet/HC-05%20Datasheet.pdf. [Đã truy cập 15 tháng 10 năm 2024].

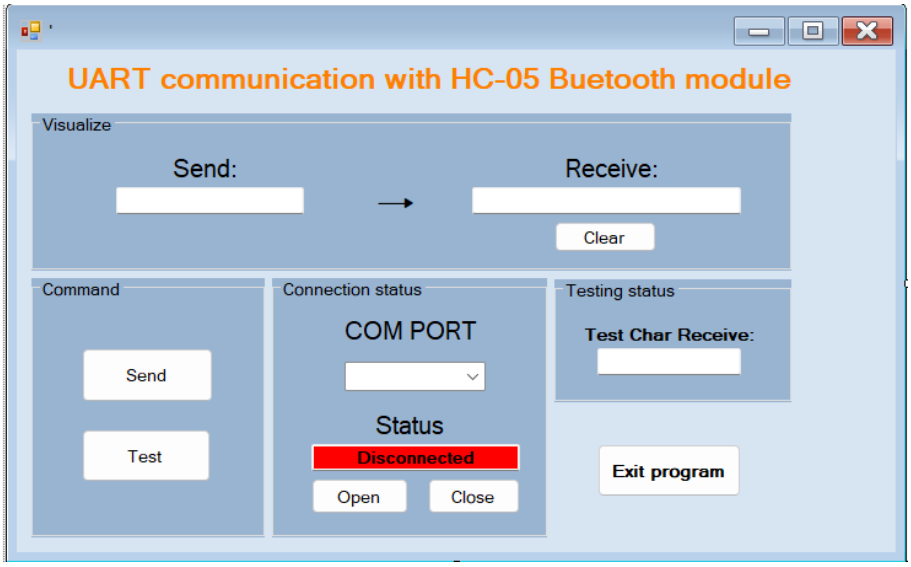
[8] "mouser.com," Silicon Labs, [Trực tuyến]. Available: https://www.mouser.com/ds/2/368/cp2102-42497.pdf. [Đã truy cập 27 tháng 10 năm 2024].

[9] M. R. Laddha và A. P. Thakare, "A Review on Serial Communication by UART," *International Journal of Advanced Research in Computer Science and Software Engineering,* pp. 366-369, tháng 1 năm 2013.

[10] Saeed, "saeedsolutions.blogspot.com," [Trực tuyến]. Available: https://saeedsolutions.blogspot.com/2012/10/pic16f84a-software-uart-bit-banging.html?utm_source=zalo&utm_medium=zalo&utm_campaign=zalo. [Đã truy cập 29 tháng 9 năm 2024].

PHỤ LỤC

Software:



Ảnh giao diện

Code giao diện:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;
using System.IO.Ports;
using System.Xml;
using System.Threading;
namespace GiaoDien
{
    public partial class Form1 : Form
```

```
{
    string ReceiveData = String.Empty;
    string TransmitData = String.Empty;
    bool test_sta = false;
    private StringBuilder receivedBuffer = new StringBuilder();
    public Form1()
    {
        InitializeComponent();
    }
    private void Form1_Load(object sender, EventArgs e)
    {
        serialPort.PortName = "Select COM Port...";
        serialPort.BaudRate = 4800;
        serialPort.DataBits = 8;
        serialPort.Parity = Parity.None;
        serialPort.StopBits = StopBits.One;
        //Doc thong tin cac cong COM co trong PC
        string[] ports = SerialPort.GetPortNames();
        //Them ten  cua tat ca cac cong vao muc COM Port
        foreach (string port in ports)
        {
            comboBox_com.Items.Add(port);
        }
    }
    private void Form1_FormClosed(object sender, FormClosedEventArgs e)
    {
        if (serialPort.IsOpen)
            serialPort.Close();
    }
    private void button_open_Click(object sender, EventArgs e)
```

