

Bài mở đầu

Bài 1 Giao tiếp với led đơn.

Bài 2 Giao tiếp với nút nhấn.

Bài 3 Giao tiếp với rơ le,speaker.

Bài 4 Giao tiếp với LCD, giới thiệu về công cụ debug Serial().

Bài 5 Giao tiếp với led 7 đoạn

Bài 6 ADC-đọc tín hiệu một số loại cảm biến.

Bài 7 Ngắt ngoài .

Bài 8 Đọc cảm biến nhiệt độ, độ ẩm DHT11,cảm biến khoảng cách.

Bài 9 Giao tiếp I2C -đọc thời gian thực.

Bài 10 PWM -điều chỉnh độ sáng của bóng đèn.

Bài 11 TIMER-Ngắt timer.

Bài 12 Điều khiển động cơ DC

Bài 13 Điều khiển động cơ servo

Bài 14 Điều khiển động cơ bước

Bài 15 Điều khiển và đo tốc độ động cơ DC sử dụng encoder

Bài 16 Giao tiếp SPI - giao tiếp thẻ từ RFID.

Bài 17 UART-giao tiếp giữa 2 arduino

Bài 18 Điều khiển từ xa bằng RF

Bài 19 Điều khiển từ xa bằng hồng ngoại

Bài 20 Điều khiển từ xa bằng bluetooth

Bài 21 Thiết kế mạch trên Altium Designer.

BÀI MỞ ĐẦU: TỔNG QUAN VỀ ARDUINO

Giới thiệu về arduino

Arduino là một board mạch vi xử lý, dùng để xử lý thu thập tín hiệu từ các cảm biến, và xuất ra các tín hiệu điều khiển để điều khiển các cơ cấu chấp hành. 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.

Một số board arduino thông dụng:



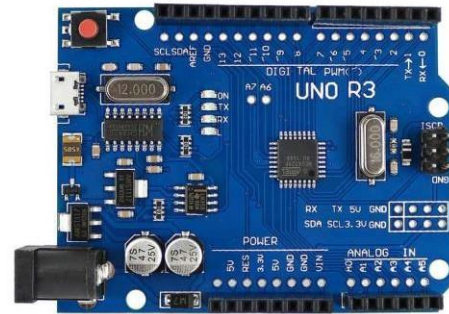
Arduino pro mini



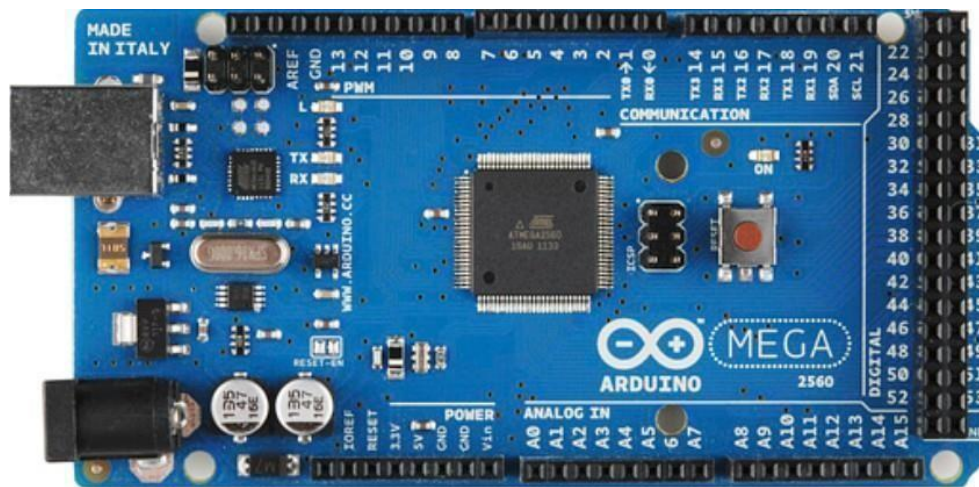
Arduino nano



Arduino uno chip cắm



Arduino uno chip dán

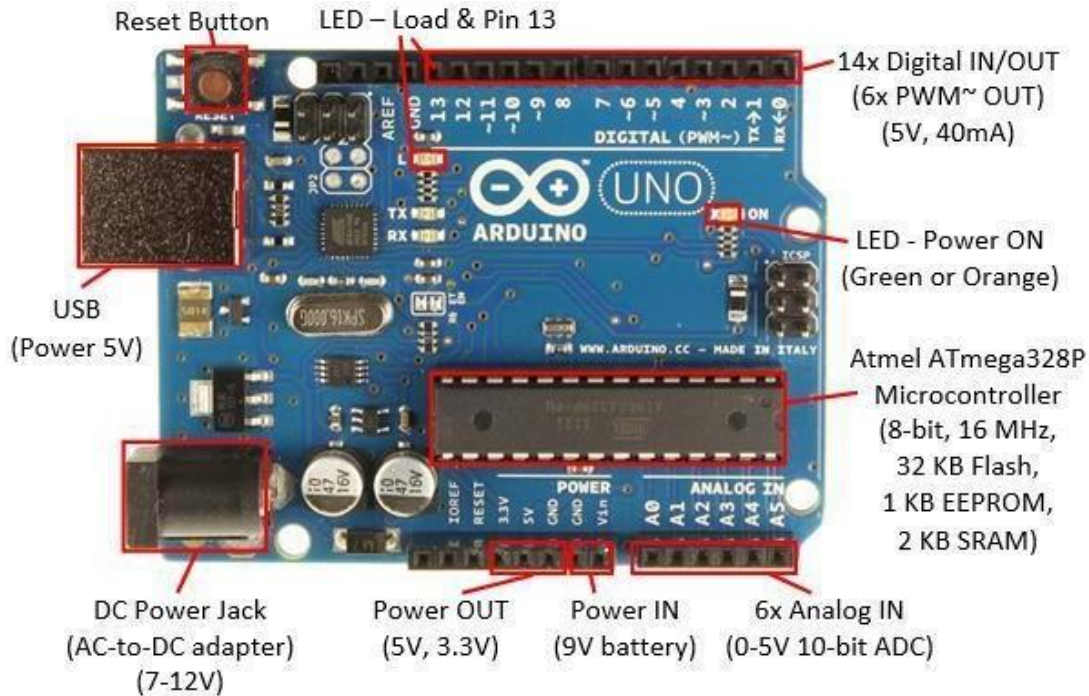


Arduino mega 2560

Một vài thông số của Arduino UNO R3

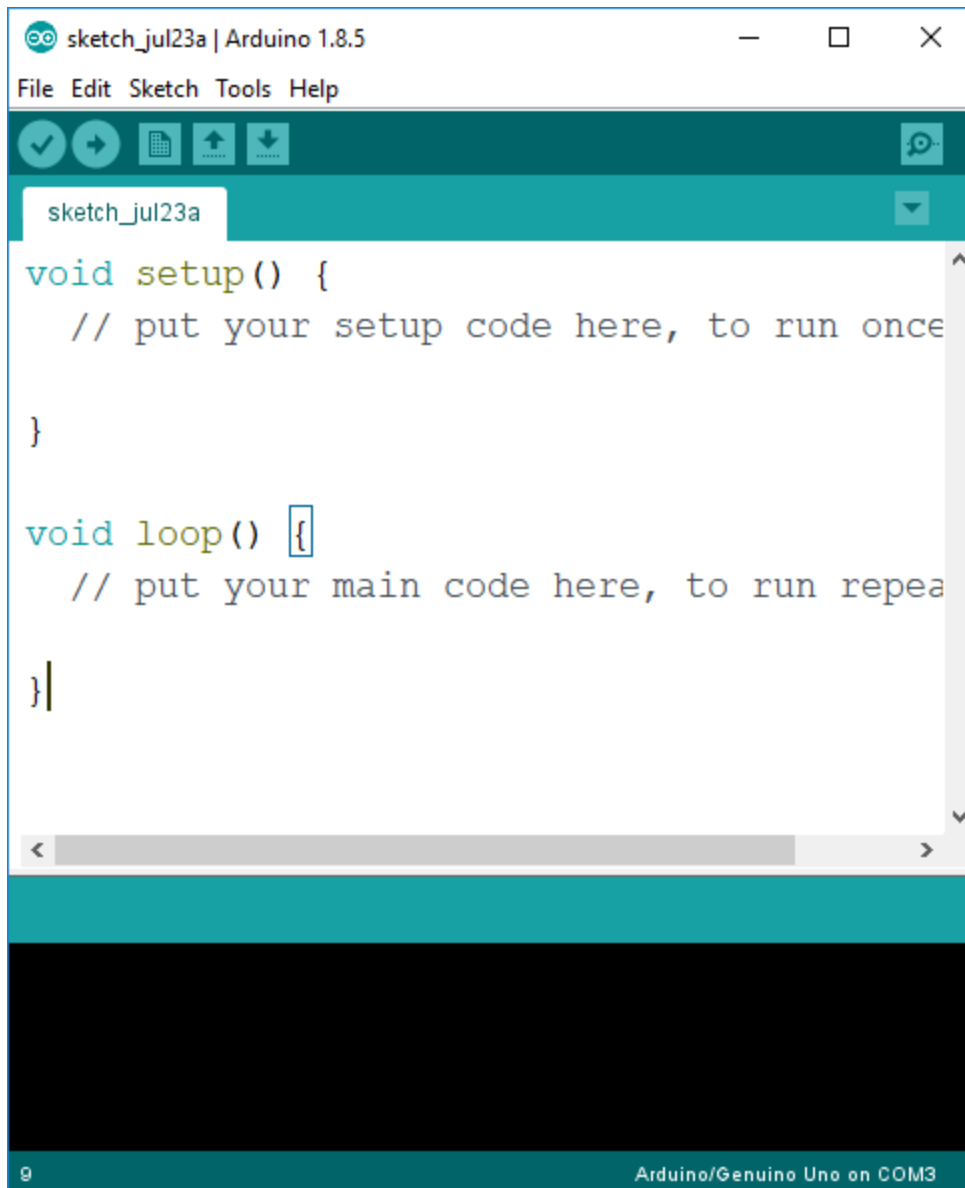
Vi điều khiển	ATmega328 họ 8bit
Điện áp hoạt động	5V DC (chỉ được cấp qua cổng USB)
Tần số hoạt động	16 MHz
Dòng tiêu thụ	khoảng 30mA
Điện áp vào khuyến dùng	7-12V DC

Điện áp vào giới hạn	6-20V DC
Số chân Digital I/O	14 (6 chân hardware PWM)
Số chân Analog	6 (độ phân giải 10bit)
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	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)

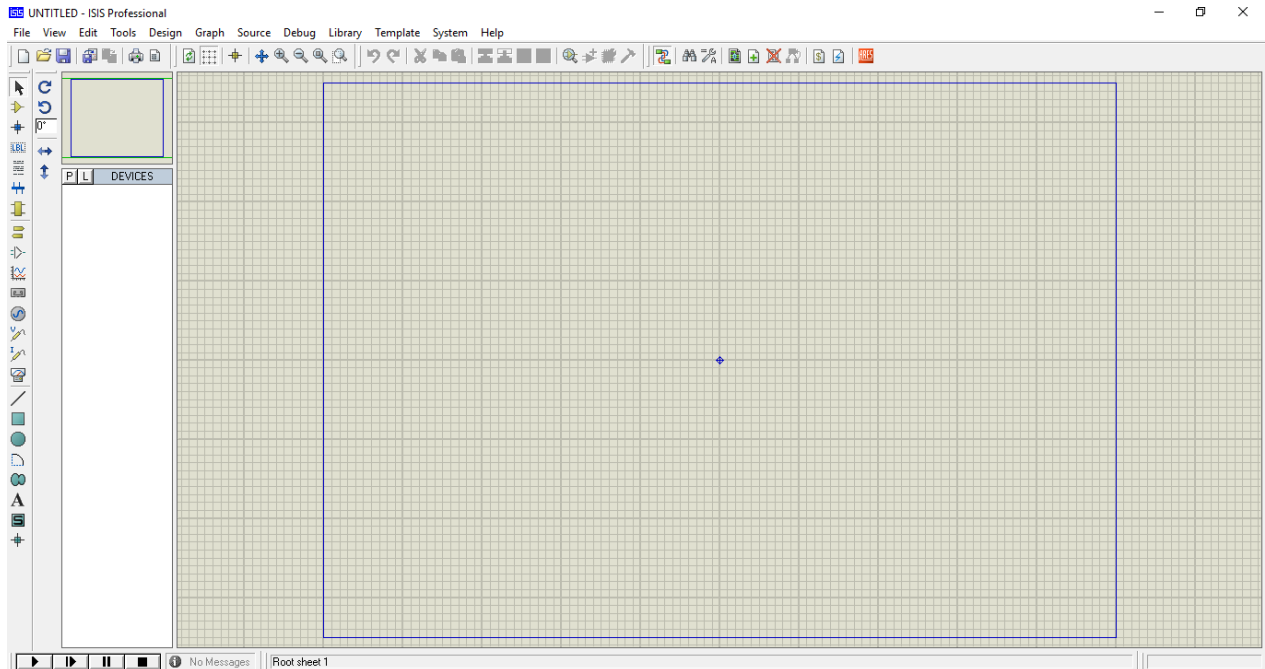


Sơ đồ chân chức năng

Phân mềm để lập trình



Arduino IDE



Phần mềm proteus

Một số keyword tìm kiếm trong proteus:

Arduino uno: arduino

Đèn led: led

Nút nhấn : button

Điện trở: res

Biến trở: pot-hg

Quang trở: ldr

Tụ phân cực: radial

Tụ không phân cực: cap

Đi-ốt: diode

Tranzito: NPN,PNP

Cầu đi-ốt: bridge

Thạch anh: crystal

Nút nhấn: button

Nguồn AC: $v \sin$

Nguồn DC: battery

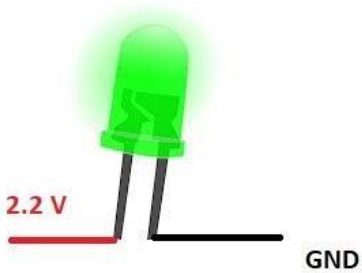
Rơ le : relay

Led 7 đoạn : 7SEG

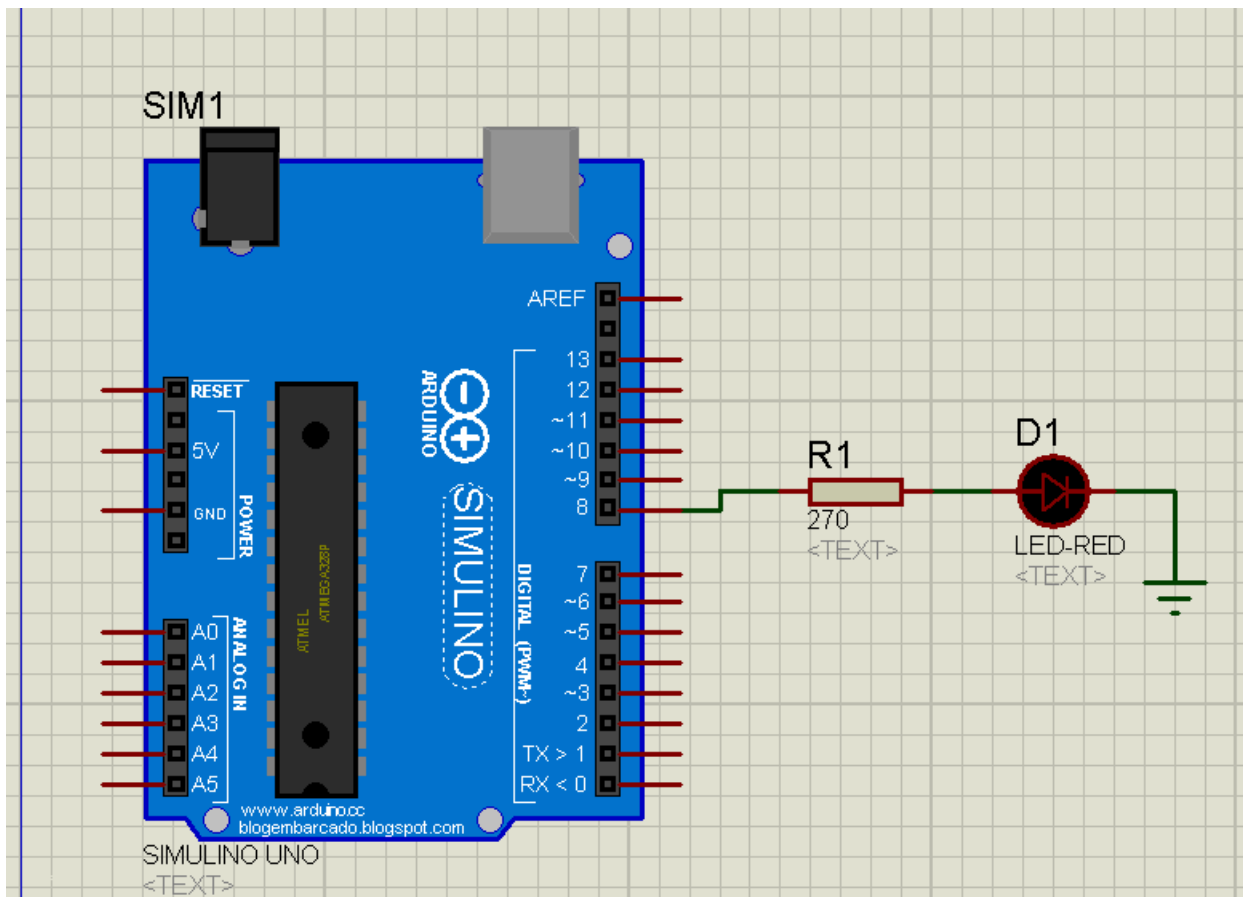
Bóng đèn: lamp.

BÀI 1 ĐIỀU KHIỂN LED ĐƠN

Làm thế nào để đèn led có thể sáng



Ví dụ: nhấp nháy led đơn



```

1 void setup() {
2
3   pinMode(8, OUTPUT); //cấu hình chân số 8 là ngõ ra
4 }
5
6
7 void loop() {
8   digitalWrite(8, HIGH); //xuất mức 1 (5v) trên chân số 8
9   delay(1000);           //dừng chương trình 1s
10  digitalWrite(8, LOW);  //xuất mức 0 (0v) trên chân số 8
11  delay(1000);          //dừng chương trình 1s
12 }

```

Bài tập 1: Điều khiển led ở chân số 3 sáng trong 0.3 s tắt trong 0.7 s

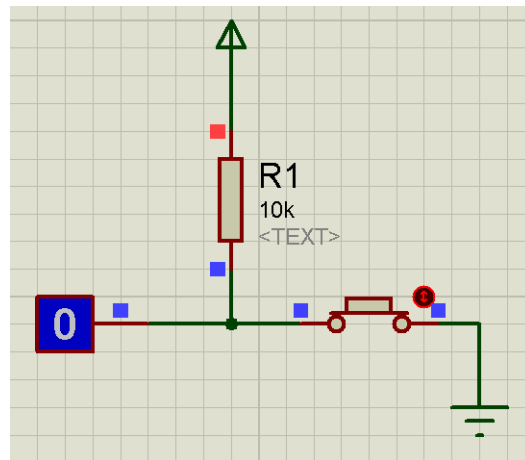
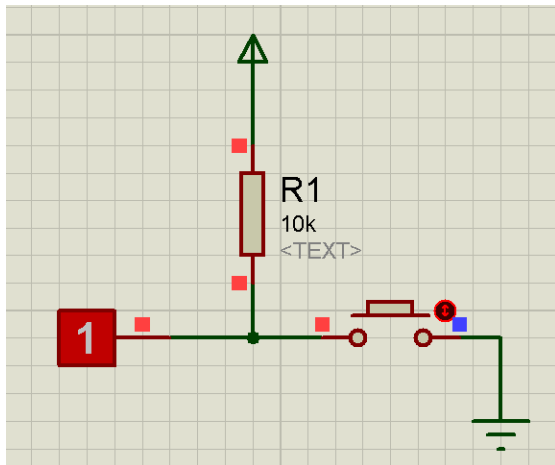
Bài tập 2: Làm nhấp nháy 8 led

Bài tập 3: Viết hiệu ứng sáng xen kẽ 8 led

Bài tập 4: Viết hiệu ứng sáng lần lượt sau tắt lần lượt của 8 led

BÀI 2 GIAO TIẾP NÚT NHẤN

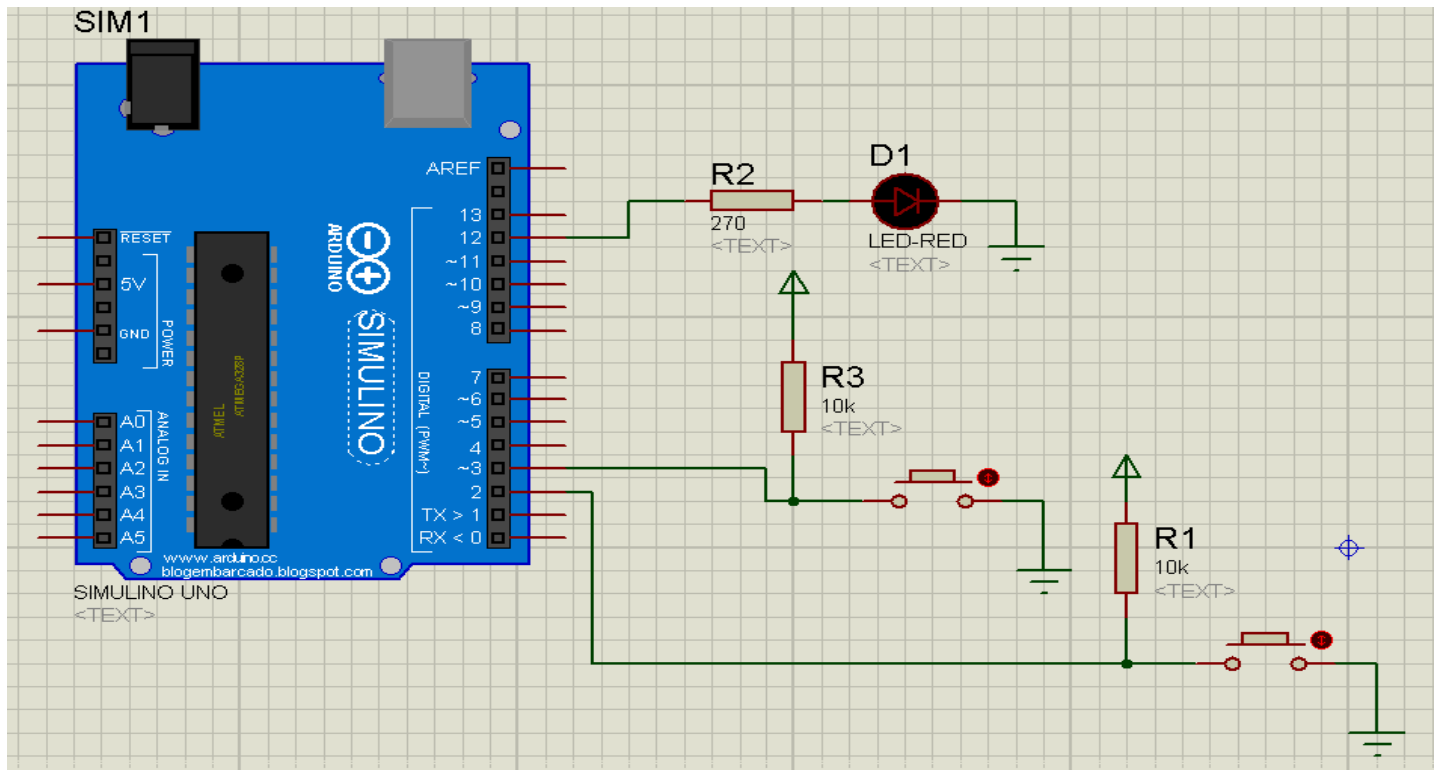
Làm thế nào để có thể biết mình đang nhấn nút?



