LAFVIN

Super Starter Kit for Arduino Uno

Content

Company Profile....1
Packing list....3
Lesson 0 Installing IDE....5
Lesson 1 Add Libraries and Open Serial Monitor....16
Lesson 2 Blink....25
Lesson 3 LED....35
Lesson 4 RGB LED....42
Lesson 5 Digital Inputs....51
Lesson 6 Tilt Ball Switch....56
Lesson 7 Active buzzer....60
Lesson 8 Passive Buzzer....64
Lesson 9 DHT11 Temperature and Humidity Sensor....68
Lesson 10 IR Receiver Module....74
Lesson 11 Servo....80
Lesson 12 Analog Joystick Module....84
Lesson 13 Water Level Detection Sensor Module....89
Lesson 14 Ultrasonic Sensor Module....94
Lesson 15 Sound Sensor Module....99
Lesson 16 Membrane Switch Module....105
Lesson 17 Eight LED with 74HC595....111
Lesson 18 The Serial Monitor....118
Lesson 19 Photocell....124
Lesson 20 LCD1602 I2C Module....129
Lesson 21 Thermometer....134
Lesson 22 74HC595 And Segment Display....139
Lesson 23 Four Digital Seven Segment Display....145
Lesson 24 DC Motors....150
Lesson 25 Relay....160
Lesson 26 Stepper Motor....165
Lesson 27 Controlling Stepper Motor With Remote....173

Company Profile

Established in 2011, lafvin is a manufacturer and trader specialized in research,development and production of 2560 uno,nano boards,and all kinds of accessories or sensors use for arduino,rasperrry.We also complete starter kits designed for interested lovers of any levels to learn Arduino or Raspberry.We are located in Shenzhen,China.All of our products comply with international quality standards and are greatly appreciated in a variety of different markets throughout the world.

Customer Service

We are cooperating with a lot of companies from diffirent countries.Also help them to purchase electronic component products in china,and became the biggest supplier of them. We look forward to build cooperate with more companies in future.

By the way,We also look forward to hearing from you and any of your critical comment or suggestions.Pls email us by lafvin_service@163.com if you have any questions or suggestions.

As a continuous and fast growing company. We keep striving our best to offer you excellent products and quality service.

Our Store

Aliexpress store: https://www.aliexpress.com/store/1942043

Alibaba Store:https://lafvin.en.alibaba.com/

Brand in Amazon:LAFVIN

Product Catalog

https://drive.google.com/drive/folders/0BwvEeRN9dKllblZING00TkhYbGs?usp=sharing

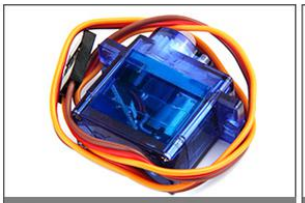
Tutorial

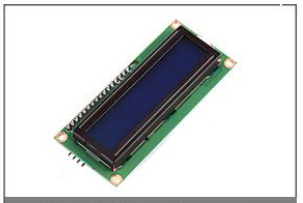
This tutorial include codes,labraries and lessons. It is designed for beginners. You will learn all the basic knowledge about how to use Arduino controller board, sensors and components.

Packing list


65 Jumper Wire 1PCS


Stepper Motor 1PCS


Servo Motor(SG90) 1PCS

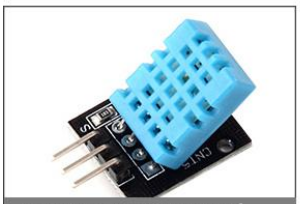

LCD1602 Module (with pin header)1PCS


Joystick Module 1PCS


Membrance Switch Module 1PCS

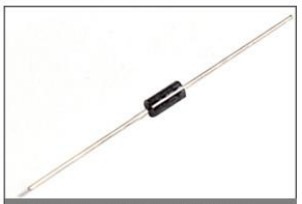

5V Relay 1PCS


F-M Dupont Wire 10PCS


DHT11 Temperature and Humidity Module 1PCS


Potentiometer(10K) 1PCS


9V Battery Connector with DC 1PCS


Diode Rectifier (1N4007) 2PCS


Small Button 5PCS


Thermistor 1PCS

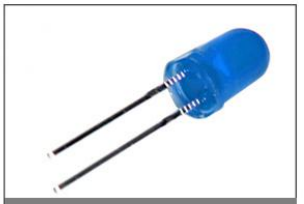

PN2222 2PCS

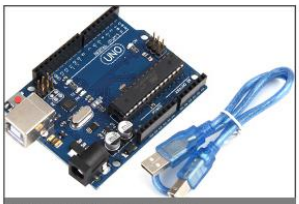

RGB Led 1PCS


Red Led 5PCS


White Led 5PCS


Yellow Led 5PCS


Blue Led 5PCS



UNO R3 Board with Cable 1PCS



Prototype Expansion 1PCS



830 Tie-Points Breadboard 1PCS



Remote 1PCS



Power Supply Module 1PCS



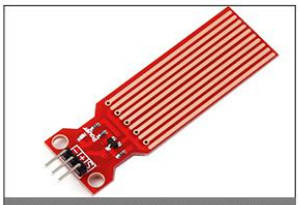
Resistor 100PCS



ULN2003 Stepper Motor Driver Board 1PCS



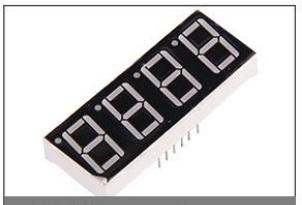
Ultrasonic Sensor 1PCS



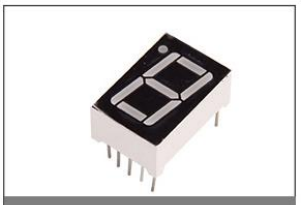
Water Lever Detection Sensor Module 1PCS



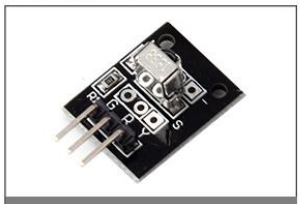
Fan Blade and 3V DC Motor each 1PCS



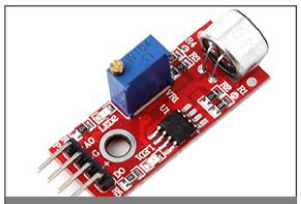
4 Digit 7-segment Display 1PCS



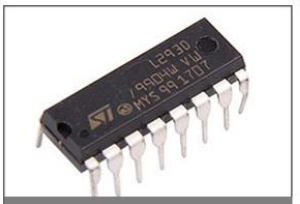
1 Digit 7-segment Display 1PCS



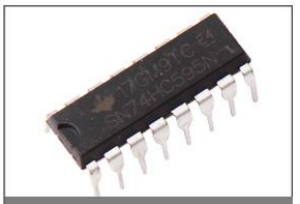
IR Receiver Module 1pcs



Sound Sensor Module 1PCS



L293D 1PCS



74HC595 1PCS



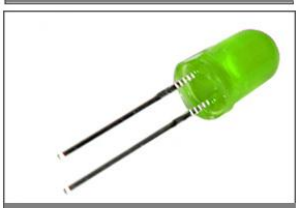
Tilt Switch 1PCS



Active Buzzer 1PCS
Passive Buzzer 1PCS



Photoresistor 2PCS



Green Led 5PCS

Lesson 0 Installing IDE

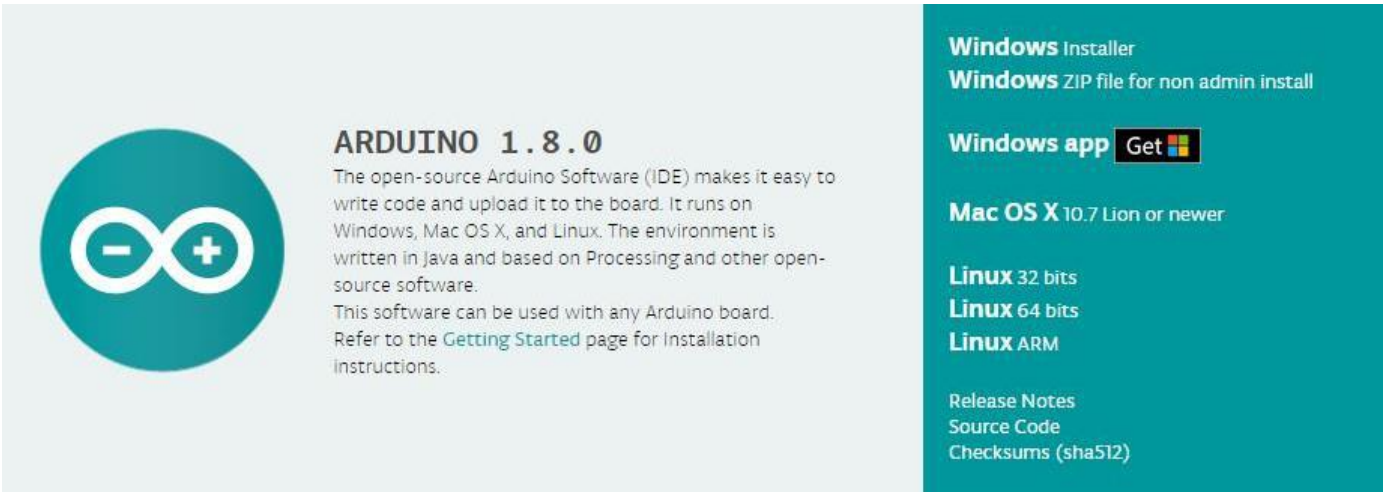
Introduction

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

In this lesson, you will learn how to setup your computer to use Arduino and how to set about the lessons 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/Main/Software and find below page.](https://www.arduino.cc/en/Main/Software)



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.

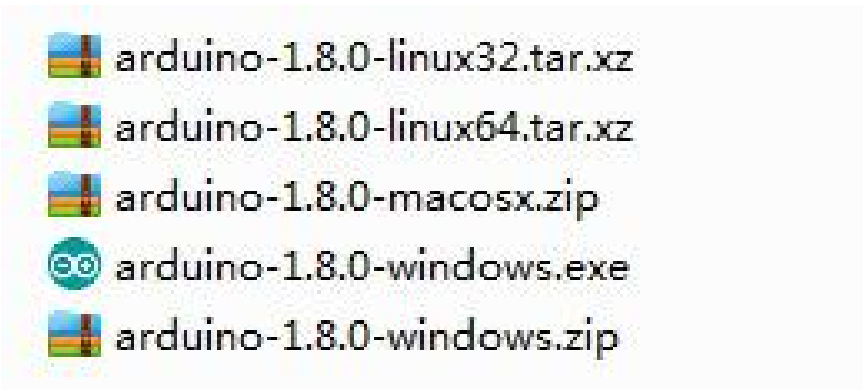


Click Windows Installer.



Click JUST DOWNLOAD.

Also version 1.8.0 is available in the material we provided, and the versions of our materials are the latest versions when this course was made.



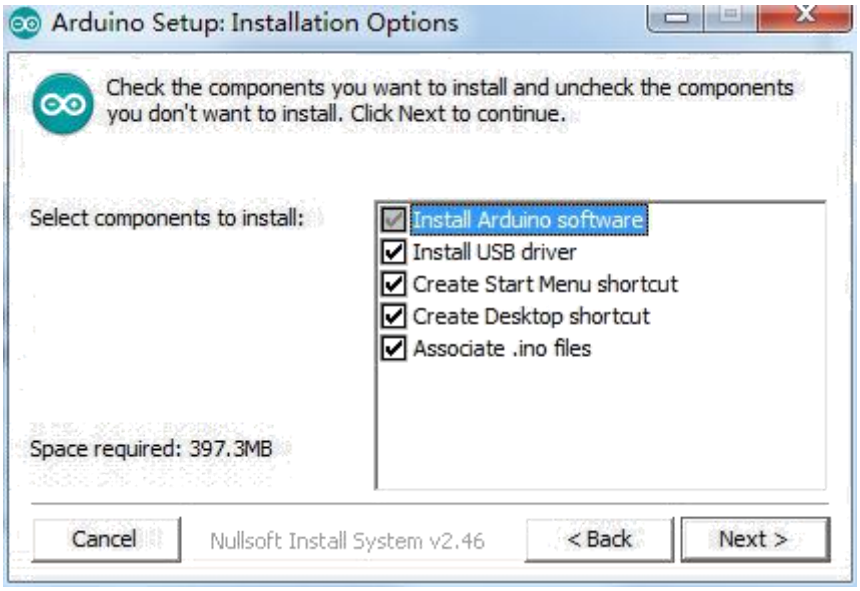
Installing Arduino (Windows)

Install Arduino with the exe. Installation package.

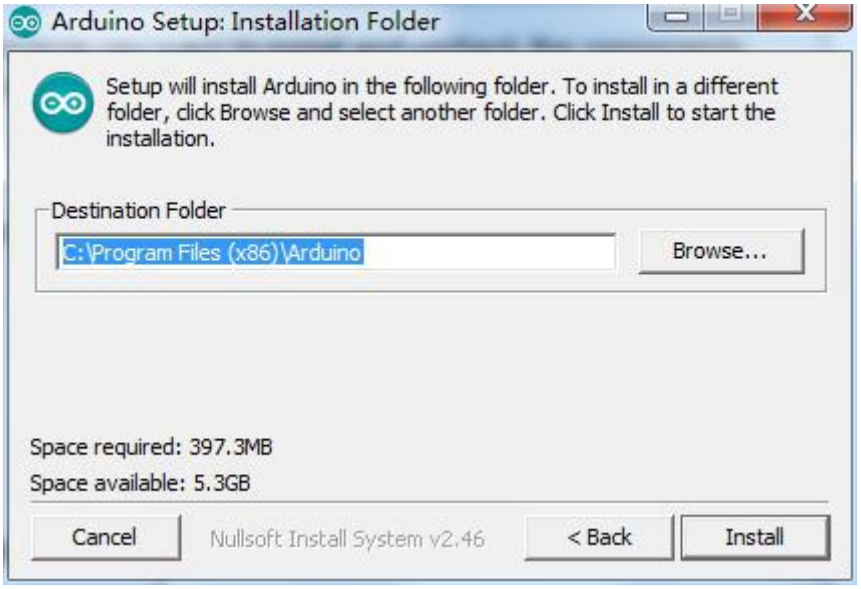




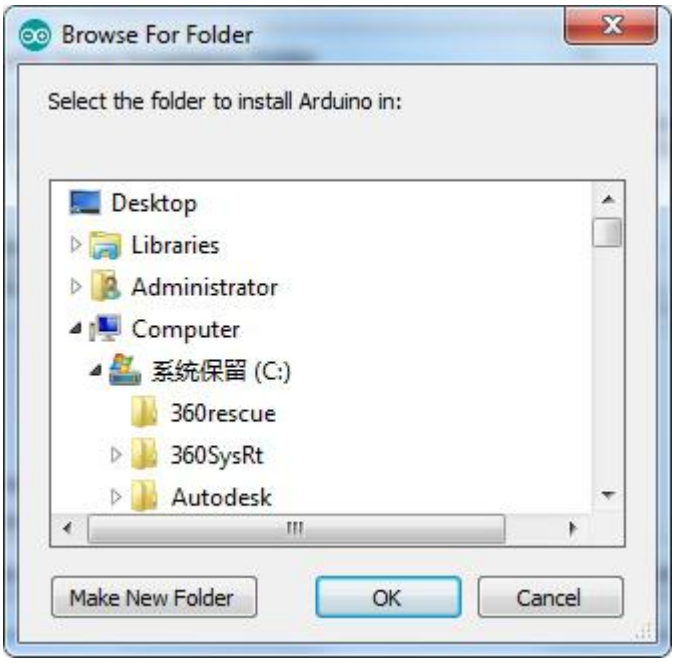
Click I Agree to see the following interface



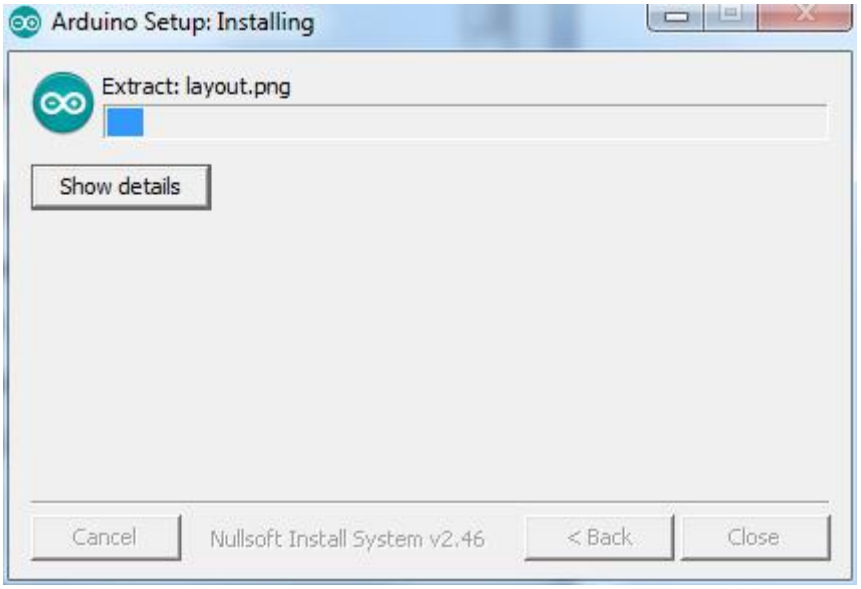
Click Next



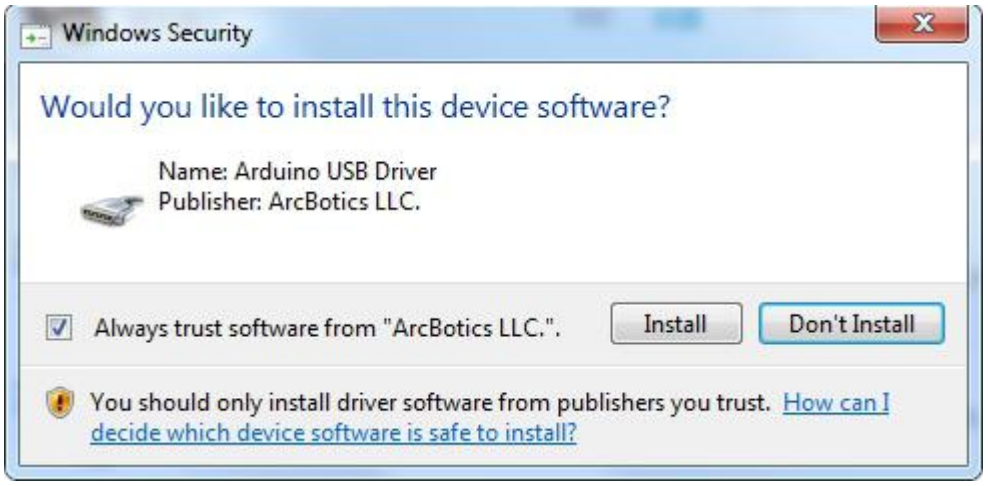
You can press Browse… to choose an installation path or directly type in the directory you want.



Click Install to initiate installation



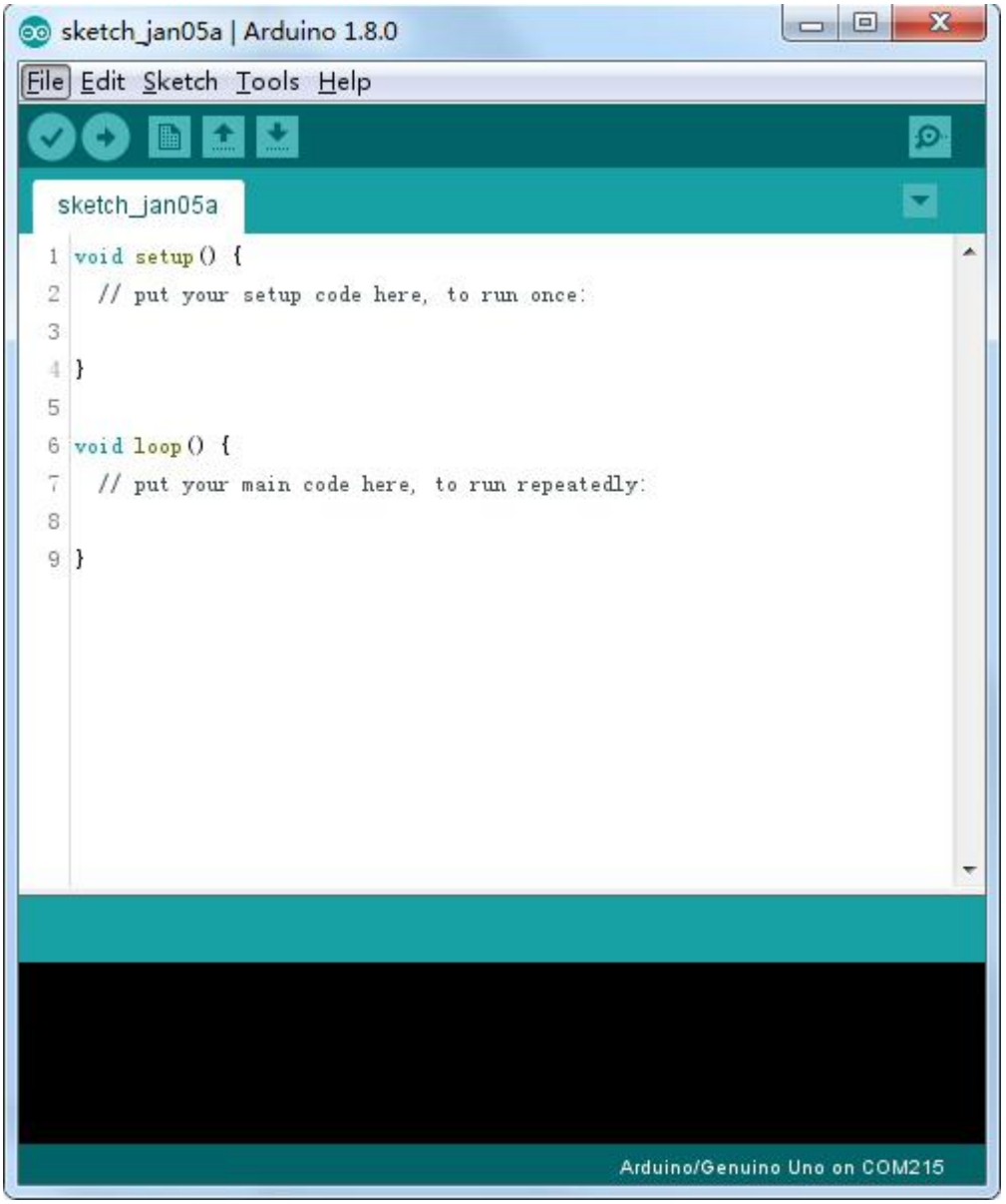
Finally, the following interface appears, click Install to finish the installation.



Next, the following icon appears on the desktop



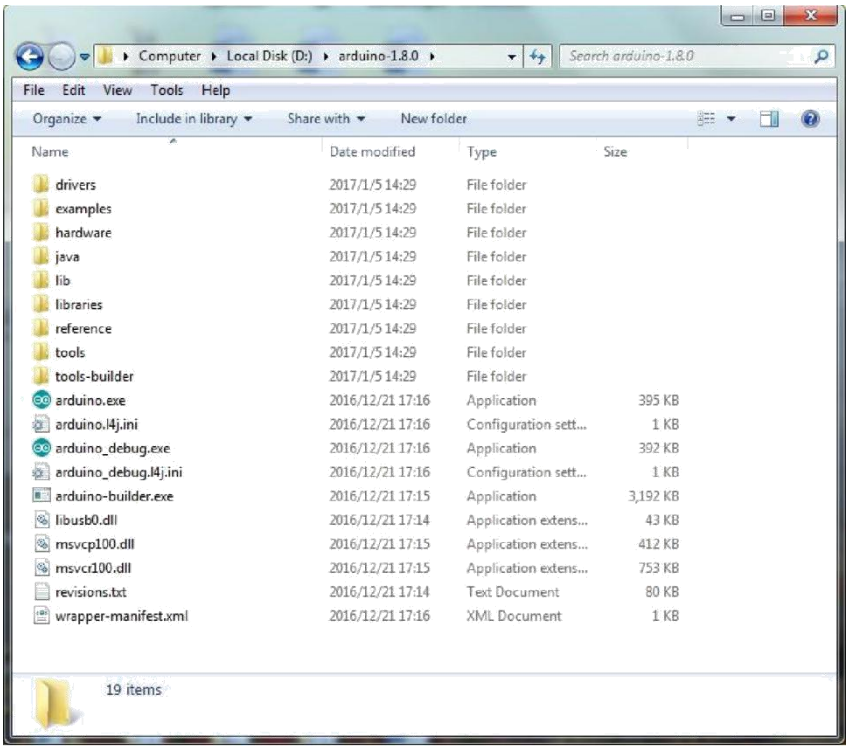
Double-click to enter the desired development environment



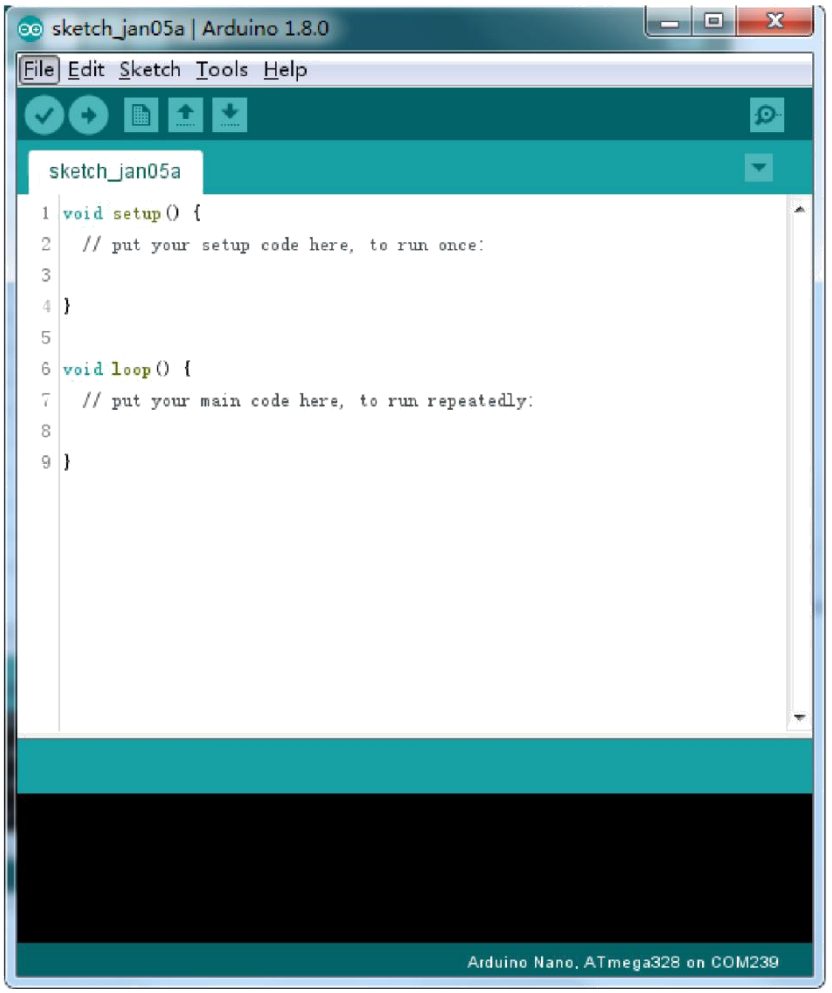
You may directly choose the installation package for installation and skip the contents below and jump to the next section. But if you want to learn some methods other than the installation package, please continue to read the section.

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




Computer ▸ Local Disk (D:) ▸ arduino-1.8.0 ▸
Search arduino-1.8.0
File Edit View Tools Help
Organize Include in library Share with New folder
Name Date modified Type Size
drivers 2017/1/5 14:29 File folder
examples 2017/1/5 14:29 File folder
hardware 2017/1/5 14:29 File folder
java 2017/1/5 14:29 File folder
lib 2017/1/5 14:29 File folder
libraries 2017/1/5 14:29 File folder
reference 2017/1/5 14:29 File folder
tools 2017/1/5 14:29 File folder
tools-builder 2017/1/5 14:29 File folder
arduino.exe 2016/12/21 17:16 Application 395 KB
arduino.l4j.ini 2016/12/21 17:16 Configuration sett... 1 KB
arduino_debug.exe 2016/12/21 17:16 Application 392 KB
arduino_debug.l4j.ini 2016/12/21 17:16 Configuration sett... 1 KB
arduino-builder.exe 2016/12/21 17:15 Application 3,192 KB
libusb0.dll 2016/12/21 17:14 Application extens... 43 KB
msvcp100.dll 2016/12/21 17:15 Application extens... 412 KB
msvcr100.dll 2016/12/21 17:15 Application extens... 753 KB
revisions.txt 2016/12/21 17:14 Text Document 80 KB
wrapper-manifest.xml 2016/12/21 17:16 XML Document 1 KB
19 items


arduino.exe


sketch_jan05a | Arduino 1.8.0
File Edit Sketch Tools Help
sketch_jan05a
1 void setup() {
2 // put your setup code here, to run once:
3
4 }
5
6 void loop() {
7 // put your main code here, to run repeatedly:
8
9 }
Arduino Nano, ATmega328 on COM239

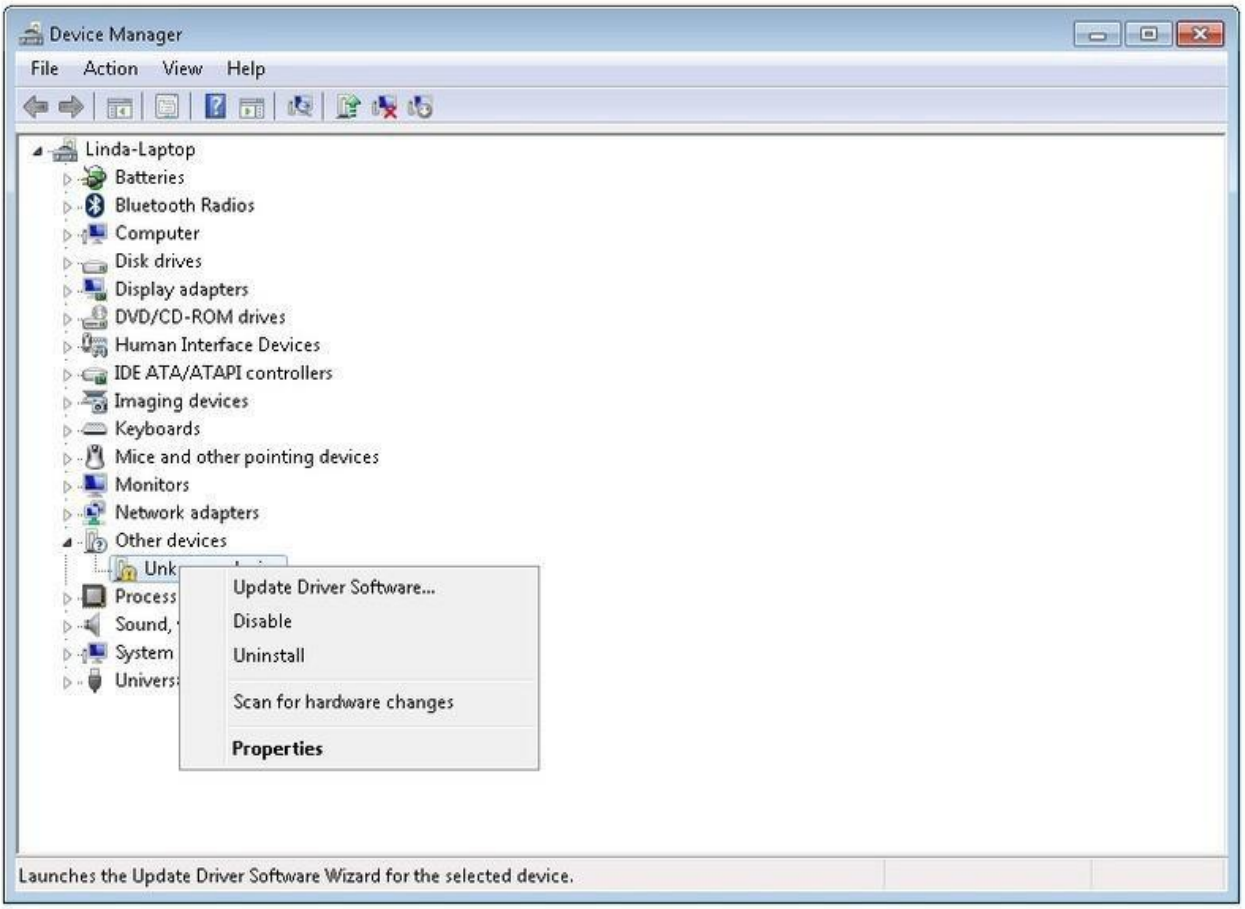
However, this installation method needs separate installation of driver.

