PLUSIVO

THE MOST COMPLETE STARTER KIT

TUTORIAL FOR UNO

Table of Contents

1. Lesson 0. Installing IDE

1.1 Introduction

The Arduino device has a software side called The Arduino Integrated Development Environment (IDE).

First, you will learn how to properly setup your computer to use your Arduino device and install the software. The IDE that will be used to program your Arduino is available for Windows, Mac and Linux, but the installation is not the same for all three.

STEP 1: Go to https://www.arduino.cc/en/Main/Software and locate this page.

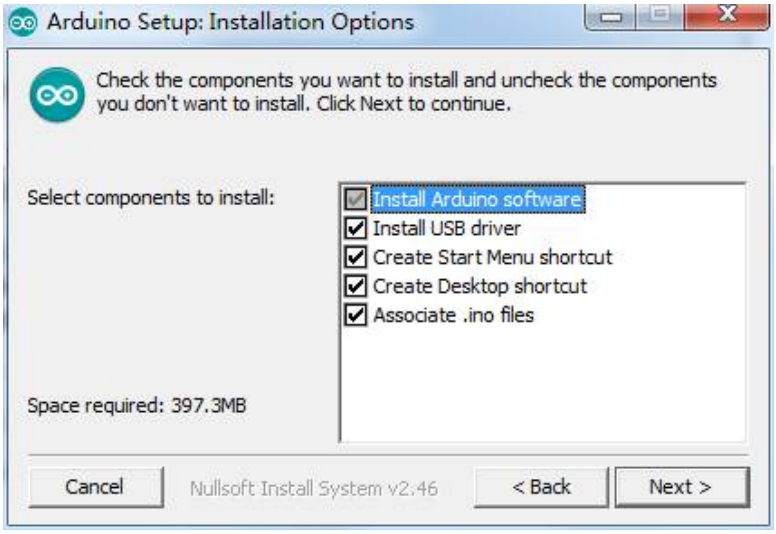
You will probably want to download the latest version, but here we will be using Arduino 1.8.0.

STEP 2: Dowload the version of software that is compatible with your computer's operating system. Click JUST DOWNLOAD (or choose to contribute if you wish).

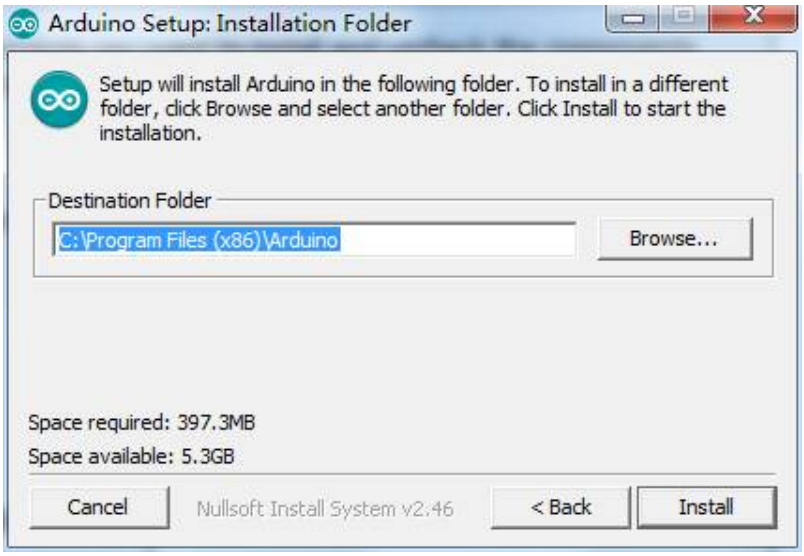
1.2 Installing Arduino (Windows)

Click the .exe installation package to install Arduino for Windows.

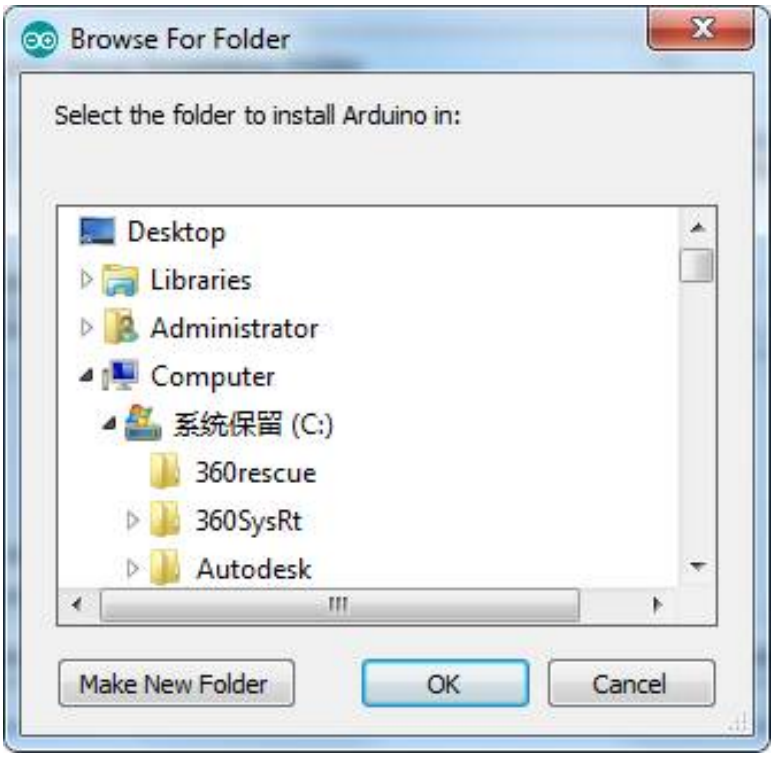
After reading the license agreement, click I agree.


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
Cancel
Nullsoft Install System v2.46
< Back
Next >

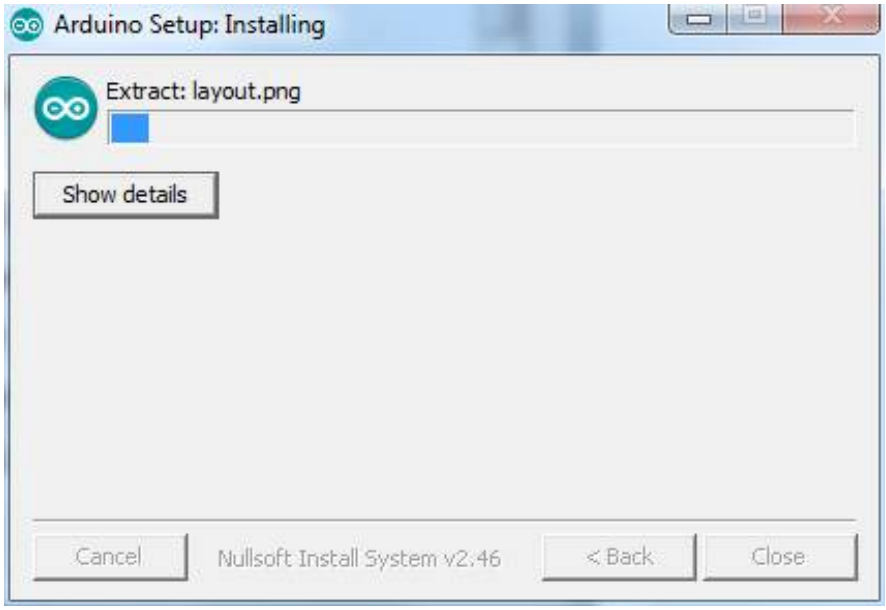
Click Next.


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
Browse...
Space required: 397.3MB
Space available: 5.3GB
Cancel
Nullsoft Install System v2.46
< Back
Install

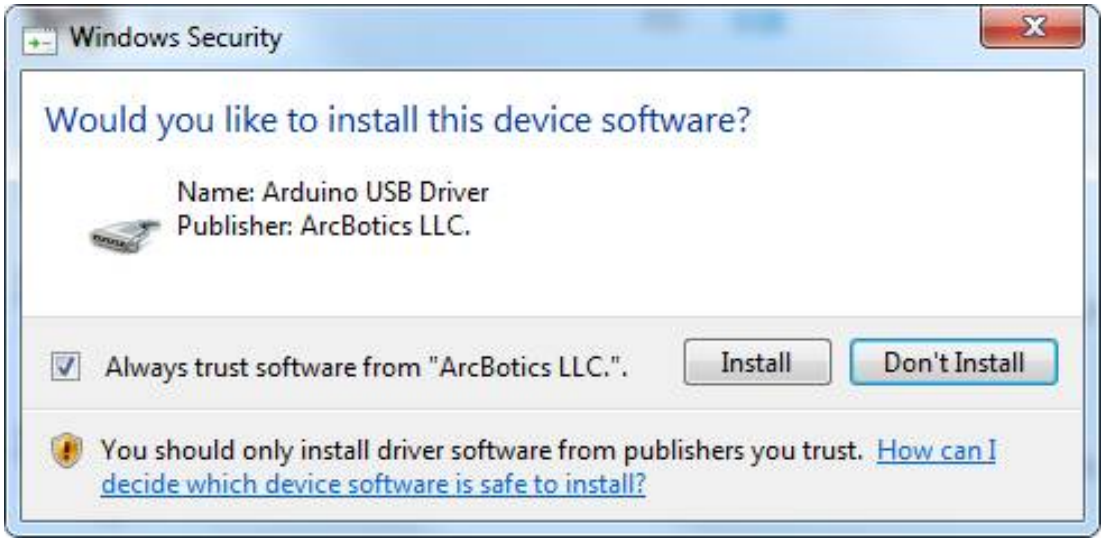
Choose the destination folder where you want the software to be located.



Click Install to initiate installation.

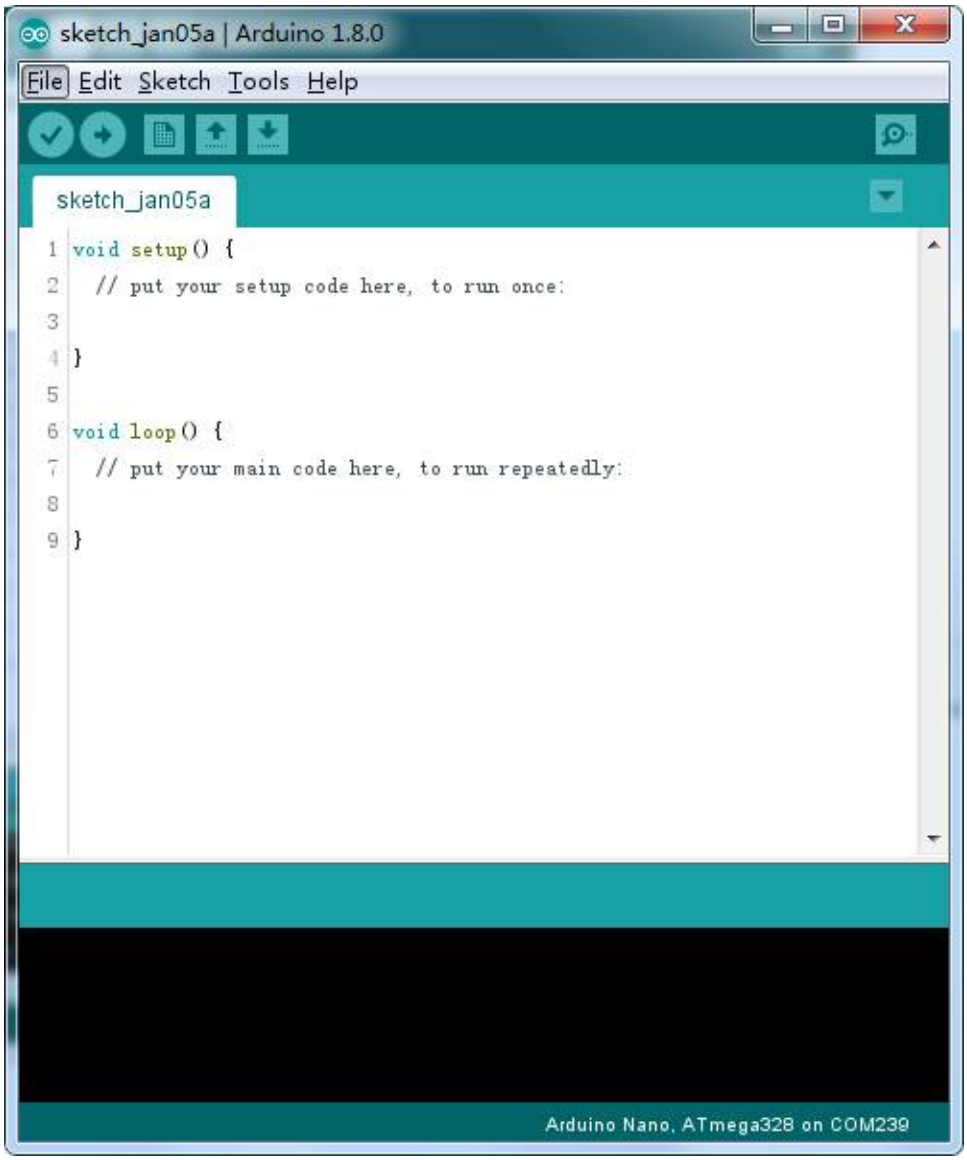


Finish the installation.



You will see this icon appear on your desktop.

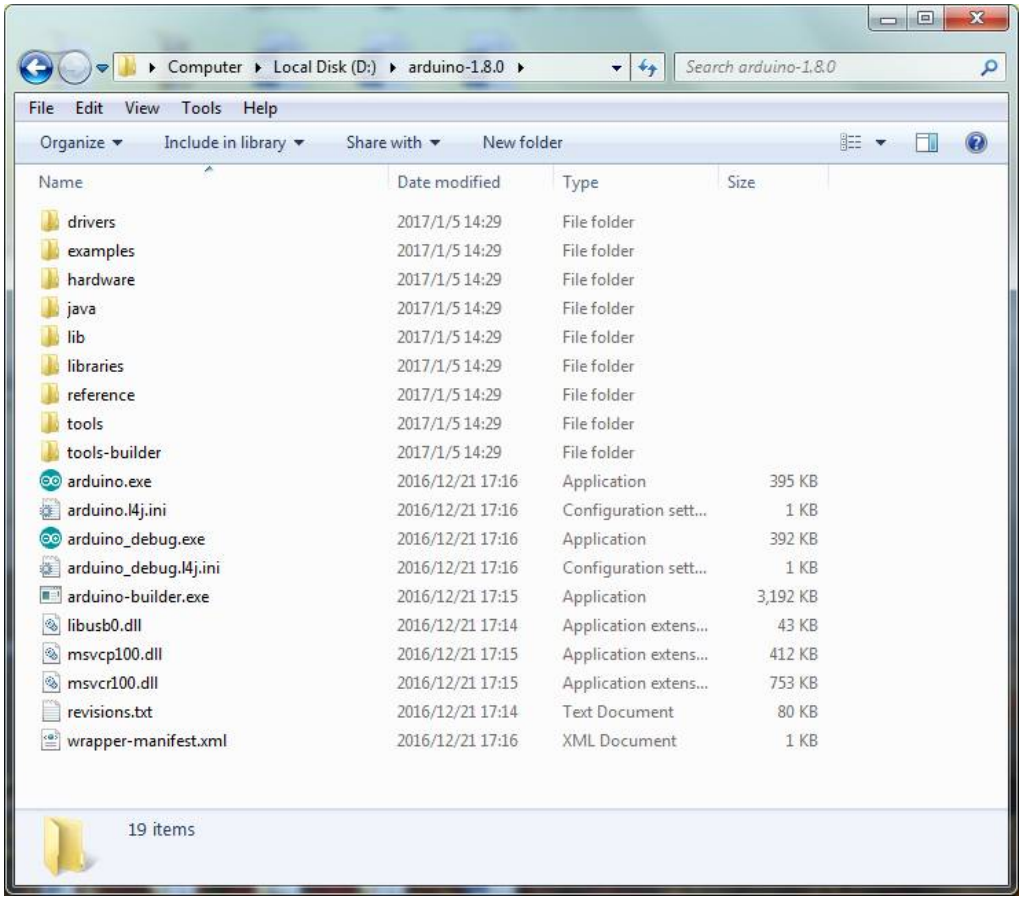
Now, open the icon, Arduino IDE.

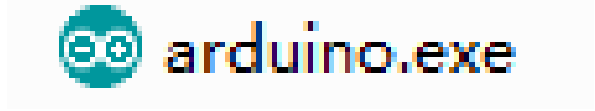


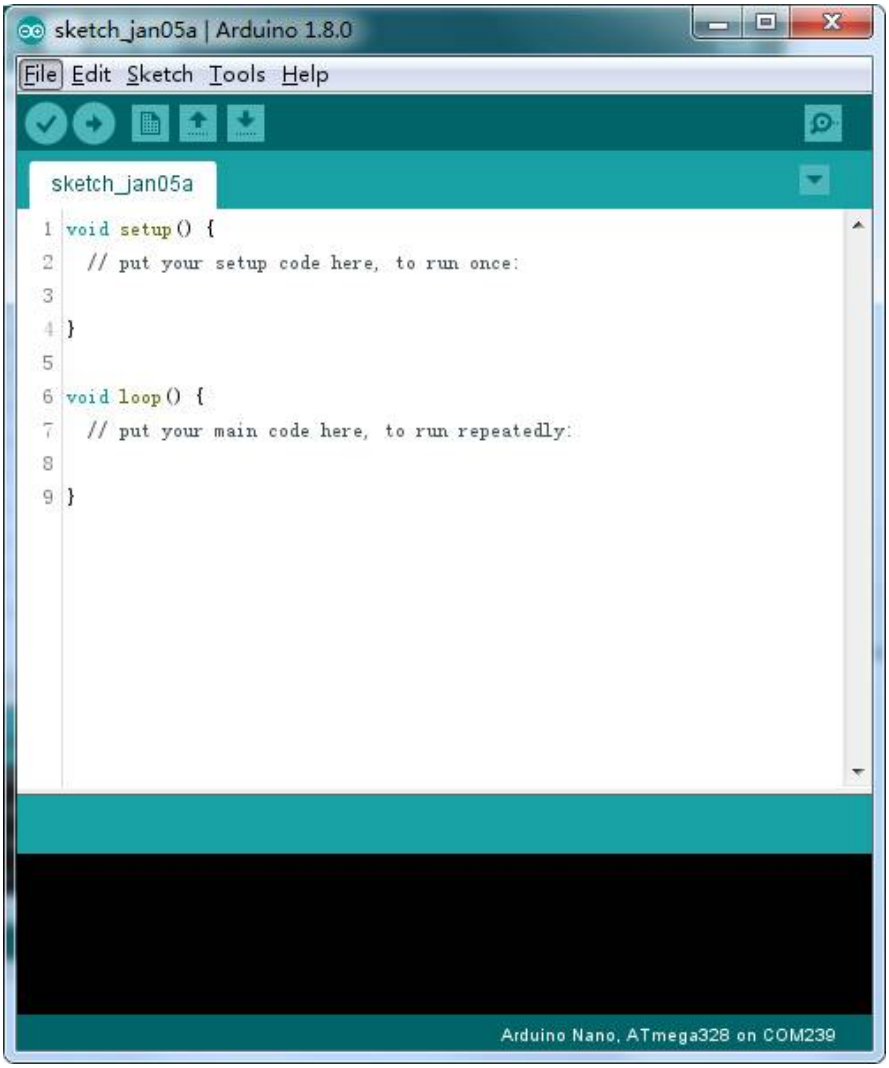
If you will directly choose the installation package, skip the contents below and jump to the next section. For different methods of installation, please continue reading.

Extract or unzip the downloaded zip file, open the program and enter the development environment. This installation method requires separate installation of the drivers.


arduino-1.8.0-windows.zip


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


arduino.exe

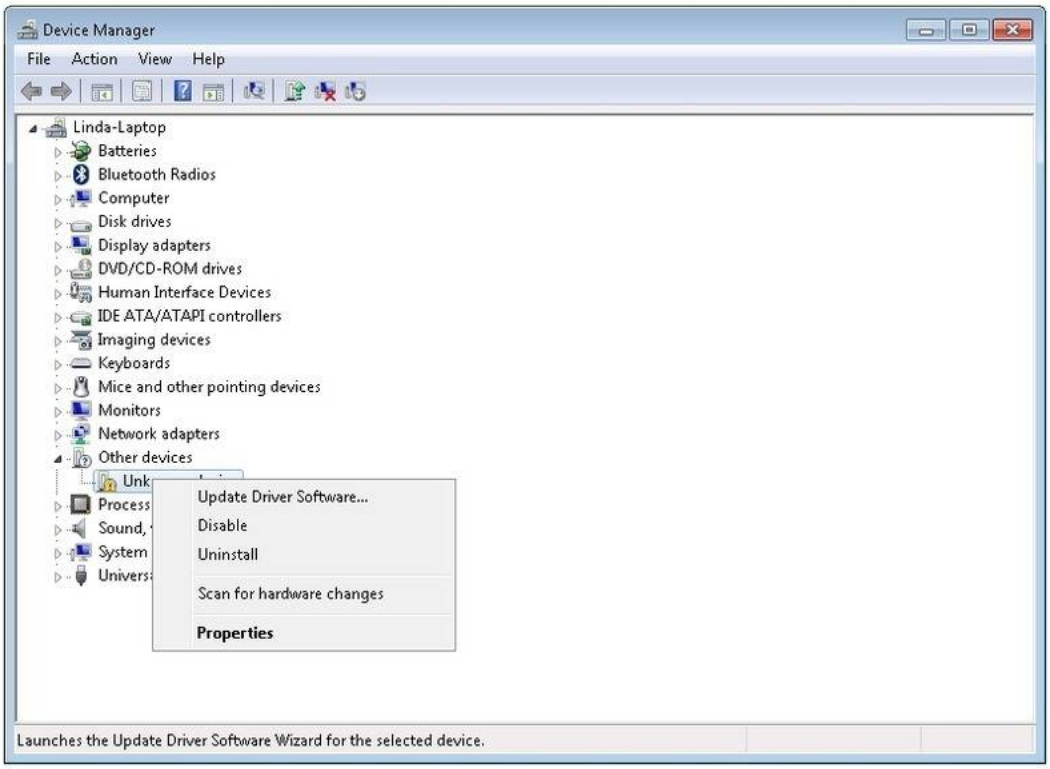


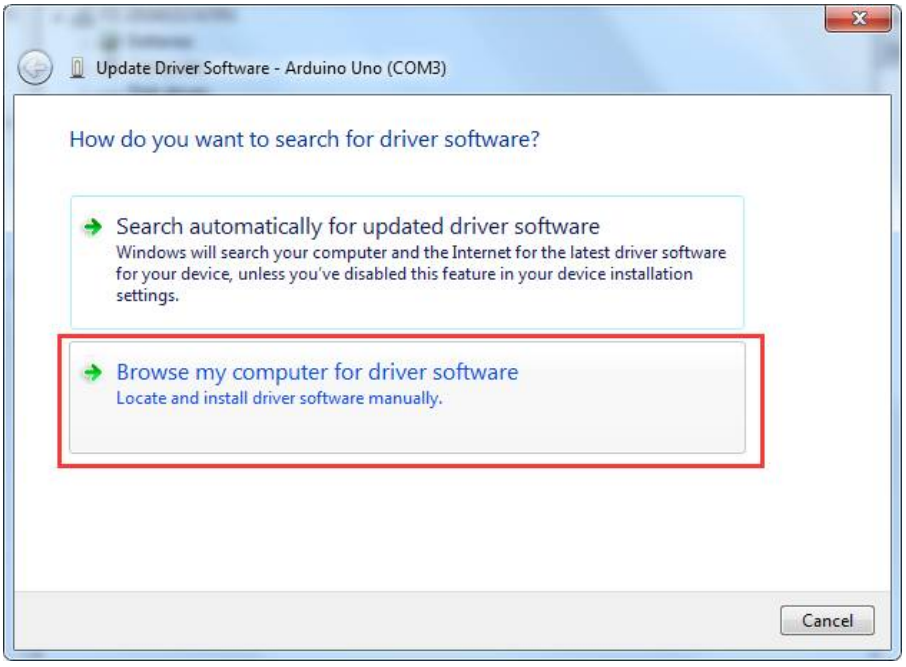
The software and the drivers that allow the Arduino to be connected to your computer can be both found inside the Arduino folder. Before launching, you need to install the USB drivers.

Connect your Arduino to the USB socket of your computer using a USB cable. The LED on your device will light up and a 'Found New Hardware' message from Windows may appear. Close the message and do not let the Windows install the drivers automatically.

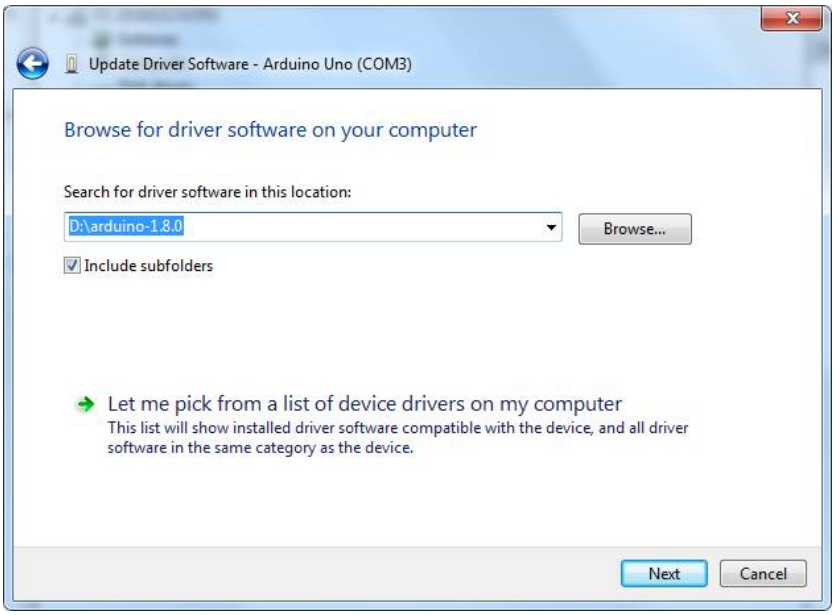
The best way to install the needed drivers is using the Device Manager. In Windows 7, open the Control Panel, and you should find the Device Manager in the list. The method on how to do this depends on your version of Windows.

In ‘Other Devices’, your Arduino is symbolized by an icon for ‘unknown device’ with a little yellow warning triangle next to it.





Select, then right-click on the device and choose Update Driver Software. Select 'Browse my computer for driver software'option and go to X\arduino1.8.0\drivers.

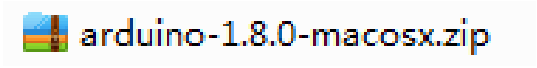


Click 'Next' and allow the software to be installed. You will get a confirmation message when the installation is successful.

The following installation instructions are for Mac and Linux users so Windows users may proceed to Lesson 1.

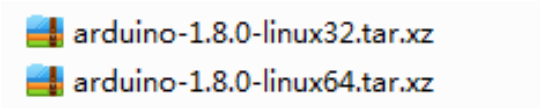
1.3 Installing Arduino (Mac OS X)

Download and unzip installation file, open the Arduino IDE; the system will request you to install Java runtime library if you don't have it. You can run the Arduino IDE once the installation is complete.

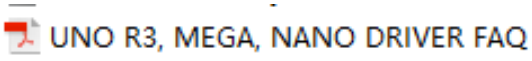


1.4 Installing Arduino (Linux)

You need to use the make install command. It is recommended to install the Arduino IDE from the software center of Ubuntu, if you are using that particular system.



TIPS: If you encounter problems while installing the drivers, please see UNO R3, MEGA, NANO DRIVER FAQ document.



2. Lesson 1: Add Libraries and Open Serial Monitor

2.1 Installing Additional Arduino Libraries

By now, you should be familiar with the Arduino software and you may want to explore capabilities of your Arduino with additional libraries.

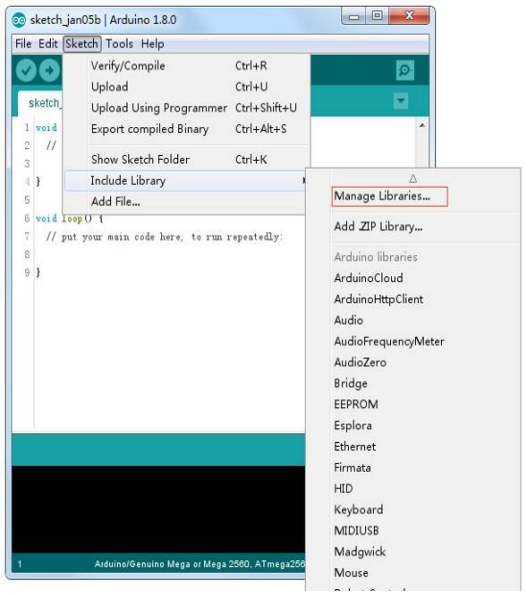
2.2 What are Libraries?

Libraries consist of lots of code, written to make it easier for you to connect to a sensor, display, module, etc. As an example, the built-in LiquidCrystal library makes it easy to use LCD displays. On the internet, there are hundreds of additional libraries available for download. You will need to install those additional libraries in order to use them.

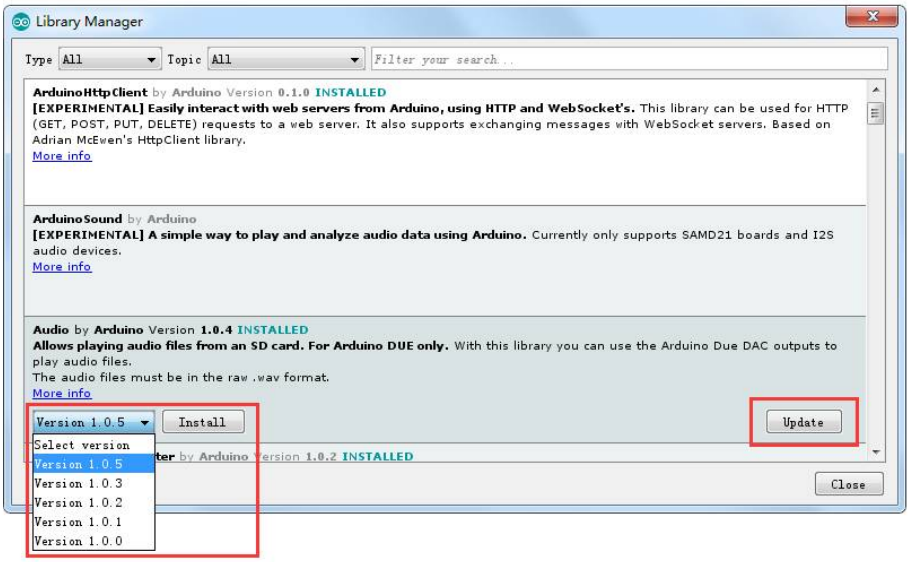
2.3 How to Install a Library

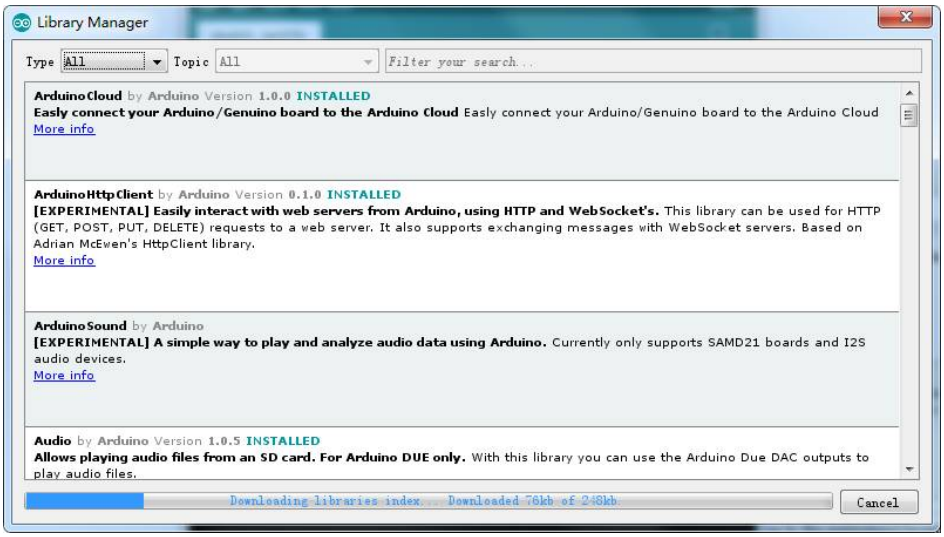
Using the Library Manager

You can install a new library through the Library Manager (available from IDE version 1.8.0). Open the IDE and on the "Sketch" menu, choose Include Library > Manage Libraries.

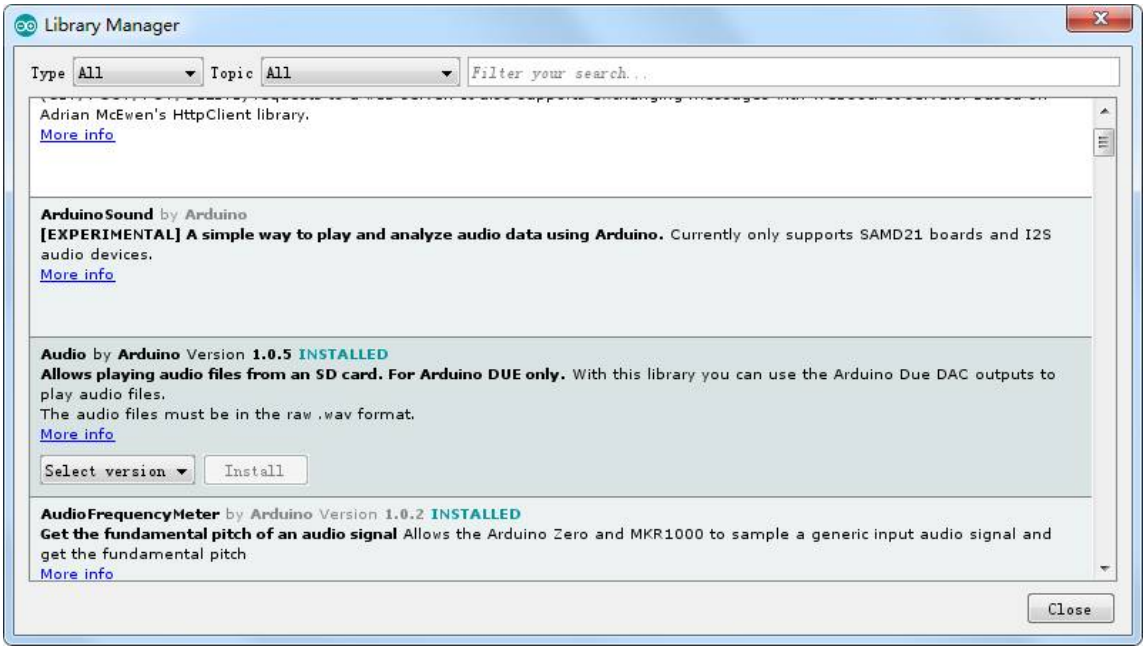


When the library manager opens, a list of installed libraries will be displayed as well as those libraries that are ready for installation. Take for example the installation of the Bridge library. Look for it in the list and then install the version of the library that you prefer. There are times that version selection menu does not appear, but that is normal.





Click “Install” and wait for the new library to be installed. Once the process has finished, you can see an INSTALLED tag next to the Bridge library.

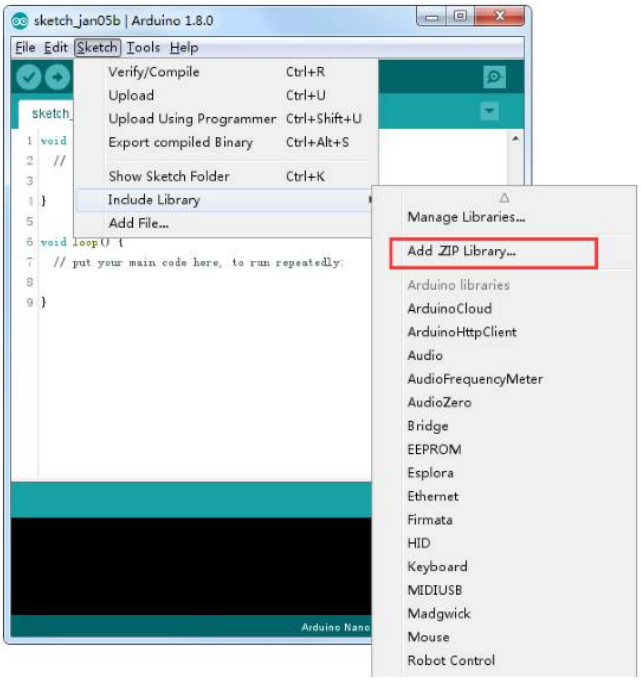


The new library can now be located inside the Include Library menu. You can open a new issue on Github, if you wish to add your own library.

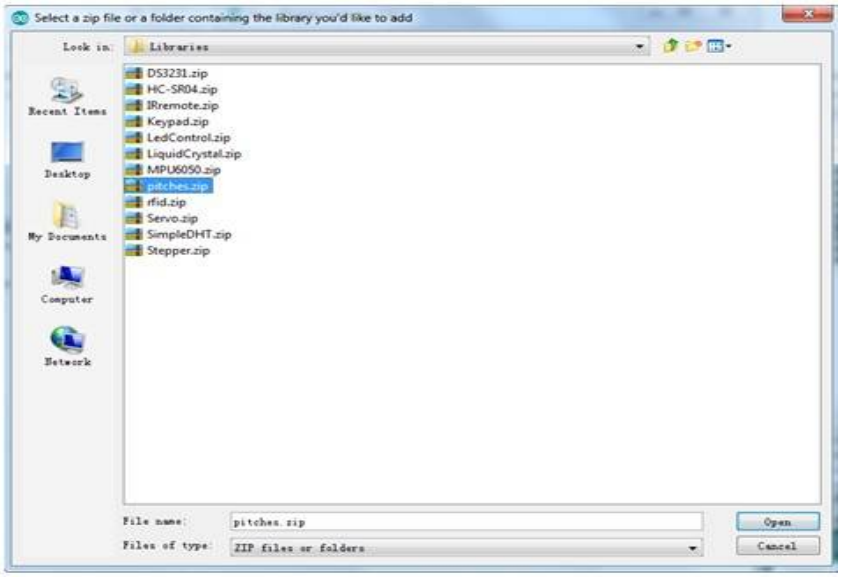
2.4 Importing a .zip Library

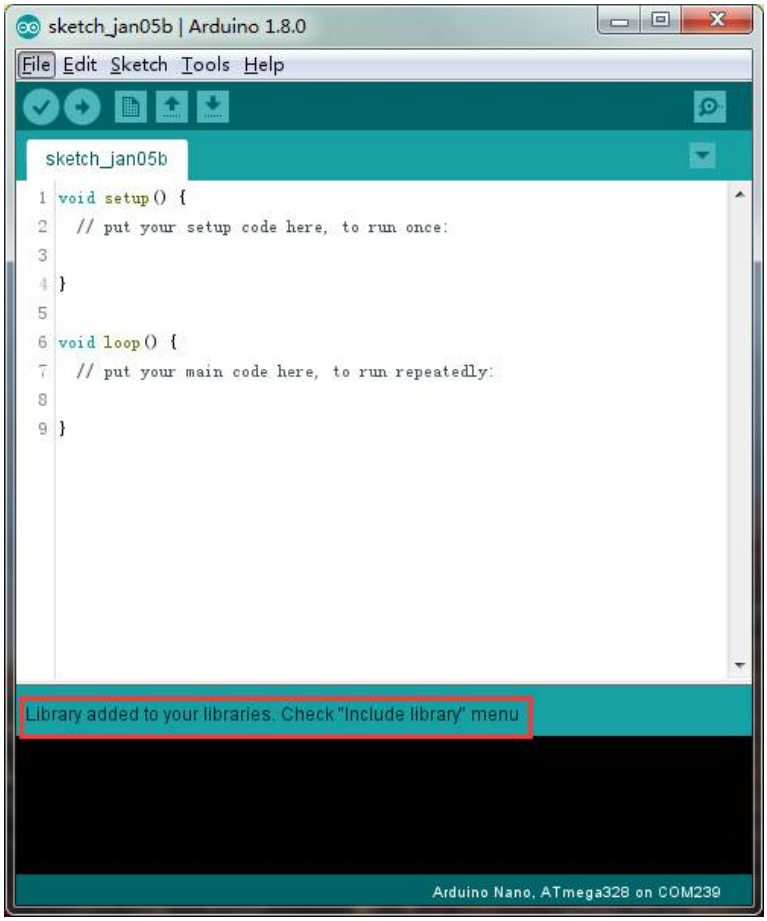
Libraries are most frequently distributed as a ZIP file or folder, with the name of the folder being the name of the library. Inside the folder, it will contain a .cpp file, a .h file and often a keywords.txt file, an examples folder, and any other files that are required by the library. Starting with version 1.0.5, 3rd party libraries can be installed in the IDE, without the need to unzip the downloaded library.

In the Arduino IDE, click Sketch menu > Include Library. Select "Add .ZIP Library".



Next step is you will need to select the library you would like to add. Go to the .zip file's location and open it.





Go back to the Sketch > Import Library menu. The library can now be seen at the bottom of the menu, indicating that it is ready to be used. The zip file will be present in the libraries folder in your Arduino sketches directory.

NB: The Library will be available to use in sketches, but examples will be displayed in the File > Examples only after the IDE restarts..

Those two are the most common method, with MAC and Linux systems being handled similarly. You can also proceed to manual installation, as presented below, but it is seldom used and you may skip it.

2.5 Manual installation

To install the library, first make sure the Arduino application is closed. Extract the files from the ZIP folder, you should now have a folder with the name of the library, containing files with the extensions .cpp and.h inside, with a few other additional ones. (In the case that these files are not there, you need to create them yourself).

Let's take for example a library called ArduinoLove. Drag the ArduinoLove folder into your libraries folder. For Windows users, it usually appears as "My Documents\Arduino\libraries". For Mac and Linux users, it should be the 'libraries' folder where your Arduino documents are located.

Your Arduino library folder will be like this (on Windows):

My Documents\Arduino\libraries\ArduinoLove\ArduinoLove.cpp
My Documents\Arduino\libraries\ArduinoLove\ArduinoLove.h
My Documents\Arduino\libraries\ArduinoLove\examples

or like this (on Mac and Linux):

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

There are often more files other than the .cpp and .h, you just have to ensure that they can be found there (if you put the .cpp and .h files directly into the libraries folder, the library may not work or if they're inside an extra folder).

For example:

Documents\Arduino\libraries\ArduinoLove.cpp and
Documents\Arduino\libraries\ArduinoLove\ArduinoLove\ArduinoLove.cpp won't work.

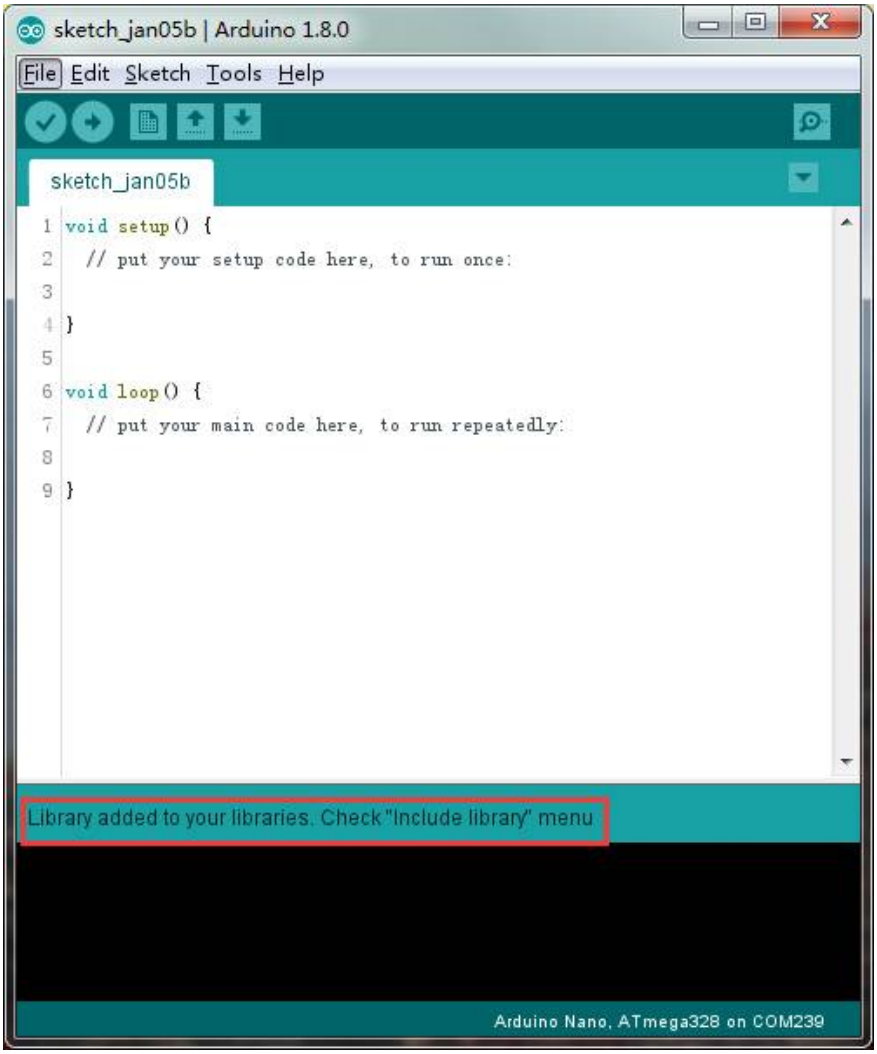
Restart the Arduino application. The new library should now appear in the Sketch->Import Library menu item of the software. Congratulations, you've installed a library!

