INTERNET OF THINGS

1. Tìm hiểu về Arduino.
2. Arduino là gì?

Arduino là một nền tảng mà mọi thiết bị phần cứng đều được chuẩn hóa và làm sẵn, người sử dụng chỉ cần chọn các linh kiện điện tử mình cần để ráp lại là có thể chạy được.

Một Arduino bao gồm:

Board mạch: Vi điều khiển, nguồn cung cấp, cổng USB để kết nối PC, I/O,...

Môi trường tích hợp IDE: Với nhiều ví dụ có sẵn và sự hỗ trợ lớn từ cộng đồng.

Nguồn mở: Hardware và software. Cung cấp đầy đủ sơ đồ nguyên lý, thiết kế, mã nguồn...có thể sửa đổi cho phù hợp với nhu cầu.

Hệ thống module tiêu chuẩn: Mỗi module có một chức năng ( Việc thiết kế chính là kết nối các khối chức năng đã chuẩn hóa )

1. Arduino có thể kết nối với những gì?

Một hệ thống Arduino có thể cung cấp cho bạn rất nhiều sự tương tác với môi trường xung quanh.

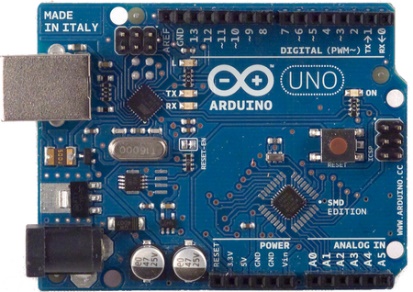
* Hệ thống cảm biến da dạng về chuẩn loại ( cảm biến nhiệt độ, độ ẩm, gia tốc, cường độ ánh sáng, màu sắc vật thể, lưu lượng nước, phát hiện chuyển động, phát hiện kim loại, khí độc,...)
* Các thiết bị hiển thị ( màn hình LCD, LED,... ).
* Các module chức năng ( shield ) hỗ trợ kết nối có dây với các thiết bị khác hoặc kết nối không dây thông dụng ( 3G, wifi, GPRS, Bluetooth, 2.4GHz,...).
* Định vị GPS, nhắn tin SMS,...
* ...và rất nhiều công dụng khác nữa.

1. Các loại Arduino.

Vì Arduino là một hệ thống bao gồm các module được kết nối với nhau nên Arduino rất đa dạng về loại. Trong lịch sử phát triển của Arduino đã có rất nhiều mẫu Arduino được ra đời. Một số Arduino phổ biến được kể đến như Arduino MEGA 2560, Arduino NANO, Arduino UNO,.... Trong bài viết này tôi sử dụng Arduino Uno R3. Một Arduino phổ biến và được sử dụng rộng rãi để giới thiệu.

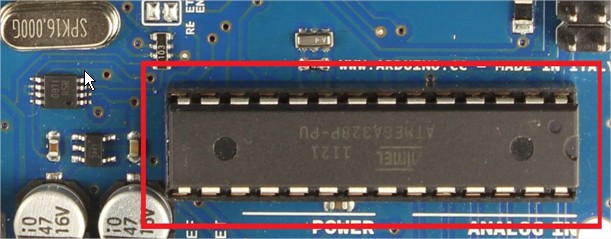
1. Arduino UNO R3.

Nhắc tới dòng Arduino để lập trình, cái người ta nhắc tới đầu tiên chính là Arduino Uno. Hiện dòng mạch này đã phát triển tới thế hệ thứ 3 (R3).





1. Vi điều khiển.



Arduino UNO có thể sử dụng 3 vi điều khiển họ 8bit AVR là ATmega8, ATmega168, ATmega328. Bộ não này có thể xử lý các tác vụ đơn giản như điều khiển LED nhấp nháy, xử lý tín hiệu cho xe điều khiển từ xa, làm trạm đo nhiệt độ - độ ẩm,...

1. Nguồn cấp.

Arduino UNO có thể cấp nguồn 5V thông qua cổng USB kết nối với máy tính hoặc cấp nguồn ngoài với điện áp khuyên dùng là 7-12V DC và giới hạn là 6-20V. Thường chúng ta sẽ cấp nguồn 9V cho Arduino nếu không có sẵn nguồn từ USB. Nếu cấp nguồn vượt ngưỡng giới hạn trên, bạn sẽ làm hỏng Arduino.

1. Các chân năng lượng.

* GND (Ground): Cực âm của nguồn điện cấp cho Arduino UNO. Khi bạn dùng các thiết bị sử dụng những nguồn điện riêng biệt thì những chân này phải được kết nối với nhau.
* 5V: cấp điện áp 5V đầu ra. Dòng tối đa cho phép ở chân này là 500mA.
* 3.3V: cấp điện áp 3.3V đầu ra. Dòng tối đa cho phép ở chân này là 50mA.
* Vin (Voltage input): để cấp nguồn ngoài cho Arduino UNO, bạn nối cực dương của nguồn với chân này và cực âm của nguồn với chân GND.
* IOREF: điện áp hoạt động của vi điều khiển trên Arduino UNO có thể đo được ở chân này. Và dĩ nhiên nó luôn là 5V. Mặc dù vậy bạn không được lấy nguồn 5V từ chân này để sử dụng vì chức năng của nó không phải là để cấp nguồn.
* RESET: việc nhấn nút reset trên board để reset vi điều khiển tương đương với việc chân RESET được nối với GND qua 1 điện trở 10k ôm.

