**THIẾT KẾ MÔ HÌNH GIÁM SÁT PHƯƠNG TIỆN RA VÀO CHUNG CƯ**

**CHƯƠNG 2: CƠ SỞ LÍ THUYẾT**

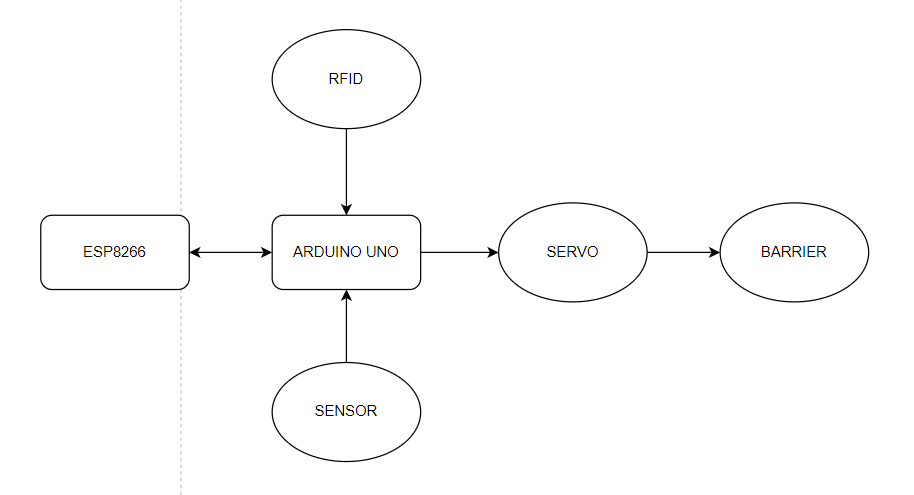
1. ***Thiết bị.***

* Thẻ từ: chứa các thông tin về phương tiện, chủ phương tiện.
* Barrier: sau khi xác thực được phương tiện sẽ được servo điều khiển để tự động đóng mở.
* Camera biển số đọc và ghi nhận thông tin về biển số xe.
* Camera toàn cảnh kiểm soát thông tin loại xe, người điều khiển, tổng thể khu vực ra vào trạm kiểm soát.
* Servo: điều khiển rào chắn.
* Module RFID để đọc thẻ.
* Nguồn cấp cho hệ thống.
* Cảm biến hồng ngoại phát hiện xe đã đi qua barrier và phát tín hiệu để đóng barrier.
* Gờ giảm tốc.

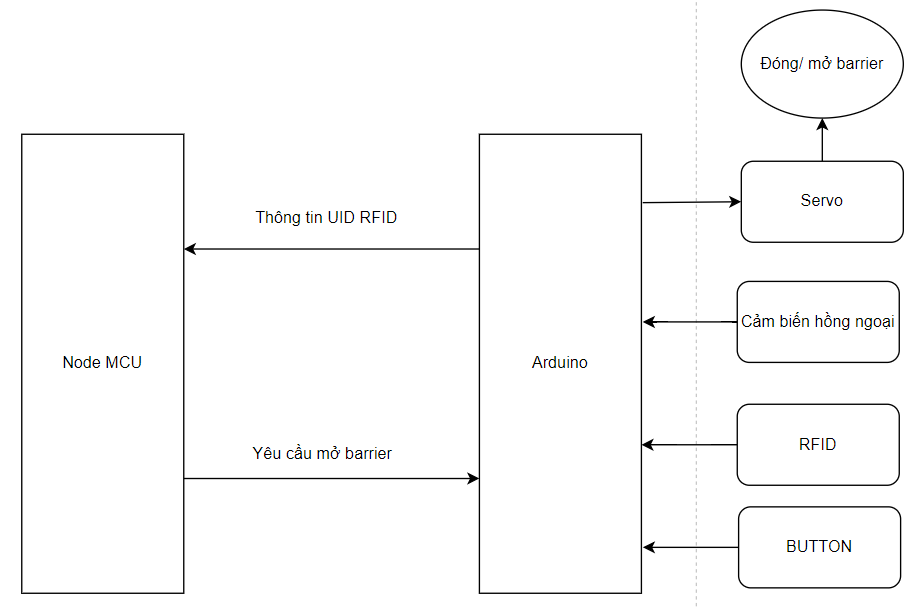
1. ***Cách thức hoạt động.***

* Giám sát phương tiện ra vào chung cư, ta chia làm 3 trường hợp sau:
* Trường hợp 1: đối tượng là chủ phương tiện: đối tượng quẹt thẻ cư dân và dữ liệu được lưu vào hệ thống.
* Trường hợp 2: đối tượng là người quen của chủ phương tiện và là người trong chung cư: khi quẹt thẻ, hệ thống sẽ lưu dữ liệu id thẻ cư dân của đối tượng và biển số xe vào hệ thống.
* Trường hợp 3: đối tượng là người bên ngoài chung cư: đối tượng được nhân viên phát một thẻ trắng. Hệ thống ghi lại dữ liệu biển số xe, id thẻ trắng và khuôn mặt đối tượng. Tại lối ra, nhân viên sẽ kiểm tra biển số xe, id thẻ và khuôn mặt của đối tượng là trùng khớp thì sẽ mở barrier.
* Sau khi quẹt thẻ xong, barrier sẽ mở cho đối tượng ra/ vào.
* Gờ giảm tốc để giảm tốc độ của chủ phương tiện khi đến bãi đậu xe.
* Ta có cảm biến hồng ngoại nhận biết hoạt động xe đi qua barrier và gửi yêu cầu đóng barrier.

