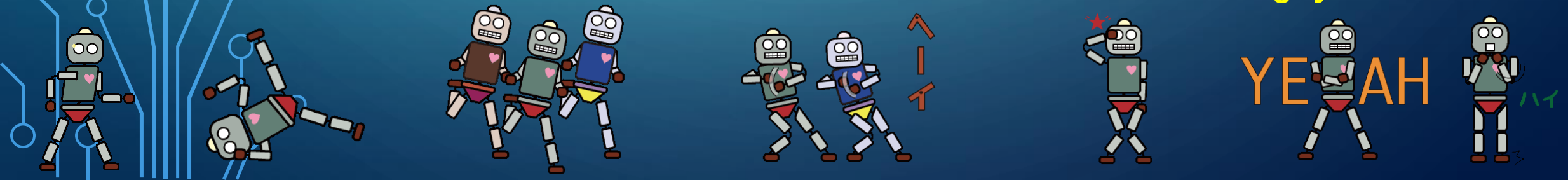
LY TU TRONG
COLLEGE

TRƯỜNG CAO ĐẲNG LÝ TỰ TRỌNG TP.HCM
KHOA ĐIỆN – ĐIỆN TỬ

HỌC PHẦN

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

GV: **Trần Nguyên Bảo Trân**



Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.1 Tổng quan

4.2 Phát hiện chuyển động dùng cảm biến độ nghiêng

4.3 Phát hiện ánh sáng dùng quang trở

4.4 Đo khoảng cách dùng cảm biến siêu âm

4.5 Đọc thẻ RFID

4.6 La bàn số

Bài tập

Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.1 Tổng quan

Các cảm biến sử dụng các phương pháp sau để cung cấp thông tin:

+ Cảm biến trả về 2 trạng thái On/ Off

như cảm biến độ nghiêng, cảm biến chuyển động

+ Cảm biến trả về tín hiệu tương tự điện áp tỷ lệ thuận với những gì được cảm nhận, chẳng hạn như nhiệt độ hoặc mức độ ánh sáng

cảm biến phát hiện ánh sáng, chuyển động, độ rung, âm thanh và gia tốc

+ Cảm biến trả về độ rộng xung

như cảm biến siêu âm PING))) trả về dữ liệu độ rộng xung tỷ lệ với giá trị khoảng cách

+ Cảm biến trả về dữ liệu thông qua giao tiếp nối tiếp

như đầu đọc RFID và GPS

+ Cảm biến trả về dữ liệu thông qua giao thức đồng bộ: I2C, SPI

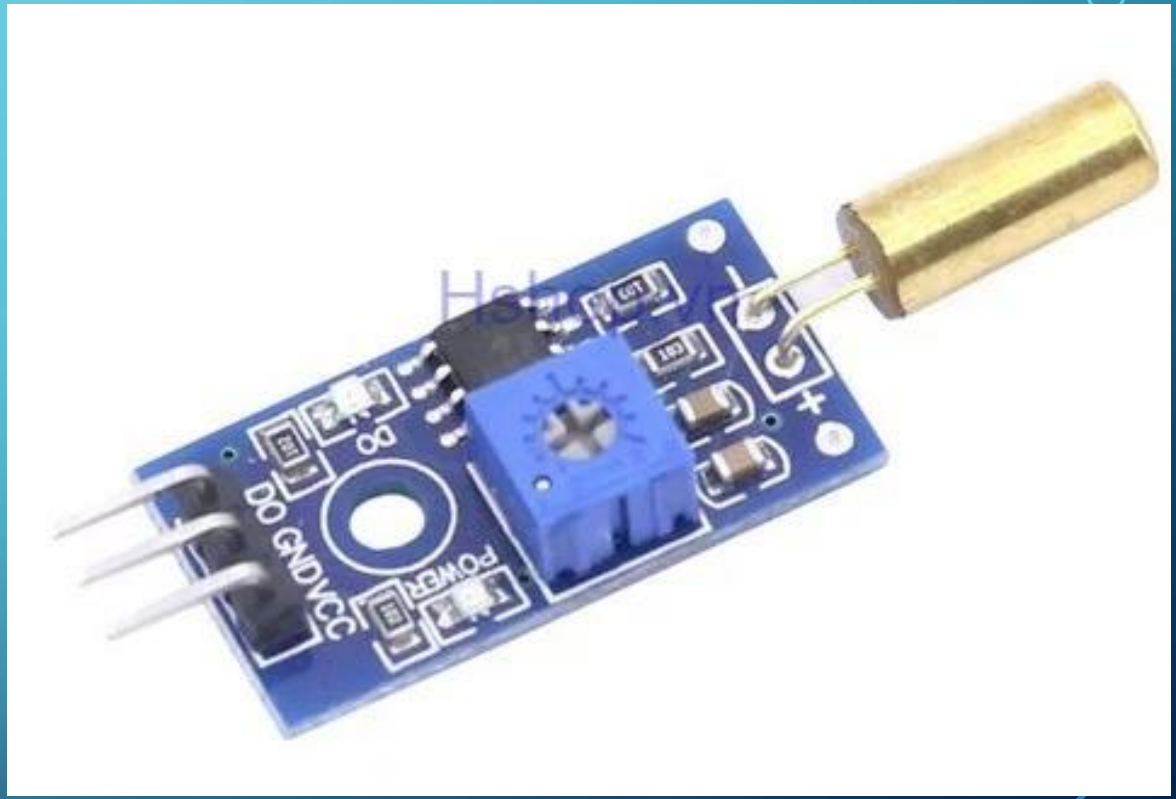
như mô-đun la bàn

Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.2 Phát hiện chuyển động dùng cảm biến độ nghiêng

Mạch cảm biến góc nghiêng Tilt Switch SW520 được sử dụng để **phát hiện góc nghiêng** (tilt) hoặc **rung động** (vibration) của vật thể.

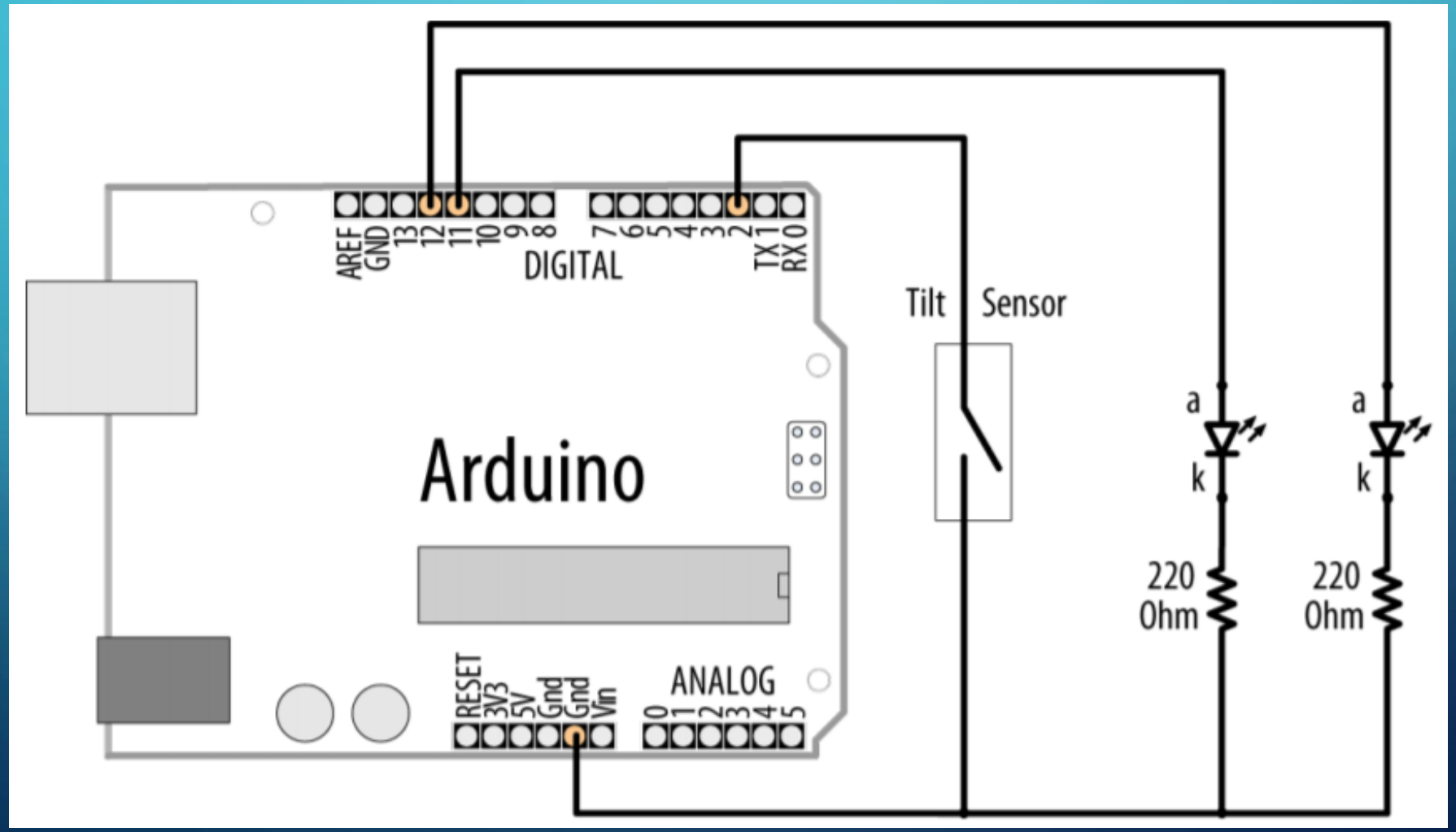
Cảm biến trả về tín hiệu dạng Digital (2 trạng thái On / Off)



Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.2 Phát hiện chuyển động dùng cảm biến độ nghiêng

Sơ đồ mạch nguyên lý



Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.2 Phát hiện chuyển động dùng cảm biến độ nghiêng

Chương trình

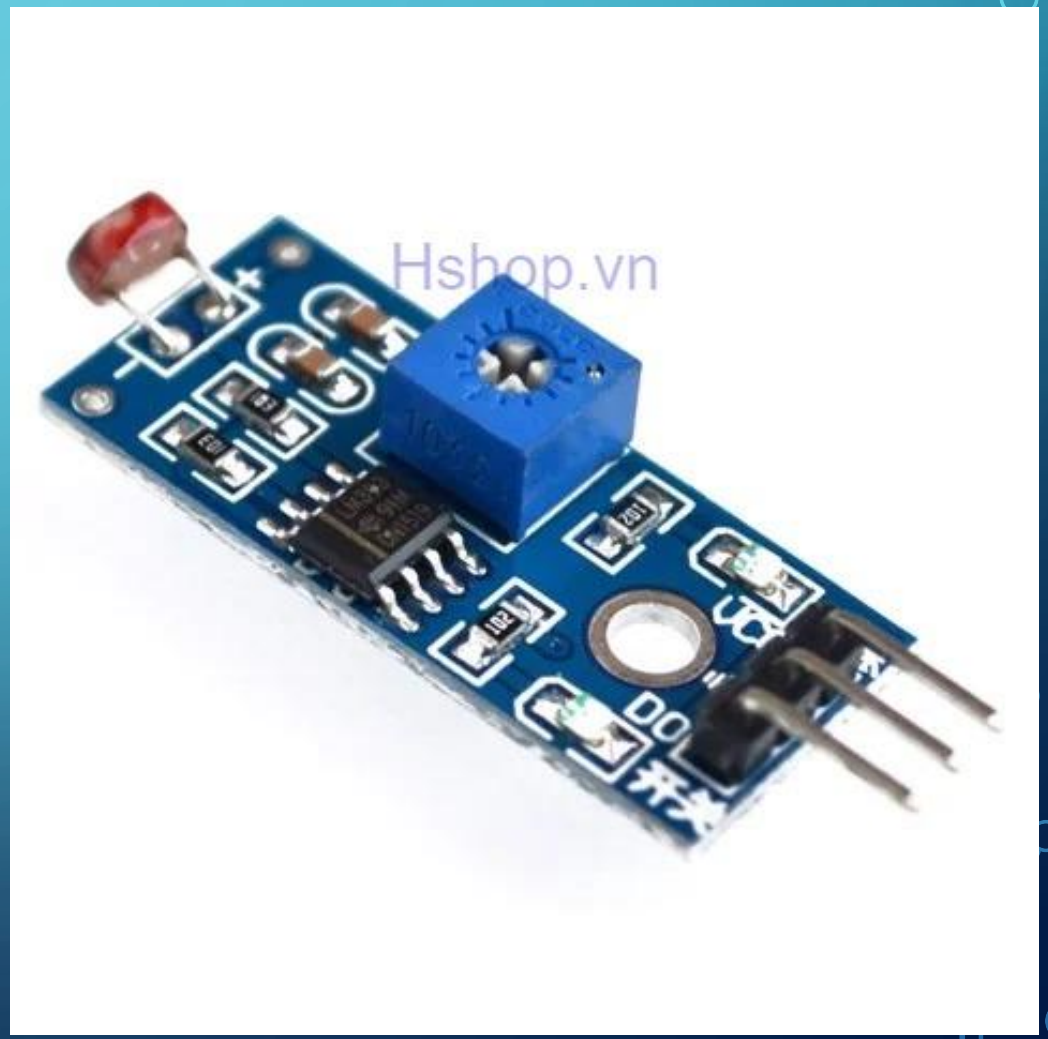
```
const int tiltSensorPin = 2;       //pin the tilt sensor is connected to
const int firstLEDPin = 11;        //pin for one LED
const int secondLEDPin = 12;       //pin for the other
void setup()
{
 pinMode (tiltSensorPin, INPUT);       //the code will read this pin
 digitalWrite (tiltSensorPin, HIGH);   //and use a pull-up resistor
 pinMode (firstLEDPin, OUTPUT);        //the code will control this pin
 pinMode (secondLEDPin, OUTPUT);       //and this one
}

void loop()
{
 if (digitalRead(tiltSensorPin)){      //check if the pin is high
 digitalWrite(firstLEDPin, HIGH);      //if it is high turn on firstLED
 digitalWrite(secondLEDPin, LOW);      //and turn off secondLED
 }
 else{ //if it isn't
 digitalWrite(firstLEDPin, LOW);       //do the opposite
 digitalWrite(secondLEDPin, HIGH);
 }
}
```

Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.3 Phát hiện ánh sáng dùng quang trở

- Cảm biến ánh sáng quang trở CDS Light Sensor sử dụng để nhận biết cường độ ánh sáng môi trường

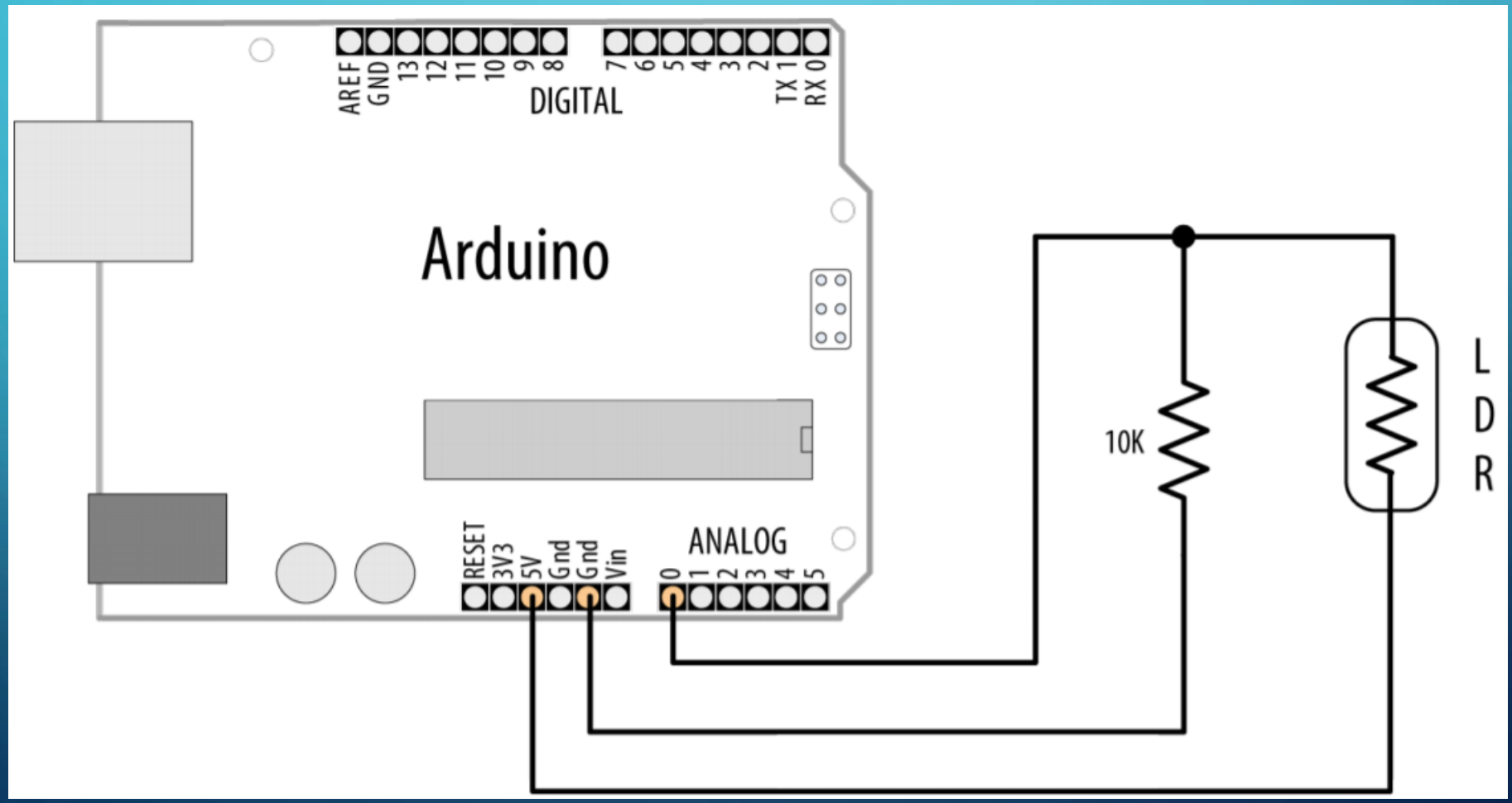
- Cảm biến trả về tín hiệu dạng Analog



Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.3 Phát hiện ánh sáng dùng quang trở

Sơ đồ mạch nguyên lý



Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.3 Phát hiện ánh sáng dùng quang trở

Chương trình

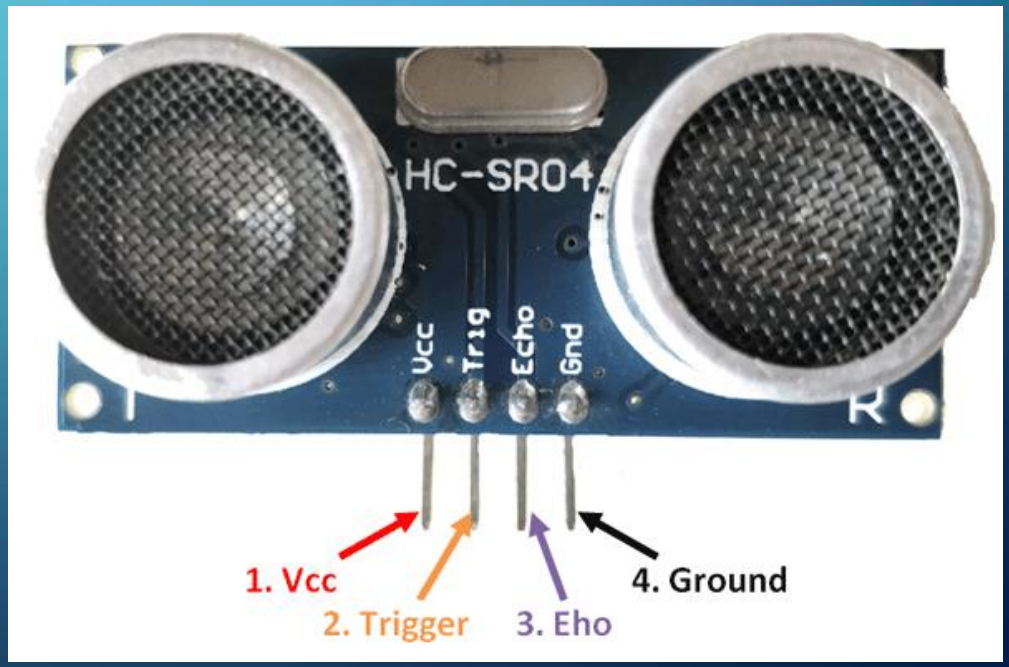
```
const int ledPin = 13;       // LED connected to digital pin 13
const int sensorPin = 0;     // connect sensor to analog input 0
void setup()
{
 pinMode(ledPin, OUTPUT);    // enable output on the led pin
}
void loop()
{
 int rate = analogRead(sensorPin);      // read the analog input
 digitalWrite(ledPin, HIGH);           // set the LED on
 delay(rate);                          // wait duration dependent on light level
 digitalWrite(ledPin, LOW);            // set the LED off
 delay(rate);
}
```

Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.4 Đo khoảng cách dùng cảm biến siêu âm

Ultrasonic Distance Sensor (Cảm biến siêu âm đo khoảng cách) sử dụng sóng siêu âm để đo khoảng cách đến vật cản phía trước, thích hợp để làm các loại robot tránh vật cản, chống trộm, đo khoảng cách,...