Lưu ý:

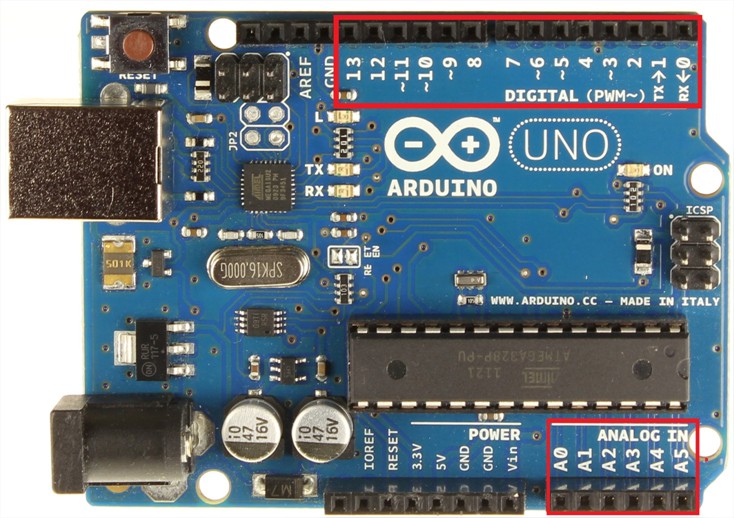
* Arduino UNO không có bảo vệ cắm ngược nguồn vào. Do đó bạn phải hết sức cẩn thận, kiểm tra các cực âm – dương của nguồn trước khi cấp cho Arduino UNO. Việc làm chập mạch nguồn vào của Arduino UNO sẽ biến nó thành một miếng nhựa chặn giấy. mình khuyên bạn nên dùng nguồn từ cổng USB nếu có thể.
* Các chân 3.3V và 5V trên Arduino là các chân dùng để cấp nguồn ra cho các thiết bị khác, không phải là các chân cấp nguồn vào. Việc cấp nguồn sai vị trí có thể làm hỏng board. Điều này không được nhà sản xuất khuyến khích.
* Cấp nguồn ngoài không qua cổng USB cho Arduino UNO với điện áp dưới 6V có thể làm hỏng board.
* Cấp điện áp trên 13V vào chân RESET trên board có thể làm hỏng vi điều khiển ATmega328.
* Cường độ dòng điện vào/ra ở tất cả các chân Digital và Analog của Arduino UNO nếu vượt quá 200mA sẽ làm hỏng vi điều khiển.
* Cấp điệp áp trên 5.5V vào các chân Digital hoặc Analog của Arduino UNO sẽ làm hỏng vi điều khiển.
* Cường độ dòng điện qua một chân Digital hoặc Analog bất kì của Arduino UNO vượt quá 40mA sẽ làm hỏng vi điều khiển. Do đó nếu không dùng để truyền nhận dữ liệu, bạn phải mắc một điện trở hạn dòng.

1. Bộ nhớ.

Vi điều khiển ATmega328 tiêu chuẩn cung cấp cho người dùng:

* 32KB bộ nhớ flash: những đoạn lệnh bạn lập trình sẽ được lưu trữ trong bộ nhớ flash của vi điều khiển. Thường thì sẽ có khoảng vài KB trong số này sẽ được dùng cho bootloader.
* 2KB cho SRAM (Static Random Access Memory): giá trị các biến bạn khai báo khi lập trình sẽ lưu tại đây. Bạn khai báo càng nhiều biến thì cần càng nhiều bộ nhớ RAM. Khi mất điện, bộ nhớ SRAM sẽ mất hết dữ liệu.
* 1KB cho EEPROM (Electrically Eraseble Programmable Read Only Memory): đây giống như một chiếc ổ cứng mini – nơi bạn có thể đọc và ghi dữ liệu của mình vào đây mà không phải lo bị mất khi mất điện giống như trên SRAM.

1. Các cổng vào/ra.



Arduino UNO có 14 chân digital dùng để đọc hoặc xuất tín hiệu. Chúng chỉ có 2 mức điện áp là 0V và 5V với dòng vào ra tối đa trên mỗi chân là 40mA. Ở mỗi chân đều có các điện trở pull-up từ được cài đặt ngay trong vi điều khiển ATmega328 (mặc định thì các điện trở này không được kết nối).

