BÀI GIẢNG MÔN HỌC
LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Biên soạn:

GV: Trần Giang Nam

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

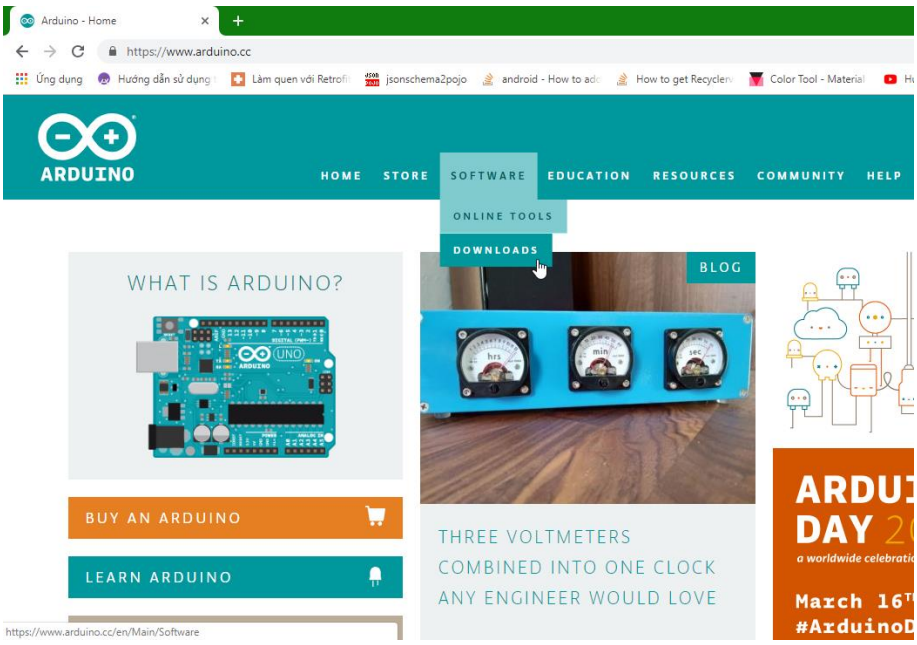
CHƯƠNG 1: TỔNG QUAN

Arduino là một bo mạch vi điều khiển do một nhóm giáo sư và sinh viên 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 từ 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

1.1 Cài đặt IDE (Intergrated Development Environment)

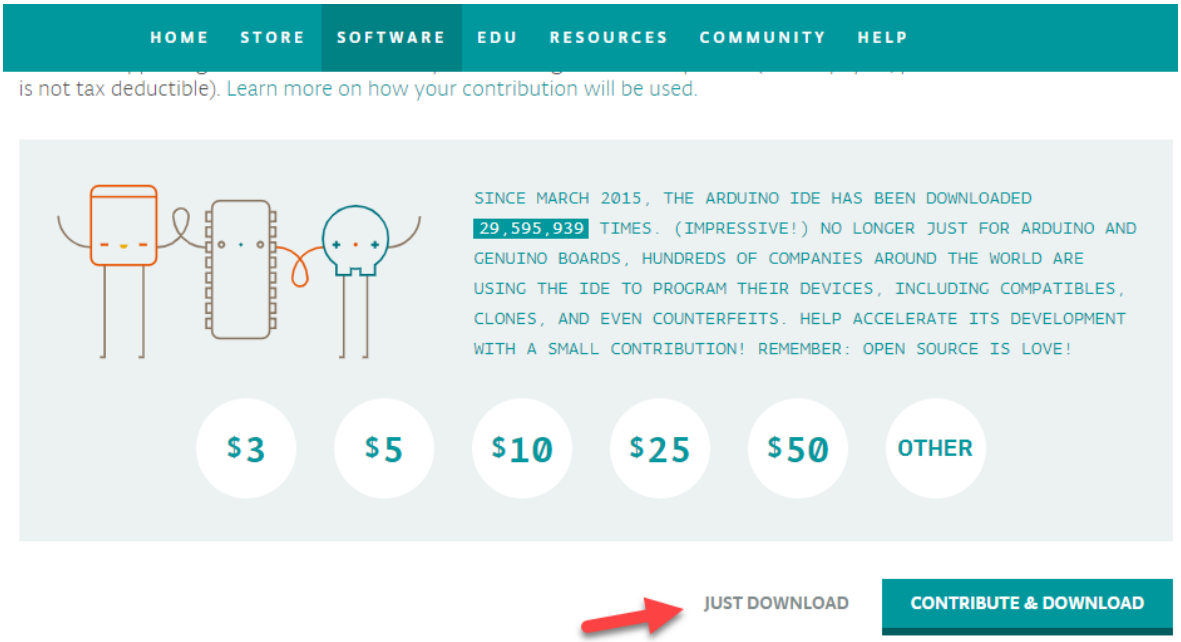
- **Vào trang chủ của Arduino: arduino.cc và nhấn vào mục download**



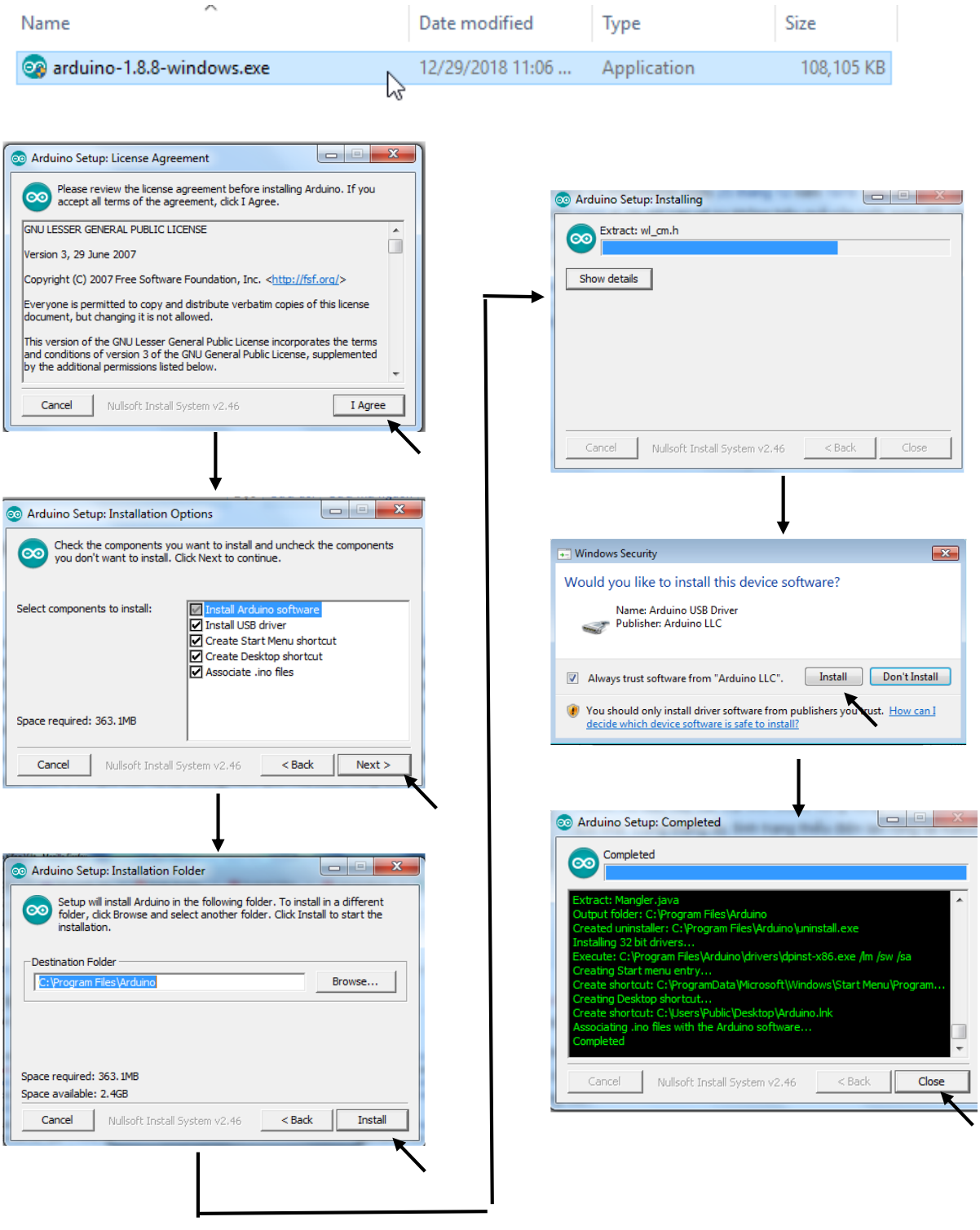
- **Chọn hệ điều hành**, ví dụ hệ điều hành là Windows:

Có 2 tùy chọn cho hệ điều hành Windows là Windows installer và Windows zip file for non admin install. Tùy chọn thứ nhất dùng cho người là admin của máy, tùy chọn thứ 2 cho người không phải admin của máy. Thông thường đa số chọn tùy chọn thứ nhất, tức Windows installer.

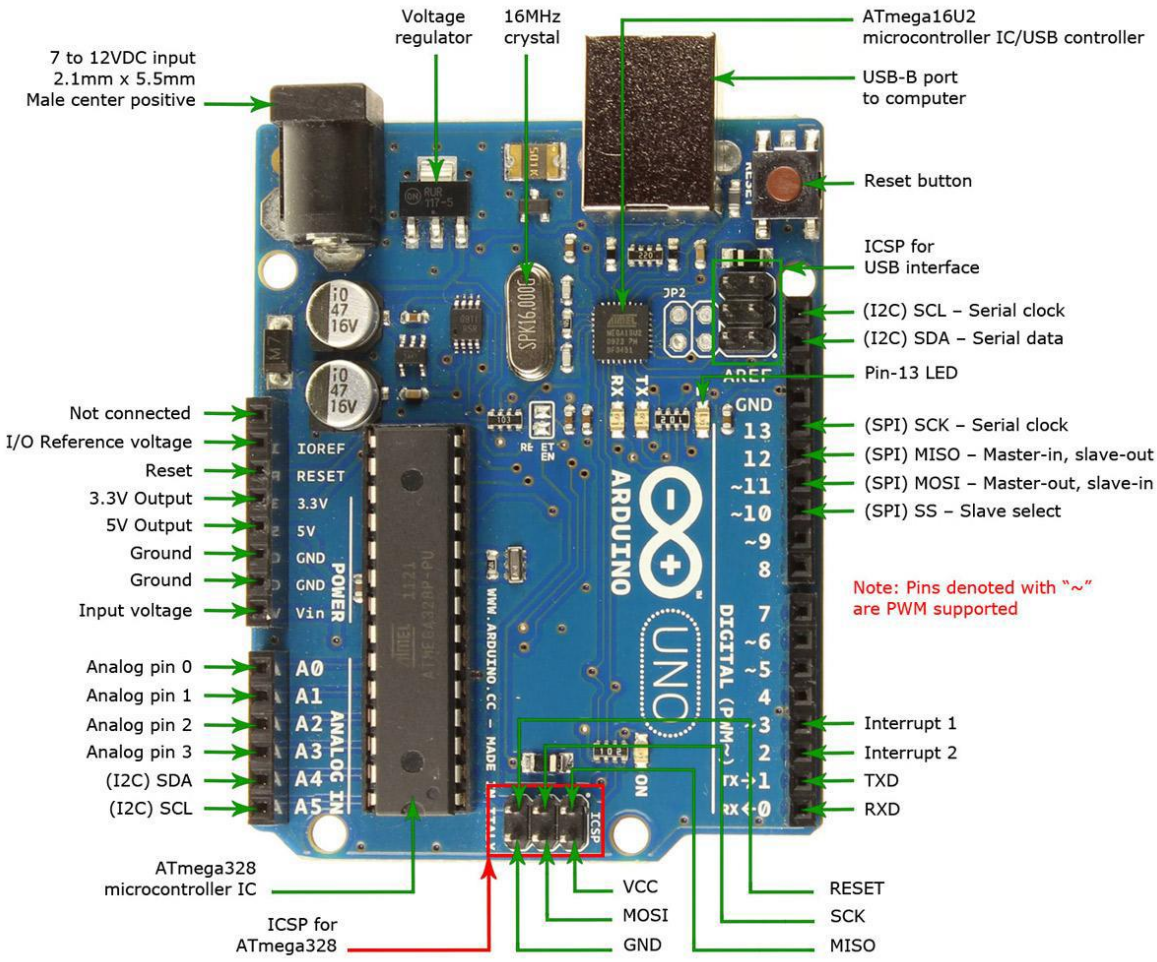
- **Download chương trình về để cài đặt:** chọn JUST DOWNLOAD



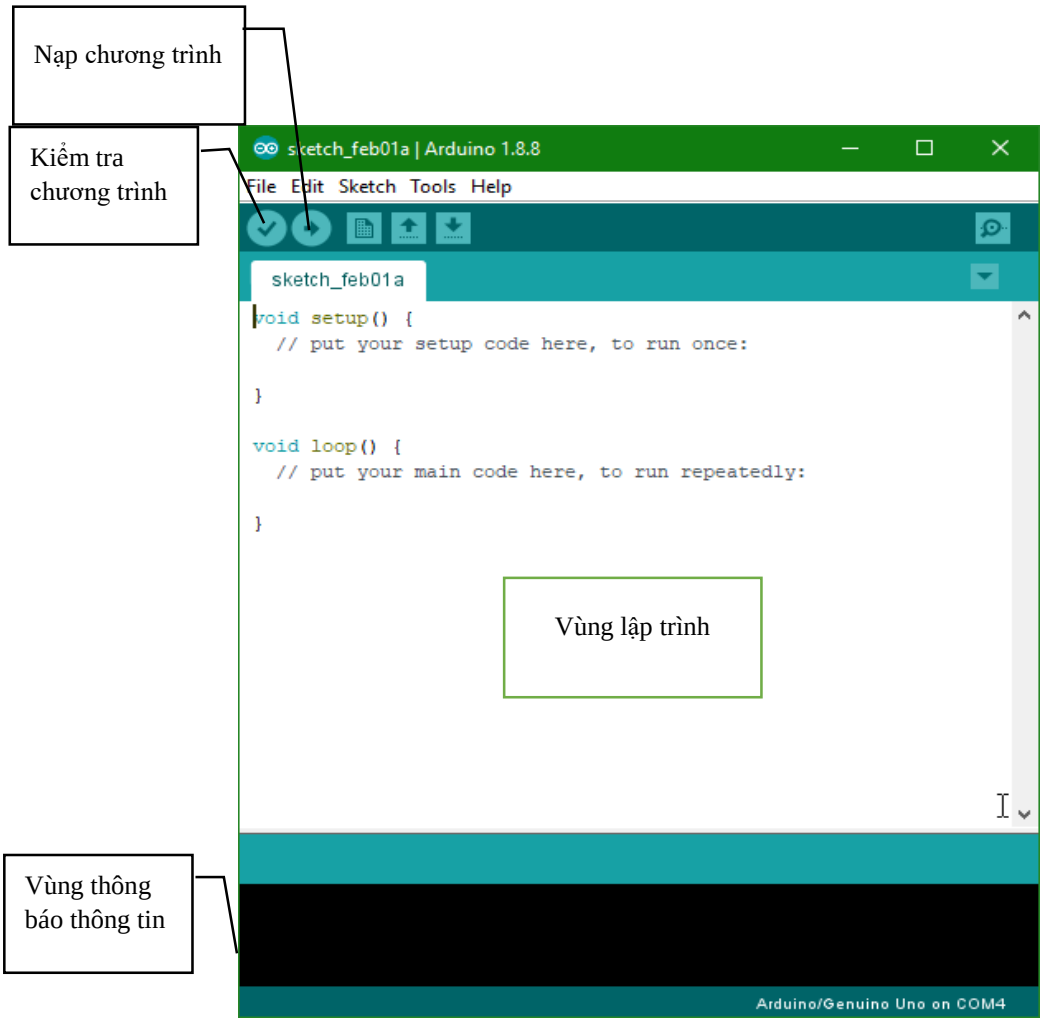
- **Cài đặt:** Để cài đặt, nhấp đúp vào file đó, phần mềm sẽ bắt đầu cài đặt.