1. ***Mô hình.***
   1. ***Sơ đồ hệ thống phần cứng.***

******

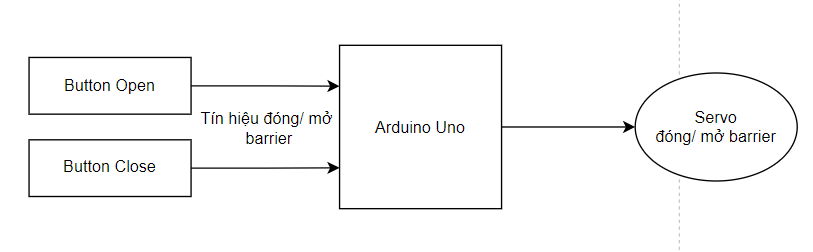
* 1. ***Sơ đồ luồng dữ liệu mô hình.***

******

1. ***Các chế độ hoạt động.***

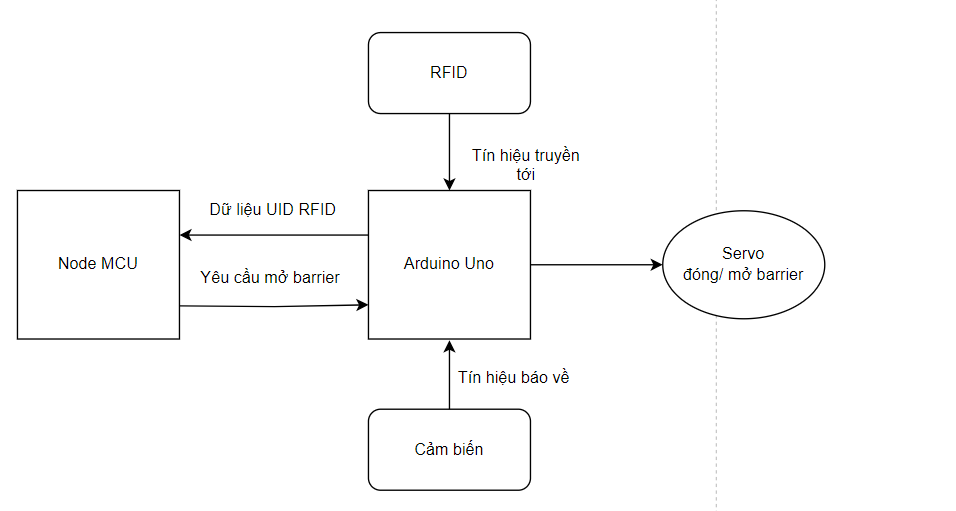
***4.1 Chế độ thủ công.***

Đây là chế độ người dùng sẽ sử dụng các nút button để điều khiển đóng/ mở barrier. Sau khi Button Open hoặc Button Close được nhấn sẽ gửi tín hiệu đóng/ mở barrier tới arduino và điều khiển servo hoạt động.

******

* 1. ***Chế độ tự động.***

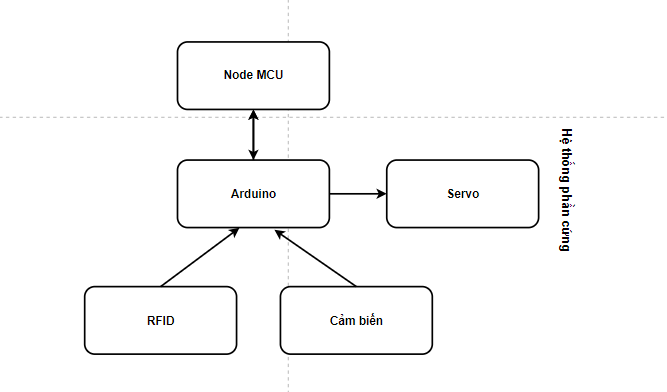
Sau khi người dùng quẹt thẻ, thông tin về UID của thẻ được truyền tới Arduino Uno, sau đó sẽ được gửi lên Node MCU và được kiểm tra với thông tin dữ liệu UID trên server. Nếu UID của thẻ có trong list dữ liệu của server thì lúc này sẽ có yêu cầu truyền xuống điều khiển servo mở barrier. Cảm biến hồng ngoại được sử dụng để nhận biết đối tượng đã đi qua barrier hay chưa. Nếu nhận thấy đối tượng, lúc này Arduino Uno sẽ điều khiển servo thực hiện đóng barrier.

******

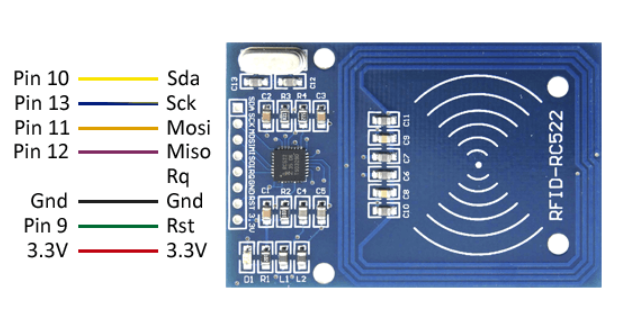
***CHƯƠNG 1: GIỚI THIỆU BÀI TOÁN***

***1.2 System model bài toán***

***Sơ đồ toàn bộ hệ thống.***

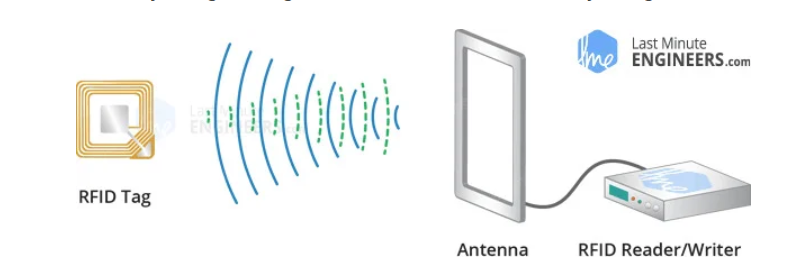
******

1. ***.Hệ thống bao gồm các module chính.***
2. ***Embedded system.***
3. ***Module RFID.***