Khi chưa nhấn nút thì pin sẽ được nối với nguồn trạng thái khi ấy sẽ là mức **HIGH**.
Khi nhấn nút pin sẽ được kéo xuống đất khi ấy trạng thái sẽ ở mức **LOW**.

Ví dụ: Dùng 2 nút nhấn điều khiển 1 bóng đèn. 1 nút có chức năng bật. Nút còn lại có chức năng tắt.

Giải:



```

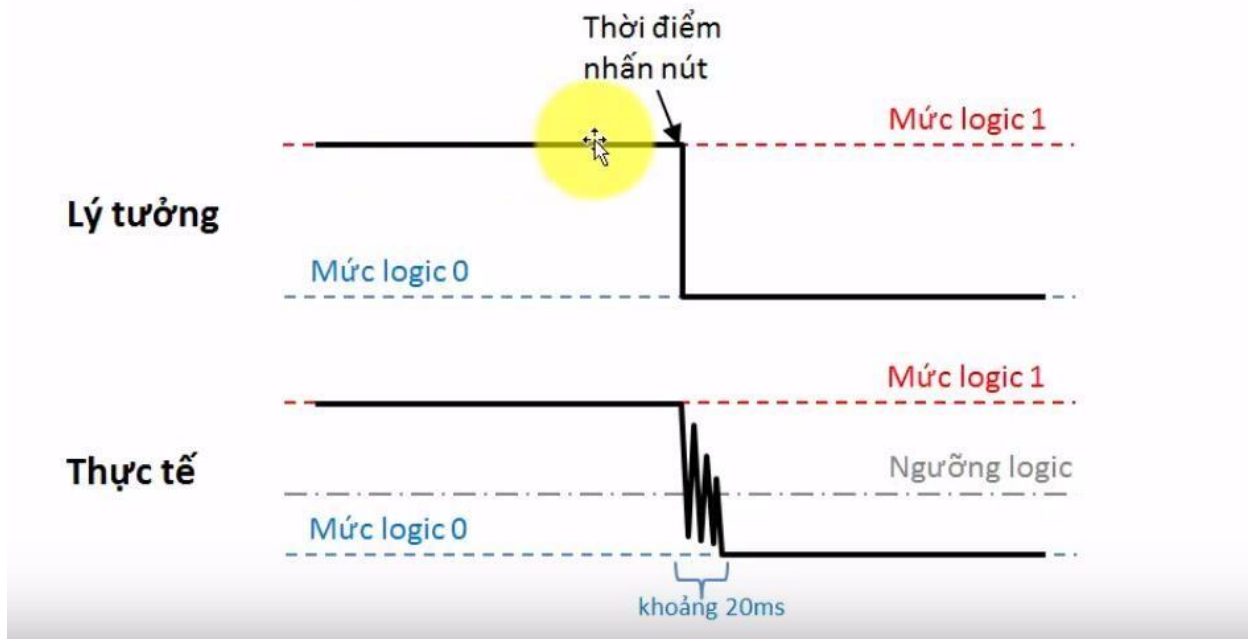
1 void setup() {
2
3   pinMode(2, INPUT); //cấu hình chân số 2 là ngõ vào
4   pinMode(3, INPUT); //cấu hình chân số 3 là ngõ vào
5   pinMode(12,OUTPUT); //cấu hình chân số 12 là ngõ ra
6 }
7
8 void loop() {
9   if(digitalRead(2)==LOW)//đọc trạng thái chân số 2 nếu như có nhấn nút thì xuất chân 12 lên mức 1
10  {
11    digitalWrite(12,HIGH); //xuất chân 12 lên mức 1
12  }
13  if(digitalRead(3)==LOW)//đọc trạng thái chân số 3 nếu như có nhấn nút thì xuất chân 12 xuống mức 0
14  {
15    digitalWrite(12,LOW); //xuất chân 12 xuống mức 0
16  }
17 }

```

Bài tập 1 : Dùng 1 nút nhấn điều khiển 1 đèn. Nhấn lần thứ nhất bóng đèn sáng, lần thứ 2 bóng đèn tắt. cứ thế lặp lại.

Lưu ý: hiện tượng đội phím.

Hiện tượng dội phím



Bài tập 2: Dùng 3 nút nhấn điều khiển 8 đèn.

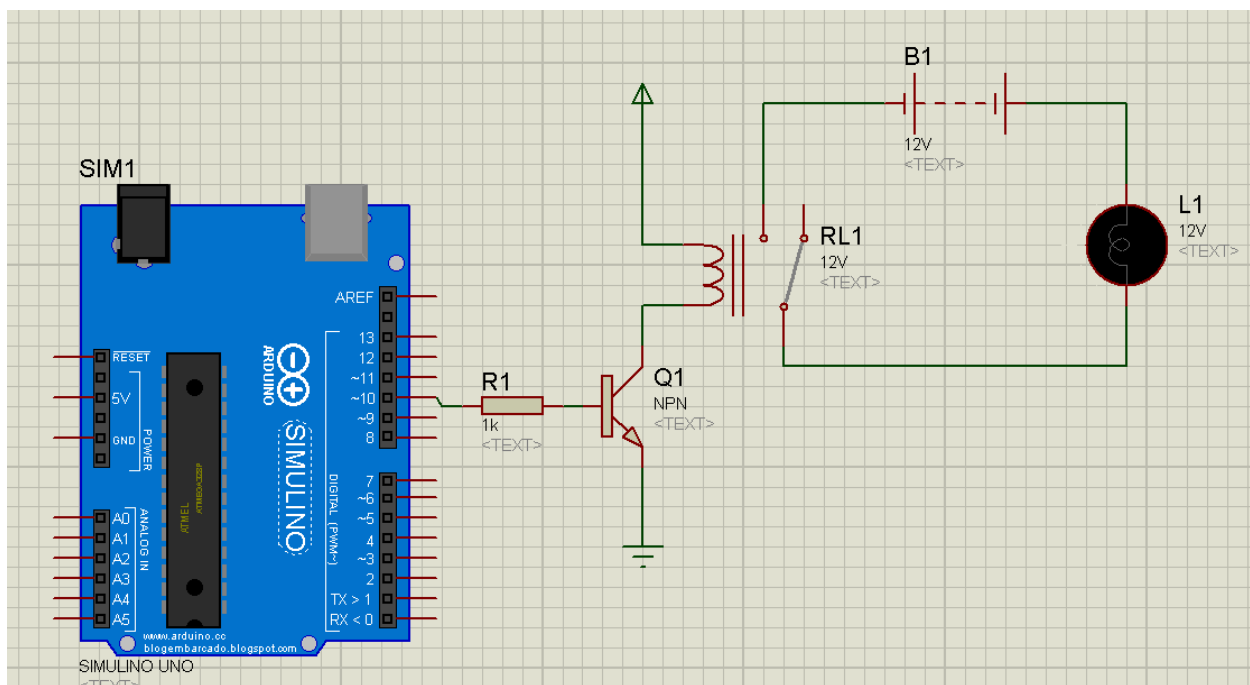
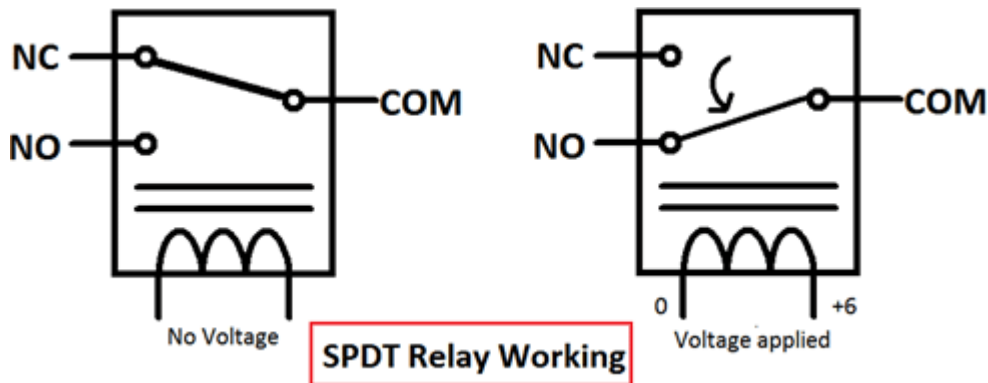
Chức năng của từng nút:

Nút thứ 1: bật hết

Nút thứ 2: tắt hết

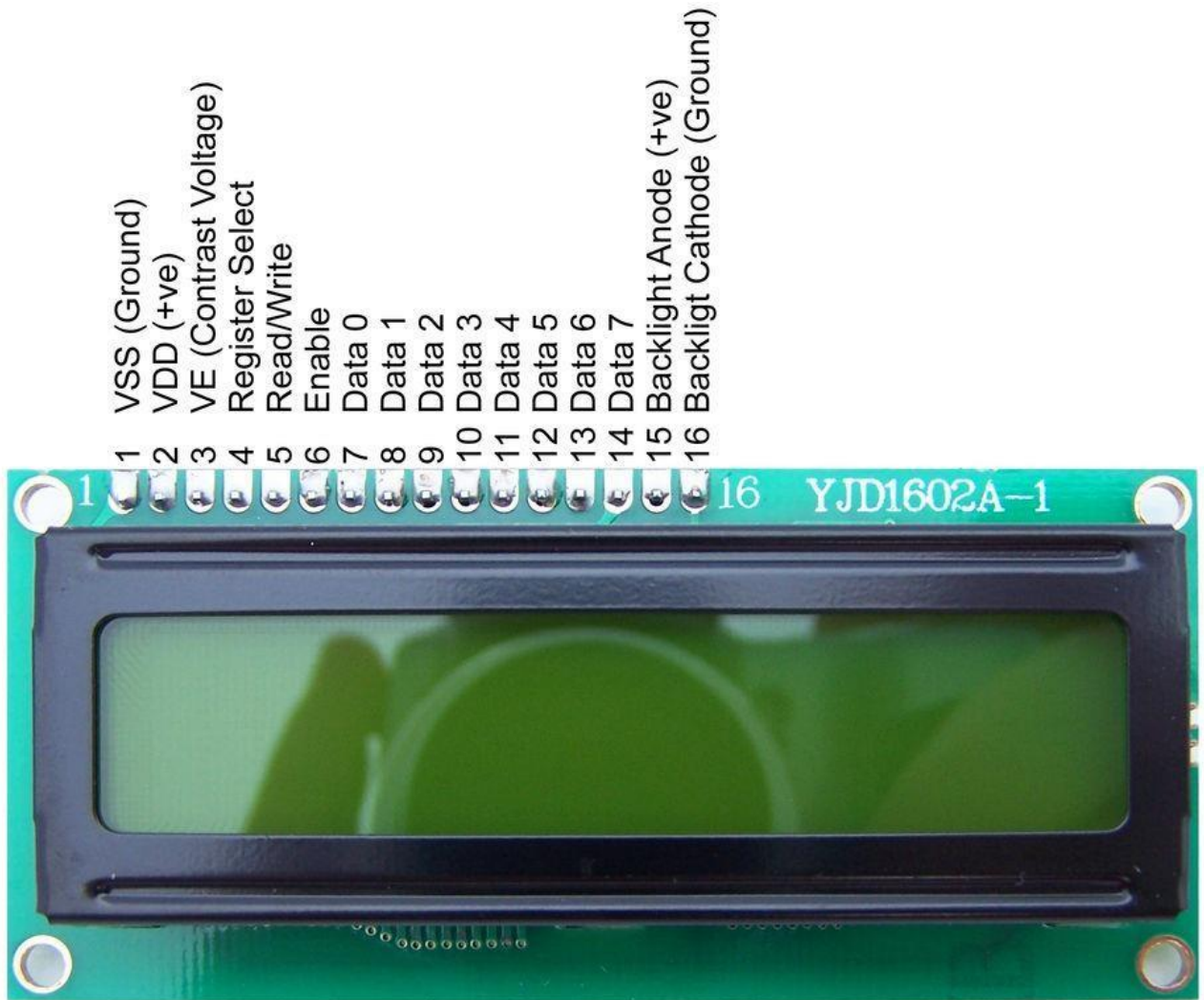
Nút thứ 3: bật từng cái (ấn 1 cái bật 1 bóng, ấn 2 bật 2...ấn 8 bật 8, ấn 9 tắt, tiếp tục quay về).

BÀI 3 GIAO TIẾP VỚI RƠ LE,SPEAKER



BÀI 4 GIAO TIẾP VỚI LCD 1602

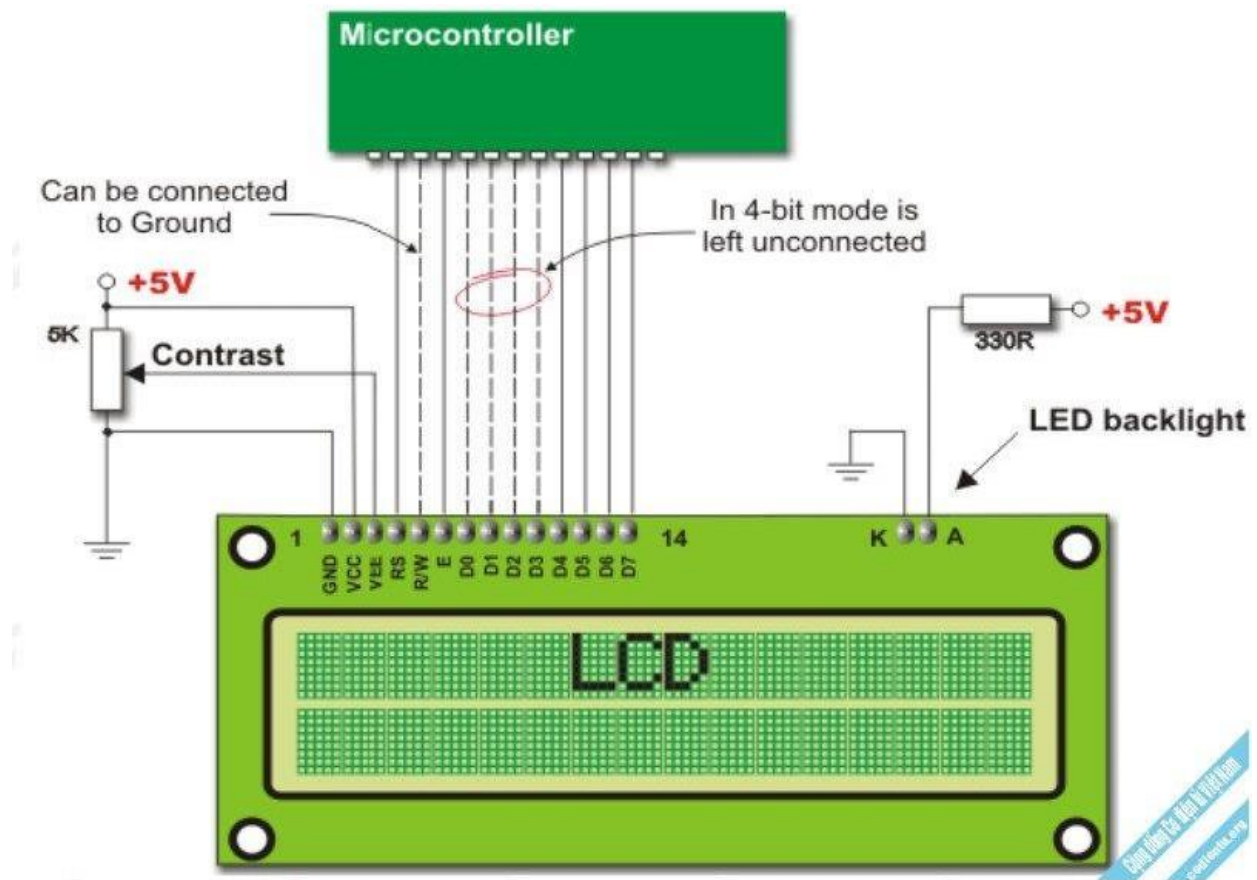
I Cấu tạo LCD

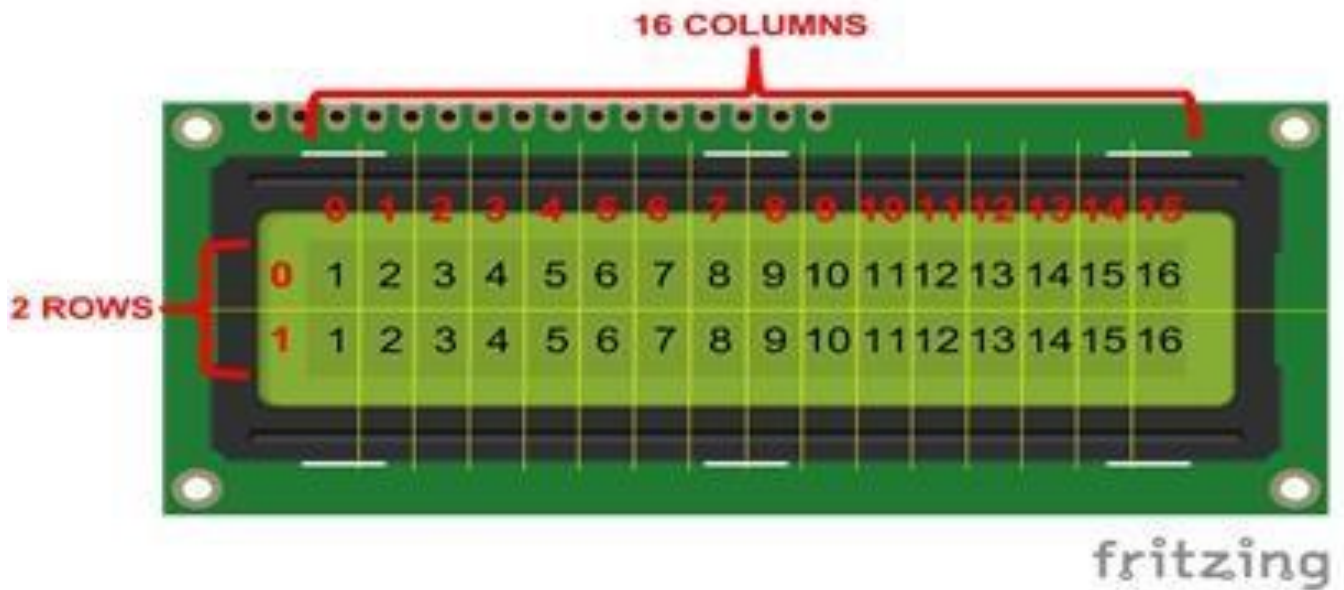


- VSS: tương đương với GND - cực âm
- VDD: tương đương với VCC - cực dương (5V)
- Contrast Voltage (V_o): điều khiển độ sáng màn hình
- Register Select (RS): điều khiển địa chỉ nào sẽ được ghi dữ liệu
- Read/Write (RW): Bạn sẽ đọc (read mode) hay ghi (write mode) dữ liệu? Nó sẽ phụ thuộc vào bạn gửi giá trị gì vào.
- Enable pin: Cho phép ghi vào LCD

- D0 - D7: 8 chân dữ liệu, mỗi chân sẽ có giá trị HIGH hoặc LOW nếu bạn đang ở chế độ đọc (read mode) và nó sẽ nhận giá trị HIGH hoặc LOW nếu đang ở chế độ ghi (write mode)
- Backlight (Backlight Anode (+) và Backlight Cathode (-)): Tắt bật đèn màn hình LCD.

đồ kết nối tới bộ xử lí





II, Giao tiếp với arduino

-sử dụng thư viện LiquidCrystal bằng cách khai báo thư viện :

```
#include <LiquidCrystal.h>
```

```
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);//khởi tạo lcd
```

Một số phương thức hay sử dụng:

```
lcd.begin(16, 2);//lcd 16 cột 2 dòng
```

```
// Print a message to the LCD.
```

```
lcd.setCursor(columns ,row);
```

```
lcd.print(string);//in một chuỗi ra màn hình LCD tại vị trí con trỏ hiện tại
```

```
lcd.blink() - lcd.noBlink()   bật tắt con trỏ ô vuông
```

```
lcd.cursor() – lcd.noCursor() bật tắt con trỏ dạng gạch dưới
```

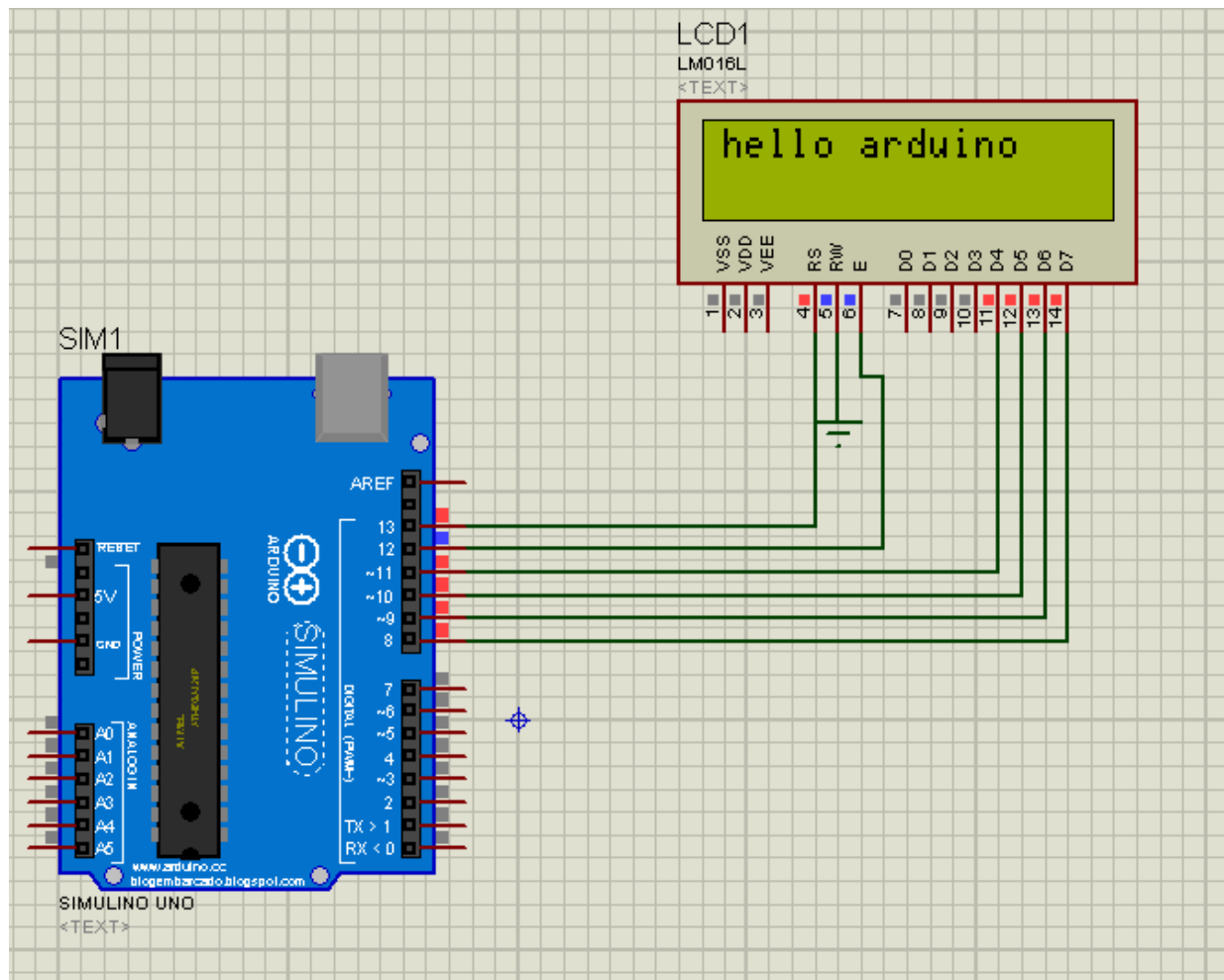
```
lcd.scrollDisplayRight(); cuộn từ phải sang trái
```

```
lcd.scrollDisplayLeft(); (); cuộn từ trái sang phải
```

```
lcd.home(); di chuyển về đầu
```


Ví dụ: hiện thị dòng chữ “hello Arduino” trên vị trí bất kì.