Name
Date modified
Type
Size
arduino-1.8.8-windows.exe
12/29/2018 11:06 ...
Application
108,105 KB
Arduino Setup: License Agreement
Please review the license agreement before installing Arduino. If you accept all terms of the agreement, click I Agree.
GNU LESSER GENERAL PUBLIC LICENSE
Version 3, 29 June 2007
Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.
This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.
Cancel
Nullsoft Install System v2.46
I Agree
Arduino Setup: Installation Options
Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.
Select components to install:
Install Arduino software
Install USB driver
Create Start Menu shortcut
Create Desktop shortcut
Associate .ino files
Space required: 363.1MB
Cancel
Nullsoft Install System v2.46
< Back
Next >
Arduino Setup: Installation Folder
Setup will install Arduino in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.
Destination Folder
C:\Program Files\Arduino
Browse...
Space required: 363.1MB
Space available: 2.4GB
Cancel
Nullsoft Install System v2.46
< Back
Install
Arduino Setup: Installing
Extract: wl_cm.h
Show details
Cancel
Nullsoft Install System v2.46
< Back
Close
Windows Security
Would you like to install this device software?
Name: Arduino USB Driver
Publisher: Arduino LLC
Always trust software from "Arduino LLC".
Install
Don't Install
You should only install driver software from publishers you trust. How can I decide which device software is safe to install?
Arduino Setup: Completed
Completed
Extract: Mangler.java
Output folder: C:\Program Files\Arduino
Created uninstaller: C:\Program Files\Arduino\uninstall.exe
Installing 32 bit drivers...
Execute: C:\Program Files\Arduino\drivers\dpinst-x86.exe /lm /sw /sa
Creating Start menu entry...
Create shortcut: C:\ProgramData\Microsoft\Windows\Start Menu\Program...
Creating Desktop shortcut...
Create shortcut: C:\Users\Public\Desktop\Arduino.lnk
Associating .ino files with the Arduino software...
Completed
Cancel
Nullsoft Install System v2.46
< Back
Close

1.2 Mạch Arduino

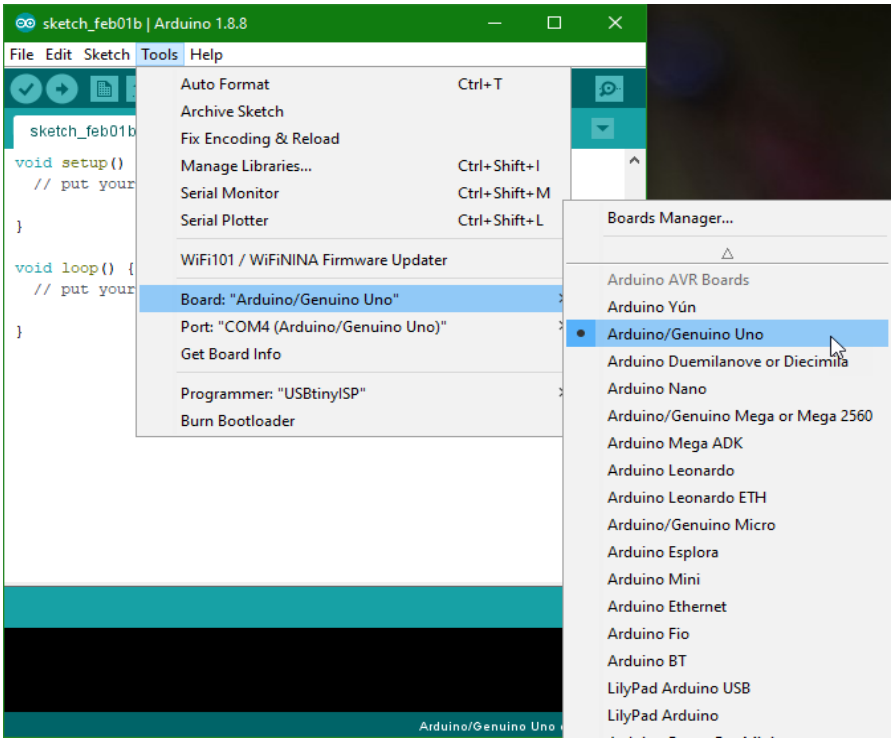


1.3 Giao diện lập trình cho Arduino

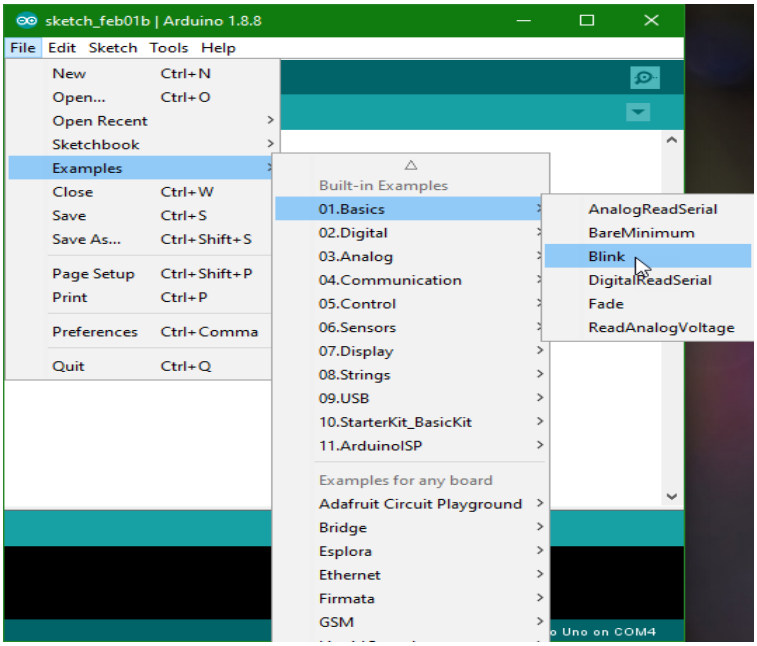


1.4 Nạp và chạy chương trình

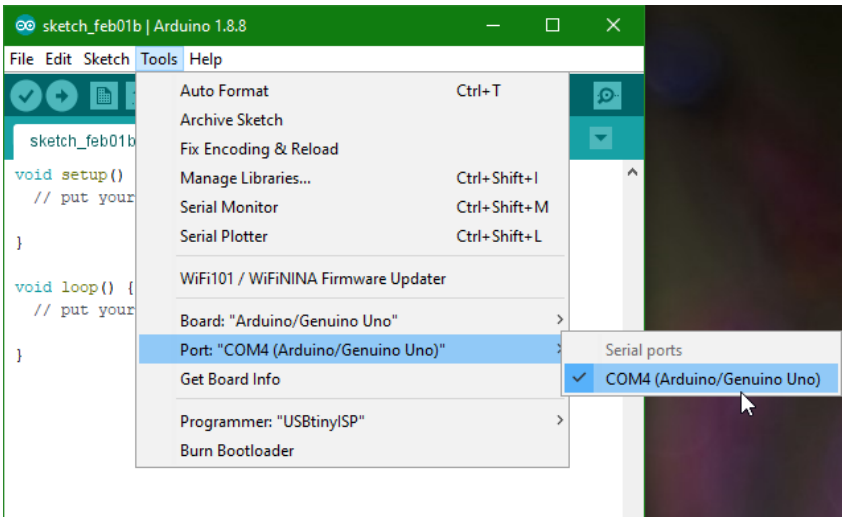
- Chọn loại board cần nạp code



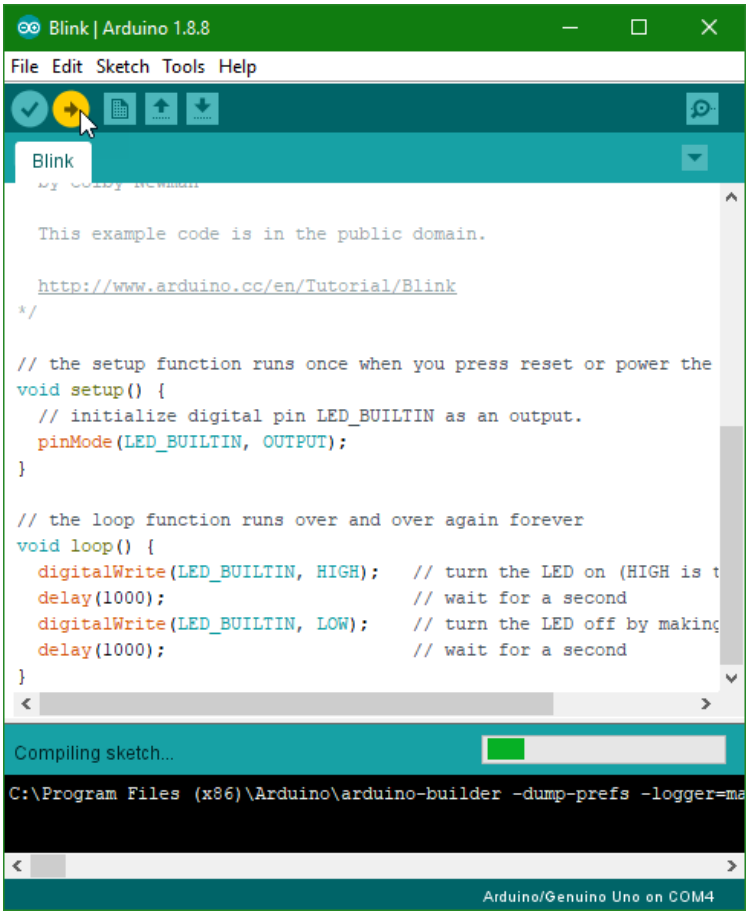
- Load một chương trình mẫu để nạp thử vào board Arduino



- Tiến hành kết nối board vào máy tính. Sau đó chọn đúng cổng COM kết nối

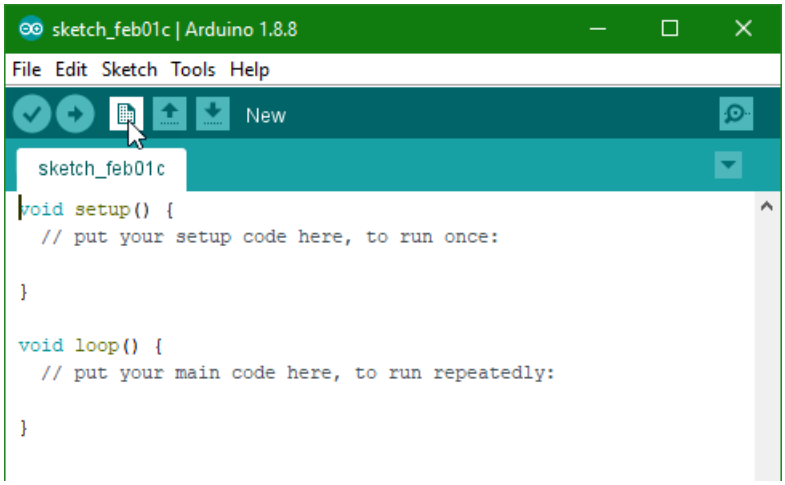


- Nạp và chạy chương trình

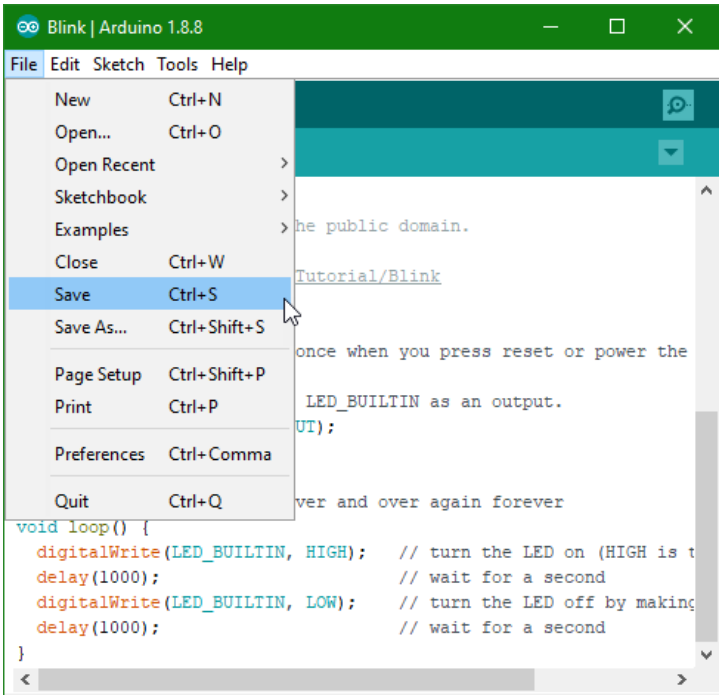


1.5 Tạo và lưu chương trình

- Tạo chương trình mới. File => New

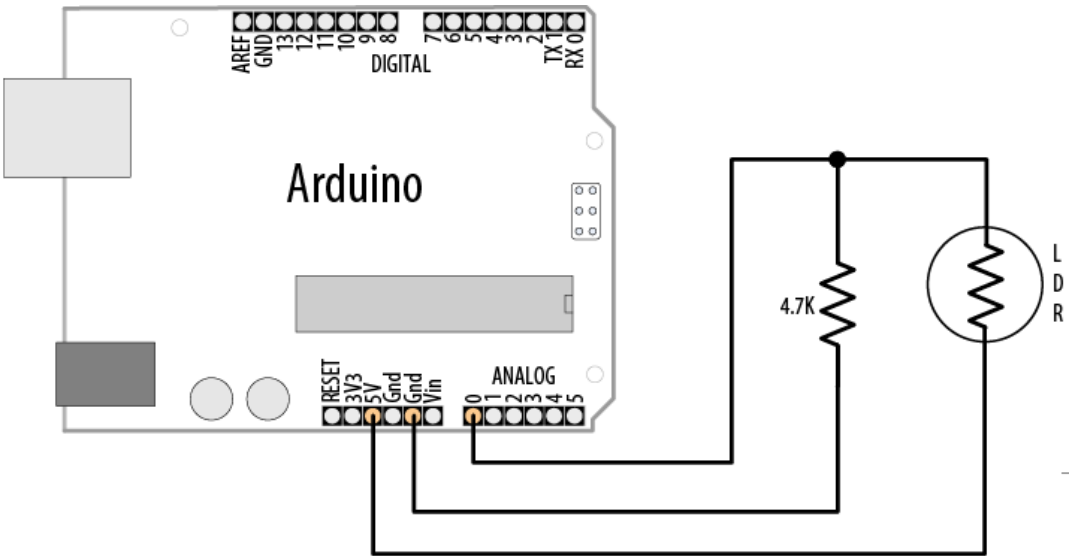


- Sau khi viết code, để lưu chương trình ta làm theo hình sau



- Chọn đường dẫn cần lưu và thực hiện lưu chương trình.

