Part 1

lesson 1

Packing list

Preface

Our Company

Established in 2011, Elegoo Inc. is a thriving technology company dedicated to open- source hardware research & development, with the integration of production and marketing. Located in Shenzhen, the Silicon Valley of China, we have grown to over 150 employees with a 10,763+ square ft. factory.

Our product lines range from DuPont wires and UNO R3 boards to complete starter kits designed for customers of any level to learn Arduino knowledge. In addition, we also sell products of Raspberry Pi accessories like 2.8’’ TFT touch and STM32. In the future we will devote more energy and investment to 3D printer products and so on. All of our products comply with international quality standards and are greatly appreciated in a variety of different markets throughout the world.

Official Website: http://www.elegoo.com

US Amazon Storefront: http://www.amazon.com/shops/A2WWHQ25ENKVJ1

CA Amazon Storefront: http://www.amazon.ca/shops/A2WWHQ25ENKVJ1

UK Amazon Storefront: http://www.amazon.co.uk/shops/AZF7WYXU5ZANW

DE Amazon Storefront: http://www.amazon.de/shops/AZF7WYXU5ZANW

FR Amazon Storefront: http://www.amazon.fr/shops/AZF7WYXU5ZANW

ES Amazon Storefront: http://www.amazon.es/shops/AZF7WYXU5ZANW

IT Amazon Storefront: http://www.amazon.it/shops/AZF7WYXU5ZANW

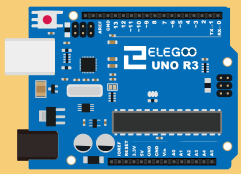
Our Tutorial

This tutorial is designed for beginners. You will learn all the basic information about how to use Arduino controller boards, sensors and components. If you want to study Arduino in more depth, we recommend that you read the Arduino Cookbook written by Michael Margolis.

Some code in this tutorial are edited by Simon Monk. Simon Monk is an author of a number of books relating to Open Source Hardware. They are available in Amazon: Programming Arduino, 30 Arduino Projects for the Evil Genius and Programming the Raspberry Pi.

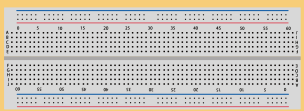
Customer Service

As a continuous and fast growing technology company we keep trying our best to offer you. Excellent products and quality service in order to meet your expectations and you can reach out to us by simply droping a line at service@elegoo.com or EUservice@elegoo.com. We look forward to hearing from you and any of your critical comment or suggestion would be much valuable to us.
Any problems and questions you have with our products will be promptly replied by our experienced engineers within 12 hours (24hrs during holiday)

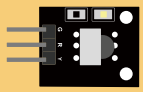


UNO R3 Controller Board
1PC

USB Cable
1PC

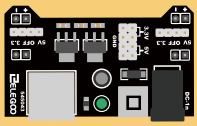


830 Tie-Points Breadboard
1PC



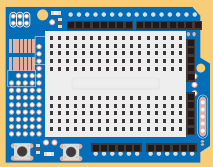
IR Receiver Module
1PC

Servo Motor SG90
1PC

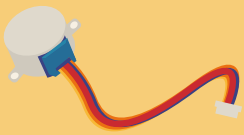


Power Supply Module
1PC

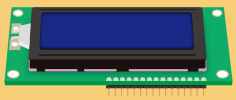
9V Battery with Snap-on
Connector Clip
1PC



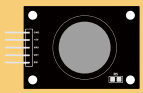
Prototype Expansion Module
1PC



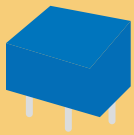
Stepper Motor
1PC



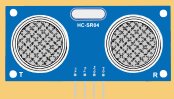
LCD1602 Module
(with pin header)
1PC



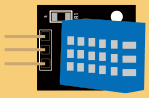
Joystick Module
1PC



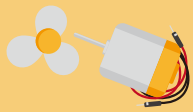
5V Relay
1PC



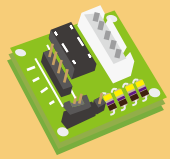
Ultrasonic Sensor
1PC



DHT11 Temperature and
Humidity Module
1PC



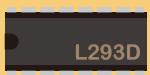
Fan Blade and 3-6V Motor
1PC



ULN2003 Stepper Motor
Driver Module
1PC

1 Digit 7-Segment Display
1PC

4 Digit 7-Segment Display
1PC



L293D
1PC

74HC595 IC
1PC

Button
5PCS

Remote Control
1PC

Active Buzzer
1PC

Passive Buzzer
1PC

Breadboard Jumper Wire
65PCS

Female-to-Male Dupont Wire
10PCS

Tilt Ball Switch
1PC

Resistor
120PCS

Thermistor
1PC

NPN Transistor PN2222
2PCS

Diode Rectifier
2PCS

Potentiometer
1PC

Photoresistor(Photocell)
2PCS

RGB LED
2PCS

White LED
5PCS

Yellow LED
5PCS

Red LED
5PCS

Blue LED
5PCS

Green LED
5PCS

Part 1

lesson 2

First Look

Arduino

1.What is Arduino?

Based on the idea of implementing easy-to-use hardware and software, Arduino is an open-source electronics platform.

Arduino boards are able to read input signals:

- ◯ Acceleration and steering
- ◯ a finger on a button
- ◯ a Twitter message
- ◯ light on a sensor……

and turn it into an output signals:

- ◯ activating a motor
- ◯ turning on an LED
- ◯ publishing something online……

You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Do you want to build a robot that can run, dance and be controlled by you?

Do you want to make a smart bracelet that can not only detect your heart rate and step number but also display time,date and play songs?

Do you think you can control deliveries at home even when you are at work?

Do you want to be a robot that can sweep the floor for you?

Do you want to make a device that can automatically pour coffee for you?

Do you want to achieve all kinds of cool and romantic lighting effects?

Do you want an infinity Gauntlet like the one in the Avenger Alliance?

As long as you want, arduino can help you achieve it.

2.The history of Arduino

- As an open-source project founded by Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, and David Mellis, Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments.

- When it comes to its history,Arduino was initially developed at the Interaction Design Institute Ivrea, in northern Italy. It derives from Wiring, a platform built by Hernando Barragan as his master's thesis at Interaction-Ivrea. Hernando was advised by Massimo and Casey Reas. Wiring and, in turn, Arduino build on previous work by both Massimo and Casey -- Massimo's Programma2003 electronics prototyping platform and the Processing platform by Casey and Ben Fry. Early versions of both Wiring and Arduino also relied upon Pascal Stang' s avrlib libraries.

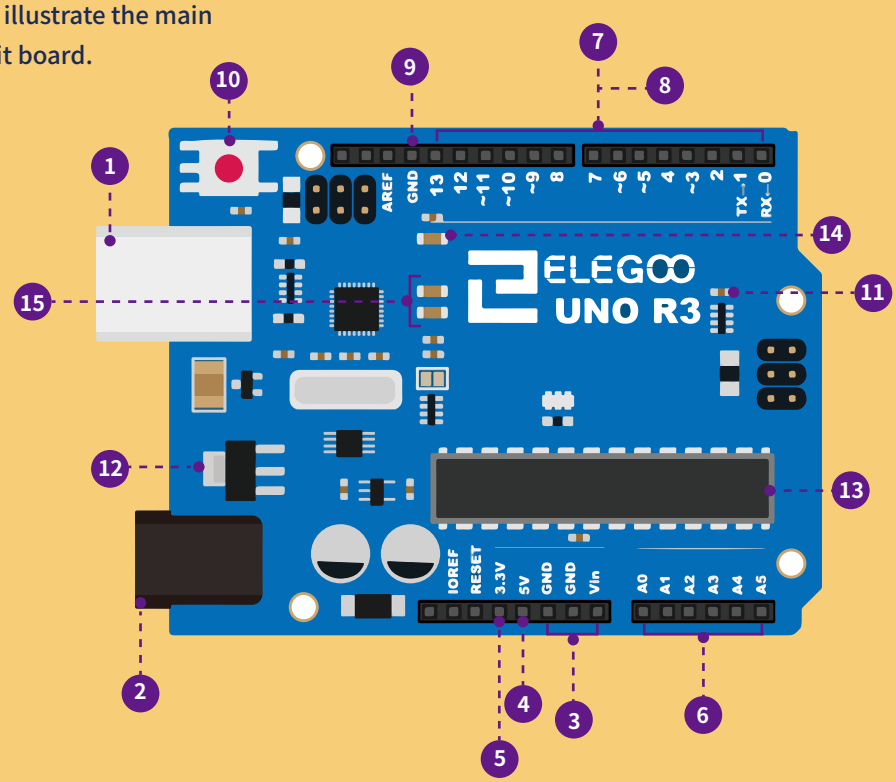
- The initial Arduino boards were designed by Massimo Banzi and David Cuartielles. David Mellis developed the initial Arduino software based on Wiring, with many contributions early on by Nicholas Zambetti. Tom Igoe at ITP in New York was an early adopter and advisor on the Arduino project. Gianluca Martino helped with manufacturing and hardware design.The Arduino project is now supported by an international company, with offices and people around the world.

  Arduino takes its name from a bar in Ivrea, which was named after an early king of Italy from Ivrea.

3.Introduction of board

We use elegoo uno as an example to illustrate the main circuit components of Arduino circuit board.

① the USB connection
② the 6V~12V barrel jack
③ GND
④ 5V pin supplies
⑤ 3.3V pin supplies
⑥ analog pin
⑦ digital pin
⑧ (pwm~)
⑨ GND
⑩ Reset Button
⑪ Power LED Indicator
⑫ Voltage Regulator
⑬ Main IC
⑭ Pin D13 indicator LED
⑮ Serial communication indicator LEDs



Volts (USB / Barrel Jack)

- Every Arduino board needs a way to be connected to a power source. The Arduino UNO can be powered from a USB cable coming from your computer or a wall power supply (like this) that is terminated in a barrel jack. In the picture above the USB connection is labeled (1) and the barrel jack is labeled (2).

- The USB connection is also how you will load code onto your Arduino board. More on how to program with Arduino can be found in our Installing and Programming Arduino tutorial.

NOTE: Do not use a power supply that provide more than 20 Volts as you will overpower (and thereby destroy) your Arduino. The recommended voltage for most Arduino models is between 6 and 12 Volts.

Pins (5V, 3.3V, GND, Analog, Digital, PWM, AREF)

- The pins on your Arduino are your way to connect wires to construct a circuit (probably in conjuction with a breadboard and some wire). They usually have black plastic 'headers' that allow you to just plug a wire right into the board. The Arduino has several different kinds of pins, each of which is labeled on the board and used for different functions.

GND (3): Short for 'Ground' . There are several GND pins on the Arduino, any of which can be used to ground your circuit.

5V (4) & 3.3V (5): As you might guess, the 5V pin supplies 5 volts of power, and the 3.3V pin supplies 3.3 volts of power. Most of the simple components used with the Arduino run happily off of 5 or 3.3 volts.

Analog (6): The area of pins under the 'Analog In' label (A0 through A5 on the UNO) are Analog In pins. These pins can read the signal from an analog sensor (like a temperature sensor) and convert it into a digital value that we can read.

Digital (7): Across from the analog pins are the digital pins (0 through 13 on the UNO). These pins can be used for both digital input (like telling if a button is pushed) and digital output (like powering an LED).

PWM (8): You may have noticed the tilde (~) next to some of the digital pins (3, 5, 6, 9, 10, and 11 on the UNO). These pins act as normal digital pins, but can also be used for something called Pulse-Width Modulation (PWM). We have a tutorial on PWM, but for now, think of these pins as being able to simulate analog output (like fading an LED in and out).

AREF (9): Stands for Analog Reference. Most of the time you can leave this pin alone. It is sometimes used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

If you are confused about the code in the following course, you can hold down Ctrl and click on the following website to refer to the grammar.
https://www.arduino.cc/reference/en/#functions

Part 1

lesson 5

Blink and Add Libraries

Overview

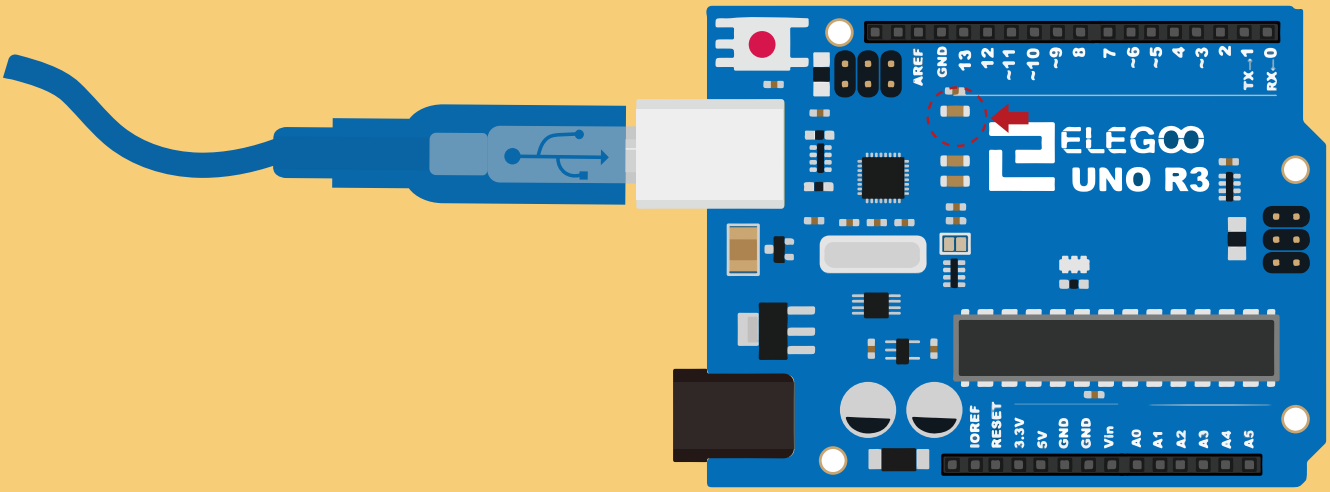
In this lesson, you will learn how to program your UNO R3 controller board to blink the Arduino' s built-in LED, and how to download programs by basic steps.
In addition, we need to learn how to add libraries for which, we can use library functions in future learning to expand Arduino functions more easily.

Component Required

(1) x Elegoo 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 extend 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.



