



Sensor Module Starter Kit



Content

Preface	3
Packing list	4
How to Install Arduino IDE	5
How to Install Arduino Driver	13
How to Add Arduino Libraries	22
Blink Test	28
Lesson 1 0.96 OLED Display Module	38
Lesson 2 1602 I2C Module	42
Lesson 3 Temp and Humidity module	44
Lesson 4 Soil Humidity Sensor	50
Lesson 5 Infrared Obstacle Avoidance Module	54
Lesson 6 PIR Motion Sensor	58
Lesson 7 HC-SR04 Ultrasonic Sensor	61
Lesson 8 Capacitive Touch	64
Lesson 9 Joystick Module	67
Lesson 10 5V Relay Module	69
Lesson 11 Servo	72
Lesson 12 8X8 LED Dot Matrix Module	75
Lesson 13 Four Digital Seven Segment Display	82



Preface

Company Profile

Founded in 2014, Shenzhen Lonten Technology Co., Ltd. focuses on the design, research production of Electronics Module for robotics related products. Consisting of professional researchers and skilled engineers, our R&D team constantly strives for creative function and excellent user experience. The company's R&D investments on arduino kits raspberry pi kits, as well as 3D printer and robots that back up STEAM education.

Customer Service

Our self-owned factory is certificated with BSCI and SO, covering an area of 5,000 square meters, and achieving an annual production capacity of over 10,000 units. Our products are all certified to CE, FCC, and ROHS standards, have exported to more than 100 countries including, but not limited to France, the United States of America, Australia, Russia, the United Kingdom, Germany, Singapore, Egypt, and India, bringing technological innovation to all walks of life.

Tutorial

This tutorial include codes, libraries and detailed user documentation. It is designed for beginners. You will learn all the basic knowledge about how to use Arduino controller board, sensors and components.

Packing list

			
R3 CH340 x1	BreadBoard (4.5x3.5cm) x2	IIC LCD 1602 x1	Ultrasonic Module x1
			
SG90 Servo x1	1-Way Relay Module x1	HC-SR501 PIR Motion Sensor x1	0.96 OLED x1
			
8*8 Red Dot Matrix Screen x1	Joystick Module x1	DHT11 Module x1	0.96 inch OLED (Yellow & Blue) x1
			
TTP223B Touch Sensor x1	Obstacle Avoidance Module x1	Green/ Blue/ Yellow/ Red each one x5 (all 20pcs)	Resistor-220R/1k/10k x10 (all 30pcs)
			
Soil Humidity Sensor x1	F-M DuPont Cable x20	M-M DuPont Cable x40	USB Cable x1



How to Install Arduino IDE

Introduction

The Arduino Integrated Development Environment (IDE) is the software side of the Arduino platform.

In this Project, you will learn how to setup your computer to use Arduino and how to set about the Projects that follow.

The Arduino software that you will use to program your Arduino is available for Windows, Mac and Linux. The installation process is different for all three platforms and unfortunately there is a certain amount of manual work to install the software.

STEP 1: Go to <https://www.arduino.cc/en/software>.

The screenshot shows the Arduino IDE 2.1.1 download page. On the left, there is a section with the Arduino logo and the title "Arduino IDE 2.1.1". Below the title, it says "The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger." It then says "For more details, please refer to the [Arduino IDE 2.0 documentation](#)." Below that, it says "Nightly builds with the latest bugfixes are available through the section below." At the bottom of this section, it says "SOURCE CODE" and "The Arduino IDE 2.0 is open source and its source code is hosted on [GitHub](#)." On the right, there is a section titled "DOWNLOAD OPTIONS". It lists three operating systems: Windows, Linux, and macOS. For each, it provides the version requirements and the download format. For Windows, it says "Win 10 and newer, 64 bits" and lists "MSI installer" and "ZIP file". For Linux, it says "ApplImage 64 bits (X86-64)" and "ZIP file 64 bits (X86-64)". For macOS, it says "Intel, 10.14: 'Mojave' or newer, 64 bits" and "Apple Silicon, 11: 'Big Sur' or newer, 64 bits". At the bottom of the download options section, it says "Release Notes".

DOWNLOAD OPTIONS	
Windows	Win 10 and newer, 64 bits
Windows	MSI installer
Windows	ZIP file
Linux	ApplImage 64 bits (X86-64)
Linux	ZIP file 64 bits (X86-64)
macOS	Intel, 10.14: "Mojave" or newer, 64 bits
macOS	Apple Silicon, 11: "Big Sur" or newer, 64 bits
Release Notes	

The version available at this website is usually the latest version, and the actual version may be newer than the version in the picture.

STEP2: Download the development software that is compatible with the operating.

system of your computer. Take Windows as an example here.







Click Windows Win 10 and newer,64 bits.



Click JUST DOWNLOAD.

Also version 2.1.1 is available in the material we provided, and the versions of our materials are the latest versions when this course was made.

 arduino-ide_2.1.1_Linux_64bit
 arduino-ide_2.1.1_macOS_64bit
 arduino-ide_2.1.1_Windows_64bit
 arduino-ide_2.1.1_Windows_64bit

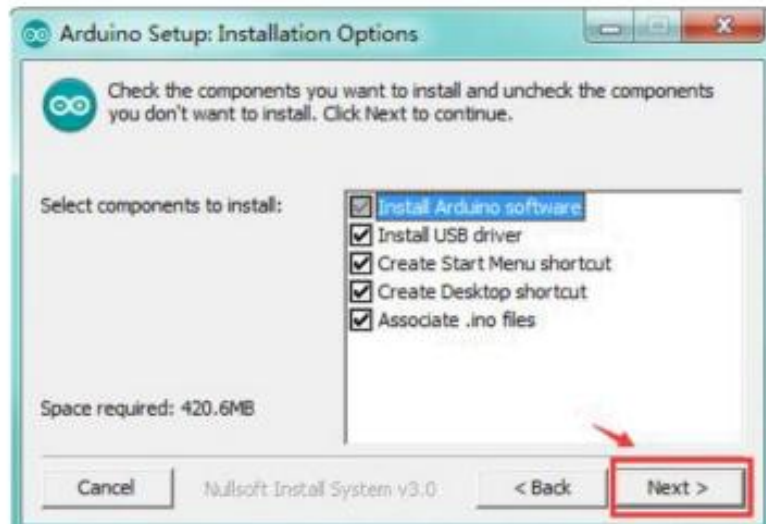
Installing Arduino (Windows)

Install Arduino with the exe. Installation package.

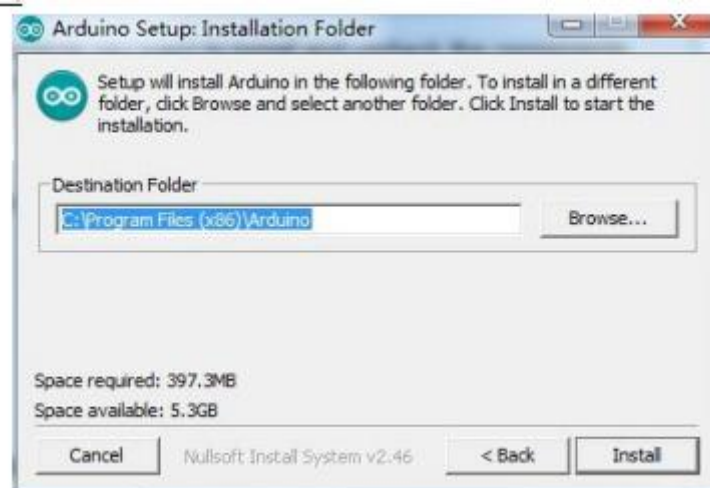
 arduino-ide_2.1.1_Windows_64bit



Click I Agree to see the following interface.



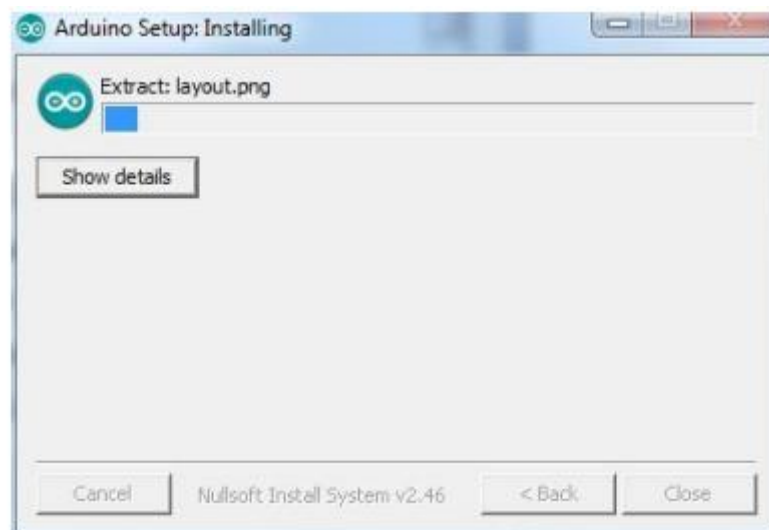
Click Next



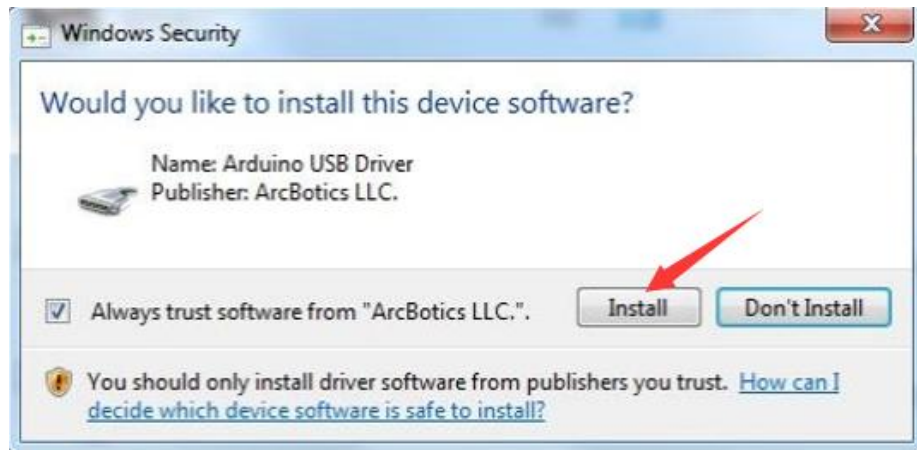
You can press Browse... to choose an installation path or directly type in the directory you want.



Click Install to initiate installation



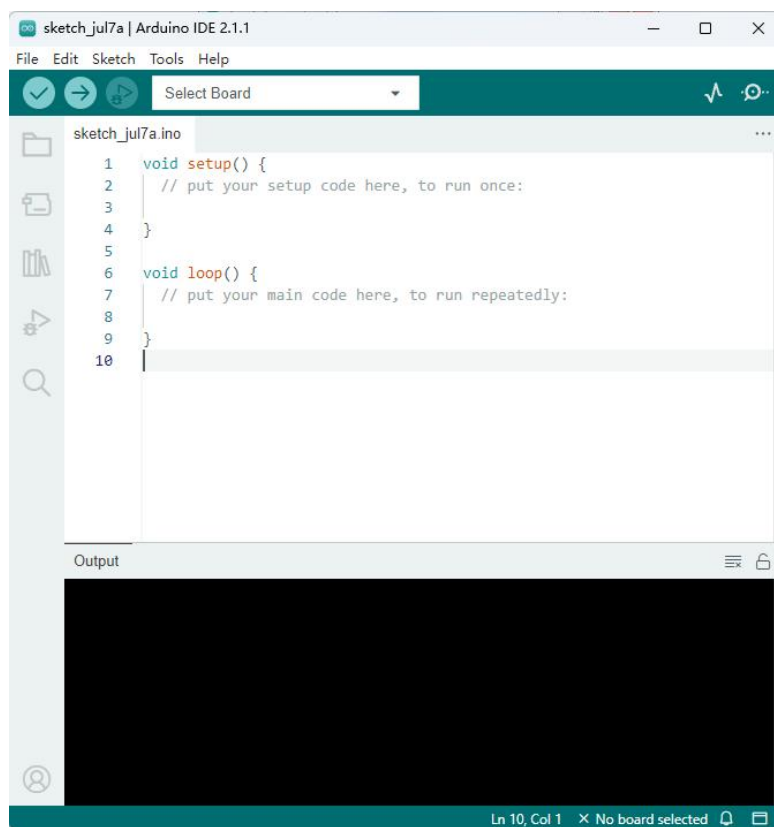
Finally, the following interface appears, click Install to finish the installation.



Next, the following icon appears on the desktop



Double-click to enter the desired development environment



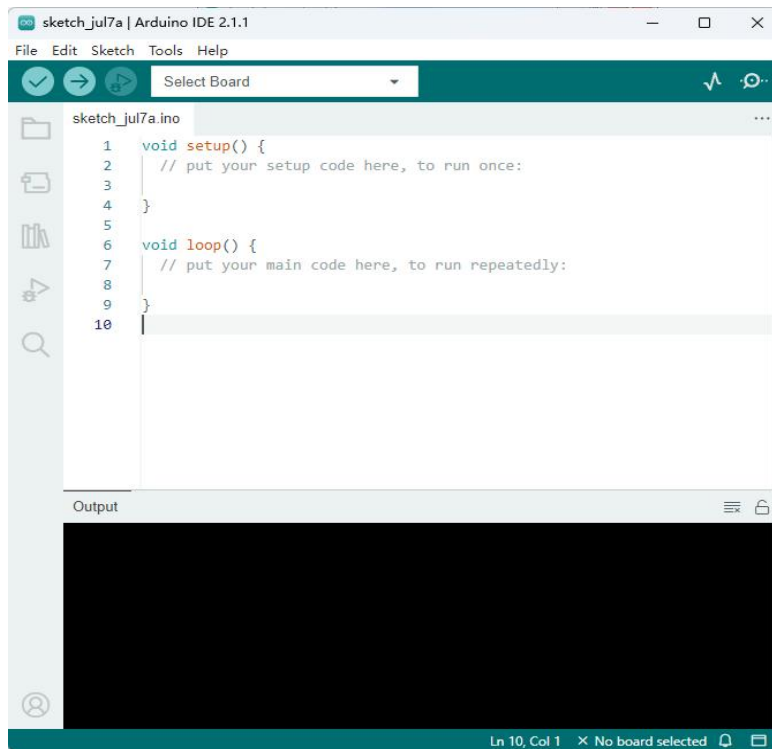
You may directly choose the installation package for installation and

skip the contents below and jump to the next section. But if you want to learn some methods other than the installation package, please continue to read the section.

Unzip the zip file downloaded, Double-click to open the program and enter the desired development environment.

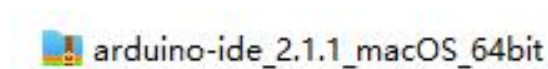
arduino-ide_2.1.1_Windows_64bit

名称	修改日期	类型	大小
drivers	2023/7/5 21:45	文件夹	
examples	2023/7/5 21:45	文件夹	
hardware	2023/7/5 21:45	文件夹	
java	2023/7/5 21:45	文件夹	
lib	2023/7/5 21:45	文件夹	
libraries	2023/7/5 21:45	文件夹	
reference	2023/7/5 21:45	文件夹	
tools	2023/7/5 21:45	文件夹	
tools-builder	2023/7/5 21:45	文件夹	
arduino	2017/6/1 0:58	应用程序	395 KB
arduino.l4j	2017/6/1 0:58	配置设置	1 KB
arduino_debug	2017/6/1 0:58	应用程序	393 KB
arduino_debug.l4j	2017/6/1 0:58	配置设置	1 KB
arduino-builder	2017/6/1 0:58	应用程序	3,214 KB
libusb0.dll	2017/6/1 0:58	应用程序扩展	43 KB
msvcp100.dll	2017/6/1 0:58	应用程序扩展	412 KB
msvcr100.dll	2017/6/1 0:58	应用程序扩展	753 KB
revisions	2017/6/1 0:58	文本文档	83 KB
uninstall	2023/7/5 21:45	应用程序	404 KB



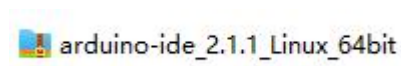
Installing Arduino (Mac OS X)

Download and Unzip the zip file, double click the Arduino.app to enter Arduino IDE; the system will ask you to install Java runtime library if you don't have it in your computer. Once the installation is complete you can run the Arduino IDE.



Installing Arduino (Linux)

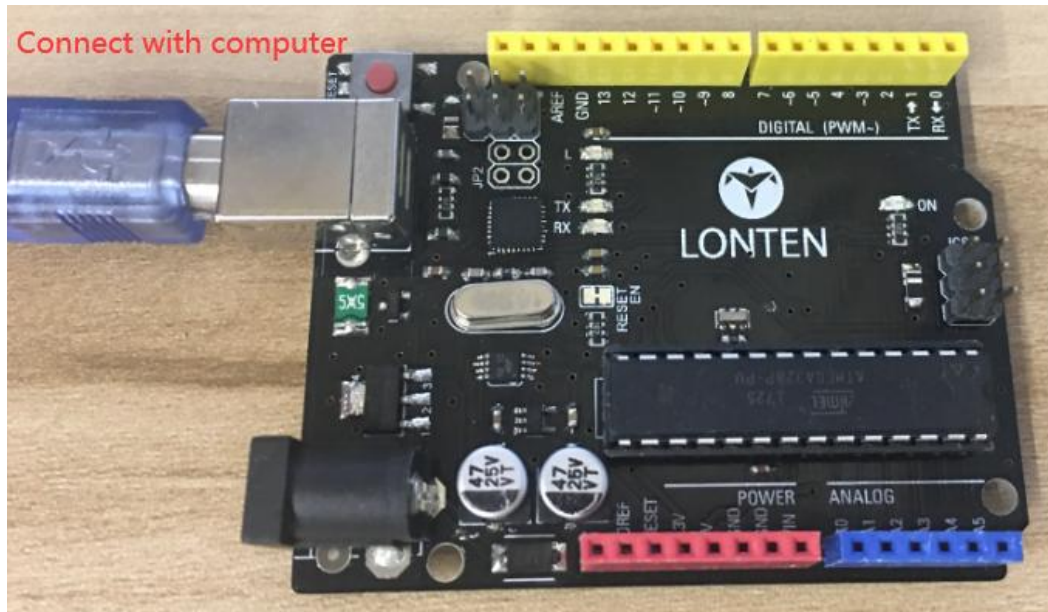
You will have to use the make install command. If you are using the Ubuntu system, it is recommended to install Arduino IDE from the software center of Ubuntu.



How to Install Arduino Driver

For Windows

Arduino UNO



The USB to serial port chip of this control board is CH340G. So you need to install the driver for the chip. You can click the driver file [here](#).