2.6 Arduino Serial Monitor (Windows, Mac, Linux)

As mentioned in the beginning, the software side of the Arduino platform is called the Arduino Integrated Development Environment (IDE). In this software, a serial terminal is included because using such terminal is such a major part of working with an Arduino or other microcontrollers. The terminal is called the Serial Monitor.

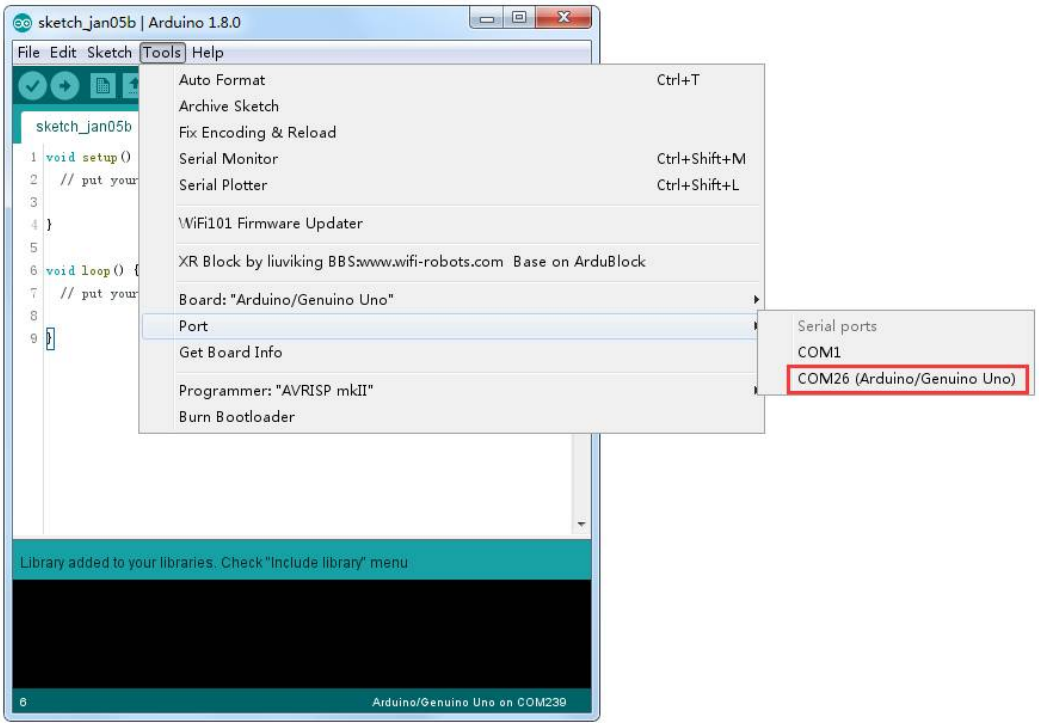
2.7 Making a Connection

The serial monitor comes with all versions of the Arduino IDE. Click the Serial Monitor icon to open it.

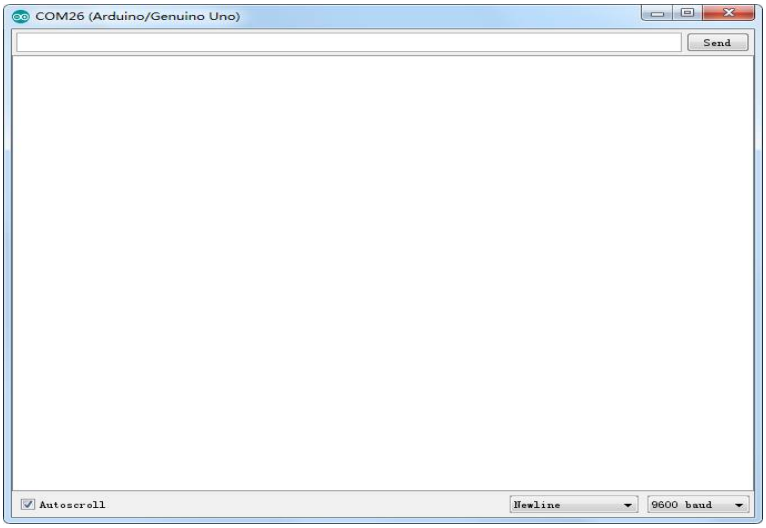


Choosing which port to open in the Serial Monitor is very similar to choosing a port for uploading your code. To do that, go to Tools -> Serial Port, and select the adequate port.

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

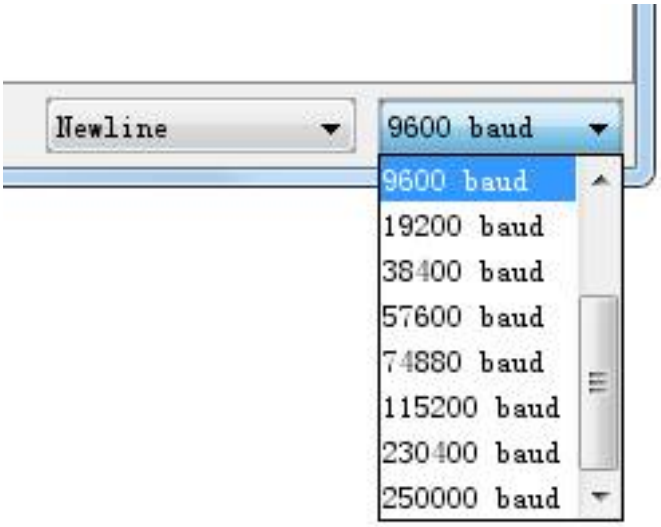


Once it is open, you should see something like below:

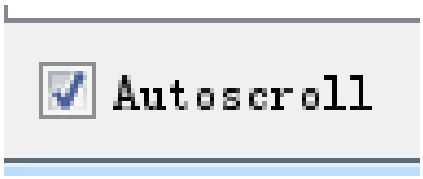


2.8 Settings

There are limited settings on the Serial Monitor, but enough to handle most of serial communication needs.. The baud rate is the first setting you can change. Choose the correct baud rate (9600 baud) using the drop-down menu on selecting baud rate.



You can also set the terminal to Autoscroll by checking the box beside the Autoscroll option.



Pros

The Serial Monitor is a quick and easy way to create a serial connection with your Arduino. But if you are already working in the Arduino IDE, there's no need to set up a separate terminal to display data.

Cons

There is a lack of settings in the Serial Monitor, so it is not suitable for advanced serial communications.

3. Lesson 2 Blink

3.1 Overview

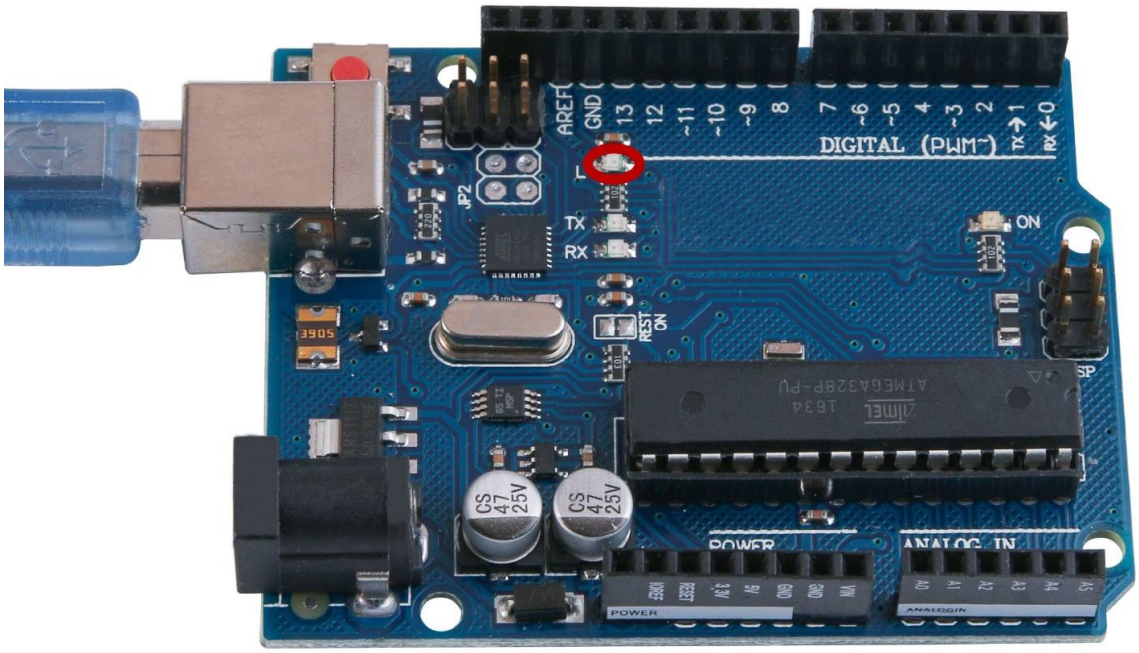
This chapter will teach you to program the UNO R3 controller board to blink the built-in LED in your Arduino and the fundamental steps in downloading programs.

3.2 Component Required

(1)x Plusivo Uno R3 board

3.3 Principle

You can find in the UNO R3 board, along both sides, the rows of connectors. These are used to connect to several electronic devices and plug-in 'shields' that extend its capability. It also has a single LED, built onto the board, that you can control from your sketches,. This is often referred to as the 'L' LED, same as how it is labelled on the board.



When you connect your board to a USB plug, you should see that the 'L' LED already blinks, as the 'Blink' sketch is generally pre-installed in the board.

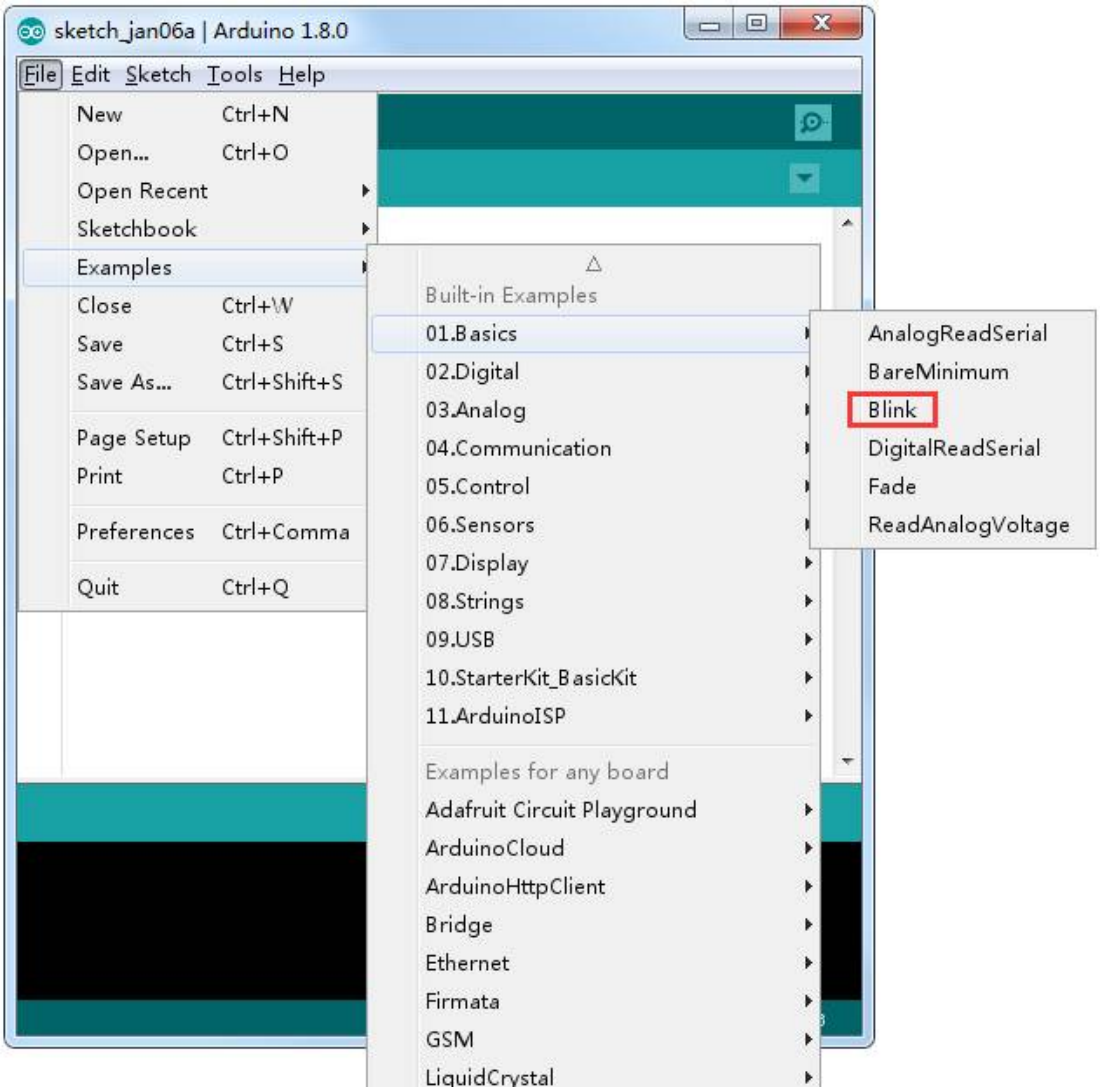
In this chapter, we will rewrite our own Blink sketch and then change the rate at which it blinks.

Blink

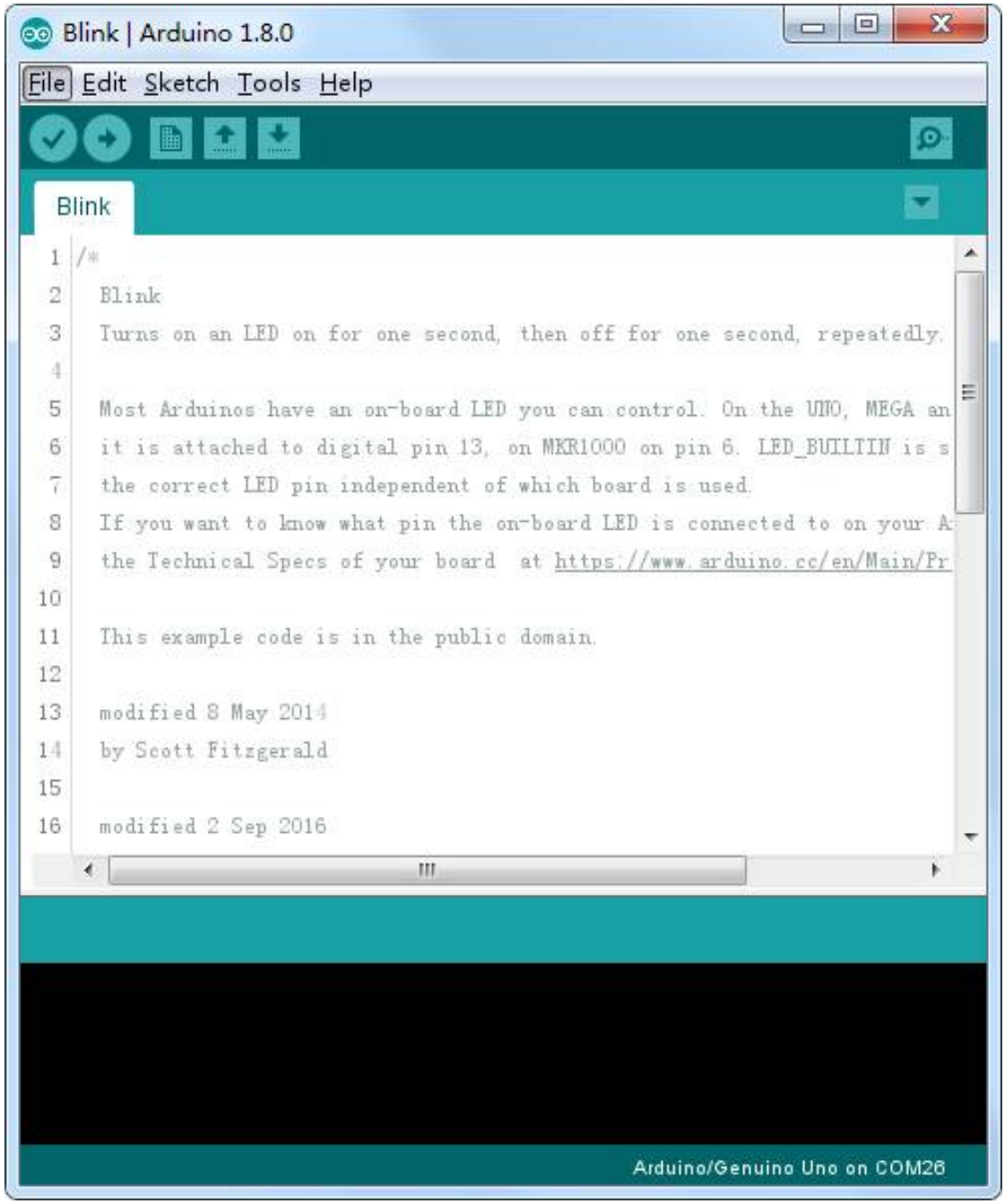
In the first lesson, we covered how to set up your Arduino IDE and ensure that you will be able to find the right serial port for it to connect to your UNO R3 board. Now it is time to test that connection and program your UNO R3 board.

The Arduino IDE consists of a large collection of example sketches that you can load up and use, including an example sketch on how to make the 'L' LED blink.

In the IDE's menu system folder, go to File >Examples > 01.Basics and load the 'Blink' sketch.

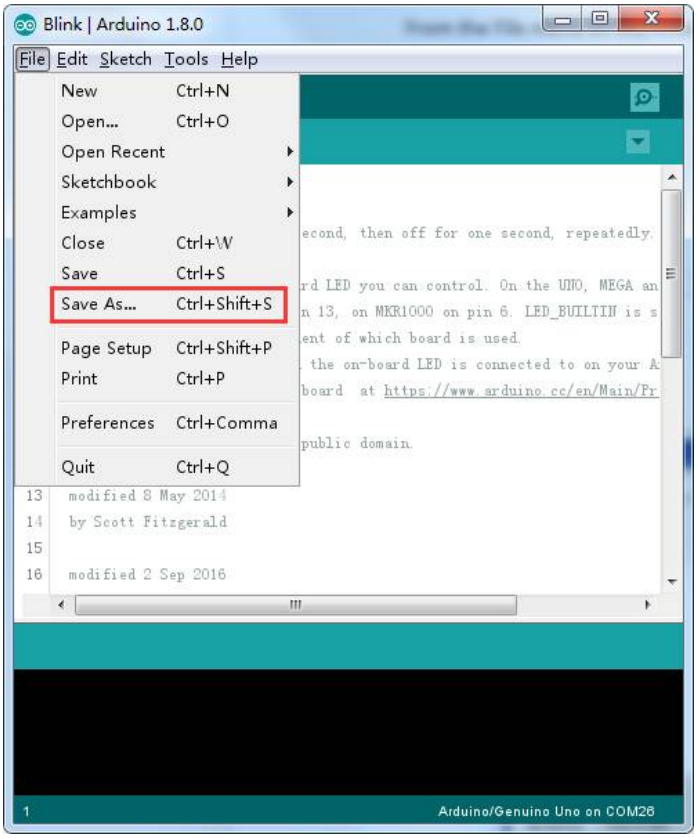


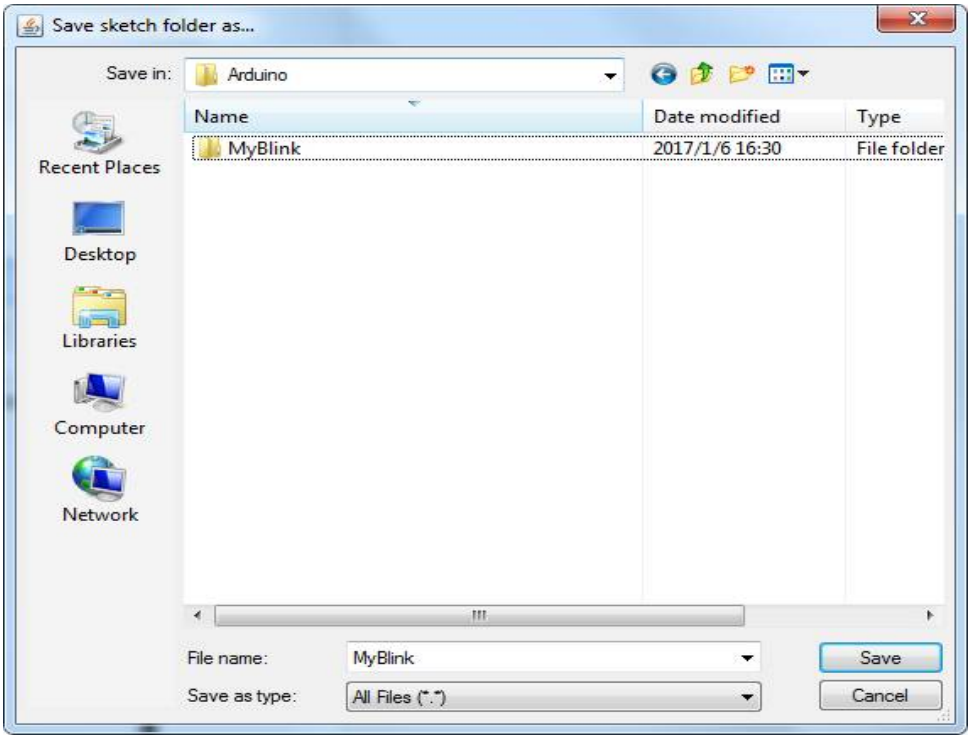
Enlarge the sketch window to see the entire sketch.



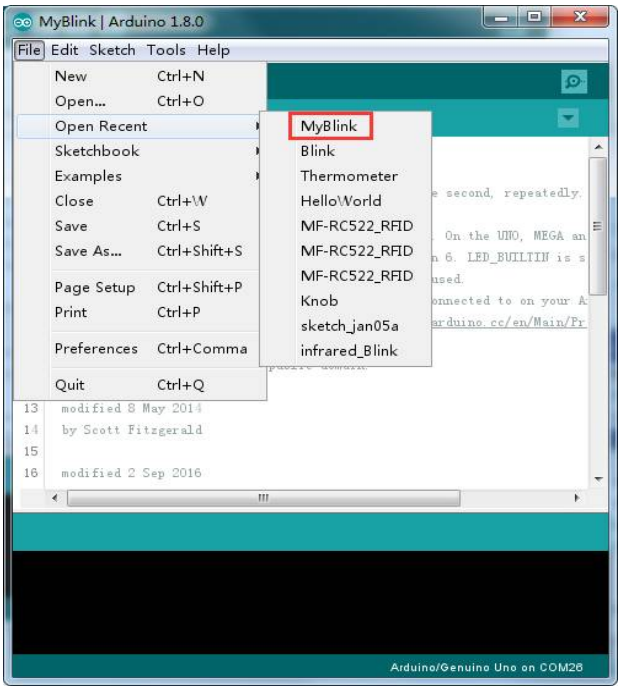
The example sketches are 'read-only', you can change them but cannot be saved as the same file. Thus, first you have to save your own copy as a new file since we are going to change this sketch.

On the Arduino IDE File menu, select 'Save As..', and save it with filename 'MyBlink'.

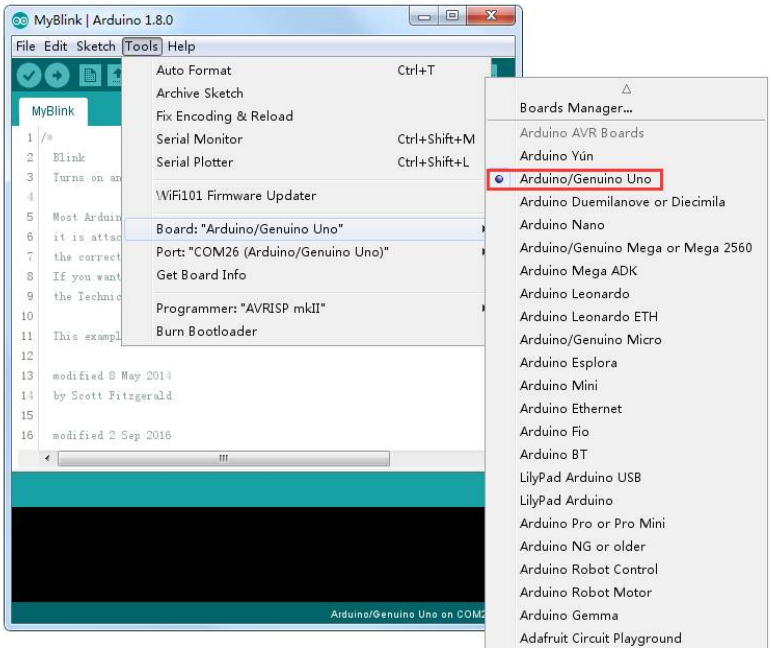


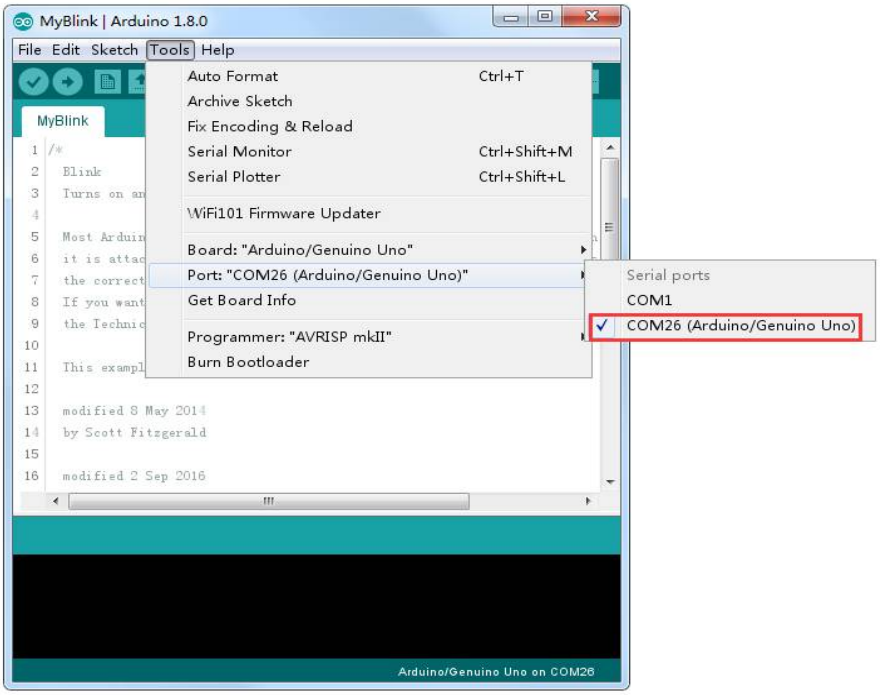


You have just saved your copy for blinking the 'L' LED as 'MyBlink' in your sketchbook, and when you want to use it you can just open it from the File > Sketchbook menu option.



Using USB cable, connect your Arduino board to your computer and check that both the 'Board Type' and 'Serial Port' are set correctly.

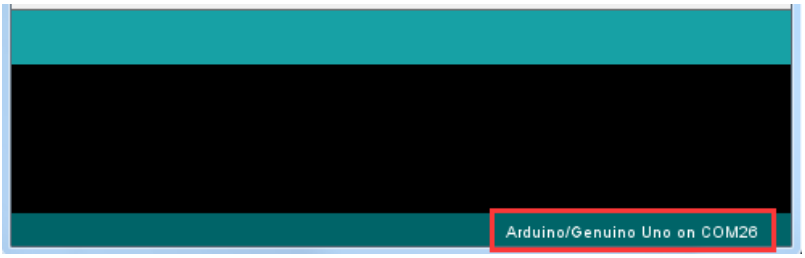




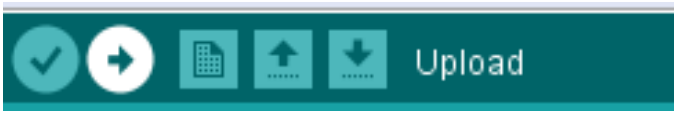
Note: The Board Type and Serial Port shown here may appear differently as the Serial Port displayed for everyone is different. COM26 that is shown here may be COM3 or COM4 on your

computer. The right COM port should be COMX (arduino XXX), chosen according to the certification criteria. Also, if you are using 2560, the Board type you will have to choose is Mega 2560, or choose the type adequate to your situation.

The current settings for the board are shown at the bottom corner of the window.



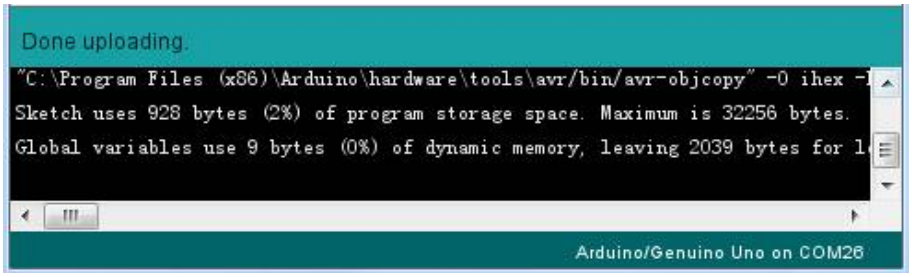
Click on the 'Upload' button, the second icon on the toolbar.



On the status area of the IDE, a progress bar and a series of messages will be shown. It will say 'Compiling Sketch...' in the beginning, indicating that it is converting the sketch into a suitable format for uploading to the board.

Next, the status will become 'Uploading'. During this time, the sketch is being transferred and it would cause the LEDs on the Arduino to start flickering

Lastly, the status will be 'Done uploading'.



There is also a message that says that the sketch is using 928 bytes of the 32,256 bytes available. After the 'Compiling Sketch..' stage, the following error message may appear:

This can mean that your board is not properly connected, that the necessary drivers have not been installed or maybe you selected a wrong serial port. If you encounter this error, you should revisit Lesson 0. Once the upload is complete, the board should restart and start blinking.

Please note that a big part of this sketch is composed of comments. These are not actual program instructions, but an explanation on how the program works. A block comment is everything between /* and /*, and a single line comment is the one that starts with //. Everything up until the end of that line is considered a comment.

Below is the first line of code:

```
int led = 13;
```

This code is giving a name to the pin where the LED is attached to. On most Arduinos, including the UNO and Leonardo, this is pin 13. Then, we have the 'setup' function. This is executed when the reset button is pressed, and also whenever the board resets for any reason 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 you can add other instructions between

the { and the } brackets. In this case, there is only one command telling the Arduino that we are using the LED pin as an output. To have a 'loop' function is also mandatory for a sketch. After a reset, the 'loop' function will immediately start again after it has finished running its commands. This is unlike the 'setup' function that only runs once.

```
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 turn the LED pin on (HIGH), then 'delay' for 1000 milliseconds (1 second), then turn the LED pin off (LOW) and pause for another second.

To make your LED blink faster, you need to change the parameter in the brackets () 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 }
```

If you want your LED to blink twice as fast, change the value from 1000 to 500 milliseconds. The setting would then pause for half a second each delay rather than a whole second.

Upload the sketch again and your LED should now start to blink more quickly.

4. Lesson 3 LED

4.1 Overview

Here, you will learn how to change the brightness of a LED using different values of resistors.

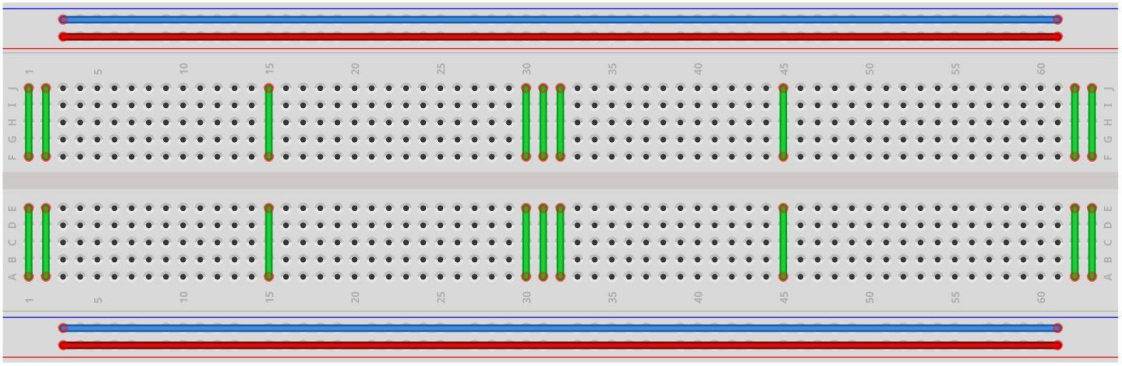
4.2 Components Required

(1) x Plusivo Uno R3
(1) x 5mm red LED
(1) x 220 ohm resistor
(1) x 1k ohm resistor
(1) x 10k ohm resistor
(2) x M-M wires (Male to Male jumper wires)

4.3 Component Introduction

BREADBOARD MB-102

In a breadboard, you can prototype your circuits quickly without the need to solder the connections. You can see below an example of a breadboard.



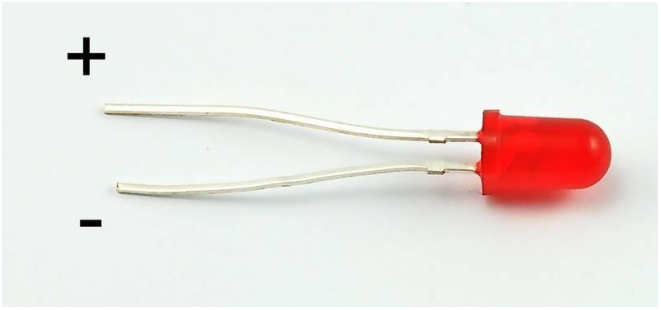
These breadboards has various sizes and configurations. They simplest is just a grid of holes in a plastic block, where strips of metal inside provide electrical connection between the holes in the shorter rows. If you push the legs of two different components into the same row, they will be joined together electrically. The deep channel running down the middle indicates a break in connections there, which means placing a chip in with the legs at either side of the channel does not connect them together.

Some breadboards have two strips of holes (also called rails) running along the long edges of the board separated from the main grid, with strips running down the length of the board inside that enable you to connect at a common voltage. They are usually for +5 volts and ground.

While breadboards are great for prototyping, they have some limitations due to potential poor connections. Because the connections are temporary, they are not as reliable as soldered connections.

LED

LEDs are great for making indicator lights as they use very little electricity and they last longer. In this lesson, the most common of all LEDs will be used: a 5mm-diameter red LED (5mm LED). You cannot connect it directly to a battery or voltage source, as the LED has a positive and a negative lead and won't light if connected the wrong way and a resistor must be used with it to limit the current flowing through it so that it won't burn out.



Not using a resistor with a LED will instantly destroyed it, as the excessive current flowing through will harm the 'junction' where the light is produced.

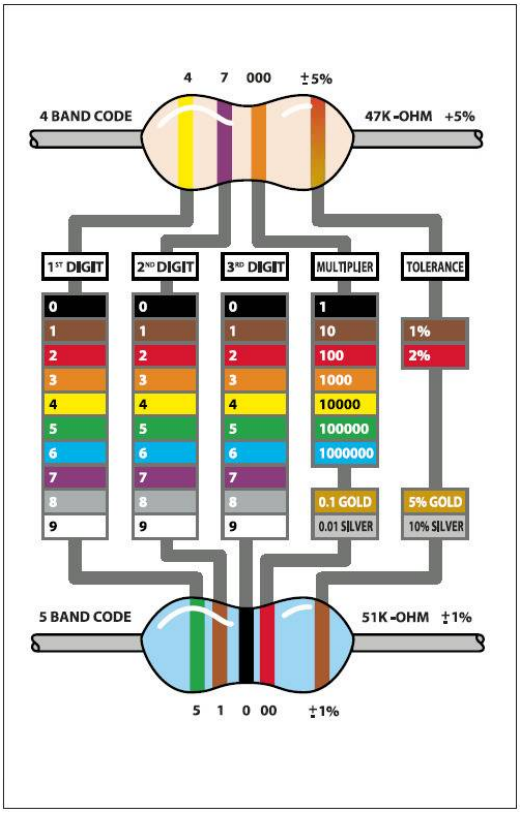
There are two ways to identify the two leads (positive and negative) of the LED. First, the longer lead is the positive lead. Second, you can see a flat edge to the case of the LED where the negative lead enters the body of the LED. If you have a LED with flat side next to the longer lead, it can be assumed that the longer lead is positive.

RESISTORS

Resistors resist the flow of electricity, which means higher value of the resistor means that the more it can resist and less electrical current will flow through it. We use these to control how much electricity flows through the LED and eventually, how brightly the LED will shine.

The unit of resistance is Ohm with symbol Ω, from the Greek letter Omega. We also represent the values of resistors in kΩ (1,000 Ω) and MΩ (1,000,000 Ω), called kilo-ohms and mega-ohms.

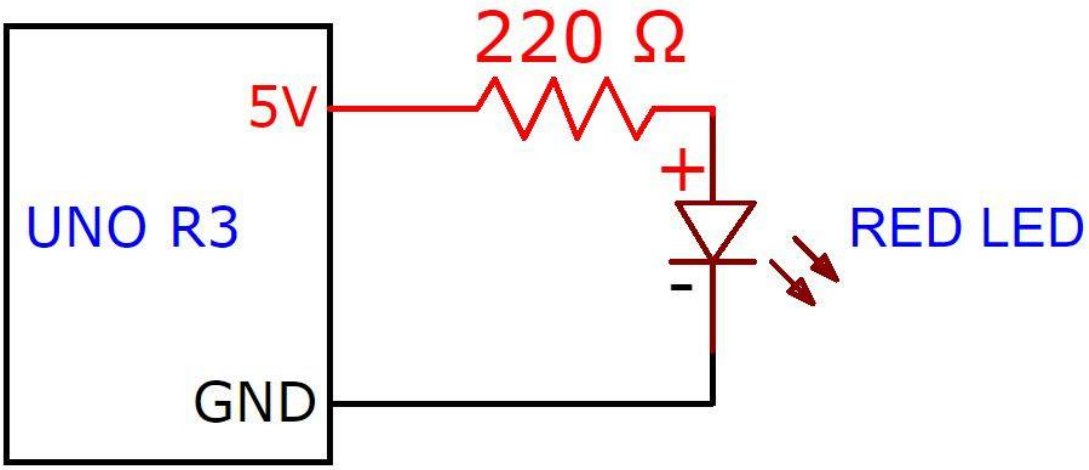
In this lesson, three different values of resistor will be used: 220Ω, 1kΩ and 10kΩ. These have the same appearance but have different colored stripes on them, which indicates the value of the resistor. The color code of a resistor is a three-colored stripes and a gold stripe at one end as seen below.



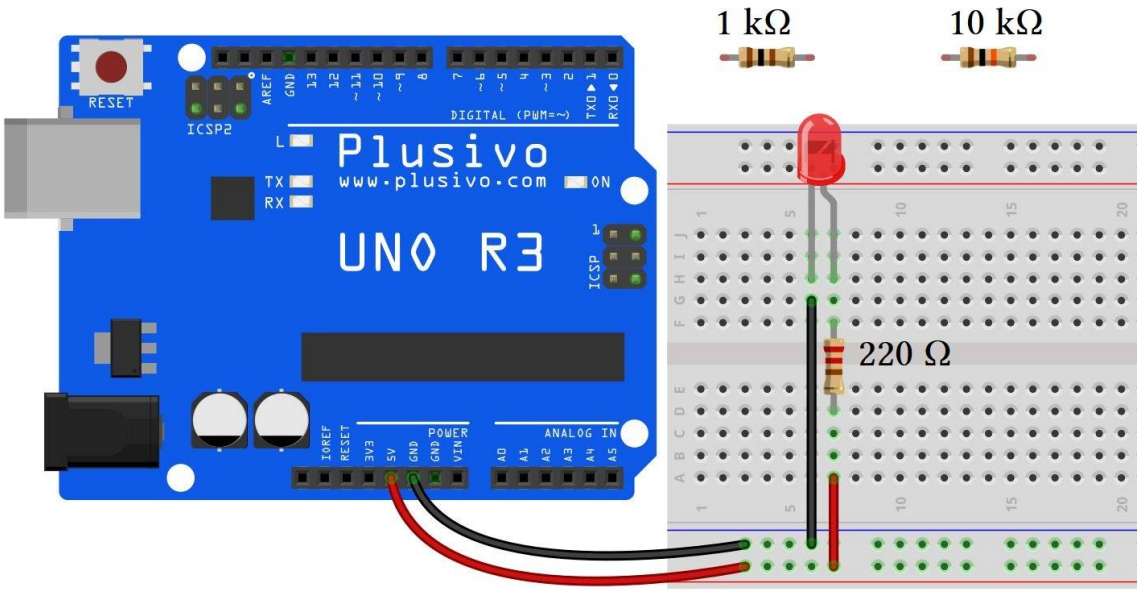
Resistors, unlike LEDs, can be connected either way as they don't have a positive and negative lead. If you find this colored lines method too complicated, you can use a digital multimeter to determine the resistance value of a resistor.

4.4 Connection

Schematic



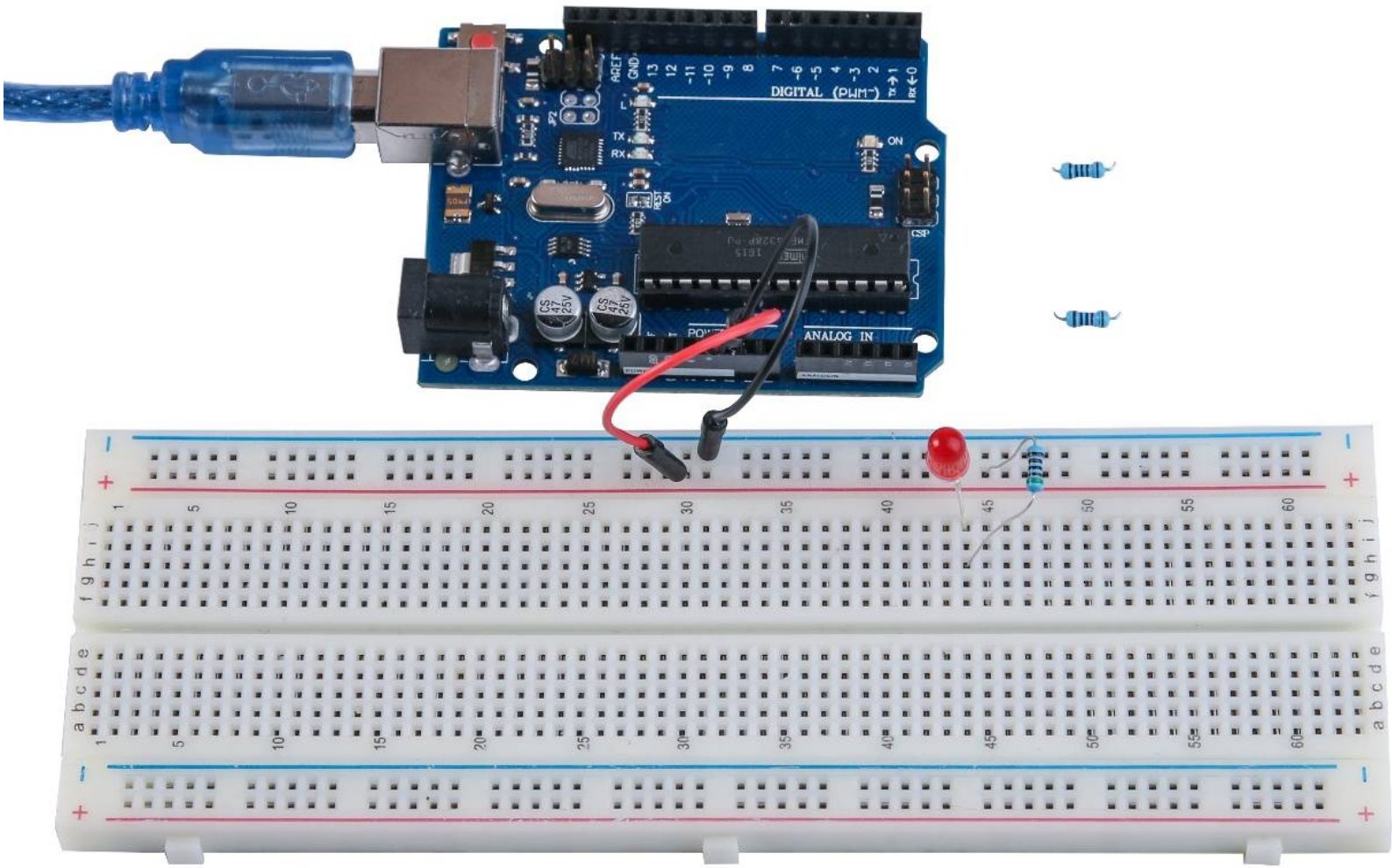
Wiring diagram



Plug your UNO (a convenient source of 5 volts) into your computer as this will be used to provide power to the LED and the resistor. The LED should be quite bright with the 220 Ω resistor in place. Alternatively, by using a 1kΩ resistor, the LED will appear a little dimmer. Finally, when you use the 10 kΩ resistor, the LED will be barely visible. To notice the difference, you can use the red jumber as a switch by pulling the red jumper lead out of the breadboard, touching it into the hole and immediately removing it.

Now, you have the 5V going to one leg of the resistor, then the other leg of the resistor going to the positive leg of the LED and the other leg of the LED going to GND. If we reposition the resistor and place it after the LED, as shown below, the LED will still light, because it won't matter where we put the resistor, on either side of the LED, as long as it is there.