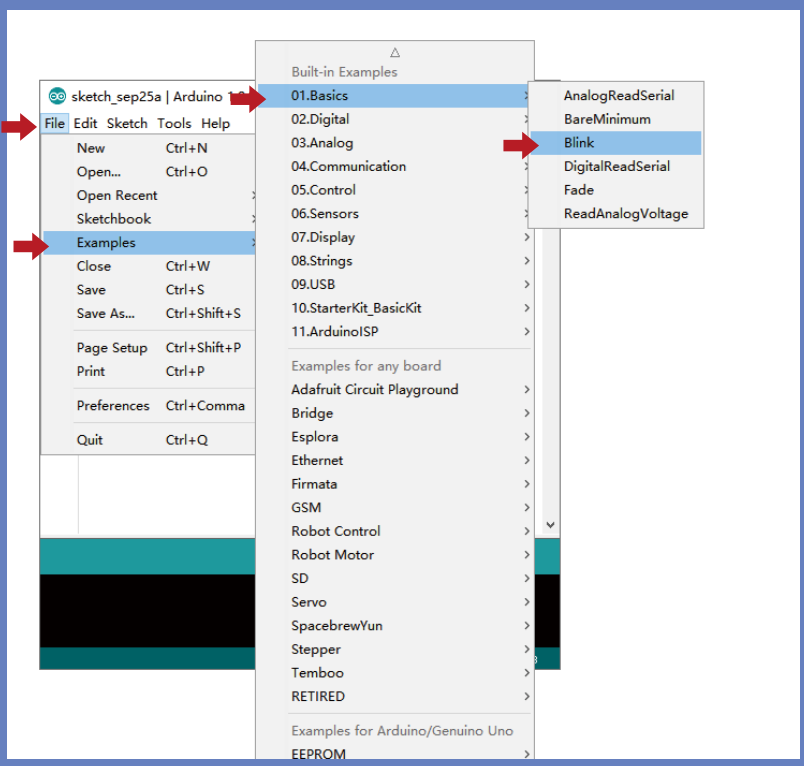
You may find that your UNO R3 board's 'L' LED already blinks when you connect it to a USB plug. This is because the boards are generally shipped with the 'Blink' sketch pre-installed.

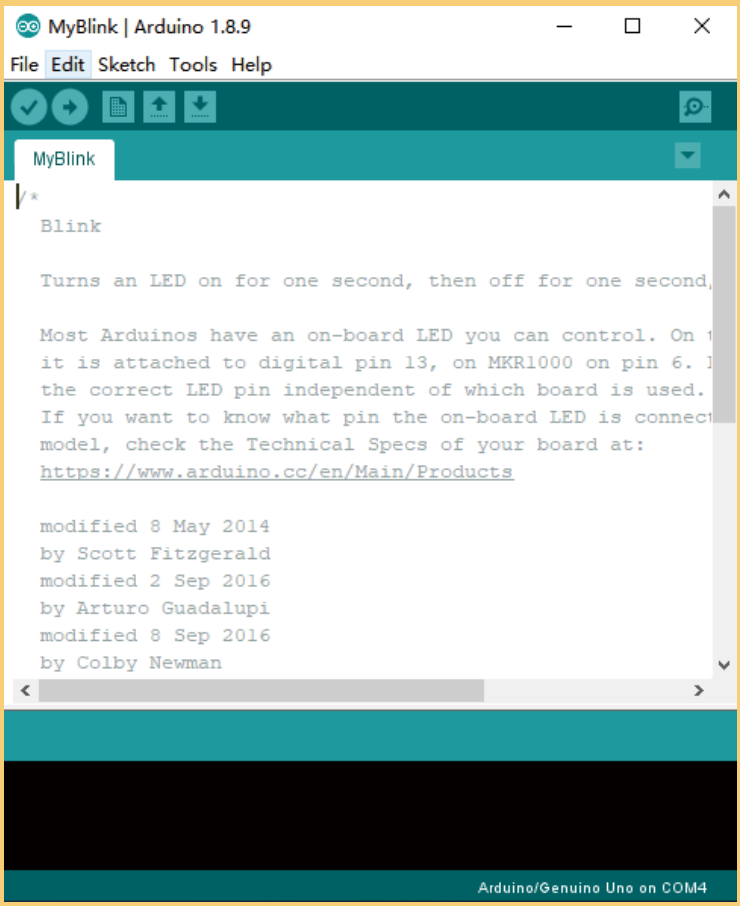
In this lesson, we will reprogram the UNO R3 board with our own Blink sketch and then change the rate at which it blinks.

In Lesson , 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.

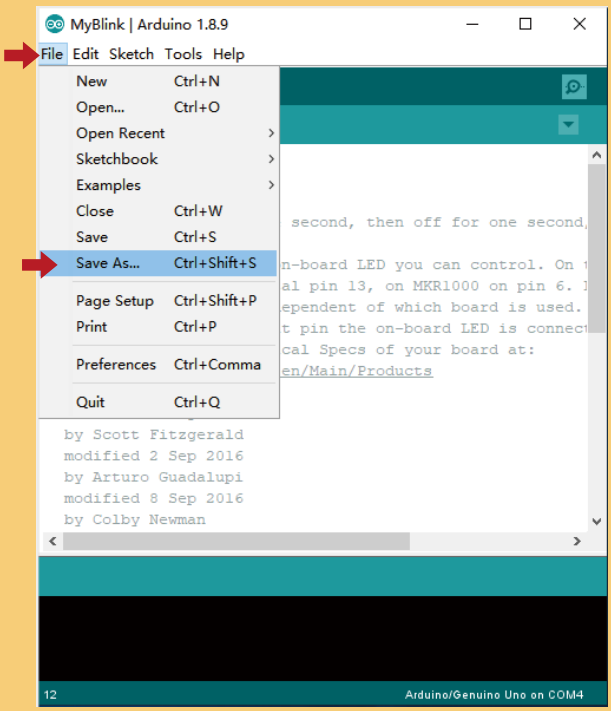


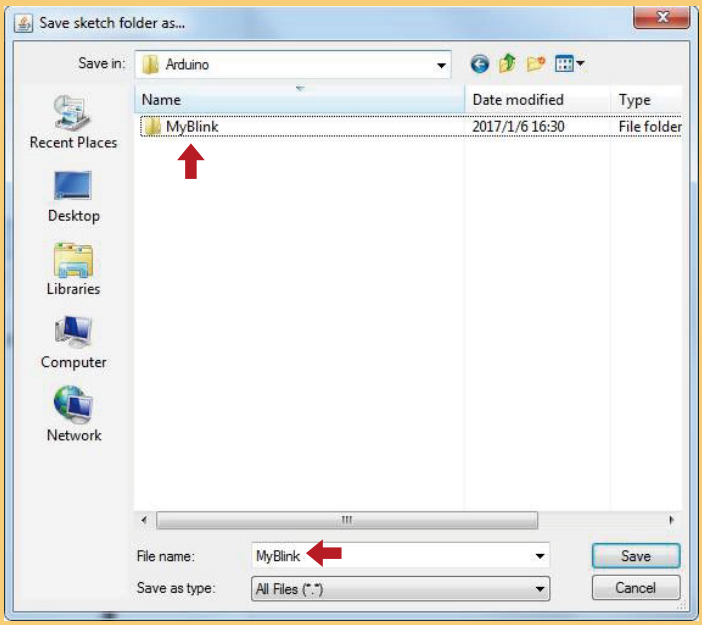


When the sketch window opens, enlarge it so that you can see the entire sketch in the window.

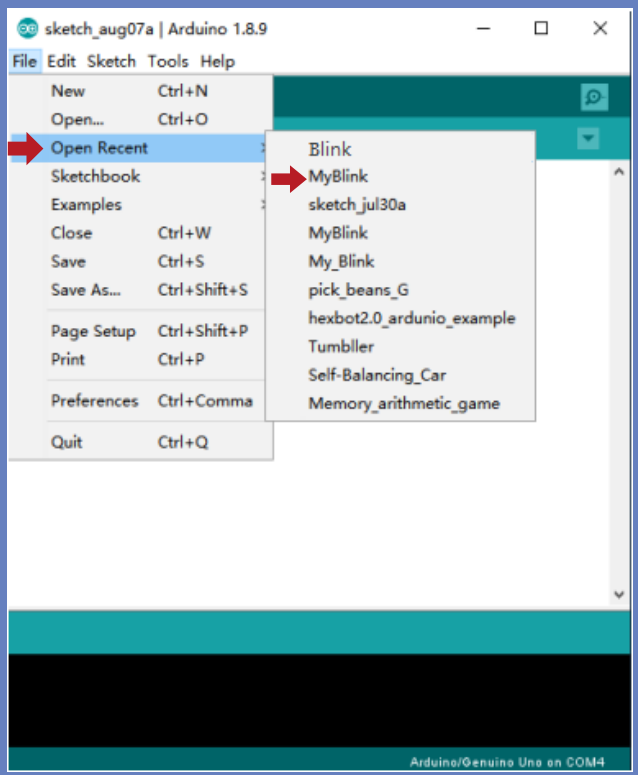
The example sketches included with the Arduino IDE are 'read-only'.
That is, you can upload them to an UNO R3 board, but if you change them, you cannot save them as the same file.

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'.

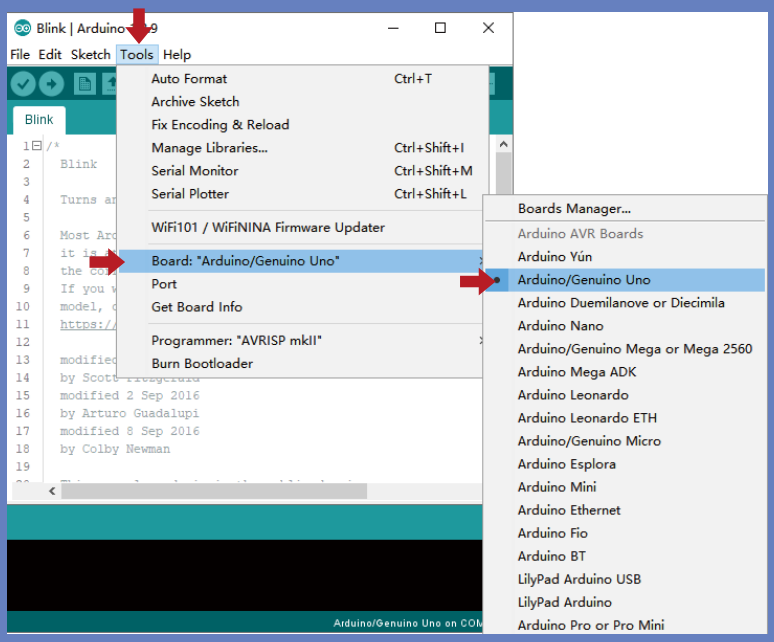


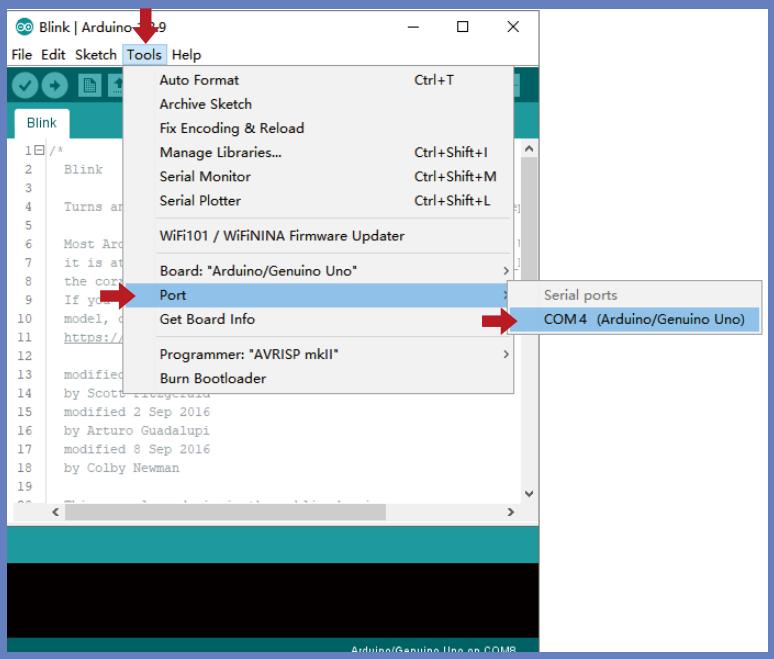


You have saved your copy of 'Blink' in your sketchbook. This means that if you ever want to find it again, you can just open it using the File -> Open Recent.



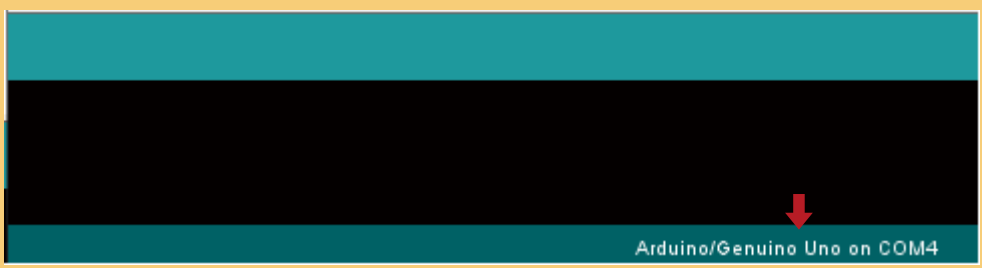
Attach your Arduino board to your computer with the USB cable and check that the 'Board Type' and 'Serial Port' are set correctly.





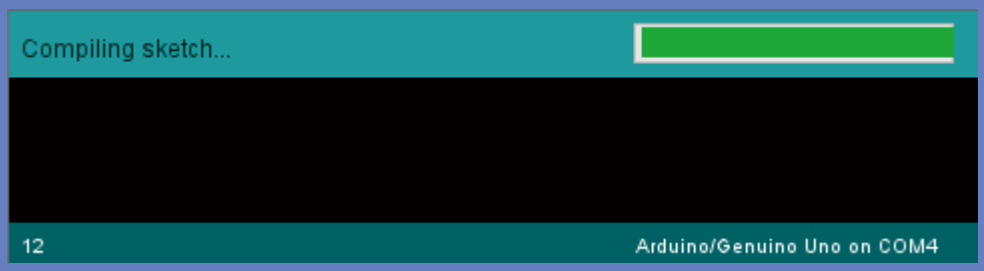
Note: The Board Type and Serial Port here are not necessarily the same as shown in picture.
If you are using 2560, then you will have to choose Mega 2560 as the Board Type, other choices can be made in the same manner.
And the Serial Port displayed for everyone is different, despite COM 4 chosen here, it could be COM3 or COM5 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.

