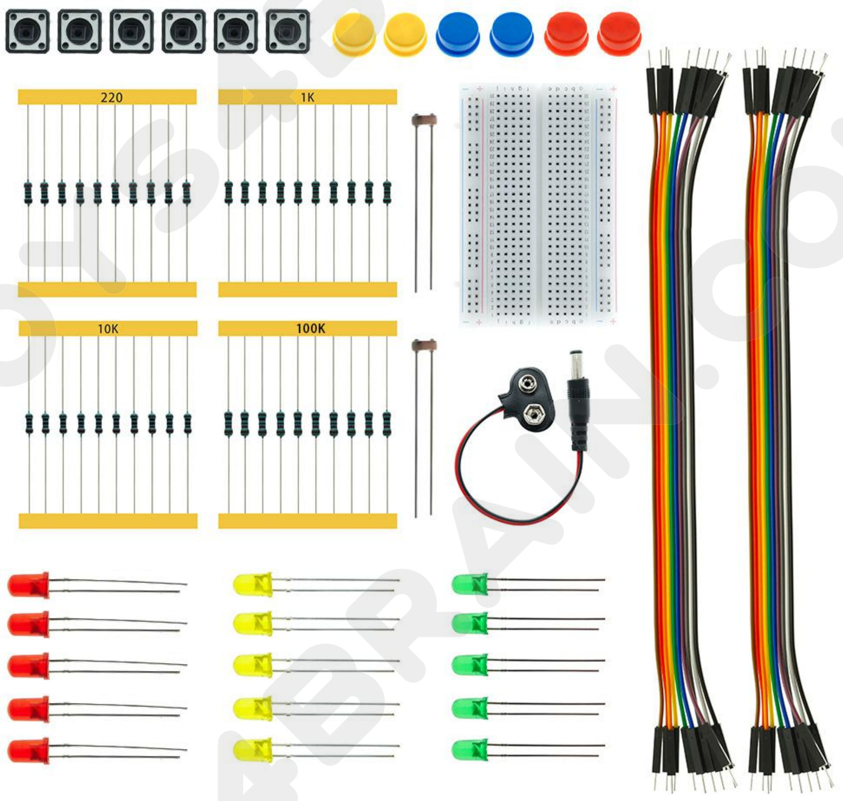
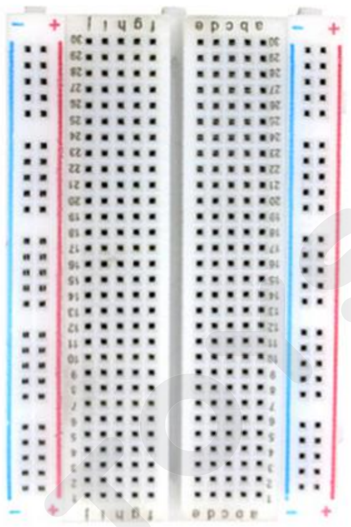
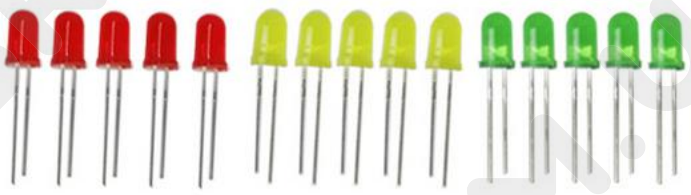
Starter Kit UNO R3 Mini Breadboard LED Jumper Wire Button for arduino Diy Kit

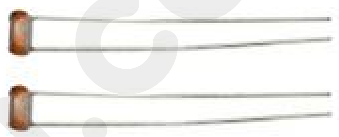


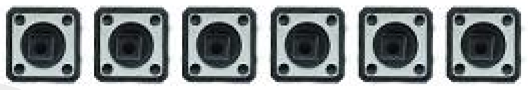
V1.0.19.5.27

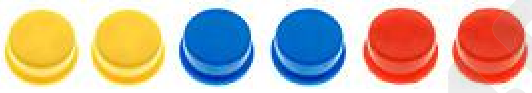
PACKING LIST

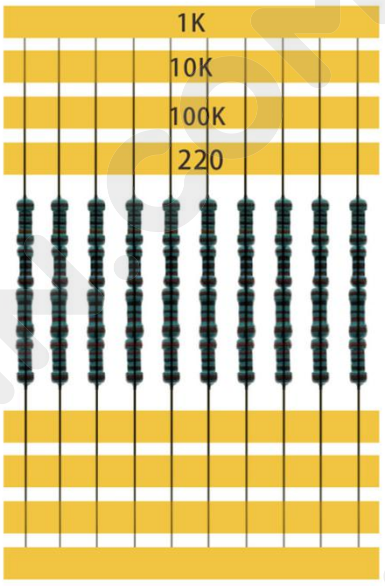












Content

Lesson 1 Installing IDE

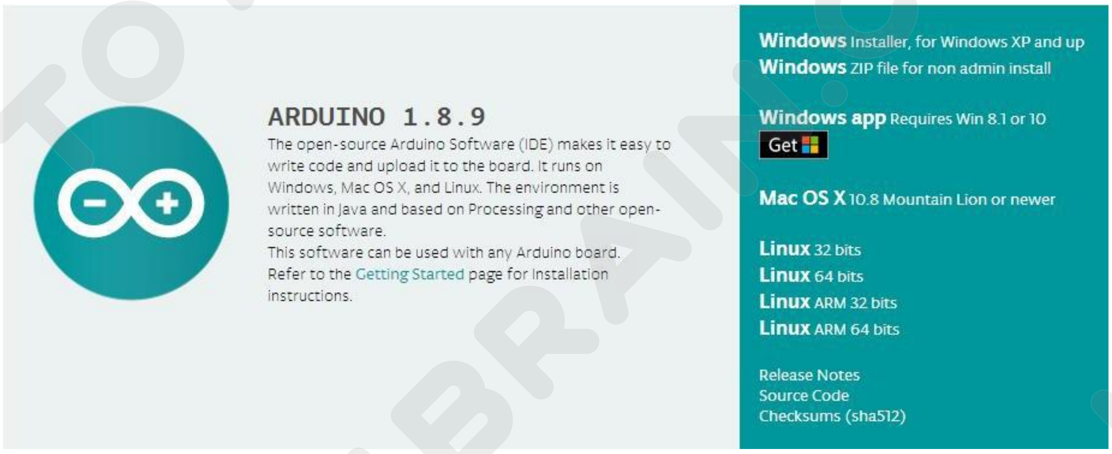
Introduction

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.



ARDUINO 1.8.9

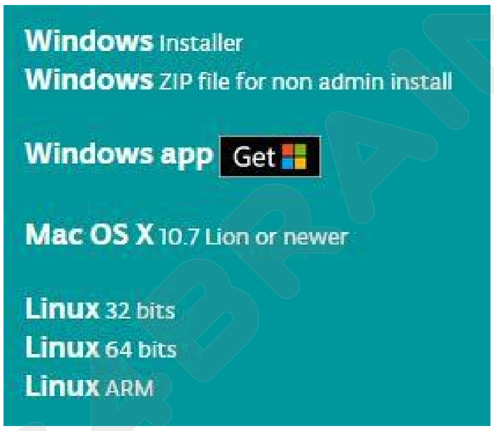
The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.

This software can be used with any Arduino board. Refer to the Getting Started page for installation instructions.

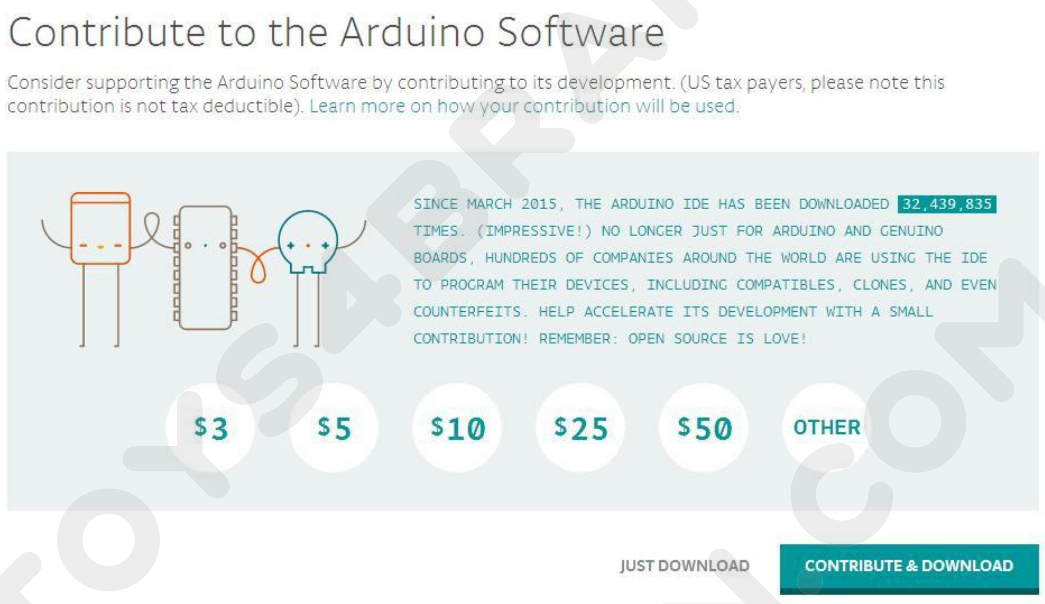
Windows Installer, for Windows XP and up
Windows ZIP file for non admin install
Windows app Requires Win 8.1 or 10
Get
Mac OS X 10.8 Mountain Lion or newer
Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits
Release Notes
Source Code
Checksums (sha512)

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.



Contribute to the Arduino Software

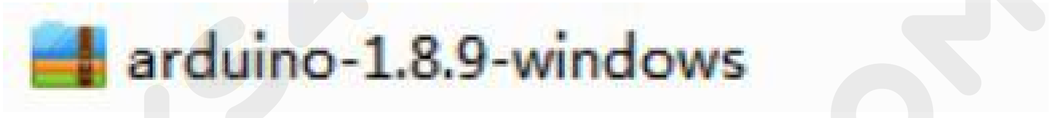
Consider supporting the Arduino Software by contributing to its development. (US tax payers, please note this contribution is not tax deductible). Learn more on how your contribution will be used.