4.5 Example picture



5. Lesson 4 RGB LED

5.1 Overview

These RGB LEDs are a fun and easy way to add some color to your projects. Using them is easy and connecting them is pretty much the same because they are just like 3 regular LEDs in one and they mostly come in Common Anode or Common Cathode versions

The Common Anode connects to the 5 V on the common pin and Common Cathode connects to ground. Just like with any LED, we need to limit the current being drawn, so we need to connect some resistors inline (3 total).

In the sketch that we will do, we will start in Red color state of the LED, then will fade to Green, then fade to Blue and finally back to the Red color state. Through this, we will be able to cycle through most of the colors.

5.2 Components Required

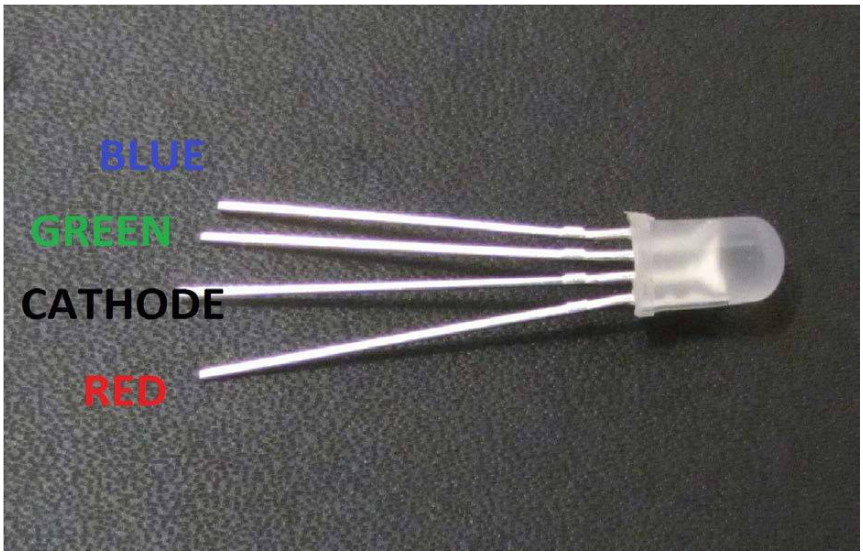
(1) x Plusivo Uno R3
(1) x 830 Tie Points Breadboard
(4) x M-M wires (Male to Male jumper wires)
(1) x RGB LED
(3) x 220 ohm resistors

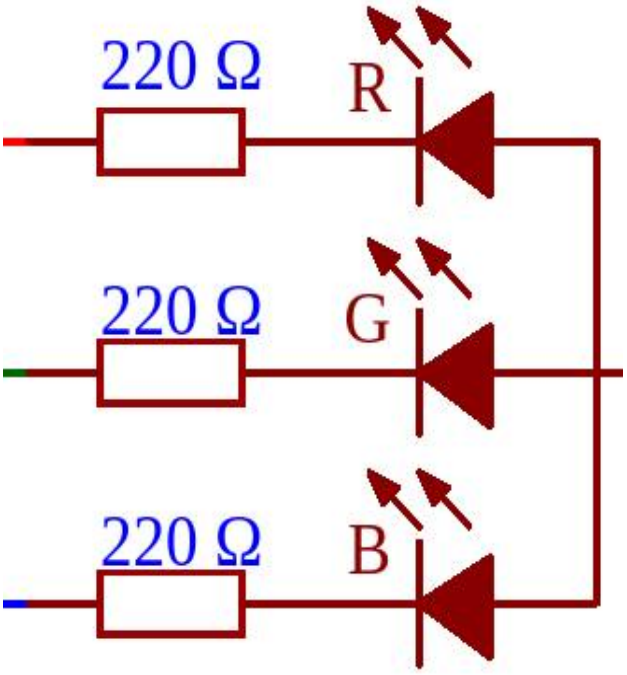
5.3 Component Introduction

RGB

If you take a look at RGB (Red, Green and Blue) LEDs, they look just like regular LEDs. But there are actually three LEDs inside the usual LED package, one of each of the primary colours (red, green, blue). You can can mix any color that you want by controlling the brightness of each of the individual LEDs.

How we mix colors of the LED is by adjusting the brightness of each of the three LEDs, just like how you mix paint on a palette. Or to use different value resistors or variable resistors, which is the harder way and a lot of work. Fortunately, the analogWrite function of UNO R3 board with pins marked with a ~ can be used to output a variable amount of power to the appropriate LEDs.



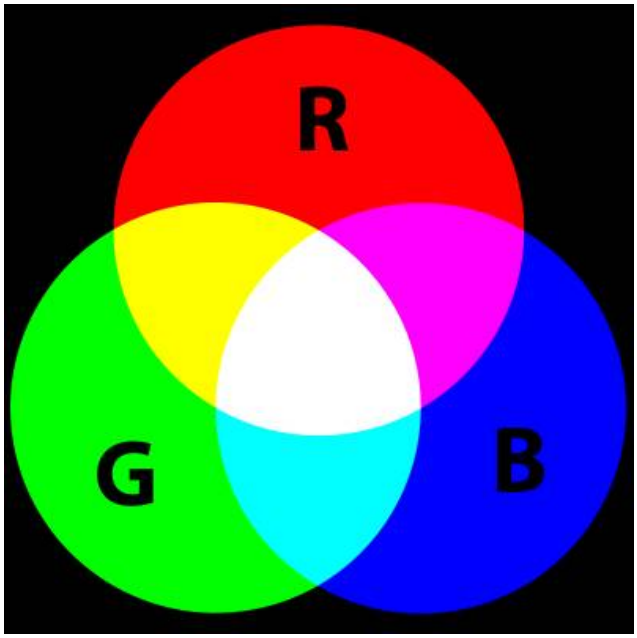


In the picture above, you can see 4 electrode LEDs with each pin for the Green, Blue or Red color called Anode. The Anode will always be connected to "+" (power), while Cathode goes to "-" (ground). The LED won't light if you connect it the other way. The second pin from the flat side is the common negative connection of the LED package. You can easily see it because it is also the longest among the four leads that will be connected to the ground. It is required that each LED inside the package will have its own 220Ω resistor in order to prevent too much current flowing through it. These resistors are used when you connect the three positive leads of the LEDs (one red, one green and one blue) to UNO output pins.

COLOR

By varying the quantities of red, green and blue light, you can mix any color that you like. Your eye has three types of light receptors (red, green and blue) and together with your brain, they can process the amounts of red, green and blue and convert it into a color of the spectrum.

In a way, we are playing a trick on the eye by using the three LEDs. This is also the mechanism used in TVs, where the red, green and blue color dots (in LCD) next to each other make a pixel.



For example, the overall color of the light will be white when we set the brightness of all three LEDs to be the same. By turning off the blue LED, the light will be yellow since there are just the red and green LEDs with the same brightness.

We can mix any color we like just by controlling the brightness of each of the red, green and blue parts of the LED separately.

Black is the absence of light, thus we can have black when we turn off all three colors of LED.

5.4 Common Anode vs Common Cathode

These two types of RGB LEDs do not connect in the same way eventhough they look the same. The UNO R3 can control both types of LEDs due to its symmetrical ability to source and sink exactly the same amount of current.

Common Cathode is straightforward, meaning, having a **higher** the current means a brighter coresponding LED. Here, the current is flowing from the board to the LED and is called **Current Sourcing.**

Common Anode is a bit different, which means, if the current is **lower**, it will result in a brighter corresponding LED. Here, the current is flowing from the LED to the board. and is called **Current Sinking**.

5.5 How to determine if the LED is common anode or common cathode

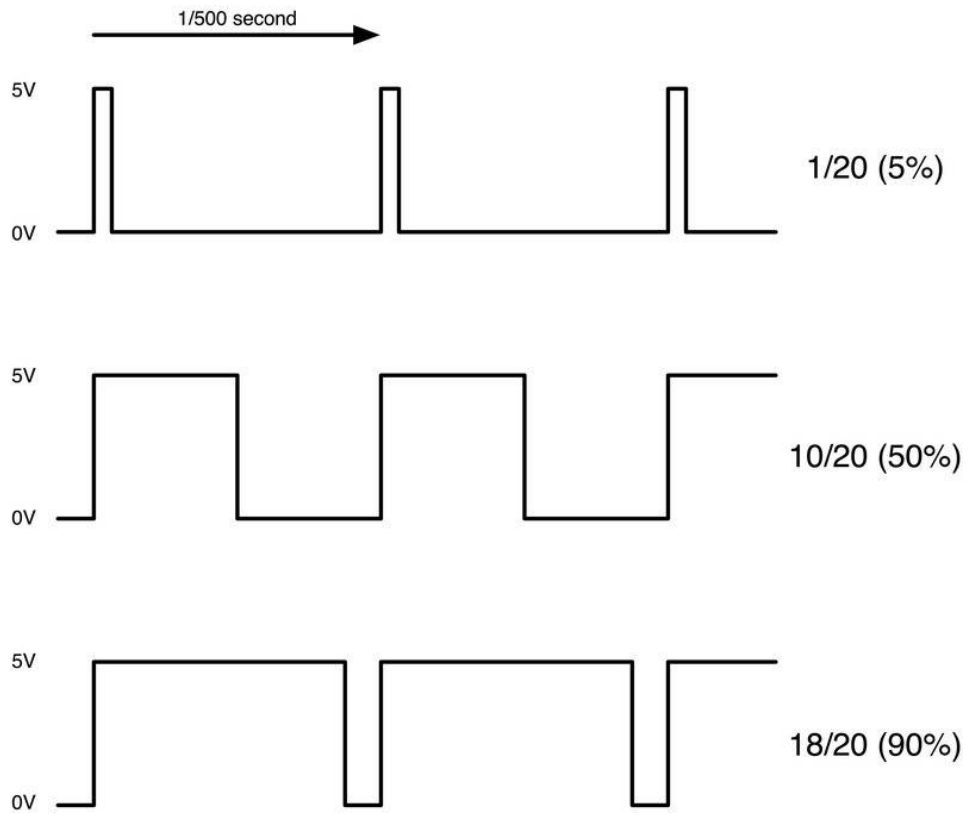
There is no visible difference between the common anode and common cathode, thus, to determine which is which is by testing it.

You may follow these steps to indentify if it is a common anode or common cathode:

- Power up the development board.
- Have the longest leg of the RGB LED connected to **GND.**
- Using a 220 Ω resistor in series, connect **QUICKLY (a fraction of a second)** one of the legs remaining to the **5 V**. It is better to mount all of them on a breadboard.
- It is a **COMMON CATHODE** if the LED lights**,** if not, it is **COMMON ANODE.**

5.6 Theory (PWM)

The technique for controlling power is called Pulse Width Modulation (PWM). Here, we also use it to control the brightness of each of the LEDs. Shown in the diagram below the signal from one of the PWM pins on the UNO.



The PWM output will produce a pulse at roughly every 1/500 of a second. The 'analogWrite' function controls the length of this pulse, thus, specifying a value between 0 and 255 won't produce any pulse at all at 'analogWrite(0)' and a pulse that lasts all the way until the next pulse is due will be at 'analogWrite(255)'.

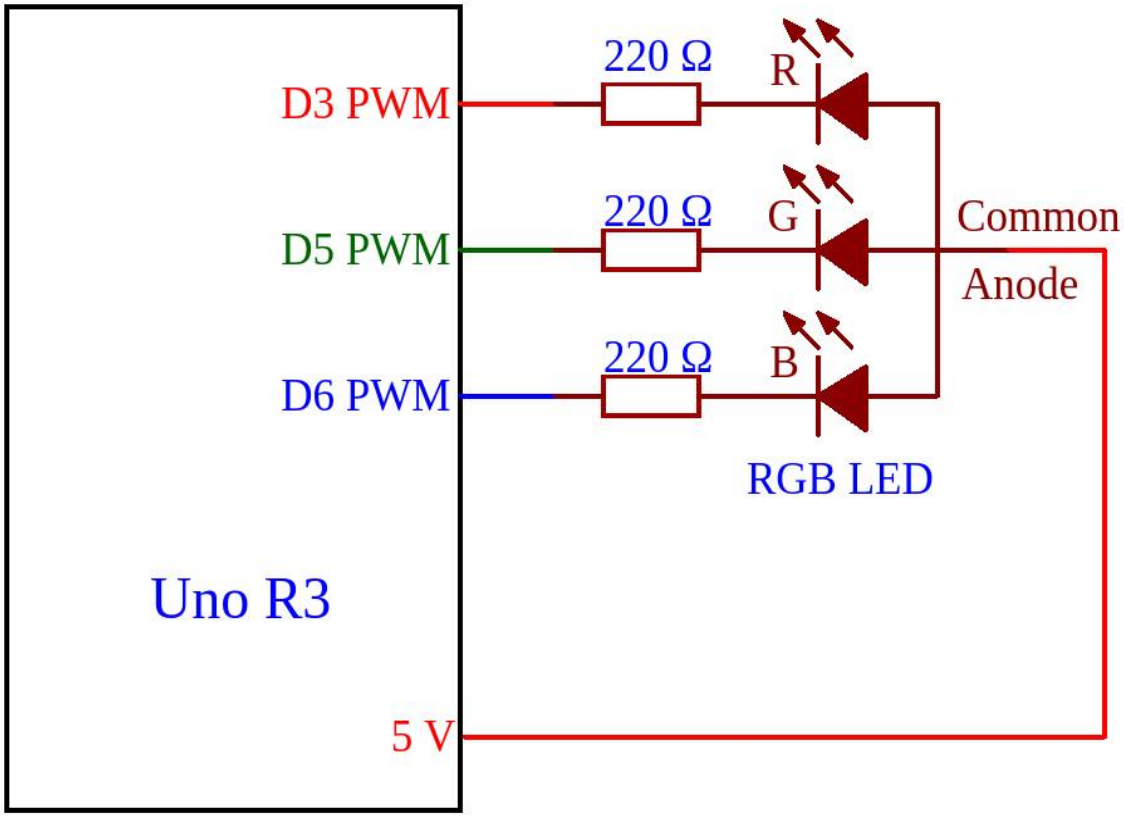
If the output pulse is only high for 5% of the time, then whatever we are driving will only receive 5% of full power. However, in the case that the output is at 5V for 90% of the time, then the load will get 90% of the power delivered to it. We cannot see the LEDs rapidly turning on and off, so to the human eye, it just looks like the brightness is changing.

5.7 Connection

Schematic

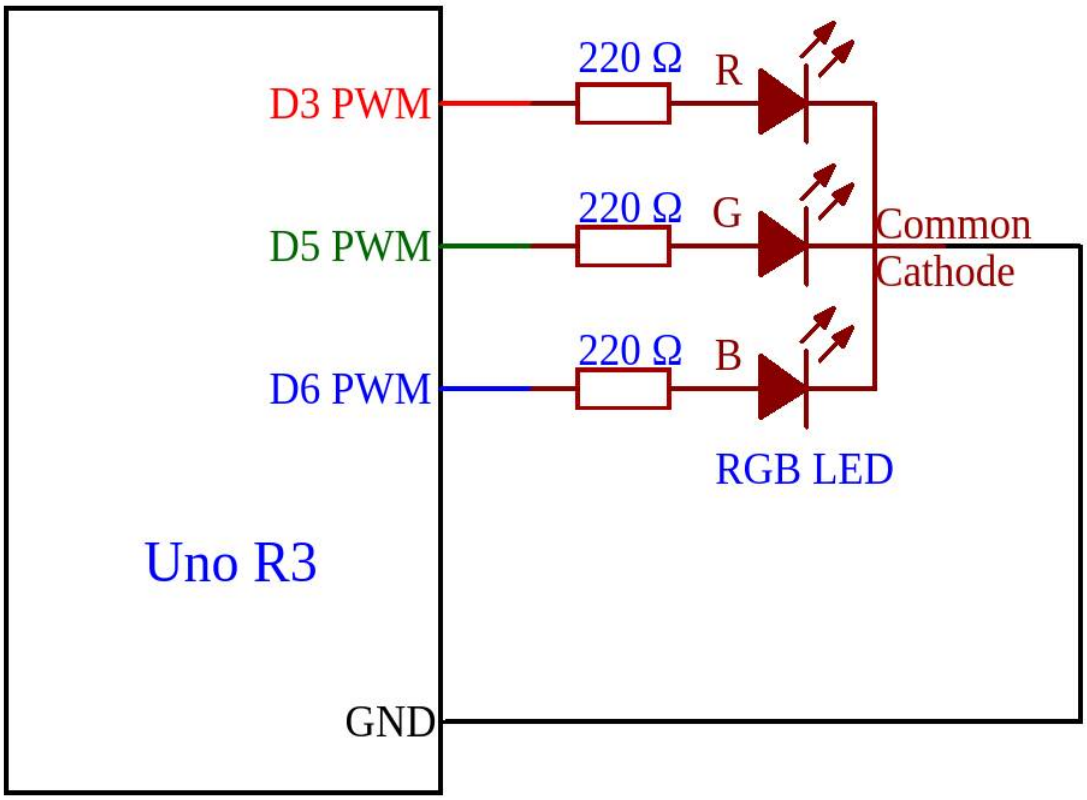
Common Anode

- You can see from the diagram that the longest leg directly connects to the **5 V** and the other pins connects (in series with a 220Ω resistor each) to 3 digital pins (capable of PWM).

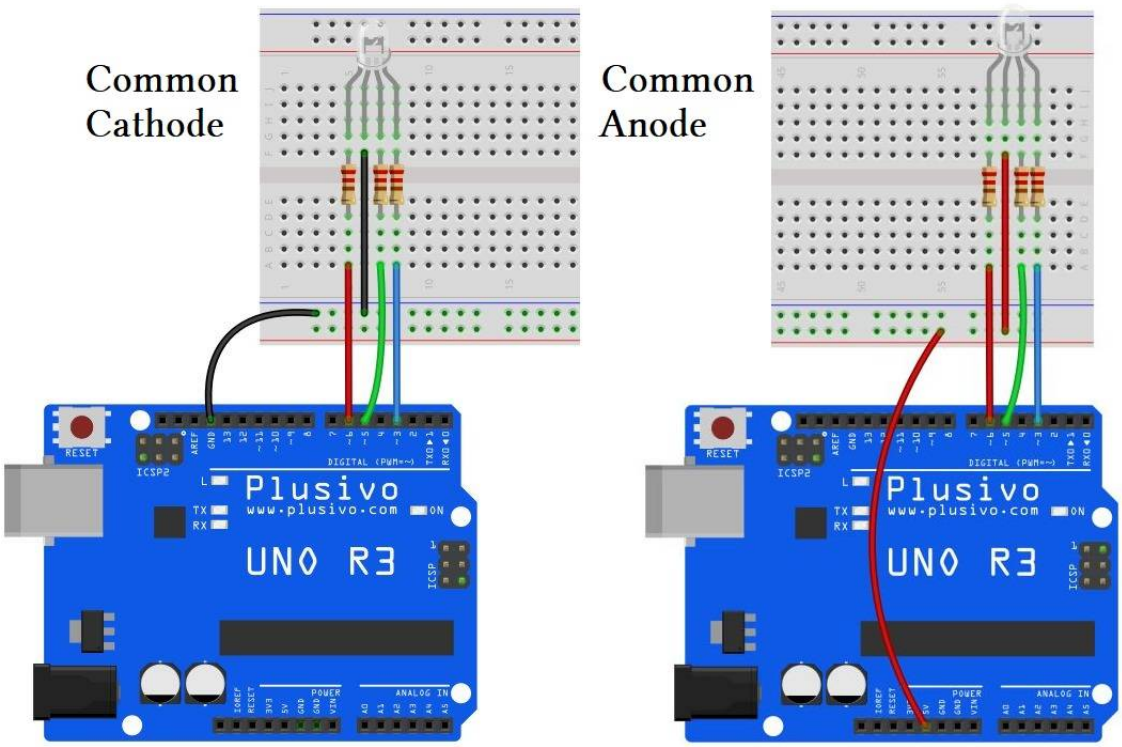


Common Cathode

- You can see from the diagram that the longest leg directly connects to the **GND** and the other pins connects (in series with a 220Ω resistor each) to 3 digital pins (capable of PWM).



Wiring diagram



5.8 Code

After wiring, find and open the program located in the folder - Lesson 4 RGB LED, and UPLOAD to upload the code. If there are any errors, see Lesson 2 for details about program uploading

FOR loops will be used in our code to be able to cycle through the colors. The first will go from RED to GREEN, then GREEN to BLUE and the third is from BLUE to RED.

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

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

Then we will write the 'setup' function. This function only runs once after the Arduino has reset as shown in the previous lesson. Here, the function just need to define the three pins we are using as being outputs.

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

Let’s have a look at the last function in the sketch and define variables.

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

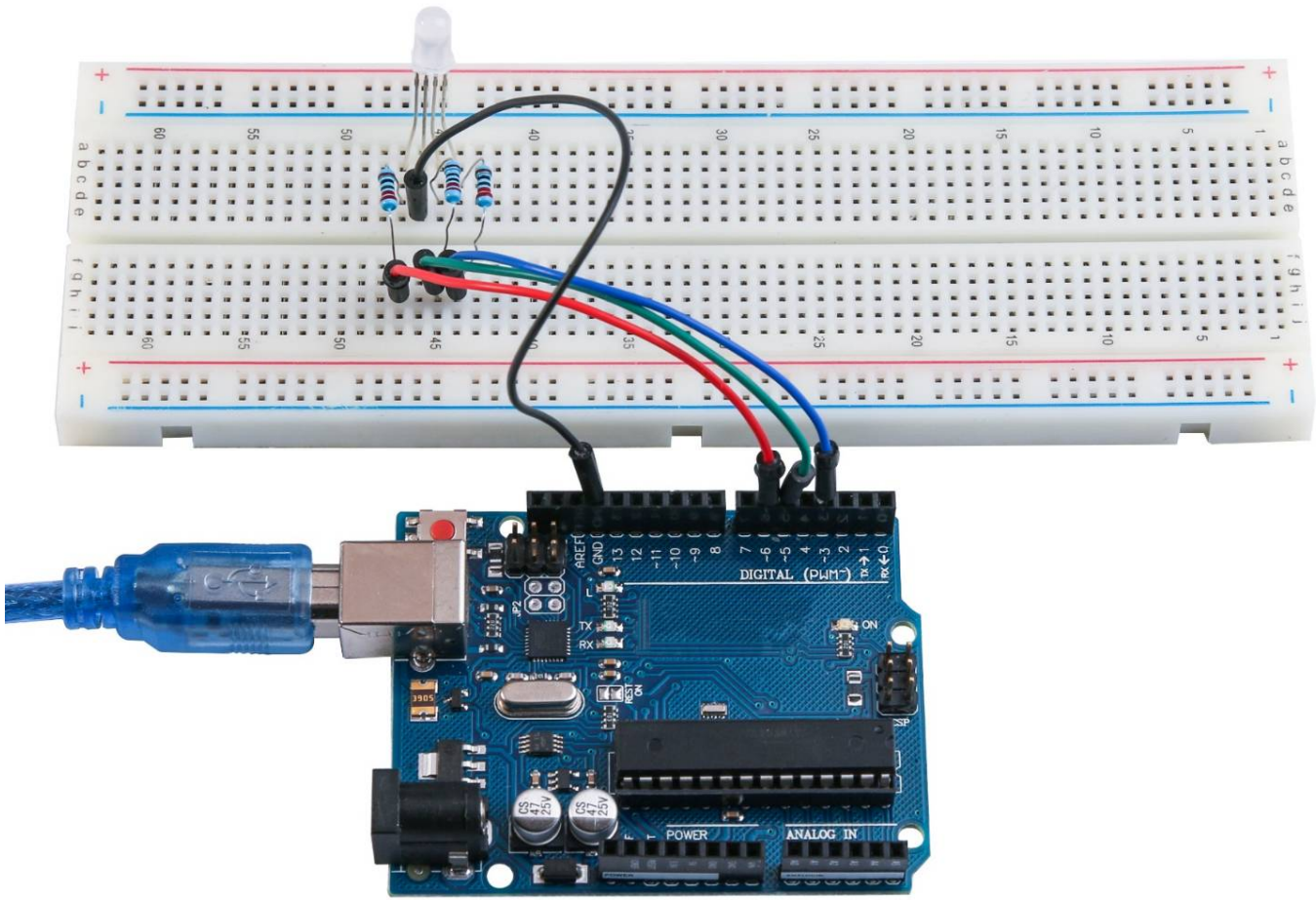
There are three arguments in this function, one argument is for the brightness of each of the red, green and blue LEDs, with values between 0 (means off) and 255 (maximum brightness). The 'analogWrite' argument of the function will then be used to set the brightness of each LED.

Looking at the 'loop' function, it can be seen that we are specifying the amount of red, green and blue light that we want to produce, then pause for a second before moving on to the next color.

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

Now, try to add few colors of your own to the sketch and observe the effect on your LED.

5.9 Example picture



6. Lesson 5 Digital Inputs

6.1 Overview

In this lesson, it will be discussed how to use push buttons containing digital inputs to turn a LED on and off. Pressing the button will turn the LED on and off.

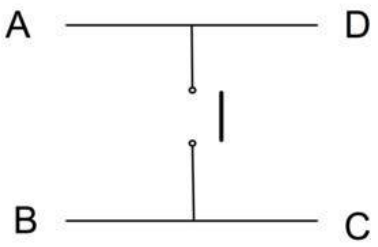
6.2 Components Required

(1) x Plusivo Uno R3
(1) x 830 Tie-points Breadboard
(1) x 5mm red LED
(1) x 220 ohm resistor
(2) x push switches
(7) x M-M wires (Male to Male jumper wires)

6.3 Component Introduction

PUSH SWITCHES

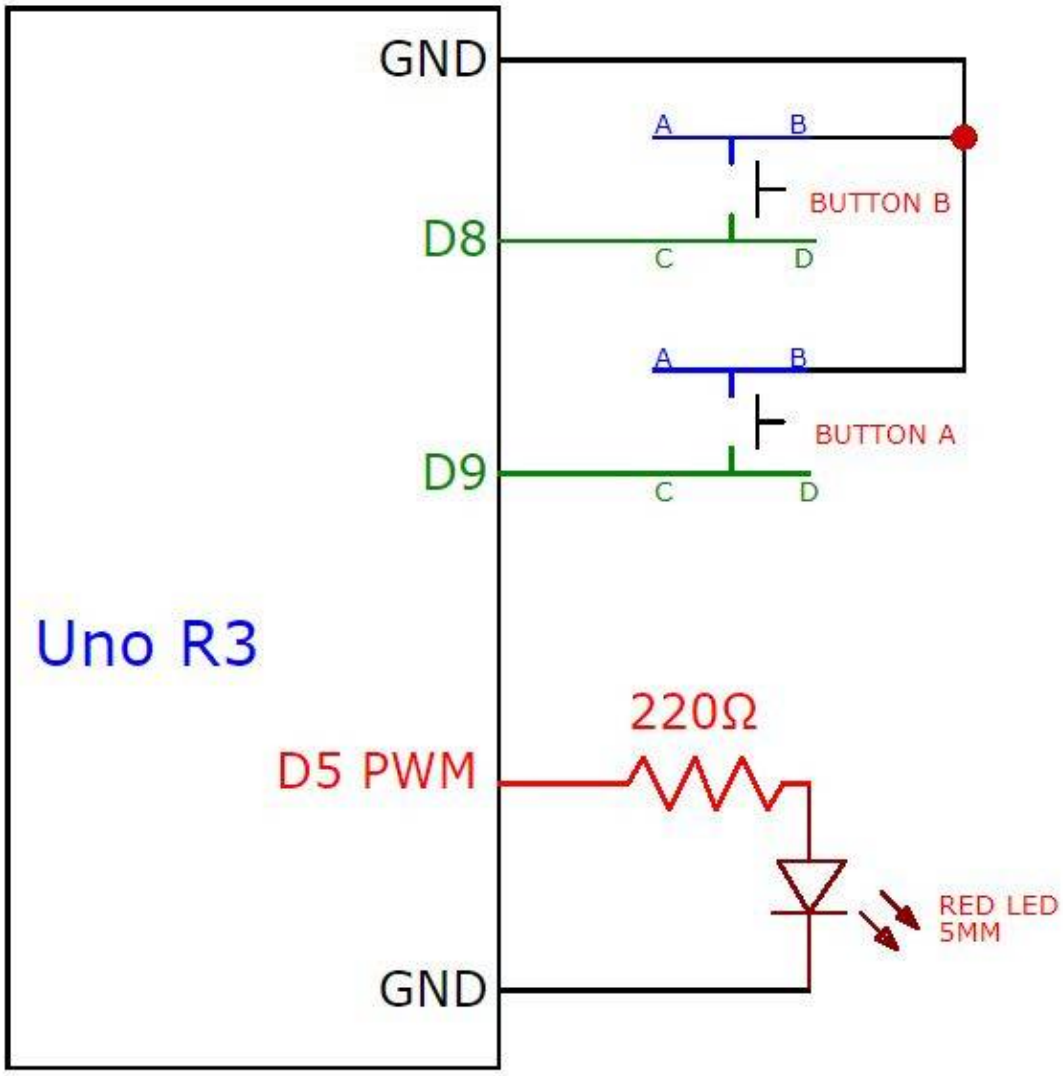
Switches are simple components that works in the following manner: when a button is pressed or a lever is flipped, it causes two contacts to connected together resulting to flow of electricity through them. The switches in this lesson have four connections.



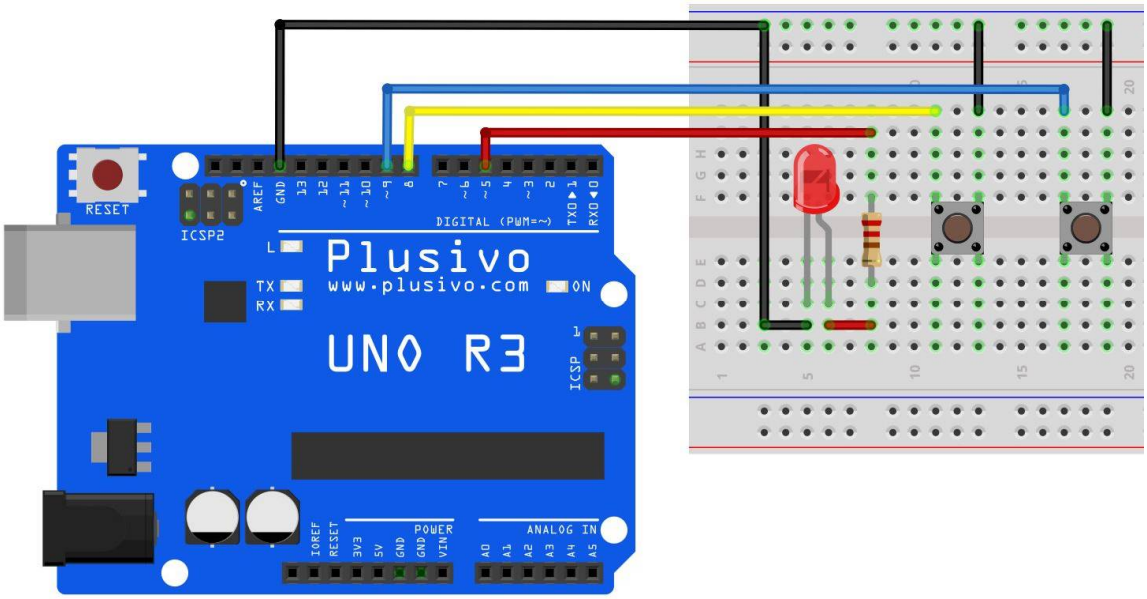
There are only two electrical connections inside the switch package: pins B – C and A - D that are connected together.

6.4 Connection

Schematic



Wiring diagram



The pins of switches protrude from opposite sides of the square bodies of the component. This means that the distance between the pins will only be enough when they are placed correctly on the breadboard. Keep in mind that the LED has to have the shorter negative lead to the left.

6.5 Code

After wiring, find and open the program located in the folder - Lesson 5 Digital Inputs, and UPLOAD the code. If there are any errors, see Lesson 2 for details about program uploading.

Onto the UNO board, load the sketch. The code tells us the LED will turn ON when left button is pressed, and it will turn OFF when the right button is pressed.

In the sketch, three variables are defined for the three pins that will be used. The output is the 'ledPin', the 'buttonApin' is the switch closer to the top of the breadboard and other switch is 'buttonBpin'.

The ledPin is the OUTPUT defined by the 'setup' function, but we have the two inputs to set up. In this case, we use the pinMode as 'INPUT_PULLUP' as below:

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

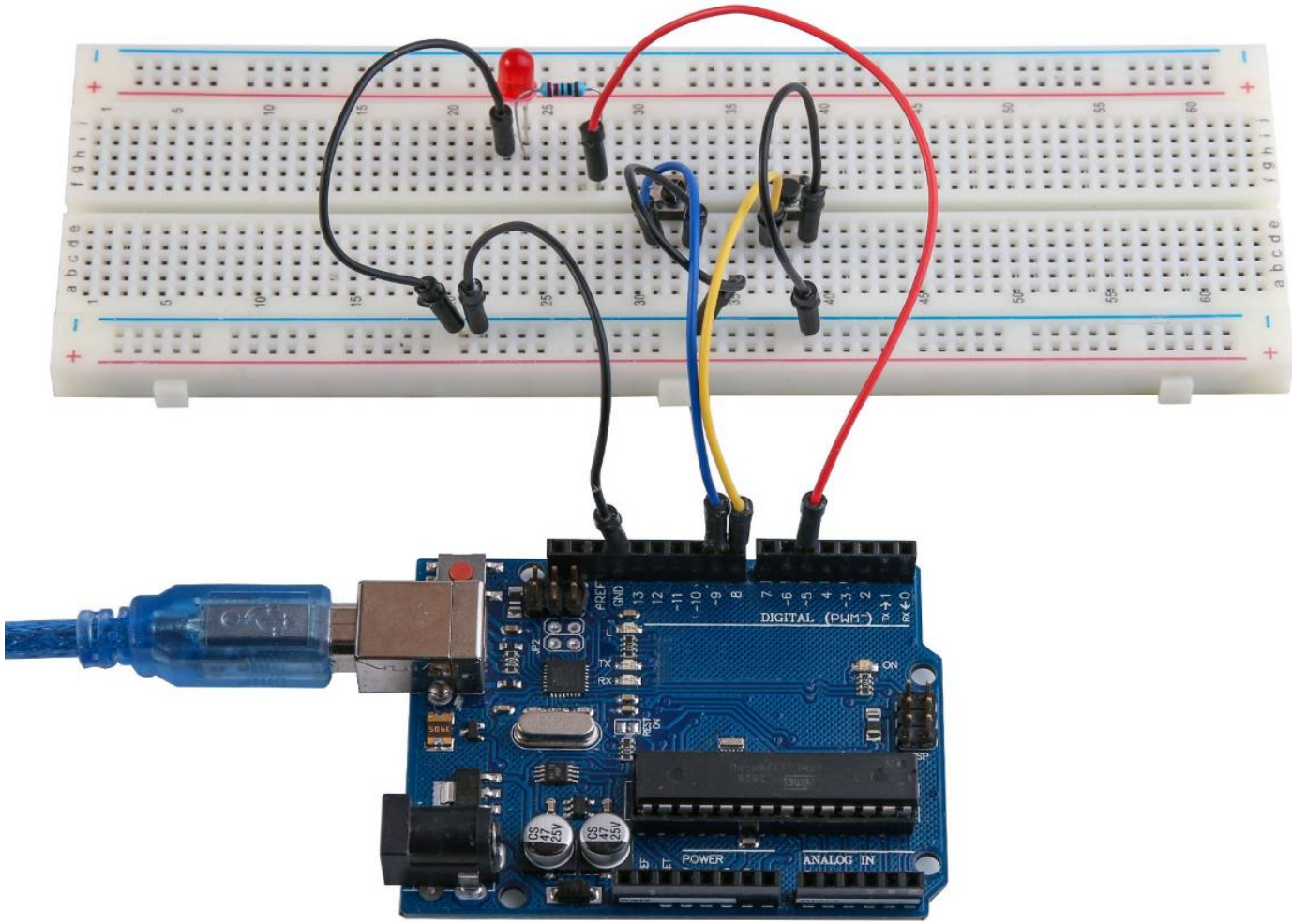
The INPUT_PULLUP pin mode tells us that the pin is used as an input, and it should then be 'pulled up' to HIGH if nothing else is connected. That basically means that the default value is HIGH for the input, unless it is pulled LOW by a press of the button. Thus, switches are usually connected to GND. Pressing a switch will connect the input pin to GND, and it will no longer be HIGH.

Since default value is HIGH unless it is pulled LOW by a press of the button, the logic is a bit backwards. This is resolved in the 'loop' function.

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

We have two 'if' statements in the 'loop' function, one for each button. Each of the statements does a 'digitalRead', verifying the appropriate input. Keep in mind that pressing a button will result in LOW input. Thus, if we have button A that is low, the 'digitalWrite' on the ledPin will turn it on. Likewise, a LOW is written to the ledPin if button B is pressed.

6.6 Example picture



7. Lesson 6 Active Buzzer

7.1 Overview

In this lesson, we will be going over the basics about generating sounds with an active buzzer.

7.2 Components Required

(1) x Plusivo Uno R3
(1) x Active buzzer
(2) x F-M wires (Female to Male DuPont wires)

7.3 Component Introduction

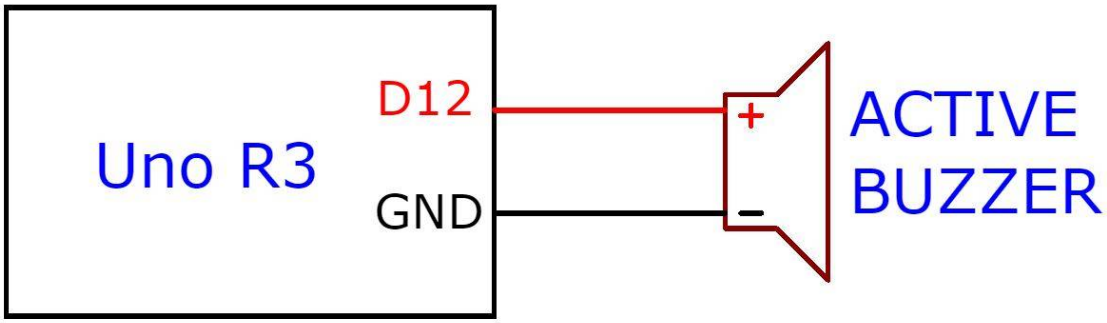
BUZZER

Electronic buzzers have an integrated circuit and are DC-powered. They are commonly used in alarms, automotive electronic devices, computers, printers, photocopiers, electronic toys, telephones, timers and other electronic products for voice devices. Buzzers can be either active or passive. Look at the pins of the buzzers, you can tell that it is the passive buzzer if it has the green circuit and the one enclosed with a black tape is an active buzzer.

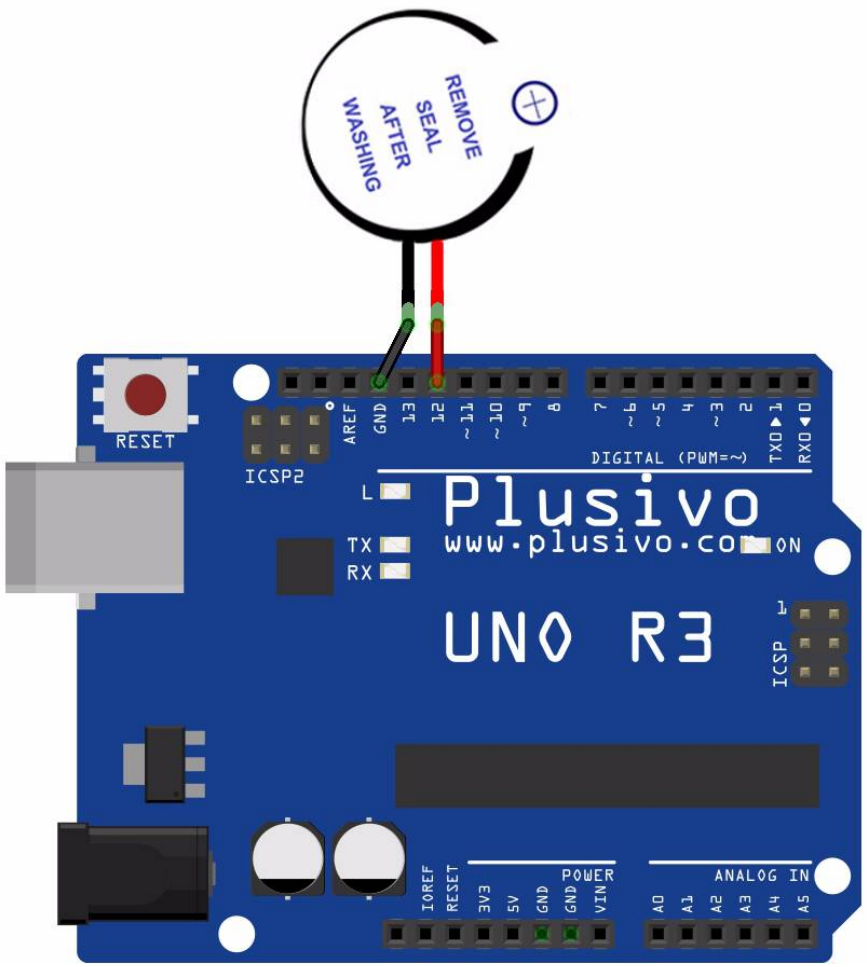
The two types of buzzers differ in such a way that active buzzer generate a sound when electrified while the passive buzzer won't produce any sound. This is because active ones has a built-in oscillating source while the passive buzzer one lacks such a source. Thus for passive buzzer, square waves with a frequency between 2K and 5K need to be used in order to drive it. And because of multiple built-in oscillating circuits, the active buzzer is often more expensive than the passive one.

7.4 Connection

Schematic



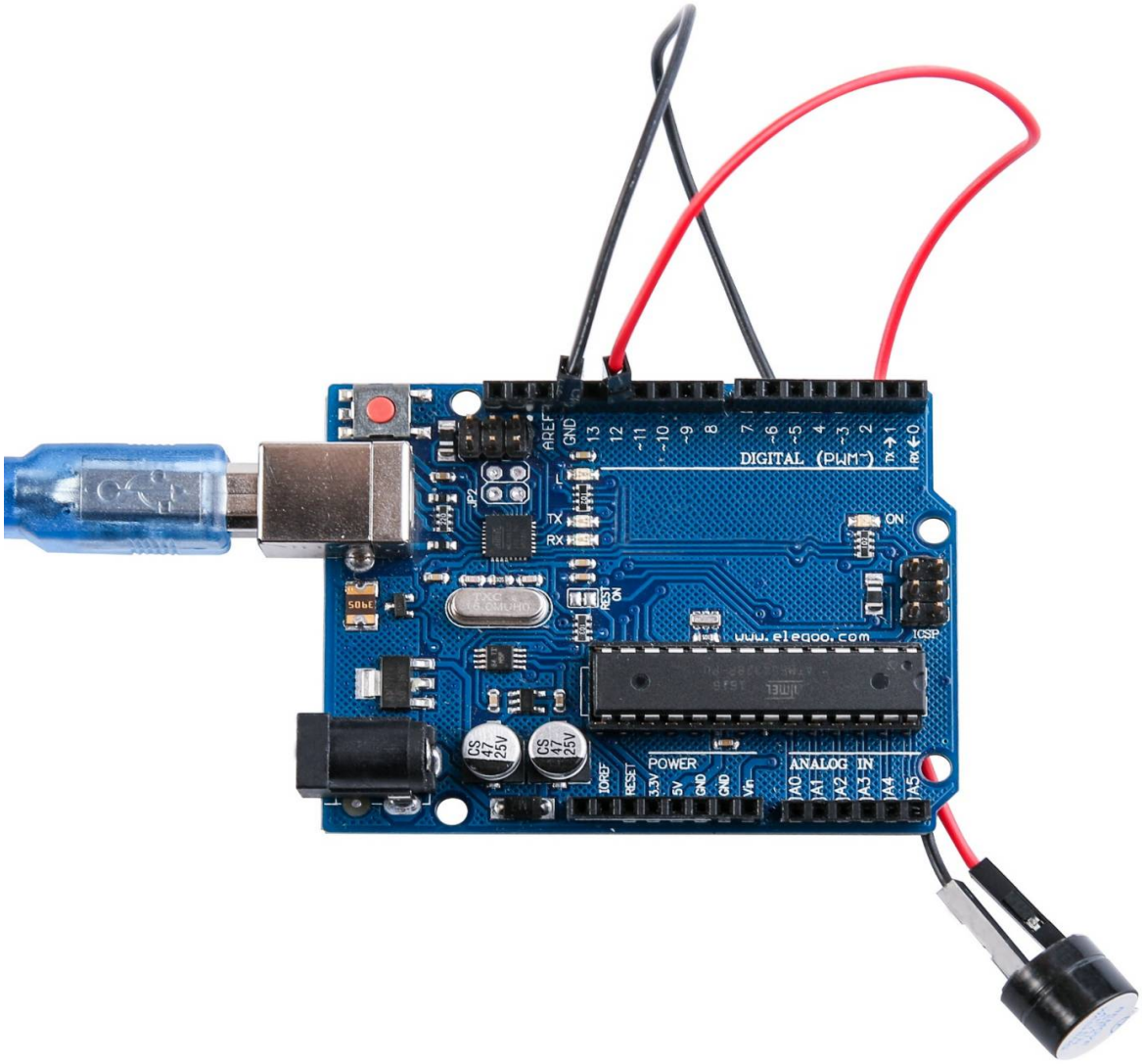
Wiring diagram



7.5 Code

After wiring, find and open the program located in the folder - Lesson 6 Making Sounds, and UPLOAD the code. If there are any errors, see Lesson 2 for details about program uploading.