Giải:



```

3 #include <LiquidCrystal.h>
4
5 const int rs = 13, en = 12, d4 = 11, d5 = 10, d6 = 9, d7 = 8;
6 LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
7
8 void setup() {
9   // Cấu hình LCD:
10  lcd.begin(16, 2);
11  lcd.setCursor(0,0);
12  lcd.print("hello arduino");
13 }
14
15 void loop() {}
16
17 }

```

Bài tập 1: nhấp nháy dòng chữ “helllo Arduino” ở bài 1, nhấp nháy con trỏ ở 2 dạng.

Bài tập 2: chạy chữ “chao mung den voi lop hoc arduino” phải sang trái và ngược lại.

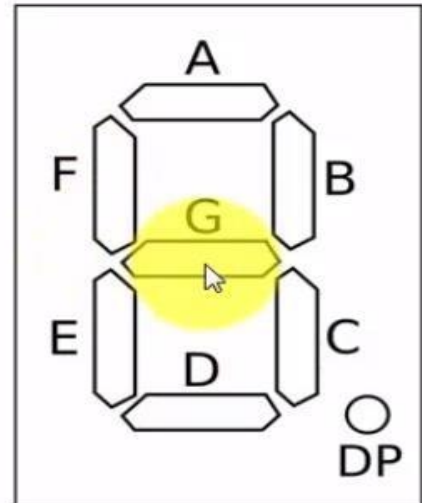
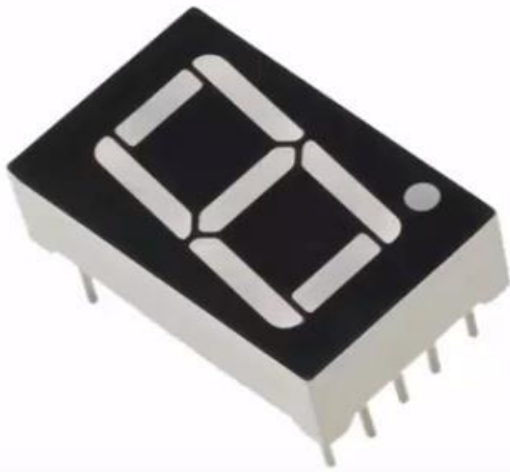
Serial công cụ debug trực tiếp

Để sử dụng ta khai báo **Serial.begin(9600);**//tốc độ baud 9600

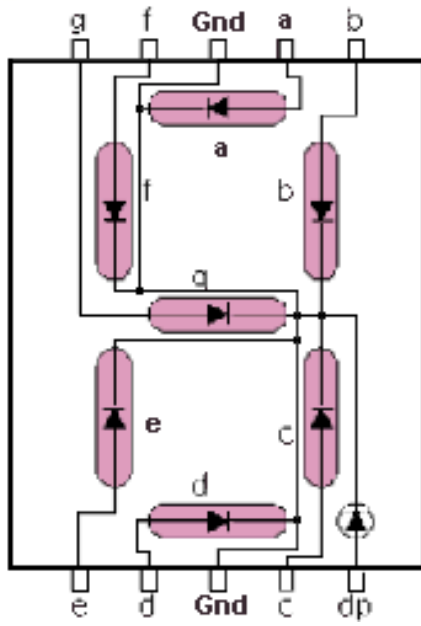
Để in ra màn hình ta dùng **Serial.print(string);**//in không xuống hàng.

Serial.println(string);//in có xuống hàng.

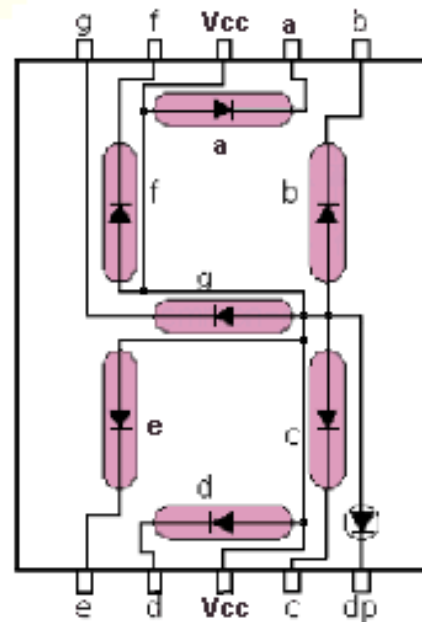
BÀI 5 GIAO TIẾP VỚI LED 7 ĐOẠN

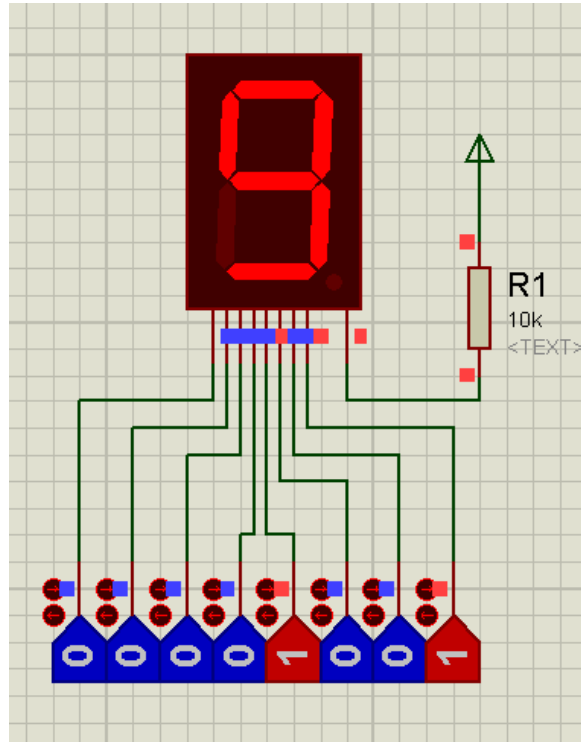


Common Cathode



Common Anode





Bài 1: hiển thị các số từ 0->9 mỗi lần cách nhau 1s.

Bài 2: dùng 2 nút nhấn điều khiển số hiển thị trên led 7 đoạn. khi ấn nút tăng thì led tăng thêm 1 đơn vị, khi giảm thì giảm 1 đơn vị.

Bài 3: đếm từ 00->99 hiển thị trên led 7 đoạn.

BÀI 6 ADC-ĐỌC TÍN HIỆU TỪ MỘT SỐ CẢM BIẾN.

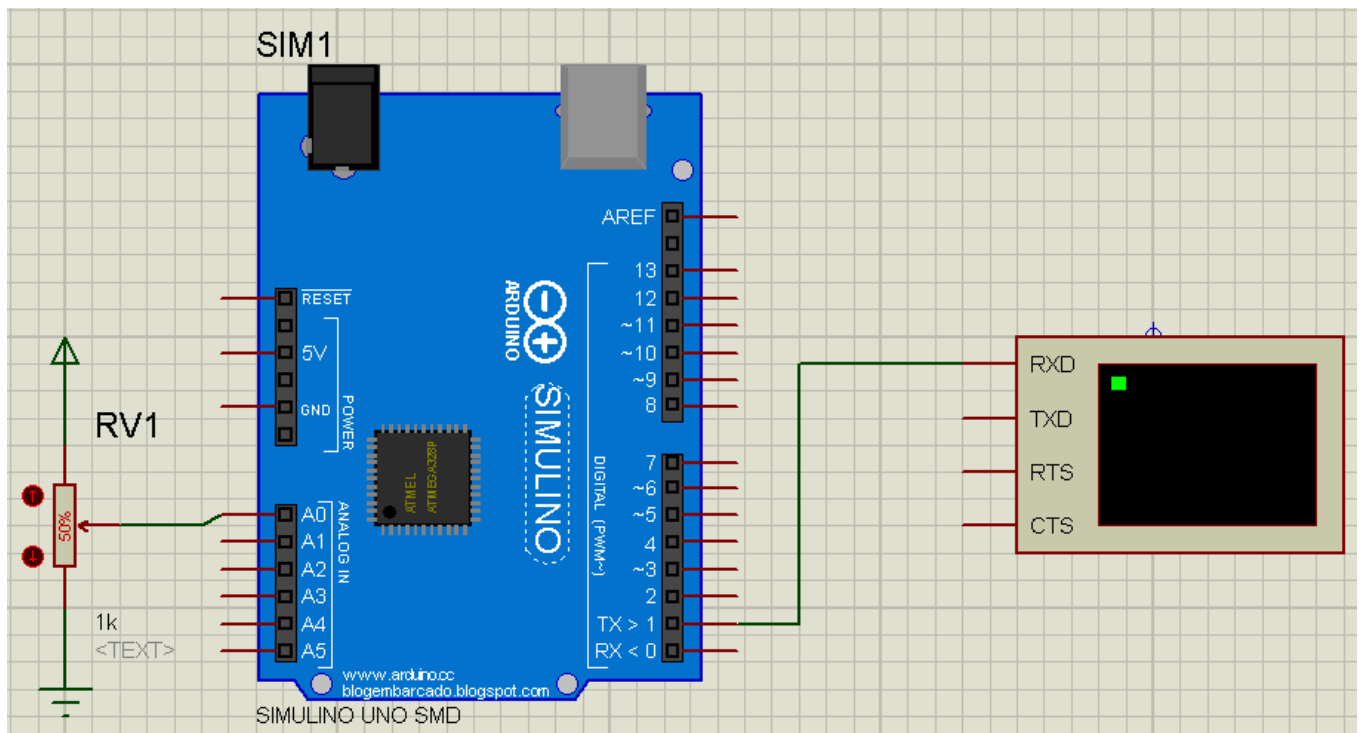
ADC là gì? ADC là bộ chuyển đổi tín hiệu tương tự sang số. độ chính xác của bộ ADC phụ thuộc vào độ phân giải.

Độ phân giải bộ ADC của Arduino là 10 bit ($2^{10}-1$).

Tức là khi điện áp đầu vào thay đổi từ 0-5v thì bộ ADC sẽ thay đổi từ 0-1023;

Để đọc giá trị ADC ta sử dụng: `analogRead(channel)` channel có giá trị từ A0-A5;

Ví dụ: đọc giá trị ADC hiển thị lên Serial



```
1 void setup() {
2
3 }
4 int adc;//biến lưu trữ giá trị adc
5 void loop() {
6   adc=analogRead(A0);//đọc giá trị adc
7   Serial.println(adc); //hiển thị lên Serial
8 }
```

Bài 1: đọc giá trị ADC hiển thị lên LCD;

Bài 2: thiết kế mạch đo điện áp.

Bài 3: thiết kế mạch đo nhiệt độ sử dụng LM35

Ghi chú: LM35 là cảm biến nhiệt độ, đầu ra là tín hiệu điện áp biến đổi theo nhiệt độ, cứ 10mV tương ứng với 1 độ C và dải đo của IC này là -55 độ đến 150 độ C, điện áp cung cấp từ 4-20VDC.

BÀI 7 NGẮT NGOÀI

Ngắt (interrupt) là những lời gọi hàm tự động khi hệ thống sinh ra một sự kiện. Những sự kiện này được nhà sản xuất vi điều khiển thiết lập bằng phần cứng và được cấu hình trong phần mềm bằng những tên gọi cố định.

Vì sao cần phải dùng đến ngắt?

Ngắt giúp chương trình gọn nhẹ và xử lý nhanh hơn. Chẳng hạn, khi kiểm tra 1 nút nhấn có được nhấn hay không, thông thường bạn cần kiểm tra trạng thái nút nhấn bằng hàm `digitalRead()` trong đoạn chương trình `loop` {}. Với việc sử dụng ngắt, bạn chỉ cần nối nút nhấn đến đúng chân có hỗ trợ ngắt, sau đó cài đặt ngắt sẽ sinh ra khi trạng thái nút chuyển từ HIGH->LOW. Thêm 1 tên hàm sẽ gọi khi ngắt sinh ra.

Cú pháp: `attachInterrupt(digitalPinToInterrupt(pin), ISR, mode);`

Trong đó: pin: chân ngắt đối với uno thì 2 hoặc 3

ISR: trình phục vụ ngắt-khi có ngắt sẽ thực thi trong này

Mode: `HIGH,LOW,RISING,FALLING`;

Ví dụ: đếm số lần ấn nút sử dụng ngắt ngoài.

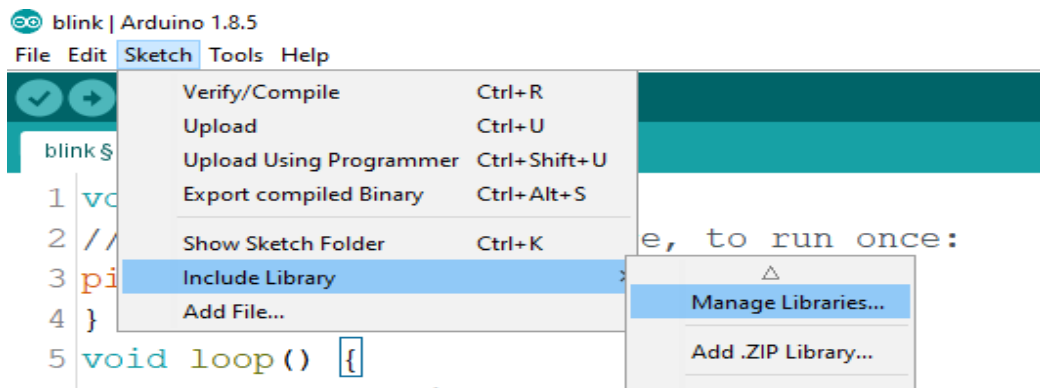
Bài tập 1: vừa nhấp nháy đèn chu kì 1s vừa điều khiển bật tắt 1 đèn khác tắt bằng nút bấm.

BÀI 8 ĐỌC CẢM BIẾN NHIỆT ĐỘ, ĐỘ ẨM CẢM BIẾN KHOẢNG CÁCH

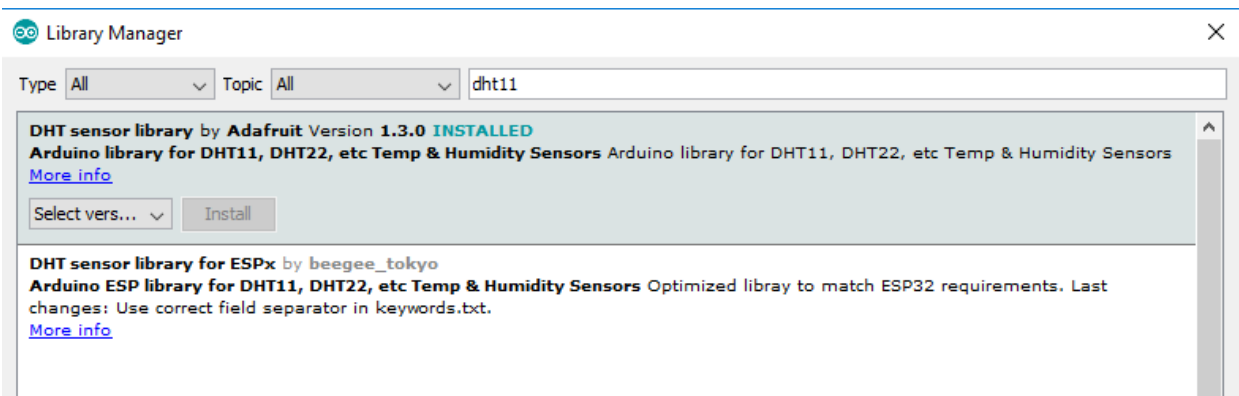
Cảm biến nhiệt độ, độ ẩm DHT11



Cách thêm thư viện từ arduino IDE



Kích vào Install để cài đặt thư viện:



```
#include "DHT.h"
```

```
const int DHTPIN = 2; //Đọc dữ liệu từ DHT11 ở chân 2 trên mạch Arduino
```



```
const int DHTTYPE = DHT11; //Khai báo loại cảm biến, có 2 loại là DHT11 và DHT22
```

```
DHT dht(DHTPIN, DHTTYPE);
```

```
void setup() {
```

```
    dht.begin(); // Khởi động cảm biến
```

```
}
```

```
void loop() {
```

```
    float h = dht.readHumidity(); //Đọc độ ẩm
```

```
    float t = dht.readTemperature(); //Đọc nhiệt độ
```

```
}
```

II, Cảm biến khoảng cách

Cảm biến khoảng cách siêu âm HC-SR04 được sử dụng rất phổ biến để xác định khoảng cách vì RẺ và CHÍNH XÁC. Cảm biến sử dụng sóng siêu âm và có thể đo khoảng cách trong khoảng từ 2 -> 300 cm, với độ chính xác gần như chỉ phụ thuộc vào cách lập trình.



- **Vcc:** cấp nguồn cho cảm biến.

- **Trigger:** kích hoạt quá trình phát sóng âm. Quá trình kích hoạt khi một chu kì điện cao / thấp diễn ra.
- **Echo:** bình thường sẽ ở trạng thái 0V, được kích hoạt lên 5V ngay khi có tín hiệu trả về, sau đó trở về 0V.
- **Gnd:** nối với cực âm của mạch
- **OUT:** không sử dụng

Thuật toán đo khoảng cách:

1. Kích hoạt cảm biến bằng việc bật PIN Trigger theo thứ tự LOW(delay 2us) – HIGH(delay 10us) - LOW qua hàm `digitalWrite`.
2. Sử dụng hàm `pulseIn(HIGH)` để trả về thời gian từ khi phát tín hiệu đến khi truyền tín hiệu.
3. Tính toán khoảng cách: vận tốc âm thanh $340(\text{m/s}) = 340 \cdot 100 / (10^6) = 0.034$ (cm/us);
4. \Rightarrow khoảng cách = vận tốc x thời gian = $0.034 \cdot \text{time} / 2 = 300$ (cm)

Viết hàm tính toán khoảng cách:

```
float GetDistance()
{
float duration, distanceCm;

digitalWrite(TRIG_PIN, LOW);

delayMicroseconds(2);

digitalWrite(TRIG_PIN, HIGH);

delayMicroseconds(10);

digitalWrite(TRIG_PIN, LOW);

duration = pulseIn(ECHO_PIN, HIGH, 17000);

// convert to distance
```

```
distanceCm = 0.034*duration / 2=300;
```

```
return distanceCm;
```

```
}
```

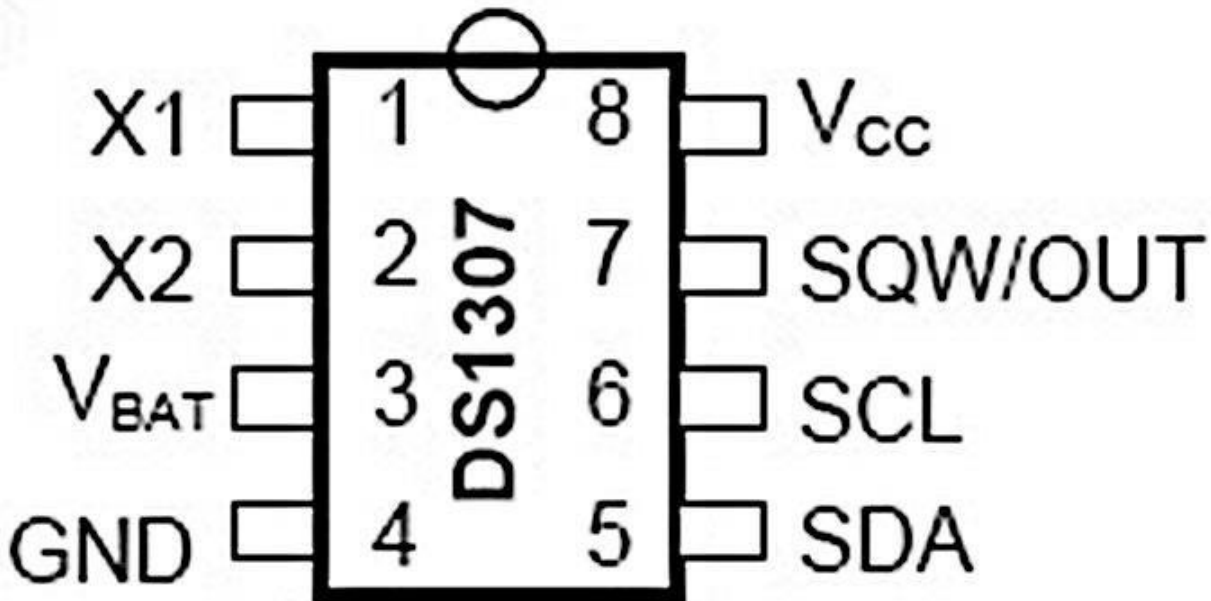
BÀI 9 GIAO TIẾP I2C- THIẾT KẾ ĐỒNG HỒ VỚI DS1307,DS1302,DS3231

I, Giới thiệu I2C

I2C là viết tắt của "Inter-Integrated Circuit", một chuẩn giao tiếp được phát minh bởi Philips' semiconductor division (giờ là NXP) nhằm đơn giản hóa việc trao đổi dữ liệu giữa các IC. Đôi khi nó cũng được gọi là Two Wire Interface (TWI) vì chỉ sử dụng 2 dây để truyền tải dữ liệu, 2 dây của giao tiếp I2C gồm: SDA (Serial Data Line) và SCL (Serial Clock Line).

Trên board Arduino UNO, SDA là chân A4, SCL là chân A5.