```
    {
        if (comboBox_com.Text == "")
            MessageBox.Show("Select COM Port.", "Warning", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
        else
        {
            //Xu ly mo cong COM da chon
            try
            {
                if (serialPort.IsOpen) //xu ly truong hop da ket noi
                {
                    MessageBox.Show("COM Port is connected and ready for use.",
"Information", MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                else //xu ly truong hop chua ket noi
                {
                    serialPort.Open(); //Mo cong COM
                    MessageBox.Show(comboBox_com.Text + " is connected.", "Information",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                    textBox_status.BackColor = Color.Lime; //Hieu chinh mau va thong tin
                    textBox_status.Text = "Connected";
                    comboBox_com.Enabled = false;
                    ReceiveData = String.Empty;
                    TransmitData = String.Empty;
                }
            }
            catch (Exception) // Xu ly xuat hien loi khong thay thiet bi
            {
                textBox_status.BackColor = Color.Red;
                textBox_status.Text = "Disconnected";
                MessageBox.Show("COM Port is not found. Please check your COM or Cable.",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
            }
        }
    }
    private void button_close_Click(object sender, EventArgs e)
    {
        try
        {
            if (serialPort.IsOpen) //Xu ly truong hop da ket noi
            {
                serialPort.Close();
                textBox_status.BackColor = Color.Red;
                textBox_status.Text = "Disconnected!";
                MessageBox.Show("COM port is disconnected!", "Information",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                comboBox_com.Enabled = true;
            }
            else //xu ly truong hop chua ket noi
            {
                MessageBox.Show("COM port have been disconnected. Please reconnect to
use", "Information", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
        }
        catch (Exception) // xu ly khi xuat hien loi
        {
            MessageBox.Show("Disconnection appears error. Unable to disconnect.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
    private void button_send_Click(object sender, EventArgs e)
    {
        TransmitData = textBox_send.Text;
```

```
    serialPort.Write(TransmitData);
}
private void serialPort_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
    string receivedData = serialPort.ReadExisting();
    this.Invoke(new Action(() =>
    {
        textBox_receive.AppendText(receivedData);
    }));
    if (test_sta)
    {
        receivedBuffer.Append(receivedData);
        int ampersandCount = receivedBuffer.ToString().Count(c => c == '&');

        // Cập nhật giao diện hiển thị số ký tự '&'
        this.Invoke(new Action(() =>
        {
            textBox_ampersandCount.Text = ampersandCount.ToString();
        }));
        // Nếu tổng ký tự đã nhận được đủ 100, kết thúc kiểm tra
        if (receivedBuffer.Length >= 100)
        {
            test_sta = false; // Tắt chế độ kiểm tra
            receivedBuffer.Clear(); // Xóa bộ đệm
        }
    }
}
private void comboBox_com_SelectedIndexChanged(object sender, EventArgs e)
{
    //Xu ly khi chon cong COM
```

```
            serialPort.Close();
            textBox_status.BackColor = Color.Red;
            textBox_status.Text = "Disconnected!";
            serialPort.PortName = comboBox_com.Text;
        }
        private void button_test_Click(object sender, EventArgs e)
        {
            test_sta = true;
            TransmitData =
"&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&";
            serialPort.Write(TransmitData);
        }
        private void button_exit(object sender, EventArgs e)
        {
            DialogResult answer = MessageBox.Show("Ban co muon thoat chuong trinh
khong?", "Question", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
            if (answer == DialogResult.Yes)
            {
                if (serialPort.IsOpen)
                {
                    serialPort.Close();
                }
                this.Close();
            }
        }
        private void button_clear(object sender, EventArgs e)
        {
            textBox_receive.Text = " ";
        }
    }
```

```
}
```

Firmware:

Code định dạng :

```
#ifndef __SOFT_UART_H
#define __SOFT_UART_H
#ifndef _XTAL_FREQ
 // This definition is required to calibrate __delay_us() and __delay_ms()
 #define _XTAL_FREQ 20000000
#endif
#define Baudrate            4800                // bps
#define OneBitDelay         (1000000/Baudrate)          // microseconds
#define DataBitCount        8                   // no parity, no flow control
#define UART_RX             RA0                                 // UART RX pin
#define UART_TX             RA2                                 // UART TX pin
#define UART_RX_DIR                     TRISA0                              //
UART RX pin direction register
#define UART_TX_DIR                     TRISA2                              //
UART TX pin direction register
//Function Declarations
void InitSoftUART(void);
unsigned char UART_Receive(void);
void UART_Transmit(const char);
#endif
```

Code UART mềm:

```
#include "a.h"
void InitSoftUART(void)             // Cai dat cac chan Rx Tx
{
        UART_TX = 1;                        // TX cao khi khong lam gi