***Nguyên lí hoạt động:***

RFID là viết tắt của **“Radio Frequency Identification”**, có nghĩa là nhận dạng bằng tần số radio. Đây là một công nghệ sử dụng sóng radio để truyền và nhận dạng thông tin từ một thẻ hoặc một thiết bị RFID. Hệ thống RFID bao gồm ít nhất hai thành phần chính: một thẻ RFID (hay còn gọi là tag RFID) và một đầu đọc RFID (hay còn gọi là reader RFID).



Thẻ RFID thường được gắn vật lên vật thể, nó chứa một anten để thu phát sóng radio và một chip chứa thông tin cần được nhận dạng. Đầu đọc RFID sử dụng sóng radio để truyền đi và nhận lại thông tin từ thẻ RFID. Khi thẻ RFID được đặt trong phạm vi hoạt động của đầu đọc RFID, thông tin từ thẻ sẽ được truyền đến đầu đọc RFID để xử lý và xác định thông tin nhận dạng.

***Tổng quan phần cứng:***

Module RFID RC522 là một module sử dụng công nghệ RFID để đọc và ghi thông tin từ các thẻ RFID.

Module **RFID Arduino** bao gồm một chip đọc RFID MFRC522 và các thành phần khác như anten và các linh kiện điện tử. Chip MFRC522 có khả năng đọc và ghi dữ liệu từ các thẻ RFID tiêu chuẩn, bao gồm các loại thẻ MIFARE và các loại thẻ tương thích.

Module RFID RC522 kết nối với Arduino thông qua [giao tiếp SPI](https://arduinokit.vn/chuan-giao-tiep-spi-la-gi/) (Serial Peripheral Interface) hoặc I2C và [**UART**](https://arduinokit.vn/giao-tiep-arduino-voi-may-tinh-thong-qua-serial-uart/)**.**

Để sử dụng module RFID RC522, cần có thư viện hỗ trợ cho Arduino như **thư viện MFRC522,** giúp dễ dàng giao tiếp và điều khiển module. Thông qua thư viện này, bạn có thể đọc thông tin từ thẻ RFID, xác minh chúng và thực hiện các hoạt động như đọc mã số, ghi dữ liệu, kiểm tra trạng thái và điều khiển anten.

***Các đặc tính RC522:***

RFID RC522 sử dụng cảm ứng điện từ để kích hoạt thẻ và có tần số 13,56 MHz để truyền dữ liệu.

Thẻ RFID có thể sử dụng được với cả hai mặt của module khoảng cách tối đa 5cm.

Điện áp 3.3V được yêu cầu để hoạt động.

Chế độ ngủ tự động giúp module tiêu thụ ít điện năng hơn.

Module có ba loại giao tiếp( UART, SPI, I2C). Do đó, có thẻ sử dụng được với hầu hết mọi vi điều khiển.

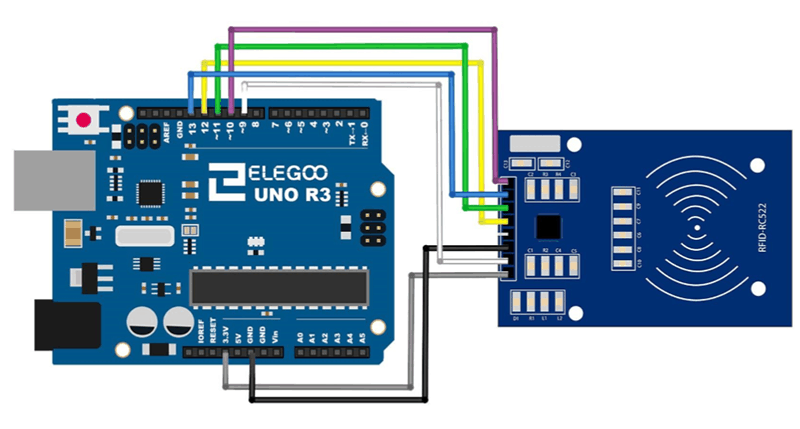
Có thể truyền dữ liệu lên đến 10Mb/ s.

***Cách sử dụng RFID RC522:***

Việc sử dụng RC522 vừa đơn giản vừa phức tạp. Ngay cả thư viện hỗ trợ của nó cũng có quá nhiều phức tạp để sử dụng. Trước tiên, hãy hiểu rằng các ví dụ và thư viện hỗ trợ đều cho giao thức SPI nhưng cùng một thư viện có thể sử dụng được cho các giao tiếp UART và I2C Serial khác.

*Ví dụ giao tiếp với Arduino:*

Để sử dụng RC522 với Arduino, bảng mạch phổ biến nhất như sau:



Trong sơ đồ mạch trên, RFID kết nối với Arduino Uno thông qua các chân SPI cụ thể của nó.