1.6 Sử dụng mạch Arduino



Khảo sát mạch trên, viết chương trình led chớp tắt với thời gian delay phụ thuộc vào độ sáng chiếu vào LDR

Chương trình mẫu:

```
const int ledPin = 13;          // LED connected to digital pin 13
const int sensorPin = 0;        // connect sensor to analog input 0
const int minDuration = 100; // minimum wait between blinks
const int maxDuration = 1000; // maximum wait between blinks
void setup()
{
pinMode(ledPin, OUTPUT);        // enable output on the led pin
Serial.begin(9600);                     // initialize Serial
}
void loop()
{
int rate = analogRead(sensorPin); // read the analog input
```

```
rate = map(rate, 200,800,minDuration, maxDuration); // convert to blink rate
Serial.println(rate); // print rate to serial monitor
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 2: TẠO CHƯƠNG TRÌNH

2.1 Cấu trúc chương trình Arduino

Khai báo

```
void setup() {
  // put your setup code here, to run once:
 // code cài đặt (thiết lập) , chỉ chạy 1 lần
}
void loop() {
  // put your main code here, to run repeatedly:
 // code chính, chạy lập lại
}
```

Vi dụ:

Ví dụ làm cho led nhấp nháy:

```
#define led  13 // khai báo chân led là chân 13
// const int led = 13; //cũng có thể khai báo chân theo kiểu hằng
void setup() {
 pinMode(led, OUTPUT); //Thiết lập chân led (chân 13) là chân ra (OUTPUT)
}
void loop() {
digitalWrite(led, HIGH);     // đặt chân led lên mức cao
delay(1000);  // Tạo trễ thời gian trong 1s (1000ms), có thể thay đổi giá trị này
digitalWrite(led, LOW); // đặt chân led xuống mức thấp
delay(1000);  // Tạo trễ thời gian trong 1s (1000ms), có thể thay đổi giá trị này
}
```

2.2 Hàm

Một hàm là một nhóm các lệnh đi cùng nhau để thực hiện một nhiệm vụ. Trong Arduino mặc định khi tạo mới luôn có sẵn 2 hàm là setup() và loop().

Ví dụ: hàm blink1() cho led chớp tắt 1 lần

```
void blink1()
```

```
{
digitalWrite(13,HIGH); // turn the LED on
delay(500); // wait 500 milliseconds
digitalWrite(13,LOW); // turn the LED off
delay(500); // wait 500 milliseconds
}
```

Kiểu dữ liệu đi trước tên hàm cho biết kiểu trả về (hoặc kiểu không có trả về nếu là void)

Ví dụ:

- Kiểu trả về

```
int sensorPercent(int pin)
{
int percent;
val = analogRead(pin); // read the sensor (ranges from 0 to 1023)
percent = map(val,0,1023,0,100); // percent will range from 0 to 100.
return percent;
}
```

- Kiểu không có trả về

```
void swap()
{
int temp;
temp = x;
x = y;
y = temp;
}
```

2.3 Lệnh điều kiện

Cấu trúc:

```
if()
{
Các câu lệnh
}
else
{
Các câu lệnh
}
```

Ví dụ:

```
/*
Pushbutton sketch
a switch connected to pin 2 lights the LED on pin 13
*/
const int ledPin = 13; // choose the pin for the LED
const int inputPin = 2; // choose the input pin (for a pushbutton)
void setup() {
pinMode(ledPin, OUTPUT); // declare LED pin as output
pinMode(inputPin, INPUT); // declare pushbutton pin as input
}
void loop(){
int val = digitalRead(inputPin); // read input value
if (val == HIGH) // check if the input is HIGH
{
// do this if val is HIGH
digitalWrite(ledPin, HIGH); // turn LED on if switch is pressed
}
else
```

```
{
// else do this if val is not HIGH
digitalWrite(ledPin, LOW); // turn LED off
}
}
```

2.4 Vòng Lập While

Vòng lặp cho phép thực hiện một lệnh hoặc một nhóm lệnh nhiều lần khi điều kiện biểu thức là đúng.

Ví dụ:

```
while(analogRead(sensorPin) > 100)
{
flashLED(); // call a function to turn an LED on and off
}
```

Vòng lặp do ... while tương tự như vòng lặp while, nhưng các khối mã được thực thi trước khi điều kiện được kiểm tra. Sử dụng biểu mẫu này khi bạn phải thực thi mã ít nhất một lần, ngay cả khi biểu thức là sai.

Ví dụ:

```
do
{
flashLED(); // call a function to turn an LED on and off
}
while (analogRead(sensorPin) > 100);
```

Mã trên sẽ nháy đèn LED ít nhất một lần và sẽ nhấp nháy miễn là giá trị đọc từ cảm biến lớn hơn 100. Nếu giá trị không lớn hơn 100, đèn LED sẽ chỉ nhấp nháy một lần.

2.5 Vòng Lập For

Bạn muốn lặp lại một hoặc nhiều câu lệnh một số lần nhất định. Vòng lặp for tương tự như vòng lặp while, nhưng bạn có nhiều quyền kiểm soát hơn đối với các điều kiện bắt đầu và kết thúc.

Ví dụ:

```
/*
ForLoop sketch
demonstrates for loop
*/
void setup() {
Serial.begin(9600);}
void loop(){
Serial.println("for(int i=0; i < 4; i++)");
for(int i=0; i < 4; i++)
{
Serial.println(i);
}
}
```

Kết quả:

```
for(int i=0; i < 4; i++)
0
1
2
3
```

Một vòng lặp for bao gồm ba phần: khởi tạo, kiểm tra có điều kiện và lặp lại (một câu lệnh được thực thi ở cuối mỗi lần đi qua vòng lặp). Mỗi phần được phân tách bằng dấu chấm phẩy. Như ví dụ trên thì int i = 0 là khởi tạo biến i = 0; i<4 kiểm tra biến để xem nếu nó nhỏ hơn 4; và i ++ tăng i lên 1 đơn vị.

2.6 So sánh logic

Symbol	Function
&&	Logical And
||	Logical Or
!	Not

Toán tử logic trả về giá trị true hoặc false dựa trên mối quan hệ logic.

Toán tử And logic && sẽ trả về true nếu cả hai toán hạng của nó là true và ngược lại là false. Toán tử Or logic || sẽ trả về true nếu một trong hai toán hạng của nó là true và false nếu cả hai toán hạng đều false. Toán tử Not ! chỉ có một toán hạng, có giá trị đảo ngược lại trạng thái logic của toán hạng đó, nghĩa là false nếu toán hạng của nó là true và true nếu toán hạng của nó là false.

2.7 Các toán tử xử lý bit

Symbol	Function	Comment	Example
&	Bitwise And	Sets bits in each place to 1 if both bits are 1; otherwise, bits are set to 0.	3 & 1 equals 1 (11 & 01 equals 01)
|	Bitwise Or	Sets bits in each place to 1 if either bit is 1.	3 | 1 equals 3 (11 | 01 equals 11)
^	Bitwise Exclusive Or	Sets bits in each place to 1 only if one of the two bits is 1.	3 ^ 1 equals 2 (11 ^ 01 equals 10)
~	Bitwise Negation	Inverts the value of each bit. The result depends on the number of bits in the data type.	~1 equals 254 (~00000001 equals 11111110)

Bảng chân trị của các phép toán logic:

AND logic

Bit 1	Bit 2	Bit 1 and Bit 2
0	0	0
0	1	0
1	0	0
1	1	1

OR logic

Bit 1	Bit 2	Bit 1 or Bit 2
0	0	0
0	1	1
1	0	1
1	1	1

XOR logic

Bit 1	Bit 2	Bit 1 ^ Bit 2
0	0	0
0	1	1
1	0	1
1	1	0

CHƯƠNG 3: CÁC PHÉP TOÁN

3.1 Cộng, trừ, nhân, chia

```
int myValue;
myValue = 1 + 2; // addition
myValue = 3 - 2; // subtraction
myValue = 3 * 2; // multiplication
myValue = 3 / 2; // division (the result is 1)
```

3.2 Tăng, giảm

```
int myValue = 0;
myValue = myvalue + 1; // this adds one to the variable myValue
myValue += 1; // this does the same as the above
myValue = myvalue - 1; // this subtracts one from the variable myValue
myValue -= 1; // this does the same as the above
myValue = myvalue + 5; // this adds five to the variable myValue
myValue += 5; // this does the same as the above
```

3.3 Đặt và đọc bit

bitSet(x, bitPosition) đặt lên 1 tại vị trí bitPosition của biến x

bitClear(x, bitPosition) đặt xuống 0 tại vị trí bitPosition của biến x

bitRead(x, bitPosition) trả về 0 hoặc 1 tại vị trí bitPosition của biến x

bitWrite(x, bitPosition, value) đặt value (0 hoặc 1) tại vị trí bitPosition của biến x

bit(bitPosition) trả về giá trị của vị trí bit đã cho: bit(0) là 1, bit(1) là 2, bit(2) là 4, …

3.4 Dịch chuyển bit

Sử dụng các toán tử << (dịch bit trái) và >> (dịch bit phải) để dịch chuyển các bit của một giá trị.

Ví dụ:

```
int x = 6;
int result = x << 1; // 6 shifted left 1 is 12
Serial.println(result);
```

```
int result = x >> 2; // 12 shifted right 2 is 3;
Serial.println(result);
```

CHƯƠNG 4: GIAO TIẾP NỐI TIẾP

Truyền thông nối tiếp cung cấp một cách dễ dàng và linh hoạt để bo mạch Arduino của bạn tương tác với máy tính và các thiết bị khác. Chương này giải thích cách gửi và nhận thông tin bằng khả năng này.

4.1 Gửi thông tin từ Arduino đến máy tính

```
/*
* SerialOutput sketch
* Print numbers to the serial port
*/
void setup()
{
Serial.begin(9600); // send and receive at 9600 baud
}
int number = 0;
void loop()
{
Serial.print("The number is ");
Serial.println(number); // print the number
delay(500); // delay half second between numbers
number++; // to the next number
}
```

Nhấp vào biểu tượng Serial Monitor trong IDE và bạn sẽ thấy đầu ra được hiển thị như sau:

```
The number is 0
The number is 1
The number is 2
```

❖ Gửi văn bản định dạng và dữ liệu số từ Arduino:

Bạn có thể in dữ liệu lên cổng nối tiếp ở nhiều định dạng khác nhau

```
/*
```

```
* SerialFormatting
* Print values in various formats to the serial port
*/
char chrValue = 65; // these are the starting values to print
int intValue = 65;
float floatValue = 65.0;
void setup()
{
Serial.begin(9600);
}
void loop()
{
Serial.println("chrValue: ");
Serial.println(chrValue);
Serial.println(chrValue,BYTE);
Serial.println(chrValue,DEC);
Serial.println("intValue: ");
Serial.println(intValue);
Serial.println(intValue,BYTE);
Serial.println(intValue,DEC);
Serial.println(intValue,HEX);
Serial.println(intValue,OCT);
Serial.println(intValue,BIN);
Serial.println("floatValue: ");
Serial.println(floatValue);
delay(1000); // delay a second between numbers
chrValue++; // to the next value
intValue++;
}
```

The output (condensed here onto a few lines) is as follows:

chrValue: A A 65

intValue: 65 A 65 41 101 1000001

floatValue: 65.00

chrValue: B B 66

intValue: 66 B 66 42 102 1000010

Output formats using Serial.print

Data type	Print (val)	Print (val,DEC)	Print (val,BYTE)	Print (val,HEX)	Print (val,OCT)	Print (val,BIN)
char	A	65	A	41	101	1000001
int	65	65	A	41	101	1000001
long	Format of long is the same as int					
float	65.00	Formatting not supported for floating-point values				
double	65.00	double is the same as float				

In một chuỗi văn bản rất đơn giản: Serial.print ("hello world"); gửi chuỗi văn bản "hello world" đến một thiết bị ở đầu kia của cổng nối tiếp. Nếu bạn muốn đầu ra của mình in một dòng mới, hãy sử dụng Serial.println () thay vì Serial.print ().

4.2 Gửi thông tin từ máy tính đến Arduino

Bạn muốn nhận dữ liệu trên Arduino từ máy tính hoặc thiết bị nối tiếp khác; cho ví dụ, để Arduino thực thi với các lệnh hoặc dữ liệu được gửi từ máy tính của bạn. Nó dễ dàng nhận được các giá trị 8 bit (char và byte), vì các hàm Nối tiếp sử dụng các giá trị 8- bit. Ví dụ dưới đây nhận được một chữ số (các ký tự đơn từ 0 đến 9) và nhấp nháy đèn LED trên chân 13 theo tỷ lệ với giá trị chữ số nhận được:

```
/*
* SerialReceive sketch
* Blink the LED at a rate proportional to the received digit value
*/
```

```
const int ledPin = 13; // pin the LED is connected to
int blinkRate=0; // blink rate stored in this variable
void setup()
{
Serial.begin(9600); // Initialize serial port to send and receive at 9600 baud
pinMode(ledPin, OUTPUT); // set this pin as output
}
void loop()
{
if ( Serial.available()) // Check to see if at least one character is available
{
char ch = Serial.read();
if(ch >= '0' && ch <= '9') // is this an ascii digit between 0 and 9?
{
blinkRate = (ch - '0'); // ASCII value converted to numeric value
blinkRate = blinkRate * 100; // actual blinkrate is 100 mS times received digit
}
}
blink();
}
// blink the LED with the on and off times determined by blinkRate
void blink()
{
digitalWrite(ledPin,HIGH);
delay(blinkRate); // delay depends on blinkrate value
digitalWrite(ledPin,LOW);
delay(blinkRate);
}
```

Chuyển đổi các ký tự ASCII nhận được thành các giá trị số :

blinkRate = (ch - '0'); // ASCII value converted to numeric value

Điều này được thực hiện bằng cách trừ 48, vì 48 là giá trị ASCII của chữ số 0.

Ví dụ: nếu ch đại diện cho ký tự 1, giá trị ASCII của nó là 49. Biểu thức 49- '0' là giống như 49-48. Điều này bằng 1, là giá trị số của ký tự 1. Nói cách khác, biểu thức (ch - '0') giống với (ch - 48); cái này chuyển đổi giá trị ASCII của biến ch thành giá trị số.

Một cách tiếp cận khác để chuyển đổi chuỗi văn bản đại diện cho số là sử dụng hàm chuyển đổi ngôn ngữ C được gọi là atoi (cho biến int) hoặc atol (cho biến long). Các hàm này chuyển đổi một chuỗi thành số nguyên hoặc số nguyên dài. Để sử dụng chúng, bạn phải nhận và lưu trữ toàn bộ chuỗi trong một mảng ký tự trước khi bạn có thể gọi hàm chuyển đổi.

```
const int MaxChars = 5; // an int string contains up to 5 digits and
// is terminated by a 0 to indicate end of string
char strValue[MaxChars+1]; // must be big enough for digits and terminating null
int index = 0; // the index into the array storing the received digits
void loop()
{
if( Serial.available())
{
char ch = Serial.read();
if(index < MaxChars && ch >= '0' && ch <= '9'){
strValue[index++] = ch; // add the ASCII character to the string;
}
else
{
// here when buffer full or on the first non digit
strValue[index] = 0; // terminate the string with a 0
blinkRate = atoi(strValue); // use atoi to convert the string to an int
index = 0;
}
```

}

blink();

}

Với:

strValue là một chuỗi số được xây dựng từ các ký tự nhận được từ cổng nối tiếp.

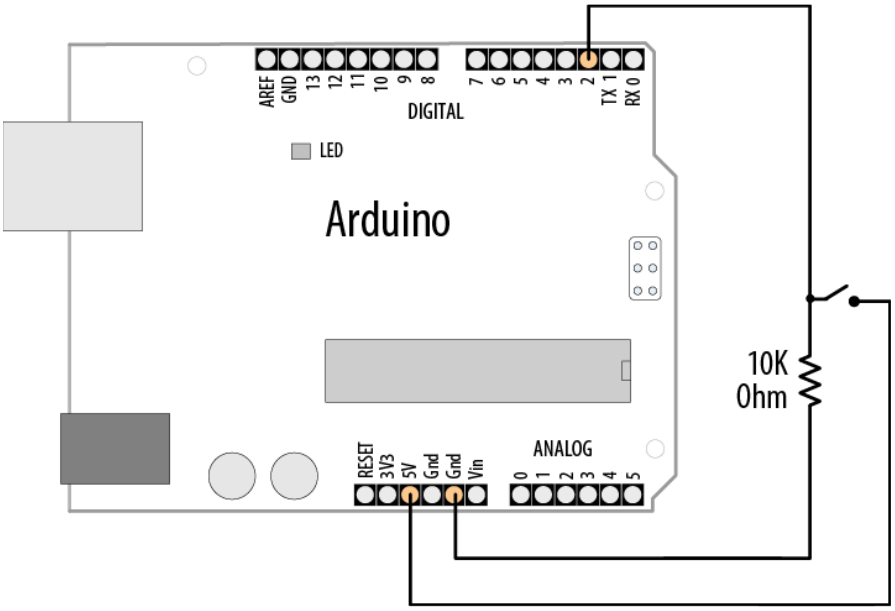
atoi (viết tắt của ASCII thành số nguyên) là một hàm chuyển đổi một chuỗi ký tự thành một số nguyên (atol chuyển đổi thành một số nguyên dài).

CHƯƠNG 5: NGÕ VÀO TÍN HIỆU SỐ VÀ TƯƠNG TỰ

5.1 Sử dụng công tắc

Sử dụng digitalRead để xác định trạng thái của một công tắc được kết nối với chân kỹ thuật số Arduino được đặt làm đầu vào.

Cho mạch điện như hình:




```
/*
Pushbutton sketch
a switch connected to pin 2 lights the LED on pin 13
*/
const int ledPin = 13; // choose the pin for the LED
const int inputPin = 2; // choose the input pin (for a pushbutton)
void setup() {
pinMode(ledPin, OUTPUT); // declare LED as output
pinMode(inputPin, INPUT); // declare pushbutton as input
}
void loop(){
int val = digitalRead(inputPin); // read input value
if (val == HIGH) // check if the input is HIGH
```

```
{
digitalWrite(ledPin, HIGH); // turn LED on if switch is pressed
}
else
{
digitalWrite(ledPin, LOW); // turn LED off
}
}
```

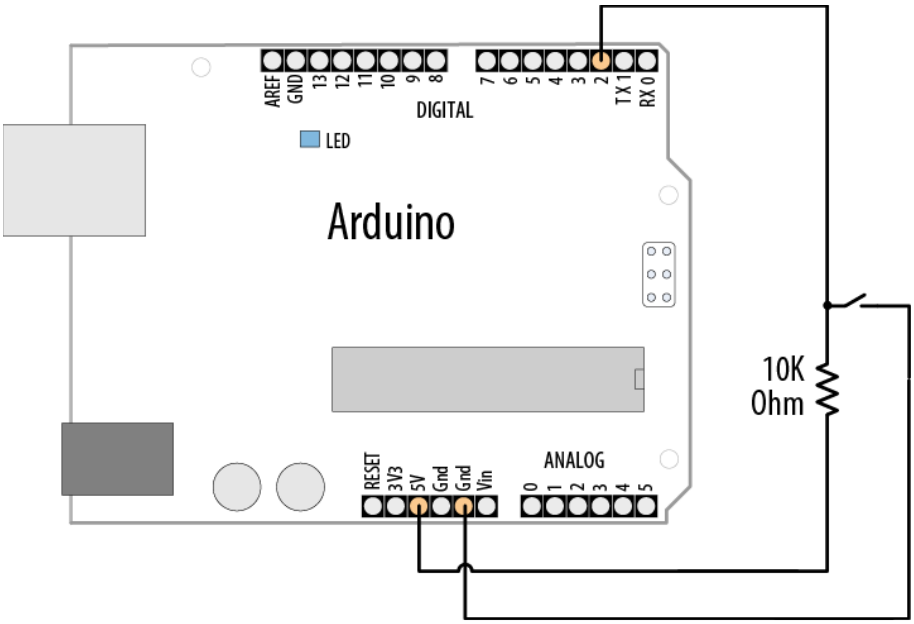
Hoặc cũng có thể viết theo cách ngắn hơn:

```
void loop()
{
digitalWrite(ledPin, digitalRead(inputPin)); // turn LED ON if input pin is HIGH, else
                                             //turn OFF
}
```

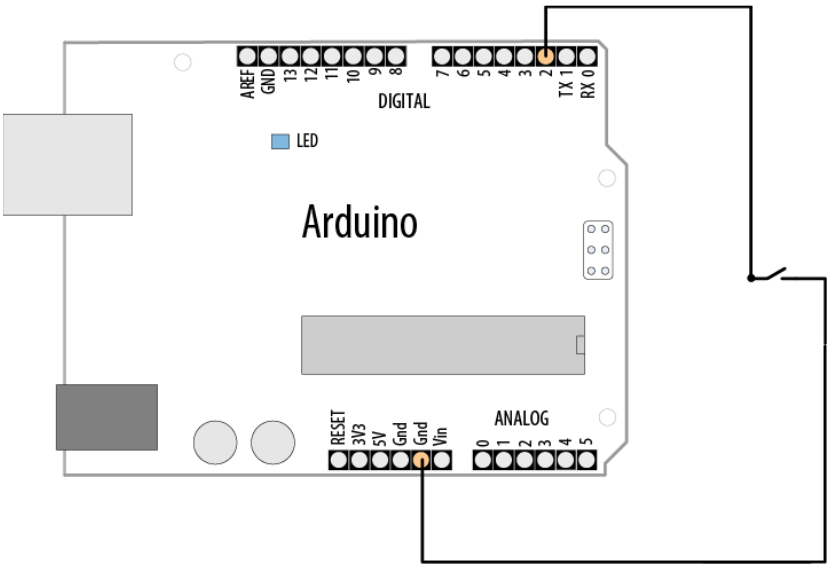
Đoạn code này không lưu trữ trạng thái nút thành một biến. Thay vào đó, nó đặt hoặc tắt đèn LED trực tiếp từ giá trị thu được từ digitalRead.

Ngược lại mạch trên, nếu công tắc mắc tích cực mức thấp thì code sẽ thay đổi như sau:

```
void loop()
{
int val = digitalRead(inputPin); // read input value
if (val == HIGH) // check if the input is HIGH
{
digitalWrite(ledPin, LOW); // turn LED OFF
}
else
{
digitalWrite(ledPin, HIGH); // turn LED ON
}
}
```



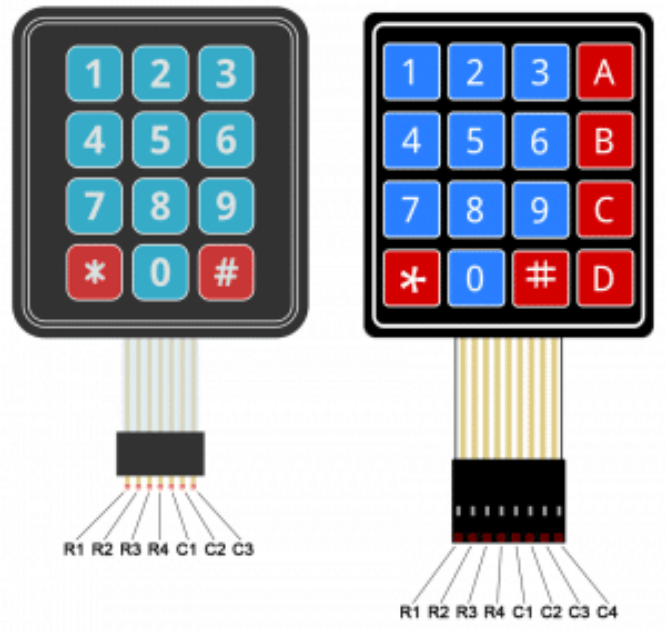
❖ Sử dụng công tắc không có điện trở bên ngoài:

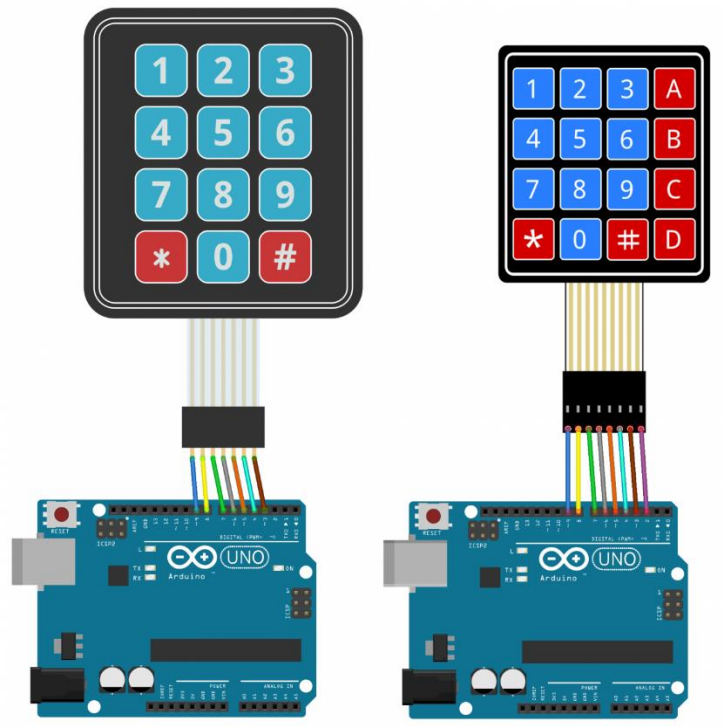



```
/*
Pullup sketch
a switch connected to pin 2 lights the LED on pin 13
*/
const int ledPin = 13; // output pin for the LED
const int inputPin = 2; // input pin for the switch
void setup() {
```

```
pinMode(ledPin, OUTPUT);
pinMode(inputPin, INPUT);
digitalWrite(inputPin,HIGH); // turn on internal pull-up on the inputPin
}
void loop(){
int val = digitalRead(inputPin); // read input value
if (val == HIGH) // check if the input is HIGH
{
digitalWrite(ledPin, HIGH); // turn LED OFF
}
else
{
digitalWrite(ledPin, LOW); // turn LED ON
}
}
```

5.2 Đọc bàn phím






```
/*
Keypad sketch
prints the key pressed on a keypad to the serial port
*/
const int numRows = 4; // number of rows in the keypad
const int numCols = 3; // number of columns
const int debounceTime = 20; // number of milliseconds for switch to be stable
// keymap defines the character returned when the corresponding key is pressed
const char keymap[numRows][numCols] = {
{ '1', '2', '3' } ,
{ '4', '5', '6' } ,
{ '7', '8', '9' } ,
{ '*', '0', '#' }
};
// this array determines the pins used for rows and columns
const int rowPins[numRows] = { 9, 8, 7, 6 }; // chân 7,6,5,4 keypad
```

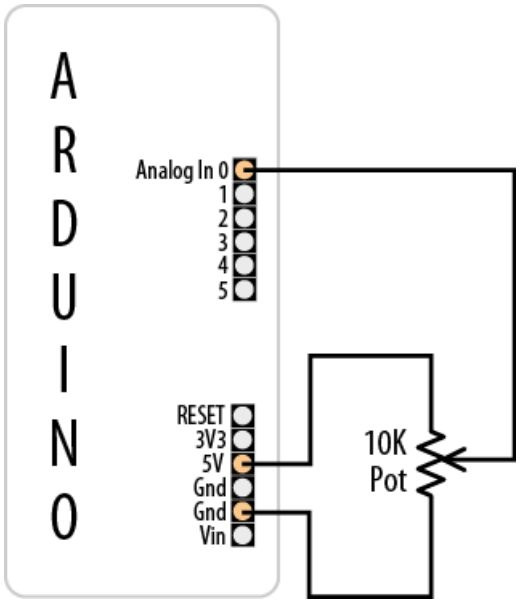
```
const int colPins[numCols] = { 5, 4, 3 }; // chân 3,2,1 keypad
void setup()
{
Serial.begin(9600);
for (int row = 0; row < numRows; row++)
{
pinMode(rowPins[row],INPUT); // Set row pins as input
digitalWrite(rowPins[row],HIGH); // turn on Pull-ups
}
for (int column = 0; column < numCols; column++)
{
pinMode(colPins[column],OUTPUT); // Set column pins as outputs for writing
digitalWrite(colPins[column],HIGH); // Make all columns inactive
}
}
void loop()
{
char key = getKey();
if( key != 0) { // if the character is not 0 then it's a valid key press
Serial.print("Got key ");
Serial.println(key);
}
}
// returns with the key pressed, or 0 if no key is pressed
char getKey()
{
char key = 0; // 0 indicates no key pressed
for(int column = 0; column < numCols; column++)
{
```

```
digitalWrite(colPins[column],LOW); // Activate the current column.
for(int row = 0; row < numRows; row++) // Scan all rows for a key press.
{
if(digitalRead(rowPins[row]) == LOW) // Is a key pressed?
{
delay(debounceTime); // debounce
while(digitalRead(rowPins[row]) == LOW)
; // wait for key to be released
key = keymap[row][column]; // Remember which key was pressed.
}
}
digitalWrite(colPins[column],HIGH); // De-activate the current column.
}
return key; // returns the key pressed or 0 if none
}
```

Bàn phím ma trận thường bao gồm các công tắc Thông thường mở kết nối một hàng với một cột khi nhấn. (Công tắc thường mở chỉ tạo kết nối điện khi được ấn.) Hình trên cho thấy cách các dây dẫn bên trong kết nối các hàng và cột của nút với đầu nối bàn phím. Mỗi trong bốn hàng được kết nối với một pin đầu vào và mỗi cột được kết nối với một chân đầu ra.

Hàm getkey tuần tự đặt mã pin cho từng cột THẤP và sau đó kiểm tra xem liệu có bất kỳ chân hàng nào là THẤP không. Nếu chúng ở mức THẤP, điều này cho biết rằng công tắc cho hàng và cột đó đã bị đóng. Delay tạo độ trễ được sử dụng để đảm bảo rằng công tắc không bị nảy (xem Công thức 5.3); code chờ cho công tắc được nhả và ký tự tương đương với công tắc được nhấn trong mảng sơ đồ bàn phím và được trả về từ hàm. Trả về 0 nếu không có công tắc nào được nhấn.

5.3 Đọc tín hiệu tương tự



Chương trình này đọc điện áp trên một chân analog và nháy đèn LED theo tỷ lệ tương ứng với giá trị được trả về từ hàm analogRead. Điện áp được điều chỉnh bằng một biến trở được kết nối như hình.

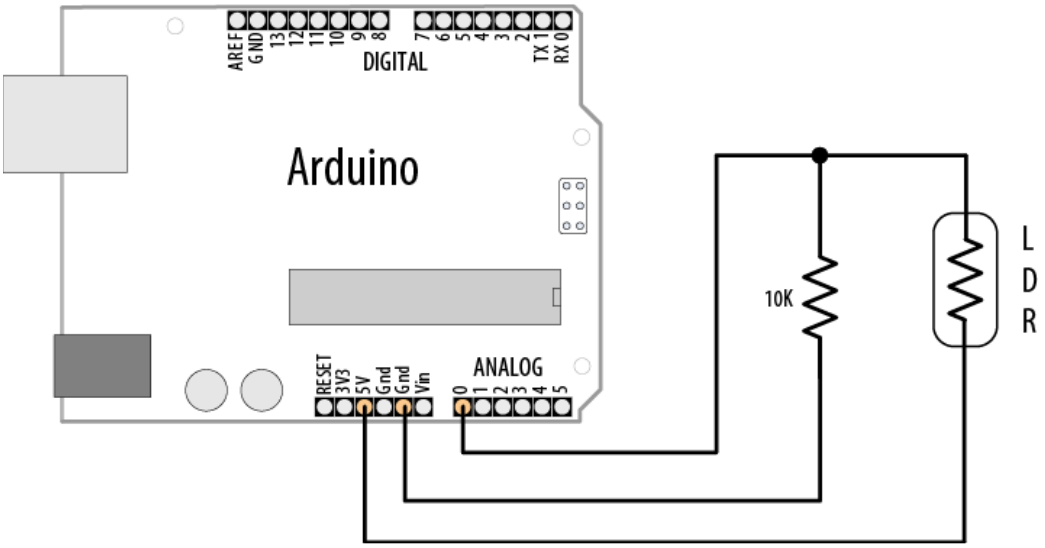
```
/*
Pot sketch
blink an LED at a rate set by the position of a potentiometer
*/
const int potPin = 0; // select the input pin for the potentiometer
const int ledPin = 13; // select the pin for the LED
int val = 0; // variable to store the value coming from the sensor
void setup()
{
pinMode(ledPin, OUTPUT); // declare the ledPin as an OUTPUT
}
void loop() {
val = analogRead(potPin); // read the voltage on the pot
digitalWrite(ledPin, HIGH); // turn the ledPin on
```

