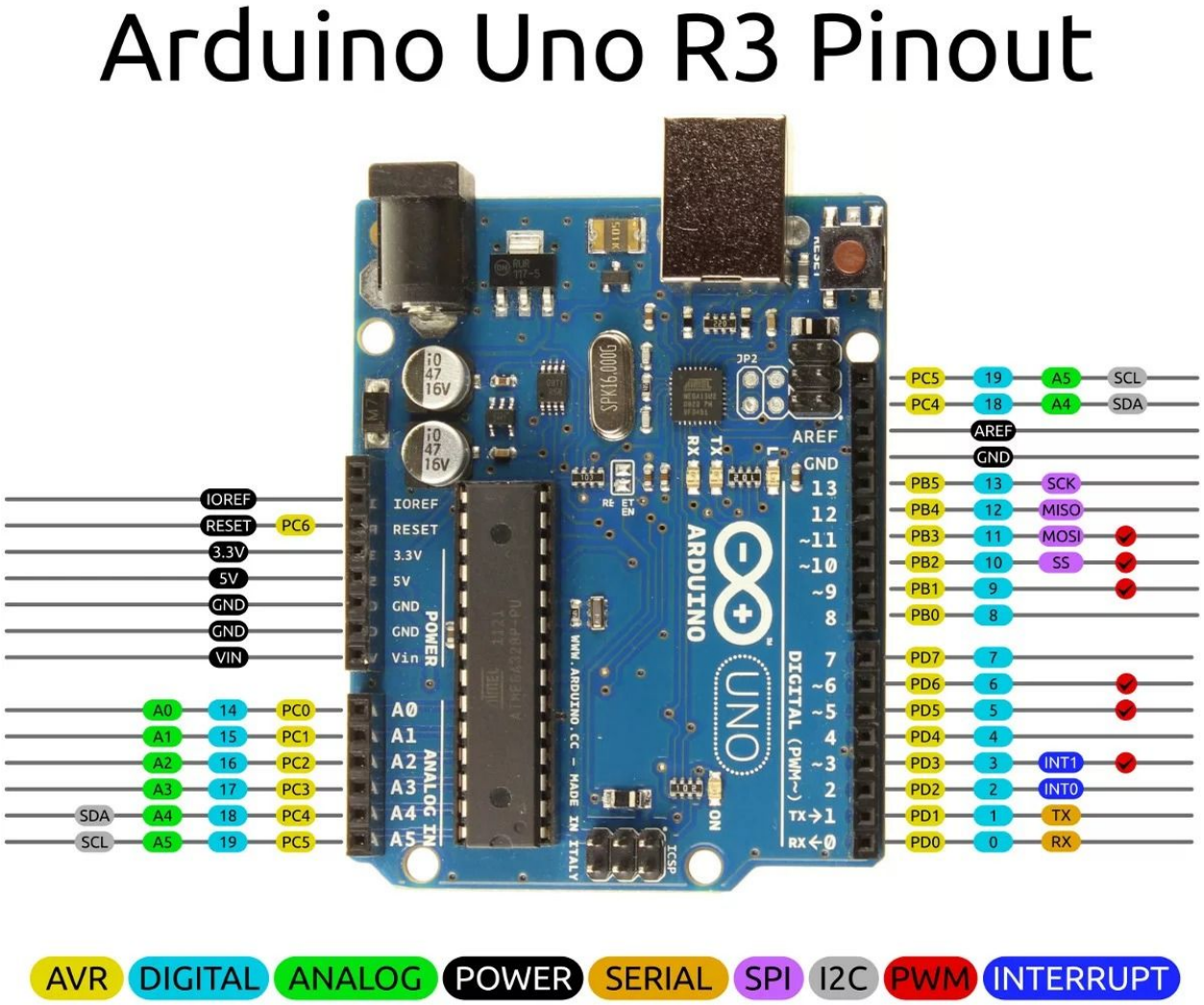
VI ĐIỀU KHIỂN

Microcontroller

Giáo viên giảng dạy: BTTT
Biên soạn: NVAT

I. Thông số kỹ thuật **Arduino Uno R3**:



- ATmega328P based Microcontroller
- Volt vận hành là 5V
- Volt input 7V - 12V
- Digital pins: 14
- Digital (PWM) pins: 6 (pin có dấu ~) (3, 5, 6, 9, 10, 11) (8-bit: 0 -> 255)
- Analog pins: 6 (10-bit: 0 -> 1023)
- DC current cho mỗi I/O pin: 20mA
- DC current sử dụng cho 3.3V pin: 50mA

- Flash memory ~32KB, 0.5KB sử dụng bởi boot loader
- SRAM là 2KB
- EEPROM là 1KB
- Clock pin (CLK) có tốc độ 16MHz. Mỗi chu kỳ clock pin sẽ đẩy 1 bit ra DATA pin… (*không dạy*)
- LED trên mạch dùng chung chân 13.

II. Arduino IDE:

```
void setup(){}   // hàm khởi tạo, được chạy mỗi khi Arduino khởi động
void loop(){}    // lặp đi lặp lại sau khi hàm khởi tạo chạy
```

Hàm I/O cơ bản:

Digital pin

- `pinMode(pin, mode)`

Thiết lập `pin` vận hành theo chế độ `mode` (`INPUT` hoặc `OUTPUT`)

- `digitalWrite(pin, value)`

Viết ra `pin` giá trị `value` (`HIGH` hoặc `LOW`)

- `digitalRead(pin)`

Đọc giá trị `pin`. Trả về `HIGH` hoặc `LOW`.

Analog pin

- `analogReference(type)` (cô nói học cho biết, không ra)

"Map" vùng giá trị volt từ **analog input** vào một vùng giá trị **volt internal**.

```
                  _______  -> 1023
                    3.0v

________          _______  -> ~300
  1.2V

  0.0V              0.0v
________          _______  -> 0
```

Analog Volt Internal | Vùng giá trị

Vd: Giả sử vùng giá trị volt từ analog input là **1.2V**, và vùng giá trị của *DEFAULT* là 3.3 volt. Vì cơ chế Analog sử dụng việc viết Volt từ thấp tới cao để mô tả giá trị từ 0->1023, nếu ta không bao giờ viết tới 3.3V thì ta cũng không bao giờ đạt được max là 1023.
Bằng cách chỉnh vùng giá trị internal là 3.3V xuống thấp hơn, hoặc cao hơn, ta có thể tận dụng tối ưu miền 1024 giá trị, thu được giá trị đọc chính xác hơn. Mặc định **type** là **DEFAULT**.

- `analogRead(pin)`

- **Chỉ có thể** đọc trên pin Analog.
- Đọc giá trị trả về là số nguyên từ 0 -> 1023 (2^10)
- Sử dụng thiết bị A/D Converter, là thiết bị có 10 bit, để đọc. (thông tin thêm)

- `analogWrite(pin, value)`

- **Chỉ có thể** viết ra pin digital PWM (chân digital có dấu ~ ở trước).
- Viết analog, chỉ viết đc từ 0->255 (2^8)
- Sử dụng thiết bị PWM timer, là thiết bị có 8 bit, để viết. (thông tin thêm)

Hàm I/O nâng cao:

- `shiftOut(dataPin, clockPin, bitOrder, value)` (cô nói hàm này khó, chắc bỏ qua được)

`shiftOut()` có nhiệm vụ chuyển 1 `byte` (gồm 8 bit) ra ngoài từng bit một. Bit được chuyển đi có thể được bắt đầu từ bit nằm bên trái nhất (**leftmost**) hoặc từ bit nằm bên phải nhất (**rightmost**). Các bit này được xuất ra tại chân `dataPin` sau khi chân `clockPin` được `pulsed` (có mức điện thế là **HIGH**, sau đó bị đẩy xuống **LOW**).

Tham số:

- dataPin: pin sẽ được xuất ra tín hiệu (*`int`*)
- clockPin: pin dùng để xác nhận việc gửi từng bit của **dataPin** (*`int`*)
- bitOrder: một trong hai giá trị **MSBFIRST** hoặc **LSBFIRST**. (Bắt đầu từ bit bên phải nhất hoặc Bắt đầu từ bit bên trái nhất)
- value: dữ liệu cần được shiftOut. (*`byte`*)

`shiftOut()` chỉ xuất được dữ liệu kiểu byte. Nếu bạn muốn xuất một kiểu dữ liệu lớn hơn thì bạn phải shiftOut 2 lần (hoặc nhiều hơn), mỗi lần là 8 bit.

Code nguồn của hàm:

```
void shiftOut(uint8_t dataPin, uint8_t clockPin, uint8_t bitOrder, uint8_t val)
{
	uint8_t i;
	for (i = 0; i < 8; i++) {
		if (bitOrder == LSBFIRST) digitalWrite(dataPin, !!(val & (1 << i)));
		else digitalWrite(dataPin, !!(val & (1 << (7 - i))));
		digitalWrite(clockPin, HIGH); // clockPin == HIGH khi shift xong 1 bit
		digitalWrite(clockPin, LOW);  // clockPin == LOW khi chuẩn bị shift bit tiếp theo
	}
}
```

- `pulseIn(pin, value, timeout)`

Đọc một **pulse** (hoặc là **HIGH** hoặc là **LOW**) trên pin.
Ví dụ: Với **value** là **HIGH**, **pulseIn()** chờ cho **pin** đi từ **LOW** tới **HIGH**, bắt đầu đếm giờ, sau đó chờ cho pin về **LOW** và dừng đếm giờ. Trả về độ dài của pulse đó ở micro-giây hoặc là trả về 0 nếu việc đọc pulse không được hoàn tất trong khoảng thời gian **timeout**.

- **pin**: Chân mà muốn đọc pulse (int)
- **value**: loại xung mà bạn muốn đọc (**HIGH** or **LOW**) (int)
- **timeout**: khoảng thời gian (micro-giây) để chờ pulse bắt đầu (chờ **pin** có giá trị là **value**). Mặc định **timeout** là 1 giây.

Trả về: Độ dài của pulse (micro-giây) hoặc 0 nếu không thấy pulse trong khoảng thời gian **timeout**.

Hàm timer:

- `delay(ms)`

Chờ **ms** *(unsigned long)* mili-giây. Trả về: **void**

- `delaymicroseconds(us)`

Chờ **us** *(uint)* micro-giây. Trả về: **void**

- `millis()`

Trả về lượng mili-giây kể từ khi mạch arduino bắt đầu chạy. *(ulong)*

- `micros()`

Trả về lượng micro-giây kể từ khi mạch arduino bắt đầu chạy. *(ulong)*

Hàm truyền thông:

- `Serial.begin(speed)` Bắt đầu giao tiếp Serial. **Speed** nên luôn để **9600**.
- `Serial.available()` Trả về số lượng Bytes sẵn sàng để được đọc.
- `Serial.read()` Đọc 1 Byte đầu tiên trong Serial, trả về Byte đó. (kiểu **int**)
- `Serial.print(val)` In ra **val**. **val** có thể là **int**, **char**, **byte**,...
- `Serial.println(val)` Giống như trên nhưng có kèm thêm ký tự xuống dòng.

Hàm ngắt:

Mạch **Arduino Uno** chỉ có 2 ngắt **INT.0** và **INT.1** tương ứng với chân digital 2 và chân digital 3. Để biết được chân digital nào ứng với ngắt nào, sử dụng: **`digitalPinToInterrupt()`**. Ví dụ, gọi **`digitalPinToInterrupt(3)`** sẽ trả **`1`** tương ứng với ngắt số 1 (**INT.1**).

- `attachInterrupt(digitalPinToInterrupt(pin), ISR, mode)`

Gắn ngắt ở **pin** nếu xảy ra theo chế độ **mode** thì thực hiện hàm **ISR**. **ISR** là một function không nhận tham số và không trả về.

Params:
- **interrupt**: the number of interrupt (int)
- **pin**: pin number
- **ISR**: the interrupt service routine (ISR) to call when the interrupt occurs; This **function** must take no params and return nothing. This function is sometimes called ISR
- **mode**: defines when the interrupt should be triggered. (LOW, CHANGE, RISING, FALLING)

Returns: **None**

- `detachInterrupt(interrupt)`

Params:
- **interrupt**: số của interrupt mà bạn muốn gỡ hàm xử lý interrupt của nó.

Returns: **None**

- `Interrupts(); noInterrupts();`

Bật, hoặc tắt cơ chế Interrupt trên Arduino. Khi tắt, Arduino sẽ không phản ứng với interrupts.
Params: None
Returns: None

Hàm toán học:

- `min(x, y)`
- `max(x, y)`
- `random(min, max)`

Returns: a random number in `[min, max)`

- `abs(x)`
- `pow(base, exponent)`
- `sqrt(x)`
- `constrain(x, a, b)`

Params:

- **x**: number to constrain
- **a**: lower end
- **b**: higher end

Returns:

- **x**: if **a <= x <= b**
- **a**: if **x < a**
- **b**: if **b < x**

- `map(value, fromLow, fromHigh, toLow, toHigh)`

Ánh xạ các giá trị ở đoạn giá trị [***fromLow, fromHigh]*** đến đoạn giá trị ***[toLow, toHigh]***. Sau đó trả về ***giá trị ở*** **[toLow, toHigh]** mà tương ứng với ***value*** **nằm ở [fromLow, fromHigh]**.

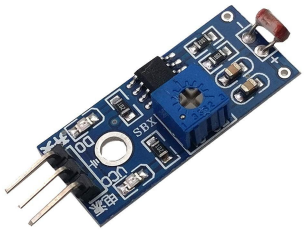
Ví dụ:

`Value = 5, fromRange=[0, 10], toRange[0, 100] thì trả về sẽ là 50.`

Tưởng tượng như chấm một điểm lên sợi dây su và kéo nó ra, điểm chấm nó ở đâu thì đó là giá trị mà hàm `map()` trả về.

