Elecrow UNO R3 Starter Kit for Arduino

Content

Lesson 0 Installing IDE

Introduction

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

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

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

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

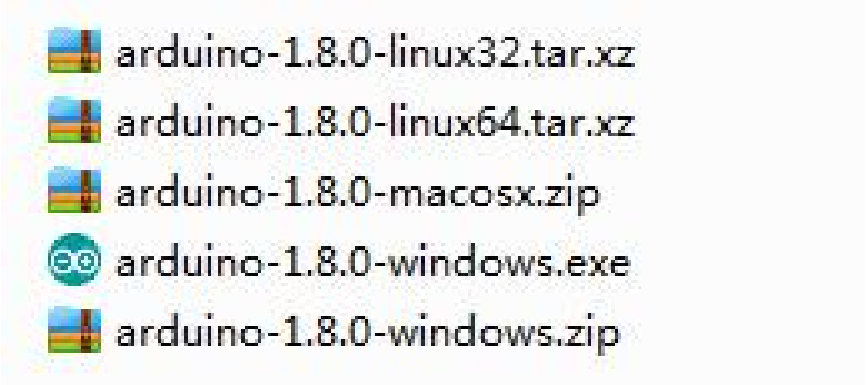
The version available at this website is usually the latest version, and the actual version may be newer than the version in the picture.

STEP2：Download the development software that is compatible with the operating system of your computer. <u>Take Windows as an example here.</u>

Click *Windows Installer*.

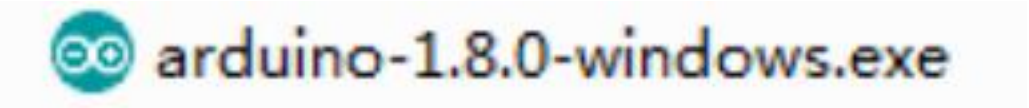
Click *JUSTDOWNLOAD*.

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

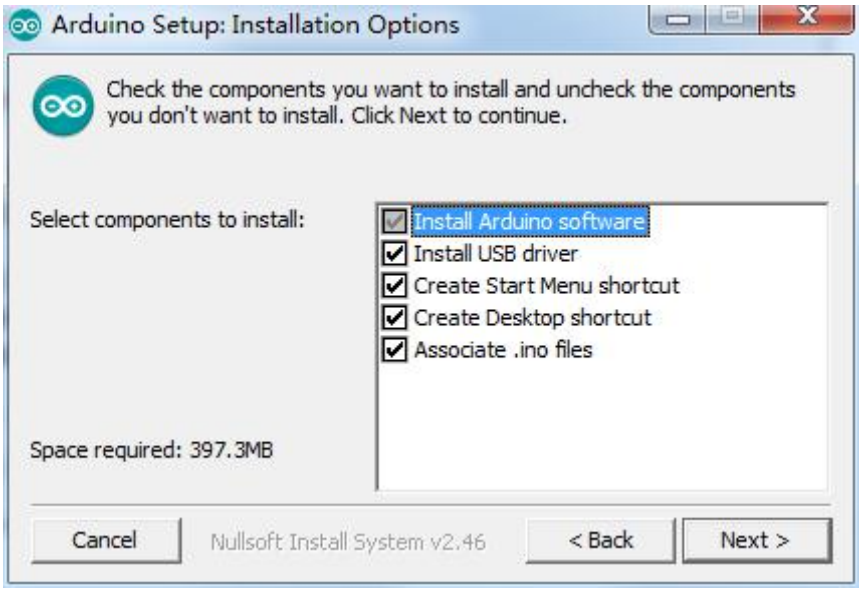


Installing Arduino (Windows)

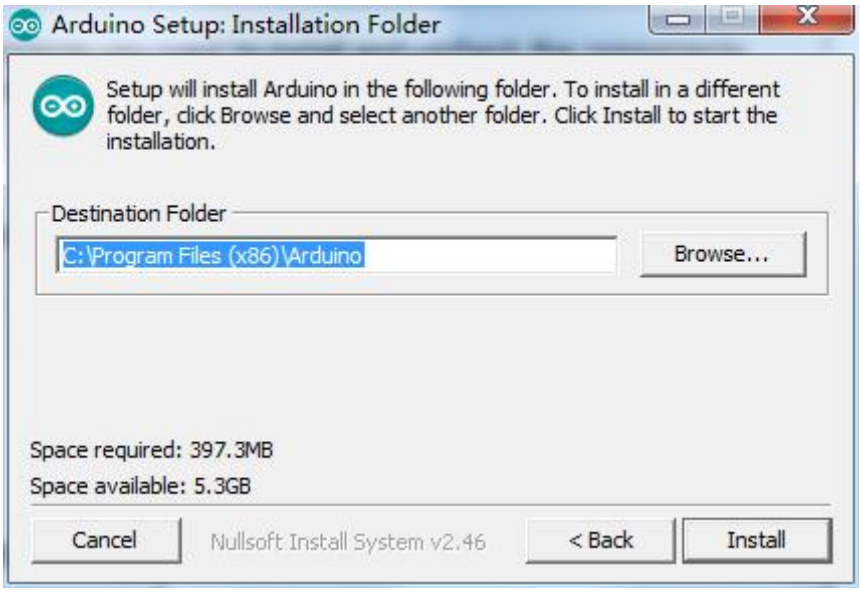
Install Arduino with the exe. Installation package.



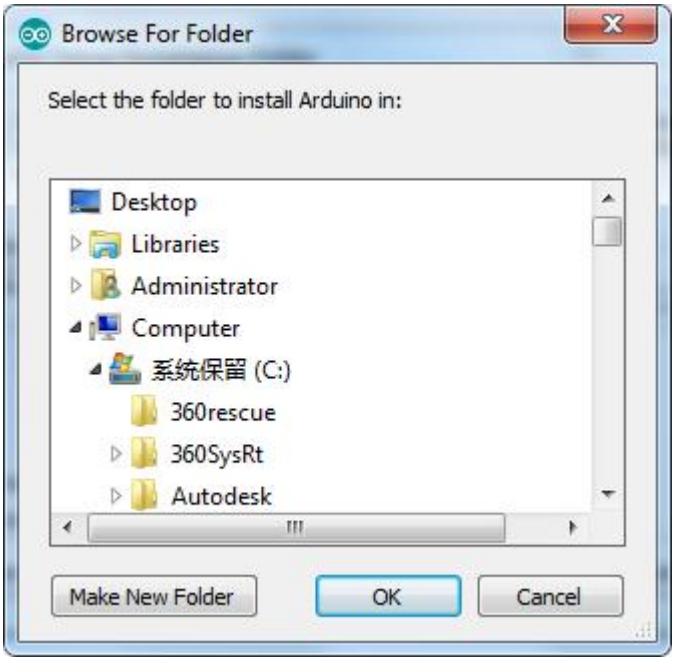
Click *I Agree* to see the following interface



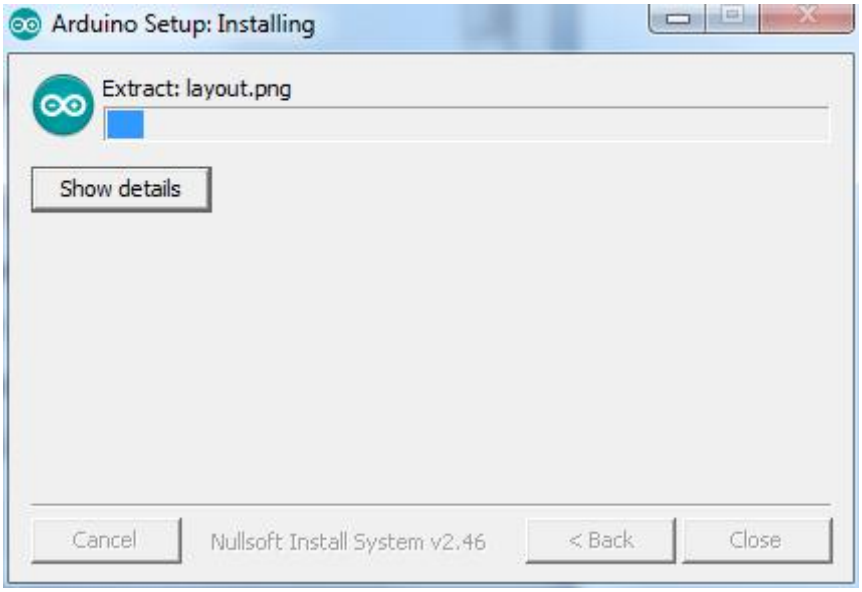
Click *Next*



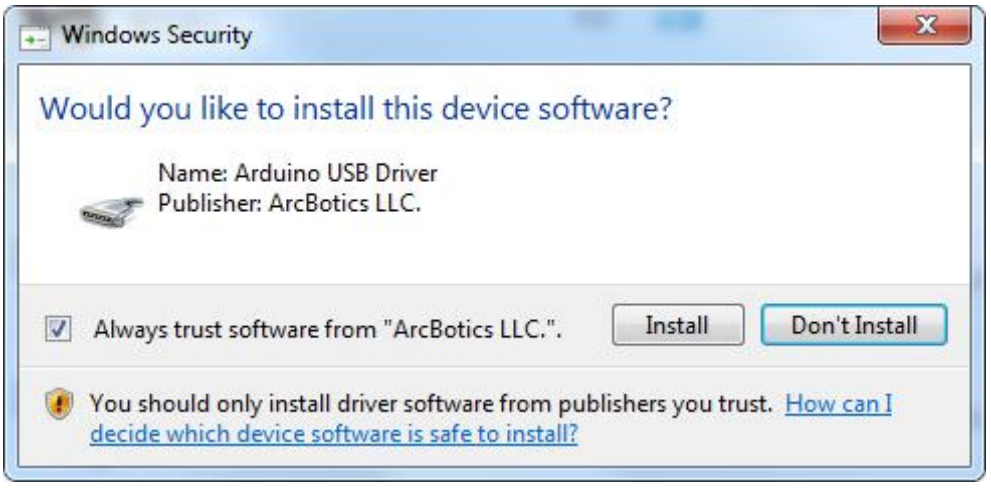
You can press Browse… to choose an installation path or directly type in the directory you want.



Click *Install* to initiate installation

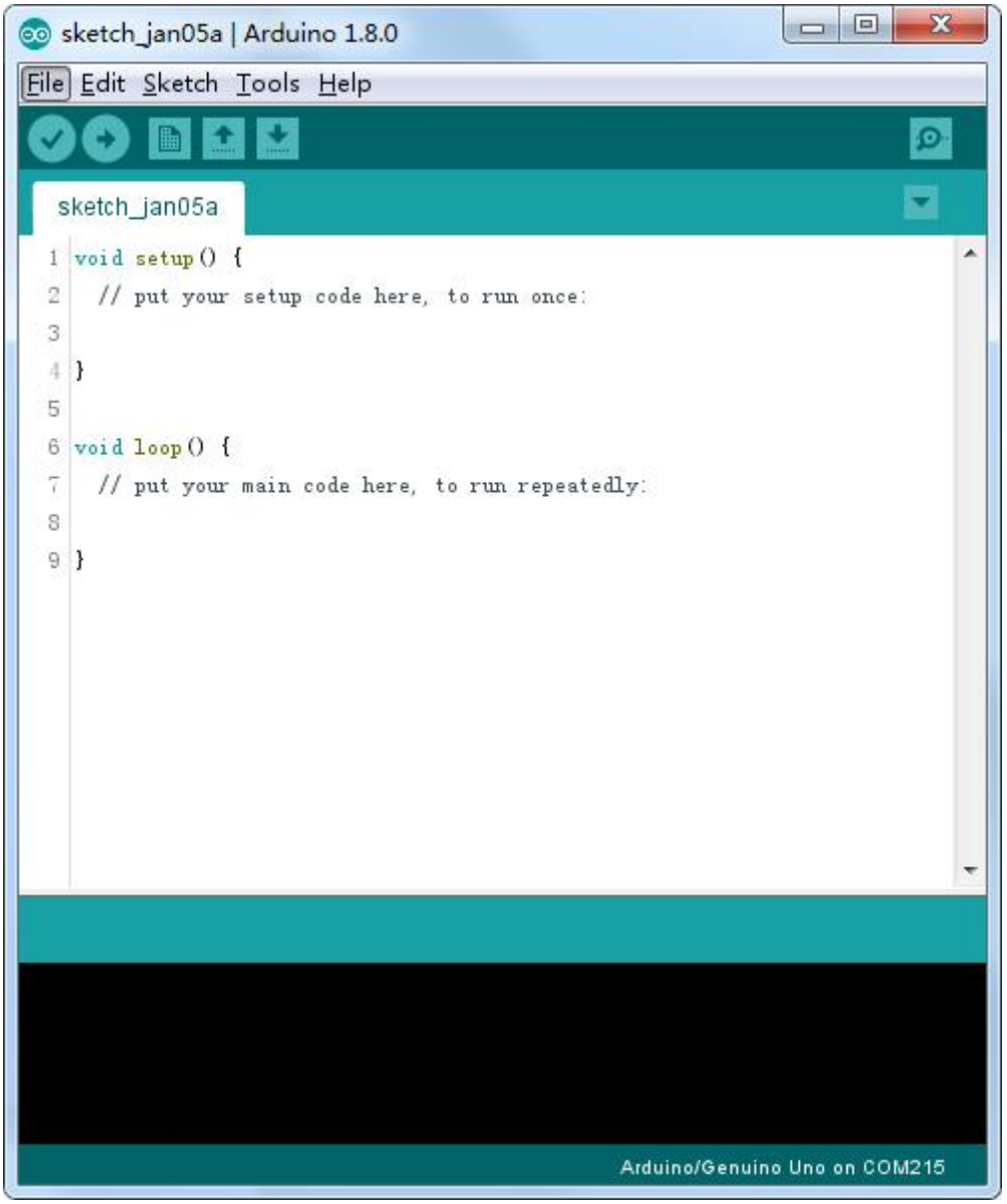


Finally, the following interface appears, click *Install* to finish the installation.



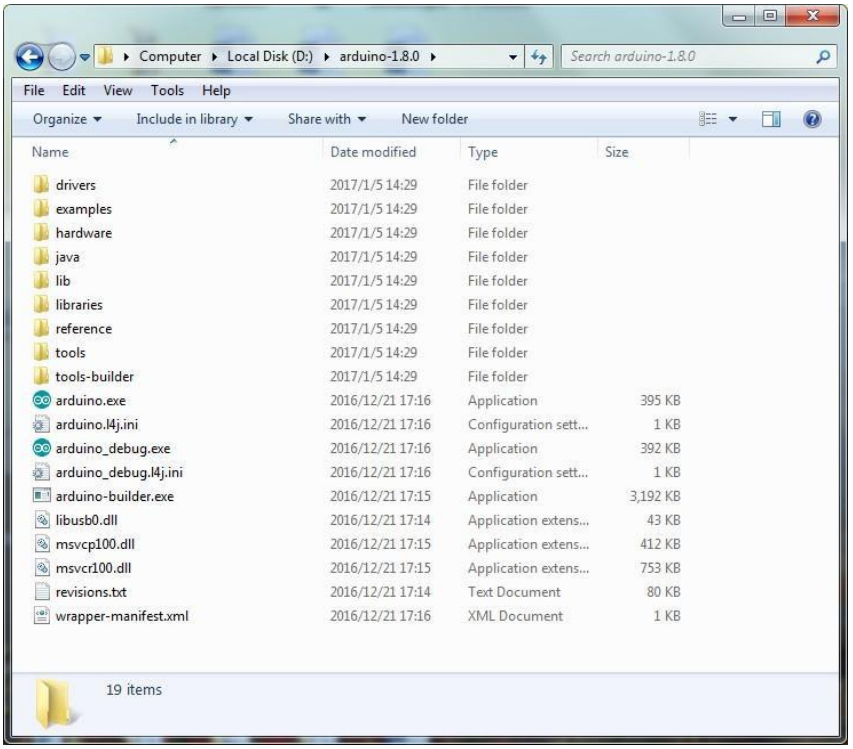
Next, the following icon appears on the desktop

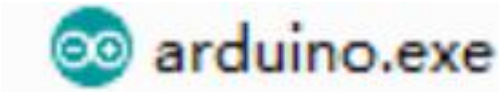
Double-click to enter the desired development environment

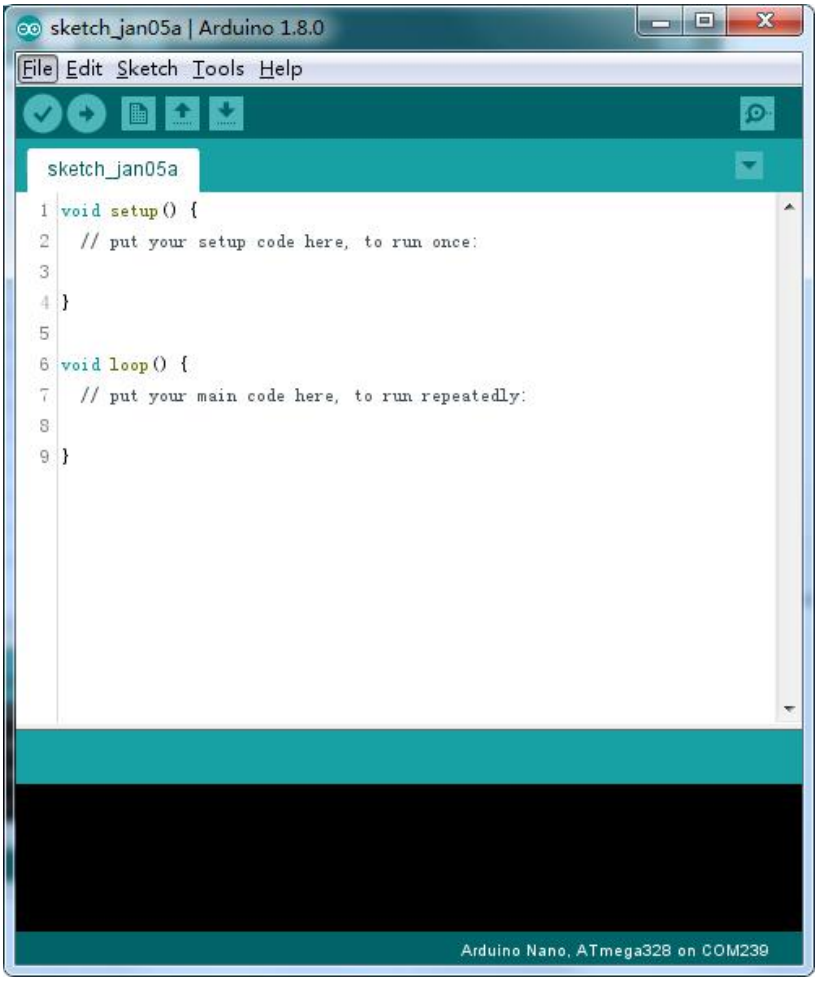


You may directly choose the installation package for installation and skip the contents below and jump to the next section. But if you want to learn some methods other than the installation package, please continue to read the section.

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


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


arduino.exe


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

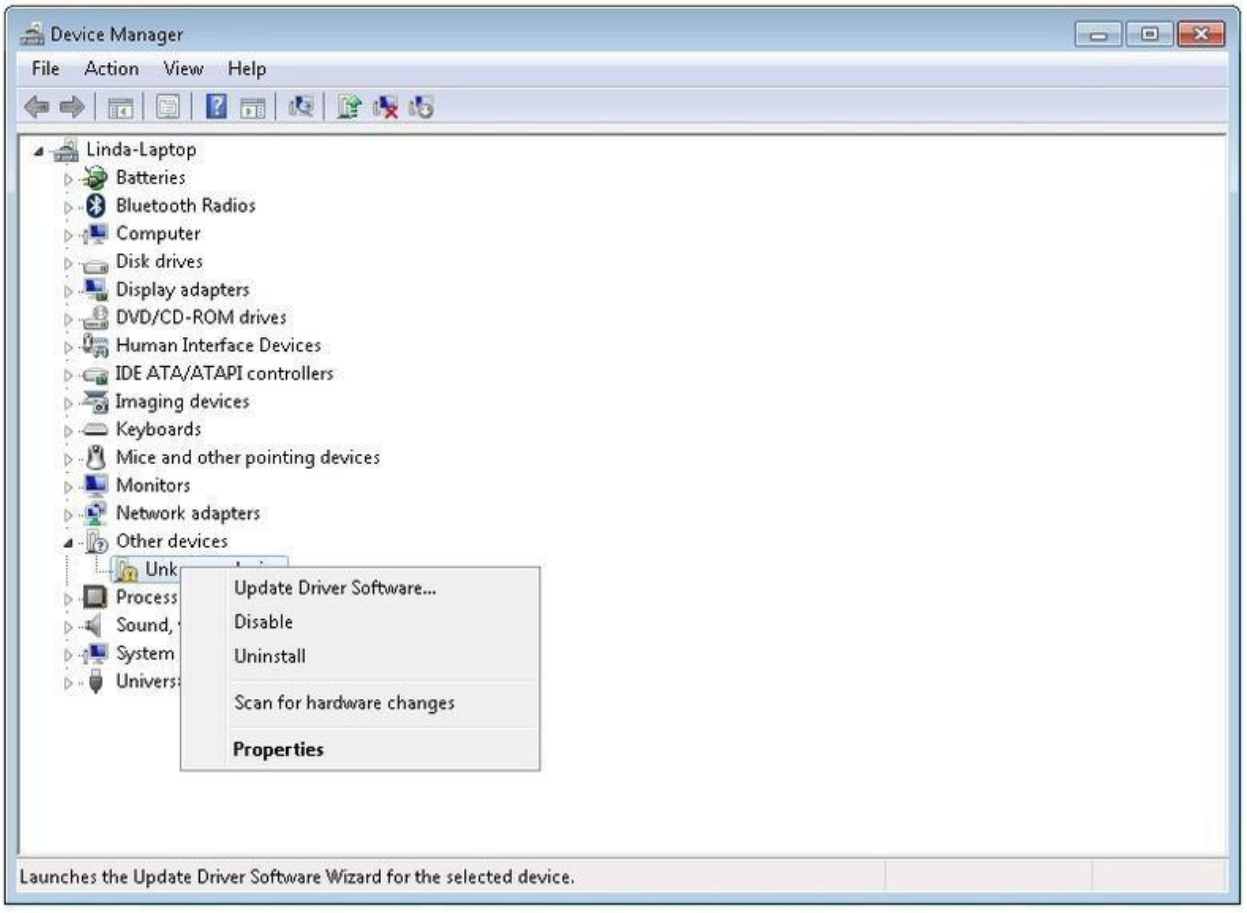
However, this installation method needs separate installation of driver.

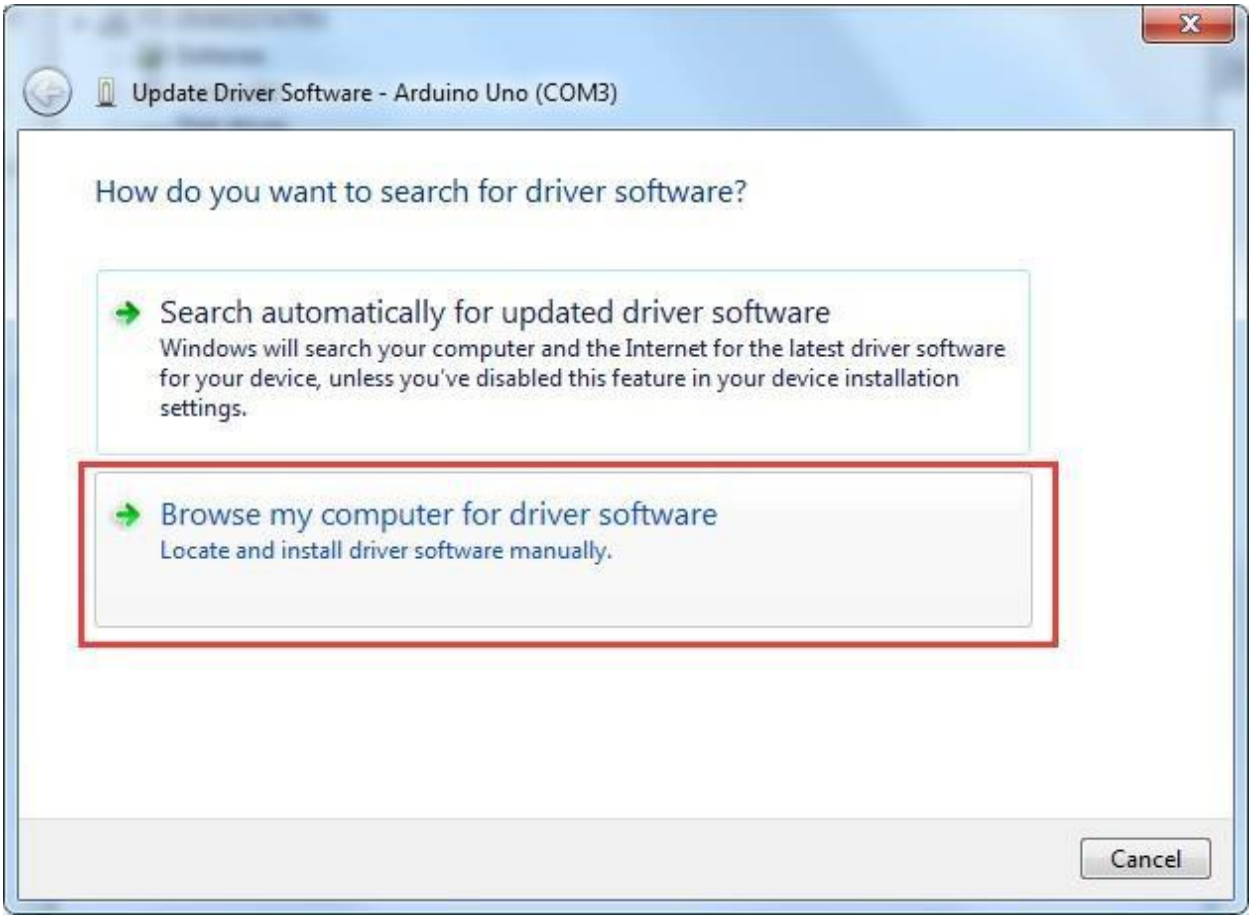
The Arduino folder contains both the Arduino program itself and the drivers that allow the Arduino to be connected to your computer by a USB cable. Before we launch the Arduino software, you are going to install the USB drivers.

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

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

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





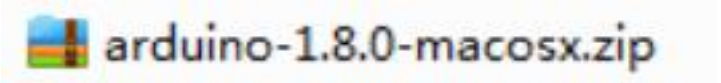
Right-click on the device and select the top menu option (Update Driver Software...). You will then be prompted to either 'Search Automatically for updated driver software' or 'Browse my computer for driver software'. Select the option to browse and navigate to the X\arduino1.8.0\drivers.

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

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

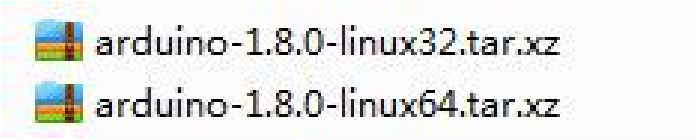
Installing Arduino (Mac OS X)

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



Installing Arduino (Linux)

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



TIPS: If you have problems in installing the drivers, please refer to the UNO R3, MEGA, NANO DRIVER FAQ.

Lesson 1 Add Libraries and Open Serial Monitor

Installing Additional Arduino Libraries

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

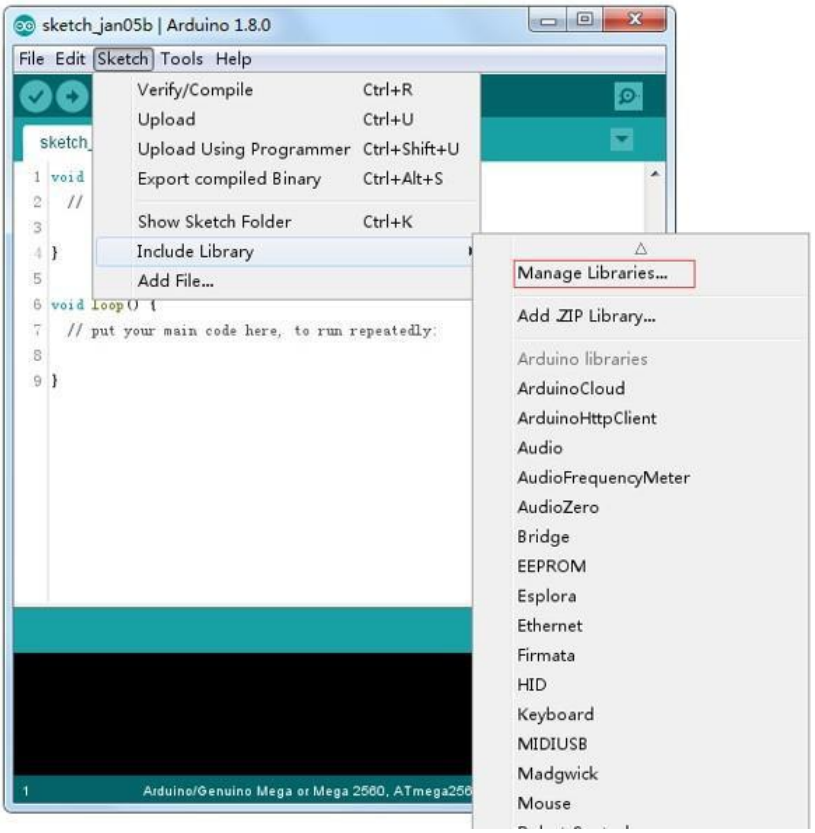
What are Libraries?

Libraries are a collection of code that makes it easy for you to connect to a sensor, display, module, etc. For example, the built-in LiquidCrystal library makes it easy to talk to character LCD displays. There are hundreds of additional libraries available on the Internet for download. The built-in libraries and some of these additional libraries are listed in the reference. To use the additional libraries, you will need to install them.

How to Install a Library

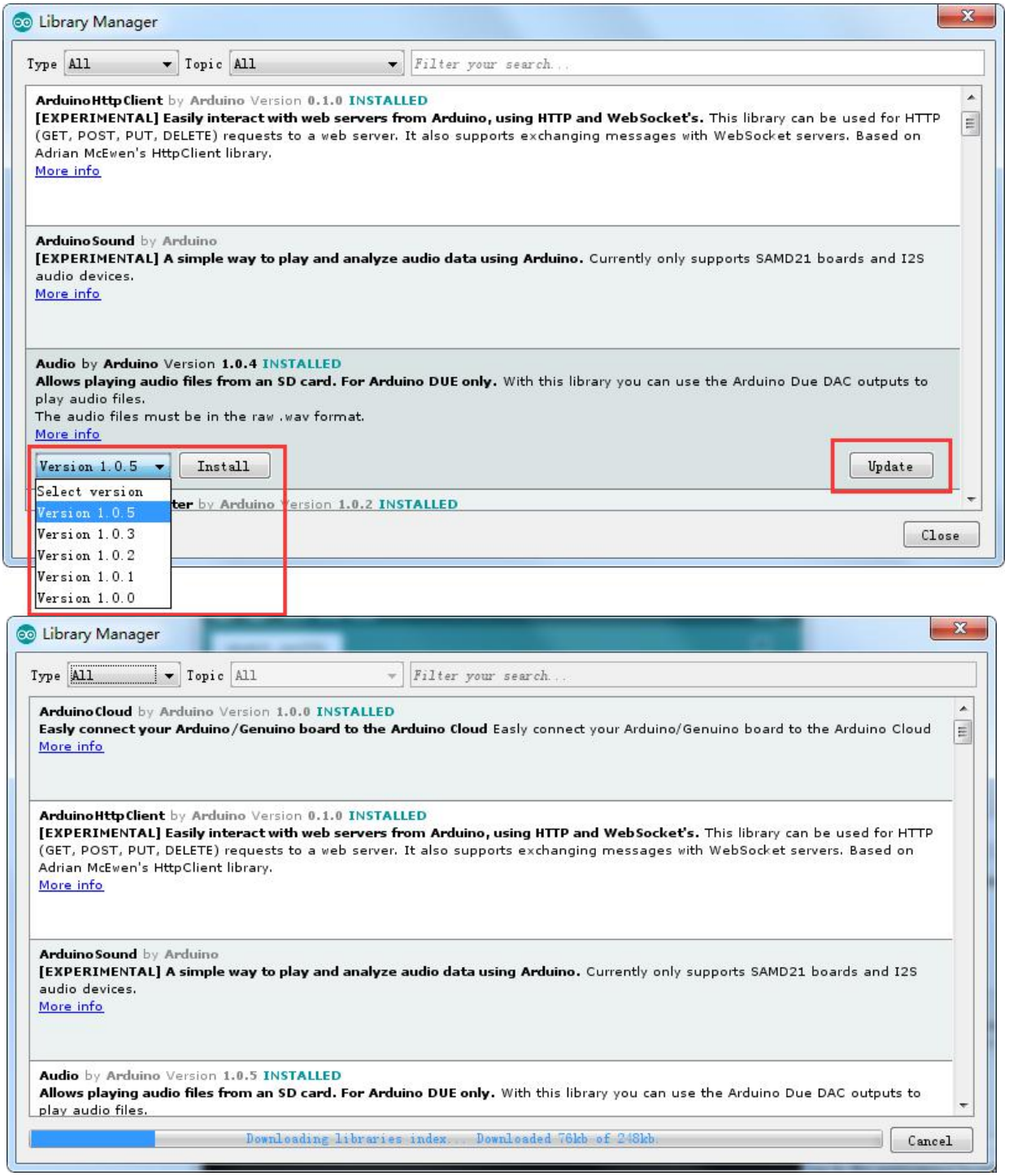
Using the Library Manager

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

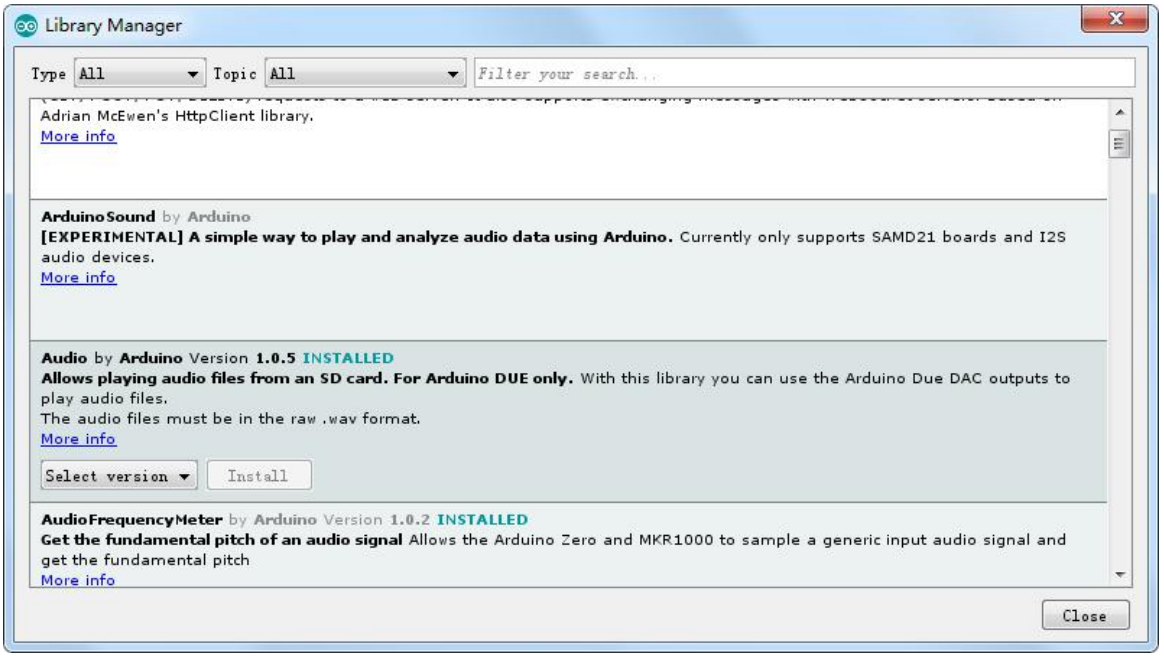


Then the library manager will open and you will find a list of libraries that are already installed or ready for installation. In this example we will install the Bridge library. Scroll the list to find it, then select the version of the library you want to install. Sometimes only one version of the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

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



Finally click on install and wait for the IDE to install the new library. Downloading may take time depending on your connection speed. Once it has finished, an Installed tag should appear next to the Bridge library. You can close the library manager.

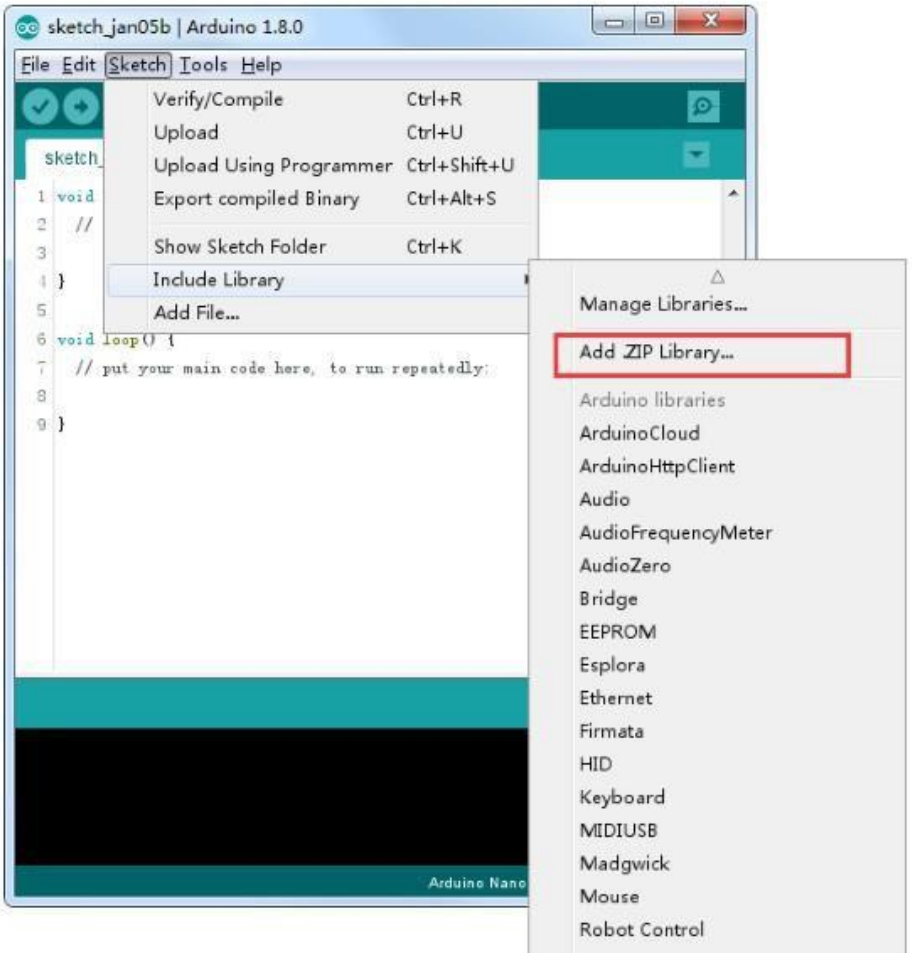


You can now find the new library available in the Include Library menu. If you want to add your own library open a new issue on Github.

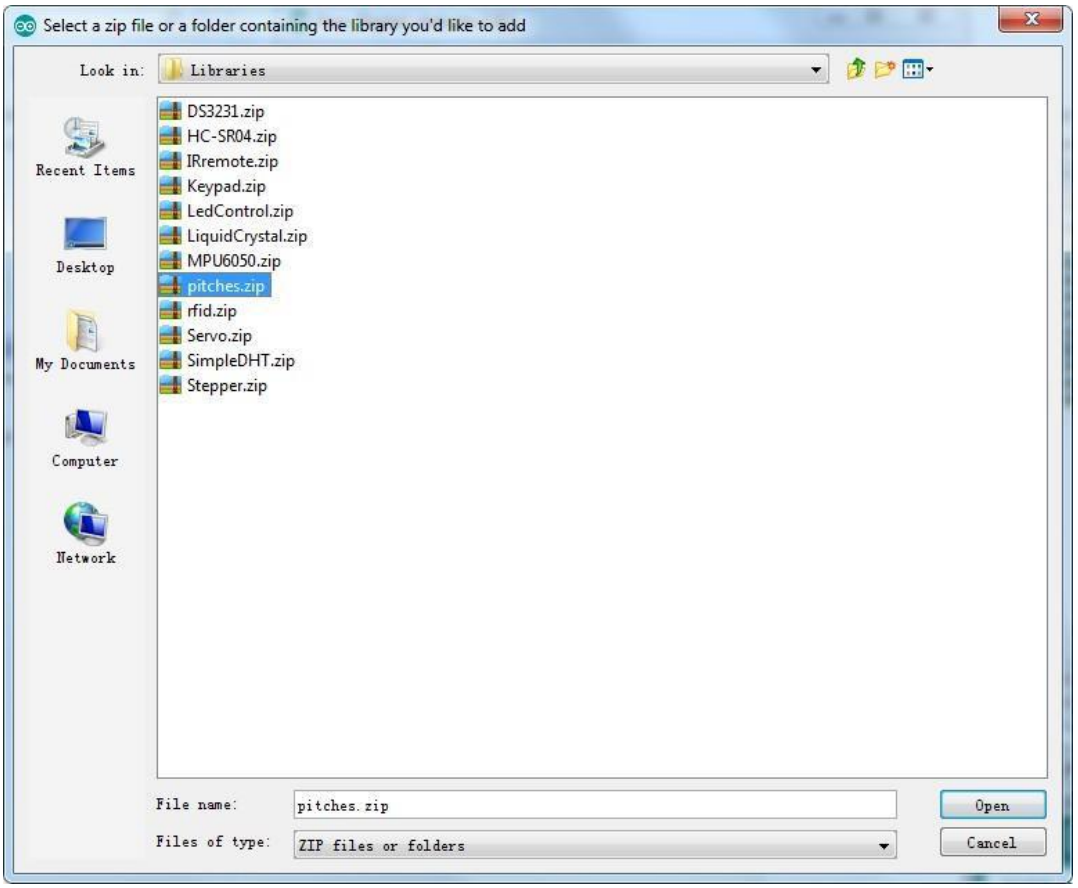
Importing a .zip Library

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

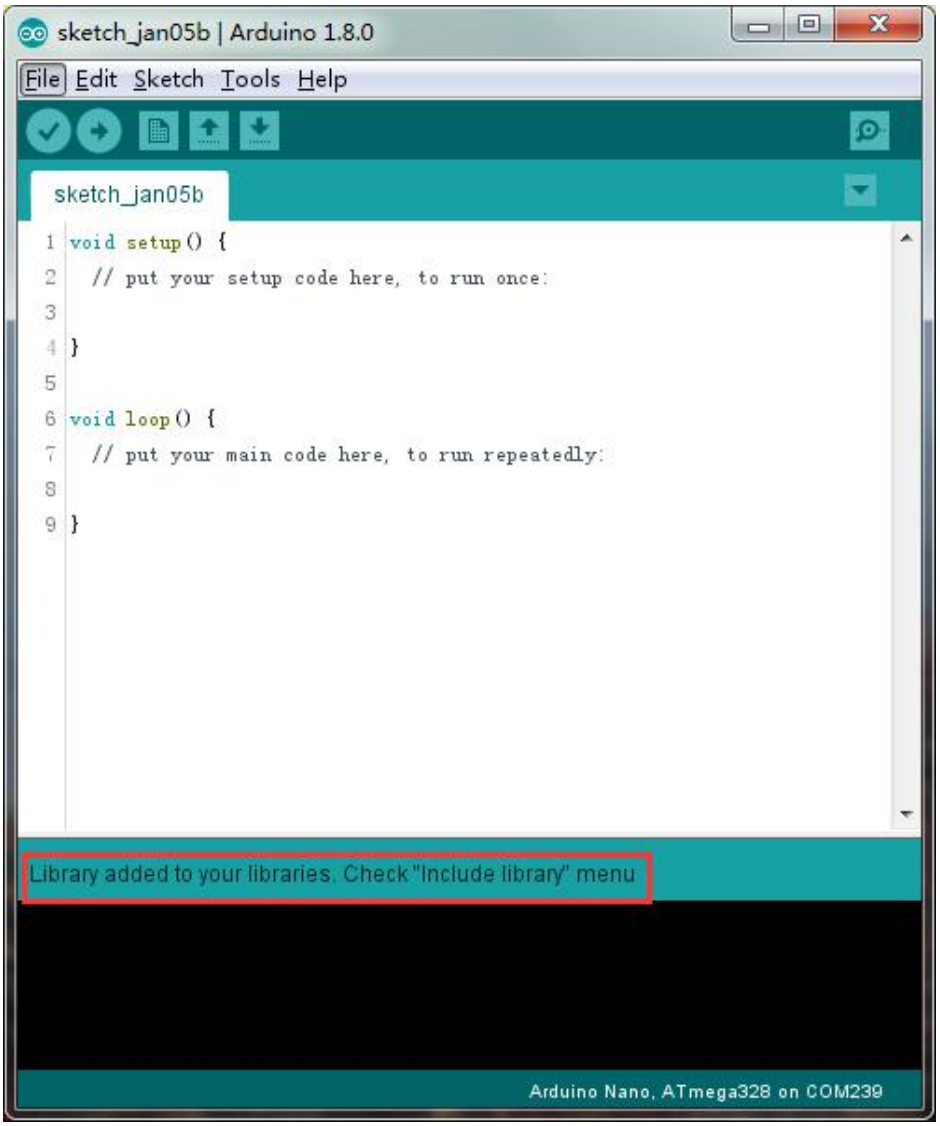
In the Arduino IDE, navigate to Sketch > Include Library. At the top of the drop down list, select the option to "Add .ZIP Library".



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



4



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

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

Manual installation

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

"ArduinoParty", uncompress ArduinoParty.zip. It should contain a folder calledArduinoParty, with files like ArduinoParty.cpp and ArduinoParty.h inside. (If the .cpp and .h files aren't in a folder, you'll need to create one. In this case, you'd make a folder called "ArduinoParty" and move into it all the files that were in the ZIP file, like ArduinoParty.cpp and ArduinoParty.h.)
Drag the ArduinoParty folder into this folder (your libraries folder). Under Windows, it will likely be called "My Documents\Arduino\libraries". For Mac users, it will likely be called "Documents/Arduino/libraries". On Linux, it will be the "libraries" folder in your sketchbook.

Your Arduino library folder should now look like this (on Windows):

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

or like this (on Mac and Linux):

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

There may be more files than just the .cpp and .h files, just make sure they're all there. (The library won't work if you put the .cpp and .h files directly into the libraries folder or if they're nested in an extra folder. For example: Documents\Arduino\libraries\ArduinoParty.cpp and Documents\Arduino\libraries\ArduinoParty\ArduinoParty\ArduinoParty.cpp won't work.)
Restart the Arduino application. Make sure the new library appears in the Sketch->Import Library menu item of the software. That's it! You've installed a library!

