



37 SENSOR KIT TUTORIAL

FOR UNO V2.0

Preface

Our Company

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

Our product lines include DuPont wires, UNO R3 boards and complete starter kits designed for customers of any level to learn Arduino knowledge. In addition, we also sell Raspberry Pi accessories like TFT touch screens. Additionally, we plan to expand our offerings to include other technologies, including products related to 3D printing. All of our products comply with international quality standards and have been praised by our customers in a variety of different marketplaces throughout the world.

Official website:

<http://www.elegoo.com>

US Amazon storefront: <http://www.amazon.com/shops/A2WWHQ25ENKVJ1>

CA Amazon storefront: <http://www.amazon.ca/shops/A2WWHQ25ENKVJ1>

UK Amazon storefront: <http://www.amazon.co.uk/shops/AZF7WYXU5ZANW>

DE Amazon storefront: <http://www.amazon.de/shops/AZF7WYXU5ZANW>

FR Amazon storefront: <http://www.amazon.fr/shops/AZF7WYXU5ZANW>

ES Amazon storefront: <http://www.amazon.es/shops/AZF7WYXU5ZANW>

IT Amazon storefront: <http://www.amazon.it/shops/AZF7WYXU5ZANW>

Our Tutorial

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

Some code in this tutorial has been edited by Simon Monk. Simon Monk is the author of a number of books relating to Open Source Hardware. Some of his works include “Programming Arduino”, “30 Arduino Projects for the Evil Genius” and “Programming the Raspberry Pi” and can be found on Amazon

Customer Service

As a continuously and quickly growing technology company, we strive to offer you excellent products and quality service. You can reach out to us by email at service@elegoo.com or EUservice@elegoo.com. We look forward to hearing from you and any comments or suggestions are of great value to us.

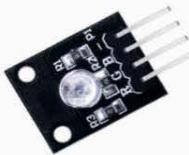
Any problems or questions that you have with our products will be promptly answered by our experienced engineers within 12 hours (24hrs during holiday)

Packing list

 www.elegoo.com



JOYSTICK



RGB LED



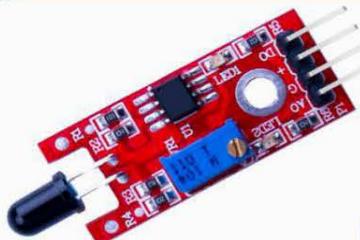
IR RECEIVER



AVOIDANCE



18B20 TEMP



FLAME



TRACKING

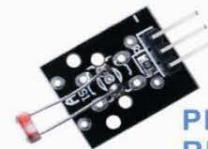
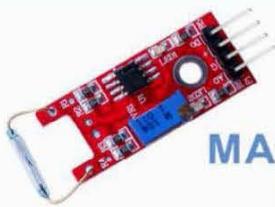


PHOTO RESISTOR

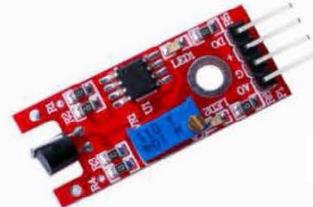


IR EMISSION

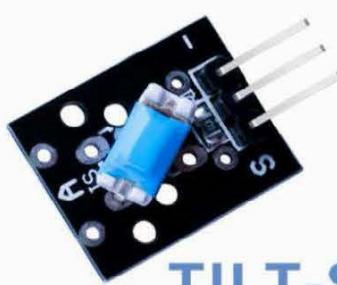
Contact us : service@elegoo.com



MAGNETIC
SPRING



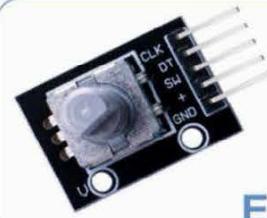
METAL
TOUCH



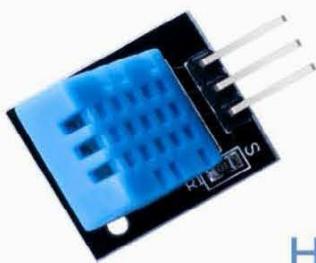
TILT-SWITCH



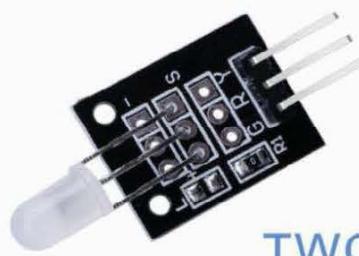
SMD
RGB



ROTARY
ENCODER

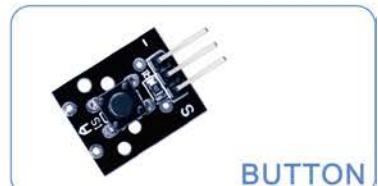
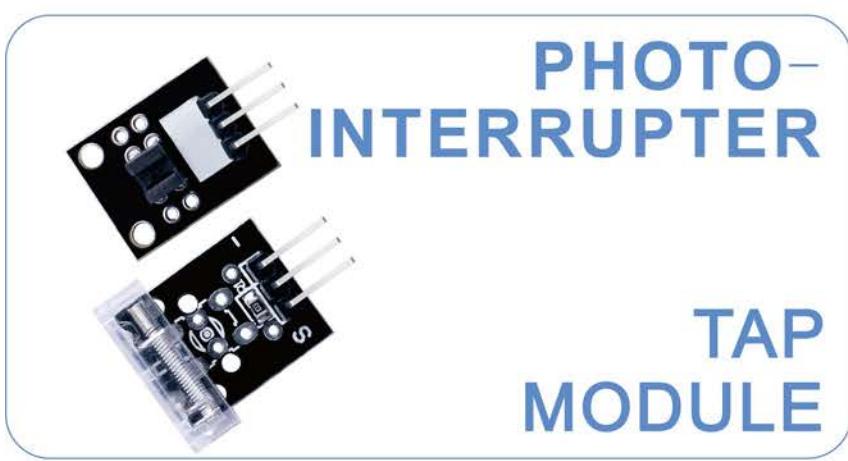
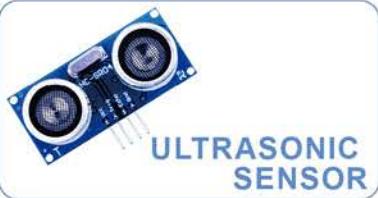


TEMP
AND
HUMIDITY

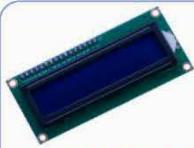


TWO-COLOR

Contact us : service@elegoo.com



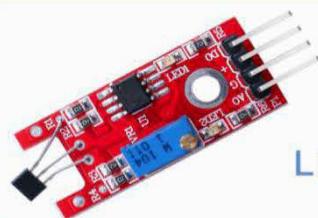
Contact us : service@elegoo.com



LCD 1602
MODULE
(WITH PIN HEADER)



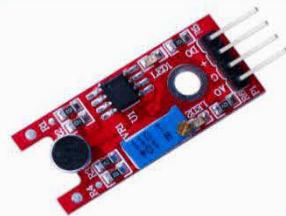
RELAY



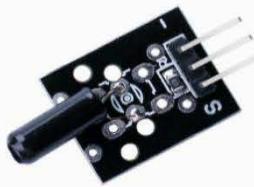
LINEAR
HALL



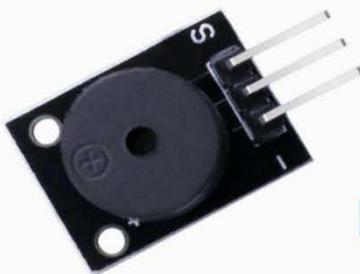
POWER
SUPPLY
MODULE



SMALL
SOUND



SHOCK



PASSIVE
BUZZER



HC-SR501 PIR
MOTION SENSOR

Content

Lesson 0 Installing IDE	10
Lesson 1 Add Libraries	24
Lesson 2 Open Serial Monitor	37
Lesson 3 Blink	43
Lesson 4 Temperature and Humidity Module.....	58
Lesson 5 DS18B20 Temperature Sensor Module	72
Lesson 6 Button Switch Module	85
Lesson 7 Switch Modules.....	92
Lesson 8 IR Receiver and Emission Module	102
Lesson 9 Active Buzzer Module	113
Lesson 10 Passive Buzzer Module	121
Lesson 11 Laser Module	128
Lesson 12 SMD RGB Module and RGB Module.....	134
Lesson 13 Photo-Interrupter Module	143
Lesson 14 Two Color LED Module.....	150
Lesson 15 Light Dependent Resistor Module.....	156
Lesson 16 Large Microphone Module and Small Microphone Module	163
Lesson 17 Reed Switch Module	173
Lesson 18 Digital Temperature Module.....	182
Lesson 19 Linear Magnetic Hall Sensor Module.....	195
Lesson 20 Flame Sensor Module	204
Lesson 21 Touch Sensor Module	213
Lesson 22 Seven-Color Flash LED Module.....	220
Lesson 23 Joystick Module	225
Lesson 24 Line Tracking Module.....	232
Lesson 25 Obstacle Avoidance Module	237
Lesson 26 Rotary Encode Module	246

Lesson 27 Relay Module	255
Lesson 28 LCD Display.....	261
Lesson 29 Ultrasonic Sensor Module	269
Lesson 30 GY-521 Module	277
Lesson 31 HC-SR501 PIR Sensor	289
Lesson 32 Water Level Detection Sensor Module	304
Lesson 33 Real Time Clock Module	313
Lesson 34 Keypad Module	322

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 IDE 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 there is a certain amount of manual work to install the software.

STEP 1: Go to <https://www.arduino.cc/en/Main/Software> and Download the Arduino IDE.

ARDUINO 1.8.5

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

This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions.

Windows Installer
Windows ZIP file for non admin install

Windows app

Mac OS X 10.7 Lion or newer

Linux 32 bits
Linux 64 bits
Linux ARM

[Release Notes](#)
[Source Code](#)
[Checksums \(sha512\)](#)

The version of the Arduino IDE available at this website is usually the latest version, so you may find the that version number is different from the example screenshot above.

STEP2: Download the Arduino IDE that is compatible with the operating system of your computer. Take Windows as an example here.



Click Windows Installer.

Support the Arduino Software

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

A screenshot of the Arduino contribution page. It features a cartoon robot icon on the left. To its right, text reads: "SINCE MARCH 2015, THE ARDUINO IDE HAS BEEN DOWNLOADED 8,808,272 TIMES. (IMPRESSIVE!) NO LONGER JUST FOR ARDUINO AND GENUINO BOARDS, HUNDREDS OF COMPANIES AROUND THE WORLD ARE USING THE IDE TO PROGRAM THEIR DEVICES, INCLUDING COMPATIBLES, CLONES, AND EVEN COUNTERFEITS. HELP ACCELERATE ITS DEVELOPMENT WITH A SMALL CONTRIBUTION! REMEMBER: OPEN SOURCE IS LOVE!" Below this are six circular buttons with contribution amounts: "\$3", "\$5", "\$10", "\$25", "\$50", and "OTHER". At the bottom, there are two buttons: "JUST DOWNLOAD" and "CONTRIBUTE & DOWNLOAD".

Click JUST DOWNLOAD.

By the way, you also could download the Arduino IDE from our official web-

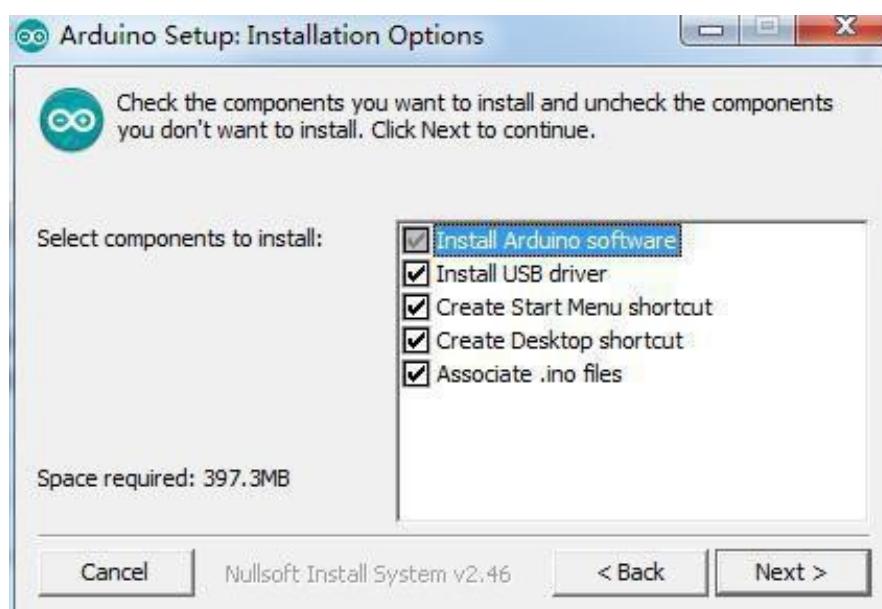
<http://www.elegoo.com/download/>

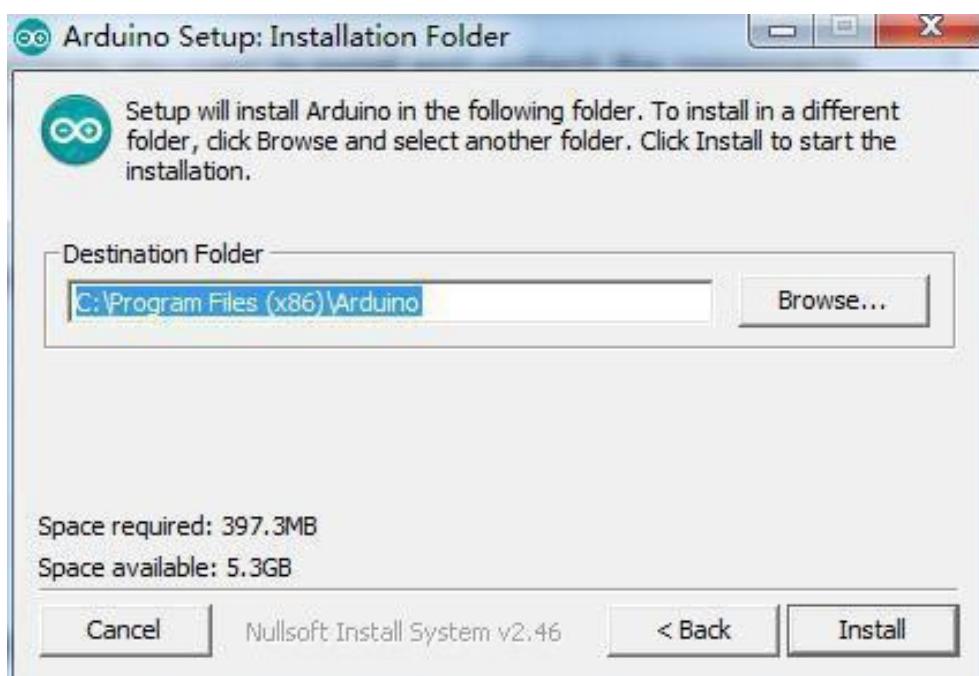
Installing Arduino IDE(Windows)

Install Arduino with the exe Installation package.

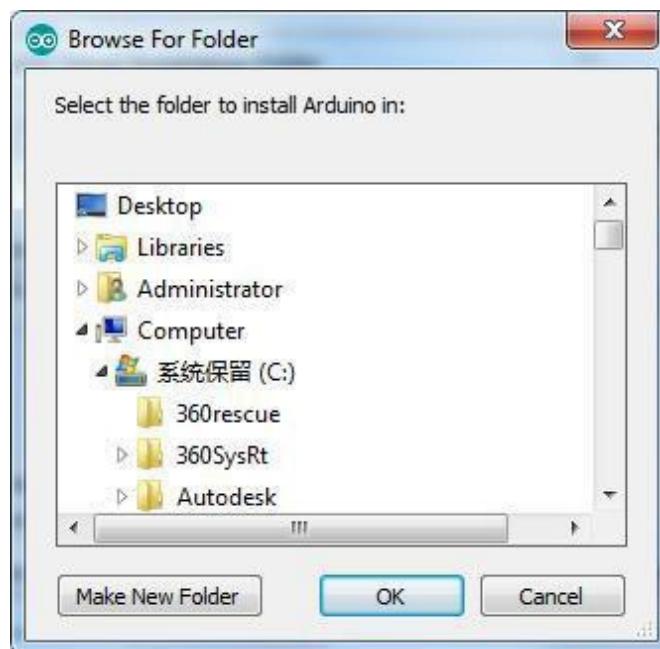


Click *I Agree* to see the following interface

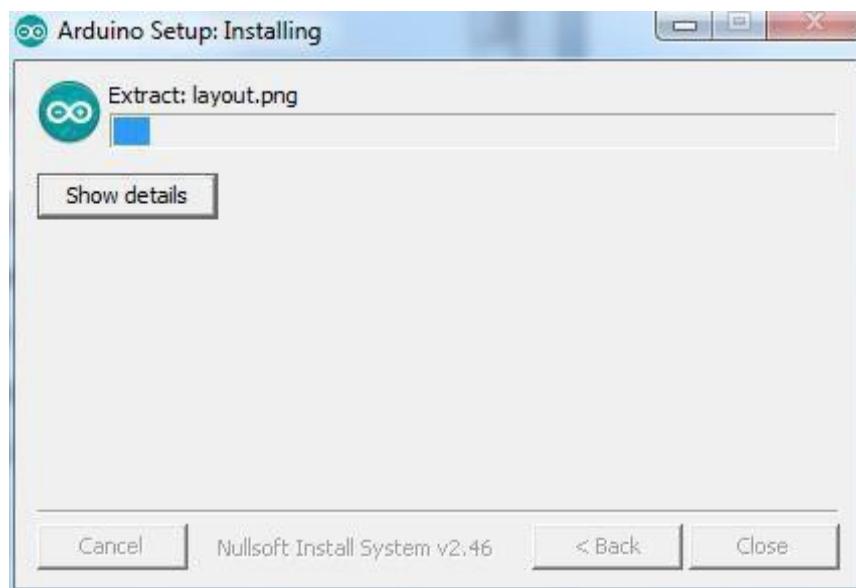


Click Next

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



Click *Install* to initiate installation



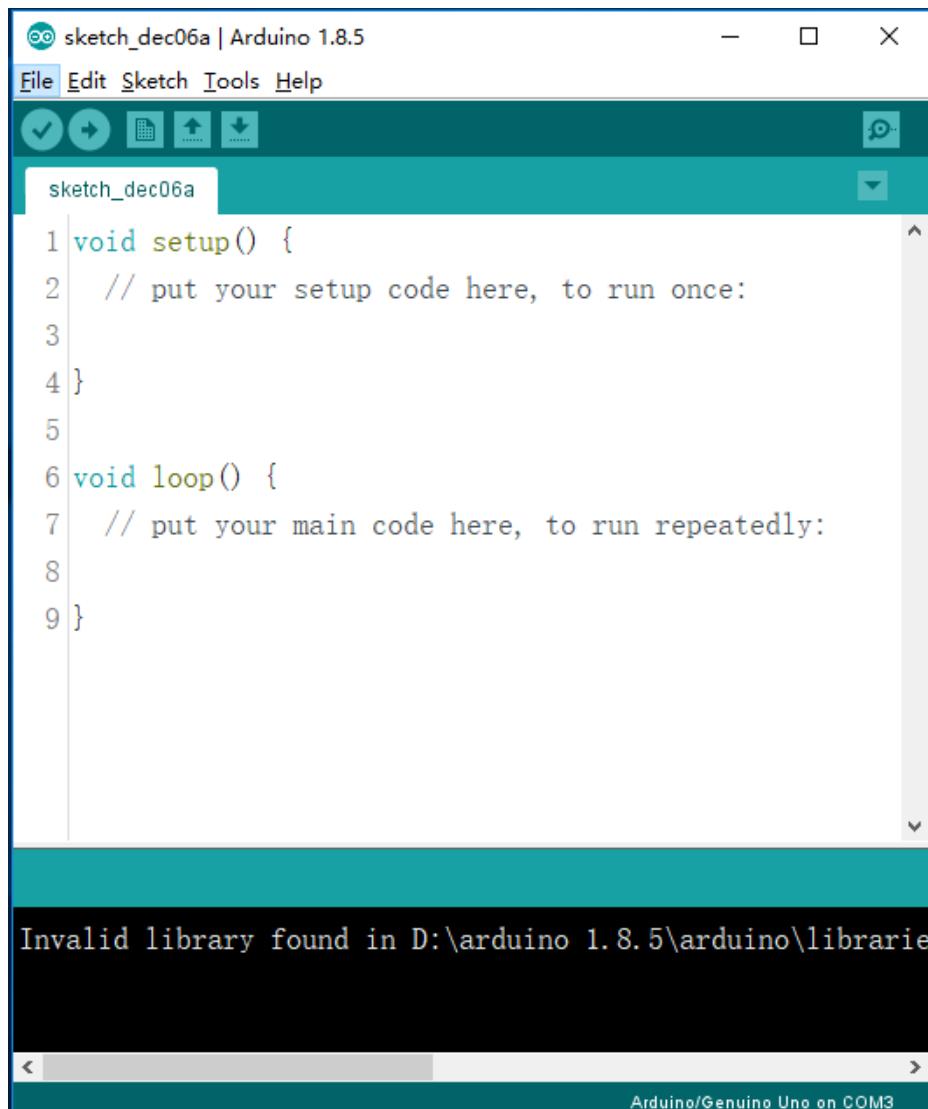
Finally, the following interface appears, click *Install* to finish the installation.



Next, the following icon appears on the desktop



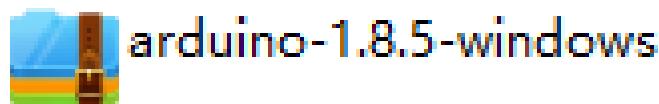
Double-click to enter the Arduino IDE

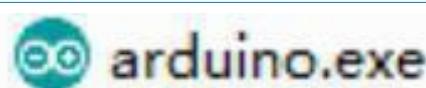
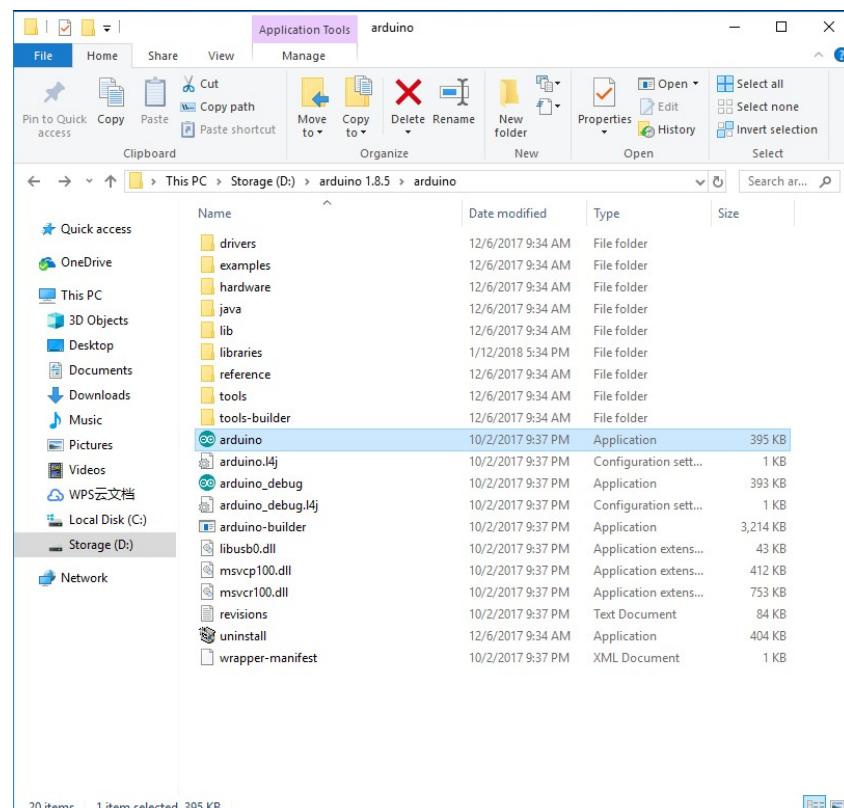


We have already explained to you how to install the Arduino IDE with the “exe” installation package.

We are now introducing other ways to install the software, you can simply skip the contents below and jump to the next section (lesson 1) if you want.

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





File Edit Sketch Tools Help

sketch_dec06a | Arduino 1.8.5

```

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/Genuino Uno on COM3

However, this installation method requires installation of a separate driver.

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

1 Use your USB cable to connect Arduino and the USB port on your computer.

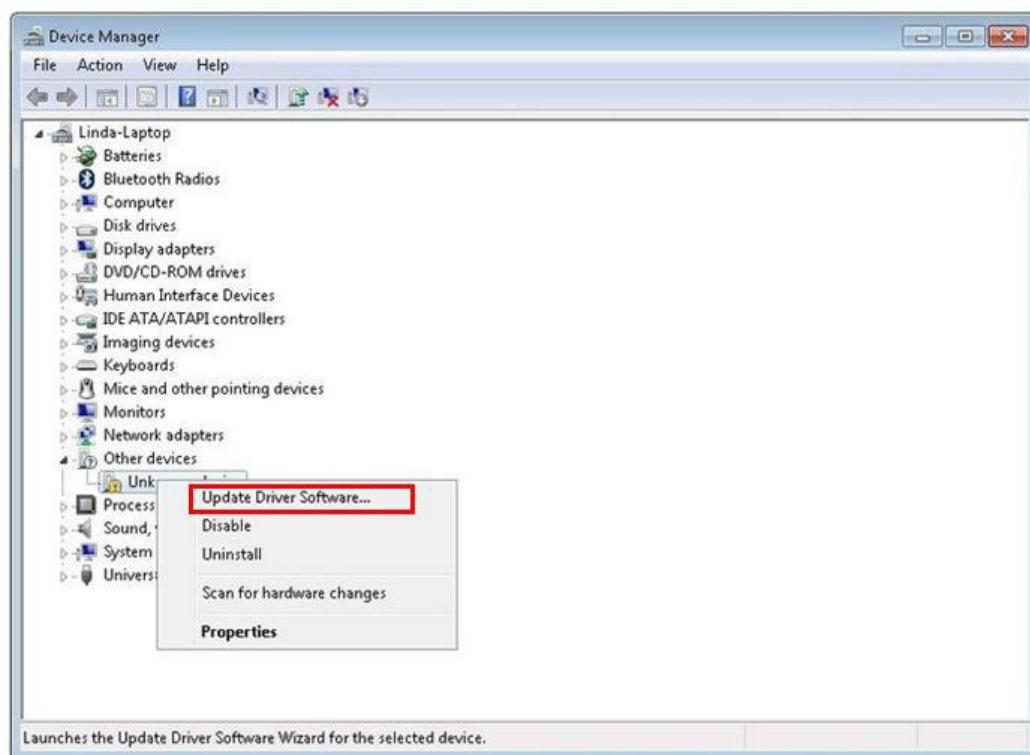
2 Check LED light indication (one flashing and one stable)

3 Ignore "Found New Hardware" message

4 cancel any attempts that Windows try and install drivers automatically.

The most reliable method of installing the USB drivers is to use the Device Manager. You can access the Device Manager in different ways depending on your version of Windows. In Windows 7, you have to open the Control Panel, first 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 device.





Right-click on the device and select the option on the top of the menu. (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\arduino1.8.5\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.

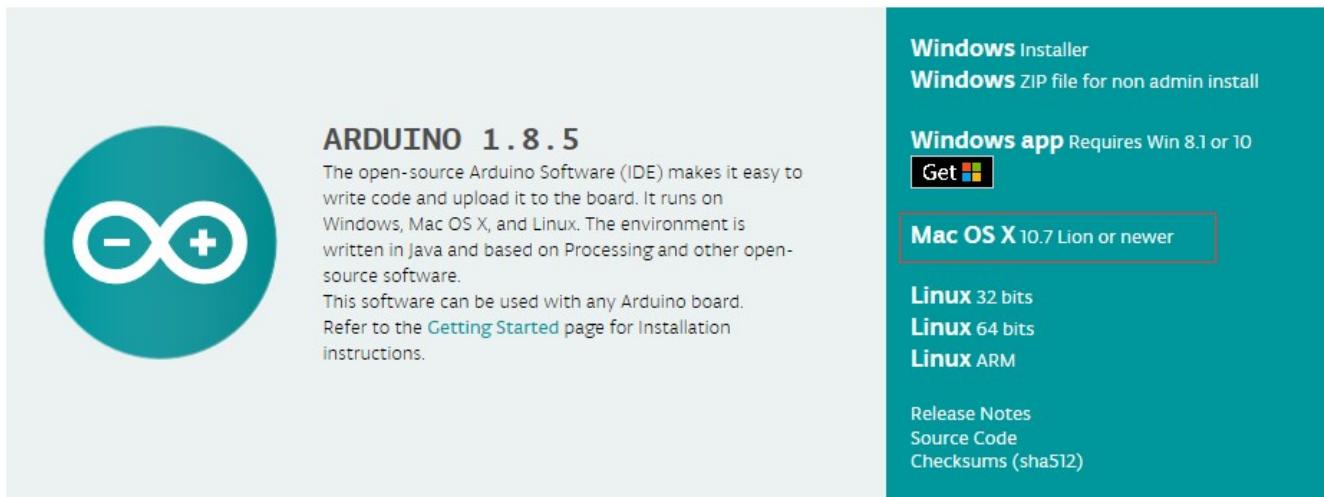


Windows users may skip the installation directions for Mac and Linux systems and jump to Lesson 1. Mac and Linux users may continue to read this section.

Install Arduino (Mac OS X)

Step One: Download Arduino Software (IDE)

- Open the URL: <https://www.arduino.cc/en/Main/Software> with browser



- Click Mac OS X 10.7 Lion or newer

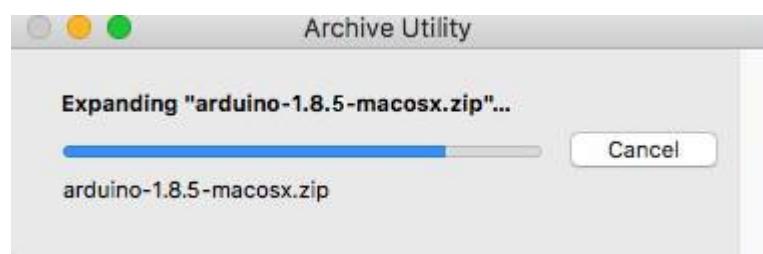
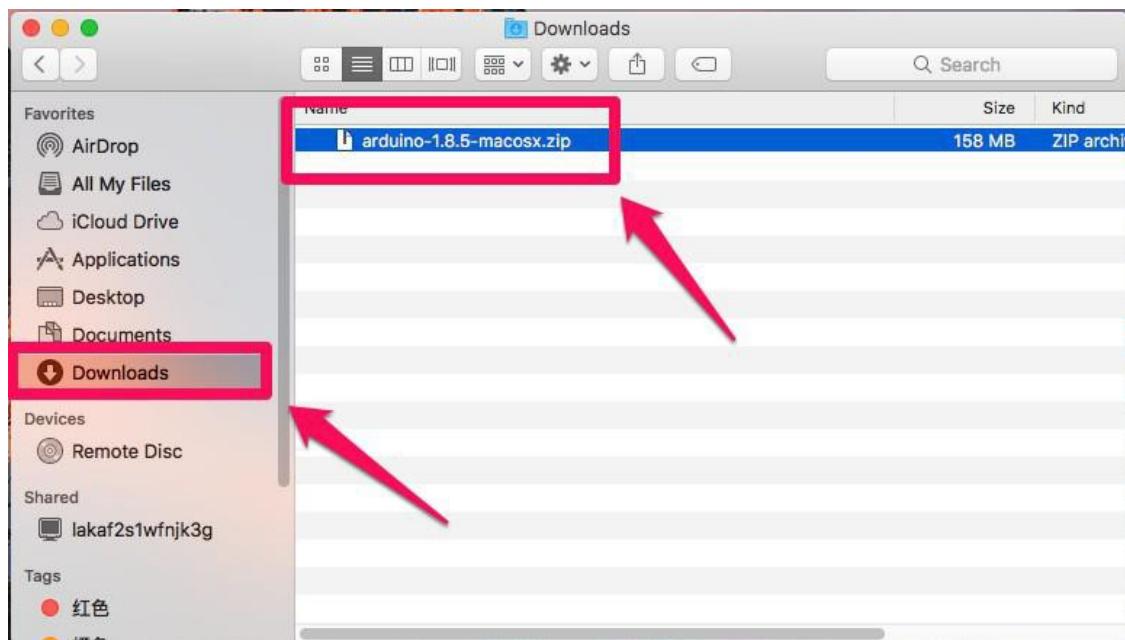
Contribute to the Arduino Software

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

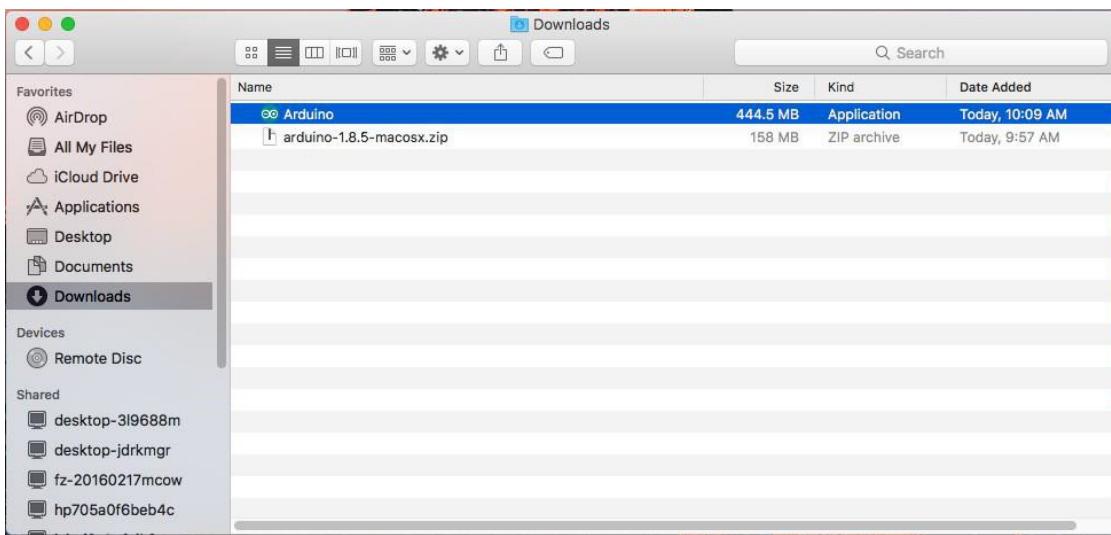


- Click JUST DOWNLOAD

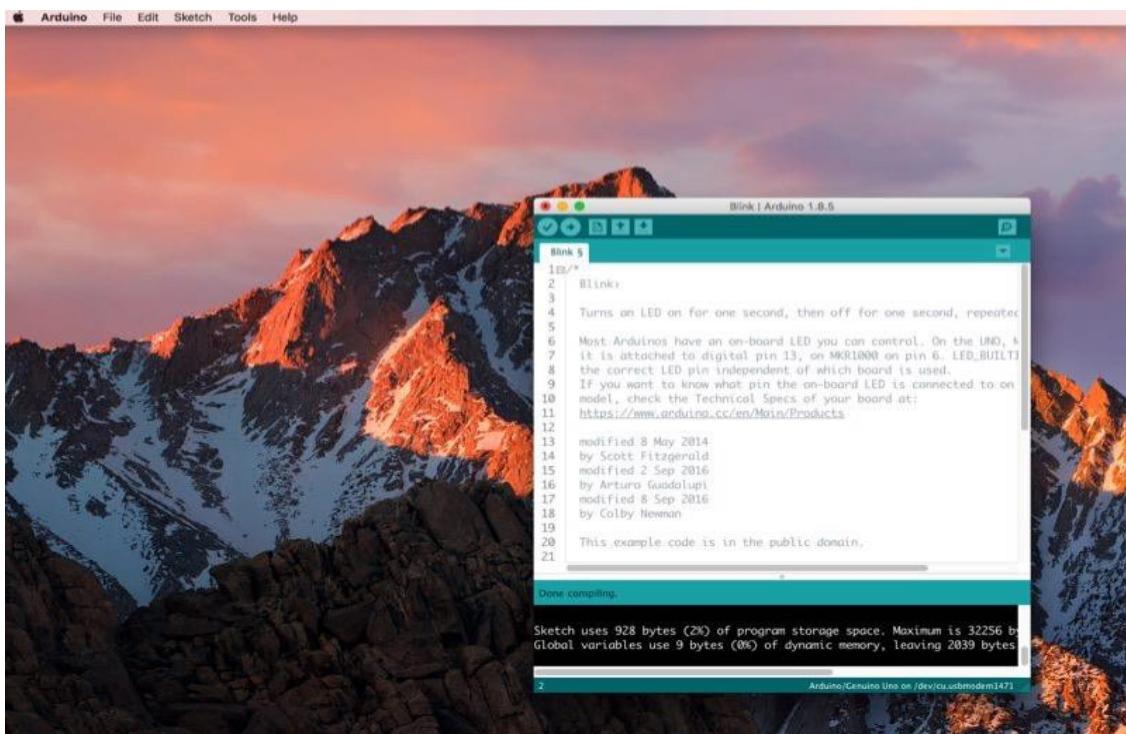
Download and Unzip the zip file, double click the Arduino.app to enter Arduino IDE; the system will ask you to install Java runtime library if you don't have it in your computer. Once the installation is complete you can run the Arduino IDE.



·Arduino file is the Arduino application, just double click to open it



·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 Ubuntu Software Center.

You can refer to the following links which offer you the specific installation steps.

<https://www.arduino.cc/en/guide/linux>



INSTALL ARDUINO ON LINUX.pdf



arduino-1.8.5-linux32.tar.xz



arduino-1.8.5-linux64.tar.xz

Lesson 1 Add Libraries

Install other 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.

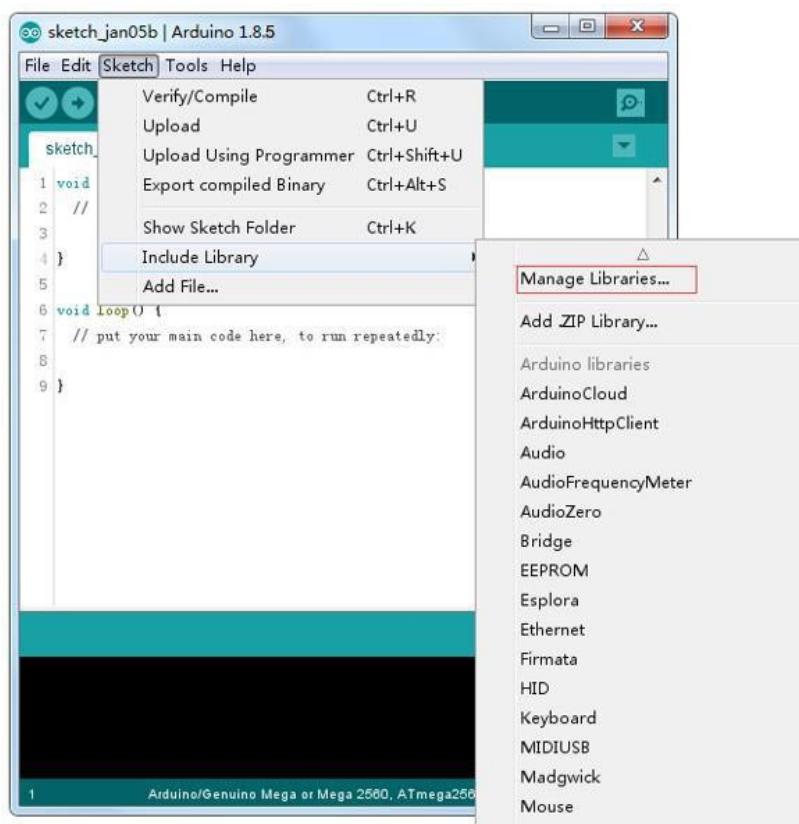
What are Libraries?

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 Liquid Crystal 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

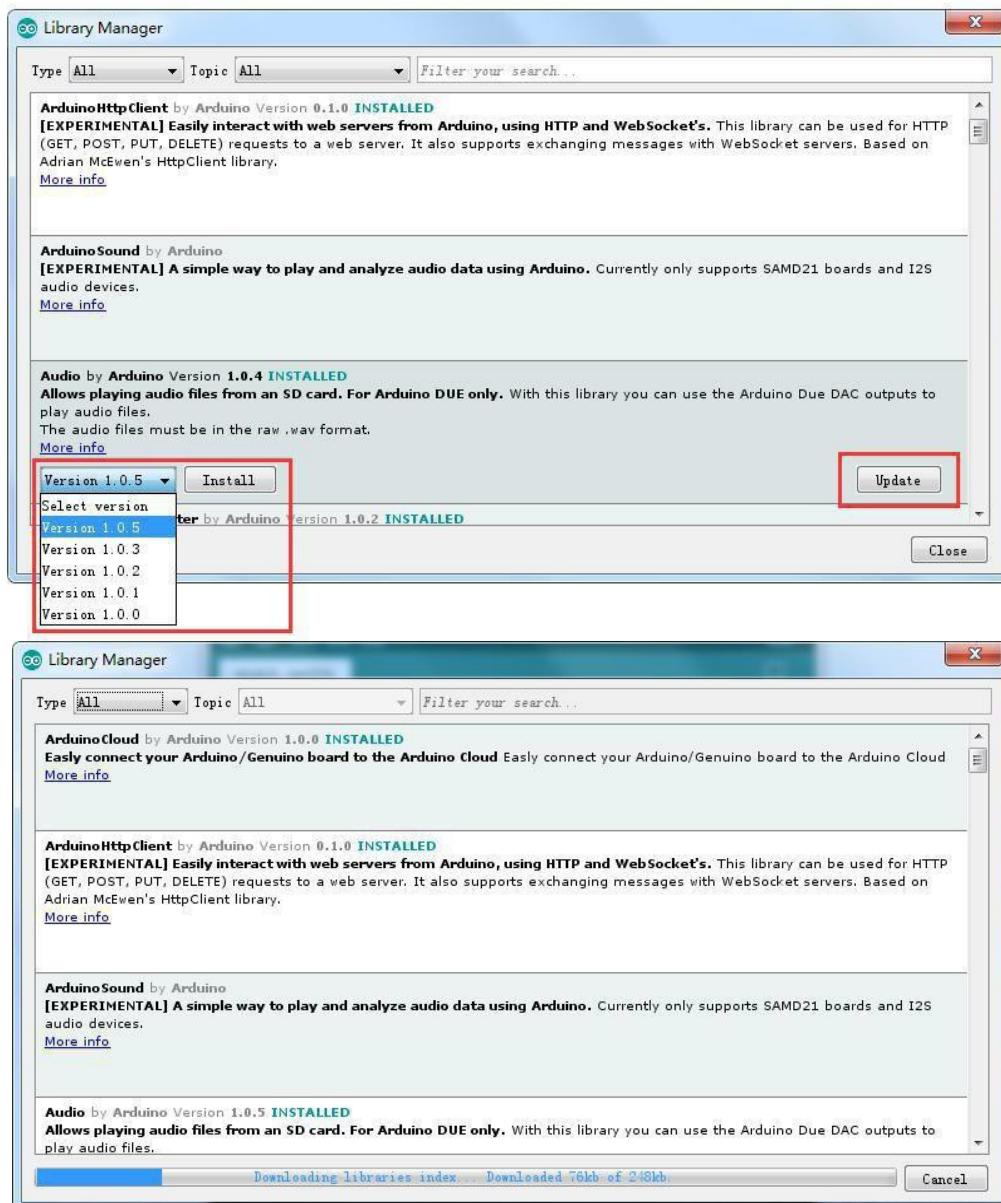
Using the Library Manager

To install a new library into your Arduino IDE you can use the Library Manager (available from IDE version 1.8.5). 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 and it depends on your connection speed. Once it has finished, an Installed tag should appear next to the Bridge library. You can close the library manager.

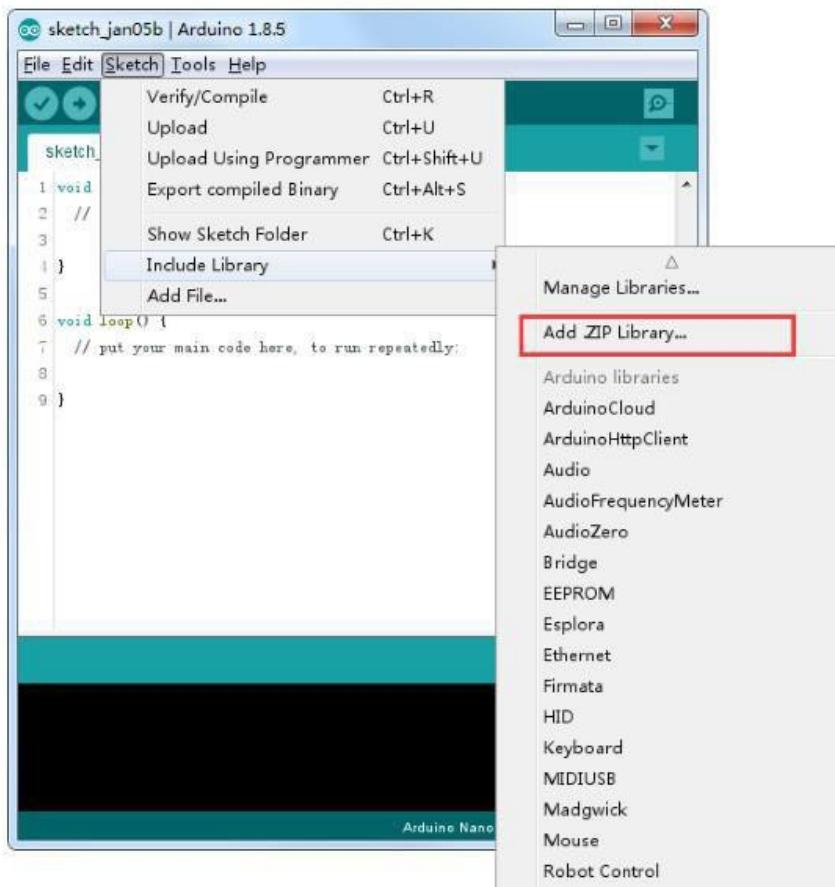


You can now find the new library available in the Include Library menu. If you want to add your own library open a new issue on [Github](#).

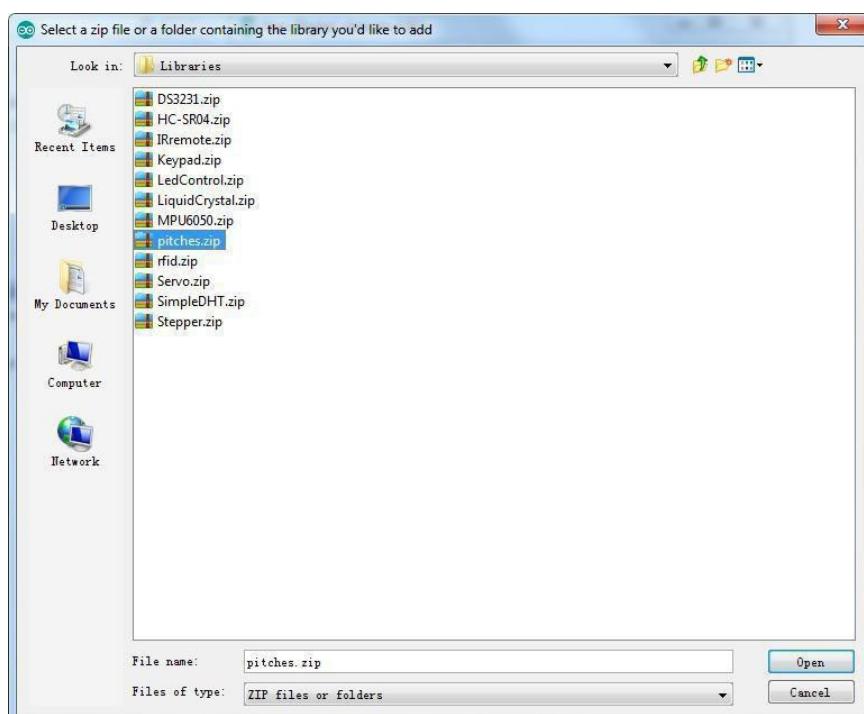
Importing a .zip Library

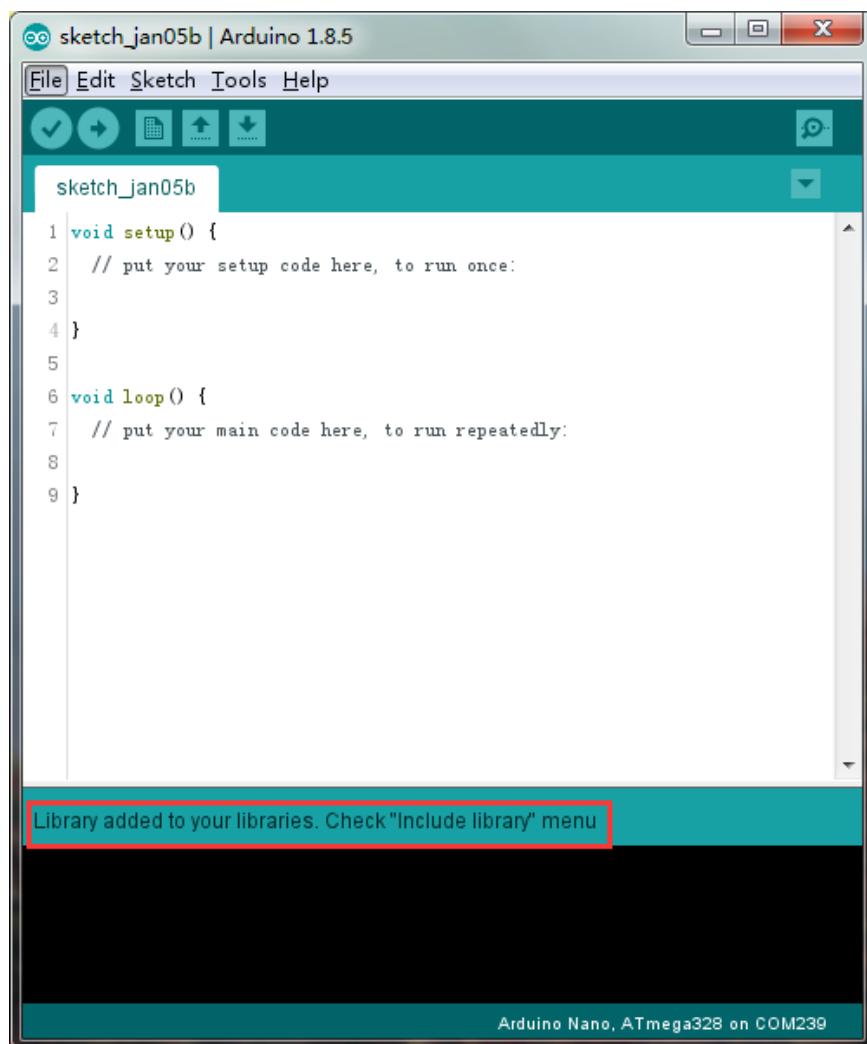
Libraries are often distributed as a ZIP file or folder. The name of the folder is the name of the library. Inside the folder will be a .cpp file, a .h file and often a keywords.txt file, examples folder, and other files required by the library. Starting with version 1.0.5, you can install 3rd party 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 Libraries you would like to add. Navigate to the .zip file's location and open it.





Return to the Sketch > Import Library menu. You should now see the library at the bottom of the dropdown 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, and then unzip the ZIP file containing the library. For example, if you're installing a library called "Arduino Party", uncompress ArduinoParty.zip. It should contain a folder called ArduinoParty, with files like ArduinoParty.cpp and ArduinoParty.h inside. (If the .cpp and .h files aren't in a folder, you'll need to create one. In this case, you'd make a folder called "ArduinoParty" and move into it all the files that were in the ZIP file, like ArduinoParty.cpp and ArduinoParty.h.)

Drag the Arduino Party 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):

MyDocuments\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.)

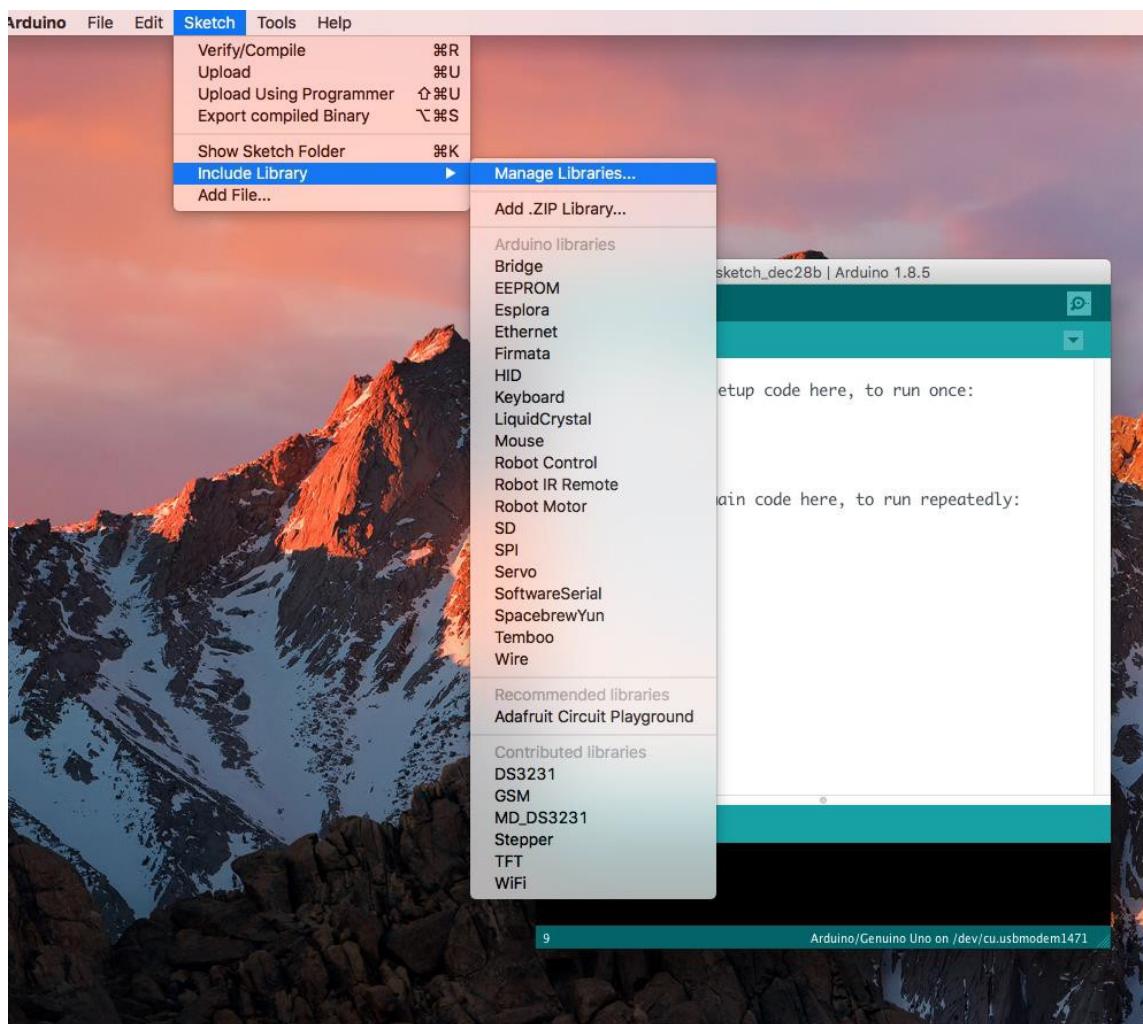
Restart the Arduino application. Make sure the new library appears in the Sketch-

>Import Library menu item of the software. That's it! You've installed a library!

How to install library files in Mac environment

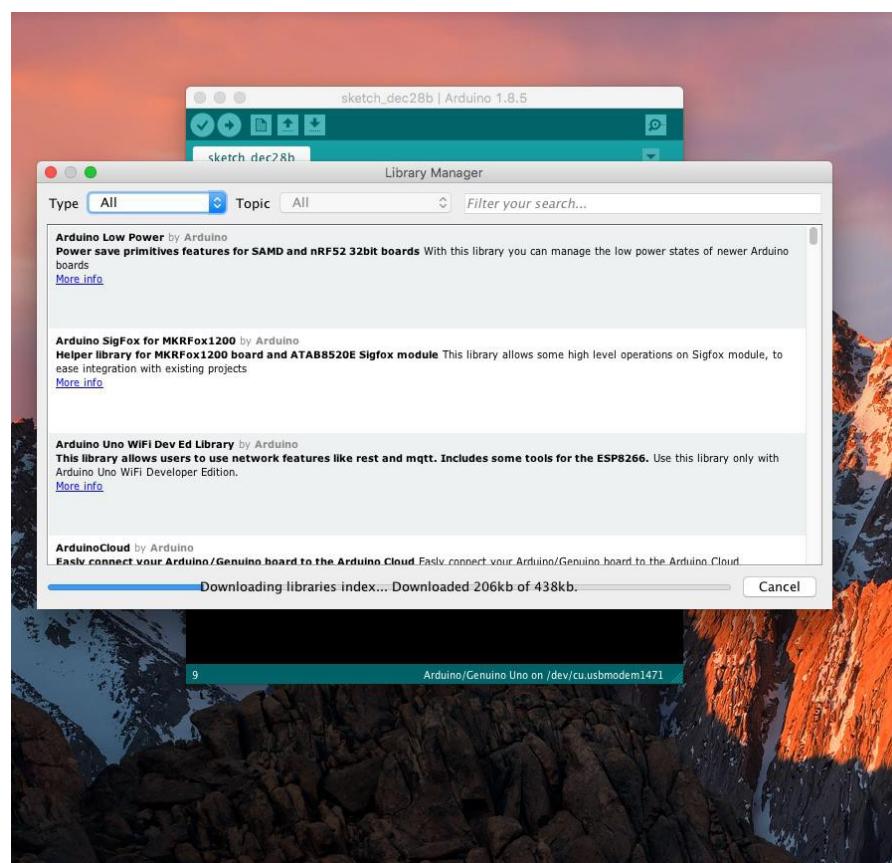
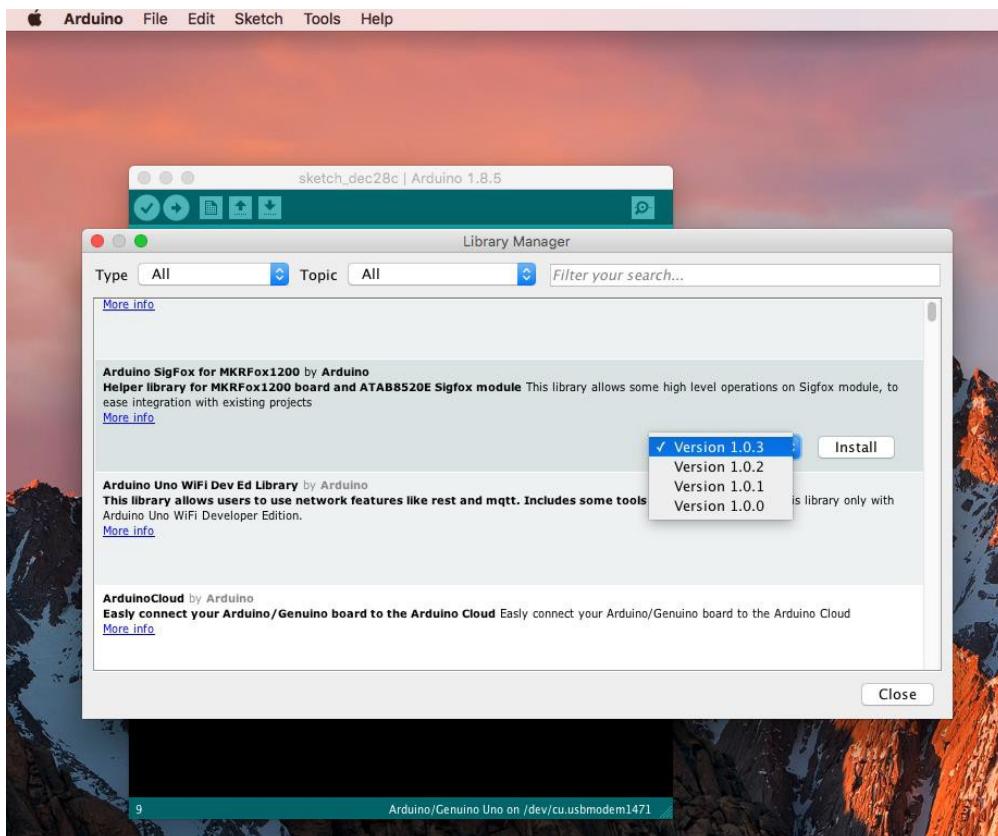
Using the Library Manager

To install a new library into your Arduino IDE you can use the Library Manager (available from IDE version 1.8.5). Open the IDE and click to the "Sketch" menu and then Include Library > Manage Libraries.

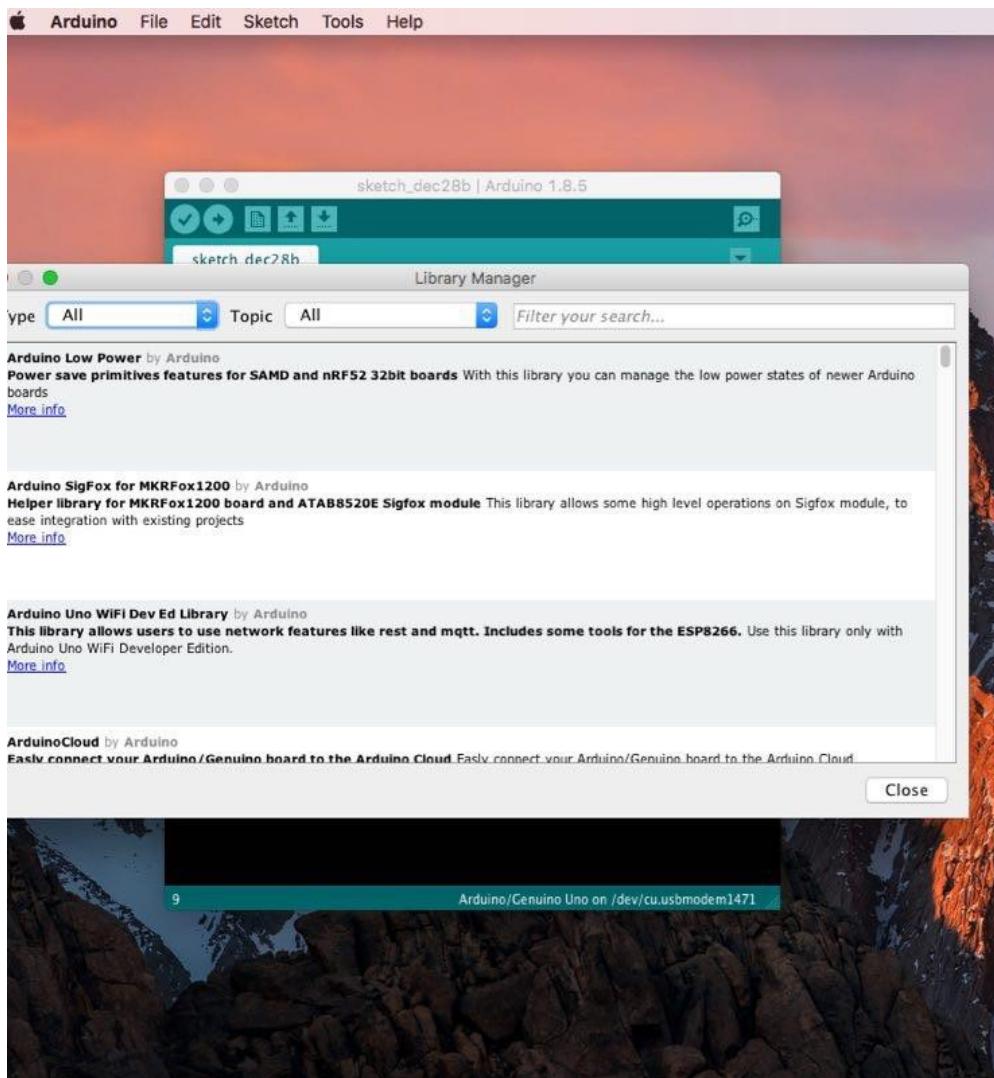


Then the library manager will open and you will find a list of libraries that are already installed or ready for installation. In this example we will install the Bridge library. Scroll the list to find it, and then select the version of the library you want to install. Sometimes only one version of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

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



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

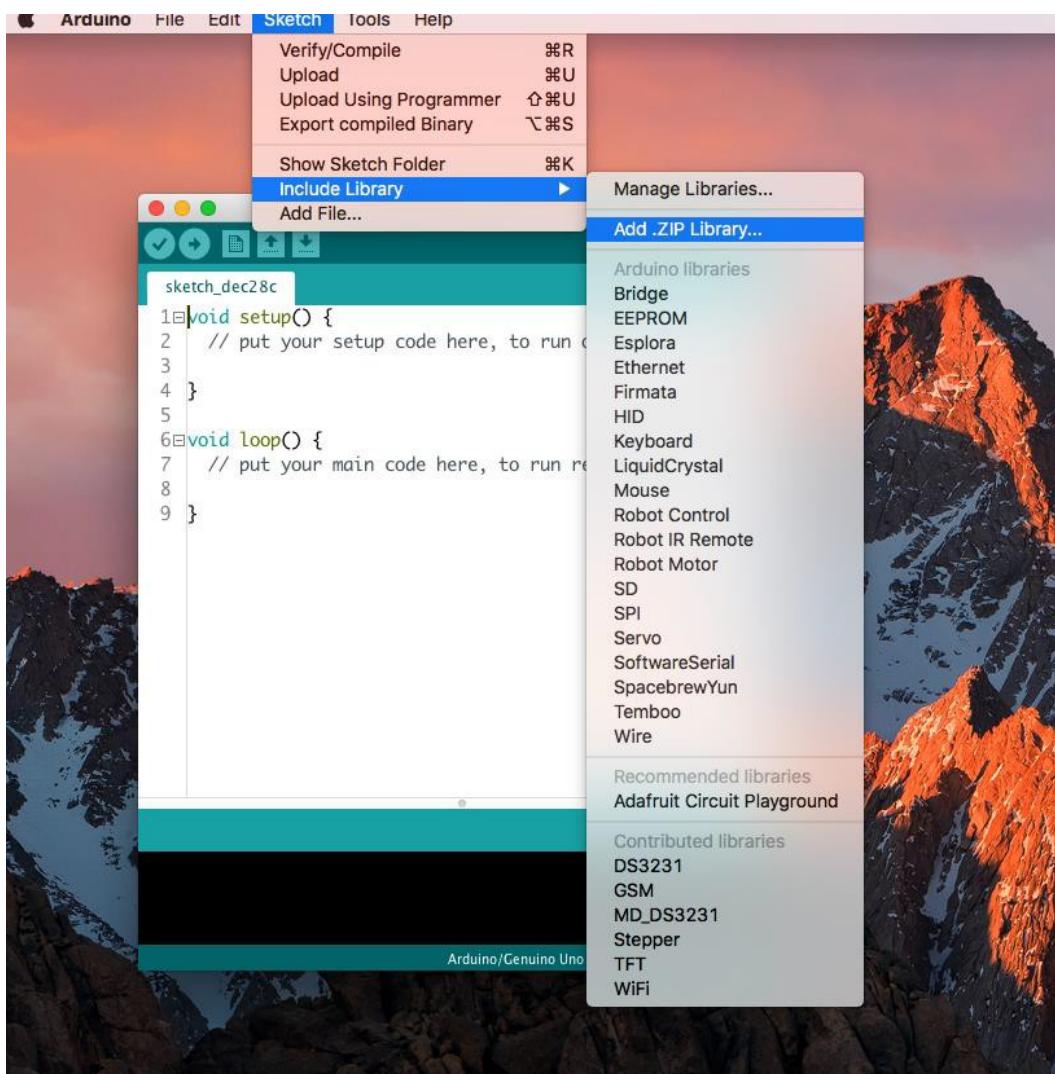


You can now find the new library available in the Include Library menu. If you want to add your own library open a new issue on Github.

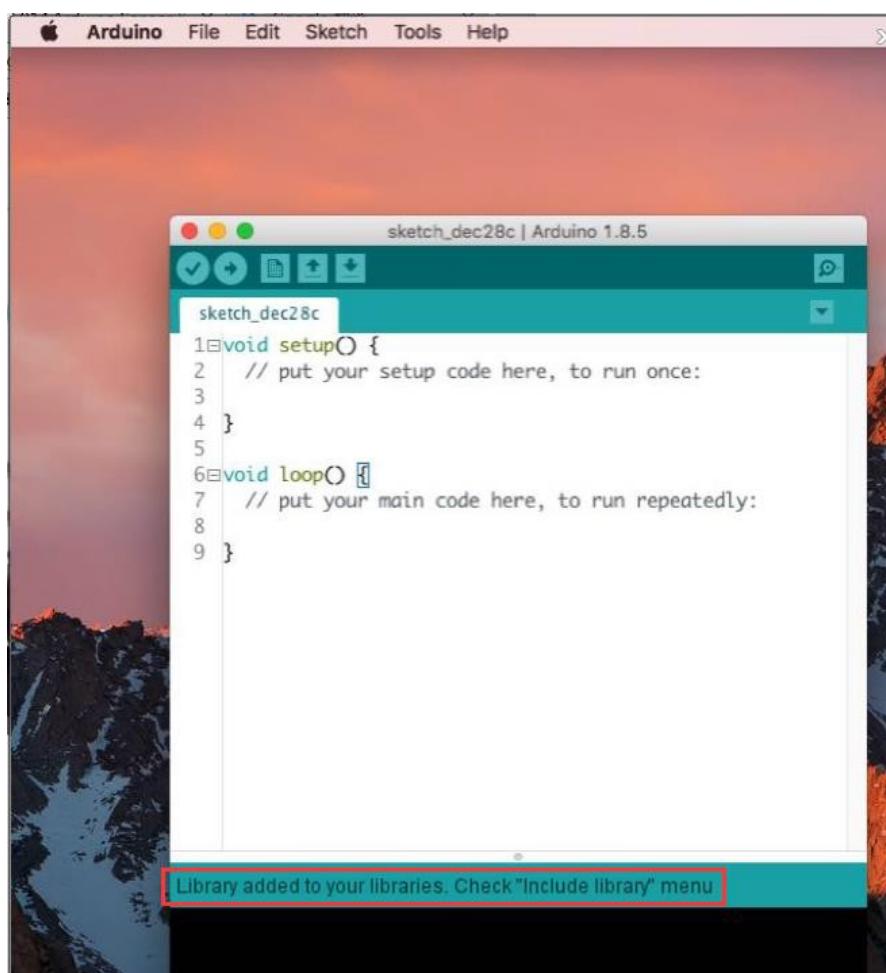
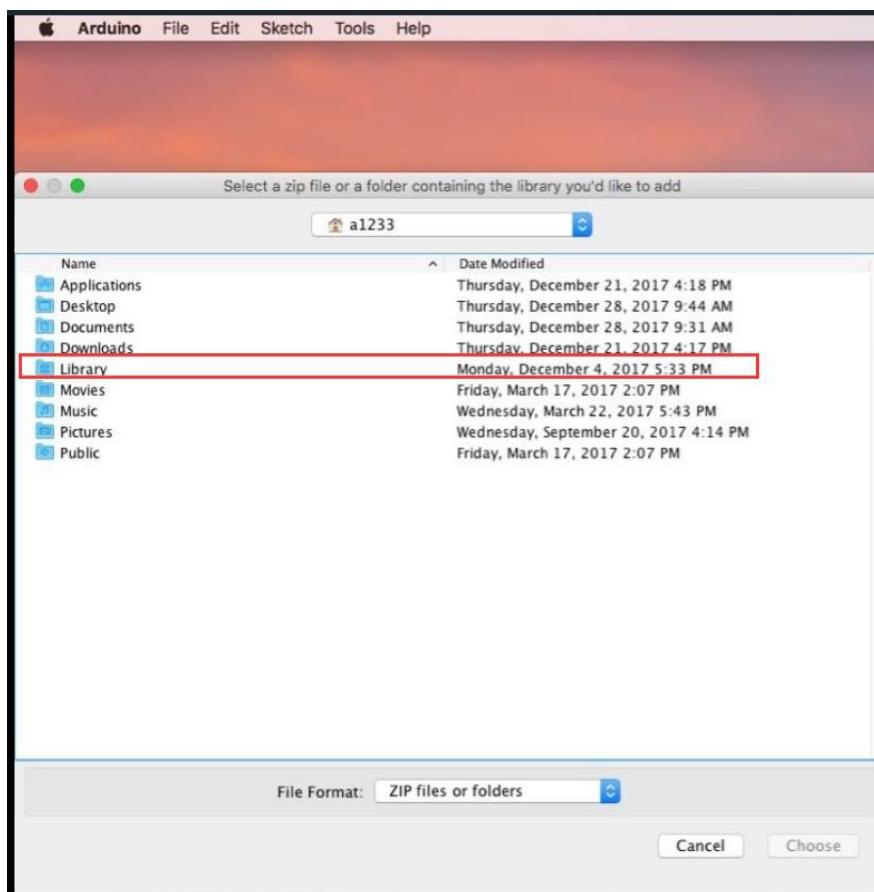
Importing a .zip Library

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

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



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



Return to the Sketch > Import Library menu. You should now see the library at the bottom of the drop-down menu. It is ready to be used in your sketch. The zip file will have been expanded in the libraries folder in your Arduino sketches directory.

NB: the Library will be available to use in sketches, but examples for the library will not be exposed in the File > Examples until after the IDE has restarted.

Those two are the most common approaches. 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, and then unzip the ZIP file containing the library. For example, if you're installing a library called "ArduinoParty", uncompress ArduinoParty.zip. It should contain a folder called Arduino Party, with files like Arduino Party.cpp and Arduino Party.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 Arduino Party 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\Arduino Party.cpp and Documents\Arduino\libraries\ArduinoParty\ArduinoParty.cpp won't work.)

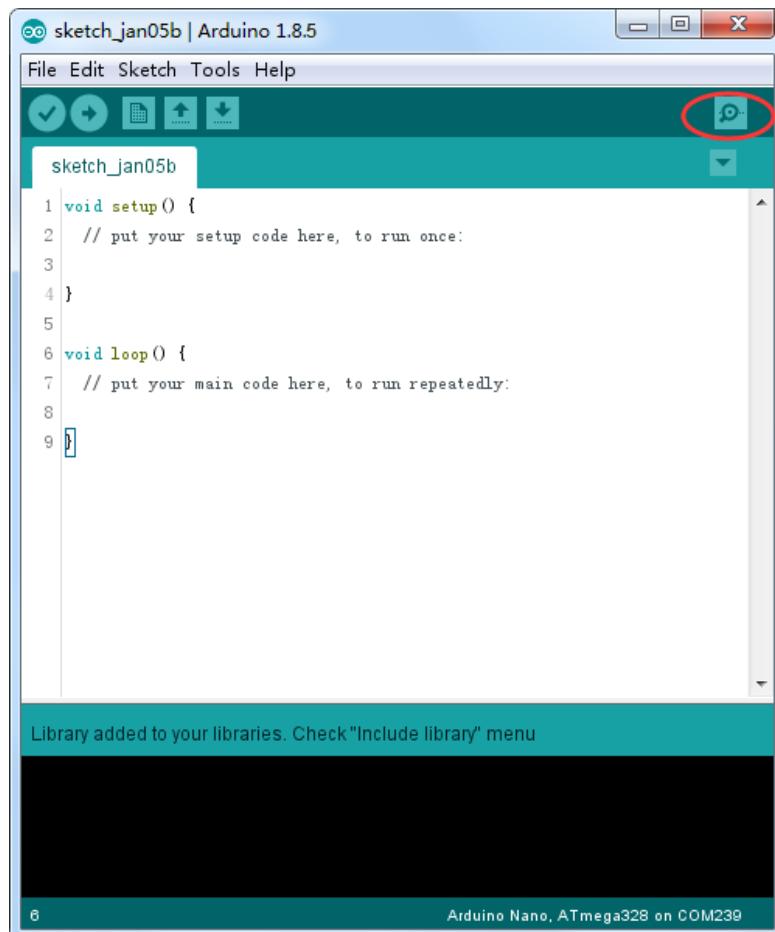
Restart the Arduino application. Make sure the new library appears in the Sketch->Import Library menu item of the software. That's it! You've installed a library!

Lesson 2 Open Arduino Serial Monitor

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.

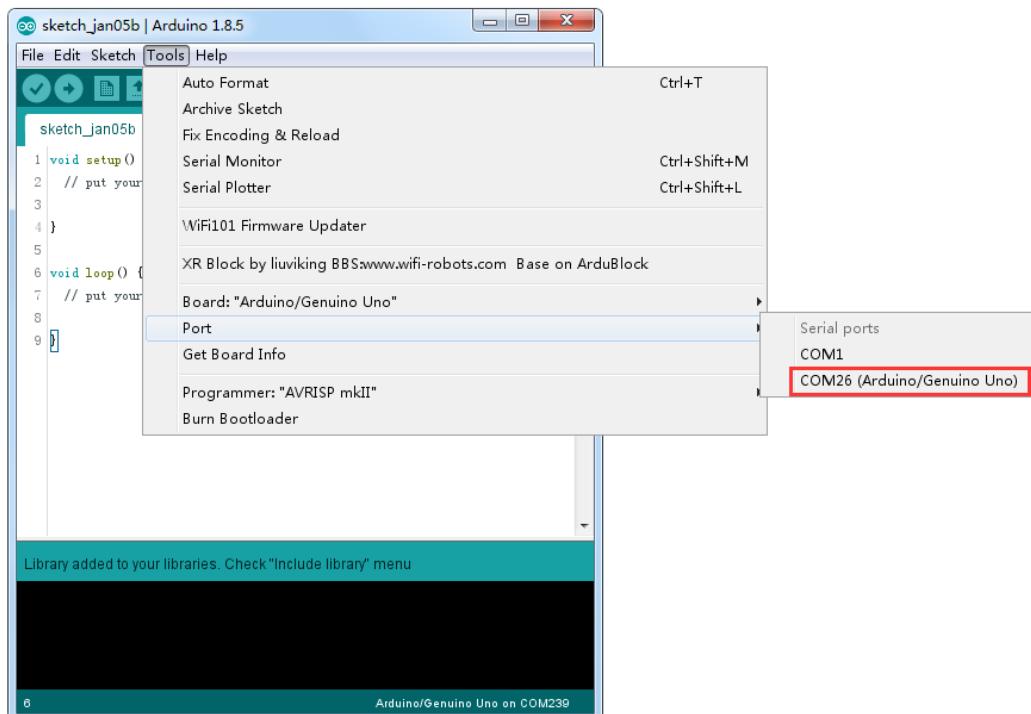
Making a Connection

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

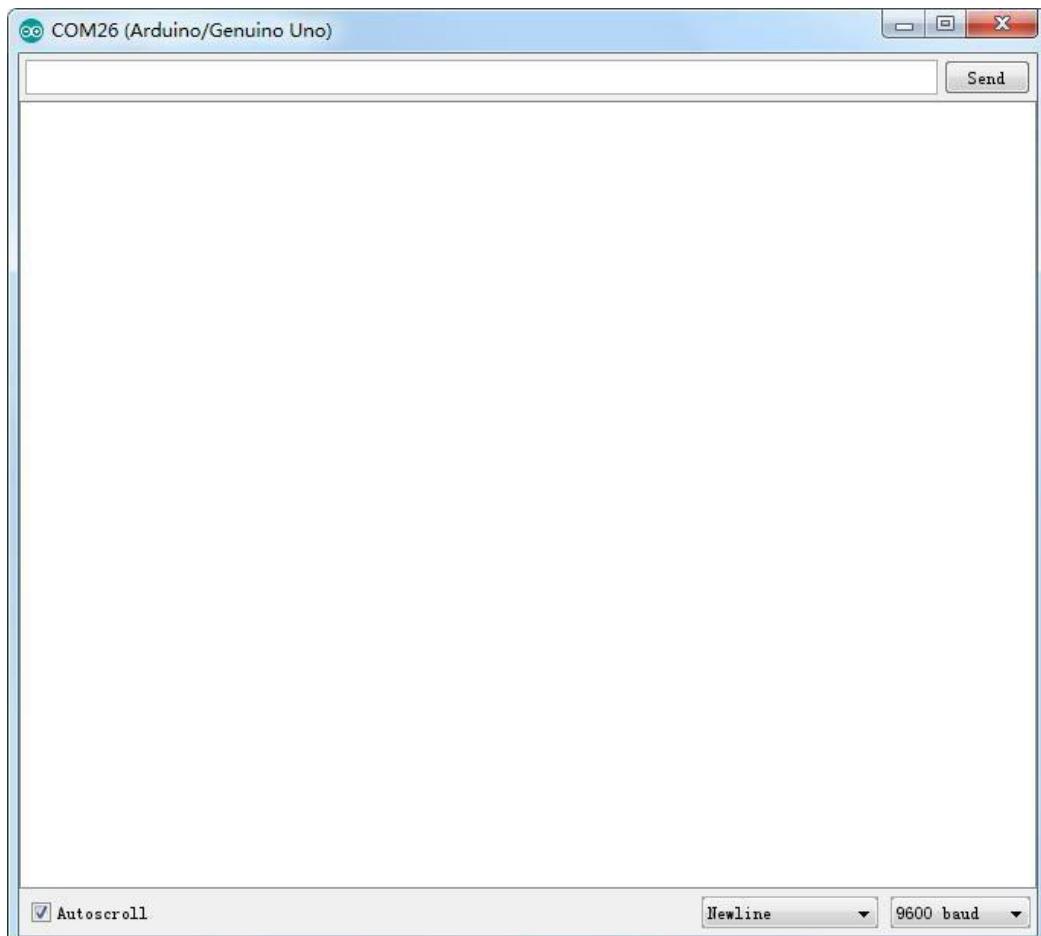


Selecting which port to open in the Serial Monitor is the same as selecting a port for uploading Arduino code. Go to Tools -> Serial Port, and select the correct port.

Tips: Choose the same COM port that is shown in Device Manager.

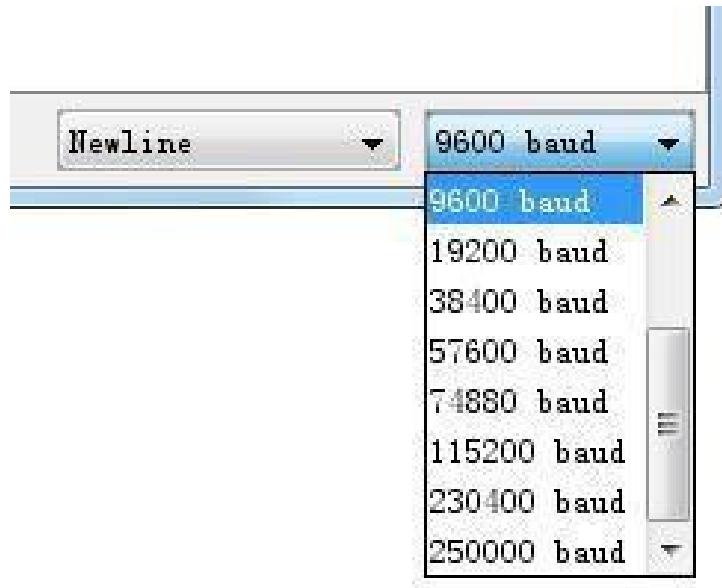


Once open, you should see something like this:

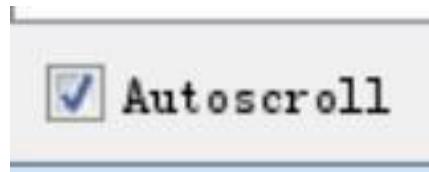


Settings

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 Auto scroll or not by checking the box in the bottom left corner.



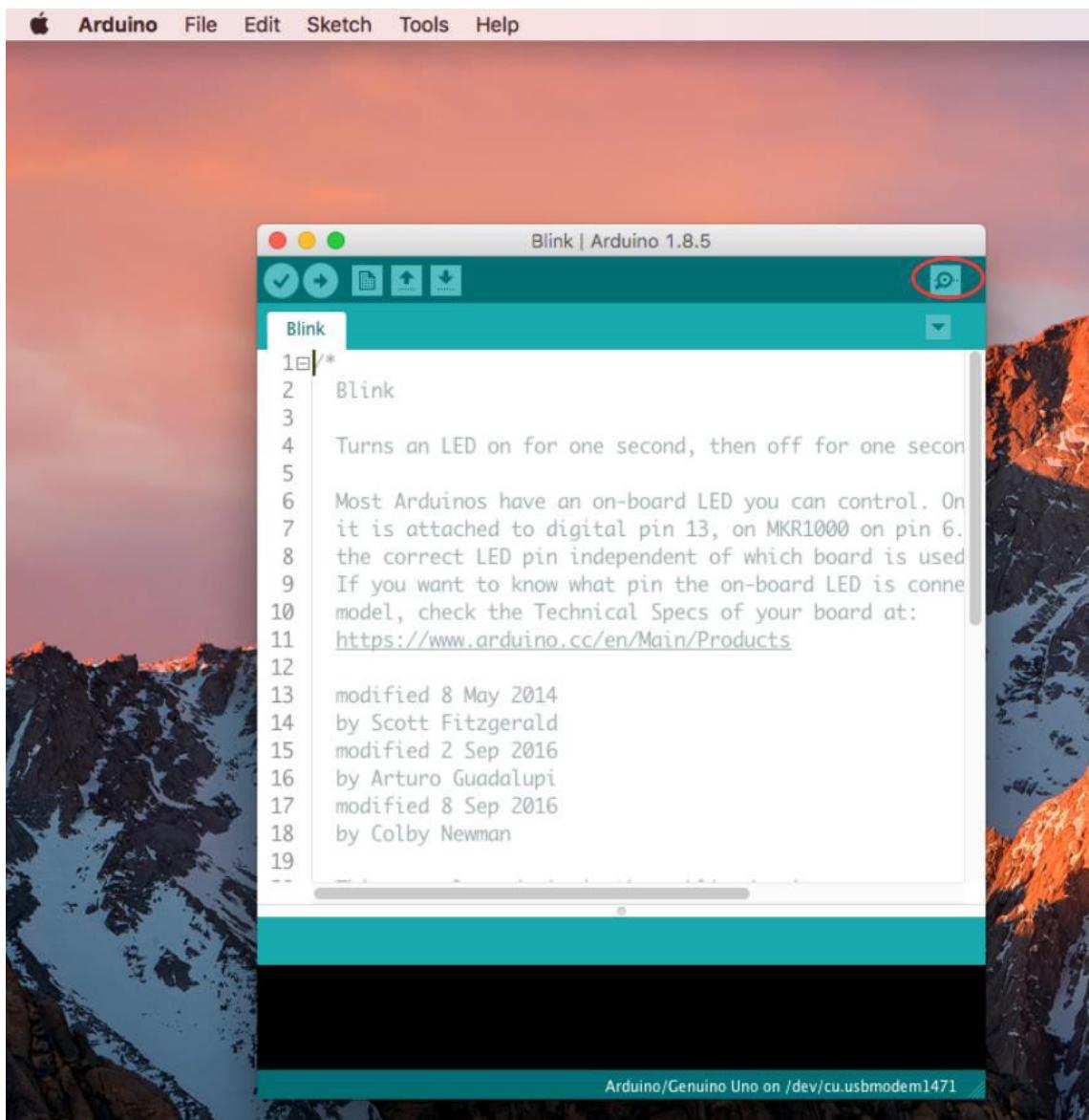
Pros

The Serial Monitor is a great way to quickly and easily 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

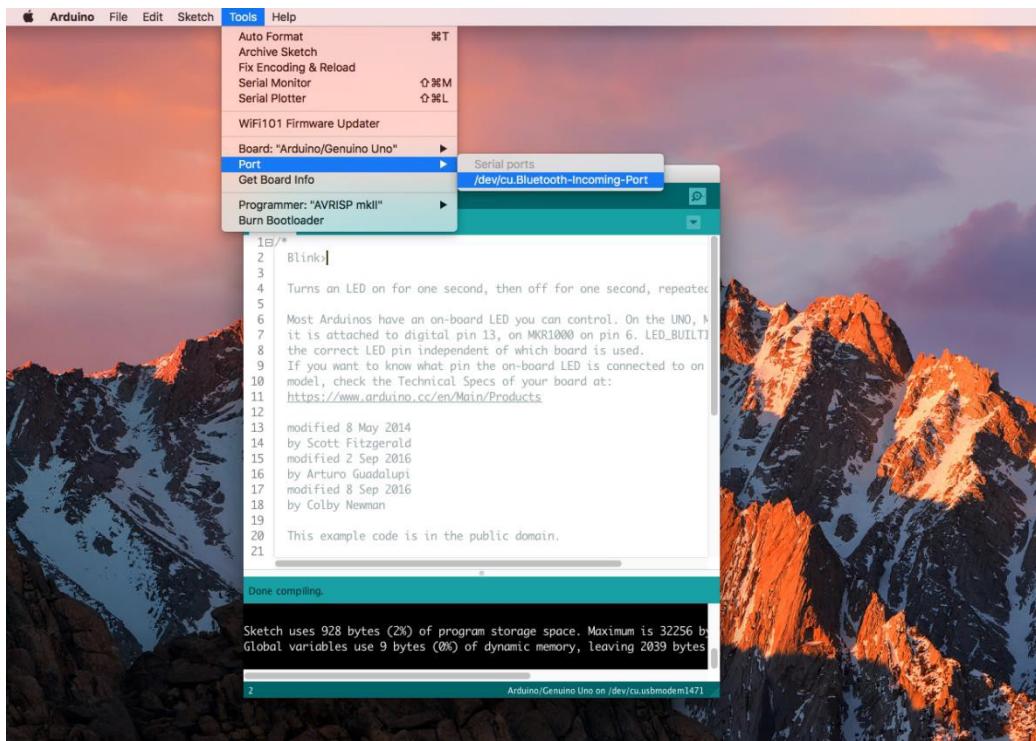
The lack of settings leaves much to be desired in the Serial Monitor, and, for advanced serial communications, it may not do the trick.

In Mac environment, open the serial port monitor:

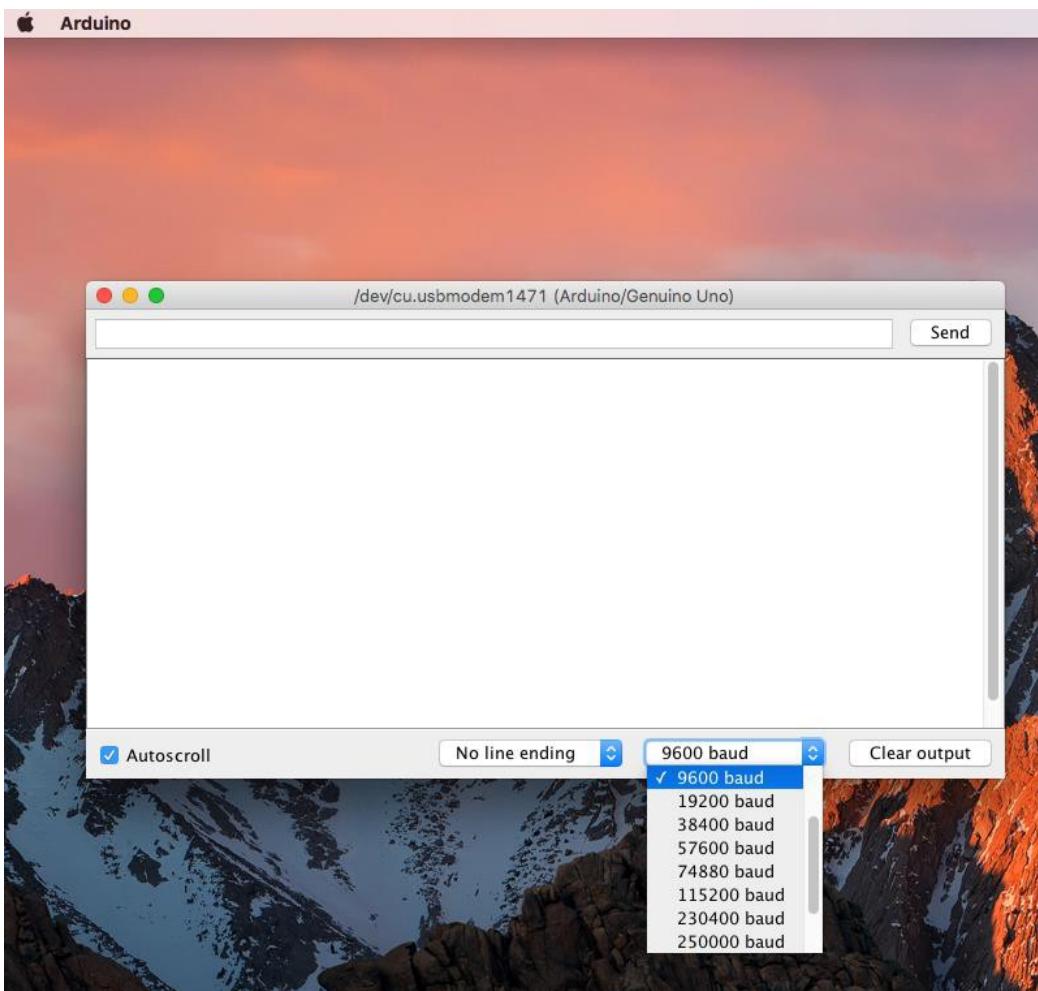


Selecting which port to open in the Serial Monitor is the same as selecting a port for uploading Arduino code. Go to Tools -> Serial Port, and select the correct port.

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

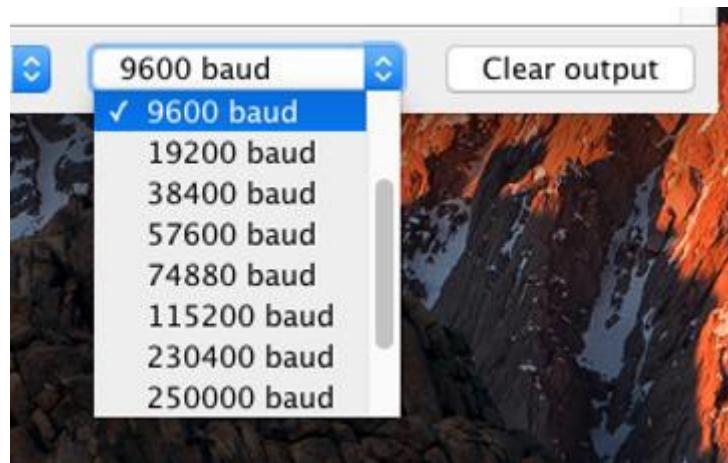


Once open, you should see something like this:

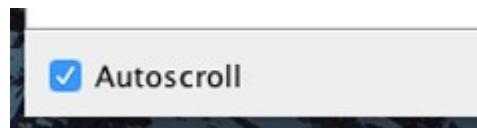


Settings

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 Auto scroll 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.

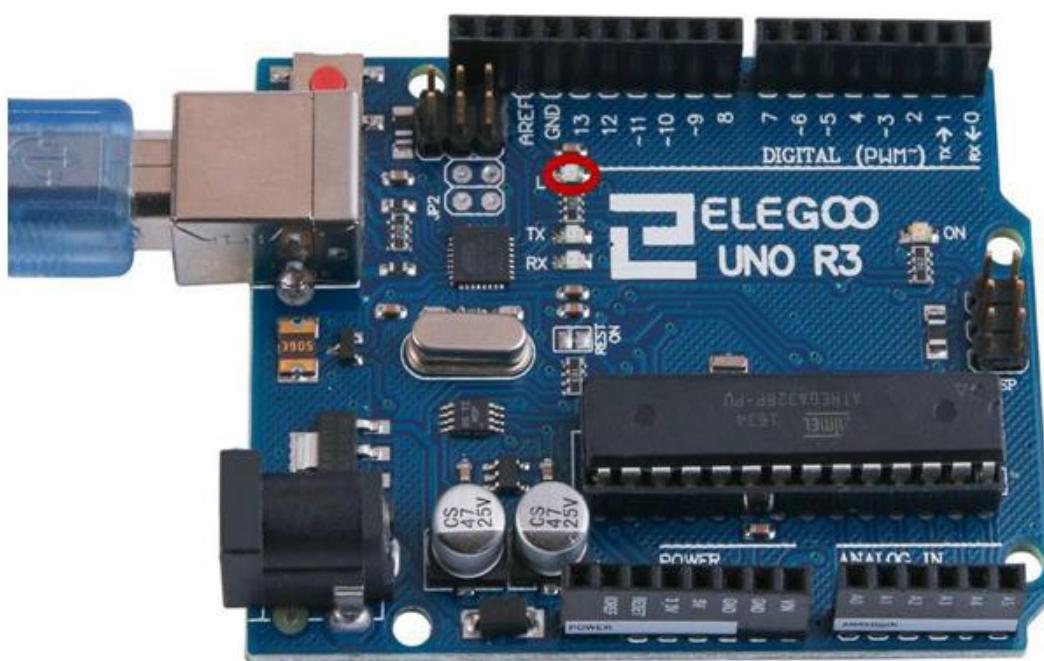
Lesson 3 Blink

Overview

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

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.



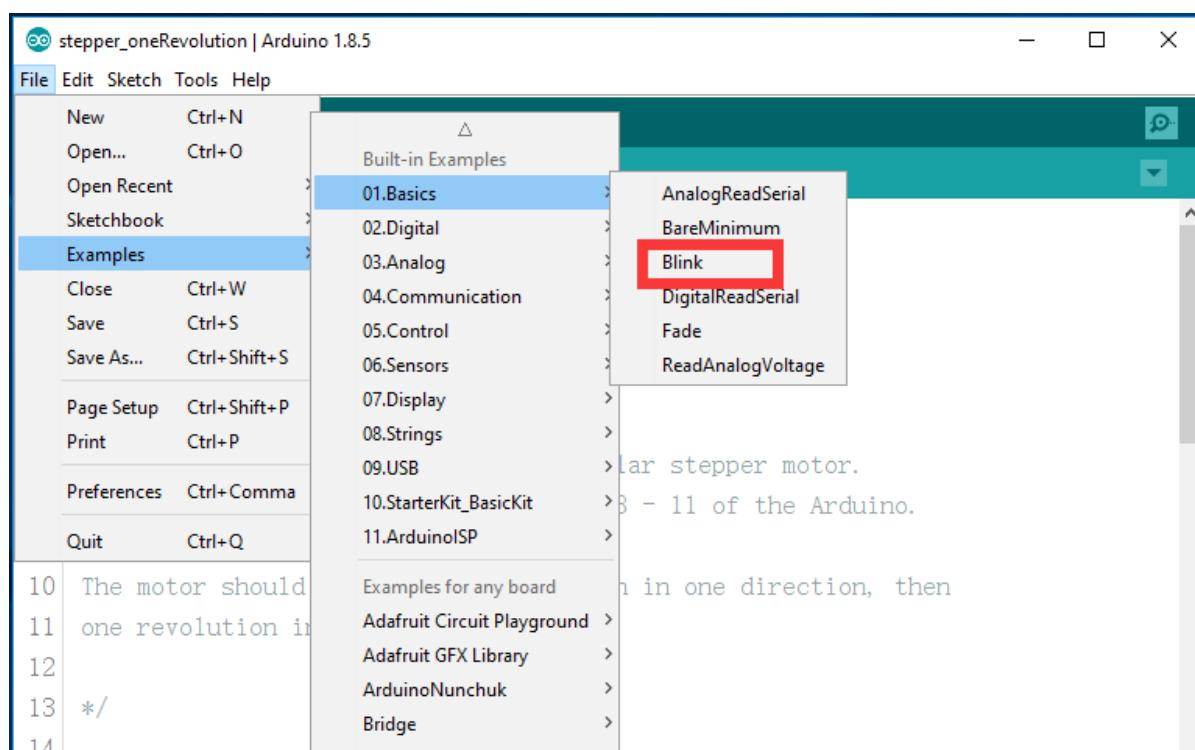
You may find that your UNO R3 board's 'L' LED already blinks when you connect it to a USB plug. This is because the boards are generally shipped with the 'Blink' sketch pre-installed.

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

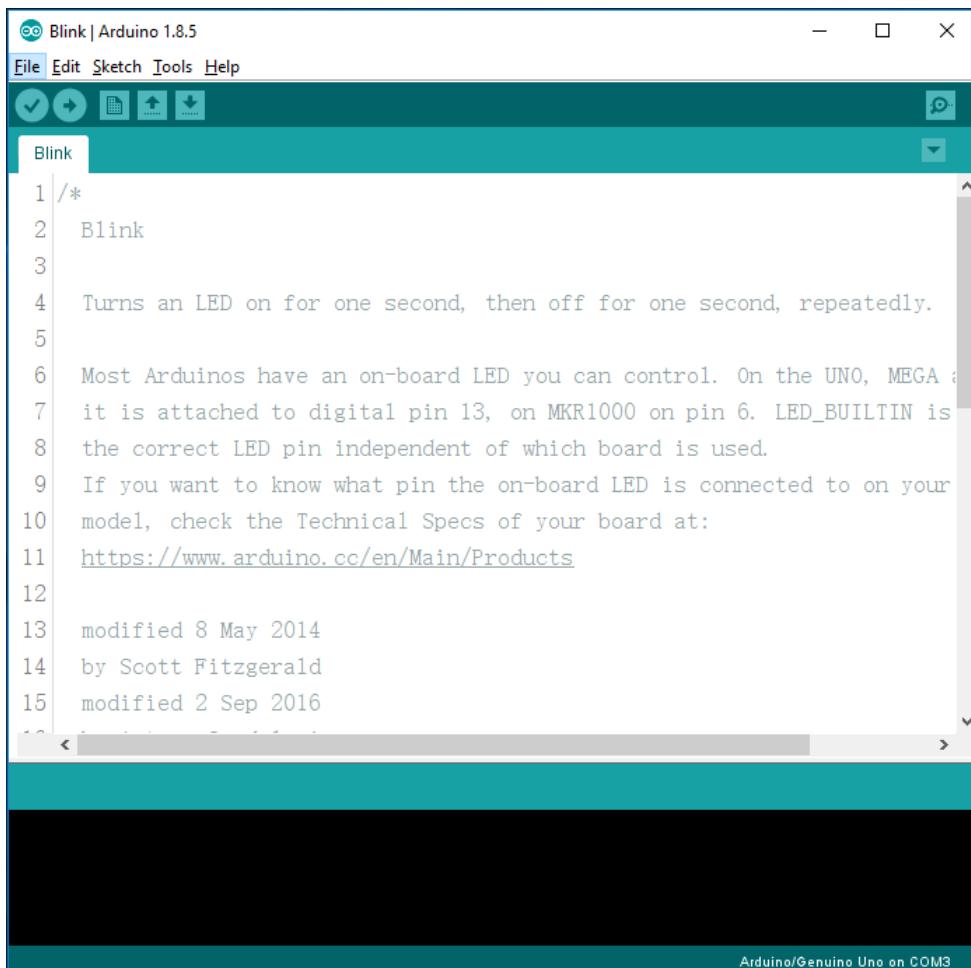
In Lesson 0, you set up your Arduino IDE and made sure that you could find the right serial port for it to connect to your UNO R3 board. The time has now come to put that connection to the test and program your UNO R3 board.

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

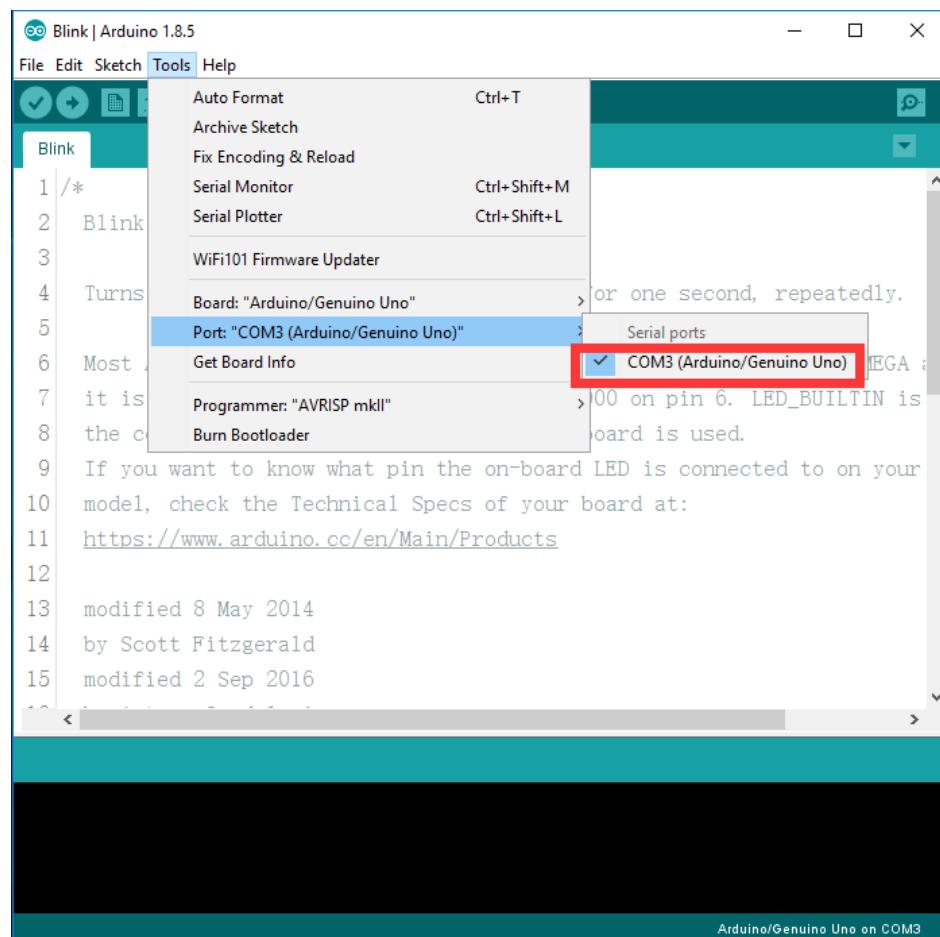
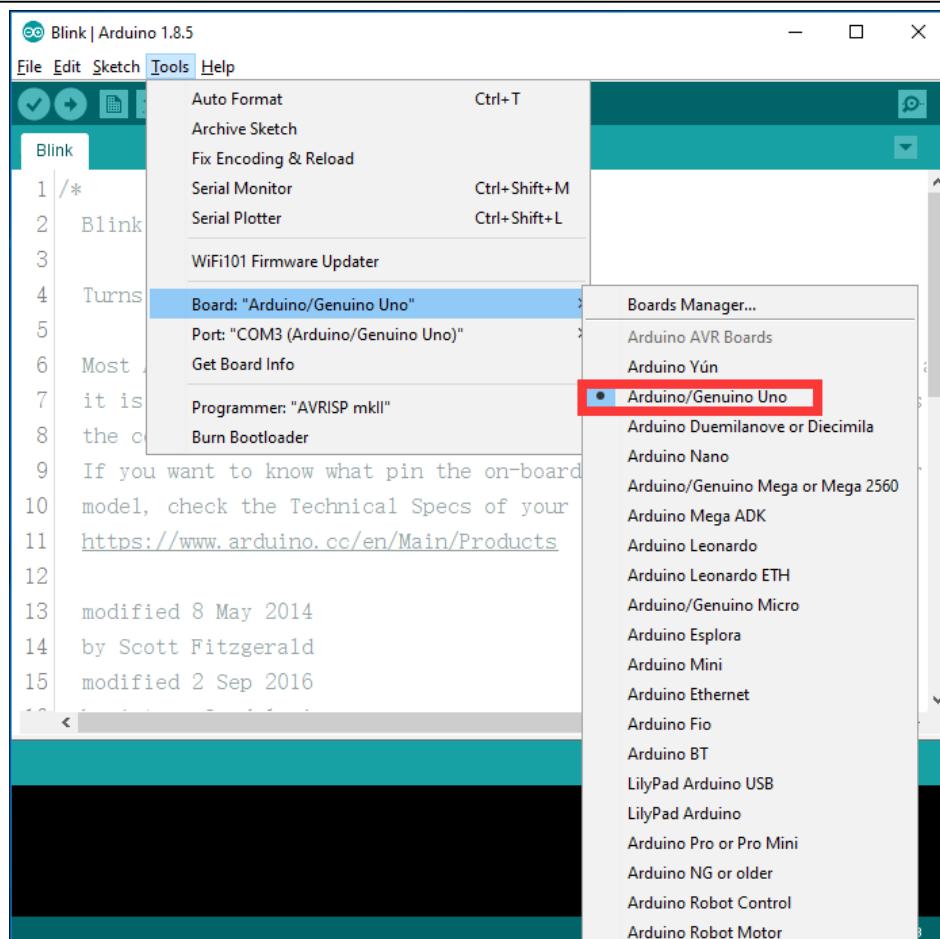


When the sketch window opens, enlarge it so that you can see the entire sketch in the window.



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.

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



Note: The Board Type and Serial Port here are not necessarily the same as shown in picture. If you are using 2560, then you will have to choose Mega 2560 as the Board Type, other choices can be made in the same manner. And the Serial Port displayed for everyone is different, despite COM 3 chosen here, it could be COM4 or COM8 on your computer. A right COM port is supposed to be COMX (arduino XXX), which is by the certification criteria.

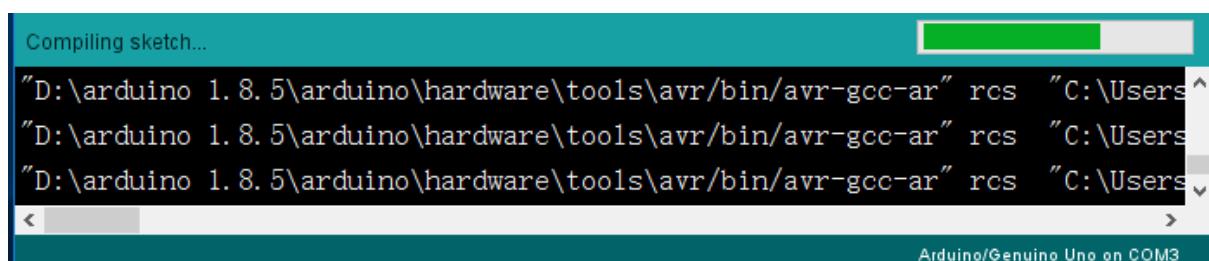
The Arduino IDE will show you the current settings for board at the bottom of the window.



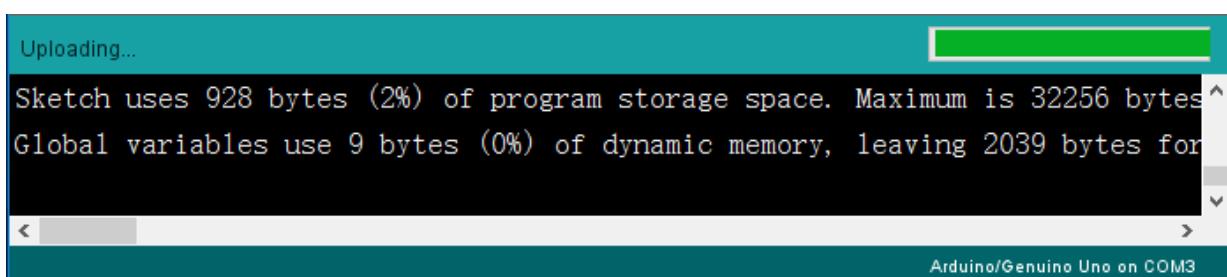
Click on the 'Upload' button. On the toolbar, this is the second button from the left, represented by an arrow pointing to the right.



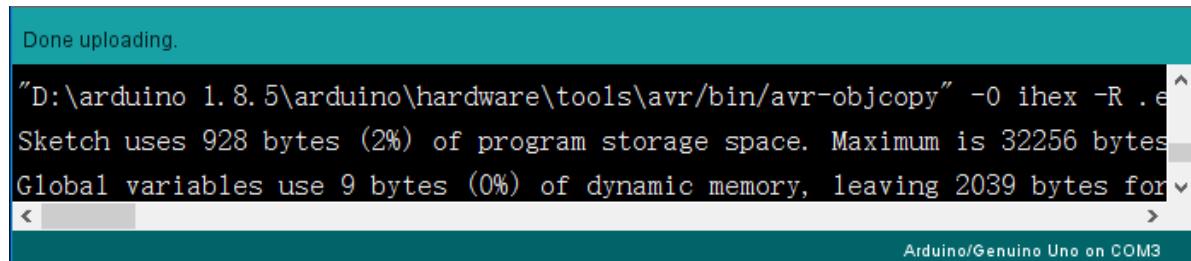
If you watch the status area of the IDE, you will see a progress bar and a series of messages. At first, it will say 'Compiling Sketch...'. This converts the sketch into a format suitable for uploading to the board.



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 status will change to 'Done'.



Done uploading.
"D:\arduino 1.8.5\arduino\hardware\tools\avr/bin/avr-objcopy" -O ihex -R .e
Sketch uses 928 bytes (2%) of program storage space. Maximum is 32256 bytes
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for v
Arduino/Genuino Uno on COM3

The other message tells us that the sketch is using 928 bytes of the 32,256 bytes available. : If you fail uploading, after the 'Compiling Sketch...', you will get the following error message:



Problem uploading to board. See http://www.arduino.cc/en/Troubleshooting/UploadProblem
avrduude: st1k500_recv() programmer is not responding
avrduude: st1k500_getsync() attempt 10 of 10: not in sync resp=0x22
Problem uploading to board. See http://www.arduino.cc/en/Guide/Troubleshooting
Arduino/Genuino Uno on COM1

This error is generic and can mean one of three things: Either your board isn't connected, the drivers haven't been installed, or 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.

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 immediately start again after it has finished running its commands.

