ỦY BAN NHÂN DÂN HUYỆN CỦ CHI
TRƯỜNG TRUNG CẤP NGHỀ CỦ CHI

GIÁO TRÌNH

MÔ ĐUN: LẬP TRÌNH VI ĐIỀU KHIỂN

NGHỀ: ĐIỆN TỬ CÔNG NGHIỆP

TRÌNH ĐỘ: TRUNG CẤP NGHỀ

Ban hành kèm theo Quyết định số: 89 /QĐ-TCN ngày 15 tháng 8 năm 2024 của Hiệu trưởng trường Trung Cấp Nghề Củ Chi

Củ Chi, năm 2024

TUYÊN BỐ BẢN QUYỀN

LỜI GIỚI THIỆU

Để thực hiện biên soạn giáo trình đào tạo nghề Điện tử công nghiệp ở trình độ TCN, giáo trình Mô đun Lập trình vi điều khiển là một trong những giáo trình mô học đào tạo được biên soạn theo nội dung chương trình khung được Sở Lao động - Thương binh và Xã hội TPHCM và Trường trung cấp nghề Củ Chi ban hành dành cho hệ Trung Cấp Nghề Điện tử công nghiệp.

Nội dung biên soạn ngắn gọn, dễ hiểu, tích hợp kiến thức và kỹ Năng chặt chẽ với nhau, logíc.

Khi biên soạn, người biên soạn đã cố gắng cập nhật những kiến thức mới có liên quan đến nội dung chương trình đào tạo và phù hợp với mục tiêu đào tạo, nội dung lý thuyết và thực hành được biên soạn gắn với nhu cầu thực tế học tập đồng thời có tính thực tiễn cao. Nội dung giáo trình được biên soạn với dung lượng thời gian đào tạo 90 giờ gồm có:

Bài 1: Cài đặt phần mềm vi điều khiển

Bài 2: Lập trình giao tiếp led đơn

Bài 3: Lập trình giao tiếp led 7 đoạn

Bài 4: Lập trình sử dụng timer/counter

Bài 5: Lập trình nút nhấn

Bài 6: Lập trình điều khiển động cơ

Bài 7: Lập trình cảm biến

Bài 8: Lập trình điều khiển từ xa

Bài 9: Lập trình điều khiển không dây

Bài 10: Lập trình giao tiếp LCD

Bài 11: Lập trình giao tiếp ADC

Trong quá trình sử dụng giáo trình, tuỳ theo yêu cầu cũng như khoa học và công nghệ phát triển có thể điều chỉnh thời gian và bổ sung những kiến thức mới cho phù hợp. Trong giáo trình, Tôi có đề ra nội dung bài tập của từng bài để người học củng cố và áp dụng kiến thức phù hợp với kỹ năng.

Mặc dù đã cố gắng tổ chức biên soạn để đáp ứng được mục tiêu đào tạo nhưng không tránh được những khiếm khuyết. Rất mong nhận được đóng góp ý kiến của các thầy, cô giáo, bạn đọc để người biên soạn sẽ hiệu chỉnh hoàn thiện hơn.

Tp. HCM, ngày tháng năm 2024

Giáo viên biên soạn

MỤC LỤC

Trang

MÔ ĐUN LẬP TRÌNH VI ĐIỀU KHIỂN

Tên mô đun: LẬP TRÌNH VI ĐIỀU KHIỂN
Mã mô đun: MĐ18
Thời gian thực hiện mô đun: 90 giờ; (Lý thuyết: 20 giờ; Thực hành, thí nghiệm, thảo luận, bài tập: 68 giờ; Kiểm tra: 2 giờ)

I. Vị trí, tính chất của mô đun:

- Vị trí: Mô đun được bố trí dạy cuối chương trình sau khi học xong các môn học cơ bản như linh kiện điện tử, đo lường điện, kỹ thuật xung số, điện tử công suất.
- Tính chất: Là mô đun bắt buộc.

II. Mục tiêu mô đun:

- Kiến thức:
  - Giải thích được nguyên lý làm việc các hệ điều khiển ứng dụng vi xử lý.
  - Phân tích được nguyên lý hoạt động của vi điều khiển
- Kỹ năng:
  - Lập trình được vi điều khiển arduino
  - Cải tiến được chức năng của vi điều khiển theo yêu cầu.
  - Phát triển được các hệ điều khiển trên cơ sở khối trung tâm là vi xử lý.
  - Sửa chữa các hư hỏng trong lập trình vi điều khiển
- Năng lực tự chủ và trách nhiệm:
  - Rèn luyện tính khéo léo, nhanh nhẹn khi thao tác, tiếp xúc với điện thế.
  - Hình thành tính tỉ mỉ, cẩn thận, chính xác, khoa học và tác phong công nghiệp

BÀI 1: CÀI ĐẶT PHẦN MỀM VI ĐIỀU KHIỂN

Giới thiệu:

Arduino Uno là một board mạch vi điều khiển được phát triển bởi Arduino.cc, một nền tảng điện tử mã nguồn mở chủ yếu dựa trên vi điều khiển AVR Atmega328P. Với Arduino chúng ta có thể xây dựng các ứng dụng điện tử tương tác với nhau thông qua phần mềm và phần cứng hỗ trợ.

Khi arduino chưa ra đời, để làm được một dự án điện tử nhỏ liên quan đến lập trình, biên dịch, chúng ta cần đến sự hỗ trợ của các thiết bị biên dịch khác để hỗ trợ. Ví dụ như, dùng Vi điều khiển PIC hoặc IC vi điều khiển họ 8051..., chúng ta phải thiết kế chân nạp onboard, hoặc mua các thiết bị hỗ trợ nạp và biên dịch như mạch nạp 8051, mạch nạp PIC...

Mục tiêu của bài:

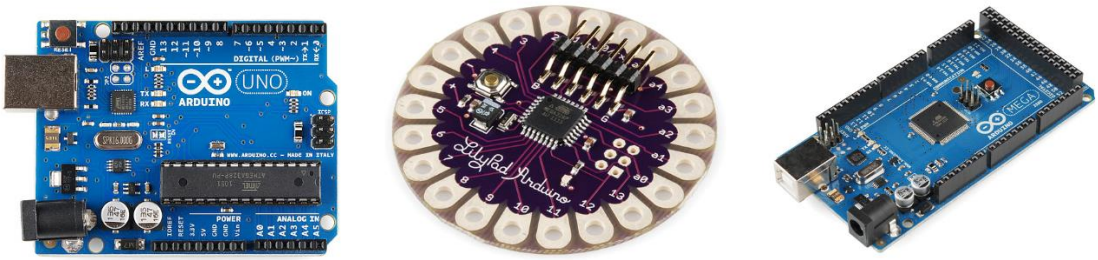
- Trình bày được cấu trúc cơ bản của vi điều khiển
- Sử dụng được ngôn ngữ C cho vi điều khiển
- Phân tích được các hàm cơ bản của Arduino
- Cài đặt được thư viện cho Arduino
- Cài đặt và sử dụng các phần mềm hỗ trợ: Proteus, Arduino IDE
- Rèn luyện tính chính xác, nghiêm túc trong học tập và trong thực hiện công việc.

Nội dung chính:

1. Lập trình C cho vi điều khiển.

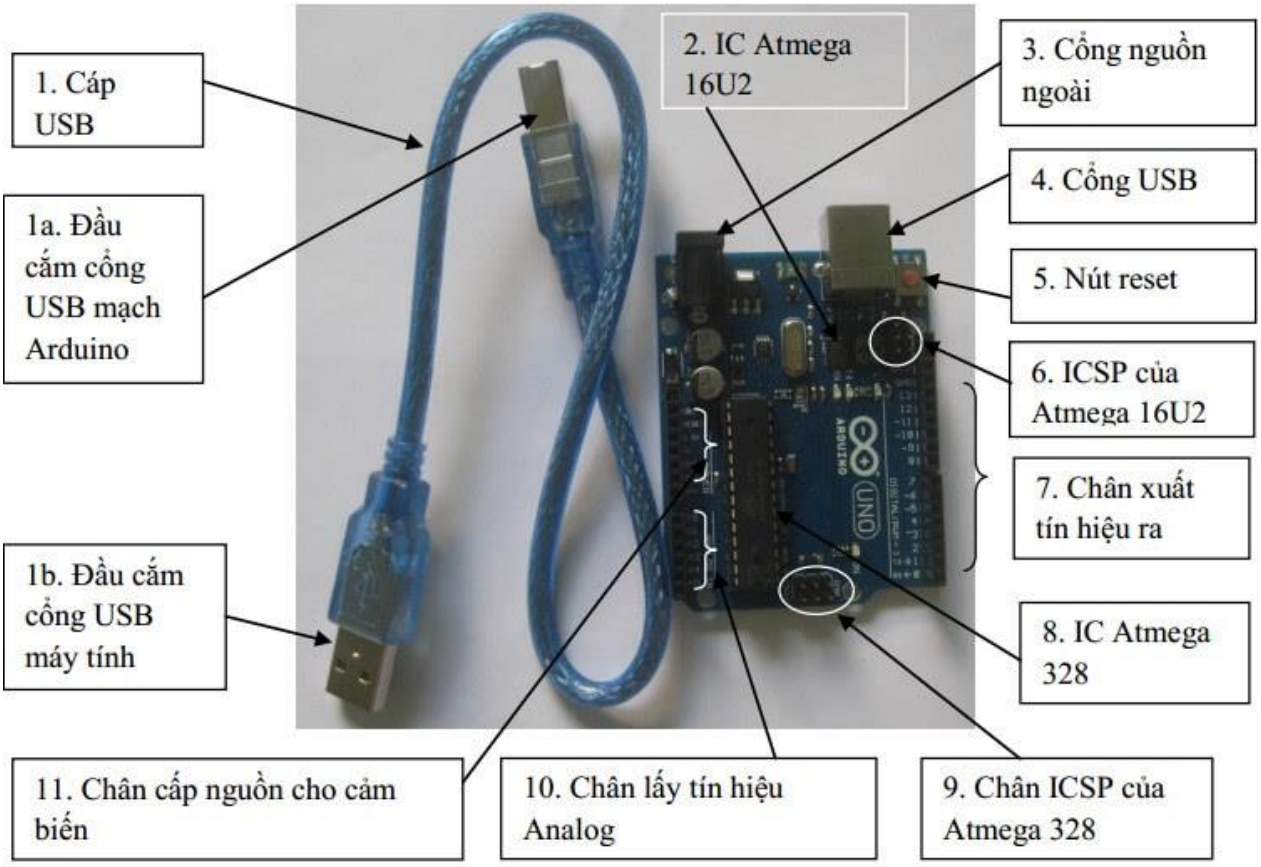
1.1. Giới thiệu vi điều khiển.

Arduino là một bo mạch vi điều khiển do một nhóm giáo sư và sinh viên nước Ý thiết kế và đưa ra đầu tiên vào năm 2005. Mạch Arduino được sử dụng để cảm nhận và điều khiển nhiều đối tượng khác nhau. Nó có thể thực hiện nhiều nhiệm vụ lấy tín hiệu từ cảm biến đến điều khiển đèn, động cơ, và nhiều đối tượng khác. Ngoài ra mạch còn có khả năng liên kết với nhiều module khác nhau như module đọc thẻ từ, ethernet shield, sim900A, …. để tăng khả ứng dụng của mạch.



Hình 1.1.2: Arduino Uno (R3) - Lilypad Arduino - Arduino Mega (R3)

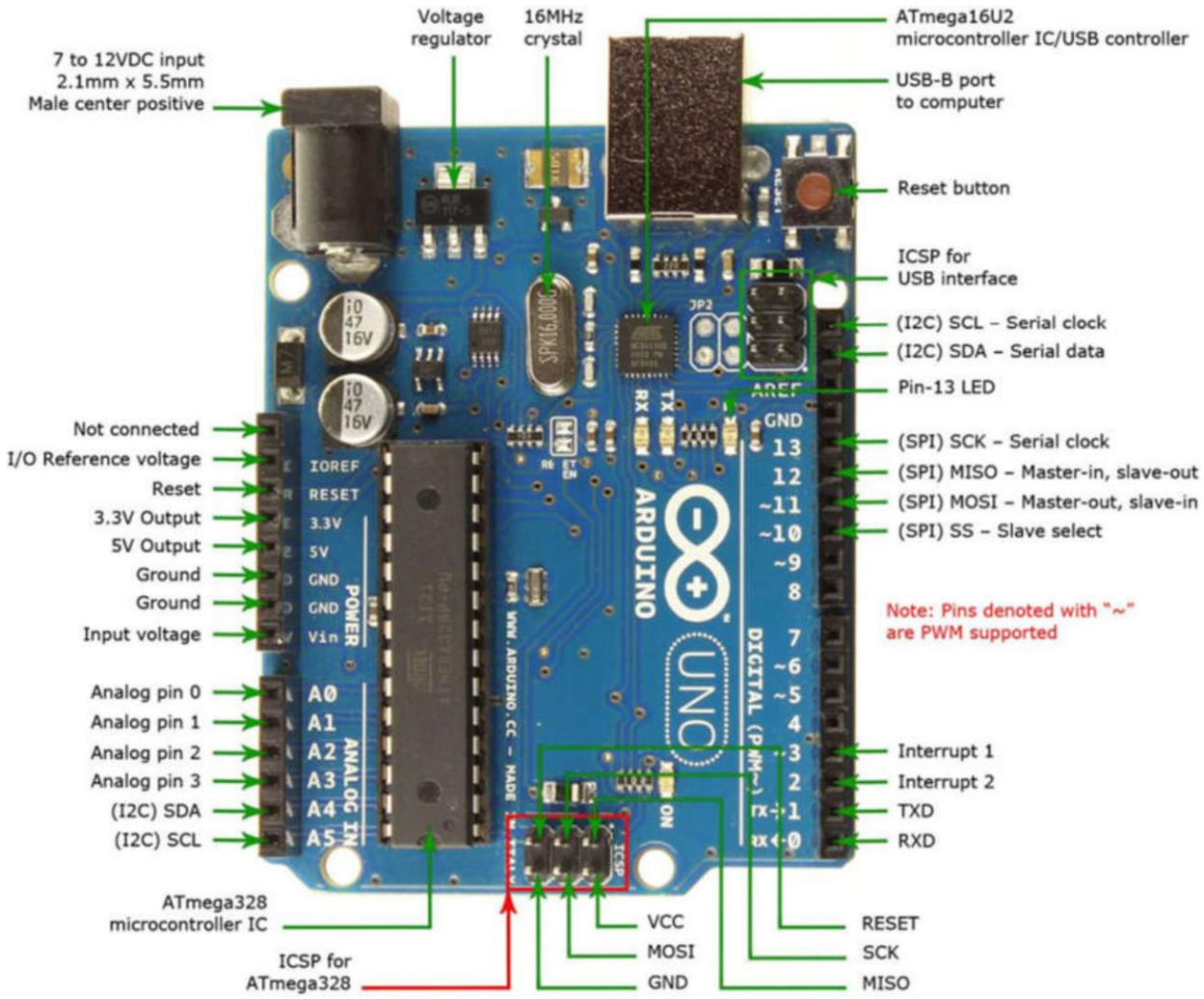
Phần cứng bao gồm một board mạch nguồn mở được thiết kế trên nền tảng vi xử lý AVR Atmel 8bit, hoặc ARM, Atmel 32-bit,…. Hiện phần cứng của Arduino có tất cả 6 phiên bản, Tuy nhiên phiên bản thường được sử dụng nhiều nhất là Arduino Uno và Arduino Mega.



Hình 1.1.1: Cấu tạo của Arduino Uno R3

Thông số cơ bản của Arduino Uno R3

Vi điều khiển	Atmega 328 (họ 8 bit)
Điện áp hoạt động	5V – DC (cấp qua cổng USB)
Tần số hoạt động	16 MHz
Dòng tiêu thụ	30mA
Điện áp vào khuyên dùng	7 – 12V – DC
Điện áp vào giới hạn	6 – 20V – DC
Số chân Digital I/O	14 (6 chân PWM)
Số chân Analog	6 (độ phân giải 10 bit)
Dòng tối đa trên mỗi chân I/O	30 mA
Dòng ra tối đa (5V)	500 mA
Dòng ra tối đa (3.3V)	50 mA
Bộ nhớ flash	32 KB (Atmega328) với 0.5KB dùng bởi bootloader
SRAM	2KB (Atmega328)
EEPROM	1KB (Atmega328)



Hình 1.1.3: Chức năng của các chân trên Arduino

1.2. Lập trình C cho vi điều khiển.

Arduino sử dụng C++ làm ngôn ngữ lập trình chính. Chương trình Arduino có thể được chia làm 3 phần: cấu trúc (structure), biến số (variable) và hằng số (constant), hàm và thủ tục (function).

CẤU TRÚC	GIÁ TRỊ	HÀM VÀ THỦ TỤC
• setup() • loop() **Cấu trúc điều khiển** • if • if...else • switch / case • for • while • break • continue	**Hằng số** • HIGH | LOW • INPUT | INPUT_PULLUP | OUTPUT • LED_BUILTIN • true | false • Hằng số nguyên (integer constants) • Hằng số thực (floating point constants)	**Nhập xuất Digital *(Digital I/O)*** • pinMode() • digitalWrite() • digitalRead() **Nhập xuất Analog *(Analog I/O)*** • analogReference() • analogRead() • analogWrite() - PWM - PPM

- return
- goto

Cú pháp mở rộng

- ; (dấu chấm phẩy)
- {} (dấu ngoặc nhọn)
- // (single line comment)
- /* */ (multi-line comment)
- #define
- #include

Toán tử số học

- = (phép gán)
- + (phép cộng)
- - (phép trừ)
- * (phép nhân)
- / (phép chia)
- % (phép chia lấy dư)

Toán tử so sánh

- == (so sánh bằng)
- != (khác bằng)
- > (lớn hơn)
- < (bé hơn)
- >= (lớn hơn hoặc bằng)
- <= (bé hơn hoặc bằng)

Toán tử logic

- && (và)
- || (hoặc)
- ! (phủ định)
- ^ (loại trừ)

Phép toán hợp nhất

- ++ (cộng thêm 1 đơn vị)
- -- (trừ đi 1 đơn vị)
- += (phép rút gọn của phép cộng)

Kiểu dữ liệu

- void
- boolean
- char
- unsigned char
- byte
- int
- unsigned int
- word
- long
- unsigned long
- short
- float
- double
- array
- string (chuỗi kí tự biểu diễn bằng array)
- String (object)

Chuyển đổi kiểu dữ liệu

- char()
- byte()
- int()
- word()
- long()
- float()

Phạm vi của biến và phân loại

- Phạm vi hiệu lực của biến
- static - biến tĩnh
- const - biến hằng
- volatile

Hàm hỗ trợ

- sizeof()

Hàm thời gian

- millis()
- micros()
- delay()
- delayMicroseconds()

Hàm toán học

- min()
- max()
- abs()
- map()
- pow()
- sqrt()
- sq()
- isnan()
- constrain()
- exp(x)
- frexp(x, int *exp)
- ldexp(x, int exp)
- log(x)
- log10(x)
- modf(x, *i)
- ceil(x)
- floor(x)
- atoi(a[])

Hàm lượng giác

- cos()
- sin()
- tan()
- asin(x)
- acos(x)
- atan(x)
- atan2(x, y)
- cosh(x)
- sinh(x)
- tanh(x)

Sinh số ngẫu nhiên

- randomSeed()
- random()

• -= (phép rút gọn của phép trừ) • *= (phép rút gọn của phép nhân) • /= (phép rút gọn của phép chia)		**Nhập xuất nâng cao *(Advanced I/O)*** • tone() • noTone() • shiftOut() • shiftIn() • pulseIn() **Xử lý chuỗi** • isAscii() • isWhitespace() • isAlpha() • isAlphaNumeric() • isControl() • isDigit() • isGraph() • isLowerCase() • isPrintable() • isPunct() • isSpace() • isUpperCase() • isHexadecimalDigit() • tolower() • toupper() **Bits và Bytes** • lowByte() • highByte() • bitRead() • bitWrite() • bitSet() • bitClear() • bit() **Ngắt (interrupt)** • attachInterrupt() • detachInterrupt() • interrupts() • noInterrupts() **Giao tiếp** • Serial

Ví dụ: Lập trình led chớp tắt

```
/*
  Blink - Nhấp nháy
  Đoạn code làm nhấp nháy một đèn LED cho trước
 */

// 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(1000);               // dừng chương trình trong 1 giây => thây đèn sáng được 1 giây
  digitalWrite(led, LOW);    // tắt đèn led
  delay(1000);               // dừng chương trình trong 1 giây => thấy đèn tối được 1 giây
}
```

1.3. Hư hỏng thường gặp, nguyên nhân và cách khắc phục.

STT	Hư hỏng thường gặp	Nguyên nhân	Cách khắc phục
1	Không upload được chương trình lên mạch Arduino	- Chương trình Arduino IDE. - Driver cho mạch Arduino. - Các kết nối vật lý tới mạch Arduino	- Chọn đúng mạch cần lập trình trong mục Tools > Board, Chọn Arduino Uno R3  - Sau đó kiểm tra phần tử được chọn trong thẻ Tools > Serial Port có phải là cổng Serial được kết nối với Arduino hay không (nếu chưa thấy hãy gỡ kết nối USB từ máy tính và tắt Arduino IDE, sau đó gắn USB lại và bật Arduino).
2	Thông báo java.lang.StackOverflowError khi đang trong qua trình upload chương trình.	- Arduino IDE sử dụng một số biểu thức thường dùng để minh họa trong chương trình (Ví dụ như để thể một ký tự thì ta đặt ký tự đó giữa	Kiểm tra một cách cẩn thận những dấu nháy đơn, nháy kép, dấu gạch chéo ngược \, comments,... Ví dụ: Nếu viết như thế này: '\'' thì sẽ lỗi (hãy

	hai dấu nháy hơn, nhưng để thể hiện một chuỗi thì ta lại đặt chuỗi đó giữa một dấu nháy kép,...). Vì có một số nhầm lẫn về vấn đề sử dụng những biểu thức này nên đã gặp lỗi trên. Một đoạn thông báo lỗi có thể như sau: java.lang.StackOverflowError	thay cặp dấu nháy đơn bên ngoài thành cặp dấu nháy kép như thế này "\"")

2. Cài đặt phần mềm Arduino

2.1. Cấu trúc chương trình Arduino.

Phần 1: Khai báo biến

Đây là phần khai báo kiểu biến, tên các biến, định nghĩa các chân trên board một số kiểu khai báo biến thông dụng: #define

Nghĩa của từ "define" là định nghĩa, hàm #define có tác dụng định nghĩa, hay còn gọi là gán, tức là gán một chân, một ngõ ra nào đó với 1 cái tên.

Ví dụ: #define led 13

Chú ý: sau #define thì không có dấu "," (dấy phẩy)

Khai báo các kiểu biến khác như: int (kiểu số nguyên), float,…

Phần 2: Thiết lập (void setup())

Phần này dùng để thiết lập cho chương trình, cần nhớ rõ cấu trúc của nó

```
void setup()
{
…..
}
```

Cấu trúc của nó có dấu ngoặc nhọn ở đầu và ở cuối, nếu thiếu phần này khi kiểm tra chương trình thì chương trình sẽ báo lỗi.

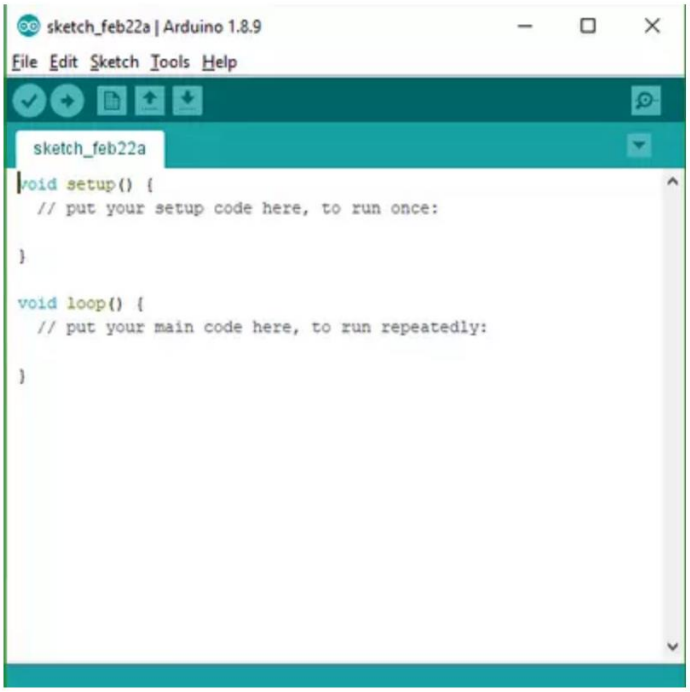
Phần này dùng để thiết lập các tốc độ truyền dữ liệu, kiểu chân là chân ra hay chân vào. Trong đó:

Serial.begin(9600);	Dùng để truyền dữ liệu từ board Arduino lên máy tính
pinMode(biến, kiểu và hoặc ra) Ví dụ: `pinMode(ChanDO, INPUT);`	Dùng để xác định kiểu chân là đầu vào hay đầu ra

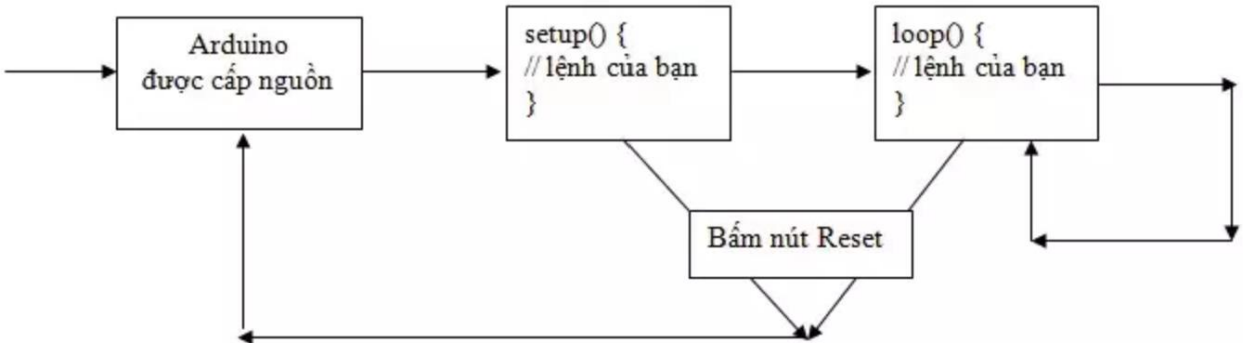
Phần 3: Vòng lặp

Dùng để viết các lệnh trong chương trình để mạch Arduino thực hiện các nhiệm vụ mà chúng ta mong muốn, thường bắt đầu bằng:

```
void loop()
{
.....
}
```



Hình 1.2.1: Cấu trúc chương trình Arduino



Hình 1.2.2: Tổng quan quá trình xử lý chương trình Arduino.

Một số ký hiệu và câu lệnh thường gặp

Ký hiệu, câu lệnh	Ý nghĩa
//	Dấu // dùng để giải thích, khi nội dung giải thích nằm trên 1 dòng, khi kiểm tra chương trình thì phần kiểm tra sẽ bỏ qua phần này, không kiểm tra

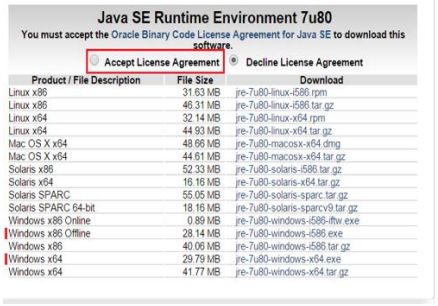
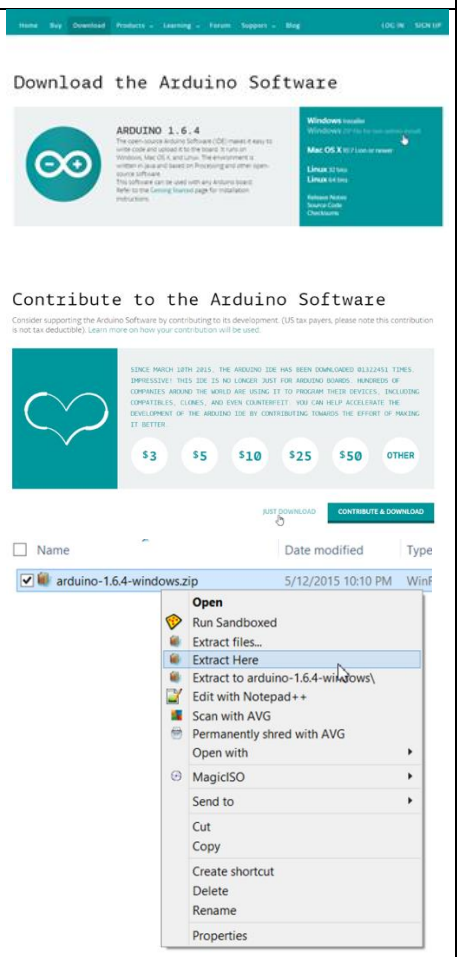
/* …. */	Ký hiệu này cũng dùng để giải thích, nhưng giải thích dành cho 1 đoạn, tức có thể xuống dòng được
#define biến chân	Define nghĩa là định nghĩa, xác định. Câu lệnh này nhằm gán tên 1 biến vào 1 chân nào đó. Ví dụ #define led 13
digitalWrite(chân, trạng thái);	Dùng để tắt, mở 1 chân ra. Cú pháp của nó là digitalWrite(chân,trạng thái chân);. Ở đây trạng thái chân có thể là HIGH hoặc LOW. Ví dụ: digital(led,HIGH); , hoặc digital(led,LOW); . Chú ý dấu chấm phẩy đằng sau câu lệnh.
analogWrite(chân, giá trị);	Có ý nghĩa dùng để băm xung (PWM), thường dùng để điều khiển tốc độ động cơ, độ sáng led,..
digitalRead(chân);	Read nghĩa là đọc, lệnh này dùng để đọc giá trị digital tại chân muốn đọc
analogRead(chân);	Read nghĩa là đọc, lệnh này dùng để đọc giá trị analog tại chân muốn đọc
delay(thời gian);	Delay nghĩa là chờ, trì hoãn, duy trì. Lệnh này dùng để duy trì trạng thái đang thực hiện chờ một thời gian. Thời gian ở đây được tính bằng mili giây, 1 giây bằng 1 ngàn mili giây.

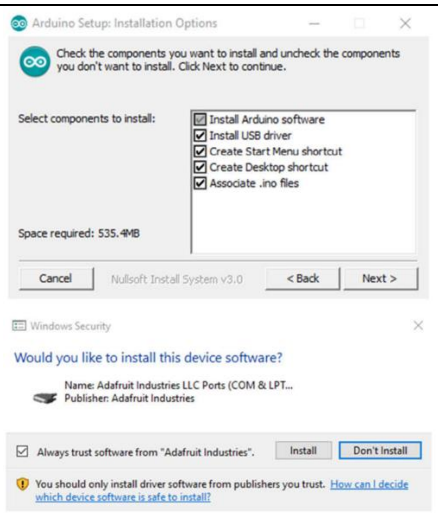
if()
{
Các câu lệnh}
else (){
Các câu lệnh}
	if nghĩa là nếu, sau if là dấu (), bên trong dấu ngoặc là một biểu thứ so sánh. Ví dụ trong bài về cảm biến độ ẩm đất (phần 5) thì: if (giatriAnalog>500) //nếu giá trị đọc được của biến giatriAnalog lớn hơn 500
	{
	digitalWrite(Led,HIGH); //Ra lệnh cho led sáng
	delay(1000);//chờ 1s}
	else nghĩa là ngược lại

Serial.print()	In ra màn hình máy tính, lệnh này in không xuống dòng
Serial.println()	In ra màn hình máy tính, in xong xuống dòng, giá trị tiếp theo sẽ được in ở dòng kế tiếp

2.2. Cài đặt phần mềm Arduino.

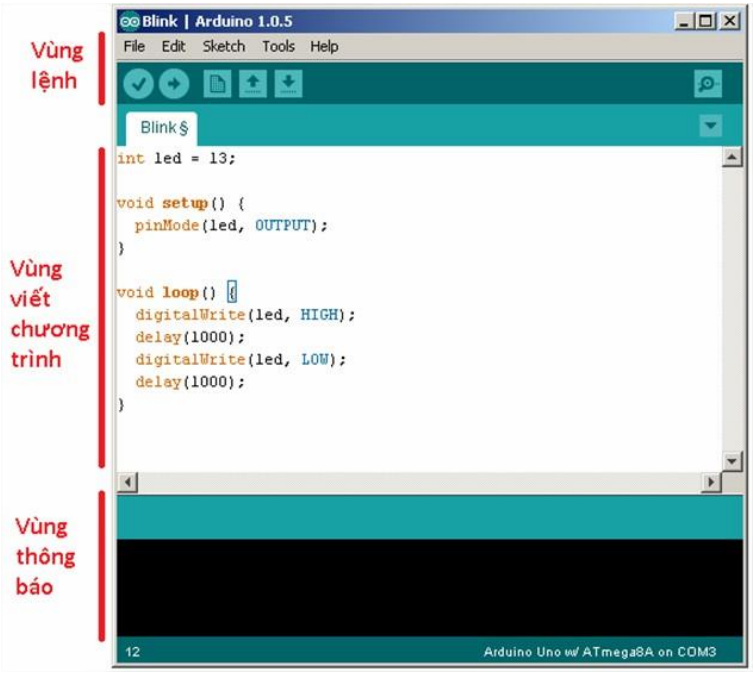
Phiếu hướng dẫn thực hành cài đặt phần mền Arduino

TT	Nội dung	Mô tả	Hình ảnh minh họa
Bước 1	**Cài đặt Java Runtime Environment (JRE)**	Arduino IDE được viết trên Java nên cần phải cài đặt JRE trước Arduino IDE. Link tải: http://www.oracle.com/technetwork/ja... 2 bản JRE phổ biến nhất là bản dành cho Windows 32bit (x86) và Windows 64bit (x64). Chọn "Accept License Agreement".	
Bước 2	Down phần mền Arduino IDE	- Truy cập địa chỉ http://arduino.cc/en/Main/Software/ Đây là nơi lưu trữ cũng như cập nhật các bản IDE của Arduino. - Bấm vào mục Windows ZIPfile for non admin install - Bấm chọn JUST DOWNLOAD - Bấm chuột phải vào file vừa download arduino-1.6.4-windows.zip và chọn “Extract here” để giải nén	
Bước 3	**Cài đặt phần mền arduino**	- Bấm vào file tải về để tiến hành cài đặt, chọn “I Agree” để tiếp tục cài đặt chương trình.	

		- Lựa chọn việc cài đặt các phần liên quan được hiện ra và chọn "Next" để tiếp tục quá trình cài đặt. - Trong quá trình cài đặt 1 số phiên bản sẽ hỏi có cài driver USB cho phần mềm IDE không? Tích chọn vào ô vuông "Always trust software from "Arduino LLC"", sau đó bấm "Install" để cài đặt driver USB. Cần phải cài driver này thì chương trình mới nhận cổng USB của mạch Arduino	
Bước 4	**Khởi động chương trình**	Bấm chạy icon Arduino vừa cài đặt để khởi động chương trình 	
Bước 5	**Cài đặt Driver**	Để máy tính và board Arduino giao tiếp được với nhau, cần phải cài đặt driver trước tiên. Nếu dùng Windows 8, trong một số trường hợp Windows không cho phép cài Arduino driver (do driver không được kí bằng chữ kí số hợp lệ). Do vậy cần vào Windows ở chế độ Disable driver signature enforcement thì mới cài được driver Đầu tiên, chạy file arduino-1.6.4\drivers\dpinst-x86.exe	

		(Windows x86) hoặc arduino-1.6.4\drivers\dpinst-amd64.exe (Windows x64). Cửa sổ "Device Driver Installation Wizard" hiện ra, chọn Next để tiếp tục.	

*** Giao diện**



*** Vùng lệnh:** Bao gồm các nút lệnh menu (File, Edit, Sketch, Tools, Help). Phía dưới là các icon cho phép sử dụng nhanh các chức năng thường dùng của IDE được miêu tả như sau:

Icon	Chức năng
	Biên dịch chương trình đang soạn thảo để kiểm tra các lỗi lập trình.
	Biên dịch và upload chương trình đang soạn thảo.
	Mở một trang soạn thảo mới.
	Mở các chương trình đã lưu.
	Lưu chương trình đang soạn.
	Mở cửa sổ Serial Monitor để gửi và nhận dữ liệu giữa máy tính và board Arduino.