II Giao tiếp với DS1307



X1-X2 nối với thạch anh 32.768 khz

Vbat-nối với pin 3v

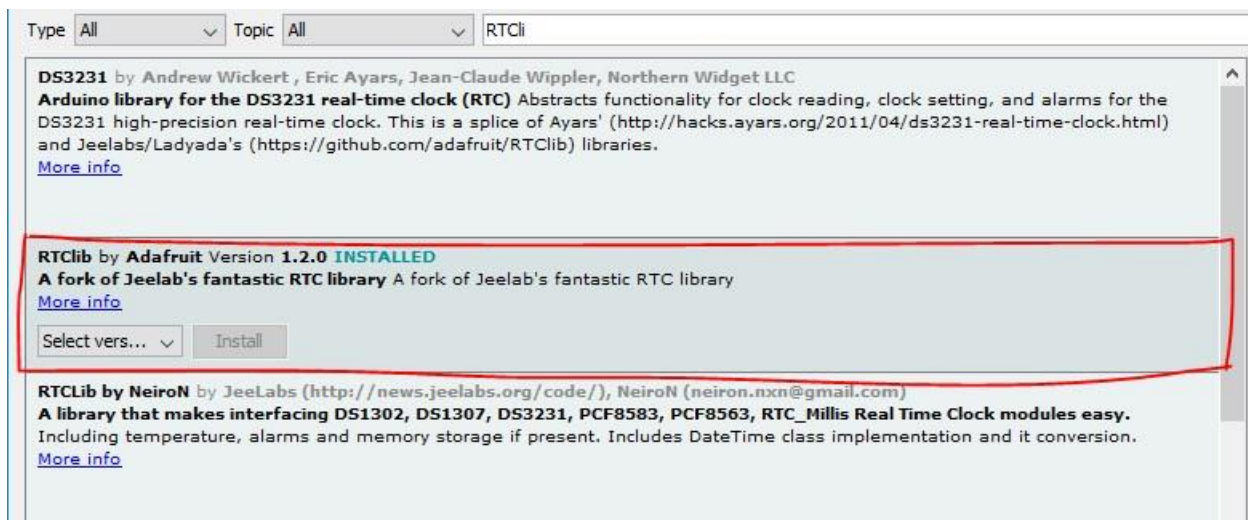
Vcc nối nguồn 5v

SQW-chân ngõ ra 1HZ

SCL-SDA 2 chân giao tiếp chuẩn I2C

Để giao tiếp được với DS1307 ta dùng thư viện RTCLib

Để thêm thư viện ta vào Sketch > Include Library > Manage Library tìm có từ khóa RTCLib kích vào install để cài đặt.



Để sử dụng: #include "RTCLib.h"

Một số phương thức chính:

RTC_DS1307 rtc;//khởi tạo đối tượng RTC

Datetime t=rtc.now() trả về kiểu dữ liệu DateTime;

rtc.adjust(**Datetime**(nam,thang,ngay,h,m,s));//chỉnh thời gian

t.**year**();//trả về năm hiện tại

t.**month**();//trả về tháng hiện tại

t.**day**();//trả về ngày hiện tại

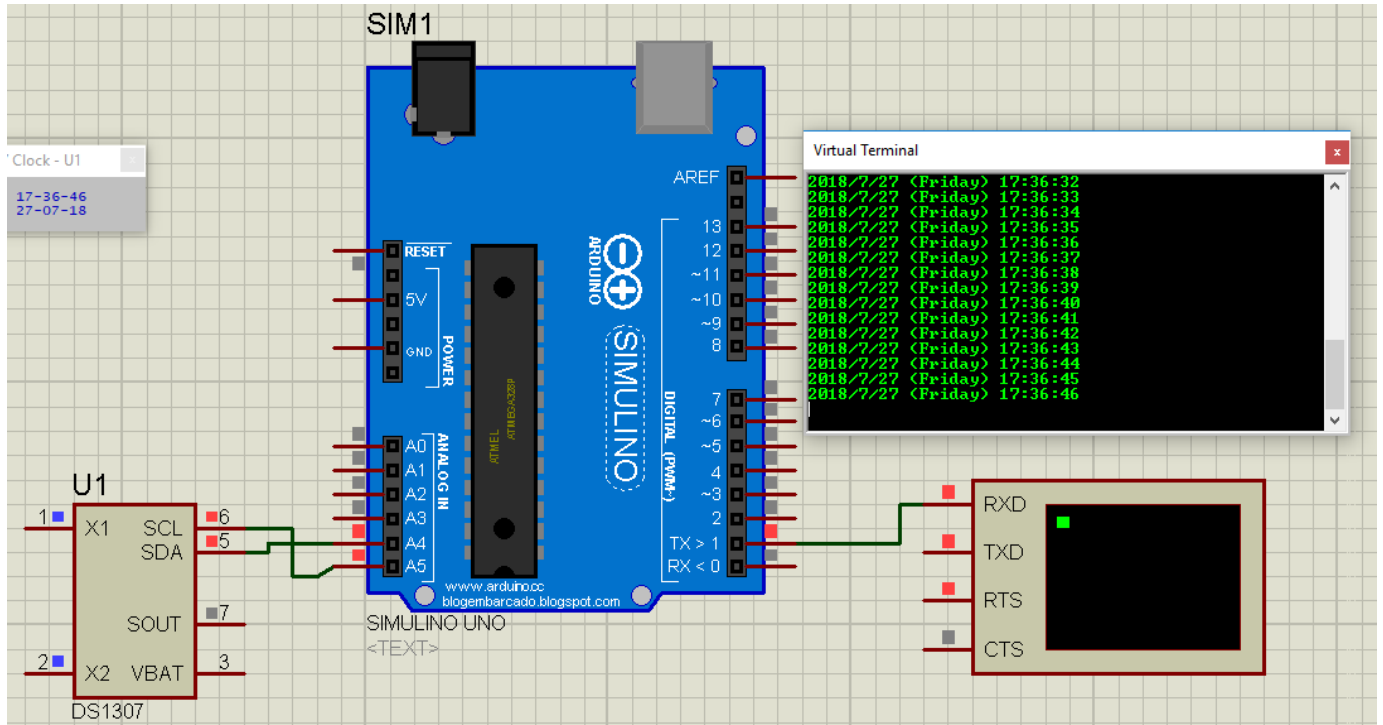
t.**dayOfTheWeek**()//trả về thứ kết quả trả về từ 0-7 tương ứng từ chủ nhật đến thứ 7

now.**hour**();//trả về giờ hiện tại

now.minute();//trả về phút hiện tại

now.second();//trả về giây hiện tại

ví dụ: lấy thời gian hiện tại và hiện thi qua Serial

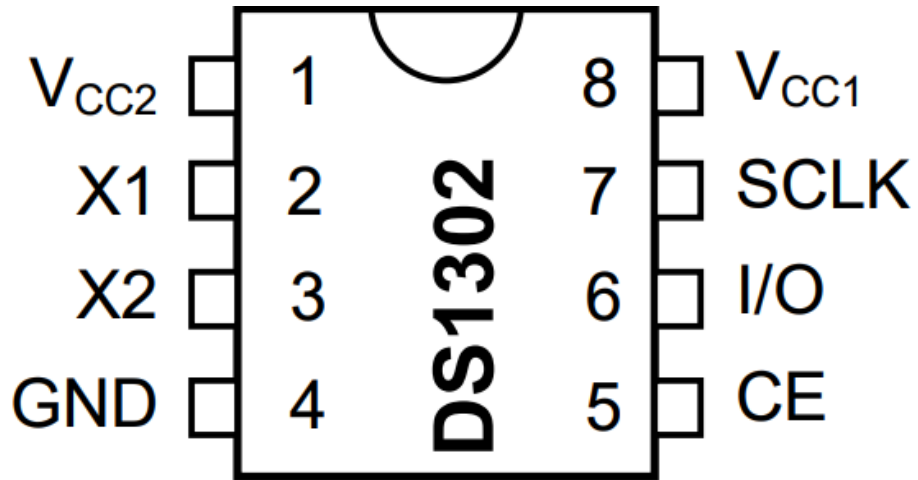


```

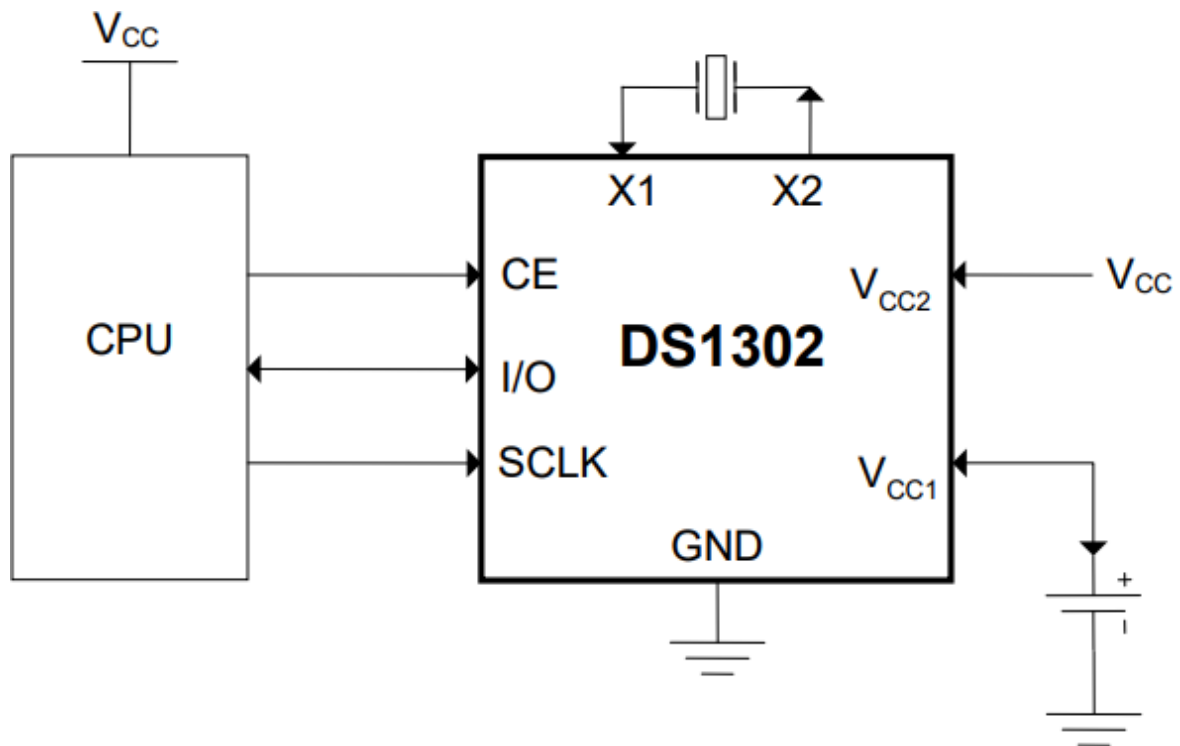
1 #include <Wire.h>//khai bao thu vien I2C
2 #include "RTClib.h"//Khai báo thu viện DS1307
3
4 RTC_DS1307 rtc;//tạo đối tượng DS1307
5
6 char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};
7
8 void setup () {
9
10     Serial.begin(9600);//tốc độ baud 9600
11 }
12
13 void loop () {
14     DateTime now = rtc.now();//lấy thời gian hiện tại bao gồm năm tháng ngày h,m,s
15
16     Serial.print(now.year(), DEC);//lấy năm
17     Serial.print('/');
18     Serial.print(now.month(), DEC);//lấy tháng
19
20     Serial.print(" ");
21     Serial.print(daysOfTheWeek[now.dayOfTheWeek()]); //lấy thứ
22     Serial.print(" ");
23     Serial.print(now.hour(), DEC);//lấy h
24     Serial.print(':');
25     Serial.print(now.minute(), DEC);//lấy m
26     Serial.print(':');
27     Serial.print(now.second(), DEC);//lấy s
28     Serial.println();
29     delay(1000);
30 }

```

II, DS1302



Kết nối tới bộ xử lí



Download library:

https://drive.google.com/file/d/1PZuGVVudPNUmr_yFZ13i40_OI1G1tK-t/view?usp=sharing

Sử dụng thư viện:

```
#include <DS1302.h>
```

```
DS1302 rtc(RST, DATA, CLK);
```

```
Time t = rtc.getTime();
```

```
t.dow // trả về thứ
```

```
t.date // trả về ngày
```

```
t.month // trả về tháng
```

```
t.year // trả về năm
```

```
t.hour // trả về giờ
```

```
t.min // trả về phút
```

```
t.sec // trả về giây
```

```
rtc.setDOW(FRIDAY); // cài đặt thứ
```

```
rtc.setTime(12, 0, 0); // cài đặt giờ phút giây
```

```
rtc.setDate(6, 8, 2010); // cài đặt ngày tháng năm
```

Bài tập 1: hiển thị ngày tháng năm giờ phút giây qua màn hình LCD 1602

Bài tập 2: thêm chức năng cho bài tập 1 bằng cách sử dụng nút nhấn có thể điều chỉnh được thời gian.

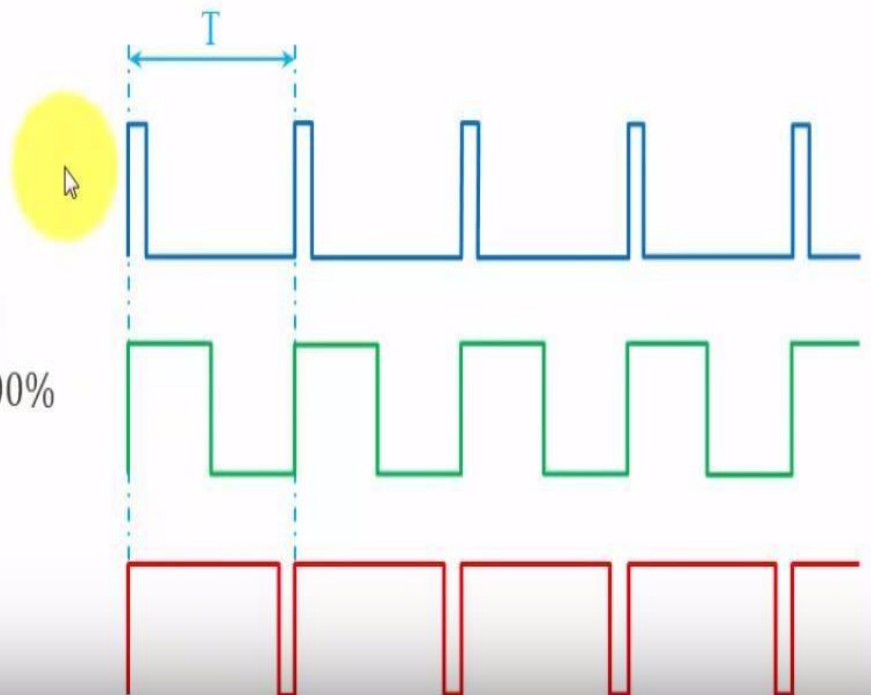
BÀI 10 PWM ĐIỀU KHIỂN ĐỘ SÁNG CỦA BÓNG ĐÈN

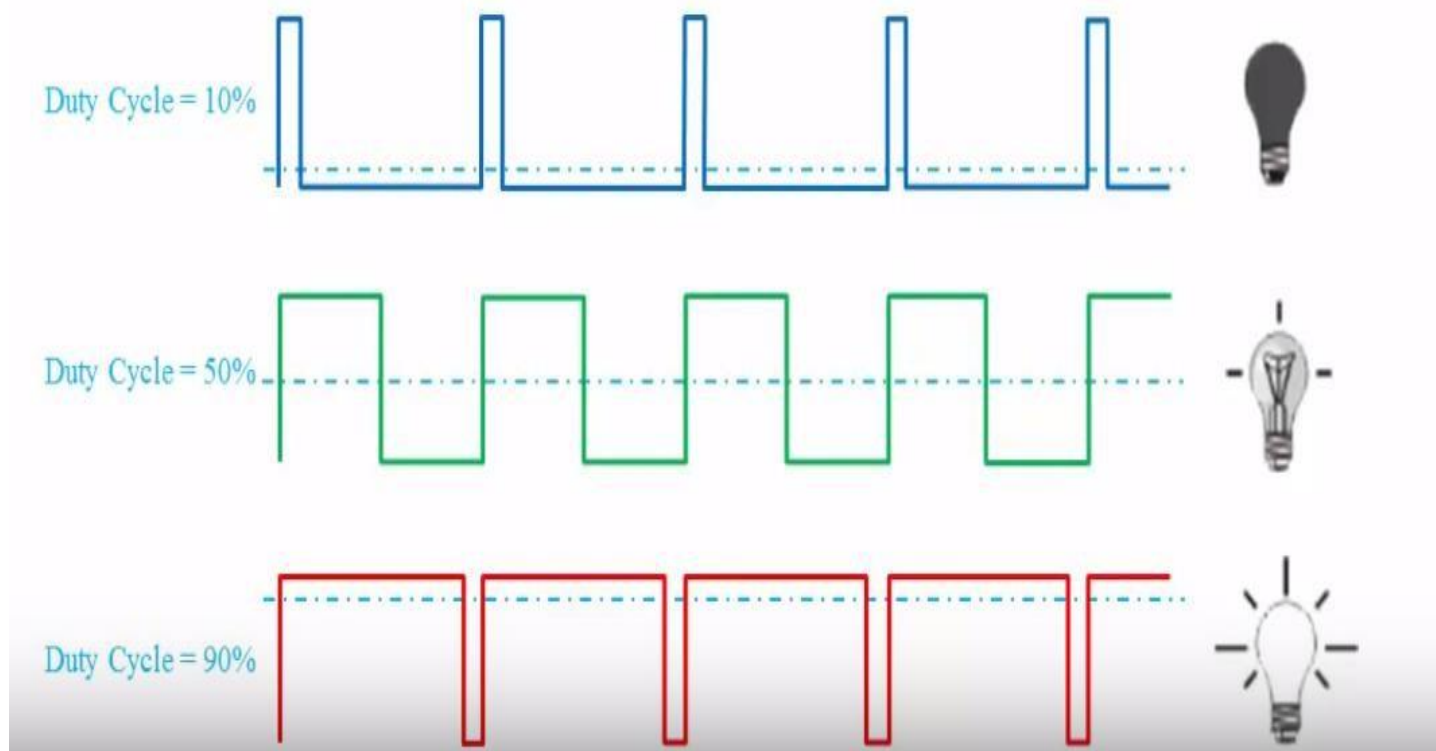
PWM (Pulse Width Modulation) là gì?

Tạo ra một dạng xung có tần số không đổi, nhưng thời gian của mức cao trong một chu kỳ có thể thay đổi được.

Các thông số:

- Tần số: f
- Chu kỳ: T
- Thời gian ở mức cao: T_{ON}
- Thời gian ở mức thấp: T_{OFF}
- $Duty\ Cycle = \frac{T_{ON}}{T_{ON}+T_{OFF}} \times 100\%$
 $= \frac{T_{ON}}{T} \times 100\%$
- Điện áp ra trung bình
 $V_{avg} = V_H \times Duty\ Cycle$





Để sử dụng PWM trong arduino ta sử dụng :

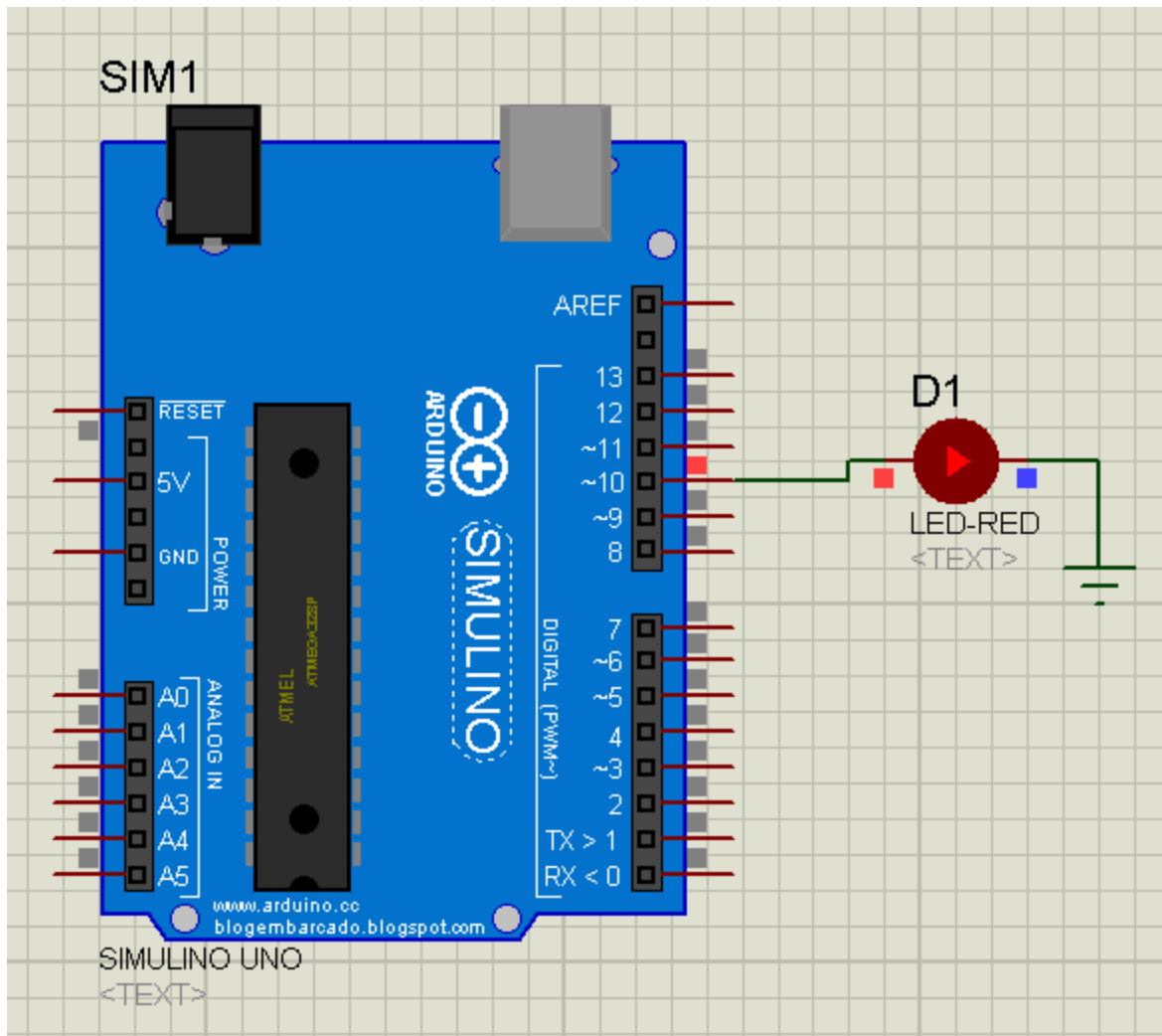
`analogWrite(pin,value);` <-> `digitalWrite(pin,HIGH)`

`//value*5/255=3`

pin: các chân có hỗ trợ PWM 3,5,6,9,10,11(các chân có kí hiệu dấu ~)

value:0-255;

ví dụ: điều khiển độ sáng bóng đèn.



```

1 int led=10;
2 void setup() {
3   // put your setup code here, to run once:
4   pinMode(led,OUTPUT);
5 }
6
7 void loop() {
8   // put your main code here, to run repeatedly:
9   analogWrite(led,100);
10 }

```

Bài tập 1: điều khiển độ sáng tăng dần sau đó giảm dần

Bài tập 2: kết hợp với biến trở điều chỉnh độ sáng của bóng đèn

BÀI 11 TIMER-NGẮT TIMER

I, Timer.