        UART_RX_DIR = 1;                    // Input
```

```
    UART_TX_DIR = 0;                    // Output
}

unsigned char UART_Receive(void)
{
    unsigned char DataValue = 0;
    while(UART_RX==1);
    __delay_us(OneBitDelay);
    for ( unsigned char i = 0; i < DataBitCount; i++ )
    {
        if ( UART_RX == 1 )
        {
            DataValue += (1<<i);
        }

        __delay_us(OneBitDelay);
    }

    // Check for stop bit
    if ( UART_RX == 1 )
    {
        __delay_us(OneBitDelay/2);
        return DataValue;
    }
    else
    {
        __delay_us(OneBitDelay/2);
        return 0x000;
    }
```

```
}
void UART_Transmit(const char DataValue)
{

    UART_TX = 0;
    __delay_us(OneBitDelay);

    for ( unsigned char i = 0; i < DataBitCount; i++ )
    {
        if( ((DataValue>>i)&0x1) == 0x1 )
        {
            UART_TX = 1;
        }
        else
        {
            UART_TX = 0;
        }

      __delay_us(OneBitDelay);
    }

    UART_TX = 1;
    __delay_us(OneBitDelay);
}
```

Code chính:

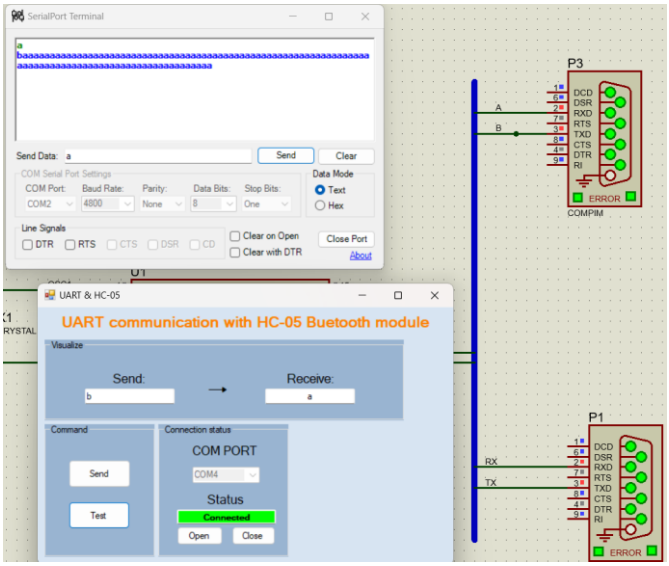
```
#pragma config FOSC = HS        // Oscillator Selection bits (HS oscillator: High-speed crystal/resonator on RA6/OSC2/CLKOUT and RA7/OSC1/CLKIN)

#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled and can be enabled by SWDTEN bit of the WDTCON register)
```

```
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT disabled)
#pragma config MCLRE = OFF      // RE3/MCLR pin function select bit (RE3/MCLR pin function is digital input, MCLR internally tied to VDD)
#pragma config CP = OFF         // Code Protection bit (Program memory code protection is disabled)
#pragma config CPD = OFF        // Data Code Protection bit (Data memory code protection is disabled)
#pragma config BOREN = OFF      // Brown Out Reset Selection bits (BOR disabled)
#pragma config IESO = OFF       // Internal External Switchover bit (Internal/External Switchover mode is disabled)
#pragma config FCMEN = OFF      // Fail-Safe Clock Monitor Enabled bit (Fail-Safe Clock Monitor is disabled)
#pragma config LVP = OFF        // Low Voltage Programming Enable bit (RB3 pin has digital I/O, HV on MCLR must be used for programming)