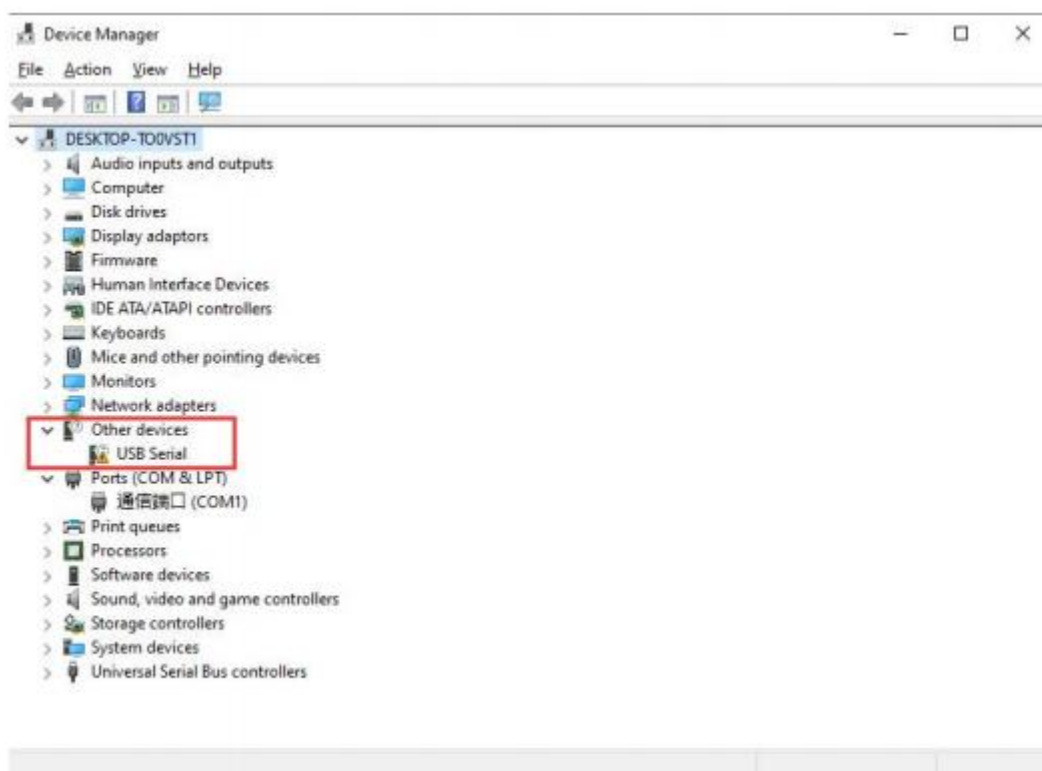
In different systems, the driver installation is similar. Here we start to install the driver on the Win10 system. You can find the “USB_Drive_CH341_3_1” folder in the information we provide, this is the driver file we want to install.



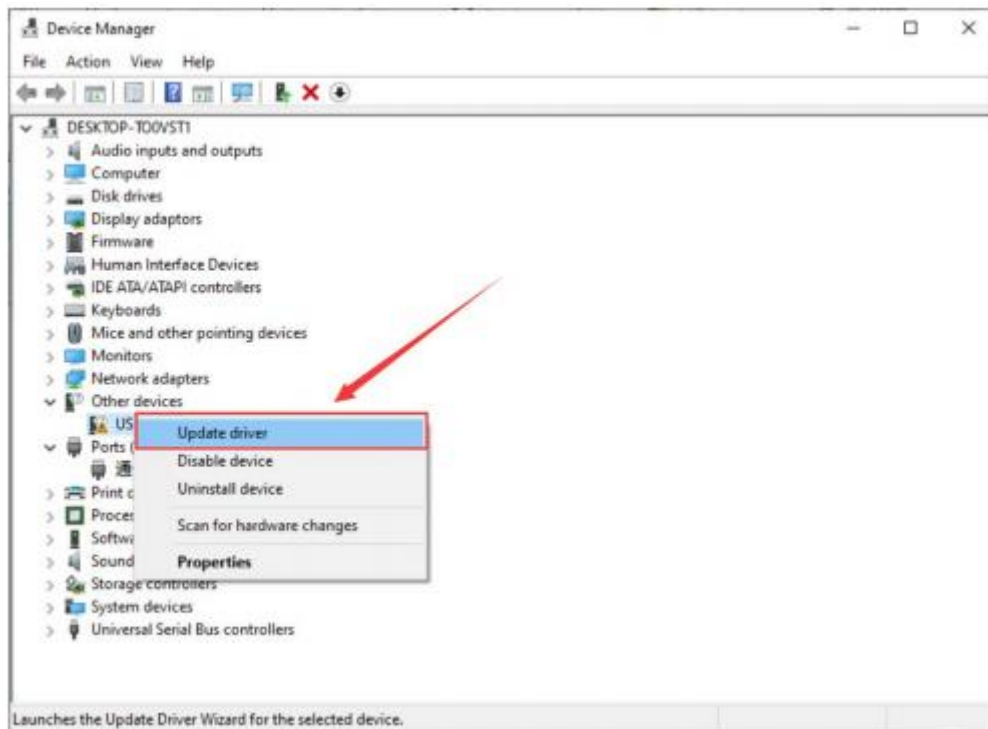
USB_Drive
_CH341_3_
1_For_Win
dows

Plug one end of your USB cable into the Arduino UNO CH340 Board and the other into a USB socket on your computer.

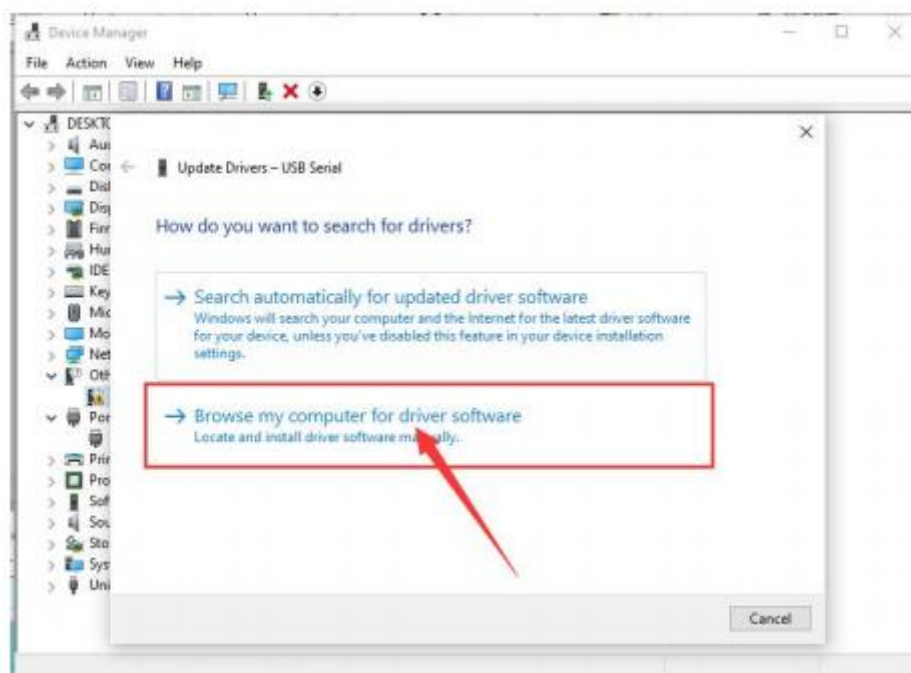
When you connect the Arduino UNO CH340 Board to your computer at the first time, right click your “My Computer”—>for “Properties”—>click the “Device manager”, under Other devices, you should see the “USB-Serial” or “Unknown device ”.Or you can search for "devi" in your computer, or you can open the device manager of your computer.



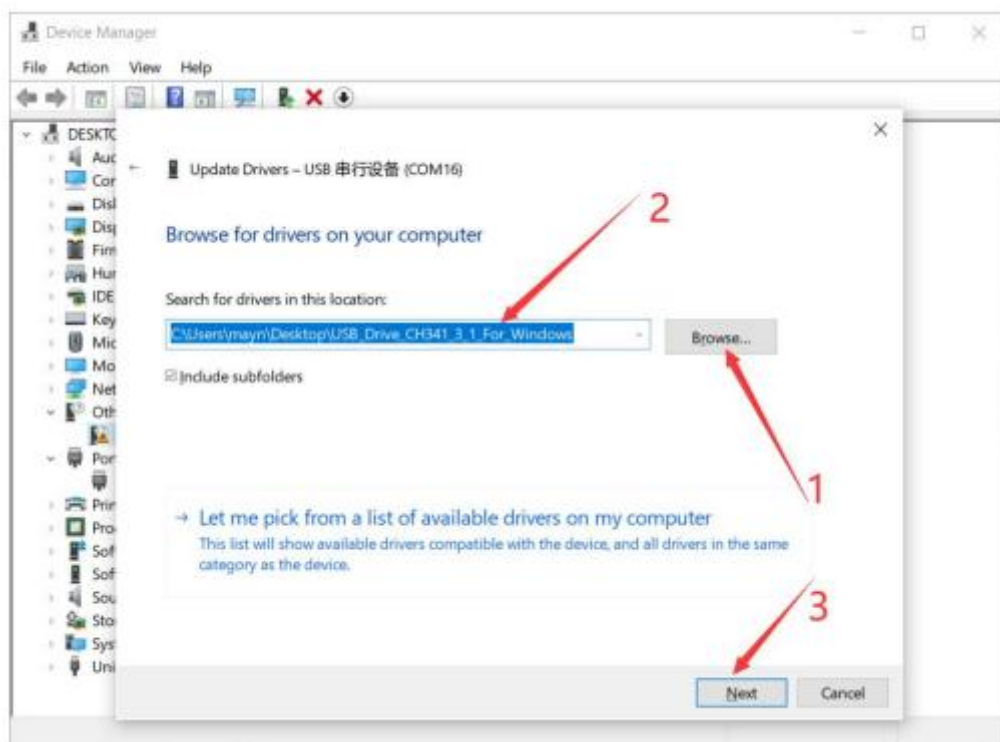
Then right-click on the device and select the top menu option (Update Driver Software...) shown as the figure below.



Then it will be prompted to either “Search Automatically for updated driver software” or “Browse my computer for driver software”. Shown as below. In this page, select “Browse my computer for driver software”.

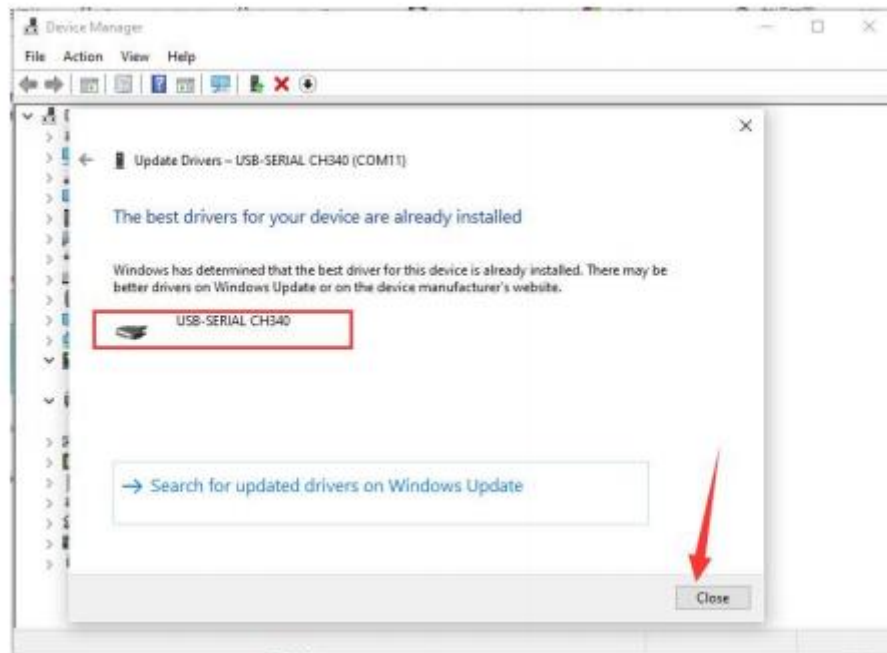


After that, select the browse option and navigate to the drive folder "USB_Drive_CH341_3_1", which can be found in the information we provide. (Note that the file path selects the location of the .For example, I store this driver file on the computer desktop, so the file path I choose is `C:\Users\mayn\Desktop\USB_Drive_CH341_3_1_For_Windows`)

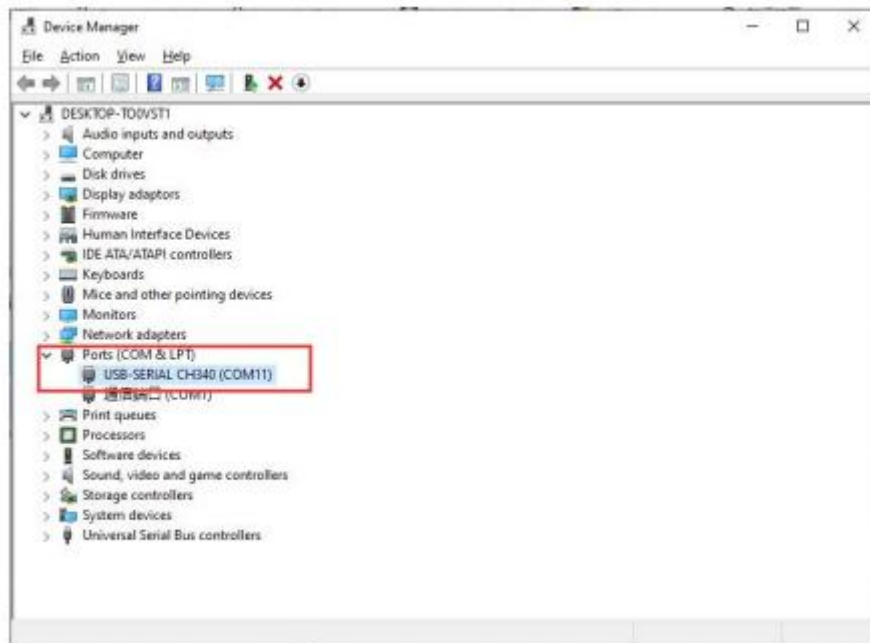


Once the software has been installed, you will get a confirmation message.

Installation completed, click "Close".



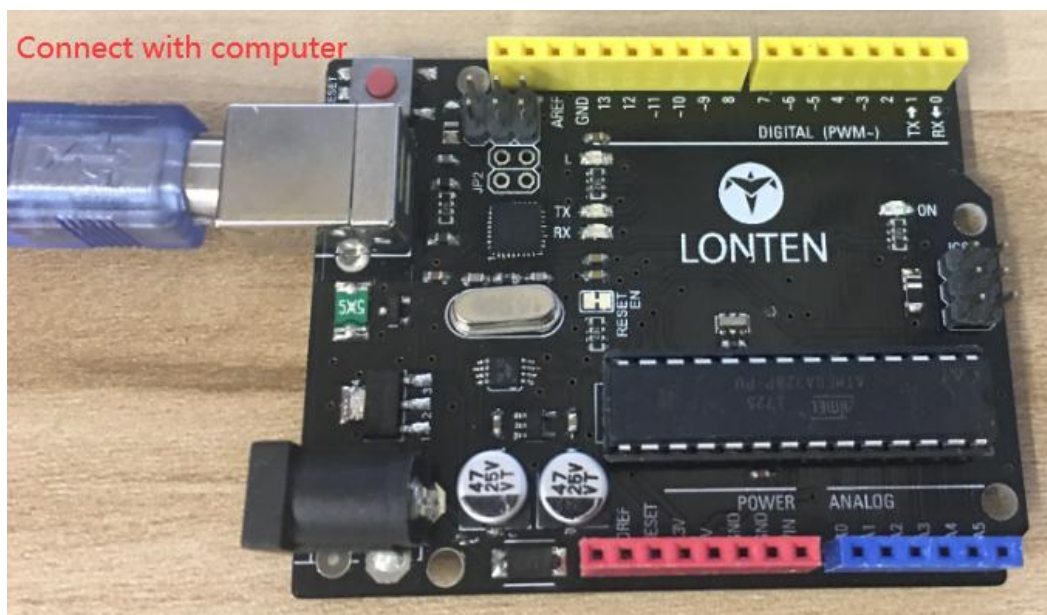
Up to now, the driver is installed well. Then you can right click “My Computer”—>for “Properties”—>click the “Device manager”, you should see the device as the figure shown below. Or you can search for "devi" in your computer, or you can open the device manager of your computer.



For MAC System

Arduino UNO

Plug one end of your USB cable into the Arduino UNO CH340 Board and the other into a USB socket on your computer.

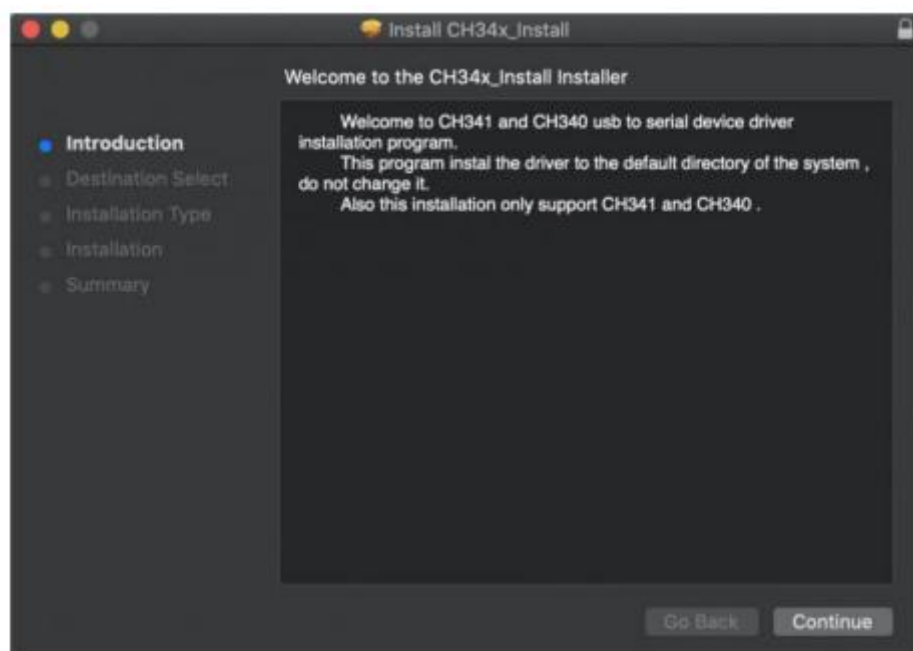


The driver file of the CH340G of the MAC system is provided in the

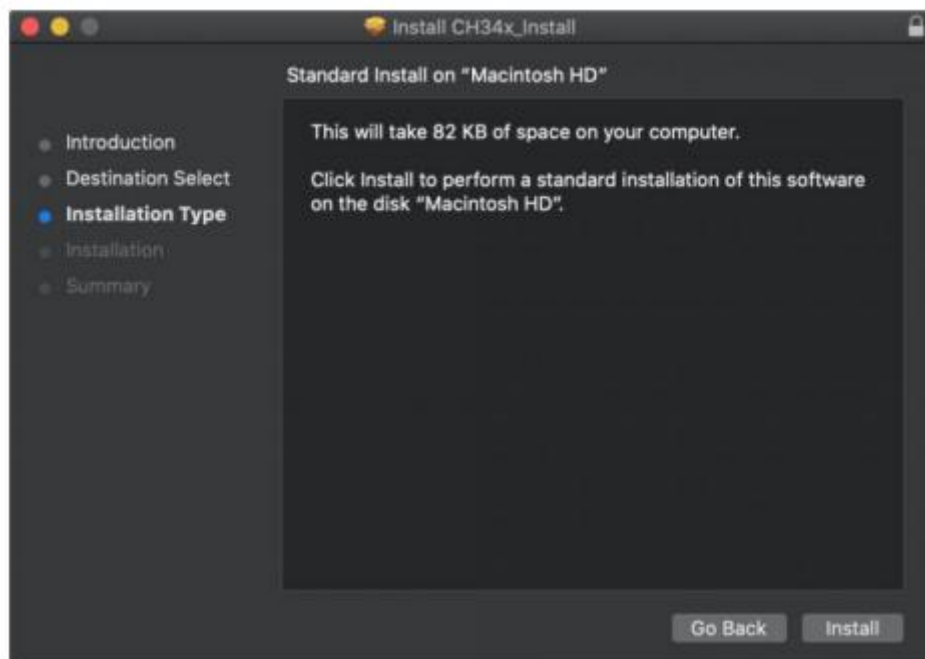
tutorial data package.



