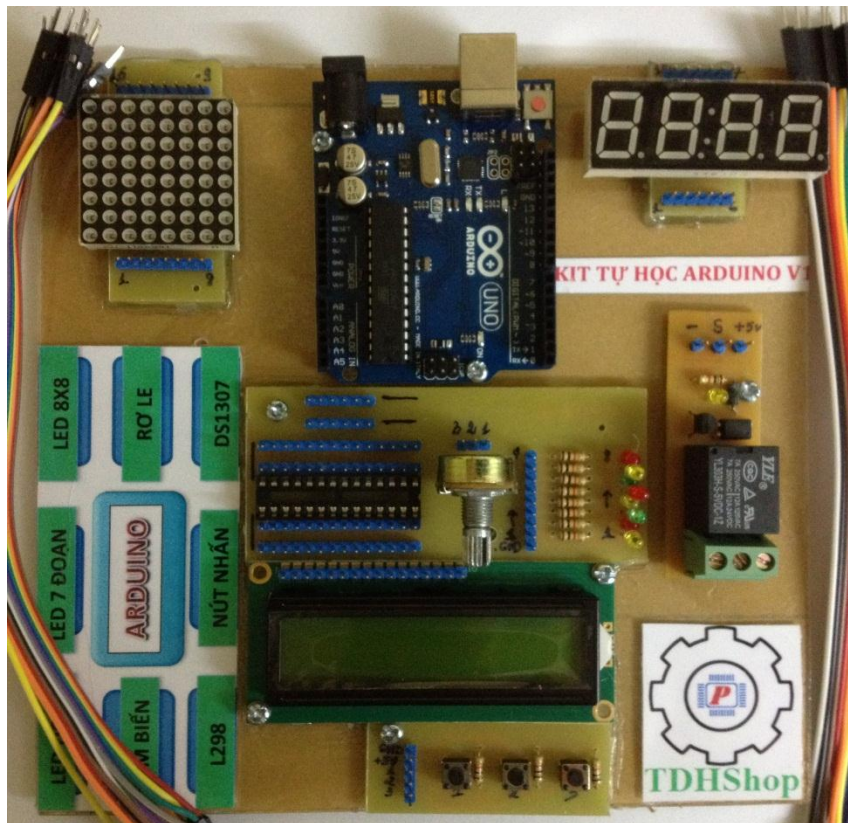


SỔ TAY ARDUINO



LẬP TRÌNH ĐIỀU KHIỂN ARDUINO CHO NGƯỜI MỚI BẮT ĐẦU V1



Web: tdhshop.com.vn – Chuyên Kit Tự Học Arduino từ cơ bản đến nâng cao

SỔ TAY ARDUINO

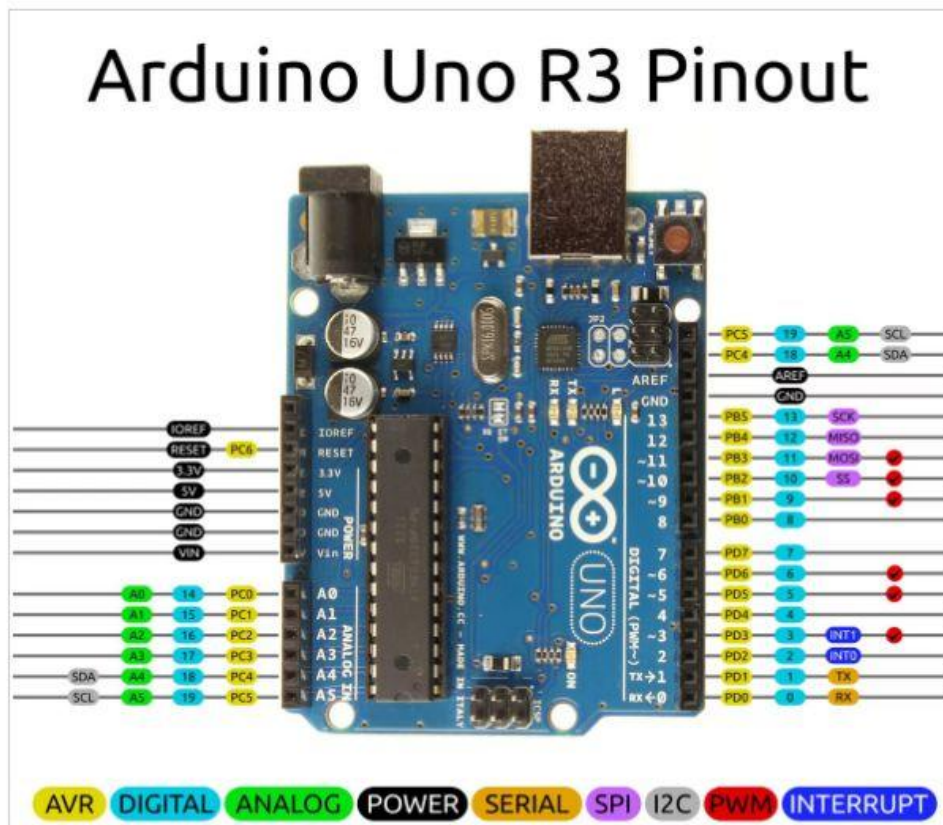
NỘI DUNG CHÍNH:

- 1 .GIỚI THIỆU SƠ LƯỢC VỀ ARDUINO UNO R3.**
- 2 .GIỚI THIỆU SƠ LƯỢC VỀ ARDUINO PRO MINI.**
- 3 .GIỚI THIỆU SƠ LƯỢC VỀ ARDUINO NANO.**
- 4 .GIỚI THIỆU SƠ LƯỢC VỀ ARDUINO MEGA 2560.**
- 5 .HƯỚNG DẪN CÀI THƯ VIỆN CHO ARDUINO IDE.**
- 6 .NGÔN NGỮ LẬP TRÌNH ARDUINO.**
- 7 .MỘT SỐ VÍ DỤ LẬP TRÌNH ARDUINO CƠ BẢN.**

SỔ TAY ARDUINO

1. Arduino Uno R3:

Arduino Uno được xây dựng với phân nhân là vi điều khiển ATmega328P sử dụng thạch anh có chu kỳ dao động là 16 MHz. Với vi điều khiển này, ta có tổng cộng 14 pin (ngõ) ra / vào được đánh số từ 0 tới 13 (trong đó có 6 pin PWM, được đánh dấu ~ trước mã số của pin). Song song đó, ta có thêm 6 pin nhận tín hiệu analog được đánh kí hiệu từ A0 - A5, 6 pin này cũng có thể sử dụng được như các pin ra / vào bình thường (như pin 0 - 13). Ở các pin được đề cập, pin 13 là pin đặc biệt vì nối trực tiếp với LED trạng thái trên board.



Trên board còn có 1 nút reset, 1 ngõ kết nối với máy tính qua cổng USB và 1 ngõ cấp nguồn sử dụng jack 2.1mm lấy năng lượng trực tiếp từ AC-DC adapter hay thông qua ắc-quy nguồn.

Khi làm việc với Arduino board, một số thuật ngữ sau cần được lưu ý:

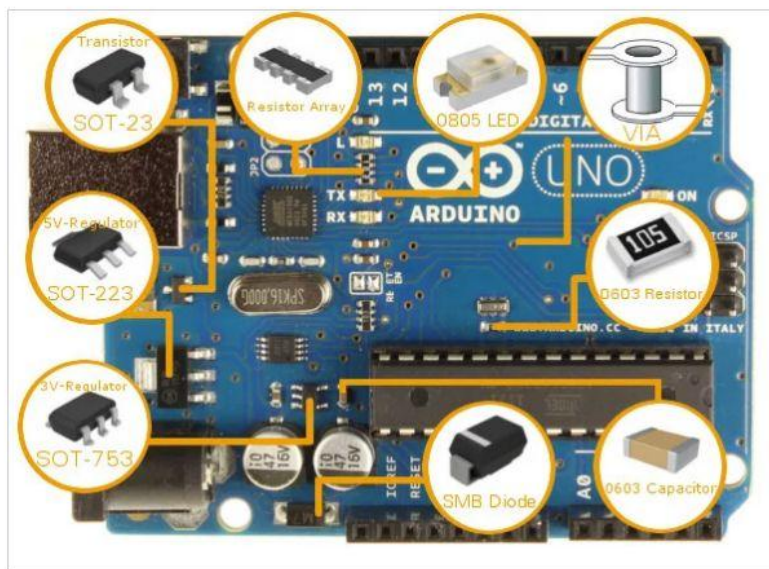
- **Flash Memory:** bộ nhớ có thể ghi được, dữ liệu không bị mất ngay cả khi tắt điện. Về vai trò, ta có thể hình dung bộ nhớ này như ổ cứng để chứa dữ liệu trên board. Chương trình được viết cho Arduino sẽ được lưu ở đây. Kích thước của vùng nhớ này thông thường dựa vào vi điều khiển được sử dụng,

SỔ TAY ARDUINO

ví dụ như ATmega8 có 8KB flash memory. Loại bộ nhớ này có thể chịu được khoảng 10,000 lần ghi / xóa.

- **RAM:** tương tự như RAM của máy tính, sẽ bị mất dữ liệu khi ngắt điện nhưng bù lại tốc độ đọc ghi xóa rất nhanh. Kích thước nhỏ hơn Flash Memory nhiều lần.
- **EEPROM:** một dạng bộ nhớ tương tự như Flash Memory nhưng có chu kỳ ghi / xóa cao hơn - khoảng 100,000 lần và có kích thước rất nhỏ. Để đọc / ghi dữ liệu ta có thể dùng thư viện EEPROM của Arduino.

Ngoài ra, board Arduino còn cung cấp cho ta các pin khác nhau như pin cấp nguồn 3.3V, pin cấp nguồn 5V, pin GND...



Thông số kỹ thuật của Arduino board được tóm tắt trong bảng sau:

Vi điều khiển	ATmega328P
Điện áp hoạt động	5V
Điện áp vào khuyến dùng	7-12V
Điện áp vào giới hạn	6-20V
Digital I/O pin	14 (trong đó 6 pin có khả năng băm xung)
PWM Digital I/O Pins	6
Analog Input Pins	6
Cường độ dòng điện trên mỗi I/O pin	20 mA
Cường độ dòng điện trên mỗi 3.3V pin	50 mA
Flash Memory	32 KB (ATmega328P) 0.5 KB được sử dụng bởi bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)

SỔ TAY ARDUINO

Tốc độ	16 MHz
Chiều dài	68.6 mm
Chiều rộng	53.4 mm
Trọng lượng	25 g

Lập trình cho arduino uno:

Ở bài viết này các bạn sẽ được hướng dẫn cách nạp chương trình đơn giản điều khiển đèn LED nhấp nháy theo chu kì 2 giây cho Arduino Uno R3.

Trước tiên hãy đảm bảo rằng máy tính của bạn đã cài đặt Arduino IDE và Arduino driver .

Khi phần mềm xong ta hãy chuẩn bị phần cứng như sau:

- + Một board Arduino uno r3.
- + Một sợi dây cáp (để kết nối arduino với máy tính).

Các bạn hãy làm theo các bước sau để nạp được chương trình cho arduino uno r3.

Bước 1: Kết nối arduino uno r3 với máy tính.

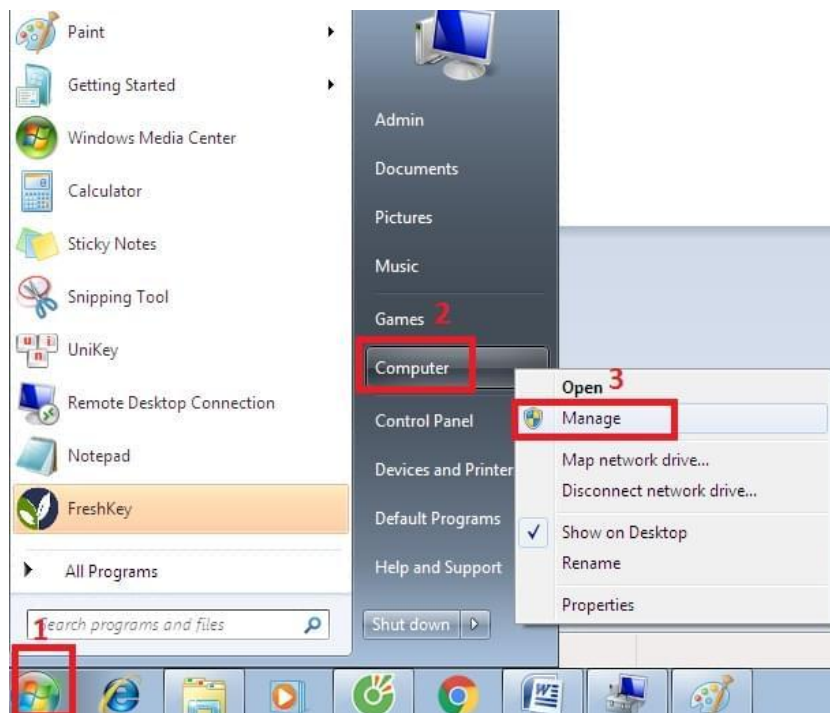


Kết nối arduino uno r3 với máy tính

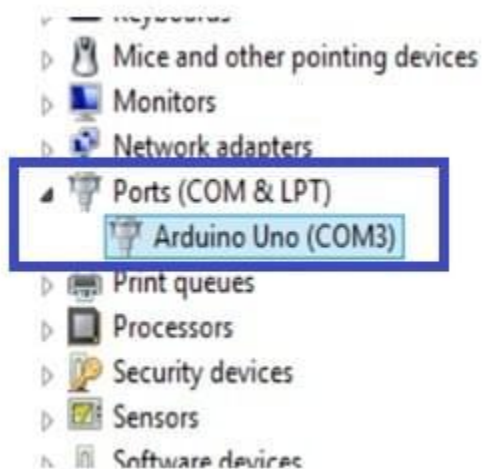
Bước 2: Tìm cổng kết nối của Arduino Uno R3 với máy tính.

Khi Arduino Uno R3 kết nối với máy tính, nó sẽ sử dụng một cổng COM để máy tính và bo mạch có thể truyền tải dữ liệu qua lại thông qua cổng này. Windows có thể quản lí đến 256 cổng COM. Để tìm được cổng COM đang được sử dụng để máy tính và mạch Arduino UNO R3 giao tiếp với nhau, bạn phải mở chức năng Device Manager của Windows.

SỔ TAY ARDUINO



Mở cửa sổ Device Manager lên để tìm cổng COM kết nối với arduino uno r3
Mở mục Ports (COM & LPT), bạn sẽ thấy cổng COM Arduino Uno R3 đang kết nối.



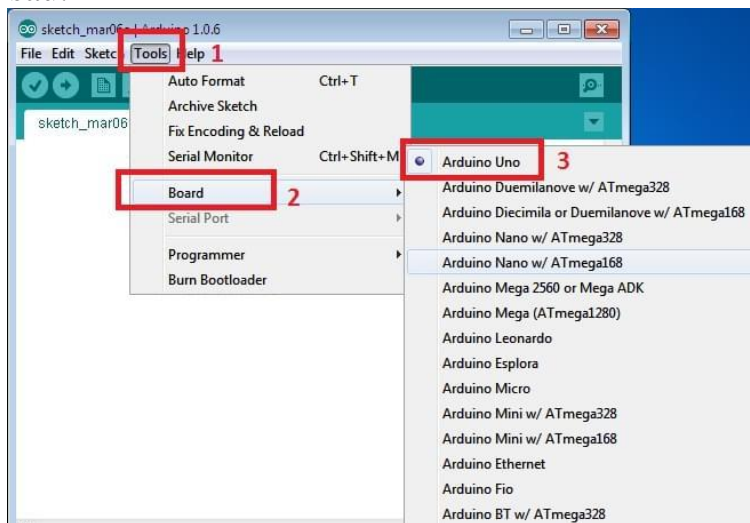
Ở đây arduino uno r3 được kết nối với COM3

Thông thường Windows sẽ sử dụng lại cổng COM3 để kết nối nên bạn không cần thực hiện thêm thao tác tìm cổng COM này nữa.

Bước 3: Nạp chương trình cho arduino uno r3.

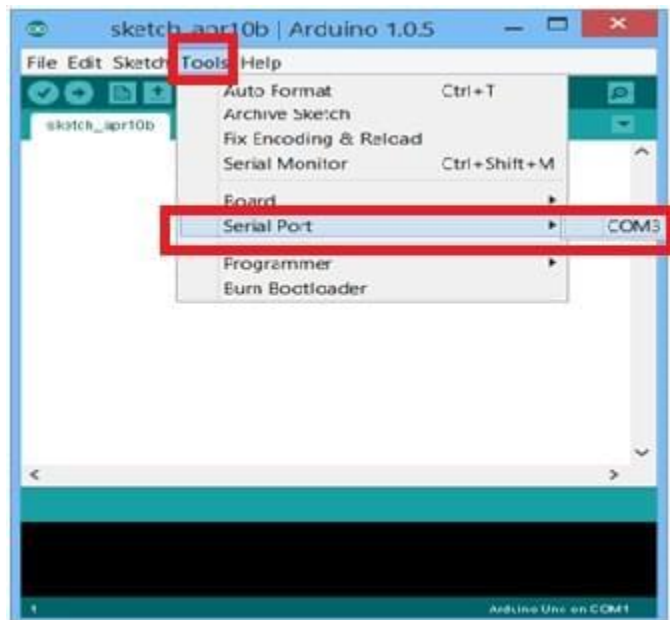
SỔ TAY ARDUINO

Ta hãy thử nạp chương trình có sẵn trong arduino IDE trước. Trước tiên bạn hãy làm các thao tác sau.



