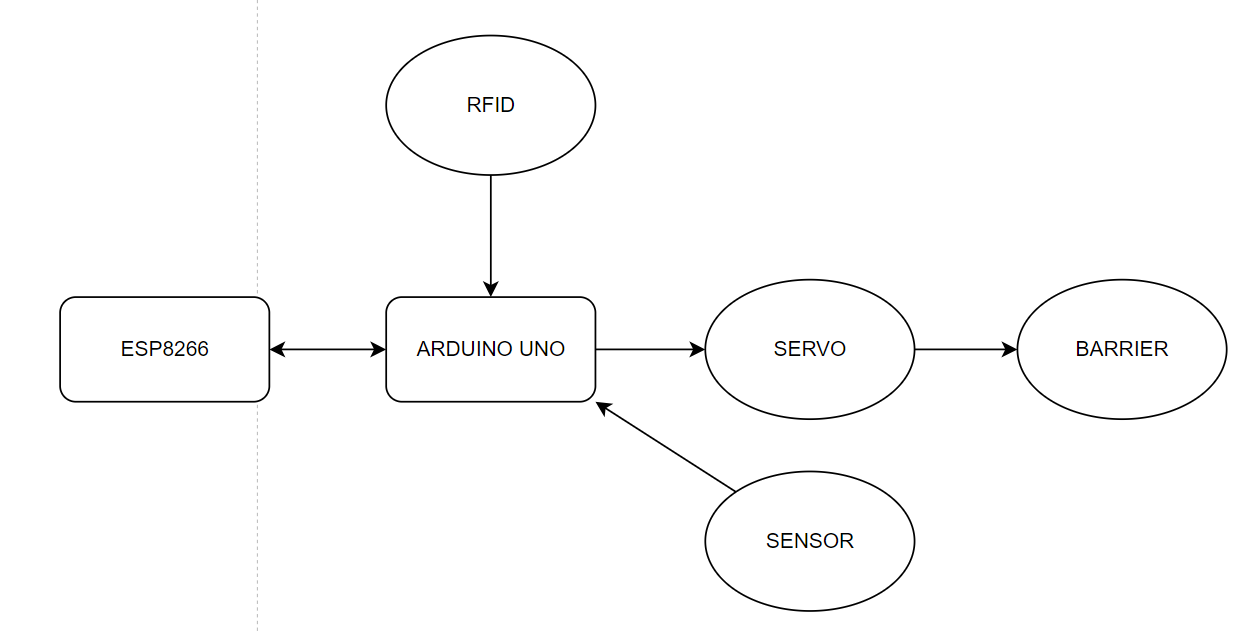
**THIẾT KẾ MÔ HÌNH GIÁM SÁT PHƯƠNG TIỆN RA VÀO CHUNG CƯ**

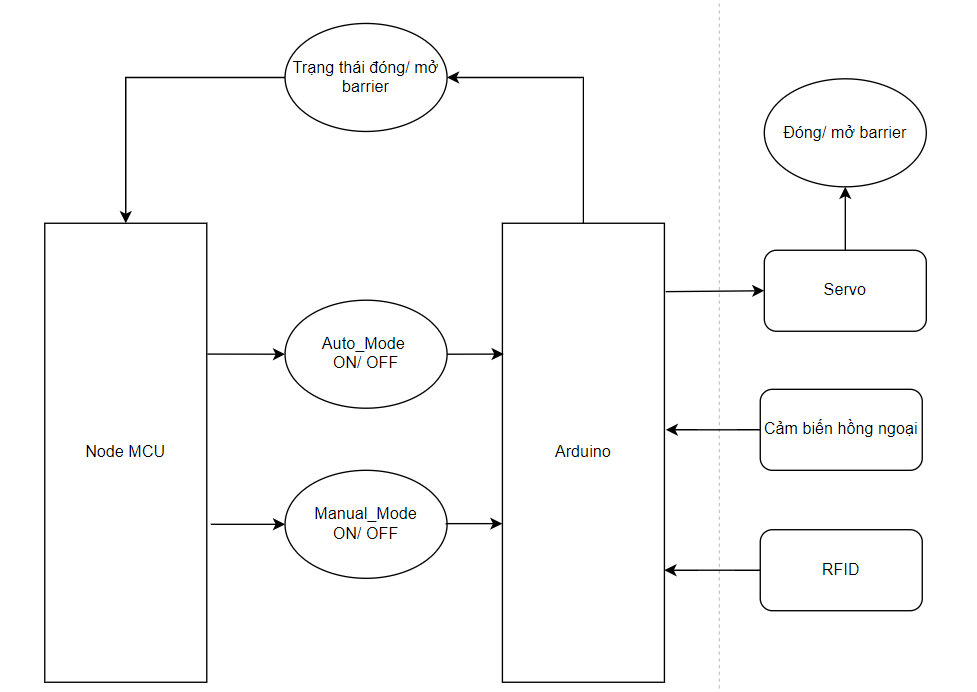
1. ***Thiết bị.***

* 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.
* Màn hình led: hiển thị thông tin thẻ- biển số xe, hiển thị thông tin số chỗ trống đỗ xe.
* 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.