Một số chân digital có chức năng đặc biệt như sau:

* 2 chân Serial: 0 (RX) và 1(TX): dùng để gửi (transmit – TX) và nhận (receive – RX) dữ liệu TTL Serial. Arduino UNO có thể giao tiếp với các thiết bị khác thông qua 2 chân này. Kết nối bluetooth thường thấy nói nôm na chính là kết nối Serial không dây. Nếu không cần giao tiếp Serial, bạn không nên sử dụng 2 chân này nếu không cần thiết.
* Chân PWM (~) 3, 5, 6, 9, 10 và 11: Cho phép bạn xuất ra xung PWM với độ phân giải 8bit (giá trị từ 0 -> 28-1 tương đương với từ 0V -> 5V) bằng hàm analogWrite(). Nói một cách đơn giản, bạn có thể điều chỉnh được điện áp ra ở chân này từ mức 0V đến 5V thay vì cố định như các chân khác.
* Chân giao tiếp SPI: 10 (SS), 11(MOSI), 12(MISO), 13(SCK). Ngoài các chức năng thông thường, 4 chân này còn dùng để truyền phát dữ liệu bằng giao thức SPI với các thiết bị khác.
* LED 13: trên Arduino UNO có 1 đèn LED màu cam (kí hiệu L). Khi bấm nút Reset, bạn sẽ thấy đèn này nhấp nháy đề báo hiệu. Nó được nối với chân số 13. Khi chân này được sử dụng, LED sẽ sáng.
* Arduino có 6 chân analog (A0 -> A5) cung cấp độ phân giải tín hiệu 10bit (0->210-1) để đọc giá trị điện áp trong khoảng 0V -> 5V. Với chân AREF trên board, bạn có thể đưa vào điện áp tham chiếu khi sử dụng các chân analog. Tức là nếu bạn cấp điện áp 2.5V vào chân này thì bạn có thể dùng các chân analog để đo điện áp trong khoảng từ 0V -> 2.5V với độ phân giải vẫn là 10bit.
* Đặc biệt, Arduino UNO có 2 chân A4 (SDA) và A5(SCL) hỗ trợ giao tiếp I2C/TWI với các thiết bị khác.

1. Lập trình cho Arduino.

Các thiết bị dựa trên nền tảng Arduino được lập trình bằng ngôn riêng. Ngôn ngữ này dựa trên ngôn ngữ Wiring được viết cho phần cứng nói chung. Và Wiring lại là một biến thể của C/C++. Một số người gọi nó là Wiring, một số khác thì gọi là C hay C/C++. Riêng mình thì gọi nó là “*ngôn ngữ Arduino*”, và đội ngũ phát triển Arduino cũng gọi như vậy. Ngôn ngữ Arduino bắt nguồn từ C/C++ phổ biến hiện nay do đó rất dễ học, dễ hiểu. Nếu học tốt chương trình Tin học 11 thì việc lập trình Arduino sẽ rất dễ thở đối với bạn.

Để lập trình cũng như gửi lệnh và nhận tín hiệu từ mạch Arduino, nhóm phát triển dự án này đã cũng cấp đến cho người dùng một môi trường lập trình Arduino được gọi là Arduino IDE (**I**ntergrated **D**evelopment **E**nvironment)  như hình dưới đây.

* Hướng dẫn cài đặt môi trường làm việc với Arduino.

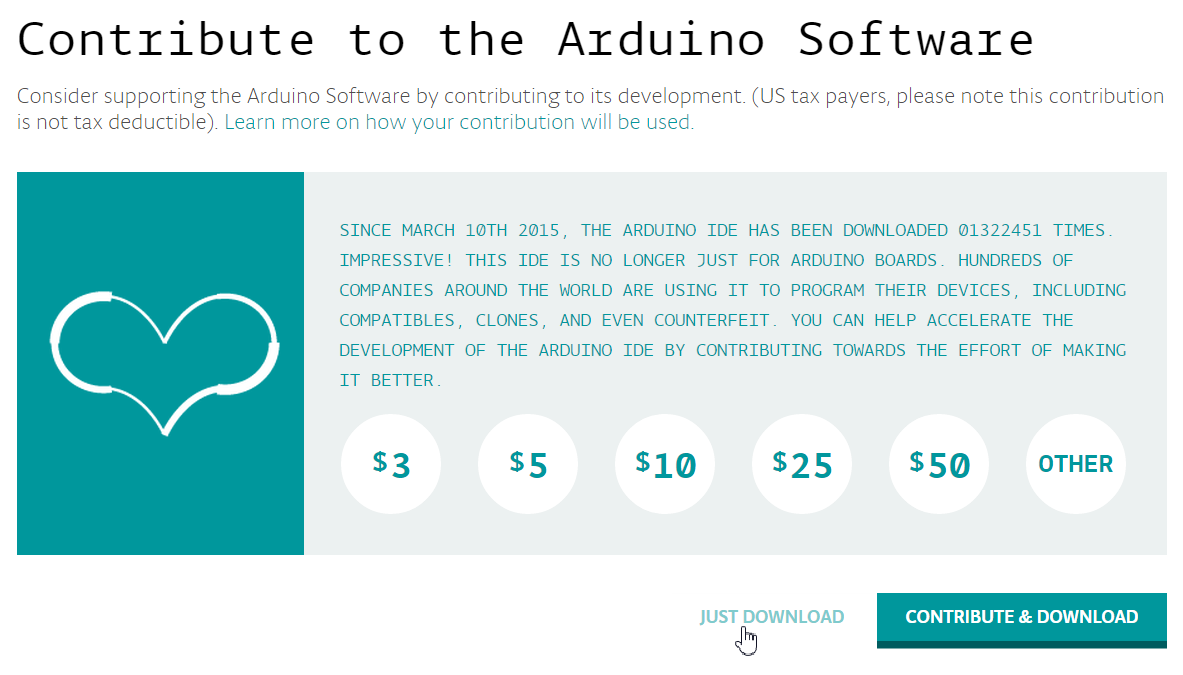
Vì Arduino được viết trên Java nên chúng ta cần phải cài JSE trước Arduino IDE. Link download:

<http://www.oracle.com/technetwork/java/javase/downloads/jre7-downloads-1880261.html>

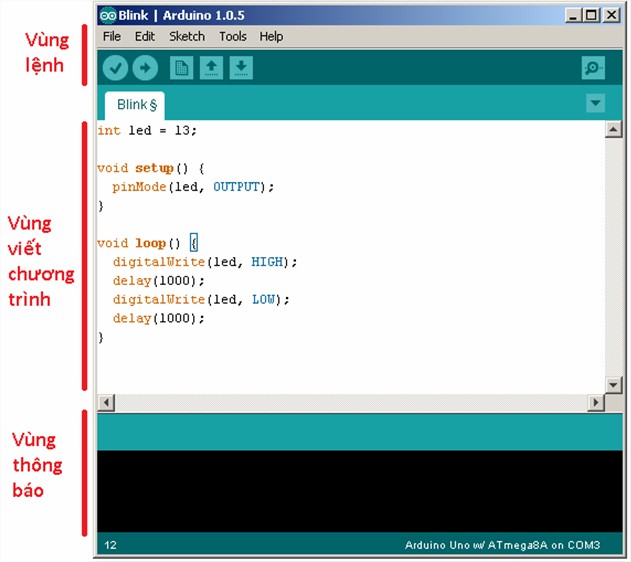
* Cài đặt Arduino IDE.
  + Truy cập <http://arduino.cc/en/Main/Software/> chọn phiên bản phù hợp với hệ điều hành trên máy tính bạn đang sử dụng để tải



* + Bấm JUST DOWNLOAD để bắt đầu tải (hoặc có thể quyên góp tiền cho nhà phát triển bằng cách bấm vào CONTRIBUTE & DOWNLOAD và làm theo hướng dẫn)

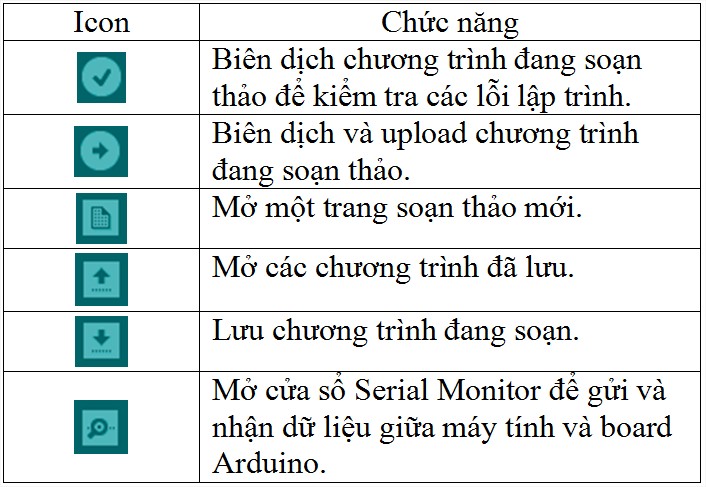


* + Sau khi tải xong bạn mở file vừa tải và cài đặt bình thường. (Khi cài đặt Arduino IDE sẽ hỏi bạn có cài Sau khi cài đặt xong giao diện làm việc của Arduino IDE sẽ như sau:



* + Vùng lệnh:

Bao gồm các nút lệnh menu (File, Edit, Sketch, Tools, Help). Phía dưới là các icons cho phép sử dụng nhanh các chức năng thường dùng của IDE:



* + Vùng viết chương trình:

Vùng để bạn viết các đoạn mã chương trình của mình tại đây.

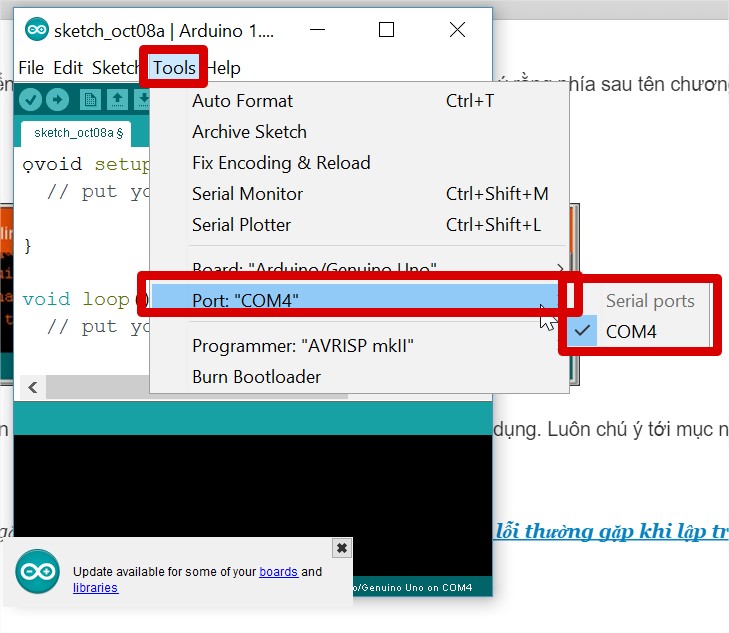
* + - Void setup (): Vùng để viết các đoạn mã chạy 1 lần
    - Loop setup(): Vùng để viết các đoạn chính của bạn, các đoạn mã này sẽ lặp đi lặp lại liên tục.
  + Vùng thông báo (debug):

