

**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

# **PROJECT I**

**Máy đọc thẻ Reader đa chức năng kết nối xa**

**Mai Xuân Ngọc**

ngoc.mx204769@sis.hust.edu.vn

**Nghành Kỹ thuật Máy tính**

**Giảng viên hướng dẫn:** ThS. Nguyễn Đức Tiến

\_\_\_\_\_  
Chữ kí GVHD

**Khoa:**

Kỹ thuật máy tính

**Trường:**

Công nghệ thông tin và Truyền thông

**HÀ NỘI, 11/2022**

# LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành đến thầy **Nguyễn Đức Tiến** - giảng viên Khoa Kỹ thuật máy tính - Trường Công nghệ Thông tin và Truyền thông đã cung cấp cho em những kiến thức, kỹ năng để hoàn thành đề tài này.

Trong quá trình nghiên cứu và thực hiện đề tài, do kiến thức và kỹ năng chuyên ngành còn hạn chế nên em còn nhiều thiếu sót khi tìm hiểu, đánh giá và trình bày đề tài. Em rất mong nhận được sự quan tâm, góp ý của thầy để đề tài của em được đầy đủ và hoàn chỉnh hơn.

Xin chân thành cảm ơn!

# TÓM TẮT NỘI DUNG ĐỒ ÁN

Hiện nay chuẩn kết nối RS485 mới ra đời đã cho phép chúng ta có thể kết nối xa hơn so với chuẩn kết nối RS232 cũ. Thế nhưng hiện tại các máy tính nhúng Arduino (Uno, NodeMCU,...) đang sử dụng chuẩn kết nối RS232 này. Trong Đồ án này em sẽ sử dụng Module để chuyển tín hiệu RS232 của Arduino thành RS485 để tăng khoảng cách trao đổi dữ liệu. Ngoài ra trong đồ án này em sử dụng máy tính nhúng NodeMCU ESP8266 tích hợp sẵn module Wifi để kết nối với Internet. Kết quả thu được là chúng ta sẽ có một máy đọc thẻ có thể sử dụng kết nối với máy tính nhúng qua chuẩn kết nối tầm xa RS485 và có thể gửi dữ liệu từ xa về Sever.

## MỤC LỤC

<b>CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI.....</b>	<b>1</b>
1.1 Đặt vấn đề.....	1
1.2 Mục tiêu và phạm vi đề tài.....	1
1.3 Định hướng giải pháp.....	1
1.4 Các công nghệ sử dụng .....	1
1.4.1 Git.....	1
1.4.2 Arduino IDE.....	1
1.4.3 SiOT.....	2
1.5 Một số thiết bị phần cứng .....	2
1.5.1 Mạch Arduino NodeMCU ESP8266-12E .....	2
1.5.2 USB TO COM CP2102 .....	3
1.5.3 Máy đọc thẻ Reader.....	4
<b>CHƯƠNG 2. KHẢO SÁT .....</b>	<b>5</b>
2.1 Tổng quan chức năng .....	5
2.2 Sơ đồ lắp mạch.....	5
2.3 Quy trình nghiệp vụ .....	6
2.4 Yêu cầu phi chức năng .....	6
<b>CHƯƠNG 3. CƠ SỞ LÝ THUYẾT .....</b>	<b>7</b>
3.1 Giao thức RS232 .....	7
3.1.1 Cơ chế hoạt động .....	7
3.1.2 Ưu - Nhược điểm .....	8
3.2 Giao thức RS485 .....	8
3.2.1 Cơ chế hoạt động .....	8
3.2.2 Ưu - Nhược điểm .....	9

3.3 Wifi.....	9
3.3.1 Ưu điểm của Wifi.....	9
<b>CHƯƠNG 4. KẾT NỐI, CẬP NHẬT DỮ LIỆU LÊN SIOT SEVER.....</b>	<b>10</b>
4.1 Cài đặt chương trình.....	10
4.2 Tạo thiết bị ảo website <code>siot.soict.ai</code> .....	10
4.3 Kết nối sever .....	11
4.4 Kết quả thu được .....	11
<b>CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN .....</b>	<b>13</b>
5.1 Kết luận .....	13
5.2 Một số lỗi trong quá trình thực hiện.....	13
5.3 Hướng phát triển.....	15
<b>TÀI LIỆU THAM KHẢO.....</b>	<b>17</b>

## DANH MỤC HÌNH VẼ

Hình 1.1	NodeMCU ESP8266-12E . . . . .	2
Hình 1.2	USB COM CP2102 . . . . .	3
Hình 1.3	6 mode hoạt động của USB COM CP2102 . . . . .	3
Hình 1.4	Máy đọc thẻ . . . . .	4
Hình 2.1	Sơ đồ lắp mạch . . . . .	5
Hình 2.2	Quy trình nghiệp vụ của bài toán. . . . .	6
Hình 3.1	Giao thức RS232 . . . . .	7
Hình 4.1	Tạo thiết bị ảo . . . . .	10
Hình 4.2	Dữ liệu từ cổng COM . . . . .	11
Hình 4.3	Dữ liệu từ web . . . . .	11
Hình 5.1	Lỗi trong quá trình đưa dữ liệu lên Sever . . . . .	14
Hình 5.2	Sau khi fix lỗi. Dữ liệu đã được đưa lên sever thành công . . .	14
Hình 5.3	Cấu hình wifi . . . . .	15

# CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

## 1.1 Đặt vấn đề

Mục tiêu của máy đọc thẻ Đa kết nối từ xa có thể kết hợp với camera, sinh trắc học... để tạo thành một thiết bị an ninh hoạt động trên nền nhúng. Các thiết bị nhúng hiện tại có một nhược điểm đó là sử dụng giao thức truyền RS232 với khoảng cách truyền ngắn. Giả sử muốn kiểm soát người ra vào kho thì phải sử dụng máy tính cá nhân đặt ở cửa ra vào dẫn tới việc lãng phí tài nguyên và vấn đề an toàn của cả máy tính. Bài toán đặt ra là làm như thế nào để có thể sử dụng được máy đọc thẻ linh hoạt trong mọi tình huống.

