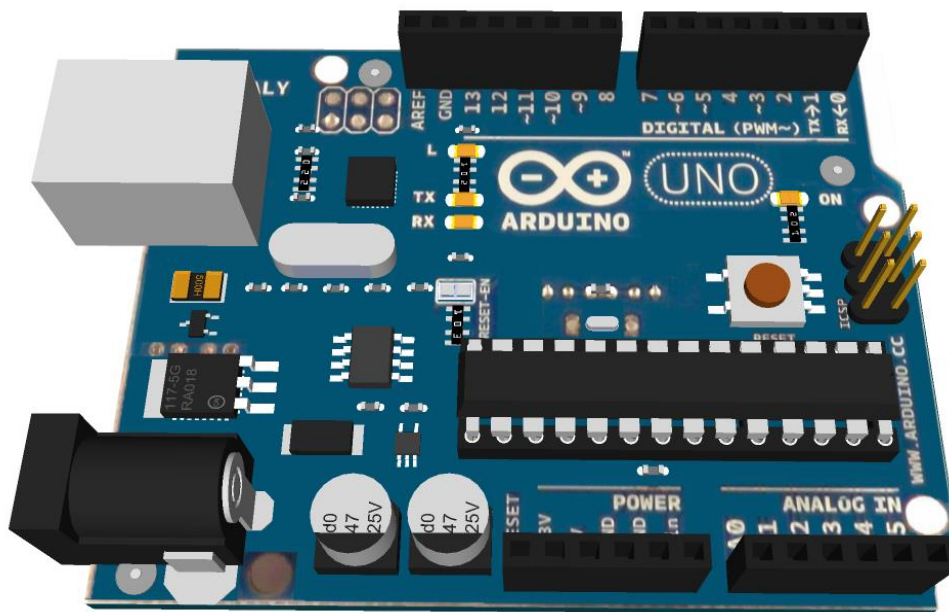


HỌC VIỆN HÀNG KHÔNG VIỆT NAM
KHOA ĐIỆN TỬ VIỄN THÔNG



HƯỚNG DẪN SỬ DỤNG CƠ BẢN ARDUINO



TP.HCM, Tháng 5, Năm 2014.

MỤC LỤC:

Lời nói đầu.

Chương 1: Tổng quan về Arduino Uno.....	1
1. Tổng quan.....	1
2. Sơ đồ chân của Arduino.....	2
Chương 2: Cài đặt chương trình Arduino IDE và Driver cho Arduino	4
1. Cài đặt chương trình Arduino IDE	4
2. Cài đặt Driver	5
3. Arduino IDE	7
Chương 3: Hướng dẫn cài đặt bản mô phỏng Arduino trên Proteus.....	11
Chương 4: Giao tiếp Arduino với một số linh kiện điện tử.....	13
1) Project 1: Led nhấp nháy.....	13
2) Project 2 : Đèn sáng khi nhấn phím.....	18
3) Project 3 : Led sáng dần từ led 1 đến led 10 và ngược lại.....	21
4) Project 4 : Led sáng dần từ led 1 đến led 10 và ngược lại thời gian delay thay đổi được.....	24
5) Project 5: Điều khiển tốc độ động cơ bằng PWM.....	27
6) Project 6 : Điều khiển động cơ bằng L293D.....	31
7) Project 7: Giao tiếp Arduino với LCD 16x2.....	34
8) Project 8: Giao tiếp với máy tính.....	47
9) Project 9. Đo nhiệt độ môi trường dùng LM35D hiển thị LCD và Serial Monitor.....	49
10) Project 10: Giao tiếp Arduino với Servo motor.....	54
Tài liệu tham khảo.....	56

Lời Nói Đầu.

Arduino đã và đang được sử dụng rất rộng rãi trên thế giới, và ngày càng chứng tỏ được sức mạnh của chúng thông qua vô số ứng dụng độc đáo của người dùng trong cộng đồng nguồn mở. Arduino thực sự đã gây sóng gió trên thị trường người dùng trên toàn thế giới trong vài năm gần đây, số lượng người dùng cực lớn và đa dạng với trình độ trải rộng từ bậc phổ thông lên đến đại học đã làm cho ngay cả những người tạo ra chúng phải ngạc nhiên về mức độ phổ biến. Tuy nhiên tại Việt Nam Arduino vẫn còn chưa được biết đến nhiều, các tài liệu liên quan đến nó vẫn còn rất hạn chế. Được sự giới thiệu và chỉ dẫn của thầy **Nguyễn Thanh Dũng**, sau một thời gian tìm hiểu tác giả đã biên soạn tài liệu “**HƯỚNG DẪN SỬ DỤNG CƠ BẢN ARDUINO**”. Trong tài liệu này cung cấp cho bạn đọc một lượng kiến thức cơ bản nhất về Arduino cũng như các ứng dụng thực tế của nó.

Tài liệu gồm có các nội dung sau:

Chương 1: Tổng quan về Arduino Uno.

Chương 2: Cài đặt chương trình Arduino IDE và Driver cho Arduino.

Chương 3: Hướng dẫn cài đặt bản mô phỏng Arduino trên Proteus.

Chương 4: Giao tiếp Arduino với một số linh kiện điện tử.

Khi biên soạn, tác giả đã tham khảo một số tài liệu nước ngoài để tài liệu vừa đảm bảo về mặt nội dung vừa có thể tiếp cận được với bạn đọc.

Khi viết tác giả đã cố gắng để tài liệu được hoàn chỉnh nhất song chắc chắn không tránh khỏi sai sót, vì vậy rất mong nhận được sự góp ý của bạn đọc.

Mọi ý kiến đóng góp xin liên hệ: trungtin.vaa@gmail.com

Tác giả

SV: NGUYỄN TRUNG TÍN

Chương 1: Tổng quan về Arduino Uno.

1. Tổng quan.

Arduino thật ra là một bo mạch vi xử lý được dùng để lập trình tương tác với các thiết bị phần cứng như cảm biến, động cơ, đèn hoặc các thiết bị khác. Đặc điểm nổi bật của Arduino là môi trường phát triển ứng dụng cực kỳ dễ sử dụng, với một ngôn ngữ lập trình có thể học một cách nhanh chóng ngay cả với người ít am hiểu về điện tử và lập trình. Và điều làm nên hiện tượng Arduino chính là mức giá rất thấp và tính chất nguồn mở từ phần cứng tới phần mềm.

Arduino Uno là sử dụng chip Atmega328. Nó có 14 chân digital I/O, 6 chân đầu vào (input) analog, thạch anh dao động 16Mhz. Một số thông số kỹ thuật như sau :

Chip	ATmega328
Điện áp cấp nguồn	5V
Điện áp đầu vào (input) (kiến nghị)	7-12V
Điện áp đầu vào(giới hạn)	6-20V
Số chân Digital I/O	14 (có 6 chân điều chế độ rộng xung PWM)
Số chân Analog (Input)	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32KB (ATmega328) với 0.5KB sử dụng bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Xung nhịp	16 MHz

2. Sơ đồ chân của Arduino.



Hình 1: Arduino Uno.

a) USB (1).

Arduino sử dụng cáp USB để giao tiếp với máy tính. Thông qua cáp USB chúng ta có thể Upload chương trình cho Arduino hoạt động, ngoài ra USB còn là nguồn cho Arduino.

b) Nguồn (2 và 3).

Khi không sử dụng USB làm nguồn thì chúng ta có thể sử dụng nguồn ngoài thông qua jack cắm 2.1mm (cực dương ở giữa) hoặc có thể sử dụng 2 chân V_{in} và GND để cấp nguồn cho Arduino.

Bo mạch hoạt động với nguồn ngoài ở điện áp từ 5 – 20 volt. Chúng ta có thể cấp một áp lớn hơn tuy nhiên chân 5V sẽ có mức điện áp lớn hơn 5 volt. Và nếu sử dụng nguồn lớn hơn 12 volt thì sẽ có hiện tượng nóng và làm hỏng bo mạch. Khuyết cáo các bạn nên dùng nguồn ổn định là 5 đến dưới 12 volt.

Chân 5V và chân 3.3V (Output voltage) : các chân này dùng để lấy nguồn ra từ nguồn mà chúng ta đã cung cấp cho Arduino. Lưu ý : không được cấp nguồn vào các chân này vì sẽ làm hỏng Arduino.

GND: chân mass.

c) Chip Atmega328.

Chip Atmega328 Có 32K bộ nhớ flash trong đó 0.5k sử dụng cho bootloader. Ngoài ra còn có 2K SRAM, 1K EEPROM.

d) Input và Output (4, 5 và 6).

Arduino Uno có 14 chân digital với chức năng input và output sử dụng các hàm *pinMode()*, *digitalWrite()* và *digitalRead()* để điều khiển các chân này tôi sẽ đề cập chúng ở các phần sau.

Cũng trên 14 chân digital này chúng ta còn một số chân chức năng đó là:

Serial : chân 0 (Rx), chân 1 (Tx). Hai chân này dùng để truyền (Tx) và nhận (Rx) dữ liệu nối tiếp TTL. Chúng ta có thể sử dụng nó để giao tiếp với cổng COM của một số thiết bị hoặc các linh kiện có chuẩn giao tiếp nối tiếp.

PWM (pulse width modulation): các chân 3, 5, 6, 9, 10, 11 trên bo mạch có dấu “~” là các chân PWM chúng ta có thể sử dụng nó để điều khiển tốc độ động cơ, độ sáng của đèn...

SPI : 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK), các chân này hỗ trợ giao tiếp theo chuẩn SPI.

I2C: Arduino hỗ trợ giao tiếp theo chuẩn I2C. Các chân A4 (SDA) và A5 (SCL) cho phép chúng ta giao tiếp giữa Arduino với các linh kiện có chuẩn giao tiếp là I2C.

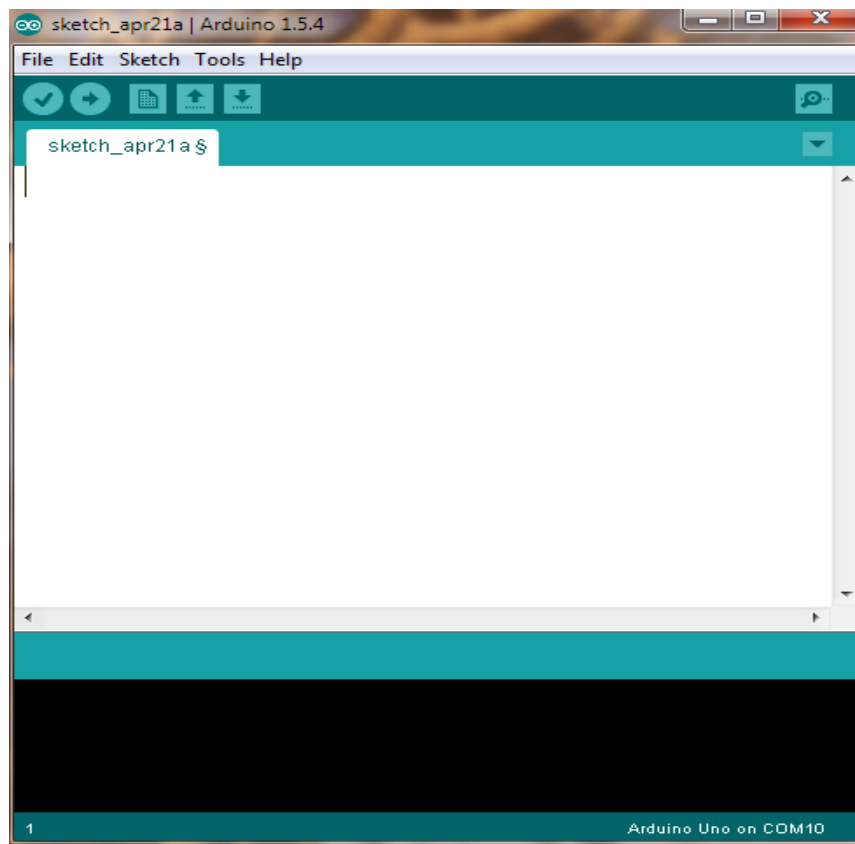
e) Reset (7): dùng để reset Arduino.

Chương 2: Cài đặt chương trình Arduino IDE và Driver cho Arduino

1. Cài đặt chương trình Arduino IDE

Các bạn truy cập vào trang web <http://arduino.cc/en/Main/Software> và tải về chương trình Arduino IDE phù hợp với hệ điều hành của máy mình bao gồm Windows, Mac OS hay Linux. Đối với Windows có bản cài đặt (.exe) và bản Zip, đối với Zip thì chỉ cần giải nén và chạy chương trình không cần cài đặt.

Sau khi cài đặt xong thì giao diện chương trình như sau:



Hình 2: Arduino IDE

2. Cài đặt Driver

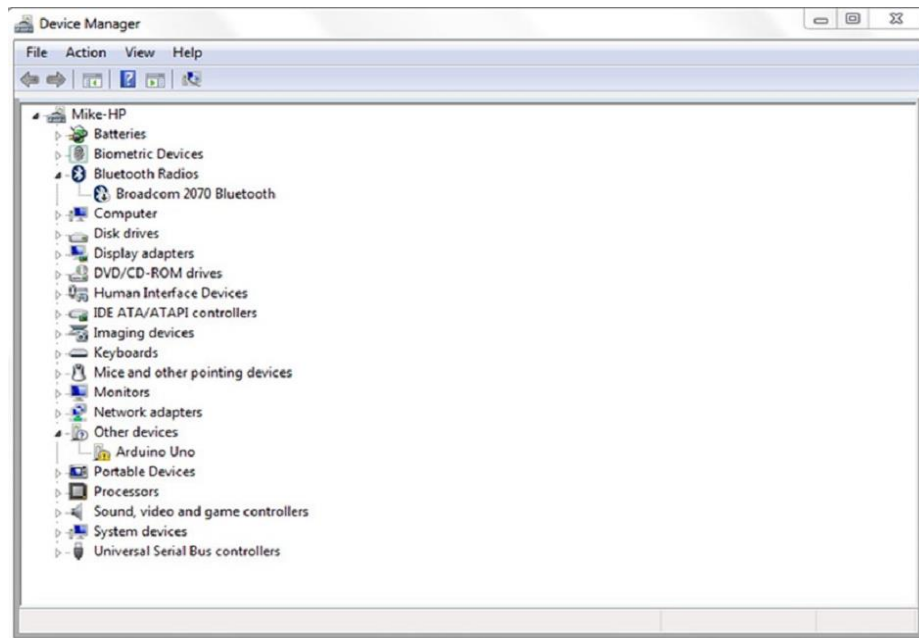
Sử dụng cáp USB kết nối Arduino với máy tính, lúc này bạn sẽ thấy đèn led power của bo sáng. Máy tính sẽ nhận dạng thiết bị và bạn sẽ nhận được thông báo:

“Device driver software was not successfully installed”



Hình 3: Driver Software Installation.