*** Vùng viết chương trình.**

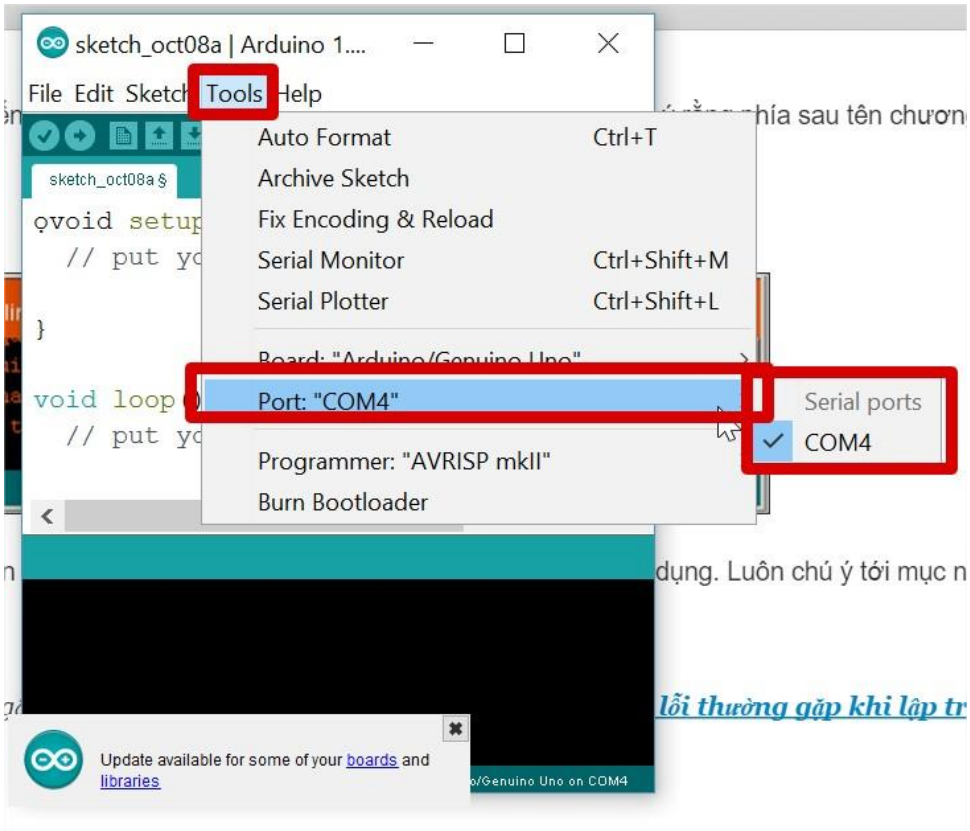
Viết các đoạn mã của mình tại đây. Tên chương trình được hiển thị ngay dưới dãy các Icon, ở đây nó tên là "Blink". Để ý rằng phía sau tên chương trình có một dấu "§". Điều đó có nghĩa là đoạn chương trình chưa được lưu lại.

*** Vùng thông báo (debug)**

Những thông báo từ IDE sẽ được hiển thị tại đây. Để ý rằng góc dưới cùng bên phải hiển thị loại board Arduino và cổng COM được sử dụng. Luôn chú ý tới mục này bởi nếu chọn sai loại board hoặc cổng COM, sẽ không thể upload được code của mình.

**** Lưu ý***

Khi lập trình, cần chọn port (cổng kết nối khi gắn board vào) và board (tên board mà sử dụng). Giả sử, đang dùng mạch Arduino Uno, và khi gắn board này vào máy tính bằng cáp USB nó được nhận là COM4 thì chỉnh như thế này là có thể lập trình đươc.



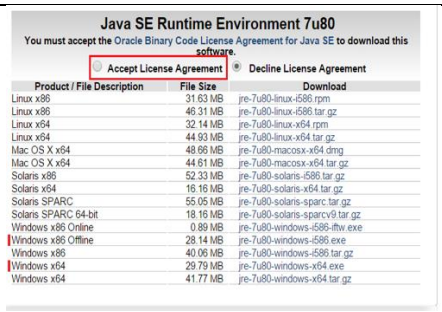
2.3. Hư hỏng thường gặp, nguyên nhân và cách khắc phục

STT	Hư hỏng thường gặp	Nguyên nhân	Cách khắc phục
1	Xuất hiện lỗi: "Build folder disappeared or could not be written"	Không upload được chương trình	Copy file Arduino.app ra khỏi file image (và dán vào thư mục Applications). Nếu không, sẽ không thể upload được chương trình
2	Chương trình Arduino IDE bị treo khi đang upload chương trình	Do sự sung đột của chương trình Arduino IDE với một tiến trình của chương trình Logitech ('LVPrcSrv.exe').	Hãy bật Task Manager và End process nó.
3	Chương trình đã upload lên mạch rồi mà không hoạt động.	Vấn đề này sẽ xảy ra khi đang gửi dữ liệu qua cổng Serial của Arduino (lúc mà nó chưa khởi động). Bởi vì trước khi khởi động một chương trình Arduino, thì bootloader sẽ được khởi động trước và nó đọc tín hiệu từ cổng Serial (cho đến khi hết thì thôi). Và nếu cứ xuất hiện tín hiệu từ cổng Serial thì chương trình sẽ không bao giờ hoạt động được.	Ngắt kết nối cổng TX và RX hoặc nối TX và RX với nhau
4	Arduino IDE đã báo là upload thành công nhưng mạch không hoạt động	Chọn chip sai trong Tools > Microcontroller.	Kiểm tra chip là gì (Dòng chữ lớn nhất được ghi trên con chip, chẳng hạn như là ATmega328, ATmega8,...) và chọn. Kiểm tra thử nguồn điện của có bị nhiễu hay không, thi thoảng thì cũng vì lý do này mà chương trình bị mất. Chương trình quá lớn (sau khi biên dịch và quá sức chứa bộ nhớ flash của chip), vì vậy đôi khi trong các phiên bản cũ của Arduino IDE, chương trình

		được biên dịch một phần (Từ đầu đến lúc đạt dung lượng tối đa). Vì vậy, khi mạch Arduino nhận được một chương trình lỗi như thế, nó sẽ cứ lặp đi lặp lại lệnh reset, rồi lại dừng rồi lại reset,..

CÂU HỎI ÔN TẬP

1. Cấu trúc của chương trình Arduino bao gồm mấy phần? Hãy giải thích chức năng của từng phần?
2. Điền vào các phần còn thiếu trong bảng hướng dẫn thực hiện cài đặt phần mền Arduino?

TT	Nội dung	Mô tả	Hình ảnh minh họa
Bước 1	**Cài đặt Java Runtime Environment (JRE)**		
Bước 2	Down phần mền Arduino IDE		
Bước 3	**Cài đặt phần mền arduino**		
Bước 4	**Khởi động chương trình**		
Bước 5	**Cài đặt Driver**		

BÀI 2: LẬP TRÌNH GIAO TIẾP LED ĐƠN

Giới thiệu:

Arduino Uno là một bảng mạch vi điều khiển nguồn mở dựa trên vi điều khiển Microchip ATmega328 được phát triển bởi Arduino.cc. Bảng mạch được trang bị các bộ chân đầu vào/ đầu ra Digital và Analog có thể giao tiếp với các bảng mạch mở rộng khác nhau. Mạch Arduino Uno thích hợp cho những bạn mới tiếp cận và đam mê về điện tử, lập trình…Dựa trên nền tảng mở do Arduino.cc cung cấp các bạn dễ dàng xây dựng cho mình một dự án nhanh nhất (lập trình Robot, xe tự hành, điều khiển bật tắt led…).

Mục tiêu của bài:

- Hiển thị được thông tin lên các thiết bị ngoại vi
- Phân tích các chương trình có sẵn.
- Lập trình được với led đơn
- Thực hiện các ứng dụng thực tế thông qua các mô hình.
- Rèn luyện tính chính xác, nghiêm túc trong học tập và trong thực hiện công việc.

Nội dung chính:

1. Led đơn.

1.1. Led đơn tích cực mức thấp.

Khi nối đèn LED vào mạch, phải nối chính xác các cực tương ứng.

Không như điện trở, đèn LED là một linh kiện điện tử có phân cực - trong đó anode là cực dương + và cathode - là cực âm. Chân dài hơn của đèn LED là cực dương + và chân ngắn hơn là cực âm -.

Trong trường hợp 2 chân bằng nhau, có thể quan sát bên trong đèn, đầu nhỏ (bên trái của hình trên) là cực dương + và đầu lớn hơn là cực âm -.

1.2. Led đơn tích cực mức cao.

Mỗi loại đèn LED hoạt động ở một hiệu điện thế khác nhau. Thông thường, với loại đèn LED siêu sáng thì hiệu điện thế hoạt động phổ biến trong khoảng từ 1.7V đến 3.3V. Theo lý thuyết, mạch Arduino cấp nguồn ra ở mức hiệu điện thế là 5V. Do đó, phải mắc thêm một điện trở để giảm hiệu điện thế, tránh gây hư hỏng linh kiện.

Giả định đèn LED chịu được hiệu điện thế tối đa là 1.7V, trở kháng của điện trở mắc vào được tính theo công thức sau:

$$R = (Vs - Vf) \div I$$

Trong đó:

R: trở kháng của điện trở mắc vào.

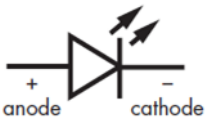
Vs (supply voltage): hiệu điện thế nguồn cấp cho đèn LED, cũng chính là nguồn cấp từ Arduino - 5V.

Vf (forward voltage): hiệu điện thế mà đèn LED cần sử dụng.

I: cường độ dòng điện qua đèn LED - với Arduino sẽ là 10mA.

Dựa theo công thức trên, trở kháng của điện trở trong trường hợp này là 330Ω. Tuy nhiên, đây là ngưỡng mà đèn LED của có thể chịu được. Để an toàn cho mạch, nên sử dụng một điện trở lớn hơn, trong trường hợp này ở mức 560 Ω là hợp lý.

Trong lược đồ thiết kế mạch, đèn LED được biểu diễn bằng ký hiệu:



Color of LED	Voltage Drop (Volt)
Red	1.63 ~ 2.03
Yellow	2.10 ~ 2.18
Orange	2.03 ~ 2.10
Blue	2.48 ~ 3.7
Green	1.9 ~ 4.0
Violet	2.76 ~ 4.0
UV	3.1 ~ 4.4
White	3.2 to 3.6

2. Lập trình giao tiếp led đơn

2.1. Chuẩn bị dụng cụ, thiết bị vật liệu.

Định luật Ohm xác định mối quan hệ giữa điện áp, điện trở và cường độ dòng điện với công thức là:

$$I = \frac{V}{R}$$

Từ công thức trên chúng ta có thể tính được điện trở tương ứng cho LED, với LED 5MM dòng điện tiêu thụ là 20mA, điện áp tiêu thụ của LED đỏ là 2V.

Thiết bị, linh kiện	Số lượng
Board Arduino	01
Testboard	01
Dây cắm board	02
Led 5mm	01
Điện trở 220 Ohm	01

2.2. Lập trình giao tiếp led đơn

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 dụ này 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, digitalWrite HIGH cho chân số 13 (chân Digtal 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 có thể các loại điện trở khác <= 10kOhm). Để tắt một đèn LED, bạn sử dụng hàm: (xem thêm về LOW)

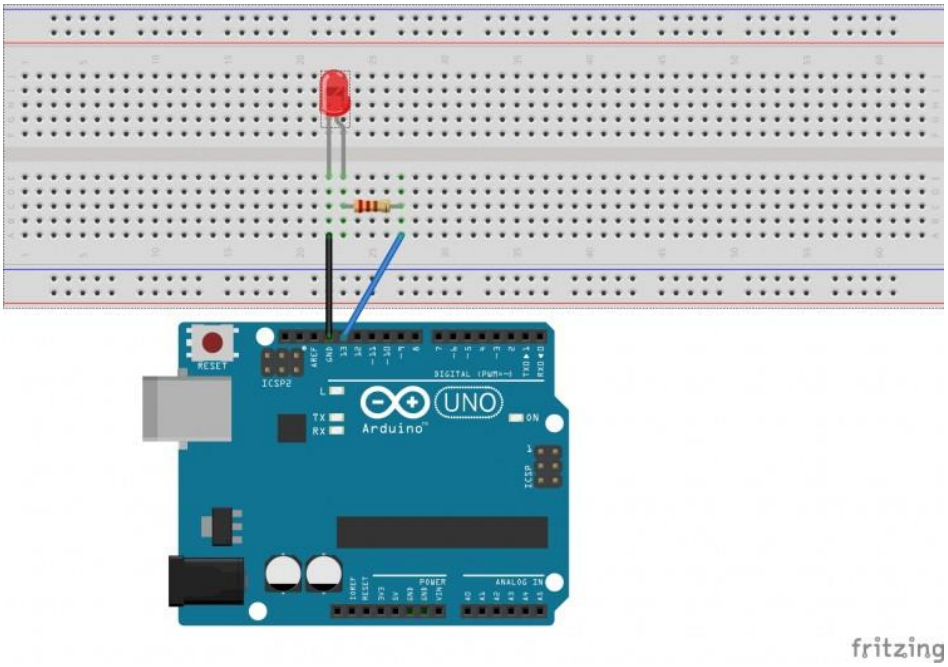
digitalWrite(13,LOW);

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 dừng 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)!

Đoạn code lập trình.

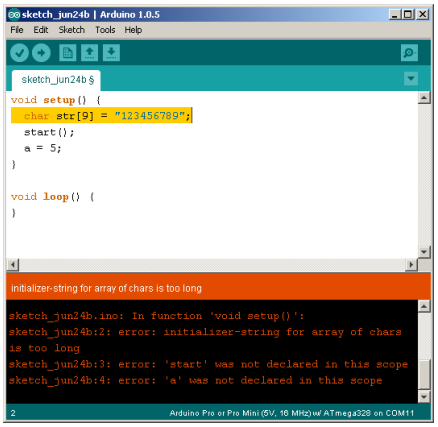
```
/*
  Blink - Nhấp nháy
  Đoạn code làm nhấp nháy một đèn LED cho trước
 */

// 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(1000);               // dừng chương trình trong 1 giây => thấy đèn sáng được 1 giây
  digitalWrite(led, LOW);    // tắt đèn led
  delay(1000);               // dừng chương trình trong 1 giây => thấy đèn tối được 1 giây
}
```

2.3. Kết nối dây dẫn và vận hành mạch

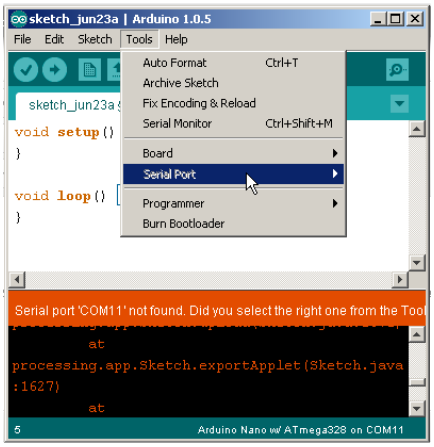
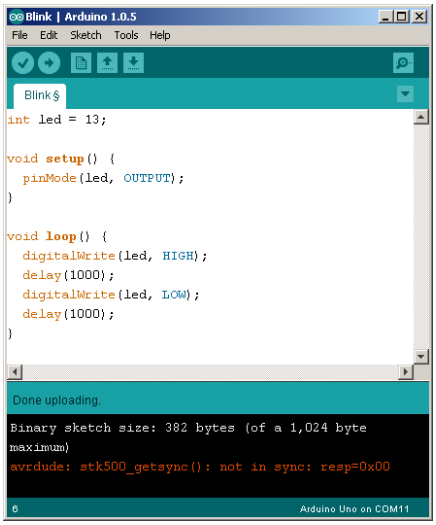
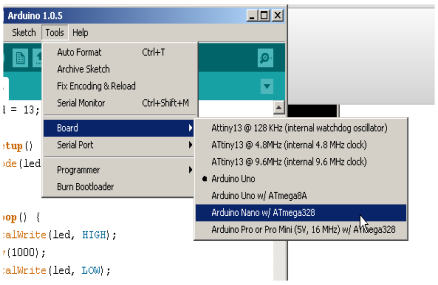


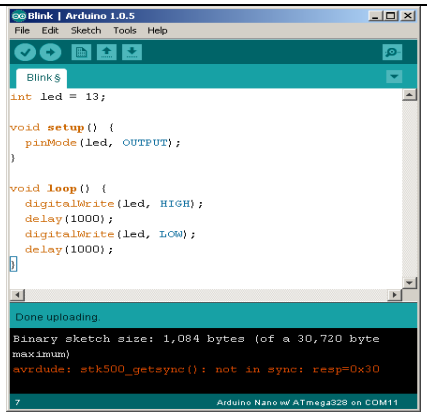
2.4. Hư hỏng thường gặp, nguyên nhân và cách khắc phục.

STT	Hư hỏng thường gặp	Nguyên nhân	Cách khắc phục

1	Biên dịch gặp lỗi
	 
	sketch_jun24b:2: error: initializer-string for array of chars is too long
	sketch_jun23a:3: error: 'start' was not declared in this scope
	sketch_jun23a:4: error: 'a' was not declared in this scope
		Cửa sổ soạn thảo sketch_jun23a, dòng 2, độ dài của chuỗi vượt quá độ dài khai báo
		Cửa sổ soạn thảo sketch_jun23a, dòng 3, không tìm thấy đối tượng "start"
		Cửa sổ soạn thảo sketch_jun23a, dòng 4, không tìm thấy đối tượng "a"
			Khai báo kiểu dữ liệu cho biến ***"a"*** và định nghĩa hàm ***start(),*** chỉnh lại kích thước chuỗi ***str*** cho đúng.
			`int a;`
			`char str[10] = "123456789"; //xem phần "Chú ý" của string`
			`void setup() {`
			`  start();`
			`  a = 6;`
			`}`
			
			`void loop() {`
			`}`
			
			`void start() {`
			`  Serial.begin(9600);`
			`  Serial.println("Hello Arduino");`
			`}`

2	Lỗi Serial port "COM11" not found 	Không tìm thấy mạch Arduino nào ở cổng Serial COM11.	Vào menu Tool -> Serial Port để chọn đúng cổng Serial. Cổng Serial đang dùng luôn được hiển thị ở góc dưới cùng bên phải như trên hình (COM11).

			Nếu đã vào menu Tool -> Serial Port rồi nhưng lại gặp lỗi này  có thể đã quên kết nối Arduino với máy tính hoặc dây cáp kết nối bị lỏng.
3	Lỗi avrdude: stk500_getsync(): not in sync: resp=0x00  	Chọn sai board Arduino khi upload chương trình. Chân RESET trên board Arduino bị nối xuống GND hay đang ở mức LOW.	Vào menu Tools -> Board để chọn lại đúng mạch Arduino đang sử dụng.  Đặt lại chân RESET ở mức HIGH hoặc đơn giản là tháo các dây đang nối với chân này ra. Theo thiết kế mạch mặc định của Arduino thì chân RESET luôn được nối với Vcc (đặt ở mức HIGH).
4	Lỗi avrdude: stk500_getsync(): not in sync: resp=0x30	Cổng Serial trên Arduino đang được sử dụng, không thể upload chương trình. Chưa cài đặt bootloader. Chưa cài đặt driver cho Arduino.	Tháo bất cứ dây nào đang cắm vào chân Digital 0 (chân RX) trên mạch Arduino của bạn. Cài đặt bootloader cho Arduino. Cài đặt driver cho Arduino.



3. Bài tập mở rộng.

3.1. Thay đổi độ sáng của đèn.

Phần cứng

Arduino Uno

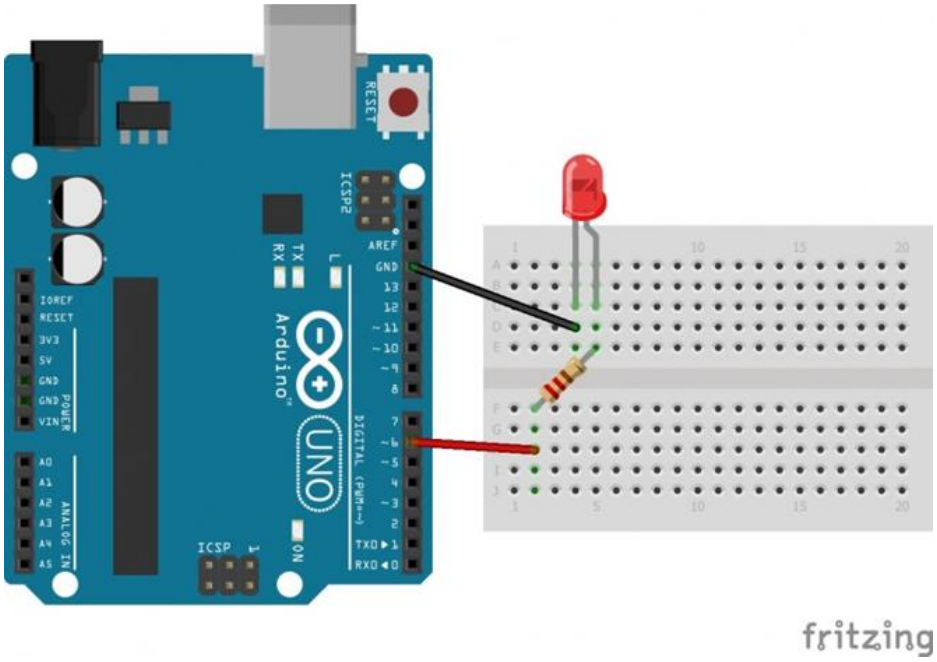
Breadboard

Dây cắm breadboard

1 điện trở 560 Ohm (hoặc 220 Ohm hoặc 1kOhm)

1 đèn LED siêu sáng

Lắp mạch



Lập trình Arduino.

Lưu ý những chân digital có dấu ~ phía trước và những chân analog mới hỗ trợ analogWrite

```
1. int led = 6; // cổng digital mà LED được nối vào
2. int brightness = 0; // mặc định độ sáng của đèn là
```

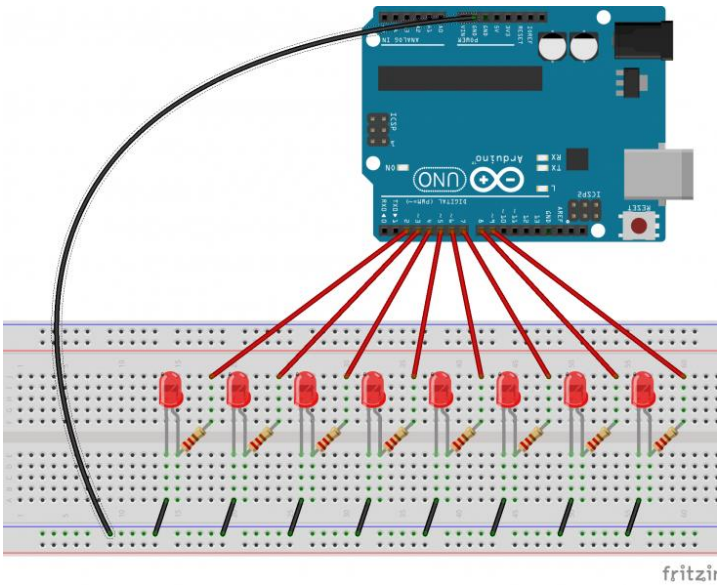
```
3. int fadeAmount = 5; // mỗi lần thay đổi độ sáng thì thay đổi với giá
   trị là bao nhiêu
4.
5.
6. void setup() {
7. // pinMode đèn led là OUTPUT
8. pinMode(led, OUTPUT);
9. }
10.
11.      void loop() {
12.      //xuất giá trị độ sáng đèn LED
13.      analogWrite(led, brightness);
14.
15.      // thay đổi giá trị là đèn LED
16.      brightness = brightness + fadeAmount;
17.
18.      // Đoạn code này có nghĩa nếu độ sáng == 0 hoặc bằng == 255 thì
   sẽ đổi chiều của biến thay đổi độ sáng. Ví dụ, nếu đèn từ sáng yếu -
   -> sáng mạnh thì fadeAmount dương. Còn nếu đèn sáng mạnh --> sáng
   yếu thì fadeAmmount lúc này sẽ có giá trị âm
19.      if (brightness == 0 || brightness == 255) {
20.      fadeAmount = -fadeAmount ;
21.      }
22.      //đợi 30 mili giây để thấy sự thay đổi của đèn
23.      delay(30);
24.      }
```

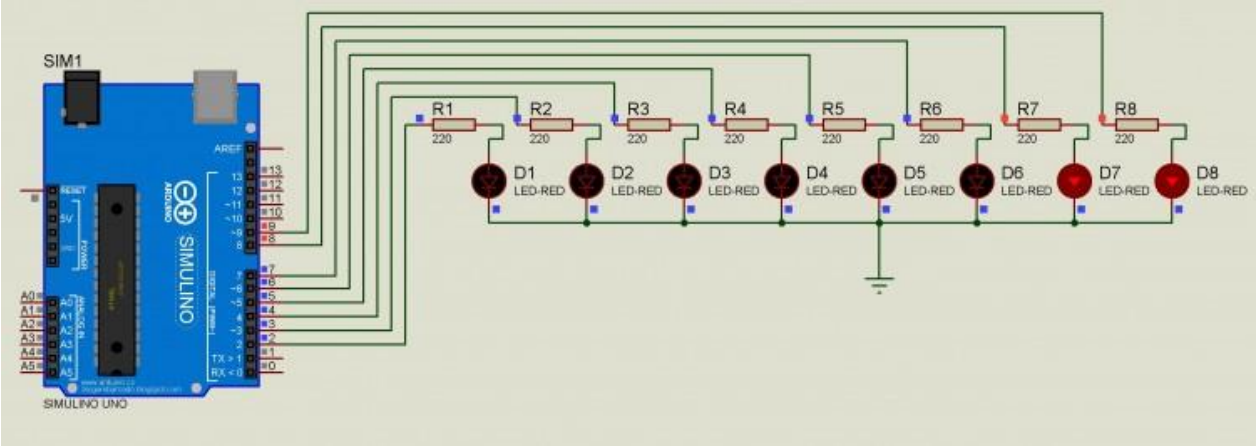
3.2. Điều khiển 8 đèn LED sáng.

Phần cứng

- Arduino Uno
- 8 điện trở 560 Ohm (hoặc 220 Ohm hoặc 1kOhm)
- Breadboard
- Dây cắm breadboard
- 8 đèn LED siêu sáng

Lắp mạch





1. pinMode
2. digitalWrite
3. array
4. for
5. delay
6. int
7. byte
8. sizeof

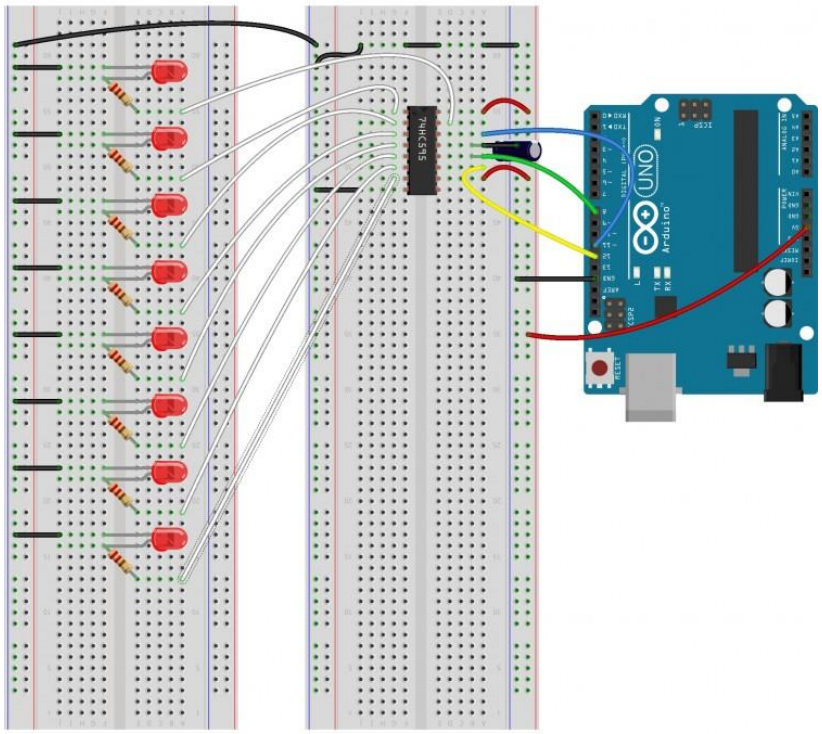
```
byte ledPin[] = {2,3,4,5,6,7,8,9}; // Mảng lưu vị trí các chân Digital mà các đèn LED sử dụng theo thứ tự từ 1->8. Bạn có thể thêm các LED bằng cách thêm các chân digital vào mảng này
byte pinCount; // Khai báo biến pinCount dùng cho việc lưu tổng số chân LED
void setup() {
  pinCount = sizeof(ledPin);  //Xem thêm thông tin về hàm sizeof tại http://arduino.vn/reference/sizeof
  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
  }
}
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) {
```

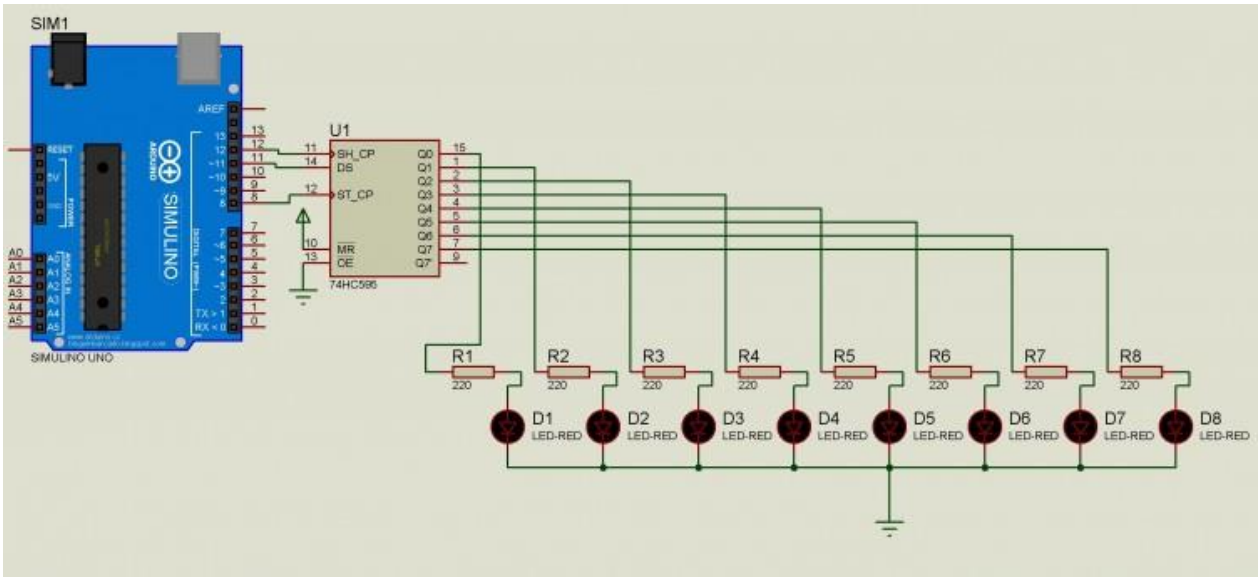
```
    digitalWrite(ledPin[i],LOW); // Tắt đèn
    delay(500); // Dừng để các đèn LED tắt dần
  }
}
```

3.3. Điều khiển nhiều đèn LED sáng dùng IC 595.

Để tạo mạch LED 8 đèn thì cần 20 phút.

IC 74HC595 (trên thị trường nhất nhiều loại, chỉ cần tìm IC nào có chữ này là được). Nếu muốn điều khiển nhiều LED hơn thì hãy mua nhiều hơn 1 IC (cứ 1 IC điều khiển được 8 LED).



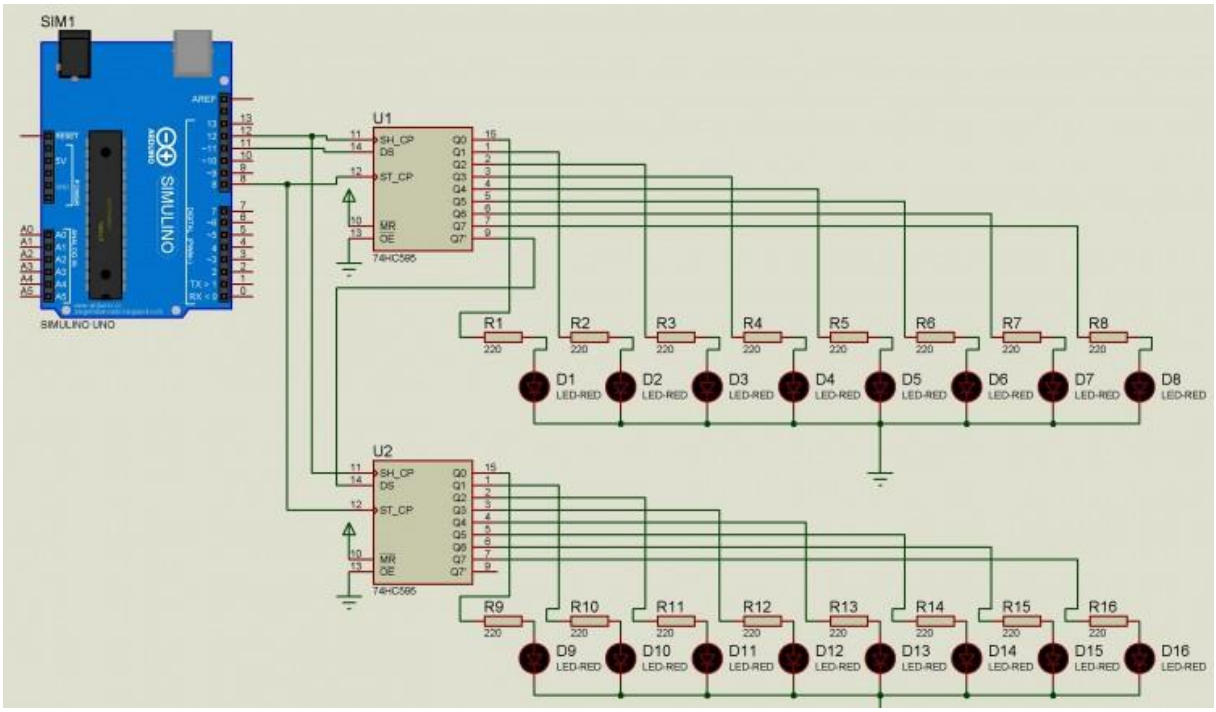


```
/*
shiftOut với 8 LED bằng 1 IC HC595
*/
//chân ST_CP của 74HC595
int latchPin = 8;
//chân SH_CP của 74HC595
int clockPin = 12;
//Chân DS của 74HC595
int dataPin = 11;

//Trạng thái của LED, hay chính là byte mà ta sẽ gửi qua shiftOut
byte ledStatus;
void setup() {
  //Bạn BUỘC PHẢI pinMode các chân này là OUTPUT
  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
}

void loop() {
  /*
    Trong tin học, ngoài các phép +, -, *, / hay % mà bạn đã biết trên hệ cơ
số 10.
    Thì còn có nhiều phép tính khác nữa. Và một trong số đó là Bit Math (toán
bit) trên hệ cơ số 2.
    Để hiểu những gì tôi viết tiếp theo sau, bạn cần có kiến thức về Bit Math.
    Để tìm hiểu về Bit Math, bạn vào mục Tài liệu tham khảo ở bảng chọn nằm
phía trên cùng trang web và chạy xuống khi bạn kéo chuột trên trang Arduino.VN

  */
  //Sáng tuần tự
  ledStatus = 0;//mặc định là không có đèn nào sáng hết (0 = 0b00000000)
  for (int i = 0; i < 8; i++) {
    ledStatus = (ledStatus << 1) | 1;//Đẩy toàn bộ các bit qua trái 1 bit và
cộng bit có giá trị là 1 ở bit 0

    /**
      Bắt buộc phải có để shiftOut
    **/

    digitalWrite(latchPin, LOW); //các đèn LED sẽ không sáng khi bạn digital
LOW

    //ShiftOut ra IC
    shiftOut(dataPin, clockPin, MSBFIRST, ledStatus);

    digitalWrite(latchPin, HIGH);//các đèn LED sẽ sáng với trạng thái vừa được
cập nhập

    /**
      Kết thúc bắt buộc phải có
    **/

    delay(500); // Dừng chương trình khoảng 500 mili giây để thấy các hiệu ứng
của đèn LED
  }
  //Tắt tuần tự
  for (int i = 0;i<8;i++) {
    ledStatus <<= 1; //Đẩy tất cả các bit qua bên trái 1 bit
```

```
    digitalWrite(latchPin, LOW);
    shiftOut(dataPin, clockPin, MSBFIRST, ledStatus);
    digitalWrite(latchPin, HIGH);
    delay(500);
  }
}
```

BÀI TẬP

Điều khiển nhiều đèn LED bằng Arduino với nhiều IC HC 595




```
/*
shiftOut với 8 LED bằng 1 IC HC595
*/
//chân ST_CP của 74HC595
int latchPin = 8;
//chân SH_CP của 74HC595
int clockPin = 12;
//Chân DS của 74HC595
int dataPin = 11;

     //Trạng thái của LED, hay chính là byte mà ta sẽ gửi qua shiftOut
     const int HC595_COUNT = 2;//Nếu bạn dùng nhiều hơn thì thay bằng một số lớn hơn 2.
     byte ledStatus[HC595_COUNT]= {0};
     void setup() {
     //Bạn BUỘC PHẢI pinMode các chân này là OUTPUT
     pinMode(latchPin, OUTPUT);
     pinMode(clockPin, OUTPUT);
     pinMode(dataPin, OUTPUT);
     }

     void fillValueToArray(byte value) {
```

```
22.      for (int i = 0;i < HC595_COUNT; i += 1) {
23.      ledStatus[i] = value;
24.      }
25.      }
26.
27.      void shiftOutHC595(int dataPin, int clockPin, byte ledStatus[])
   {
28.      digitalWrite(latchPin, LOW);
29.
30.      for (int i = 0; i < HC595_COUNT; i++) {
31.      shiftOut(dataPin,clockPin,LSBFIRST,ledStatus[i]); // Chúng ta
   dùng LSBFIRST thay cho MSBFIRST là có lý do của nó, cái này tôi
   không biết trình như thế nào để bạn hiểu rõ nhất
32.      // Tốt nhất bạn hãy thay LSBFIRST thành MSBFIRST để rõ ràng
   những gì tôi muốn nói.
33.      }
34.
35.      digitalWrite(latchPin, HIGH);
36.      }
37.
38.      void loop() {
39.      /*
40.      Trong tin học, ngoài các phép +, -, *, / hay % mà bạn đã biết
   trên hệ cơ số 10.
41.      Thì còn có nhiều phép tính khác nữa. Và một trong số đó là Bit
   Math (toán bit) trên hệ cơ số 2.
42.      Để hiểu những gì tôi viết tiếp theo sau, bạn cần có kiến thức
   về Bit Math.
43.      Để tìm hiểu về Bit Math, bạn vào mục Tài liệu tham khảo ở bảng
   chọn nằm phía trên cùng trang web và chạy xuống khi bạn kéo chuột
   trên trang Arduino.VN
44.      */
45.      //Sáng tuần tự
46.
47.      //vì ledStatus là một mảng vì vậy để mặc định tất cả đèn tắt
   thì chúng ta phải for đến từng giá trị của mảng rồi đặt giá trị là
   0.
48.      fillValueToArray(0);
49.      //Bật tuần tự
50.      for (int i = 0; i < HC595_COUNT; i++) {
51.      for (byte j=0;j<8;j++) {
52.      ledStatus[i] = (ledStatus[i] << 1) | 1;
53.      shiftOutHC595(dataPin,clockPin,ledStatus);
54.      delay(100); // Dừng chương trình khoảng 500 mili giây để thấy
   các hiệu ứng của đèn LED
55.      }
56.      }
57.
58.      //Tắt tuần tự
59.      for (int i = 0; i < HC595_COUNT; i++) {
60.      for (byte j=0;j<8;j++) {
61.      ledStatus[i] = (ledStatus[i] << 1);
62.      shiftOutHC595(dataPin,clockPin,ledStatus);
63.      delay(100); // Dừng chương trình khoảng 500 mili giây để thấy
   các hiệu ứng của đèn LED
64.      }
65.      }
```

```
66.
67.      /*
68.      Một số thuật toán nhấp nháy khác, tôi chia sẻ với các bạn đây,
   hãy làm một cái gì đấy tặng người thân yêu mình nhé.
69.      Cộng đồng của chúng ta không chỉ hướng tới Arduino, mà còn
   hướng tới cuộc sống tinh thần của các bạn nữa :)
70.      */
71.
72.      //Nhấp nháy nhanh
73.      for (byte k = 0; k < 20; k++) {
74.      fillValueToArray(0b10101010);
75.      shiftOutHC595(dataPin,clockPin,ledStatus);
76.      delay(50);
77.      fillValueToArray(0b01010101);
78.      shiftOutHC595(dataPin,clockPin,ledStatus);
79.      delay(50);
80.      }
81.
82.      //sáng 1 đèn rồi cách 1 đèn ko sáng rồi lại sáng những đèn chưa
   bật
83.      fillValueToArray(0);
84.      for (int i = 0; i < HC595_COUNT; i++) {
85.      for (byte j = 0;j<8;j += 2) {
86.      ledStatus[i] |= 1 << j;
87.      shiftOutHC595(dataPin,clockPin,ledStatus);
88.      delay(200);
89.      }
90.      }
91.      for (int i = 0; i < HC595_COUNT; i++) {
92.      for (byte j = 1;j<8;j += 2) {
93.      ledStatus[i] |= 1 << j;
94.      shiftOutHC595(dataPin,clockPin,ledStatus);
95.      delay(200);
96.      }
97.      }
98.
99.      //Tắt dần theo thứ tự trên
100.     for (int i = HC595_COUNT - 1; i >= 0; i--) {
101.     for (int j = 7;j >= 0;j -= 2) {
102.     ledStatus[i] &= ~(1 << j);
103.     shiftOutHC595(dataPin,clockPin,ledStatus);
104.     delay(200);
105.     }
106.     }
107.     for (int i = HC595_COUNT - 1; i >= 0; i--) {
108.     for (int j = 6;j >= 0;j -= 2) {
109.     ledStatus[i] &= ~(1 << j);
110.     shiftOutHC595(dataPin,clockPin,ledStatus);
111.     delay(200);
112.     }
113.     }
114.
115.     /// Hãy khám phá thế giới lập trình này nhé :)
116.     }
```

BÀI 3: LẬP TRÌNH GIAO TIẾP LED 7 ĐOẠN

Giới thiệu:

LED 7 đoạn là một loại đèn LED được sắp xếp thành 7 đoạn tách biệt với nhau để hiển thị các số hoặc ký tự. Mỗi đoạn LED 7 đoạn được kết nối với một chân điện và có thể được bật hoặc tắt độc lập để tạo ra các số hoặc ký tự khác nhau. Các LED 7 đoạn thường được sử dụng trong các ứng dụng hiển thị số liệu như đồng hồ số, bộ đếm, hoặc các thiết bị đo lường.