## 1.2 Mục tiêu và phạm vi đề tài

Mục tiêu của dự án này là (i) sử dụng Arduino NodeMCU wifi, (ii) chuyển đổi tín hiệu từ RS232 sang RS485, (iii) gửi các chuỗi mã quét được lên sever SiOT thông qua thư viện SiOTCore.

## 1.3 Định hướng giải pháp

Từ việc xác định rõ nhiệm vụ cần giải quyết ở phần 1.2, định hướng giải pháp của em theo trình tự sau: (i) Lắp ráp các thiết bị phần cứng; Tiếp theo, (ii) lập trình đọc dữ liệu từ thiết bị; và sau cùng, (iii) gửi dữ liệu đọc được lên Sever SiOT.

## 1.4 Các công nghệ sử dụng

### 1.4.1 Git

Git là hệ thống kiểm soát phiên bản phân tán mà nguồn mở (Open Source Distributed Version Control System). Các dự án thực tế thường có nhiều nhà phát triển làm việc song song. Vì vậy, một hệ thống kiểm soát phiên bản như Git là cần thiết để đảm bảo không có xung đột mã giữa các nhà phát triển. Ngoài ra, các yêu cầu trong dự án thay đổi thường xuyên. Vì vậy, cần một hệ thống cho phép nhà phát triển quay lại phiên bản cũ hơn của mã.

### 1.4.2 Arduino IDE

Arduino là một nền tảng mã nguồn mở được sử dụng để xây dựng các dự án điện tử. Arduino bao gồm cả bảng mạch lập trình (thường được gọi là vi điều khiển) và một phần mềm ( IDE ) được sử dụng để lập trình viết và tải mã máy tính lên bo mạch. Nhờ sự đơn giản và dễ tiếp cận, Arduino đã được sử dụng trong hàng nghìn dự án và ứng dụng khác nhau. Phần mềm Arduino rất dễ sử dụng cho người mới bắt đầu, nhưng đủ linh hoạt cho người dùng nâng cao. Không giống như hầu hết các bo mạch lập trình trước đây, Arduino không cần phần cứng riêng để tải mã mới lên bo mạch - bạn có thể chỉ cần sử dụng cáp USB. Ngoài ra, Arduino IDE sử dụng

phiên bản đơn giản của C++, giúp việc học lập trình dễ dàng hơn.

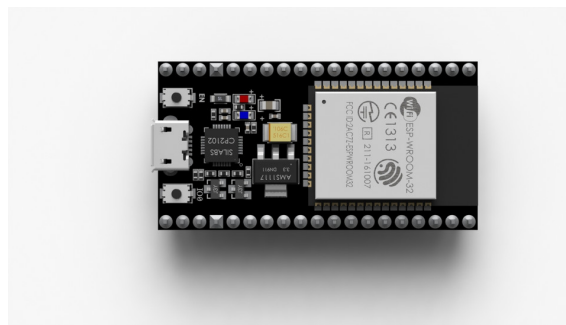
### 1.4.3 SIOT

SIoT là sự kết hợp của hai nền tảng chính bao gồm: SIoT Platform và SIoT Sphere. Các thiết bị nhúng trong SIoT sẽ được tích hợp bộ thư viện firmware SIoT Platform để chúng có thể kết nối tới hệ thống đám mây SIoT Sphere một cách nhanh chóng.

SIoT Platform là bộ thư viện firmware được tích hợp trong thiết bị nhúng gồm hai thành phần: (1) SIoT Interface: Gồm các hàm giao diện để triệu gọi các dịch vụ từ đám mây (2) SIoT Core: Là các dịch vụ hạ tầng cơ bản cho các thiết bị để khởi động, bảo trì, và kết thúc hoạt động.

## 1.5 Một số thiết bị phần cứng

### 1.5.1 Mạch Arduino NodeMCU ESP8266-12E



**Hình 1.1:** NodeMCU ESP8266-12E

Thu phát wifi ESP8266 12E là module wifi giá rẻ và được đánh giá rất cao cho các ứng dụng liên quan đến Internet và Wifi cũng như các ứng dụng truyền nhận sử dụng thay thế cho các module RF khác.

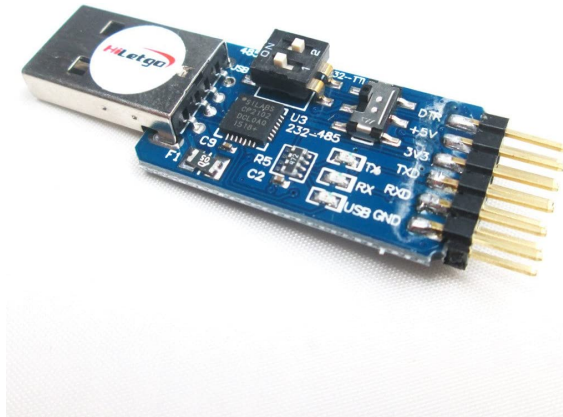
ESP8266 là một chip tích hợp cao, được thiết kế cho nhu cầu của một thế giới kết nối mới, thế giới Internet of thing (IOT). Nó cung cấp một giải pháp kết nối mạng Wi-Fi đầy đủ và khép kín, cho phép nó có thể lưu trữ các ứng dụng hoặc để giảm tải tất cả các chức năng kết nối mạng Wi-Fi từ một bộ xử lý ứng dụng.

Mạch thu phát wifi ESP8266 12E có khả năng xử lý và lưu trữ mạnh mẽ cho phép nó được tích hợp với các bộ cảm biến, vi điều khiển và các thiết bị ứng dụng cụ thể khác thông qua GPIOs với một chi phí tối thiểu và một PCB tối thiểu.