Cảm biến trả về độ rộng xung

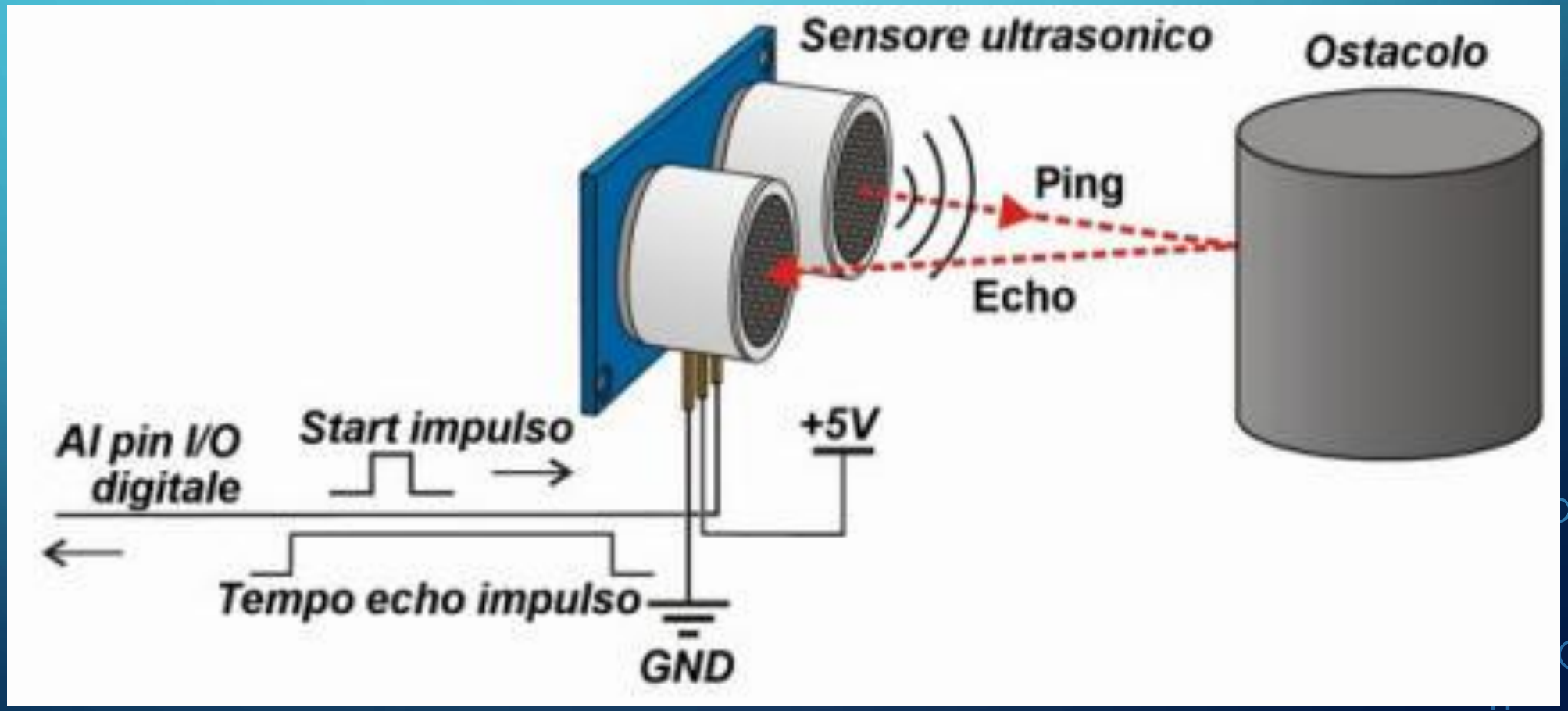


Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.4 Đo khoảng cách dùng cảm biến siêu âm

Cảm biến siêu âm sử dụng sóng siêu âm và có thể đo khoảng cách trong khoảng từ 2 -> 300cm, với độ chính xác gần như chỉ phụ thuộc vào cách lập trình.

Nguyên lý:



Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.4 Đo khoảng cách dùng cảm biến siêu âm

Cảm biến siêu âm SR04 sử dụng nguyên lý phản xạ sóng siêu âm.

Cảm biến gồm 2 module:
+ 1 module phát ra sóng siêu âm
+ 1 module thu sóng siêu âm phản xạ về.

Đầu tiên cảm biến sẽ phát ra 1 sóng siêu âm với tần số 40Khz.

Nếu có chướng ngại vật trên đường đi, sóng siêu âm sẽ phản xạ lại và tác động lên module thu sóng.

Bằng cách đo thời gian từ lúc phát đến lúc nhận sóng ta sẽ tính được khoảng cách từ cảm biến đến chướng ngại vật.

Khoảng cách = (thời gian * vận tốc âm thanh (340 m/s) / 2

Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

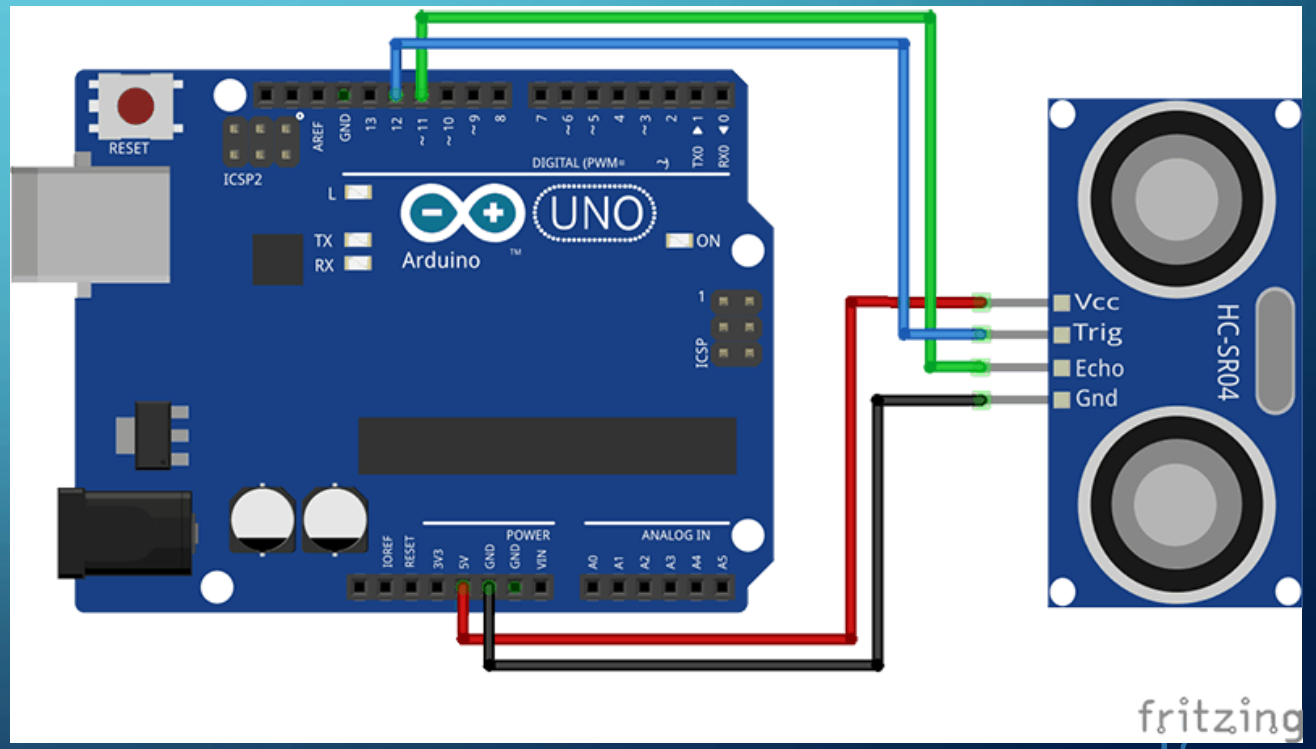
4.4 Đo khoảng cách dùng cảm biến siêu âm

Cảm biến siêu âm 4 chân HC-SR04, HC-SR05

Cảm biến siêu âm sử dụng sóng siêu âm và có thể đo khoảng cách trong khoảng từ 2 -> 300cm, với độ chính xác gần như chỉ phụ thuộc vào cách lập trình.

Kết nối

+ VCC → 5V
+ Trig → chân điều khiển phát
+ Echo → chân nhận tín hiệu phản hồi
+ GND → nối đất



Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.4 Đo khoảng cách dùng cảm biến siêu âm

Chương trình HC-SR04

```
const int trig = 12;     // chân trig của HC-SR04
const int echo = 11;     // chân echo của HC-SR04

void setup()
{
    Serial.begin(9600);     // giao tiếp Serial với baudrate 9600
    pinMode(trig,OUTPUT);   // chân trig sẽ phát tín hiệu
    pinMode(echo,INPUT);    // chân echo sẽ nhận tín hiệu
}
```

Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.4 Đo khoảng cách dùng cảm biến siêu âm

Chương trình HC-SR04

```
void loop()
{
    unsigned long duration; // biến đo thời gian
    int distance;           // biến lưu khoảng cách

    /* Phát xung từ chân trig */
    digitalWrite(trig,0);   // tắt chân trig
    delayMicroseconds(2);
    digitalWrite(trig,1);   // phát xung từ chân trig
    delayMicroseconds(5);   // xung có độ dài 5 microSeconds
    digitalWrite(trig,0);   // tắt chân trig
```

Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.4 Đo khoảng cách dùng cảm biến siêu âm

Chương trình HC-SR04

```
        /* Tính toán thời gian */
        // Đo độ rộng xung HIGH ở chân echo.
        duration = pulseIn(echo,HIGH);
        // Tính khoảng cách đến vật.
        distance = int(duration/2/29.412);

        /* In kết quả ra Serial Monitor */
        Serial.print(distance);
        Serial.println("cm");
        delay(200);
    }
```

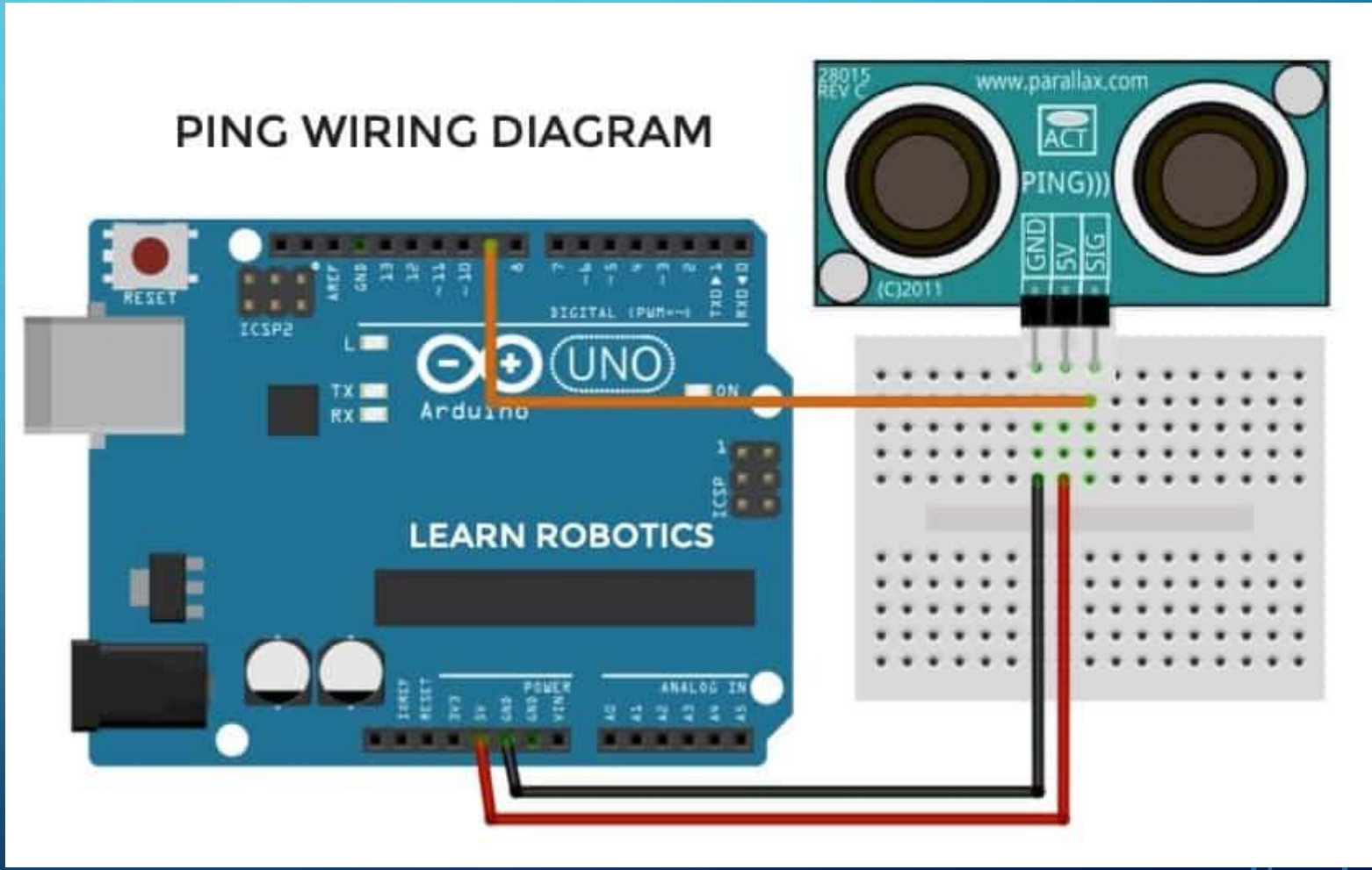
Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.4 Đo khoảng cách dùng cảm biến siêu âm

Cảm biến siêu âm 3 chân PING)))

Kết nối

+ VCC → 5V
+ GND → nối đất
+ SIG → nối chân 9



Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.4 Đo khoảng cách dùng cảm biến siêu âm

Chương trình PING

```
const int pingPin = 9;

void setup() {
   Serial.begin(9600);              // tốc độ baud khi kết nối với máy tính
}

void loop() {
  long duration, inches, cm;       // biến thời gian, khoảng cách
```

Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.4 Đo khoảng cách dùng cảm biến siêu âm

Chương trình PING

```
pinMode(pingPin, OUTPUT);  // cấp 1 xung thấp trước hết để xóa tín hiệu cũ
digitalWrite(pingPin, LOW);
delayMicroseconds(2);
digitalWrite(pingPin, HIGH); // Tín hiệu PING tạo bởi xung cao trong ít nhất 2us
delayMicroseconds(5);
digitalWrite(pingPin, LOW);

pinMode(pingPin, INPUT);      // tín hiệu trở về trên cùng 1 chân
duration = pulseIn(pingPin, HIGH);    // đo thời gian xung phản hồi

// chuyển sang khoảng cách bằng 2 hàm
inches = microsecondsToInches(duration);
cm = microsecondsToCentimeters(duration);
```

Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.4 Đo khoảng cách dùng cảm biến siêu âm

Chương trình PING

```
    Serial.print(inches);          // hiển thị trên máy tính
    Serial.print("in, ");
    Serial.print(cm);
    Serial.print("cm");
    Serial.println();

    delay(100);                    // chờ 100ms
  }
```

Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.4 Đo khoảng cách dùng cảm biến siêu âm

Chương trình PING

```
long microsecondsToInches(long microseconds) {
  // Theo Parallax, cần gần 74 us/in để âm thanh đi và về
  // nên ta chia 2
  // //www.parallax.com/dl/docs/prod/acc/28015-PING-v1.3.pdf
  return microseconds / 74 / 2;
}

long microsecondsToCentimeters(long microseconds) {
  // Tốc độ âm thanh 340 m/s, cỡ 29 us/cm
  return microseconds / 29 / 2;
}
```

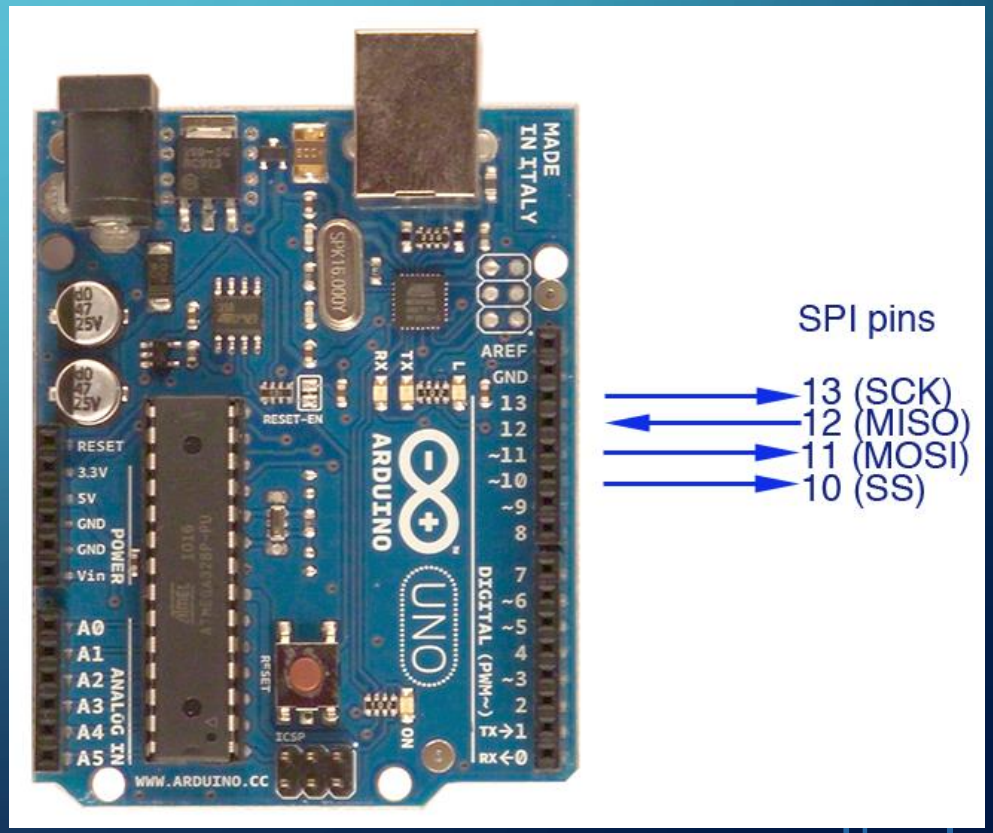
Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.5 Giao tiếp SPI

4.5.1 SPI là gì?

SPI (**Serial Peripheral Bus**) là chuẩn truyền thông nối tiếp tốc độ cao kiểu Master-Slave. Trong đó có 1 chip Master điều phối quá trình tuyền thông và các chip Slaves được điều khiển bởi Master.