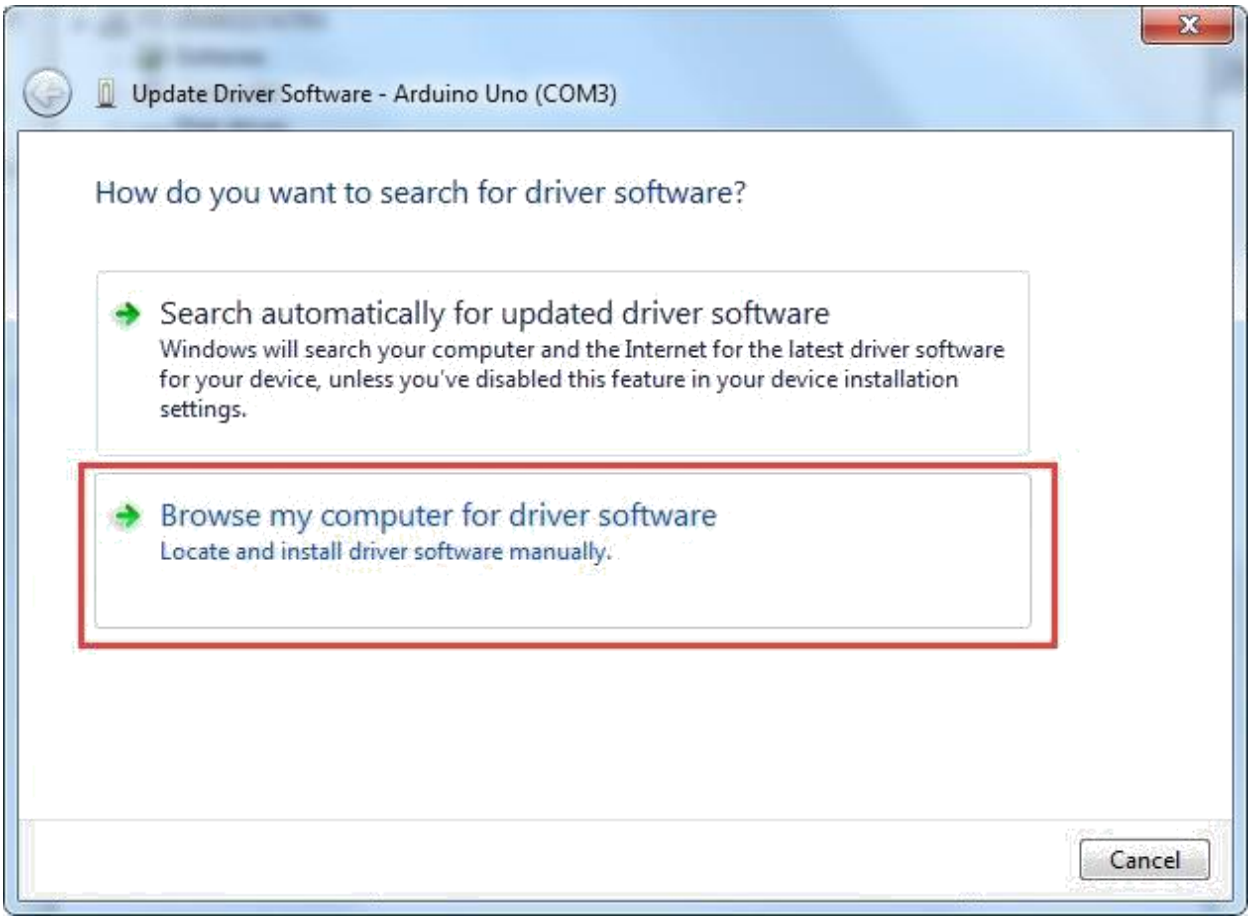
The Arduino folder contains both the Arduino program itself and the drivers that allow the Arduino to be connected to your computer by a USB cable. Before we launch the Arduino software, you are going to install the USB drivers.

Plug one end of your USB cable into the Arduino and the other into a USB socket on your computer. The power light on the LED will light up and you may get a 'Found New Hardware' message from Windows. Ignore this message and cancel any attempts that Windows makes to try and install drivers automatically for you.

The most reliable method of installing the USB drivers is to use the Device Manager. This is accessed in different ways depending on your version of Windows. In Windows 7, you first have to open the Control Panel, then select the option to view Icons, and you should find the Device Manager in the list.

Under ‘Other Devices’, you should see an icon for ‘unknown device’ with a little yellow warning triangle next to it. This is your Arduino.





Right-click on the device and select the top menu option (Update Driver Software...).

You will then be prompted to either 'Search Automatically for updated driver software' or 'Browse my computer for driver software'. Select the option to browse and navigate to the X\arduino1.8.0\drivers.



Click 'Next' and you may get a security warning, if so, allow the software to be installed. Once the software has been installed, you will get a confirmation message.



Windows users may skip the installation directions for Mac and Linux systems and jump to Lesson 1. Mac and Linux users may continue to read this section.

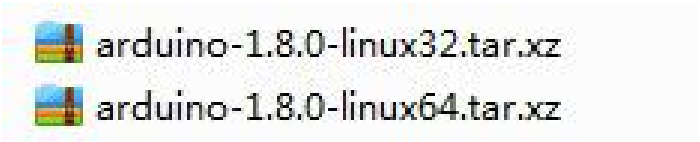
Installing Arduino (Mac OS X)

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.



Lesson 1 Add Libraries and Open Serial Monitor

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.

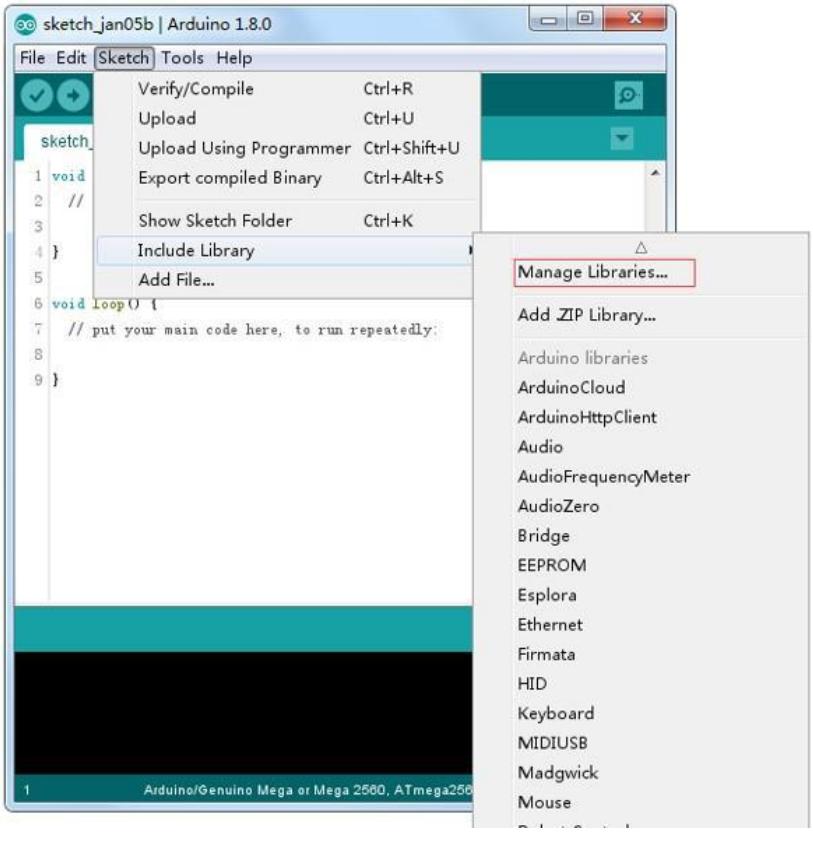
What are Libraries?

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 LiquidCrystal 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

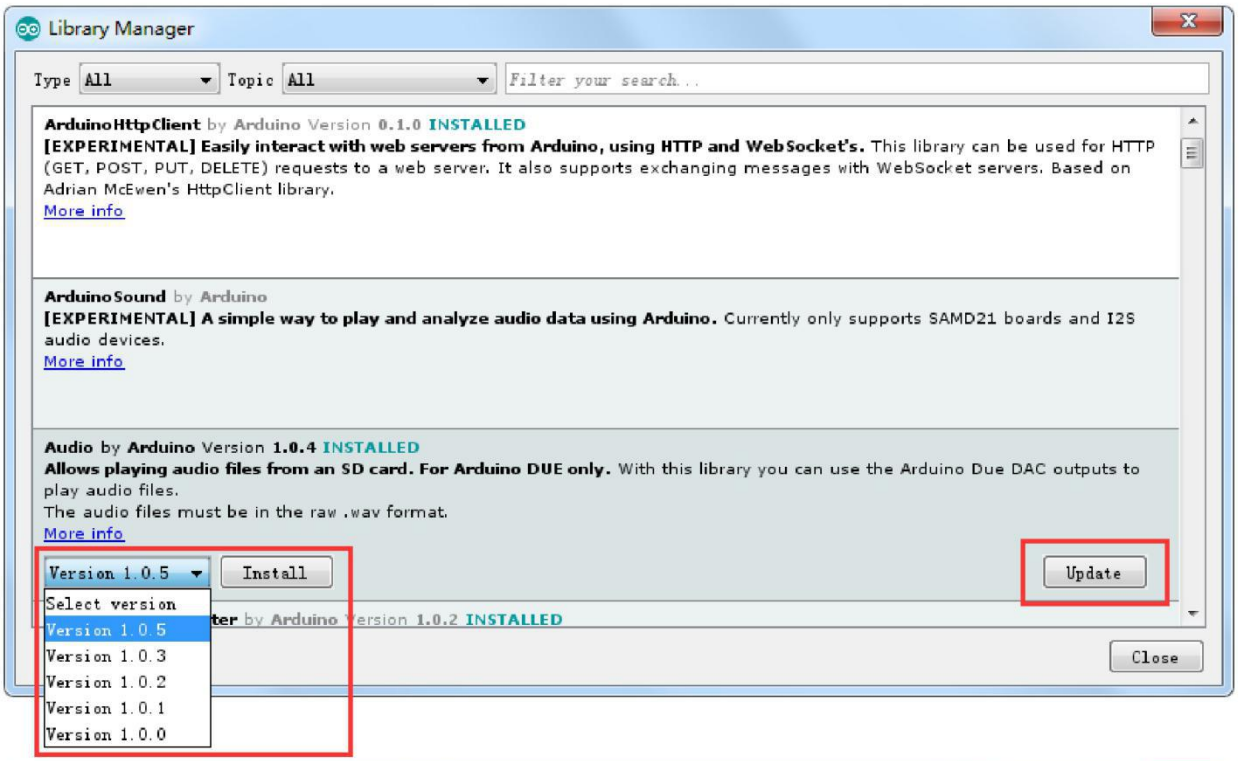
Using the Library Manager

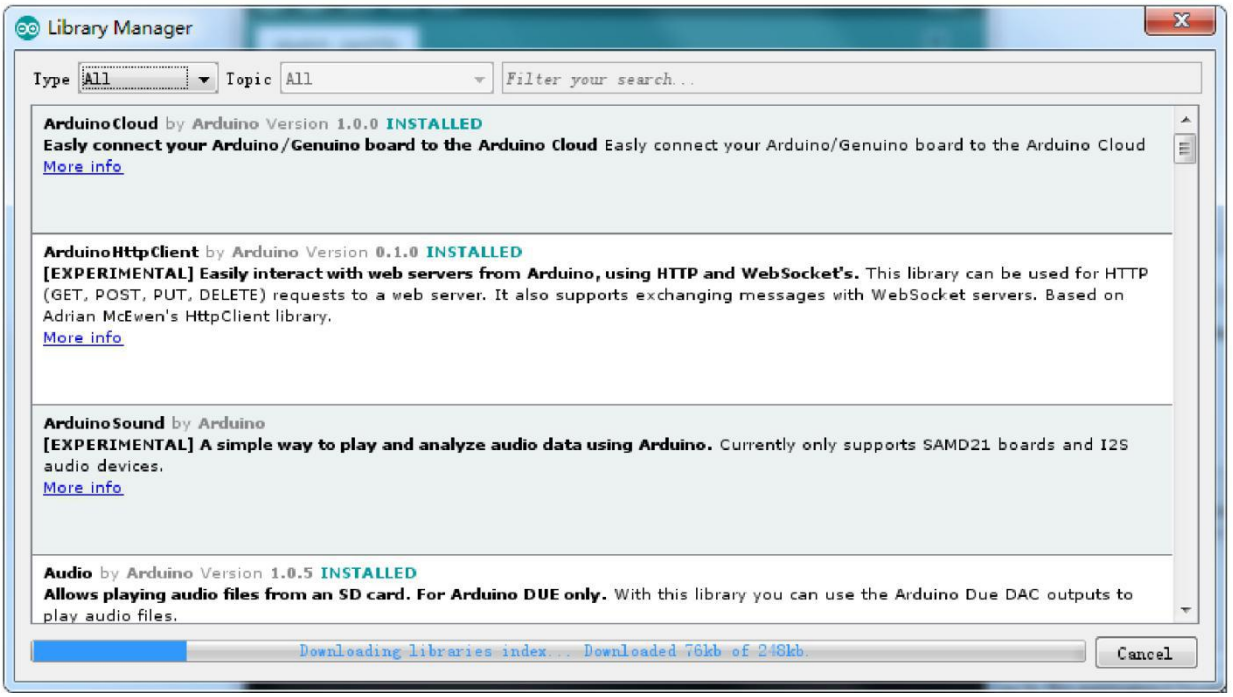
To install a new library into your Arduino IDE you can use the Library Manager (available from IDE version 1.8.0). Open the IDE and click to the "Sketch" menu and then Include Library > Manage Libraries.



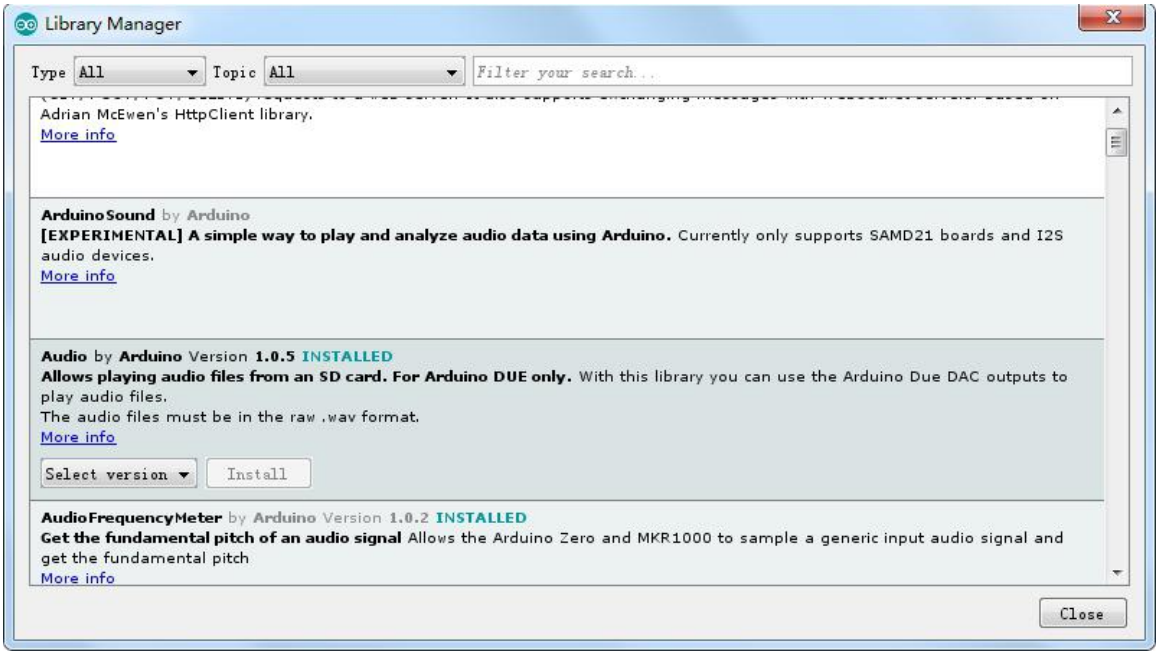
Then the library manager will open and you will find a list of libraries that are already installed or ready for installation. In this example we will install the Bridge library. Scroll the list to find it, then select the version of the library you want to install. Sometimes only one version of the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

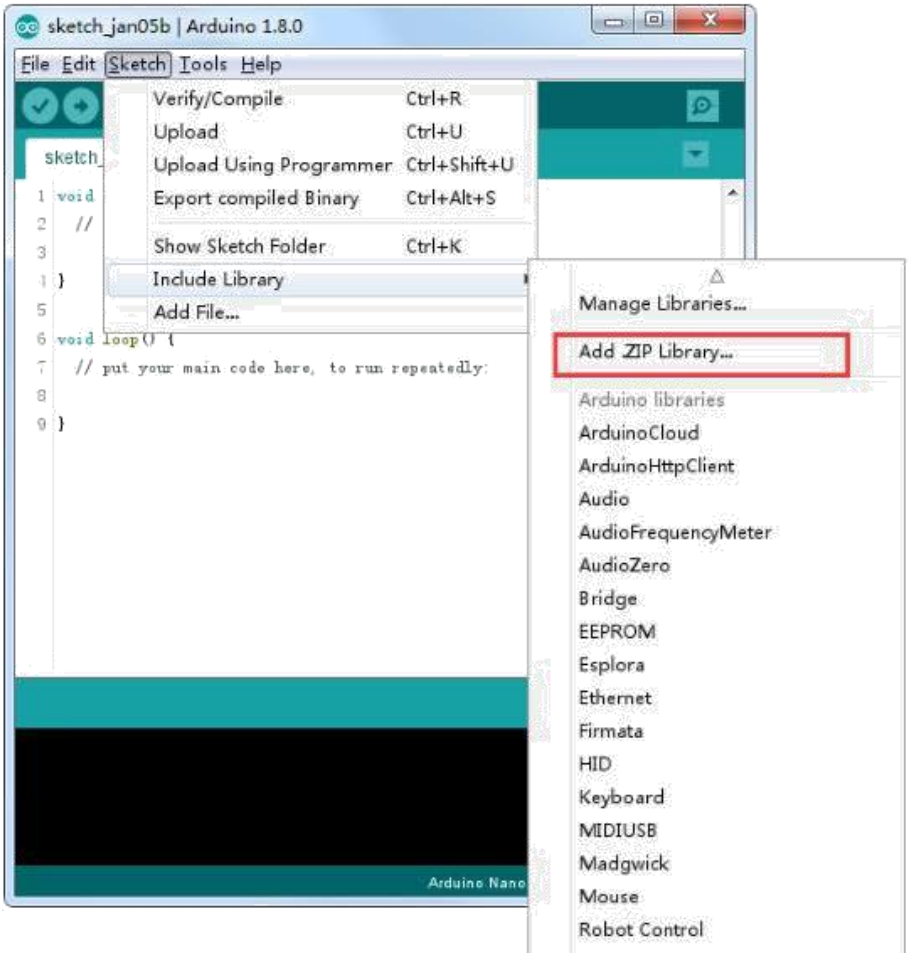


You can now find the new library available in the Include Library menu. If you want to add your own library open a new issue on Github.

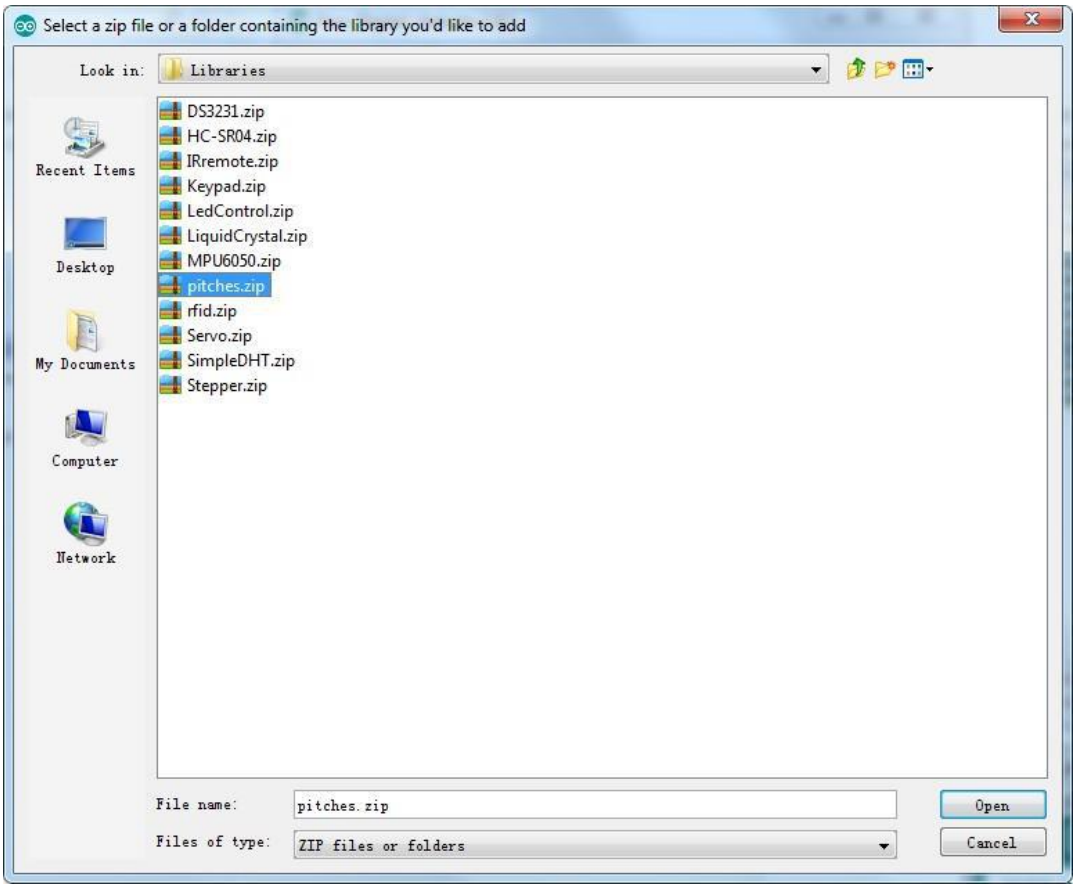
Importing a .zip Library

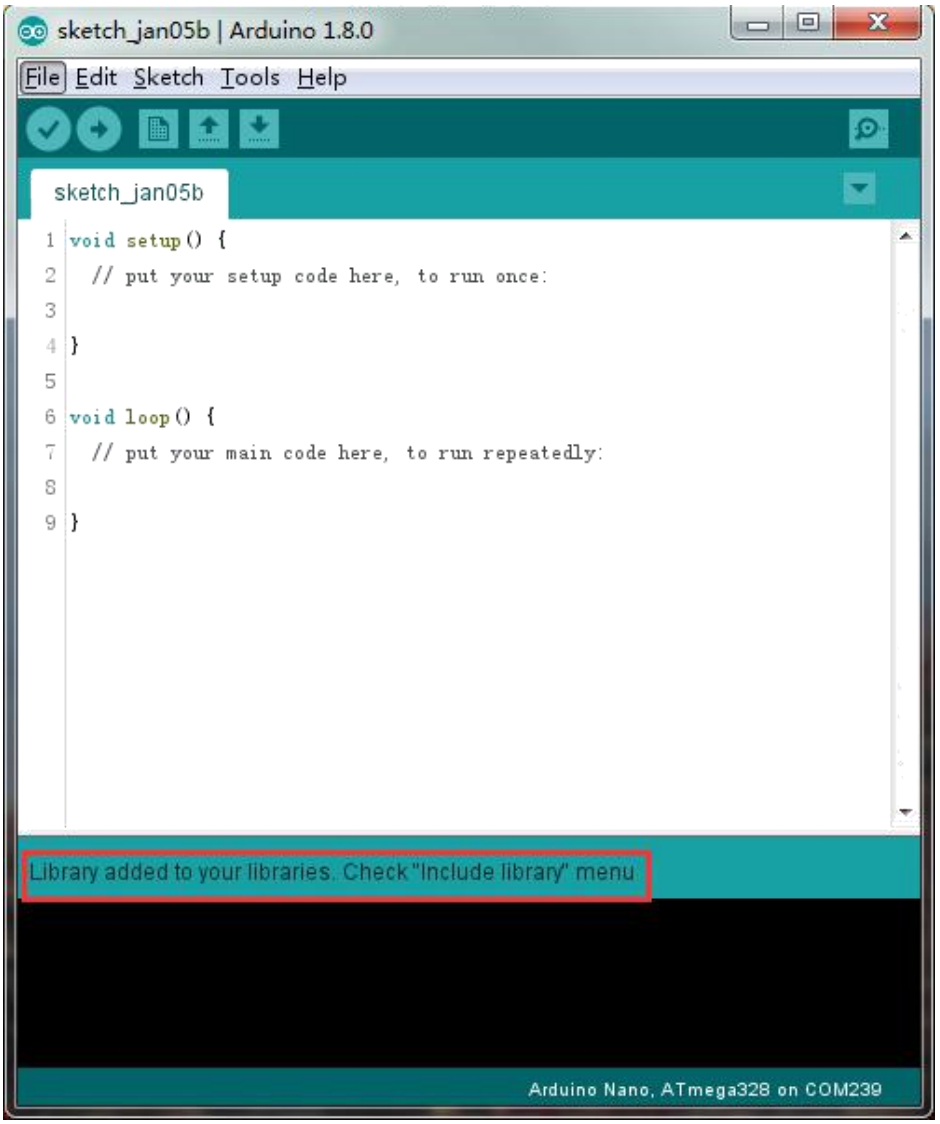
Libraries are often distributed as a ZIP file or folder. The name of the folder is the name of the library. Inside the folder will be a .cpp file, a .h file and often a keywords.txt file, examples folder, and other files required by the library. Starting with version 1.0.5, you can install 3rd party 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 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.





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.

Those two are the most common approaches. MAC and Linux systems can be handled likewise. The manual installation to be introduced below as an alternative may be seldom used and users with no needs may skip it.

Manual installation

To install the library, first quit the Arduino application. Then uncompress the ZIP file containing the library. For example, if you're installing a library called

"ArduinoParty", uncompress ArduinoParty.zip. It should contain a folder calledArduinoParty, with files like ArduinoParty.cpp and ArduinoParty.h inside. (If the .cpp and .h files aren't in a folder, you'll need to create one. In this case, you'd make a folder called "ArduinoParty" and move into it all the files that were in the ZIP file, like ArduinoParty.cpp and ArduinoParty.h.)

Drag the ArduinoParty folder into this folder (your libraries folder). Under Windows, it will likely be called "My Documents\Arduino\libraries". For Mac users, it will likely be called "Documents/Arduino/libraries". On Linux, it will be the "libraries" folder in your sketchbook.

Your Arduino library folder should now look like this (on Windows): My Documents\Arduino\libraries\ArduinoParty\ArduinoParty.cpp My Documents\Arduino\libraries\ArduinoParty\ArduinoParty.h My Documents\Arduino\libraries\ArduinoParty\examples

or like this (on Mac and Linux):

Documents/Arduino/libraries/ArduinoParty/ArduinoParty.cpp
Documents/Arduino/libraries/ArduinoParty/ArduinoParty.h
Documents/Arduino/libraries/ArduinoParty/examples
....

There may be more files than just the .cpp and .h files, just make sure they're all there. (The library won't work if you put the .cpp and .h files directly into the libraries folder or if they're nested in an extra folder. For example: Documents\Arduino\libraries\ArduinoParty.cpp and Documents\Arduino\libraries\ArduinoParty\ArduinoParty\ArduinoParty.cpp won't work.)

Restart the Arduino application. Make sure the new library appears in the Sketch->Import Library menu item of the software. That's it! You've installed a library!

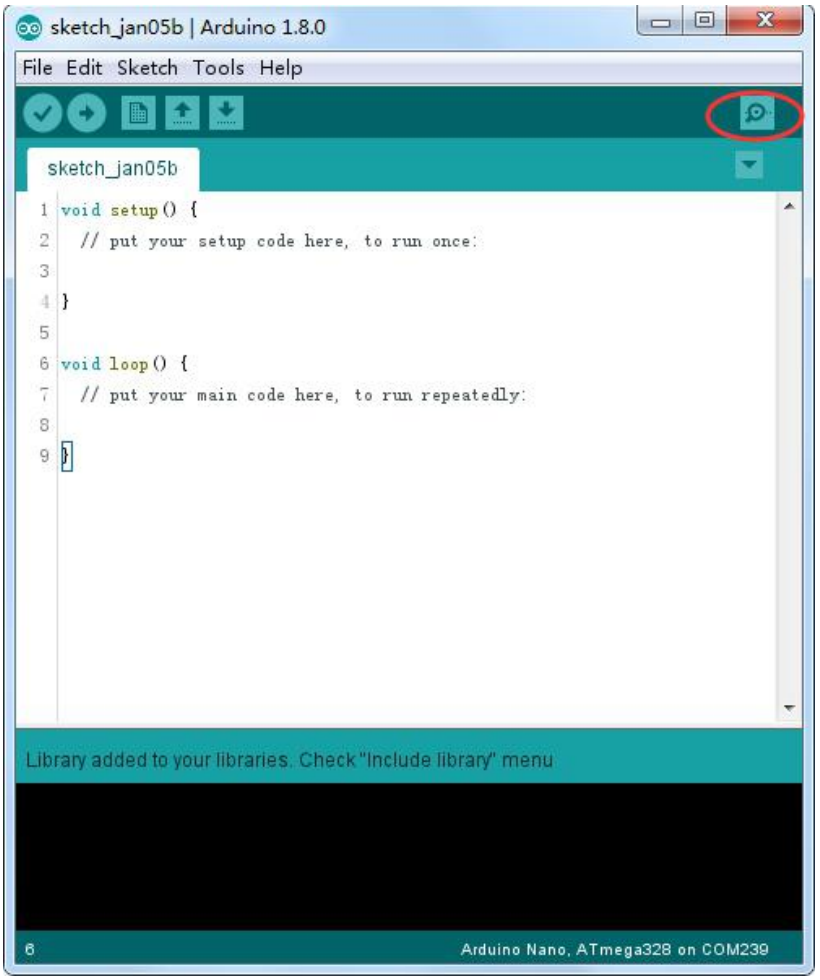
Arduino Serial Monitor (Windows, Mac, Linux)

The Arduino Integrated Development Environment (IDE) is the software side of the Arduino platform. And, because using a terminal is such a big part of working with

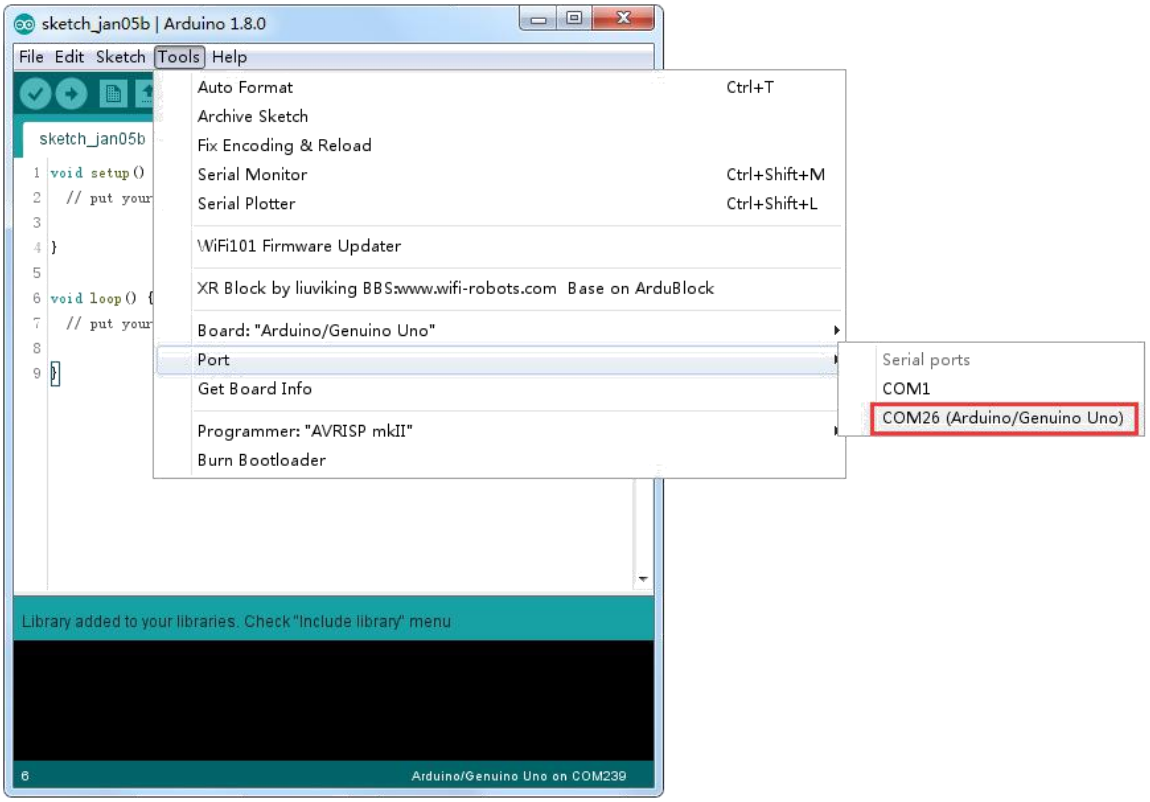
Arduinos and other microcontrollers, they decided to include a serial terminal with the software. Within the Arduino environment, this is called the Serial Monitor.