```
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 two times faster, change the value from 1000 to 500. Causes the program to instead pause for only half of a second between each statechange.

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

Under the Mac environment

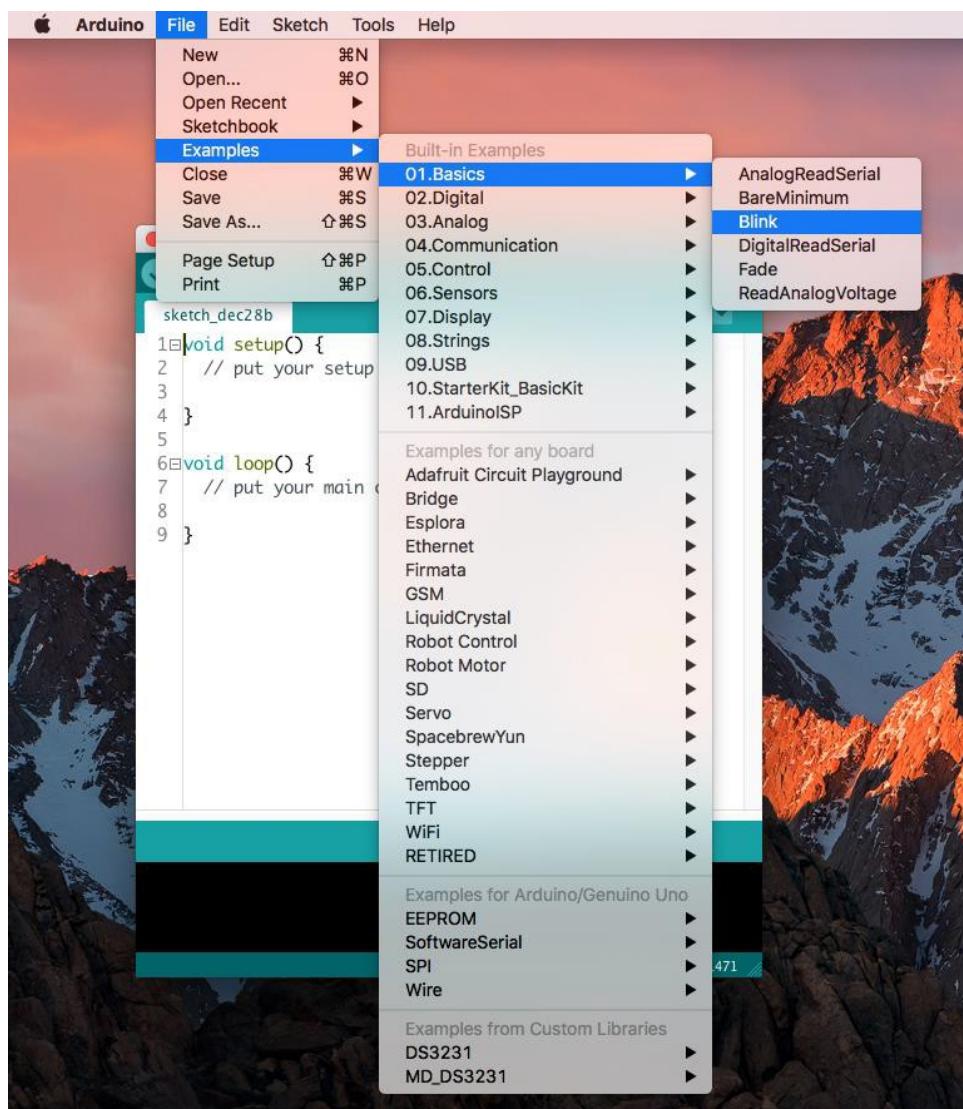
You may find that your UNO R3 board's 'L' LED already blinks when you connect it to a USB plug. This is because the boards are generally shipped with the 'Blink' sketch pre-installed.

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

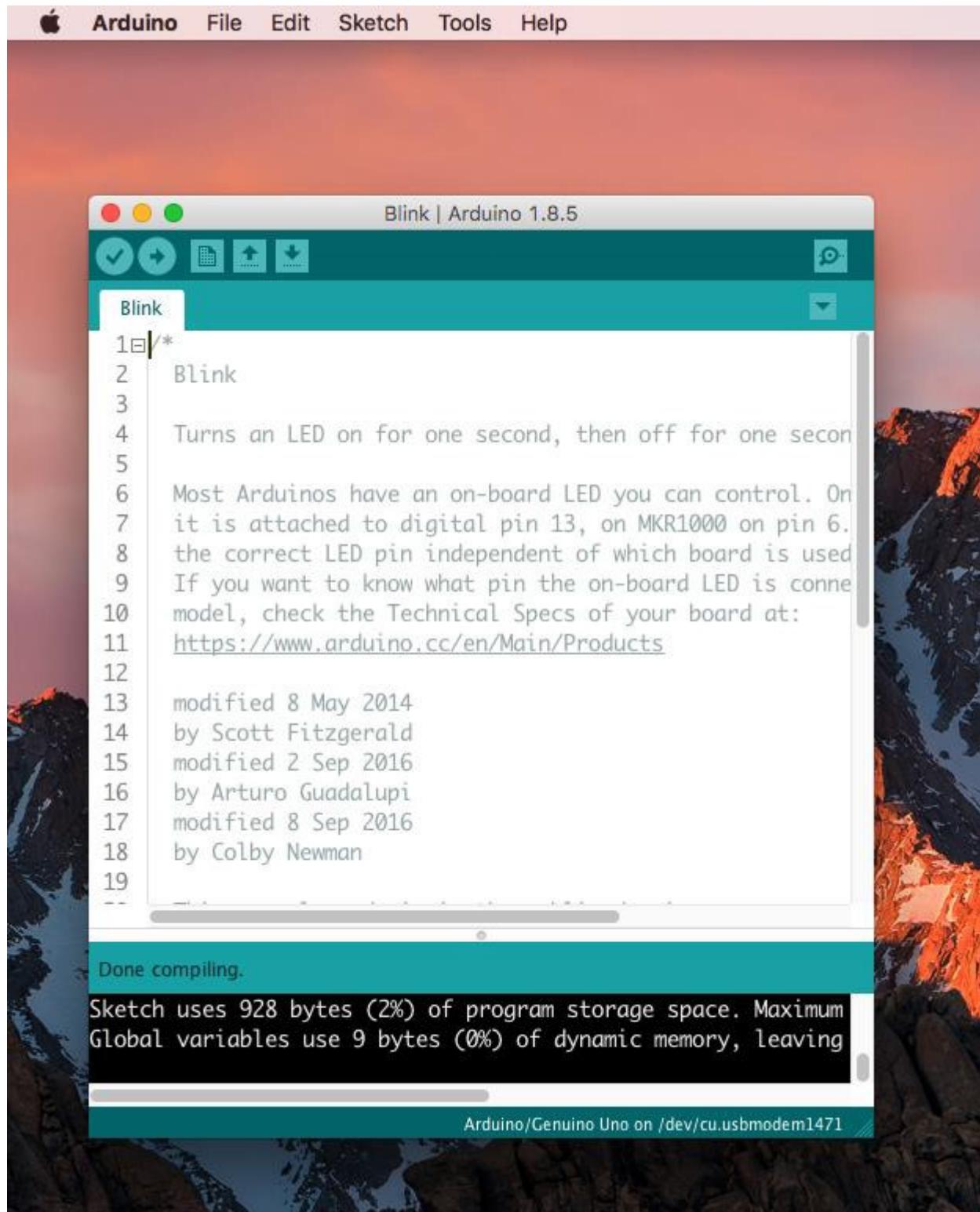
In Lesson 0, you set up your Arduino IDE and made sure that you could find the right serial port for it to connect to your UNO R3 board. The time has now come to put that connection to the test and program your UNO R3 board.

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.

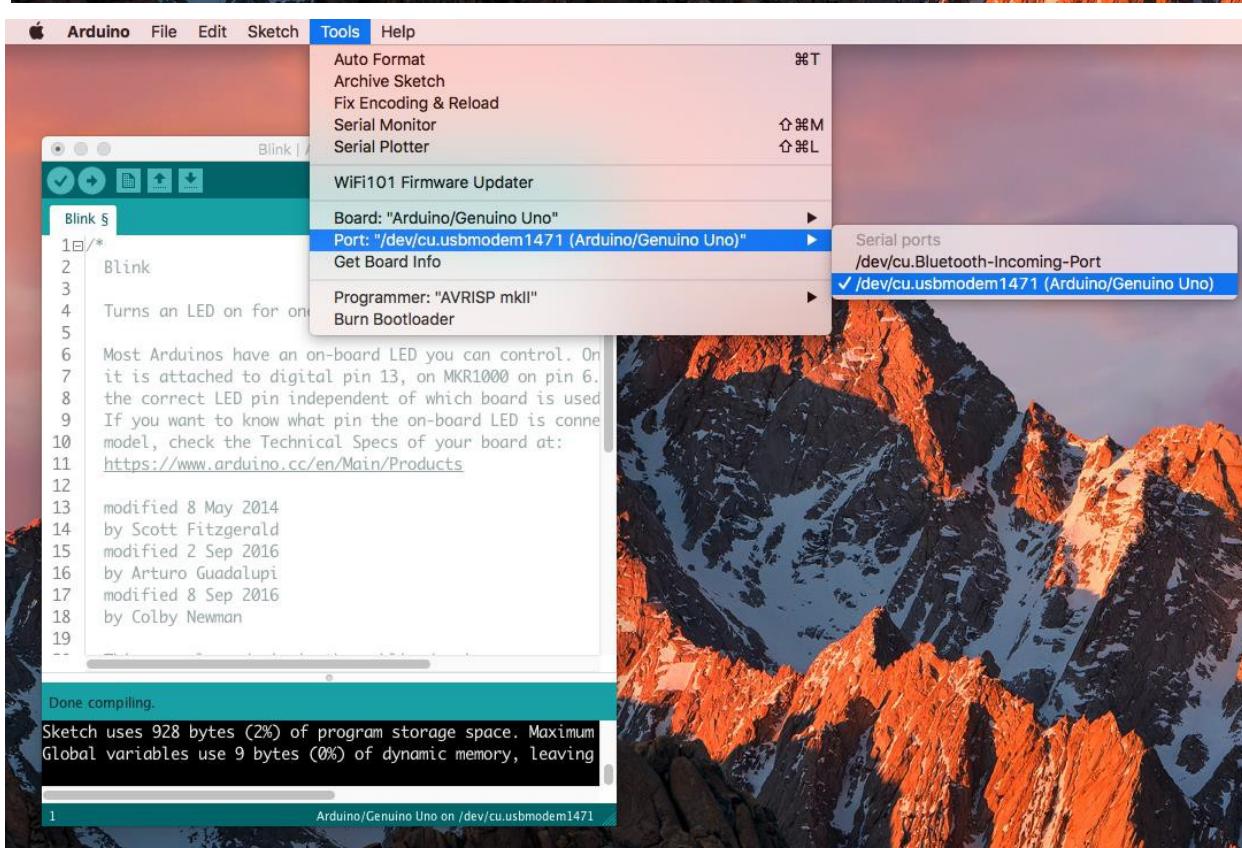
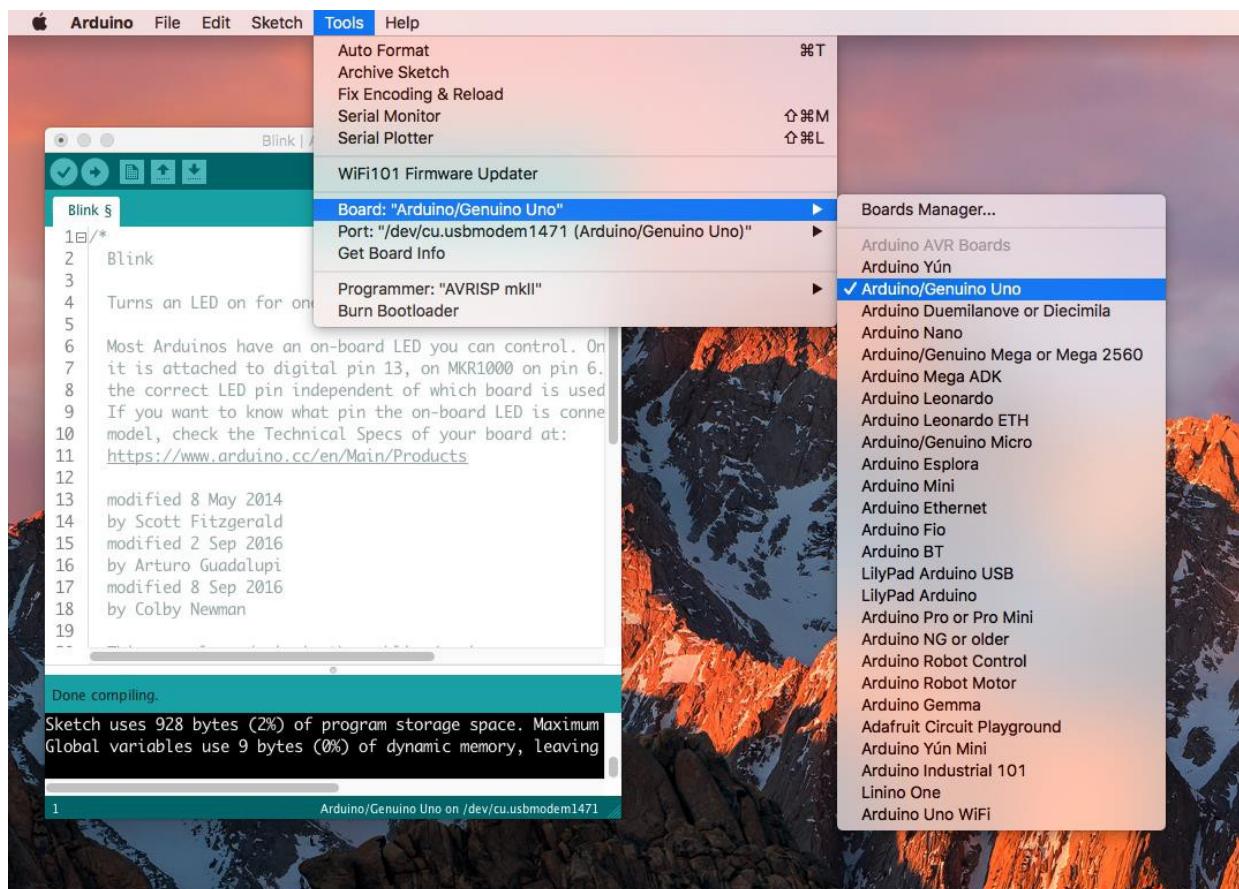
Load the 'Blink' sketch that you will find in the IDE's menu system under File > Examples > 01. Basics



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



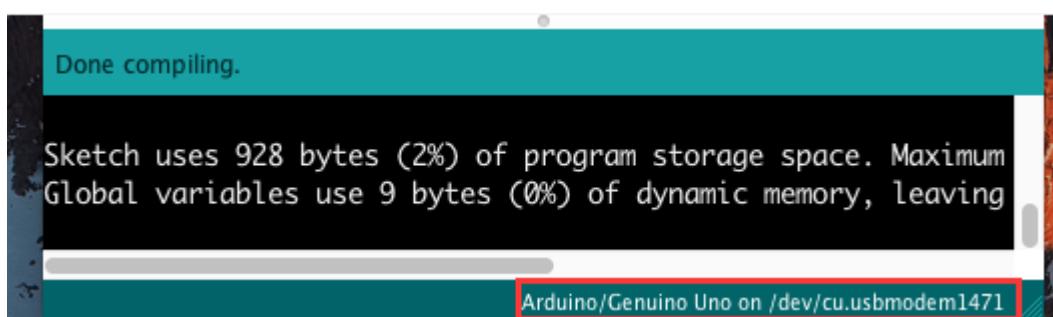
Attach your Arduino board to your computer with the USB cable and check that the 'Board Type' and 'Serial Port' are set correctly.



Note: The Board Type and Serial Port here are not necessarily the same as shown in picture. If you are using 2560, then you will have to choose Mega 2560 as the Board Type, other choices can be made in the same manner. And the Serial Port displayed for everyone is different, despite /dev/cu.usbmodem1471 (Arduino/Genuino Uno),

Chosen here, it could be /dev/cu.usbmodemfa121(Arduino Uno) on your computer. A right COM port is supposed to be /dev/cu.usbmodemfaXX (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.



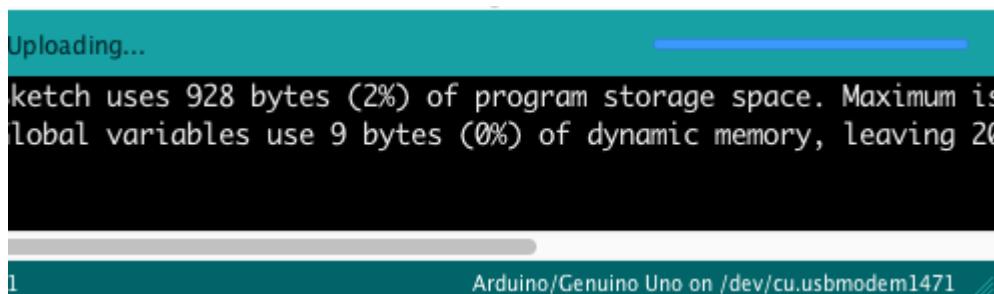
Click on the 'Upload' button. The second button from the left on the toolbar.



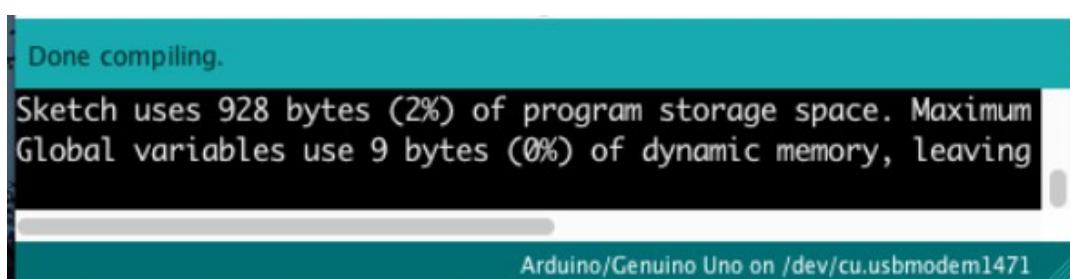
If you watch the status area of the IDE, you will see a progress bar and a series of messages. At first, it will say 'Compiling Sketch...'. This converts the sketch into a format suitable for uploading to the board.



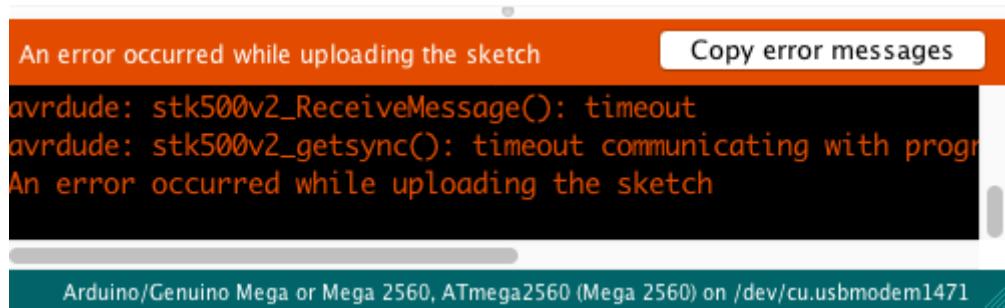
Next, the status will change to 'Uploading'. At this point, the LEDs on the Arduino should start to flicker as the sketch is transferred.



Finally, the status 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 4 Temperature and Humidity Module

Overview

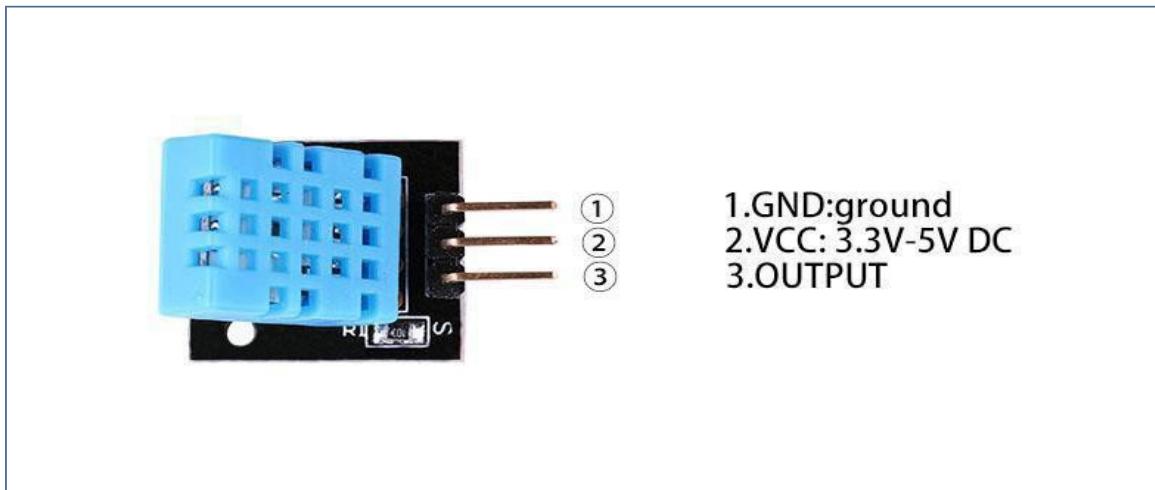
In this tutorial we will learn how to use a DHT11 Temperature and Humidity Sensor. It's accurate enough for most projects that need to keep track of humidity and temperature readings.

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

Temperature and Humidity

A module with a temperature/humidity sensor type DHT11, Temperature range: -20 - 60°C (+/-1°C), Rel. humidity: 5-95% (+/5%), Supply voltage: 3 to 5.5V. With a built-in 10 K ohm pull up resistor.

Library: DHT.h



Component Required:

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

Component Introduction

Temp and humidity sensor:

DHT11 digital temperature and humidity sensor is a composite Sensor which contains a calibrated digital signal output of the temperature and humidity. The dedicated digital modules collection technology and the temperature and humidity sensing technology are applied to ensure that the product has high reliability and excellent long-term stability. The sensor includes a resistive sense of wet components and a NTC temperature measurement devices, and connects with a hig-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: 16 Bit
Repeatability: $\pm 1\% \text{ RH}$

Accuracy: At 25°C $\pm 5\% \text{ RH}$

Interchangeability: fullyinterchangeable

Response time: 1 / e (63%) of 25°C 6s

1m / s air 6s

Hysteresis: $<\pm 0.3\% \text{ RH}$

Long-term stability: $<\pm 0.5\% \text{ RH} / \text{yrin}$

Temperature:

Resolution: 16 Bit

Repeatability: $\pm 0.2^\circ\text{C}$

Range: At 25°C $\pm 2^\circ\text{C}$

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

Electrical Characteristics

Power supply: DC3.5 \sim 5.5V

Supply Current: measurement 0.3mA standby $60\mu\text{A}$

Sampling period: more than 2seconds

Pin Description:

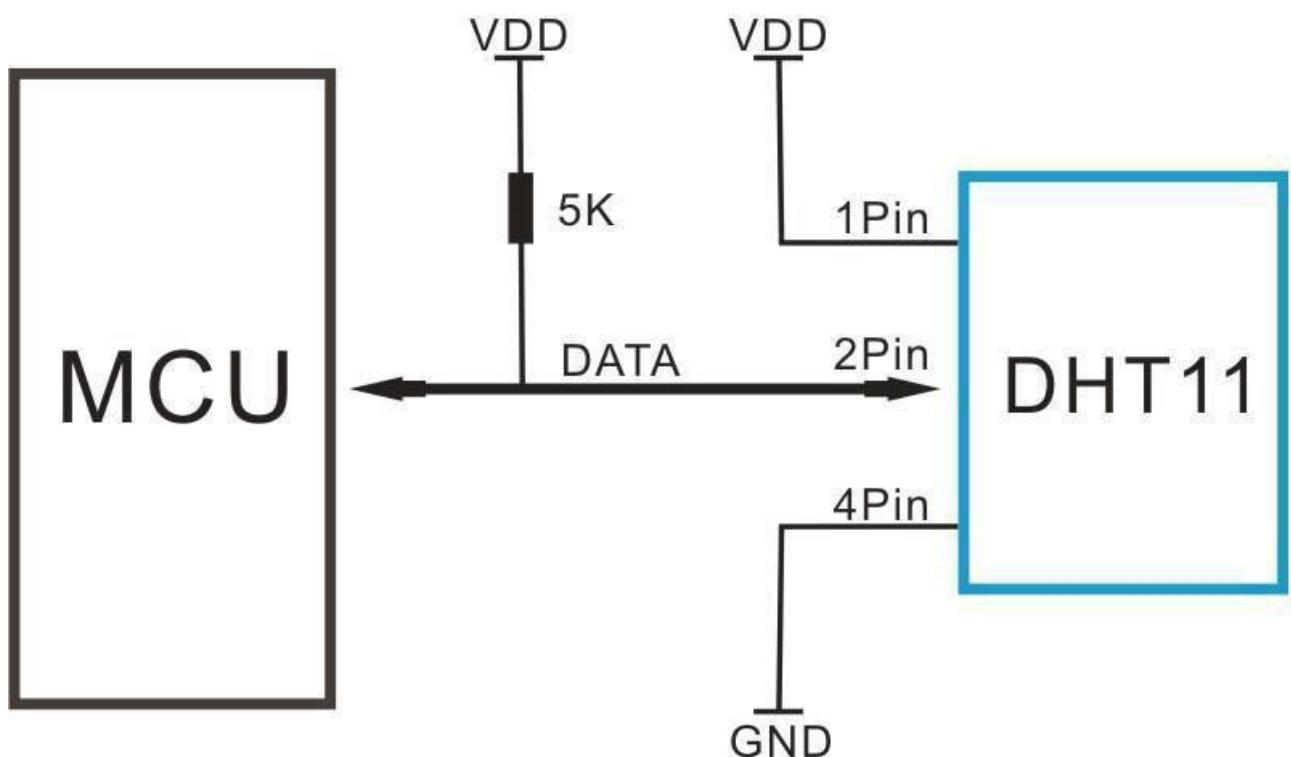
1. the VDD power supply 3.5 \sim 5.5V DC

2. DATA serial data, a single bus

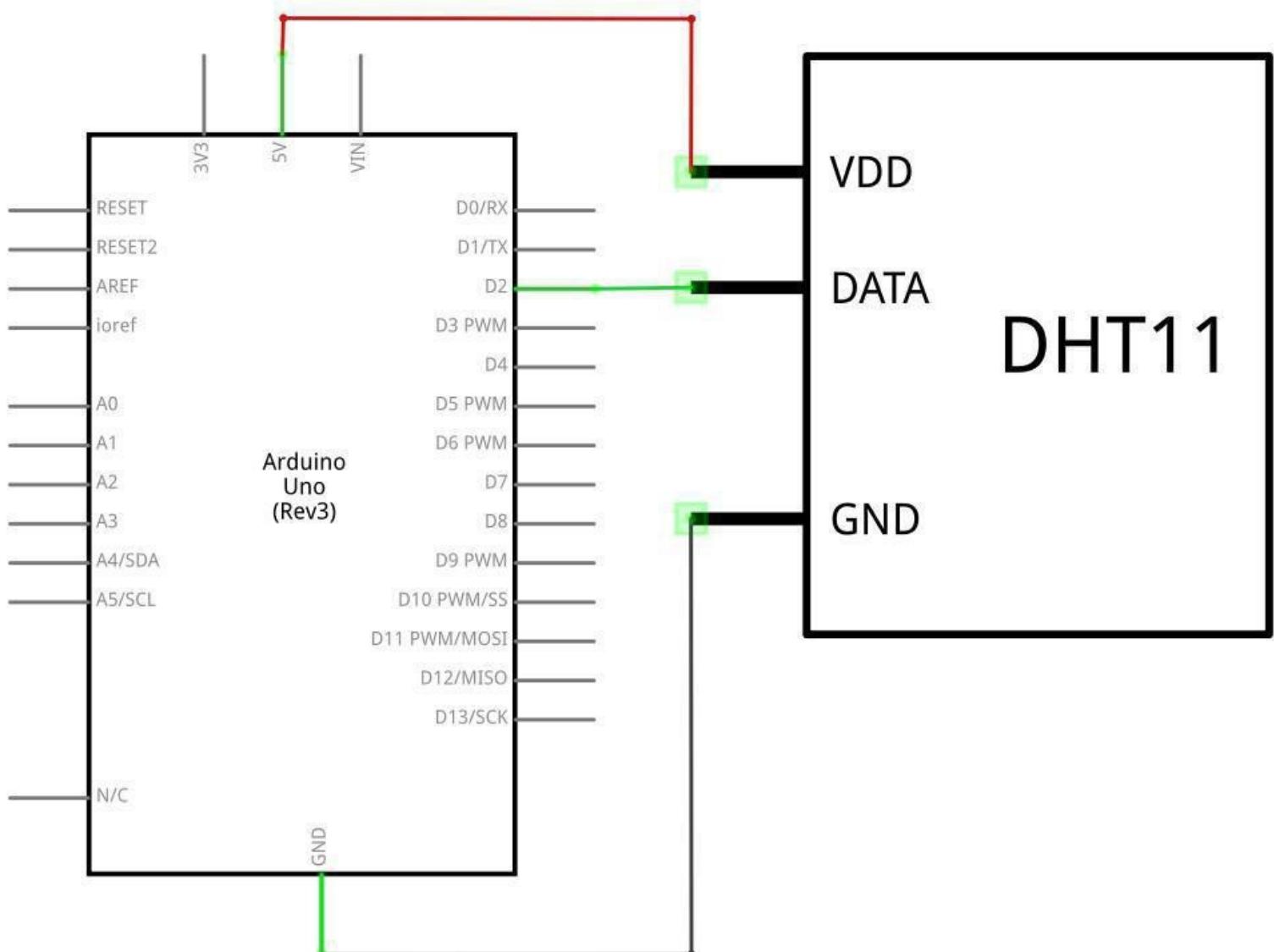
3. NC, empty pin

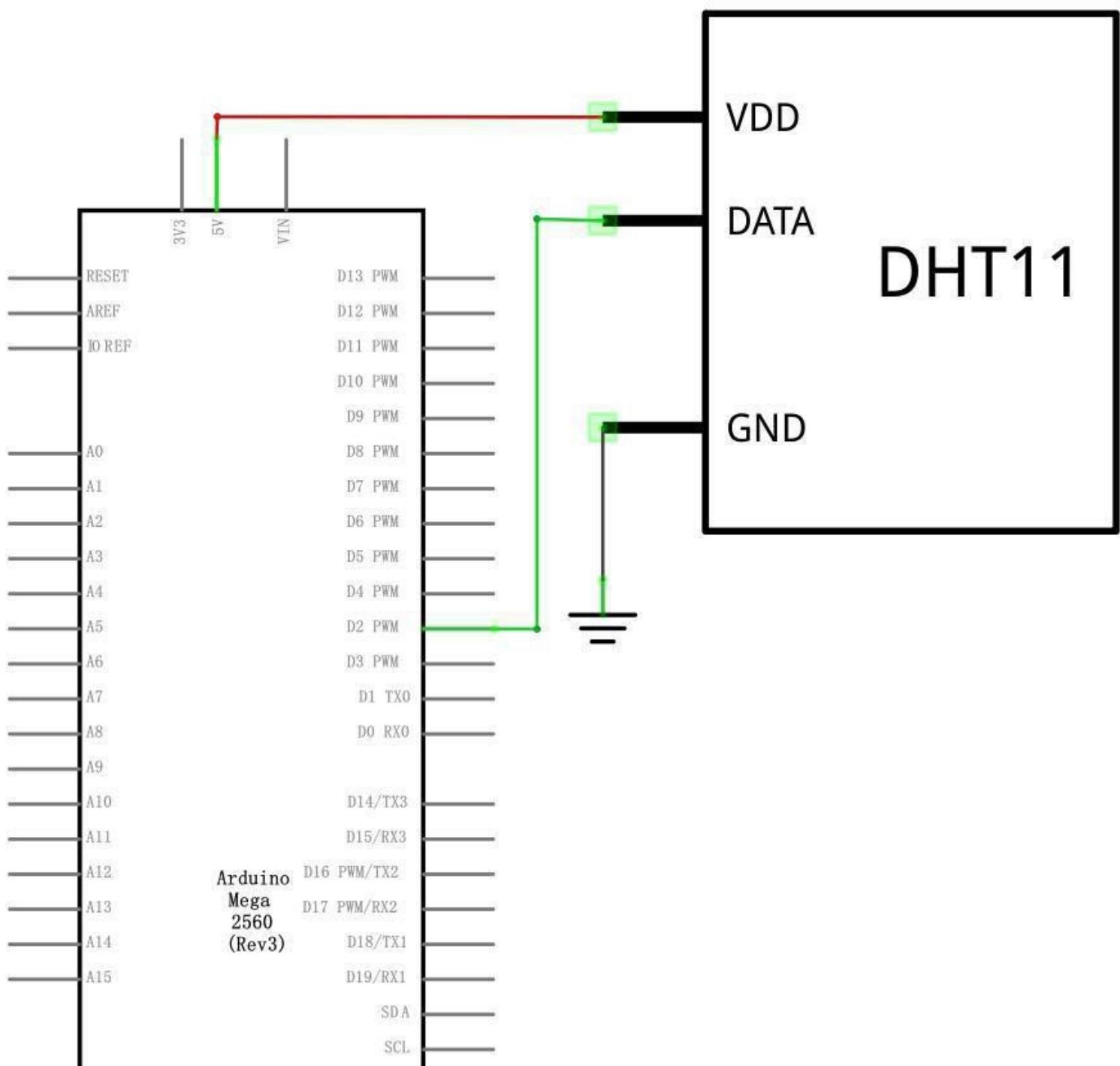
4. GND ground, the negative power

Typical Application

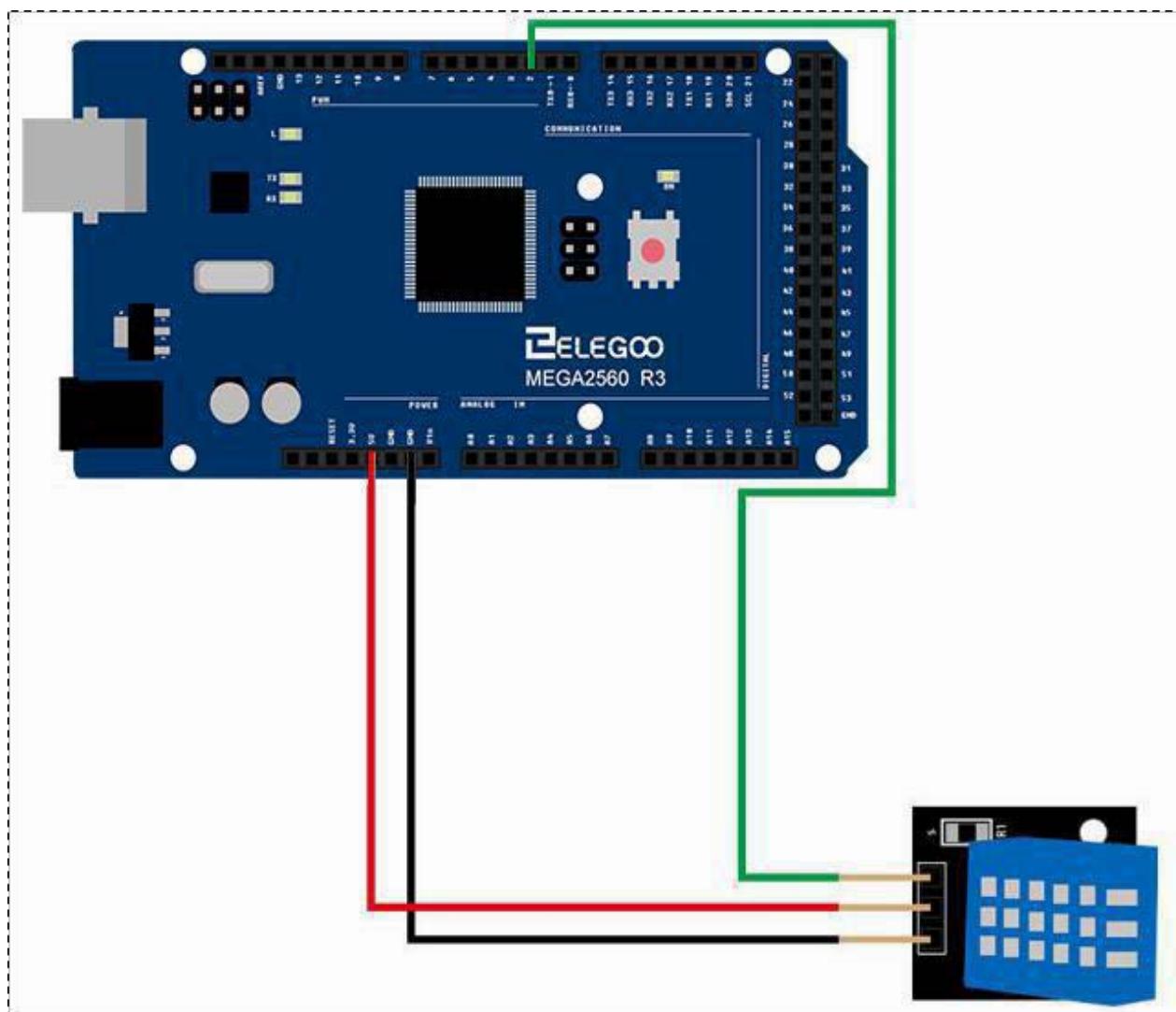
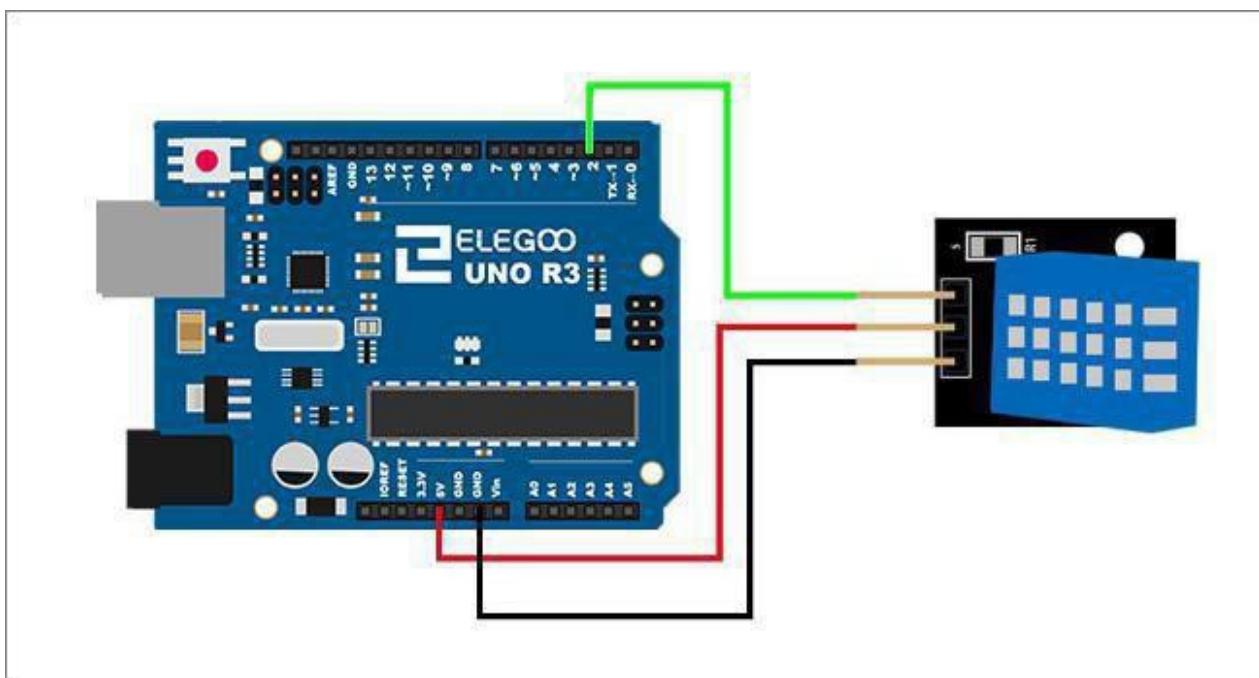


Connection Schematic





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.

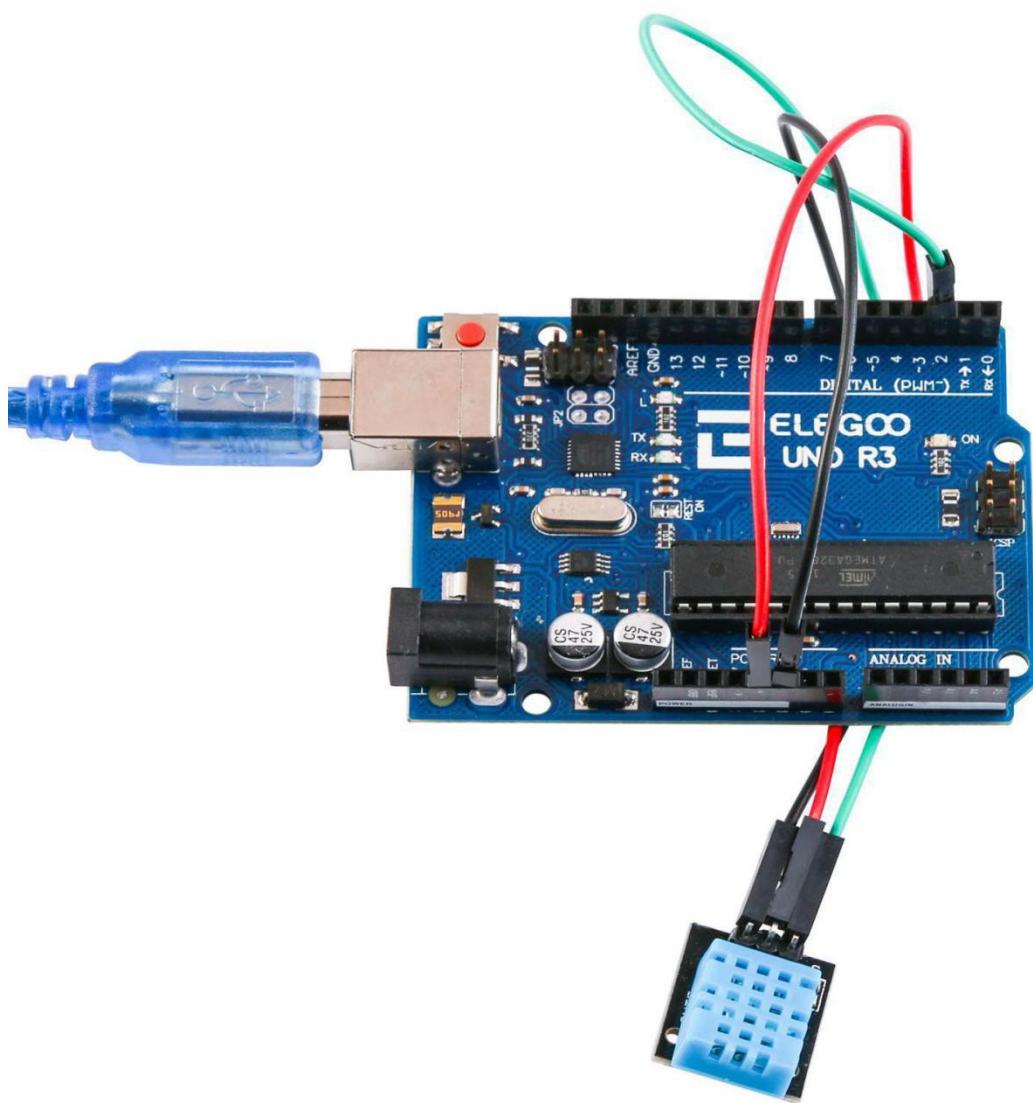
Code

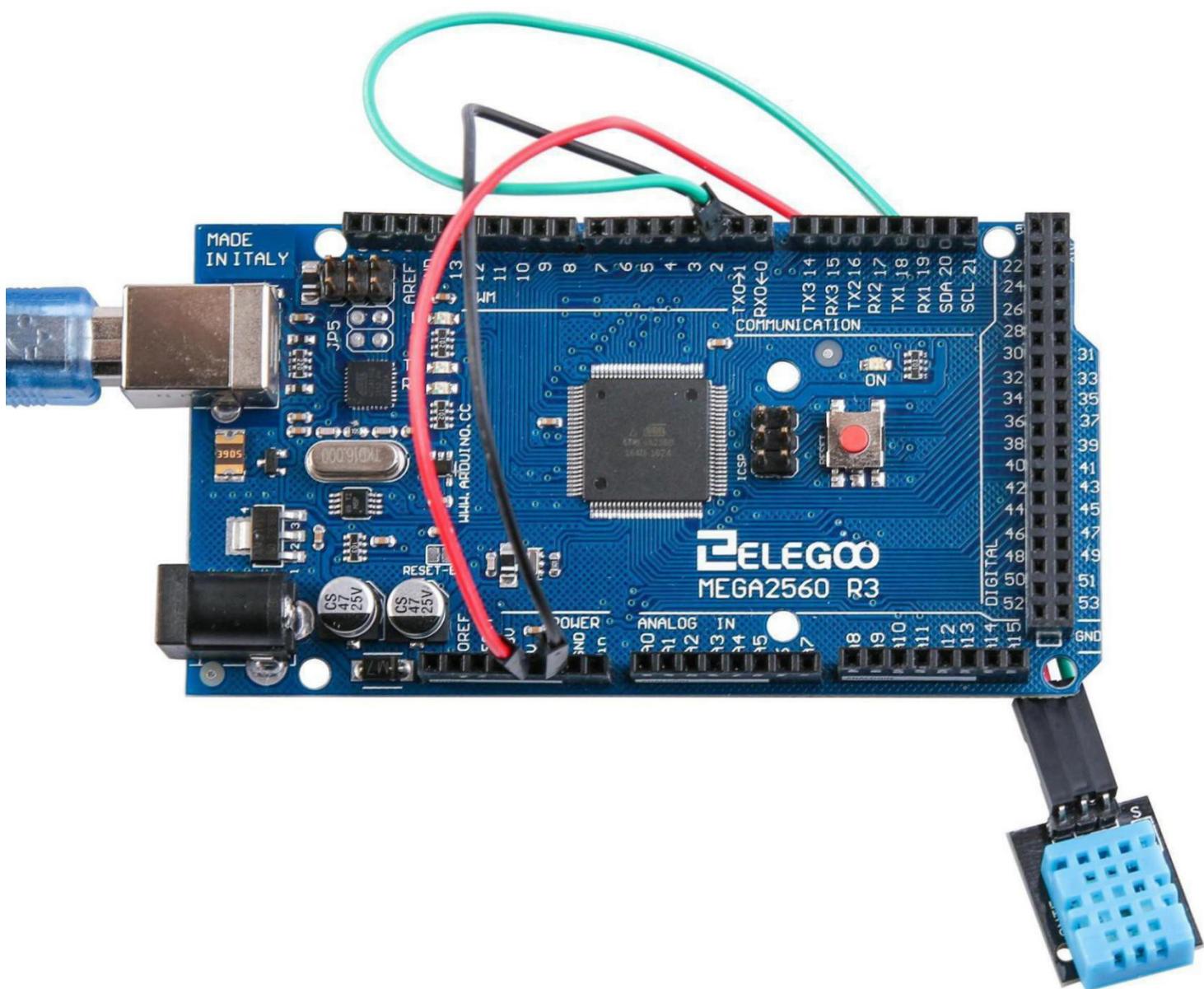
After wiring, please open the program in the code folder- (Lesson 4 TEMP AND HUMIDITY MODULE) and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

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

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

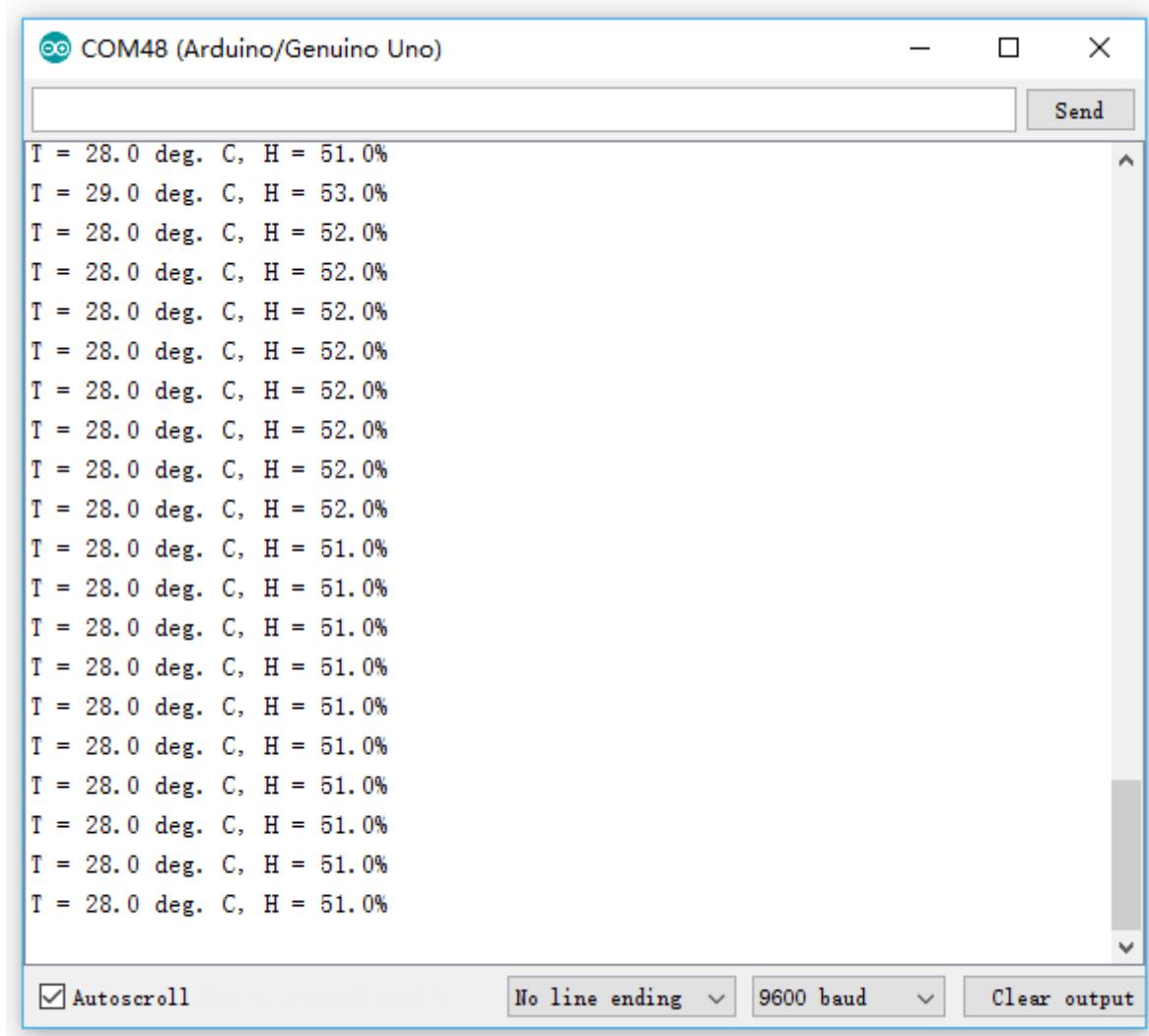
Result





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 28 degree)

Click the Serial Monitor button to turn on the serial monitor. The basics about the serial monitor are introduced in details in Lesson 2.



The followings are the code used in this experiment and their explanations:

```
#include<dht_nonblocking.h>

#define DHT_SENSOR_TYPE DHT_TYPE_11
//#define DHT_SENSOR_TYPE DHT_TYPE_21
//#define DHT_SENSOR_TYPE DHT_TYPE_22

static const int DHT_SENSOR_PIN = 2;
DHT_nonblocking dht_sensor( DHT_SENSOR_PIN, DHT_SENSOR_TYPE );

void setup( )
{
    Serial.begin( 9600 );
}

void loop( )
{
    float temperature;
    float humidity;

    if(dht_sensor.measure(&temperature, &humidity)){
        Serial.print( "T = " );
        Serial.print( temperature, 1 );
        Serial.print( " deg. C, H = " );
        Serial.print( humidity, 1 );
        Serial.println( "%" );
    }
}
```

From the above program we learned the programming syntax control structure:

(1)if

if tests whether a certain condition has been reached, such as an input being above a certain number.

The format for an if test is: if (someVariable > 50) { // do something here }

The program tests to see if someVariable is greater than 50. If it is, the program takes a particular action.

Put another way, if the statement in parentheses is true, the statements inside the brackets are run. If not, the program skips over the code.

The brackets may be omitted after an if statement. If this is done, the next line (defined by the semicolon) becomes the only conditional statement.

```
if (x > 120) digitalWrite(LEDpin, HIGH);  
if (x > 120) digitalWrite(LEDpin, HIGH);  
if (x > 120) {digitalWrite(LEDpin, HIGH);} // all are correct
```

The statements being evaluated inside the parentheses require the use of one or more operators:

Operators: $x == y$ (x is equal to y) $x != y$ (x is not equal to y) $x < y$ (x is less than y) $x > y$ (x is greater than y) $x \leq y$ (x is less than or equal to y) $x \geq y$ (x is greater than or equal to y)

Warning:

Beware of accidentally using the single equal sign (e.g. if ($x = 10$)). The single equal sign is the assignment operator, and sets x to 10. Instead use the double equal sign (e.g. if ($x == 10$)), which is the comparison operator, and tests whether x is equal to 10 or not. The latter statement is only true if x equals 10, but the former statement will always be true. This is because C evaluates the statement if ($x=10$) as follows: 10 is assigned to x , so x now contains 10. Then the 'if' conditional evaluates 10, which always evaluates to TRUE, since any non-zero number evaluates to TRUE. Consequently, if ($x = 10$) will always evaluate to TRUE, which is not the desired result when using an 'if' statement. Additionally, the variable x will be set to 10, which is also not a desired action.

if can also be part of a branching control structure using the if...else] construction.

(2)for statements

The for statement is used to repeat a block of statements enclosed in curly braces. An increment counter is usually used to increment and terminate the loop. The for statement is useful for any repetitive operation, and is often used in combination with arrays to operate on collections of data/pins. There are three parts to the for loop header: for (initialization; condition; increment) { //statement(s); } The initialization happens first and exactly once. Each time through the loop, the condition is tested; if it's true, the statement block, and the increment is executed, then the condition is tested again. When

the condition becomes false, the loop ends.

Example

```
// Dim an LED using a PWM pin int PWMpin = 10; // LED in series with 1k resistor on pin 10
void setup()
{
    // no setup needed
}
void loop()
{
    for (int i=0; i <= 255; i++)
    {
        analogWrite(PWMpin, i);
        delay(10);
    }
}
```

Coding Tip

The C for loop is much more flexible than for loops found in some other computer languages, including BASIC. Any or all of the three header elements may be omitted, although the semicolons are required. Also the statements for initialization, condition, and increment can be any valid C statements with unrelated variables. These types of unusual for statements may provide solutions to some rare programming problems.

Lesson 5 DS18B20 Temperature Sensor Module

Overview

In this experiment, we will learn how to use DS18B20 module test the environmental temperature and make a thermometer.

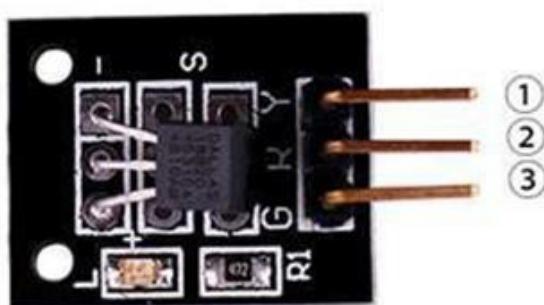
Since the previous temperature sensor output is analog, we need to add additional A/D and D/A chip into the circuit. These cause a big challenge. So we are create the Ds18b20 module.

The new DS18B20 Temperature Sensor Module is very good solve the problem. is economical, has a unique 1-wire bus and is fully compatible with the Arduino platform .Users can easily form a sensor network with this module.

DS18B20 Temperature Sensor

A module with a digital 'One Wire' temperature sensor (DS18B20). A 4.7K ohm pullup resistor is included for the bus signal. Additional sensors can be added to the bus and individually addressed. Only one pullup resistor should be connected to the bus, irrespective of the number of sensors connected.

- Temperature range: -55 to +125°C
- Typical accuracy: 0.5°C
- Resolution: 9-12Bit, depending on the program



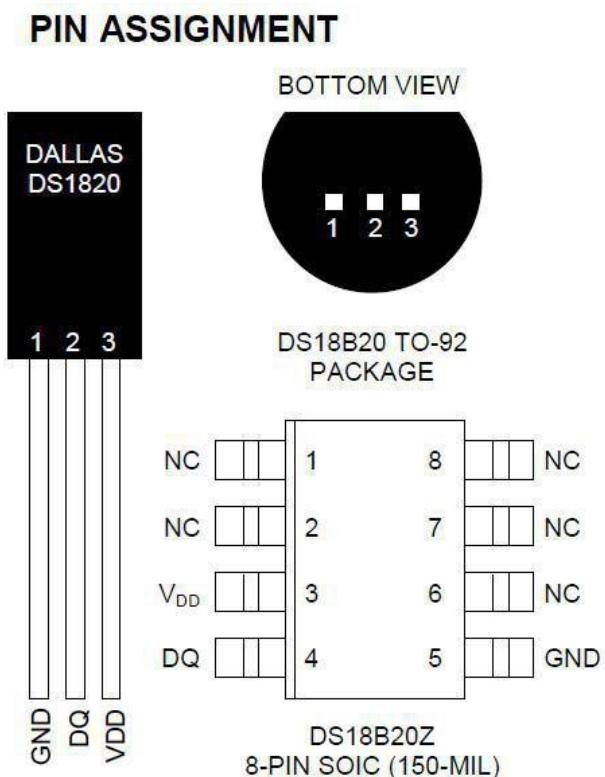
1.OUTPUT
2.VCC: 3.3V-5V DC
3.GND:ground

DS18B20 TEMP SENSORMODULE

Component Required:

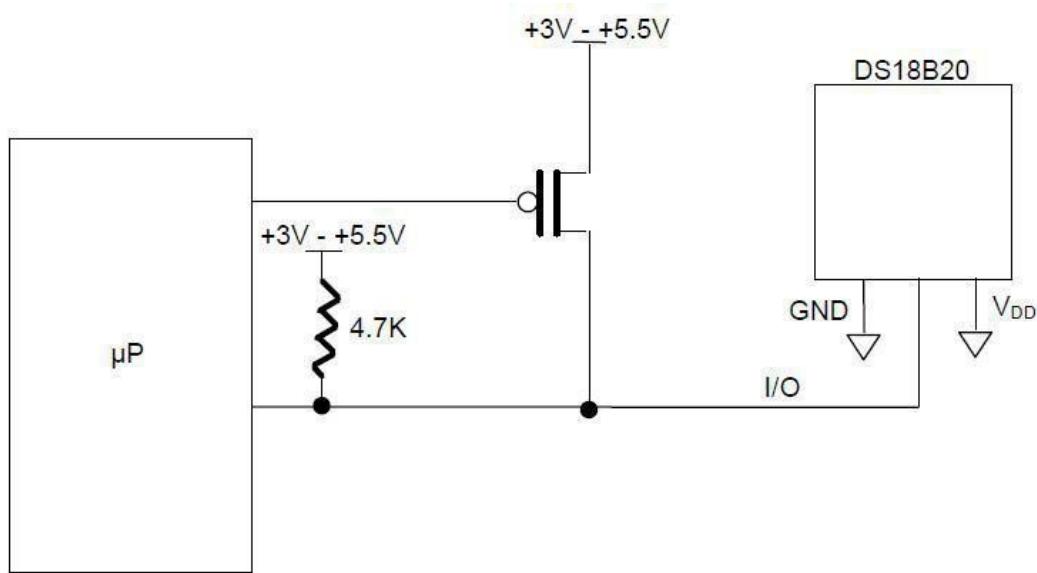
- 1 x Elegoo Uno R3
- 1 x USB cable
- 1 x DS18B20 module
- 3 x F-M wires

Component Introduction DS18B20:



PIN DESCRIPTION

- GND - Ground
- DQ - Data In/Out
- V_{DD} - Power Supply Voltage
- NC - No Connect



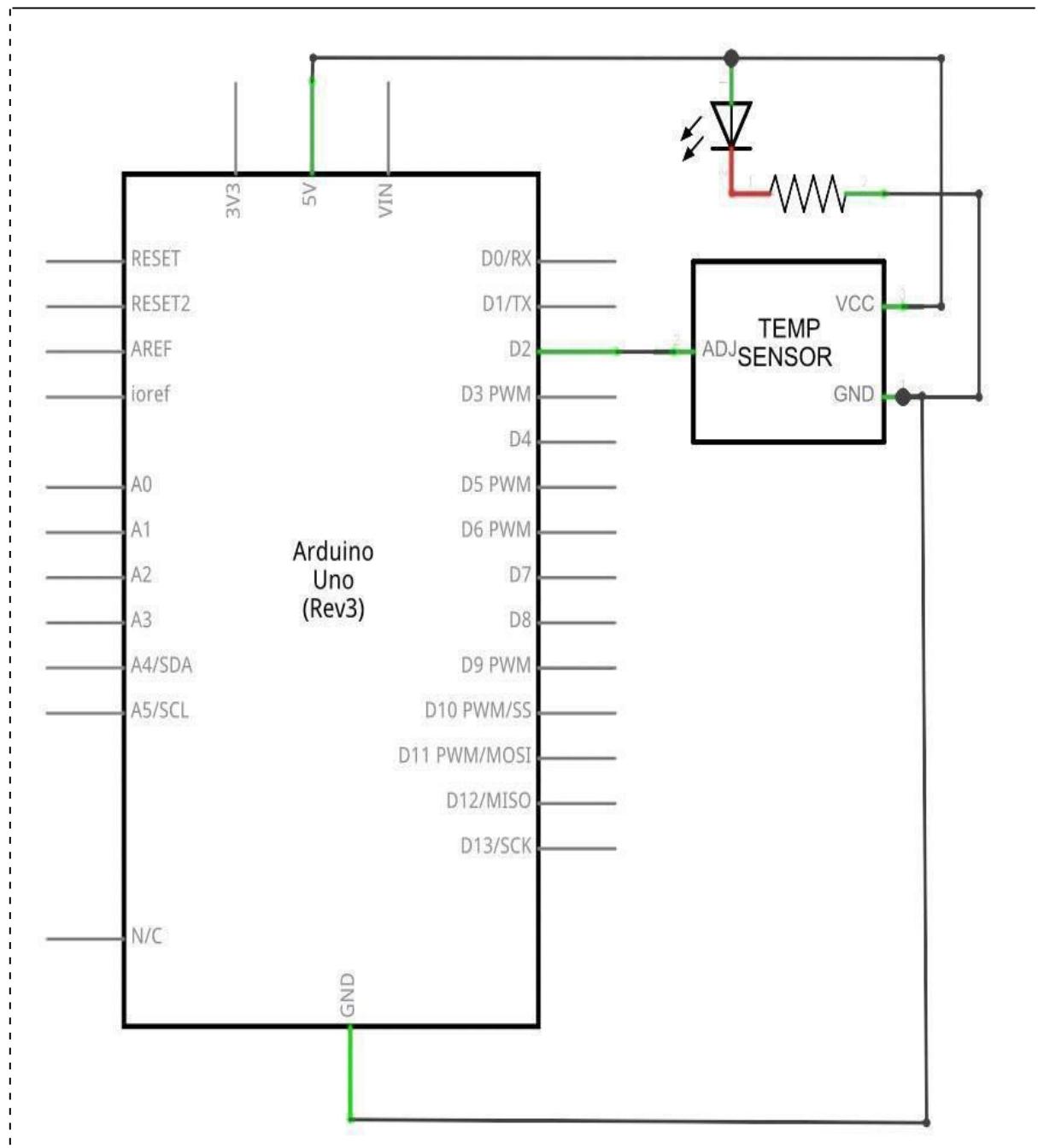
Principle

DS18B20 module uses a single bus. The power supply voltage range of 3.0 V to 5.5 V and no standby power supply. It can Measure temperature range from-55 degree to +125 degree with accuracy of $\pm 0.5^{\circ}\text{C}$. The temperature sensor features programmable DPI which can be set between 9 and 12. The temperature itself is represented by 12 bits and can be reported at a maximum rate of once every 750 milliseconds.

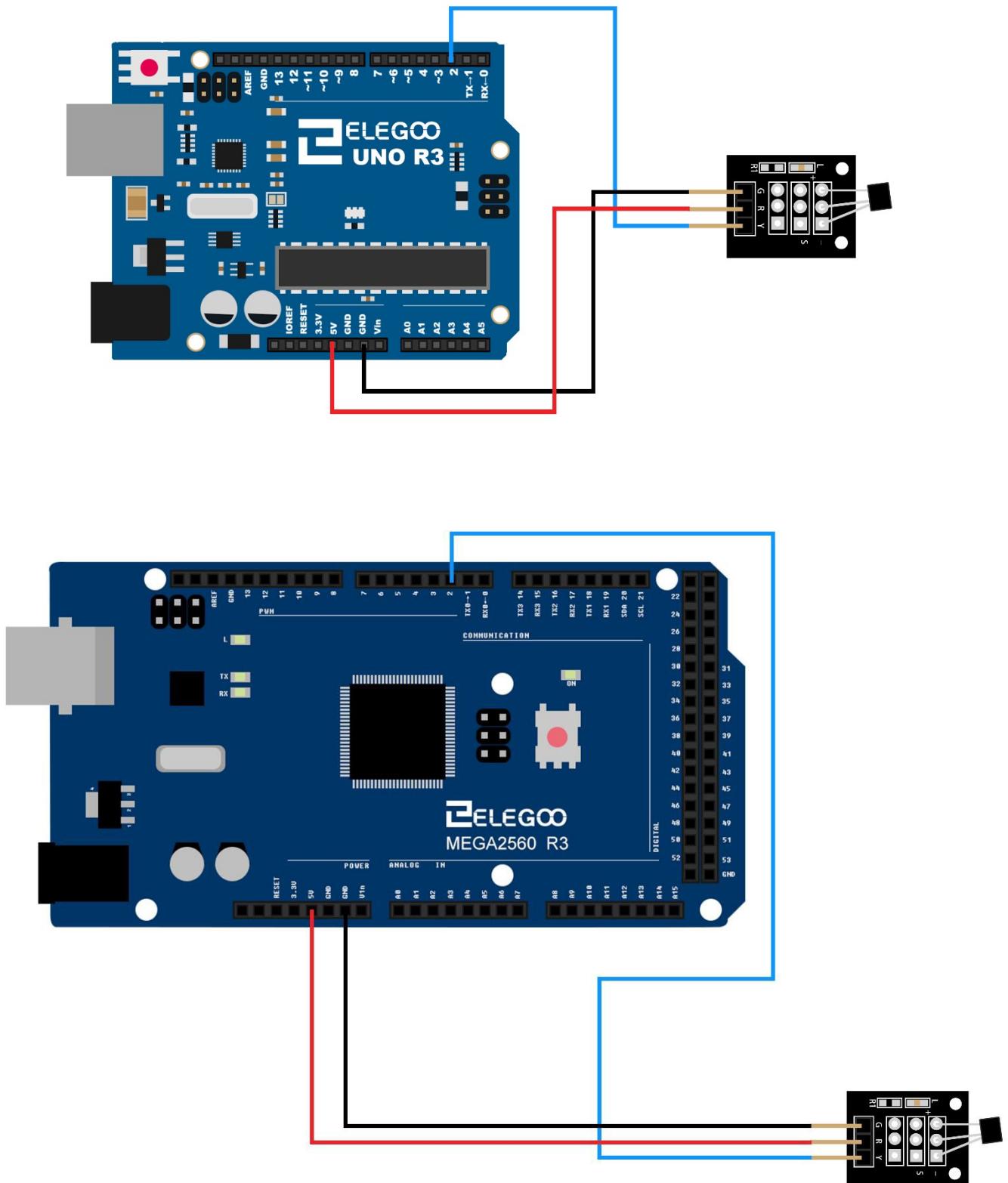
Each DS18B20 contains a unique serial number so that multiple DS18B20 chips can exist in a bus.

Connection

Schematic



Wiringdiagram



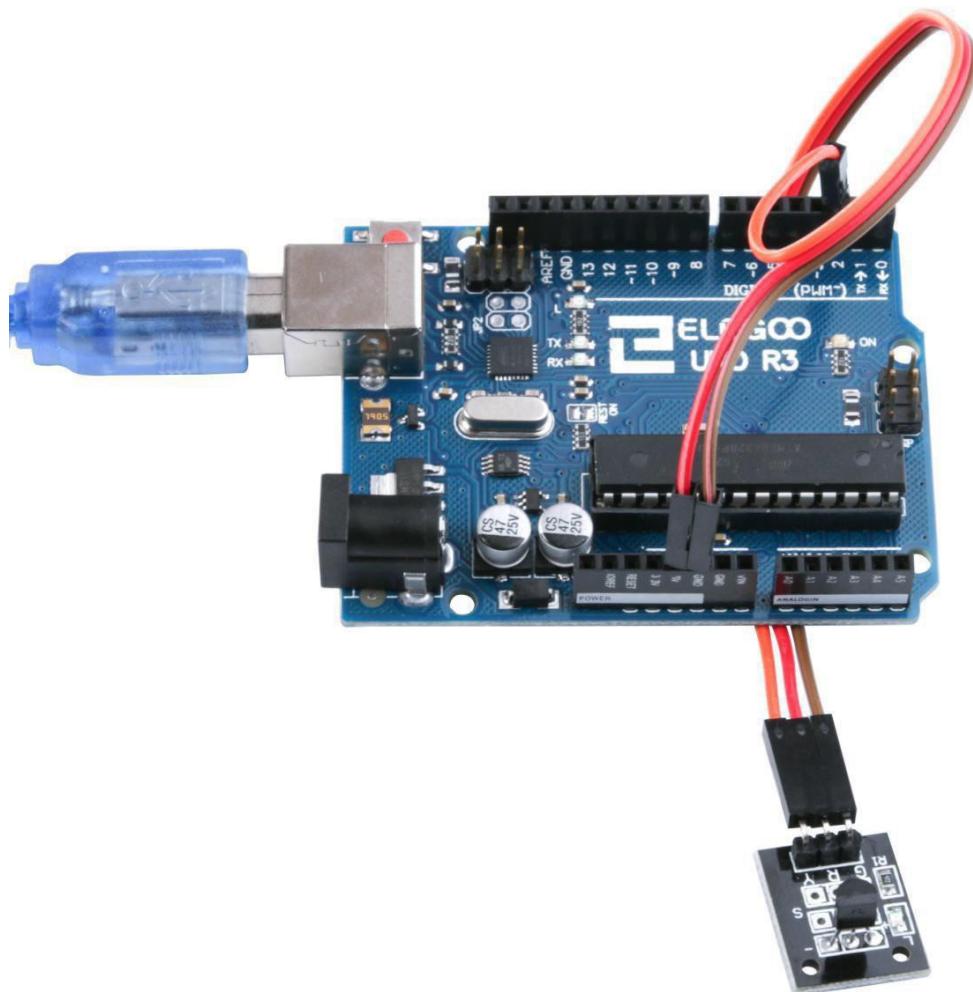
Code

After wiring, please open the program in the code folder- (Lesson 5 DS18B20 DIGITAL TEMPERATURE SENSOR 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 < Dallas Temperature> and < OneWire> library or re-install it, if necessary. Otherwise, your code won't work.

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

Result



Temperature sensor can detect temperature in numbers of different places at the same time. Upload the program then open the monitor, we can see the data asbelow:

```
Requesting temperatures... DONE
Temperature for the device 1 (index 0) is: 24.25
Requesting temperatures... DONE
Temperature for the device 1 (index 0) is: 24.25
Requesting temperatures... DONE
Temperature for the device 1 (index 0) is: 24.19
Requesting temperatures... DONE
Temperature for the device 1 (index 0) is: 24.19
Requesting temperatures... DONE
Temperature for the device 1 (index 0) is: 24.19
Requesting temperatures... DONE
Temperature for the device 1 (index 0) is: 24.12
Requesting temperatures... DONE
Temperature for the device 1 (index 0) is: 24.12
Requesting temperatures... DONE
Temperature for the device 1 (index 0) is: 24.12
Requesting temperatures... DONE
Temperature for the device 1 (index 0) is: 24.12
Requesting temperatures... 
```

The followings are the code used in this experiment and their explanations:

```
/* Include the libraries we need*/
#include <OneWire.h>
#include <DallasTemperature.h>

/* Data wire is plugged into port 2 on the Arduino*/
#define ONE_WIRE_BUS 2

/* Setup a oneWire instance to communicate with any OneWire devices (not just Maxim/Dallas
temperature ICs)*/
One Wire one Wire(ONE WIRE BUS);

/*Pass our oneWire reference to Dallas Temperature. */
Dallas Temperature sensors (&one Wire);

/*
 * The setup function. We only start the sensors here
 */
void setup (void)
{
    /* start serial port*/
    Serial. begin(9600);
    Serial. println("Dallas Temperature IC Control Library Demo");

    /*Start up the library*/
    Sensors .begin ();
}

/*
 * Main function, get and show the temperature
 */

```

```
void loop(void)
{
/* call sensors. Request Temperatures () to issue a global temperature */
/*request to all devices on the bus*/
Serial.print("Requesting temperatures...");
sensors.requestTemperatures(); /*Send the command to get temperatures*/
Serial.println("DONE");
/* After we got the temperatures, we can print them here.*/
/*We use the function ByIndex, and as an example get the temperature from the first sensor only.*/
Serial.print("Temperature for the device 1 (index 0) is: ");
Serial.println(sensors.getTempCByIndex(0));
}
```

From the above program we learn about programming serial communication syntax content:

(1) begin ()

Description

Sets the data rate in bits per second (baud) for serial data transmission. For communicating with the computer, use one of these rates: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, or 115200. You can, however, specify other rates - for example, to communicate over pins 0 and 1 with a component that requires a particular baud rate.

An optional second argument configures the data, parity, and stop bits. The default is 8 data bits, no parity, one stop bit.

Syntax

Serial.begin(speed)

Serial.begin(speed, config)

Arduino Mega only:

Serial1.begin(speed)

Serial2.begin(speed)

Serial3.begin(speed)

Serial1.begin(speed, config)

`Serial2.begin(speed, config)`

`Serial3.begin(speed, config)`

Parameters

`speed`: in bits per second (baud) - long

`config`: sets data, parity, and stop bits. Valid values are:

`SERIAL_5N1`

`SERIAL_6N1`

`SERIAL_7N1`

`SERIAL_8N1` (the default)

`SERIAL_5N2`

`SERIAL_6N2`

`SERIAL_7N2`

`SERIAL_8N2`

`SERIAL_5E1`

`SERIAL_6E1`

`SERIAL_7E1`

`SERIAL_8E1`

`SERIAL_5E2`

`SERIAL_6E2`

`SERIAL_7E2`

`SERIAL_8E2`

`SERIAL_5O1`

`SERIAL_6O1`

`SERIAL_7O1`

`SERIAL_8O1`

`SERIAL_5O2`

`SERIAL_6O2`

`SERIAL_7O2`

`SERIAL_8O2`

Returns

nothing

Example:

```
void setup() {
```

```
    Serial.begin(9600); // opens serial port, sets data rate to 9600 bps
```

```
}
```

```
void loop() {}
```

[Get Code]

Arduino Mega example:

```
// Arduino Mega using all four of its Serial ports
```

```
// (Serial, Serial1, Serial2, Serial3),
```

```
// with different baud rates:
```

```
void setup(){
```

```
    Serial.begin(9600);
```

```
    Serial1.begin(38400);
```

```
    Serial2.begin(19200);
```

```
    Serial3.begin(4800);
```

```
    Serial.println("Hello Computer");
```

```
    Serial1.println("Hello Serial 1");
```

```
    Serial2.println("Hello Serial 2");
```

```
    Serial3.println("Hello Serial 3");
```

```
}
```

```
void loop() {}
```

(2) **println()**

Prints data to the serial port as human-readable ASCII text followed by a carriage return character (ASCII 13, or '\r') and a newline character (ASCII 10, or '\n'). This command takes the same forms as Serial.print().

Syntax

```
Serial.println(val)
```

```
Serial.println(val, format)
```

Parameters

val: the value to print - any data type

format: specifies the number base (for integral data types) or number of decimal places (for floating point types)

Returns

size_t (long): println() returns the number of bytes written, though reading that number is optional

Example:

```
/*
Analog input
```

reads an analog input on analog in 0, prints the value out.

```
*/
```

```
int analogValue = 0; // variable to hold the analog value
void setup() {
    // open the serial port at 9600 bps:
    Serial.begin(9600);
}
void loop() {
```

```
// read the analog input on pin 0:  
analog Value = analog Read(0);  
  
// print it out in many formats:  
Serial.println(analogValue);      // print as an ASCII-encoded decimal  
Serial.println(analogValue, DEC); // print as an ASCII-encoded decimal  
Serial.println(analogValue, HEX); // print as an ASCII-encoded hexadecimal  
Serial.println(analogValue, OCT); // print as an ASCII-encoded octal  
  
Serial.println (analog Value, BIN); // print as an ASCII-encoded binary  
// delay 10 milliseconds before the next reading:  
delay(10);  
}
```

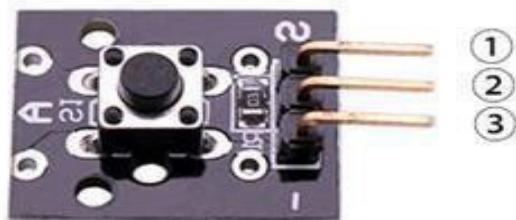
Lesson 6 Button Switch Module

Overview

In this experiment, we will learn how to use button switch.

Button Switch module

A built-in 10 K ohm resistor is connected between the center pin and the 'S' pin and can be used as a pull up or pull down resistor. The push button connects the two outer pins.



- 1.OUTPUT
- 2.VCC: 3.3V-5V DC
- 3.GND:ground

Component Required:

1x Elegoo Uno R3

1x USB cable

1x Button module

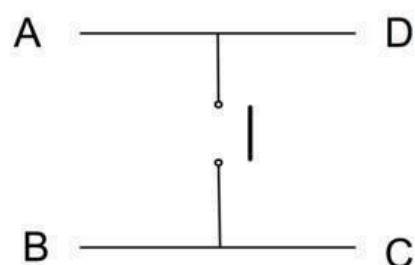
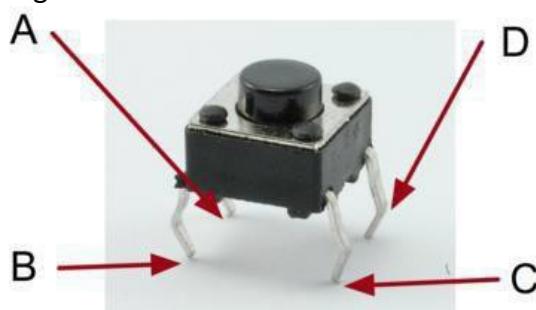
3x F-M wires

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.