SPI truyền **song công** (***full duplex***), nghĩa là tại cùng một thời điểm quá trình truyền và nhận có thể xảy ra đồng thời.

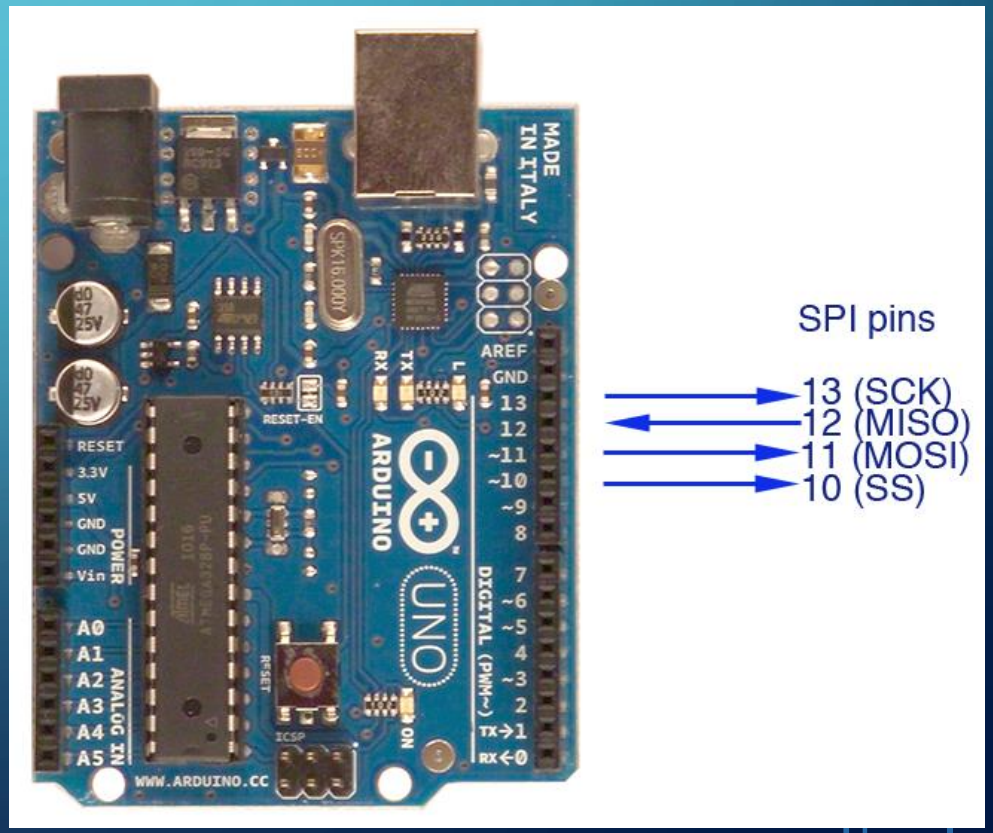


Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.5 Giao tiếp SPI

4.5.1 SPI là gì? SPI có 4 đường giao tiếp trong chuẩn này là:

- **SCK** (Serial Clock): Xung giữ nhịp cho giao tiếp SPI (đồng bộ)
- **MISO** (Master Input Slave Output): Mang các dữ liệu từ các thiết bị SPI về arduino
- **MOSI** (Master Ouput Slave Input): Mang các dữ liệu từ Arduino đến các thiết bị SPI
- SS (Slave Select): là đường chọn Slave cần giao tiếp

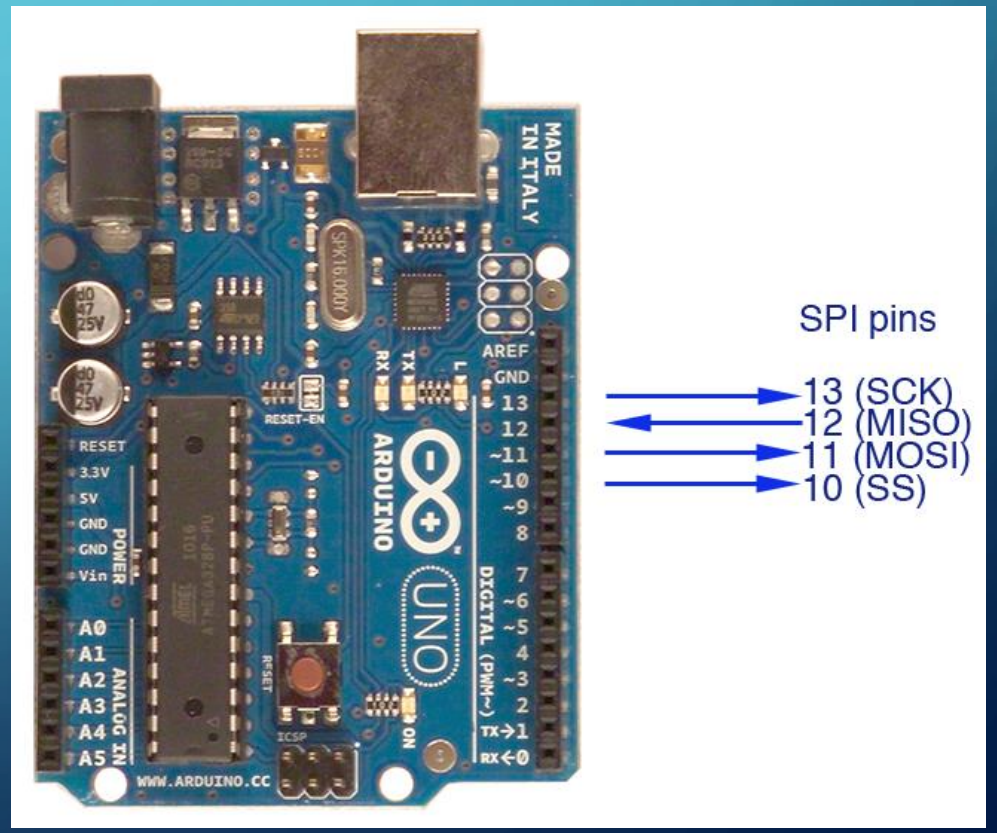


Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.5 Giao tiếp SPI

4.5.1 SPI là gì?

Các chân giao tiếp SPI trên Arduino:



Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.5 Giao tiếp SPI

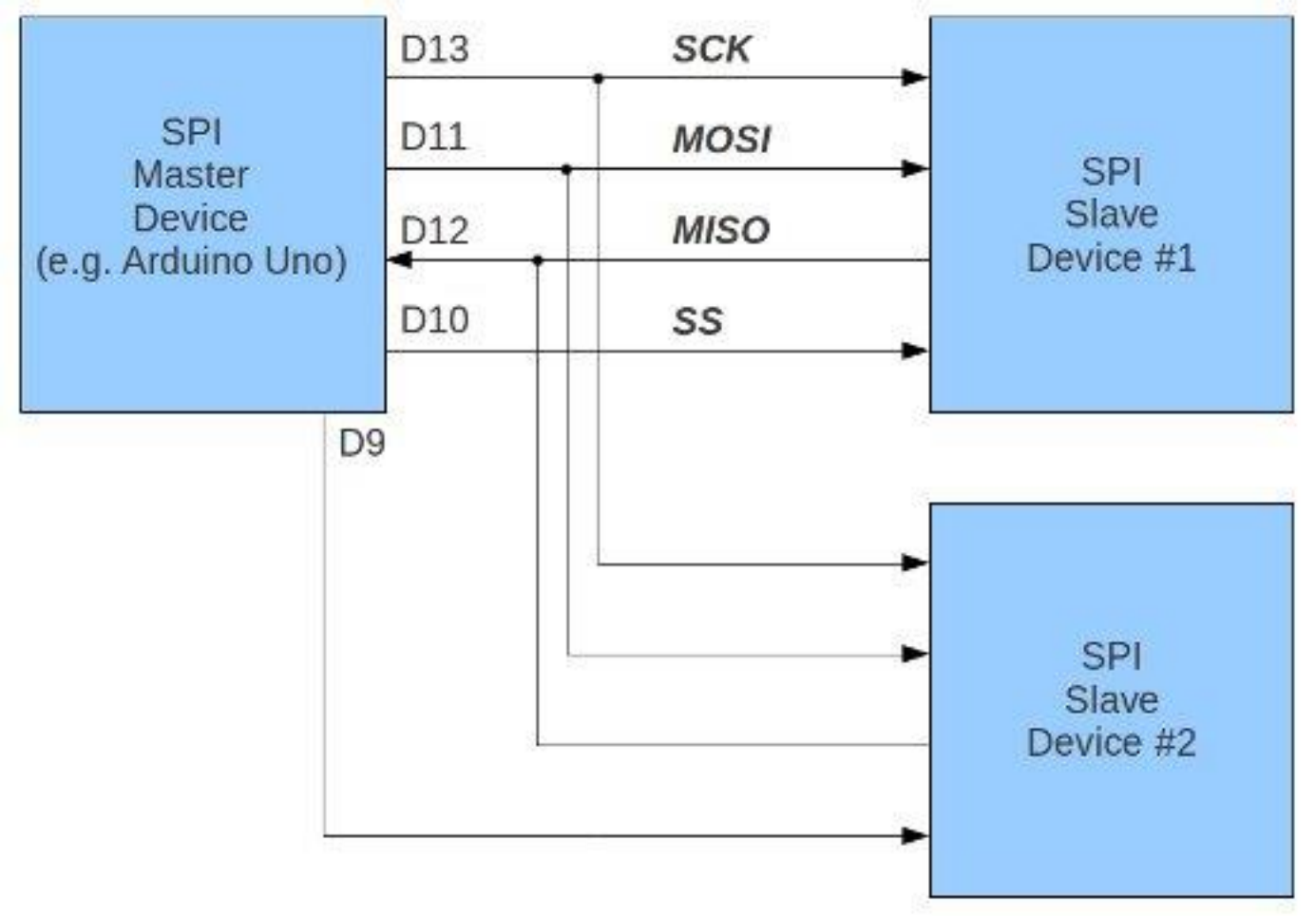
4.5.1 SPI là gì?

Có thể kiểm soát 1 hoặc nhiều thiết bị sử dụng SPI

Trên các chip Slave đường SS sẽ ở mức cao khi không làm việc.

Nếu chip Master kéo đường SS của một Slave nào đó xuống mức thấp thì việc giao tiếp sẽ xảy ra giữa Master và Slave đó.

Chỉ có 1 đường SS trên mỗi Slave nhưng có thể có nhiều đường điều khiển SS trên Master, tùy thuộc vào thiết kế của người dùng.



Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.5 Giao tiếp SPI

4.5.2 Cách giao tiếp SPI

1. Trước hết, chúng ta cần phải **sử dụng thư viện SPI**.

```
#include "SPI.h"
```

2. Trong **void setup ()** khai báo pin (s) để sử dụng cho dòng SS và cài đặt chúng ở dạng OUTPUT. Ví dụ:

```
pinMode(ss, OUTPUT);
```

3. Kích hoạt giao tiếp SPI

```
SPI.begin();
```

Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.5 Giao tiếp SPI

4.5.2 Cách giao tiếp SPI

4. Xác định cách để gửi dữ liệu, MSB hay LSB trước bằng cách sử dụng:

```
SPI.setBitOrder(MSBFIRST);
```

Hoặc là

```
SPI.setBitOrder(LSBFIRST);
```

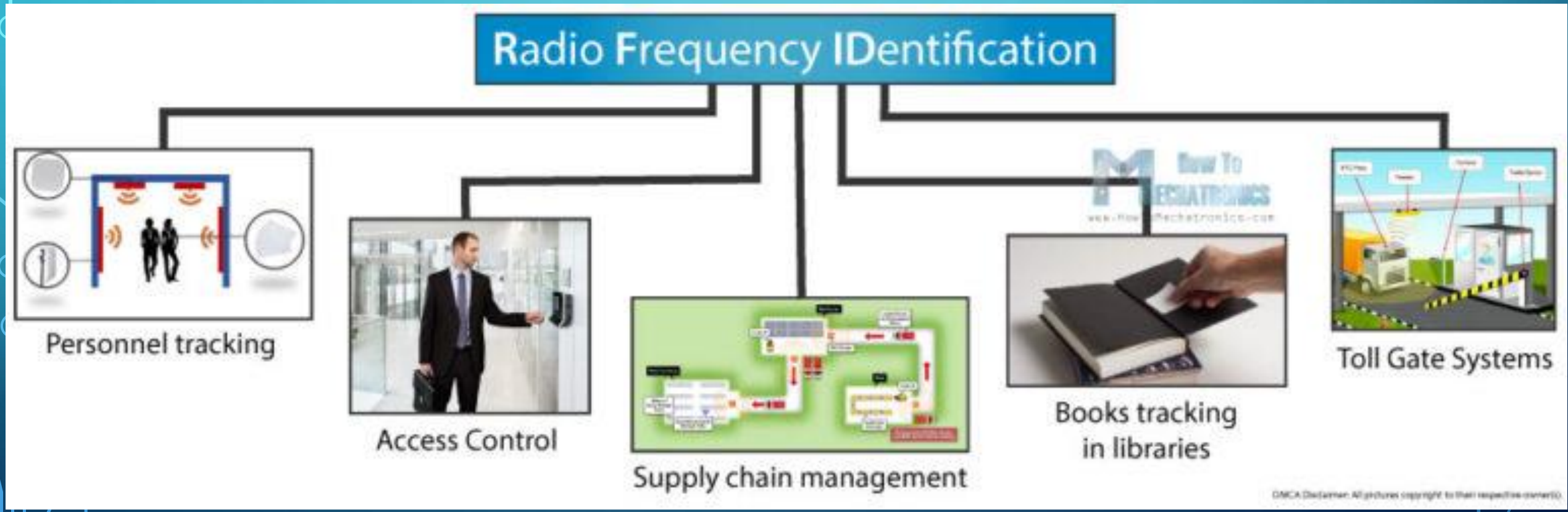
5. Đưa dòng SS về mức thấp, gửi dữ liệu và đưa về mức cao

```
digitalWrite(SS, LOW);
SPI.transfer(value);
digitalWrite(ss, HIGH);
```

Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.6 Đọc thẻ RFID

4.6.1 RFID là gì?



Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.6 Đọc thẻ RFID

4.6.1 RFID là gì?

RFID là chữ viết tắt của **Radio Frequency Indentification** có nghĩa là **nhận dạng tần số sóng vô tuyến**, cho phép nhận biết các đối tượng thông qua hệ thống thu phát sóng radio. Nhờ đó có thể quản lý, giám sát đối tượng chi tiết, nhanh chóng, chính xác, hiệu quả.

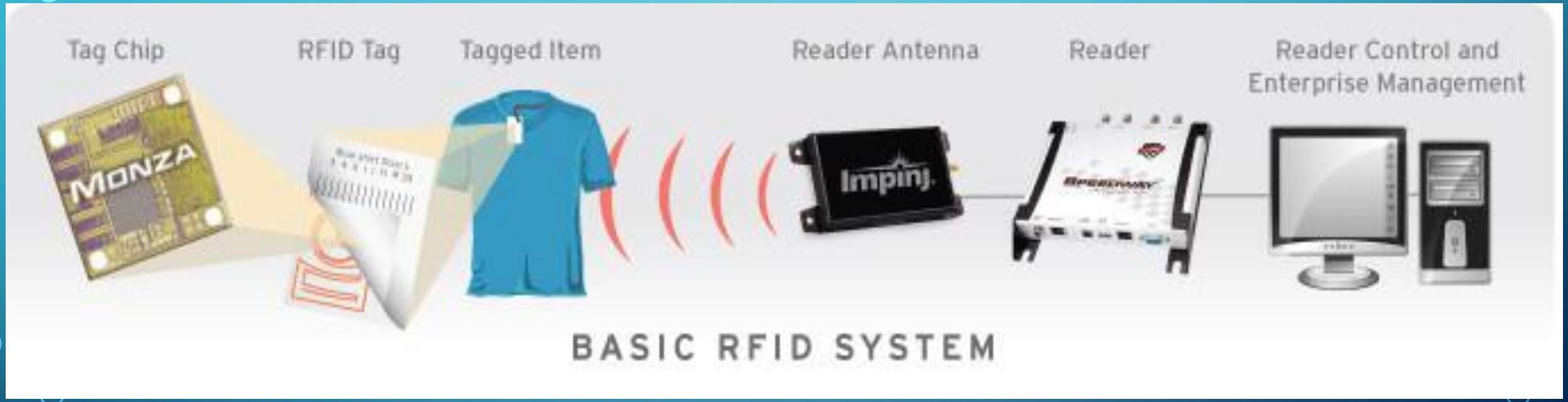
Hiểu một cách đơn giản thì hai thiết bị phát ra sóng điện từ, có cùng một tần số khi gặp nhau có thể nhận dạng được nhau.

Tần số 125Khz và 900Khz là hai tần số thường được sử dụng trong RFID mà chúng ta có thể gặp.

Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.6 Đọc thẻ RFID

4.6.2 Hệ thống RFID:



Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

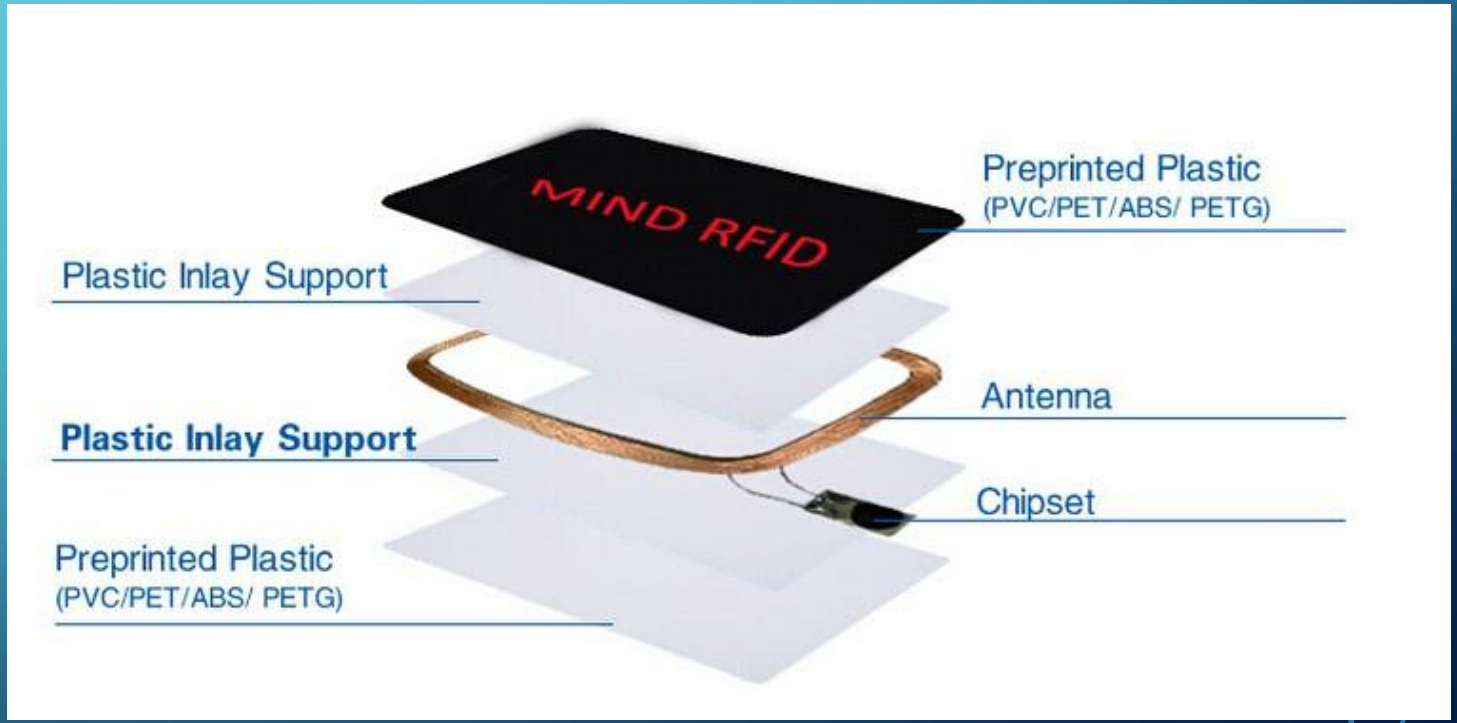
4.5 Đọc thẻ RFID

4.5.2 Hệ thống RFID:

4.5.2.1Tag (Nhãn)

Dùng để dán lên từng thiết bị

Là một thẻ gắn chip + Anten. Được lập trình điện tử với thông tin duy nhất



* **Chip:** (bộ nhớ của chip có thể chứa tới 96 bit đến 512 bit dữ liệu, gấp 64 lần so với mã vạch) lưu trữ một số thứ tự duy nhất hoặc thông tin khác dựa trên loại thẻ: read-only, read-write, hoặc write-once-read-many

* **Antenna:** được gắn với vi mạch truyền thông tin từ chip đến reader. Antenna càng lớn cho biết phạm vi đọc càng lớn

Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.5 Đọc thẻ RFID

4.5.2 Hệ thống RFID:

4.5.2.1Tag (Nhãn)

Dùng để dán lên từng thiết bị

Có hai loại tag: **Active tag** **Passive tag**

Tag có thể phát ra tần số và chuyển tới thiết bị đọc. Tag có nguồn điện (thường sử dụng pin gắn trong thẻ) để duy trì mạch điện của chip và phát ra tần số

Không chứa pin. Nguồn điện được cung cấp từ thiết bị đọc reader.
Loại tag này có nhiều yếu điểm hơn, nên giá cũng rẻ hơn nên có thể áp dụng đại trà được.

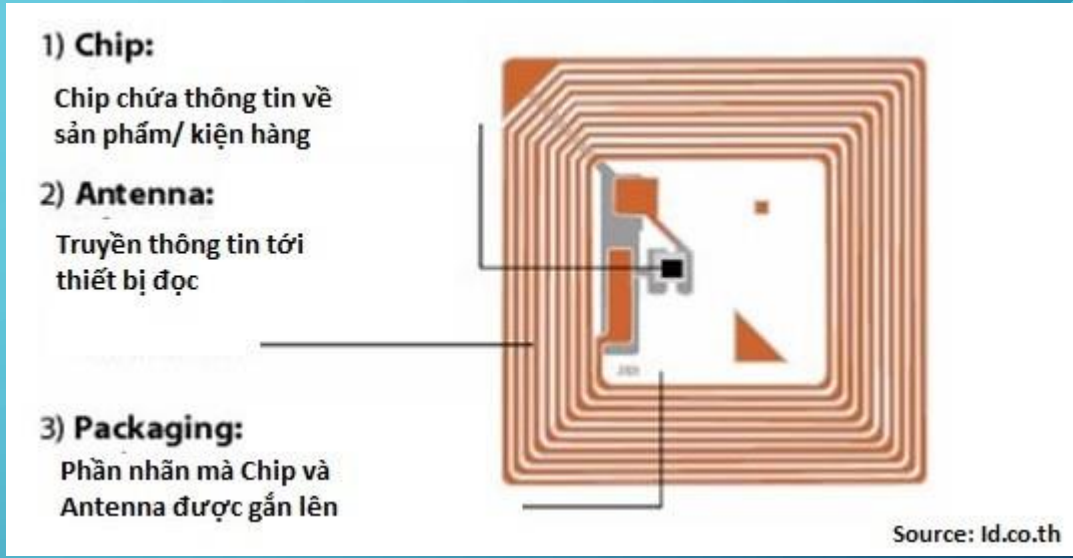
Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.5 Đọc thẻ RFID

4.5.2 Hệ thống RFID:

4.5.2.1Tag (Nhãn)

Passive tag



	Active tag	Passive tag
Nguồn điện	Nhãn có nguồn điện	Nhãn không có nguồn điện mà được cung cấp bởi thiết bị đọc (Reader)
Sử dụng pin	Có	Không
Khả năng phát tần số	Liên tục	Chỉ trong phạm vi giới hạn của thiết bị đọc
Khoảng cách có thể đọc	Trong vòng 100m hoặc hơn	Khoảng 3m đổ lại
Yêu cầu của tín hiệu thiết bị đọc	Tín hiệu yếu cũng ok	Tín hiệu phải mạnh
Dung lượng	Lớn và có thể lưu trữ thông tin phức tạp	Dung lượng nhỏ, thông tin đơn giản
		source: assettrackit.com

Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.5 Đọc thẻ RFID

4.5.2 Hệ thống RFID:

4.5.2.2 Antenna: Thiết thu nhận tín hiệu phát ra từ tag

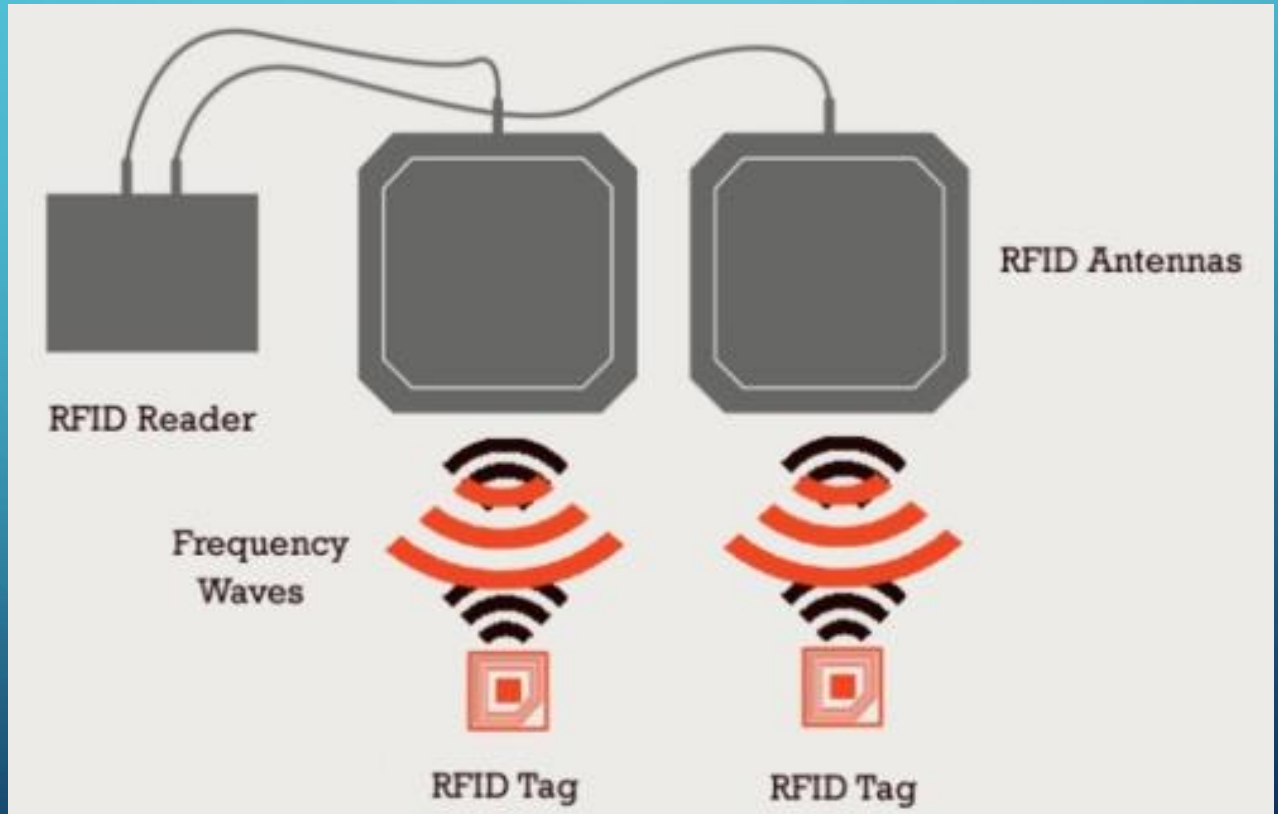


Figure 1. In monostatic systems, each antenna transmits and receives RF energy.

Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.5 Đọc thẻ RFID

4.5.2 Hệ thống RFID:

4.5.2.3 Reader: Khi nhận tín hiệu từ Antenna, thiết bị đọc sẽ đọc dữ liệu và cập nhật lên hệ thống

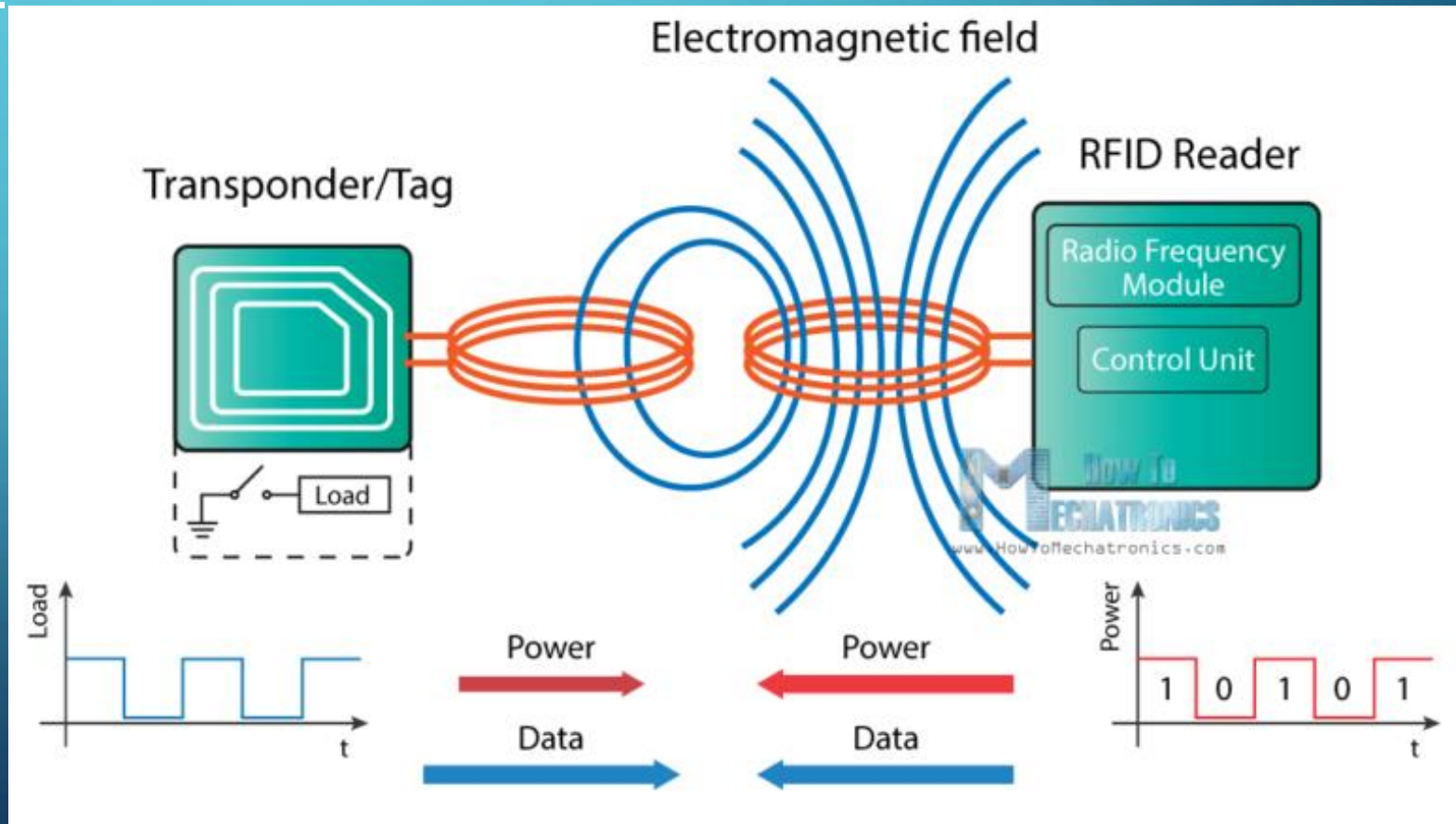
Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.5 Đọc thẻ RFID

4.5.3 Nguyên tắc hoạt động:

Đầu đọc RFID bao gồm một mô-đun tần số vô tuyến, bộ điều khiển và cuộn ăng ten tạo ra trường điện từ tần số cao.

Thẻ thường là một linh kiện thụ động, chỉ bao gồm ăng-ten và vi mạch điện tử, do đó, khi nó ở gần trường điện từ của bộ thu phát, do cảm ứng, một điện áp được tạo ra trong cuộn ăng ten của nó và điện áp phục vụ như là nguồn điện cho vi mạch



Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

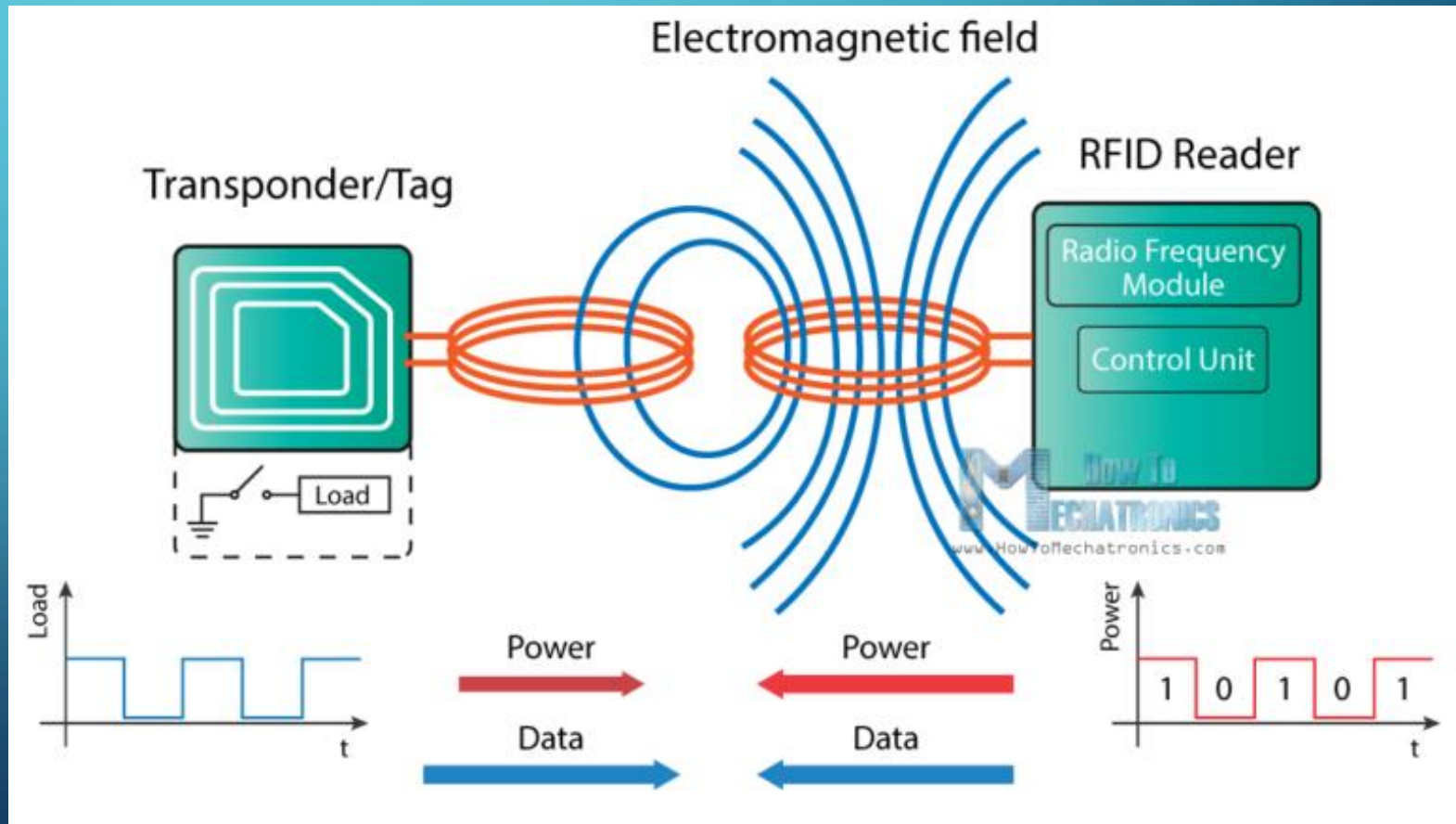
4.5 Đọc thẻ RFID

4.5.3 Nguyên tắc hoạt động:

Bật và tắt tải ở ăng-ten của thẻ sẽ ảnh hưởng đến mức tiêu thụ điện của ăng-ten của bộ đọc có thể được đo là sụt áp.