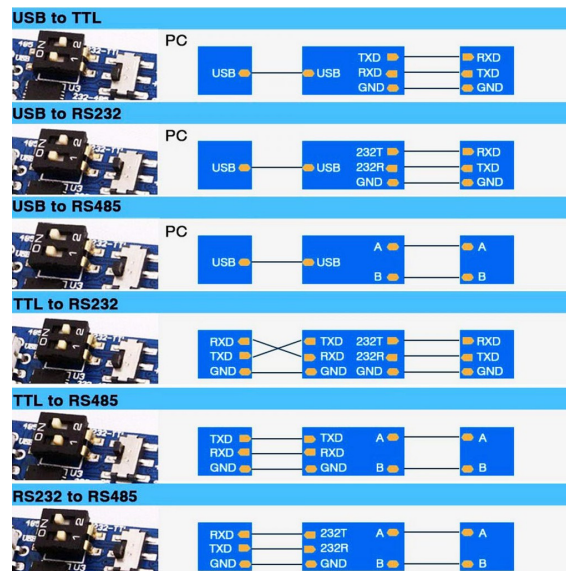
ESP8266 12E có kích thước nhỏ gọn, ra chân đầy đủ của IC ESP8266, mạch được thiết kế và gia công chất lượng tốt với vỏ bọc kim loại chống nhiễu và anten Wifi PCB tích hợp cho khoảng cách truyền xa và ổn định.



## 1.5.2 USB TO COM CP2102



**Hình 1.2:** USB COM CP2102



**Hình 1.3:** 6 mode hoạt động của USB COM CP2102

Giao tiếp COM sang TTL theo phương thức UART. CP2102 là chip USB to UART của Silabs. CP2102 TTL RS232/485 có kích thước nhỏ gọn và yêu cầu rất ít thành phần bên ngoài để hoạt động được ngay. USB to COM C2102 có những ưu điểm như sau: (i) Khả năng tương thích với điện thế 3.3V và 5V. (ii) Bảo vệ quá dòng: Cầu chì đặt trên bo mạch, ngay cả khi ngắn mạch tích cực và tiêu cực sẽ không làm hỏng bảnh mạch hoặc máy tính. Nếu có đoản mạch hoặc dòng điện quá 500mA, cầu chì sẽ tự động ngắt để bảo vệ bảnh mạch và máy tính. (iii) Giao diện TTL: có thể kết nối trực tiếp với các thiết bị nhúng khác như là STC, AVR, ARM, Atmega328... (iv) Giai diện RS232 và RS485. (v) Tốc độ truyền lên tới 2Mbps. Và cuối cùng, (vi) USB to COM CP2102 có 6 chế độ hoạt động. Do đó Module này được sử dụng rộng rãi hiện nay.

### 1.5.3 Máy đọc thẻ Reader



**Hình 1.4:** Máy đọc thẻ

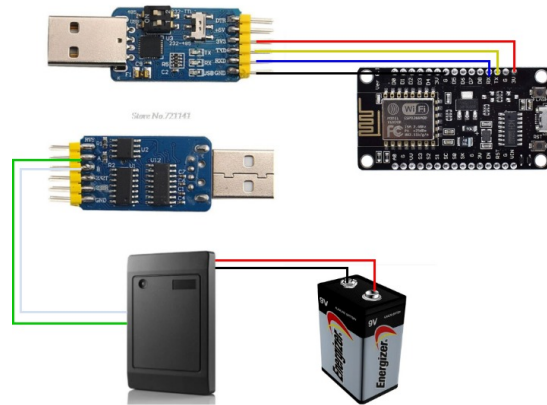
Thiết bị đọc thẻ này đọc được thẻ với các băng tần: 125KHz, 13.56MHz và sử dụng giao thức truyền xa RS485 giúp truyền số liệu đi xa.

## CHƯƠNG 2. KHẢO SÁT

### 2.1 Tổng quan chức năng

Phần 2.1 này nêu lên chức năng của dự án là đọc dữ liệu từ thẻ RFID 125 KHZ, thẻ NFC 13.56MHz, thẻ NFC NTAG215... và lưu trữ trên Sever SiOT.

### 2.2 Sơ đồ lắp mạch



**Hình 2.1:** Sơ đồ lắp mạch

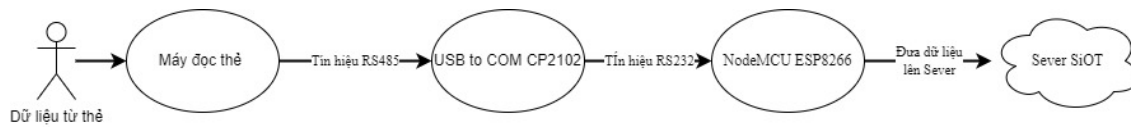
Mạch sẽ được lắp theo như sơ đồ như hình vẽ. Chi tiết cho từng thiết bị sẽ là như sau:

- Thứ nhất, USB to COM CP2102: (i) Chọn Mode TTL to 485. (ii) Chân A nối với dây Green (4R+), (iii) chân B nối với dây White (4R-), (iv) chân 3V3 nối với dây Red (lấy nguồn từ NodeMCU), (v) chân Gnd nối với dây Black (Tiếp đất từ NodeMCU), (vi) chân TXD, RXD nối với dây Yellow, Blue tương ứng trên NodeMCU.
- Thứ hai, NodeMCU ESP8266-E12: (i) chân TXD, RXD nối với dây Yellow, Blue tương ứng trên USB. (ii) chân 3V3 nối với dây Red chân Gnd nối với dây Black cung cấp năng lượng cho USB. (iii) Kết nối cổng USB type-B với 2 mục đích là cấp nguồn và nạp mã nguồn cho thiết bị.
- Thứ ba, Máy đọc thẻ: (i) Máy đọc thẻ sẽ được cấp nguồn 9V từ pin hoặc từ bộ nguồn Adapter, (ii) 2 dây Green, White lần lượt được nối vào chân A và chân B để truyền dữ liệu sang USB to COM CP2102.

Như vậy chúng ta đã hoàn thành việc lắp đặt các thiết bị phần cứng cho dự án này. Đây là phần quan trọng nhất của một hệ thống IoT, nó trực tiếp thực hiện các theo tác điều khiển và thu thập dữ liệu. Nhưng trong sự phát triển của nền công nghiệp 4.0 các thiết bị này sẽ ở một mức cao hơn đó chính là IoT - Internet of Things, mà

linh hồn của nó chính là kết nối với Internet chúng ta sẽ tìm hiểu nó trong phần 2.3 - Quy trình nghiệp vụ.