Mục tiêu của bài:

- Hiển thị được thông tin lên các thiết bị ngoại vi
- Phân tích các chương trình có sẵn.
- Lập trình được với led đơn
- Thực hiện các ứng dụng thực tế thông qua các mô hình.
- Rèn luyện tính chính xác, nghiêm túc trong học tập và trong thực hiện công việc.

Nội dung chính:

1. Phương pháp hiển thị led 7 đoạn.

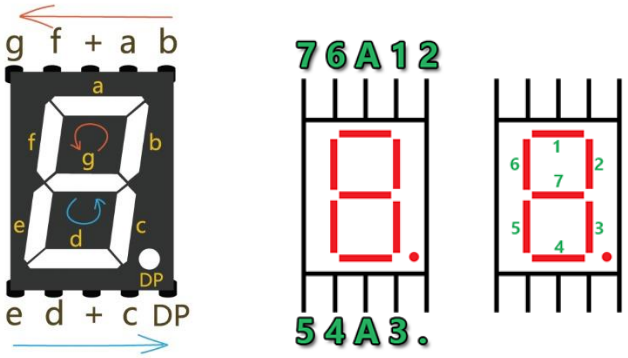
1.1. Cấu trúc Led 7 đoạn.

LED 7 đoạn thường được dùng để hiển thị số, về cấu tạo LED 7 đoạn là các đèn LED được lắp theo thứ tự và hình ảnh giống số 8. Về phương thức hoạt động tương tự như LED bình thường.

Mỗi đoạn tương ứng một đèn LED nên cũng có cực âm và cực dương. Tuy nhiên có 2 loại đèn LED:

Anode các chân LED có chung 1 chân cực dương.

Cathode các chân LED có chung 1 chân cực âm.



Hình 3.1.1: Cấu trúc led 7 đoạn

1.2. Phương pháp quét, phương pháp chốt.

Từ số 1 → 7 tượng trưng cho chân cắm tương đương với vị trí các đèn LED ở hình kế bên.

Dấu . tượng cho dấu chấm trên đèn LED.

Chân A tượng trưng cho chân nguồn của đèn. Tùy vào đây là đèn LED gì mà chân A khác nhau:

Đèn LED Anode: chân A được nối vào VCC.

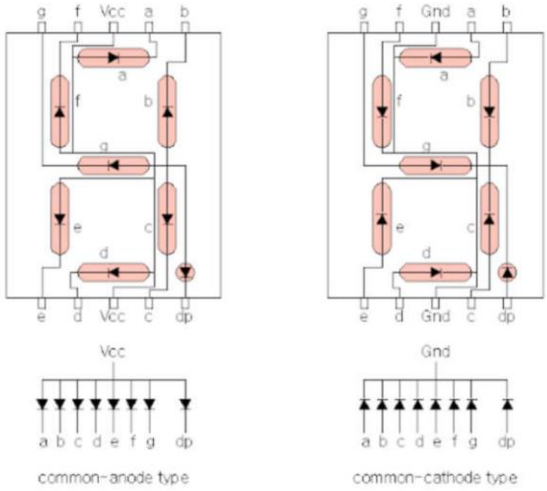
Đèn LED Cathode: chân A được nối vào GND.

Do LED 7 đoạn này là Anode, để cho một trong các đèn LED của “đoạn" đó của LED 7 đoạn sáng, Nối chân của "đoạn" LED đó với GND. Ví dụ muốn để đoạn số 1 của LED 7 đoạn sáng thì cắm chân số 1 như hình trên.

Vì LED 7 đoạn cần nguồn nhỏ tức ~ < 3V, trong khi đó nguồn mà Arduino cung cấp là 5V. Vì thế cần hạ áp dòng trước khi đưa vào đèn, tuy nhiên nếu sử dụng đèn Cathode thì phải nối nguồn 5V cho từng chân đèn LED dẫn đến việc tốn nhiều điện trở. Nếu sử dụng Anode, chỉ cần sử dụng một điện trở để hạ nguồn cho toàn bộ LED.

Led 7 đoạn chung cực dương (Anode chung): Mỗi đèn LED có 2 chân (1 dương 1 âm). Ở loại LED 7 đoạn này tất cả cực dương sẽ được nối chung cực dương. Để làm các đèn LED trong LED 7 đoạn sáng thì chỉ cần cấp cực âm vào các chân của đèn. Với loại LED 7 đoạn này chỉ cần 1 điện trở là đủ.

Chung cực âm (Cathode chung): Tương tự nhưng ngược lại và cần đến 8 điện trở cho các chân dương của LED.



2. Lập trình giao tiếp led 7 đoạn

2.1. Chuẩn bị dụng cụ, thiết bị vật liệu.

Thiết bị, linh kiện	Số lượng
Board Arduino	01
Testboard	01
Dây cắm board	
Led 7 đoạn (chung cực dương)	02
IC HC595 (dành cho việc ShiftOut)	02
Tụ điện 1uf (từ 6,3v trở lên)	01
Điện trở 560 Ohm (hoặc 220 ohm hoặc 1kohm)	01

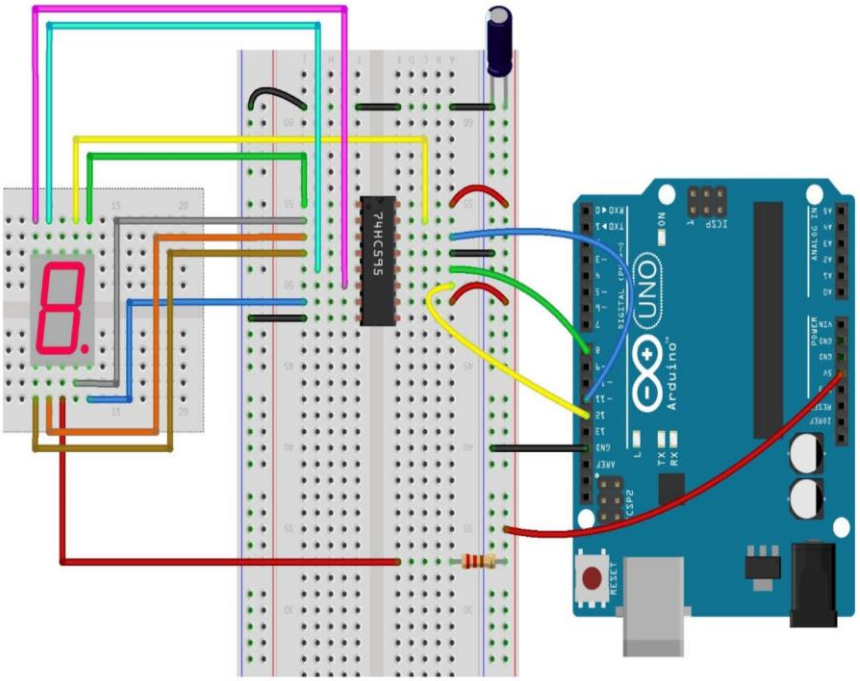
2.2. Lập trình giao tiếp led 7 đoạn.

Để điều khiển được LED 7 đoạn qua IC HC 595, chúng ta phải làm quen với một kỹ thuật, được gọi là shiftOut.

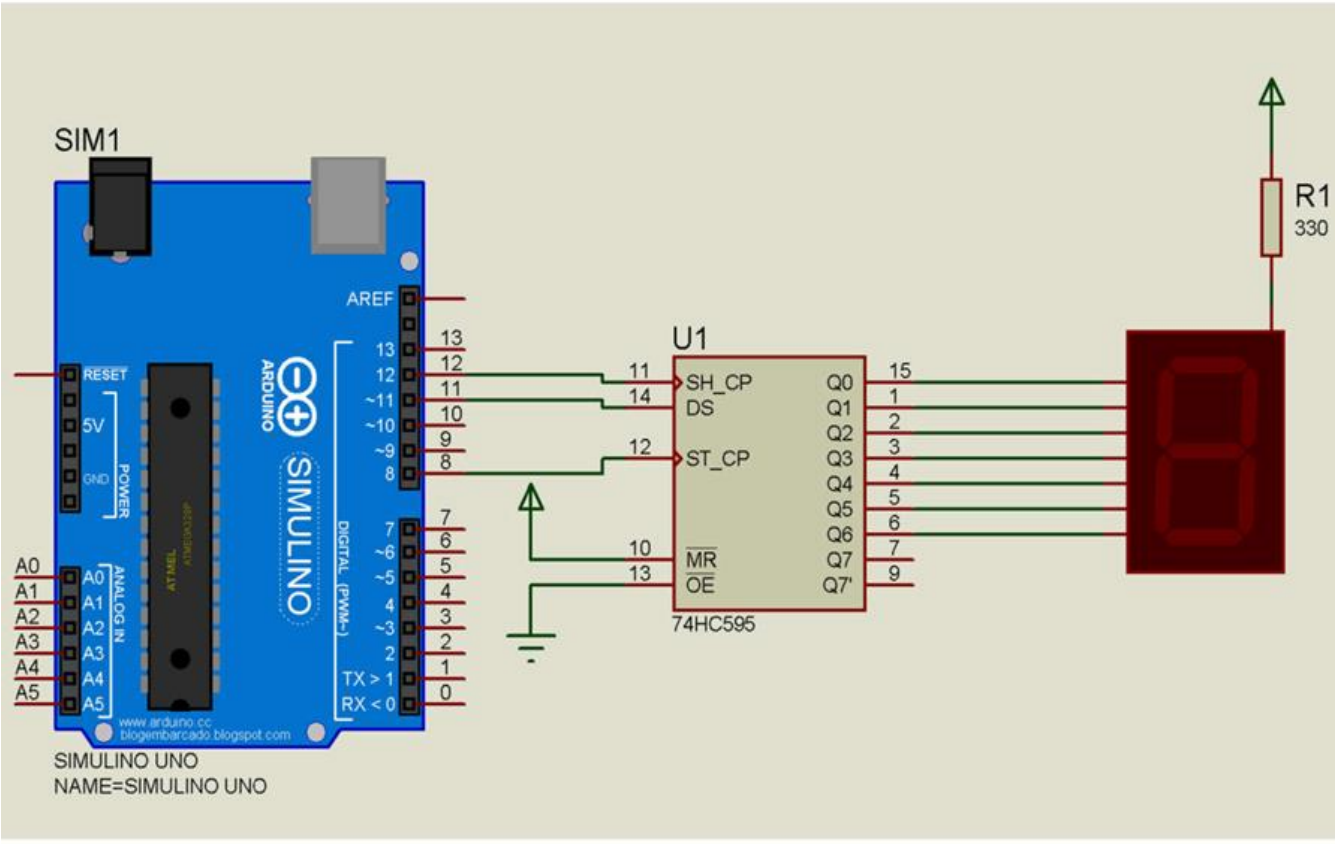
ShiftOut là việc gửi tín hiệu cho 1 IC có hỗ trợ shiftOut (ví dụ IC HC595), cứ mỗi lần gửi nó gửi 1 byte (không hơn không kém), mỗi 1 bit (có tổng cộng 8 bit trong 1 byte) sẽ quản lý giá trị điện tại chân tín hiệu của HC 595 (các chân có tên là Q0-Q7).

```
/*
shiftOut ra 1 Module LED 7 đoạn đơn
*/
//chân ST_CP của 74HC595
int latchPin = 8;
//chân SH_CP của 74HC595
int clockPin = 12;
//Chân DS của 74HC595
int dataPin = 11;
// Ta sẽ xây dựng mảng hằng số với các giá trị cho trước
// Các bit được đánh số thứ tự (0-7) từ phải qua trái (tương ứng với A-F,DP)
// Vì ta dùng LED 7 đoạn chung cực dương nên với các bit 0
// thì các đoạn của LED 7 đoạn sẽ sáng
// với các bit 1 thì đoạn ấy sẽ tắt
//mảng có 10 số (từ 0-9) và
const int Seg[10] = {
0b11000000,//0 - các thanh từ a-f sáng
0b11111001,//1 - chỉ có 2 thanh b,c sáng
0b10100100,//2
0b10110000,//3
0b10011001,//4
0b10010010,//5
0b10000011,//6
0b11111000,//7
0b10000000,//8
0b10010000,//9
};
void setup() {
//Bạn BUỘC PHẢI pinMode các chân này là OUTPUT
pinMode(latchPin, OUTPUT);
pinMode(clockPin, OUTPUT);
pinMode(dataPin, OUTPUT);
}
void loop() {
static int point = 0;
//shiftout - start
digitalWrite(latchPin, LOW);
//Xuất bảng ký tự ra cho Module LED
shiftOut(dataPin, clockPin, MSBFIRST, Seg[point]);
digitalWrite(latchPin, HIGH);
//shiftout - end
point = (point + 1) % 10; // Vòng tuần hoàn từ 0--9
delay(500);//Đợi 0.5 s cho mỗi lần tăng số
}

```

2.3. Kết nối dây dẫn và vận hành mạch.



LED 7 đoạn	HC 595
1 (e)	Q4 (4)
2 (d)	Q3 (3)
4 (c)	Q2 (2)
5 (dp)	Q7 (7)
6 (b)	Q1 (1)
7 (a)	Q0 (15)
9 (f)	Q5 (5)
10 (g)	Q6 (6)



2.4. Hư hỏng thường gặp, nguyên nhân và cách khắc phục.

Khi lập trình Arduino để điều khiển LED 7 đoạn, người lập trình có thể gặp phải một số lỗi phổ biến. Dưới đây là các lỗi thường gặp và cách khắc phục chúng:

STT	Hư hỏng thường gặp	Nguyên nhân	Cách khắc phục
1	Kết nối sai chân (Pin)	Kết nối các chân của LED 7 đoạn với Arduino không đúng thứ tự.	Kiểm tra lại sơ đồ chân của LED 7 đoạn và đảm bảo rằng chúng được kết nối đúng với các chân của Arduino theo đúng sơ đồ.
2	Sai cấu hình kiểu LED (Anode chung/Cathode chung)	Sử dụng kiểu LED 7 đoạn anode chung thay vì cathode chung hoặc ngược lại mà không thay đổi cấu hình trong mã nguồn.	Xác định rõ kiểu LED 7 đoạn đang sử dụng và điều chỉnh mã nguồn cho phù hợp với loại đó.
3	Sai chân điều khiển	Gán sai chân Arduino cho các đoạn LED trong mã nguồn.	Đảm bảo rằng các chân điều khiển của Arduino trong mã nguồn khớp với cách bạn đã kết nối phần cứng.
4	Thiếu điện trở hạn dòng	Không sử dụng điện trở hạn dòng cho các đoạn LED, dẫn đến cháy LED hoặc Arduino.	hêm điện trở hạn dòng (thường là 220 ohm hoặc 330 ohm) vào mỗi đoạn LED để bảo vệ LED và Arduino.
5	Sai logic hiển thị số	Các đoạn LED không sáng theo thứ tự để hiển thị đúng số hoặc ký tự mong muốn.	Kiểm tra lại logic điều khiển các đoạn LED trong mã nguồn và đảm bảo rằng các đoạn sáng lên đúng theo ký tự hoặc số bạn muốn hiển thị.

6	Không sử dụng biến số hoặc mảng đúng cách	Không tổ chức mã nguồn một cách hợp lý, dẫn đến khó khăn trong việc điều khiển nhiều đoạn LED hoặc hiển thị các ký tự khác nhau.	Sử dụng mảng và các biến số để tổ chức mã nguồn dễ dàng hơn
			int segments[] = {2, 3, 4, 5, 6, 7, 8}; // Chân điều khiển các đoạn LED
			int digitPatterns[10][7] = {
			{1, 1, 1, 1, 1, 1, 0}, // Hiển thị số 0
			{0, 1, 1, 0, 0, 0, 0}, // Hiển thị số 1
			//... các số khác
			};
			void displayDigit(int digit) {
			for (int i = 0; i < 7; i++) {
			digitalWrite(segments[i], digitPatterns[digit][i]);
			}
			7}

			void setup() {
			  for (int i = 0; i < 7; i++) {
			  pinMode(segments[i], OUTPUT);
			  }
			}
			void loop() {
			  for (int i = 0; i < 10; i++) {
			  displayDigit(i);
			  delay(1000);
			  }
			}

3. Bài tập mở rộng.

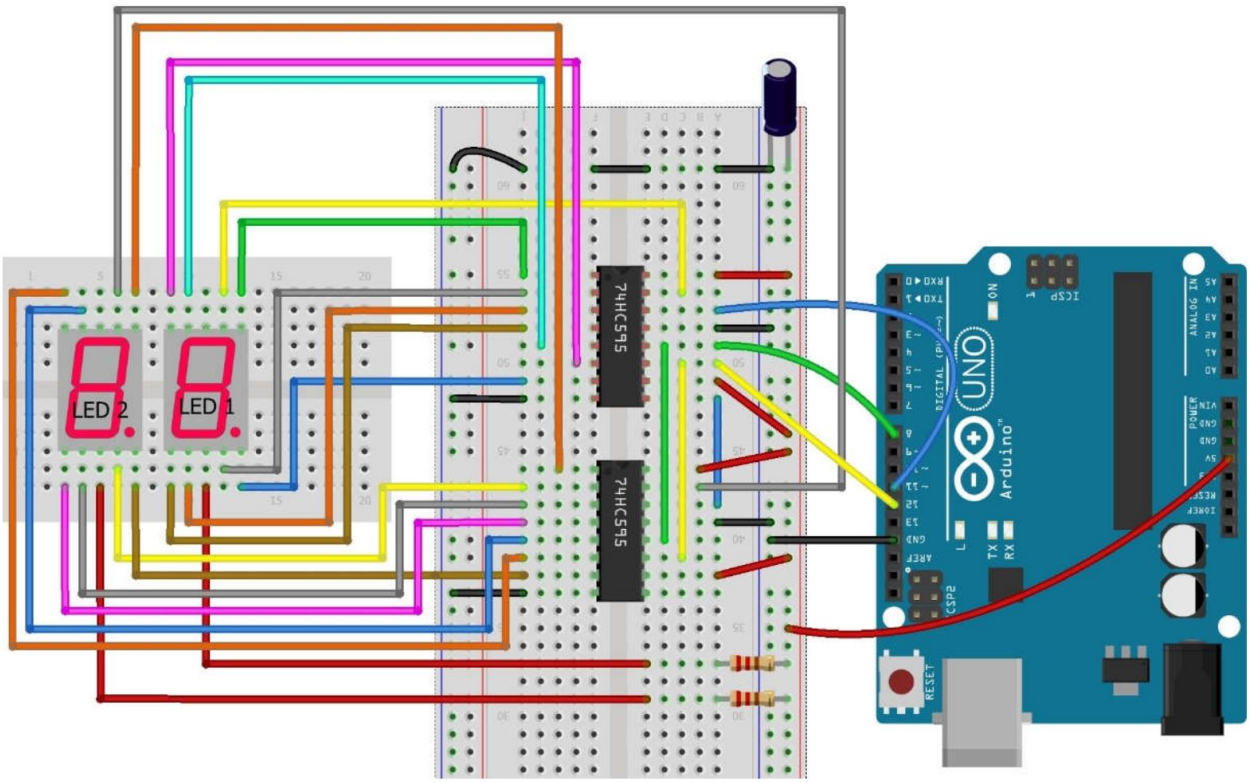
2 Module LED 7 đoạn đơn và nhiều hơn.

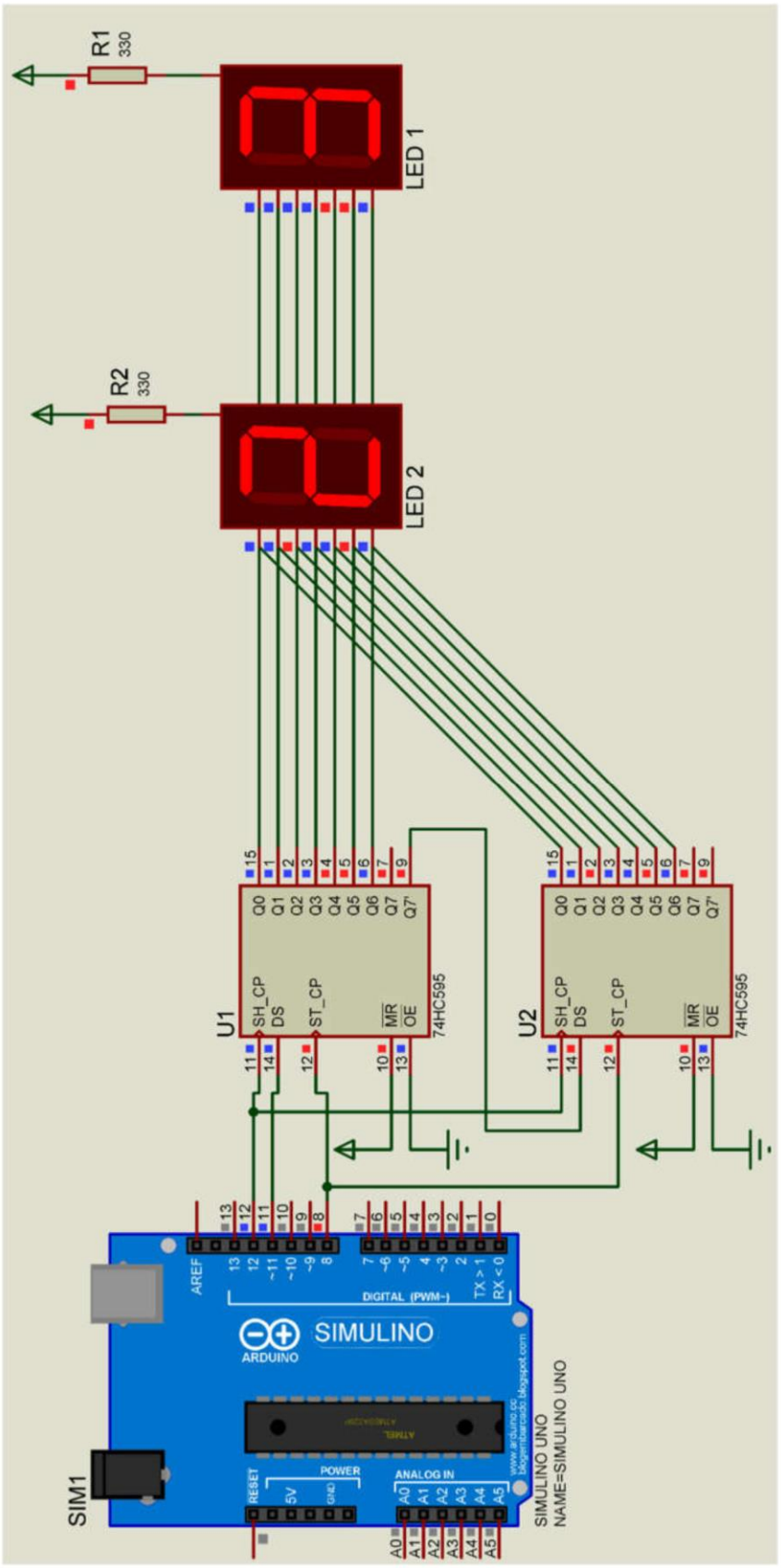
Kể từ IC HC595 thứ 2 trở đi, sẽ lắp như thế này. IC thứ 2 thì mắc vào IC thứ 1, IC thứ 3 thì mắc vào IC thứ 2,... Sau đó xem phần lập trình để xây dựng bảng số.

Lắp LED 7 đoạn thứ 2 và IC HC595 như trên.

Lắp mạch IC HC595 thứ 1 và thứ 2 theo sơ đồ sau (con thứ 3 trở đi thì đấu dây tương tự):

HC 595 thứ 1	HC 595 thứ 2
11	11
12	12
9	14




R1
330
LED 1
R2
330
LED 2
U1
74HC595
U2
74HC595
SIM1
SIMULINO
ARDUINO
SIMULINO UNO
NAME=SIMULINO UNO

```
/*
shiftOut ra 1 Module LED 7 đoạn đơn
*/
//chân ST_CP của 74HC595
int latchPin = 8;
//chân SH_CP của 74HC595
int clockPin = 12;
//Chân DS của 74HC595
int dataPin = 11;

// Ta sẽ xây dựng mảng hằng số với các giá trị cho trước
// Các bit được đánh số thứ tự (0-7) từ phải qua trái (tương ứng với A-F,DP)
// Vì ta dùng LED 7 đoạn chung cực dương nên với các bit 0
// thì các đoạn của LED 7 đoạn sẽ sáng
// với các bit 1 thì đoạn ấy sẽ tắt

//mảng có 10 số (từ 0-9) và
const byte Seg[10] = {
0b11000000,//0 - các thanh từ a-f sáng
0b11111001,//1 - chỉ có 2 thanh b,c sáng
0b10100100,//2
0b10110000,//3
0b10011001,//4
0b10010010,//5
0b10000010,//6
0b11111000,//7
0b10000000,//8
0b10010000,//9
};

void setup() {
//Bạn BUỘC PHẢI pinMode các chân này là OUTPUT
pinMode(latchPin, OUTPUT);
pinMode(clockPin, OUTPUT);
pinMode(dataPin, OUTPUT);
}

void HienThiLED7doan(unsigned long Giatri, byte SoLed = 2) {

byte *array= new byte[SoLed];
for (byte i = 0; i < SoLed; i++) {
//Lấy các chữ số từ phải quá trái
array[i] = (byte)(Giatri % 10UL);
Giatri = (unsigned long)(Giatri /10UL);
}
digitalWrite(latchPin, LOW);
for (int i = SoLed - 1; i >= 0; i--)
shiftOut(dataPin, clockPin, MSBFIRST, Seg[array[i]]);

digitalWrite(latchPin, HIGH);
free(array);
}


void loop() {
static unsigned long point = 0;

HienThiLED7doan(point, 2);

point = (point + 1) % 100UL; // Vòng tuần hoàn từ 0--99
delay(500);//Đợi 0.5 s cho mỗi lần tăng số
}

```

BÀI 4: LẬP TRÌNH SỬ DỤNG TIMER/COUNTER

Giới thiệu:

Trên Arduino, các bộ Timer được phân loại dựa trên bit độ phân giải của chúng (8-bit hoặc 16-bit) và các chức năng mà chúng có thể thực hiện. Dưới đây là phân loại các Timer trên Arduino dựa trên vi điều khiển ATmega328P, được sử dụng trong các board phổ biến như Arduino Uno.

Arduino, đặc biệt là các dòng sử dụng vi điều khiển ATmega328P (như Arduino Uno), có 3 bộ Timer:

Timer 0: 8-bit, được sử dụng cho hàm millis() và delay().

Timer 1: 16-bit, có thể dùng cho PWM và các nhiệm vụ thời gian khác.

Timer 2: 8-bit, có thể dùng cho PWM và các nhiệm vụ thời gian khác.

Mục tiêu của bài:

- Phân tích được số lượng timer/ counter.
- Trình bày được cấu trúc của các thanh ghi cơ bản của các timer/counter
- Giải thích được ứng dụng của timer/counter.
- Thiết kế và lập trình được các mạch ứng dụng
- Kiểm tra và sửa chữa lỗi hư hỏng
- Kết nối được chương trình với mô hình
- Rèn luyện tính chính xác, nghiêm túc trong học tập và trong thực hiện công việc.

Nội dung chính:

1. Timer/counter

1.1. Phân loại các timer?

1. Timer 0

- **Độ phân giải:** 8-bit
- **Chức năng:**
  - Timer 0 được sử dụng cho các hàm thời gian mặc định của Arduino như millis(), micros(), và delay().
  - PWM trên chân 5 và 6 (analogWrite).
- **Đặc điểm:**
  - Bộ Timer này có thể đếm từ 0 đến 255, sau đó quay lại 0.
  - Được sử dụng nhiều trong các hàm thư viện của Arduino, do đó nên hạn chế việc can thiệp trực tiếp để tránh xung đột.

2. Timer 1

- **Độ phân giải:** 16-bit
- **Chức năng:**
  - Timer 1 có khả năng tạo ra các xung PWM trên các chân 9 và 10.

- Được sử dụng cho các ứng dụng yêu cầu độ chính xác cao và khoảng thời gian lớn hơn do có độ phân giải cao hơn (16-bit).

- **Đặc điểm:**
  - Bộ Timer này có thể đếm từ 0 đến 65535, sau đó quay lại 0.
  - Rất linh hoạt và mạnh mẽ, phù hợp cho các ứng dụng cần tính toán thời gian chính xác.

3. Timer 2

- **Độ phân giải:** 8-bit
- **Chức năng:**
  - Timer 2 có thể tạo ra các xung PWM trên các chân 3 và 11.
  - Có thể được sử dụng cho các chức năng thời gian tùy chỉnh khác mà không ảnh hưởng đến các hàm thời gian mặc định của Arduino.
- **Đặc điểm:**
  - Bộ Timer này có thể đếm từ 0 đến 255, sau đó quay lại 0.
  - Thích hợp cho các ứng dụng yêu cầu thời gian chính xác nhưng không yêu cầu độ phân giải cao như Timer 1.

1.2. Các thanh ghi cơ bản của Timer/Counter

Bảng Tổng Hợp Các Timer Trên Arduino Uno (ATmega328P)

Timer	Độ phân giải	PWM Chân	Các hàm sử dụng
Timer 0	8-bit	5, 6	millis(), micros(), delay()
Timer 1	16-bit	9, 10	Có thể tùy chỉnh cho các ứng dụng đặc biệt
Timer 2	8-bit	3, 11	Có thể tùy chỉnh cho các ứng dụng đặc biệt

Lưu ý khi sử dụng Timer

1. **Xung đột giữa các Timer:** Tránh can thiệp trực tiếp vào Timer 0 nếu bạn sử dụng các hàm thời gian mặc định của Arduino.
2. **Sử dụng thư viện:** Có nhiều thư viện giúp làm việc với Timer dễ dàng hơn, như TimerOne và TimerThree cho các bộ Timer 1 và Timer 3 trên các board Arduino khác.
3. **Hiệu chỉnh Timer:** Khi thay đổi cấu hình của Timer, hãy chắc chắn rằng các thay đổi này không ảnh hưởng đến các chức năng khác của chương trình.