```
delay(val); // blink rate set by pot value (in milliseconds)
digitalWrite(ledPin, LOW); // turn the ledPin off
delay(val); // turn led off for same period as it was turned on
}
```

Điện áp được đo bằng cách sử dụng analogRead, cung cấp giá trị tỷ lệ với điện áp thực tế trên chân analog. Giá trị sẽ là 0 khi có 0v trên pin và 1.023 khi có 5v.

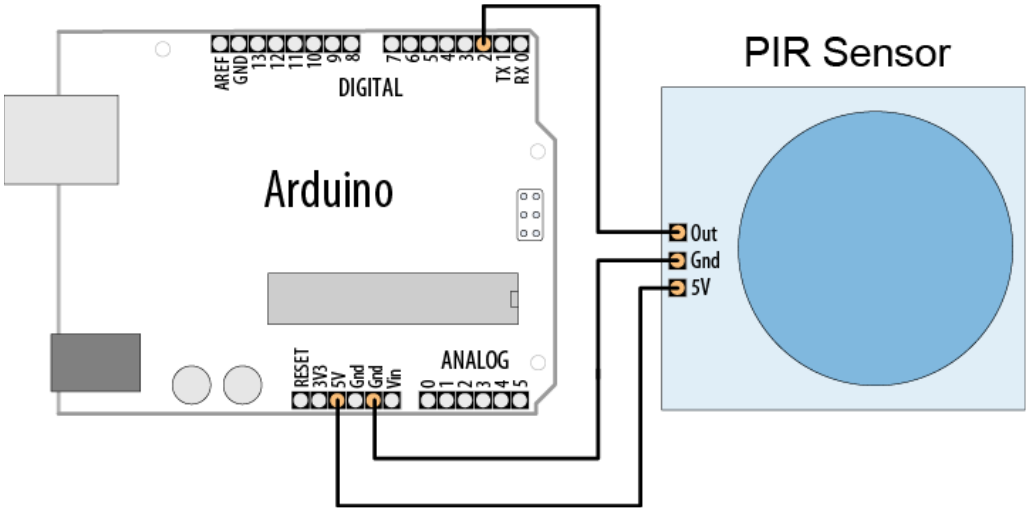
CHƯƠNG 6: NHẬN TÍN HIỆU TỪ CẢM BIẾN

6.1 Cảm biến ánh sáng



Cách dễ nhất để phát hiện các mức ánh sáng là sử dụng quang trở (LDR). Điều này thay đổi điện trở khi thay đổi mức ánh sáng và khi được kết nối trong mạch như trong hình trên, nó tạo ra sự thay đổi điện áp mà các chân đầu vào tương tự Arduino có thể cảm nhận được. Phần lập trình đơn thuần là đọc tín hiệu tương tự ở chân analog.

6.2 Cảm biến chuyển động




```
/*
PIR sketch
a Passive Infrared motion sensor connected to pin 2
```

```
lights the LED on pin 13
*/
const int ledPin = 13; // choose the pin for the LED
const int inputPin = 2; // choose the input pin (for the PIR sensor)
void setup() {
pinMode(ledPin, OUTPUT); // declare LED as output
pinMode(inputPin, INPUT); // declare pushbutton as input
}
void loop(){
int val = digitalRead(inputPin); // read input value
if (val == HIGH) // check if the input is HIGH
{
digitalWrite(ledPin, HIGH); // turn LED on if motion detected
delay(500);
digitalWrite(ledPin, LOW); // turn LED off
}
}
```

6.3 Cảm biến khoảng cách

Tốc độ của âm thanh trong không khí là 340 m/s (hằng số vật lý), tương đương với 29,412 microSeconds/cm (106 / (340*100)). Khi đã tính được thời gian, ta sẽ chia cho 29,412 để nhận được khoảng cách.

Vcc	5V
Trig	Một chân Digital output
Echo	Một chân Digital input
GND	GND

Sơ đồ nối chân giữa HC-SR04 và Arduino

```
#define ECHOPIN 12                    // Pin to receive echo pulse
#define TRIGPIN 11                    // Pin to send trigger pulse
int duration;    // biến đo thời gian
int distance;          // biến lưu khoảng cách
void setup()
{
  Serial.begin(9600);
  pinMode(ECHOPIN, INPUT);
  pinMode(TRIGPIN, OUTPUT);
}
void loop()
{
  digitalWrite(TRIGPIN, LOW);              // Set the trigger pin to low for 2uS
  delayMicroseconds(2);
  digitalWrite(TRIGPIN, HIGH);             // Send a 10uS high to trigger ranging
  delayMicroseconds(10);
  digitalWrite(TRIGPIN, LOW);              // Send pin low again
  duration = pulseIn(ECHOPIN, HIGH);       // Read in times pulse
  distance= duration /58;                  // Calculate distance from time of pulse
  Serial.println(distance);
  delay(50);                               // Wait 50mS before next ranging
}
```

Hàm pulseIn() được dùng để đo độ rộng của xung, duration sẽ bằng độ dài xung HIGH ở chân echo (tính theo micro giây).

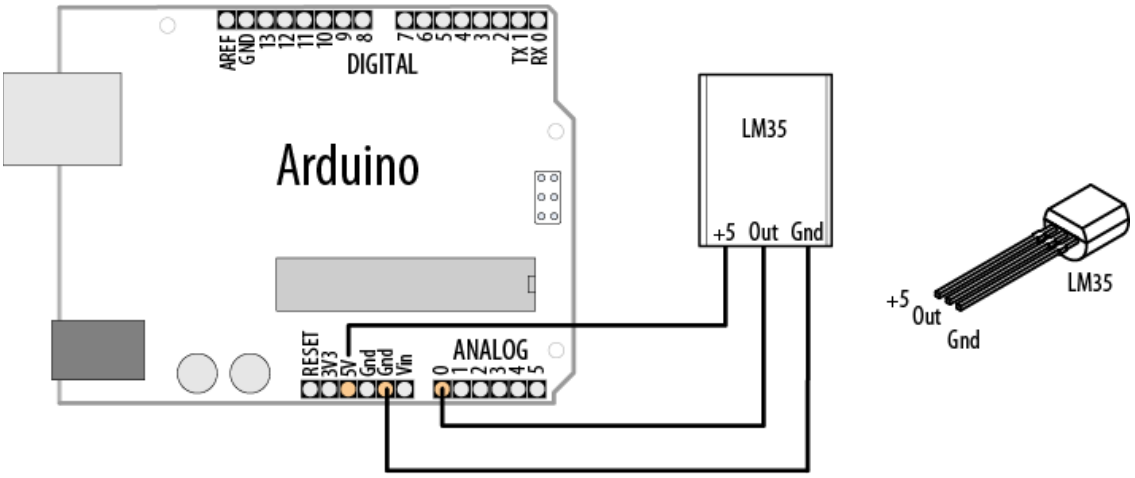
distance = int(duration/2/29.412);

hoặc distance = duration /58;// lấy gần đúng

Thời gian sóng truyền từ cảm biến đến vật sẽ bằng duration/2, sau đó ta chia tiếp cho 29,412 để tính khoảng cách.

6.4 Cảm biến nhiệt độ

Cảm biến nhiệt độ LM35 tạo ra một điện áp tương tự tỷ lệ thuận với nhiệt độ với công suất 1 millivolt mỗi 0,1 ° C (10 mV mỗi độ).

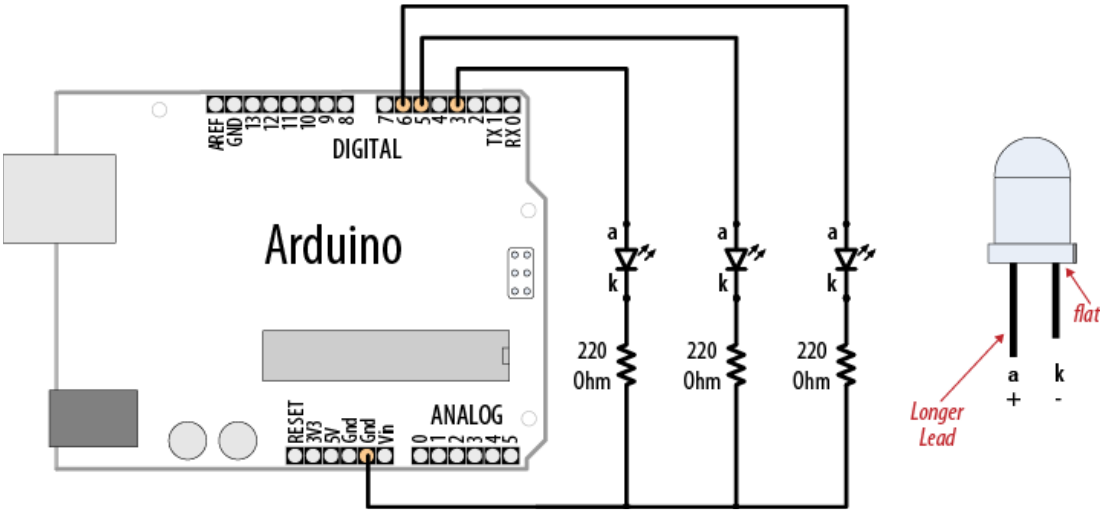



```
/*
lm35 sketch
prints the temperature to the Serial Monitor
*/
const int inPin = 0; // analog pin
void setup()
{
Serial.begin(9600);
}
void loop()
{
int value = analogRead(inPin);
Serial.print(value); Serial.print(" > ");
float millivolts = (value / 1024.0) * 5000;
float celsius = millivolts / 10; // sensor output is 10mV per degree Celsius
Serial.print(celsius);
Serial.print(" degrees Celsius, ");
Serial.print( (celsius * 9)/ 5 + 32 ); // converts to fahrenheit
```

```
Serial.println(" degrees Fahrenheit");
delay(1000); // wait for one second
}
```

CHƯƠNG 7: HIỂN THỊ

7.1 Điều khiển Led sáng tắt




```
/*
LEDs sketch
Blink three LEDs each connected to a different digital pin
*/
const int firstLedPin = 3; // choose the pin for each of the LEDs
const int secondLedPin = 5;
const int thirdLedPin = 6;
void setup()
{
pinMode(firstLedPin, OUTPUT); // declare LED pins as output
pinMode(secondLedPin, OUTPUT); // declare LED pins as output
pinMode(thirdLedPin, OUTPUT); // declare LED pins as output
}
void loop()
{
// flash each of the LEDs for 1000 milliseconds (1 second)
blinkLED(firstLedPin, 1000);
blinkLED(secondLedPin, 1000);
```

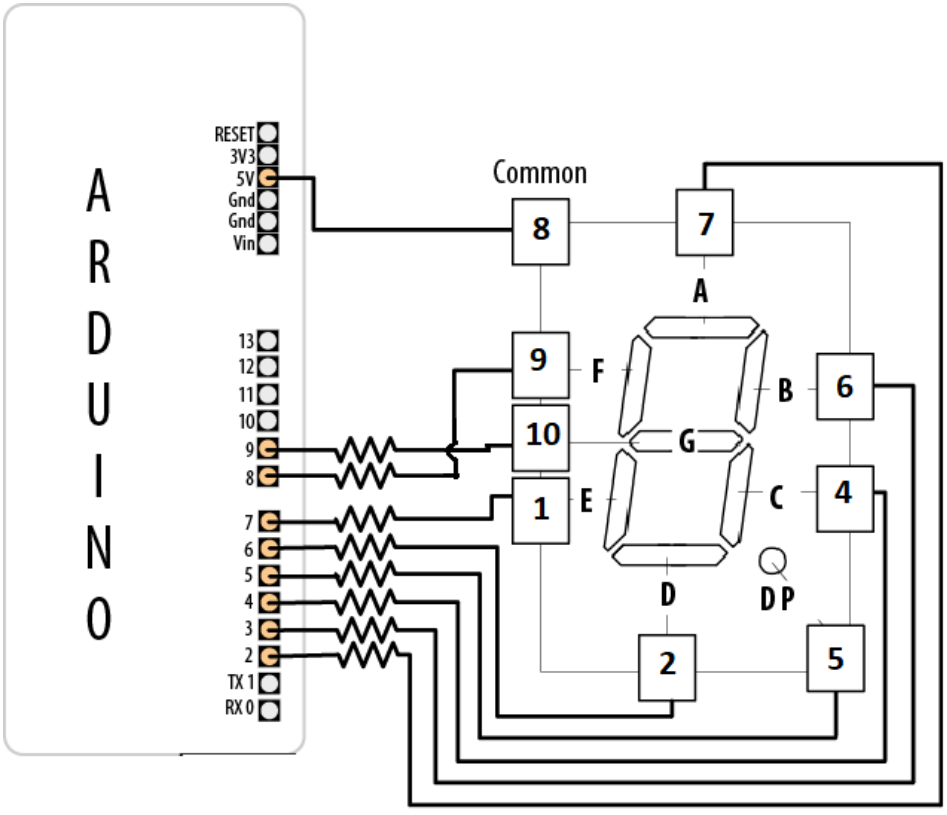
```
blinkLED(thirdLedPin, 1000);
}
void blinkLED(int pin, int duration)
{
digitalWrite(pin, HIGH); // turn LED on
delay(duration);
digitalWrite(pin, LOW); // turn LED off
delay(duration);
}
```

7.2 Điều chỉnh độ sáng Led

```
/*
* LedBrightness sketch
* controls the brightness of LEDs on analog output ports
*/
const int firstLed = 3; // specify the pin for each of the LEDs
const int secondLed = 5;
const int thirdLed = 6;
int brightness = 0;
int increment = 1;
void setup()
{
// pins driven by analogWrite do not need to be declared as outputs
}
void loop()
{
if(brightness > 255)
{
increment = -1; // count down after reaching 255
}
```

```
else if(brightness < 1)
{
increment = 1; // count up after dropping back down to 0
}
brightness = brightness + increment; // increment (or decrement sign is minus)
// write the brightness value to the LEDs
analogWrite(firstLed, brightness);
analogWrite(secondLed, brightness);
analogWrite(thirdLed, brightness );
delay(10); // 10ms for each step change means 2.55 secs to fade up or down
}
```

7.3 Điều khiển led 7 đoạn




```
* Shows numerals ranging from 0 through 9 on a single-digit display
* This example counts seconds from 0 to 9
*/
```