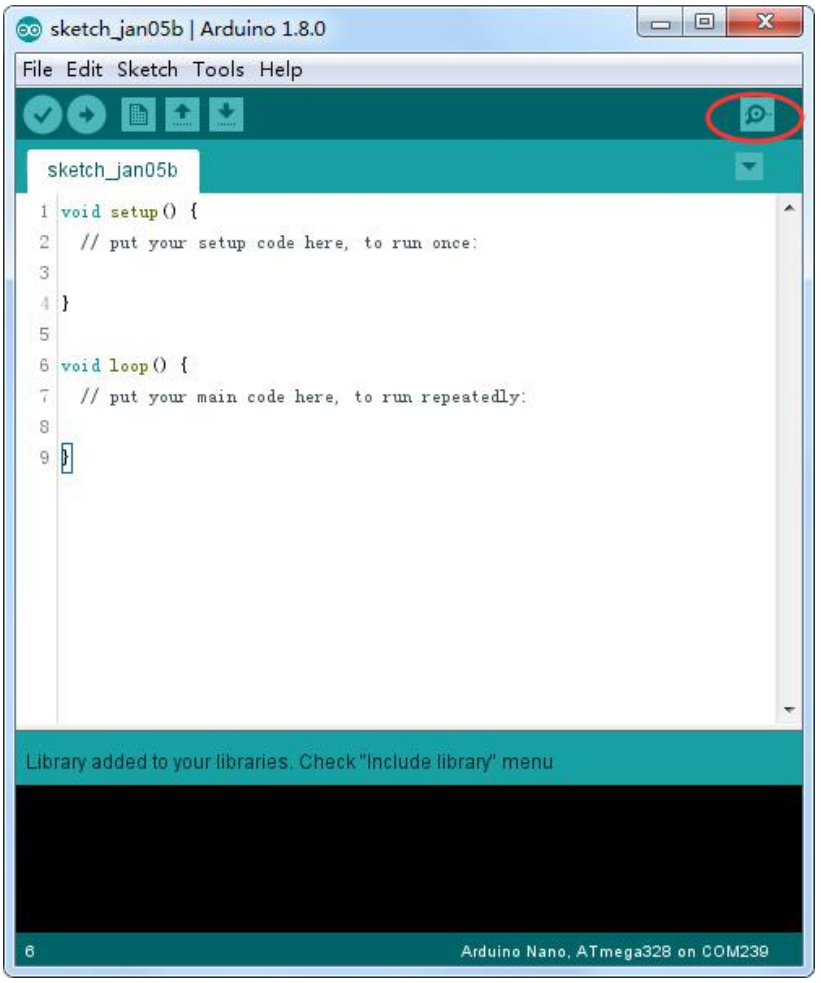
Arduino Serial Monitor (Windows, Mac, Linux)

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

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

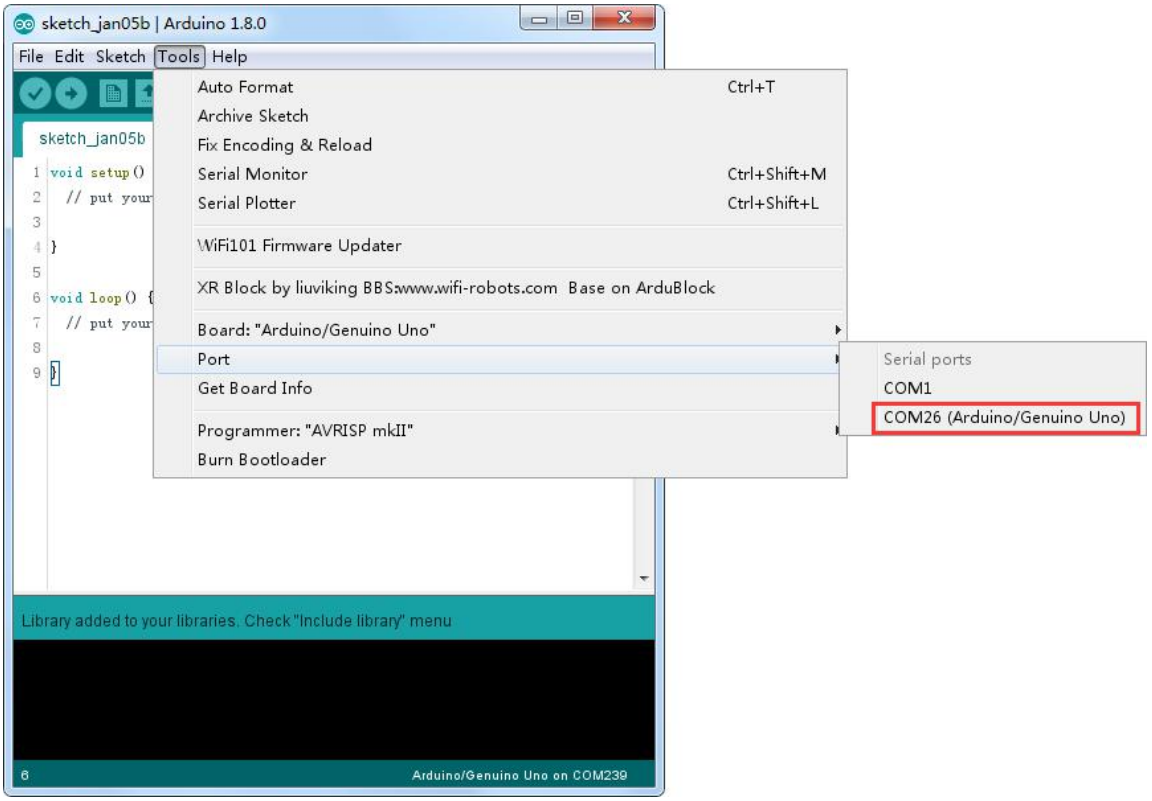
Making a Connection

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

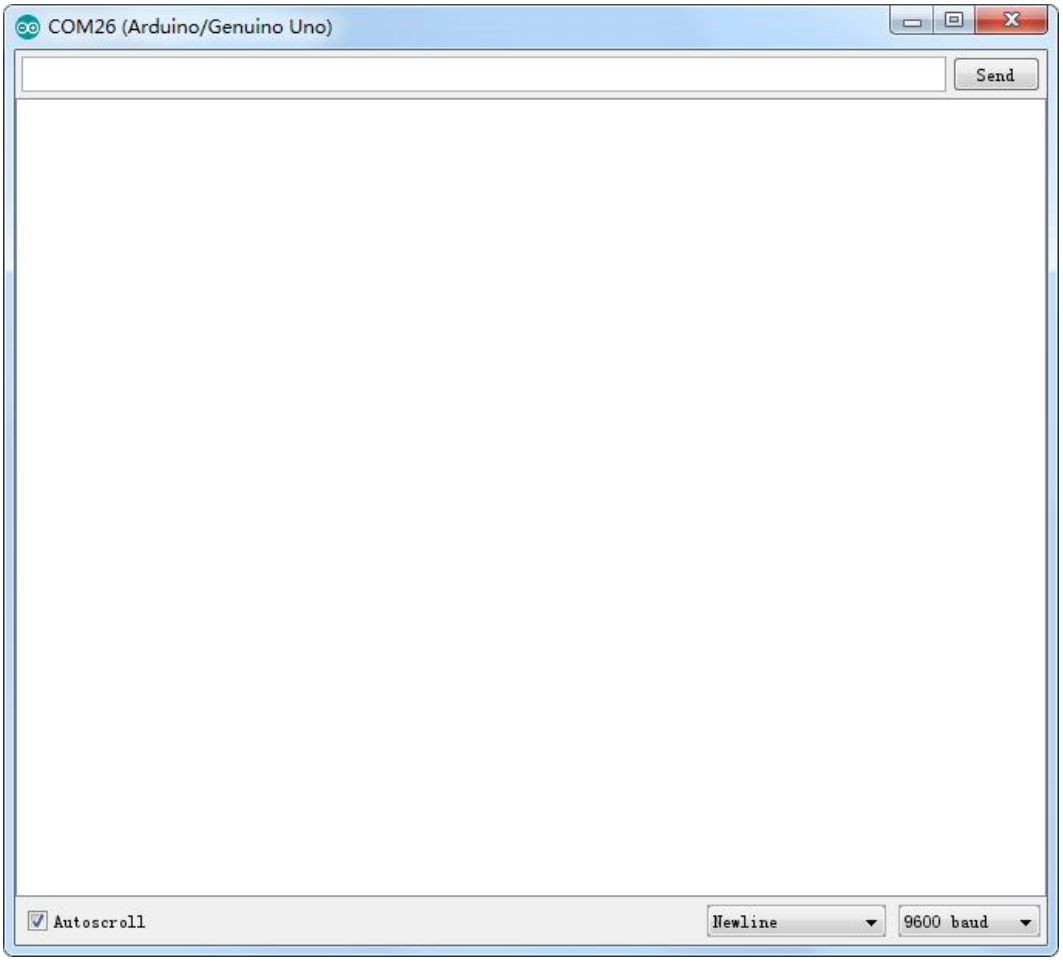


Selecting which port to open in the Serial Monitor is the same as selecting a port for uploading Arduino code. Go to Tools -> Serial Port, and select the correct port.

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

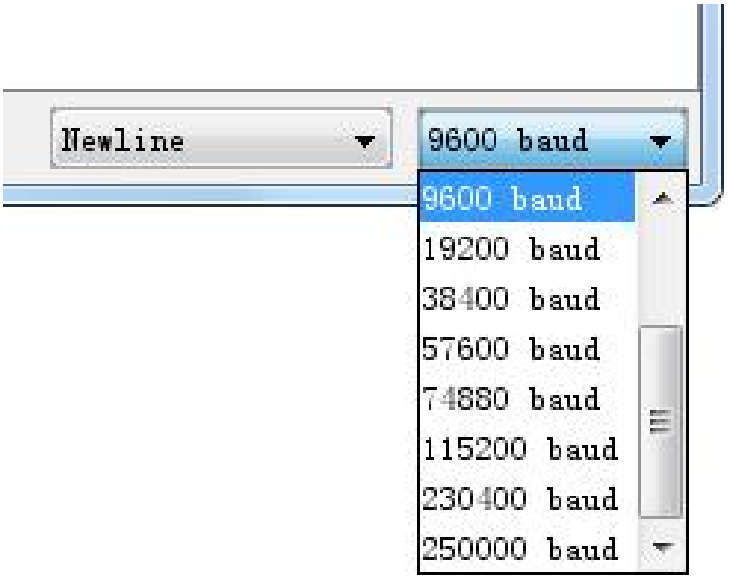


Once open, you should see something like this:



Settings

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



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

Pros

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

Cons

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

Lesson 2 Blink

Overview

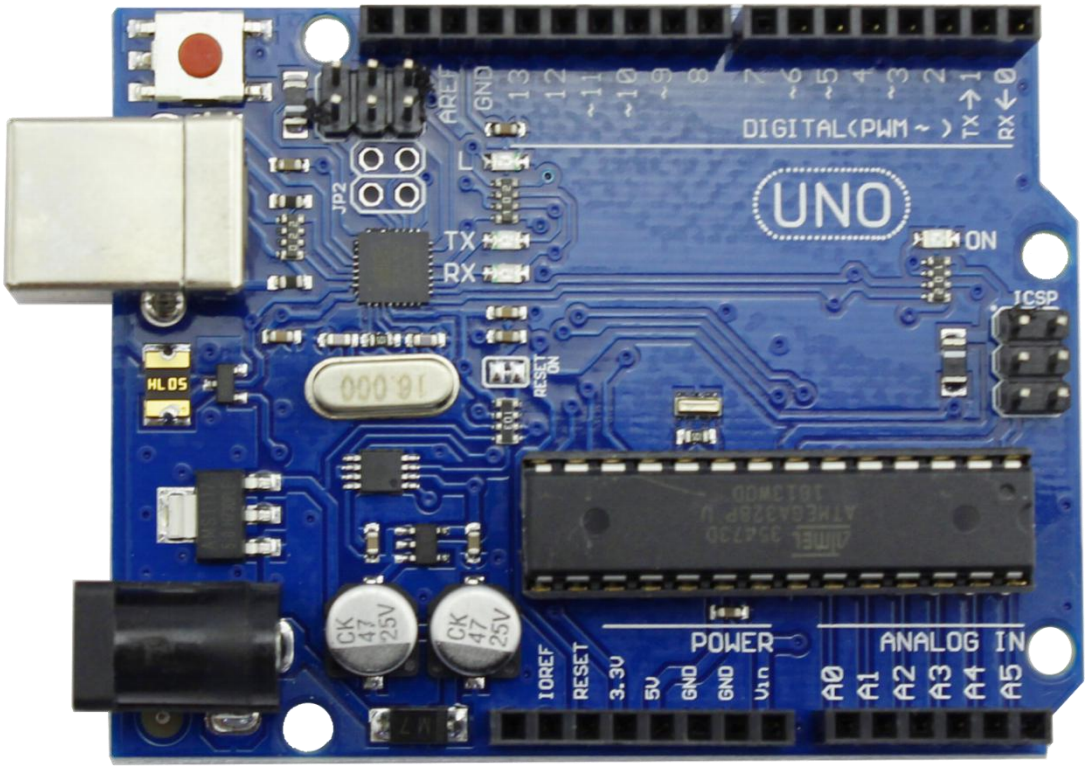
In this lesson, you will learn how to program your UNO R3 controller board to blink the Arduino's built-in LED, and how to download programs by basic steps.

Component Required:

(1) x Uno R3

Principle

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



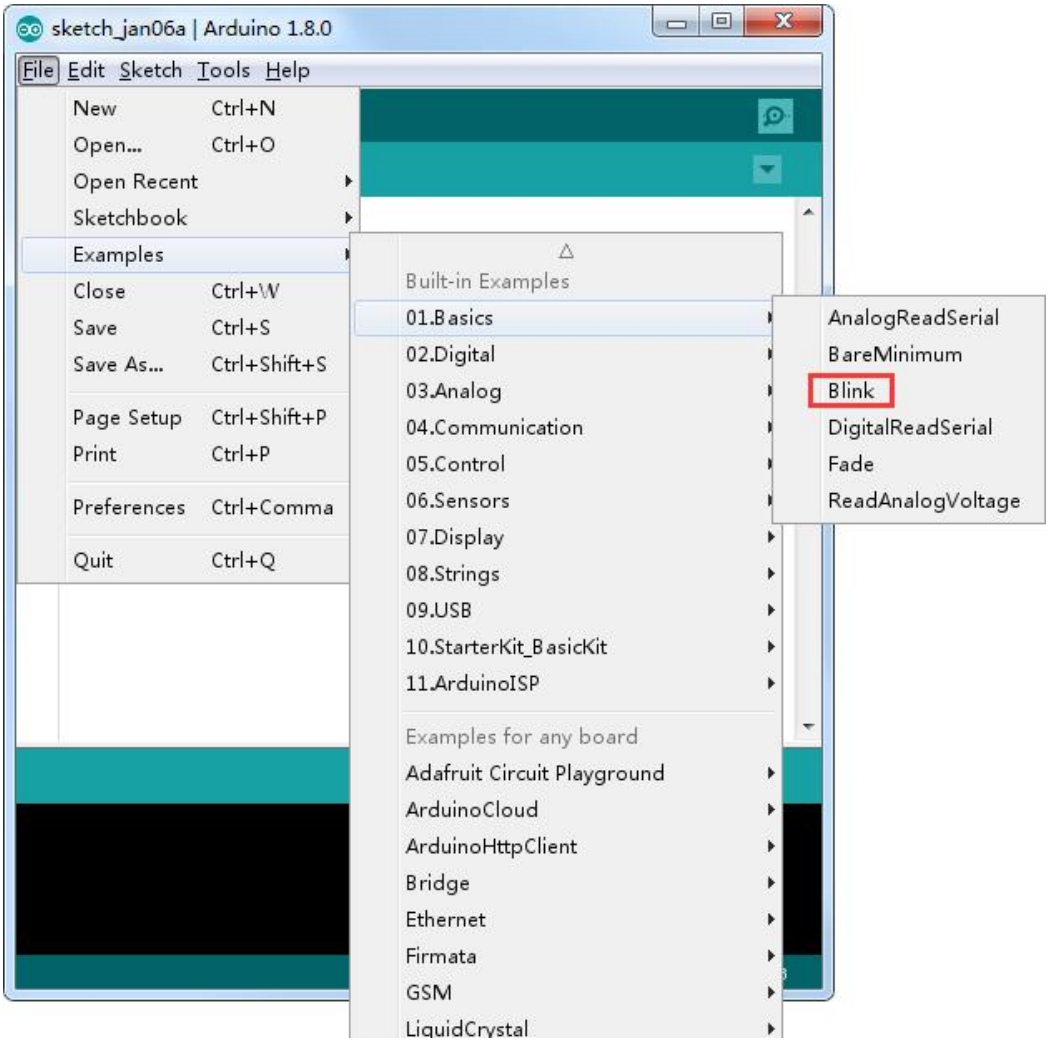
You may find that your UNO R3 board's 'L' LED already blinks when you connect it to a USB plug. This is because the boards are generally shipped with the 'Blink' sketch pre-installed.

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

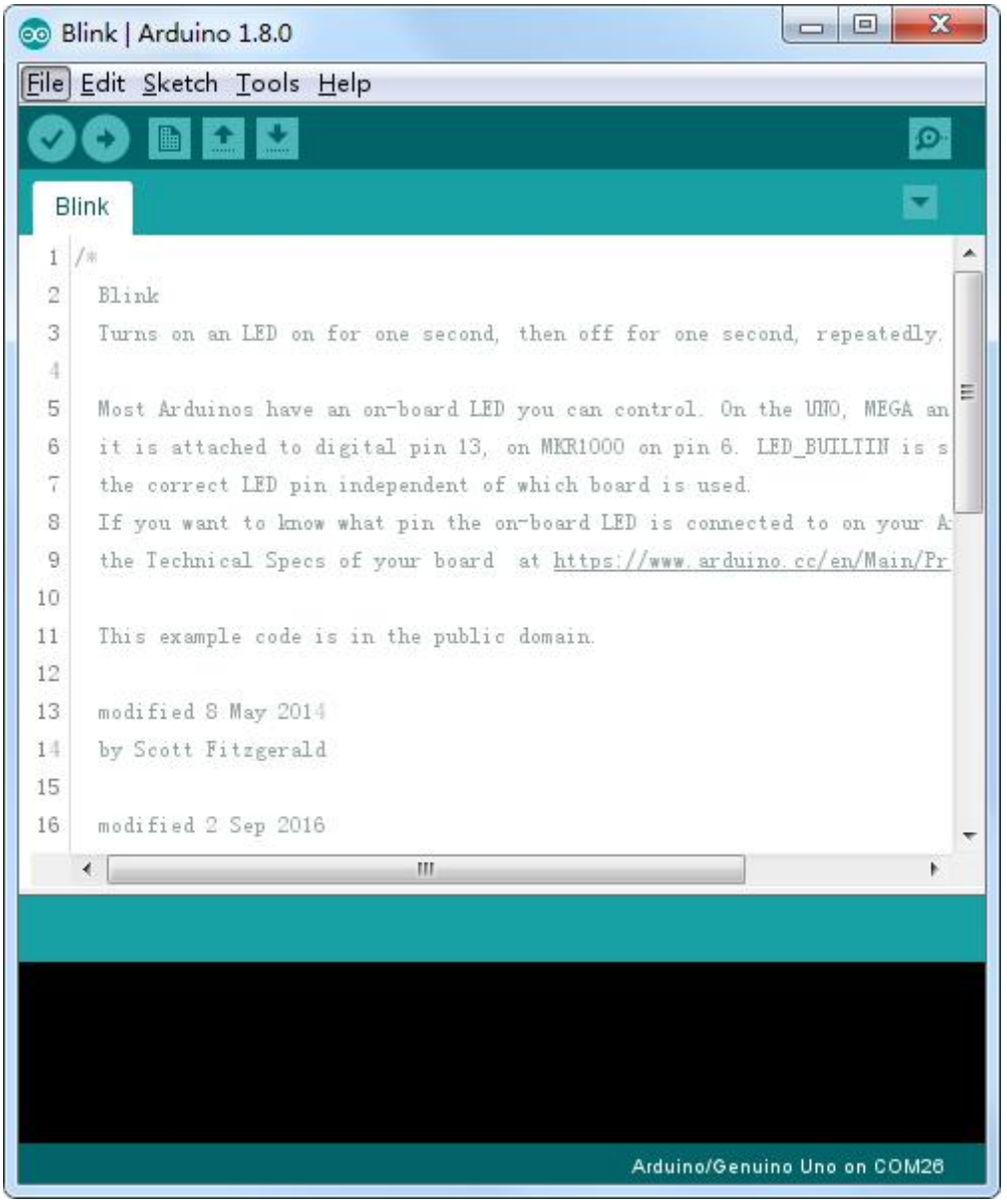
In Lesson 0, you set up your Arduino IDE and made sure that you could find the right serial port for it to connect to your UNO R3 board. The time has now come to put that connection to the test and program your UNO R3 board.

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

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



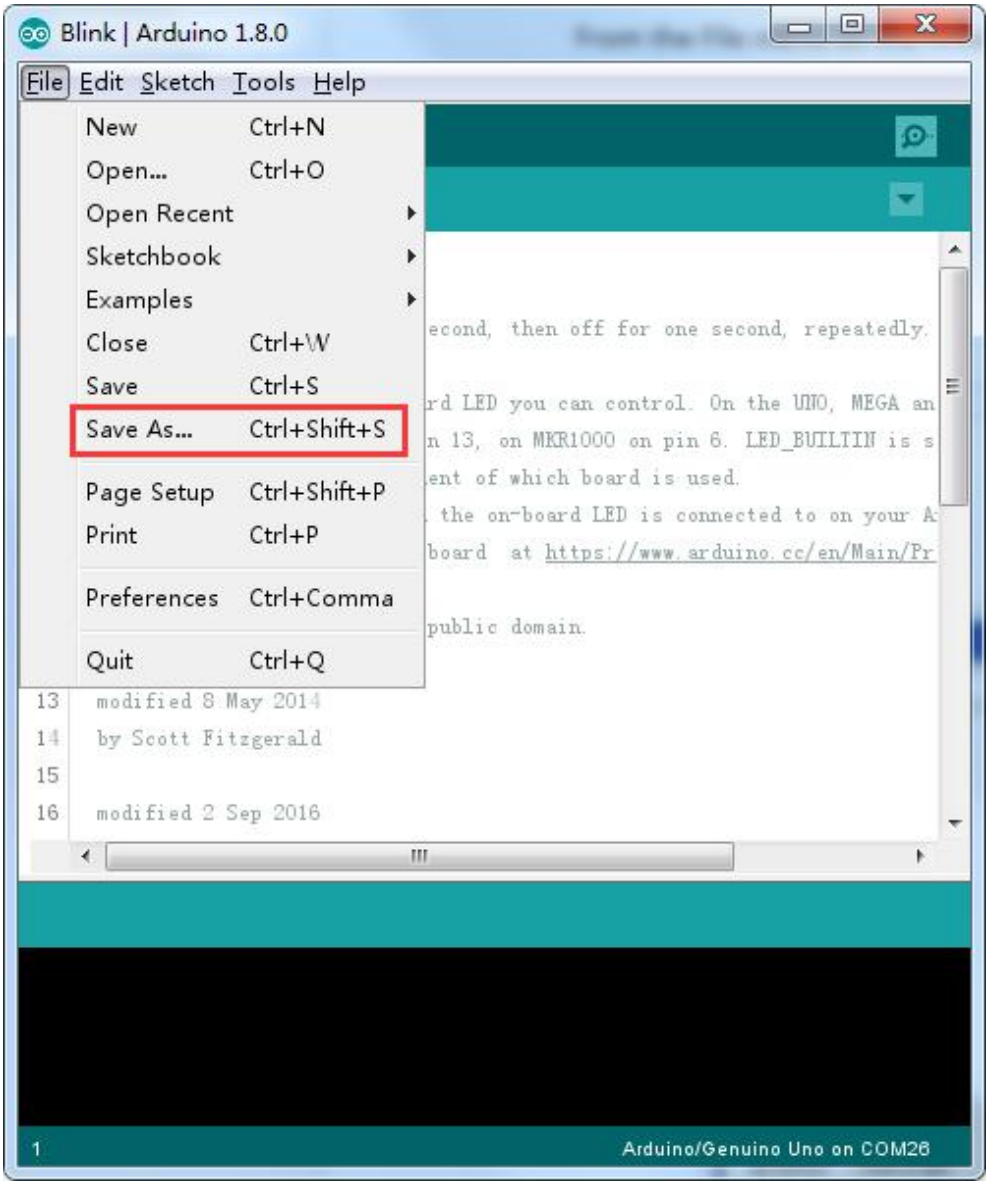
When the sketch window opens, enlarge it so that you can see the entire sketch in the window.

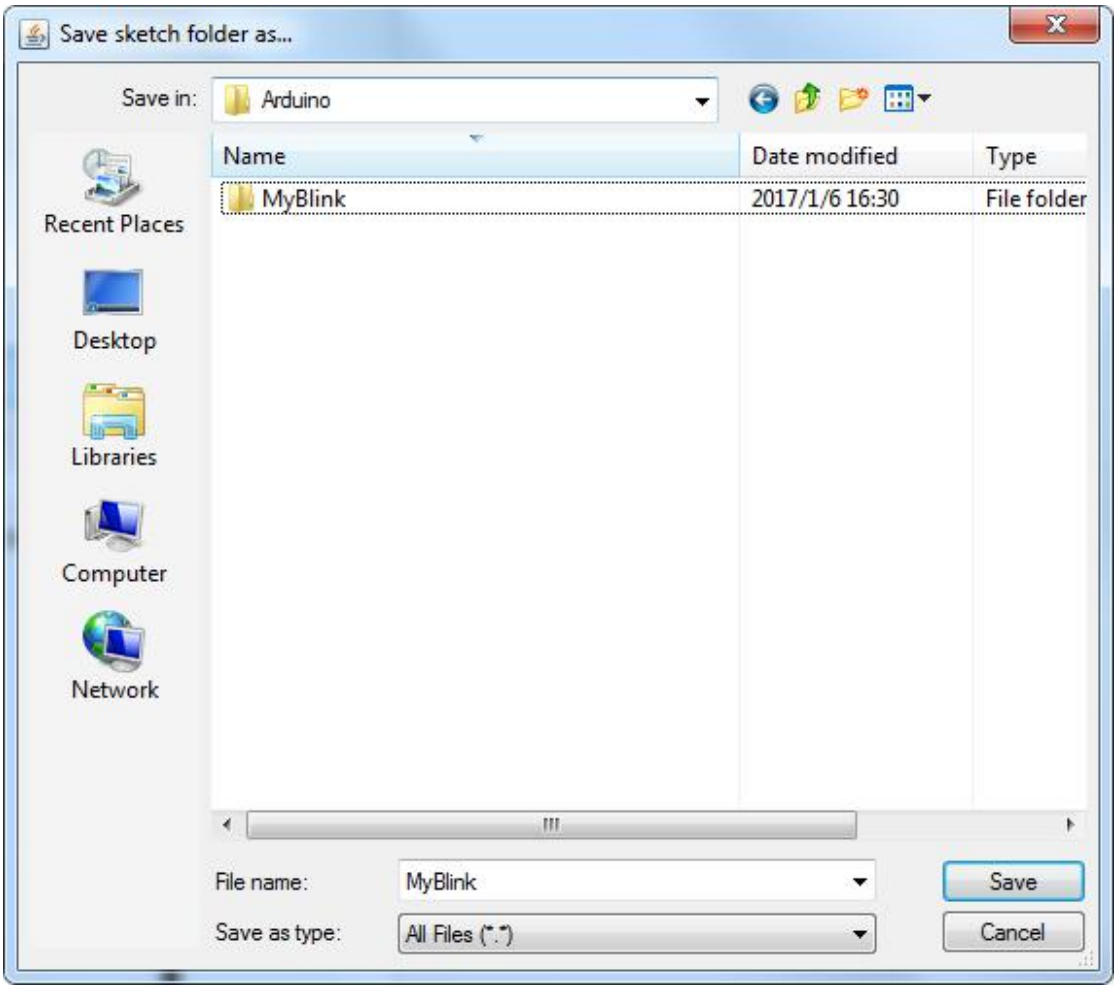


The example sketches included with the Arduino IDE are 'read-only'. That is, you can upload them to an UNO R3 board, but if you change them, you cannot save them as the same file.

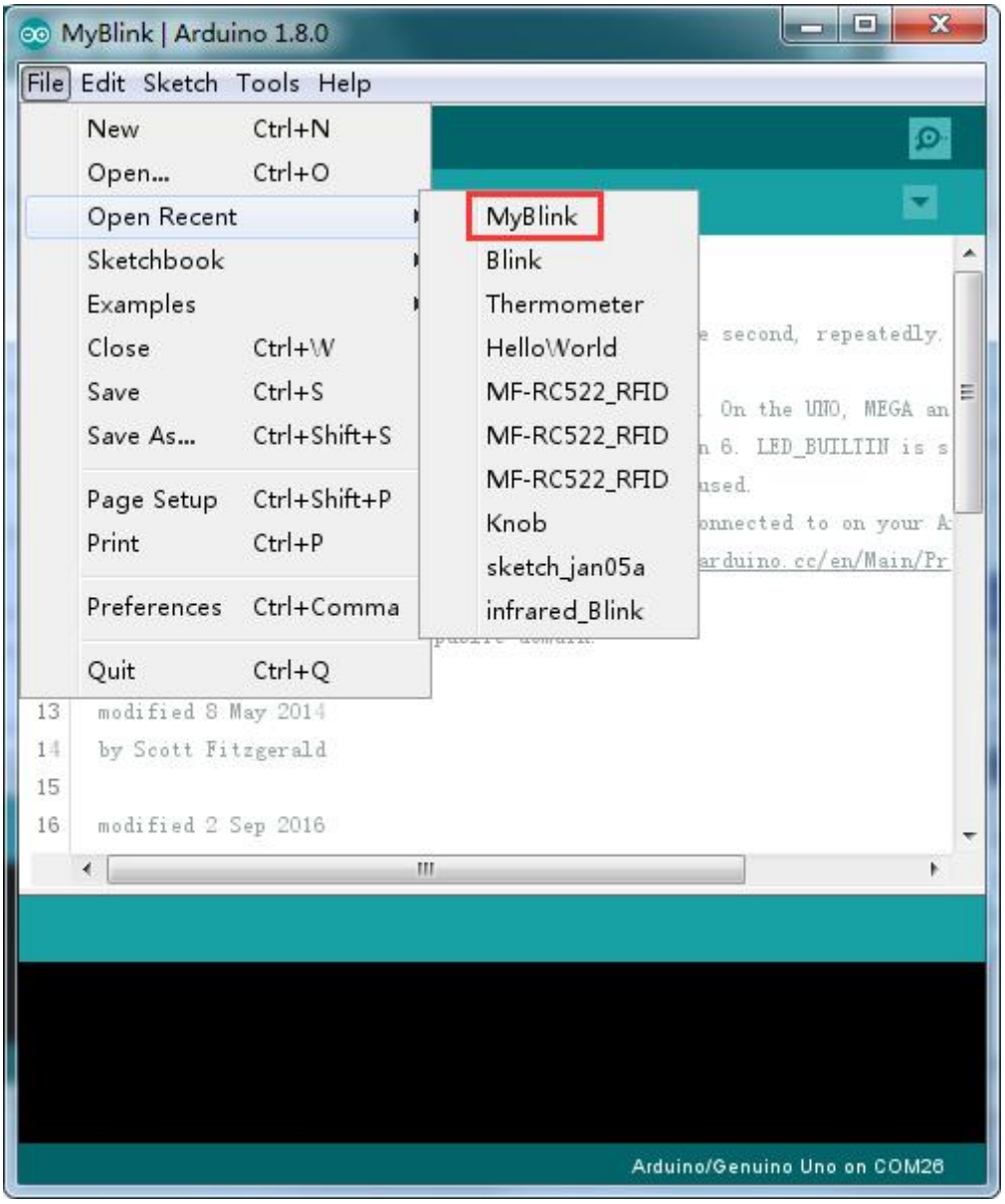
Since we are going to change this sketch, the first thing you need to do is save your own copy.

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

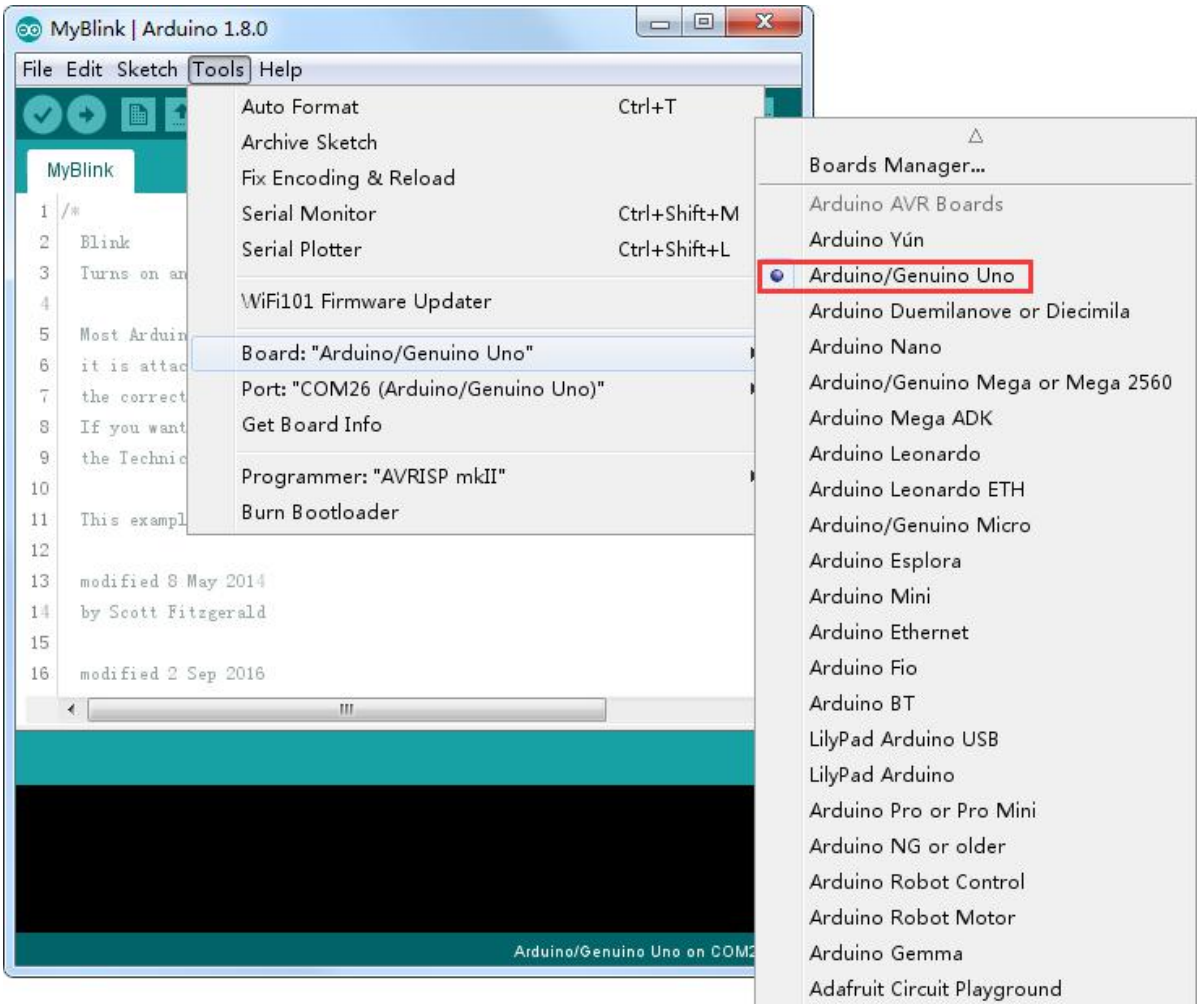

Blink | Arduino 1.8.0
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
13 modified 8 May 2014
14 by Scott Fitzgerald
15
16 modified 2 Sep 2016
1
Arduino/Genuino Uno on COM26

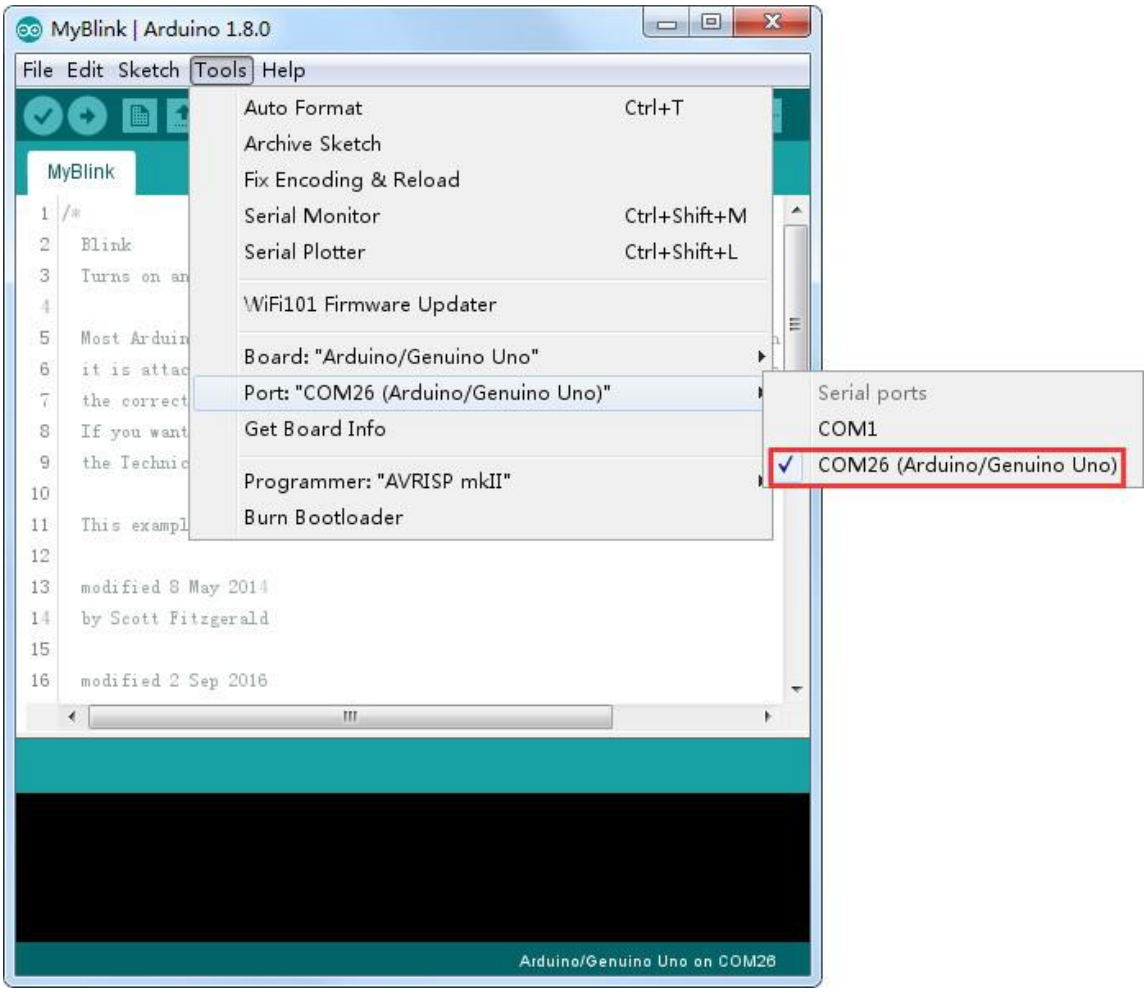


You have saved your copy of 'Blink' in your sketchbook. This means that if you ever want to find it again, you can just open it using the File > Sketchbook menu option.



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


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


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

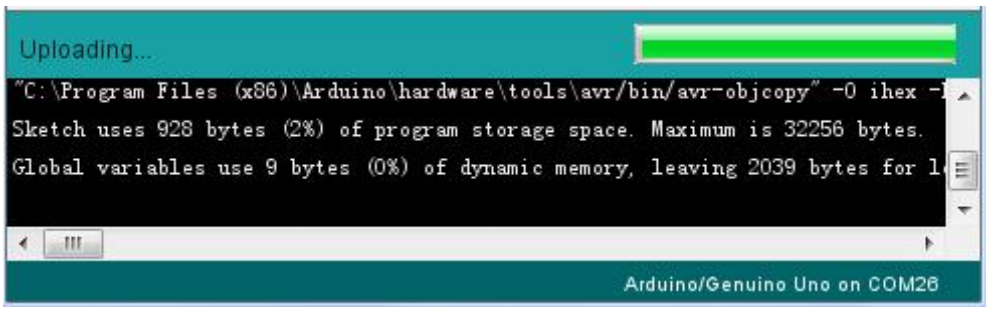
Note: The Board Type and Serial Port here are not necessarily the same as shown in picture. If you are using 2560, then you will have to choose Mega 2560 as the Board Type, other choices can be made in the same manner. And the Serial Port displayed for everyone is different, despite COM 26 chosen here, it could be COM3 or COM4 on your computer. A right COM port is supposed to be COMX (arduino XXX), which is by the certification criteria.

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

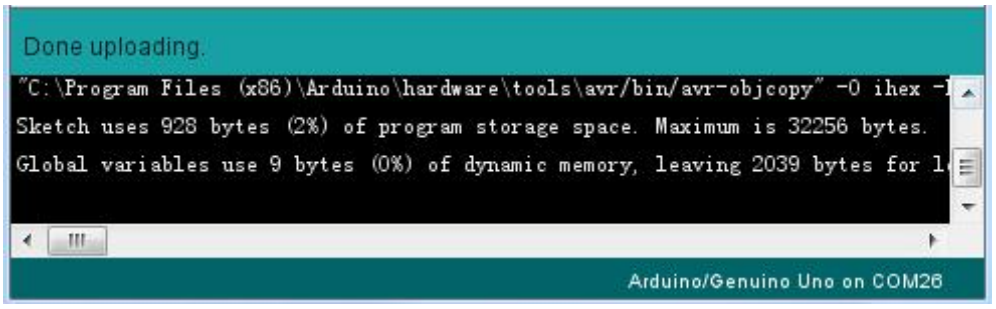
Click on the 'Upload' button. The second button from the left on the toolbar.

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

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



Finally, the staus will change to 'Done'.



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

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

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

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

Open the code

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

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

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

The first line of code is:

```
int led = 13;
```

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

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

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

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

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

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

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

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

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

```
30 // the loop function runs over and over again forever
31 void loop() {
32   digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the volt
33   delay(500)                         // wait for a second
34   digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the vo
35   delay(500)                         // wait for a second
36 }
```

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

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

Lesson 3 LED

Overview

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

Component Required:

(1) x Uno R3

(1) x 5mm red LED

(1) x 220 ohm resistor

(1) x 1k ohm resistor

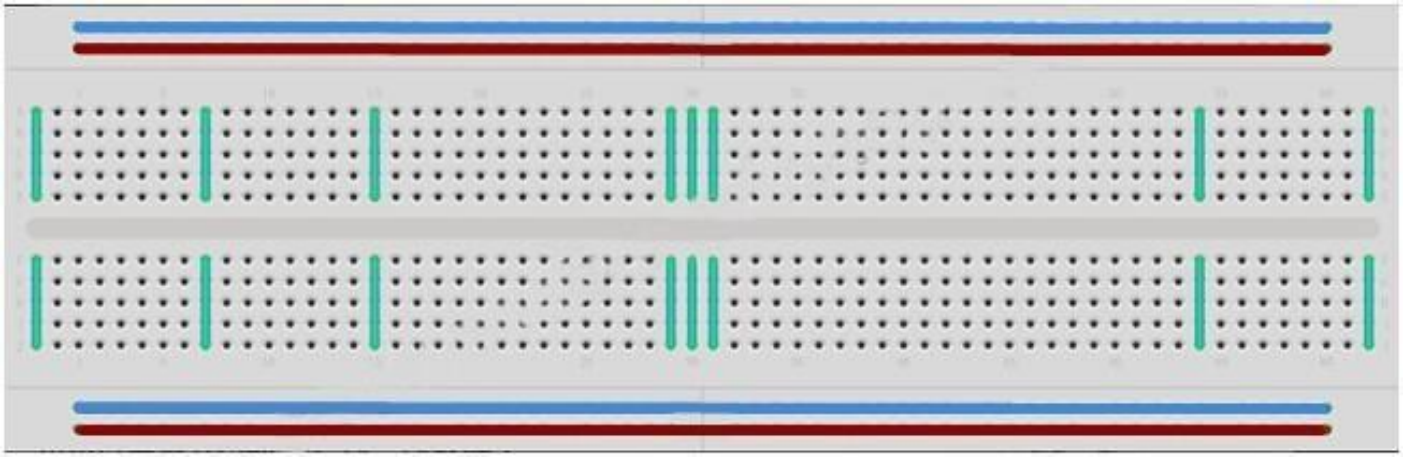
(1) x 10k ohmresistor

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

Component Introduction

BREADBOARD MB-102：

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



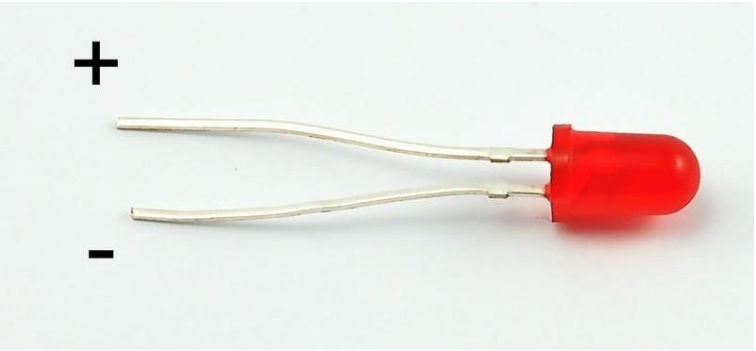
Breadboards come in various sizes and configurations. The simplest kind is just a grid of holes in a plastic block. Inside are strips of metal that provide electrical connection between holes in the shorter rows. Pushing the legs of two different components into the same row joins them together electrically.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

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

LED:

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

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

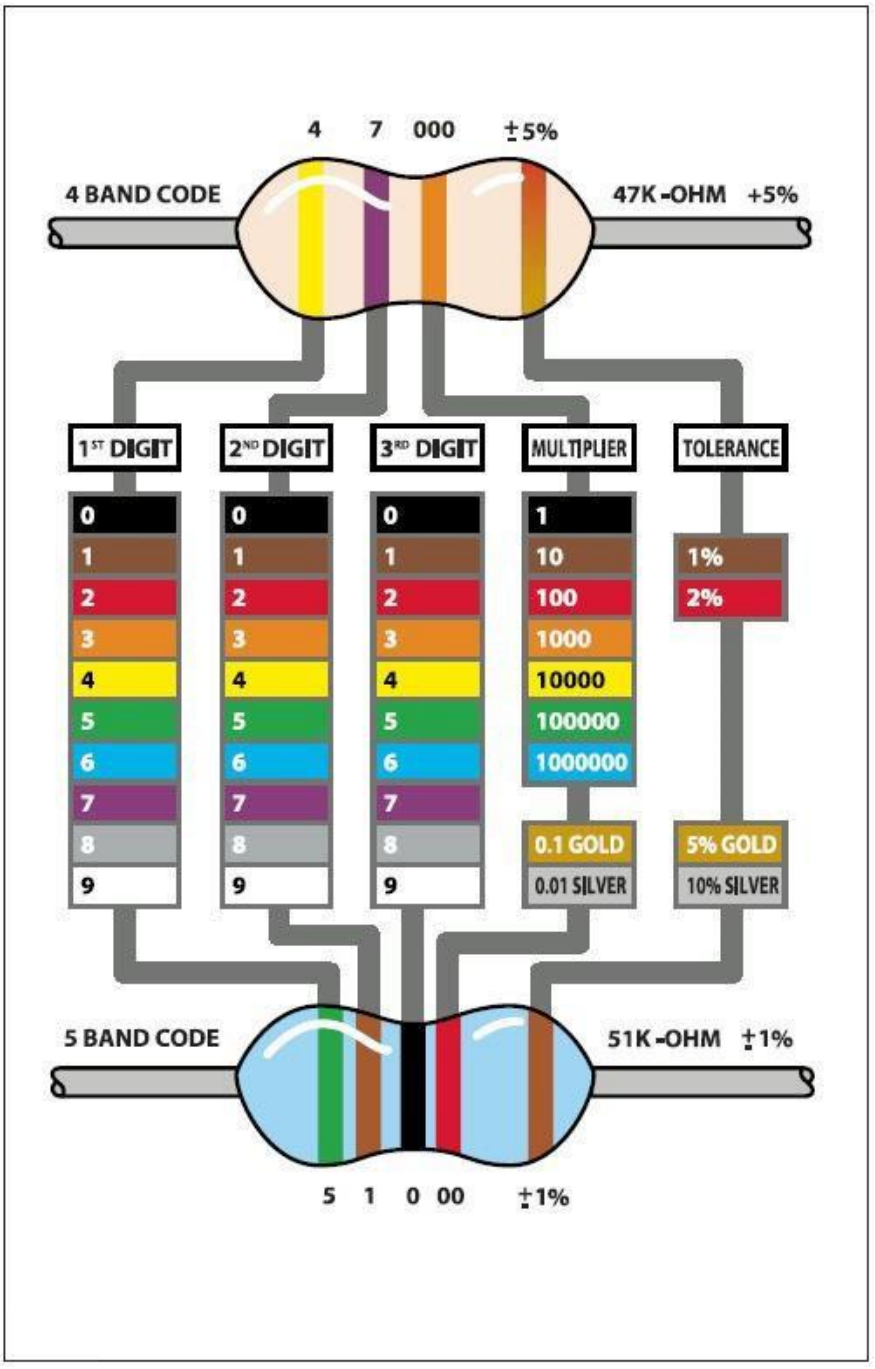


If you do not use a resistor with an LED, then it may well be destroyed almost immediately, as too much current will flow through, heating it and destroying the 'junction' where the light is produced.
There are two ways to tell which is the positive lead of the LED and which the negative.
Firstly, the positive lead is longer.
Secondly, where the negative lead enters the body of the LED, there is a flat edge to the case of the LED.
If you happen to have an LED that has a flat side next to the longer lead, you should assume that the longer lead is positive.

RESISTORS:

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

But first, more about resistors...
The unit of resistance is called the Ohm, which is usually shortened to Ω the Greek letter Omega. Because an Ohm is a low value of resistance (it doesn't resist much at all), we also denote the values of resistors in kΩ (1,000 Ω) and MΩ (1,000,000 Ω). These are called kilo-ohms and mega-ohms.
In this lesson, we are going to use three different values of resistor: 220Ω, 1kΩ and 10kΩ. These resistors all look the same, except that they have different colored stripes on them. These stripes tell you the value of the resistor.
The resistor color code has three colored stripes and then a gold stripe at one end.