Bây giờ bạn click vào *Start Menu* chọn *Control Panel* kể đến chúng ta chọn *System and Security*, click *System* và sau đó chọn *Device Manager*.



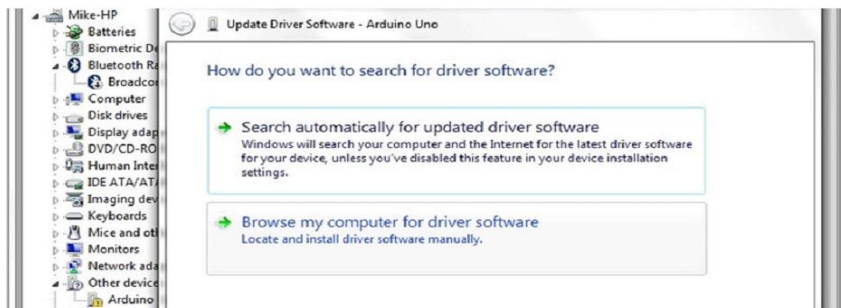
Hình 4: Device Manager.

Chúng ta sẽ thấy cảnh báo màu vàng thiếu driver trên Arduino. Click chuột phải trên Arduino Uno icon sau đó chọn “Update Driver Software”



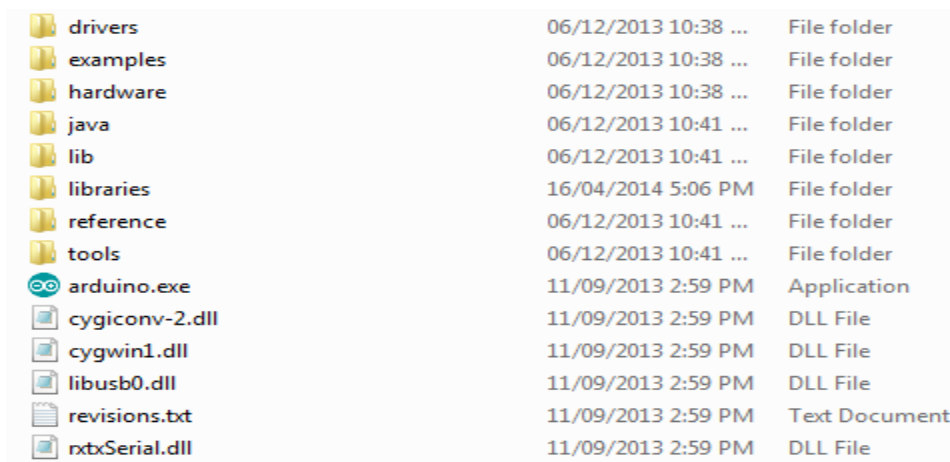
Hình 5: Right click và chọn ”Update Driver Software”

Chọn “Browse my computer for driver software”.



Hình 6: Click chọn “Browse my computer for driver software”

Chọn đường dẫn tới folder “driver” nơi mà phần mềm Arduino được lưu trữ.

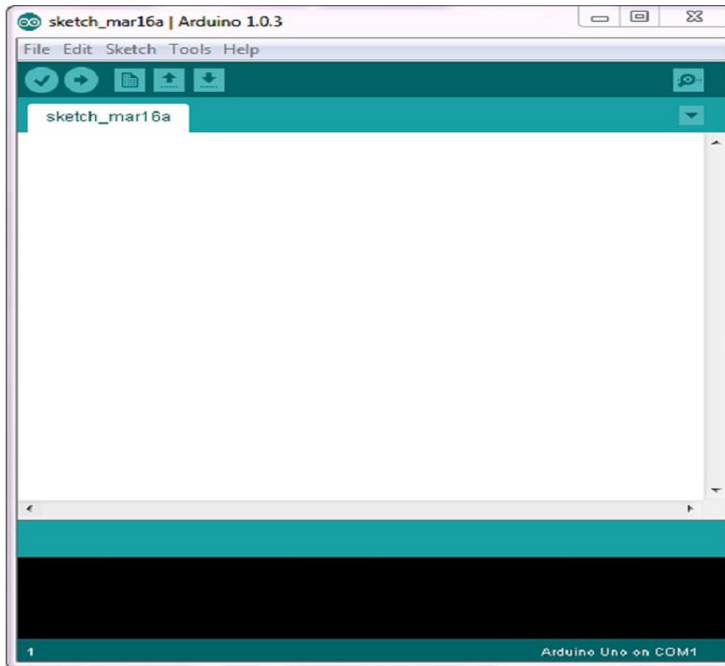


Hình 6: Driver

Click “Next” Windown tự động cài đặt driver, qua trình cài đặt driver hoàn tất.

3. Arduino IDE

Arduino IDE là nơi để soạn thảo code, kiểm tra lỗi và upload code cho arduino







Hình 7: Arduino IDE.

a) Arduino Toolbar: có một số button và chức năng của chúng như sau :



Hình 8: Arduino Toolbar.

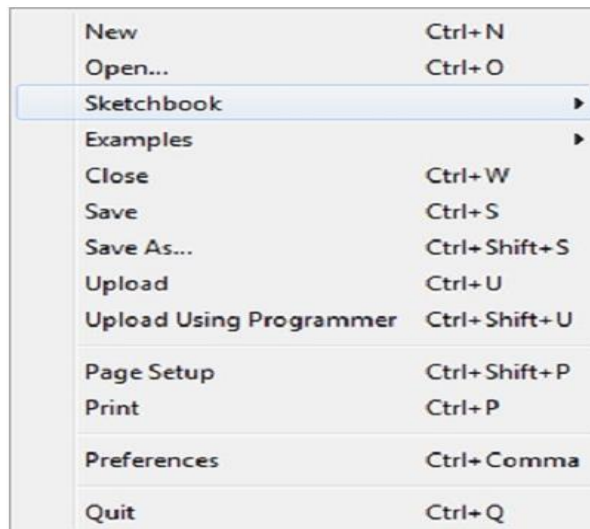
- Verify : kiểm tra code có lỗi hay không 
- Upload: nạp code đang soạn thảo vào Arduino 
- New, Open, Save : Tạo mới, mở và Save sketch 
- Serial Monitor : Đây là màn hình hiển thị dữ liệu từ Arduino gửi lên máy tính 

b) Arduino IDE Menu:



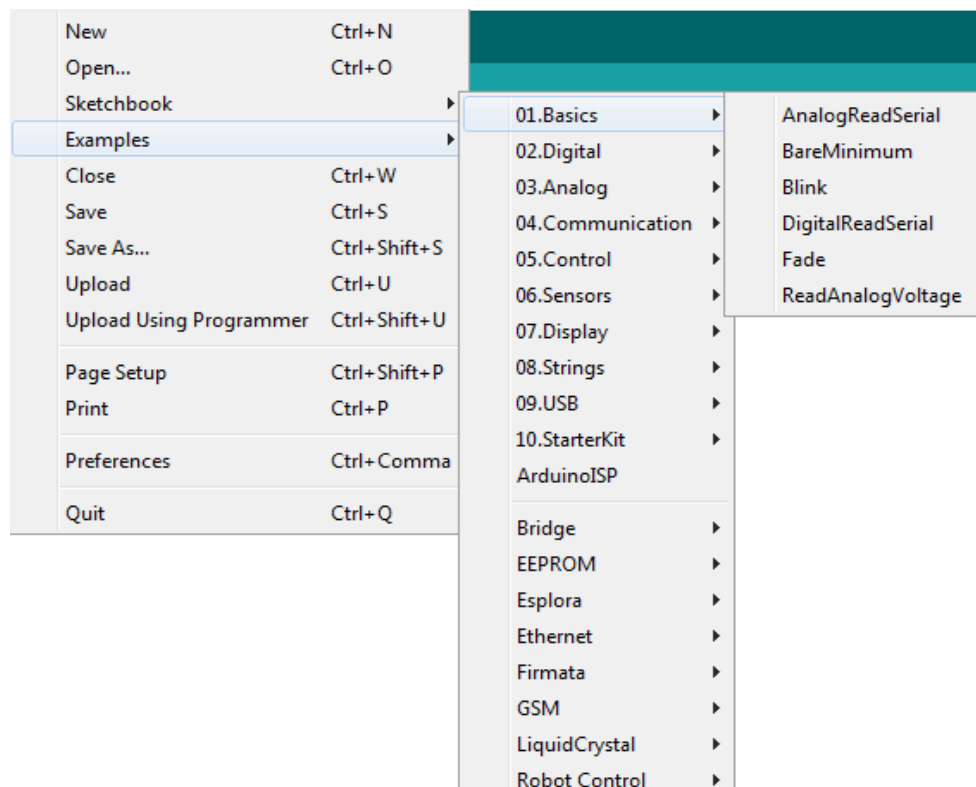
Hình 9: IDE Menu

➤ File menu:



Hình 10: File menu.

Trong file menu chúng ta quan tâm tới mục Examples đây là nơi chứa code mẫu ví dụ như: cách sử dụng các chân digital, analog, sensor ...



Hình 11: Click Examples.

➤ Edit menu:

Undo	Ctrl+Z
Redo	Ctrl+Y
<hr/>	
Cut	Ctrl+X
Copy	Ctrl+C
Copy for Forum	Ctrl+Shift+C
Copy as HTML	Ctrl+Alt+C
Paste	Ctrl+V
Select All	Ctrl+A
<hr/>	
Comment/Uncomment	Ctrl+Slash
Increase Indent	Ctrl+Close Bracket
Decrease Indent	Ctrl+Open Bracket
<hr/>	
Find...	Ctrl+F
Find Next	Ctrl+G
Find Previous	Ctrl+Shift+G
Use Selection For Find	Ctrl+E

Hình 11: Edit menu

➤ Sketch menu

Verify / Compile	Ctrl+R
<hr/>	
Show Sketch Folder	Ctrl+K
Add File...	
Import Library...	▶

Hình 12: Sketch menu

Trong Sketch menu :

- ✓ Verify/ Compile : chức năng kiểm tra lỗi code.
- ✓ Show Sketch Folder : hiển thị nơi code được lưu.
- ✓ Add File : thêm vào một Tập code mới.
- ✓ Import Library : thêm thư viện cho IDE

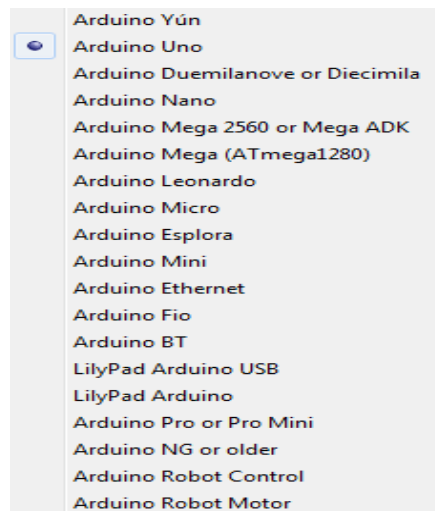
➤ Tool menu:



Hình 13: Tool menu.

Trong Tool menu ta quan tâm các mục Board và Serial Port

Mục Board : các bạn cần phải lựa chọn bo mạch cho phù hợp với loại bo mà bạn sử dụng nếu là Arduino Uno thì phải chọn như hình:



Hình 14: Chọn Board

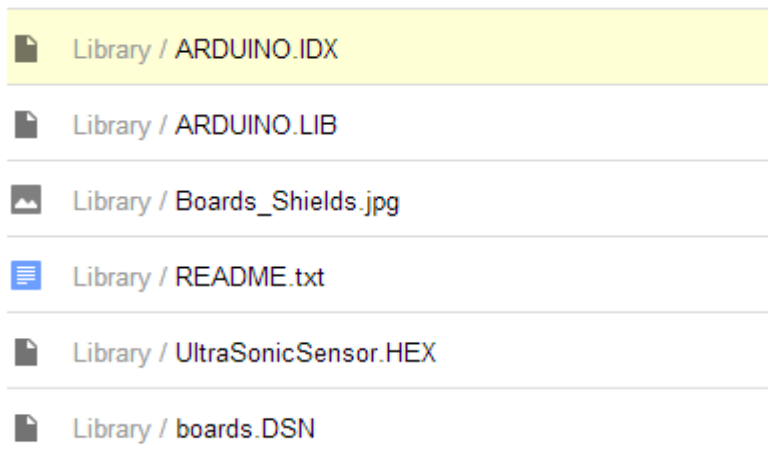
Nếu các bạn sử dụng loại bo khác thì phải chọn đúng loại bo mà mình đang có nếu sai thì code Upload vào chip sẽ bị lỗi.

Serial Port: đây là nơi lựa chọn cổng Com của Arduino. Khi chúng ta cài đặt driver thì máy tính sẽ hiện thông báo tên cổng Com của Arduino là bao nhiêu, ta chỉ việc vào Serial Port chọn đúng cổng Com để nạp code, nếu chọn sai thì không thể nạp code cho Arduino được.