III. Cảm biến:

Cảm biến cảm nhận ánh sáng (Light sensor)



Không dùng thư viện

Tên model: *LM393 Light Sensor*

Bốn chân:

- **VCC (5V), GND.**
- **D0**: Độ sáng *Sáng hơn ngưỡng* sẽ LOW, Độ sáng *Tối hơn ngưỡng* sẽ cho HIGH.

Ngưỡng là một giá trị điều chỉnh được (tăng lên, tăng xuống) bởi chiết áp 4 chấu trên mạch.

- **A0**: Sáng cho giá trị thấp, Tối cho giá trị cao

```
int a = A0; // analog pin vào A0
int d = 7;  // digital pin vào 7

void setup() {
  pinMode(a, INPUT);  pinMode(d, INPUT);   Serial.begin(9600);
}

void loop() {
  int va = analogRead(a);   int vd = digitalRead(d);
  Serial.print("Analog = "); Serial.println(va);
  Serial.print("Digital = "); Serial.println(vd);
  if (vd == 0) {
    Serial.println("Troi sang");
  } else {
    Serial.println("Troi toi");
  }
  delay(500);
}
```

Cảm biến nhiệt độ (Thermistor)

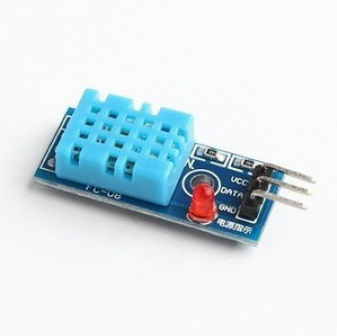
Cảm biến nhiệt độ số (Digital) (***không dùng***)

Cảm biến nhiệt độ tương tự (Analog)

Không dùng thư viện
Tên model: *LM35*
Ba chân:
- **VCC (5V), GND.**
- **A0 (analog):**
Nhiệt độ cao, điện trở thấp và giá trị cao. Nhiệt độ thấp, điện trở cao và giá trị thấp.

```
int a = A0; // Output gan vao chan A0
void setup() {
  pinMode(a, INPUT); Serial.begin(9600);
}

void loop() {
  float sensor = analogRead(a);
  float tempC = (sensor * 5000) / 1024 / 10;
  Serial.println(tempC); // nhiet do C
  if (tempC > 31.0) Serial.println("Nhiet do cao - Hot! ");
  else Serial.println("Nhiet do thap - Cold!");
  delay(500);
}
```

Cảm biến độ ẩm và nhiệt độ (Humidity and Temperature Sensor)

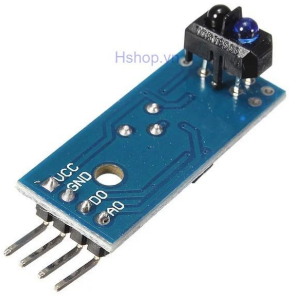


Có dùng thư viện, thư viện không có sẵn
Tên model: *DHT11*
Ba chân:

- VCC (5V), GND
- DATA (digital)

```
//https://www.circuitbasics.com/wp-content/uploads/2015/10/DHTLib.zip
// Tải .zip về > Arduino IDE Menu > Sketch > Include Library >
// > *Add ZIP Library... > Chọn file ZIP vừa tải
// Lưu ý: Nếu khác Thư viện, code bên dưới sẽ không chạy.
#include <dht.h>

int DHT_IN = 7; // chan Digital input cua sensor
dht dht_sensor;

void setup() {
  Serial.begin(9600);
}
void loop() {
  int check = dht_sensor.read11(DHT_IN); // DHT11
  float h = dht_sensor.humidity; // % do am
  float t = dht_sensor.temperature; // nhiet do C
  Serial.print("Do am: "); Serial.print(h); Serial.println(" %");
  Serial.print("Nhiet do: "); Serial.print(t); Serial.println(" C");
  delay(500);
}
```

Cảm biến dò line (Line-tracking Sensor)



Không dùng thư viện
Tên model: TCRT5000
Bốn chân:

- **VCC (5V), GND**.
- D0 (Digital)
- A0 (Analog)

Bề mặt càng trắng A0 càng thấp, ngược lại A0 càng cao.
Bề mặt trắng dưới *ngưỡng* thì D0 là LOW, ngược lại D0 là HIGH.
Ngưỡng là một giá trị điều chỉnh được (tăng lên, tăng xuống) bởi chiết áp 4 chấu trên mạch.

```
const int d = 7; // d0 vao Digital 7
const int a = A0; // a0 vao Analog 0

void setup() {
  pinMode(d, INPUT); pinMode(a, INPUT); Serial.begin(9600);
}

void loop() {
  int va = analogRead(a); // WHITE gia thi thap | else gia tri cao
  int vd = digitalRead(d); // WHITE thi LOW | else HIGH
  Serial.print(va); Serial.print(" | "); Serial.println(vd);
  if (vd == LOW) Serial.println("Da thay White LINE!!");
  else Serial.println("Khong thay LINE..");
  delay(100);
}
```

Cảm biến siêu âm (Ultrasonic Sensor)

Không dùng thư viện
Tên model: HC-SR04
Bốn chân:

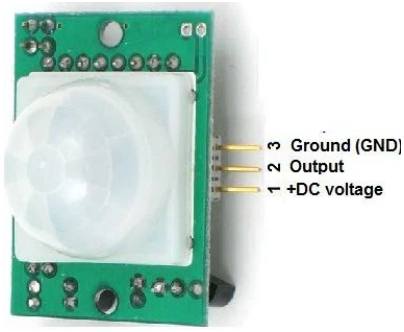
- **VCC (5V), GND**
- **TRIG, ECHO** (cả hai là Digital)

Cách hoạt động:

- Chân **TRIG** dùng để phát ra tín hiệu, tín hiệu sau đó va chạm vào mặt phẳng phía trước và phản lại, sau đó sensor sẽ bắt tín hiệu đó và viết ra chân **ECHO**.
- Khoảng thời gian phát ra và thu về (sẽ được tính bằng hàm `pulseIn()`) được sử dụng để tính khoảng cách, nhân nó cho một hằng số để có được (*cm*) hoặc (*inch*).

```
const int trigPin = 6; // TRIG gắn vào digital 6
const int echoPin = 7; // ECHO gắn vào digital 7
void setup() {
  pinMode(trigPin, OUTPUT); pinMode(echoPin, INPUT);
}

void loop() {
  digitalWrite(trigPin, LOW);  // set LOW một lúc để clear
  delayMicroseconds(2);
  //
  digitalWrite(trigPin, HIGH); // set trigPin HIGH để phát tín hiệu
  delayMicroseconds(10);
  //
  digitalWrite(trigPin, LOW);  // set trigPin về LOW, dừng phát
  //
  int duration = pulseIn(echoPin, HIGH); // đọc độ dài pulse ở echoPin
  //
  double cm = (double) duration / 2 / 29.1; // chia cho hằng số
  Serial.print(cm); Serial.println("cm");
  // cm lúc này là khoảng cách từ Sensor đến
  // bức tường gần nhất ở Centimeter.
}
```

Cảm biến hồng ngoại số phát hiện di chuyển (Digital Infrared Motion Sensor)



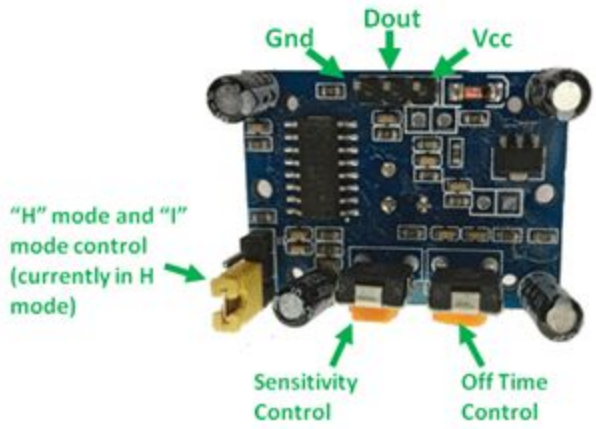
Không dùng thư viện

Tên model: HC-SR501

Ba chân: VCC (5V), GND, OUTPUT (Digital)

Tips:

- Tên của Chân được viết dưới cái nhựa hình mái vòm, cậy nó ra để xem đúng chân.



Cách hoạt động: Sensor có hai mode hoạt động:

1. *Kích hoạt một lần*

- Khi không có chuyển động, OUTPUT là **LOW**.
- Khi phát hiện chuyển động, chân OUTPUT cho ra **HIGH** trong 5s kế tiếp, sau đó 5s kế tiếp sẽ luôn cho ra **LOW** để reset chân output. Trong 10s từ lần phát hiện chuyển động, sensor không phát hiện lại chuyển động.

2. *Kích hoạt nhiều lần*

- Thay vì phát hiện một lần HIGH 5s tiếp và LOW 5s tiếp thì nó sẽ luôn cho HIGH khi vẫn thấy chuyển động. Khi không thấy nữa sẽ LOW 5s kế tiếp để reset value.

Chuyển mode: Để chuyển mode, tìm tại 4 cái trụ (tụ điện) ở bốn góc sẽ có một góc có cái phích (thường là màu vàng) + một chân lòi ra ở bên hông của sensor. Nếu tháo nó ra và lắp qua chân kia một chân là đã chuyển mode. Nếu làm mất gói thì không rõ là mode gì.

Chiết áp Sensitivity: chỉnh độ nhạy khi detect motion, vặn ngược chiều đồng hồ là càng nhạy.

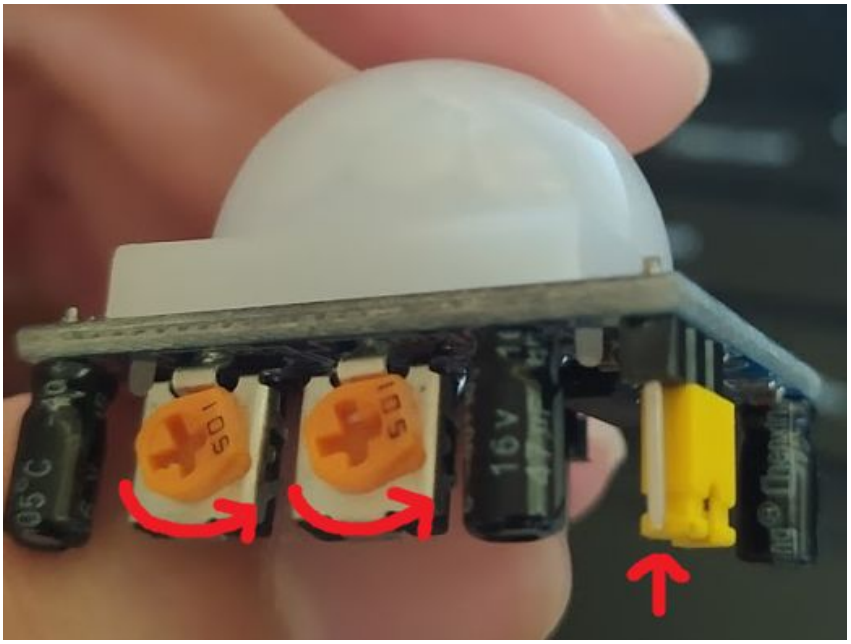
Chiết áp Off-time: 5s là thời gian ngắn nhất có thể, vặn ngược chiều là càng ngắn.

Code thử

```
void setup() {
  pinMode(7, INPUT); // OUTPUT vào D7
  Serial.begin(9600);
}

void loop() {
  Serial.println(digitalRead(7));
  //delay(100);
}
```

Chú thích

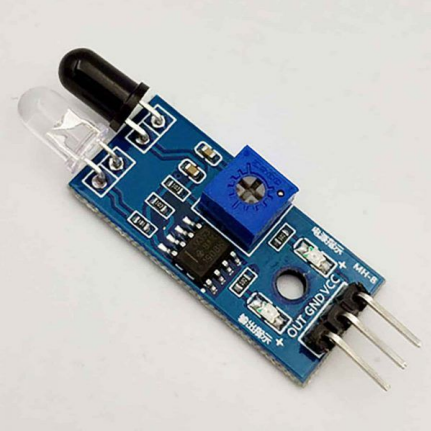
Sensor này hay lỗi. Mình thì không có vấn đề gì, nhưng các bạn khác gặp trục trặc khi sử dụng sensor này. Để sử dụng giống mình, đảm bảo chiết áp và cái túi vàng trên sensor ở trạng thái như sau:



Chiết áp hãy vặn theo chiều mình vẽ hết mức có thể. Còn cái túi vàng hãy để lộ chân như trên hình. Như vậy, nó sẽ ở mode detect một lần. Tức là khi có chuyển động, 5s đầu luôn là HIGH, 5s tiếp theo luôn là LOW rồi sẽ quay lại chế độ sẵn sàng detect motion.

Còn nếu nó vẫn hoạt động không đúng ý muốn thì … chắc đành sử dụng con ở bên dưới thay thế nó. Tuy nhiên con bên dưới không detect chuyển động mà chỉ detect vật thể.

Cảm biến vật cản hồng ngoại (infrared proximity/obstacle detection sensor)



(không có trong sách nhưng bỏ vào đây để đề phòng)
Không dùng thư viện
Tên model: *FC-51 Obstacle sensor*
Ba chân:

- **VCC**: 5V
- **GND**: GND
- **OUT**: Digital. Nếu có vật thể trong vùng detection thì sẽ cho HIGH, ngược lại sẽ cho LOW. Điều chỉnh độ rộng của vùng detection bằng cách vặn chiết áp 4 chấu có trên module.