7.6 Example picture



8. Lesson 7 Passive Buzzer

8.1 Overview

This lesson will teach you the basics about using a passive buzzer. The goal of this chapter is to produce eight different sounds of 0.5 seconds each: from Alto Do (523Hz), Re (587Hz), Mi (659Hz), Fa (698Hz), So (784Hz), La (880Hz), Si (988Hz) to Treble Do (1047Hz).

8.2 Components Required

(1) x Plusivo Uno R3
(1) x Passive buzzer
(2) x F-M wires (Female to Male DuPont wires)

8.3 Component Introduction

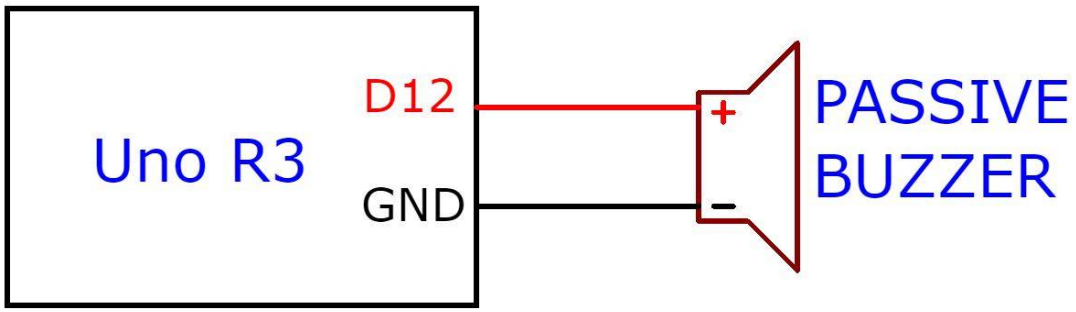
Passive Buzzer

How a passive buzzer works is by using PWM generating audio to make the air vibrate and generate different sounds. For instance, sending a pulse of 523Hz generates Alto Do, a pulse of 587Hz generates midrange Re, and so on. Thus, you can even play a song using the buzzer.

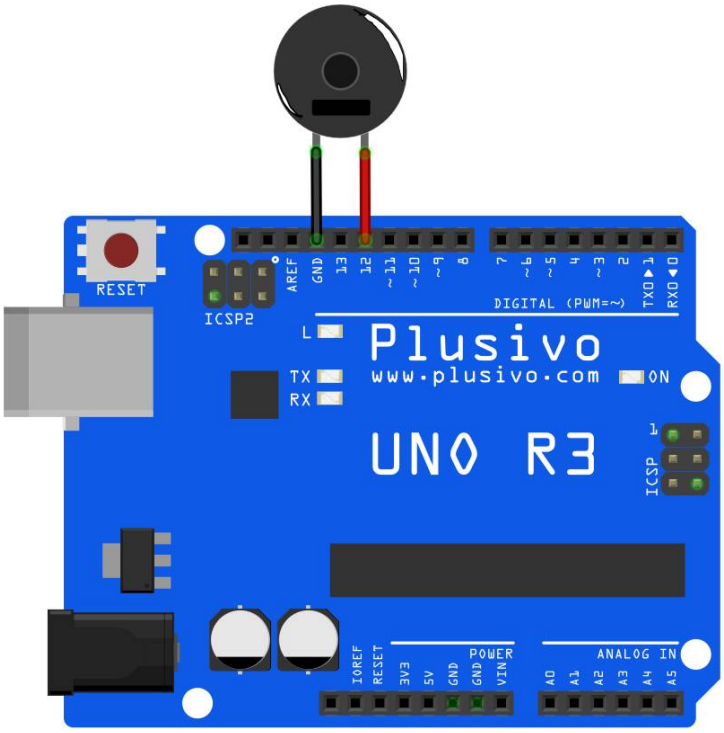
This time we do not use the Arduino board analog Write () function to generate a pulse to the buzzer, as the function will always output 500Hz.

8.4 Connection

Schematic



Wiring diagram

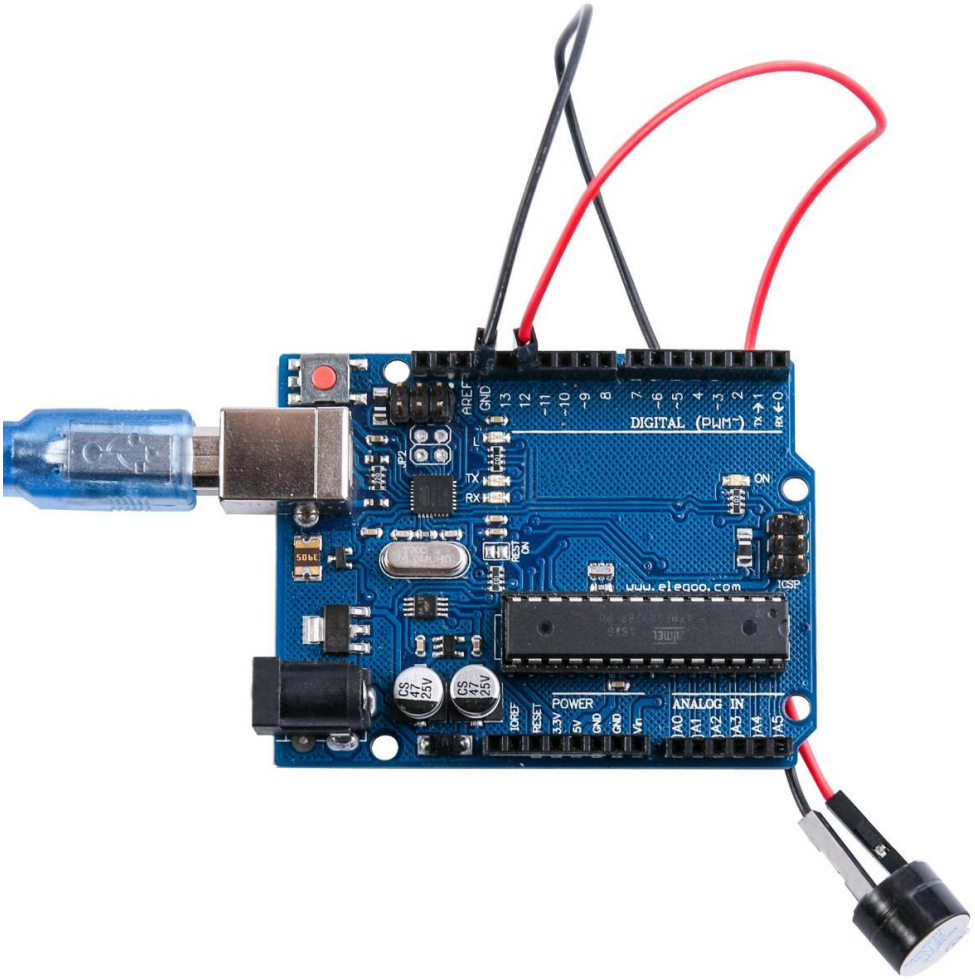


Wiring the buzzer connected to the Arduino board, the red wire (+) going to the pin8 and the black wire (-) going to the GND.

8.5 Code

After wiring, please open the program located in the folder - Lesson 7 Passive Buzzer, and press UPLOAD to upload the code. If there are any errors, see Lesson 2 for details about program uploading. 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. For details about loading the library file, see Lesson 1.

8.6 Example picture



9. Lesson 8 Tilt Ball Switch

9.1 Overview

This lesson will teach you the basics on using a tilt ball switch to identify a small inclination angle.

9.2 Components Required

(1) x Plusivo Uno R3
(1) x Tilt Ball switch
(2) x F-M wires (Female to Male DuPont wires)

9.3 Component Introduction

Tilt sensor

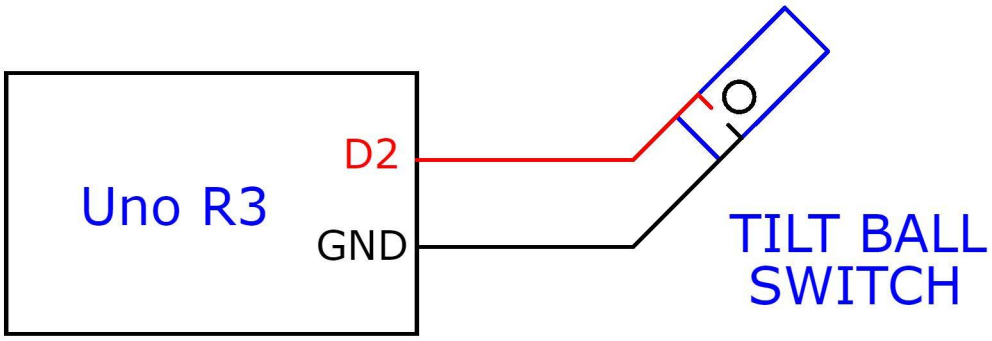
Tilt sensors are used to detect inclination or orientation. They are reliable, low-power, long-lasting and very inexpensive. Because they are so simple, they are quite popular for appliances, gadgets and toys. They are also known as "mercury switches", "tilt switches" or "rolling ball sensors".

They usually consist of a cavity (most often of cylindrical shape) with a conductive free mass inside, for example a blob of mercury or rolling ball. When the end of the cavity, which contains two poles, is pointed downwards, the mass rolls onto the poles and shorts them, acting as a switch throw.

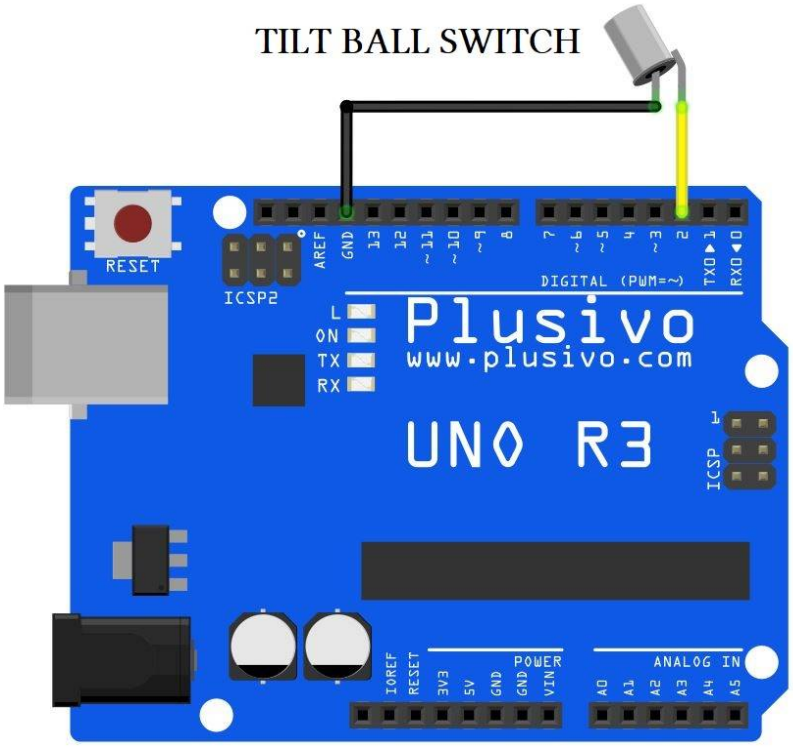
Tilt sensors can detect motion or orientation, but they are not as accurate or flexible as a full accelerometer. The big ones have the capacity of switching power independently, whereas accelerometers require extra circuit components for evaluating output digital or analog voltage.

9.4 Connection

Schematic



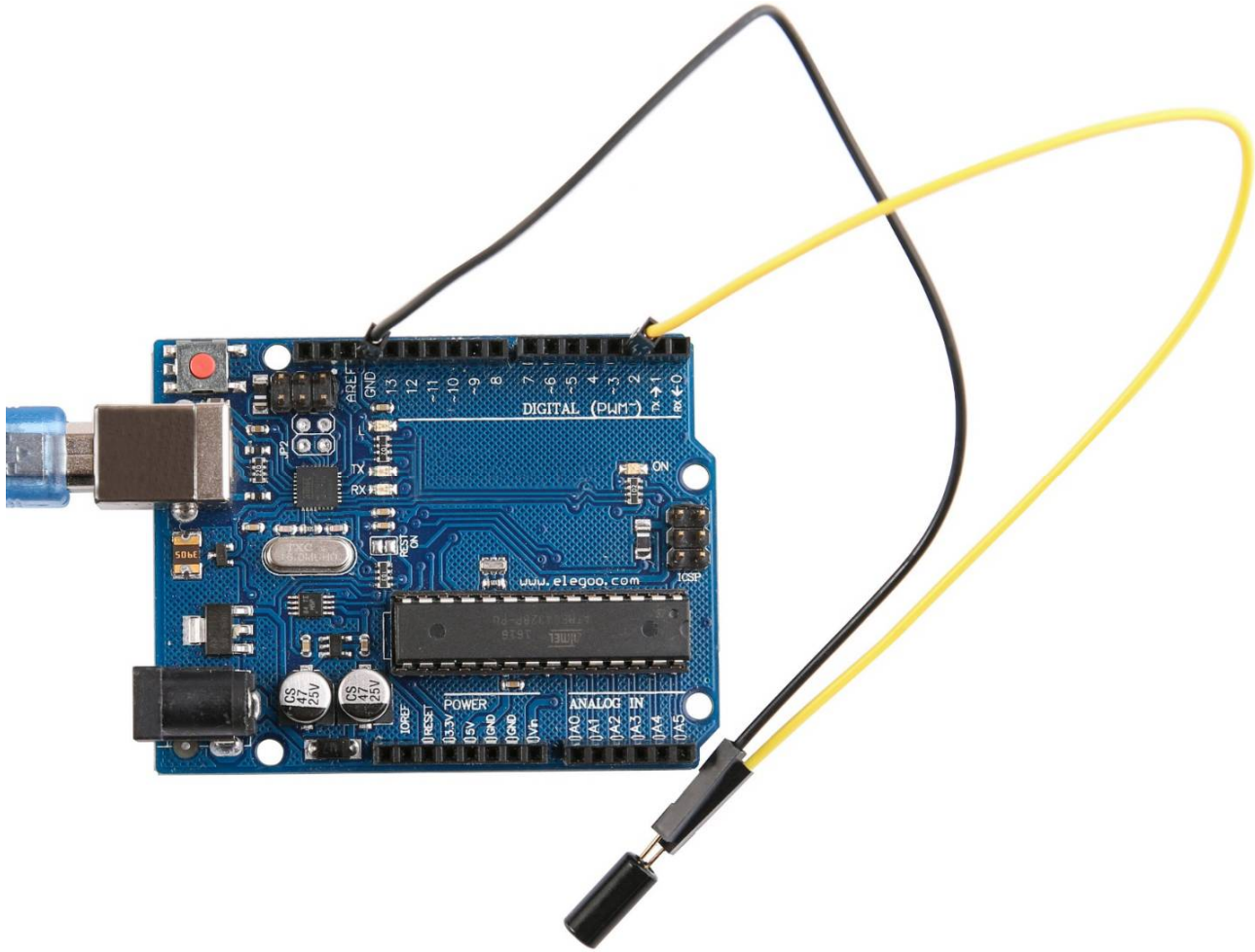
Wiring diagram



9.5 Code

After wiring, please open the program located in the folder - Lesson 8 Ball Switch, and press UPLOAD to upload the code. If there are any errors, see Lesson 2 for details about program uploading.

9.6 Example picture



10. Lesson 9 Servo

10.1 Overview

A servo motor is a geared one, only capable of rotating 180 degrees and is commanded by transmitting electrical pulses from your Arduino, which inform the motor of what position it should go to. The Servo has three wires: a brown one, which should be linked to the GND port, a red one, which should be linked to the 5V port and an orange one, which is the signal wire and should be linked to the Dig #9 port.

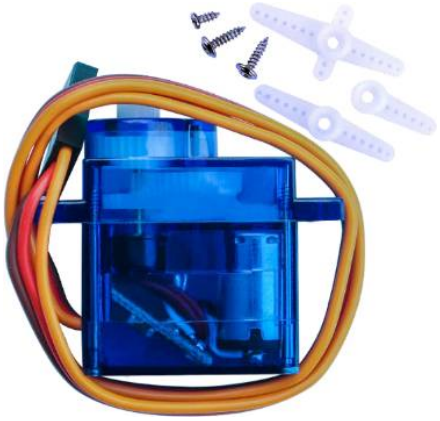
10.2 Components Required

(1) x Plusivo Uno R3
(1) x Servo (SG90)
(3) x M-M wires (Male to Male jumper wires)

10.3 Component Introduction

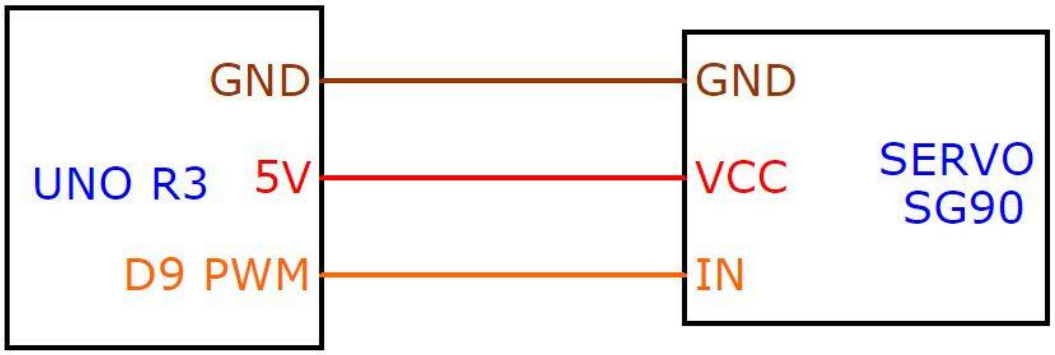
SG90

- 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 to 6V
- Dimensions: 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)

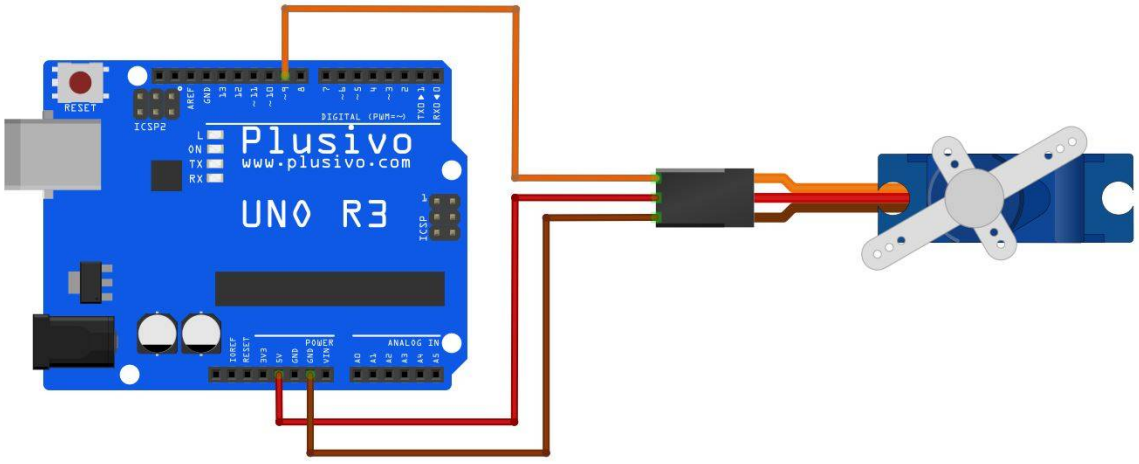


10.4 Connection

Schematic



Wiring diagram

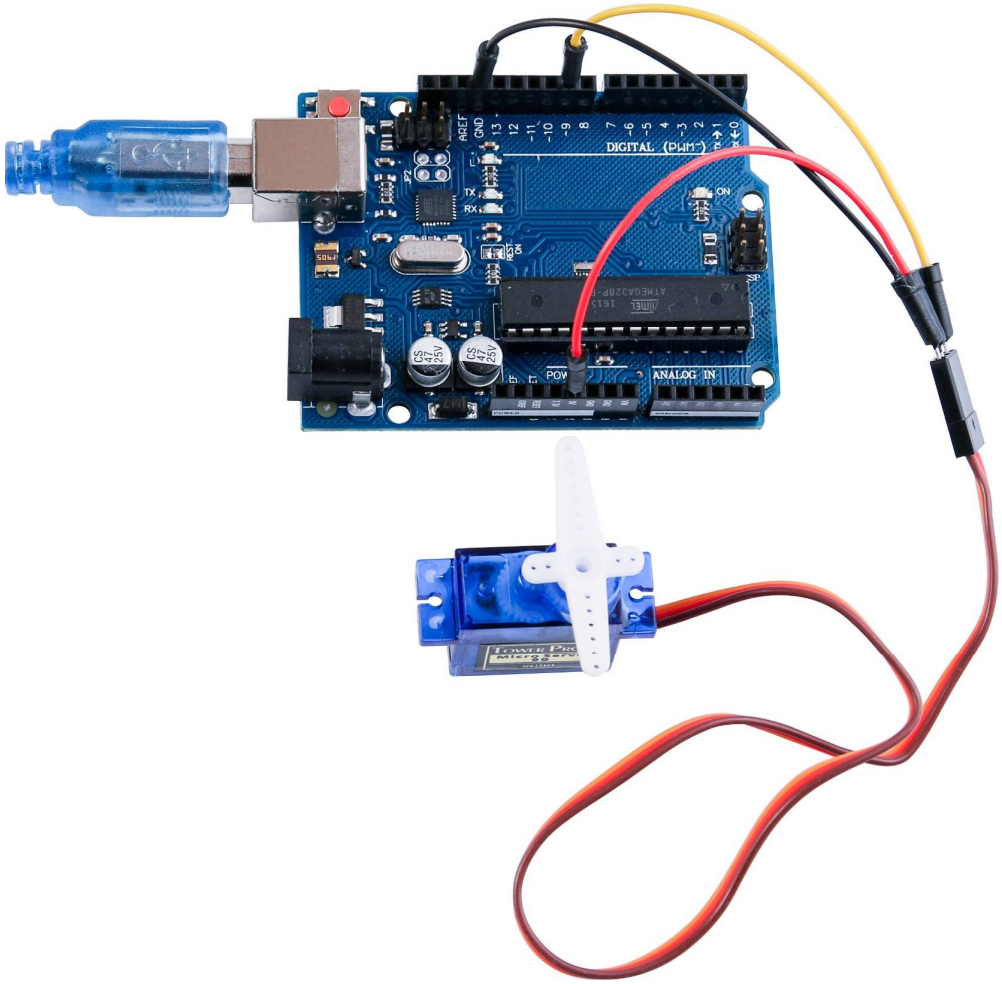


10.5 Code

After wiring, please open the program located in the folder - Lesson 9 Servo, and press UPLOAD to upload the code. If there are any errors, see Lesson 2 for details about program uploading. 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. For details about loading the library file, see Lesson 1.

10.6 Example picture

As the photo below shows, each wire is adapted via corresponding M-M wires: the brown wire of the servo with the black M-M wires, the red wire with the red M-M wires, and the orange wire with the yellow M-M wires.



11. Lesson 10 Ultrasonic Sensor Module

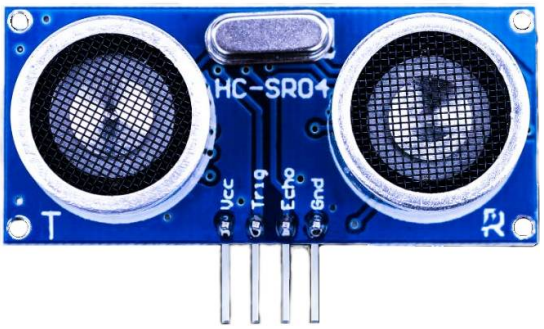
11.1 Overview

The Ultrasonic sensor is a great component for all kinds of projects that need distance measurements such as avoiding obstacles.

The HC-SR04 is inexpensive and very easy to use, as we will be using a Library designed in particular for this sensor.

11.2 Components Required

(1) x Plusivo Uno R3
(1) x Ultrasonic sensor module
(4) x F-M wires (Female to Male DuPont wires)



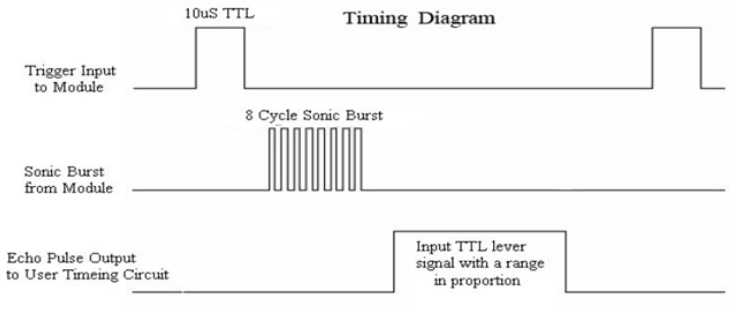
11.3 Component Introduction

Ultrasonic sensor

The ultrasonic sensor module HC-SR04 has a capacity of 2cm to 400cm indirect measurement, with a ranging precision of up to 3mm. The modules includes ultrasonic transmitters, receivers and control circuits. The fundamental working formula is:

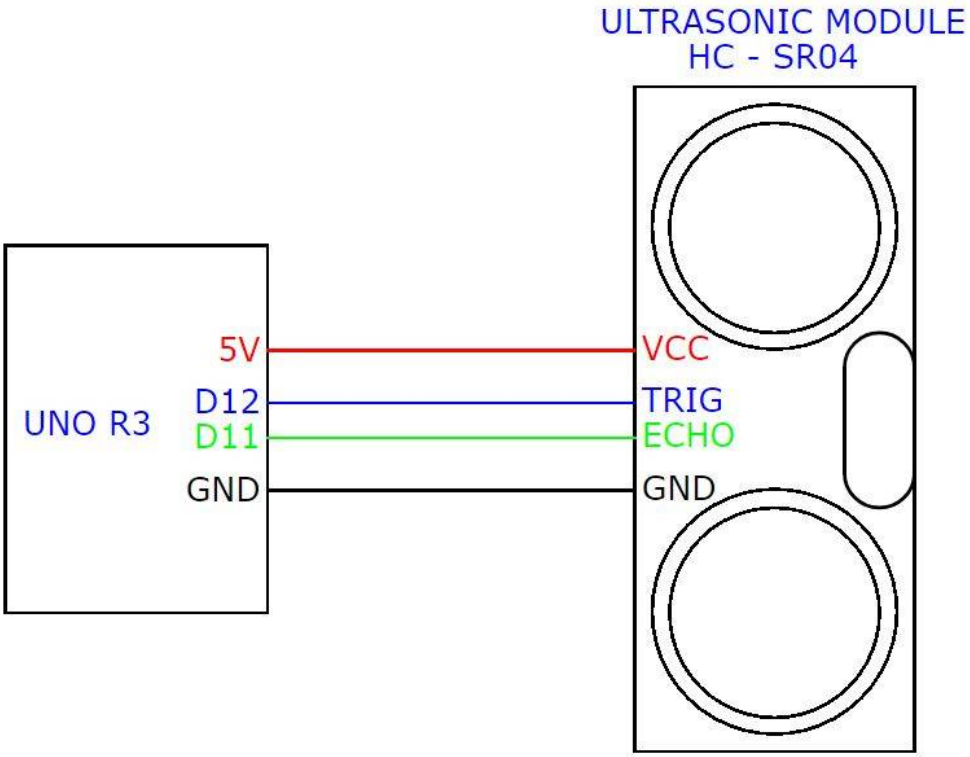
Test distance = (high level time × velocity of sound (340m/s))/2

You can observe the Timing Diagram below. As you can see. Only a short 10ms pulse is necessary to trigger the device to start the ranging. Afterwards, the module will send out 8-cycle bursts of ultrasound at 40 kHz and raise its echo, thus the range through the time interval between sending trigger signal and receiving echo signal can be measured. The formula is: ms/58 in centimeters or ms/148 in inches or the range = high level time * velocity (340M/S) / 2. We recommend using measurement cycles over 60ms.

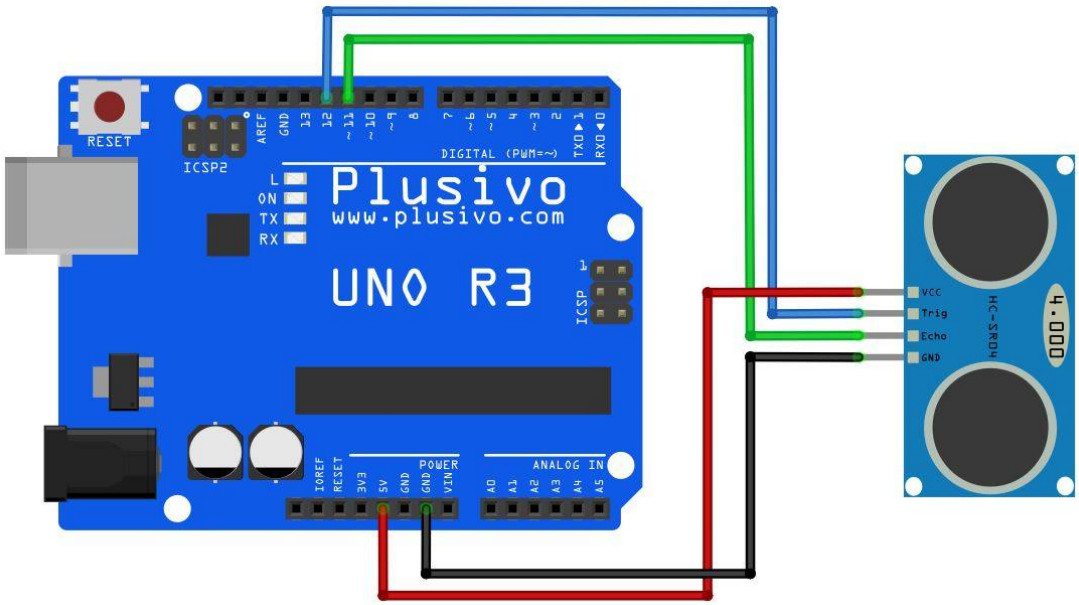


11.4 Connection

Schematic



Wiring diagram

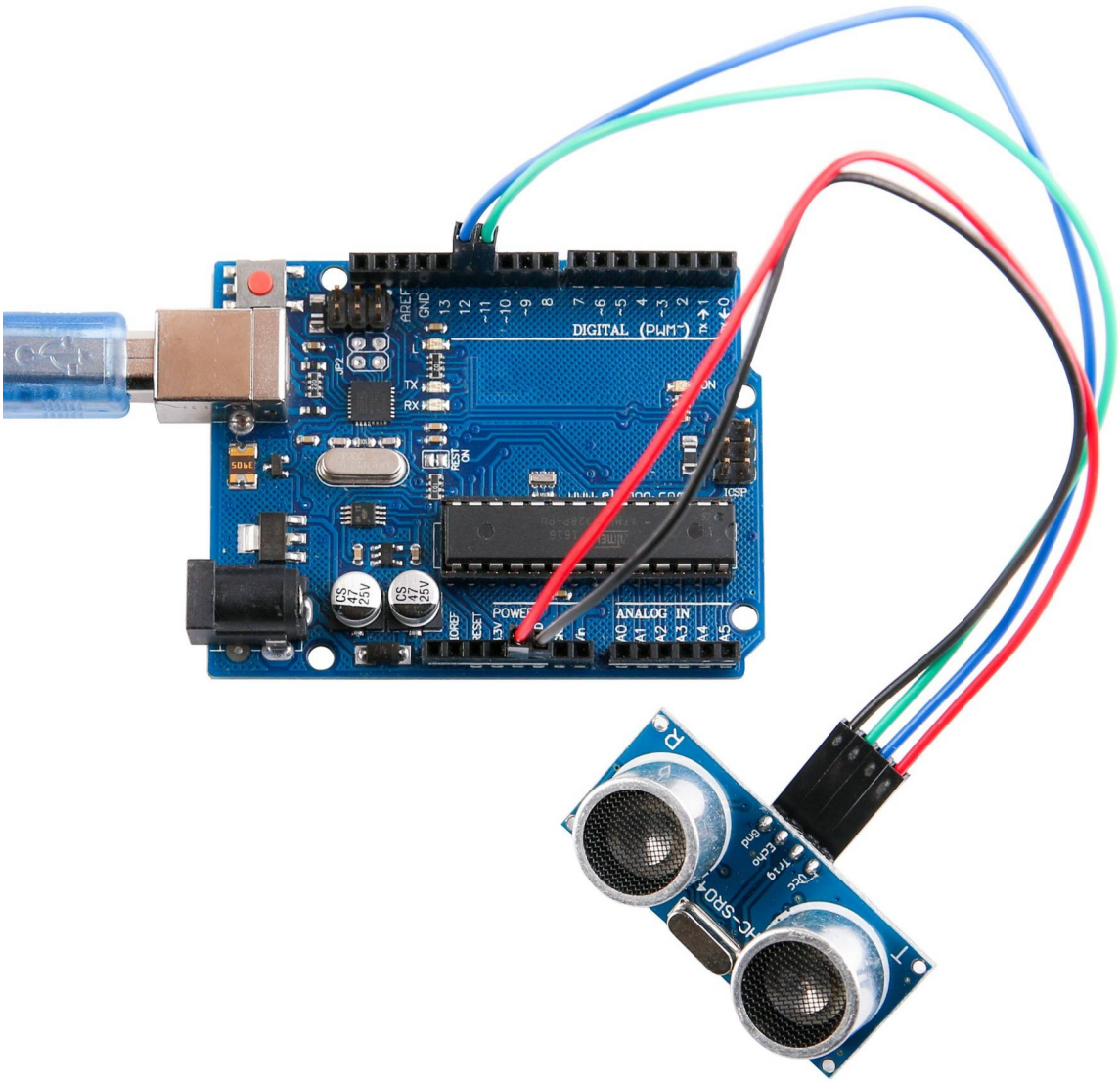


11.5 Code

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

After wiring, please open the program located in the folder - Lesson 10 Ultrasonic Sensor Module, and press UPLOAD to upload the code. If there are any errors, see Lesson 2 for details about program uploading. 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. For details about loading the library file, see Lesson 1.

11.6 Example picture



12. Lesson 11 DHT11 Temperature and Humidity Sensor

12.1 Overview

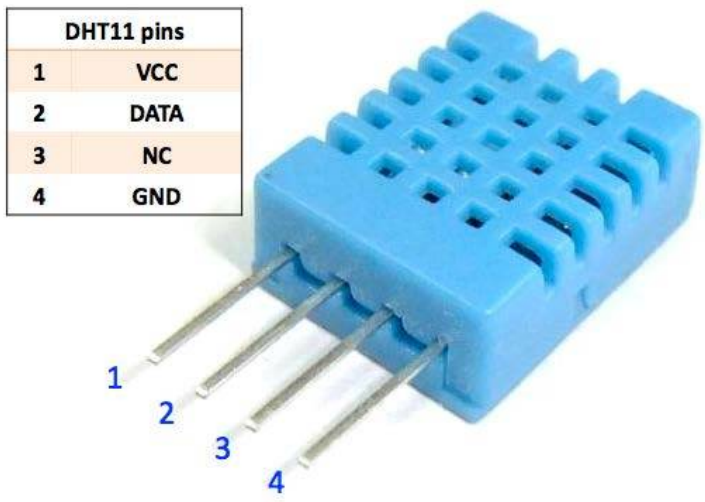
In this lesson, we will learn the basics on using a DHT11 Temperature and Humidity Sensor. It's sufficiently precise for our basic personal projects which require the reading of humidity or temperature. To make our program easier to write, we will use a specific library created for this sensor.

12.2 Components Required

(1) x Plusivo Uno R3
(1) x DHT11 Temperature and Humidity module
(3) x F-M wires (Female to Male DuPont wires)

12.3 Component Introduction

Temperature and humidity sensor



The DHT11 digital temperature and humidity sensor contains a calibrated digital signal output of the temperature and humidity. The dedicated humidity sensing technology and digital modules collection technology ensure that the product is reliable and is very long-lasting. The sensor includes a sensor of wet components and a temperature measurement device.

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

Product parameters

Relative humidity:

- Resolution: 16Bit
- Repeatability: ±1% RH
- Accuracy: At 25℃ ±5% RH
- Interchangeability: fully interchangeable
- Response time: 1 / e (63%) of 25℃ 6s 1m / s air 6s
- Hysteresis: <± 0.3% RH
- Long-term stability: <± 0.5% RH / yr in

Temperature:

- Resolution: 16Bit
- Repeatability: ±0.2℃
- Range: At 25℃ ±2℃
- Response time: 1 / e (63%) 10S

Electrical Characteristics:

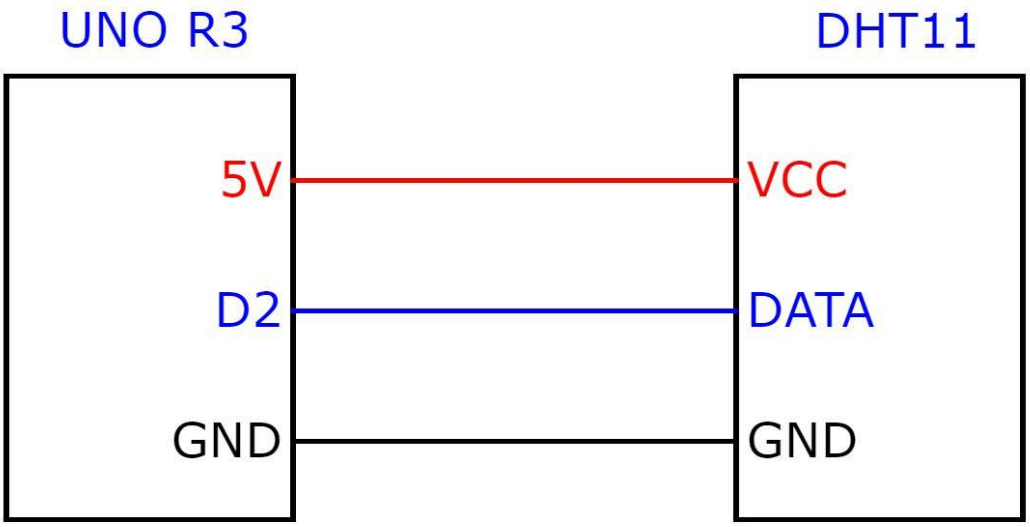
- Power supply: DC 3.5to5.5V
- Supply Current: measurement 0.3mA, standby 60μA
- Sampling period: more than 2 seconds

Pin Description:

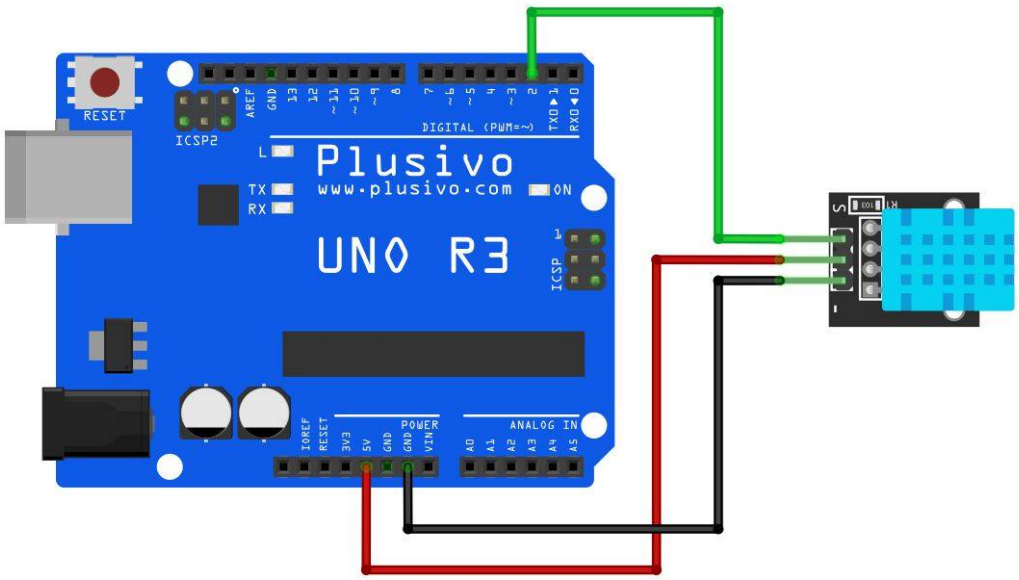
1. VDD power supply 3.5 to 5.5V DC
2. DATA serial data, a single bus
3. NC, empty pin
4. GND ground, the negative power

12.4 Connection

Schematic



Wiring diagram

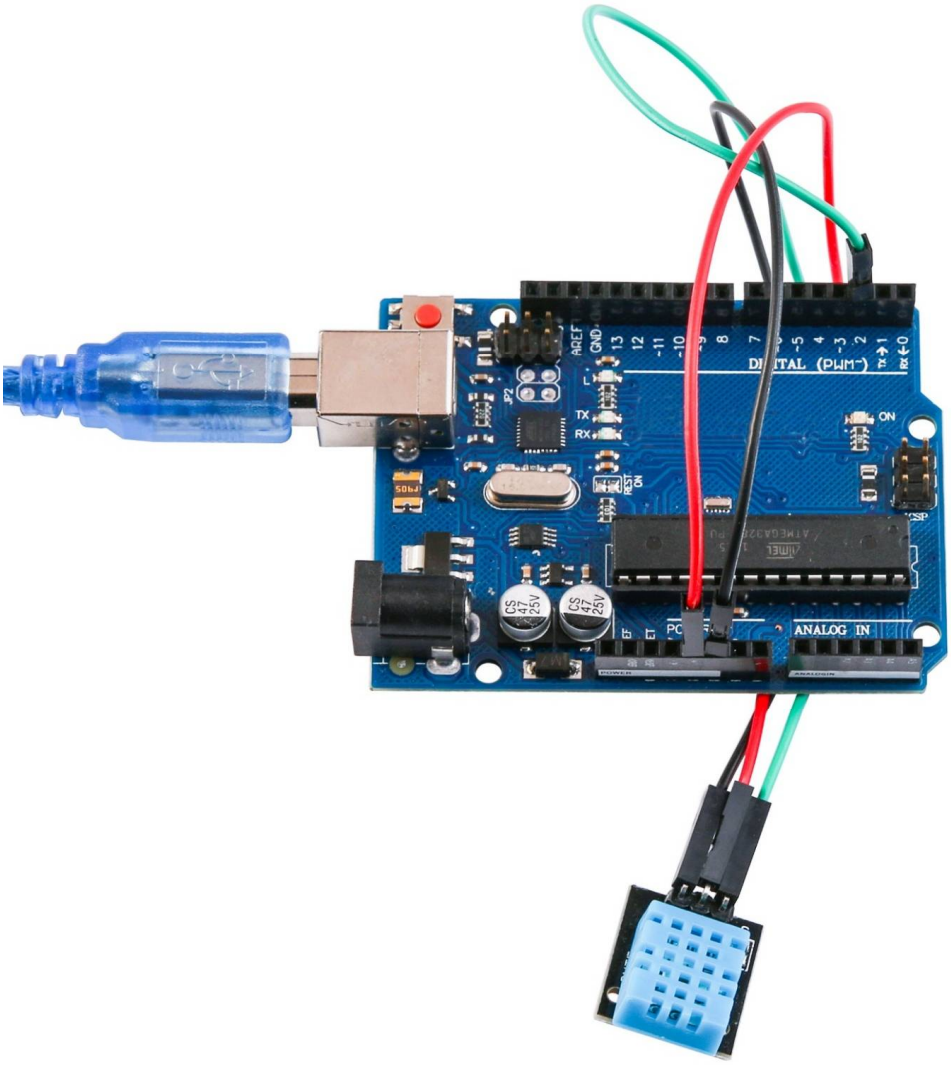


As shown in the pictures, we only need 3 connections to the sensor: Voltage, Ground and Signal, which can be connected to any pin on our device.