2. Lập trình sử dụng timer/ counter

2.1. Chuẩn bị dụng cụ, thiết bị vật liệu.

1 x Arduino Uno
1 x Breadboard
3 x Đèn LED (Đỏ, Vàng, Xanh)
3 x Điện trở 220Ω
Dây kết nối

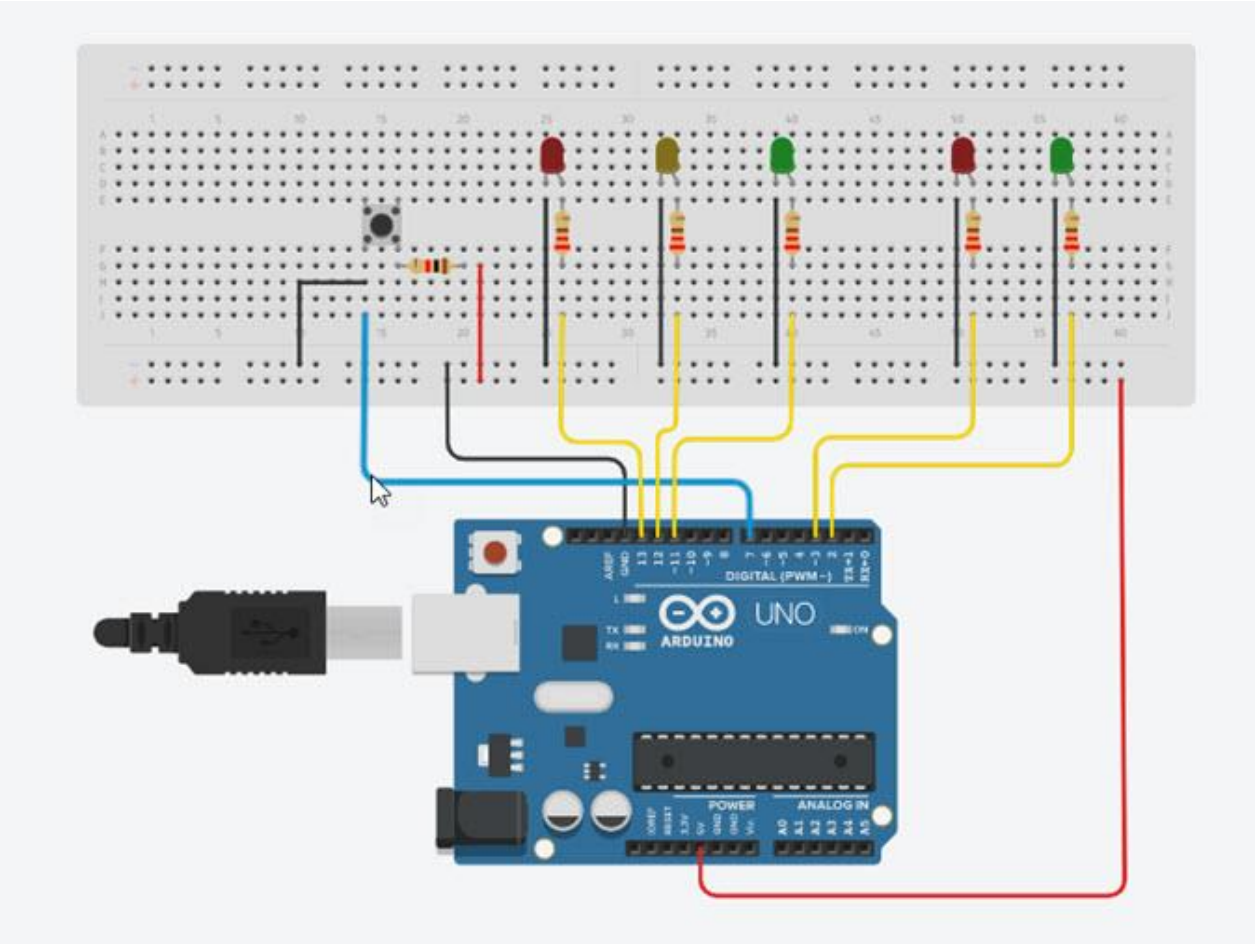
2.2. Lập trình sử dụng timer/counter

Kết Nối LED:

- Kết nối chân dương (anode) của LED đỏ đến chân số 2 của Arduino qua một điện trở 220Ω.
- Kết nối chân dương (anode) của LED vàng đến chân số 3 của Arduino qua một điện trở 220Ω.
- Kết nối chân dương (anode) của LED xanh đến chân số 4 của Arduino qua một điện trở 220Ω.
- Kết nối chân âm (cathode) của các LED với GND của Arduino.

```
// Định nghĩa chân cho các LED
const int redLED = 2;
const int yellowLED = 3;
const int greenLED = 4;

void setup() {
  // Thiết lập các chân là OUTPUT
  pinMode(redLED, OUTPUT);
  pinMode(yellowLED, OUTPUT);
  pinMode(greenLED, OUTPUT);
}

void loop() {
  // Bật đèn đỏ
  digitalWrite(redLED, HIGH);
  delay(5000); // Đèn đỏ sáng trong 5 giây

  // Tắt đèn đỏ, bật đèn xanh
  digitalWrite(redLED, LOW);
  digitalWrite(greenLED, HIGH);
  delay(5000); // Đèn xanh sáng trong 5 giây

  // Tắt đèn xanh, bật đèn vàng
  digitalWrite(greenLED, LOW);
  digitalWrite(yellowLED, HIGH);
  delay(2000); // Đèn vàng sáng trong 2 giây

  // Tắt đèn vàng, chu kỳ lặp lại
  digitalWrite(yellowLED, LOW);
}
```

2.3. Kết nối dây dẫn và vận hành mạch



1. Khai báo các chân:

```
const int redLED = 2;
const int yellowLED = 3;
const int greenLED = 4;
```

• Khai báo các chân số 2, 3 và 4 để kết nối với LED đỏ, vàng và xanh tương ứng.

2. Hàm setup():

```
void setup() {
  pinMode(redLED, OUTPUT);
  pinMode(yellowLED, OUTPUT);
  pinMode(greenLED, OUTPUT);
}
```

• Thiết lập các chân LED làm ngõ ra (OUTPUT).

3. Hàm loop():

```
void setup() {
```

```
void loop() {
  digitalWrite(redLED, HIGH);
  delay(5000); // Đèn đỏ sáng trong 5 giây

  digitalWrite(redLED, LOW);
  digitalWrite(greenLED, HIGH);
  delay(5000); // Đèn xanh sáng trong 5 giây

  digitalWrite(greenLED, LOW);
  digitalWrite(yellowLED, HIGH);
  delay(2000); // Đèn vàng sáng trong 2 giây

  digitalWrite(yellowLED, LOW);
}
```

• Trong hàm loop(), đèn đỏ bật sáng trong 5 giây, sau đó tắt và bật đèn xanh trong 5 giây, tiếp đó tắt đèn xanh và bật đèn vàng trong 2 giây. Cuối cùng, tắt đèn vàng và lặp lại chu kỳ.

2.4. Hư hỏng thường gặp, nguyên nhân và cách khắc phục

STT	Hư hỏng thường gặp	Nguyên nhân	Cách khắc phục
1	Đèn LED không thay đổi trạng thái	Kết nối chân của Arduino không đúng với chân được định nghĩa trong mã.	Kiểm tra lại kết nối chân của Arduino với các chân của đèn LED theo đúng định nghĩa trong mã.
2	Lỗi Lập Trình	Sai sót trong mã lập trình hoặc logic lập trình không đúng	Xem lại mã lập trình để đảm bảo logic đúng. Sử dụng các công cụ gỡ lỗi như Serial.print() để kiểm tra các biến và trạng thái trong chương trình.
3	Lỗi Ngắt Kết Nối	Ngắt kết nối không được thiết lập đúng hoặc chân không đúng.	Kiểm tra lại thiết lập ngắt kết nối trong mã và đảm bảo sử dụng đúng chân ngắt kết nối trên Arduino.

BÀI 5: LẬP TRÌNH NÚT NHẤN

Giới thiệu:

Giống với công tắc đóng / mở bạn thấy ở bất cứ đâu, nút nhấn cũng có cơ chế hoạt động giống như vậy. Thay vì chỉ có 2 chân như công tắc, nút nhấn có 4 chân chia làm 2 cặp. Những chân trong cùng một cặp được nối với nhau, những chân khác cặp thì ngược lại. Khi bạn nhấn nút, cả 4 chân của nút nhấn đều được nối với nhau, cho phép dòng điện từ một chân bất kì có thể tới 3 chân còn lại, khi ngừng nhấn, 2 cặp sẽ tách dời, dòng điện sẽ không còn liên thông nữa.

Mục tiêu của bài:

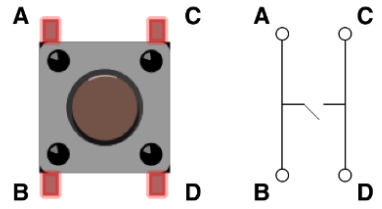
- Trình bày được cấu trúc của nút nhấn
- Thiết kế và lập trình được các mạch ứng dụng sử dụng nút nhấn
- Kiểm tra và sửa chữa lỗi hư hỏng
- Kết nối được chương trình với mô hình
- Phát huy tính chủ động trong học tập và trong công việc.

Nội dung chính:

1. Nút nhấn.

Button (thường) (6mm hoặc 12mm). Đây là loại button rất phổ biến, cũng như đèn LED, loại button này cũng có các kính thước cạnh 6mm hoặc 12m. Loại 6mm hay được dùng trong các dự án nhỏ và loại còn lại dùng cho các dự án bự hơn và cần nút to. Giá thành thì khá rẻ, loại 6mm có giá khoảng 1500 dồng và 2500 đồng cho loại 12mm.

Loại này tuy là 4 chân, nhưng thực chất cũng chỉ là 2 chân



Hình 5.1.1: Nút nhấn 6mm và Nút nhấn 12mm

2. Lập trình nút nhấn.

2.1. Chuẩn bị dụng cụ, thiết bị vật liệu.

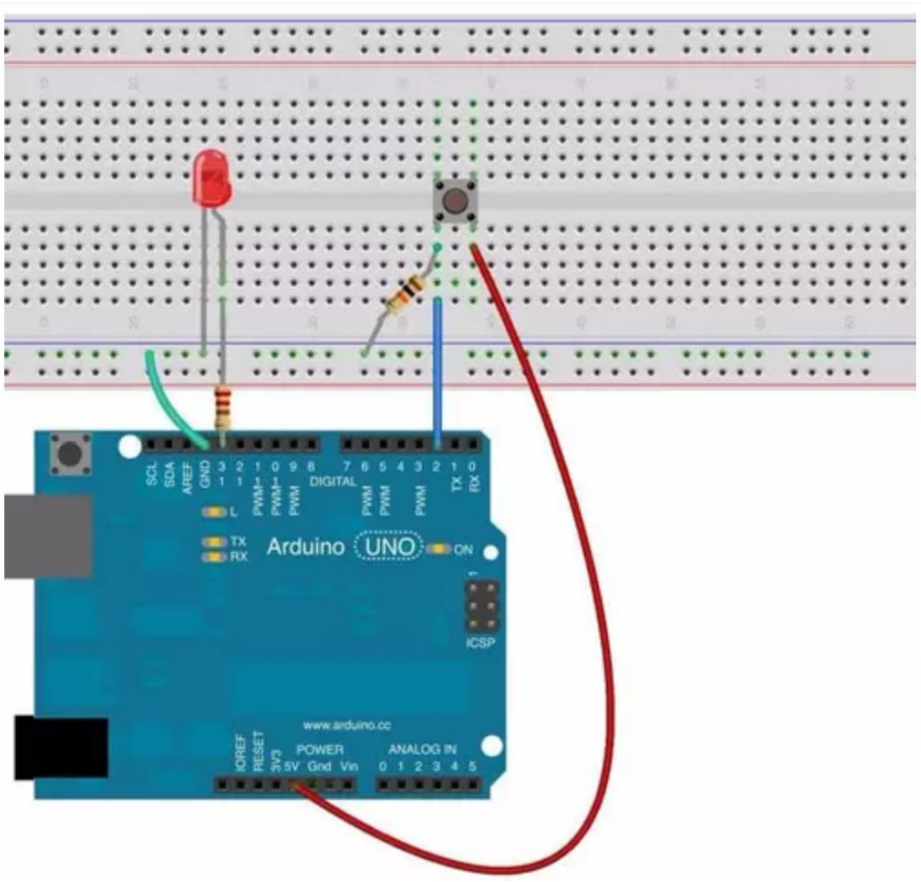
- Arduino Uno
- Breadboard
- Dây cắm breadboard
- 1 điện trở 560 Ohm (hoặc 220 Ohm hoặc 1kOhm)

- 1 điện trở 10 kOhm
- 1 đèn LED siêu sáng
- 1 button (nút nhấn)

2.2. Lập trình nút nhấn.

```
const int buttonPin = 2;     // pin nối button để điều khiển
const int ledPin =  13;      // pin nối LED

// Tạo một biến nhận diện trạng thái button:
int buttonState = 0;

void setup() {
  // set ledPin là output
  pinMode(ledPin, OUTPUT);
  // set buttonPin là input để đọc giá trị từ button
  pinMode(buttonPin, INPUT);
}

void loop() {
  // đọc giá trị button rồi lưu vào buttonState
  buttonState = digitalRead(buttonPin);

  // nếu button được nhấn, buttonState nhận giá trị HIGH, và ngược lại
  if (buttonState == HIGH) {
    // Bật LED:
    digitalWrite(ledPin, HIGH);
  } else {
    // Tắt LED:
    digitalWrite(ledPin, LOW);
  }
}
}
```

2.3. Kết nối dây dẫn và vận hành mạch



Trên sơ đồ lặp đặt trên đấu chân âm của đèn LED với chân GND của mạch, chân dương LED nối với chân pin 13 thông qua một điện trở.

Với button có 4 chân đều như nhau, quan trọng là cặp button, chọn 1 chân bất kỳ nối với GND qua trở và nối với pin 2 trực tiếp, sau đó chọn 1 chân ở cặp còn lại nối với nguồn 5V

Vẫn lưu ý đấu nối với Arduino, có thể dùng bất kỳ chân nào khác từ 0 đến 13 đều được không có gì khác biệt cả, chỉ khác khi lập trình.

Sau khi đấu nối xong, sử dụng dây USB để kết nối Arduino với máy tính để tiến hành nạp code, hoặc nếu thích thì nạp code cho Arduino rồi đấu nối sau)

2.4. Hư hỏng thường gặp, nguyên nhân và cách khắc phục.

STT	Hư hỏng thường gặp	Nguyên nhân	Cách khắc phục
1	Nút nhấn không hoạt động	Kết nối không đúng: Chân của nút nhấn không được kết nối đúng cách với Arduino hoặc mạch điện. Kết nối GND sai: Nút nhấn không được kết nối đúng cách với chân GND.	Kiểm tra lại kết nối: Đảm bảo rằng một chân của nút nhấn được kết nối với một chân số của Arduino và chân còn lại được kết nối với GND. Sử dụng điện trở kéo: Nếu nút nhấn không hoạt động ổn định, sử dụng điện trở kéo (pull-down resistor hoặc pull-up

			resistor) để đảm bảo mức tín hiệu ổn định. Arduino có thể sử dụng điện trở kéo tích hợp bằng cách cấu hình trong mã lập trình.

2	Arduino không phản ứng khi nhấn nút	Mã lập trình sai: Mã lập trình không đọc đúng chân số hoặc không xử lý đúng sự kiện nhấn nút.
		Cấu hình chân không đúng: Chân số của Arduino không được cấu hình đúng cách.
			Kiểm tra mã lập trình: Đảm bảo rằng chân số được định nghĩa đúng và có mã để xử lý sự kiện nhấn nút.
			Sử dụng Serial Monitor: Sử dụng Serial.print() để kiểm tra xem mã có đọc đúng trạng thái của nút nhấn hay không.
			`const int buttonPin = 2; // Chân số kết nối nút nhấn`
			
			`void setup() {`
			`pinMode(buttonPin, INPUT_PULLUP); // Sử dụng điện trở kéo tích hợp`
			`Serial.begin(9600);`
			`}`
			
			`void loop() {`
			`int buttonState = digitalRead(buttonPin);`
			`if (buttonState == LOW) {`
			`Serial.println("Button pressed");`
			`} else {`
			`Serial.println("Button not pressed");`
			`}`
			`delay(100); // Tránh in quá nhiều thông tin`
			`}`

3	Nút nhấn hoạt động không ổn định	Nút nhấn bị lỗi: Nút nhấn có thể bị hỏng hoặc không hoạt động đúng cách. Tiếp xúc kém: Các chân của nút nhấn hoặc dây kết nối không tiếp xúc tốt.	Thay thế nút nhấn: Thử sử dụng một nút nhấn khác để kiểm tra. Kiểm tra kết nối: Đảm bảo rằng các dây kết nối và chân của nút nhấn đều được kết nối chắc chắn.

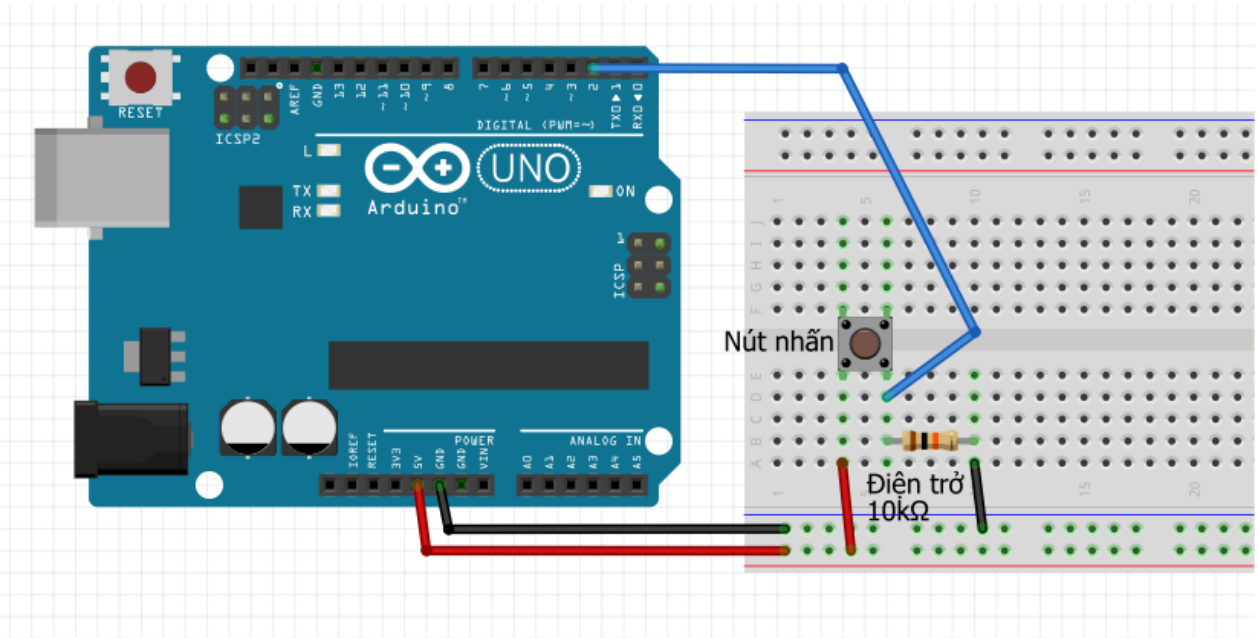
Bài tập

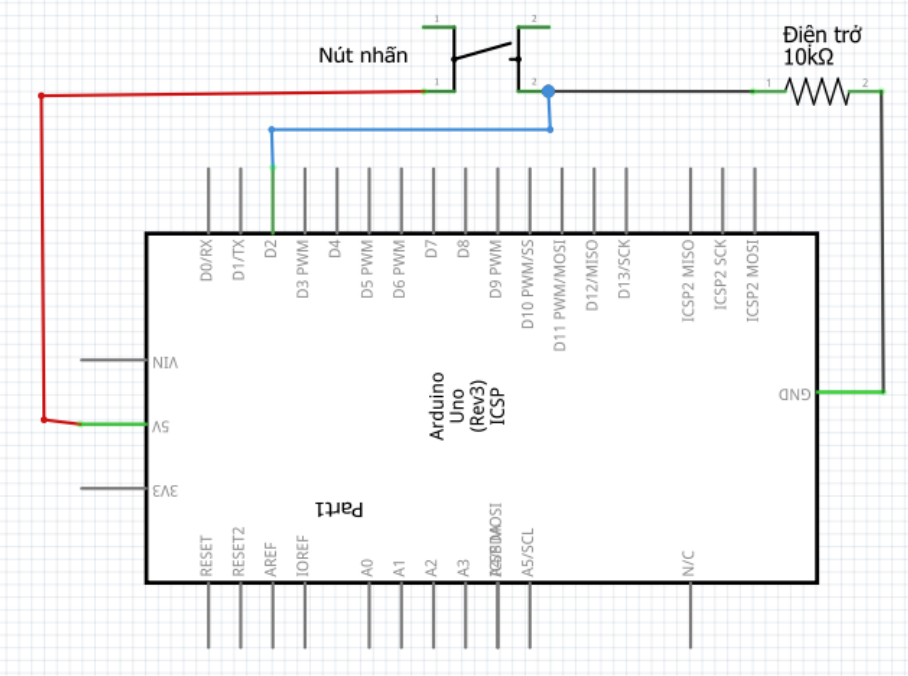
1. Đếm số lần nhấn một button – ButtonStateChange

```
// dùng const đặt trước khi khai báo biến sẽ coi biến này là một hằng số
// Bạn có thể đọc được giá trị biến hoặc thực hiện các phép toán
// Nhưng không thay đổi được giá trị của hằng số này.
const int buttonPin = 11;// hằng số buttonPin mang giá trị là chân digital được nối với button
const int ledPin = 2; // hằng số ledPin mang giá trị là chân digital đươc nối với led

// Các biến này có thể thay đổi giá trị được
int buttonPushCounter = 0; // số lần button được nhấn
int buttonState = 0; // trạng thái hiện tại của button
int lastButtonState = 0; // trạng thái trước đó của button

void setup() {
pinMode(buttonPin, INPUT); // Cài đặt button là INPUT
pinMode(ledPin, OUTPUT); // Cài đặt đèn LED là OUTPUT

Serial.begin(9600); //Bật cổng Serial ở baudrate 9600
}


void loop() {
// đọc giá trị hiện tại của button
buttonState = digitalRead(buttonPin);

// so sánh với giá trị trước đó
if (buttonState != lastButtonState) {
if (buttonState == HIGH) {
// Nếu trạng thái bây giờ là button đang được nhấn
// thì hiển nhiên trước đó là button chưa được nhấn (điều kiện trên)
// chúng ta sẽ tăng số lần nhấn button lên 1
buttonPushCounter++;
Serial.println("Dang nhan");
Serial.print("So lan nhan button la: ");
Serial.println(buttonPushCounter);
}
else {
// Nếu trạng thái bây giờ là button đang được THẢ
// thì hiển nhiên trước đó là button đang được nhấn (điều kiện trên)
// Chúng ta sẽ thông báo là button đang được thả và không làm gì cả
Serial.println("off");
}
}
// lưu lại trạng thái button cho lần kiểm tra tiếp theo
lastButtonState = buttonState;

```

```
// Đã đếm được số lần nhấn button, bây giờ sẽ là phần sau bao nhiêu
// lần thì button sẽ làm đèn sáng
// Trong ngôn ngữ lập trình Arduino, chúng ta có thêm một phép toán mới
// đó là phép chia lấy dư (khác với các phép +, -, *, / được học trong trường
// phép này sẽ trả về phần dư của một phép chia.
// Ví dụ: 6 % 4 = 2 (% là toán tử) (vì 6 chia 4 = 1 dư 2). Xem thêm tại:
    http://arduino.vn/reference/modulo
// Áp dụng:
// Chẳng hạn, bạn làm button này cứ sau 4 lần bấm sẽ làm đèn led sáng vì bạn làm như
    sau:
// Mới upload code thì đèn sáng do buttonPushCounter = 0. 0 % 4 = 0
// Sau đó cứ mỗi lần nhấn nút thì buttonPushCounter được tăng lên.
//... 1 % 4 = 1 --> tắt
//... 2 % 4 = 2 --> tắt
//... 3 % 4 = 3 --> tắt
//... 4 % 4 = 0 --> bật
//... 5 % 4 = 1 --> tắt
//...
if (buttonPushCounter % 4 == 0) {
digitalWrite(ledPin, HIGH);
Serial.println("Da bat den");
} else {
digitalWrite(ledPin, LOW);
}

}
```

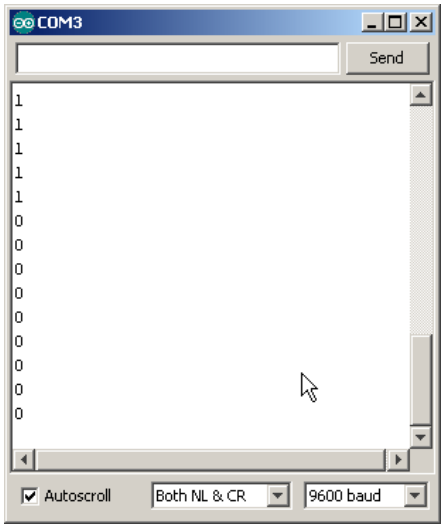
2. Xác định trạng thái của một nút nhấn (button)






```
int button = 2;
void setup() {
  Serial.begin(9600);   //Mở cổng Serial ở baudrate 9600 để giao tiếp với máy tính
  pinMode(button, INPUT);  //Cài đặt chân D2 ở trạng thái đọc dữ liệu
}
void loop() {
  int buttonStatus = digitalRead(button);    //Đọc trạng thái button
  Serial.println(buttonStatus);              //Xuất trạng thái button
  delay(200);                                //Chờ 200ms
}
```

Sau khi upload code lên mạch Arduino, bạn bấm Ctrl + Shift + M để mở cửa sổ Serial Monitor để xem trạng thái button được mạch Arduino gửi về máy tính.



BÀI 6: LẬP TRÌNH ĐIỀU KHIỂN ĐỘNG CƠ

Giới thiệu:

Động cơ bước là một loại động cơ mà ở đó bạn sẽ có thể quy định chính xác số góc quay và động cơ bước sẽ phải quay. Không giống như Servo, động cơ bước có thể quay bao nhiêu độ tùy ý và mỗi lần quay nó sẽ quay được 1 step, 1 step ở đây là bao nhiêu còn phụ thuộc vào động cơ bước của bạn. Ví dụ, động cơ bước của bạn có 72 step thì nó sẽ cần quay 72 step để hoàn thành một vòng quay. Số step này là hằng số, nhưng bạn có thể dùng công nghệ micro step để "cải thiện" số vòng quay động cơ bước

Mục tiêu của bài:

- Trình bày được cấu trúc của động cơ
- Thiết kế và lập trình được các mạch ứng dụng động cơ
- Kiểm tra và sửa chữa lỗi hư hỏng
- Kết nối được chương trình với mô hình
- Phát huy tính chủ động trong học tập và trong công việc.

Nội dung chính:

1. PWM.

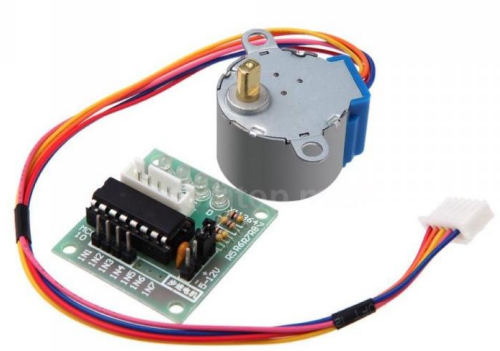
Động cơ bước (stepper motor) là một loại động cơ điện có khả năng xoay một góc cố định (bước) mỗi lần nhận một xung điều khiển từ hệ thống điều khiển. Động cơ này được điều khiển bằng cách đưa các xung điều khiển vào các dây cuộn của động cơ để tạo ra từng bước xoay.

Các đặc điểm chính của động cơ bước bao gồm:

Bước (Step): Là góc quay nhỏ nhất mà động cơ có thể xoay được khi nhận một xung điều khiển.

Tốc độ (Speed): Tốc độ quay của động cơ được điều khiển bằng cách thay đổi tần số của xung điều khiển.

Điều kiện đứng yên (Holding Torque): Là mô-men xoắn tối đa mà động cơ có thể cung cấp khi không có xung điều khiển.



Hình 6.1.1: Động cơ bước

Driver A4988 là một trong những driver phổ biến được sử dụng để điều khiển động cơ bước. Nó có khả năng cung cấp các xung điều khiển và dòng điện cần thiết để hoạt động động cơ bước. Các đặc điểm của A4988 bao gồm:

Dòng Điều Khiển: Có thể điều chỉnh được dòng điều khiển đầu ra cho phù hợp với động cơ sử dụng.

Tích Hợp Bộ Chia Bước: Có thể cấu hình để điều khiển số bước một vòng quay của động cơ bước.

Bảo Vệ Quá Dòng: Tích hợp chức năng bảo vệ quá dòng để bảo vệ driver và động cơ khỏi hỏng hóc.

Giao Tiếp Dễ Dàng: Có các chân giao tiếp dễ dàng kết nối với vi điều khiển như Arduino.

Rated Current/phase (dòng tiêu thụ tối đa mỗi pha)	2.0A
Phase Resistance (điện trở từng pha)	1.4ohms
Voltage (hiệu điện thế)	2.8V

2. Lập trình điều khiển động cơ.

2.1. Chuẩn bị dụng cụ, thiết bị vật liệu.

Arduino UNO hoặc mạch Arudino nào cũng được ha.

Driver điều khiển động cơ A4988 hoặc DRV8825.

Breadboard

Nguồn cấp 12V (tối thiểu 1A).

Dây breadboard.

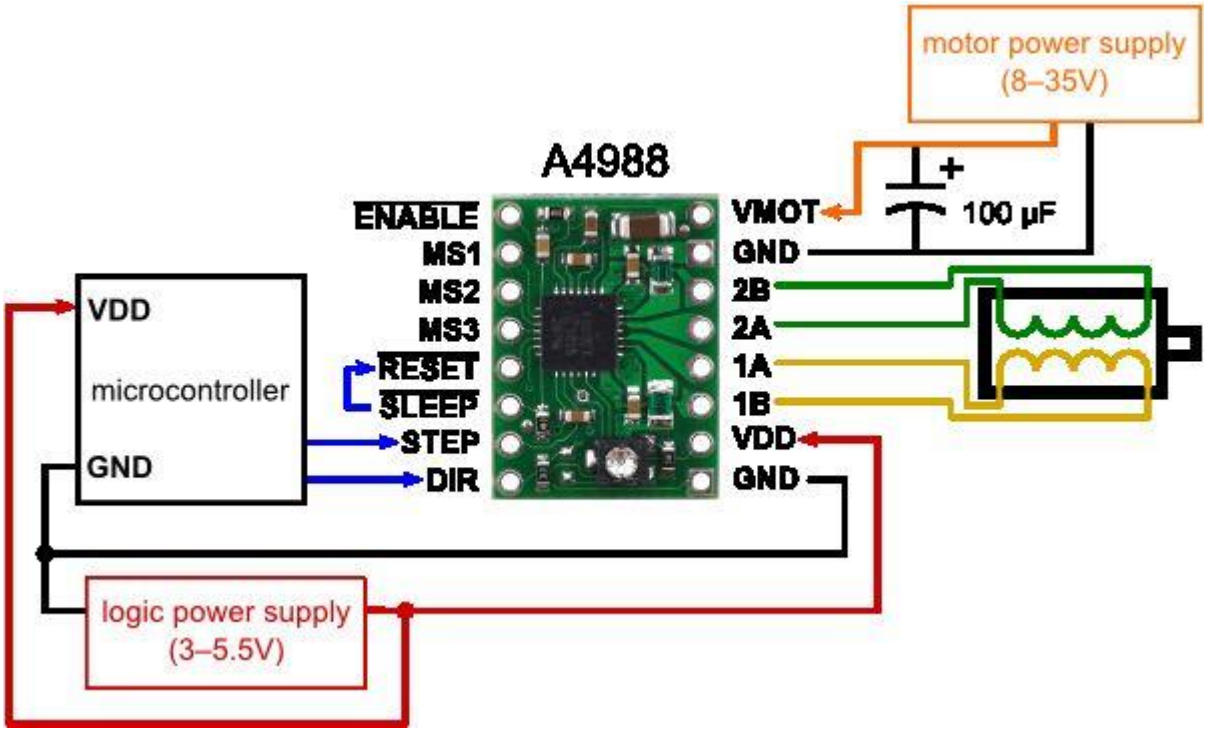
2.2. Lập trình điều khiển động cơ.

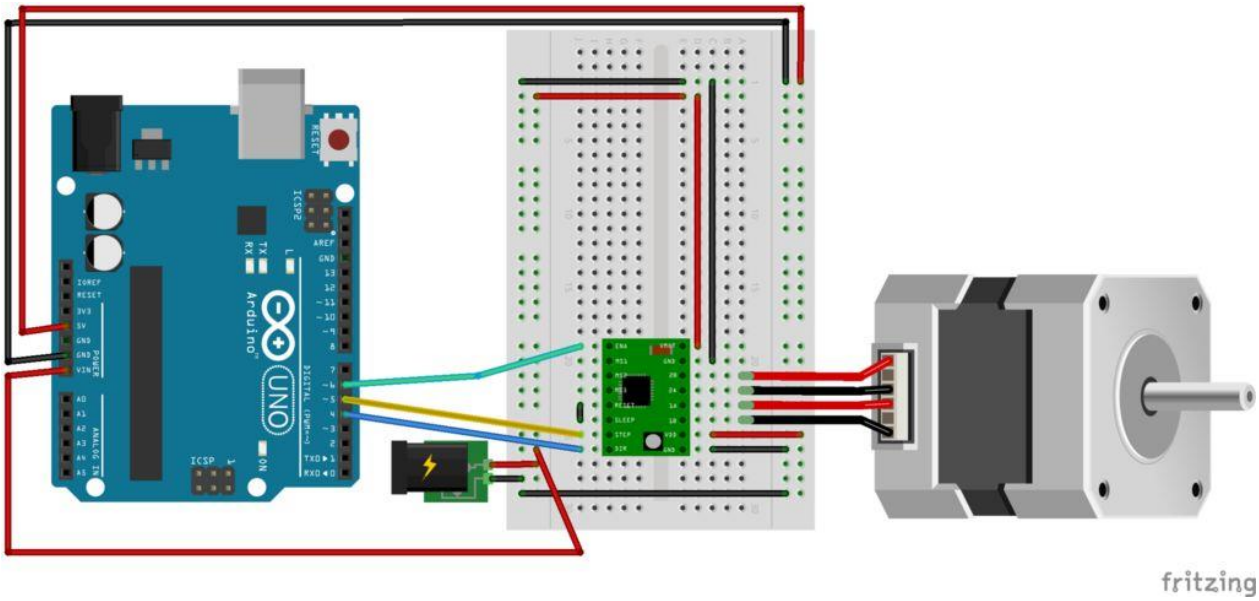
```
// Run a A4998 Stepstick from an Arduino UNO.
// Paul Hurley Aug 2015 - http://www.instructables.com/id/Drive-a-Stepper-Motor-with-an-Arduino-and-a-A4988-/
int x;
#define BAUD (9600)


void setup()
{
  Serial.begin(BAUD);
  pinMode(6,OUTPUT); // Enable pin - chân khởi động - nối vào GND sẽ giúp ta bật động cơ bước, nối vô VCC động cơ bước được thả ra. Nôm na: GND = servo.attach, VCC = servo.detach
  pinMode(5,OUTPUT); // Step pin
  pinMode(4,OUTPUT); // Dir - pin
```

```
  digitalWrite(6,LOW); // Set Enable low
}

void loop()
{
  digitalWrite(6,LOW); // Đặt Enable ở trạng thái LOW
  digitalWrite(4,HIGH); // Đặt Dir  ở trạng thái HIGH
  Serial.println("Cho chay 200 steps (1 vong)");
  for(x = 0; x < 200; x++) // Cho chay 1 vong
  {
    digitalWrite(5,HIGH); // Output high
    delay(10); // chờ
    digitalWrite(5,LOW); // Output low
    delay(100); // chờ
  }
  Serial.println("Pause");
  delay(1000); // dừng 1 s rồi quay tiếp
}
```

Nhớ gắn điện 12V

2.3. Kết nối dây dẫn và vận hành mạch.





Tăng sức mạnh cho động cơ bước bằng vi bước (micro stepping)

Để bật chức năng này (micro stepping - vi bước), phải nối ba chân MS1, MS2, MS3 vào VCC!

Nói một cách nôm na, cứ mỗi chân MS1, MS2, MS3 được nối vô VCC, nó sẽ tăng số bước của động cơ bước lên. Nghĩa là thay vì chỉ cần quay 200 bước là đươc một vòng. Bây giờ sẽ phải quay nhiều hơn, lúc đó sẽ có độ chính xác cao hơn!

MS1	MS2	MS3	Vi bước
Không nối	Không nối	Không nối	1
VCC	Không nối	Không nối	1/2
Không nối	VCC	Không nối	1/4
VCC	VCC	Không nối	1/8
VCC	VCC	VCC	1/16

Thứ nối MS1, MS2, MS3, ta sẽ đươc vi bước 1/16, nghĩa là với động cơ bước có 200 bước thì ta sẽ chia nhỏ mỗi bước ra thành 16 bước => Tổng số bước của động cơ lúc này là 200 * 16 = 3200.

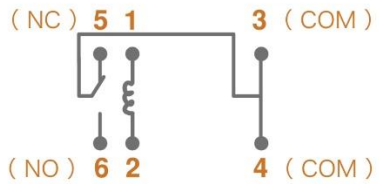
2.4. Hư hỏng thường gặp, nguyên nhân và cách khắc phục.

STT	Hư hỏng thường gặp	Nguyên nhân	Cách khắc phục
1	Động cơ không hoạt động	Nguồn cấp không đủ: Động cơ bước yêu cầu dòng điện và điện áp nhất định, nếu không đủ, động cơ sẽ không hoạt động.	Kiểm tra nguồn cấp: Đảm bảo rằng nguồn cấp điện đủ dòng và điện áp cho động cơ. Kiểm tra thông số kỹ thuật của động cơ và sử dụng nguồn phù hợp.