Making a Connection

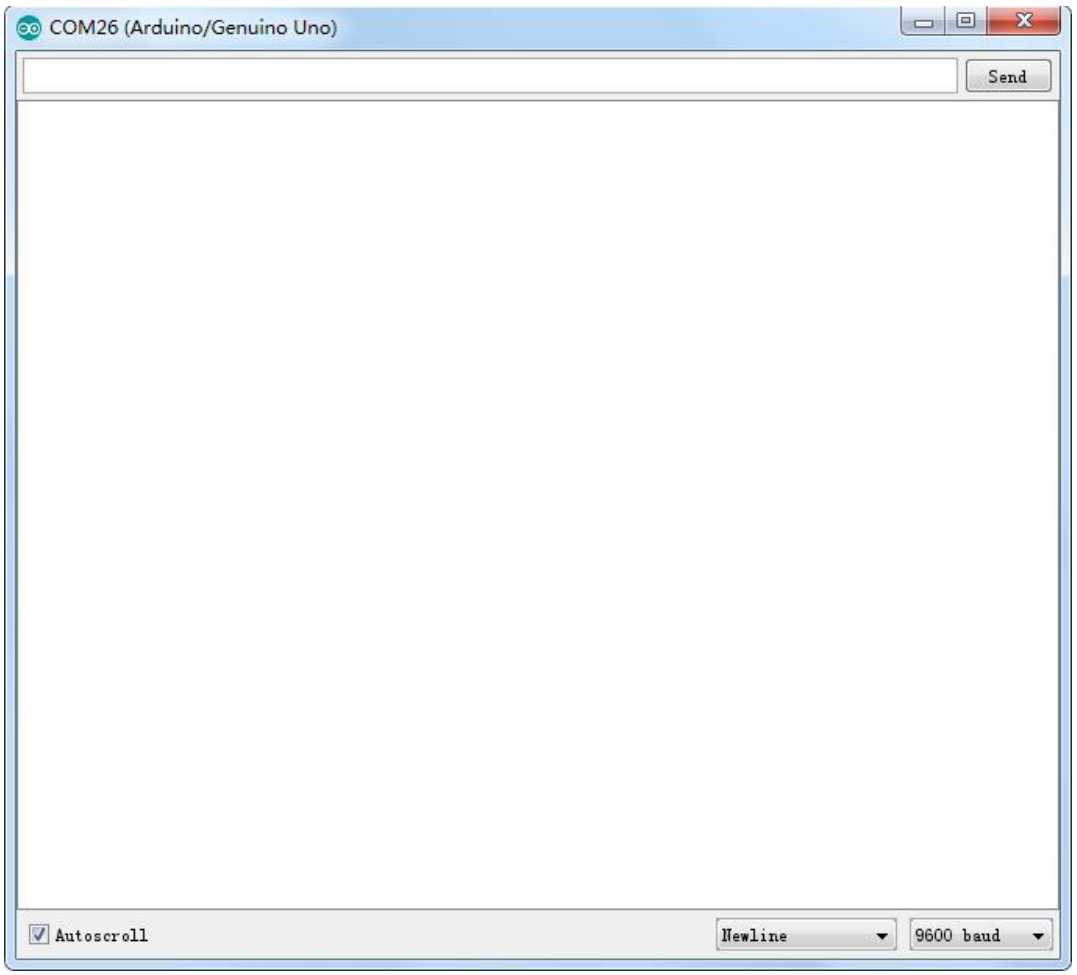
Serial monitor comes with any and all version of the Arduino IDE. To open it, simply click the Serial Monitor icon.



Selecting which port to open in the Serial Monitor is the same as selecting a port for uploading Arduino code. Go to Tools -> Serial Port, and select the correct port. Tips: Choose the same COM port that you have in Device Manager.

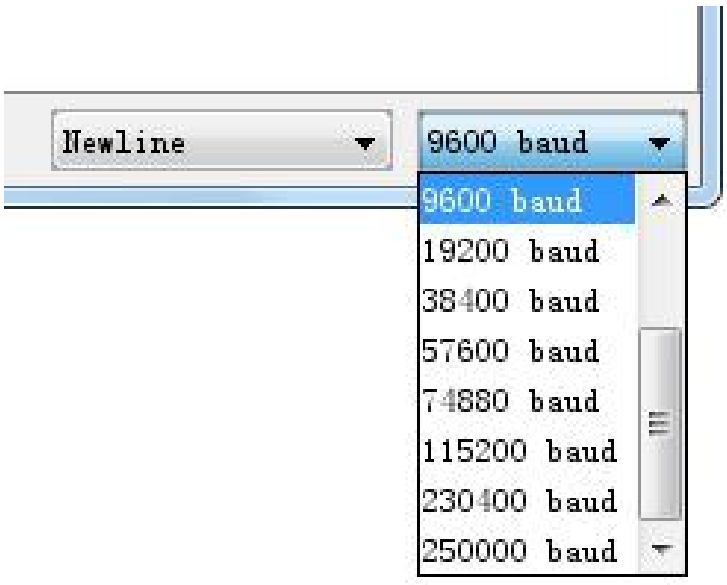


Once open, you should see something like this:



Settings

The Serial Monitor has limited settings, but enough to handle most of your serial communication needs. The first setting you can alter is the baud rate. Click on the baud rate drop-down menu to select the correct baud rate. (9600 baud)



Last, you can set the terminal to Autoscroll or not by checking the box in the bottom left corner.



Pros

The Serial Monitor is a great quick and easy way to establish a serial connection with your Arduino. If you're already working in the Arduino IDE, there's really no need to open up a separate terminal to display data.

Cons

The lack of settings leaves much to be desired in the Serial Monitor, and, for advanced serial communications, it may not do the trick.

Lesson 2 Blink

Overview

In this lesson, 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.

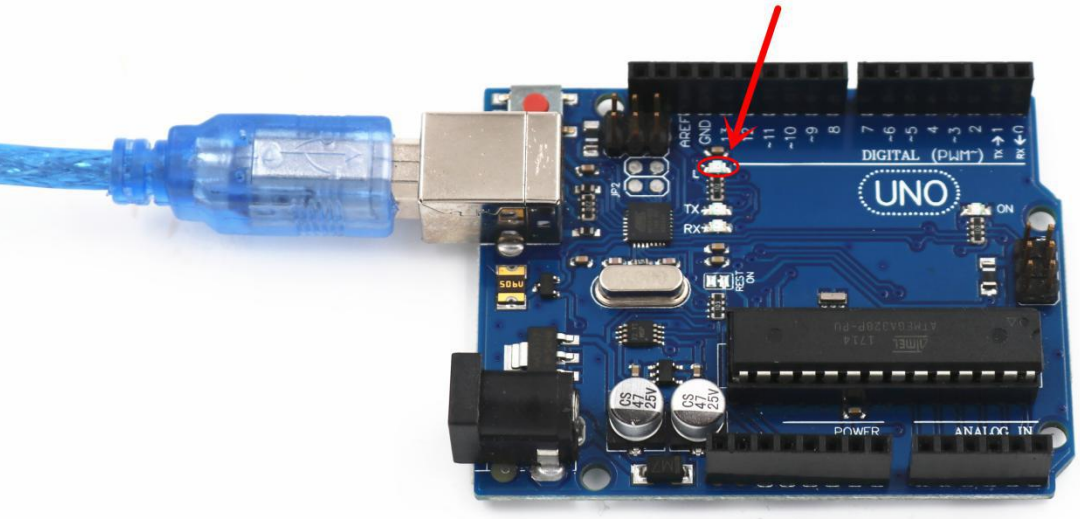
Component Required:

1 x Uno R3 Board

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.



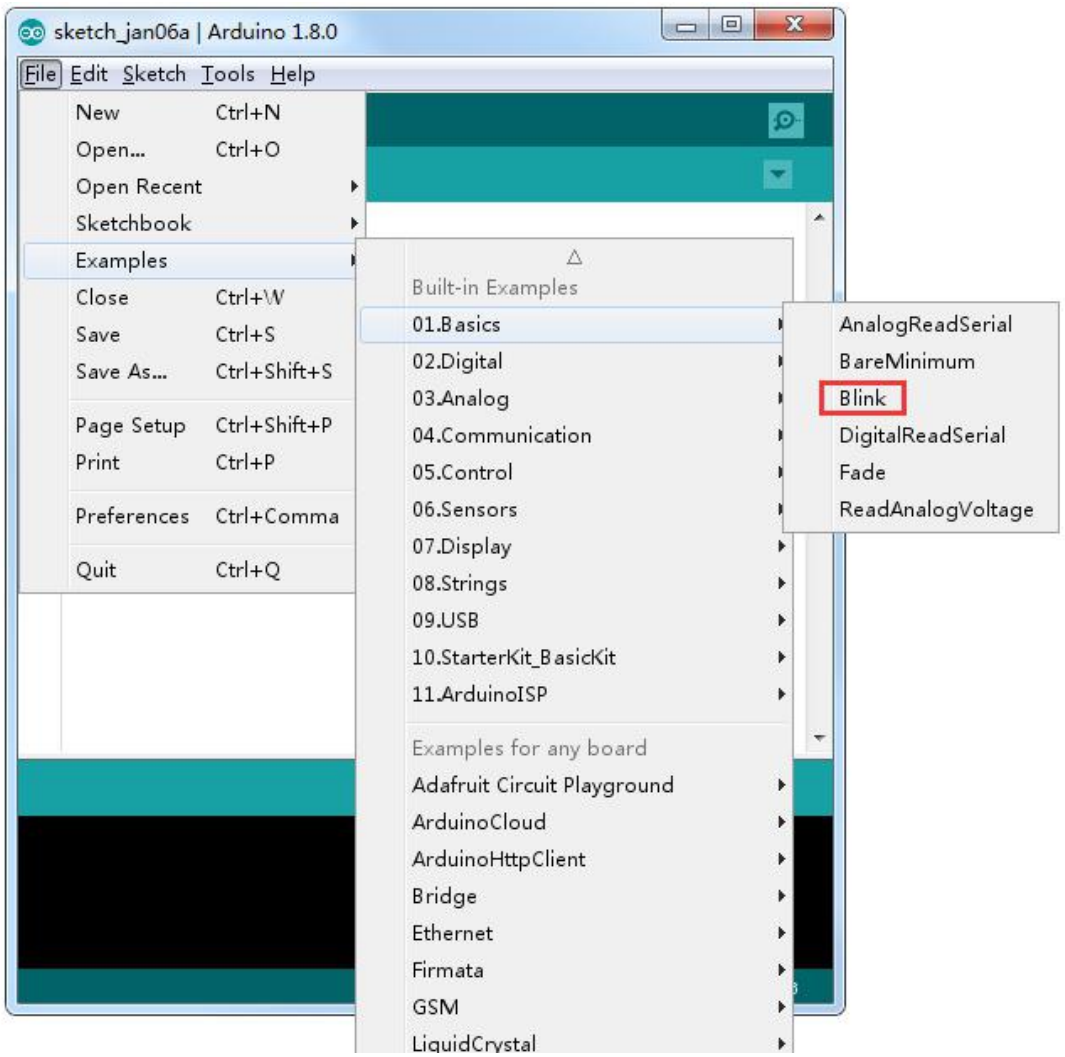
You may find that your UNO R3 board's 'L' LED already blinks when you connect it to a USB plug. This is because the boards are generally shipped with the 'Blink' sketch pre-installed.

In this lesson, we will reprogram the UNO R3 board with our own Blink sketch and then change the rate at which it blinks.

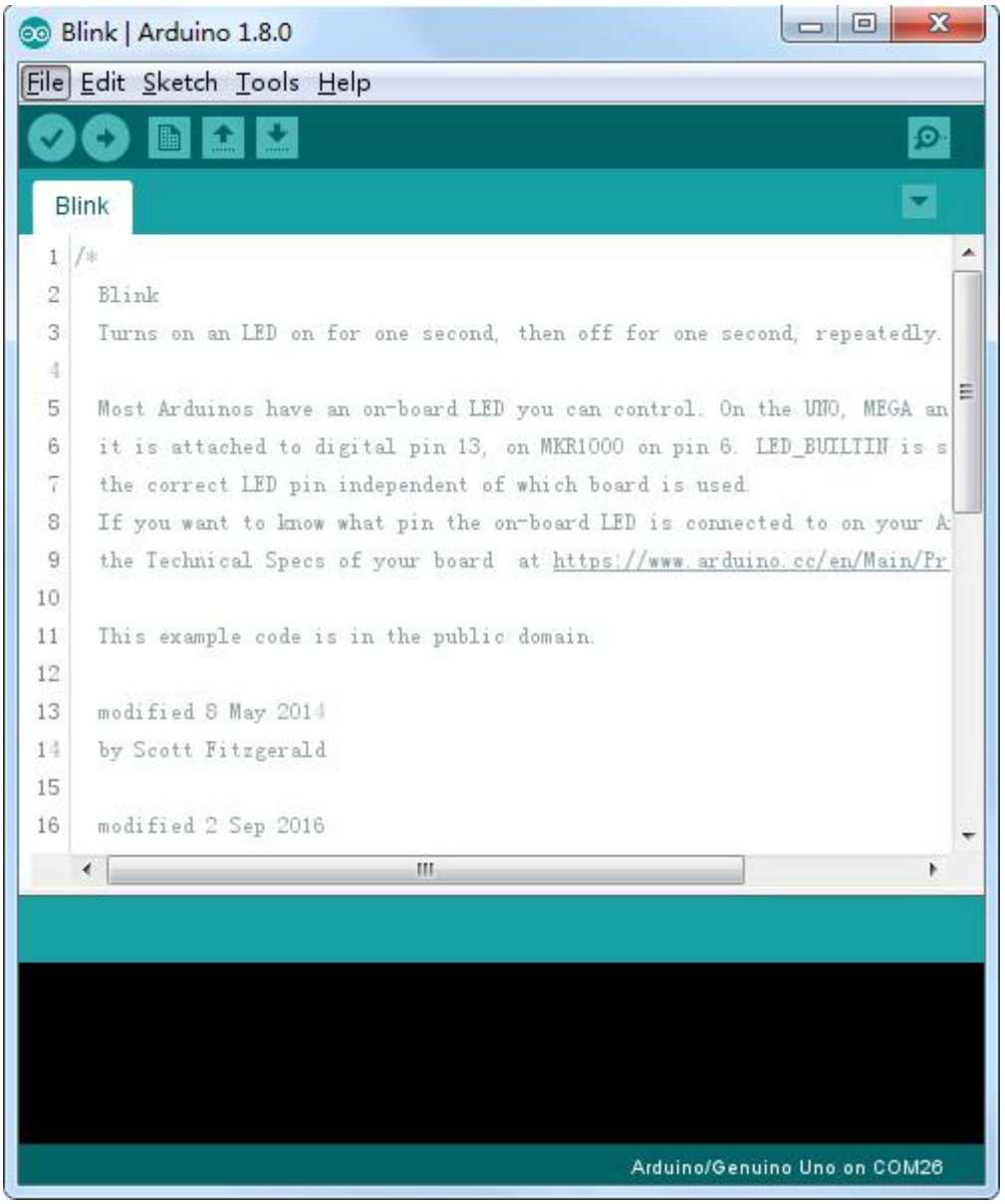
In Lesson 0, you set up your Arduino IDE and made sure that you could find the right serial port for it to connect to your UNO R3 board. The time has now come to put that connection to the test and program your UNO R3 board.

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



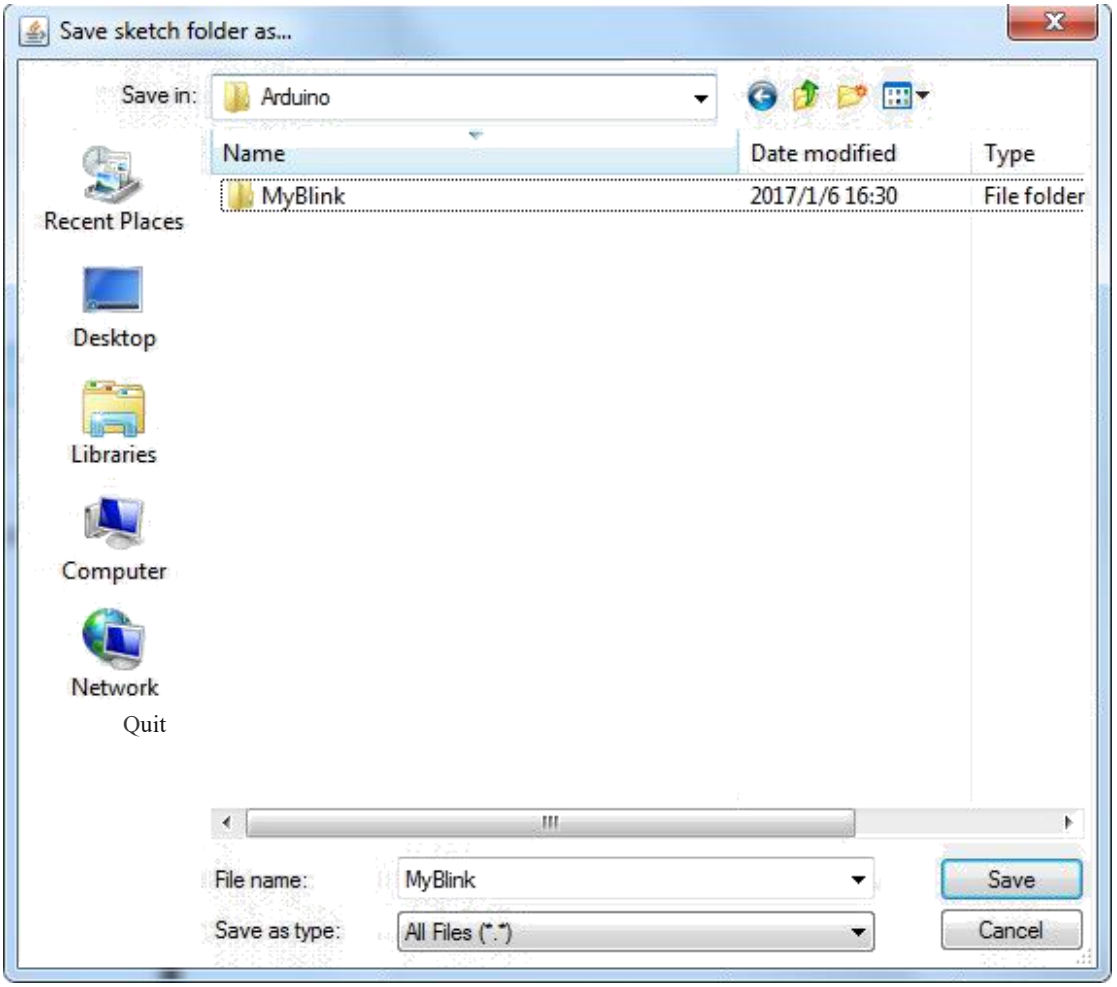
When the sketch window opens, enlarge it so that you can see the entire sketch in the window.



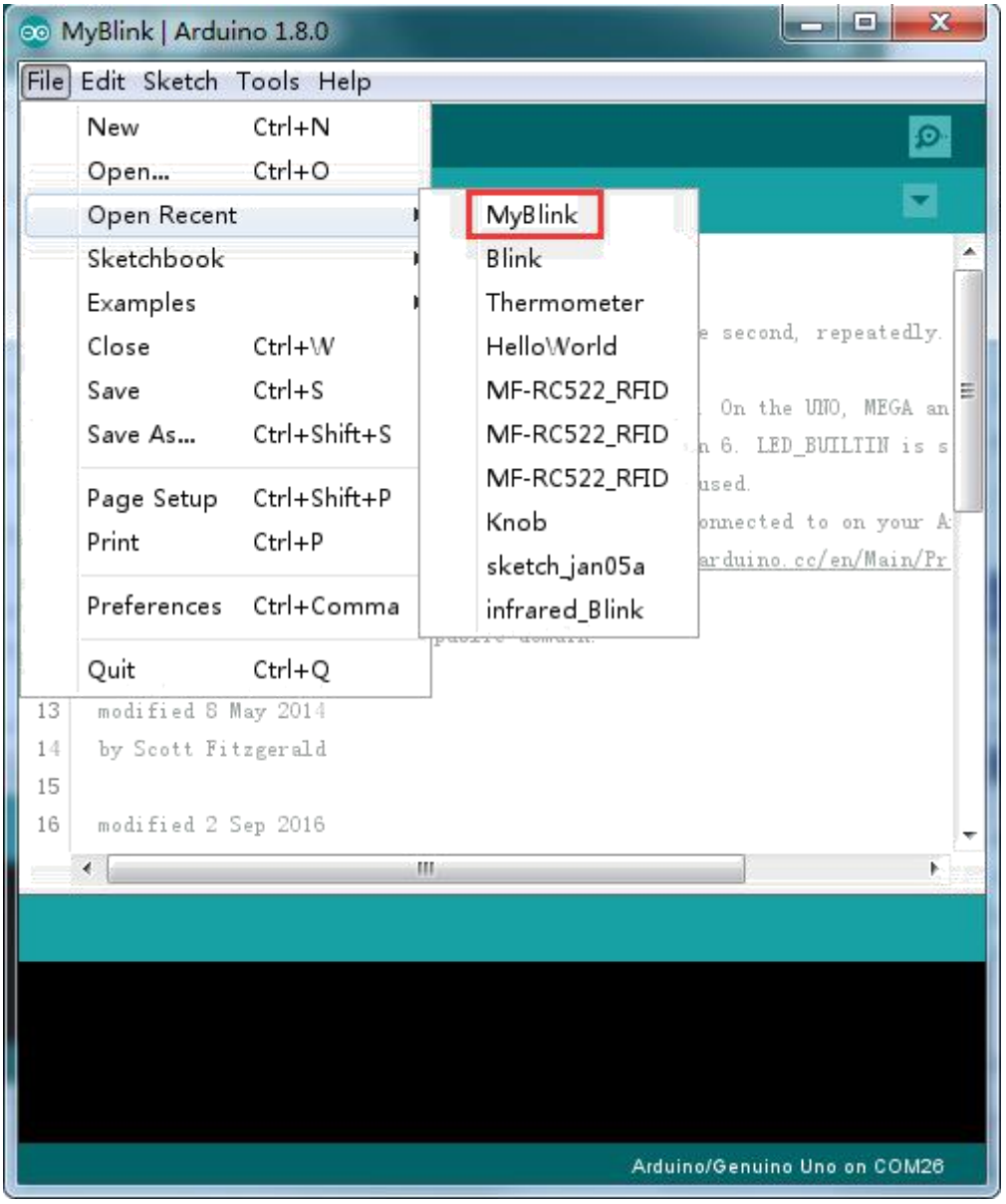
The example sketches included with the Arduino IDE are 'read-only'. That is, you can upload them to an UNO R3 board, but if you change them, you cannot save them as the same file.

Since we are going to change this sketch, the first thing you need to do is save your own copy.

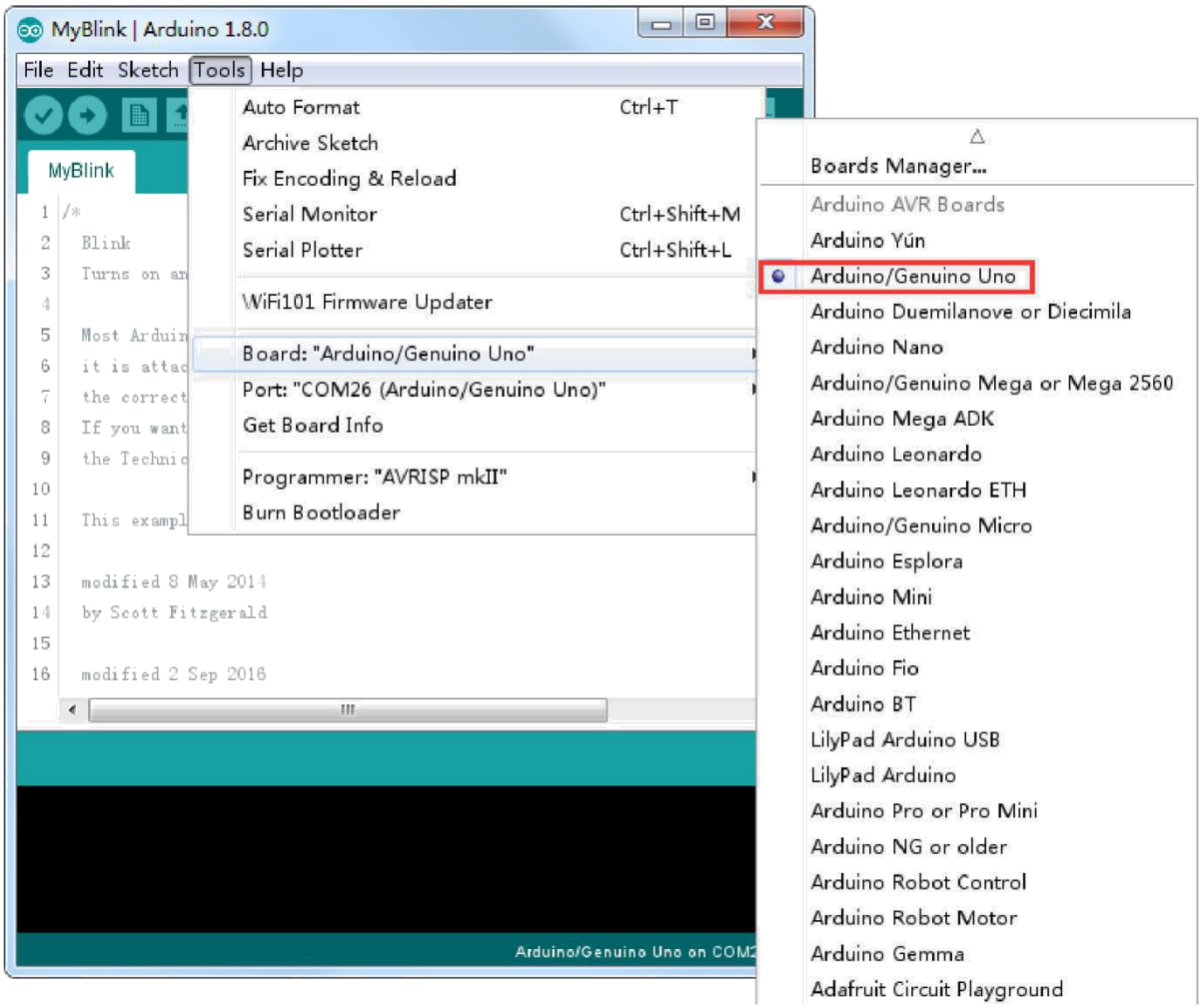
From the File menu on the Arduino IDE, select 'Save As..' and then save the sketch with the name 'MyBlink'.

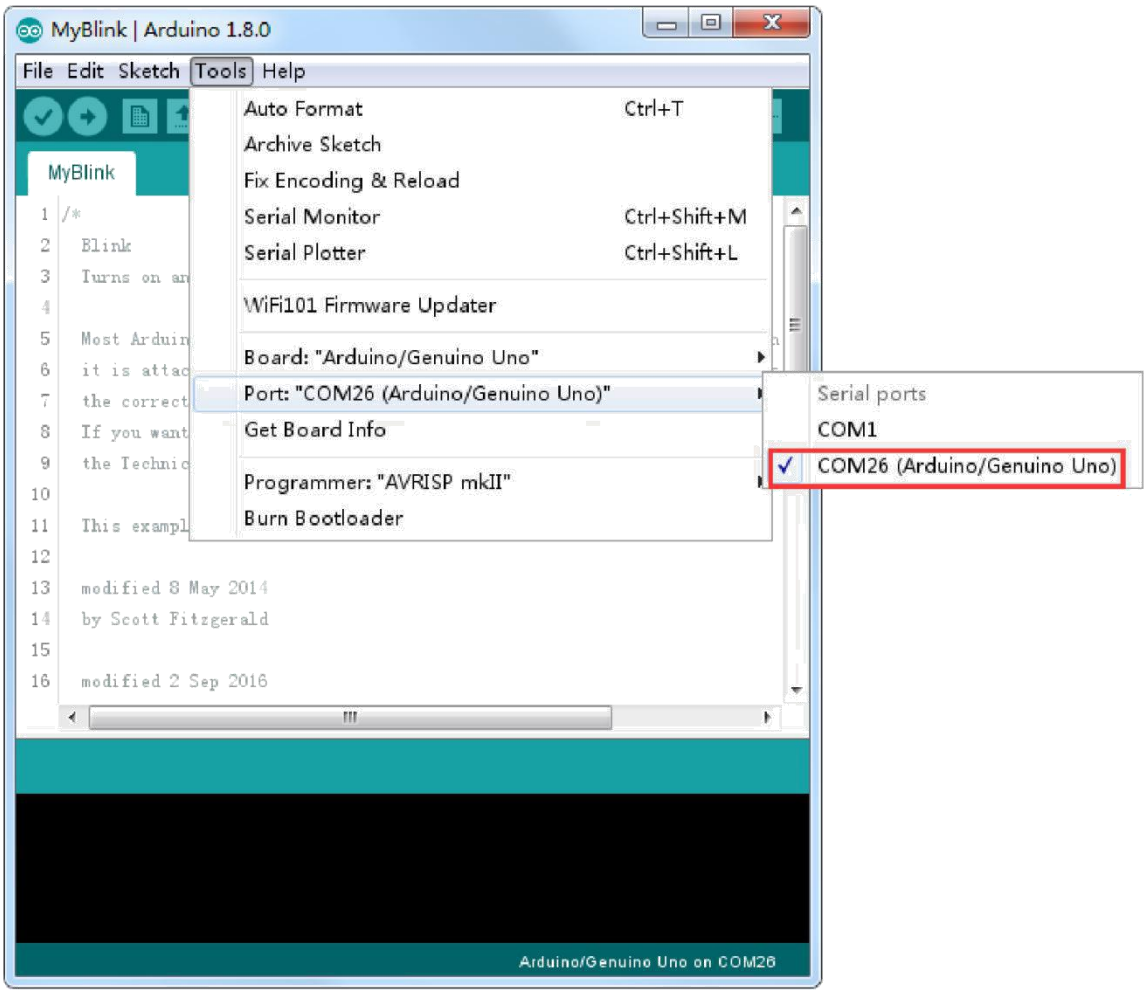


You have saved your copy of 'Blink' in your sketchbook. This means that if you ever want to find it again, you can just open it using the File > Sketchbook menu option.