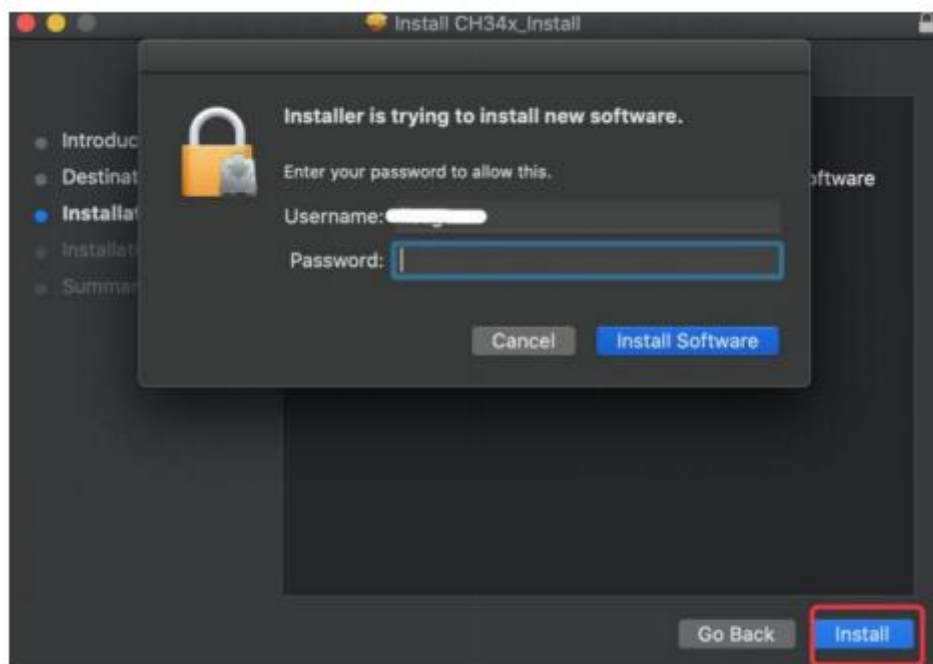
Double-click installation package and tap Continue



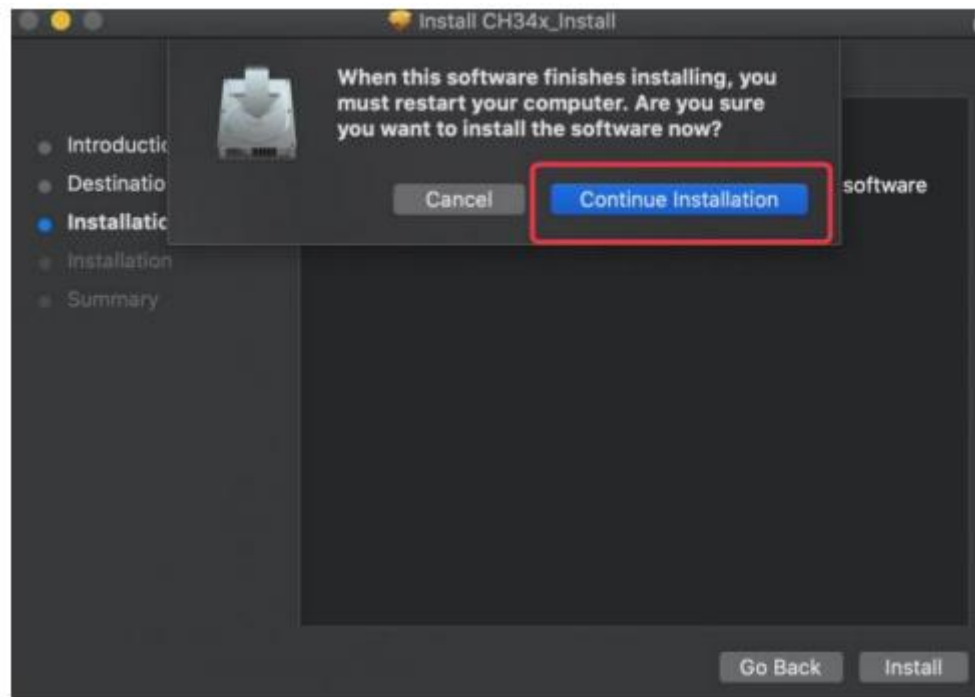
Click Install



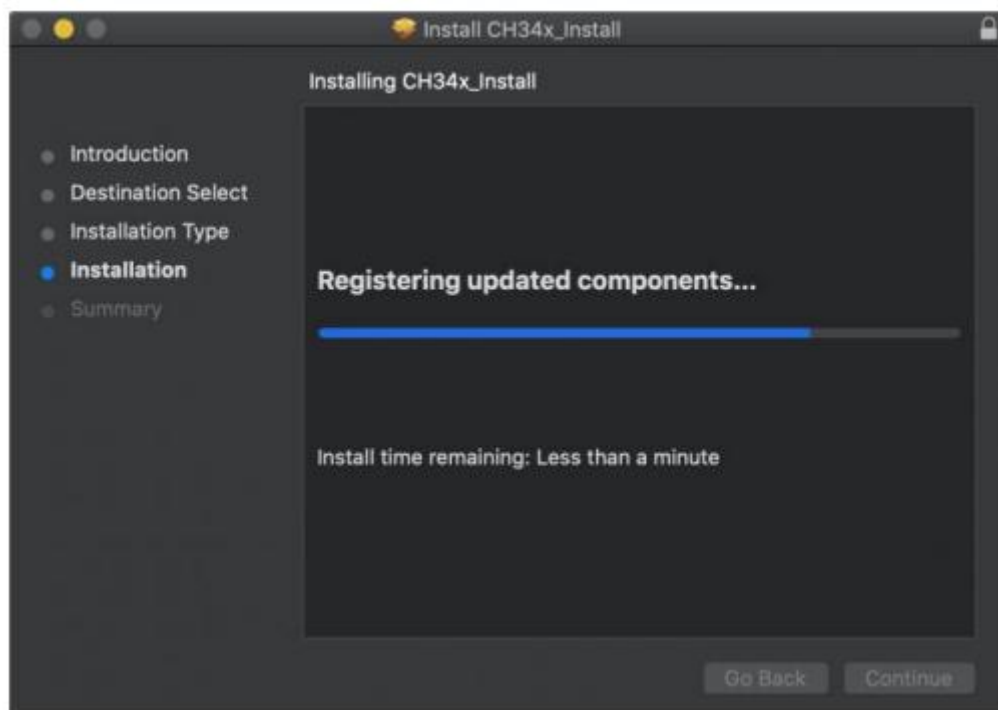
Input your user password and click Install Software



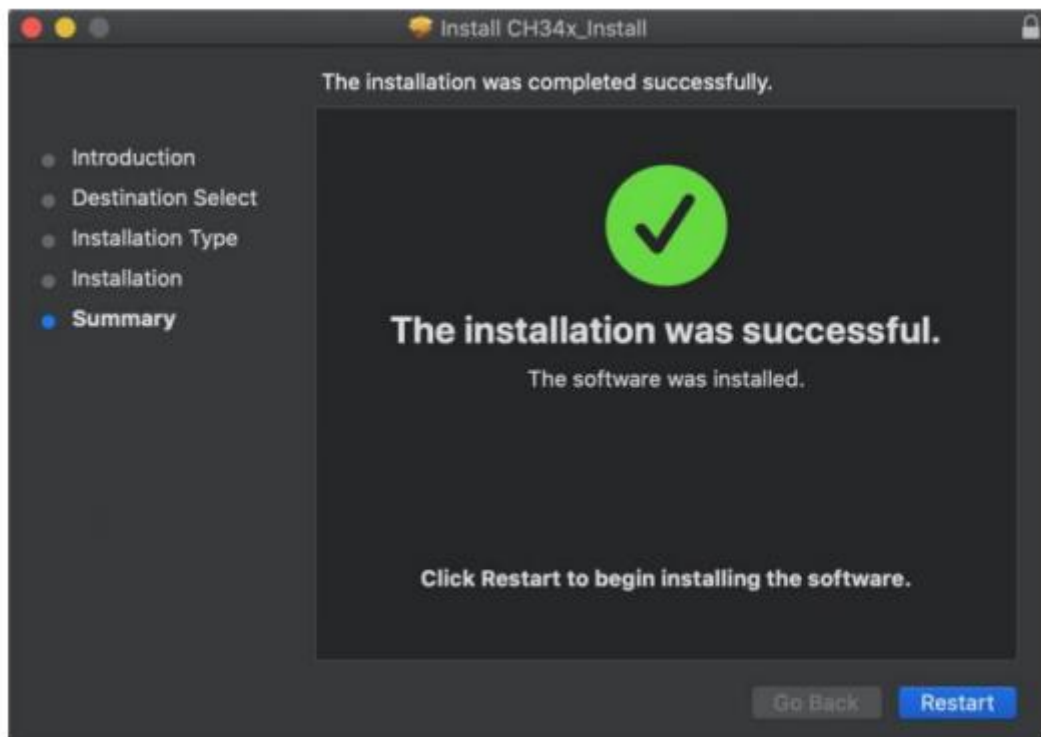
Tap Continue Installation



Wait to install



Click Restart after the installation is finished



How to Add Arduino Libraries

Installing Additional Arduino Libraries

Once you are comfortable with the Arduino software and using the built-in functions, you may want to extend the ability of your Arduino with additional libraries.

What are Libraries?

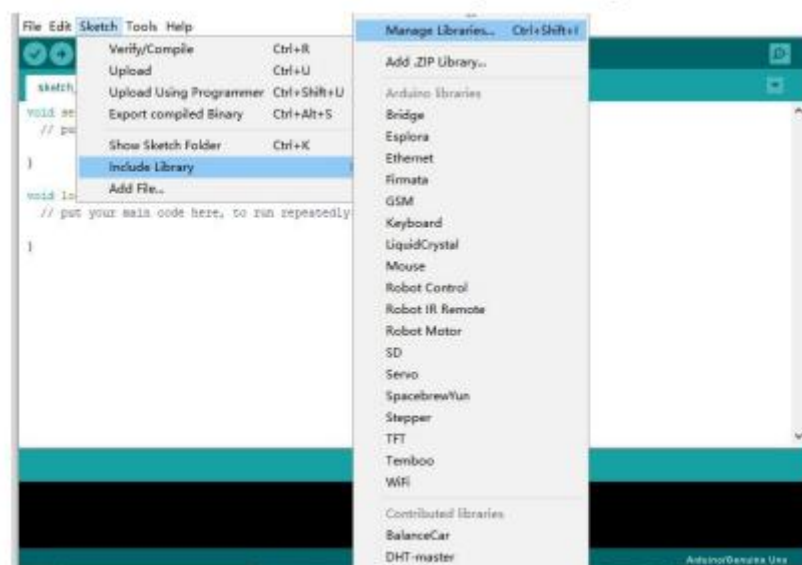
Libraries are a collection of code that makes it easy for you to connect to a sensor, display, module, etc. For example, the built-in Liquid Crystal library makes it easy to talk to character LCD displays. There are hundreds of additional libraries available on the Internet for download.

The built-in libraries and some of these additional libraries are listed in the reference. To use the additional libraries, you will need to install them.

How to Install a Library

Using the Library Manager

To install a new library into your Arduino IDE you can use the Library Manager (available from IDE version 1.8.0). Open the IDE and click to the "Sketch" menu and then Include Library > Manage Libraries.

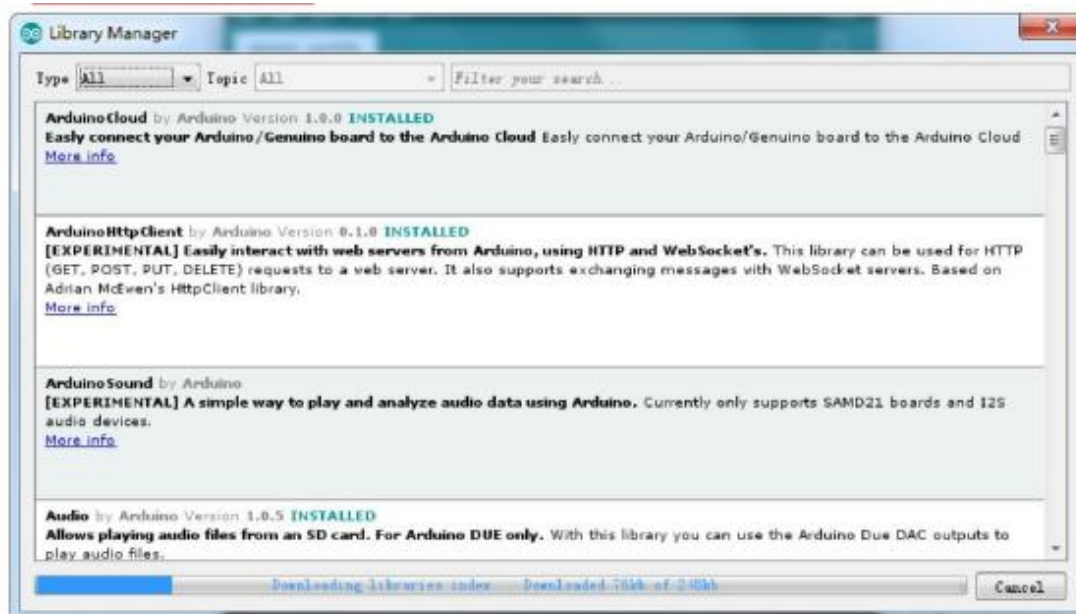
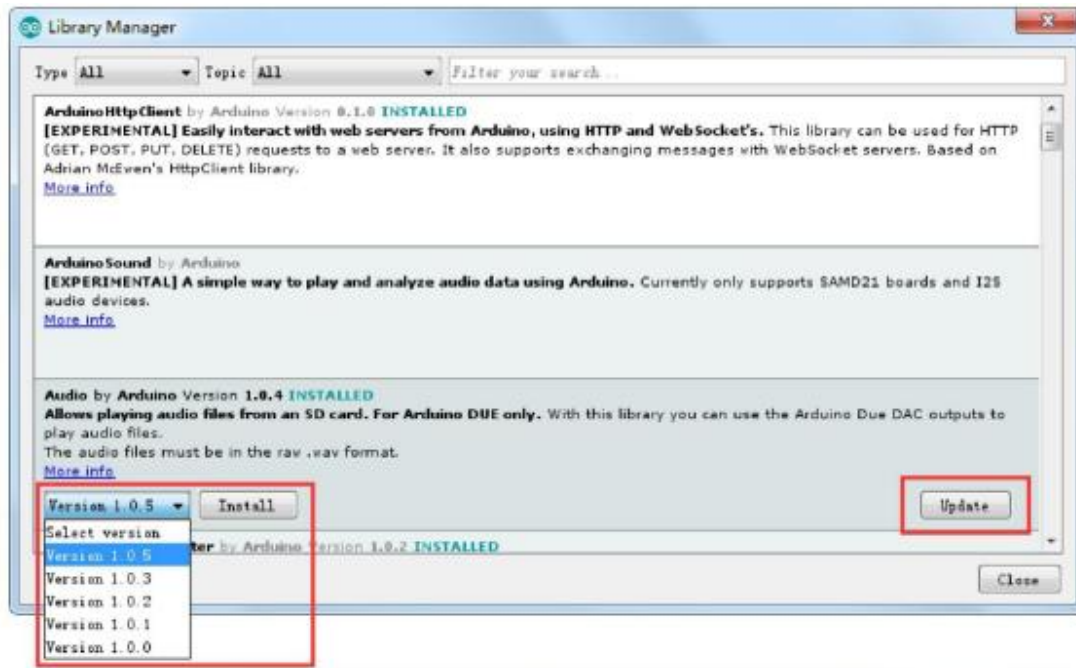


Then the library manager will open and you will find a list of libraries that are already installed or ready for installation. In this example we will install the Bridge library. Scroll the list to find it, then select the version of the library you want to install. Sometimes only one version of the library is available. If the version selection menu does not appear, don't

LROBRUYA

worry: it is normal.

There are times you have to be patient with it, just as shown in the figure. Please refresh it and wait.



Finally click on install and wait for the IDE to install the new library.

Downloading may take time depending on your connection speed. Once it has finished, an Installed tag should appear next to the Bridge library. You can close the library manager.



You can now find the new library available in the Include Library menu. If you want to add your own library open a new issue on [Github](#).

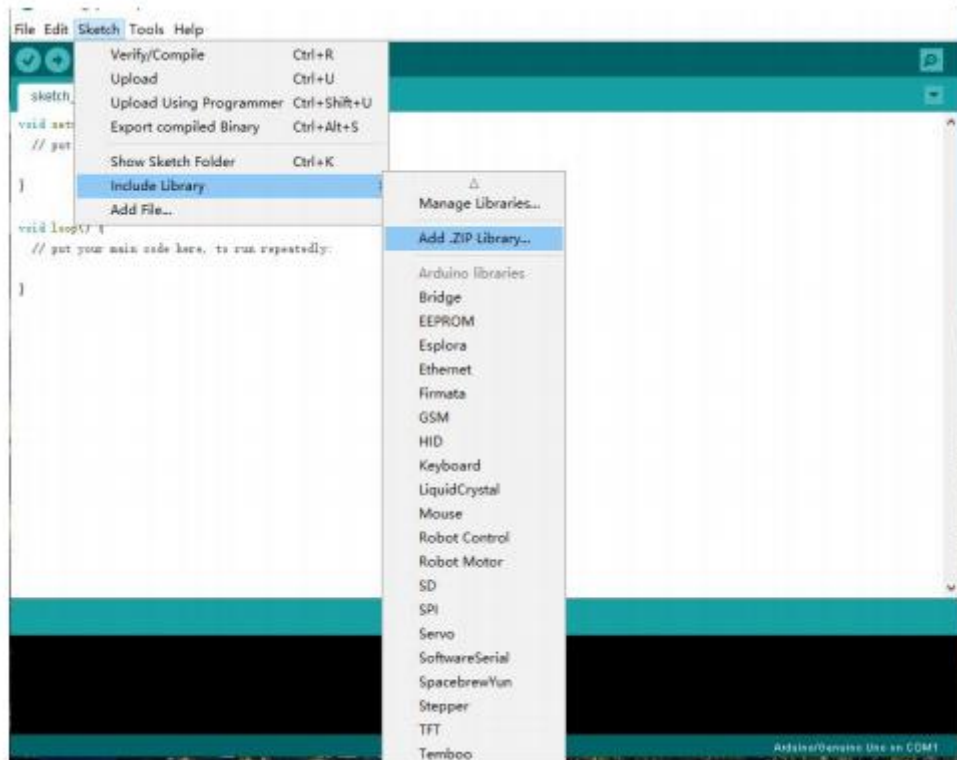
Importing a .zip Library

Libraries are often distributed as a ZIP file or folder. The name of the folder is the name of the library. Inside the folder will be a .cpp file, a .h file and often a keywords.txt file, examples folder, and other files required by the library. Starting with version 1.0.5, you can install 3rd party libraries in the IDE. Do not unzip the downloaded library, leave it as is.