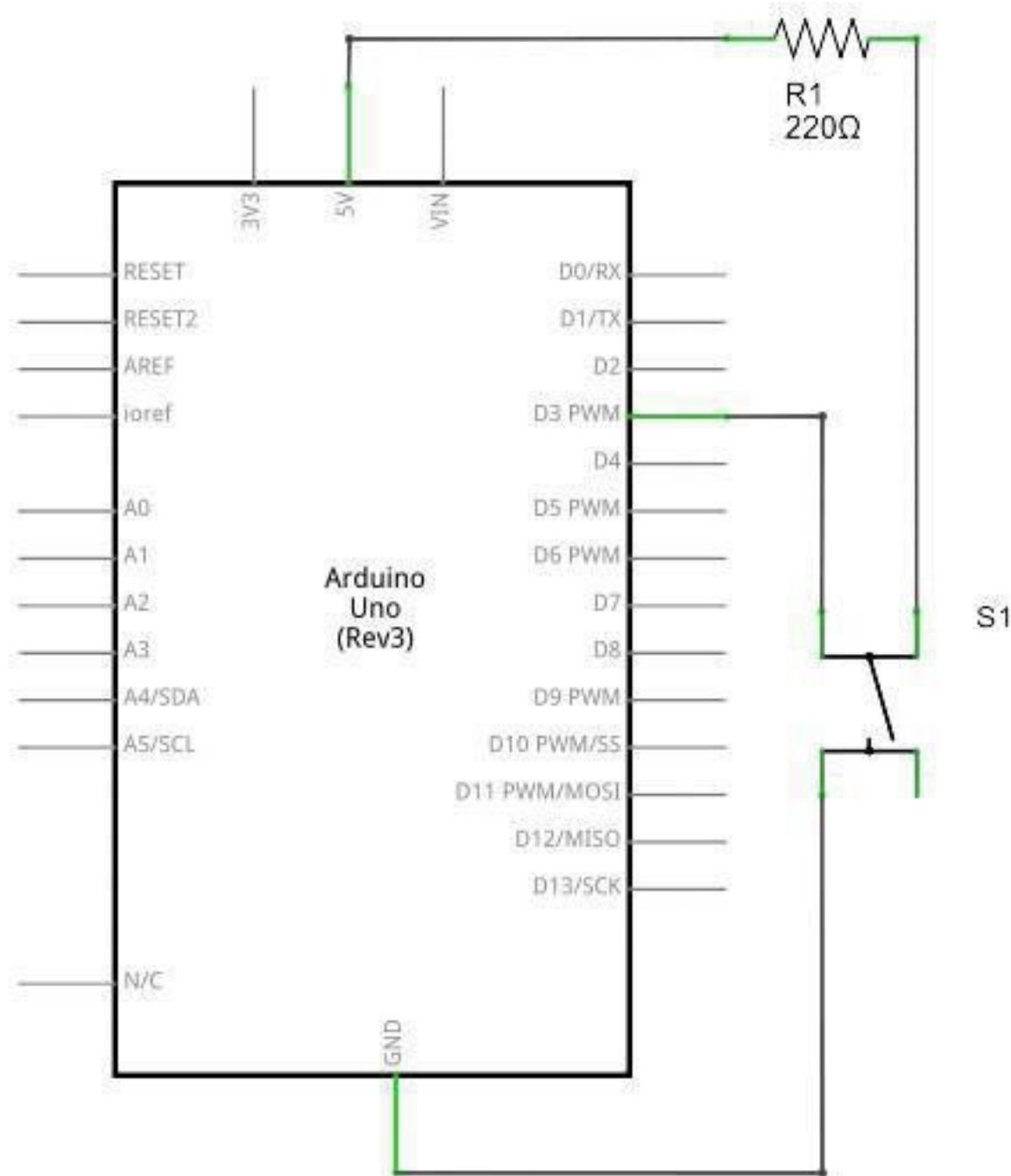
Actually, there are only really two electrical connections, as inside the switch package pins B and C are connected together, as are A and D.

Principle

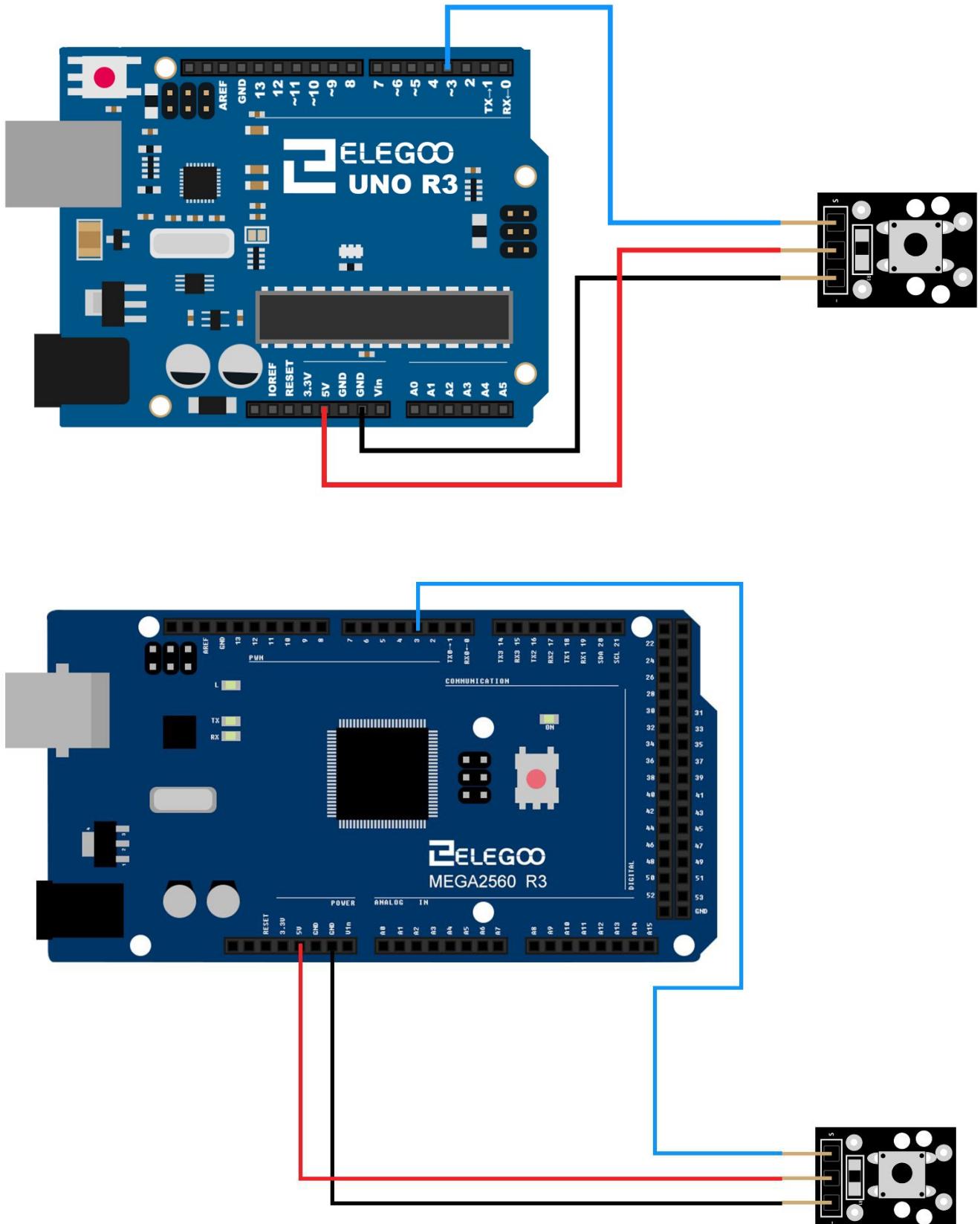
Button Switch and number 13 port have the built-in LED simple circuit. To produce a switch flasher, we can use connect the digital port 13 to the built-in LED and connect the Button Switch port to number 3 port of Elegoo Uno board. When the switch sensing, LED twinkle light to the switch signal.

Connection

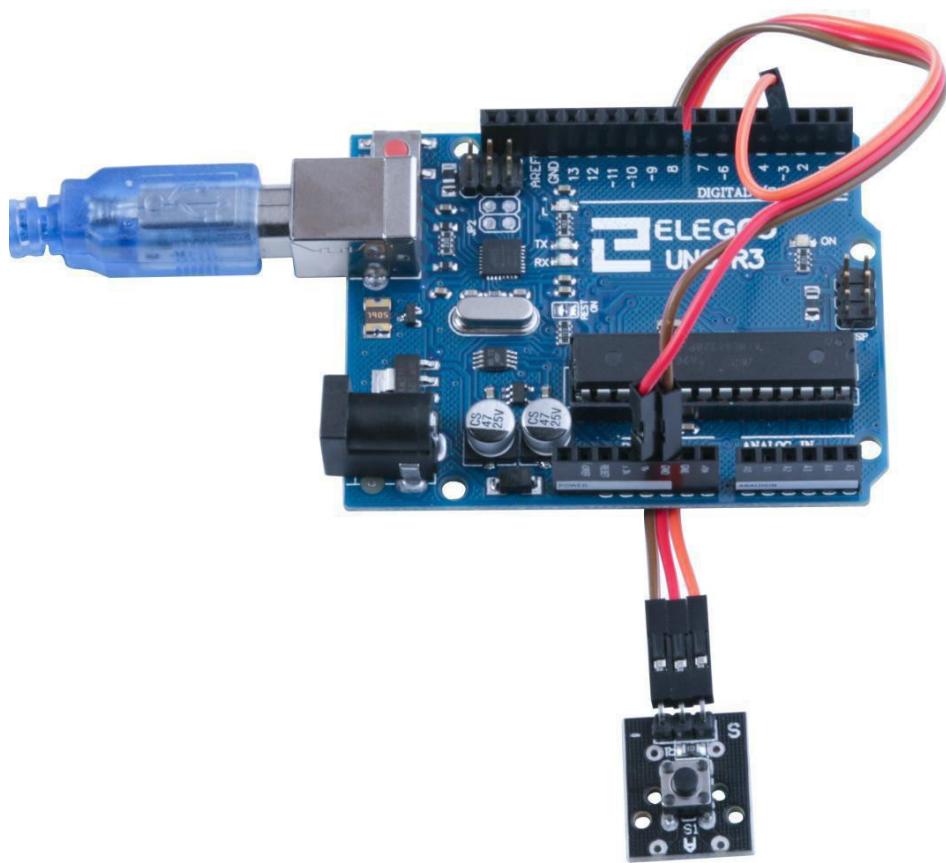
Schematic



Wiring diagram



Example picture



Code

After wiring, please open the program in the code folder- (Lesson 6 BUTTON SWITCH MODULE) and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors .Then push the button, you can see led on and off.

The followings are the code used in this experiment and their explanations:

```
//define LED port
int Led = 13;
//define shock port
int Shock = 3;
//define digital variable val
int val;
void setup()
{
//define LED as a output port
PinMode (Led, OUTPUT);
//define shock sensor as a output port

pinMode(Shock, INPUT);
}
void loop()
{
//read the value of the digital interface 3 assigned to val

val = digital Read(Shock);
//when the shock sensor have signal, LED blink

if (val == HIGH)
{
  digitalWrite(Led, LOW);
}
else
{
  digitalWrite(Led, HIGH);
}
}
```

From the above program we learn to program control structure content:

(1) if/else

if/else allows greater control over the flow of code than the basic if statement, by allowing multiple tests to be grouped together. For example, an analog input could be tested and one action taken if the input was less than 500, and another action taken if the input was 500 or greater. The code would look like this:

```
if (pinFiveInput < 500)
```

```
{ // action A }
```

```
else
```

```
{ // action B }
```

else can proceed another if test, so that multiple, mutually exclusive tests can be run at the same time:

```
if (pinFiveInput < 500)
```

```
{ // do Thing A }
```

```
else if (pinFiveInput >= 1000)
```

```
{ // do Thing B }
```

```
else
```

```
{ // do Thing C }
```

You can have an unlimited number of such branches. (Another way to express branching, mutually exclusive tests is with the switch case statement.)

Coding Note: If you are using if/else, and you want to make sure that some default action is always taken, it is a good idea to end your tests with an else statement set to your desired default behavior.

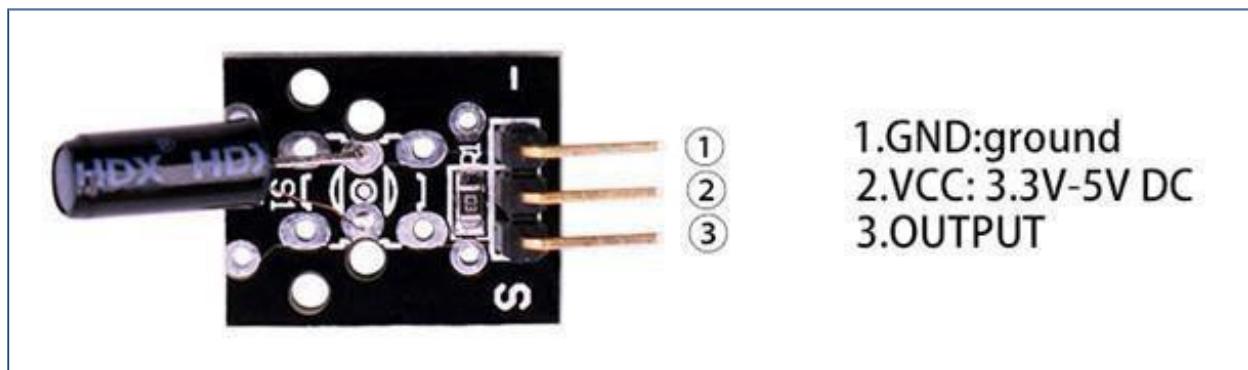
Lesson 7 Switch Modules

Overview

In this lesson, we will learn how to use switch modules. Including Shock switch, Tilt switch and Tap switch module.

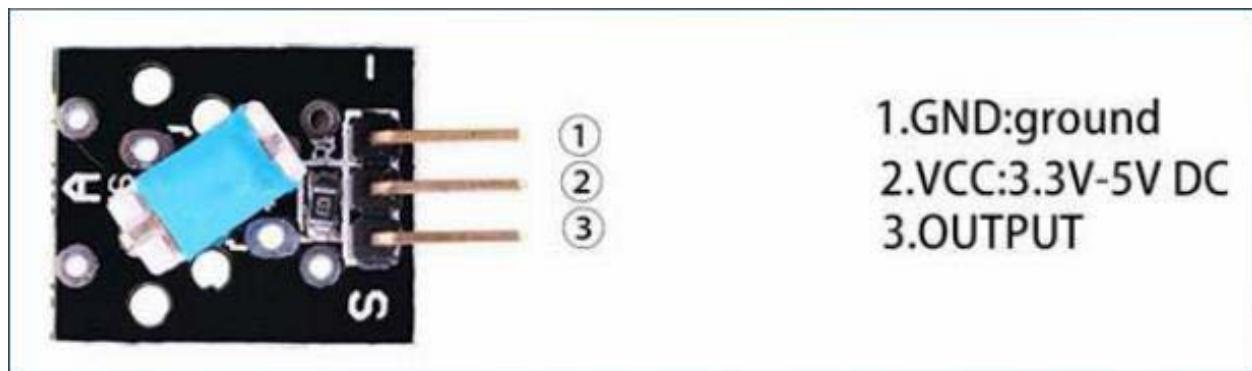
Shock switch module

A built-in 10 K ohm resistor is connected between the center pin and the 'S' pin and can be used as a pull-up or pull-down resistor. The switch contacts connect to the two outer pins.



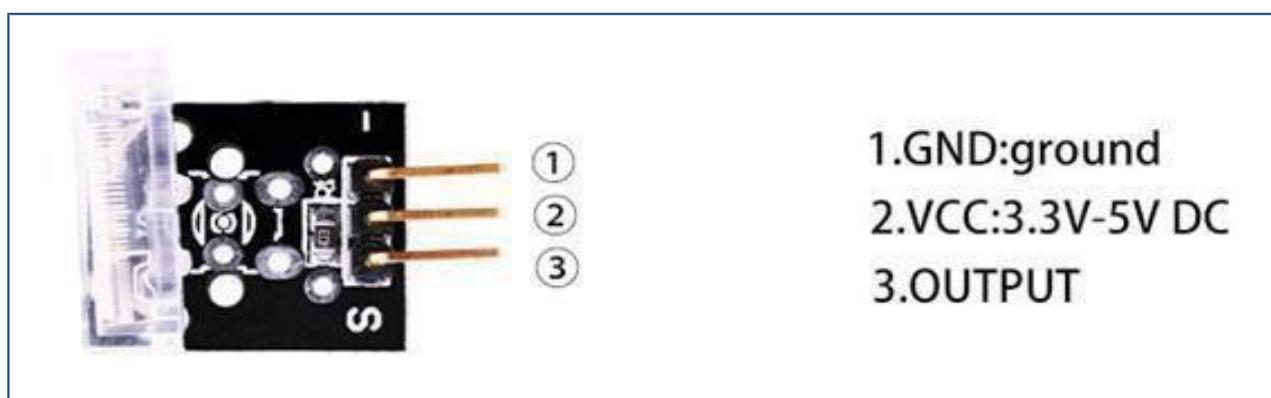
Tilt switch module

A built-in 10 K Ohm resistor connected between the middle and 'S' pin is available for pull up or pull down use. The switch contacts connect to the two outer pins. Load switching max: 12VDC 50mA.



Tap switch module

Also called the vibration sensor module. The momentary switch contacts are connected between the two outer pins.



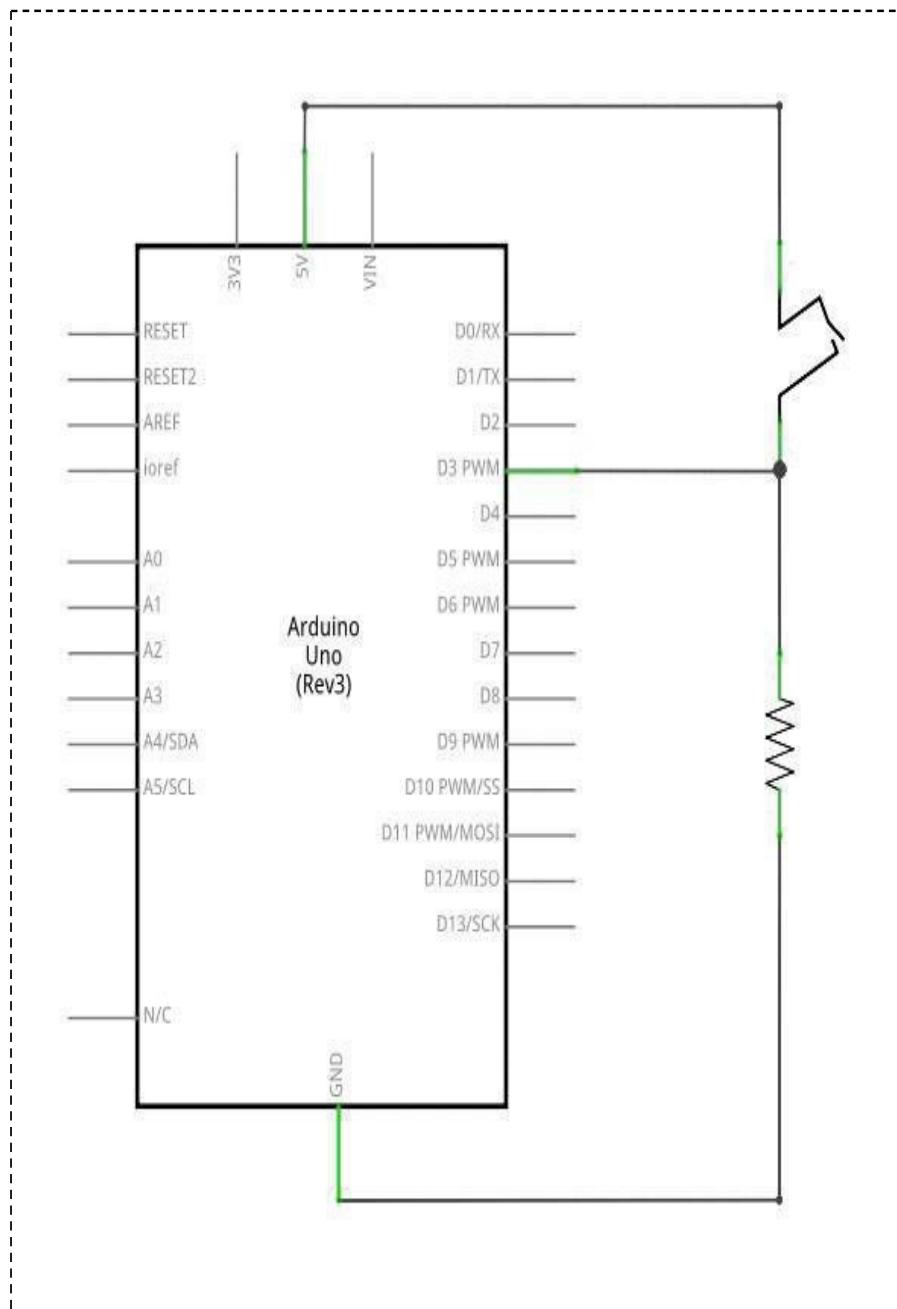
Component Required:

- 1 x Elegoo Uno R3
- 1 x USB cable
- 1 x Shock switch module
- 1 x Tilt switch module
- 1 x Tap switch module
- 3 x F-M wires

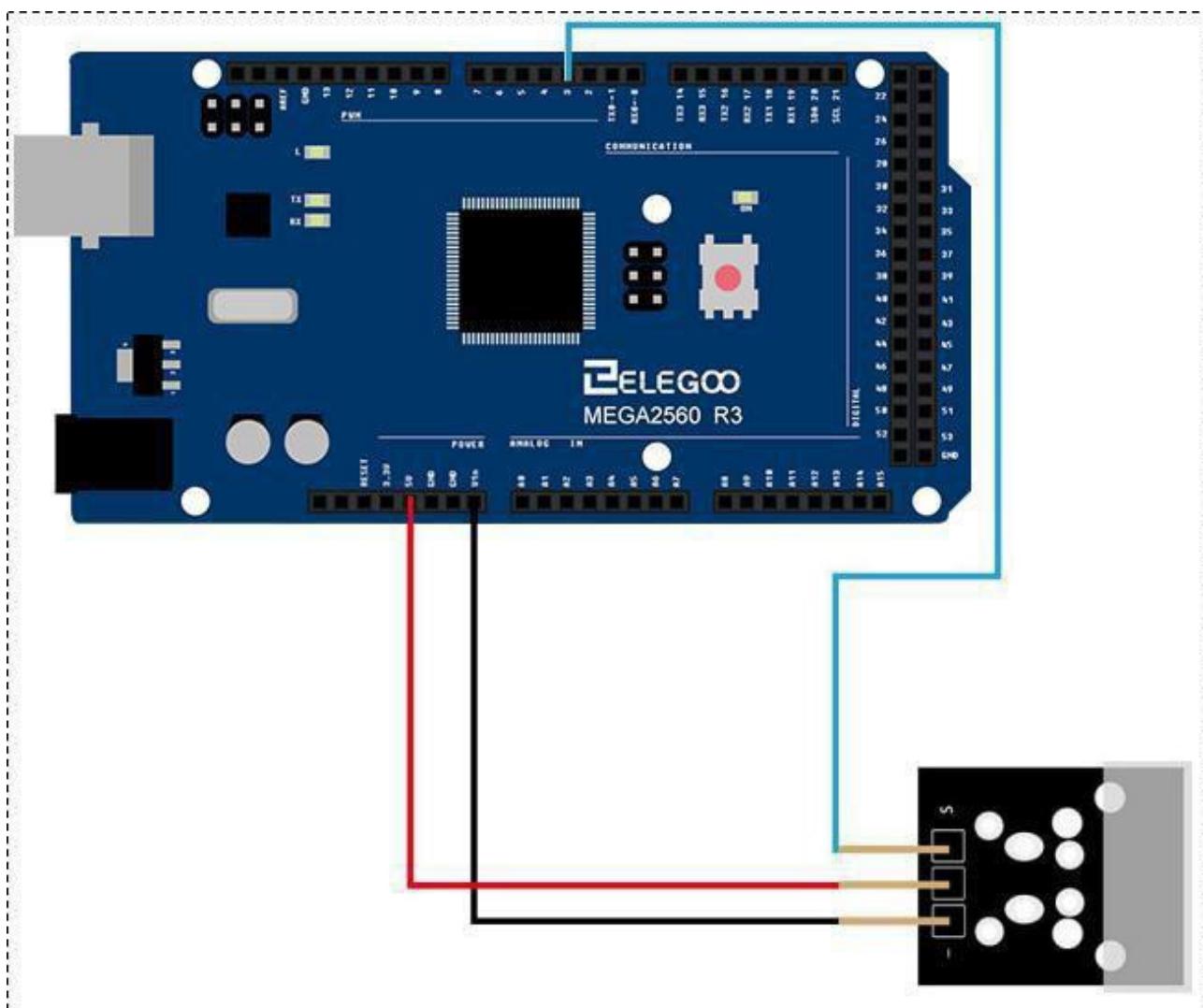
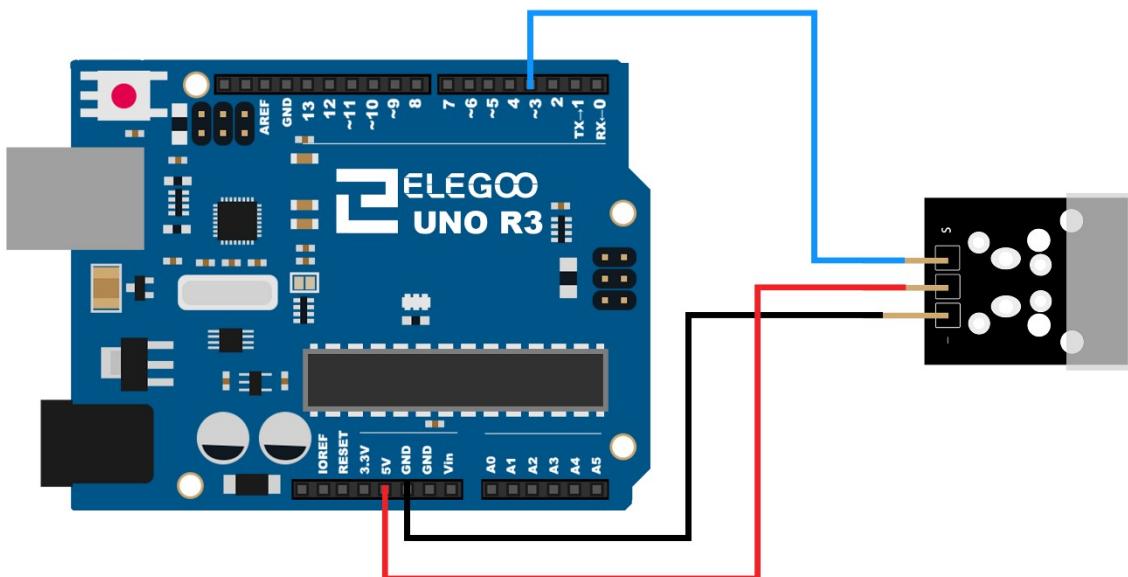
Principle

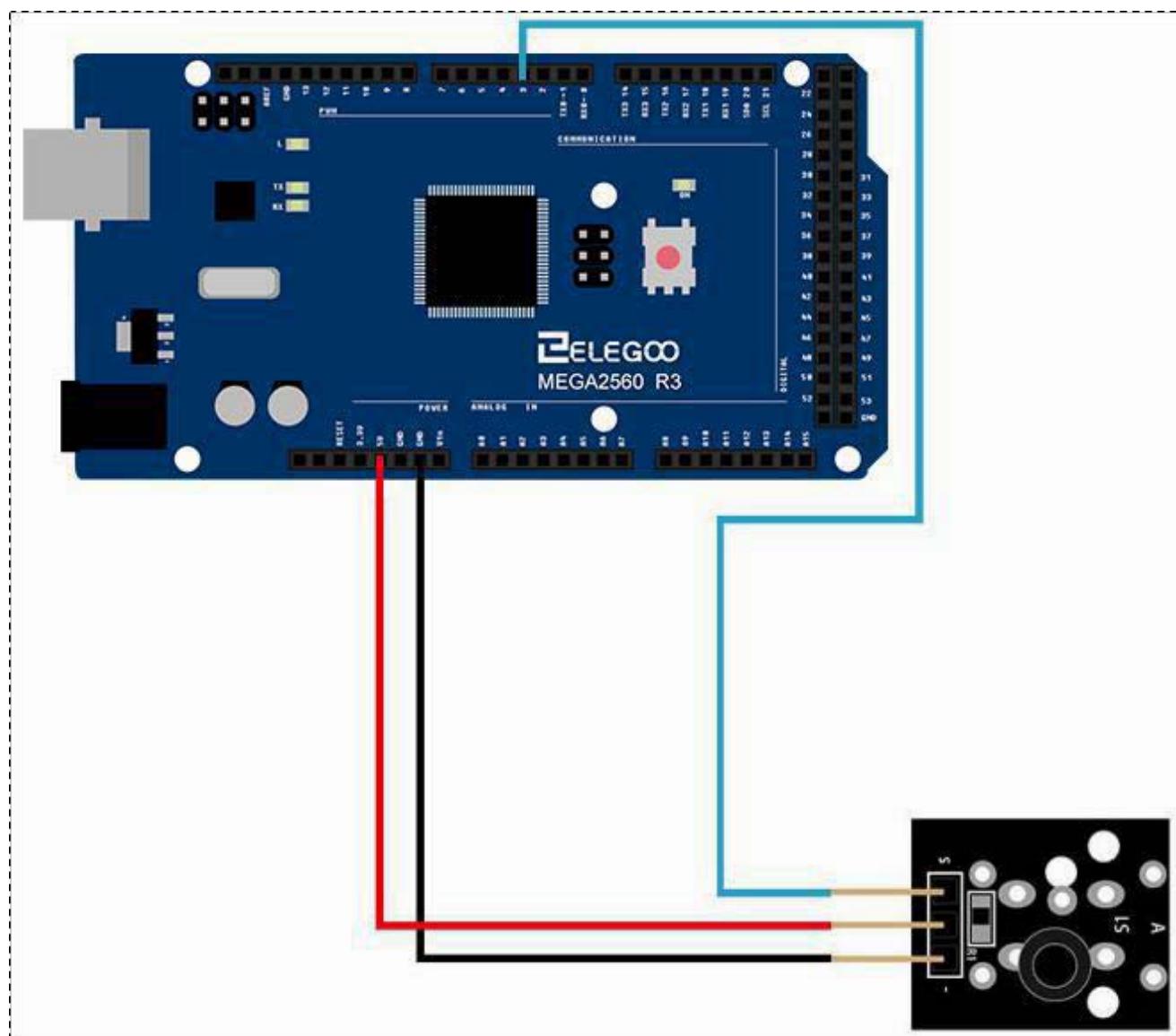
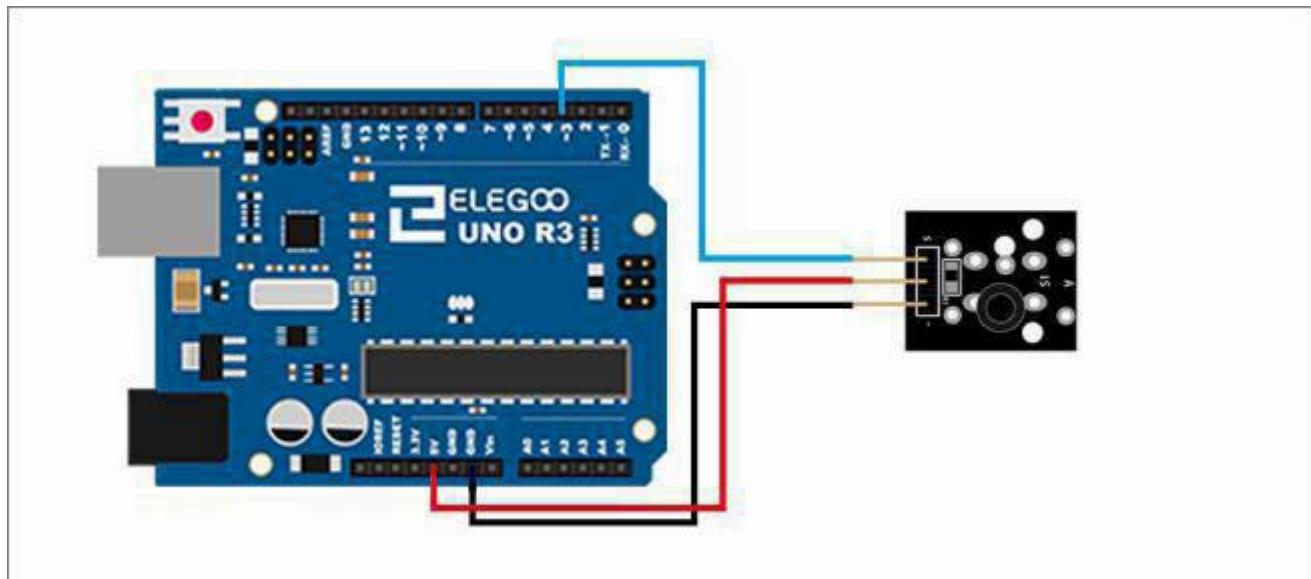
The Switch module and digital 13 port on the UNO board (or Mega 2560 board) have built-in LED circuits. Therefore to make a SWITCH flasher, you will only need to connect the "S" pin on the switch module to digital 3 port of Elegoo Uno board ((or Mega 2560 board). After connected, the module sends signal, and when the switch is sensing, and the LED will flash.

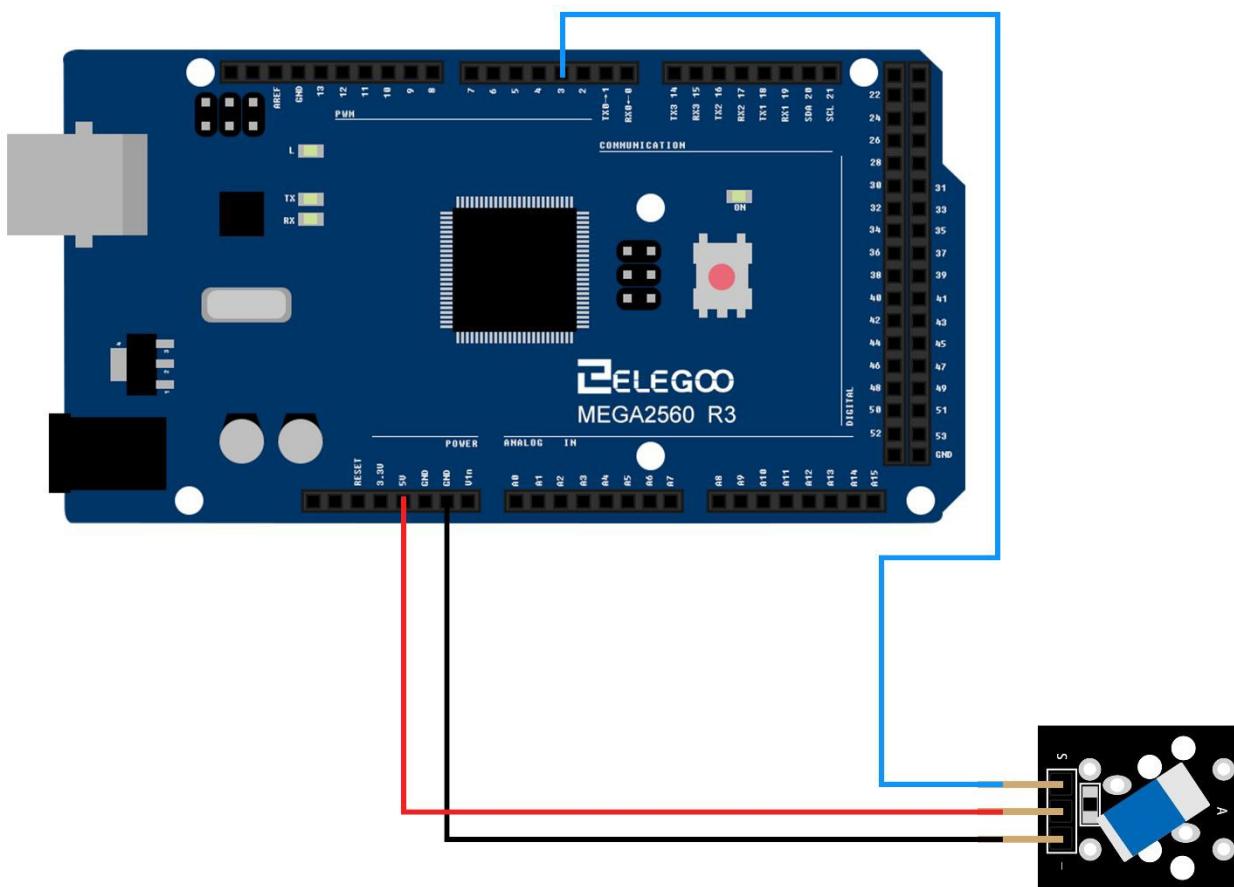
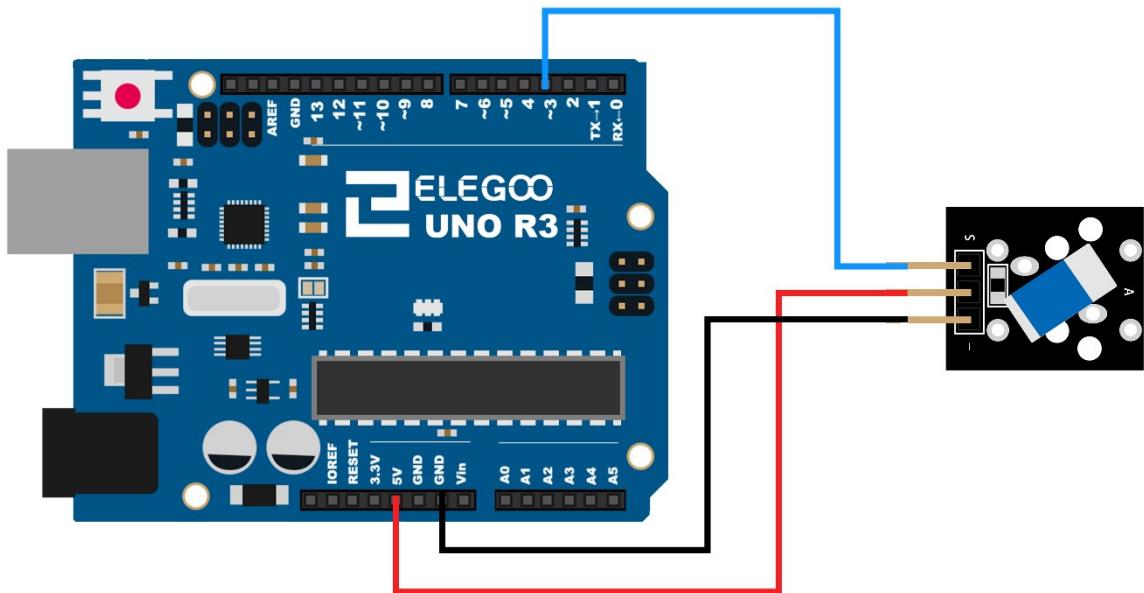
Connection Schematic



wiringdiagram



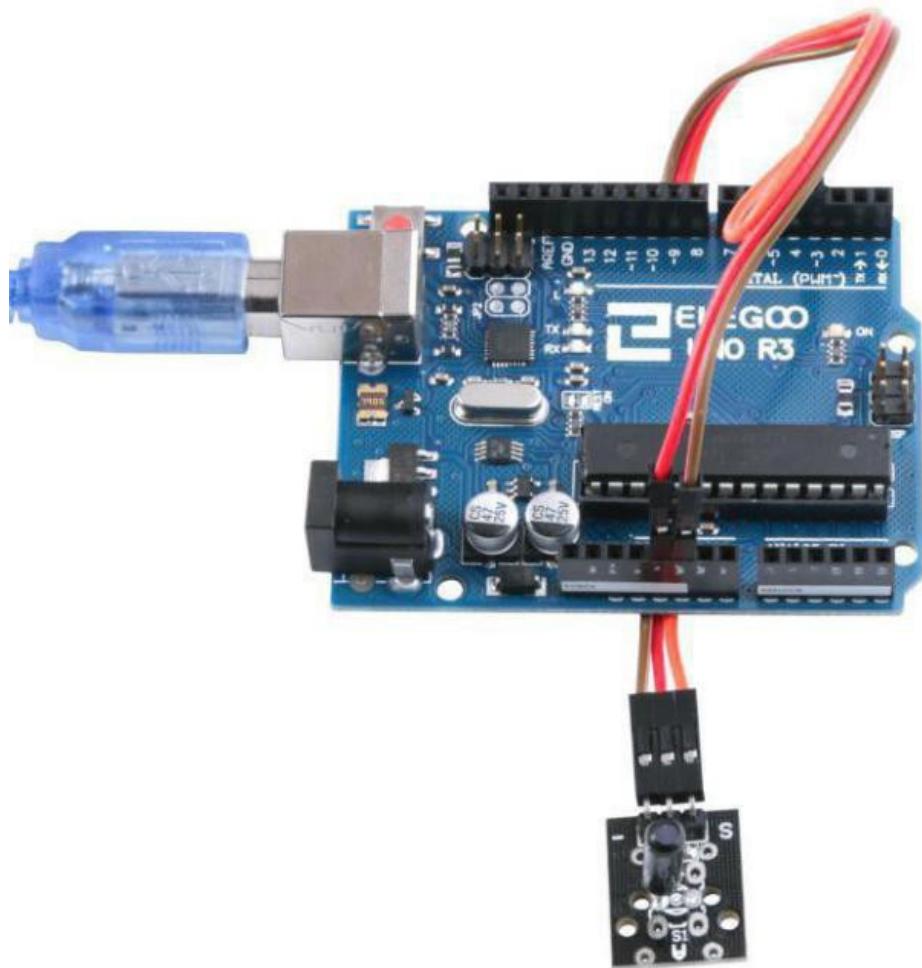


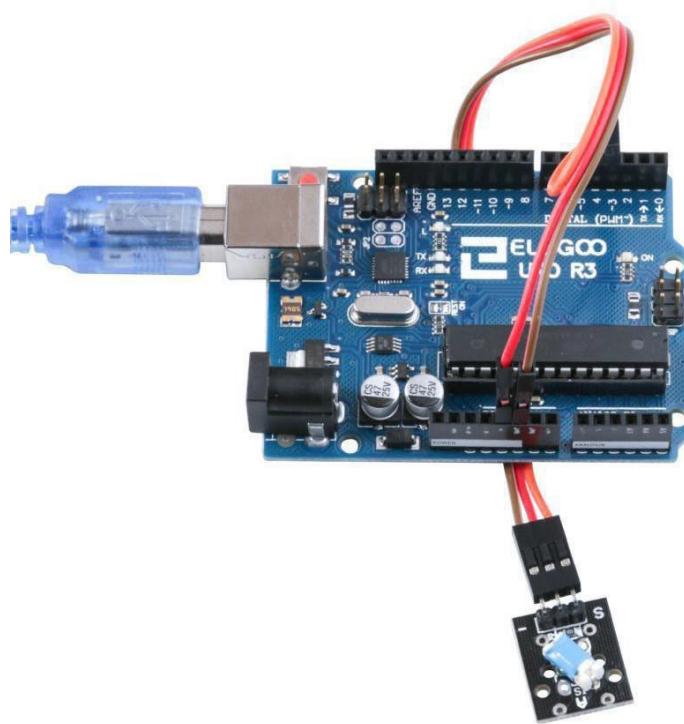
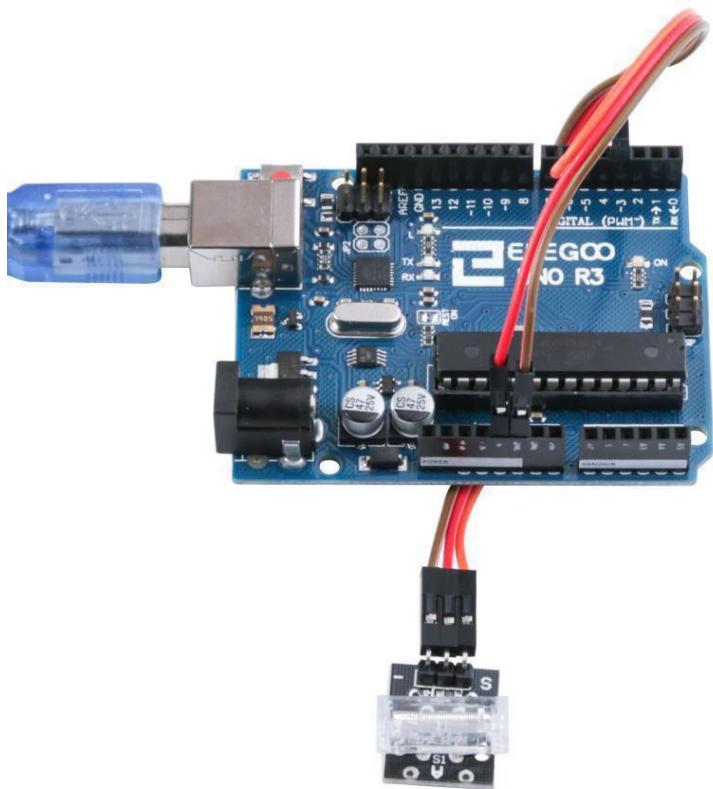


Code

After wiring, please open the program in the code folder- (Lesson 7 Each type of vibration switch) and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors. Then lean or shock the sensor,you can see led on and off

Example picture





The followings are the code used in this experiment and their explanations:

```
//define LED port
int Led=13;
//define switch port
int buttonpin=3;
//define digital variable val
int    val;
void   setup()
{
//define LED as a output port
pinMode(Led,OUTPUT);

//define switch as a output port
pinMode(buttonpin,INPUT);
void   loop()
{
//read the value of the digital interface 3 assigned to val

val=digitalRead(buttonpin);
//when the switch sensor have signal, LED blink
if(val==HIGH)
{
/*You can change the state of the control light by LOW or HIGH*/
digitalWrite (Led,HIGH);
}
else
{
digitalWrite (Led,LOW);
}
}
```

From the above program we learn to program control structure content:

(1) if/else

if/else allows greater control over the flow of code than the basic if statement, by allowing multiple tests to be grouped together. For example, an analog input could be tested and one action taken if the input was less than 500, and another action taken if the input was 500 or greater. The code would look like this: if (pinFiveInput < 500)

```
{ // action A }
```

```
else
```

```
{ // action B }
```

else can proceed another if test, so that multiple, mutually exclusive tests can be run at the same time:

```
if (pinFiveInput < 500)
```

```
{ // do Thing A }
```

```
else if (pinFiveInput >= 1000)
```

```
{ // do Thing B }
```

```
else
```

```
{ // do Thing C }
```

You can have an unlimited number of such branches. (Another way to express branching, mutually exclusive tests is with the switch case statement.)

Coding Note: If you are using if/else, and you want to make sure that some default action is always taken, it is a good idea to end your tests with an else statement set to your desired default behavior.

Lesson 8 IR Receiver and IR Transmitter Module

Overview

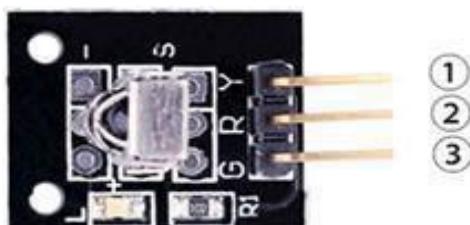
In this experiment, we will learn how to use Infrared Receiver and IR transmitter module.

In fact now in our daily life they play role in a lot of household electrical appliances are used to this kind of device, such as air conditioning, TV, DVD, etc. Actually it is based on its wireless remote sensing and it is very convenient by using them.

IR Receiver

Infrared sensor type 1838 for use with 38 KHz IR signals.

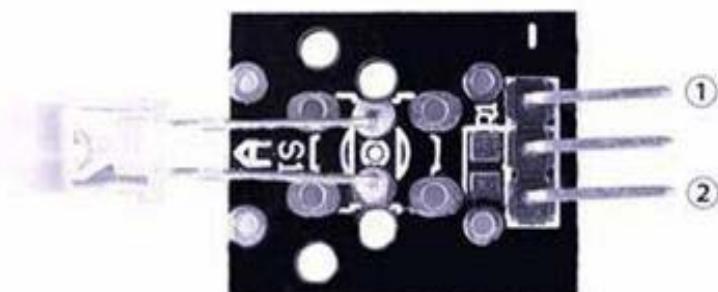
- Supply voltage: 2.7 to 5.5 V
- Frequency: 37.9 KHz
- Receiver range: 18m (typical)
- Receiving angle: 90°



1.OUTPUT
2.VCC:3.3V-5V DC
3.GND:ground

IR Transmitter module

The IR-LED can be used to build a light barrier or an IR remote control signal transmitter.



1.GND:ground
2.INPUT

Component Required:

2x Elegoo Uno R3

2x USB cable

1x IR Receiver module

1x IR Emission module

5 x F-M wires

Component Introduction

IR RECEIVER SENSOR:

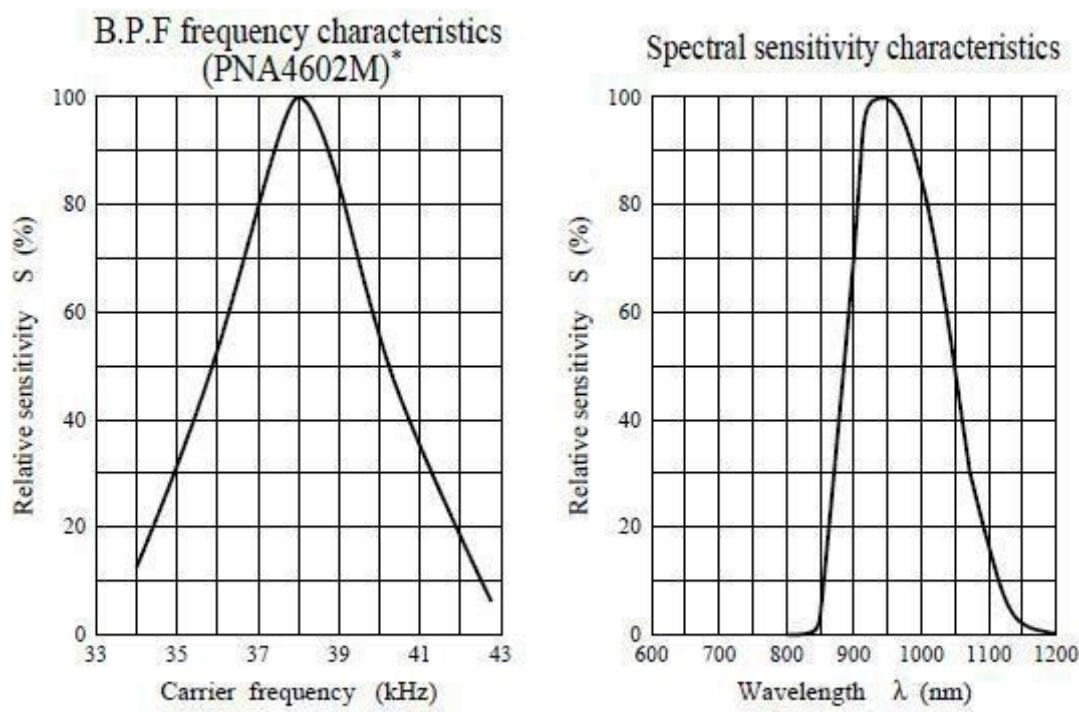
IR detectors are little microchips with a photocell that are tuned to listen to infrared light. They are almost always used for remote control detection - every TV and DVD player has one of these in the front to listen for the IR signal from the clicker. Inside the remote control is a matching IR LED, 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 differences between these and say a CdS Photocells:

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



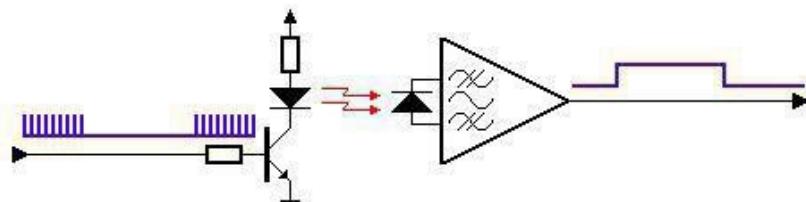
* The peaks for PNA4601M, PNA4608M, and PNA4610M are all f_0 .

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 wont detect as well from afar. Likewise, you can use 850 to 1100 nm LEDs but they wont work as well as 900 to 1000 nm so make sure to get matching LEDs! Check the datasheet for your IR LED to verify the wavelength. Try to get a 940 nm - remember that 940 nm is not visible light (its Infra Red)!

Principle

Firstly, let's know the structure of the infrared receiving head: there are two important elements inside the infrared receiving head, IC and PD. IC is receiving head processing components, mainly composed of silicon and circuit. It is a highly integrated device. The main function is filter, plastic, decoding, amplification, etc. PD is a photosensitive diode. The main function is to receive the light signal.

Below is a brief working principle diagram:



Infrared emitting diode launch out the modulation signal and infrared receiver head will receive, decode, filter and soon to regain the signal.

Infrared emitting diode: keep clean and in good condition. All the parameters in the process of working shall not exceed the limit value (positive To the current 30~60mA, positive pulse current 0.3~1A, reverse voltage 5V, dissipation power 90mW, working temperature range -25~+80 °C, and storage temperature range between 40~100 °C, the welding temperature 260°C) infrared to be with a closed head should be matching use, otherwise it will influence the sensitivity.

Connection

Code

After wiring, please open the program in the code folder- (Lesson 8 IR Receiver and IR Emission) 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 the tutorial on the loading of library file, see Lesson 1.

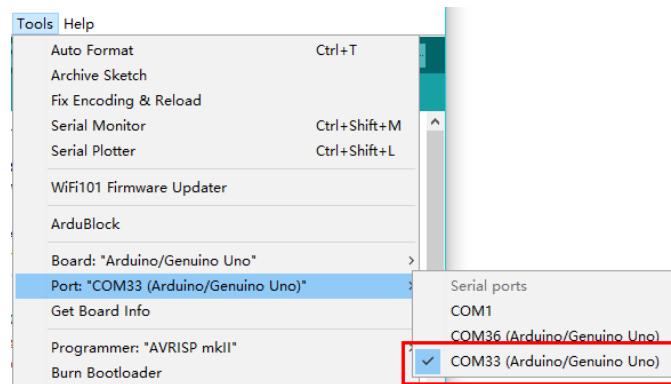
Firstly, take a UNO board and connect the Infrared Emission module according to the attached picture. Then follow the previous procedure to write the Infrared Emission program.

IR_Emission

Next, take another UNO board, connect the Infrared Receiver module as shown below and write IR Receiver program.

IR_Receiver

Note: Before downloading, be sure to reselect the download port. There will be two serial ports when you connect two UNO boards, we must choose the new one, as the picture shown.



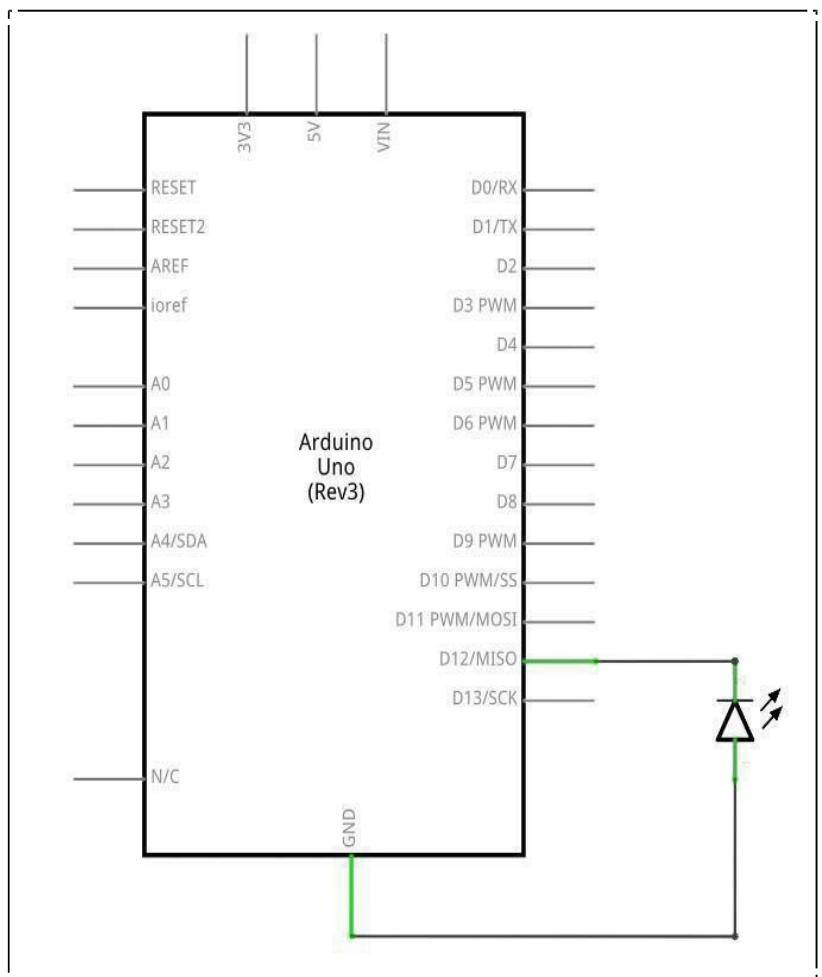
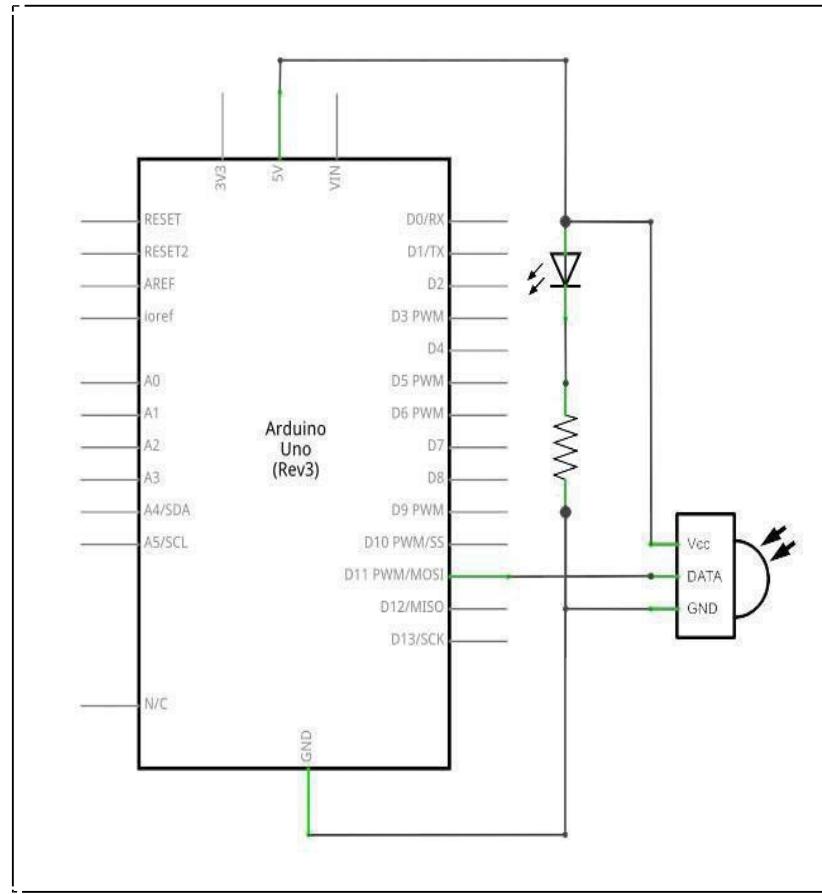
After wrote successfully, align the infrared emission module with the infrared receiver module.

Observe the UNO of the Infrared receiver, if you can see the L light flashes, it says the experiment is successful.

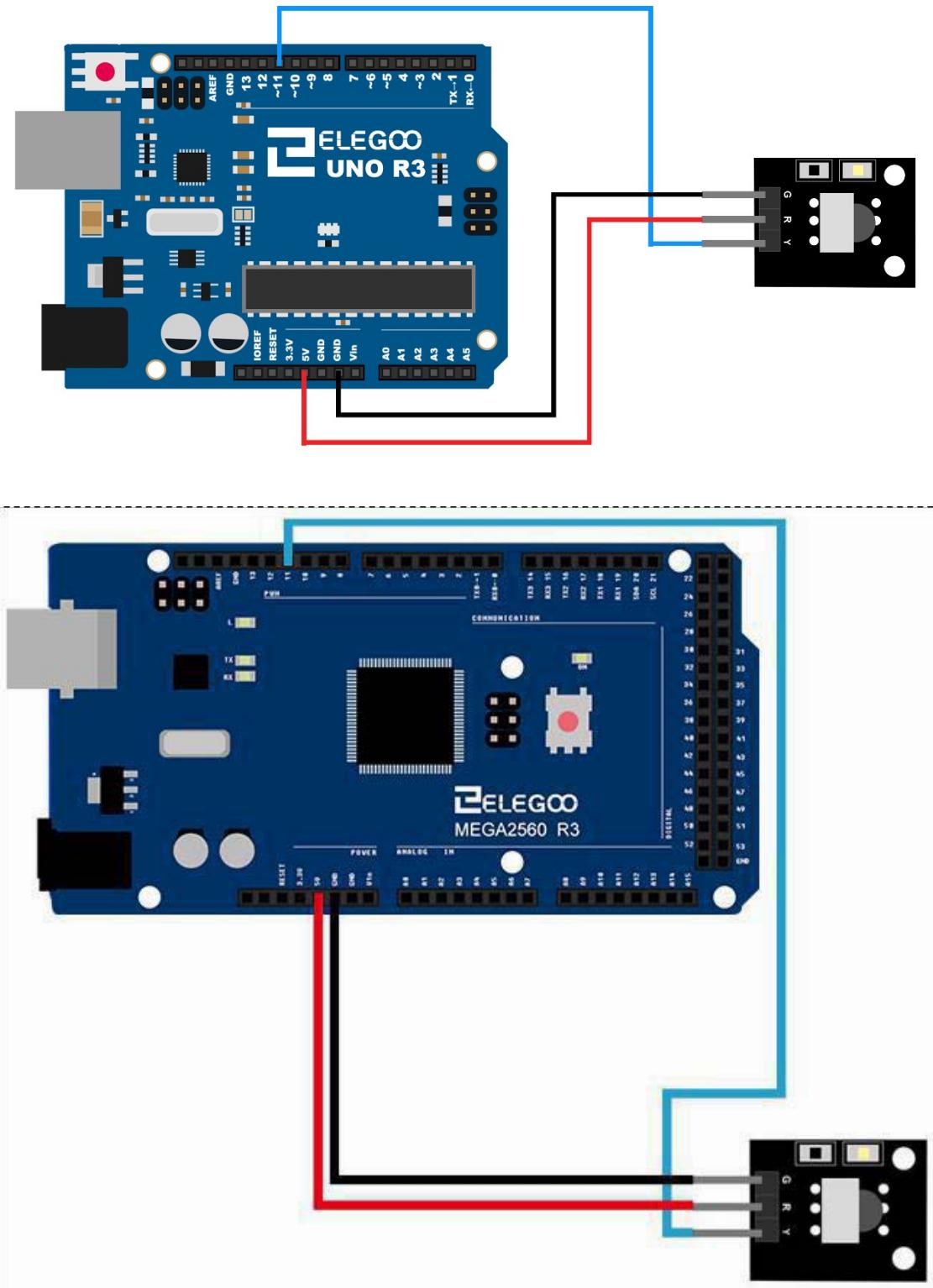
The principle of the experiment is that the infrared emission module transmits the specific coded signal. After the infrared receiver module received, the UNO will decode and operates the L lamp to blink.

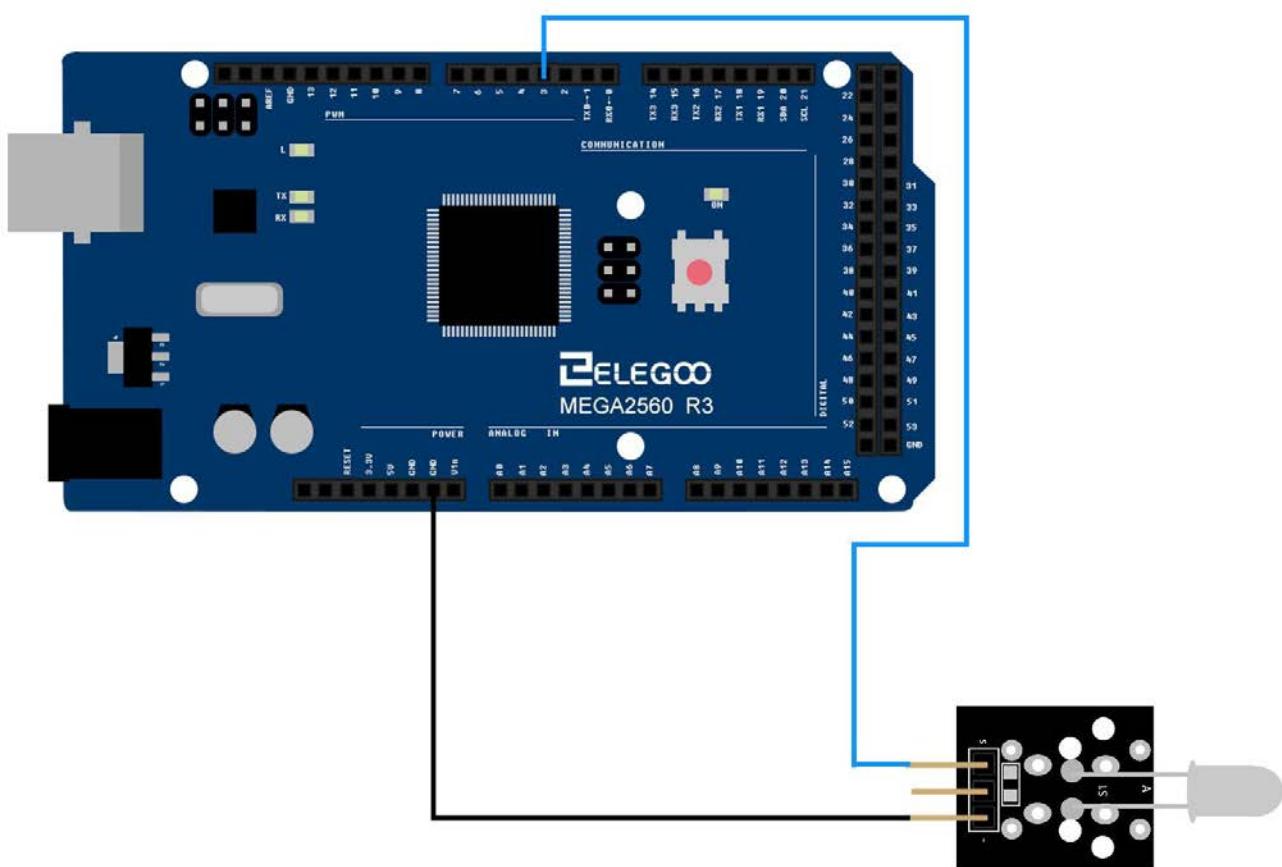
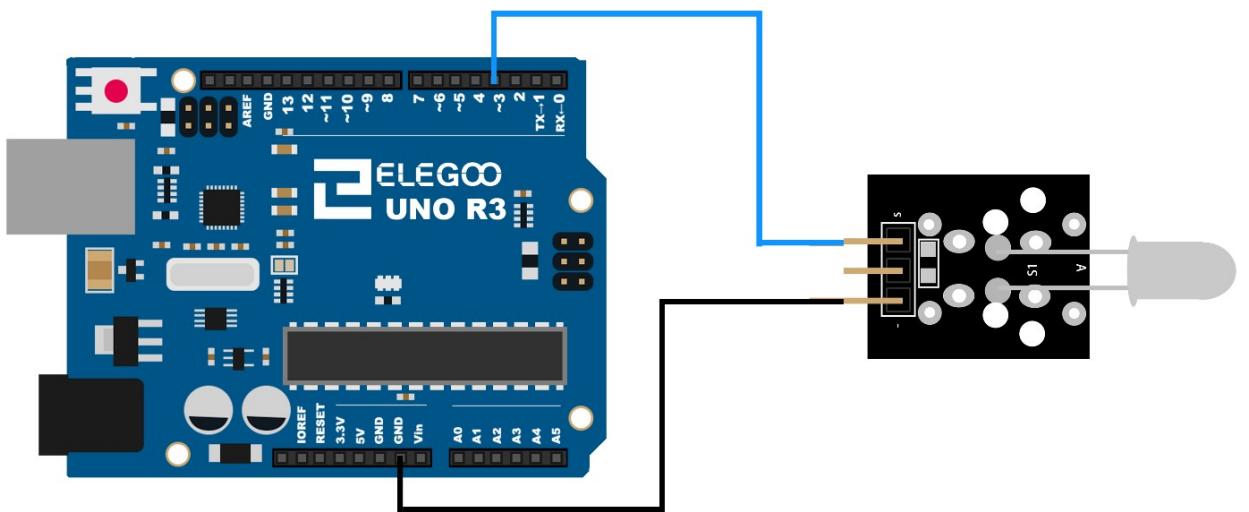
This experiment can be extended use: The IR signal control UNO to operate other modules such as robots、toys、appliances etc.

Schematic

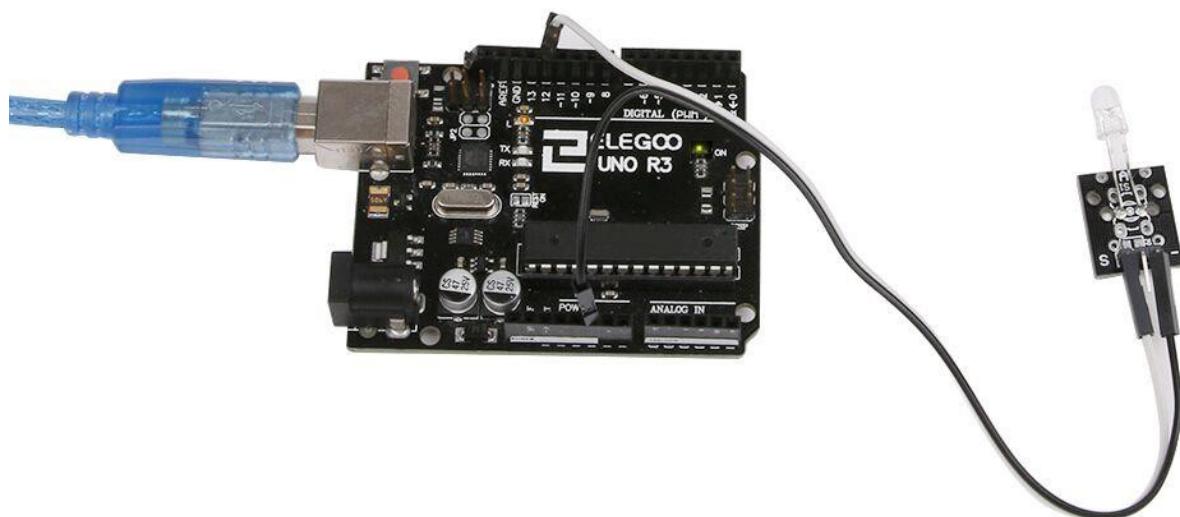
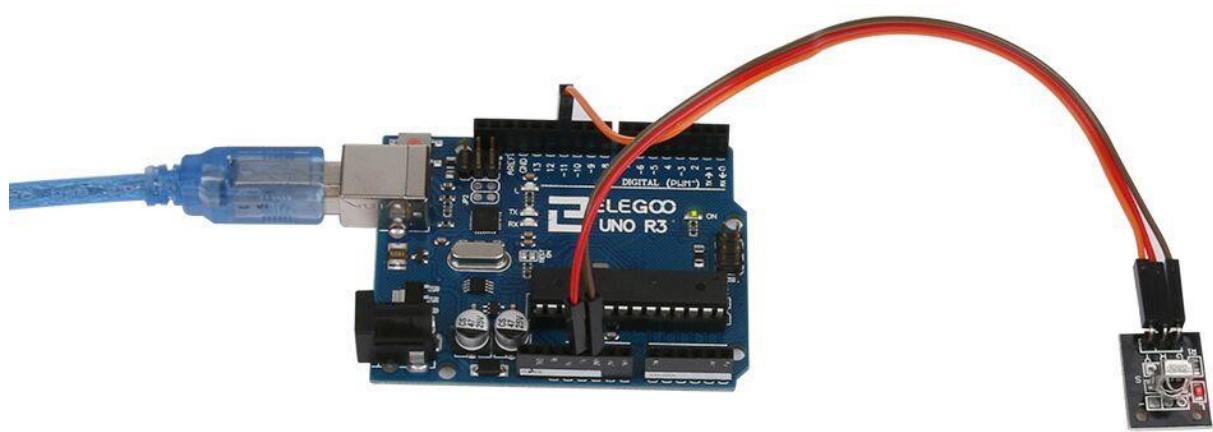


Wiring diagram





Example picture



Troubleshooting

If the L lamp didn't flash, it may be the error of connection or the wrong operation when you wrote the program. Please refer to the previous steps and modify the mistakes.

But if you are sure the connection and program are correct. There are two ways to help you determine whether the IR receiver module and the IR emission module are damaged.

1. The receiver module.

Please use your TV remote control to control the IR receiver module. If the IR receiver module flashes, it shows the IR receiver module is normal. If not, it couldn't be used.

2. The emission module.

Open your Android phone's camera, put in front of the IR emission's LED. If you could see the purple light, it shows this module is correct, or else it's wrong.

If the modules came across defective, please contact us and we will send the replacement to your place.

The followings are the code used in this experiment and their explanations:

(1) infrared launch program:

```
/* An IR LED must be connected to Arduino PWM pin 3.*/
/*The <IRremote.h> library is referenced*/
#include <IRremote.h>
IRsend irsend;
void setup()
{}
void loop()
{
/* send 0x0 code (8 bits)*/
irsend.sendRC5(0x0, 8);
delay(200);
irsend.sendRC5 (0x1, 8);
delay(200);
}
```

(2) infrared receiver program:

```
/*Referenced the <IRremote.h> library file*/  
  
#include <IRremote.h>  
  
/* Infrared signal receiving pin */  
  
#define RECV_PIN 11  
/* define LED pin */  
  
#define LED 13  
  
IRrecv irrecv(RECV_PIN);  
decode_results results;  
  
void setup() {  
/* initialize LED as an output */  
pinMode(LED, OUTPUT);  
Serial.begin(9600);  
/* Start the receiver */  
irrecv.enableIRIn();  
}  
  
void loop() {  
if (irrecv.decode(&results)) {  
int state;  
if ( results.value == 1 ) {  
state = HIGH;  
}  
else{  
State = LOW;  
}  
digitalWrite( LED, state );  
Serial.println(results.value);  
/* prepare to receive the next value */  
irrecv.resume();  
}  
}
```

Lesson 9 Active Buzzer Module

Overview

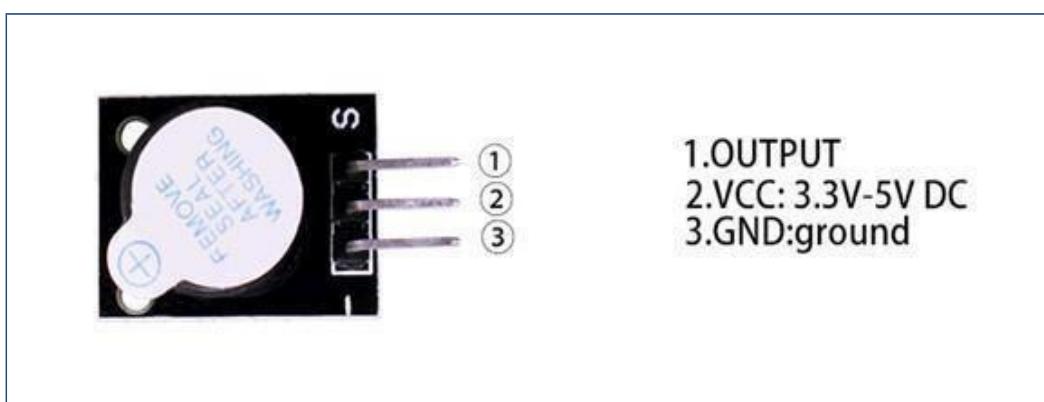
In this experiment, we will learn how to use buzzer module.

With the Arduino we can complete a lot of interactive work, commonly what we used is the light shows. And we have been use the LED small lights in the experiment before. In this experiment we will make the circuit having noise. The common components that can make sound are buzzer and speakers. Compared to the speaker, buzzer is more simple and easy to use so in this experiment we adopt the buzzer.

Component Introduction

Active Buzzer Module:

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.

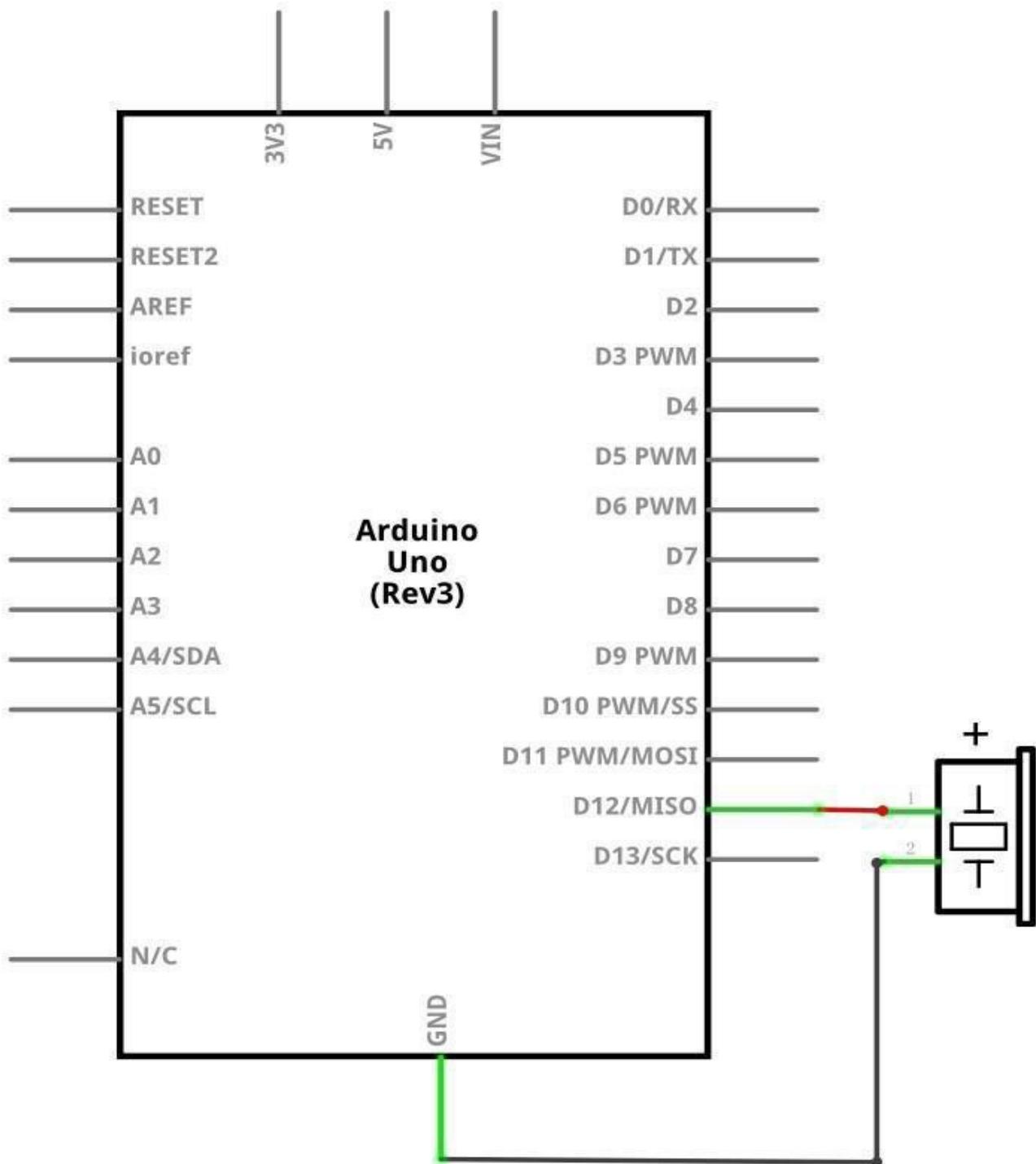


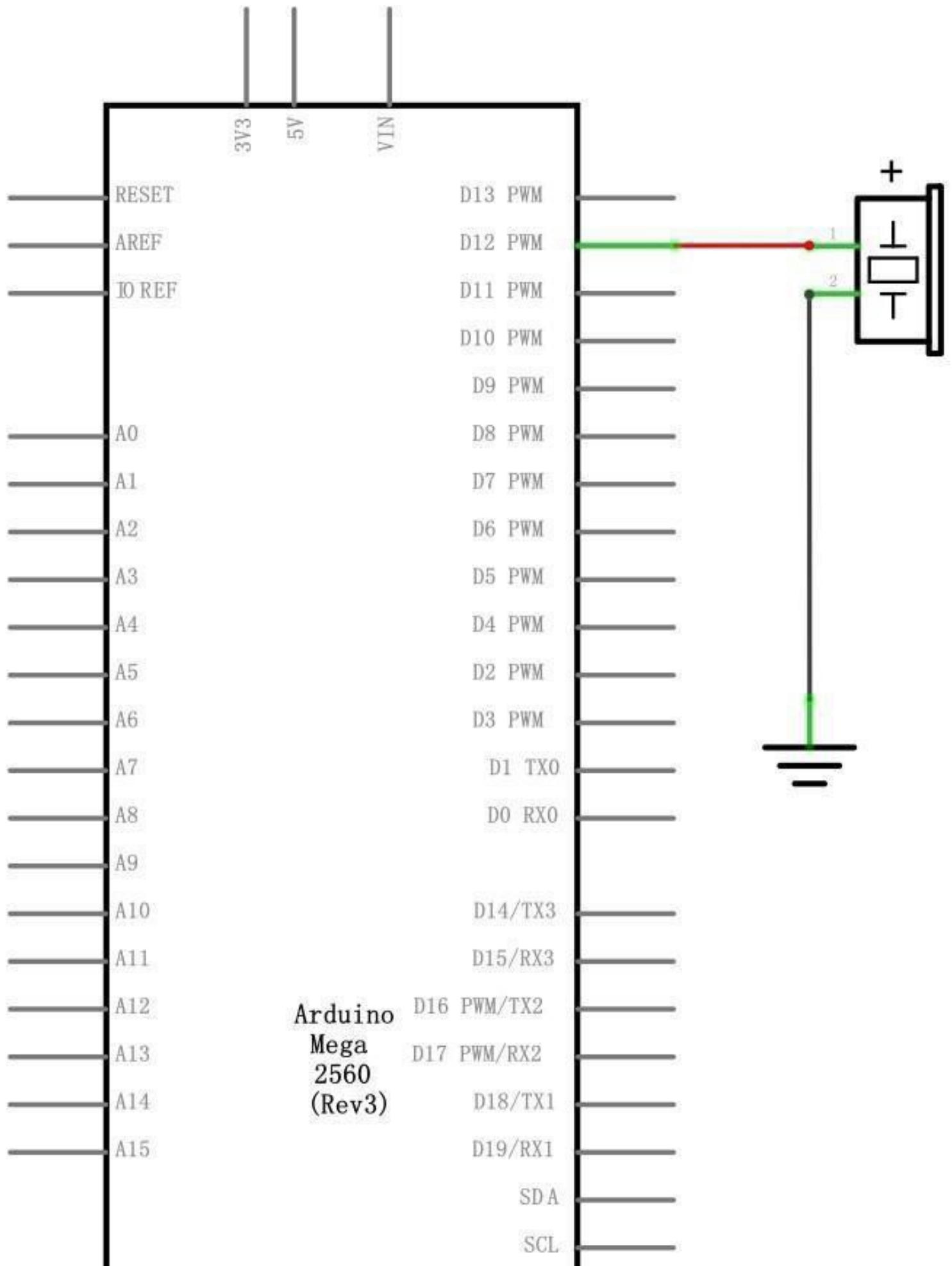
Component Required:

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

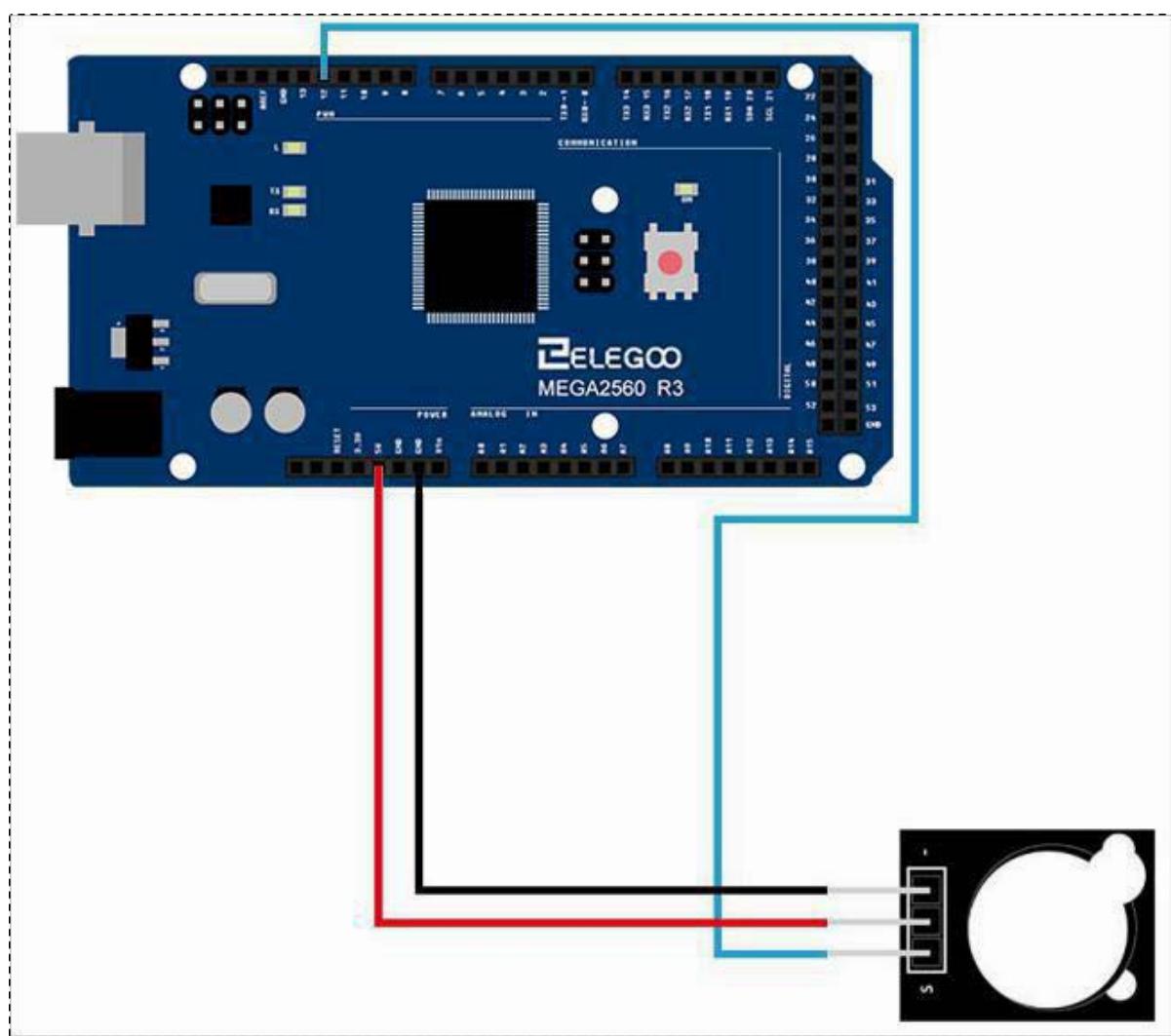
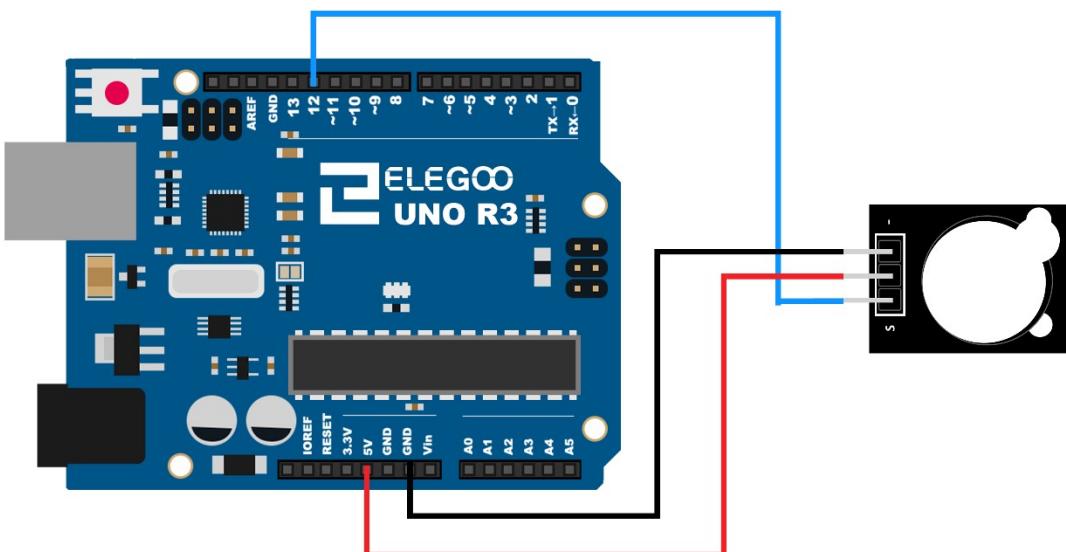
Connection

Schematic





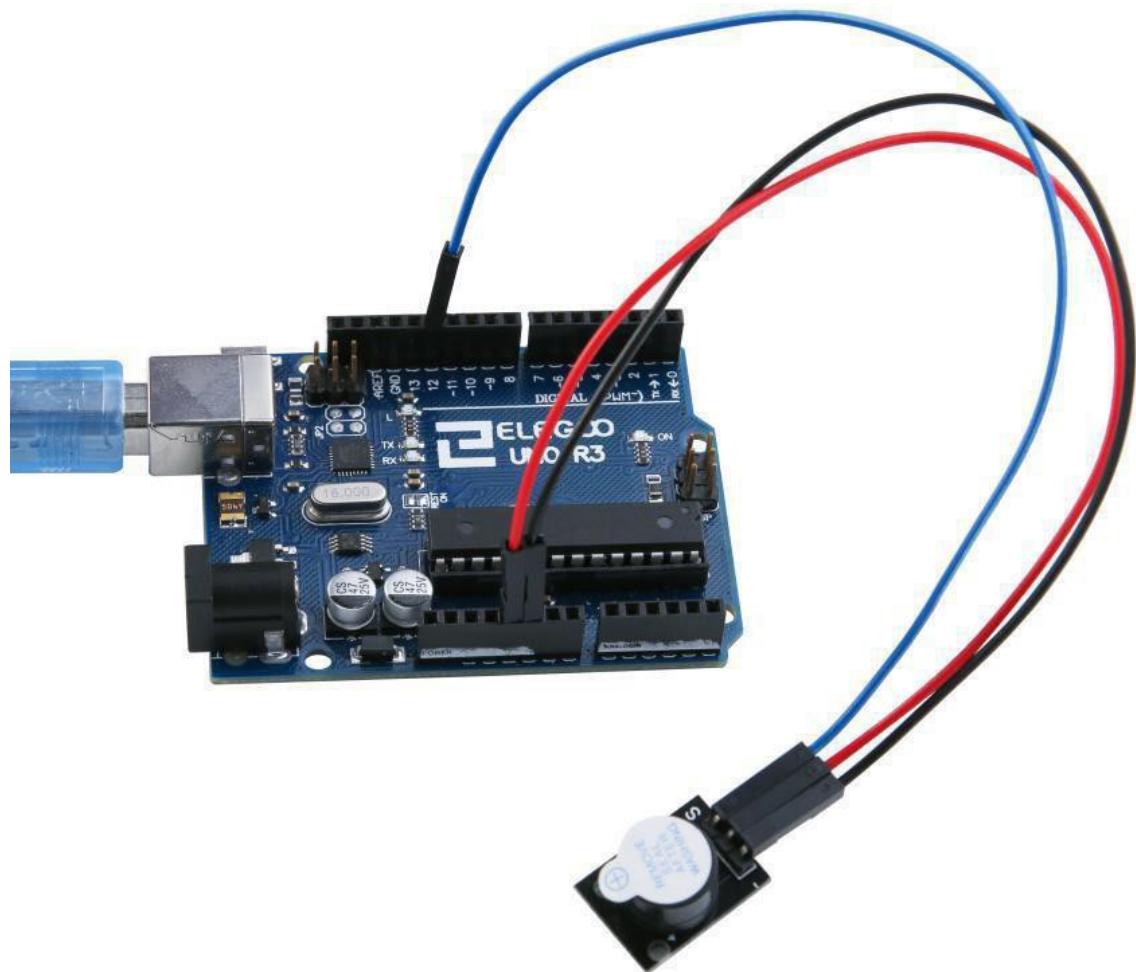
Wiring diagram



Code

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

Example picture



The followings are the code used in this experiment and their explanations:

```
/* the pin of the active buzzer */
int buzzer = 12;

void setup()
{
    /* initialize the buzzer pin as an output */
    pinMode(buzzer, OUTPUT);
}

void loop()
{
    /*define a char variable i */
    unsigned char i;
    while(1)
    {
        /* output an frequency */
        for(i=0;i<80;i++)
        {
            /*When the buzzer is high, it sounds*/
            digitalWrite(buzzer,HIGH);

            delay(1);          //wait for 1ms
            digitalWrite(buzzer,LOW);

            delay(1);          //wait for 1ms
        }

        /* output another frequency */
        for(i=0;i<100;i++)
        {
            digitalWrite(buzzer,HIGH);
            delay(2);          //wait for 2ms
            digitalWrite(buzzer,LOW);
            delay(2);          //wait for 2ms
        }
    }
}
```

From the above program we learned the programming syntax control structure:

(1) while loops

Description

while loops will loop continuously, and infinitely, until the expression inside the parenthesis, () becomes false. Something must change the tested variable, or the while loop will never exit. This could be in your code, such as an incremented variable, or an external condition, such as testing a sensor.

Syntax

```
while(expression){ // statement(s) }
```

Parameters

expression - a (boolean) C statement that evaluates to true or false

Example

```
var = 0; while(var < 200){ // do something repetitive 200 times var++; }
```

(2) for statements

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

There are three parts to the for loop header: for (initialization; condition; increment) { //statement(s); }
The initialization happens first and exactly once. Each time through the loop, the condition is tested; if it's true, the statement block, and the increment is executed, then the condition is tested again. When the condition becomes false, the loop ends.

Example

```
// Dim an LED using a PWM pin int PWMPin = 10; // LED in series with 1k resistor on pin 10
void setup()
{
    // no setup needed
}
void loop()
{
    for (int i=0; i <= 255; i++)
    {
        analogWrite(PWMPin, i);
        delay(10);
    }
}
```

Coding Tip

The C for loop is much more flexible than for loops found in some other computer languages, including BASIC. Any or all of the three header elements may be omitted, although the semicolons are required. Also the statements for initialization, condition, and increment can be any valid C statements with unrelated variables. These types of unusual for statements may provide solutions to some rare programming problems.

Lesson 10 Passive Buzzer Module

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 lasts 0.5 seconds: from Alto Do (523 Hz), Re (587 Hz), Mi (659 Hz), Fa (698 Hz), So (784 Hz), La (880 Hz), Si (988 Hz) to treble Do (1047 Hz).