Sau khi thiết kế mạch, cần khai báo hai thư viện sau:



Thư viện MFRC522 giúp giải mã và mã hóa dữ liệu từ module RFID và SPI.h sẽ giúp thiết lập giao tiếp SPI. Hai thư viện này phụ thuộc vào nhau. Sau khi khởi tạo các thư viện, cần khai báo chân reset và chọn SS thông qua lệnh sau:

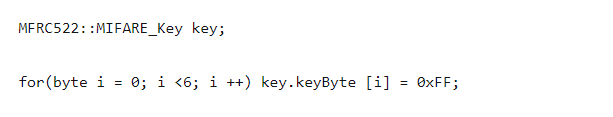


Thay đổi SS\_PIN thành chân điều khiển SS của RFID tương ứng trên arduino và RST\_PIN thành chân điều khiển chân reset của RFID. Bây giờ phải khai báo phần khởi tạo giao tiếp SPI và phần khởi tạo module cần được thực hiện qua 2 lệnh sau:



Khi viết 2 lệnh này việc khởi tạo giao tiếp các module sẽ hoàn tất nhưng phần trên hơi phức tạp. Thư viện RC522 cung cấp nhiều lệnh nhưng để sử dụng các lệnh đó thì phải hiểu một số chức năng.

Trong RC522, thẻ RFID có thể đọc được khi có thẻ đặt gần module. Khi một thẻ được đọc thì module sẽ lưu mã trừ khi nó bị sai. Lệnh sau sẽ tạo mã mặc định cho module:

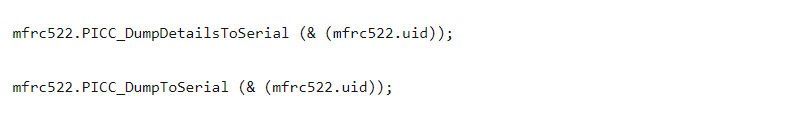


MIFARE\_Key sẽ tự động ghi vào module và vòng lặp sẽ giúp lưu mã.

Khi các giá trị mặc định của thẻ được thiết lập, các lệnh sau sẽ cho phép phát hiện thẻ mới:

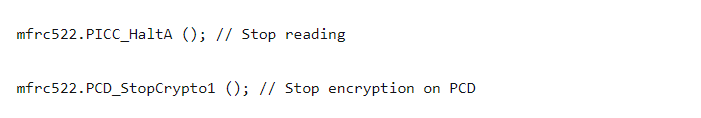


Lệnh đầu tiên là để phát hiện thẻ và lệnh tiếp theo là đọc mã thẻ. Đôi khi thẻ có thể ở gần module nhưng nó không có dữ liệu mã. Do đó, lệnh thứ hai ReadCardSerial sẽ giúp phát hiện điều đó. Sau đó, sử dụng lệnh sau để xem dữ liệu để có thể giải mã:



Sử dụng lệnh đầu tiên cho hệ thập phân và lệnh hai cho hệ HEX. Giúp lưu trữ và đọc mã dữ liệu. Luôn nhớ rằng thiết bị có thể đọc và xuất/ giải mã dữ liệu của 1 thẻ tại một thời điểm. Do đó, trong quá trình xử lí, thiết bị sẽ không thể đọc thẻ mới.

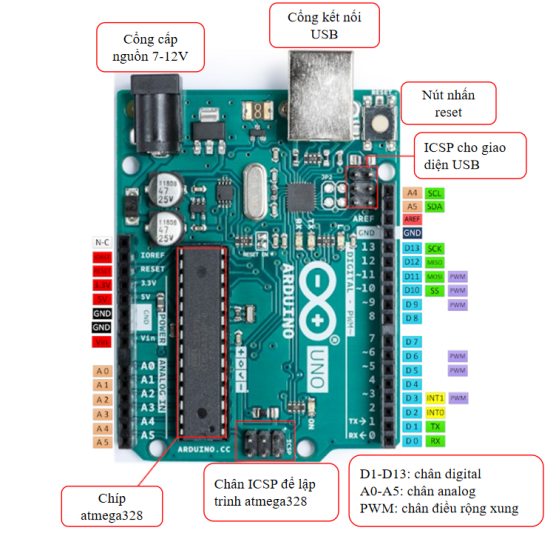
Khi thẻ đọc các lệnh, nó sẽ không bao giờ dừng lại cho đến khi có lệnh dừng được gửi đi. Để dừng mã hóa tín hiệu, các lệnh cần được gửi từ bộ vi điều khiển. Do đó, các lệnh dừng cho cả hai chức năng đọc và mã hóa sẽ là:



1. ***Arduino Uno R3.***

[Arduino](https://mesidas.com/arduino-la-gi/) 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.

***Sơ đồ chân của Arduino Uno R3:***



*Chân cấp năng lượng:*

* 5V: cấp điện áp 5V đầu ra, dùng để cấp nguồn cho các linh kiện điện tử kết nối với Arduino.
* 3.3V: chức năng tương tự như cấp nguồn 5V nhưng đây là cấp điện áp 3.3V đầu ra.
* Ground: hay còn gọi là chân GND, là cực âm của nguồn điện cấp cho Arduino Uno. Khi bạn dùng các thiết bị sử dụng những nguồn điện riêng biệt thì những chân này phải được nối với nhau.
* Vin( Voltage Input): tương tự như chân 5V, nhưng thêm chức năng cấp nguồn ngoài cho Arduino Uno thay vì cắm USB, bạn nối cực dương của nguồn với chân này và cực âm của nguồn với chân GND.