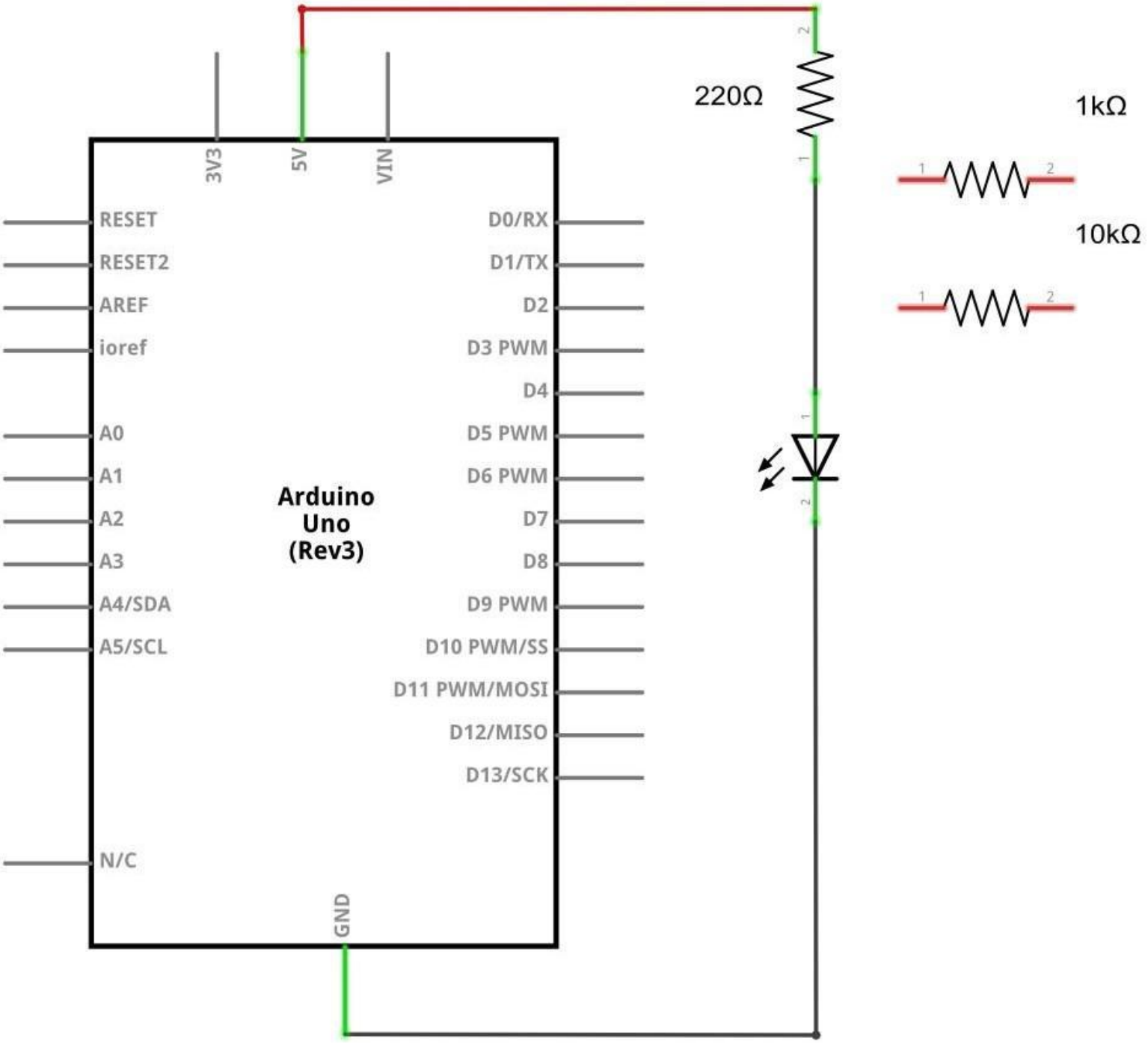


Unlike LEDs, resistors do not have a positive and negative lead. They can be connected either wayaround.

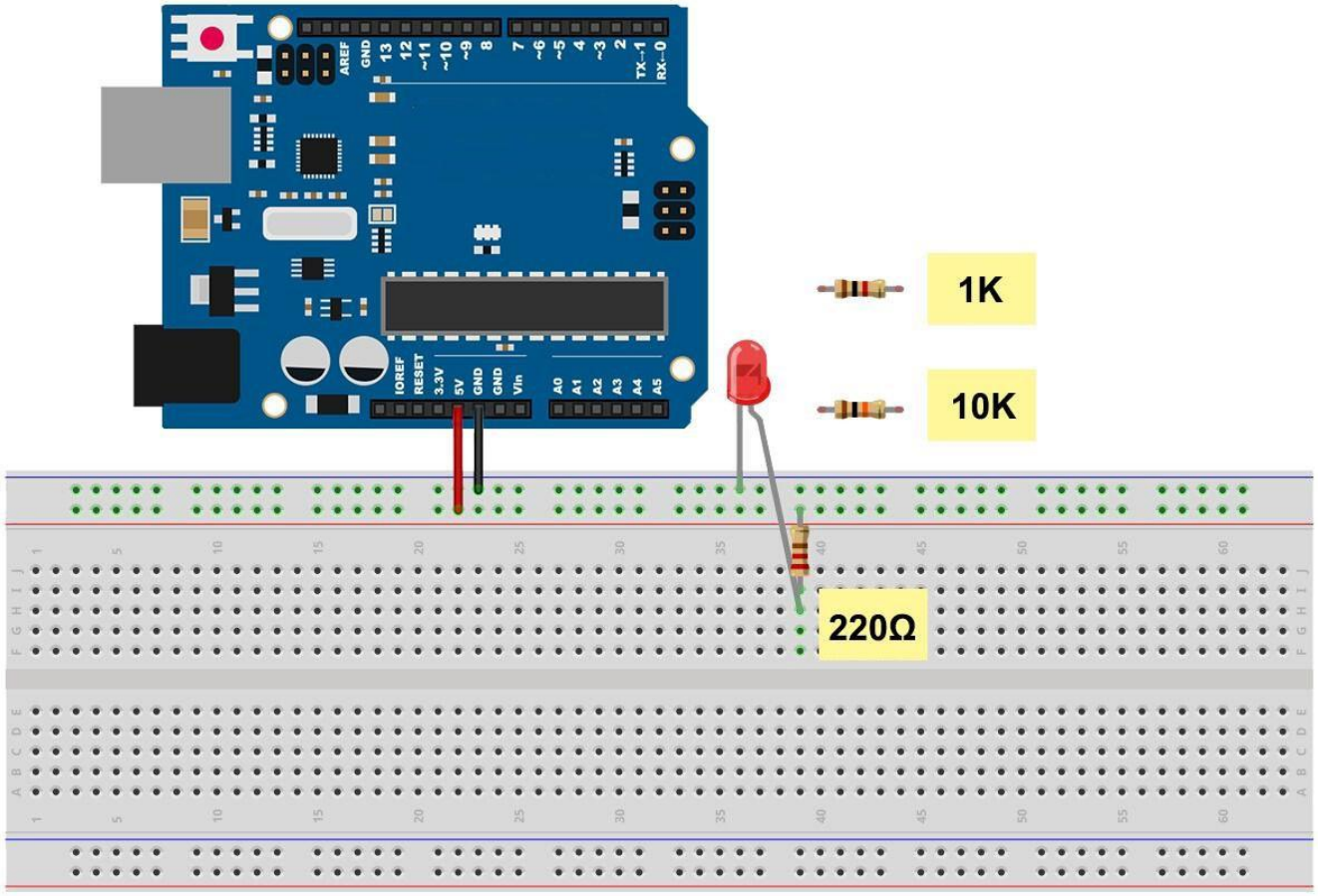
If you find this approach method too complicated, you can read the color ring flag on our resistors directly to determine its resistance value. Or you may use a digital multimeter instead.

Connection

Schematic



Wiring diagram



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

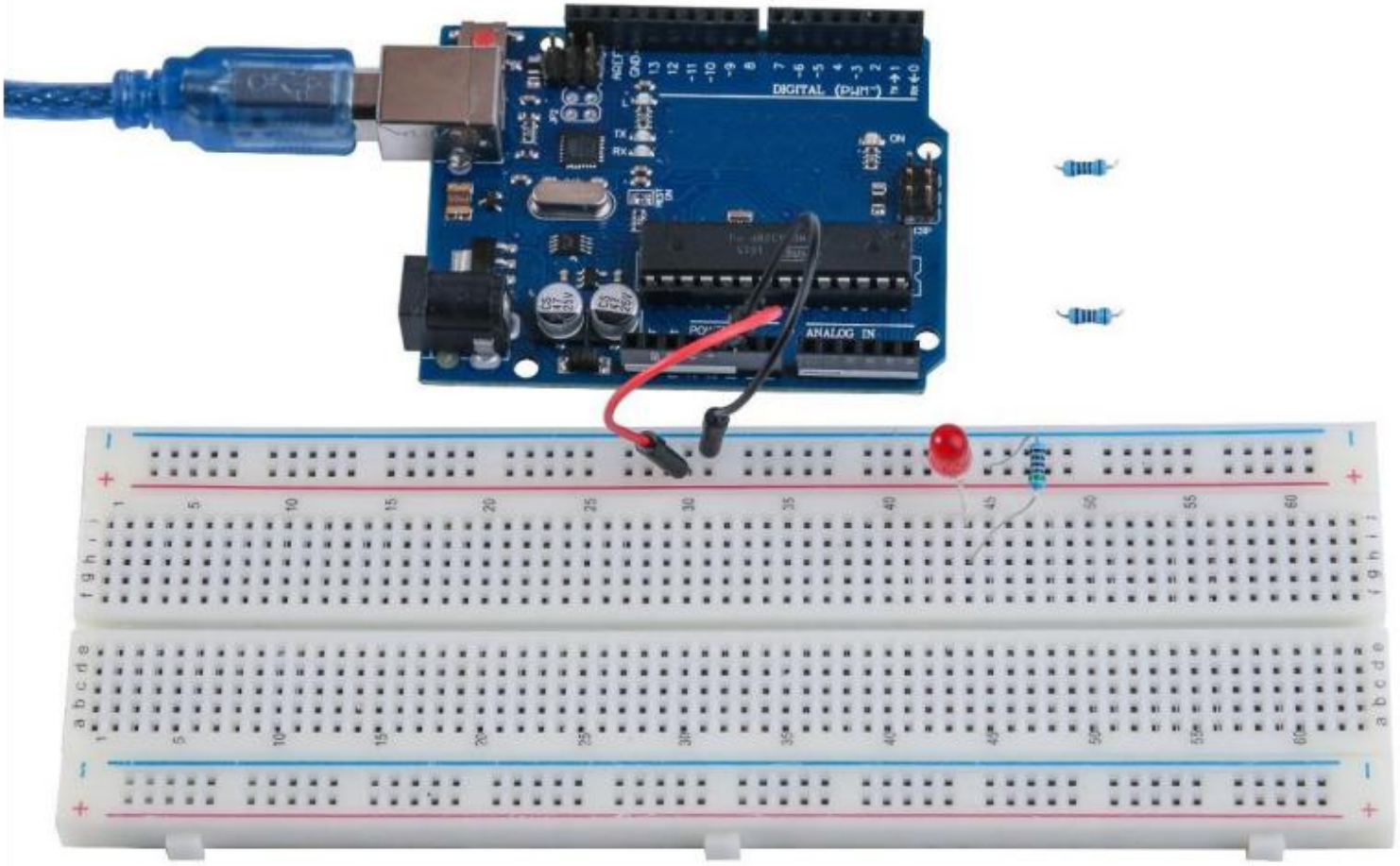
With the 220 Ω resistor in place, the LED should be quite bright. If you swap out the 220 Ω resistor for the 1kΩ resistor, then the LED will appear a little dimmer. Finally, with the 10 kΩ resistor in place, the LED will be just about visible. Pull the red jumper lead out of the breadboard and touch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

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

You will probably want to put the 220Ω resistor back in place.

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

Example picture



Lesson 4 RGB LED

Overview

RGB LEDs are a fun and easy way to add some color to your projects. Since they are like 3 regular LEDs in one, how to use and connect them is not much different.
They come mostly in 2 versions: Common Anode or Common Cathode.
Common Anode uses 5V on the common pin, while Common Cathode connects to ground.
As with any LED, we need to connect some resistors inline (3 total) so we can limit the current being drawn.
In our sketch, we will start with the LED in the Red color state, then fade to Green, then fade to Blue and finally back to the Red color. By doing this we will cycle through most of the color that can be achieved.

Component Required:

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

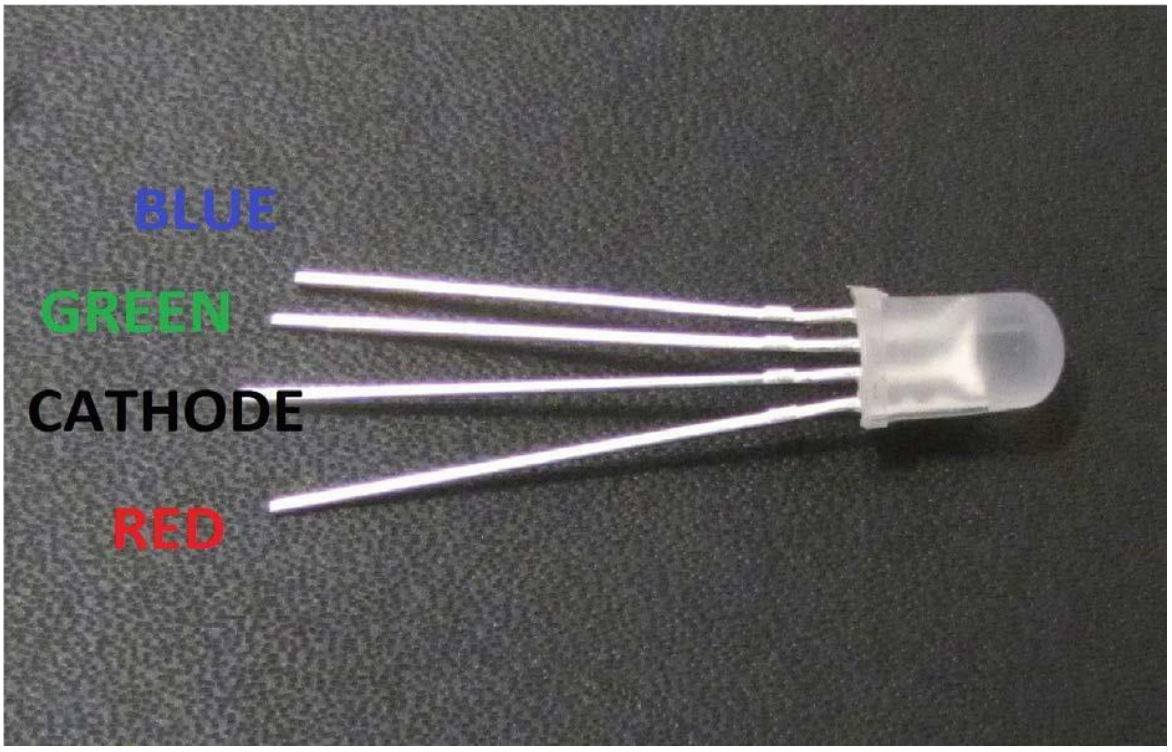
Component Introduction

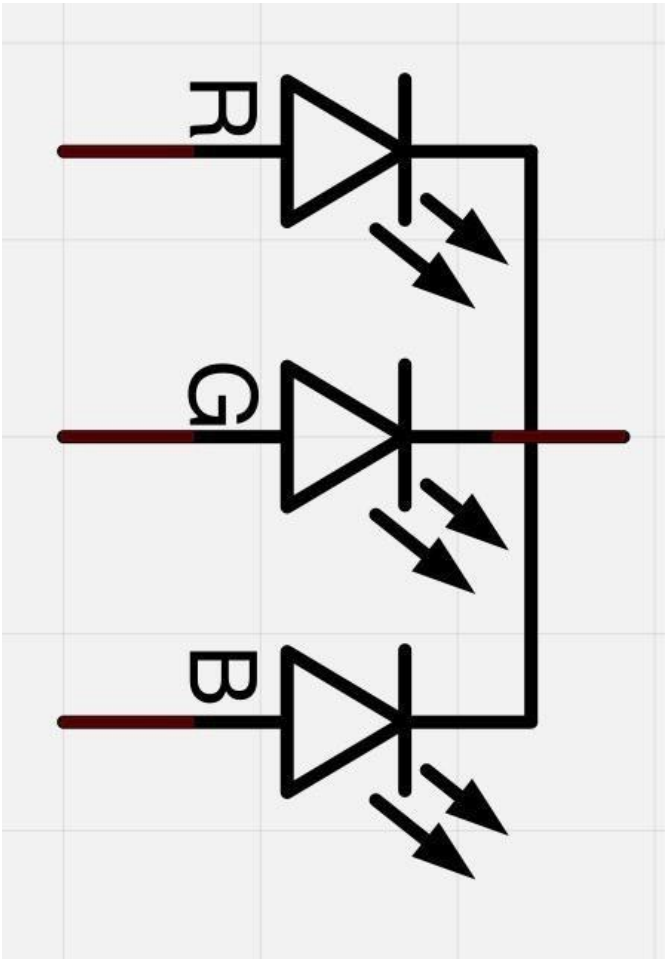
RGB:

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

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

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





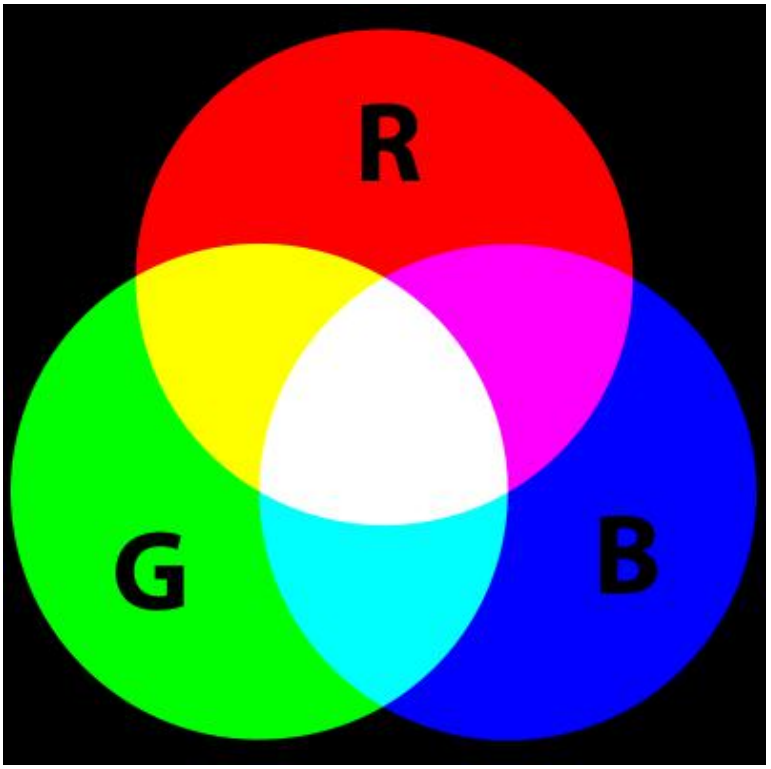
Here on the photographs you can see 4 electrode LED. Every separate pin for Green or Blue or Red color is called Anode. You will always connect “+” to it. Cathode goes to “-“(ground). If you connect it other way round the LED will not light.

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

COLOR:

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

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



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

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

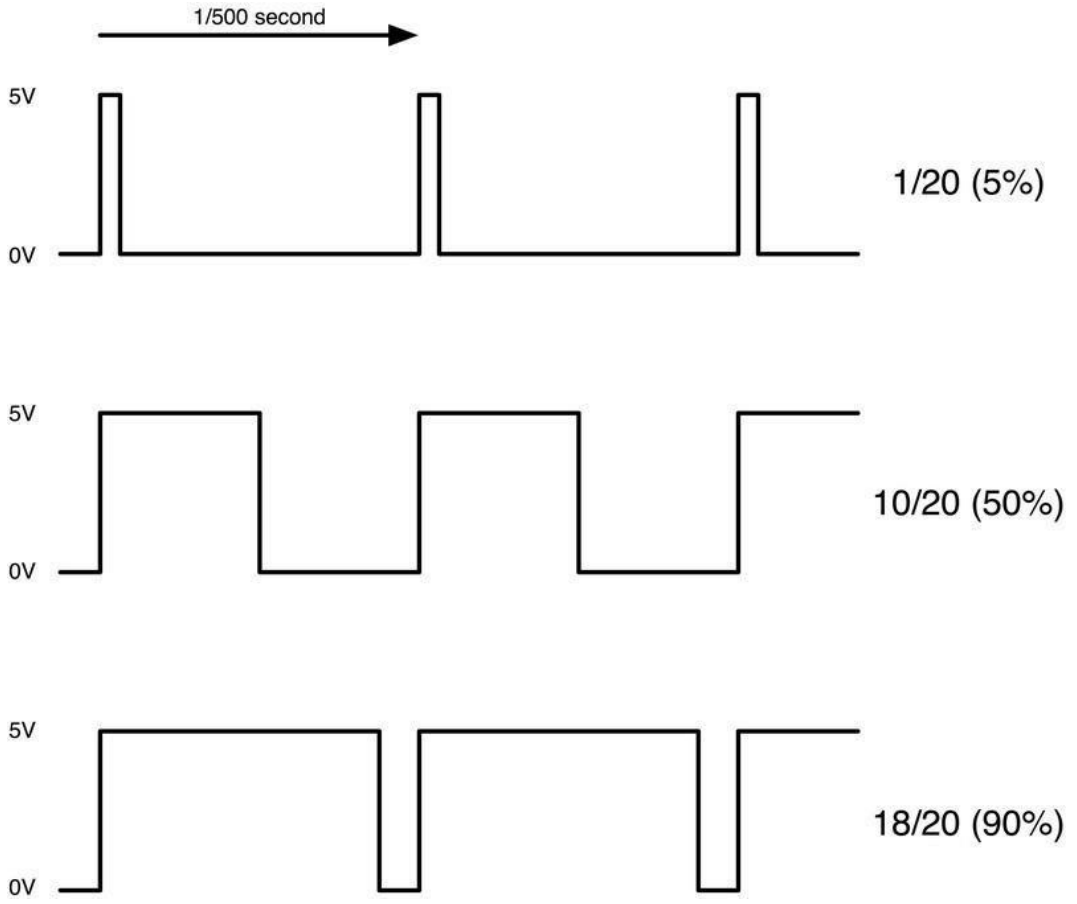
Black is not so much a color as an absence of light. Therefore, the closest we can come to black with our LED is to turn off all three colors.

Theory (PWM)

Pulse Width Modulation (PWM) is a technique for controlling power.

We also use it here to control the brightness of each of the LEDs.

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

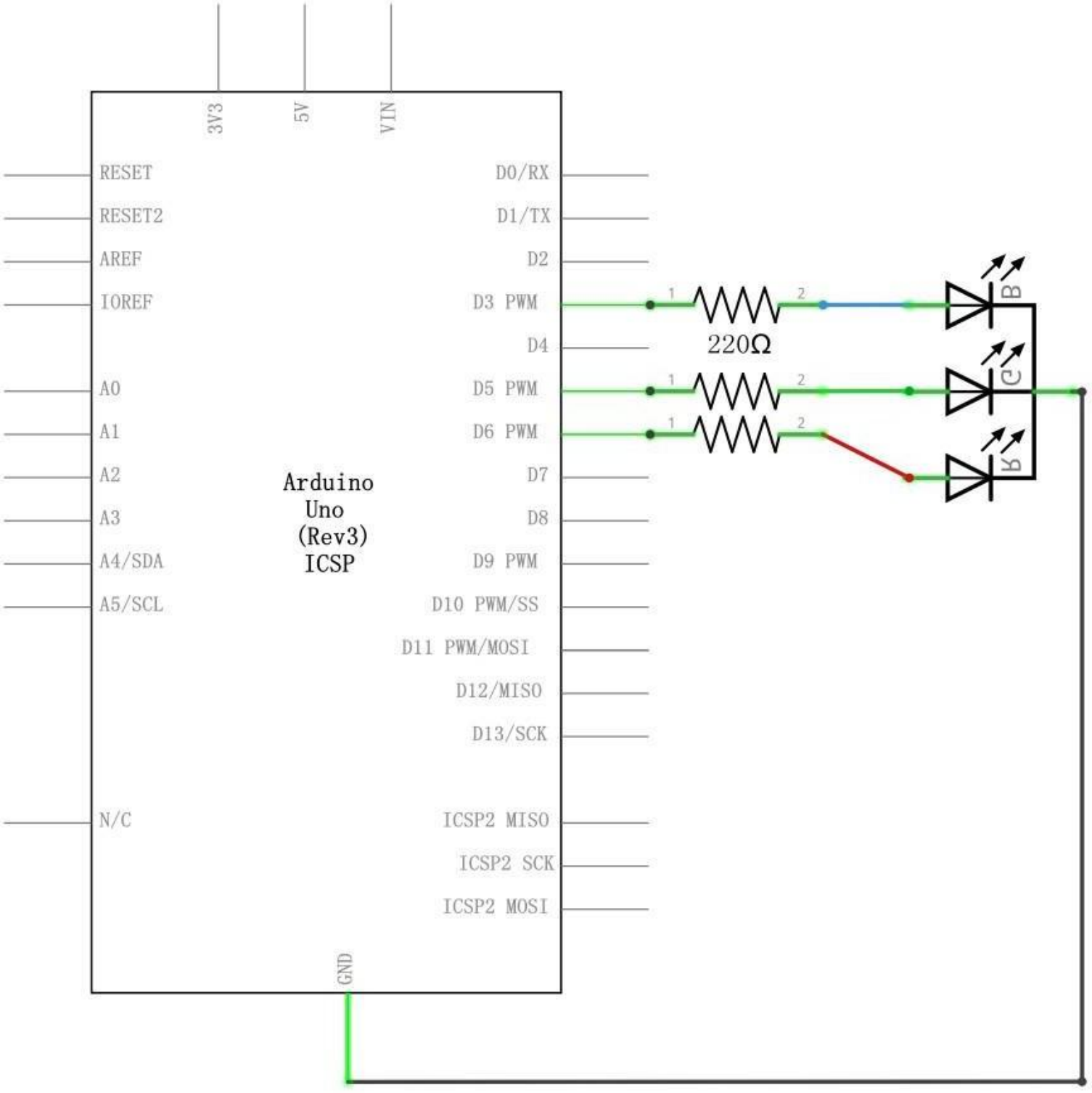


Roughly every 1/500 of a second, the PWM output will produce a pulse. The length of this pulse is controlled by the 'analog Write' function. So 'analog Write(0)' will not produce any pulse at all and 'analog Write(255)' will produce a pulse that lasts all the way until the next pulse is due, so that the output is actually on all the time. If we specify a value in the analog Write that is somewhere in between 0 and 255, then we will produce a pulse. If the output pulse is only high for 5% of the time, then whatever we are driving will only receive 5% of full power.

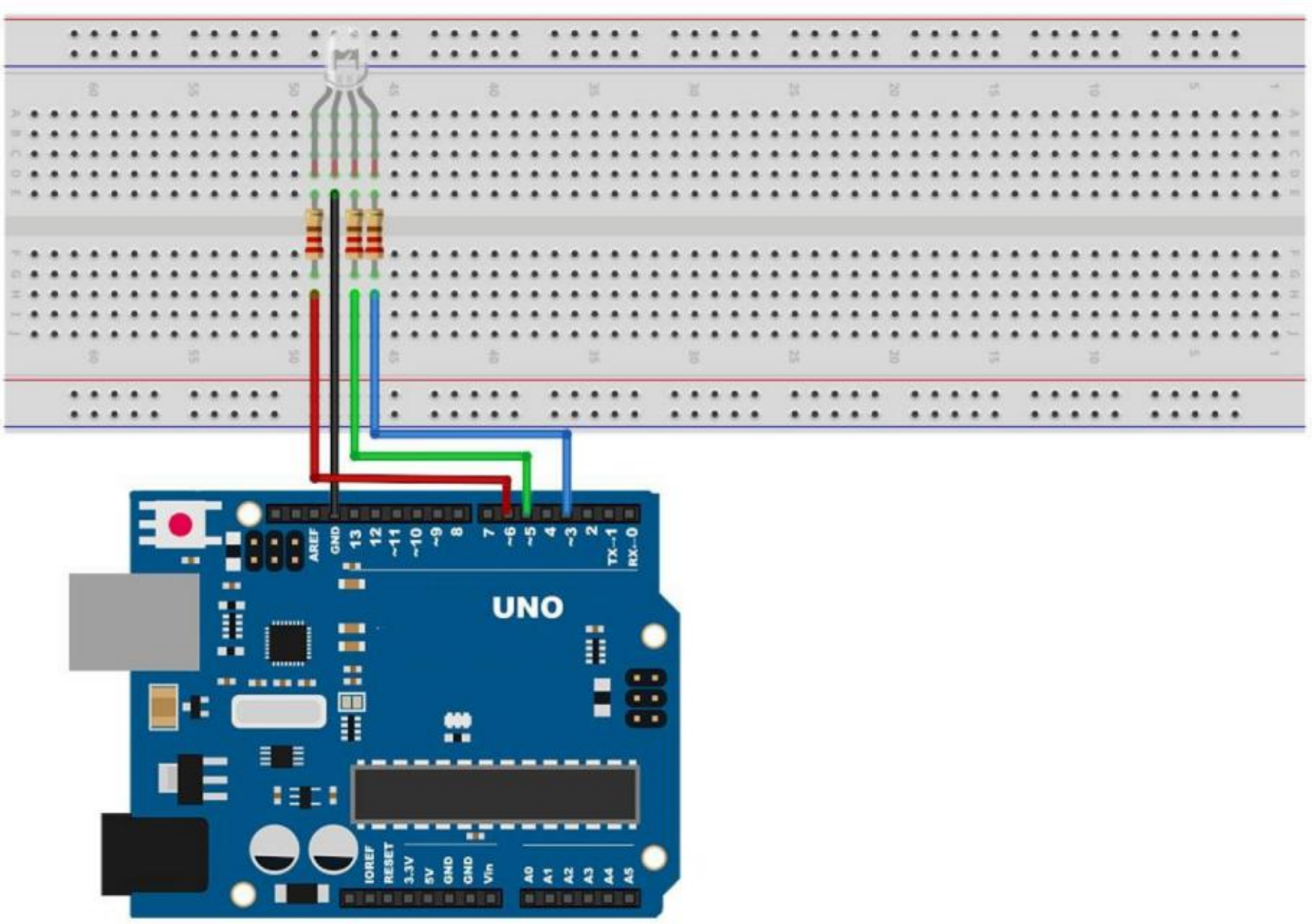
If, however, the output is at 5V for 90% of the time, then the load will get 90% of the power delivered to it. We cannot see the LEDs turning on and off at that speed, so to us, it just looks like the brightness is changing.

Connection

Schematic



Wiring diagram



Code

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

Our code will use FOR loops to cycle through the colors.
The first FOR loop will go from RED to GREEN.
The second FOR loop will go from GREEN to BLUE.
The last FOR loop will go from BLUE to RED.
Try the sketch out and then we will dissect it in some detail.....

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

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

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

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

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

```
redValue=255;//chooseavaluebetween1and255tochangethecolor.
greenValue = 0;
```

blueValue = 0;

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

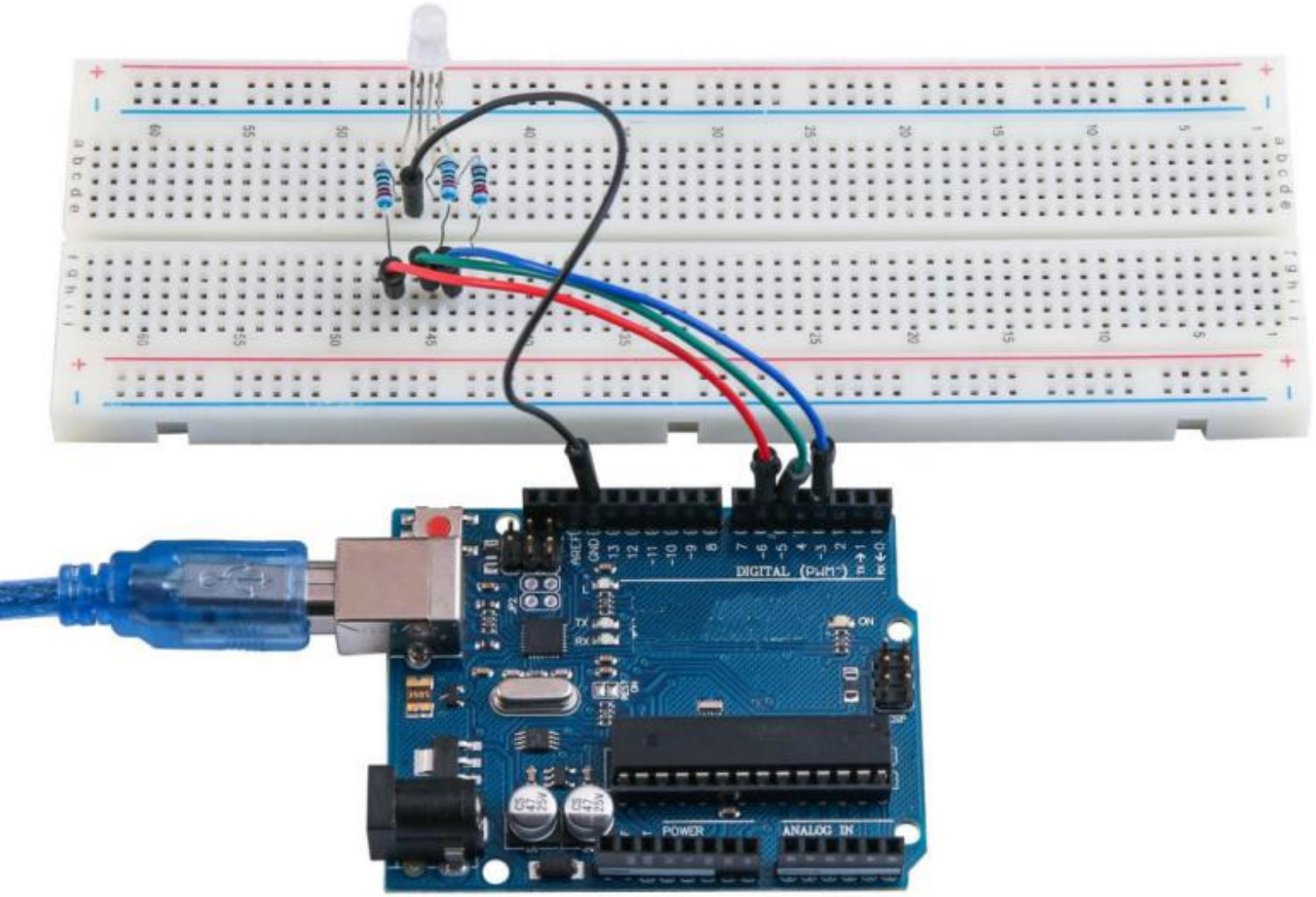
If you look at the 'loop' function you can see that we are setting the amount of red, green and blue light that we want to display and then pausing for a second before moving on to the next color.

#define delayTime 10 // fading time between colors

Delay(delayTime);

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

Example picture



Lesson 5 Digital Inputs

Overview

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

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

Component Required:

(1) x Uno R3

(1) x 830 Tie-points Breadboard

(1) x 5mm red LED

(1) x 220 ohm resistor

(2) x push switches

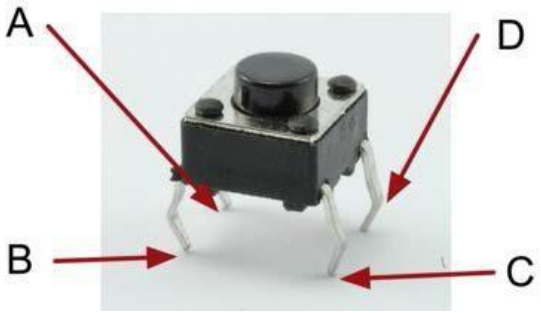
(7) x M-M wires (Male to Male jumper wires)

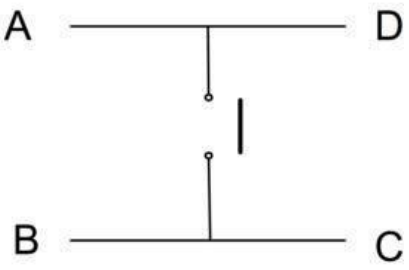
Component Introduction

PUSH SWITCHES:

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

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

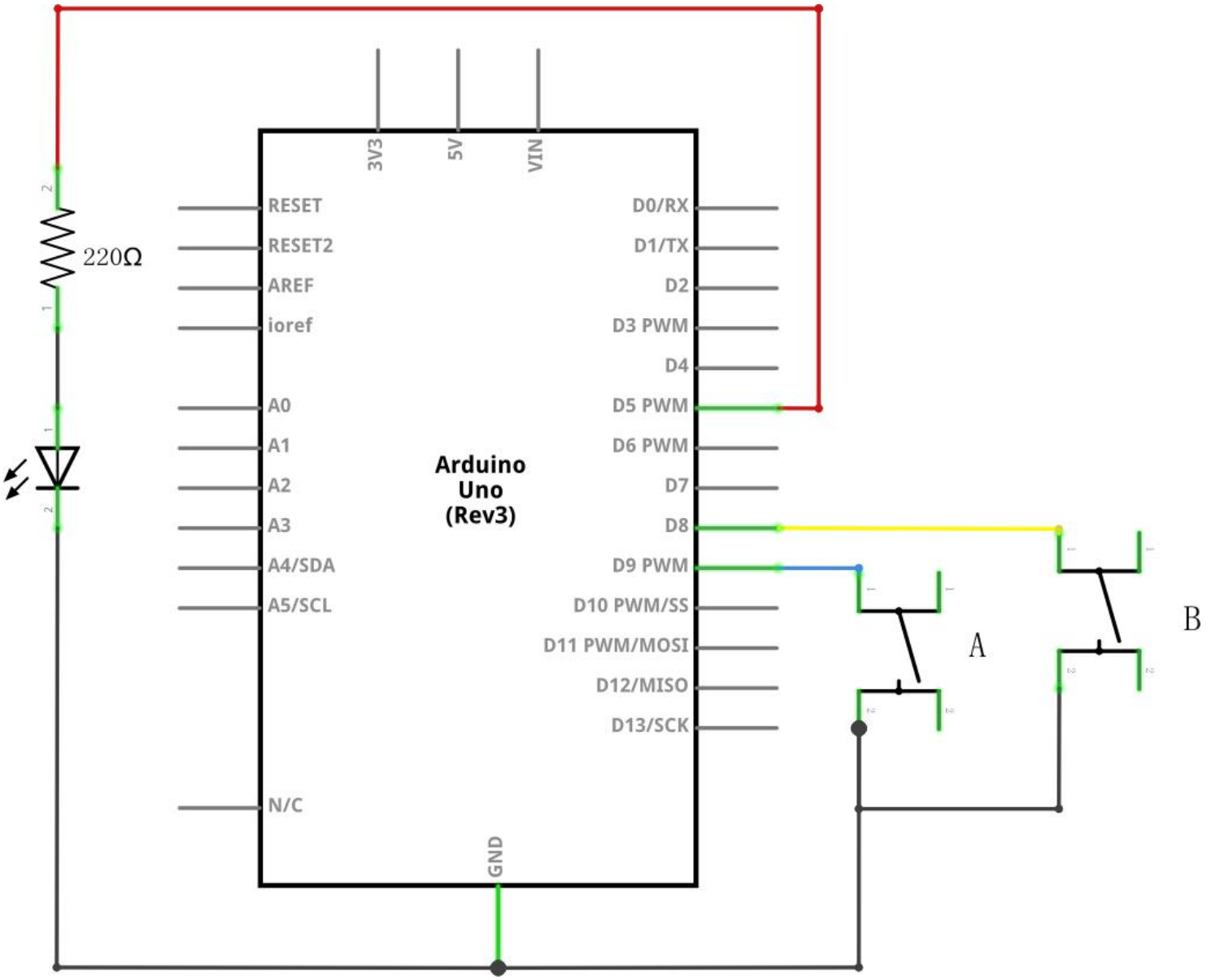




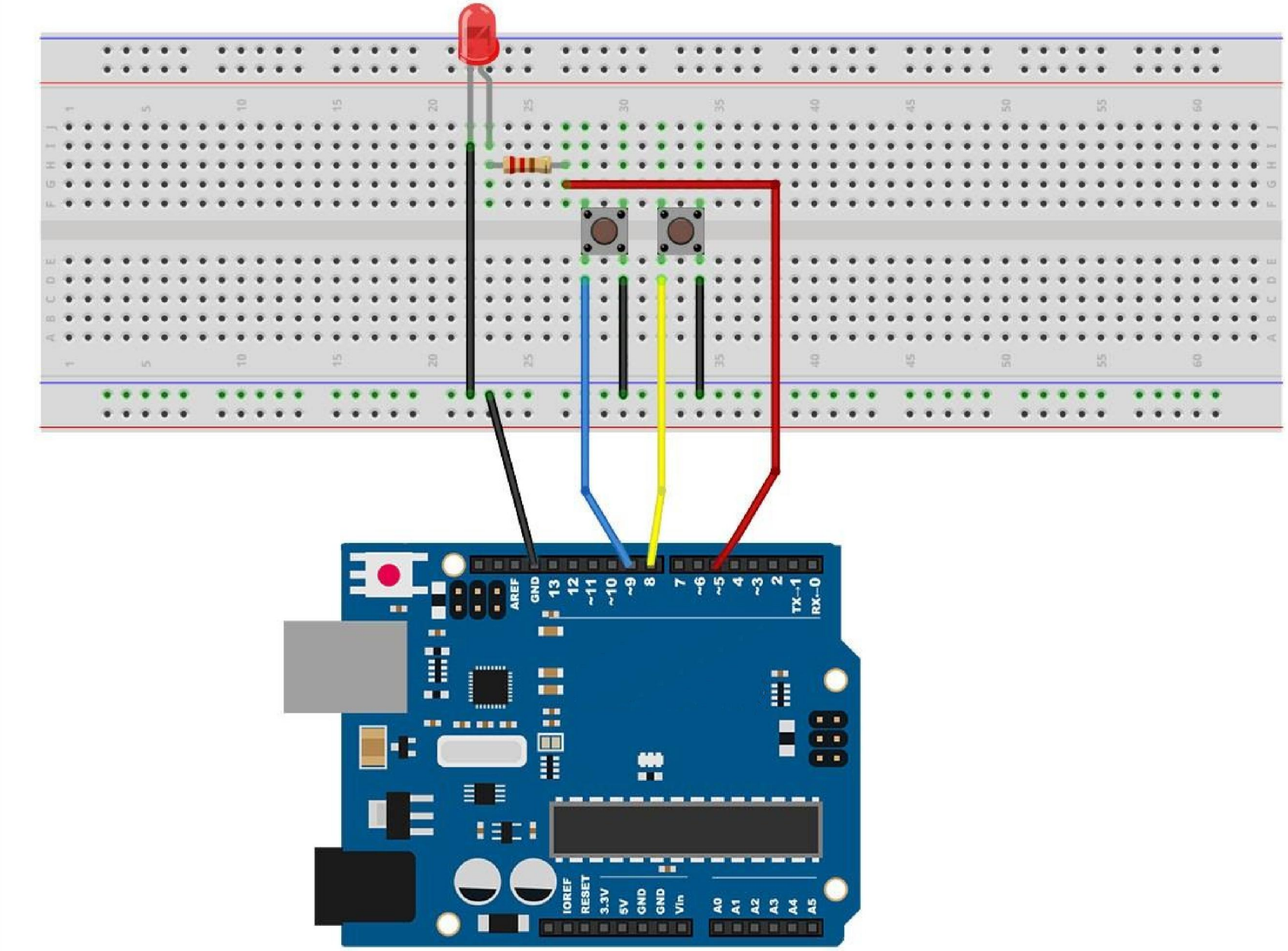
Actually, there are only really two electrical connections. Inside the switch package, pins B and C are connected together, as are A and D.

Connection

Schematic



Wiring diagram



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

Code

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

Load the sketch onto your UNO board. Pressing the left button will turn the LED on while pressing the right button will turn it off.
The first part of the sketch defines three variables for the three pins that are to be used. The 'ledPin' is the output pin and 'buttonApin' will refer to the switch nearer the top of the breadboard and 'buttonBpin' to the other switch.
The 'setup' function defines the ledPin as being an OUTPUT as normal, but now we have the two inputs to deal with. In this case, we use the set the pinMode to be 'INPUT_PULLUP' like this:

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

The pin mode of INPUT_PULLUP means that the pin is to be used as an input, but that if nothing else is connected to the input, it should be 'pulled up' to HIGH. In other words, the default value for the input is HIGH, unless it is pulled LOW by the action of pressing the button.
This is why the switches are connected to GND. When a switch is pressed, it connects the input pin to GND, so that it is no longer HIGH.
Since the input is normally HIGH and only goes LOW when the button is pressed, the logic is a little upside down. We will handle this in the 'loop' function.