12.5 Code

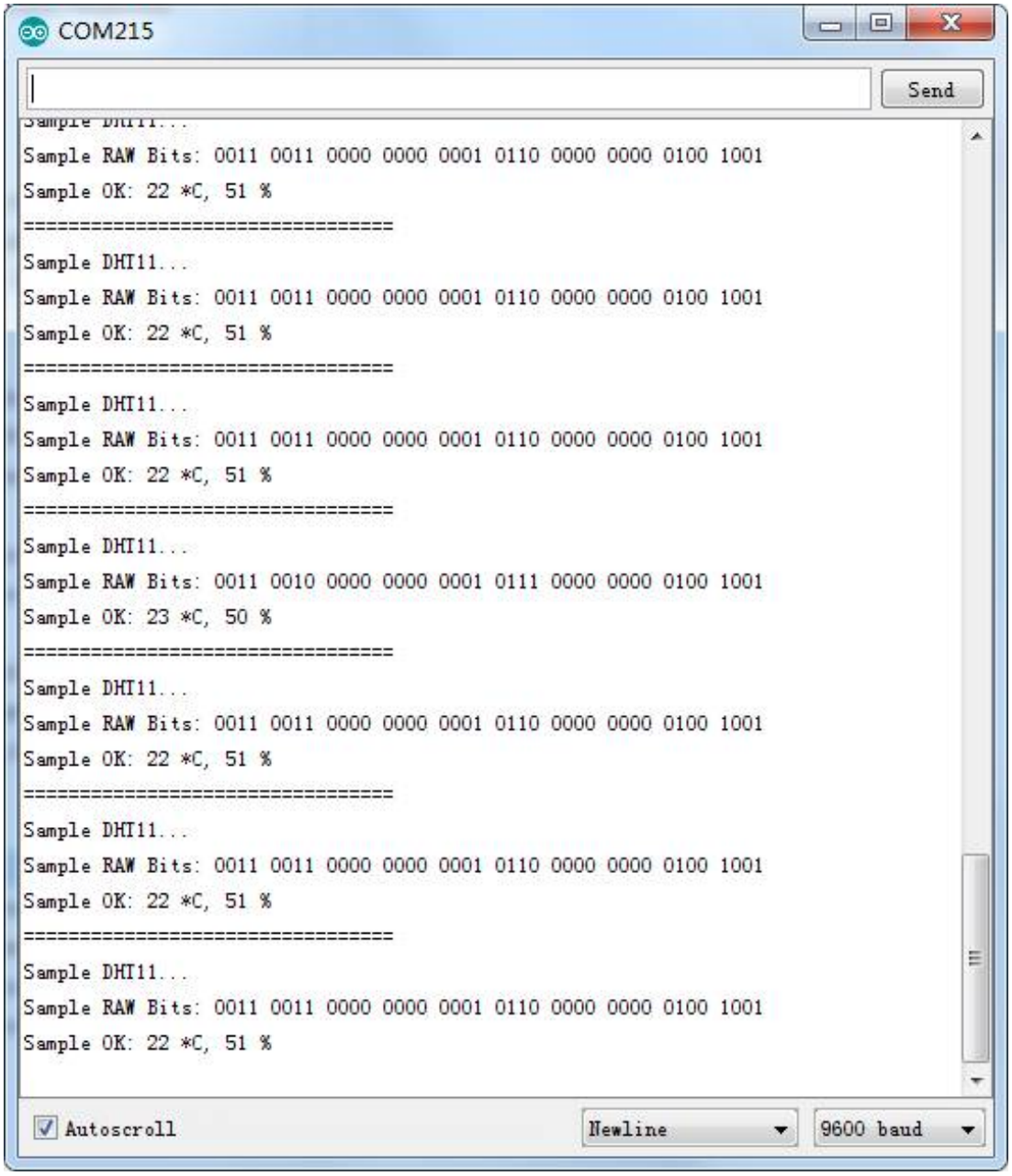
After wiring, please open the program located in the folder - Lesson 11 DHT11 Temperature and Humidity Sensor, and press UPLOAD to upload the code. If there are any errors, see Lesson 2 for details about program uploading. Before you can run this, make sure that you have installed the <SimpleDHT> library or re-install it, if necessary. Otherwise, your code won't work. For details about the tutorial on the loading of library file, see Lesson 1.

12.6 Example picture



Load the program, then open the monitor so we can see the figure below: (showing the temperature of the environment, it now being 22 degrees)

Press the Serial Monitor button to turn it on. The serial monitor is comprehensively introduced in Lesson 1.



13. Lesson 12 Analog Joystick Module

13.1 Overview

In this tutorial we will learn the basics on using the analog joystick module, which is very useful for controling components in your projects.

13.2 Components Required

(1) x Plusivo Uno R3
(1) x Joystick module
(5) x F-M wires (Female to Male DuPont wires)

13.3 Component Introduction

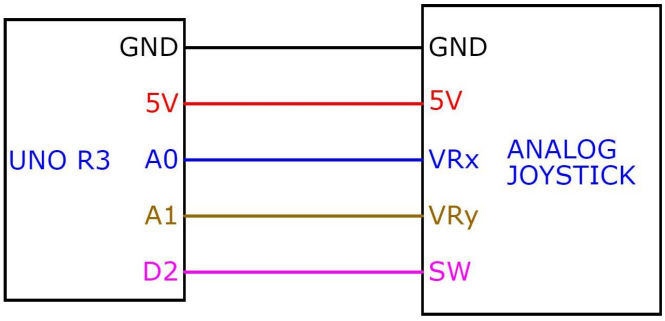
Joystick

The module has 5 pins: VCC, Ground, X, Y, Key, but your labels may be distinctive. The thumb stick is analog and provides precise readings compared to a simple 'directional' joysticks tact which use either buttons or mechanical switches. Additionally, you can press the joystick down to activate a 'press to select' push-button. '

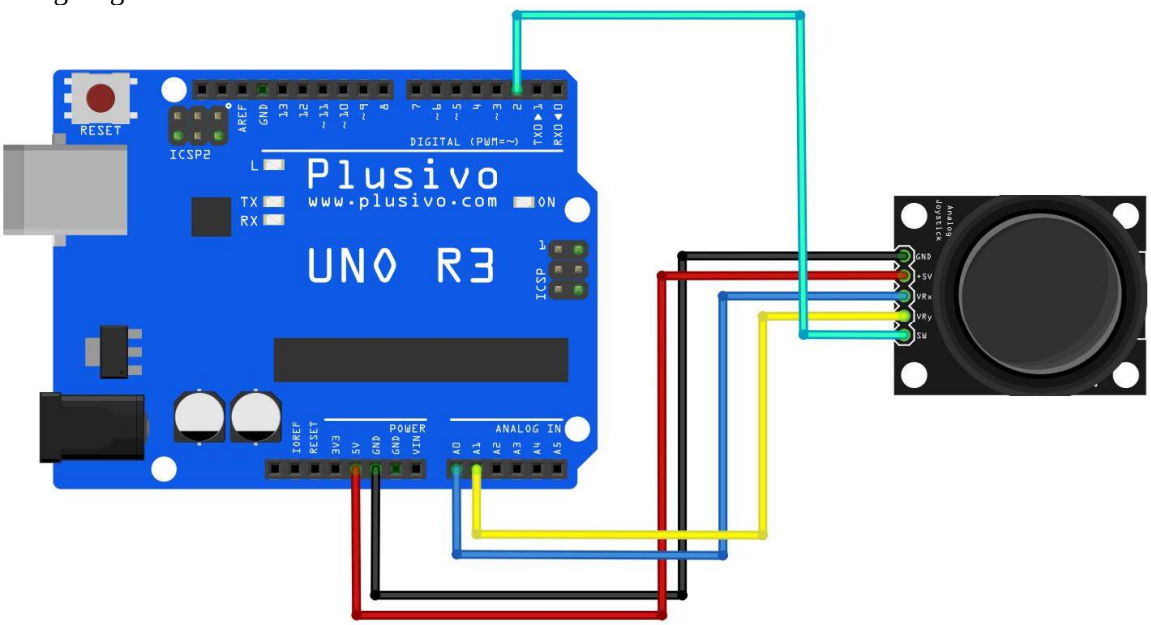
We are using analog UNO pins to receive data from the X/Y pins, and a digital pin to read the button. When the joystick is pressed down, the Key pin is linked to ground, and is floating otherwise. To get accurate readings from the Key/Select pin, it should be connected to VCC with a pull-up resistor, which we can do using the built in resistors on the UNO digital pins, as following:

13.4 Connection

Schematic



Wiring diagram



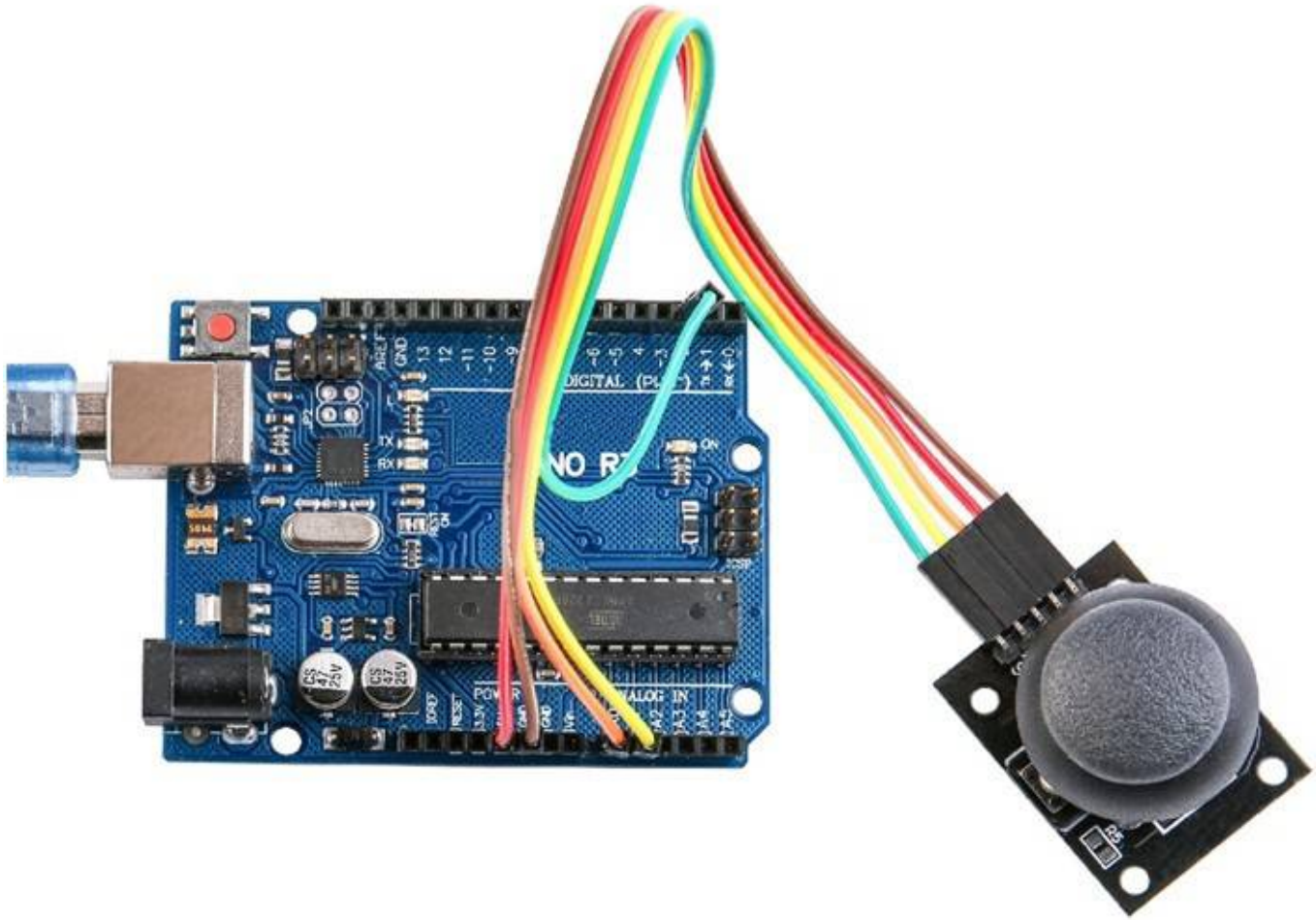
We need 5 connections: Key, Y, X, Voltage and Ground. “Y and X” are Analog and “Key” is Digital.

13.5 Code

After wiring, please open the program located in the folder - Lesson 13 Analog Joystick Module, and press UPLOAD to upload the code. If there are any errors, see Lesson 2 for details about program uploading.

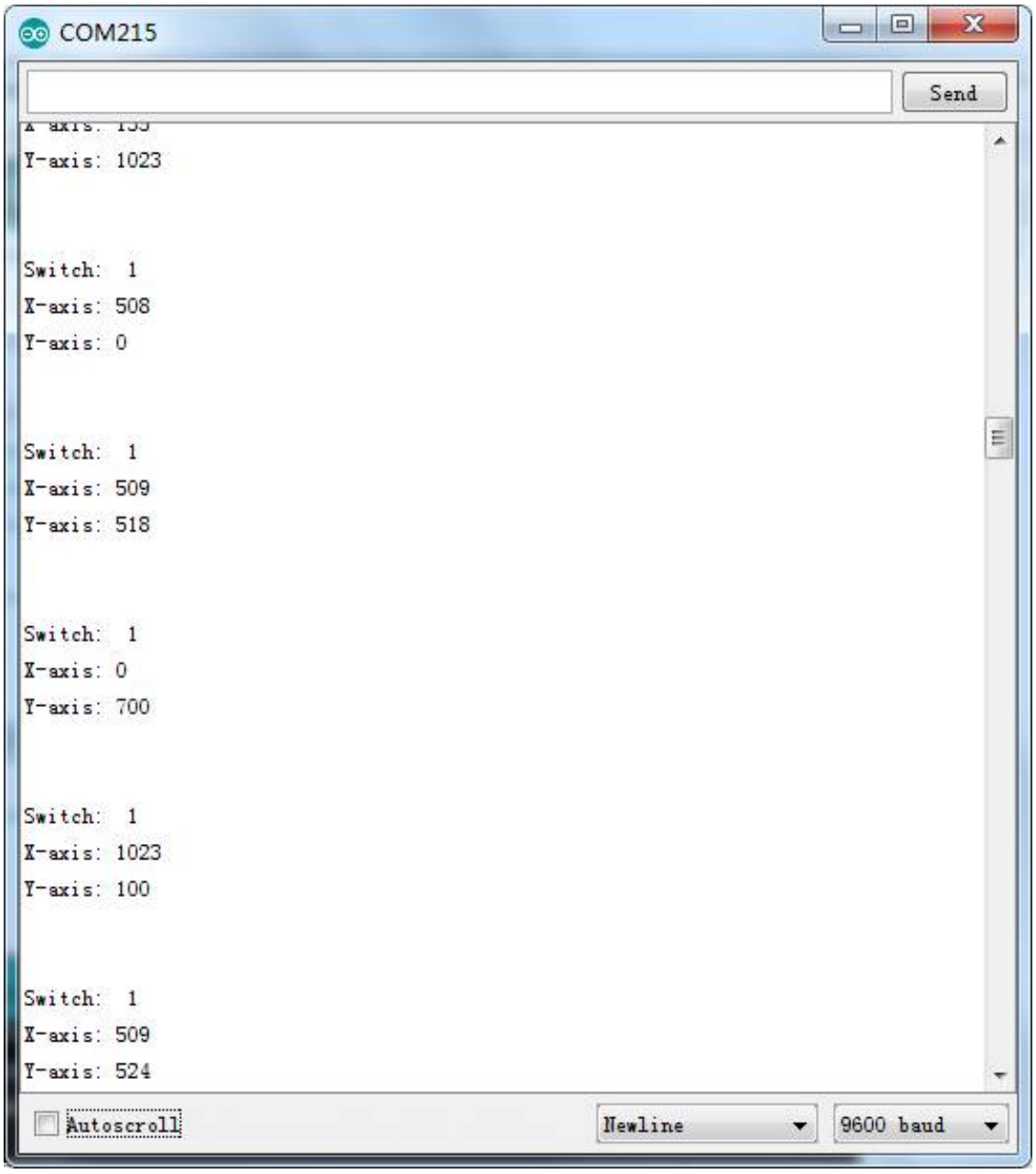
Analog joysticks are essentially potentiometers, so they return analog values. The range of values is from 0 to 1024. From this, we understand that when the joystick is in the resting position or in the middle, it will have a value of approximately 512.

13.6 Example picture



Open the monitor, so you can see the following figure:

Press the Serial Monitor button to turn it on. The serial monitor is comprehensively introduced in Lesson 1.



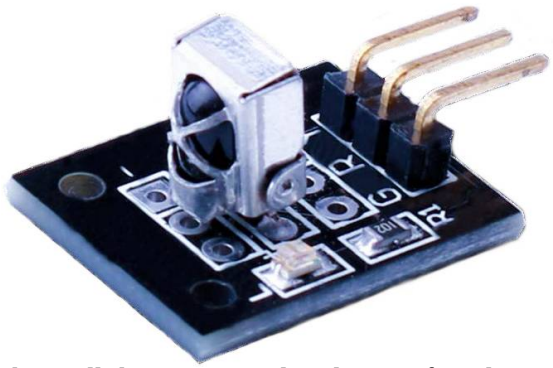
14. Lesson 13 IR Receiver Module

14.1 Overview

Wireless control on your projects can be easily achieved using an Infrared (IR) Remote, as they are simple to understand and use. In this tutorial we will be learning to program the IR receiver, using a Library designed specifically for it.

In our code, we will have access to all the IR Hexadecimal codes available on this remote, we will check if the code was identified and whether we are pressing a key.

14.2 Components Required



(1) x Plusivo Uno R3
(1) x IR receiver module
(1) x IR remote
(3) x F-M wires (Female to Male DuPont wires)

14.3 Component Introduction

IR RECEIVER SENSOR

IR detectors are essentially small microchips with a photocell that are created to detect infrared light, commonly used for remote control detection, especially in TVs and DVD players. TV commands are given by the infrared light inside the remote control: to power on, change the volume, etc. This light can not be seen by the human eye, which makes it harder for us to test a setup.

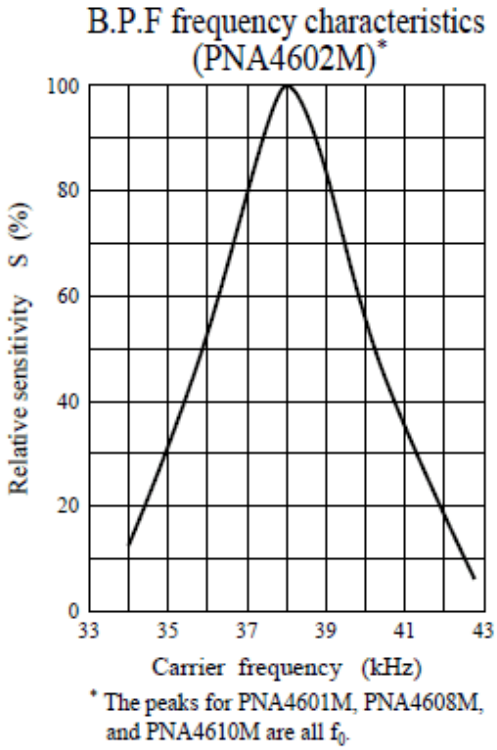
We can spot some differences between these and photocells:

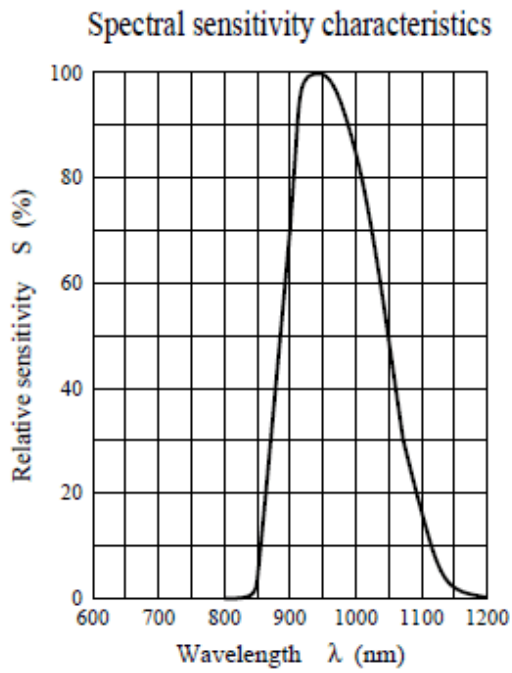
IR detectors are not suitable for visible light, as they are especially created for IR light. On the other hand, photocells are the opposite: good at detecting yellow/green light, but not for IR light.

Photocells do not have demodulators and they can distinguish any frequency within the speed of the photocell's response (which is about 1KHz). Alternatively, IR detectors have a demodulator inside that searches for modulated IR at 38 KHz.

IR detectors are straightforward - they either detect a 38KHz IR signal and output low (0V) or they do not find any and output high (5V). Photocells behave similar to resistors, the value changing according to the intensity of light that they are exposed to.

What You Can Measure

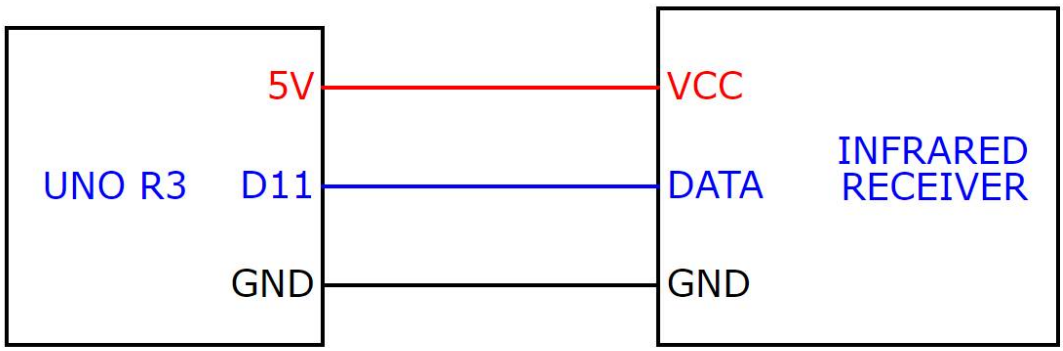




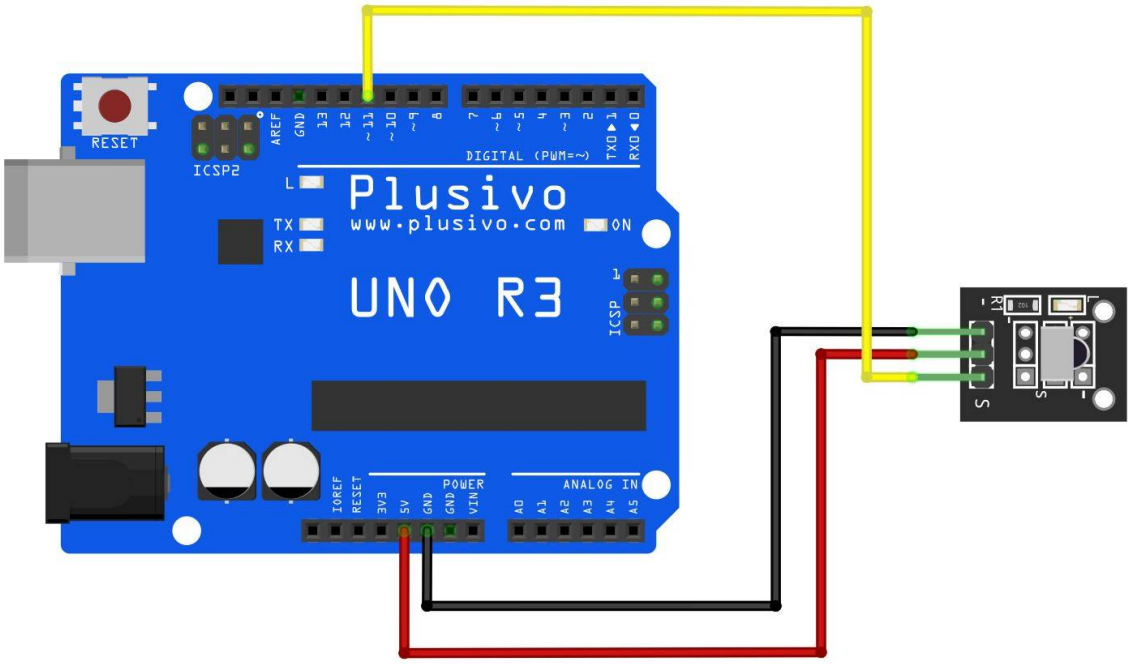
These graphs show us that the peak frequency detection is at 38 KHz and the peak LED color is 940 nm. You can use a frequency of about 35 to 41 KHz, but it won't identify as accurately from a bigger distance. Remember to look at the datasheet for your IR LED to check the wavelength and get the correct LED, preferably a 940nm.

14.4 Connection

Schematic



Wiring diagram



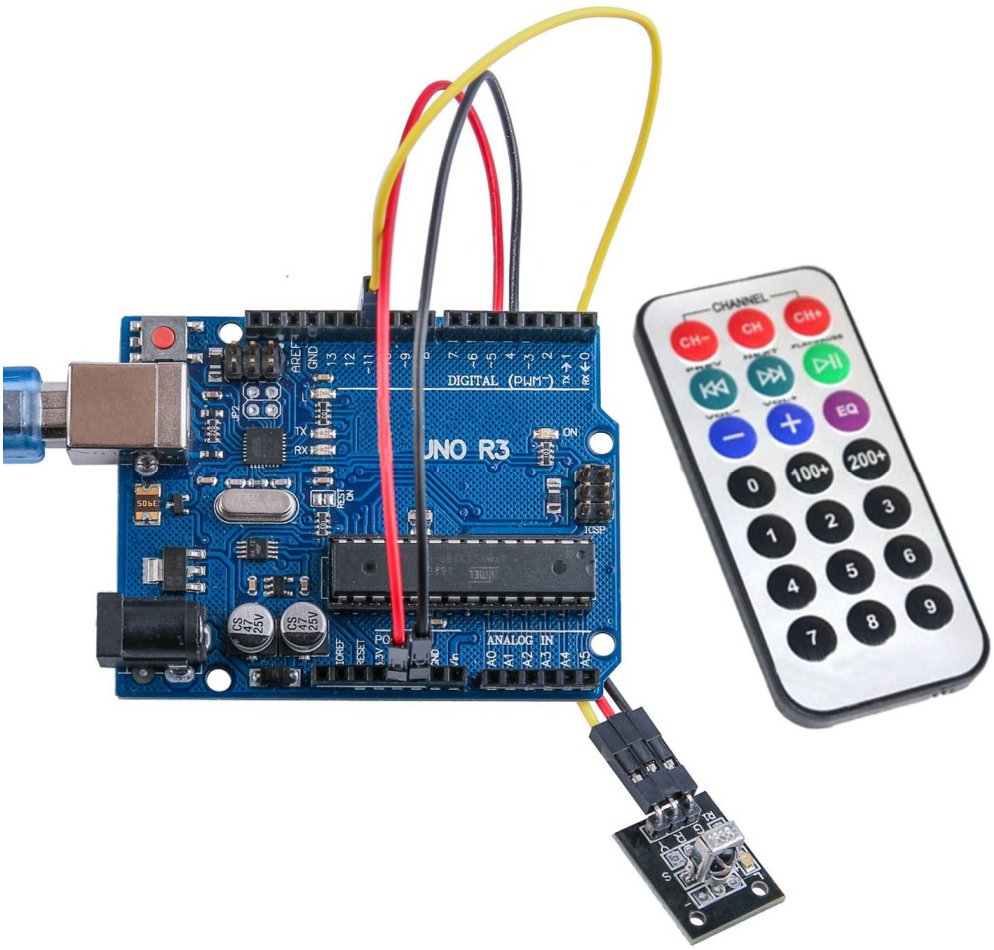
There are 3 connections to the IR Receiver: Signal, Voltage and Ground. The “-” is the Ground, “S” is signal, and middle pin is Voltage 5V.

14.5 Code

After wiring, please open the program located in the folder - Lesson 14 IR Receiver Module, and press UPLOAD to upload the code. If there are any errors, see Lesson 2 for details about program uploading. Before you can run this, make sure that you have installed the <IRremote > library or re-install it, if necessary. Otherwise, your code won't work. For details about loading the library file, see Lesson 1.

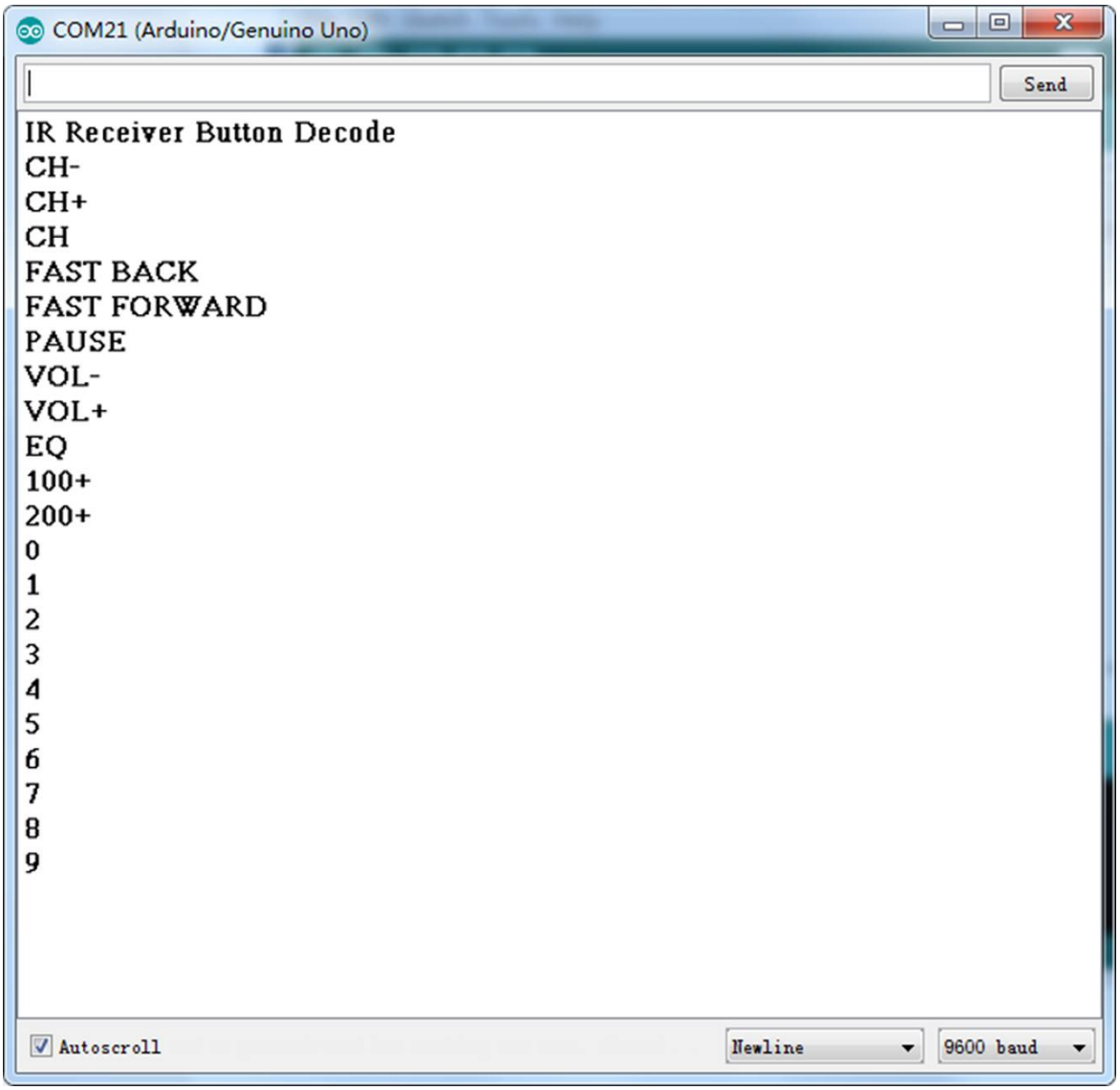
Now, we will move the <RobotIRremote> out of the Library folder, because that library clashes with the one we are going to be utilizing. You can undo this once you are done with this lesson. After installing the library, restart your IDE.

14.6 Example picture



Open the monitor, so you can see the following figure:

Press the Serial Monitor button to turn it on. The serial monitor is comprehensively introduced in Lesson 1.



15. Lesson 14 LCD Display

15.1 Overview

In this section, you will learn the basics on using an alphanumeric LCD display, which includes an LED backlight and can display two rows with up to 16 characters on each row. If you look closely, you can observe the individual rectangles for each character on the display and the pixels that create these characters.

We will be running the example program for the LCD library, following with getting showing the temperature on our display, using sensors.

15.2 Components Required

(1) x Plusivo Uno R3
(1) x LCD1602 module
(1) x Potentiometer (10k)
(1) x 830 tie-points Breadboard
(16) x M-M wires (Male to Male jumper wires)

15.3 Component Introduction

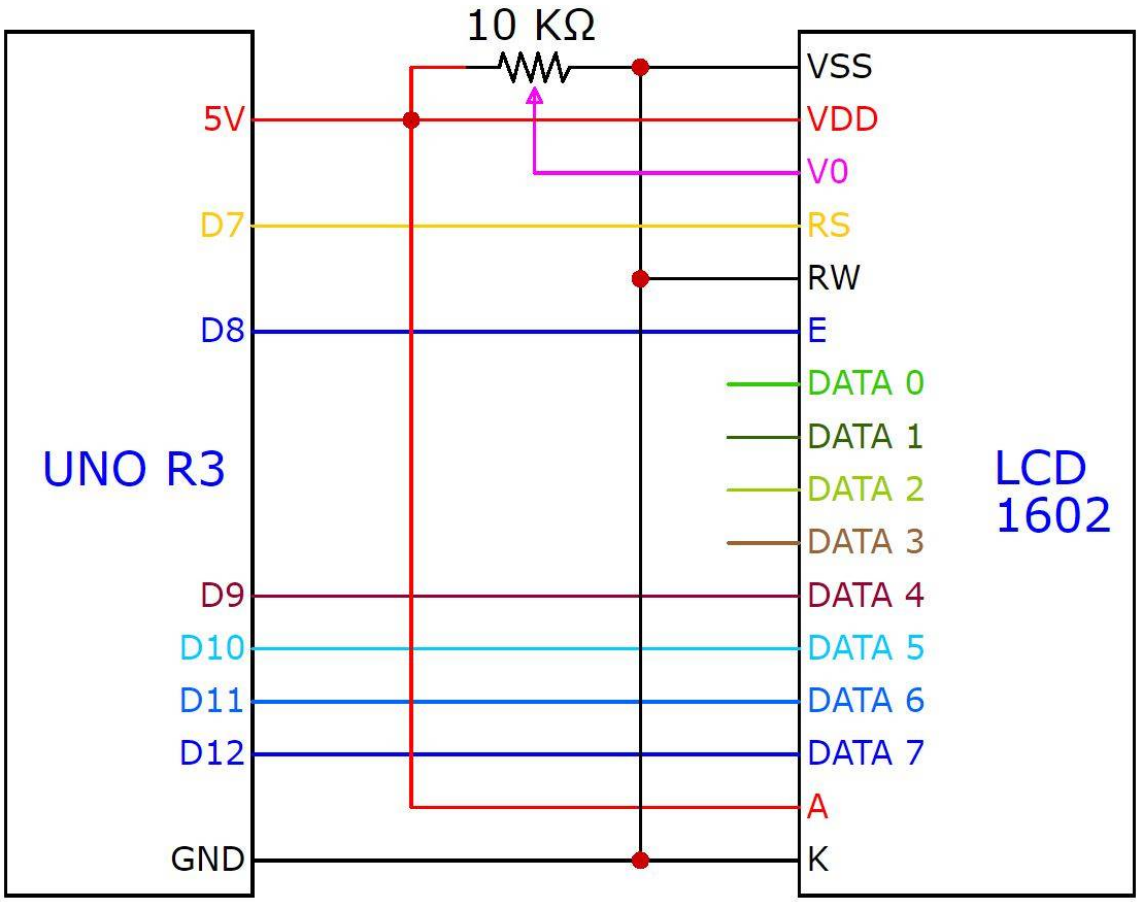
LCD1602

Introduction to the pins of LCD1602:

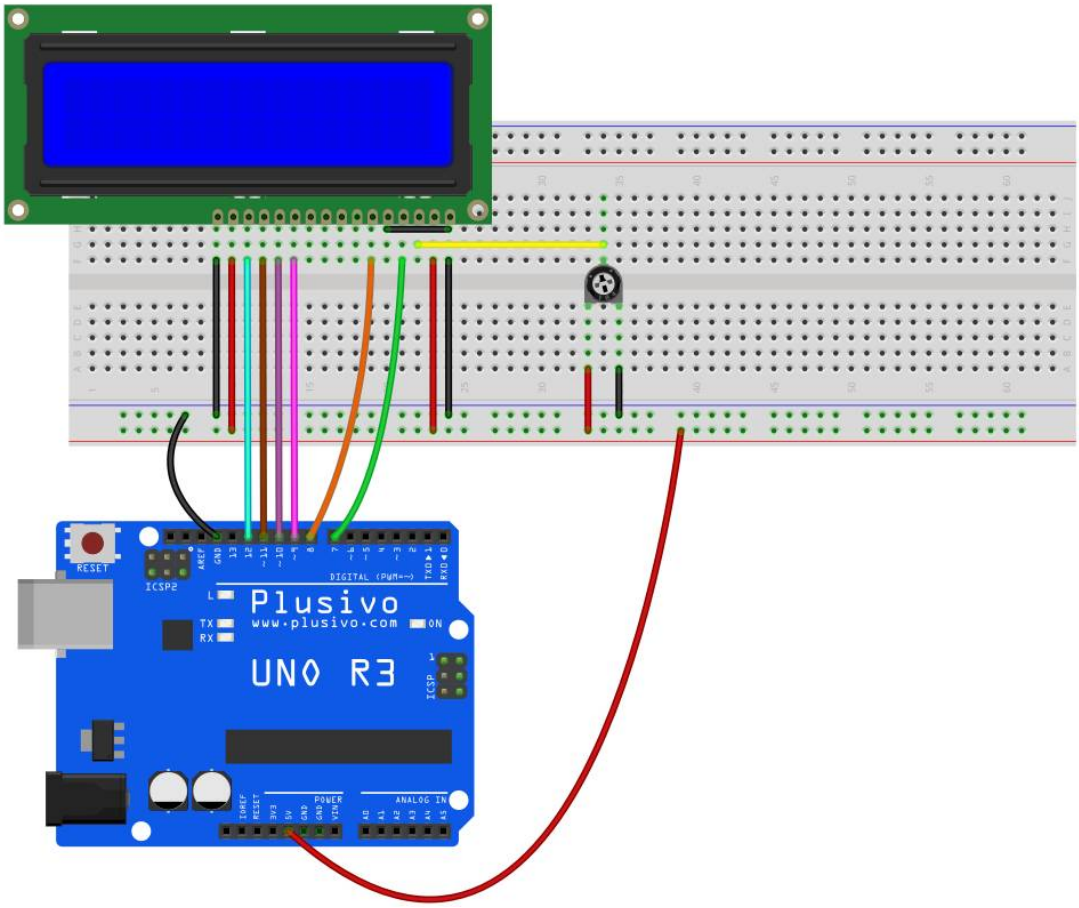
VSS: A pin that connects to ground
VDD: A pin that connects to a +5V power supply
VO: A pin that adjusts the contrast of LCD1602
RS: A register select pin that controls where in the LCD's memory you are writing data: either the data register, which holds what is displayed on the screen, or an instruction register, which is where the LCD's controller looks for instructions on what to do next.
R/W: A Read/Write pin that selects reading mode or writing mode
E: An enabling pin that causes the LDC module to execute relevant instructions when supplied with low-level energy.
D0-D7：Pins that read and write data
A and K: Pins that control the LED backlight

15.4 Connection

Schematic



Wiring diagram



The LCD display requires six UNO pins as digital outputs. Additionally, it needs 5V and GND connections. Aligning the display with the breadboard should aid you in identifying its pins effortlessly, especially if the breadboard has its rows numbered. In the case that your display is supplied without header pins attached to it, take a glance at the following instructions.

15.5 Code

After wiring, please open the program located in the folder - Lesson 22 LCD, and press UPLOAD to upload the code. If there are any errors, see Lesson 2 for details about program uploading. Before you can run this, make sure that you have installed the < LiquidCrystal >library or re-install it, if necessary. Otherwise, your code won't work. For details about loading the library file, see Lesson 1.

Load the sketch onto your UNO board. You should see the message 'hello, world' displayed, and a number that counts up from zero afterwards. Let's look at the following line of code:

```
#include <LiquidCrystal.h>
```

This informs our UNO that we want to use the Liquid Crystal library.

Next, this function defines the pins of the device linked to the pins of the display, which we had to change according to our project.

```
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);
```

Following, ensure that the backlight is turned on, and turn the potentiometer until you see the text message. The 'setup' function has two commands:

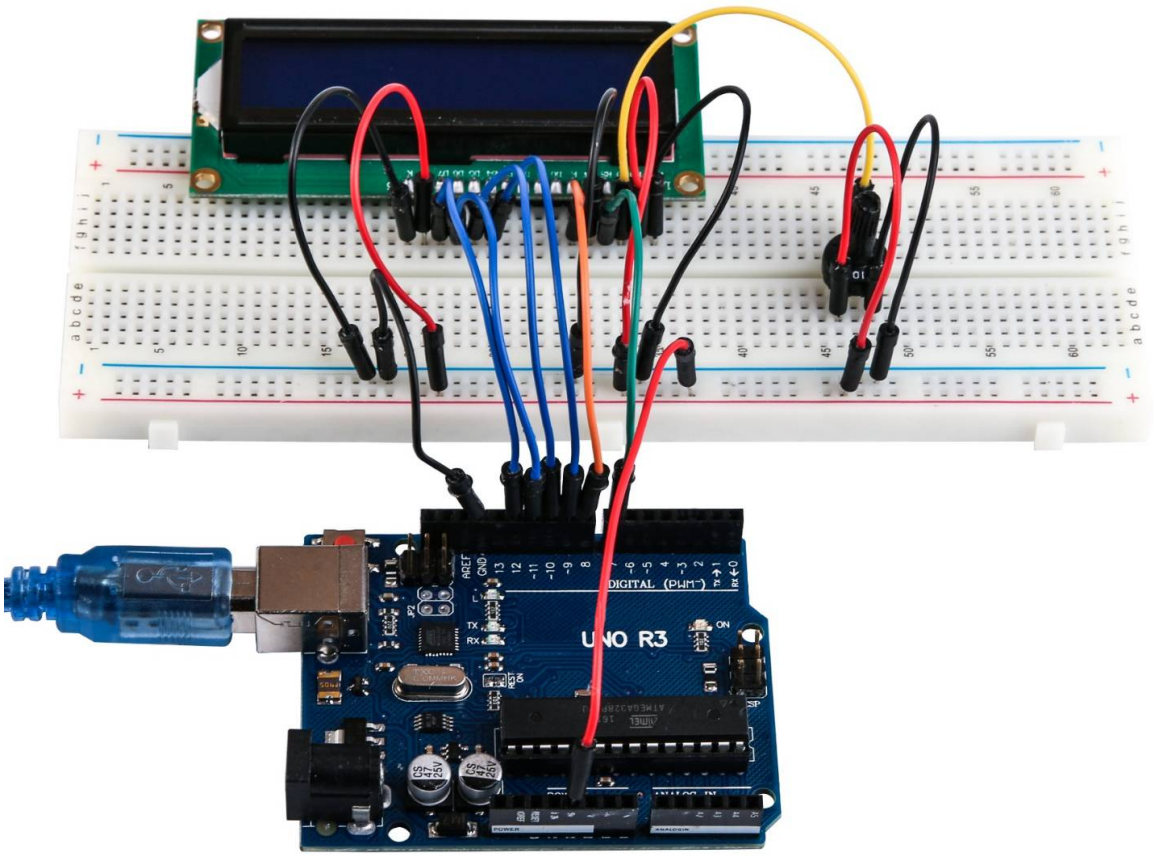
```
lcd.begin(16, 2);
lcd.print("Hello, World!");
```

The line of code in the beginning sends the Liquid Crystal library information about how many columns and rows are in that display. The print command holds the message that we want to be on the top line of the screen. In the 'loop' function there are two commands:

```
lcd.setCursor(0, 1);
lcd.print(millis()/1000);
```

This places the cursor (where the next text will show up) to column 0 & row 1, with 0 being the starting point for both measurements. Afterwards, it displays the time in milliseconds that has passed since the Arduino was reset.

15.6 Example picture



16. Lesson 15 Thermometer

16.1 Overview

This lesson will teach you to show the temperature using an LCD display.

16.2 Components Required

(1) x Plusivo Uno R3
(1) x LCD1602 Module
(1) x 10k ohm resistor
(1) x Thermistor
(1) x Potentiometer
(1) x 830 tie-points Breadboard
(18) x M-M wires (Male to Male jumper wires)

16.3 Component Introduction

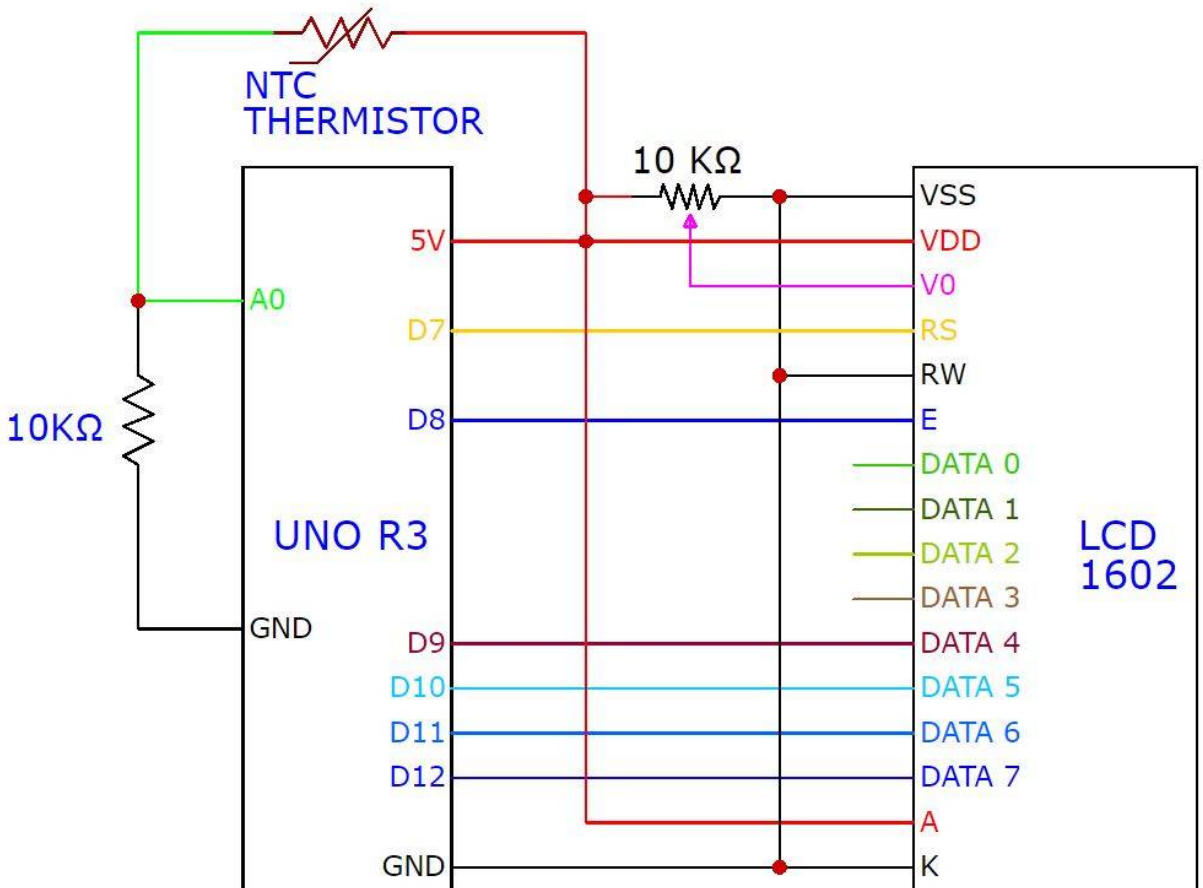
Thermistor

A thermistor is simply a thermal resistor - a resistor that changes its resistance according to the temperature. Theoretically, all resistors are thermistors, because their resistance modifies a little with temperature, but it is generally too little to determine. On the other hand, with thermistors, the resistance changes intensely with temperature, 100 ohms or more per degree, to be exact.