Những thông báo của IDE sẽ được hiển thị tại đây



* + Lưu ý:

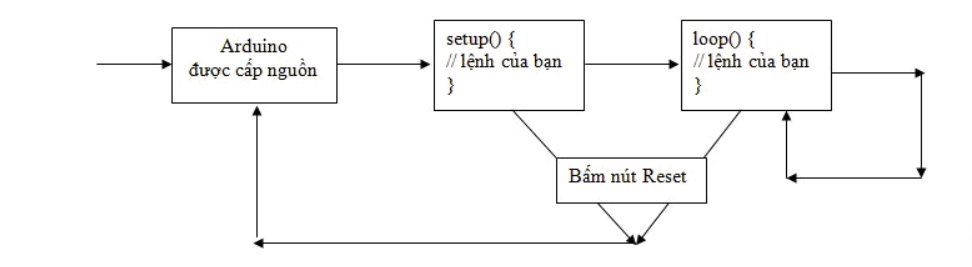
Khi kết nói Arduino với máy tính để lập trình, các bạn cần chọn cổng port (cổng kết nối khi gắn board vào) và board (tên board mà bạn sử dụng). Giả sử bạn đang dùng mạch Arduino UNO và khi gắn board này vào máy tính bằng cáp USB, nó được nhận là COM 4 thì bạn chỉnh như sau để có thể lập trình được:



1. Các hàm và thư viện trong Arduino
   1. Setup() và loop().

Những lệnh trong setup() sẽ được chạy 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() được chạy. Chúng sẽ lặp đi lặp lại liên tục cho tới khi nào bạn ngắt nguồn của board Arduino mới thôi.

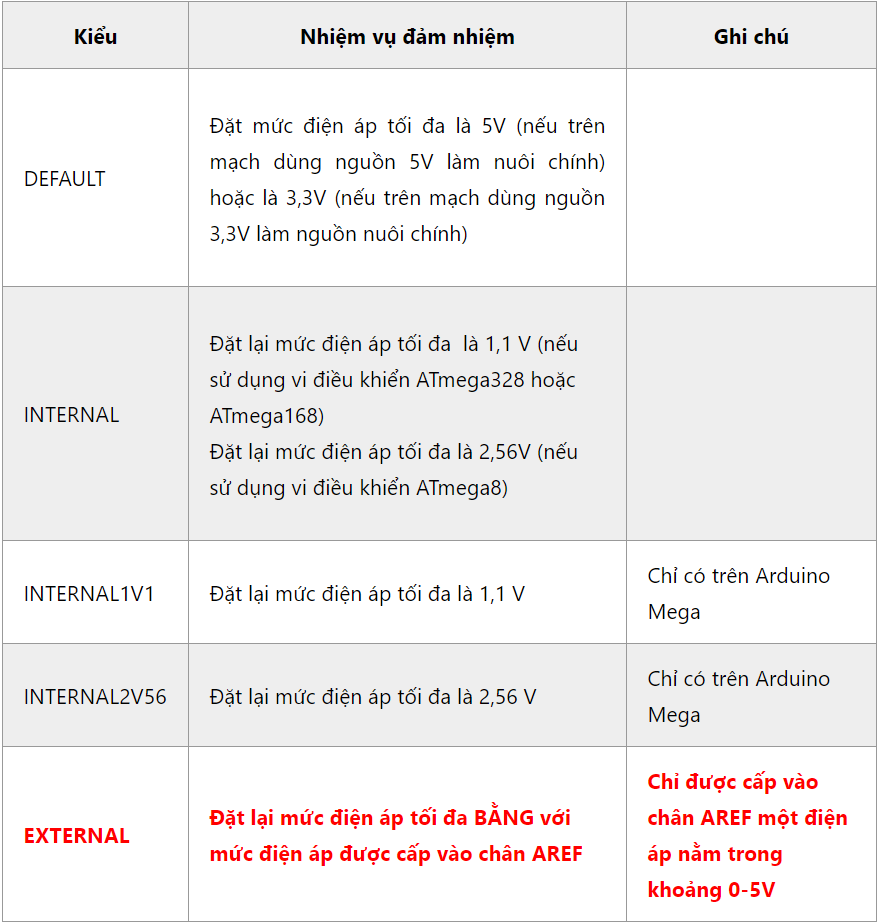
Bất cứ khi nào bạn nhất nút Reset, chương trình của bạn sẽ trở về lại trạng thái như khi Arduino mới được cấp nguồn