Chương 3: Hướng dẫn cài đặt bản mô phỏng Arduino trên Proteus.

Để mô phỏng được Arduino trên proteus thì chúng ta cần phải download thư viện arduino cho proteus. Để có được thư viện này các bạn cần truy cập vào trang web:

<http://blogembarcado.blogspot.com/search/label/Proteus>



Hình 15: Thư viện mô phỏng Arduino.

Sau khi download về các bạn chép 2 file ARDUINO.IDX và ARDUINO.LIB vào thư mục:

Proteus 7:

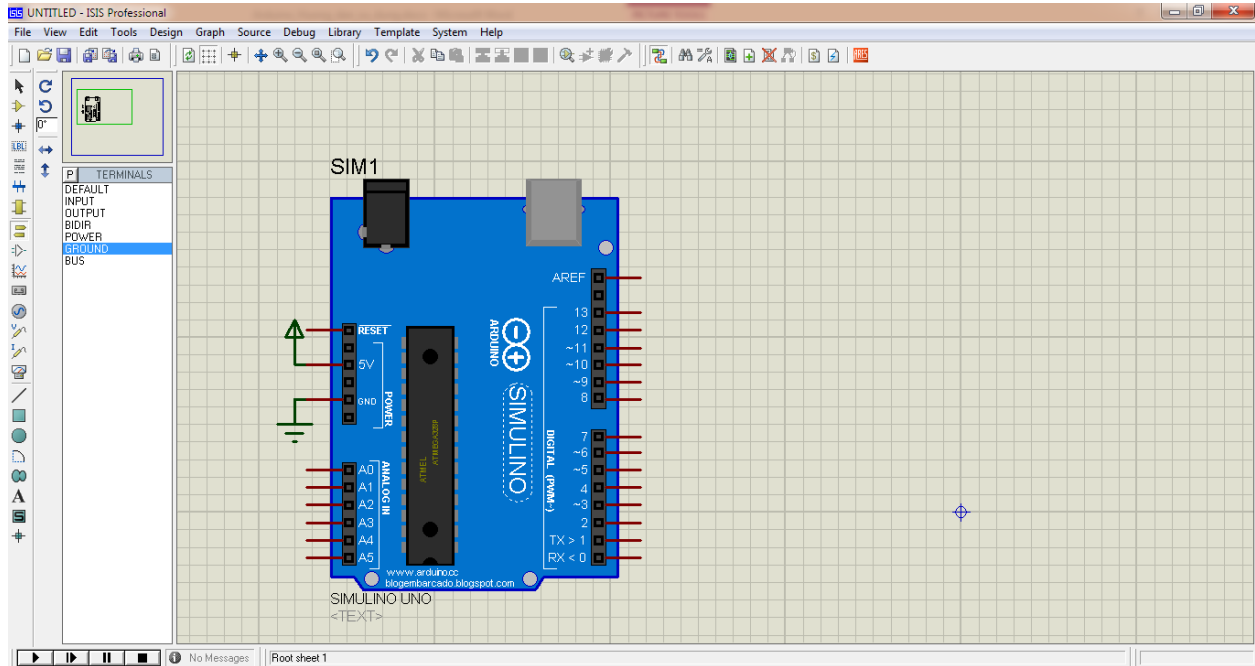
C:\Program Files (hoặc x86) \Labcenter Electronics\Proteus 7 Professional\LIBRARY

Proteus 8:

C:\Program Files (hoặc x86) \Labcenter Electronics\Proteus 8 professional\Data\LIBRARY

Trong thư viện này hỗ trợ 5 loại board Arduino khác nhau trong đó gồm có Arduino Uno, MEGA, NANO, LILYPAD và UNO SMD và một cảm biến siêu âm Untrasonic.

Sau khi chép xong chúng ta khởi động Proteus lên vào thư viện linh kiện bằng cách bấm phím P và gõ từ khoá là ARDUINO chúng sẽ hiện ra danh sách các board hiện có ở đây tôi chọn Arduino Uno.



Hình 16: Mô phỏng Arduino bằng Proteus.

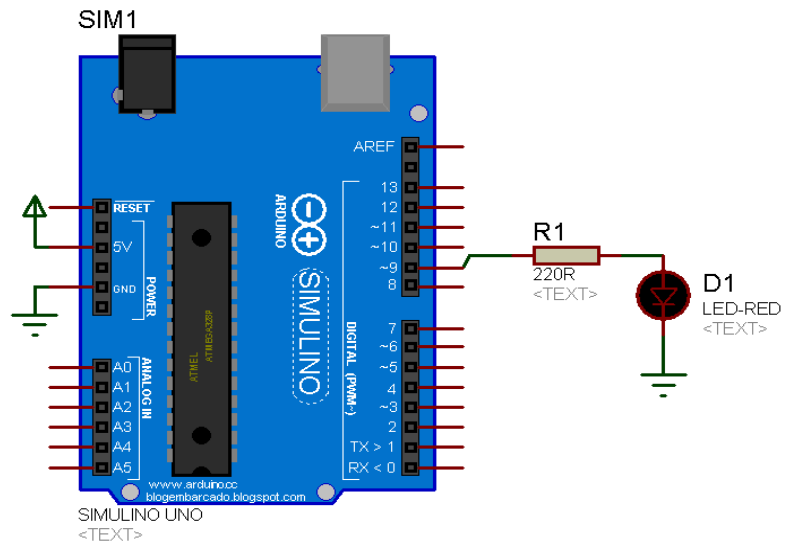
Lưu ý chúng ta cần phải cấp nguồn vào 2 chân 5V và Gnd trên mạch như hình trên.

Chương 4: Giao tiếp Arduino với một số linh kiện điện tử.

1) Project 1: Led nhấp nháy.

Sau đây tôi sẽ tạo một project nhấp nháy led thời gian delay là 1 giây sử dụng proteus để mô phỏng.

✓ Sơ đồ mạch:




Hình 17: Led nhấp nháy.

✓ Code chương trình.

```
int ledPin = 9;

void setup() {
    pinMode(ledPin, OUTPUT);
}

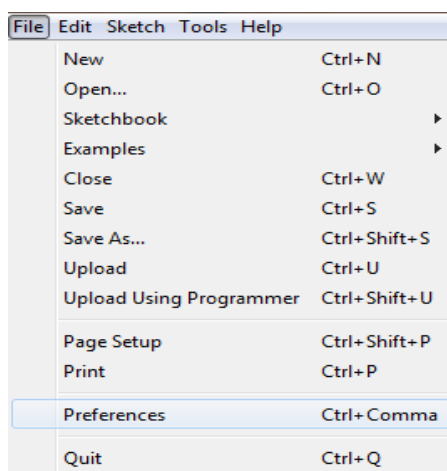
void loop() {
    digitalWrite(ledPin, HIGH);
    delay(1000);
    digitalWrite(ledPin, LOW);
    delay(1000);
}
```

Sau khi gõ code vào chương trình soạn thảo bạn cần click và  để kiểm tra lỗi.

Tạo File Hex.

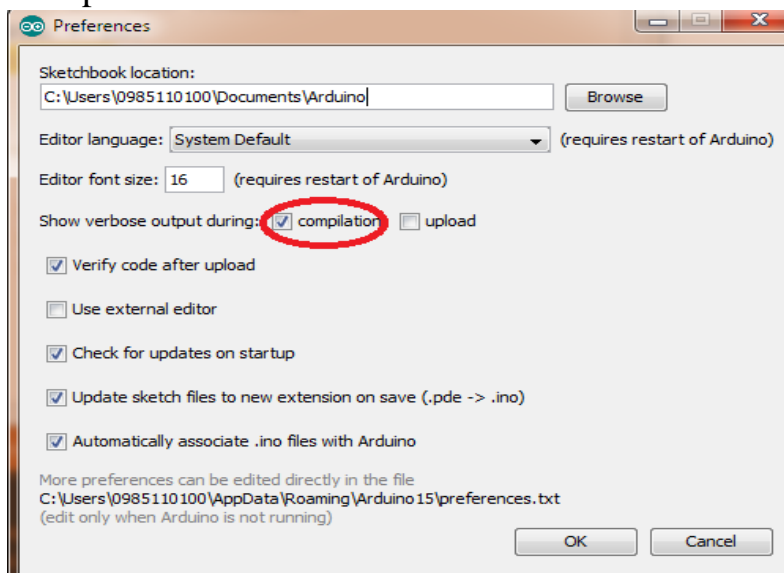
Chúng ta cần phải có file Hex để cung cấp cho proteus và khi bấm play chương trình mới hoạt động được. Cách tạo file Hex trên Arduino IDE như sau:

Click vào File chọn Preferences.




Hình 18: Click Preferences.

Các bạn check vào compilation và OK.



Hình 19: Check compilation.

Sau đó tiếp tục bấm 

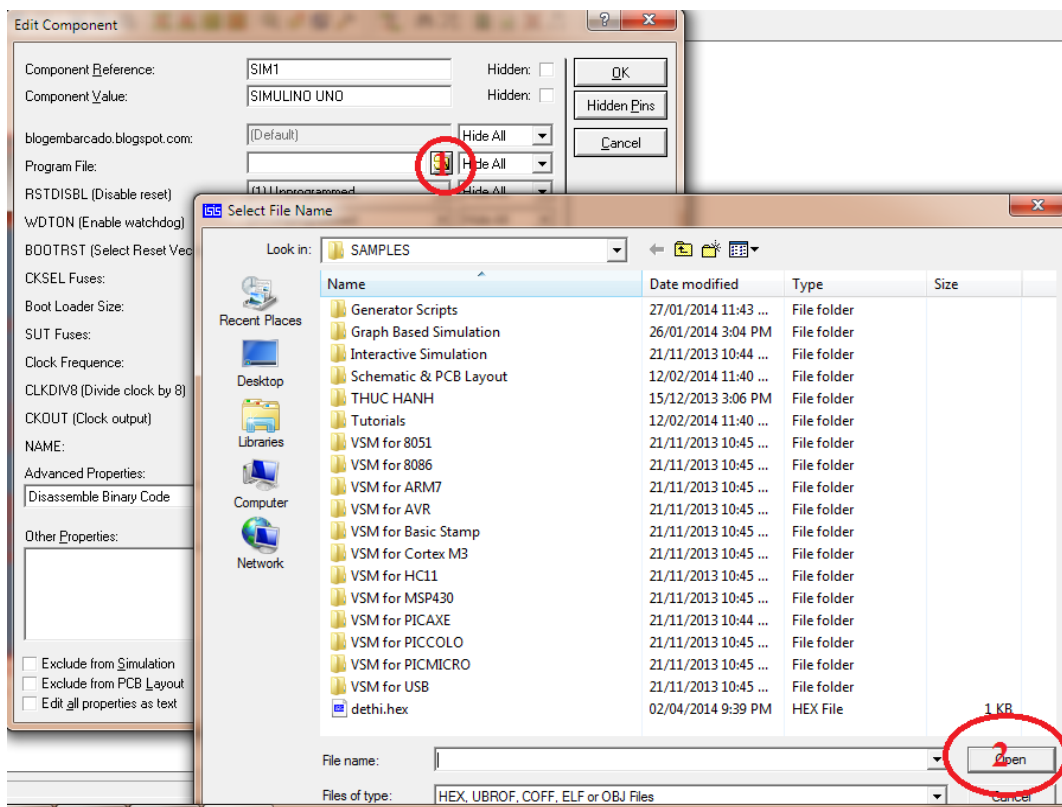
Chương trình sẽ tự động build một file hex được lưu ở đường dẫn như hình dưới

```
C:\Users\098511~1\AppData\Local\Temp\build9118448727674380763.tmp/Blink.cpp.hex

Sketch uses 1,116 bytes (3%) of program storage space. Maximum is 32,256 bytes.
Global variables use 11 bytes (0%) of dynamic memory, leaving 2,037 bytes for local variables. Maximum is 2,048 bytes.
```

Hình 20: Đường dẫn chứa file hex.

Các bạn chép file hex ra một thư mục nào đó sau đó mở proteus lên và double click vào Aruino Uno.



Hình 21: Add file Hex cho Proteus.

Bấm vào vị trí số 1 và chọn nơi lưu file hex ở trên chọn tiếp Open, OK và Play.

Các bạn sẽ thấy led nhấp nháy tắt và sáng thời gian delay là 1s.

- ✓ Giải thích chương trình.

```
int ledPin = 9;
```

Khai báo một giá trị biến integer là *ledPin = 9*.

```
void setup() {  
  
pinMode(ledPin, OUTPUT); }
```

Trong Arduino sketch cần phải có hàm *setup()* và *loop()* nếu không có thì chương trình báo lỗi. Hàm *Setup()* chỉ chạy một lần kể từ khi bắt đầu chương trình. Hàm này có chức năng thiết lập chế độ vào, ra cho các chân digital hay tốc độ baud cho giao tiếp Serial...

Cấu trúc của hàm *pinMode()* là như sau:

```
pinMode(pin, Mode);
```

pin : là vị trí chân digital.

Mode: là chế độ vào (INPUT), ra (OUTPUT).