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

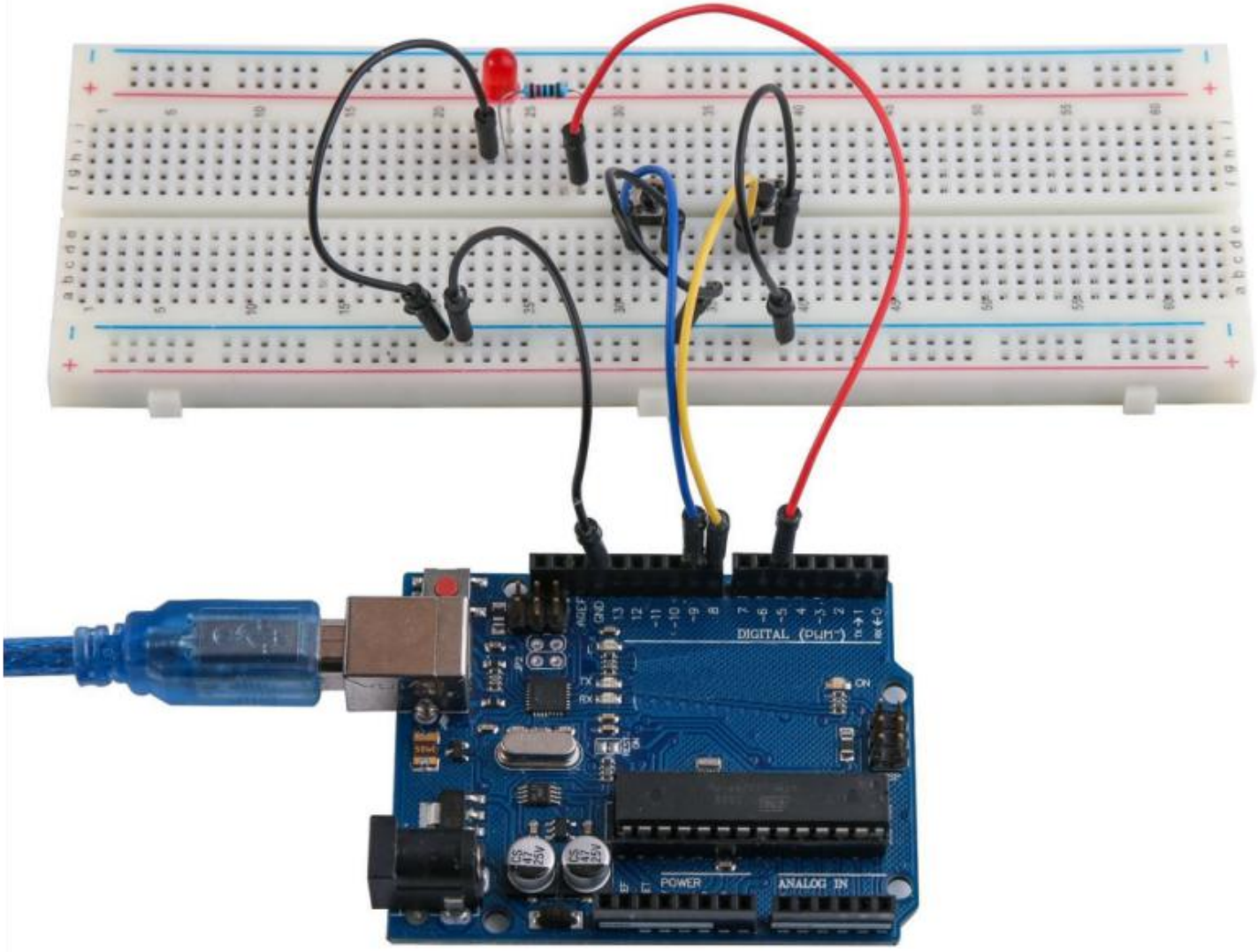
```
  {
    digitalWrite(ledPin, LOW);
  }
}
```

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

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

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

Example picture



Lesson 6 Active buzzer

Overview

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

Component Required:

(1) x Uno R3

(1) x Active buzzer

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

Component Introduction

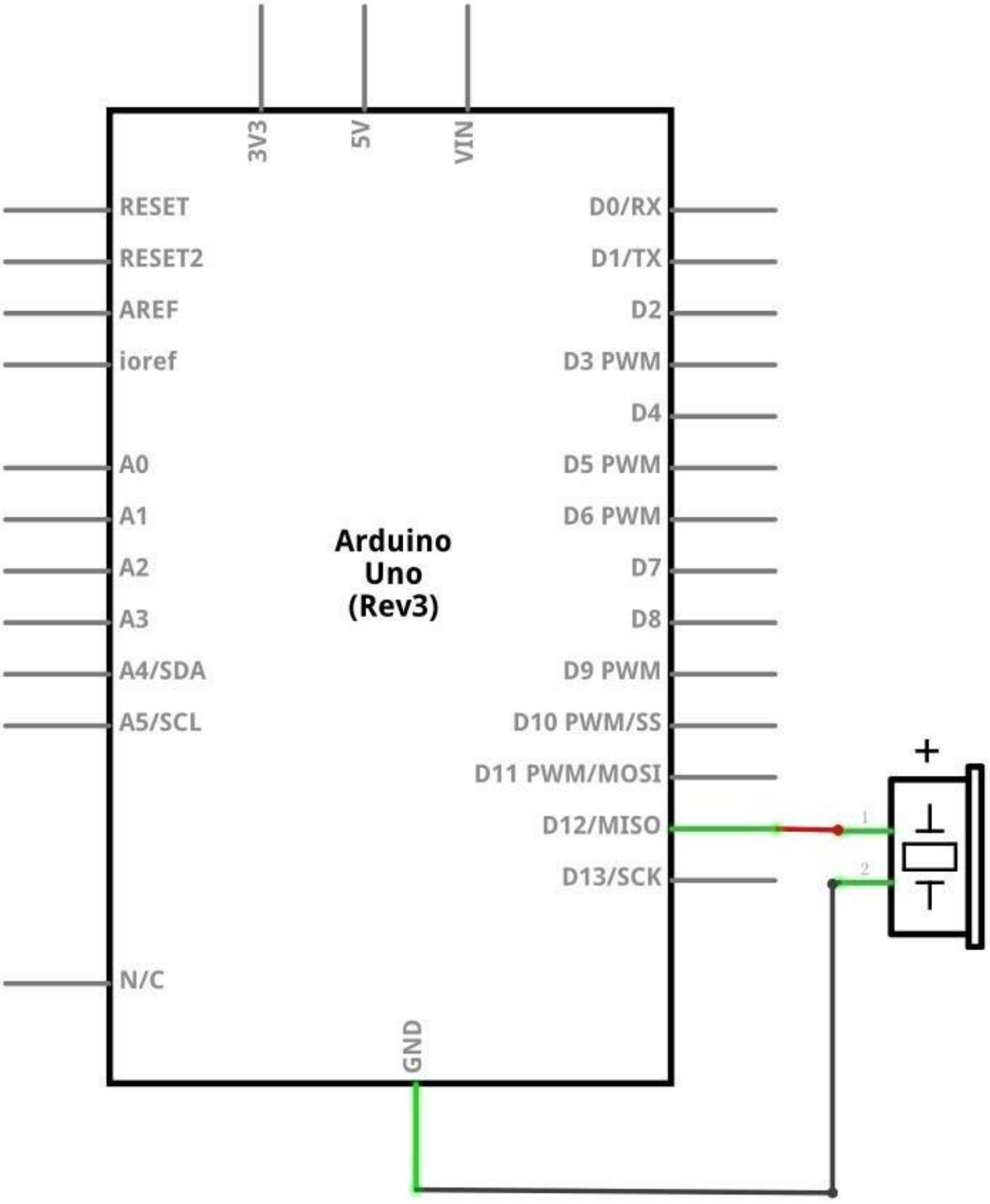
BUZZER:

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

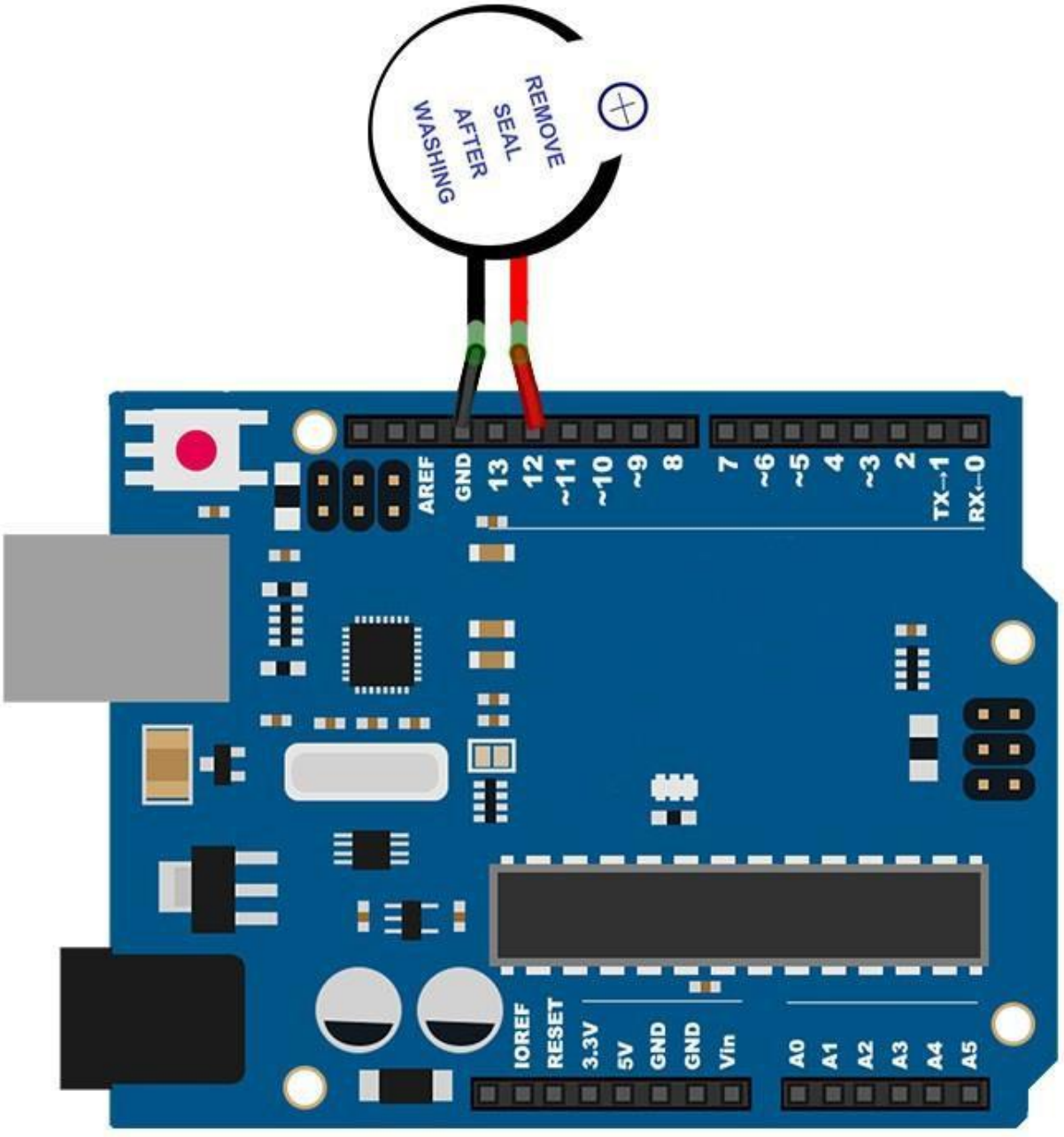
The difference between the two is that an active buzzer has a built-in oscillating source, so it will generate a sound when electrified. A passive buzzer does not have such a source so it will not tweet if DC signals are used; instead, you need to use square waves whose frequency is between 2K and 5K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

Connection

Schematic



Wiring diagram



Code

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

Lesson 7 Passive Buzzer

Overview

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

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

Component Required:

(1) x Uno R3

(1) x Passive buzzer

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

Component Introduction

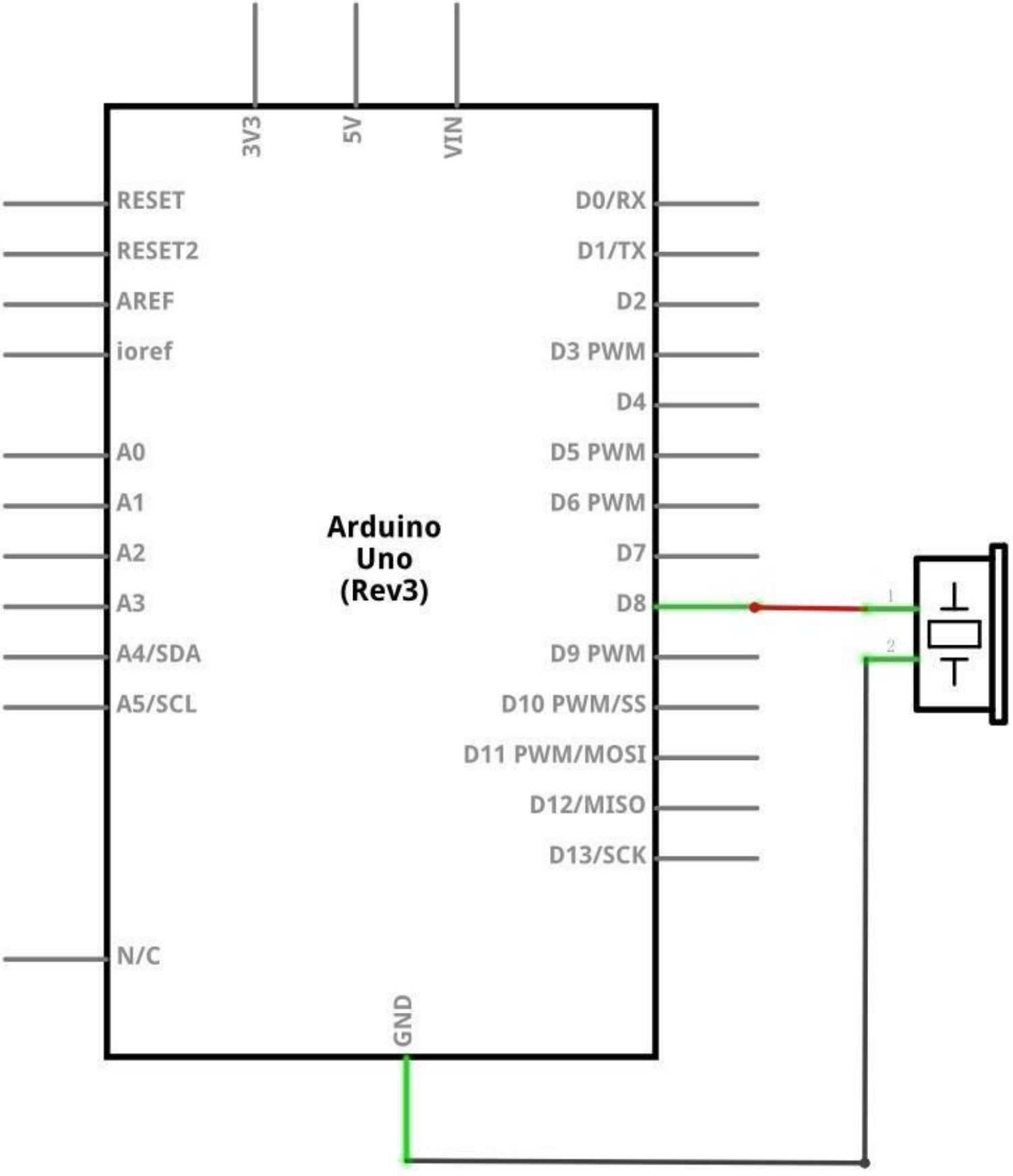
Passive Buzzer:

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

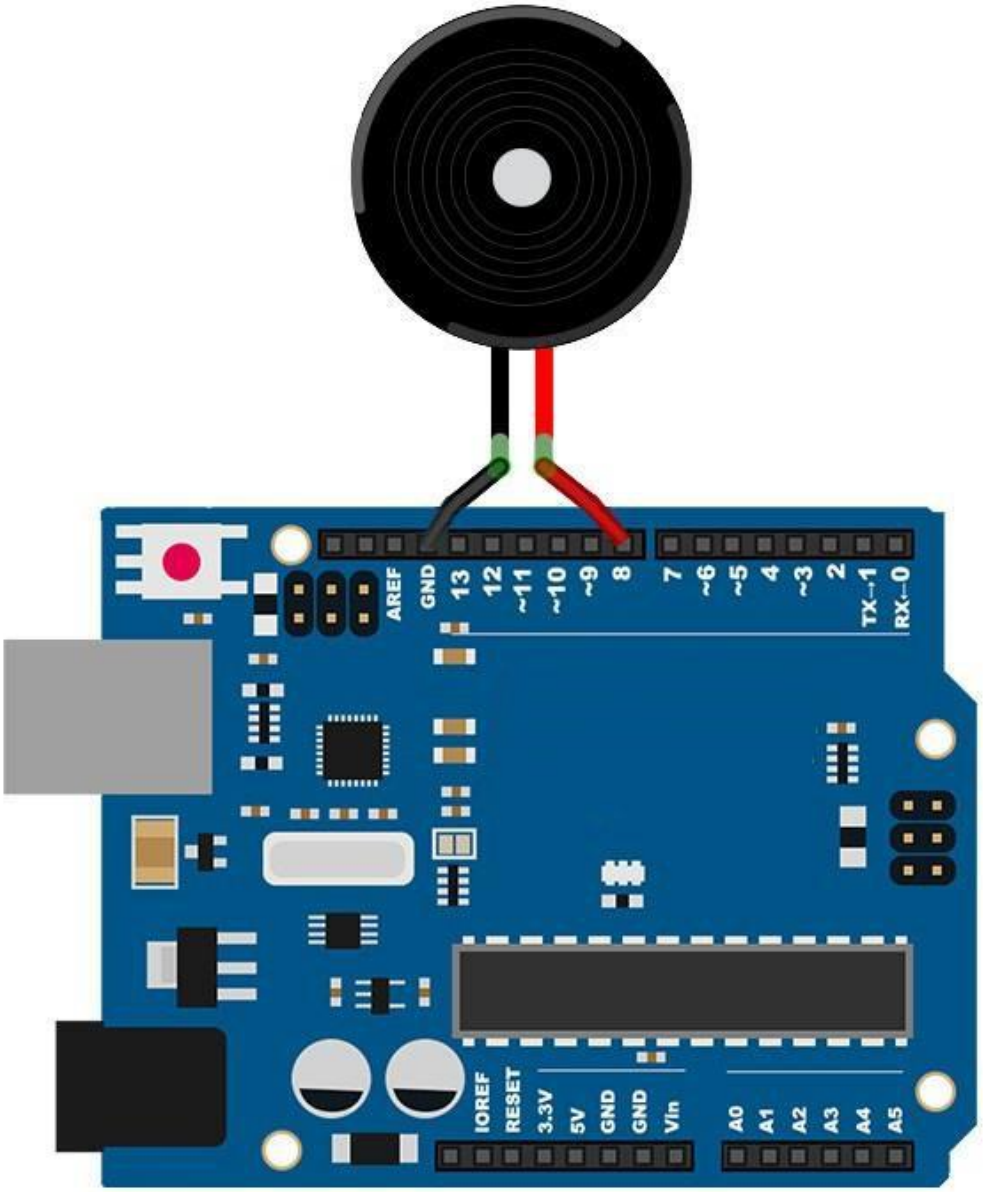
We should be careful not to use the UNO R3 board analog Write () function to generate a pulse to the buzzer, because the pulse output of analog Write () is fixed (500Hz).

Connection

Schematic



Wiring diagram



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

Code

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

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

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

Lesson 8 Tilt Ball Switch

Overview

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

Component Required:

(1) x Uno R3

(1) x Tilt Ball switch

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

Component Introduction

Tilt sensor:

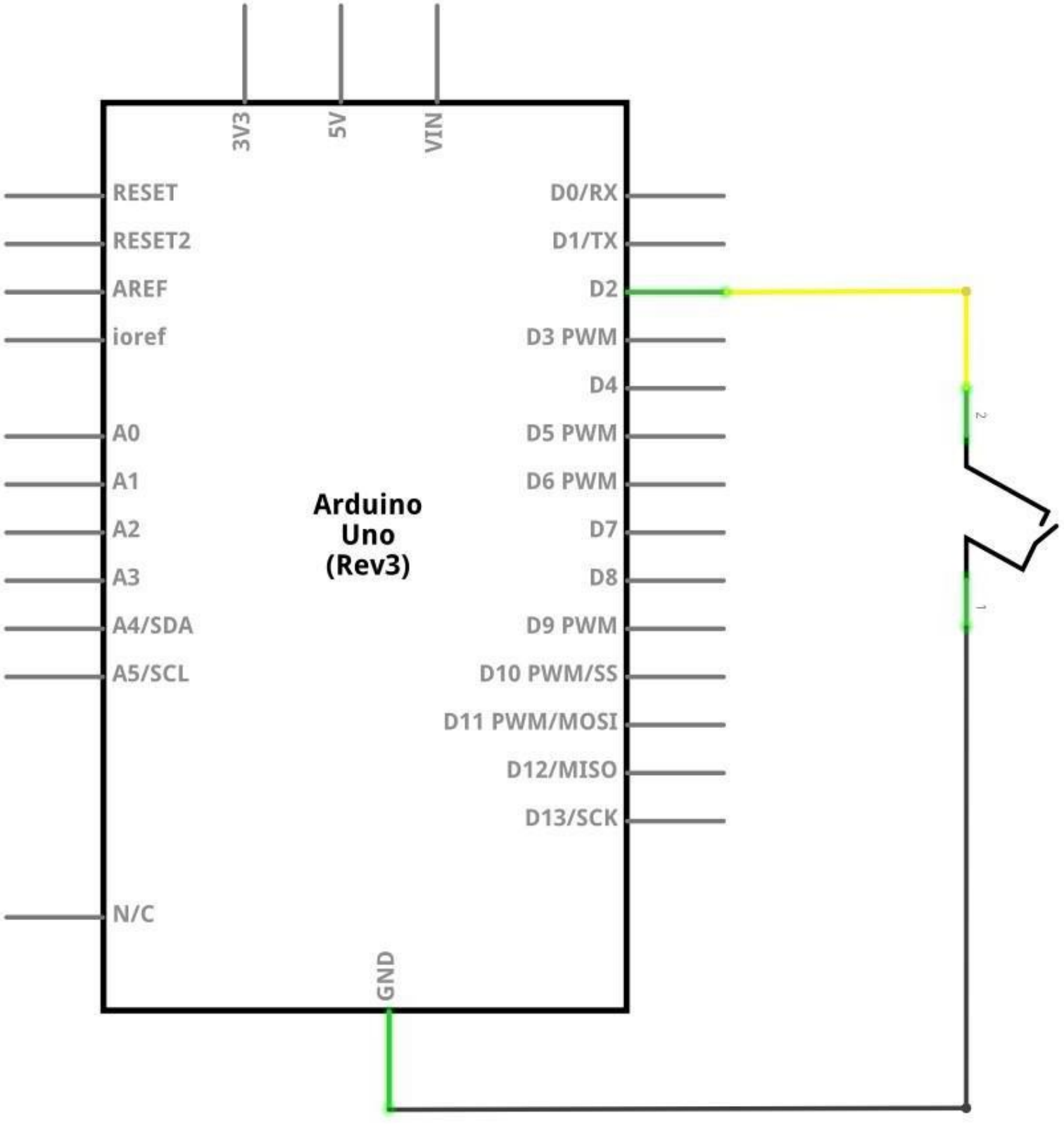
Tilt sensors (tilt ball switch) allow you to detect orientation or inclination. They are small, inexpensive, low-power and easy-to-use. If used properly, they will not wear out. Their simplicity makes them popular for toys, gadgets and appliances. Sometimes, they are referred to as "mercury switches",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

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

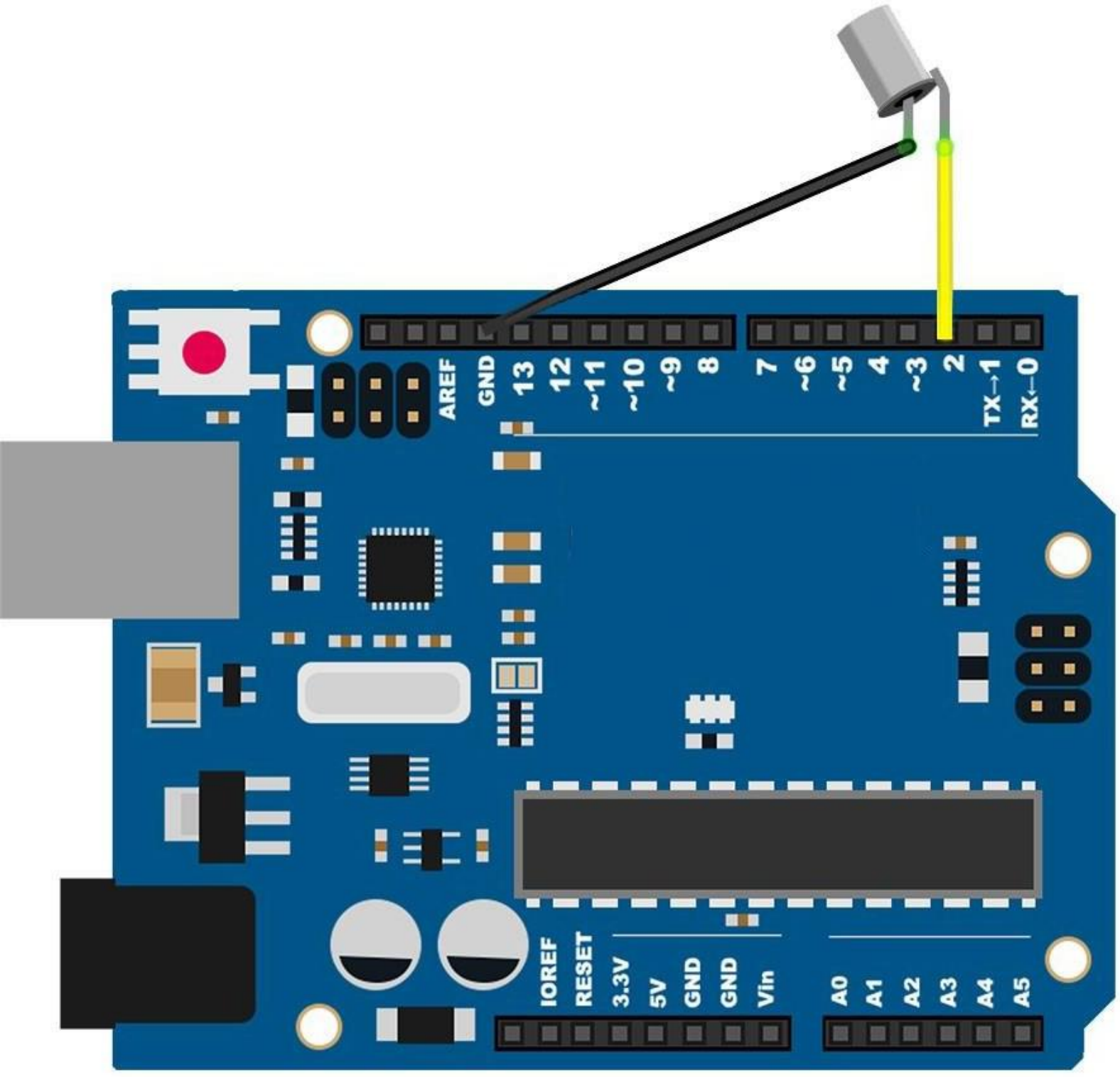
While not as precise or flexible as a full accelerometer, tilt switches can detect motion or orientation. Another benefit is that the big ones can switch power on their own. Accelerometers, on the other hand, output digital or analog voltage that must then be analyzed using extra circuitry.

Connection

Schematic



Wiring diagram



Code

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

Lesson 9 Servo

Overview

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

Component Required:

(1) x Uno R3

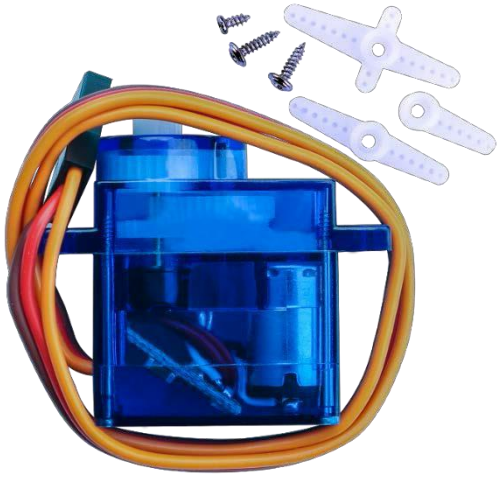
(1) x Servo (SG90)

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

Component Introduction

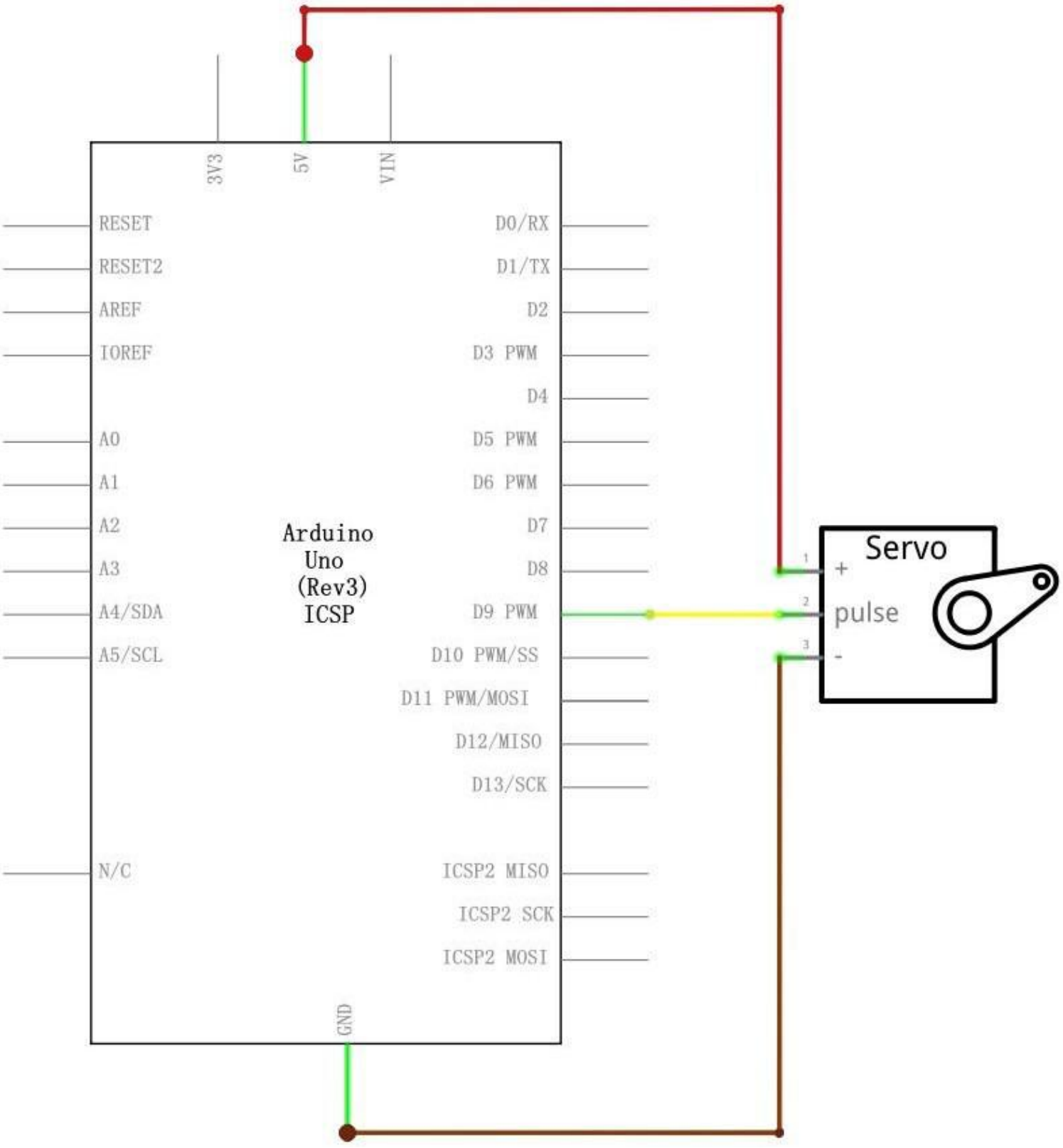
SG90

- Universal for JR and FP connector
- Cable length : 25cm
- No load; Operating speed: 0.12 sec / 60 degree (4.8V), 0.10 sec / 60 degree (6.0V)
- Stall torque (4.8V): 1.6kg/cm
- Temperature : -30~60'C
- Dead band width: 5us
- Working voltage: 3.5~6V
- Dimension : 1.26 in x 1.18 in x 0.47 in (3.2 cm x 3 cm x 1.2 cm)
- Weight : 4.73oz (134 g)

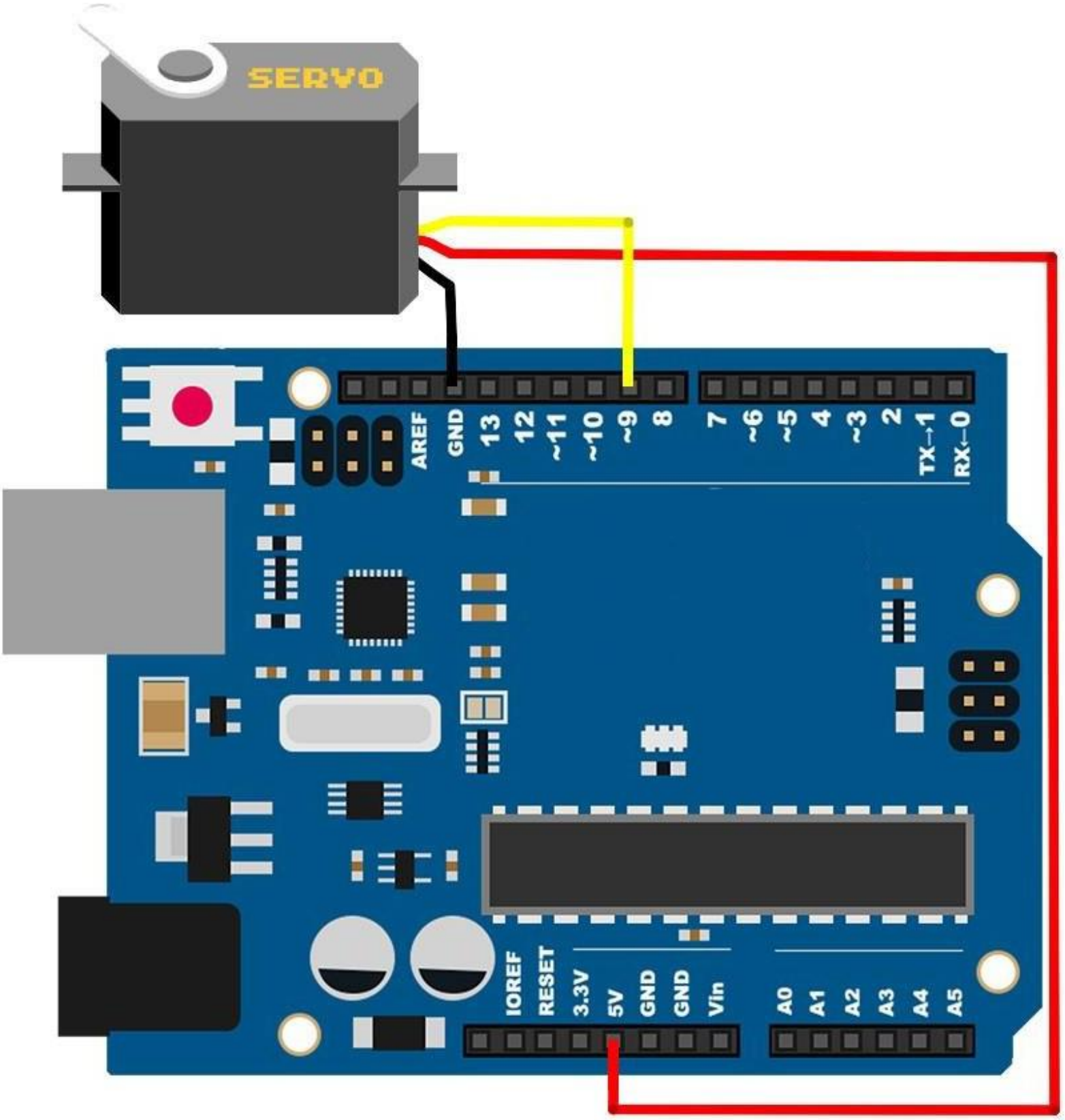


Connection

Schematic



Wiring diagram



Code

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

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

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

Example picture

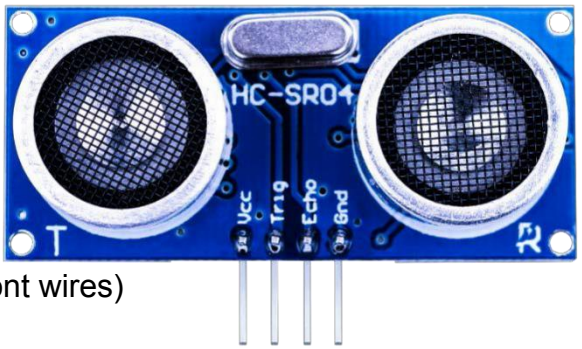
In the picture, the brown wire of servo is adapted via the black M-M wires, the red one is adapted via the red M-M wires, and the orange one is adapted via the yellow M-M wires.

Lesson 10 Ultrasonic Sensor Module

Overview

Ultrasonic sensor is great for all kind of projects that need distance measurements, avoiding obstacles as examples.

The HC-SR04 is inexpensive and easy to use since we will be using a Library specifically designed for these sensor.



Component Required:

(1) x Uno R3

(1) x Ultrasonic sensor module

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

Component Introduction

Ultrasonic sensor

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

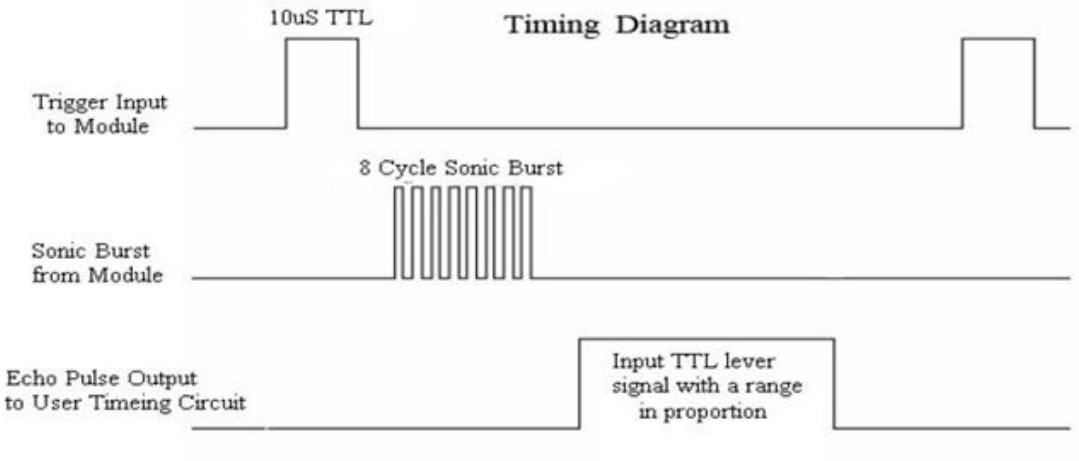
(1) Using IO trigger for at least 10us high level signal,

(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.

(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic tore turning.

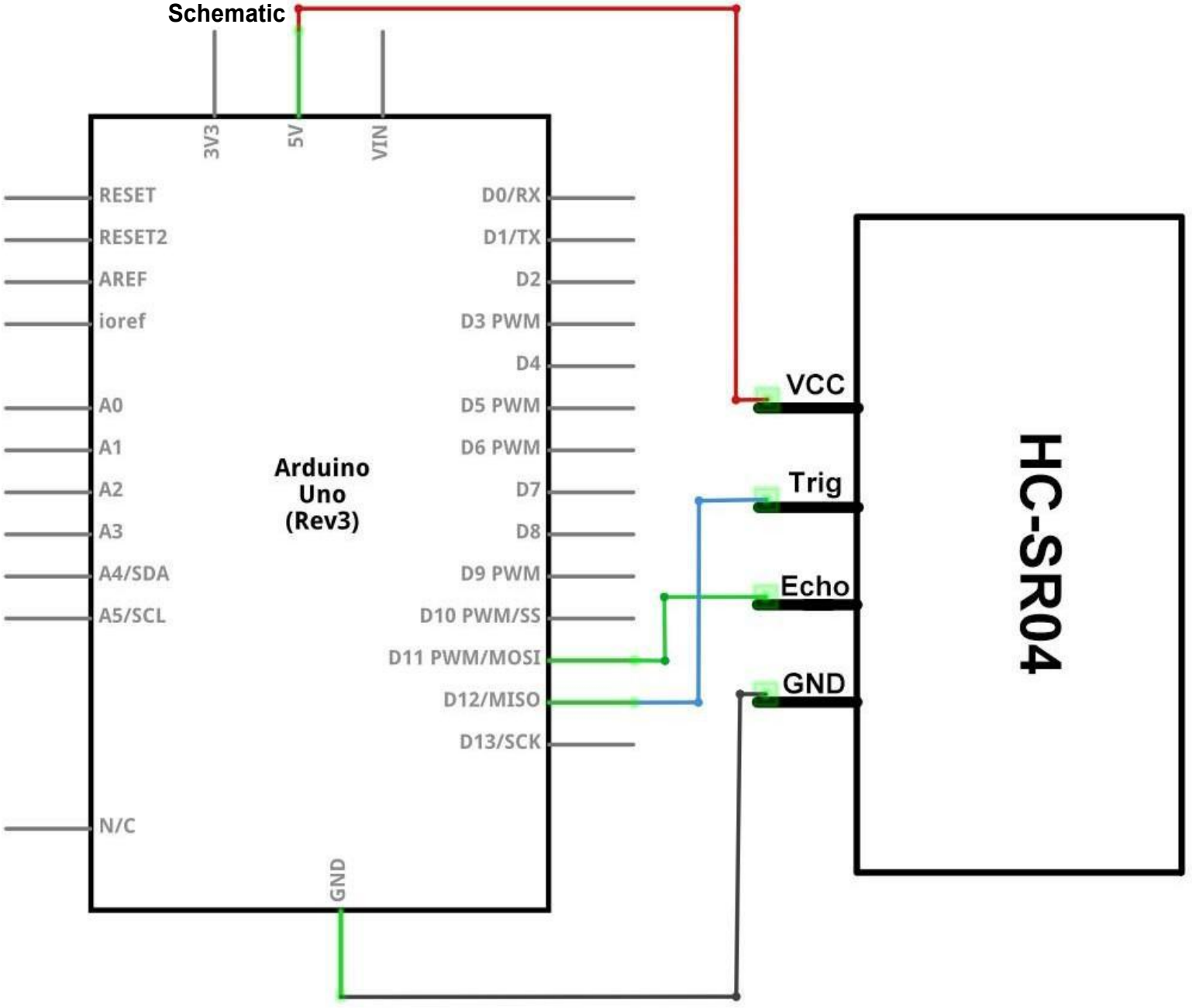
Test distance = (high level time × velocity of sound (340m/s) /2

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

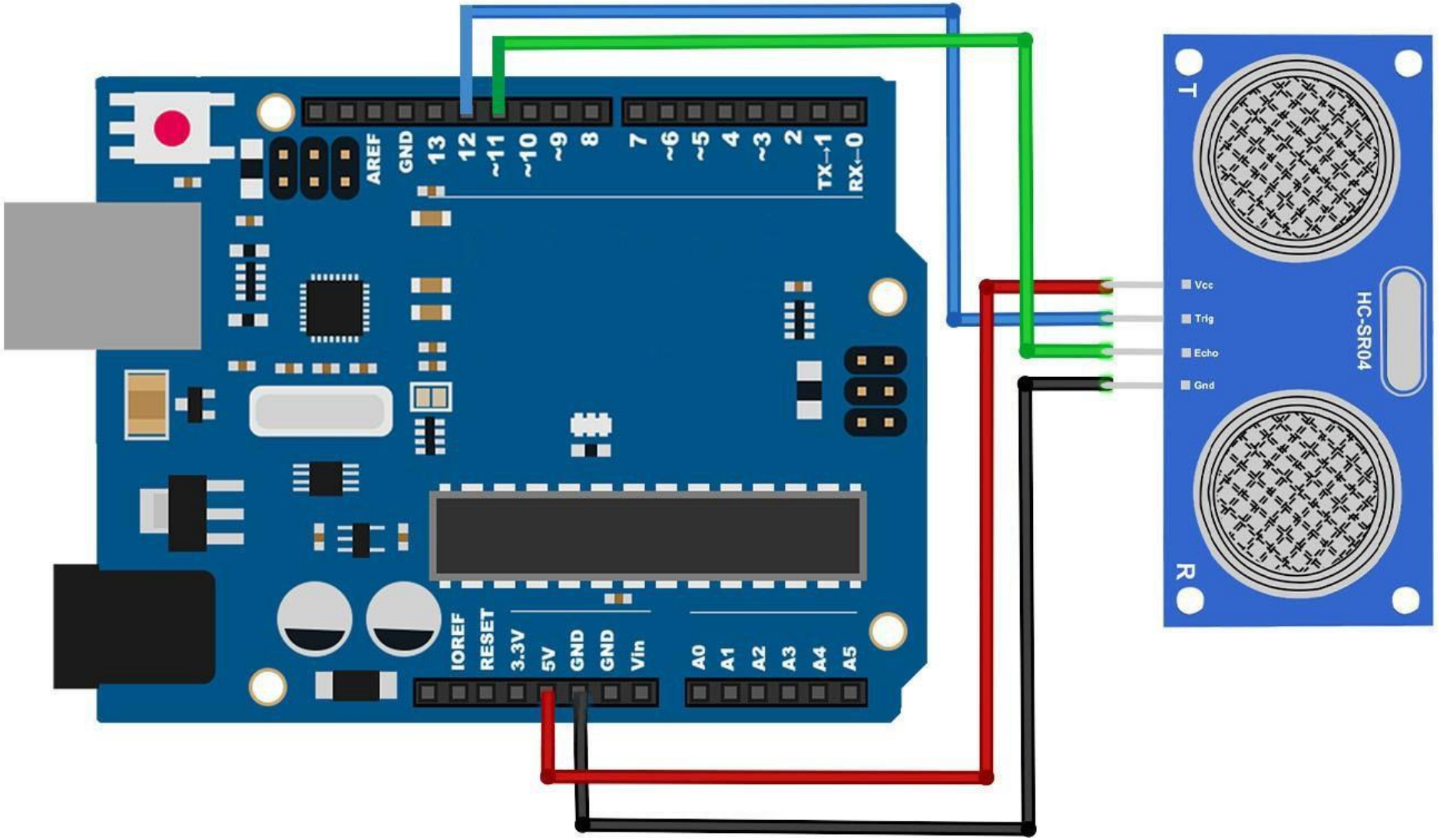


Connection

Schematic



Wiring diagram



Code

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

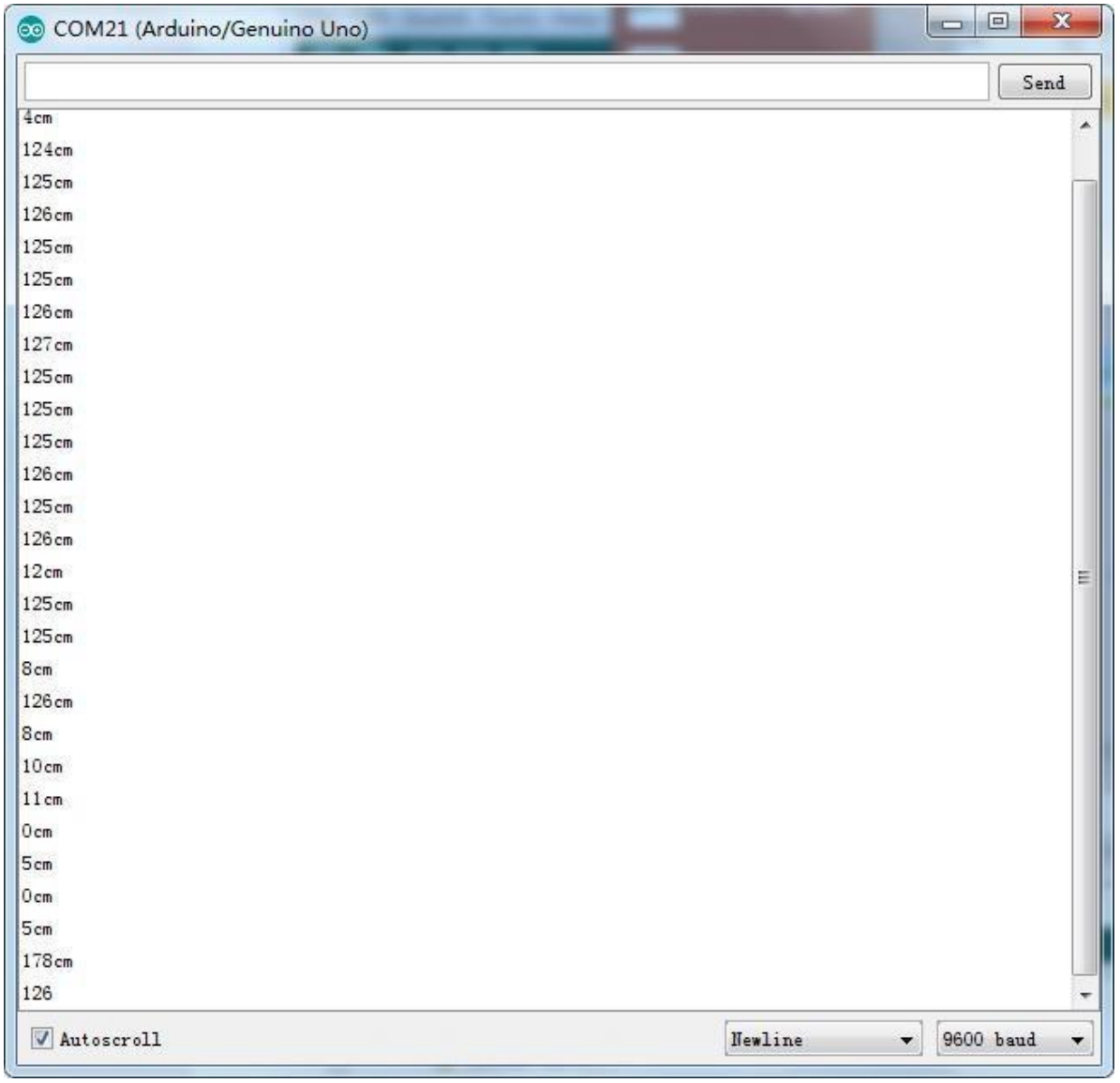
After wiring, please open the program in the code folder- Lesson 10 Ultrasonic Sensor Module and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are anyerrors.

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.

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

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



Lesson 11 DHT11 Temperature and Humidity Sensor

Overview

In this tutorial we will learn how to use a DHT11 Temperature and Humidity Sensor. It's accurate enough for most projects that need to keep track of humidity and temperature readings.