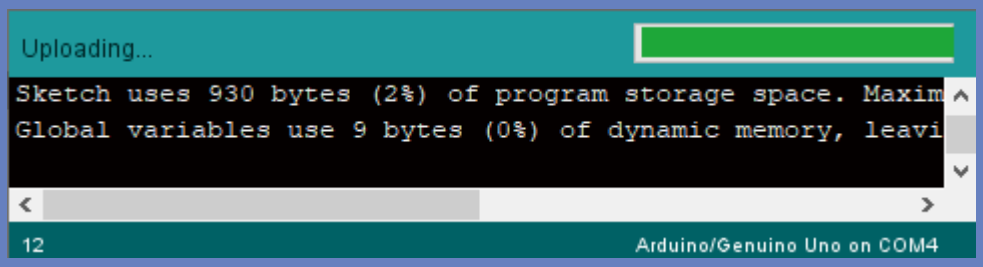


Click on the 'Upload' button. The second button from the left on the toolbar.

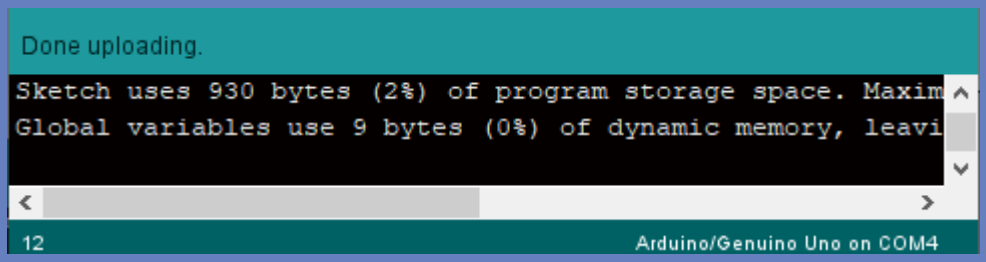
If you watch the status area of the IDE, you will see a progress bar and a series of messages. At first, it will say 'Compiling Sketch...'. This converts the sketch into a format suitable for uploading to the board.



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 930 bytes of the 32,256 bytes available.
After the 'Compiling Sketch..' stage you could get the following error message:

It means 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 3 and check your installation.
Once the upload has completed, the board should restart and start blinking.

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.

The Sketch start with

```
/*
Blink
Turns an LED on for one second, then off for one second, repeatedly.
...
This example code is in the public domain. http://www.arduino.cc/en/Tutorial/Blink
*/
// the setup function runs once when you press reset or power the board
```

//
[Further Syntax]
Description

Line comments are lines in the program that are used to inform yourself or others about the way the program works. They are ignored by the compiler, and not exported to the processor, so they don' t take up any space in the microcontroller' s flash memory. Comments' only purpose is to help you understand (or remember), or to inform others about how your program works.

A single line comment begins with **//** (two adjacent slashes). This comment ends automatically at the end of a line. Whatever follows **//** till the end of a line will be ignored by the compiler.

/**/
[Further Syntax]
Description

The beginning of a block comment or a multi-line comment is marked by the symbol /* and the symbol */ marks its end. This type of comment is called so as this can extend over more than one line; Once the compiler reads the /* it ignores whatever follows until it encounters a */.

The first block of code is :

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

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.

setup()
[Sketch]
Description

The setup() function is called when a sketch starts. Use it to initialize variables, pin modes, libraries initialisation, etc. The setup() function will only run once, after each powerup or reset of the Arduino board.

{ ...}
[Further Syntax]
Description

Curly braces (also referred to as just "braces" or as "curly brackets") are a major part of the C++ programming language. They are used in several different constructs, outlined below, and this can sometimes be confusing for beginners.

An opening curly brace **{ must always be followed by a closing curly brace }**. This is a condition that is often referred to as the braces being balanced. The Arduino IDE (Integrated Development Environment) includes a convenient feature to check the balance of curly braces. Just select a brace, or even click the insertion point immediately following a brace, and its logical companion will be highlighted.

Unbalanced braces can often lead to cryptic, impenetrable compiler errors that can sometimes be hard to track down in a large program. Because of their varied usages, braces are also incredibly important to the syntax of a program and moving a brace one or two lines will often dramatically affect the meaning of a program.

It is also mandatory for a sketch to have a 'loop' function.

```
// the loop function runs over and over again forever
void loop() {
digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
delay(1000);      // wait for a second
digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
delay(1000);// wait for a second
}
```

Loop : Unlike the 'setup' function that only runs once, oppositely, the loop function runs constantly.

digitalWrite(Pin num , HIGH/LOW):Write a HIGH or a LOW value to a digital pin.
If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value: 5V for HIGH, 0V (ground) for LOW.

delay(ms):Pauses the program for the amount of time (in milliseconds) specified as parameter. (There are 1000 milliseconds in a second).

ms : the number of milliseconds to pause.(range:0~4394967295).

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.

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?

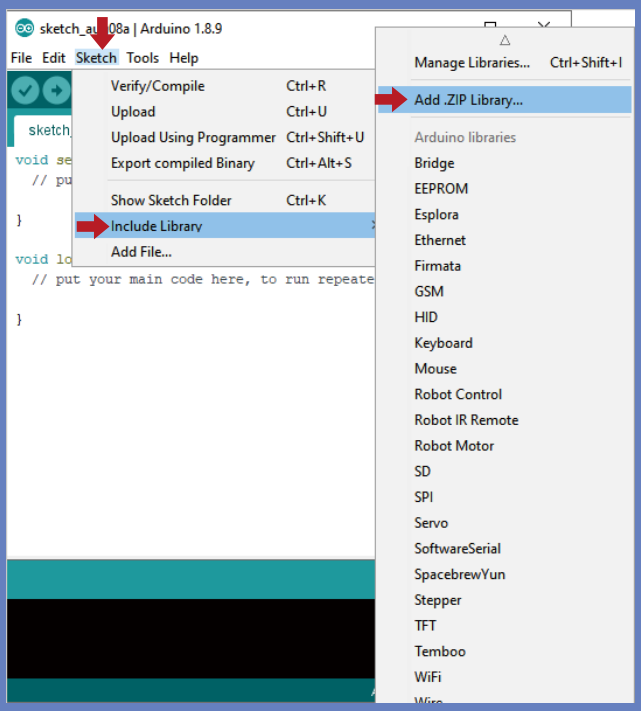
Libraries are a collection of code that makes it easy for you to connect to a sensor, display, module, etc. For example, the built-in LiquidCrystal library makes it easy to talk to character LCD displays.
There are hundreds of additional libraries available on the Internet for download. The built-in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

1) 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.

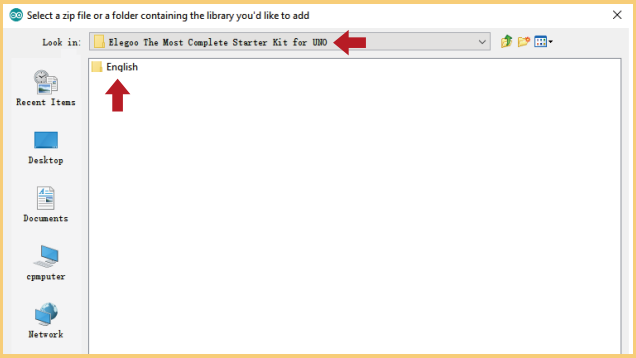
In the Arduino IDE, navigate to Sketch > Include Library. At the top of the drop down list, select the option to "Add .ZIP Library".

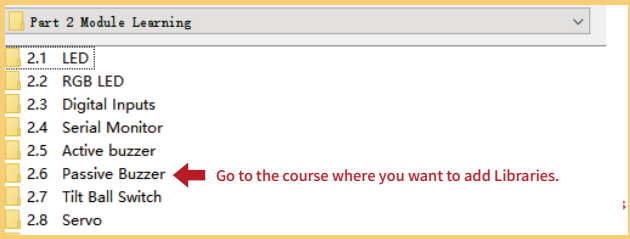


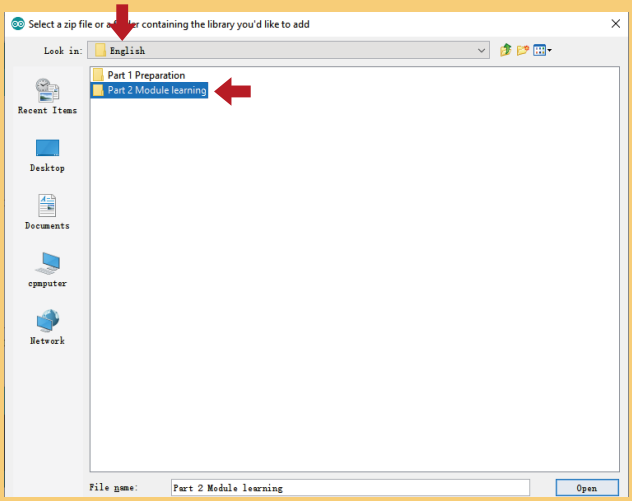
You will be prompted to select the library you would like to add. Navigate to the .zip file's location and open it.

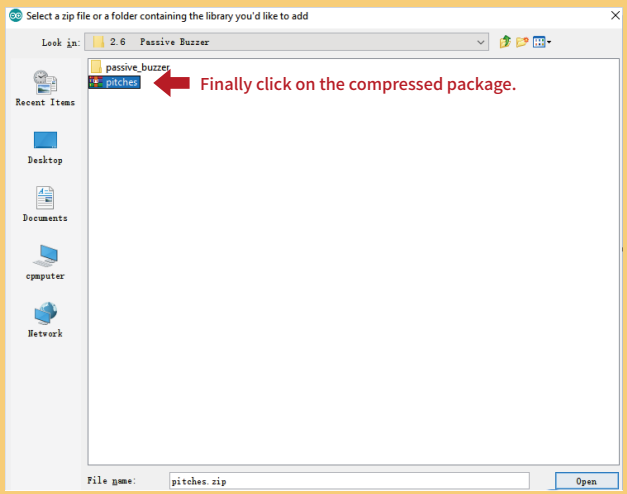
If the library is used in each course folder, it will be provided with the corresponding library compression package.

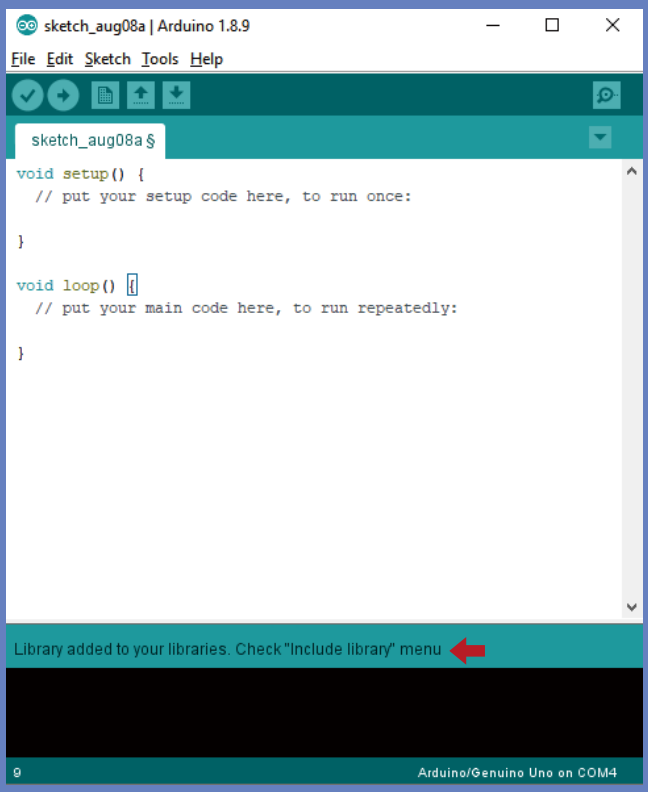
Take Lesson 11 as an example







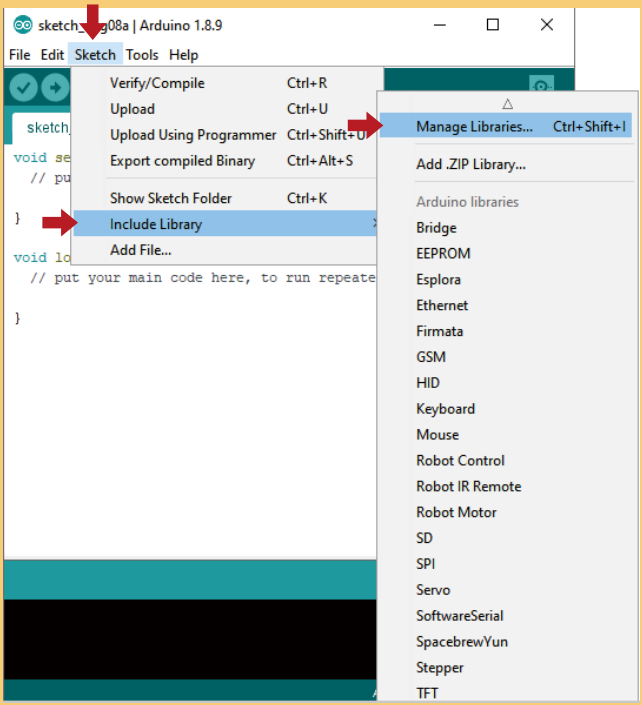




Return to the Sketch > Import Library menu. You should now see the library at the bottom of the drop-down menu. It is ready to be used in your sketch. The zip file has been expanded in the libraries folder in your Arduino sketches directory.
NB: the Library will be available to use in sketches, but examples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

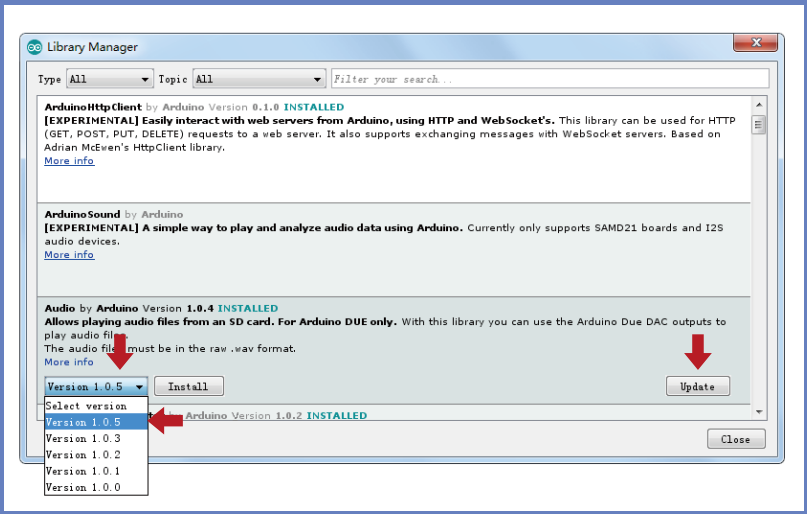
2) Using the Library Manager

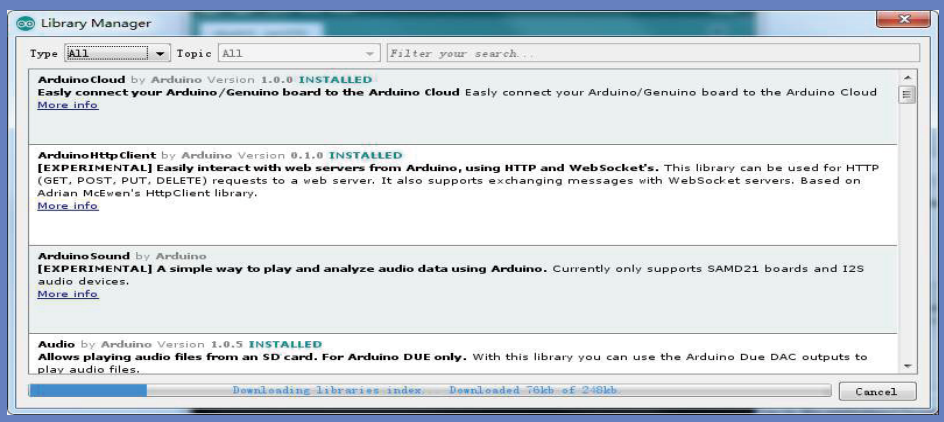
To install a new library into your Arduino IDE you can use the Library Manager (available from IDE version 1.8.9). Open the IDE and click to the "Sketch" menu and then Include Library > Manage Libraries.