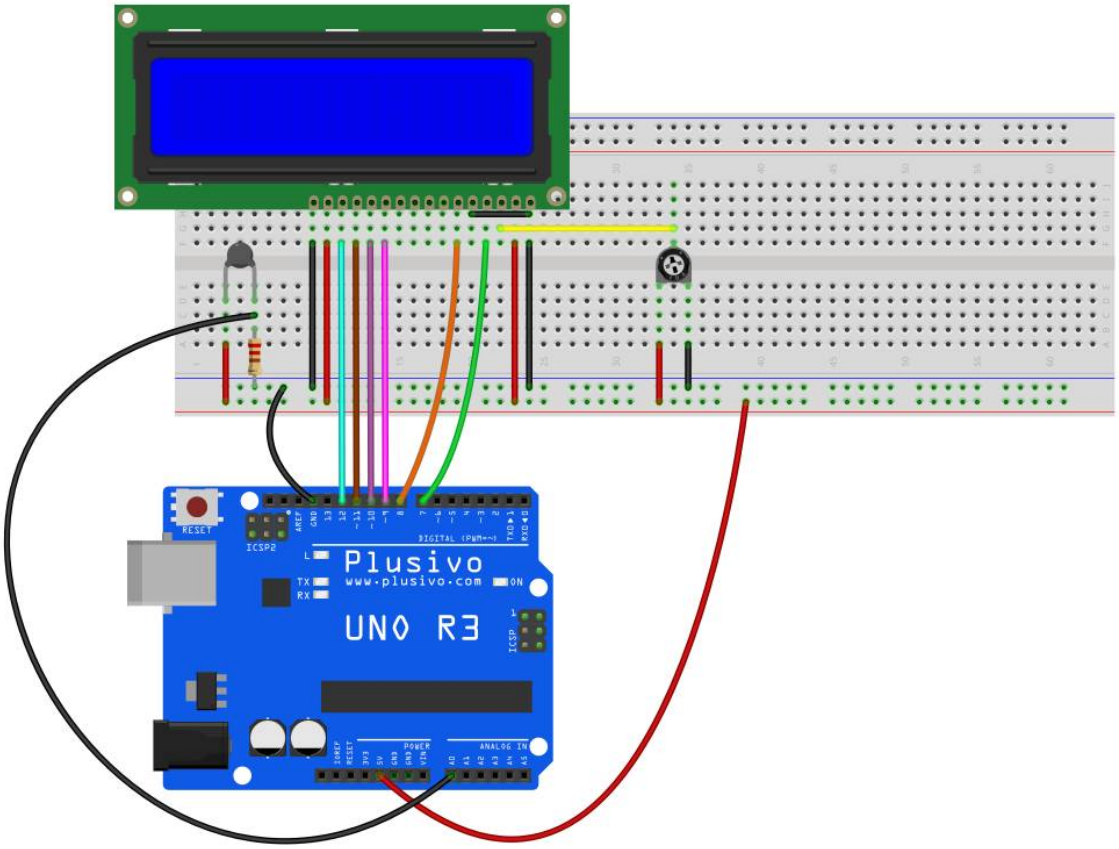
There are two kinds of thermistors, PTC (positive temperature coefficient) and NTC (negative temperature coefficient). For the most part, NTC sensors are used for temperature measurements, while PTCs are used as resettable fuses. The working mechanism is that the temperature and resistance increase at the same pace, so the more current passes through them, the more they heat up and resist. For their flexibility, they prove very useful for protecting circuits!

16.4 Connection

Schematic



Wiring diagram



The breadboard layout similar to the one in Lesson 22, so it will make it easier for you if you still have this particular layout. The 10 kΩ resistor and thermistor new additions to the board and some of the jumper wires have been moves slightly.

16.5 Code

After wiring, please open the program located in the folder - Lesson 23, and press UPLOAD to upload the code. If there are any errors, see Lesson 2 for details about program uploading. Before you can run this, make sure that you have installed the < LiquidCrystal >library or re-install it, if necessary. Otherwise, your code won't work. For details about loading the library file, see Lesson 1.

The sketch is largely built on the one in lesson 22. Load it up and you can see that by simply putting your finger on, it will increase the temperature reading. You can write a comment line to explain what pins are used.

```
// BS   E   D4 D5   D6 D7
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);
```

This makes it simpler for you, should you wish to change them. In the 'loop' function, we have to convert the raw value from the temperature sensor into an actual temperature and then, we have to display it.

Firstly, let's glance at the temperature calculations:

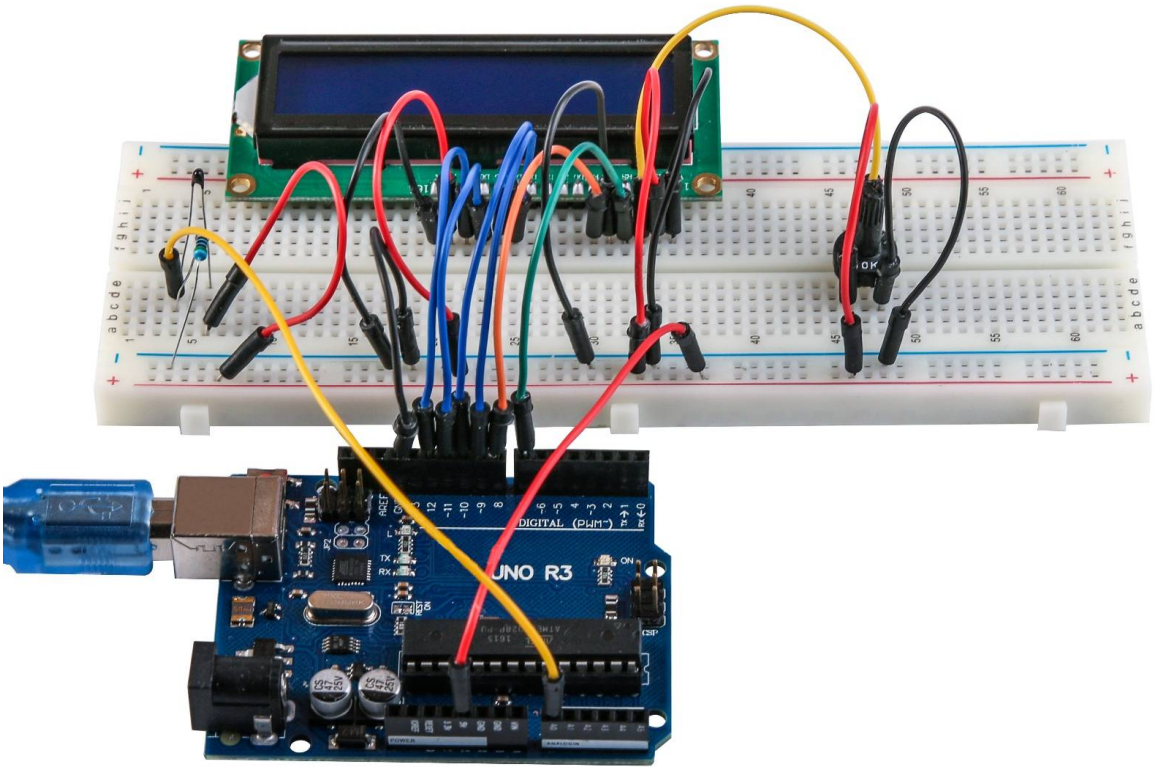
```
int tempReading = analogRead(tempPin);
double tempK = log(10000.0 * ((1024.0 / tempReading - 1)));
tempK = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * tempK * tempK )) * tempK );
float tempC = tempK - 273.15;
float tempF = (tempC * 9.0)/ 5.0 + 32.0;
```

Displaying values that are constantly changing on an LCD display can be complicated. The biggest problem we may encounter is that the reading does not always contain the same number of digits. For example, if the temperature changed from 108.50 to 98.00 then the extra digit from the old reading can be accidentally left on the display. To avoid this, repeat the lines of the LCD default values code each time around.

```
lcd.setCursor(0, 0);
lcd.print("Temp        C   ");
lcd.setCursor(6, 0);
lcd.print(tempF);
```

The comment reminds you of the 16 columns of the display. You have the ability print a string of that length, with spaces left for the actual readings. Move the cursor to the appropriate position and then print it.

16.6 Example picture



17. Lesson 16 Eight LED with 74HC595

17.1 Overview

This lesson will be showing you how to use eight large red LEDs with an Arduino without using up 8 output pins! Even though you could wire up eight LEDs with a resistor for each of them to a pin, you would quickly run out of pins for other components as we usually have an array of devices such as buttons, sensors, servos, etc. for most projects. Instead, we are using a chip called the 74HC595 Serial to Parallel Converter, which has eight outputs and three inputs that you use give data to.

This chip makes the board a little slower in use (you can change the LEDs about 500,000 times a second compared to 8,000,000 a second) but it's still much faster than the human eye can distinguish, so it's not an issue.

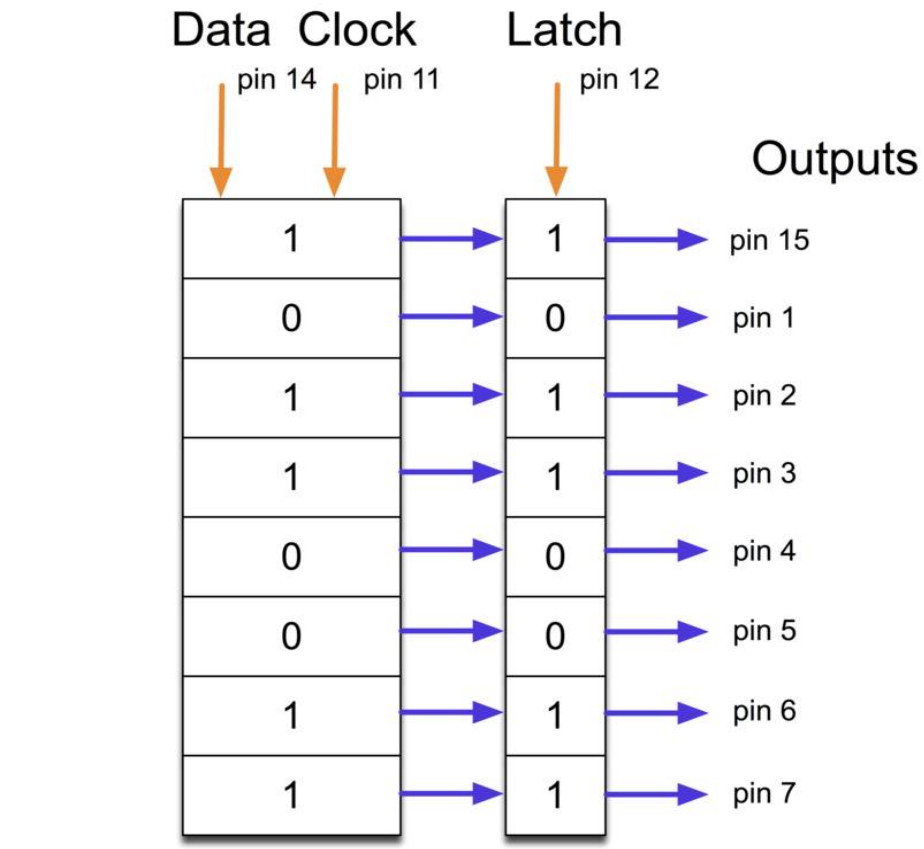
17.2 Components Required

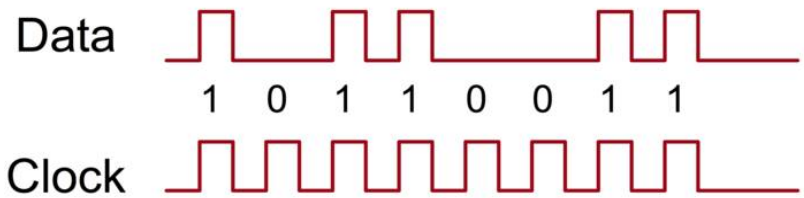
(1) x Plusivo Uno R3
(1) x 830 tie-points breadboard
(8) x leds
(8) x 220 ohm resistors
(1) x 74hc595 IC
(14) x M-M wires (Male to Male jumper wires)

17.3 Component Introduction

74HC595 Shift Register

The shift register is a kind of chip containing eight memory locations, with the values 1 or 0. We input the data using the 'Data' and 'Clock' pins of the chip to set these values on or off.



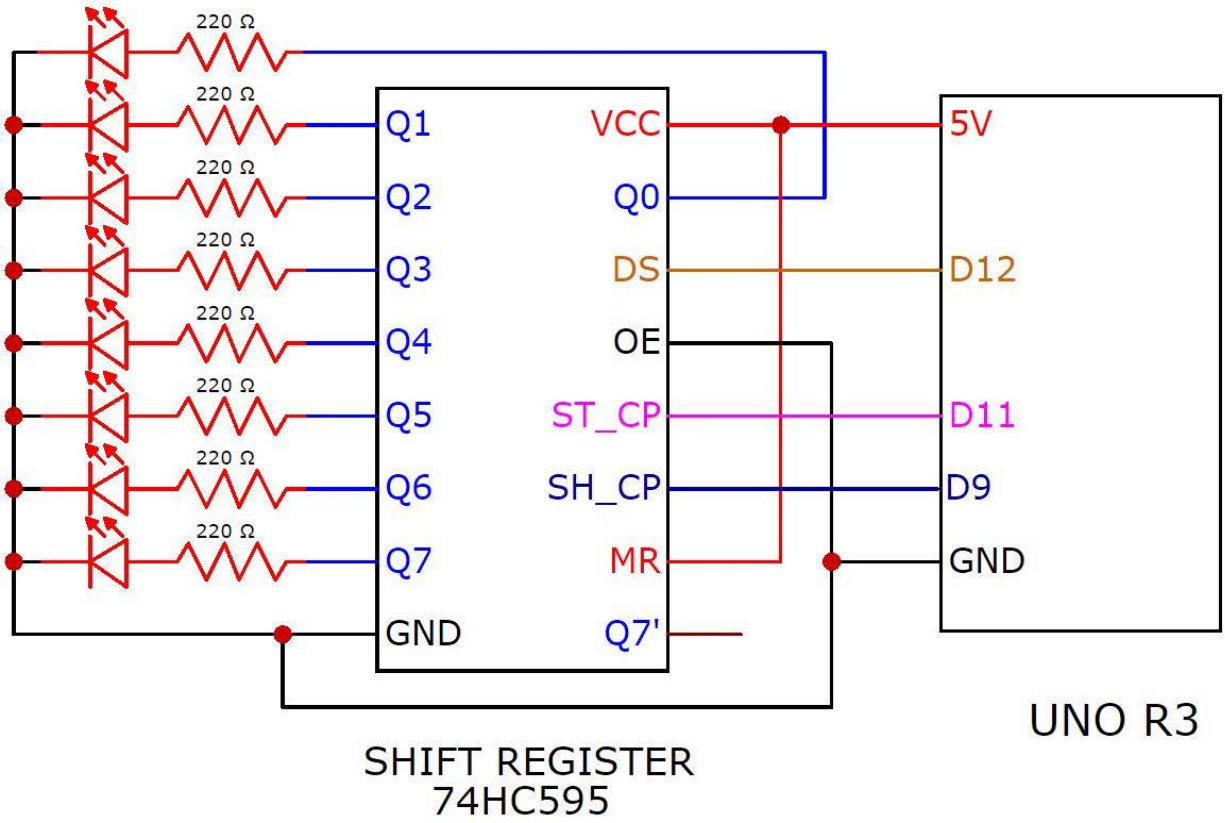


The clock pin should receive eight pulses. For each of these, if the data pin is either high or low, then a 1 or a 0 goes into the shift register in correspondance. After all the pulses are received, enabling the 'Latch' pin copies those eight values to the latch register.

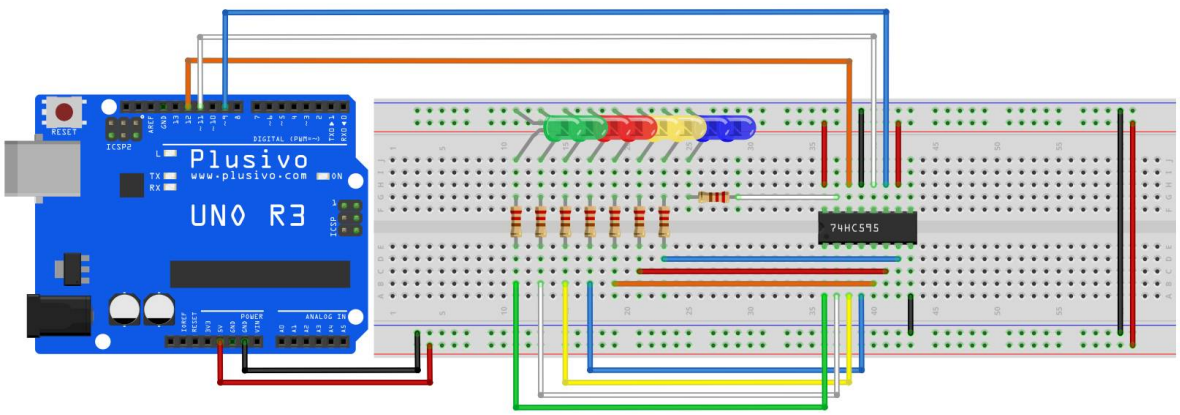
Additionally, the chip has an output enable (OE) pin, utilized to wither enable or disable the outputs collectively. This can be linked to a PWM-capable Arduino pin, allowing the use of 'analogWrite' to control the light level of the LEDs.

17.4 Connection

Schematic



Wiring diagram



We have eight LEDs and eight resistors to connect, so let us begin.

We recommend you place the 74HC595 chip in the beginning, as almost everything else connects to it. Place it so the U-shaped notch is towards the top of the breadboard. Pin 1 of the chip is to the left of this notch.

Digital 12 from the UNO goes to pin #14 of the shift register
Digital 11 from the UNO goes to pin #12 of the shift register
Digital 9 from the UNO goes to pin #11 of the shift register

All except one of the outputs is on the left side of the chip, where the LEDs are as well, to make the connection effortless.

Then, place the resistors on the breadboard. You should check that the leads of the resistors are not touching each other before you connect the power to your UNO. You can shorten the leads so that they are closer to the surface of the breadboard, if you are encountering any difficulties

Afterwards, put the LEDs in place. The longer positive leads must all be near the chip, irrespective of which side of the breadboard they are on. Connect the jumper wires, but remember the one that goes from pin 8 of the IC to the GND column.

Attempt the sketch mentioned. Each LED should light in an order until they are all on. Afterwards, they all turn off and the sequence repeats.

17.5 Code

After wiring, please open the program located in the folder - Lesson 24 Eight LED with 74HC595, and press UPLOAD to upload the code. If there are any errors, see Lesson 2 for details about program uploading.

We start by defining the three pins we will be utilizing as the digital outputs that will be connected to the latch, clock and data pins of the 74HC595.

```
int latchPin = 11;
int clockPin = 9;
int dataPin = 12;
```

Then, we define a variable called ‘leds’, which holds the order of the LEDs being turned on or off. Data of type 'byte' represents numbers using eight bits, with each one being on or off, so this is what we will be using to monitor which of our eight LEDs are powered on or off.

```
byte leds = 0;
```

The 'setup' function establishes the three pins to be used as digital outputs.

```
void setup()
{
    pinMode(latchPin, OUTPUT);
    pinMode(dataPin, OUTPUT);
    pinMode(clockPin, OUTPUT);
}
```

The 'loop' function turns all the LEDs off in the beginning, by setting the variable ‘leds' to 0. Afterwards, it calls the 'updateShiftRegister' function, which sends the 'leds' sequence to the shift register in order to turn all the LEDs off.

The loop (‘for’) function pauses for half a second and then counts through the number of leds (‘i’ variable), from 0 to 7. Every time, it utilizes the 'bitSet' function to set the bit that controls that LED in the variable 'leds'. It then calls 'updateShiftRegister' so the LEDs are up to date. Afterwards, there is a half second delay before the incrementation of 'i' and the next LED is turned on.

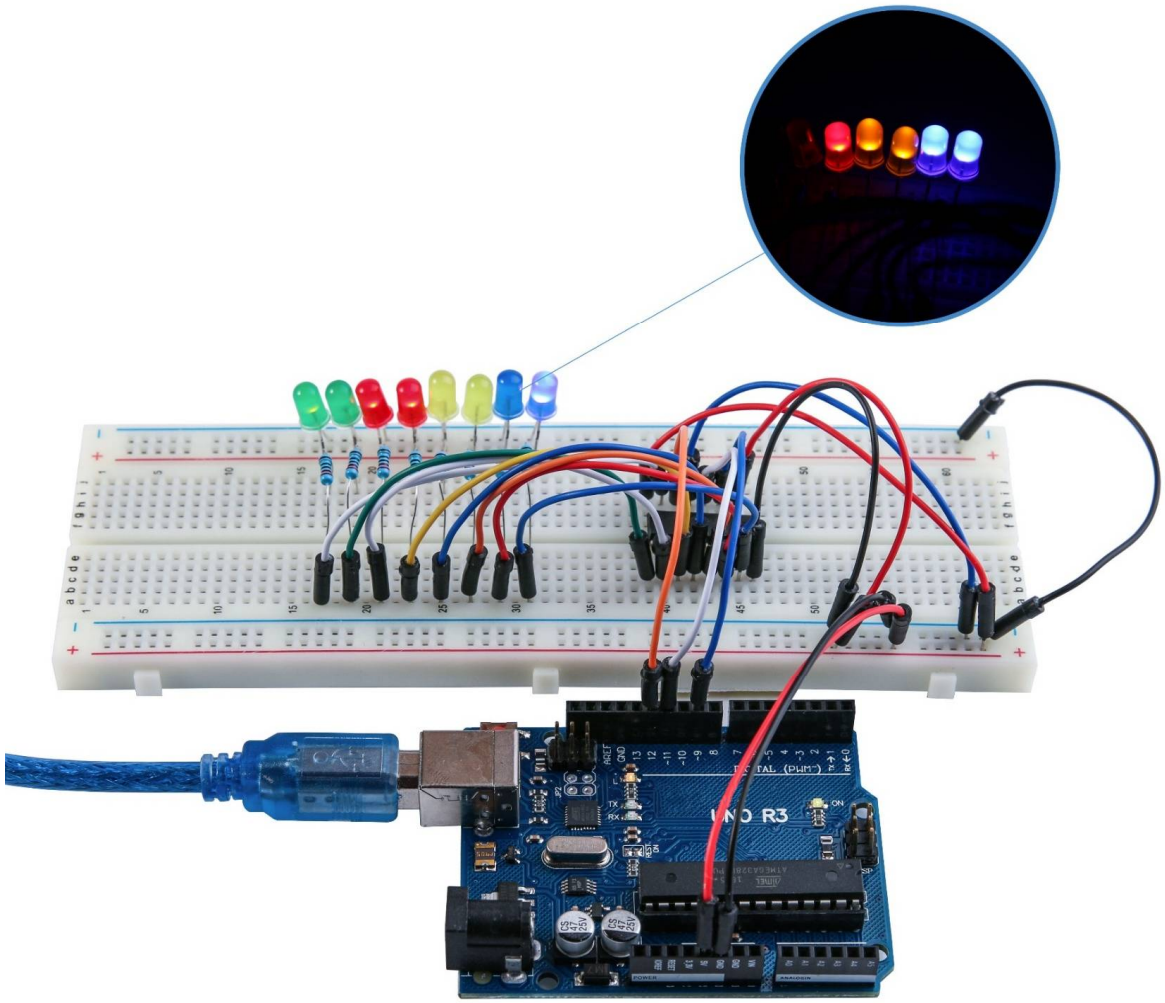
```
void loop()
{
    leds = 0;
    updateShiftRegister();
    delay(500);
    for (int i = 0; i < 8; i++)
    {
      bitSet(leds, i);
      updateShiftRegister();
      delay(500);
    }
}
```

First of all, 'updateShiftRegister' sets the latchPin to low, following with a call to the 'shiftOut' function, which takes four parameters: the first two are the pins for Data and Clock, the third is for which end of the data you wish to begin at and the last is for the actual data to be shifted into the shift register. We are beginning with the rightmost bit, also called the 'Least Significant Bit' (LSB). Then, we set the 'latchPin' to high again.

```
void updateShiftRegister()
{
    digitalWrite(latchPin, LOW);
    shiftOut(dataPin, clockPin, LSBFIRST, leds);
    digitalWrite(latchPin, HIGH);
}
```

To turn one of the LEDs off, you would call the ‘bitClear’ function with the 'leds' variable, and then calling the 'updateShiftRegister' to bring the actual LEDs up to date.

17.6 Example picture



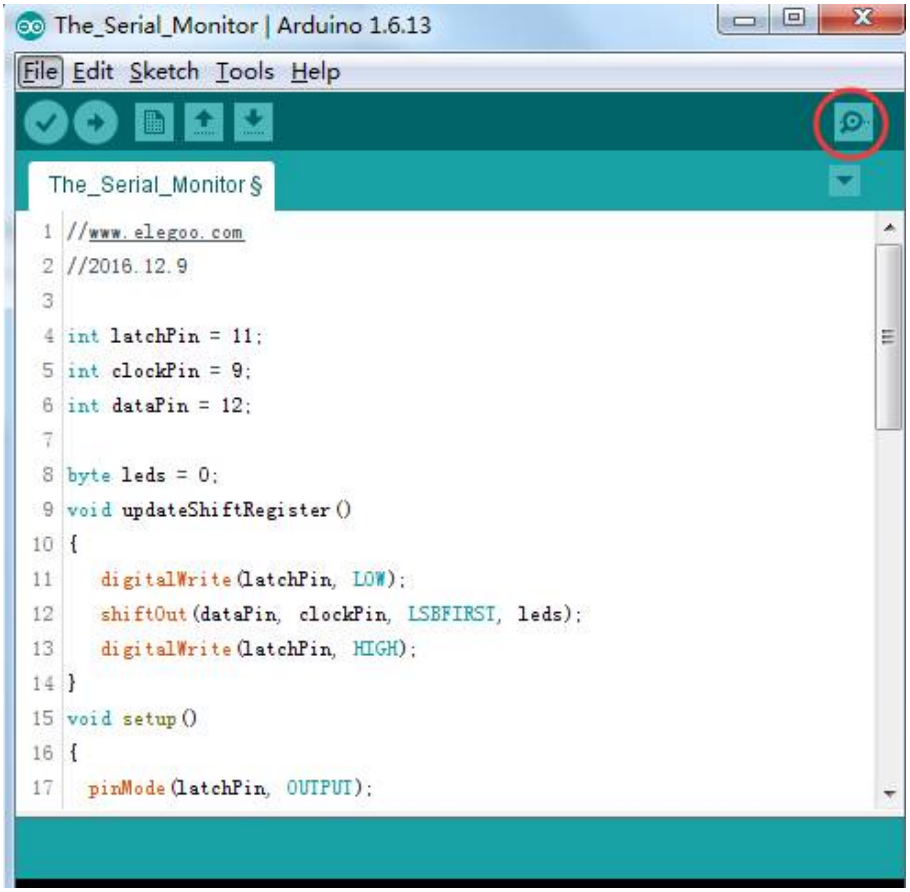
18. Lesson 17 The Serial Monitor

18.1 Overview

This lesson will be based on the former one, with the option of managing the LEDs from your computer. We will be utilizing the Serial Monitor, which is the connection between the computer and the Arduino, enabling you to send and receive debugging text messages. You will be able to send commands from your computer to turn the LEDs on, using the same exact parts and a similar breadboard layout.

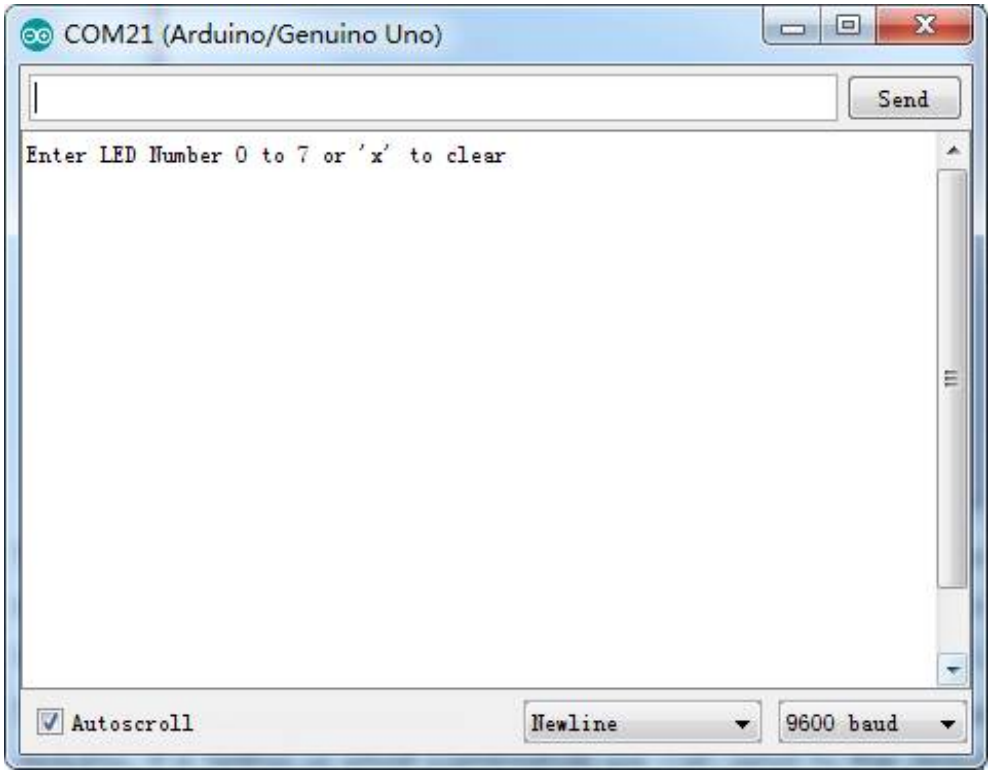
18.2 Steps taken

After uploading this code onto your device, click on the right-most button (as shown below) on the toolbar in the IDE.



This window will open.

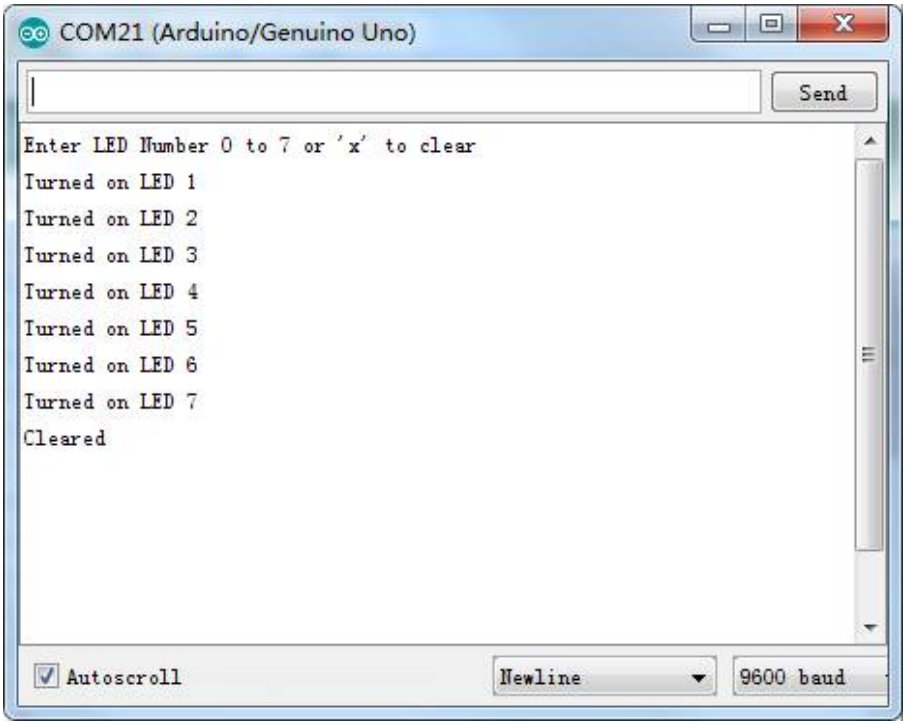
Press the Serial Monitor button to turn on the serial monitor. We discussed the serial monitor in detail in Lesson 1.



This window is called the Serial Monitor and it is part of the IDE software. Its purpose is to allow the UNO board and your computer to communicate.

The message “Enter LED Number 0 to 7 or 'x' to clear” has been sent by the device, which informs us of the commands we have the choice to send to the Arduino: 'x' to turn all the LEDs off or the number of the LED you want to turn on (where 0 is the bottom one going to 7 for the top one). Attempt the following commands into the top area of the Serial Monitor and press 'Send': x 0 3 5

If the LEDs are already turned off, typing ‘x’ will not do anything, but as you enter each number, the matching LED will turn on and you will get a confirmation message. The Serial Monitor will show as seen below.



Type ‘x’ one more time and click on ‘Send’ to turn off all the LEDs.

18.3 Code

After wiring, please open the program located in the folder - Lesson 25 The Serial Monitor, and press UPLOAD to upload the code. If there are any errors, see Lesson 2 for details about program uploading.

As anticipated, this sketch is largely built on the one used in Lesson 24, so we only need to explain the new lines of code. You may deem it useful to refer to the complete sketch in your IDE. In the 'setup' function, there are three new lines at the end:

```
void setup()
{
pinMode(latchPin, OUTPUT);
pinMode(dataPin, OUTPUT);
pinMode(clockPin, OUTPUT);
updateShiftRegister();
Serial.begin(9600);
while (! Serial); // Wait until Serial is ready - Leonardo
Serial.println("Enter LED Number 0 to 7 or 'x' to clear");
 }
```

First of all, there is the 'Serial.begin(9600)' command. The value 9600 is the 'baud rate' of the connection and it determines the speed of the data being sent. This can be modified to a higher value, but you will also need to change the Serial monitor to the same value. For now, leave it at 9600. This function basically starts serial communication, so that the UNO can send out commands through the USB connection.

The 'while' loop assures us that there is something on the other side of the USB connection for the device to communicate with. This line is only needed if you are using an Leonardo board because the Arduino board routinely resets itself when you open the Serial Monitor.

The final new lines in 'setup' send out the message we see at the top of the Serial Monitor. Let's take a look at the 'loop' function:

```
void loop()
{
if (Serial.available())
{
char ch = Serial.read();
if (ch >= '0' && ch <= '7')
{
int led = ch - '0';
bitSet(leds, led);
updateShiftRegister();
Serial.print("Turned on LED ");
Serial.println(led);
}
if (ch == 'x')
{
leds = 0;
updateShiftRegister();
Serial.println("Cleared");
}
}
}
```

If the call to the built-in function 'Serial.available()' is not 'true', nothing else will happen. Serial.available() will return 'true' if the buffer, where incoming messages are held, is not empty. If a message has been accepted, then we move on to the next part of the code:

```
char ch = Serial.read();
```

This extracts the next character from the buffer and assigns it to the variable 'ch'. The variable 'ch' is of type 'char', which holds a single character. This variable will either be a number between 0 and 7 or the letter 'x'.

The 'if' statement in the following line checks if it is a single digit by confirming that the value of 'ch' is between '0' and '7'. While it may seem peculiar comparing characters like this, it is perfectly reliable, as each character is illustrated by a unique number, called an ASCII value. Thus,

when we compare characters using the operators <= and >=, we are actually comparing the ASCII values. In the case that the ‘if’ result is positive, we proceed on to the next line:

Next, we are subtracting the digit '0' from whatever digit was entered. Thus, typing '0' then '0' - '0' will equal 0, whereas typing '7' then '7' - '0' will equal the number 7 as the ASCII values are actually the ones being used in the operation. Seeing as we know the number of the LED we want to power on, we just need to store that in the variable 'leds' and update the shift register.

```
bitSet(leds, led);
updateShiftRegister();
```

The following lines send a confirmation message back to the Serial Monitor.

```
Serial.print("Turned on LED ");
Serial.println(led);
```

The first one uses Serial.print, rather than Serial.println, as we do not want to begin a new line after printing, to print the message in two parts: 'Turned on LED ' and the number of the LED as an ‘int’ variable. Ensuing the 'if' statement that handles the single digit case, there is a second 'if' one that checks whether 'ch' is the letter 'x'.

```
if (ch == 'x')
{
leds = 0;
updateShiftRegister();
    Serial.println("Cleared");
}
```

In the affirmative case, it clears each one of the LEDs and sends a message of confirmation.

19. Lesson 18 Photocell

19.1 Overview

This lesson will be showing you the basics on how to check the intensity of light and how to use that value to manage the number of LEDs being lit., using a photocell.

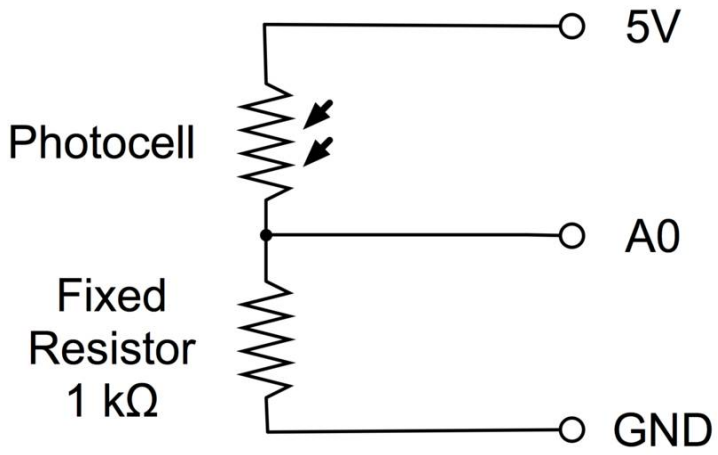
19.2 Components Required

(1) x Plusivo Uno R3
(1) x 830 tie-points breadboard
(8) x leds
(8) x 220 ohm resistors
(1) x 1k ohm resistor
(1) x 74hc595 IC
(1) x Photoresistor (Photocell)
(16) x M-M wires (Male to Male jumper wires)

19.3 Component Introduction

PHOTOCELL

The photocell used is of a type called a light dependent resistor ’). From the name, we draw the conclusion that these components behave like a resistor, except that the resistance changes according to the level of light they are exposed to, with values between 50 kΩ in near darkness and 500 Ω in bright light. To transform this fluctuating value of resistance into something we can measure on our Arduino’s analog input, it has to be converted into a voltage. The most basic way to do so is to integrate it with a fixed resistor.



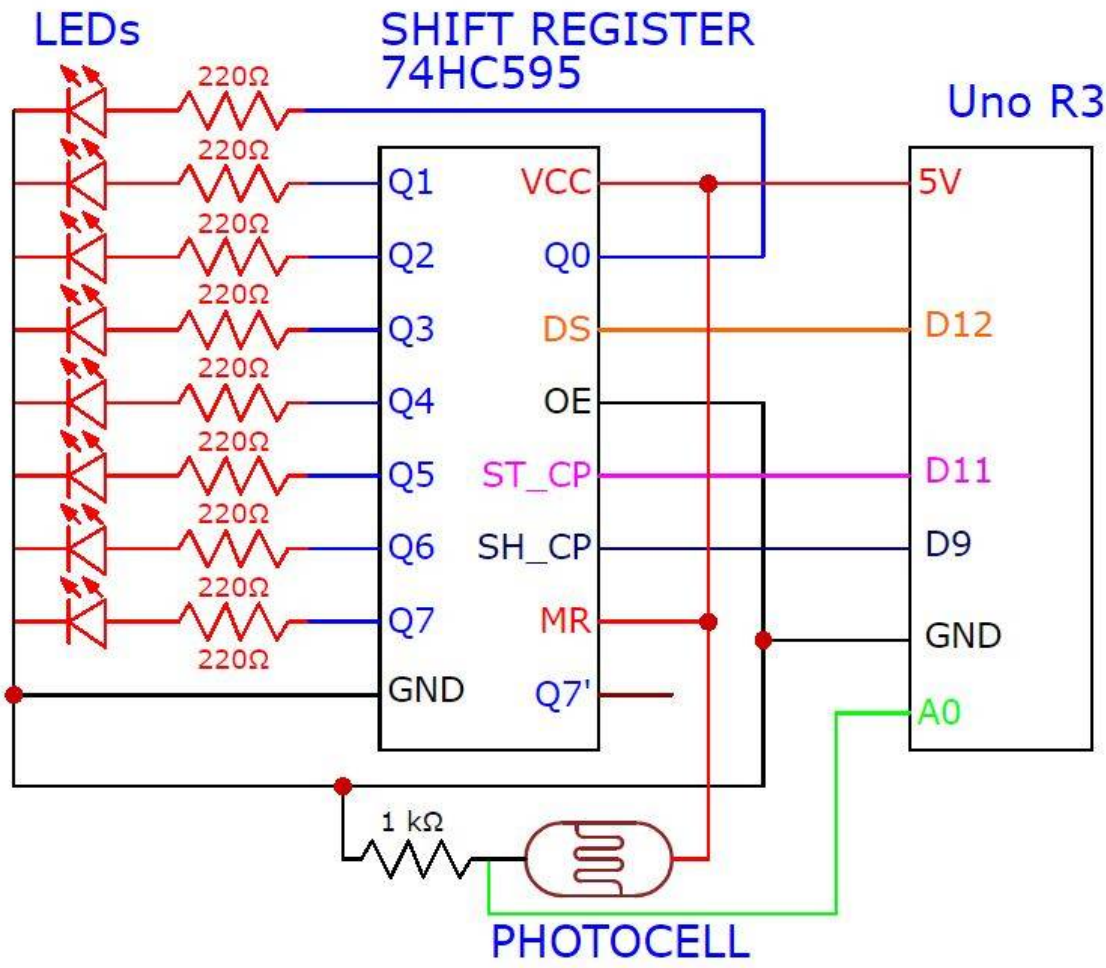
Photocell

The resistor and photocell together behave like a pot. Thus, as the light is very bright, the resistance of the photocell is insignificant contrasting with the fixed value resistor, so it behaves as if it were a pot turned to maximum.

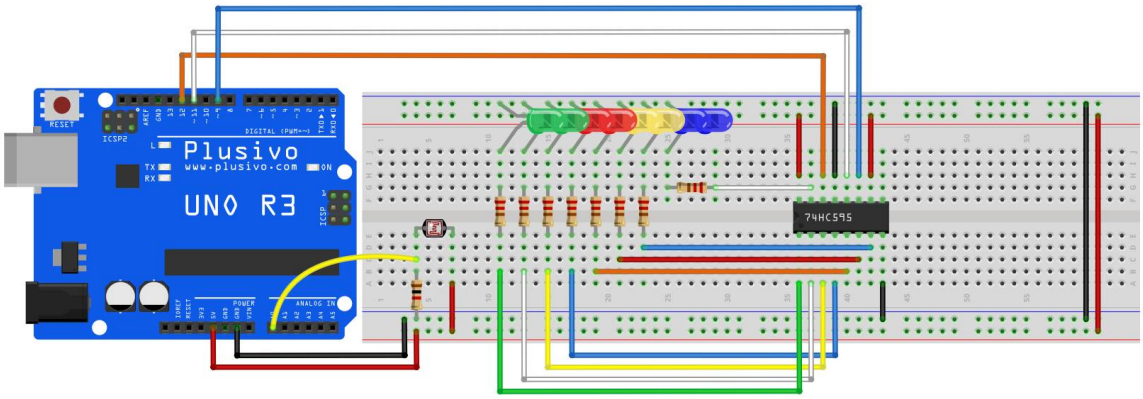
Alternatively, when the photocell is in dull light, the resistance becomes greater than the fixed 1 kΩ resistor and it is as if the pot were being turned towards GND. Use the code found in the next section and try covering the photocell or holding it near a light source to see the values change.