-arduino có tất cả 3 timer : timer 0,timer 1,timer 2

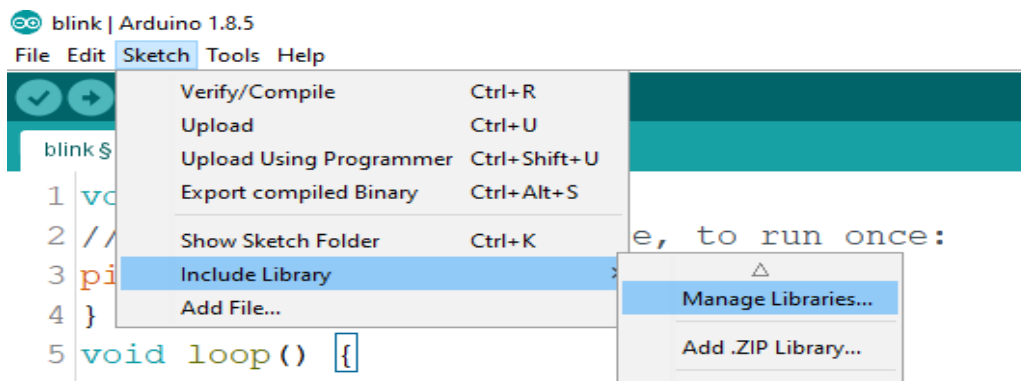
Timer 0: timer 8 bit sử dụng cho hàm
delay(),delayMicroseconds(),micros(),milis();

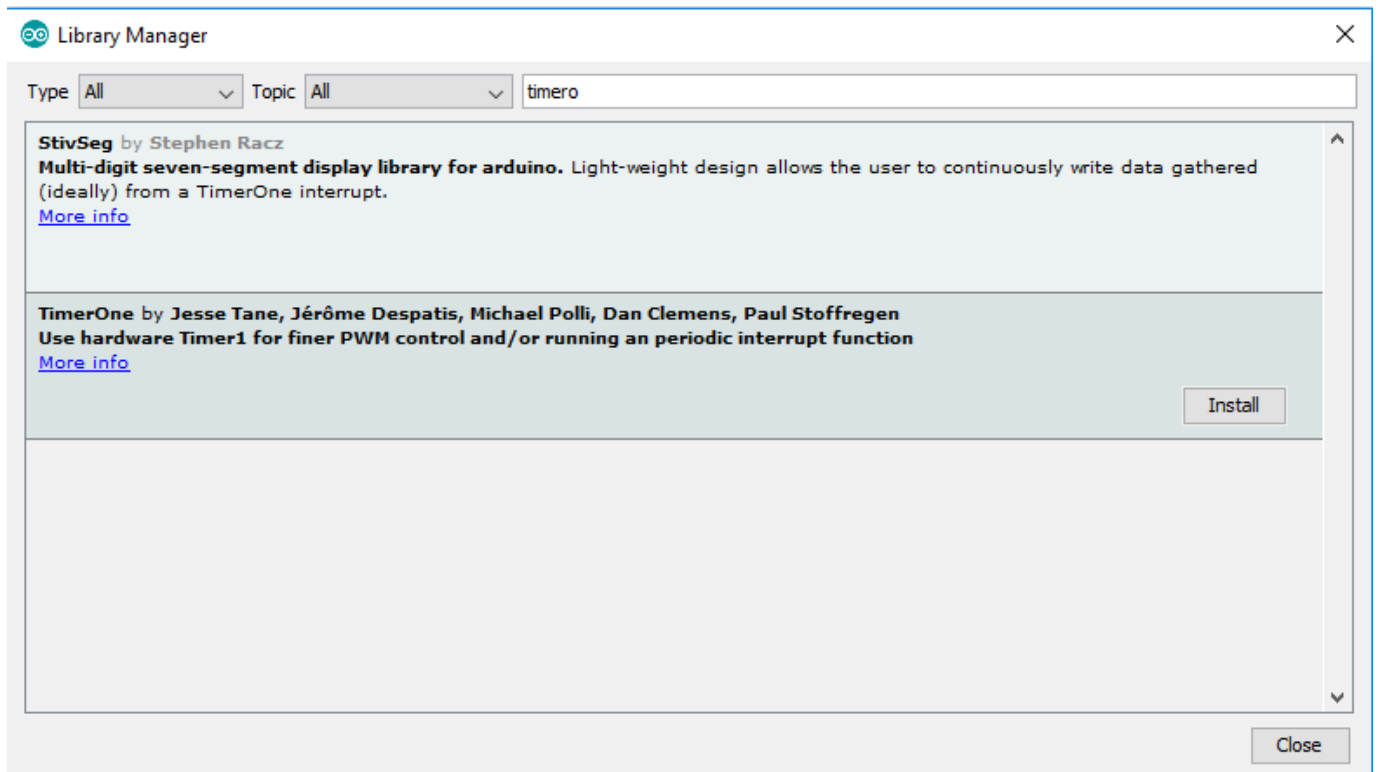
Timer 1: timer 16 bit sử dụng để điều khiển servo;

Timer 2: timer 8 bit dùng cho hàm tone();

II, Ngắt timer

Thêm thư viện timer 1 vào arduino.





Để sử dụng : `#include <TimerOne.h>`

Một số hàm lưu ý:

`Timer1.initialize(period);` //period chu kì thực hiện ngắt tính theo us

`Timer1.attachInterrupt(func);` //khi xảy ra ngắt thì hàm func sẽ thực thi.

Ví dụ: nhấp nháy led với chu kì 1s.

```

1 #include <TimerOne.h>
2 const int led = 13; // the pin with a LED
3
4 void setup(void)
5 {
6   pinMode(led, OUTPUT);
7   Timer1.initialize(1000000); //khởi tạo timer cứ sau 1000000 us=1s sẽ ngắt một lần
8   Timer1.attachInterrupt(blinkLED); //cứ 1s sẽ nhảy vào hàm blinkLED
9 }
10 int ledState = LOW;
11 void blinkLED(void)
12 {
13   if (ledState == LOW) {
14     ledState = HIGH;
15   } else {
16     ledState = LOW;
17   }
18   digitalWrite(led, ledState);
19 }
20 void loop(void)
21 {
22 }

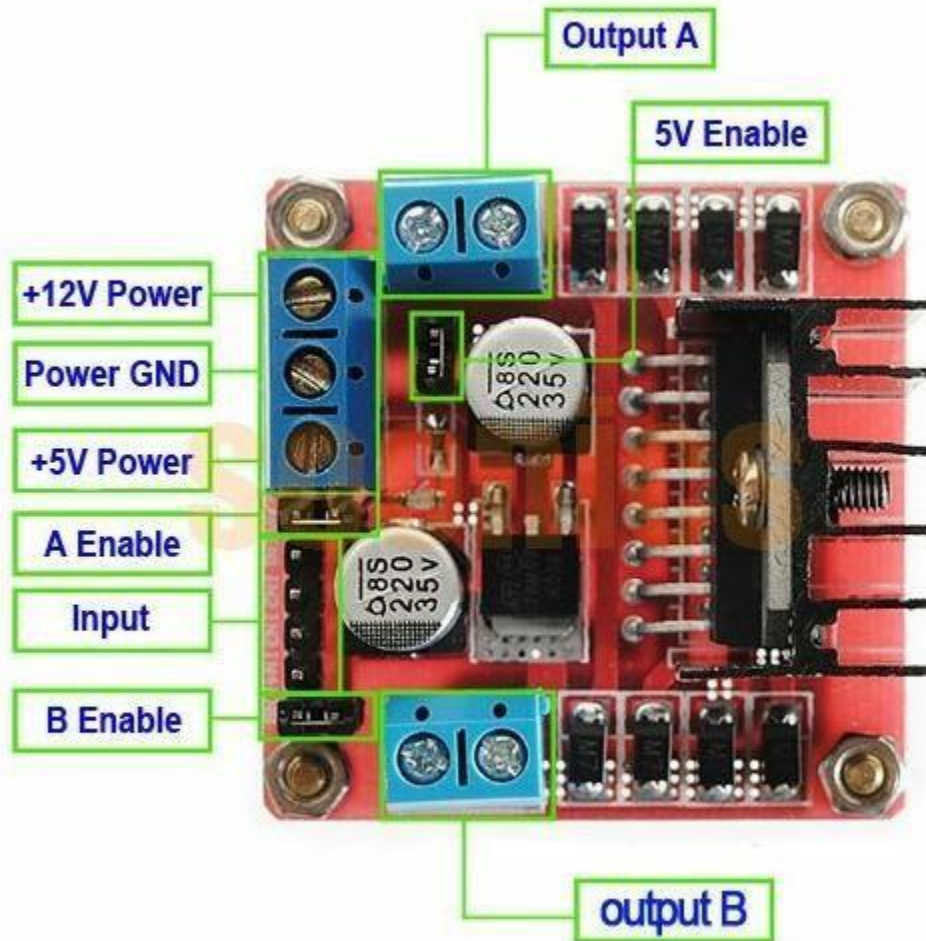
```

Bài tập 1: vừa nhấp nháy đèn chu kì 1s vừa điều khiển bật tắt 1 đèn khác tắt bằng nút bấm.

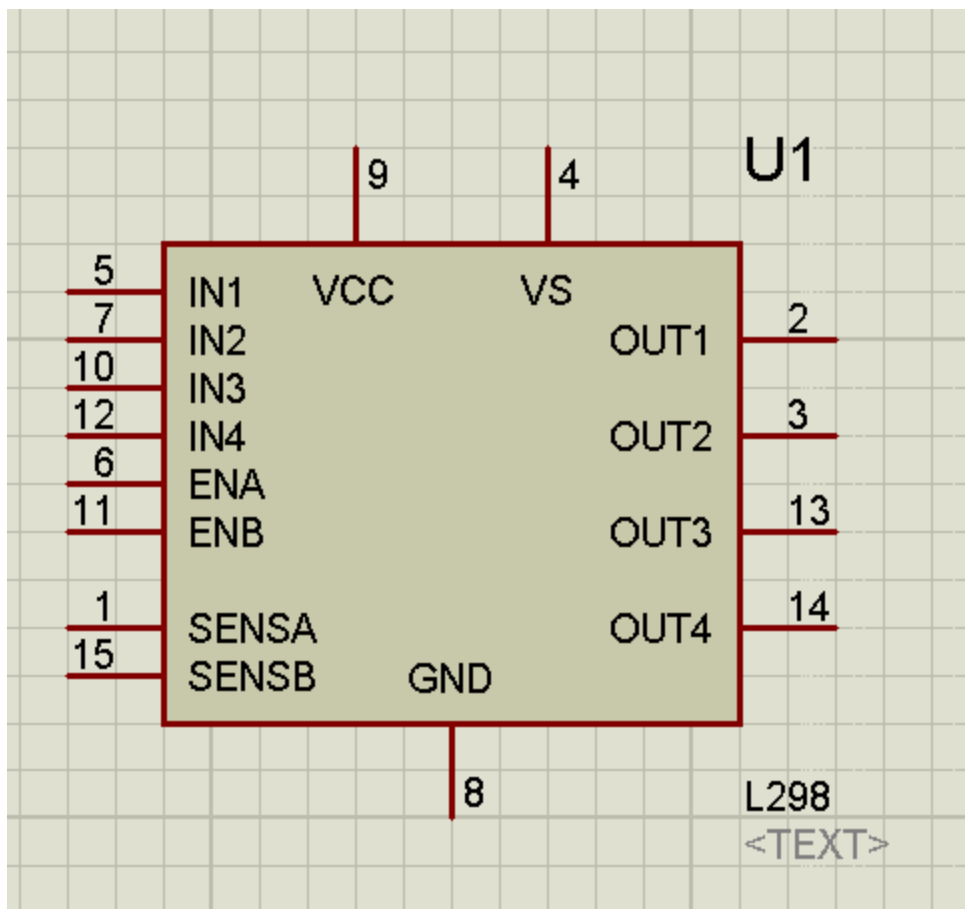
Bài tập 2: quét led 7 đoạn sử dụng ngắt timer, kết hợp với 2 nút nhấn tăng giảm.

BÀI 12 ĐIỀU KHIỂN ĐỘNG CƠ DC

Module điều khiển động cơ DC: L293D,L298N



Module L298N



Chức năng các chân.

Vcc - chân nối nguồn 5v

Vs - chân nối nguồn cho động cơ từ 5-46V

IN1-IN2 điều khiển OUT1 OUT2

IN1=1,IN2=0: động cơ quay thuận

IN1=0,IN2=1: động cơ quay nghịch

IN1=IN2=0 hoặc IN1=IN2=1 động cơ dừng

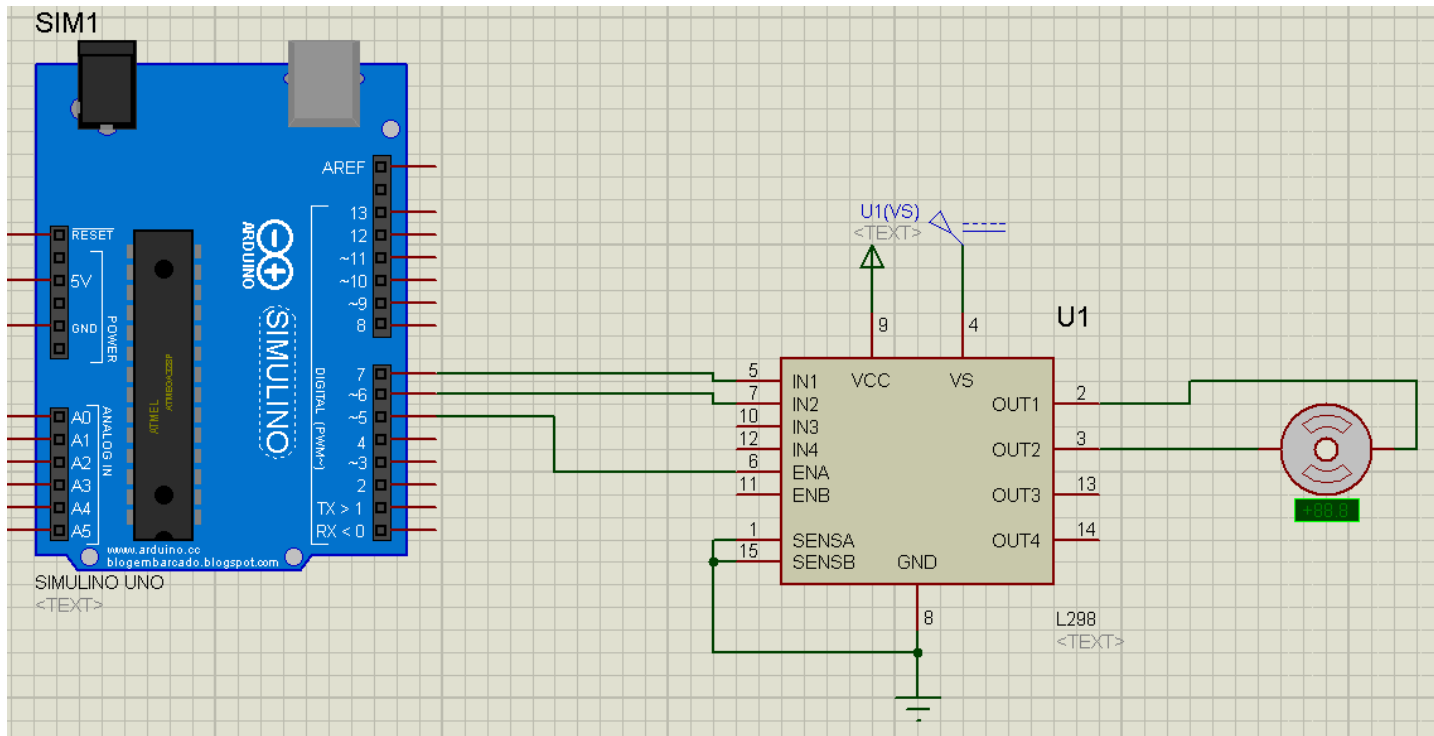
IN3-IN4 điều khiển OUT3 OUT4

ENA=1-cho phép ngõ ra OUT1-OUT2, ENA=0 không cho phép

ENB-cho phép ngõ ra OUT3-OUT4

SENSA- SENSEB nối 0v.

Ví dụ: thiết kế mạch điều khiển động cơ quay thuận 5s quay nghịch 5s.



//định nghĩa các chân nối tới L298

```
#define IN1 7
```

```
#define IN2 6
```

```
#define EN 5
```

```
void quayThuan()//để động cơ quay thuận IN1=1,IN2=0,EN=1;
```

```
{
```

```
    digitalWrite(IN1,HIGH);
```

```
    digitalWrite(IN2,LOW);
```

```
    digitalWrite(EN,HIGH);
```

```
}
```

```
void quayNgich()//để động cơ quay thuận thì IN1=1,IN2=0,EN=1;
```

```
{
```

```
    digitalWrite(IN1,LOW);
```

```
    digitalWrite(IN2,HIGH);
```

```

digitalWrite(EN,HIGH);
}
void setup() {
  // cấu hình các chân là ngõ ra
  pinMode(IN1,OUTPUT);
  pinMode(IN2,OUTPUT);
  pinMode(EN,OUTPUT);
}

void loop() {
  quayThuan();
  delay(5000);
  quayNghich();
  delay(5000)
}

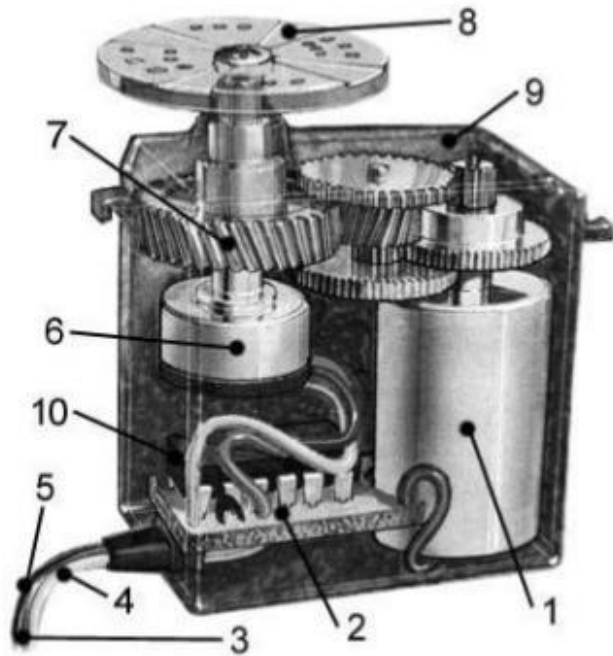
```

Bài tập 1 : thiết kế mạch điều khiển động cơ với 3 nút nhấn
 ấn nút thứ nhất: động cơ quay thuận
 ấn nút thứ hai: động cơ quay nghịch
 ấn nút thứ 3 : động cơ dừng


Bài tập 2: như bài 1 nhưng kết hợp LCD hiển thị động cơ đang chạy hay dừng.

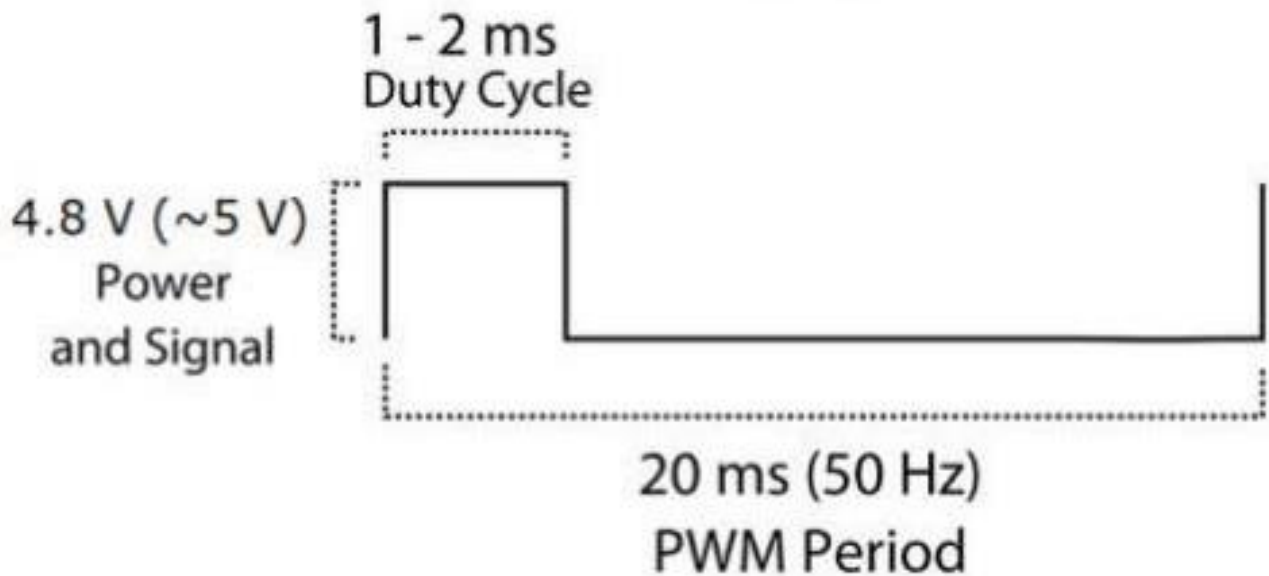
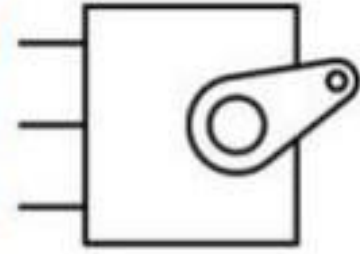
BÀI 13 ĐIỀU KHIỂN ĐỘNG CƠ SERVO

Cấu tạo động cơ servo



1. Động cơ
2. Mạch điện tử
3. Dây dương (đỏ)
4. Dây tín hiệu (vàng hoặc trắng)
5. Dây âm hoặc dây mát (đen)
6. Đồng hồ đo điện thế
7. Trục ra
8. Đầu nối gắn trên động cơ
9. Vỏ động cơ
10. Chip điều khiển tích hợp trên mạch điện tử

PWM=Orange ()
Vcc = Red (+)
Ground=Brown (-)



Nguyên lí hoạt động của động cơ servo

Lập trình trên arduino.