Các bạn lưu ý nhớ chọn đúng board arduino mình sử dụng

Tiếp theo vào menu Tools -> Serial Port -> chọn cổng Arduino đang kết nối với máy tính.



Chọn cổng COM cho arduino IDE

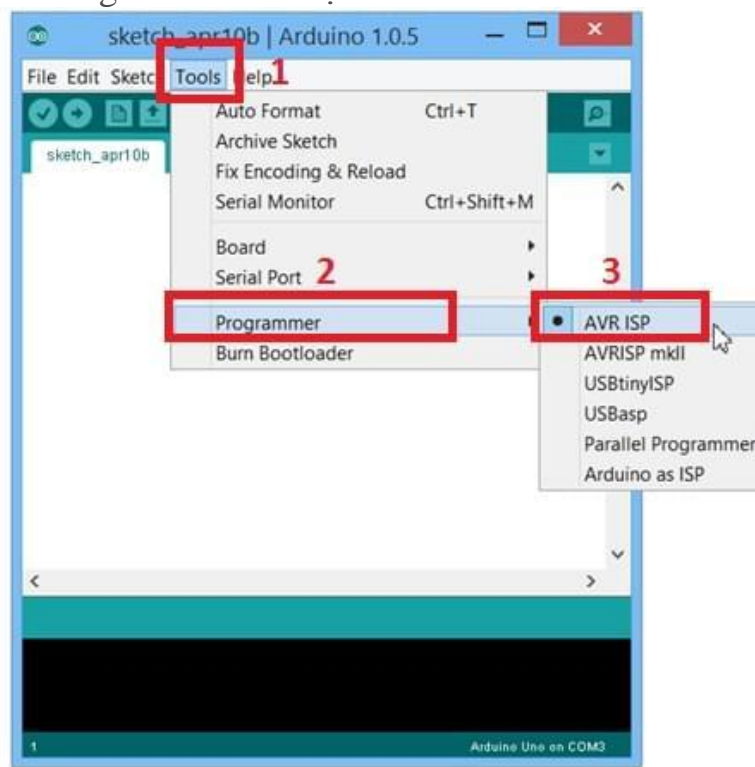
Xác nhận cổng COM của Arduino IDE ở góc dưới cùng bên phải cửa sổ làm việc.

SỔ TAY ARDUINO



Xác nhận cổng COM

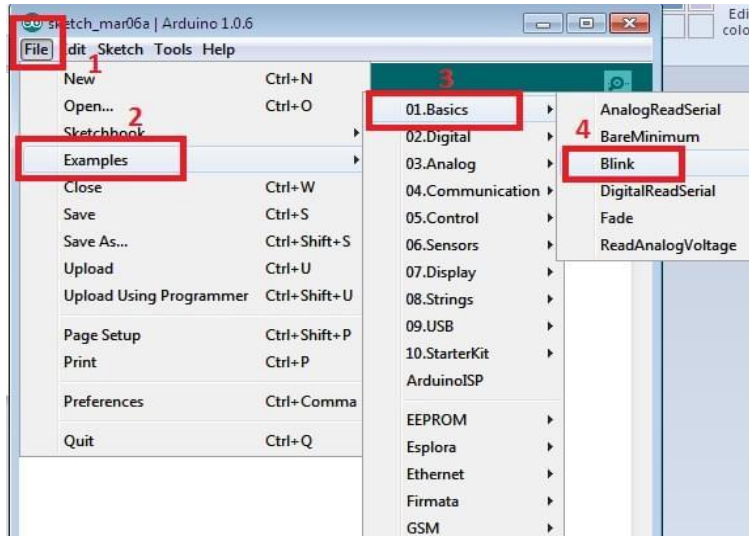
Vào menu Tools -> Programmer -> chọn AVR ISP



Lưu ý các bạn phải chọn AVR ISP

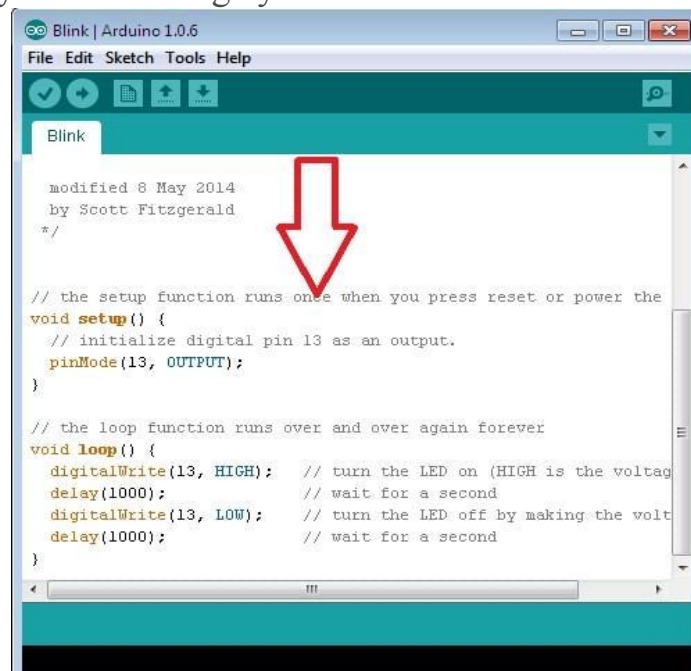
Tiếp theo ta nạp mã nguồn chương cho arduino uno r3.

SỔ TAY ARDUINO



Mở chương trình mẫu của arduino IDE

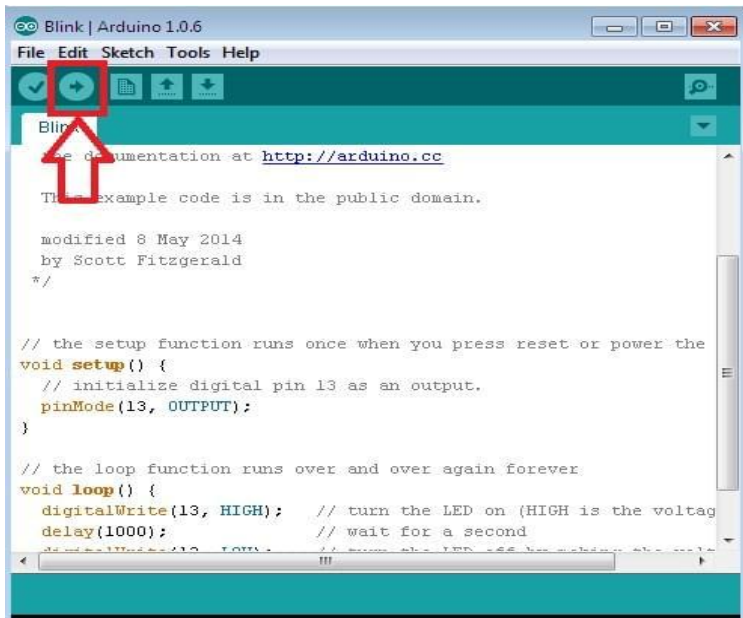
Bạn sẽ thấy Arduino IDE mở một cửa sổ mới chứa mã nguồn **Blink**. Chương trình này có chức năng là điều khiển đèn LED 13 màu cam trên mạch Arduino Uno R3 nhấp nháy với chu kì 1 giây.



Cửa sổ chương trình "Blink" hiện ra

Và ta làm 1 thao tác cuối cùng để nạp chương trình.

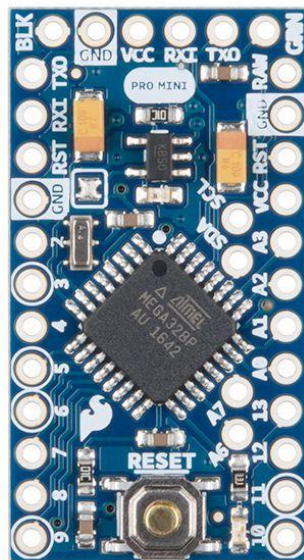
SỔ TAY ARDUINO



Đúp chuột vào chỗ chỉ của dấu mũi tên để đổ chương trình xuống arduino uno r3

CHÚC CÁC BẠN THÀNH CÔNG!!!

2. Arduino Pro Mini (là board Arduino rất nhỏ, sử dụng chip ATmega328 SMD).



Đặc biệt thích hợp cho các ứng dụng thực tế đòi hỏi sự gọn gàng.

Board Arduino Pro Mini 5V 16MHz mặc định sử dụng nguồn 5V và IC ATmega328 chạy ở xung nhịp 16MHZ. Tuy nhiên trên board có sẵn ngõ vào

Web: tdhshop.com.vn – Chuyên Kit Tự Học Arduino từ cơ bản đến nâng cao

SỔ TAY ARDUINO

RAW để cấp nguồn thông qua mạch điều áp. Nguồn vào cho ngõ RAW có thể từ 3.3V - 12V (max 12V)

+ **RAW**: cấp nguồn thông qua mạch điều áp

+ **Vcc**: cấp nguồn 5V hoặc 3.3V (Lưu ý: nguồn > 5.5V sẽ gây hỏng IC)



Vì sử dụng chung dòng chip ATmega328P nên việc lập trình và thiết kế ứng dụng hoàn toàn tương tự board Arduino Uno R3. Ngoài ra có 1 sự khác biệt nhỏ là board *Arduino Pro Mini có tới 8 cổng analog* (thay vì 6 như trên Arduino Uno R3). Trong đó 2 ngõ analog A6, A7 không thể xuất tín hiệu digital!

Arduino Pro Mini không có sẵn giao tiếp USB.

Điều này có nghĩa là bạn không thể cắm trực tiếp board Arduino Pro Mini vào máy tính như Arduino Mega 2560, Arduino Uno R3, Arduino Nano.

Nếu bạn cần 1 board arduino kích thước nhỏ, có sẵn cổng USB để kết nối với máy tính thì Arduino Nano là sự lựa chọn thích hợp hơn là Arduino Pro Mini.

Tuy nhiên bạn có thể dễ dàng sử dụng board Arduino Uno R3 sẵn có của mình để lập trình cho Arduino Pro Mini.

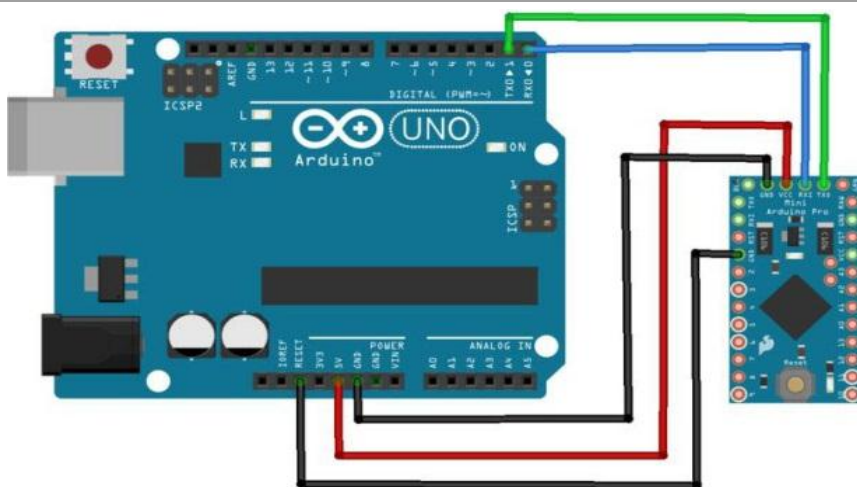
Cách làm như sau:

SỔ TAY ARDUINO

+ **Tháo chip ATmega328** trên board Arduino Uno R3 ra.

+ Gắn chân theo sơ đồ sau:

Arduino Pro Mini	Arduino Uno R3
RST	RESET
Vcc	5V
Gnd	Gnd
Tx	Tx
Rx	Rx



+ Cắm Arduino Uno R3 vào cổng USB trên máy tính. Nếu lần đầu sử dụng Arduino Pro Mini, bạn sẽ thấy LED trên Arduino Pro Mini nhấp nháy.

+ Chọn: Tools > Board > **Arduino Pro or Pro Mini (5V, 16MHz) w/ ATmega328**

+ Chọn cổng COM thích hợp

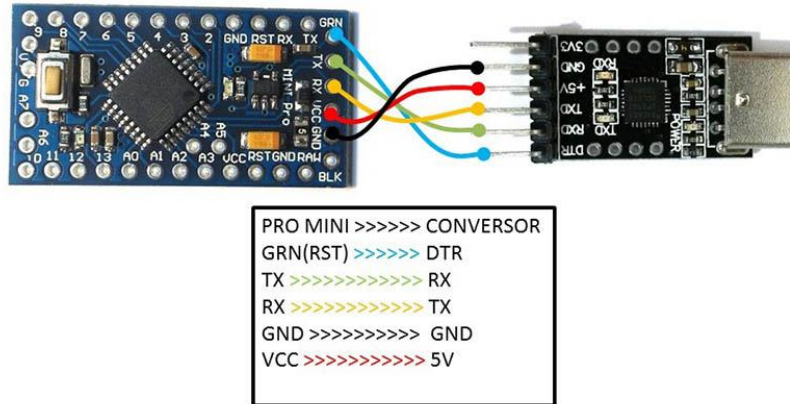
Như vậy là bạn đã có thể Upload chương trình của mình cho Arduino Pro Mini rồi.

***Các bạn có thể nạp code** cho *Board Pro Mini* bằng **Board USB to Serial UART (CP2102,...)**. Cách kết nối như sau mạch nạp với Arduino Pro Mini như sau:

SỔ TAY ARDUINO

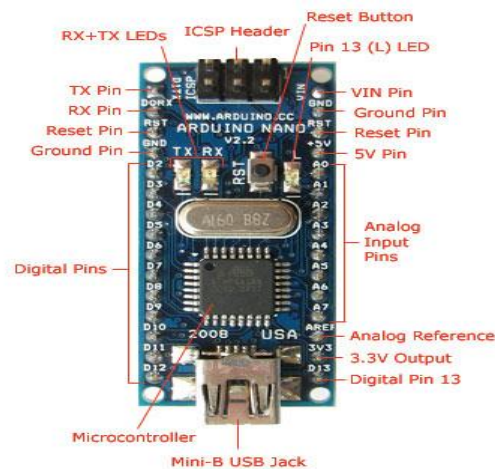
+ Chọn: Tools > Board > **Arduino Pro or Pro Mini (5V, 16MHz) w/ ATmega 328**

+ Chọn cổng COM thích hợp



3. Arduino Nano (Nhỏ, tiện lợi, mang trên mình tinh hoa của Arduino Uno).

Khi tiếp xúc với Arduino Nano, đó là sự tiện dụng, đơn giản, có thể lập trình trực tiếp bằng máy tính (như UNO R3) và đặc biệt hơn cả đó là kích thước của nó. Kích thước của Arduino Nano cực kì nhỏ chỉ tương đương đồng 2 nghìn gấp lại 2 lần thôi (1.85cm x 4.3cm), rất thích hợp cho các newbie, vì giá rẻ hơn Arduino Uno nhưng dùng được tất cả các thư viện của mạch này. Hôm nay, tớ viết bài này nhằm mục đích giới thiệu về mạch Arduino Nano và các thông số kĩ thuật, cùng với đó là những gợi ý ứng dụng khi bắt đầu với mạch này.