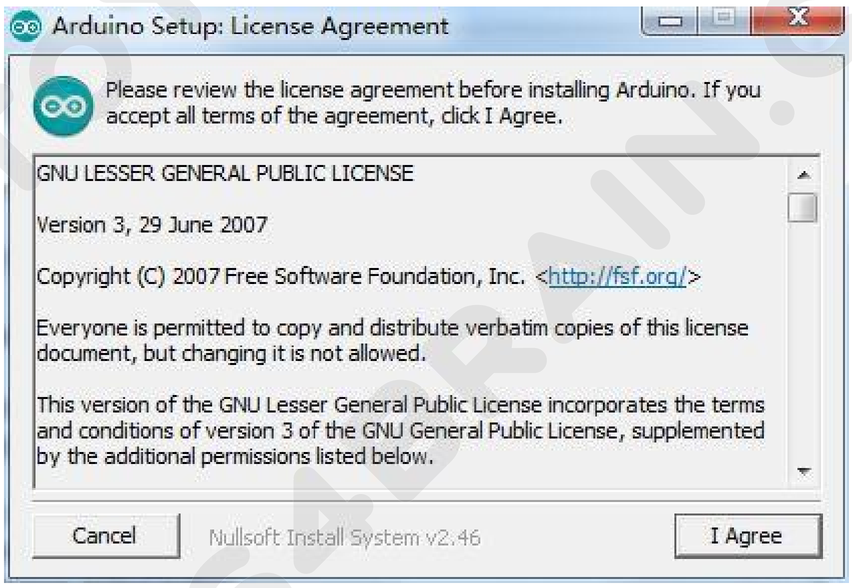
SINCE MARCH 2015, THE ARDUINO IDE HAS BEEN DOWNLOADED 32,439,835 TIMES. (IMPRESSIVE!) NO LONGER JUST FOR ARDUINO AND GENUINO BOARDS, HUNDREDS OF COMPANIES AROUND THE WORLD ARE USING THE IDE TO PROGRAM THEIR DEVICES, INCLUDING COMPATIBLES, CLONES, AND EVEN COUNTERFEITS. HELP ACCELERATE ITS DEVELOPMENT WITH A SMALL CONTRIBUTION! REMEMBER: OPEN SOURCE IS LOVE!

Click JUST DOWNLOAD.

Also version 1.8.9is available in the material we provided, and the versions of our materials are the latest versions when this course was made.

Installing Arduino (Windows)





Arduino Setup: License Agreement

Please review the license agreement before installing Arduino. If you accept all terms of the agreement, click I Agree.

GNU LESSER GENERAL PUBLIC LICENSE

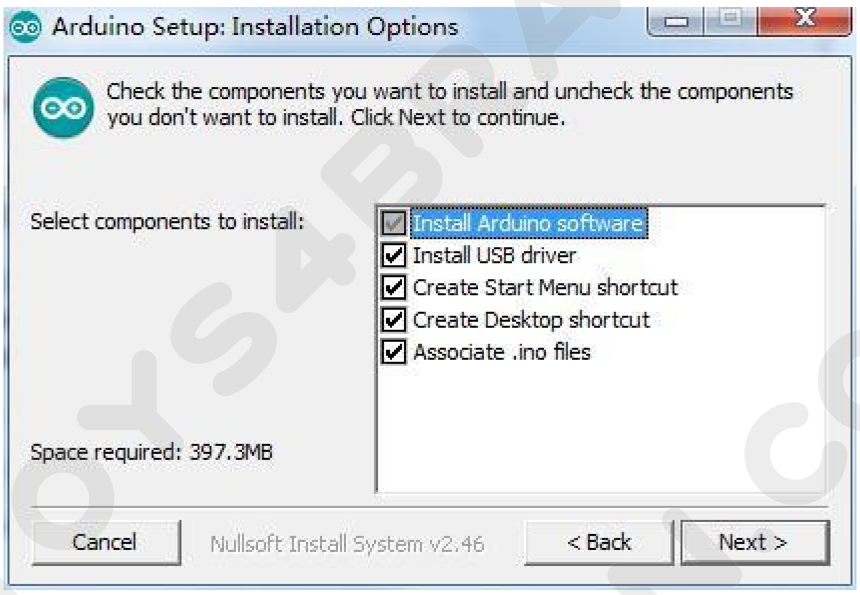
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

Click I Agree to see the following interface


Arduino Setup: Installation Options

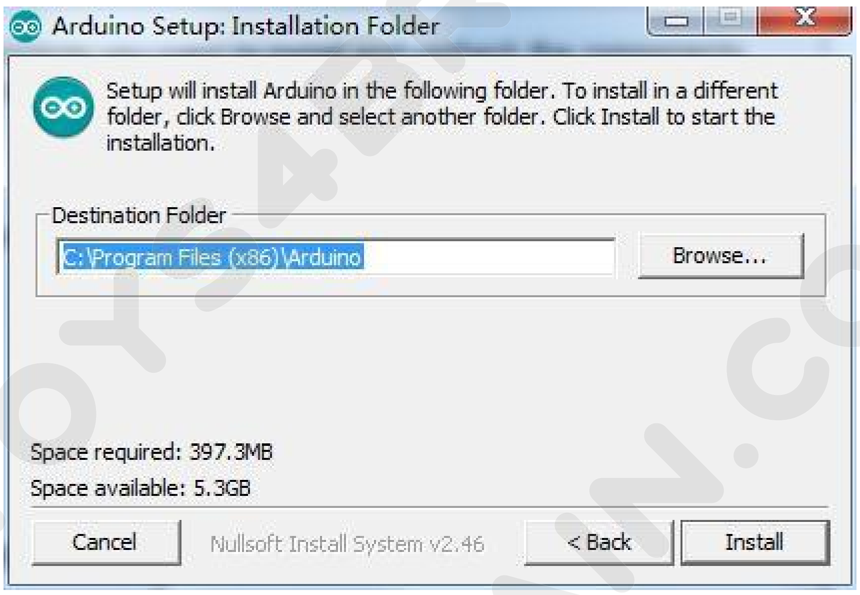
Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:

- [✓] Install Arduino software
- [✓] Install USB driver
- [✓] Create Start Menu shortcut
- [✓] Create Desktop shortcut
- [✓] Associate .ino files

Space required: 397.3MB

Click Next

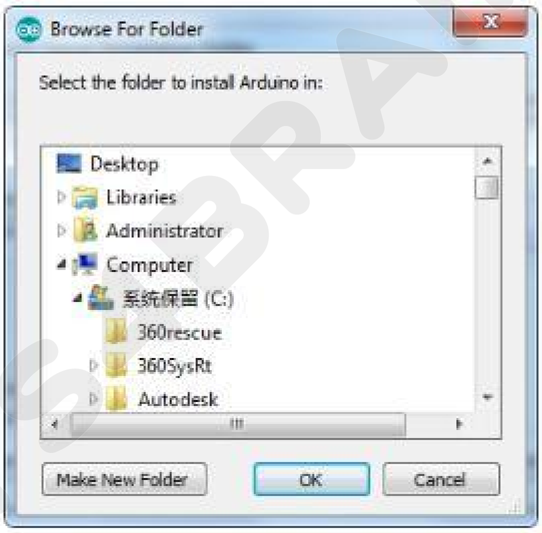

Arduino Setup: Installation Folder

Setup will install Arduino in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

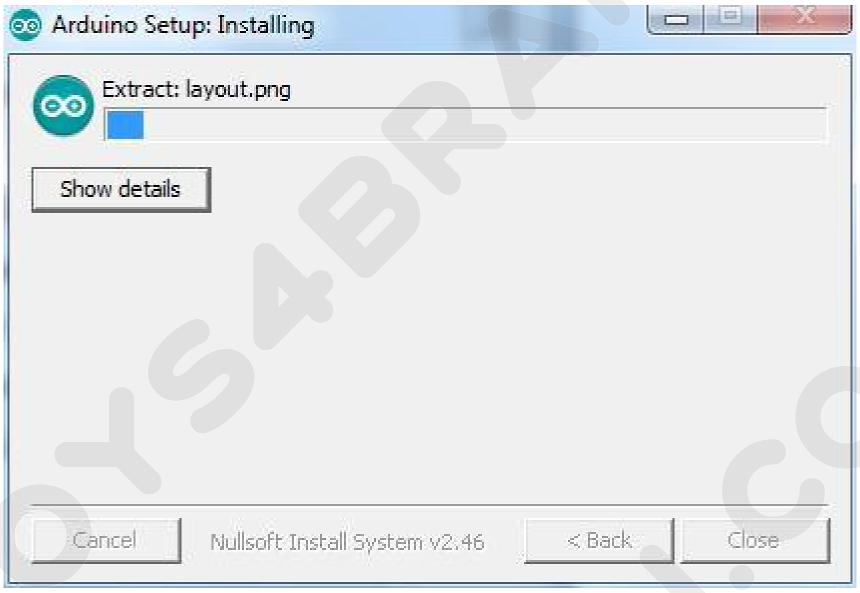
Destination Folder: C:\Program Files (x86)\Arduino

Space required: 397.3MB
Space available: 5.3GB

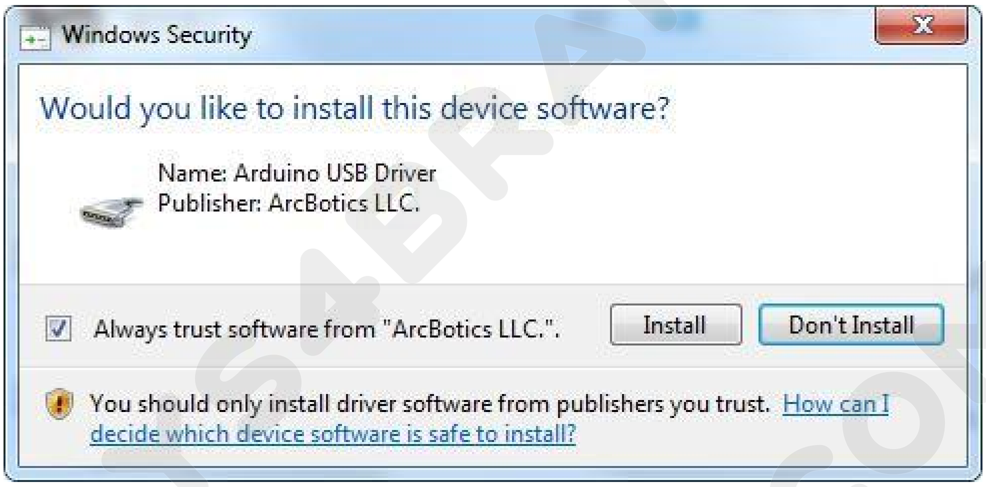
You can press Browse… to choose an installation path or directly type in the directory you want.