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


MyBlink | Arduino 1.8.0
File Edit Sketch Tools Help
MyBlink
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "Arduino/Genuino Uno"
Port: "COM26 (Arduino/Genuino Uno)"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Boards Manager...
Arduino AVR Boards
Arduino Yún
Arduino/Genuino Uno
Arduino Duemilanove or Diecimila
Arduino Nano
Arduino/Genuino Mega or Mega 2560
Arduino Mega ADK
Arduino Leonardo
Arduino Leonardo ETH
Arduino/Genuino Micro
Arduino Esplora
Arduino Mini
Arduino Ethernet
Arduino Fio
Arduino BT
LilyPad Arduino USB
LilyPad Arduino
Arduino Pro or Pro Mini
Arduino NG or older
Arduino Robot Control
Arduino Robot Motor
Arduino Gemma
Adafruit Circuit Playground
Arduino/Genuino Uno on COM


MyBlink | Arduino 1.8.0
File Edit Sketch Tools Help
MyBlink
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "Arduino/Genuino Uno"
Port: "COM26 (Arduino/Genuino Uno)"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
COM1
COM26 (Arduino/Genuino Uno)
Arduino/Genuino Uno on COM26

Note: The Board Type and Serial Port here are not necessarily the same as shown in picture. If you are using 2560, then you will have to choose Mega 2560 as the Board Type, other choices can be made in the same manner. And the Serial Port displayed for everyone is different, despite COM 26 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.



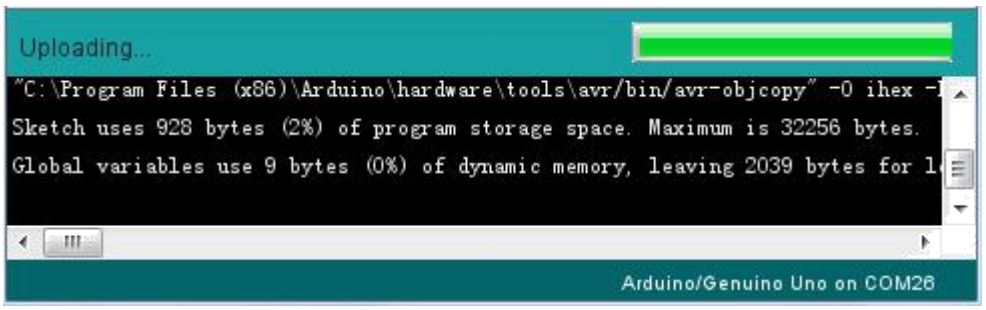
Click on the 'Upload' button. The second button from the left on the toolbar.



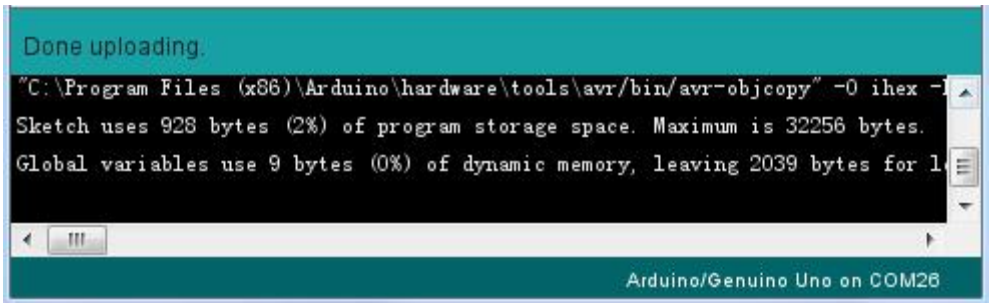
If you watch the status area of the IDE, you will see a progress bar and a series of messages. At first, it will say 'Compiling Sketch...'. This converts the sketch into a format suitable for uploading to the board.



Next, the status will change to 'Uploading'. At this point, the LEDs on the Arduino should start to flicker as the sketch is transferred.



Finally, the staus will change to 'Done'.



The other message tells us that the sketch is using 928 bytes of the 32,256 bytes available.After the 'Compiling Sketch..' stage you could get the following error message:



It can mean that your board is not connected at all, or the drivers have not been installed (if necessary) or that the wrong serial port is selected.

If you encounter this, go back to Lesson 0 and check your installation.

Once the upload has completed, the board should restart and start blinking.
Open the code

Note that a huge part of this sketch is composed of comments. These are not actual program instructions; rather, they just explain how the program works. They are there for your benefit.

Everything between /* and */ at the top of the sketch is a block comment; it explains what the sketch is for.

Single line comments start with // and everything up until the end of that line is considered a comment.

The first line of code is:

```
int led = 13;
```

As the comment above it explains, this is giving a name to the pin that the LED is attached to. This is 13 on most Arduinos, including the UNO and Leonardo.

Next, we have the 'setup' function. Again, as the comment says, this is executed when the reset button is pressed. It is also executed whenever the board resets for any reason, such as power first being applied to it, or after a sketch has been uploaded.

```
void setup() {
// initialize the digital pin as an output.
pinMode(led, OUTPUT);
}
```

Every Arduino sketch must have a 'setup' function, and the place where you might want to add instructions of your own is between the { and the }.

In this case, there is just one command there, which, as the comment states tells the Arduino board that we are going to use the LED pin as an output.

It is also mandatory for a sketch to have a 'loop' function. Unlike the 'setup' function that only runs once, after a reset, the 'loop' function will, after it has finished running its commands, immediately start again.

```
void loop()
  { digitalWrite(led, HIGH)
  delay(1000);                  // turn the LED on (HIGH is the voltage
  digitalWrite(led, LOW);       level) // wait for a second
  delay(1000);
                                 // turn the LED off by making the voltage
}                                LOW // wait for a second
```

Inside the loop function, the commands first of all turn the LED pin on (HIGH), then 'delay' for 1000 milliseconds (1 second), then turn the LED pin off and pause for another second.

You are now going to make your LED blink faster. As you might have guessed, the key to this lies in changing the parameter in () for the 'delay' command.

```
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 volt
33   delay(500)                         // wait for a second
34   digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the vo
35   delay(500)                         // wait for a second
36 }
```

This delay period is in milliseconds, so if you want the LED to blink twice as fast, change the value from 1000 to 500. This would then pause for half a second each delay rather than a whole second.

Upload the sketch again and you should see the LED start to blink more quickly.

Lesson 3 LED

Overview

In this lesson, you will learn how to change the brightness of an LED by using different values of resistor.

Component Required:

(1) x Uno R3 Board

(1) x 5mm red LED

(1) x 220 ohm resistor

(1) x 1k ohm resistor

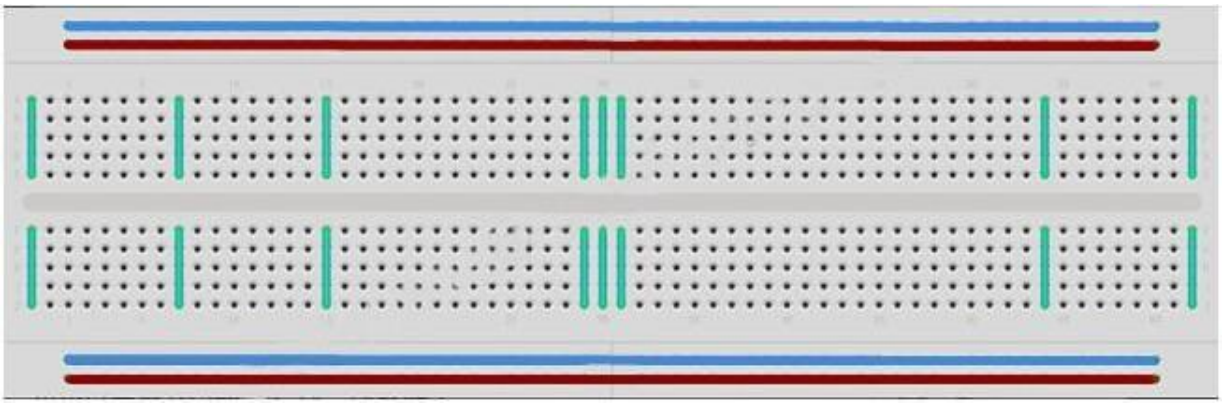
(1) x 10k ohm resistor

(2) x M-M wires (Male to Male jumper wires)

Component Introduction

BREADBOARD MB-102:

A breadboard enables you to prototype circuits quickly, without having to solder the connections. Below is an example.



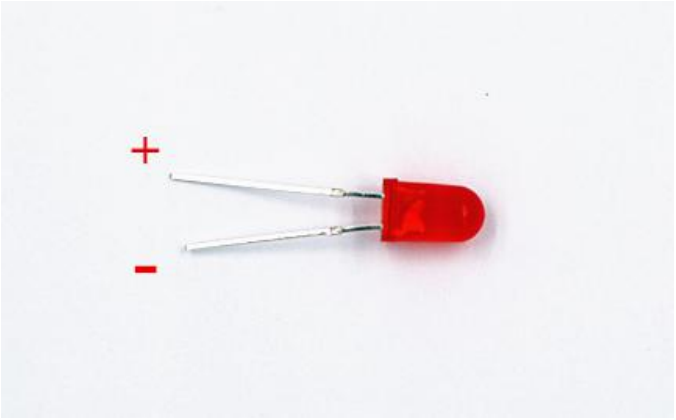
Breadboards come in various sizes and configurations. The simplest kind is just a grid of holes in a plastic block. Inside are strips of metal that provide electrical connection between holes in the shorter rows. Pushing the legs of two different components into the same row joins them together electrically. A deep channel running down the middle indicates that there is a break in connections there, meaning, you can push a chip in with the legs at either side of the channel without connecting them together. Some breadboards have two strips of holes running along the long edges of the board that are separated from the main grid. These have strips running down the length of the board inside and provide a way to connect a common voltage. They are usually in pairs for +5 volts and ground. These strips are referred to as rails and they enable you to connect power to many components or points in the board.

While breadboards are great for prototyping, they have some limitations. Because the connections are push-fit and temporary, they are not as reliable as soldered connections. If you are having intermittent problems with a circuit, it could be due to a poor connection on a breadboard.

LED:

LEDs make great indicator lights. They use very little electricity and they pretty much last forever.

In this lesson, you will use perhaps the most common of all LEDs: a 5mm red LED. 5mm refers to the diameter of the LED. Other common sizes are 3mm and 10mm. You cannot directly connect an LED to a battery or voltage source because 1) the LED has a positive and a negative lead and will not light if placed the wrong way and 2) an LED must be used with a resistor to limit or 'choke' the amount of current flowing through it; otherwise, it will burn out!



If you do not use a resistor with an LED, then it may well be destroyed almost immediately, as too much current will flow through, heating it and destroying the 'junction' where the light is produced.

There are two ways to tell which is the positive lead of the LED and which the negative.

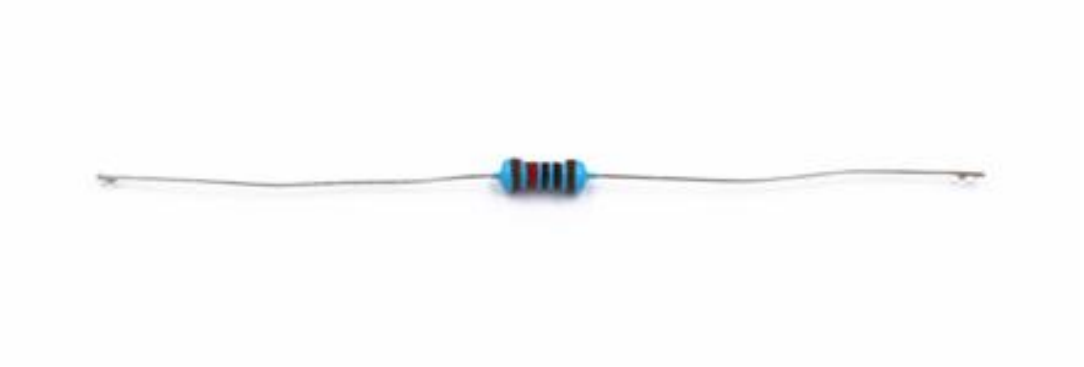
Firstly, the positive lead is longer.

Secondly, where the negative lead enters the body of the LED, there is a flat edge to the case of the LED.

If you happen to have an LED that has a flat side next to the longer lead, you should assume that the longer lead is positive.

RESISTORS:

As the name suggests, resistors resist the flow of electricity. The higher the value of the resistor, the more it resists and the less electrical current will flow through it. We are going to use this to control how much electricity flows through the LED and therefore, how brightly it shines.



But first, more about resistors...

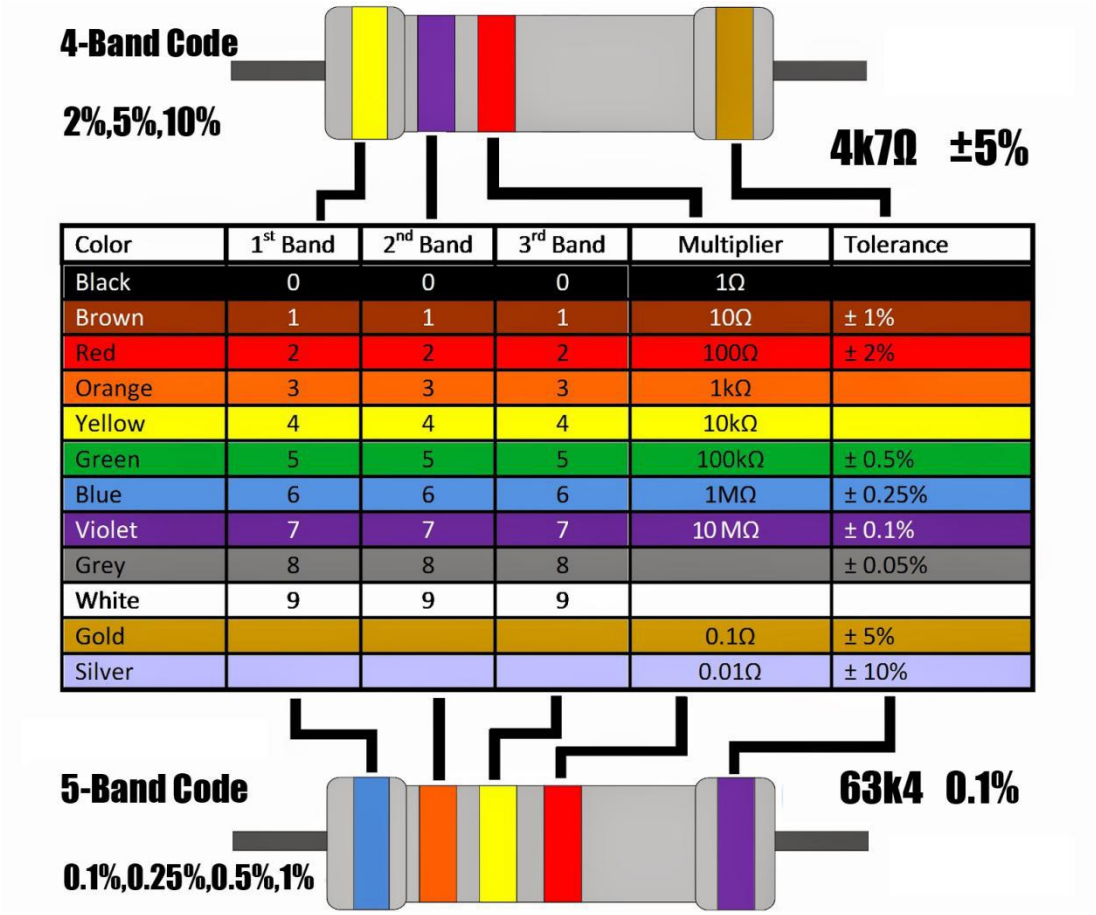
The unit of resistance is called the Ohm, which is usually shortened to Ω the Greek letter Omega. Because an Ohm is a low value of resistance (it doesn't resist much at all), we also denote the values of resistors in kΩ (1,000 Ω) and MΩ (1,000,000 Ω).

These are called kilo-ohms and mega-ohms.

In this lesson, we are going to use three different values of resistor: 220Ω, 1kΩ and

10kΩ. These resistors all look the same, except that they have different colored stripes on them. These stripes tell you the value of the resistor.

The resistor color code has three colored stripes and then a gold stripe at one end.



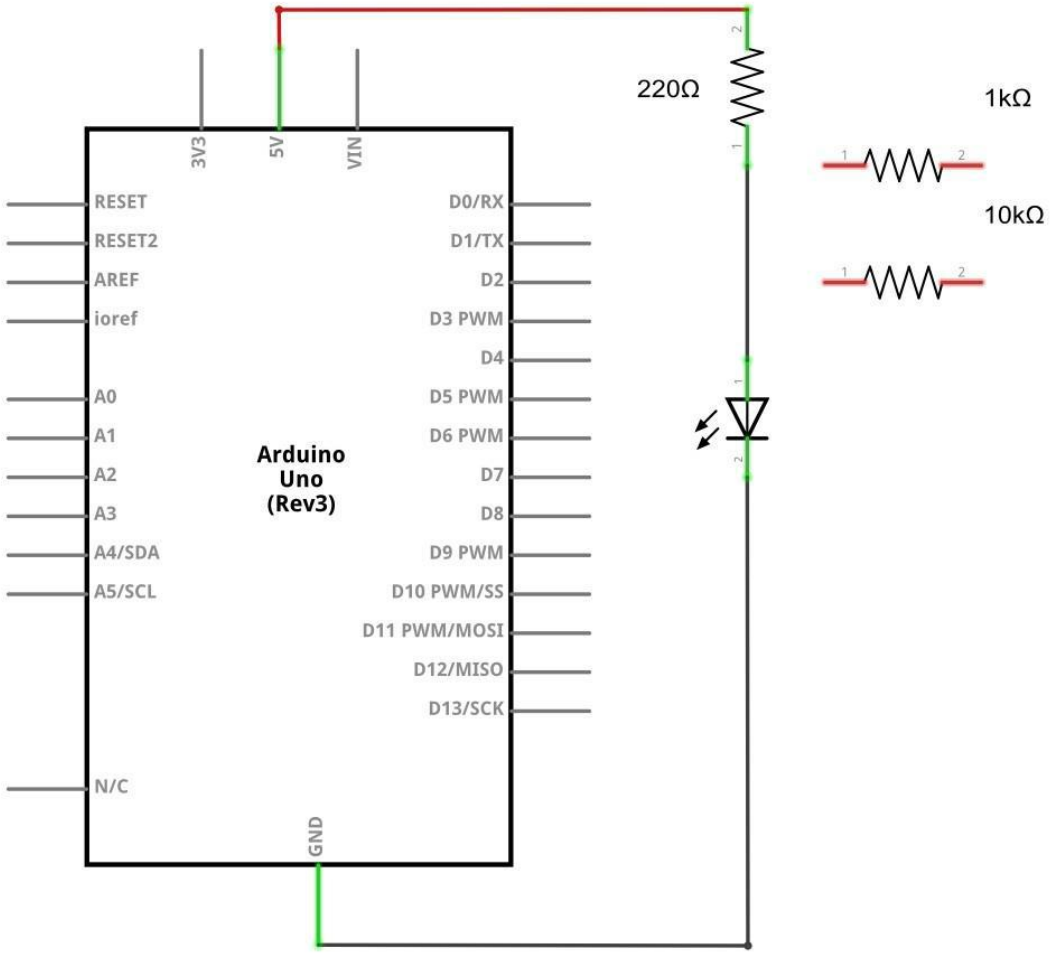
Color	1st Band	2nd Band	3rd Band	Multiplier	Tolerance
Black	0	0	0	1Ω	
Brown	1	1	1	10Ω	± 1%
Red	2	2	2	100Ω	± 2%
Orange	3	3	3	1kΩ	
Yellow	4	4	4	10kΩ	
Green	5	5	5	100kΩ	± 0.5%
Blue	6	6	6	1MΩ	± 0.25%
Violet	7	7	7	10 MΩ	± 0.1%
Grey	8	8	8		± 0.05%
White	9	9	9		
Gold				0.1Ω	± 5%
Silver				0.01Ω	± 10%

Unlike LEDs, resistors do not have a positive and negative lead. They can be connected either way around.

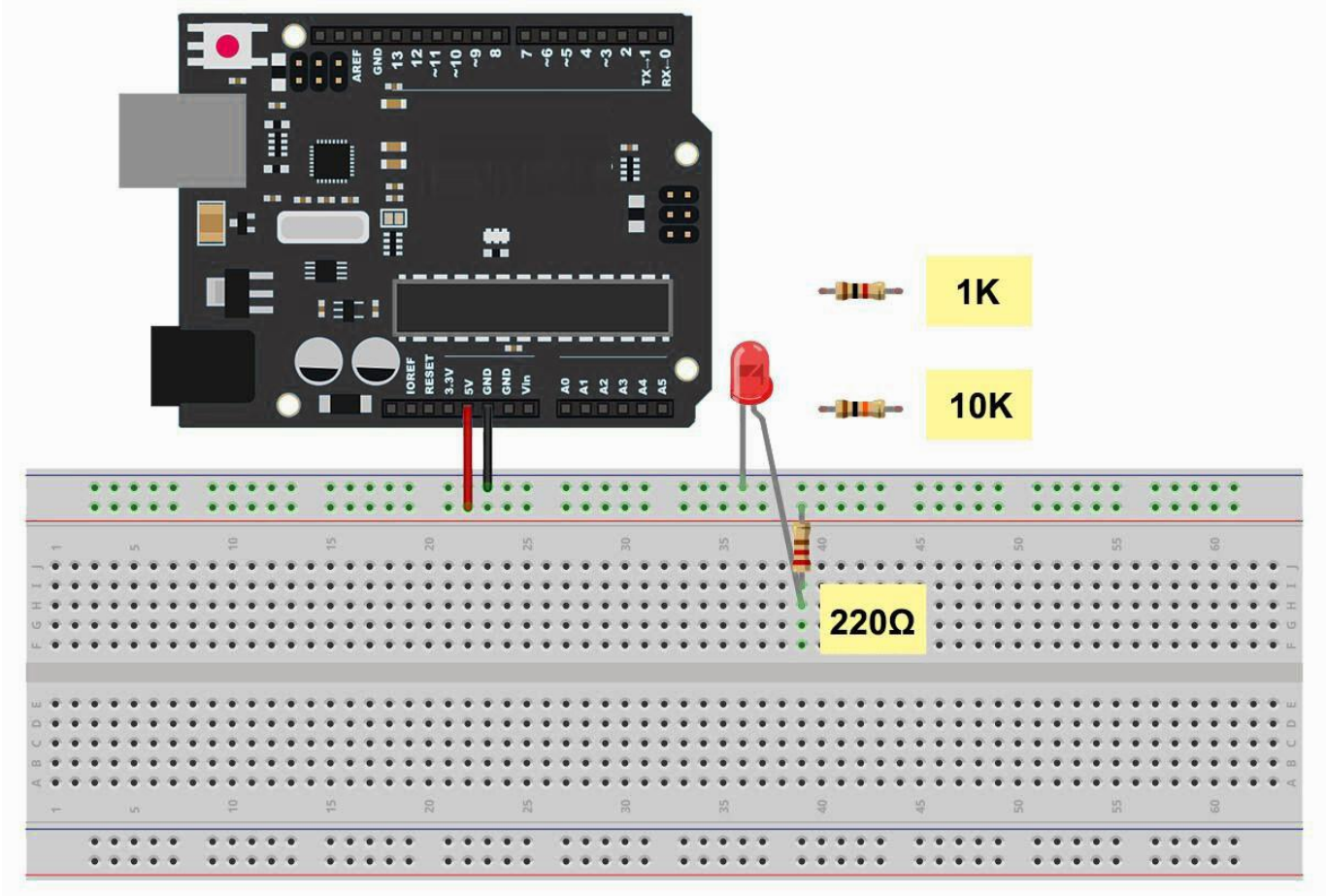
If you find this approach method too complicated, you can read the color ring flag on our resistors directly to determine its resistance value. Or you may use a digital multimeter instead.

Connection

Schematic



Wiring diagram



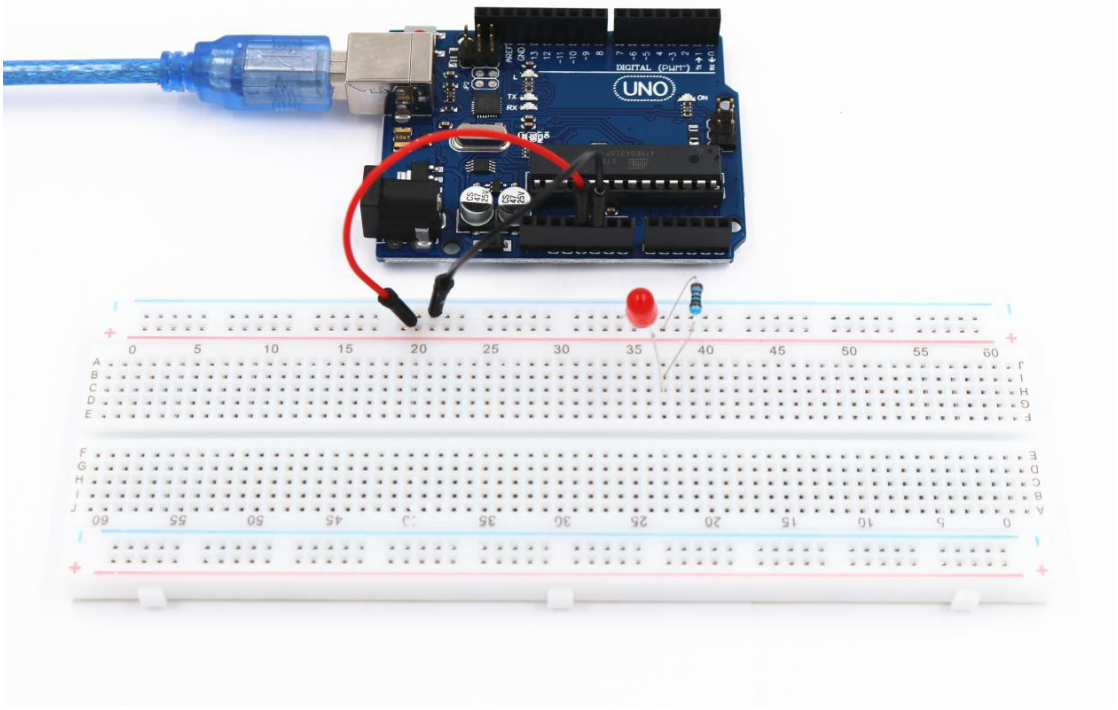
The UNO is a convenient source of 5 volts, which we will use to provide power to the LED and the resistor. You do not need to do anything with your UNO, except to plug it into a USB cable.

With the 220 Ω resistor in place, the LED should be quite bright. If you swap out the 220 Ω resistor for the 1kΩ resistor, then the LED will appear a little dimmer. Finally, with the 10 kΩ resistor in place, the LED will be just about visible. Pull the red jumper lead out of the breadboard and touch it into the hole and remove it, so that it acts like a switch. You should just be able to notice the difference.

At the moment, you have 5V going to one leg of the resistor, the other leg of the resistor going to the positive side of the LED and the other side of the LED going to GND. However, if we moved the resistor so that it came after the LED, as shown below, the LED will still light.
You will probably want to put the 220Ω resistor back in place.

It does not matter which side of the LED we put the resistor, as long as it is there somewhere

Example picture



Lesson 4 RGB LED

Overview

RGB LEDs are a fun and easy way to add some color to your projects. Since they are like 3 regular LEDs in one, how to use and connect them is not much different. They come mostly in 2 versions: Common Anode or Common Cathode.

Common Anode uses 5V on the common pin, while Common Cathode connects to ground.

As with any LED, we need to connect some resistors inline (3 total) so we can limit the current being drawn.

In our sketch, we will start with the LED in the Red color state, then fade to Green, then fade to Blue and finally back to the Red color. By doing this we will cycle through most of the color that can be achieved.

Component Required:

(1) x Uno R3 Board

(1) x 830 Tie Points Breadboard

(4) x M-M wires (Male to Male jumper wires)

(1) x RGB LED

(3) x 220 ohm resistors

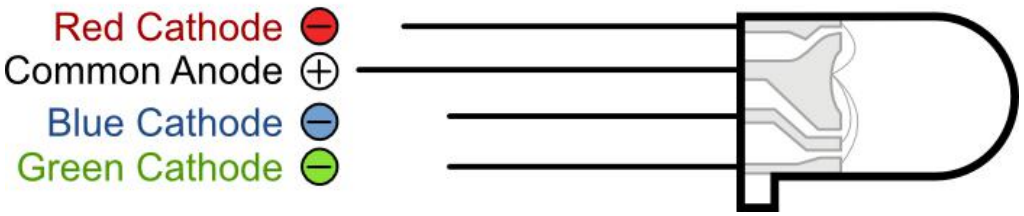
Component Introduction

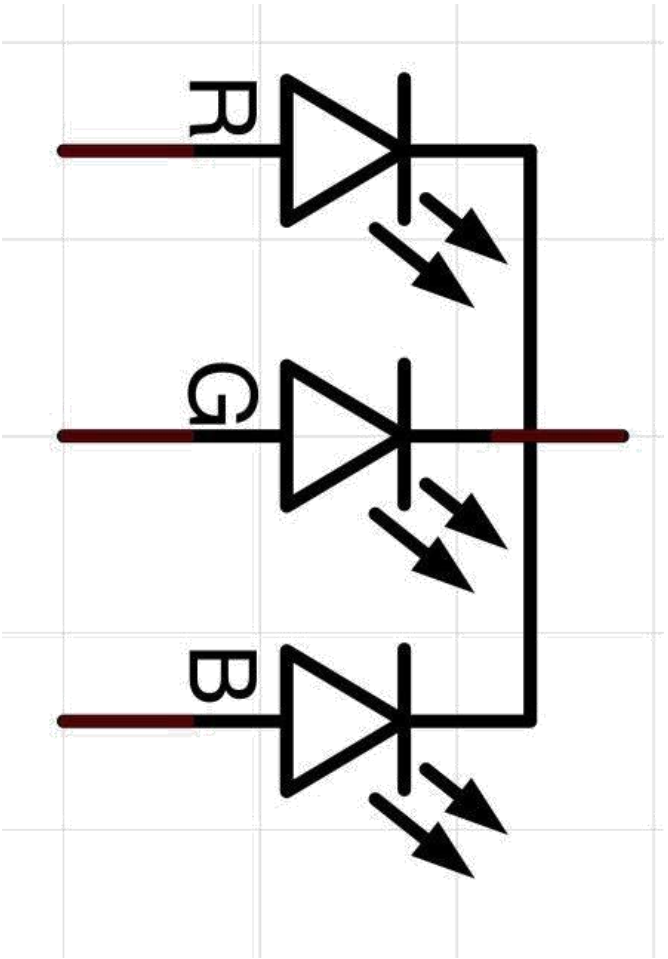
RGB:

At first glance, RGB (Red, Green and Blue) LEDs look just like regular LEDs. However, inside the usual LED package, there are actually three LEDs, one red, one green and yes, one blue. By controlling the brightness of each of the individual LEDs you can mix pretty much any color you want.