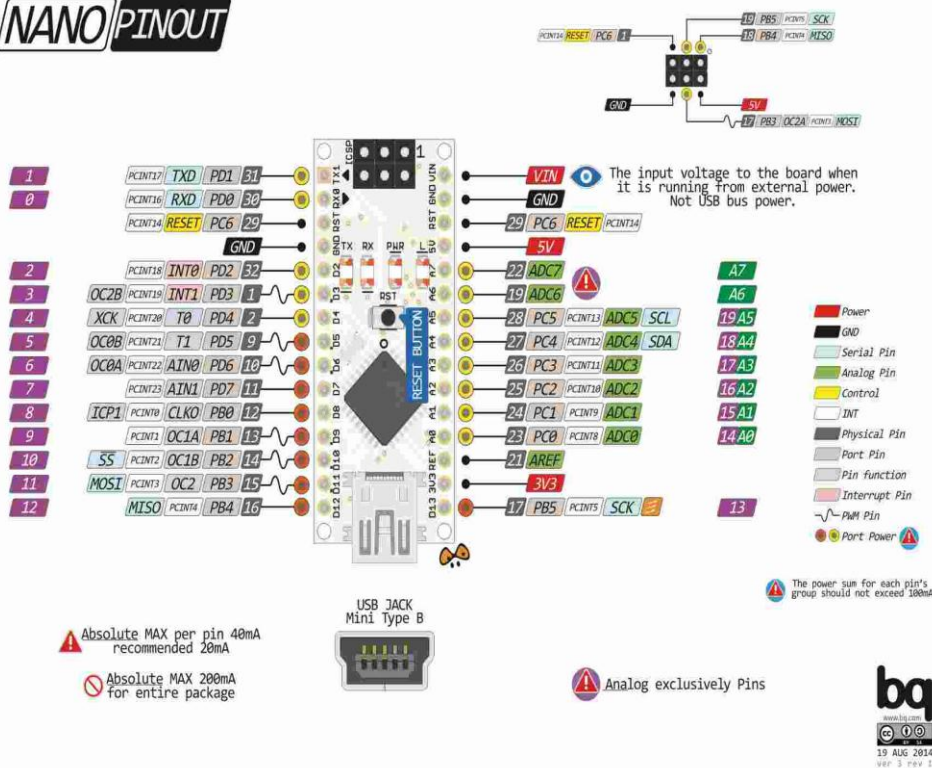


SỔ TAY ARDUINO

Một vài thông số của Arduino Nano:

Vi điều khiển	ATmega328 (họ 8bit)
Điện áp hoạt động	5V – DC
Tần số hoạt động	16 MHz
Dòng tiêu thụ	30mA
Điện áp vào khuyến dùng	7-12V – DC
Điện áp vào giới hạn	6-20V – DC
Số chân Digital I/O	14 (6 chân PWM)
Số chân Analog	8 (độ phân giải 10bit)
Dòng tối đa trên mỗi chân I/O	40 mA
Dòng ra tối đa (5V)	500 mA
Dòng ra tối đa (3.3V)	50 mA
Bộ nhớ flash	32 KB (ATmega328) với 2KB dùng bởi bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Kích thước	1.85cm x 4.3cm

NANO PINOUT



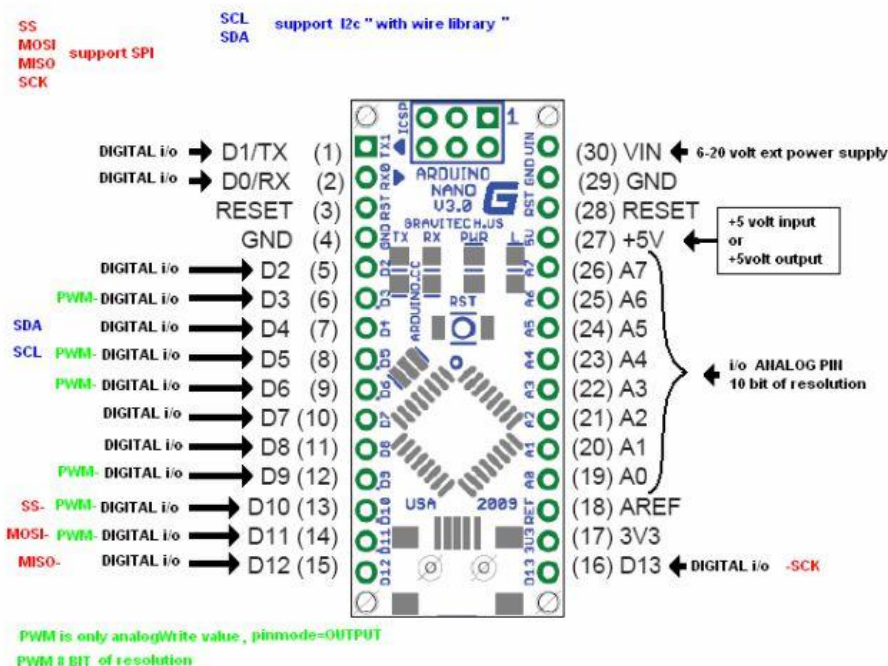
Các thông số kỹ thuật của Arduino Nano hầu như giống hoàn toàn arduino uno r3, vì vậy các thư viện trên Arduino Uno đều hoạt động tốt trên Arduino Uno. Tuy nhiên, ở Nano có một lợi thế cực kỳ quan trọng, nhờ đó Arduino Nano đã được ứng dụng rất nhiều trong các dự án DIY, đó chính là kích thước của nó. Đồng thời Nano còn số lượng chân Analog

SỔ TAY ARDUINO

nhiều hơn Uno (2 chân A6, A7 chỉ dùng để đọc) cùng với dòng ra tối đa của mỗi chân IO lên đến 40mA. Nhưng, có một điểm trừ nhẹ cho Nano, đó là mạch này Nano cần đến 2KB bộ nhớ cho bootloader (ở Uno là 0.5KB). Tuy nhiên, bạn đừng lo lắng, bạn còn đến tận 30KB bộ nhớ flash để lập trình, để dùng hết được 30KB này với tôi, đó là cả "một vấn đề lập trình"

Cổng kết nối với Arduino Nano

Khác với Arduino uno sử dụng cổng USB type B, Nano lại sử dụng một cổng nhỏ hơn có tên là mini USB. Vì sử dụng cổng này nên kích thước board (về chiều cao) cũng giảm đi khá nhiều



Lập trình cho Arduino Nano

Cũng tương tự như bên Arduino uno r3, Arduino Nano sử dụng chương trình Arduino IDE để lập trình, và ngôn ngữ lập trình cho Arduino cũng tên là Arduino (được xây dựng trên ngôn ngữ C). Tuy nhiên, nếu muốn lập trình cho Arduino Nano, bạn cần phải thực hiện một số thao tác trên máy tính. Sau đây, tôi sẽ hướng dẫn bạn từng bước để có thể lập trình cho Arduino Nano.

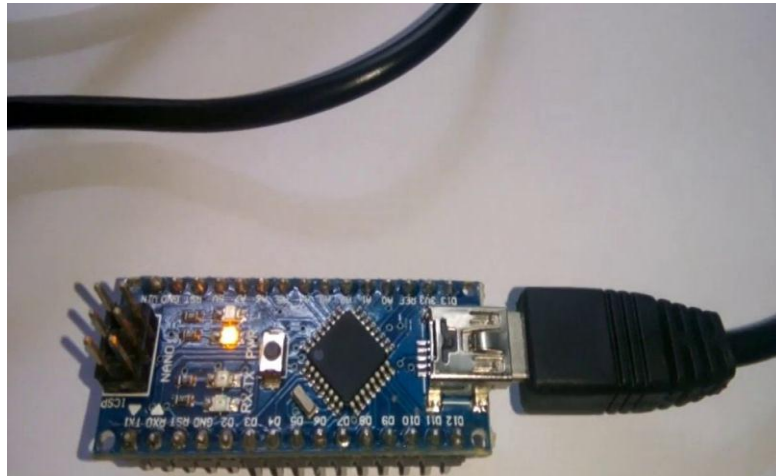
1. Đầu tiên, bạn cần cài Driver của Arduino Nano và tải về bản Arduino IDE mới nhất cho máy tính, các bước cài đặt hoàn toàn tương tự như Arduino Uno R3, bạn có thể tham khảo tại đây: <http://tdhshop.com.vn/cai-dat-chuong-trinh-arduino-ide-va-driver-cho-may-tinh>

SỔ TAY ARDUINO

2. Trường hợp máy tính không nhận cổng COM bạn có thể cài thêm driver **CH340G**

Cho máy tính. Link tải: <http://www.mediafire.com/file/5xwofcfc1n8xsm/CH341SER.zip>

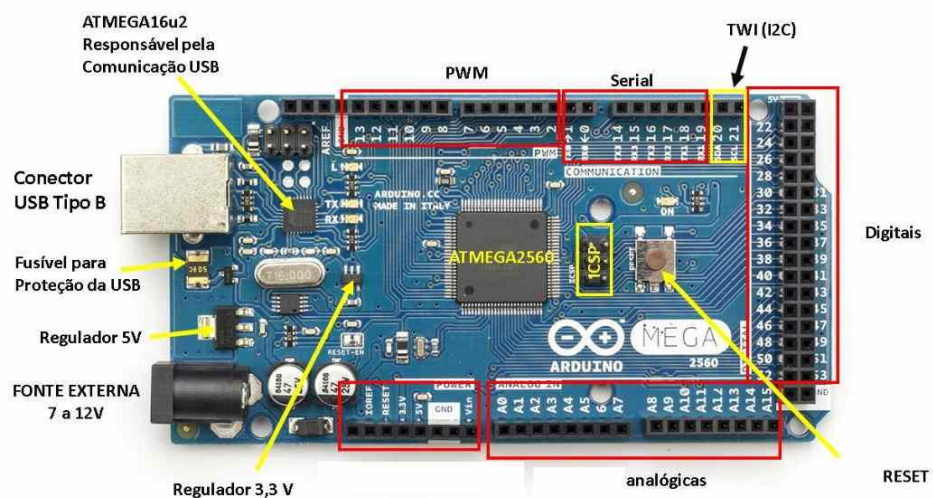
Sau đó, bạn cắm lại board lúc đó máy tính sẽ nhận cổng COM kết nối với arduino nano.



Sau khi hoàn thành quá trình cài đặt, nếu muốn quay lại lập trình cho Arduino Uno, thì bạn chỉ cần chỉnh tên board là Arduino Uno và "Serial Port" thành cổng Serial mà con Uno của bạn đang kết nối.

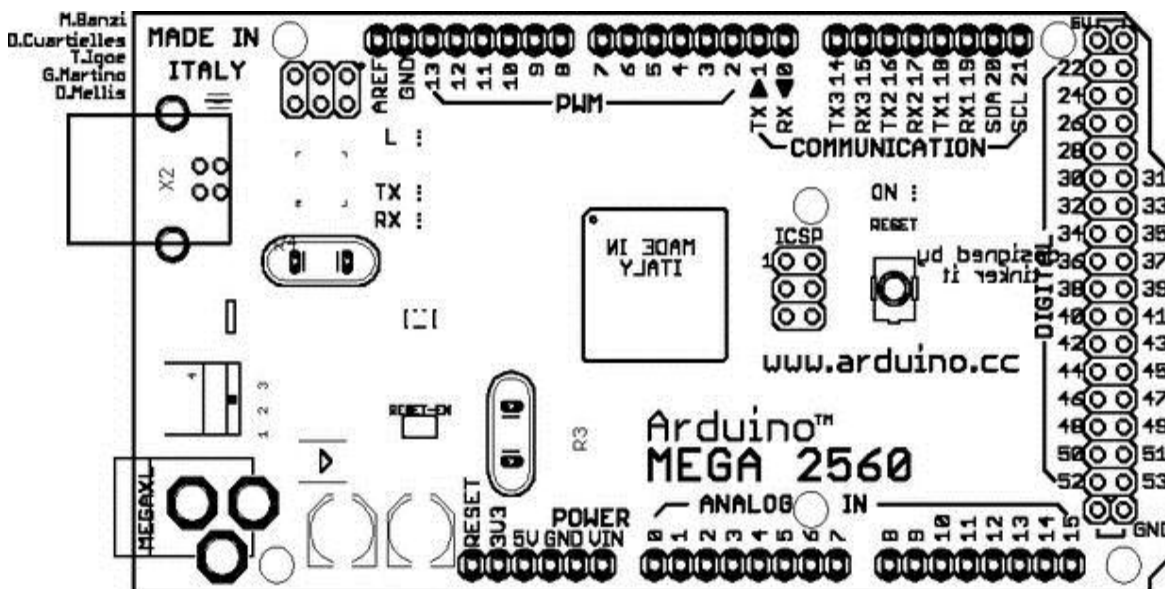
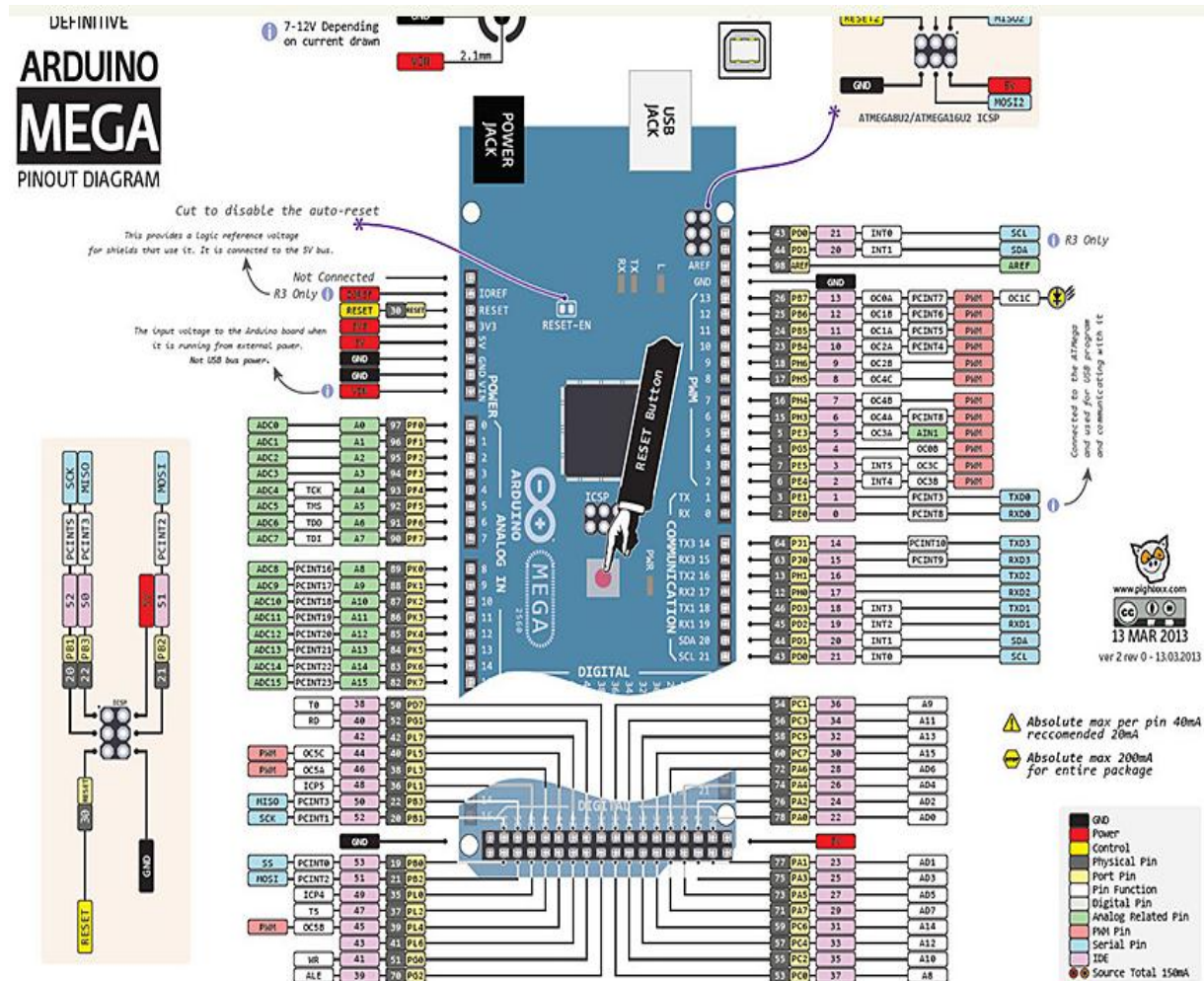
4. Arduino Mega 2560:

Chắc hẳn các bạn đã quá quen thuộc với Arduino Uno R3 rồi. Hôm nay bài viết này sẽ giới thiệu cho các bạn một loại Arduino mới có ứng dụng nhiều hơn và được sử dụng rộng rãi hơn: Arduino Mega2560.