* 1. Các hằng số.
  + HIGH: Trong lập trình trên Arduino, ***HIGH***là một hằng số có giá trị nguyên là 1. Trong điện tử, ***HIGH***là một mức điện áp lớn hơn 0V. G*iá trị của****HIGH****được định nghĩa khác nhau trong các mạch điện khác nhau, nhưng thường được quy ước ở các mức như 1.8V, 2.7V, 3.3V 5V, 12V, ...*
  + Dù HIGH không có một trị số nào rõ ràng nhưng nhất quyết rằng giá trị của nó luôn lớn hơn 0V
  + LOW: ***LOW***là một hằng số có giá trị nguyên là 0. Trong điện tử, ***LOW***là mức điện áp 0V hoặc gần bằng 0V, giá trị này được định nghĩa khác nhau trong các mạch điện khác nhau, nhưng thường là 0V hoặc hơn một chút xíu.
  + INPUT: Các pin của Arduino ( Atmega ) được cấu hình là một INPUT với pinMode ( ) có nghĩa là làm cho pin ấy có trở kháng cao (không cho dòng điện đi ra). Nói dễ hiểu thì khi một pin được cấu hình là INPUT thì bạn sẽ dễ dàng đọc được các tín hiệu điện và đọc được từ bất cứ thứ gì (Có điện <= 5V)
  + INPUT\_PULLUP: Chip Atmega trên Arduino có nội kéo lên điện trở (điện trở kết nối với hệ thống điện nội bộ) mà bạn có thể truy cập. Nếu bạn không thích mắc thêm một điện trở ở mạch ngoài, bạn có thể dùng tham số INPUT\_PULLUP trong pinMode()
  + OUTPUT: Để thiết đặt pin là một OUTPUT, chúng ta dùng pinMode ( )[,](http://arduino.vn/reference/thiet-dat-digital-pins-nhu-la-input-inputpullup-va-output) điều này có nghĩa là làm cho pin ấy có một trở kháng thấp (cho dòng điện đi ra). Điều này có nghĩa, pin sẽ cung cấp một lượng điện đáng kể cho các mạch khác
  + LED\_BUILTIN: Hầu hết các mạch Arduino đều có một pin kết nối với một on-board LED (led nằm trên mạch) nối tiếp với một điện trở. LED\_BUILTIN là một hằng số thay thế cho việc tuyên bố một biến có giá trị điều khiển on-board LED. Hầu hết trên các mạch Arduino, chúng có giá trị là 13
  1. Nhập xuất Digital (Digital I/O).
* pinMode(): Cấu hình 1 pin quy định hoạt động như là một đầu vào (INPUT) hoặc đầu ra (OUTPUT)
  + Cú pháp : pinMode(pin , mode);
    - pin: Số thứ tự của chân digital mà bạn muốn thiết đặt.
    - mode: [INPUT](http://arduino.vn/reference/thiet-dat-digital-pins-nhu-la-input-inputpullup-va-output), [INPUT\_PULLUP](http://arduino.vn/reference/thiet-dat-digital-pins-nhu-la-input-inputpullup-va-output) hoặc [OUTPUT](http://arduino.vn/reference/thiet-dat-digital-pins-nhu-la-input-inputpullup-va-output)
* digitalWrite(): Xuất tín hiệu ra các chân digital, có 2 giá trị là [HIGH](http://arduino.vn/reference/high) hoặc là [LOW](http://arduino.vn/reference/low). Nếu một pin được thiết đặt là OUTPUT bởi [pinMode()](http://arduino.vn/reference/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](http://arduino.vn/reference/high), và 0V nếu được xuất tín hiệu là [LOW](http://arduino.vn/reference/low)[.](http://arduino.vn/reference/digitalwrite)
  + Cú pháp : digitalWrite(pin, value);
    - pin : Số thứ tự của chân digital mà bạn muốn thiết đặt.
    - value: [HIGH](http://arduino.vn/reference/high) hoặc [LOW](http://arduino.vn/reference/low)
* digitalRead(): Đọc tín hiệu điện từ một chân digital (được thiết đặt là [INPUT](http://arduino.vn/reference/thiet-dat-digital-pins-nhu-la-input-inputpullup-va-output)). Trả về 2 giá trị [HIGH](http://arduino.vn/reference/high) hoặc [LOW](http://arduino.vn/reference/low).
  + Cú pháp : digitalRead(pin);
    - pin: Số thứ tự của chân digital mà bạn muốn đọc.
  1. Nhập xuất Analog (Analog I/O).
* 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

=> độ chính xác kém hơn.