We mix colors the same way you would mix paint on a palette - by adjusting the brightness of each of the three LEDs. The hard way to do this would be to use different value resistors (or variable resistors) as we did with in Lesson 2, but that's a lot of work! Fortunately for us, UNO R3 board has an analogWrite function that you can use with pins marked with a ~ to output a variable amount of power to the appropriate LEDs.

The RGB LED has four leads. There is one lead going to the positive connection of each of the single LEDs within the package and a single lead that is connected to all three negative sides of the LEDs.





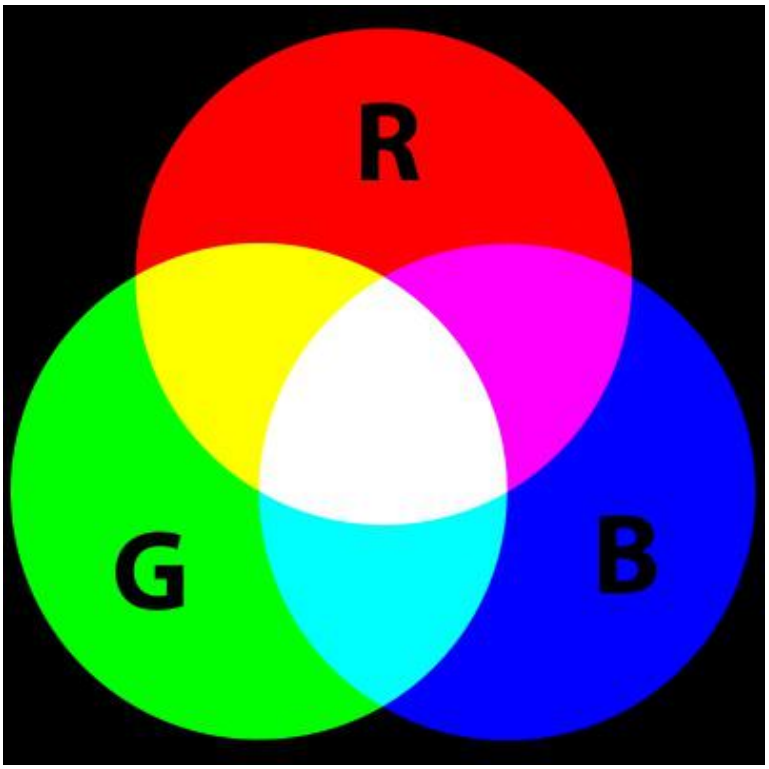
Here on the photographs you can see 4 electrode LED. Every separate pin for Green or Blue or Red color is called Anode. You will always connect “+” to it. Cathode goes to “-“(ground). If you connect it other way round the LED will not light.

The common negative connection of the LED package is the second pin from the flat side. It is also the longest of the four leads and will be connected to the ground. Each LED inside the package requires its own 220Ω resistor to prevent too much current flowing through it. The three positive leads of the LEDs (one red, one green and one blue) are connected to UNO output pins using these resistors.

COLOR:

The reason that you can mix any color you like by varying the quantities of red, green and blue light is that your eye has three types of light receptor in it (red, green and blue). Your eye and brain process the amounts of red, green and blue and convert it into a color of the spectrum.

In a way, by using the three LEDs, we are playing a trick on the eye. This same idea is used in TVs, where the LCD has red, green and blue color dots next to each other making up each pixel.



If we set the brightness of all three LEDs to be the same, then the overall color of the light will be white. If we turn off the blue LED, so that just the red and green LEDs are the same brightness, then the light will appear yellow.

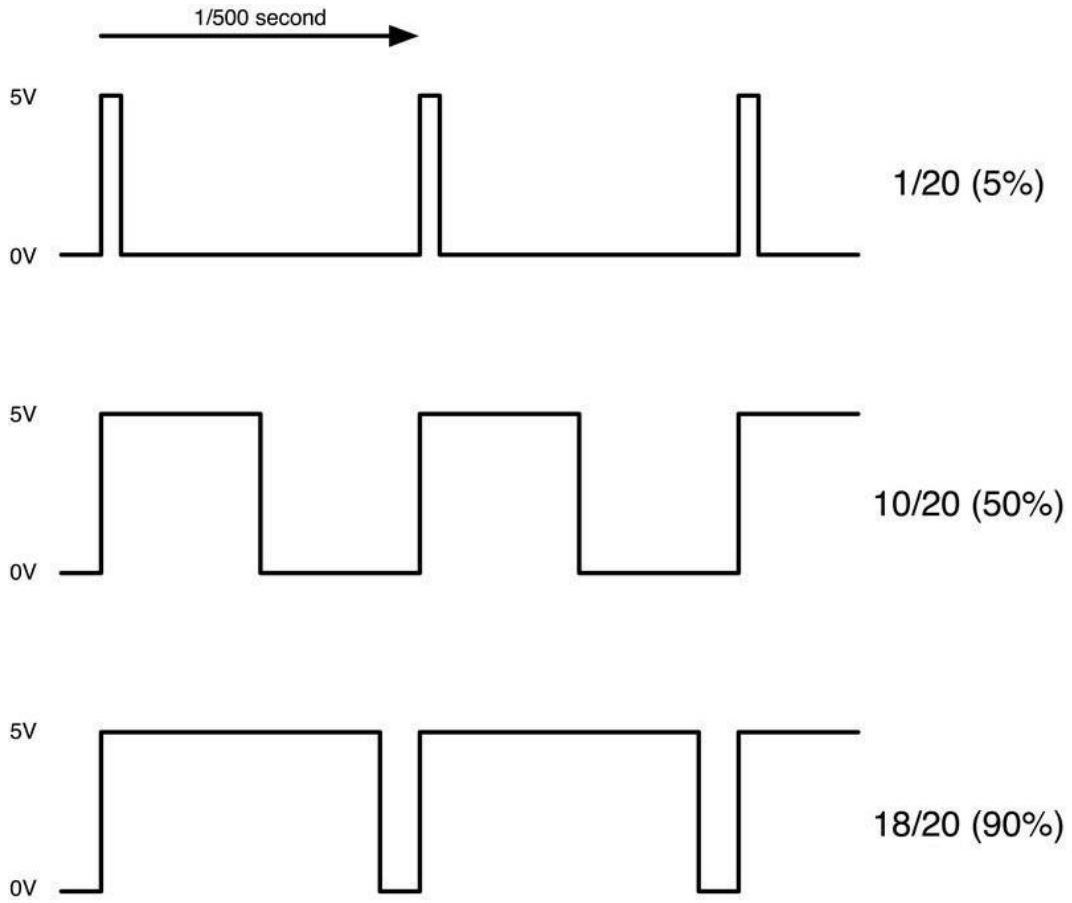
We can control the brightness of each of the red, green and blue parts of the LED separately, making it possible to mix any color we like.

Black is not so much a color as an absence of light. Therefore, the closest we can come to black with our LED is to turn off all three colors.

Theory (PWM)

Pulse Width Modulation (PWM) is a technique for controlling power. We also use it here to control the brightness of each of the LEDs.

The diagram below shows the signal from one of the PWM pins on the UNO.



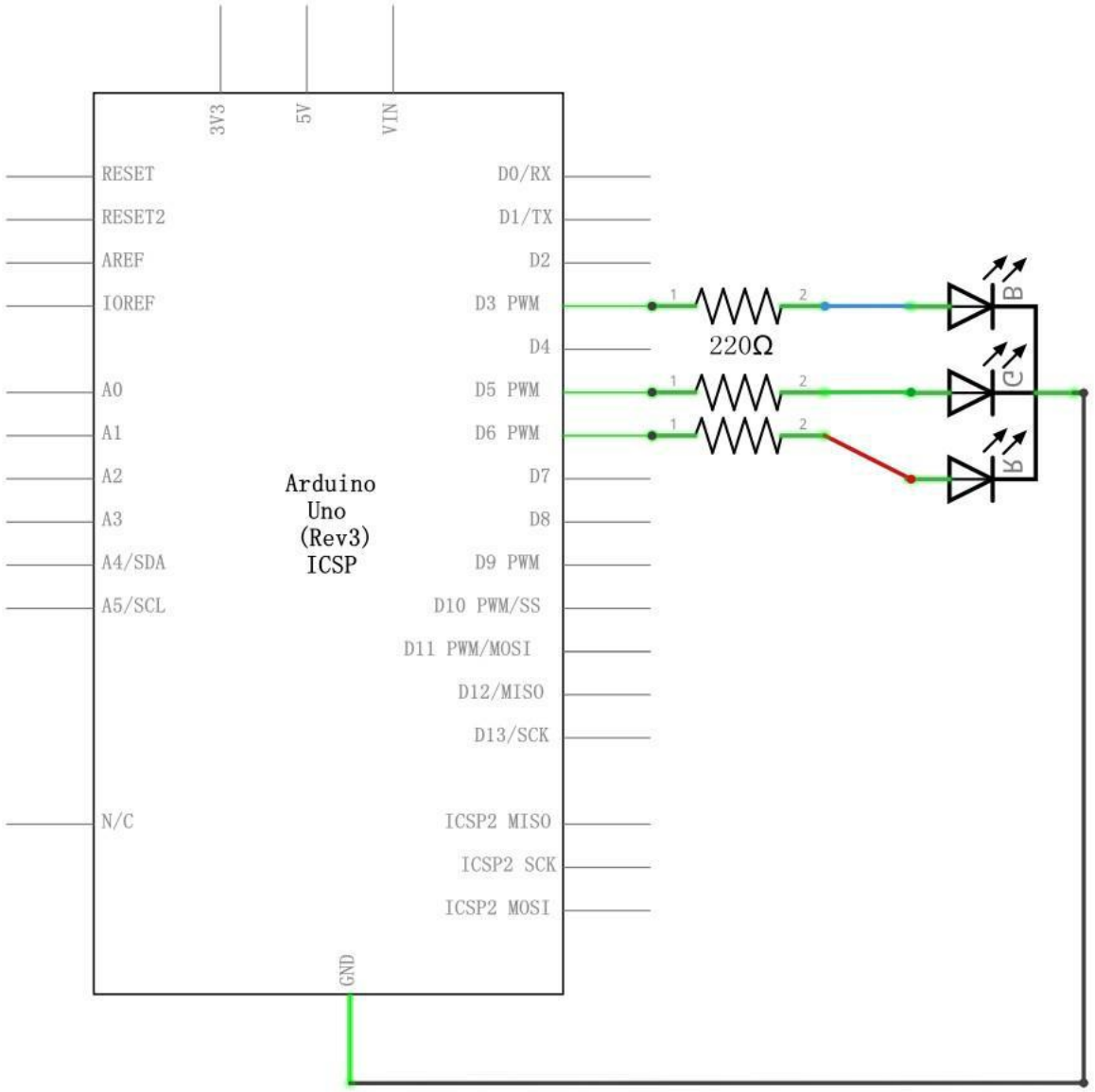
Roughly every 1/500 of a second, the PWM output will produce a pulse. The length of this pulse is controlled by the 'analogWrite' function. So 'analogWrite(0)' will not produce any pulse at all and 'analogWrite(255)' will produce a pulse that lasts all the way until the next pulse is due, so that the output is actually on all the time.

If we specify a value in the analogWrite that is somewhere in between 0 and 255, then we will produce a pulse. If the output pulse is only high for 5% of the time, then whatever we are driving will only receive 5% of full power.

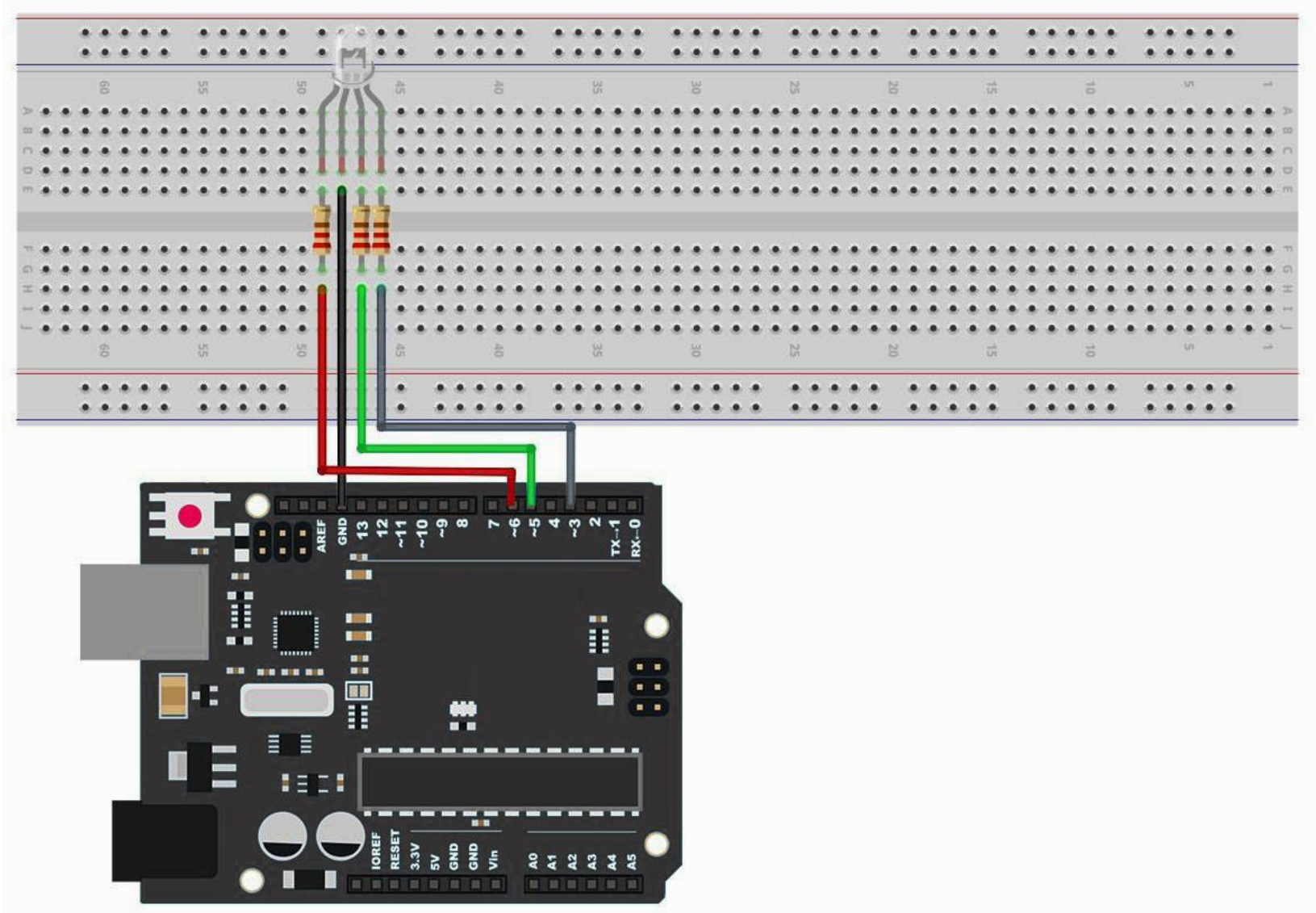
If, however, the output is at 5V for 90% of the time, then the load will get 90% of the power delivered to it. We cannot see the LEDs turning on and off at that speed, so to us, it just looks like the brightness is changing.

Connection

Schematic



Wiring diagram



Code

After wiring, please open the program in the code folder- Lesson 4 RGB LED, and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

Our code will use FOR loops to cycle through the colors.

The first FOR loop will go from RED to GREEN.

The second FOR loop will go from GREEN to BLUE.

The last FOR loop will go from BLUE to RED.

Try the sketch out and then we will dissect it in some detail.....

The sketch starts by specifying which pins are going to be used for each of the colors:

```
// Define Pins
#define BLUE 3
#define GREEN 5
#define RED 6
```

The next step is to write the 'setup' function. As we have learnt in earlier lessons, the setup function runs just once after the Arduino has reset. In this case, all it has to do is define the three pins we are using as being outputs.

```
void setup()
{
pinMode(RED, OUTPUT);
pinMode(GREEN, OUTPUT);
pinMode(BLUE, OUTPUT);
digitalWrite(RED, HIGH);
digitalWrite(GREEN, LOW);
digitalWrite(BLUE, LOW);
}
```

Before we take a look at the 'loop' function, let’s look at the last function in the sketch.

The define variables

```
redValue = 255; // choose a value between 1 and 255 to change the color.
greenValue = 0;
```

blueValue = 0;

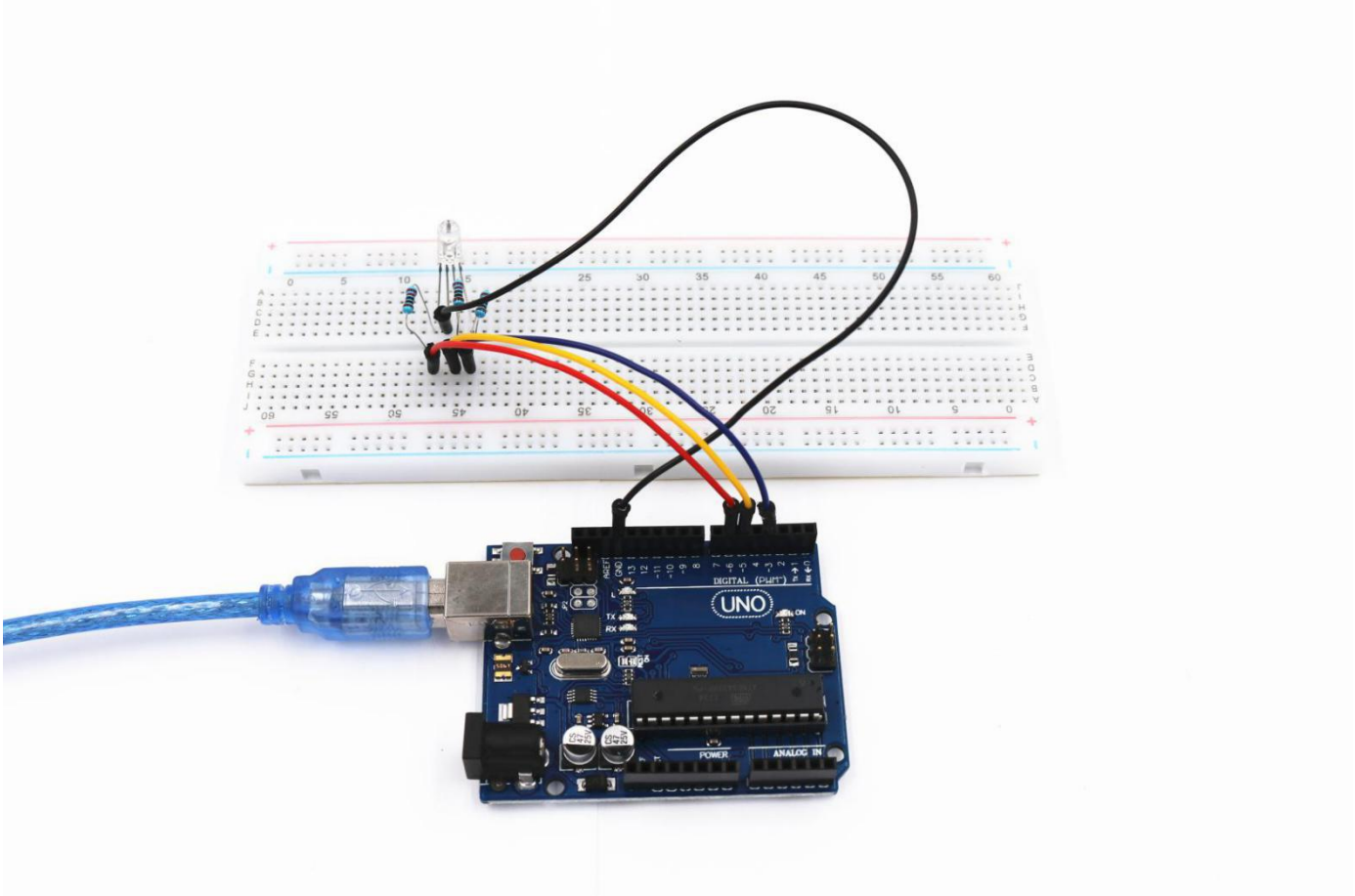
This function takes three arguments, one for the brightness of the red, green and blue LEDs. In each case the number will be in the range 0 to 255, where 0 means off and 255 means maximum brightness. The function then calls 'analogWrite' to set the brightness of each LED.

If you look at the 'loop' function you can see that we are setting the amount of red, green and blue light that we want to display and then pausing for a second before moving on to the next color.

#define delayTime 10 // fading time between colors
Delay(delayTime);

Try adding a few colors of your own to the sketch and watch the effect on your LED.

Example picture



Lesson 5 Digital Inputs

Overview

In this lesson, you will learn to use push buttons with digital inputs to turn an LED on and off.

Pressing the button will turn the LED on; pressing the other button will turn the LED off.

Component Required:

(1) x Uno R3 Board

(1) x 830 Tie-points Breadboard

(1) x 5mm red LED

(1) x 220 ohm resistor

(2) x push switches

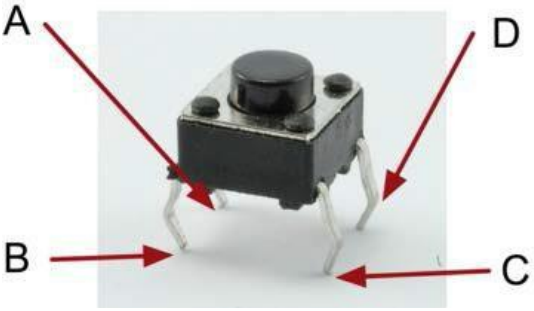
(7) x M-M wires (Male to Male jumper wires)

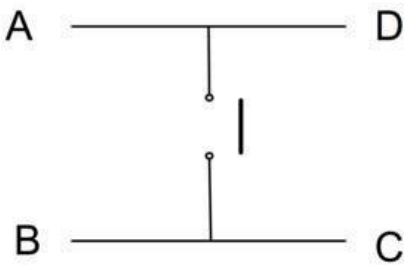
Component Introduction

PUSH SWITCHES:

Switches are really simple components. When you press a button or flip a lever, they connect two contacts together so that electricity can flow through them.

The little tactile switches that are used in this lesson have four connections, which can be a little confusing.

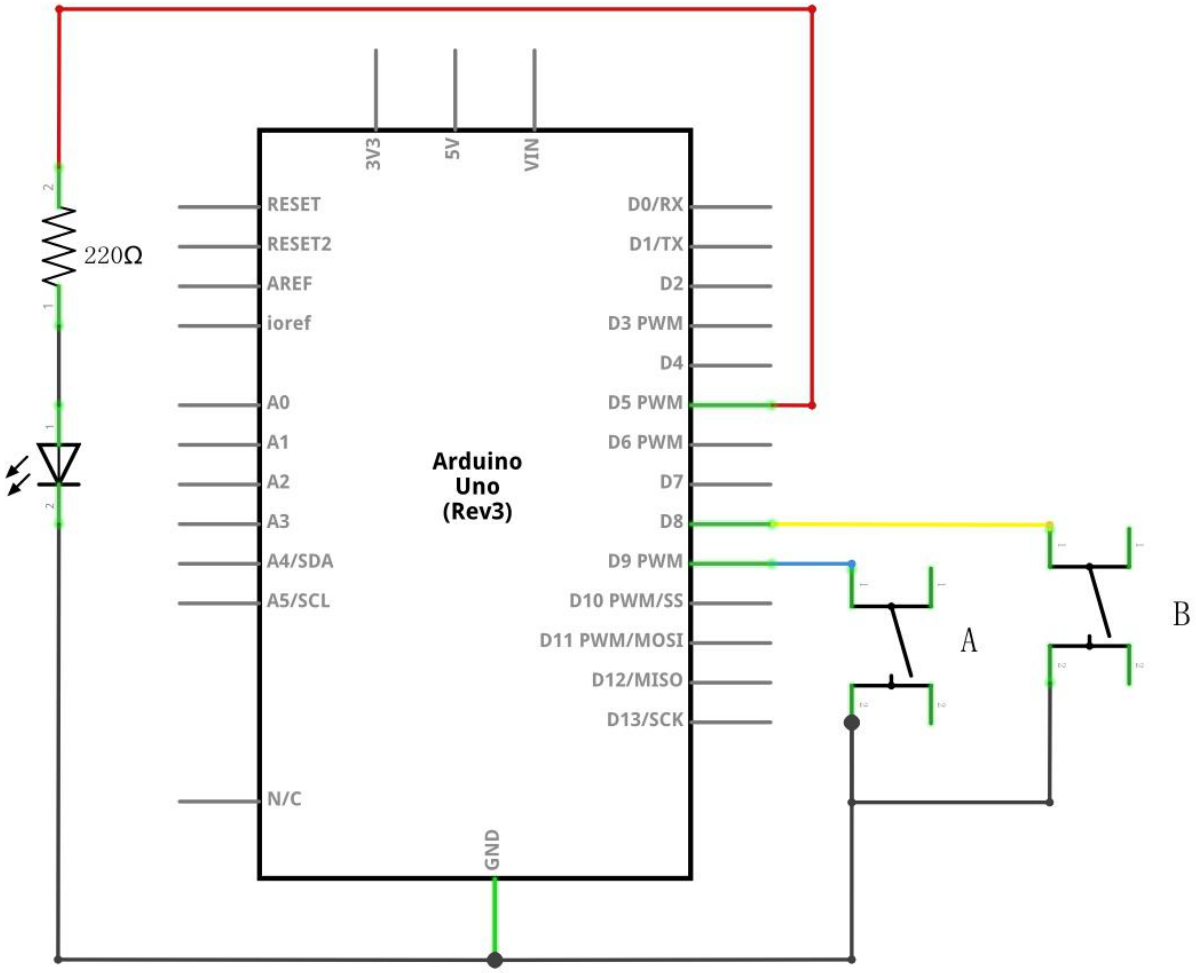




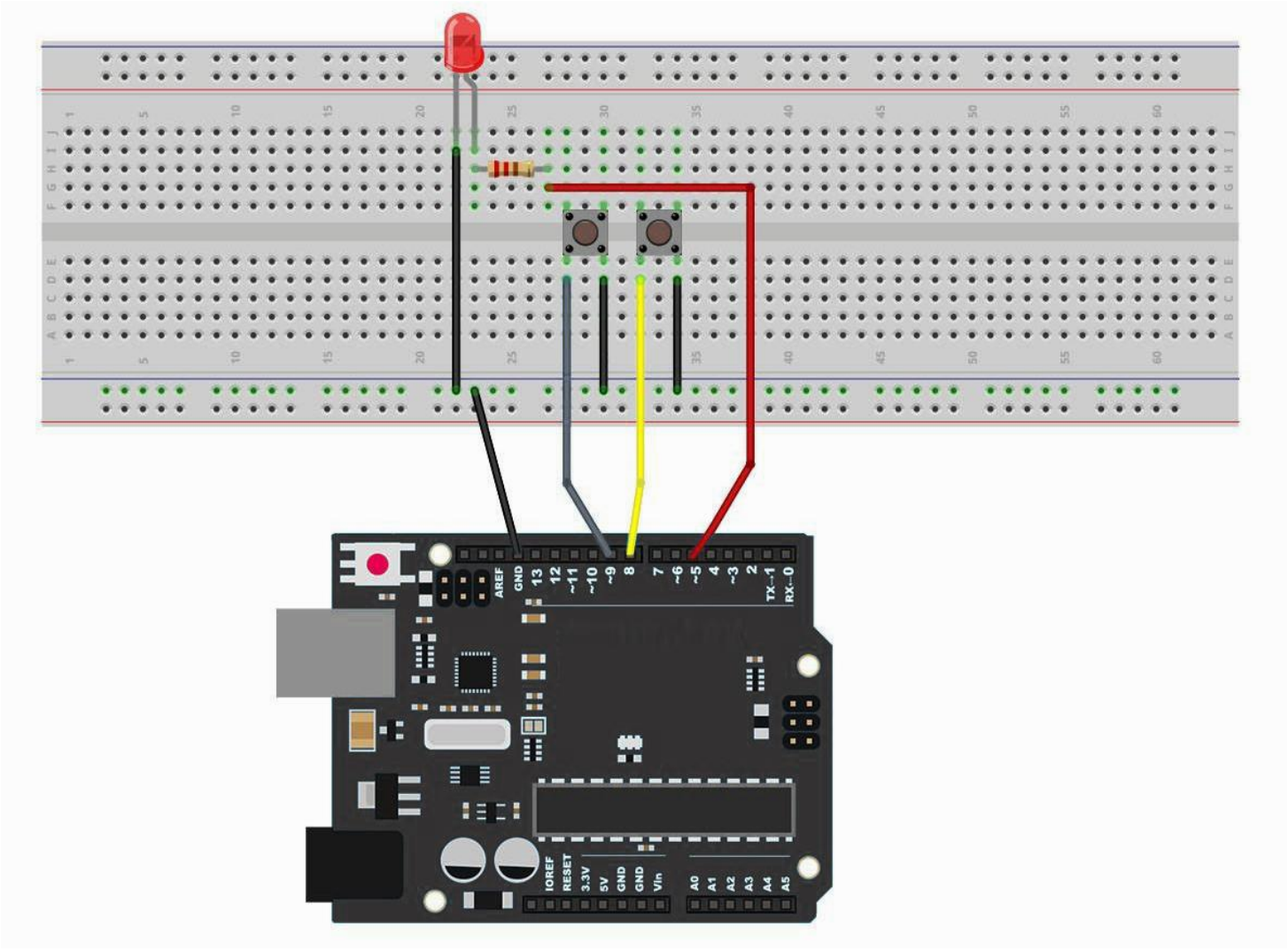
Actually, there are only really two electrical connections. Inside the switch package, pins B and C are connected together, as are A and D.

Connection

Schematic



Wiring diagram



Although the bodies of the switches are square, the pins protrude from opposite sides of the switch. This means that the pins will only be far enough apart when they are placed correctly on the breadboard.

Remember that the LED has to have the shorter negative lead to the left.

Code

After wiring， please open program in the code folder- Lesson 5 Digital Inputs, and press UPLOAD to upload the program. If errors are prompted, see Lesson 2 for details about the tutorial on program upload.

Load the sketch onto your UNO board. Pressing the left button will turn the LED on while pressing the right button will turn it off.

The first part of the sketch defines three variables for the three pins that are to be used. The 'ledPin' is the output pin and 'buttonApin' will refer to the switch nearer the top of the breadboard and 'buttonBpin' to the other switch.

The 'setup' function defines the ledPin as being an OUTPUT as normal, but now we have the two inputs to deal with. In this case, we use the set the pinMode to be 'INPUT_PULLUP' like this:

```
pinMode(buttonApin, INPUT_PULLUP);
pinMode(buttonBpin, INPUT_PULLUP);
```

The pin mode of INPUT_PULLUP means that the pin is to be used as an input, but that if nothing else is connected to the input, it should be 'pulled up' to HIGH. In other words, the default value for the input is HIGH, unless it is pulled LOW by the action of pressing the button.

This is why the switches are connected to GND. When a switch is pressed, it connects the input pin to GND, so that it is no longer HIGH.