Click Install to initiate installation



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



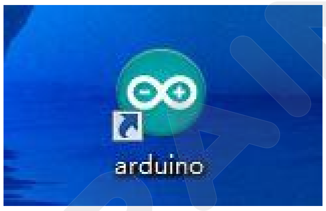
Would you like to install this device software?

Name: Arduino USB Driver
Publisher: ArcBotics LLC.

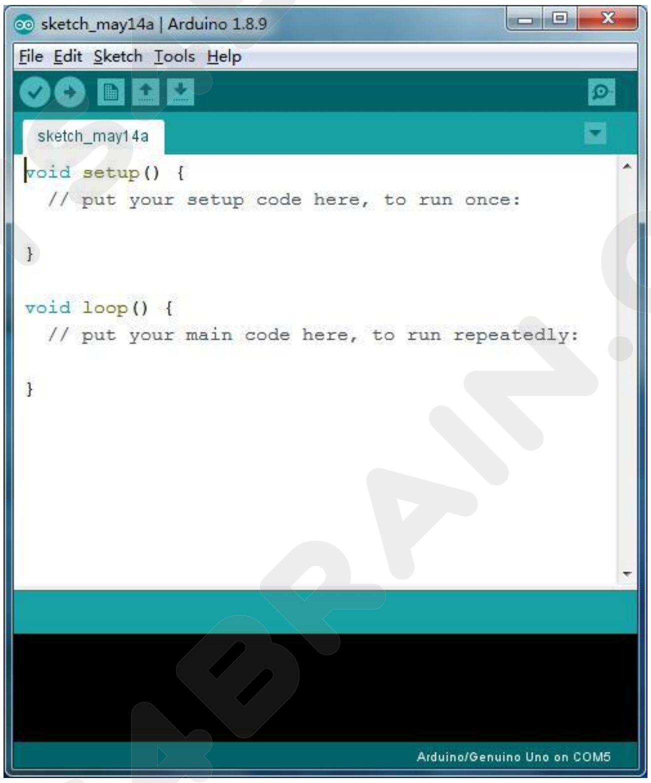
☑ Always trust software from "ArcBotics LLC.".

You should only install driver software from publishers you trust. How can I decide which device software is safe to install?

Next, the following icon appears on the desktop



Double-click to enter the desired development environment


sketch_may14a | Arduino 1.8.9
File Edit Sketch Tools Help
sketch_may14a

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Arduino/Genuino Uno on COM5

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 2 Add 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.

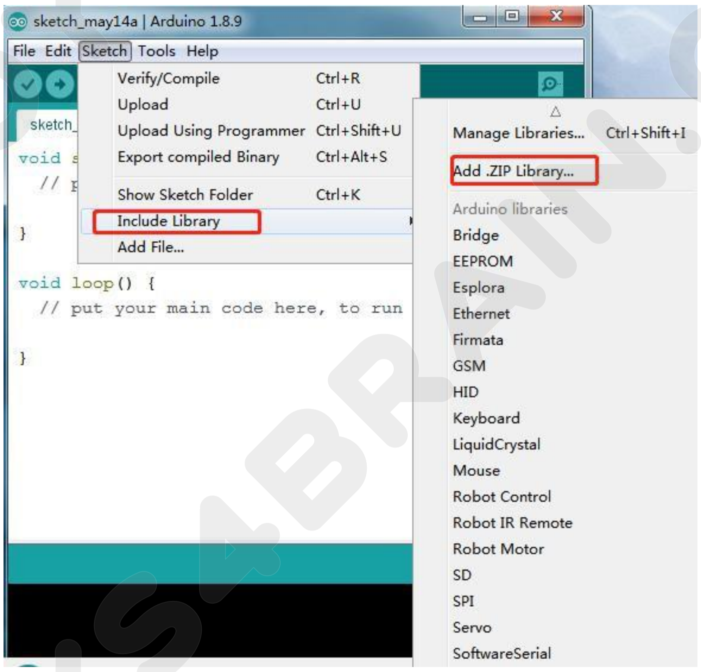
What are Libraries?

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 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.

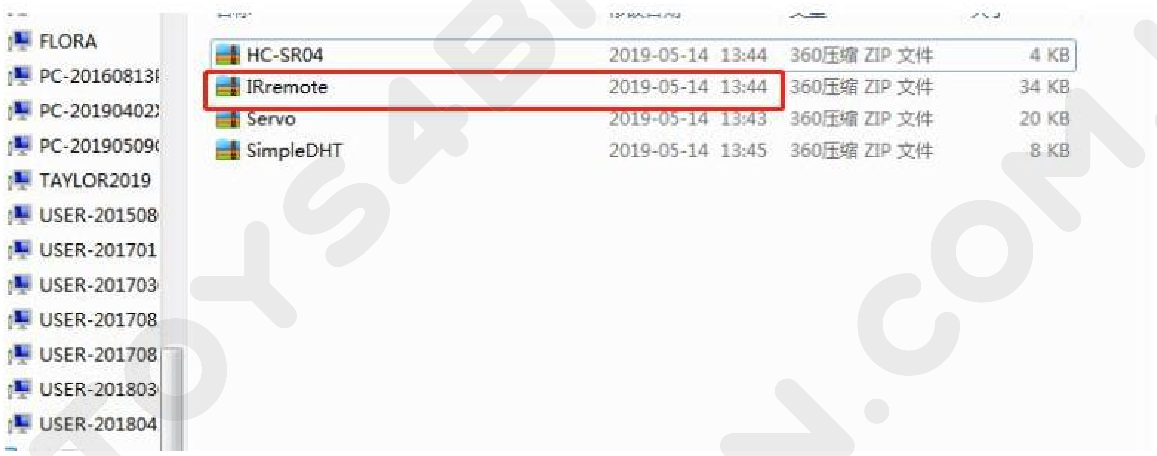
How to Install a Library

Using the Library Manager

To install a new library into your Arduino IDE you can use the Library Manager (available from IDE version 1.8.9). Open the IDE and click to the "Sketch" menu and then Include Library > Manage Libraries.



Then we check to see if the library is installed correctly.



FLORA
PC-20160813I
PC-20190402
PC-20190509
TAYLOR2019
USER-201508
USER-201701
USER-201703
USER-201708
USER-201708
USER-201803
USER-201804

HC-SR04	2019-05-14 13:44	360压缩 ZIP 文件	4 KB
IRremote	2019-05-14 13:44	360压缩 ZIP 文件	34 KB
Servo	2019-05-14 13:43	360压缩 ZIP 文件	20 KB
SimpleDHT	2019-05-14 13:45	360压缩 ZIP 文件	8 KB

Example: IRromote

Open arduino software - project - load library - add a .zip library

Add method two:

Copy the library folder to the Libraries folder in the Arduino installation directory. Restart Arduino and the added library will take effect.

Lesson 3 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.

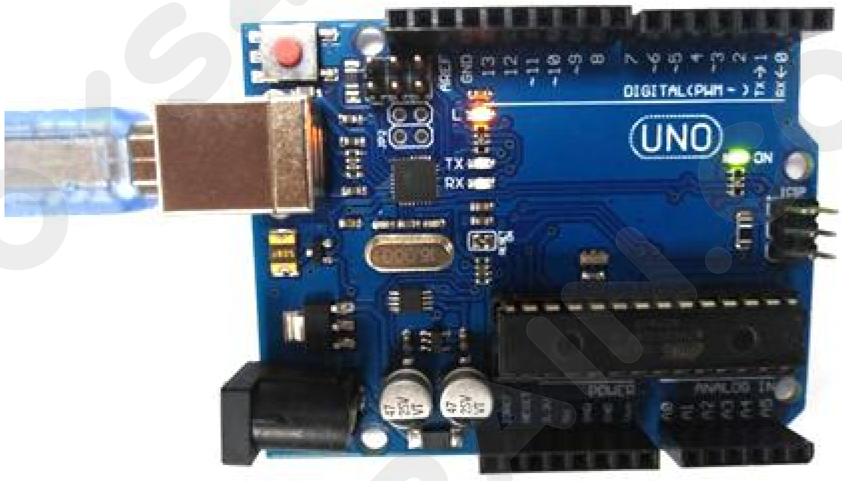
Component Required:

1x Arduino UNO R3

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.