```
// bits representing segments A through G (and decimal point) for numerals 0-9
const byte numeral[10] = {
//ABCDEFG /dp
B11111100, // 0
B01100000, // 1
B11011010, // 2
B11110010, // 3
B01100110, // 4
B10110110, // 5
B00111110, // 6
B11100000, // 7
B11111110, // 8
B11100110, // 9
};
// pins for decimal point and each segment
// dp,G,F,E,D,C,B,A
const int segmentPins[8] = { 5,9,8,7,6,4,3,2};
void setup()
{
for(int i=0; i < 8; i++)
{
pinMode(segmentPins[i], OUTPUT); // set segment and DP pins to output
}
}
void loop()
{
for(int i=0; i <= 10; i++)
{
showDigit(i);
```

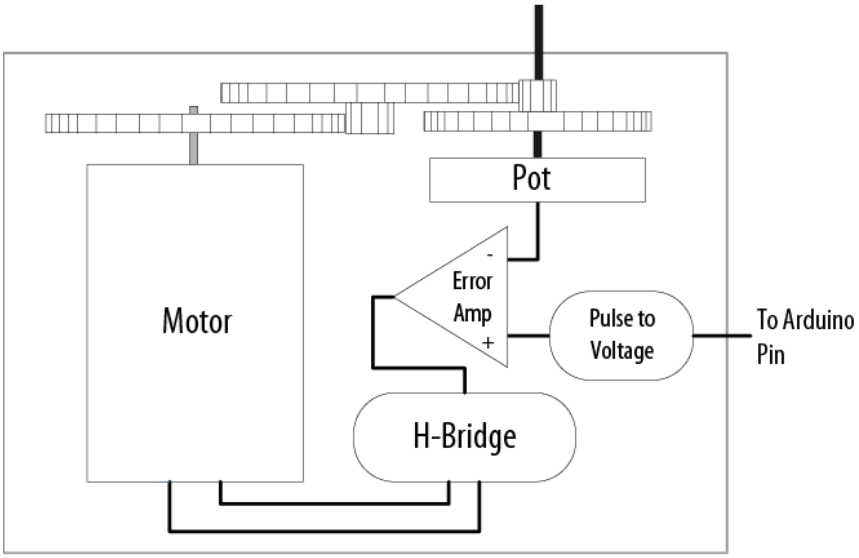
```
delay(1000);
}
// the last value if i is 10 and this will turn the display off
delay(2000); // pause two seconds with the display off
}
// Displays a number from 0 through 9 on a 7-segment display
// any value not within the range of 0-9 turns the display off
void showDigit( int number)
{
boolean isBitSet;
for(int segment = 1; segment < 8; segment++)
{
if( number < 0 || number > 9){
isBitSet = 0; // turn off all segments
}
else{
// isBitSet will be true if given bit is 1
isBitSet = bitRead(numeral[number], segment);
}
isBitSet = ! isBitSet; // remove this line if common cathode display
digitalWrite( segmentPins[segment], isBitSet);
}
}
```

CHƯƠNG 8: ĐIỀU KHIỂN ĐỘNG CƠ

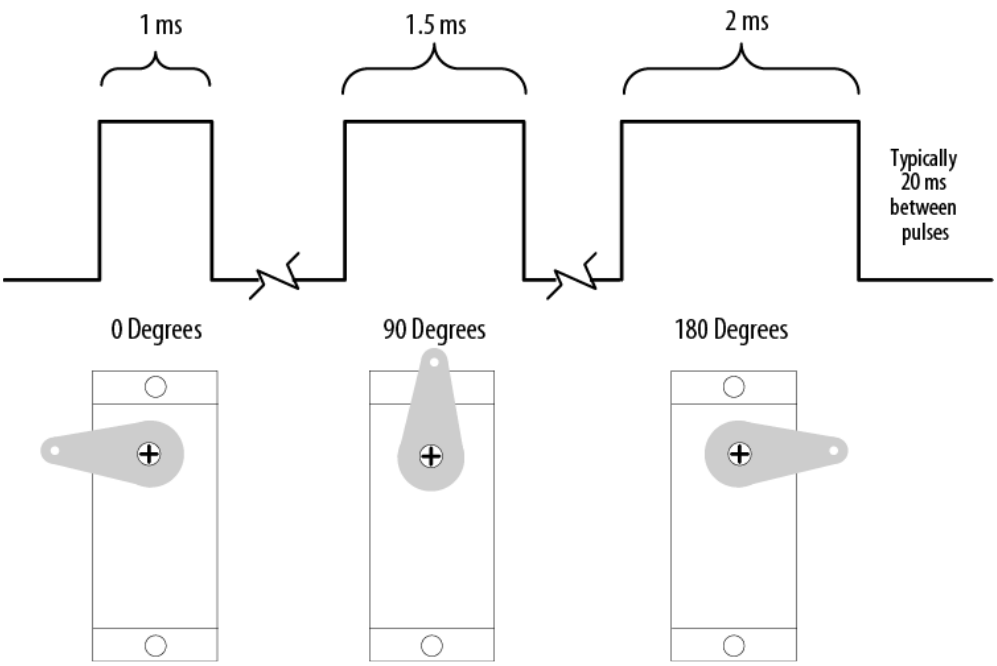
8.1 Điều khiển vị trí động cơ Servo

Servos cho phép bạn điều khiển chính xác chuyển động vật lý vì chúng thường di chuyển đến một vị trí thay vì xoay liên tục. Chúng rất lý tưởng để làm cho một cái gì đó xoay trong phạm vi từ 0 đến 180 độ. Các servo dễ dàng kết nối và điều khiển vì trình điều khiển động cơ được tích hợp vào servo.

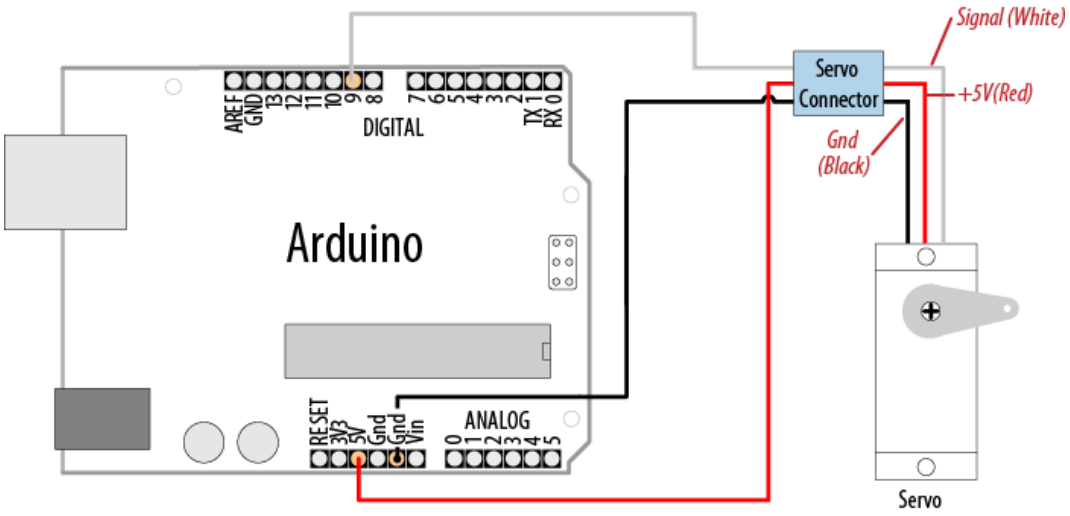
Servos chứa một động cơ nhỏ được kết nối thông qua các bánh răng với trục đầu ra. Trục đầu ra cũng được kết nối với một chiết áp để cung cấp phản hồi vị trí cho mạch điều khiển bên trong. Bạn có thể nhận được các động cơ quay liên tục có ngắt phản hồi vị trí để bạn có thể ra lệnh cho servo xoay liên tục theo chiều kim đồng hồ và ngược chiều kim đồng hồ với một tốc độ kiểm soát.



Servos đáp ứng với những thay đổi trong thời gian của một xung. Xung ngắn từ 1 ms trở xuống sẽ khiến servo quay đến một cực; thời lượng xung từ 2 ms trở lên sẽ xoay servo sang cực khác. Các xung nằm giữa các giá trị này sẽ xoay servo đến một vị trí tỷ lệ thuận với độ rộng xung. Không có tiêu chuẩn nào cho mối quan hệ chính xác giữa các xung và vị trí, và bạn có thể cần phải sửa lại các lệnh trong chương trình của mình để điều chỉnh phạm vi của các servo.



❖ Sơ đồ kết nối Servo với Arduino:

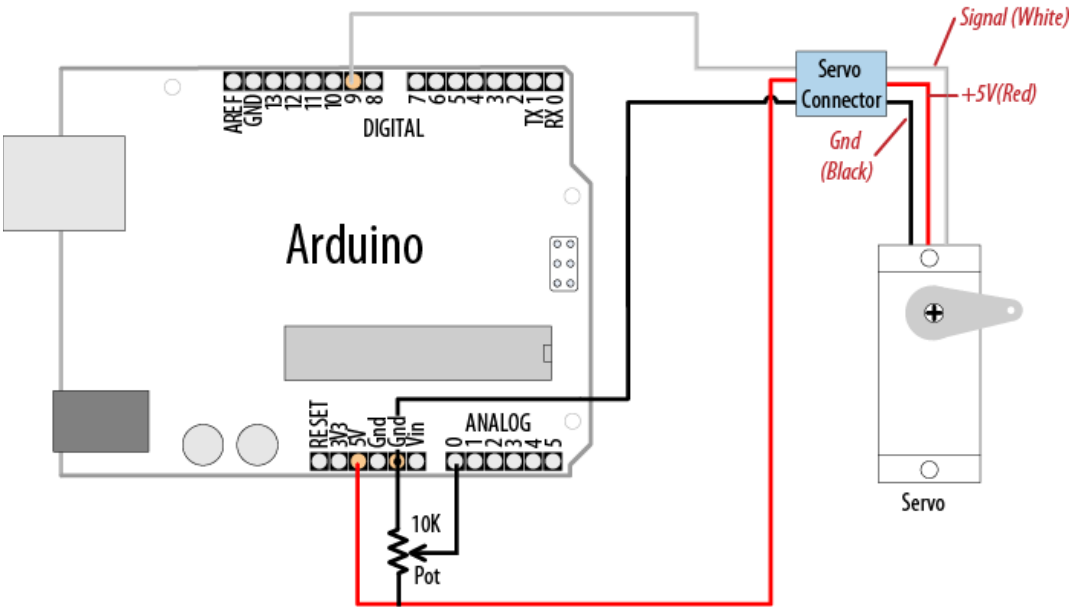


Chương trình mẫu:

```
#include <Servo.h>
Servo myservo; // create servo object to control a servo
int angle = 0; // variable to store the servo position
void setup()
{
```

```
myservo.attach(9); // attaches the servo on pin 10 to the servo object
}
void loop()
{
for(angle = 0; angle < 180; angle += 1) // goes from 0 degrees to 180 degrees
{ // in steps of 1 degree
myservo.write(angle); // tell servo to go to position in variable 'angle'
delay(20); // waits 20ms between servo commands
}
for(angle = 180; angle >= 1; angle -= 1) // goes from 180 degrees to 0 degrees
{
myservo.write(pos); // tell servo to go to position in variable 'pos'
delay(20); // waits 20ms between servo commands
}
}
```

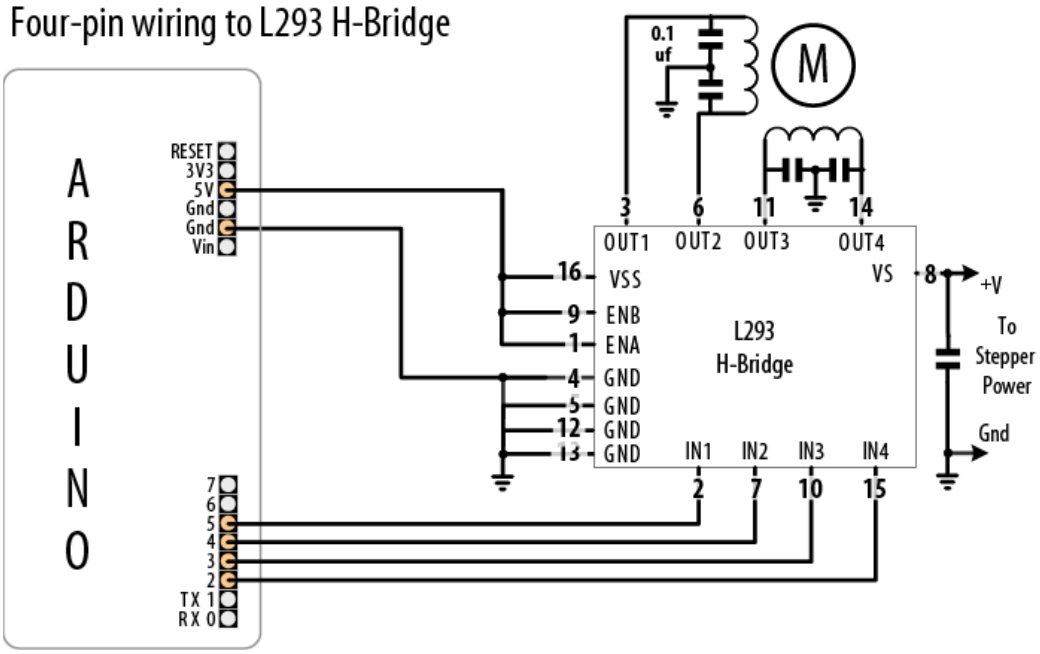
8.2 Điều khiển động cơ Servo bằng biến trở




```
#include <Servo.h>
Servo myservo; // create servo object to control a servo
```

```
int potpin = 0; // analog pin used to connect the potentiometer
int val; // variable to read the value from the analog pin
void setup()
{
myservo.attach(9); // attaches the servo on pin 9 to the servo object
}
void loop()
{
val = analogRead(potpin); // reads the value of the potentiometer
val = map(val, 0, 1023, 0, 179); // scale it to use it with the servo
myservo.write(val); // sets position to the scaled value
delay(15); // waits for the servo to get there
}
```

8.3 Điều khiển động cơ bước



Chương trình:

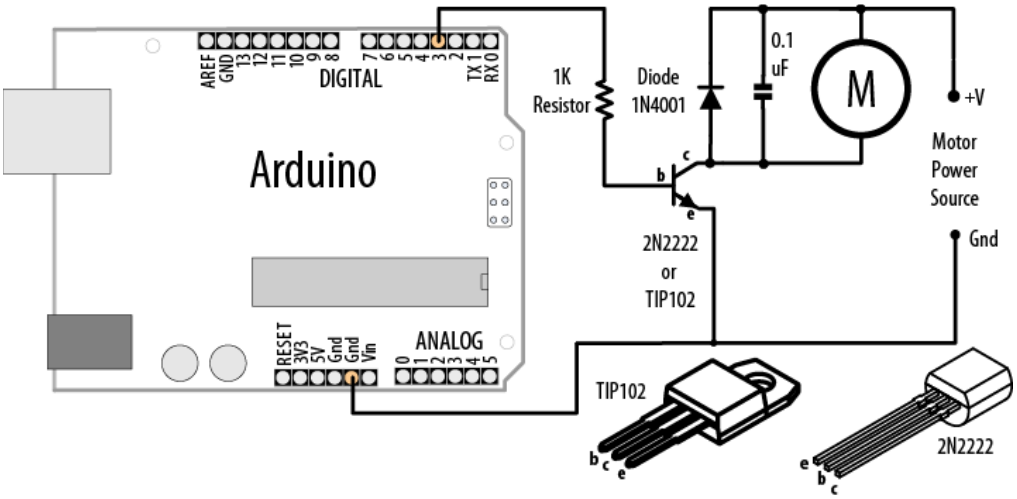
```
/*
* Stepper_bipolar sketch
```

```
*
* stepper is controlled from the serial port.
* a numeric value followed by '+' or '-' steps the motor
*
*
* http://www.arduino.cc/en/Reference/Stepper
*/
#include <Stepper.h>
// change this to the number of steps on your motor
#define STEPS 24
// create an instance of the stepper class, specifying
// the number of steps of the motor and the pins it's
// attached to
Stepper stepper(STEPS, 2, 3, 4, 5);
int steps = 0;
void setup()
{
// set the speed of the motor to 30 RPMs
stepper.setSpeed(30);
Serial.begin(9600);
}
void loop()
{
if ( Serial.available()) {
char ch = Serial.read();
if(ch >= '0' && ch <= '9'){ // is ch a number?
steps = steps * 10 + ch - '0'; // yes, accumulate the value
}
else if(ch == '+'){
```

```
stepper.step(steps);
steps = 0;
}
else if(ch == '-'){
stepper.step(steps * -1);
steps = 0;
}
}
}
```

8.4 Điều khiển động cơ DC

❖ Điều khiển động cơ sử dụng Transistor



Chương trình:

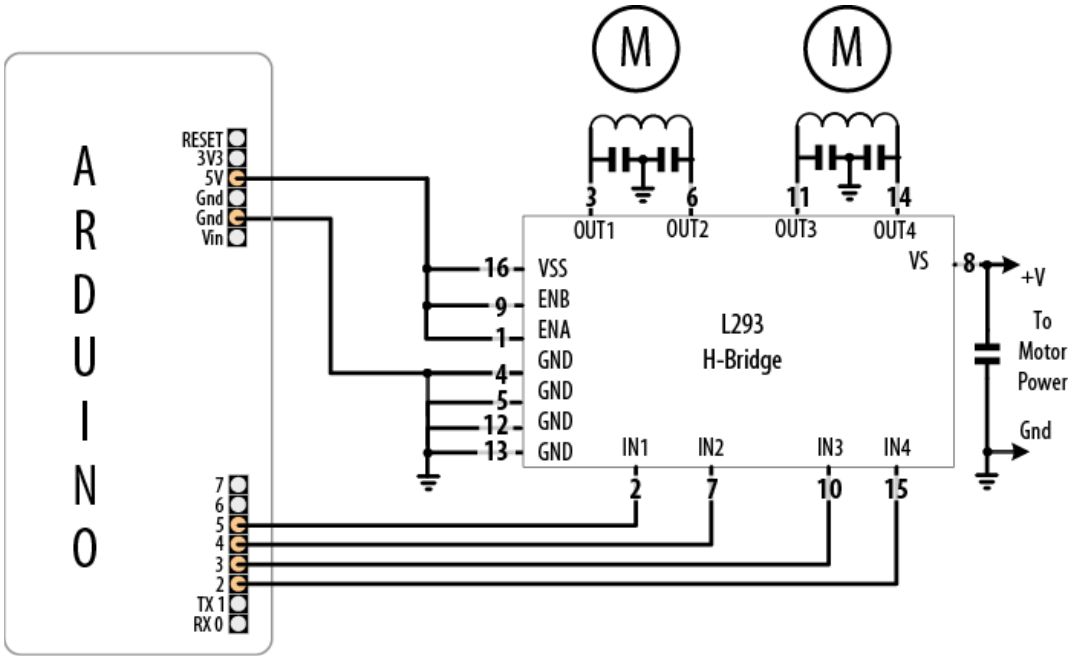
```
/*
* SimpleBrushed sketch
* commands from serial port control motor speed
* digits '0' through '9' are valid where '0' is off, '9' is max speed
*/
const int motorPins = 3; // motor driver is connected to pin 3
void setup()
```