Since the input is normally HIGH and only goes LOW when the button is pressed, the logic is a little upside down. We will handle this in the 'loop' function.

```
void loop()
{
  if (digitalRead(buttonApin) == LOW)
  {
    digitalWrite(ledPin, HIGH);
  }
  if (digitalRead(buttonBpin) == LOW)
```

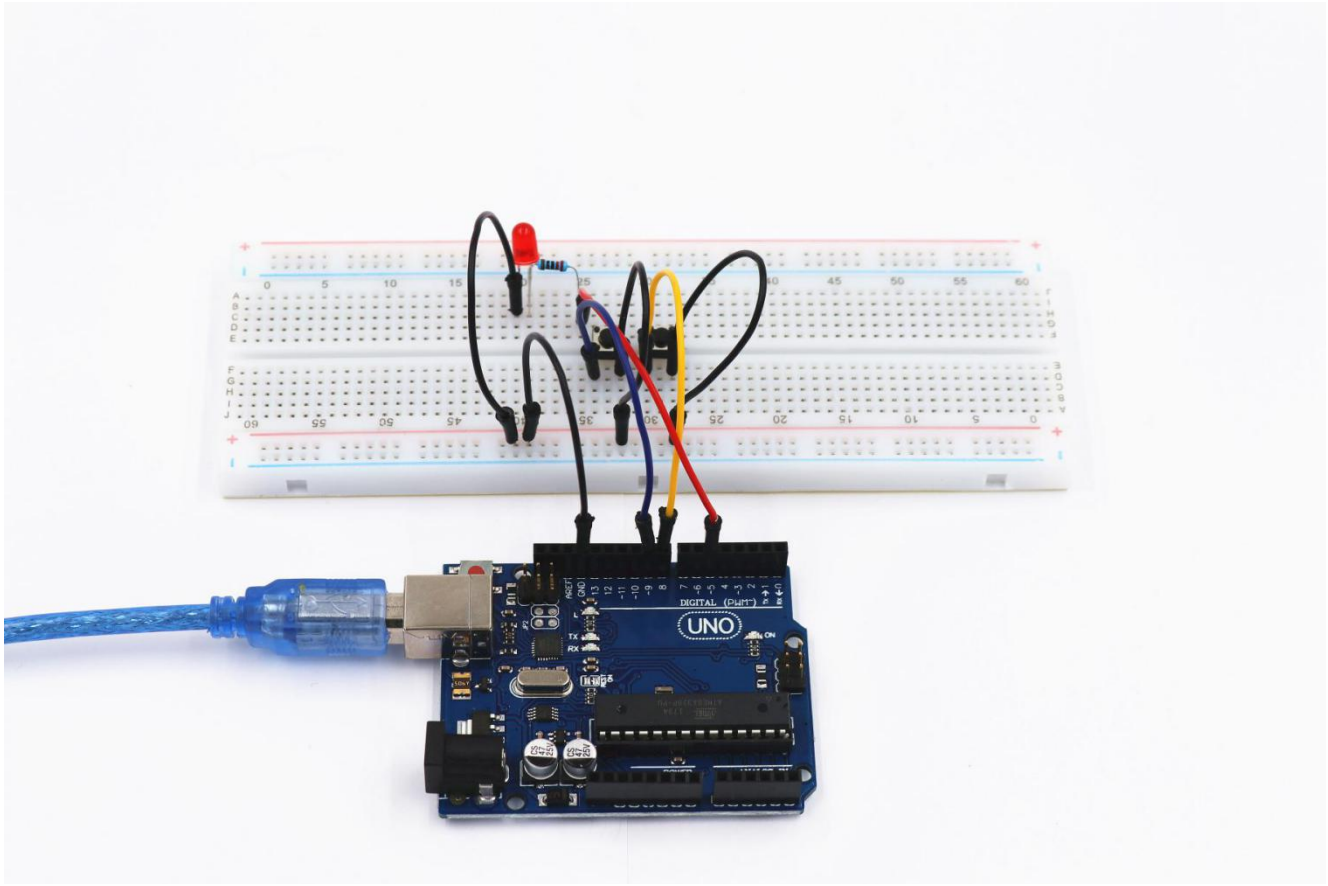
```
  {
    digitalWrite(ledPin, LOW);
  }
}
```

In the 'loop' function there are two 'if' statements. One for each button. Each does an 'digitalRead' on the appropriate input.

Remember that if the button is pressed, the corresponding input will be LOW, if button A is low, then a 'digitalWrite' on the ledPin turns it on.

Similarly, if button B is pressed, a LOW is written to the ledPin.

Example picture



Lesson 6 Tilt Ball Switch

Overview

In this lesson, you will learn how to use a tilt ball switch in order to detect small angle of inclination.

Component Required:

(1) x Uno R3 Board

(1) x Tilt Ball switch

(2) x F-M wires (Female to Male DuPont wires)



Component Introduction

Tilt sensor:

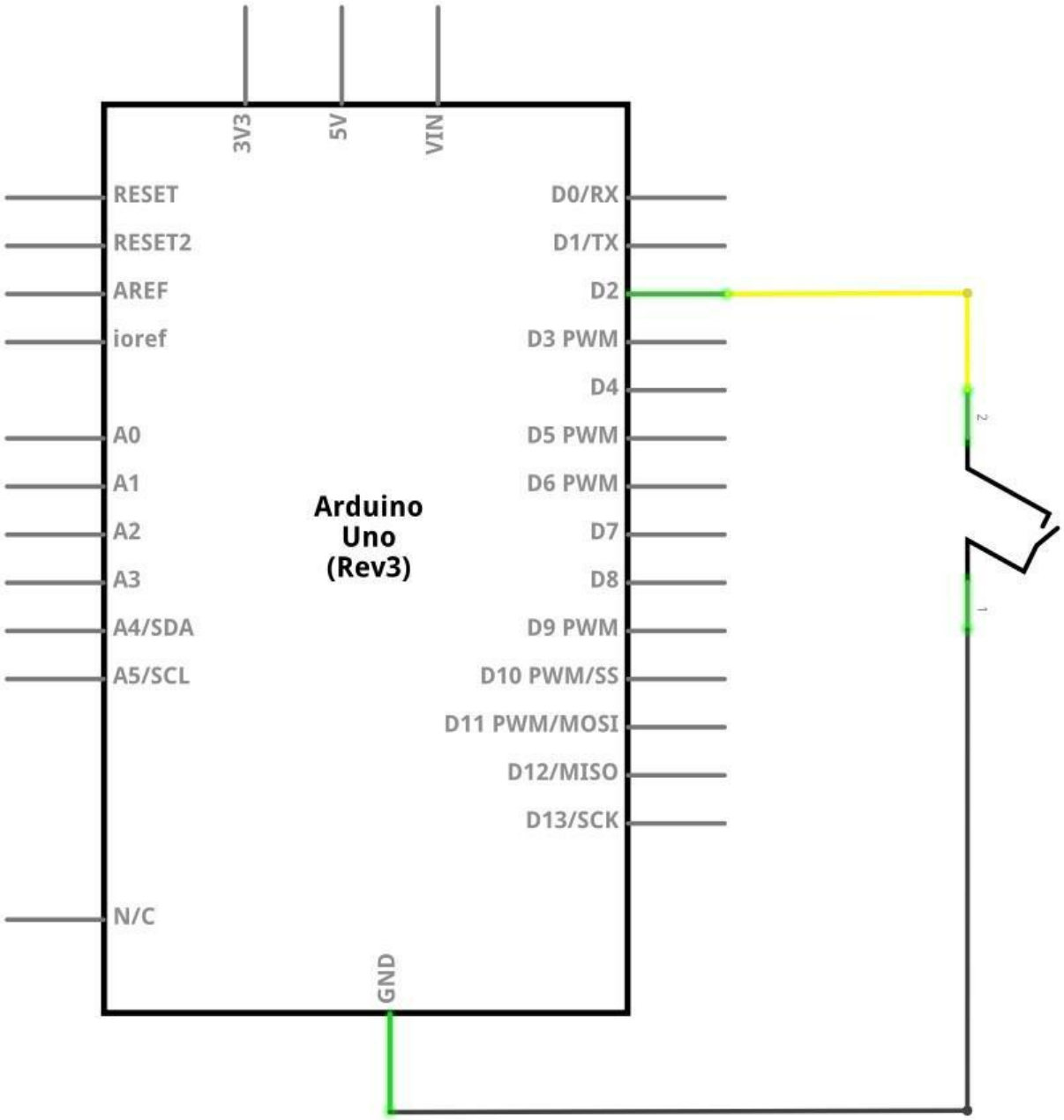
Tilt sensors (tilt ball switch) allow you to detect orientation or inclination. They are small, inexpensive, low-power and easy-to-use. If used properly, they will not wear out. Their simplicity makes them popular for toys, gadgets and appliances. Sometimes, they are referred to as "mercury switches", "tilt switches" or "rolling ball sensors" for obvious reasons.

They are usually made up of a cavity of some sort (cylindrical is popular, although not always) with a conductive free mass inside, such as a blob of mercury or rolling ball. One end of the cavity has two conductive elements (poles). When the sensor is oriented so that that end is downwards, the mass rolls onto the poles and shorts them, acting as a switch throw.

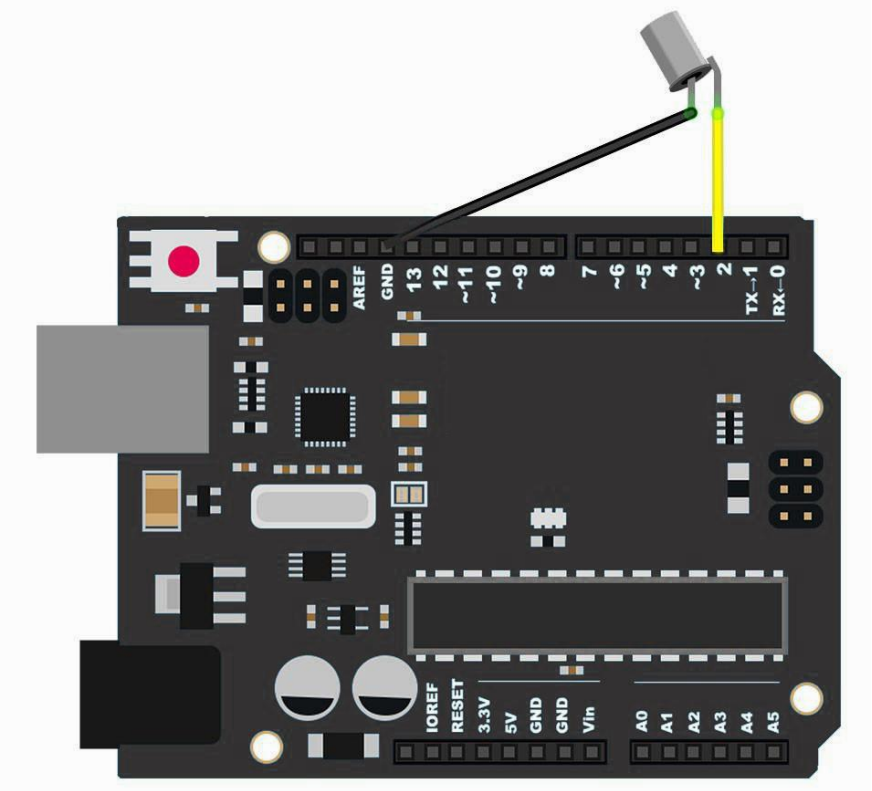
While not as precise or flexible as a full accelerometer, tilt switches can detect motion or orientation. Another benefit is that the big ones can switch power on their own. Accelerometers, on the other hand, output digital or analog voltage that must then be analyzed using extra circuitry.

Connection

Schematic



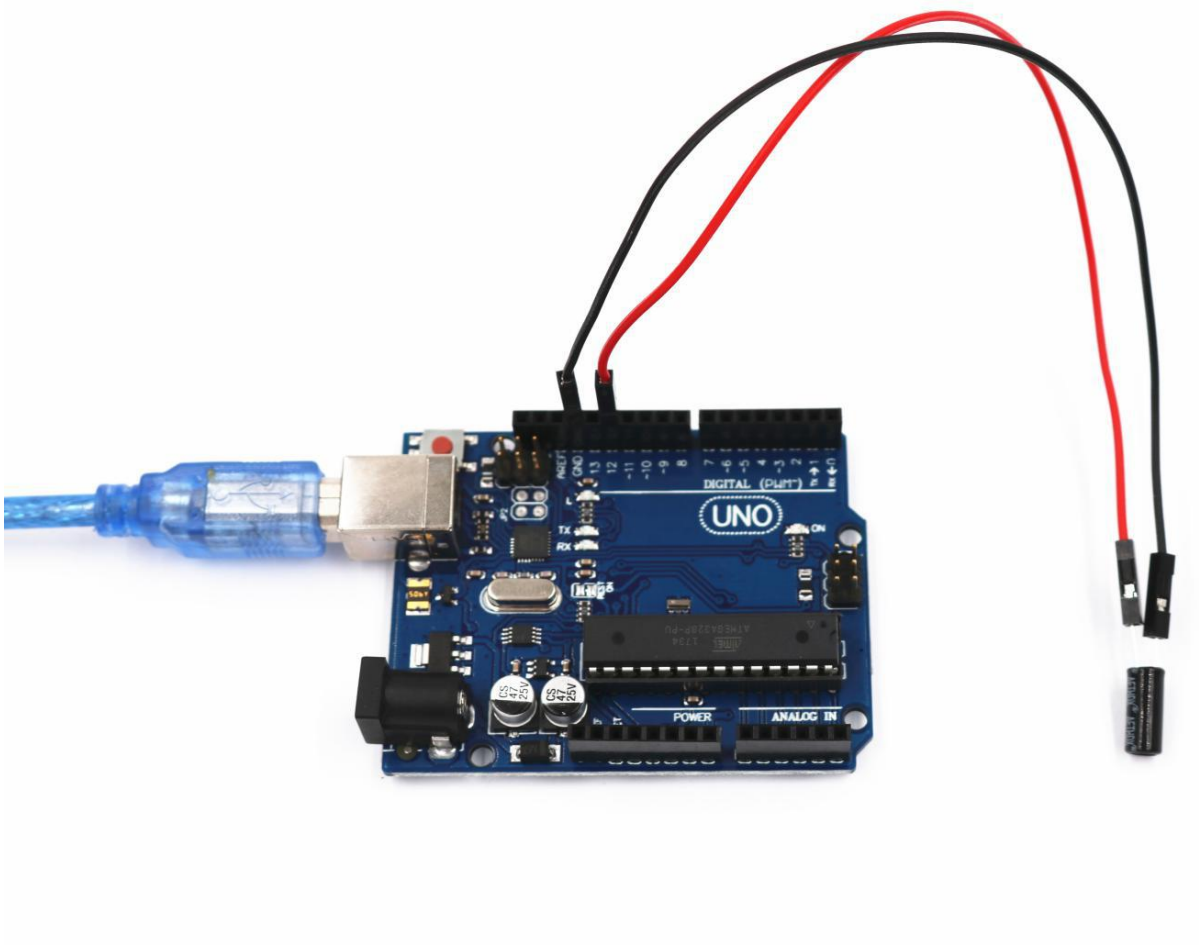
Wiring diagram



Code

After wiring, please open the program in the code folder- Lesson 6 Ball Switch and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

Example picture



Lesson 7 Active buzzer

Overview

In this lesson, you will learn how to generate a sound with an active buzzer.

Component Required:

(1) x Uno R3 Board

(1) x Active buzzer

(2) x F-M wires (Female to Male DuPont wires)

Component Introduction

BUZZER:

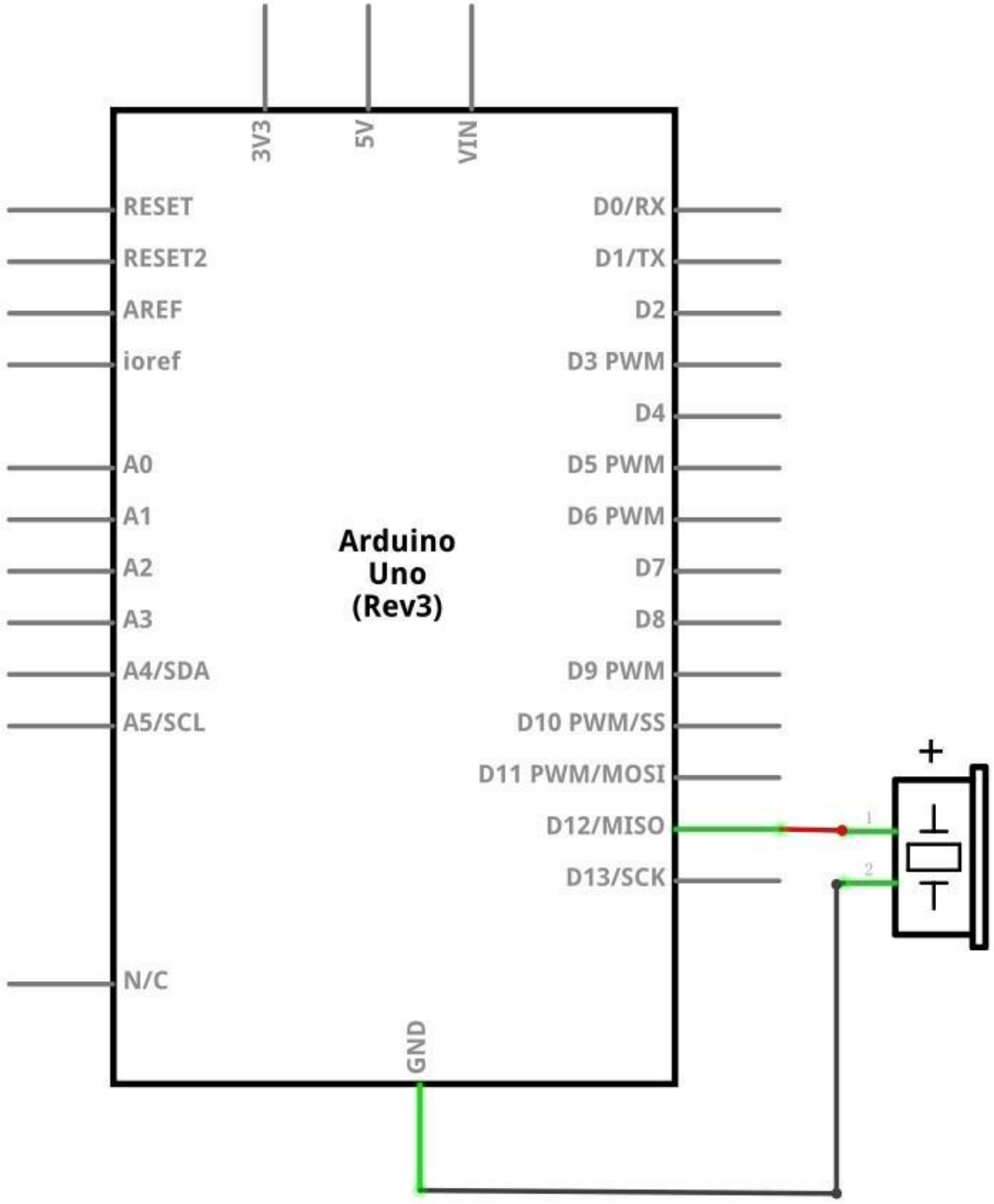
Electronic buzzers are DC-powered and equipped with an integrated circuit. They are widely used in computers, printers, photocopiers, alarms, electronic toys, automotive electronic devices, telephones, timers and other electronic products for voice devices. Buzzers can be categorized as active and passive ones. Turn the pins of two buzzers face up. The one with a green circuit board is a passive buzzer, while the other enclosed with a black tape is an active one.

The difference between the two is that an active buzzer has a built-in oscillating source, so it will generate a sound when electrified. A passive buzzer does not have such a source so it will not tweet if DC signals are used; instead, you need to use square waves whose frequency is between 2K and 5K to drive it. The active buzzer is often more expensive than the passive one because of multiple built-in oscillating circuits.

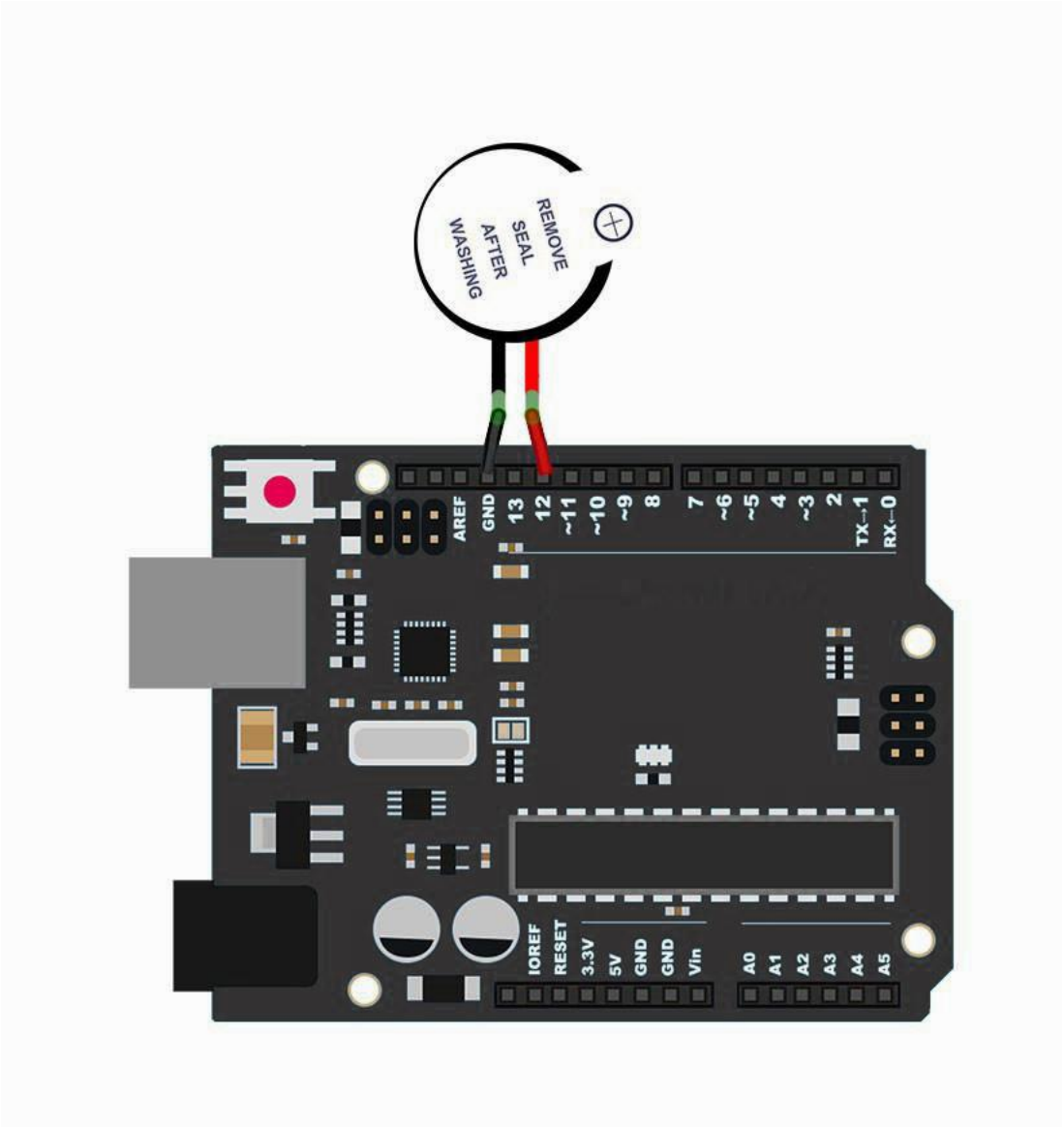


Connection

Schematic



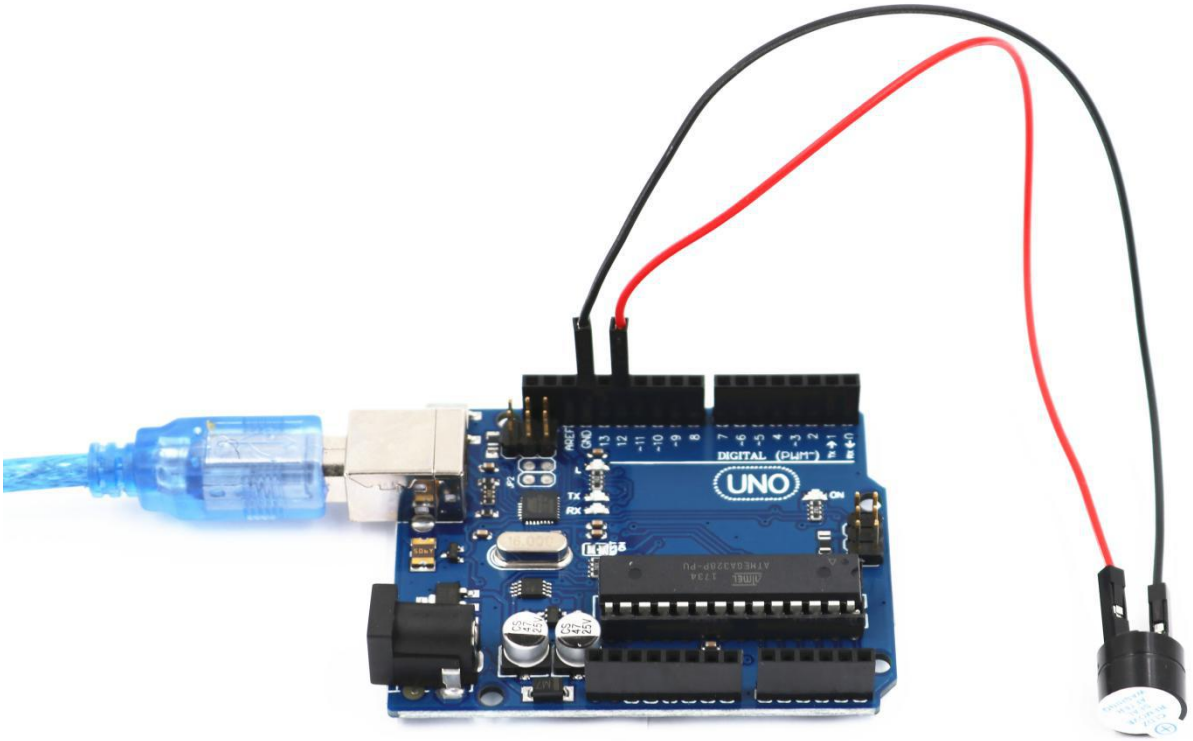
Wiring diagram



Code

After wiring, please open the program in the code folder- Lesson 7 Making Sounds and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

Example picture



Lesson 8 Passive Buzzer

Overview

In this lesson, you will learn how to use a passive buzzer.

The purpose of the experiment is to generate eight different sounds, each sound lasting 0.5 seconds: from Alto Do (523Hz), Re (587Hz), Mi (659Hz), Fa (698Hz), So (784Hz), La (880Hz), Si (988Hz) to Treble Do (1047Hz).

Component Required:

(1) x Uno R3 Board

(1) x Passive buzzer

(2) x F-M wires (Female to Male DuPont wires)

Component Introduction

Passive Buzzer:

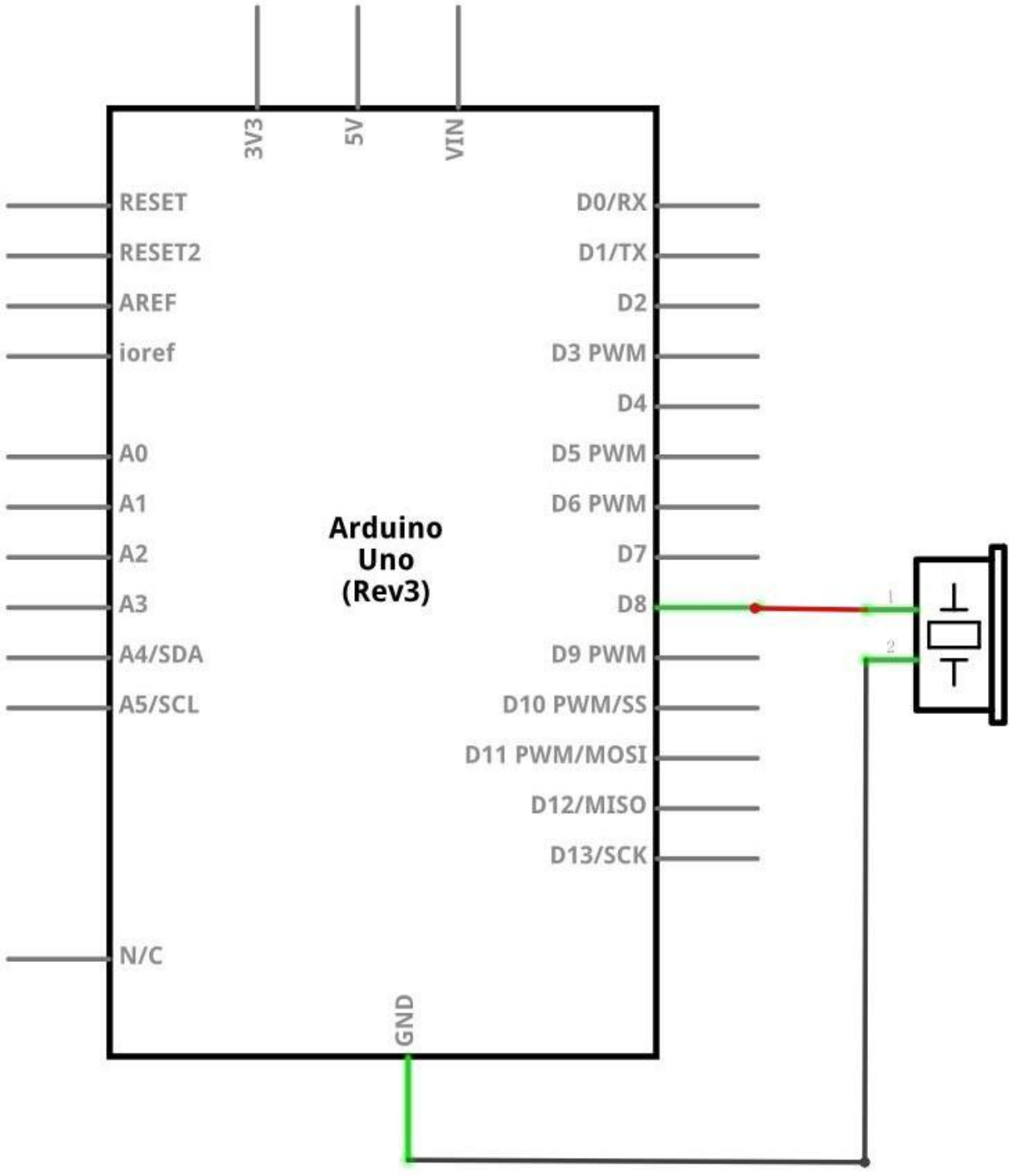
The working principle of passive buzzer is using PWM generating audio to make the air to vibrate. Appropriately changed as long as the vibration frequency, it can generate different sounds. For example, sending a pulse of 523Hz, it can generate Alto Do, pulse of 587Hz, it can generate midrange Re, pulse of 659Hz, it can produce midrange Mi. By the buzzer, you can play a song.

We should be careful not to use the UNO R3 board analog Write () function to generate a pulse to the buzzer, because the pulse output of analog Write () is fixed (500Hz).