In the Arduino IDE, navigate to Sketch > Include Library. At the top of

LROBRUYA

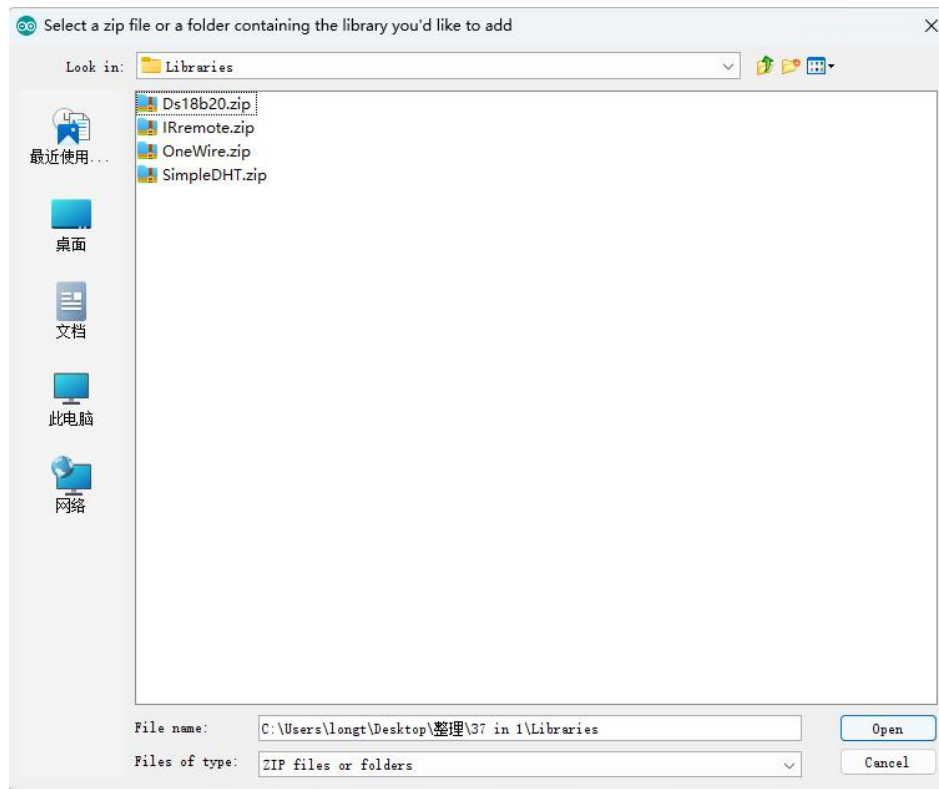
the drop down list, select the option to "Add .ZIP Library".



You will be prompted to select the library you would like to add.

Navigate to the .zip file's location and open it.

LROBRUYA





Return to the Sketch > Import Library menu. You should now see the library at the bottom of the drop-down menu. It is ready to be used in your sketch. The zip file will have been expanded in the libraries folder in your Arduino sketches directory. **NB: the Library will be available to use in sketches, but examples for the library will not be exposed in the File > Examples until after the IDE has restarted.**

Blink Test

Overview

In this Project, you will learn how to program your UNO R3 controller board to blink the Arduino's built-in LED, and how to download programs by basic steps.

Component Required:

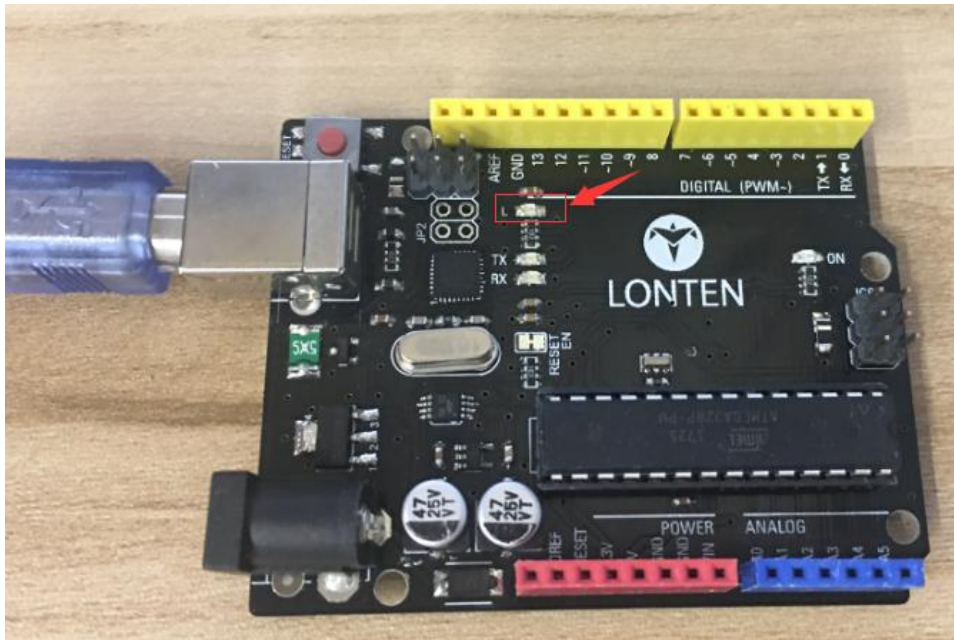
LONTEN Uno R3 Board* 1

Principle

The UNO R3 board has rows of connectors along both sides that are used to connect to several electronic devices and plug-in 'shields' that extends its capability.

It also has a single LED that you can control from your sketches. This LED is built onto the UNO R3 board and is often referred to as the 'L' LED as this is how it is labeled on the board.

LROBRUYA



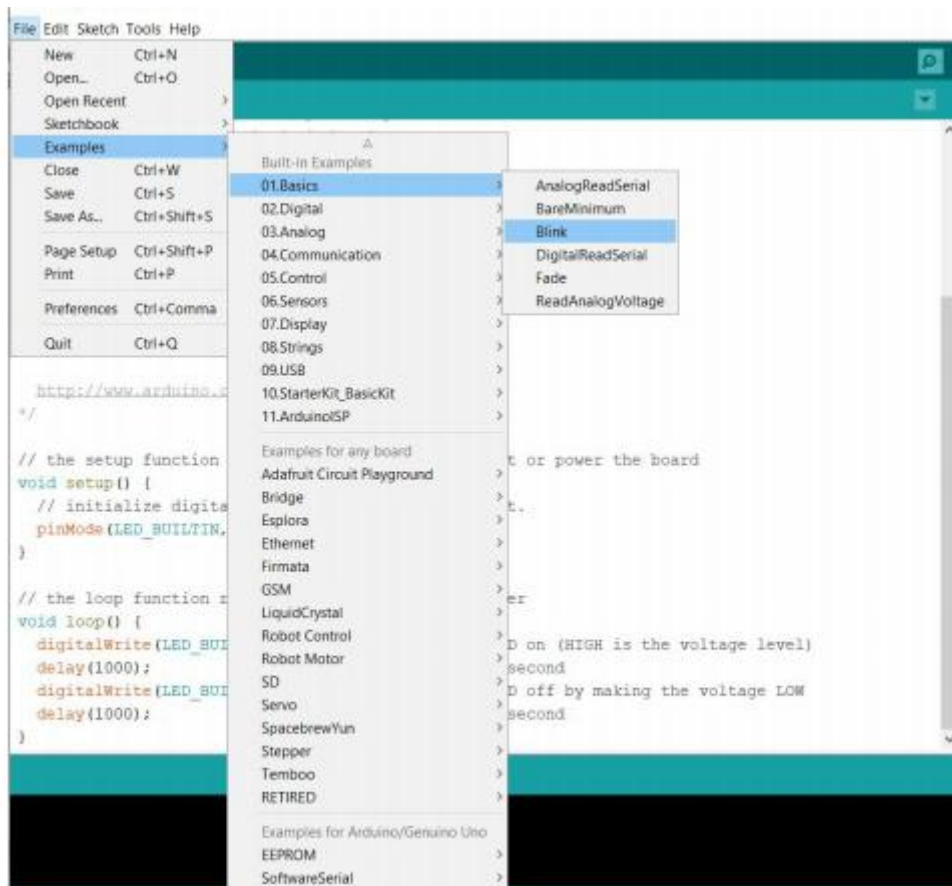
In this Project, we will reprogram the UNO board with our own Blink sketch and then change the rate at which it blinks.

In the previous chapter-How to install Arduino IDE, you set up your Arduino IDE and made sure that you could find the right serial port for it to connect to your UNO board. The time has now come to put that connection to the test and program your UNO board.

The Arduino IDE includes a large collection of example sketches that you can load up and use. This includes an example sketch for making the 'L' LED blink.

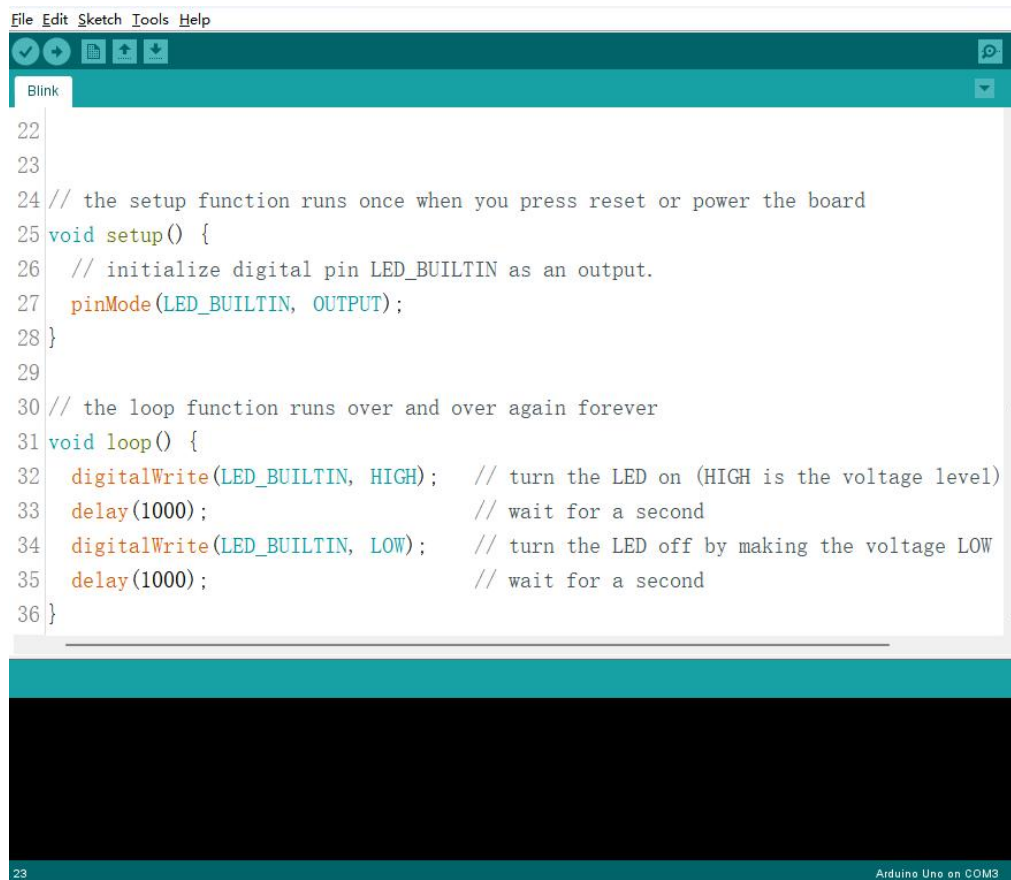
Load the 'Blink' sketch that you will find in the IDE's menu system under File > Examples > 01.Basics > Blink

LROBRUYA



When the sketch window opens, enlarge it so that you can see the entire sketch in the window.

LROBRUYA

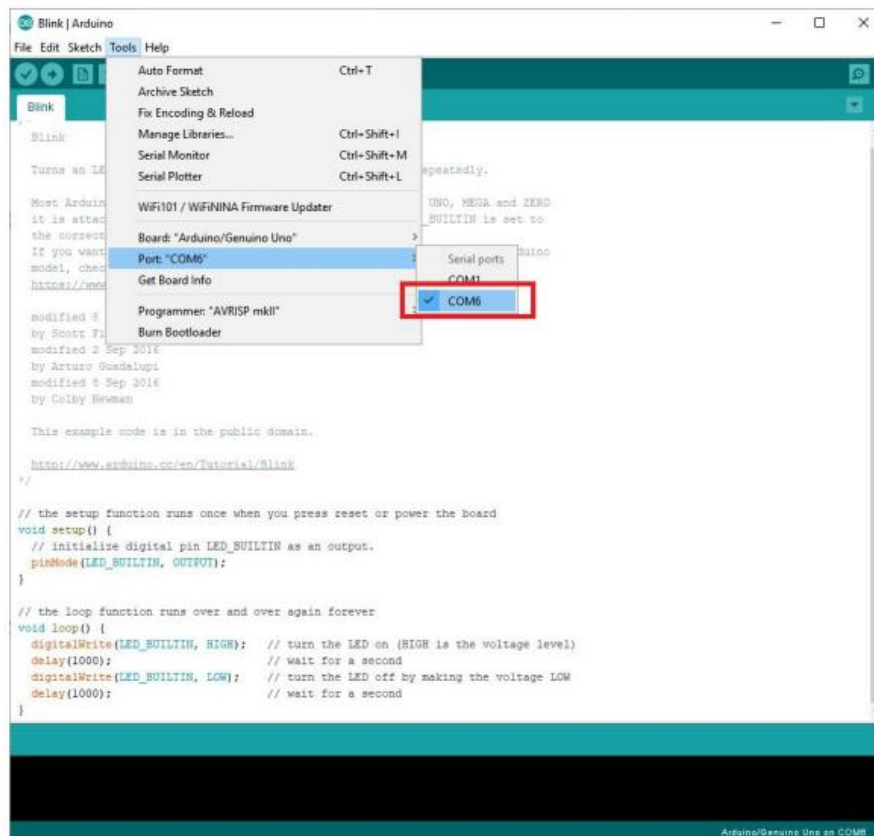
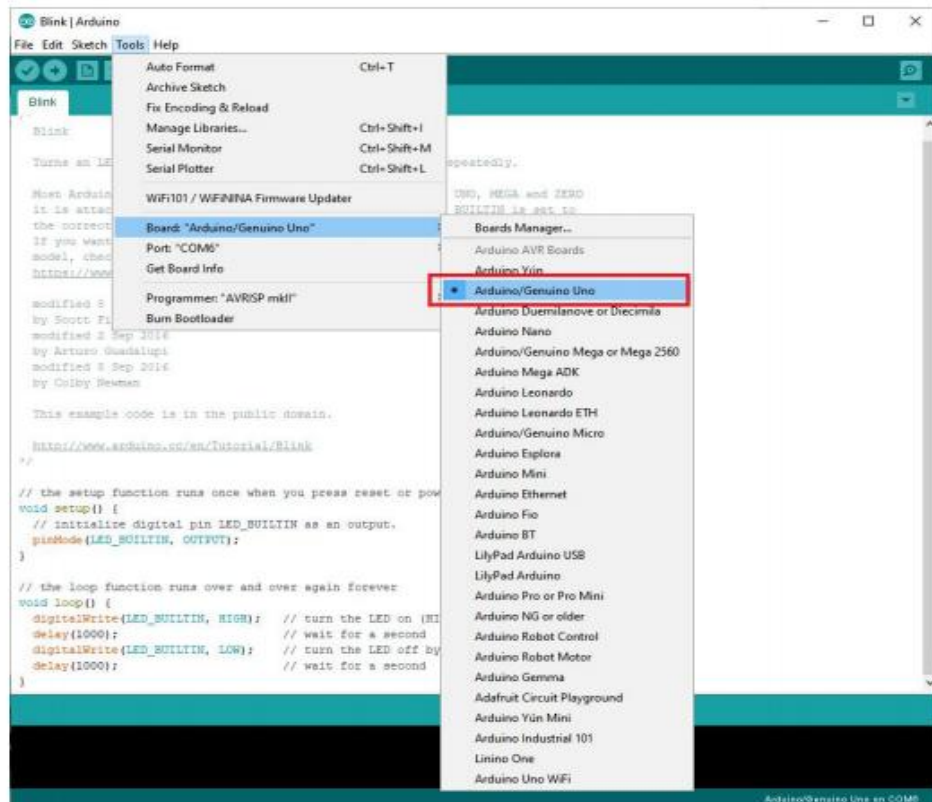
A screenshot of the Arduino IDE interface. The top menu bar includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. Below the menu is a toolbar with icons for opening files, saving, and other functions. The main text area displays the 'Blink' sketch code, which is a standard example for controlling the built-in LED. The code is as follows:

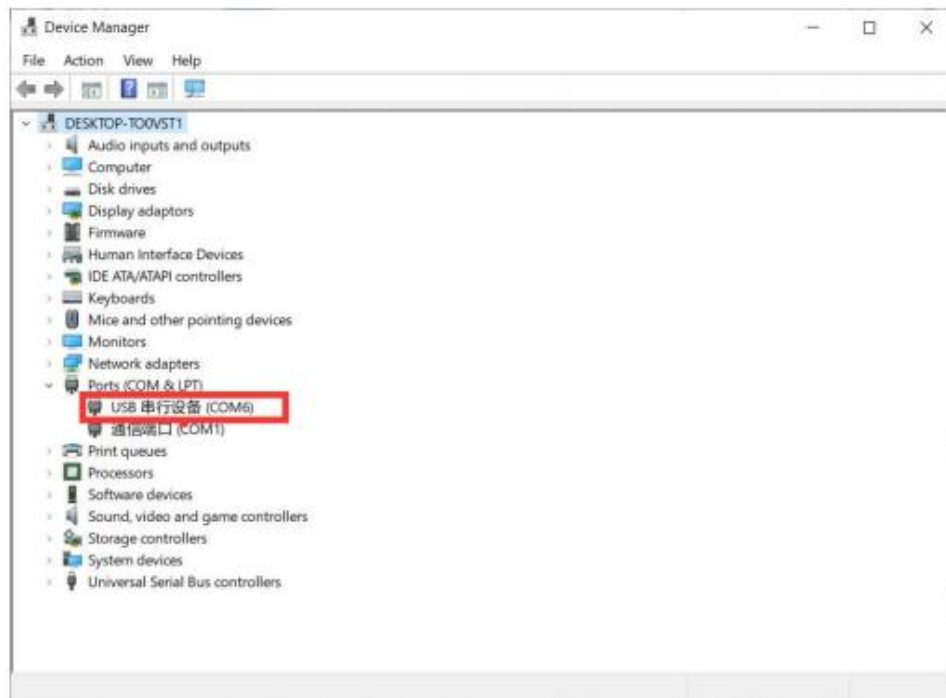
```
22
23
24 // the setup function runs once when you press reset or power the board
25 void setup() {
26   // initialize digital pin LED_BUILTIN as an output.
27   pinMode(LED_BUILTIN, OUTPUT);
28 }
29
30 // the loop function runs over and over again forever
31 void loop() {
32   digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
33   delay(1000); // wait for a second
34   digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
35   delay(1000); // wait for a second
36 }
```

The bottom status bar shows '23' on the left and 'Arduino Uno on COM3' on the right.

Attach your Arduino board to your computer with the USB cable and check that the 'Board Type' and 'Serial Port' are set correctly.

LROBRUYA



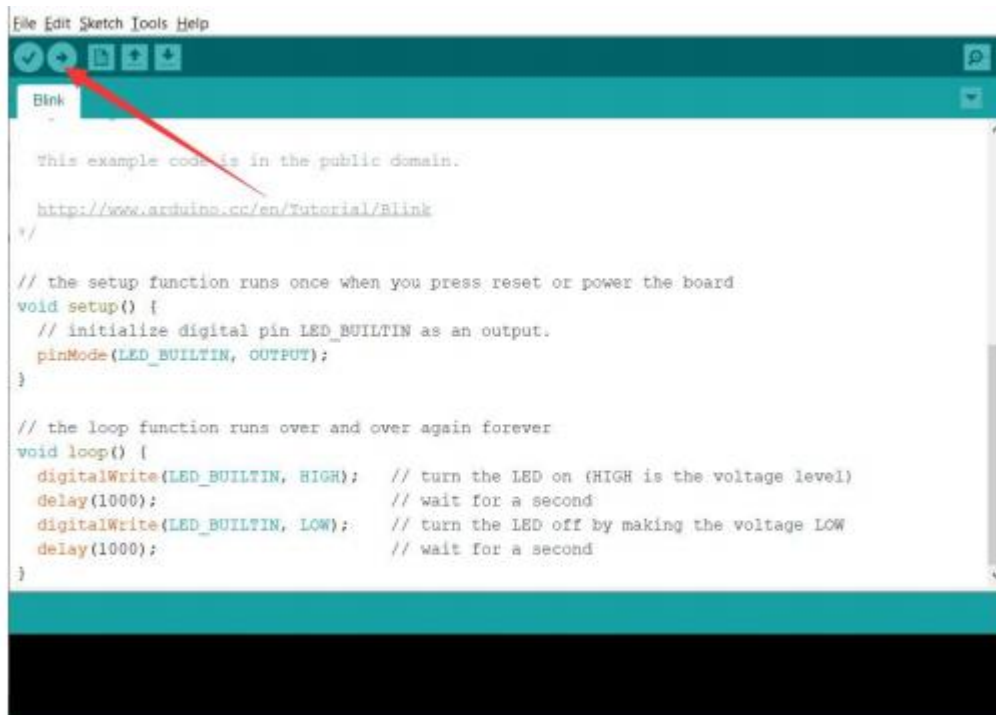


Note: The Board Type and Serial Port here are not necessarily the same as shown in picture. If you are using UNO, then you will have to choose Arduino UNO as the Board Type, other choices can be made in the same manner. And the Serial Port displayed for everyone is different, despite COM 6 chosen here, it could be COM3 or COM4 on your computer. A right COM port is supposed to be COMX (arduino XXX), which is by the certification criteria.