Sự thay đổi điện áp này sẽ được ghi lại dưới dạng 0 và 1 và đó là cách dữ liệu được truyền từ thẻ đến đầu đọc

Khi thẻ được cấp nguồn, nó có thể trích xuất thông tin được truyền từ đầu đọc và gửi tin nhắn trở lại đầu đọc, gọi là thao tác tải.



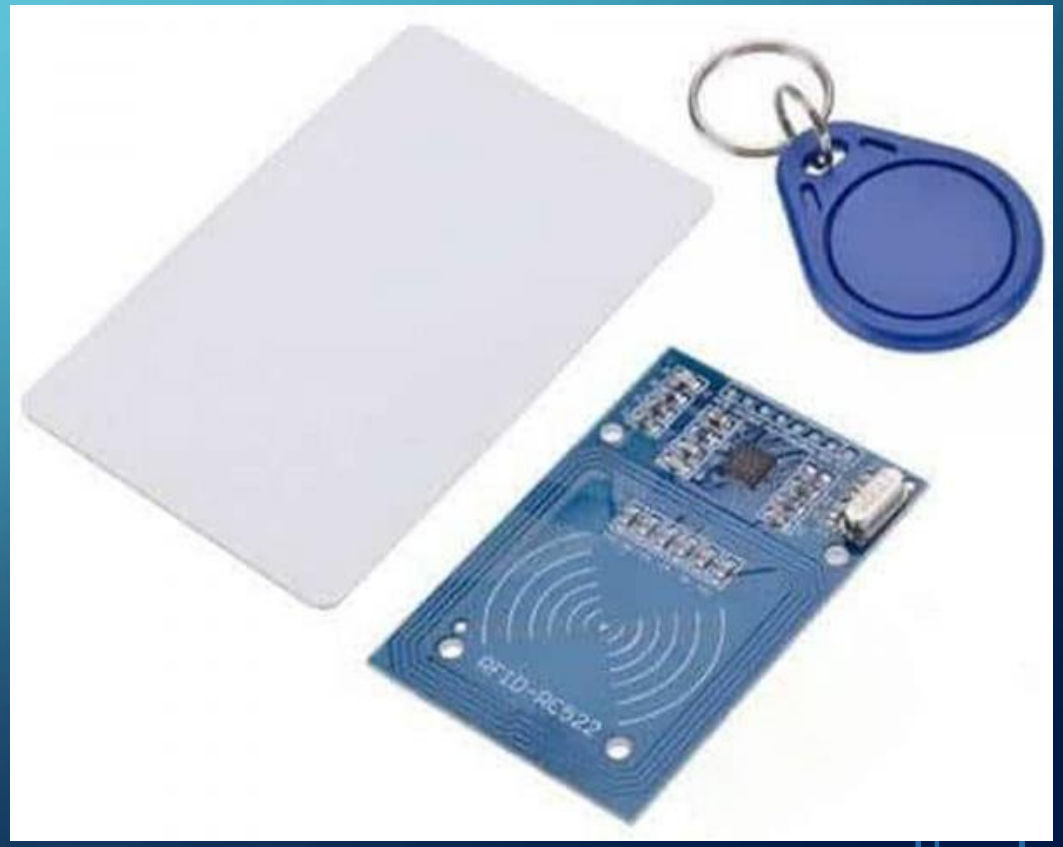
Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.5 Đọc thẻ RFID

4.5.4 Đọc thông tin từ thẻ RFID: Module đọc thẻ **RFID RC522 NFC 13.56Mhz** sử dụng IC **MFRC522** của Phillip dùng để đọc và ghi dữ liệu cho thẻ NFC tần số 13.56mhz,

4.5.4.1 Thông số:

- Nguồn: 3.3VDC, 13 – 26mA
- Dòng ở chế độ chờ: 1013mA
- Dòng ở chế độ nghỉ: <80uA
- Tần số sóng mang: 13.56MHz
- Khoảng cách hoạt động: 0～60mm（mifare1 card）
- Giao tiếp: SPI
- Tốc độ truyền dữ liệu: tối đa 10Mbit/s
- Các loại card RFID hỗ trợ: mifare1 S50, mifare1 S70, mifare UltraLight, mifare Pro, mifare Desfire
- Kích thước: 40mm × 60mm

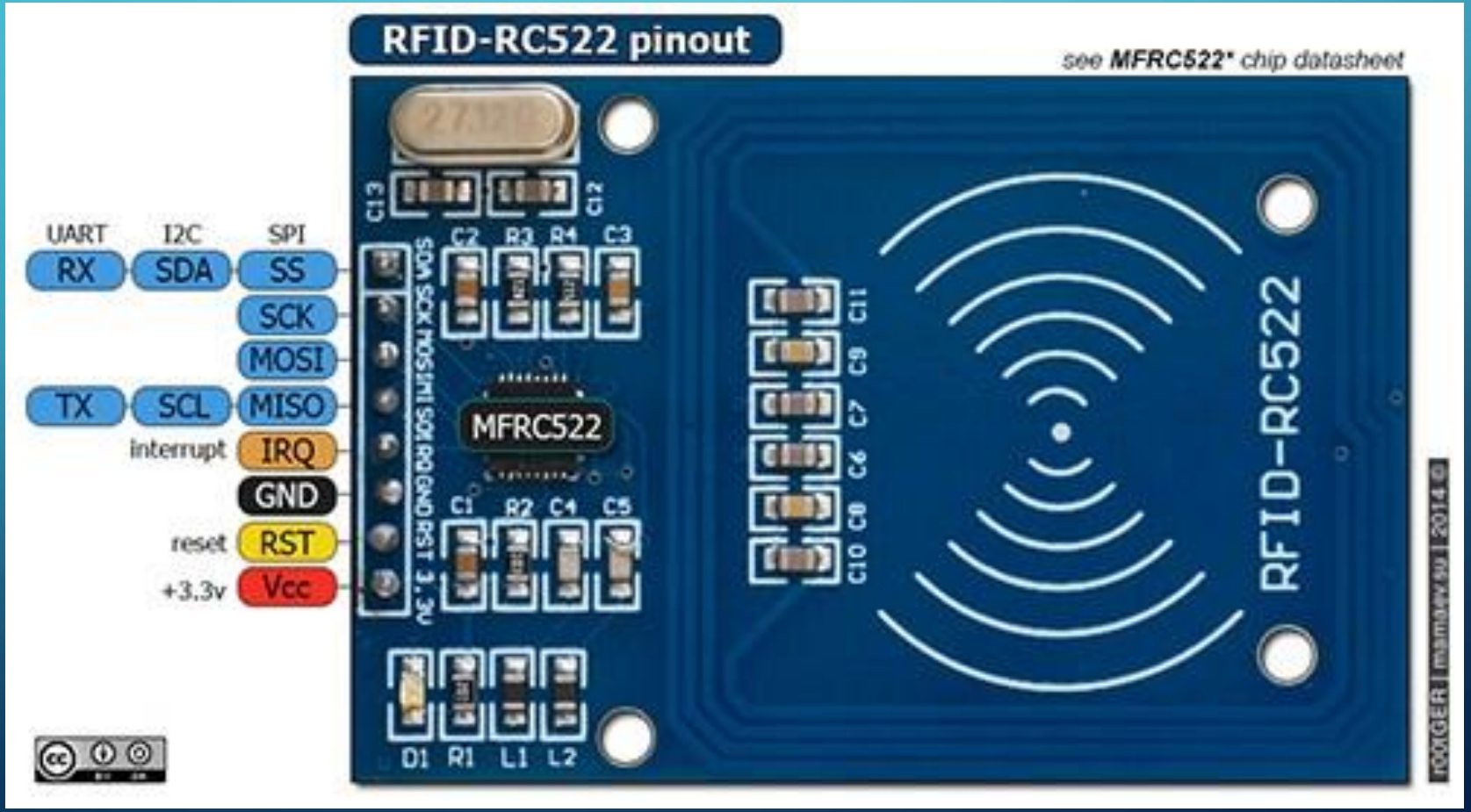


Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.5 Đọc thẻ RFID

4.5.4 Đọc thông tin từ thẻ RFID:

4.5.4.2 Sơ đồ chân:

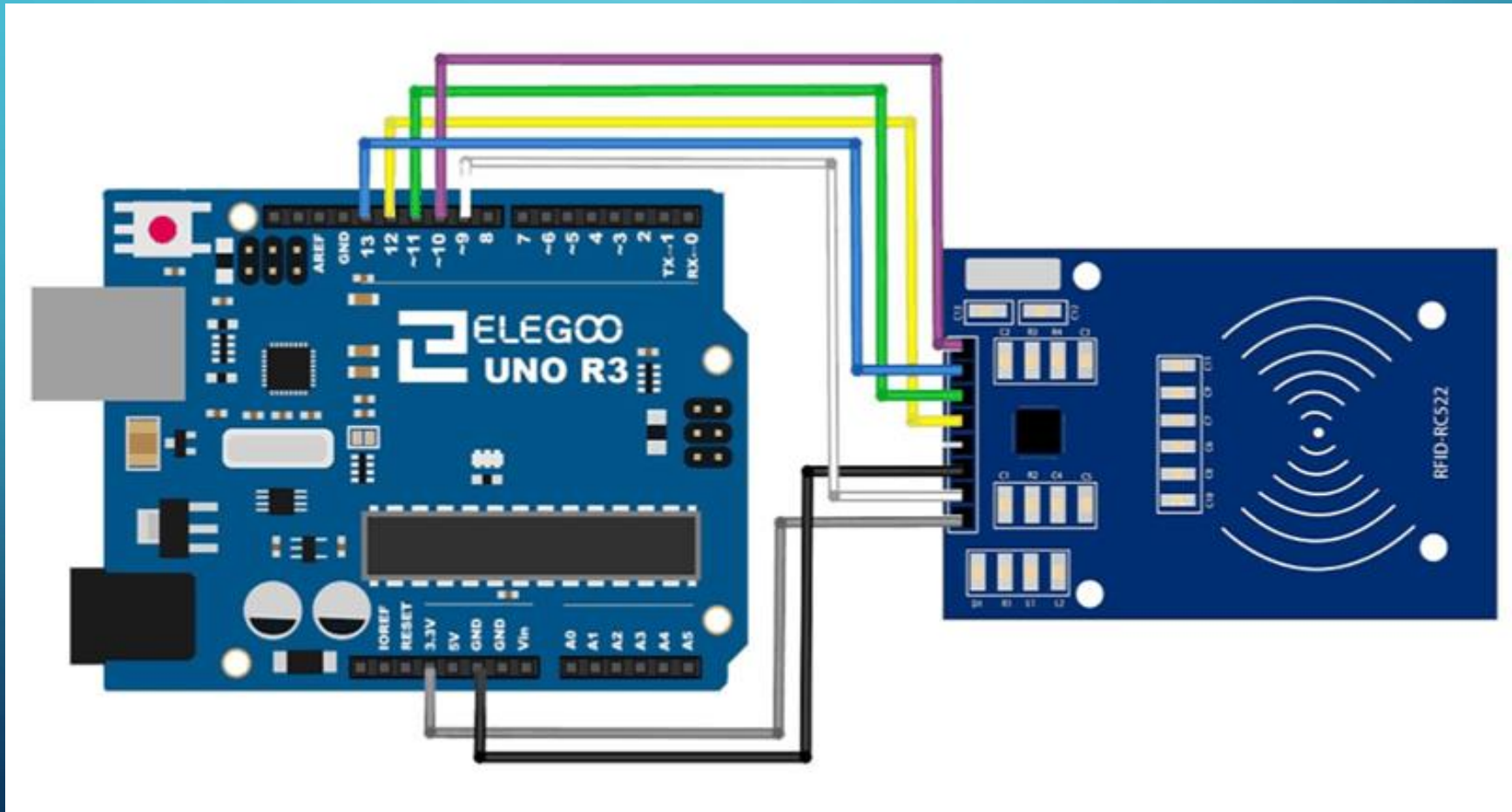


Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.5 Đọc thẻ RFID

4.5.4 Đọc thông tin từ thẻ RFID:

4.5.4.3 Sơ đồ kết nối:



Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.5 Đọc thẻ RFID

4.5.4 Đọc thông tin từ thẻ RFID:

4.5.4.3 Chương trình:

```
/*
*              UNO       MEGA
SDA (SS)     10          53
SCK          13          52
MOSI         11          51
MISO         12          50
GND          GND         GND
RST          9           9
3.3V         3.3V        3.3V
*/
```

Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.5 Đọc thẻ RFID

4.5.4 Đọc thông tin từ thẻ RFID:

4.5.4.3 Chương trình:

```
#include <SPI.h>
#include <MFRC522.h>

#define RST_PIN 9
#define SS_PIN 10
MFRC522 mfrc522(SS_PIN, RST_PIN);

void setup() {
 Serial.begin(9600);
 SPI.begin();
 mfrc522.PCD_Init();
 Serial.println("Lectura del UID");
}
```

4.5.4.3 Chương trình:

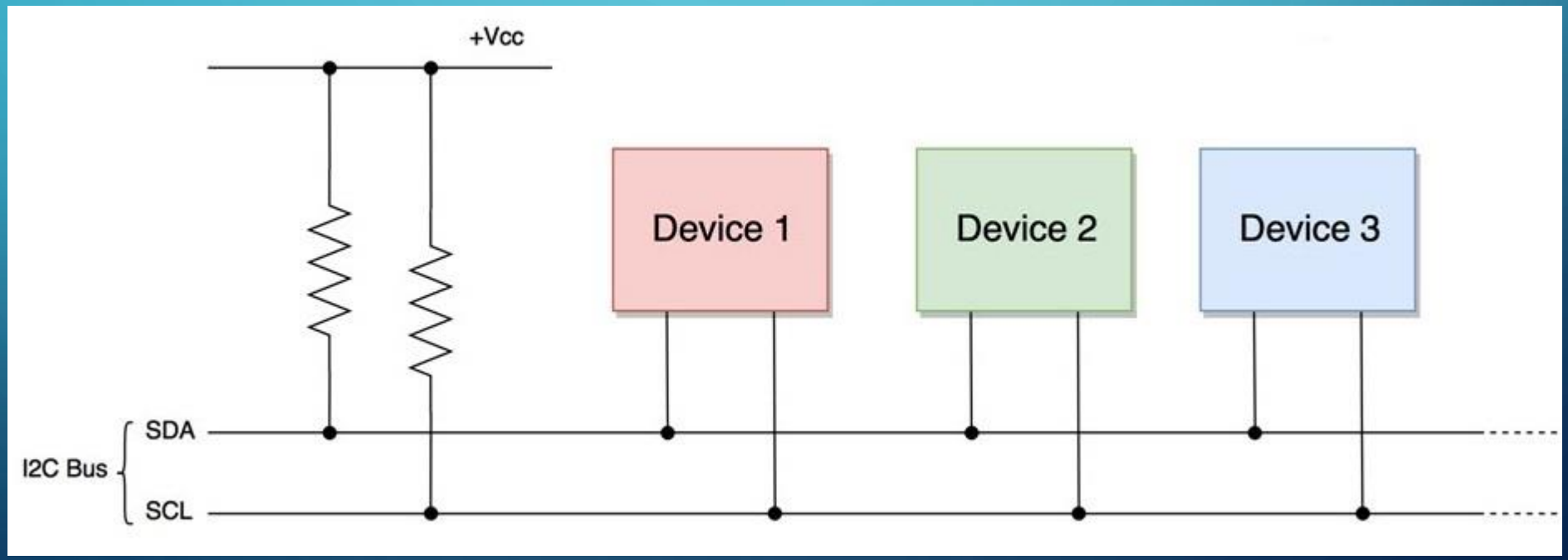
```
void loop()
{
  if(mfrc522.PICC_IsNewCardPresent())
  {
    if(mfrc522.PICC_ReadCardSerial())
    {
      Serial.print("Card UID:");
      for (byte i = 0; i < mfrc522.uid.size; i++)
      {
        Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
        Serial.print(mfrc522.uid.uidByte[i], HEX);
      }
      Serial.println();
      mfrc522.PICC_HaltA();
    }
  }
}
```

Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.7 Giao tiếp I2C

4.7.1 I2C là gì?

I2C - viết tắt của "**Inter-Integrated Circuit**" - là giao thức **giao tiếp nối tiếp đồng bộ** được Philips Semiconductors phát triển để truyền dữ liệu giữa một bộ xử lý trung tâm với nhiều IC trên cùng một board mạch chỉ sử dụng hai đường truyền tín hiệu



Giao tiếp nối tiếp đồng bộ: *các bit dữ liệu được truyền từng bit một theo các khoảng thời gian đều đặn được thiết lập bởi một xung clock tham chiếu*

Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.7 Giao tiếp I2C

4.7.2 Đặc điểm của giao thức giao tiếp I2C:

- Chỉ cần 2 đường dây chung để điều khiển bất kỳ thiết bị / IC nào trên mạng I2C
- Không cần thỏa thuận trước về tốc độ truyền dữ liệu như trong giao tiếp UART ➔ tốc độ truyền dữ liệu có thể được điều chỉnh khi cần thiết
- Cơ chế đơn giản để xác thực dữ liệu được truyền
- Sử dụng hệ thống địa chỉ 7 bit để xác định một thiết bị / IC cụ thể trên mạng I2C
- Các mạng I2C dễ dàng mở rộng: thiết bị mới có thể được kết nối đơn giản với hai đường dây chung I2C

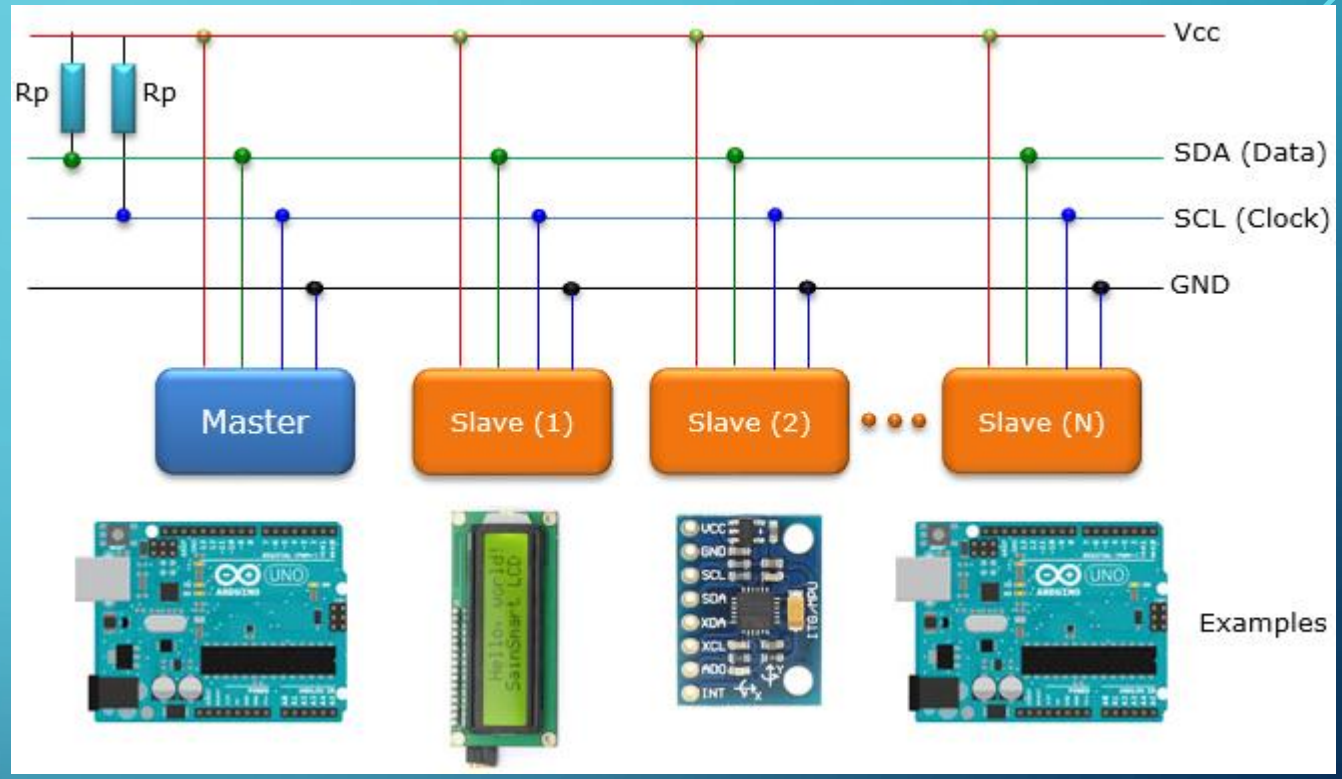
Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.7 Giao tiếp I2C