*Các cổng vào/ ra( I/ O):*

Arduino cung cấp nhiều các chân I/ O( hay còn gọi là Pin) để ta giao tiếp hay gửi lệnh điều khiển các thiết bị, dưới đây là sẽ nói về các chân sử dụng nhiều nhất và phân chúng làm các loại như sau:

*Các chân Digital:*

Arduino Uno R3 được sở hữu 14 chân digital từ 0 đến 13 dùng để đọc hoặc xuất tín hiệu. Chúng chỉ có 2 mức điện áp có thể điều khiển là 0V và 5V với dòng vào/ ra tối đa trên mỗi chân là 40mA. Ngoài ra một số chân digital có chức năng đặc biệt là chân PWM.

* Chân PWM: là các chân có dấu ‘~’ đằng trước, các chân này cho phép bạn xuất ra xung PWM với độ phân giải 8 bti( giá trị từ 0 đến 255) tương ứng với mức giao động điện áp của chân từ 0V đến 5V, khác với các chân không phải PWM, chỉ có thể chọn giá trị 0V hoặc 5V.

*Các chân Analog:*

Arduino Uno có 6 chân analog( A0 đến A5) cung cấp độ phân giải tín hiệu 10 bit( 0 đến 1023) để đọc giá trị điện áp trong khoảng 0V đến 5V.

Arduino Uno có 2 chân A4( SDA) và A5( SCL) để hỗ trợ giao tiếp I2C/ TWI với các thiết bị khác.

*Chân TXD và RXD:*

Đây là các chân Serial dùng để gửi( transmit- TX) và nhận( receive- RX) dữ liệu TTL Serial. Arduino Uno có thể giao tiếp dữ liệu với các thiết bị cần sử dụng thông qua 2 chân này, ngoài ra có thể sử dụng 2 chân này để nạp code cho mạch mac không cần thông qua USB của mạch.

***Thông số cơ bản:***

|  |  |
| --- | --- |
| Vi điều khiển | Atmega 328 (họ 8 bit) |
| Điện áp hoạt động | 5V – DC (cấp qua cổng USB) |
| Tần số hoạt động | 16 MHz |
| Dòng tiêu thụ | 30mA |
| Điện áp vào khuyên dùng | 7 – 12V – DC |
| Điện áp vào giới hạn | 6 – 20V – DC |
| Số chân Digital I/O | 14 (6 chân PWM) |
| Số chân Analog | 6 (độ phân giải 10 bit) |
| Dòng tối đa trên mỗi chân I/O | 30 mA |
| Dòng ra tối đa (5V) | 500 mA |
| Dòng ra tối đa (3.3V) | 50 mA |
| Bộ nhớ flash | 32 KB (Atmega328) với 0.5KB dùng bởi bootloader |
| SRAM | 2KB (Atmega328) |
| EEPROM | 1KB (Atmega328) |