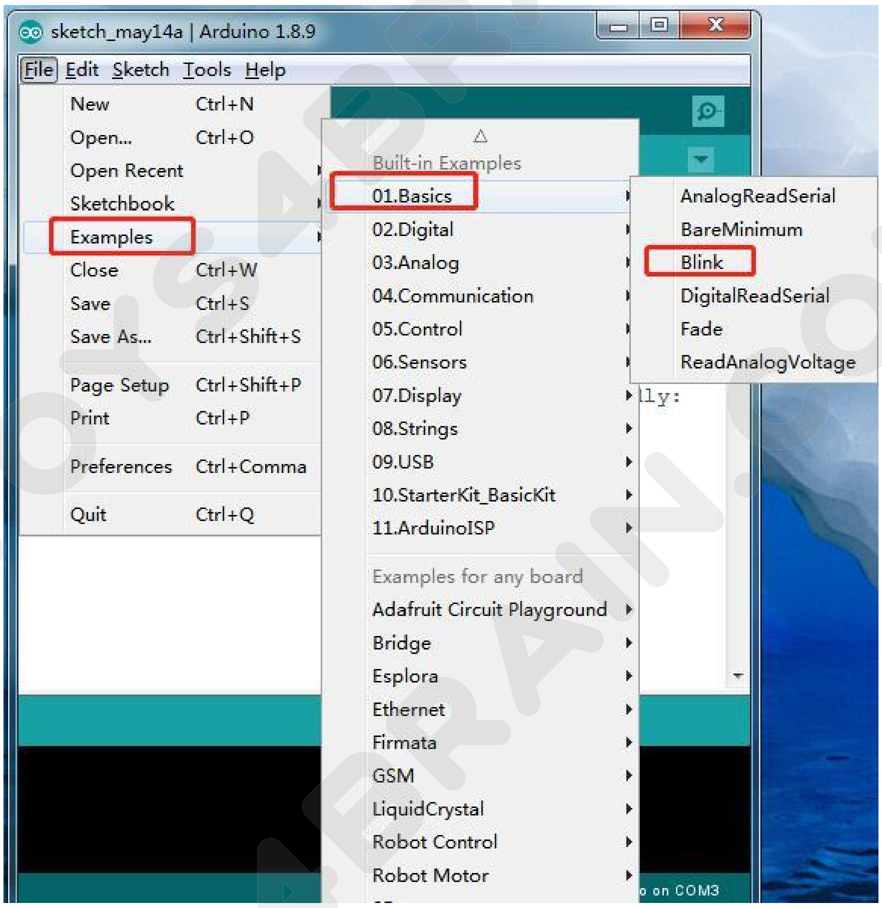
In this lesson, we will reprogram the UNO R3 board with our own Blink sketch and then change the rate at which it blinks.

In Lesson 1, 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.

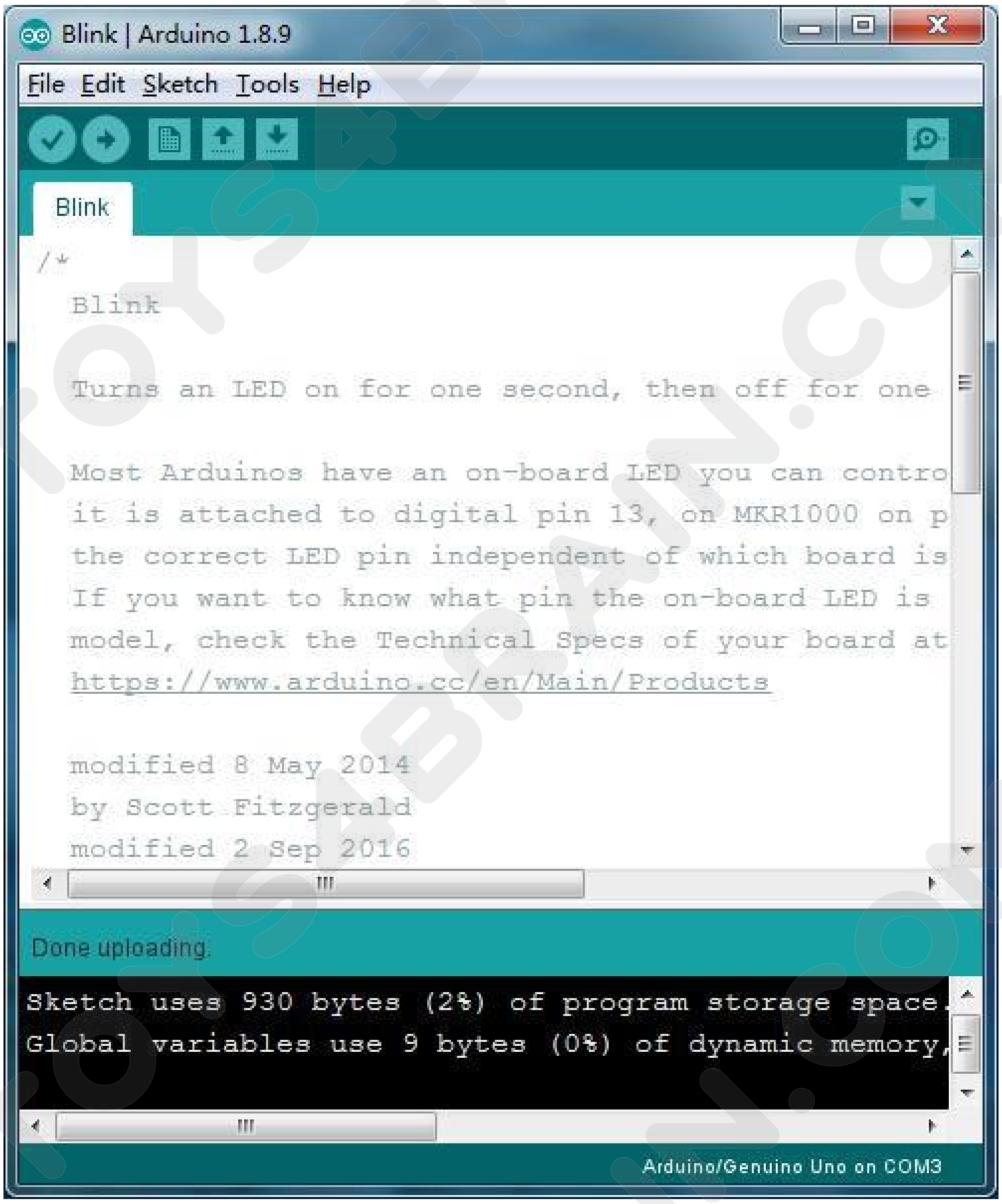
The Arduino IDE includes a large collection of example sketches that you can load up and use. This includes an example sketch for making the 'L' LED blink.

Load the 'Blink' sketch that you will find in the IDE's menu system under

File > Examples > 01.Basics



Open as shown below:



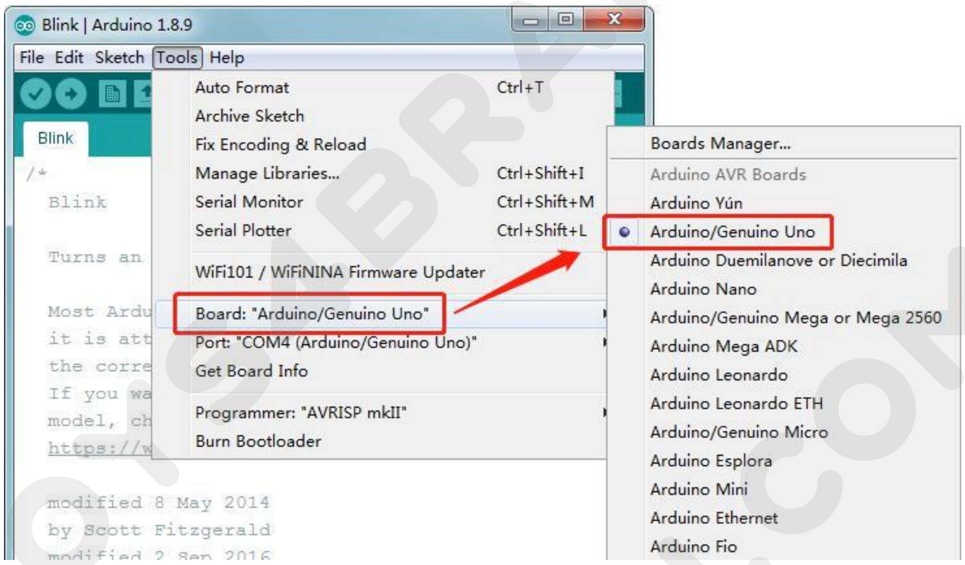
```
/*
  Blink

  Turns an LED on for one second, then off for one

  Most Arduinos have an on-board LED you can contro
  it is attached to digital pin 13, on MKR1000 on p
  the correct LED pin independent of which board is
  If you want to know what pin the on-board LED is
  model, check the Technical Specs of your board at
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
```

Done uploading.

```
Sketch uses 930 bytes (2%) of program storage space.
Global variables use 9 bytes (0%) of dynamic memory,
```

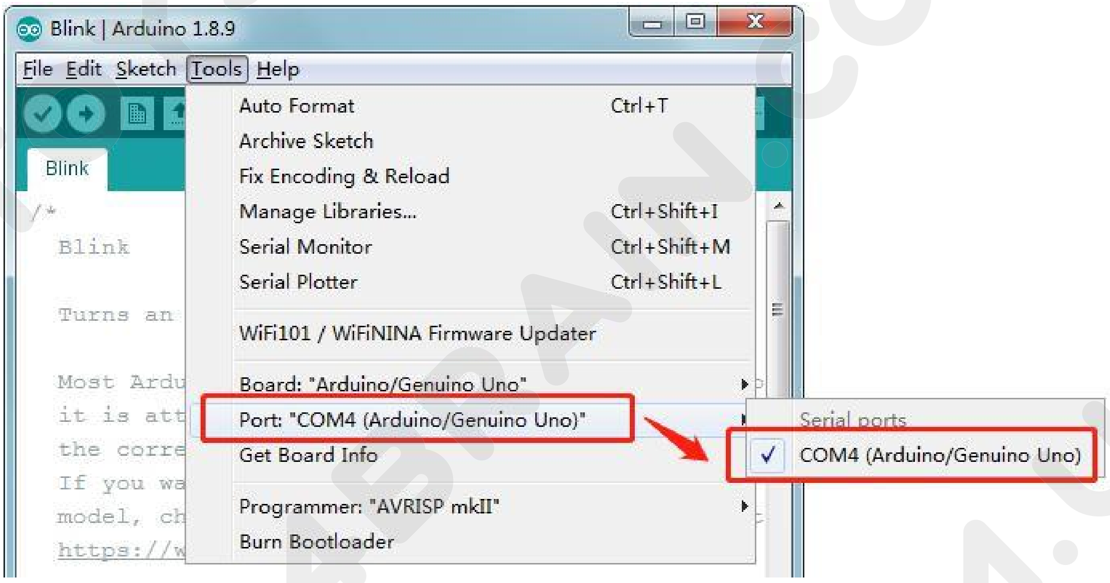
Arduino/Genuino Uno on COM3

Upload code needs to select the board model Arduino/Genuino Uno



The upload code also needs to select the board port number. Our computer displays the port number of arduino as: "COM4 Arduio / Genuino Uno", you may be the other serial port number.