Lệnh tiếp theo.

```
pinMode(ledPin, OUTPUT);
```

Lệnh này thiết lập chân số 9 trên board là chân ngõ ra (*OUTPUT*). Nếu không khai báo “ *int ledPin = 9;* ” thì bạn có thể viết cách sau nhưng ý nghĩa không thay đổi:

```
pinMode(9, OUTPUT);
```

Bắt buộc khai báo một hàm *loop()* trong Arduino IDE. Hàm này là vòng lặp vô hạn

```
void loop() {  
  
digitalWrite(ledPin, HIGH);  
  
delay(1000);  
  
digitalWrite(ledPin, LOW);
```

```
    delay(1000);  
}
```

Tiếp theo ta sẽ phân tích hàm *digitalWrite(ledPin, HIGH)*; lệnh này có ý nghĩa là xuất ra chân digital có tên là *ledPin* (chân 9) mức cao (*HIGH*), mức cao tương ứng là 5 volt.

```
    delay(1000);
```

Lệnh này tạo một khoảng trễ với thời gian là 1 giây. Trong hàm *delay()* của IDE thì 1000 tương ứng với 1 giây.

```
    digitalWrite(ledPin, LOW);
```

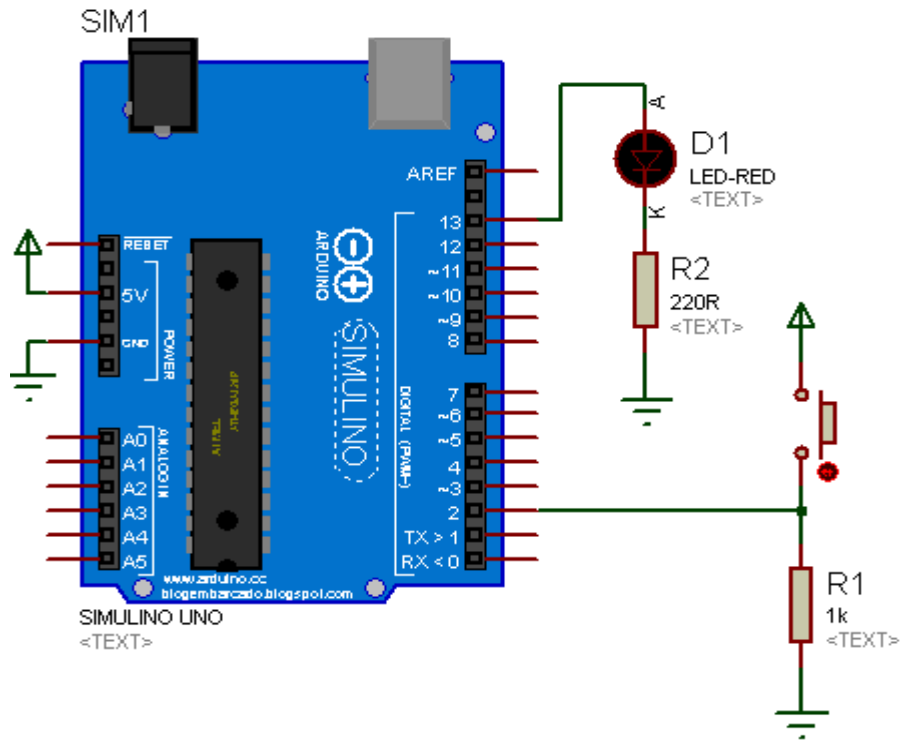
Cũng giống như *digitalWrite(ledPin, HIGH)*; lệnh này xuất ra chân *ledPin* mức thấp (*LOW*) tức là 0 volt.

Và tiếp tục là một hàm *delay()*.

Như vậy chúng ta có thể thấy chương trình sẽ thực hiện tắt sáng led liên tục không ngừng trừ khi ta ngắt nguồn.

2) Project 2 : Đèn sáng khi nhấn phím.

✓ Sơ đồ mạch:



Hình 22: Đèn sáng khi nhấn phím

✓ Code chương trình :

```
const int buttonPin = 2;
const int ledPin = 13;
int buttonState = LOW;
void setup() {
    pinMode(ledPin, OUTPUT);
    pinMode(buttonPin, INPUT); }
void loop(){
    buttonState = digitalRead(buttonPin);
    if (buttonState == HIGH) {
```

```

        digitalWrite(ledPin, HIGH);    }
    else {
        digitalWrite(ledPin, LOW);
    }
}

```

✓ Giải thích chương trình :

Trước tiên ta khai báo hai biến để lưu trữ vị trí chân của phím nhấn và led :

```

const int buttonPin = 2;

const int ledPin = 13;

```

Phím nhấn sẽ ở vị trí chân số 2 và led chân số 13.

Ta khai báo một biến trạng thái của phím nhấn là *int buttonState = LOW*;

Trong hàm *setup()* là khai chế độ (Mode) cho chân button và chân led. Chân button là chân ngõ vào và chân led là chân ngõ ra.

```

pinMode(ledPin, OUTPUT);

pinMode(buttonPin, INPUT);

```

Trong hàm *loop()* ta có câu lệnh đầu tiên là :

```

buttonState = digitalRead(buttonPin);

```

Câu lệnh này có nghĩa là gán giá trị đọc được từ chân button (chân 2) cho biến *buttonState*.. *buttonState* sẽ có giá trị 0 nếu như button không được nhấn và có giá trị 1 nếu được nhấn. Bằng cách sử dụng hàm *digitalRead()* ta có thể kiểm tra được các chân digital đang ở mức cao hay thấp.

Sau khi đọc được giá trị có ở chân *buttonPin* (chân 2) ta kiểm tra xem là button có nhấn hay không.

Nếu có tức là *buttonState = HIGH* thì lúc này ta bật led bằng lệnh *digitalWrite()*

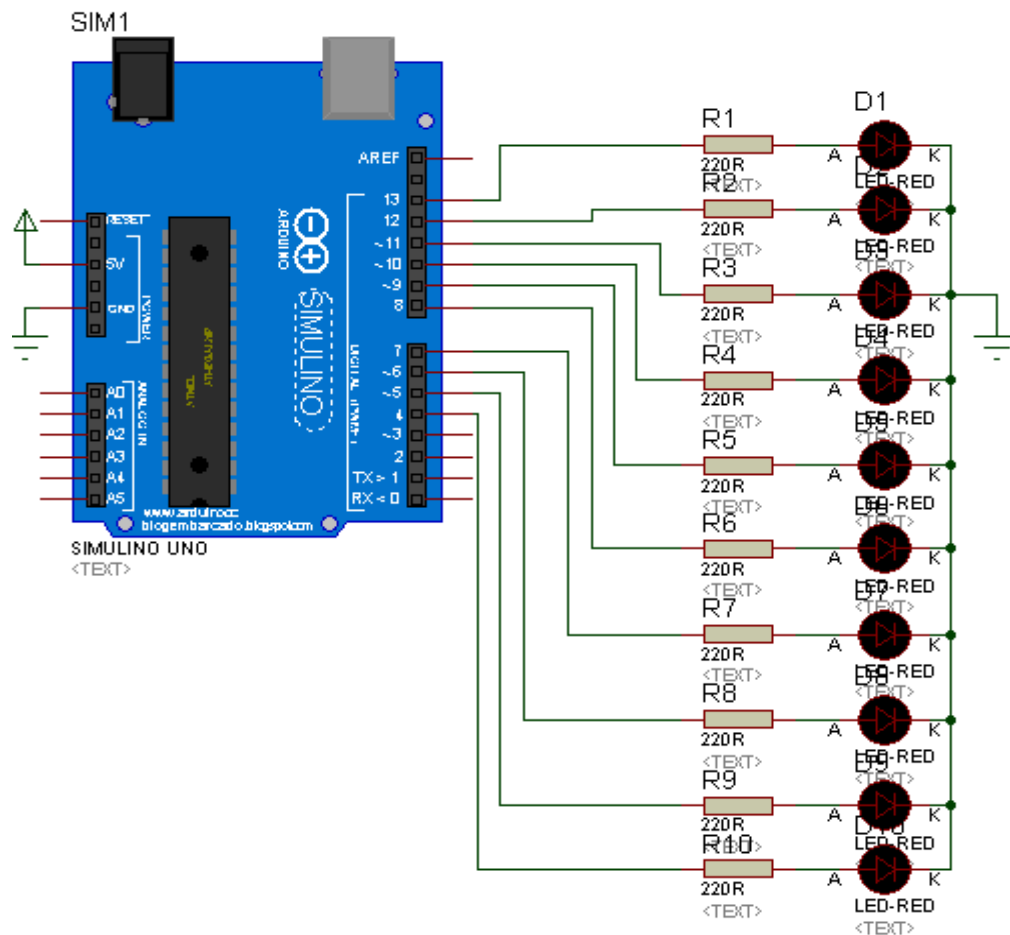
```
if (buttonState == HIGH) {  
    digitalWrite(ledPin, HIGH); }  
}
```

Ngược lại thì ta một lần nữa sử dụng hàm *digitalWrite()* để tắt led

```
else {  
    digitalWrite(ledPin, LOW);  
}
```

3) Project 3 : Led sáng dần từ led 1 đến led 10 và ngược lại.

✓ Sơ đồ mạch.



Hình 23: Led sáng dần từ led 1 đến led 10 và ngược lại.

✓ Code chương trình.

```
byte ledPin[] = {4, 5, 6, 7, 8, 9, 10, 11, 12, 13};  
int direction = 1;  
int currentLED = 0;  
void setup() {  
    for (int x=0; x<10; x++) {  
        pinMode(ledPin[x], OUTPUT);} }  
}
```



```

void loop() {
    for (int x=0; x<10; x++) {
        digitalWrite(ledPin[x], LOW);
    }
    digitalWrite(ledPin[currentLED], HIGH);
    currentLED += direction;

    if (currentLED == 9) {direction = -1;}
    if (currentLED == 0) {direction = 1;}
    delay(500);
}

```

✓ Giải thích chương trình.

Trong Project này chúng ta sử dụng 10 chân digital để điều khiển 10 led, để cho chương trình ngắn gọn thì ở đây tôi sử dụng mảng 1 chiều gồm 10 phần tử trong đó chứa 10 vị trí chân led mà ta sử dụng trong project

```
byte ledPin[] = {4, 5, 6, 7, 8, 9, 10, 11, 12, 13};
```

Tiếp tục khai báo 2 biến integer là :

```
int direction = 1;
```

```
int currentLED = 0;
```

Trong hàm *setup()* tôi sử dụng một vòng lặp để định nghĩa mode cho các chân led. Tôi nghĩ là không khó để hiểu được các câu lệnh này.

Tiếp theo là hàm *loop()*, đầu tiên tôi tắt tất cả các led bằng các câu lệnh:

```

for (int x=0; x<10; x++) {
    digitalWrite(ledPin[x], LOW);}

```

Sau đó cho sáng led đầu tiên bằng câu lệnh :

```
digitalWrite(ledPin[currentLED], HIGH);
```

Vì ta đã khai báo $currentLED = 0$ nên mãng sẽ truy xuất phần tử đầu tiên trong mảng có giá trị là 4 vì thế led ở vị trí chân digital số 4 sẽ sáng.

```
currentLED += direction;
```

Tăng $currentLED$ lên 1 đơn vị ($direction = 1$). Vòng lặp tiếp theo sẽ là led ở chân digital 5 sáng và cứ như thế cho đến led ở chân số 13 sáng, thì lúc này $currentLED == 9$, câu lệnh “ $if (currentLED == 9) \{direction = -1;\}$ ” sẽ thực hiện và led sẽ sáng ngược lại từ led 10 xuống led thứ 1.

Hai câu lệnh :

```
if (currentLED == 9) \{direction = -1;\}
```

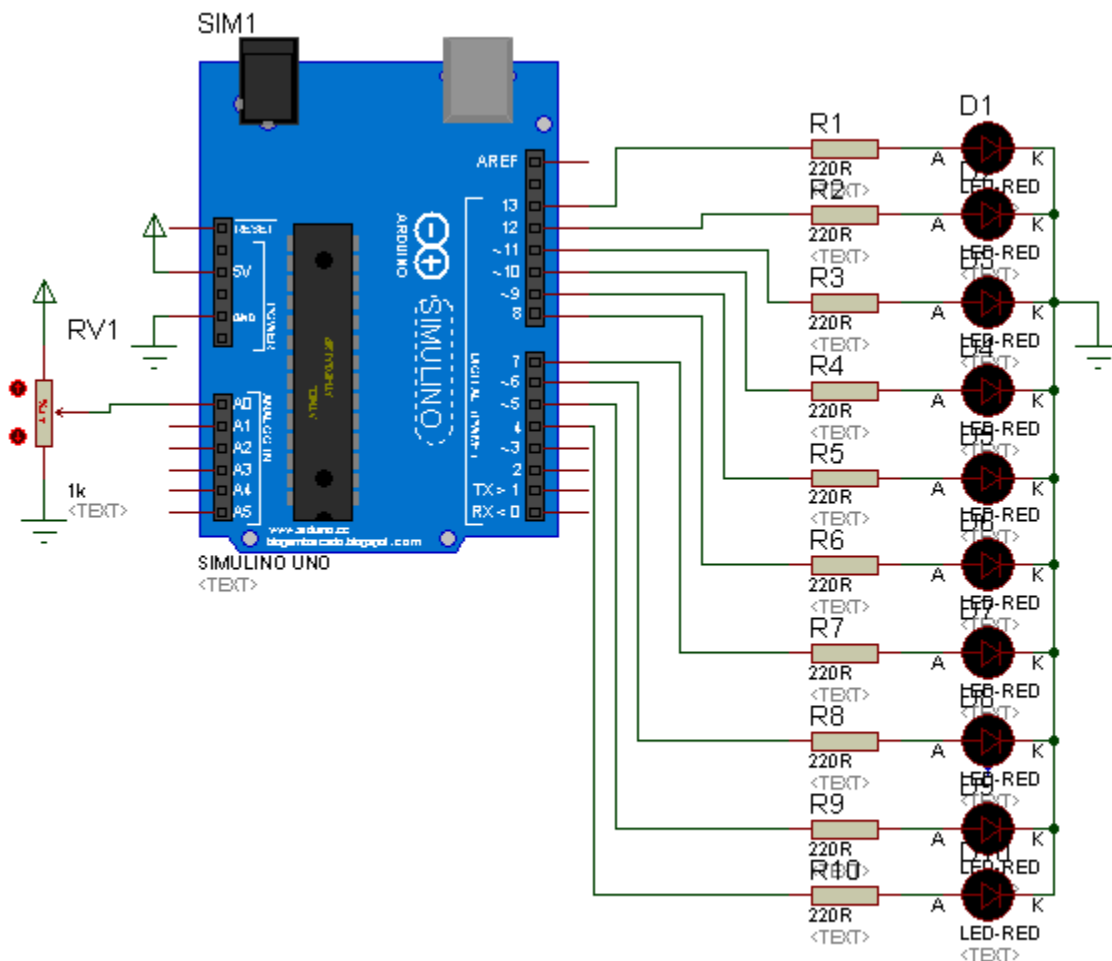
```
if (currentLED == 0) \{direction = 1;\}
```

dùng để quy định chiều sáng của led là tăng dần hay giảm dần. Nếu là Led thứ 10 sáng thì tiếp theo sẽ giảm xuống led thứ 9 và ngược lại nếu led thứ 0 sáng thì chu kỳ tiếp theo led 1 sẽ sáng.