### 2.3 Quy trình nghiệp vụ



**Hình 2.2:** Quy trình nghiệp vụ của bài toán.

Quy trình này được thực hiện như sau: Người dùng sẽ dùng thẻ định danh của mình quét qua máy đọc thẻ, tín hiệu từ máy đọc thẻ truyền đi là tín hiệu RS485 có thể truyền đi xa được. Tín hiệu này được truyền vào USB to COM CP2102, tín hiệu ra là tín hiệu RS232 - tín hiệu mà NodeMCU ESP8266 có thể xử lý được. Tín hiệu RS232 này sẽ được xử lý tại NodeMCU ESP8266. Và như đã đề cập ở phần 2.2, linh hồn của một hệ thống IoT là kết nối với Internet. Bộ vi xử lý NodeMCU ESP8266 có tích hợp sẵn module Wifi, Trong quá trình cài đặt, em đã sử dụng thư viện SiOTCORE thực hiện kết nối với Internet, và dùng chính vi xử lý này để đưa dữ liệu lên trên sever SiOT. Tại đây chúng ta đã lưu trữ được dữ liệu và có thể xử lý, khai thác, áp dụng các thuật toán AI để phục vụ các hoạt động trong thực tế.

### 2.4 Yêu cầu phi chức năng

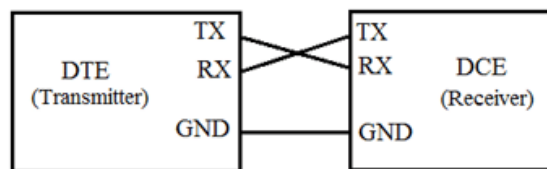
Các yêu cầu phi chức năng bao gồm độ tin cậy, bảo mật cao, dễ dùng, dễ bảo trì, dễ sửa chữa và thay thế, CSDL tối ưu, có khả năng tích hợp với các thiết bị bảo mật khác như Camera, thiết bị sinh trắc học, rào chắn... Và với yêu cầu cao hơn nữa là lấy dữ liệu từ thực tế để triển khai, huấn luyện mô hình AI để tạo ra một sản phẩm thông minh có khả năng thay thế cho con người.

## CHƯƠNG 3. CƠ SỞ LÝ THUYẾT

Trong phần này em sẽ đi tìm hiểu các giai thức truyền thông mà em sử dụng trong dự án này: RS232, RS485 và wifi. Chúng ta hãy cùng tìm hiểu những ưu điểm, nhược điểm, và ứng dụng nó vào thực tế để tạo ra những công cụ đặc lực nhằm nâng cao đời sống của con người.

### 3.1 Giao thức RS232

RS232 là một giao thức tiêu chuẩn được sử dụng cho giao tiếp nối tiếp, nó được sử dụng để kết nối máy tính và các thiết bị ngoại vi của nó để cho phép trao đổi dữ liệu nối tiếp giữa chúng. Vì nó thu được điện áp cho đường dẫn được sử dụng để trao đổi dữ liệu giữa các thiết bị. Nó được sử dụng trong giao tiếp nối tiếp lên đến 50 feet với tốc độ 1.492kbps. Theo định nghĩa của EIA, RS232 được sử dụng để kết nối Thiết bị truyền dữ liệu (DTE) và Thiết bị truyền dữ liệu (DCE) .



**Hình 3.1:** Giao thức RS232

Bộ thu và phát dữ liệu không đồng bộ đa năng (UART) được sử dụng cùng với RS232 để truyền dữ liệu giữa máy in và máy tính. Các bộ vi điều khiển không thể xử lý các mức điện áp như vậy, các đầu nối được kết nối giữa các tín hiệu RS232. Các đầu nối này được gọi là Đầu nối DB-9 như một cổng nối tiếp và chúng thuộc hai loại Đầu nối đực (DTE) Đầu nối cái (DCE).

#### 3.1.1 Cơ chế hoạt động

RS232 hoạt động trên giao tiếp hai chiều trao đổi dữ liệu với nhau. Có hai thiết bị được kết nối với nhau, Thiết bị truyền dữ liệu (DTE) Thiết bị truyền dữ liệu (DCE) có các chân như TXD, RXD và RTS CTS . Bây giờ, từ nguồn DTE , RTS tạo yêu cầu gửi dữ liệu. Sau đó, từ phía bên kia DCE , CTS, xóa đường dẫn để nhận dữ liệu. Sau khi xóa một đường dẫn, nó sẽ gửi tín hiệu đến RTS của nguồn DTE để gửi tín hiệu. Sau đó, các bit được truyền từ DTE đến DCE . Bây giờ một lần nữa từ DCE nguồn, yêu cầu có thể được tạo bởi RTS và CTS của nguồn DTE sẽ xóa đường dẫn nhận dữ liệu và đưa ra tín hiệu để gửi dữ liệu. Đây là toàn bộ quá trình thông qua đó việc truyền dữ liệu diễn ra.

### 3.1.2 Ưu - Nhược điểm

Giao thức RS232 sẽ ưu điểm như sau:

- RS232 phổ biến, dễ kiểm và chi phí rẻ
- Tương thích nhiều thiết bị
- Kết nối giao tiếp đơn giản
- Tốc độ truyền khá nhanh
- Khả năng chống nhiễu tốt
- Có thể tháo lắp nóng
- Có thể cấp nguồn cho thiết bị luôn

Bên cạnh các ưu điểm đó thì RS232 có những nhược điểm rất lớn.

- Một là tốc độ truyền dữ liệu. Dữ liệu có thể được chuyển ở mức khoảng 20 kbyte/s. Đó là khá chậm so với những gì mọi người đang sử dụng cho đến nay.
- Một vấn đề khác với RS232 là chiều dài tối đa của cáp là khoảng 15 mét. Điện trở dây và sụt điện áp trở thành một vấn đề với cáp dài hơn thế này. Đây là một lý do khiến RS232 không được sử dụng nhiều để kéo đi xa.