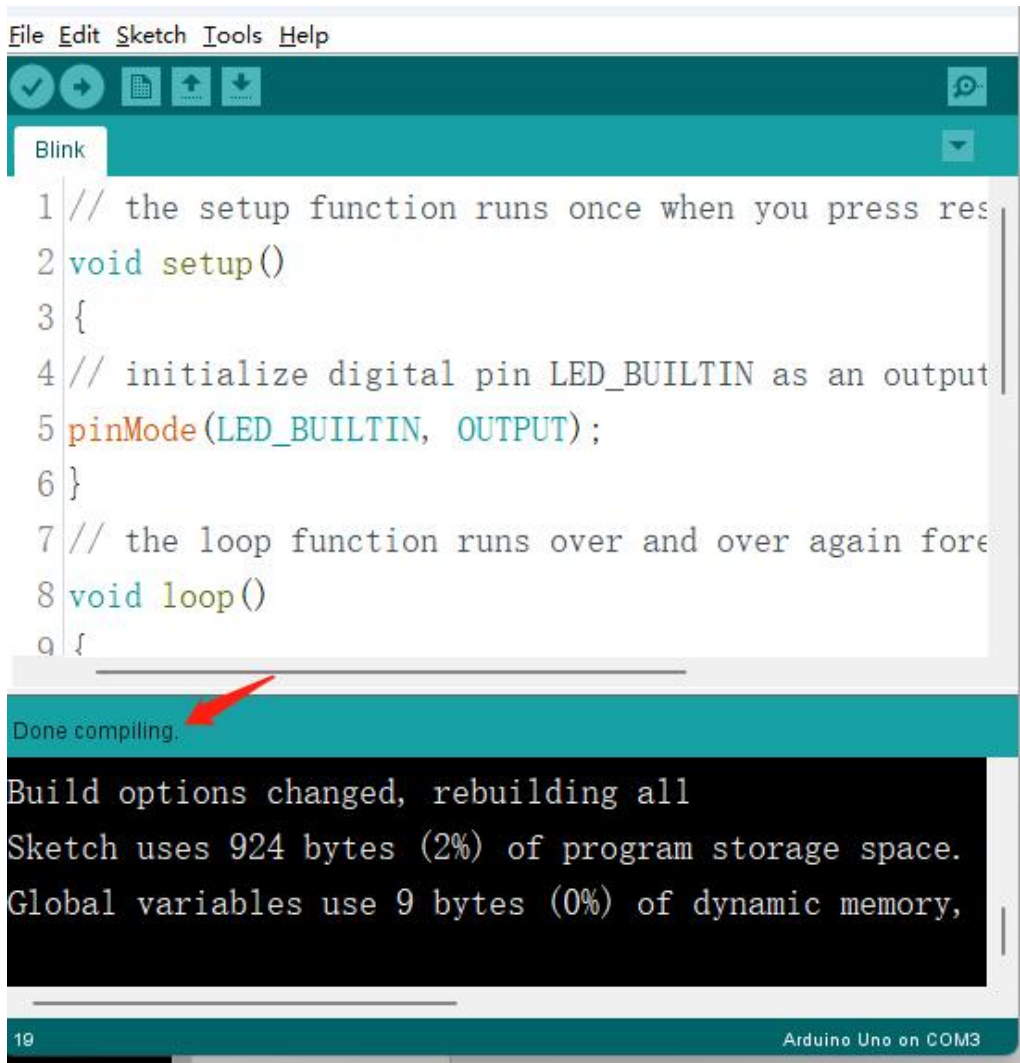
The Arduino IDE will show you the current settings for board at the bottom of the window.



Click on the 'Upload' button. The second button from the left on the toolbar.



When the status bar prompts "Done uploading", it means the code upload is successful.



The screenshot shows the Arduino IDE interface. The menu bar includes File, Edit, Sketch, Tools, and Help. The toolbar contains icons for checking, running, saving, and uploading. The sketch is named "Blink". The code in the editor is as follows:

```
1 // the setup function runs once when you press res
2 void setup()
3 {
4 // initialize digital pin LED_BUILTIN as an output
5 pinMode(LED_BUILTIN, OUTPUT);
6 }
7 // the loop function runs over and over again fore
8 void loop()
9 {
```

Below the code editor, a teal status bar indicates "Done compiling." with a red arrow pointing to it. Below that, a black console window displays the following build options:

```
Build options changed, rebuilding all
Sketch uses 924 bytes (2%) of program storage space.
Global variables use 9 bytes (0%) of dynamic memory,
```

The bottom status bar shows "19" and "Arduino Uno on COM3".

If an error message appears.



The screenshot shows the Arduino IDE interface with an error message displayed in the console. The error message is as follows:

```
Problem uploading to board. See http://www.arduino.cc/en/Guide/Troubleshooting#upload for suggestions.
An error occurred while uploading the sketch
avrdude: ser_open(): can't open device '\\.\COM15': The system cannot find the file specified.
Problem uploading to board. See http://www.arduino.cc/en/Guide/Troubleshooting#upload for sugges
```

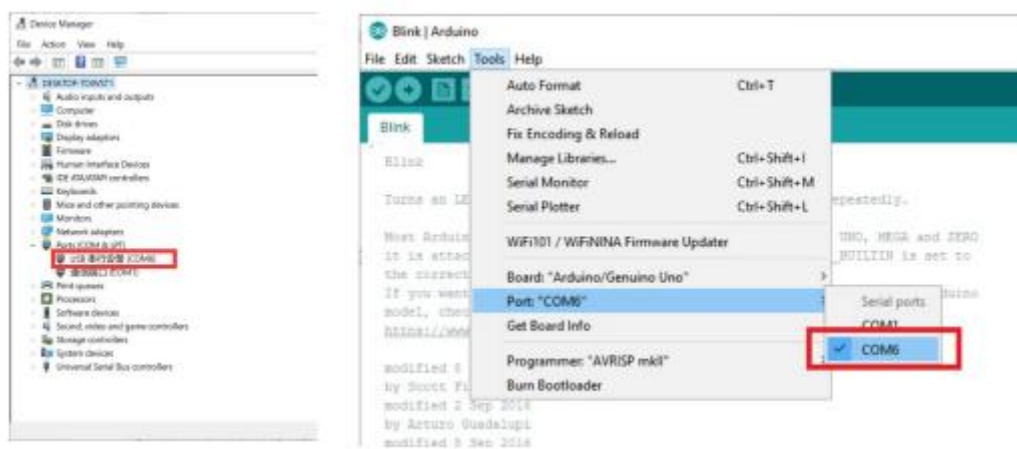
The bottom status bar shows "Arduino/Genuino Uno on COM15".

There can be several reasons:

1. The arduino uno driver software is not installed successfully, please

refer to the course for the installation steps: [How to Install Arduino Driver](#).

2. The communication serial port selection of arduino uno is wrong; you can check the communication port COMx of your arduino uno in the computer in the device manager.



3. If your Arduino uno is connected to a Bluetooth module, it will occupy the communication serial port. You need to remove the Bluetooth module connection before uploading the code.

4. The USB data cable is not firmly connected. Check if there are any of the above problems. After correcting, follow the previous steps to re-operate.

Sample Program

// the setup function runs once when you press reset or power the board

```
void setup()
```

```
{
```




```
// initialize digital pin LED_BUILTIN as an output.

pinMode(LED_BUILTIN, OUTPUT);

}

// the loop function runs over and over again forever

void loop()

{

  digitalWrite(LED_BUILTIN, HIGH);

  // turn the LED on (HIGH is the voltage level)

  delay(1000);

  // wait for a second

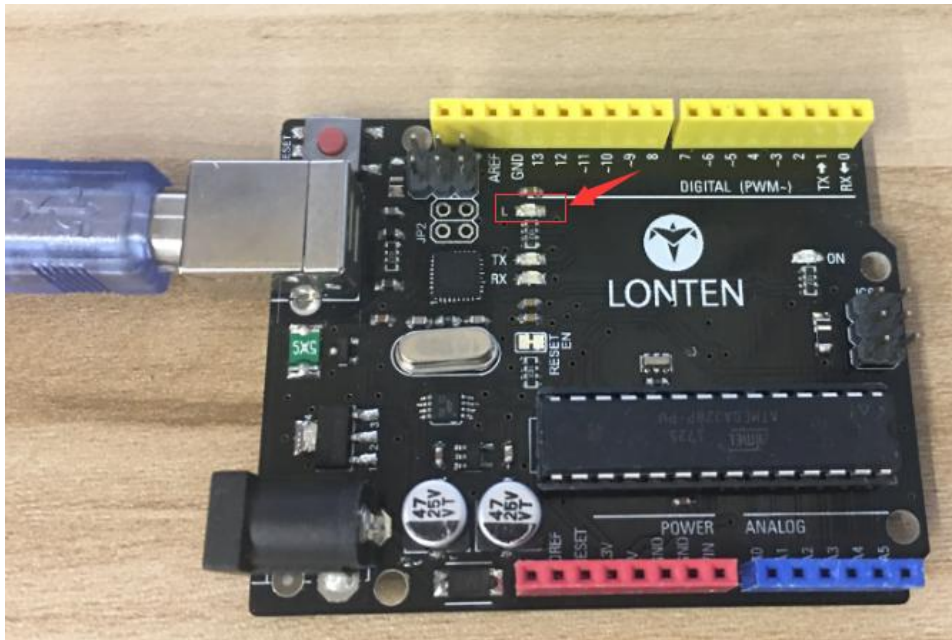
  digitalWrite(LED_BUILTIN, LOW);

  // turn the LED off by making the voltage LOW

  delay(1000);

  // wait for a second

}
```



After the code is successfully uploaded, the "L" character LED will flash once per second. So far, you have completed the testing process of your first program.

Lesson 1 0.96 OLED Display Module

Introduction

OLED is short for organic light emitting diode. On the microscopic level, an OLED display is a matrix of organic LEDs that light up when they emit energy. Old LCD (Liquid Crystal Display) technology uses electronically controlled polarizers to change the way light passes or does not pass through them. This requires an external backlight that lights up the whole display underneath. This uses a lot of energy because at the time the display is on, enough light for all pixels must be provided.

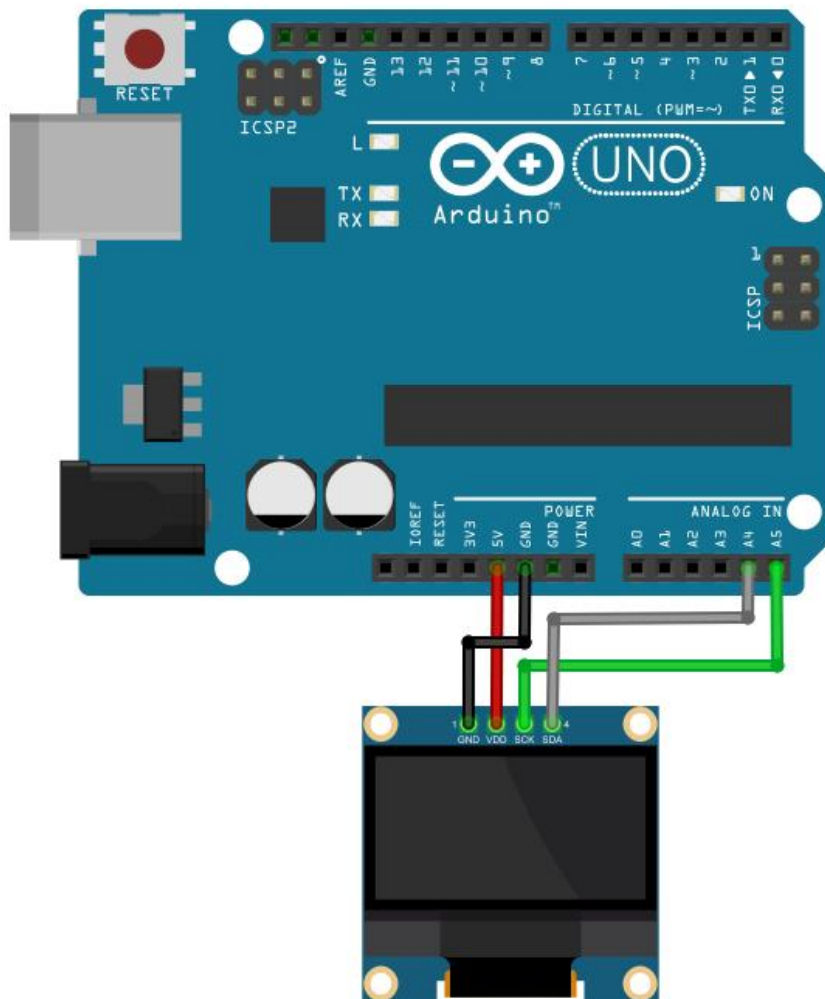
The new OLED technology only uses electricity per pixel. Because each pixel creates its own light, only the pixels that are on use electricity. This makes OLED technology very efficient; also, the way these types of OLEDs are built allows them to be very thin compared to LCD.



Specification

- Supply Voltage: 3.3V to 5V
- Number of Pixels: 128×64
- Color Depth: blue
- Communication way: IIC

Connection Diagram



Sample Code

```
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

// Initialize OLED display object and set resolution to 128x64
Adafruit_SSD1306 display = Adafruit_SSD1306(128, 64, &Wire);

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

  // Initialize OLED display screen
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.clearDisplay();
}
```

```
// Set Text Style
display.setTextSize(1);
display.setTextColor(WHITE);

// Display prompt information Drawing Shapes
display.setCursor(30, 8);
display.println("Drawing Shapes");

// Drawing Text Square:
display.setTextSize(1);
display.setCursor(10, 25);
display.println("Square:");

// Draw a square inside the screen
display.drawRect(15, 40, 20, 20, WHITE); // (x, y, width, height, color)

// Display Text Circle:
display.setTextSize(1);
display.setCursor(60, 25);
display.println("Circle:");

// Draw circles inside the screen
display.drawCircle(80, 50, 12, WHITE); // (centerX, centerY, radius, color)

// Update display content
display.display();
}

void loop() {

}
```

Result

Wiring as the above diagram and burning the code, after power-on, OLED display squares and circles.

Lesson 2 1602 I2C Module

Introduction

1602 I2C module is a 16 character by 2 line LCD display with Blue background and White backlight.

The original 1602 LCD needs 7 IO ports to be up and running, but ours is built with Arduino IIC/I2C interface, saving you 5 IO ports.

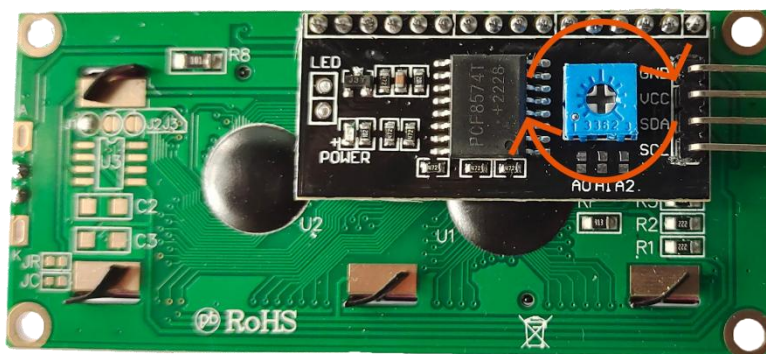
This LCD is ready-to-use because it is compatible with the Arduino Liquid Crystal Library.

LCDs are great for printing data and showing values. Adding an LCD to your project will make it super portable and allow you to integrate up to 32 characters (16x2) of information.

On the back of LCD display there is a blue potentiometer. You can turn the potentiometer to adjust the contrast.

Notice that the screen will get brighter or darker and that the characters become more visible or less visible.

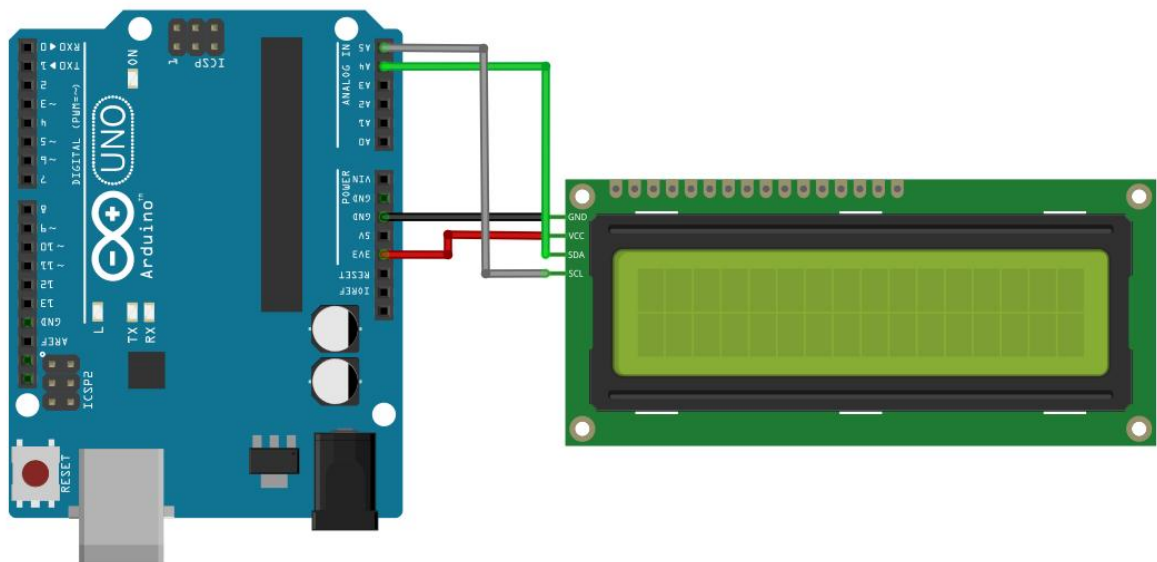
Adjust Contrast



Specification

- I2C Address: 0x27
- Backlight (Blue with white char color)
- Supply voltage: 5V
- Adjustable contrast
- GND: A pin that connects to ground
- VCC: A pin that connects to a +5V power supply
- SDA: A pin that connects to analog port A4 for IIC communication
- SCL: A pin that connects to analog port A5 for IIC communication

Connection Diagram



Sample Code

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2); // set the LCD address to 0x27 for a16 chars and
2 line display
```