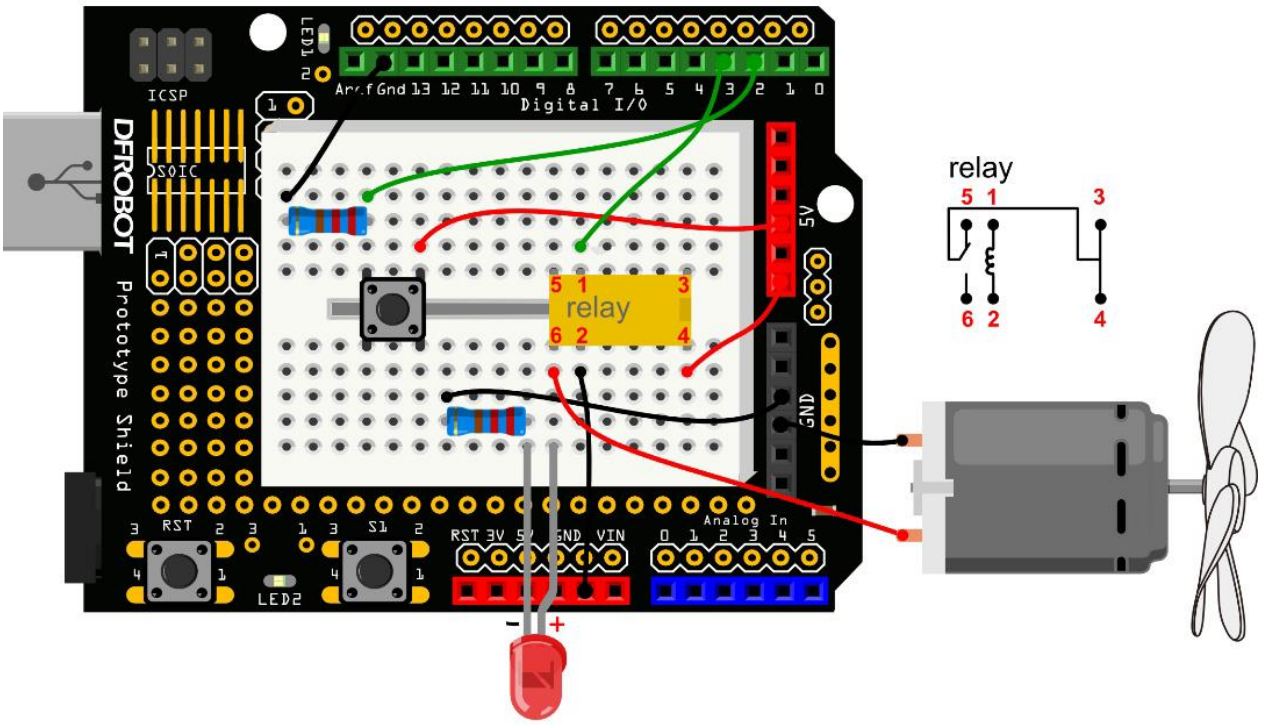
		Kết nối không đúng: Các dây của động cơ không được kết nối đúng cách với driver hoặc Arduino. Lỗi driver động cơ: Driver có thể bị hỏng hoặc không được kết nối đúng cách.	Kiểm tra kết nối: Đảm bảo rằng các dây của động cơ được kết nối đúng cách với driver và các chân của driver được kết nối đúng cách với Arduino. Kiểm tra driver: Thử sử dụng một driver khác để kiểm tra nếu driver hiện tại bị hỏng.
2	Động cơ bị rung hoặc không quay đúng hướng	Chuỗi xung điều khiển không đúng: Tín hiệu điều khiển từ Arduino không đúng tần số hoặc thứ tự. Kết nối cuộn dây sai: Các cuộn dây của động cơ không được kết nối đúng cách. Driver bị lỗi: Driver có thể không hoạt động đúng cách hoặc không tương thích với động cơ.	Kiểm tra mã lập trình: Đảm bảo rằng mã lập trình gửi tín hiệu điều khiển đúng thứ tự và tần số cho driver động cơ. Kiểm tra kết nối cuộn dây: Xác định và kết nối đúng các cuộn dây của động cơ bước. Sử dụng datasheet của động cơ để xác định các chân của cuộn dây. Thay thế driver: Thử sử dụng driver khác nếu driver hiện tại không hoạt động đúng cách.
3	Động cơ bị mất bước	Quá tải: Động cơ bị quá tải do tải quá nặng hoặc tốc độ quá cao. Điều khiển không chính xác: Tín hiệu điều khiển không chính xác hoặc không đủ độ chính xác.	Giảm tải: Giảm tải trên động cơ hoặc giảm tốc độ hoạt động. Cải thiện điều khiển: Sử dụng mã lập trình với độ chính xác cao hơn và tối ưu hóa tần số điều khiển.

Bài tập.

Điều khiển động cơ quạt bằng nút nhấn sử dụng Arduino.



Cấu tạo delay



Tên linh kiện	**Số lượng**
Arduino Uno R3	1
Dây cắm	9
Breadboard	1
Button	1
LED 5mm	1
Trở 220R	2
Relay	1
Motor	1
Fan	1

```
int buttonPin = 2;
int relayPin = 3;
int relayState = HIGH;
int buttonState;
int lastButtonState = LOW;
long lastDebounceTime = 0;
long debounceDelay = 50;

void setup() {
  pinMode(buttonPin, INPUT);
  pinMode(relayPin, OUTPUT);

  digitalWrite(relayPin, relayState);
}
void loop() {

  int reading = digitalRead(buttonPin);

  if (reading != lastButtonState) {
    lastDebounceTime = millis();
  }

  if ((millis() - lastDebounceTime) > debounceDelay) {

    if (reading != buttonState) {
      buttonState = reading;

      if (buttonState == HIGH) {
        relayState = !relayState;
      }
    }
  }

   // set the relay:
  digitalWrite(relayPin, relayState);

  lastButtonState = reading;
}
```

BÀI 7: LẬP TRÌNH CẢM BIẾN

Giới thiệu:

Cảm biến (tên tiếng Anh: Sensor) là một thiết bị có khả năng phát hiện và phản hồi một số loại đầu vào (ánh sáng, nhiệt độ, âm thanh, độ ẩm,…) từ môi trường. Đầu ra sẽ là tín hiệu đã được chuyển đổi và hiển thị trên màn hình điều khiển.

Cảm biến đóng vai trò quan trọng trong Internet vạn vật (IoT), giúp tạo ra một hệ sinh thái để thu thập và xử lý các tín hiệu khác nhau từ môi trường. Từ đó, các tính hiệu này được theo dõi, quản lý và kiểm soát một cách dễ dàng và hiệu quả hơn.

Các cảm biến thường được phân loại dựa trên nguyên lý hoạt động, dữ liệu đầu vào hoặc phạm vi ứng dụng. Do đó, các loại cảm biến có thể được phân thành ba loại dựa trên các nguyên tắc hoạt động như sau: cảm biến vật lý, cảm biến hóa học và cảm biến sinh học.

Cảm biến vật lý: Được chế tạo từ các đặc tính vật lý của thành phần biến đổi cụ thể. Ngoài ra, cảm biến vật lý còn phụ thuộc tính chất vật lý của vật liệu chức năng.

Cảm biến hóa học: Đây là một phản ứng điện hóa, giúp chuyển đổi thành phần, nồng độ của các hợp chất vô cơ hoặc hữu cơ thành tín hiệu điện.

Cảm biến sinh học: Bằng cách sử dụng các hóa chất hoạt tính sinh học, cảm biến sinh học có khả năng phát hiện và đo được các hợp chất sinh hóa.

Mục tiêu của bài:

- Trình bày được cấu trúc của các loại cảm biến
- Thiết kế và lập trình được các mạch ứng dụng dùng cảm biến
- Kiểm tra và sửa chữa lỗi hư hỏng
- Kết nối được chương trình với mô hình
- Phát huy tính chủ động trong học tập và trong công việc.

Nội dung chính:

1. Lập trình cảm biến nhiệt độ.

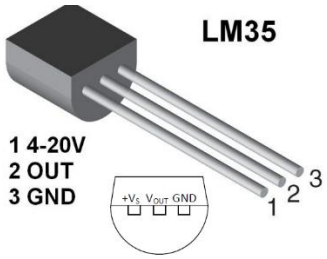
1.1. Lập trình cảm biến nhiệt độ LM35.

Cảm biến nhiệt độ LM35 là một loại cảm biến tương tự rất hay được ứng dụng trong các ứng dụng đo nhiệt độ thời gian thực. Vì nó hoạt động khá chính xác với sai số nhỏ, đồng thời với kích thước nhỏ và giá thành rẻ là một trong những ưu điểm của nó. Vì đây là cảm biến tương tự (analog sensor) nên ta có thể dễ dàng đọc được giá trị của nó bằng hàm analogRead()

Nguyên vật liệu thực hành

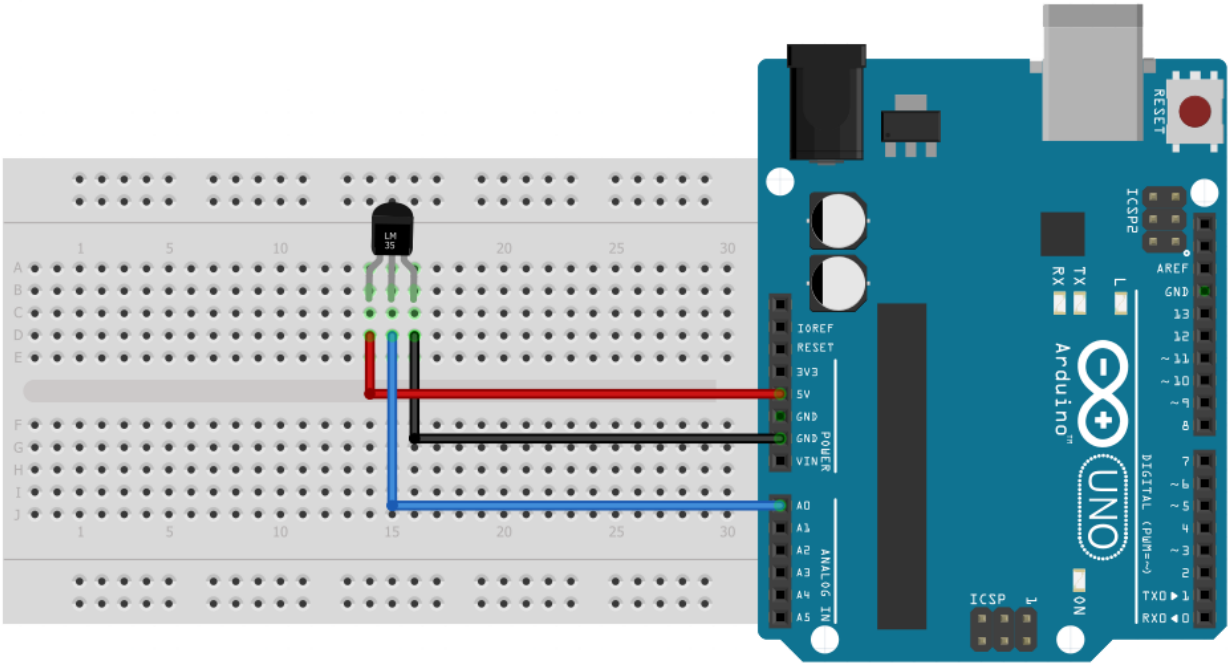
STT	Nguyên vật liệu	Số lượng	Thông số
1	Arduino Uno R3	1	
2	Dây cáp nạp	1	
3	Cảm biến nhiệt LM35	1	
4	Breadboard (Bo test)	1	
5	Dây cắm (Đực – Đực)		

Arduino UNO



Hình 7.1.1: Sơ đồ chân của LM35

Arduino Uno	**Cảm biến nhiệt độ LM35**
VCC	+Vs (4 – 20V)
GND	GND
A0	VOUT



```
int sensorPin = A0;
void setup()
{
   Serial.begin(9600);
}
void loop()
{
  int reading = analogRead(sensorPin);
```

```
  float voltage = reading * 5.0 / 1024.0;
  float temp = voltage * 100.0;
  Serial.println(temp);
  delay(1000);
}
```

Hàm analogRead() có nhiệm vụ đọc giá trị điện áp từ một chân Analog (ADC), hàm này 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. Hàm analogRead() cần 100 micro giây để thực hiện.

Vậy reading = analogRead(sensorPin) có nghĩa là đọc giá trị điện áp từ cảm biến nhiệt độ LM35.

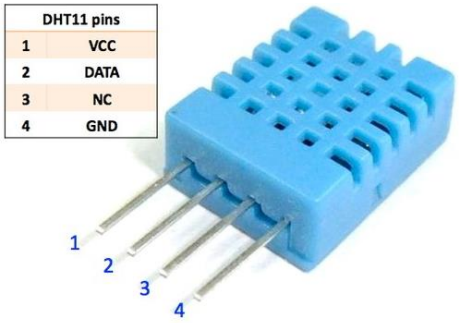
float voltage = reading * 5.0 / 1024.0;

Công thức tính ra giá trị hiệu điện thế từ giá trị cảm biến (đơn vị Volt) Voltage = giá trị điện áp từ cảm biến chia cho mức analog cao nhất (1024) rồi nhân với mức điện áp 5V.

Như ở trên ta thấy nhiệt độ thay đổi tuyến tính 10mV/°C nên đổi từ Vol sang °C thì ta chỉ cần nhân giá trị điện thế với 100 là ra nhiệt độ.

float temp = voltage * 100.0;

1.2. Lập trình cảm biến nhiệt độ và độ ẩm DHT11.



Thông số kĩ thuật:

Điện áp hoạt động: 3-5.5V DC

Ngưỡng độ ẩm: 20 - 90%

Sai số độ ẩm: ± 5%

Ngưỡng nhiệt độ: 0 - 55oC

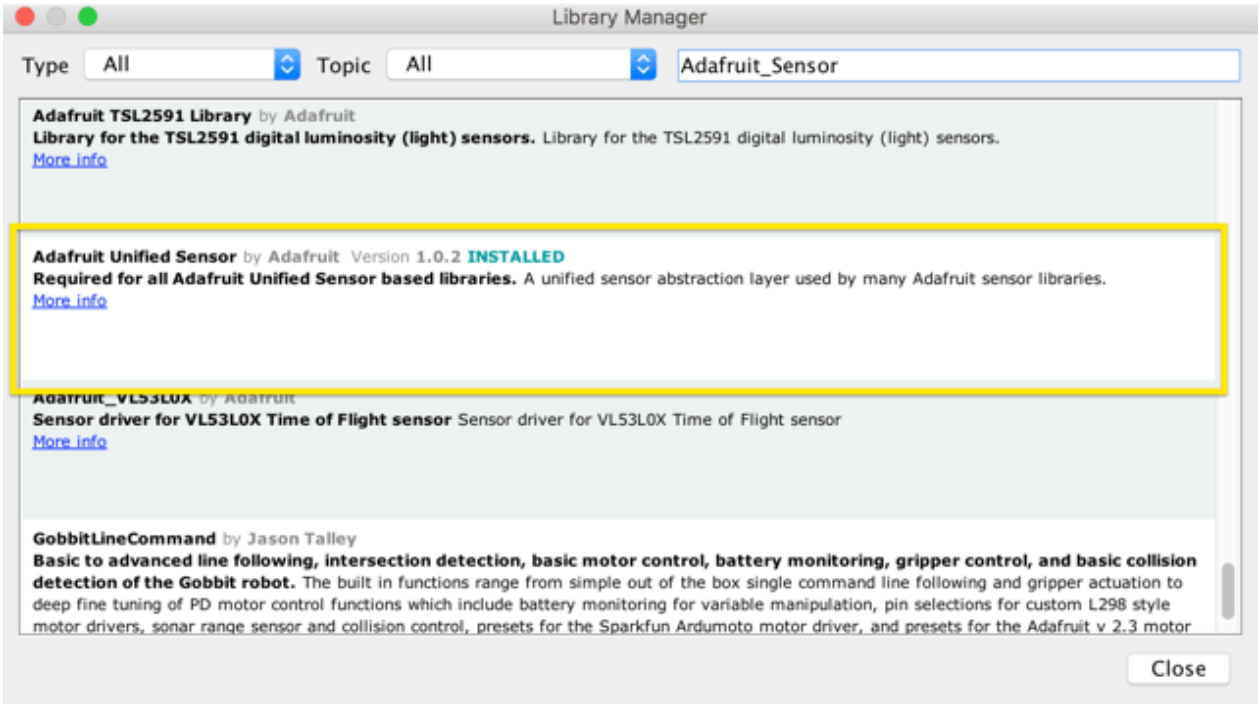
Sai số nhiệt độ: ± 2oC

Kết nối cảm biến DHT11 với mạch Arduino

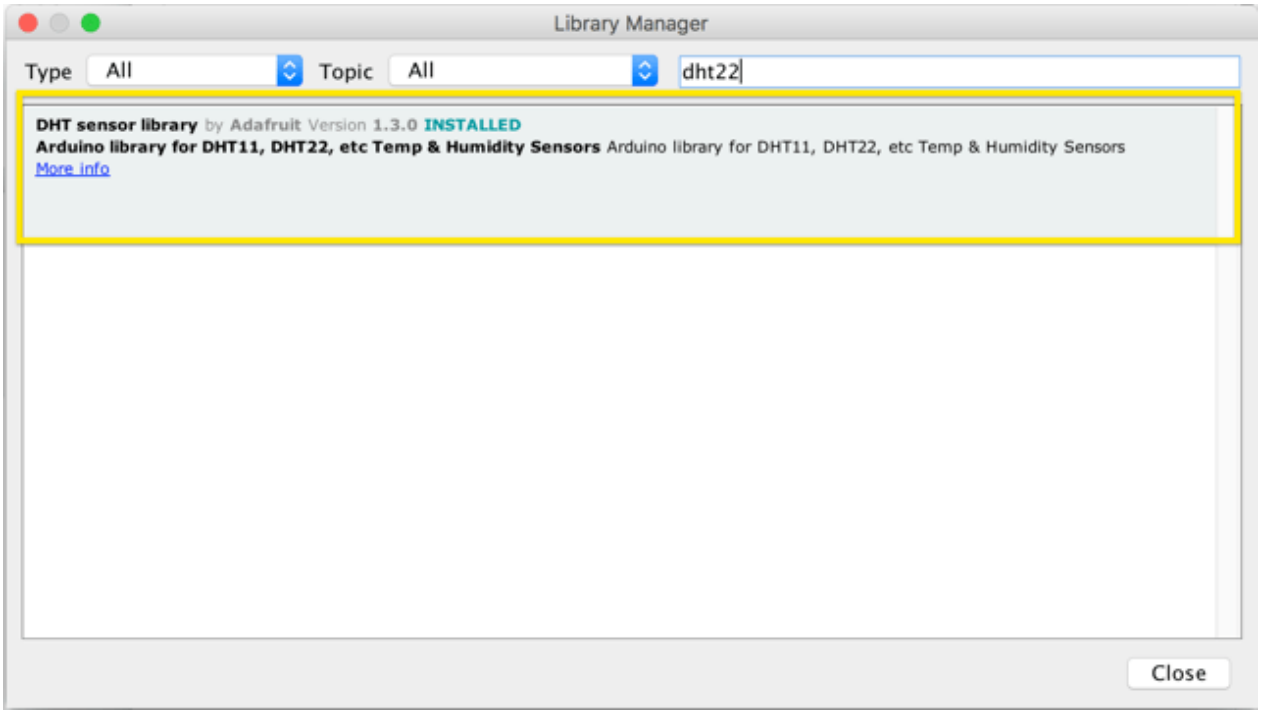
DHT11	Arduino UNO R3
GND	GND
Vcc	5V
Signal	D2

Để lập trình Arduino điều khiển được sensor DHT11 ta cần cài đặt thêm 2 thư viện. Bạn có thể cài đặt thêm thư viện trực tiếp trong Libraries manager hoặc download từ link.

Đầu tiên là Adafruit_Sensor: https://github.com/adafruit/Adafruit_Sensor



Sau đó là thư viện DHT: https://github.com/adafruit/DHT-sensor-library




```
/**
 * Example for reading temperature and humidity
```

```
 * using the DHT22 and ESP8266
 *
 * Copyright (c) 2016 Losant IoT. All rights reserved.
 * https://www.losant.com
 */

#include "DHT.h"

#define DHTPIN 14     // what digital pin the DHT sensor is conected to
#define DHTTYPE DHT11   // there are multiple kinds of DHT sensors

DHT dht(DHTPIN, DHTTYPE, 15);

void setup() {
  Serial.begin(9600);
  Serial.setTimeout(2000);

  // Wait for serial to initialize.
  while(!Serial) { }

  Serial.println("Device Started");
  Serial.println("-------------------------------------");
  Serial.println("Running DHT!");
  Serial.println("-------------------------------------");

}

int timeSinceLastRead = 0;
void loop() {

   // Reading temperature or humidity takes about 250 milliseconds!
   // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
   float h = dht.readHumidity();
   // Read temperature as Celsius (the default)
   float t = dht.readTemperature();
   // Read temperature as Fahrenheit (isFahrenheit = true)
```

```
  float f = dht.readTemperature(true);

  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println("Failed to read from DHT sensor!");
    timeSinceLastRead = 0;
    return;
  }

  // Compute heat index in Fahrenheit (the default)
  float hif = dht.computeHeatIndex(f, h);
  // Compute heat index in Celsius (isFahreheit = false)
  float hic = dht.computeHeatIndex(t, h, false);

  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print(" *C ");
  Serial.print(f);
  Serial.print(" *F\t");
  Serial.print("Heat index: ");
  Serial.print(hic);
  Serial.print(" *C ");
  Serial.print(hif);
 delay(2000);
}
```

Dòng 1: khai báo với hệ thống là trong chương trình ta có sử dụng thư viện DHT của Adafruit.

Dòng 3: do ở đây ta sử dụng cổng Digital 0 trên Node Wifi nên pin sử dụng là 14 để điều khiển module DHT11, vì vậy ta đặt một hằng số với tên gọi DHTPIN có giá trị là 14. Ta cũng khai báo loại DHT là DHT11 ở dòng kế tiếp. Có nhiều loại cảm biến khác nhau trong dòng DHT như DHT22 hay DHT21.

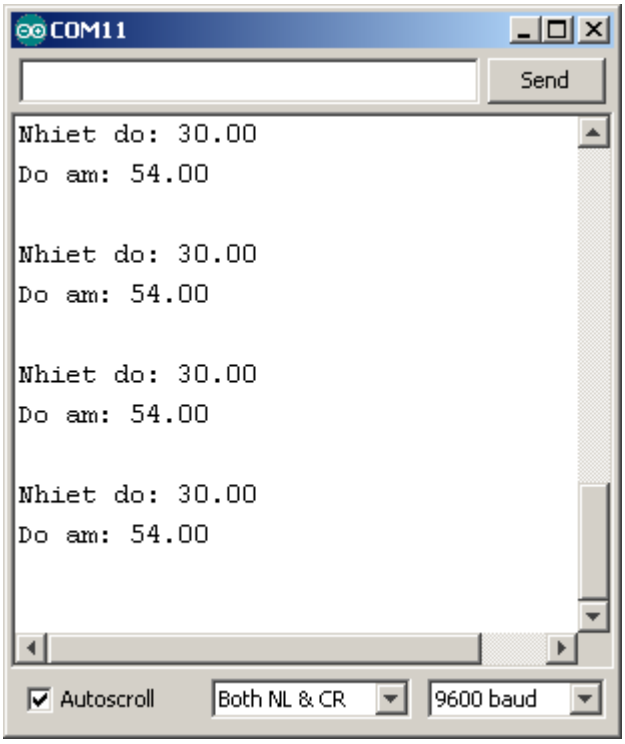
Dòng 5: khai báo đối tượng DHT, ta cần khai báo pin và loại DHT bằng các hằng số đã khai báo ở trên.

Trong hàm setup, ta cần khởi tạo đối tượng Serial để sử dụng in ra giá trị đọc được từ cảm biến trong hàm loop.

Trong hàm loop, ta lần lượt đọc các giá trị độ ẩm, nhiệt độ (độ C) và nhiệt độ (độ F) và in ra cửa sổ Serial.

Tiến hành biên dịch và upload code lên Arduino. Lúc này, bạn sẽ thấy các giá trị đọc được từ cảm biến sẽ được in ra trong cửa sổ Serial

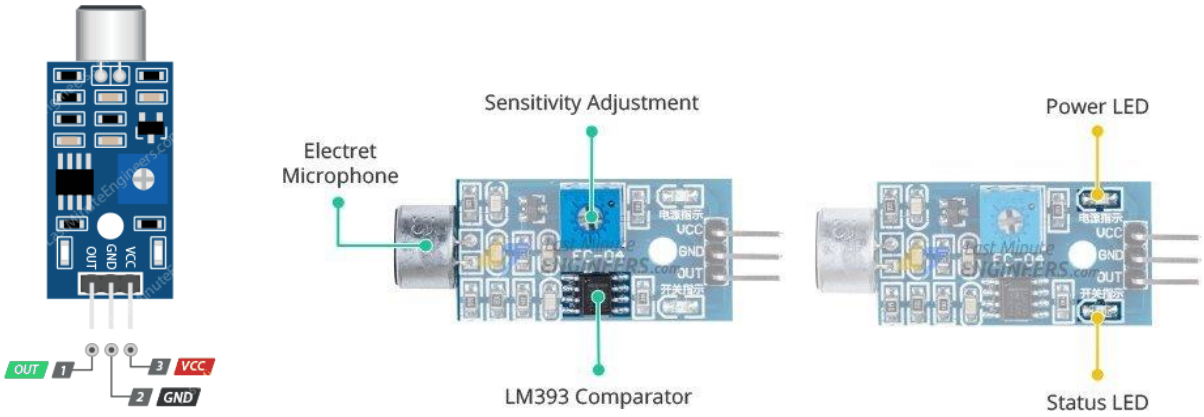
Sau khi upload chương trình lên mạch Arduino, bạn hãy bấm Ctrl + Shift + M để mở cửa sổ Serial Monitor và xem kết quả.



2. Lập trình cảm biến âm thanh.

Module cảm biến âm thanh là một board mạch được trang bị một microphone (50Hz – 100kHz), để thu nhận âm thanh từ môi trường xung quanh. Microphone này chuyển đổi âm thanh thành tín hiệu điện.

Tín hiệu điện này được đưa đến bộ so sánh LM393 trên bo mạch, IC so sánh này sẽ số hóa và truyền tín hiệu ở chân OUT.



Hình 7.2.1: Mạch cảm biến âm thanh

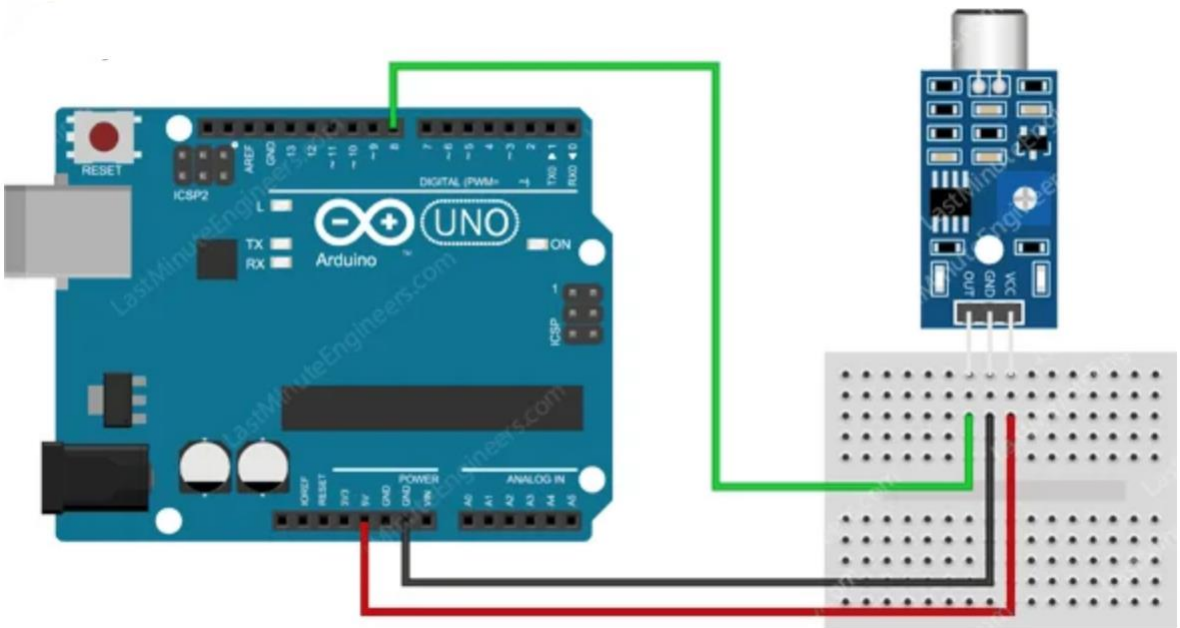
VCC: Cấp nguồn cho cảm biến. Khuyến cáo nguồn cấp cho cảm biến từ 3,3V đến 5V.

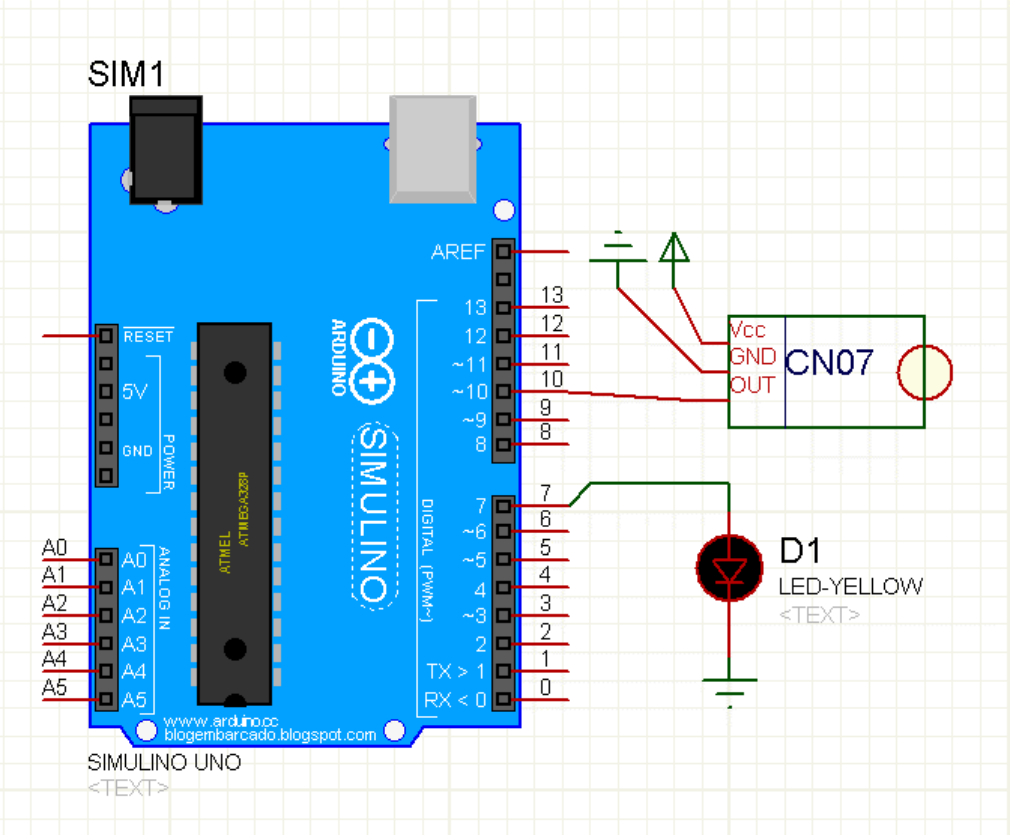
GND: Là chân nối đất.

OUT: Đầu ra mức ở CAO trong điều kiện yên tĩnh và mức THẤP khi phát hiện âm thanh. Có thể kết nối với các chân Digital trên Arduino hoặc nối trực tiếp với Relay.

STT	TÊN LINH KIỆN	SỐ LƯỢNG	
1	Arduino Uno R3	1	
2	Cảm biến âm thanh	1	
3	Dây cắm	1	
4	Breadboard	1	
5	Relay 5V DC	1	

Chân VCC của cảm biến sẽ được nối với chân 5V của board Arduino và chân GND sẽ nối với GND, chân nhận tín hiệu OUT sẽ được nối vào chân Digital (8) của bo Arduino Uno R3.



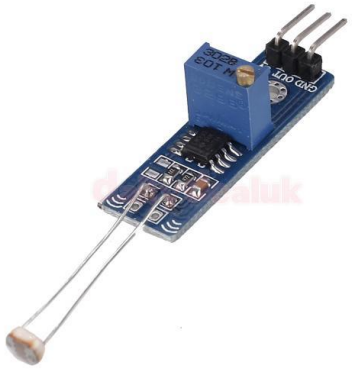



```
#define sensorPin 8

// Variable to store the time when last event happened
unsigned long lastEvent = 0;

void setup() {
   pinMode(sensorPin, INPUT);    // Set sensor pin as an INPUT
   Serial.begin(9600);
}

void loop() {
   // Read Sound sensor
   int sensorData = digitalRead(sensorPin);

   // If pin goes LOW, sound is detected
   if (sensorData == LOW) {

      // If 25ms have passed since last LOW state, it means that
      // the clap is detected and not due to any spurious sounds
      if (millis() - lastEvent > 25) {
```

```
            Serial.println("Clap detected!");
        }

        // Remember when last event happened
        lastEvent = millis();
    }
}
```

3. Bài tập.

3.1. Lập trình cảm biến ánh sáng dùng quang trở.



Trên mạch có 1 biến trở 10K ohm dùng để điều chỉnh độ nhạy sáng:

Vặn về bên trái (nhìn theo hướng từ dưới lên quang trở): sẽ tăng độ nhạy của cảm biến với ánh sáng: chỉ cần lượng ánh sáng nhỏ thì mạch sẽ tự ngắt.

Vặn về bên phải: sẽ giảm độ nhạy của cảm biến với ánh sáng, cần lượng ánh sáng với cường độ mạnh hơn để ngắt mạch.

Cảm biến này có thể sử dụng kết hợp với Arduino để lập trình bật tắt thay vì mạch Rơ-le.

Cảm biến này là một dạng cảm biến Digital - tín hiệu xuất ra là giá trị Digital HIGH (5V) và LOW. Tại chân OUT, mạch trả về mức HIGH (5V) khi trời tối (cường độ ánh sáng chiếu vào thấp) và LOW nếu ngược lại.

Mạch cảm biến ánh sáng dùng quang trở có ưu điểm:

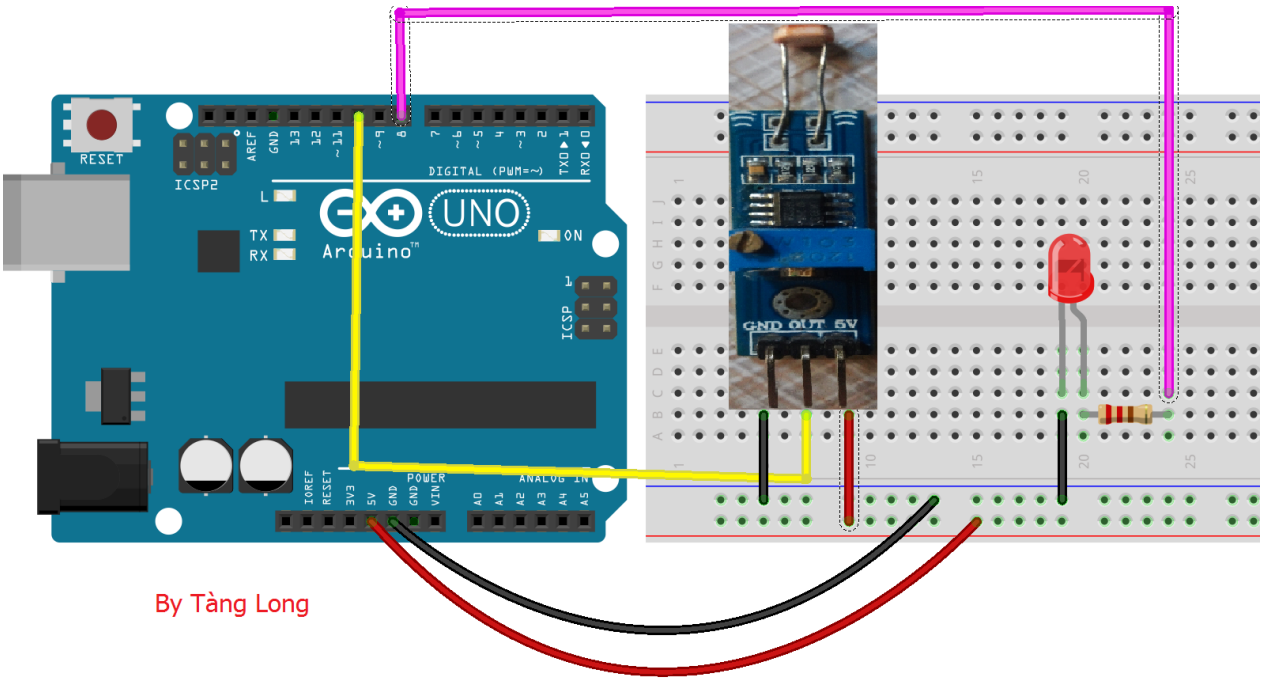
Nhỏ gọn.

Độ chính xác cao.

Các thành phần phụ như điện trở, tụ điện... cần thiết cho mạch đã được gắn đầy đủ. Chỉ cần cấp nguồn, nối dây điều khiển vào rơ le là có thể tắt/mở bóng đèn hay các thiết bị điện khác theo cường độ ánh sáng chiếu vào cảm biến.

Giá thành thấp, khoảng 50.000 đồng.