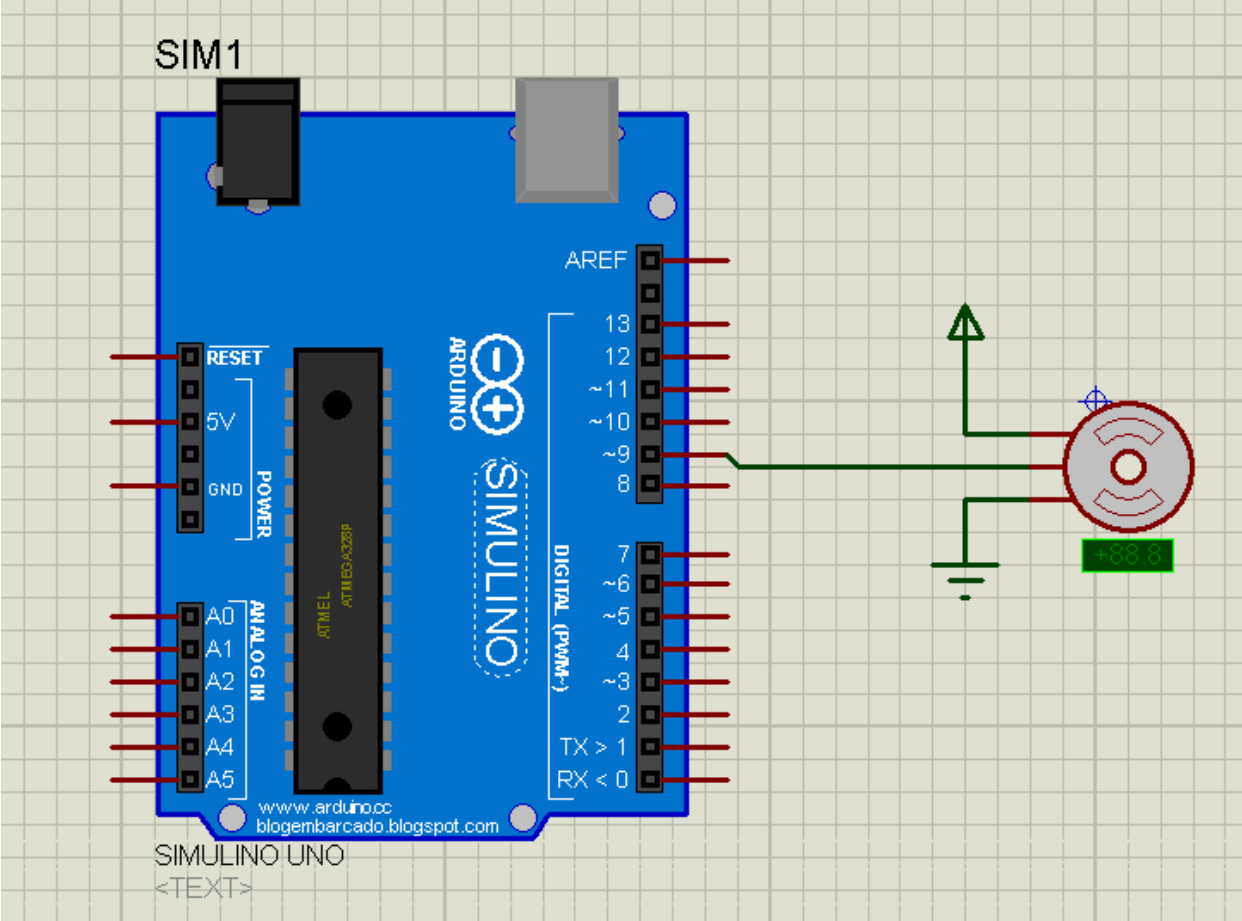
Ta sử dụng thư viện servo bằng cách `#include <Servo.h>`

Các hàm quan trọng: `servo.attach(pin)` pin chân điều khiển servo.

`servo.attach(pin,minPulse,maxPulse);`

`servo.write(goc)` quay servo đến vị trí goc (goc có giá trị từ 0-180)

Ví dụ:

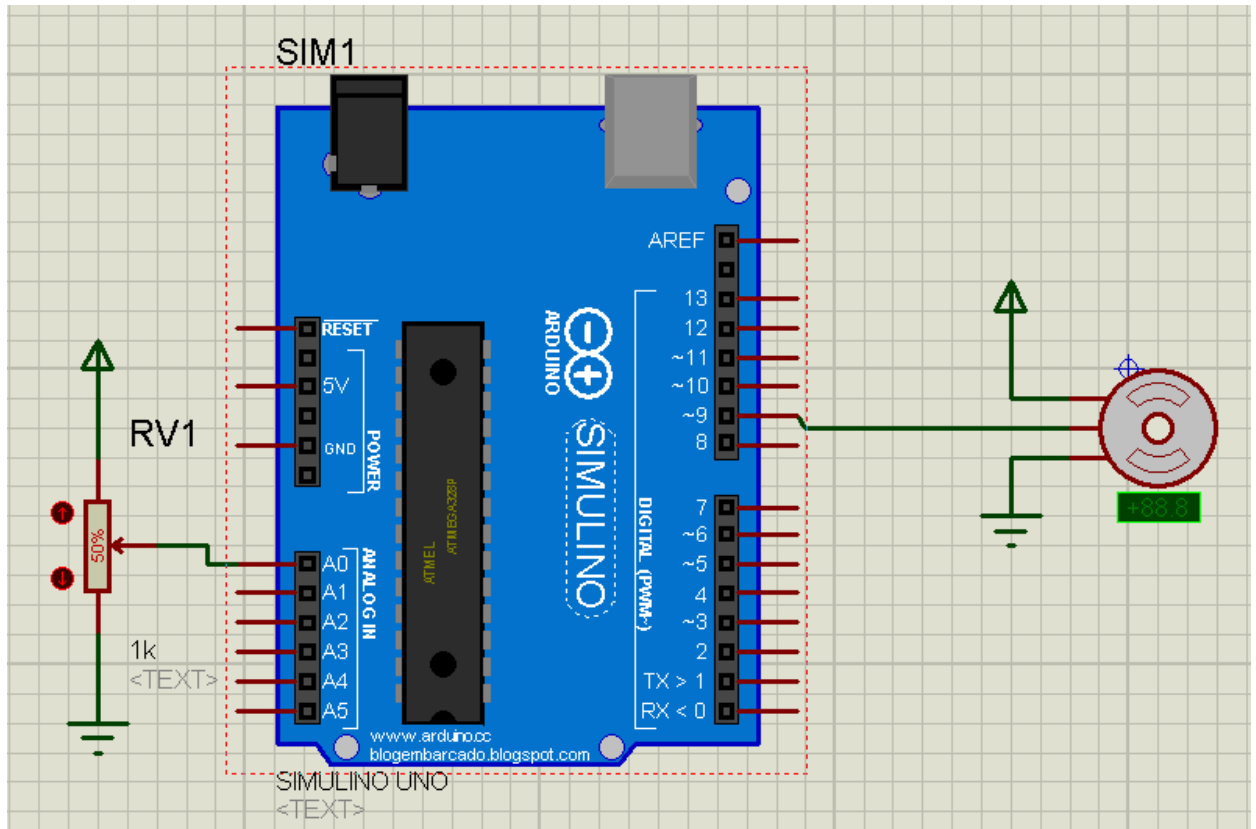


```

1 #include <Servo.h>
2
3 Servo myservo; // tạo một đối tượng servo
4
5 int pos = 0;    // biến lưu vị trí của servo
6
7 void setup() {
8   myservo.attach(9); // chân xung của động cơ nối vào chân số 9 của arduino
9 }
10
11 void loop() {
12   for (pos = 0; pos <= 180; pos += 1) {
13     myservo.write(pos);           //cho quay đến góc pos
14     delay(15);
15   }
16   for (pos = 180; pos >= 0; pos -= 1) {
17     myservo.write(pos);           //cho quay đến góc pos
18     delay(15);
19   }

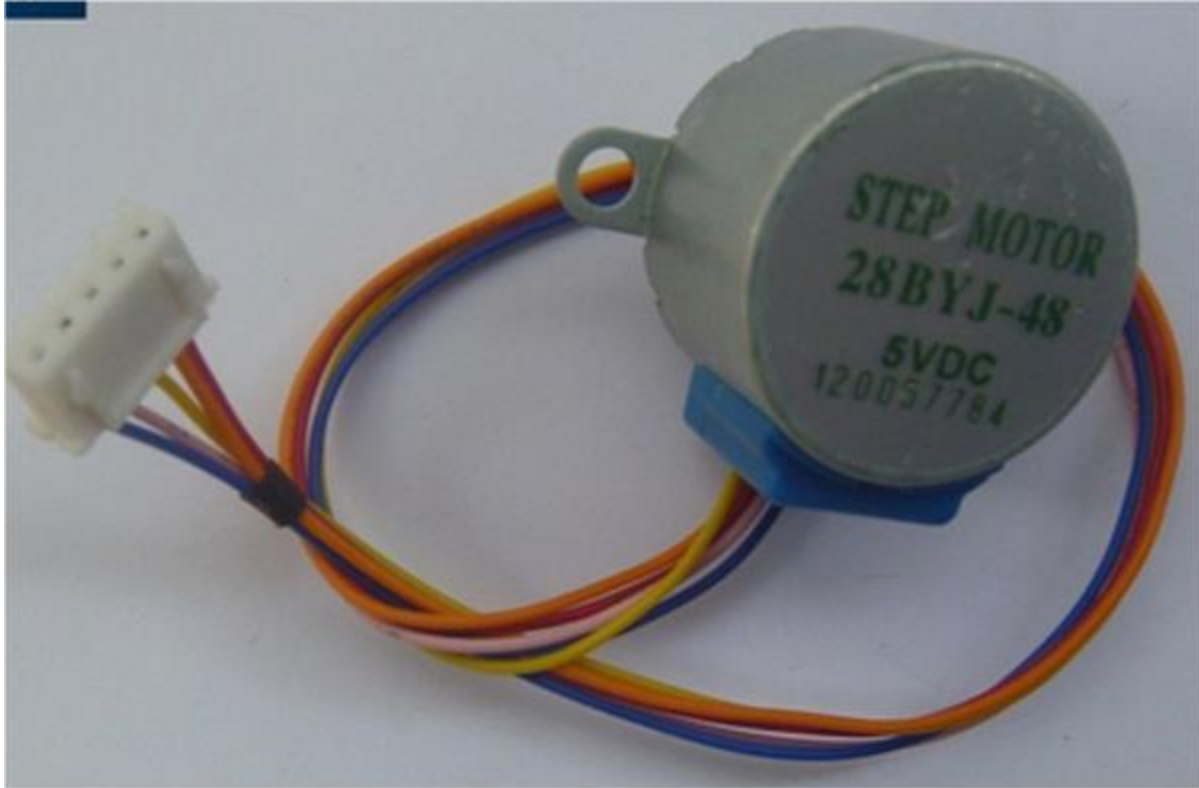
```

Bài tập: dùng biến trở điều khiển động cơ servo ,biến trở quay từ vị trí 0%-100% thì động cơ quay một góc 0-180 độ,



BÀI 14 ĐIỀU KHIỂN ĐỘNG CƠ BƯỚC.

Động cơ bước là một loại động cơ mà ở đó bạn sẽ có thể quy định chính xác số góc quay và động cơ bước sẽ phải quay. Không giống như Servo, động cơ bước có thể quay bao nhiêu độ tùy ý và mỗi lần quay nó sẽ quay được 1 step, 1 step ở đây là bao nhiêu còn phụ thuộc vào động cơ bước của bạn. Ví dụ, động cơ bước của bạn có 72 step thì nó sẽ cần quay 72 step để hoàn thành một vòng quay. Số step này là hằng số, nhưng bạn có thể dùng công nghệ micro step để "cải thiện" số vòng quay động cơ bước của bạn.



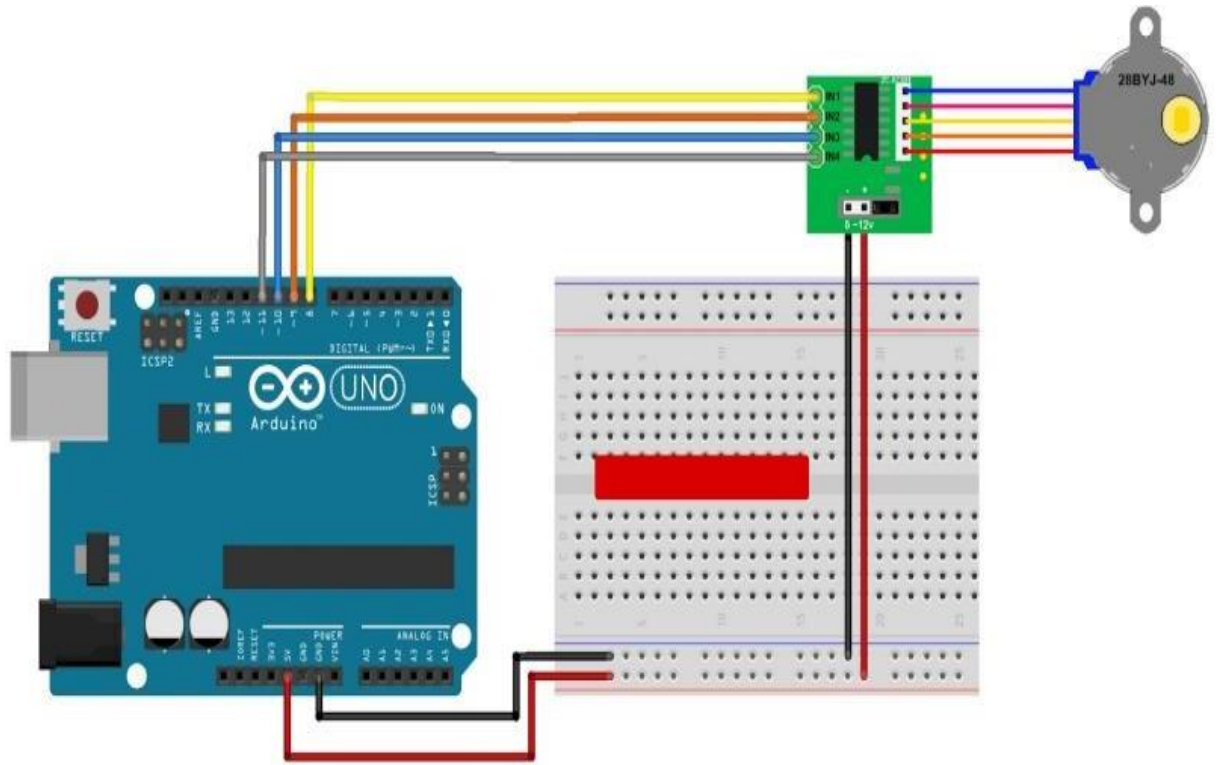
Động cơ bước sử dụng trong bài toán là động cơ bước 4 pha (thực ra là 2 pha được chia ra làm 2 ở mỗi pha ngay tại vị trí giữa) (gồm 5 dây), 4 trong 5 dây này được kết nối với 2 cuộn dây trong động cơ và 1 dây là dây nguồn chung cho cả 2 cuộn dây. Mỗi bước của động cơ quét 1 góc 5.625 độ, vậy để quay 1 vòng động cơ phải thực hiện 64 bước. Do trong động cơ bước có hộp giảm tốc tỉ lệ 1/64 cho nên phải thực hiện $64 \cdot 64 = 4096$ bước thì động cơ mới quay được một vòng.

Driver điều khiển động cơ bước ULN2003.



Ví dụ: cho động cơ bước quay cùng chiều kim đồng hồ 1 vòng sau đấy quay ngược lại.

Sơ đồ kết nối tới arduino



```

1 #include <Stepper.h>
2
3 const int stepsPerRevolution = 4096; //số bước để quay hết một vòng
4
5 Stepper myStepper(stepsPerRevolution, 8, 9, 10, 11); // nối 4 dây của động cơ bước
6 //vào 4 chân arduino 8, 9,10,11
7 void setup() {
8   myStepper.setSpeed(10); //tốc độ
9   Serial.begin(9600);
10 }
11
12 void loop() {
13
14   Serial.println("clockwise");
15   myStepper.step(stepsPerRevolution); //thực hiện quay 4096 bước tức là quay 1 vòng
16   delay(500);
17
18   Serial.println("counterclockwise");
19   myStepper.step(-stepsPerRevolution); //quay ngược lại
20   delay(500);
21 }

```

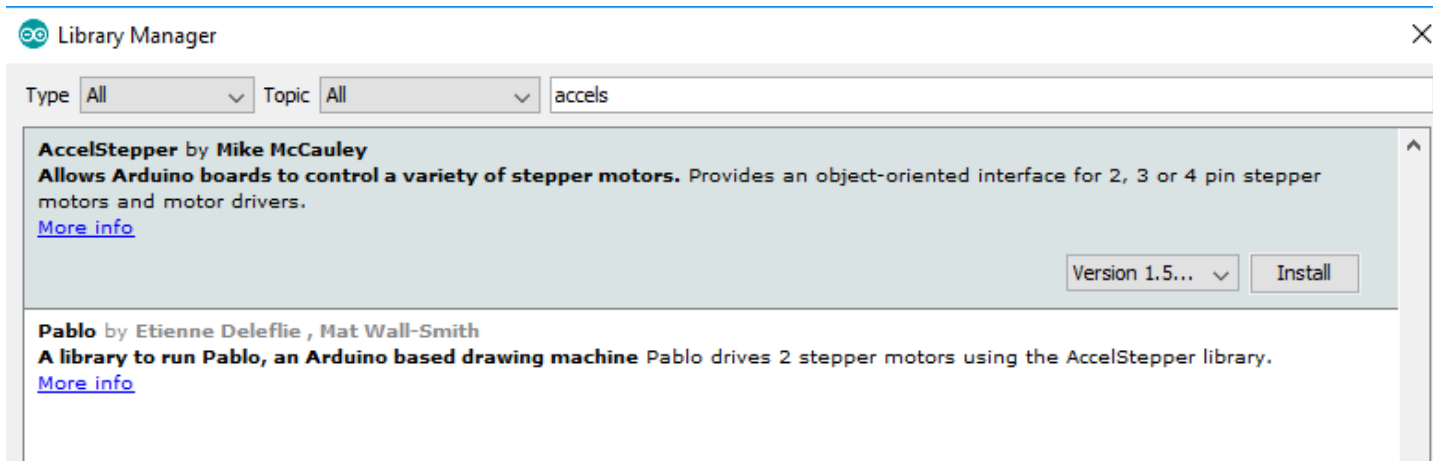
Thư viện trên có ưu điểm: tích hợp sẵn trong arduino IDE.

Nhược điểm: không điều khiển cùng lúc được nhiều động cơ.

Vậy làm thế nào có thể điều khiển được nhiều động cơ?

Thật may mắn cho ta vì đã có thư viện AccelSteper cho phép ta điều khiển nhiều động cơ cùng 1 lúc.

Để thêm thư viện ta vào Sketch > Include Library > Manage Library tìm có từ khóa accelstepper kích vào install để cài đặt.



Để sử dụng ta `#include <AccelStepper.h>`

Một số phương thức quan trọng.

`AccelStepper stepper1(AccelStepper::FULL4WIRE, pin1, pin2, pin3, pin4);`//khởi tạo khi điều khiển chế độ 4 dây

`AccelStepper stepper2(AccelStepper::FULL2WIRE,step, dir);`//khởi tạo khi điều khiển chế độ 2 dây

`stepper1.setSpeed(speed);`//đặt tốc độ

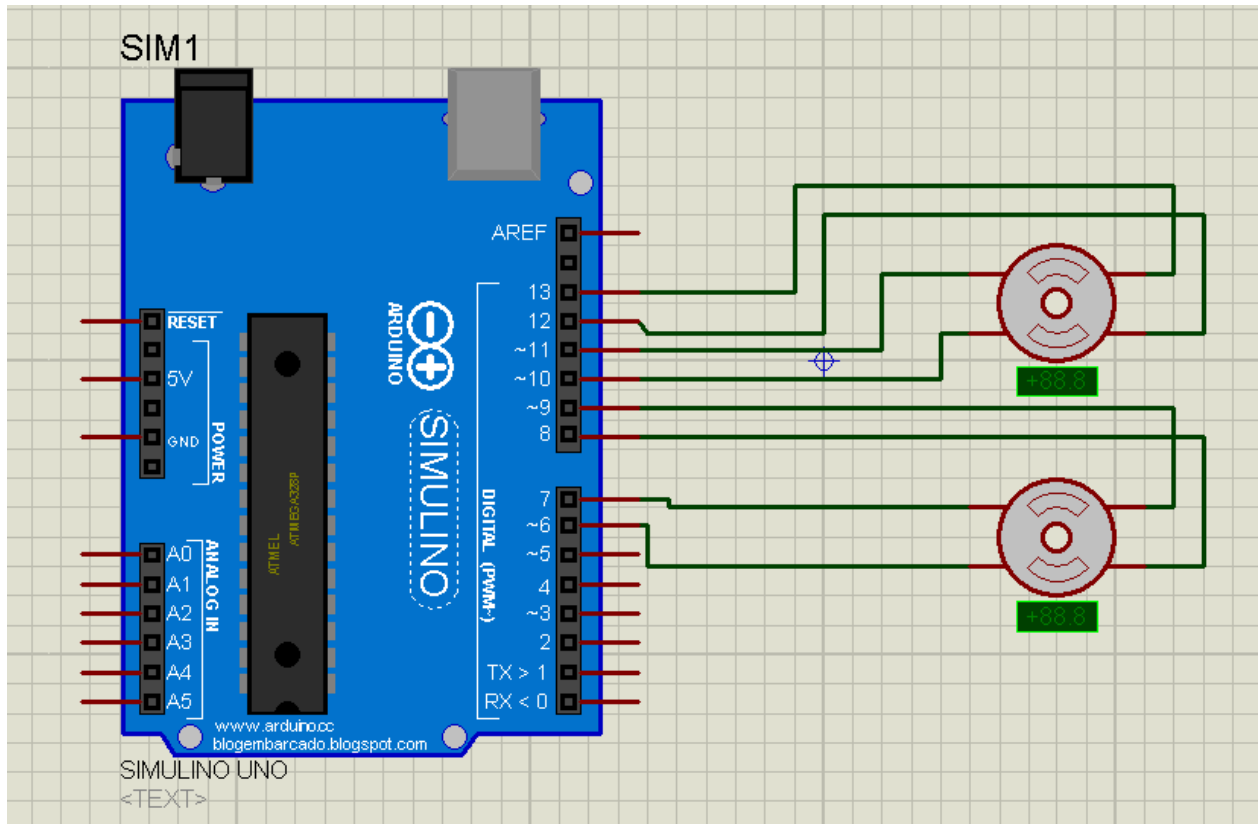
`stepper1.setAcceleration(accel);`//đặt gia tốc

`stepper1.moveTo(absolute)`//cho động cơ chạy đến vị trí tuyệt đối so với vị trí ban đầu tính theo bước.

`stepper1.move(relative)`//cho động cơ chạy đến vị trí so với vị trí hiện tại của động cơ tính theo bước.