4.7.3 Bus vật lý I2C

SCL: Serial Clock Line → đường xung clock đồng bộ

SDA: Serial Data Line → đường truyền dữ liệu



Ở bất cứ thời điểm nào thì chỉ có duy nhất một **thiết bị Master** ở trang thái hoạt động trên bus I2C

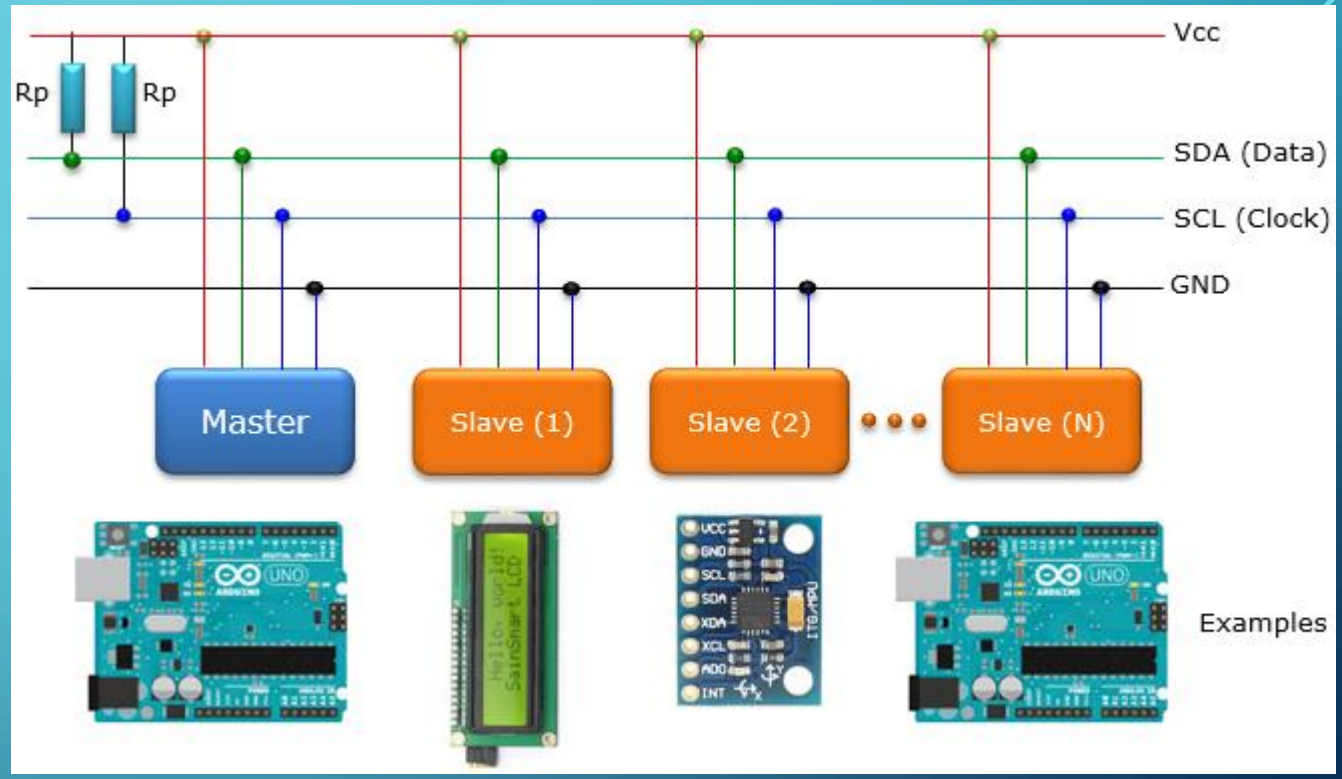
→ điều khiển đường tín hiệu đồng hồ SCL
và quyết định hoạt động nào sẽ được thực hiện trên đường dữ liệu SDA

Thiết bị Slave đáp ứng các hướng dẫn từ thiết bị Master này

Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.7 Giao tiếp I2C

4.7.3 Bus vật lý I2C



Để phân biệt giữa nhiều thiết bị Slave được kết nối với cùng một bus I2C, mỗi thiết bị Slave được gán một **địa chỉ vật lý 7-bit cố định**

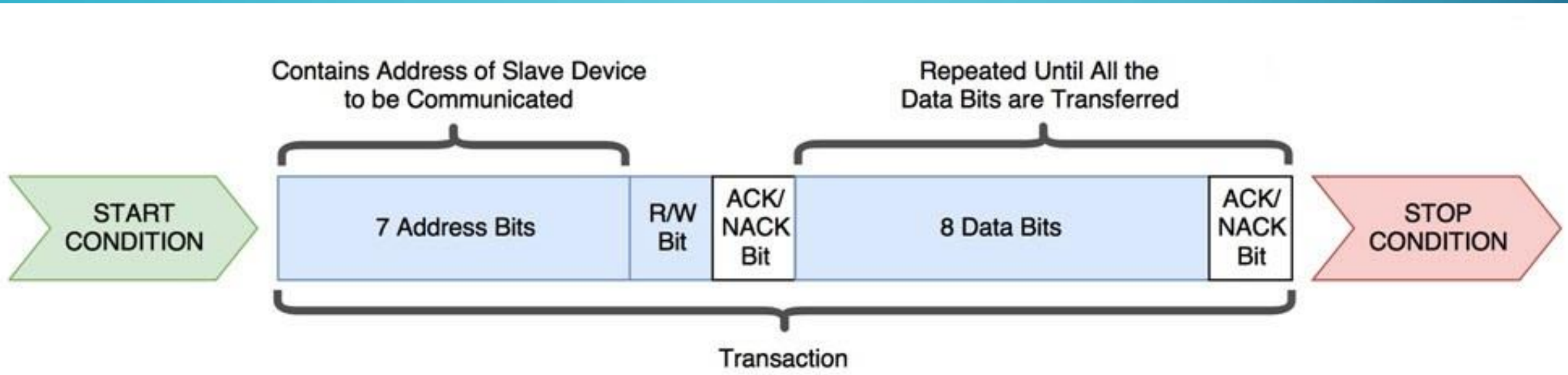
➔ Khi thiết bị Master muốn **truyền dữ liệu đến** hoặc **nhận dữ liệu** từ một thiết bị Slave, nó xác định địa chỉ thiết bị Slave cụ thể này trên đường SDA và sau đó tiến hành truyền dữ liệu

Tất cả các thiết bị Slave khác không phản hồi trừ khi địa chỉ của chúng được chỉ định bởi thiết bị Master trên dòng SDA

Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.7 Giao tiếp I2C

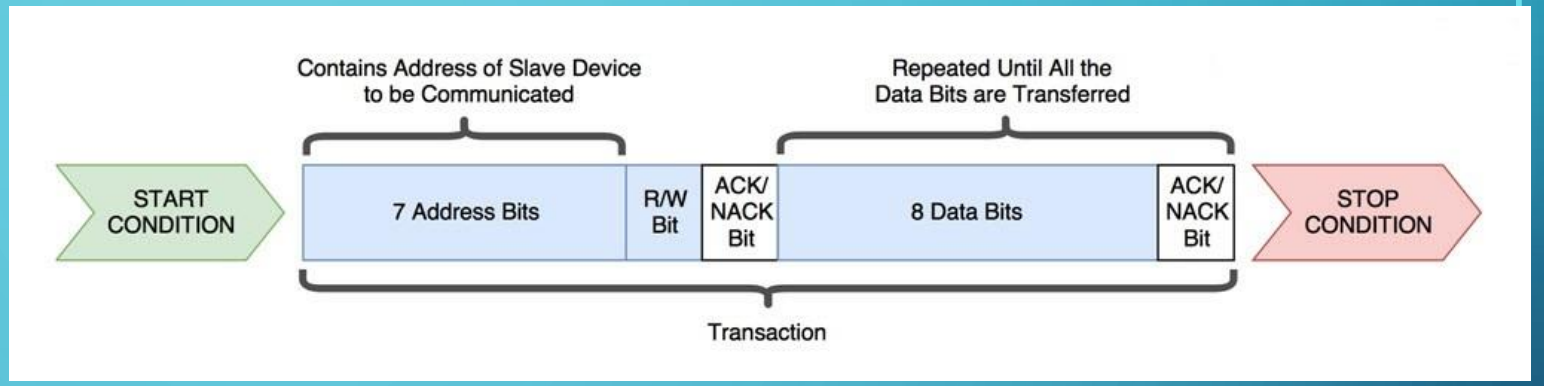
4.7.4 Giao thức truyền dữ liệu



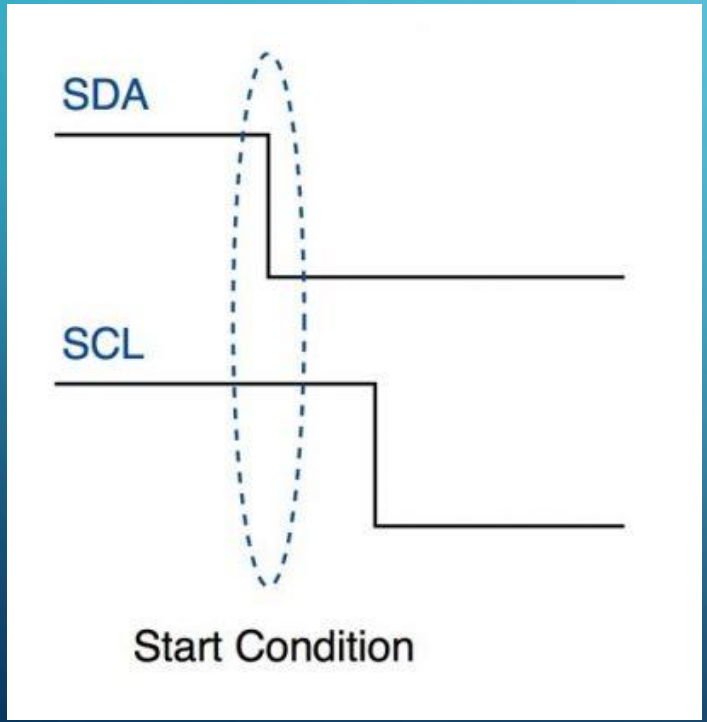
Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.7 Giao tiếp I2C

4.7.4 Giao thức truyền dữ liệu



Start Condition

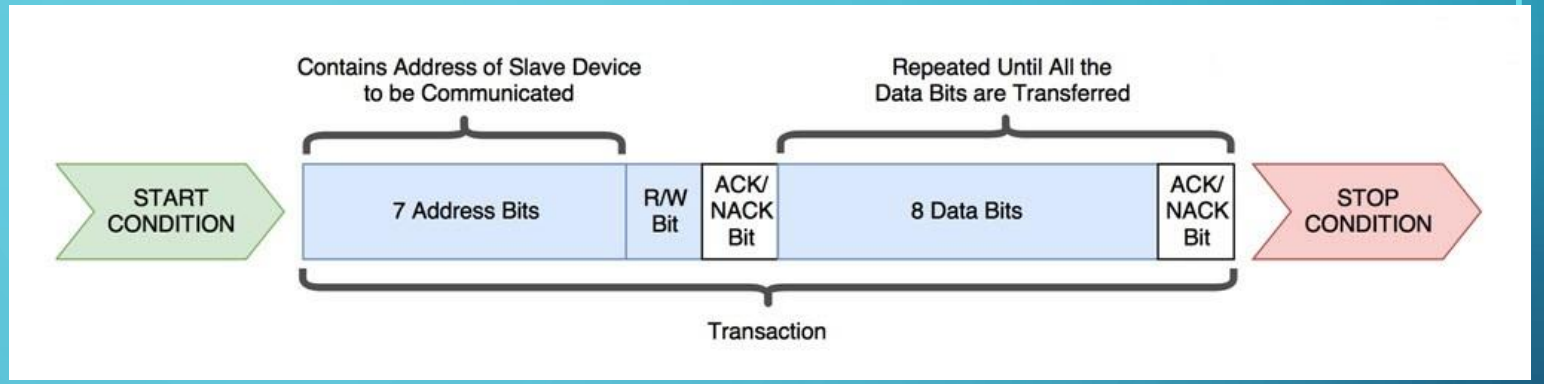


➔ tất cả các thiết bị Slave chuyển sang trạng thái hoạt động

Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.7 Giao tiếp I2C

4.7.4 Giao thức truyền dữ liệu



Khối địa chỉ

7 bit địa chỉ của thiết bị Slave cần giao tiếp

➔Tất cả các thiết bị Slave trên bus I2C so sánh các bit địa chỉ này với địa chỉ của chúng

Bit Read / Write

xác định hướng truyền dữ liệu

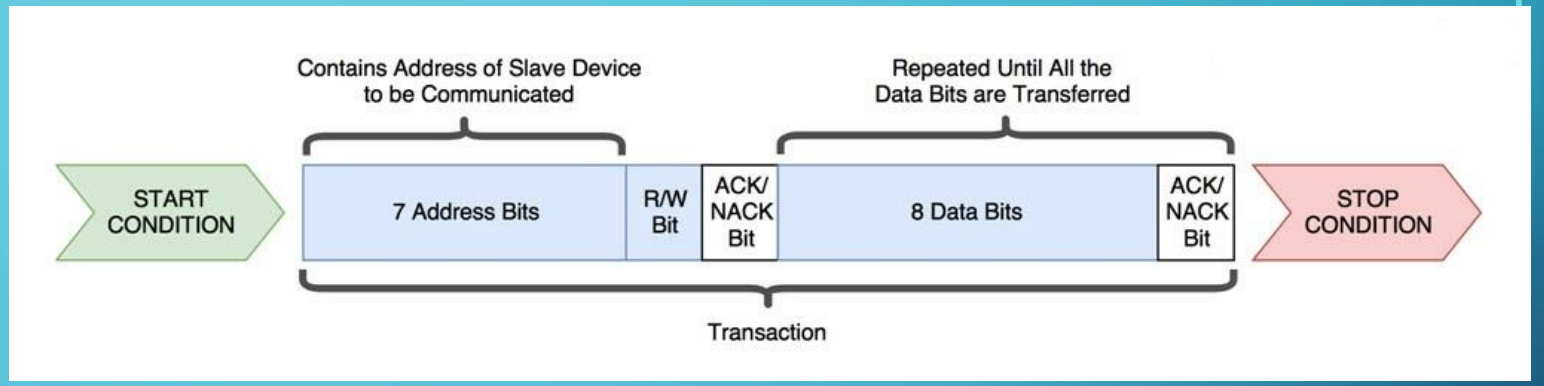
0: thiết bị Master cần **gửi** dữ liệu đến thiết bị Slave

1: thiết bị Master cần **nhận** dữ liệu từ thiết bị Slave

Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.7 Giao tiếp I2C

4.7.4 Giao thức truyền dữ liệu



Bit ACK / NACK

viết tắt của **Acknowledged/Not-Acknowledged**

1: giá trị mặc định.
0: được thiế bị Slave đặt lại khi địa chỉ vật lý của thiết bị Slave trùng với địa chỉ do thiết bị Master phát

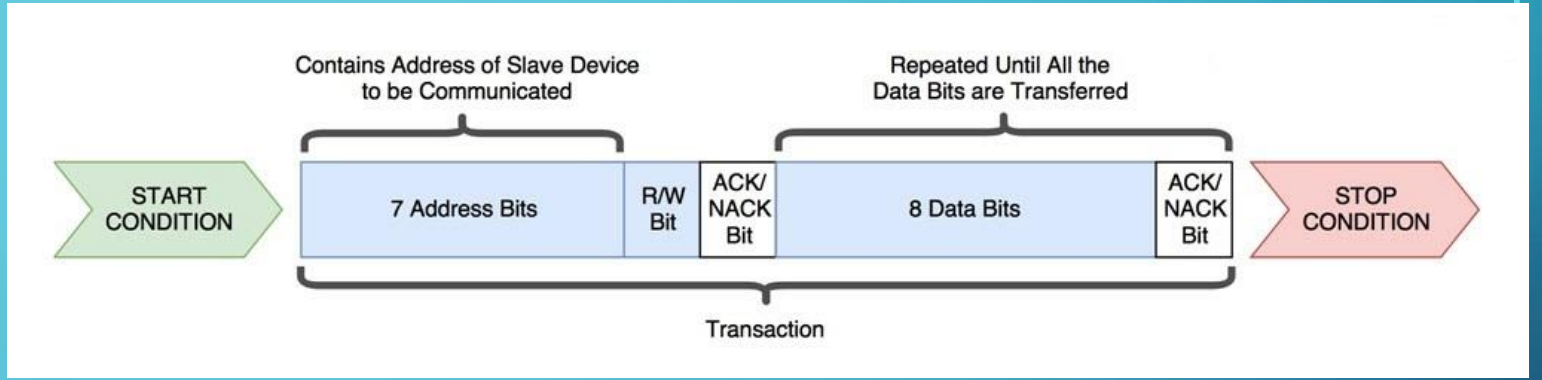
Khối dữ liệu có 8 bit được thiết lập bởi bên gửi
là các bit dữ liệu cần truyền tới bên nhận

Khối này được theo sau bởi một bit ACK / NACK
*(được set thành '**0**' bởi bên nhận nếu nó nhận thành công dữ liệu)*