```
void setup ()
{
  lcd.init (); // initialize the lcd
  lcd.init (); // Print a message to the LCD.
  lcd.backlight ();
  lcd.setCursor (3,0);
  lcd.print ("Hello, world!"); // LED print hello, world!
  lcd.setCursor (3,1);
  lcd.print ("LONTEN"); // LED print LONTEN
}
void loop ()
{
}
```

Result

Hookup well and upload the code to board, you should see the words

“Hello, World!” and “LONTEN” pop up on your LCD. Remember you can adjust the contrast using the potentiometer if you can’t make out the words clearly.

Lesson 3 Temp and Humidity module

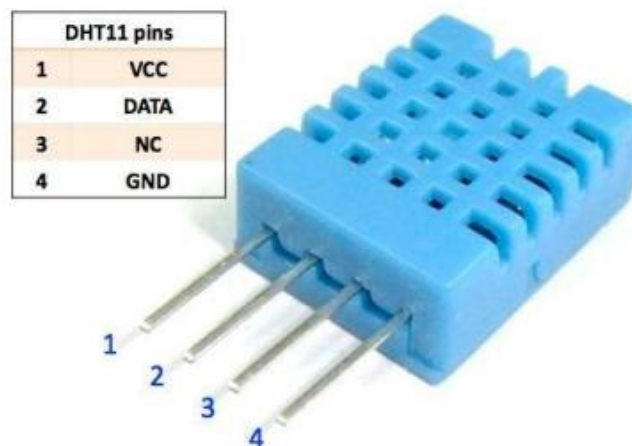
Introduction

After the first two classes, we have learned how to use these two display screens. Next, we will combine them with sensors to display relevant data.

This DHT11 Temperature and Humidity Sensor features calibrated digital signal output with the temperature and humidity sensor complex. Its technology ensures high reliability and excellent long-term stability. A

high-performance 8-bit micro controller is connected. This sensor includes a resistive element and a sense of wet NTC temperature measuring devices.

Applications: HVAC, dehumidifier, testing and inspection equipment, consumer goods, automotive, automatic control, data loggers, weather stations, home appliances, humidity regulator, medical and other humidity measurement and control.



Specification

Humidity:

Resolution: 16Bit

Repeatability: $\pm 1\%$ RH

Accuracy: At 25°C $\pm 5\%$ RH

Interchangeability: fully interchangeable

Response time: $1/e(63\%)$ of 25°C 6s



1m / s air 6s

Hysteresis: $<\pm 0.3\%RH$

Long-term stability: $<\pm 0.5\% RH / yr$ in

Temperature:

Resolution: 16Bit

Repeatability: $\pm 0.2^{\circ}C$

Range: At $25^{\circ}C \pm 2^{\circ}C$

Response time: 1 / e (63%) 10S

Electrical Characteristics

Power supply: DC 3.5~5.5V

Supply Current: measurement 0.3mA standby 60 μ A

Sampling period: more than 2 seconds

Pin Description:

1 the VDD power supply 3.5~5.5V DC

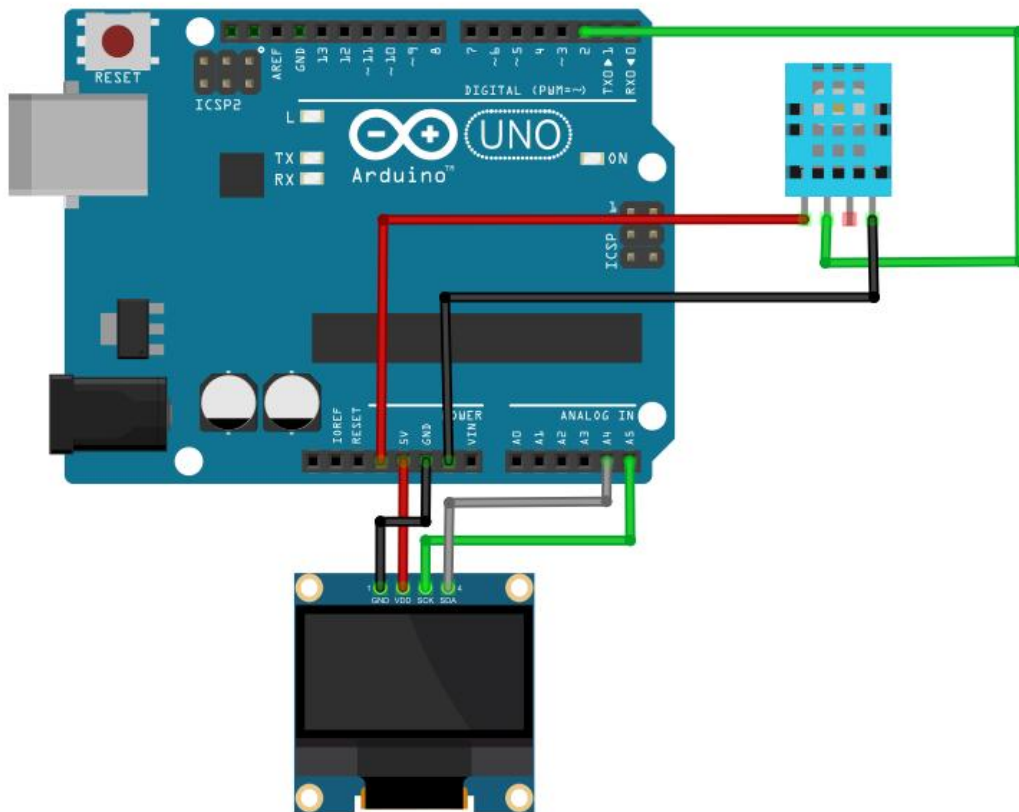
2 DATA serial data, a single bus

3 NC, empty pin

4 GND ground, the negative power

Connection Diagram

LROBRUYA



Sample Code

```
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <SimpleDHT.h>
// Initialize OLED display object and set resolution to 128x64
Adafruit_SSD1306 display = Adafruit_SSD1306(128, 64, &Wire);

// for DHT11,
//      VCC: 5V or 3V
//      GND: GND
//      DATA: 2
int pinDHT11 = 2;
SimpleDHT11 dht11;

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



```
// Initialize OLED display screen
display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
display.clearDisplay();
}

void loop() {
    // start working...
    Serial.println("=====");
    Serial.println("Sample DHT11...");

    // read with raw sample data.
    byte temperature = 0;
    byte humidity = 0;
    byte data[40] = {0};
    if (dht11.read(pinDHT11, &temperature, &humidity, data)) {
        Serial.print("Read DHT11 failed");
        return;
    }

    Serial.print("Sample RAW Bits: ");
    for (int i = 0; i < 40; i++) {
        Serial.print((int)data[i]);
        if (i > 0 && ((i + 1) % 4) == 0) {
            Serial.print(' ');
        }
    }
    Serial.println("");

    Serial.print("Sample OK: ");

    Serial.print((int)temperature); Serial.print(" °C, ");

    Serial.print((int)humidity); Serial.println(" %");
    // Set Text Style
    display.setTextSize(1);
    display.setTextColor(WHITE);

    display.setCursor(45, 8);
    display.println("T & H");

    display.setCursor(0, 18);
    display.println("TEMP:");
```

```
display.setCursor(45, 18);  
display.println(temperature);  
display.setCursor(60, 18);  
display.println("C");  
  
display.println("H:");  
display.setCursor(45, 28);  
display.println(humidity);  
display.setCursor(60, 28);  
display.println("%RH");  
display.display();  
  
// DHT11 sampling rate is 1HZ.  
delay(1000);  
display.clearDisplay();  
}
```

Result

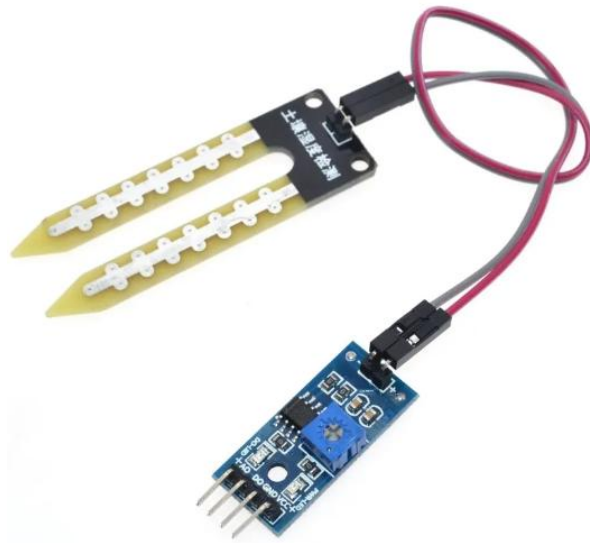




Lesson 4 Soil Humidity Sensor

Introduction

This is a simple soil humidity sensor aims to detect the soil humidity. If the soil is in lack of water, the analog value output by the sensor will decrease; otherwise, it will increase. If you use this sensor to make an automatic watering device, it can detect whether your botany is thirsty to prevent it from withering when you go out. Using the sensor with Arduino controller makes your plant more comfortable and your garden smarter. The soil humidity sensor module is not as complicated as you might think, and if you need to detect the soil in your project, it will be your best choice. The sensor is set with two probes inserted into the soil, then with the current go through the soil, the sensor will get resistance value by reading the current changes between the two probes and convert such resistance value into moisture content. The higher moisture (less resistance), the higher conductivity the soil has. Insert it into the soil and then use the AD converter to read it. With the help of this sensor, the plant can remind of you: I need water.



Specification

Power Supply Voltage: 3.3V or 5V

Working Current: $\leq 20\text{mA}$

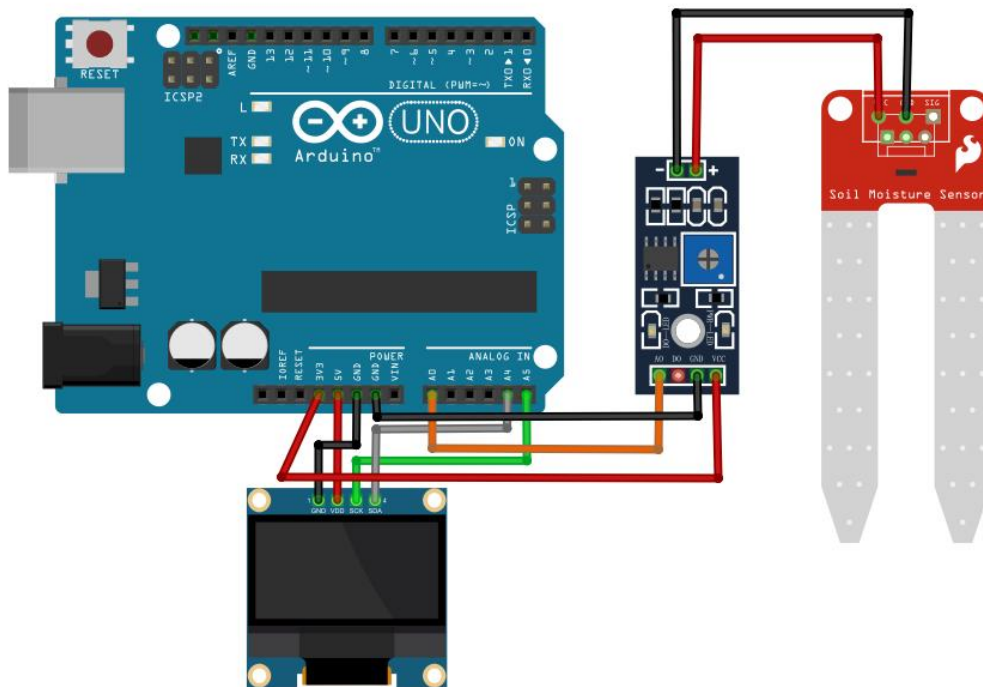
Sensor type: Analog output

Interface definition: OUT- signal, GND- GND, VCC- VCC

Packaging : Electrostatic bag sealing

Connection Diagram

LROBRUYA



Sample Code

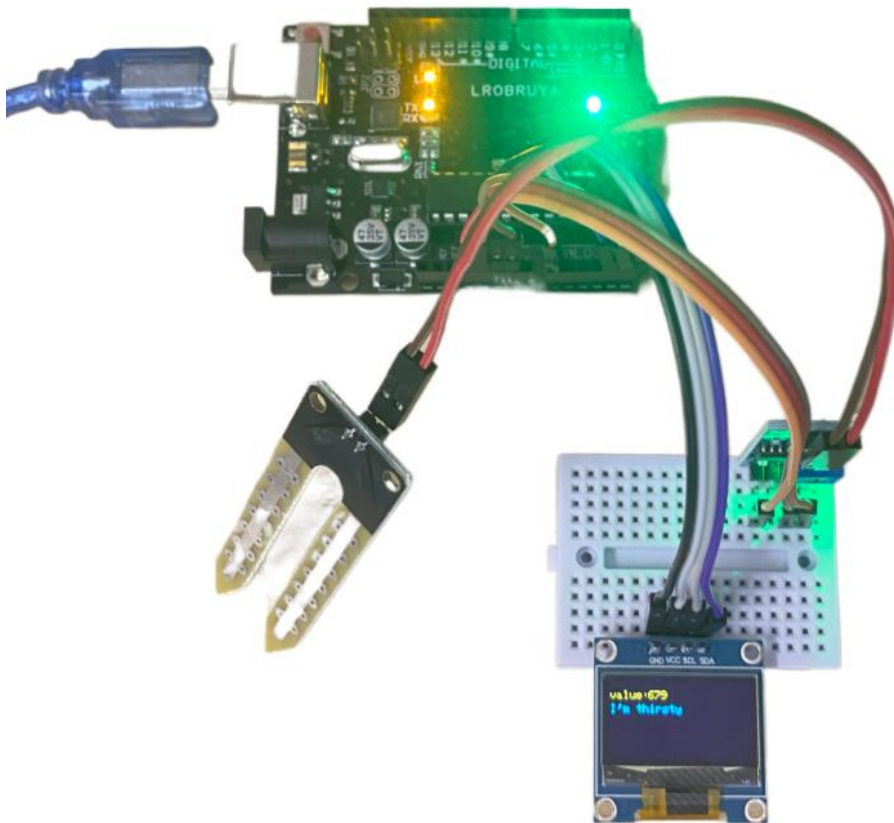
```
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <SimpleDHT.h>
// Initialize OLED display object and set resolution to 128x64
Adafruit_SSD1306 display = Adafruit_SSD1306(128, 64, &Wire);

int soil_value=0;
void setup() {
  Serial.begin(9600);
  // Initialize OLED display screen
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.clearDisplay();
}

void loop() {
  soil_value = analogRead(0); // get adc value
  // Set Text Style
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 8);
```

```
display.println("value:");  
display.setCursor(35, 8);  
display.println(soil_value);  
if(soil_value>600)  
{  
  Serial.println("I'm thirsty");  
  display.setCursor(0, 18);  
  display.println("I'm thirsty");  
}  
Serial.println(soil_value);  
display.display();  
delay(1000);  
display.clearDisplay();  
}
```

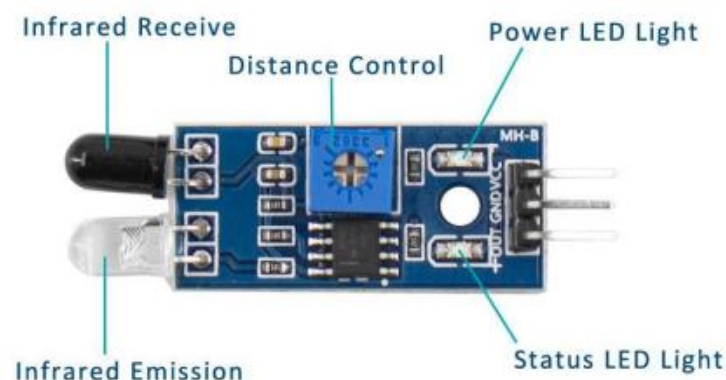
Result



Lesson 5 Infrared Obstacle Avoidance Module

Introduction

The infrared obstacle detector sensor has a pair of infrared transmitting and receiving tubes. The transmitter emits an infrared rays of a certain frequency. When the detection direction encounters an obstacle(reflecting surface), the infrared rays are reflected back, and receiving tube will receive it. At this time, the indicator (green LED) lights up. After processed by the circuit, the signal output terminal will output Digital signal. You can rotate the potentiometer on the shield to adjust the detection distance. It is better to adjust the potentiometer to make the green LED in a state between on and off. The detection distance is the best, almost 10cm.



Read the signal level of obstacle detector sensor to judge whether detect obstacles or not.



When detects an obstacle, sensor's signal pin outputs LOW (display 0); otherwise, output HIGH (display 1).

Show the result on the serial monitor, and control the external LED module turn ON/OFF.

Specification

Working voltage: DC 3.3V-5V

Working current: $\geq 20\text{mA}$

Operating temperature: $-10\text{ }^{\circ}\text{C} - +50\text{ }^{\circ}\text{C}$

detection distance :2-40cm

IO Interface: 4-wire interfaces (- / + / S / EN)

Output signal: TTL level (low level there is an obstacle, no obstacle high)

Adjustment: adjust multi-turn resistance

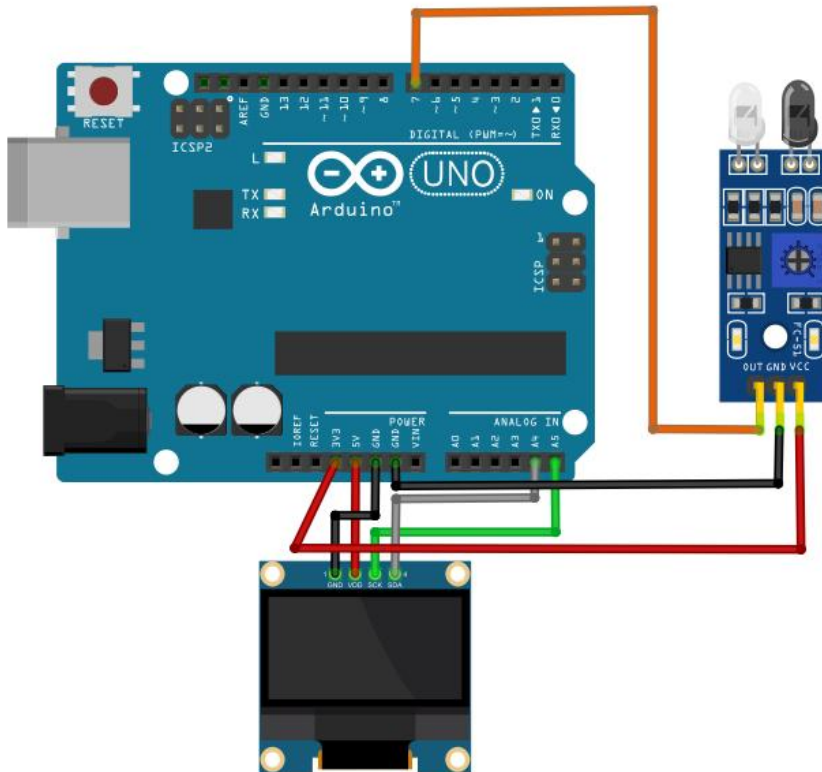
Effective angle: 35°

Size: 28mm \times 23mm

Weight Size:9g

Connection Diagram

LROBRUYA



Sample Code

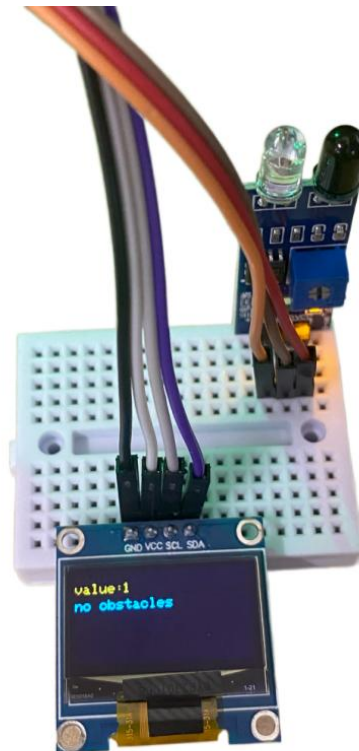
```
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <SimpleDHT.h>
// Initialize OLED display object and set resolution to 128x64
Adafruit_SSD1306 display = Adafruit_SSD1306(128, 64, &Wire);

int value=0;
void setup() {
  // Initialize OLED display screen
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.clearDisplay();
  pinMode(7,INPUT);
}

void loop() {
  display.setTextSize(1);
  display.setTextColor(WHITE);
  value = digitalRead(7); // get adc value
  // Set Text Style
```

```
display.setCursor(0, 8);  
display.println("value:");  
display.setCursor(35, 8);  
display.println(value);  
if(value==1){  
    display.setCursor(0, 18);  
    display.println("no obstacles");  
} else {  
    display.setCursor(0, 18);  
    display.println("obstacles");  
}  
display.display();  
delay(1000);  
display.clearDisplay();  
}
```

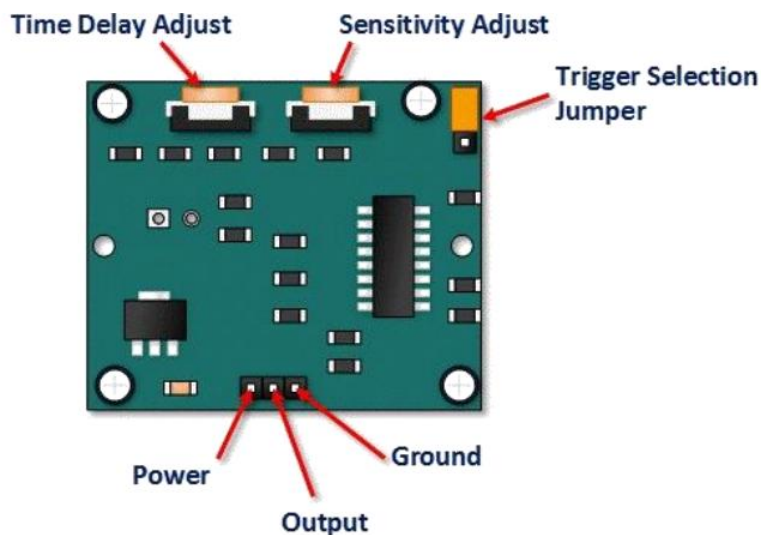
Result



Lesson 6 PIR Motion Sensor

Introduction

The human body infrared motion sensor can detect infrared signals from moving people or animals, and output switching signals. It can be applied to various occasions to detect human movement. The conventional pyroelectric infrared sensor has a larger volume, a complicated circuit and a lower reliability. Now, we have introduced a human body infrared motion sensor specially designed for Arduino. The sensor integrates an integrated digital pyroelectric infrared sensor and connection pins. It has higher reliability, lower power consumption and simpler peripheral circuits.