SỔ TAY ARDUINO

Thành phần Arduino Mega:



Web: tdhshop.com.vn – Chuyên Kit Tự Học Arduino từ cơ bản đến nâng cao

SỔ TAY ARDUINO

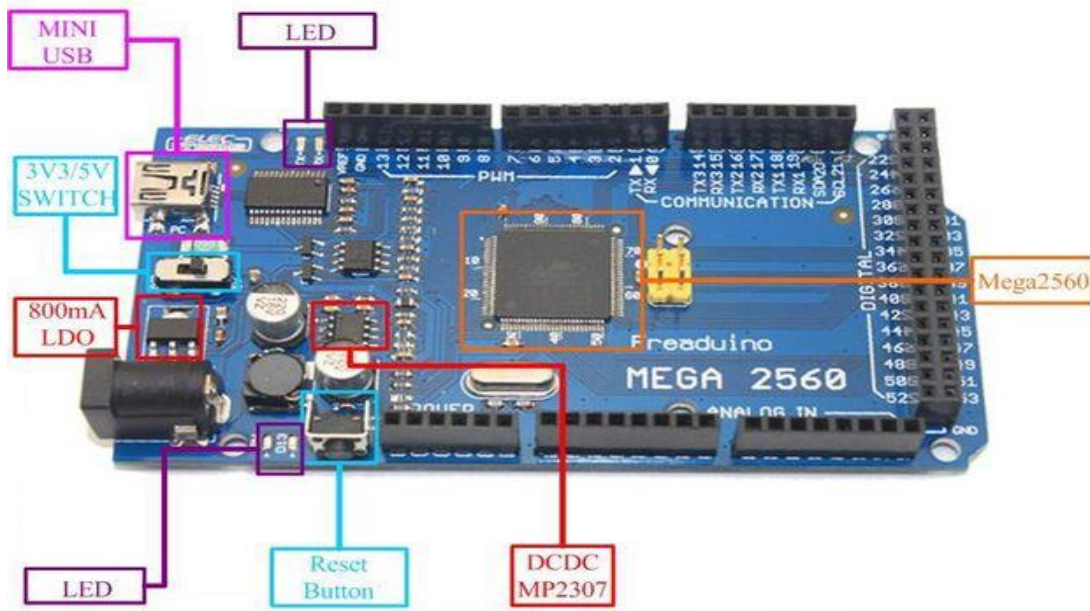
Arduino Mega2560 là một vi điều khiển bằng cách sử dụng ATmega2560.

Bao gồm:

- 54 chân digital (15 có thể được sử dụng như các chân PWM)
- 16 đầu vào analog,
- 4 UARTs (cổng nối tiếp phân cứng),
- 1 thạch anh 16 MHz,
- 1 cổng kết nối USB,
- 1 jack cắm điện,
- 1 đầu ICSP,
- 1 nút reset.

Nó **chứa tất cả mọi thứ** cần thiết để hỗ trợ các vi điều khiển.

Arduino Mega2560 khác với tất cả các vi xử lý trước giờ vì không sử dụng FTDI chip điều khiển chuyển tín hiệu từ USB để xử lý. Thay vào đó, nó sử dụng ATmega16U2 lập trình như là một công cụ chuyển đổi tín hiệu từ USB. Ngoài ra, Arduino Mega2560 **cơ bản vẫn giống Arduino Uno R3**, chỉ khác số lượng chân và nhiều tính năng mạnh mẽ hơn, nên các bạn vẫn có thể lập trình cho con vi điều khiển này bằng chương trình lập trình cho Arduino Uno R3.



- 5 Chân GND
- 3 chân 5V

SỔ TAY ARDUINO

- 1 chân 3.3v
- 1 nút reset
- 16 chân analog
- 4 chân UART
- 54 Chân digital trong đó có 15 chân chúng ta có thể sử dụng như PWM
- 6 Chân lập trình ISP

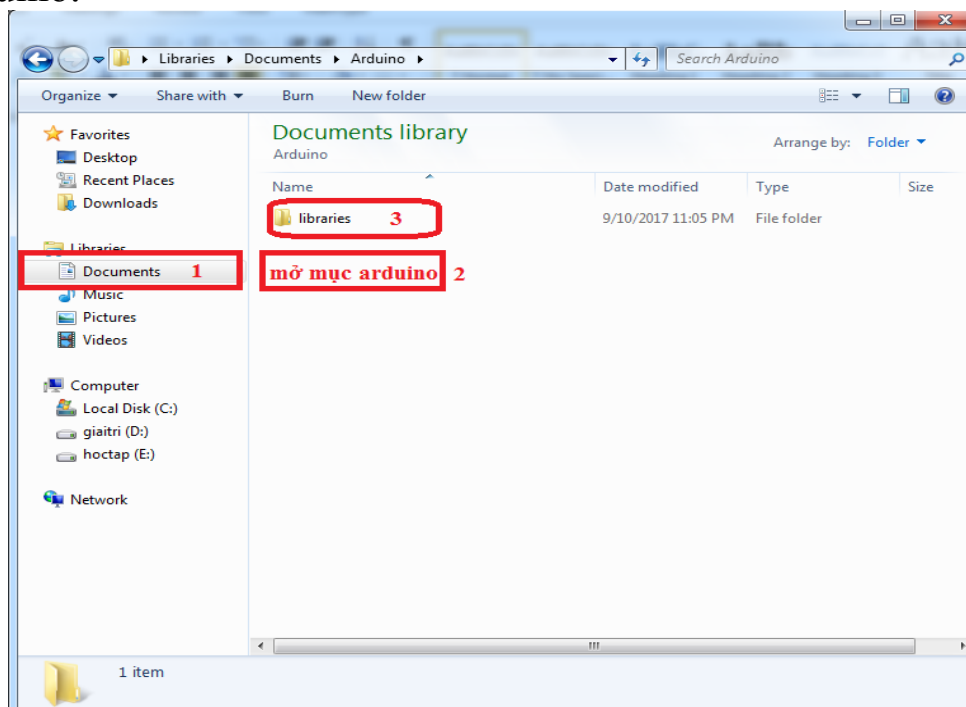
Và nhiều thành phần khác...

Lập trình cho arduino mega 2560 giống hệt như Arduino uno r3 chỉ khác ở chỗ các bạn chọn board là **arduino mega 2560**.

5. Hướng dẫn cài thư viện vào arduino IDE, Hỗ trợ cho việc lập trình:

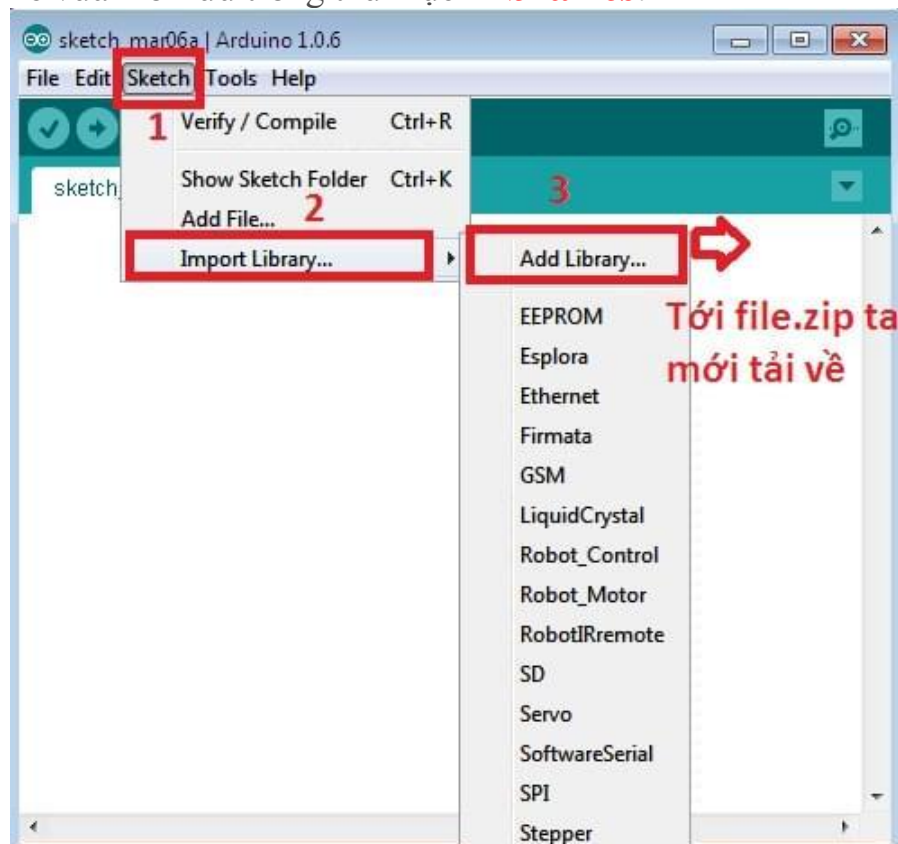
Thư viện trong Arduino chứa các mã nguồn có những đặc điểm chung, được xây dựng thành một gói bao gồm file:examples, .h, .cpp,...Nhằm giúp người sử dụng giải quyết được vấn đề nhanh chóng, trong bài viết này tôi sử dụng thư viện **I2C** làm ví dụ để cài đặt vào Arduino IDE.

Lưu ý: Tất cả thư viện của arduino ta tải về nên lưu bên trong thư mục **Libraries** của arduino.



SỔ TAY ARDUINO

Sau khi tải thư viện **I2C** về với file .zip, ta vào Sketch/ Include Library/Add sau đó chọn file vừa mới lưu trong thư mục **Libraries**:



Các bước cài thư viện vào Arduino IDE

Nếu cửa sổ Arduino IDE báo dòng chữ dưới đây thì đã cài thành công.

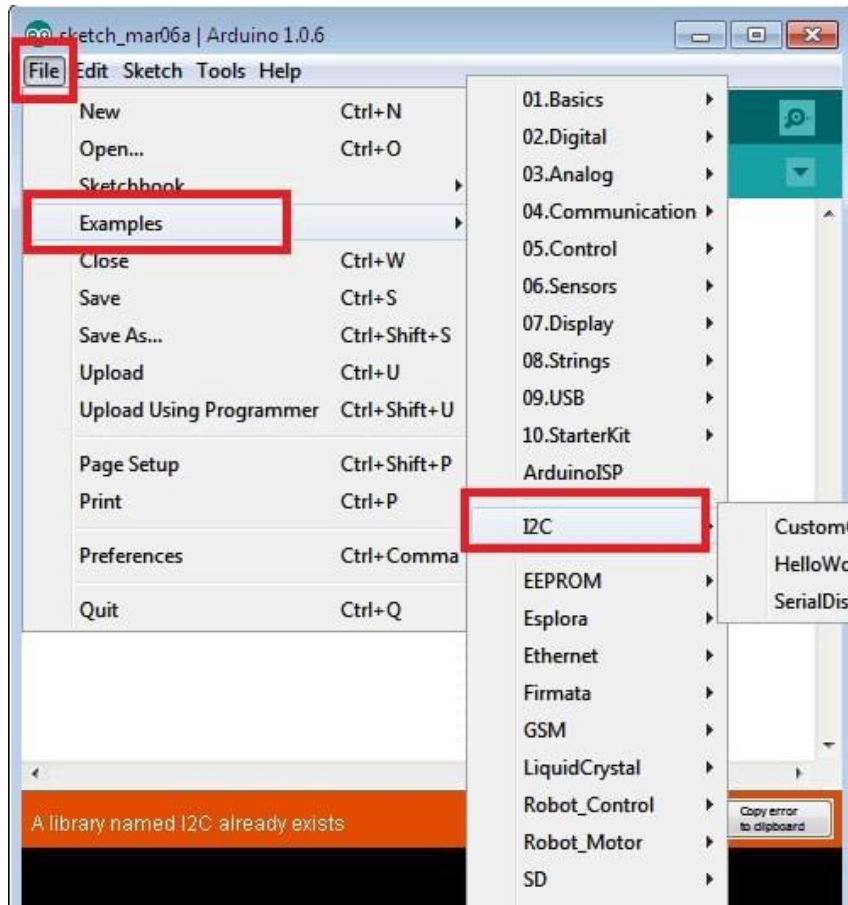


Dòng trong ô đỏ là báo đã cài thư viện vào arduino IDE thành công

Tất cả thư viện của arduino IDE ta nên đặt trong 1 file.

Cài xong ta reset lại Arduino IDE và xem lại thư viện **I2C** đã có trong IDE chưa.

SỔ TAY ARDUINO



Như vậy là đã cài xong thư viện I2C vào arduino IDE
CHÚC CÁC BẠN THÀNH CÔNG!!!

SỔ TAY ARDUINO

6. Ngôn Ngữ Lập Trình Arduino: gồm 3 phần chính là **Câu lệnh cấu trúc – Giá trị - Hàm và thủ tục**

a. Câu lệnh cấu trúc:

SETUP() và LOOP()

Những lệnh trong `setup()` sẽ được chỵ khi chương trình của bạn khởi động.

Bạn có thể sử dụng nó để khai báo giá trị của biến, khai báo thư viện, thiết lập các thông số,...

Sau khi `setup()` chạy xong, những lệnh trong `loop()` sẽ được chạy.

Chúng sẽ lặp đi lặp lại liên tục cho tới khi nào bạn ngắt nguồn của board Arduino mới thôi.

Bất cứ khi nào bạn nhất nút Reset, chương trình của bạn sẽ trở về lại trạng thái như khi Arduino mới được cấp nguồn.

Ví dụ

```
int led = 10;
void setup() {
  pinMode(led, OUTPUT);
}
void loop() {
  digitalWrite(led, HIGH);
  delay(5000);
  digitalWrite(led, LOW);
  delay(5000);
}
```

SỔ TAY ARDUINO

Giải thích:

Khi bạn cấp nguồn cho Arduino, lệnh “*pinMode(led, OUTPUT);*” sẽ được chạy 1 lần để khai báo.

Sau khi chạy xong lệnh ở *setup()*, lệnh ở *loop()* sẽ được chạy và được lặp đi lặp lại liên tục, tạo thành một chuỗi:

```
digitalWrite(led, HIGH);  
delay(5000);  
digitalWrite(led, LOW);  
delay(5000);  
digitalWrite(led, HIGH);  
delay(5000);  
digitalWrite(led, LOW);  
delay(5000);  
digitalWrite(led, HIGH);  
delay(5000);  
digitalWrite(led, LOW);  
delay(5000);  
.....
```

if...else

Trước khi đọc lệnh *if ... else* thì chúng ta phải tìm hiểu về một số toán tử SAU:

Toán tử logic:

Toán tử	Ý nghĩa	Ví dụ tham khảo
and (<i>&&</i>)	Và	(<i>a && b</i>) trả về TRUE nếu a và b đều mang giá trị TRUE. Nếu một trong a hoặc b là FALSE thì (<i>a && b</i>) trả về FALSE

SỔ TAY ARDUINO

or ()	Hoặc	(a b) trả về TRUE nếu có ít nhất 1 trong 2 giá trị a và b là TRUE, trả về FALSE nếu a và b đều FALSE
not (!)	Phủ định	Nếu a mang giá trị TRUE thì (!a) là FALSE và ngược lại
xor (^)	Loại trừ	(a ^ b) trả về TRUE nếu a và b mang hai giá trị TRUE/FALSE khác nhau, các trường hợp còn lại trả về FALSE

Toán tử SO SÁNH:

Các toán tử so sánh thường dùng để so sánh 2 số có cùng một kiểu dữ liệu.

Toán tử	Ý nghĩa	ví dụ
==	so sánh bằng	(a == b) trả về TRUE nếu a bằng b và ngược lại
!=	so sánh không bằng	(a != b) trả về TRUE nếu a khác b và ngược lại
>	so sánh lớn	(a > b) trả về TRUE nếu a lớn hơn b và FALSE nếu a bé hơn hoặc bằng b
<	so sánh bé	(a < b) trả về TRUE nếu a bé hơn b và FALSE nếu ngược lại
<=	Bé hơn hoặc bằng	(a <= b) tương đương với ((a < b) or (a = b))
>=	Lớn hơn hoặc bằng	(a >= b) tương đương với ((a > b) or (a = b))

Câu lệnh if

Cú pháp:

```
if ([biểu thức 1] [toán tử so sánh] [biểu thức 2]) { //biểu thức điều kiện
    [câu lệnh 1]
} else {
```


SỔ TAY ARDUINO

```
[câu lệnh 2]
```

```
}
```

Nếu biểu thức điều kiện trả về giá trị TRUE, [câu lệnh 1] sẽ được thực hiện, ngược lại, [câu lệnh 2] sẽ được thực hiện.

Ví dụ:

```
int a = 0;
if (a == 0) {
    a = 10;
} else {
    a = 1;
}
// a = 10
```

Lệnh if không bắt buộc phải có nhóm lệnh nằm sau từ khóa else

```
int a = 0;
if (a == 0) {
    a = 10;
}
// a = 10
```

Bạn có thể kết hợp nhiều biểu thức điều kiện khi sử dụng lệnh if. Chú ý rằng mỗi biểu thức con phải được bao bằng một ngoặc tròn và phải luôn có một cặp ngoặc tròn bao toàn bộ biểu thức con.

switch / case:

Mô tả lệnh switch / case:

SỔ TAY ARDUINO

Giống như if, switch / case cũng là một dạng lệnh nếu thì, nhưng nó được thiết kế chuyên biệt để bạn xử lý giá trị trên một biến chuyên biệt.

Ví dụ, bạn có một biến là action sẽ nhận trị từ những module khác qua serial. Nhưng action sẽ nằm trong một các giá trị nào đó thì lúc này bạn hãy sử dụng switch / case.