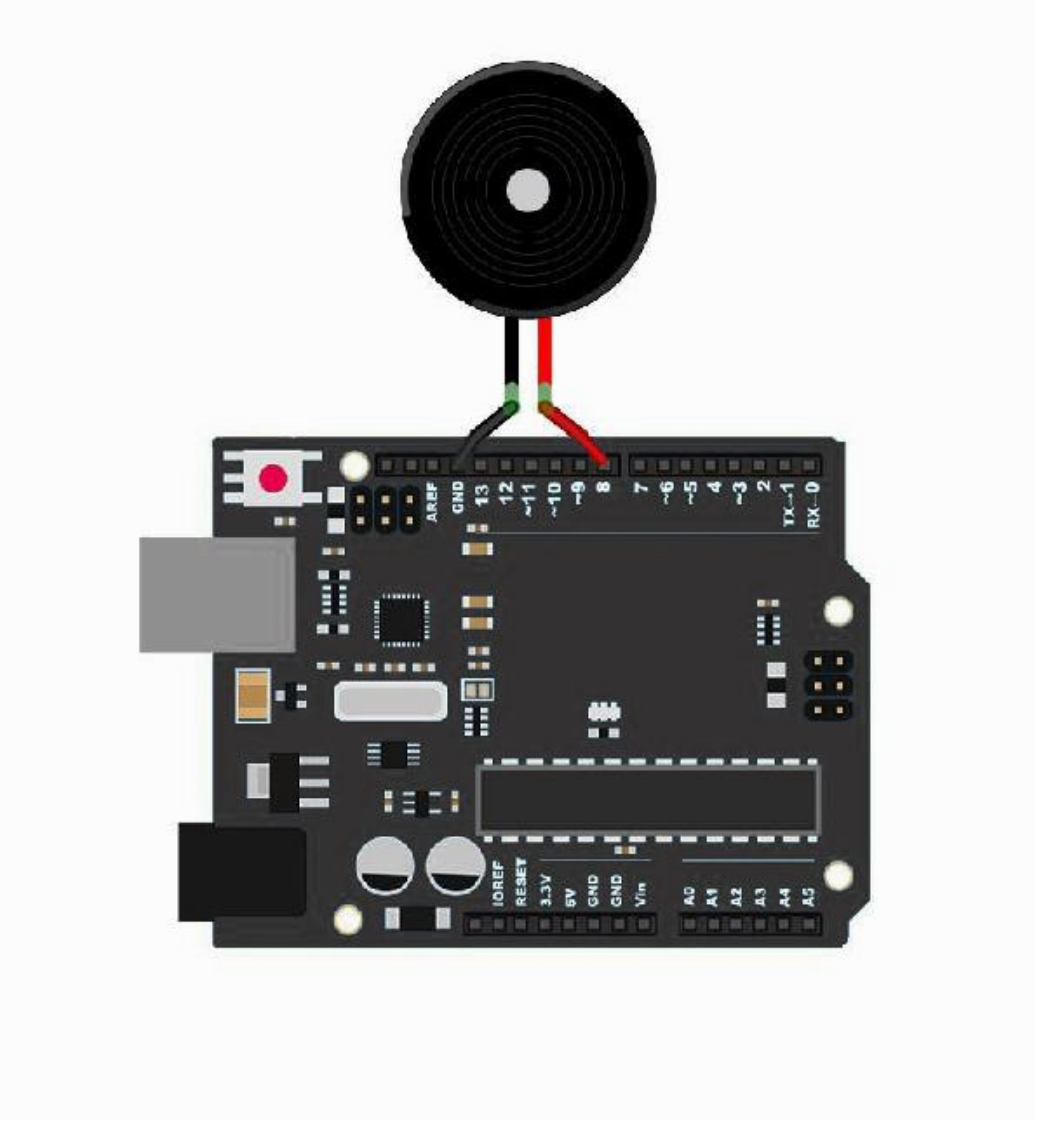


Connection

Schematic



Wiring diagram



Wiring the buzzer connected to the UNO R3 board, the red (positive) to the pin8, black wire (negative) to the GND.

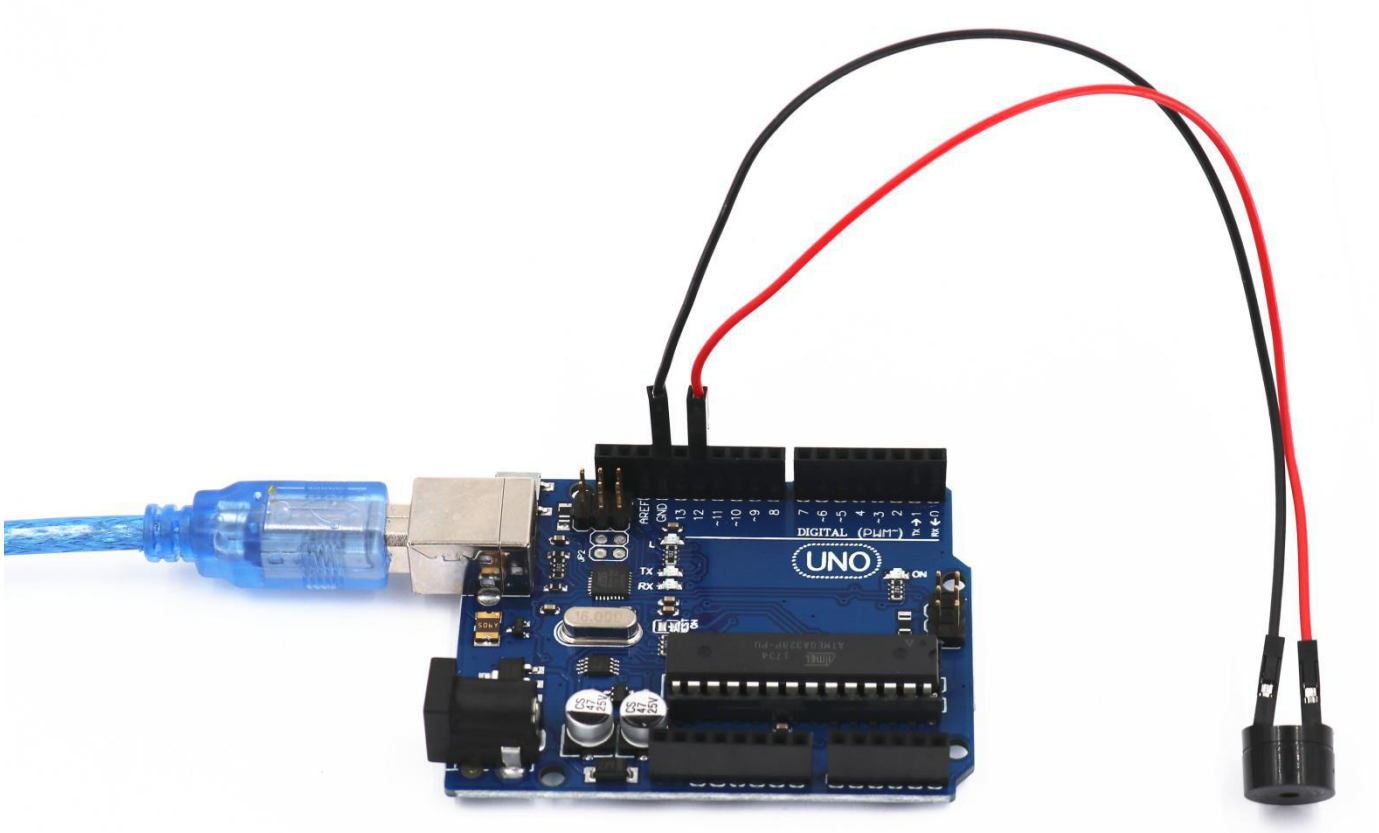
Code

After wiring, please open the program in the code folder- Lesson 8 Passive Buzzer and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

Before you can run this, make sure that you have installed the <pitches> library or re-install it, if necessary. Otherwise, your code won't work.

For details about loading the library file, see Lesson 1.

Example picture



Lesson 9 DHT11 Temperature and Humidity Sensor

Overview

In this tutorial we will learn how to use a DHT11 Temperature and Humidity Sensor.

It's accurate enough for most projects that need to keep track of humidity and temperature readings.

Again we will be using a Library specifically designed for these sensors that will make our code short and easy to write.

Component Required:

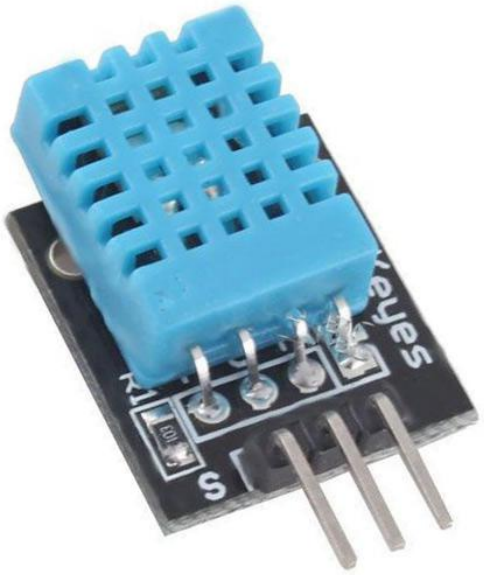
(1) x Uno R3 Board

(1) x DHT11 Temperature and Humidity module

(3) x F-M wires (Female to Male DuPont wires)

Component Introduction

Temp and humidity sensor:



DHT11 digital temperature and humidity sensor is a composite Sensor which contains a calibrated digital signal output of the temperature and humidity. The dedicated digital modules collection technology and the temperature and humidity sensing technology are applied to ensure that the product has high reliability and

excellent long-term stability. The sensor includes a resistive sense of wet components and a NTC temperature measurement devices, and connects with a high-performance 8-bit microcontroller.

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.

Product parameters

Relative humidity:

Resolution: 16Bit

Repeatability: ±1% RH

Accuracy: At 25℃ ±5% RH

Interchangeability: fully interchangeable

Response time: 1 / e (63%) of 25℃ 6s

1m / s air 6s Hysteresis:
<± 0.3% RH

Long-term stability: <± 0.5% RH / yr in

Temperature:

Resolution: 16Bit

Repeatability: ±0.2℃

Range: At 25℃ ±2℃

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

Electrical Characteristics

Power supply: DC 3.5～5.5V

Supply Current: measurement 0.3mA standby 60μA

Sampling period: more than 2 seconds

Pin Description:

1. the VDD power supply 3.5～5.5V DC

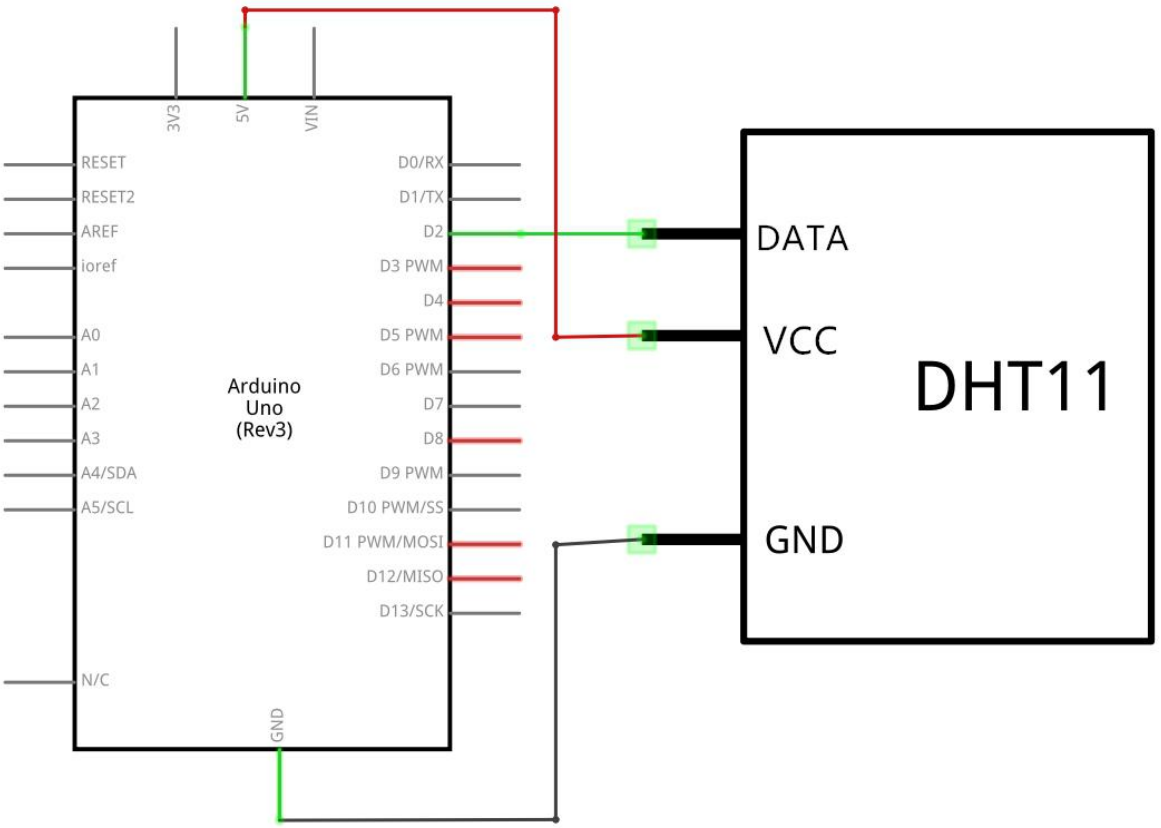
2. DATA serial data, a single bus

3. NC, empty pin

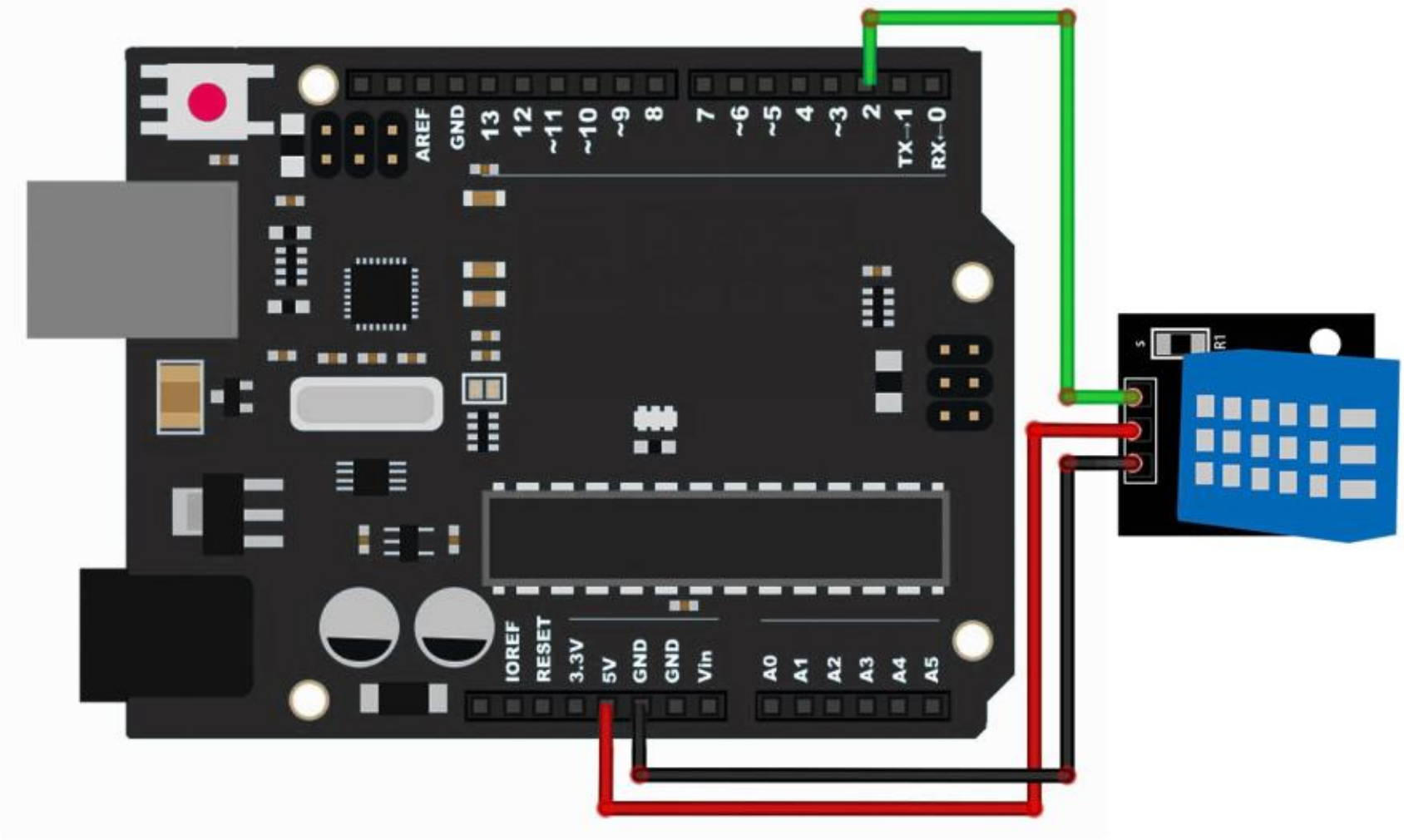
4. GND ground, the negative power

Connection

Schematic



Wiring diagram



As you can see we only need 3 connections to the sensor, since one of the pin is not used.

The connections are: Voltage, Ground and Signal which can be connected to any Pin on our UNO.

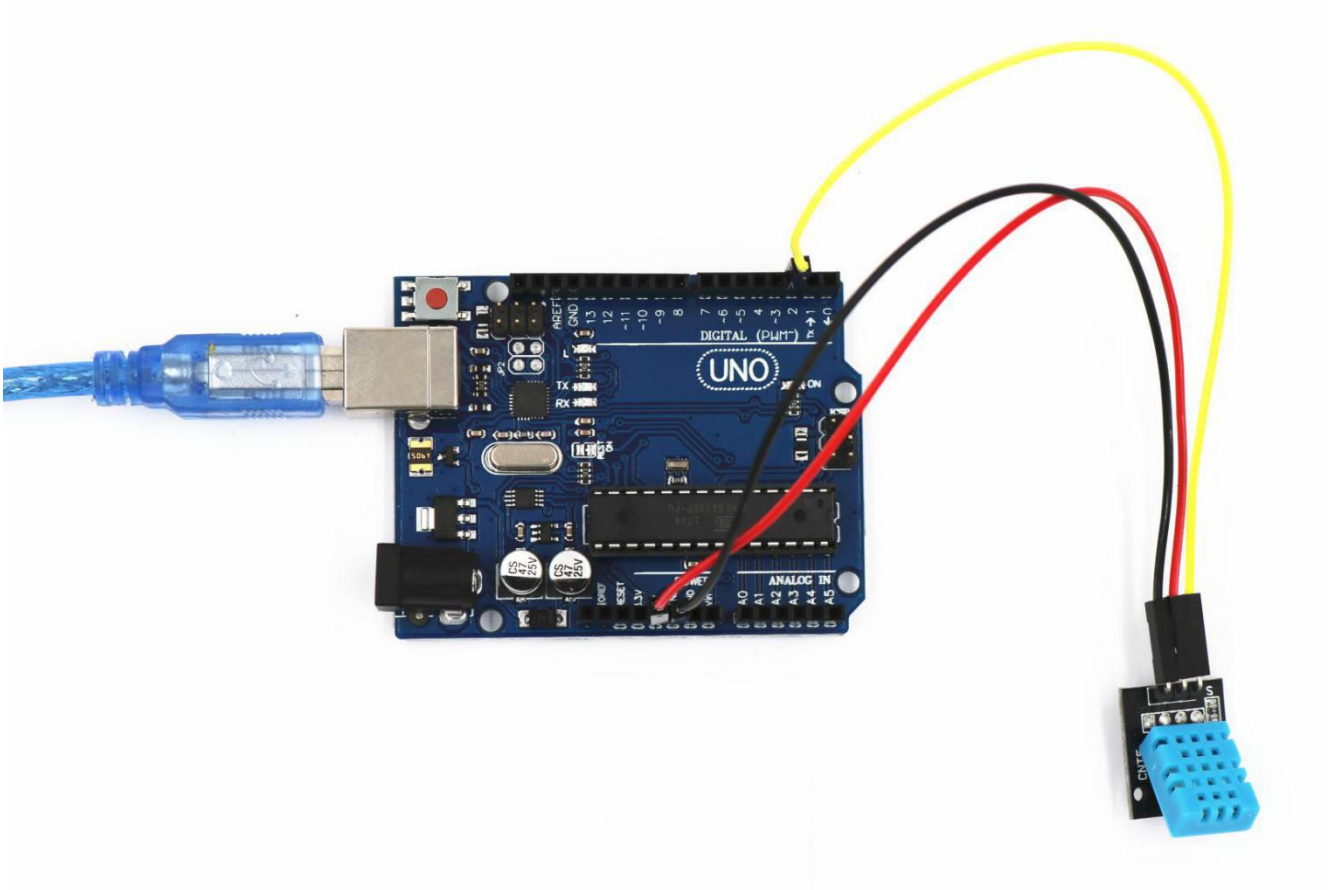
Code

After wiring, please open the program in the code folder- Lesson 9 DHT11 Temperature and Humidity Sensor and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

Before you can run this, make sure that you have installed the < SimpleDHT> library or re-install it, if necessary. Otherwise, your code won't work.

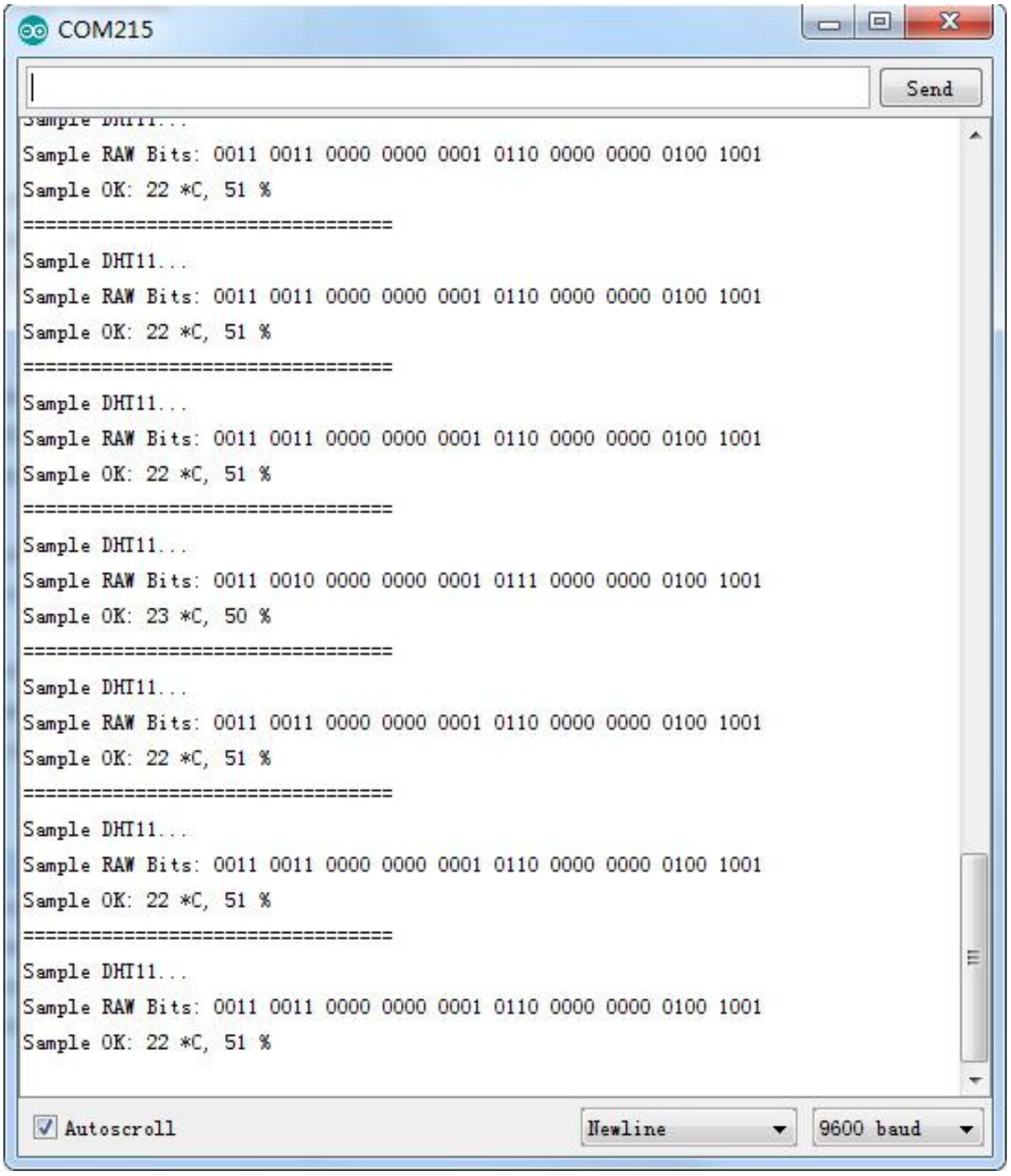
For details about the tutorial on the loading of library file, see Lesson 1.

Example picture



Upload the program then open the monitor, we can see the data as below: (It shows the temperature of the environment, we can see it is 22 degree)

Click the Serial Monitor button to turn on the serial monitor. The basics about the serial monitor are introduced in details in Lesson 1.



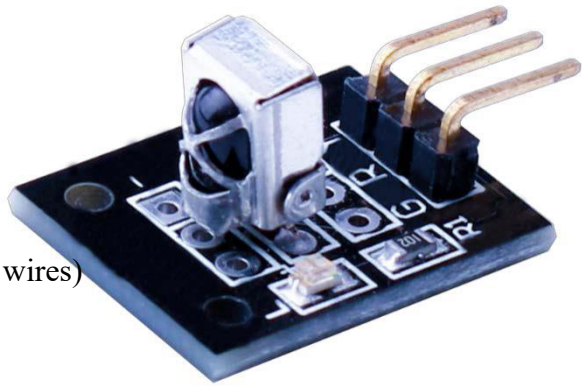
Lesson 10 IR Receiver Module

Overview

Using an IR Remote is a great way to have wireless control of your project. Infrared remotes are simple and easy to use. In this tutorial we will be connecting the IR receiver to the UNO, and then use a Library that was designed for this particular sensor.

In our sketch we will have all the IR Hexadecimal codes that are available on this remote, we will also detect if the code was recognized and also if we are holding down a key.

Component Required:



(1) x Uno R3 Board

(1) x IR receiver module

(1) x IR remote

(3) x F-M wires (Female to Male DuPont wires)

Component Introduction

IR RECEIVER SENSOR:

IR detectors are little microchips with a photocell that are tuned to listen to infrared light. They are almost always used for remote control detection - every TV and DVD player has one of these in the front to listen for the IR signal from the clicker. Inside the remote control is a matching IR LED, which emits IR pulses to tell the TV to turn on, off or change channels. IR light is not visible to the human eye, which means it takes a little more work to test a setup.

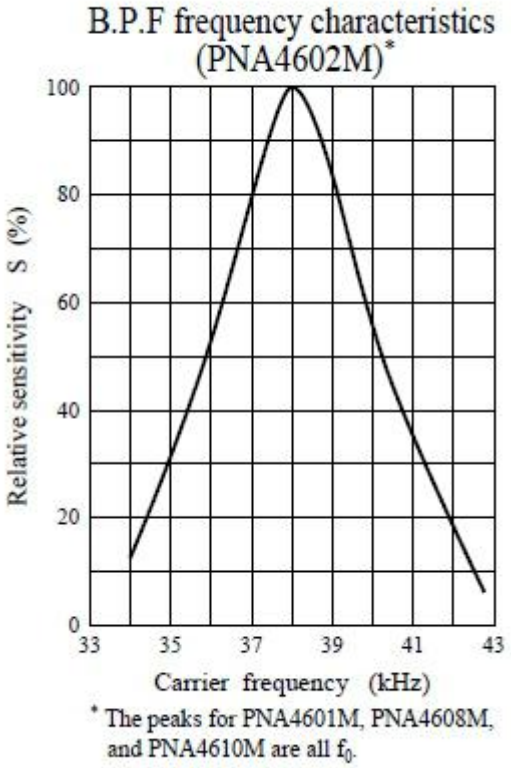
There are a few difference between these and say a CdS Photocells:

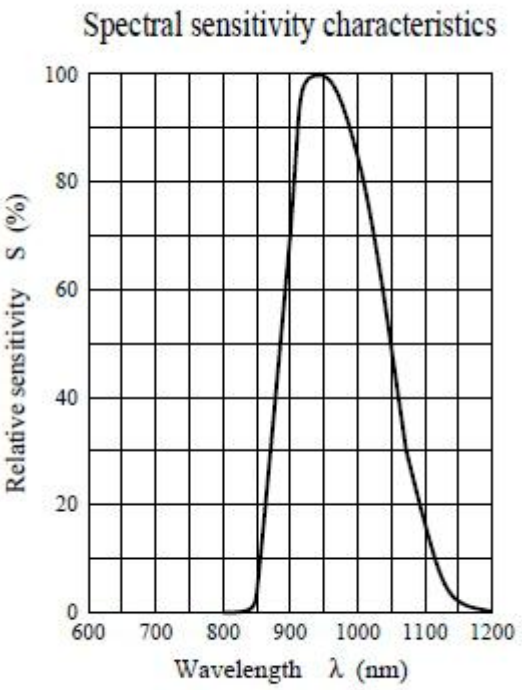
IR detectors are specially filtered for IR light, they are not good at detecting visible light. On the other hand, photocells are good at detecting yellow/green visible light, and are not good at IR light.

IR detectors have a demodulator inside that looks for modulated IR at 38 KHz. Just shining an IR LED won't be detected, it has to be PWM blinking at 38KHz. Photocells do not have any sort of demodulator and can detect any frequency (including DC) within the response speed of the photocell (which is about 1KHz)

IR detectors are digital out - either they detect 38KHz IR signal and output low (0V) or they do not detect any and output high (5V). Photocells act like resistors, the resistance changes depending on how much light they are exposed to.

What You Can Measure



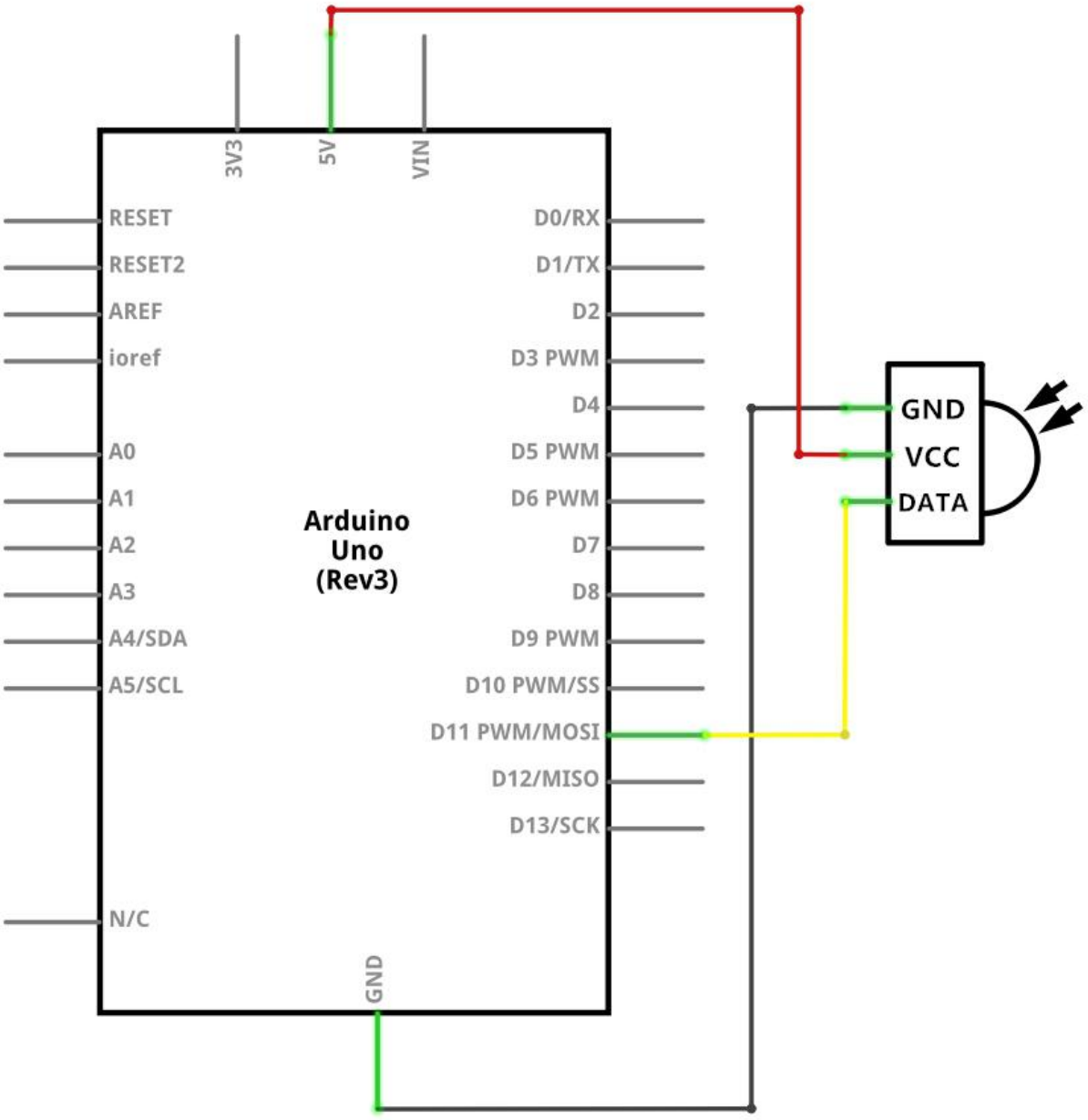


As you can see from these datasheet graphs, the peak frequency detection is at 38 KHz and the peak LED color is 940 nm. You can use from about 35 KHz to 41 KHz but the sensitivity will drop off so that it won't detect as well from afar. Likewise, you can use 850 to 1100 nm LEDs but they won't work as well as 900 to 1000nm so make sure to get matching LEDs! Check the datasheet for your IR LED to verify the wavelength.