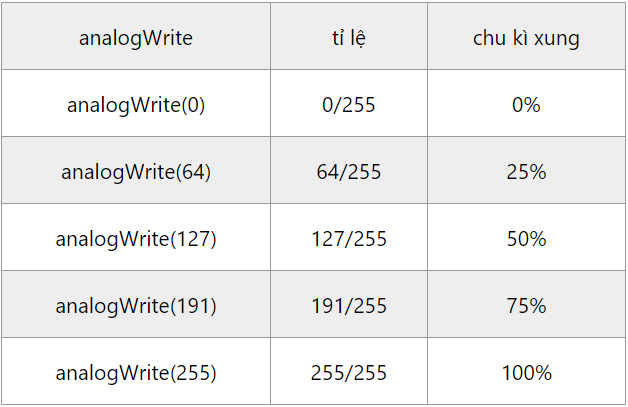
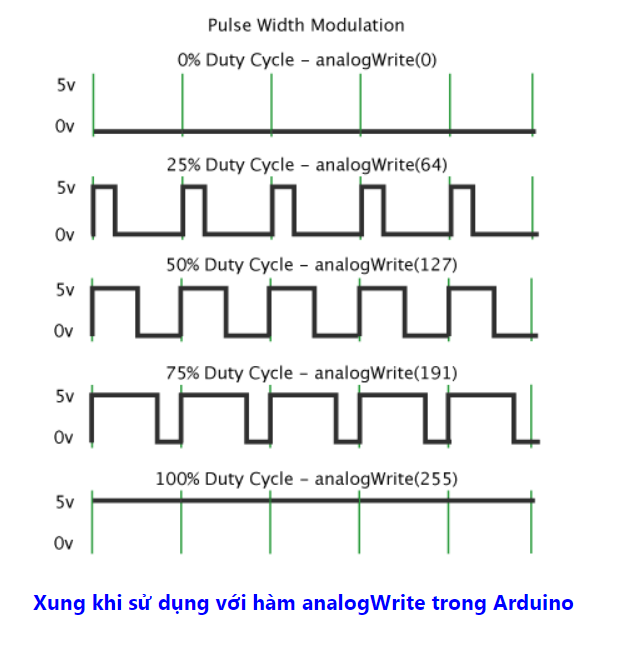
* + Cú pháp: analogReference(type);
    - Type một trong các kiểu giá trị sau: DEFAULT, INTERNAL, INTERNAL1V1, INTERNAL2V56, hoặc EXTERNAL.
* LƯU Ý: Nếu bạn sử dụng kiểu EXTERNAL cho hàm analogReference thì bạn BUỘC phải cấp cho nó một nguồn nằm trong khoảng từ 0-5V, và nếu bạn đã cấp nguồn điện thỏa mãn điều kiện trên vào chân AREF thì bạn BUỘC phải gọi dòng lệnh analogReference(EXTERNAL) trước khi sử dụng analogRead() (NẾU không mạch của bạn có thể sẽ hỏng)



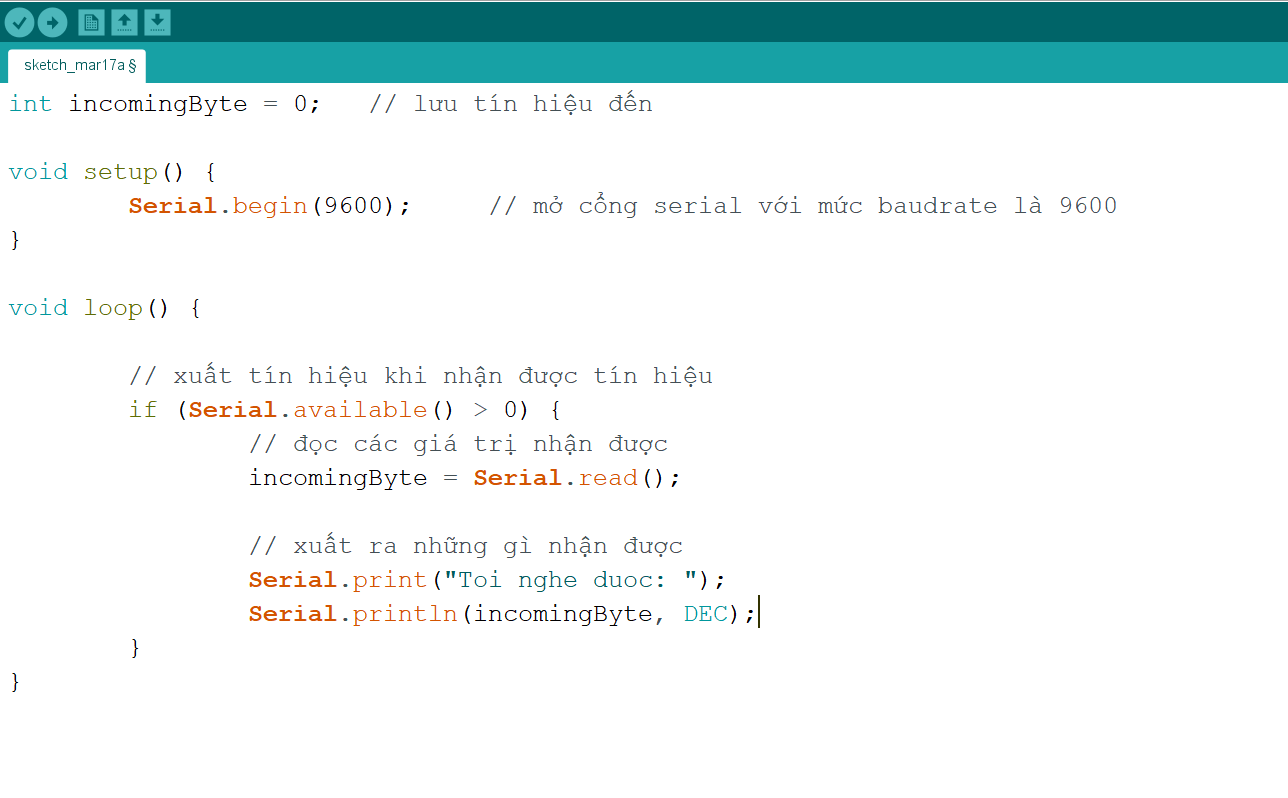
* analogRead(): Nhiệm vụ của analogRead() là đọc giá trị điện áp từ một chân Analog (ADC), analogRead() **luôn** trả về 1 số nguyên nằm trong khoảng từ 0 đến 1023 tương ứng với thang điện áp (mặc định) từ 0 đến 5V. Bạn có thể điều chỉnh thang điện áp này bằng hàm analogReference()
  + Cú pháp: analogRead([Chân điện áp]);
* 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: analogWrite([Chân phát xung PWM], [Giá trị phát xung PWM]);