Cú pháp lệnh switch / case

```
switch (var) {  
  case label:  
    //đoạn lệnh  
    break;  
  case label:  
    // Đoạn lệnh  
    break;  
  /*  
  case ... more and more  
  */  
  default:  
    // statements  
}
```

Tham số lệnh switch / case

Var : biến mà bạn muốn so sánh

label: sẽ đem giá trị của biến SO SÁNH BẰNG với nhãn này

SỔ TAY ARDUINO

Hàm For:

Giới thiệu hàm for:

Hàm for có chức năng làm một vòng lặp. Vậy vòng lặp là gì? Hãy hiểu một cách đơn giản, nó làm đi làm lại một công việc có một tính chất chung nào đó. Chẳng hạn, bạn bật tắt một con LED thì dùng digitalWrite xuất HIGH delay rồi lại LOW rồi lại delay. Nhưng nếu bạn muốn làm nhiều hơn 1 con LED thì mọi đoạn code của bạn sẽ dài ra (không đẹp và khi chỉnh sửa thì chẳng lẽ ngồi sửa lại từng dòng?)

Với 1 con led, bạn lập trình như thế này:

```
digitalWrite(led1,HIGH);  
delay(1000);  
digitalWrite(led1,LOW);  
delay(1000);
```

Với 10 con led, nếu bạn không dùng for, đoạn code nó sẽ dài như thế này:

```
digitalWrite(led1,HIGH);  
delay(1000);  
digitalWrite(led1,LOW);  
delay(1000);  
digitalWrite(led2,HIGH);  
delay(1000);  
digitalWrite(led2,LOW);  
delay(1000);  
...  
digitalWrite(led10,HIGH);  
delay(1000);  
digitalWrite(led10,LOW);  
delay(1000);
```

Bây giờ chúng ta hãy lấy một ví dụ đơn giản khác:

SỔ TAY ARDUINO

Tôi muốn xuất 10 chữ số (từ 1 - 10) ra Serial. Và viết đó thực hiện như sau

Nếu bạn chưa đọc bài này và cũng chưa biết kiến thức về for, bạn sẽ lập trình như sau nếu bạn không biết câu lệnh **for**:

```
void setup() {  
    Serial.begin(9600);  
    Serial.println(1);  
    Serial.println(2);  
    Serial.println(3);  
    Serial.println(4);  
    Serial.println(5);  
    Serial.println(6);  
    Serial.println(7);  
    Serial.println(8);  
    Serial.println(9);  
    Serial.println(10);  
}  
void loop() {  
    // không làm gì;  
}
```

Cấu trúc lệnh for:

Theo quan điểm của tôi, nếu bạn chưa biết về vòng lặp hoặc hàm for, để hiểu được hàm for, bạn cần nắm được 4 phần:

1. Hàm for là một vòng lặp có giới hạn - nghĩa là chắc chắn nó sẽ kết thúc (không sớm thì muộn).
2. Nó sẽ bắt đầu từ một vị trí xác định và đi đến một vị trí kết thúc.
3. Cứ mỗi bước xong, nó lại thực hiện một đoạn lệnh
4. Sau đó, nó lại bước đi tiếp, nó có thể bước 1 bước hoặc nhiều bước, nhưng không được thay đổi theo thời gian.

SỔ TAY ARDUINO

Theo ví dụ trên, ta có đoạn code sử dụng hàm for như sau:

```
for (i = 1; i <= 10; i = i + 1) {  
    Serial.println(i);  
}
```

Gồm có 2 lệnh for là tiến và for lùi:

- For tiến (xuất phát từ một vị trí nhỏ chạy đến vị trí lớn hơn) <vị trí xuất phát> bé hơn <vị trí kết thúc>
- For lùi (xuất phát từ một vị trí lớn chạy về vị trí nhỏ hơn) <vị trí xuất phát> lớn hơn <vị trí kết thúc>

Và khi đã hiểu được một cách sâu sắc thì đây là cú pháp chính của hàm For:

```
for (<biến chạy> = <start>; <điều kiện>; <bước>) {  
    //lệnh  
}
```

Lệnh **while**:

Giới thiệu lệnh cấu trúc while

Vòng lặp while là một dạng vòng lặp theo điều kiện, mình không thể biết trước số lần lặp của nó nhưng mình quản lý lúc nào thì nó ngừng lặp!

Giống như **lệnh for**, cũng có vài khái niệm mà bạn cần nắm

Cú pháp lệnh cấu trúc while

```
while (<điều kiện>) {  
    //các đoạn lệnh;  
}
```

SỔ TAY ARDUINO

Ví dụ lệnh cấu trúc while

```
int day = 1;
int nam = 2014; // Năm 2014
while (day < 365) { //Chừng nào day < 365 thì còn chạy (<=364). Khi day == 365 thì
hết 1 năm...
    day += 1; //
    delay(60*60*24); // Một ngày có 24 giờ, mỗi giờ có 60 phút, mỗi phút có 60 giây
}
nam += 1; //... bây giờ đã là một năm mới !
```

Lệnh Continue:

Giới thiệu lệnh cấu trúc continue

continue là một lệnh có chức năng bỏ qua một chu kì lặp trong một vòng lặp (**for**, **do**, **while**) chứa nó trong đó. Khi gọi lệnh continue, những lệnh sau nó và ở trong cùng vòng lặp với nó sẽ bị bỏ qua để thực hiện những chu kì lặp kế tiếp.

Ví dụ lệnh cấu trúc continue

```
int a = 0;
int i = 0;
while (i < 10) {
    i = i + 1;
    continue;
    a = 1;
}
//a vẫn bằng 0
```

SỔ TAY ARDUINO

Lệnh **Break**:

Giới thiệu lệnh cấu trúc Break

break là một lệnh có chức năng dừng ngay lập tức một vòng lặp (do, for, while) chứa nó trong đó. Khi dừng vòng lặp, tất cả những lệnh phía sau **break** và ở trong vòng lặp chịu ảnh hưởng của nó sẽ bị bỏ qua.

Ví dụ lệnh cấu trúc Break

```
int a = 0;
while (true) {
    if (a == 5) break;
    a = a + 1;
}
//a = 5
```

```
while (true) {
    while (true) {
        a++;
        if (a > 5) break;
    }
    a++;
    if (a > 100) break;
}
//a = 101
```

SỔ TAY ARDUINO

Lệnh **Return**:

Giới thiệu lệnh cấu trúc return

return có nhiệm vụ trả về một giá trị (cùng kiểu dữ liệu với hàm) mà nó được gọi!

Cú pháp lệnh cấu trúc return

```
return;  
return value; // cả 2 đều đúng
```

Thông số

value: bất kỳ giá trị hoặc một đối tượng.

Ví dụ lệnh cấu trúc return

```
//Hàm kiểm tra giá trị của cảm biến có hơn một ngưỡng nào đó hay không  
int checkSensor(){  
    if (analogRead(0) > 400) {  
        return 1;  
    }  
    else{  
        return 0;  
    }  
}
```


SỔ TAY ARDUINO

Lệnh goto:

Giới thiệu Lệnh cấu trúc goto

Nó có nhiệm vụ tạm dừng chương trình rồi chuyển đến một nhãn đã được định trước, sau đó lại chạy tiếp chương trình!

Cú pháp Lệnh cấu trúc goto

```
label: //Khai báo một nhãn có tên là label  
goto label; //Chạy đến nhãn label rồi sau đó thực hiện tiếp những đoạn chương trình sau nhãn đó
```

Thủ thuật

Không nên dùng lệnh goto trong chương trình Program hay bất cứ chương trình nào sử dụng ngôn ngữ C. Nhưng nếu sử dụng một cách khôn ngoan bạn sẽ tối ưu hóa được nhiều điều trong một chương trình!

Vậy nó hữu ích khi nào, đó là lúc bạn đang dùng nhiều vòng lặp quá và muốn thoát khỏi nó một cách nhanh chóng!

Ví dụ Lệnh cấu trúc goto

```
for(byte r = 0; r < 255; r++){  
    for(byte g = 255; g > -1; g--){  
        for(byte b = 0; b < 255; b++){  
            if (analogRead(0) > 250){ goto bailout;}  
            //thêm nhiều câu lệnh nữa  
        }  
    }  
}bailout:
```

SỔ TAY ARDUINO

Cú pháp mở rộng {} dấu ngoặc nhọn:

Giới thiệu Cú pháp mở rộng {} dấu ngoặc nhọn

Rất đơn giản, {} là một cặp dấu ngoặc nhọn, nên một khi bạn đã mở ngoặc thì phải đóng ngoặc lại cho nó!

Nhiệm vụ của nó là cung cấp một cú pháp để gọi những lệnh cho những cấu trúc đặc biệt như (if, **while**, **for**,...)

Tốt nhất là các bạn xem các ví dụ sau đây, thì chắc chắn sẽ hiểu được điều mà {} muốn nói!

Cách sử dụng Cú pháp mở rộng {} dấu ngoặc nhọn

Trong hàm và thủ tục

```
void myfunction(<kiểu dữ liệu> <tham số>){  
    <lệnh>;  
}
```

Vòng lặp

```
while (<điều kiện>)  
{  
    <câu lệnh>  
}  
  
do  
{  
    <câu lệnh>  
} while (<điều kiện>;
```

SỔ TAY ARDUINO

```
for (<khởi tạo>; <điều kiện>; <bước>)  
{  
  <câu lệnh>  
}
```

Lệnh rẽ nhánh

```
if (<điều kiện 1>)  
{  
  <lệnh 1>  
}  
  
else if (<điều kiện 2>)  
{  
  <lệnh 2>  
}  
  
else  
{  
  <lệnh 3>  
}
```

SỔ TAY ARDUINO

Comments - Viết ghi chú trong khi viết

Giới thiệu về Comments - Viết ghi chú trong khi viết code Arduino

Bạn rất khó ghi nhớ từng dòng code một trong một chương trình thật là dài, với những thuật toán phức tạp, vì vậy Arduino đã làm cho bạn một cú pháp để giải quyết vấn đề này, đó là Comments. Comments sẽ giúp bạn ghi chú cho từng dòng code hoặc trình bày nhiệm vụ của nó để bạn hoặc những người khác có thể hiểu được chương trình này làm được những gì. Và comments sẽ không được Arduino biên dịch nên cho dù bạn viết nó dài đến đâu thì cũng không ảnh hưởng đến bộ nhớ flash của vi điều khiển. Để comments trong Arduino, bạn có 2 cách.

Ví dụ về Comments - Viết ghi chú trong khi viết code Arduino

```
x = 5; // Đây là kiểu "single line comment", để làm được điều này, bạn gõ "//"
        // nó sẽ ghi chú tất cả những chữ (text, câu lệnh,... everything) nằm sau dấu //
cho đến khi hết dòng

/* Còn đây là "multiline comment" - Bạn bắt đầu ghi chú với ký tự kia.
Nó sẽ "ghi chú" tất cả những gì nằm trong cặp dấu "/" "*" và "*" "/" ( không có dấu cách
nhé)
if (gwb == 0){ // ngoài ra bạn có thể dùng single line trong này.
x = 3;        /* nhưng dùng một multiline comment khác thì sẽ bị lỗi cú pháp ngay
*/
}
}
```

SỔ TAY ARDUINO

b. Hàm GIÁ TRỊ trong chương trình Arduino:

Kiểu dữ liệu **int**

Kiểu **int** là kiểu số nguyên chính được dùng trong chương trình Arduino.

Kiểu **int** chiếm 2 byte bộ nhớ !

Trên mạch Arduino Uno, nó có đoạn giá trị từ -32,768 đến 32,767 (-2^{15} đến $2^{15}-1$) (16 bit)

Trên mạch Arduino Due, nó có đoạn giá trị từ -2,147,483,648 đến 2,147,483,647 (-2^{31} đến $2^{31}-1$) (32 bit) (lúc này nó chiếm 4 byte bộ nhớ)

Ví dụ về kiểu dữ liệu **int**:

```
int ledPin = 13;
```

Cú pháp kiểu dữ liệu **int**:

```
int var = val;
```

var: tên biến

val: giá trị

Kiểu dữ liệu **float**:

Để định nghĩa 1 kiểu số thực, bạn có thể sử dụng kiểu dữ liệu **float**.

Một biến dùng kiểu dữ liệu này có thể đặt một giá trị nằm trong khoảng -3.4028235E+38 đến 3.4028235E+38. Nó chiếm 4 byte bộ nhớ.

SỔ TAY ARDUINO

Với kiểu dữ liệu float bạn có từ 6-7 chữ số có nghĩa nằm ở bên mỗi bên dấu ".". Điều đó có nghĩa rằng bạn có thể đặt một số thực dài đến 15 ký tự (bao gồm dấu .)

Lưu ý trong sử dụng kiểu dữ liệu float:

Để biểu diễn giá trị thực của một phép chia bạn phải 2 số thực chia cho lẫn nhau. Ví dụ: bạn xử lý phép tính $5.0 / 2.0$ thì kết quả sẽ trả về là 2.5. Nhưng nếu mà bạn xử lý phép tính $5 / 2$ thì kết quả sẽ là 2 (vì hai số nguyên chia nhau sẽ ra một số nguyên).

Ví dụ về kiểu dữ liệu float:

```
float myfloat;  
float sensorCalbrate = 1.117;
```

Cú pháp kiểu dữ liệu float:

```
float var = val;
```

var: tên biến

val: giá trị

Code tham khảo sử dụng kiểu dữ liệu float:

```
int x;  
int y;  
float z;  
x = 1;  
y = x / 2;           // y sẽ trả về kết quả là 0  
z = (float)x / 2.0; //z sẽ có kết quả là 0.5 (bạn nhập 2.0, chứ không phải là 2)
```

SỔ TAY ARDUINO

Kiểu dữ liệu **char**:

Kiểu dữ liệu char là kiểu dữ liệu biểu diễn cho 1 ký tự.

Nếu bạn cần biểu diễn một chuỗi trong chương trình Arduino - bạn cần sử dụng kiểu dữ liệu String. Kiểu dữ liệu này chiếm 1 byte bộ nhớ!

Kiểu char chỉ nhận các giá trị trong bảng mã ASCII.

Kiểu char được lưu dưới dạng 1 số nguyên byte có số âm (có các giá trị từ -127 - 128), thay vì thiết đặt một biến kiểu char có giá trị là 'A', bạn có thể đặt là 65. Để hiểu rõ hơn bạn xem ví dụ dưới đây.

Ví dụ về kiểu dữ liệu char:

```
char myChar = 'A';  
char myChar = 65; // cả 2 cách khai báo đều hợp lệ
```

Kiểu dữ liệu **void**:

Mô tả về kiểu dữ liệu Void

"void" là một từ khóa chỉ dùng trong việc khai báo một function. Những function được khai báo với "void" sẽ không trả về bất kì dữ liệu nào khi được gọi.

Ví dụ về kiểu dữ liệu Void

```
led = 13;  
void setup() {  
  pinMode(led, OUTPUT);  
}
```

SỔ TAY ARDUINO

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

Giải thích

- "blink" là một function được định nghĩa với từ khóa "void", do đó nó không trả về một giá trị nào. Nhiệm vụ của "blink" chỉ là làm nhấp nháy đèn LED ở chân số 13 trên mạch Arduino.
- Bạn có thể thấy rằng những function kiểu này không dùng lệnh "return" để trả về giá trị của function.

Kiểu dữ liệu **byte**:

Giới thiệu về kiểu dữ liệu byte

Là một kiểu dữ liệu biểu diễn số nguyên nằm trong khoảng từ 0 đến 255. Bạn sẽ mất 1 byte bộ nhớ cho mỗi biến mang kiểu *byte*

Ví dụ về kiểu dữ liệu byte

```
byte a = 123; //khai báo biến a mang kiểu byte, có giá trị là 123
```


SỔ TAY ARDUINO

Kiểu dữ liệu **string**:

Giới thiệu về kiểu dữ liệu string

string tiếng Anh nghĩa là chuỗi. Trong một chương trình Arduino có 2 cách để định nghĩa chuỗi, cách thứ nhất là sử dụng mảng ký tự để biểu diễn chuỗi. Bài viết này xin mô tả chi tiết về cách thứ nhất.

Cách khai báo về kiểu dữ liệu string

```
char Str1[15]; // khai báo chuỗi có độ dài là 15 ký tự.
```

```
char Str2[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o'}; //khai báo chuỗi có độ dài tối đa là 8 ký tự và đặt nó giá trị ban đầu là arduino (7 ký tự). Buộc phải khai báo chuỗi nằm giữa hai dấu nháy đơn nhé!
```

```
char Str3[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o', '\0'}; //khai báo chuỗi có độ dài tối đa là 8 ký tự và đặt nó giá trị ban đầu là arduino<ký tự null> (8 ký tự)
```

```
char Str4[ ] = "arduino"; // Chương trình dịch sẽ tự động điều chỉnh kích thước cho chuỗi Str4 này và ngoài ra bạn phải đặt một chuỗi trong dấu ngoặc kép
```

```
char Str5[8] = "arduino"; // Một cách khai báo như Str3
```

```
char Str6[15] = "arduino"; // Một cách khai báo khác với độ dài tối đa lớn hơn
```

CHÚ Ý: mỗi chuỗi đều cần có 1 ký tự NULL, nếu bạn không khai báo ký tự NULL (\0) ở cuối thì trình biên dịch sẽ tự động thêm vào. Đó là lý do vì sao Str2, Str4 lại có độ dài là 8 nhưng chỉ chứa một chuỗi 7 ký tự. Ký tự NULL này để làm gì? Nó dùng để trình biên dịch biết điểm dừng của một chuỗi! Nếu không nó sẽ đọc tiếp những phần bộ nhớ khác (mà phần ấy không lưu chuỗi)

SỔ TAY ARDUINO

Bạn có thể khai báo một chuỗi dài như sau:

```
char myString[] = "This is the first line"  
" this is the second line"  
" etcetera"
```

Mảng chuỗi

Khi cần phải thao tác với một lượng lớn chuỗi (ví dụ như trong các ứng dụng trả lời người dùng bằng LCD) thì bạn cần sử dụng một mảng chuỗi. Mà bản chất của chuỗi là mảng các ký tự. Vì vậy để khai báo 1 mảng chuỗi bạn cần sử dụng một mảng 2 chiều!