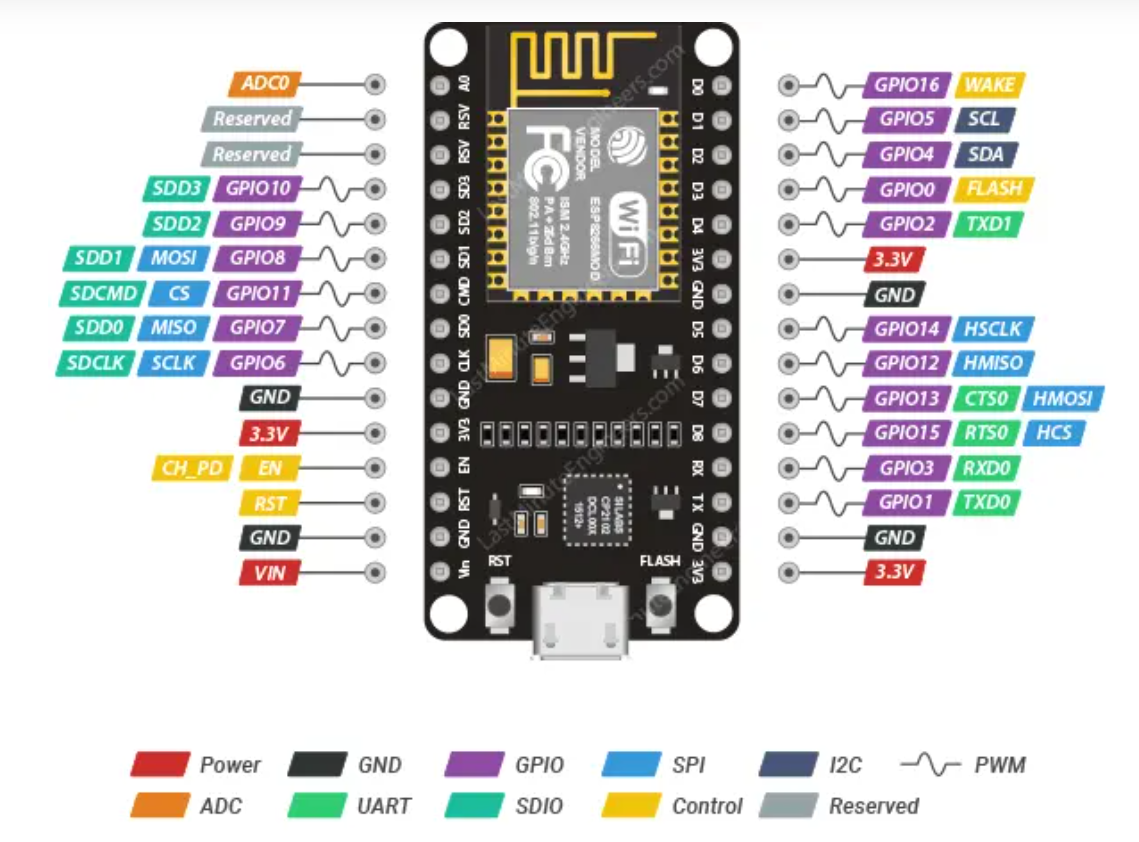
1. ***ESP8266 nodeMCU.***

**ESP8266 NodeMCU** là một nền tảng IoT mã nguồn mở, được phát triển bởi một nhóm kỹ sư tại Trung Quốc. Nền tảng này được xây dựng trên ESP8266, một vi điều khiển Wi-Fi SoC (System on a Chip) được sản xuất bởi Espressif Systems. NodeMCU cung cấp một bộ SDK để lập trình cho ESP8266 bằng ngôn ngữ Lua hoặc C++. Với các tính năng như Wi-Fi, GPIO, ADC, I2C, SPI, PWM và một số tính năng khác, **NodeMCU ESP 8266** được sử dụng rộng rãi trong các ứng dụng IoT như kiểm soát thiết bị, thu thập dữ liệu và giao tiếp với các thiết bị khác.

***Thông số kỹ thuật của NodeMCU***

* Microcontroller: ESP8266EX
* Điện áp hoạt động: 3.3V DC
* Số chân I/O: 17 chân GPIO
* Kết nối mạng: WiFi 802.11 b/g/n
* Giao diện mạng: TCP/IP
* Đồng hồ thời gian thực (RTC): không tích hợp
* Bộ nhớ trong: 4MB
* RAM: 80KB
* Cổng nạp: Micro-USB
* Hỗ trợ các giao thức: MQTT, CoAP, HTTP/HTTPS
* Kích thước: 49 x 24.5 x 13mm

***Sơ đồ chân ESP8266 nodeMCU:***

******

*Chân ADC:*

ESP8266 NodeMCU có một chân ADC( Analog- to- Digital Converter) duy nhất, được kí hiệu là A0. Chân ADC này cho phép đọc giá trị Analog từ các cảm biến hoặc linh kiện có đầu ra Analog.

*Chân SPI:*

ESP8266 có hai giao tiếp SPI, đó là SPI chính( SPI) và SPI phụ( HSPI). Cả hai giao thức SPI này hỗ trợ các tính năng và cấu hình sau:

Giao tiếp SPI cho phép bạn chọn từ 4 chế độ thời gian truyền dữ liệu khác nhau, cung cấp linh hoạt trong việc truyền và nhận dữ liệu qua SPI.

ESP8266 hỗ trợ tốc độ truyền dữ liệu SPI lên đến 80 MHz, cho phép truyền dữ liệu nhanh chóng và hiệu quả.

Xung clock SPI có thể được chia để tạo ra tần số hoạt động chính xác cho giao thức SPI.

Cả SPI chính và SPI phụ đều hỗ trợ FIFO( First- In- First- Out) với bộ đệm 64 byte. Điều này giúp đảm bảo truyền dữ liệu liên tục và ổn định trong quá trình truyền và nhận thông tin qua SPI.

*Chân I2C:*

Phần cúng của ESP8266 không được tích hợp sẵn I2C, nhưng nó có thể được thực hiện bằng phương pháp ‘bitbanging’.

Theo mặc định, GPIO4( SDA) và GPIO5( SCL) được sử dụng làm chân giao tiếp I2C. Tuy nhiên, bạn có thể sử dụng bất kì hai chân GPIO khác để làm chân I2C bằng cách gọi **wire.begin( SDA, SCL)** trong Arduino IDE.

*Chân UART:*

ESP8266 có hai giao tiếp UART, đó là UART0 và UART1, hỗ trợ giao tiếp không đồng bộ( RS232 và RS485) với tốc độ lên tới 4,5 Mbps.

*Chân PWM:*

Tất cả chân GPIO từ GPIO0 đến GPIO15 trên ESP8266 đều có khả năng lập trình và sử dụng để điều chế độ rộng xung( PWM).

Trên ESP8266, tín hiệu PWM có độ phân giải 10 bit, tức là có thể có đến 1024 mức điều chỉnh khác nhau. Dải tần số của tín hiệu PWM có thể điều chỉnh từ khoảng 100Hz đến 1 kHz.

*Chân SDIO:*

ESP8266 có một SDIO phụ để kết nối thẻ nhớ SD. Hỗ trợ SDIO v1.1 và SDIO v2.0.

*Chân nguồn:*

Chân VIN được sử dụng để cấp nguồn trực tiếp cho ESP8266 và các thiết bị ngoại vi.

Chân 3V3 là đầu ra được điều chỉnh từ IC ổn áp trên mạch.

GND là chân nối đất.

*Chân ngắt( Interrupt):*

Tất cả các chân GPIO của NodeMCU ESP8266 có thể được cấu hình như ngắt, trừ GPIO16.

*Chân điều khiển:*

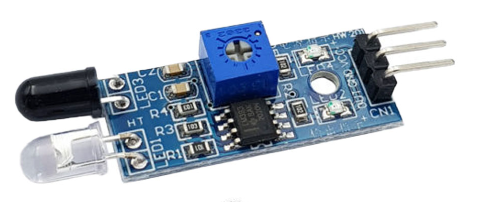
Chân EN( còn được gọi là CH\_PD hoặc Chip Power Down) là chân kích hoạt cho ESP8266, mặc định được kéo lên mức cao. Khi kéo lên mức cao, chip được kích hoạt, khi kéo xuống mức thấp, chip bị vô hiệu hóa.