## 3.2 Giao thức RS485

RS485 hay được biết đến với tên gọi đầy đủ là chuẩn giao tiếp RS485 hay cáp RS485, đây là phương thức giao tiếp kết nối với máy tính và các thiết bị khác. RS485 không chỉ đơn thuần là giao diện đơn lẻ mà nó chính là tổ hợp truyền thông có khả năng tạo ra các mạng đơn giản của nhiều thiết bị.

Chuẩn giao tiếp RS485 có thể kết nối max lên đến 32 thiết bị trên một cặp dây đơn và một hệ thống dây nối đất ở khoảng cách lên đến 1200m.

### 3.2.1 Cơ chế hoạt động

Cáp RS485 được cấu tạo rất đơn giản, chỉ từ các sợi dây được xoắn lại với nhau theo từng cặp. Tuy nhiên, chính cấu tạo này lại sinh ra một nhược điểm nghiêm trọng, khi hiện tượng nhiễu xuất hiện ở 1 cặp dây thì ngay lập tức cặp dây khác cũng sẽ bị. Điều này dẫn đến điện áp hoạt động giữa 2 dây sẽ không có quá nhiều sự chênh lệch, bộ phận thu của RS485 vẫn có thể nhận được tín hiệu vì bộ thu đã loại bỏ hết được hiện tượng nhiễu.

Nguyên lý hoạt động của RS485 khá đơn giản, dữ liệu sẽ được truyền qua 2 dây khi xoắn lại với nhau, dây này được gọi là cáp xoắn. Khi dây được xoắn lại sẽ tạo cho RS485 khả năng chống nhiễu cao và khả năng truyền tín hiệu đường dài tốt hơn.

### 3.2.2 Ưu - Nhược điểm

Tuy giao thức RS232 phổ biến nhưng lại mắc một số nhược điểm như tốc độ truyền thấp, tín hiệu sẽ bị suy hao nhanh trong quá trình truyền. Do đó RS485 ra đời để khắc phục những yếu điểm mà RS232 để lại

- Cáp RS485 là chuẩn giao tiếp duy nhất có thể kết nối cùng lúc nhiều máy phát và máy thu trên cùng một hệ thống mạng.
- Những máy thu có điện trở đầu vào lên đến 12k thì RS485 vẫn có thể kết nối lên 32 thiết bị. Ngoài ra, với các đầu vào khác, RS485 có thể kết nối tối đa lên 256 thiết bị.
- Khi RS485 đang kết nối các thiết bị ở khoảng cách khá xa thì người sử dụng có thể khắc phục bằng cách lắp thêm bộ lặp để tăng số lượng thiết bị kết nối, giúp tín hiệu ổn định hơn, tránh nhiễu đường truyền. RS485 có lắp đặt 2 dây truyền tín hiệu nên tín hiệu sẽ được truyền đi nhanh hơn trên khoảng cách xa và rộng hơn

### 3.3 Wifi

Wifi là viết tắt của Wireless Fidelity được hiểu là sử dụng sóng vô tuyến để truyền tín hiệu. Loại sóng này có khả năng kết nối với mạng khác hay với máy tính bằng sóng vô tuyến. Nó tương tự như sóng điện thoại, truyền hình, radio. Hầu như Các thiết bị điện tử như: máy tính, laptop, điện thoại, tablet... đều có thể truy cập wifi để truy cập mạng thỏa mãn nhu cầu của người dùng. Kết nối wifi hiện nay được dựa trên chuẩn liên kết IEEE 802.11 và chạy trên băng tần 54Mbps với quy mô phát tín hiệu trong vòng 30m với điều kiện không có vật cản. Nếu có vật cản, tốc độ truyền dẫn tín hiệu wifi sẽ bị kém đi nhiều.

#### 3.3.1 Ưu điểm của Wifi

- Trong những năm gần đây, wifi đã có những bước phát triển vượt bậc với chuẩn 802.11n có tốc độ 150Mb/s hoặc 802.11ac có tốc độ lên đến 866.7Mb/s. Wifi 6 có tốc độ lý thuyết là khoảng 9,6 Gb/s.
- Sử dụng môi trường lan truyền là sóng điện từ nên có không cần phải kết nối dây lằng nhằng. Giúp thuận tiện trong việc bố trí và sắp xếp.

Như vậy chúng ta đã tìm hiểu xong 3 giao thức truyền thông trên. Dựa vào tình hình thực tế mà chúng ta sẽ lựa công nghệ tùy thuộc vào từng trường hợp. RS232 được dùng trong trường hợp giá thành rẻ, không yêu cầu về chất lượng và khoảng cách truyền. RS485 sẽ được dùng trong yêu cầu truyền dữ liệu đi xa còn Wifi sẽ dùng trong môi trường có Wifi (kết nối Internet) để truyền dữ liệu.

## CHƯƠNG 4. KẾT NỐI, CẬP NHẬT DỮ LIỆU LÊN SIOT SEVER

### 4.1 Cài đặt chương trình

Chi tiết mã nguồn có tại link github: [https://github.com/mxngocqb/Reader\\_Device.git](https://github.com/mxngocqb/Reader_Device.git)

### 4.2 Tạo thiết bị ảo website `siot.soict.ai`

Sever SiOT cho phép chúng ta tạo các thiết bị ảo với các thuộc tính. Trình tự thực hiện lấy ID của thiết bị như sau:

- Bước 1: Đăng ký tài khoản miễn phí ở web `siot.soict.ai`.
- Bước 2: Trong phần **My devices** chọn **Register New Device** để đăng kí một thiết bị ảo mới.
- Bước 3: Trong phần **Attributes**, tạo những thuộc tính muốn lưu trữ.
- Bước 4: Trong phần **General** lấy các ID ở *Bearer token* của phần *You rdevie*, lấy các URL *HTTP Endpoint* ở phần *Attribute API*.