Để khai báo một mảng chuỗi, rất đơn giản:

```
char* myStrings[] = {"I'm number 1", "I'm number 2"};
```

Chỉ cần thêm dấu * sau chữ char và trong dấu ngoặc vuông phía sau myStrings bạn có thể thiết đặt số lượng phần tử tối đa của mảng chuỗi!

Ví dụ về kiểu dữ liệu string

```
char* myStrings[]={ "This is string 1", "This is string 2", "This is string 3",  
"This is string 4", "This is string 5", "This is string 6"};  
void setup(){  
  Serial.begin(9600);  
}  
void loop(){  
  for (int i = 0; i < 6; i++){  
    Serial.println(myStrings[i]);  
    delay(500);  
  }  
}
```

Kiểu dữ liệu array:

SỔ TAY ARDUINO

Giới thiệu về kiểu dữ liệu mảng array

Array là mảng (tập hợp các giá trị có liên quan và được đánh dấu bằng những chỉ số). Array được dùng trên Arduino chính là Array trong ngôn ngữ lập trình C.

Các cách khởi tạo một mảng

```
int myInts[6]; // tạo mảng myInts chứa tối đa 6 phần tử (được đánh dấu từ 0-5), các phần tử này đều có kiểu là int => khai báo này chiếm 2*6 = 12 byte bộ nhớ
```

```
int myPins[] = {2, 4, 8, 3, 6}; // tạo mảng myPins chứa 5 phần tử (lần lượt là 2, 4, 8, 3, 6). Mảng này không giới hạn số lượng phần tử vì có khai báo là "[]"
```

```
int mySensVals[6] = {2, 4, -8, 3, 2}; // tạo mảng mySensVals chứa tối đa 6 phần tử, trong đó 5 phần tử đầu tiên có giá trị lần lượt là 2, 4, -8, 3, 2
```

```
char message[6] = "hello"; // tạo mảng ký tự (dạng chuỗi) có tối đa 6 ký tự!
```

Truy cập các phần tử trong mảng

Chú ý: Phần tử đầu tiên trong mảng luôn được đánh dấu là **số 0**.

```
mySensVals[0] == 2, mySensVals[1] == 4, vâng vâng
```

Điều này có nghĩa rằng, việc khai báo một mảng có tối đa 10 phần tử, thì phần tử cuối cùng (thứ 10) được đánh dấu là **số 9**

```
int myArray[10]={9,3,2,4,3,2,7,8,9,11};
```

```
// myArray[9] có giá trị là 11
```

```
// myArray[10] sẽ trả về một giá trị "hên xui" nằm trong khoảng giá trị của int
```

Vì vậy, hãy chú ý trong việc truy cập đến giá trị trong mảng, nếu bạn muốn truy cập đến phần tử cuối cùng thì hãy truy cập đến ô giới hạn của mảng - 1.

SỔ TAY ARDUINO

Hãy ghi nhớ rằng, trong trình biên dịch ngôn ngữ C, nó không kiểm tra bạn có truy cập đến một ô có nằm trong bộ nhớ hay không! Nên nếu không cẩn thận trong việc truy cập mảng, chương trình của bạn sẽ mắc lỗi logic và rất khó để tìm lỗi đấy!

Gán một giá trị cho một phần tử

```
mySensVals[0] = 10;
```

Đọc một giá trị của một phần tử và gán cho một biến nào đó cùng kiểu dữ liệu

```
x = mySensVals[0]; //10
```

Dùng mảng trong vòng lặp

Mảng rất thường được dùng trong vòng lặp (chẳng hạn như dùng để lưu các chân digital quản lý đèn led). Trong đó, biến chạy của hàm for sẽ đi hết (hoặc một phần) của mảng, tùy thuộc vào yêu cầu của bạn mà thôi! Ví dụ về việc in 5 phần tử đầu của mảng myPins:

```
int i;  
for (i = 0; i < 5; i = i + 1) {  
    Serial.println(myPins[i]);  
}
```

c. Hàm và Thủ Tục:

Hàm toán học **min()**:

Mô tả về hàm toán học min()

Hàm min có nhiệm vụ trả về giá trị nhỏ nhất giữa hai biến.

Cú pháp về hàm toán học min()

SỔ TAY ARDUINO

```
min(x, y);
```

Tham số

x: số thứ nhất, mọi kiểu dữ liệu đều được chấp nhận.

y: số thứ hai, mọi kiểu dữ liệu đều được chấp nhận.

Trả về

Số nhỏ nhất trong 2 số.

Gợi ý

Hàm min được dùng để lấy chặn trên (không để giá trị vượt quá một mức quy định nào đó).

Cảnh báo cú pháp về hàm toán học min()

```
min(a++, 100); // nếu bạn nhập như thế này thì sẽ bị lỗi đấy
```

```
a++;
```

```
min(a, 100); // nhưng nếu nhập như thế này thì ổn! Và hãy ghi nhớ là không được để bất cứ phép tính nào bên trong.
```

Hàm thời gian **delay()**:

Giới thiệu về hàm thời gian delay

delay có nhiệm vụ dừng chương trình trong thời gian mili giây. Và cứ mỗi 1000 mili giây = 1 giây.

Cú pháp về hàm thời gian delay

```
delay(ms)
```

SỔ TAY ARDUINO

Thông số

ms: thời gian ở mức mili giây. ms có kiểu dữ liệu là **unsigned long**

Ví dụ về hàm thời gian delay

Điều khiển đèn LED sáng nhấp nháy theo yêu cầu bằng arduino uno R3.

```
void setup() {  
  pinMode(13, OUTPUT);  
}  
  
void loop() {  
  digitalWrite(13, HIGH); // turn the LED on  
  delay(1000);           // led sáng trong vòng 1 giây.  
  digitalWrite(13, LOW); // turn the LED off  
  delay(1000);          // led tắt trong vòng 1 giây.  
}
```

Hàm thời gian **micros()**:

Giới thiệu về hàm thời gian micros()

micros() có nhiệm vụ trả về một số - là thời gian (tính theo micro giây) kể từ lúc mạch Arduino bắt đầu chương trình của bạn. Nó sẽ tràn số và quay số 0 (sau đó tiếp tục tăng) sau 70 phút. Tuy nhiên, trên mạch Arduino 16MHz (ví dụ Duemilanove và Nano) thì giá trị của hàm này tương đương 4 đơn vị micro giây. Ví dụ micros() trả về giá trị là 10 thì có nghĩa chương trình của bạn đã chạy được

SỔ TAY ARDUINO

40 microgiây. Tương tự, trên mạch 8Mhz (ví dụ LilyPad), hàm này có giá trị tương đương 8 micro giây.

Lưu ý: 10^6 micro giây = 1 giây

Trả về

một số nguyên kiểu **unsigned long** là thời gian kể từ lúc thương trình Arduino được khởi động.

Ví dụ về hàm thời gian micros()

```
unsigned long time;

void setup(){
  Serial.begin(9600);
}

void loop(){
  Serial.print("Time: ");
  time = micros();
  // in ra thời gian kể từ lúc chương trình được bắt đầu
  Serial.println(time);
  // đợi 1 giây trước khi tiếp tục in
  delay(1000);
}
```

SỔ TAY ARDUINO

Hàm thời gian **millis()**:

Giới thiệu về hàm thời gian **millis()**

millis() có nhiệm vụ trả về một số - là thời gian (tính theo mili giây) kể từ lúc mạch Arduino bắt đầu chương trình của bạn. Nó sẽ tràn số và quay số 0 (sau đó tiếp tục tăng) sau 50 ngày.

Trả về

Một số nguyên kiểu **unsigned long** là thời gian kể từ lúc thương trình Arduino được khởi động

Ví dụ về hàm thời gian **millis()**

```
unsigned long time;
void setup(){
  Serial.begin(9600);
}
void loop(){
  Serial.print("Time: ");
  time = millis();
  // in ra thời gian kể từ lúc chương trình được bắt đầu
  Serial.println(time);
  // đợi 1 giây trước khi tiếp tục in
  delay(1000);
}
```


SỔ TAY ARDUINO

Lưu ý quan trọng về hàm thời gian millis()

Các hàm về thời gian trong Arduino gồm millis() và micros() sẽ bị tràn số sau 1 thời gian sử dụng. Với hàm millis() là khoảng 50 ngày. Tuy nhiên, do là kiểu số nguyên không âm (unsigned long) nên ta dễ dàng khắc phục điều này bằng cách sử dụng hình thức ép kiểu.

```
unsigned long time;
byte ledPin = 10;
void setup()
{
  // khởi tạo giá trị biến time là giá trị hiện tại
  // của hàm millis();
  time = millis();
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, LOW);
}
void loop()
{
  // Lưu ý các dấu ngoặc khi ép kiểu
  // đoạn chương trình này có nghĩa là sau mỗi 1000 mili giây
  // đèn Led ở chân số 10 sẽ thay đổi trạng thái
  if ( (unsigned long) (millis() - time) > 1000)
  {
    // Thay đổi trạng thái đèn led
    if (digitalRead(ledPin) == LOW)
    {
      digitalWrite(ledPin, HIGH);
    } else {
      digitalWrite(ledPin, LOW);
    }
  }
}
```

SỔ TAY ARDUINO

```
}  
    // cập nhật lại biến time  
    time = millis();  
}  
}
```

Thông thường, nếu ta có 2 số A, B và B lớn hơn A ($B > A$) thì phép trừ thu được A-B là một số âm. Nhưng khi ép kiểu **unsigned long** là kiểu số nguyên dương, không có số âm nên giá trị trả về là 1 số nguyên dương lớn.

Ví dụ: kết quả của phép trừ:

```
unsigned long ex = (unsigned long) (0 - 1);
```

là 4294967295, con số này chính là giá trị lớn nhất của kiểu số unsigned long.

Xung PWM chương trình arduino:

Kiến thức cần nắm về xung:

Xung là các trạng thái cao / thấp về mức điện áp được lặp đi lặp lại. Đại lượng đặc trưng cho 1 xung PWM (Pulse Width Modulation) bao gồm **tần số** (frequency) và **chu kì xung** (duty cycle).

Vậy tần số tần số là gì???

Tần số là số lần lặp lại trong 1 đơn vị thời gian. Đơn vị tần số là Hz, tức là số lần lặp lại dao động trong 1 giây.

Lấy ví dụ, 1Hz = 1 dao động trong 1 giây. 1KHz = 1000 dao động trong 1 giây. 16MHz = 16 triệu dao động trong 1 giây.

Cách xác định 1 dao động như thế nào? Đa phần các bạn mới nghiên cứu điện tử thường mắc sai lầm ở việc xác định 1 dao động. Dao động được xác định từ trạng thái bắt đầu và kết thúc ngay trước khi trạng thái bắt đầu được lặp lại.

SỞ TAY ARDUINO



Khoảng cách giữa hai vạch đỏ là 1 dao động

Như vậy thông thường, 1 dao động sẽ bao gồm 2 trạng thái điện: mức cao (x giây) và mức thấp (y giây). Tỷ lệ phần trăm thời gian giữa 2 trạng thái điện này chính là chu kỳ xung.

Với $x/y = 0\%$ ta có xung chứa toàn bộ điện áp thấp.

Với $x/y = 50\%$ thì 50% thời gian đầu, xung có điện áp cao, 50% sau xung có điện áp thấp.

Với $x/y=100\%$ ta có xung chứa toàn bộ điện áp cao.

Tóm lại, với 1 xung ta có:

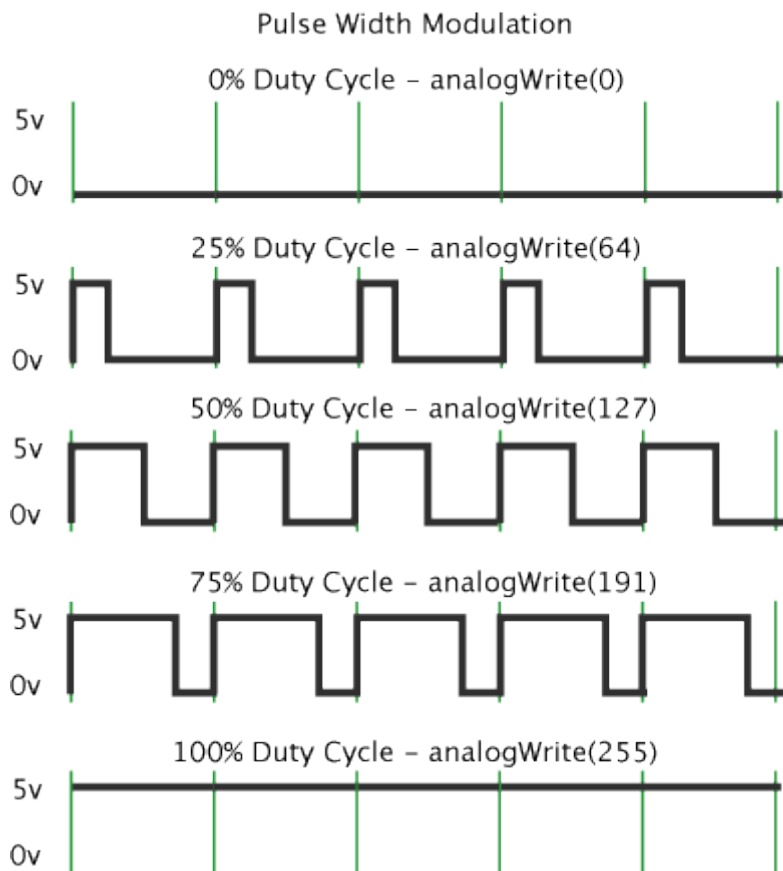
- + Tần số: để tính toán ra được thời gian của 1 xung
- + Chu kỳ xung: bao nhiêu thời gian xung có mức áp cao, bao nhiêu thời gian xung có mức áp thấp.

Và trong arduino khái niệm xung PWM ta hiểu như thế nào??

Hàm **analogWrite()** trong Arduino giúp việc tạo 1 xung dễ dàng hơn. Hàm này truyền vào tham số cho phép thay đổi chu kỳ xung. Tần số xung được Arduino thiết lập mặc định.

Đối với board Arduino Uno, xung trên các chân 3,9,10,11 có tần số là 490Hz, xung trên chân 5,6 có tần số 980Hz.

SỔ TAY ARDUINO



Xung được sử dụng với hàm `analogWrite` trong Arduino

Hàm nhập xuất `analogWrite()`:

Nhiệm vụ của hàm `analogWrite()`:

`analogWrite()` là lệnh xuất ra từ một chân trên mạch Arduino một mức tín hiệu analog (phát xung PWM). Người ta thường điều khiển mức sáng tối của đèn LED hay hướng quay của động cơ servo bằng cách phát xung PWM như thế này.

Bạn không cần gọi hàm `pinMode()` để đặt chế độ OUTPUT cho chân sẽ dùng để phát xung PWM trên mạch Arduino.

Cú pháp hàm `analogWrite()`:

SỔ TAY ARDUINO

```
analogWrite([chân phát xung PWM], [giá trị xung PWM]);
```

Giá trị mức xung PWM nằm trong khoảng từ 0 đến 255, tương ứng với mức duty cycle từ 0% đến 100%

Ví dụ hàm `analogWrite()`:

```
int led = 10;  
  
void setup() {  
}  
  
void loop() {  
    for (int i = 0; i <= 255; i++)  
        {  
            analogWrite(led,i);  
            delay(20);  
        }  
    }
```

Đoạn code trên có chức năng làm sáng dần một đèn LED được kết nối vào chân số 10 trên mạch Arduino.

SỔ TAY ARDUINO

Hàm nhập xuất **analogRead()**:

Nhiệm vụ của hàm `analogRead()`:

Là đọc giá trị điện áp từ một chân Analog . Trên mạch Arduino UNO có 6 chân Analog In, được kí hiệu từ A0 đến A5. Trên các mạch khác cũng có những chân tương tự như vậy với tên chữ "A" đứng đầu, sau đó là số hiệu của chân.

`analogRead()` luôn trả về 1 số nguyên nằm trong khoảng từ 0 đến 1023 tương ứng với thang điện áp (mặc định) từ 0 đến 5V. Bạn có thể điều chỉnh thang điện áp này bằng hàm **`analogReference()`**.