Chân RST là chân reset cho ESP8266, mặc định được kéo lên mức cao. Khi kéo xuống mức thấp trong một khoảng thời gian ngắn, nó sẽ khởi động lại ESP8266. Điều này giống như với việc nhấn nút RST trên bo mạch.

Chân FLASH được sử dụng để xác định khi nào khởi động vào chế độ nạp chương trình. Nếu chân này được giữ ở mức thấp trong quá trình khởi động, nó sẽ bắt đầu quá trình nạp chương trình.

Chân WAKE được sử dụng để đánh thức ESP8266 từ chế độ ngủ sâu( deep sleep).

1. ***Cảm biến vật cản hồng ngoại.***

Cảm biến hồng ngoại được hiểu là thiết bị tự động được hoạt động dựa trên nguyên tắc điện dung, có chức năng đo và phát hiện những bức xạ hồng ngoại. Bức xạ hồng ngoại là những nguồn ánh sáng mà mắt của con người không thể nhìn thấy được.

***Cấu tạo:***

* Đèn led hồng ngoại: là thiết bị được phát ra từ nguồn sáng hồng ngoại.
* Máy dò hồng ngoại: là thiết bị nhận tín hiệu và phát hiện ra bức xạ hồng ngoại phản xạ trở lại.
* Điện trở: là thiết bị có tác dụng đi cường độ dòng điện quá lớn chạy quá đèn led làm cho hệ thống chập cháy.
* Dây điện: tác dụng chính là kết nối các chi tiết để tạo nên cảm biến hoạt động ổn định.

***Nguyên lí hoạt động:*** Vật thể nào cũng có thể phát ra được bức xạ hồng ngoại với mức độ nhiều hay ít. Vậy nên khi một người hoặc là vật thể lạ đi ngang qua thiết bị cảm biến thì sẽ xuất hiện ngay một tín hiệu. Và tín hiệu này sẽ được cảm biến thu vào và để cho mạch xử lí tạo tác dụng để điều khiển hoặc sẽ báo động khi cần thiết.

1. ***Servo.***

Servo là hệ thống hồi tiếp vòng kín và truyền động. Khi mà các động vơ Servo nhận được tín hiệu báo về từ lệnh PLC và từ đó thực hiện chính xác, nhanh chóng các thao tác.

Bộ Servo được xem là hoàn chỉnh sẽ gồm có: 1 motor Servo, 1 encoder và 1 Servo driver. Chức năng của động cơ Servo chính là thay đổi tốc độ chính xác, điều khiển vị trí, điều chỉnh momen sao cho phù hợp với những ứng dụng trong công việc. Thiết bị sẽ được làm việc dựa trên các cơ chế phản hồi âm về. Điều đó đồng nghĩa với việc nó sẽ chuyển đổi tín hiệu điện năng biến thành những chuyển động có thể kiểm soát.



***Cấu tạo:*** gồm 2 phần chính: rotor, stator.

***Nguyên lí điều khiển Servo:***

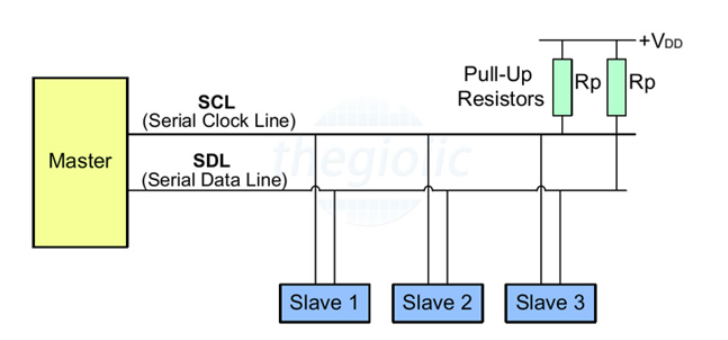
* Động cơ motor Servo sở hữu một rotor là một nam châm vĩnh cửu. Từ trường tạo ra mạnh. Stator của động cơ có thể quấn vào các cuộn dây một cách riêng biệt. Sau đó, chúng sẽ được cung cấp nguồn điện và sẽ hoạt động theo đúng một trình tự thích hợp nhất để làm quay động cơ rotor.
* Khi truyền điện vào cuộn dây đúng thời điểm thì lúc đó chuyển động quay của động cơ rotor sẽ bị phụ thuộc vào pha của dòng điện và tần số của dòng điện. Do đó, dòng điện sẽ dẫn tới bị phân cực và chạy trong cuộn dây stator.
* Động cơ Servo được tạo ra từ hệ thống vòng khép kín. Mạch điều khiển được nối với tín hiệu đầu ra của Servo. Khi mà động cơ quay liên tục kéo theo vận tốc và vị trí sẽ hồi tiếp rồi truyền đến mạch này.
* Nếu có những tác nhân ảnh hưởng tới chuyển động quay và ngăn cản động cơ quay thì khi đó cơ cấu hồi tiếp sẽ nhận được tín hiệu chưa đạt vị trí như mong muốn. Sau đó mạch sẽ tiếp tục điều chỉnh, sửa chữa những sai lệch này cho đến khi mà động cơ đã đạt được vị trí và tốc độ chính xác nhất.

1. ***Cách thức truyền dữ liệu theo chuẩn giao tiếp I2C.***

***Chuẩn giao tiếp I2C là gì?***

I2C hay IIC (Inter – Integrated Circuit) là 1 giao thức giao tiếp nối tiếp đồng bộ được phát triển bởi Philips Semiconductors, sử dụng để truyền nhận dữ liệu giữa các IC với nhau chỉ sử dụng hai đường truyền tín hiệu.