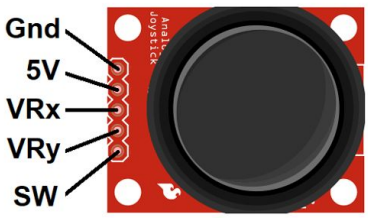
Code thử

```
const int pin = 7;
void setup() {
  pinMode(pin, INPUT);
  pinMode(13, OUTPUT); // LED trên bo
}

void loop() {
  digitalWrite(13, digitalRead(pin));
}
```

Chú thích

Ngoài ra, có thể test sensor mà không cần cắm dây **OUT** bằng đèn LED ở trên chữ **OUT** trên mạch của sensor. Nếu có vật thể trong vùng detection nó sẽ sáng lên, ngược lại nó sẽ mặc định là tối.

Joystick Module



Không dùng thư viện
Tên model: *KY-023 Dual Axis Joystick Module*
Năm chân:
VCC (5V), GND, VRx (Analog), **VRy** (Analog), **SW** (Digital).
- **VRx, VRy, SW** đều là chân **INPUT**.
- **VRx,y** ở vị trí thăng bằng có giá trị 511
- **SW** trả về **LOW** nếu núm được nhấn xuống, ngược lại **HIGH**. Giá trị bị ngược lại là bởi vì pin phải được set ở mode **INPUT_PULLUP**.

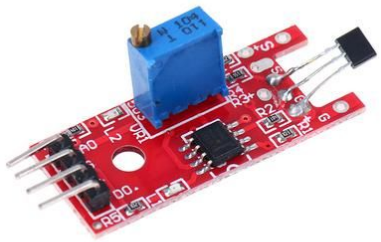
```
const int vx = A0;
const int vy = A1;
const int sw = 7;

void setup() {
  pinMode(vx, INPUT); pinMode(vy, INPUT);
  pinMode(sw, INPUT_PULLUP); // neu chi la INPUT thi phai mac dien tro
  Serial.begin(9600);
}

void loop() {
  int x = analogRead(vx);
  int y = analogRead(vy);
  int s = digitalRead(sw); // HIGH neu khong bam, LOW neu bam
  Serial.print("X = "); Serial.println(x);
  Serial.print("Y = "); Serial.println(y);
  Serial.print("s = "); Serial.println(s);
  delay(500);
}
```

Cảm biến Gas (Gas sensor)

Không dùng thư viện
Tên model: MQ2 Gas Sensor
Bốn chân: VCC (5V), GND, A0 (Analog), D0 (Digital).
- **A0**: Thể hiện nồng độ khí Gas/Smoke đang cảm nhận. Nồng độ càng cao thì giá trị **A0** càng cao.
- **D0**: Nếu chỉ detect smoke thì không cần quan tâm chân này.

```
const int MQ2pin = A0;

void setup() {
  Serial.begin(9600);
  Serial.println("Gas sensor warming up!");
  delay(20000); // cho phep MQ-2 khoi dong
}

void loop() {
  int sensorValue = analogRead(MQ2pin); // read analog input pin 0

  Serial.print("Sensor Value: ");  Serial.print(sensorValue);
  if(sensorValue > 200) Serial.print("!! Smoke detected !!");
  Serial.println("");
  delay(1000);
}
```

Cảm biến Hall (Hall sensor)



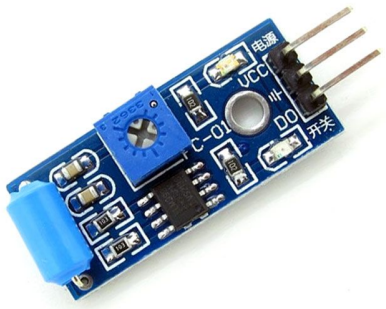
Không dùng thư viện
Tên model: KY-024, *Hall sensor là được.*
Bốn chân:

- **VCC (5V), GND**
- D0 (Digital)
- A0 (Analog)

D0 **HIGH** khi không có từ, **LOW** khi có từ.
A0 có *giá trị cao* khi **không có từ**, thấp khi **có từ**. Cũng sử dụng ngưỡng và sử dụng điều chỉnh chiết áp để chỉnh ngưỡng.

```
const int d = 7;
const int a = A0;

void setup() {
  pinMode(d, INPUT); pinMode(a, INPUT); Serial.begin(9600);
}

void loop() {
  int vd = digitalRead(d);
  int va = analogRead(a);
  Serial.print("Vd: "); Serial.println(vd);
  Serial.print("Va: "); Serial.println(va);
  if (vd) Serial.println(“Khong co tu.”);
  else Serial.println(“Co tu.”);
  delay(200);
}
```

Cảm biến rung (Vibration sensor)



Không dùng thư viện
Tên model: SW-420
Ba chân:

- **VCC (5V), GND**
- D0 (Digital)

D0 **LOW** nếu có rung động. Không có gì hết là **HIGH**.

```
const int d = 7;

void setup() {
  pinMode(d, INPUT);
  Serial.begin(9600);
}

void loop() {
  int vd = digitalRead(d);
  if (vd) Serial.println("Khong co rung!");
  else Serial.println("Co rung.");
  delay(500);
}
```

Cảm biến cháy (Flame sensor)

Không dùng thư viện

Tên model: KY-026 FLAME SENSOR

Có bốn chân:

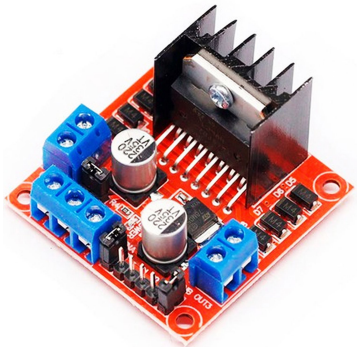
- **VCC (5V), GND**
- **D0**
- **A0**

Khi phát hiện có ngọn lửa (tia lửa nhưng phải nhiều tí) thì chân **D0** cho **LOW** và ngược lại, không có cho **HIGH**. **A0** cũng vậy, nhưng cho giá trị cao khi có lửa, ngược lại giá trị thấp.

```
const int d = 7;
void setup() {
    pinMode(d, INPUT);
    Serial.begin(9600);
}
void loop() {
    int sensorVal = digitalRead(d);
    Serial.println(sensorVal);
    if (sensorVal == HIGH) Serial.println("Khong co Lua");
    else Serial.println("Co lua”);
}
```

IV. Động cơ:

Động cơ DC (DC Motor 5V + L298N)

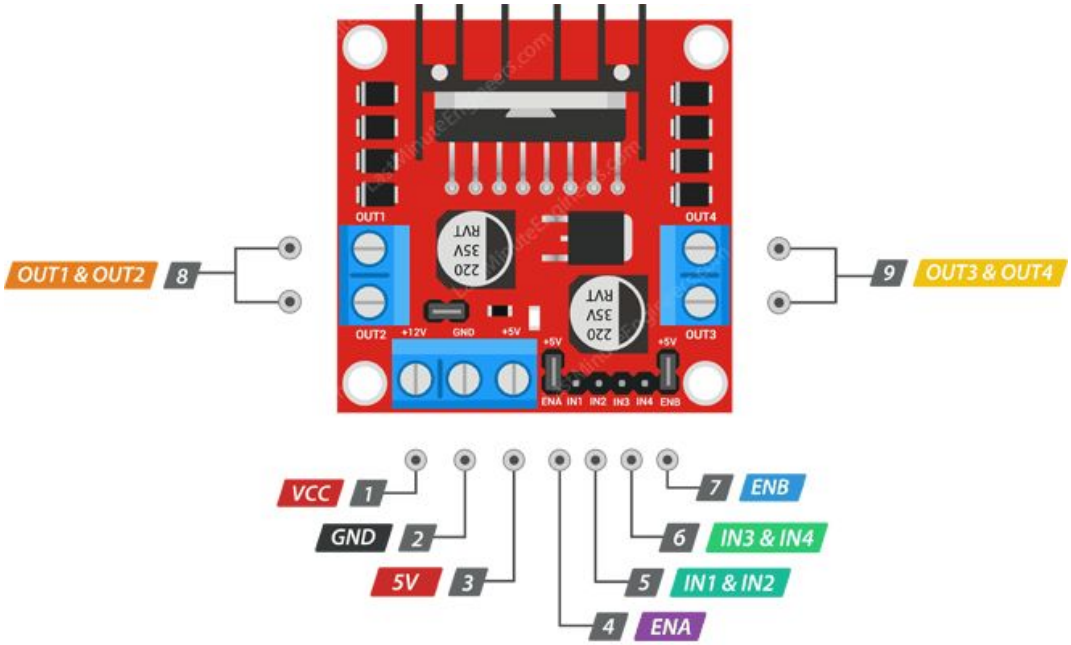


Không dùng thư viện

- Ta sẽ sử dụng động cơ DC 5V nhỏ (loại tròn).
- Để điều khiển nó (xoay chiều và đổi tốc độ) ta sử dụng mạch điều khiển **L298N** (*hình bên trái*)
- Ta sẽ không dùng nguồn điện ngoài như các tài liệu khác mà chỉ dùng trên Arduino

Đầu vào ra L298N

Ta sẽ sử dụng tên các đầu vào ra của L298N như hình bên dưới đây:



Một L298N có thể điều khiển được đến hai động cơ DC, nên mới có nhiều đầu vào ra như vậy. Ý nghĩa của các đầu vào ra là:

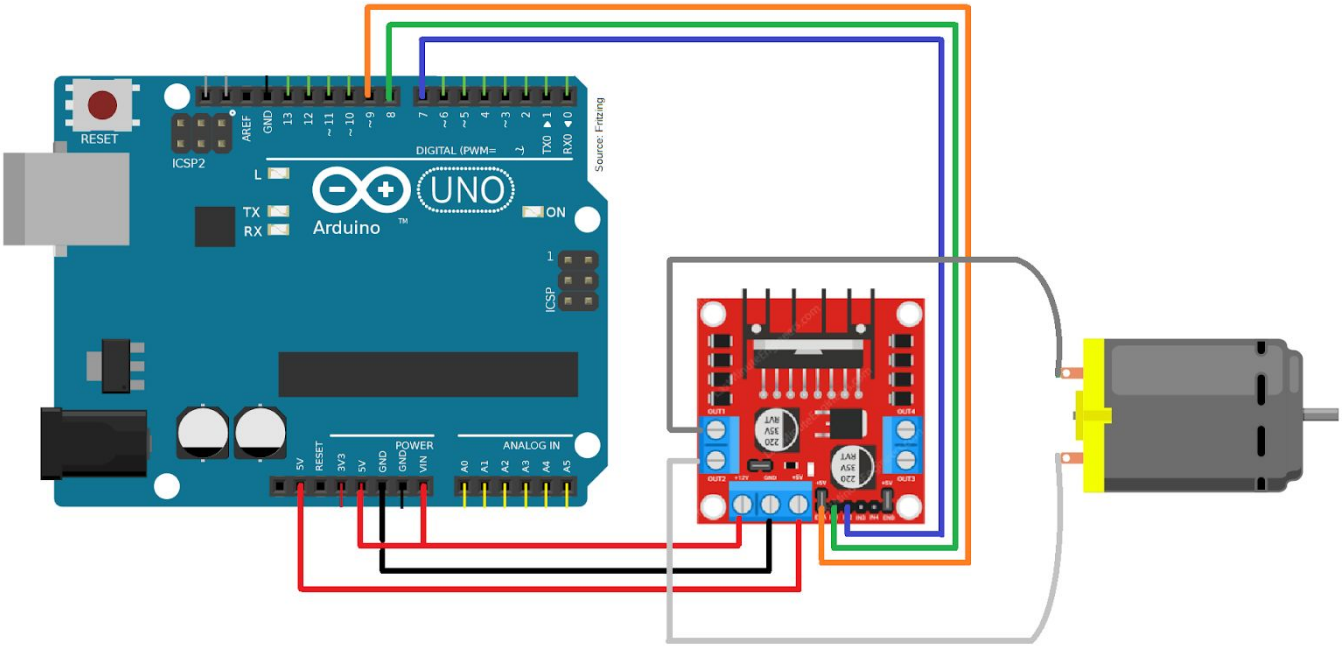
- **OUT1 & OUT2**: Đây là hai đầu mà bạn sẽ nối với hai chân của động cơ DC. (Vặn vít lên, nhét đầu đực của dây vào, vặn vít xuống để khóa nó; còn đầu đực còn lại thì cứ bẻ cong nó rồi móc vào DC)
- **OUT3 & OUT4**: Tương tự với động cơ DC thứ 2.

- **VCC**: Đầu này sẽ cấp điện cho Motor. Vì một lý do nào đó, mạch sẽ làm gì đó với nguồn điện vào nên Volt đầu vào sẽ bị giảm đi 2V. Nên để cấp đủ, cắm đầu này thì bạn cần nhét đến hai dây đực-đực. Hai dây này sau đó được gắn vào chân **5V** và chân **Vin** có trên Arduino Uno. (có thể lấy điện từ **Vin** nhưng sẽ không ổn định, đừng dùng nó để cấp điện cho sensor hay những thứ 5V khác)
- **GND**: Gắn vào chân đất GND.
- **5V**: Đây là đầu vào sẽ cấp điện cho mạch logic trên L298N (không phải cho DC). Đầu này nối vào cổng **5V** thứ 2 trên Arduino.
- **ENA**: Là chân điều khiển tốc độ động cơ **A**, có thể gắn nó vào Digital để viết LOW (tắt động cơ) hoặc viết HIGH (bật động cơ). Ngoài ra, nó còn có thể gắn vào chân PWM (chân digital có dấu ~ bên cạnh) để analogWrite vào đó, chỉnh tốc độ động cơ.
- **ENB**: tương tự, điều khiển tốc độ động cơ **B**.
- **IN1 & IN2**: điều khiển chiều động cơ **A**. Cả hai cắm vào hai chân Digital trên Arduino. Nếu hai chân này được viết khác giá trị, động cơ sẽ quay. Cụ thể:

  `IN1 HIGH | IN2 LOW | Quay tới`

  `IN1 LOW | IN2 HIGH | Quay lui`
- **IN3 & IN4**: Tương tự, điều khiển chiều động cơ **B**.