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

Component Required:

(1) x Uno R3

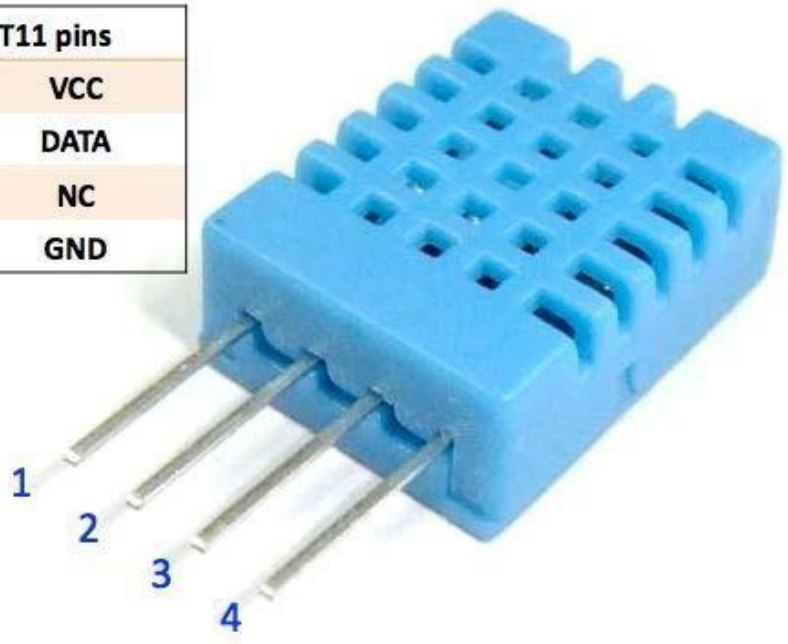
(1) x DHT11 Temperature and Humidity module

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

Component Introduction

Temp and humidity sensor:

DHT11 pins	
1	VCC
2	DATA
3	NC
4	GND



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

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

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

Product parameters

Relative humidity:

Resolution: 16Bit

Repeatability: ±1% RH

Accuracy: At 25℃ ±5% RH

Interchangeability: fully interchangeable

Response time: 1 / e (63%) of 25℃ 6s

1m / s air 6s

Hysteresis: <± 0.3% RH

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

Temperature:

Resolution: 16Bit

Repeatability: ±0.2℃

Range: At 25℃ ±2℃

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

Electrical Characteristics

Power supply: DC 3.5～5.5V

Supply Current: measurement 0.3mA standby 60μA

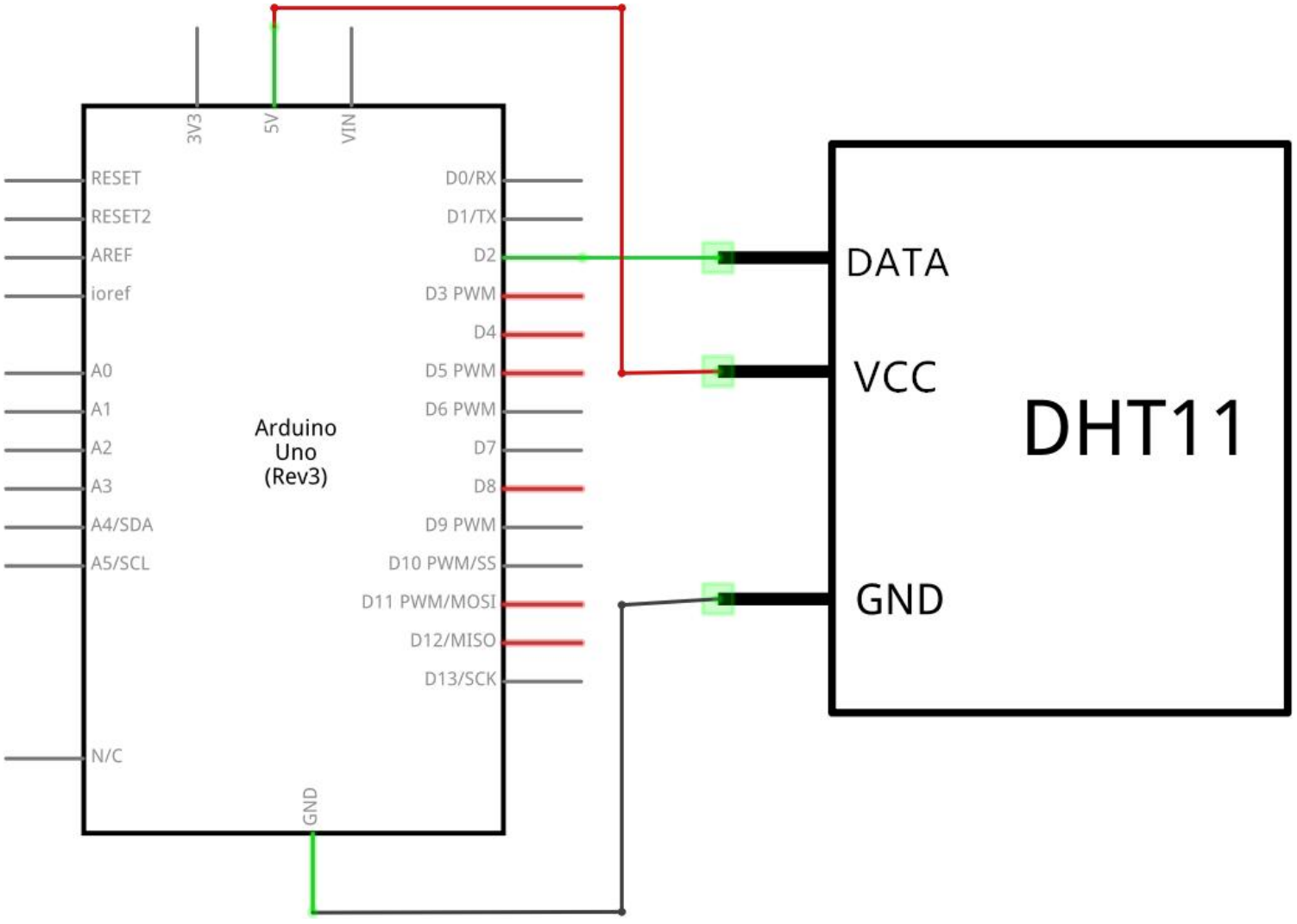
Sampling period: more than 2 seconds

Pin Description:

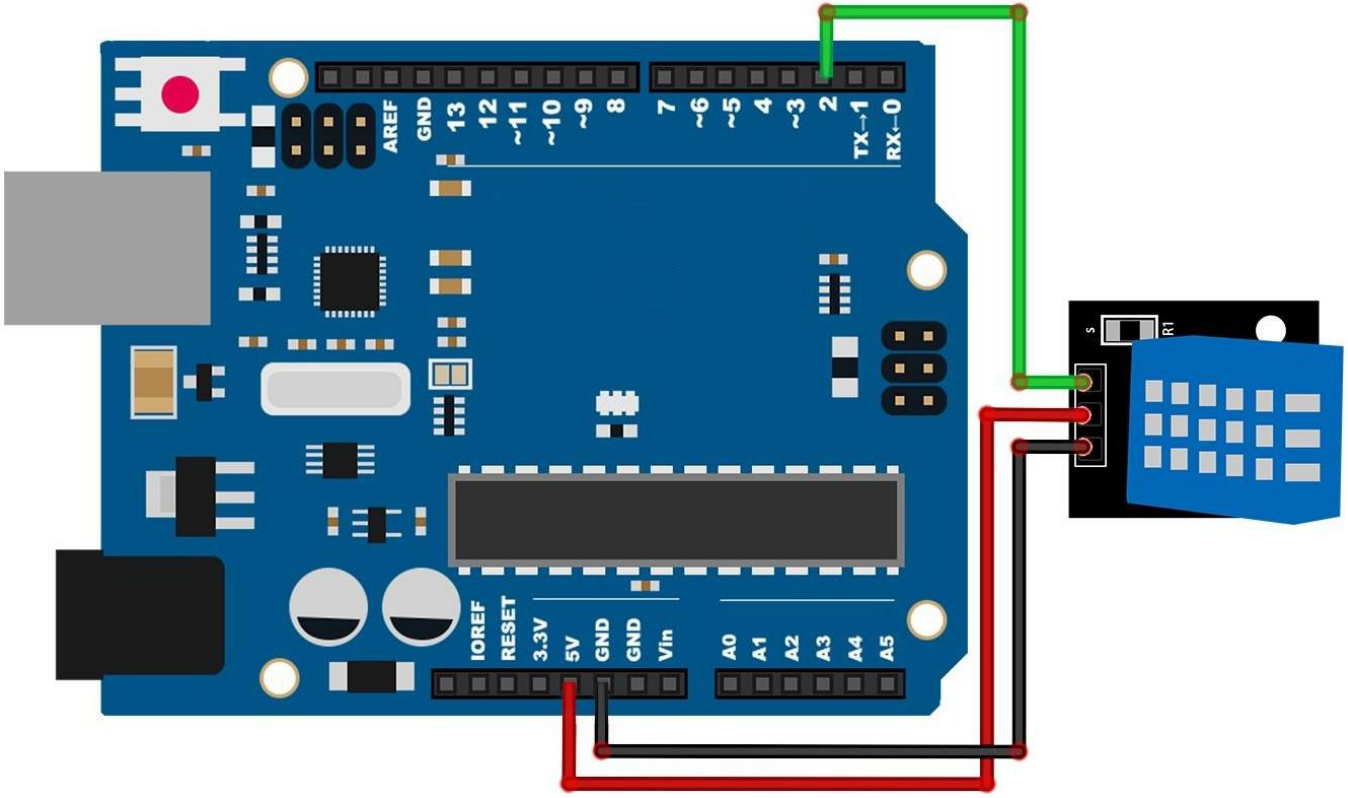
1. the VDD power supply 3.5～5.5V DC
2. DATA serial data, a single bus
3. NC, empty pin
4. GND ground, the negative power

Connection

Schematic



Wiring diagram



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

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

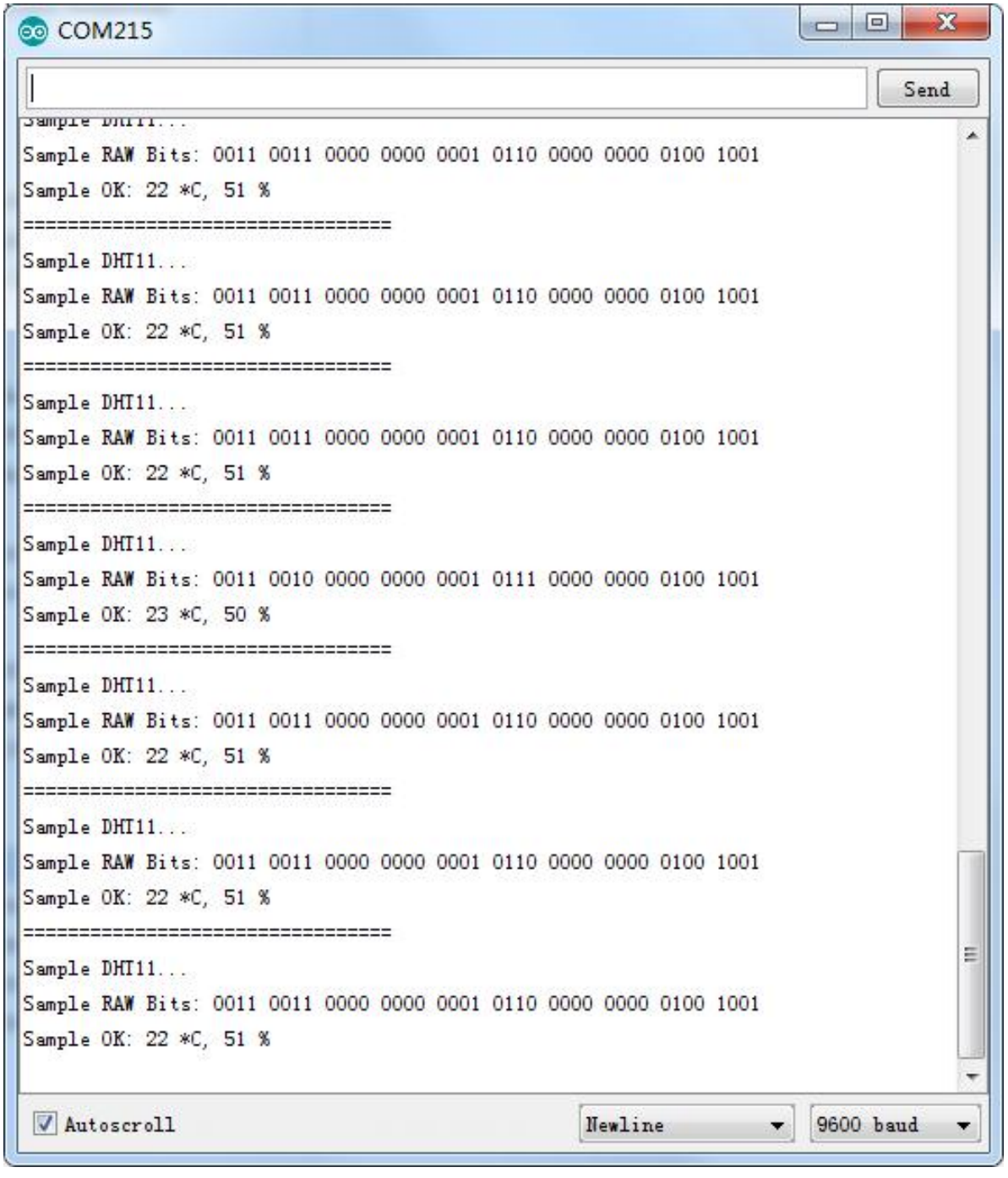
Code

After wiring, please open the program in the code folder- Lesson 12 DHT11 Temperature and Humidity Sensor and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

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

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

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



Lesson 12 Analog Joystick Module

Overview

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

Component Required:

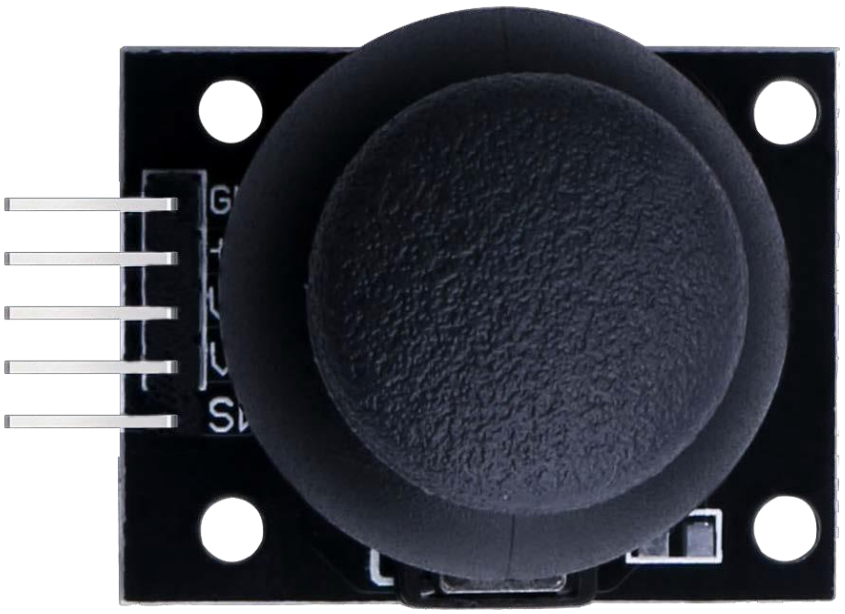
(1) x Uno R3
(1) x Joystick module
(5) x F-M wires (Female to Male DuPont wires)

Component Introduction

Joystick

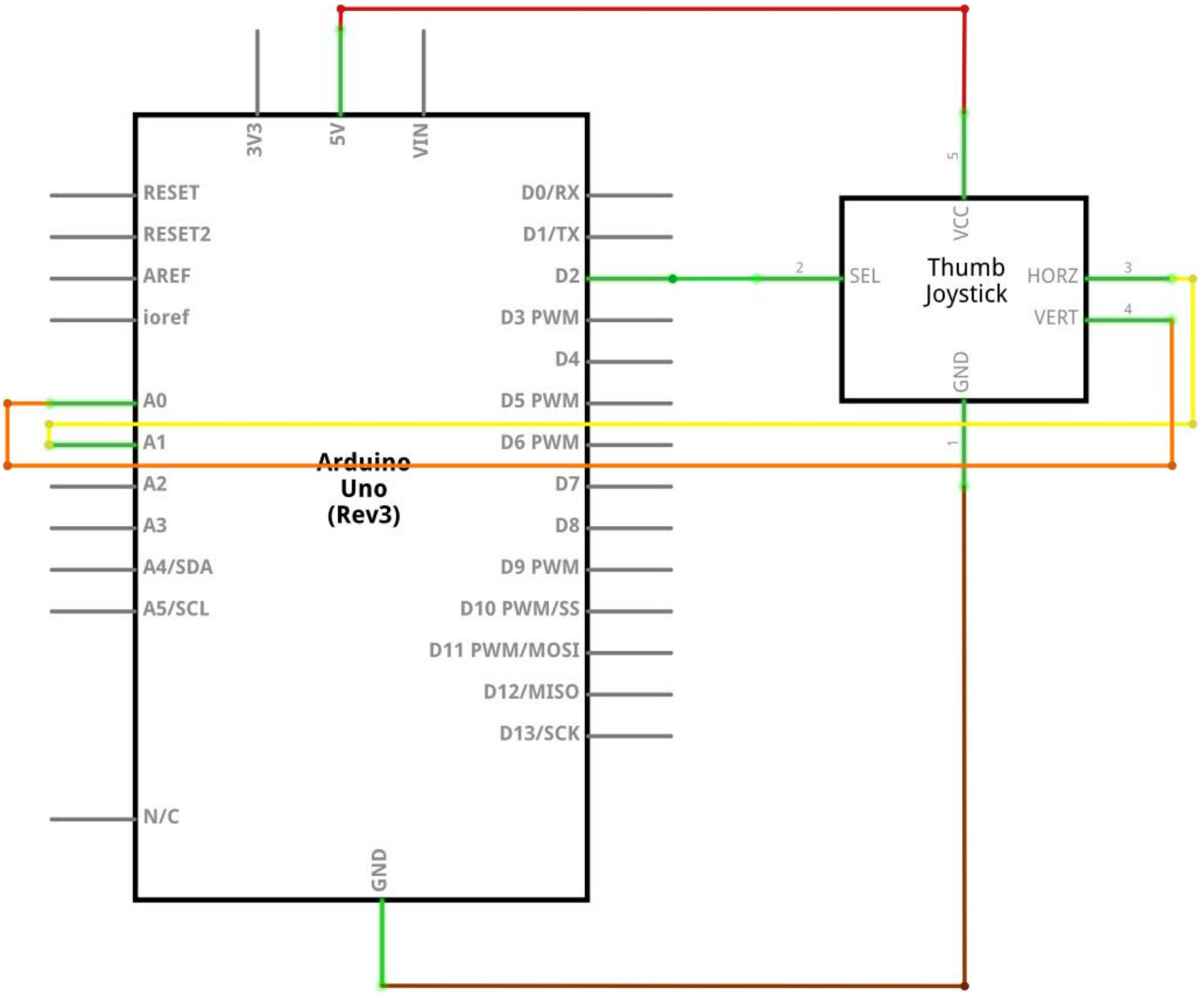
The module has 5 pins: VCC, Ground, X, Y, Key. Note that the labels on yours may be slightly different, depending on where you got the module from. The thumb stick is analog and should provide more accurate readings than simple 'directional' joysticks tact use some forms of buttons, or mechanical switches.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

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

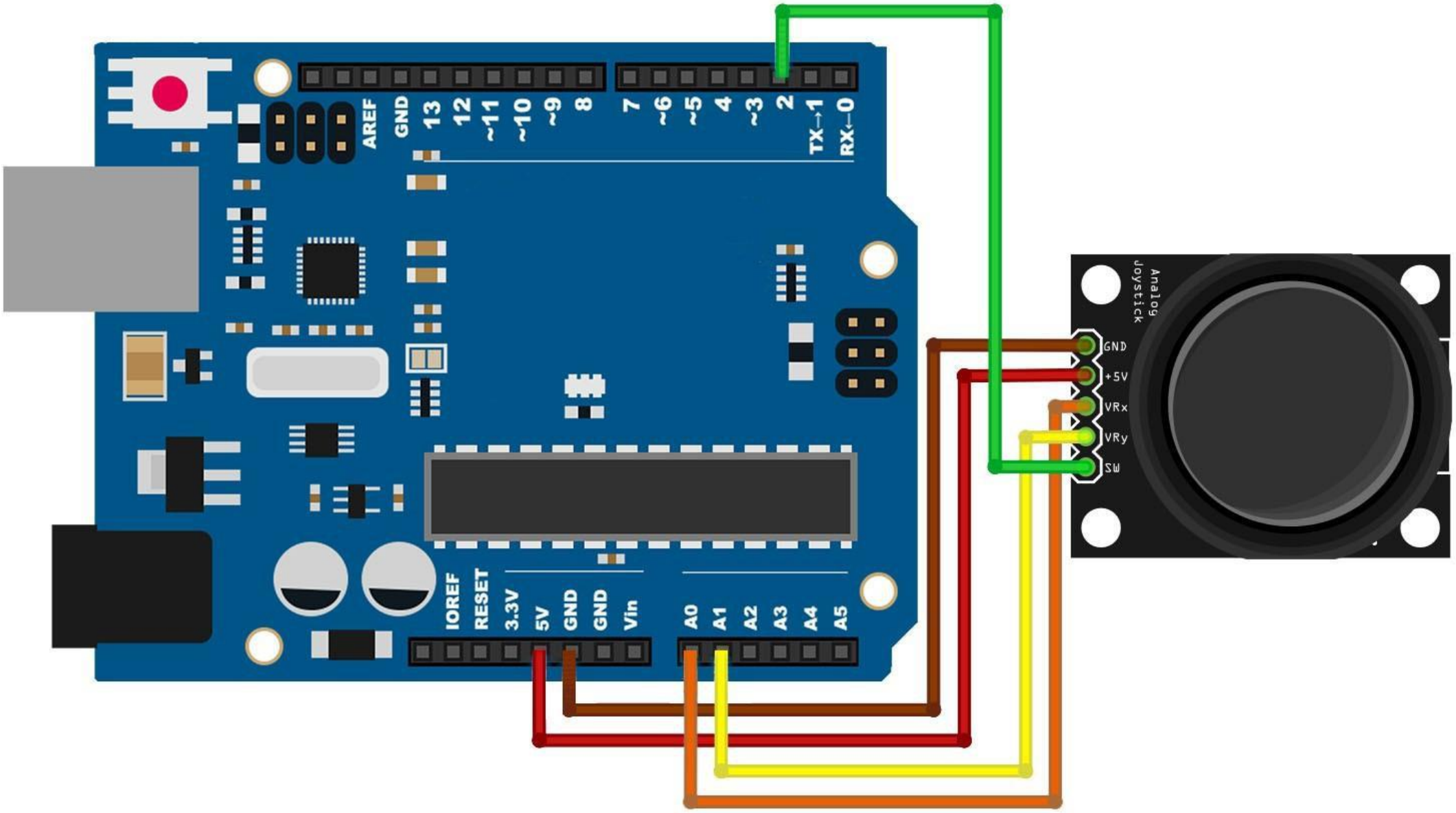


Connection

Schematic



Wiring diagram



We need 5 connections to the joystick.
The connections are: Key, Y, X, Voltage and Ground.
“Y and X” are Analog and “Key” is Digital. If you don’t need the switch then you can use only 4 pins.

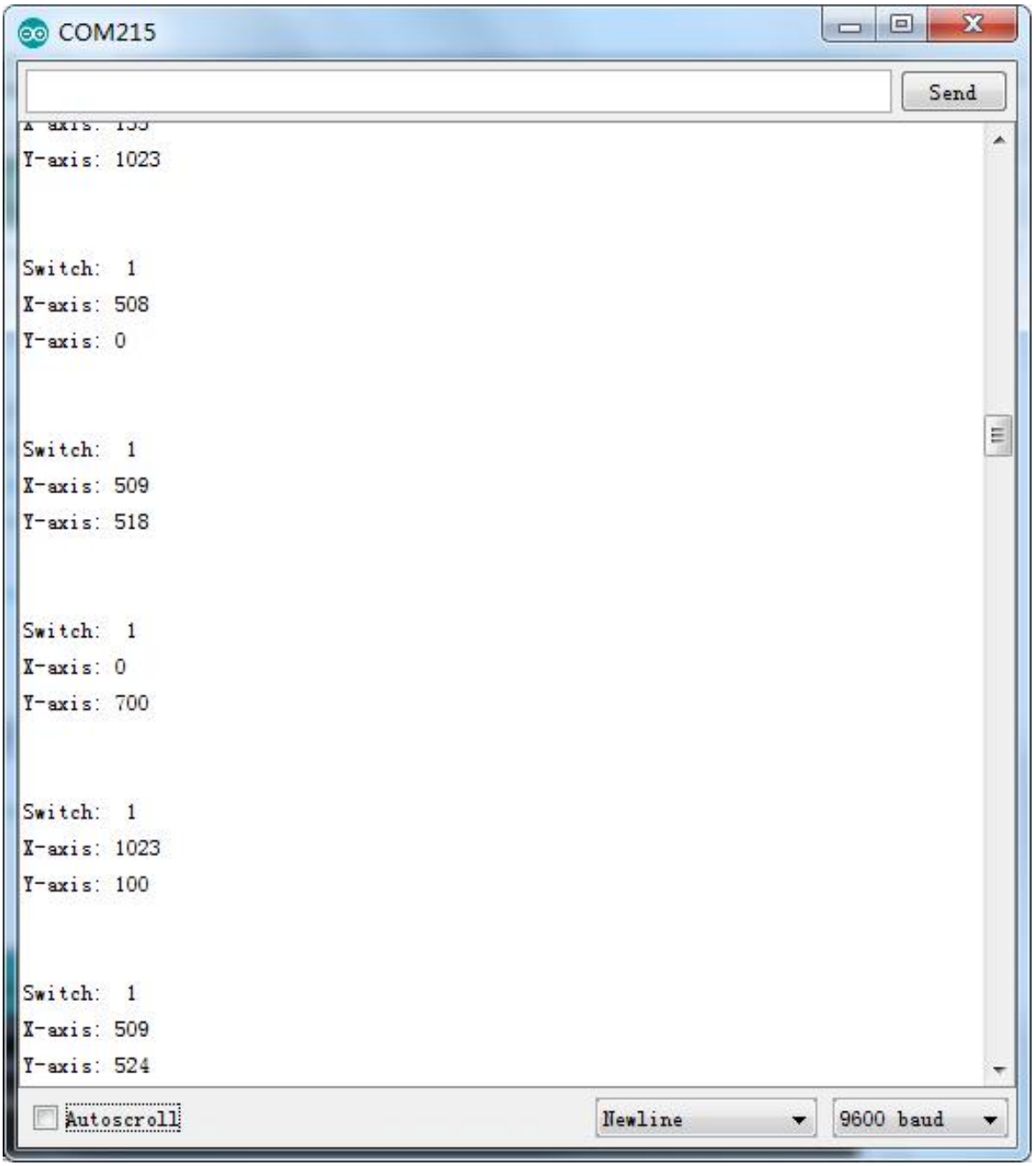
Code

After wiring, please open the program in the code folder- Lesson 13 Analog Joystick Module and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are anyerrors.

Analog joysticks are basically potentiometers so they return analog values. When the joystick is in the resting position or middle, it should return a value of about 512.
The range of values goes from 0 to 1024.

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

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



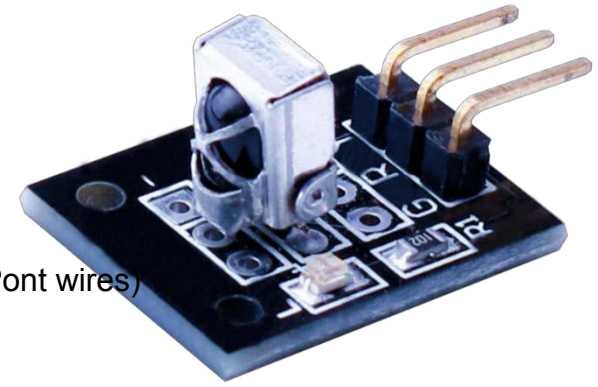
Lesson 13 IR Receiver Module

Overview

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

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

Component Required:



(1) x Uno R3

(1) x IR receiver module

(1) x IR remote

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

Component Introduction

IR RECEIVER SENSOR:

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

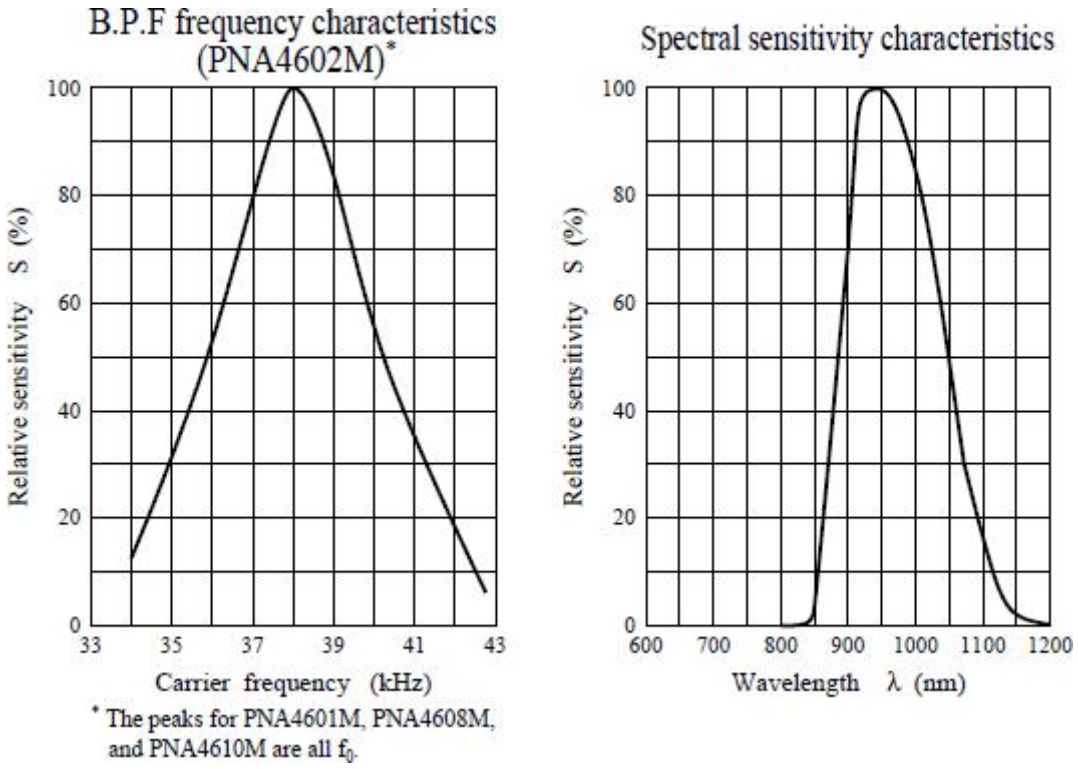
There are a few difference between these and say a CdS Photocells:

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

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

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

What You Can Measure

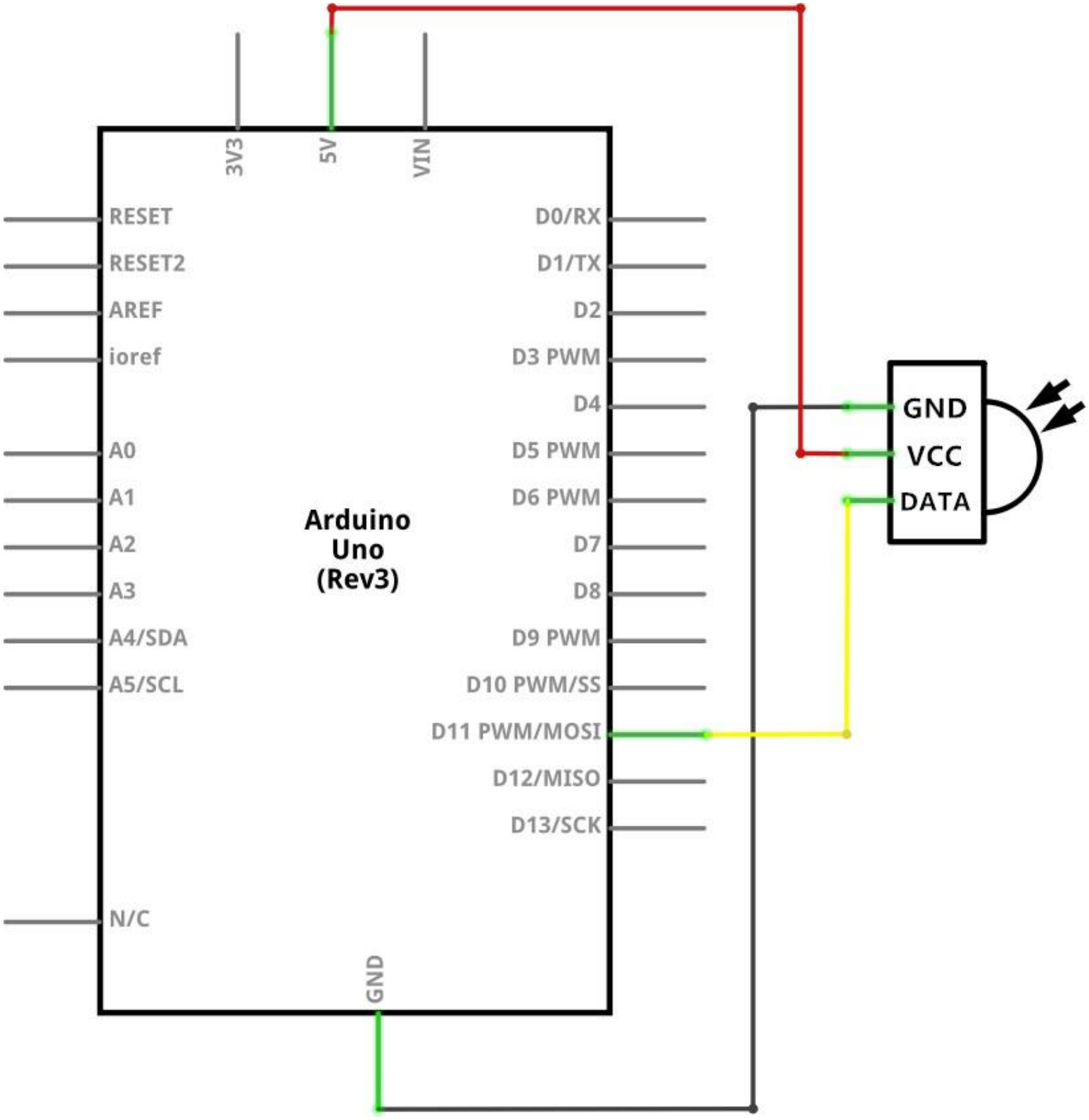


As you can see from these datasheet graphs, the peak frequency detection is at 38 KHz and the peak LED color is 940 nm. You can use from about 35 KHz to 41 KHz but the sensitivity will drop off so that it won't detect as well from afar. Likewise, you can use 850 to 1100 nm LEDs but they won't work as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

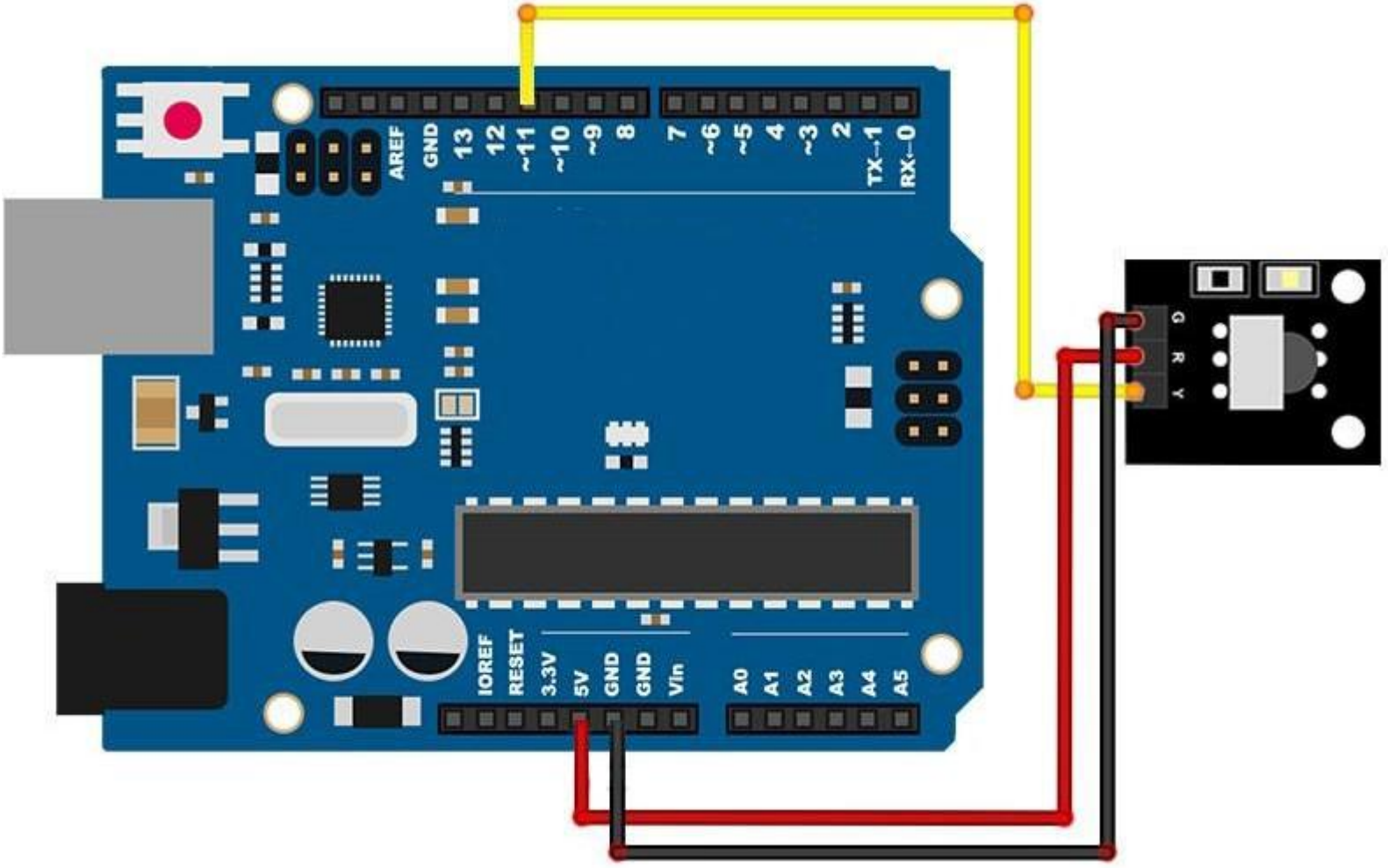
Try to get a 940nm - remember that 940nm is not visible light!

Connection

Schematic



Wiring diagram



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

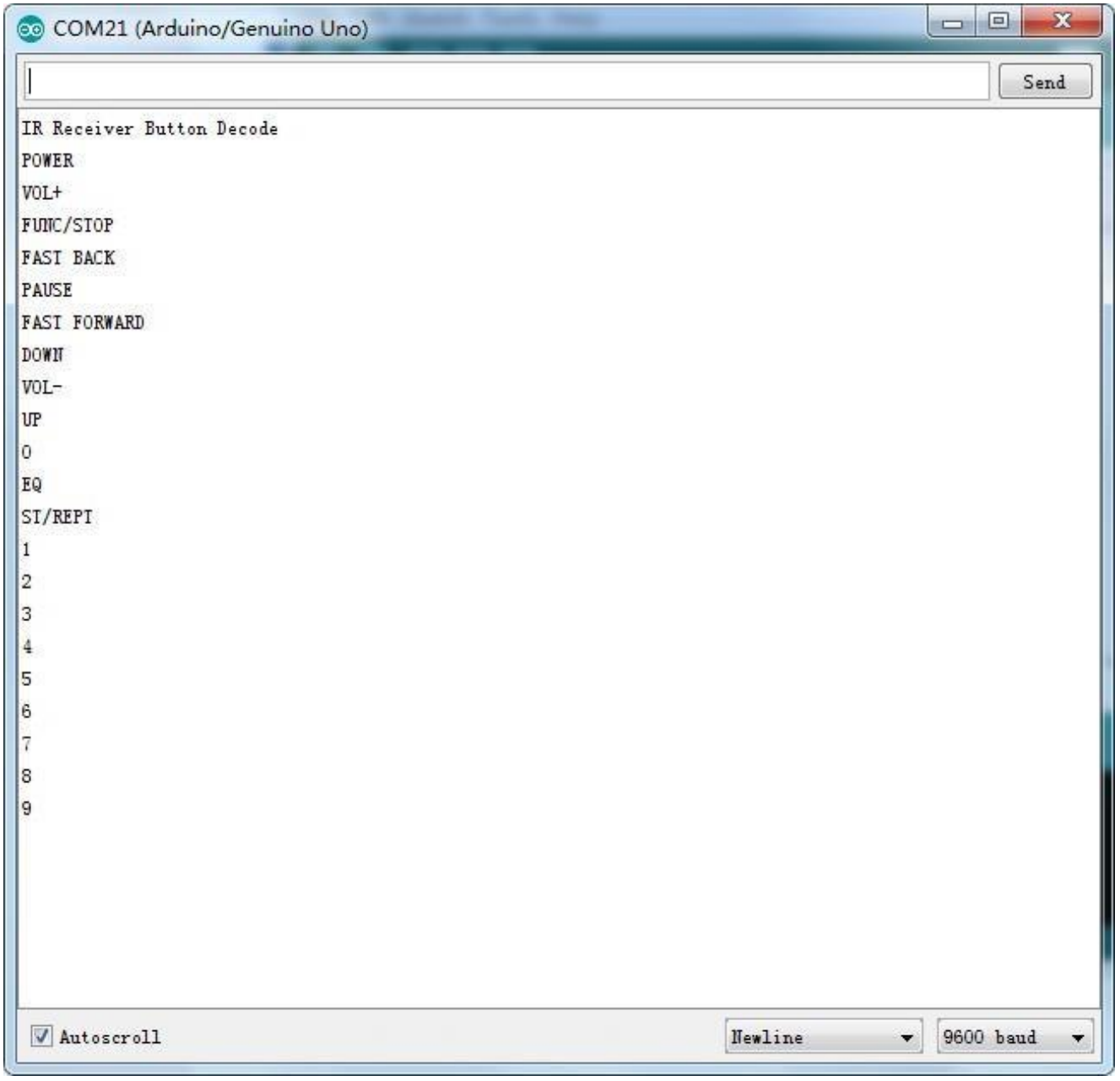
Code

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

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

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

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



Lesson 14 LCD Display

Overview

In this lesson, you will learn how to wire up and use an alphanumeric LCD display. The display has an LED backlight and can display two rows with up to 16 characters on each row. You can see the rectangles for each character on the display and the pixels that make up each character. The display is just white on blue and is intended for showing text.

In this lesson, we will run the Arduino example program for the LCD library, but in the next lesson, we will get our display to show the temperature, using sensors.

Component Required:

(1) x 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)

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 adjust the contrast of LCD1602

RS: A register select pin that controls where in the LCD's memory you are writing data to. You can select either the data register, which holds what goes 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

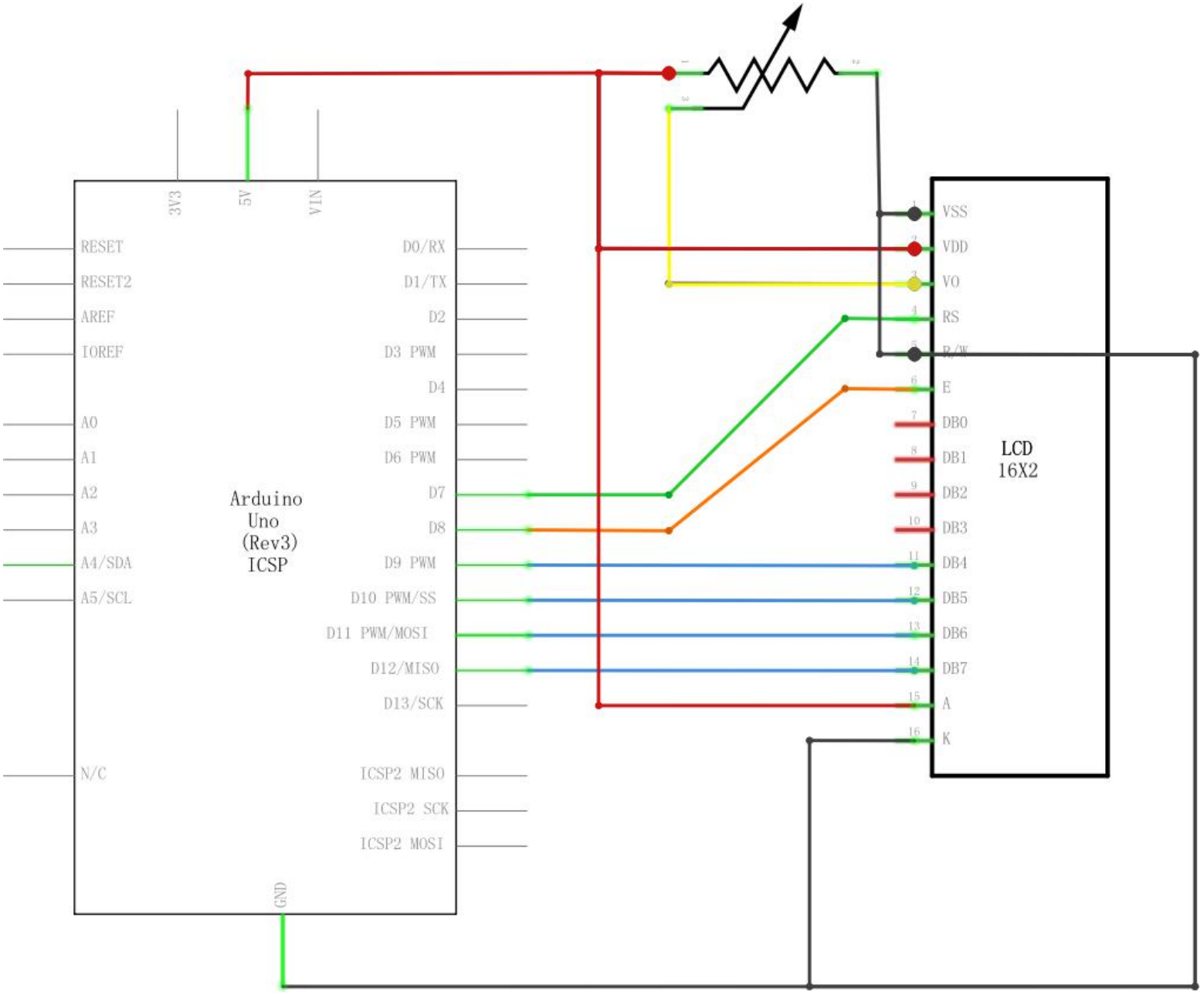
E: An enabling pin that, when supplied with low-level energy, causes the LDC module to execute relevant instructions.

D0-D7：Pins that read and write data

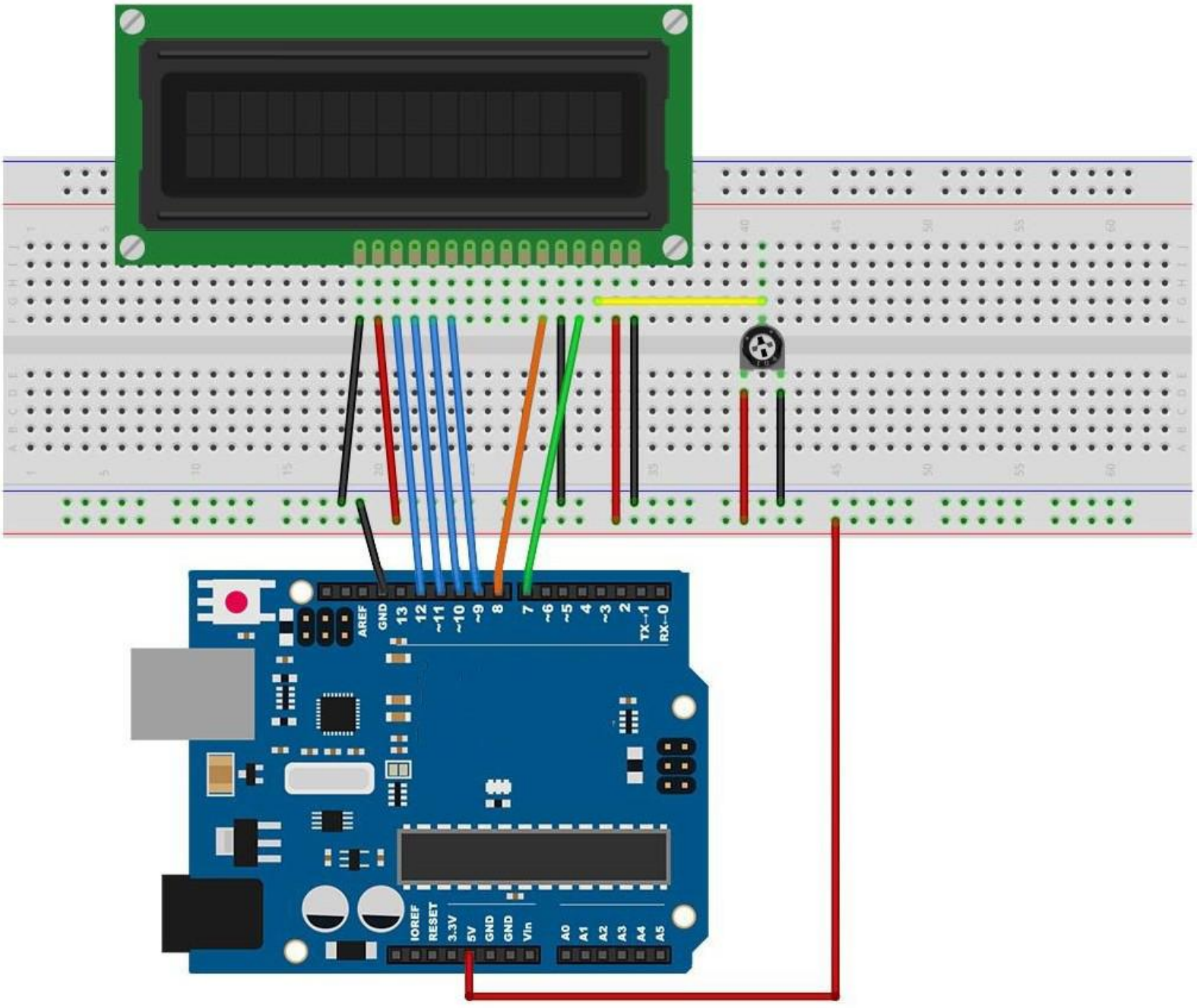
A and K: Pins that control the LED backlight

Connection

Schematic



Wiring diagram



The LCD display needs six Arduino pins, all set to be digital outputs. It also needs 5V and GND connections.

There are a number of connections to be made. Lining up the display with the top of the breadboard helps to identify its pins without too much counting, especially if the breadboard has its rows numbered with row 1 as the top row of the board. Do not forget, the long yellow lead that links the slider of the pot to pin 3 of the display. The 'pot' is used to control the contrast of the display.

You may find that your display is supplied without header pins attached to it. If so, follow the instructions in the next section.

Code

After wiring, please open the program in the code folder- Lesson 22 LCD Display and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

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

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

Upload the code to your Arduino board and you should see the message 'hello, world' displayed, followed by a number that counts up from zero.

The first thing of note in the sketch is the line:

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

This tells Arduino that we wish to use the Liquid Crystal library.

Next we have the line that we had to modify. This defines which pins of the Arduino are to be connected to which pins of the display.

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

After uploading this code, make sure the backlight is lit up, and adjust the potentiometer all the way around until you see the text message

In the 'setup' function, we have two commands:

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

The first tells the Liquid Crystal library how many columns and rows the display has. The second line displays the message that we see on the first line of the screen.

In the 'loop' function, we aso have two commands:

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

The first sets the cursor position (where the next text will appear) to column 0 & row 1. Both column and row numbers start at 0 rather than 1.

The second line displays the number of milliseconds since the Arduino was reset.

Example picture

Lesson 15 Thermometer

Overview

In this lesson, you will use an LCD display to show the temperature.