Hàm `analogRead()` cần 100 micro giây để thực hiện.

Cú pháp hàm `analogRead()`:

```
analogRead(chân đọc điện áp);
```

Ví dụ hàm `analogRead()`:

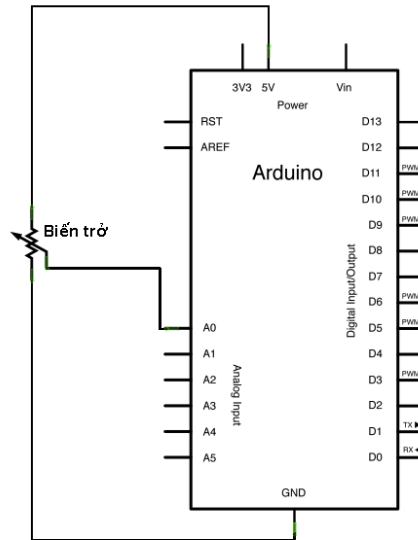
```
int voltage = analogRead(A0);
```

Trong đó A0 là chân dùng để đọc giá trị điện áp.

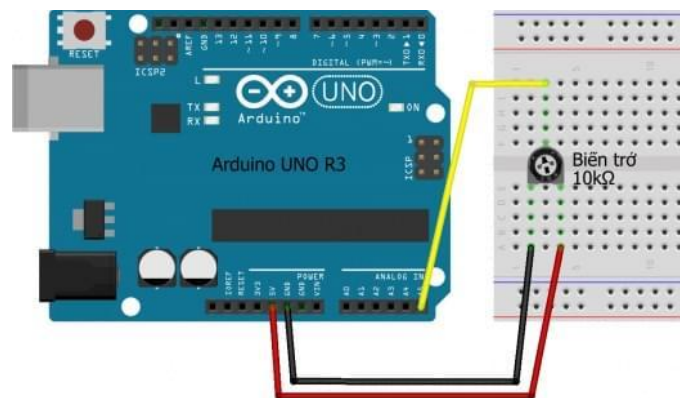
Nếu bạn chưa kết nối chân đọc điện áp, hàm `analogRead()` sẽ trả về một giá trị ngẫu nhiên trong khoảng từ 0 đến 1023. Để khắc phục điều này, bạn phải mắc thêm một điện trở có trị số lớn (khoảng 10k ohm trở lên) hoặc một tụ điện 104 từ chân đọc điện áp xuống GND.

SỔ TAY ARDUINO

Sơ đồ đấu nối dây của arduino để đo điện áp sử dụng hàm *analogRead*(chân đọc điện áp):



sơ đồ đấu nối dây của arduino để đo điện áp sử dụng hàm *analogRead*



sơ đồ đấu nối trực tiếp sử dụng điện trở 10k và mạch arduino

SỔ TAY ARDUINO

Hàm nhập xuất **analogReference()**:

Hàm **analogReference()** có nhiệm vụ:

Đặt lại mức (điện áp) tối đa khi đọc tín hiệu **analogRead**. Ứng dụng như sau, giả sử bạn đọc một tín hiệu dạng analog có hiệu điện thế từ 0-1,1V. Nhưng mà nếu dùng mức điện áp tối đa mặc định của hệ thống (5V) thì khoảng giá trị sẽ ngắn hơn dẫn đến độ chính xác kém hơn và vì vậy hàm này ra đời để giải quyết việc đó.

Cú pháp hàm **analogReference():**

analogReference(type)

type: một trong các kiểu giá trị sau: DEFAULT, INTERNAL, INTERNAL1V1, INTERNAL2V56, hoặc EXTERNAL

Trong đó:

+ DEFAULT : Đặt mức điện áp tối đa là 5V (nếu trên mạch dùng nguồn 5V làm nuôi chính) hoặc là 3,3V (nếu trên mạch dùng nguồn 3,3V làm nguồn nuôi chính).

+ INTERNAL : Đặt lại mức điện áp tối đa là 1,1 V (nếu sử dụng vi điều khiển ATmega328 hoặc ATmega168)

Đặt lại mức điện áp tối đa là 2,56V (nếu sử dụng vi điều khiển ATmega8).

+ INTERNAL1V1 : Đặt lại mức điện áp tối đa là 1,1 V (Chỉ có trên Arduino Mega).

+ INTERNAL2V56 : Đặt lại mức điện áp tối đa là 2,56 V (Chỉ có trên Arduino Mega).

+ EXTERNAL : Đặt lại mức điện áp tối đa BẰNG với mức điện áp được cấp vào chân AREF (Chỉ được cấp vào chân AREF một điện áp nằm trong khoảng 0-5V).

Chú ý:

Trường hợp, bạn sử dụng kiểu EXTERNAL cho hàm **analogReference** thì bạn phải cấp nó một nguồn nằm trong khoảng từ 0-5V, và nếu bạn đã cấp một

SỔ TAY ARDUINO

nguồn điện thỏa mãn điều kiện trên vào chân AREF thì bạn phải gọi dòng lệnh **analogReference(EXTERNAL)** trước khi sử dụng **analogRead()** (nếu không sẽ làm hư mạch của bạn).

Lệnh **digitalRead ()**:

Cú pháp lệnh **digitalRead()**:

```
digitalRead(pin)
```

Thông số lệnh **digitalRead()**:

pin (chân) : giá trị của digital muốn đọc

Trả về giá trị

HIGH hoặc LOW

Ví dụ về lệnh **digitalRead()**:

Ví dụ này sẽ làm cho đèn led tại pin 13 nhận giá trị như giá trị tại pin 2

```
int ledPin = 13; // chân led 13
int inPin = 2; // button tại chân 2
int val = 0; // biến "val" dùng để lưu tín hiệu từ digitalRead
void setup()
{
  pinMode(ledPin, OUTPUT); // đặt pin digital 13 là output
  pinMode(inPin, INPUT); // đặt pin digital 2 là input
}
void loop()
```

SỔ TAY ARDUINO

```
{  
  val = digitalRead(inPin); // đọc tín hiệu từ digital2
```

```
  digitalWrite(ledPin, val); // thay đổi giá trị của đèn LED là giá trị của digital 2  
}
```

Lệnh **digitalWrite()**:

Xuất tín hiệu ra các chân digital, có 2 giá trị là **HIGH** hoặc là **LOW**

Nếu một pin được thiết đặt là OUTPUT bởi **pinMode()**. Và bạn dùng **digitalWrite** để xuất tín hiệu thì điện thế tại chân này sẽ là 5V (hoặc là 3,3 V trên mạch 3,3 V) nếu được xuất tín hiệu là HIGH, và 0V nếu được xuất tín hiệu là LOW.

Nếu một pin được thiết đặt là INPUT bởi **pinMode()**. Lúc này **digitalWrite** sẽ bật (HIGH) hoặc tắt (LOW) hệ thống điện trở pullup nội bộ. Chúng tôi khuyên bạn nên dùng **INPUT_PULLUP** nếu muốn bật hệ thống điện trở pullup nội bộ.

Lệnh digitalWrite() có cú pháp:

```
digitalWrite(pin,value)
```

Thông số của lệnh digitalWrite():

pin: Số của chân digital mà bạn muốn thiết đặt

value: **HIGH** hoặc **LOW**

Ví dụ về lệnh digitalWrite()

```
int ledPin = 13; // đèn LED được kết nối với chân digital 13  
void setup()  
{
```

SỔ TAY ARDUINO

```
pinMode(ledPin, OUTPUT); // thiết đặt chân ledPin là OUTPUT
}
void loop()
{
digitalWrite(ledPin, HIGH); // bật đèn led
delay(1000); // dừng trong 1 giây
digitalWrite(ledPin, LOW); // tắt đèn led
delay(1000); // dừng trong 1 giây
}
```

Lệnh **pinMode()**:

Cấu hình một pin (chân) quy định hoạt động như là một đầu vào (**INPUT**) hoặc đầu ra (**OUTPUT**).

Như trong phiên bản Arduino IDE nó có thể kích hoạt các điện trở pullup nội bộ với chế độ **INPUT_PULLUP**. Ngoài ra, chế độ **INPUT** vô hiệu hóa một cách rõ ràng điện trở pullups nội bộ.

Câu lệnh có cú pháp: pinMode (pin, mode)

Thông số lệnh: pinMode()

pin : Số của chân digital mà bạn muốn thiết đặt

mode : có thể là **INPUT**, **OUTPUT** hoặc **INPUT_PULLUP**.

Ví dụ về câu lệnh pinMode():

```
int ledPin = 13; //điều khiển đèn LED chân số 13 của arduino
void setup()
{
```

SỔ TAY ARDUINO

```
pinMode(ledPin, OUTPUT); // thiết đặt chân ledPin là OUTPUT
}
void loop()
{
digitalWrite(ledPin, HIGH); // bật đèn led
delay(5000); // dừng trong 5 giây
digitalWrite(ledPin, LOW); // tắt đèn led
delay(5000); // dừng trong 5 giây
}
```

7. MỘT SỐ VÍ DỤ LẬP TRÌNH ARDUINO CƠ BẢN:

7.1. Điều khiển nhấp nháy led đơn theo yêu cầu:

Với những ai muốn học về arduino thì mình khuyên nên bắt đầu học từ điều khiển led đơn, rất đơn giản và dễ hiểu đó là nền tảng giúp ta sẽ nắm bắt được những chương trình khó hơn phức tạp hơn.

Nếu bạn đã đọc đến trang này thật sự bạn học arduino đã đi đúng hướng, mình tin chắc bạn sẽ thành công.

Chúng ta hãy cùng bắt tay vào thực hành.

Chuẩn bị linh kiện:

- 1 arduino uno R3/ mega... (mới học bạn nên bắt đầu từ Uno R3).
- 1 led đơn.
- 1 boartest.
- Dây cắm test.

Các bạn có thể tải code tại đây để hạn chế lỗi:

<http://tdhshop.com.vn/lap-trinh-dieu-khien-arduino-cho-nguoi-moi-bat-dau>

SỔ TAY ARDUINO

Lập trình cho arduino:

```
/*  
Blink - Nhấp nháy  
*/  
  
// chân digital 13 cần được kết nối với đèn LED  
  
// và chân digital 13 này sẽ được đặt tên là 'led'. Biến 'led' này có kiểu dữ liệu là int và có giá trị  
là 13  
  
int led = 13;  
  
// Hàm setup chạy một lần duy nhất khi khởi động chương trình  
  
void setup() {  
  
    // đặt 'led' là OUTPUT  
  
    pinMode(led, OUTPUT);  
  
    }  
  
// Hàm loop chạy mãi mãi sau khi kết thúc hàm setup()  
  
void loop() {  
  
    digitalWrite(led, HIGH); // bật đèn led sáng  
  
    delay(2000); // dừng chương trình trong 2 giây => thấy đèn sáng được 2 giây  
  
    digitalWrite(led, LOW); // tắt đèn led  
  
    delay(5000); // dừng chương trình trong 5 giây => thấy đèn tắt được 5 giây  
  
    }  
}
```

Trước tiên, cứ mỗi khi dùng một con LED, chúng ta phải **pinMode** OUTPUT chân Digital mà ta sử dụng cho con đèn LED. Trong ví

SỔ TAY ARDUINO

dụ, chúng ta sử dụng chân LED là chân digital 13. Nên đoạn code sau cần nằm trong void setup()

pinMode(13, OUTPUT);

Để bật một con đèn LED, bạn phải **digitalWrite HIGH** cho chân số 13 (chân Digital được kết nối với con LED). Đoạn code này nằm trong **void loop()**

digitalWrite(13,HIGH);

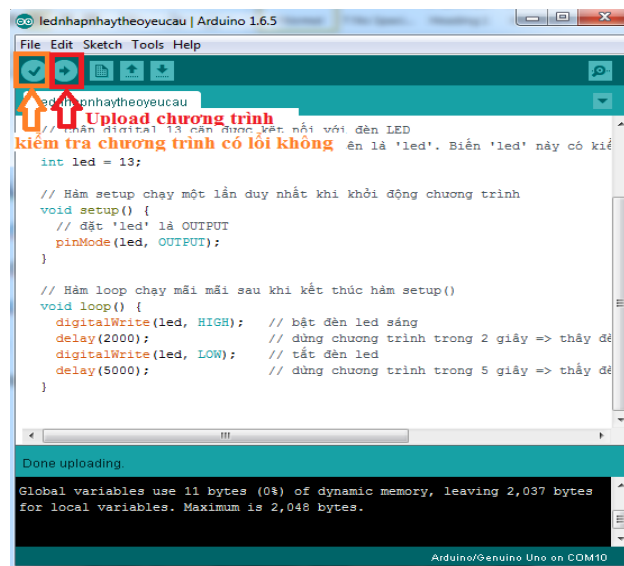
Dòng lệnh trên sẽ cấp một điện thế là 5V vào chân số Digital 13. Điện thế sẽ đi qua điện trở 220ohm rồi đến đèn LED (sẽ làm nó sáng mà không bị cháy, ngoài ra bạn có thể các loại điện trở khác $\leq 10k\Omega$). Để tắt một đèn LED, bạn sử dụng hàm:

digitalWrite(13,LOW);

Lúc này điện thế tại chân 13 sẽ là 0 V => đèn LED tắt. Và để thấy được trạng thái bật và tắt của đèn LED bạn phải dùng chương trình trong một khoảng thời gian đủ lâu để mắt cảm nhận được. Vì vậy, hàm delay được tạo ra để làm việc này (Dừng hẳn chương trình bao nhiêu mili giây(1000 mili giây = 1 giây).

Upload chương trình cho arduino:

Trước khi upload chương trình cho arduino các bạn cần lưu ý đã chọn đúng **board** chưa? Cổng **COM** đúng chưa? Và chọn **programmer** đúng chưa???đó là 3 điểm cần lưu ý. Các bạn có thể xem hướng dẫn nạp chương trình arduino tại: <http://tdhshop.com.vn/huong-dan-nap-chuong-trinh-cho-arduino-uno-r3-chuong-trinh-arduino>



```
lednhapnhaythoeyeucau | Arduino 1.6.5
File Edit Sketch Tools Help
lednhapnhaythoeyeucau
Upload chương trình
Kiểm tra chương trình có lỗi không
int led = 13;

// Hàm setup chạy một lần duy nhất khi khởi động chương trình
void setup() {
  // đặt 'led' là OUTPUT
  pinMode(led, OUTPUT);
}

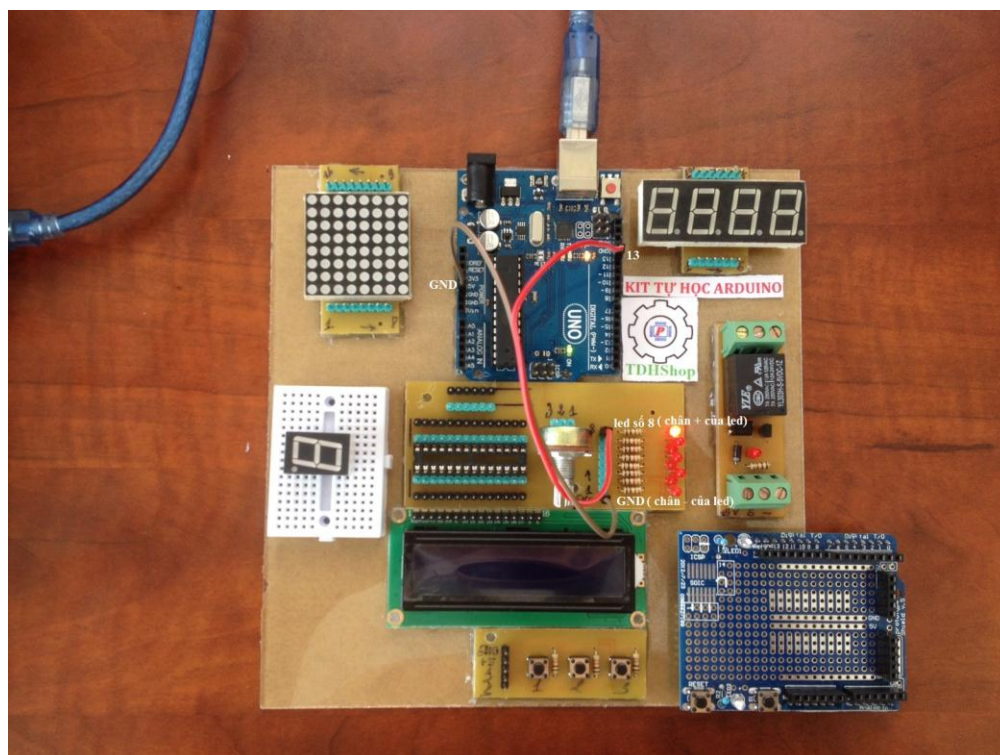
// Hàm loop chạy mãi mãi sau khi kết thúc hàm setup()
void loop() {
  digitalWrite(led, HIGH); // bật đèn led sáng
  delay(2000); // dừng chương trình trong 2 giây => thấy đèn sáng
  digitalWrite(led, LOW); // tắt đèn led
  delay(5000); // dừng chương trình trong 5 giây => thấy đèn tắt
}

Done uploading.
Global variables use 11 bytes (0%) of dynamic memory, leaving 2,037 bytes
for local variables. Maximum is 2,048 bytes.
Arduino/Genuino Uno on COM10
```

SỔ TAY ARDUINO

Sơ đồ nối dây:

Trường hợp các bạn có kit tự học arduino này thì quá tiện giúp học nhanh hơn và dễ hiểu hơn, nhưng không sao mình sẽ hướng dẫn chi tiết.