Note: A correct COM port should be COMX (arduino) XXX), which is certified by the standard.


Blink | Arduino 1.8.9
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino/Genuino Uno"
Port: "COM4 (Arduino/Genuino Uno)"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
✓ COM4 (Arduino/Genuino Uno)

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);   // turn the LED on (HIGH is the voltage level) delay(1000);

// wait for a second

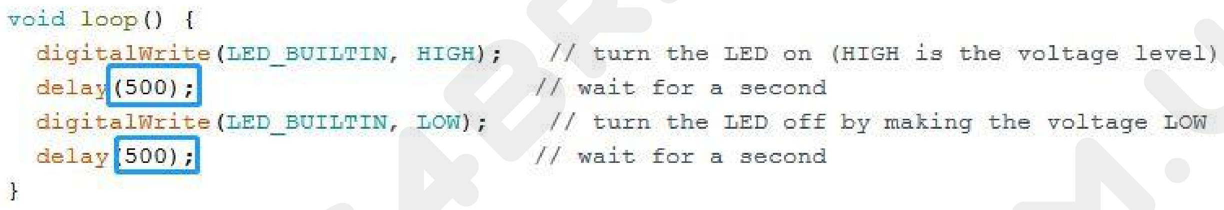
digitalWrite(led, LOW);   // turn the LED off by making the voltage LOW

delay(1000);   // 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.

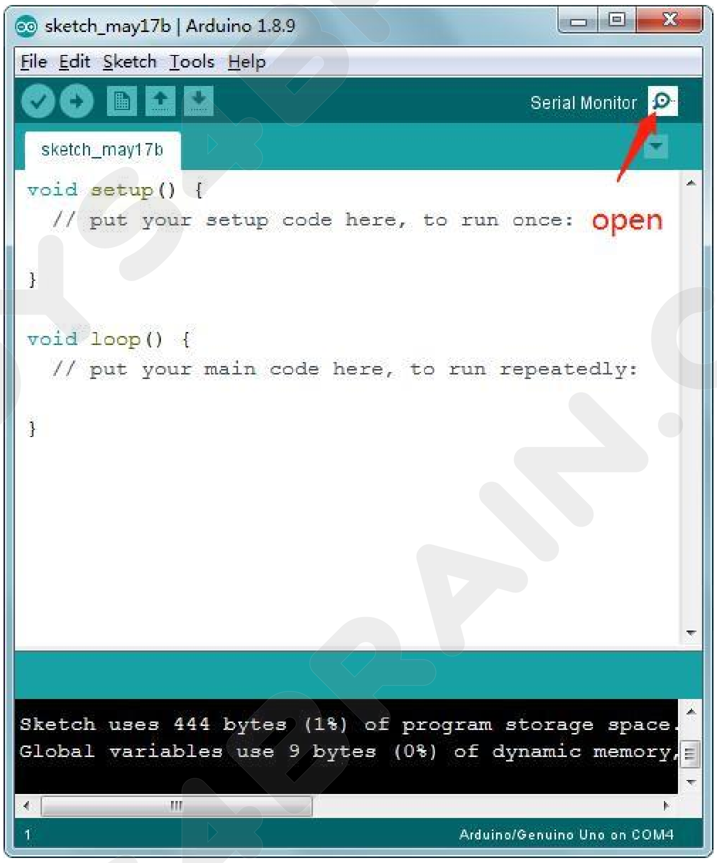


```
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(500);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(500);                       // wait for a second
}
```

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 eachdelay rather than a whole second.

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

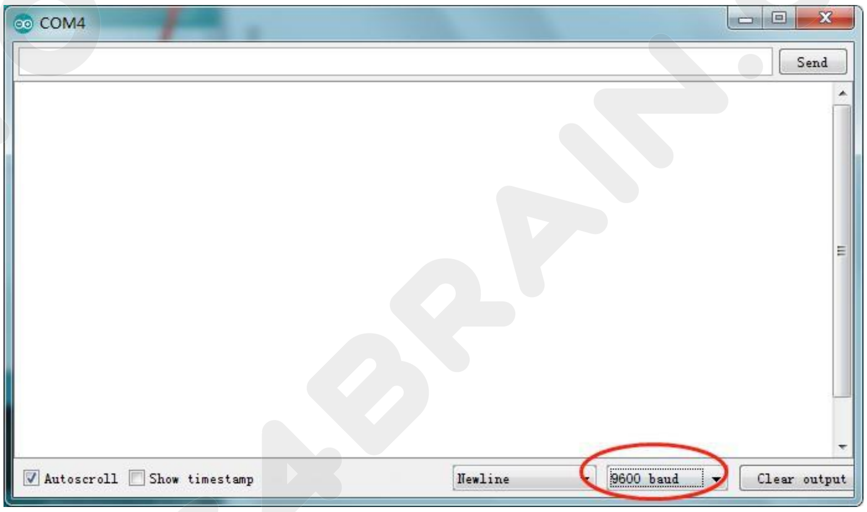
Note: I want to add some information about how to open the serial monitor here.


sketch_may17b | Arduino 1.8.9
File Edit Sketch Tools Help
Serial Monitor
sketch_may17b
open
1
Arduino/Genuino Uno on COM4

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

```
Sketch uses 444 bytes (1%) of program storage space.
Global variables use 9 bytes (0%) of dynamic memory,
```

Serial port is generally selected 9600HZ



Lesson 4 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 Arduino Uno R3

1 x 5mm red LED

1 x 220 ohm resistor

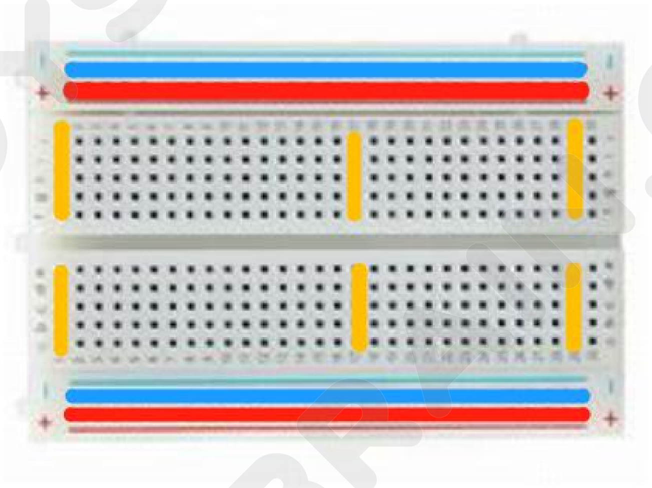
1x 1k ohm resistor

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

Component Introduction:

400 Tie Points:

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.

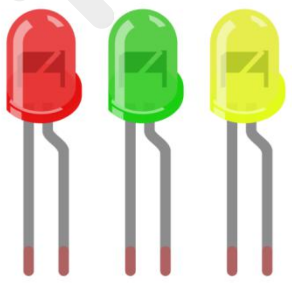
While breadboards are great for prototyping, they have some limitations. Because the connections are push-fit and temporary, they are not as reliable as soldered connections. If you are having intermittent problems with a circuit, it could be due to a poor connection on a breadboard.

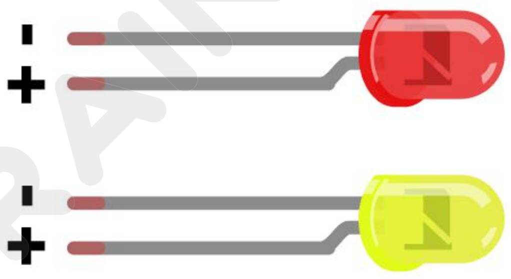
LED:

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.

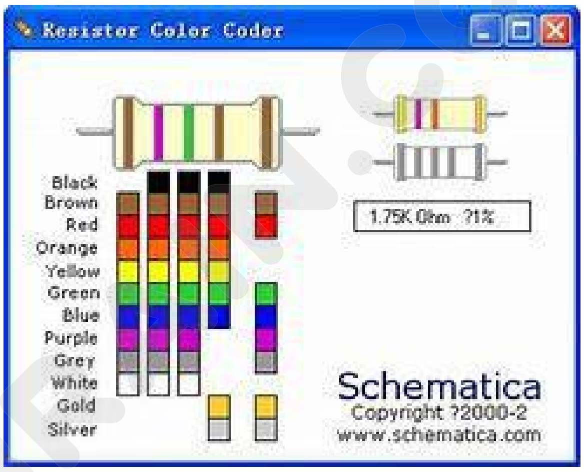
Firstly, the positive lead is longer.

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.

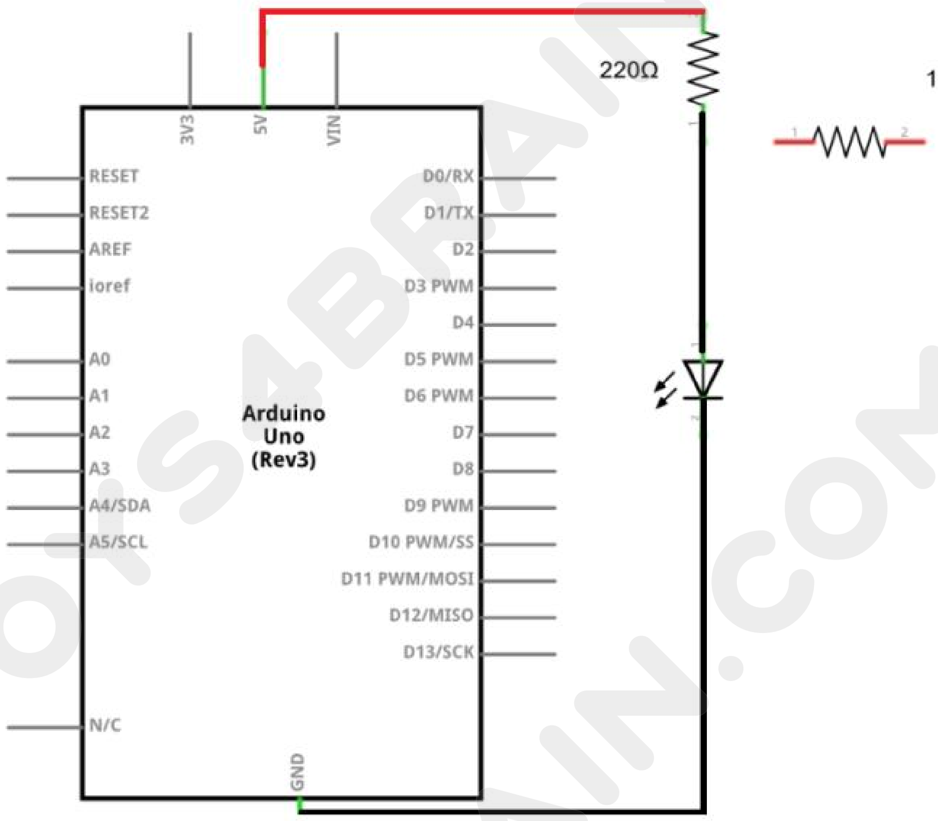
RESISTORS:

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.