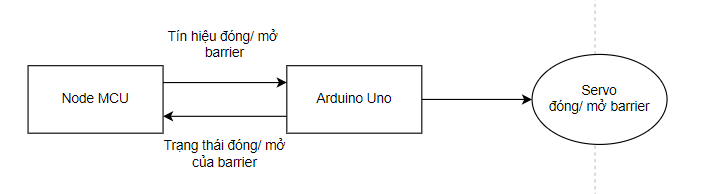
1. ***Mô hình.***
   1. ***Sơ đồ hệ thống phần cứng.***
   2. ***Sơ đồ luồng dữ liệu mô hình.***

******

1. ***Các Module trong Arduino.***

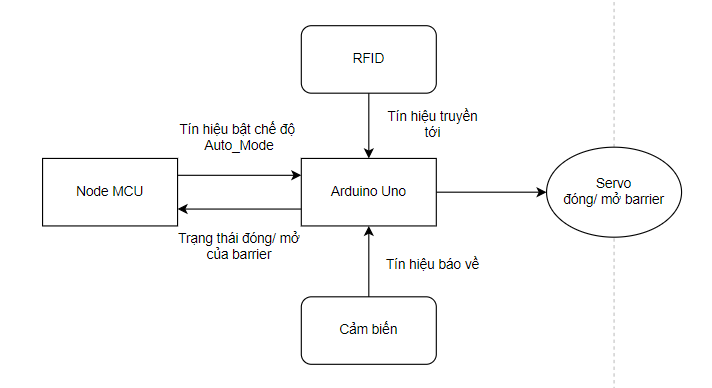
***4.1 Module Manual\_Mode.***

Đây là chế độ Node MCU sẽ gửi tín hiệu đóng/ mở barrier về cho Arduino. Arduino sẽ điều khiển Servo thực hiện việc đóng/ mở barrier và gửi lại trạng thái đóng/ mở barrier cho Node MCU bằng tín hiệu( đóng/ mở barrier thành công, đóng/ mở barrier lỗi).

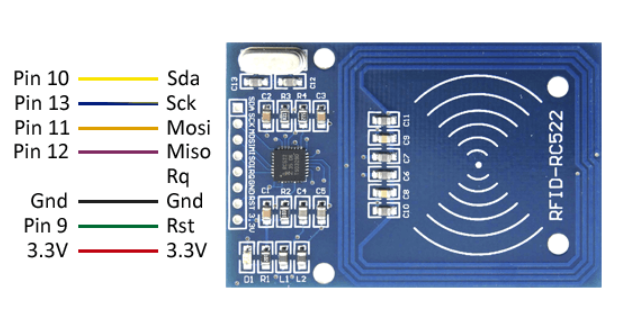
******

* 1. ***Module Auto\_Mode.***

Node MCU nhận dữ liệu từ Server và gửi tín hiệu kích hoạt chế độ AUTO\_MODE cho Arduino. Khi ở trong chế độ AUTO\_MODE, sẽ xảy ra hai sự kiện. Thứ nhất, Arduino sẽ đóng barrier theo tín hiệu của cảm biến hồng ngoại. Khoảng cách hợp lí giữa cảm biến hồng ngoại và Servo, khi vật cản đi tới phạm vi nhận biết của cảm biến hồng ngoại, Arduino sẽ điều khiển Servo thực hiện việc đóng barrier. Thứ hai, Arduino sẽ điều khiển Servo thực hiện việc mở barrier sau khi nhận được tín hiệu từ RFID. Trạng thái đóng/ mở cửa sẽ được gửi qua tín hiệu đến Node\_MCU.

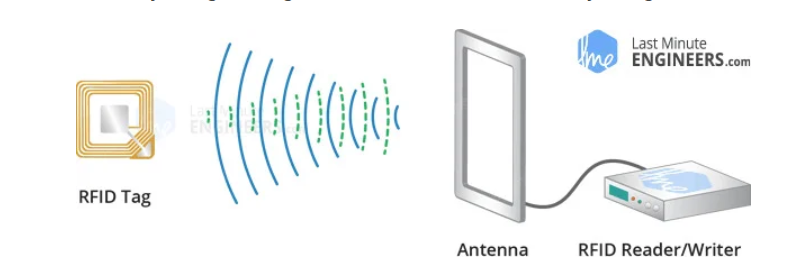
******

1. ***Mô hình mạch.***
2. ***Linh kiện.***
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/).