Component Required:

(1) x 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)

Component Introduction

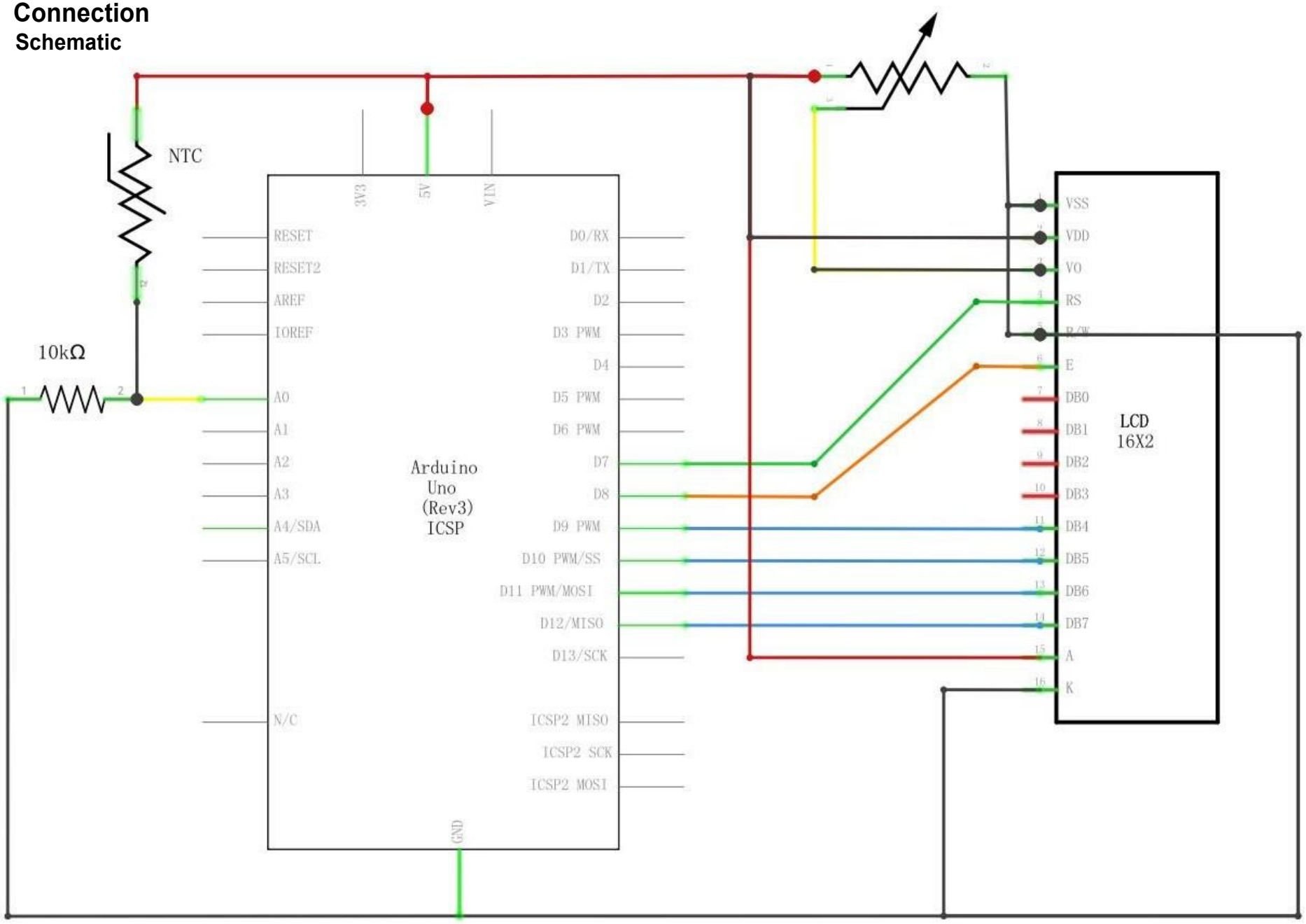
Thermistor

A thermistor is a thermal resistor - a resistor that changes its resistance with temperature. Technically, all resistors are thermistors - their resistance changes slightly with temperature - but the change is usually very small and difficult to measure. Thermistors are made so that the resistance changes drastically with temperature so that it can be 100 ohms or more of change per degree!

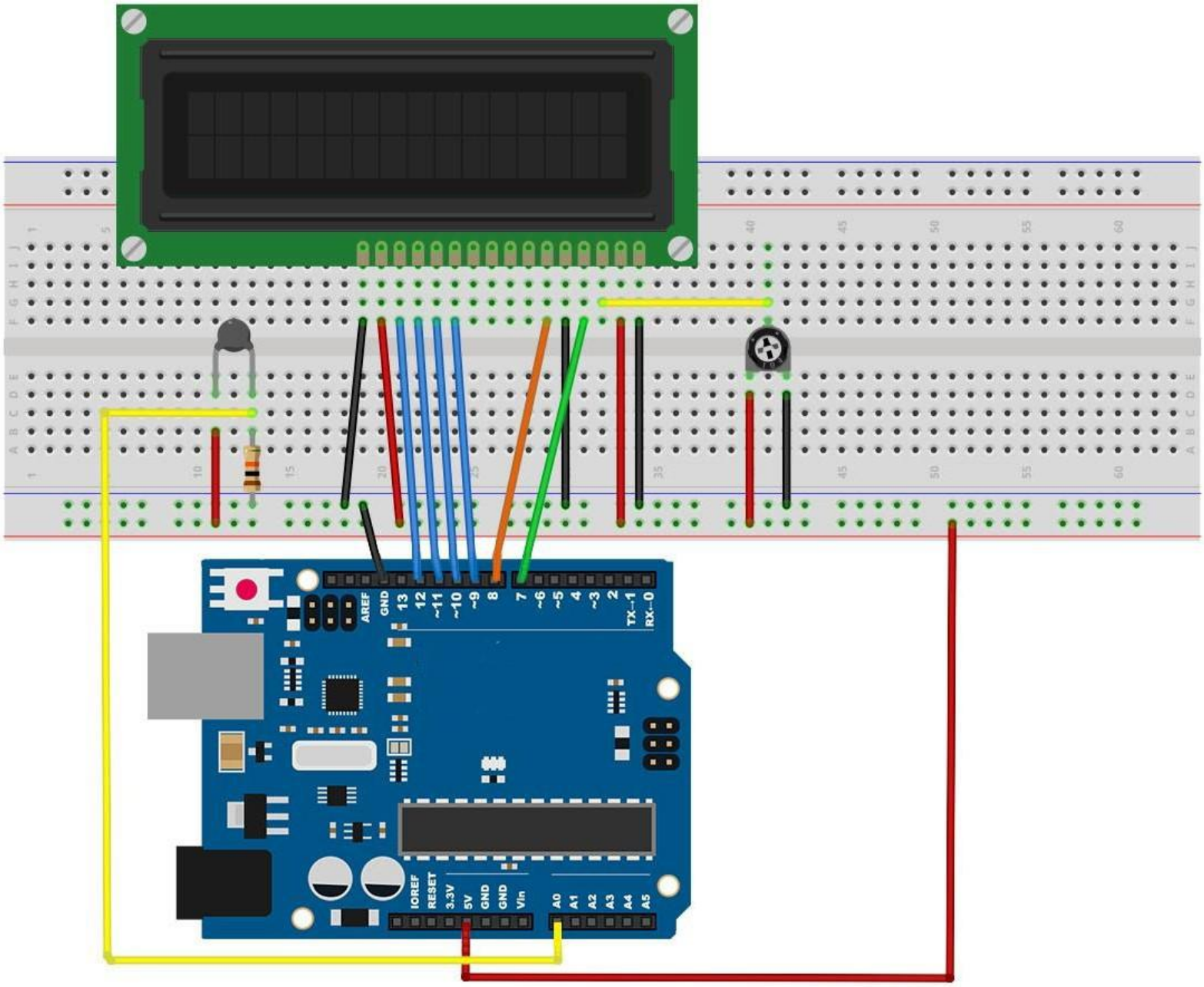
There are two kinds of thermistors, NTC (negative temperature coefficient) and PTC (positive temperature coefficient). In general, you will see NTC sensors used for temperature measurement. PTC's are often used as resettable fuses - an increase in temperature increases the resistance which means that as more current passes thru them, they heat up and 'choke back' the current, quite handy for protecting circuits!

Connection

Schematic



Wiring diagram



The breadboard layout is based on the layout from Lesson 22, so it will simplify things if you still have this on the breadboard.
There are a few jumper wires near the pot that have been moved slightly on this layout.
The 10 kΩ resistor and thermistor are all new additions to the board.

Code

After wiring, please open the program in the code folder- Lesson 23 Thermometer and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.
Before you can run this, make sure that you have installed the < 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 for this is based on that of lesson 22. Load it up onto your Arduino and you should find that warming the temperature sensor by putting your finger on it will increase the temperature reading.
I find it useful to put a comment line above the 'lcd' command.

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

This makes things easier if you decide to change which pins you use.
In the 'loop' function there are now two interesting things going on. Firstly we have to convert the analog from the temperature sensor into an actual temperature, and secondly we have to work out how to display them.
First of all, let's look at calculating the temperature.

```
  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 changing readings on an LCD display can be tricky. The main problem is that the reading may not always be the same number of digits. So, if the temperature changed from 101.50 to 99.00 then the extra digit from the old reading is in danger of being left on the display.
To avoid this, write the whole line of the LCD each time around the loop.

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

The rather strange comment serves to remind you of the 16 columns of the display. You can then print a string of that length with spaces where the actual reading will go.

To fill in the blanks, set the cursor position for where the reading should appear and then print it.

Example picture

Lesson 16 Eight LED with 74HC595

Overview

In this lesson, you will learn how to use eight large red LEDs with an UNO without needing to give up 8 output pins!

Although you could wire up eight LEDs each with a resistor to an UNO pin you would rapidly start to run out of pins on your UNO. If you don't have a lot of stuff connected to your UNO. It's OK to do so - but often times we want buttons, sensors, servos, etc. and before you know it you've got no pins left. So, instead of doing that, you are going to use a chip called the 74HC595 Serial to Parallel Converter. This chip has eight outputs (perfect) and three inputs that you use to feed data into it a bit at a time.

This chip makes it a little slower to drive the LEDs (you can only change the LEDs about 500,000 times a second instead of 8,000,000 a second) but it's still really fast, way faster than humans can detect, so it's worth it!

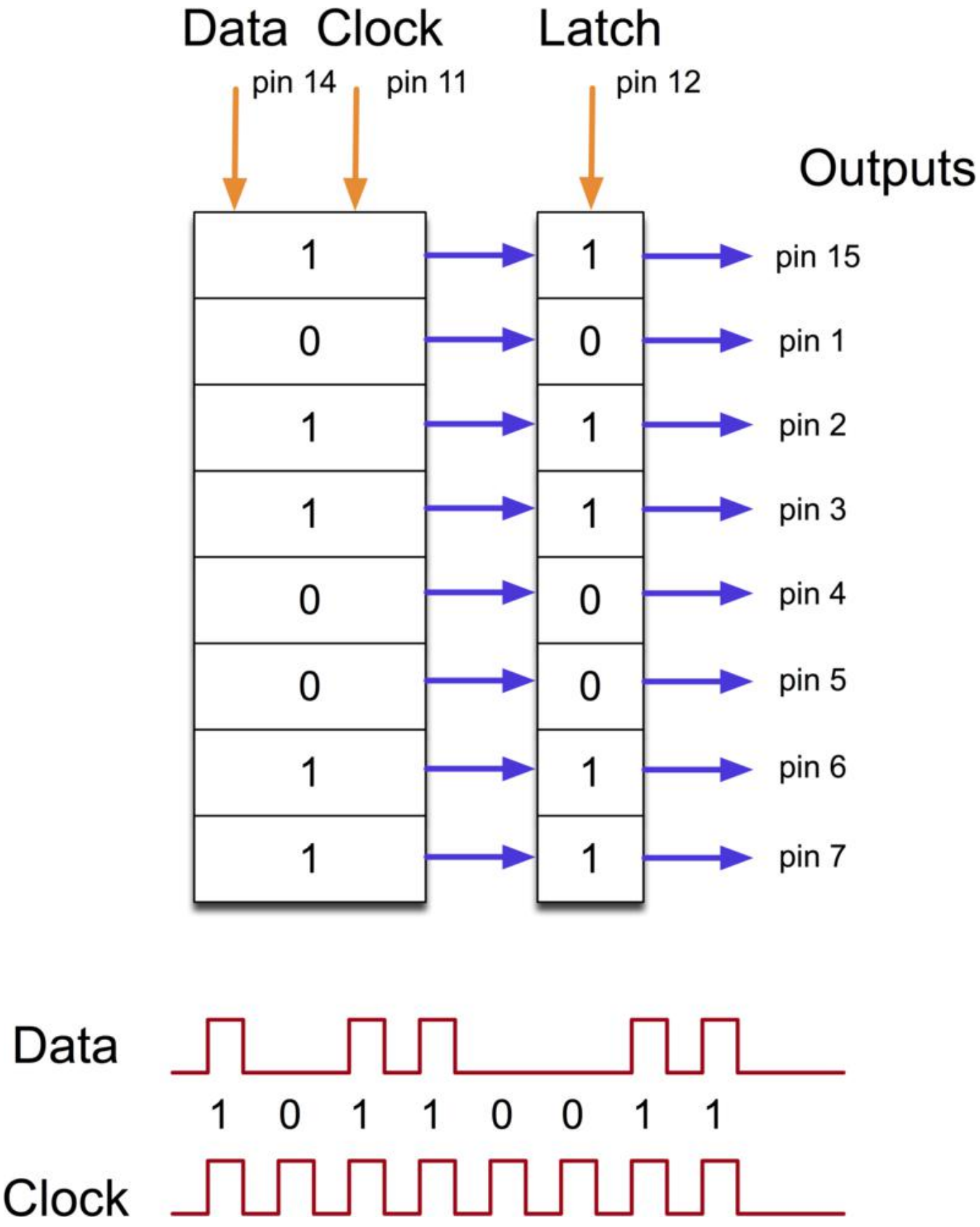
Component Required:

(1) x 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)

Component Introduction

74HC595 Shift Register:

The shift register is a type of chip that holds what can be thought of as eight memory locations, each of which can either be a 1 or a 0. To set each of these values on or off, we feed in the data using the 'Data' and 'Clock' pins of the chip.

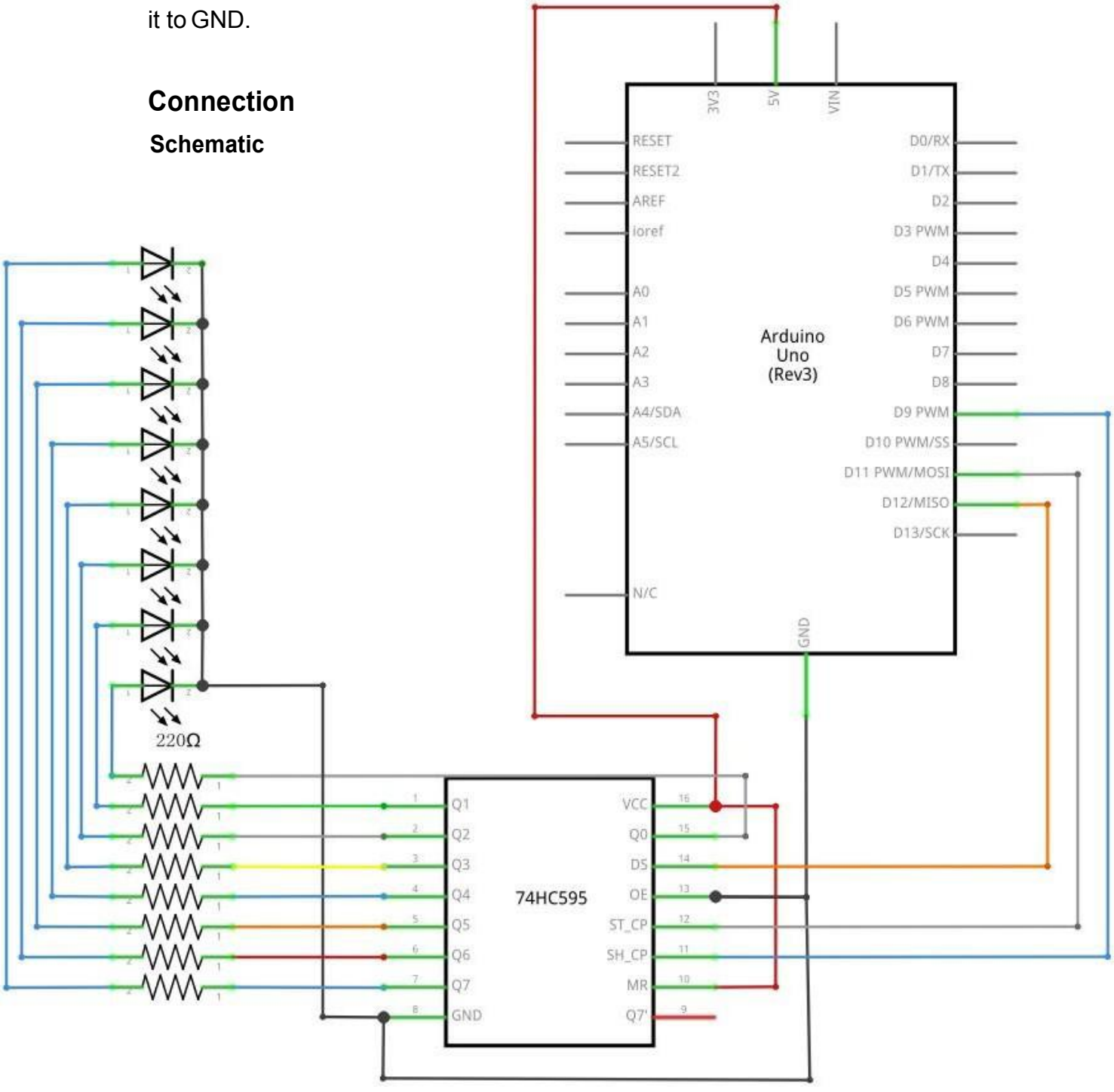

Data
Clock
Latch
pin 14
pin 11
pin 12
Outputs
1
0
1
1
0
0
0
1
1
0
1
1
1
0
0
1
pin 15
pin 1
pin 2
pin 3
pin 4
pin 5
pin 6
pin 7
Data
1 0 1 1 0 0 1 1
Clock

The clock pin needs to receive eight pulses. At each pulse, if the data pin is high, then a 1 gets pushed into the shift register; otherwise, a 0. When all eight pulses have been received, enabling the 'Latch' pin copies those eight values to the latch register. This is necessary; otherwise, the wrong LEDs would flicker as the data is being loaded into the shift register.

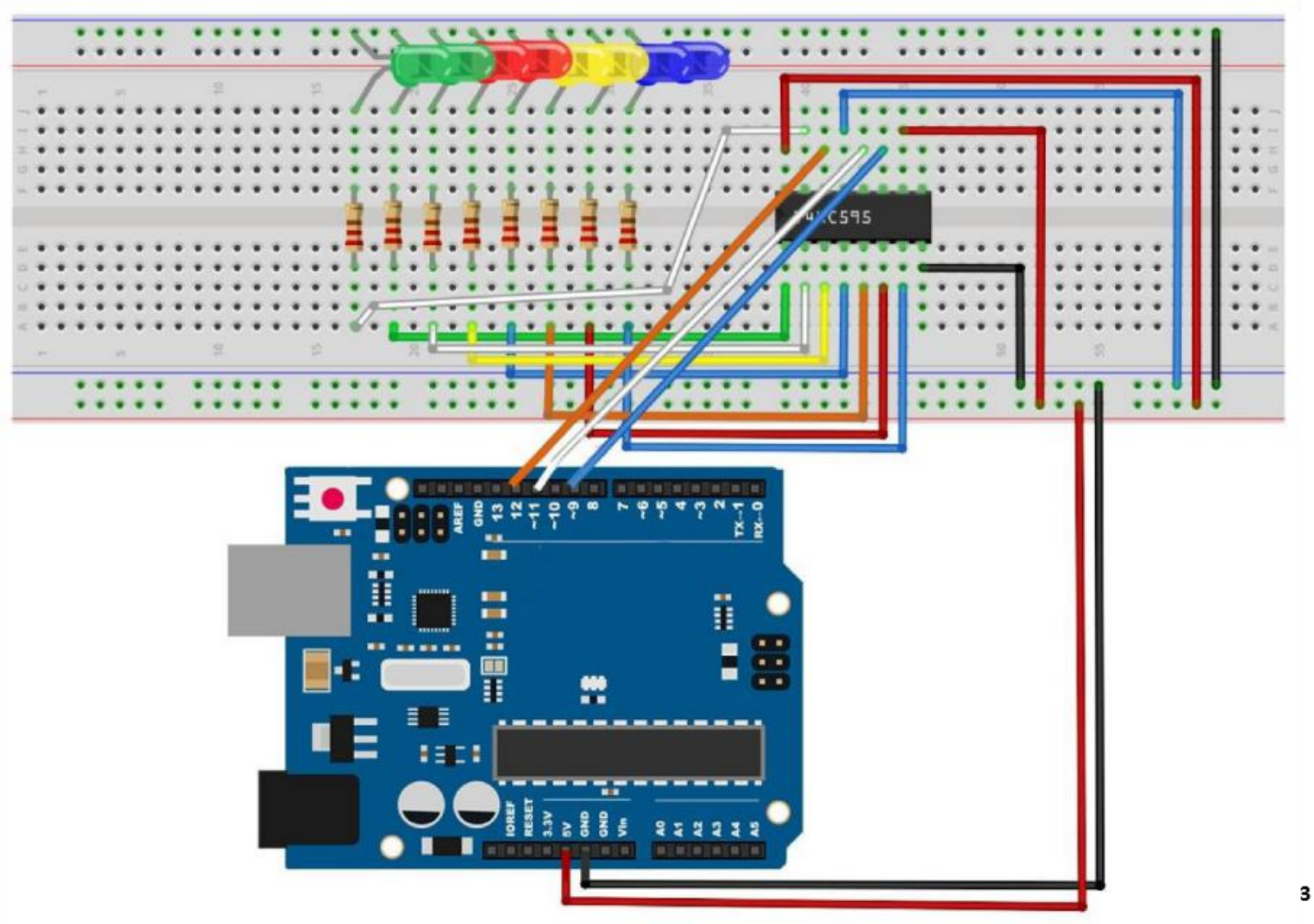
The chip also has an output enable (OE) pin, which is used to enable or disable the outputs all at once. You could attach this to a PWM-capable UNO pin and use 'analogWrite' to control the brightness of the LEDs. This pin is active low, so we tie it to GND.

Connection

Schematic



Wiring diagram



As we have eight LEDs and eight resistors to connect, there are actually quite a few connections to be made.

It is probably easiest to put the 74HC595 chip in first, as pretty much everything else connects to it. Put it so that the little 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 but one of the outputs from the IC is on the left side of the chip. Hence, for ease of connection, that is where the LEDs are, too.

After the chip, put the resistors in place. You need to be careful that none of the leads of the resistors are touching each other. You should check this again before you connect the power to your UNO. If you find it difficult to arrange the resistors without their leads touching, then it helps to shorten the leads so that they are lying closer to the surface of the breadboard.

Next, place the LEDs on the breadboard. The longer positive LED leads must all be towards the chip, whichever side of the breadboard they are on.

Attach the jumper leads as shown above. Do not forget the one that goes from pin 8 of the IC to the GND column of the breadboard.

Load up the sketch listed a bit later and try it out. Each LED should light in turn until all the LEDs are on, and then they all go off and the cycle repeats.

Code

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

The first thing we do is define the three pins we are going to use. These are the UNO 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;
```

Next, a variable called 'leds' is defined. This will be used to hold the pattern of which LEDs are currently turned on or off. Data of type 'byte' represents numbers using eight bits. Each bit can be either on or off, so this is perfect for keeping track of

which of our eight LEDs are on or off.

```
byte leds = 0;
```

The 'setup' function just sets the three pins we are using to be digital outputs.

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

The 'loop' function initially turns all the LEDs off, by giving the variable 'leds' the value 0. It then calls 'updateShiftRegister' that will send the 'leds' pattern to the shift register so that all the LEDs turn off. We will deal with how 'updateShiftRegister' works later.

The loop function pauses for half a second and then begins to count from 0 to 7 using the 'for' loop and the variable 'i'. Each time, it uses the Arduino function 'bitSet' to set the bit that controls that LED in the variable 'leds'. It then also calls 'updateShiftRegister' so that the leds update to reflect what is in the variable 'leds'. There is then a half second delay before 'i' is incremented and the next LED is lit.

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

The function 'updateShiftRegister', first of all sets the latchPin to low, then calls the UNO function 'shiftOut' before putting the 'latchPin' high again. This takes four parameters, the first two are the pins to use for Data and Clock respectively.

The third parameter specifies which end of the data you want to start at. We are going to start with the right most bit, which is referred to as the 'Least Significant

Bit' (LSB).

The last parameter is the actual data to be shifted into the shift register, which in this case is'leds'.

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

If you wanted to turn one of the LEDs off rather than on, you would call a similar Arduino function (bitClear) with the 'leds' variable. This will set that bit of 'leds' to be 0 and you would then just need to follow it with a call to 'updateShiftRegister' to update the actual LEDs.

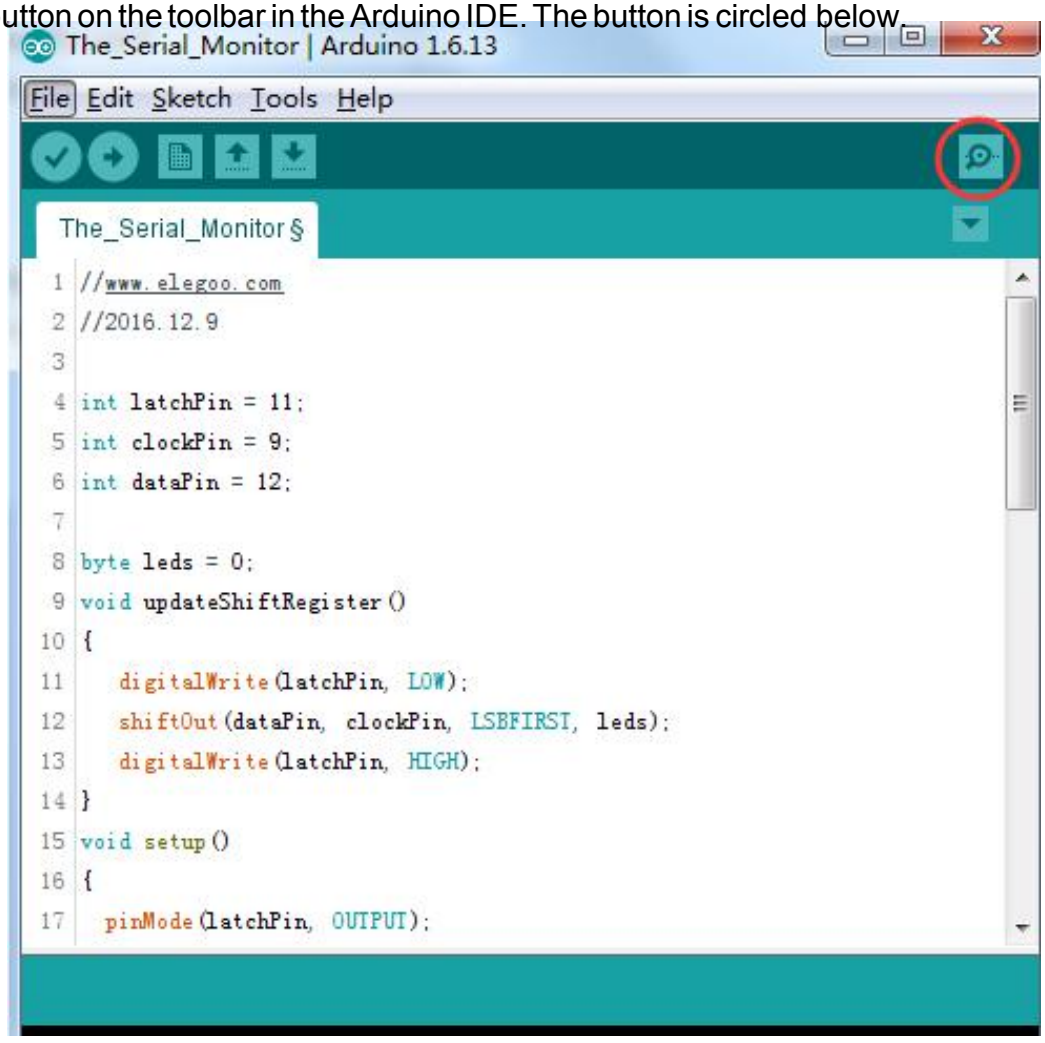
Lesson 17 The Serial Monitor

Overview

In this lesson, you will build on Lesson 16, adding the facility to control the LEDs from your computer using the Arduino Serial Monitor. The serial monitor is the 'tether' between the computer and your UNO. It lets you send and receive text messages, handy for debugging and also controlling the UNO from a keyboard! For example, you will be able to send commands from your computer to turn on LEDs. In this lesson, you will use exactly the same parts and a similar breadboard layout as Lesson 16. So, if you have not already done so, follow Lesson 16 now.

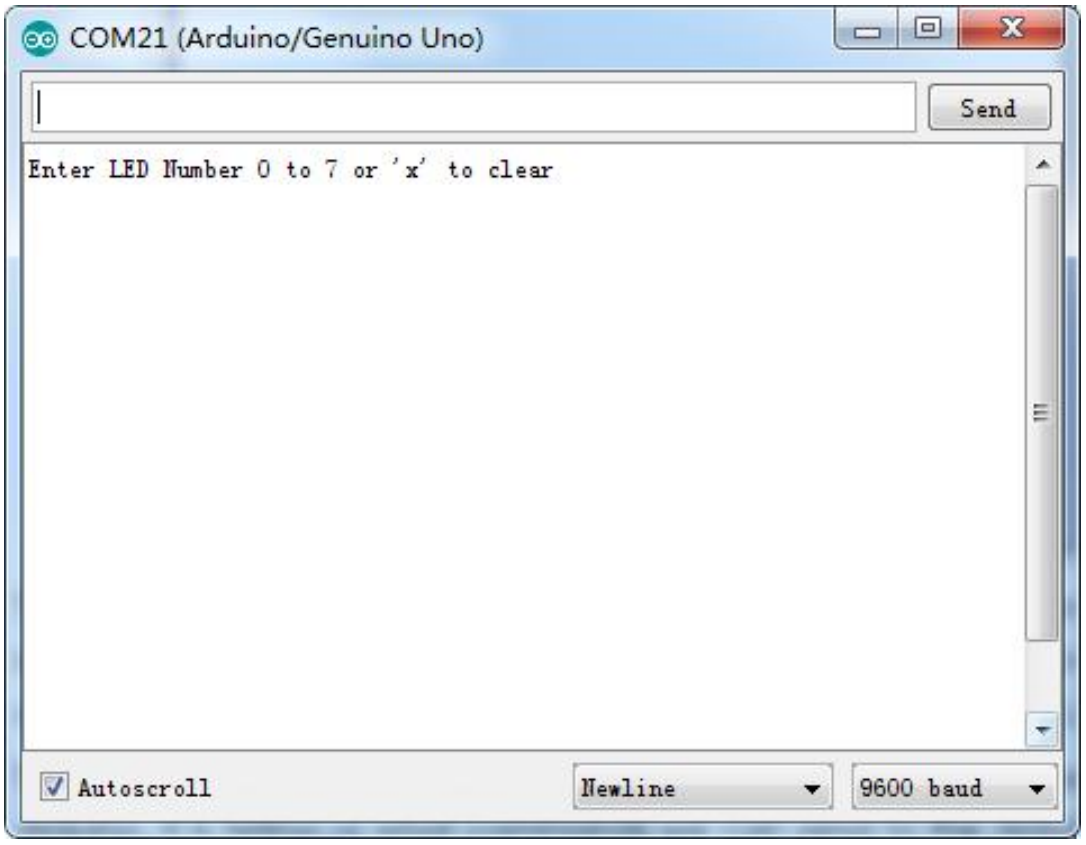
Steps taken

After you have uploaded this sketch onto your UNO, click on the right-most button on the toolbar in the Arduino IDE. The button is circled below.



The following window will open.

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

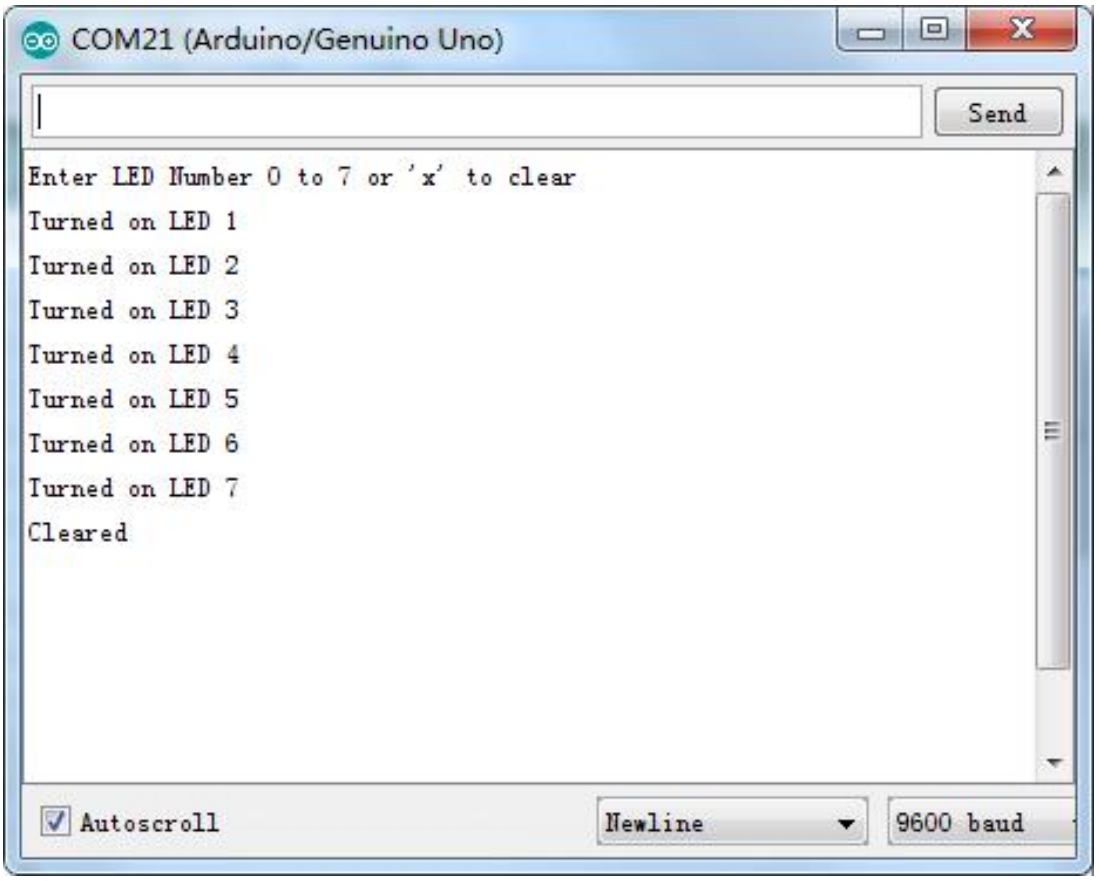


This window is called the Serial Monitor and it is part of the Arduino IDE software. Its job is to allow you to both send messages from your computer to an UNO board (over USB) and also to receive messages from the UNO.

The message “Enter LED Number 0 to 7or 'x' to clear” has been sent by the Arduino. It is telling us what commands we can send to the Arduino: either send the 'x' (to turn all the LEDs off) or the number of the LED you want to turn on (where 0 is the bottom LED, 1 is the next one up, all the way to 7 for the top LED).

Try typing the following commands into the top area of the Serial Monitor that is level with the 'Send' button. Press 'Send', after typing each of these characters: x 0 3 5

Typing x will have no effect if the LEDs are already all off, but as you enter each number, the corresponding LED should light and you will get a confirmation message from the UNO board. The Serial Monitor will appear as shown below.



Type x again and press ‘Send’ to turn off all LEDs.

Code

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

As you might expect, the sketch is based on the sketch used in Lesson 24. So, we will just cover the new bits here. You will find it useful to refer to the full sketch in your Arduino 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");
}
```

Firstly, we have the command 'Serial.begin(9600)'. This starts serial communication, so that the UNO can send out commands through the USB connection. The value 9600 is called the 'baud rate' of the connection. This is how fast the data is to be sent. You can change this to a higher value, but you will also have to change the Arduino Serial monitor to the same value. We will discuss this later; for now, leave it at 9600.

The line beginning with 'while' ensures that there is something at the other end of the USB connection for the Arduino to talk to before it starts sending messages. Otherwise, the message might be sent, but not displayed. This line is actually only necessary if you are using an Arduino Leonardo because the Arduino UNO automatically resets the Arduino board when you open the Serial Monitor, whereas this does not happen with the Leonardo.

The last of the new lines in 'setup' sends out the message that we see at the top of the Serial Monitor.

The 'loop' function is where all the action happens:

```
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");
      }
    }
}
```

Everything that happens inside the loop is contained within an 'if' statement. So unless the call to the built-in Arduino function 'Serial.available()' is 'true' then nothing else will happen.

Serial.available() will return 'true' if data has been send to the UNO and is there ready to be processed. Incoming messages are held in what is called a buffer and Serial.available() returns true if that buffer is Not empty.

If a message has been received, then it is on to the next line of code:

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

This reads the next character from the buffer, and removes it from the buffer. It also assigns it to the variable 'ch'. The variable 'ch' is of type 'char' which stands for 'character' and as the name suggests, holds a single character.

If you have followed the instructions in the prompt at the top of the Serial Monitor, then this character will either be a single digit number between 0 and 7 or the letter 'x'.

The 'if' statement on the next line checks to see if it is a single digit by seeing if 'ch' is greater than or equal to the character '0' and less than or equal to the character '7'. It looks a little strange comparing characters in this way, but is perfectly acceptable.

Each character is represented by a unique number, called its ASCII value. This means that when we compare characters using <= and >= it is actually the ASCII values that were being compared.

If the test passes, then we come to the next line:

```
int led = ch – '0';
```

Now we are performing arithmetic on characters! We are subtracting the digit '0' from whatever digit was entered. So, if you typed '0' then '0' – '0' will equal 0. If you typed '7' then '7' – '0' will equal the number 7 because it is actually the ASCII values that are being used in the subtraction.

Since that we know the number of the LED that we want to turn on, we just need to set that bit in the variable 'leds' and update the shift register.

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

The next two lines write back a confirmation message to the Serial Monitor.

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

The first line uses Serial.print rather than Serial.println. The different between the two is that Serial.print does not start a new line after printing whatever is in its parameter. We use this in the first line, because we are printing the message in two parts. Firstly the general bit: 'Turned on LED ' and then the number of the LED.

The number of the LED is held in an 'int' variable rather than being a text string. Serial.print can take either a text string enclosed in double-quotes, or an 'int' or for that matter pretty much any type of variable.

After the 'if' statement that handles the case, when a single digit has been handled, there is a second 'if' statement that checks to see if 'ch' is the letter 'x'.

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

If it is, then it clears all the LEDs and sends a confirmation message.

Lesson 18 Photocell

Overview

In this lesson, you will learn how to measure light intensity using an Analog Input. You will build on lesson 16 and use the level of light to control the number of LEDs to be lit.

The photocell is at the bottom of the breadboard, where the pot was above.

Component Required:

(1) x 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)

Component Introduction

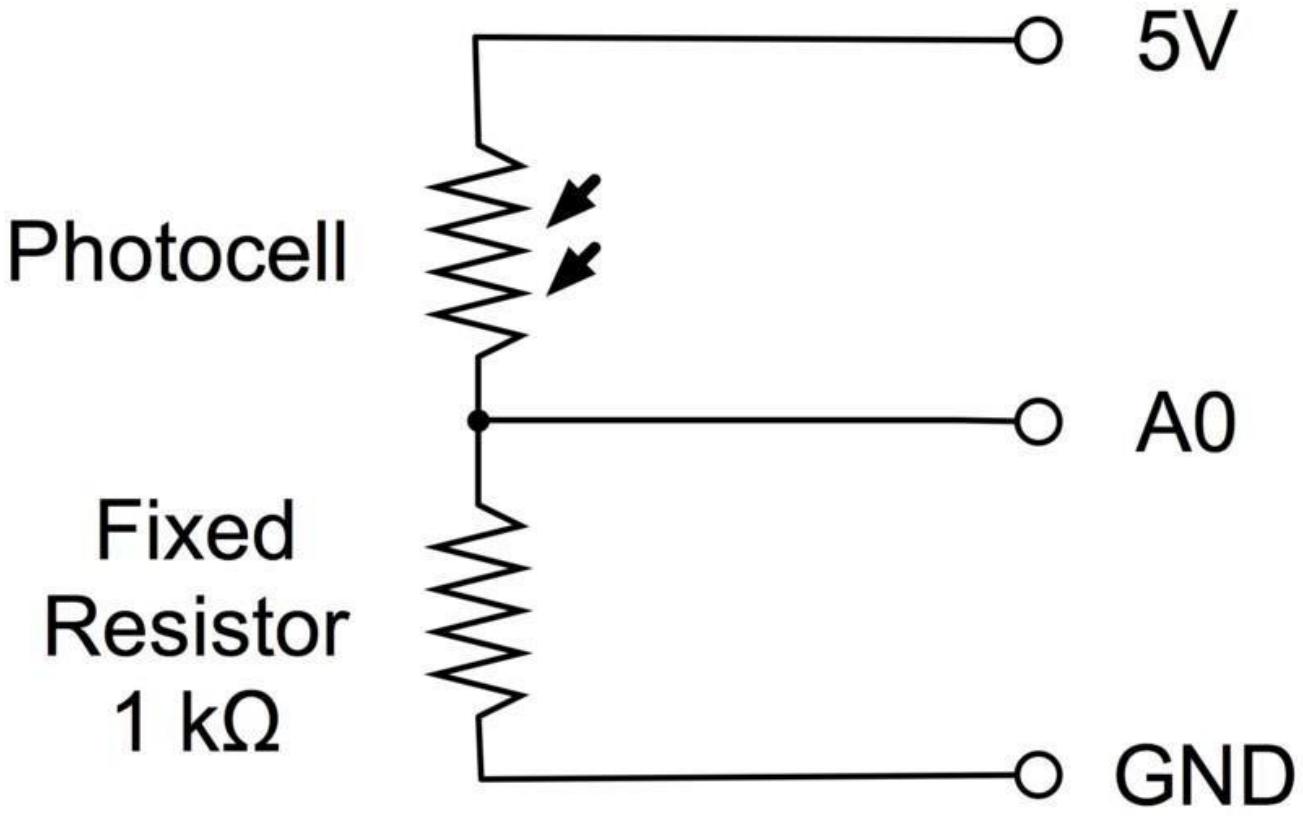
PHOTOCELL:

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

This one has a resistance of about 50 kΩ in near darkness and 500 Ω in bright light.

To convert this varying value of resistance into something we can measure on an UNO R3 board's analog input, it needs to be converted into a voltage.

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

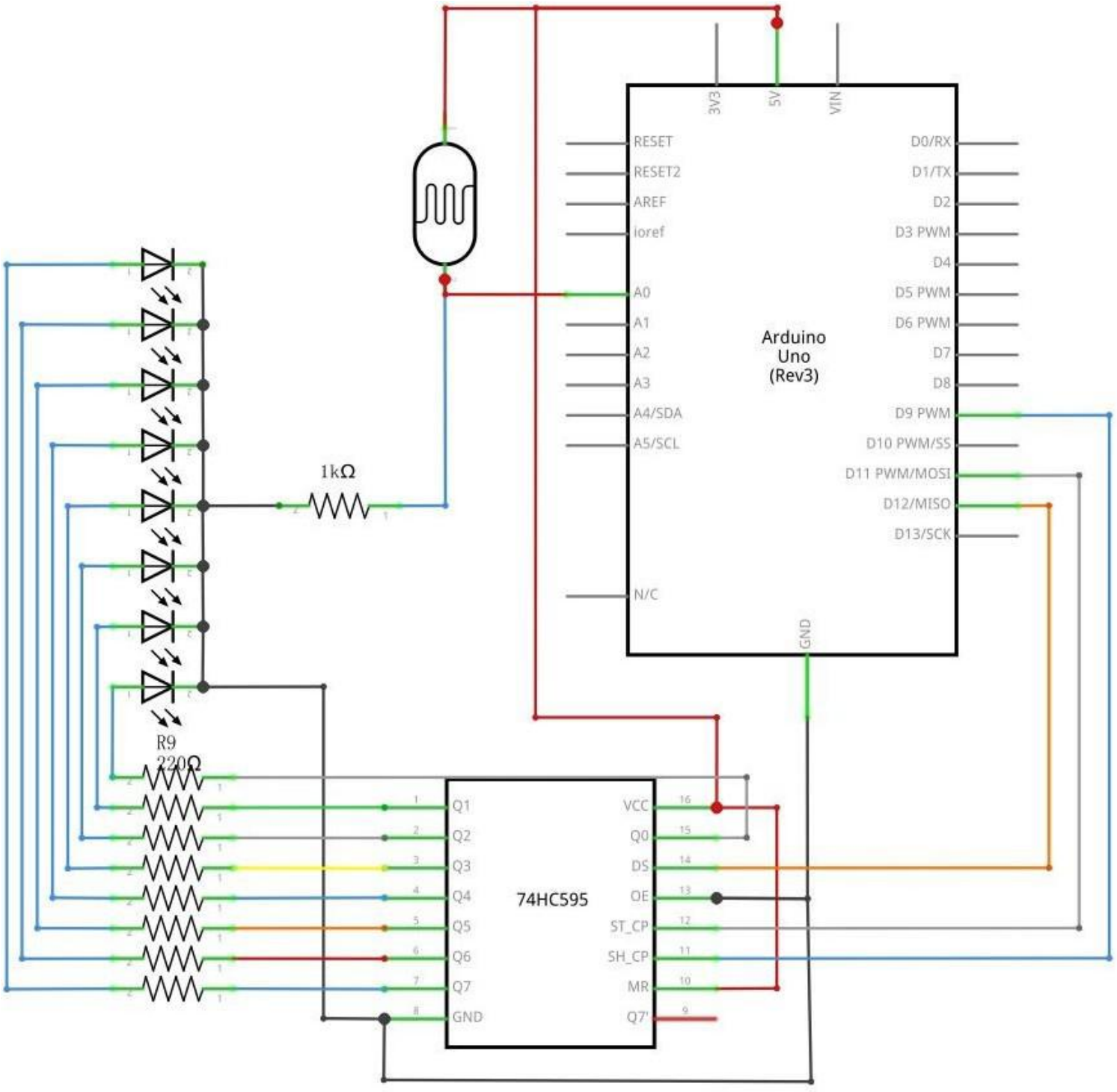


The resistor and photocell together behave like a pot. When the light is very bright, then the resistance of the photocell is very low compared with the fixed value resistor, and so it is as if the pot were turned to maximum.

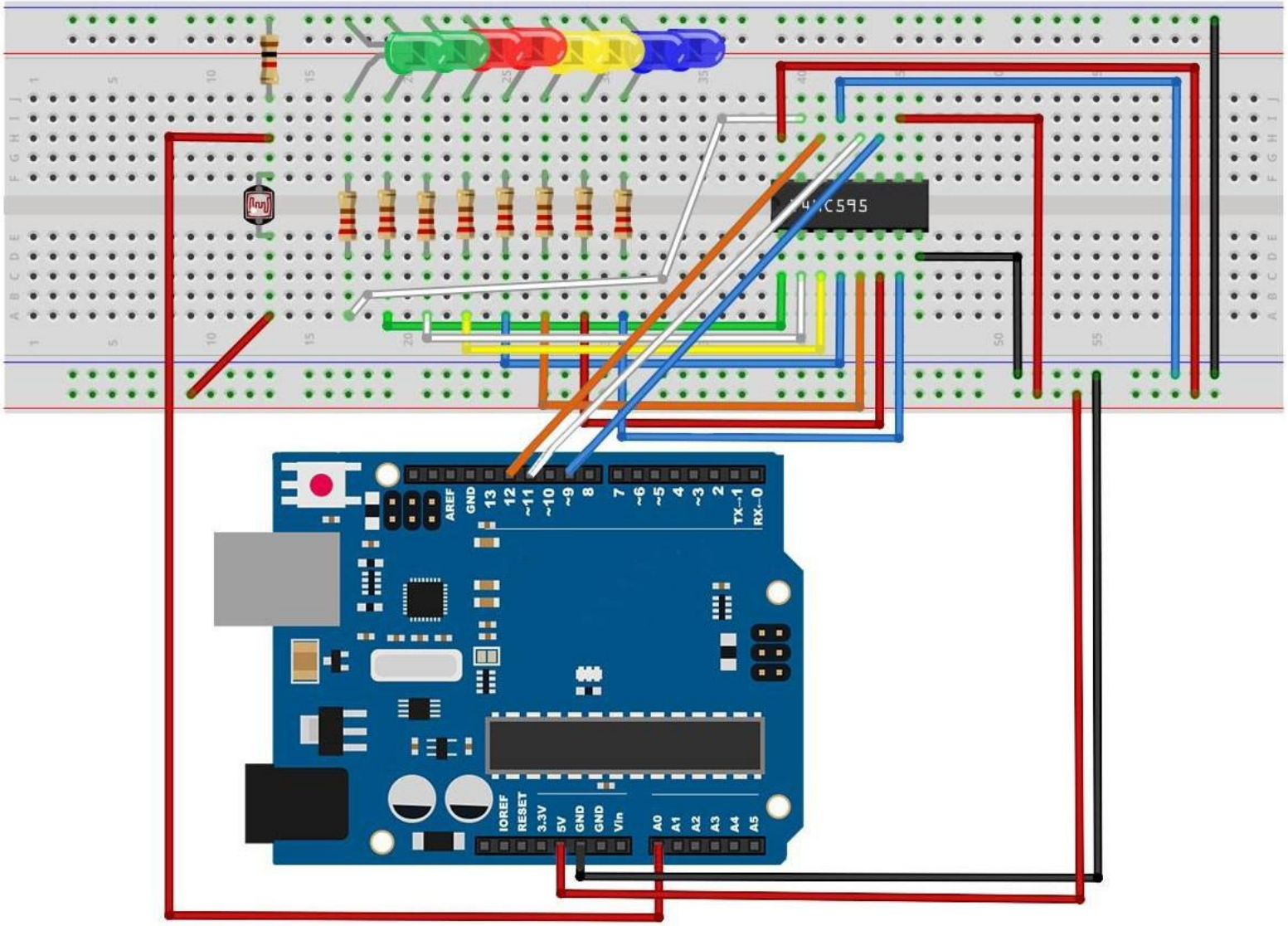
When the photocell is in dull light, the resistance becomes greater than the fixed 1 kΩ resistor and it is as if the pot were being turned towards GND.

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

Connection
Schematic



Wiring diagram



Code

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

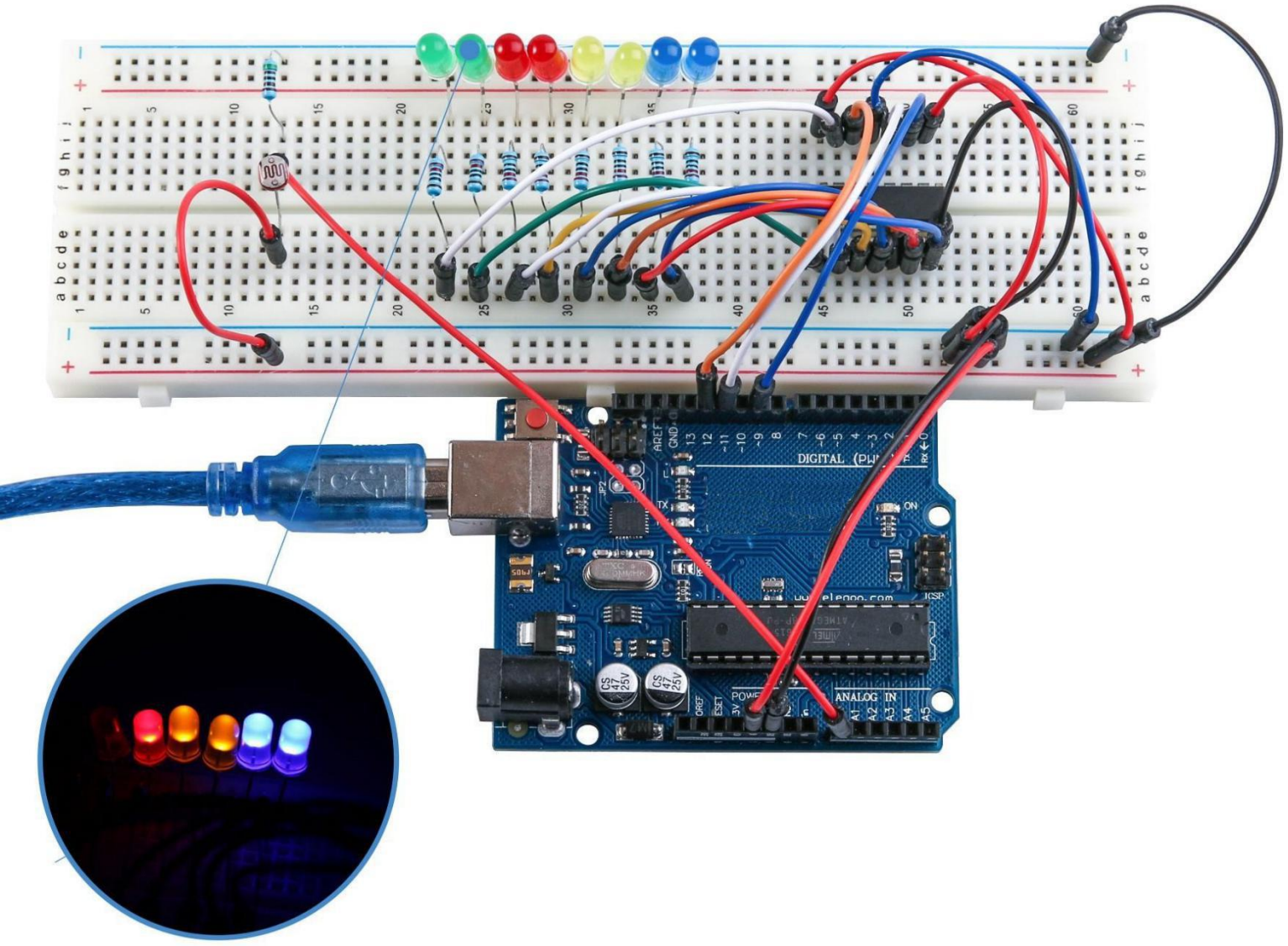
The first thing to note is that we have changed the name of the analog pin to be 'lightPin' rather than 'potPin' since we no longer have a pot connected.

The only other substantial change to the sketch is the line that calculates how many of the LEDs to light:

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

This time, we divide the raw reading by 57 rather than 114. In other words, we divide it by half as much as we did with the pot to split it into nine zones, from no LEDs lit to all eight lit. This extra factor is to account for the fixed 1 kΩ resistor. This means that when the photocell has a resistance of 1 kΩ (the same as the fixed resistor), the raw reading will be 1023 / 2 = 511. This will equate to all the LEDs being lit and then a bit (numLEDSLit) will be 8.

Example picture



Lesson 19 74HC595 And Segment Display

Overview

After learning Lesson 24、25 and Lesson 26, we will use the 74HC595 shift register to control the segment display. The segment display will show number from 9-0.

Component Required:

(1) x Uno R3

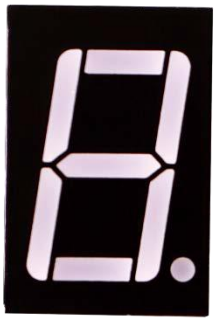
(1) x 830 tie-points breadboard

(1) x 74HC595 IC

(1) x 1 Digit 7-Segment Display

(8) x 220 ohm resistors

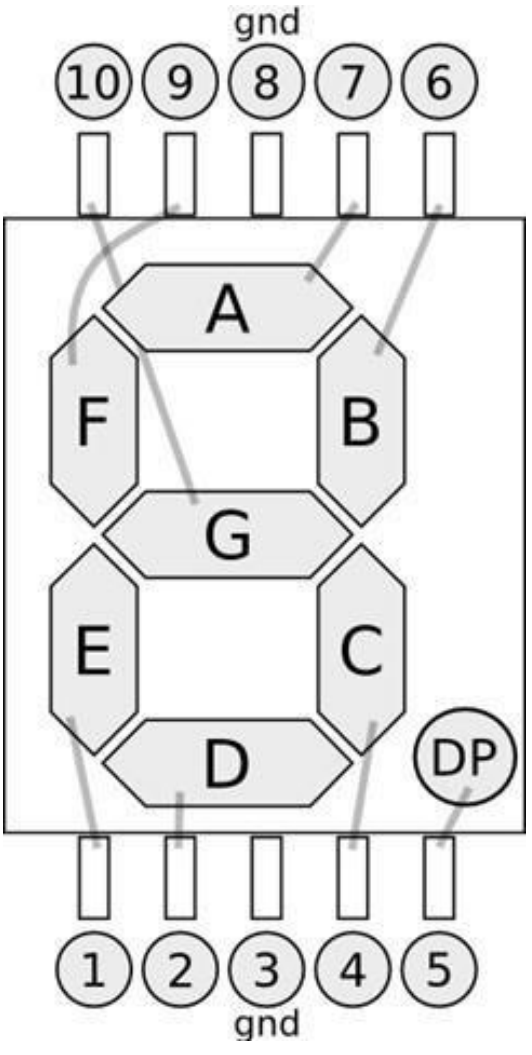
(26) x M-M wires (Male to Male jumper wires)



Component Introduction

Seven segment display

Below is the seven-segment pin diagram.

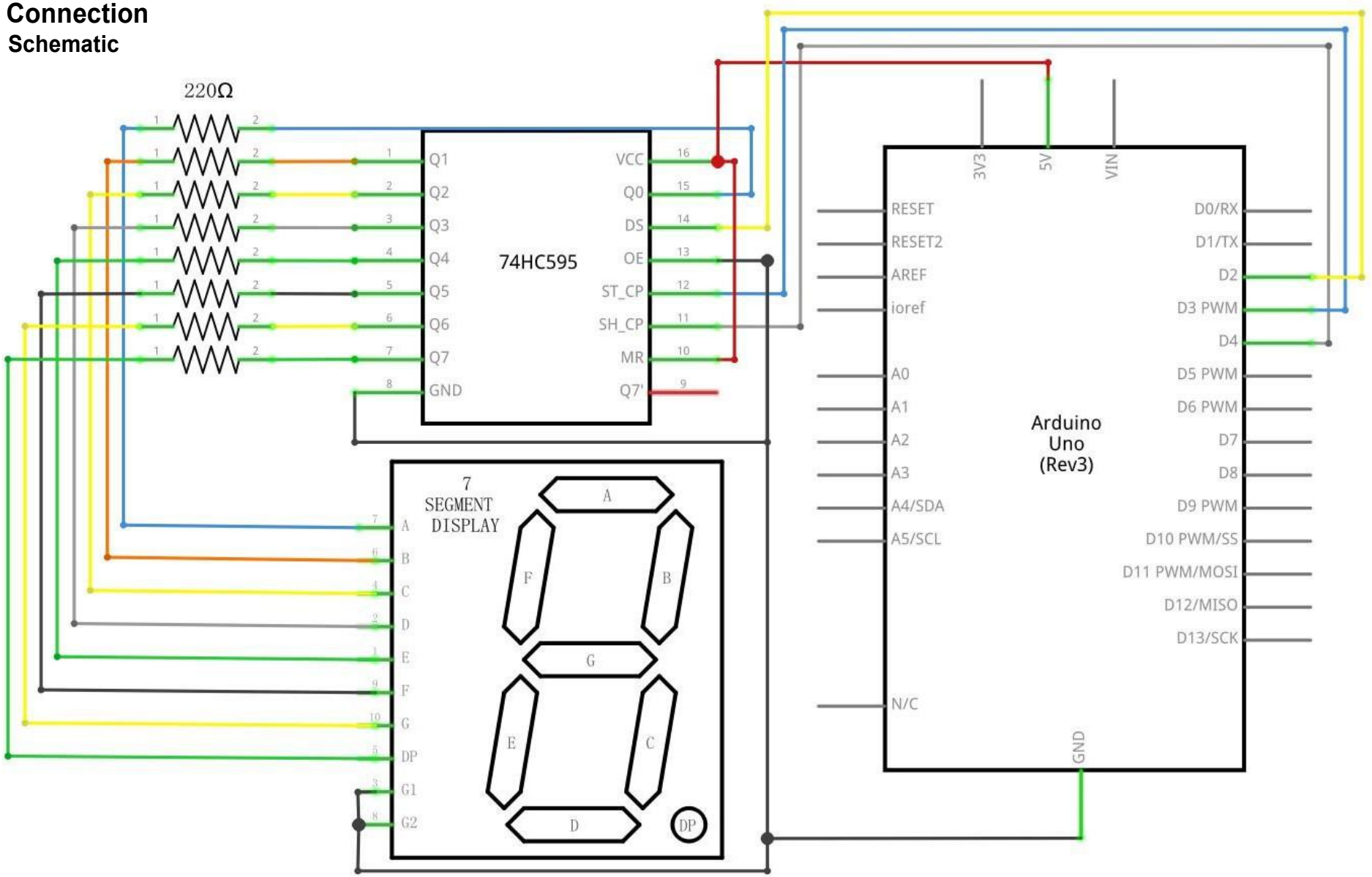


0-9 ten digits correspond with each segment are as follows (the following table applies common cathode seven segment display device, if you are using a common anode, the table should be replaced every 1 0 0 should all replaced by 1):

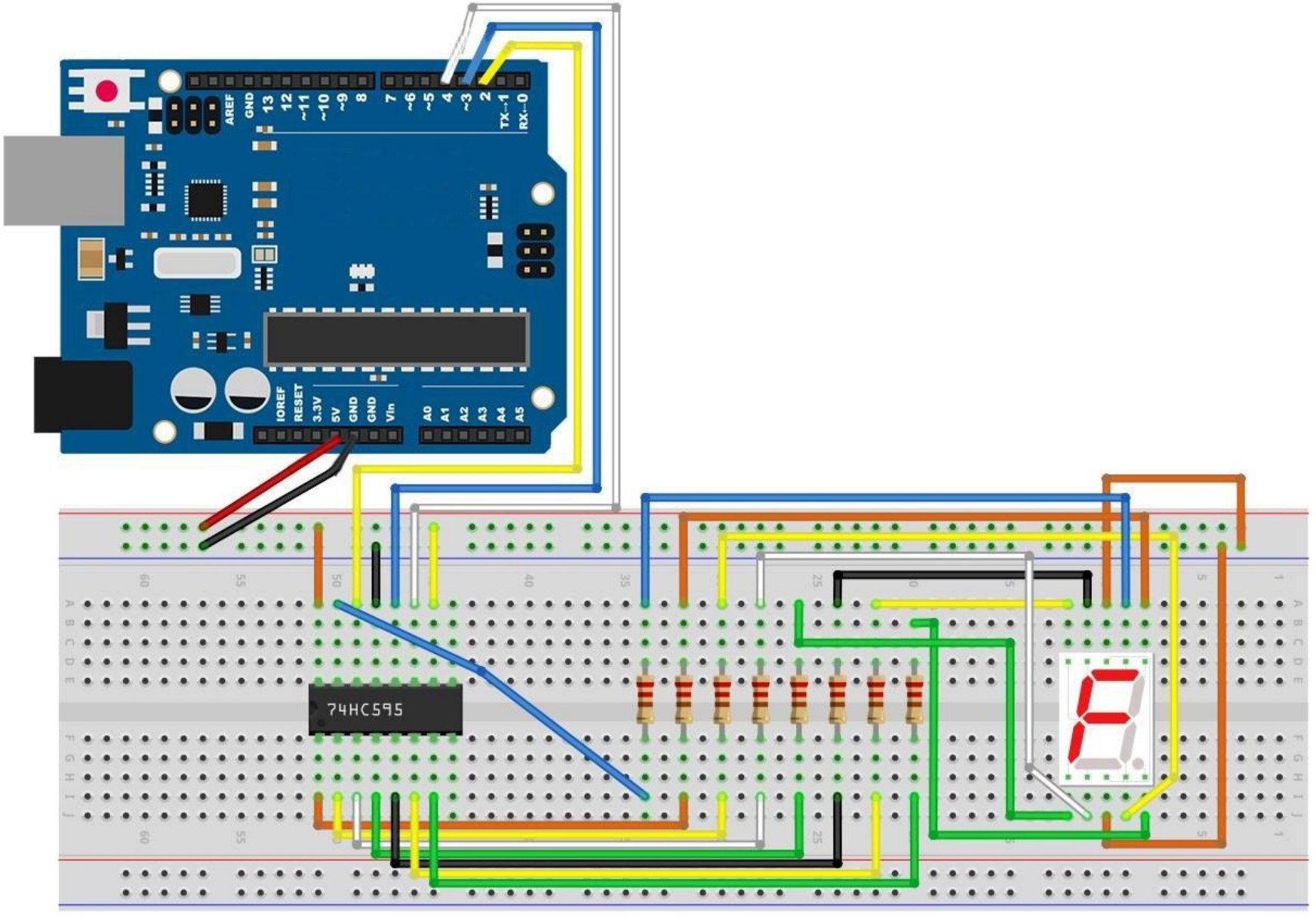
Display digital	dp	a	b	c	d	e	f	g
0	0	1	1	1	1	1	1	0
1	0	0	1	1	0	0	0	0
2	0	1	1	0	1	1	0	1
3	0	1	1	1	1	0	0	1
4	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1
6	0	1	0	1	1	1	1	1
7	0	1	1	1	0	0	0	0
8	0	1	1	1	1	1	1	1
9	0	1	1	1	1	0	1	1

Connection

Schematic



Wiring diagram



The following table shows the seven-segment display 74HC595 pin correspondence 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 74HC595

First, the wiring is connected to power and ground:

VCC (pin 16) and MR (pin 10) connected to 5V

GND (pin 8) and OE (pin 13) to ground

Connection DS, ST_CP and SH_CP pin:

DS (pin 14) connected to UNO R3 board pin 2 (the figure below the yellow line)

ST_CP (pin 12, latch pin) connected to UNO R3 board pin 3 (FIG blue line below)

SH_CP (pin 11, clock pin) connected to UNO R3 board pin 4 (the figure below the white line)

Step two: Connect the seven segment display

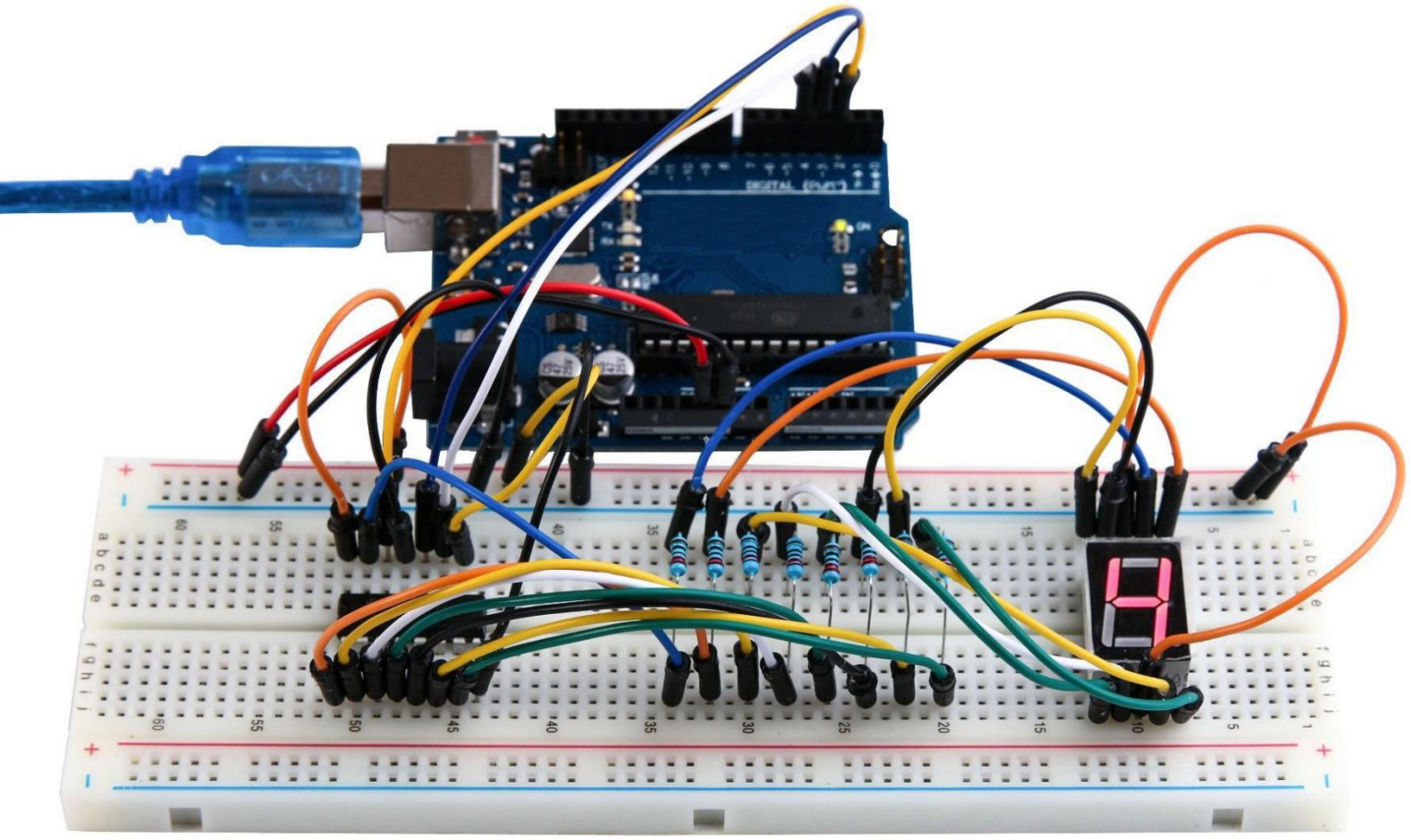
The seven-segment display 3, 8 pin to UNO R3 board GND (This example uses the common cathode, if you use the common anode, please connect the 3, 8 pin to UNO R3 board + 5V)

According to the table above, connect the 74HC595 Q0 ~ Q7 to seven-segment display corresponding pin (A ~ G and DP), and then each foot in a 220 ohm resistor in series.

Code

After wiring, please open the program in the code folder- Lesson 27 74HC595 And Segment Display and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

Example picture



Lesson 20 Four Digital Seven Segment Display

Overview

In this lesson, you will learn how to use a 4-digit 7-segment display.

When using 1-digit 7-segment display, please notice that if it is common anode, the common anode pin connects to the power source; if it is common cathode, the common cathode pin connects to the GND.

When using 4-digit 7-segment display, the common anode or common cathode pin is used to control which digit is displayed. Even though there is only one digit working, the principle of Persistence of Vision enables you to see all numbers displayed because each the scanning speed is so fast that you hardly notice the intervals.

Component Required:

(1) x Uno R3

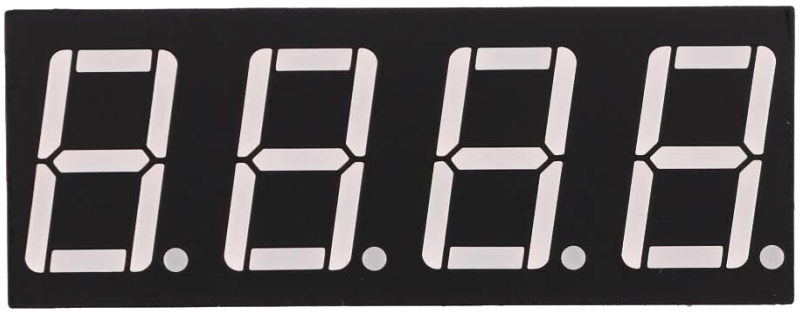
(1) x 830 tie-points breadboard

(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)

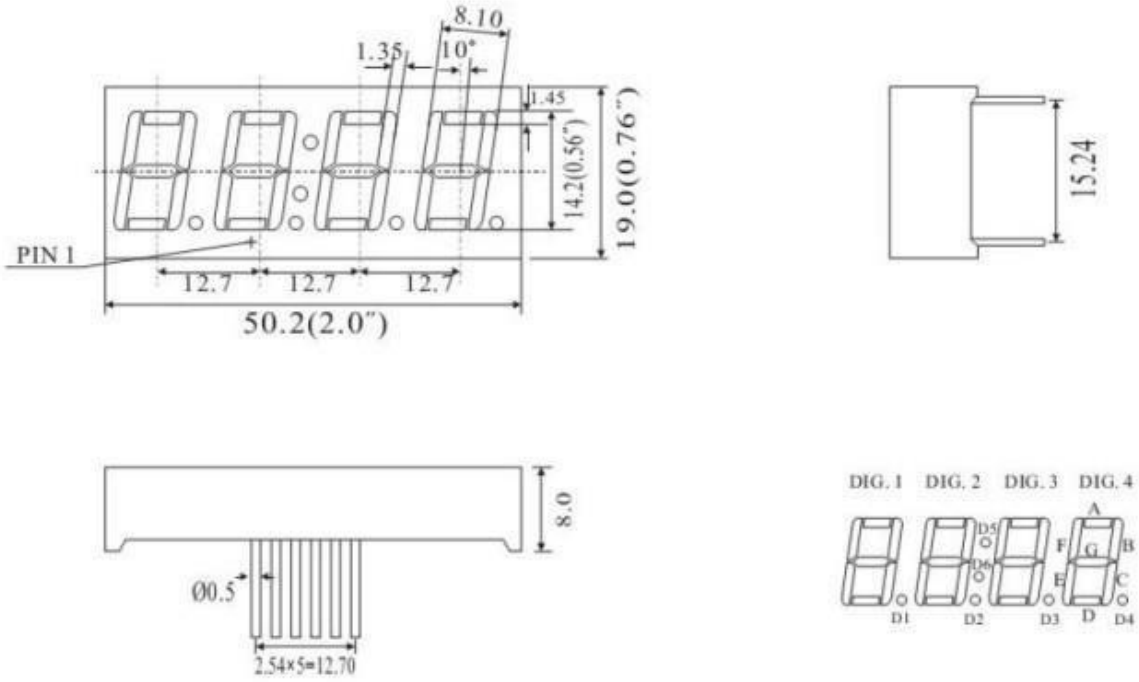


Component Introduction

Four Digital Seven segment display

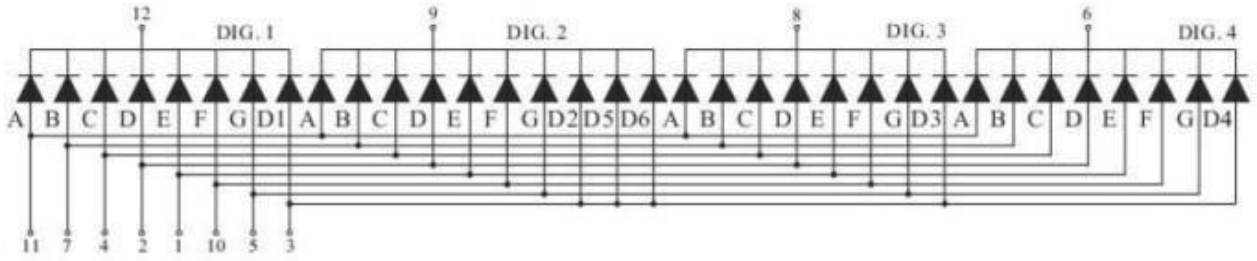
Package Dimensions

CPS05643AB

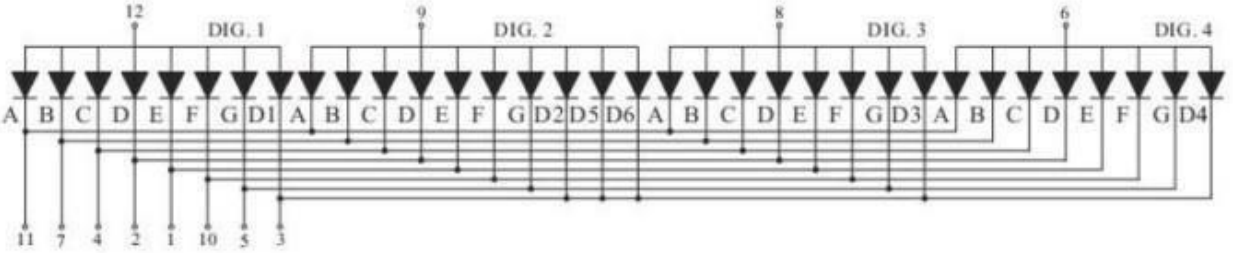


UNIT: MM(INCH) TOLERANCE: ±0.25(0.01")

Internal Circuit Diagram



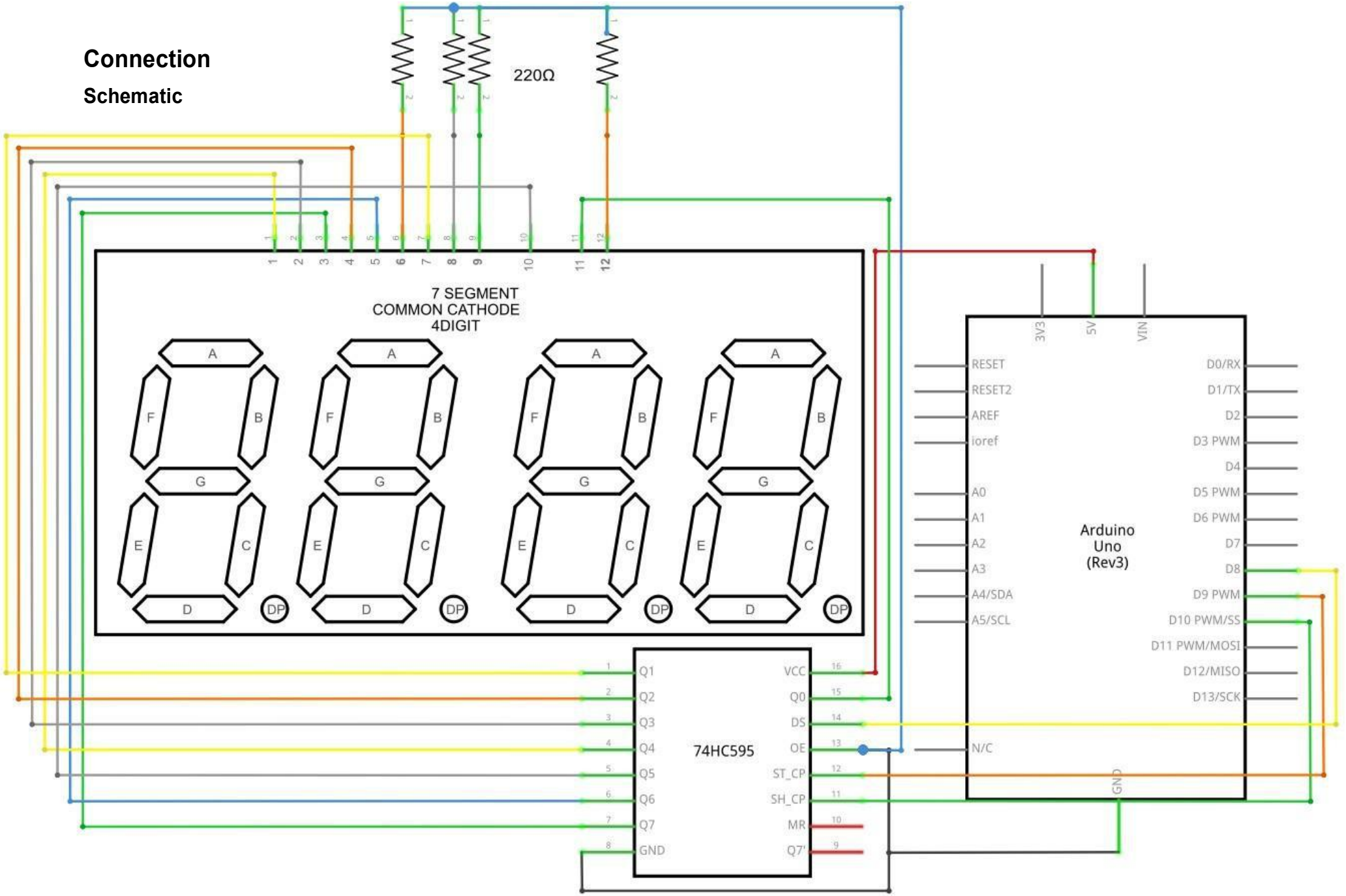
5643A



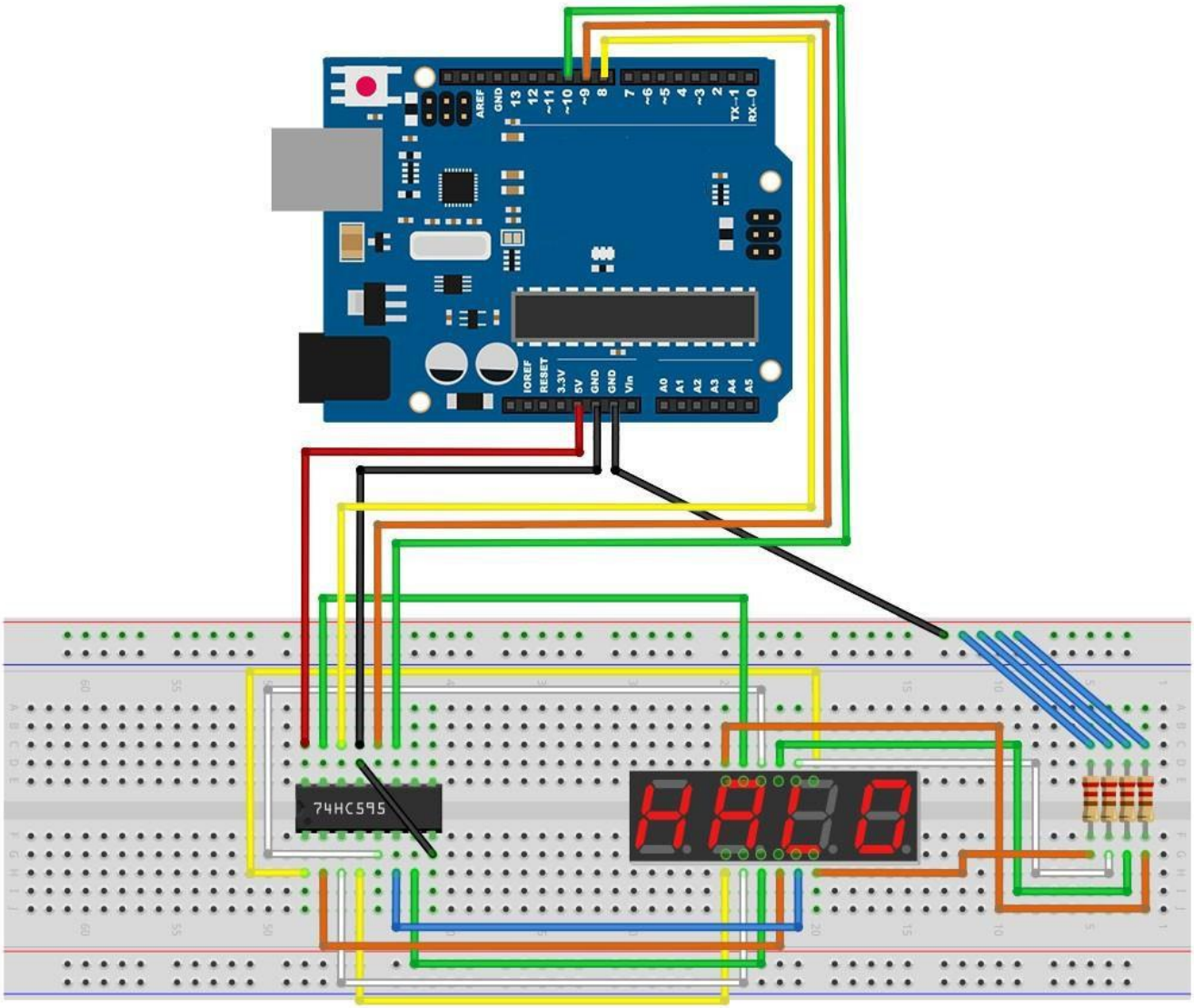
5643B

Connection

Schematic



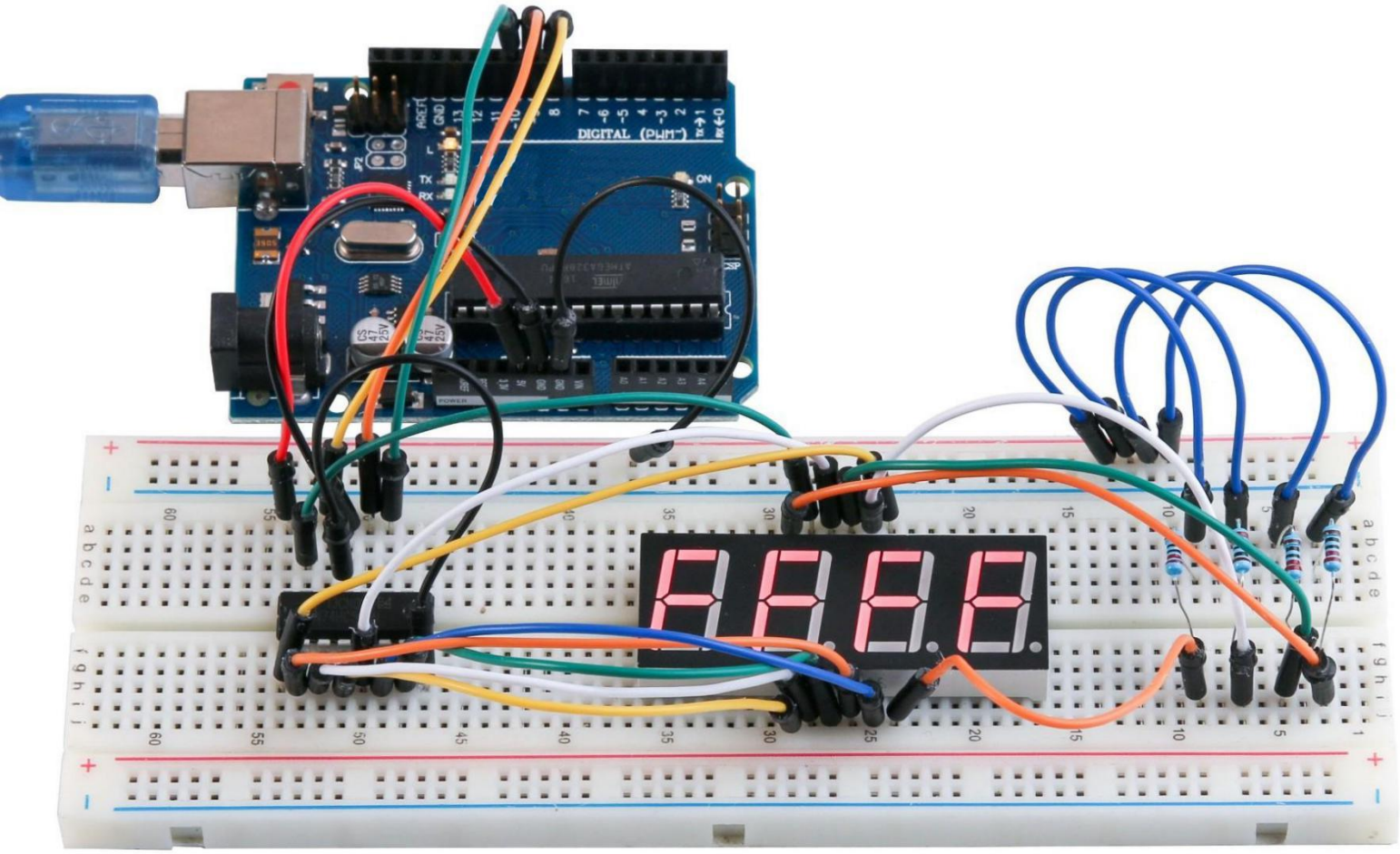
Wiring diagram



Code

After wiring, please open the program in the code folder- Lesson 28 Four Digital Seven Segment Display and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

Example picture



Lesson 21 DC Motors

Overview

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

Component Required:

(1) x 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 9V1A adapter

(1) x Power Supply Module

Component Introduction

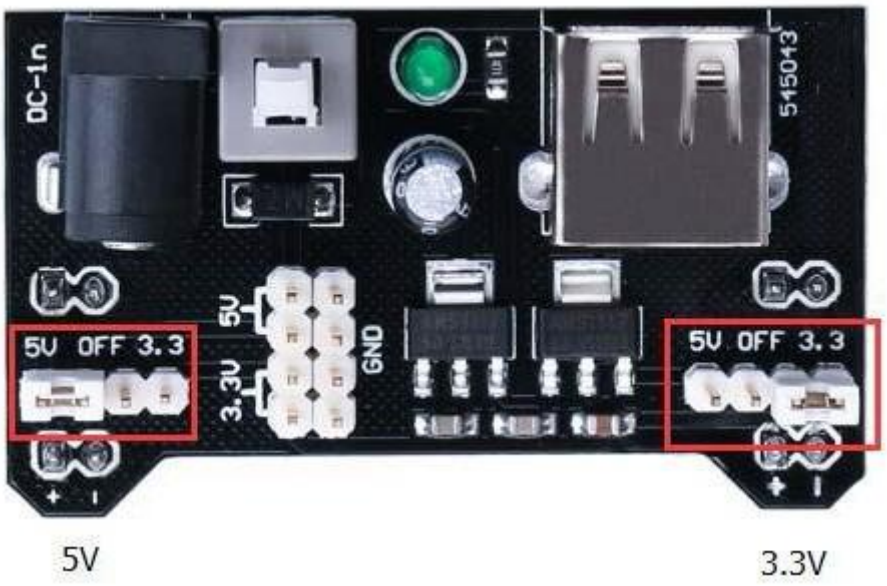
Breadboard Power Supply

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

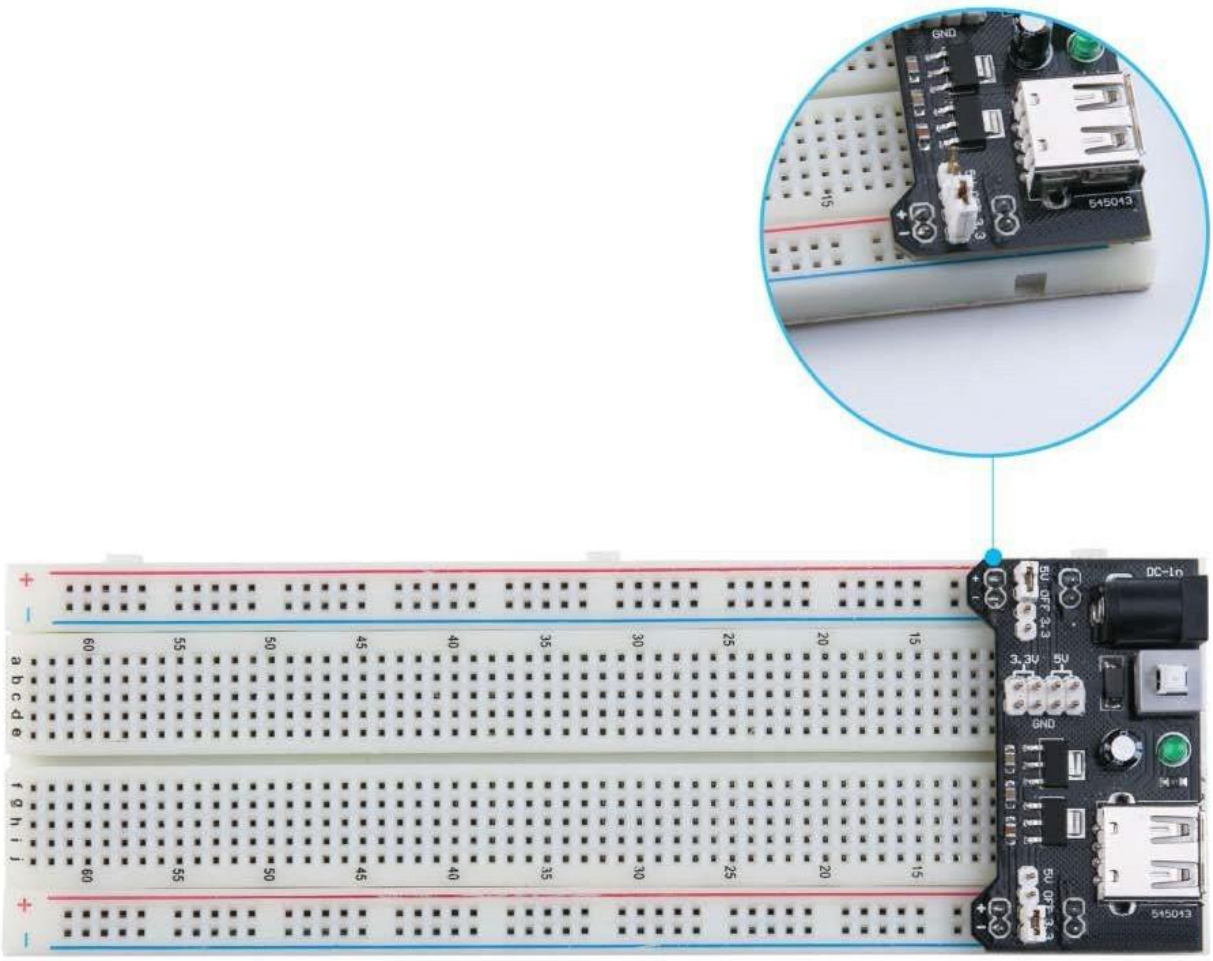
Product Specifications:

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

Setting up output voltage:



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

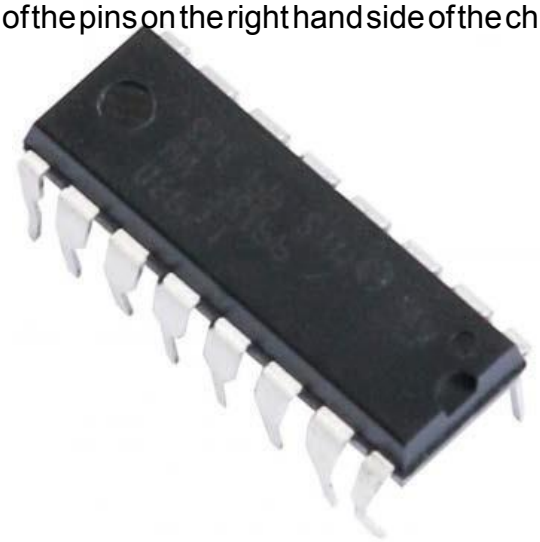


Important note:

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

L293D

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



Product Specifications:

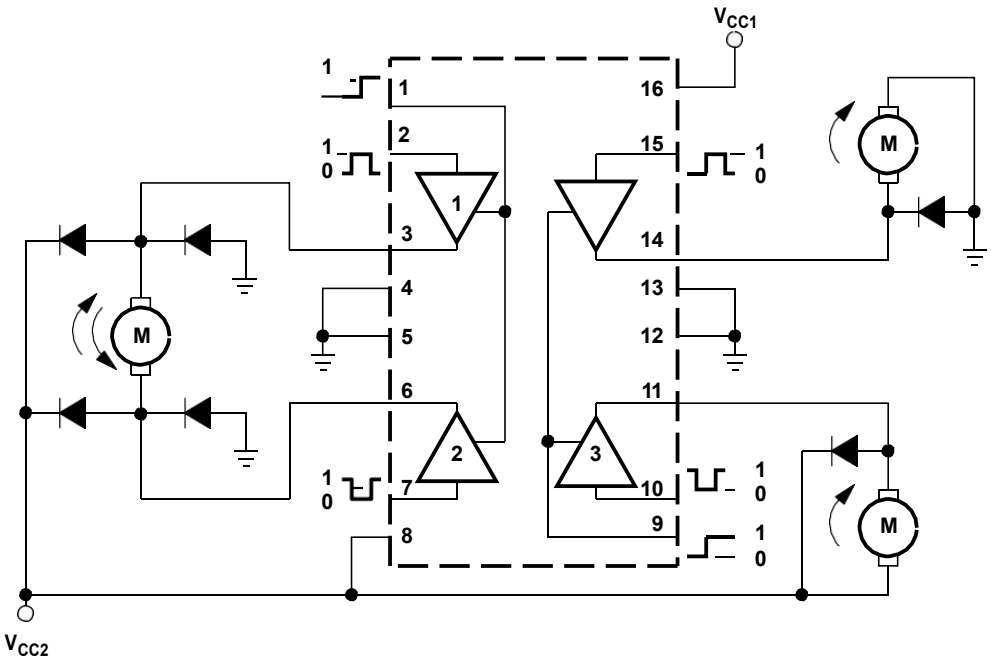
- Featuring Unitrode L293 and L293D Products Now From Texas Instruments
- Wide Supply-Voltage Range: 4.5 V to 36 V
- Separate Input-Logic Supply
- Internal ESD Protection
- Thermal Shutdown
- High-Noise-Immunity Inputs
- Functionally Similar to SGS L293 and SGS L293D
- Output Current 1 A Per Channel (600 mA for L293D)
- Peak Output Current 2 A Per Channel (1.2 A for L293D)
- Output Clamp Diodes for Inductive 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. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