The screenshot displays the 'Your device' section of the siot.soict.ai interface. It lists various attributes for a device named 'READER\_RS232\_TO\_RS485'. The 'Bearer token' is highlighted with a red circle. Below this, the 'Attribute API' section is shown, where the 'HTTP Endpoint' is also highlighted with a red circle. The endpoint is '/api/devices/reader\_rs232\_to\_rs485-2wunb/attributes/id-52a5e'. Other fields include 'Rest method' (Post), 'HTTP Header Format', 'HTTP Body Format', and 'Value type' (string). A 'Delete' button is visible at the bottom.

Your device	
Device name	READER_RS232_TO_RS485
Device ID	1f8a716f-5a74-489c-81ca-c04b7184ee84
Slug API key	reader_rs232_to_rs485-2wunb
<b>Bearer token</b>	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiJkYzE5Mi0yMTVLTjOTYyYWM1Ni02OGY3NTU5NzBjZyYiLCJpYXQiOiJlZ2NjgkMzk3Mzh9.S_Psblgdf60nGfp-R46qbGAvaTDxnXLC-YFwN1cR0IE
Current Platform Version	Unknown <a href="#">View release list</a>
Current Platform Version Key	Unknown
Allow auto Update	<input type="checkbox"/> (This feature is only available when the Platform Version is installed on embedded device)
Device image	

Attribute API	
Attribute: Id	
Rest method:	Post
<b>HTTP Endpoint:</b>	/api/devices/reader_rs232_to_rs485-2wunb/attributes/id-52a5e
HTTP Header Format:	'Platform-Version': {Current Platform Version Key} (Optional) 'Content-Type': application/json 'Authorization': Bearer {Bearer token}
HTTP Body Format:	{ value={Attribute value} }
Value type:	string

[Delete READER\\_RS232\\_TO\\_RS485](#)

Hình 4.1: Tạo thiết bị ảo



- Bước 5: Dán vào trong file *Allinfo.h* trong thư mục *src* ấn save.

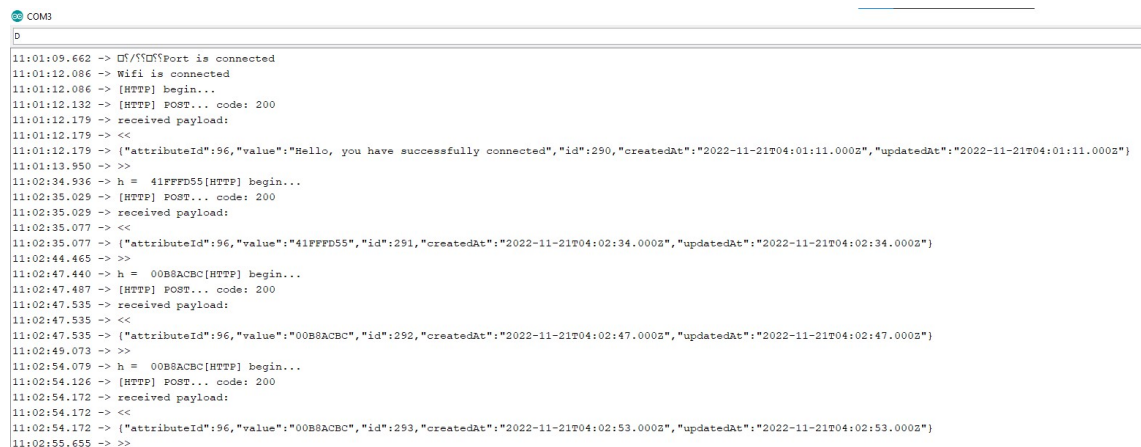
Như vậy chúng ta đã hoàn thành việc tạo một thiết bị ảo và sẵn sàng cho việc đẩy dữ liệu từ NodeMCU ESP8266 lên sever SiOT.

### 4.3 Kết nối sever

Trong thư viện *siotcore.h* có sẵn hàm *.init()* chúng ta chỉ cần gọi hàm này và sau khi nạp mã nguồn vào Board. Board sẽ có khả năng kết nối với wifi. Chúng ta sẽ kết nối wifi và NodeMCU ESP8266 sẽ tự động kết nối với sever SiOT.