`stepper1.distanceToGo()`//kiểm tra xem động cơ đã chạy tới vị trí đã được đặt bởi hàm `moveTo` hoặc `move` chưa

Ví dụ: điều khiển 2 động cơ quay ngược chiều nhau

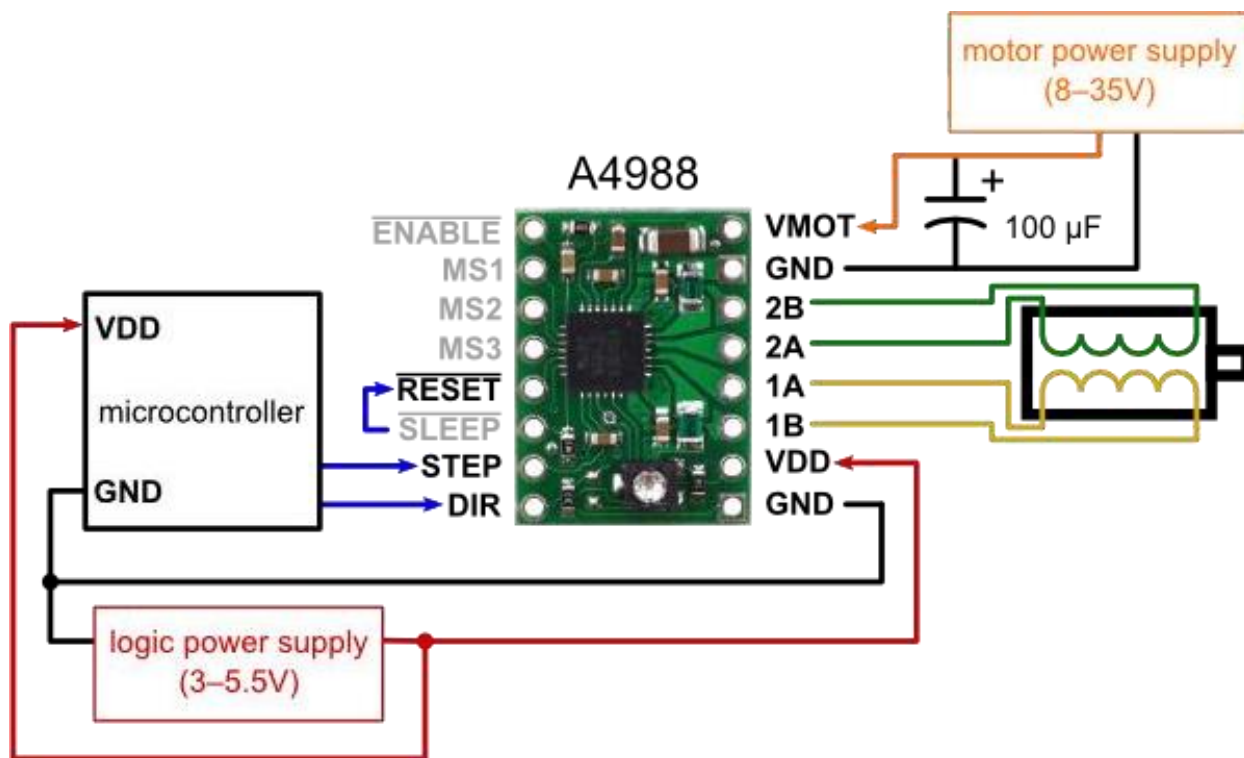


```

1 #include <AccelStepper.h>
2 AccelStepper stepper1(AccelStepper::FULL4WIRE, 6, 7, 8, 9);
3 AccelStepper stepper2(AccelStepper::FULL4WIRE, 10, 11, 12, 13);
4
5 void setup()
6 {
7     stepper1.setSpeed(1);
8     stepper1.setAcceleration(0);
9     stepper1.moveTo(100);
10    stepper2.setSpeed(1);
11    stepper2.setAcceleration(0);
12    stepper2.moveTo(-100);
13
14 }
15 void loop()
16 {
17     stepper1.run();
18     stepper2.run();
19 }

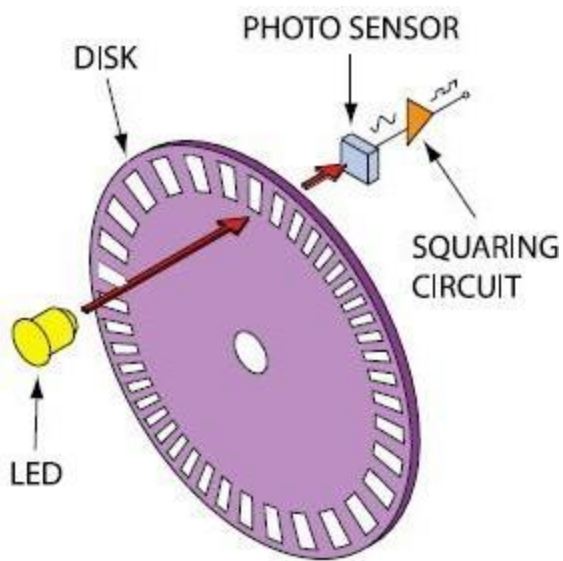
```


Giới thiệu driver A4988.



BÀI 15 ĐIỀU KHIỂN VÀ ĐO TỐC ĐỘ ĐỘNG CƠ DC SỬ DỤNG ENCODER

1, Giới thiệu encoder



Encoder thường sẽ có 4 dây, 2 dây cấp nguồn và 2 dây(kênh A,B) xuất tín hiệu ra.

Encoder có cấu tạo gồm 1 đĩa có chia rãnh (được gọi là số xung). khi quay hết 1 vòng thì số xung tương ứng sẽ xuất hiện ở .

Ví dụ encoder 300 xung thì khi quay một vòng sẽ trả về mỗi kênh A và B là 300 xung 2 kênh này lệch nhau 90 độ để ta dễ dàng xác định được chiều quay.

Thuật toán đo tốc độ động cơ:

Vì encoder trả về xung nên ta sẽ tạo một ngắt ngoài để đọc xung. Cứ mỗi lần xảy ra ngắt ta sẽ tăng biến đếm lên.

Ta tạo tiếp một bộ định thời gian chẳng hạn như 300 ms ta sẽ đo vận tốc 1 lần. ta kiểm tra biến đếm xem có giá trị bao nhiêu ta sẽ đo đc vận tốc.

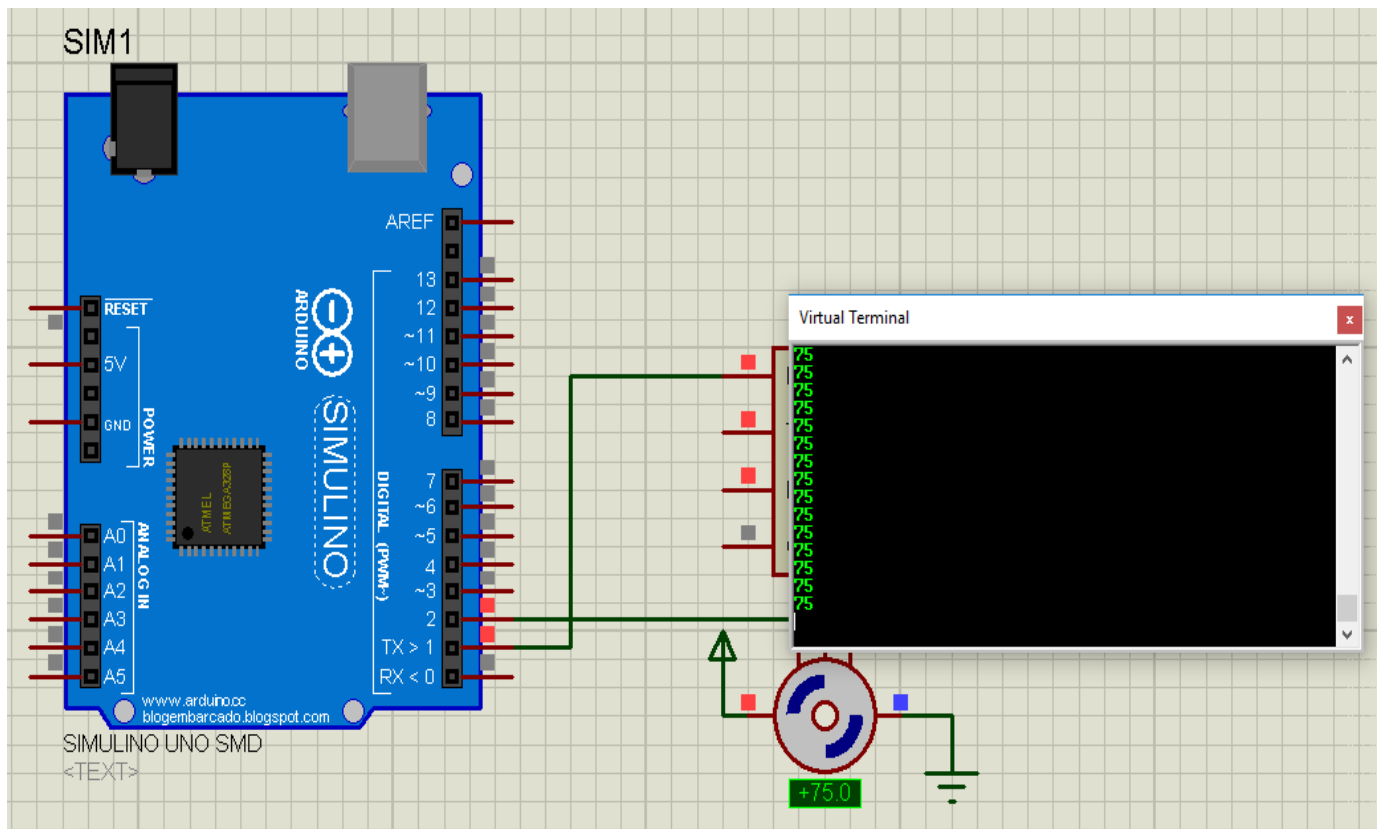
Công thức tính vận tốc:

$$\text{vantoc} = \text{xung} * 1000 * 60 / (\text{dophangiai} * \text{tglm}) \text{ (vòng/phút);}$$

trong đây:

- tglm: thời gian lấy mẫu (ms).
- xung :số xung đo được trong khoảng tglm (ms)
- dophangiai: số xung của encoder.

Hướng dẫn đo mô phỏng trên proteus.



```
int xung=0;
long t=0,told=0;
int vantoc=0;
const int tglm=300;//ms
const int dophangiai=400;
```

```

void setup() {
  pinMode(2,INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(2), docxung, RISING);
  Serial.begin(9600);
}

void loop() {
  t=millis();//trả về thời gian hiện tại
  if(t-told>=tglm)
  {
    told=t;
    vantoc=(long)xung*1000*60/((long)dophangiai*tglm);//vong/giay
    //Serial.println(xung);
    xung=0;
    Serial.println(vantoc);

  }
}

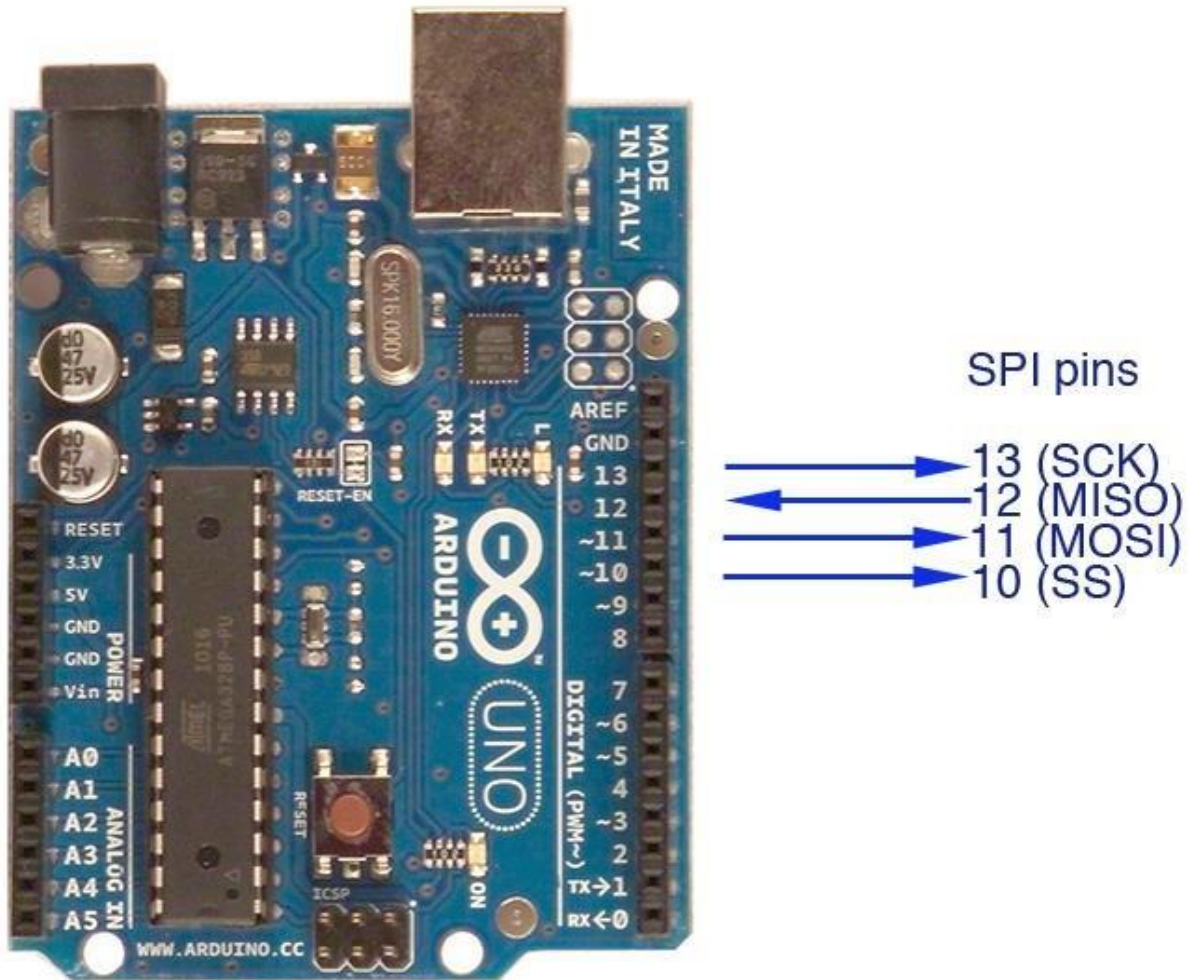
void docxung()
{
  xung++;
}

```

BÀI 16 GIAO TIẾP SPI- ĐỌC THẺ TỬ RFID RC522

I, Giao tiếp SPI

SPI (Serial Peripheral Bus) là một chuẩn truyền thông nối tiếp tốc độ cao do hãng Motorola đề xuất. Đây là kiểu truyền thông Master-Slave, trong đó có 1 chip Master điều phối quá trình truyền thông và các chip Slaves được điều khiển bởi Master vì thế truyền thông chỉ xảy ra giữa Master và Slave. SPI là một cách truyền song công (full duplex) nghĩa là tại cùng một thời điểm quá trình truyền và nhận có thể xảy ra đồng thời. SPI đôi khi được gọi là chuẩn truyền thông “4 dây” vì có 4 đường giao tiếp trong chuẩn này đó là SCK (Serial Clock), MISO (Master Input Slave Output), MOSI (Master Output Slave Input) và SS (Slave Select).



- **MISO** - Mang các dữ liệu từ các thiết bị SPI về arduino
- **MOSI** - Mang các dữ liệu từ Arduino đến các thiết bị SPI
- **SS** - Chọn thiết bị SPI cần làm việc
- **SCK** - dòng đồng bộ

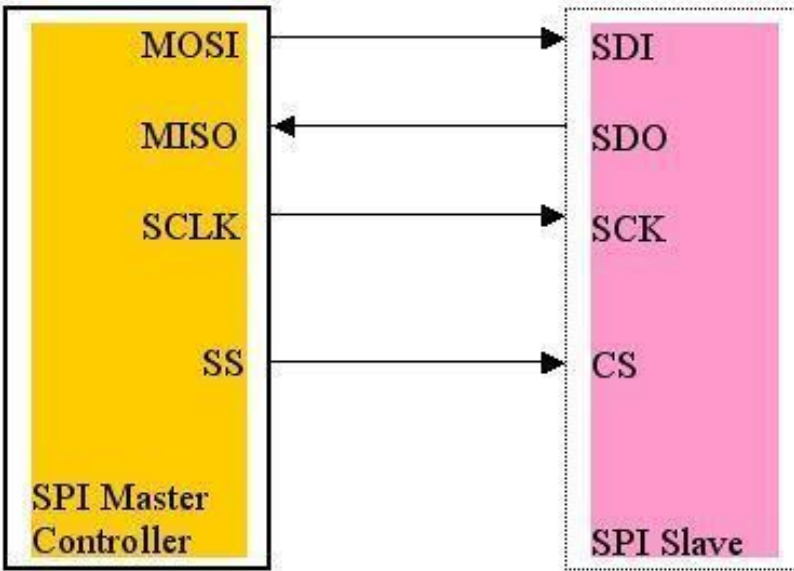
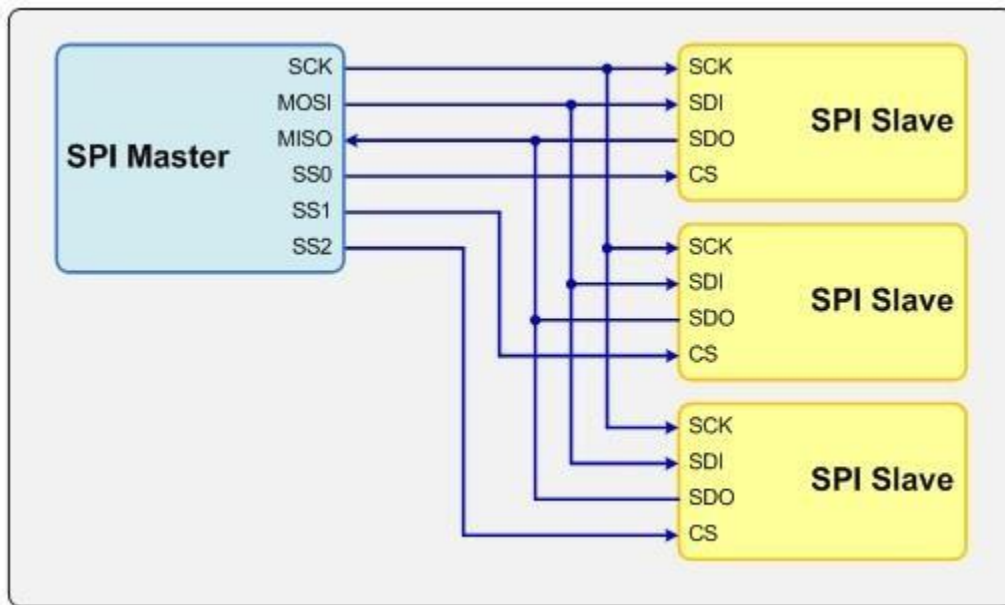


Fig-1 (Single master, single slave SPI implementation)

Kết nối 1 master với 1 slaver



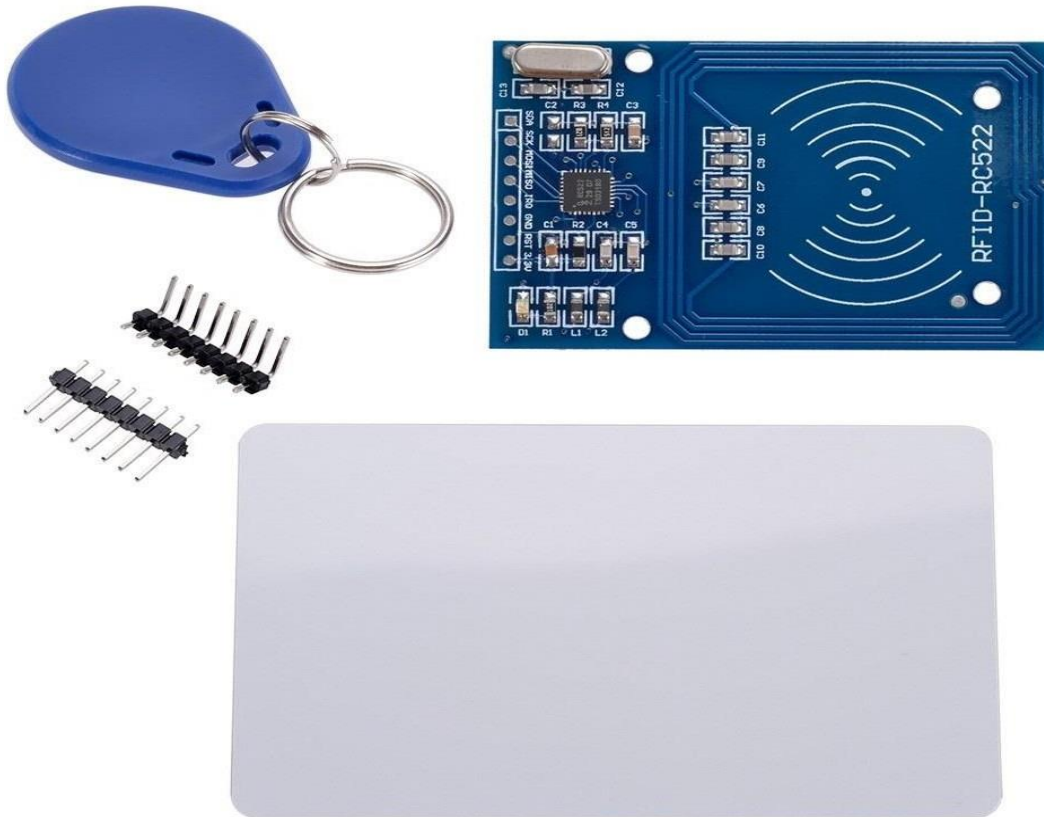
Kết nối 1 master với nhiều slaver

II, Giao tiếp với thẻ từ RFID RC522

Thư viện RFID chưa có sẵn trong IDE nên chúng ta phải thêm vào.

Link tải thư viện : http://k3.arduino.vn/img/2017/04/11/0/3664_812450-1491920019-0-rfid.zip

Thêm thư viện bằng cách: Sketch > Include Library > add .Zip Library chọn đến file vừa tải về.



Sơ đồ nối chân

Arduino uno	RFID RC522	Ghi chú
MOSI-11	MOSI	Chân cố định không được thay đổi
MISO-12	MISO	Chân cố định không được thay đổi
SCK-13	SCK	Chân cố định không được thay đổi
SS-10	SDA	Có thể thay đổi chân bất kì
RST-9	RST	Có thể thay đổi chân bất kì
3.3V	3.3V	
GND	GND	

Để sử dụng ta phải khai báo 2 thư viện sau

```
#include <SPI.h> //khai báo thư viện SPI
```

```
#include <RFID.h> //khai báo thư viện RFID
```

```
RFID rfid(SS_PIN, RST_PIN); // khai báo đối tượng gồm 2 tham số chân SS và chân RST kết nối tới arduino.
```

Một số phương thức cần lưu ý:

```
SPI.begin(); // khởi tạo giao tiếp SPI
```

```
rfid.init(); // khởi tạo đối tượng rfid
```

```
rfid.isCard() // kiểm tra xem có thẻ hay không
```

```
rfid.readCardSerial() // kiểm tra xem nếu đọc
```

```
void ReadCard(unsigned char card[])
```

```
{
```

```
  for (i = 0; i < 5; i++) {
```

```
    card[i] = rfid.serNum[i]; //Luu mã thẻ đọc được vào mảng card
```

```
  }
```

```
}
```

```
rfid.halt(); // cho phép tiếp tục đọc thẻ
```

Ví dụ: đọc thẻ từ RFID hiển thị lên Serial.


```

8 #include <SPI.h>
9 #include <RFID.h>
10
11 #define SS_PIN 10
12 #define RST_PIN 9
13 RFID rfid(SS_PIN, RST_PIN);
14 unsigned char reading_card[5]; // Mảng đọc mã card
15 unsigned char i, j;
16
17 void setup()
18 {
19
20     Serial.begin(9600);
21     SPI.begin();
22     rfid.init();
23 }
24
--

```

Bài tập : đọc 2 thẻ từ và điều khiển servo. Thẻ thứ nhất quay động cơ 90 độ, thẻ thứ 2 quay về 0 độ.

```

27 void loop()
28 {
29     if (rfid.isCard()) { //nếu có thẻ
30         if (rfid.readCardSerial()) // Nếu đọc được mã thẻ
31             {
32
33                 for (i = 0; i < 5; i++) {
34
35                     reading_card[i] = rfid.serNum[i]; //Luu mã thẻ đọc được vào mảng reading_card
36                 }
37                 for (i = 0; i < 5; i++) {
38                     Serial.print(reading_card[i]);
39                 }
40                 Serial.print("\n");
41             }
42             rfid.halt();
43         }
44 }

```

BÀI 17 GIAO TIẾP UART - GIAO TIẾP GIỮA HAI 2 ARDUINO

I, Giới thiệu về UART

UART là viết tắt của Universal Asynchronous Receiver – Transmitter. Thường là một mạch tích hợp được sử dụng trong việc truyền dẫn dữ liệu nối tiếp giữa máy tính và các thiết bị ngoại vi. Rất nhiều vi điều khiển hiện nay đã được tích hợp UART, vì vấn đề tốc độ và độ điện dụng của UART không thể so sánh với các giao tiếp mới hiện nay nên các dòng PC & Laptop đời mới không còn tích hợp cổng UART. Như các bạn đã biết giao tiếp SPI và I2C có 1 dây truyền dữ liệu và 1 dây được sử dụng để truyền xung clock (SCL) để đồng bộ trong giao tiếp. Với UART thì không có dây SCL, vấn đề được giải quyết khi mà việc truyền UART được dùng giữa 2 vi xử lý với nhau, đồng nghĩa với việc mỗi vi xử lý có thể tự tạo ra xung clock cho chính nó sử dụng.