Sử dụng điện áp chuẩn 5V tương thích với nền tảng Arduino.




```
int cambien = 10;// khai báo chân digital 10 cho cảm biến

int Led = 8;//kháo báo chân digital 8 cho đèn LED

void setup (){

pinMode(Led,OUTPUT);//pinMode xuất tín hiệu đầu ra cho led

pinMode(cambien,INPUT);//pinMode nhận tín hiệu đầu vào cho cảm biê

}

void loop (){

int value = digitalRead(cambien);//lưu giá trị cảm biến vào biến value

digitalWrite(Led,value);//xuất giá trị ra đèn LED

}
```

3.2. Lập trình biến khoảng cách HC-SR04.

Phần cứng

Arduino UNO

Breadboard

Dây cắm breadboard

1 cảm biến siêu âm HC-SR04

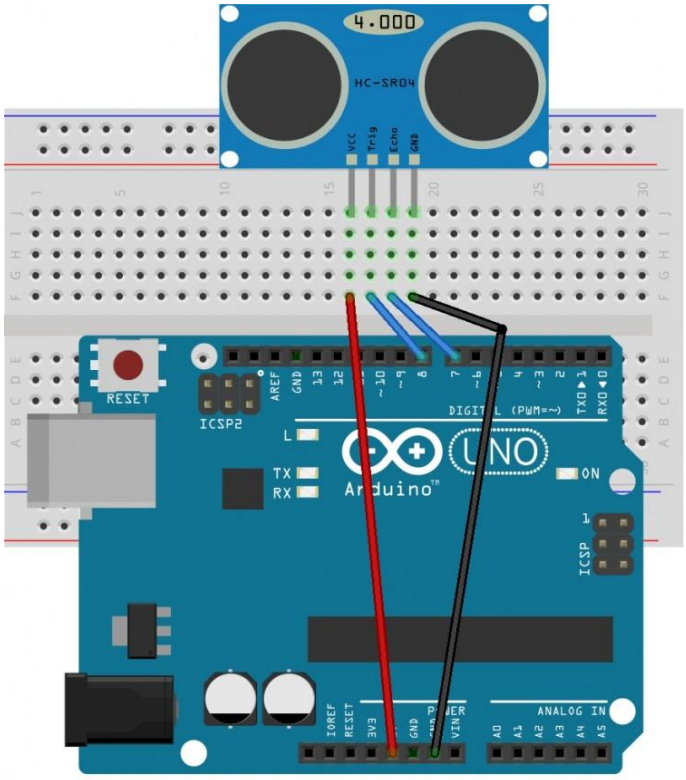
Cảm biến khoảng cách siêu âm HC-SR04 được sử dụng rất phổ biến để xác định khoảng cách vì RẺ và CHÍNH XÁC. Cảm biến sử dụng sóng siêu âm và có thể đo khoảng cách trong khoảng từ 2 -> 300 cm, với độ chính xác gần như chỉ phụ thuộc vào cách lập trình.

Cảm biến HC-SR04 có 4 chân là: Vcc, Trig, Echo, GND.

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

Để đo khoảng cách, ta sẽ phát 1 xung rất ngắn (5 microSeconds - ú) từ chân Trig. Sau đó, cảm biến sẽ tạo ra 1 xung HIGH ở chân Echo cho đến khi nhận lại được sóng phản xạ ở pin này. Chiều rộng của xung sẽ bằng với thời gian sóng siêu âm được phát từ cảm biến và quay trở lại.

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



```
1. const int trig = 8; // chân trig của HC-SR04
2. const int echo = 7; // chân echo của HC-SR04
3.
4. void setup()
5. {
6. Serial.begin(9600); // giao tiếp Serial với baudrate 9600
7. pinMode(trig,OUTPUT); // chân trig sẽ phát tín hiệu
8. pinMode(echo,INPUT); // chân echo sẽ nhận tín hiệu
9. }
10.
11. void loop()
12. {
13. unsigned long duration; // biến đo thời gian
14. int distance; // biến lưu khoảng cách
15.
16. /* Phát xung từ chân trig */
17. digitalWrite(trig,0); // tắt chân trig
18. delayMicroseconds(2);
19. digitalWrite(trig,1); // phát xung từ chân trig
20. delayMicroseconds(5); // xung có độ dài 5 microSeconds
21. digitalWrite(trig,0); // tắt chân trig
22.
23. /* Tính toán thời gian */
24. // Đo độ rộng xung HIGH ở chân echo.
25. duration = pulseIn(echo,HIGH);
26. // Tính khoảng cách đến vật.
27. distance = int(duration/2/29.412);
28.
29. /* In kết quả ra Serial Monitor */
30. Serial.print(distance);
31. Serial.println("cm");
32. delay(200);
33. }
```

3.3. Lập trình cảm biến mưa.

Mạch cảm biến mưa gồm 2 phần:

Mạch cảm biến mưa được gắn ngoài trời

Mạch điều chỉnh độ nhạy cần được che chắn

Mạch cảm biến mưa hoạt động bằng cách so sánh hiệu điện thế của mạch cảm biến nằm ngoài trời với giá trị định trước (giá trị này thay đổi được thông qua 1 biến trở màu xanh) từ đó phát ra tín hiệu đóng / ngắt rơ le qua chân D0. Vì vậy, chúng ta dùng một chân digital để đọc tín hiệu từ cảm biến mưa.

Khi cảm biến khô ráo (trời không mưa), chân D0 của module cảm biến sẽ được giữ ở mức cao (5V). Khi có nước trên bề mặt cảm biến (trời mưa), đèn LED màu đỏ sẽ sáng lên, chân D0 được kéo xuống thấp (0V).

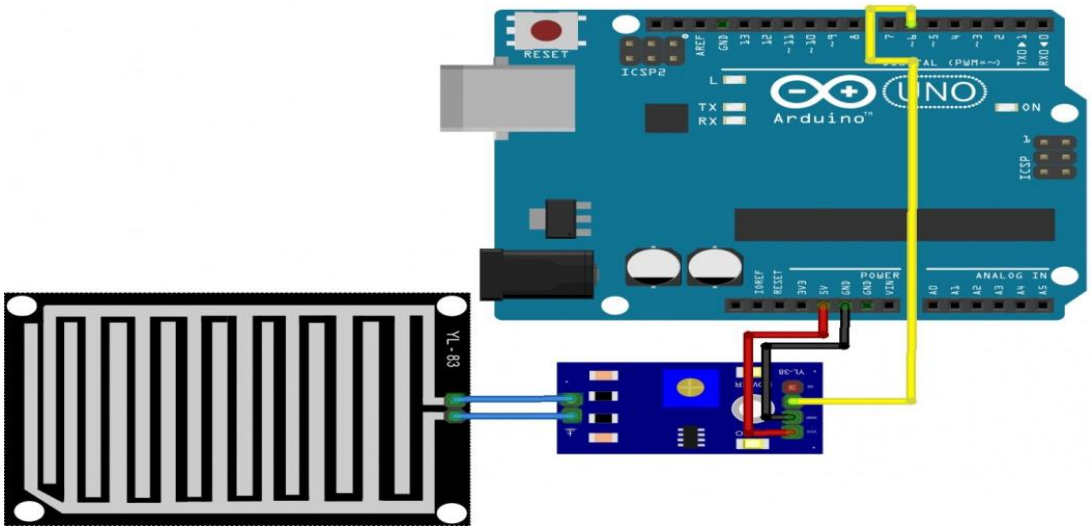
Phần cứng

Arduino UNO

Breadboard

Dây cắm breadboard

1 module cảm biến mưa



Sơ đồ chân nối

Cảm biến mưa	Arduino Uno
GND	GND
VCC	5V
D0	Digital 6

```
int rainSensor = 6; // Chân tín hiệu cảm biến mưa ở chân digital 6 (arduino)
void setup() {
  pinMode(rainSensor,INPUT);// Đặt chân cảm biến mưa là INPUT, vì tín hiệu sẽ được truyền đến cho Arduino
  Serial.begin(9600);// Khởi động Serial ở baudrate 9600
  Serial.println("Da khoi dong xong");
}
```

```
void loop() {
  int value = digitalRead(rainSensor);//Đọc tín hiệu cảm biến mưa
  if (value == HIGH) { // Cảm biến đang không mưa
    Serial.println("Dang khong mua");
  } else {
    Serial.println("Dang mua");
  }
  delay(1000); // Đợi 1 tí cho lần kiểm tra tiếp theo. Bạn hãy tham khảo bài "Viết chương trình không dùng làm delay" trên Arduino.VN để kết hợp đoạn code này và cả chương trình của bạn
}
```

3.4. Lập trình cảm biến chuyển động.

PIR là chữ viết tắt của Passive InfraRed sensor (PIR sensor), tức là bộ cảm biến thụ động dùng nguồn kích thích là tia hồng ngoại. Tia hồng ngoại (IR) chính là các tia nhiệt phát ra từ các vật thể nóng. Trong các cơ thể sống, trong chúng ta luôn có thân nhiệt (thông thường là ở 37 độ C), và từ cơ thể chúng ta sẽ luôn phát ra các tia nhiệt, hay còn gọi là các tia hồng ngoại, người ta sẽ dùng một tế bào điện để chuyển đổi tia nhiệt ra dạng tín hiệu điện và nhờ đó mà có thể làm ra cảm biến phát hiện các vật thể nóng đang chuyển động. Cảm biến này gọi là thụ động vì nó không dùng nguồn nhiệt tự phát (làm nguồn tích cực, hay chủ động) mà chỉ phụ thuộc vào các nguồn tha nhiệt, đó là thân nhiệt của các thực thể khác, như con người con vật…

Infrared Radial Sensor

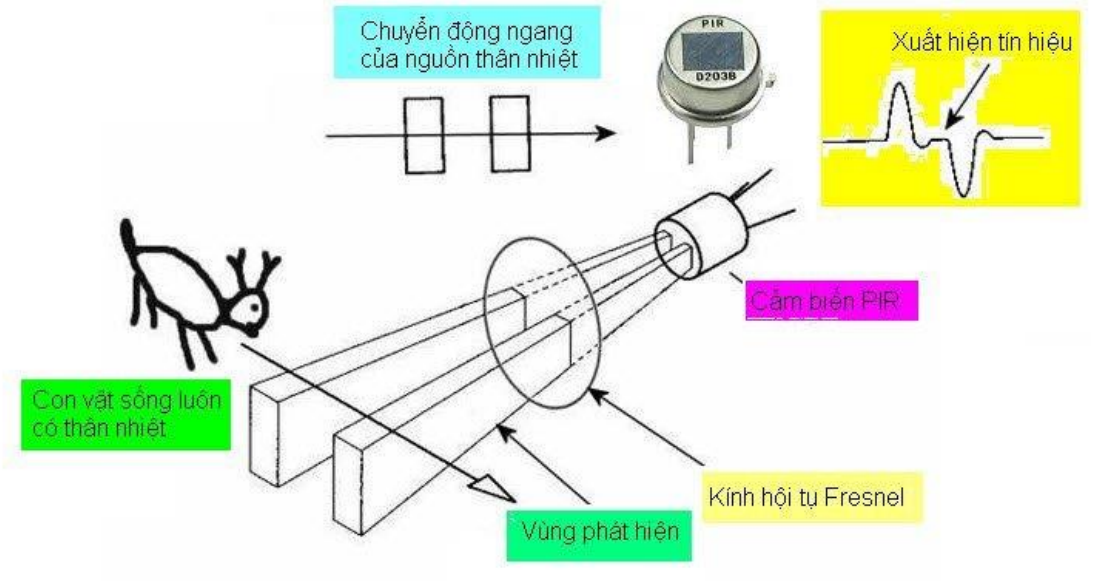
Features
- Dual Compensating Elements
- Excellent Operating Stability
- Supply Voltage: 3-15V
- Larger Sense Window for Area Detection
- Body Dimensions: 9.1mm Diameter, 4.5mm High excluding pins, Pins - 13.5mm High

Fresnel Lens

Features
- Designed for Use with above Sensors
- Optimized for Dual Element Pyroelectric Devices
- White Light Immunity to Reduce False Triggers
- UV Resistant for Outdoor Applications
- Designed for Uniform Sensitivity to Reduce Electronic Gain

Đây là đầu dò PIR, loại bên trong gắn 2 cảm biến tia nhiệt, nó có 3 chân ra, một chân nối masse, một chân nối với nguồn volt DC, mức áp làm việc có thể từ 3 đến 15V. Góc dò

lớn. Để tăng độ nhậy cho đầu dò, Bạn dùng kính Fresnel, nó được thiết kế cho loại đầu có 2 cảm biến, góc dò lớn, có tác dụng ngăn tia tử ngoại.



Nguyên lý phát hiện chuyển động ngang của các nguồn thân nhiệt

Các nguồn nhiệt (với người và con vật là nguồn thân nhiệt) đều phát ra tia hồng ngoại, qua kính Fresnel, qua kích lọc lấy tia hồng ngoại, nó được cho tiêu tụ trên 2 cảm biến hồng ngoại gắn trong đầu dò, và tạo ra điện áp được khuếch đại với transistor FET. Khi có một vật nóng đi ngang qua, từ 2 cảm biến này sẽ cho xuất hiện 2 tín hiệu và tín hiệu này sẽ được khuếch đại để có biên độ đủ cao và đưa vào mạch so áp để tác động vào một thiết bị điều khiển hay báo động.

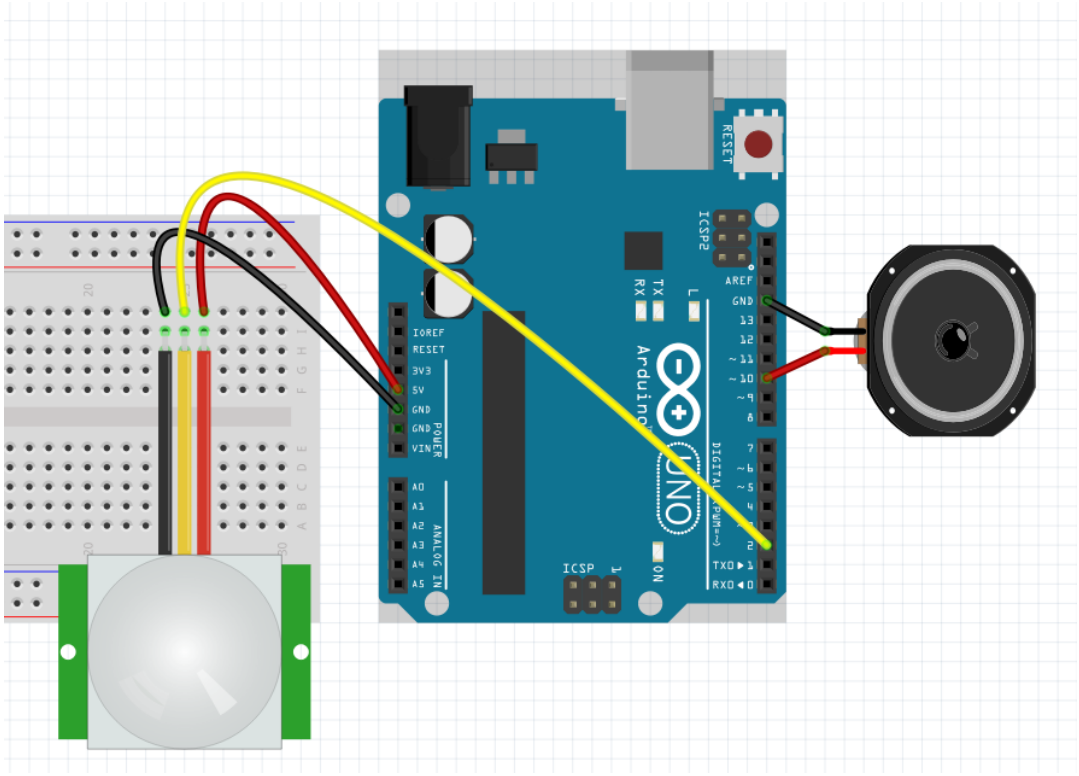
Chuẩn bị

Board Arduino UNO R3

Module PIR - Cảm biến chuyển động

Một đèn LED

Loa




```
1.  int ledPin = 13; // chọn chân 13 báo hiệu LED
2.  int inputPin = 2; // chọn ngõ tín hiệu vào cho PIR
3.  int pirState = LOW; // Bắt đầu với không có báo động
4.  int val = 0;
5.  int pinSpeaker = 10; //chọn chân cho chuông khi có đột nhập
6.
7.  void setup()
8.  {
9.  pinMode(ledPin, OUTPUT);
10. pinMode(inputPin, INPUT);
11. pinMode(pinSpeaker, OUTPUT);
12. Serial.begin(9600);
13. }
14.
15. void loop()
16. {
17. val = digitalRead(inputPin); // đọc giá trị đầu vào.
18. if (val == HIGH) // nếu giá trị ở mức cao.(1)
19. {
20. digitalWrite(ledPin, HIGH); // LED On
21. playTone(300, 160); // thời gian chuông kêu
```

```
22. delay(150);
23.
24. if (pirState == LOW)
25. {
26. Serial.println("Motion detected!");
27. pirState = HIGH;
28. }
29. }
30. else
31. {
32. digitalWrite(ledPin, LOW);
33. playTone(0, 0);
34. delay(300);
35. if (pirState == HIGH)
36. {
37. Serial.println("Motion ended!");
38. pirState = LOW;
39. }
40. }
41. }
42.
43. void playTone(long duration, int freq)
44. {
45. duration *= 1000;
46. int period = (1.0 / freq) * 1000000;
47. long elapsed_time = 0;
48. while (elapsed_time < duration)
49. {
50. digitalWrite(pinSpeaker,HIGH);
51. delayMicroseconds(period / 2);
52. digitalWrite(pinSpeaker, LOW);
53. delayMicroseconds(period / 2);
54. elapsed_time += (period);
55. }
56. }
```

BÀI 8: LẬP TRÌNH ĐIỀU KHIỂN TỪ XA

Giới thiệu:

Cảm biến vật cản hồng ngoại (Infrared Obstacle Sensor) là một loại cảm biến sử dụng công nghệ hồng ngoại để phát hiện và đo khoảng cách vật cản. Cảm biến này sử dụng tia hồng ngoại để phát ra một tín hiệu và đo thời gian mà tín hiệu này trở lại sau khi va chạm với vật cản, từ đó tính toán khoảng cách giữa cảm biến và vật cản.

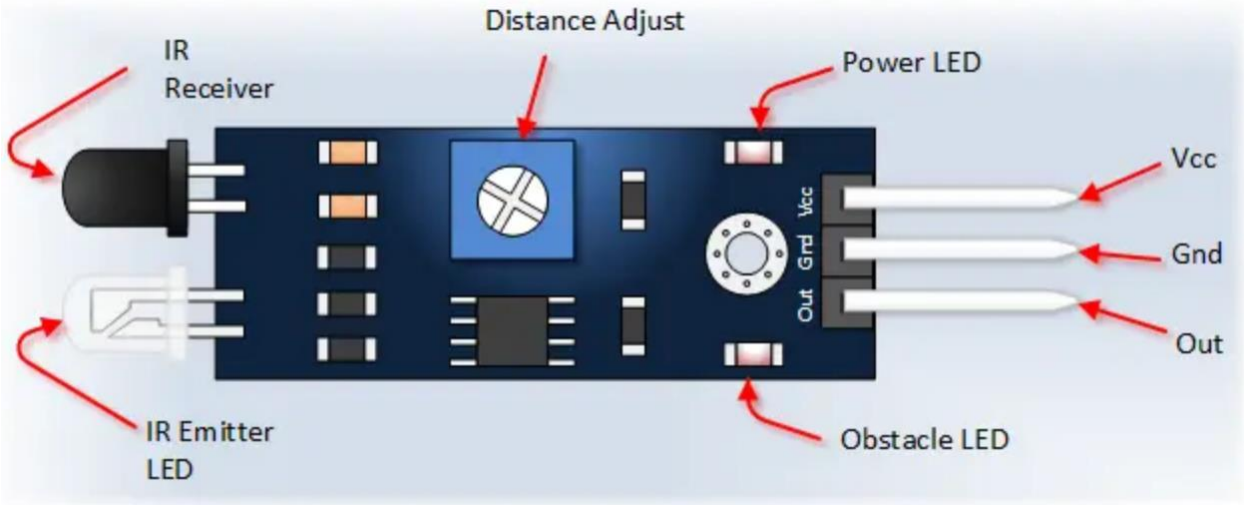
Cảm biến phát hiện vật cản thường được sử dụng trong các ứng dụng như robot tự động hóa, xe tự lái, hệ thống giám sát khoảng cách, và hệ thống an ninh tự động. Nó có thể hoạt động ở mức năng suất cao, đồng thời cũng đảm bảo tính an toàn trong các ứng dụng tự động.

Mục tiêu của bài:

- Trình bày được cấu trúc của các thiết bị từ xa
- Thiết kế và lập trình được các mạch ứng dụng điều khiển từ xa
- Kiểm tra và sửa chữa lỗi hư hỏng
- Kết nối được chương trình với mô hình
- Phát huy tính chủ động trong học tập và trong công việc.

Nội dung chính:

1. Điều khiển qua hồng ngoại



Hình 8.1.1: Cấu tạo modul hồng ngoại.

Chân VCC: là chân cấp nguồn cho module, thường được nối với nguồn 5V DC.

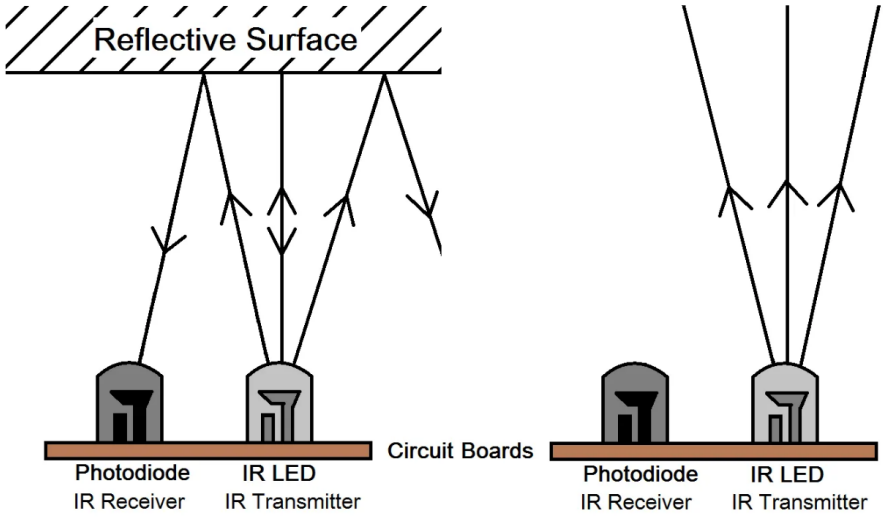
Chân GND: là chân đất, được nối với chân đất của nguồn.

Chân OUT: là chân dữ liệu, sẽ đưa ra tín hiệu khi có sự phát hiện của đối tượng.

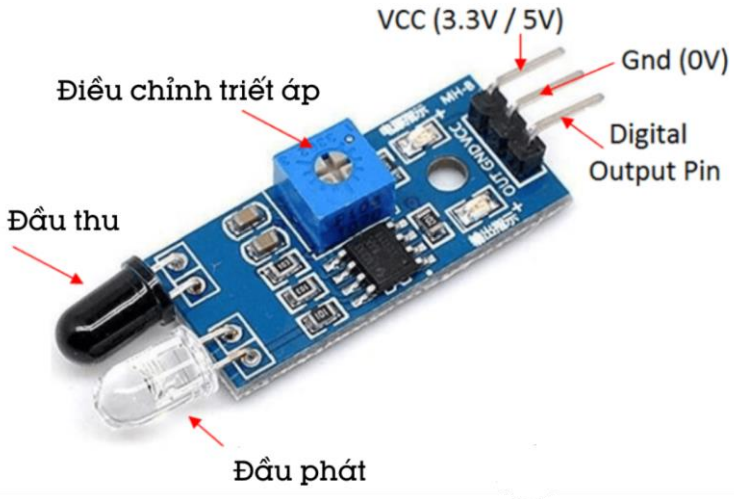
Nguyên lý hoạt động của cảm biến vật cản hồng ngoại

Cảm biến vật cản hồng ngoại hoạt động dựa trên nguyên lý thu phát sóng hồng ngoại. Khi cảm biến nhận được tín hiệu từ một nguồn phát sóng hồng ngoại, nó sẽ phản hồi lại tín

hiệu đó và đo lường thời gian phản hồi của sóng để xác định khoảng cách từ cảm biến đến vật thể.



Cảm biến sử dụng một bộ phát sóng hồng ngoại để phát ra tín hiệu sóng hồng ngoại. Khi sóng hồng ngoại va chạm với vật thể, nó sẽ bị phản xạ trở lại và được nhận bởi một bộ thu sóng hồng ngoại trên cảm biến. Cảm biến sau đó sử dụng độ trễ thời gian giữa tín hiệu phát và tín hiệu nhận để tính toán khoảng cách từ cảm biến đến vật thể.



2. Lập trình điều khiển qua hồng ngoại

2.1. Chuẩn bị dụng cụ, thiết bị vật liệu.

Tên linh kiện	**Số lượng**
Arduino Uno R3	1
Cáp nạp	1

Bread Board (Bo Test)	1
Relay 5V 1 kênh	1
Dây cắm (Đực – Cái)	1
Dây cắm (Đực – Đực)	1
Cảm biến vật cản hồng ngoại	2
LCD16X2	1
Board I2C LCD16X2	1
LED 5V/5MM	1
Nguồn DC 12V/2A	1

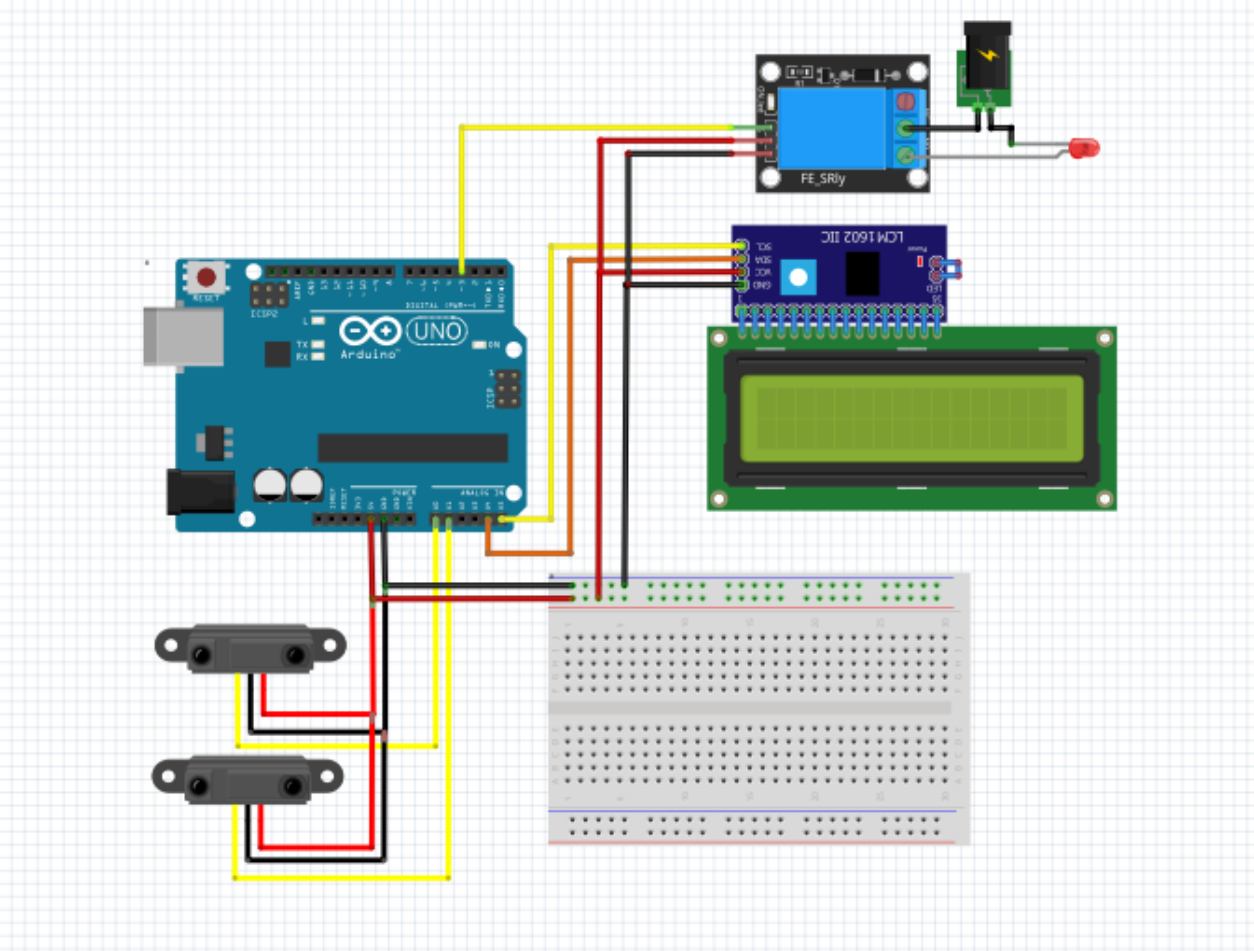
2.2. Lập trình điều khiển qua hồng ngoại.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x3F, 16, 2);
#define in 14
#define out 17
#define relay 3
int count=0;
void IN()
{
   count++;
   lcd.clear();
   lcd.print("SO NGUOI:");
   lcd.setCursor(0,1);
   lcd.print(count);
   delay(1000);
}
void OUT()
{
  count--;
```

```
    lcd.clear();
    lcd.print("SO NGUOI:");
    lcd.setCursor(0,1);
    lcd.print(count);
    delay(1000);
}
void setup()
{
  lcd.begin();
  lcd.backlight();
  Serial.begin(9600);
  lcd.setCursor(0,2);
  lcd.print(" HELLO ");
  delay(2000);
  pinMode(in, INPUT);
  pinMode(out, INPUT);
  pinMode(relay, OUTPUT);
  lcd.clear();
  lcd.print("Person In Room:");
  lcd.setCursor(0,1);
  lcd.print(count);

}
void loop()
{
   if(!digitalRead(out))
    IN();
   if(!digitalRead(in))
    OUT();
  Serial.print(count);
  Serial.println("");

  if(count<=0)
  {
    lcd.clear();
    digitalWrite(relay, LOW);
    lcd.clear();
    lcd.print("KO CO NGUOI:");
    lcd.setCursor(0,1);
```

```
    lcd.print("Den Tat");
    delay(200);
  }
  else
    digitalWrite(relay, HIGH);
}
```

2.3. Kết nối dây dẫn và vận hành mạch.



Ở hàm loop(), chúng ta kiểm tra trạng thái của chân OUT và chân IN. Nếu chân OUT được kích hoạt, chúng ta tăng biến count lên 1 và hiển thị thông tin số lượng người trên LCD thông qua hàm IN(). Nếu chân IN được kích hoạt, chúng ta giảm biến count đi 1 và hiển thị thông tin số lượng người trên LCD thông qua hàm OUT(). Sau đó, chúng ta in giá trị hiện tại của biến count ra Serial Monitor.

Nếu biến count có giá trị nhỏ hơn hoặc bằng 0, chúng ta tắt đèn trong phòng bằng cách đặt chân relay xuống mức thấp và hiển thị thông báo “KO CO NGUOI: Den Tat” trên LCD. Ngược lại, nếu biến count có giá trị lớn hơn 0, chúng ta bật đèn trong phòng bằng cách đặt chân relay lên mức cao.

2.4. Hư hỏng thường gặp, nguyên nhân và cách khắc phục.

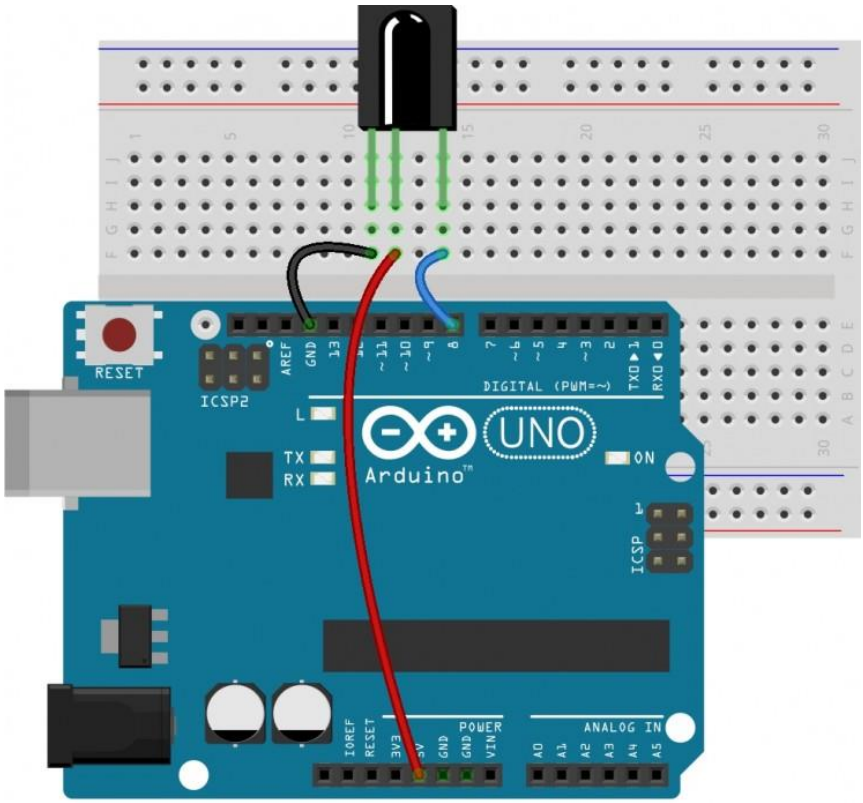
STT	Hư hỏng thường gặp	Nguyên nhân	Cách khắc phục
1	Cảm biến không nhận được tín hiệu	Cảm biến không được cấp điện đúng cách. Đấu dây không đúng hoặc lỏng. Cảm biến bị hỏng.	Kiểm tra và đảm bảo rằng cảm biến được cấp điện đúng cách. Kiểm tra và chắc chắn rằng các dây kết nối đều đúng và chắc chắn. Thay cảm biến nếu bị hỏng.
2	Cảm biến nhận diện sai hoặc không chính xác	Cảm biến bị bụi bẩn hoặc che khuất. Góc hoặc khoảng cách cảm biến không phù hợp. Lỗi trong mã lập trình.	Làm sạch cảm biến và đảm bảo không có vật cản che khuất. Điều chỉnh góc và khoảng cách của cảm biến sao cho phù hợp với yêu cầu sử dụng. Kiểm tra lại mã lập trình và đảm bảo rằng nó được viết chính xác và không có lỗi logic.
3	Cảm biến phản hồi chậm	Tần số quét của cảm biến không phù hợp. Lỗi trong lập trình, ví dụ như độ trễ không cần thiết.	Điều chỉnh tần số quét của cảm biến cho phù hợp với yêu cầu. Kiểm tra mã lập trình và loại bỏ các đoạn mã gây độ trễ không cần thiết.
4	Xung đột phần cứng hoặc phần mềm	Cảm biến sử dụng cùng chân GPIO với các thiết bị khác. Thư viện phần mềm không tương thích.	Kiểm tra sơ đồ nối dây và đảm bảo rằng các chân GPIO không bị xung đột. Sử dụng các thư viện phần mềm tương thích và kiểm tra xung đột với các thư viện khác.

3. Bài tập.

3.1. Infrared remote control (điều khiển bằng hồng ngoại) với Arduino

- ✓ 1 board Arduino UNO x1.
- ✓ IR LED (hay còn gọi là LED hồng ngoại) x1.
- ✓ Điện trở 30 ôm x1.
- ✓ IR receiver x1.
- ✓ Remote TV x1.
- ✓ Nút bấm x4.
- ✓ Dây cắm breadboard

✓ Breadboard



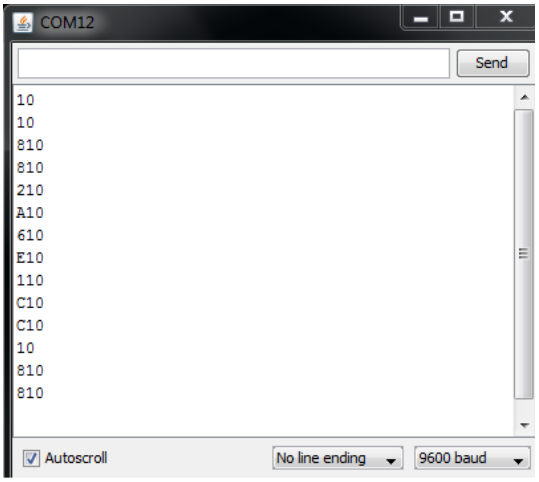
Đoạn mã này dùng để đọc tín hiệu từ remote. Vào link sau để tải thư viện hỗ trợ IR remote: https://github.com/shirriff/Arduino-IRremote/

Sau khi tải thư viện về, mở cửa sổ Arduino, chọn Sketch--->Import Library...---> Add Library....sau đó chọn file .zip vừa tải về để có thể sử dụng thư viện.

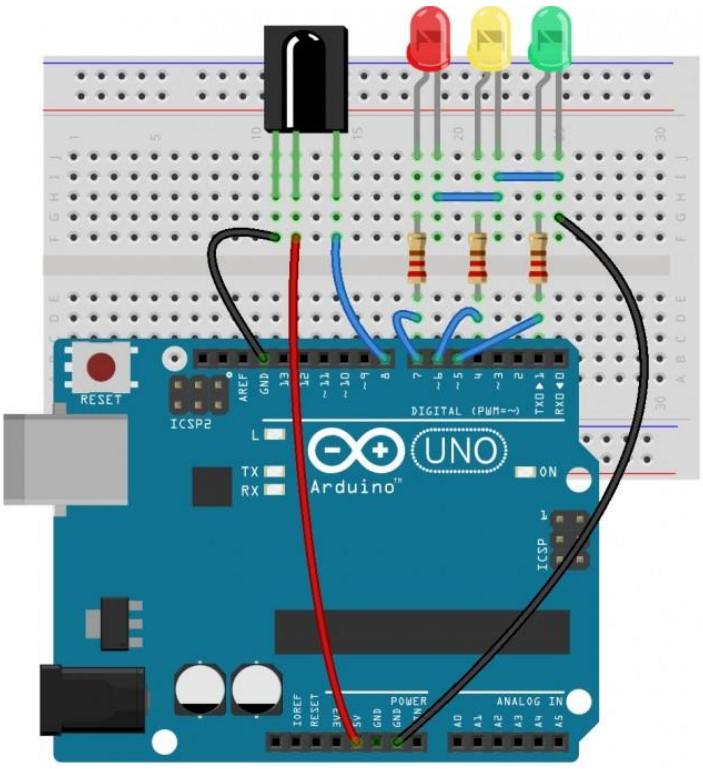
```
#include <IRremote.h> // thư viện hỗ trợ IR remote

const int receiverPin = 8; // chân digital 8 dùng để đọc tín hiệu
IRrecv irrecv(receiverPin); // tạo đối tượng IRrecv mới
decode_results results;// lưu giữ kết quả giải mã tín hiệu
void setup()
{
  Serial.begin(9600);// serial baudrate 9600
  irrecv.enableIRIn(); // start the IR receiver
}
void loop()
{
  if (irrecv.decode(&results)) // nếu nhận được tín hiệu
  {
```

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

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



3.2. Dùng remote để bật tắt LED .