Mắc dây cho một động cơ A



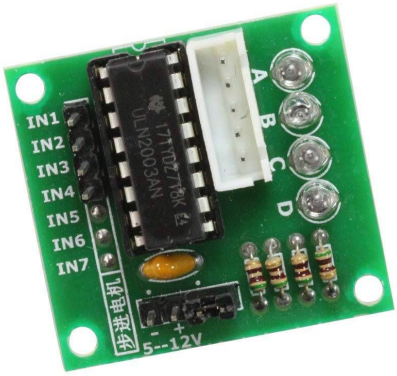
- **OUT1** & **OUT2**: Hai đầu của DC Motor 5V
- **VCC**: nhét 2 dây đực-đực vào, sau đó cắm vào **5V** và **Vin** (chỉ dùng 5V sẽ quay yếu, nhưng có quay)
- **GND**: vào chân đất GND
- **5V**: vào 5V (arduino có 2 cổng 5V)
- **IN1**: vào D7. **IN2**: vào D8. **ENA:** vào D9 (pin PWM).

Code thử

```
const int enA = 9; // chân digital có dấu ~
const int in1 = 8;
const int in2 = 7;

void setup() {
  pinMode(enA, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);

  // lúc đầu tắt motor
  digitalWrite(in1, LOW); digitalWrite(in2, LOW);
}

void loop() {
  // quay tới ở max tốc độ
  digitalWrite(in1, HIGH); digitalWrite(in2, LOW);
  analogWrite(enA, 255);
  delay(2000);

  // quay lui, tốc độ tăng dần
  digitalWrite(in1, LOW); digitalWrite(in2, HIGH);
  for (int sp=0; sp<256; sp++) {
    analogWrite(enA, sp);
    delay(20);
  }

  // tắt động cơ trong 2s
  digitalWrite(in1, LOW); digitalWrite(in2, LOW);
  delay(2000);
}
```

Nếu nạp code mà **trong 10s tới** động cơ DC không quay, hãy thử ngắt nguồn điện của mạch L298N (cả 2 dây VCC và 1 dây 5V) và cắm vào lại thử.

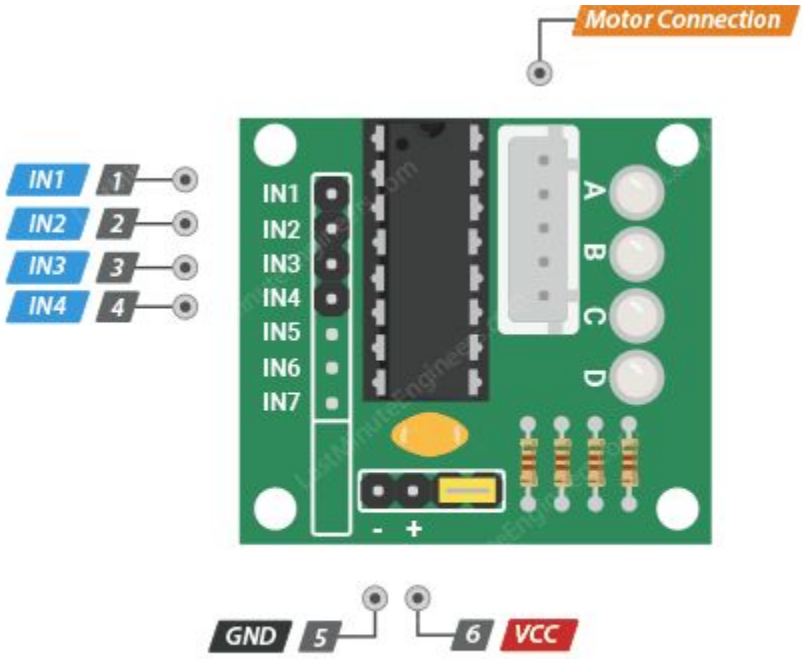
Động cơ bước (28BYJ-28 + ULN2003)



Có dùng thư viện, thư viện có sẵn <Stepper.h>

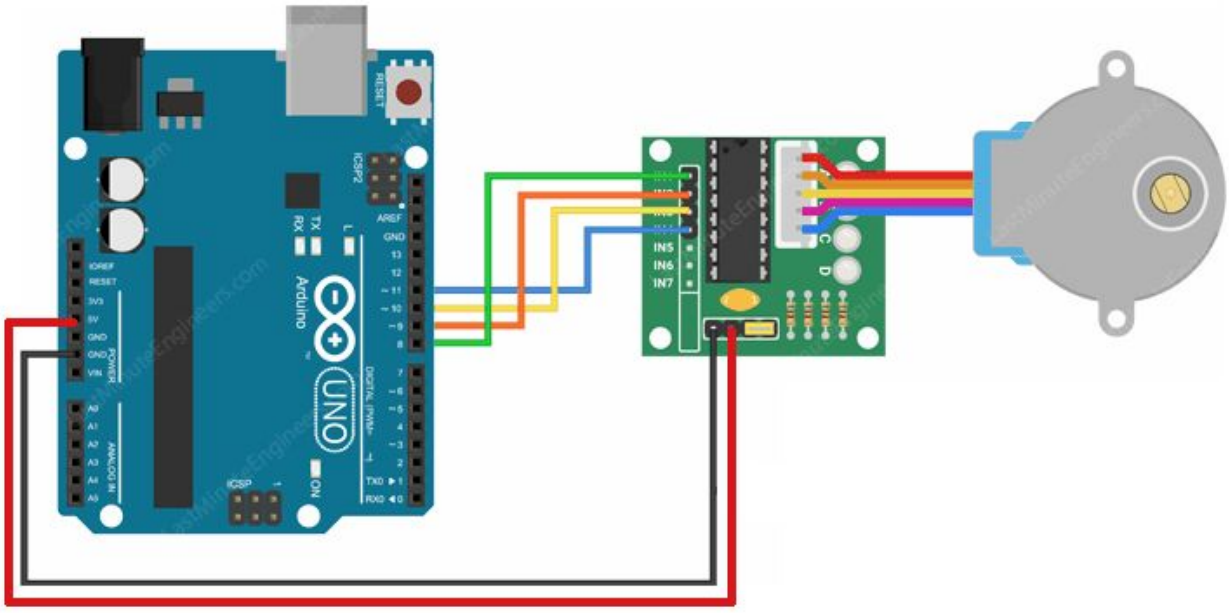
Ta sẽ sử dụng động cơ bước **28BYJ-28** 5V như hình bên trái. Để điều khiển nó, ta cũng dùng một mạch điều khiển là Driver Board ULN2003. Cách cắm dây đơn giản hơn nhiều so với DC Motor.

Đầu vào ra L298N



- **Motor Connection**: là một đầu cắm đặc biệt dành cho 28BYJ-48, bạn sẽ cắm thẳng đầu cắm của vào đây mà không lo gì cả.
- **IN1, IN2, IN3, IN4**: là các chân digital mà thư viện Stepper chúng ta sử dụng sẽ dùng để điều khiển step motor.
- **GND, VCC**: là hai chân nguồn.

Mắc dây vào 28BYJ-48



- **Motor Connection:** Đầu cắm của motor.
- **IN1**: D8, **IN2**: D9, **IN3**: D10, **IN4**: D11
- **VCC**: 5V trên Arduino
- **GND**: GND trên Arduino

Nhiều nguồn tài liệu khuyên không nên cắm thẳng nguồn 5V Arduino vì động cơ 28BYJ-48 rút rất nhiều Ampe nên sẽ có hại cho Arduino. Nhưng chúng ta chỉ làm motor này trong một thời gian ngắn, không để nó chạy lâu dài, kết hợp với việc để mọi thứ đơn giản nên chúng ta sẽ dùng thẳng nguồn do Arduino cấp.

Code thử

```
#include <Stepper.h>

#define STEPS 2038 // số bước trong một vòng quay
// Số trên là hằng số với mỗi loại Stepper Motor.

Stepper stepper(STEPS, 8, 10, 9, 11);
void setup() {
  // không cần làm gì cả, thư viện Stepper sẽ lo setup động cơ cho ta
}

void loop() {
  stepper.setSpeed(3); // 3 rpm
  stepper.step(STEPS); // quay một vòng cùng chiều đồng hồ
  delay(1000);
  stepper.setSpeed(6); // 6 rpm
  stepper.step(-STEPS); // quay một vòng ngược chiều đồng hồ
  delay(1000);
}
```

Chú thích một vài điều quan trọng

- **STEPS** là một hằng số sẽ phụ thuộc vào thiết kế bánh răng của Motor, nên nếu ta dùng 28BYJ-28 thì để hẵng nó là 2038
- Hàm `.setSpeed(rpm)` nhận một tham số RPM đó là số vòng trong một phút, nếu số này càng lớn thì motor quay càng nhanh. **TUY NHIÊN**, mình thực nghiệm thì thấy số này chỉ có thể từ **0 đến 14** thì Motor hoạt động tốt cả hai chiều. Lớn hơn sẽ khiến nó đứng im hoặc chỉ quay một chiều.
- Hàm `.step(steps)` nhận một tham số steps là số bước nó sẽ bước. Để quy chuyển nó ra số độ để quay thì bạn cần phải làm một phép biến đổi:

STEPS=2038 tương ứng với **360 độ.** Để quay 90 độ, tức là 360/4 độ thì bạn cần để **steps=2038/4.** Tức là để quay **X** độ, số steps bạn cần truyền vào là `2038/(360/X)`. Tuy nhiên không nên để X quá nhỏ (mình nghĩ nên lớn hơn 11 độ). Dưới đây là hàm chuyển từ độ sang số bước, bạn có thể sử dụng nó để đỡ phần tính toán:

```
int degreeToSteps(int degree, int STEPS = 2038) {
  if (degree == 0) return 0;
  return STEPS / (360 / degree); // (tính âm dương bảo toàn)
}
```

Động cơ Servo (SG90)

Có dùng thư viện, thư viện có sẵn <Servo.h>
Đây là động cơ dễ nhất trong 3 động cơ.
Ta sẽ sử dụng **SG90**.

Đầu vào ra và mắc dây

SG90 có 3 đầu:

- **GND (màu đà)**: chân đất
- **5V (màu đỏ)**: chân 5V
- **Control** (**màu vàng**): chân PWM (Digital có dấu ~)

Vì chân được đánh dấu phụ thuộc vào màu sắc, nếu bạn không có khả năng nhận diện màu sắc thì nên lấy một vật nhọn khắc lên đầu của chân cắm để đánh dấu.

Mắc dây rất đơn giản:

- **GND** vào GND của Arduino
- **5V** vào 5V của Arduino
- **Control** vào chân PWM (trong bài sử dụng chân 9)

Code thử

```
#include <Servo.h>

Servo myservo;

const int ServoPin = 9;
void setup() {
  myservo.attach(ServoPin);
}
int pos = 0;

void loop() {
  for (pos = 0; pos <= 180; pos += 1) {
    myservo.write(pos);
    delay(15);
  }
  for (pos = 180; pos >= 0; pos -= 1) {
    myservo.write(pos);
    delay(15);
  }
}
```

Chú thích một vài điều quan trọng

- Hàm `.attach(ServoPin)` là hàm liên kết chân trên Arduino với Servo. Hàm có tác dụng khởi tạo cho chân trên Arduino giao tiếp với Servo. Cắm Control của Servo vào chân nào trên Arduino thì attach chân đó.
- Hàm `.write(pos)` với pos là một số **từ 0 đến 180**. Khi gọi hàm này, servo sẽ quay đến vị trí có góc là pos. Lưu ý là nó **QUAY ĐẾN** vị trí có **GÓC LÀ POS**.

V. Module truyền thông:

Thu phát hồng ngoại

Bộ thu phát hồng ngoại bao gồm hai thành phần:

Có dùng thư viện, không có sẵn <IRremote.h>

- LED phát hồng ngoại, màu trong, 2 chân (hình bên trái)
- LED thu hồng ngoại, màu đen, 3 chân có bọc sắc nhìn ngầu lòi (hình bên phải)

Cũng có LED thu hồng ngoại 2 chân, nhưng LED đó không có khả năng giải mã hồng ngoại. Nên nếu muốn truyền nhận thông tin bằng hồng ngoại thì phải đổi rồi.

Thư viện

- Có thể cài **IRremote** thông qua Arduino IDE:
  `Thanh Menu > Sketch > Include Library > Manage Libraries…`

Sau đó tìm **IRremote** và cài.

- Cách thứ hai là tải file ZIP sau đây về: IRremote.zip. Sau đó, vào:
  `Thanh Menu > Sketch > Include Library > Add ZIP Library`

Và chọn file zip vừa tải về.

Mắc dây

Sử dụng **một bảng Arduino** cho việc thu.
Đầu thu gồm có ba chân:

- **OUT**: vào cổng Digital bất kỳ (trong bài dùng D7)
- **GND**: vào chân đất GND
- **VCC**: vào 3.3V hoặc 5V (trong bài dùng 5V)

Chú ý thứ tự như hình. Vì GND ở giữa nên không sợ nhầm và không sợ cháy.

Sử dụng **một Arduino khác** nữa cho việc phát.
LED phát có hai chân:

- Chân dài hơn: là chân tín hiệu, mắc vào **D3**. Bắt buộc phải là **D3**, thư viện đã cố định chân phát rồi.
- Chân ngắn hơn: là chân đất, vào GND.

Code thử

Bài code này sử dụng cách mắc dây như trên. Cứ mỗi 0.5s, mạch phát sẽ gửi số 1 hoặc 0, luân phiên nhau. Còn mạch thu sẽ nhận số đó, nếu là 1 thì bật LED 13 lên, nếu là 0 thì tắt LED đi.