```
{
Serial.begin(9600);
}
void loop()
{
if ( Serial.available()) {
char ch = Serial.read();
if(ch >= '0' && ch <= '9') // is ch a number?
{
int speed = map(ch, '0', '9', 0, 255);
analogWrite(3, speed);
Serial.println(speed);
}
else
{
Serial.print("Unexpected character ");
Serial.println(ch);
}
}
}
```

- Điều khiển động cơ sử dụng cầu H

Logic table for H-Bridge

EN	IN1	IN2	Function
HIGH	LOW	HIGH	Turn clockwise
HIGH	HIGH	LOW	Turn counterclockwise
HIGH	LOW	LOW	Motor stop
HIGH	HIGH	HIGH	Motor stop
LOW	Ignored	Ignored	Motor stop



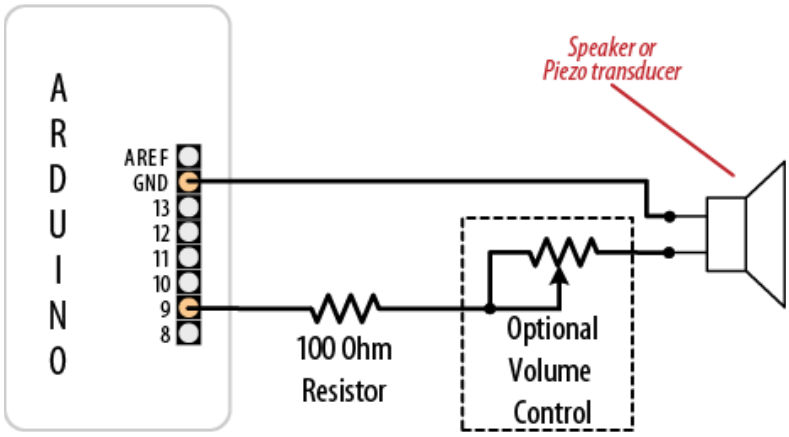
Chương trình:

```
/*
* Brushed_H_Bridge_simple sketch
* commands from serial port control motor direction
* + or - set the direction, any other key stops the motor
*/
const int in1Pin = 5; // H-Bridge input pins
const int in2Pin = 4;
void setup()
{
Serial.begin(9600);
pinMode(in1Pin, OUTPUT);
pinMode(in2Pin, OUTPUT);
Serial.println("+ - to set direction, any other key stops motor");
}
void loop()
{
```

```
if ( Serial.available()) {
char ch = Serial.read();
if (ch == '+')
{
Serial.println("CW");
digitalWrite(in1Pin,LOW);
digitalWrite(in2Pin,HIGH);
}
else if (ch == '-')
{
Serial.println("CCW");
digitalWrite(in1Pin,HIGH);
digitalWrite(in2Pin,LOW);
}
else
{
Serial.print("Stop motor");
digitalWrite(in1Pin,LOW);
digitalWrite(in2Pin,LOW);
}
}
}
```

CHƯƠNG 9: TẠO ÂM THANH

9.1 Tạo âm



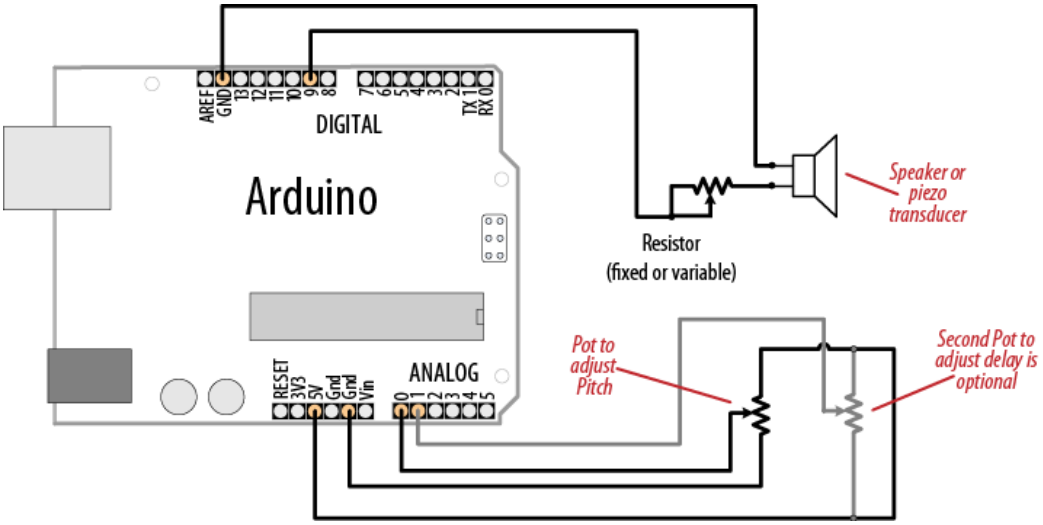
Chương trình:

```
/*
* Tone sketch
*
* Plays tones through a speaker on digital pin 9
* frequency determined by values read from analog port
*/
const int speakerPin = 9; // connect speaker to pin 9
const int pitchPin = 0; // pot that will determine the frequency of the tone
void setup()
{
}
void loop()
{
int sensor0Reading = analogRead(pitchPin); // read input to set frequency
// map the analog readings to a meaningful range
int frequency = map(sensor0Reading, 0, 1023, 100,5000); //100Hz to 5kHz
int duration = 250; // how long the tone lasts
```

tone(speakerPin, frequency, duration); // play the tone

delay(1000); //pause one second

}



Chương trình:

```
const int speakerPin = 9; // connect speaker to pin 9
const int pitchPin = 0; // input that determines frequency of the tone
const int durationPin = 1; // input that will determine the duration of the tone
void setup()
{
}
void loop()
{
int sensor0Reading = analogRead(pitchPin); // read input for frequency
int sensor1Reading = analogRead(durationPin); // read input for duration
// map the analog readings to a meaningful range
int frequency = map(sensor0Reading, 0, 1023, 100,5000); // 100Hz to 5kHz
int duration = map(sensor1Reading, 0, 1023, 100,1000); // dur 0.1-1 second
tone(speakerPin, frequency, duration); // play the tone
delay(duration); //wait for the tone to finish
}
```

9.2 Tạo giai điệu

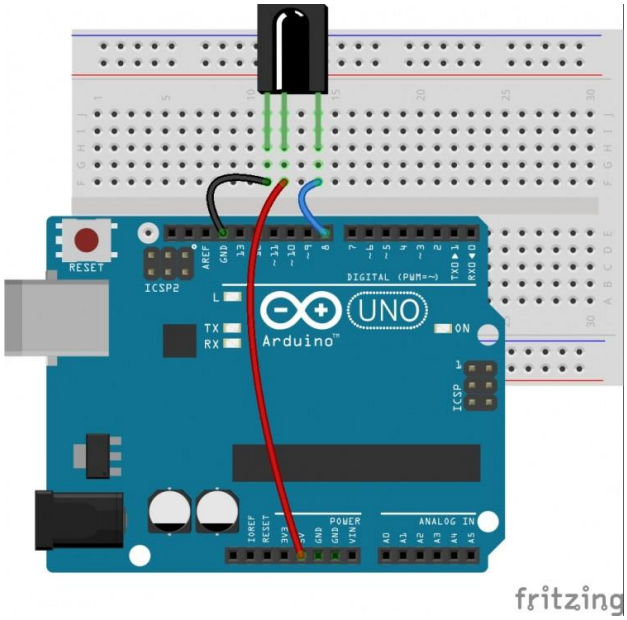
Tạo một giai điệu đơn giản

```
/*
* Twinkle sketch
*
* Plays "Twinkle, Twinkle Little Star"
*
* speaker on digital pin 9
*/
const int speakerPin = 9; // connect speaker to pin 9
char noteNames[] = {'C','D','E','F','G','a','b'};
unsigned int frequencies[] = {262,294,330,349,392,440,494};
const byte noteCount = sizeof(noteNames); // the number of notes (7 in this
example)
//notes, a space represents a rest
char score[] = "CCGGaaGFFEEDDC GGFFEEDGGFFEED CCGGaaGFFEEDDC ";
const byte scoreLen = sizeof(score); // the number of notes in the score
void setup()
{
}
void loop()
{
for (int i = 0; i < scoreLen; i++)
{
int duration = 333; // each note lasts for a third of a second
playNote(score[i], duration); // play the note
}
delay(4000); // wait four seconds before repeating the song
}
```

```
void playNote(char note, int duration)
{
// play the tone corresponding to the note name
for (int i = 0; i < noteCount; i++)
{
// try and find a match for the noteName to get the index to the note
if (noteNames[i] == note) // find a matching note name in the array
tone(speakerPin, frequencies[i], duration); // play the note using the
                                                    //frequency
}
// if there is no match then the note is a rest, so just do the delay
delay(duration);
}
```

CHƯƠNG 10: ĐIỀU KHIỂN XA

10.1 Nhận tín hiệu từ bộ điều khiển xa của tivi

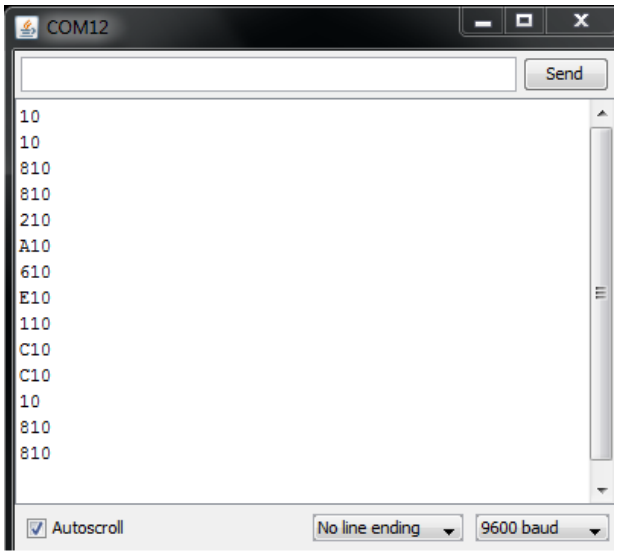


Chương trình:

```
#include <IRremote.h> // thư viện hỗ trợ IR remote
const int receiverPin = 8; // chân digital 8 dùng để đọc tín hiệu
IRrecv irrecv(receiverPin); // tạo đối tượng IRrecv mới
decode_results results;// lưu giữ kết quả giải mã tín hiệu
void setup()
{
        Serial.begin(9600);// serial baudrate 9600
        irrecv.enableIRIn(); // start the IR receiver
}
void loop()
{
 if (irrecv.decode(&results)) // nếu nhận được tín hiệu
        {
        Serial.println(results.value, HEX); // in ra Serial Monitor
        delay(200);
```

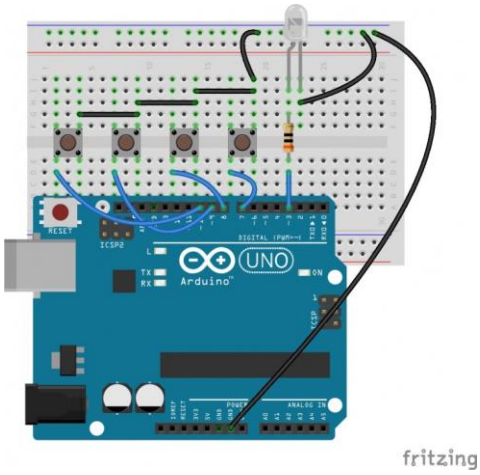
```
        irrecv.resume(); // nhận giá trị tiếp theo
    }
    }
```

Sau khi upload đoạn code trên, mở cửa sổ Serial và bấm các nút của remote, tùy loại remote sẽ có giá trị trả về khác nhau như sau:



Ví dụ: Phím số1 là 0x10, phím số2 là 0x810…

10.2 Giải mã tín hiệu từ bộ điều khiển xa của tivi

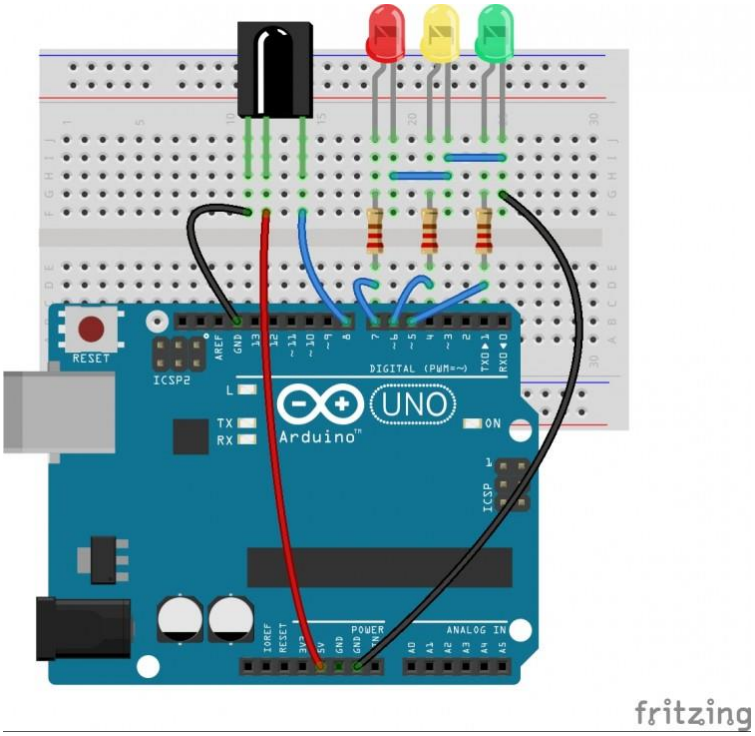


Chương trình:

```
#include <IRremote.h>
IRsend irsend;                    // gửi tín hiệu hồng hoại
const int CH1 = 7;                // nút số 1
```