4) Project 4 : Led sáng dần từ led 1 đến led 10 và ngược lại thời gian delay thay đổi được.

✓ Sơ đồ mạch :

Trong project này hoàn toàn giống project 3 chỉ thêm một biến trở dùng để điều chỉnh thời gian delay cho chương trình



Hình 24: Led sáng dần từ led 1 đến led 10 thời gian delay thay đổi được.

✓ Code chương trình.

```
int ledPin[] = {4, 5, 6, 7, 8, 9, 10, 11, 12, 13};  
int direction = 1;
```

```

int currentLED = 0;
int potPin = 0;
unsigned long changeTime;
void setup() {
    for (int x=0; x<10; x++) {
        pinMode(ledPin[x], OUTPUT);
    }
}
void loop() {
    int delayvalu= analogRead(potPin);
    for (int x=0; x<10; x++) {
        digitalWrite(ledPin[x], LOW);
    }
    digitalWrite(ledPin[currentLED], HIGH);
    currentLED += direction;

    if (currentLED == 9) {direction = -1;}
    if (currentLED == 0) {direction = 1;}
    delay(delayvalu);
}

```

✓ Giải thích chương trình.

Chương trình ta chỉ thêm và thay đổi một vài câu lệnh mà thôi ngoài ra không khác gì nhiều so với project 3, các câu lệnh đó như sau :

```

int potPin = 0 ;

int delayvalu= analogRead(potPin);

```

```
delay(delayvalu);
```

Đầu tiên chúng ta khai báo một biến chứa vị trí chân biến trở kết nối đó là vị trí A0

```
int potPin = 0;
```

Đọc giá trị từ chân analog A0 bằng câu lệnh *analogRead(potPin)* và gán nó cho biến *delayvalu*. Arduino có 6 chân đầu vào analog đánh dấu từ A0 đến A5 với 10 bit chuyển đổi từ analog sang digital (ADC). Nghĩa là chân analog có thể đọc được các giá trị điện áp từ 0 đến 5 volt tương ứng với các số integer từ 0 (0 volt) đến 1023 (5 volt).

Trong project này chúng ta cần thiết lập thời gian delay bằng cách điều chỉnh biến trở. Ta sử dụng câu lệnh *delay(delayvalu)* để tạo thời gian trễ. Nếu ta điều chỉnh biến trở sao cho điện áp đầu vào chân analog là 5 volt thì *delayvalu* sẽ có giá trị là 1023 (hơn 1 giây), nếu là 2,5 volt thì *delayvalu* sẽ là 511. Các bạn thử điều chỉnh biến trở ta sẽ thấy thời gian delay thay đổi hoặc là nhanh dần hoặc là chậm dần.

Lưu ý : đối với các chân analog chúng ta không cần thiết lập chế độ vào ra bằng hàm *pinMode* như các chân digital. Mặc định các chân analog là input.

✓ Giải thích chương trình.

Như trong sơ đồ mạch ta thấy biến trở được nối với chân A0, transistor được nối với chân số 9 thông qua điện trở 1k. Như vậy ta khai báo 2 biến chứa vị trí chân cho biến trở và transistor.

```
int potPin = 0;
```

```
int transistorPin = 9;
```

Biến `integer potValue` chứa giá trị đọc được từ chân A0.

```
int potValue = 0
```

Một câu hỏi đặt ra ở đây là nếu chúng ta không kết nối transistor điều khiển động cơ vào chân số 9 mà thay vào đó là chân số 1 hoặc 2 để điều khiển tốc độ động cơ thì có được không ?.

Câu trả lời là **Không**. Vậy tại sao Không ?

Tôi sẽ trả lời câu hỏi này sau. Nhưng trước hết tôi nói về PWM. PWM (pulse width modulation) là phương pháp điều chế dựa trên sự thay đổi độ rộng của chuỗi xung dẫn đến sự thay đổi điện áp ra.

Để tạo ra được PWM trên Arduino thì chúng ta sử dụng lệnh `analogWrite(Pin, Value);`

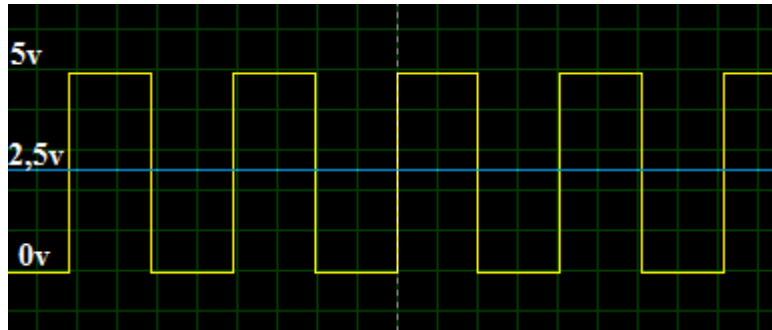
Trong đó:

`Pin`: là vị trí chân, đối với Arduino Uno thì chỉ có các chân 3, 5, 6, 9, 10 & 11 mới có chức năng tạo PWM. Vậy chúng ta có thể trả lời được câu hỏi bên trên, các chân digital còn lại của có thể đọc hoặc xuất 2 giá trị là 0 và 1 mà thôi.

`Value`: Giá trị nằm trong khoảng 0 đến 255.

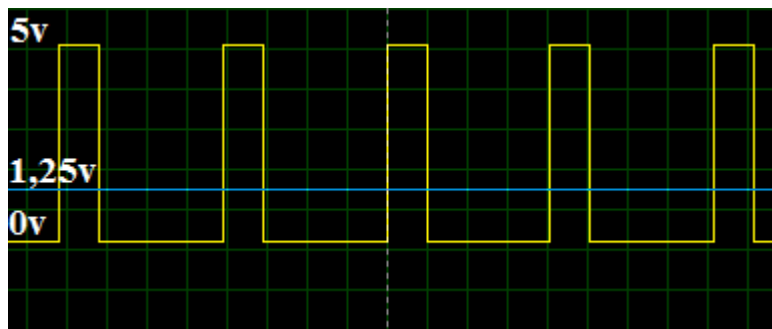
Để hiểu rõ hơn về PWM tôi sẽ minh họa qua ví dụ sau.

Nếu tôi sử dụng lệnh `analogWrite(transistorPin, 127);` thì dạng xung ở chân 9 (`transistorPin = 9`) sẽ như hình dưới và giá trị trung bình ngõ ra sẽ là 2,5V (50%).



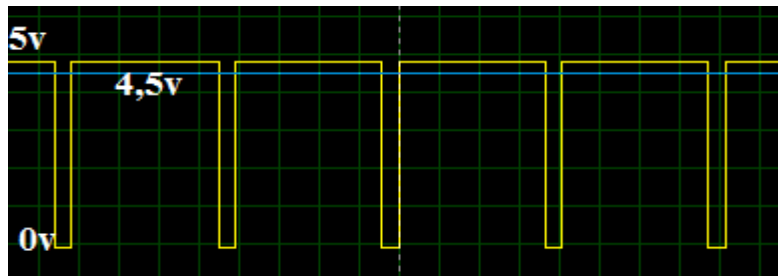
Hình 26: PWM 50%.

Nếu $Value = 64$ (hay 25%) thì dạng xung như sau:



Hình 27: PWM 25%.

$Value = 229$ (hay 90%) thì dạng xung sẽ là :



Hình 28: PWM 90%.

Từ ví dụ trên ta thấy sự thay đổi độ rộng của chuỗi xung dẫn đến sự thay đổi điện áp ra. Ta cũng thấy rằng điện áp trên motor cũng thay đổi tuyến tính theo sự thay đổi điện áp ngõ ra trên chân 9. Tức là nếu điện áp trung bình trên chân 9 là 2,5 volt (50%) thì điện áp trên hai đầu motor là 6 volt (nguồn motor là 12 volt).

Quay lại chương trình ta cần quan tâm tới một câu lệnh đó là:

$$potValue = analogRead(potPin) / 4;$$

Chúng ta cần phải chia giá trị đọc được cho 4 vì giá trị analog sẽ nằm trong khoảng 0 (0 volt) đến 1023 (5 volt), nhưng giá trị cần xuất ra ngoài chân 9 lại nằm trong khoảng 0 đến 255 đó chính là lý do tại sao có chia 4 ở đây.

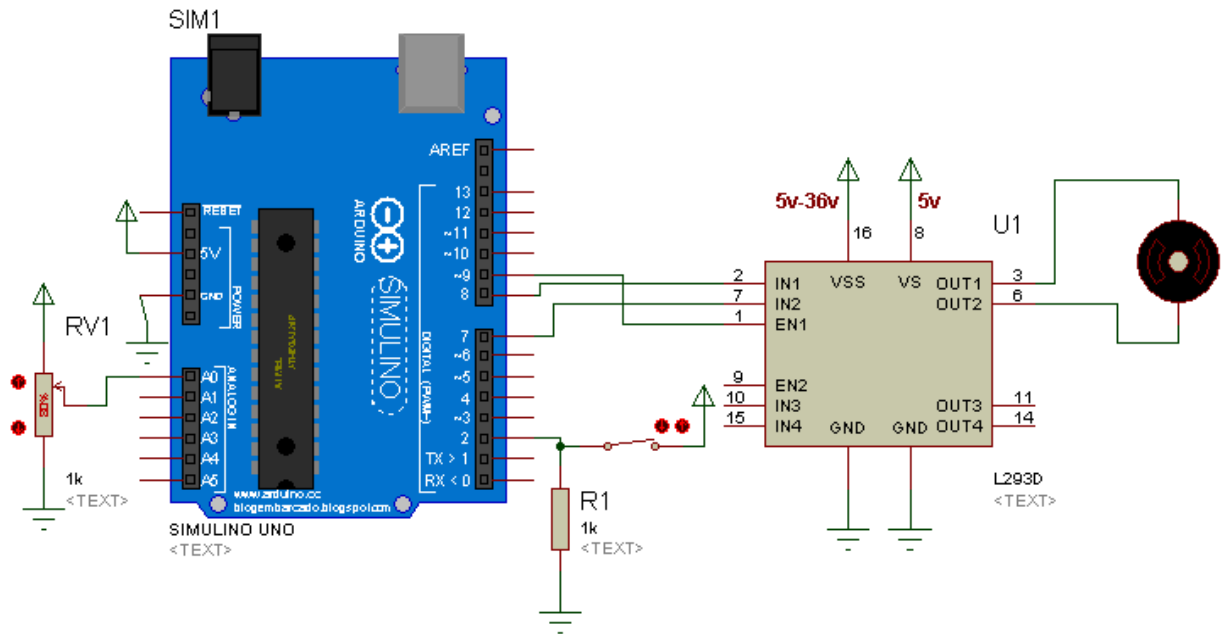
Và câu lệnh cuối cùng là tạo PWM trên chân 9 để điều khiển tốc độ động cơ.

```
analogWrite(transistorPin, potValue);
```

Nếu như đã điều khiển được tốc độ động cơ bằng PWM rồi thì việc điều khiển độ sáng của Led hay đèn đối với các bạn bây giờ là chuyện quá đơn giản. Các bạn chỉ cần nối chân số 9 với một Led có điện trở hạn dòng là 220 ohm và code chương trình hoàn toàn giống như điều khiển động cơ. Lưu ý là khi mô phỏng các bạn sẽ không thấy được led thay đổi độ sáng mà chỉ thấy nhấp nháy nguyên nhân là do phần mềm proteus không đáp ứng kịp sự thay đổi của các xung PWM. Nhưng khi làm thực tế các bạn sẽ thấy được sự thay đổi độ sáng của Led rõ rệt.

6) Project 6 : Điều khiển động cơ bằng L293D.

✓ Sơ đồ mạch:



Hình 29: Điều khiển động cơ bằng L293D.