Trường hợp các bạn không có kit tự học thì các bạn nối dây theo hình dưới:

Arduino uno

D13

GND

led

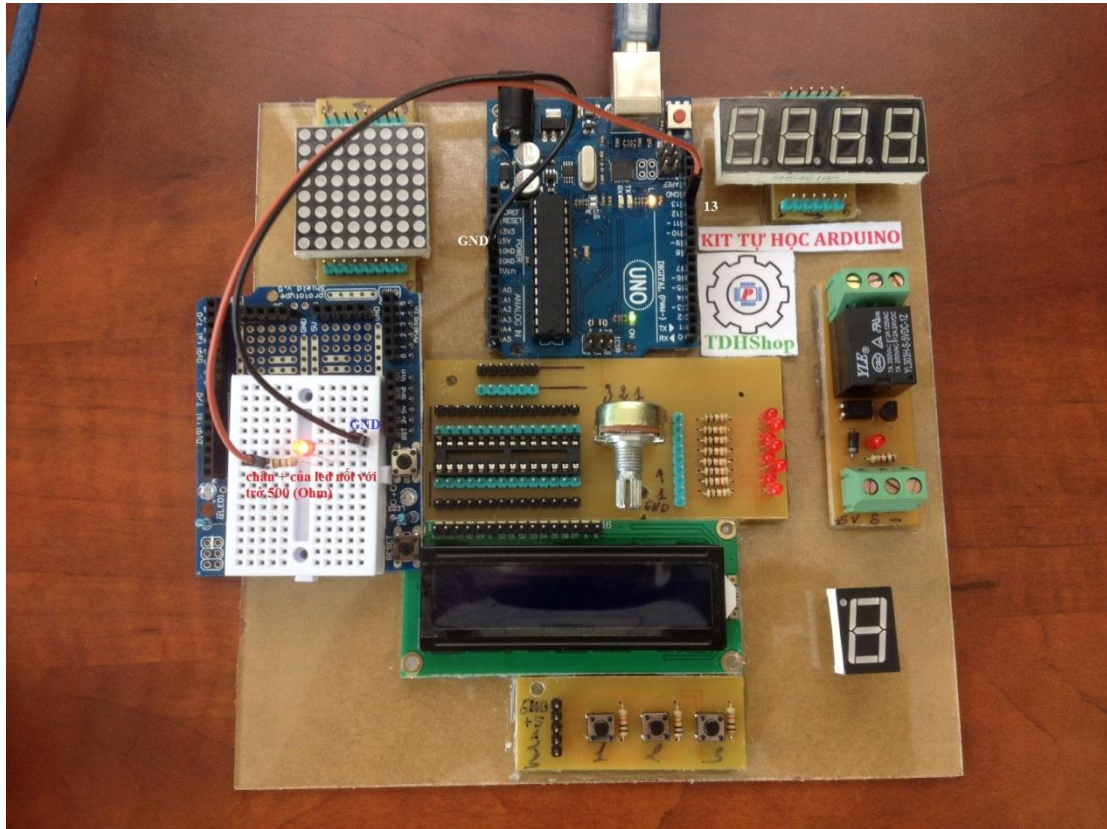
+

-

**chân + của led các bạn nhớ nối thêm
trở 500 Ohm để bảo vệ Led nhé**

SỔ TAY ARDUINO

Nối dây thông qua breadboard mini:



Sau khi nạp code xong các bạn sẽ thấy led sáng 2 giây và tắt 5 giây , vấn đề sáng tắt của led các bạn có thể **thay đổi thời gian** được.

Chúc các bạn thành công!!!

7.2. Điều khiển 8 led đơn theo yêu cầu:

Bây giờ chúng ta sẽ điều khiển 8 led đơn với hiệu ứng chạy led của rất đơn giản.

Chuẩn bị linh kiện:

- 1 arduino uno R3/ mega... (mới học bạn nên bắt đầu từ Uno R3).
- 8 led đơn.
- 8 điện trở 500 Ohm
- 1 boartest.
- Dây cắm test.

Web: tdhshop.com.vn – Chuyên Kit Tự Học Arduino từ cơ bản đến nâng cao

SỔ TAY ARDUINO

Các bạn có thể tải code tại đây để hạn chế lỗi:

<http://tdhshop.com.vn/lap-trinh-dieu-khien-arduino-cho-nguoi-moi-bat-dau>

Lập trình cho arduino:

```
int del=100; // 100 milliseconds

void setup()
{
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
}

void loop()
{
  digitalWrite(2, HIGH); // turn on LED on pin 2
  delay(del);           // dừng chương trình 100 mili giây
  digitalWrite(2, LOW); // turn it off
  digitalWrite(3, HIGH); // turn on LED on pin 3
  delay(del);           // dừng chương trình 100 mili giây
  digitalWrite(3, LOW); // turn it off
```

Web: tdhshop.com.vn – Chuyên Kit Tự Học Arduino từ cơ bản đến nâng cao

SỔ TAY ARDUINO

```
digitalWrite(4, HIGH); // turn on LED on pin 4  
delay(del); // dừng chương trình 100 mili giây  
digitalWrite(4, LOW); // turn it off  
digitalWrite(5, HIGH); // turn on LED on pin 5  
delay(del); // dừng chương trình 100 mili giây  
digitalWrite(5, LOW); // turn it off  
digitalWrite(6, HIGH); // turn on LED on pin 6  
delay(del); // dừng chương trình 100 mili giây  
digitalWrite(6, LOW); // turn it off  
digitalWrite(7, HIGH); // turn on LED on pin 7  
delay(del); // dừng chương trình 100 mili giây  
digitalWrite(7, LOW); // turn it off  
digitalWrite(8, HIGH); // turn on LED on pin 8  
delay(del); // dừng chương trình 100 mili giây  
digitalWrite(8, LOW); // turn it off  
digitalWrite(9, HIGH); // turn on LED on pin 9  
delay(del); // dừng chương trình 100 mili giây  
digitalWrite(9, LOW); // turn it off  
digitalWrite(8, HIGH); // turn on LED on pin 8  
delay(del); // dừng chương trình 100 mili giây  
digitalWrite(8, LOW); // turn it off  
digitalWrite(7, HIGH); // turn on LED on pin 7  
delay(del); // wait
```

SỔ TAY ARDUINO

```
digitalWrite(7, LOW); // turn it off
digitalWrite(6, HIGH); // turn on LED on pin 6
delay(del); // wait
digitalWrite(6, LOW); // turn it off
digitalWrite(5, HIGH); // turn on LED on pin 5
delay(del); // dừng chương trình 100 mili giây
digitalWrite(5, LOW); // turn it off
digitalWrite(4, HIGH); // turn on LED on pin 4
delay(del); // dừng chương trình 100 mili giây
digitalWrite(4, LOW); // turn it off
digitalWrite(3, HIGH); // turn on LED on pin 3
delay(del); // dừng chương trình 100 mili giây
digitalWrite(3, LOW); // turn it off
}
```

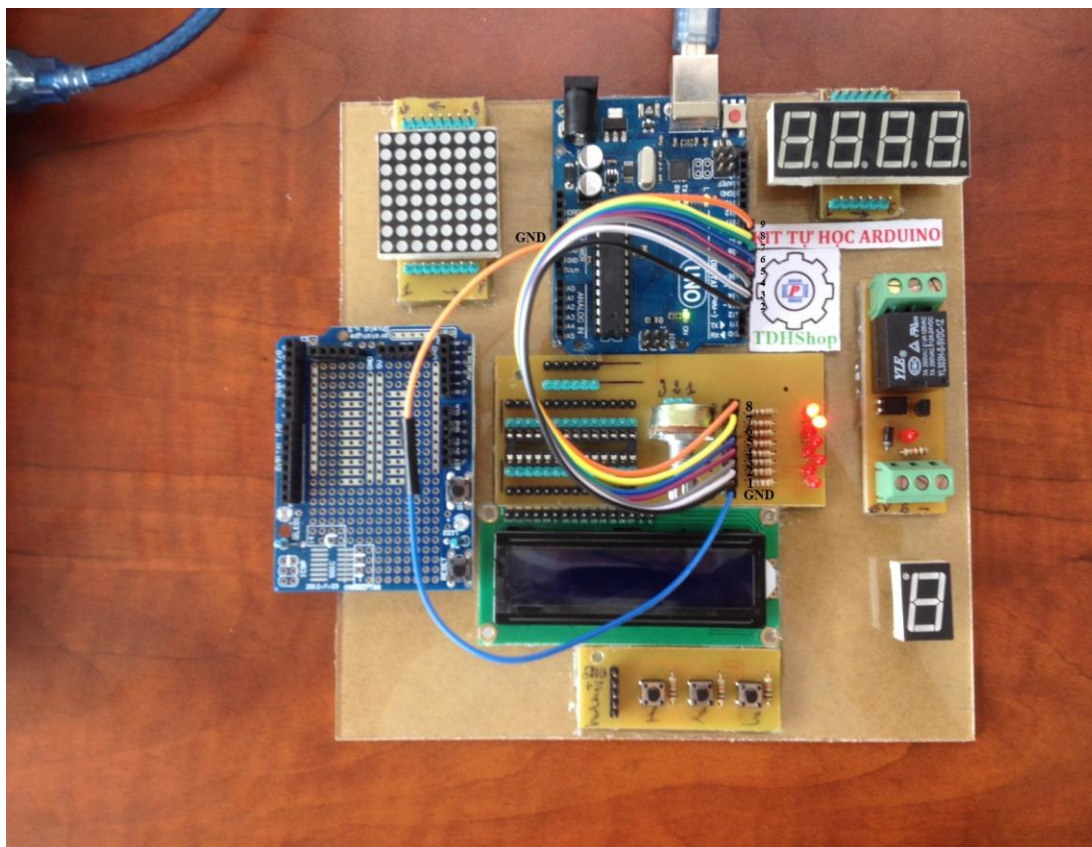
Các bạn cảm thấy chương trình quá dài nhưng không sao mình mới học thì sử dụng lệnh cơ bản nhất, ở phần dưới mình sẽ đưa chương trình ngắn gọn hơn nhất nhiều.

Sơ đồ kết nối:

Arduino uno	LED
D2	led1
D3	led2
D4	led3
D5	led4
D6	led5
D7	led6
D8	led7
D9	led8
GND	chân (-)

các bạn nhớ mắc thêm trở 500 Ohm để bảo vệ led nha

SỔ TAY ARDUINO



Sau khi đã kết nối xong các bạn Upload code cho arduino nhé và hiệu ứng sáng tắt liên tục từ led số 1 đến led số 8. Và thời gian sáng tắt của led chúng ta có thể thay đổi được.

Và đây là code gọn hơn và đỡ tốn bộ nhớ của chip điều khiển:

```
byte ledPin[] = {2,3,4,5,6,7,8,9}; // Mảng lưu vị trí các chân Digital mà các đèn LED sử dụng theo thứ tự từ 1->8. Bạn có thể thêm các LED bằng cách thêm các chân digital vào mảng này
```

```
byte pinCount; // Khai báo biến pinCount dùng cho việc lưu tổng số chân LED
```

```
void setup() {
```

```
    pinCount = sizeof(ledPin);
```

```
    for (int i=0;i<pinCount;i++) {
```

```
        pinMode(ledPin[i],OUTPUT); //Các chân LED là OUTPUT
```

```
        digitalWrite(ledPin[i],LOW); //Mặc định các đèn LED sẽ tắt
```

Web: tdhshop.com.vn – Chuyên Kit Tự Học Arduino từ cơ bản đến nâng cao

SỔ TAY ARDUINO

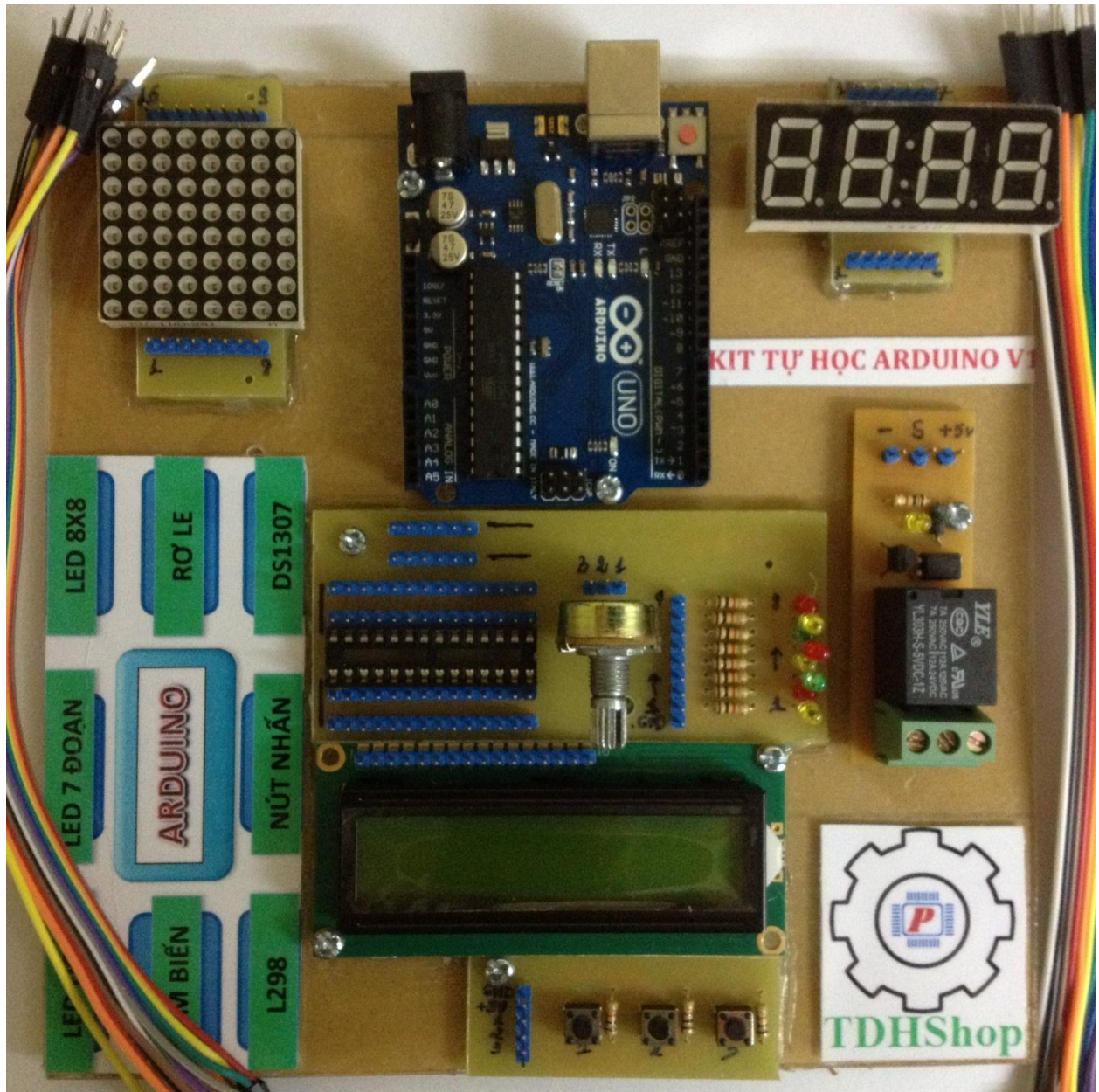
```
}  
}  
void loop() {  
  /*  
   Bật tuần tự các đèn LED  
  */  
  for (int i=0; i < pinCount; i++) {  
    digitalWrite(ledPin[i],HIGH); //Bật đèn  
    delay(500); // Dừng để các đèn LED sáng dần  
  }  
  /*  
   Tắt tuần tự các đèn LED  
  */  
  for (int i = 0; i < pinCount; i += 1) {  
    digitalWrite(ledPin[i],LOW); // Tắt đèn  
    delay(500); // Dừng để các đèn LED tắt dần  
  }  
}
```

Quá gọn phải không các bạn chức năng giống như chương trình ở trên. Bạn cứ thử upload chương trình và xem kết quả ha.

Chúc các bạn thành công!!!

Việc học arduino quá thú vị phải không các bạn mình sẽ upload **tài liệu học lập trình cho người mới bắt đầu V2 lên sau ha.**

SỔ TAY ARDUINO



Điều khiển led chân số 13 trên board arduino uno r3