I2C kết hợp các tính năng tốt nhất của SPI và UART. I2C có thể kết nối nhiều slave với một master duy nhất (như SPI) và có thể có nhiều master điều khiển một hoặc nhiều slave. Điều này thực sự cần thiết khi muốn có nhiều hơn một vi điều khiển ghi dữ liệu vào một thẻ nhớ duy nhất hoặc hiển thị văn bản trên một màn hình LCD.

******

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

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

Các bit dữ liệu sẽ được truyền từng bit một dọc theo một đường duy nhất (SDA) theo các khoảng thời gian đều đặn được thiết lập bởi 1 tín hiệu đồng hồ (SCL).

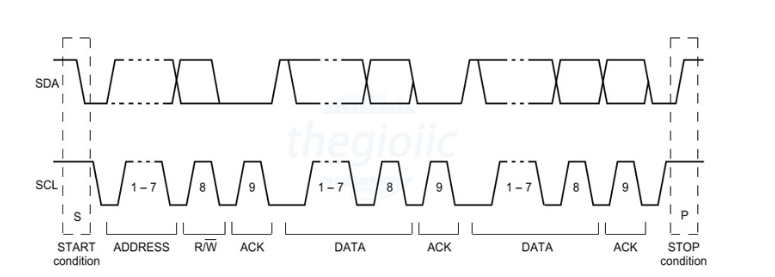
***Cách hoạt động của I2C:***

Giao tiếp I2C bao gồm quá trình truyền nhận dữ liệu giữa các thiết bị chủ tớ, hay Master - Slave.

Thiết bị Master là 1 vi điều khiển, nó có nhiệm vụ điều khiển đường tín hiệu SCL và gửi nhận dữ liệu hay lệnh thông qua đường SDA đến các thiết bị khác.

Các thiết bị nhận các dữ liệu lệnh và tín hiệu từ thiết bị Master được gọi là các thiết bị Slave. Các thiết bị Slave thường là các IC, hoặc thậm chí là vi điều khiển.

Master và Slave được kết nối với nhau bằng hai đường bus SCL và SDA đều hoạt động ở chế độ Open Drain, nghĩa là bất cứ thiết bị nào kết nối với mạng I2C này cũng chỉ có thể kéo 2 đường bus này xuống mức thấp (LOW), nhưng lại không thể kéo được lên mức cao. Vì để tránh trường hợp bus vừa bị 1 thiết bị kéo lên mức cao vừa bị 1 thiết bị khác kéo xuống mức thấp gây hiện tượng ngắn mạch. Do đó cần có 1 điện trờ ( từ 1 – 4,7 kΩ) để giữ mặc định ở mức cao.

******

Với I2C, dữ liệu được truyền trong các tin nhắn. Tin nhắn được chia thành các khung dữ liệu. Mỗi tin nhắn có một khung địa chỉ chứa địa chỉ nhị phân của địa chỉ slave và một hoặc nhiều khung dữ liệu chứa dữ liệu đang được truyền. Thông điệp cũng bao gồm điều kiện khởi động và điều kiện dừng, các bit đọc / ghi và các bit ACK / NACK giữa mỗi khung dữ liệu:

* Điều kiện khởi động: Đường SDA chuyển từ mức điện áp cao xuống mức điện áp thấp trước khi đường SCL chuyển từ mức cao xuống mức thấp.
* Điều kiện dừng: Đường SDA chuyển từ mức điện áp thấp sang mức điện áp cao sau khi đường SCL chuyển từ mức thấp lên mức cao.
* Bit địa chỉ: Thông thường quá trình truyền nhận sẽ diễn ra với rất nhiều thiết bị, IC với nhau. Do đó để phân biệt các thiết bị này, chúng sẽ được gắn 1 địa chỉ vật lý 7 bit cố định.
* Bit đọc / ghi: Bit này dùng để xác định quá trình là truyền hay nhận dữ liệu từ thiết bị Master. Nếu Master gửi dữ liệu đi thì ứng với bit này bằng ‘0’, và ngược lại, nhận dữ liệu khi bit này bằng ‘1’.
* Bit ACK / NACK: Viết tắt của Acknowledged / Not Acknowledged. Dùng để so sánh bit địa chỉ vật lý của thiết bị so với địa chỉ được gửi tới. Nếu trùng thì Slave sẽ được đặt bằng ‘0’ và ngược lại, nếu không thì mặc định bằng ‘1’.
* Bit dữ liệu: Gồm 8 bit và được thiết lập bởi thiết bị gửi truyền đến thiết bị nhân. Sau khi các bit này được gửi đi, lập tức 1 bit ACK/NACK được gửi ngay theo sau để xác nhận rằng thiết bị nhận đã nhận được dữ liệu thành công hay chưa. Nếu nhận thành công thì bit ACK/NACK được set bằng ‘0’ và ngược lại.

***Quá trình truyền nhận:***

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

***Các chế độ hoạt động của I2C:***

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

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

Để tránh điều đó, khi làm việc ở chế độ này, mỗi thiết bị Master cần phát hiện xem đường SDA đang ở trạng thái nào. Nếu SDA ở mức 0, nghĩa là đang có 1 thiết bị Master khác đang có quyền điều khiển và phải chờ đến khi truyền xong. Ngược lại nếu SDA ở mức 1, nghĩa là đường truyền SDA đã an toàn và có sử dụng.