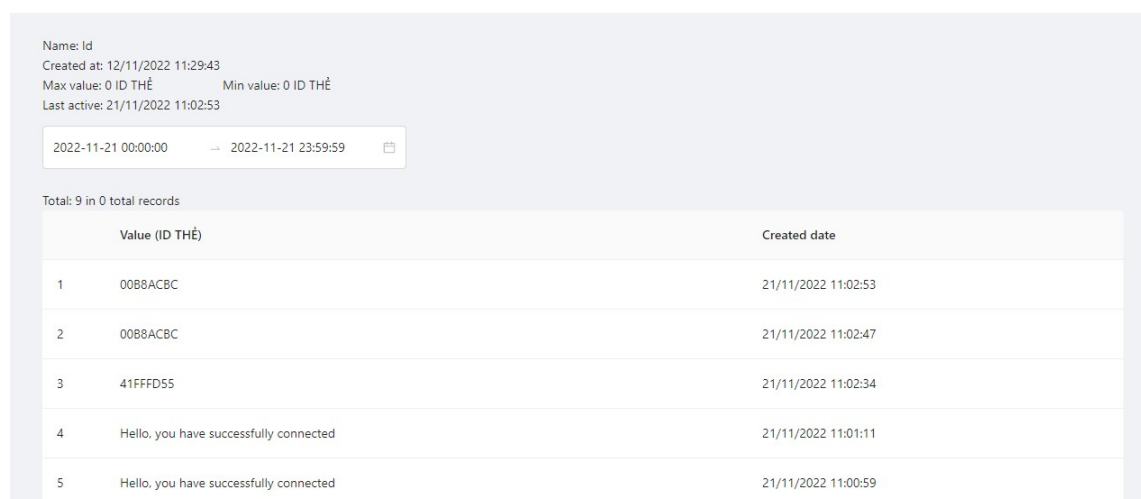
### 4.4 Kết quả thu được

Kết quả hiển thị khi đọc dữ liệu qua cổng COM:



```
COM3
D
11:01:09.662 -> [SIO]Port is connected
11:01:12.086 -> Wifi is connected
11:01:12.086 -> [HTTP] begin...
11:01:12.132 -> [HTTP] POST... code: 200
11:01:12.179 -> received payload:
11:01:12.179 -> <<
11:01:12.179 -> {"attributeId":96,"value":"Hello, you have successfully connected","id":290,"createdAt":"2022-11-21T04:01:11.000Z","updatedAt":"2022-11-21T04:01:11.000Z"}
11:01:13.950 -> >>
11:02:34.936 -> h = 41FFFD55[HTTP] begin...
11:02:35.029 -> [HTTP] POST... code: 200
11:02:35.029 -> received payload:
11:02:35.077 -> <<
11:02:35.077 -> {"attributeId":96,"value":"41FFFD55","id":291,"createdAt":"2022-11-21T04:02:34.000Z","updatedAt":"2022-11-21T04:02:34.000Z"}
11:02:44.465 -> >>
11:02:47.440 -> h = 00B8ACBC[HTTP] begin...
11:02:47.487 -> [HTTP] POST... code: 200
11:02:47.535 -> received payload:
11:02:47.535 -> <<
11:02:47.535 -> {"attributeId":96,"value":"00B8ACBC","id":292,"createdAt":"2022-11-21T04:02:47.000Z","updatedAt":"2022-11-21T04:02:47.000Z"}
11:02:49.073 -> >>
11:02:54.079 -> h = 00B8ACBC[HTTP] begin...
11:02:54.126 -> [HTTP] POST... code: 200
11:02:54.172 -> received payload:
11:02:54.172 -> <<
11:02:54.172 -> {"attributeId":96,"value":"00B8ACBC","id":293,"createdAt":"2022-11-21T04:02:53.000Z","updatedAt":"2022-11-21T04:02:53.000Z"}
11:02:55.655 -> >>
```

Hình 4.2: Dữ liệu từ cổng COM



Name: Id	
Created at: 12/11/2022 11:29:43	
Max value: 0 ID THỂ	Min value: 0 ID THỂ
Last active: 21/11/2022 11:02:53	
2022-11-21 00:00:00 → 2022-11-21 23:59:59	
Total: 9 in 0 total records	
Value (ID THỂ)	Created date
1 00B8ACBC	21/11/2022 11:02:53
2 00B8ACBC	21/11/2022 11:02:47
3 41FFFD55	21/11/2022 11:02:34
4 Hello, you have successfully connected	21/11/2022 11:01:11
5 Hello, you have successfully connected	21/11/2022 11:00:59

Hình 4.3: Dữ liệu từ web

Giải thích:

- Port is connected: cổng COM đã được kết nối
- Wifi is connected: Wifi đã được kết nối, đồng thời sẽ truyền một chuỗi dữ liệu

lên web : Hello, you have successfully connecte"

- Received payload: Dữ liệu mà web nhận được

## CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

### 5.1 Kết luận

Đề tài đã được hoàn thành và giải quyết được những vấn đề giáo viên hướng dẫn đã giao, ưu điểm: (i) Thực hiện kết nối, cập nhật dữ liệu thành công với SiOT Sever. (ii) Cung cấp giải pháp đa kết nối từ xa. (iii) Cung cấp khả năng mở rộng, tương thích với nhiều module khác.

Bên cạnh những thành quả đạt được, dự án này còn một số thiếu sót: (i) Chỉ hỗ trợ kết nối Wifi, không hỗ trợ kết nối 4G, (ii) Vấn đề về tối ưu năng lượng cho máy đọc thẻ.

### 5.2 Một số lỗi trong quá trình thực hiện

Vì là còn thiếu các kỹ năng, kinh nghiệm trong lĩnh vực nên trong quá trình thực hiện dự án. Em đã nghiên cứu, tìm tòi và đã rút ra được một số bài học như sau:

- Lần đầu tiên khi đọc dữ liệu từ USB bằng C. Em chỉ đọc được một chuỗi ký tự rất lạ. Sau đó em tìm ra nguyên nhân là mình bị nhầm thứ tự của dây. Thứ tự đúng là dây Green, White của máy đọc thẻ sẽ lần lượt cắm vào chân A và chân B của USB to COM CP 2102
- Máy tính hiện tên thiết bị (cả NodeMCU ESP8266 và USB to COM CP2102): Nguyên nhân là máy tính thiếu bộ driver CP210x. Tải bộ driver ở <https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads> là sẽ khắc phục được lỗi.
- Lỗi thiếu một số thư viện và đã số những bạn mới hay gặp phải: (i) Thư viện *CircularBuffer.h*, link tải <https://github.com/rlogiacco/CircularBuffer>, (ii) thư viện *PubsubClient.h*, link tải <https://www.arduino.cc/reference/en/libraries/pubsubclient/>. Sau khi tải về thành công 2 thư viện trên, truy cập vào *sketch* → *Include Library* → *Add.ZIP Library* rồi chọn 2 tệp ZIP vừa tải về và ấn *open* là cài đặt thành công.
- Chuỗi RxBuffer có chứa một vài ký tự đặc biệt nên ảnh hưởng tới quá trình đưa dữ liệu lên Sever. Em khắc phục bằng cách lấy một SubString từ ký tự 1 → 9 chỉ lấy một mình dữ liệu thực sự là ID của thẻ. Sở dĩ chỉ lấy từ 1 → 9 bởi vì dữ liệu của thẻ là 8 chữ số Hexa và bị ảnh hưởng bởi 1 ký tự đặc biệt ban đầu.

```

22:39:15.556 -> >>
22:39:34.542 -> h = 41FFFD55
22:39:34.585 -> [HTTP] begin...
22:39:37.548 -> [HTTP] POST... code: 400
22:39:52.568 -> h = 41FFFD55
22:39:52.568 -> [HTTP] begin...
22:39:52.883 -> [HTTP] POST... code: 400