Then 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 **Audio** 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.
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.

3) 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 called ArduinoParty, 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 !

Part 2

lesson 1

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 Elegoo Uno R3

(1) x 5mm red LED

(1) x 220Ω resistor

(1) x 1kΩ resistor

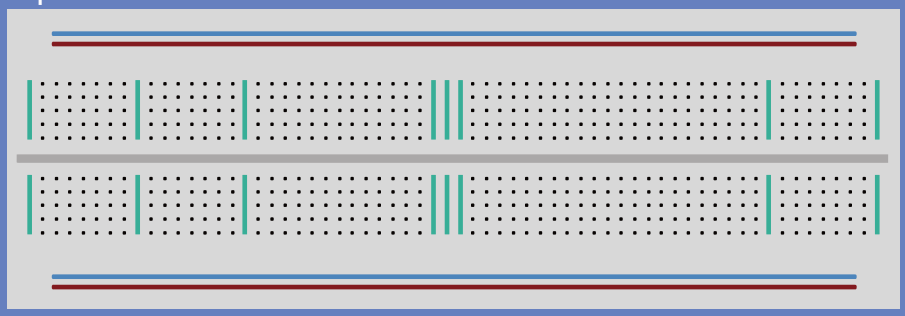
(1) x 10kΩ resistor

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

Component Introduction

BREADBOARD MB-102:

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



Breadboards come in various sizes and configurations.

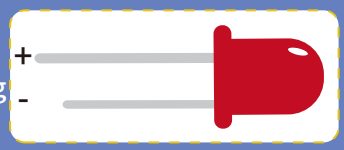
①The simplest kind is just a grid of holes in a plastic block. Inside are strips of metal that provide electrical connection between holes in the shorter rows. Pushing the legs of two different components into the same row joins them together electrically. A deep channel running down the middle indicates 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.

②Some 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:

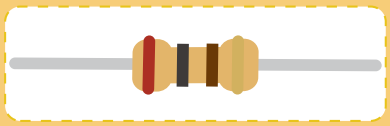
- **LEDs** LEDs make great indicator lights. They use very little electricity and they pretty much last forever.
- **In** this lesson, you will use perhaps the most common of all LEDs: a 5mm red LED.
- **5mm** refers to the diameter of the LED. Other common sizes are 3mm and 10mm.
- **You** cannot directly 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.
  ②an 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 it, 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.
  ① the positive lead is longer.
  ② 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.

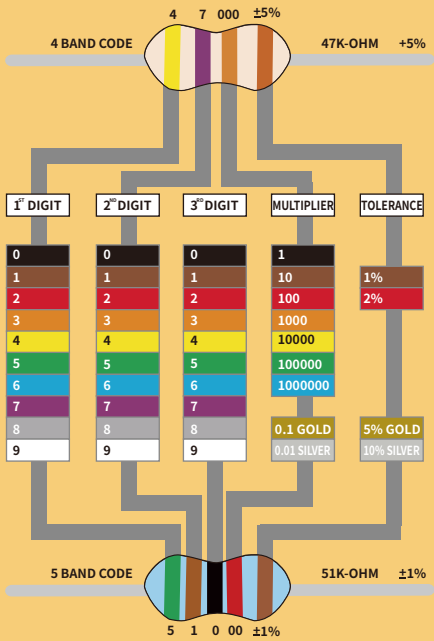
RESISTORS:

- 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 a resistor to limit how much current 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.

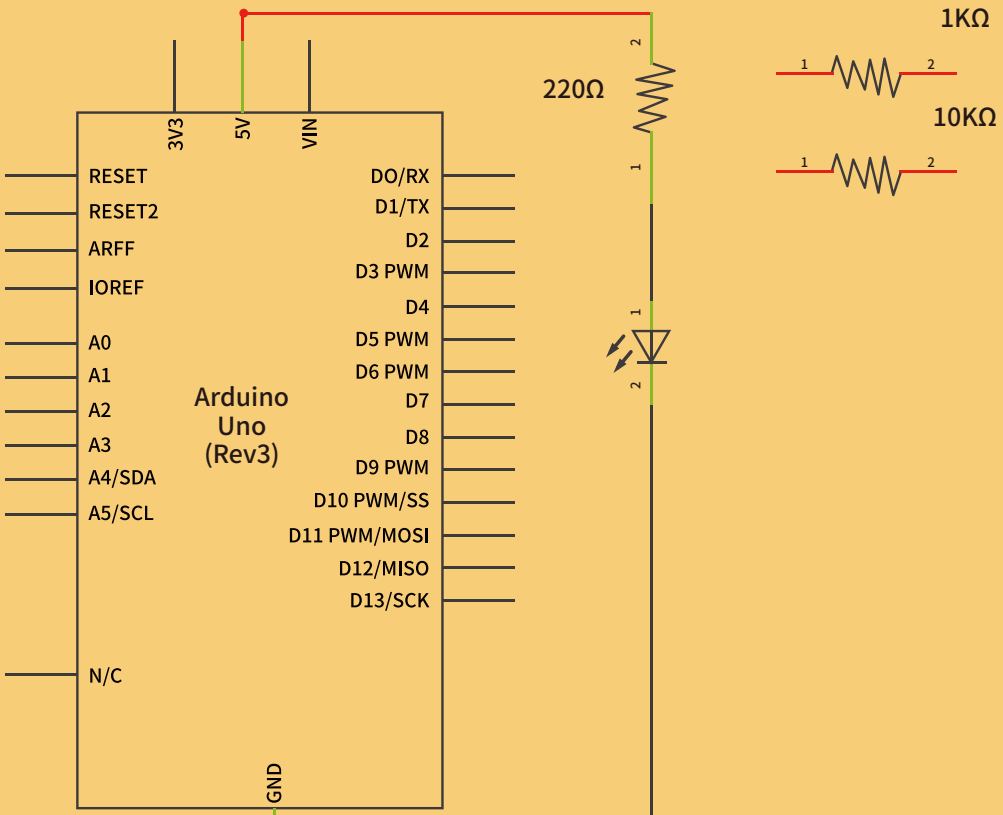


- In this lesson, we are going to use three different values of resistor: 220Ω, 1kΩ and 10kΩ. These resistors all look the same, except that they have different colored stripes on them. These stripes tell you the value of the resistor.

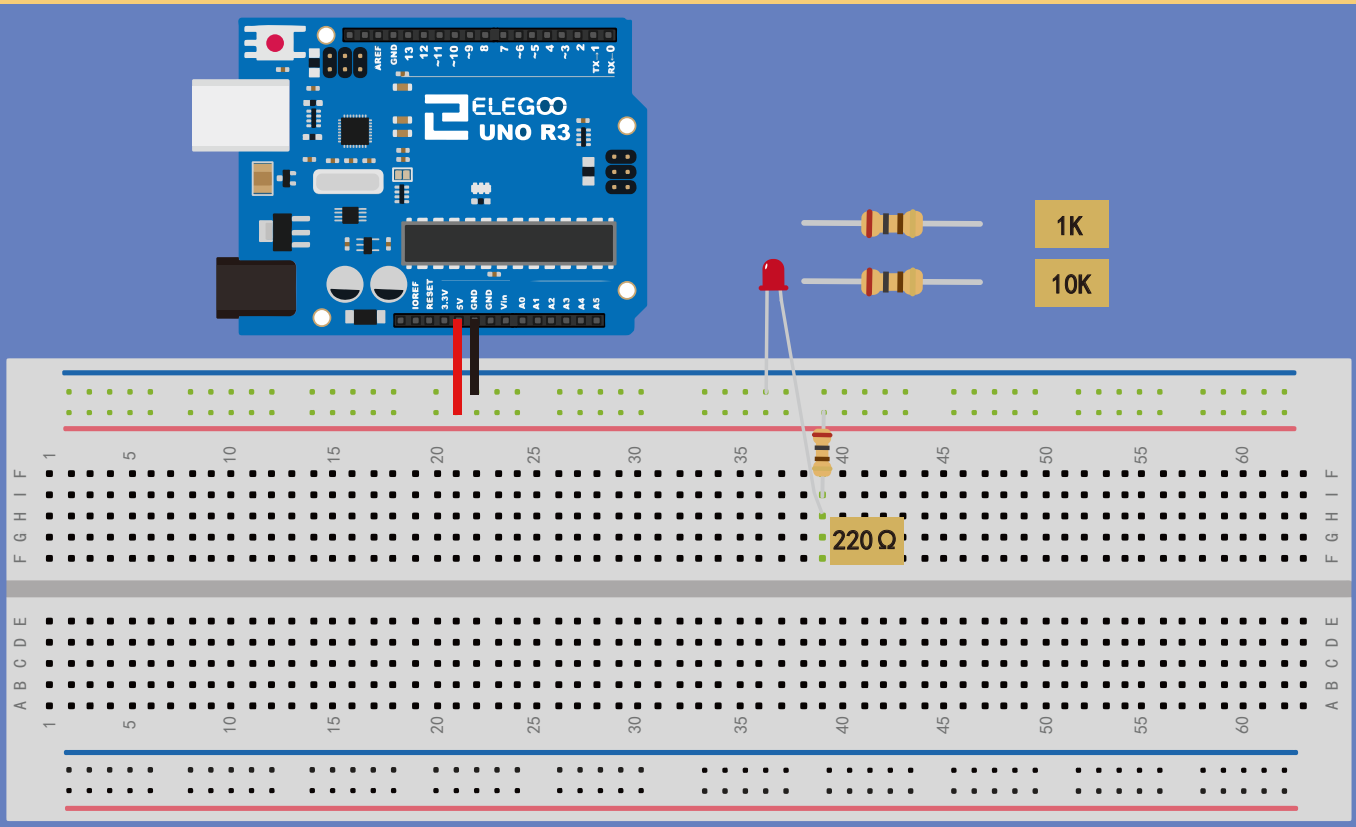
- The resistor color code has three colored stripes and then a gold stripe at one end.

- 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.

Part 2

lesson 2

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, using and connecting them is not much different.

They come mostly in 2 versions: **Common Anode** or **Common Cathode**.
Common Anode: uses 5V on the common pin.
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 Elegoo Uno R3

(1) x 830 Tie Points Breadboard

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

(1) x RGB LED

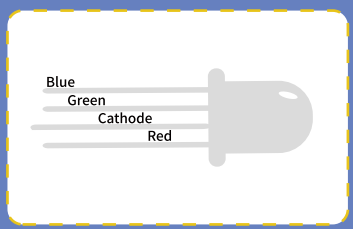
(3) x 220 ohm resistors

Component Introduction

RGB:

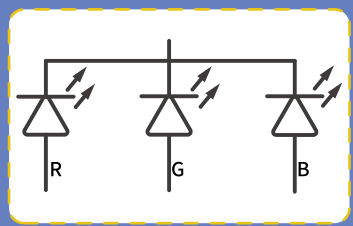
At first glance, RGB (Red, Green and Blue) LEDs look just like regular LEDs. However, inside the usual LED package, there are actually three LEDs, one red, one green and one blue. By controlling the brightness of each of the individual LEDs you can mix pretty much any color you want.

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.

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.



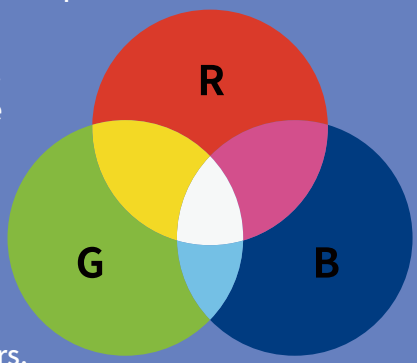
Each LED inside the package requires its own 220Ω resistor to prevent too much current flowing through it. The three positive leads of the LEDs (one red, one green and one blue) are connected to UNO output pins using these resistors.

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.