Code truyền (LED trắng)

```
#include <IRremote.h>
IRsend irsend;

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

int val=0;
void loop() {
  val = val ^ 1; // đổi giá trị val
  Serial.println("Sending: " + String(val) + "...");
  irsend.sendRC5(val, 8);
  // Vì mình chỉ gửi 0 hoặc 1 nên số bit tối thiểu là 1
  // Nhưng số 8 ở đây có ý nghĩa là gửi tối đa 8 bit
  // Vậy là mình gửi thừa 7 bit, nhưng không sao.
  // Bạn cũng có thể để lên tới 10, 11 bit.
  // Chỉ cần chú ý là số bit phải lớn hơn kích cỡ của dữ
  // liệu bạn muốn gửi. Thừa còn hơn thiếu.
  delay(500);
}
```

Chú thích code:

- Có nhiều hàm khác ngoài hàm `.sendRC5(v, x)` nhưng mình chỉ test và dùng hàm này. Hàm này có ý nghĩa gửi một số nguyên 32-bit **v** và gửi tối đa là **x** bit dữ liệu. Nếu bạn muốn gửi **1022** đi, bạn cần phải để **x** là **10** vì số **1022** cần tới 10 bit để biểu diễn. Lưu ý điều này nếu bạn muốn gửi đi một số là phép `analogRead()`
- Bạn có thể để **x** lên **32** luôn. Về lý thuyết là OK nhưng mình chưa test.

Code nhận (LED đen)

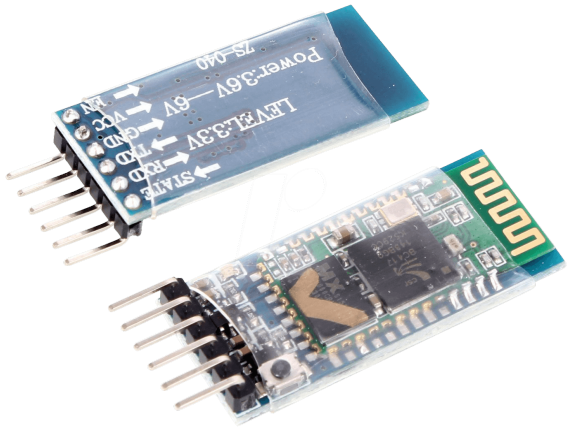
```
#include <IRremote.h>
const int rpin = 7;

IRrecv irrecv(rpin);
decode_results rs;

void setup() {
  irrecv.enableIRIn();
  Serial.begin(9600);
  pinMode(13, OUTPUT);
}

void loop() {
  if (irrecv.decode(&rs)) {     // nếu có mã và decode, hàm này trả 1
    int value = rs.value;       // lấy giá trị của mã
    Serial.println(value);
    digitalWrite(13, value);
    irrecv.resume();            // phải Resume sau khi decode
  }
  delay(1);
}
```

Chú thích code:

- Hàm `.decode(&rs)` sẽ lấy một tham chiếu của đối tượng **decode_results** có định nghĩa trong thư viện **IRremote** và viết kết quả lên đó. Nếu thật sự có dữ liệu đang chờ và đã decode thành công thì hàm này sẽ trả về 1, cho phép ta xử lý thông tin mới được decode.
- Ta có thể lấy giá trị nguyên 32bit đã decode bằng cách truy cập **.value** của đối tượng **decode_results** đó.
- Hàm `.resume()` là bắt buộc bởi vì sau khi decode, việc decode sẽ bị gián đoạn. Ta phải gọi **resume** để tiếp tục decode, nếu không thì nó sẽ không làm nữa.

Bluetooth (HC-05)

Có dùng thư viện, có sẵn trong Arduino IDE <SoftwareSerial.h>



Tên model: HC-05

Sáu chân:

- **VCC (5V), GND**
- **TXD, RXD** (hai chân Digital) Transmit và Receive
- **EN** (hay **KEY**) chân Digital, *nếu là **HIGH** trước khi có nguồn điện vào module sẽ để module ở mode AT Command* để thay đổi setting, còn lại thì luôn ở Data mode.
- **STATE**, chân Digital, nối trực tiếp với LED trên bo, có thể đọc chân này để lấy trạng thái hoạt động của module

LED:

- Nếu nháy mỗi 2s một cái, module đang ở Command mode
- Nếu nháy liên tục, module đang ở Data mode và đang chờ bắt cặp
- Nếu nháy đúp mỗi vài giây, module đã bắt cặp thành công ở Data mode.

Tips:

- Gắn **EN** vào 3.3V trước rồi mới cho điện vào **HC05** (trước khi gắn **VCC** hoặc **GND**) thì sẽ để **HC05** ở chế độ AT Command (baud rate 38400). Làm cách này có thể nhanh chóng đổi mode chỉ với việc rút dây mà không cần upload code hay nhấn giữ nút. (Tuy nhiên mình không biết nó có tai hại gì không khi gắn nó vào 5V)
- Nếu Serial Monitor in ra nhiều ký tự kỳ cục, hãy đảm bảo baud rate trùng với baud rate đã set, và lựa chọn ký tự xuống dòng là `Both NL & CR`.

Nhảy nhanh xuống QUY TRÌNH THIẾT LẬP nếu bạn muốn

Ở Command mode

Tập lệnh AT:

`AT` Lệnh test, nó sẽ trả về OK nếu module đã hoạt động ở Command Mode

`AT+RESET` `AT+ORGL` Reset module, khôi phục module về setting mặc định

`AT+VERSION?` trả về firmware hiện tại của module

`AT+ROLE?` trả về Role hiện tại của module (0=Slave, 1=Master, 2=Slave-loop)

`AT+ROLE=X` set Role là `X` (0=Slave, 1=Master, 2=Slave-loop)

`AT+PSWD?` `AT+PSWD=6969` xem Password (mặc định **1234**), đổi password sang **6969**

`AT+NAME?` `AT+PSWD=Blue` xem Name (mặc định **HC-05**), đổi name sang **Blue**

`AT+ADDR?` Xem địa chỉ của thiết bị hiện tại

`AT+UART?` `AT+UART=9600,0,0` xem cấu hình UART, set thiết lập baudrate 9600,1 bit stop, no parity

`AT+RMAAD` Ngắt kết nối với các thiết bị đã ghép

`AT+CMODE=0` Cho phép kết nối với bất kì địa chỉ nào

`AT+INQ` Bắt đầu tìm kiếm thiết bị để ghép nối

Sau khi chạy hai dòng trên, ta thu được các thiết bị ở dạng:

`+INQ:address,type,signal,(name?)`

Phần địa chỉ (address) sẽ có định dạng như sau: **0123:45:6789**. Để sử dụng địa chỉ này trong các lệnh tiếp theo ta phải thay dấu ":" thành ",":

0123:45:6789 -> **0123,45,6789**

`AT+PAIR=<address>,<timeout>` : Đặt timeout(s) khi kết nối với 1 địa chỉ slave

VD: `AT+PAIR=0123,45,6789,5` thử pair với địa chỉ **0123:45:6789**, nếu không thành công sau khoảng 5s, ngừng thử.

`AT+LINK=<address>` Chủ động kết nối với slave ở `<address>`.

`AT+BIND=<address>` Nhớ slave ở `<address>` và sẽ luôn thử kết nối với nó mỗi lần khởi động.

Ở Data mode

Ngoài việc sử dụng chân RX, TX của UNO, một cách phổ biến hơn là **HC-05** sử dụng đối tượng `SoftwareSerial` để truyền thông dữ liệu. Của đối tượng `SoftwareSerial`, ta truyền dữ liệu sử dụng `write()` để gửi dữ liệu đi; đọc dữ liệu sử dụng `read()`; và `available()` sẽ cho ta biết nếu có dữ liệu cần đọc.

```
#include <SoftwareSerial.h>

SoftwareSerial BTSerial(10, 11);
void setup() {
  Serial.begin(9600);  BTSerial.begin(9600);
}

void loop() {
  if (BTSerial.available())
    Serial.write(BTSerial.read());
  if (Serial.available())
    BTSerial.write(Serial.read());
}
```

Cùng với một code bên trái, cả hai thiết bị đã bắt cặp và kết nối có thể giao tiếp với nhau thông qua `BTSerial.write()` và `BTSerial.read()`.

Ở 1 thiết bị, bạn có thể `BTSerial.write('1');` và ở thiết bị còn lại, bạn kiểm tra `BTSerial.read() == '1'`. Nếu đúng thì bật đèn, bật quạt hay làm bất cứ thứ gì bạn muốn.

CÁC BƯỚC THIẾT LẬP

Mắc dây HC-05

- **VCC** vào **5V**, **GND** vào ground
- **RXD** vào chân mà mình chọn làm chân **TX** (trong bài sử dụng 11), **TXD** vào chân mà mình chọn làm chân **RX** (trong bài sử dụng 10)
- **EN**: Nếu muốn vào Command Mode, cắm **EN** vào **3.3V** trước khi cắm nguồn vào HC05. Sau đó, khi cắm nguồn, nó sẽ vào Command Mode. Nếu muốn vào Data mode, chỉ cần không cắm **EN** mà cứ cắm nguồn là xong. Kiểm tra cách mà LED nháy (đã nói ở trên) để đảm bảo đã vào đúng mode.
- Không dùng **State**.

Thiết lập AT Command

- Sử dụng hai Arduino Uno, hai **HC-05**. Mỗi HC05 cắm vào một Uno như trên, tất cả đều để ở Command Mode.
- Nạp code sau đây vào cả hai Arduino. Code này có tác dụng cho phép nhập lệnh AT nếu cả hai đã ở Command Mode:

```
#include <SoftwareSerial.h>
SoftwareSerial BTSerial(10, 11); // (Rx, Tx)

void setup() {
  Serial.begin(9600);
  Serial.println("Enter AT Commands:");
  BTSerial.begin(38400);
}

void loop() {
  if (BTSerial.available())
    Serial.write(BTSerial.read());
  if (Serial.available())
    BTSerial.write(Serial.read());
}
```

- Mở Serial Monitor lên, đảm bảo BaudRate 9600, Both NL & CR.
- Kiểm tra **Thanh menu > Tools > Port > COM..**, nếu cắm 2 Uno thì sẽ có 2 COM. Ở mỗi COM, bật Monitor và thử lệnh **AT**. Nếu ở cả hai đều có trả về 0K thì đã thành công.

1. Chọn một Uno+HC05 để làm Slave. Tại nó, hãy làm như sau:

- Mở monitor, nhập lệnh sau:

    `AT`

    `AT+ROLE=0`

    `AT+NAME=`Ten_Cua_Ban**`-HC05-S`**

    `AT+CMODE=0`

- Sau đó, hãy chuyển Slave về Data mode.

2. Chuyển qua Uno+HC05 còn lại (Master). Tại nó, hãy làm như sau:

- Mở monitor, nhập lệnh sau:

    `AT`

    `AT+ROLE=1`

    `AT+NAME=`Ten_Cua_Ban**`-HC05-M`**

    `AT+CMODE=0`

    `AT+INQ`

- Sau đó chờ một lát để HC05 Master quét các thiết bị Bluetooth lân cận. Khi thấy nó hoàn tất và trả về OK, thì màn hình của bạn sẽ như sau:

    ```
    +INQ:2:72:D2224,3E0104,FFBC,Bluetooth Speaker
    +INQ:1111:22:333333,1F1F,FFC1,Ten_Cua_Ban-HC05-S
    +INQ:4444:55:6666,1F1F,FFC0,Bluetooth ABC
    OK
    ```

- Nhìn trên màn hình trên, cụm từ cuối chính là tên của Slave của bạn. Ở dòng đó, địa chỉ của Slave cũng được gửi theo, đó là `1111:22:333333`. Nếu không thấy, hãy thử rút nguồn của Slave rồi cắm lại, và chạy **`AT+INQ`** lại nhiều lần. Nhớ phải để INQ trả về OK thì mới thử lại.
- Ví dụ với địa chỉ `1111:22:333333`, đổi dấu `:` thành dấu phẩy `,` thành `1111,22,333333`. Ta sẽ dùng cái này để kết nối đến Slave.

  Sau đó chạy các câu lệnh sau ở Master:

    `AT+PAIR=``1111,22,333333`

    `AT+LINK=``1111,22,333333`

    `AT+BIND=``1111,22,333333`

3. Sau đó, hai cho hai cái về Data Mode và khởi động lại. Lúc đầu cả hai đều nháy nhanh, sau nhiều nhất 1 phút, sẽ có lúc cả hai sẽ dừng chớp nhanh, chuyển qua mỗi vài giây nháy một lúc 2 lần. Như vậy nghĩa là cả hai đã kết nối thành công.

Thử giao tiếp HC05

Master và Slave chỉ có ý nghĩa khi bắt cặp thôi. Khi đã bắt cập thì ai cũng có thể truyền và nhận cả. Nên Receiver hay Transmitter không phụ thuộc vào là Master hay là Slave.

- Nạp code sau đây vào Master (làm chức năng Receiver):

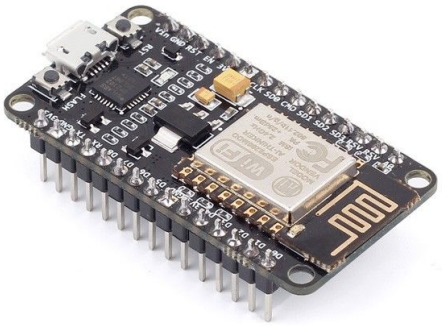
```
#include <SoftwareSerial.h>
SoftwareSerial BTSerial(10, 11);

void setup() {
  Serial.begin(9600);
  BTSerial.begin(9600);
  pinMode(13, OUTPUT);
}

void loop() {
  if (BTSerial.available()) {
    char input=BTSerial.read();
    Serial.write(input);
    if (input == '1') digitalWrite(13, HIGH);
    if (input == '0') digitalWrite(13, LOW);
  }
}
```

- Nạp code sau đây vào Slave (làm chức năng Transmitter):

```
#include <SoftwareSerial.h>
SoftwareSerial BTSerial(10, 11);

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

void loop() {
  BTSerial.write('1');
  delay(2000);
  BTSerial.write('0');
  delay(2000);
}
```

- Sau đó, nếu ở Master, cứ mỗi 2s cái LED trên bo nhấp nháy thì giao tiếp đã thành công.