✓ Code chương trình

```
#define switchPin 2 // chân công tắc
#define motorPin1 8 // L293D Input 1
#define motorPin2 7 // L293D Input 2
#define speedPin 9 // L293D enable chân 1
#define potPin 0 // chân biến trở nối với A0
int Mspeed = 0;
void setup() {
    pinMode(switchPin, INPUT);
    pinMode(motorPin1, OUTPUT);
    pinMode(motorPin2, OUTPUT);
    pinMode(speedPin, OUTPUT);
}
```

```

    }

void loop() {
    Mspeed = analogRead(potPin)/4;
    analogWrite (speedPin, Mspeed);
    if (digitalRead(switchPin)) {
        digitalWrite(motorPin1, LOW);
        digitalWrite(motorPin2, HIGH); }
    else {
        digitalWrite(motorPin1, HIGH);
        digitalWrite(motorPin2, LOW);
    }
}
}

```

✓ Giải thích chương trình:

Code của project này hoàn toàn đơn giản. Trước tiên ta định nghĩa các chân sẽ sử dụng trên arduino.

```

#define switchPin 2 // chân công tắc
#define motorPin1 8 // L293D Input 1
#define motorPin2 7 // L293D Input 2
#define speedPin 9 // L293D enable chân 1
#define potPin 0 // chân biến trở nối với A0

```

Khai báo một biến chứa tốc độ đọc từ biến trở.

```
int Mspeed = 0;
```

Tiếp theo trong *setup()* ta thiết lập chế độ vào, ra cho các chân vừa định nghĩa.

```

pinMode(switchPin, INPUT);
pinMode(motorPin1, OUTPUT);
pinMode(motorPin2, OUTPUT);

```

```
pinMode(speedPin, OUTPUT)
```

Trong vòng `loop()` chúng ta đọc giá trị từ biến trở kết nối với chân A0 và gán nó cho *Mspeed* :

```
Mspeed = analogRead(potPin)/4;
```

Thiết lập tốc độ cho động cơ bằng câu lệnh:

```
analogWrite (speedPin, Mspeed);
```

Kiểm tra xem công tắc có được bật hay không, nếu có thì thiết lập *motorPin1 = LOW* và *motorPin2 = HIGH* ta sẽ thấy động cơ quay ngược chiều kim đồng hồ.

```
if (digitalRead(switchPin)) {
```

```
digitalWrite(motorPin1, LOW);
```

```
digitalWrite(motorPin2, HIGH); }
```

và nếu công tắc không được bật thì motor sẽ quay cùng chiều kim đồng hồ:

```
else {
```

```
digitalWrite(motorPin1, HIGH);
```

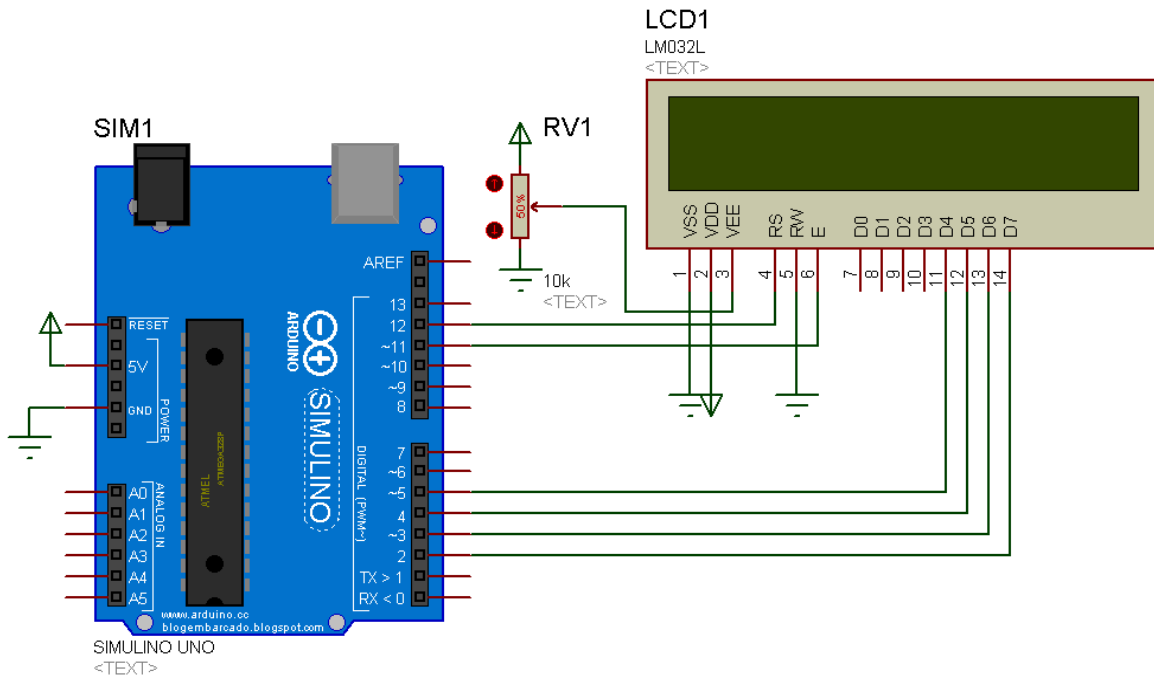
```
digitalWrite(motorPin2, LOW);
```

```
}
```

7) Project 7: Giao tiếp Arduino với LCD 16x2.

Giao tiếp giữa Arduino và LCD 16x2 rất đơn giản bởi vì Arduino IDE đã có sẵn thư viện cho LCD là *LiquidCrystal.h*, công việc của chúng ta là hiểu và biết cách sử dụng thư viện này mà thôi.

✓ Sơ đồ mạch:



Hình 30: giao tiếp với LCD 16x2.

✓ Code chương trình :

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // thiết lập chân cho LCD
void setup() {
  lcd.begin(16, 2); // thiết lập loại LCD sử dụng là 16 cột và 2 dòng
}
void loop() {
  introduce();
  basicPrintDemo();
  displayOnOffDemo();
  setCursordemo();
}
```

```

scrollLeftDemo();
scrollRightDemo();
cursorDemo();
createGlyphDemo();
}

void introduce(){
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("HV HANG KHONG VN");
    lcd.setCursor(1,1);
    lcd.print("HD SD ARDUINO");
    delay (1000);
    for(int x=0; x < 3; x++) {
        lcd.noDisplay(); //tat hien thi
        delay(300);
        lcd.display(); // bat hien thi
        delay(300);
    }
}

void basicPrintDemo() {
    lcd.clear(); // xoa man hinh
    lcd.print("HV HANG KHONG");
    delay(1000);
}

void displayOnOffDemo() {
    lcd.clear();
    lcd.print("BAT/TAT MAN HINH");
}

```

```

    for(int x=0; x < 3; x++) {
        lcd.noDisplay();
        delay(500);
        lcd.display();
        delay(500);
    }
}

void setCursorDemo() {
    lcd.clear();
    lcd.print("Setup Cursor");
    delay(500);
    lcd.clear();
    lcd.setCursor(5,0); // thiet lap con tro o vi tri cot 5 hang 0
    lcd.print("5,0");
    delay(500);
    lcd.setCursor(10,1); // thiet lap con tro cot 10 hang thu 1
    lcd.print("10,1");
    delay(500);
    lcd.setCursor(3,1); // thiet lap con tro o vi tri cot 3 hang 1
    lcd.print("3,1");
    delay(500);
}

void scrollLeftDemo() {
    lcd.clear();
    lcd.print("Scroll Left Demo");
    delay(500);
    lcd.clear();
}

```

```

    lcd.setCursor(7,0);
    lcd.print("Beginning");
    lcd.setCursor(9,1);
    lcd.print("Arduino");
    delay(500);
    for(int x=0; x<16; x++) {
        lcd.scrollDisplayLeft(); // cuon man hinh sang trai
        delay(250);
    }
}

void scrollRightDemo() {
    lcd.clear();
    lcd.print("Scroll Right");
    lcd.setCursor(0,1);
    lcd.print("Demo");
    delay(500);
    lcd.clear();
    lcd.print("Beginning");
    lcd.setCursor(0,1);
    lcd.print("Arduino");
    delay(500);
    for(int x=0; x<16; x++) {
        lcd.scrollDisplayRight(); // cuon mang hinh sang phai
        delay(250);
    }
}
}

```



```

void cursorDemo() {
    lcd.clear();
    lcd.cursor(); // bat con tro
    lcd.print("Cursor On");
    delay(1000);
    lcd.clear();
    lcd.noCursor(); // tat con tro
    lcd.print("Cursor Off");
    delay(1000);
    lcd.clear();
    lcd.cursor();
    lcd.blink(); // nhap nhay con tro
    lcd.print("Cursor Blink On");
    delay(1000);
    lcd.noCursor();
    lcd.noBlink(); // tat nhap nhay con tro
}

```

```

void createGlyphDemo() {
    lcd.clear();
    byte char1[8] = { //tạo chữ "ô"
        B01110,
        B10001,
        B00000,
        B01110,
        B10001,
        B10001,
    };
}

```

```

    B01110,
    B00000
};
byte char2[8] = { //tạo chữ “o”
    B01110,
    B00001,
    B00010,
    B01110,
    B10001,
    B10001,
    B01110,
    B00000
};
lcd.createChar(0, char1); // tạo ký tự tùy chỉnh 0
lcd.createChar(1, char2); // tạo ký tự tùy chỉnh 1
for(int x=0; x<5; x++) {
    lcd.setCursor(8,0);
    lcd.write(byte(0)); // in chữ “ô” ra màn hình
    delay(1000);
    lcd.setCursor(8,0);
    lcd.write(byte(1)); // in chữ “o” ra màn hình
    delay(1000);
}
}

```

✓ Giải thích chương trình:

Đầu tiên chúng ta khai báo thư viện mà chúng ta sẽ sử dụng để điều khiển LCD. Như đã nói ở trên chúng ta sẽ sử dụng thư viện có tên là *LiquidCrystal.h*. Có rất nhiều thư viện và code mẫu cho những loại LCD khác, bạn có thể truy cập vào trang web <http://www.arduino.cc/playground/Code/LCD> để tải về sử dụng.

Để khai báo thư viện cho LCD hay bất cứ thư viện nào khác ta dùng câu lệnh

```
#include <tenthuvien.h>
```

Trong trường hợp này ta khai báo là :

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

Tiếp theo tạo một đối tượng và gán chân cho nó bằng câu lệnh :

```
LiquidCrystal Object(RS, E, D4, D5, D6, D7);
```

Như vậy trong đoạn code trên tôi đã khai báo một đối tượng có tên là *lcd* (các bạn có thể thay thế *lcd* bằng những từ khác mà các bạn muốn) và chân 12 của Arduino nối với chân RS, chân 11 nối với E và các chân 5 đến chân 2 lần lượt nối với D4 đến D7 trên LCD 16x2.

```
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
```

Trong hàm *setup()* chúng ta cần khai báo loại LCD mà chúng ta sử dụng. Vì trong thư viện *LiquidCrystal.h* hỗ trợ rất nhiều loại LCD chẳng hạn như 16x2, 16x4, 20x2, 20x4, GLCD....Ở đây chúng ta sử dụng 16x2 thì ta khai báo.

```
lcd.begin(16,2);
```

Trong *loop()* chúng ta có 8 chương trình con, và tôi sẽ giải thích từng chương trình con.

❖ Chương trình con thứ 1: *introduce()*.

Đây là chương trình giới thiệu.

❖ Chương trình con thứ 2: *basicPrintDemo()*

Trong chương trình con này chúng ta sẽ điều khiển sao cho LCD hiển thị dòng chữ mà ta mong muốn.

Đầu tiên chúng ta xoá tất cả màn hình bằng câu lệnh:

```
lcd.clear();
```

Chúng ta cần lưu ý đối tượng *lcd* : nếu như ban đầu chúng ta khai báo đối tượng là *LCD16x2* thì chúng ta phải viết câu lệnh là *LCD16x2.clear()*.

Để hiển thị một dòng ký tự bất kỳ lên màn hình thì ta dùng câu lệnh *print()* cụ thể trong trường hợp này là:

```
lcd.print("HV HANG KHONG");
```

Các ký tự bên trong ngoặc kép sẽ được hiển thị lên màn hình, nếu tổng các ký tự lớn hơn 16, thì các ký tự từ thứ 17 trở đi sẽ không được hiển thị lên màn hình.

❖ Chương trình con thứ 3: *displayOnOffDemo()*

Trong chương trình con này hướng dẫn cho chúng ta các câu lệnh chức năng bật và tắt màn hình. Ta cần quan tâm tới 2 câu lệnh sau.

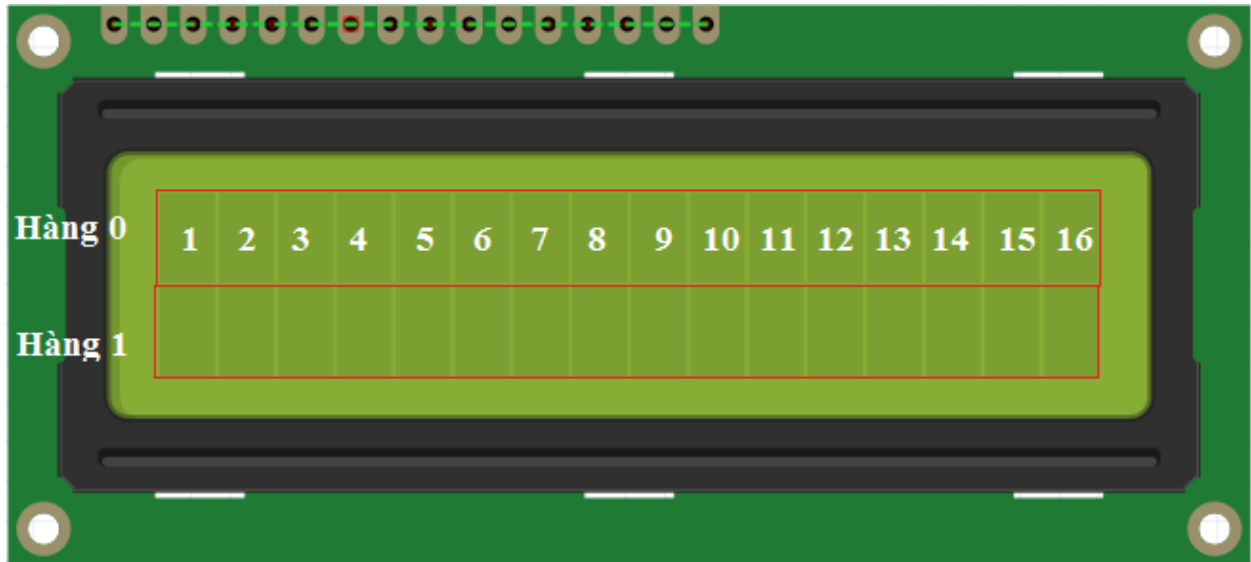
```
lcd.noDisplay();
```

Câu lệnh này có chức năng tắt màn hình hiển thị.