```
#include <IRremote.h> // thư viện hỗ trợ IR remote

const int receiverPin = 8; // chân digital 8 dùng để đọc tín hiệu
IRrecv irrecv(receiverPin); // tạo đối tượng IRrecv mới
decode_results results;// lưu giữ kết quả giải mã tín hiệu

const int RED = 7;// LED đỏ
const int YELLOW = 6;// LED vàng
const int GREEN = 5;// LED xanh
/* trạng thái của các LEDs*/
boolean stateRED = false;
boolean stateYELLOW = false;
boolean stateGREEN = false;
void setup()
{
  Serial.begin(9600);// serial
  irrecv.enableIRIn(); // start the IR receiver
  pinMode(RED, OUTPUT);
  pinMode(YELLOW, OUTPUT);
  pinMode(GREEN, OUTPUT);
}
// translate IR signals
void translateIR()
{
  switch(results.value)
  {
    case 0x10:  stateRED = !stateRED;
                digitalWrite(RED, stateRED);
                break;
    case 0x810: stateYELLOW = !stateYELLOW;
                digitalWrite(YELLOW, stateYELLOW);
                break;
    case 0x410: stateGREEN = !stateGREEN;
                digitalWrite(GREEN, stateGREEN);
                break;
    case 0xA90: stateRED = stateYELLOW = stateGREEN = false;
                digitalWrite(RED, 0);
                digitalWrite(YELLOW, 0);
```

```
            digitalWrite(GREEN, 0);
  }
}
void loop()
{
  if (irrecv.decode(&results)) // nếu nhận được tín hiệu
  {
    translateIR();
    Serial.println(results.value, HEX);
    delay(200);
    irrecv.resume(); // nhận giá trị tiếp theo
  }
}
```

GIẢI THÍCH

Với đoạn mã trên, khi nhấn các nút 1, 2, 3 trên remote, các LED đỏ, vàng, xanh sẽ lần lượt on-off, khi nhấn nút Power thì toàn bộ LEDs sẽ tắt.

Các keywords của thư viện như:

IRrecv irrecv(receiverPin);: tạo đối tượng IRrecv mới có tên là irrecv sử dụng tham số là receiverPin.

decode_results results; : lưu kết quả giải mã được

irrecv.enableIRIn(); : bắt đầu giải mã tín hiệu IR

irrecv.decode(&results);: trả về true nếu có tín hiệu đến

irrecv.resume; : đợi tín hiệu IR tiếp theo.

BÀI 9: LẬP TRÌNH ĐIỀU KHIỂN KHÔNG DÂY

Giới thiệu:

ESP8266 là một module Wi-Fi với khả năng kết nối Internet và được tích hợp sẵn trên một số board nhúng như NodeMCU, Wemos, và ESP-01. ESP8266 có thể hoạt động như một điểm truy cập (access point), một client kết nối đến một điểm truy cập khác, hoặc cả hai đều được. Nó được sử dụng rộng rãi trong các ứng dụng IoT (Internet of Things) như cảm biến thông minh, hệ thống kiểm soát thiết bị, hoặc các ứng dụng điều khiển từ xa. Module này có giá thành rẻ và rất dễ sử dụng, cùng với đó là khả năng tương thích với nhiều loại vi điều khiển khác nhau.

ESP8266 là dòng chip tích hợp Wi-Fi 2.4Ghz có thể lập trình được, rẻ tiền được sản xuất bởi một công ty bán dẫn Trung Quốc: Espressif Systems. Được phát hành đầu tiên vào tháng 8 năm 2014, đóng gói đưa ra thị trường dạng Module ESP-01, được sản xuất bởi bên thứ 3: AI-Thinker. Có khả năng kết nối Internet qua mạng Wi-Fi một cách nhanh chóng và sử dụng rất ít linh kiện đi kèm. Với giá cả có thể nói là rất rẻ so với tính năng và khả năng ESP8266 có thể làm được.

ESP8266 có một cộng đồng các nhà phát triển trên thế giới rất lớn, cung cấp nhiều Module lập trình mã mở giúp nhiều người có thể tiếp cận và xây dựng ứng dụng rất nhanh.

Hiện nay tất cả các dòng chip ESP8266 trên thị trường đều mang nhãn ESP8266EX, là phiên bản nâng cấp của ESP8266

Mục tiêu của bài:

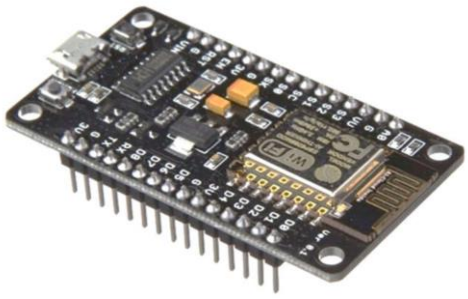
- Trình bày được cấu trúc của các thiết bị không dây
- Thiết kế và lập trình được các mạch ứng dụng điều khiển không dây
- Kiểm tra và sửa chữa lỗi hư hỏng
- Kết nối được chương trình với mô hình
- Phát huy tính chủ động trong học tập và trong công việc.

Nội dung chính:

1. Điều khiển qua wifi.

ESP8266 ESP-01 là một module IoT được thiết kế dựa trên vi điều khiển ESP8266 của công ty Espressif Systems. Đây là phiên bản module nhỏ gọn nhất trong các phiên bản của ESP8266 với kích thước chỉ 24mm x 14mm.

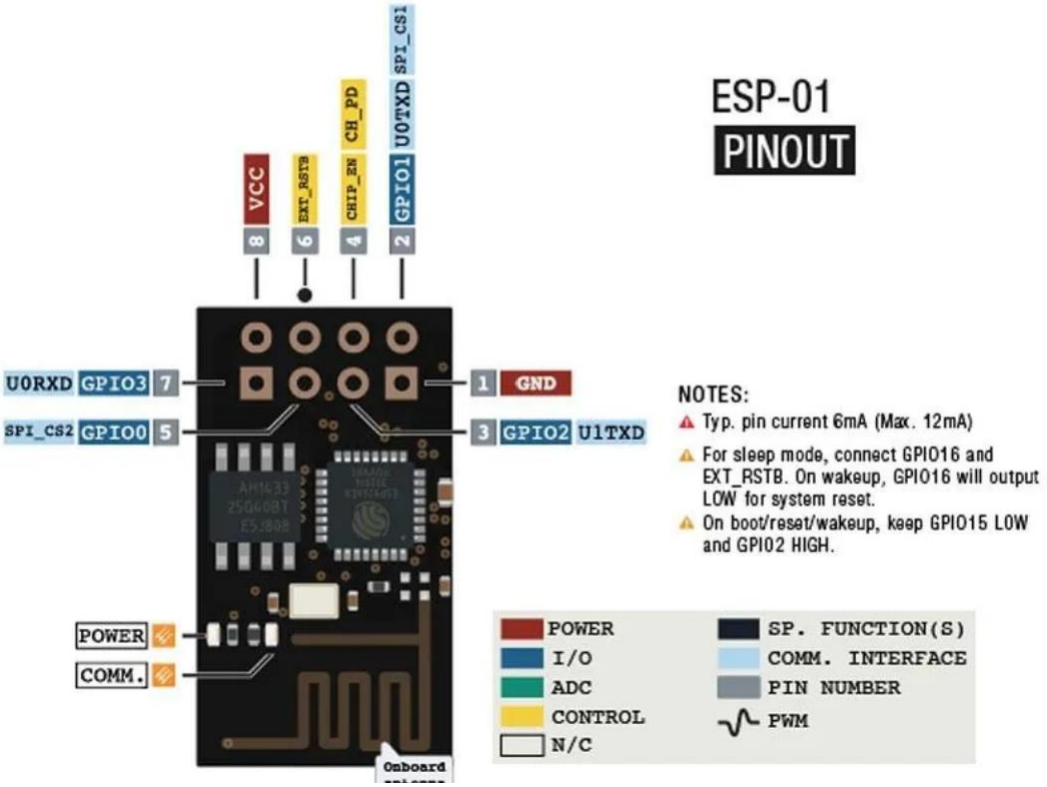
ESP-01 có tích hợp sẵn một bộ phát WiFi, đủ để kết nối với mạng internet và truyền dữ liệu. Module này còn được tích hợp một cổng giao tiếp chuẩn UART, cho phép truyền dữ liệu giữa ESP8266 và các thiết bị khác như Arduino, Raspberry Pi, hoặc máy tính thông qua cổng COM. ESP8266 01 cũng có khả năng lập trình và nạp firmware thông qua cổng UART, giúp cho việc phát triển ứng dụng IoT trở nên đơn giản hơn.



Hình 9.1.1: Module ESP8266

Thông số kỹ thuật của Module Wifi ESP8266 ESP-01:

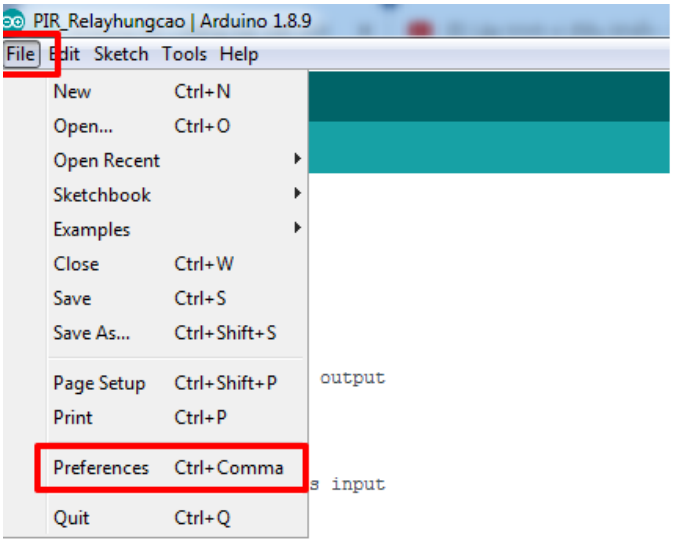
Chip Wi-Fi: ESP8266EX
Nguồn cấp: 3.0V ~ 3.6V DC
Tiêu thụ dòng điện:
Chế độ hoạt động: 80mA ~ 170mA
Chế độ chờ: < 1.0mA
Chuẩn giao tiếp: Wi-Fi 802.11 b/g/n
Tốc độ truyền dữ liệu: 110 ~ 460800bps (tùy chọn)
Điện áp: 3.3V DC
Kích thước: 24mm x 14mm x 3mm
Anten: PCB Anten hoặc IPEX anten ngoài (tùy chọn)



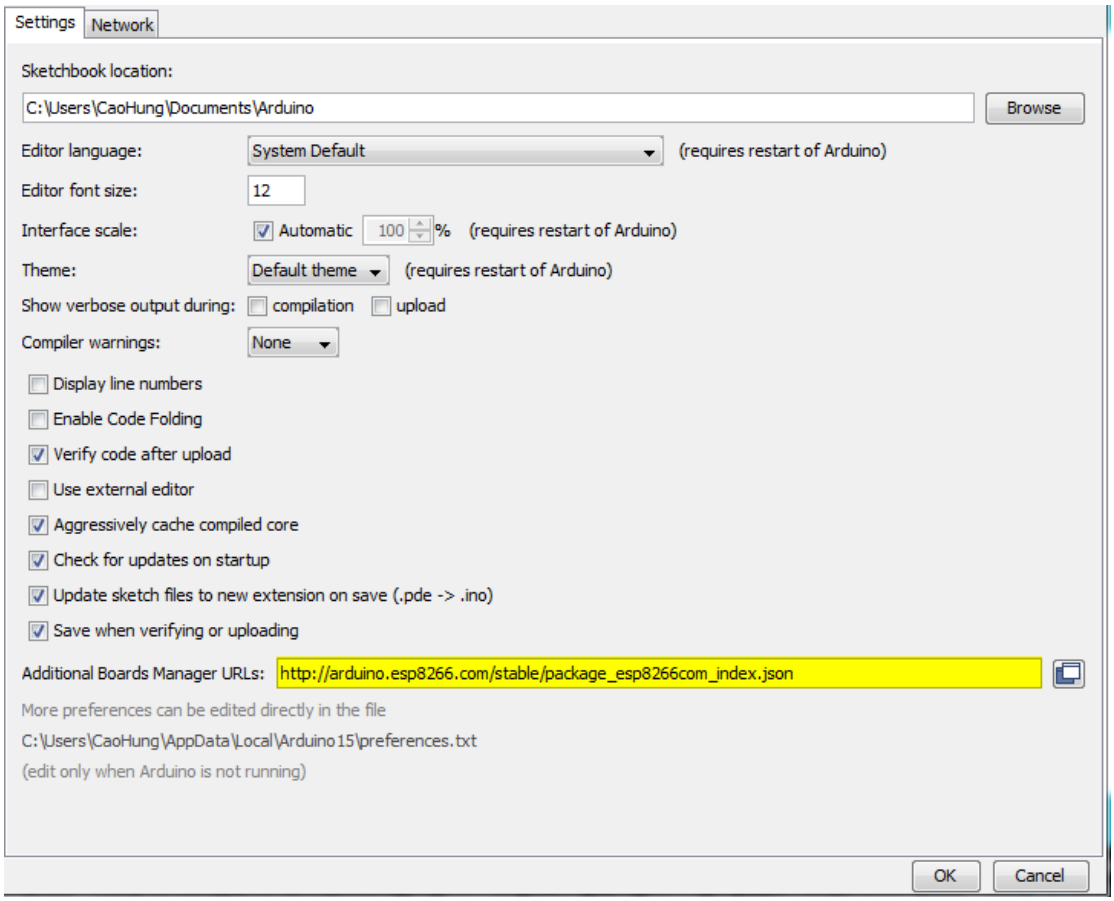
Hình 9.1.2: Sơ đồ chân ESP8266 ESP-01

Hướng dẫn nạp code cho ESP8266 NodeMCU

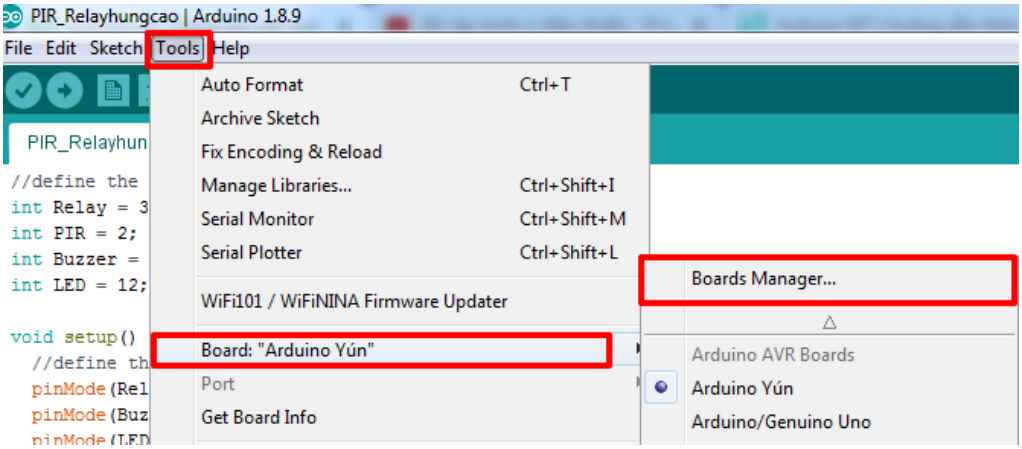
Bước 1: Mở Arduino IDE lên, click vào File trên thanh công cụ chọn Preferences (Ctrl+Comma).



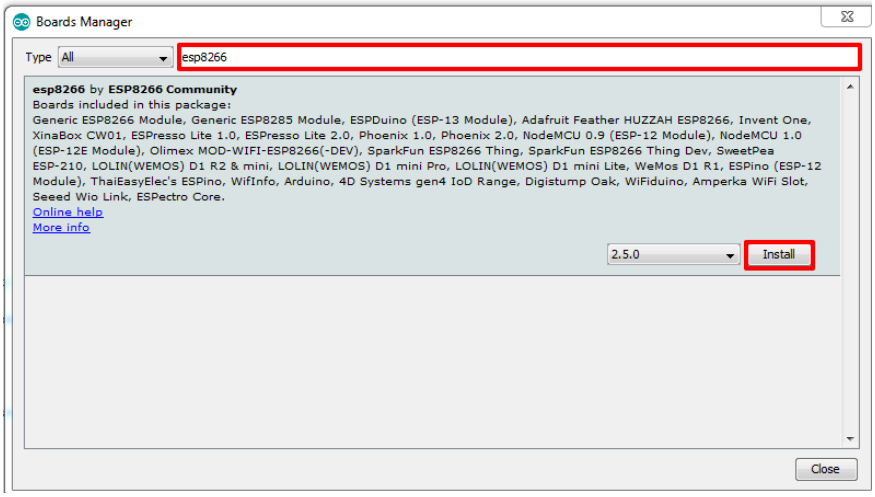
Bước 2: Copy đường Link bên dưới bỏ vào ô tô màu vàng và nhấn OK là xong.
Copy Link tại đây: http://arduino.esp8266.com/stable/package_esp8266com_index.json
chen-link-cho-esp8266

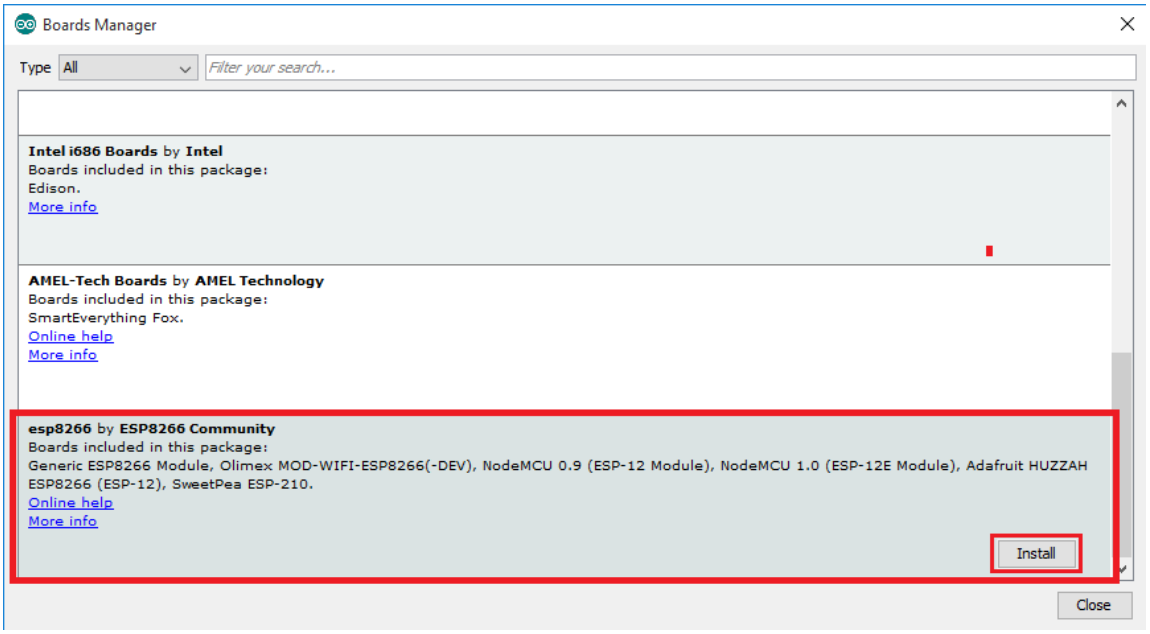


Vào Tools > Board > Boards Manager



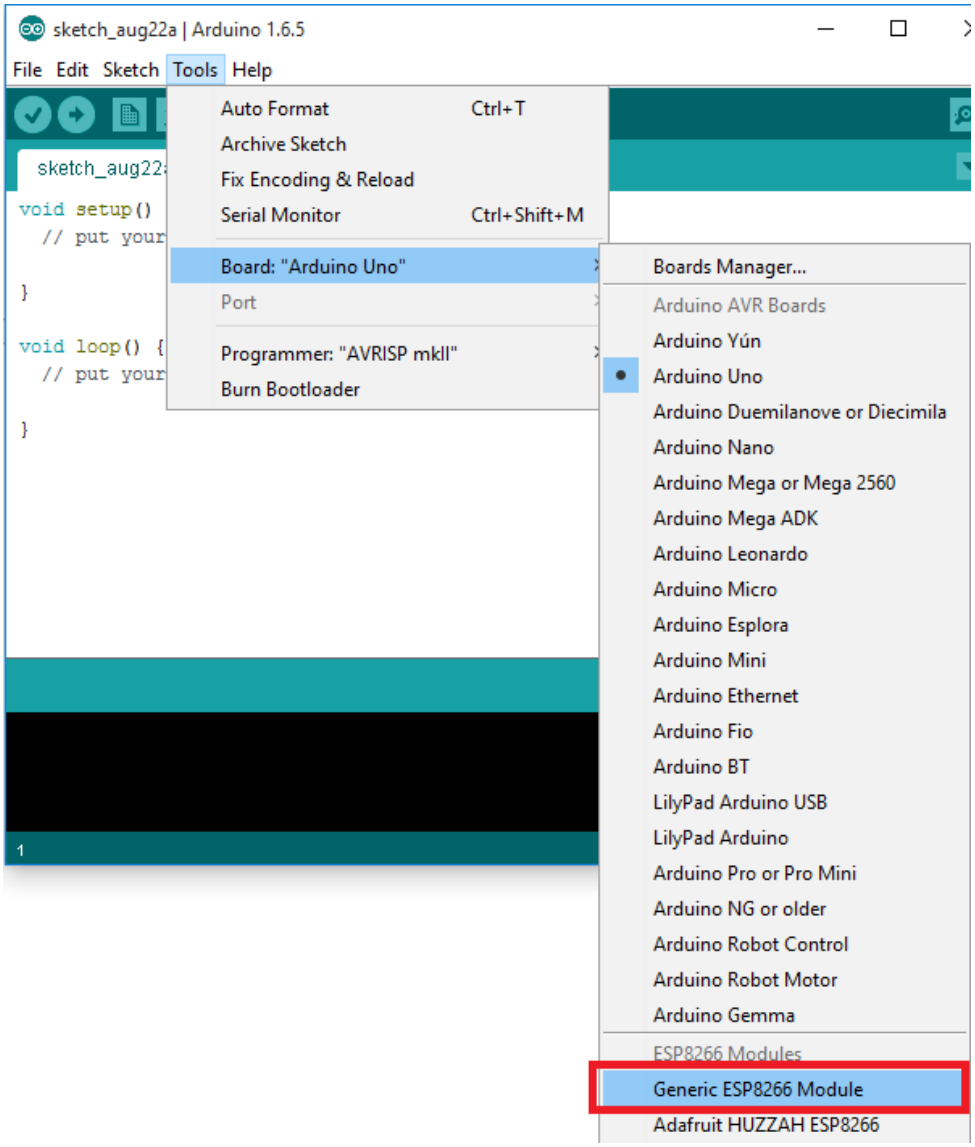
Cửa sổ mở lên ta Seach "Esp8266" để tải danh mục của các Board về. Nhấn Install để tiến hành cài đặt. Đợi một lát để chương trình tìm kiếm. Ta kéo xuống và click vào ESP8266 by ESP8266 Community, click vào Install. Chờ phần mềm tự động download và cài đặt.





Chọn Board để lập trình cho ESP8266:

Kết nối mudule USB-to-UART vào máy tính. Vào Tool→Board→Generic ESP8266 Module, chọn cổng COM tương ứng với module USB-to-UART tương ứng.



Sau khi kết nối UART vs ESP8266. có thể test code ESP8266 ở đây:

```
int pin = 2;

void setup() {

  pinMode(pin, OUTPUT);
}


void loop() {
```

```
  digitalWrite(pin, HIGH);   //bật led
  delay(1000);               //dừng 1s
  digitalWrite(pin, LOW);    //tắt led
  delay(1000);               //dừng 1s
}
```

2. Lập trình điều khiển qua wifi

2.1. Chuẩn bị dụng cụ, thiết bị vật liệu.

Cài sẵn Arduino IDE trên máy

Cài thư viện Blynk theo đường link đã gắn

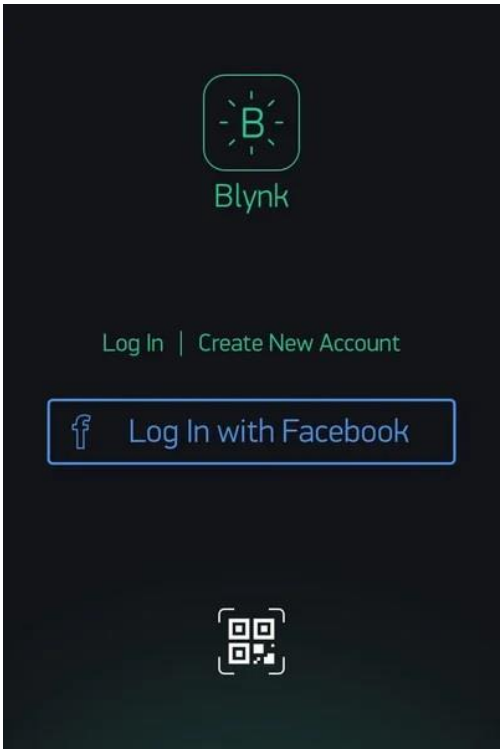
Chọn kết nối với ESP32 theo đúng board và đúng cổng COM

Cài đặt tiện ích hỗ trợ lập trình ESP32 trong Arduino IDE

Tạo một sketch mới để viết code ESP32 Blynk

1. Tải ứng dụng Blynk từ App Store hoặc Google Play

2. Sau khi tải xuống, bạn hãy mở ứng dụng và tạo tài khoản (Create New Account). Nếu đã có tài khoản, bạn nhấn vào Log In (đăng nhập). Lưu ý: Bạn cần nhập email chính xác, vì chúng ta sẽ dùng chúng để nhận mã xác thực ở bước tiếp theo.



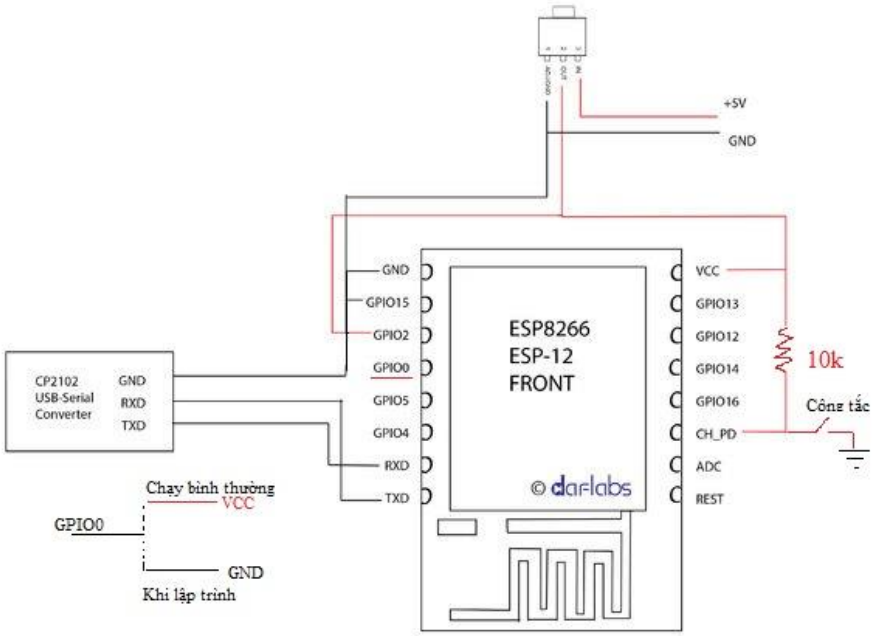
3. Tạo một dự án mới, chọn thiết bị là ESP32 và chọn loại kết nối là WiFi để bắt đầu dự án ESP32 Blynk này.

4. Sau khi nhấn nút tạo, một mã xác thực sẽ được gửi về email, bạn hãy mở email kiểm tra và xác thực nhé.

5. Click vào bất kỳ đâu trên màn hình để mở hộp tiện ích. Trong dự án ESP32 Blynk – điều khiển đèn LED này, chúng ta sẽ sử dụng nút (button). Bạn hãy kéo 1 nút ra màn hình chính nhé.

6. Click vào nút để thay đổi cài đặt. Trong phạm vi bài này, mình sẽ cấu hình là Digital_GP21, sau đó click chọn chế độ để chuyển đổi.

7. Cuối cùng, click vào nút PLAY. Giao diện sẽ chuyển từ chế độ EDIT (chỉnh sửa) sang chế độ PLAY. Ở PLAY, bạn có thể tương tác và điều khiển các thiết bị ngoại vi. Tuy nhiên, lúc này thì bạn không thể kéo và thả vào widget mới, bạn phải nhấn STOP và trở về chế độ EDIT thì mới làm được việc này.



2.2. Lập trình điều khiển qua wifi.

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>

MDNSResponder mdns;

ESP8266WebServer server(80);
String webPage;
const char* ssid     = "wifi";      //Thay tên wifi ở đây
const char* password = "password";  //Thay tên password ở đây

void setup() {
```

```
  pinMode(12, OUTPUT);  //led chân 12
  pinMode(13, OUTPUT);  //led chân 13

  webPage += "<h1>ESP8266 Web Server</h1><p>Socket #1 <a
href=\"socket1On\"><button>ON</button></a> <a
href=\"socket1Off\"><button>OFF</button></a></p>";
  webPage += "<p>Socket #2 <a
href=\"socket2On\"><button>ON</button></a> <a
href=\"socket2Off\"><button>OFF</button></a></p>";

  Serial.begin(115200);
  delay(100);


  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());

  if (mdns.begin("esp8266", WiFi.localIP()))
    Serial.println("MDNS responder started");

  server.on("/", [](){
    server.send(200, "text/html", webPage);
  });
  server.on("/socket1On", [](){
```

```
   server.send(200, "text/html", webPage);
   //Bật led 12
   digitalWrite(12, HIGH);
   delay(1000);
  });
  server.on("/socket1Off", []{
   server.send(200, "text/html", webPage);
   //Tắt led 12
   digitalWrite(12, LOW);
   delay(1000);
  });
  server.on("/socket2On", []{
   server.send(200, "text/html", webPage);
   //Bật led 13
   digitalWrite(13, HIGH);
   delay(1000);
  });
  server.on("/socket2Off", []{
   server.send(200, "text/html", webPage);
   //Tắt led 13
   digitalWrite(13, LOW);
   delay(1000);
  });
  server.begin();
  Serial.println("HTTP server started");
}

void loop() {
  server.handleClient();
}
```

Đoạn code trên sẽ đăng nhập vào wifi + password được chỉnh sửa sẵn. Nếu sai sẽ xuất hiện dấu chấm..... liên tục.

ESP8266 có 2 loại server:

ESP ở chế độ Access Point: ESP8266 tự trở thành 1 điểm phát wifi, có ip riêng khi truy cập vào (thường là 192.168.4.1) và cũng tạo được web page.
ESP truy cập vào wifi chung: Khi đó ESP8266 sẽ có địa chỉ riêng của mình, mở trình duyệt bất kỳ (Chrome, Firefox...) gõ vào ô địa chỉ ip sẽ vô webpage.

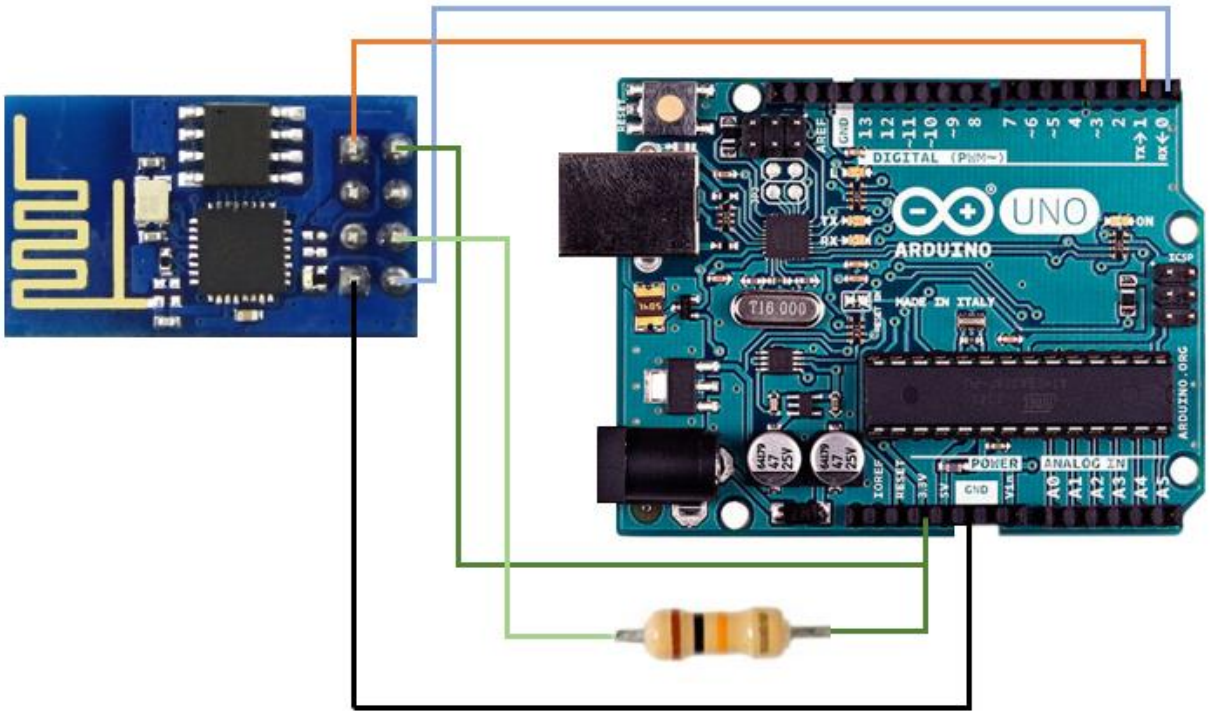
Cả 2 phương thức chỉ khác nhau về cách truy cập, còn code tạo web page giống nhau.

```
server.on("/", [](){
   server.send(200, "text/html", webPage);
 });
```

Server theo dõi đường link "/" nghĩa là trang chủ. Ví dụ ip của mình là 192.168.0.105/

```
server.on("/socket1On", [](){
   server.send(200, "text/html", webPage);
   //Bật led 12
   digitalWrite(12, HIGH);
   delay(1000);
 });
```

2.3. Kết nối dây dẫn và vận hành mạch.



2.4. Hư hỏng thường gặp, nguyên nhân và cách khắc phục.

STT	Hư hỏng thường gặp	Nguyên nhân	Cách khắc phục
1	ESP8266 không kết nối được với mạng Wi-Fi	Sai SSID hoặc mật khẩu Wi-Fi.	Kiểm tra và chắc chắn rằng SSID và mật khẩu Wi-Fi đúng.

		Tín hiệu Wi-Fi yếu hoặc không ổn định. Các thiết lập cấu hình mạng không đúng.	Đặt ESP8266 gần router Wi-Fi để cải thiện tín hiệu. Kiểm tra và cấu hình lại các thiết lập mạng cho đúng.
2	Không điều khiển được LED qua giao diện web hoặc ứng dụng	Lỗi trong mã lập trình, đặc biệt là trong phần xử lý yêu cầu từ giao diện web. Cổng GPIO không được thiết lập đúng. Kết nối phần cứng giữa ESP8266 và LED không đúng.	Kiểm tra và sửa lỗi mã lập trình. Đảm bảo rằng cổng GPIO được thiết lập chính xác trong mã. Kiểm tra kết nối phần cứng giữa ESP8266 và LED, đảm bảo rằng LED được kết nối đúng cực và chân GPIO.
3	LED nhấp nháy không đều hoặc không phản hồi	Nguồn điện không ổn định. Sử dụng GPIO không phù hợp (một số chân GPIO có chức năng đặc biệt trên ESP8266). Lỗi logic trong mã điều khiển.	Sử dụng nguồn điện ổn định, đảm bảo rằng ESP8266 và LED được cấp nguồn đủ. Sử dụng các chân GPIO khuyến nghị để điều khiển LED (ví dụ: GPIO2, GPIO4). Kiểm tra và sửa lỗi logic trong mã điều khiển, đảm bảo các lệnh bật/tắt LED được thực thi đúng.
4	Không tải được mã lập trình lên ESP8266	Cổng COM không đúng hoặc bị chiếm dụng bởi phần mềm khác. Kết nối USB không chắc chắn hoặc cáp USB kém chất lượng. Sai tốc độ baud hoặc thiết lập không đúng trong phần mềm IDE (Arduino IDE, PlatformIO, v.v.).	Kiểm tra và chọn đúng cổng COM trong phần mềm IDE. Sử dụng cáp USB chất lượng tốt và đảm bảo kết nối chắc chắn. Kiểm tra và thiết lập đúng tốc độ baud (thường là 115200) và các thiết lập khác trong phần mềm IDE.