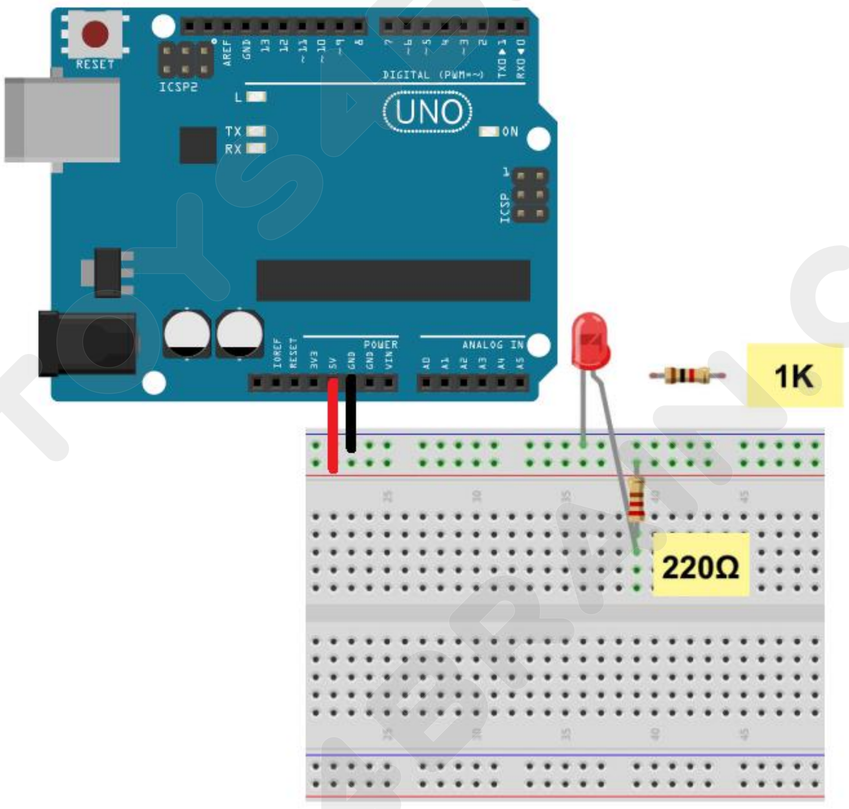


Unlike LEDs, resistors do not have a positive and negative lead. They can be connected either way around.
If you find this approach method t oo complicated, you can read the color ring flag on our resistors directly to determine its resistance value. Or you may use a digital multimeter instead.

Connection Diagram:



Wiring schematic:



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.

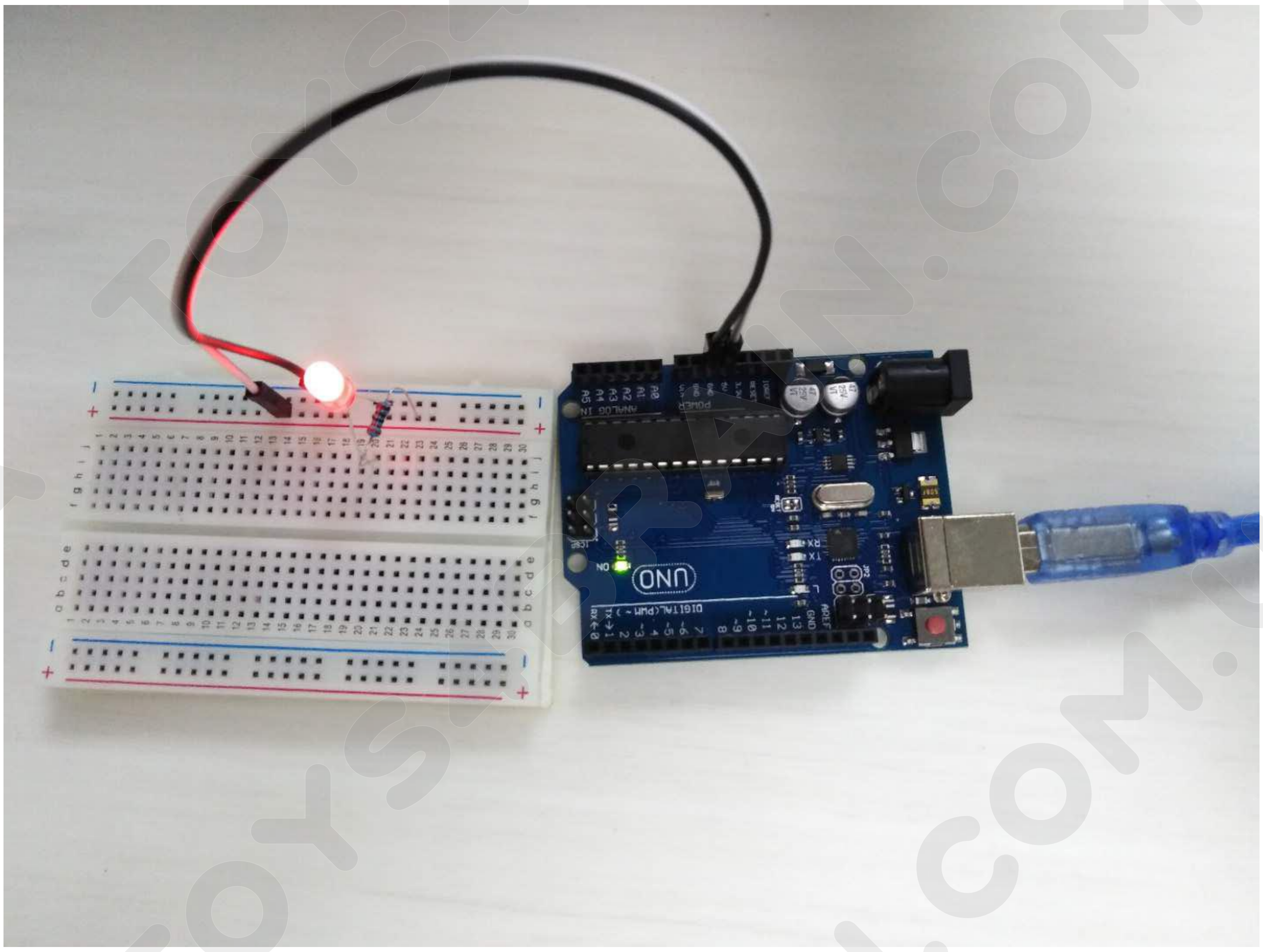
With the 220 Ω resistor in place, the LED should be quite bright. If you swap out the 220 Ω resistor for the 1kΩ resistor, then the LED will appear a little dimmer. Finally, with the 10 kΩ resistor in place, the LED will be just about visible. Pull the red jumper lead out of the breadboard and touch 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

Physical wiring diagram:



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 ArduinoUno R3

1 x 400 Tie-points Breadboard

1 x 5mm red LED

1 x 220 ohm resistor

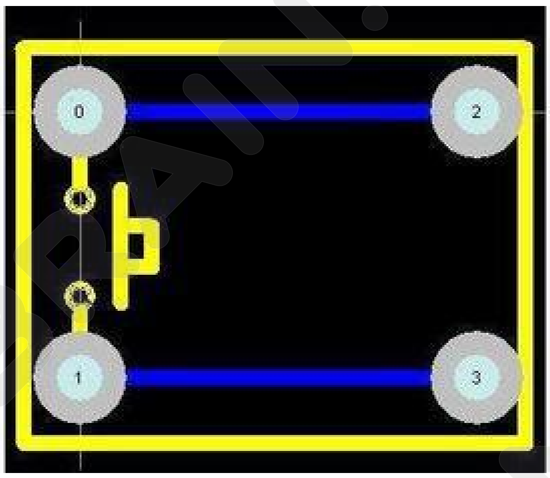
2 x push switches

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

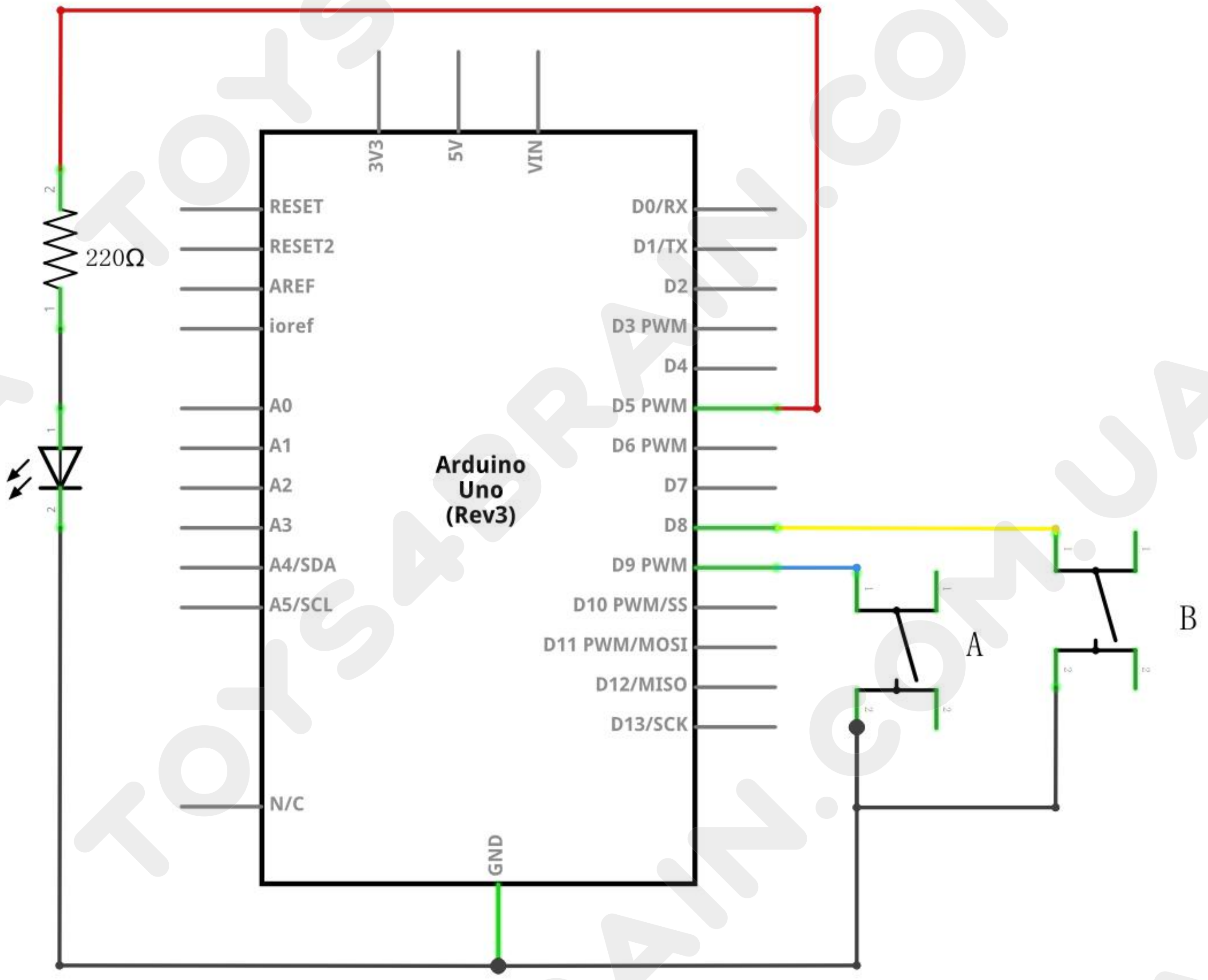
Component Introduction:

Button:

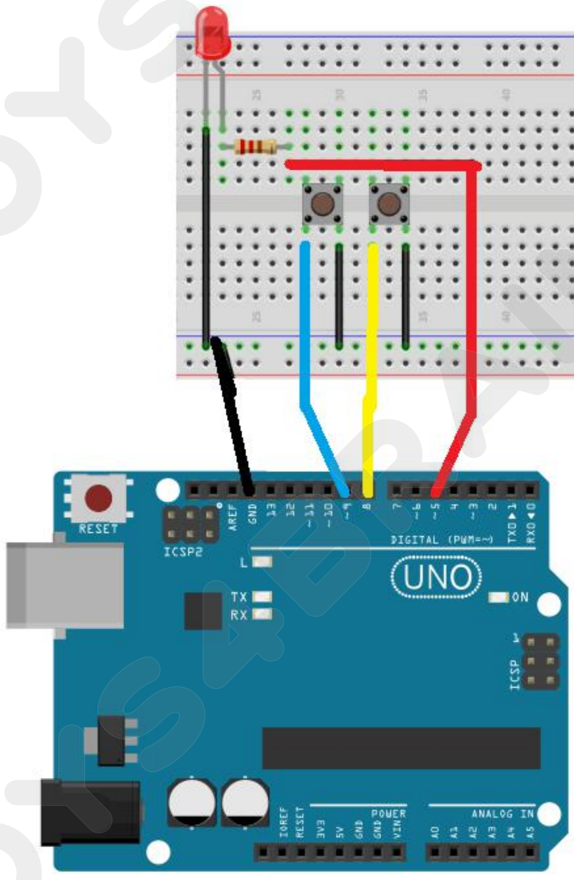
The button as an I/O port input device has seen the effect of the last use. It is actually used to control the on/off of the line, and the effect of changing the I/O port signal is achieved by adding a device such as a pull-up resistor.



Connection Diagram:



Wiring schematic:



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:

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)
```

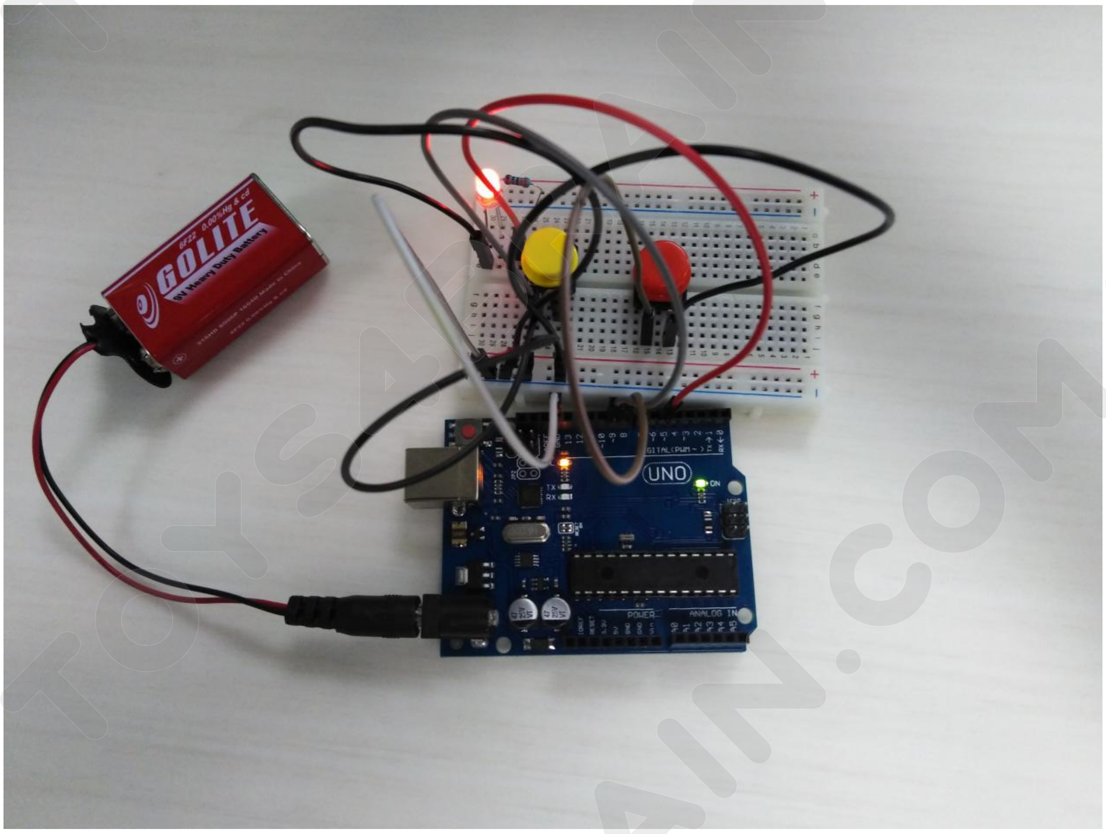
```
    {
      digitalWrite(ledPin, LOW);
    }
 }
```

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.

Physical wiring diagram:



Lesson 6 Photocell Experiment

Overview

A new component-photoresistance will be introduced in this lesson. It can be seen from the name, this component is dependent on the effect of light. In a dark environment, the photoresistance has a very high resistance resistor. More stronger the light, more lower the resistance. By reading this resistance value, you can check the light and darkness of the light.

Component Required:

1 x ArduinoUno R3

1 x 400 Tie-points Breadboard

1 x 5mm red LED

1 x 220 ohm resistor

1 x 10K ohm resistor

1 x Photoresistor (Photocell)

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

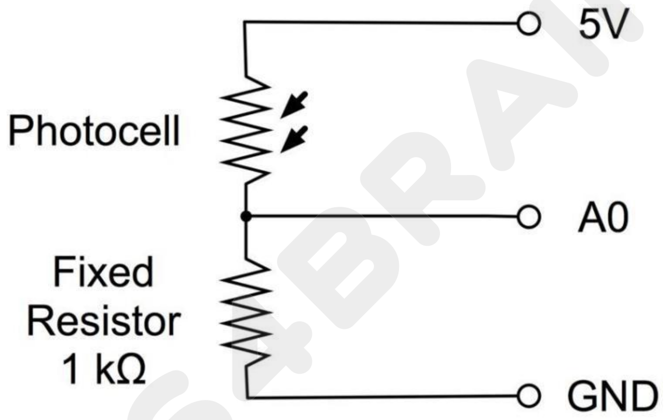
Component Introduction:

PHOTOCELL:

The photocell used is of a type called a light dependent resistor, sometimes called an LDR. As the name suggests, these components act just like a resistor, except that the resistance changes in response to how much light is falling on them.

This one has a resistance of about 50 kΩ in near darkness and 500 Ω in bright light. To convert this varying value of resistance into something we can measure on an UNO R3 board's analog input, it needs to be converted into a voltage.

The simplest way to do that is to combine it with a fixed resistor.

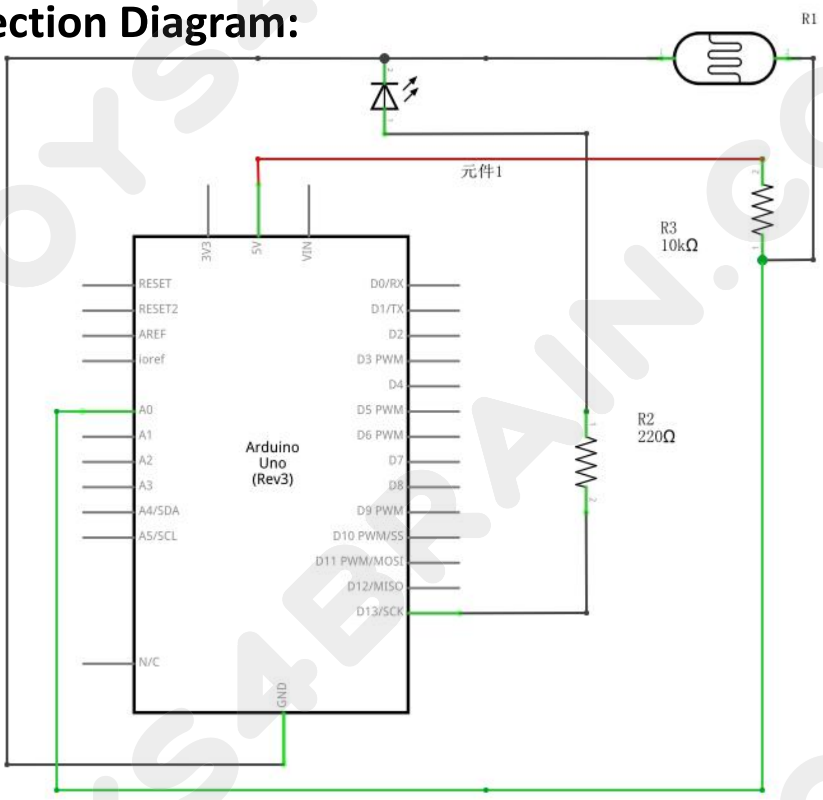


The resistor and photocell together behave like a pot. When the light is very bright, then the resistance of the photocell is very low compared with the fixed value resistor, and so it is as if the pot were turned to maximum.

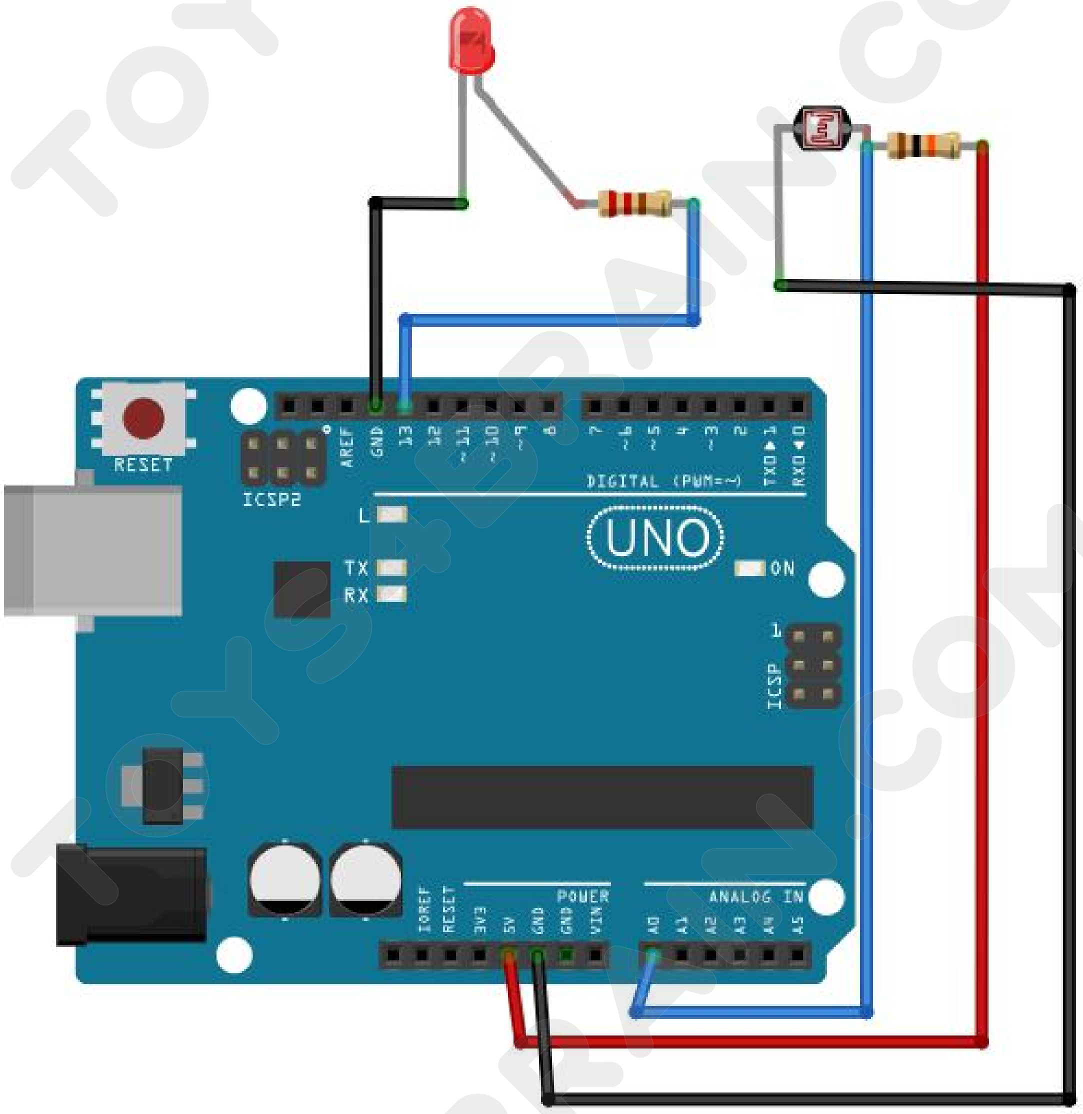
When the photocell is in dull light, the resistance becomes greater than the fixed 1 kΩ resistor and it is as if the pot were being turned towards GND.

Load up the sketch given in the next section and try covering the photocell with your finger, and then holding it near a light source.

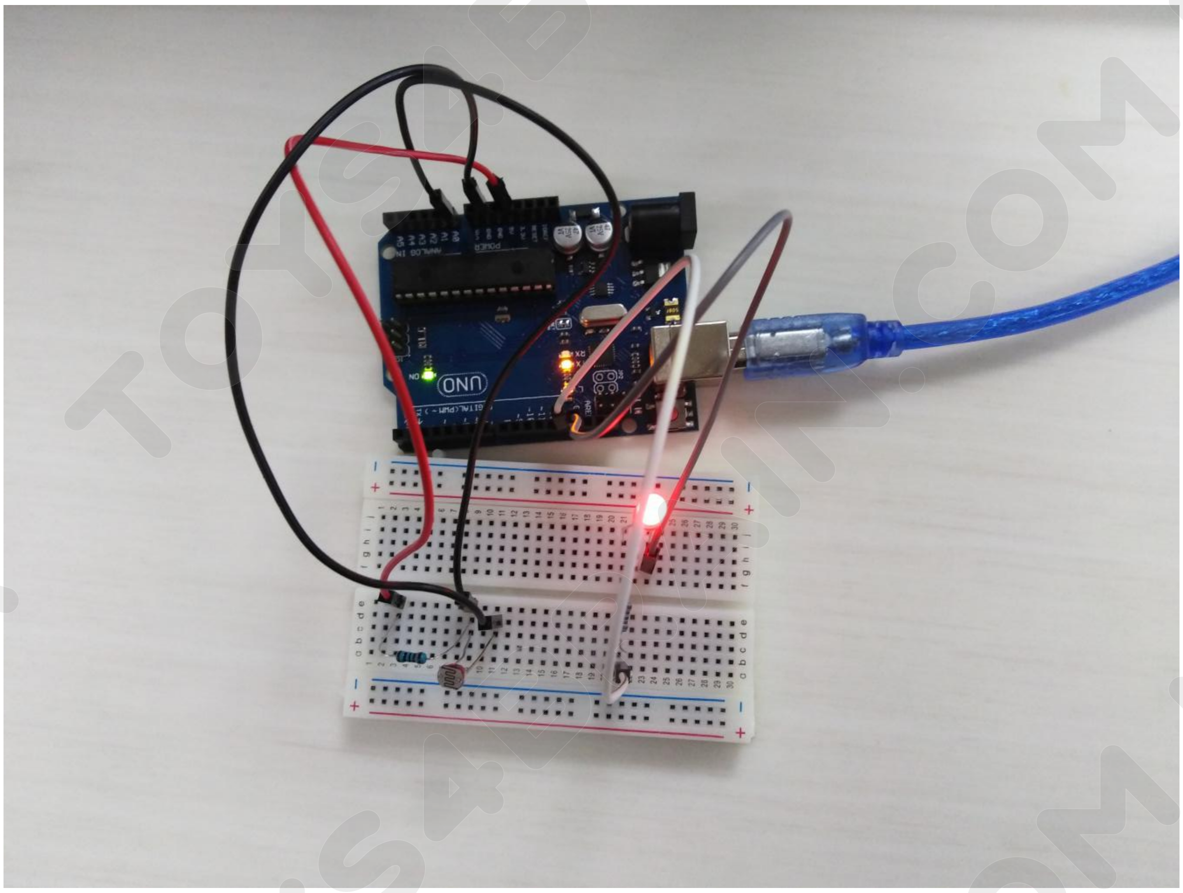
Connection Diagram:



Wiring schematic:



Physical wiring diagram:



Code:

```
int LED = 13;                              //Set the LED light to digital pin 13
int val = 0;                                   //Set analog pin 0 to read the photodiode
voltage

void setup(){
        pinMode(LED,OUTPUT);               // LED for output mode
        Serial.begin(9600);           // The serial port baud rate is set to 9600
}
```

```
void loop(){
        val = analogRead(0);            // Read voltage value 0~1023
        Serial.println(val);            // Serial port to view voltage changes
        if(val<120){                    // Once the value is less than the set value,
the LED light is turned off
                digitalWrite(LED,LOW);
        }else{
                digitalWrite(LED,HIGH);
        }
        delay(10);                      // Delay 10 ms
}
```

For the Arduino syntax, you can refer to the following link:

https://www.arduino.cc/reference/en/