```
lcd.display();
```

Câu lệnh này cho phép hiển thị màn hình.

❖ Chương trình con thứ 4: *setCursorDemo()*



Hình 31: LCD 16x2.

Chương trình con này hướng dẫn chúng ta các câu lệnh dịch chuyển vị trí con trỏ theo ý muốn, các câu lệnh cần quan tâm đó là:

```
lcd.setCursor(5,0);
```

```
lcd.print("5,0");
```

Dịch con trỏ đến cột thứ 5 hàng thứ 0. Sau đó xuất ra màn hình LCD “5,0” từ cột thứ 5 hàng 0 trở đi.

```
lcd.setCursor(10,1); // thiết lập con trỏ cot 10 hang thu 1
```

```
lcd.print("10,1");
```

Dịch con trỏ đến vị trí cột 10 hàng thứ 1. Xuất ra màn hình "10,1"

Tương tự như vậy đối với 2 câu lệnh cuối là :

```
lcd.setCursor(3,1); // thiết lập con trỏ o vi tri cot 3 hang 1
```

```
lcd.print("3,1");
```

❖ Chương trình con thứ 5: *scrollLeftDemo()*

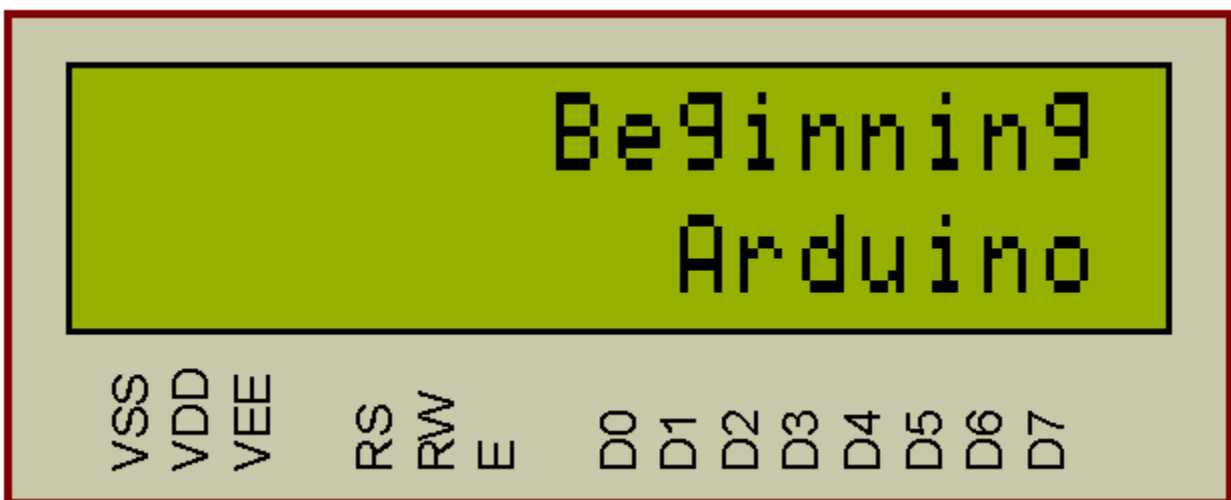
Chương trình con này sẽ dịch các ký tự đang hiển thị trên màn hình sang bên trái. Các câu lệnh trong chương trình con này không khó, chúng ta chỉ quan tâm tới các câu lệnh sau:

```
for(int x=0; x<16; x++) {  
  
    lcd.scrollDisplayLeft(); // cuộn màn hình sang trái  
  
    delay(250);  
  
}
```

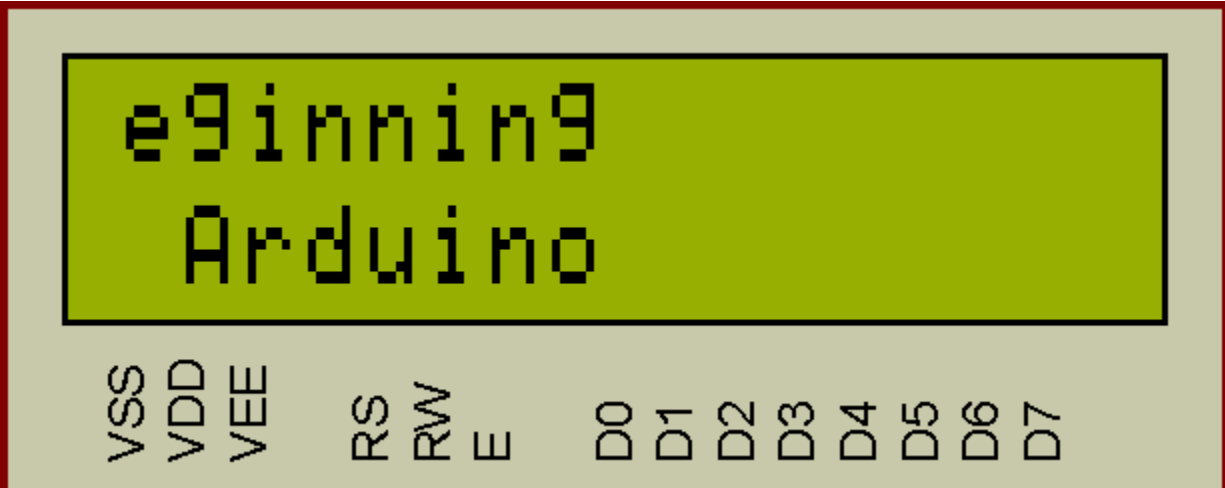
Trong vòng lặp chúng ta có câu lệnh:

```
lcd.scrollDisplayLeft();
```

Mỗi lần chương trình thực hiện câu lệnh này sẽ dịch tất cả các ký tự đang hiển thị trên màn hình sang bên trái 1 cột. Chúng ta có vòng lặp 16 lần như vậy các ký tự sẽ được dịch hết về bên trái.



Hình 32: Trước khi dịch trái.



Hình 33: Dịch trái lần thứ 8.

Khi vòng lặp thực hiện được 16 lần thì màn hình sẽ trống hoàn toàn.

❖ Chương trình con thứ 6: *scrollRightDemo()*

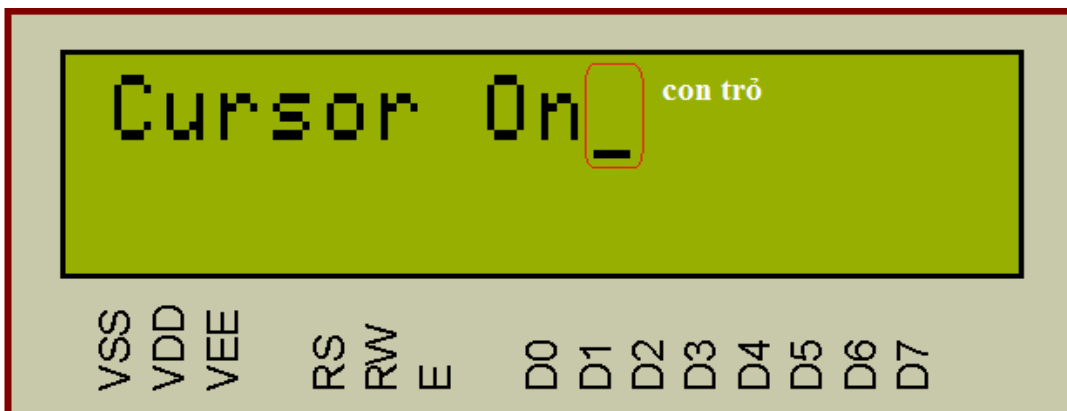
Hoàn toàn tương tự như *scrollLeftDemo()* chương trình con này sẽ thực hiện dịch phải các ký tự trên màn hình.

Câu lệnh cần quan tâm là: *lcd.scrollDisplayRight();*

❖ Chương trình con thứ 7: *cursorDemo()*

Ở phần này chúng ta sẽ tìm hiểu các câu lệnh điều khiển con trỏ bật, tắt và nhấp nháy.

lcd.cursor() : câu lệnh này cho phép chúng ta bật con trỏ.



Hình 34: Con trỏ LCD

lcd.noCursor(): tắt con trỏ

lcd.blink(): nhấp nháy con trỏ

❖ Chương trình con thứ 8: *createGlyphDemo()*

Bây giờ chúng ta sẽ tìm hiểu cách tự tạo ra một ký tự không thuộc hệ thống mã ASCII, chẳng hạn như các chữ cái có dấu trong hệ thống chữ cái tiếng việt như ă, â, ô,ơ

Đối với LCD 16x2 cứ mỗi ký tự trong một ô sẽ được tạo thành từ 5x8 ô nhỏ (5 cột, 8 dòng)

Để tạo một ký tự thì chúng ta dùng một mảng gồm 8 phần tử, mỗi phần tử là 1 byte, nhưng chỉ sử dụng 5 bit thấp của 1 byte để biểu diễn ký tự đó.

```
byte happy[8] = { //tạo chữ “ô”
```

```
B01110,
```

```
B10001,
```

```
B00000,
```

```
B01110,
```

```
B10001,
```

```
B10001,
```

```
B01110,
```

```
B00000
```

```
};
```

```
byte sad[8] = { tạo chữ “ơ”
```

```
B01110,
```

```
B00001,
```

```
B00010,
```

```
B01110,
```

```
B10001,
```



```
B10001,  
B01110,  
B00000  
};
```

Sau khi đã tạo được ký tự mong muốn ta sử dụng câu lệnh:

```
lcd.createchar(num,data);
```

Trong đó:

num: là các chữ số từ 0 đến 7.

data: là các mãng chứa ký tự của chúng ta.

Câu lệnh này sẽ gán ký tự ta đã tạo vào một chữ số.

```
lcd.createChar(0, happy);
```

```
lcd.createChar(1, sad);
```

Để hiển thị một ký tự ra màn hình ta dùng câu lệnh *lcd.write(data)*.

Hiển thị chữ “ô” ra màn hình LCD.

```
lcd.write(byte(0));
```

Hiển thị chữ “ơ” ra màn hình LCD

```
lcd.write(byte(1));
```

8) Project 8: Giao tiếp với máy tính.

Trong phần này tôi sẽ trình bày cách để giao tiếp giữa Arduino với máy tính thông qua chuẩn giao tiếp nối tiếp không đồng bộ UART.

Điều khiển bật tắt bằng cách gửi lệnh từ máy tính.

✓ Sơ đồ mạch:


Dùng cáp USB kết nối Arduino với máy tính. Led nối với chân 13 thông qua điện trở 220 ohm.

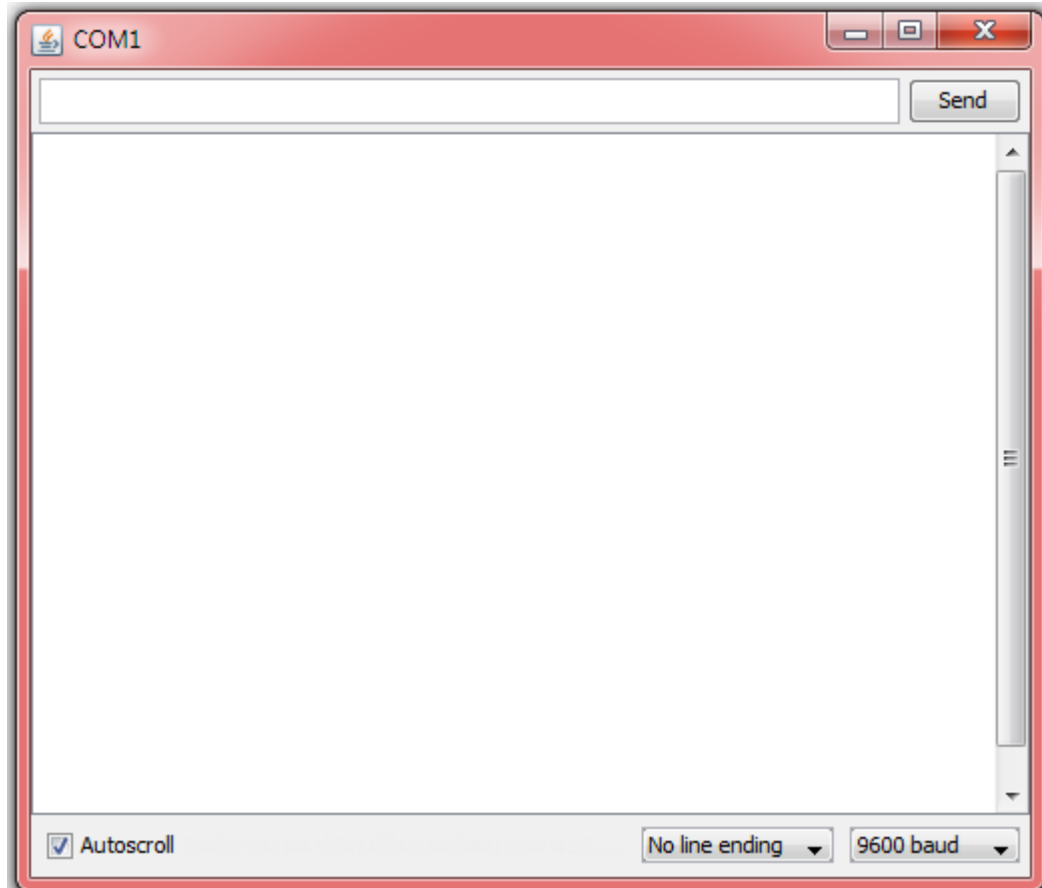
✓ Code chương trình:

```
int ledpin =13;
void setup() {
  Serial.begin(9600);
  pinMode(ledpin,OUTPUT);
}

void loop() {
  if( Serial.available(>0)
  char setupled = Serial.read();}
  switch(setupled)
  {
  case '1': {
    digitalWrite(ledpin,HIGH); break;
  }
  case '0':
  {
    digitalWrite(ledpin,LOW); break; }}}}
```

✓ Giải thích chương trình:

Để có thể điều khiển được led bật tắt chúng ta cần mở Serial monitor bằng cách nhấp vào biểu tượng  :



Hình 35: Serial Monitor.

Trong chương trình ta cần chú ý tới các câu lệnh sau:

```
Serial.begin(9600);
```

Câu lệnh này thiết lập tốc độ truyền dữ liệu của chúng ta là 9600 bps.

Chúng ta có thể thiết lập các tốc độ khác như 300, 1200, 2400, 4600, 9600, 19200, 57600, 115200. Cần lưu ý rằng để tốc độ truyền giữa máy tính và thiết bị phải giống nhau, nếu không thì dữ liệu nhận được sẽ bị lỗi.

Trong vòng lặp *loop()* chúng ta có câu lệnh:

```
Serial.available()>0
```

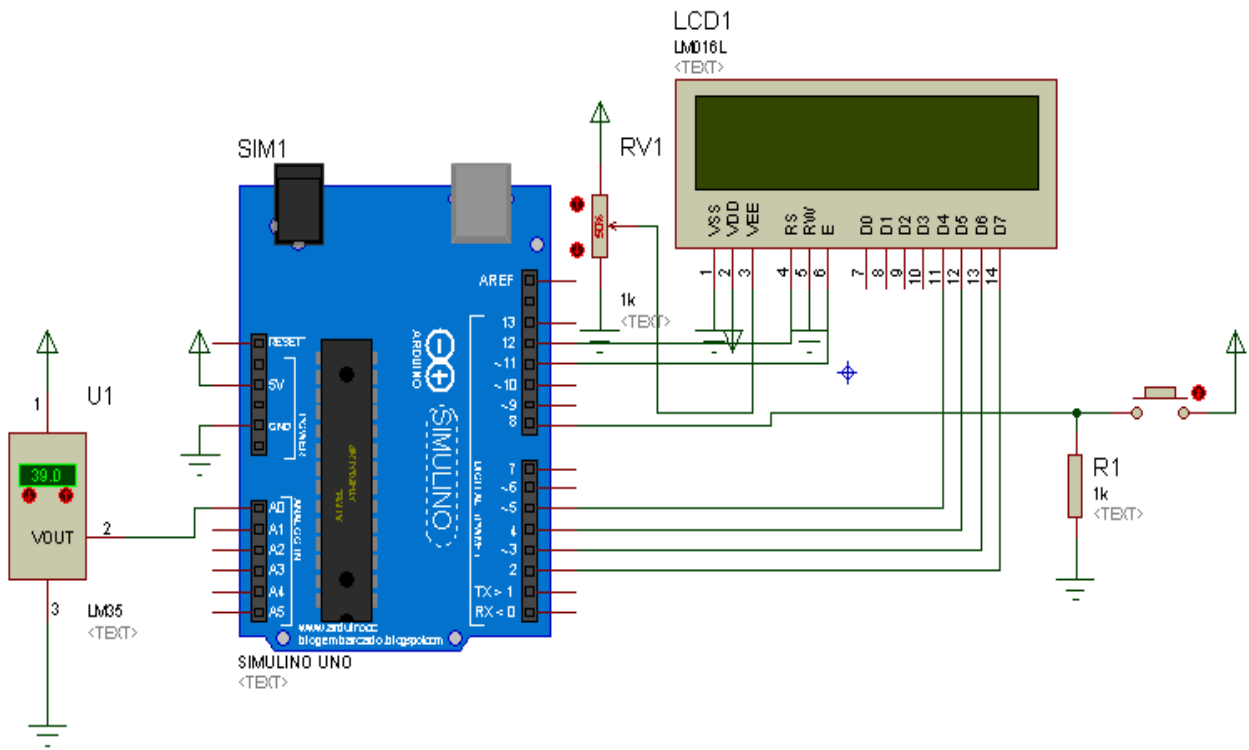
Câu lệnh này dùng để kiểm tra xem có dữ liệu truyền tới hay không. Ngoài ra *Serial.available()* còn trả về cho chúng ta số ký tự đã được truyền tới Arduino

```
char setupled = Serial.read();
```

Khi dữ liệu được truyền tới Arduino thì dữ liệu sẽ được lưu vào bộ nhớ đệm. Chúng ta khai báo biến *setupled* với kiểu dữ liệu char và dùng hàm *Serial.read()* để truy suất dữ liệu trong bộ nhớ đệm và lưu vào trong nó. Như vậy ký tự đầu tiên trong chuỗi ký tự được truyền tới sẽ được gán vào *setupled*. Dùng hàm Switch-case để kiểm tra, nếu là “1” thì sáng led, nếu là “0” thì tắt led, các trường hợp còn lại thì không làm gì.

9) Project 9. Đo nhiệt độ môi trường dùng LM35D hiển thị LCD và Serial Monitor.

✓ Sơ đồ mạch.



Hình 36: giao tiếp với LM35, LCD và Serial monitor.

✓ Code chương trình:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // định nghĩa chân cho LCD
int maxC=0, minC=100, maxF=0, minF=212;
```

```

int scale = 1;
int buttonPin=8; //chan ket noi voi phim nhan
void setup() {
    lcd.begin(16, 2); // khai bao su dung lcd 16x2
    analogReference(INTERNAL); // khai bao dien ap tham chieu cho lm35
    Serial.begin(9600);
    pinMode(buttonPin, INPUT);
    lcd.clear();
}
void loop() {
    lcd.setCursor(0,0);
    int sensor = analogRead(0); // doc du lieu tu lm35
    int buttonState = digitalRead(buttonPin);
    if (buttonState==1) {
        scale=-scale;
        lcd.clear();
    }
    switch (scale) {
        case 1:
            celsius(sensor);
            break;
        case -1:
            fahrenheit(sensor);}
        delay(250);
    }
    void celsius(int sensor) {
        lcd.setCursor(0,0);

```

```

    int temp = sensor* 0.1074188;
    Serial.print("Nhiệt độ môi trường :");
    Serial.print(temp);
    Serial.println(" độ C");
    lcd.print(temp);
    lcd.write(B11011111); // "0"
    lcd.print("C ");
    if (temp>maxC) {maxC=temp;}
    if (temp<minC) {minC=temp;}
    lcd.setCursor(0,1);
    lcd.print("H=");
    lcd.print(maxC);
    lcd.write(B11011111);
    lcd.print("C L=");
    lcd.print(minC);
    lcd.write(B11011111);
    lcd.print("C ");
}

void fahrenheit(int sensor) {
    lcd.setCursor(0,0);
    float temp = ((sensor * 0.1074188) * 1.8)+32; // chuyển đổi sang độ F
    Serial.print("Nhiệt độ môi trường :");
    Serial.print(int(temp));
    Serial.println("độ F");
    lcd.print(int(temp));
    lcd.write(B11011111);
    lcd.print("F ");
}

```

```

    if (temp>maxF) {maxF=temp;}
    if (temp<minF) {minF=temp;}
    lcd.setCursor(0,1);
    lcd.print("H=");
    lcd.print(maxF);
    lcd.write(B11011111);
    lcd.print("F L=");
    lcd.print(minF);
    lcd.write(B11011111);
    lcd.print("F ");
}

```

✓ Giải thích chương trình:

Tổng quan: chương trình sẽ kiểm tra mức điện áp ngõ ra của LM35 tương ứng với nhiệt độ của môi trường hiển thị trên Lcd và Serial Monitor. Nhiệt độ của môi trường được tính bằng độ C và độ F, nếu nhấn phím thì sẽ thay đổi hiển thị là độ C hay độ F.

Bắt đầu chương trình ta khai báo thư viện Lcd và định nghĩa chân cũng như các biến cần dùng cho toàn bộ chương trình.

```

#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // định nghĩa chân cho LCD
int maxC=0, minC=100, maxF=0, minF=212;
int scale = 1;
int buttonPin=8;// chân kết nối với phím nhấn

```

Trong hàm *setup()* chúng ta cần quan tâm tới câu lệnh:

```

analogReference(INTERNAL);

```

Khai báo điện áp tham chiếu cho bộ chuyển đổi ADC bên trong Arduino Uno là 1,1V (đây là giá trị điện áp tham chiếu nội mặc định của Arduino Uno).

Để tính toán đúng nhiệt độ môi trường chúng ta cần lưu ý những điểm sau:

Bộ chuyển đổi ADC gồm 10 bit tức là 1024 mức.

Điện áp tham chiếu cho bộ chuyển đổi ADC là 1,1V.

Đối với LM35 thì điện áp ngõ ra tuyến tính với nhiệt độ của môi trường.

Hệ số chuyển đổi điện áp sang nhiệt độ là $10\text{mV}/1^{\circ}\text{C} = 0,01\text{V}/1^{\circ}\text{C}$.

Xây dựng công thức tính nhiệt độ.

Ta biết:

1,1 volt (1100 mV) có 1024 mức biểu diễn, vậy 1 mức sẽ là 1,1/1024 (volt), để chuyển đổi từ điện áp sang nhiệt độ thì ta chia tiếp cho 0,01V. Từ đây ta thấy cứ 1 mức chuyển đổi của ADC tương ứng với $\frac{1,1}{1024 \cdot 0,01}^{\circ}\text{C} = 0.10742188^{\circ}\text{C}$.

Như vậy chúng ta chỉ cần đọc giá trị đầu vào ở chân A0 (giá trị nằm trong khoảng 0 - 1023) và gán chúng cho biến integer *sensor*.

```
int sensor = analogRead(0);
```

Để tính ra nhiệt độ chính xác chúng ta nhân giá trị của *sensor* với 0.10742188.

Trong chương trình con *celsius(int sensor)* ta có câu lệnh:

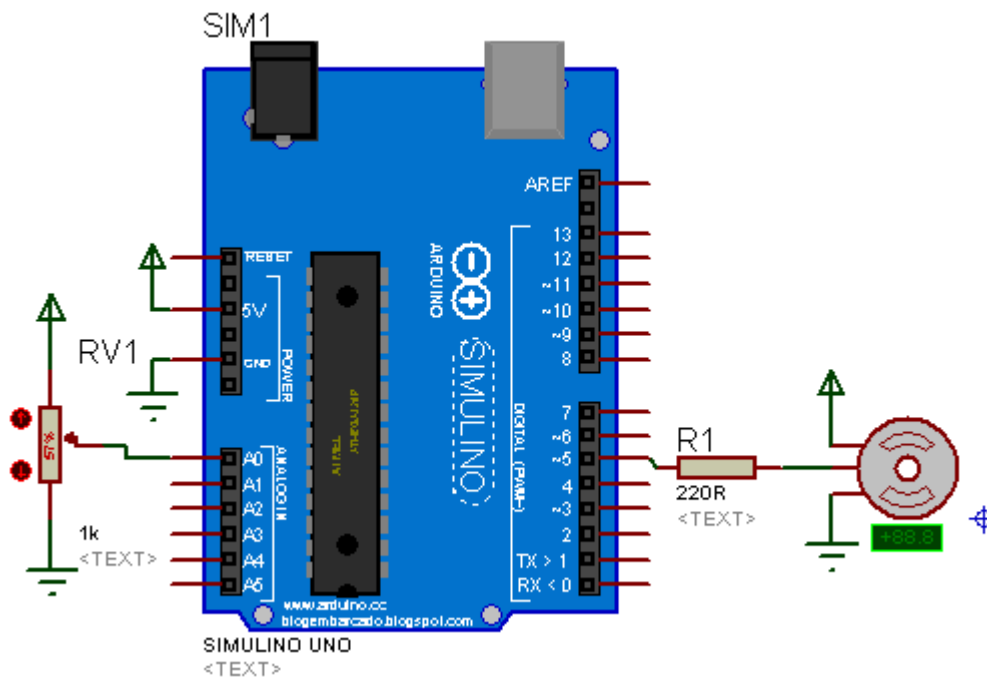
```
int temp = sensor * 0.1074188;
```

Trong chương trình con *fahrenheit(int sensor)* ta có câu lệnh chuyển đổi từ độ C sang độ F là:

```
float temp = ((sensor * 0.1074188) * 1.8) + 32;
```


10) Project 10: Giao tiếp Arduino với Servo motor.

- ✓ Sơ đồ mạch



Hình 37: Giao tiếp với Servo motor

- ✓ Code chương trình.

```
#include <Servo.h>

Servo servo1;

void setup()
{
    servo1.attach(5);
}

void loop()
{
    int angle = analogRead(0); // Read the pot value
    angle=map(angle, 0, 1023, 0, 180);
```

```
servo1.write(angle);  
delay(15);  
}
```

✓ Giải thích chương trình:

Khai báo thư viện cho Servo motor bằng câu lệnh.

```
#include <Servo.h>
```

Khai báo đối tượng có tên là *servo1*

Trong hàm setup() ta định nghĩa chân cho Servo:

```
servo1.attach(5);
```

Chân số 5 của Arduino sẽ nối với chân input của Servo motor.

Đọc giá trị điện áp của biến trở và gán nó cho biến integer *angle* :

```
int angle = analogRead(0);
```

Giá trị đọc được từ biến trở sẽ nằm trong khoảng 0 đến 1023 và góc quay của Servo từ 0^0 đến 180^0 ta sử dụng câu lệnh :

```
angle=map(angle, 0, 1023, 0, 180);
```

Câu Lệnh này sẽ chuyển đổi từ giá trị của biến trở sang góc quay tương ứng của Servo.

Để điều khiển góc quay của Servo ta dùng câu lệnh:

```
servo1.write(angle);
```

Tài liệu tham khảo.

- Beginning Arduino - Mike McRoberts
- Arduino cookbook – Michael Margolis
- <http://arduino.cc>
- <http://blogembarcado.blogspot.com>