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

* Xung PWM: Xung là các trạng thái cao / thấp (HIGH/LOW) 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).
  + Tần số: để tính toán ra được thời gian của 1 xung
  + Chu kì xung: bao nhiêu thời gian xung có mức áp cao, bao nhiêu thời gian xung có mức áp thấp.

* 1. Thư viện Serial giao tiếp giữa các mạch Arduino.
* Thư viện Serial được dùng trong việc giao tiếp giữa các board mạch với nhau (hoặc board mạch với máy tính hoặc với các thiết bị khác). Tất cả các mạch Arduino đều có ít nhất 1 cổng **Serial** (hay còn được gọi là UART hoặc USART). Giao tiếp Serial được thực hiện qua 2 cổng digital 0 (RX) và 1 (TX) hoặc qua cổng USB tới máy tính. Vì vậy, nếu bạn đang sử dụng các hàm của thư viện Serial này, bạn không thể sử dụng các chân digital 0 và digital 1 để làm việc khác được!
* Bạn có thể sử dụng bảng Serial monitor có sẵn trong Arduino IDE để giao tiếp với Arduino qua giao thức Serial. Kích vào **Serial Monitor** trong tools hoặc nhấn tổ hợp phím **Ctrl+Shift+M** để mở bảng Serial Monitor, sau đó bạn kích chuột vào bảng chọn như hình dưới để chọn baudrate giống với baudrate được dùng trong quá trình lập trình của bạn. Mặc định là 9600.
* **begin():** Khởi động một cổng Serial với một baudrate cho trước có trên Arduino. Để giao tiếp với máy tính, bạn phải dùng một trong các mức baudrate sau: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, hoặc 115200. Ngoài ra, bạn có thể thay thế mức baudrate khác những mức trên trong trường hợp giao tiếp với một mạch nào đó có sẵn mức baudrate xác định và không thay đổi được.
  + Cú pháp: Serial1.begin(speed);
  + speed:  bits / giây (baud) -> long
* **available():** Trả về số byte (ký tự) tối đa mà ta có thể đọc qua Serial. Các dữ liệu đến được lưu vào một bộ nhớ đệm có dung lượng 64KB.
  + Cú pháp: Serial.available();



* **end():** Đóng cổng Serial, để dùng các chân digital 0, 1 hoặc các chân tương tự trên (Arduino Mega) cho công tác pinMode. Để bật lại, bạn dùng Serial.begin()[.](http://arduino.vn/reference/library/serial/1/huong-dan-ham/end)
  + Cú pháp: Serial.end();
* **print():** Hàm này sẽ xuất dữ liệu ra cổng Serial dưới dạng chuỗi con người có thể đọc được. Hàm này có thể được sử dụng dưới nhiều dạng khác nhau. Các chữ số của một số (nguyên hoặc thực) được chuyển thành chuỗi và xuất ra màn hình. Tham số thứ 2 (có thể có hoặc không) sẽ giúp hệ thống Arduino in dữ liệu dưới dạng mà bạn muốn (thường là dùng để debug). Các giá trị hợp lệ là:
  + - BIN: in dữ liệu dưới dạng hệ nhị phân (hệ cơ số 2)
    - DEC: in dữ liệu dưới dạng hệ thập phân (hệ cơ số 10)
    - OCT: in dữ liệu dưới dạng hệ bát phân (hệ cơ số 8)
    - HEX: in dữ liệu dưới dạng hệ thập lục phân  (hệ cơ số 16)
    - Cứ pháp: Serial1.print(value, format);
    - value: Giá trị bất kì
    - format: như trên (có thể có hoặc không)