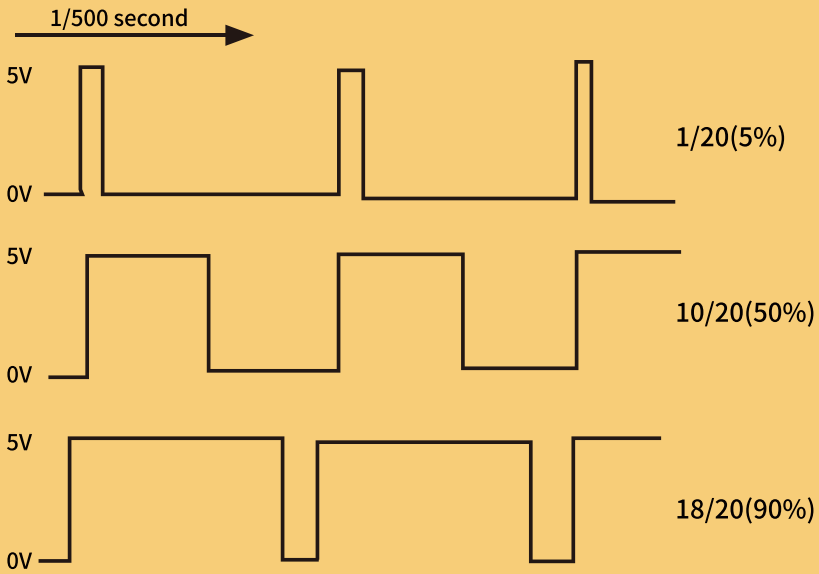
- **If** we set the brightness of all three LEDs to be the same, then the overall color of the light will be white. If we turn off the blue LED, so that just the red and green LEDs are the same brightness, then the light will appear yellow. We can control the brightness of each of the red, green and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

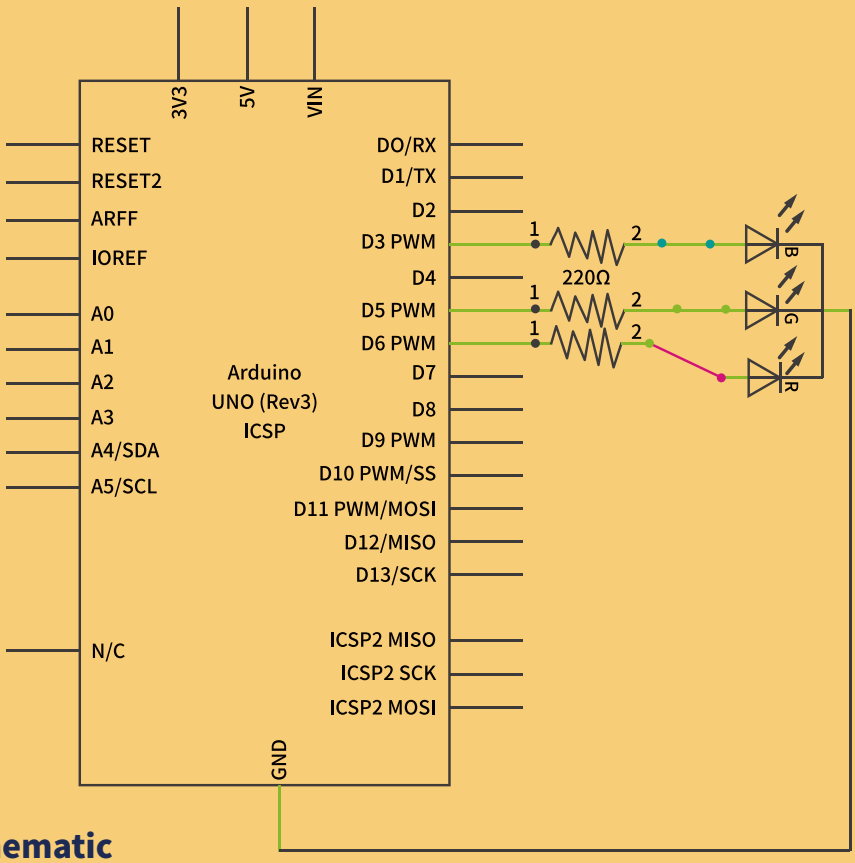
- Pulse Width Modulation (PWM) is a technique for controlling power. We also use it here to control the brightness of each of the LEDs.
  The diagram below shows the signal from one of the PWM pins on the UNO.



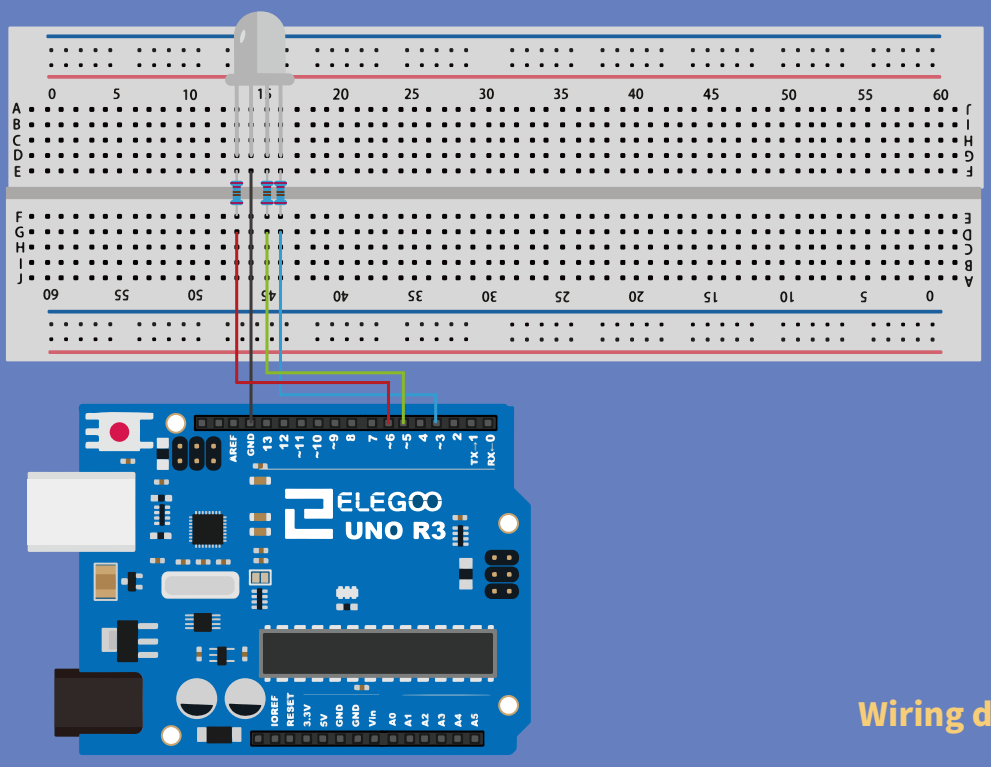
- Roughly every 1/500 of a second, the PWM output will produce a pulse. The length of this pulse is controlled by the 'analogWrite' function. So 'analogWrite(0)' will not produce any pulse at all and 'analogWrite(255)' will produce a pulse that lasts all the way until the next pulse is due, so that the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



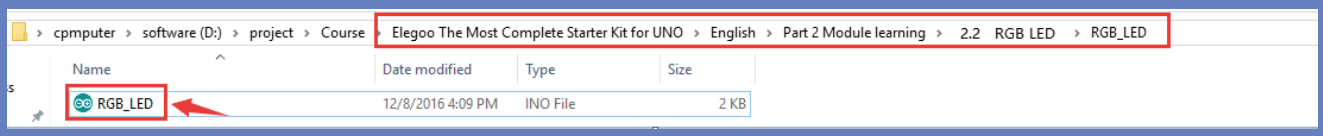
Connection Schematic



Wiring diagram

Code

- After wiring, please open the Sketch in folder path:
\Elegoo The Most Complete Starter Kit for UNO\English\Part 2 Module Learning\2.2 RGB LED,
and click UPLOAD to upload the program.



- See Lesson 5 in part 1 for details about program uploading if there are any errors.
- The sketch starts by specifying which pins are going to be used for each of the colors:

```
// Define Pins
#define BLUE 3 // The compiler will replace any mention of ledPin with the value 3 at compile time.
#define GREEN 5
#define RED 6
```

#define constantName value
is a useful C++ component that allows the programmer to give a name to a constant value before the program is compiled. Defined constants in arduino don' t take up any program memory space on the chip. The compiler will replace references to these constants with the defined value at compile time.

This can have some unwanted side effects though, if for example, a constant name that had been #defined is included in some other constant or variable name. In that case the text would be replaced by the #defined number (or text).

Parameters
constantName: the name of the macro to define.
value: the value to assign to the macro.

Notes and Warnings

- There is no semicolon after the #define statement. If you include one, the compiler will throw cryptic errors further down the page.
- **#define ledPin 3; // this is an error**
- Similarly, including an equal sign after the #define statement will also generate a cryptic compiler error further down the page.
- **#define ledPin = 3 // this is also an error**

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.

pinMode(pin, mode) Configures the specified pin to behave either as an input or an output.

As of Arduino 1.0.1, it is possible to enable the internal pullup resistors with the mode INPUT_PULLUP.
Additionally, the INPUT mode explicitly disables the internal pullups.

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

Parameters

pin: the Arduino pin number to set the mode of.

mode: INPUT, OUTPUT, or INPUT_PULLUP. See the Digital Pins page for a more complete description of the functionality.

Int: is the Data Types. Integers are your primary data-type for number storage.

```
// define variables
int redValue;
int greenValue;
int blueValue;
```

On the Arduino Uno (and other ATmega based boards) an int stores a 16-bit (2-byte) value. This yields a range of -32,768 to 32,767 (minimum value of -2^15 and a maximum value of (2^15) - 1). On the Arduino Due and SAMD based boards (like MKR1000 and Zero), an int stores a 32-bit (4-byte) value.
This yields a range of -2,147,483,648 to 2,147,483,647 (minimum value of -2^31 and a maximum value of (2^31) - 1).

int' s store negative numbers with a technique called (2' s complement math). The highest bit, sometimes referred to as the "sign" bit, flags the number as a negative number. The rest of the bits are inverted and 1 is added.

Syntax

```
int var = val;
```

Parameters

var: variable name.

val: the value you assign to that variable.

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.

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

```
for (int i = 0; i < 255; i += 1) // fades out red bring green full when i=255
{
redValue -= 1;
greenValue += 1;
// The following was reversed, counting in the wrong directions
// analogWrite(RED, 255 - redValue);
// analogWrite(GREEN, 255 - greenValue);
analogWrite(RED, redValue);
analogWrite(GREEN, greenValue);
delay(delayTime);
}
```

for

[Control Structure]

Description

The for statement is used to repeat a block of statements enclosed in curly braces.
An increment counter is usually used to increment and terminate the loop.
The for statement is useful for any repetitive operation, and is often used in combination with arrays to operate on collections of data/pins.

Syntax

```
for ( initialization; condition; increment ) {
  // statement(s);
}
```

Parameters

initialization: happens first and exactly once.
condition: each time through the loop, condition is tested; if it' s true, the statement blocks, and the increment is executed, then the condition is tested again. When the condition becomes false, the loop ends.
increment: executed each time through the loop when condition is true.

=

[Arithmetic Operators]

Description

The single equal sign = in the C++ programming language is called the assignment operator. It has a different meaning than in algebra class where it indicated an equation or equality. The assignment operator tells the microcontroller to evaluate whatever value or expression is on the right side of the equal sign, and store it in the variable to the left of the equal sign.

Notes and Warnings

The variable on the left side of the assignment operator (= sign) needs to be able to hold the value stored in it. If it is not large enough to hold a value, the value stored in the variable will be incorrect.

Don' t confuse the assignment operator [=] (single equal sign) with the comparison operator [==] (double equal signs), which evaluates whether two expressions are equal.

+= / -=

[Compound Operators]

Description

This is a convenient shorthand to perform addition/subtraction on a variable with another constant or variable.

Syntax

```
x += y; // equivalent to the expression x = x + y;
x -= y; // equivalent to the expression x = x - y;
```

Parameters

x: variable. Allowed data types: int, float, double, byte, short, long.
y: variable or constant. Allowed data types: int, float, double, byte, short, long.

Part 2

lesson 3

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 Elegoo Uno R3

(1) x 830 Tie-points Breadboard

(1) x 5mm red LED

(1) x 220 Ω resistor

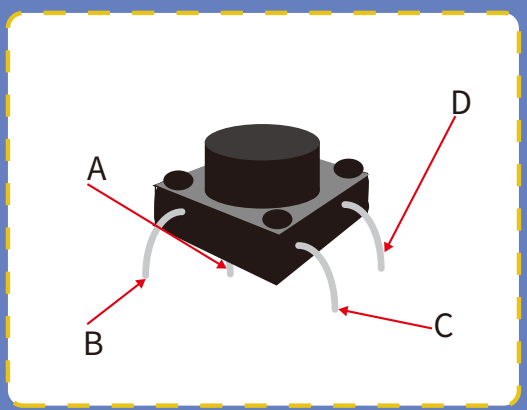
(2) x push switches

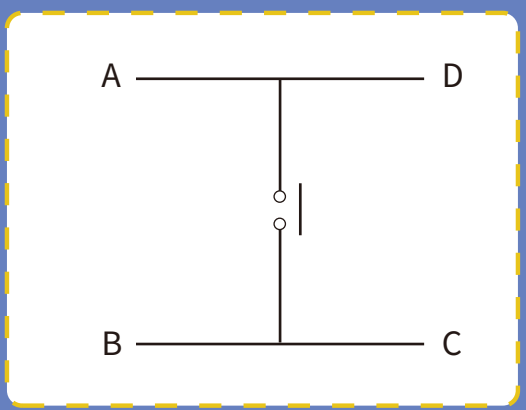
(7) x M-M wires (Male to Male jumper wires)

Component Introduction

PUSH SWITCHES:

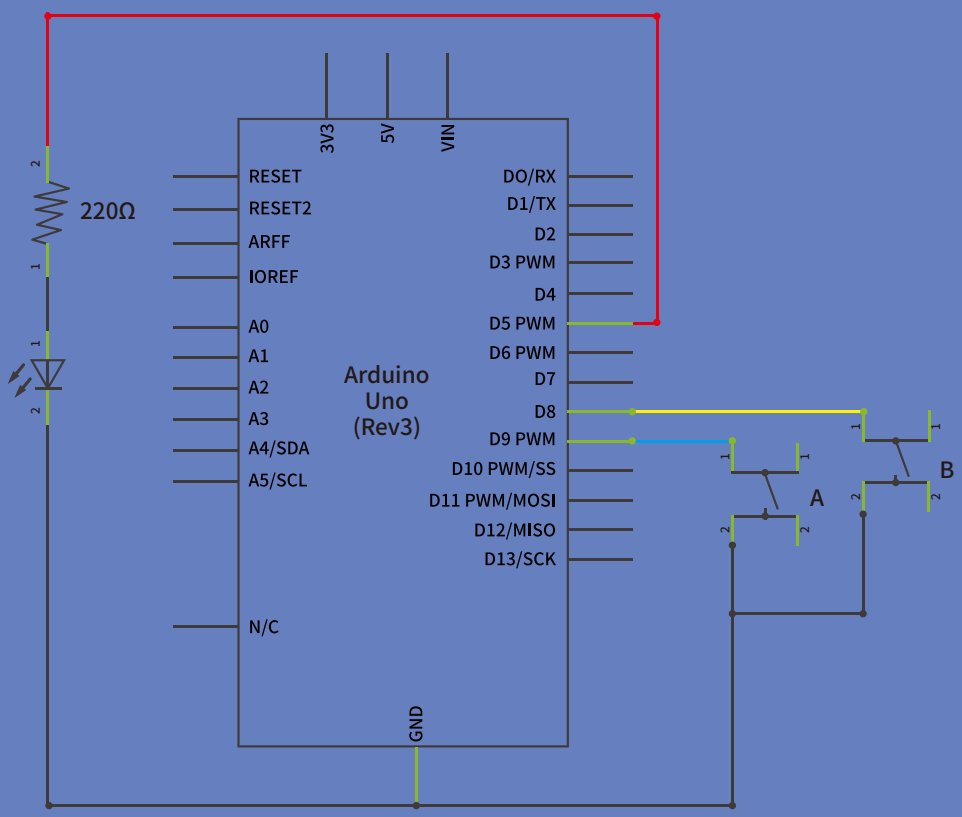
Switches are really simple components. When you press a button or flip a lever, they connect two contacts together so that electricity can flow through them.The little tactile switches that are used in this lesson have four connections, which can be a little confusing.