Component Required:

1x Elegoo Uno R3

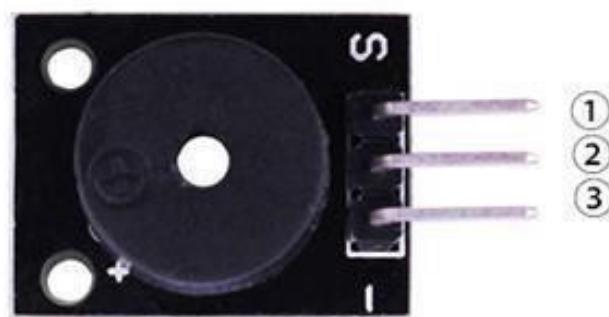
1x Passive buzzer

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

Component Introduction

Passive Buzzer:

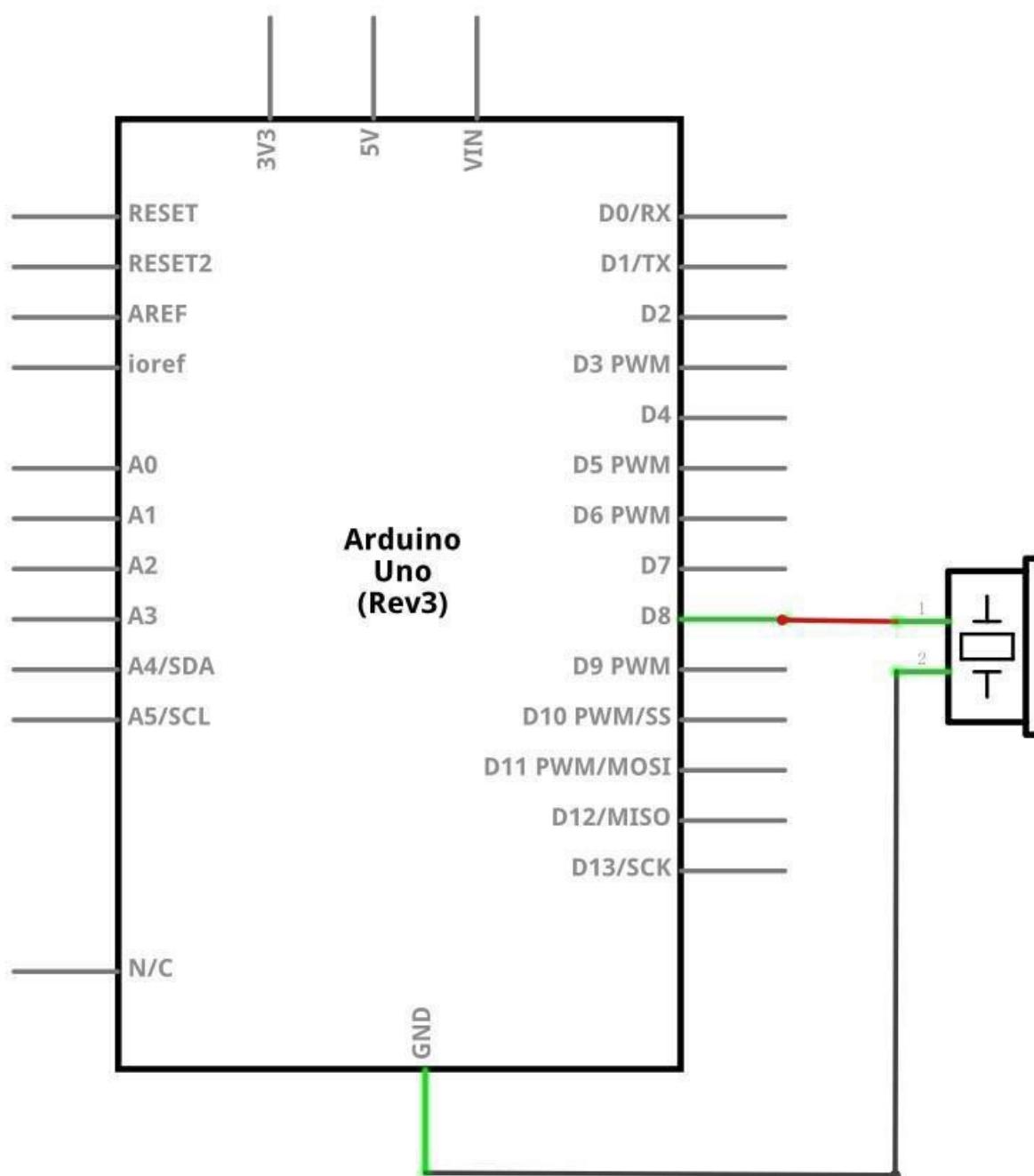
The working principle of the passive buzzer is using PWM generating audio to make the air vibrate. As long as the vibration frequency changes appropriately For example, sending the pulse of 523 Hz, can generate Alto Do, pulse of 587 Hz, it can generate midrange Re, pulse of 659 Hz.



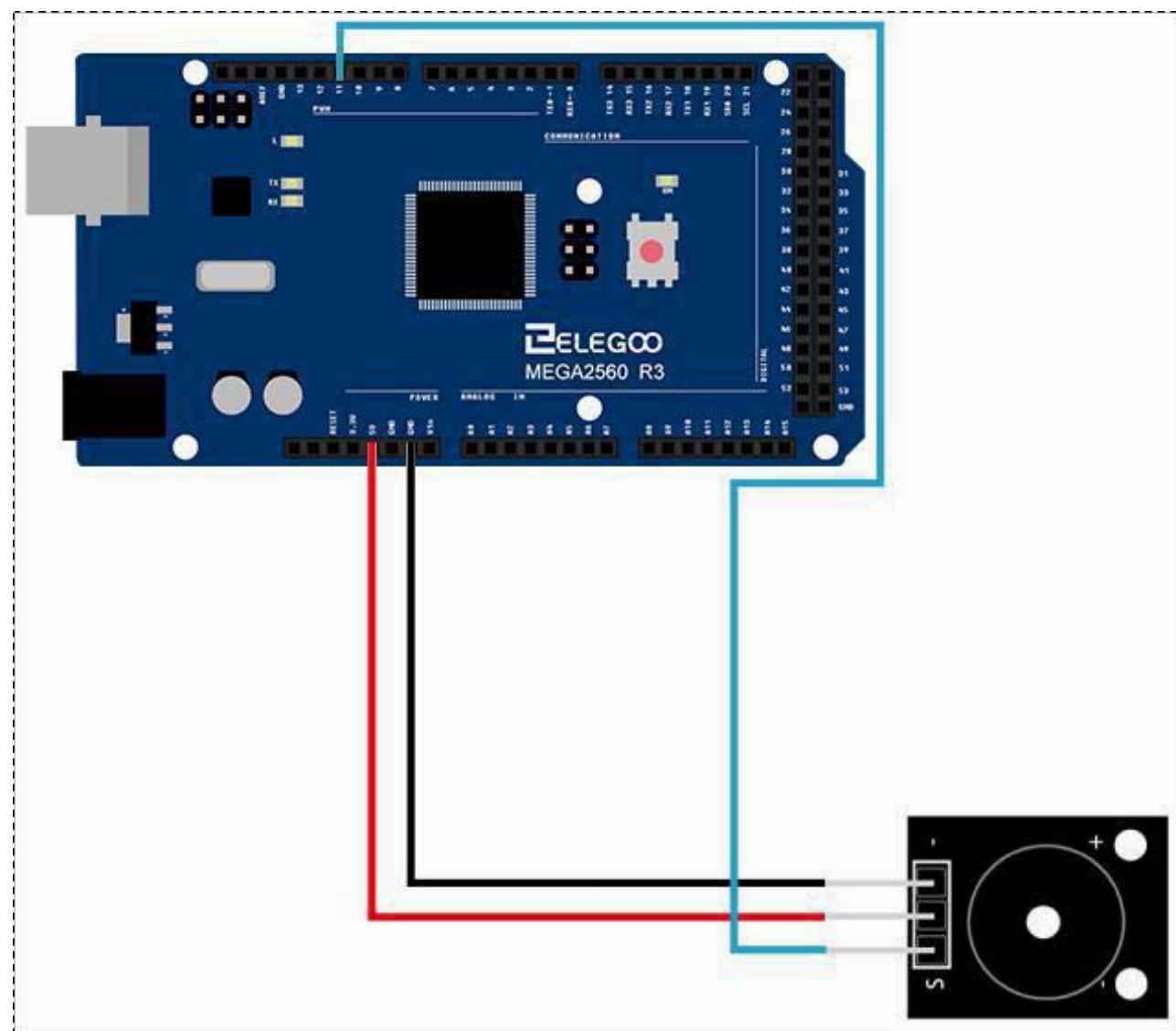
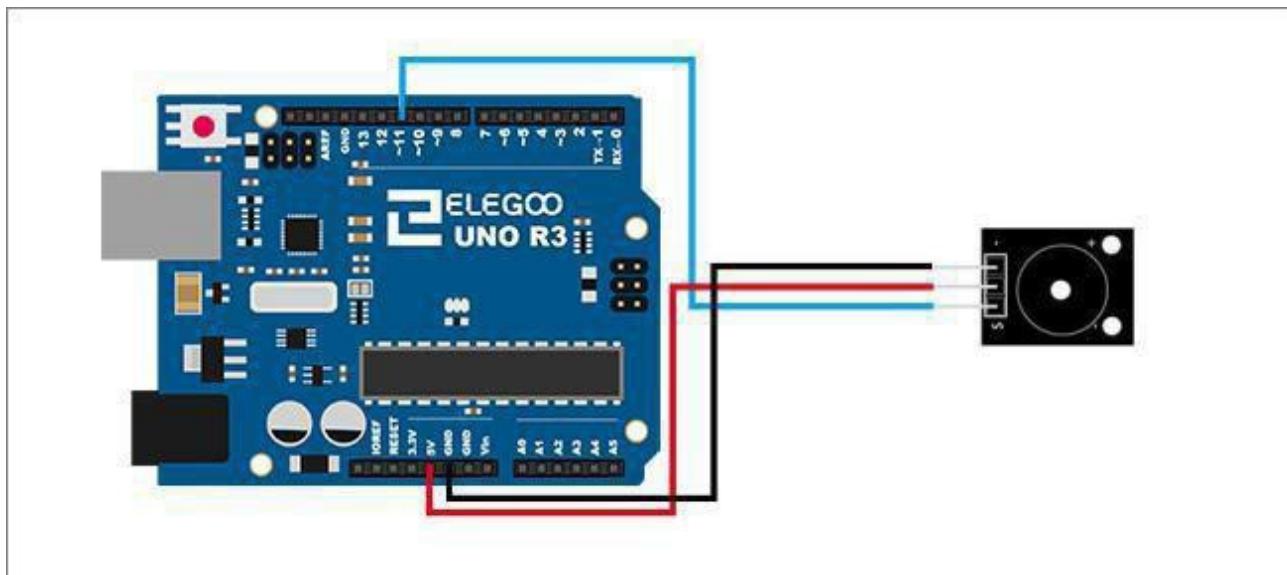
- 1.OUTPUT
- 2.VCC: 3.3V-5V DC
- 3.GND:ground

Connection

Schematic



Wiring diagram



Code

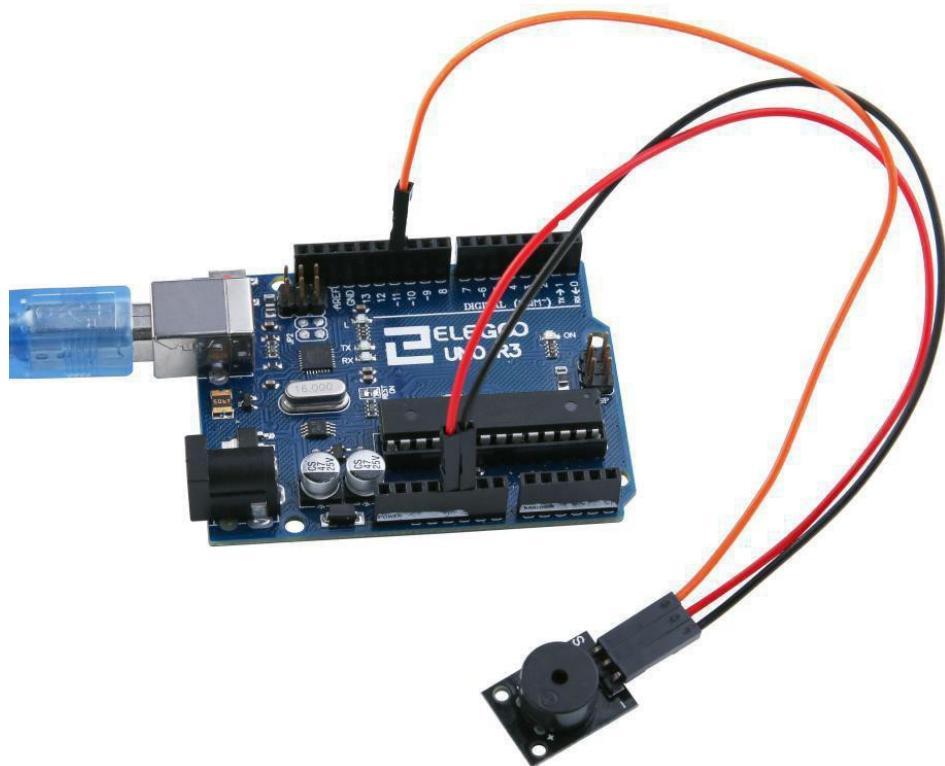
After wiring, please open the program in the code folder- (Lesson 10 Passive buzzer) and click UPLOAD to upload the program. Please refer to Lesson2's details which is about the uploading program and see whether there is any error.

Before you run this, make sure that you have installed the < pitches> library or re-install it, if necessary.

library or re-install it, if necessary. Otherwise, your code won't work.

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

Example picture



The followings are the code used in this experiment and their explanations:

```
#include "pitches.h"

// notes in the melody:
int melody[] = {
    NOTE_C5, NOTE_D5, NOTE_E5, NOTE_F5, NOTE_G5, NOTE_A5, NOTE_B5, NOTE_C6};
int duration = 500; // 500 milliseconds

void setup() {

}

void loop() {
    for (int thisNote = 0; thisNote < 8; thisNote++) {
        // pin8 output the voice, every scale is 0.5 second
        tone(11, melody[thisNote], duration);

        // Output the voice after several minutes
        delay(1000);
    }

    // restart after two seconds
    delay (2000);
}
```

From the above program we learned the programming syntax controlling structure:

(1) while loops

Description

while loops will loop continuously, and infinitely, until the expression inside the parenthesis, () becomes false. Something must change the tested variable, or the while loop will never exit. This could be in your code, such as an incremented variable, or an external condition, such as testing a sensor.

Syntax

```
while(expression){ // statement(s) }
```

Parameters

expression - a (boolean) C statement that evaluates to true or false

Example

```
var = 0; while(var < 200){ // do something repetitive 200 times var++; }
```

(2) for statements

The for statement is used to repeat a block of statements enclosed in curly braces. An increment counter is usually used to increment and terminate the loop. The for statement is useful for any repetitive operation, and is often used in combination with arrays to operate on collections of data/pins. There are three parts to the for loop header: for (initialization; condition; increment) { //statement(s); } The initialization happens first and exactly once. Each time through the loop, the condition is tested; if it's true, the statement block, and the increment is executed, then the condition is tested again. When the condition becomes false, the loop ends.

Example

```
// Dim an LED using a PWM pin int PWMpin = 10; // LED in series with 1k resistor on pin 10
```

```
void setup()
```

```
{ // no setup needed }
```

```
void loop()
```

```
{ for (int i=0; i <= 255; i++)
```

```
{
```

```
analog Write(PWMpin, i);
```

```
delay(10);
```

```
}
```

```
}
```

Coding Tip

The C for loop is much more flexible than for loops found in some other computer languages, including BASIC. Any or all of the three header elements may be omitted, although the semicolons are required. Also the statements for initialization, condition, and increment can be any valid C statements with unrelated variables. These types of unusual for statements may provide solutions to some rare programming problems.

Lesson 11 Laser Module

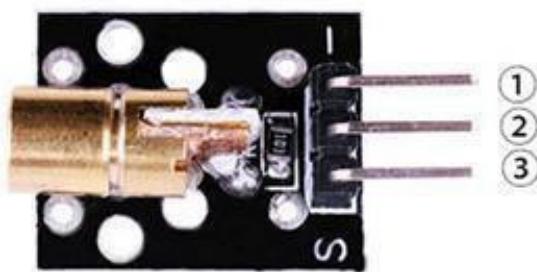
Overview

In this experiment, we will learn how to use laser module.

Laser emitter

Connect the first ("-") pin on the ir module to GND pin on the board, and the second pin to the 5V pin on board, last but not least, the third ("S") pin to the digital port 9.

Transmission Wavelength: 650 nm



- 1.GND:ground
- 2.VCC:3.3V-5V DC
- 3.OUTPUT

Component Required:

1x Elegoo Uno R3

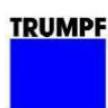
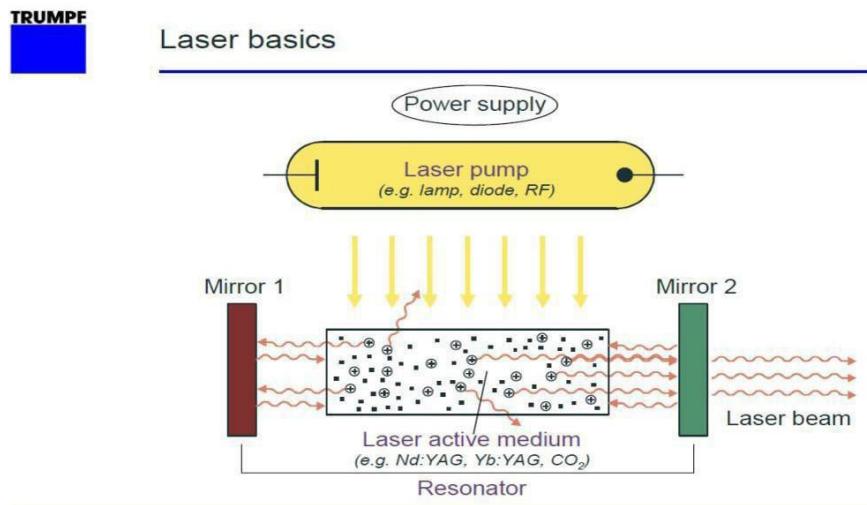
1x USB cable

1x Laser module

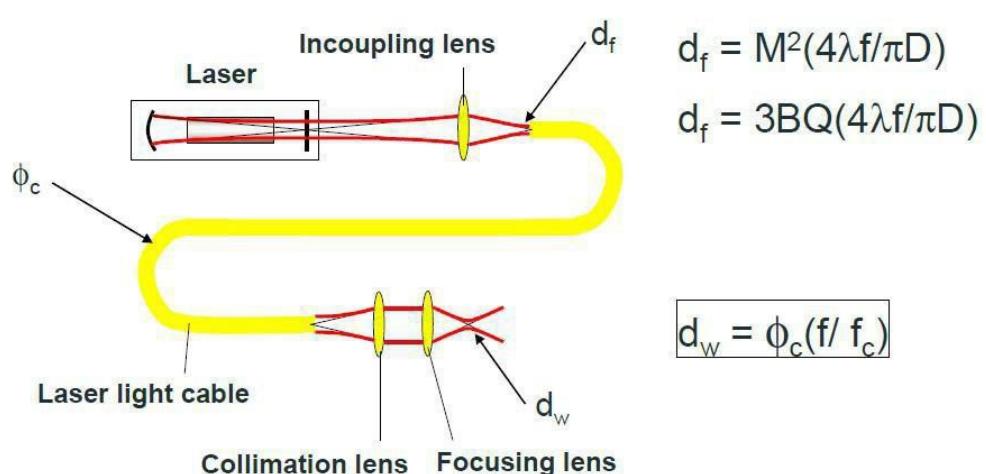
3x F-M wires

Component Introduction

Lasersensor:

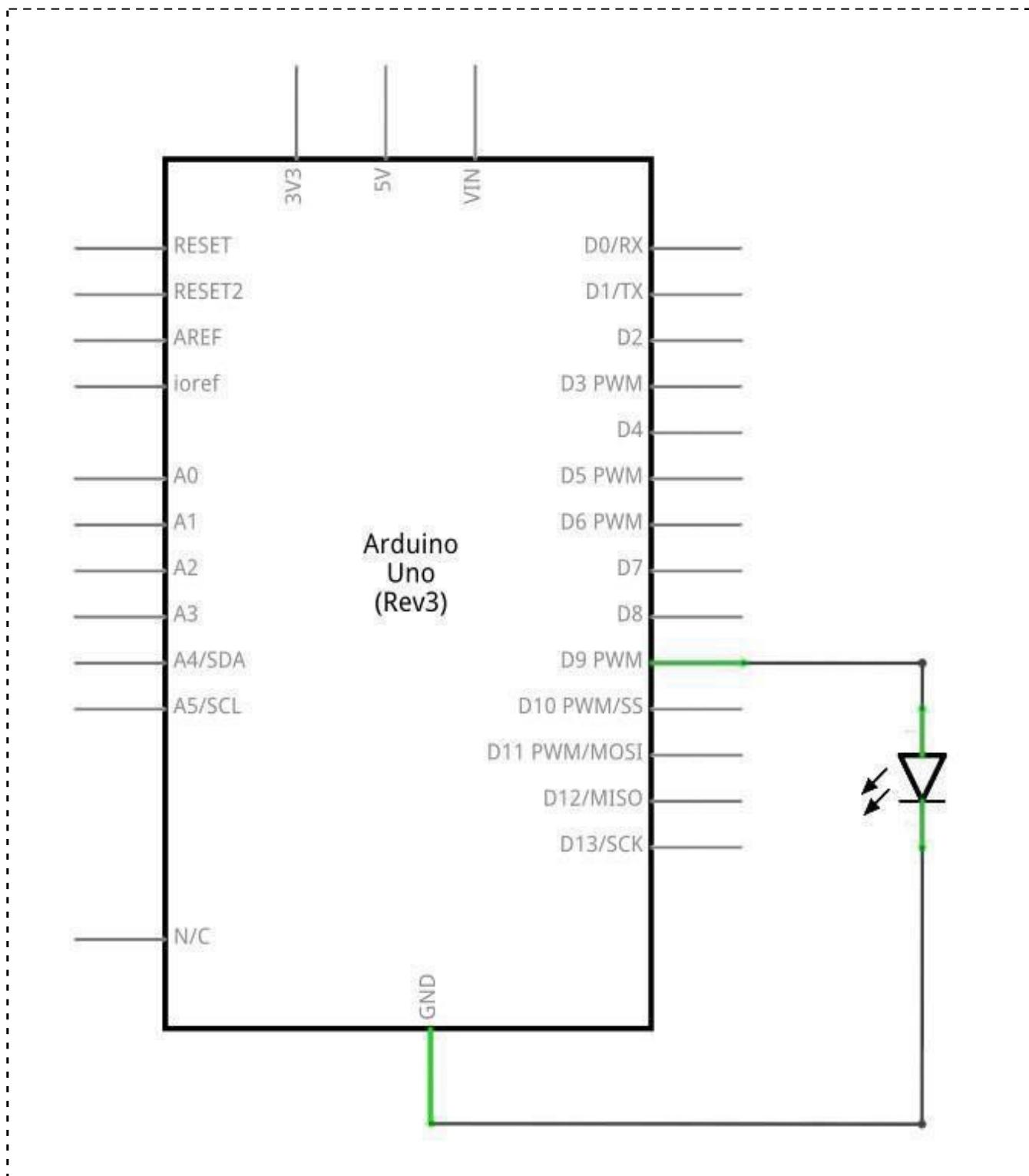


Spot size - YAG

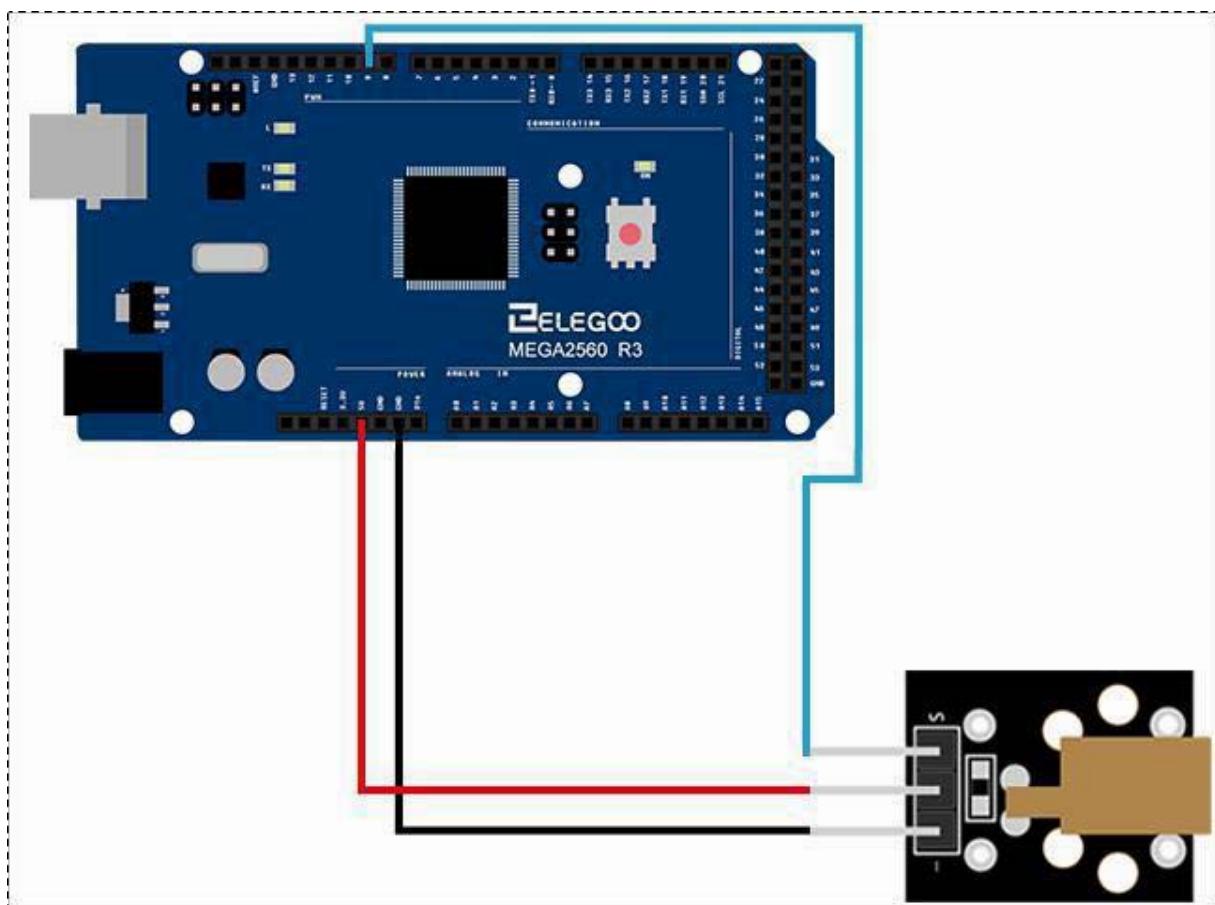
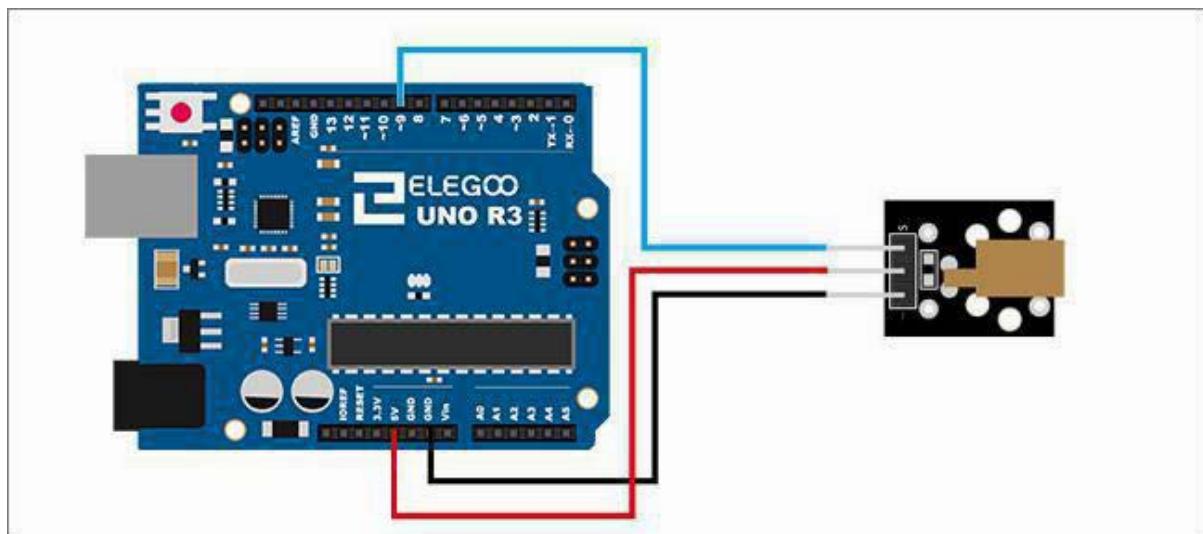


Connection

Schematic



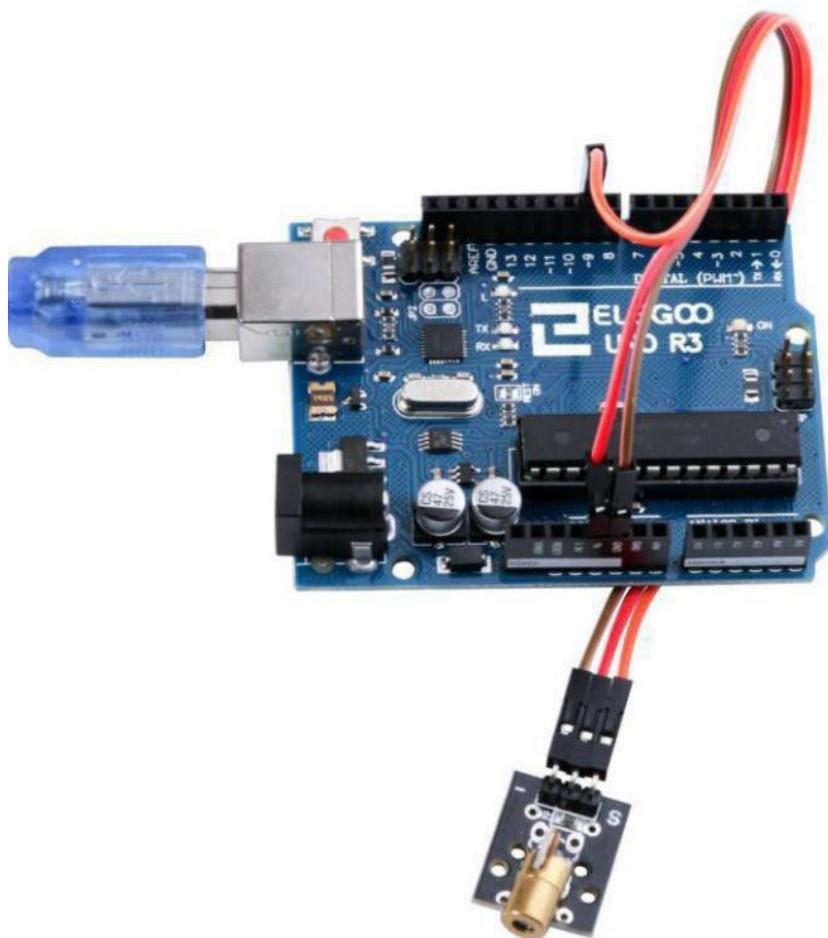
Wiring diagram



Code

After we connect the circuits, we find "Lesson 11 Laser Module" in the "code" folder of the tutorials and run the loader. We can see that the module emits laser light.

Example picture



The followings are the code used in this experiment and their explanations:

```
/*Define an integer variable pos and assign it to zero*/
int pos = 0;

/* void setup () The setup () function is called when the Arduino board is started. Use it to initialize variables, pin
patterns, get started with a library, and much more. This function runs only once for each Arduino board power-
up and reset.*/
void setup()

{

/*Define module pins and set as output type*/

pinMode (9,OUTPUT);

}

void loop()

{

for (pos = 0; pos <= 255; pos += 1)

{

/*Read the value of the module variable*/

analog Write(9,pos);

delay(25);

}

for (pos = 255; pos >= 0; pos -= 1)

{

analog Write(9,pos);

delay(25);

}

}
```

Lesson 12 SMD RGB Module and RGB Module

Overview

In this experiment, we will learn how to use SMD RGB LED module and RGB LED module.

Actually, the function of SMD RGB LED module and RGB LED module are almost the same. But we can choose the shape we like or we need.

SMD RGB LED module and RGB LED module are made from a patch of full-color LED. By adjusting the voltage input of R, G, B pins, we can adjust the strength of the three primary colors (red/blue/green) so as to implementation result of full color effect.

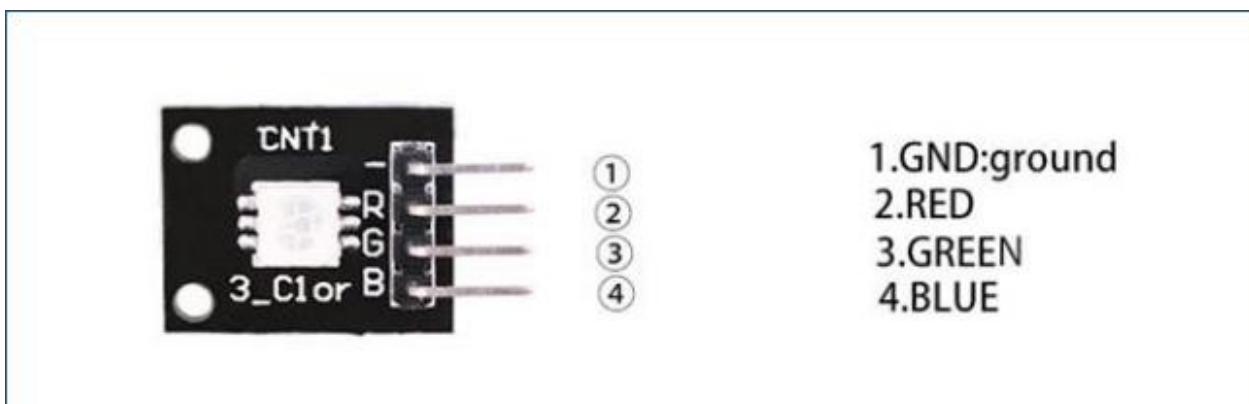
RGB LED Module

RGB-LED with clear lens and built-in 150 ohm series resistor for 5 V operation.



SMD RGB LED Module

RGB-LED with an SMD housing and no series resistor.



Component Required:

1x Elegoo Uno R3

1x USB cable

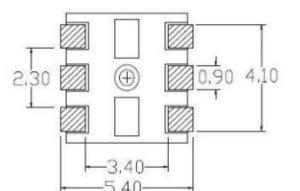
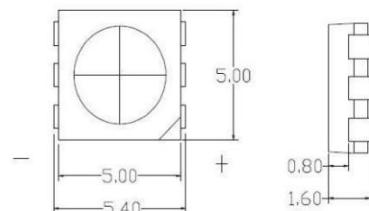
1x SMD RGB module

1x RGB module

4x F-M wires

Component Introduction

SMD RGB:

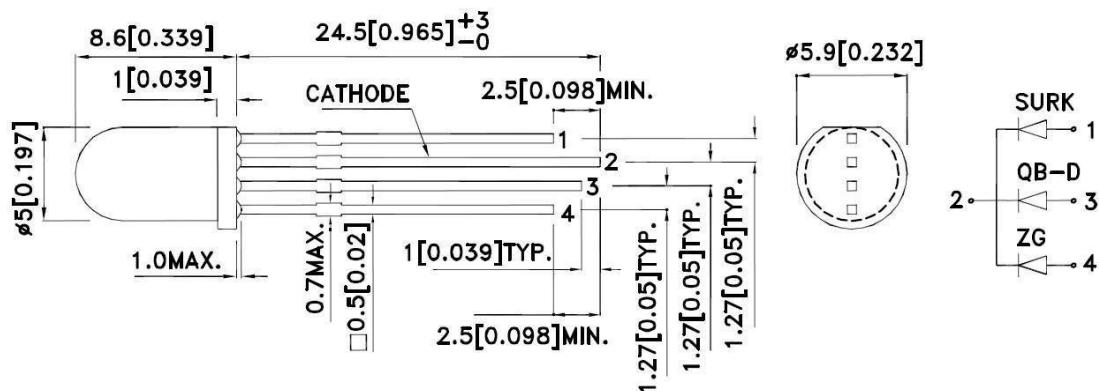


All
dime
nsio
ns
are

in millimeter

Tolerance is $\pm 0.25\text{mm}(0.10")$ unless otherwise noted

Parameter	Symbol	Value	Unit
Forward Current	If	20	mA
Reverse Voltage	Vr	5	V
Operating Temperature	Topr	-25~+85	°C
Storage Temperature	Tstg	-35~+85	°C
Soldering temperature	Tsol	260±5°C (for 4sec)	°C
Power Dissipation	Pd	R=40 CD-60	mW
Pulse Current	I _{FP}	100	mA

RGB LED:**Electrical / Optical Characteristics at TA=25°C**

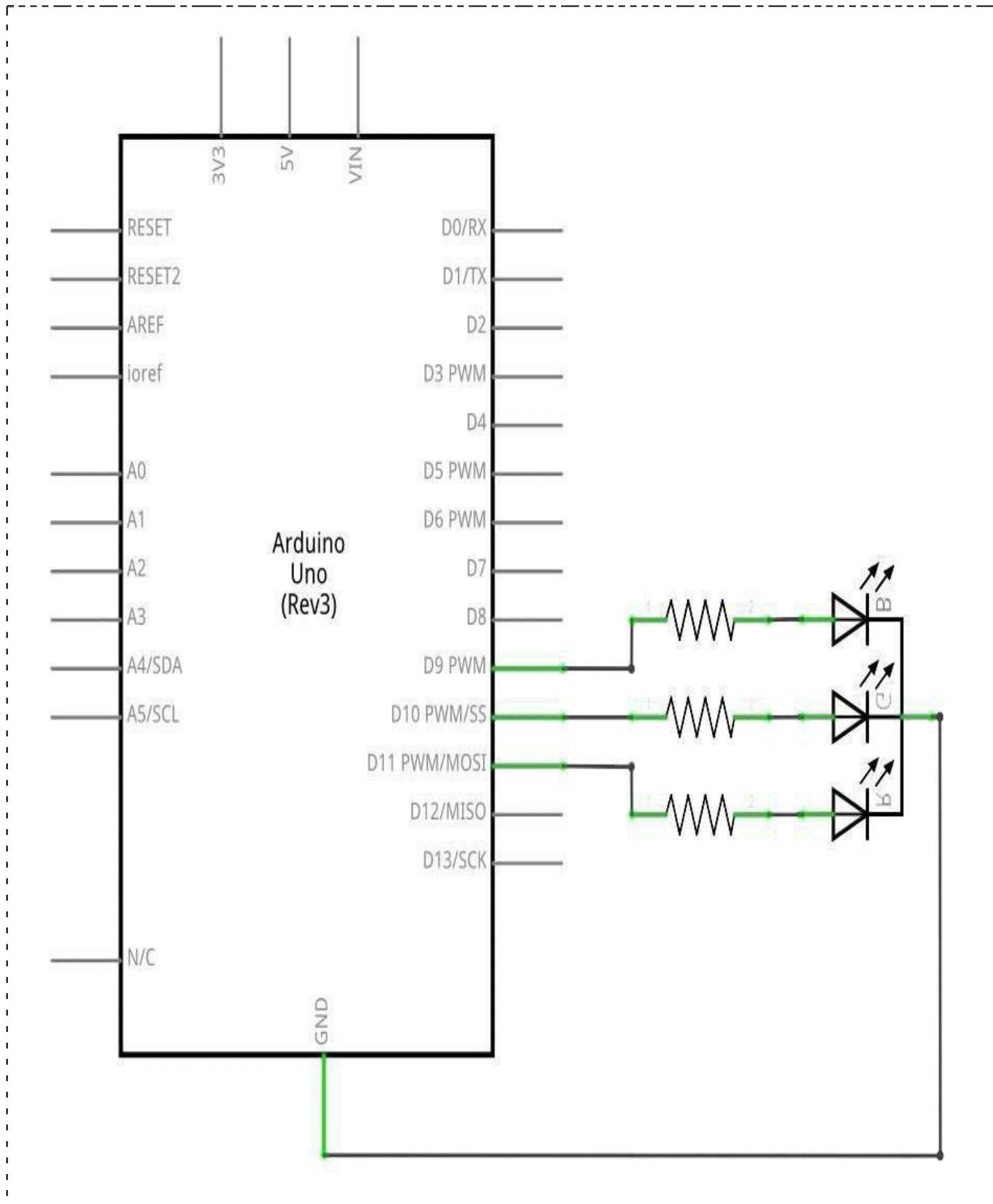
Symbol	Parameter	Device	Typ.	Max.	Units	Test Conditions
λ_{peak}	Peak Wavelength	Hyper Red Blue Green	650 468 515		nm	$I_F=20mA$
λ_D [1]	Dominant Wavelength	Hyper Red Blue Green	630 470 525		nm	$I_F=20mA$
$\Delta\lambda_{1/2}$	Spectral Line Half-width	Hyper Red Blue Green	28 25 30		nm	$I_F=20mA$
C	Capacitance	Hyper Red Blue Green	35 100 45		pF	$V_F=0V; f=1MHz$
V_F [2]	Forward Voltage	Hyper Red Blue Green	1.95 3.3 3.3	2.5 4 4.1	V	$I_F=20mA$
I_R	Reverse Current	Hyper Red Blue Green		10 50 50	uA	$V_R=5V$

Notes:

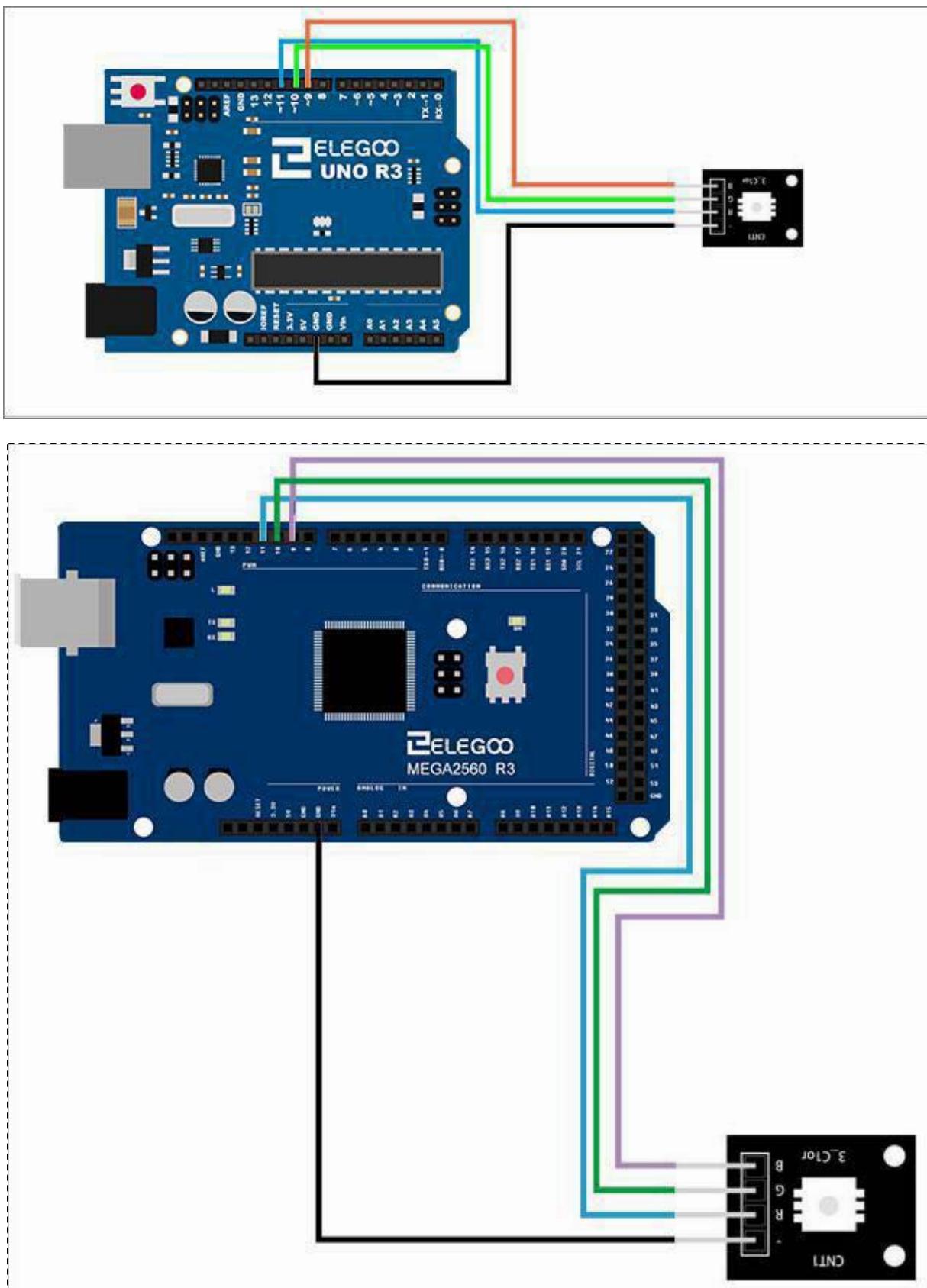
1. Wavelength: +/-1nm.
2. Forward Voltage: +/-0.1V.

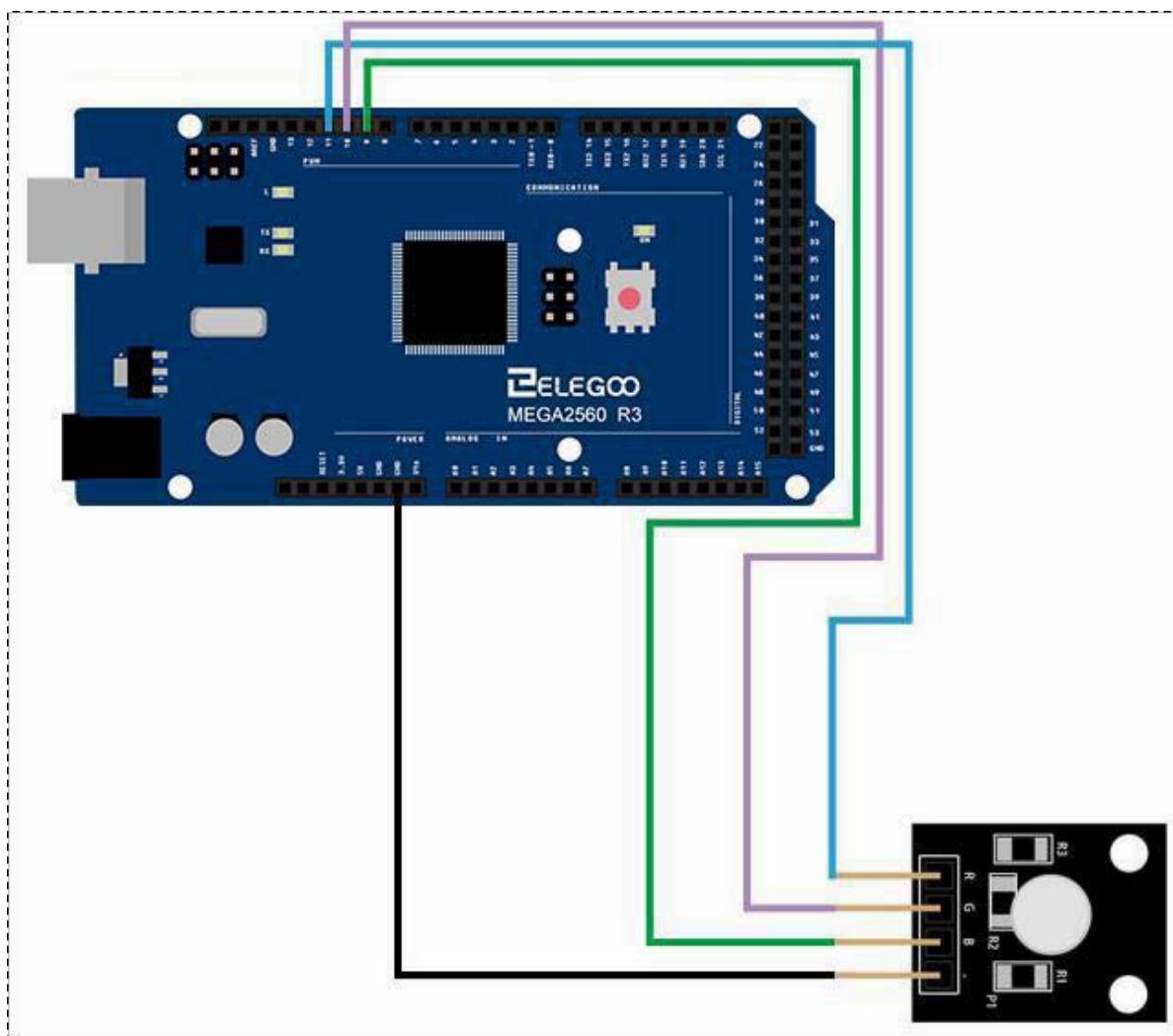
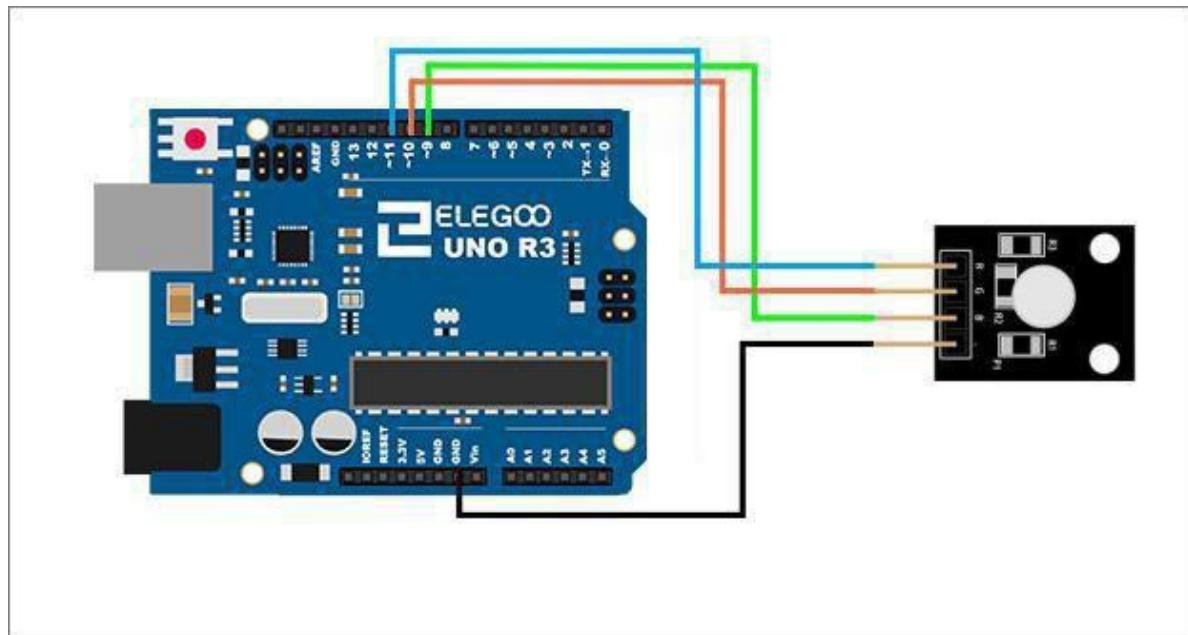
Connection

Schematic



Wiring diagram

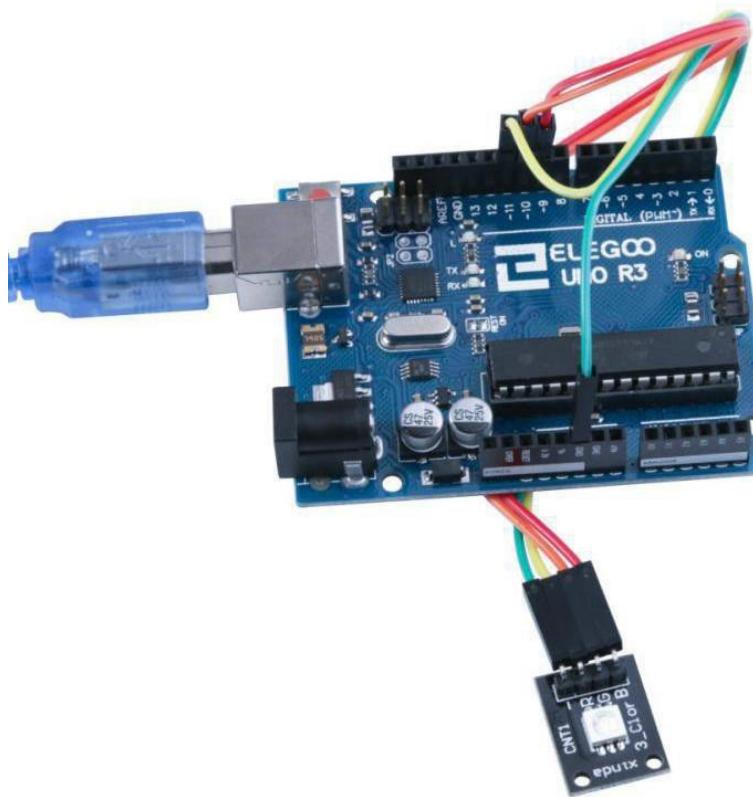


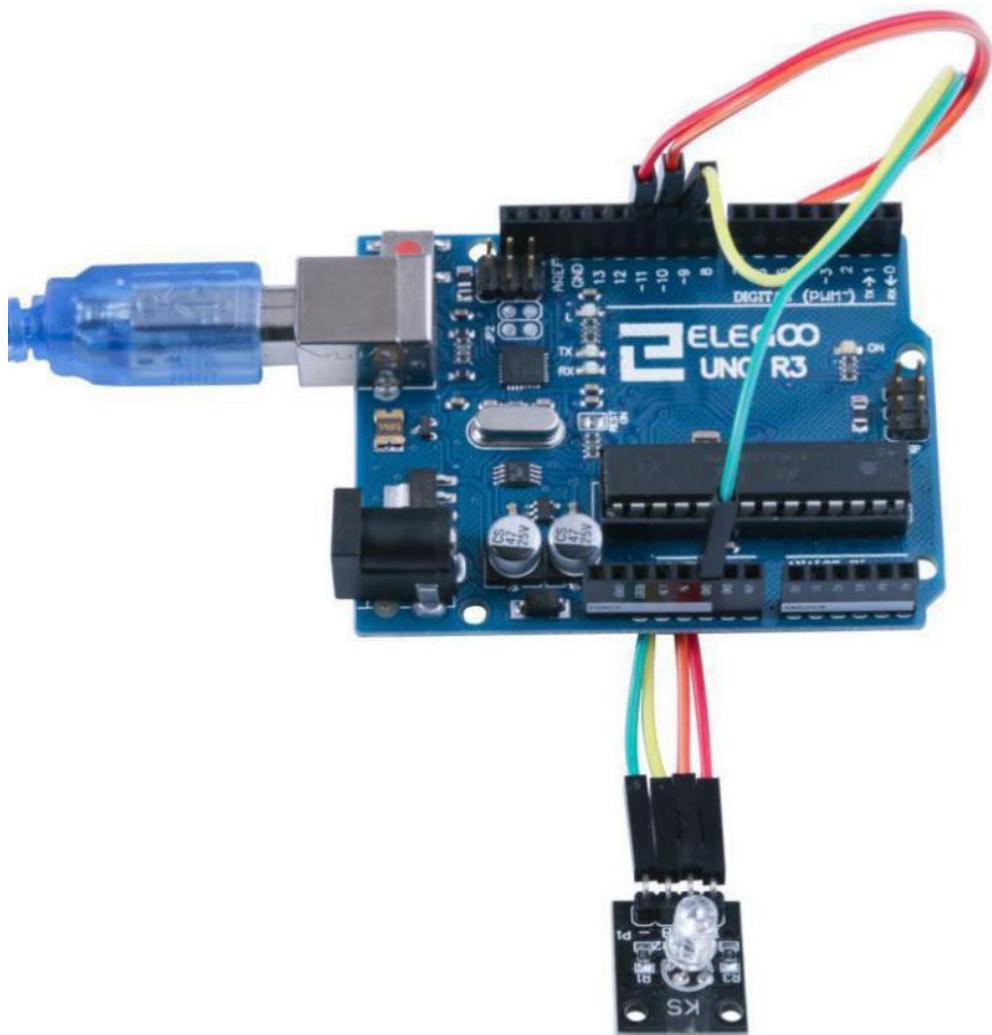


Result

After we connect the circuits, we open the "code" folder in our documentation to find the "Lesson 12 SMD RGB MODULE AND RGB MODULE" folder. Open the Program Upload Run Loader. We can see that the module changes their color as a code set. If you want to change color in different ways, you can modify the code.

Example picture





The followings are the code used in this experiment and their explanations:

```
/* select the pin for the red LED */
int redpin = 11;

/* select the pin for the green LED */
int greenpin =10;

/* select the pin for the blue LED */
int bluepin =9;

/*Define an integer variable val*/
int val;

void setup(){

/*Set redpin, bluepin, greenpin output type*/
pinMode(redpin, OUTPUT);
pinMode(bluepin, OUTPUT);
pinMode(greenpin, OUTPUT);
Serial.begin(9600);

}

void loop(){

for(val=255; val>0; val--)

{analogWrite(11, val);
analogWrite(10, 255-val);
analogWrite(9, 128-val);
delay(50);
}

for(val=0; val<255; val++)

{

analogWrite(11, val);
analogWrite(10, 255-val);
analogWrite(9, 128-val);
delay(50);
}

Serial.println(val, DEC);
}
```

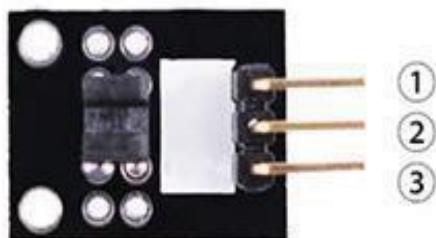
Lesson 13 Photo-Interrupter Module

Overview

In this experiment, we will learn how to use Photo-interrupter module.

Light blocking

Slotted light barrier. The middle pin connects to + 5 V supply and the pin marked '-' connects to ground. The output signal (with a 10 K ohm pullup to +5 V) is available on the pin on the right.



- 1.OUTPUT
- 2.VCC:3.3V-5V DC
- 3.GND:ground

Component Required:

- 1 x Elegoo Uno R3
- 1 x USB cable
- 1 x Photo-interrupter MODULE
- 3 x F-M wires

Component Introduction

Opto Interrupter Sensor:



Opto Interrupters are commonly used in many arcade games e.g. steering assembly in older driving games, scoring switches in Whack A Crock etc. Uninterrupted light beam will turn the Phototransistor "ON" connecting the ground to the game board input. When the light beam is interrupted the phototransistor turns "OFF" , the ground is disconnected from the input and the pull-up resistor R1 forces the input to go "HIGH" (5Vlevel).

Principle

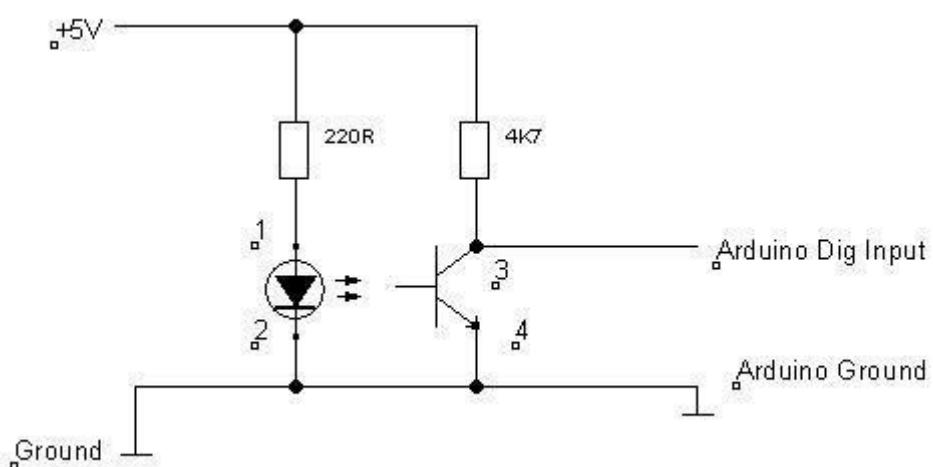
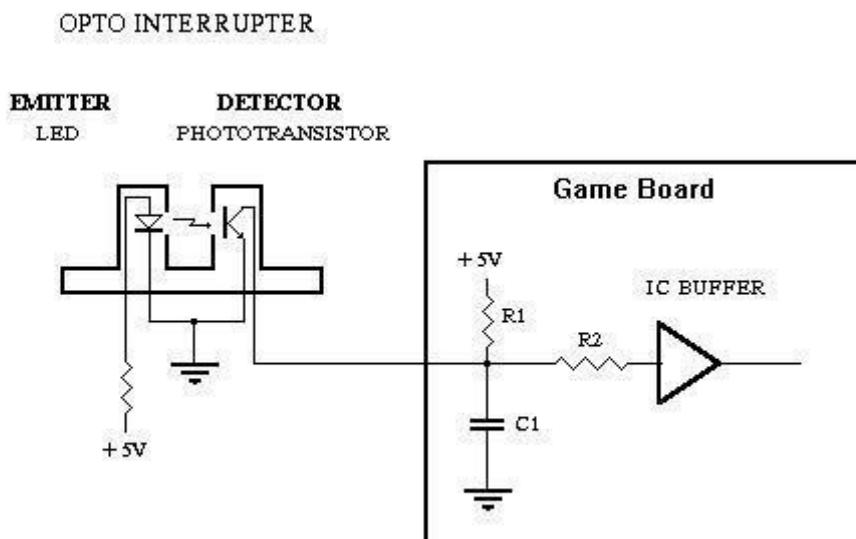
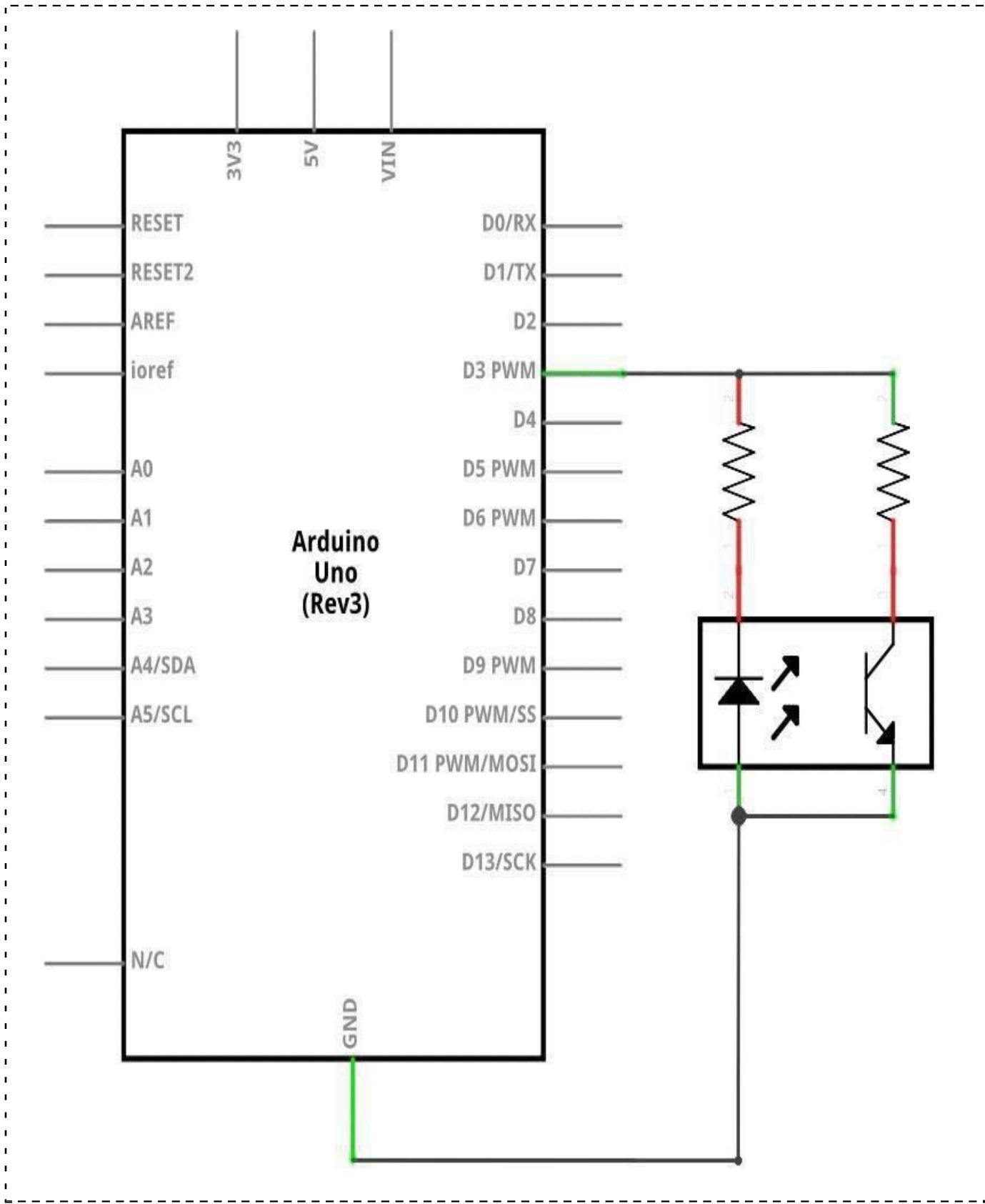


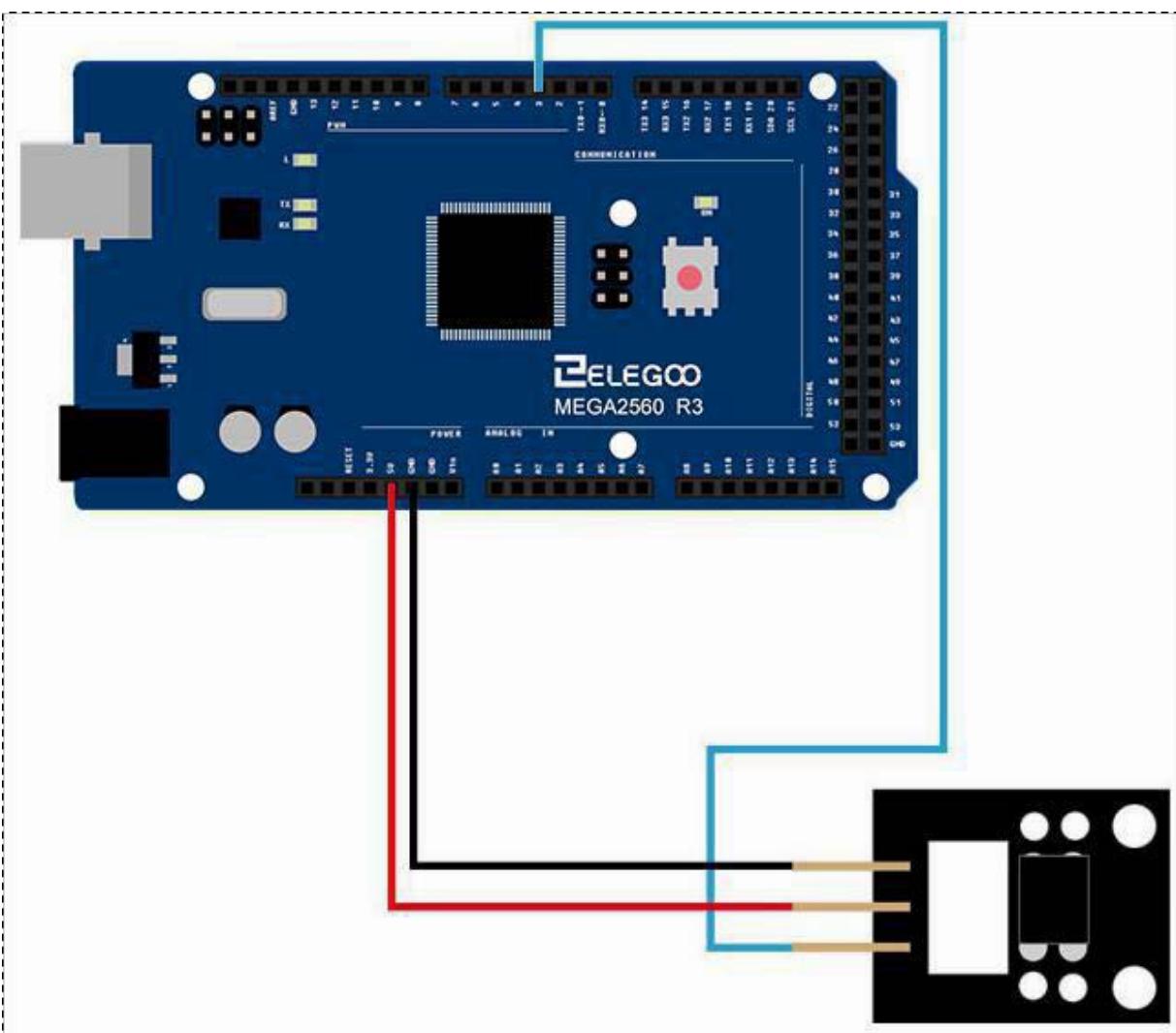
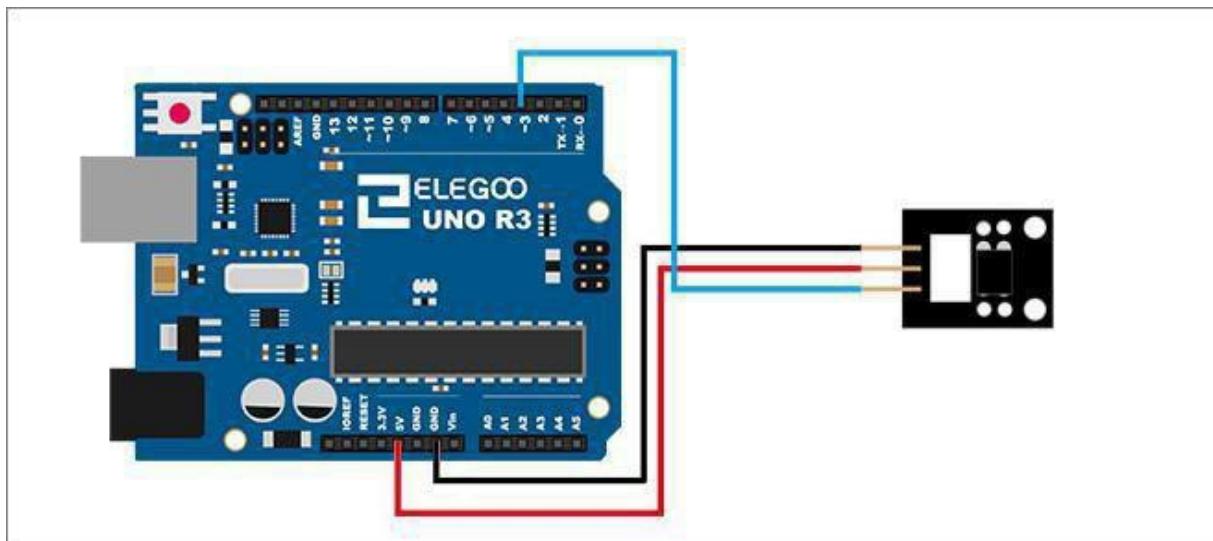
Photo-interrupter module and number 13 port have the built-in LED simple circuit. To produce a switch flasher, we can use connect the digital port 13 to the built-in LED and connect the Photo-interrupter MODULE S port to number 3 port of Elegoo Uno board. When the switch sensing, LED twinkle light to the switch signal.

Connection

Schematic



Wiring diagram

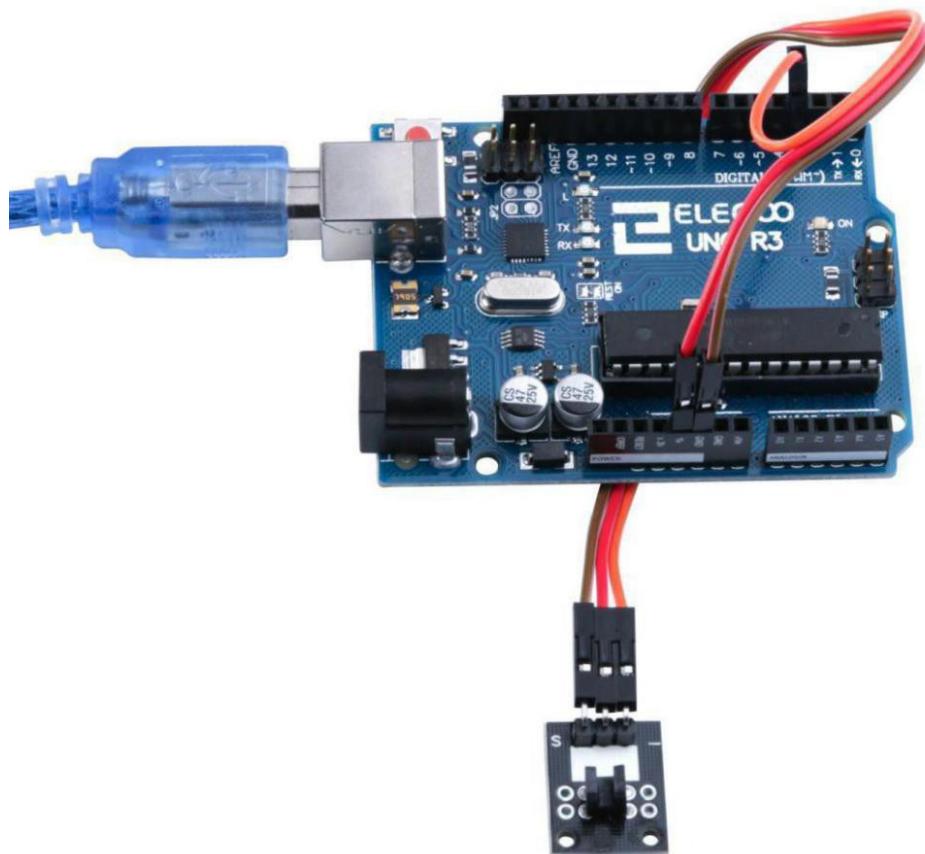


Result

After you finish the wiring, open the tutorial file and go to code > Lesson 13 Photo-interrupter MODULE and run the program (the .ino file).

Put a piece of paper in the groove on the module and take it away, then you will see the LED 13 goes out and lights up.

Example picture



The following is the code needed for this experiment and the corresponding explanation:

```
/* define the led's port */
int Led=13;
/* define the port of light blocking module */
int buttonpin=3;
/* define digital variable val */
int val;
void setup ()
{
/* define digital variable val */
pinMode(Led,OUTPUT);
/* define light blocking module as a output port */
pinMode(buttonpin,INPUT);
}
void loop()
{
/* read the value of the digital interface 3 assigned to val */
val=digitalRead(buttonpin);
/* when the light blocking sensor have signal, LED blink */
if(val==HIGH)
{
digitalWrite(Led,LOW);
}
else
{
digitalWrite(Led,HIGH);
}
}
```

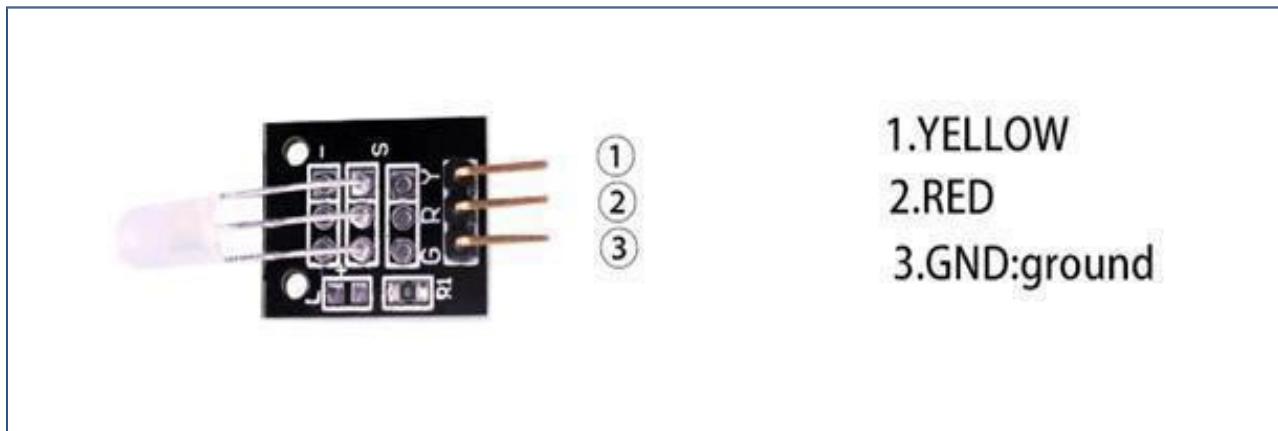
Lesson 14 Two Color LED Module (5 mm)

Overview

In this experiment, we will learn how to use Dual-color Common-Cathode LED.

Two-Color LED Module 5 mm

The 5 mm LED has a common cathode connected to the '-' pin on the PCB. The center pin connects to the red anode and the 'S' pin connects to the green anode. No resistor in series is included in circuit. A suitable value of resistance for low voltage operation would be 220 ohms.



Component Required:

1x Elegoo Uno R3

1x USB cable

1 x Dual-color Common-Cathode LED

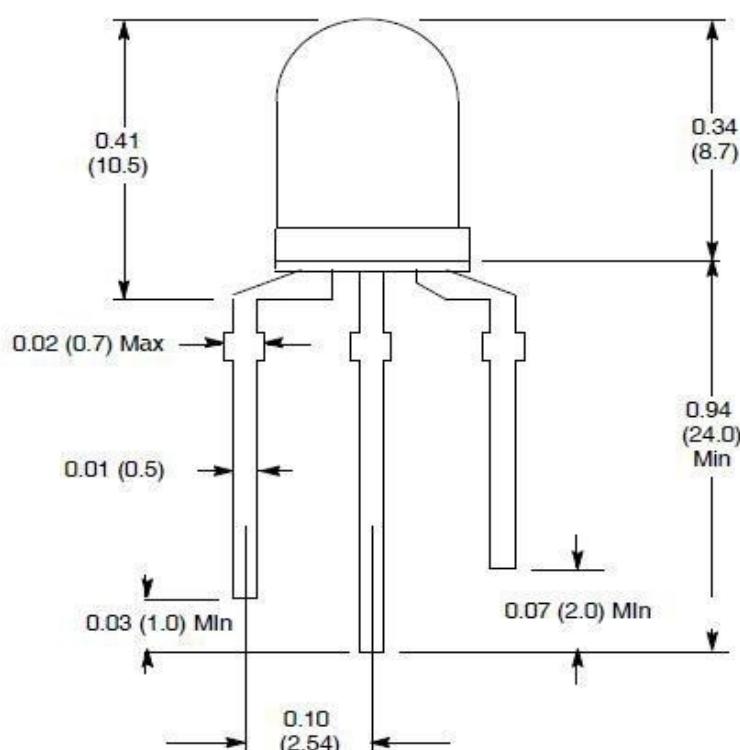
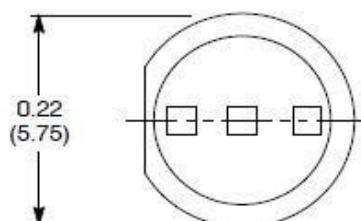
2x F-M wires

Component Introduction

Dual-color Common-Cathodeled:

Electro-Optical Characteristics: ($T_A = +25^\circ\text{C}$ unless otherwise specified)

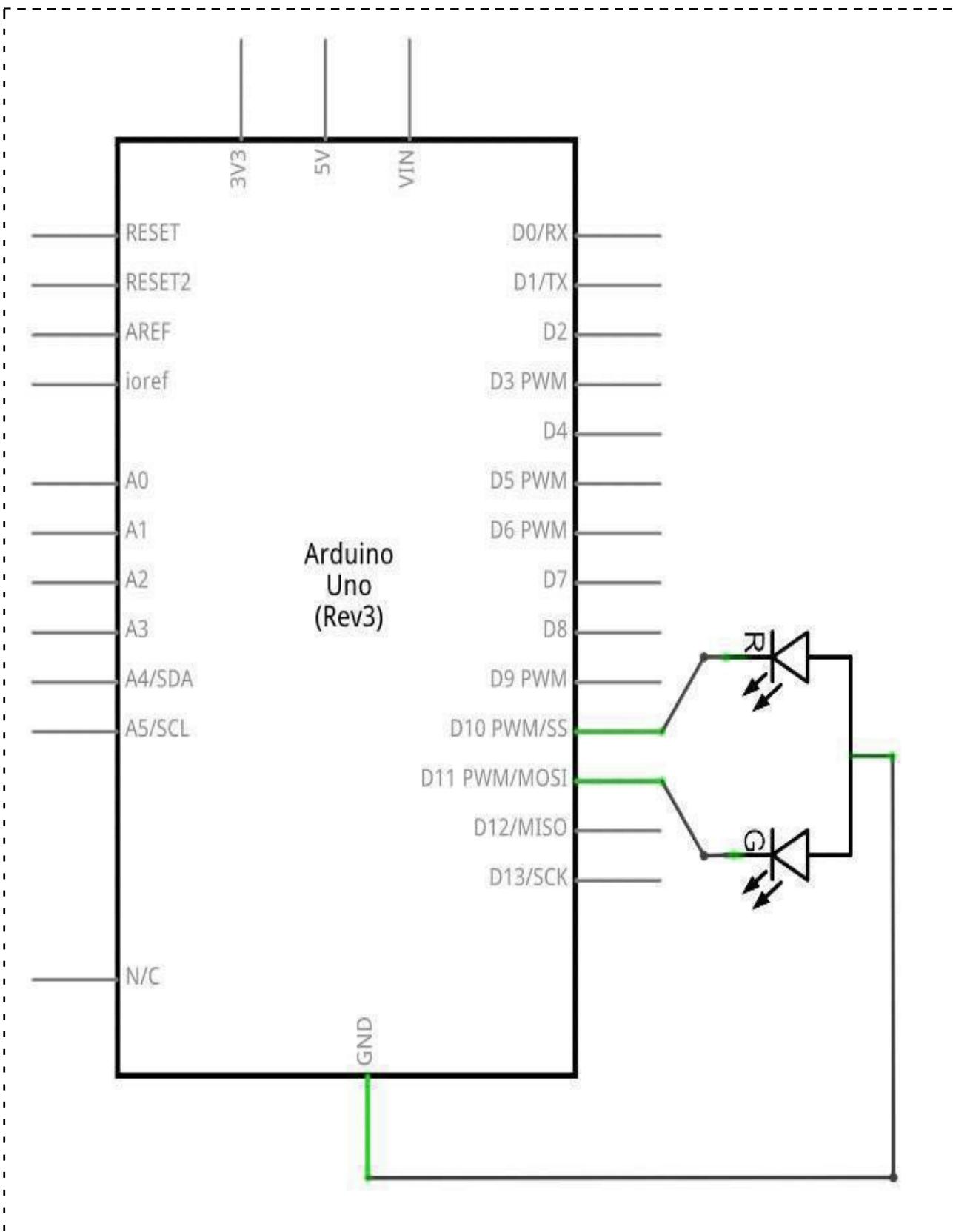
Parameter	Symbol	Test Conditions	Min	Typ	Max	Unit
View Angle of Half Power	$2\theta_{1/2}$	IF = 20mA	-	40	-	deg
Forward Voltage High Efficiency Red	VF	IF = 20mA	-	2.05	2.80	V
Yellow-Green			-	2.15	2.80	V
Luminous Intensity (Note 1)	IV	IF = 20mA	35	60	-	mcd
Peak Emission Wavelength High Efficiency Red	λ_p	IF = 20mA	-	625	-	nm
Yellow-Green			-	570	-	nm
Dominant Wave Length (Note 2) High Efficiency Red	$\lambda_d(\text{HUE})$	IF = 20mA	-	618	-	nm
Yellow-Green			-	567	-	nm



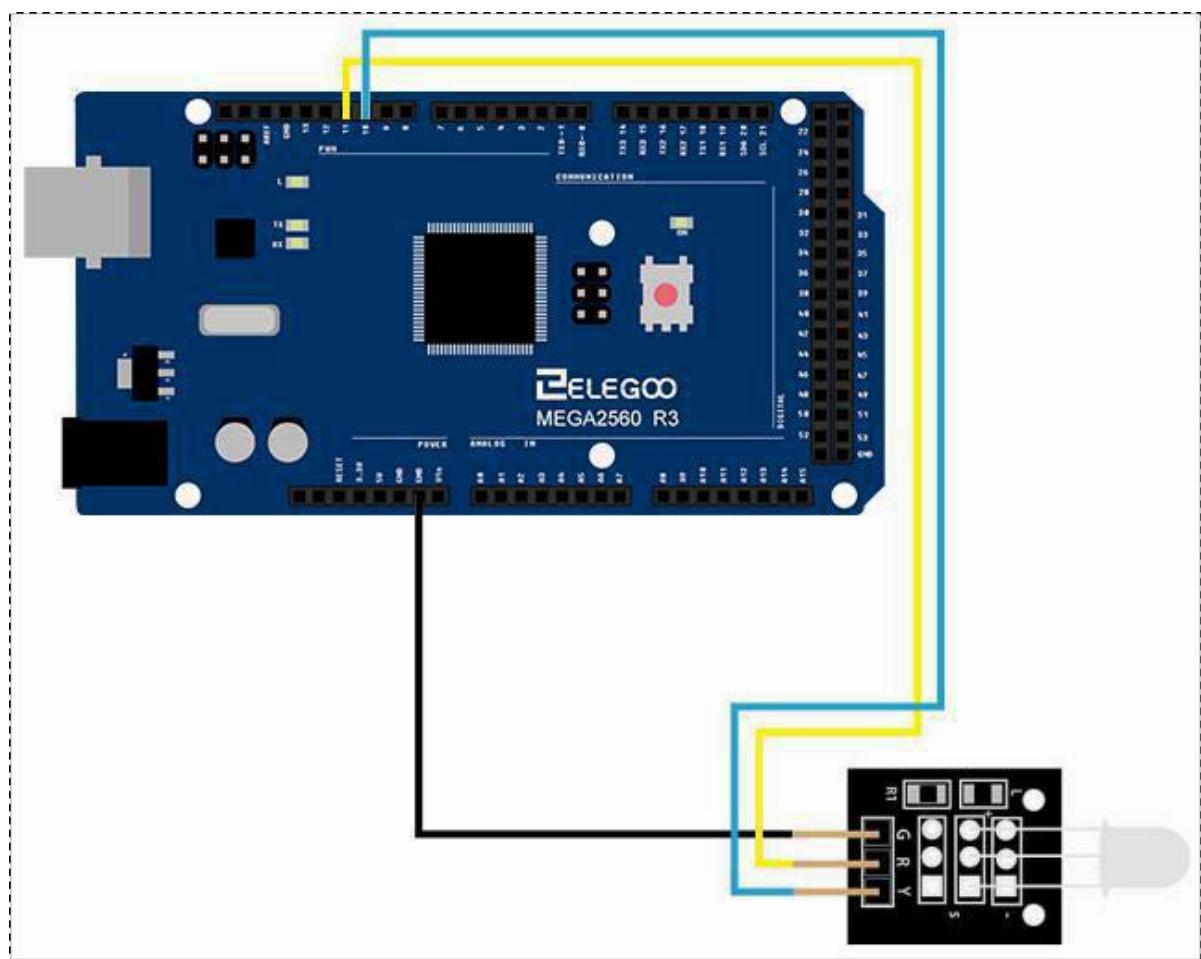
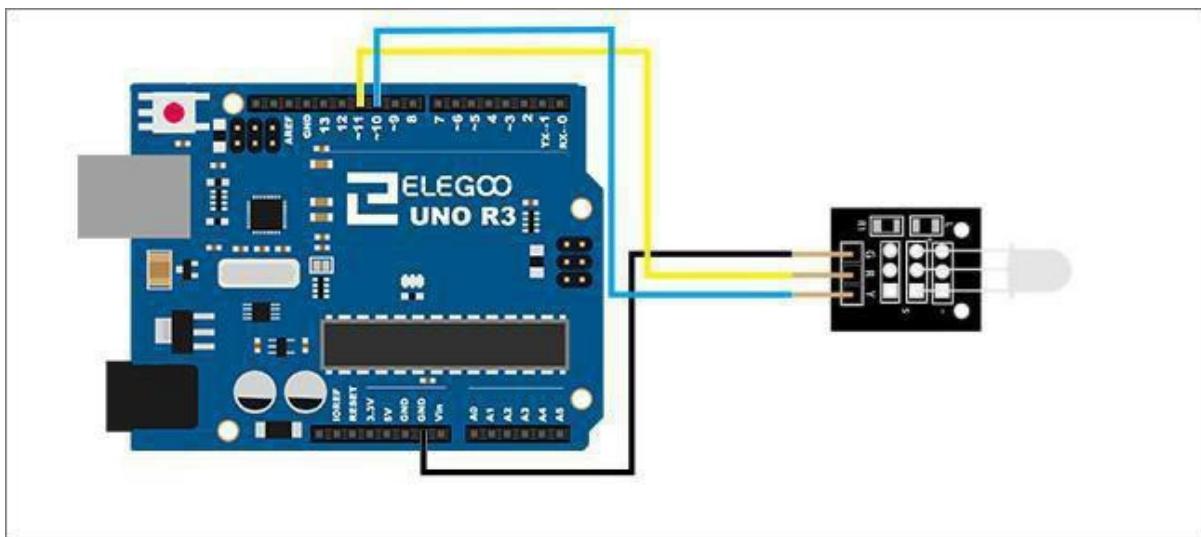
1. Red +
2. Common Lead -
3. Green +

Connection

Schematic



Wiring diagram

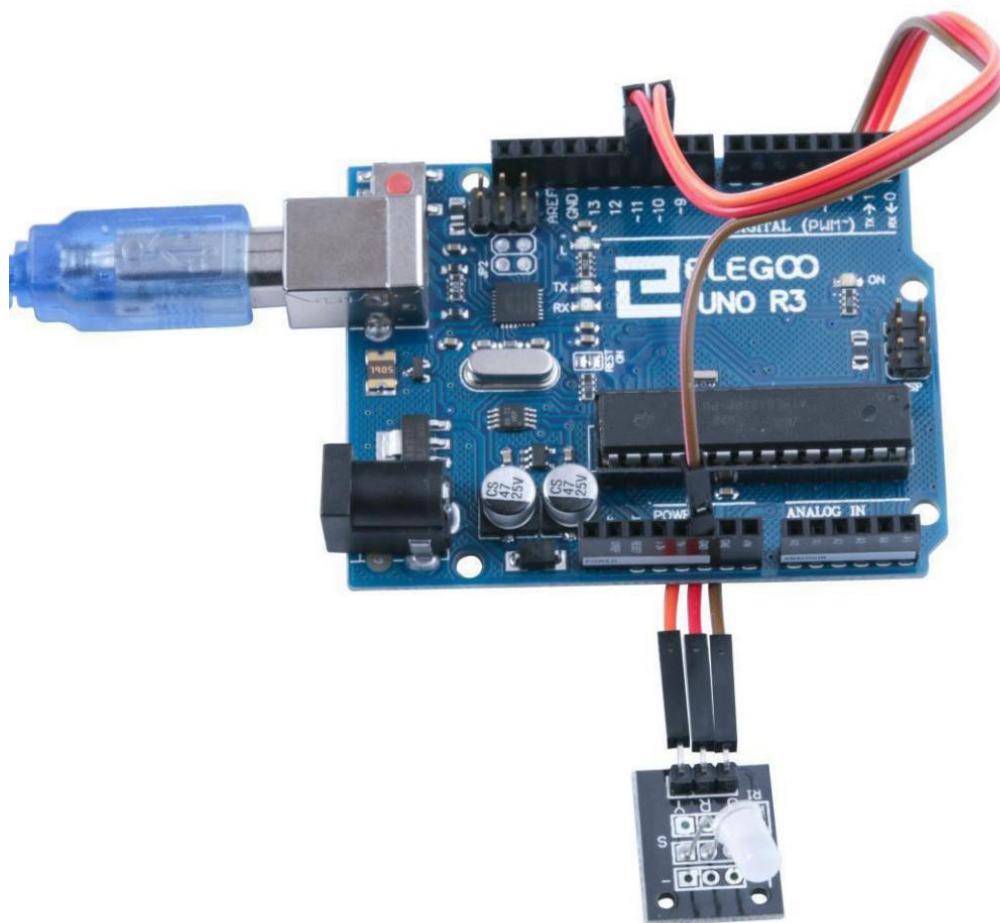


Code

After we connect the circuit, we open the "code" folder in our tutorials and find the "Lesson 14 Two Color LED Module" folder to open the program runtime loader.

We can see that the module changes their color as a code set. If you want to change color in different ways, you can modify the code.

Example picture



The followings are the code used in this experiment and their explanations:

```
/* select the pin for the red LED */
int redpin = 11;

/* select the pin for the blue LED */
int yellowpin = 10;

/*Define an integer variable val*/
int val;

void setup()
{
    /*Defined redpin as the output type*/
    pinMode(redpin, OUTPUT);

    /*Yellowpin is defined as the output type*/
    pinMode(yellowpin, OUTPUT);

    /*Set the baud rate to 9600*/
    Serial.begin(9600);
}

void loop()
{
    for (val = 255; val > 0; val--)
    {
        analogWrite(11, val);

        /*By val variable numerical changes to change the LED light color*/
        analogWrite(10, 255 - val);

        delay(15);
    }

    for (val = 0; val < 255; val++)
    {
        analogWrite(11, val);

        /*By val variable numerical changes to change the LED light color*/
        analogWrite(10, 255 - val);

        delay(15);
    }

    Serial.println(val, DEC);
}
```

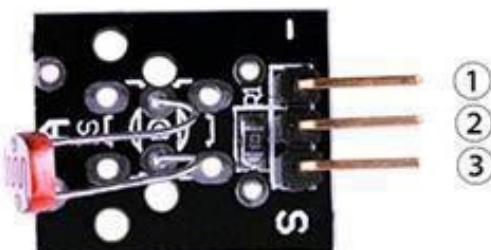
Lesson 15 Light Dependent Resistor Module

Overview

In this experiment, we will learn how to use the light dependent resistor, also called photo resistor module.

Light dependent resistor is very common in our daily life. It is mainly used in intelligent switch so as to bring convenience to our life. At the same time, in our daily life, we also use it in electronic design. So in order to use it in a better, we provide the corresponding modules to help us to use it more conveniently and efficiently.

LDR (Light Dependent Resistor). Dark resistance >20M Ohm, light <80 Ohm. The two outer pins connect to the LDR. A fixed 10 K ohm resistor connected between the middle pin and the 'S' pin is included on the module. This simplifies the building of a measurement bridge circuit.