```
const int CH2 = 8;                    // nút số 2
const int CH3 = 9;                    // nút số 3
const int POWER = 10;                 // nút Power
void setup()
{
  Serial.begin(9600);
  /* điện trở nội kéo lên cho tất cả nút bấm*/
  pinMode(CH1, INPUT_PULLUP);
  pinMode(CH2, INPUT_PULLUP);
  pinMode(CH3, INPUT_PULLUP);
  pinMode(POWER, INPUT_PULLUP);
}
void loop()
{
  if (digitalRead(CH1) == 0)          // khi nút RED được nhấn
    irsend.sendSony(0x10, 12);        // gửi tín hiệu HEX 0x10, 12 bits
  else if (digitalRead(CH2) == 0)             // khi nút YELLOW được nhấn
    irsend.sendSony(0x810, 12);               // gửi tín hiệu HEX 0x810, 12 bits
  else if (digitalRead(CH3) == 0)             // khi nút GREEN được nhấn
    irsend.sendSony(0x410, 12);               // gửi tín hiệu HEX 0x410, 12 bits
  else if (digitalRead(POWER) == 0)  // khi nút POWER được nhấn
    irsend.sendSony(0xA90, 12);               // gửi tín hiệu HEX 0xA90, 12 bits
  delay(50);
}
```

10.3 Điều khiển từ xa thiết bị



Chương trình:

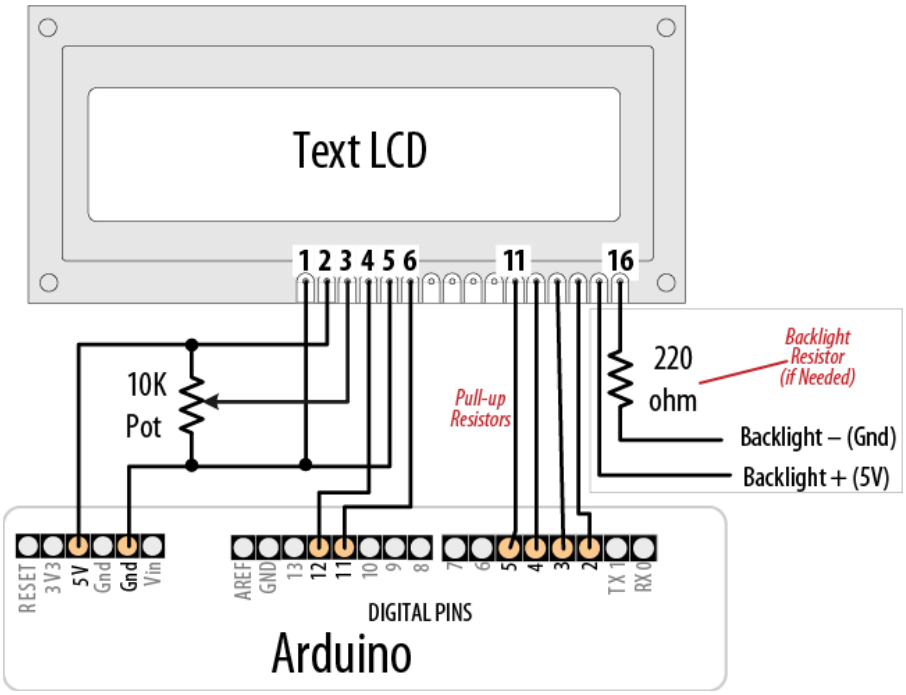
```
#include <IRremote.h>          // thư viện hỗ trợ IR remote
const int receiverPin = 8;             // chân digital 8 dùng để đọc tín hiệu
IRrecv irrecv(receiverPin);    // tạo đối tượng IRrecv mới
decode_results results;        // lưu giữ kết quả giải mã tín hiệu
const int RED = 7;             // LED đỏ
const int YELLOW = 6;                  // LED vàng
const int GREEN = 5;           // LED xanh
/* trạng thái của các LEDs*/
boolean stateRED = false;
boolean stateYELLOW = false;
boolean stateGREEN = false;
void setup()
{
```

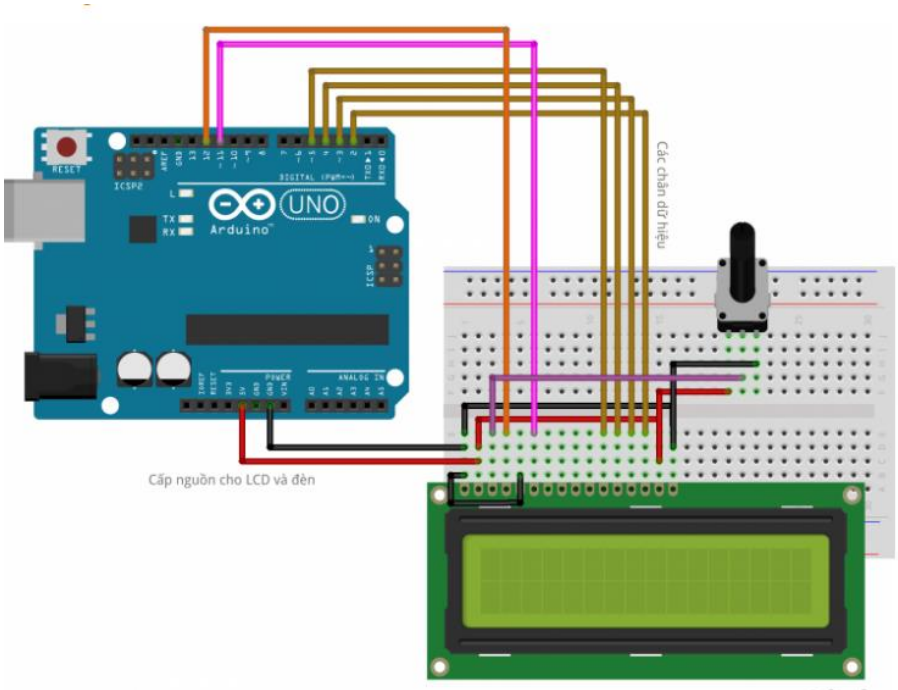
```
  Serial.begin(9600);               // serial
  irrecv.enableIRIn();              // start the IR receiver
  pinMode(RED, OUTPUT);
  pinMode(YELLOW, OUTPUT);
  pinMode(GREEN, OUTPUT);
}
// translate IR signals
void translateIR()
{
  switch(results.value)
  {
    case 0x10:  stateRED = !stateRED;
                digitalWrite(RED, stateRED);
                break;
    case 0x810: stateYELLOW = !stateYELLOW;
                digitalWrite(YELLOW, stateYELLOW);
                break;
    case 0x410: stateGREEN = !stateGREEN;
                digitalWrite(GREEN, stateGREEN);
                break;
    case 0xA90: stateRED = stateYELLOW = stateGREEN = false;
                digitalWrite(RED, 0);
                digitalWrite(YELLOW, 0);
                digitalWrite(GREEN, 0);
  }
}
void loop()
{
  if (irrecv.decode(&results))    // nếu nhận được tín hiệu
```

```
  {
    translateIR();
    Serial.println(results.value, HEX);
    delay(200);
    irrecv.resume();                    // nhận giá trị tiếp theo
  }
}
```

CHƯƠNG 11: ĐIỀU KHIỂN LCD

11.1 Kết nối và sử dụng LCD





Chương trình:

```
//Thêm thư viện LiquitCrystal - nó có sẵn vì vậy bạn không cần cài thêm gì cả
#include <LiquidCrystal.h>
//Khởi tạo với các chân
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup() {
//Thông báo đây là LCD 1602
lcd.begin(16, 2); //lcd 16 cột và 2 hàng
//In ra màn hình lcd dòng chữ Toi yeu Arduino
lcd.print("Toi yeu Arduino!");
}
void loop() {
// đặt con trỏ vào cột 0, dòng 1
// Lưu ý: dòng 1 là dòng thứ 2, lòng 0 là dòng thứ 1.
lcd.setCursor(0, 1);
// In ra dong chu
lcd.print(" Arduino.VN");
}
```

11.2 Định dạng văn bản

Phác thảo này hiển thị đếm ngược từ 9 đến 0. Sau đó, nó sẽ hiển thị một chuỗi các chữ số trong ba cột gồm bốn ký tự. Thay đổi numRows và numCols để khớp với các hàng và cột trong màn hình LCD của bạn:

Chương trình:

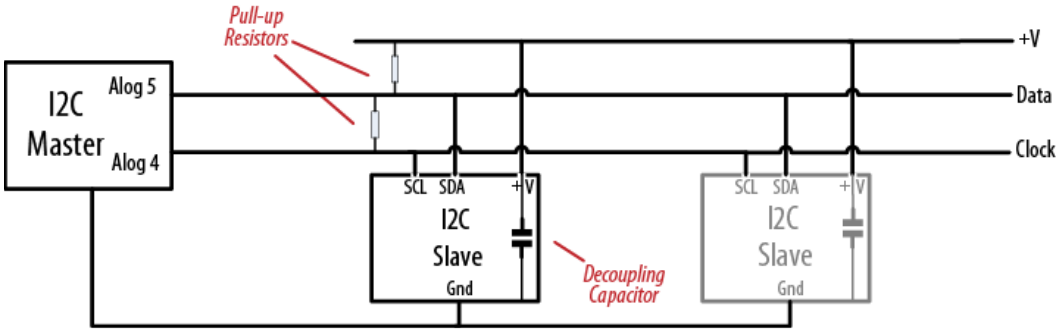
```
/*
LiquidCrystal Library - FormatText
*/
#include <LiquidCrystal.h> // include the library code:
//constants for the number of rows and columns in the LCD
const int numRows = 2;
const int numCols = 16;
```

```
int count;
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup()
{
lcd.begin(numCols, numRows);
lcd.print("Starting in "); // this string is 12 characters long
for(int i=9; i > 0; i--) // count down from 9
{
// the top line is row 0
lcd.setCursor(12,0); // move the cursor to the end of the string
printed above
lcd.print(i);
delay(1000);
}
}
void loop()
{
int columnWidth = 4; //spacing for the columns
int displayColumns = 3; //how many columns of numbers
lcd.clear();
for( int col=0; col < displayColumns; col++)
{
lcd.setCursor(col * columnWidth, 0);
count = count+ 1;
lcd.print(count);
}
delay(1000);
}
```

CHƯƠNG 12: GIAO TIẾP I2C

12.1 I2C

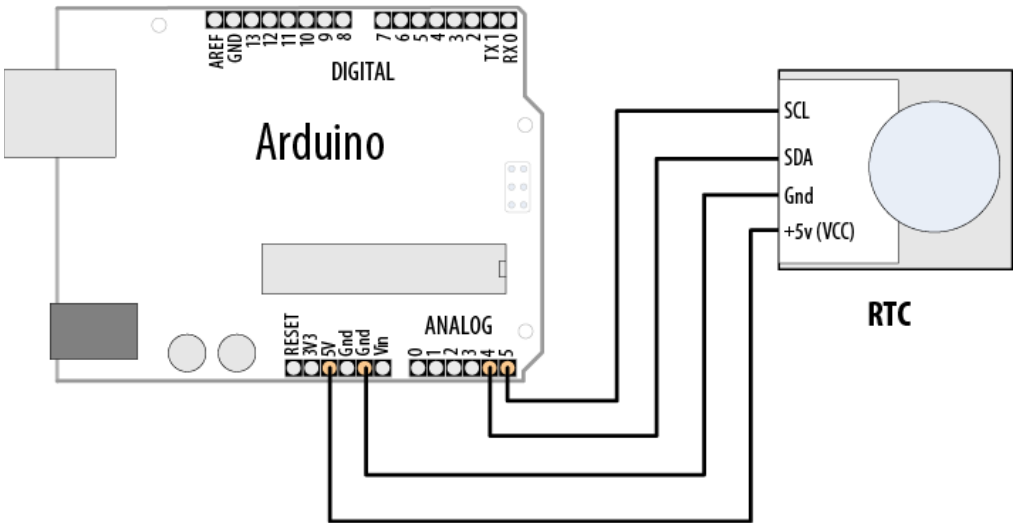
Hai kết nối cho bus I2C được gọi là SCL và SDA. Chúng có sẵn trên bo mạch Arduino tiêu chuẩn sử dụng chân analog 5 cho SCL, cung cấp tín hiệu xung clock và chân analog 4 cho SDL, dùng để truyền dữ liệu (trên Mega, sử dụng chân kỹ thuật số 20 cho SDA và chân 21 cho SCL). Một thiết bị trên bus I2C được coi là thiết bị chính(Master). Công việc của nó là phối hợp chuyển thông tin giữa các thiết bị khác (Slave) được đính kèm. Chỉ có một Master, và trong hầu hết các trường hợp, Arduino là master, điều khiển các chip khác được gắn vào nó.



Thiết bị Slave được xác định bởi số địa chỉ của nó. Mỗi Slave phải có một địa chỉ duy nhất. Một số thiết bị I2C có địa chỉ cố định, trong khi các thiết bị khác cho phép bạn định cấu hình địa chỉ của chúng bằng cách đặt chân cao hoặc thấp hoặc bằng cách gửi lệnh khởi tạo.

Thư viện Arduino Wire ẩn tất cả các chức năng cấp thấp cho I2C và cho phép các lệnh đơn giản được sử dụng để khởi tạo và giao tiếp với các thiết bị.

❖ Giao tiếp với đồng hồ thời gian thực bên ngoài

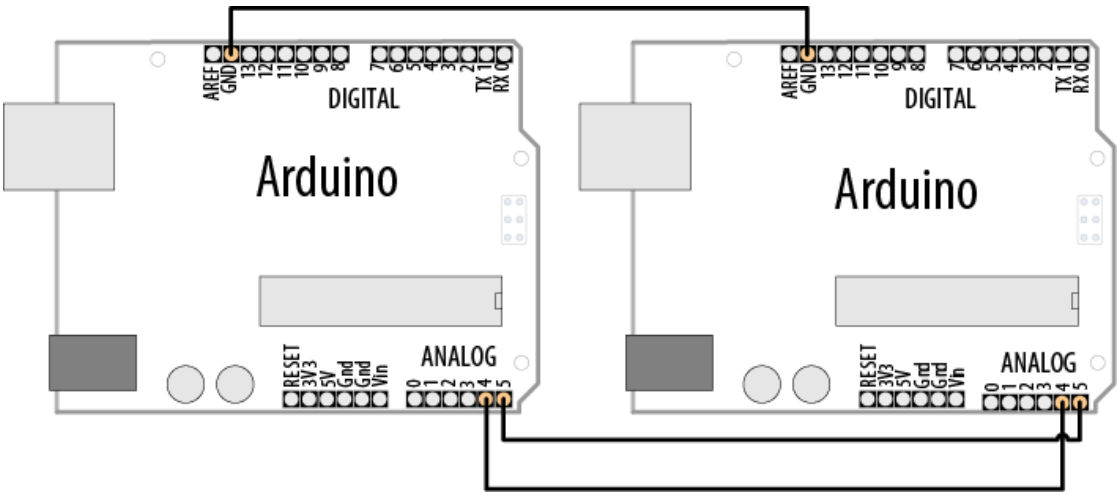



```
/*
* I2C_RTC sketch
* example code for using Wire library to access real-time clock
*
*/
#include <Wire.h>
const byte DS1307_CTRL_ID = 0x68; // address of the DS1307 real-time clock
const byte NumberOfFields = 7; // the number of fields (bytes) to request from
                                //the RTC
int Second ;
int Minute;
int Hour;
int Day;
int Wday;
int Month;
int Year;
void setup() {
Serial.begin(9600);
```

```
Wire.begin();
}
void loop()
{
Wire.beginTransmission(DS1307_CTRL_ID);
Wire.send(0x00);
Wire.endTransmission();
// request the 7 data fields (secs, min, hr, dow, date, mth, yr)
Wire.requestFrom(DS1307_CTRL_ID, NumberOfFields);
Second = bcd2dec(Wire.receive() & 0x7f);
Minute = bcd2dec(Wire.receive() );
Hour = bcd2dec(Wire.receive() & 0x3f); // mask assumes 24hr clock
Wday = bcd2dec(Wire.receive() );
Day = bcd2dec(Wire.receive() );
Month = bcd2dec(Wire.receive() );
Year = bcd2dec(Wire.receive() );
Year = Year + 2000; // RTC year 0 is year 2000
digitalClockDisplay(); // display the time
delay(1000);
}
// Convert Binary Coded Decimal (BCD) to Decimal
byte bcd2dec(byte num)
{
return ((num/16 * 10) + (num % 16));
}
void digitalClockDisplay(){
// digital clock display of the time
Serial.print(Hour);
printDigits(Minute);
```

```
printDigits(Second);
Serial.print(" ");
Serial.print(Day);
Serial.print(" ");
Serial.print(Month);
Serial.print(" ");
Serial.print(Year);
Serial.println();
}
// utility function for clock display: prints preceding colon and leading 0
void printDigits(int digits){
Serial.print(":");
if(digits < 10)
Serial.print('0');
Serial.print(digits);
}
```

12.2 Giao tiếp giữa 2 mạch Arduino



Master gửi các ký tự nhận được trên cổng nối tiếp tới một Slave Arduino bằng I2C:

```
/*
```

```
* I2C_Master
* Echo Serial data to an I2C slave
*/
#include <Wire.h>
const int address = 4; //the address to be used by the communicating devices
void setup()
{
Wire.begin();
}
void loop()
{
char c;
if(Serial.available() > 0 )
{
// send the data
Wire.beginTransmission(address); // transmit to device
Wire.send(c);
Wire.endTransmission();
}
}
```

Các Slave in các ký tự nhận được qua I2C sang cổng nối tiếp của nó:

```
/*
* I2C_Slave
* monitors I2C requests and echoes these to the serial port
*
*/
#include <Wire.h>
const int address = 4; //the address to be used by the communicating devices
void setup()
```

```
{
Wire.begin(address); // join I2C bus using this address
Wire.onReceive(receiveEvent); // register event to handle requests
}
void loop()
{
// nothing here, all the work is done in receiveEvent
}
void receiveEvent(int howMany)
{
while(Wire.available() > 0)
{
char c = Wire.receive(); // receive byte as a character
Serial.print(c); // echo
}
}
```