```

**Hình 5.1:** Lỗi trong quá trình đưa dữ liệu lên Sever

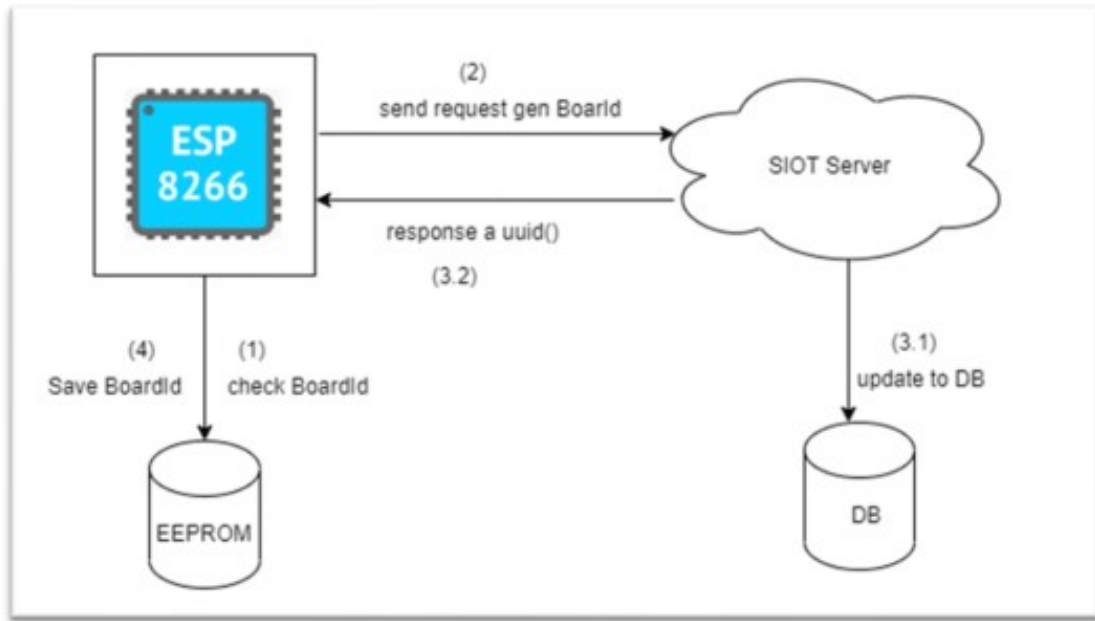
```

23:39:51.153 -> h = 41FFFD55[HTTP] begin...
23:39:51.200 -> [HTTP] POST... code: 200
23:39:51.247 -> received payload:
23:39:51.247 -> <<
23:39:51.247 -> {"attributeId":96,"value":"41FFFD55","id":274,"createdAt":"2022-11-17T16:39:51.000Z","updatedAt":"2022-11-17T16:39:51.000Z"}
23:39:52.790 -> >>
23:39:59.762 -> h = 41FFFD55[HTTP] begin...
23:39:59.809 -> [HTTP] POST... code: 200
23:39:59.856 -> received payload:
23:39:59.856 -> <<
23:39:59.856 -> {"attributeId":96,"value":"41FFFD55","id":275,"createdAt":"2022-11-17T16:39:59.000Z","updatedAt":"2022-11-17T16:39:59.000Z"}
23:40:01.588 -> >>

```

**Hình 5.2:** Sau khi fix lỗi. Dữ liệu đã được đưa lên sever thành công

- Ban đầu em không không thấy thiết bị hiện ra thiết bị Wifi. Để giải thích lỗi này, em đã tìm hiểu cấu tạo của vi xử lý bên trong NodeMCU ESP8266 - E12:  
*Thứ (i)* Vi xử lý này tích hợp sẵn RAM trong chip. *Thứ (ii)* tích hợp bộ nhớ lệnh xử định công nghệ Flash, Đây là vùng chứa code lập trình được nạp vào. *Thứ (iii)* Tích hợp sẵn bộ nhớ EEPROM, là vùng nhớ dữ liệu tự do. Khi tắt điện, nó vẫn chứa và lưu trữ dữ liệu, vì vậy dữ liệu cấu hình được lưu trữ tại đây. Trong trường hợp này thông tin về mạng wifi và mật khẩu sẽ được lưu trong EEPROM nên trong trường hợp wifi đã được lưu trước đó, khi mà chúng ta khởi động lại thì dữ liệu cũng không bị mất đi. Vì thế mà có lúc em không thấy hiện lên cấu hình wifi.



**Hình 5.3:** Cấu hình wifi

### 5.3 Hướng phát triển

Khi nhận đề tài này em thấy nó ứng dụng luôn vào thực tế, có thể làm máy chấm công, thiết bị gửi xe tự động,... Ngoài ra có thể dùng trong giáo dục, giúp các em học sinh cấp 2, cấp 3 có thể tìm tòi, nghiên cứu để lập trình những dự án thực tế với thiếu bị, cảm biến, động cơ,... thay vì việc học những thuật toán khô khan trên giấy.

[1]–[6]

## TÀI LIỆU THAM KHẢO

- [1] Đồng Quang Linh, *Xây dựng bộ firmware tiêu chuẩn và các công cụ hỗ trợ siot platform cho hệ thống nhúng*, 2021.
- [2] N. N. Thành, *Xây dựng hệ quản trị dữ liệu tập trung siot sphere cho hệ thống nhúng*, 2020.
- [3] Beat Your Bit, *Standalone atmega328 configuration*. [Online]. Available: [https://beatyourbit.com/standalone\\_atmega328/](https://beatyourbit.com/standalone_atmega328/).
- [4] Micheal, *F() macro and progmem*. [Online]. Available: [https://forum.arduino.cc/t/f\\_macro\\_and\\_progmem/107941](https://forum.arduino.cc/t/f_macro_and_progmem/107941).
- [5] P. D. Shenouda Dawoud, *Serial Communication Protocols and Standards RS232/485, UART/USART, SPI, USB, INSTEON, Wi-Fi and WiMAX*. River Publishers, 2020.
- [6] .