- 1.GND:ground
- 2.VCC:3.3V-5V DC
- 3.OUTPUT

Component Required:

1 x Elegoo Uno R3

1 x USB cable

1 x Photo resistor module

3 x F-M wires

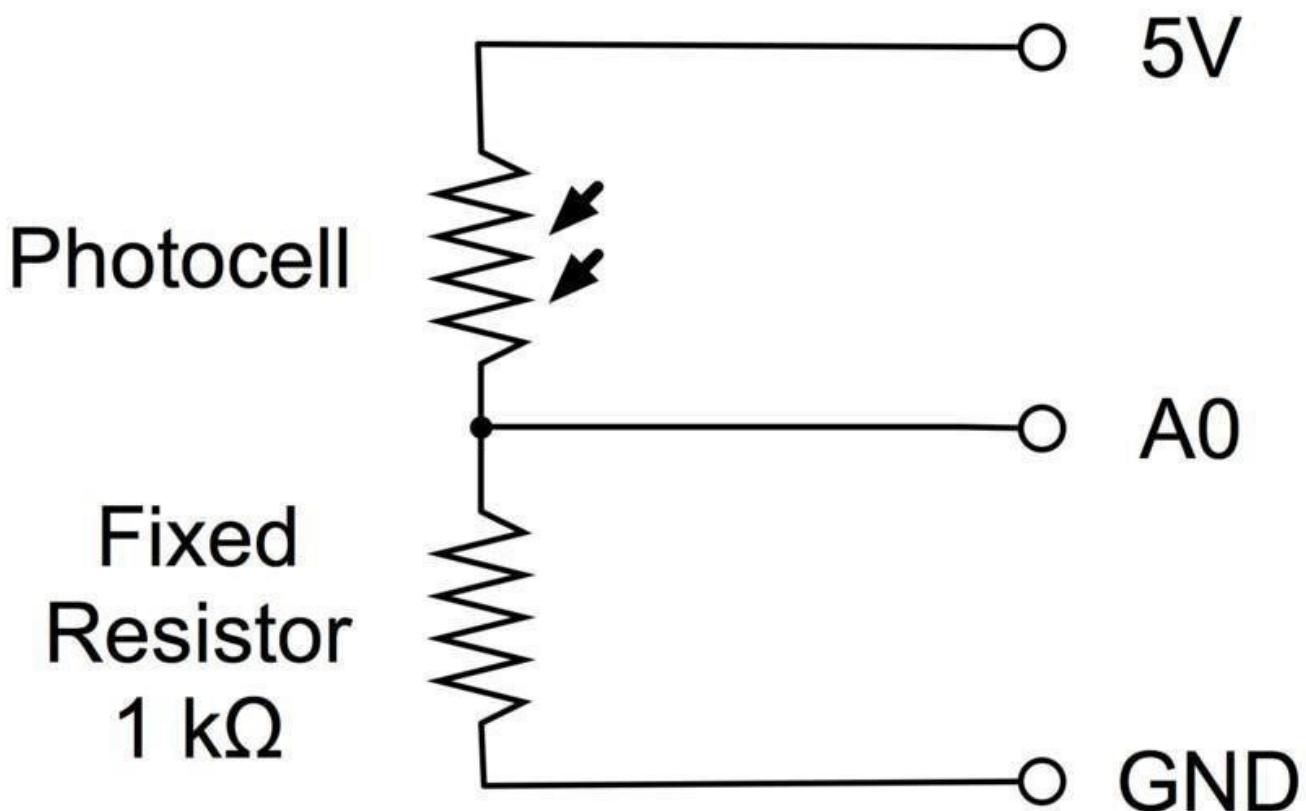
Component Introduction

Light dependent resistor:

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 $50\text{ k}\Omega$ in near darkness and $500\text{ }\Omega$ in bright light. To convert this varying value of resistance into something we can measure on an Arduino'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.



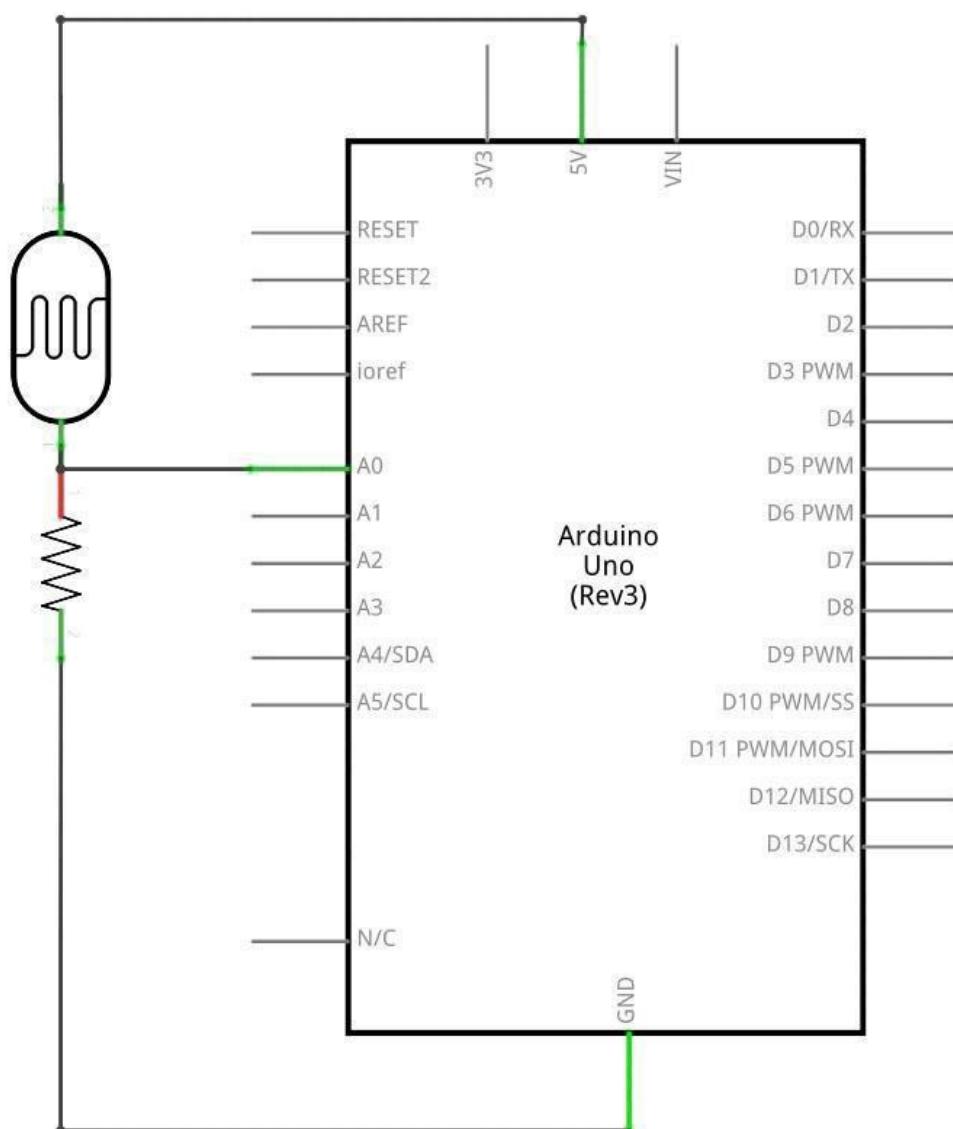
The resistor and photocell together behave rather like a pot. When the light is very bright, then the resistance of the photocell is very low compared with the fixed value resistor, and so it is as if the pot were turned to maximum.

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

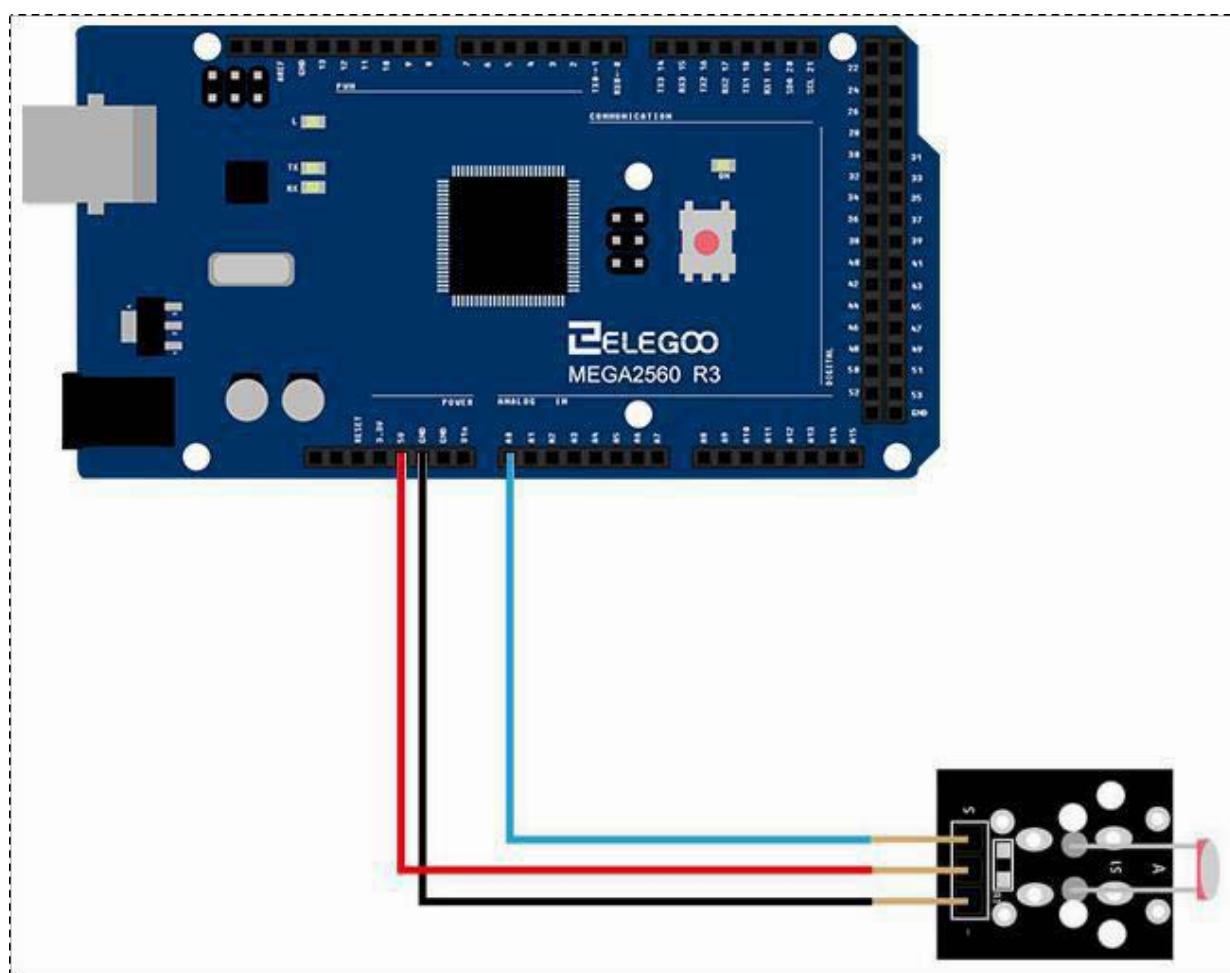
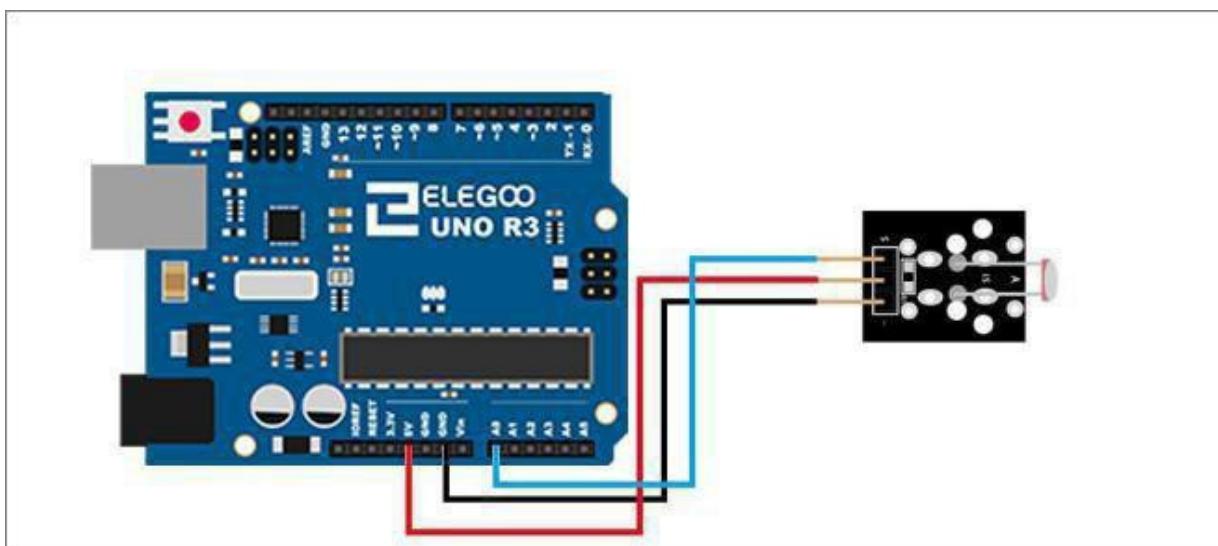
Load up the sketch given in the next section and try covering the photocell with your finger, and holding it near a lightsource.

Connection

Schematic

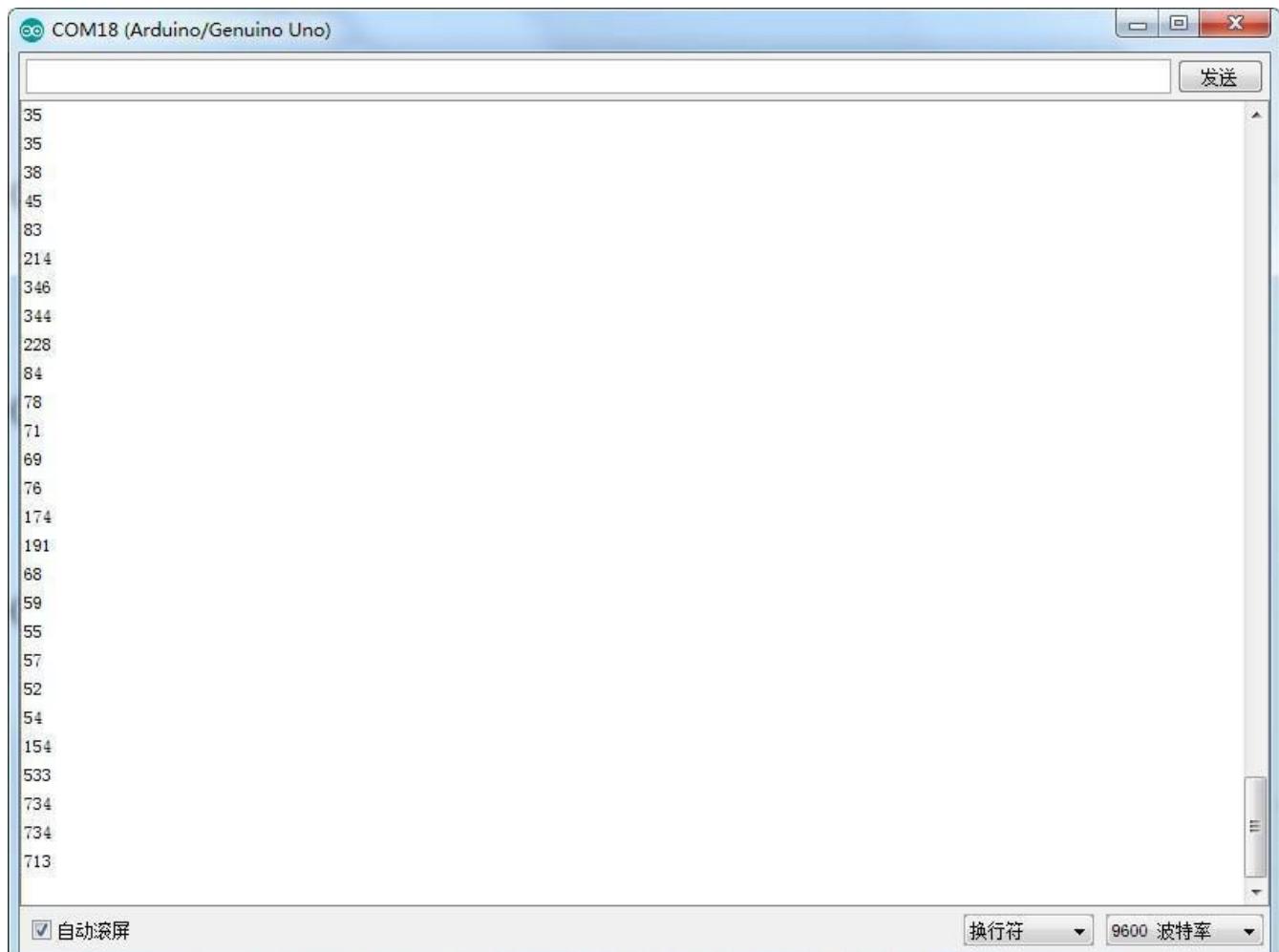


Wiring diagram



Code

After you finish the wiring, open the tutorial file and go to code > Lesson 15 PHOTO RESISTOR MODULE and run the program (the .ino file). Recall from the lesson 2 how to open the serial monitor and find the following data in it.

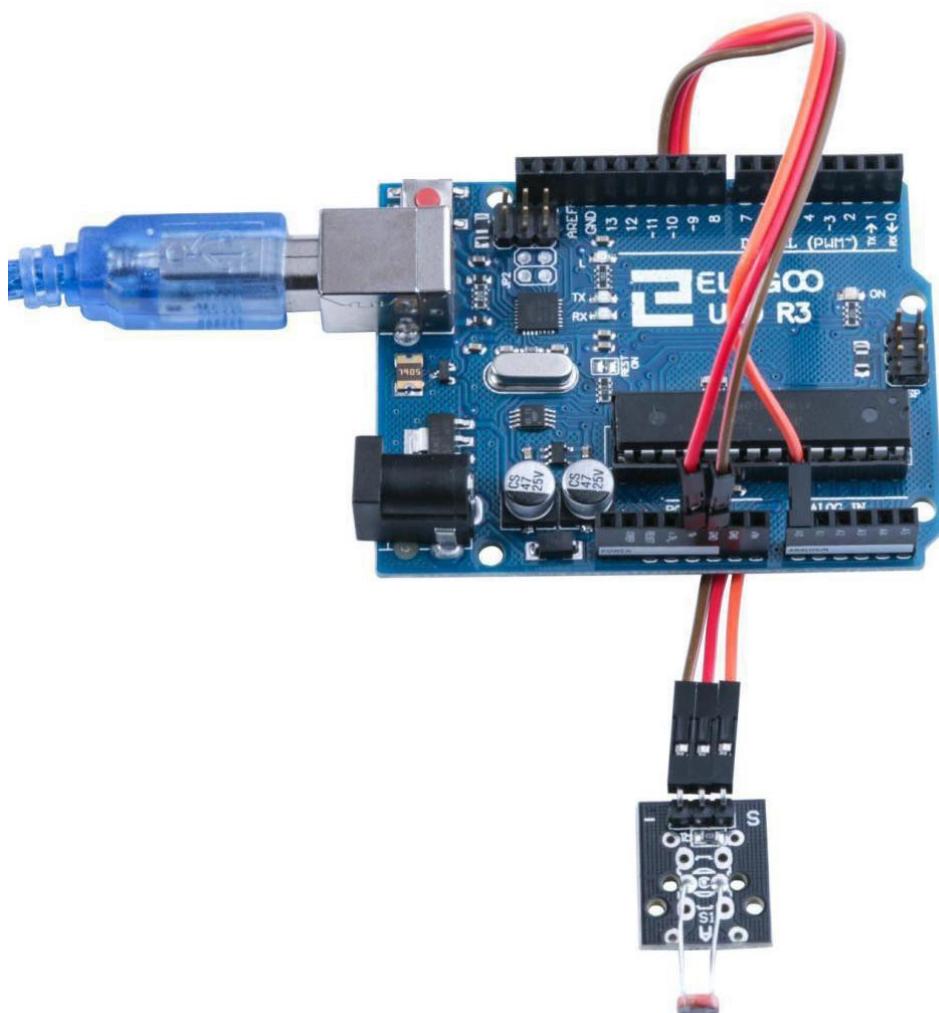


In the test, we only read the output analog voltage value of photo-resistor module.

In the test

Results , we will find that when there is lighting, high voltage output equivalently of switch on, when there is no light, low voltage equivalently of switch off. This is what we can use this in practice.

Example picture



The followings are the code used in this experiment and their explanations:

```
/* select the input pin for the potentiometer */
int sensorPin = A0;
/* select the pin for the LED */
int ledPin = 13;
/* variable to store the value coming from the sensor */
int sensorValue = 0;

void setup ()
{
    /*Define ledPin as output type*/
    pinMode(ledPin,OUTPUT);
    /*Set the baud rate to 9600*/
    Serial.begin(9600);
}

void loop()
{
    /*Read sensorPin value assigned to sensorValue*/
    sensorValue = analogRead (sensorPin);

    digitalWrite(ledPin, HIGH);
    /*Delay sensor stores the value*/
    delay(sensorValue);
    digitalWrite(ledPin, LOW);
    delay(sensorValue);
    Serial.println(sensorValue, DEC);
}
```

Lesson 16 Large Microphone Module and Small Microphone Module

Overview

In this experiment, we will learn how to use the High-sensitive Voice Sensor.

Large microphone module

A microphone module featuring a high-sensitivity large-format electret capsule.

Output 'DO' (active high) is switched when the sound level exceeds a preset level. A pot allows adjustment of the level.



- 1.DO:digital output
- 2.VCC: 3.3V-5V DC
- 3.GND:ground
- 4.AO:analog output

Small microphone module

A microphone module with a small electret capsule. Output 'DO' (active high) is switched when the sound level exceeds a preset level. A pot allows adjustment of the level. Except for the smaller size of the capsule and its lower sensitivity the module is identical to the 'Big Sound' module.



- 1.D0:digital output
- 2.VCC: 3.3V-5V DC
- 3.GND:ground
- 4.AO:analog output

Component Required:

- 1 x Elegoo Uno R3
- 1 x USB cable
- 1 x large microphone module
- 1 x Small microphone module
- 4 x F-M wires

Component Introduction

Sound sensor:

The sound sensor module provides an easy way to detect sound and is generally used for detecting sound intensity. This module can be used for security, switch, and monitoring applications. Its accuracy can be easily adjusted for the convenience of usage. It uses a microphone which supplies the input to an amplifier, peak detector and buffer. When the sensor detects a sound, it processes an output signal voltage which is sent to a microcontroller then performs necessary processing.



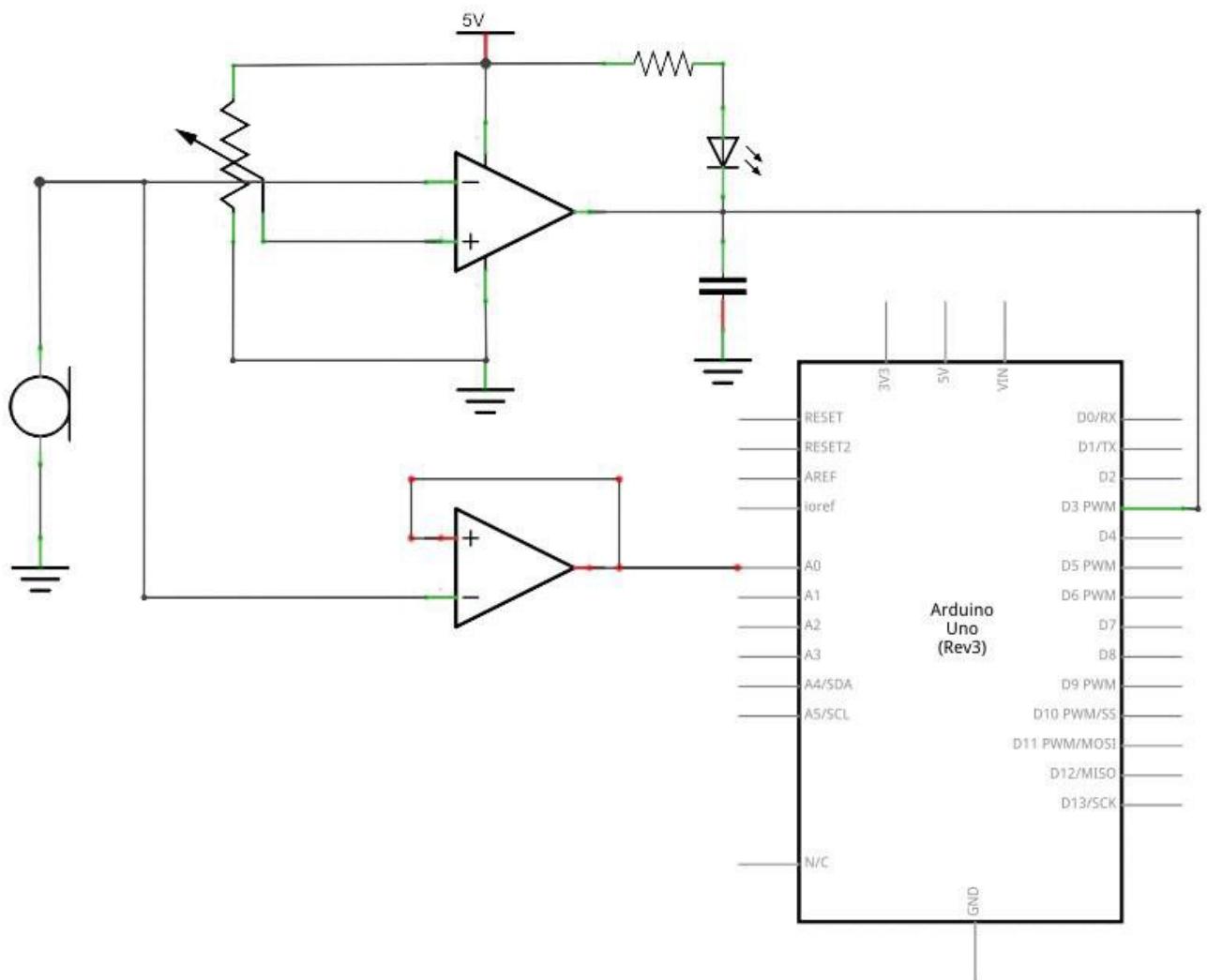
These microphones are widely used in electronic circuits to detect minor sounds or air vibrations which in turn are converted to electrical signals for further use. The two legs as shown in the image above are used to make electrical connection with the circuit.

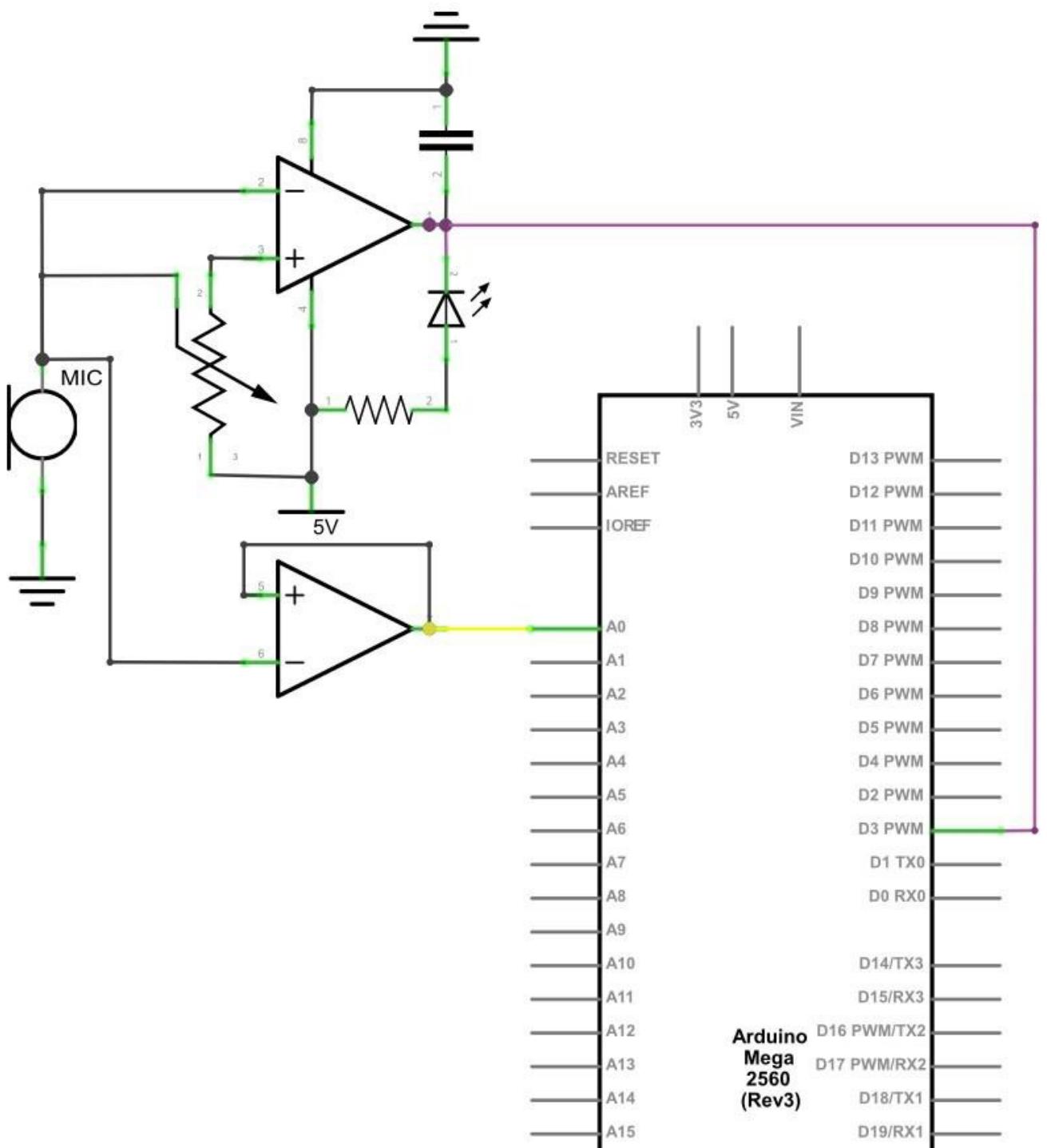


A solid conducting metal body encapsulates the various parts of the microphone. The top face is covered with a porous material with the help of glue. It acts as a filter for the dust particles. The sound signals/air vibrations passes through the porous material and falls on the diaphragm through the hole shown in the image above.

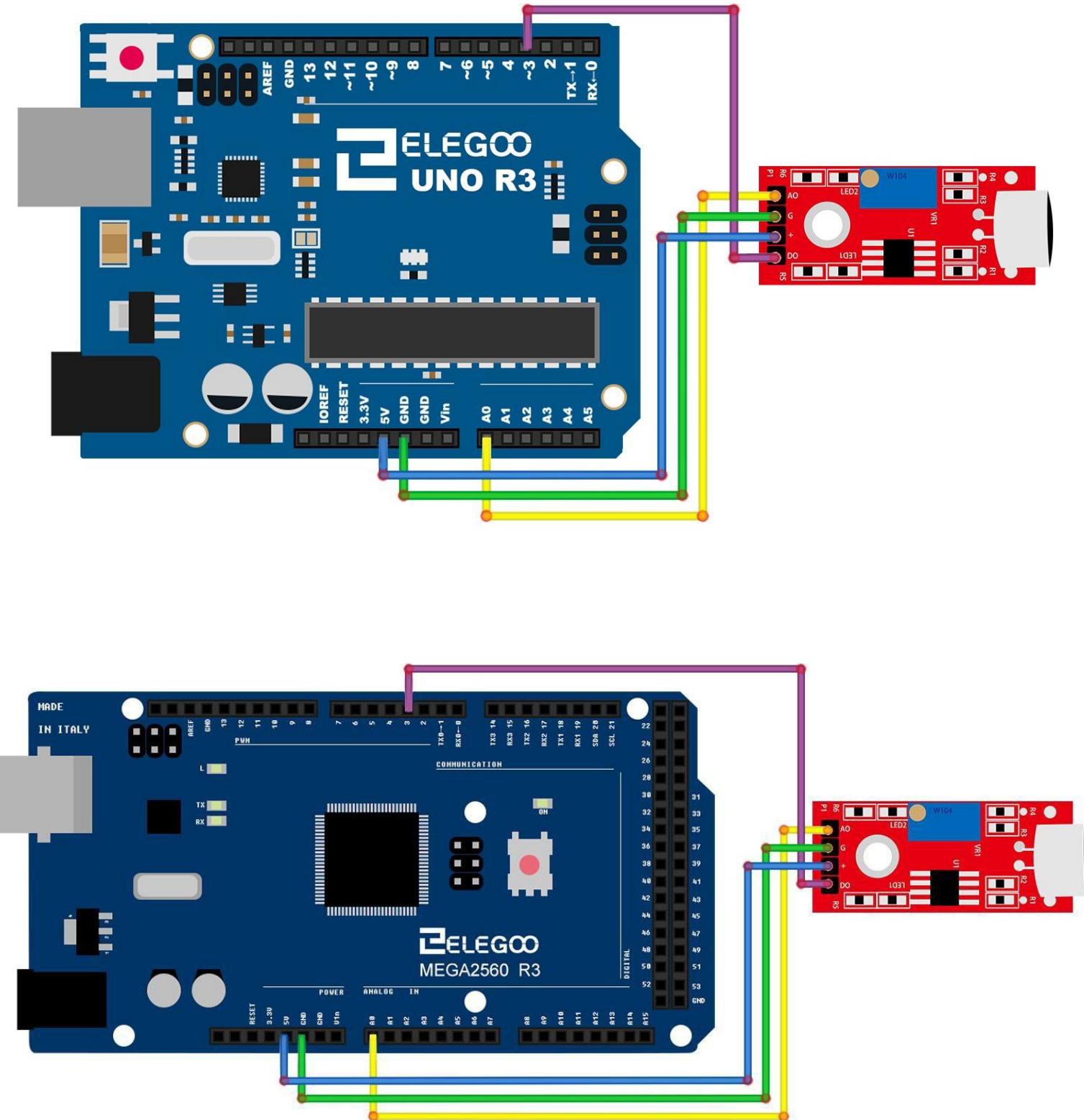
Connection

Schematic





Wiring diagram



Code

After wiring, please open the program in the code folder(Lesson 17 BIG SOUND SENSOR MODULE AND SMALL SOUND SENSOR MODULE)and click UPLOAD to upload the program.

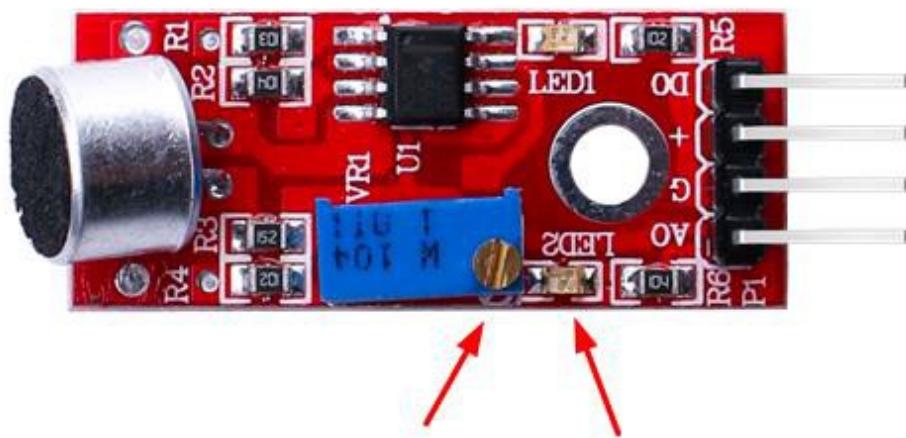
See Lesson 3 for details about program uploading if there are any errors.

High-sensitive Voice Sensor has two output:

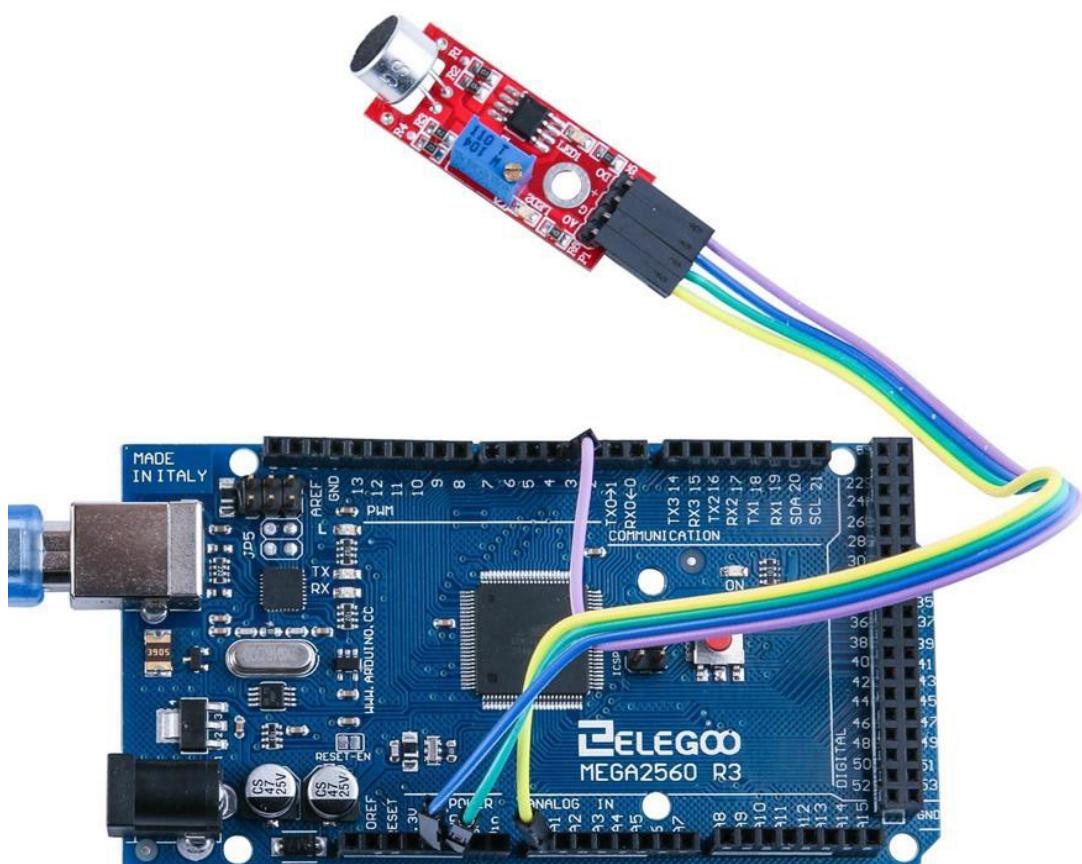
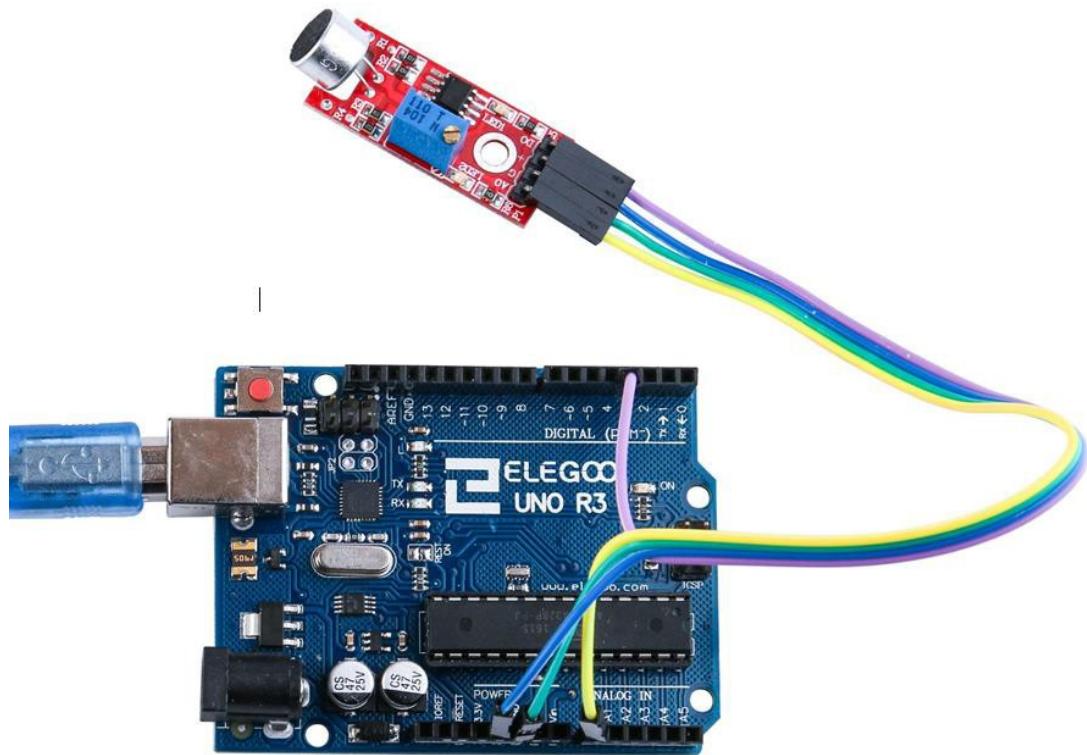
AO: analog output, real-time output voltage signal of microphone

DO: digital output when the intensity of the sound to reach a certain threshold, the output high and low level signal, the threshold-sensitivity can be achieved by potentiometer adjustment period.

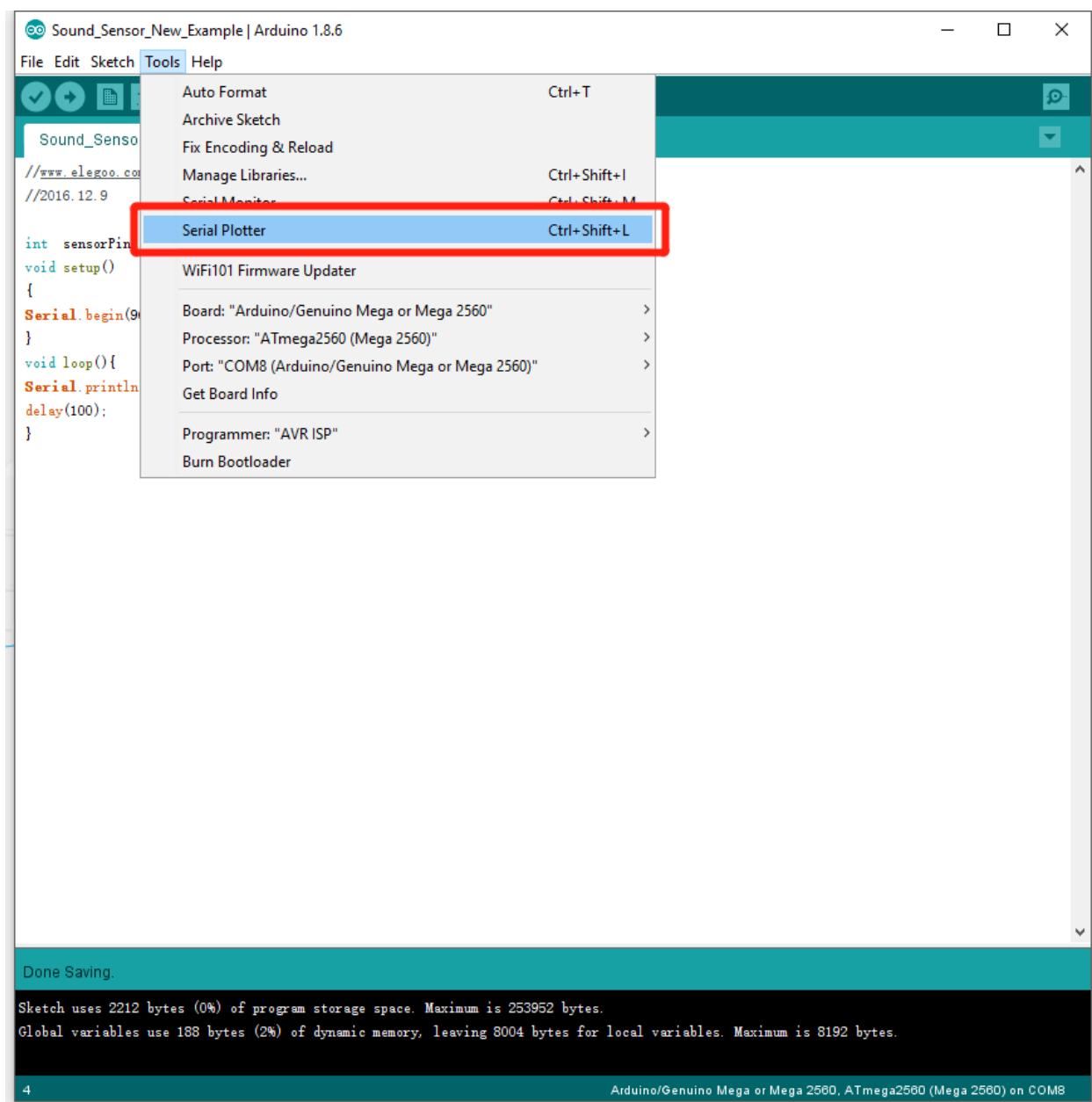
Please note that you need to spin the screw counterclockwise by a screwdriver until the LED 2 goes out, and then use the screwdriver to adjust the 10K potentiometer.



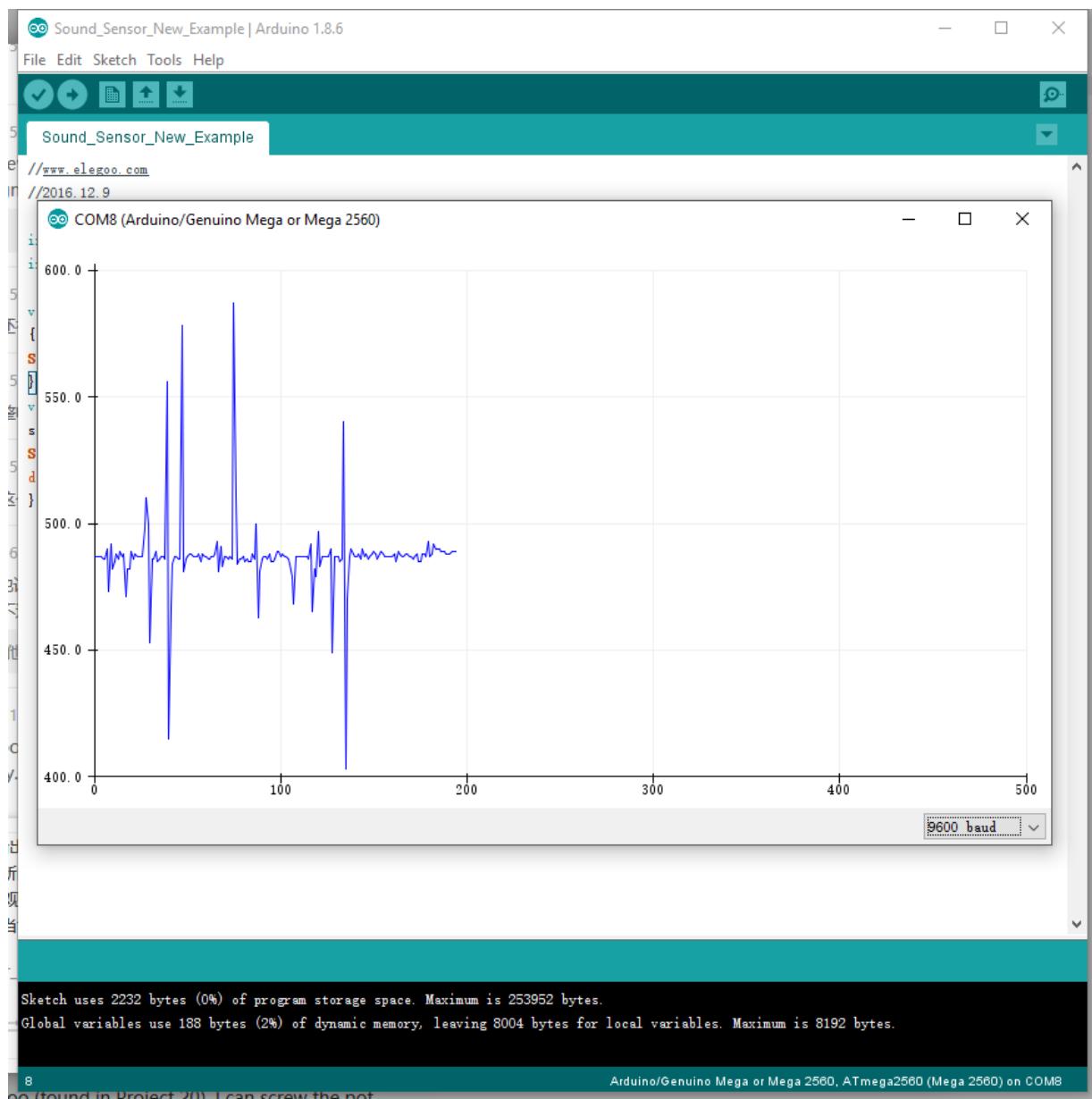
Example picture



Open Serial Plotter:



Serial Plotter Example:



When you speak into the microphone or inflate, you can observe that our waveforms have changed

The following is the procedure required to use this experiment and notes to explain:

```
void setup()
{
    Serial.begin(9600);          // The IDE settings for Serial Monitor/Plotter (preferred) must match this
speed
    pinMode(sensorDigitalPin,INPUT); // Define pin 7 as an input port, to accept digital input
    pinMode(Led13,OUTPUT);        // Define LED13 as an output port, to indicate digital trigger reached
}

void loop(){
    analogValue = analogRead(sensorAnalogPin);
                    // Read the value of the analog interface A0 assigned to digitalValue
    digitalValue=digitalRead(sensorDigitalPin);
                    // Read the value of the digital interface 7 assigned to digitalValue
    Serial.println(analogValue);
                    // Send the analog value to the serial transmit interface

    if(digitalValue==HIGH)      // When the Sound Sensor sends signla, via voltage present, light LED13 (L)
    {
        digitalWrite(Led13,HIGH);
    }
    else
    {
        digitalWrite(Led13,LOW);
    }

    delay(50);                // Slight pause so that we don't overwhelm the serial interface
}
```

Lesson 17 Reed Switch Module

Overview

In this experiment, we will learn how to use reed switch and mini reed switch module.

Reed switch module

This reed switch offers an analog as well as a digital interface. The 'G' pin connected to GND, the '+' pin to 5V DC, the 'AO' pin offers the analog output while the 'DO' offers the digital output. A potentiometer is used as a pull up resistor.



Component Required:

1x Elegoo Uno R3

1x USB cable

1x Reed switch module

4 x F-M wires

Component Introduction

Reed Switch and Reed Sensor Activation:

Although a reed switch can be activated by placing it inside an electrical coil, many reed switches and reed sensors are used for proximity sensing and are activated by a magnet. As the magnet is brought into the proximity of the reed sensor/switch, the device activates. As the magnet is removed from the proximity of the reed sensor/switch, the device deactivates. However, the magnetic interaction involved in activating the reed switch contacts is not necessarily obvious. One way of thinking about the interaction is that the magnet induces magnetic poles into the metal parts of the reed switch and the resulting attraction between the electrical contacts causes the reed switch to activate.

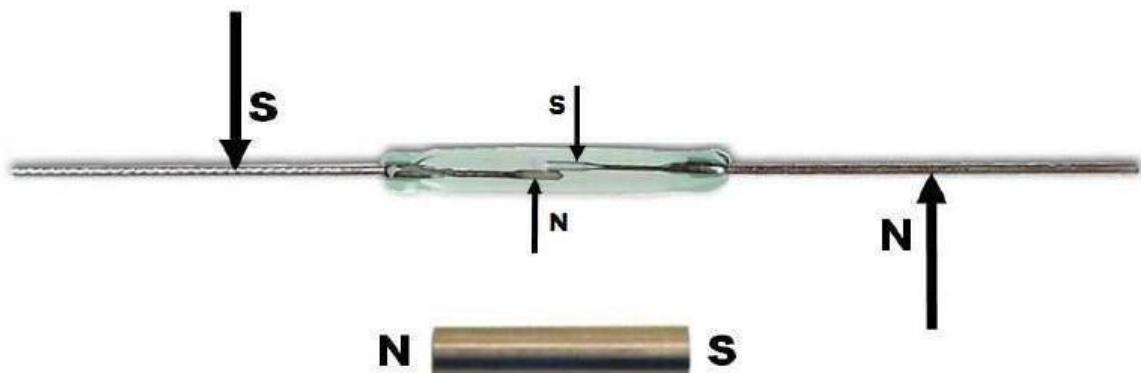


Figure 1 – Magnetic Induction

Another equally valid way of thinking about the interaction between a magnet and a reed switch is that the magnet induces magnetic flux through the electrical contacts. When the magnetic flux is high enough, the magnetic attraction between the contacts causes the reed switch to close.

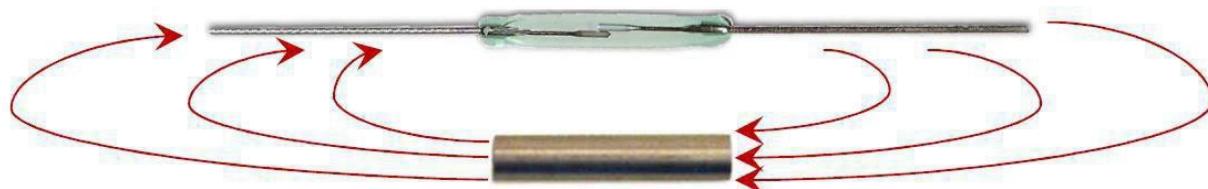


Figure 2 – Magnetic Flux

The following are examples of typical reed switch and reed sensor activate distances.

Difference between the reed switch module and mini reed switch module

As we can see, the reed switch module is bigger than the mini reed switch module. So the bigger one may have more function than the mini one. The reed switch can output in two ways: digital and analog. The mini reed can only output in digital.

In the 37 sensor kit, there have 7 red pcb modules. The difference between the red and small pcb is same as above.

Principle

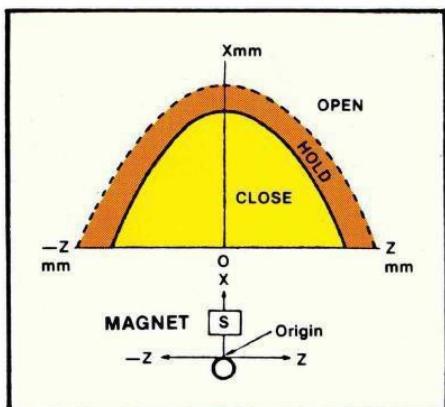


Figure 3 – Magnet Parallel to Reed Sw.

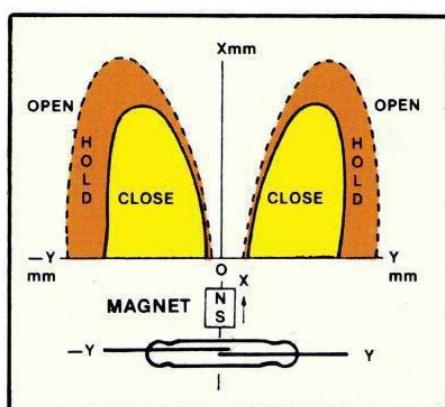


Figure 4 – Magnet Perpendicular to Reed Sw.

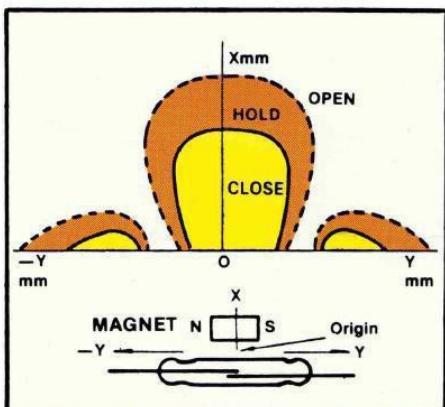
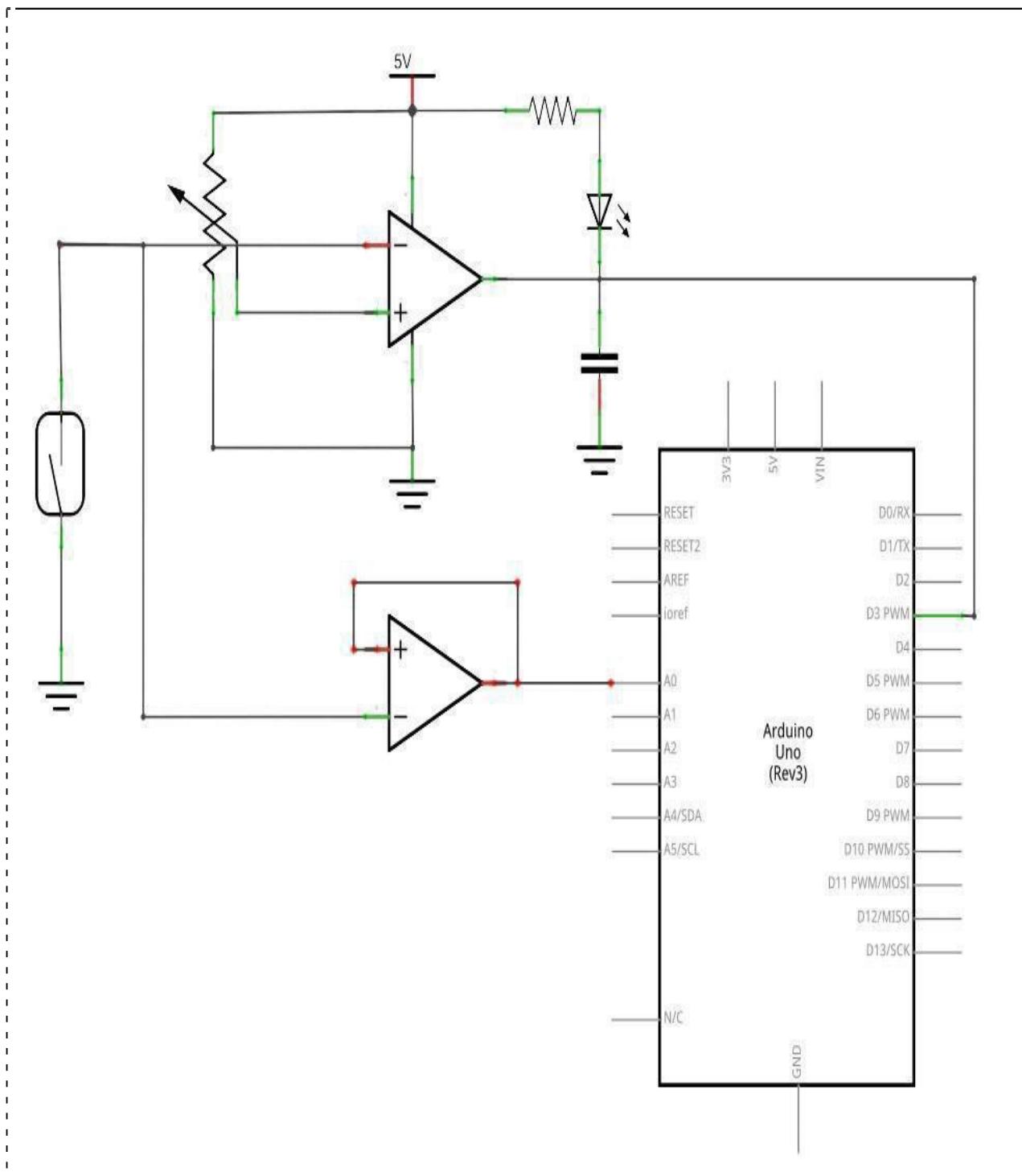


Figure 5 – Magnet Parallel to Reed Sw.

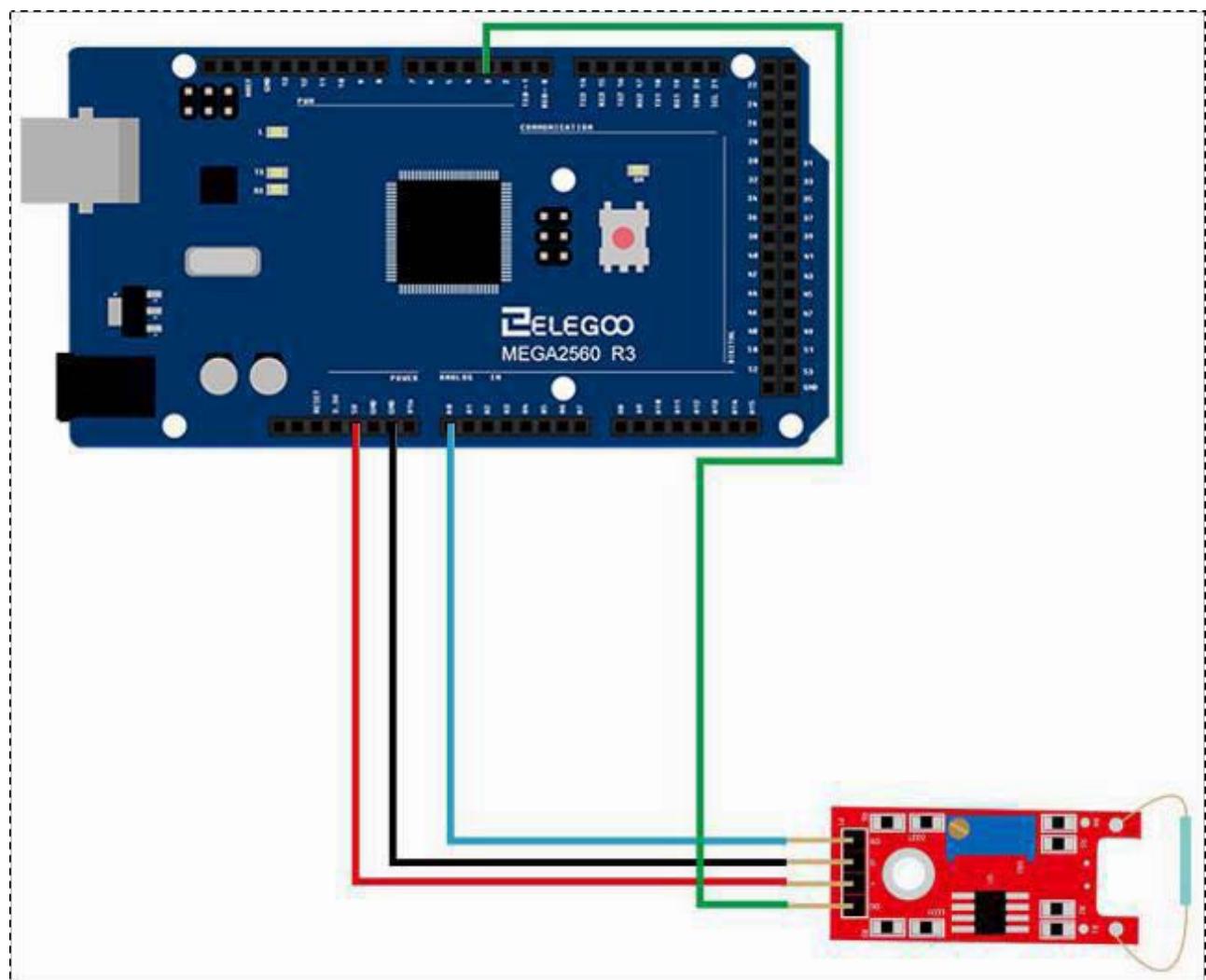
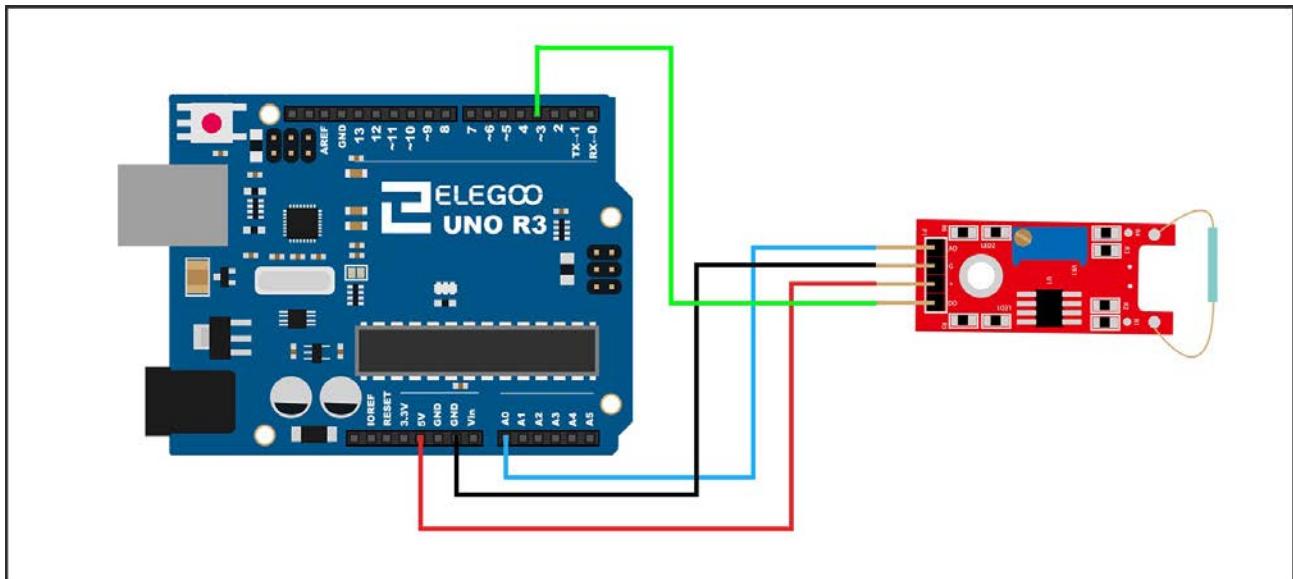
As can be seen, the magnetic orientation and location relative to the reed switch play important roles in the activation distances. In addition, the size of the activate regions(lobes) will vary depending on the strength of the magnet and the sensitivity of the reed switch. Proper orientation of the magnet with respect to the reed sensor/switch is an important consideration in meeting the application's requirements across the tolerance range for mechanical systems, magnetic strength and reed sensor or reed switch sensitivity.

Connection of reed switch module

Schematic



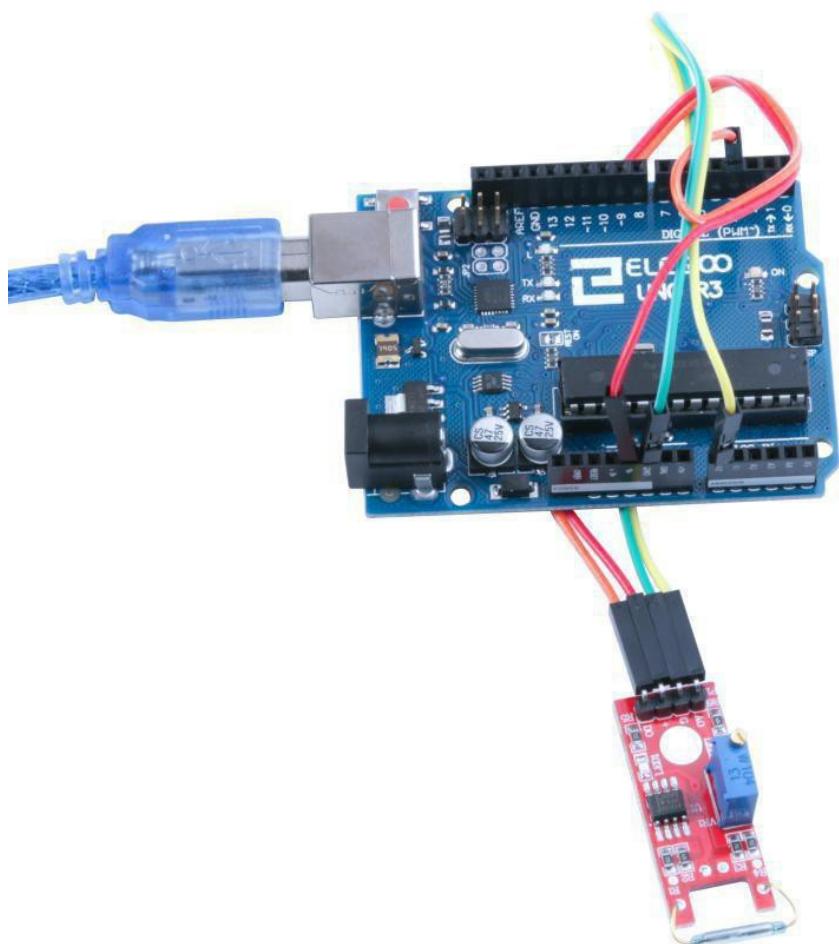
Wiring diagram

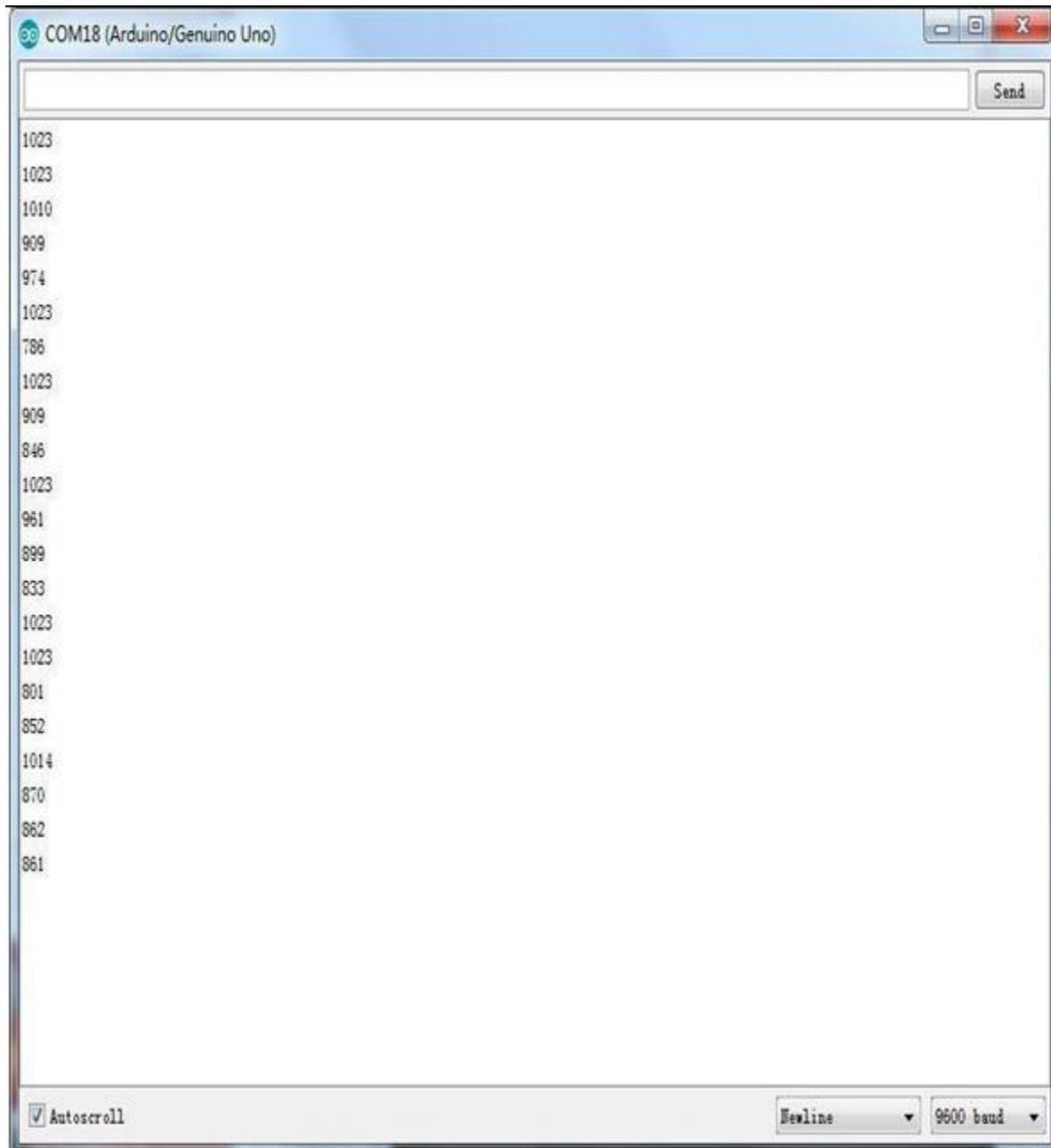


Code

After the circuit is connected, please open folder "code" in the tutorial, next open the folder "Lesson 17 REED SWITCH AND MINI REED SWITCH MODULE ", and then run the compiler loader. When the sensor senses magnetism, the module will output the data that reflect the strength of magnetism. The value is between 0 and 1023. You will see the LED flashing, then open the monitor, it will output data like the following:

Example picture





The followings are the code used in this experiment and their explanations:

(1) simulation port control program

```
// select the input pin for the potentiometer  
int sensorPin = A0;  
  
// select the pin for the LED  
int ledPin = 13;  
  
// variable to store the value coming from the sensor  
int sensorValue = 0;  
  
void setup()  
{  
    pinMode(ledPin,OUTPUT);  
    Serial.begin(9600);  
}  
  
void loop()  
{  
    sensorValue = analogRead(sensorPin);  
    digitalWrite(ledPin, HIGH);  
    delay(sensorValue);  
    digitalWrite(ledPin, LOW);  
    delay(sensorValue);  
    Serial.println(sensorValue, DEC);  
}
```

(2) digital port control program

```
/* define LED port */  
int Led=13;  
/* define switch port */  
  
int buttonpin=3;  
/* define digital variable val */  
int  val;  
void  setup()  
{  
    /* define LED as a output port */  
    pinMode(Led,OUTPUT);  
    /* define switch as a output port */  
    pinMode(buttonpin,INPUT);  
}  
void  loop()  
  
{  
    /* read the value of the digital interface 3 assigned to val */  
    val=digitalRead(buttonpin);  
    /* when the switch sensor have signal, LED blink */  
    if(val==HIGH)  
    {  
        digitalWrite(Led,HIGH);  
    }  
    else  
    {  
        digitalWrite(Led,LOW);  
    }  
}
```

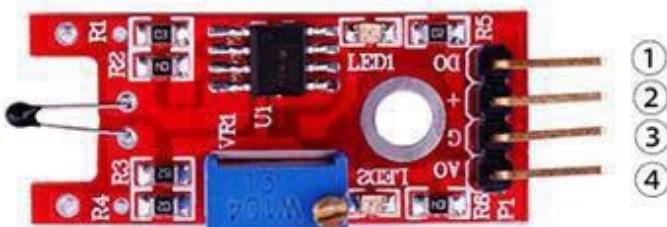
Lesson 18 Digital Temperature Module

Overview

In this experiment, we will learn how to use digital temperature module and analog temp module.

Digital Temperature Module

Temperature sensing module using an NTC thermistor. The output signal at 'DO' switches high when the preset (adjustable) temperature is reached. An analog output signal from the sensor is available at pin 'AO'.



- 1.DO:digital output
- 2.VCC: 3.3V-5V DC
- 3.GND:ground
- 4.AO:analog output

Component Required:

- 1x Elegoo Uno R3
- 1x USB cable
- 1x digital temperature module
- 4x F-M wires

Component Introduction

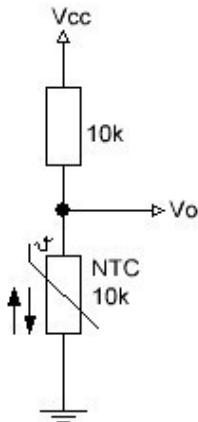
Thermistor:



What Is An NTC Thermistor

Thermistors are temperature-sensing elements made of semiconductor material that has been sintered in order to display large changes in resistance in proportion to small changes in temperature.

This resistance can be measured by using a small and measured direct current, or dc, passed through the thermistor in order to measure the voltage drop produced.



NTC Thermistors are non-linear resistors, which alter their resistance characteristics with temperature. The resistance of NTC will decrease as the temperature increases. The manner in which the resistance decreases is related to a constant known in the electronics industry as beta, or β . Beta is measured in °K.

The Arduino has several ADC ports that we can use to read a voltage, or rather an 'ADC value'. If the Analog port is connected to Vcc, the max value one reads is 1023 and of course when connected to ground it is 0.

Now if we make a voltage divider which is typically two resistors in series between Vcc and Ground and the analogport in the middle, the reading will depend on the ratio of the two resistors: If they are equal, the reading will be 512, halfway to 1023. If one of the resistors, say the bottom one is an NTC, then the readings on the analogport will vary with the temperature:

If the temperature goes down, the value of the resistor increases and so will the reading on the analog port.

Suppose we have a 10k Series resistor and an NTC that for now we call 'R'.

Then the voltage that can be measured in the middle is

$$Vo = R/(R+10K) * Vcc$$

The analogPort readings however don't give a voltage but an ADC value that can easily be calculated

ADC value= $V_o * 1023 / V_{cc}$ // if for instance the $V_o = 4$ Volt the ADC = 818

or

ADC value= $1023 * (V_o / V_{cc})$

If we now combining the two formulas or as it is called 'substitute' V_o in the formula for ADC we get the following:

ADC value= $(R / (R + 10K)) * V_{cc} * 1023 / V_{cc}$

As we multiply by V_{cc} but also divide by V_{cc} we can take that out of the equation and end up with

ADC value= $(R / (R + 10k)) * 1023$

ADC value= $1023 * R / (10 + R)$

if we want to get the value of R out of that equation, that becomes

$R = 10k / (1023 / \text{ADC} - 1)$

If that goes a bit too fast, here is the equation worked out. I prefer pictures over the 'in line' formulas as some people have problems understanding the PEMDAS / BODMAS order of operation.

$$ADC = 1023 * \frac{R}{10 + R}$$

This becomes

$$\frac{1023 * R}{ADC} = 10 + R$$

subtraction of R

$$\frac{1023 * R}{ADC} - R = 10$$

Work -R into the multiplication

$$\left(\frac{1023}{ADC} - 1 \right) * R = 10$$

As we are interested in R, divide both sides by the enclosed fracture

$$R = \frac{10}{\left(\frac{1023}{ADC} - 1 \right)}$$

$$R = \frac{10k}{\left(\frac{1023}{ADC} - 1 \right)}$$

The '10' stood for '10k'

and as we don't always use a 10k we just make it more general:

$$R_{ntc} = \frac{R_{series}}{\left(\frac{1023}{ADC} - 1 \right)}$$

So, as long as we know the value of the series resistor, we can calculate the value of the NTC from the measured ADC value. Now remember, this is valid voor a pull-up configuration. If it is a pull down configuration, the calculation of the ADC to resistor value is the inverse.

$R_{ntc} = R_{series} * (1023/ADC - 1); //$ for pull down

$R_{ntc} = R_{series} / (1023/ADC - 1); //$ for pull-up configuration

So what would that look like in a program?

//Measure NTC value

```
byte NTCPin = A0;  
const int SERIESRESISTOR = 10000;  
void setup()  
{  
    Serial.begin(9600);  
}  
void loop()
```

```
{  
    float ADCvalue;  
    float Resistance;  
    ADCvalue = analogRead(NTCPin);  
    Serial.print("Analoge ");  
    Serial.print(ADCvalue);  
    Serial.print(" = ");
```

//convert value to resistance

```
Resistance = (1023 / ADCvalue) - 1;  
Resistance = SERIESRESISTOR / Resistance;  
Serial.print(Resistance);  
Serial.println(" Ohm");  
delay(1000);  
}
```

//end program

Knowing the resistance of the NTC is nice but it doesn't tell us much about the temperature... or does it?

Well many NTC's have a nominal value that is measured at 25 degrees Centigrade, so if you have a 10k NTC and you measure it to be 10k, that means it is 25 degrees at that moment. That doesn't help you much when the measurement is different.

You could keep a table in which every resistance value stands for a temperature. Those tables are very accurate but require a lot of work and memory space.

However, there is a formula, the Steinhart-Hart equation, that does a good approximation of converting resistance values of an NTC to temperature. It's not as exact as the thermistor table (after all it is an approximation) but it's fairly accurate.

The Steinhart-Hart equation looks like this:

$$\frac{1}{T} = A + B \ln(R) + C (\ln(R))^3$$

That is a rather complex equation that requires several parameters (A, B, C) that we normally do not have for the run of the mill NTC. There are two things we can do. We can take 3 readings with a calibrated temperature and then work out the A, B and C parameters.

$$\begin{bmatrix} 1 & \ln(R_1) & \ln(R_1)^3 \\ 1 & \ln(R_2) & \ln(R_2)^3 \\ 1 & \ln(R_3) & \ln(R_3)^3 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} \frac{1}{T_1} \\ \frac{1}{T_2} \\ \frac{1}{T_3} \end{bmatrix}$$

but fortunately there is a simplification of this formula, called the B-parameter Equation. That one looks as follows:

$$\frac{1}{T} = \frac{1}{T_0} + \frac{1}{B} \ln\left(\frac{R}{R_0}\right)$$

T_0 is the nominal temperature, 25 °C in Kelvin (= 298.15 K). B is the coefficient of the thermistor (3950 is a common value). R_0 is the nominal resistance of the NTC (thus at 25 degrees). Let's say we have a 10Kohm NTC. We only need to plug in R (the resistance measured) to get T (temperature in Kelvin) which we then convert to °C.

The program looks as follows:

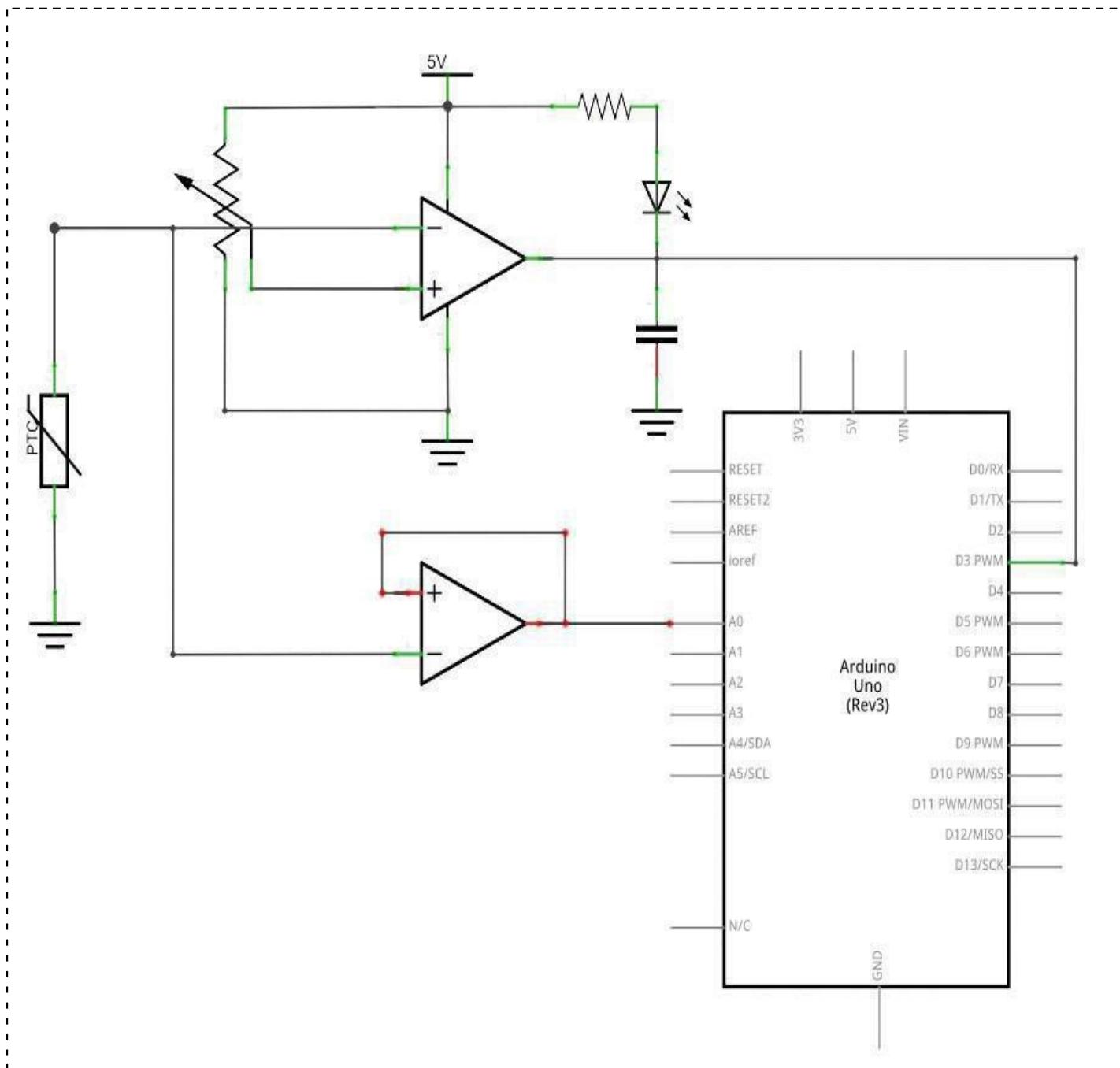
//-----

<http://www.elegoo.com>

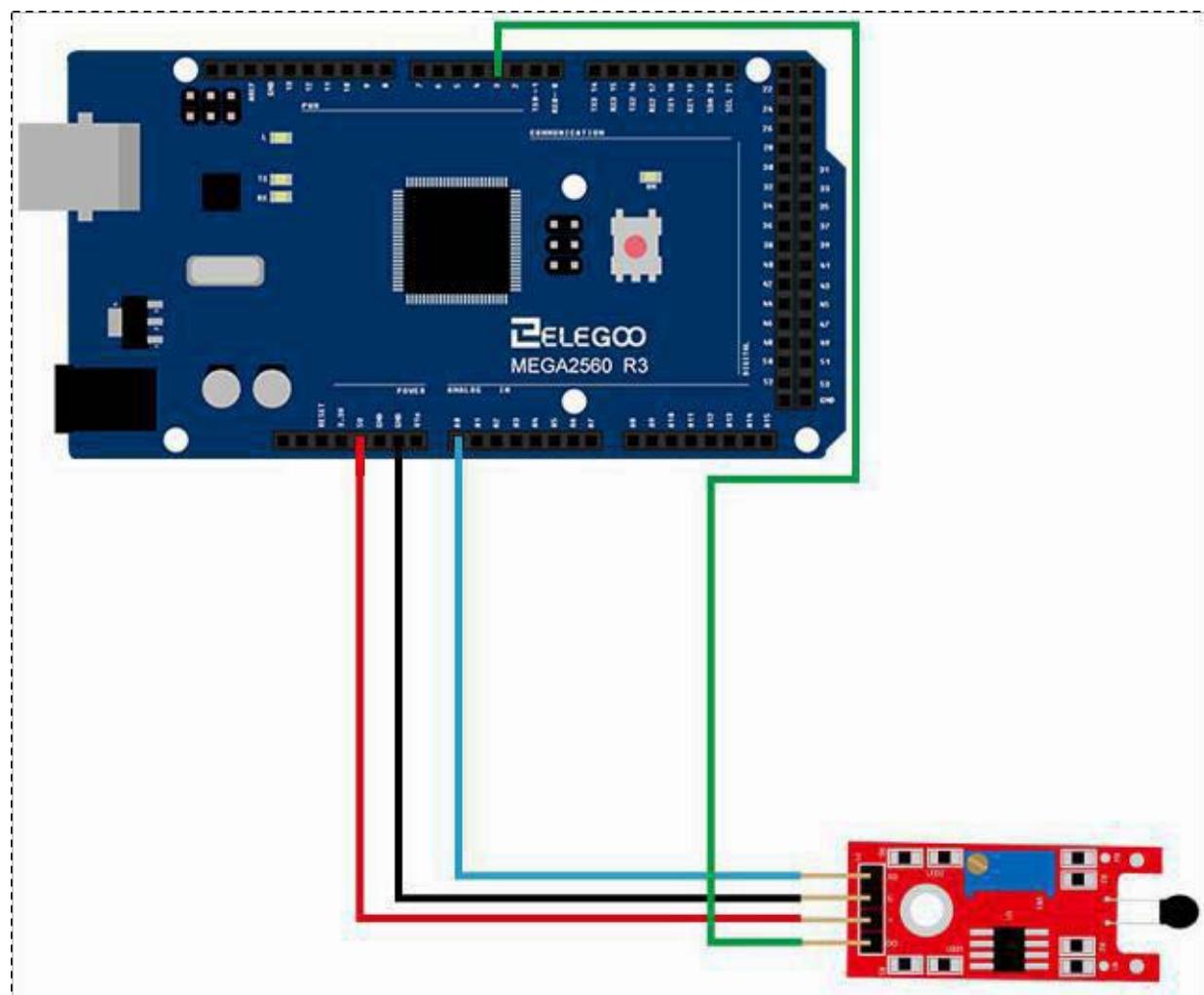
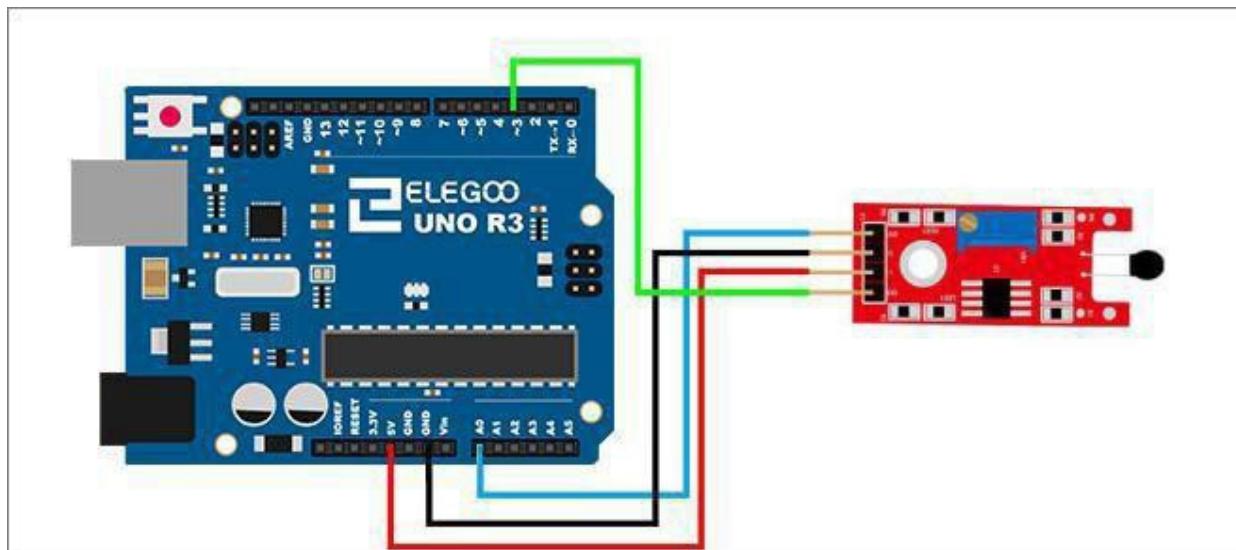
```
byt e NTCPin = A0;  
  
#define SERIESRESISTOR 10000  
  
#define NOMINAL_RESISTANCE 10000  
  
#define NOMINAL_TEMPERATURE 25  
  
#define BCOEFFICIENT 3950  
  
void setup(){  
Serial.begin(9600);  
}  
  
void loop(){  
float ADCvalue;  
  
float Resistance;  
  
ADCvalue = analogRead(NTCPin);  
  
Serial.print("Analoge ");  
Serial.print(ADCvalue);  
  
Serial.print(" = ");  
  
//convert value to resistance  
  
Resistance = (1023 / ADCvalue) - 1;  
  
Resistance = SERIESRESISTOR / Resistance;  
  
Serial.print(Resistance);  
  
Serial.println(" Ohm");  
  
float steinhart;  
  
steinhart = Resistance / NOMINAL_RESISTANCE; // (R/R0)  
  
steinhart = log(steinhart); // ln(R/R0)  
  
steinhart /= BCOEFFICIENT; // 1/B * ln(R/R0)  
  
steinhart += 1.0 / (NOMINAL_TEMPERATURE + 273.15); // + (1/T0)  
  
steinhart = 1.0 / steinhart; // Invert  
  
steinhart -= 273.15; // convert to C  
  
Serial.print ("TemperatureSerial.print(steinhart)");  
  
Serial.println(" oC");  
  
delay(1000);  
}  
  
//-----
```

Connection of temperature module

Schematic



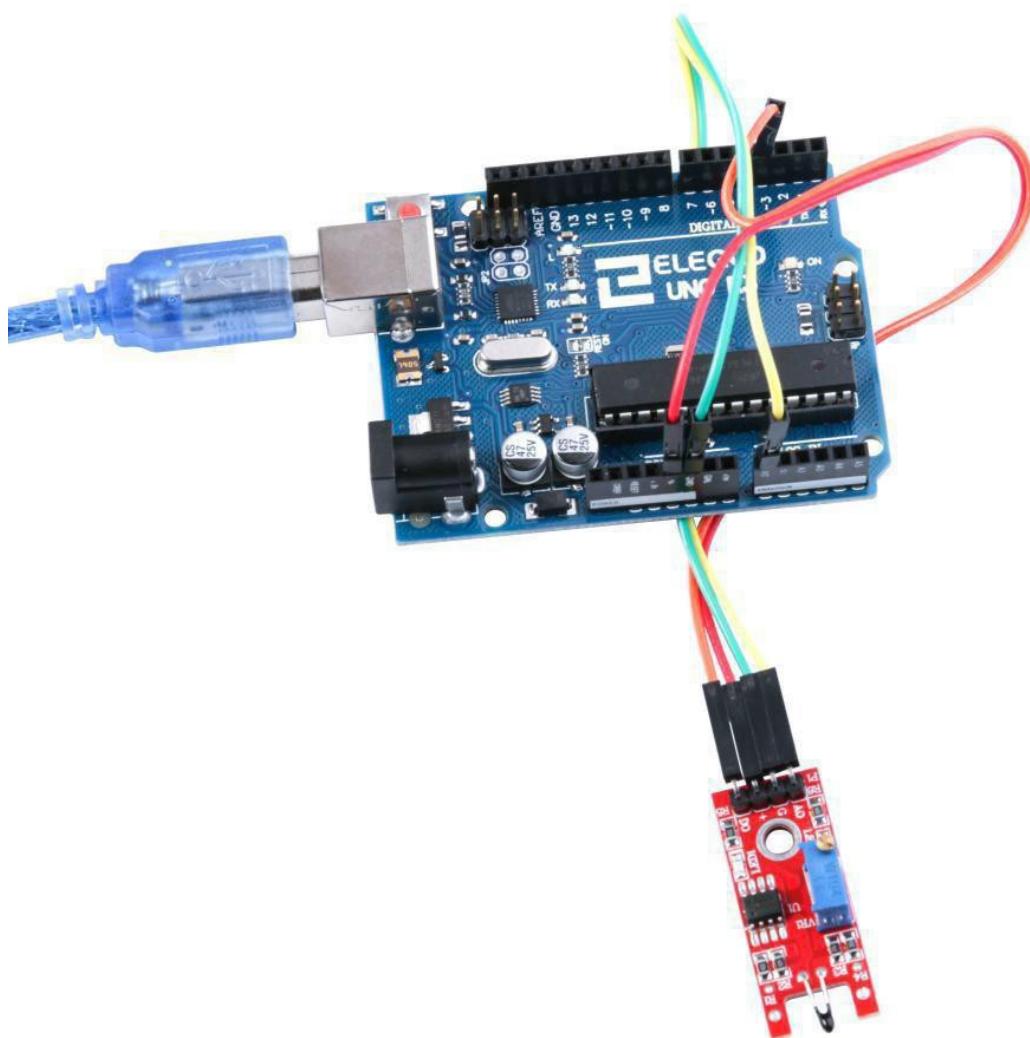
Wiring diagram

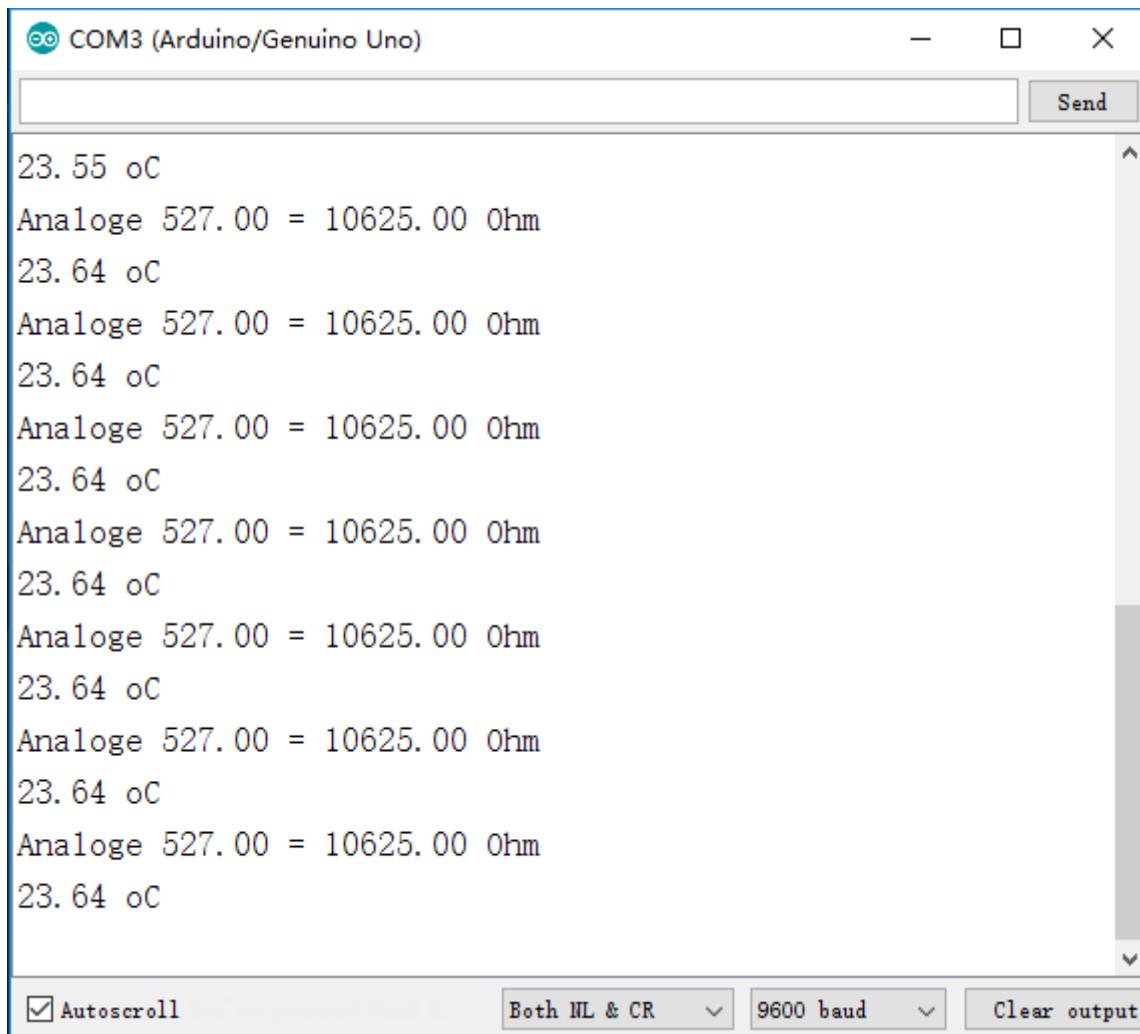


Code

After the circuit is connected, please open folder "code" in the tutorial, next open the folder "Lesson 18 DIGITAL TEMPERATURE MODULE ", and then run the compiler loader. When the sensor senses temperature, the module will output data that reflect the temperature. You will see the LED flashing, then open the monitor, it will output data like the following:

Example picture





The screenshot shows a terminal window titled "COM3 (Arduino/Genuino Uno)". The window displays a series of sensor readings. The text in the window is as follows:

```
23.55 °C
Analogue 527.00 = 10625.00 Ohm
23.64 °C
```

At the bottom of the window, there are three buttons: "Autoscroll" (checked), "Both NL & CR", "9600 baud", and "Clear output".