Wifi (NodeMCU)



Có sử dụng thư viện, cần internet để tải vài thứ
Trong bài sử dụng NodeMCU v2. Bạn cũng có thể sử dụng v3, nó không thực sự quan trọng lắm

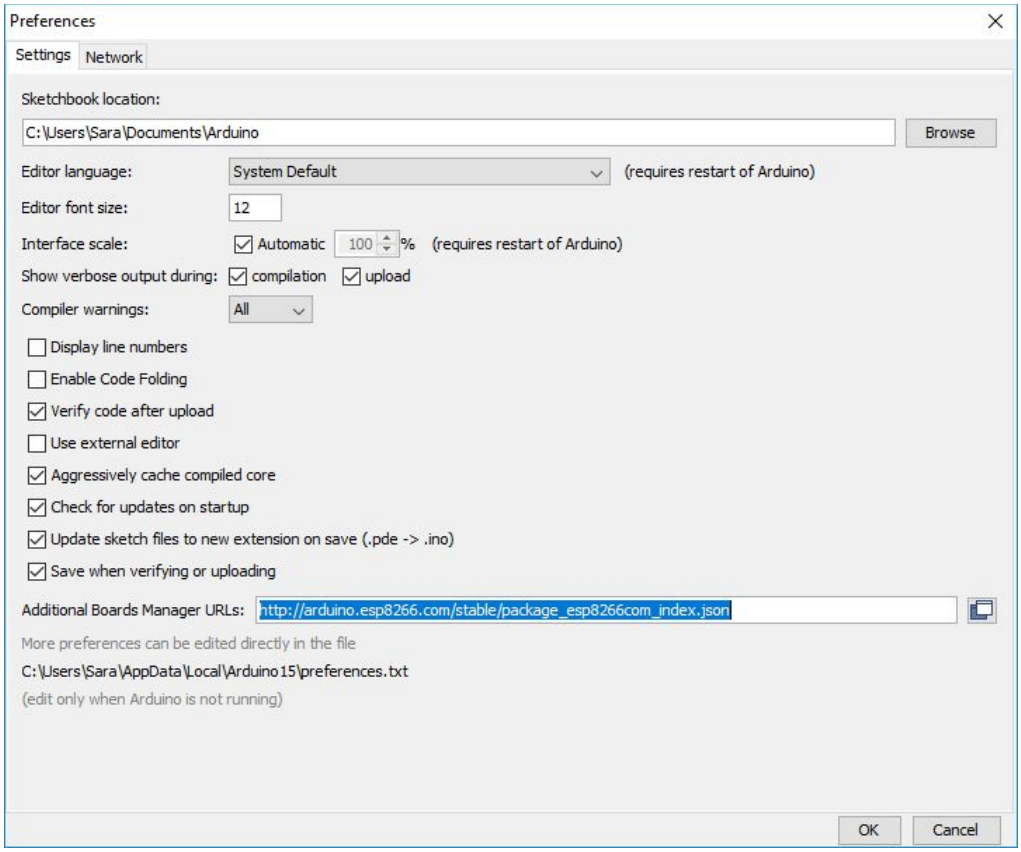
NodeMCU sử dụng Micro USB, dùng chung Arduino IDE để nạp code, tuy nhiên bạn phải thêm board ESP cho nó trước rồi mới sử dụng được.

CÁC BƯỚC THIẾT LẬP

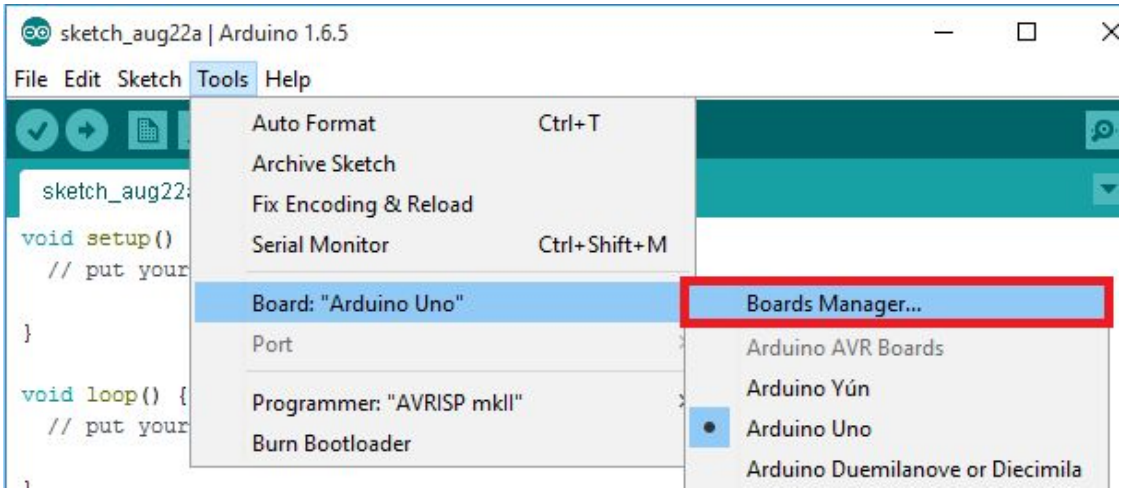
Phần này sử dụng Tutorial ở https://randomnerdtutorials.com/esp8266-web-server/

Thêm ESP8266 Add-on cho Arduino IDE

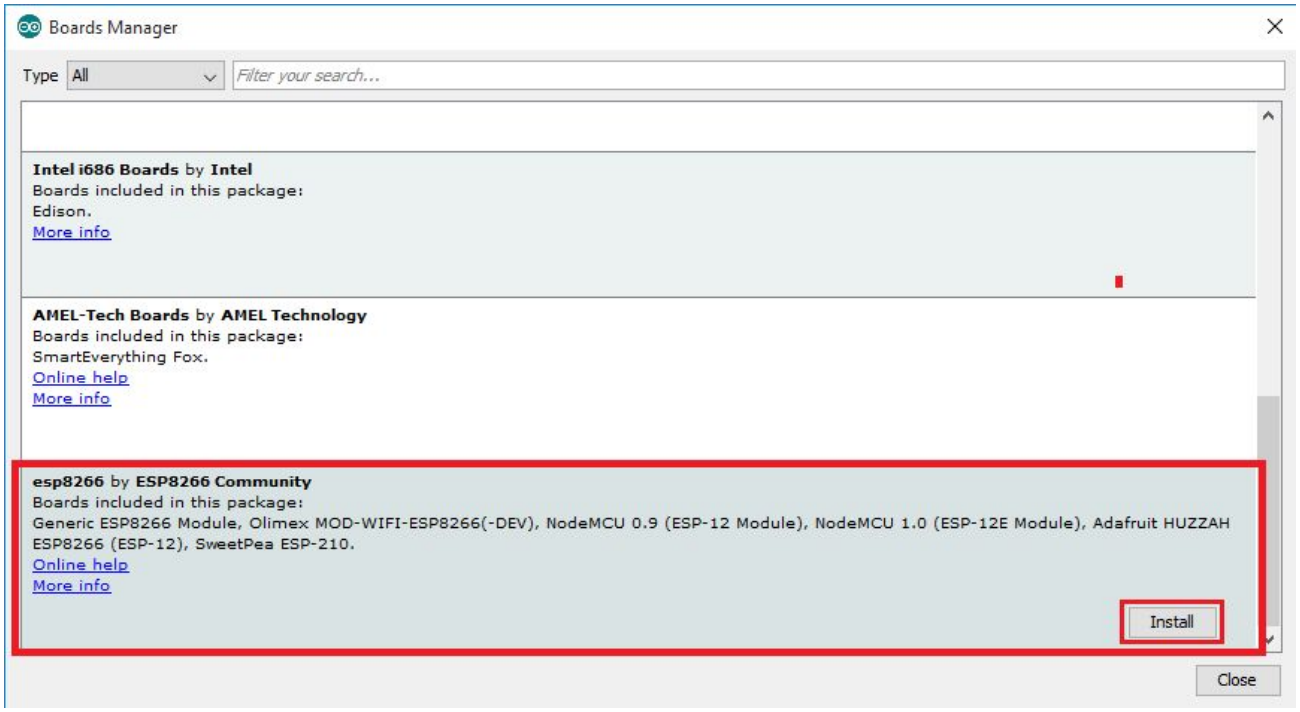
1. Mở Arduino IDE ra. Trên thanh Menu: `File > Preferences` thì nó nhảy ra một cái cửa sổ. Ở chỗ bôi xanh (**Additional Boards Manager URLs**), thêm vào dòng sau:
   `http://arduino.esp8266.com/stable/package_esp8266com_index.json`
   Và nhấn **OK**.



2. Sau đó, mở `Menu > Tools > Board > Boards Manager...`

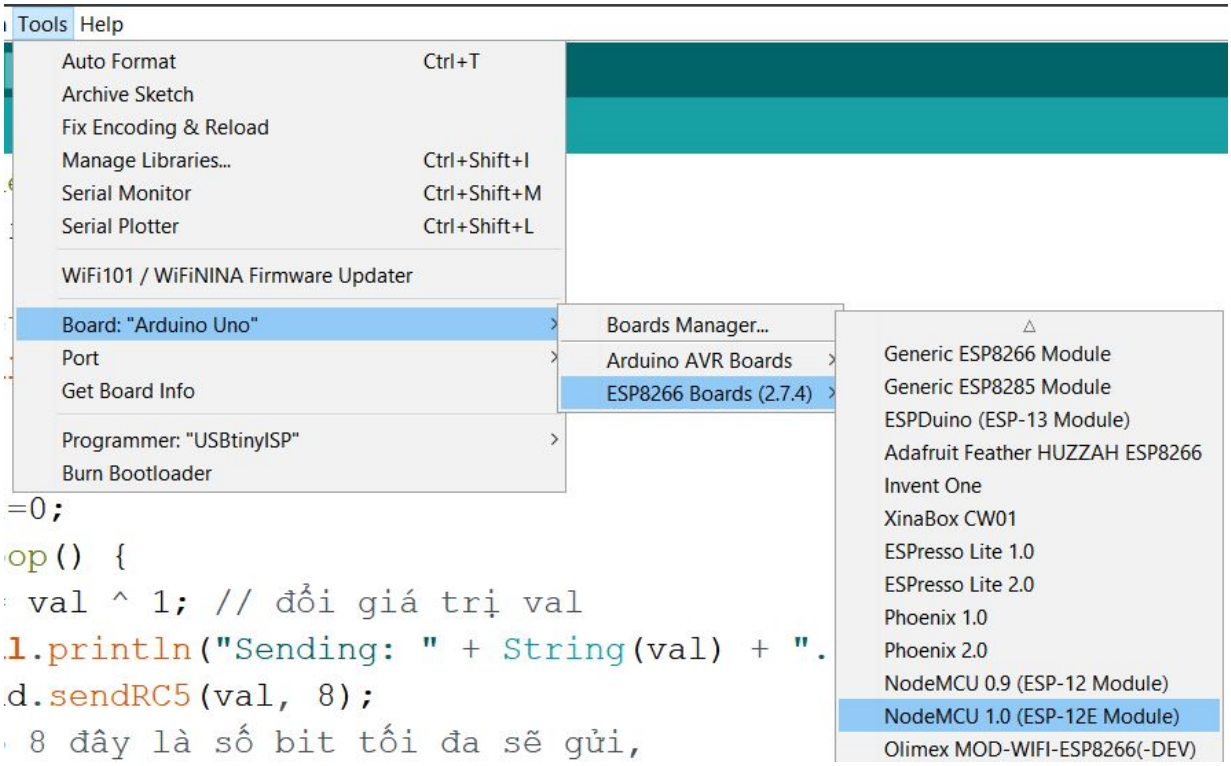


3. Ở cửa sổ mới, tìm "**esp8266**" và sau đó tìm ở cửa sổ dưới "**esp8266 by ESP8266 Community**". **Install** nó.



Sau đó, hãy tắt và bật lại **Arduino IDE**.

4. Ở Menu, chọn `Tools > Board > ESP8266 Boards (..) > Chọn board của bạn`. Vì trong bài sử dụng **NodeMCU ESP-12E** (có hình giống như tấm hình ở đầu đề) nên tôi chọn như dưới hình.



Nếu các bạn không chắc module của mình là loại nào thì có thể google tên của những board trong menu để xem hình nhé.

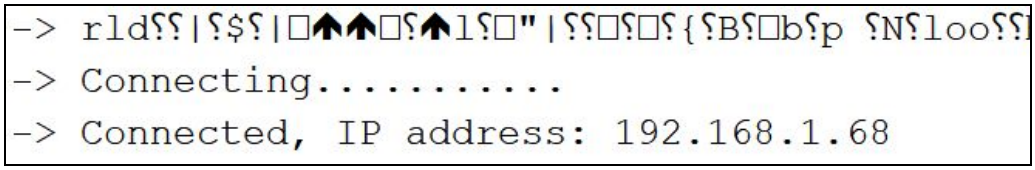
Code thử NodeMCU

NodeMCU có thể làm Access Point (phát ra WiFi) hoặc làm Station (kết nối tới WiFi). Dưới đây là Code thử cho NodeMCU kết nối tới một WiFi mà bạn muốn. Thay **tên-wifi-của-bạn** và **mật-khẩu-wifi-của-bạn** theo wifi mà bạn muốn nó kết nối, sau đó hãy nạp code.

```
#include <ESP8266WiFi.h>

#define TENWIFI "tên-wifi-của-bạn"
#define PASSWIFI "mật-khẩu-wifi-của-bạn"

void setup() {
  Serial.begin(115200);
  WiFi.begin(TENWIFI, PASSWIFI);
  // Kết nối WIFI với TÊN và MẬT KHẨU như trên
  Serial.print("\n Đang thử kết nối");
  while (WiFi.status() != WL_CONNECTED) {
    // làm cho đến khi thành công
    delay(500); Serial.print(".");
  }
  Serial.print("\n Kết nối thành công. IP address: ");
  Serial.println(WiFi.localIP());
}

void loop() {}
```

Sau một lúc, nếu có thông báo:

Thì bạn đã nạp code thành công. Bật **Serial Monitor** lên, nhớ Baud Rate là **115200**. Sau đó, nhấn nút reset có trên node MCU.
Nút đó là nút **RST**, ở bên cạnh cổng Micro USB.