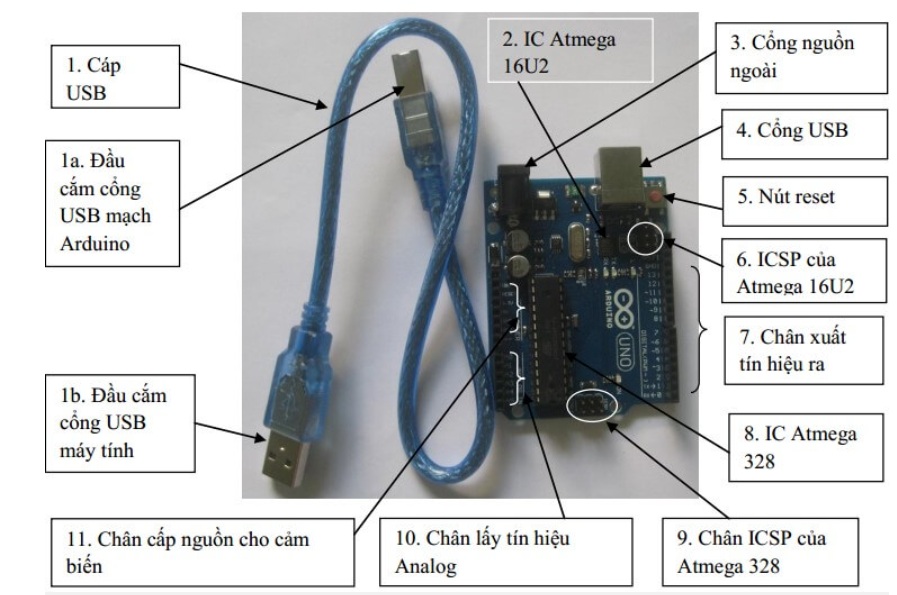
Để sử dụng module RFID RC522, cần có thư viện hỗ trợ cho Arduino như **thư viện MFRC522**, giúp dễ dàng giao tiếp và điều khiển module. Thông qua thư viện này, bạn có thể đọc thông tin từ thẻ RFID, xác minh chúng và thực hiện các hoạt động như đọc mã số, ghi dữ liệu, kiểm tra trạng thái và điều khiển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

1. ***Arduino Uno.***

[Arduino](https://mesidas.com/arduino-la-gi/) là một bo mạch vi điều khiển do một nhóm giáo sư và sinh viên nước Ý thiết kế và đưa ra đầu tiên vào năm 2005. Mạch Arduino được sử dụng để cảm nhận và điều khiển nhiều đối tượng khác nhau. Nó có thể thực hiện nhiều nhiệm vụ lấy tín hiệu từ cảm biến đến điều khiển đèn, động cơ, và nhiều đối tượng khác. Ngoài ra mạch còn có khả năng liên kết với nhiều module khác nhau như module đọc thẻ từ, ethernet shield, sim900A, ….để tăng khả ứng dụng của mạch.

Phần cứng bao gồm một board mạch nguồn mở được thiết kế trên nền tảng vi xử lý AVR Atmel 8bit, hoặc ARM, Atmel 32-bit,…. Hiện phần cứng của Arduino có tất cả 6 phiên bản, Tuy nhiên phiên bản thường được sử dụng nhiều nhất là Arduino Uno và Arduino Mega.