The followings are the code used in this experiment and their explanations:

```
byte NTCPin = A0;  
  
#define SERIESRESISTOR 10000  
  
#define NOMINAL_RESISTANCE 10000  
#define NOMINAL_TEMPERATURE 25  
#define BCOEFFICIENT 3950  
  
void setup()  
{Serial.begin(9600);  
}  
  
void loop()  
{float ADCvalue;  
float Resistance;  
ADCvalue = analogRead(NTCPin);  
Serial.print("Analoge ");  
Serial.print(ADCvalue);  
Serial.print(" = ");  
//convert value to resistance  
Resistance = (1023 / ADCvalue) - 1;  
Resistance = SERIESRESISTOR / Resistance;  
Serial.print(Resistance);  
Serial.println(" Ohm");  
float steinhart;  
steinhart = Resistance / NOMINAL_RESISTANCE; // (R/Ro)  
steinhart = log(steinhart); // ln(R/Ro)  
steinhart /= BCOEFFICIENT; // 1/B * ln(R/Ro)  
steinhart += 1.0 / (NOMINAL_TEMPERATURE + 273.15); // + (1/To)  
steinhart = 1.0 / steinhart; // Invert  
steinhart -= 273.15; // convert to C  
Serial.print(steinhart);  
Serial.println(" oC");  
delay(1000);  
}
```

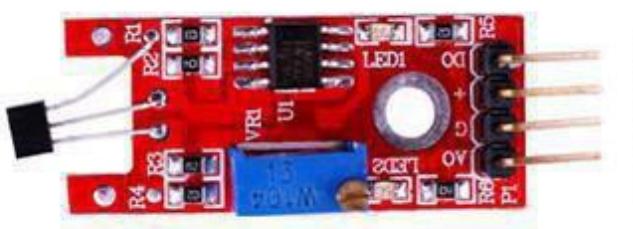
Lesson 19 Linear Magnetic Hall Sensor Module

Overview

In this experiment, we will learn how to use the linear hall sensor module.

Linear magnetic hall sensor module

Linear Hall Sensor module to detect the presence of a magnetic field near the sensor. Variables such as field strength, polarity and position of the magnet relative to the sensor will affect point at which the 'DO' output switches to a high level (i.e. active high). The circuit sensitivity can be adjusted with a pot. An analog output signal from the sensor is available at pin 'AO'.



- ①
- ②
- ③
- ④

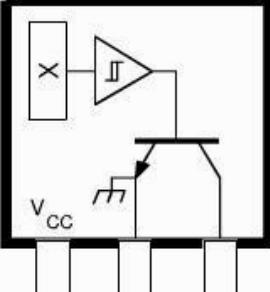
- 1.D0:digital output
- 2.VCC: 3.3V-5V DC
- 3.GND:ground
- 4.AO:analog output

Component Required:

- 1x Elegoo Uno R3
- 1x linear magnetic hall sensor module
- 1x USB cable
- 4x F-M wires

Component Introduction

Hall Sensor:

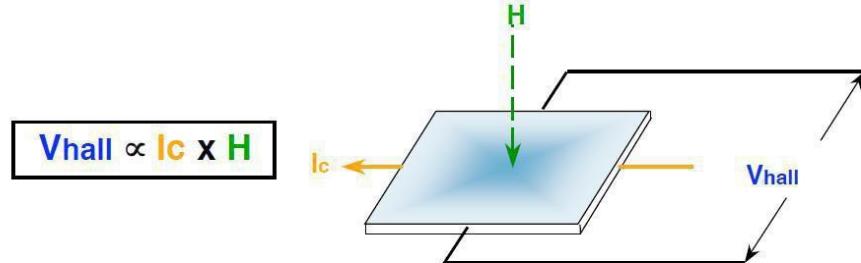


Dwg. PH-003A

Pinning is shown viewed from branded side.

**ABSOLUTE MAXIMUM RATINGS
at $T_A = +25^\circ\text{C}$**

Supply Voltage, V_{CC}	28 V
Reverse Battery Voltage, V_{RCC}	-35 V
Magnetic Flux Density, B	Unlimited
Output OFF Voltage, V_{OUT}	28 V
Reverse Output Voltage, V_{OUT}	-0.5 V
Continuous Output Current, I_{OUT}	25 mA
Operating Temperature Range, T_A	
Suffix 'E-'	-40°C to +85°C
Suffix 'L-'	-40°C to +150°C
Storage Temperature Range, T_S	-65°C to +170°C



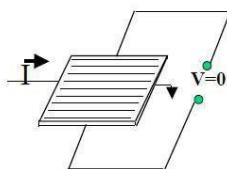
V_{Hall} = Output Hall-effect voltage

H = Magnetic Flux created by magnet or current-carrying conductor

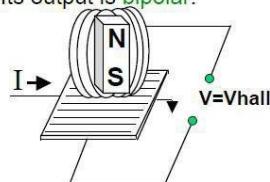
I_C = Constant supply current

Hall-effect Sensing Mechanism

- The current source is applied through a thin sheet of semiconductor material.

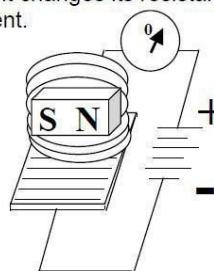


- A magnetic field applied **perpendicular** to the element creates a voltage change = V_{Hall} . Its output is **bipolar**.

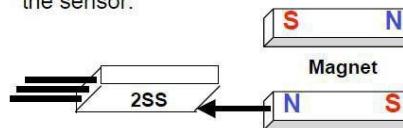


Magnetoresistive Sensing Mechanism

- A magnetic field applied **parallel** to the element changes its resistance and creates a current.



- MR is **omnipolar**—either pole will operate the sensor.



Design Factors – Magnetic Types

Unipolar: Only a south pole will operate the sensor. The sensor turns on with the south pole(+) and off when the south pole is removed.

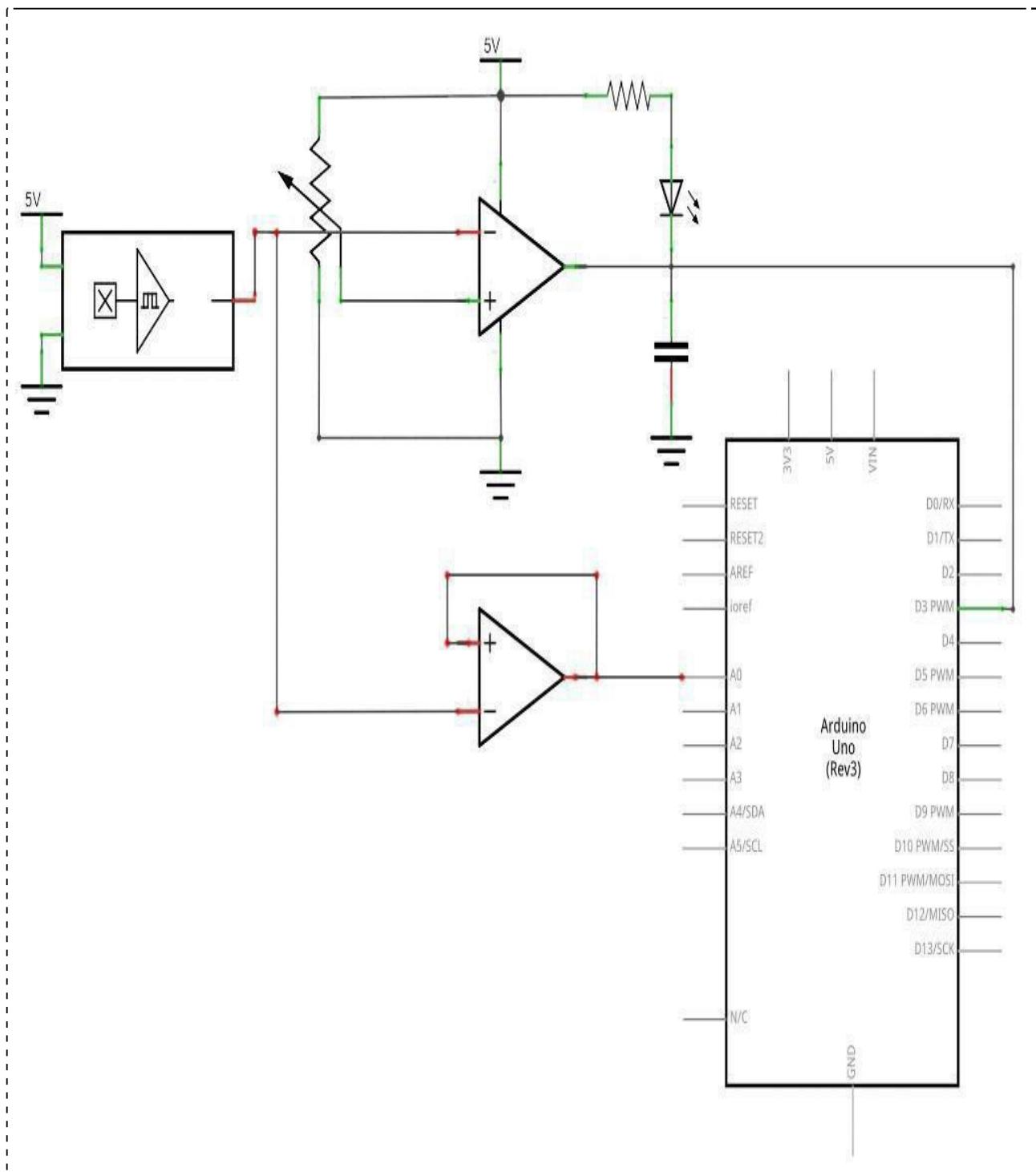
Bipolar: Sensor output is pole-dependent. A south pole (+) is designed to activate the sensor; a north pole(-) is designed to deactivate. It's possible that the sensor could turn off and still be within a positive Gauss level.

Latching: Specifications are tighter on latching. Sometimes it is designed to make certain that when the south pole(+) is removed from the sensor, it will stay on until it sees the opposite pole(-).

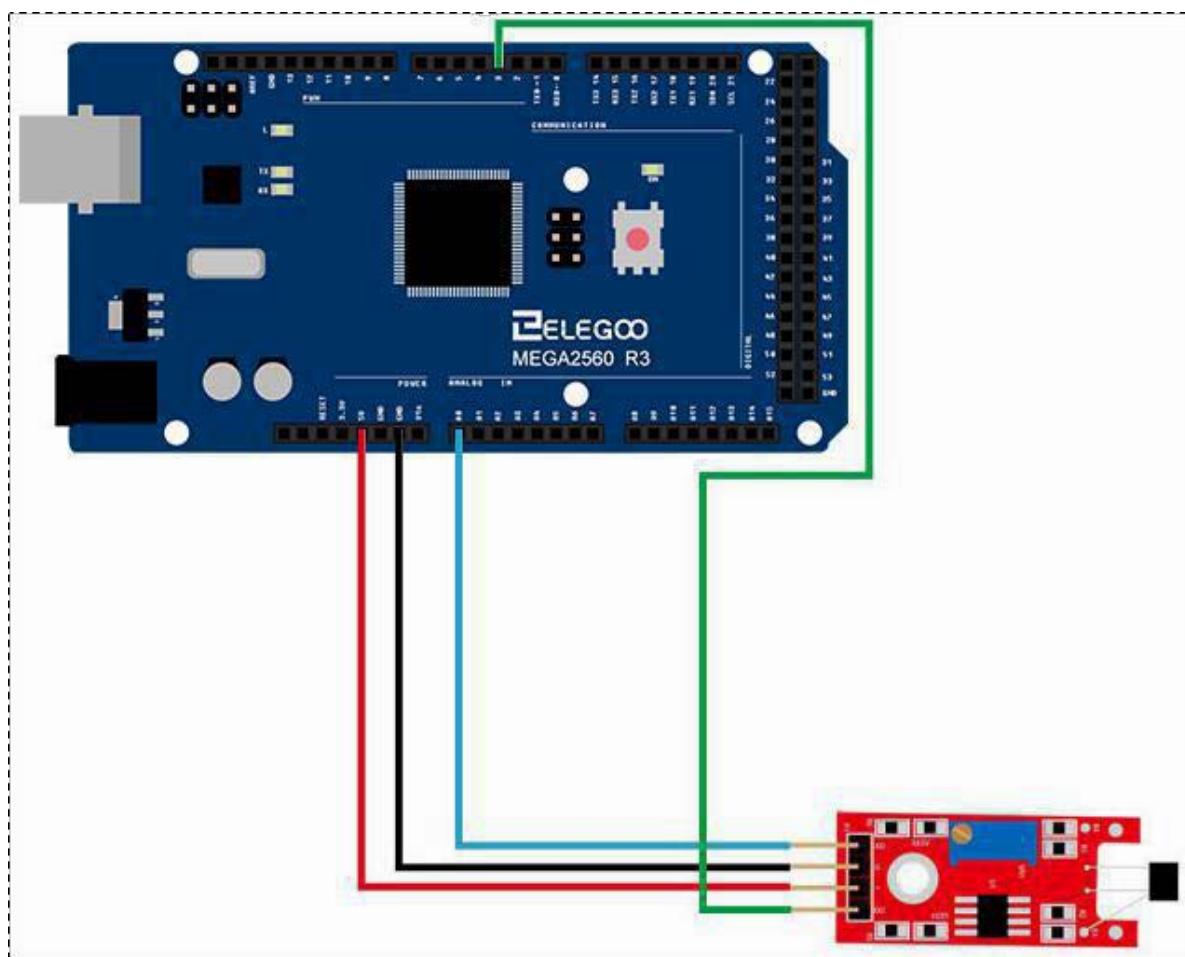
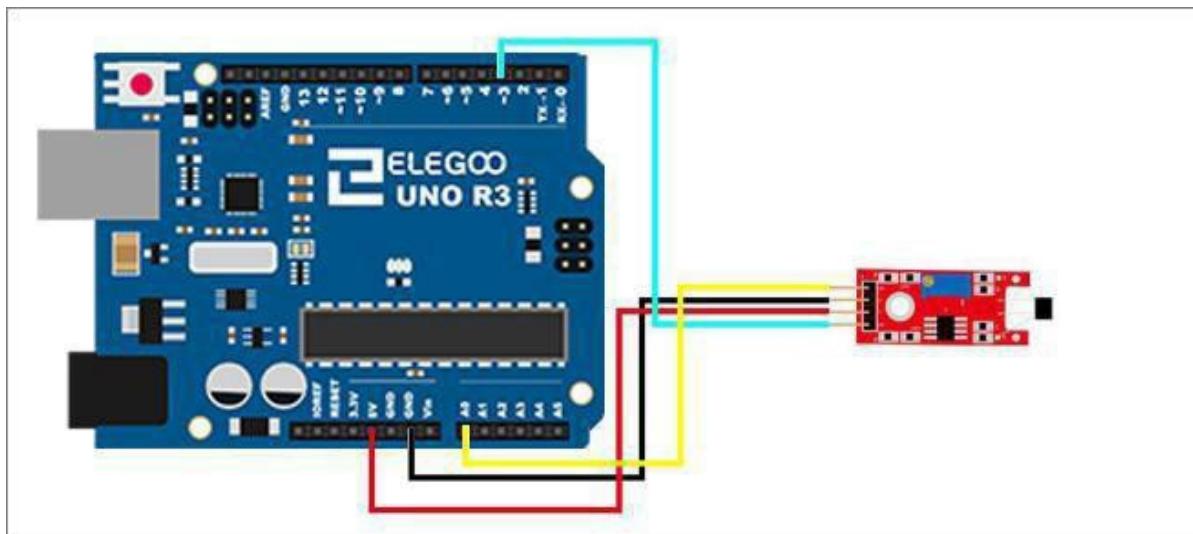
Omni polar: The sensor is designed to operate with Radiometric linear: Output is proportional to magnetic field strength. Output sensitivity range is 2.5 – 3.75 mV per unit of Gauss.

Connection

Schematic



Wiring diagram

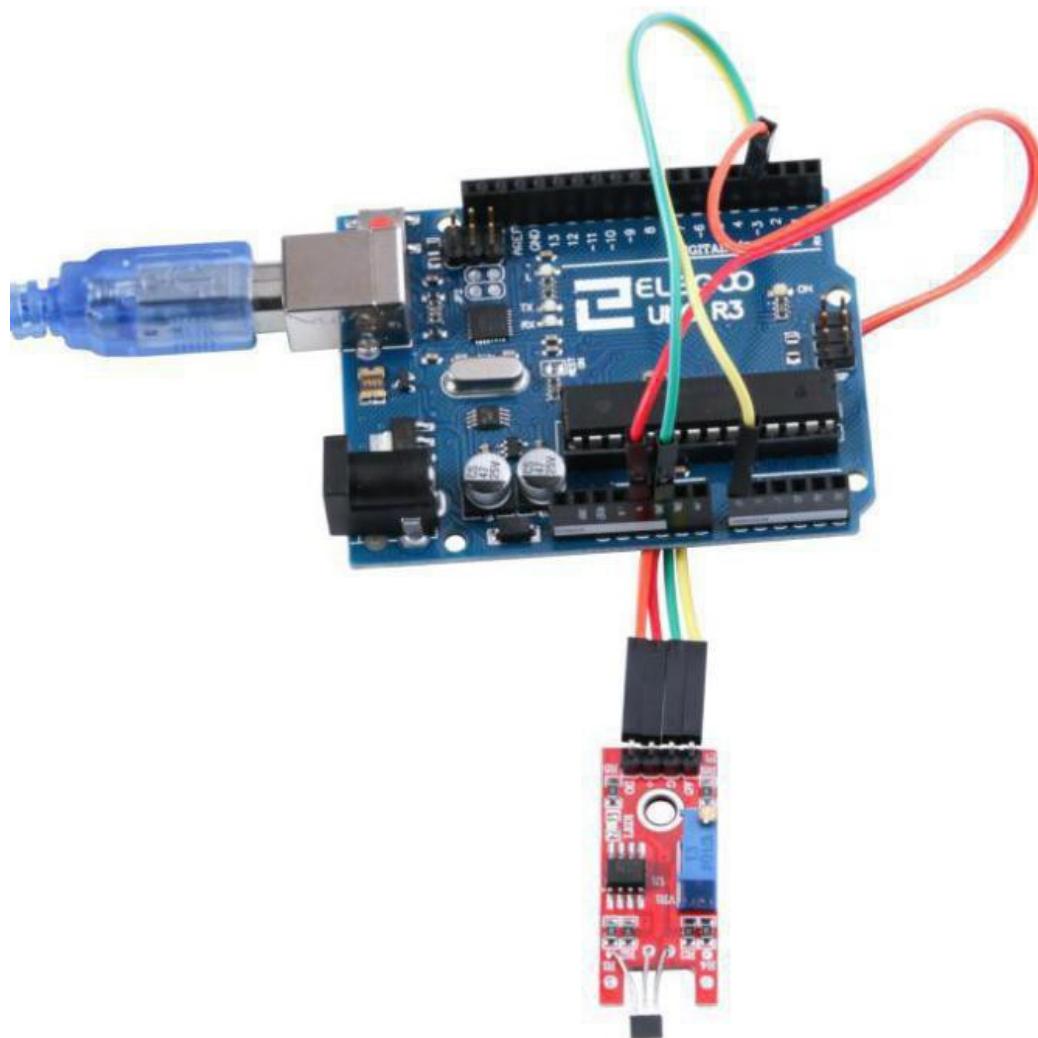


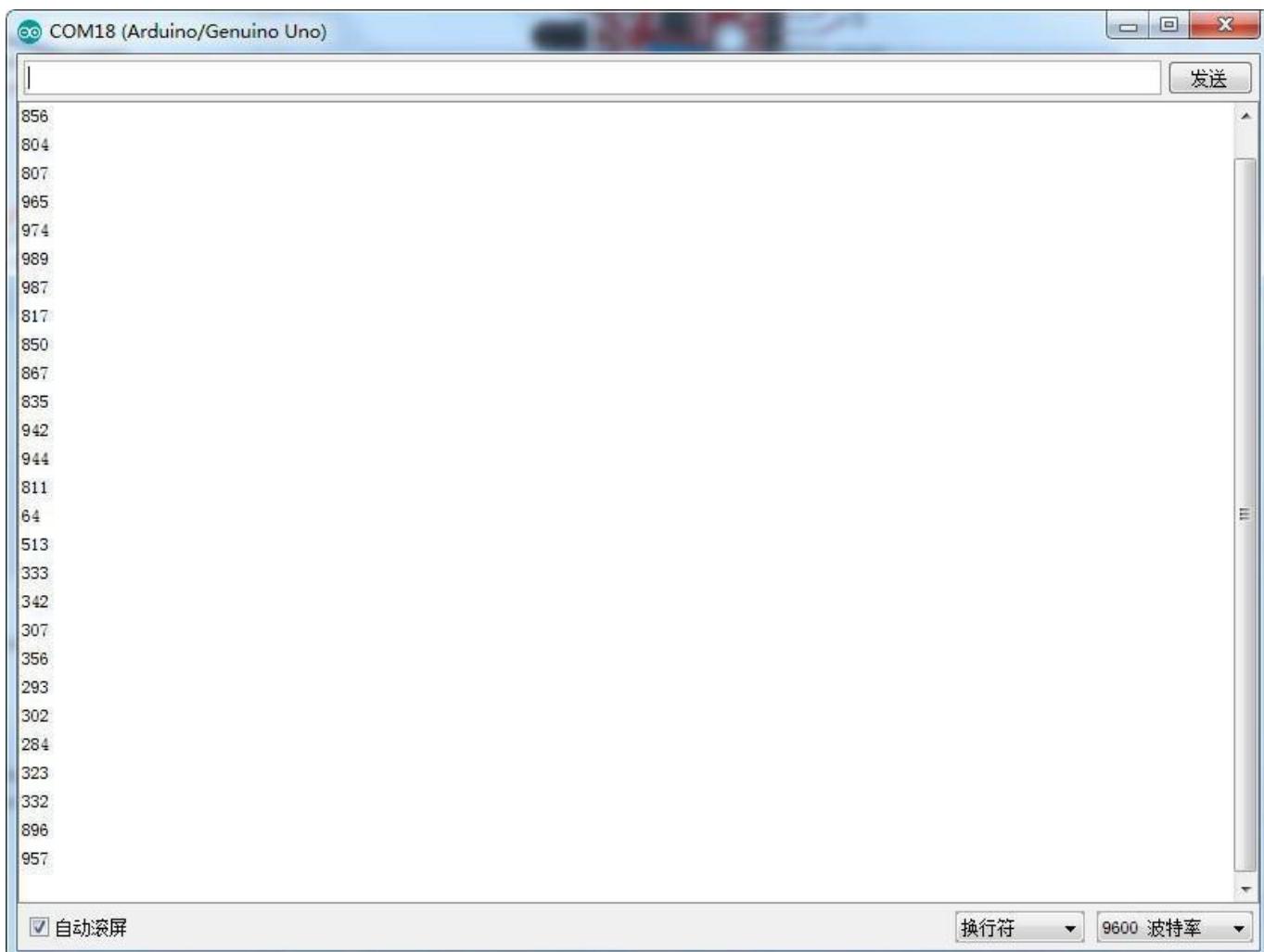
Code

After we connect the circuit, open our tutorial data "code" folder and find "Lesson 19 LINEAR HALL AND ANALOG HALL MODULE" folder open and then run the compiler upload program,

For the Hall sensor module, we can choose to output: digital output or analog output. In the following figure, we use the DO port output. So we can see that if the Hall sensor senses the magnetic force, the light will turn on, and then turn on the monitor, we can see the following data:

Example picture





The followings are the code used in this experiment and their explanations:

(1) Digital port control program

```
//define LED port  
int Led=13;  
  
//define switch port  
int buttonpin=3;  
  
//define digital variable val  
int val;  
  
void setup()  
{  
    //define LED as a output port  
    pinMode(Led,OUTPUT);  
    //define switch as a output port
```

```
pinMode(buttonpin,INPUT);
}

void loop()

{
//read the value of the digital interface 3 assigned to val
val=digitalRead(buttonpin);
//when the switch sensor have signal, LED blink
If (val==HIGH)
{
digitalWrite(Led,HIGH);
}
else
{
digitalWrite(Led,LOW);
}
}
```

(2) Simulation of mouth control procedures

```
// select the input pin for the potentiometer
int sensorPin = A0;
// select the pin for the LED
int ledPin = 13;
// variable to store the value coming from the sensor
int sensorValue = 0;
```

```
void setup()
{
pinMode(ledPin,OUTPUT);
Serial.begin(9600);
}

void loop(){
sensorValue =  analogRead(sensorPin);
digitalWrite(ledPin, HIGH);
delay(sensorValue);
digitalWrite(ledPin, LOW);
delay(sensorValue);
Serial.println(sensorValue, DEC);
}
```

Lesson 20 Flame Sensor Module

Overview

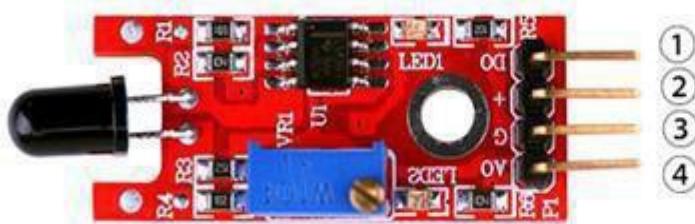
In this experiment, we will learn how to use the flame sensor module.

This module is sensitive to the flame and radiation. It also can detect ordinary light source in the range of a wavelength 760nm-1100 nm. The detection distance is up to 100 cm. The Flame sensor can output digital or analog signal. It can be used as a flame alarm or in firefighting robots.

Flame sensor module

A sensor module to detect flames. The spectral sensitivity of the sensor is optimized to detect emissions from naked flames. The output signal 'DO' is pulled high (active high) when a flame is detected. The switching threshold is adjustable via a preset pot. An analog output signal from the sensor is available at pin 'AO'.

- Typical spectral sensitivity: 720-1100 nm
- Typical detection angle: 60°



- 1.DO:digital output
- 2.VCC: 3.3V-5V DC
- 3.GND:ground
- 4.AO:analog output

Component Required:

1 x Elegoo Uno R3

1 x USB cable

1 x Flame sensor module

4 x F-M wires

Component Introduction

Flame sensor module:

- Detects a flame or a light source of a wavelength in the range of 760nm-1100 nm

Detection distance: 20cm (4.8V) ~ 100cm (1V)

Detection angle about 60 degrees, it is sensitive to the flame spectrum.

Comparator chip LM393 makes module readings stable.

Adjustable detection range.

Operating voltage 3.3V-5V

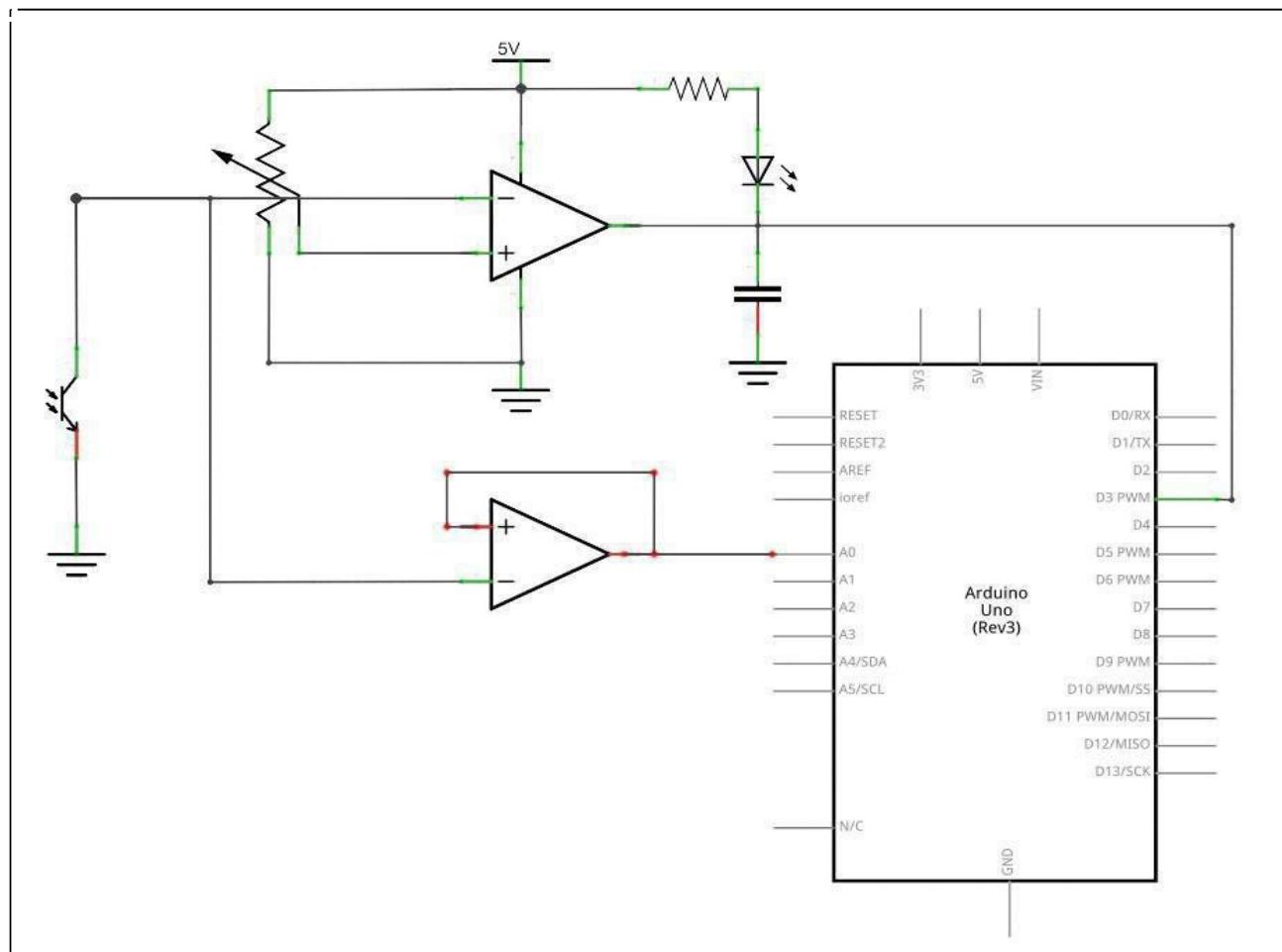
Digital and Analog Output

"DO digital switch outputs (0 and 1)" AO analog voltage output

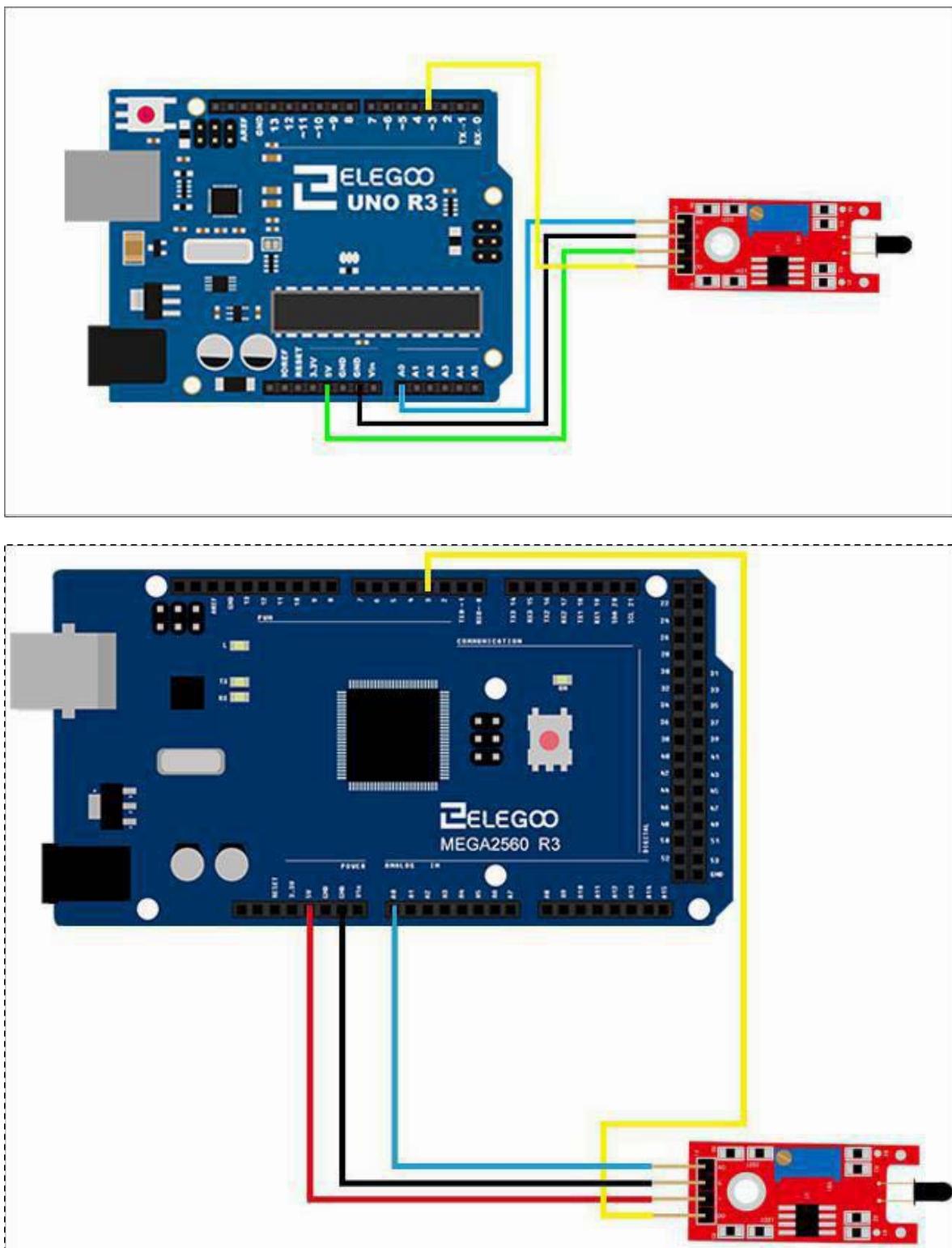
Power indicator and digital switch output indicator

Connection

Schematic



Wiring diagram

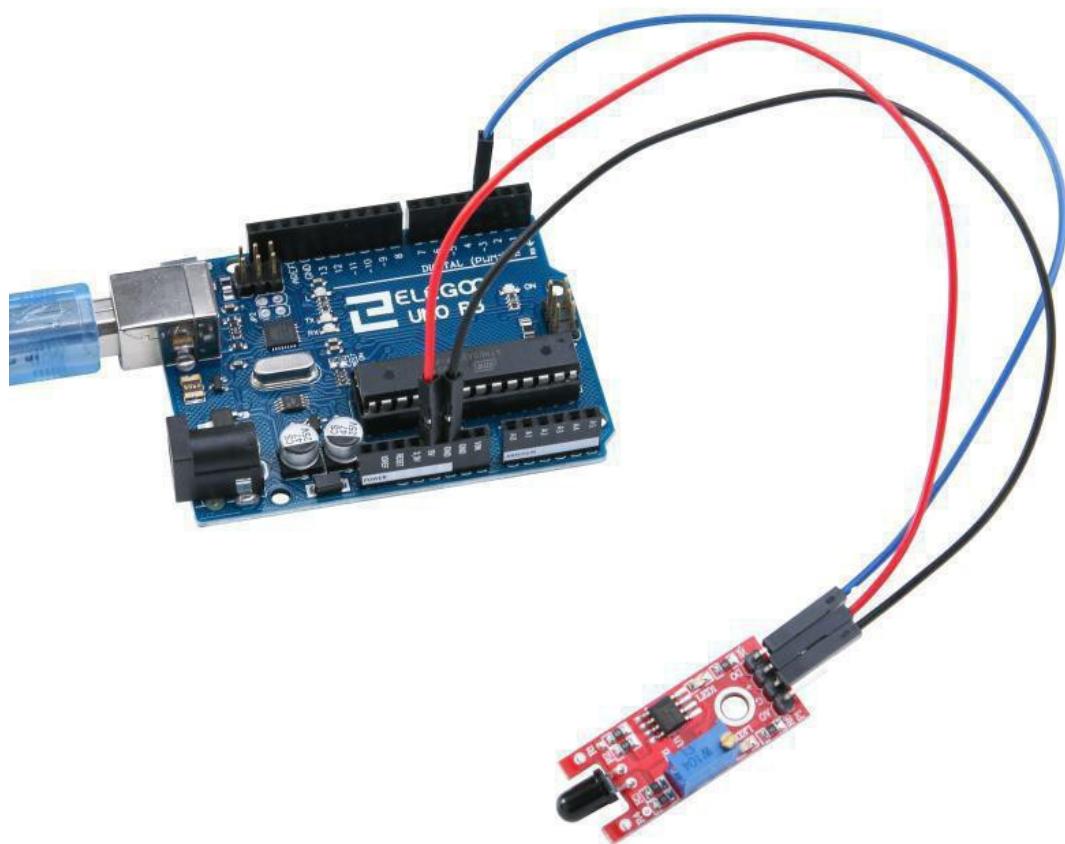


Code

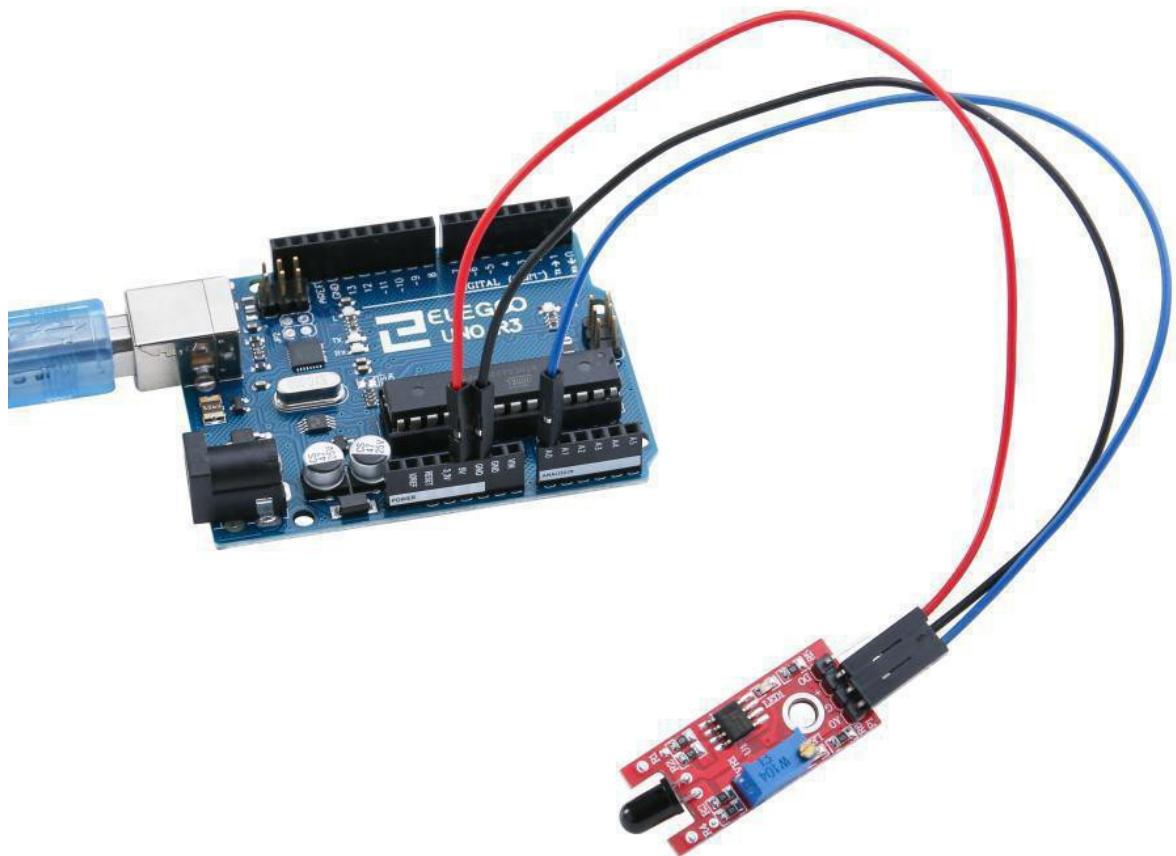
After we connect the circuit, open our tutorial data "code" folder, find "Lesson 20 FLAME SENSOR MODULE" folder open and then run the compiler upload program,

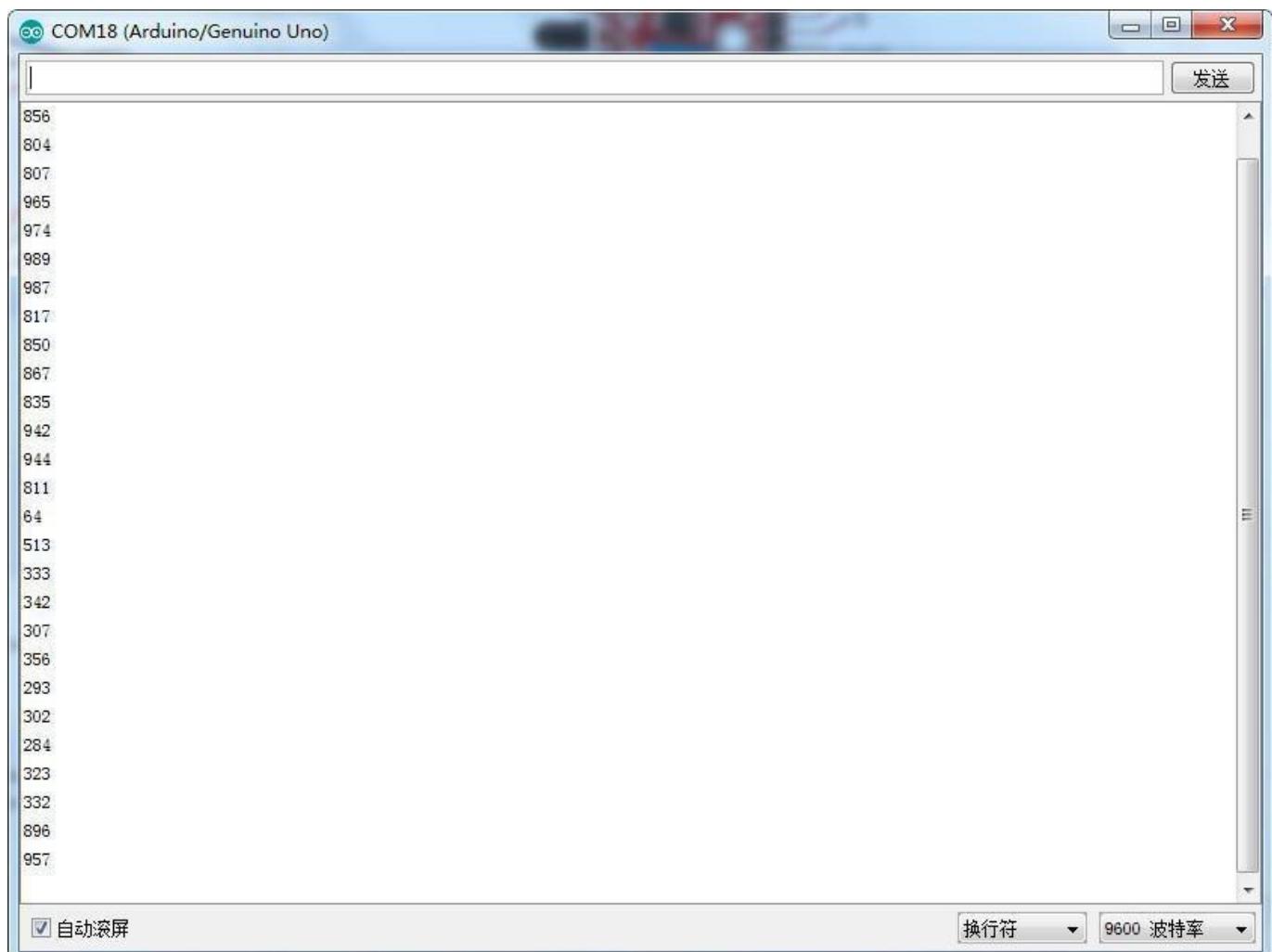
For the flame sensor module, we can choose the output: digital output or analog output. In the following figure, we use the DO port output. So we can see that if the flame sensor senses the flame, the light will turn on and then turn on the monitor, and we can see the following data:

Example picture



In the following picture, we use the AO port to output. so we can see that if the flame sensor sensing the flame, the module will output a data which reflect the strength of the flame. The number is from 0 to 1023.





The followings are the code used in this experiment and their explanations:

(1)digital port control program

```
//define LED port
int Led=13;
//define switch port
int buttonpin=3;
//define digital variable val
int  val;
void setup()
{
//define LED as a output port
pinMode(Led,OUTPUT);
//define switch as a output port
pinMode(buttonpin,INPUT);
}
void  loop()
{
//read the value of the digital interface 3 assigned to val
val=digitalRead(buttonpin);
//when the switch sensor have signal, LED blink
if(val==HIGH)
{
digitalWrite(Led,HIGH);
}
else
{
digitalWrite(Led,LOW);
}
}
```

(2) Simulation of mouth control procedures

```
// select the input pin for the potentiometer
int sensorPin = A0;

// select the pin for the LED
int ledPin = 13;

// variable to store the value coming from the sensor
int sensorValue = 0;

void setup()
{
pinMode(ledPin,OUTPUT);
Serial.begin(9600);
}

void loop()
{
sensorValue = analogRead(sensorPin);
digitalWrite(ledPin, HIGH);
delay(sensorValue);
digitalWrite(ledPin, LOW);
delay(sensorValue);
Serial.println(sensorValue, DEC);
}
```

Lesson 21 Touch Sensor Module

Overview

In this experiment, we will learn how to use the metal touch module.

Touch sensor module

Touch sensitive switch. Touching the sensor pin produces an output at the 'DO' pin. The output is not a clean signal but includes 50 Hz mains induced signals ('mains hum'). The output signal is 'active high' and the circuit sensitivity can be adjusted with a pot. An analog output signal from the sensor is available at pin 'AO'.



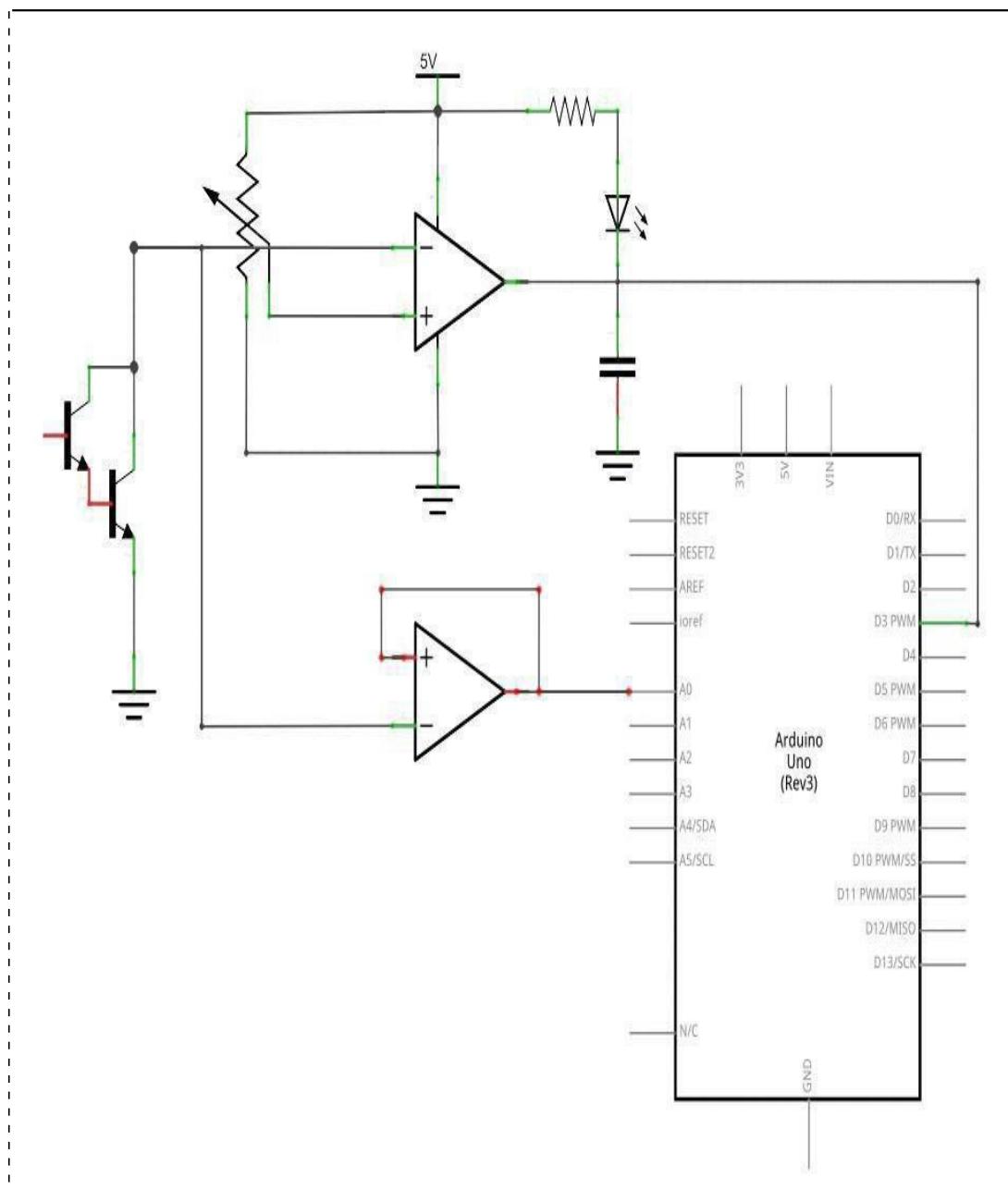
- 1.DO:digital output
- 2.VCC: 3.3V-5V DC
- 3.GND:ground
- 4.AO:analog output

Component Required:

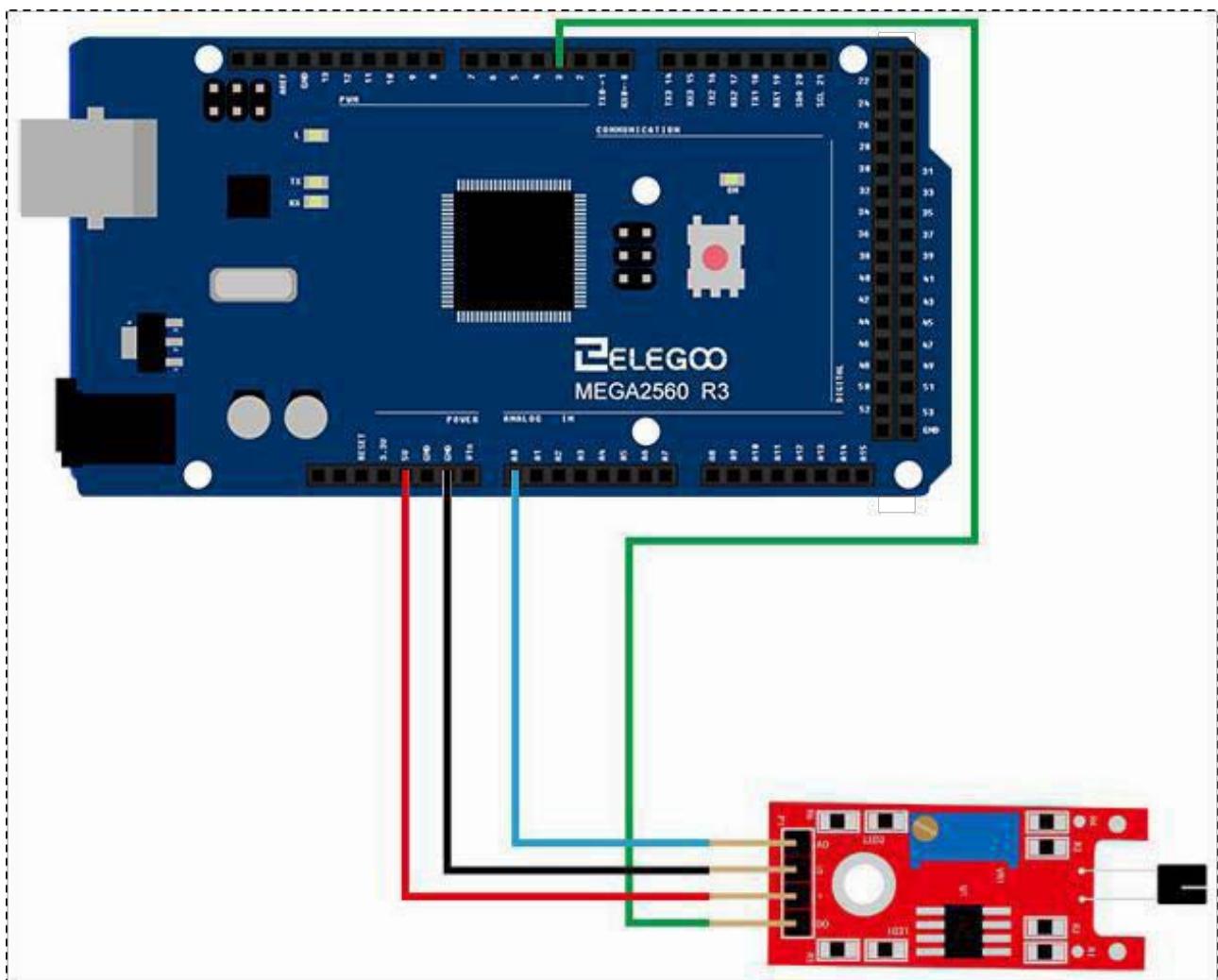
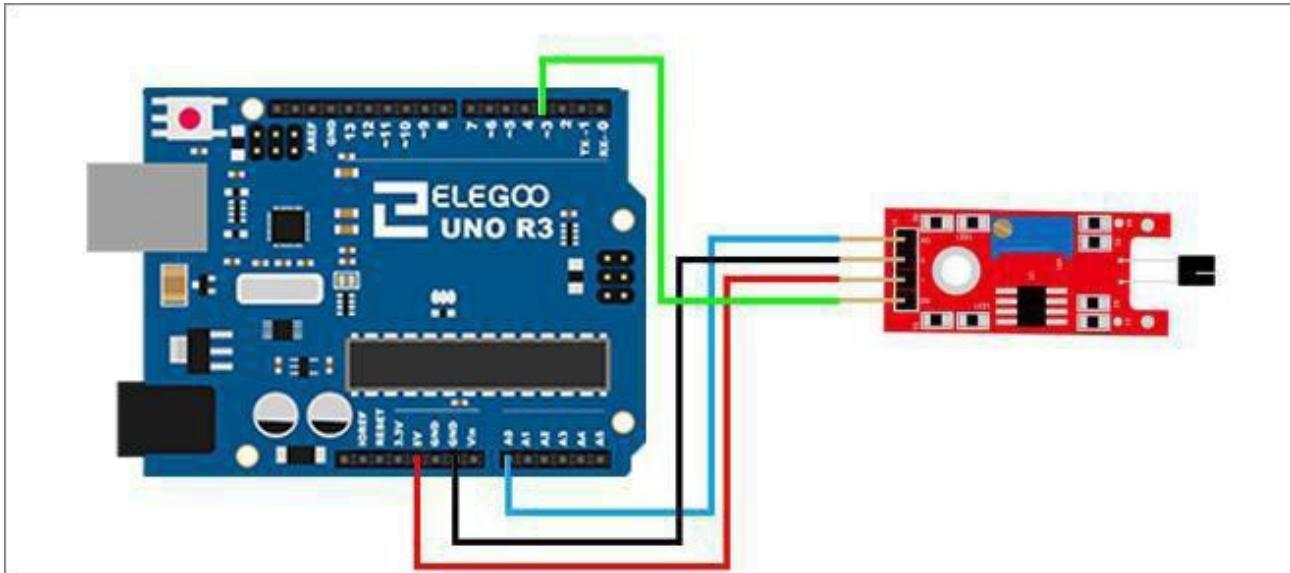
- 1 x Elegoo Uno R3
- 1 x USB cable
- 1 x Touch sensor module
- 4 x F-M wires

Connection

Schematic



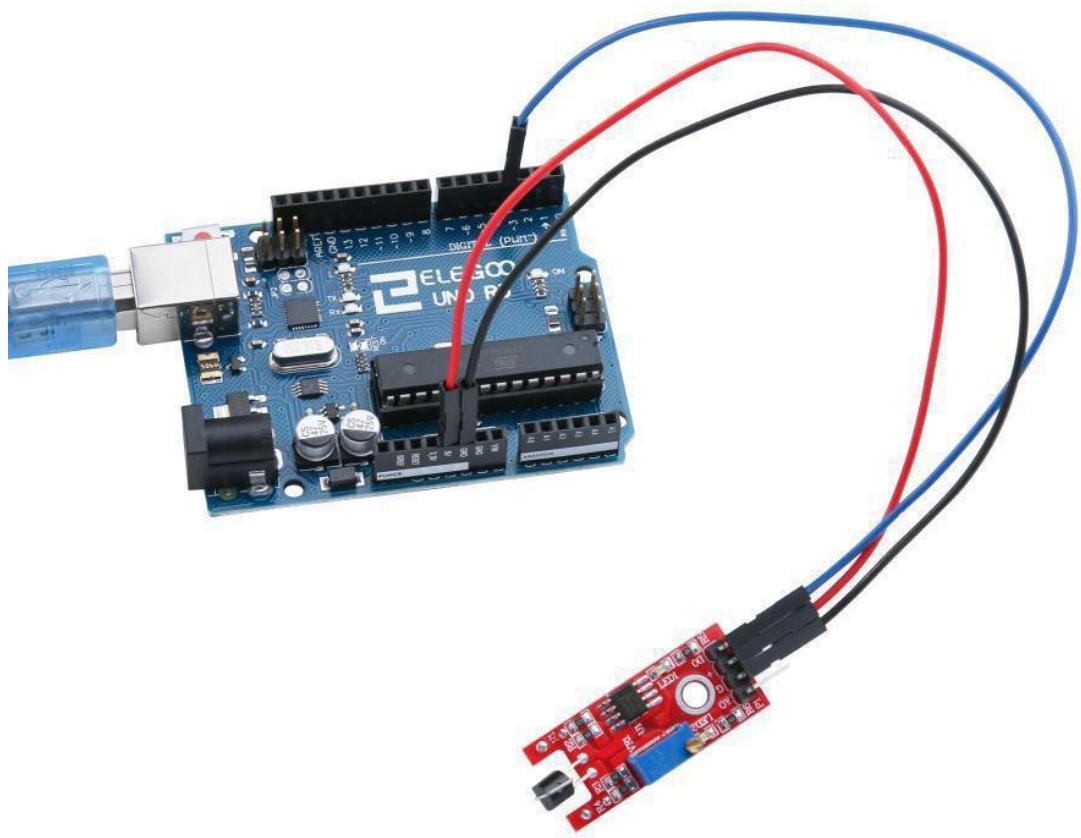
Wiring diagram



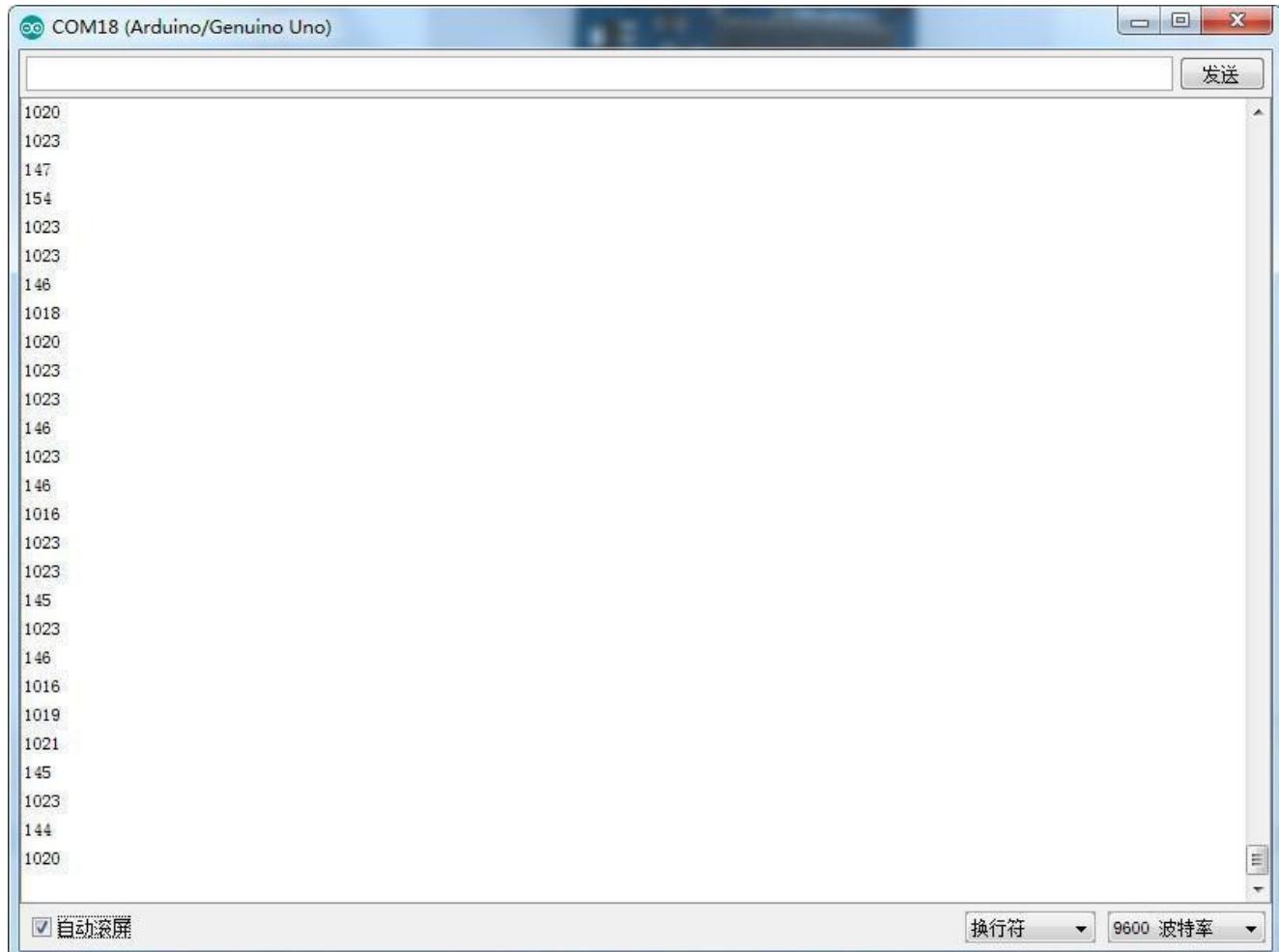
Code

After we connect the circuit, open our "code" folder and find the "Lesson 21 METAL TOUCH MODULE" folder. Open and run the compiler uploader. For the metal touch module, we have the option to output either digital output or analog output. In the following figure, we use the DO port output. So we can see that if the sensor senses, the light will be on. Then open the monitor, we can see the following data:

Example picture



In the following picture, we use the AO port to output. so we can see that if the sensor sensing, the module will output a data. The number is from 0 to 1023.



The followings are the code used in this experiment and their explanations:

(1) simulation port control program

```
//define LED port
int Led=13;
//define switch port
int buttonpin=3;
//define digital variable val
int  val;
void  setup()
{
//define LED as a output port
pinMode(Led,OUTPUT);
//define switch as a output port
pinMode(buttonpin,INPUT);
}
void  loop()
{
//read the value of the digital interface 3 assigned to val
val=digitalRead(buttonpin);
//when the switch sensor have signal, LED blink
if(val==HIGH)
{
digitalWrite(Led,HIGH);
}
else
{
digitalWrite(Led,LOW);
}
}
```

(2) digital port control program

```
// select the input pin for the potentiometer
int sensorPin = A0;

// select the pin for the LED
int ledPin = 13;

// variable to store the value coming from the sensor
int sensorValue = 0;

void setup()
{
pinMode(ledPin,OUTPUT);
Serial.begin(9600);
}

void loop()
{
sensorValue = analogRead(sensorPin);
digitalWrite(ledPin, HIGH);
delay(sensorValue);
digitalWrite(ledPin, LOW);
delay(sensorValue);
Serial.println(sensorValue, DEC);
}
```

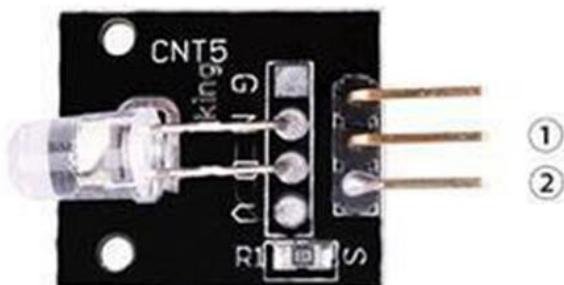
Lesson 22 Seven-Color Flash LED Module

Overview

In this experiment, we will learn how to use the seven-color flash LED module.

Seven-color flash led module

Clear 5mm LED for direct operation from 5V. The LED color automatically cycles through a seven-color sequence. The 5 V supply connects to the 'S' pin and ground on the center pin.



1. GND
2. OUTPUT

Component Required:

- 1 x Elegoo Uno R3
- 1 x USB cable
- 1 x seven-color flash led module
- 2 x F-M wires

Component Introduction

Seven-color flash led:

7 color flashing LED module automatically uses 5mm round high-brightness light-emitting diode which has the following characteristics:

Product Type: LED

Product Model: YB-3120B4 Pn YG-PM Shape:

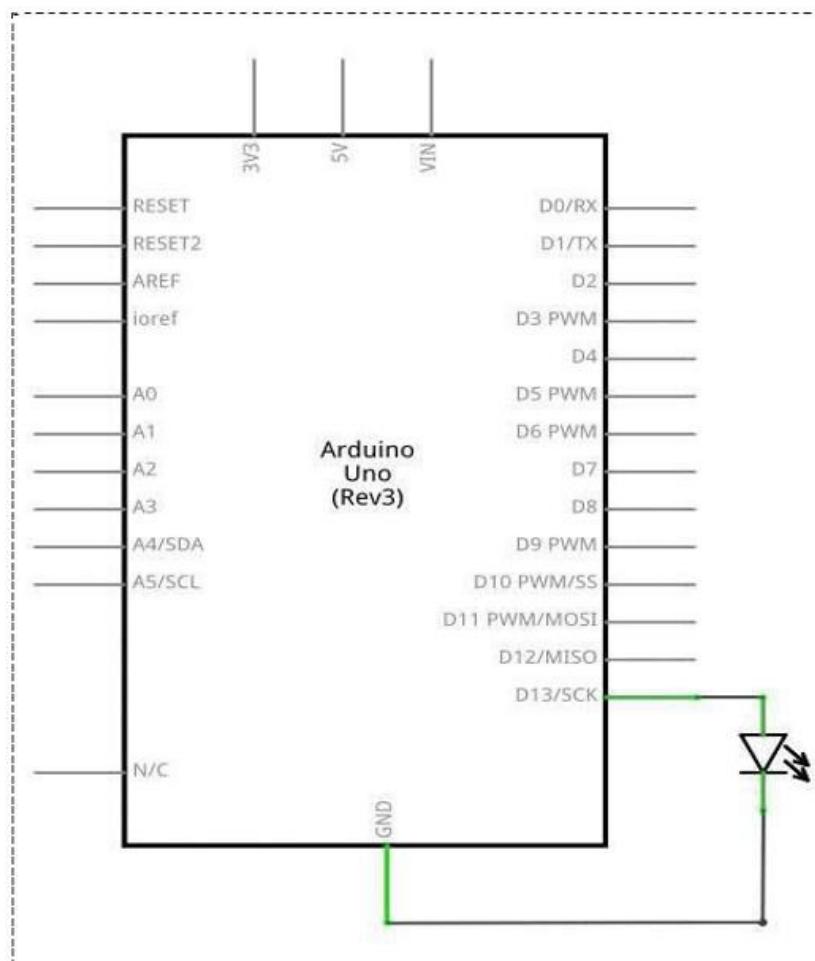
Round LED 5mm DIP type

Color: pink yellow green (highbrightness) Lens type: white mist

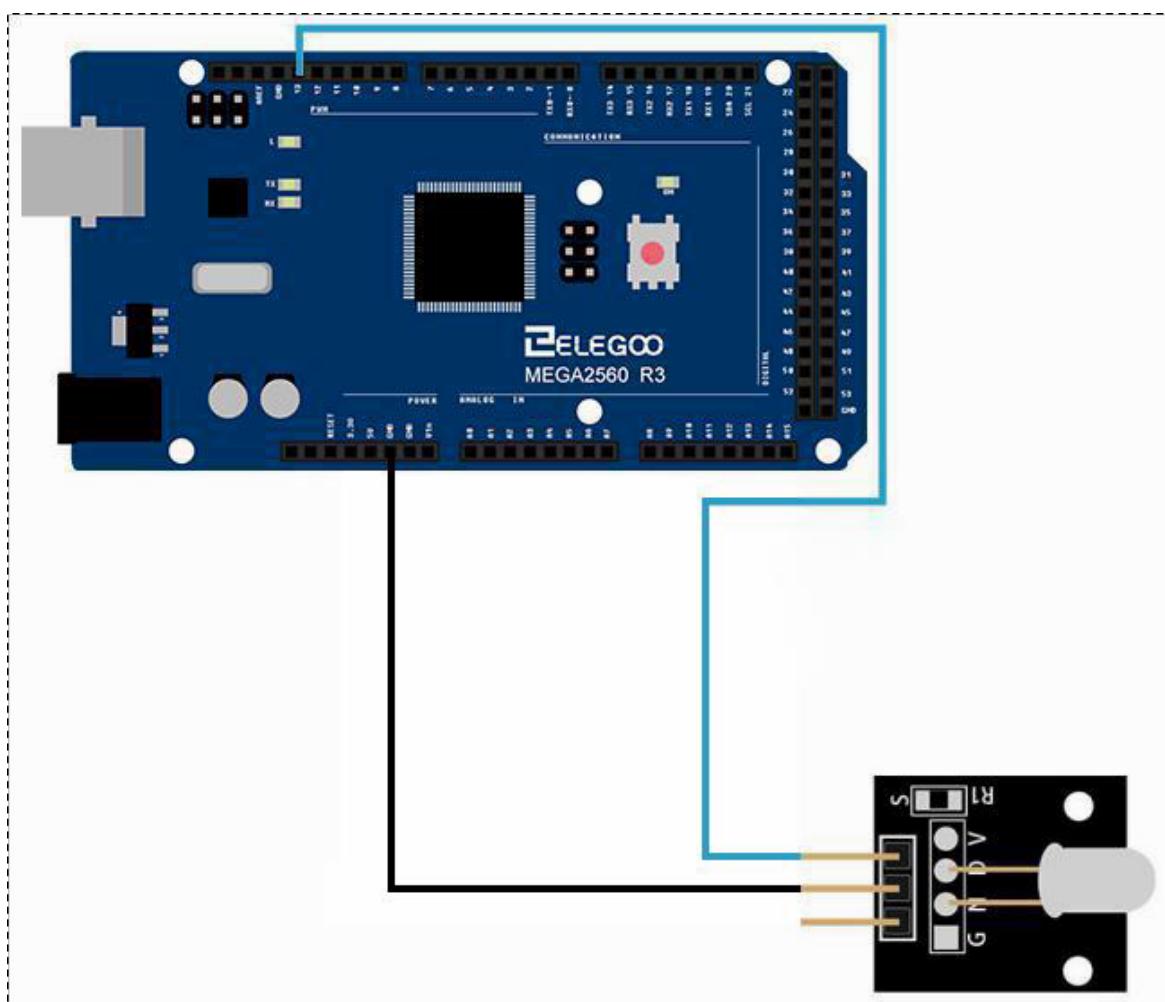
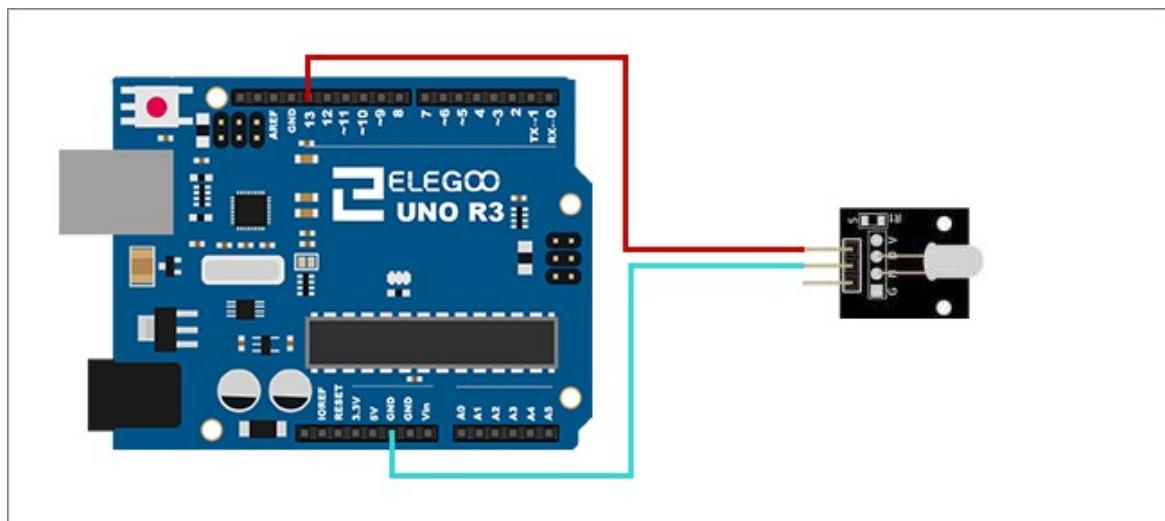
Standard Forward Voltage :3.0-4 .5V

Connection

Schematic



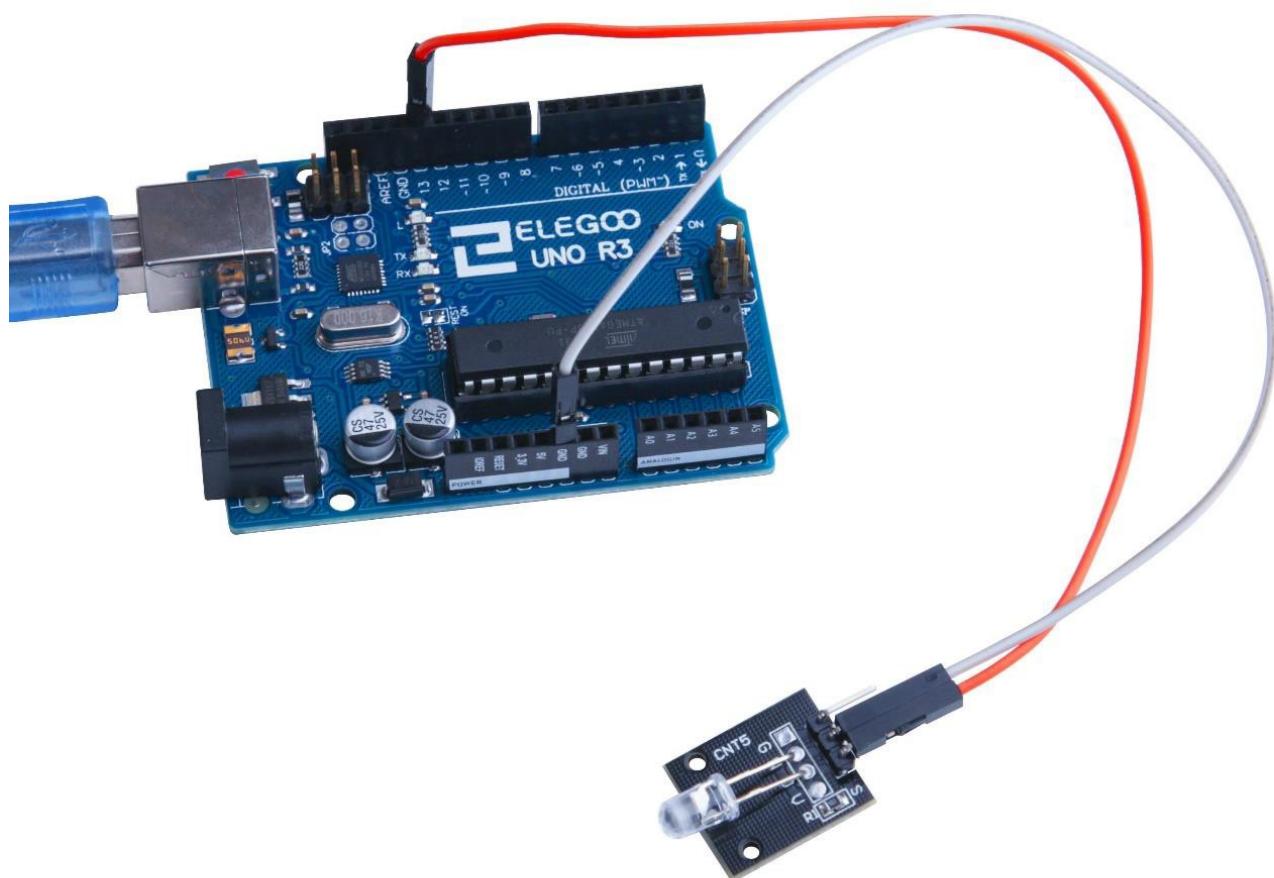
Wiringdiagram



Code

After we connect the circuit, open our tutorial data "code" folder, find "Lesson 22 7 COLOR FLASH LED MODULE" folder open and then run the compiler upload program, you can see our LED colorful flashing phenomenon;

Example picture



The followings are the code used in this experiment and their explanations:

```
void setup()
{
    // initialize the digital pin as an output.
    // Pin 13 has an LED connected on most Arduino boards:
    pinMode(13, OUTPUT);
}

void loop()
{
    digitalWrite(13, HIGH);
}
```

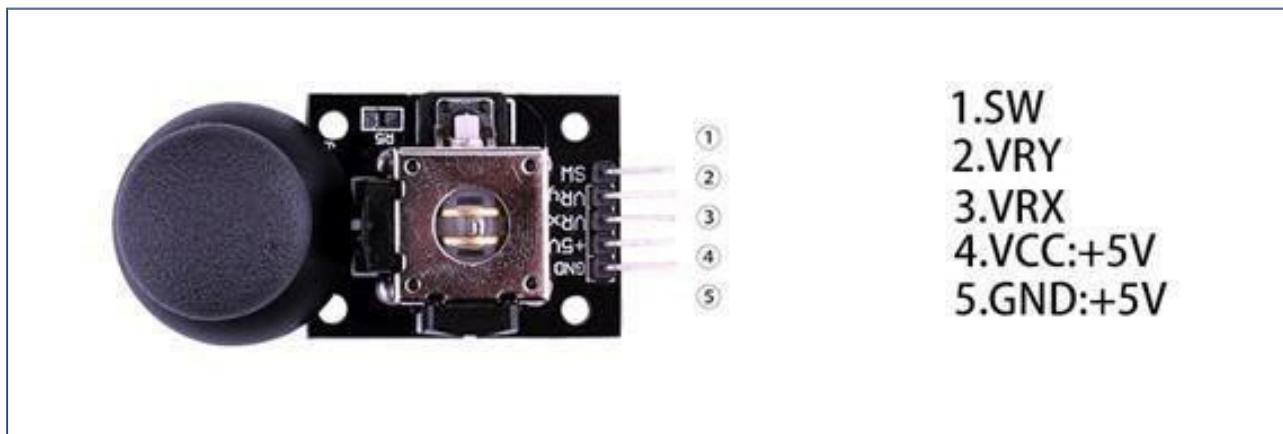
Lesson 23 Joystick Module

Overview

Just like a joystick on game console. You can control x, y and z dimensions input by this joystick module. It can be considered as combination of potentiometers and one button. Data type of the x, y dimensions are analog input signals and z dimension is digital input signal. Thus the x and y ports connect to analog pins of Sensor Shield, while z port connects to digital pin.

Joystick Module

An analog 2-axis joystick with 2x 10K ohm pots and push button function. Connector pin descriptions are printed on the PCB. A push-on operating knob is included with the module.



Component Required:

- 1 x Elegoo Uno R3
- 1 x USB cable
- 1 x Joystick module
- 5 x F-M wires

Component Introduction

Joystick sensor:

Lots of robot projects need joystick. This module provides an affordable solution. By simply connecting to two analog inputs, the robot is at your commands with X, Y control. It also has a switch that is connected to a digital pin. This joystick module can be easily connected to Arduino by IO Shield. This module is for Arduino with cables supplied.

Specification

Supply Voltage: 3.3V to 5V

Interface: Analog x2, Digitalx1 Size: 40*28mm

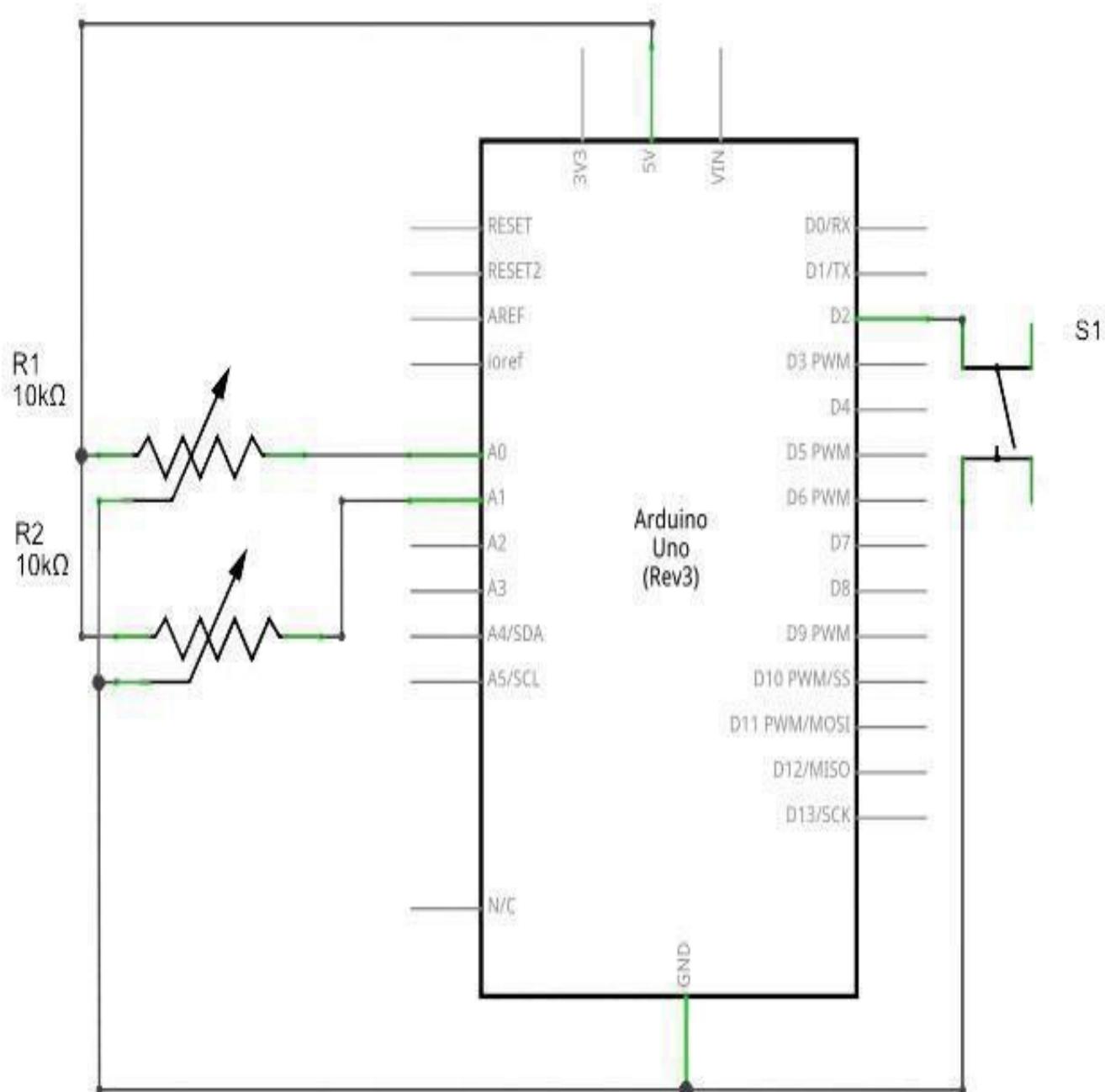
Weight: 12g

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.

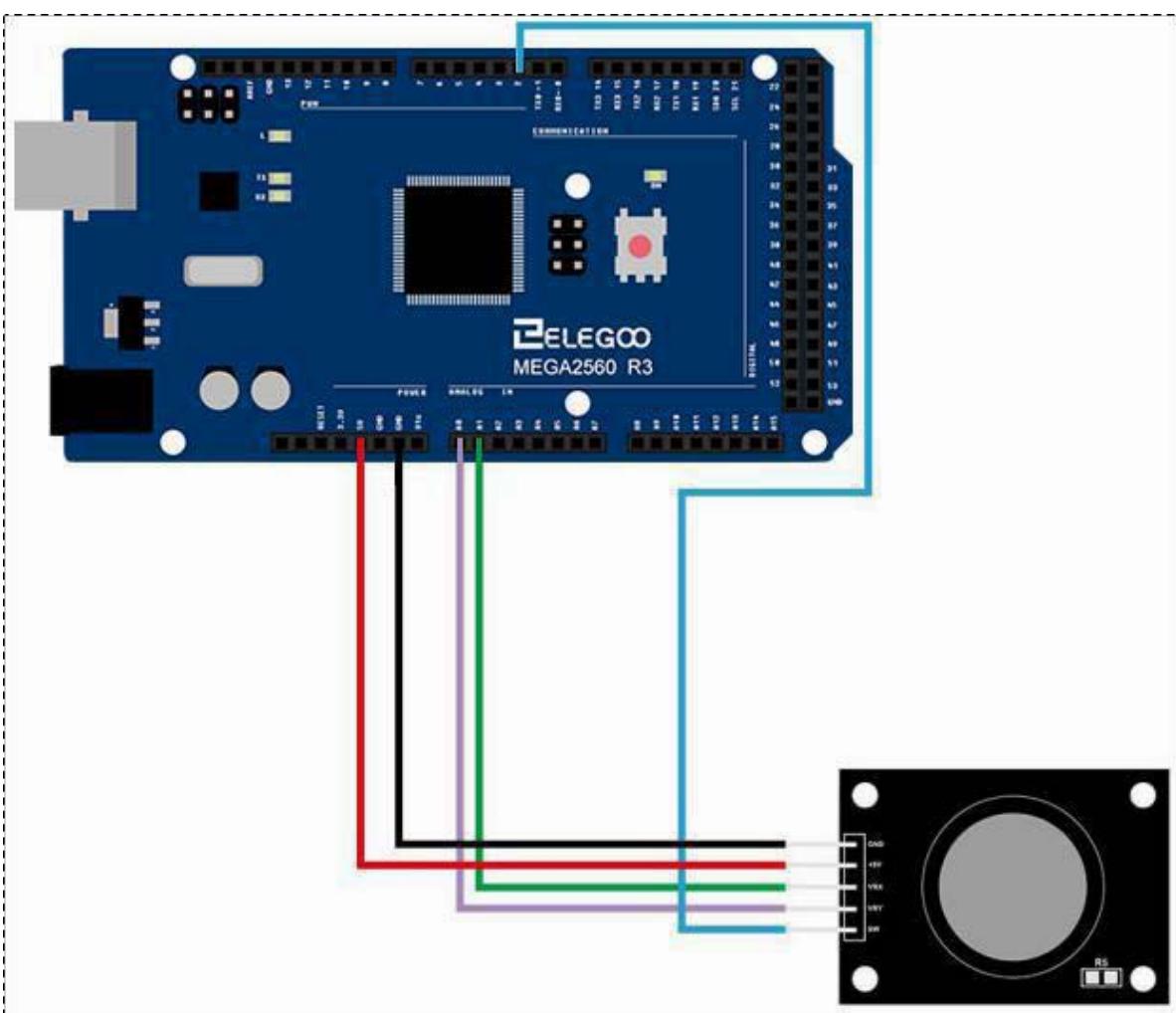
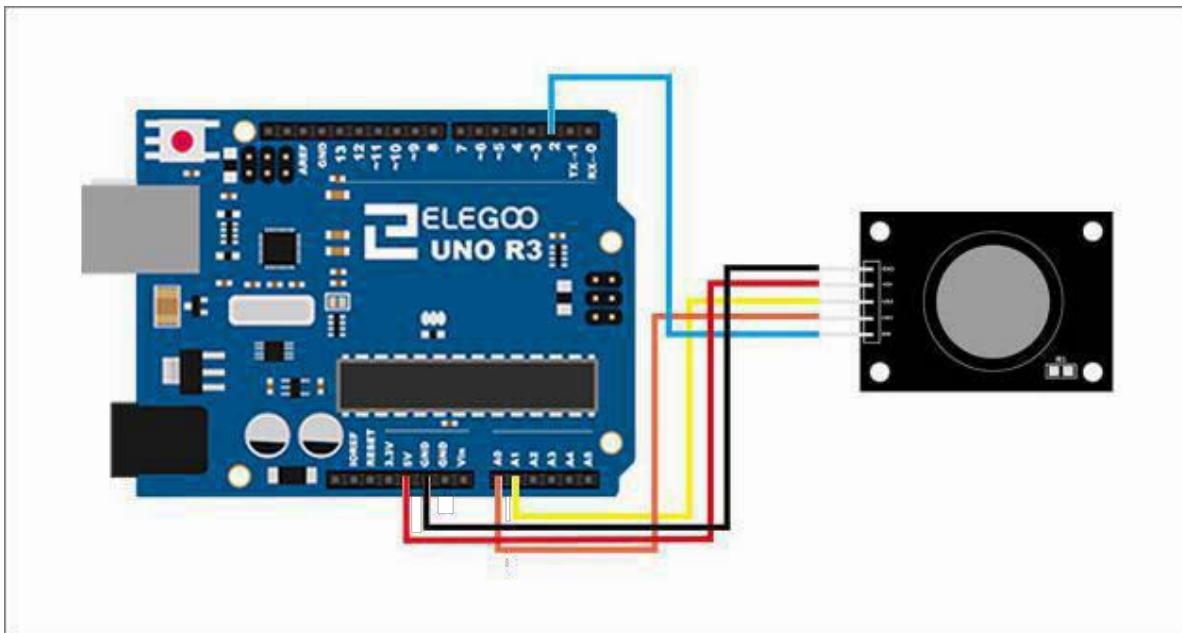
We have to use analog Arduino pins to read the data from the X/Y pins, and a digital pin to read the button. The Key pin is connected to ground, when the joystick is pressed down, and is floating otherwise. To get stable readings from the Key /Select pin, it needs to be connected to Vcc via a pull-up 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

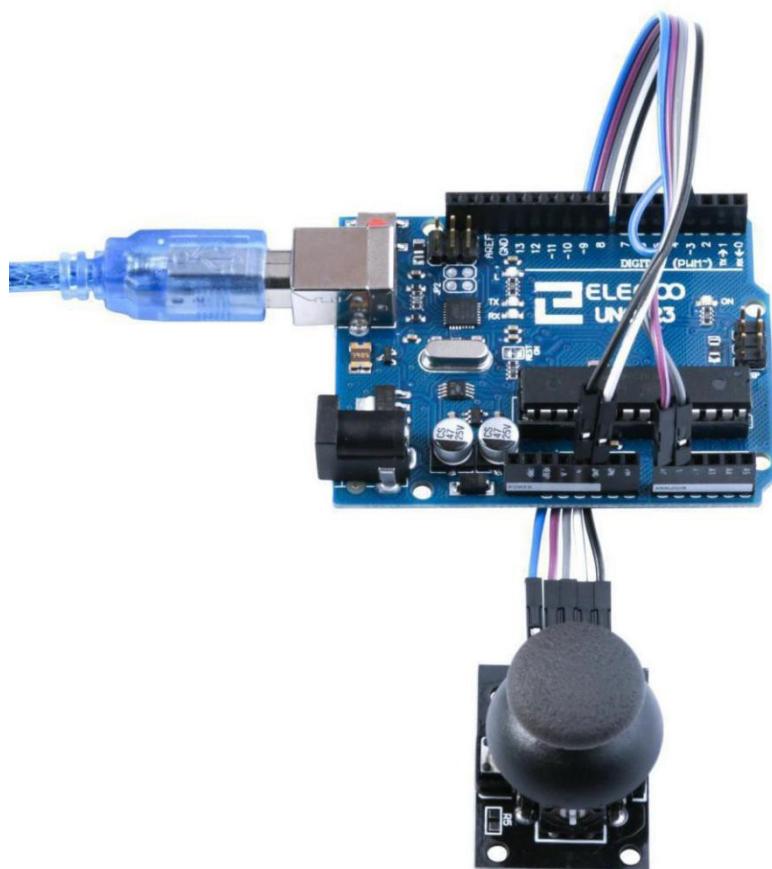


Code

Analog joysticks are basically potentiometers, so they return analog values. When the joystick is in the rest or neutral position, a value of about 512 should be returned, with values ranging from 0 to 1023. In addition; SW (Z axis) is a digital input signal, connected to the digital port, and enables the pull-up resistor. Value of SW: 1 means not pressed, 0 means pressed.

When we connect the circuit, open the "code" folder of our tutorial, find the "Lesson 23 Joystick Module" folder and run the compiler uploader, and then open the monitor, we can see the following data:

Example picture



COM18 (Arduino/Genuino Uno)

X-axis: 509
Y-axis: 515

Switch: 1
X-axis: 516
Y-axis: 1023

Switch: 1
X-axis: 0
Y-axis: 515

Switch: 0
X-axis: 509
Y-axis: 510

Switch: 0
X-axis: 0
Y-axis: 1023

Switch: 1
X-axis: 509
Y-axis: 516

Switch: 1
X-axis: 509
Y-axis: 516

自动滚屏 换行符 9600 波特率

The followings are the code used in this experiment and their explanations:

```
// Arduino pin numbers  
// digital pin connected to switch output  
const int SW_pin = 2;  
// analog pin connected to X output  
const int X_pin = A1;  
// analog pin connected to Y output  
const int Y_pin = A0;  
  
void setup() {  
    pinMode(SW_pin, INPUT);  
    digitalWrite(SW_pin, HIGH);  
    Serial.begin(9600);  
}  
  
void loop() {  
    Serial.print("Switch: ");  
    Serial.print(digitalRead(SW_pin));  
    Serial.print("\n");  
    Serial.print("X-axis: ");  
    Serial.print(analogRead(X_pin));  
    Serial.print("\n");  
    Serial.print("Y-axis: ");  
    Serial.println(analogRead(Y_pin));  
    Serial.print("\n");  
    delay(500);  
}
```

Lesson 24 Line Tracking Module

Overview

In this experiment, we will learn how to use the tracking module and avoidance module.