Specification

Input voltage: DC 5V

Working current: 15uA

LROBRUYA

Working temperature: -20 ~ 85 degrees Celsius

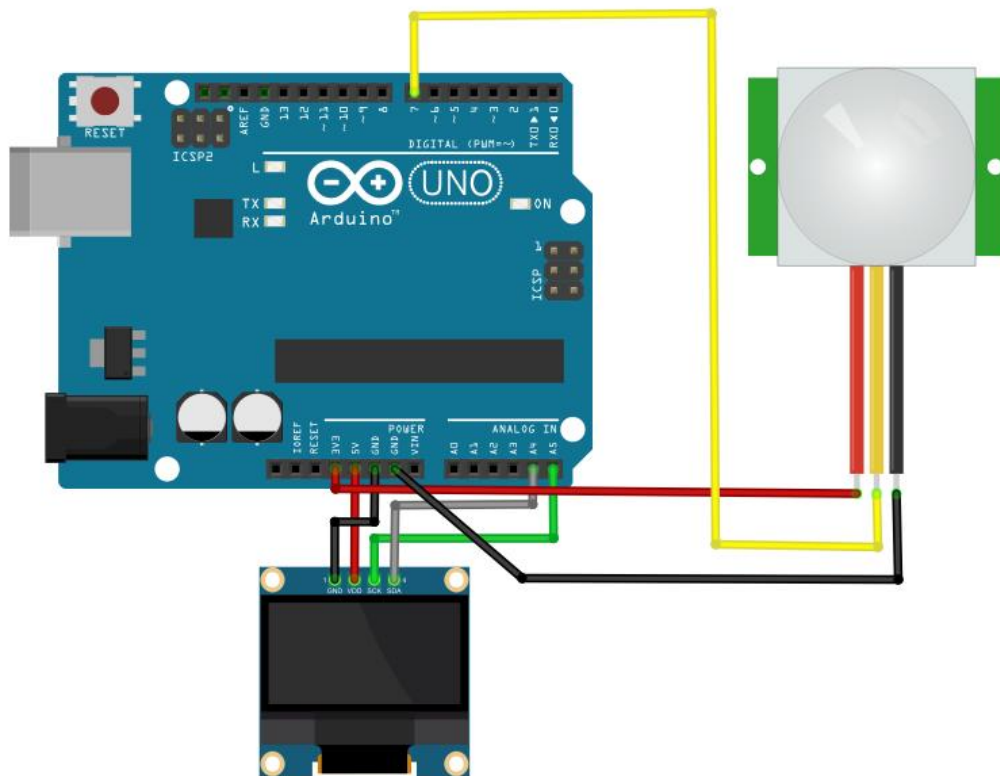
Output voltage: high 3 V, low 0 V

Detection angle: about 140 °

Detection distance: 3-4 meters

Pin limit current: 100mA

Connection Diagram



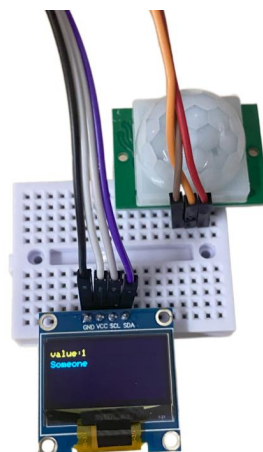
Sample Code

```
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <SimpleDHT.h>
// Initialize OLED display object and set resolution to 128x64
Adafruit_SSD1306 display = Adafruit_SSD1306(128, 64, &Wire);

int value=0;
```

```
void setup() {  
  // Initialize OLED display screen  
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);  
  display.clearDisplay();  
  pinMode(7, INPUT);  
}  
  
void loop() {  
  display.setTextSize(1);  
  display.setTextColor(WHITE);  
  value = digitalRead(7); // get adc value  
  // Set Text Style  
  display.setCursor(0, 8);  
  display.println("value:");  
  display.setCursor(35, 8);  
  display.println(value);  
  if(value==1){  
    display.setCursor(0, 18);  
    display.println("Someone");  
  } else {  
    display.setCursor(0, 18);  
    display.println("No Body");  
  }  
  display.display();  
  delay(1000);  
  display.clearDisplay();  
}
```

Result



Lesson 7 HC-SR04 Ultrasonic Sensor

Introduction

As the ultrasonic has strong direction, slow energy consumption and far spread distance in the media, so it is commonly used in the measurement of distance, such as range finder and position measuring instrument.

Using ultrasonic is more rapid, convenient, simple to calculate and more easier to achieve real-time control, so it has also been widely used in the development of mobile robots.

Ultrasonic detector module can provide 2cm-450cm non-contact sensing distance, and its ranging accuracy is up to 3mm, very good to meet the normal requirements. The module includes an ultrasonic transmitter and receiver as well as the corresponding control circuit.



Specification

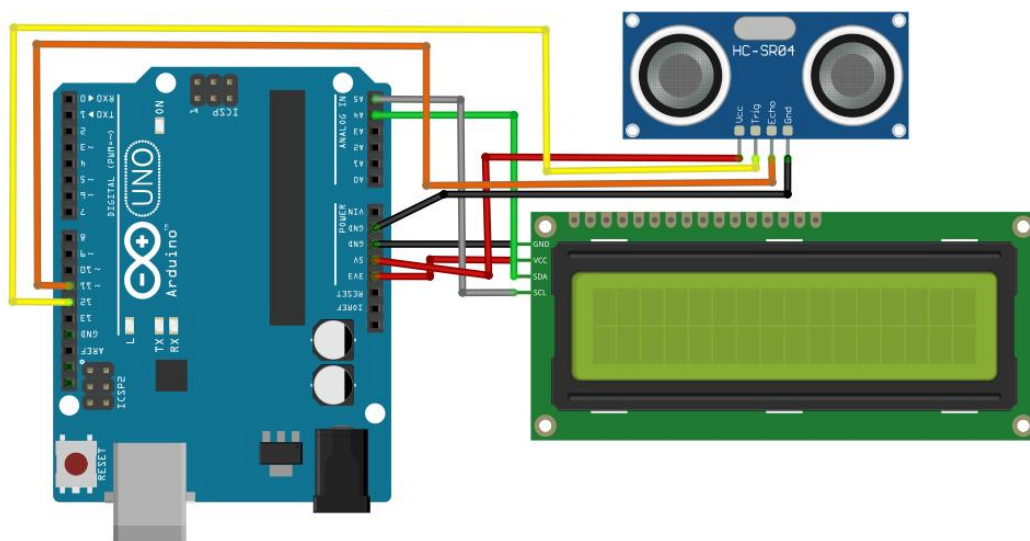
1. First pull down the TRIG, and then trigger it with at least 10us high level signal;

2. After triggering, the module will automatically transmit eight 40KHZ square waves, and automatically detect whether there is a signal to return.

3. If there is a signal returned back, through the ECHO to output a high level, the duration time of high level is actually the time from emission to reception of ultrasonic.

Test distance = high level duration * 340m/s * 0.5.

Connection Diagram



Sample Code

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd (0x27,16,2); // set the LCD address to 0x27 for a16 chars and
2 line display

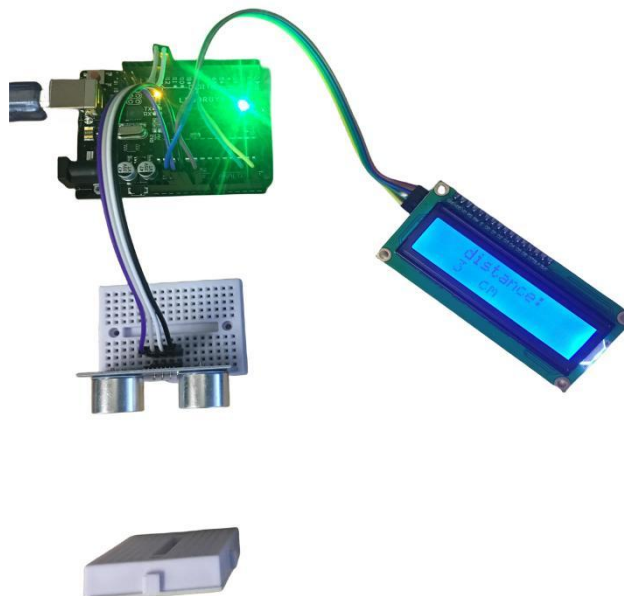
#include "SR04.h"
#define TRIG_PIN 12
#define ECHO_PIN 11
SR04 sr04 = SR04(ECHO_PIN,TRIG_PIN);
```

```
long a;

void setup() {
  lcd.init (); // initialize the lcd
  lcd.init (); // Print a message to the LCD.
  lcd.backlight ();
  Serial.begin(9600);
  delay(1000);
}

void loop() {
  a=sr04.Distance();
  Serial.print(a);
  Serial.println("cm");
  lcd.setCursor (3,0);
  lcd.print ("distance:");
  lcd.setCursor (3,1);
  lcd.print (a);
  lcd.setCursor (6,1);
  lcd.print ("cm");
  delay(1000);
  lcd.clear (); // initialize the lcd
}
```

Result

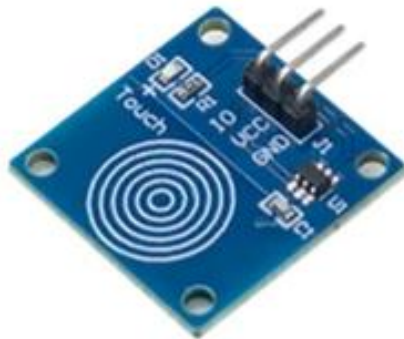


Lesson 8 Capacitive Touch

Introduction

Are you tired of clicking mechanic button? Well, try our capacitive touch sensor. You can find touch sensors mostly used on electronic device. So upgrade your Arduino project with our new version touch sensor and make it cool!

This little sensor can "feel" people and metal touch and feedback a high/low voltage level. Even isolated by some cloth and paper, it can still feel the touch. Its sensitivity decreases as isolation layer gets thicker.



Specification

Supply Voltage: 3.3V to 5V

Interface: Digital

Connection Diagram



```
int ledRed = 11;
int ledYel = 10;
int ledGre = 9;
int ledBlu = 8;
int KEY = 2; // Connect Touch sensor on Digital Pin 2

void setup(){
  pinMode(ledRed, OUTPUT);
  pinMode(ledYel, OUTPUT);
```

```
pinMode(ledGre, OUTPUT);
pinMode(ledBlu, OUTPUT);
pinMode(KEY, INPUT);          //Set touch sensor pin to input mode
}

void loop(){
  if(digitalRead(KEY)==HIGH) {    //Read Touch sensor signal
    digitalWrite(ledRed, HIGH);
    delay(500);
    digitalWrite(ledYel, HIGH);
    delay(500);
    digitalWrite(ledGre, HIGH);
    delay(500);
    digitalWrite(ledBlu, HIGH);
    delay(500);
    digitalWrite(ledRed, LOW);
    delay(500);
    digitalWrite(ledYel, LOW);
    delay(500);
    digitalWrite(ledGre, LOW);
    delay(500);
    digitalWrite(ledBlu, LOW);
  }
  else
  {
    digitalWrite(ledRed, LOW);    // if Touch sensor is LOW, then turn off
the led
    digitalWrite(ledYel, LOW);
    digitalWrite(ledGre, LOW);
    digitalWrite(ledBlu, LOW);
  }
}
```

Result

Done wiring and powered up, upload well the code, then touch the sensor with your finger, both D2 led on the sensor and four LED lights will be displayed in the form of flowing lights. Otherwise, those are turned off.

Lesson 9 Joystick Module

Introduction

Lots of robot projects need joystick. This module provides an affordable solution. By simply connecting to two analog inputs, the robot is at your commands with X, Y control.

It also has a switch that is connected to a digital pin. This joystick module can be easily connected to Arduino by IO Shield.



Specification

Supply Voltage: 3.3V to 5V

Interface: Analog x2, Digital x1

Connection Diagram



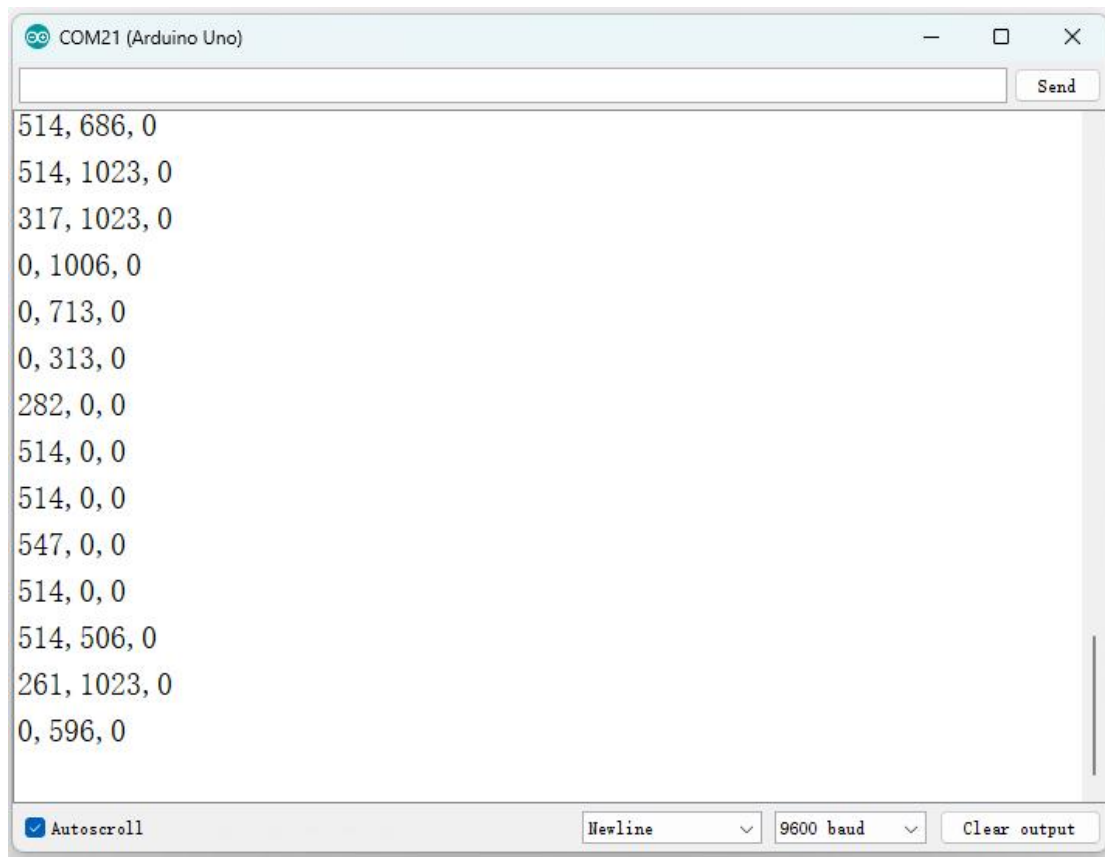
```
int JoyStick_X = 0; //x
int JoyStick_Y = 1; //y
int JoyStick_Z = 3; //key

void setup()
{
  pinMode(JoyStick_Z, INPUT);
  Serial.begin(9600); // 9600 bps
}

void loop()
{
  int x,y,z;
  x=analogRead(JoyStick_X);
  y=analogRead(JoyStick_Y);
  z=digitalRead(JoyStick_Z);
  Serial.print(x ,DEC);
  Serial.print(",");
  Serial.print(y ,DEC);
  Serial.print(",");
  Serial.println(z ,DEC);
  delay(100);
}
```

Result

Wiring well and uploading the code, open the serial monitor and set the baud rate to 9600, push the joystick, you will see the value shown below.



Lesson 10 5V Relay Module

Introduction

This single relay module can be used in interactive projects. It is active HIGH level. This module uses SONGLE 5v high-quality relay.

It can also be used to control lighting, electrical and other equipment.

The modular design makes it easy to expand with the Arduino board (not included). The relay output is by a light-emitting diode. It can be controlled through digital IO port, such as solenoid valves, lamps, motors and other high current or high voltage devices.



Specification

Type: Digital

Rated current: 10A (NO) 5A (NC)

Maximum switching voltage: 150VAC 24VDC

Digital interface

Control signal: TTL level

Rated load: 8A 150VAC (NO) 10A 24VDC (NO), 5A 250VAC (NO/NC)

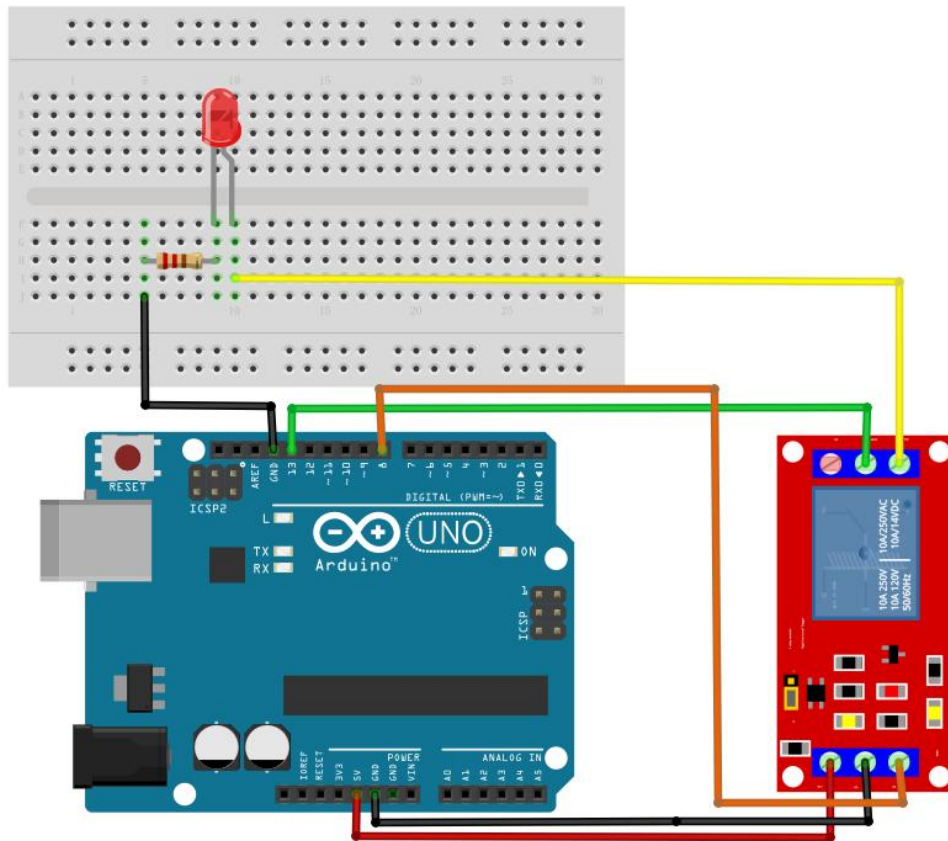
5A 24VDC (NO/NC)

Maximum switching power: AC1200VA DC240W (NO) AC625VA

DC120W (NC)

Contact action time: 10ms

Connection Diagram



Sample Code