Baud rate (tốc độ baud): Khoảng thời gian dành cho 1 bit được truyền. Phải được cài đặt giống nhau ở gửi và nhận.

- **Frame** (khung truyền): Khung truyền quy định về số bit trong mỗi lần truyền.

- **Start bit**: là bit đầu tiên được truyền trong 1 Frame. Báo hiệu cho thiết bị nhận có một gói dữ liệu sắp được truyền đến. Bit bắt buộc.

- **Data**: dữ liệu cần truyền. Bit có trọng số nhỏ nhất LSB được truyền trước sau đó đến bit MSB.

- **Parity bit**: kiểm tra dữ liệu truyền có đúng không.

- **Stop bit**: là 1 hoặc các bit báo cho thiết bị rằng các bit đã được gửi xong. Thiết bị nhận sẽ tiến hành kiểm tra khung truyền nhằm đảm bảo tính đúng đắn của dữ liệu. Bit bắt buộc.

II, Giao tiếp giữa 2 arduino với nhau qua chuẩn UART

Để giao tiếp chúng ta có thể dùng **Serial** cứng của arduino trên 2 chân 0-RX và 1-TX. nhưng vì 2 chân này phục vụ cho mục đích nạp code nên khi muốn dùng 2 chân này chúng ta phải tháo các dây đã kết nối trước đây với 2 chân này.

Khởi tạo trong hàm setup **Serial.begin(9600)** //khởi tạo tốc độ baud

Một số phương thức quan trọng

Serial.print(string); //in một chuỗi nhưng không xuống hàng.

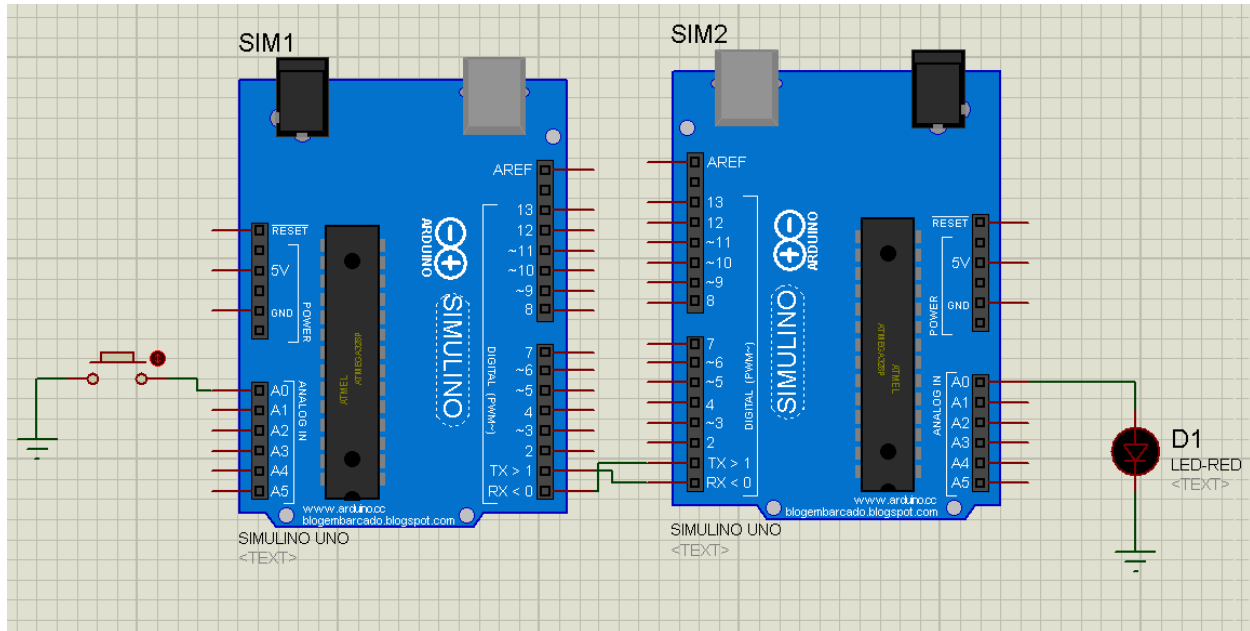
Serial.println(string); // in một chuỗi nhưng xuống hàng.

Serial.available(); //kiểm tra xem có dữ liệu truyền đến hay không

Serial.read(); //hàm này trả về một kí tự

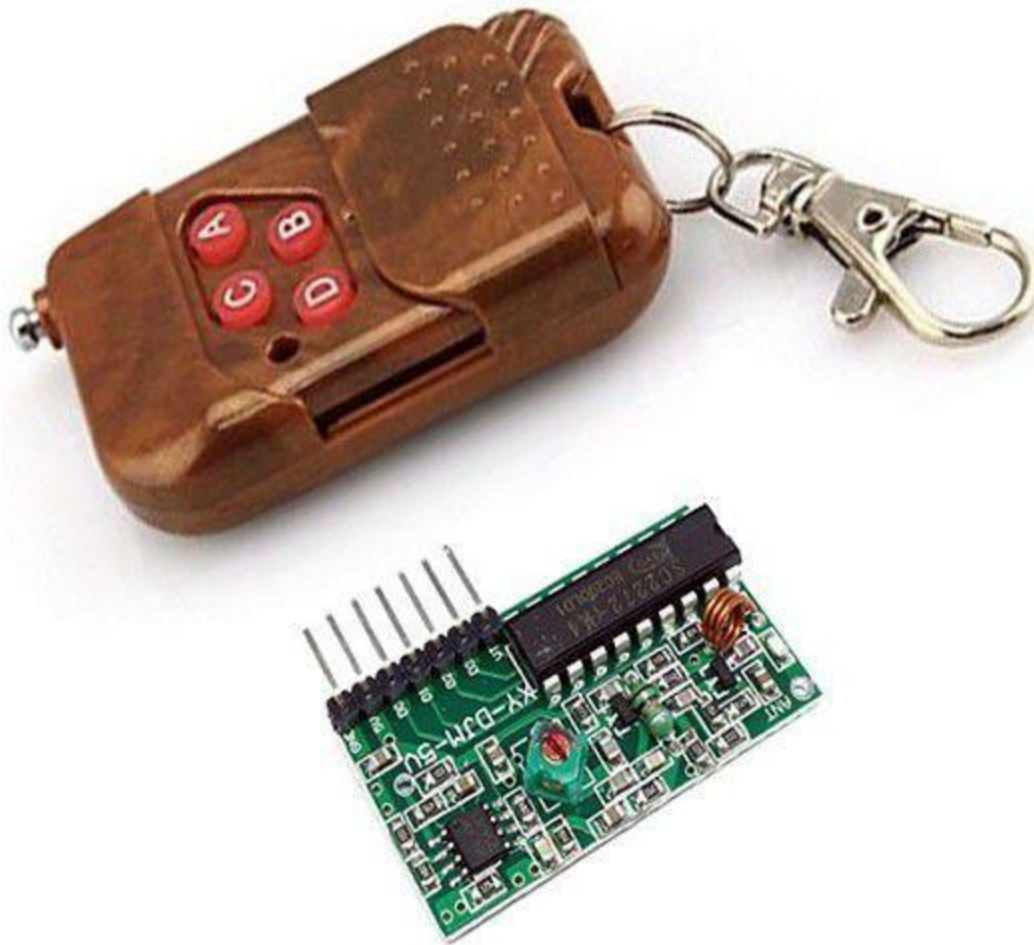
`Serial.readStringUntil(string)`//đọc toàn bộ chuỗi cho đến khi gặp chuỗi string nếu không gặp hoặc hết timeout trả về rỗng.

Ví dụ: kết nối nút nhấn vào arduino thứ nhất. điều khiển bóng đèn sáng tắt bên arduino thứ 2.



BÀI 18 ĐIỀU KHIỂN TỪ XA BẰNG RF

I, Giới thiệu module thu phát RF 4 kênh



Tay phát RF315 chip mã hóa PT2262

Bộ thu RF315 chip mã hóa PT2272 M4

- Điện áp hoạt động 5v

Nguyên lí hoạt động: khi chưa có tín hiệu các chân D0-D3 D<C<B<A ở mức thấp khi có tín hiệu các chân D0-D3 lên mức 1. Từ đây ta nối các chân vào arduino để điều khiển

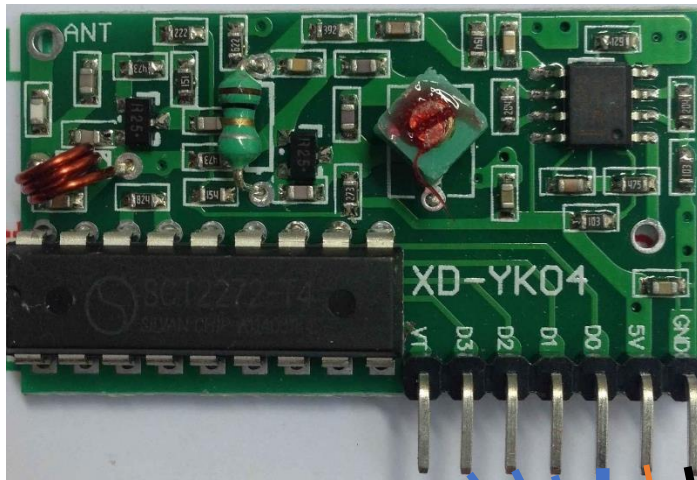
Có 3 loại RF315:

M4- khi có dữ liệu truyền đến các chân lên mức 1 khi không có dữ liệu thì các chân trở về 0.

T4- khi có dữ liệu truyền đến các chân lên mức 1 và chột lại. trở về 0 khi ấn lần thứ 2 .

L4- khi có dữ liệu truyền đến các chân lên mức 1 và chột lại. trở về 0 khi ấn lần thứ 2 hoặc nhấn nút khác.

II, kết hợp module với arduino



Ví dụ: sử dụng RF điều khiển bật tắt 4 bóng đèn led

```
int stateLed1=0,stateLed2=0,stateLed3=0,stateLed4=0;
```

```
void setup() {
```

```
    // put your setup code here, to run once:
```

```
    pinMode(2,INPUT);
```

```
    pinMode(3,INPUT);
```

```
    pinMode(4,INPUT);
```

```
    pinMode(5,INPUT);
```

```
}
```

```
void loop() {
```

```
  // put your main code here, to run repeatedly:
```

```
  if(digitalRead(2)==HIGH)
```

```
  {
```

```
    delay(20); //delay chong doi
```

```
    if(digitalRead(2)==HIGH)
```

```
    {
```

```
      stateLed1==0?1:0;
```

```
      digitalWrite(2,stateLed1);
```

```
      while(digitalRead(2)==HIGH);
```

```
    }
```

```
  }
```

```
  if(digitalRead(3)==HIGH)
```

```
  {
```

```
    delay(20); //delay chong doi
```

```
    if(digitalRead(3)==HIGH)
```

```
    {
```

```
      stateLed2==0?1:0;
```

```

digitalWrite(3,stateLed2);
while(digitalRead(3)==HIGH);
}
}
if(digitalRead(4)==HIGH)
{
delay(20);//delay chong doi
if(digitalRead(4)==HIGH)
{
stateLed1==0?1:0;
digitalWrite(4,stateLed1);
while(digitalRead(4)==HIGH);
}
}
if(digitalRead(4)==HIGH)
{
delay(20);//delay chong doi
if(digitalRead(4)==HIGH)
{
stateLed1==0?1:0;
digitalWrite(4,stateLed1);

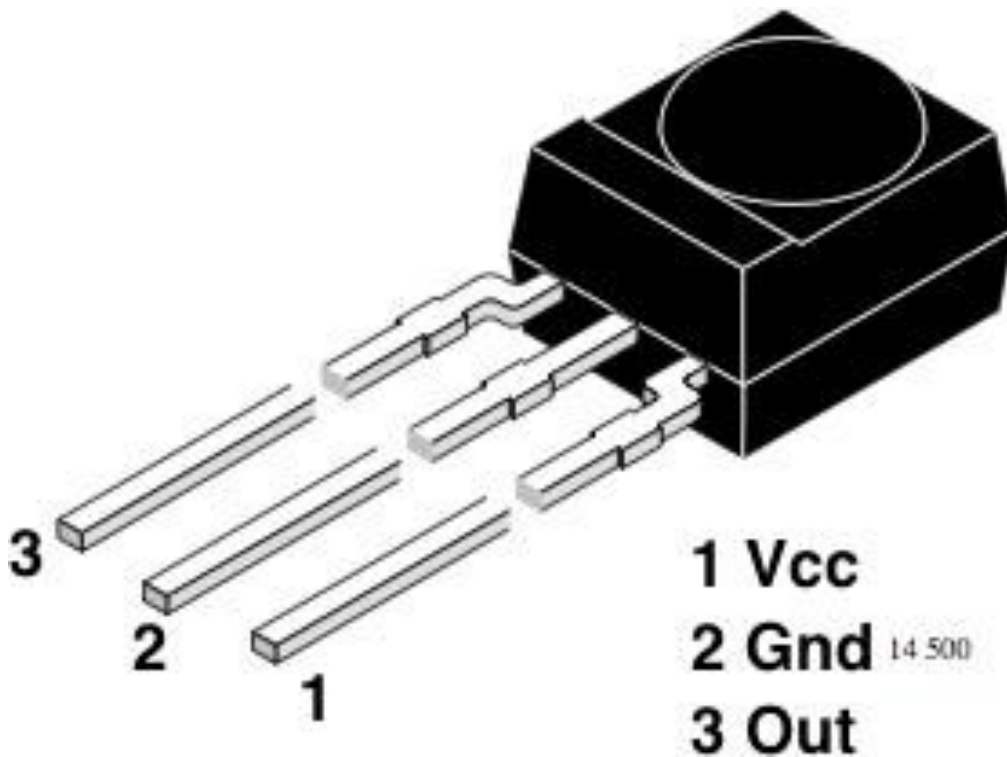
```



```
while(digitalRead(4)==HIGH);  
}  
}  
}
```

BÀI 19 ĐIỀU KHIỂN TỪ XA BẰNG HỒNG NGOẠI.

I, Mắt thu hồng ngoại



Remote hồng ngoại

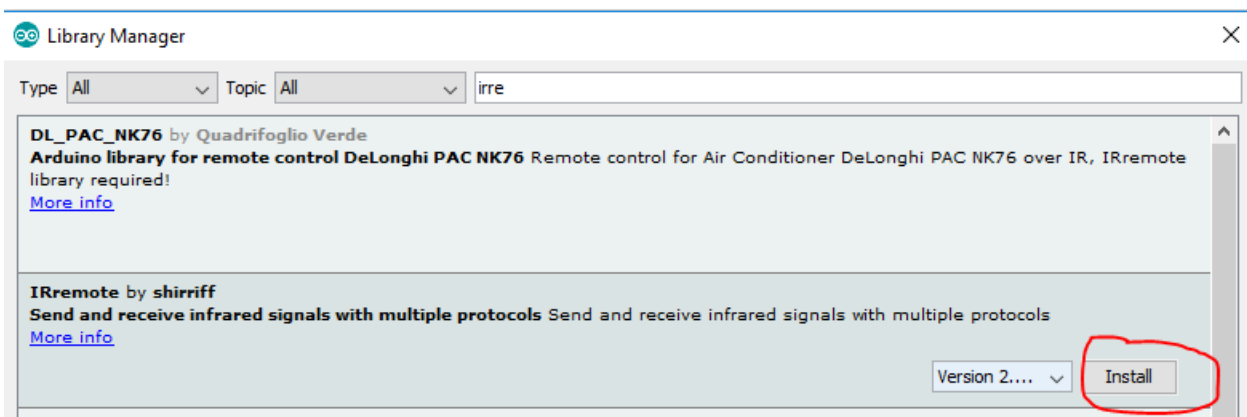


III, điều khiển từ xa bằng remote hồng ngoại

Để sử dụng ta phải thêm thư viện remote hồng ngoại

Sketch > Include Library > Manage Library

Tìm với từ khóa irre rồi kích vào install để cài đặt.



Để sử dụng thư viện #include <IRremote.h>

Một số phương thức chính:

IRrecv irrecv(PIN_INFR); // tạo đối tượng IRrecv mới với PIN_INFR là chân kết nối tới mắt thu hồng ngoại.

decode_results results; // results là lưu giữ kết quả giải mã tín hiệu

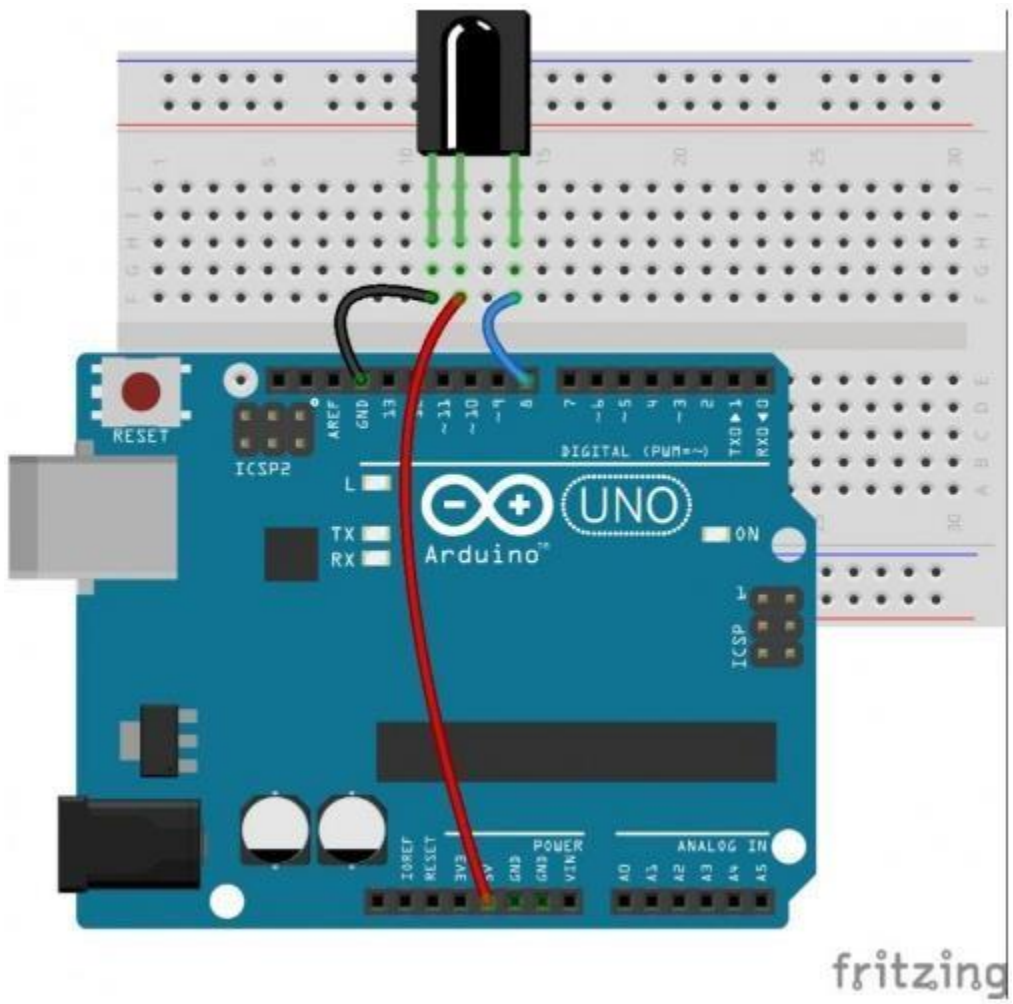
irrecv.**enableIRIn**(); // bắt đầu nhận dữ liệu

irrecv.**decode**(&results) // kiểm tra xem có đọc được mã remote nếu đọc được thì trả về true đồng thời lưu kết quả vào biến results ngược lại trả về false.

results.**value**; // trả về giá trị kiểu số nguyên unsigned long int.

irrecv.**resume**(); // nhận kết quả tiếp theo nếu không phương thức này thì chỉ đọc mã 1 lần duy nhất.

ví dụ: đọc giá trị từ Remote hồng ngoại hiển thị lên Serial



```

1 #include <IRremote.h> // thư viện hỗ trợ IR remote
2
3 const int receiverPin = 8; // chân digital 8 dùng để đọc tín hiệu
4 IRrecv irrecv(receiverPin); // tạo đối tượng IRrecv mới
5 decode_results results; // lưu giữ kết quả giải mã tín hiệu
6 void setup()
7 {
8   Serial.begin(9600); // khởi tạo serial
9   irrecv.enableIRIn(); // cho phép nhận tín hiệu hồng ngoại
10 }
11 void loop()
12 {
13   if (irrecv.decode(&results)) // nếu nhận được tín hiệu
14   {
15     Serial.println(results.value, HEX); // in ra Serial Monitor
16     delay(200);
17     irrecv.resume(); // nhận giá trị tiếp theo
18   }
19 }

```

Bài tập: điều khiển bật tắt bóng đèn bằng remote hồng ngoại.

BÀI 20 ĐIỀU KHIỂN TỪ XA BẰNG BLUETOOTH

1, Giới thiệu module bluetooth

Có 2 loại module bluetooth đây là HC 05 và HC 06;

HC 05 vừa có thể làm Master (kết nối tới các thiết bị khác) vừa có thể làm Slaver (các thiết bị khác kết nối đến)

H06 chỉ có thể làm slaver.



II Kết nối HC06 với arduino

Module HC06 hoạt động theo giao thức UART gồm 2 chân TX và RX để truyền và nhận tín hiệu cho arduino

Tốc độ baud mặc định 9600

Điện áp sử dụng 5v

Mật khẩu mặc định 1234ud

Ví dụ: điều khiển bật tắt đèn led qua module bluetooth.

Sơ đồ kết nối tới arduino.