***Cấu tạo:***

******

***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) |

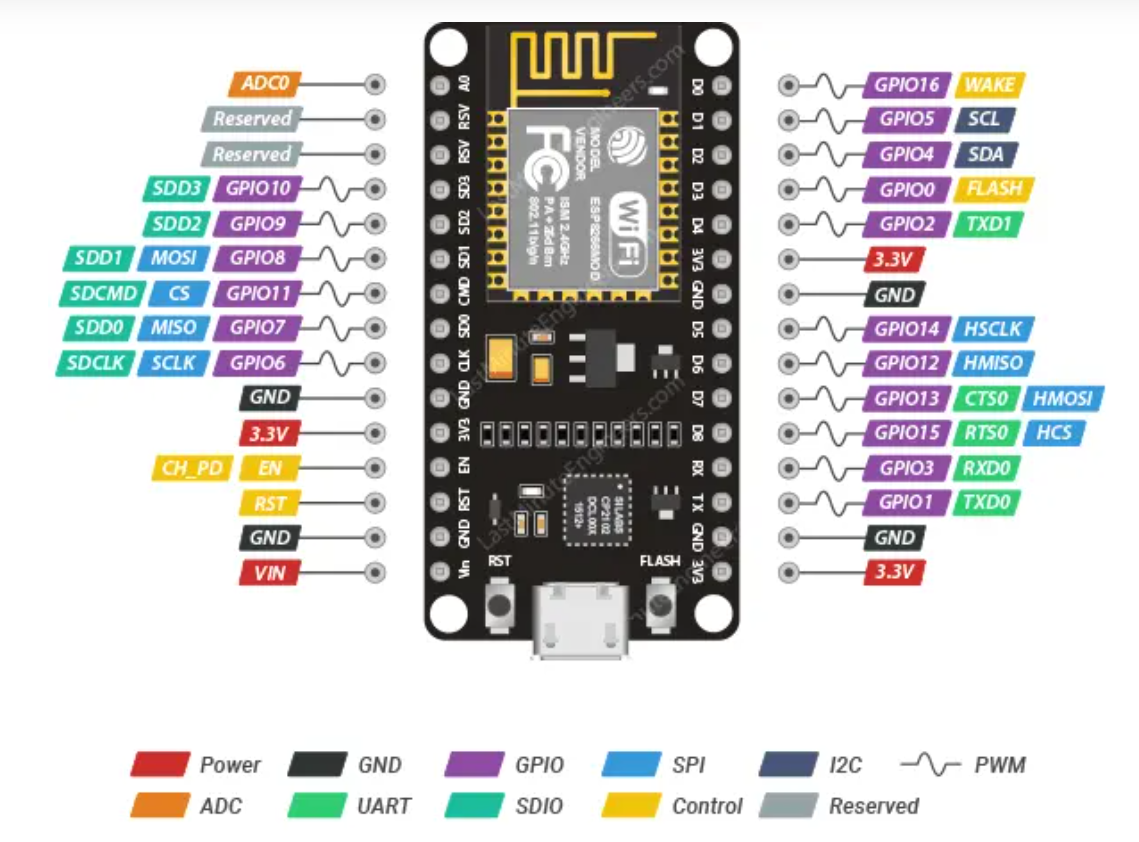
1. ***ESP8266 nodeMCU.***

**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.

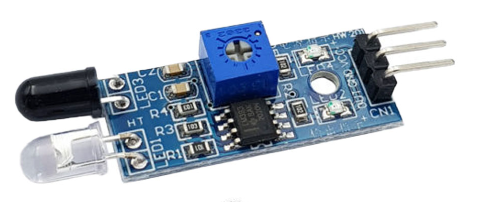
**Thông số kỹ thuật của NodeMCU**

* 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:***

******

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. ***Module camera OV7670 no FIFO.***

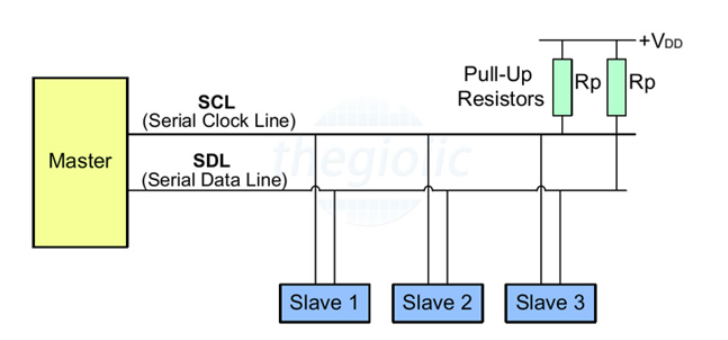
* ******OV7670 là một cảm biến hình ảnh kết hợp với xử lí tín hiệu số( DSP), cho độ phân giải VGA 640x 480, tốc độ khung hình lên tới 30 fps. Hình ảnh thu nhận được có thể được tiền xử lí bởi khối DSP trước khi được truyền đi. Việc cấu hình chế độ hoạt động của OV7670 được cấu hình qua Bus điều khiển camera tuần tự SCCB.

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).

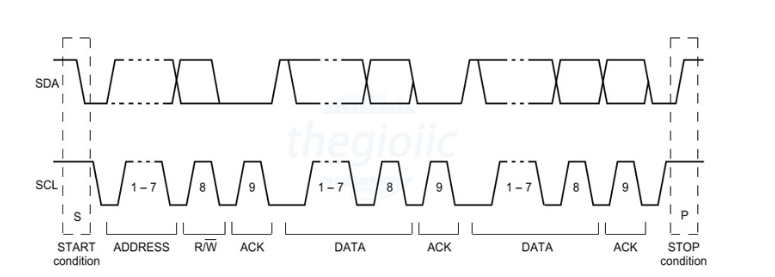
***Cách hoạt động của I2C:***

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.

1. ***Ngôn ngữ lập trình.***