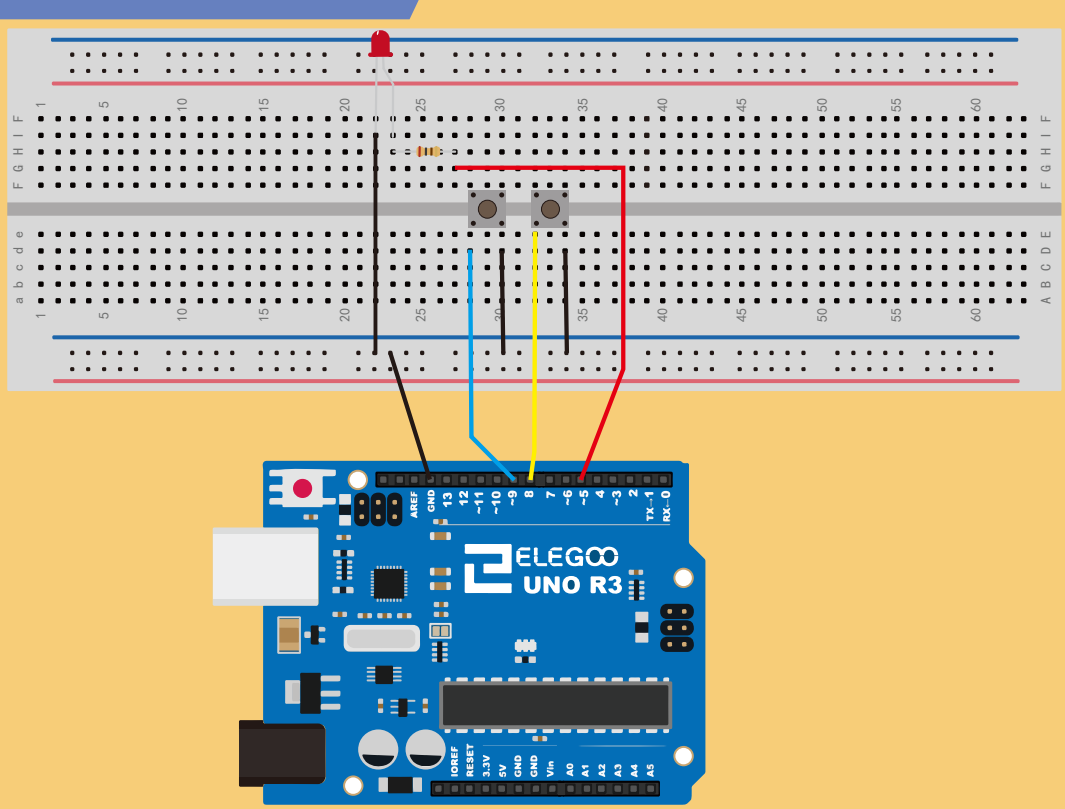




Actually, there are only really two electrical connections. Inside the switch package, pins B and C are connected together, as are A and D.

Connection Schematic:

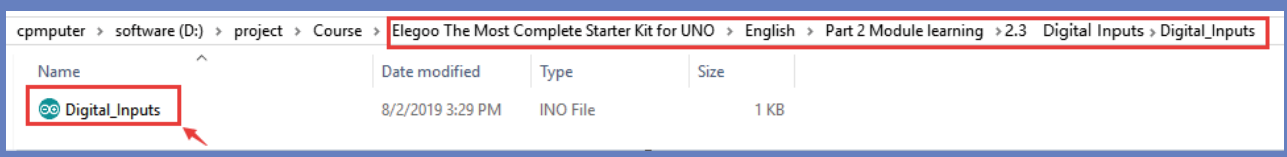




- Although the bodies of the switches are square, the pins protrude from opposite sides of the switch. This means that the pins will only be far enough apart when they are placed correctly on the breadboard.
- Remember that the LED has to have the shorter negative lead to the left.

Code

Please open the program:



- 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 (buttonApin,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.
- 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.

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

- ①If {} : The if statement checks for a condition and executes the proceeding statement or set of statements if the condition is 'true'.

Parameters

condition: a boolean expression (i.e., can be true or false).

```
Syntax
if (condition) {
  //statement(s)
}
```

- ②digitalRead() : Reads the value from a specified digital pin, either HIGH or LOW.

Syntax

digitalRead(pin)

Parameters

pin: the Arduino pin number you want to read

Returns

HIGH or LOW

==

[Comparison Operators]

Description

Compares the variable on the left with the value or variable on the right of the operator. Returns true when the two operands are equal. Please note that you may compare variables of different data types, but that could generate unpredictable results, it is therefore recommended to compare variables of the same data type including the signed/unsigned type.

Syntax

```
x == y; // is true if x is equal to y and it is false if x is not equal to y
```

Parameters

x: variable. Allowed data types: int, float, double, byte, short, long.

y: variable or constant. Allowed data types: int, float, double, byte, short, long.

Part 2

lesson 4

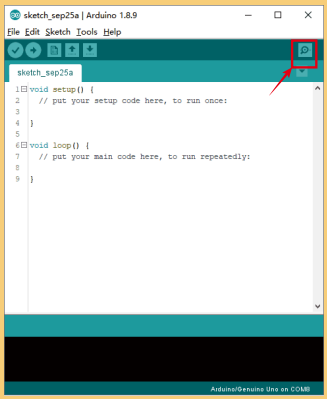
Serial Monitor

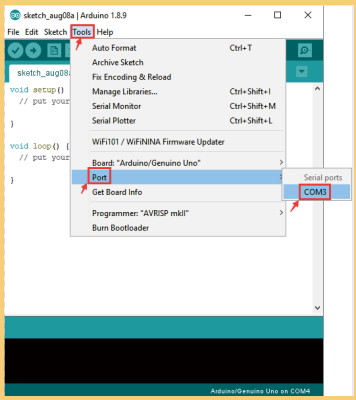
Arduino Serial Monitor (Windows, Mac, Linux)

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.

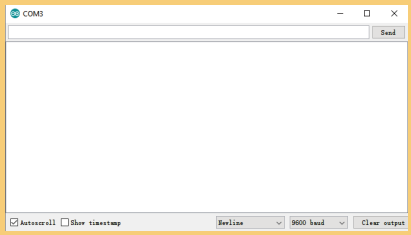




Selecting which port to open in the Serial Monitor is the same as selecting a port for uploading Arduino code. Go to **Tools ->Port**, and select the correct port.

Tips: Choose the same COM port that you have in Device Manager.

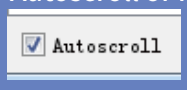
Once open, you should see something like this:

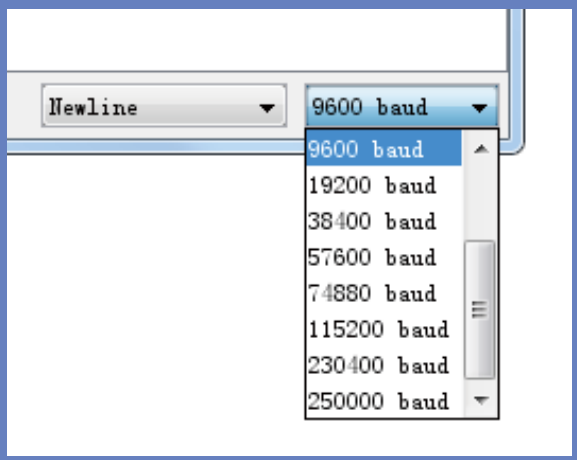


Settings

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.

Part 2

lesson 8

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 Elegoo Uno R3

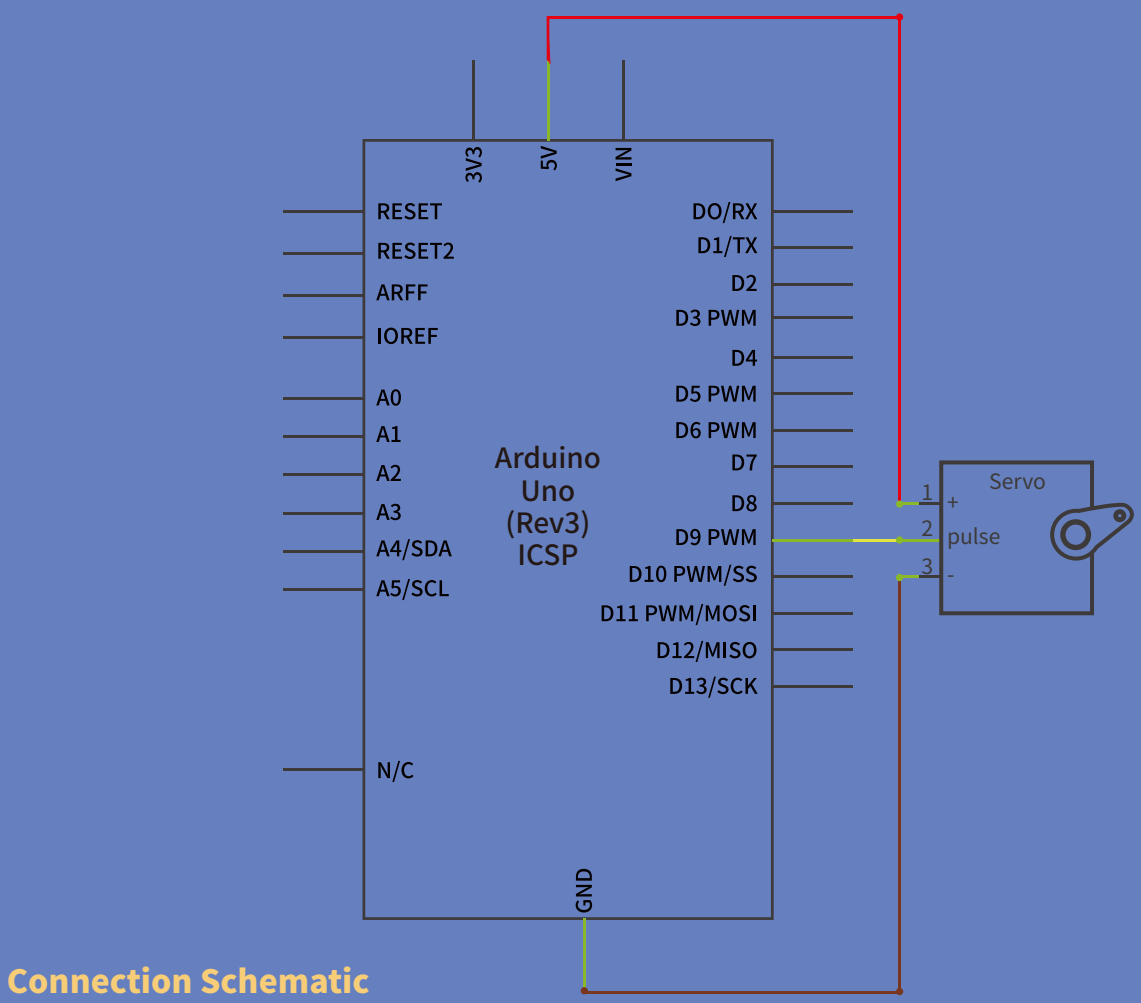
(1) x Servo (SG90)

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

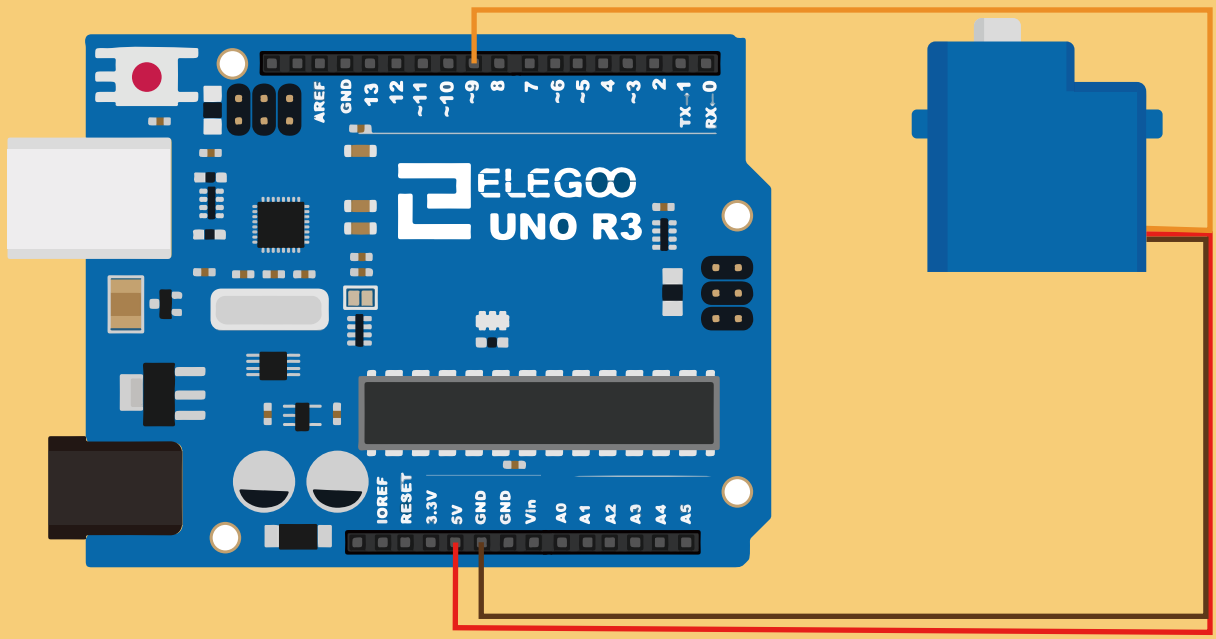
Component Introduction

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 Folder **Servo** where the course is located and click UPLOAD to upload the program. See Lesson 5 in part 1 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 5 in part 1 too.