// CONFIG2
#pragma config BOR4V = BOR40V   // Brown-out Reset Selection bit (Brown-out Reset set to 4.0V)
#pragma config WRT = OFF        // Flash Program Memory Self Write Enable bits (Write protection off)
#include "a.h"
// Config word
char buffer[96];
unsigned char i;
unsigned char buffer_index = 0;
void UART_Init(long baud_rate) {
   unsigned int x;
   x = (_XTAL_FREQ - baud_rate * 64) / (baud_rate * 64);
   if (x > 255) {
      x = (_XTAL_FREQ - baud_rate * 16) / (baud_rate * 16);
      BRGH = 1;
```

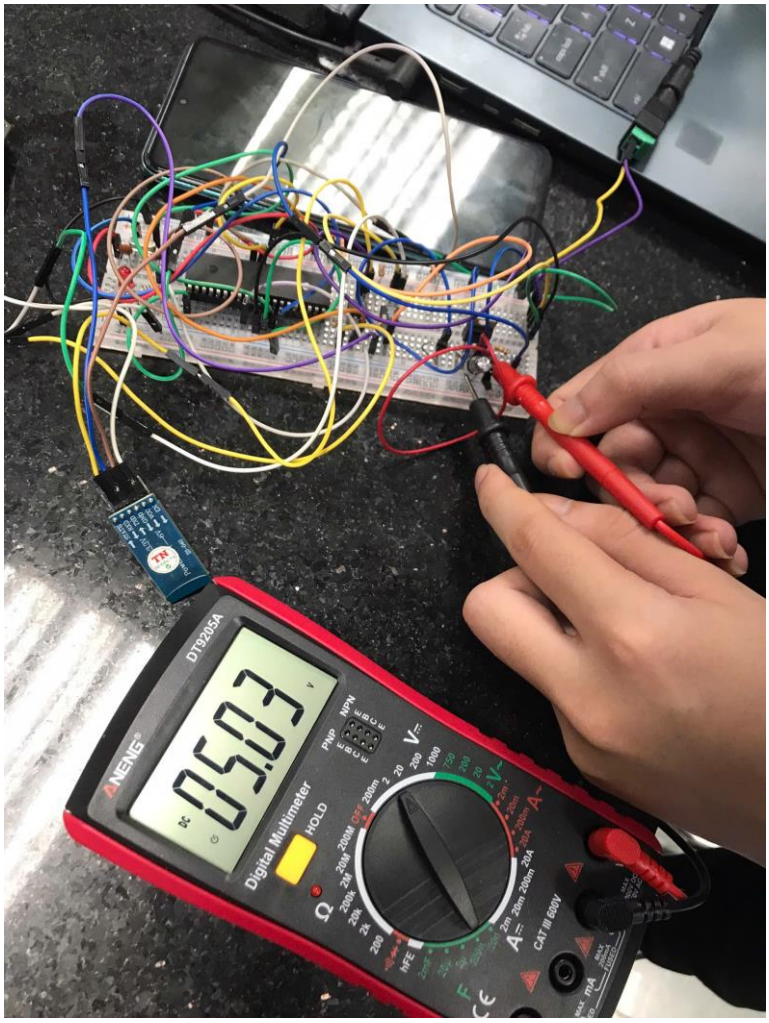
```
    } else {
        BRGH = 0;
    }
    SPBRG = x;
    SYNC = 0;
    SPEN = 1;
    TXEN = 1;
    CREN = 1;
    TX9 = 0;
    RX9 = 0;

    RCIE = 1;
    PEIE = 1;
    GIE = 1;
}
void UART_Write(char data) {
    while (!TXIF)
        ;
    TXREG = data;
}

unsigned char UART_Read() {
    while (!RCIF)
        ;
    return RCREG;
}

unsigned char UART_Data_Ready() {
    return RCIF;
```

```
}

void __interrupt() ISR() {
    if (RCIF) {
        char receivedData = RCREG;
        UART_Transmit(receivedData);
    }
}
void HandleSpecialSequence() {
    for (i = 0; i < 96; i++) {
        UART_Transmit('&');
    }
    buffer_index = 0;
}

void CheckAndRespond() {
    if (buffer_index == 96) {
        for (i = 0; i < 96; i++) {
            if (buffer[i] != '&') {
                buffer_index = 0;
                return;
            }
        }
        HandleSpecialSequence();
    }
}

// Main function
void main()
```

```
{
   unsigned char ch =0;
   unsigned char data = 0;
   UART_Init(4800);
        InitSoftUART();
   ANSEL = ANSELH = 0x00;
        while(1)
        {
        ch = UART_Receive();
        if (ch != 0x00) {
        UART_Write(ch);

        buffer[buffer_index++] = ch;
        CheckAndRespond();
        }
   }

}
```



Ảnh mô phỏng



Ảnh lắp mạch và thử nghiệm trên breadboard