Infrared obstacle avoidance sensor is designed for the design of a wheeled robot obstacle avoidance sensor distance adjustable. This ambient light sensor Adaptable, high precision, having a pair of infrared transmitter and receiver, transmitter tubes emit a certain frequency of infrared, When detecting the direction of an obstacle (reflector), the infrared receiver tube receiver is reflected back, when the indicator is lit, Through the circuit, the signal output interface output digital signal that can be detected by means of potentiometer knob to adjust the distance, the effective distance From 2 ~ 40cm, working voltage of 3.3V-5V, operating voltage range as broad, relatively large fluctuations in the power supply voltage of the situation Stable condition and still work for a variety of microcontrollers, Arduino controller, BS2 controller, attached to the robot that can sense changes in the surroundings.

Line tracking module

IR light reflection switch, useful for obstacle avoidance or line following on models that move around the floor. An obstacle in front of the sender/receiver diodes will cause the 'out' pin to be pulled low(active low). A pot allows adjustment of the circuit's sensitivity. The detection distance can be up to approximately 1 cm.



- 1.OUTPUT
- 2.VCC: 3.3V-5V DC
- 3.GND:ground

Component Required:

1x Elegoo Uno R3

1x USB cable

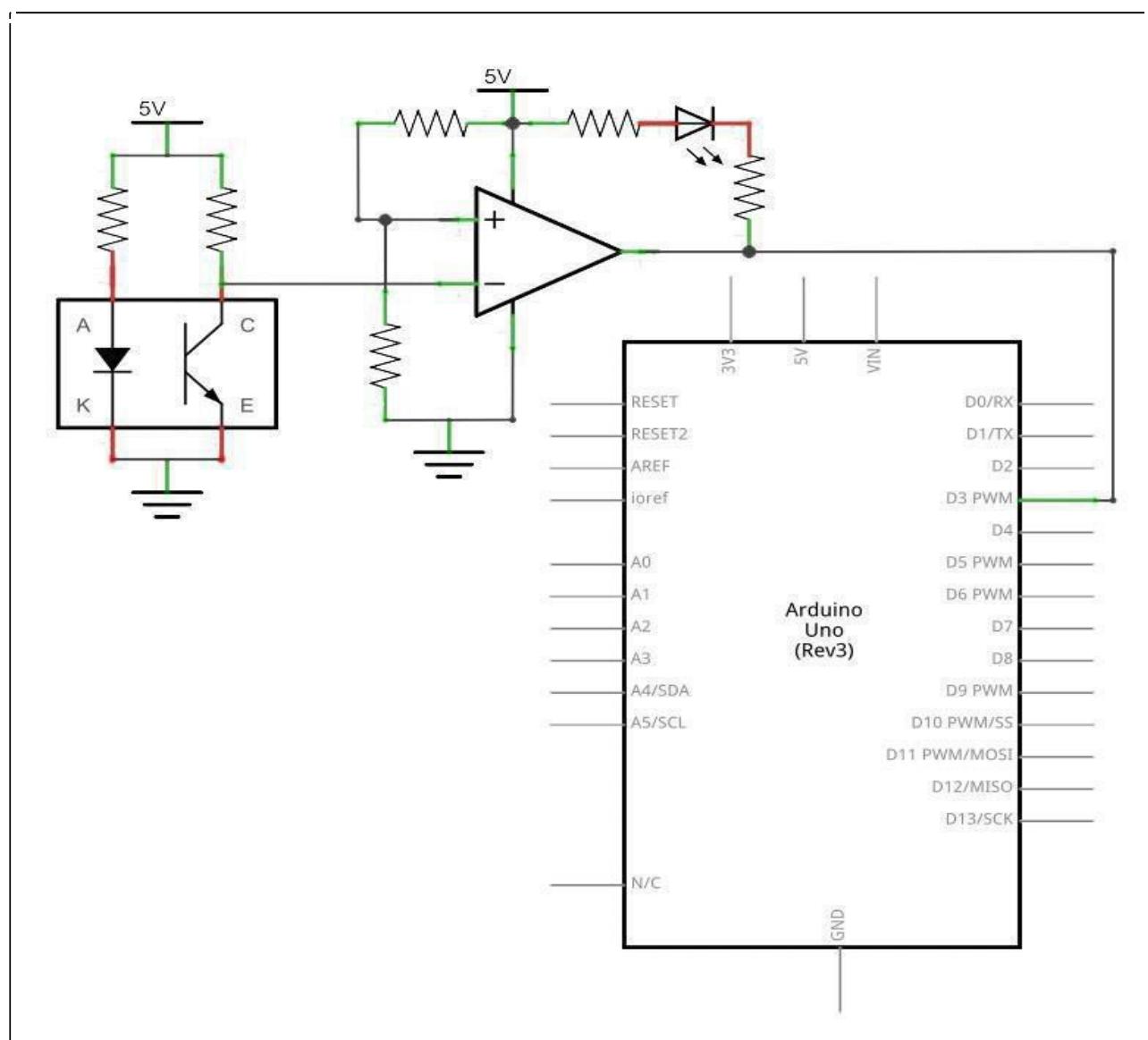
1x line tracking module

3x F-M wires

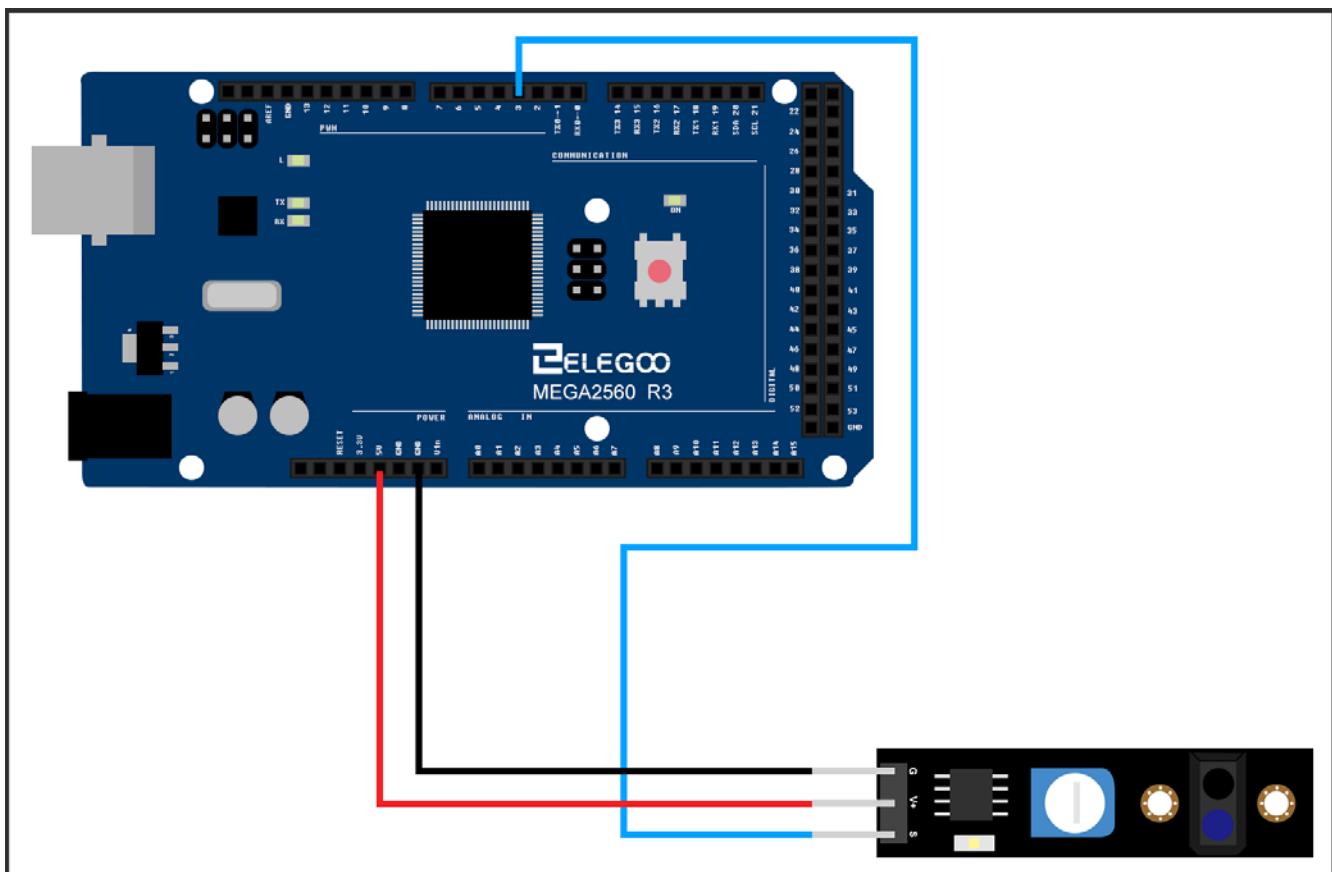
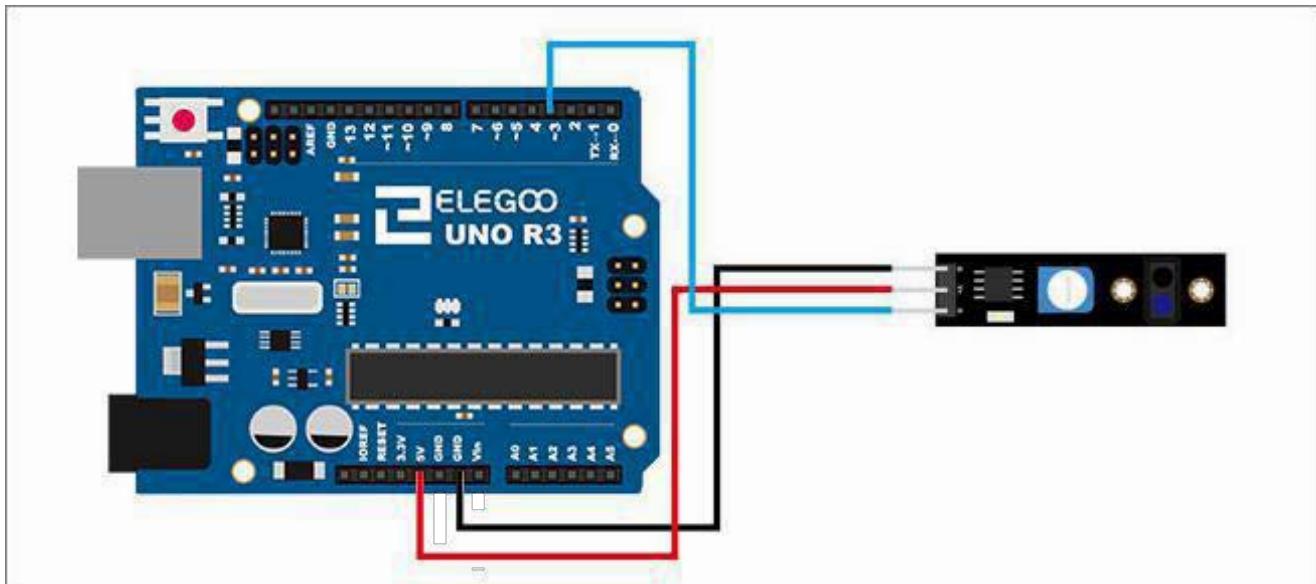
Component Introduction

Connection

Schematic



Wiring diagram



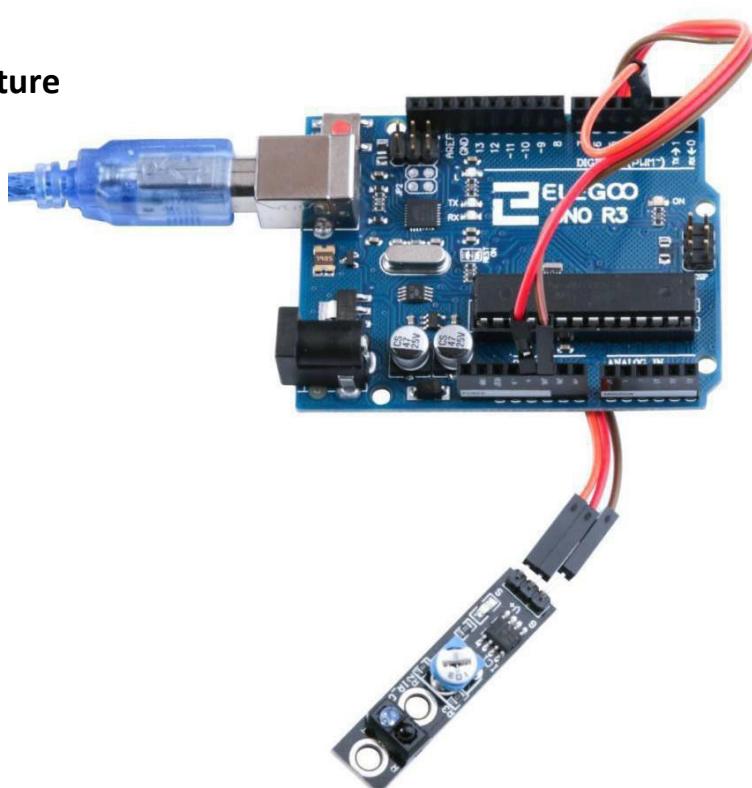
Code

When we connect the circuit, open the "code" folder of our tutorial data, find "Lesson 24 Line tracking module" folder open and then run the compiler upload program, at this time if the black or black paper block module, the board on the 13 The "L" indicator will be on when the LED is on. It is normal for the LED to shine when there is nothing obstructing it. It belongs to the principle of infrared optics; it reflects back when the IR tube illuminates a black object. The light is less, the receiver tube receives less infrared light, showing a large resistance, when there is no block, the infrared light is equivalent to the light reflected on the black object when reflected less light through the external circuit. You can read the test state; the same token, when irradiated on the white surface when the launch of the infrared more, the performance of the receiver tube resistance is small, this time through the external circuit can read another state, such as electricity Flat high and low to describe the above two phenomena will appear high and low level of points; if hand or white paper cover the module, the LED display goes out.

As shown:

(Sensing distance is adjusted by potentiometer, clockwise sensing distance becomes smaller and counterclockwise sensing distance becomes larger)

Example picture



The followings are the code used in this experiment and their explanations:

```
//define LED port
int Led=13;
//define switch port
int buttonpin=3;
//define digital variable val
int val;
void setup()
{
//define LED as a output port
pinMode(Led,OUTPUT);
//define switch as a output port
pinMode(buttonpin,INPUT);
Serial.begin(9600);
}
void loop()
{
//read the value of the digital interface 3 assigned to val
val=digitalRead(buttonpin);
//when the switch sensor have signal, LED blink
if(val==HIGH)
{
digitalWrite(Led,HIGH);
Serial.println("HIGH!");
}
else
{
digitalWrite(Led,LOW);
Serial.println("LOW!");
}
}
```

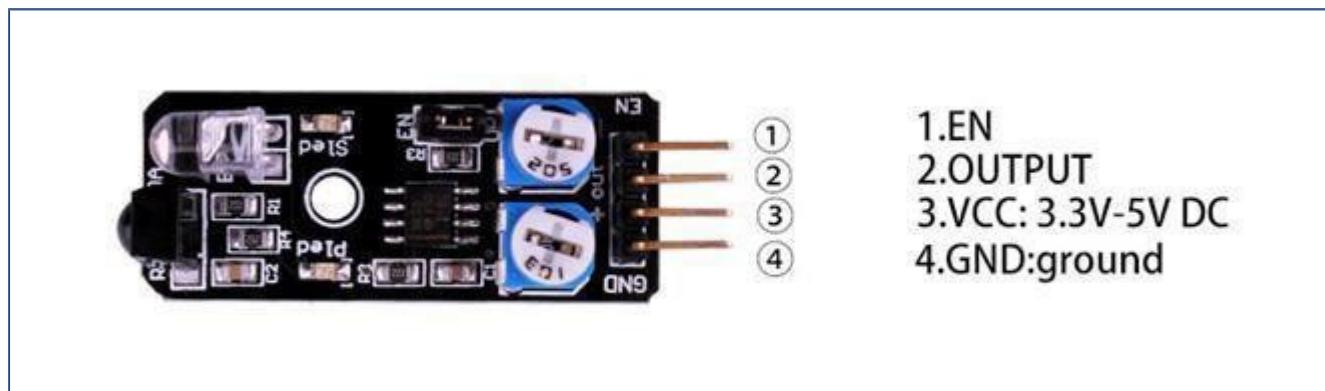
Lesson 25 Obstacle Avoidance Sensor Module

Overview

Non logic chip oscillation frequency regulation 38KHz detection circuit. This infrared 38KHZ obstacle avoidance module can completely block the traditional photoelectric obstacle detection distance and a single normally open or normally closed signal output function.

Obstacle avoidance sensor module

IR-reflection sensor, useful for obstacle avoidance applications. When an obstacle is in front of the IR sender/receiver the 'Out' pin is switched low (active low). The circuit sensitivity can be adjusted with a pot. The obstacle detection distance can be adjusted up to approximately 7cm. An enable (EN) jumper can be fitted for continuous operation. Removal of the EN jumper allows an external logic signal (at the EN pin) to switch the detector on and off (low = active, high = off).



Component Required:

- 1x Elegoo Uno R3
- 1x USB cable
- 1x Obstacle avoidance sensor module
- 4x F-M wires

Component Introduction

Obstacle avoidance module:

Specifications

Working voltage: DC3.3V-5V

Working current: ≥20mA

Operating temperature: -

10 °C - +50 °C

detection distance: 2-40cm

IO Interface: 4-wire

interfaces (- / + / S /EN)

Output signal: TTL level (low

level there is an obstacle, no

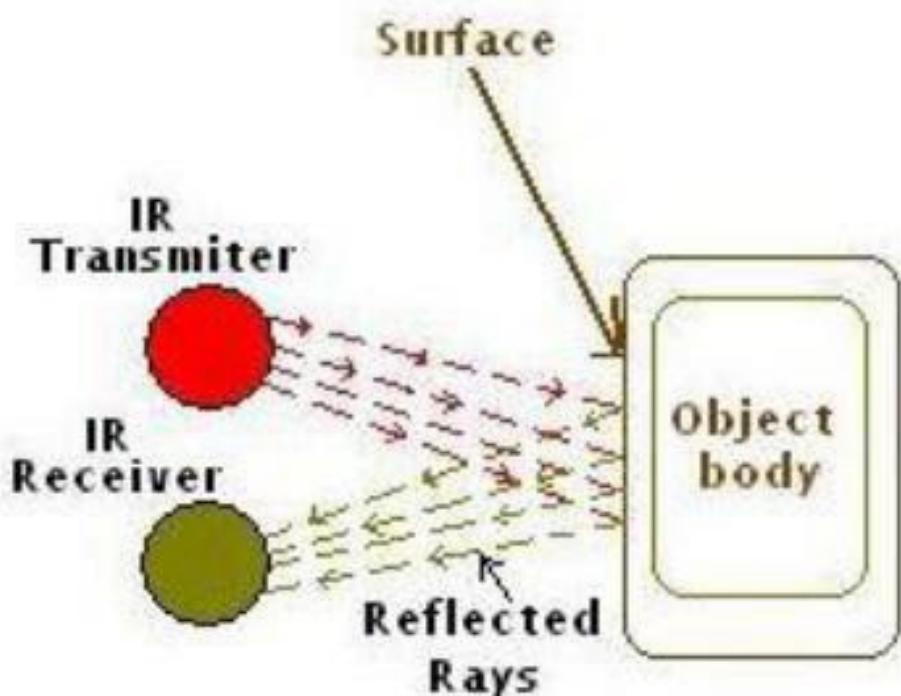
obstacle high)

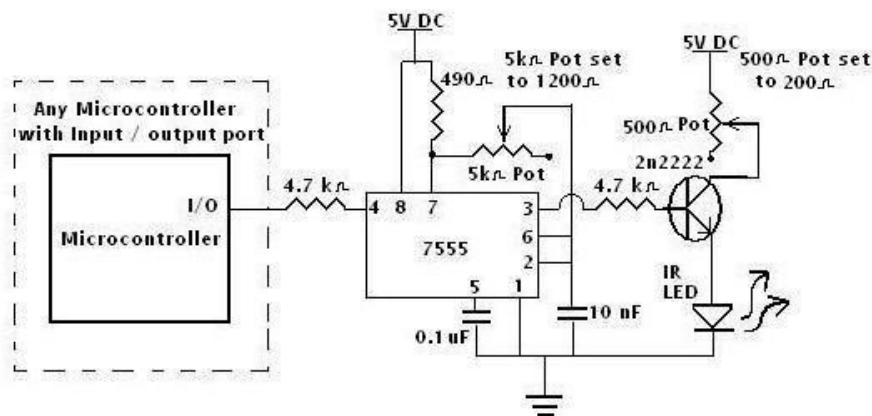
Adjustment: adjust multi-turn resistance

Effective angle: 35° Size: 28mm × 23mm

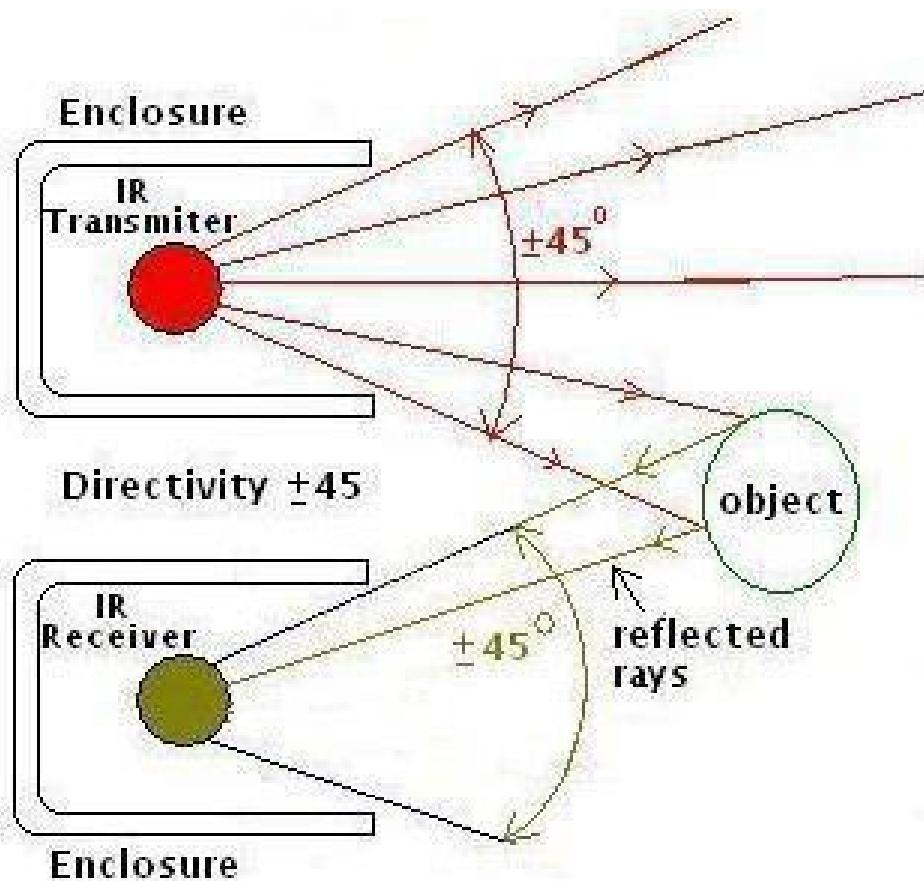
Weight Size: 9g

The basic concept of IR(infrared) obstacle detection is to transmit the IR signal(radiation) in a direction and a signal is received at the IR receiver when the IR radiation bounces back from a surface of the object.





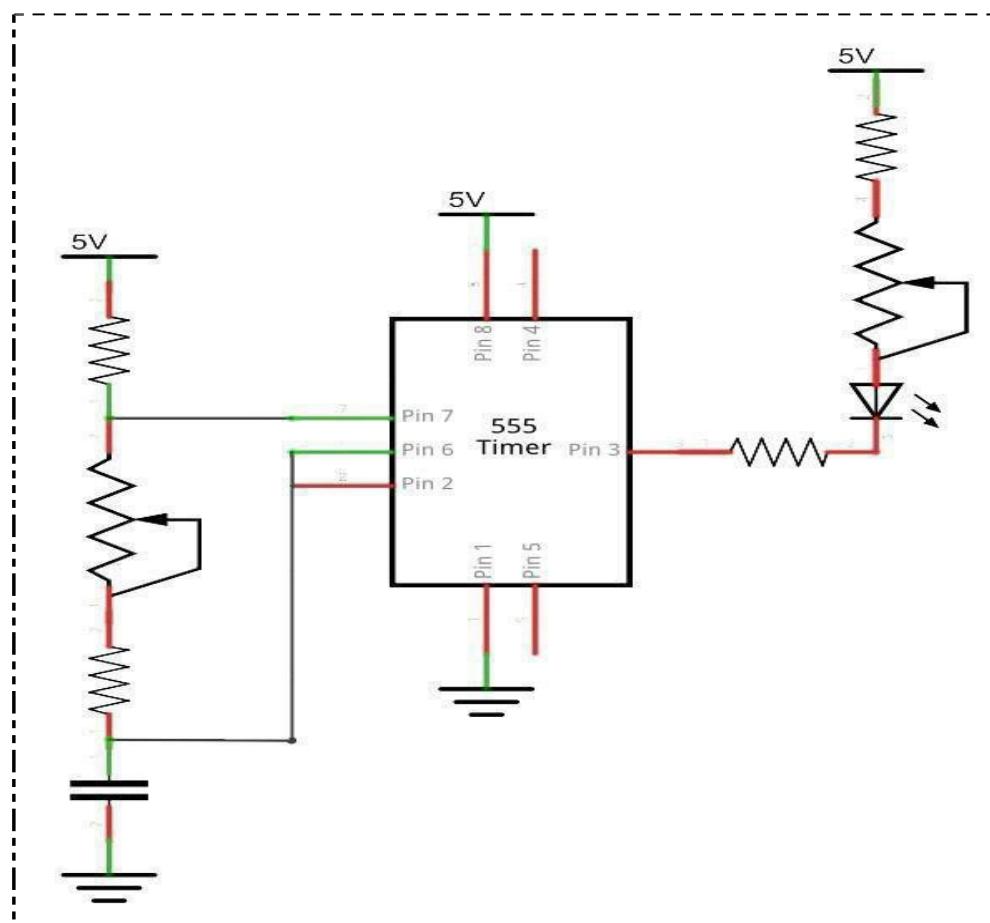
There are two potentiometers on the module one controlling operating frequency (centered at 38 kHz) the other controlling intensity. The detector was designed for 38 kHz and the onboard oscillator circuit is based on a 555 timer. Tweaking gives a little better range but I'd suggest leaving it alone because the useful range is narrow.

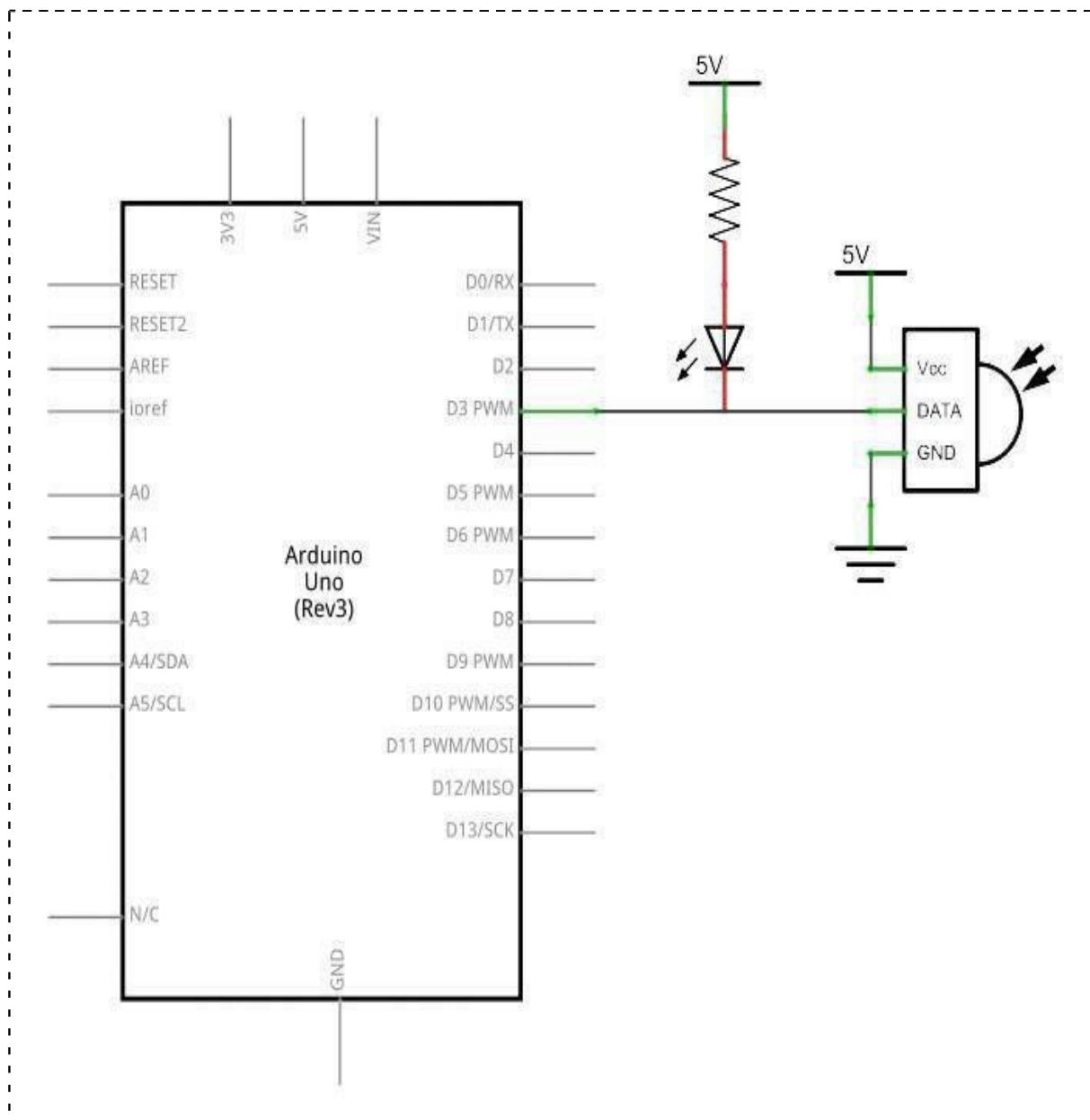


Infrared obstacle avoidance sensor is designed for the design of a wheeled robot obstacle avoidance sensor distance adjustable. This ambient light sensor Adaptable, high precision, having a pair of infrared transmitter and receiver, transmitter tubes emit a certain frequency of infrared, When detecting the direction of an obstacle (reflector), the infrared receiver tube receiver is reflected back, when the indicator is lit, Through the circuit, the signal output interface output digital signal that can be detected by means of potentiometer knob to adjust the distance, the effective distance From 2 ~ 40cm, working voltage of 3.3V-5V, operating oltage range as broad, relatively large fluctuations in the power supply voltage of the situation Stable condition and still work for a variety of microcontrollers, Arduino controller, BS2 controller, attached to the robot that Can sense changes in the ir surroundings.

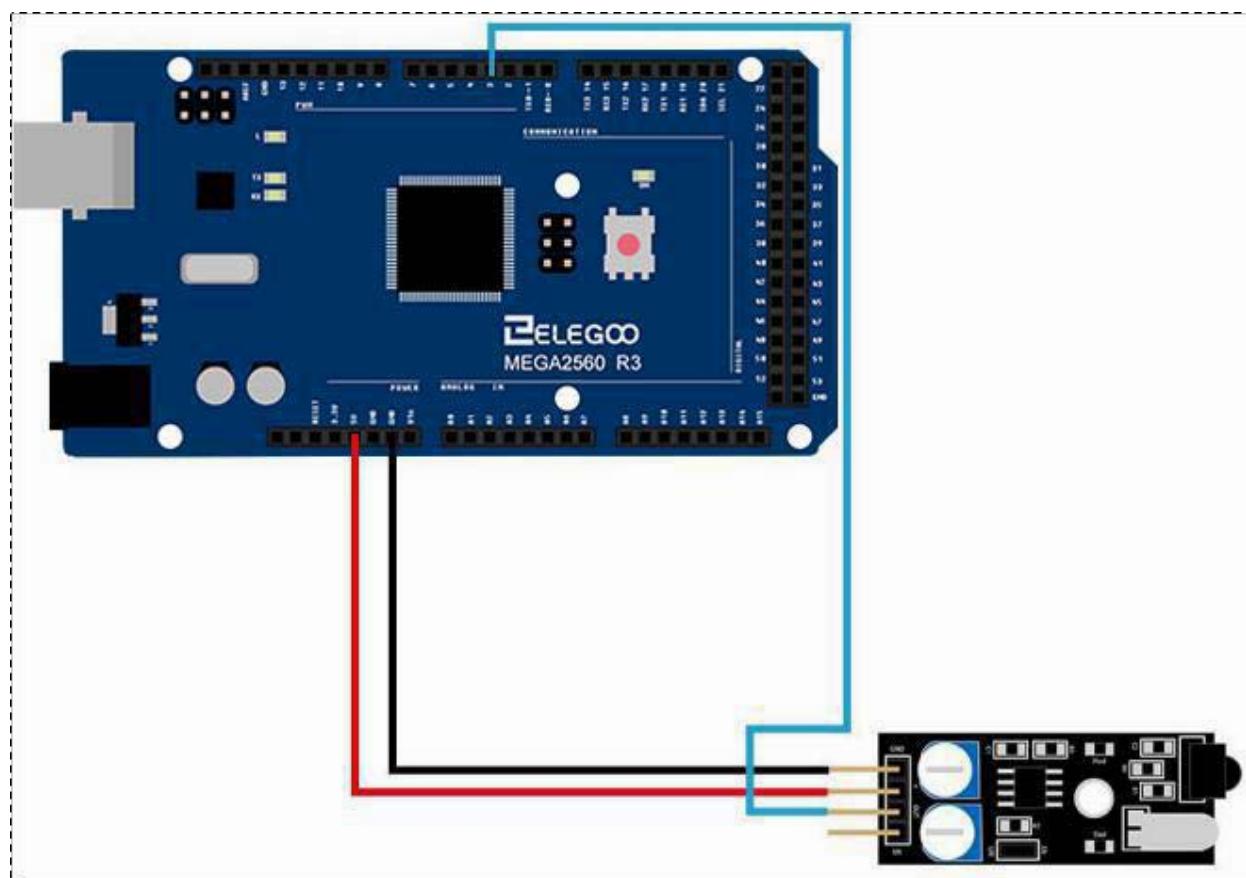
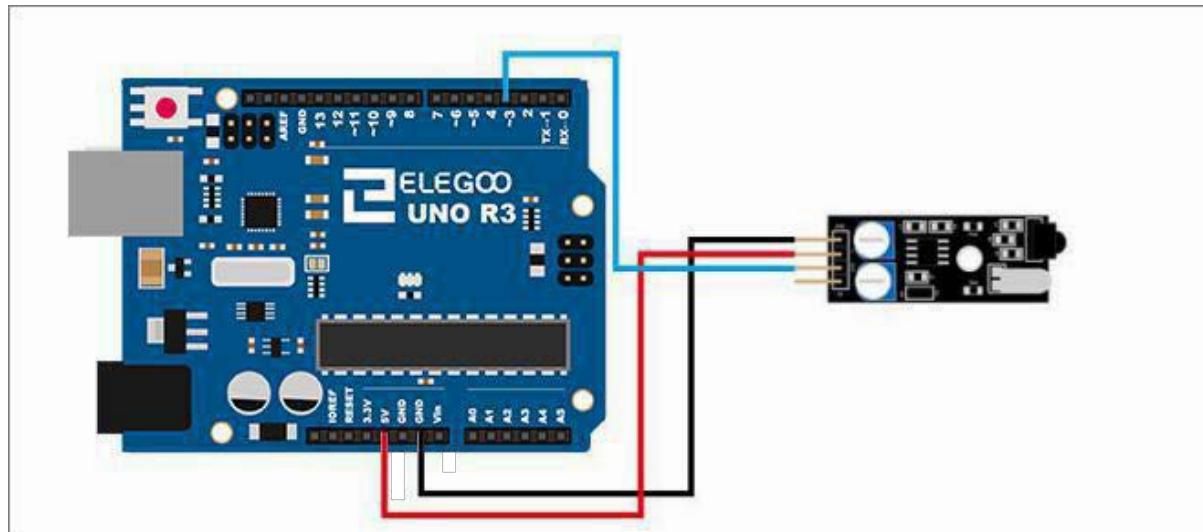
Connection

Schematic





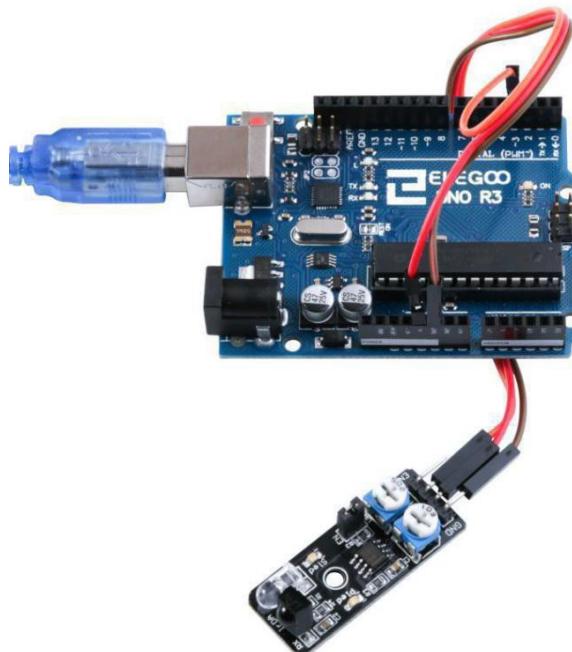
Wiring diagram



Code

When we connect the circuit, open the "code" folder of our tutorial data and find the "Lesson 25 Obstacle Avoidance Sensor Module" folder and run the compiler uploader, and we block the front of it by hand and you can see our led Bright phenomenon.

Example picture

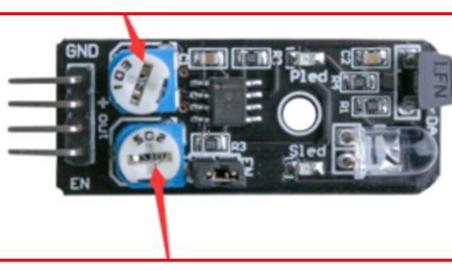


Here we use the obstacle avoidance module and a digital interface, built-in 13 LED build a simple circuit, making avoidance warning lamp, the obstacle avoidance Sensor Access Digital 3 interface, when obstacle avoidance sensor senses a signal, LED light, and vice versa off.

You need to adjust the 5K and 10K potentiometers first before you use obstacle avoidance module. Generally, you need to tweak the 5K potentiometer clockwise at about 90° and the 10K potentiometer at $10-30^\circ$. You need to be adjust them with patience to make it work.

As shown:

10K potentiometers: Ajust it clockwise at a angle between 10 to 30 degrees.



5K potentiometers: Ajust it clockwise at a angle about 90 degrees.

The followings are the code used in this experiment and their explanations:

```
//define LED port
int Led=13;
//define switch port
int buttonpin=3;
//define digital variable val
int val;
void setup()
{
//define LED as a output port
pinMode(Led,OUTPUT);
//define switch as a output port
pinMode(buttonpin,INPUT);
}
void loop()
{
//read the value of the digital interface 3 assigned to val
val=digitalRead(buttonpin);
//when the switch sensor have signal, LED blink
if(val==HIGH)
{
digitalWrite(Led,HIGH);
}
else
{
digitalWrite(Led,LOW);
}
}
```

Lesson 26 Rotary Encode Module

Overview

In this experiment, we will learn how to use the rotary encode module.

Rotary encoder

Rotary encoder useful for making an electronic pot etc. Connection idents are printed on the PCB.



- 1.CLK
- 2.DT
- 3.SW
- 4.VCC:3.3V-5V DC
- 5.GND:ground

Component Required:

1x Elegoo Uno R3

1x USB cable

1x Rotary Encoders module

5x F-M wires

Component Introduction

Rotary Encoders:

Mechanical Specifications:

- Operating Temp: -10°C to 70°C
- Storage Temp: -40°C to 85°C
- Rotational Torque: 50gf.cm max.
- No. and Pos. of detents: 12 detents (Step angle 30°±3°)
- Terminal Strength: A static load for 300gf.cm shall be applied to the tip of the terminals for 10 sec. in any direction
- Shaft push-pull strength: 5.1kgf
- Rotational life: 30,000 cycles

Note:

- RoHS Compliant

Electrical Specifications:

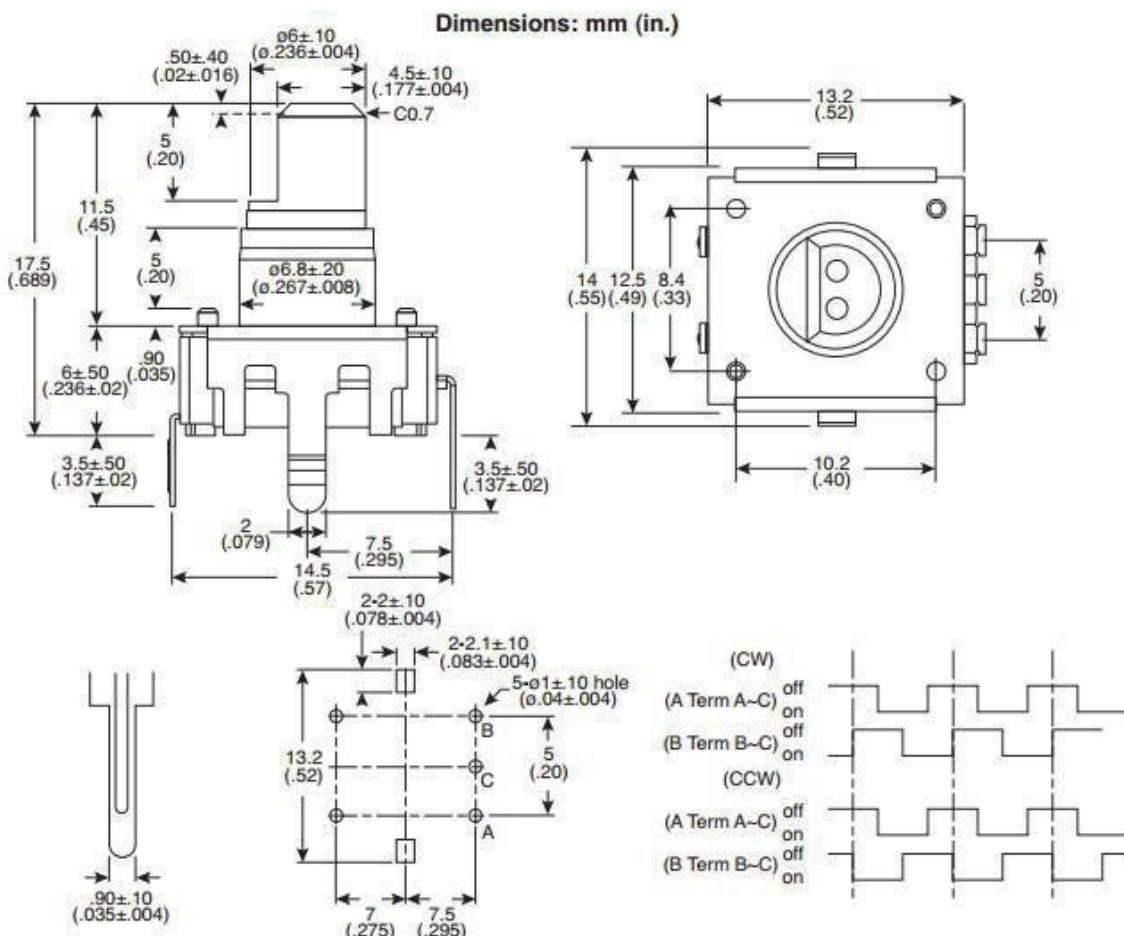
- Rating: 1mA/10VDC
- Insulation Resistance: 50VDC 10MΩ Min.
- Dielectric Strength: 50VAC for 1 min.
- Resolution: 12 pulses/360° for each phase

Soldering Specifications:

- Soldering: To be performed in 5 seconds within 260±5°C
- Manual Soldering: To be performed in 3 seconds within 350±5°C
- Preheating: The entire flow duration should not exceed 2 min., and soldering surface temperature (undersurface of PCB) shall be settled within 100°C

Push-on Switch Specifications:

- Type: Single Pole Single Throw (Push on)
- Rating: 10mA/5VDC
- Switch Travel (mm): 0.5±0.4
- Operating Force: 200~460gf
- Operating Life: 20,000 times



Principle

Incremental encoder

Incremental encoders give two-phase square wave, the phase difference between them 90 °, often referred to as A and B channels. One of the channels is given and speed-related Information, at the same time, by sequentially comparing two channel signals, the direction of rotation of the information obtained. There is also a special signal called Z or Zero channel, which gives the absolute zero position encoder, the signal is a square wave with the center line of channel A square wave coincide.

Clockwise/counterclockwise

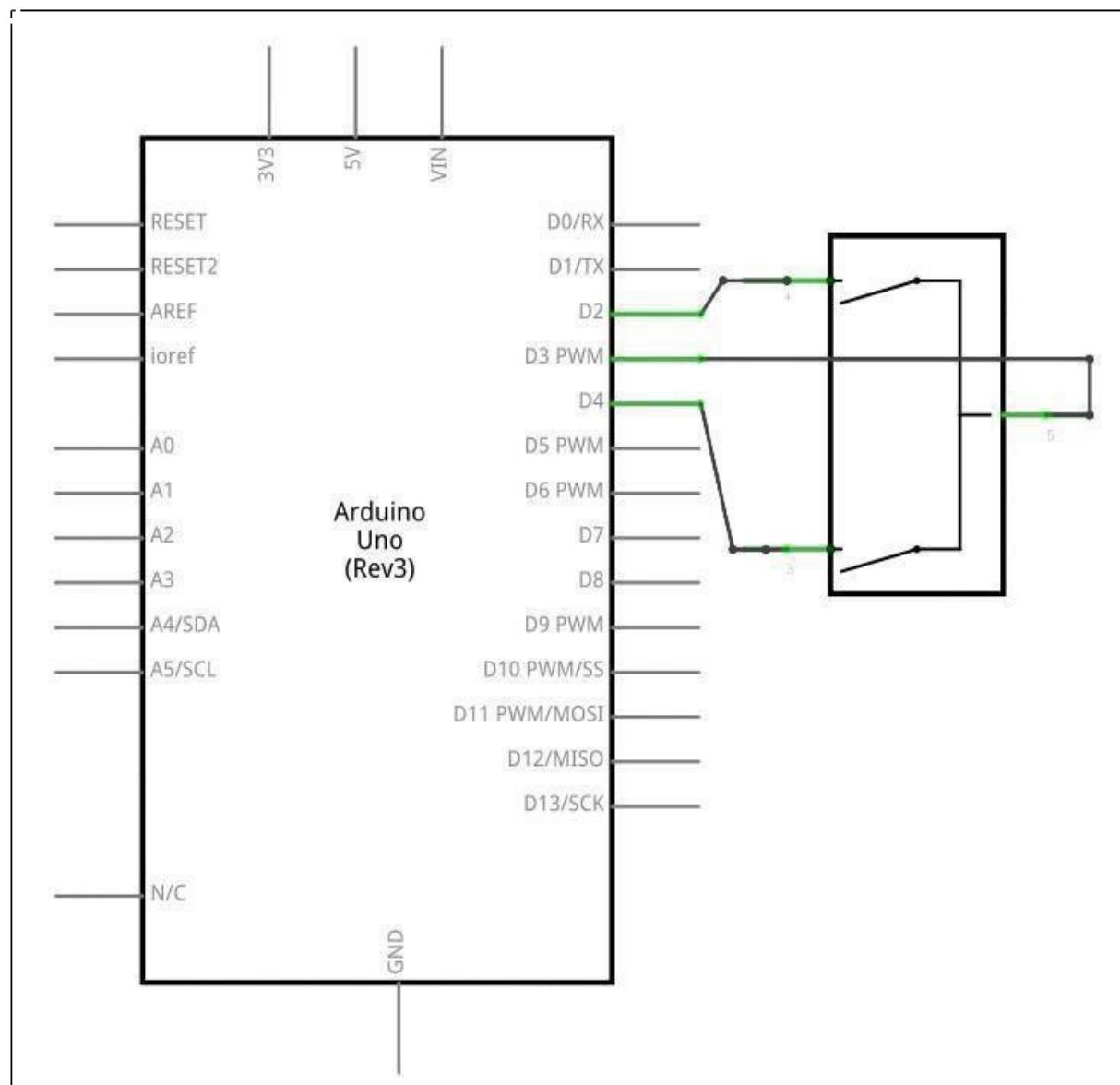
A	B
1	1
0	1
0	0
1	0
1	1
1	0
0	0
0	1

Incremental encoder accuracy depends on the mechanical and electrical two factors, these factors are: Raster indexing error, disc eccentricity, bearing eccentricity, e-reading Several means into the optical portion of the errors and inaccuracies. Determine the encoder resolution is measured in electrical degrees, the encoder accuracy depends Set the pulse encoder generates indexing. The following electrical degrees with a 360 ° rotation of the shaft to said machine, and rotation of the shaft must be a full week of Period. To know how much electrical equivalent of the mechanical angle of 360 degrees can be calculated with the following formula: Electrical 360 = Machine 360 ° / n ° pulses /revolution

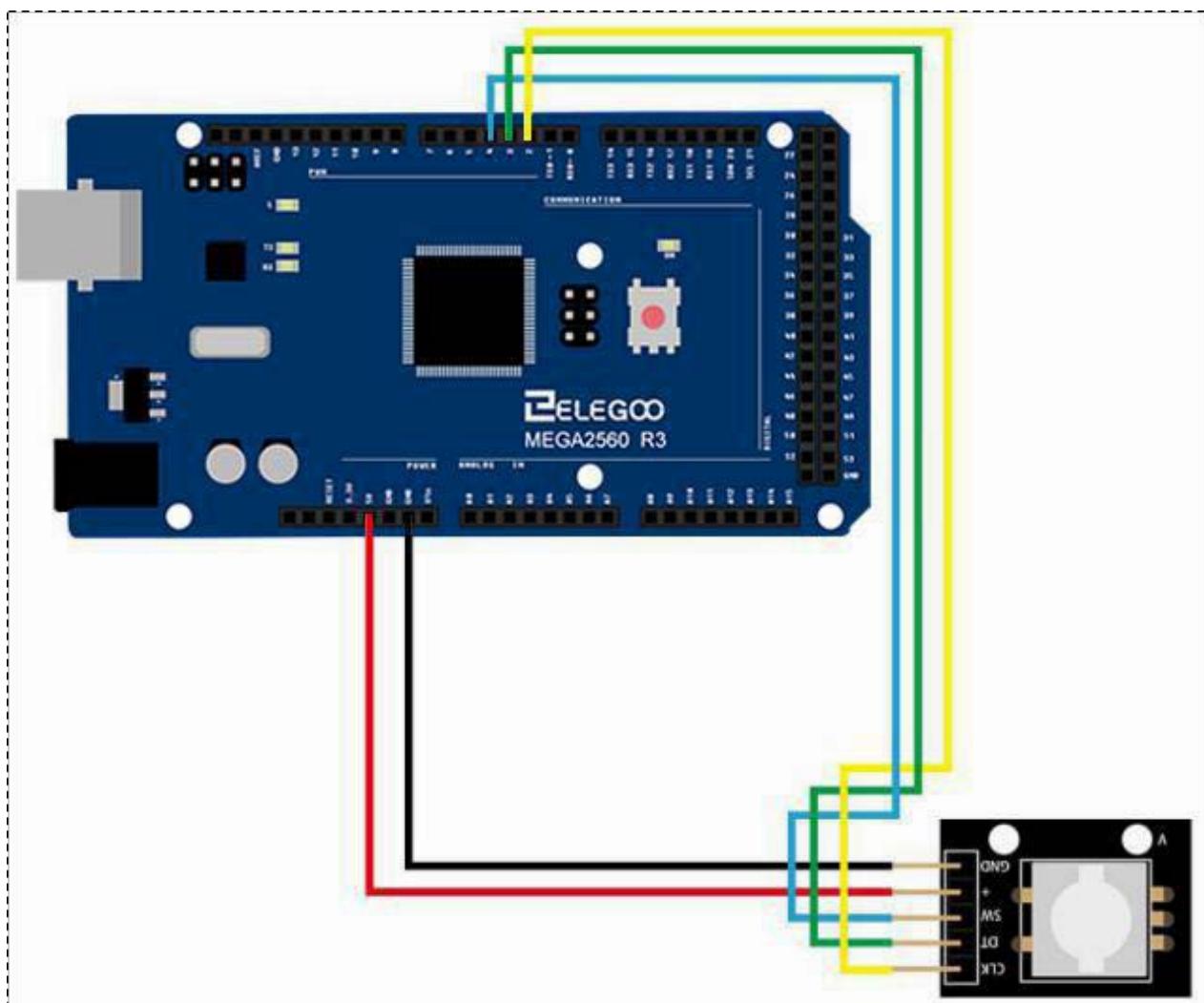
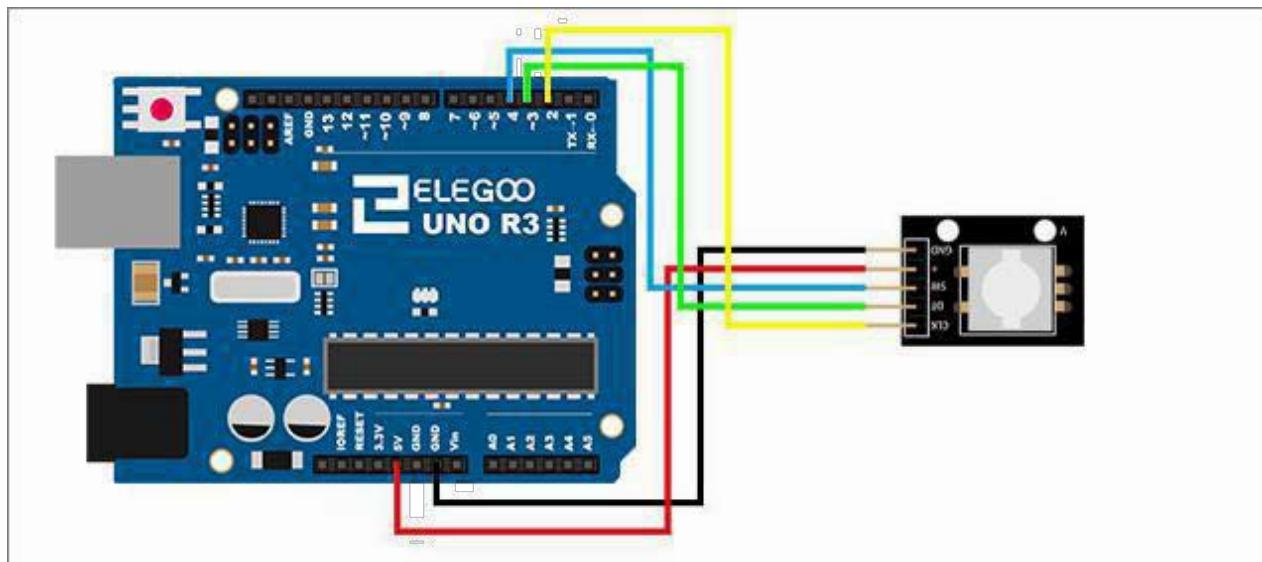
Encoder indexing error is the electrical angle of the unit two successive pulse maximum offset to represent. Error exists in any encoder, which is caused by the aforementioned factors. Eltra encoder maximum error is ± 25 electrical degrees (declared in any condition), equivalent to the rated Offset values $\pm 7\%$, as the phase difference 90° (electrical) of the two channels of the maximum deviation ± 35 electrical degrees is equal to $\pm 10\%$ deviation left Ratings Right.

Connection

Schematic



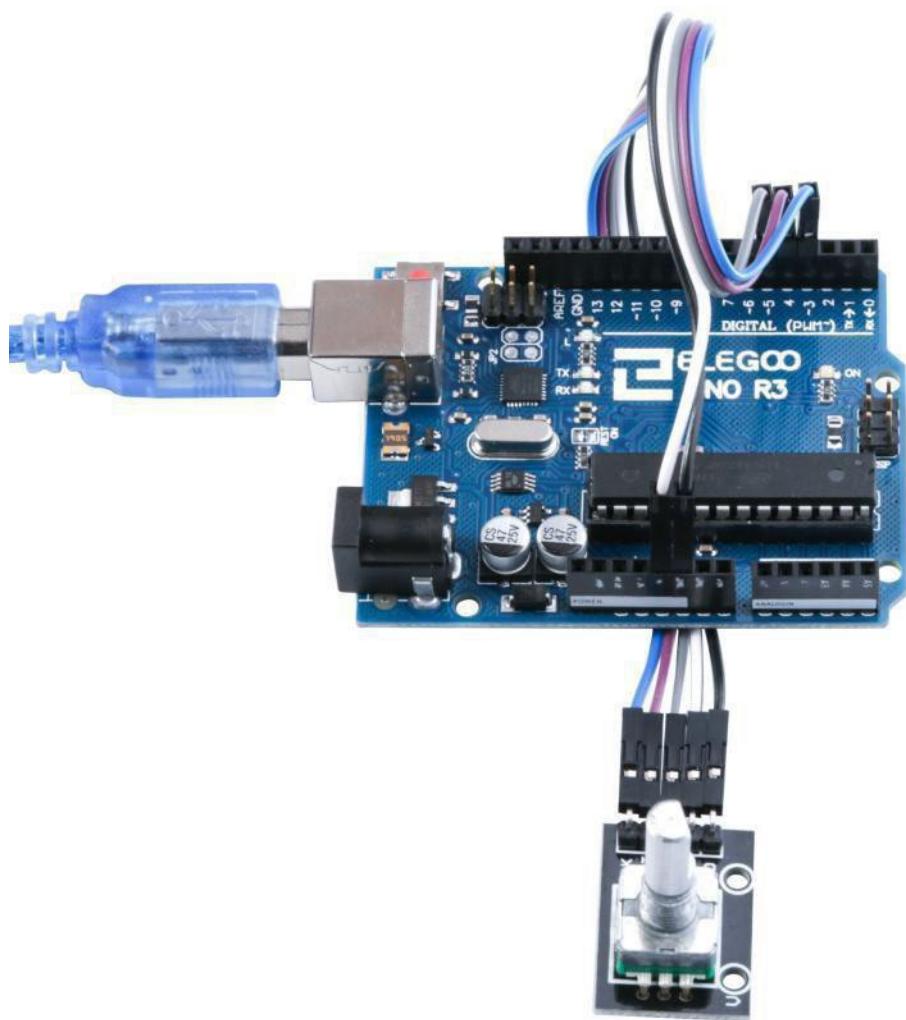
Wiring diagram



Code

After you finish the wiring, open the tutorial file and go to code > Lesson 26 Rotary Encode Module and run the program (the .ino file). Then go to the serial monitor and tweak the rotary encoder and you will see the following data.

Example picture



253/332

The followings are the code used in this experiment and their explanations:

```
//Define the pin connection
int CLK = 2;//CLK->D2
int DT = 3;//DT->D3
int SW = 4;//SW->D4
// Interrupt 0 on pin 2
const int interrupt0 = 0;

//Define the count
int count = 0;
//CLK initial value
int lastCLK = 0;
void setup()
{
    pinMode(SW, INPUT);
    digitalWrite(SW, HIGH);
    pinMode(CLK, INPUT);
    pinMode(DT, INPUT);
//Set the interrupt 0 handler, trigger level change
    attachInterrupt(interrupt0, ClockChanged, CHANGE);

    Serial.begin(9600);
}

void loop()
{//Read the button press and the count value to 0 when the counter reset
    if (!digitalRead(SW) && count != 0)
    {
        count = 0;
        Serial.print("count:");
        Serial.println(count);
    }
}
```

```
}

//The interrupt handlers

void ClockChanged()
{
    //Read the CLK pin level
    int clkValue = digitalRead(CLK);

    //Read the DT pin level
    int dtValue = digitalRead(DT);

    if (lastCLK != clkValue)
    {
        lastCLK = clkValue;

        //CLK and inconsistent DT + 1, otherwise - 1
        count += (clkValue != dtValue ? 1 : -1);

        Serial.print("count:");
        Serial.println(count);
    }
}
```

Lesson 27 Relay Module

Overview

In this experiment, we will learn how to use relay module.

A relay is a type of component which responds to a change in input variables based on specified parameters in the input circuit. The output circuit switch on as soon as the conditions are met. Our company produces relay modules at anywhere from 28V to 240V, both AC and DC. It can be used in combination with MCU to create a timing control switch. Relays are used in everything from alarm systems to toys to construction equipment. It is an electrical control device with two control systems: the input circuit and output circuit, which interact to achieve a desired output.

Relay module

A relay module suitable for direct connection to an Arduino board. The module requires 5 V power supply. The input control signal is identified with an 'S'. The relay has one change-over contact. It is capable of switching resistive loads up to 10 A at 250 VAC and up to 10 A at 30 V maximum.



- 1.GND:ground
- 2.VCC:3.3V-5V DC
- 3.OUTPUT

Component Required:

- 1x Elegoo Uno R3
- 1x USB cable
- 1x 1 channel relay module
- 3x F-M wires

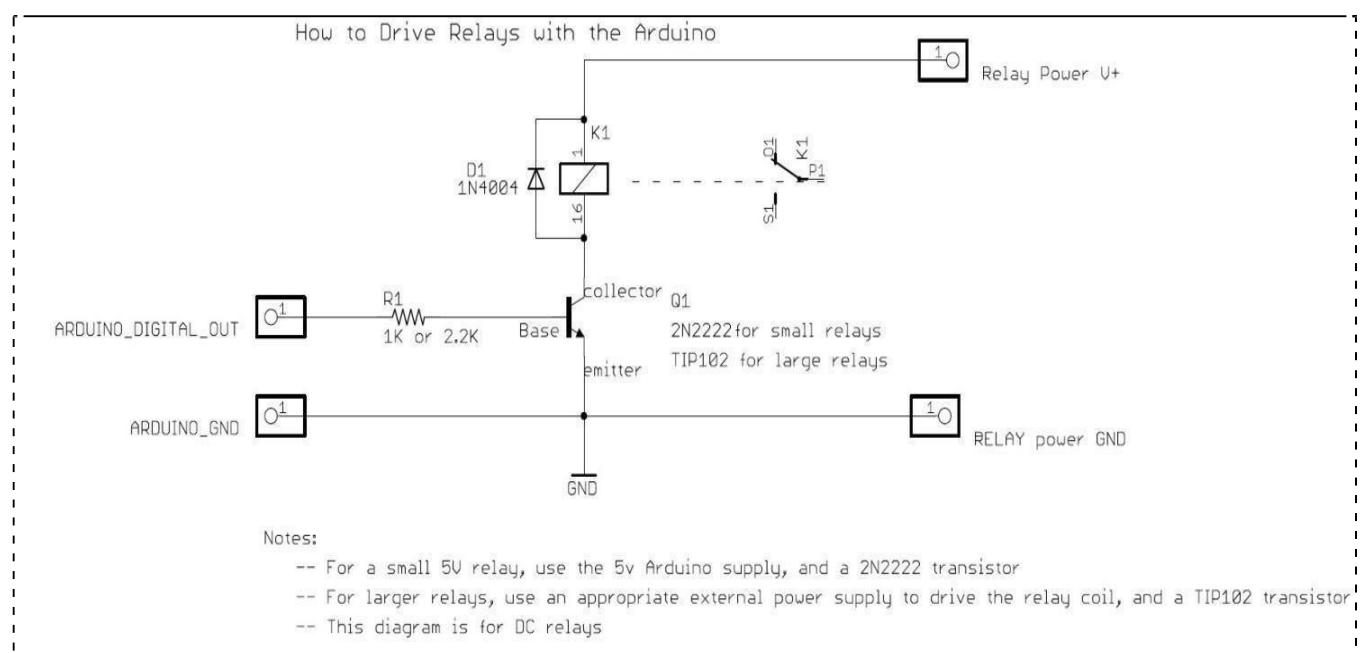
Component Introduction

Relay:

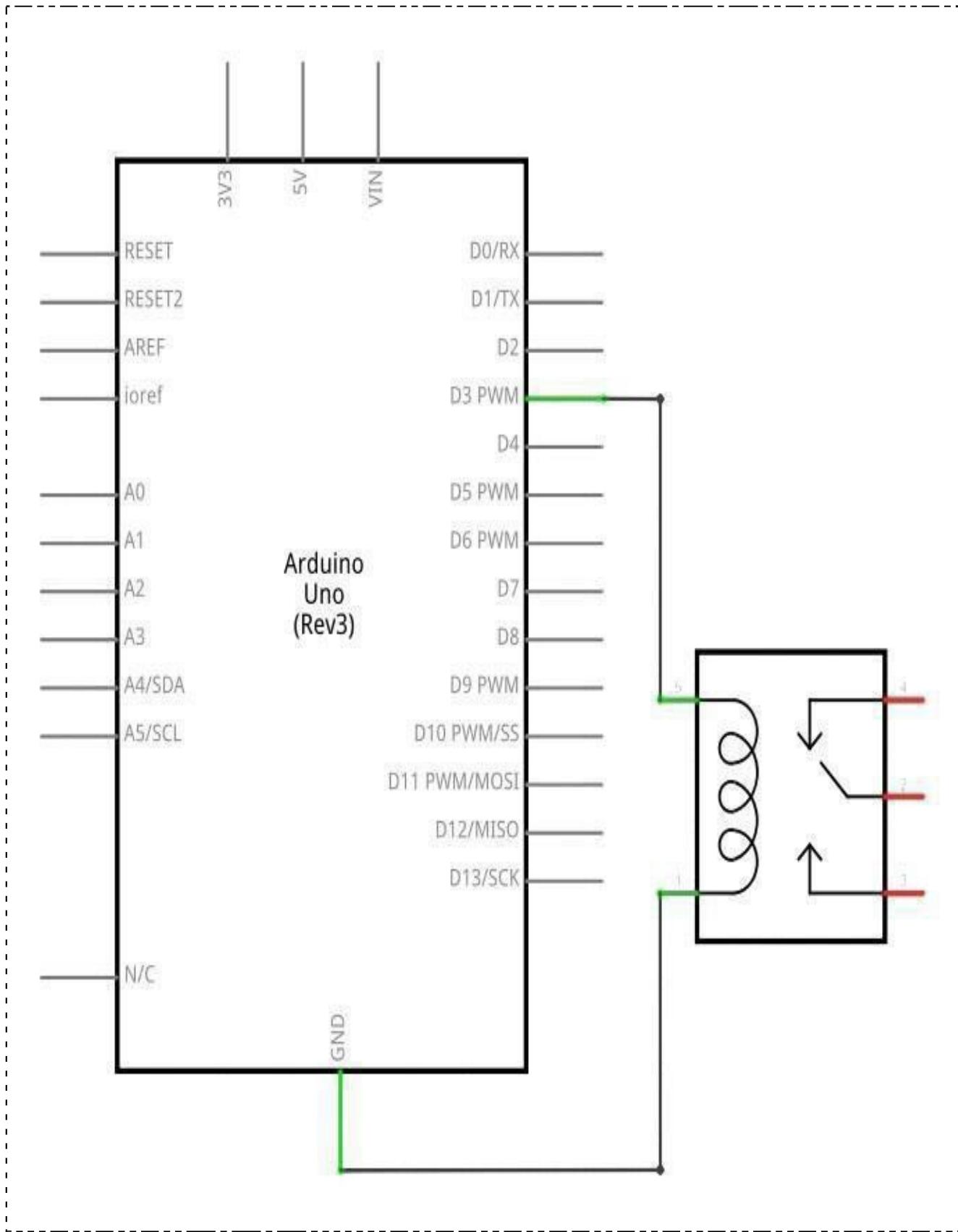
A relay is an electrically operated switch. Many relays use an electromagnet to mechanically operate a switch, but other operating principles are also used, such as solid-state relays. Relays are used where it is necessary to control a circuit by a low-power signal (with complete electrical isolation 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 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 still called "protectiverelays".

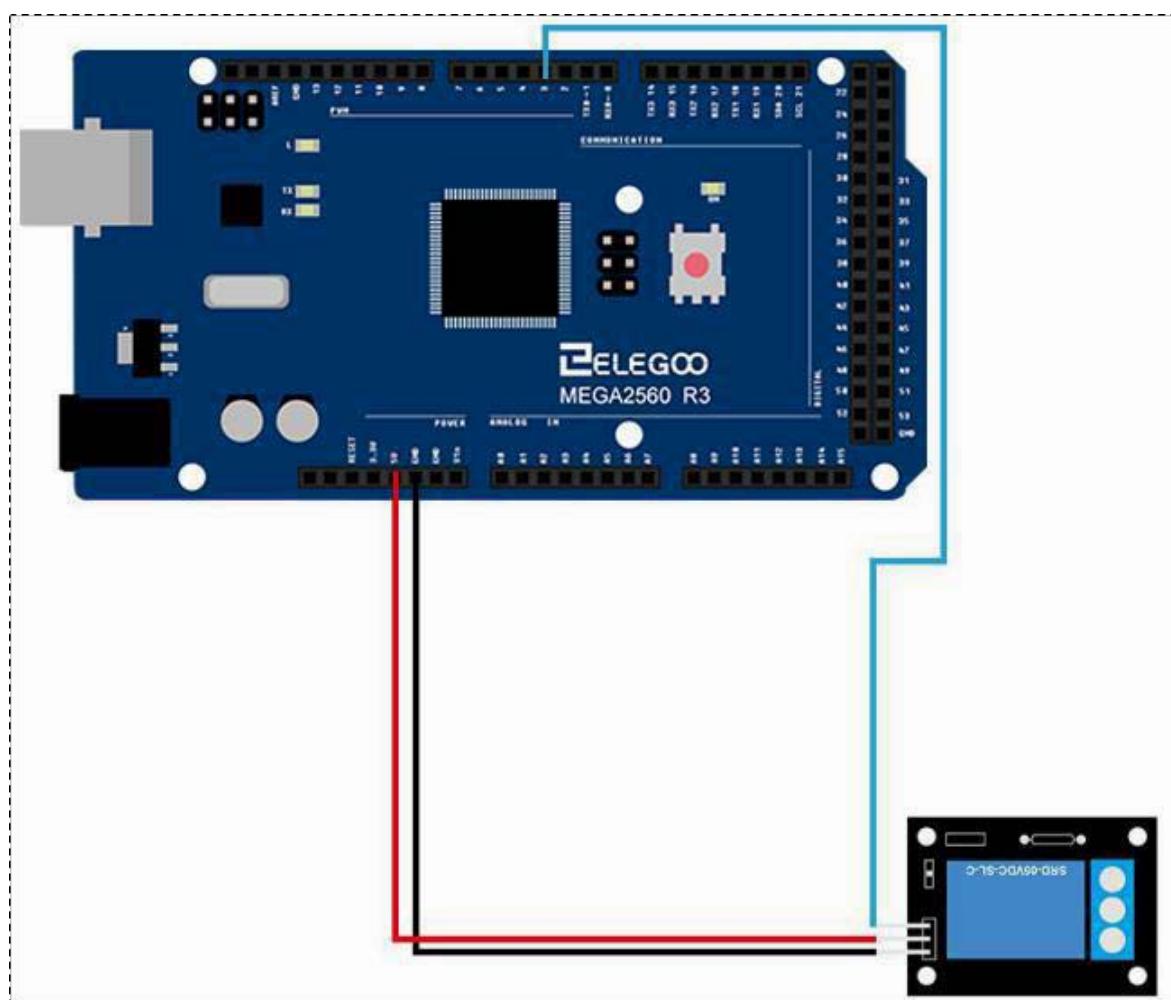
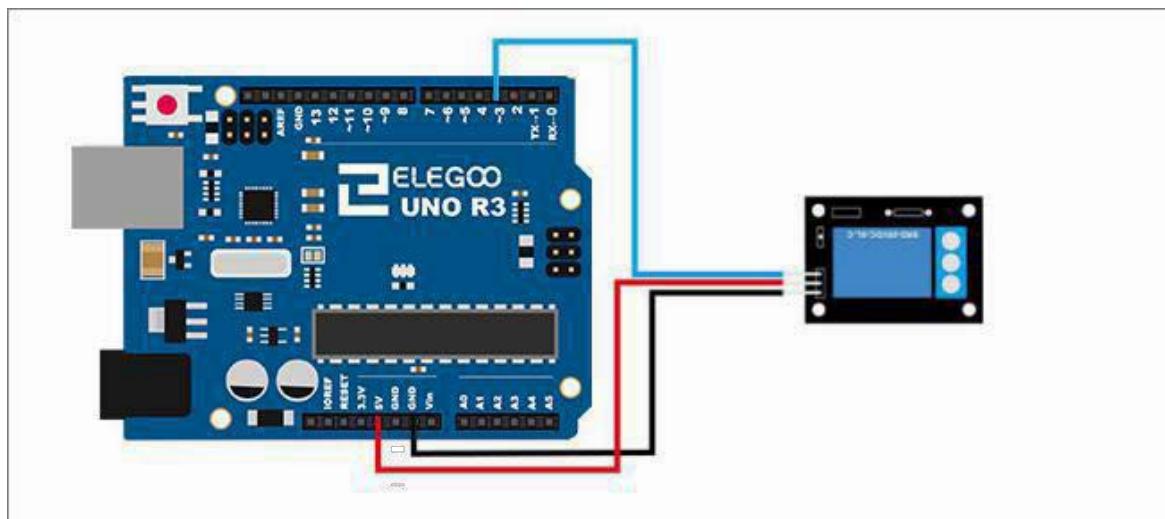
Bellow is the schematic of how to drive relay with arduino (Download from thearduino.cc)



Connection Schematic



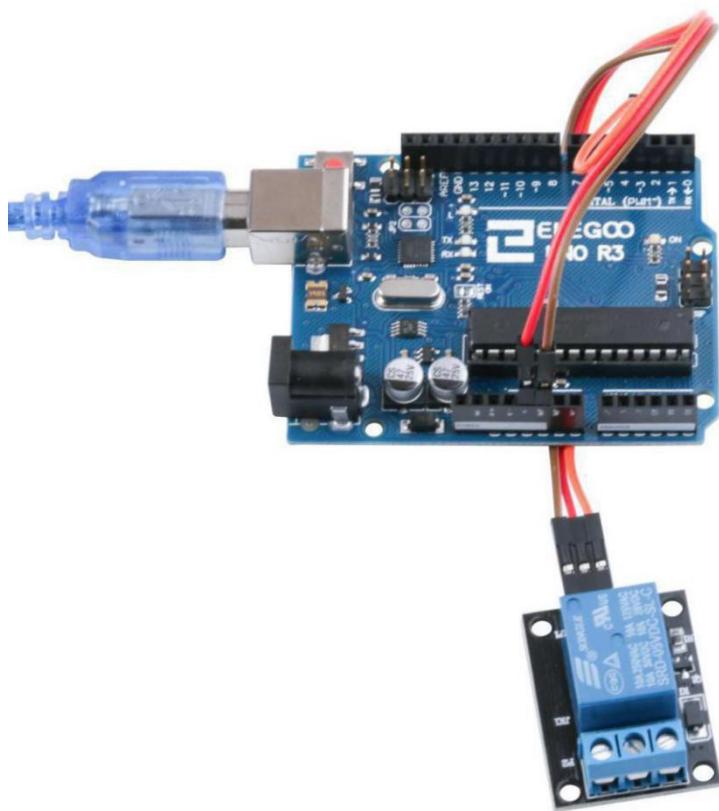
Wiring diagram



Code

After we connect the circuit, open our "code" folder and find the "Lesson 27 1 CHANNEL RELAY MODULE" folder. Open and run the compiler uploader, and we hear the ticking of the relay as the relay contacts The noise that is absorbed by the coil.

Example picture



The followings are the code used in this experiment and their explanations:

```
int relayPin = 3;  
  
void setup()  
{  
    pinMode(relayPin, OUTPUT);  
}  
  
void loop()  
{  
    // turn the relay on (HIGH is the voltage level)  
    digitalWrite(relayPin, HIGH);  
    // wait for a second  
    delay (2000);  
    // turn the relay off by making the voltage LOW  
    digitalWrite(relayPin, LOW);  
    // wait for a second  
    delay (2000);  
}
```

Lesson 28 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:

1x Elegoo Uno R3

1x LCD1602 module

1 x Resistor(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

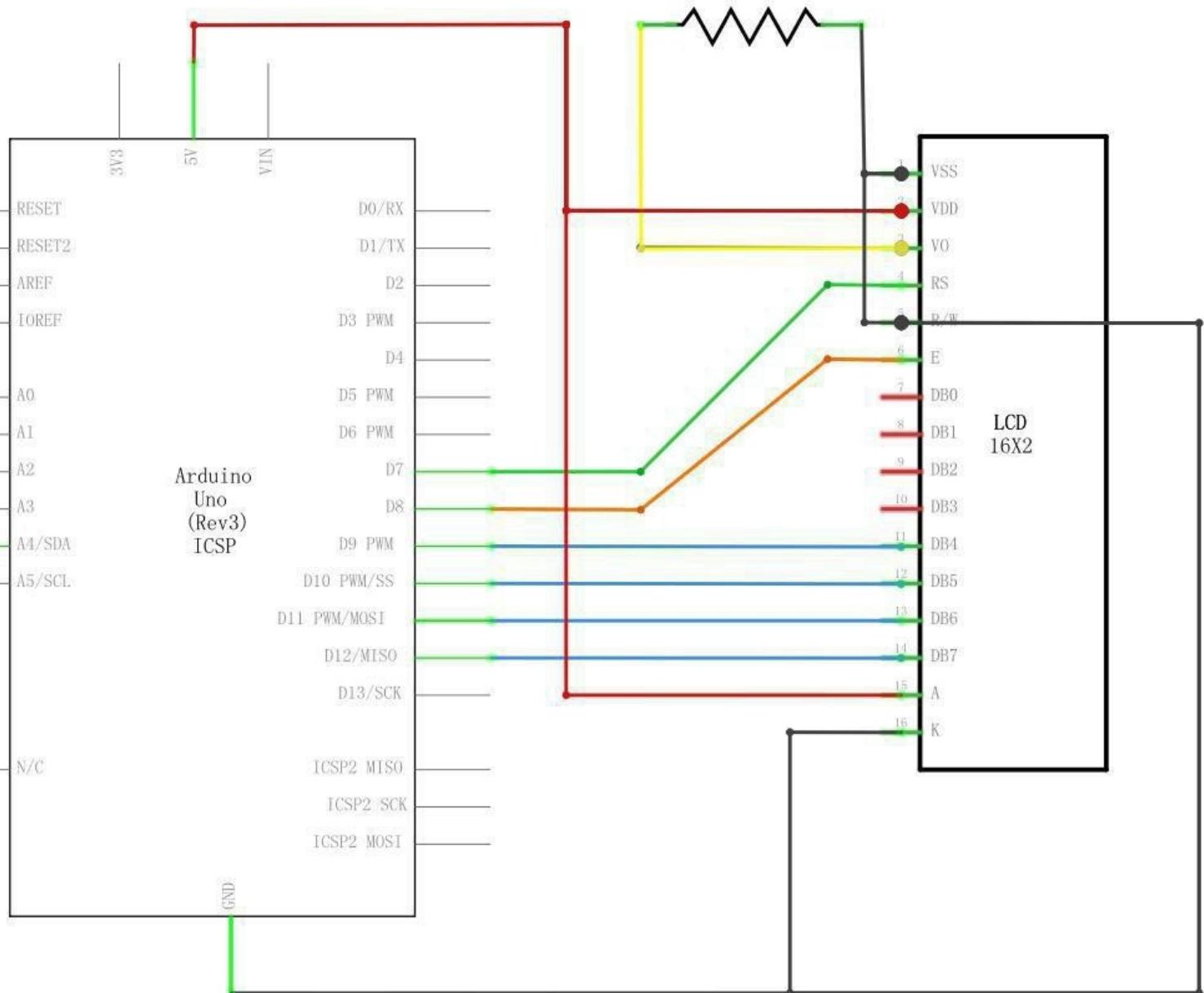
E: An enabling pin that, when supplied with low-level energy, causes the LDC module to execute relevant instructions.

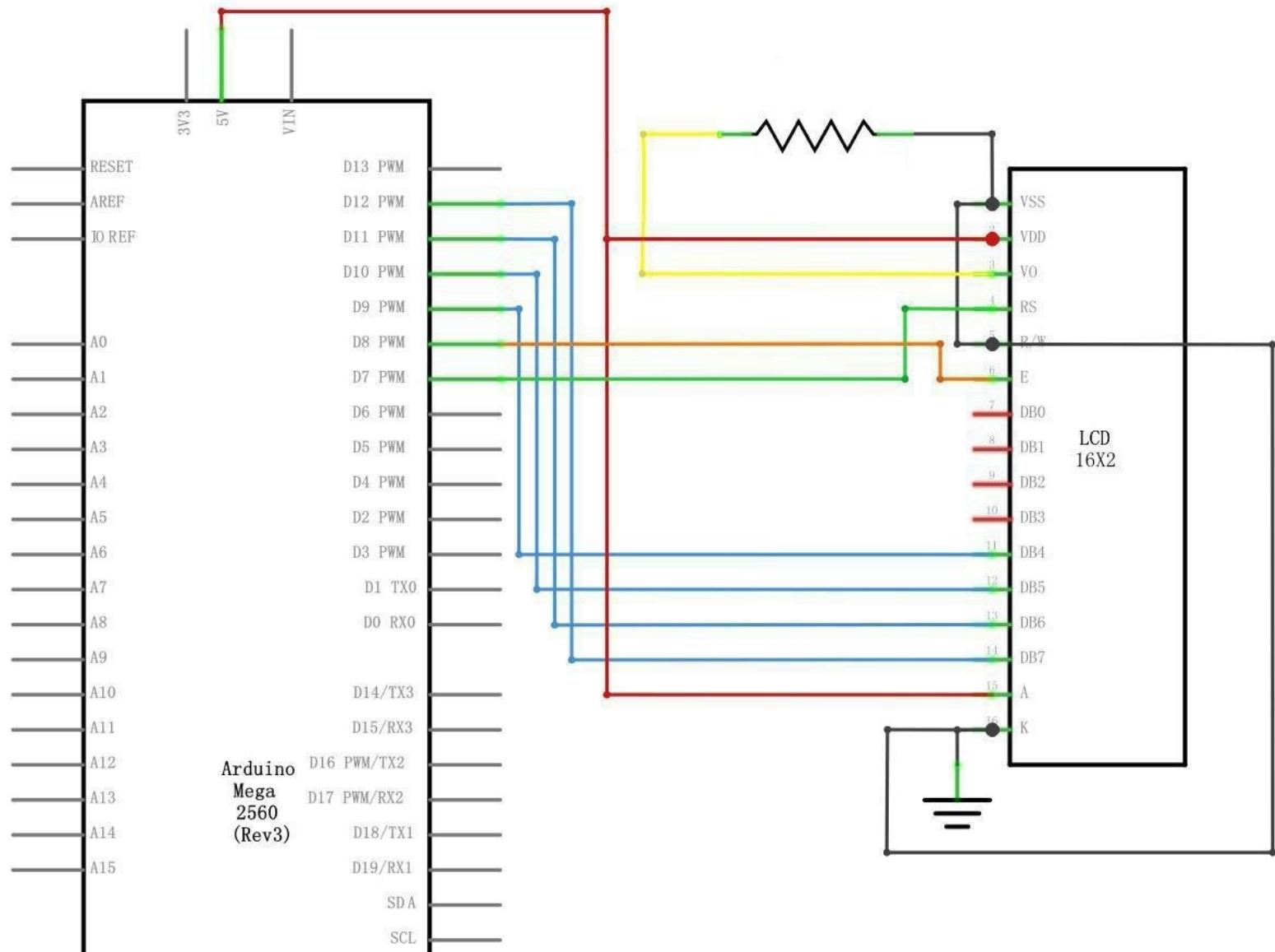
D0-D7: Pins that read and write data

A and K: Pins that control the LED back-light

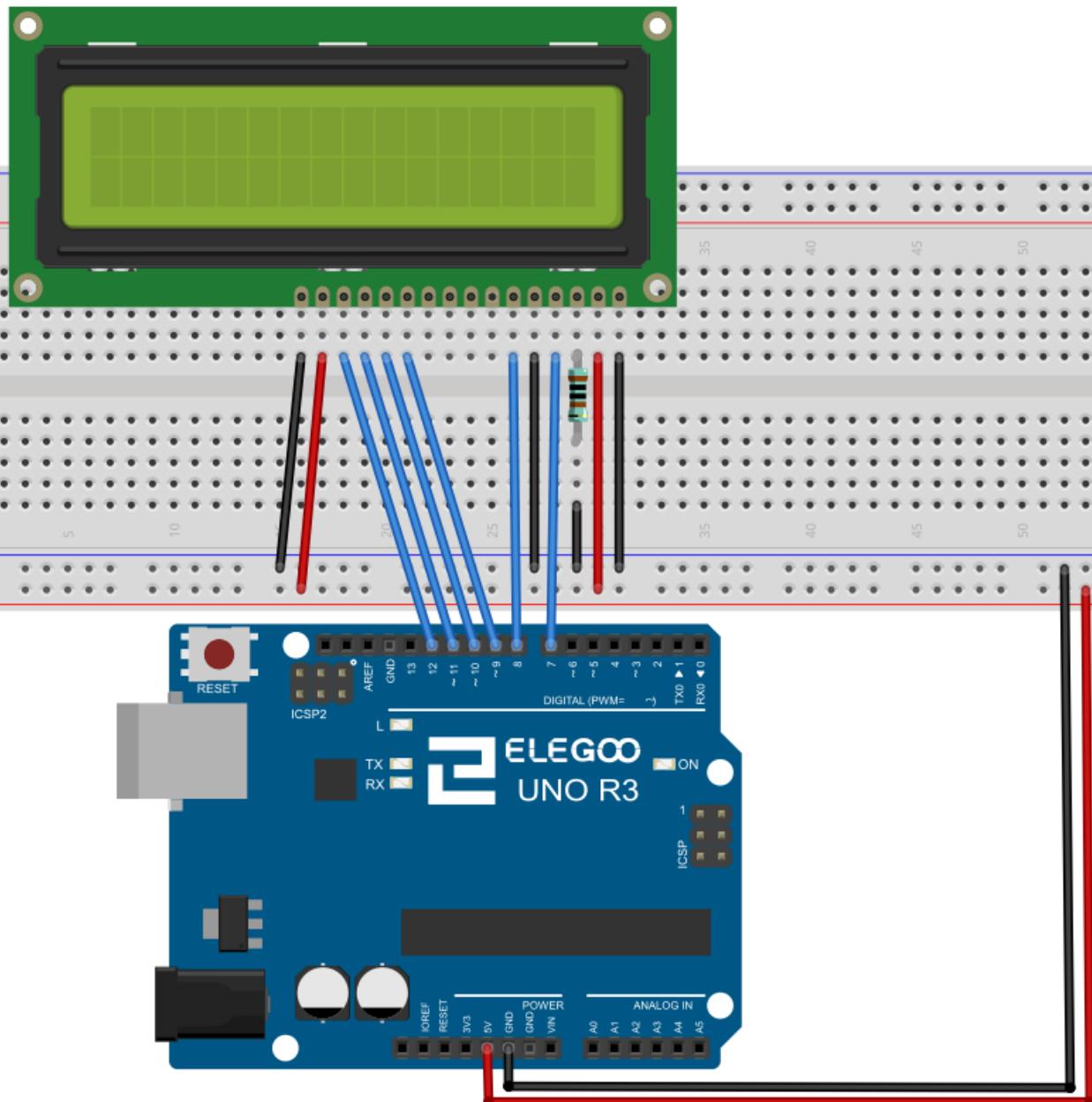
Connection

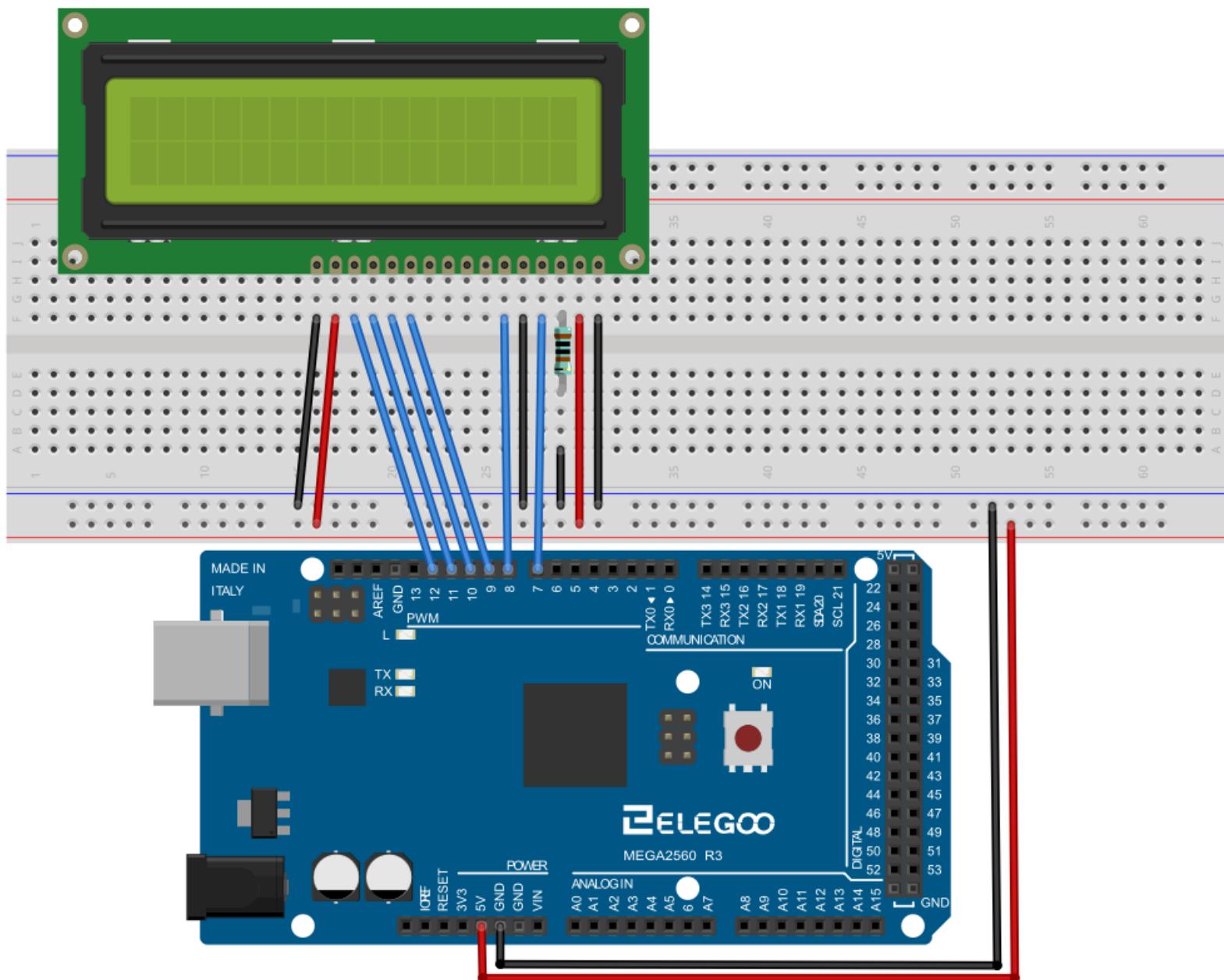
Schematic





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.

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 28 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 < Liquid Crystal > 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.

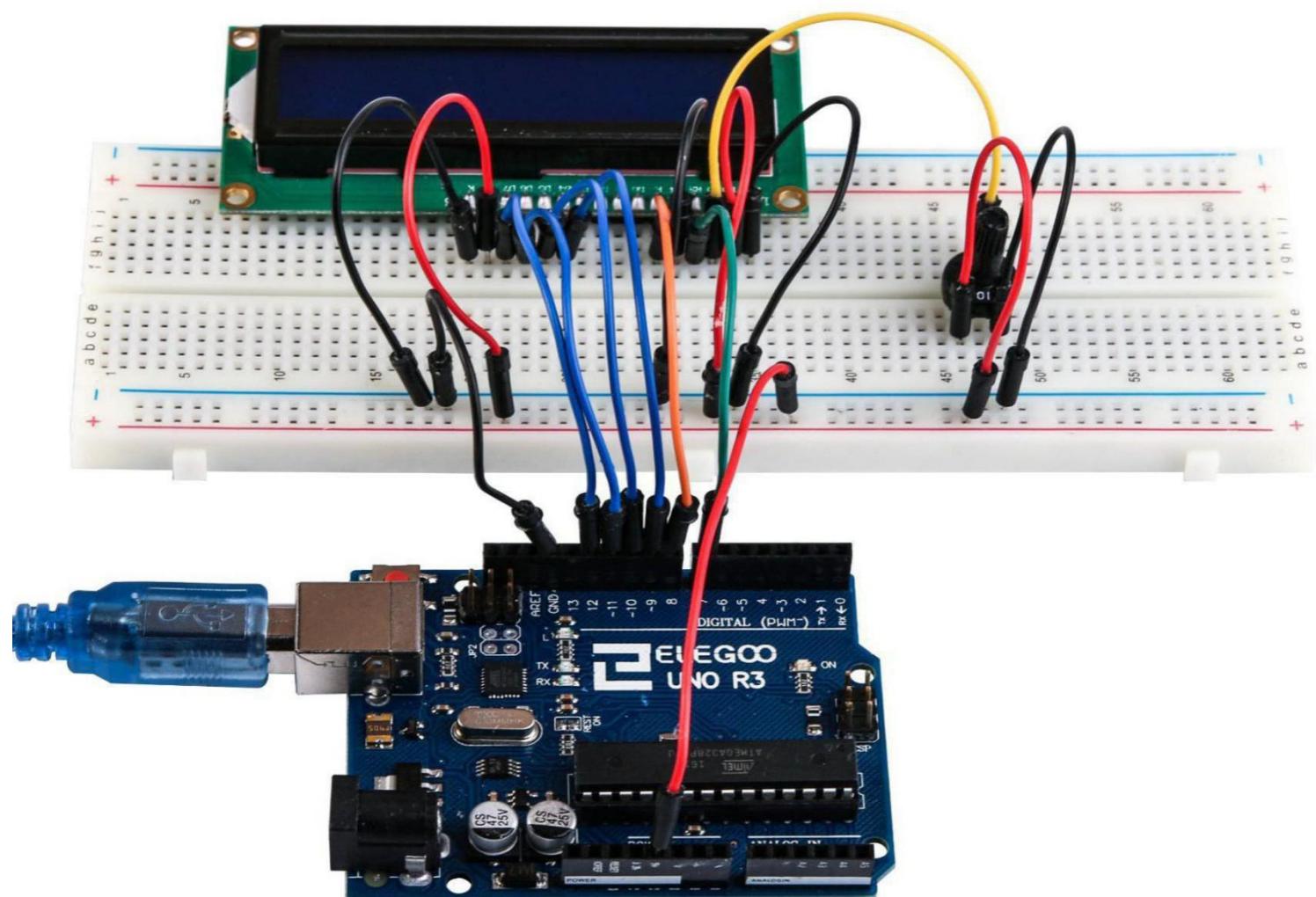
The second line displays the message that we see on the first line of the screen.