Part 2

lesson 9

Ultrasonic Sensor Module

Overview

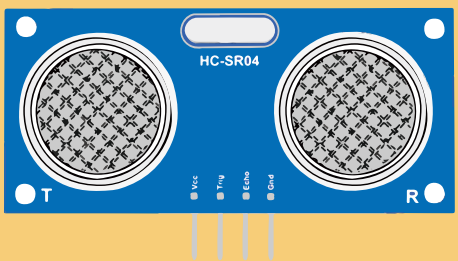
The ultrasonic sensor is great for all kinds of projects that need distance measurements or avoiding obstacles for example. The HC-SR04 is inexpensive and easy to use since we will be using a Library specifically designed for these sensor.

Component Required:

(1) x Elegoo Uno R3

(1) x Ultrasonic sensor module

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



Component Introduction

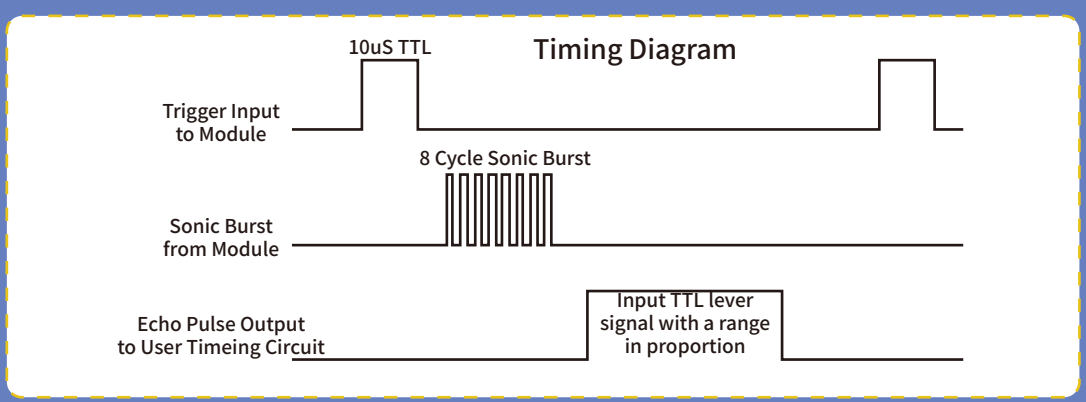
Ultrasonic sensor

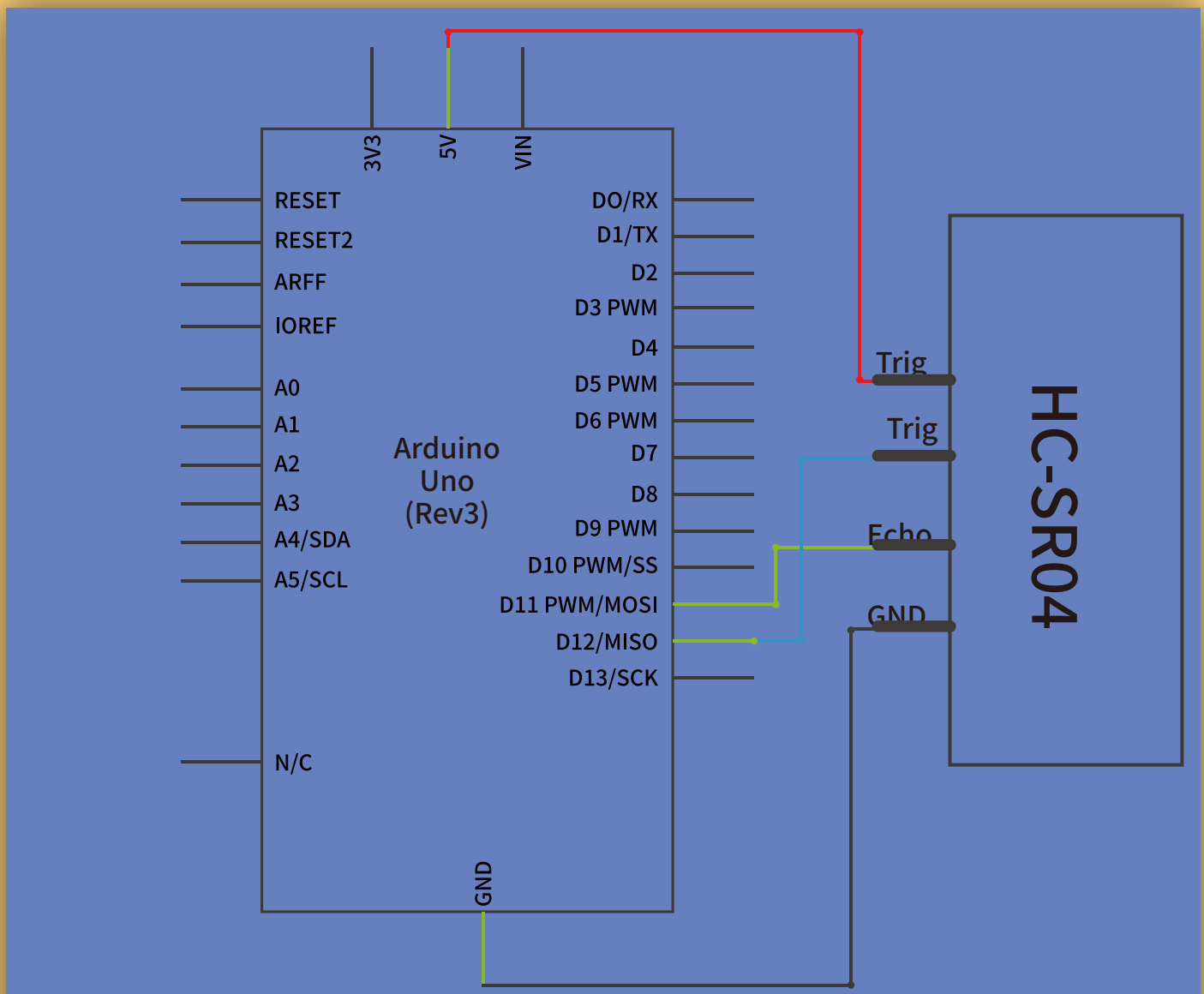
Ultrasonic sensor module HC-SR04 provides 2cm-400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:

(1) Using IO trigger for at least 10us high level signal,
(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic tore turning.

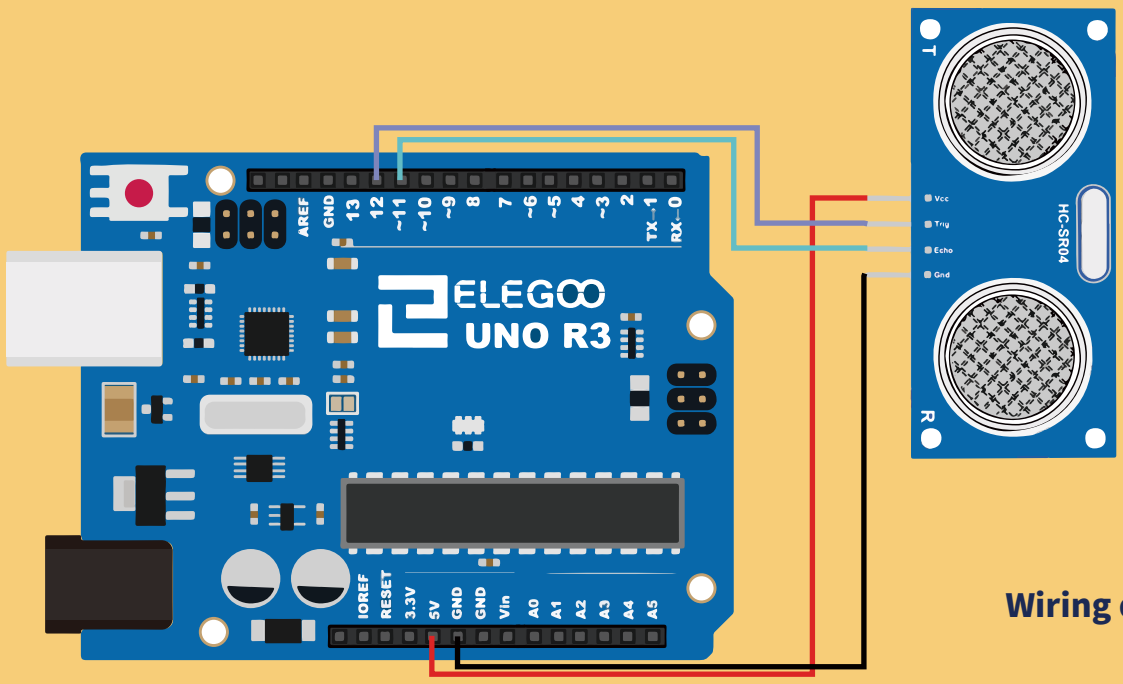
Test distance = (high level time × velocity of sound (340m/s) /2

The Timing diagram is shown below. You only need to supply a short 10us pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo pulse width is proportional to the distance of the object, or range. You can calculate through the time interval between sending trigger signal and receiving echo signal. Formula: us / 58 = centimeters or us / 148 =inch; or:
the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.





Connection Schematic



Wiring diagram

Code

Using a Library designed for these sensors will make our code short and simple. We include the library at the beginning of our code, and then by using simple commands we can control the behavior of the sensor.

After wiring, please open the program in the code folder-SR04_Example and click UPLOAD to upload the program. See Lesson 5 for details about program uploading if there are any errors.

Before you can run this, make sure that you have installed the **<HC-SR04>** library or re-install it, if necessary. Otherwise, your code won't work.

```
long a;
```

long()

[Conversion]

Description

Long keyword indicates a long integer data, which is a basic data type in programming language. It is the abbreviation of long int. it is a signed long integer by default, which contains 4 bytes, and the value range is: - 2 ^ 31 ~ (2 ^ 31-1)

Syntax

```
long(x)
(long)x (C-style type conversion)
```

Parameters

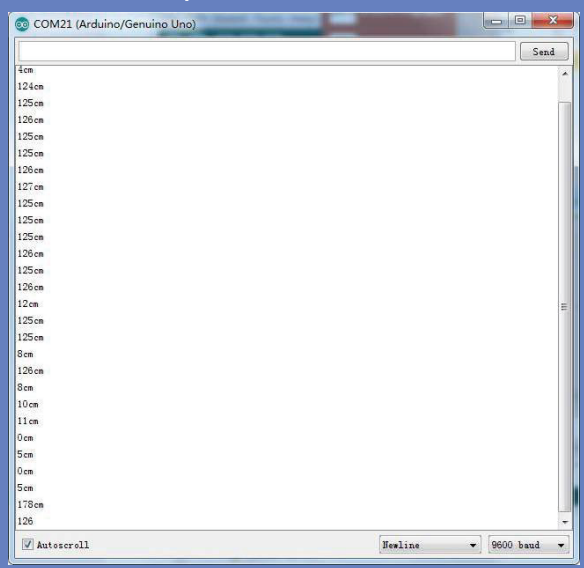
X: a value. Allowed data types: any type.

Returns

Data type: long.

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 4 of part 2.



Part 2

lesson 14

DC Motors

Overview

In this lesson, you will learn how to control a small DC motor using an UNO R3 and a L293D IC.

Component Required:

(1) x Elegoo Uno R3

(1) x 830 tie-points breadboard

(1) x L293D IC

(1) x Fan blade and 3-6v motor

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

(1) x Power Supply Module

(1)x 9V1A adapter

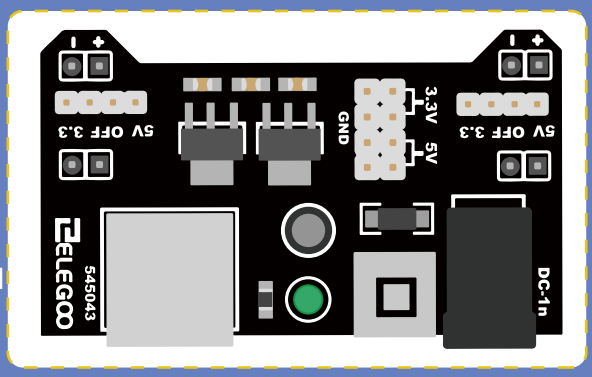
Component Introduction

Breadboard Power Supply

The small DC motor is likely to use more power than an UNO R3 board digital output can handle directly. If we tried to connect the motor straight to an UNO R3 board pin, ,then it' s likely to damage the UNO R3 board. So we use a power supply module provides power supply.

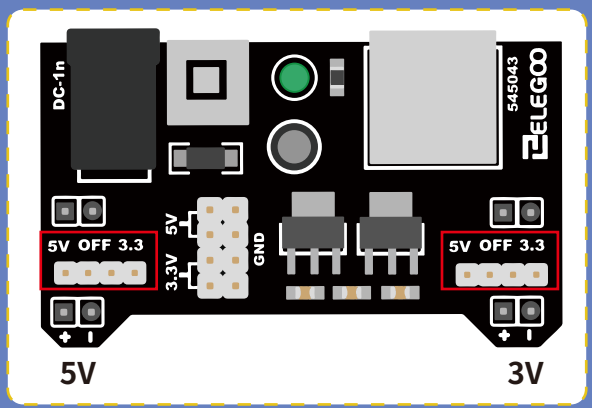
Product Specifications:

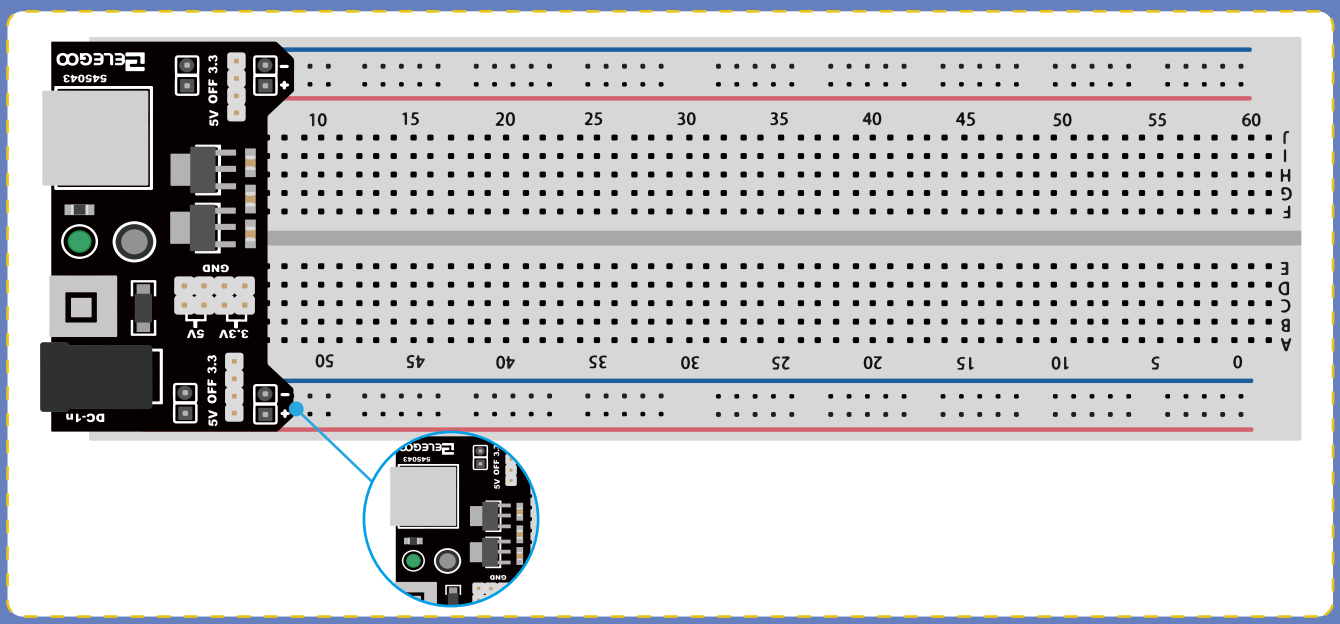
- Locking On/Off Switch
- LED Power Indicator
- Input voltage: 6.5-9v (DC) via 5.5mm x 2.1mm plug
- Output voltage: 3.3V/5v
- Maximum output current: 700 mA
- Independent control rail output. 0v, 3.3v, 5v to breadboard
- Output header pins for convenient external use
- Size: 2.1 in x 1.4 in
- USB device connector onboard to power external device



Setting up output voltage:

The left and right voltage output can be configured independently. To select the output voltage, move jumper to the corresponding pins. Note: power indicator LED and the breadboard power rails will not power on if both jumpers are in the "OFF" position.