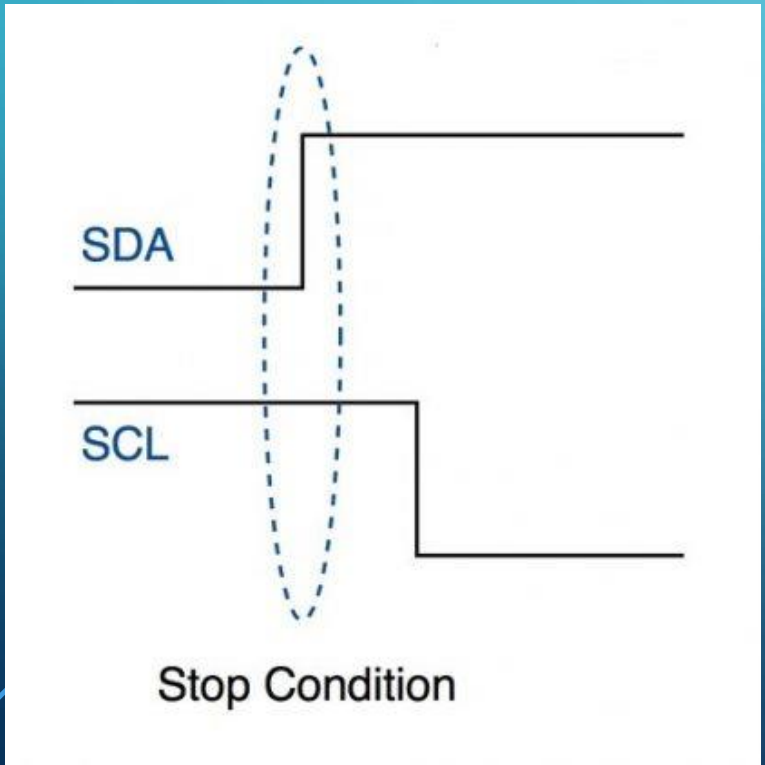
Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.7 Giao tiếp I2C

4.7.4 Giao thức truyền dữ liệu



Stop condition

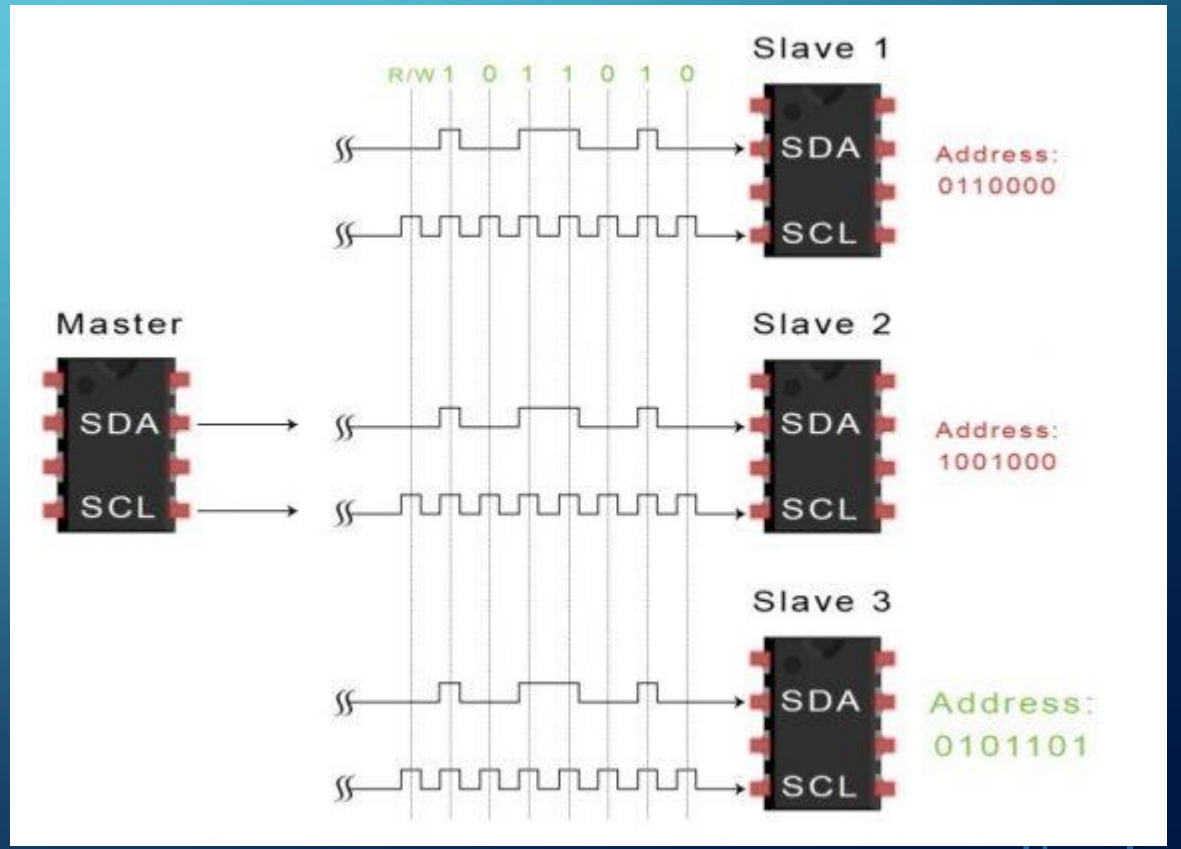


Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.7 Giao tiếp I2C

4.7.5 Quy trình hoạt động:

- Thiết bị Master gửi điều kiện bắt đầu đến tất cả các thiết bị Slave
- Thiết bị Master gửi **7 bit địa chỉ** của thiết bị Slave muốn giao tiếp cùng với bit Read/Write



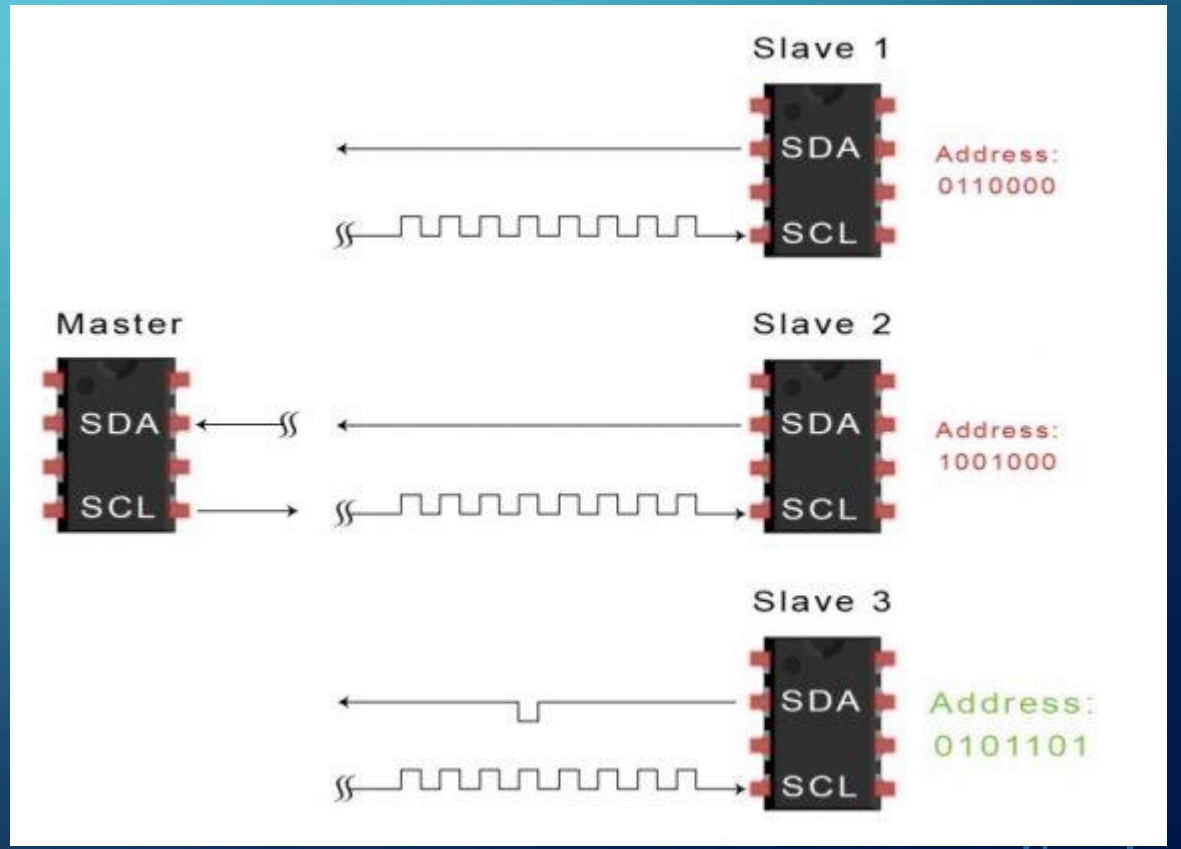
Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.7 Giao tiếp I2C

4.7.5 Quy trình hoạt động:

- Mỗi thiết bị Slave so sánh địa chỉ

  - Nếu địa chỉ trùng khớp, Slave gửi về bit ACK bằng cách kéo đường SDA xuống thấp ➔ bit **ACK / NACK = 0**
  - Nếu địa chỉ không khớp, đường SDA ở mức cao ➔ bit **ACK / NACK = 1**



Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.7 Giao tiếp I2C

4.7.5 Quy trình hoạt động:

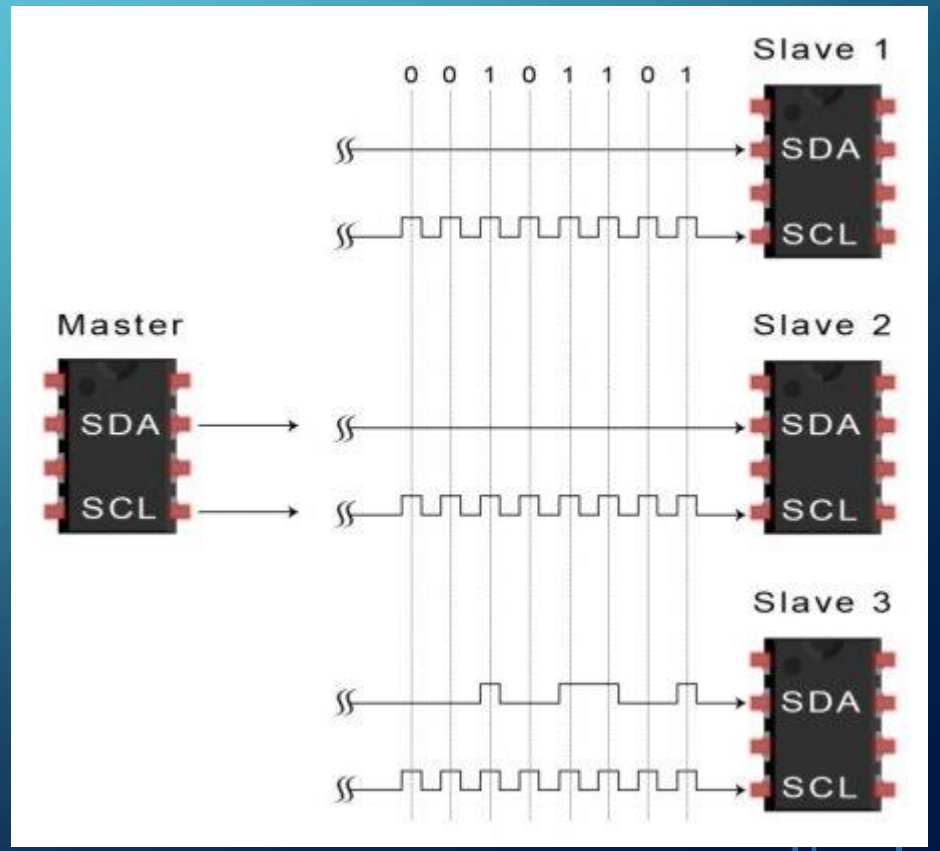
➢ Thiết bị Master gửi hoặc nhận khung dữ liệu

- Nếu thiết bị Master muốn gửi dữ liệu đến thiết bị Slave

  ➔ bit **Read / Write = 0**

- Nếu thiết bị Master đang nhận dữ liệu từ thiết bị Slave

  ➔ bit **Read / Write = 1**



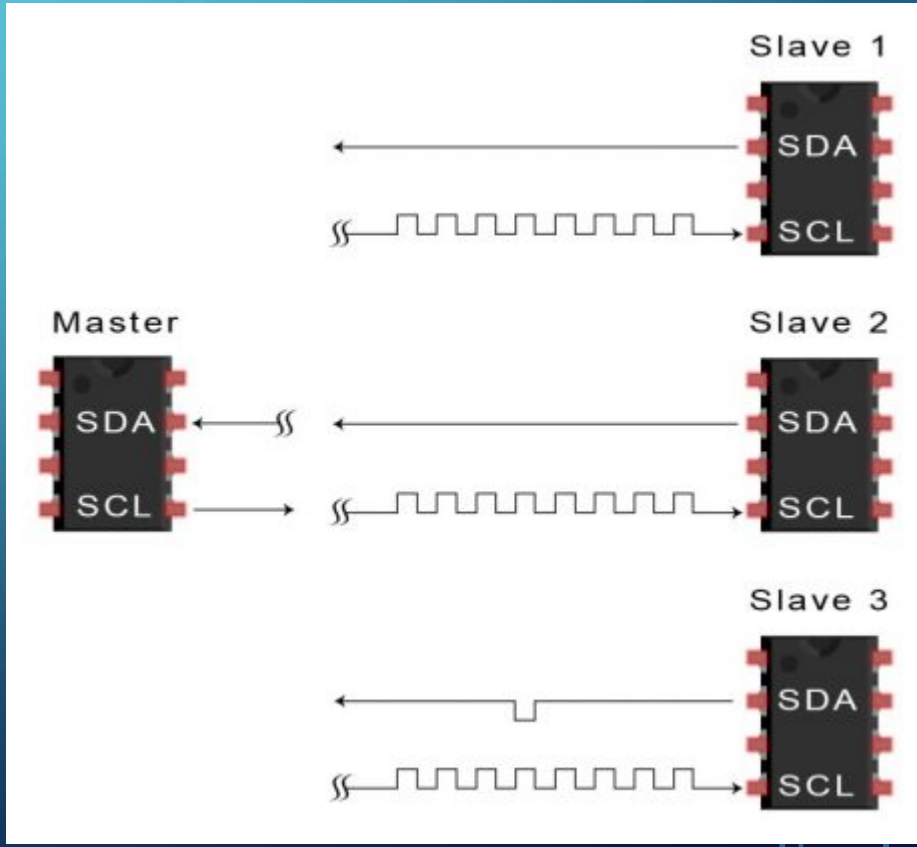
Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.7 Giao tiếp I2C

4.7.5 Quy trình hoạt động:

- Nếu khung dữ liệu được thiết bị Slave nhận được thành công

➔ Slave sẽ thiết lập bit **ACK / NACK =0**, báo hiệu cho thiết bị Master tiếp tục

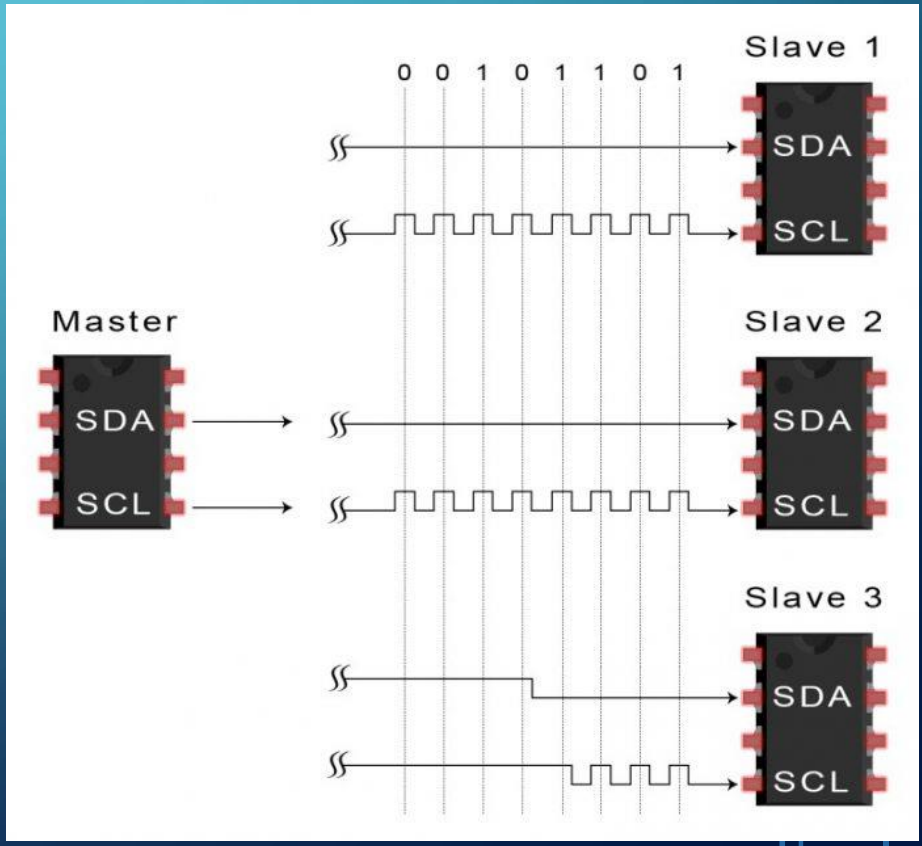


Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.7 Giao tiếp I2C

4.7.5 Quy trình hoạt động:

- Sau khi tất cả dữ liệu được gửi đến thiết bị Slave, thiết bị Master gửi điều kiện dừng để báo hiệu cho tất cả các thiết bị Slave biết rằng việc truyền dữ liệu đã kết thúc