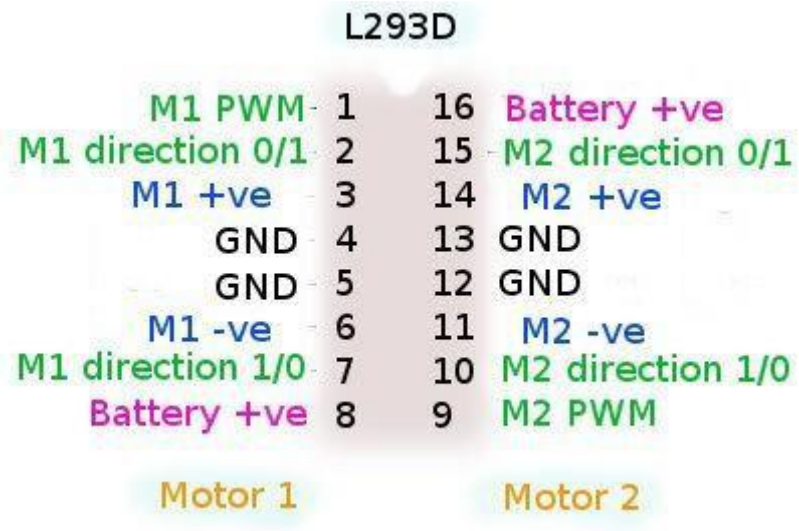
All inputs are TTL compatible. Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN. When an enable input is high, the associated drivers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

Block diagram



I got fed up with indecipherable pinout diagrams within datasheets, so have designed my own that I think gives more pertinent information.

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, the right hand side deals with a second motor.

Yes, you can run it with only one motor connected.

Arduino Connections

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

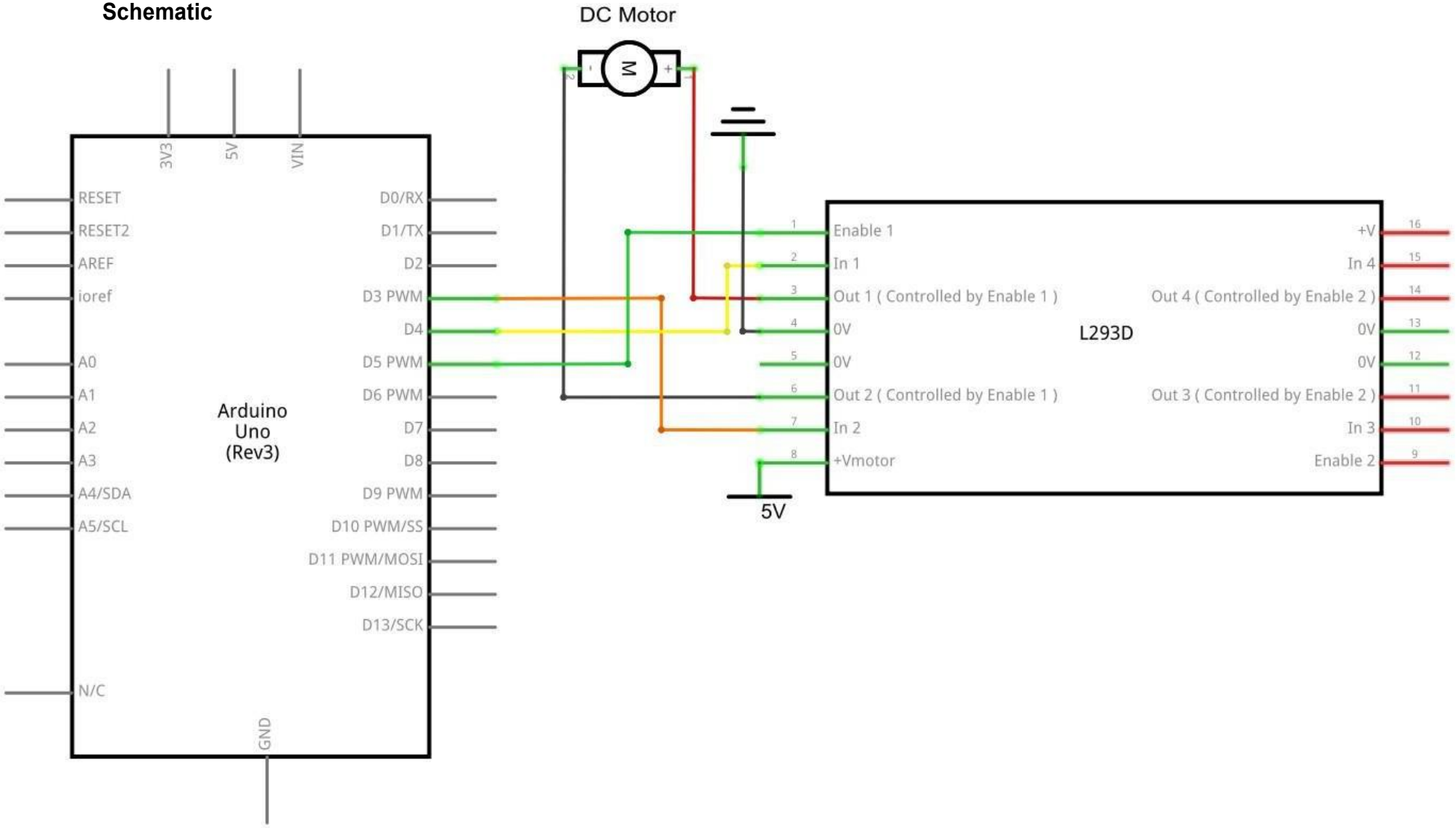
max speed.

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

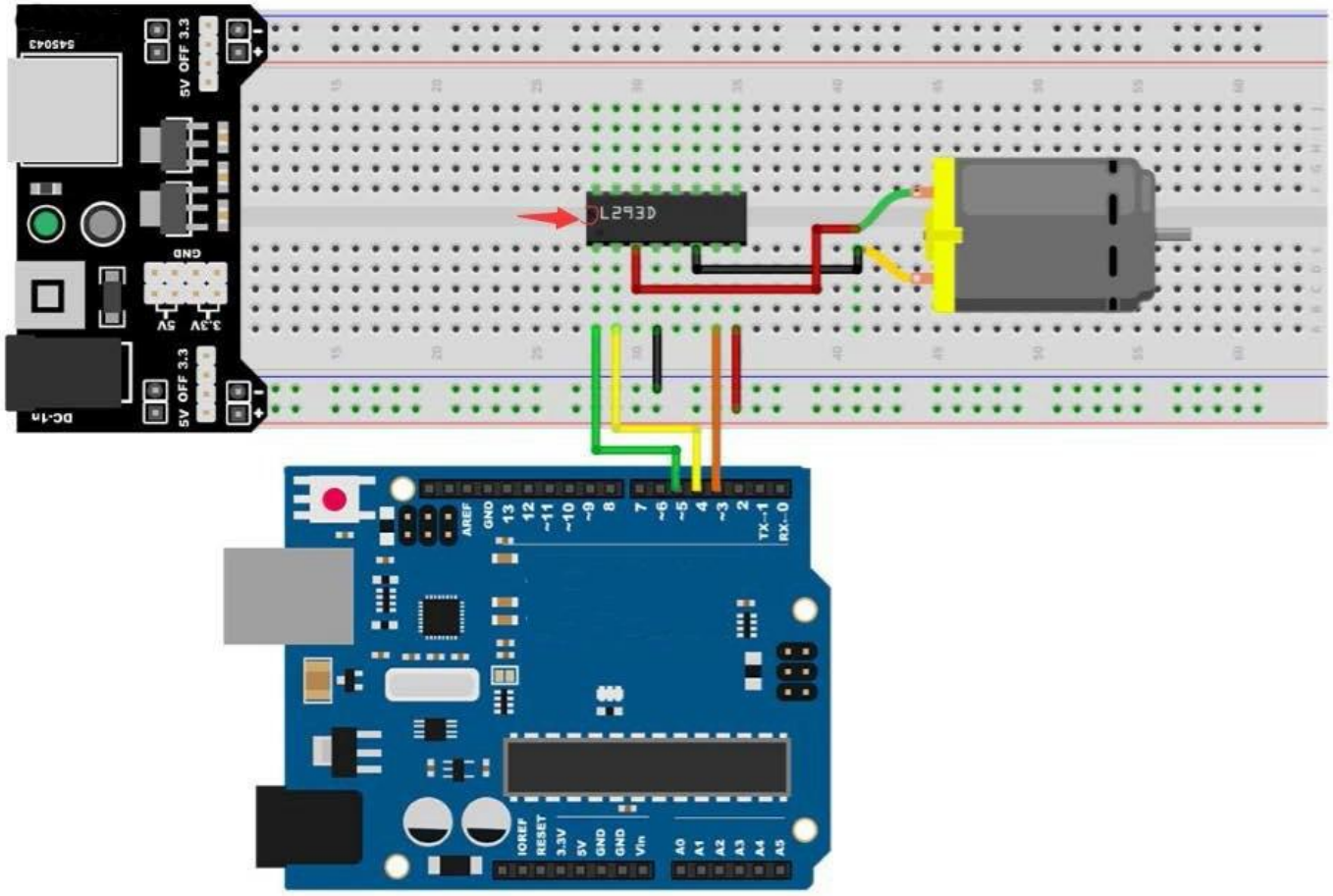
Reverse the outputs to LOW and HIGH, and the motor will spin in the other direction.

Connection

Schematic

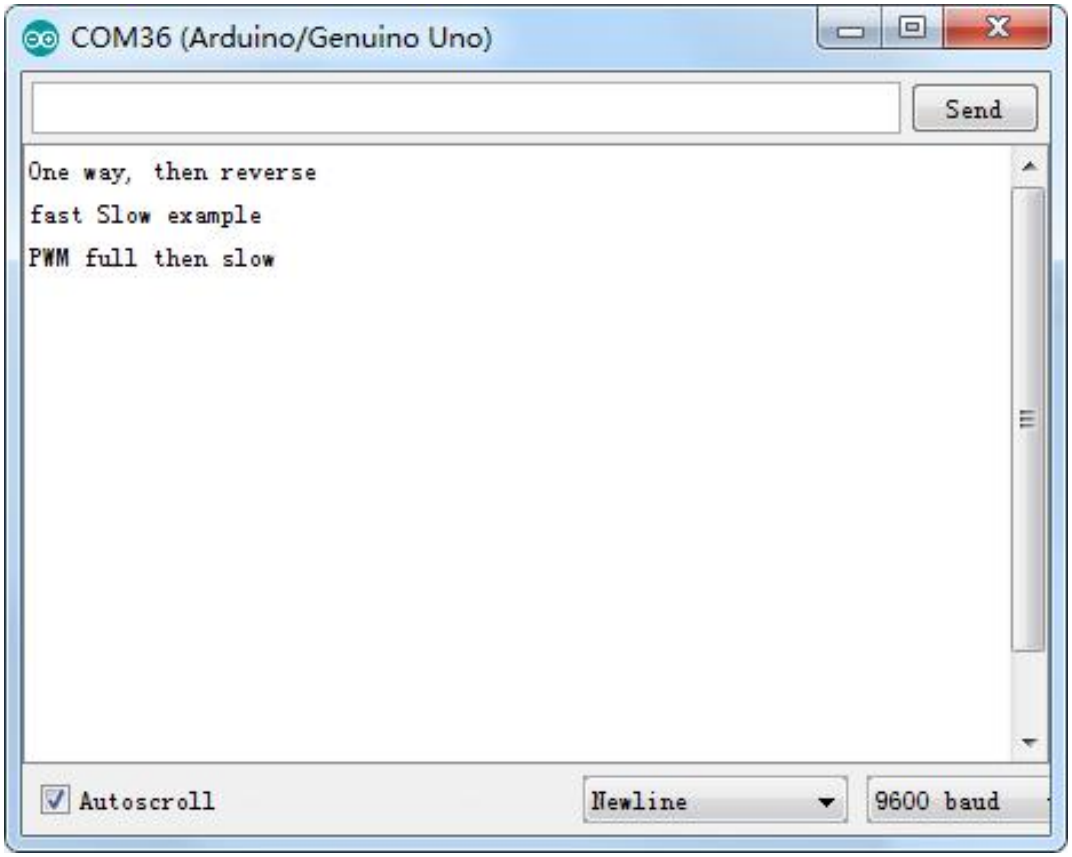


Wiring diagram



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

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



Code

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

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

Lesson 22 Relay

Overview

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

Component Required:

(1) x 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)

Component Introduction

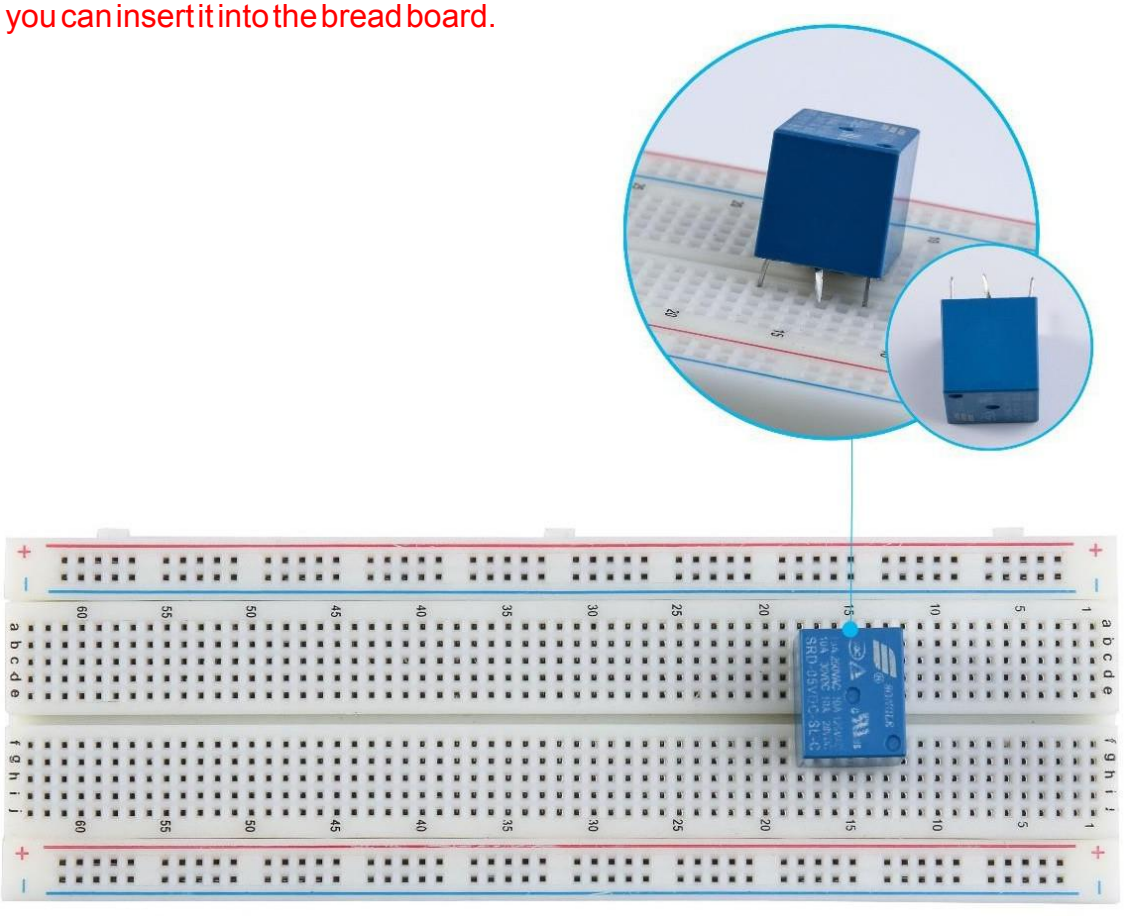
Relay:

A relay is an electrically operated switch. Many relays use an electromagnet to mechanically operate a switch, but other operating principles are also used as in solid-state relays. Relays are used where it is necessary to control a circuit by 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. They repeated the signal coming in from one circuit and re-transmitted it on another circuit. Relays were used extensively in telephone exchanges and early computers to perform logical operations.

A type of relay that can handle the high power required to directly control an electric motor or other loads is called a contactor. Solid-state relays control power circuits with no moving parts, instead using a semiconductor device to perform the switching. Relays with calibrated operating characteristics and sometimes multiple operating coils are used to protect electrical circuits from overload or faults. In modern electric power systems, these functions are performed by digital instruments called "protective relays".

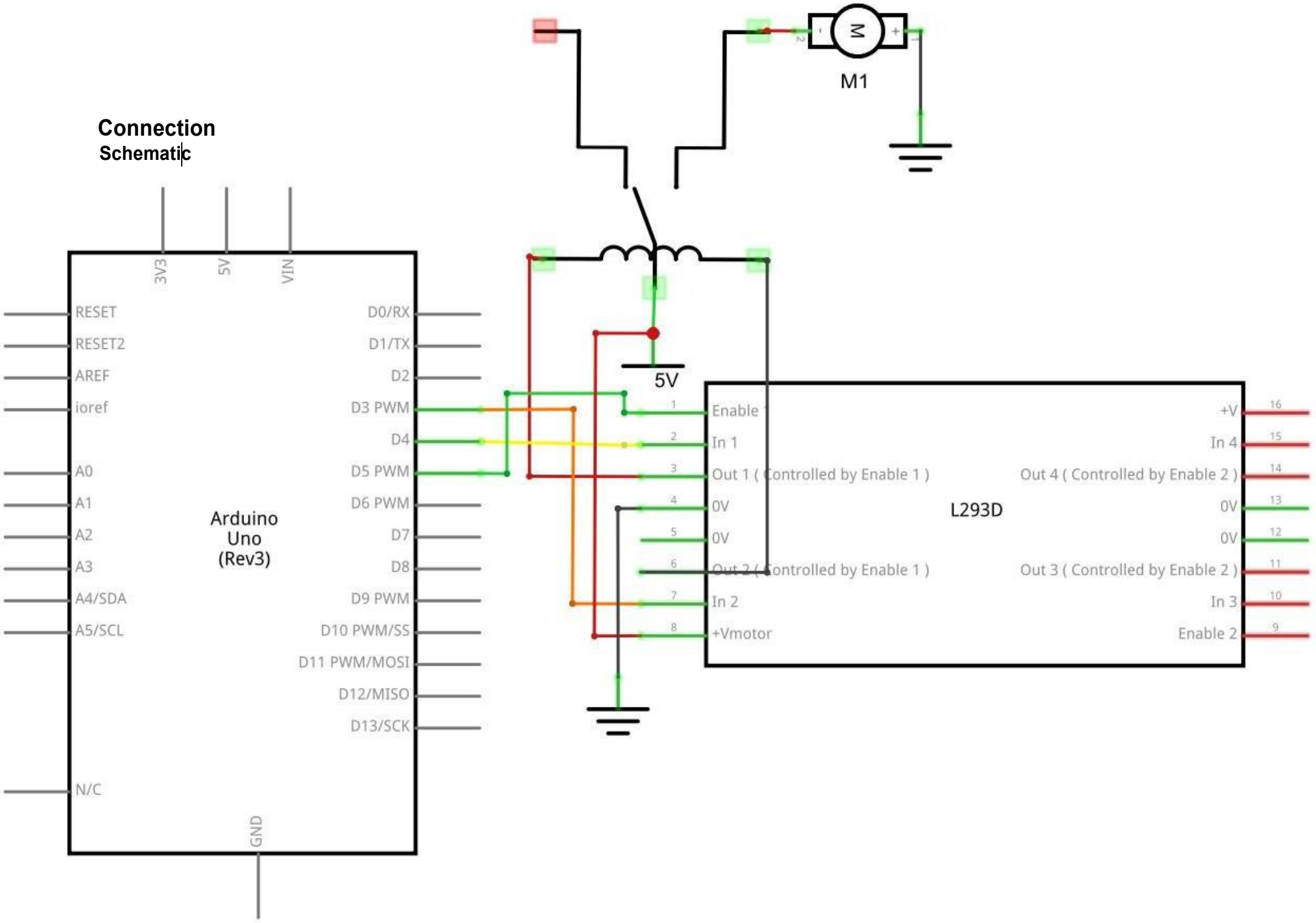
Below is the schematic of how to drive relay with Arduino.

You may be confused about how to insert the relay into the bread board. As the picture below shows, you will have to bend one of the pins of the relay slightly then you can insert it into the bread board.

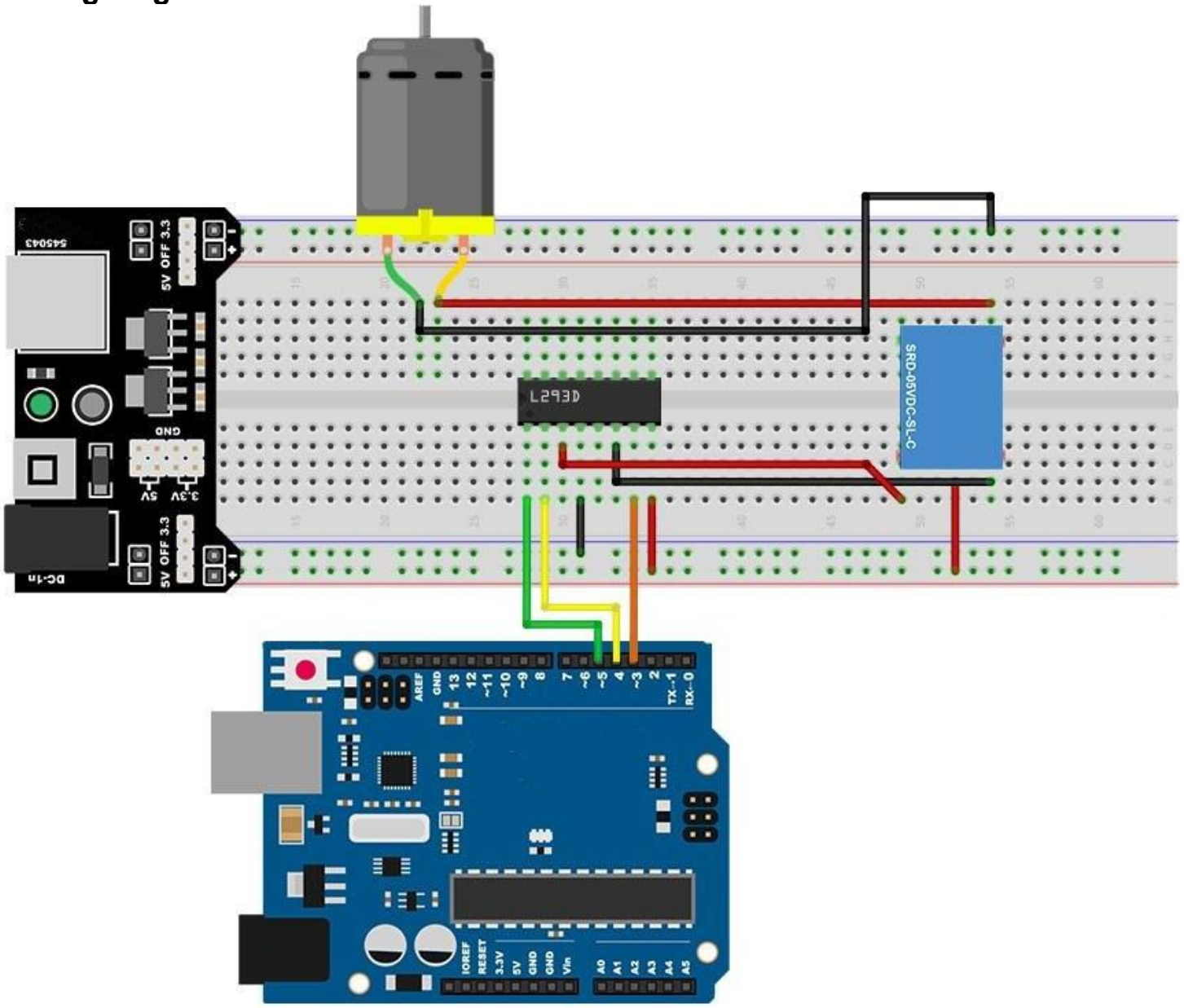


Connection

Schematic



Wiring diagram



Code

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

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

Example picture

Lesson 23 Stepper Motor

Overview

In this lesson, you will learn a fun and easy way to drive a stepper motor. The stepper we are using comes with its own driver board making it easy to connect to our UNO.

Component Required:

(1) x Uno R3

(1) x 830 tie-points breadboard

(1) x ULN2003 stepper motor driver module

(1) x Stepper motor

(1) x 9V1A Adapter

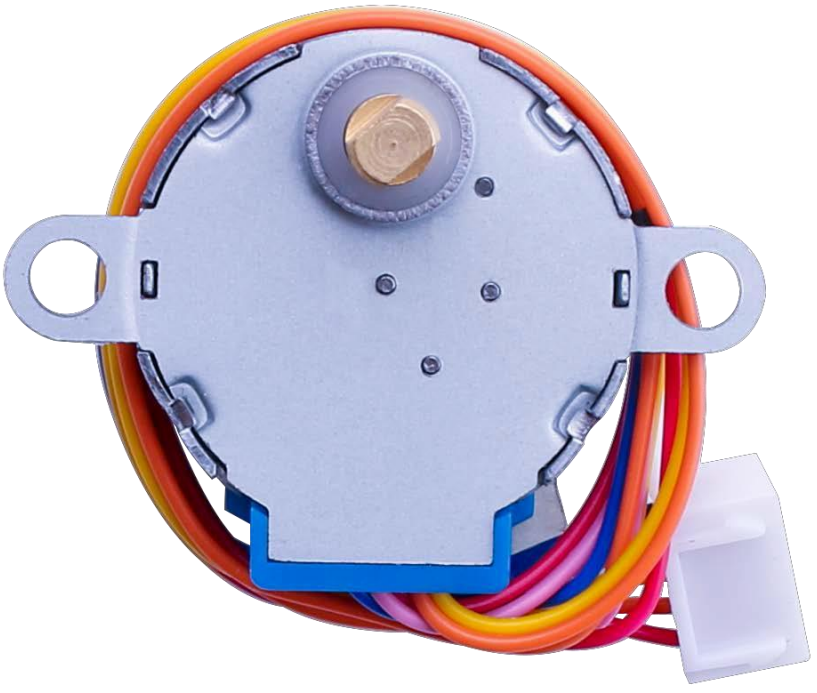
(1) x Power supply module

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

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

Component Introduction

Stepper Motor

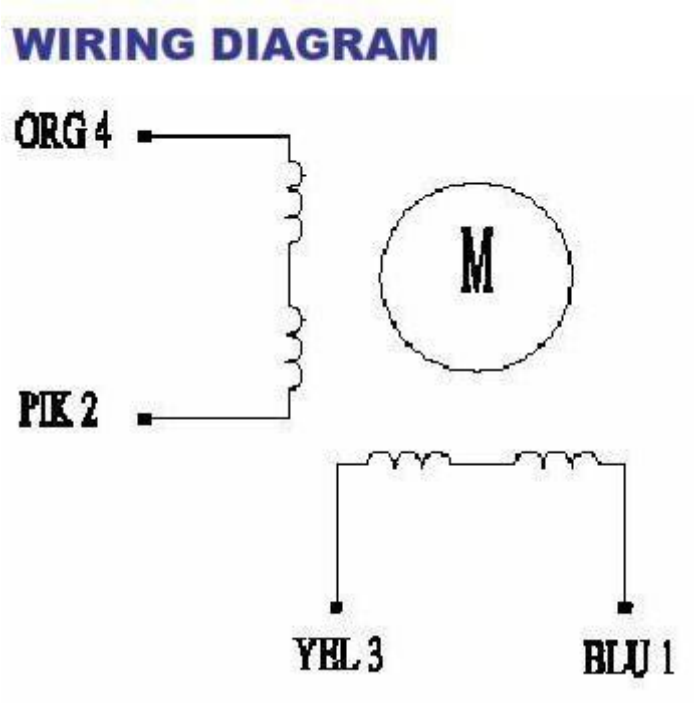


A stepper motor is an electromechanical device which converts electrical pulses into discrete mechanical movements. The shaft or spindle of a stepper motor rotates in discrete step increments when electrical command pulses are applied to it in the proper sequence. The motors rotation has several direct 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. Open loop control 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. Your position is known simply by keeping track of the input step pulses.

Stepper motor 28BYJ-48 Parameters

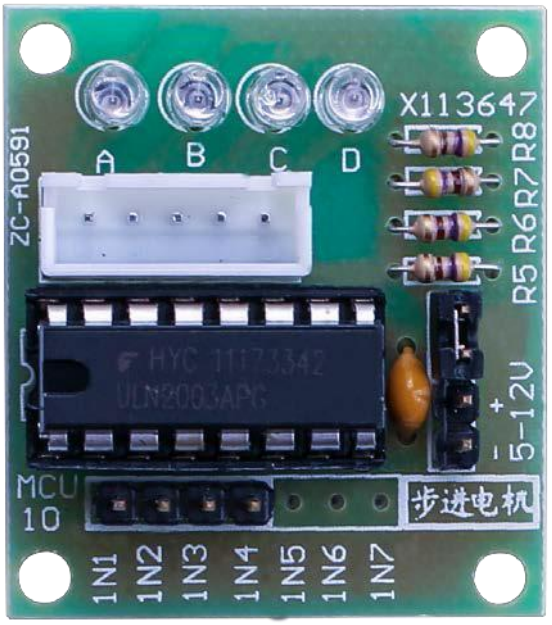
- Model: 28BYJ-48
- Rated voltage: 5VDC
- Number of Phase: 4
- Speed Variation Ratio: 1/64
- Stride Angle: 5.625°/64
- Frequency: 100Hz
- DC resistance: 50Ω±7%(25℃)
- Idle In-traction Frequency: > 600Hz
- Idle Out-traction Frequency: > 1000Hz
- In-traction Torque >34.3mN.m(120Hz)
- Self-positioning Torque >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



The bipolar stepper motor usually has four wires coming out of it. Unlike unipolar steppers, bipolar steppers have no common center connection. They have two independent sets of coils instead. You can distinguish them from unipolar steppers by measuring the resistance between the wires. You should find two pairs of wires with equal resistance. If you've got the leads of your meter connected to two wires that are not connected (i.e. not attached to the same coil), you should see infinite resistance (or nocontinuity).

ULN2003 Driver Board



Product Description

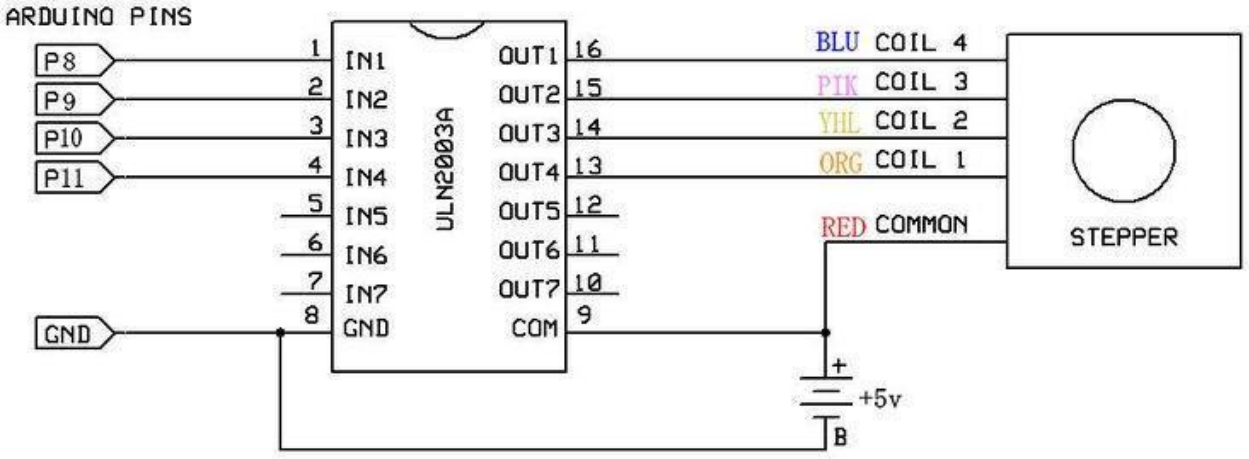
- Size: 42mmx30mm
- Use ULN2003 driver chip, 500mA
- A. B. C. D LED indicating the four phase stepper motor working condition.
- White jack is the four phase stepper motor standard jack.
- Power pins are separated
- We kept the rest pins of the ULN2003 chip for your further prototyping.

The simplest way of interfacing a unipolar stepper to Arduino is to use a breakout for ULN2003A transistor array chip. The ULN2003A contains seven Darlington transistor drivers and is somewhat like having seven TIP120 transistors all in one package. The ULN2003A can pass up to 500 mA per channel and has an internal voltage drop of about 1V when on. It also contains internal clamp diodes to dissipate voltage spikes when driving inductive loads. To control the stepper, apply voltage to each of the coils in a specific sequence.

The sequence would go like this:

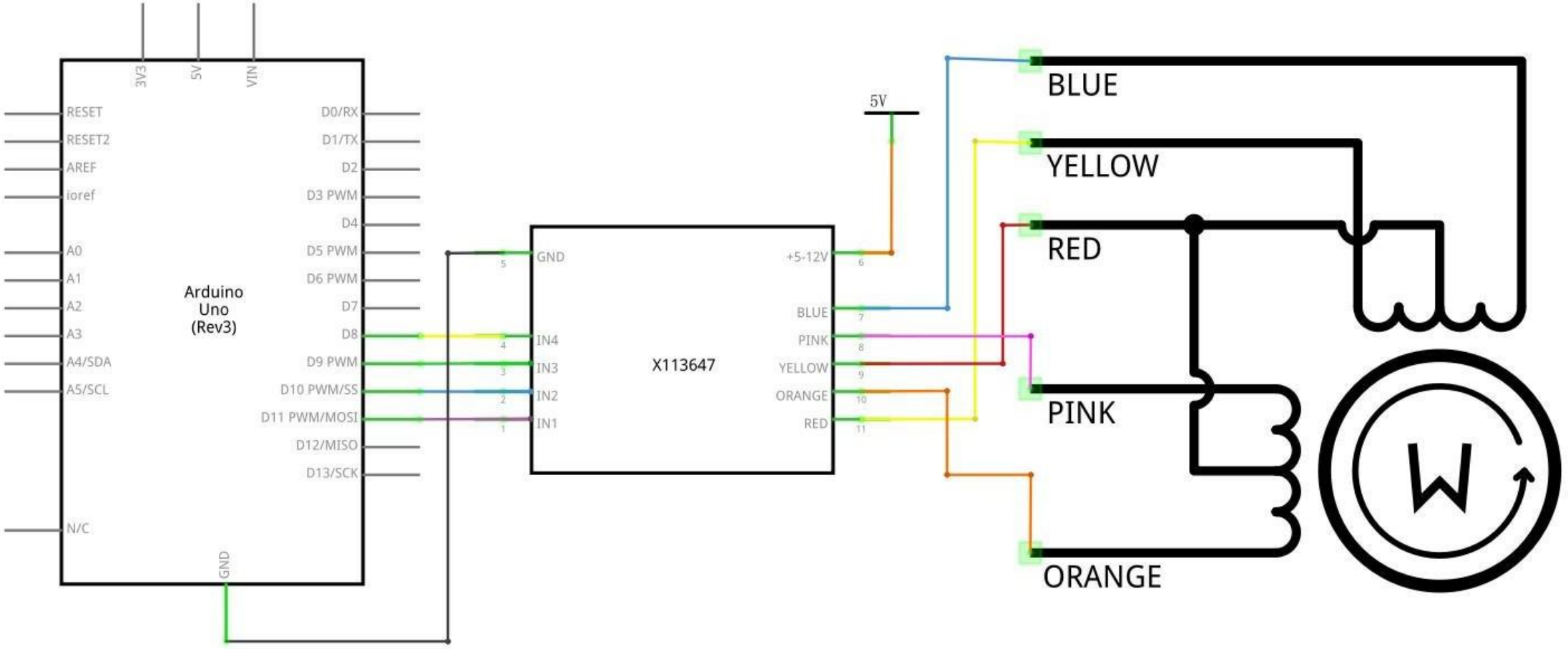
Lead Wire Color	---> CW Direction (1-2 Phase)							
	1	2	3	4	5	6	7	8
4 ORG	-	-						-
3 YEL		-	-	-				
2 PIK				-	-	-		
1 BLU						-	-	-

Here are schematics showing how to interface a unipolar stepper motor to four controller pins using a ULN2003A, and showing how to interface using four com

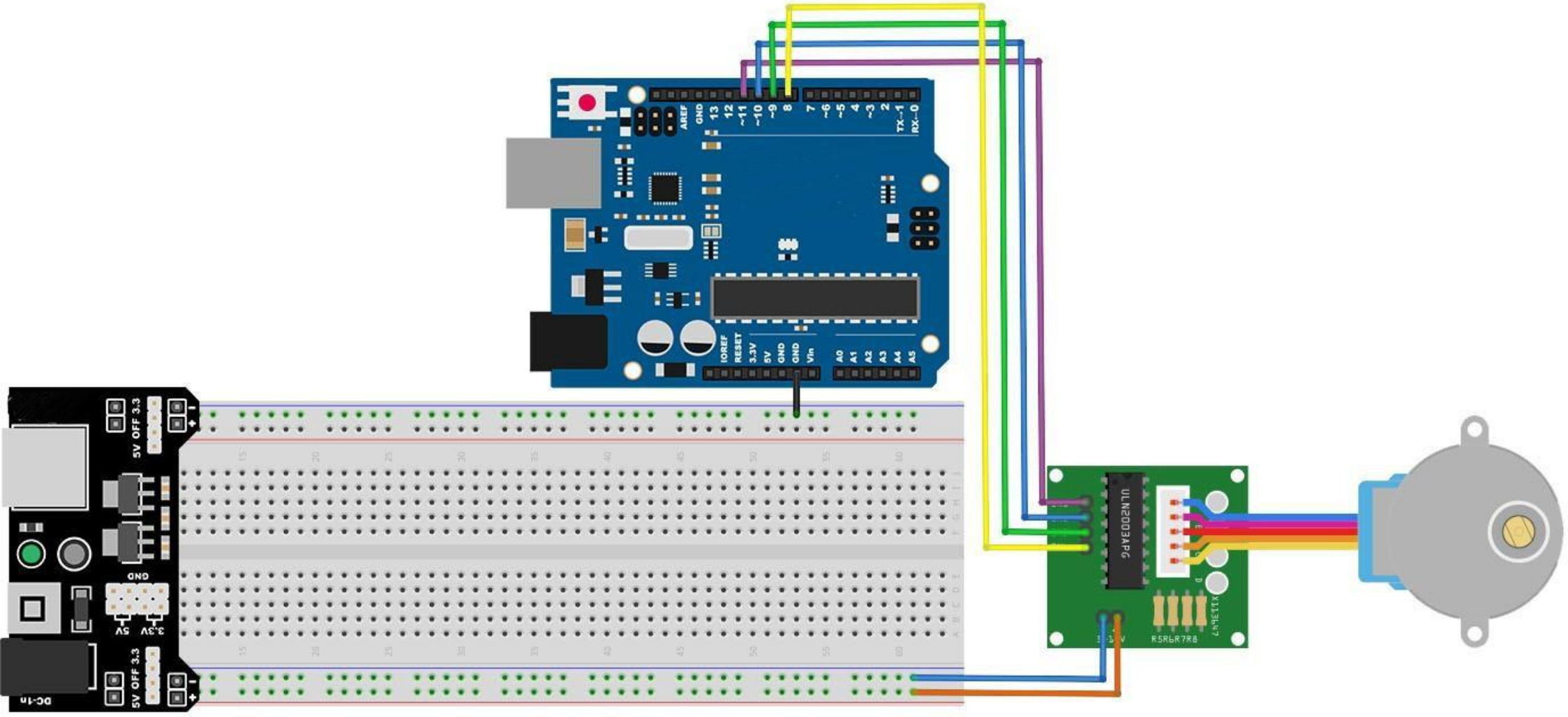


Connection

Schematic



Wiring diagram



We are using 4 pins to control the Stepper.

Pin 8-11 are controlling the Stepper motor.

We connect the Ground from to UNO to the Stepper motor.

Code

After wiring, please open the program in the code folder- Lesson 31 Stepper Motor and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

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

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

Lesson 24 Controlling Stepper Motor With Remote

Overview

In this lesson, you will learn a fun and easy way to control a stepper motor from a distance using an IR remote control.

The stepper we are using comes with its own driver board making it easy to connect to our UNO.

Since we don't want to drive the motor directly from the UNO, we will be using an inexpensive little breadboard power supply that plugs right into our breadboard and power it with a 9V 1Amp power supply.

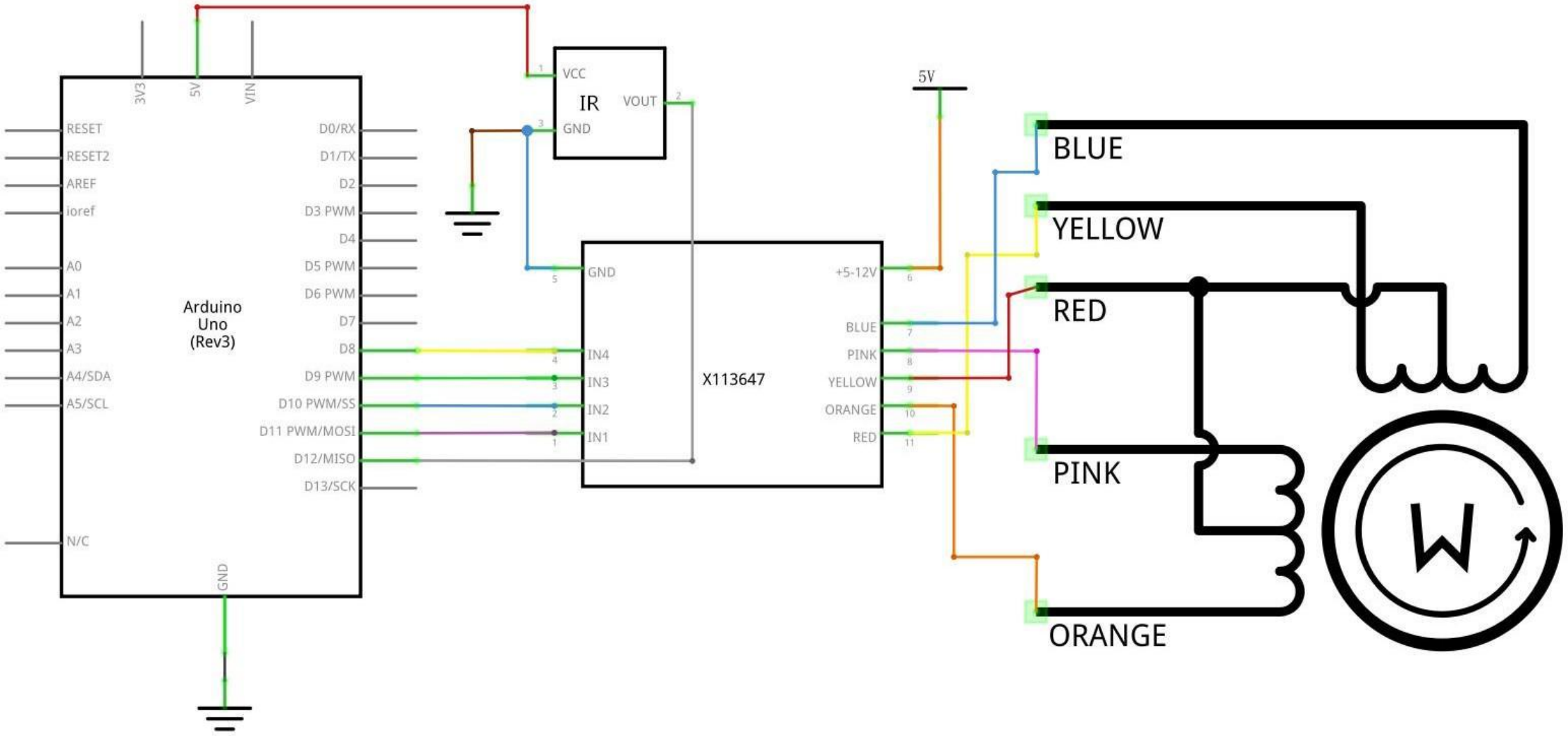
The IR sensor is connected to the UNO directly since it uses almost no power.

Component Required:

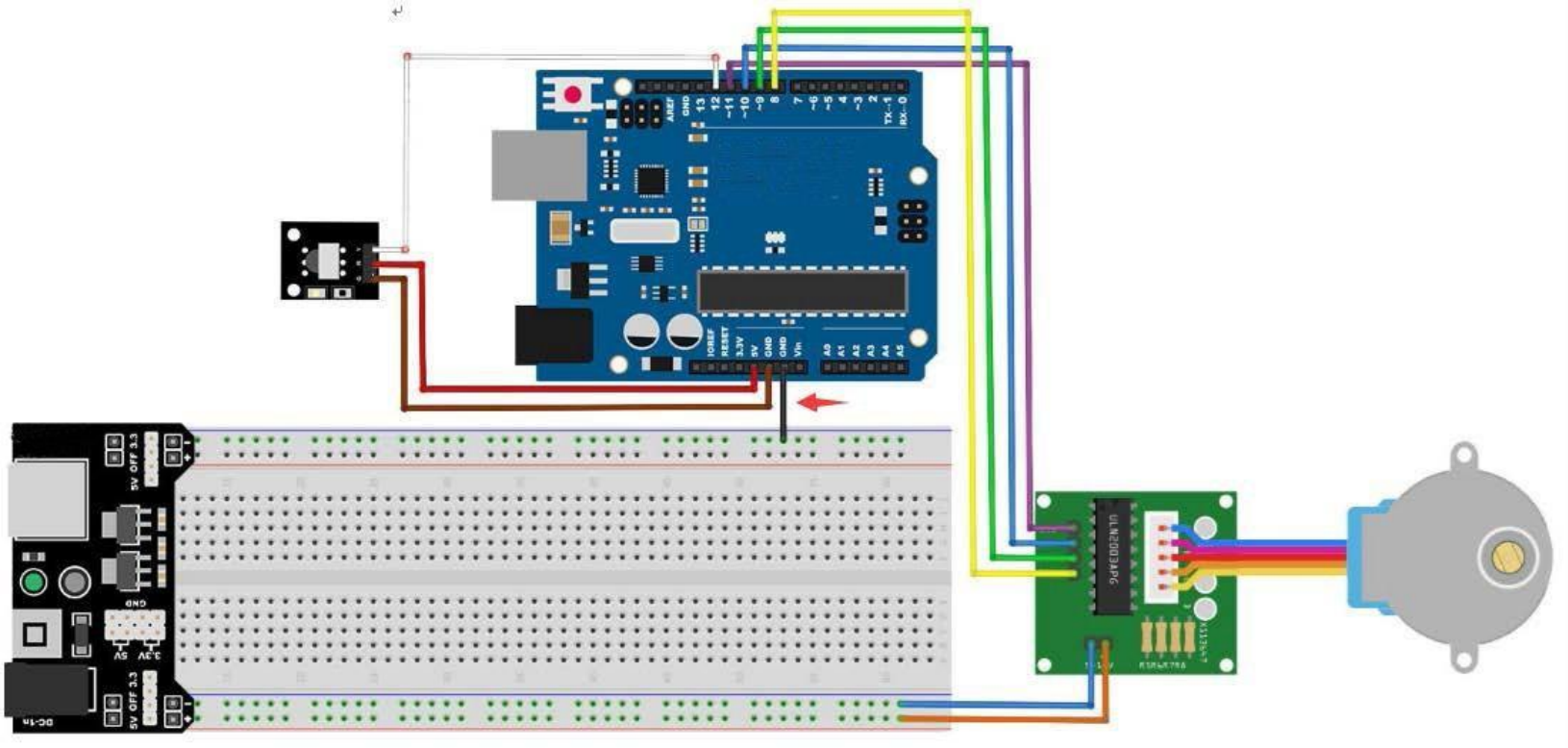
(1) x Uno R3
(1) x 830 tie-points breadboard
(1) x IR receiver module
(1) x IR remote
(1) x ULN2003 stepper motor driver module
(1) x Stepper motor
(1) x Power supply module
(1) x 9V1A Adapter
(9) x F-M wires (Female to Male DuPont wires)
(1) x M-M wire (Male to Male jumper wire)

Connection

Schematic



Wiring diagram

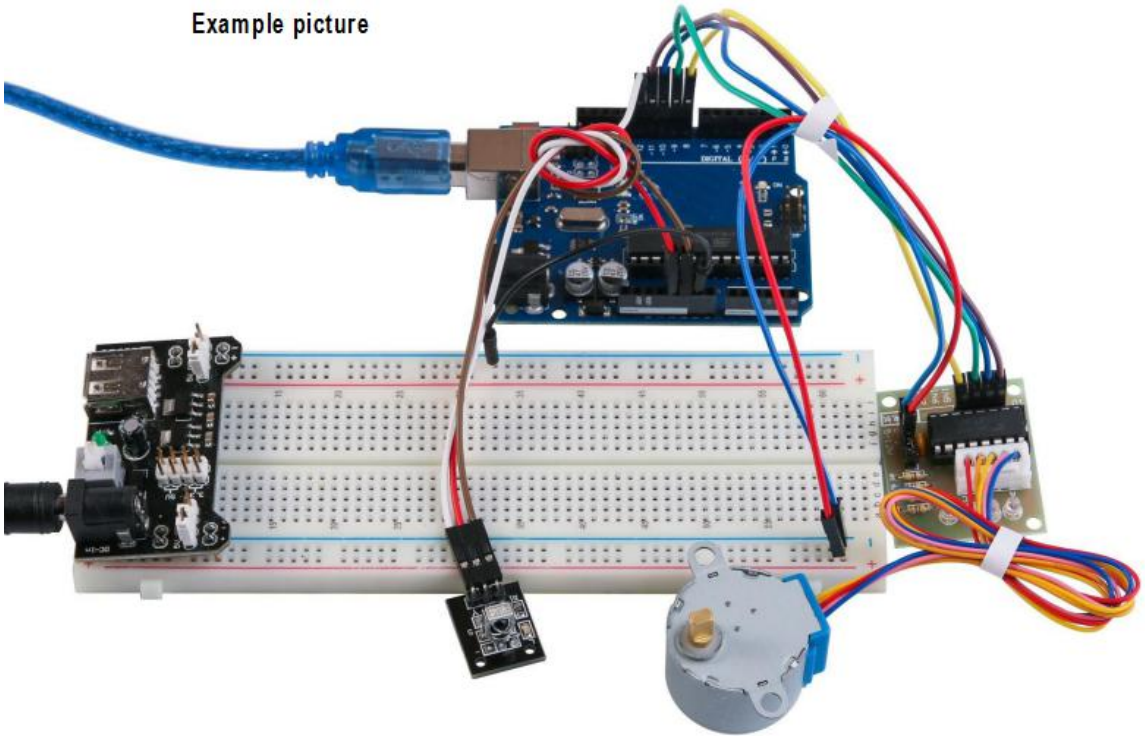


We are using 4 pins to control the Stepper and 1 pin for the IR sensor.
Pins 8-11 are controlling the Stepper motor and pin 12 is receiving the IR information.
We connect the 5V and Ground from the UNO to the sensor. As a precaution, use a breadboard power supply to power the stepper motor since it can use more power and we don't want to damage the power supply of the UNO.

Code

After wiring, please open program in the code folder- Lesson 32 Controlling Stepper Motor With Remote and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.
Before you can run this, make sure that you have installed the < IRremote > < Stepper >library or re-install it, if necessary. Otherwise, your code won'twork.
For details about loading the library file, see Lesson 1.
The code only recognize 2 values from the IR Remote control: VOL+ and VOL-.
When VOL+ is pressed on the remote the motor will make a full rotation clockwise.
VOL- will make a full rotation counter-clockwise.

Example picture



i