19.4 Connection

Schematic



Wiring diagram



19.5 Code

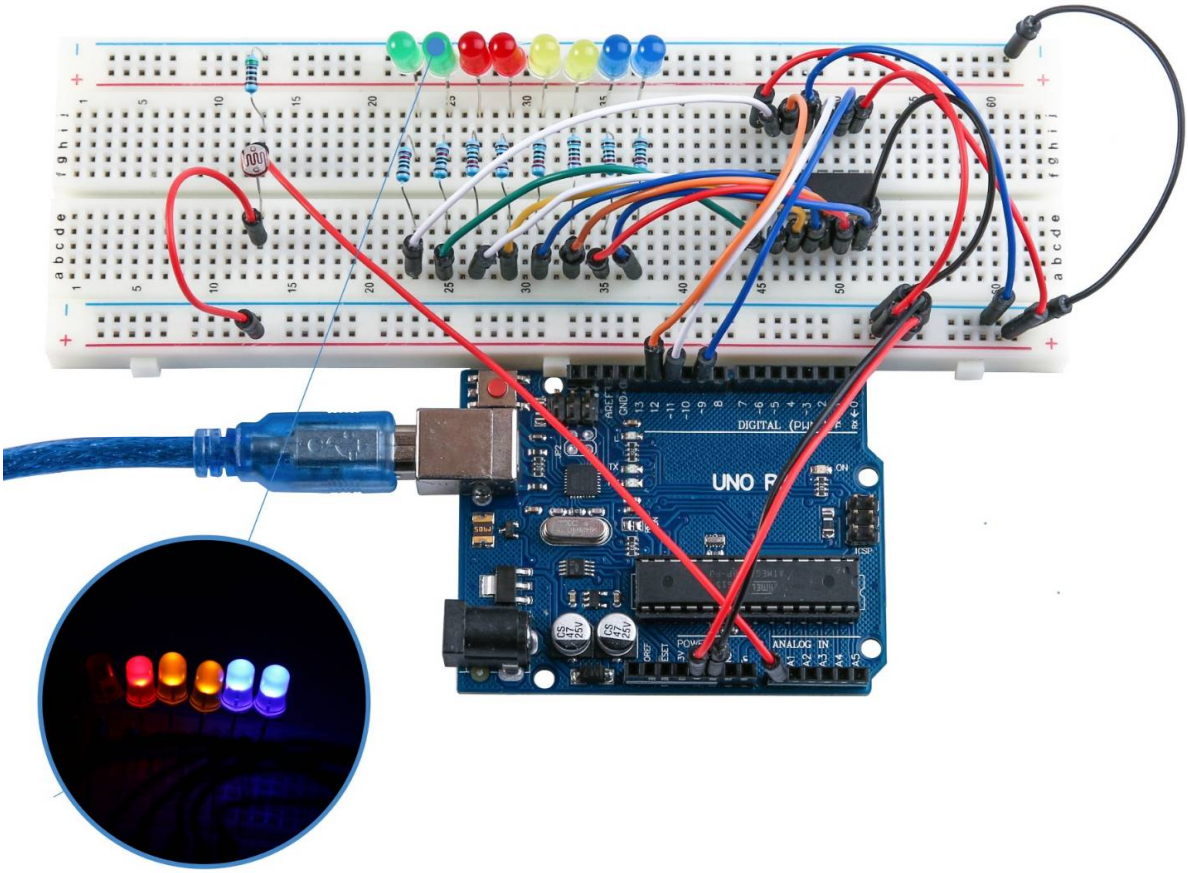
After wiring, please open the program located in the folder - Lesson 26 Photocell, and press UPLOAD to upload the code. If there are any errors, see Lesson 2 for details about program uploading.

Firstly, we see that we have modified the name of the analog pin to be 'lightPin' rather than 'potPin', as we do not have a pot connected any more. The other essential change to the code is the line where we determine the number of LEDs to light:

```
int numLEDSLit = reading / 57;          // all LEDs lit at 1k
```

In this situation, we will divide the raw reading by 57 rather than 114, so half as much as we did previously with the pot to split it into nine areas, from no LEDs lit to all eight lit. We also have to mind the fixed 1 kΩ resistor. Thus, when the photocell has the same resistance as the resistor, the raw reading will be 1023 / 2 = 511, which means that all the LEDs are being lit and that ‘numLEDSLit’ will be 8.

19.6 Example picture

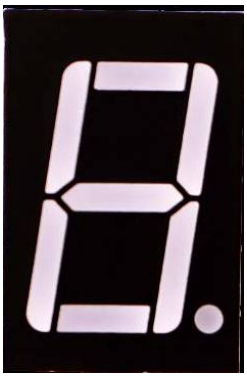


20. Lesson 19 74HC595 And Segment Display

20.1 Overview

After studying the previous lessons, we will learn the basics on using the 74HC595 shift register to control the segment display, which will show a number from 9 to 0.

20.2 Components Required

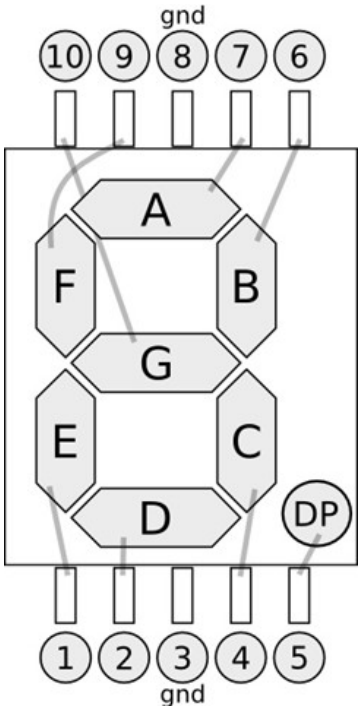


(1) x Plusivo Uno R3
(1) x 830 tie-points breadboard
(1) x 74HC595 IC
(1) x 1 Digit 7-Segment Display
(8) x 220 ohm resistors
(26) x M-M wires (Male to Male jumper wires)

20.3 Component Introduction

Seven segment display

Below, you can see the seven-segment pin diagram.

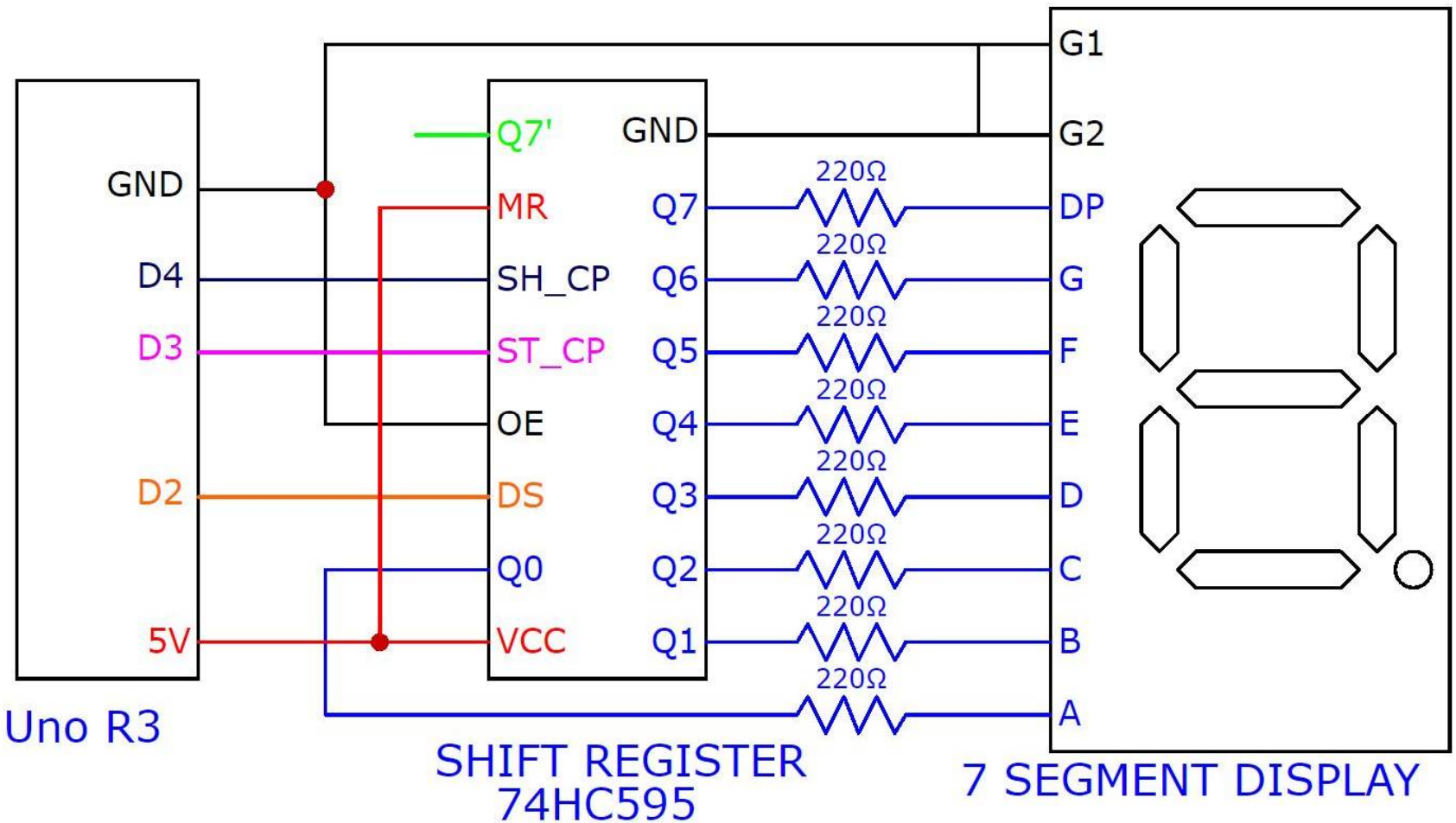


The ten digits (0-9) correlate with each segment as follows (the table below only applies to a common cathode device; when using a common anode one, every 1 0 0 sequence in the table needs to be replaced by 1):

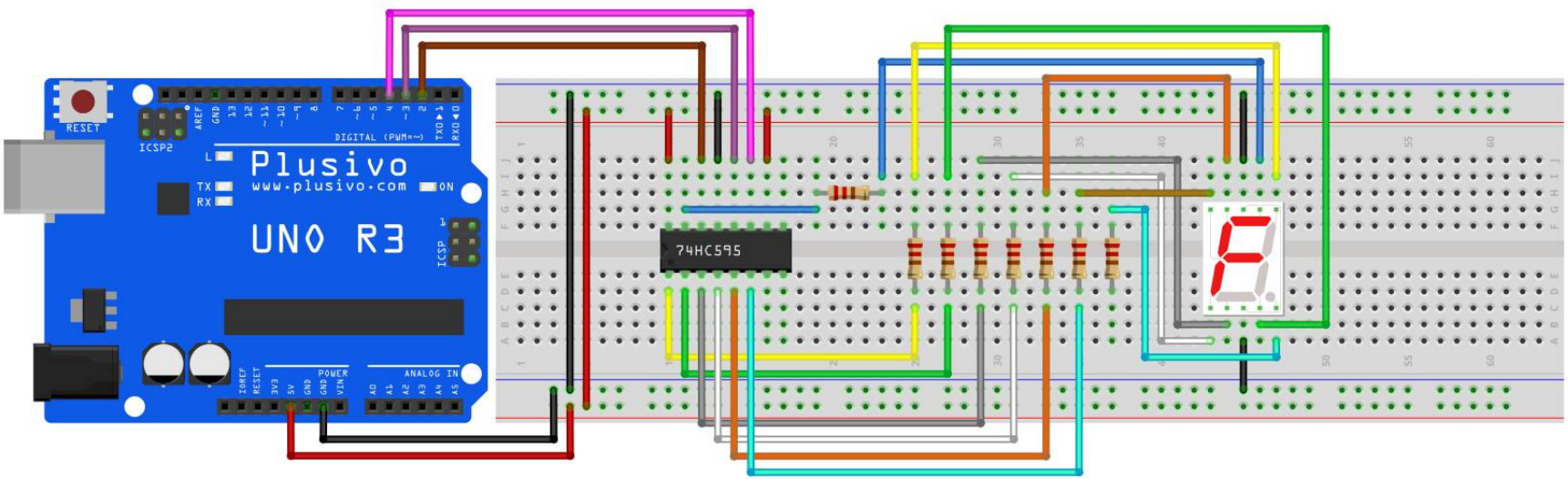
Display digital	dp	a	b	c	d	e	f	g
0	0	1	1	1	1	1	1	0
1	0	0	1	1	0	0	0	0
2	0	1	1	0	1	1	0	1
3	0	1	1	1	1	0	0	1
4	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1
6	0	1	0	1	1	1	1	1
7	0	1	1	1	0	0	0	0
8	0	1	1	1	1	1	1	1
9	0	1	1	1	1	0	1	1

20.4 Connection

Schematic



Wiring diagram



Below you can see the seven-segment display 74HC595 pin correspondance table:

74HC595 pin	Seven shows remarkable control pin (stroke)
Q0	7 (A)
Q1	6 (B)
Q2	4 (C)
Q3	2 (D)
Q4	1 (E)
Q5	9 (F)
Q6	10 (G)
Q7	5 (DP)

Step one: Connect the 74HC595

VCC (pin 16) and MR (pin 10) connected to 5V
GND (pin 8) and OE (pin 13) to ground
DS (pin 14) to Arduino board pin 2
ST_CP (pin 12, latch pin) to Arduino board pin 3
SH_CP (pin 11, clock pin) to Arduino board pin 4

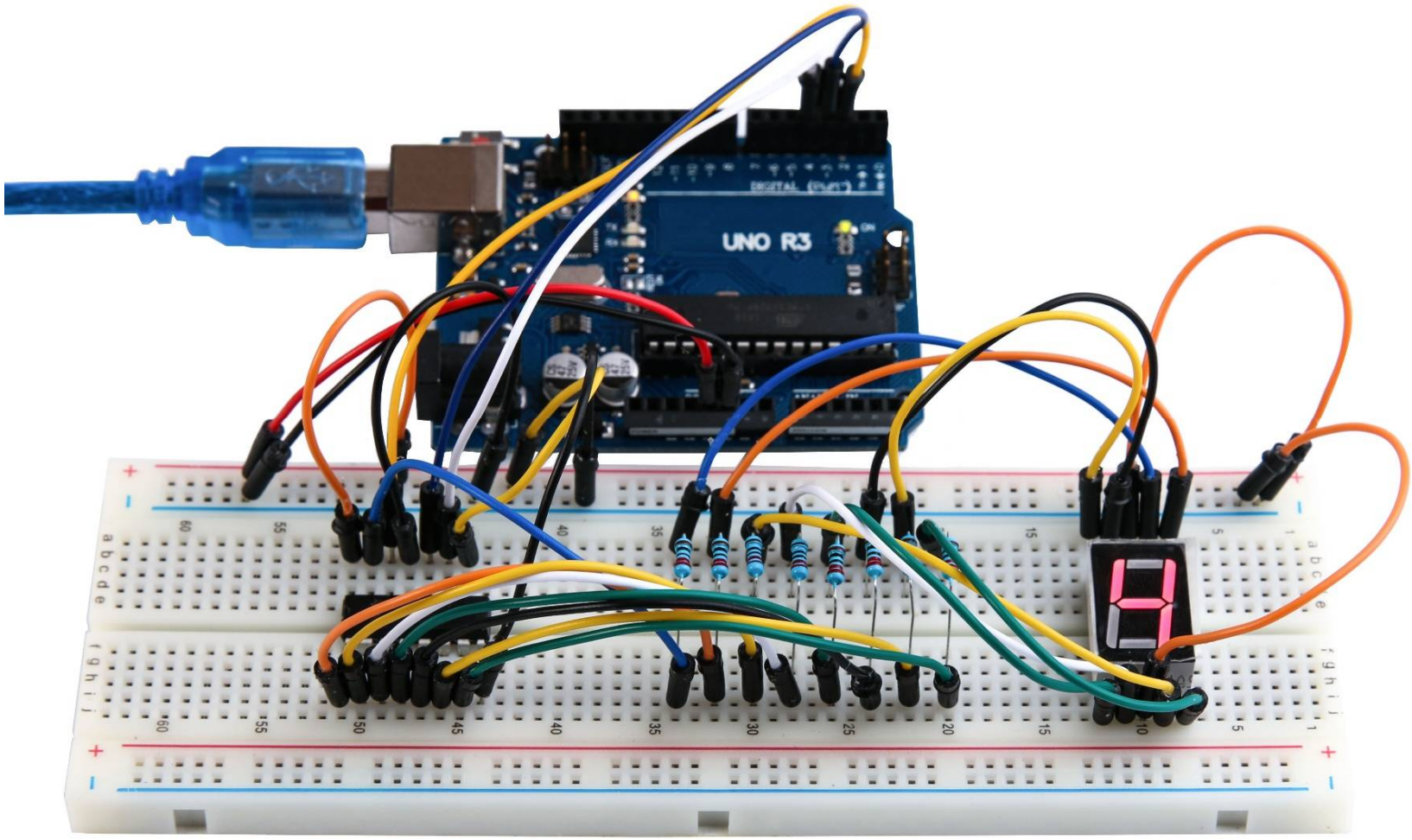
Step two: Connect the display

The seven-segment display 3, 8 pins to the Arduino board GND for common anode otherwise to the Arduino board + 5V. As the table above states, link the 74HC595 Q0 ~ Q7 to the display's corresponding pin (A ~ G and DP), followed by a 220 ohm resistor in series.

20.5 Code

After wiring, please open the program located in the folder - Lesson 27 74HC595, and press UPLOAD to upload the code. If there are any errors, see Lesson 2 for details about program uploading.

20.6 Example picture



21. Lesson 20 Four Digits Seven Segments Display

21.1 Overview

This lesson will be teaching you the basics on using a 4-digit 7-segment display. For this particular module, please pay attention to whether it is common anode or common cathode. For the latter, the corresponding pin connects to ground, while in the first case, the corresponding pin connects to the power source.

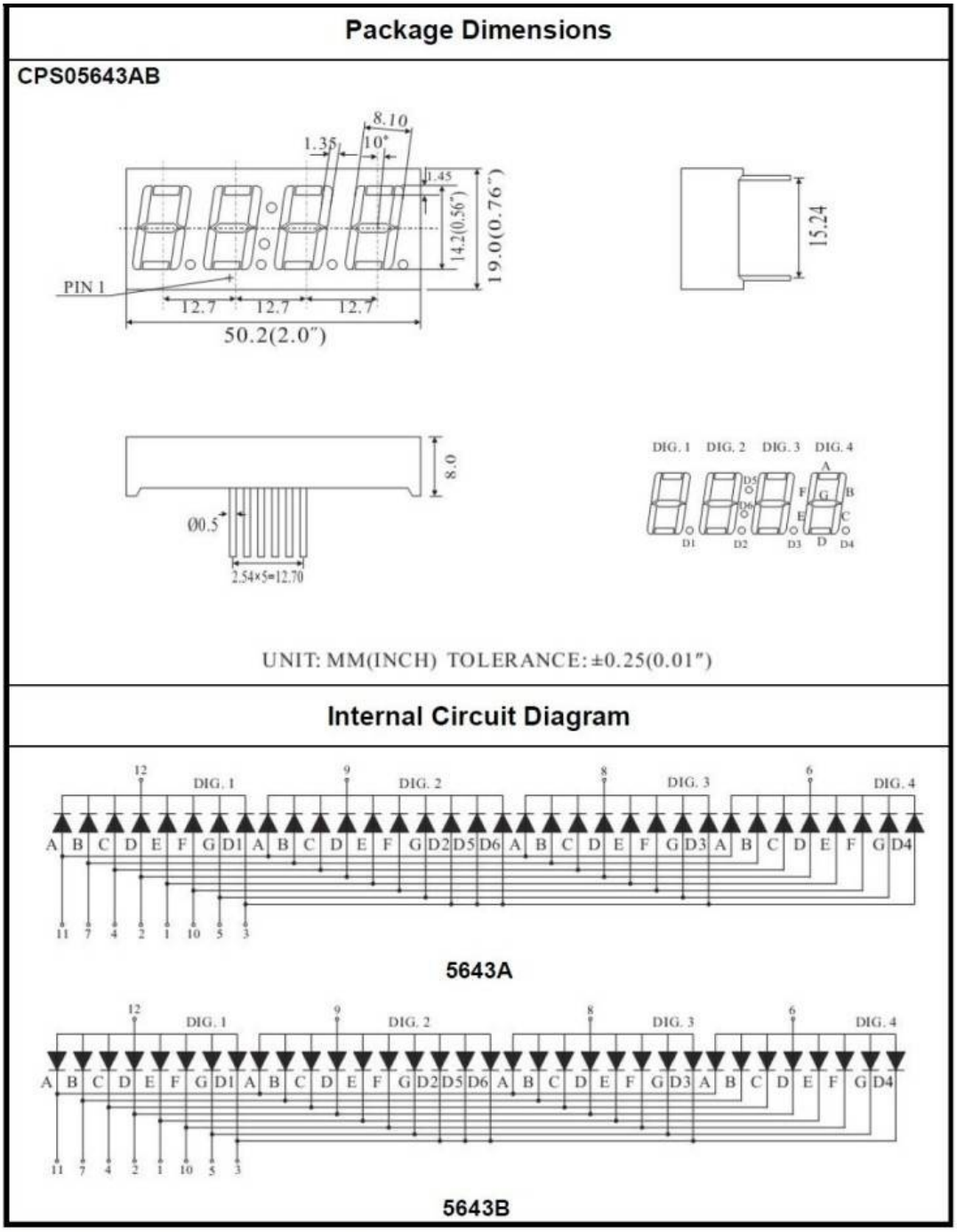
For this display, the common anode or common cathode pin controls which digit is shown. Although just one digit is working at a time, you are capable of seeing all the numbers displayed because the flashing is so quick that you barely see the interruptions, according to the principle of Persistence of Vision.

21.2 Components Required

(1) x Plusivo Uno R3
(1) x 830 tie-points breadboard
(1) x 74HC595 IC
(1) x 4 Digit 7-Segment Display
(4) x 220 ohm resistors
(23) x M-M wires (Male to Male jumper wires)

21.3 Component Introduction

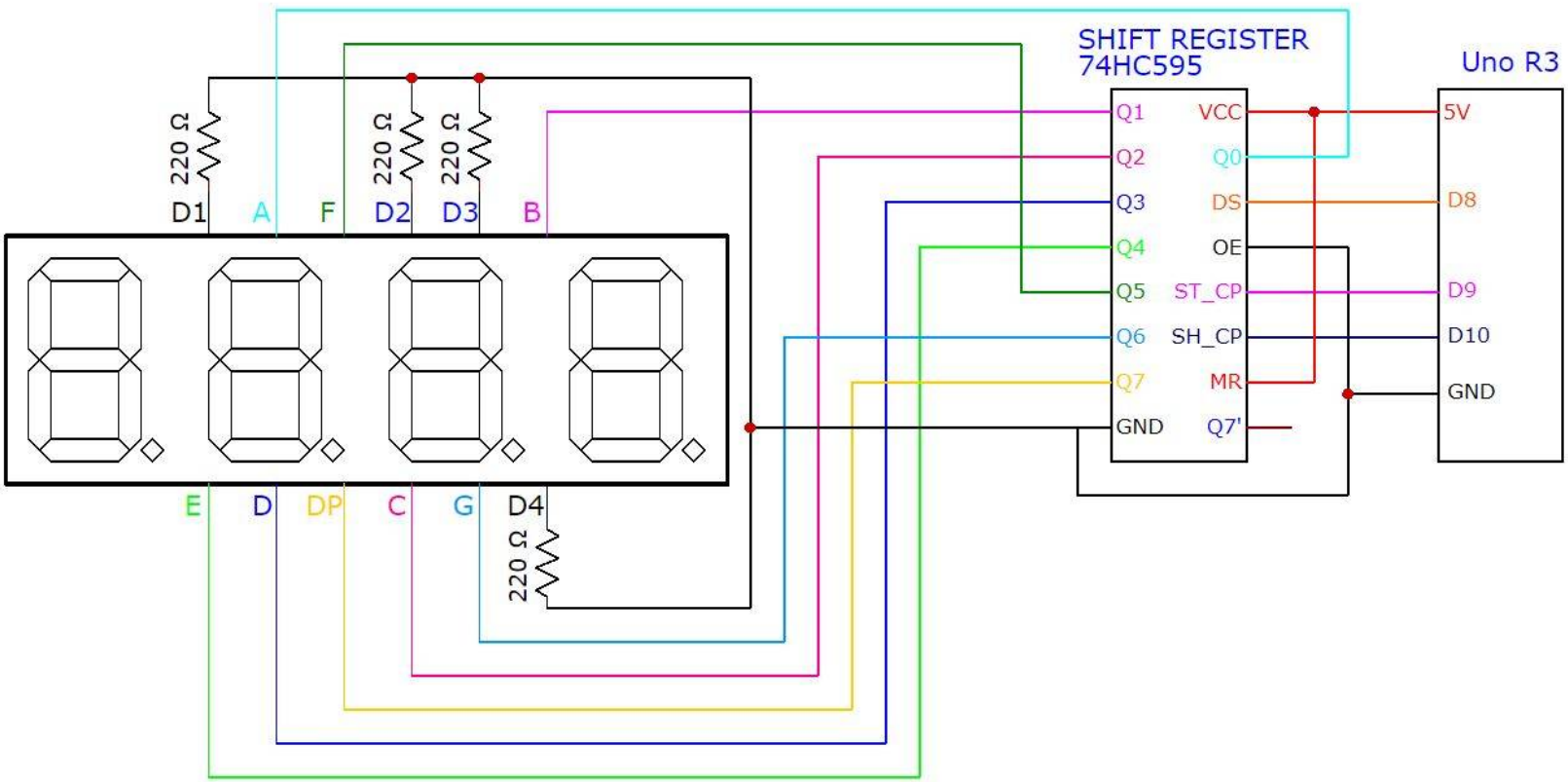
Four Digits Seven segments display



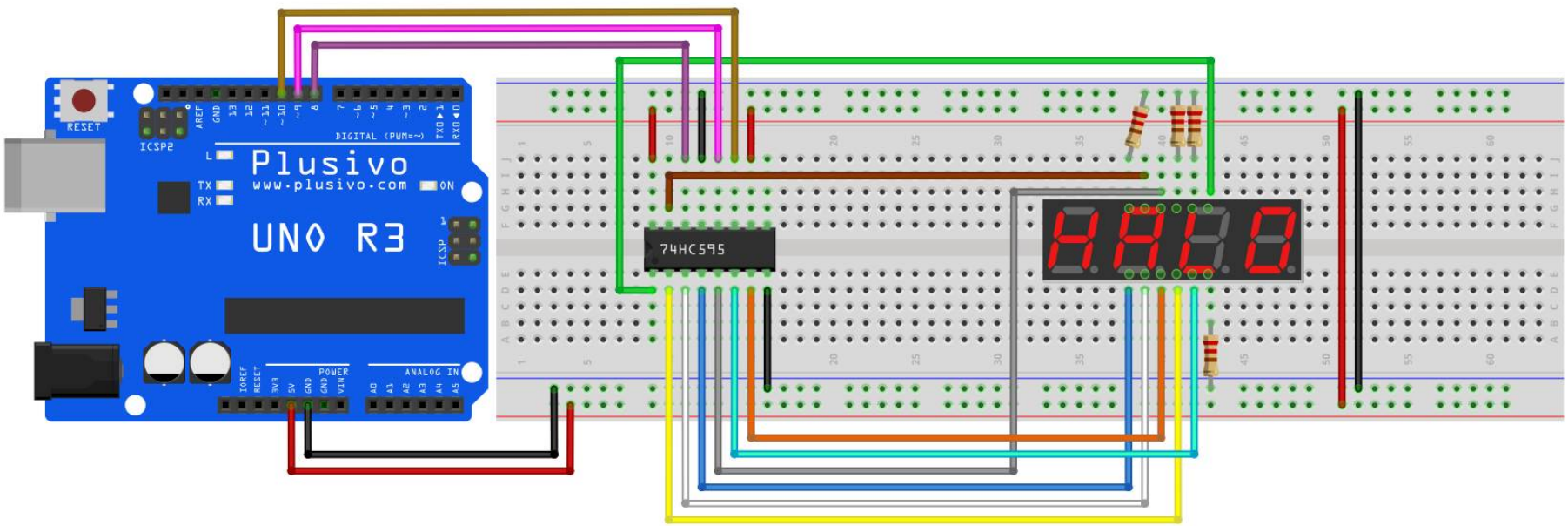
Four Digits Displays Series

21.4 Connection

Schematic



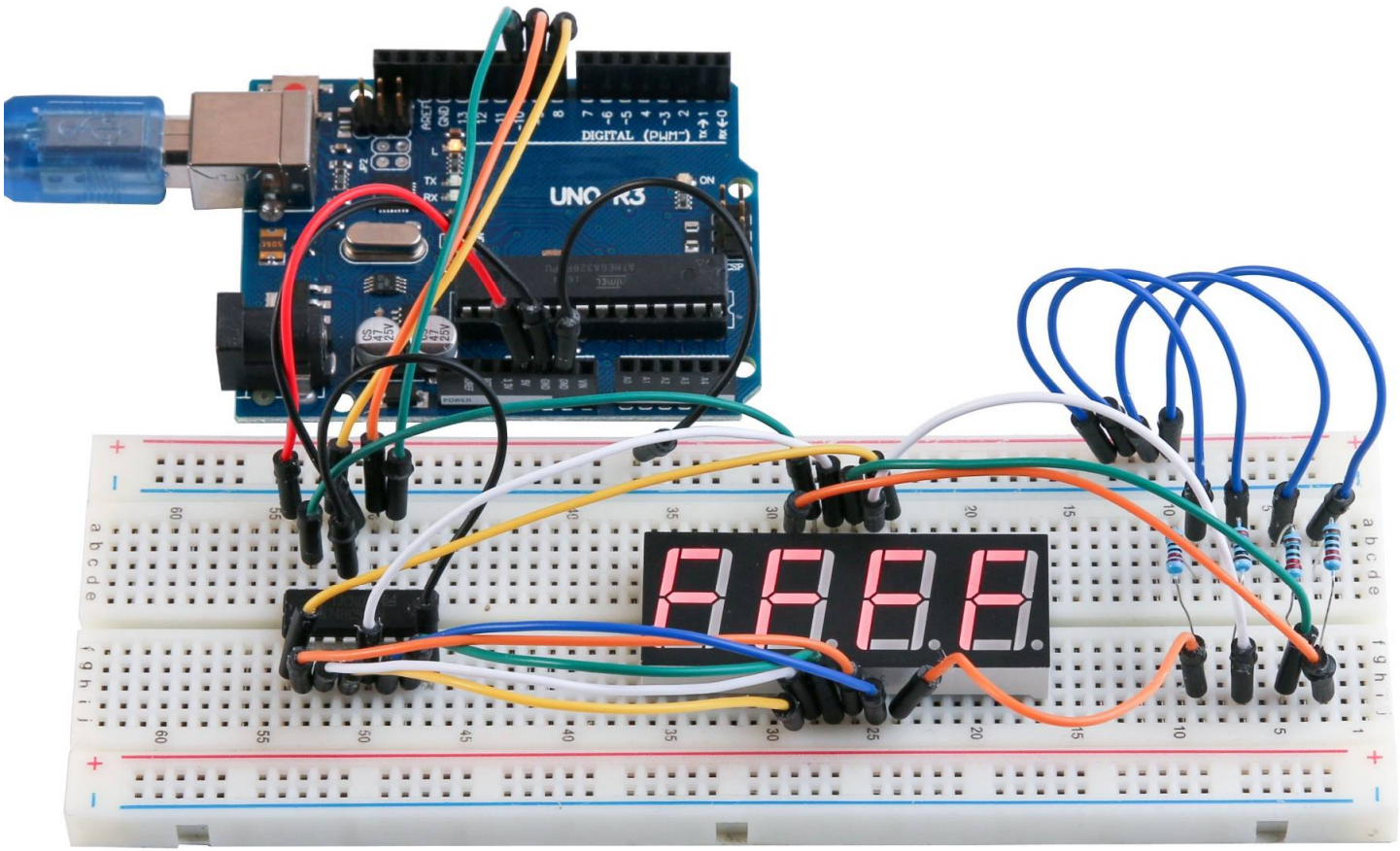
Wiring diagram



21.5 Code

After wiring, please open the program located in the folder - Lesson 28 Four Digits Seven Segments Display and press UPLOAD to upload the code. If there are any errors, see Lesson 2 for details about program uploading.

21.6 Example picture



22. Lesson 21 DC Motors

22.1 Overview

This lesson will be teaching you the basics on how to control a small DC motor using a transistor and an Arduino.

22.2 Components Required

(1) x Plusivo 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 9V 1A adapter

22.3 Component Introduction

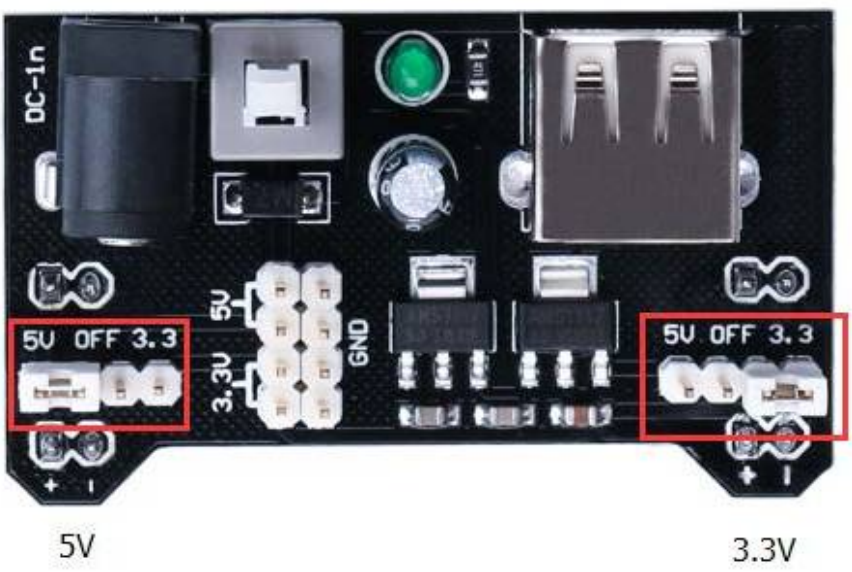
Breadboard Power Supply

The small DC motor is prone to using more power than an Arduino board digital output can manage directly, so attempting to link the motor straight to it has a chance to damage it. To avoid this situation, we simply utilize a power supply module.

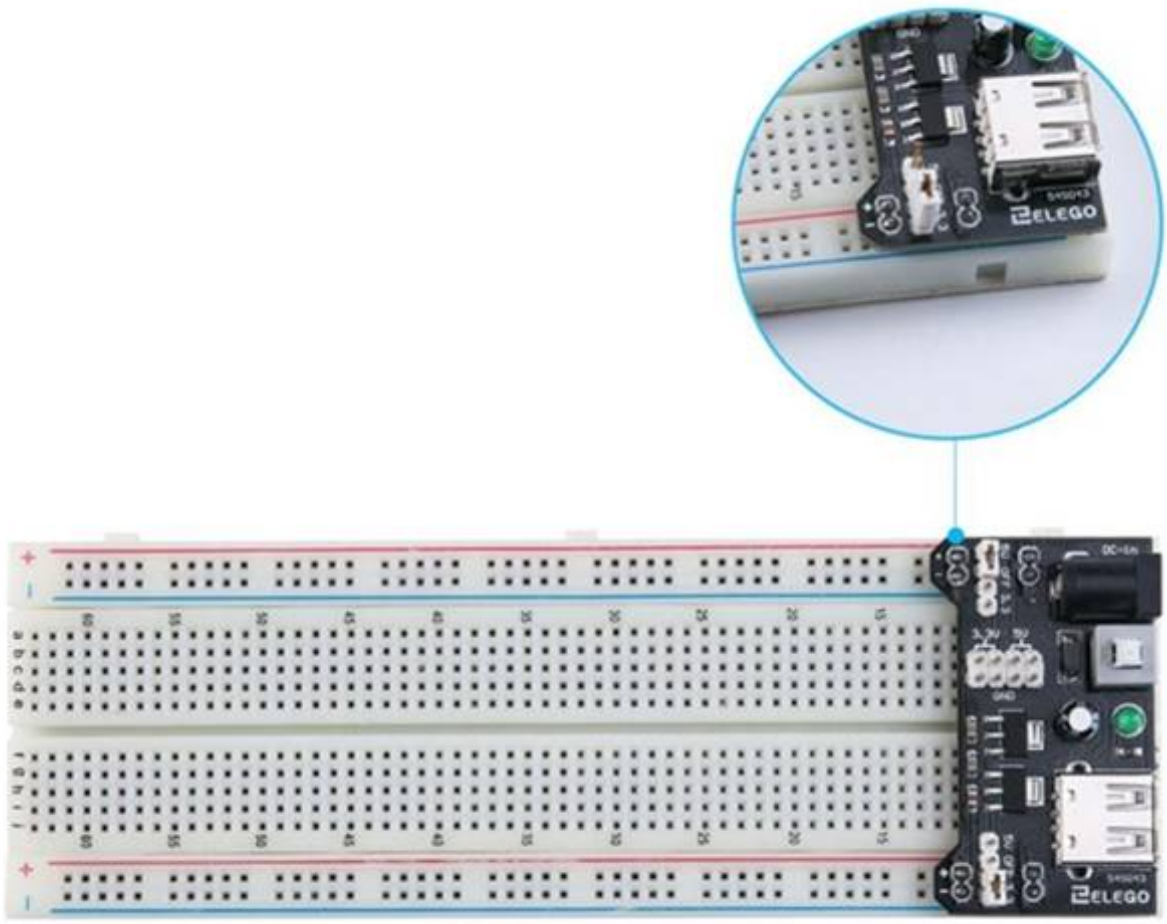
Product Specifications:

- 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

Preparing the output voltage:



Both the voltage outputs can be set up individually. To choose the output voltage, move the jumper cable to the matching pins. You should see that the power indicator LED and the breadboard power rails will not switch on if each of the jumpers are in the "OFF" state.



Important note

Be careful to align the module accurately on the breadboard. The negative pin(-) on the module matches up with the blue line(-) on breadboard, while the positive pin(+) matches up with the red line(+).

L293D

This is quite a handy chip, as it has the ability to control two motors individually. We are merely using half of it in this particular case. The pins on the right hand side of the chip are mostly for managing a second motor.

Product Specifications

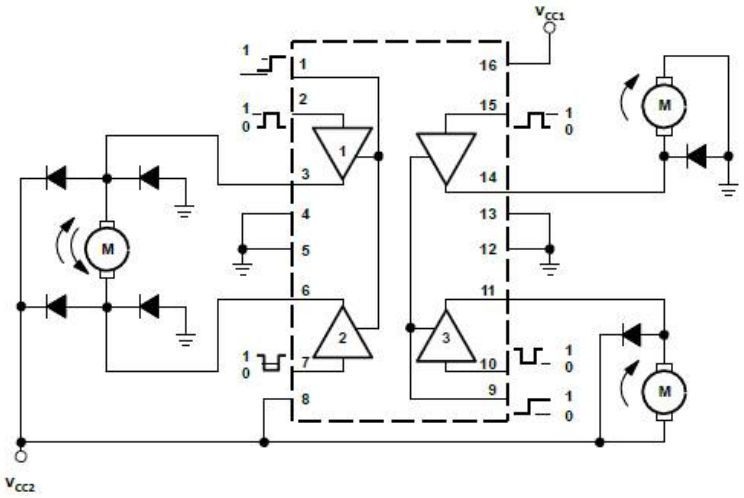
- Features Unitrode L293 and L293D Products From Texas Instruments
- Wide Supply-Voltage Range: 4.5 V to 36 V
- Separate Input-Logic Supply
- Internal ESD Protection
- Thermal Shutdown
- High-Noise-Immunity Inputs
- Functionally Similar to SGS L293 and SGS L293D
- Output Current 1 A Per Channel (600 mA for L293D)
- Peak Output Current 2 A Per Channel (1.2 A for L293D)
- Output Clamp Diodes for Inductive T ransient 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, while the L293D is designed for 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, and each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs: drivers 1 and 2 enabled by 1,2EN, 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 correspondance with their inputs. Alternatively, 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.

To use this pinout:

The left hand side deals with the first motor, while the right one manages a second motor. You can also run it with a single motor connected.

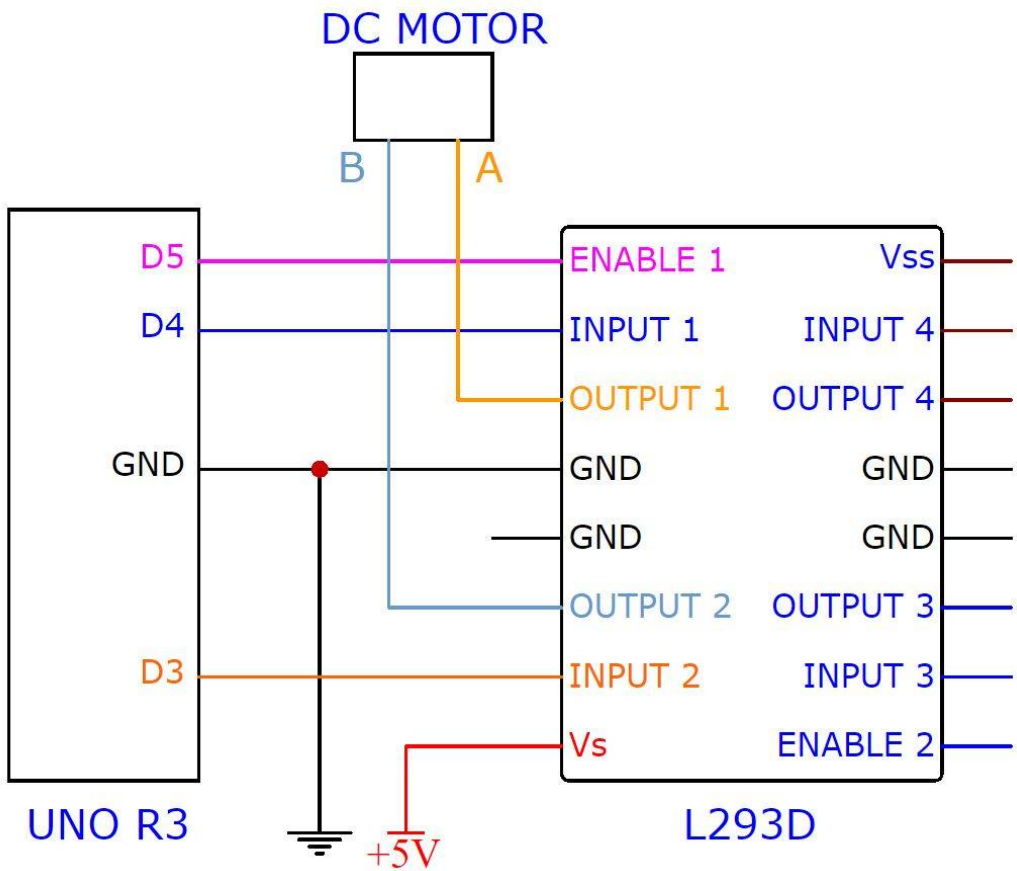
Arduino Connections

M1 PWM - connect this to a PWM pin on the Arduino. Output any integer between 0 and 255, where 0 will be off, 128 is half speed and 255 is max speed.

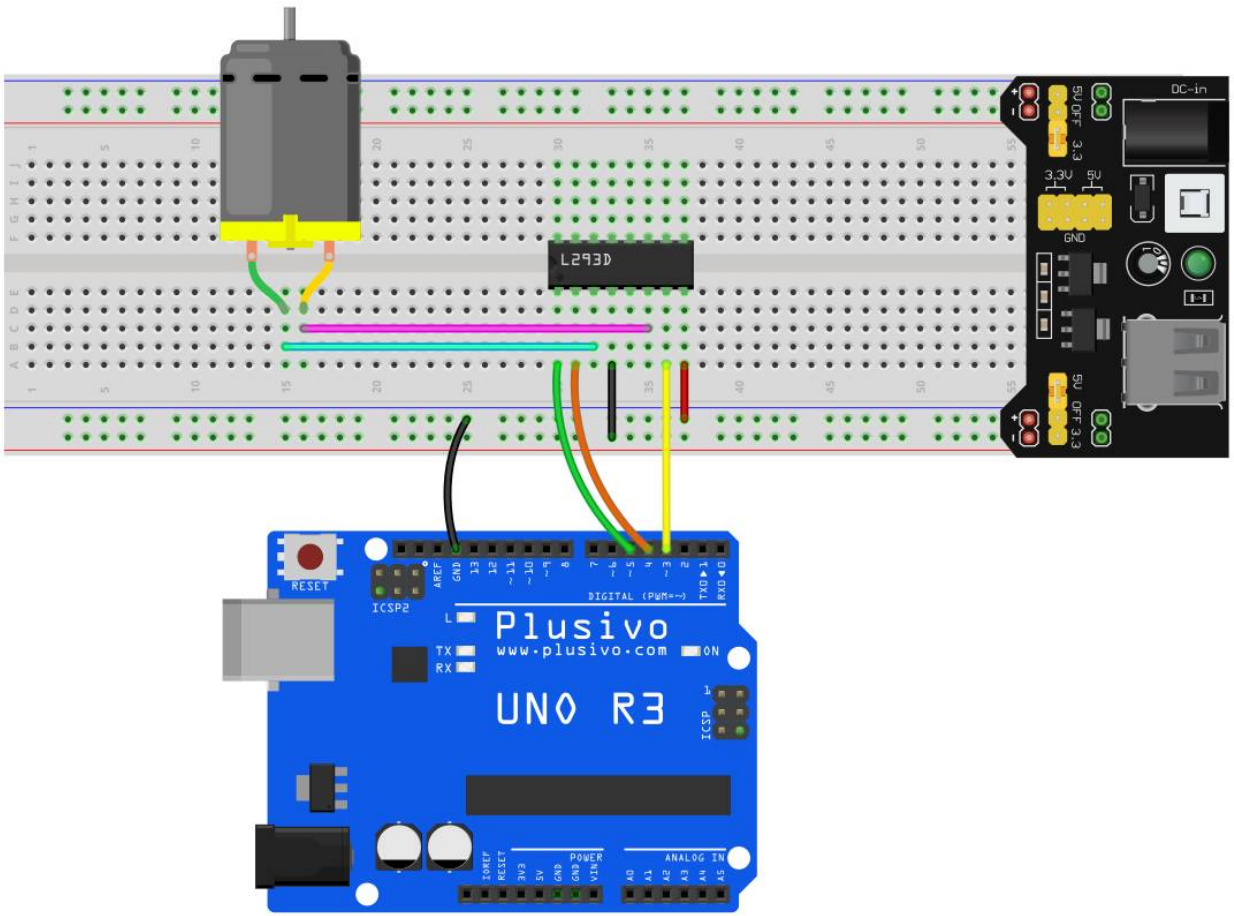
M1 direction 0/1 and M1 direction 1/0 - Connect these two to two digital Arduino pins. Output one pin as HIGH and the other pin as LOW, and the motor will spin in one direction. If you reverse them, the motor will spin in the other direction.