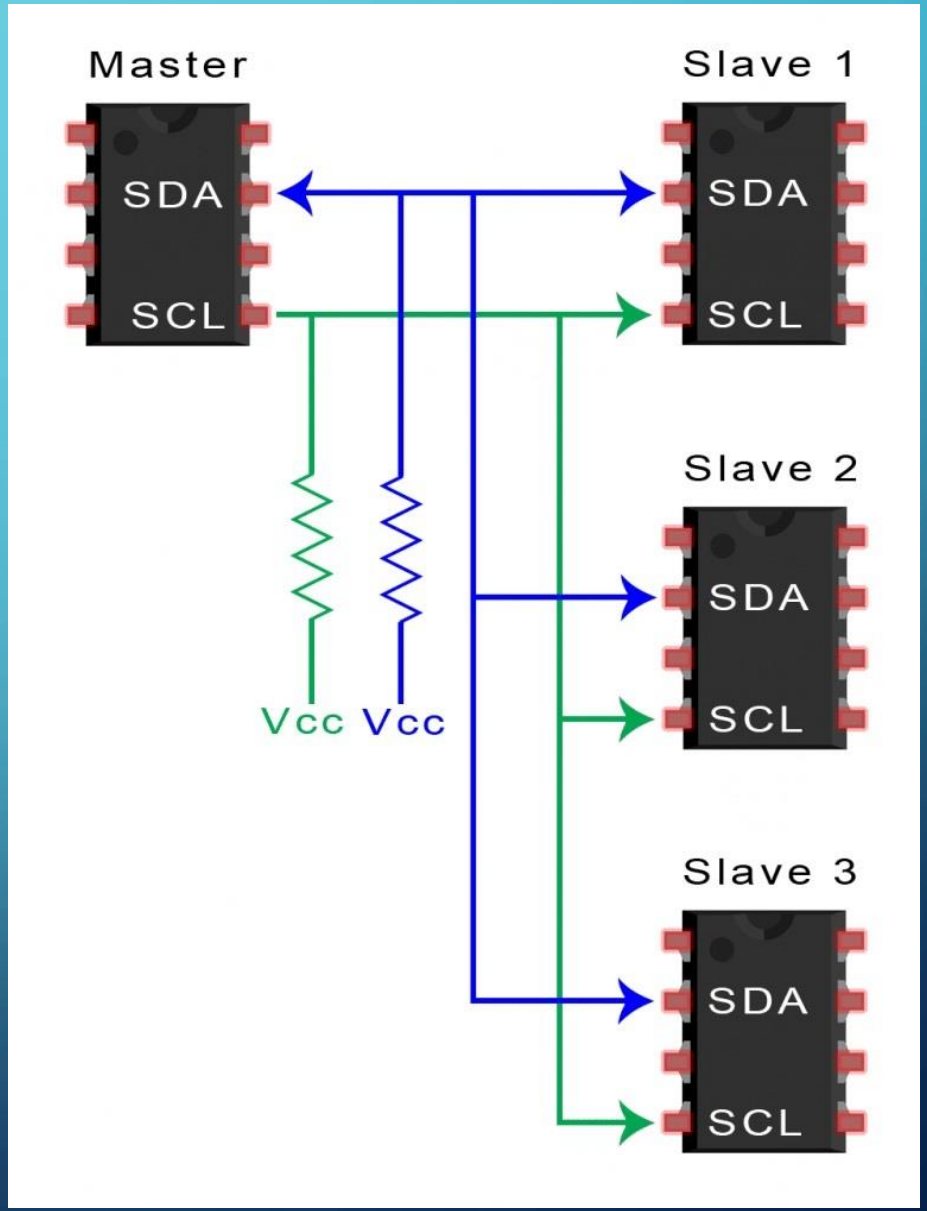


Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.7 Giao tiếp I2C

4.7.6 Các chế độ hoạt động

Giao tiếp I2C một Master nhiều Slave

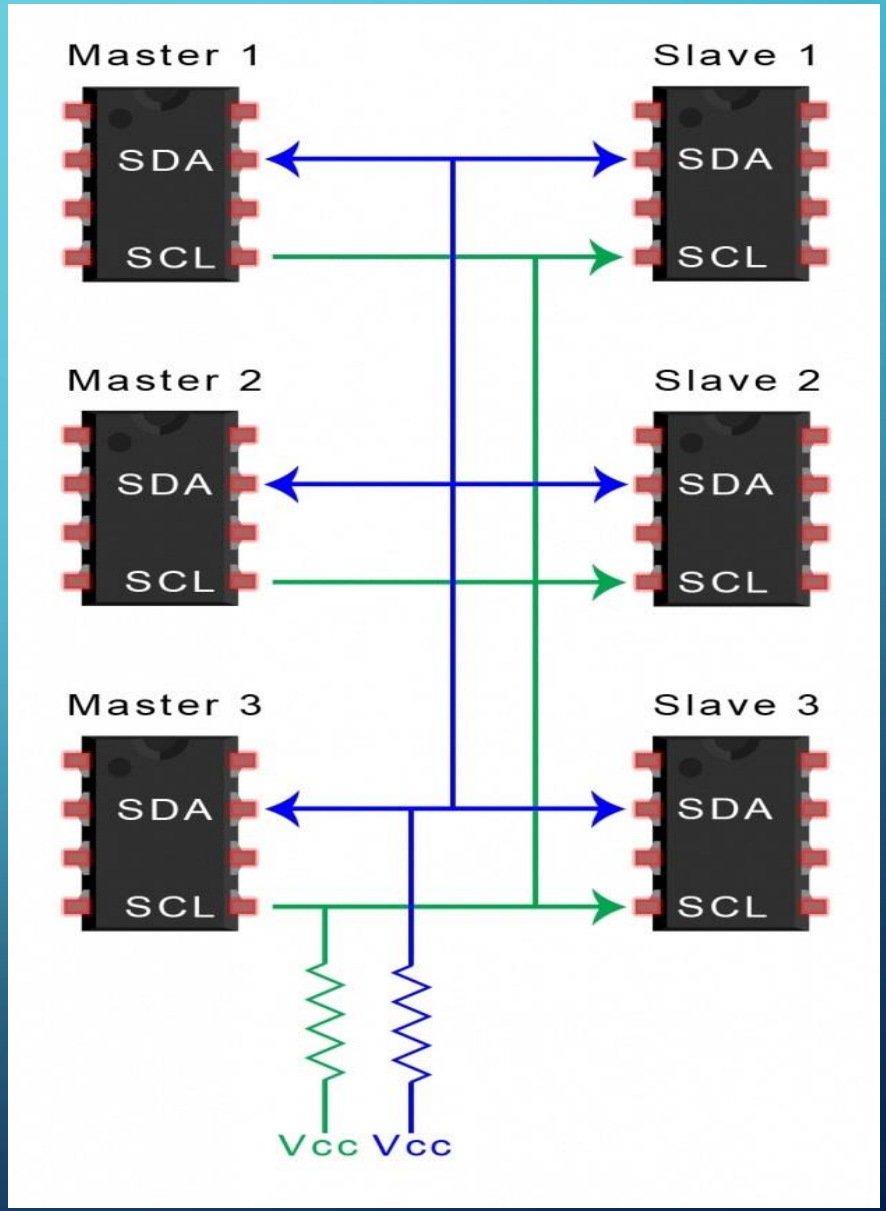


Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.7 Giao tiếp I2C

4.7.6 Các chế độ hoạt động

Giao tiếp I2C nhiều Master nhiều Slave



Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.8 Cảm biến gia tốc góc nghiêng MPU6050

4.8.1 IMU (Inertial Measurement Unit)

là một con chip để đo những chuyển động nghiêng, xoay…

Module IMU thường gồm có 2 loại cảm biến:

- **Accelerometer:** là một cảm biến đo gia tốc của bản thân module và có 3 trục xyz ứng với 3 chiều không gian

  Lưu ý: *giá trị thực khi đo sẽ bao gồm cả trọng lực*

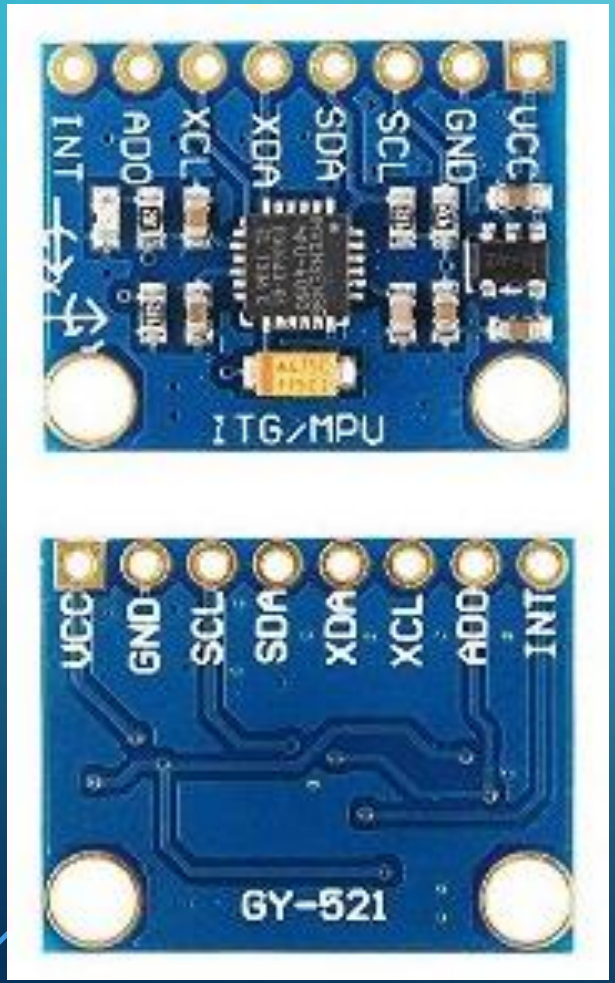
- **Gyroscope:** là loại cảm biến đo tốc độ quay của nó quanh một trục, có 3 trục xyz

Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.8 Cảm biến gia tốc góc nghiêng MPU6050

4.8.2 Cảm biến MPU6050

là cảm biến của hãng InvenSense tích hợp 6 trục cảm biến:



- Con quay hồi chuyển 3 trục (Gyroscope).
- Cảm biến gia tốc 3 chiều (Accelerometer)

Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.8 Cảm biến gia tốc góc nghiêng MPU6050

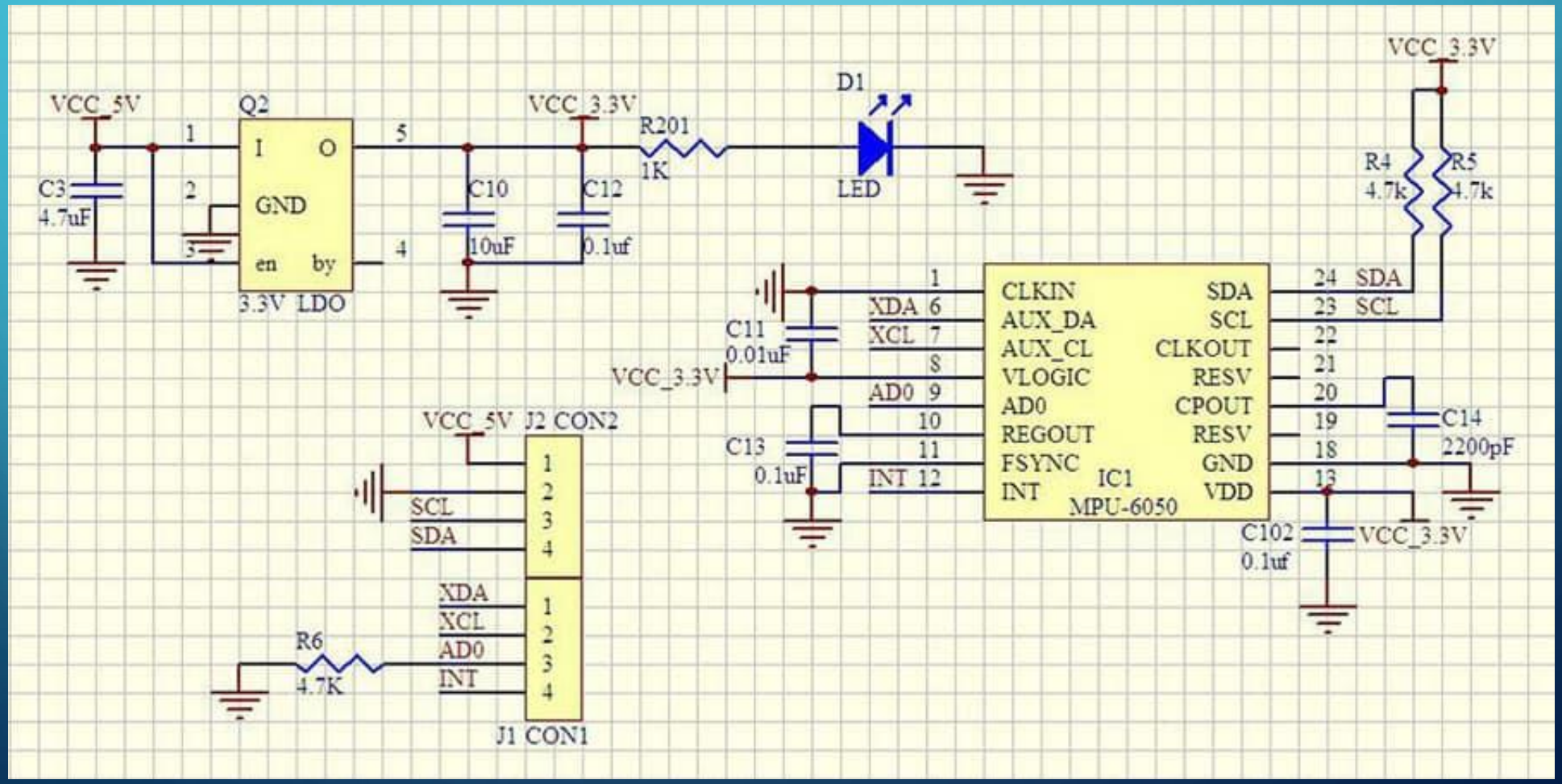
4.8.3 Thông số kỹ thuật:

- Điện áp sử dụng: 3~5VDC
- Điện áp giao tiếp: 3~5VDC
- Chuẩn giao tiếp: I2C
- Giá trị Gyroscopes trong khoảng: +/- 250 500 1000 2000 degree/sec
- Giá trị Acceleration trong khoảng: +/- 2g, +/- 4g, +/- 8g, +/- 16g

Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.8 Cảm biến gia tốc góc nghiêng MPU6050

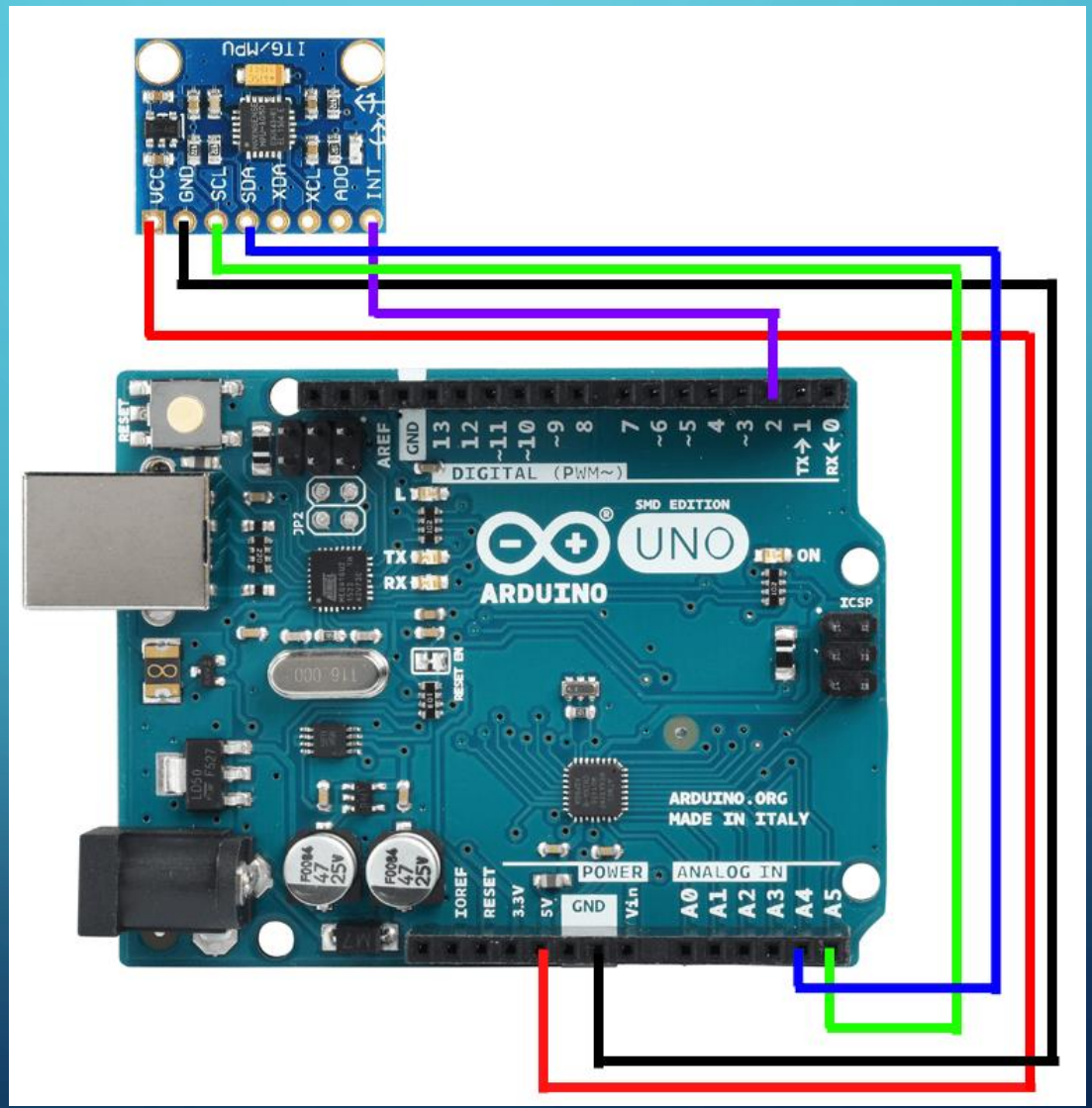
4.8.4 Sơ đồ nguyên lý:



Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.8 Cảm biến gia tốc góc nghiêng MPU6050

4.8.5 Sơ đồ kết nối:



Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.8 Cảm biến gia tốc góc nghiêng MPU6050

4.8.6 Chương trình:

/**********

Kết nối:

MPU6050	UNO R3	MEGA
VIN	5V	5V
GND	GND	GND
SCL	A5	SCL
SDA	A4	SDA

Nạp code mở Serial Monitor chọn No line ending, baud 9600.

**********/

Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.8 Cảm biến gia tốc góc nghiêng MPU6050

4.8.6 Chương trình:

```
#include <Wire.h>                 //Thư viện giao tiếp I2C
const int MPU_addr=0x68;  //Địa chỉ I2C của MPU6050
int16_t AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ; //Các biến lấy giá trị

void setup()
{
  Wire.begin();
  Wire.beginTransmission(MPU_addr); //Gửi tín hiệu đến địa chỉ MPU
  Wire.write(0x6B);
  Wire.write(0);                                  //Đưa về 0 để bật MPU
  Wire.endTransmission(true);
  Serial.begin(9600);
}
```

Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.8 Cảm biến gia tốc góc nghiêng MPU6050

4.8.6 Chương trình:

```
void loop() {
 Wire.beginTransmission(MPU_addr);     //Gửi tín hiệu đến địa chỉ MPU
 Wire.write(0x3B);                     //Gửi giá trị lên MPU
 Wire.endTransmission(false);
 Wire.requestFrom(MPU_addr,14,true);   //Đề nghị thanh ghi của MPU

 //Đọc dữ liệu
 AcX=Wire.read()<<8|Wire.read();  //Gia tốc trục x
 AcY=Wire.read()<<8|Wire.read();  //Gia tốc trục y
 AcZ=Wire.read()<<8|Wire.read();  //Gia tốc trục z
 GyX=Wire.read()<<8|Wire.read();  //Góc nghiêng trục x
 GyY=Wire.read()<<8|Wire.read();  //Góc nghiên trục y
 GyZ=Wire.read()<<8|Wire.read();  //Góc nghiên trục z
```

Chương 4: CÁC LOẠI CẢM BIẾN THÔNG DỤNG

4.8 Cảm biến gia tốc góc nghiêng MPU6050

4.8.6 Chương trình:

```
//Xuất ra Serial
  Serial.print("AcX = "); Serial.print(AcX);
  Serial.print(" | AcY = "); Serial.print(AcY);
  Serial.print(" | AcZ = "); Serial.print(AcZ);
  Serial.print(" | GyX = "); Serial.print(GyX);
  Serial.print(" | GyY = "); Serial.print(GyY);
  Serial.print(" | GyZ = "); Serial.println(GyZ);
  delay(333);                                    //Chờ (1/3)s
}
```