Important note:

Make sure that you align the module correctly on the breadboard. The negative pin(-) on module lines up with the blue line(-) on breadboard and that the positive pin(+) lines up with the red line(+). Failure to do so could result in you accidently reversing the power to your project.

L293D

This is a very useful chip. It can actually control two motors independently. We are just using half the chip in this lesson, most of the pins on the right hand side of the chip are for controlling a second motor.

Product Specifications:

- Featuring Unitrode L293 and L293D Products Now From Texas Instruments
- Wide Supply-Voltage Range: 4.5 V to 36 V
- Separate Input-Logic Supply
- Internal ESD Protection
- Thermal Shutdown
- High-Noise-Immunity Inputs
- Functionally Similar to SGS L293 and SGS L293D
- Output Current 1 A Per Channel (600 mA for L293D)
- Peak Output Current 2 A Per Channel (1.2 A for L293D)
- Output Clamp Diodes for Inductive Transient Suppression (L293D)

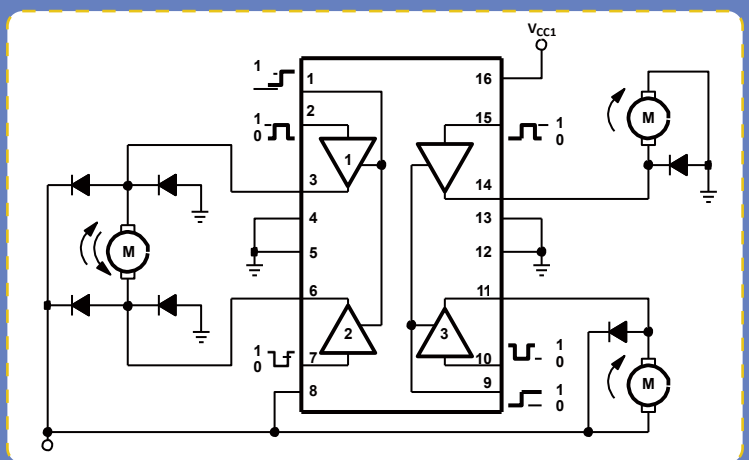
Description/ordering information

The L293 and L293D are quadruple high-current half-H drivers. The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

All inputs are TTL compatible. Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN. When an enable input is high, the associated drivers are enabled, and their outputs are active and in phase with their inputs. When the enable input is low, those drivers are disabled, and their outputs are off and in the high-impedance state. With the proper data inputs, each pair of drivers forms a full-H (or bridge) reversible drive suitable for solenoid or motor applications.

Block diagram

There are 3 wires connected to the Arduino, 2 wires connected to the motor, and 1 wire connected to a battery.



L293D

M1 PWM	1	16	Battery+ve
M1 direction 0/1	2	15	M2 direction 0/1
M1+ve	3	14	M2+ve
GND	4	13	GND
GND	5	12	GND
M1-ve	6	11	M2-ve
M1direction 1/0	7	10	M2 direciton 1/0
Battery+ve	8	9	M2 PWM
Motor 1		Motor 2	

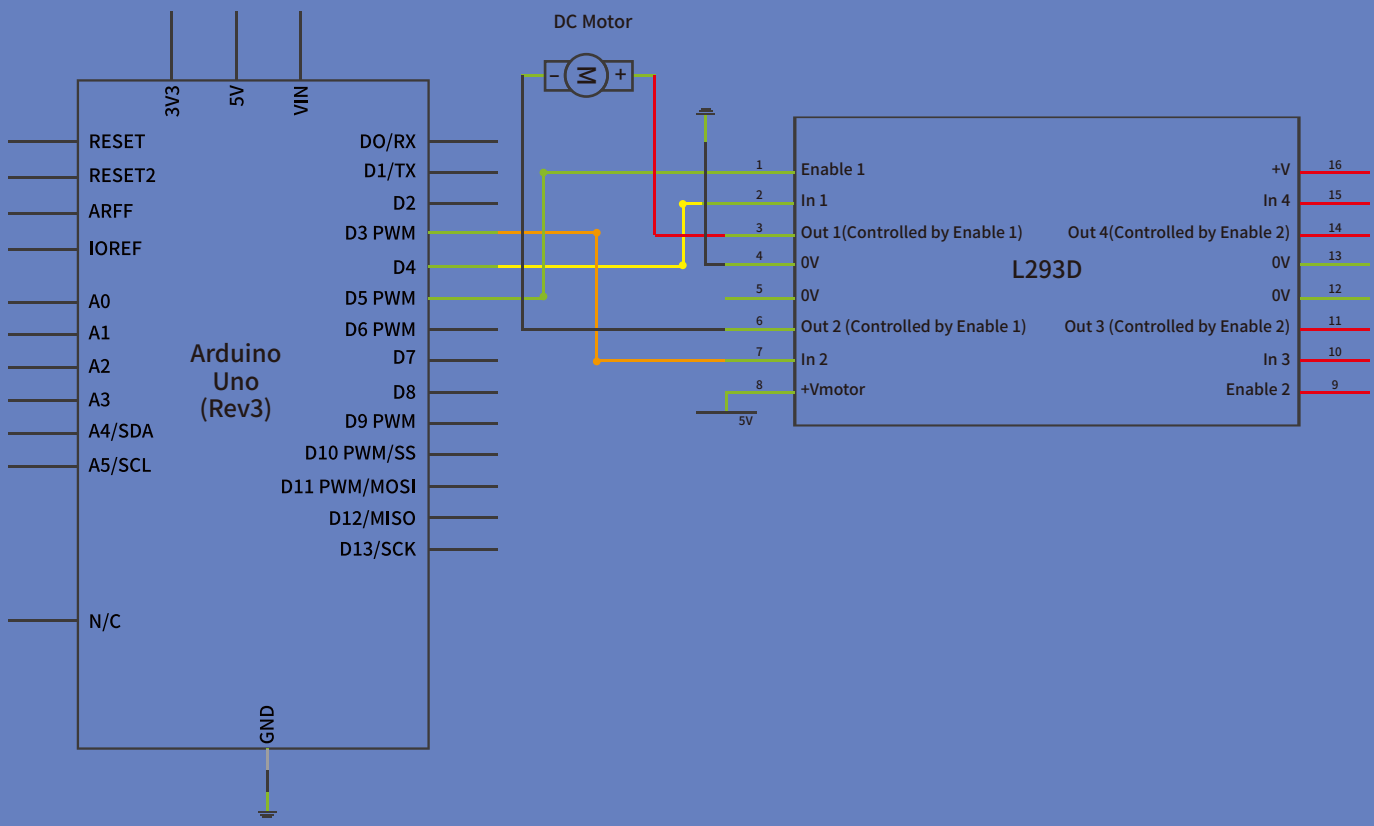
To use this pinout:
The left hand side deals with the first motor, the right hand side deals with a second motor.
Yes, you can run it with only one motor connected.

Arduino Connections

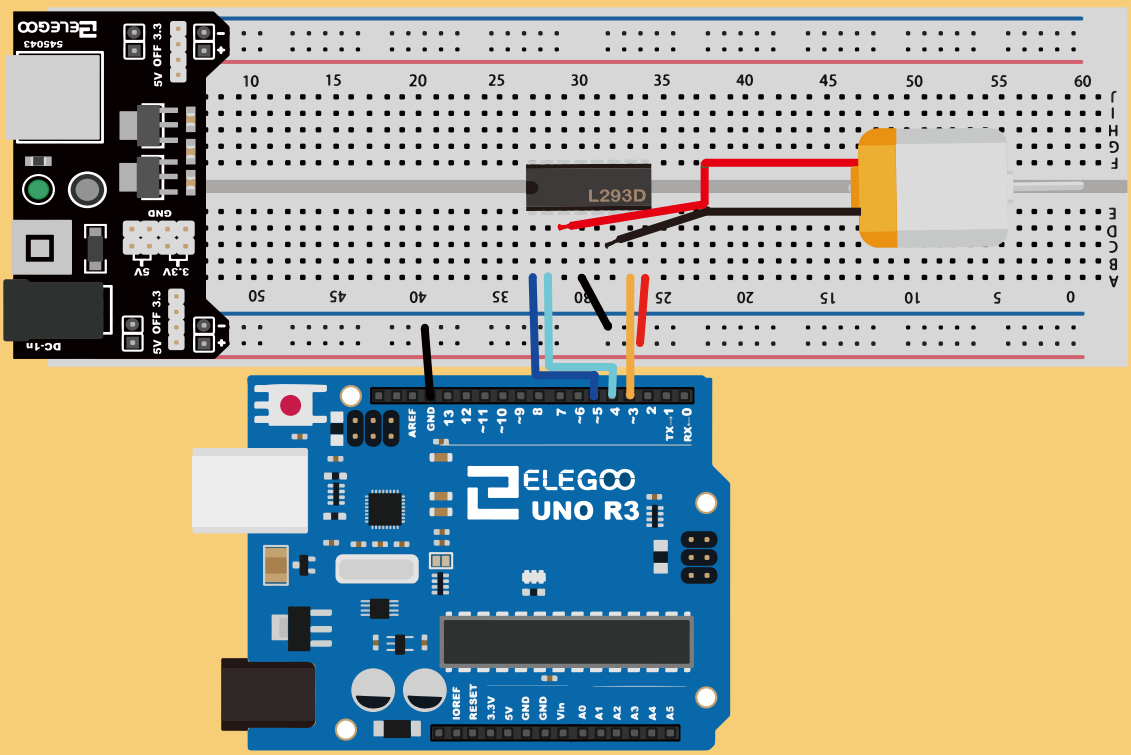
M1 PWM - connect this to a PWM pin on the Arduino. They're labelled on the Uno, pin 5 is an example. Output any integer between 0 and 255, where 0 will be off, 128 is half speed and 255 is max speed.

M1 direction 0/1 and M1 direction 1/0 - Connect these two to two digital Arduino pins. Output one pin as HIGH and the other pin as LOW, and the motor will spin in one direction.

Reverse the outputs to LOW and HIGH, and the motor will spin in the other direction.



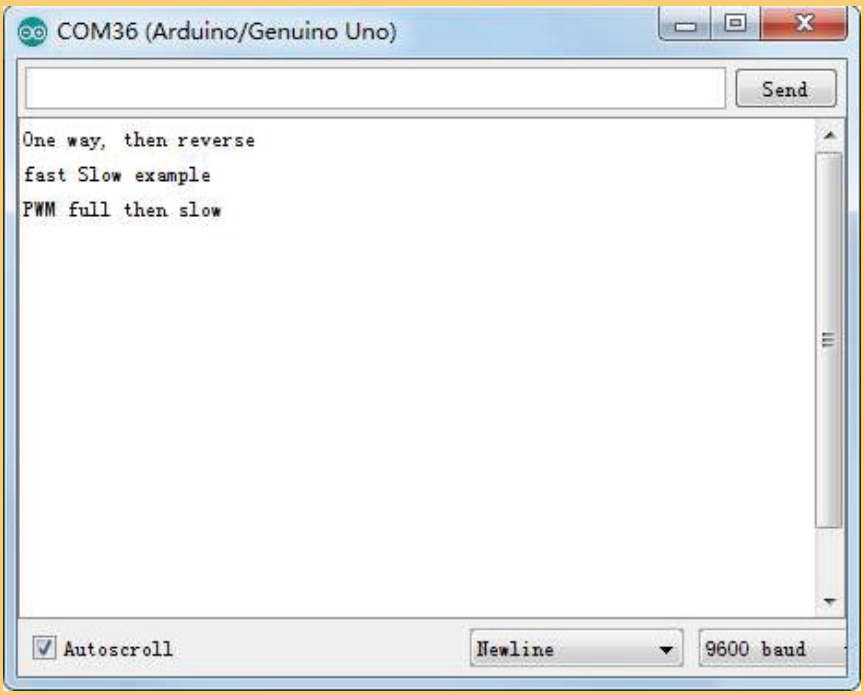
Connection Schematic



Wiring diagram

- The code below does not use a separate power supply (i.e. a battery), it uses instead the 5v power from the Arduino. Note that this would be risky without the L293D controlling it.

- You should _never_ connect a motor directly to the Arduino, because when you switch a motor off you get an electrical feedback. With a small motor, this will damage your Arduino, and with a large motor, you can watch an interesting flame and sparks effect.



Code

- After wiring, please open the program in the code folder- **DC_Motor** and click UPLOAD to upload the program. See Lesson 5 of part 1 for details about program uploading if there are any errors.

- After program loading, turn on all the power switches. The motor will slightly rotate clockwise and anticlockwise for 5 times. Then, it will continue to dramatically rotate clockwise. After a short pause, it will dramatically rotate anticlockwise. Then the controller board will send PWM signal to drive the motor, the motor will slowly reduce its maximum RPM to the minimum and increase to the maximum again. Finally, it comes to a stop for 10s until the next cycle begins.