*Đã kết nối thành công và NodeMCU đang ở IP cục bộ là **192.168.1.68***

Làm Web Server đơn giản (tự động refresh)

Sau khi đã kết nối được với WiFi, NodeMCU có thể thiết lập một WebServer tại IP của nó. Bài code này sẽ trả về một trang web HTML gồm một câu chào và cho biết đã bao nhiêu thời gian kể từ khi server được khởi động.
Những dòng code cũ ở phần trên sẽ được in nghiêng và màu nhạt hơn chút.

```
// Đây là HTTP Header, thông thường nên để nó như vậy
String httpHeader = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n";

// Đây là code HTML sẽ gửi cho truy cập, chỉnh sửa code HTML dưới đây
// để thay đổi nội dung trang web
String html = R"=====(
<!DOCTYPE html>
<html>
 <head>
 <meta name='viewport' content='width=device-width, initial-scale=1.0'/>
 <meta charset='utf-8'>
 <meta http-equiv='refresh' content='5'>
 <style>
   h2 {text-align:center; }
   p {text-align:center;}
 </style>
   <title>Web Server với NodeMCU, tự động refresh</title>
 </head>
 <body>
     <h2>Tự động update với HTML</h2>
     <div id='count'>
       <p>Thời gian từ khi server khởi động: %time% giây</p>
     </div>
 </body>
</html>
)=====";
// Sử dụng R"=====( cho phép ta tạo string trên nhiều dòng
```

```
#include <ESP8266WiFi.h>

#define TENWIFI "tên-wifi-của-bạn"
#define PASSWIFI "mật-khẩu-wifi-của-bạn"

WiFiServer server(80);

void setup(){
  Serial.begin(115200);
  WiFi.begin(TENWIFI, PASSWIFI);

  Serial.print("\n Đang thử kết nối");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500); Serial.print(".");
  }
  Serial.print("\n Kết nối thành công. IP address: ");
  Serial.println(WiFi.localIP());

  server.begin(); // Khởi động server.
  Serial.print("Server đã khởi động.");
}

void loop() {
    // Kiểm tra nếu có client,
    WiFiClient client = server.available();
    if (!client)  { return; }
    // client == 0 khi không có người truy cập

    String htmlPage = html;
    htmlPage.replace("%time%", String(millis()/1000));
    // thay thế chuỗi '%time%' bằng thời gian đã trôi qua

    client.flush();          // bỏ qua thông tin client gửi
    client.print(httpHeader);// viết HTTP Header ra trình duyệt trước
    client.print(htmlPage);  // viết code HTML ra trình duyệt

    delay(5); // chờ một chút để trình duyệt xử lý
    // Khi hàm loop kết thúc
    // đối tượng client sẽ tự bị hủy và tự disconnect
}
```

Sau khi nạp, kết quả bạn nhận được có sẽ như sau:

Sẽ mỗi 5s trang web sẽ được tự động cập nhập lại.

Như vậy, chỉ với một vài hàm đơn giản trong **ESP8266WiFi.h** kết hợp với xử lý **String**, chúng ta đã tạo được một trang web có khả năng tự động update. Các bạn có thể thay vì **millis()** mà sử dụng **analogRead()** hay **digitalRead()** hay xử lý riêng trường hợp để có thể update thời gian thực.

Làm Web Server đơn giản (bật/tắt đèn)

```
// Đây là HTTP Header, thông thường nên để nó như vậy
String httpHeader = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n";

// Đây là code HTML sẽ gửi cho truy cập, chỉnh sửa code HTML dưới đây
// để thay đổi nội dung trang web
String html = R"=====(
<!DOCTYPE html>
<html>
 <head>
 <meta name='viewport' content='width=device-width, initial-scale=1.0'/>
 <meta charset='utf-8'>
 <style>
   h2, h3, p {text-align:center; }
 </style>
   <title>Bật tắt đèn với NodeMCU</title>
 </head>
 <body>
     <h2>Bật tắt đèn với NodeMCU</h2>
     <div>
        <h3> LED D1 </p>
        <p>%button4%</p>
        <h3> LED D2 </p>
        <p>%button5%</p>
     </div>
 </body>
</html>
)=====";

#include <ESP8266WiFi.h>

#define TENWIFI "tên-wifi-của-bạn"
#define PASSWIFI "mật-khẩu-wifi-của-bạn"

WiFiServer server(80);
```

```
void setup(){
  Serial.begin(115200);
  WiFi.begin(TENWIFI, PASSWIFI);

  Serial.print("\n Đang thử kết nối");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500); Serial.print(".");
  }
  Serial.print("\n Kết nối thành công. IP address: ");
  Serial.println(WiFi.localIP());

  server.begin(); // Khởi động server.
  Serial.print("Server đã khởi động.");
  // -- khởi tạo LED
  pinMode(D1, OUTPUT); pinMode(D2, OUTPUT);
  digitalWrite(D1, LOW); digitalWrite(D2, LOW);
}

// biến tạm lưu trạng thái LEDs
int led4on = LOW, led5on = LOW;

void loop() {
    // Kiểm tra nếu có client,
    WiFiClient client = server.available();

    // client == 0 khi không có người truy cập
    if (!client)  { return; }

    // lúc này, thông tin client gửi là quan trọng, nên chúng ta sẽ đọc nó
    String request = "";
    while (client.connected()) {
      if (client.available())
        request += (char) client.read(); // ép kiểu byte sang char
      if (request.endsWith("\r\n\r\n")) // header được quy định là kết thúc
        break;            // bằng hai dòng trống, ta dùng điều này để break
    }
    Serial.println("\r\n====== START REQUEST HEADER =====");
    Serial.println(request);
    Serial.println("====== END REQUEST HEADER =====");
```

```
    // Xử lý sự kiện dựa trên URL của request:
    // Nếu truy cập vào 192.168.x.y/abc/def thì trong request
    // sẽ xuất hiện chuỗi ký tự "GET /abc/def", ta có thể dùng
    // nó như một sự kiện để xử lý.

    // nếu trong request tồn tại chuỗi "GET..."
    if (request.indexOf("GET /4/on") >= 0) led4on = HIGH;
    if (request.indexOf("GET /4/off") >= 0) led4on = LOW;
    if (request.indexOf("GET /5/on") >= 0) led5on = HIGH;
    if (request.indexOf("GET /5/off") >= 0) led5on = LOW;

    // In ra lại sau khi đã đổi
    Serial.println("LED 4: " + String(led4on));
    Serial.println("LED 5: " + String(led5on));

    // bật/tắt đèn theo trạng thái
    digitalWrite(D1, led4on); digitalWrite(D2, led5on);

    // thay thế nút button theo trạng thái đèn
    String htmlPage = html;
    if (led4on == LOW) htmlPage.replace("%button4%", "<a
href=\"/4/on\"><button>BẬT</button></a>");
    else htmlPage.replace("%button4%", "<a
href=\"/4/off\"><button>TẮT</button></a>");
    if (led5on == LOW) htmlPage.replace("%button5%", "<a
href=\"/5/on/\"><button>BẬT</button></a>");
    else htmlPage.replace("%button5%", "<a
href=\"/5/off/\"><button>TẮT</button></a>");

    client.print(httpHeader); // in HTTP Header trước
    client.print(htmlPage);  // in code HTML
    delay(5); // chờ một chút để trình duyệt xử lý
    // Khi hàm loop kết thúc, đối tượng client sẽ tự bị hủy và tự disconnect
}
```

Giao tiếp từ NodeMCU đến Arduino

Ta sẽ sử dụng giao tiếp UART. Đây là giao tiếp mà cổng Serial hay **SoftwareSerial** sử dụng. Vì vậy, ta cần dùng hai chân trên NodeMCU để làm Rx và Tx, và hai chân bên Arduino để làm Rx và Tx.

Trong bài sử dụng:

- **RX**: D1 của NodeMCU, D10 của Arduino
- **TX**: D2 của NodeMCU, D11 của Arduino

Lưu ý, giao tiếp UART cần mắc dây ngược, cho nên:

- **NodeMCU D1** vào **Uno D11** (Rx vào Tx)
- **NodeMCU D2** vào **Uno D10** (Tx vào Rx)

Bài code này sẽ tạo một Web Server trên NodeMCU, trên đó có một form cho phép nhập chân Uno và nhập giá trị muốn viết. Sau khi nhập và nhấn nút, NodeMCU sẽ xử lý **String** và gửi thông tin đó cho Arduino dưới dạng **String**. Bên Arduino sẽ nhận dữ liệu và **digitalWrite** hay **analogWrite** tương ứng theo giá trị 0, 1 hoặc lớn hơn.

Code NodeMCU

```
String httpHeader = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n";
String html = R"=====(
<!DOCTYPE html>
<html>
 <head>
 <meta name='viewport' content='width=device-width, initial-scale=1.0'/>
 <meta charset='utf-8'>
 <style>
   h2, h3, p, form {text-align: center; }
   td { text-align: right; padding: 3px; }
   th { text-align: center; padding: 5px; }
 </style>
   <title>Viết ra chân Arduino thông qua NodeMCU</title>
   <script>
      function SendGetRequest() {
        var pin = document.forms['writePinForm'].pinName.value;
        var val = document.forms['writePinForm'].pinValue.value;
         window.location.href = window.location.origin + '/' + pin + '/' + val
+ '/';
      }
   </script>
 </head>
```

```
 <body>
     <form name='writePinForm'>
          <table width="30%" height="40%" cellpadding style="border: 1px solid
black; margin: auto; text-align: center;">
            <tr> <th colspan='3'> Viết ra chân Arduino thông qua NodeMCU </th>
</tr>
           <tr>
             <td> <label for='pinName'>Chân</label> </td>
                <td colspan='2'> <input type='text' id='pinName' name='pinName'
value='D13'> </td>
           <tr>
             <td> <label for='pinValue'>Giá trị</label> </td>
              <td colspan='2'> <input type='text' id='pinValue' name='pinValue'
value='1'> </td>
           </tr>

           <tr><td colspan='3' style='text-align: center;'>
                 <input type='button' onclick='SendGetRequest()' value='Viết ra
Arduino'>
           </td></tr>
           <tr><td colspan='3' style='text-align: center;'>
                Nhập ở ô Chân là tên của chân Arduino (D11, D5,... chân A không
output được).
                 Nhập ở ô value là một giá trị số nguyên, đừng nhập HIGH, LOW.
Nếu giá trị là 0, 1 thì hàm digitalWrite
               sẽ được sử dụng, ngược lại hàm analogWrite sẽ được sử dụng.
            </td></tr>
         </table>
     </form>
 </body>
</html>
)=====";
```

```
#include <ESP8266WiFi.h>
#define TENWIFI "tên-wifi-của-bạn"
#define PASSWIFI "mật-khẩu-wifi-của-bạn"
WiFiServer server(80);
// --
#include <SoftwareSerial.h>
SoftwareSerial ss (D1, D2);

void setup(){
  Serial.begin(115200);
  WiFi.begin(TENWIFI, PASSWIFI);

  Serial.print("\n Đang thử kết nối");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500); Serial.print(".");
  }
  Serial.print("\n Kết nối thành công. IP address: ");
  Serial.println(WiFi.localIP());

  server.begin(); // Khởi động server.
  Serial.print("Server đã khởi động.");
  // --
  ss.begin(9600);
}

void loop() {
    // Kiểm tra nếu có client,
    WiFiClient client = server.available();
    // client == 0 khi không có người truy cập
    if (!client)  { return; }

    String request = "";
    while (client.connected()) {
      if (client.available())
        request += (char) client.read(); // ép kiểu byte sang char
      if (request.endsWith("\r\n\r\n")) // header được quy định là kết thúc
        break;            // bằng hai dòng trống, ta dùng điều này để break
    }
    Serial.println(request);
    int idx1 = request.indexOf("GET") + 4;
    int idx2 = request.indexOf(' ', idx1);
    if (idx1 + 1 < idx2) {
```

```
      String towrite = request.substring(idx1+1, idx2-1);
      String pin = towrite.substring(0, towrite.indexOf('/'));
      String val = towrite.substring(towrite.indexOf('/')+1);
      String sending = pin + " " + val + "\r\n";
      Serial.println("Writing to SSerial..: " + sending);
      ss.write(sending.c_str());
    }
    String htmlPage = html;
    client.print(httpHeader); // viết HTTP Header ra trình duyệt trước
    client.print(htmlPage);   // viết code HTML ra trình duyệt
    delay(5); // chờ một chút để trình duyệt xử lý
}
```

Code Arduino

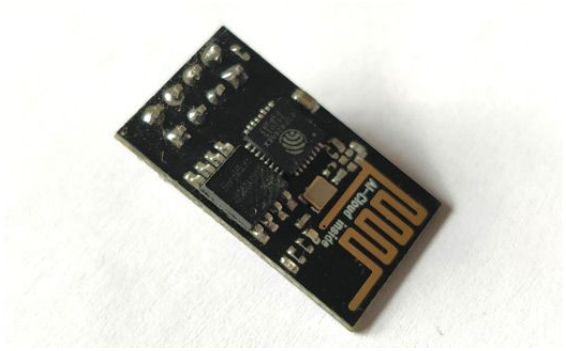
```
#include <SoftwareSerial.h>

SoftwareSerial ss (10, 11);

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

uint8_t getPin(String pinName) {
  return pinName.substring(1).toInt();
}

void loop() {
  String data = "";
  while (ss.available()) {
    data += ((char)ss.read());
    if (data.endsWith("\r\n")) break;
  }
  if (data != "") {
    Serial.println(data);
    String pinName = data.substring(0, data.indexOf(' '));
    int val = data.substring(data.indexOf(' ')+1,
data.indexOf("\r\n")).toInt();

    pinMode(getPin(pinName), OUTPUT); // set pin ở OUTPUT trước khi viết
    if (val > 1) analogWrite(getPin(pinName), val);
    else digitalWrite(getPin(pinName), val);

    ss.flush();
  }
  delay(10);
}
```

Chú thích

- Ở hai code đều có đối tượng SoftwareSerial với các chân mắc như đã trình bày. Tuy nhiên đây chỉ là giao tiếp một chiều. Muốn giao tiếp hai chiều, các bạn phải gửi String ngược lại và xử lý.
- Bạn có thể nghiên cứu và học thư viện **ArduinoJson** để gửi dữ liệu qua Serial. Học chỉ 30p nhưng sau đó code rất tiện, nhanh và không mắc lỗi truyền thông.