```
int Relay = 8;
void setup()
{
  pinMode(13, OUTPUT);      //Set Pin13 as output
  digitalWrite(13, HIGH);    //Set Pin13 High
  pinMode(Relay, OUTPUT);    //Set Pin3 as output
}
void loop()
{
  digitalWrite(Relay, HIGH); //Turn off relay
  delay(2000);
  digitalWrite(Relay, LOW);  //Turn on relay
  delay(2000);
}
```



Result

This relay module is active HIGH level.

Wire it up well, powered up, then upload the above code to the board.

You will see the relay is turned on (ON connected, NC disconnected) for two seconds, then turned off for two seconds (NC closed, ON disconnected), repeatedly and circularly.

When the relay is turned on, external LED is on. If relay is turned off, external LED is off.

Lesson 11 Servo

Introduction

Servo is a type of geared motor that can only rotate 180 degrees. It is controlled by sending electrical pulses from your UNO R3 board. These pulses tell the servo what position it should move to. The Servo has three wires, of which the brown one is the ground wire and should be connected to the GND port of UNO, the red one is the power wire and should be connected to the 5v port, and the orange one is the signal wire and should be connected to the Dig #9 port.



Specification

Universal for JR and FP connector

Cable length : 25cm

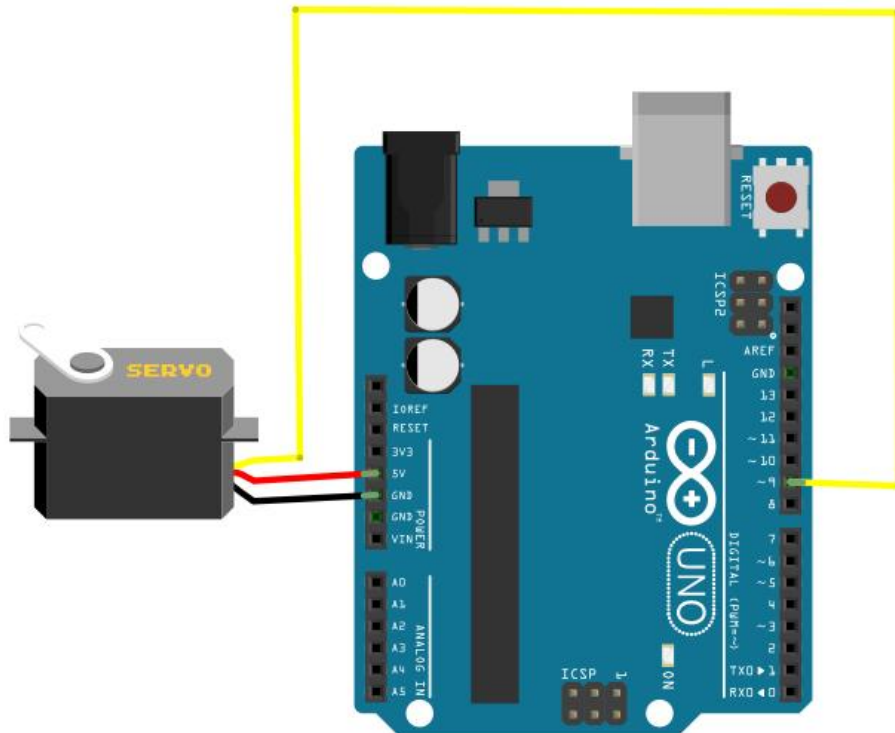
No load; Operating speed: 0.12 sec / 60 degree (4.8V), 0.10 sec / 60 degree (6.0V)

Stall torque (4.8V): 1.6kg/cm

Temperature : -30~60°C

Connection Diagram

LROBRUYA



Sample Code

```
#include <Servo.h>
Servo myservo; // create servo object to control a servo
int i=0;
void setup()
{
  Serial.begin(9600);
  myservo.attach(9); // modify each pin to adjust
  myservo.write(i);
  delay(1000);
}

void loop()
{
  for(i;i<180;i++)
  {
    myservo.write(i);
    delay(5);
  }
}
```

```
for(i;i>0;i--)  
{  
  myservo.write(i);  
  delay(5);  
}  
delay(1000);  
}
```

Result

After connecting, please open the the program and load up the code lesson11 onto your Arduino board. Before you can run this, make sure that you have installed the < Servo> library or re-install it, if necessary. Otherwise, your code won't work. For details about loading the library file, see [How to Install Arduino Driver](#).

Lesson 12 8X8 LED Dot Matrix Module

Introduction

This lattice is a common anode lattice, with row control LED anode. The code is controlled by setting high and low levels for each column, with high level 1 lit and low level 0 turned off.



Specification

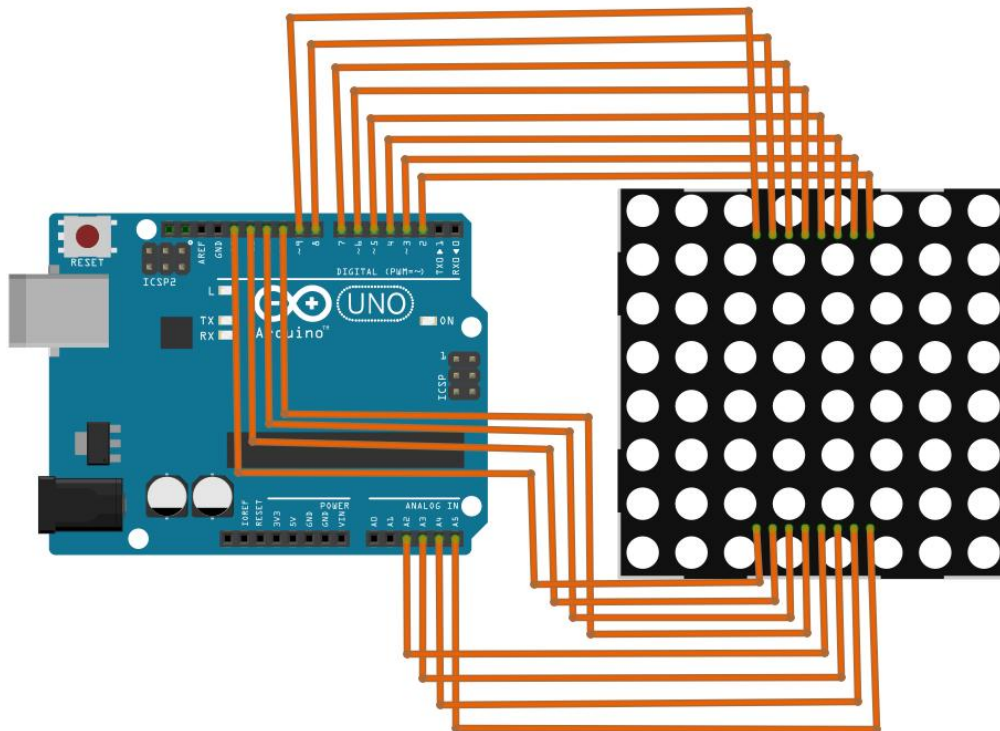
Operating Voltage: DC 3.3V - 5.3V

Typical Voltage: 3V

Operating Current: 320mA

Max Operating Current: 2A

Connection Diagram



Sample Code

```
#define H1 2  
#define H2 7  
#define H3 A5  
#define H4 5  
#define H5 13  
#define H6 A4  
#define H7 12  
#define H8 A2
```

```
#define L1 6  
#define L2 11  
#define L3 10  
#define L4 3  
#define L5 A3  
#define L6 4  
#define L7 8  
#define L8 9
```

```
unsigned char table1[8][8]=
```

LROBROYA

```
{  
  0,0,0,0,0,0,0,0,  
  0,1,1,0,0,1,1,0,  
  1,1,1,1,1,1,1,1,  
  1,1,1,1,1,1,1,1,  
  1,1,1,1,1,1,1,1,  
  0,1,1,1,1,1,1,0,  
  0,0,1,1,1,1,0,0,  
  0,0,0,1,1,0,0,0,  
};
```

```
unsigned char table2[8][8]=
```

```
{  
  0,0,0,0,0,0,0,0,  
  0,0,0,0,0,0,0,0,  
  0,0,1,0,0,1,0,0,  
  0,1,1,1,1,1,1,0,  
  0,1,1,1,1,1,1,0,  
  0,0,1,1,1,1,0,0,  
  0,0,0,1,1,0,0,0,  
  0,0,0,0,0,0,0,0,  
};
```

```
void setup()
```

```
{  
  pinMode(H1,OUTPUT);  
  pinMode(H2,OUTPUT);  
  pinMode(H3,OUTPUT);  
  pinMode(H4,OUTPUT);  
  pinMode(H5,OUTPUT);  
  pinMode(H6,OUTPUT);  
  pinMode(H7,OUTPUT);  
  pinMode(H8,OUTPUT);
```

```
  
  pinMode(L1,OUTPUT);  
  pinMode(L2,OUTPUT);  
  pinMode(L3,OUTPUT);  
  pinMode(L4,OUTPUT);  
  pinMode(L5,OUTPUT);  
  pinMode(L6,OUTPUT);
```

```
pinMode(L7,OUTPUT);
pinMode(L8,OUTPUT);

}

void loop()
{
    for(int i=0;i<100;i++)
    {
        Display(table1);    //Show Big Heart
    }
    for(int i=0;i<50;i++)
    {
        Display(table2);    //Show small heart
    }
}

void Display(unsigned char dat[8][8])
{
    digitalWrite(L1,LOW);    //First column
    digitalWrite(H1,dat[0][0]);
    digitalWrite(H2,dat[1][0]);
    digitalWrite(H3,dat[2][0]);
    digitalWrite(H4,dat[3][0]);
    digitalWrite(H5,dat[4][0]);
    digitalWrite(H6,dat[5][0]);
    digitalWrite(H7,dat[6][0]);
    digitalWrite(H8,dat[7][0]);
    delay(2);
    Clear();

    digitalWrite(L2,LOW);    //Second column
    digitalWrite(H1,dat[0][1]);
    digitalWrite(H2,dat[1][1]);
    digitalWrite(H3,dat[2][1]);
    digitalWrite(H4,dat[3][1]);
    digitalWrite(H5,dat[4][1]);
    digitalWrite(H6,dat[5][1]);
    digitalWrite(H7,dat[6][1]);
    digitalWrite(H8,dat[7][1]);
    delay(2);
```

```
Clear();
```

```
digitalWrite(L3,LOW);           //Third column
digitalWrite(H1,dat[0][2]);
digitalWrite(H2,dat[1][2]);
digitalWrite(H3,dat[2][2]);
digitalWrite(H4,dat[3][2]);
digitalWrite(H5,dat[4][2]);
digitalWrite(H6,dat[5][2]);
digitalWrite(H7,dat[6][2]);
digitalWrite(H8,dat[7][2]);
delay(2);
Clear();
```

```
digitalWrite(L4,LOW);           //Fourth column
digitalWrite(H1,dat[0][3]);
digitalWrite(H2,dat[1][3]);
digitalWrite(H3,dat[2][3]);
digitalWrite(H4,dat[3][3]);
digitalWrite(H5,dat[4][3]);
digitalWrite(H6,dat[5][3]);
digitalWrite(H7,dat[6][3]);
digitalWrite(H8,dat[7][3]);
delay(2);
Clear();
```

```
digitalWrite(L5,LOW);           //Fifth column
digitalWrite(H1,dat[0][4]);
digitalWrite(H2,dat[1][4]);
digitalWrite(H3,dat[2][4]);
digitalWrite(H4,dat[3][4]);
digitalWrite(H5,dat[4][4]);
digitalWrite(H6,dat[5][4]);
digitalWrite(H7,dat[6][4]);
digitalWrite(H8,dat[7][4]);
delay(2);
Clear();
```

```
digitalWrite(L6,LOW);           //Sixth column
digitalWrite(H1,dat[0][5]);
digitalWrite(H2,dat[1][5]);
```



```
digitalWrite(H3,dat[2][5]);
digitalWrite(H4,dat[3][5]);
digitalWrite(H5,dat[4][5]);
digitalWrite(H6,dat[5][5]);
digitalWrite(H7,dat[6][5]);
digitalWrite(H8,dat[7][5]);
delay(2);
Clear();
```

```
digitalWrite(L7,LOW); //Seventh column
digitalWrite(H1,dat[0][6]);
digitalWrite(H2,dat[1][6]);
digitalWrite(H3,dat[2][6]);
digitalWrite(H4,dat[3][6]);
digitalWrite(H5,dat[4][6]);
digitalWrite(H6,dat[5][6]);
digitalWrite(H7,dat[6][6]);
digitalWrite(H8,dat[7][6]);
delay(2);
Clear();
```

```
digitalWrite(L8,LOW); //Eighth column
digitalWrite(H1,dat[0][7]);
digitalWrite(H2,dat[1][7]);
digitalWrite(H3,dat[2][7]);
digitalWrite(H4,dat[3][7]);
digitalWrite(H5,dat[4][7]);
digitalWrite(H6,dat[5][7]);
digitalWrite(H7,dat[6][7]);
digitalWrite(H8,dat[7][7]);
delay(2);
Clear();
}
```

```
void Clear()
{
digitalWrite(L1,HIGH); // Let all the dots go out
digitalWrite(L2,HIGH);
digitalWrite(L3,HIGH);
digitalWrite(L4,HIGH);
digitalWrite(L5,HIGH);
```

```
digitalWrite(L6,HIGH);  
digitalWrite(L7,HIGH);  
digitalWrite(L8,HIGH);
```

```
digitalWrite(H1,LOW);  
digitalWrite(H2,LOW);  
digitalWrite(H3,LOW);  
digitalWrite(H4,LOW);  
digitalWrite(H5,LOW);  
digitalWrite(H6,LOW);  
digitalWrite(H7,LOW);  
digitalWrite(H8,LOW);  
}
```

Result

After running the code, you can see dynamic changes in the size of the dot matrix lights.

Lesson 13 Four Digital Seven Segment Display

Introduction

As shown in the figure, the digital tube used this time has a common cathode, and the four digital tubes have 12 pins, which can be divided into bit selection pins and segment selection pins.

Segment selection pin: 8 pins a, b, c, d, e, f, g

Bit selection pin: 4 pins D1, D2, D3, D4

The display of which digital tube is determined by the chip selection pin. If the chip selection pin is high, the digital tube will light up, and if it is low, the digital tube will turn off.

The numerical value displayed on the digital tube is determined by the segment selection pin. For example, if you want to display the number 8, a, b, c, d, e, f, and g are the high level and h is the low level; To display the number 1, b and c are high levels, and a, d, e, f, g, and h are low levels.

Display principle: The display principle of the four digital tubes is to continuously scan D1, D2, D3, D4, and then the corresponding eight segment tubes will light up sequentially. Due to its fast lighting speed, the human eye cannot see it (the refresh rate recognized by the human eye is 30Hz), so it looks like four digital tubes displaying simultaneously.





```
#define D_a 2
#define D_b 3
#define D_c 4
#define D_d 5
#define D_e 6
#define D_f 7
```



```
#define D_g 8
#define D_h 9

#define COM1 10
#define COM2 11
#define COM3 12
#define COM4 13

bool Digital_tube_number[18][8] = {
    {0,0,0,0,0,0,1,1},//0
    {1,0,0,1,1,1,1,1},//1
    {0,0,1,0,0,1,0,1},//2
    {0,0,0,0,1,1,0,1},//3
    {1,0,0,1,1,0,0,1},//4
    {0,1,0,0,1,0,0,1},//5
    {0,1,0,0,0,0,0,1},//6
    {0,0,0,1,1,1,1,1},//7
    {0,0,0,0,0,0,0,1},//8
    {0,0,0,0,1,0,0,1},//9
    {0,0,0,1,0,0,0,0},//A10
    {0,0,0,0,0,0,0,0},//B11
    {0,1,1,0,0,0,1,0},//C12
    {0,0,0,0,0,0,1,0},//D13
    {0,1,1,0,0,0,0,0},//E14
    {0,1,1,1,0,0,0,0},//F15
    {0,1,0,0,0,0,1,0},//G16
    {1,1,1,1,1,1,1,0},//decimal point.
};

int delay_time = 2;

void setup() {
    // put your setup code here, to run once:
    for(int i=D_a;i<=D_h;i++){
        pinMode(i,OUTPUT);
    }
    for(int i=COM1;i<=COM4;i++){
        pinMode(i,OUTPUT);
    }
    Digital_tube_display_sametime(5);
}
```



```
void loop() {
    // put your main code here, to run repeatedly:
    Digital_tube_dynamic_display(1,2,3,4,6,1);
    Digital_tube_dynamic_display(3,6,5,8,6,1);
    Digital_tube_dynamic_display(10,11,12,13,6,1);
}

/* dynamic display
*Display on digital tube 1, display on digital tube 2, display on digital tube 3, display
on digital tube 4,
*time1 visual persistence minimum time, time2 dynamic time
*/
void Digital_tube_dynamic_display(int a,int b,int c,int d,int time1,float time2)
{
    if(time1>6){
        time1 = 6;
    }
    for(int i=0;i<time2*250/time1;i++){
        Digital_tube_display(1,a);
        delay(time1);
        Digital_tube_display(2,b);
        delay(time1);
        Digital_tube_display(3,c);
        delay(time1);
        Digital_tube_display(4,d);
        delay(time1);
    }
}

void Digital_tube_display_sametime(int number)
{
    digitalWrite(COM1,HIGH);
    digitalWrite(COM2,HIGH);
    digitalWrite(COM3,HIGH);
    digitalWrite(COM4,HIGH);
    for(int i=0;i<8;i++){
        digitalWrite(2+i,Digital_tube_number[number][i]);
    }
}

void Digital_tube_display(int com,int number)
```

```
{
  for(int i=COM1;i<=COM4;i++){
    digitalWrite(i,LOW);
  }
  switch(com){
    case 1:
      digitalWrite(COM1,HIGH);
      break;
    case 2:
      digitalWrite(COM2,HIGH);
      break;
    case 3:
      digitalWrite(COM3,HIGH);
      break;
    case 4:
      digitalWrite(COM4,HIGH);
      break;
  }
  for(int i=0;i<8;i++){
    digitalWrite(2+i,Digital_tube_number[number][i]);
  }
}
```

Result

Dynamic changes in exhibition display numbers.