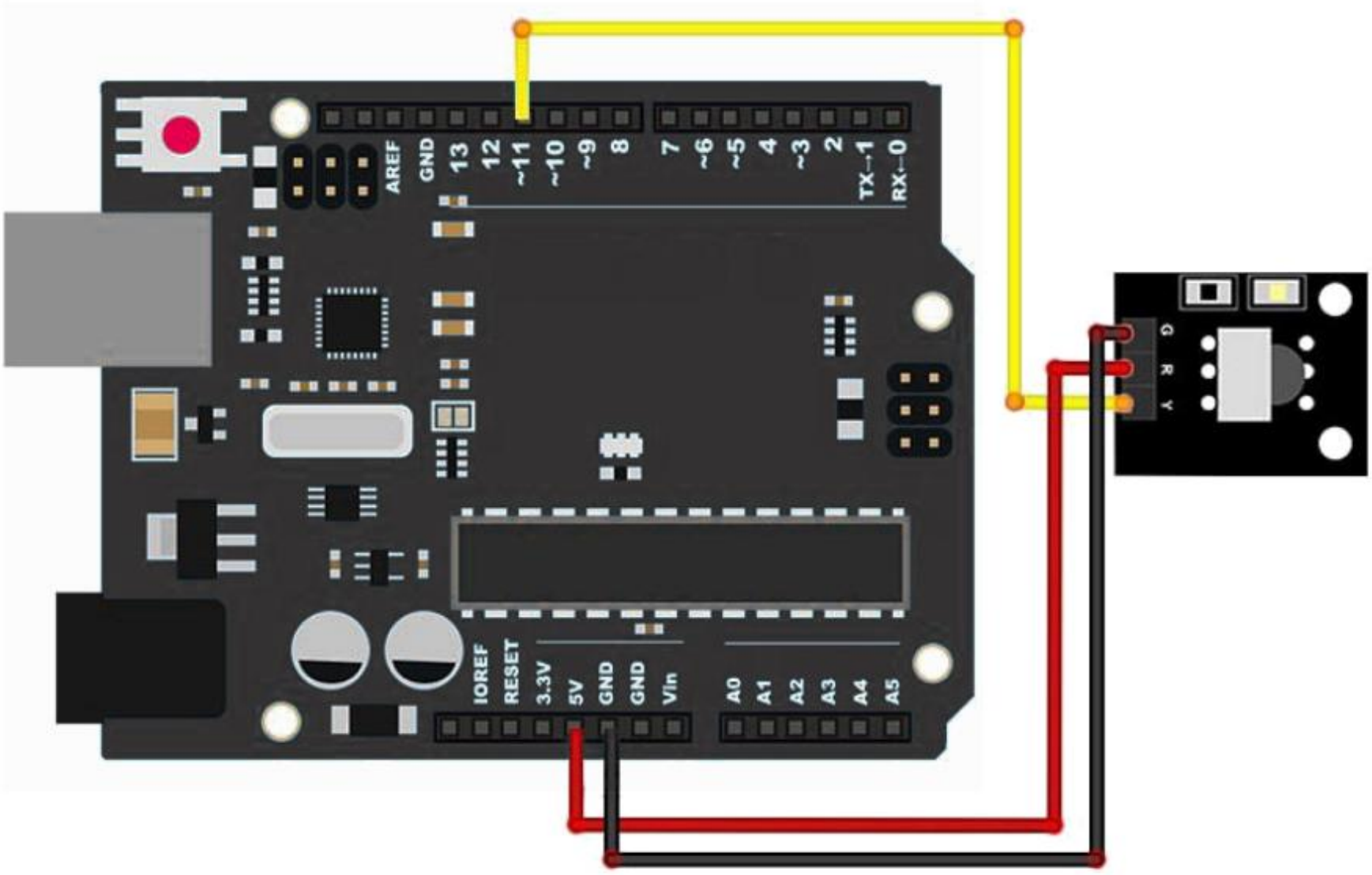
Try to get a 940nm - remember that 940nm is not visible light!

Connection

Schematic



Wiring diagram



There are 3 connections to the IR Receiver.
The connections are: Signal, Voltage and Ground.
The "-" is the Ground, "S" is signal, and middle pin is Voltage 5V.

Code

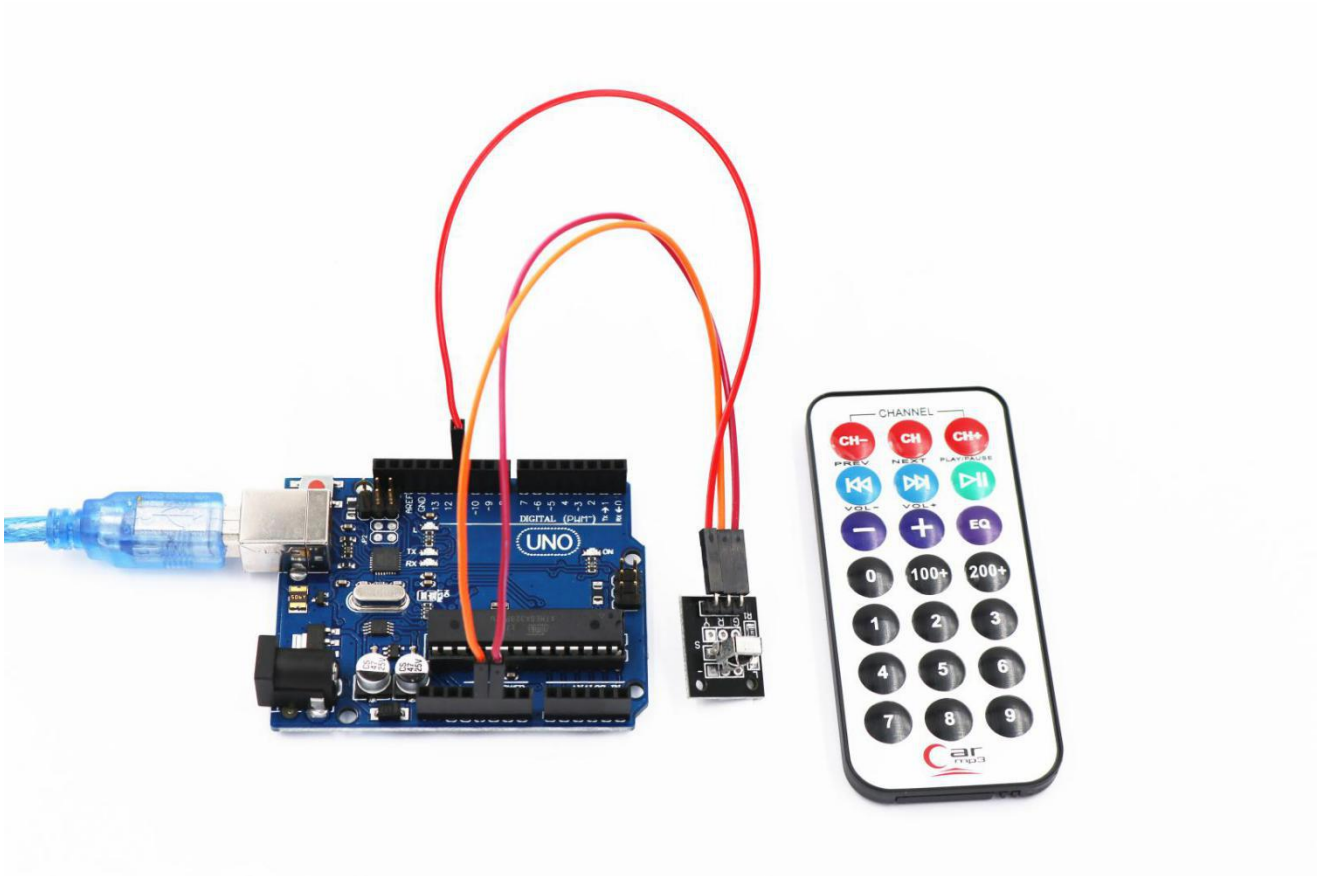
After wiring, please open the program in the code folder- Lesson 10 IR Receiver Module and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

Before you can run this, make sure that you have installed the < IRremote > library or re-install it, if necessary. Otherwise, your code won't work.

For details about loading the library file, see Lesson 1.

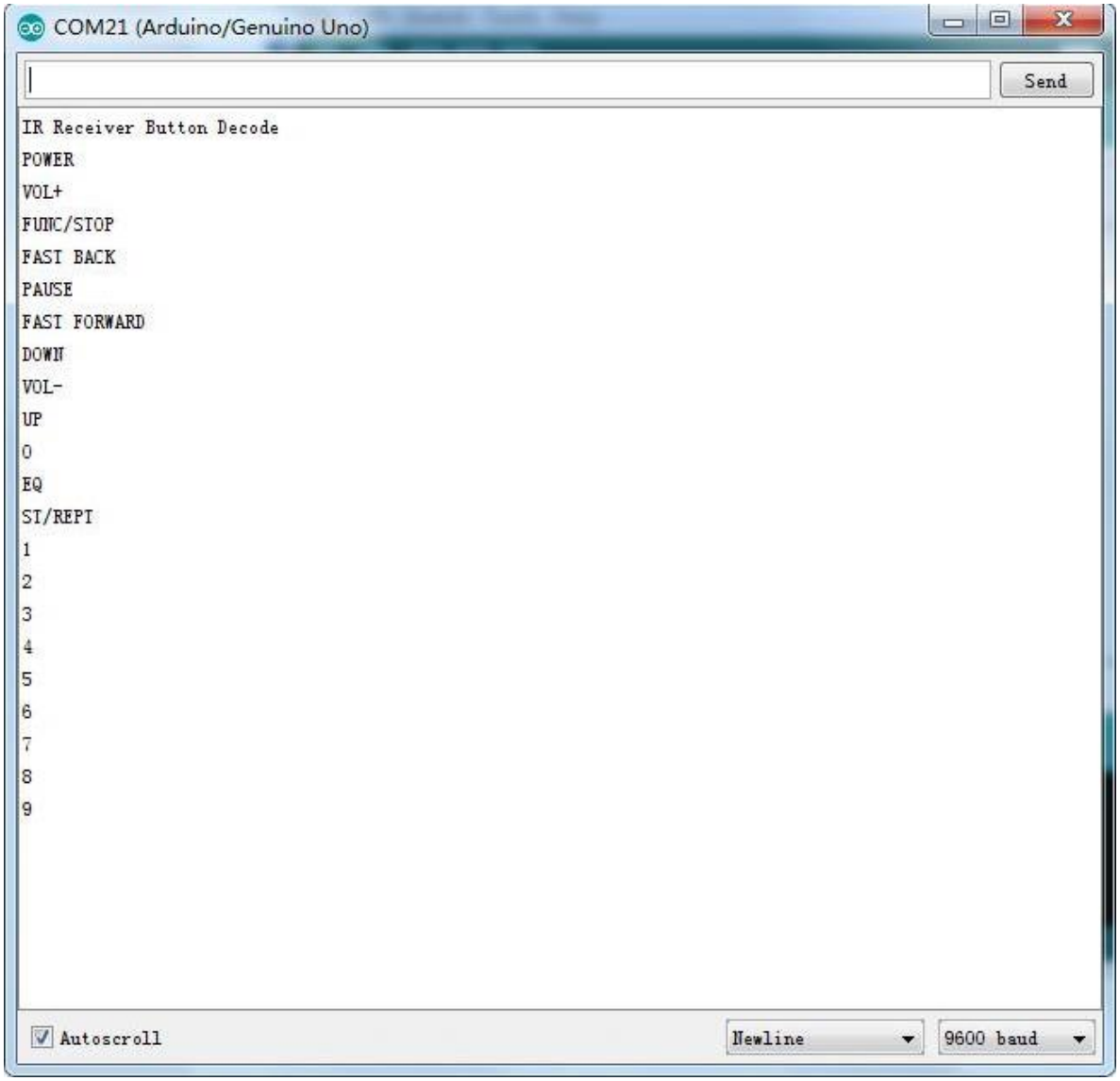
Next we will move the <RobotIRremote> out of the Library folder, we do this because that library conflicts with the one we will be using. You can just drag it back inside the library folder once you are done programming your microcontroller. Once you have installed the Library, just go ahead and restart your IDE Software.

Example picture



Open the monitor then you can see the data as blow:

Click the Serial Monitor button to turn on the serial monitor. The basics about the serial monitor are introduced in details in Lesson 1.



Lesson 11 Servo

Overview

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.

Component Required:

(1) x Uno R3 Board

(1) x Servo (SG90)

(3) x M-M wires (Male to Male jumper wires)

Component Introduction

SG90

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

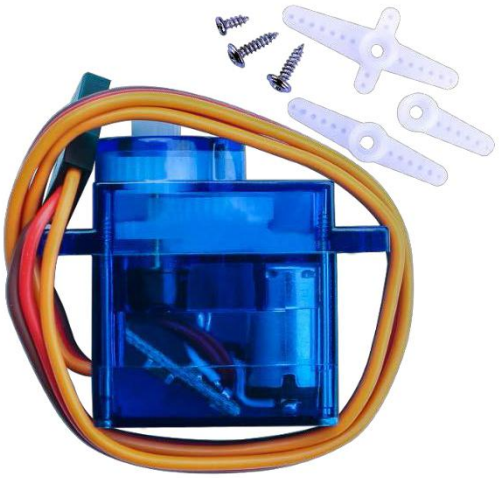
Temperature : -30~60'C

Dead band width: 5us

Working voltage: 3.5~6V

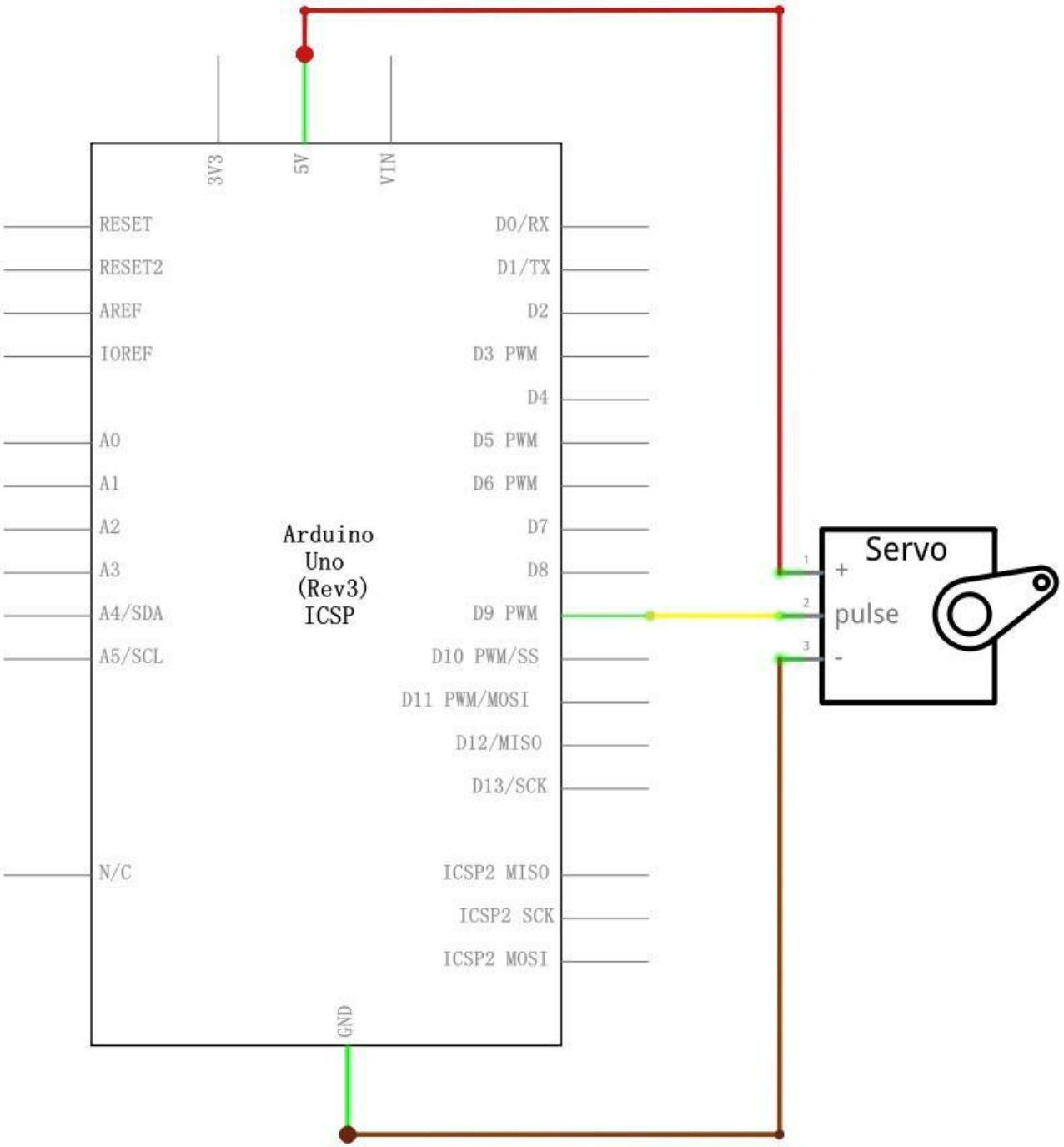
Dimension : 1.26 in x 1.18 in x 0.47 in (3.2 cm x 3 cm x 1.2 cm)

Weight : 4.73 oz (134 g)

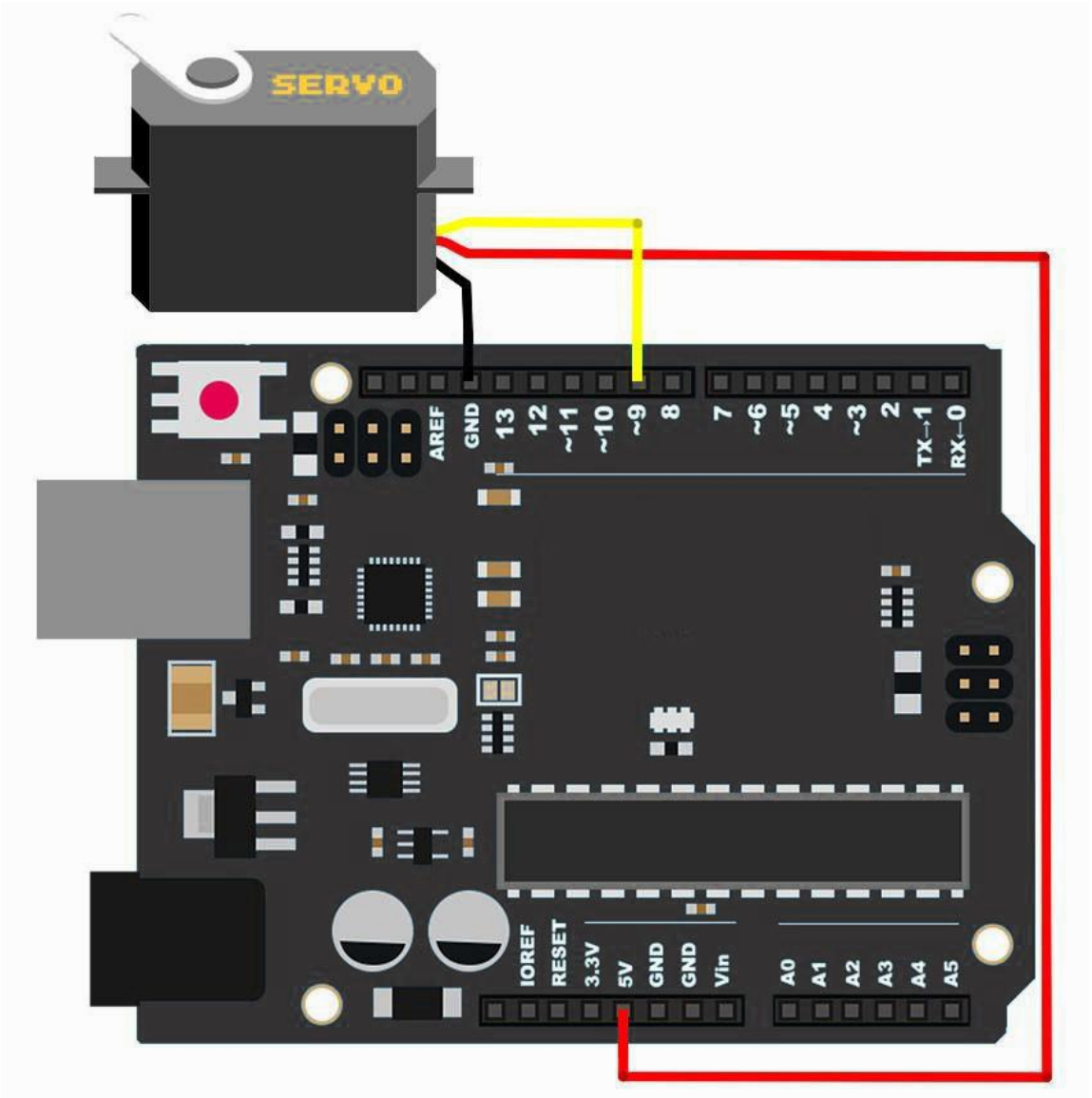


Connection

Schematic



Wiring diagram



Code

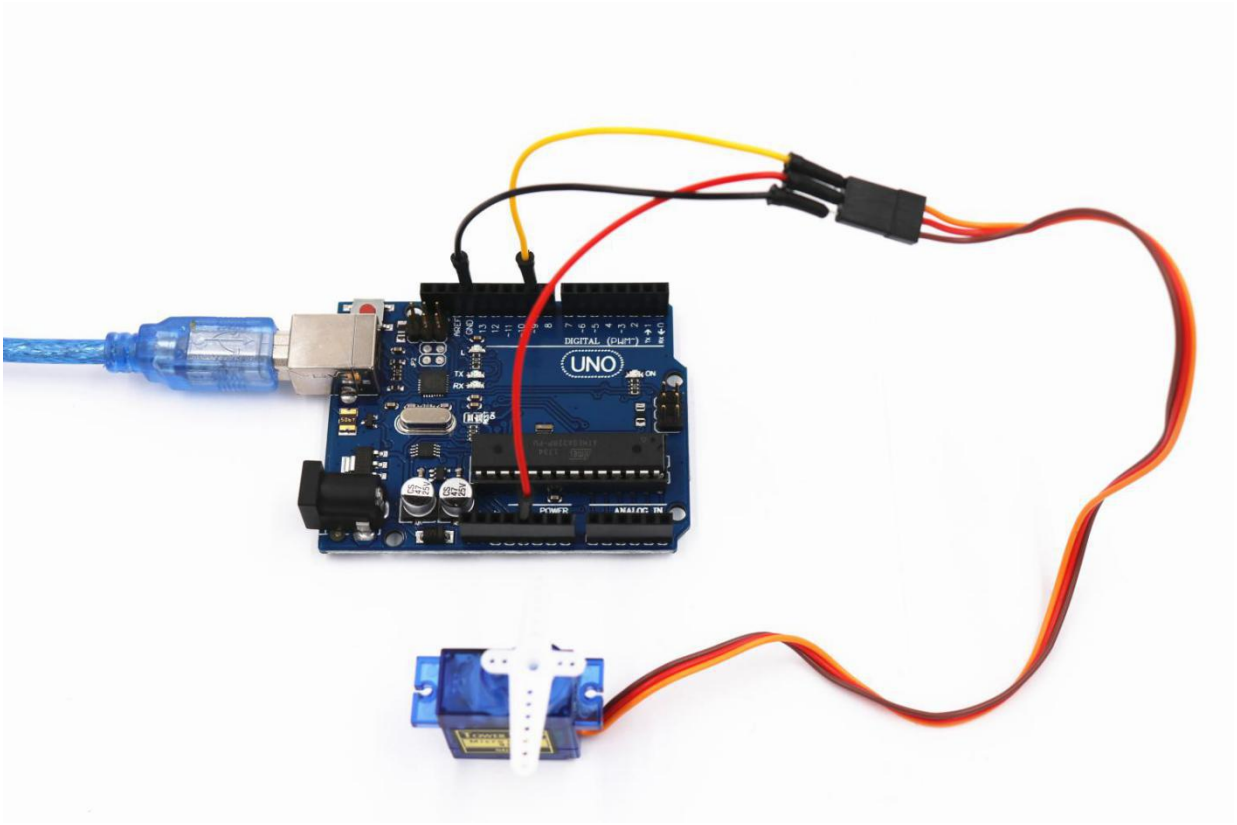
After wiring, please open the program in the code folder- Lesson 11 Servo and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

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 Lesson 1.

Example picture

In the picture, the brown wire of servo is adapted via the black M-M wires, the red one is adapted via the red M-M wires, and the orange one is adapted via the yellow M-M wires.



Lesson 12 Analog Joystick Module

Overview

Analog joysticks are a great way to add some control in your projects. In this tutorial we will learn how to use the analog joystick module.

Component Required:

(1) x Uno R3 Board

(1) x Joystick module

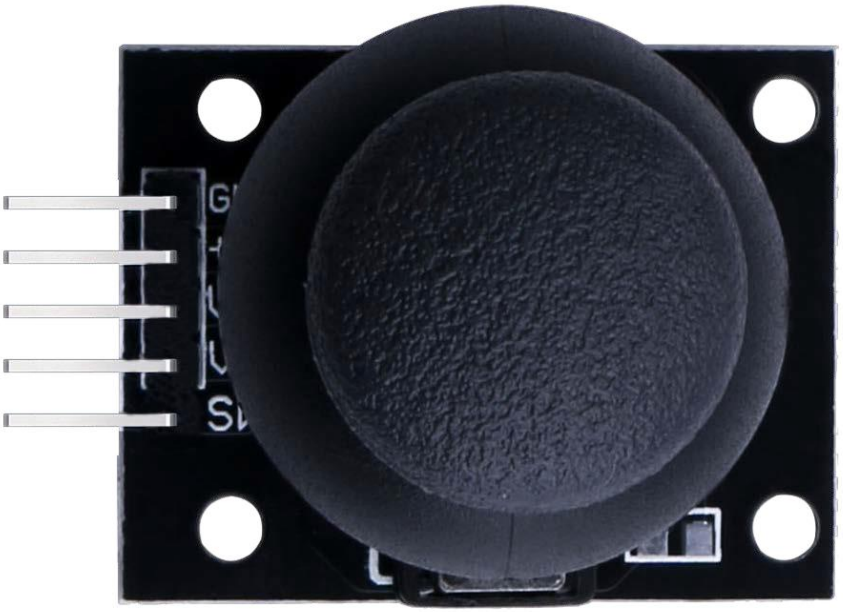
(5) x F-M wires (Female to Male DuPont wires)

Component Introduction

Joystick

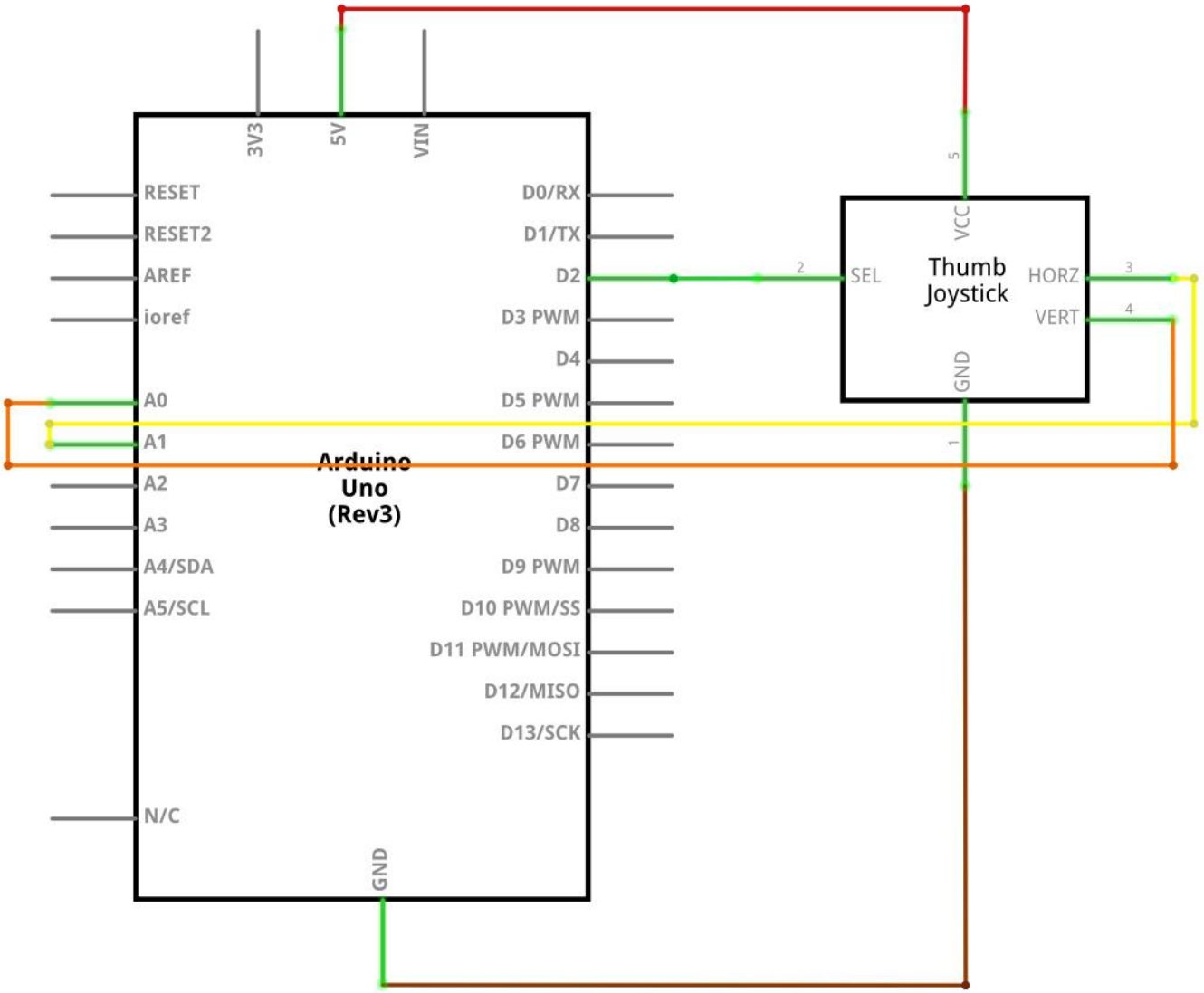
The module has 5 pins: VCC, Ground, X, Y, Key. Note that the labels on yours may be slightly different, depending on where you got the module from. The thumb stick is analog and should provide more accurate readings than simple ‘directional’ joysticks tact use some forms of buttons, or mechanical switches. Additionally, you can press the joystick down (rather hard on mine) to activate a ‘press to select’ push-button.

We have to use analog Arduino pins to read the data from the X/Y pins, and a digital pin to read the button. The Key pin is connected to ground, when the joystick is pressed down, and is floating otherwise. To get stable readings from the Key /Select pin, it needs to be connected to VCC via a pull-up resistor. The built in resistors on the Arduino digital pins can be used. For a tutorial on how to activate the pull-up resistors for Arduino pins, configured as inputs



Connection

Schematic



Wiring diagram