Wifi (ESP8266-01, không phải NodeMCU)

KHÔNG UPDATE THÊM NỮA



Có dùng thư viện, có sẵn trong Arduino IDE
Tên model: ESP8266-ESP01
Tám chân:

- **VCC** (**3.3V**), **GND**
- **RX, TX**
- **CH_EN**
- **RESET, GPIO-0, GPIO-2**

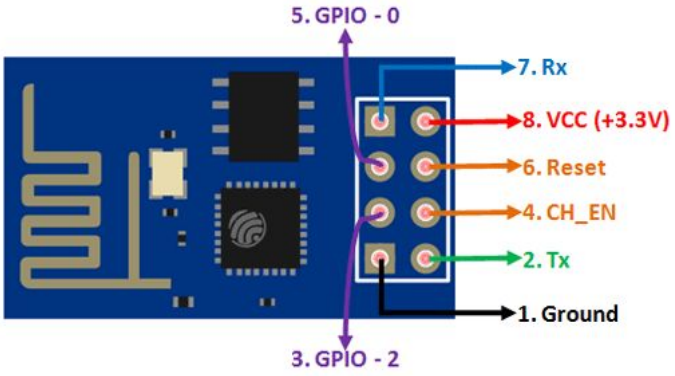
Vấn đề của ESP8266

- Nhiều tutorial bảo sử dụng hai chân **RX TX của Uno** (**pin 0, pin 1**) nhưng sẽ rất phiền phức vì khi nạp code board khi phải để hai chân đó trống thì code mới nạp được. **Nên cách tốt nhất là sử dụng 2 chân khác kèm với `SoftwareSerial`.**
- Theo thực nghiệm, ESP8266 có vấn đề về Baud Rate mặc định của nó, khiến xảy ra khó khăn khi sử dụng tập lệnh AT để thiết lập nó.

CÁC BƯỚC THIẾT LẬP

Mắc dây

- **VCC** và **CH_EN** nối với nguồn **3.3V**.
- **GND** vào chân đất (GND)
- **RX** vào pin 11, **TX** vào pin 10
- **RESET, GPIO-0, GPIO-2** thì kệ nó



Thiết lập AT Command

```
#include <SoftwareSerial.h>
SoftwareSerial esp8266(10, 11);

void setup() {
  Serial.begin(19200);
  esp8266.begin(115200);
}

void loop() {
  while(esp8266.available())
     Serial.write(esp8266.read());

  while(Serial.available())
    esp8266.write(Serial.read());
}
```

Bước 1:
Đừng cắm nguồn VCC cho wifi và nạp code này vào Arduino. (Số 115200 là baud rate **mặc định của ESP8266-01**)

Bước 2:
Mở COM Monitor lên, đảm bảo:
`[Both NL & CR]` và `[19200 baud]`
Ở thanh dưới của monitor.

Bước 3:
Cắm nguồn **VCC** của wifi vào, lúc này Monitor sẽ chạy những ký tự tầm bậy (garbage value), nhưng không sao.
Thử nhập "`AT`" và Enter vài lần, bạn sẽ đôi khi nhận được "**OK**" nhưng đôi khi nhận "**OK**" nhưng hơi bị lỗi ký tự..

Lỗi này là do Baud Rate.

Ít ra ESP của chúng ta cũng đã nhận lệnh. Đó là điều tốt.

```
sll§§s□§c□§o🡅🡅🡅§□§l§□p§<§§§§p□§§§§§§□p🡅§p §o§loo§□c🡅
23:06:29.404 -> reAT
23:06:34.548 -> □□OK
23:15:47.850 -> AT
23:15:47.850 -> □□OK
23:15:50.235 -> AT
23:15:50.235 ->
23:15:50.235 -> OK
23:15:51.826 -> AT
23:15:51.826 ->
23:15:51.826 -> OI□
23:15:53.830 -> AT
23:15:53.830 -> OK
23:15:55.745 -> AT
23:15:55.745 ->
23:15:55.745 -> OI□
```

Bước 4:
Vì lỗi do Baud Rate, ta sẽ đổi nó bằng câu lệnh AT sau: `AT+UART_DEF=19200,8,1,0,0`
Câu lệnh trên sẽ đổi Baud rate của ESP về **19200**. Sau khi Enter và nhận kết quả trả về, nếu bạn gõ "`AT`" và ENTER nhưng không thấy gì nữa thì có khả năng là đã thành công.

Bước 5:

- Rút dây **VCC** của ESP, tắt COM, quay sang code và sửa `esp8266.begin(19200);` lại Baud rate về số mà ta vừa chỉnh.
- Upload code lên lại, mở COM, và cắm VCC vào lại.

```
ʕ□□bBʕʕ8kʕ7ʕRʕXBB□P□ʕ<ʕʕ8UʕmH□b7J□sk□ʕʕ8PʕkI□b7□
ʕʕJ□ʕCʕ□ 'ʕGJz(ʕCB0b□ʕ
ready
```

Lúc này, "đống rác" ở hai dòng đầu là thông tin mặc định mà ESP sẽ "ói" ra và chúng ta không thể làm gì nó được. Quan trọng là ở dòng thứ 3 nếu có từ "**ready**" thì bạn đã thành công rồi.

Test vài lệnh:

```
AT

OK
AT+CWMODE=1

OK
AT+CWLAP
+CWLAP:(4,"[redacted]",-58,"a8:58:40[redacted]",1,-12,0)
+CWLAP:(4,"B[redacted]",-85,"d0:96:fb[redacted]",4,-22,
+CWLAP:(4,"T[redacted]",-83,"ec:84:b4:[redacted]",6,-1
+CWLAP:(4,"H[redacted]",-84,"78:44:76:[redacted]",6,-9,0)
+CWLAP:(3,"N[redacted]",-76,"a0:65:18:[redacted]
+CWLAP:(3,"T[redacted]",-85,"c0:25:e9:[redacted]",10
+CWLAP:(4,"D[redacted]",-75,"f4:28:54:[redacted]",11,-22,0
+CWLAP:(4,"D[redacted]",-82,"a4:f4:c2:[redacted]",11,0,0)
+CWLAP:(4,"H[redacted]",-86,"84:16:f9:[redacted]",11,-7,0

OK
```

Lệnh test AT -> set mode là kết nối tới wifi -> scan wifi lân cận

VI. Đề làm thử:

Hồng ngoại, Bluetooth

Các đề gồm có hai mạch arduino: mạch điều khiển và mạch xử lý. Hai mạch giao tiếp qua Hồng ngoại hoặc Bluetooth.

Đề 1:

Làm hệ thống mô phỏng điều khiển đi tới đi lui, kèm chức năng chống va chạm. Hệ thống có hai mạch, gồm xử lý và điều khiển:

1. Mạch xử lý gồm có:

- LED 1, 2, 3
- 1 Động cơ DC
- 1 Buzzer
- 1 Biến trở

2. Mạch điều khiển gồm có:

- Một nút bấm A
- Một nút bấm B
- Một sensor siêu âm SN

Các chức năng:

- Ban đầu LED 1, 2, 3 sẽ tắt, buzzer không kêu, Stepper không quay.
- Khi nhấn nút A ở bên mạch điều khiển, LED 1, 2 bên mạch xử lý sẽ nháy luân phiên nhau (như đèn của xe cảnh sát), và DC sẽ quay theo một chiều.
- Khi bấm nút B, DC sẽ đổi chiều.
- Biến trở sẽ điều khiển tốc độ của động cơ DC.
- Nếu Sensor SN khi tìm thấy vật thể ở khoảng cách trong 15cm, sẽ làm sáng LED 3. Trong lúc này, nếu nút A được bấm thì Buzzer sẽ vang lên, không cho DC quay, tuy nhiên LED 1, 2 sẽ vẫn nhấp nháy.

Đề 2:

Làm hệ thống mô phỏng bếp lửa nấu đồ, kèm chức năng nếu có khói thì sẽ tắt bếp và bật quạt lên. Hệ thống gồm hai mạch, xử lý và điều khiển:

1. Mạch xử lý gồm có:

- 5 LED: 1,2,3,4,5 xếp thẳng hàng trên board
- 1 DC Motor
- 1 Buzzer

2. Mạch điều khiển gồm có:

- Một biến trở B làm núm vặn
- Một switch S làm công tắc
- Một sensor MQ-2 làm cảm biến Gas

Các chức năng:

- Ban đầu LED 1,2,3,4,5 tắt. DC Motor không quay. Buzzer không kêu.
- Nếu công tắc S bị tắt, không có LED nào sáng.
- Nếu công tắc S được bật lên, LED 1 sẽ sáng lên, và LED 2,3,4,5 sẽ sáng tùy vào giá trị của núm vặn B:
    - Giá trị >= 250, sáng LED 2
    - Giá trị >= 500, sáng LED 2,3
    - Giá trị >= 750, sáng LED 2,3,4
    - Giá trị >= 1000, sáng LED 2,3,4,5
- Nếu MQ-2 phát hiện có khói (A0 lớn hơn bao nhiêu tự chọn), Buzzer sẽ kêu tít tít tít (không kêu liên tục mà kêu và tắt mỗi 100ms), DC Motor quay, và LED 2,4 tắt, chỉ có LED 1,3,5 sáng.

Đề 3:

Làm hệ thống mô phỏng máy chơi gắp hàng, kèm chức năng nếu có rung động thì sẽ không cho điều khiển cần cẩu gắp. Hệ thống gồm hai mạch, xử lý và điều khiển:

1. Mạch xử lý gồm có:

- Một Servo
- Một LED 1, cho biết Servo vẫn có thể được điều khiển. Cần cẩu được gọi là vẫn có thể được điều khiển khi **Nó không đang hạ xuống và nâng lên**, và **Nó không bị rung**.
- LED 2,3,4,5 tượng trưng cho việc cần cẩu hạ xuống và nâng lên.
- Một Buzzer, sẽ kêu lên nếu Servo không thể quay thêm được nữa (đã đụng biên)

2. Mạch điều khiển gồm có:

- Một module Joystick JS.
- Một module rung.

Các chức năng:

- Ban đầu LED 1 sáng. LED 2,3,4,5 tắt, và Buzzer không kêu. Servo ở vị trí mặc định là ở giữa (tự chọn pos nào là giữa, nhưng đảm bảo là servo vẫn có thể quay sang trái hoặc quay sang phải).
- Joystick JS ta sẽ bỏ qua chân Vx, chỉ đọc chân Vy và chân Sw. Nếu JS được kéo sang bên trái, Servo sẽ quay từ từ sang trái. Khi quay hết mức nhưng JS vẫn được kéo sang trái, Buzzer sẽ kêu lên. Làm tương tự với JS kéo sang bên phải.
- Nếu mạch điều khiển có rung, LED 1 tắt và mọi điều khiển của JS sẽ không có tác dụng.
- Khi Joystick JS nếu được nhấn xuống, LED 2,3,4,5 sẽ sáng như sau:

  Không sáng -> Sáng 2 -> Sáng 2,3 -> Sáng 2,3,4 ->
  Sáng 2,3,4,5 -> Sáng 2,3,4 -> Sáng 2,3 -> Sáng 2 -> Không sáng

Thời gian từ trạng thái này sang trạng thái kia là 100ms. Sau khi thực hiện xong ở trên, Servo sẽ về lại vị trí ở giữa. Trong khi đang thực hiện hành động này, cho dù có rung vẫn thực hiện, không gián đoạn.

Wifi

Đề 4:

Làm hệ thống SmartHome điều khiển qua wifi. Hệ thống sử dụng:

- 1 Arduino Uno R3
- 1 ESP8266 hoặc 1 NodeMCU (kết nối vào wifi có sẵn hoặc tự phát)
- 1 DHT11
- 1 Stepper Motor
- 1 Cảm ứng ánh sáng cho biết đang sáng hay tối.
- 1 LED

Hệ thống sẽ có một trang web, tự động cập nhập mỗi 2s. Trang web sẽ có:

- Thông số nhiệt độ DHT11 đang cảm nhận.
- Thông số độ ẩm DHT11 đang cảm nhận.
- Một hộp text cho phép nhập một số, và một nút bấm. Khi bấm nút, Stepper sẽ quay một góc bằng số nguyên (là số độ) trong hộp text. Nếu số nguyên đó là số âm thì Stepper quay ngược lại.
- Một chuỗi ký tự cho biết trời đang sáng hay đang tối.
- Một nút bấm, ghi "Bật đèn" nếu đèn đang tắt và "Tắt đèn" nếu đèn đang bật. Khi bấm thì LED sẽ đổi giá trị sáng tối.