BÀI 10: LẬP TRÌNH GIAO TIẾP LCD

Giới thiệu:

Màn hình LCD 16×2 là một linh kiện được sử dụng rộng rãi trong trong các dự án điện tử và lập trình.

LCD (Liquid Crystal Display) là màn hình tinh thể lỏng. LCD là loại thiết bị để hiển thị các ký tự, có cấu tạo bởi các điểm ảnh chứa các tinh thể lỏng (liquid crystal).

Màn hình LCD có ưu điểm là phẳng, cho hình ảnh sáng, chân thật và tiết kiệm năng lượng. Với Arduino, chúng ta có thể sử dụng LCD 16×2, gồm 2 dòng, mỗi dòng 16 ô ký tự. Vị trí các ô ký tự được đánh từ 0, 1, 2, 3, 4, 5,…,15.

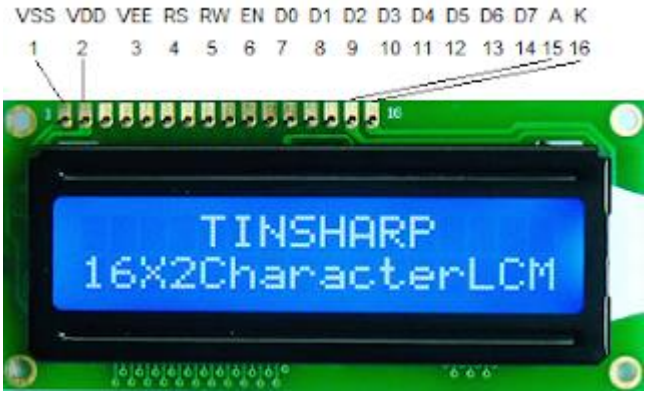
Mục tiêu của bài:

- Trình bày được cấu trúc của LCD
- Thiết kế và lập trình được các mạch ứng dụng dùng LCD
- Kiểm tra và sửa chữa lỗi hư hỏng
- Kết nối được chương trình với mô hình
- Phát huy tính chủ động trong học tập và trong công việc.

Nội dung chính:

1. Kết nối LCD.

1.1. Phân loại LCD.



Ngày nay, thiết bị hiển thị LCD 1602 (Liquid Crystal Display) được sử dụng trong rất nhiều các ứng dụng của VĐK. LCD 1602 có rất nhiều ưu điểm so với các dạng hiển thị khác như: khả năng hiển thị kí tự đa dạng (chữ, số, kí tự đồ họa); dễ dàng đưa vào mạch ứng dụng theo nhiều giao thức giao tiếp khác nhau, tiêu tốn rất ít tài nguyên hệ thống, giá thành rẻ,…

Thông số kĩ thuật của sản phẩm LCD 1602:

- Điện áp MAX : 7V
- Điện áp MIN : - 0,3V
- Hoạt động ổn định : 2.7-5.5V
- Điện áp ra mức cao : > 2.4
- Điện áp ra mức thấp : <0.4V

- Dòng điện cấp nguồn : 350uA - 600uA
- Nhiệt độ hoạt động : - 30 - 75 độ C

1.2. Sơ đồ chân và chức năng các chân của LCD.

Chân	Ký hiệu	Mô tả
1	Vss	Chân nối đất cho LCD, khi thiết kế mạch ta nối chân này với GND của mạch điều khiển
2	VDD	Chân cấp nguồn cho LCD, khi thiết kế mạch ta nối chân này với VCC=5V của mạch điều khiển
3	VEE	Điều chỉnh độ tương phản của LCD.
4	RS	Chân chọn thanh ghi (Register select). Nối chân RS với logic "0" (GND) hoặc logic "1" (VCC) để chọn thanh ghi. + Logic "0": Bus DB0-DB7 sẽ nối với thanh ghi lệnh IR của LCD (ở chế độ "ghi" - write) hoặc nối với bộ đếm địa chỉ của LCD (ở chế độ "đọc" - read) + Logic "1": Bus DB0-DB7 sẽ nối với thanh ghi dữ liệu DR bên trong LCD.
5	R/W	Chân chọn chế độ đọc/ghi (Read/Write). Nối chân R/W với logic "0" để LCD hoạt động ở chế độ ghi, hoặc nối với logic "1" để LCD ở chế độ đọc.
6	E	Chân cho phép (Enable). Sau khi các tín hiệu được đặt lên bus DB0-DB7, các lệnh chỉ được chấp nhận khi có 1 xung cho phép của chân E. + Ở chế độ ghi: Dữ liệu ở bus sẽ được LCD chuyển vào(chấp nhận) thanh ghi bên trong nó khi phát hiện một xung (high-to-low transition) của tín hiệu chân E. + Ở chế độ đọc: Dữ liệu sẽ được LCD xuất ra DB0-DB7 khi phát hiện cạnh lên (low-to-high transition) ở chân E và được LCD giữ ở bus đến khi nào chân E xuống mức thấp.
7 - 14	DB0 - DB7	Tám đường của bus dữ liệu dùng để trao đổi thông tin với MPU. Có 2 chế độ sử dụng 8 đường bus này : + Chế độ 8 bit : Dữ liệu được truyền trên cả 8 đường, với bit MSB là bit DB7. + Chế độ 4 bit : Dữ liệu được truyền trên 4 đường từ DB4 tới DB7, bit MSB là DB7
15	-	Nguồn dương cho đèn nền
16	-	GND cho đèn nền

2. Lập trình giao tiếp LCD.

2.1. Chuẩn bị dụng cụ, thiết bị vật liệu.

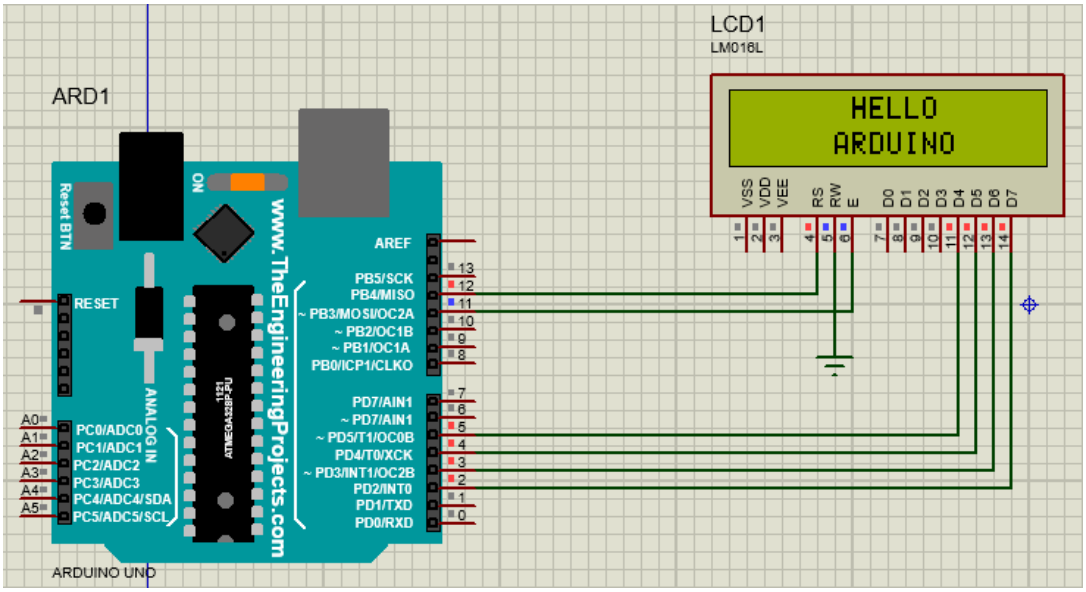
STT	Tên linh kiện	Số lượng	Ghi chú
1	Arduino UNO	1	
2	Breadboard	1	
3	Dây cắm breadboard	10	
4	Module LCD (16x02, 20x04,...	1	
5	Biến trở (1k, 4.7K, 10K,..	1	

1. VSS: tương đương với GND - cực âm
2. VDD: tương đương với VCC - cực dương (5V)
3. Constrast Voltage (Vo): điều khiển độ sáng màn hình
4. Register Select (RS): điều khiển địa chỉ nào sẽ được ghi dữ liệu
5. Read/Write (RW): Bạn sẽ đọc (read mode) hay ghi (write mode) dữ liệu? Nó sẽ phụ thuộc vào bạn gửi giá trị gì vào.
6. Enable pin: Cho phép ghi vào LCD
7. D0 - D7: 8 chân dư liệu, mỗi chân sẽ có giá trị HIGH hoặc LOW nếu bạn đang ở chế độ đọc (read mode) và nó sẽ nhận giá trị HIGH hoặc LOW nếu đang ở chế độ ghi (write mode)
8. Backlight (Backlight Anode (+) và Backlight Cathode (-)): Tắt bật đèn màn hình LCD.

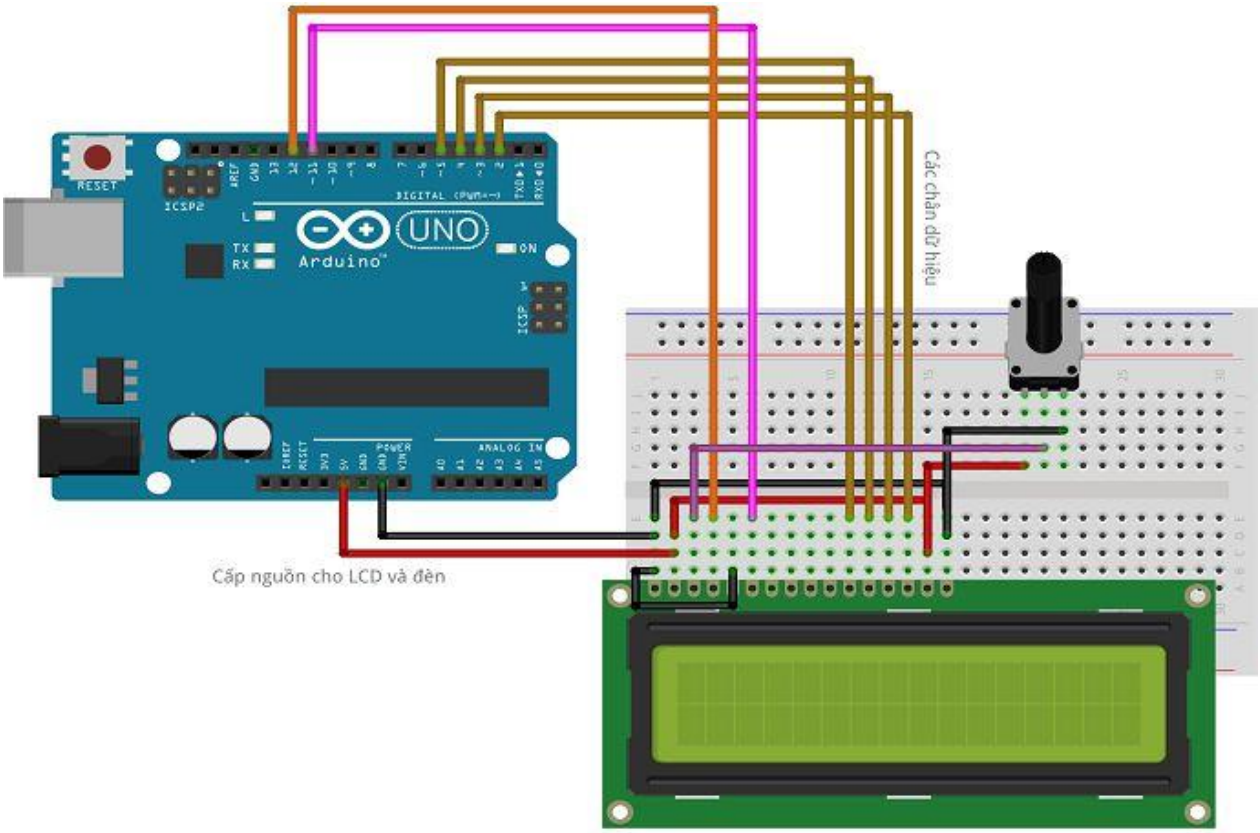
2.2. Lập trình giao tiếp LCD.

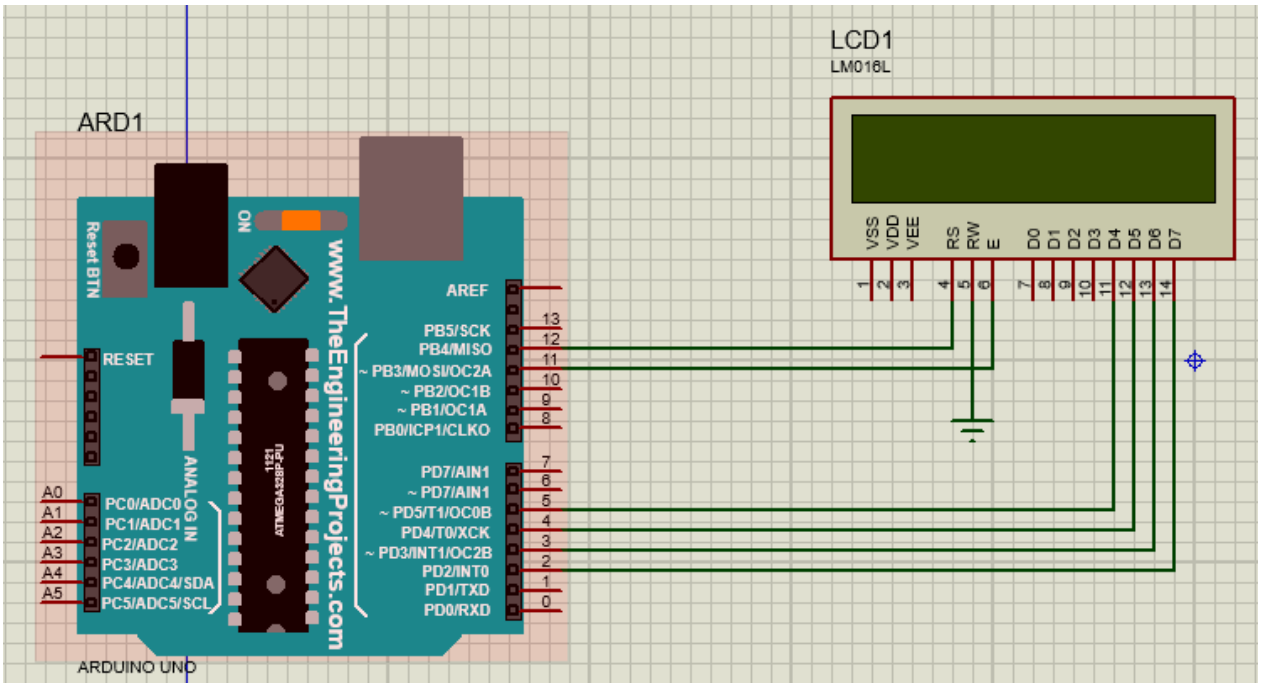
Thư viện LiquidCrystal là thư viện điều khiển LCD trên Arduino, nó được xây dựng để cho các bạn có thể lập trình điều khiển các module LCD ô vuông một cách nhanh chóng mà không cần phải lập trình nhiều. Thư viện này được viết để phù hợp với con IC HD44780 (con điều khiển module LCD), tuy nhiên, trên thị trường mình toàn thấy các con LCD của Trung Quốc và thư viện này vẫn hoạt động tốt. Nghĩa là, bạn chỉ cần mua module LCD về và gắn vào Arduino, nạp code là chạy được, không cần quan tâm đến IC điều khiển LCD.

```
#include <LiquidCrystal.h>//Khai báo thư viện
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);//Khai báo các chân RS, E, D4-D7 kết nối với Arduino
void setup() {
  lcd.begin(16, 2);//Khởi tạo màn hình LCD và xác định kích thước màn hình LCD là 16x2
}
void loop() {
  lcd.clear();//Xóa màn hình và đặt con trỏ về vị trí đầu tiên (0, 0)
  lcd.setCursor(6,0);//Di chuyển con trỏ đến cột tương ứng
  lcd.print("HELLO");//Xuất ra màn hình từ vị trí con trỏ
  lcd.setCursor(5,1);
  lcd.print("ARDUINO");
  delay(1000);
}
```



2.3. Kết nối dây dẫn và vận hành mạch.





Để LCD hoạt động:

Cần cấp nguồn dương (+) vào chân VDD của LCD, cấp nguồn âm (-) vào chân VSS.

Kết nối chân Baclight Anode với nguồn dương (+) và Backlight Cathode với nguồn âm (-) để điều khiển bật đèn màn hình.

Kết nối chân VEE với biến trở để điều khiển độ sáng màn hình.

Chân R/W kết nối với GND (R/W=0) để ghi dữ liệu vào LCD.

Kết nối chân RS và Enable với board mạch Arduino để giao tiếp với Arduino.

Điều khiển LCD ở chế độ 4 bit, kết nối 4 chân D4, D5, D6, D7 với board mạch Arduino.

Khi sử dụng mạch giả lập giao tiếp LCD với Arduino thì có thể không cần sử dụng chân VDD, VSS, VEE cũng như Backlight Anode (+) và Backlight Cathode (-).

2.4. Hư hỏng thường gặp, nguyên nhân và cách khắc phục.

STT	Hư hỏng thường gặp	Nguyên nhân	Cách khắc phục
1	Màn Hình Không Hiển Thị	Kết nối dây bị sai: Dây nối giữa Arduino và LCD có thể bị cắm sai chân hoặc lỏng. Điện áp cung cấp không đủ: LCD không nhận đủ nguồn điện để hoạt động. Thư viện LCD không được khai báo đúng: Không bao gồm đúng thư viện hoặc không khai báo đúng cấu hình trong mã nguồn.	Kiểm tra kết nối dây: Đảm bảo rằng tất cả các dây kết nối đúng theo sơ đồ mạch. Kiểm tra nguồn điện: Đảm bảo rằng nguồn điện cung cấp cho LCD là đủ (thường là 5V cho các màn hình LCD phổ biến). Kiểm tra thư viện và cấu hình: Kiểm tra lại mã nguồn để đảm bảo bạn đã bao gồm thư viện LCD (ví dụ: LiquidCrystal.h) và khai báo đúng các chân kết nối.
2	Màn Hình Hiển Thị Không Đúng Ký Tự	Lỗi mã nguồn: Mã nguồn có thể có lỗi hoặc không gửi đúng dữ liệu tới LCD. Lỗi thư viện: Phiên bản thư viện có thể không tương thích hoặc có lỗi. Lỗi giao tiếp: Tín hiệu giao tiếp giữa Arduino và LCD bị nhiễu.	Kiểm tra mã nguồn: Đảm bảo rằng mã nguồn không có lỗi và dữ liệu gửi tới LCD đúng định dạng. Cập nhật thư viện: Kiểm tra và cập nhật thư viện LCD lên phiên bản mới nhất. Kiểm tra giao tiếp: Đảm bảo rằng dây kết nối ngắn gọn, cách xa nguồn nhiễu và sử dụng dây chất lượng tốt.
3	Màn Hình Chớp Tắt Liên Tục	Nguồn điện không ổn định: Có sự dao động trong nguồn cung cấp điện. Lỗi phần mềm: Mã nguồn có thể có vòng lặp không	Kiểm tra nguồn điện: Sử dụng nguồn điện ổn định và kiểm tra dây kết nối. Tối ưu mã nguồn: Đảm bảo rằng mã nguồn được viết tối ưu

	tối ưu hoặc gây quá tải cho Arduino.	và không gây quá tải cho vi điều khiển.

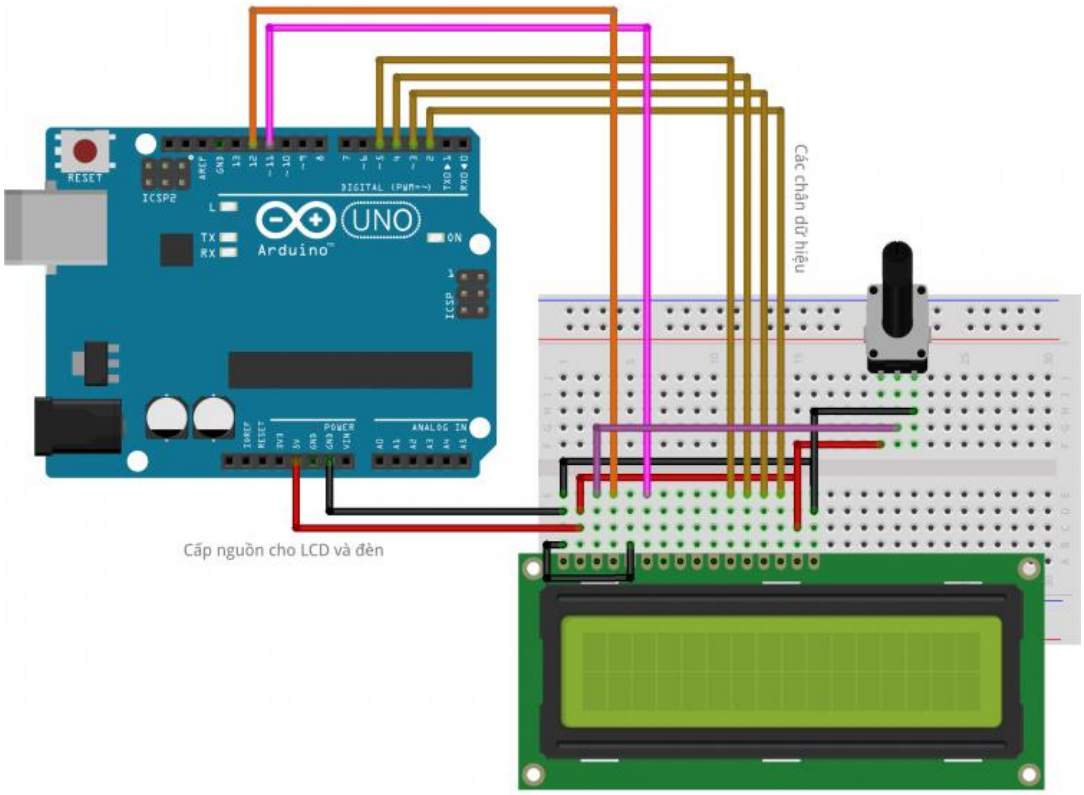
3. Bài tập.

3.1. Nhấp nháy chữ trên màn hình LCD.

*** Phần cứng**

1 con Arduino Uno R3
1 Text LCD 16x2(có thể dùng với module I2C)
1 Biến Trở 10k
1 Điện Trở 220 ohm
1 breadboard
Một số dây breadboard

*** Kết nối mạch**



*** Lập trình.**

```
//Thêm thư viện LiquidCrystal - nó có sẵn vì vậy bạn không cần cài thêm gì cả
#include <LiquidCrystal.h>

//Khởi tạo với các chân
```

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

void setup() {

  lcd.begin(16, 2);//Thông báo đây là LCD 1602
  lcd.print("Hello World!");//In ra dòng chữ, bạn có thể chỉnh chữ lại tuỳ
ý
  lcd.setCursor(0 , 1);
  lcd.print("arduino.vn");
}

void loop() {
lcd.noDisplay();//Xoá màn hình hiển thị
delay(500);//chờ 0,5 giây
lcd.Display();//Hiển thị trở lại
delay(500);
}
```

3.2. Chương trình tự dịch chuyển chữ ở dòng thứ 1 trên LCD khi bắt đầu có kí tự ở dòng thứ 2

```
// Thêm thư viện
#include <LiquidCrystal.h>

//Khai báo các chân LCD
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  // Cấu hình hàng và cột LCD
  lcd.begin(16, 2);
}

void loop() {
  // Đưa con trỏ về vị trí (0,0):
  lcd.setCursor(0, 0);
  // In ra giá trị từ 0 - 9:
  for (int thisChar = 0; thisChar < 10; thisChar++) {
```

```
    lcd.print(thisChar);
    delay(500);
  }

  // Đặt con trỏ tới vị trí (16,1):
  lcd.setCursor(16, 1);
  // Cấu hình hiển thị tự cuộn chữ:
  lcd.autoscroll();
  //IN giá trị từ 0 - 9:
  for (int thisChar = 0; thisChar < 10; thisChar++) {
    lcd.print(thisChar);
    delay(500);
  }
  // Tắt chức năng tự động cuộn
  lcd.noAutoscroll();

  // Xóa màn hình cho vòng lặp kế tiếp
  lcd.clear();
}
```

BÀI 11: LẬP TRÌNH GIAO TIẾP ADC

Giới thiệu:

ADC là viết tắt của Analog to Digital Converter. ADC được sử dụng để chuyển đổi đầu vào tương tự (thường là điện áp) sang dạng kỹ thuật số. Điều cần thiết là mọi vi điều khiển phải có ADC, vì tất cả các vi điều khiển chỉ hoạt động trên điện áp đầu vào và đầu ra kỹ thuật số.

Vì vậy, ADC lấy điện áp tương tự và chuyển đổi chúng thành kỹ thuật số, và đưa chúng đến bộ vi điều khiển. Trong bảng Arduino UNO, có một bộ ADC 10-bit đa kênh. Ở đây, 10-bit có nghĩa là điện áp đầu vào 0-3.3V hoặc 0-5V được ánh xạ thành các giá trị kỹ thuật số trong phạm vi từ 0 đến 1023 (Vì 2^10 = 1024).

Trên Arduino UNO có tổng cộng 6 chân ADC. Các chân này là A0, A1, A2, A3, A4 và A5. Để hiểu điều này một cách dễ dàng nhất, chúng ta sẽ tạo một mạch sử dụng chiết áp (potentiometer)và bo mạch Arduino UNO.

Mục tiêu của bài:

- Trình bày được công thức chuyển đổi của ADC
- Thiết kế và lập trình được các mạch ứng dụng dùng ADC
- Kiểm tra và sửa chữa lỗi hư hỏng
- Kết nối được chương trình với mô hình
- Phát huy tính chủ động trong học tập và trong công việc.

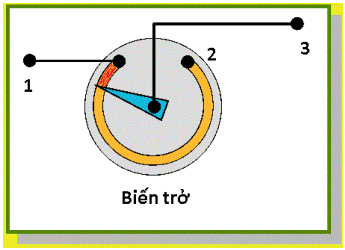
Nội dung chính:

1. Giao tiếp ADC.

Hầu hết tất cả các vi điều khiển hiện đại đều có ADC tích hợp sẵn, phổ biến nhất là Arduino dựa trên ATMega328P với độ phân giải 10 bit và STM32 với độ phân giải 12 bit.

Arduino IDE cung cấp hàm ‘analogRead ()’ để đọc điện áp analog trên một trong các chân analog và trả về giá trị số nguyên 10 bit, tức là phạm vi từ 0 đến 1023.

Biến trở chỉ đơn giản chỉ là một điện trở có thể thay đổi được trị số. Mạch Arduino không đọc trực tiếp điện trở này mà đọc gián tiếp qua điện áp mà biến trở gây ra.



Phần màu vàng là một lớp điện trở. Cây kim màu xanh được đè chặt xuống phần điện trở này. Giả sử có dòng điện đi từ 1 đến 3 thì nó sẽ phải qua phần màu vàng (được tô đỏ) và đó chính là điện trở hiện tại của biến trở. Chỉ việc vặn cây kim để tăng giảm độ dài của vùng màu đỏ, qua đó tăng giảm giá trị điện trở.

Giả sử đặt một hiệu điện thế vào 2 cực 1 và 2, sử dụng công thức định luật Ôm, ta có thể tính được điện áp lấy ra ở cực 3. Khi vặn biến trở, ta sẽ làm thay đổi điện trở ở phần màu đỏ và màu vàng (do điện tích của chúng thay đổi), qua đó làm thay đổi điện áp ở chân 3. Những cái volume vặn âm thanh to / nhỏ cũng có nguyên lí hoạt động tương tự như vậy.

Người ta gọi hệ 2 điện trở này là cầu phân áp, tức là phân chia điện áp nhờ một cầu điện trở.

2. Lập trình giao tiếp ADC.

2.1. Chuẩn bị dụng cụ, thiết bị vật liệu.

Mạch Arduino (trong bài sử dụng Arduino UNO R3).

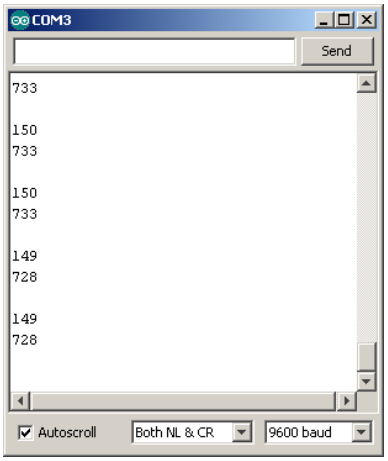
Breadboard.

Biến trở (bạn có thể chọn loại bất kì).

2.2. Lập trình giao tiếp ADC.

```
void setup() {
  Serial.begin(9600);  //Mở cổng Serial để giap tiếp | tham khảo Serial
}

void loop() {
  int value = analogRead(A0);   //đọc giá trị điện áp ở chân A0
                          //(value luôn nằm trong khoảng 0-1023)
  Serial.println(value);        //xuất ra giá trị vừa đọc

  int voltage;
  voltage = map(value,0,1023,0,5000);   //chuyển thang đo của value
                                  //từ 0-1023 sang 0-5000 (mV)
  Serial.println(voltage);              //xuất ra điện áp (đơn vị là mV)

  Serial.println();     //xuống hàng
  delay(200);           //đợi 0.2 giây
}
```

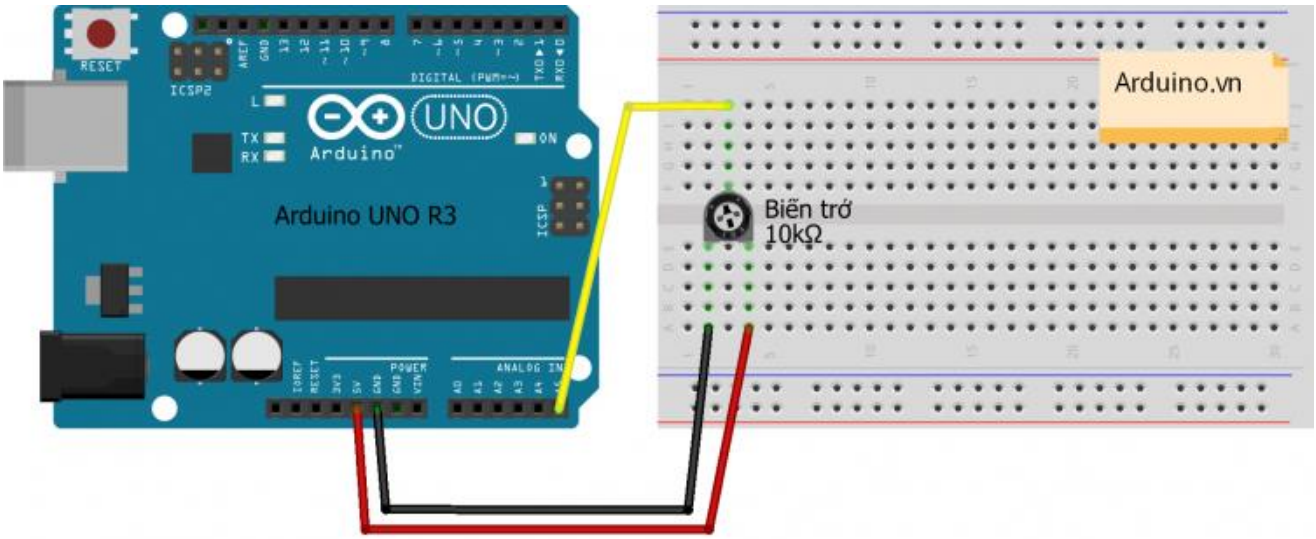
Bấm Ctrl + Shift + M để mở cửa sổ Serial Monitor và xem kết quả.

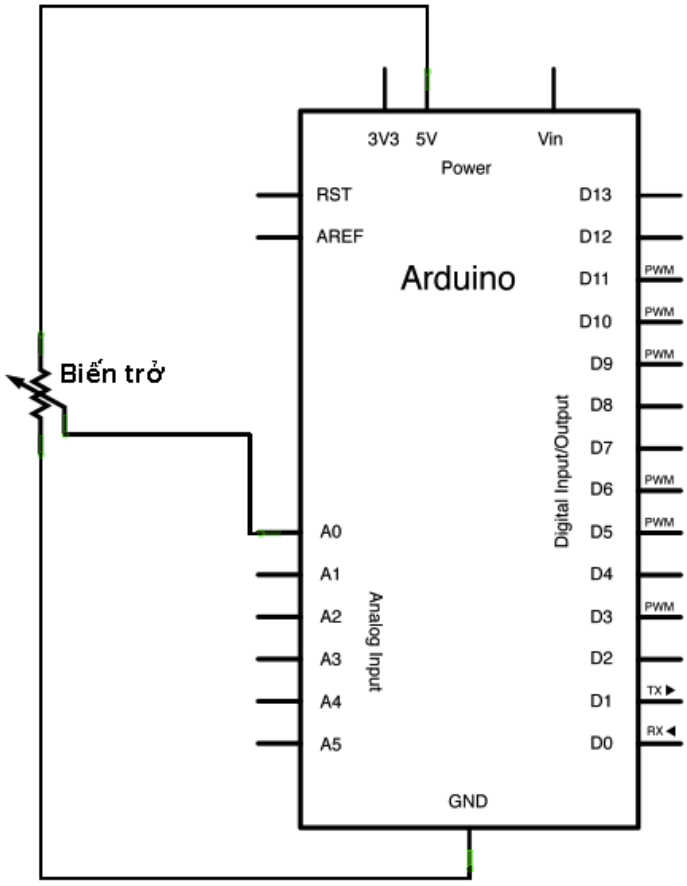


Dòng đầu tiên là giá trị điện áp đọc được bằng lệnh analogRead();

Dòng thứ 2 là giá trị điện áp (tính bằng mV) sau khi dùng hàm map() để chuyển đổi

2.3. Kết nối dây dẫn và vận hành mạch.





2.4. Hư hỏng thường gặp, nguyên nhân và cách khắc phục

STT	Hư hỏng thường gặp	Nguyên nhân	Cách khắc phục

1	Kết Quả Đọc ADC Không Chính Xác	Nhiễu điện tử: Nhiễu từ các thiết bị khác hoặc từ chính mạch Arduino. Nguồn điện không ổn định: Điện áp cung cấp cho Arduino hoặc cảm biến không ổn định. Không hiệu chỉnh cảm biến: Cảm biến hoặc mạch đo không được hiệu chỉnh đúng cách.	Giảm nhiễu: Sử dụng các tụ lọc (capacitor) để lọc nhiễu trên các đường tín hiệu và nguồn điện. Đặt Arduino và cảm biến cách xa các nguồn nhiễu điện tử. Ổn định nguồn điện: Đảm bảo rằng nguồn điện cung cấp cho Arduino và cảm biến là ổn định và đủ mạnh. Sử dụng bộ nguồn chất lượng tốt. Hiệu chỉnh cảm biến: Hiệu chỉnh cảm biến trước khi sử dụng để đảm bảo độ chính xác của kết quả đo.
2	Kết Quả Đọc ADC Thay Đổi Đột Ngột	Kết nối dây không chắc chắn: Các dây kết nối bị lỏng hoặc không cắm đúng. Sai cấu hình: Sai cài đặt trong mã nguồn, không chọn đúng chân ADC hoặc không cài đặt đúng giá trị tham chiếu.	Kiểm tra kết nối: Đảm bảo rằng tất cả các dây kết nối chắc chắn và đúng chân. Kiểm tra cấu hình: Đảm bảo rằng mã nguồn cài đặt đúng chân ADC và giá trị tham chiếu.
3	Giá Trị ADC Không Thay Đổi	Cảm biến không hoạt động: Cảm biến hoặc mạch đo có vấn đề. Mã nguồn sai: Mã nguồn không đọc đúng chân hoặc có lỗi trong logic.	Kiểm tra cảm biến: Đảm bảo rằng cảm biến hoạt động bình thường. Thử thay thế bằng cảm biến khác nếu có. Kiểm tra mã nguồn: Đảm bảo rằng mã nguồn đọc đúng chân và không có lỗi logic.

3. Bài tập.

Bài Tập 1: Đọc Giá Trị Cảm Biến Nhiệt Độ LM35

```
const int sensorPin = A0; // Chân cảm biến nối với A0

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

```
void loop() {
  int sensorValue = analogRead(sensorPin); // Đọc giá trị từ chân cảm biến
  float voltage = sensorValue * (5.0 / 1023.0); // Chuyển đổi giá trị ADC sang điện áp
  float temperatureC = voltage * 100; // LM35: 10mV/°C
  Serial.print("Temperature: ");
  Serial.print(temperatureC);
  Serial.println(" °C");
  delay(1000);
}
```

Bài Tập 2: Đo Ánh Sáng Sử Dụng Cảm Biến Quang Trở (LDR)

```
const int sensorPin = A0; // Chân cảm biến nối với A0

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

void loop() {
  int sensorValue = analogRead(sensorPin); // Đọc giá trị từ chân cảm biến
  Serial.print("Light Intensity: ");
  Serial.println(sensorValue);
  delay(1000);
}
```

TÀI LIỆU THAM KHẢO

[1] Giáo trình arduino-Trường đại học Sư Phạm Kỹ Thuật TPHCM
[2] Giáo trình vi điều khiển-Vụ trung học chuyên nghiệp và dạy nghề
[3] Datasheet atmega 328
[4] http://arduino.vn/reference
[5] Giáo trình Chuyên đề Arduino và truyền thông – Trường Cao đẳng kinh tế kỹ thuật.