22.4 Connection

Schematic

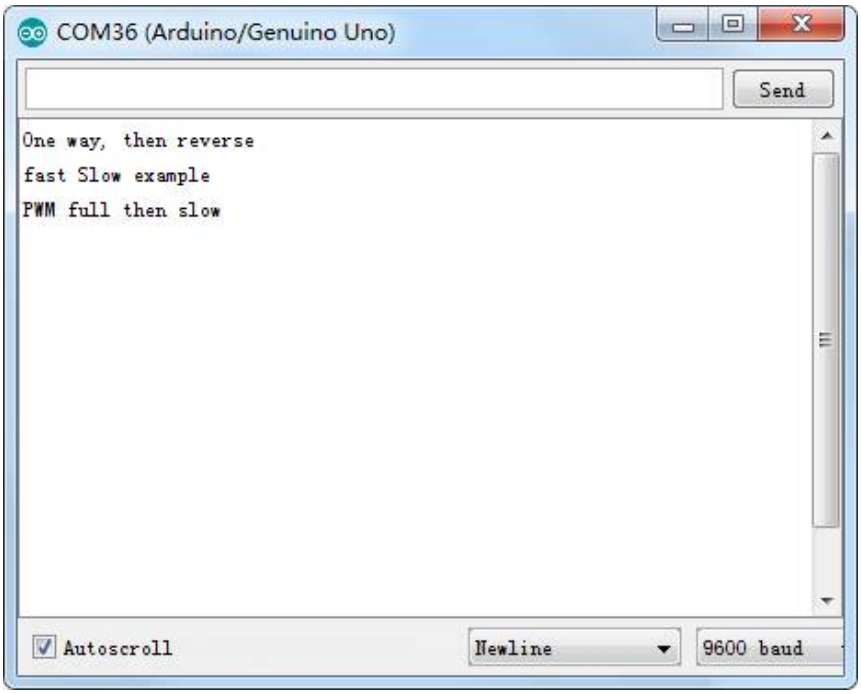


Wiring diagram



The code underneath does not require a separate power supply, it actually uses the 5V power from the Arduino. This can only be done because the L293D is controlling it.

It is not advised to connect a motor directly to the device, as you get an electrical feedback when you turn a motor off. If you have a small motor, this will damage your UNO, but with a large one, you can even see a fire emerge.

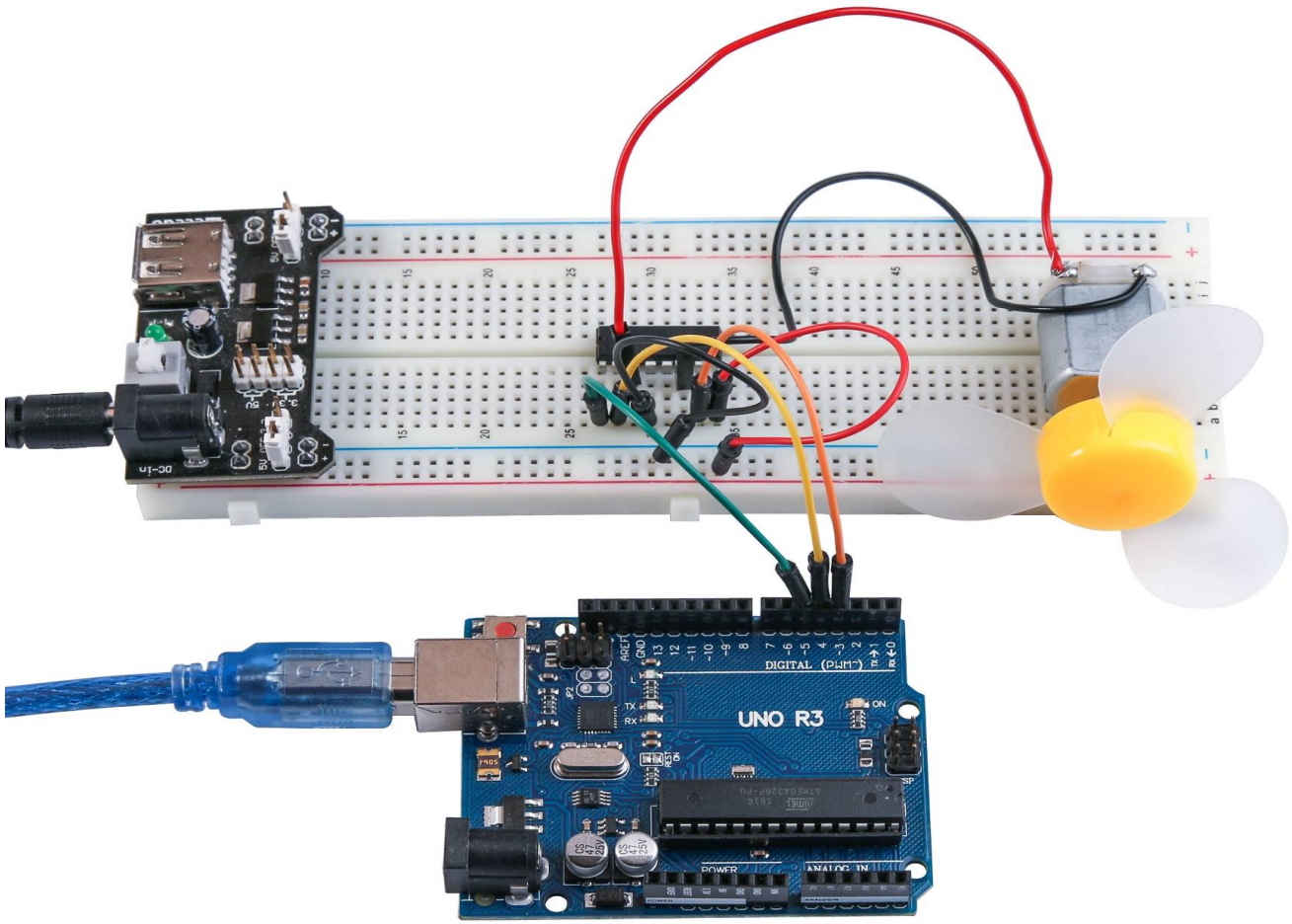


22.5 Code

After wiring, please open the program located in the folder - Lesson 29 DC Motors and press UPLOAD to upload the code. If there are any errors, see Lesson 2 for details about program uploading.

After the program loads, turn on all the power switches. The motor will slightly rotate clockwise and counterclockwise for 5 times. Then, it will continue to dramatically rotate clockwise and counterclockwise. Afterwards, the controller board will send a PWM signal to drive the motor, so the motor will gradually lower its maximum RPM to the minimum and increase to the maximum again. Finally, it stops for 10 seconds until the next cycle starts.

22.6 Example picture



23. Lesson 22 Relay

23.1 Overview

In this lesson, you will learn the basics on using a relay.

23.2 Components Required

(1) x Plusivo Uno R3
(1) x 830 tie-points breadboard
(1) x Fan blade and 3-6v dc motor
(1) x L293D IC
(1) x 5v Relay
(1) x Power Supply Module
(1) x 9V1A Adapter
(8) x M-M wires (Male to Male jumper wires)

23.3 Component Introduction

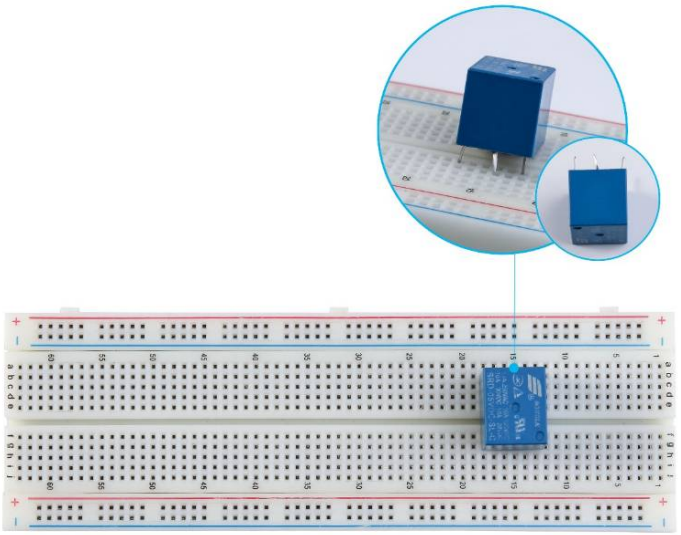
Relay

A relay is an electrically operated switch, generally using an electromagnet to mechanically operate a switch, but sometimes its operating principles are also used as in solid-state relays. Relays are used where it is necessary to control a circuit using a low-power signal (with complete electrical isolation between control and controlled circuits), or where several circuits must be controlled by one signal. The first relays were used in long-distance telegraph circuits as amplifiers, and as early computers to perform logical operations further down the line. The way they worked is that they repeated the signal coming in from one circuit and re-transmitted it on another circuit.

A contactor is a type of relay that can handle the high power needed to directly control an electric motor or other loads. Solid-state relays control power circuits using a semiconductor device to perform the switching. Relays with calibrated operating characteristics and sometimes multiple operating coils are utilized to defend electrical circuits from overload or faults. In modern electric power systems, these functions are performed by digital instruments called "protective relays".

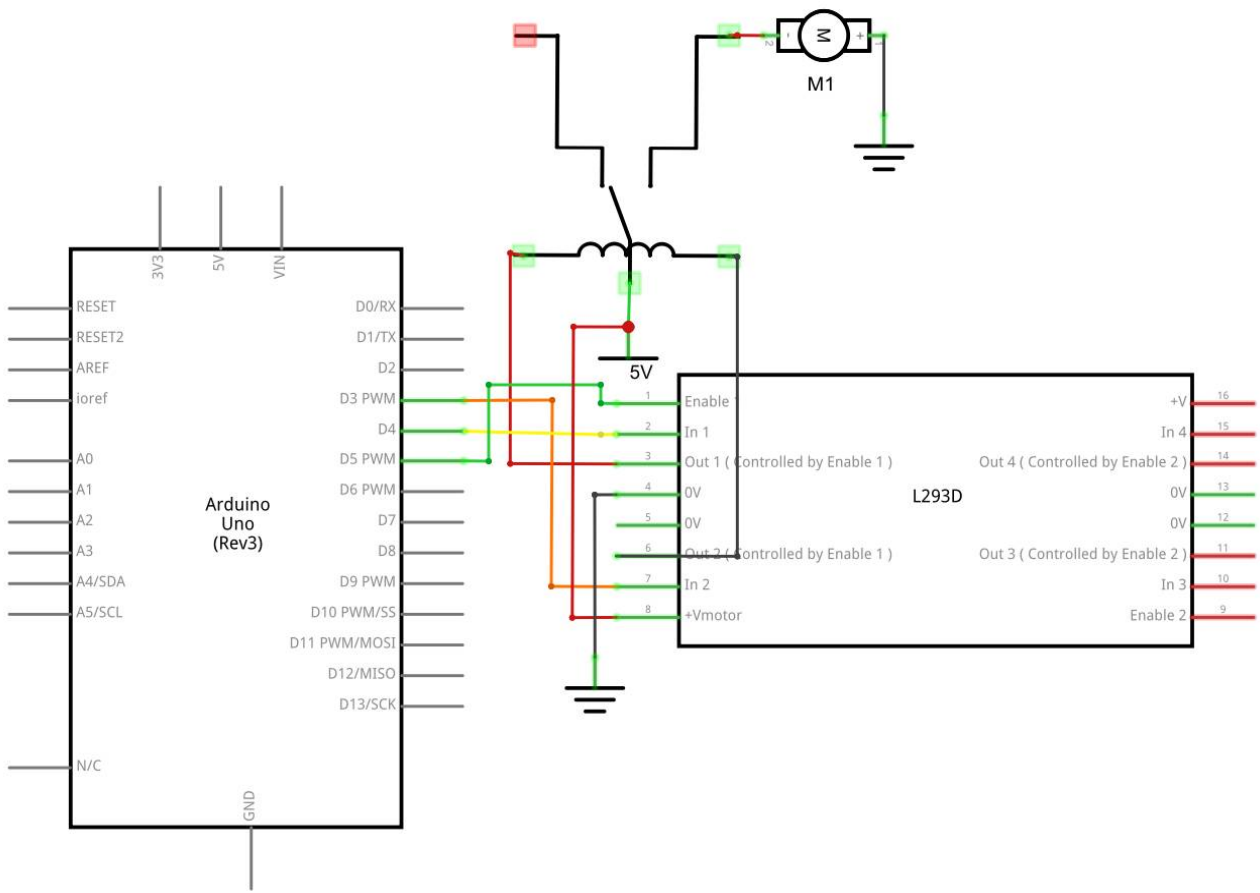
You can find the schematic showing how to drive a relay with our device below.

If you are unsure about how to insert the relay into the breadboard. As you can see in the photo below, you will need to slightly bend one of the pins of the relay so you can insert it into the breadboard.

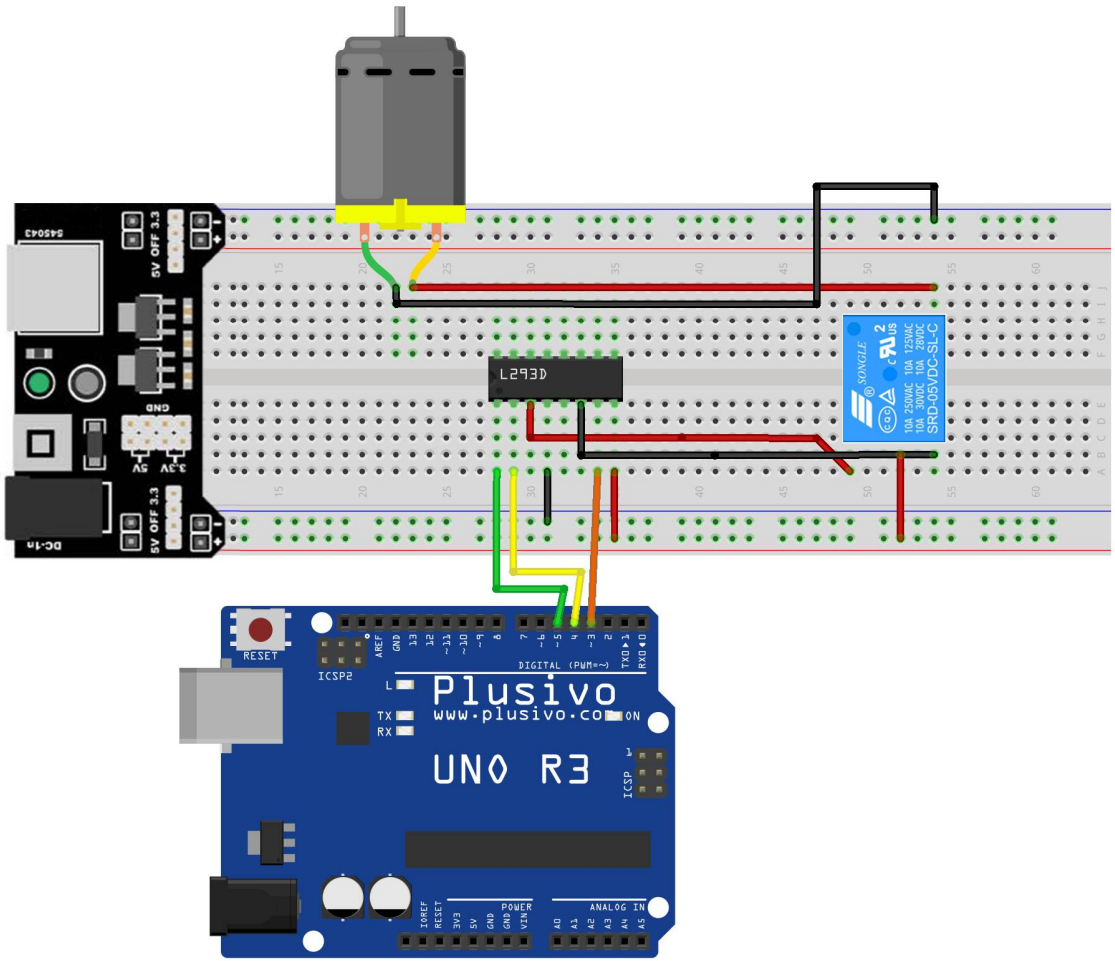


23.4 Connection

Schematic



Wiring diagram

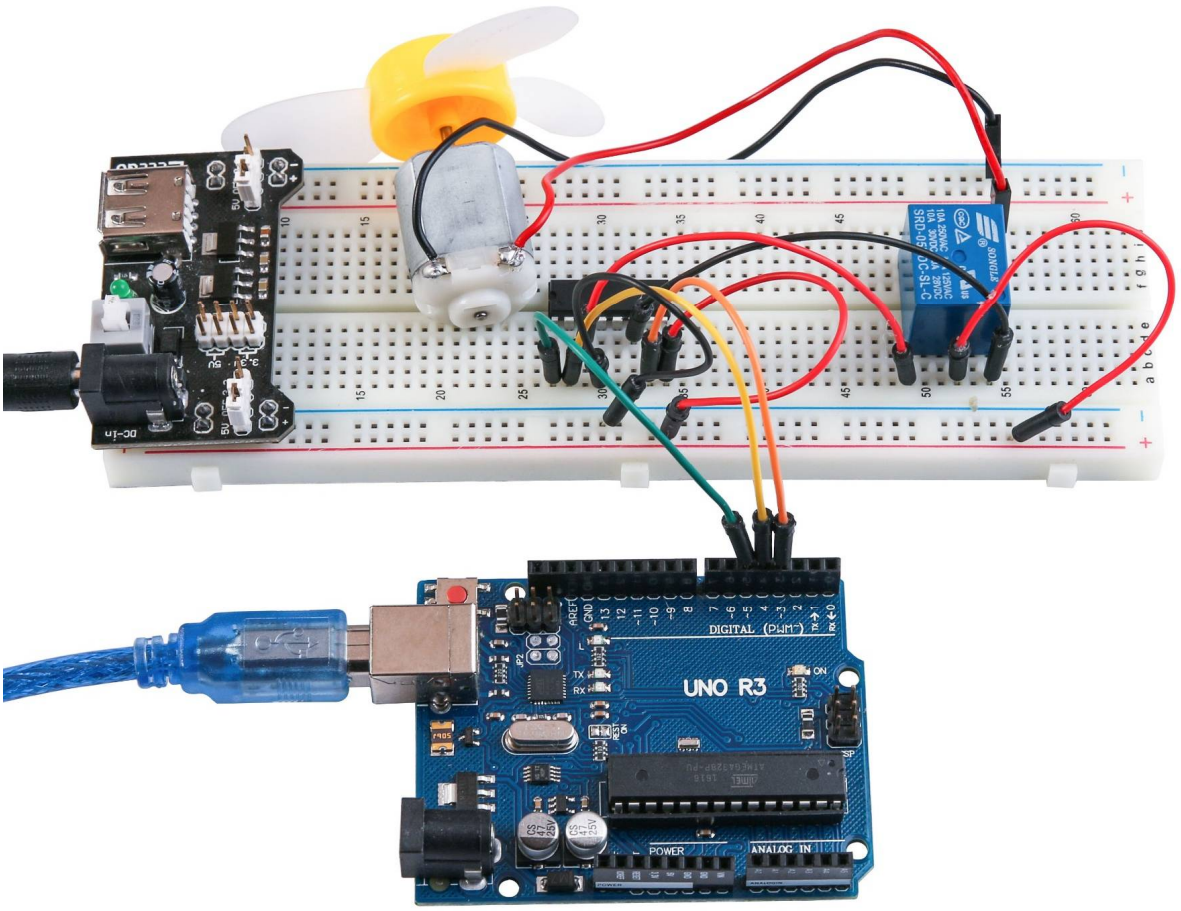


23.5 Code

After wiring, please open the program located in the folder - Lesson 30 Relay and press UPLOAD to upload the code. If there are any errors, see Lesson 2 for details about program uploading.

After the program loads, turn on all the power switches. The relay will pick up with a ringing sound. Then, the motor will rotate. Eventually, the relay will be released, and the motor stops.

23.6 Example picture



24. Lesson 23 Stepper Motor

24.1 Overview

This lesson will teach you a witty and simple method of driving which, in this case, comes with its own driver board so it's convenient to connect to our Arduino.

24.2 Components Required

(1) x Plusivo Uno R3
(1) x 830 tie-points breadboard
(1) x ULN2003 stepper motor driver module
(1) x Stepper motor
(1) x 9V1A Adapter
(1) x Power supply module
(6) x F-M wires (Female to Male DuPont wires)
(1) x M-M wire (Male to Male jumper wire)

24.3 Component Introduction

Stepper Motor

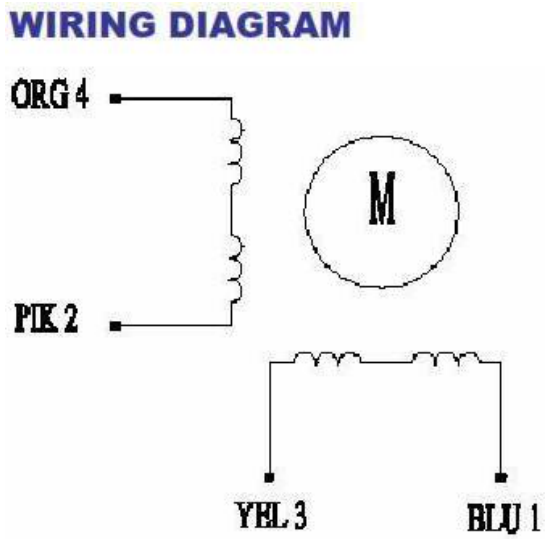
A stepper motor is an electromechanical device which converts electrical pulses into discrete mechanical movements. The shaft or spindle of a stepper motor rotates in discrete step increments when electrical command pulses are applied to it in the proper sequence. The motors rotation has several direct relationships to these applied input pulses. The sequence of the applied pulses

is directly related to the direction of motor shafts rotation. The speed of the motor shafts rotation is directly related to the frequency of the input pulses and the length of rotation is directly related to the number of input pulses applied. One of the most significant advantages of a stepper motor is its ability to be accurately controlled in an open loop system, which means no feedback information about position is needed. This type of control eliminates the need for expensive sensing and feedback devices such as optical encoders, you position being known simply by keeping track of the input step pulses.

Stepper motor 28BYJ-48 Parameters

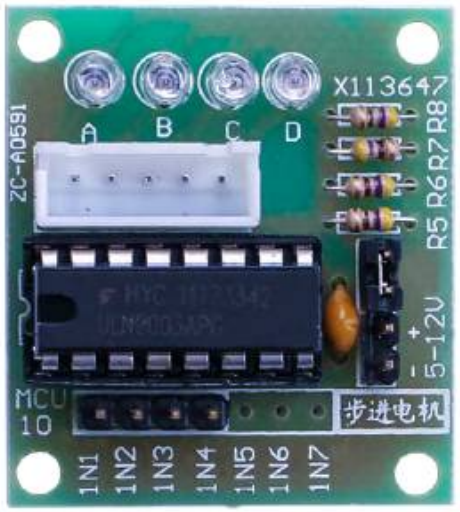
- Model: 28BYJ-48
- Rated voltage: 5VDC
- Number of Phase: 4
- Speed Variation Ratio: 1/64
- Stride Angle: 5.625° /64
- Frequency: 100Hz
- DC resistance: 50Ω±7%(25°C)
- Idle In-traction Frequency: > 600Hz
- Idle Out-traction Frequency: > 1000Hz
- In-traction Torque >34.3mN.m(120Hz)
- Self-positioning Torque >34.3mN.m
- Friction torque: 600-1200 gf.cm
- Pull in torque: 300 gf.cm
- Insulated resistance >10MΩ(500V)
- Insulated electricity power：600VAC/1mA/1s
- Insulation grade：A
- Rise in Temperature <40K(120Hz)
- Noise <35dB(120Hz,No load,10cm)

Interfacing circuits



You can commonly see four wires coming out of a bipolar stepper, which does not contain a common cener connection, but it has two separate sets of coils, dissimilar to unipolar steppers. These types of steppers can be differentiated by measuring the resistance between the wires and identifying two pairs of wires with equal resistance. You should notice infinite resistance if you've got the leads of your meter linked to two wires that are not connected.

ULN2003 Driver Board



Product Description

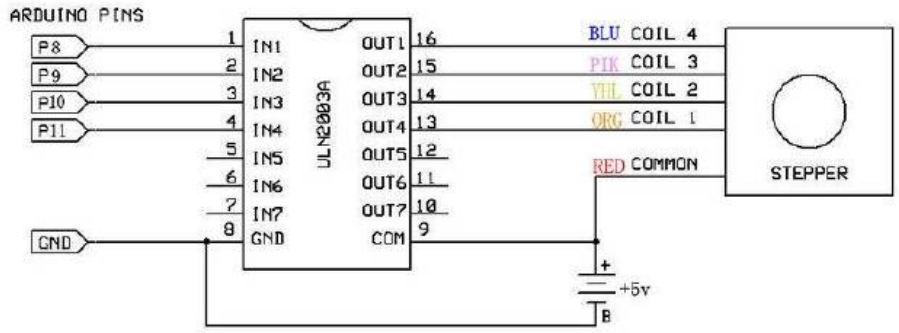
- Size: 42mmx30mm
- Use ULN2003 driver chip, 500mA
- A. B. C. D LED indicating the four phase stepper motor working condition
- White jack is the four phase stepper motor standard jack
- Power pins are separated
- We kept the rest pins of the ULN2003 chip for your further prototyping

The most basic method to interface a unipolar stepper to the UNO is to use a breakout for ULN2003A transistor array chip. This chip consists of seven Darlington transistor drivers, this it is similar to having seven TIP120 transistors all in one package. The ULN2003A has a capacity of up to 500 mA per channel and an internal voltage drop of approximately 1V when on. Moreover, it contains internal clamp diodes to disperse voltage spikes when driving inductive loads. We apply voltage to each of the threads in a specific arrangement to manage the motor.

The sequence would go like this:

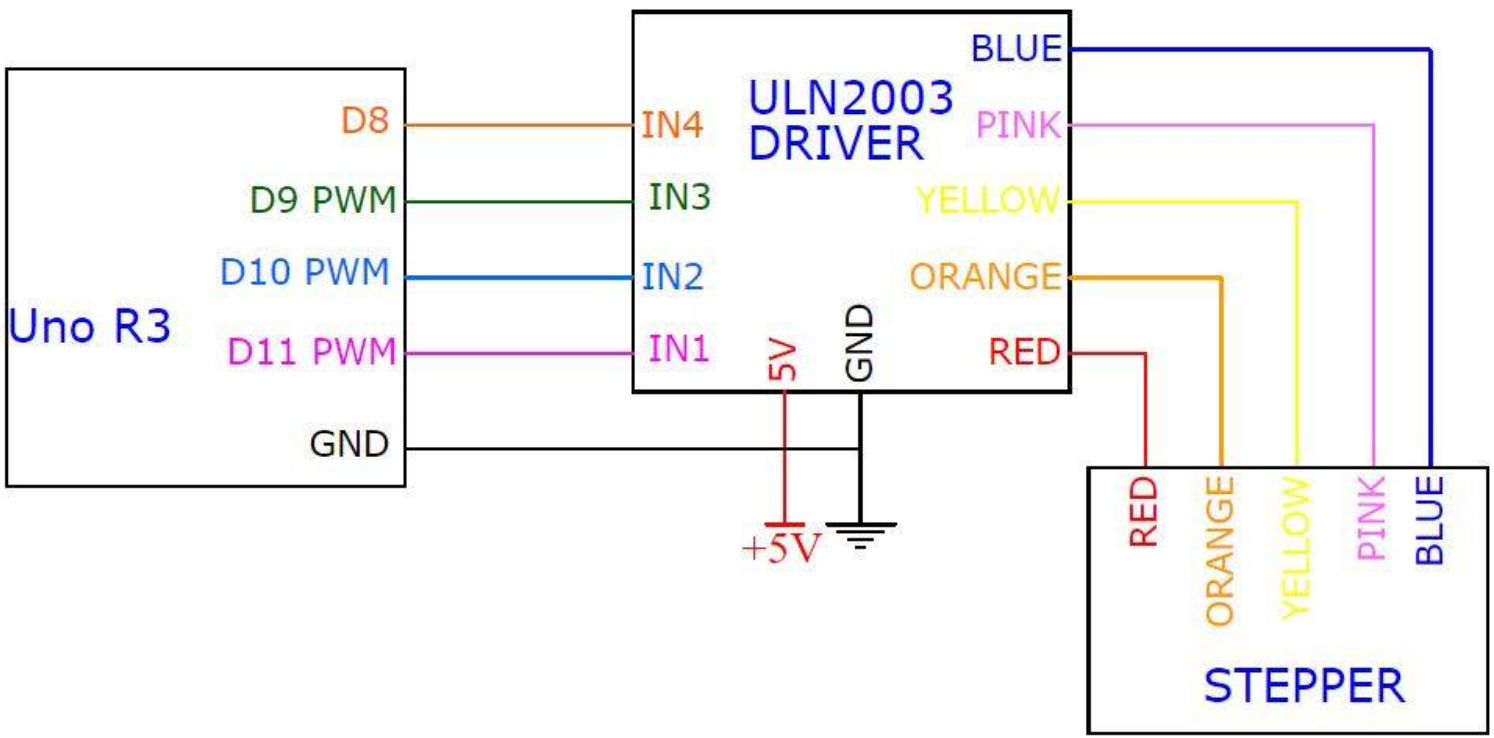
Lead Wire Color	---> CW Direction (1-2 Phase)							
	1	2	3	4	5	6	7	8
4 ORG	-	-						-
3 YEL		-	-	-				
2 PIK				-	-	-		
1 BLU						-	-	-

These pictures display the process of interfacing a unipolar stepper motor to four controller pins with a ULN2003A:

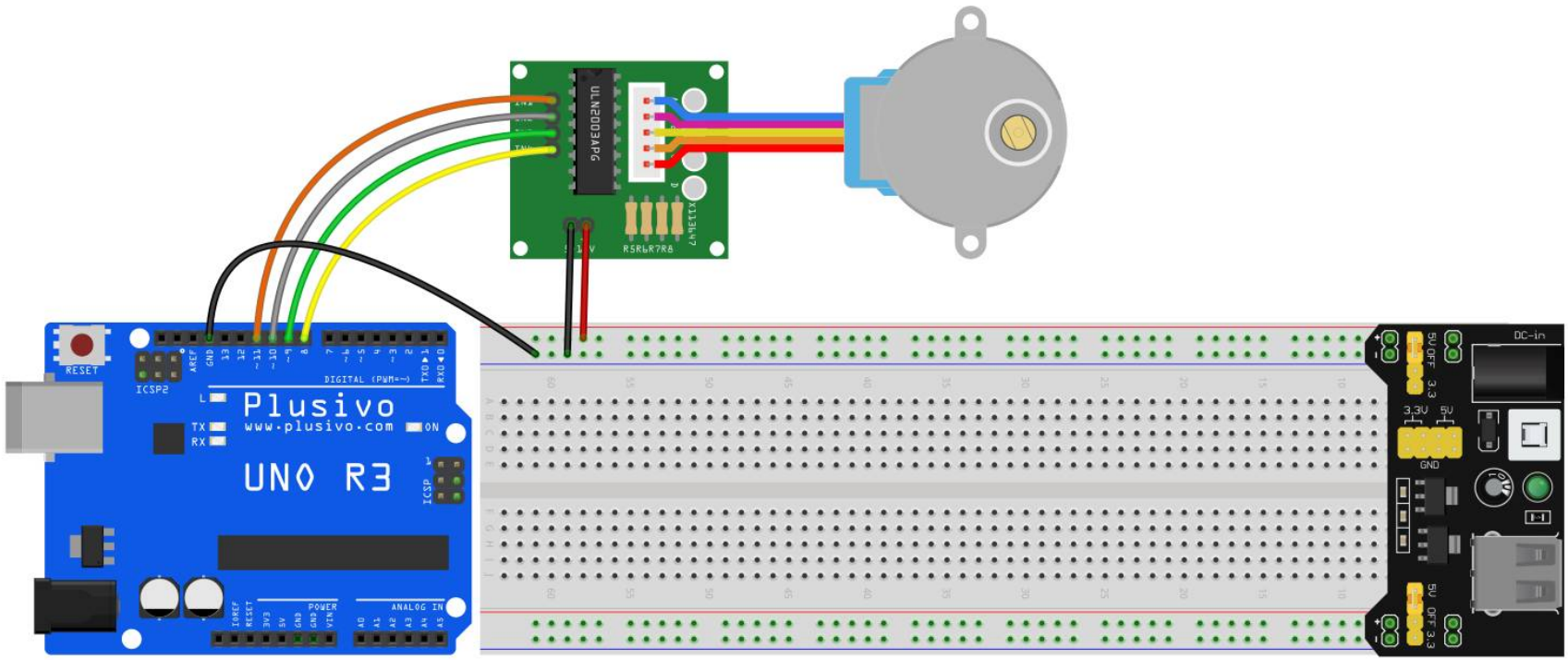


24.4 Connection

Schematic



Wiring diagram

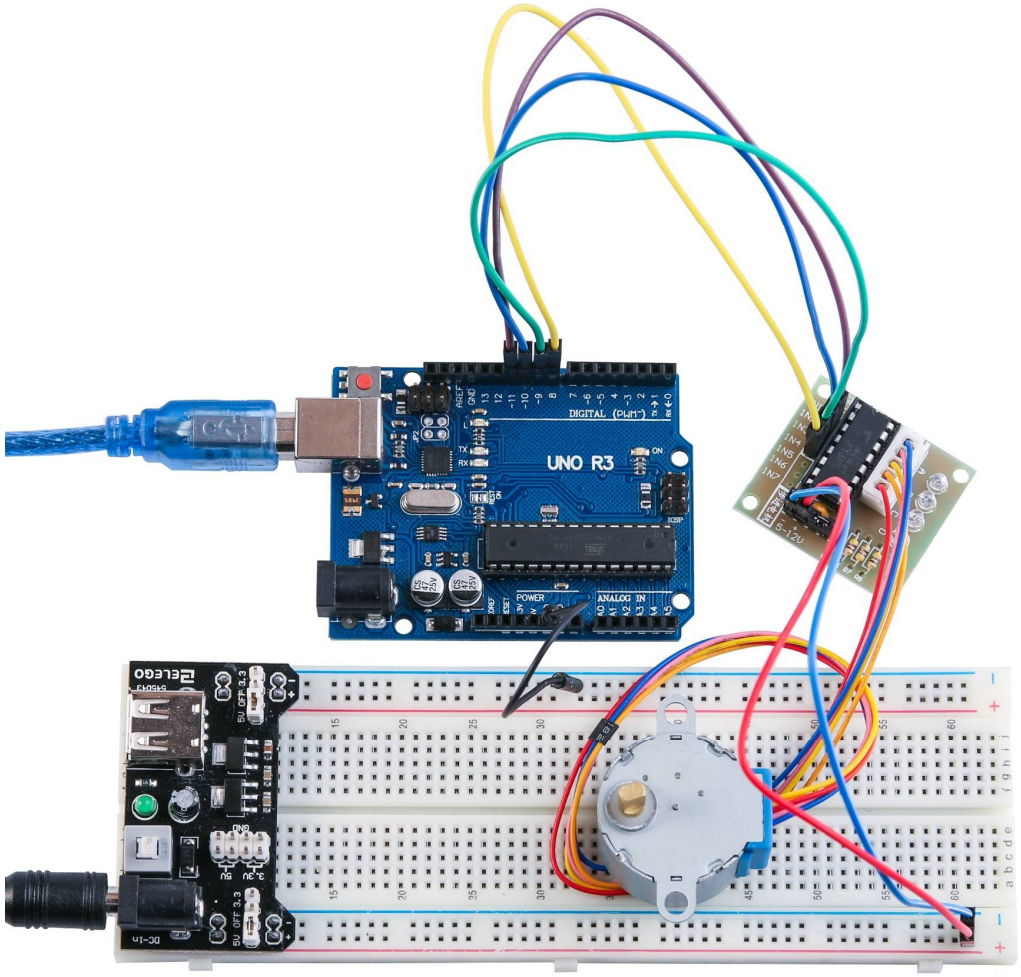


We are using 4 pins (8-11) to control the Stepper and we connect the Ground from our microcontroller to the Stepper motor.

24.5 Code

After wiring, please open the program located in the folder - Lesson 31 Stepper Motor and press UPLOAD to upload the code. If there are any errors, see Lesson 2 for details about program uploading. Before you can run this, make sure that you have installed the < Stepper > library or re-install it, if necessary. Otherwise, your code won't work. For details about loading the library file, see Lesson 1.

24.6 Example picture



25. Lesson 24 Controlling Stepper Motor With Remote

25.1 Overview

In this lesson, you will learn a simple method to control a stepper motor from a greater length by using an IR remote control.

The motor we are working with comes with its own driver board, so it is convenient to connect.

We will be using a reasonable breadboard power supply that plugs directly into our breadboard and powers it with 9V 1A, because we don't want to power the motor directly from the UNO.

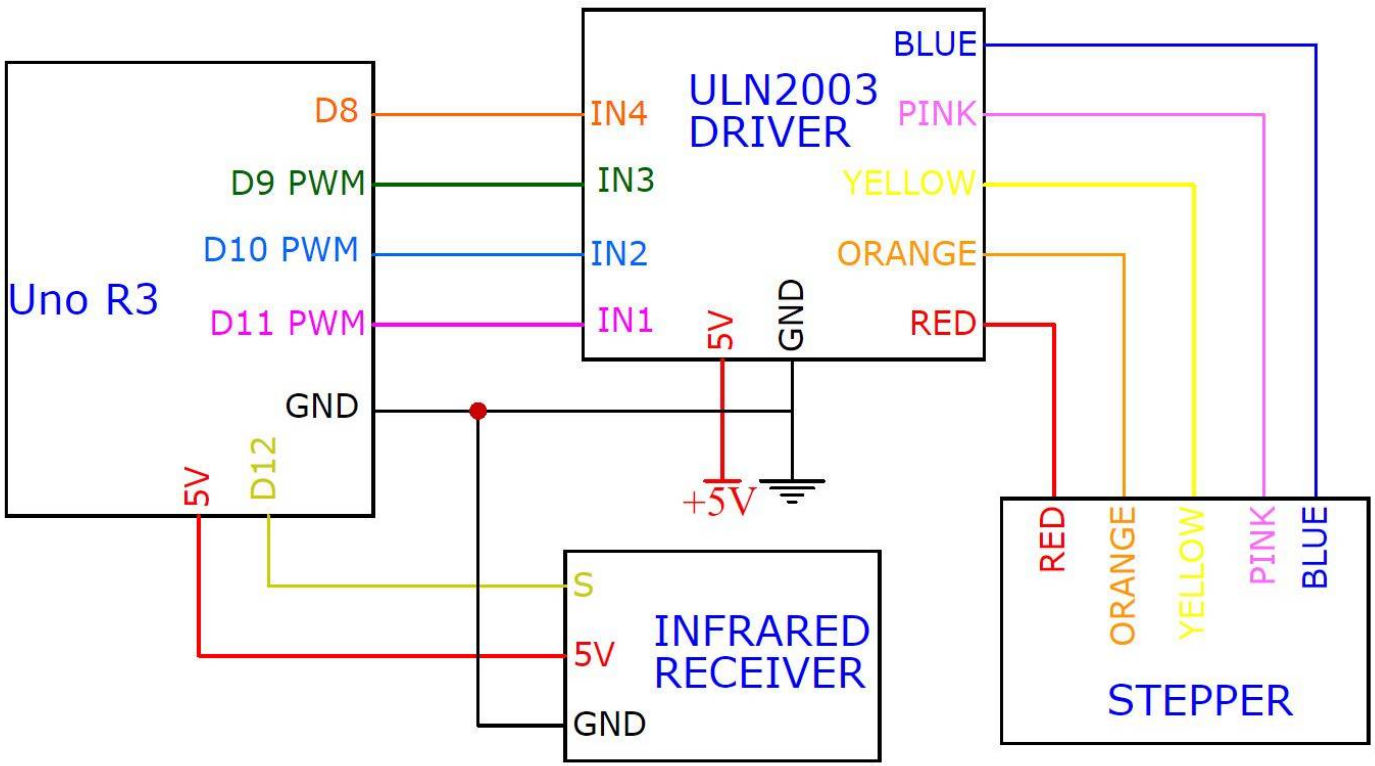
The IR sensor is directly connected to the device as it uses almost no power.

25.2 Components Required

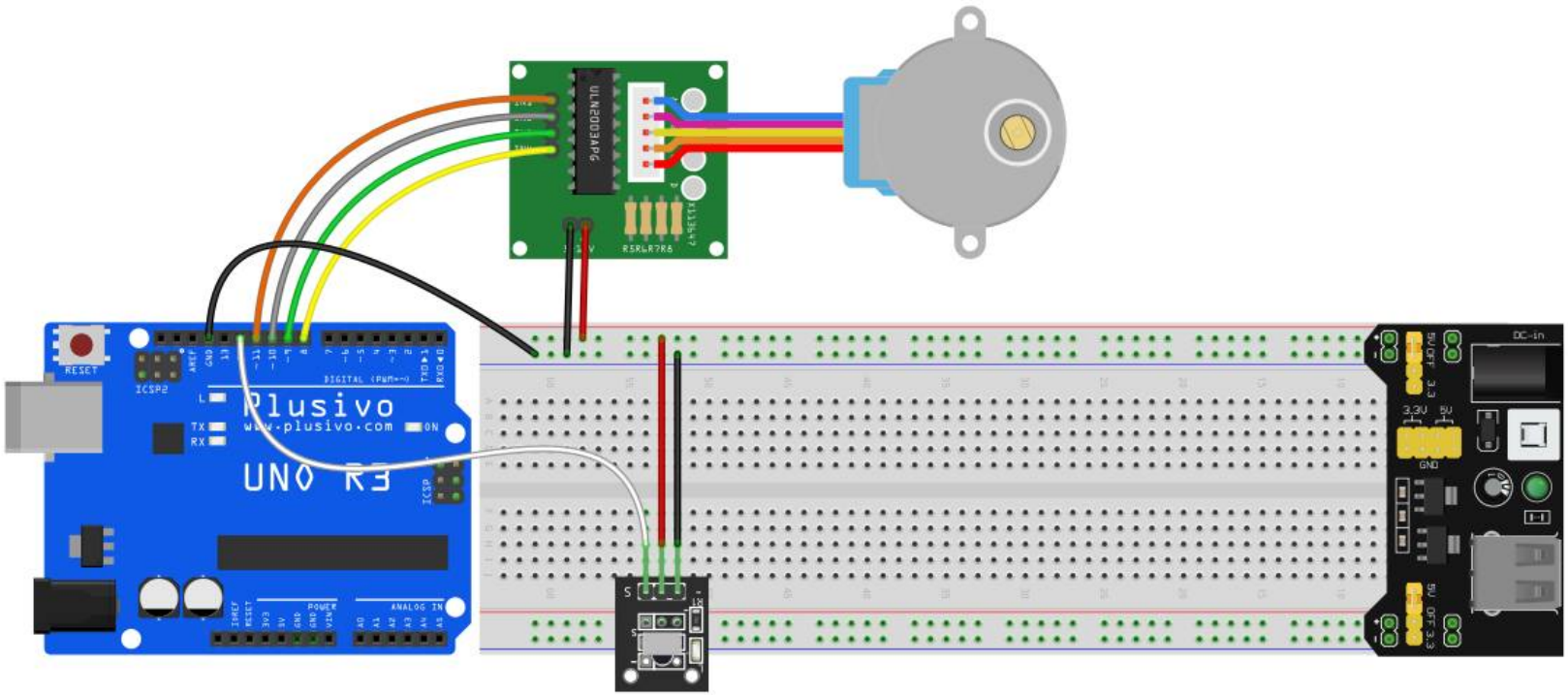
(1) x Plusivo Uno R3
(1) x 830 tie-points breadboard
(1) x IR receiver module
(1) x IR remote
(1) x ULN2003 stepper motor driver module
(1) x Stepper motor
(1) x Power supply module
(1) x 9V1A Adapter
(9) x F-M wires (Female to Male DuPont wires)
(1) x M-M wire (Male to Male jumper wire)

25.3 Connection

Schematic



Wiring diagram



We are using 4 pins (pins 8-11) to control the Stepper and 1 pin (pin 12) for the IR sensor.

We attach the 5V and Ground pins from the device to the sensor. For safety reasons, we should use a breadboard power source to power the stepper motor so we don't damage our UNO power supply.

25.4 Code

After wiring, please open the program located in the folder - Lesson 32 Controlling Stepper Motor and press UPLOAD to upload the code. If there are any errors, see Lesson 2 for details about program uploading. Before you can run this, make sure that you have installed the <IRremote> <Stepper> library or re-install it, if necessary. Otherwise, your code won't work. For details about loading the library file, see Lesson 1.

Only 2 values are distinguished from the IR Remote control: VOL+ and VOL-. Pressing VOL+ will make the motor do a full rotation clockwise. Alternatively, pressing VOL- will make the motor do a full rotation counter-clockwise.

25.5 Example picture