In the 'loop' function, we also 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



The followings are the code used in this experiment and their explanations:

// include the library code:

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

// initialize the library with the numbers of the interface pins

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

```
void setup() {
```

// set up the LCD's number of columns and rows:

```
lcd.begin(16, 2);
```

// Print a message to the LCD.

```
lcd.print("hello, world!");
```

```
}
```

```
void loop() {
```

// set the cursor to column 0, line 1

// (note: line 1 is the second row, since counting begins with 0):

```
lcd.setCursor(0, 1);
```

// print the number of seconds since reset:

```
lcd.print(millis() / 1000);
```

```
}
```

Lesson 29 Ultrasonic Sensor Module

Overview

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

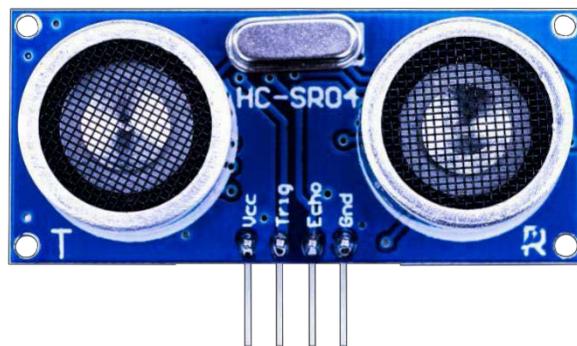
The HC-SR04 is inexpensive and easy to use since we will be using a Library specifically designed for these sensor.

Component Required:

1 x Elegoo Uno R3

1 x Ultrasonic sensor module

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



Component Introduction

Ultrasonic sensor

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

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

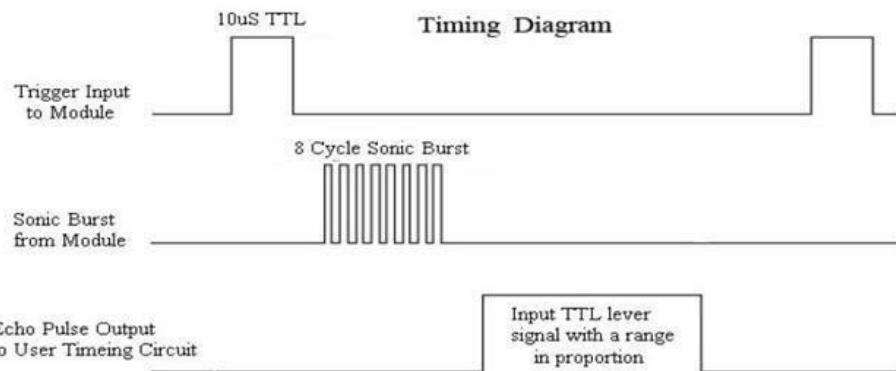
The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.

If the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to receiving.

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

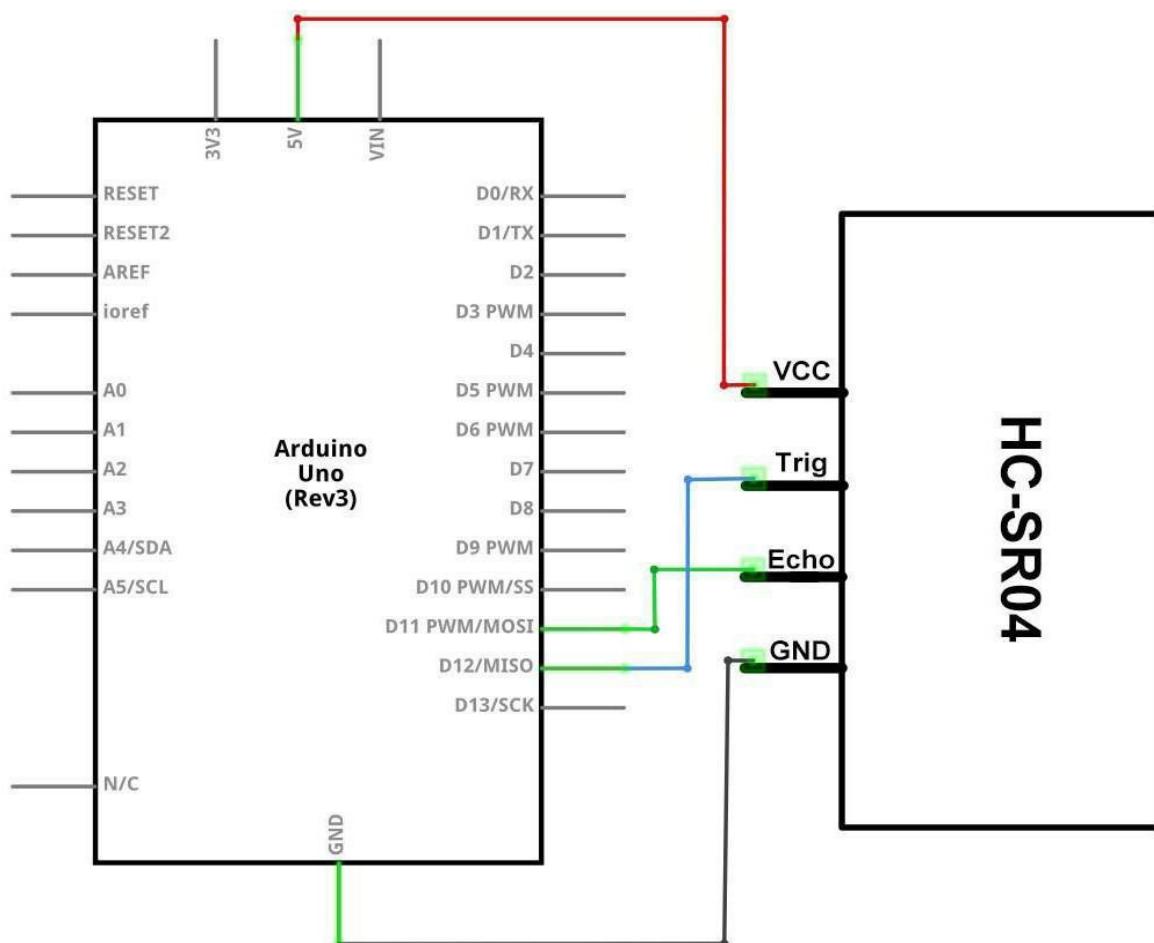
The Timing diagram is shown below. You only need to supply a short 10us pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion .You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: us

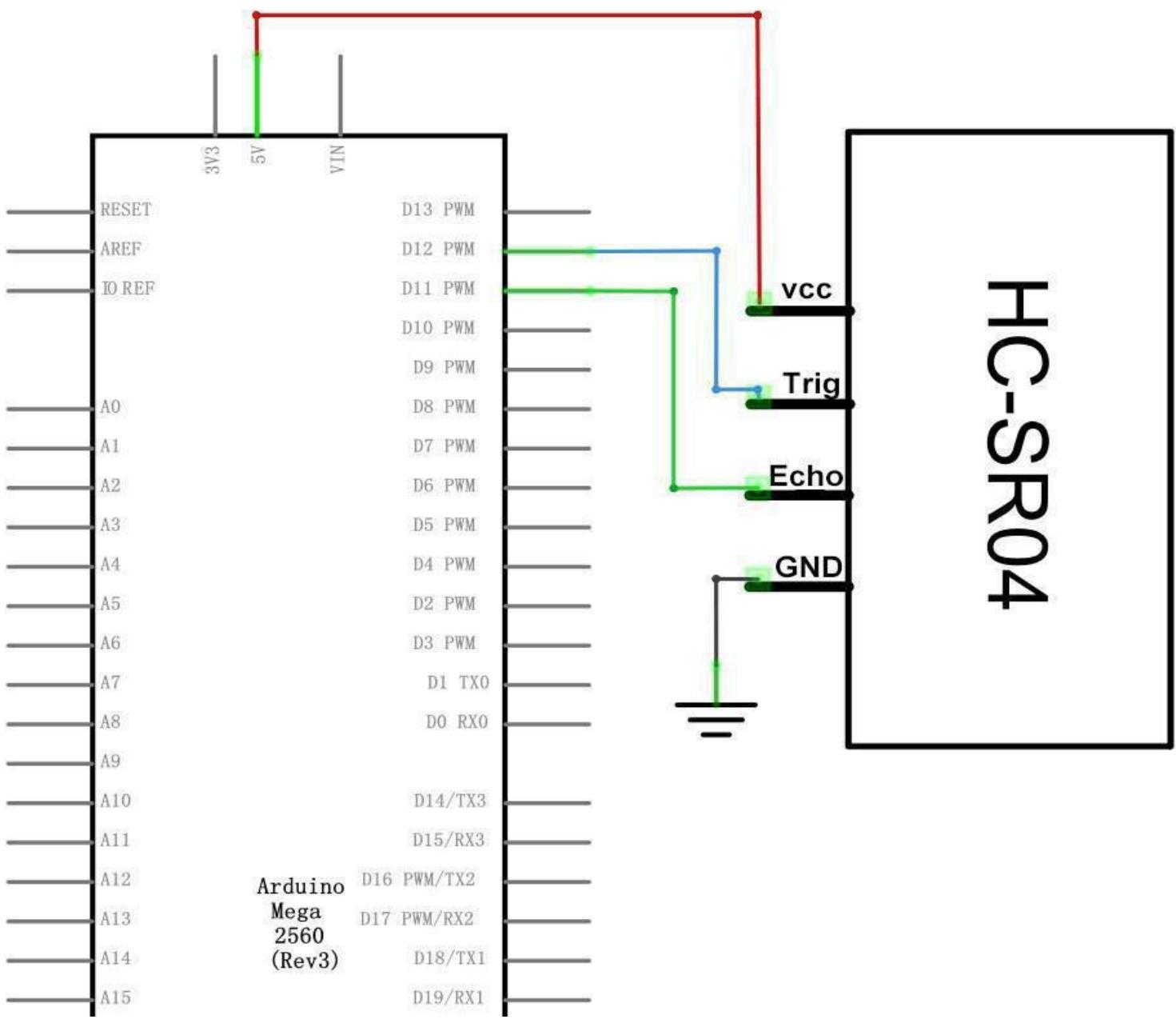
/ 58 = centimeters or us / 148 =inch; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.



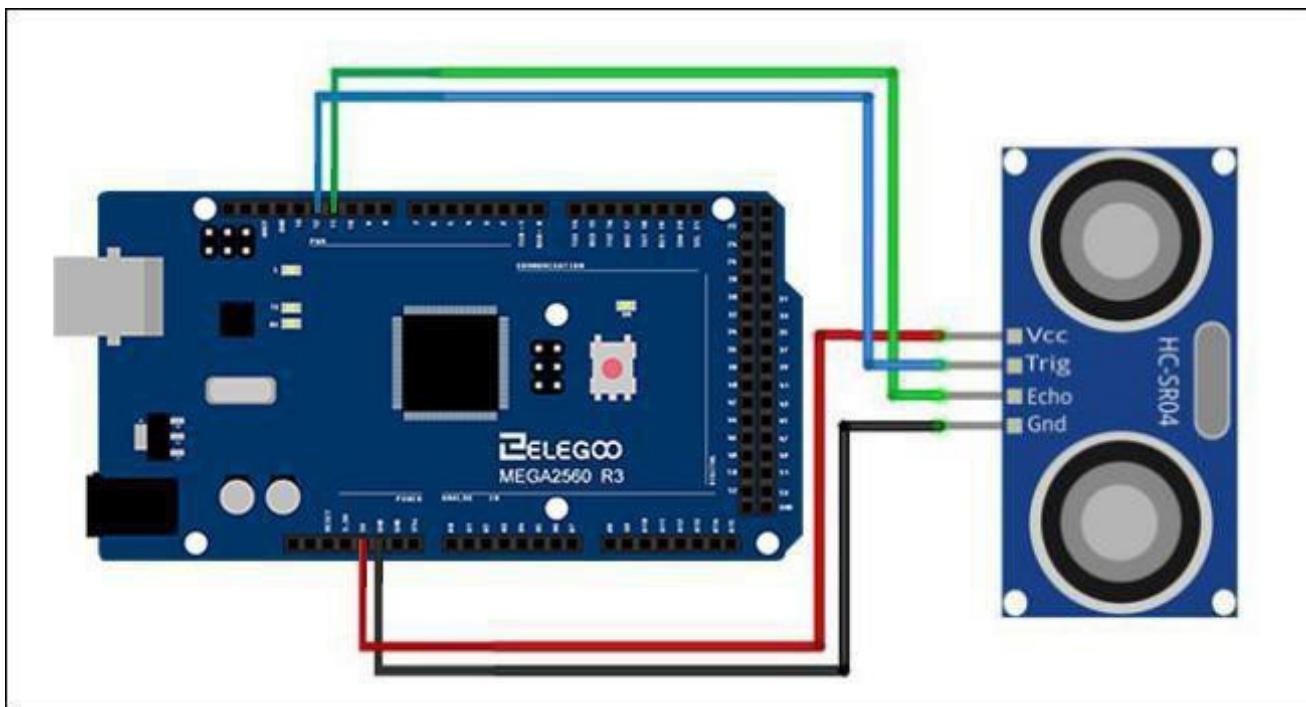
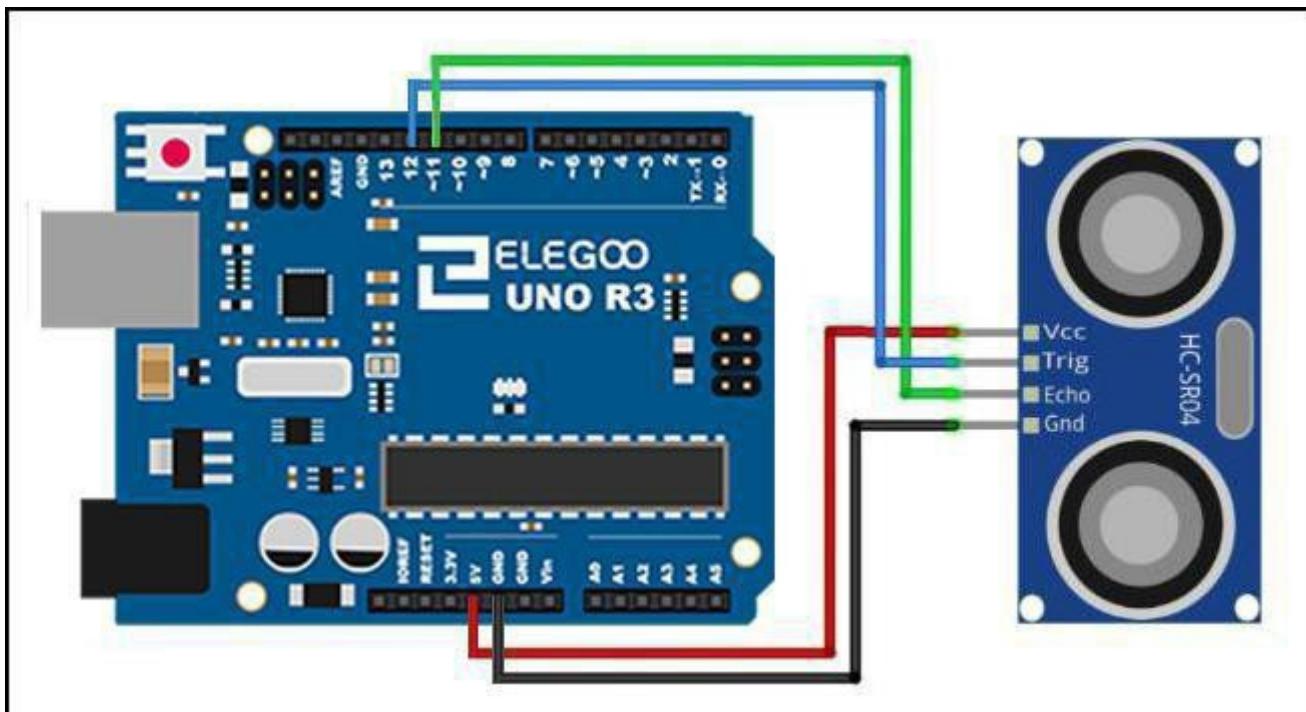
Connection

Schematic





Wiring diagram



Code

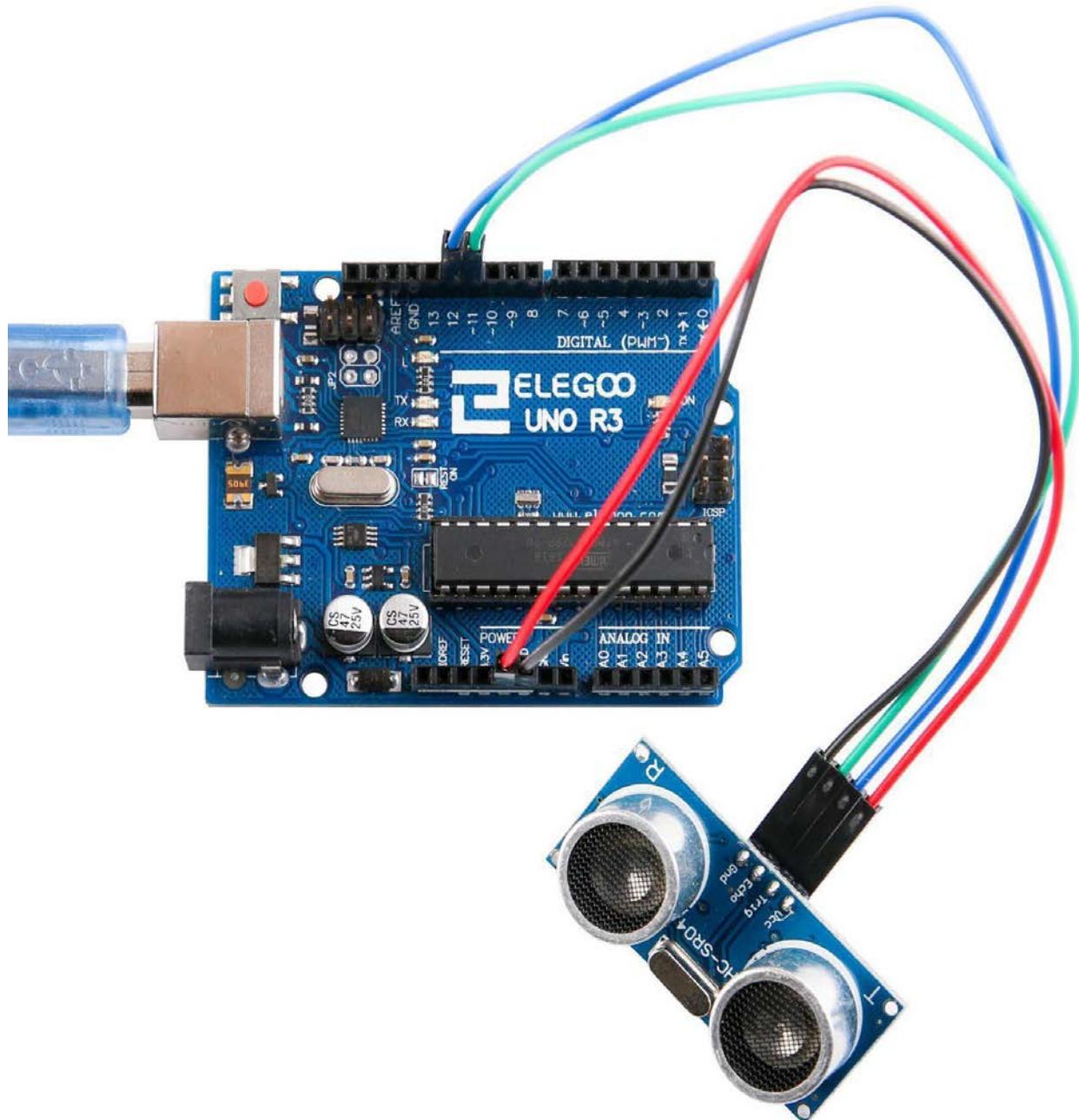
Using a Library designed for these sensors will make our code short and simple. We include the library at the beginning of our code, and then by using simple commands we can control the sensors.

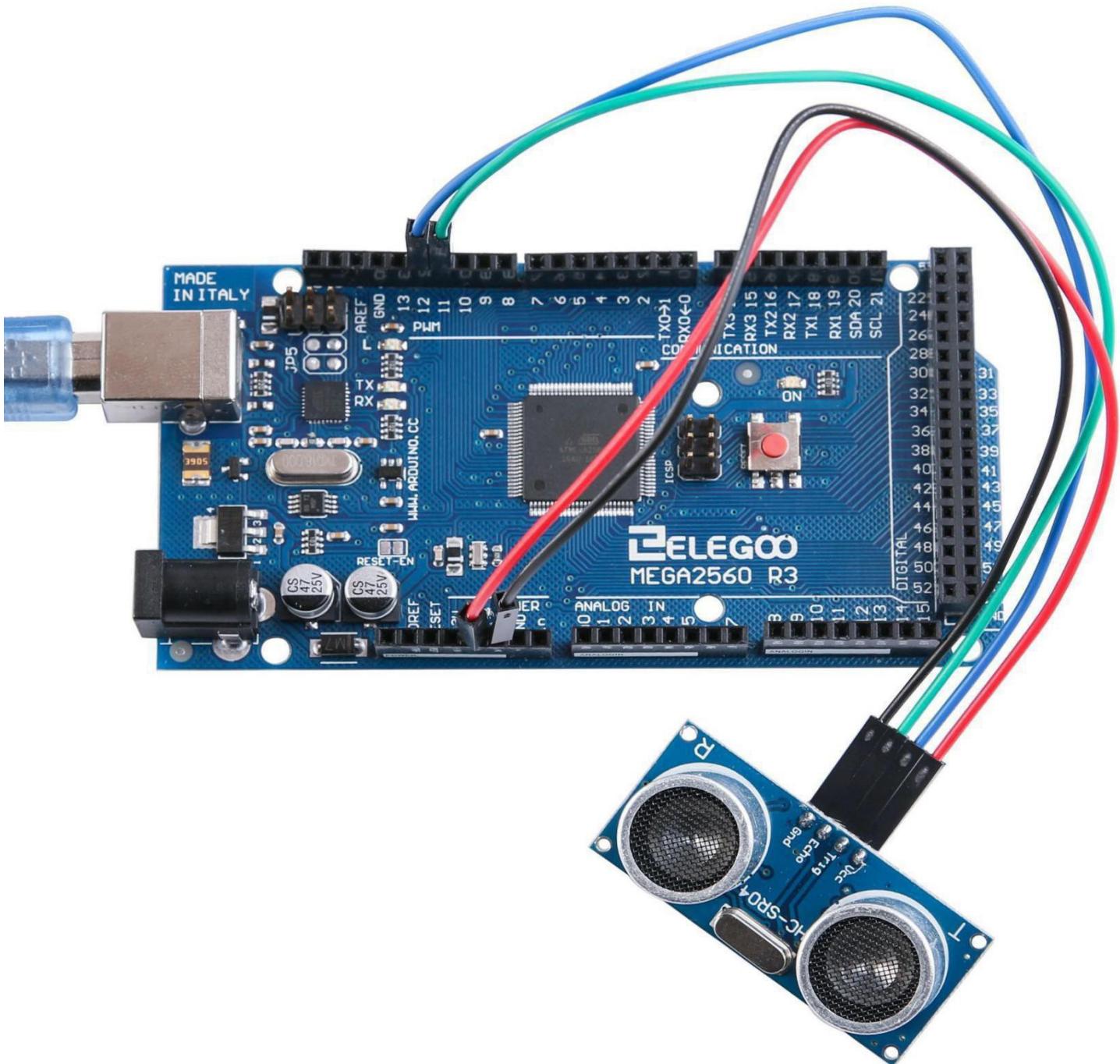
After you finished wiring, please open the program in the code folder- Lesson 29 Ultrasonic Sensor Module and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

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

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

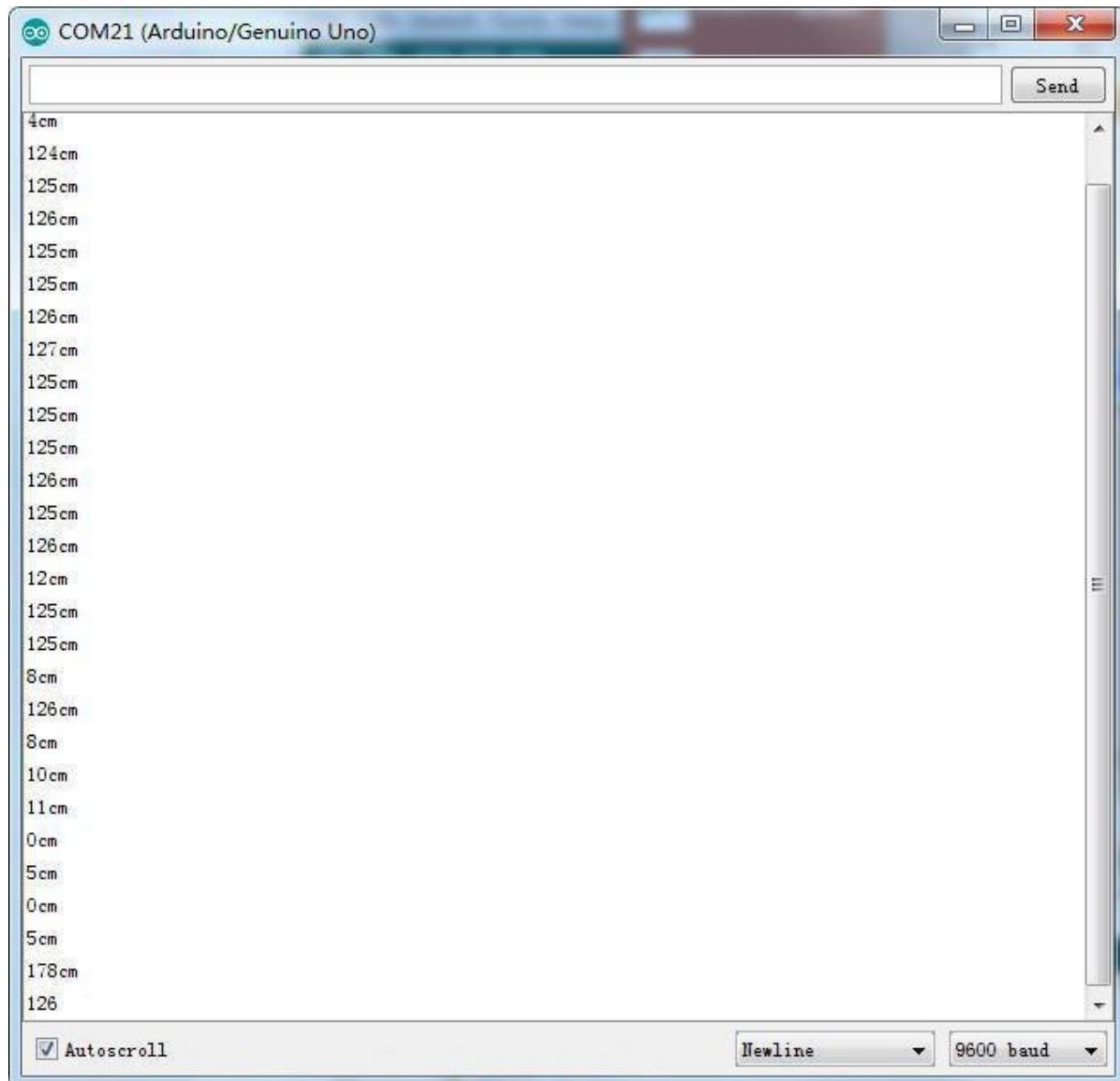
Example picture





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.



The followings are the code used in this experiment and their explanations:

```
#include <NewPing.h>

// Arduino pin tied to trigger pin on the ultrasonic sensor.

#define TRIGGER_PIN    12

// Arduino pin tied to echo pin on the ultrasonic sensor.

#define ECHO_PIN        11

/*Maximum distance we want to ping for (in centimeters).
Maximum sensor distance is rated at 400-500cm.*/

#define MAX_DISTANCE 200

// NewPing setup of pins and maximum distance.

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

void setup() {
    // Open serial monitor at 115200 baud to see ping results.

    Serial.begin(9600);
}

void loop() {
    // Wait 500ms between pings (about 2 pings/sec). 29ms should be the shortest delay between pings.

    delay(500);

    // Send ping, get ping time in microseconds (uS).

    unsigned int uS = sonar.ping();

    Serial.print("Ping: ");

    // Convert ping time to distance and print result (0 = outside set distance range, no ping echo)

    Serial.print(uS / US_ROUNDTRIP_CM);

    Serial.println("cm");

}
```

Lesson 30 GY-521 Module

Overview

In this lesson, we will learn how to use GY-521(MPU-6050) module which is one of the best IMU (Inertia Measurement Unit) sensors, compatible with Arduino.

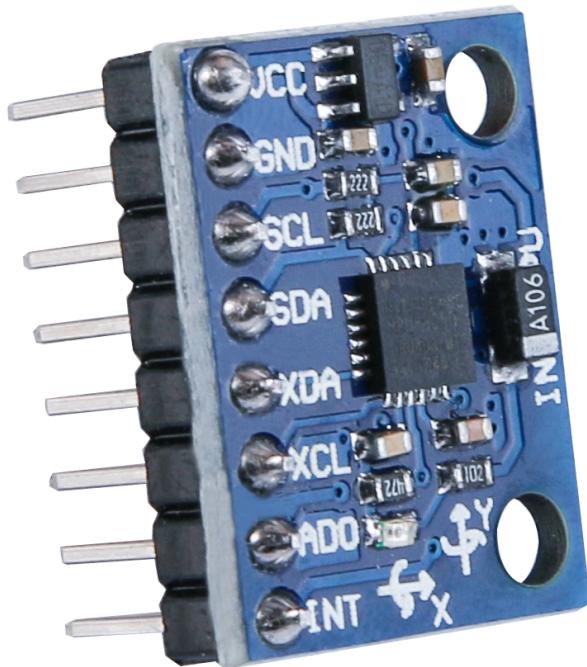
IMU sensors like the GY-521(MPU-6050) are used in self balancing robots, UAVs, smart phones, etc.

Component Required:

1 x Elegoo Uno R3

1 x GY-521 module

4 x F-M wires



Component Introduction

GY-521 SENSOR

The InvenSense GY-521 sensor contains a MEMS accelerometer and a MEMS gyro in a single chip. It is very accurate, as it contains 16-bits analog to digital conversion hardware for each channel. Therefore, it captures the x, y, and z channel at the same time. The sensor uses the I2C-bus to interface with the Arduino.

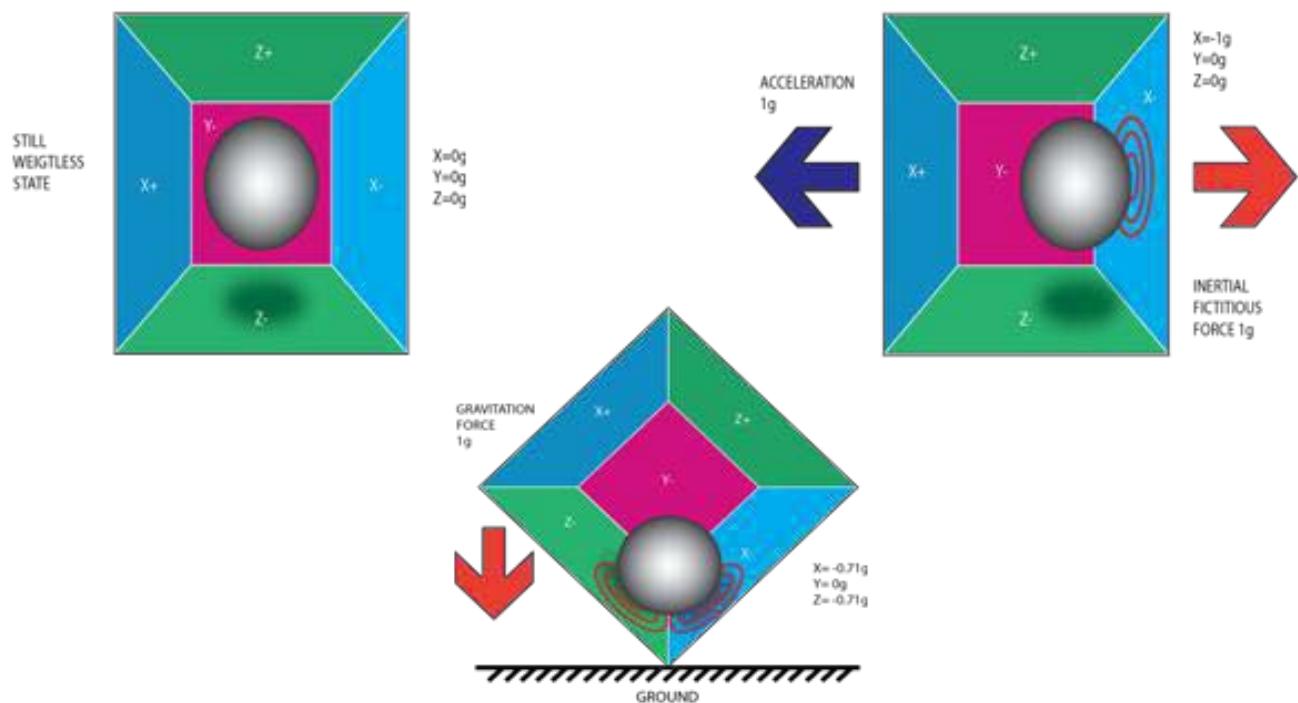
The GY-521 is not expensive, especially given the fact that it combines both an accelerometer and a gyro.

IMU sensors are one of the most inevitable type of sensors used today in all kinds of electronic gadgets. They are seen in smart phones, wearables, game controllers, etc. IMU sensors help us in getting the attitude of an object, attached to the sensor in three-dimensional space. These values usually in angles, thus help us to determine its attitude. Thus, they are used in smart phones to detect its orientation. And also, in wearable gadgets like the Nike fuel band or fit bit, which use IMU sensors to track movement.

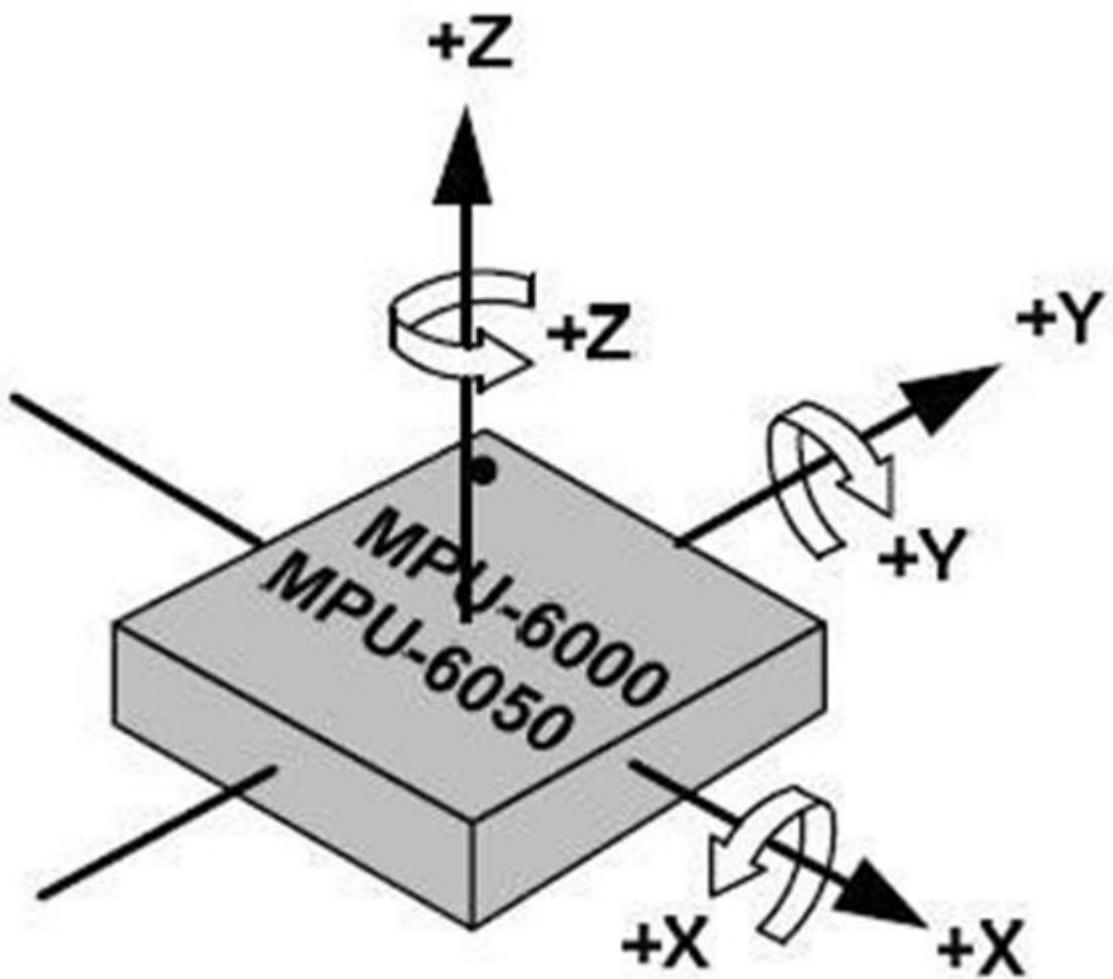
How does it work?

IMU sensors usually consists of two or more parts. Listing them by priority, they are: accelerometer, gyroscope, magnetometer and altimeter. The GY-521 is a 6 DOF (Degrees of Freedom) or a six axis IMU sensor, which means that it gives six values as output. Three values from the accelerometer and three from the gyroscope. The GY-521 is a sensor based on MEMS (Micro Electro Mechanical Systems) technology. Both the accelerometer and the gyroscope are embedded inside a single chip. This chip uses I2C (Inter Integrated Circuit) protocol for communication.

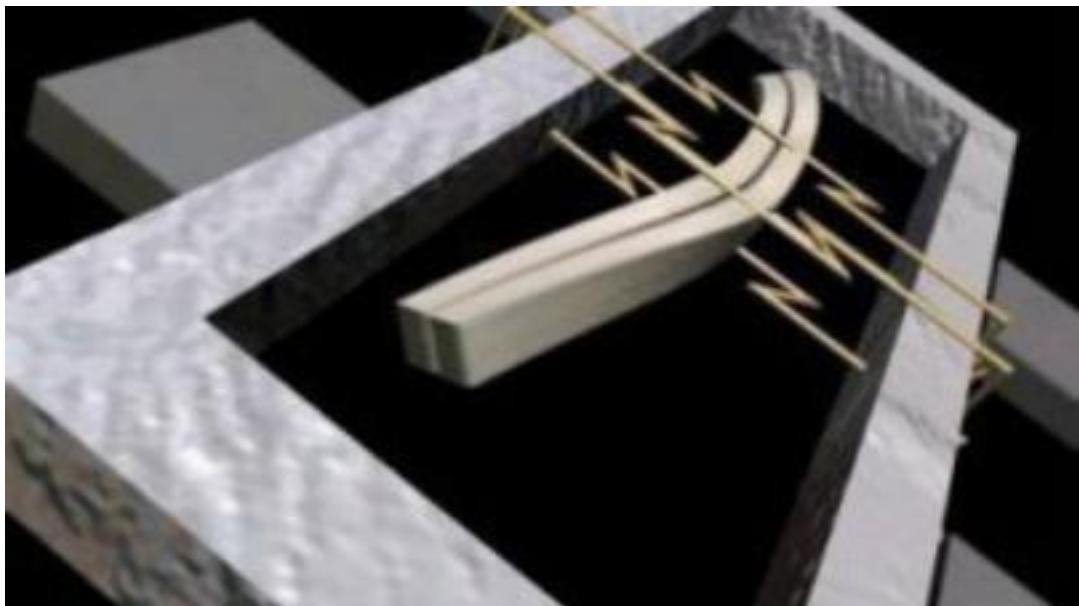
How does an acceleromet



An accelerometer works on the principle of piezo electric effect. Here, imagine a cuboidal box, having a small ball inside it, like in the picture above. The walls of this box are made with piezo electric crystals. Whenever you tilt the box, the ball is forced to move in the direction of the inclination, due to gravity. The wall with which the ball collides, creates tiny piezo electric currents. There are totally, three pairs of opposite walls in a cuboid. Each pair corresponds to an axis in 3D space: X, Y and Z axes. Depending on the current produced from the piezo electric walls, we can determine the direction of inclination and its magnitude for more information check this



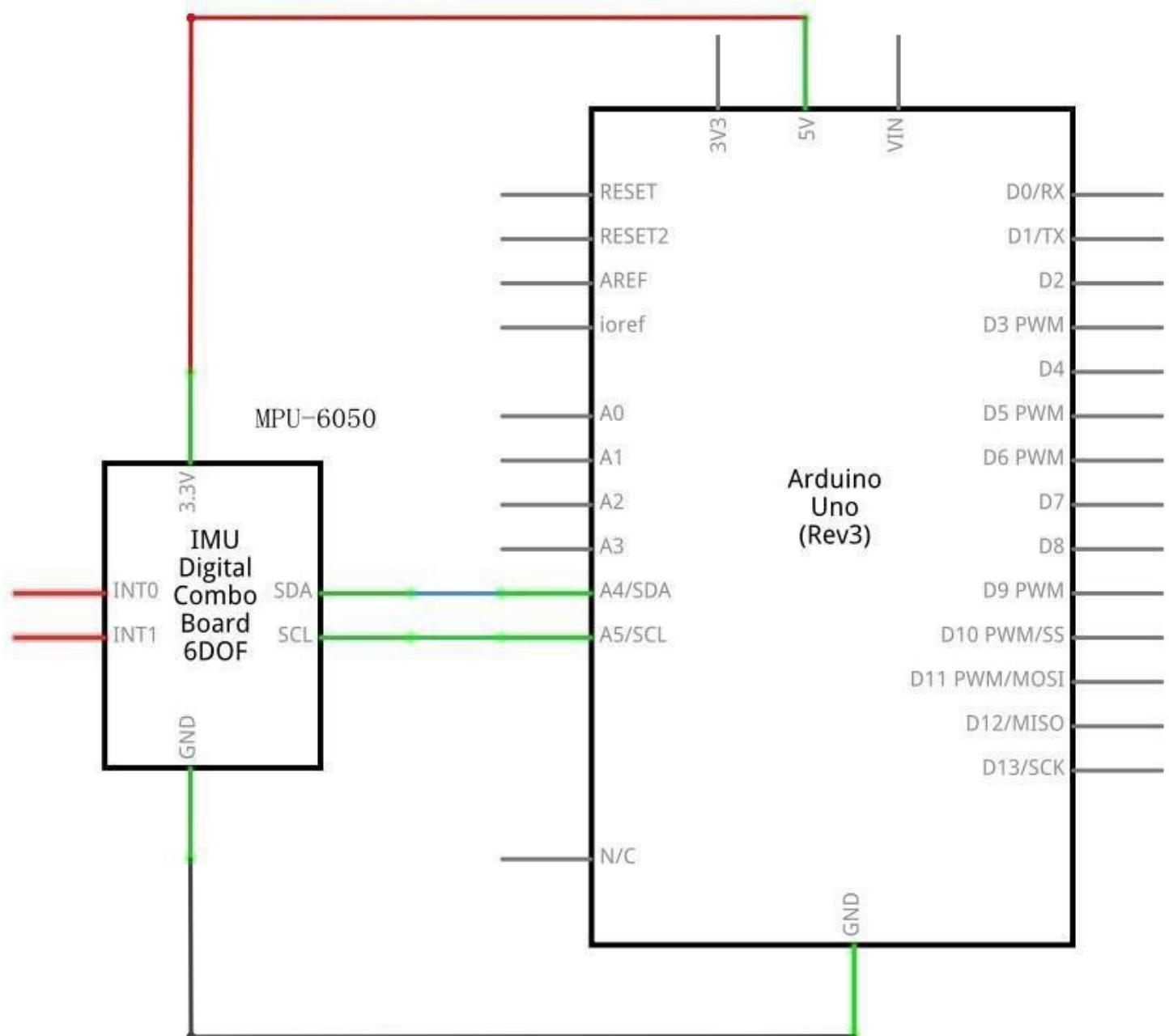
How does a gyroscope work?

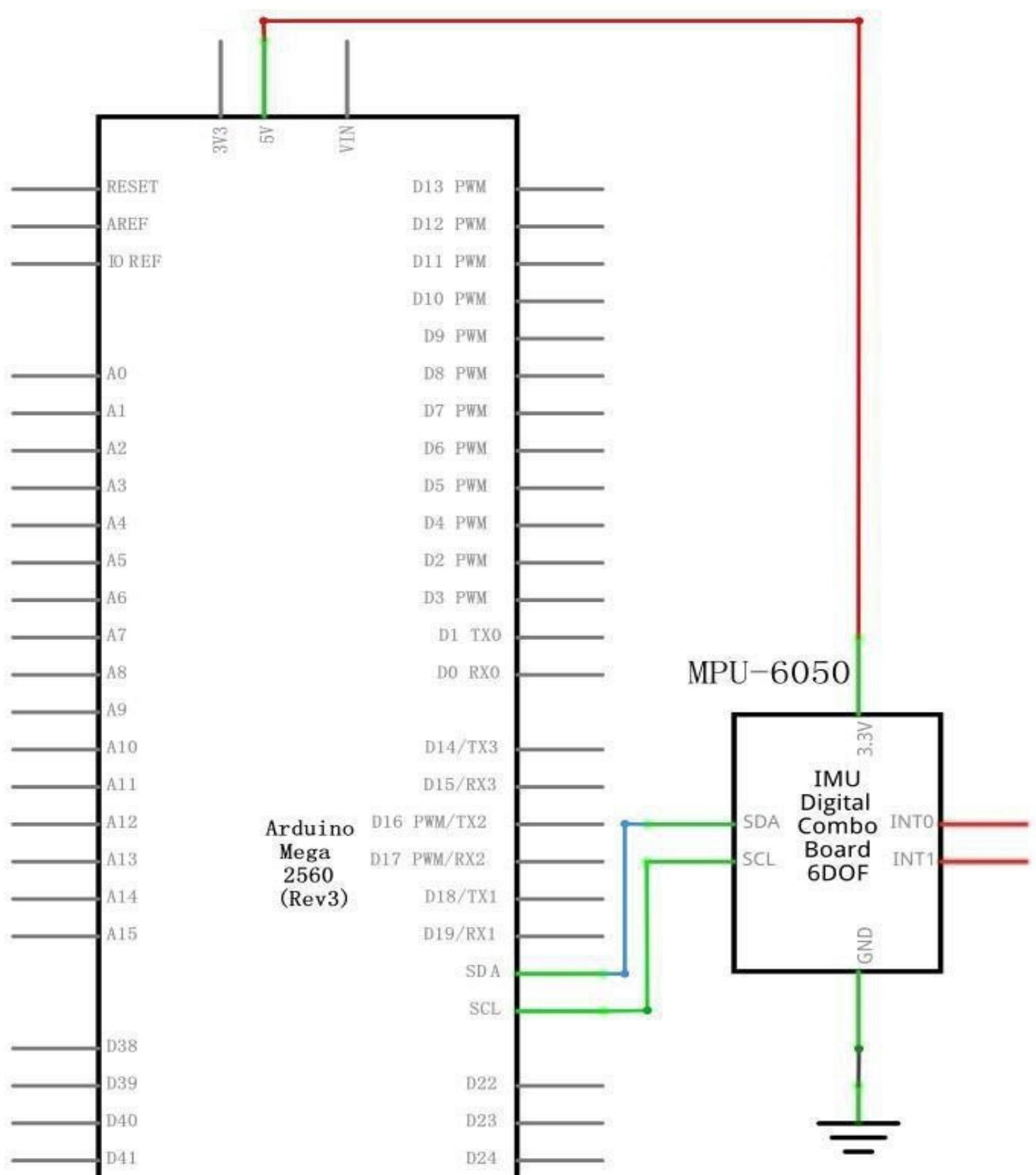


Gyrosopes work on the principle of Coriolis acceleration. Imagine that there is a fork like structure, which is in constant back and forth motion. It is held in place using piezo electric crystals. Whenever, you try to tilt this arrangement, the crystals experience a force in the direction of inclination. This is caused as a result of the inertia of the moving fork. The crystals thus produce a current in consensus with the piezo electric effect, and this current is amplified. The values are then refined by the host microcontroller.

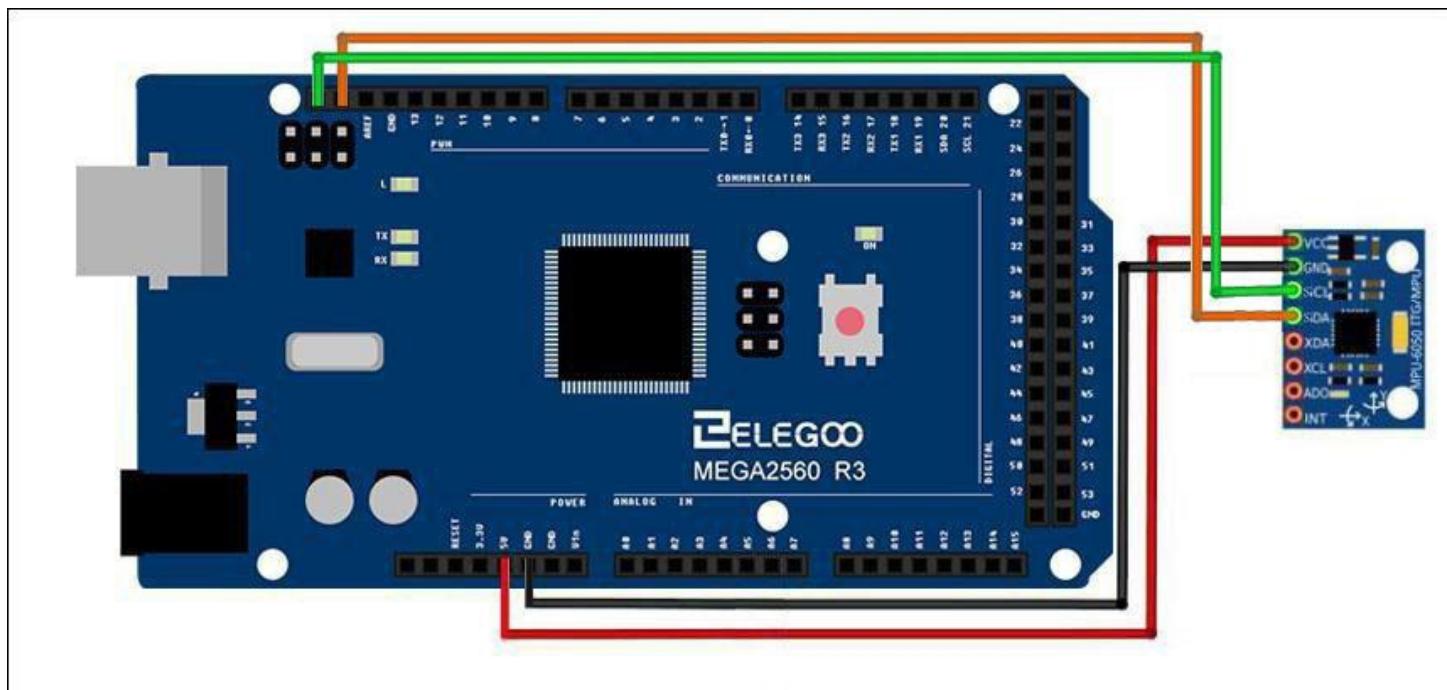
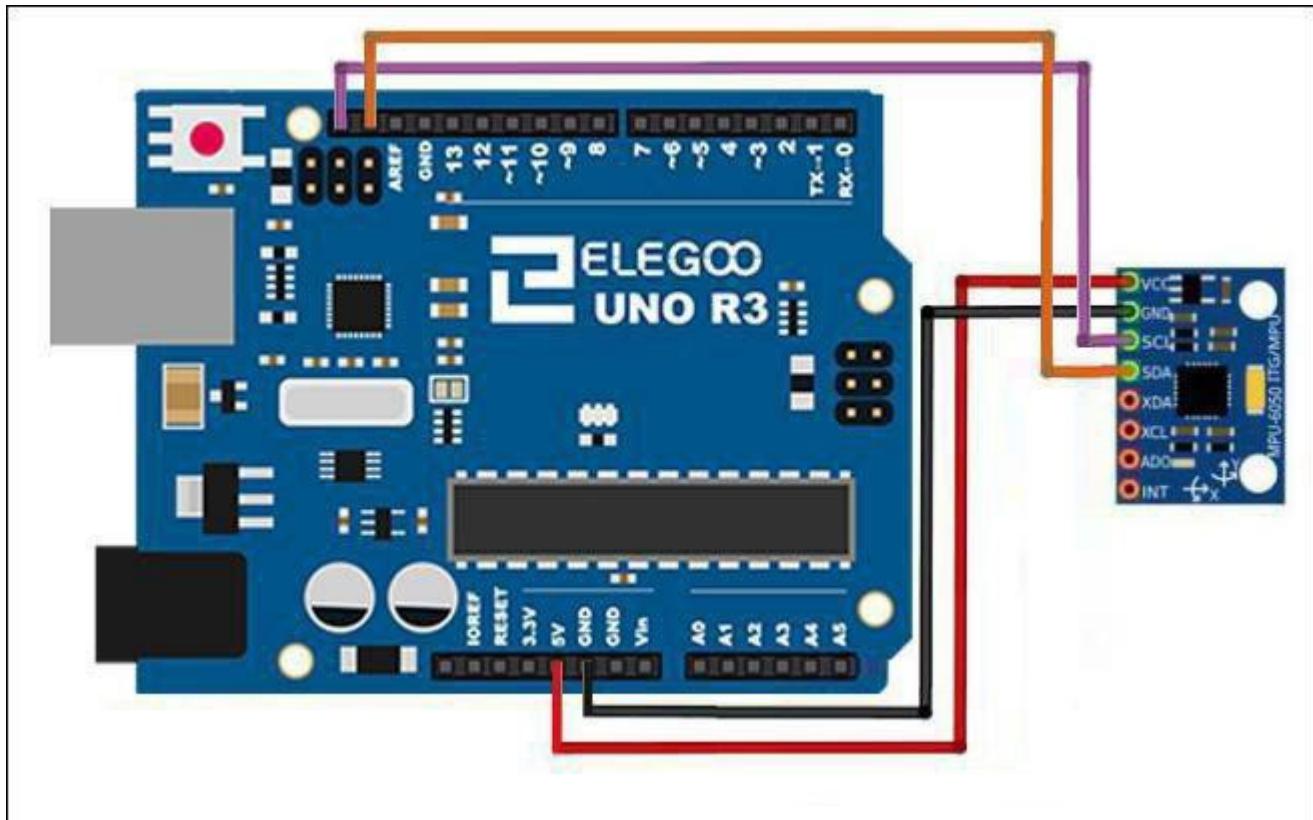
Connection

Schematic





Wiringdiagram



Next, we need to set up the I2C lines. For this connect the pin labelled as SDA on the GY-521 to the Arduino's analog pin 4 (SDA). And the pin labelled as SCL on the GY-521 to the Arduino's analog pin 5 (SCL). And that's it, you have finished wiring up the Arduino GY-521(GY-521).

Libraries needed

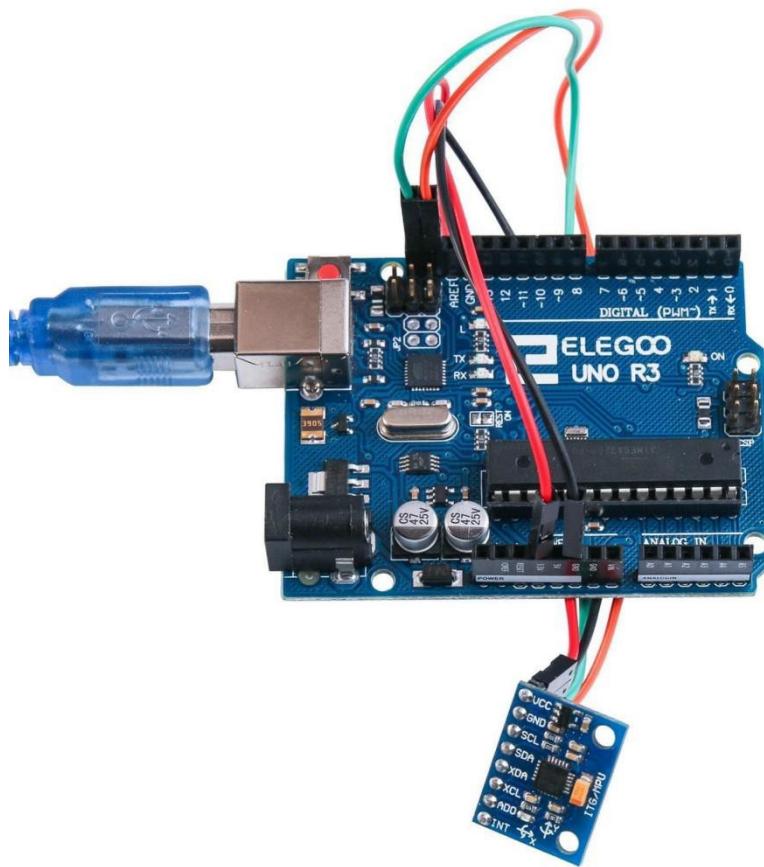
GY-521

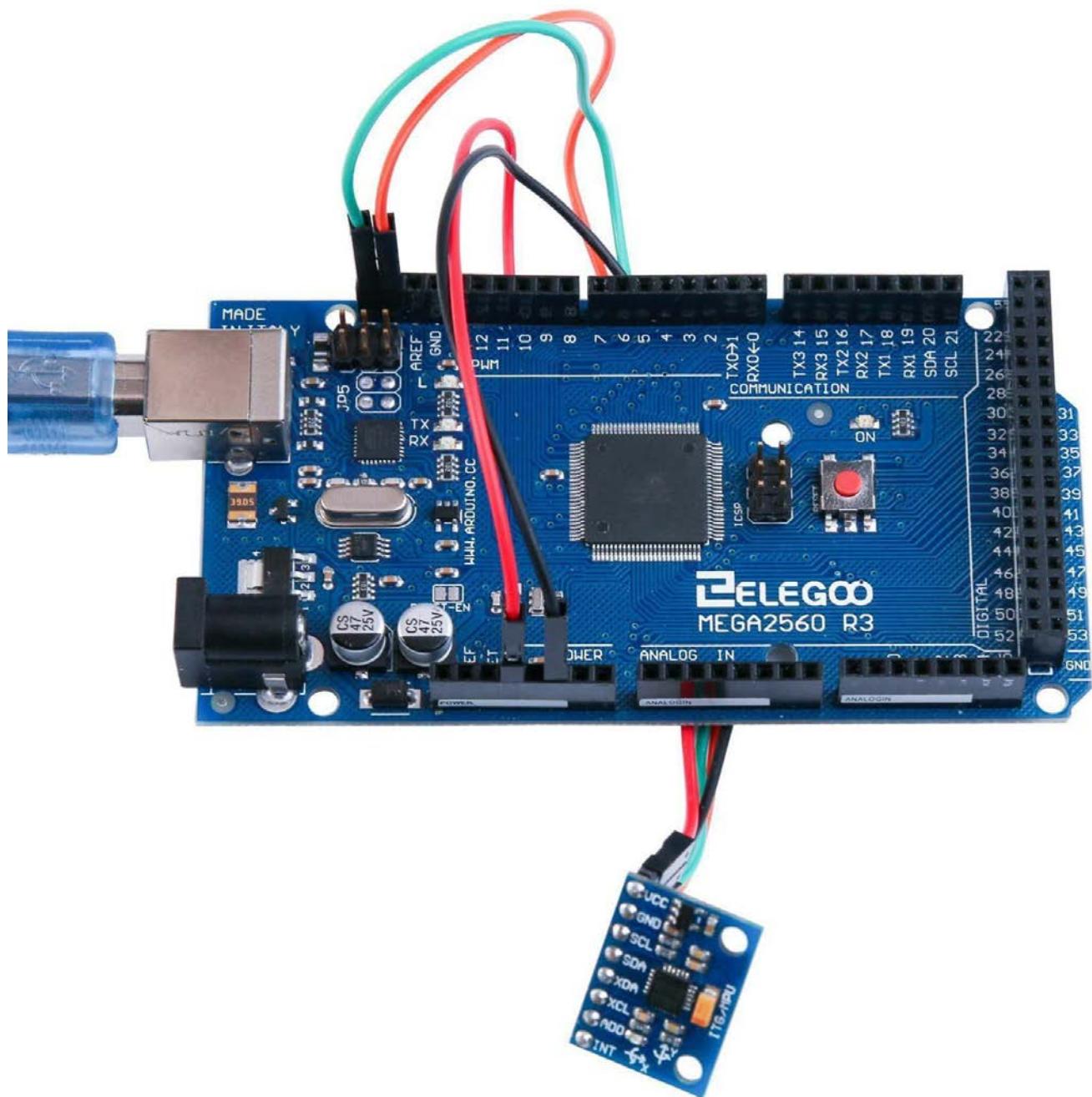
The Code

The short example sketch is a very short sketch and it shows all the raw values (Accelerometer, gyro and temperature). It should work on Arduino Uno, Nano, Leonardo, and also Due.

After wiring, please open the program in the code folder- Lesson 30 GY-521 Module and click UPLOAD to upload the program. See Lesson 3 for details about program uploading if there are any errors.

Example picture





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

Click the Serial Monitor button to turn on the serial monitor. The basics about the serial monitor are introduced in details in Lesson2.

```
AcX = 15976 | AcY = -4280 | AcZ = -596 | Imp = 24.62 | GyX = 230 | GyY = -26 | GyZ = -1231  
AcX = 15940 | AcY = -4408 | AcZ = -648 | Imp = 24.53 | GyX = -357 | GyY = -590 | GyZ = -519  
AcX = 15852 | AcY = -4328 | AcZ = -668 | Imp = 24.62 | GyX = -189 | GyY = -79 | GyZ = -198  
AcX = 15844 | AcY = -3972 | AcZ = -708 | Imp = 24.62 | GyX = -254 | GyY = -216 | GyZ = -44  
AcX = 15740 | AcY = -4232 | AcZ = -968 | Imp = 24.53 | GyX = -319 | GyY = -141 | GyZ = -227  
AcX = 15900 | AcY = -4936 | AcZ = -1008 | Imp = 24.62 | GyX = 126 | GyY = 111 | GyZ = 3870  
AcX = 15356 | AcY = -5080 | AcZ = -1192 | Imp = 24.58 | GyX = -1670 | GyY = -1741 | GyZ = -2571  
AcX = 14592 | AcY = -6504 | AcZ = -5700 | Imp = 24.53 | GyX = 662 | GyY = 264 | GyZ = 3219  
AcX = 13740 | AcY = -7020 | AcZ = -2744 | Imp = 24.58 | GyX = 8265 | GyY = 4962 | GyZ = 8163  
AcX = 3600 | AcY = -16556 | AcZ = 4244 | Imp = 24.58 | GyX = -17048 | GyY = -12197 | GyZ = 3845  
AcX = 12248 | AcY = -12292 | AcZ = 7256 | Imp = 24.62 | GyX = 12046 | GyY = 24428 | GyZ = -5483  
AcX = 588 | AcY = -3832 | AcZ = 19208 | Imp = 24.53 | GyX = 9258 | GyY = -4420 | GyZ = -4557  
AcX = 1896 | AcY = -3784 | AcZ = 6320 | Imp = 24.62 | GyX = -7486 | GyY = -32768 | GyZ = 2677  
AcX = 32767 | AcY = -19068 | AcZ = -1920 | Imp = 24.58 | GyX = -9262 | GyY = -19403 | GyZ = 25320  
AcX = -19160 | AcY = 12004 | AcZ = -2452 | Imp = 24.58 | GyX = -32768 | GyY = -32768 | GyZ = -4809  
AcX = -25124 | AcY = 1616 | AcZ = 32767 | Imp = 24.62 | GyX = 7628 | GyY = 7064 | GyZ = 6299  
AcX = 11976 | AcY = -8432 | AcZ = -32600 | Imp = 24.53 | GyX = 29381 | GyY = 32767 | GyZ = -19841  
AcX = 972 | AcY = -22992 | AcZ = -12480 | Imp = 24.62 | GyX = -31051 | GyY = -32768 | GyZ = 32767  
AcX = -27260 | AcY = 16868 | AcZ = 10704 | Imp = 24.62 | GyX = 32767 | GyY = 28603 | GyZ = -20636  
AcX = 32268 | AcY = -32468 | AcZ = -21952 | Imp = 24.58 | GyX = -27684 | GyY = -32768 | GyZ = 32767  
AcX = -22476 | AcY = -8436 | AcZ = -3976 | Imp = 24.58 | GyX = 32156 | GyY = 32767 | GyZ = 25696  
AcX = -3836 | AcY = -13428 | AcZ = -9628 | Imp = 24.58 | GyX = -30925 | GyY = -32768 | GyZ = 32767  
AcX = 3164 | AcY = -5392 | AcZ = -19464 | Imp = 24.48 | GyX = -30769 | GyY = -17986 | GyZ = 17236  
AcX = 3408 | AcY = -3584 | AcZ = -13752 | Imp = 24.58 | GyX = 1820 | GyY = -2660 | GyZ = -186  
AcX = 4404 | AcY = -5552 | AcZ = -15216 | Imp = 24.58 | GyX = -578 | GyY = -234 | GyZ = -425  
AcX = 4160 | AcY = -5456 | AcZ = -15304 | Imp = 24.53 | GyX = -445 | GyY = -154 | GyZ = -277  
AcX = 4152 | AcY = -5192 | AcZ = -15300 | Imp = 24.53 | GyX = -404 | GyY = -114 | GyZ = -262
```

The followings are the code used in this experiment and their explanations:

```
// GY-521 Short Example Sketch  
// By Arduino User John Chi  
// August 17, 2014  
// Public Domain  
  
#include<Wire.h>  
  
// I2C address of the GY-521  
const int MPU_addr=0x68;  
  
int16_t AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;  
  
void setup(){  
    Wire.begin();  
    Wire.beginTransmission(MPU_addr);  
    // PWR_MGMT_1 register  
    Wire.write(0x6B);  
    // set to zero (wakes up the GY-521)  
    Wire.write(0);  
    Wire.endTransmission(true);  
    Serial.begin(9600);  
}  
  
void loop(){  
    Wire.beginTransmission(MPU_addr);  
    // starting with register 0x3B (ACCEL_XOUT_H)  
    Wire.write(0x3B);  
    Wire.endTransmission(false);  
    // request a total of 14 registers  
    Wire.requestFrom(MPU_addr,14,true);  
    // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)  
    AcX=Wire.read()<<8|Wire.read();  
    // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)  
    AcY=Wire.read()<<8|Wire.read();  
  
    // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
```

```
AcZ=Wire.read()<<8|Wire.read();
// 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)

Tmp=Wire.read()<<8|Wire.read();
// 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)

GyX=Wire.read()<<8|Wire.read();
// 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)

GyY=Wire.read()<<8|Wire.read();
// 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)

GyZ=Wire.read()<<8|Wire.read();

Serial.print("AcX = "); Serial.print(AcX);
Serial.print(" | AcY = "); Serial.print(AcY);
Serial.print(" | AcZ = "); Serial.print(AcZ);

//equation for temperature in degrees C from datasheet

Serial.print(" | Tmp = "); Serial.print(Tmp/340.00+36.53);
Serial.print(" | GyX = "); Serial.print(GyX);
Serial.print(" | GyY = "); Serial.print(GyY);
Serial.print(" | GyZ = "); Serial.println(GyZ);
delay(333);

}
```

Lesson 31 HC-SR501 PIR Sensor

Overview

In this lesson you will learn how to use a PIR movement detector with an UNO.

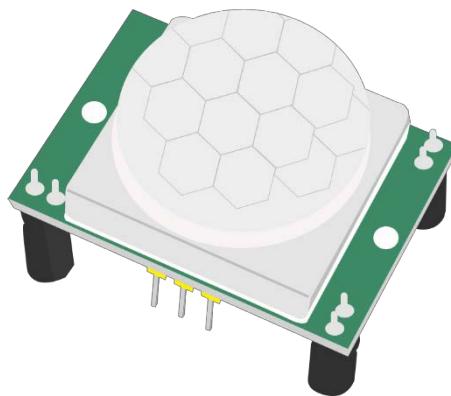
The UNO is the heart of this project. It 'listens' to the PIR sensor and when motion is detected, instructs the LED to light on or shutoff.

Component Required:

1 x Elegoo Uno R3

1 x HC-SR501 PIR motionsensor

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

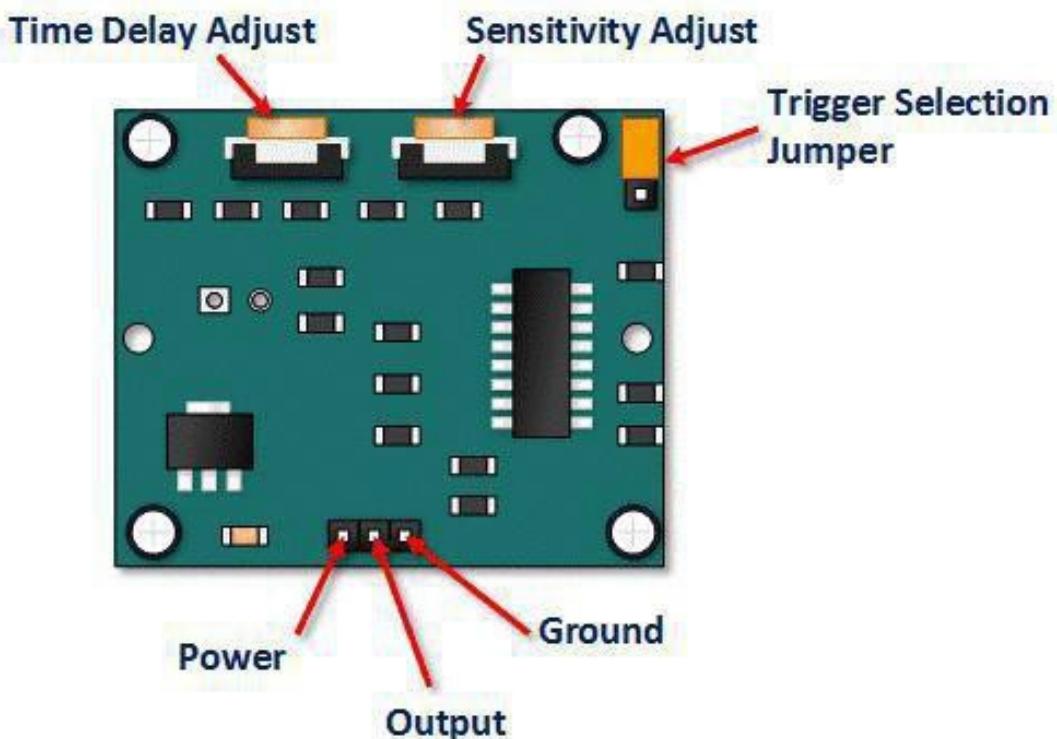


Component Introduction

PIR SENSOR:

PIR sensors are more complicated than many of the other sensors explained in this tutorial (like photocells, FSRs and tilt switches) because there are multiple variables that affect the sensors input and output..

The PIR sensor itself has two slots. Each slot is made of a special material that is sensitive to IR. The lens used here is not really doing much and so we see that the two slots can 'see' out past some distance (basically the sensitivity of the sensor). When the sensor is idle, both slots detect the same amount of IR, the ambient amount radiated from the room or walls or outdoors. When a warm body like a human or an animal passes by, it first intercepts one half of the PIR sensor, which causes a positive differential change between the two halves. When the warm body leaves the sensing area, the reverse happens, whereby the sensor generates a negative differential change. These change pulses are what is detected.



Pin or Control	Function
Time Delay Adjust	Sets how long the output remains high after detecting motion.... Anywhere from 5 seconds to 5 minutes.
Sensitivity Adjust	Sets the detection range.... from 3 meters to 7 meters
Trigger Selection Jumper	Set for single or repeatable triggers.
Ground pin	Ground input
Output Pin	Low when no motion is detected.. High when motion is detected. High is 3.3V

HC SR501 PIR Functional Description

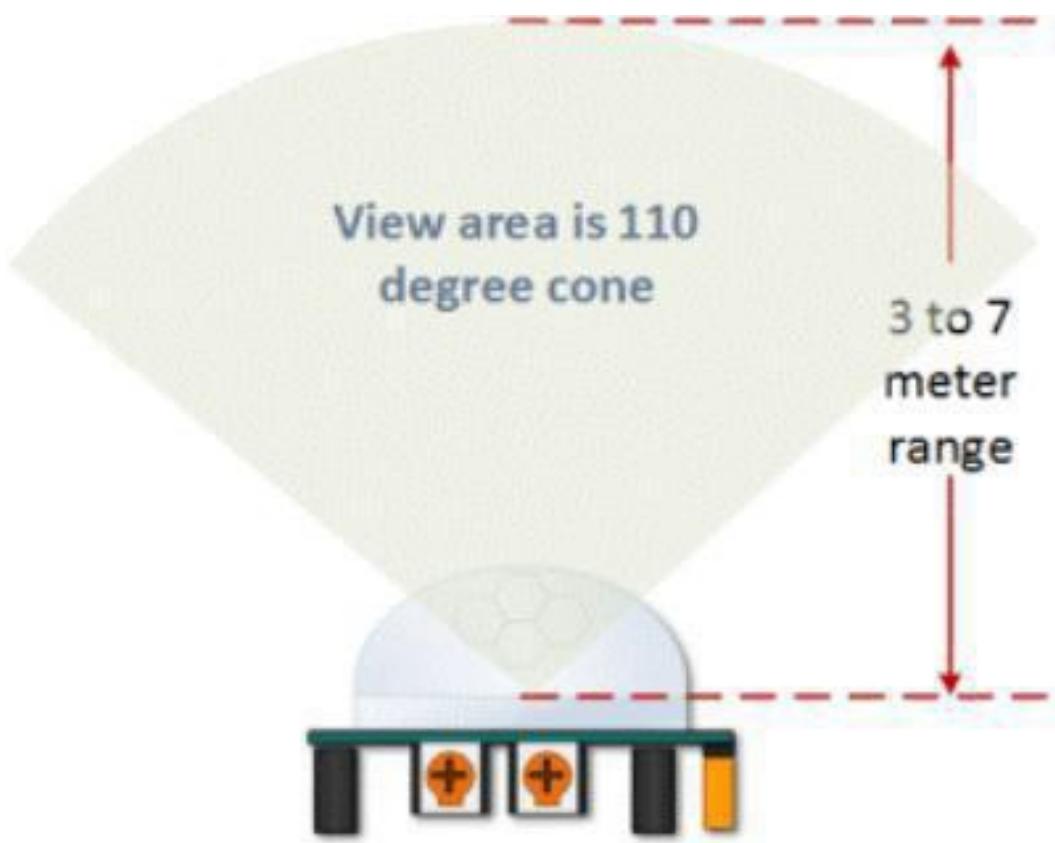
The SR501 will detect infrared changes and if interpreted as motion, will set its output low. What is or is not interpreted as motion is largely dependent on user settings and adjustments.

Device Initialization

The device requires nearly a minute to initialize. During this period, it can and often will output false detection signals. Circuit or controller logic needs to take this initialization period into consideration.

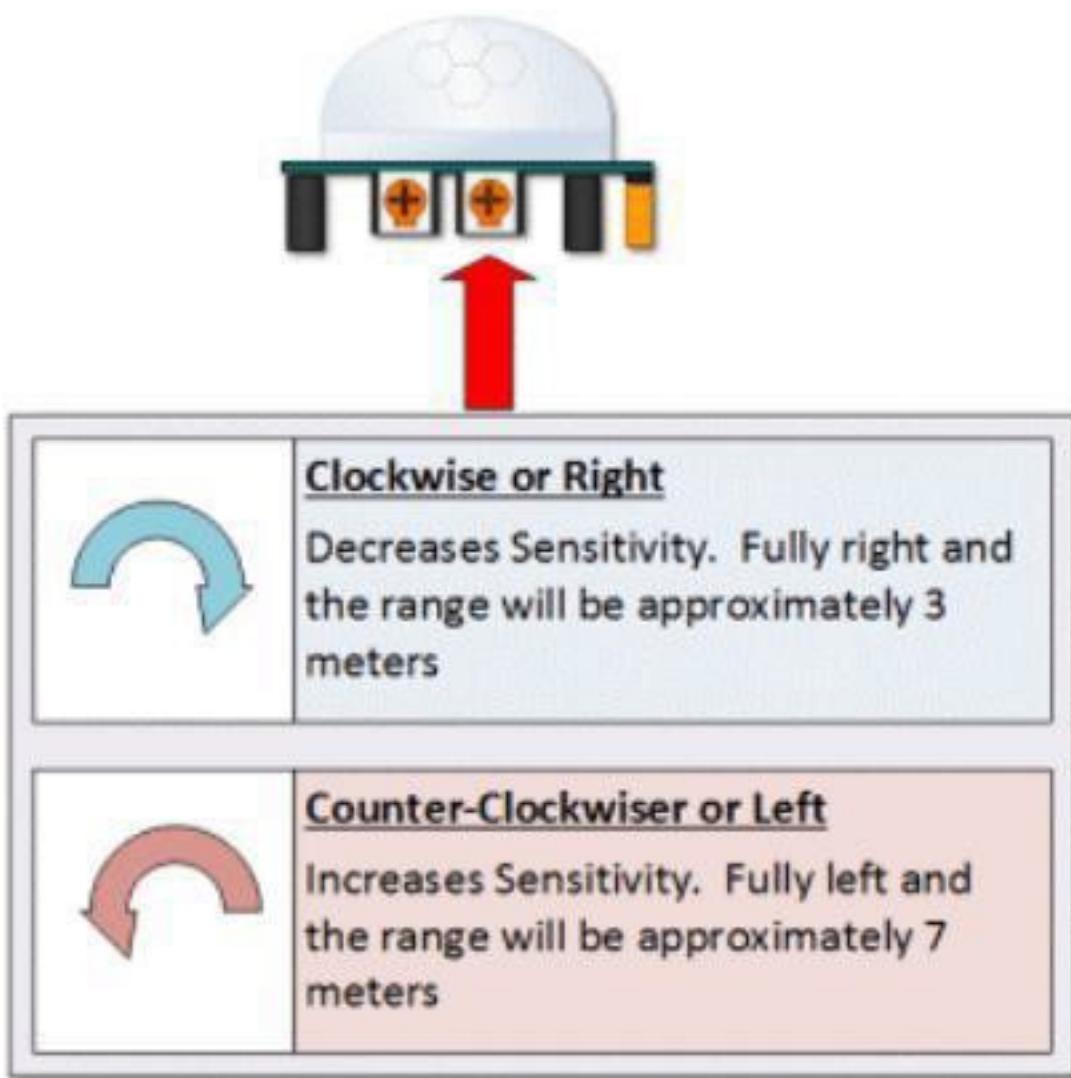
Device Area of Detection

The device will detect motion inside a 110 degree cone with a range of 3 to 7 meters.



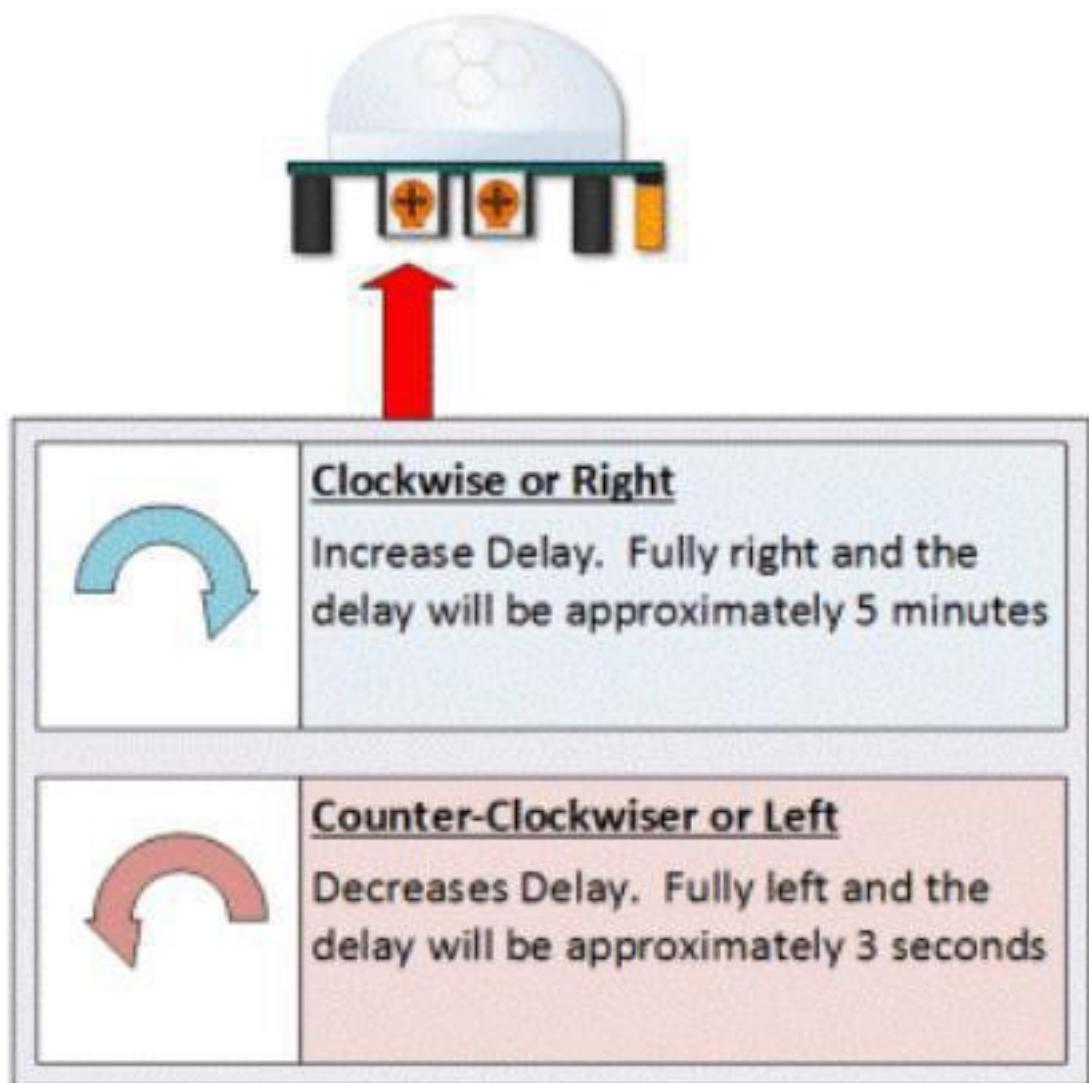
HC SR501 View Area

PIR Range (Sensitivity) Adjustment as mentioned, the adjustable range is from approximately 3 to 7 meters. The illustration below shows this adjustment.



HC SR501 Sensitivity Adjust Time Delay Adjustment

The time delay adjustment determines how long the output of the PIR sensor module will remain high after detection motion. The range is from about 3 seconds to five minutes.



HC SR501 Time Delay Adjustment

3 Seconds Off After Time Delay Completes –IMPORTANT

The output of this device will go LOW (or Off) for approximately 3 seconds AFTER the time delay completes. In other words, all motion detection is blocked during this three second period.

For Example:

Imagine you're in the single trigger mode and your time delay is set 5 seconds. The PIR will detect motion and set it high for 5 seconds.

After five seconds, the PIR will set its output low for about 3 seconds.

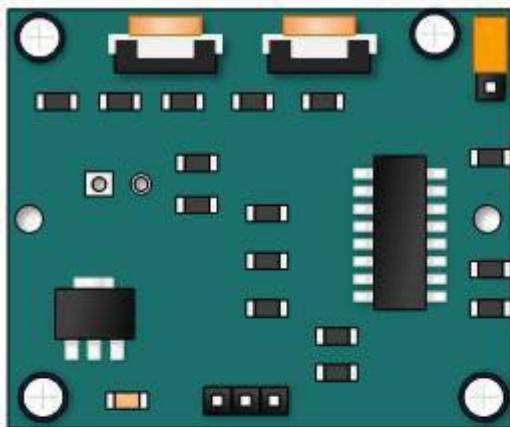
During the three seconds, the PIR will not detect motion. After three seconds, the PIR will detect motion again and detected motion will once again set the output high.

Trigger Mode Selection Jumper

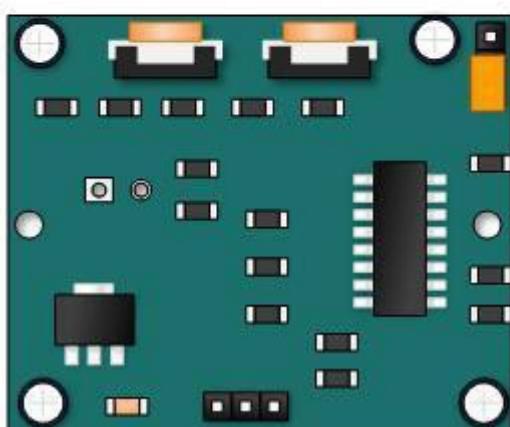
The trigger mode selection jumper allows you to select between single and repeatable triggers. The affect of this jumper setting is to determine when the time delay begins.

SINGLE TRIGGER – The time delay begins immediately when motion is first detected.

REPEATABLE TRIGGER – Each detected motion resets the time delay. Thus the time delay begins with the last motion detected.



Single Trigger Mode – Time
Delay is started immediately
upon detecting motion.
Continued detection is blocked



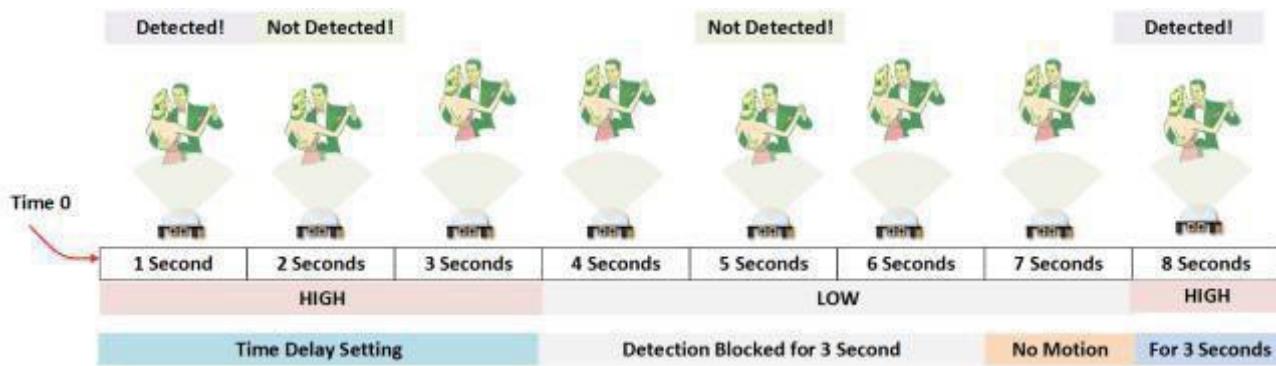
Repeatable Trigger Mode –
Time Delay is re-started every
time motion is detected.

HC-SR501 Dance Floor Application Examples

Imagine that you want to control lighting on a dance floor based upon where the dancers are dancing. Understanding how the time delay and trigger mode interact will be necessary to controlling that lighting in the manner that you want.

Example One

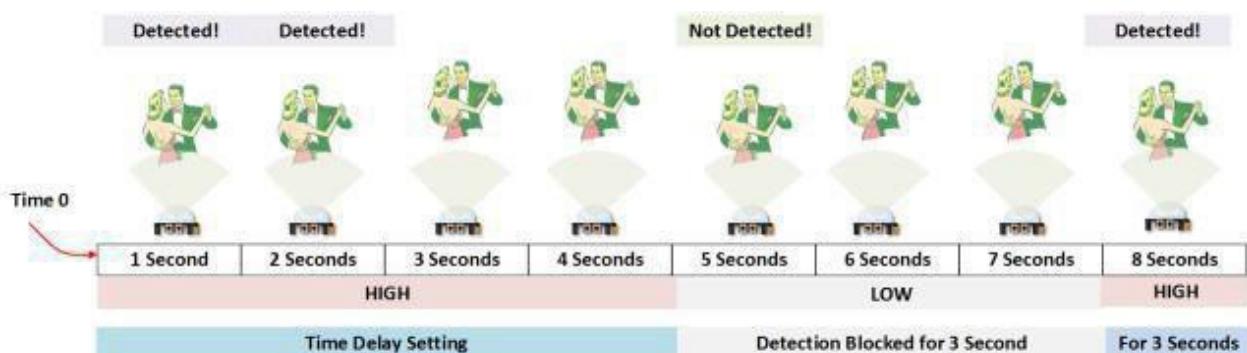
In this first example, the time delay is set to three seconds and the trigger mode is set to single. As you can see in the illustration below, the motion is not always detected. In fact, there is a period of about six seconds where motion can not be detected. Feel free to click on the image to enlarge.



Example Two

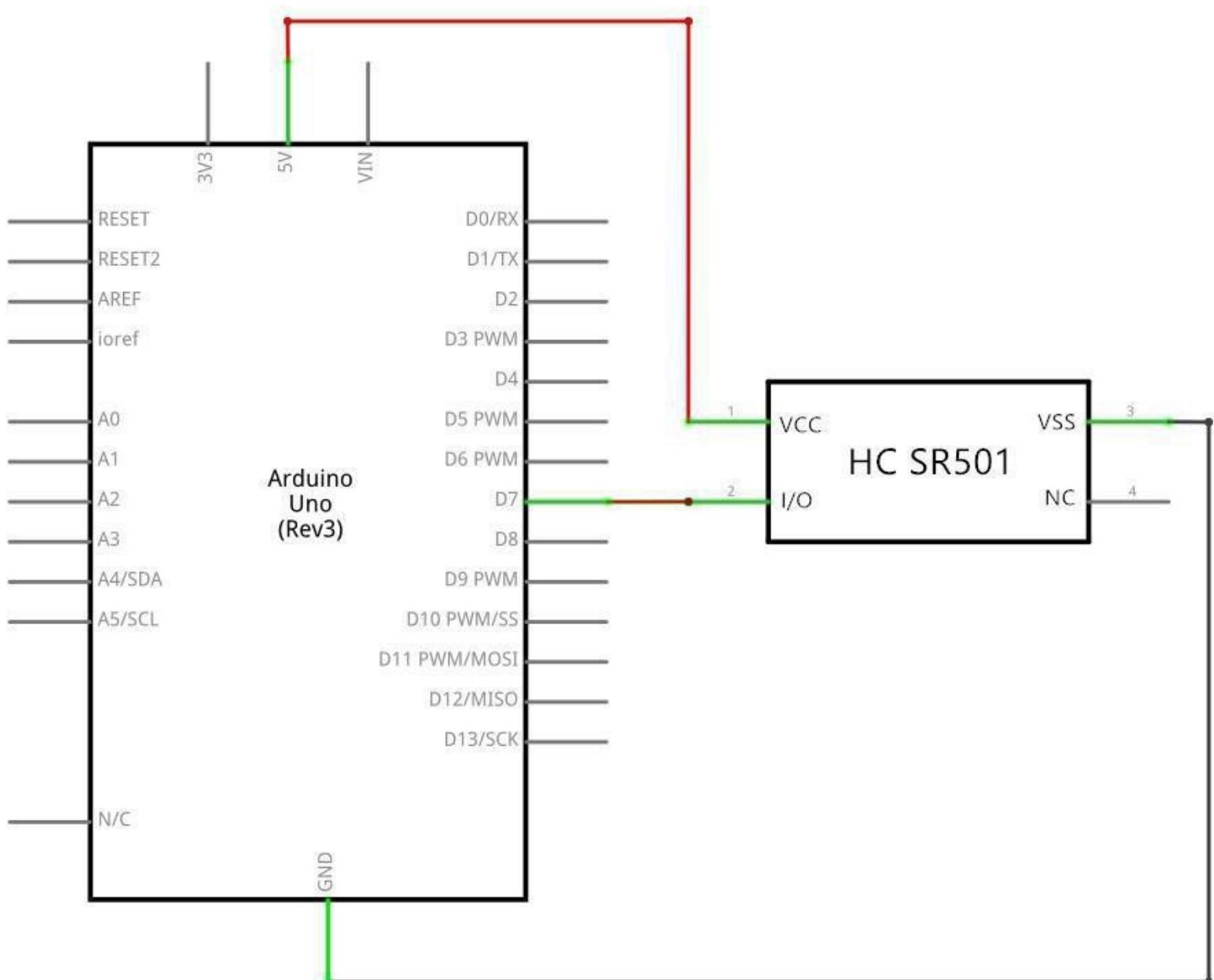
In the next example, the time delay is still at three seconds and the trigger is set to repeatable. In the illustration below, you can see that the time delay period is restarted. However, after that three seconds, detection will still be blocked for three seconds.

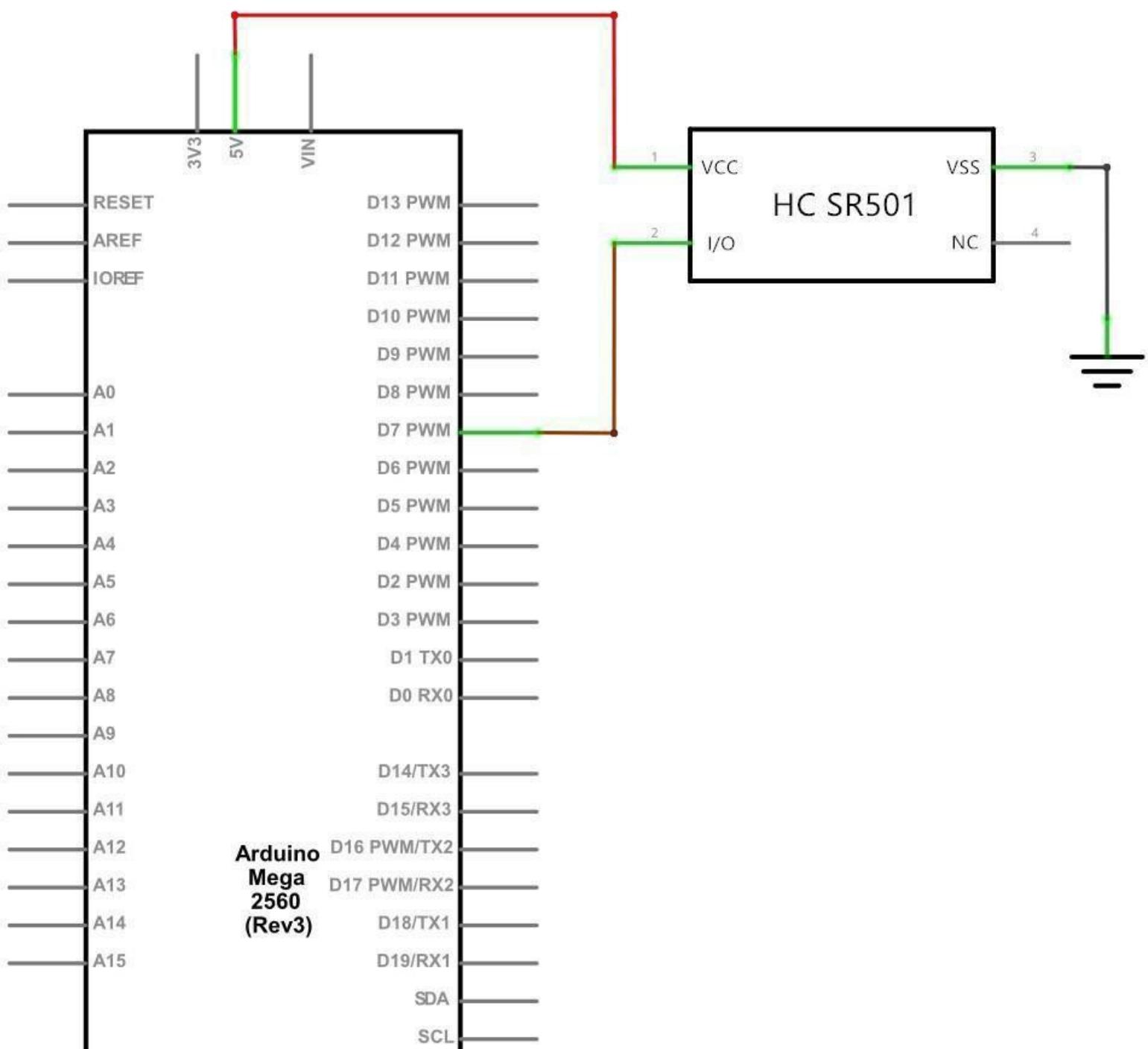
As I mentioned previously, you could override the 3 second blocking period with some creative code, but do give that consideration. Some of the electronics you use may not like an on and then off jolt. The three seconds allows for a little rest before starting backup.



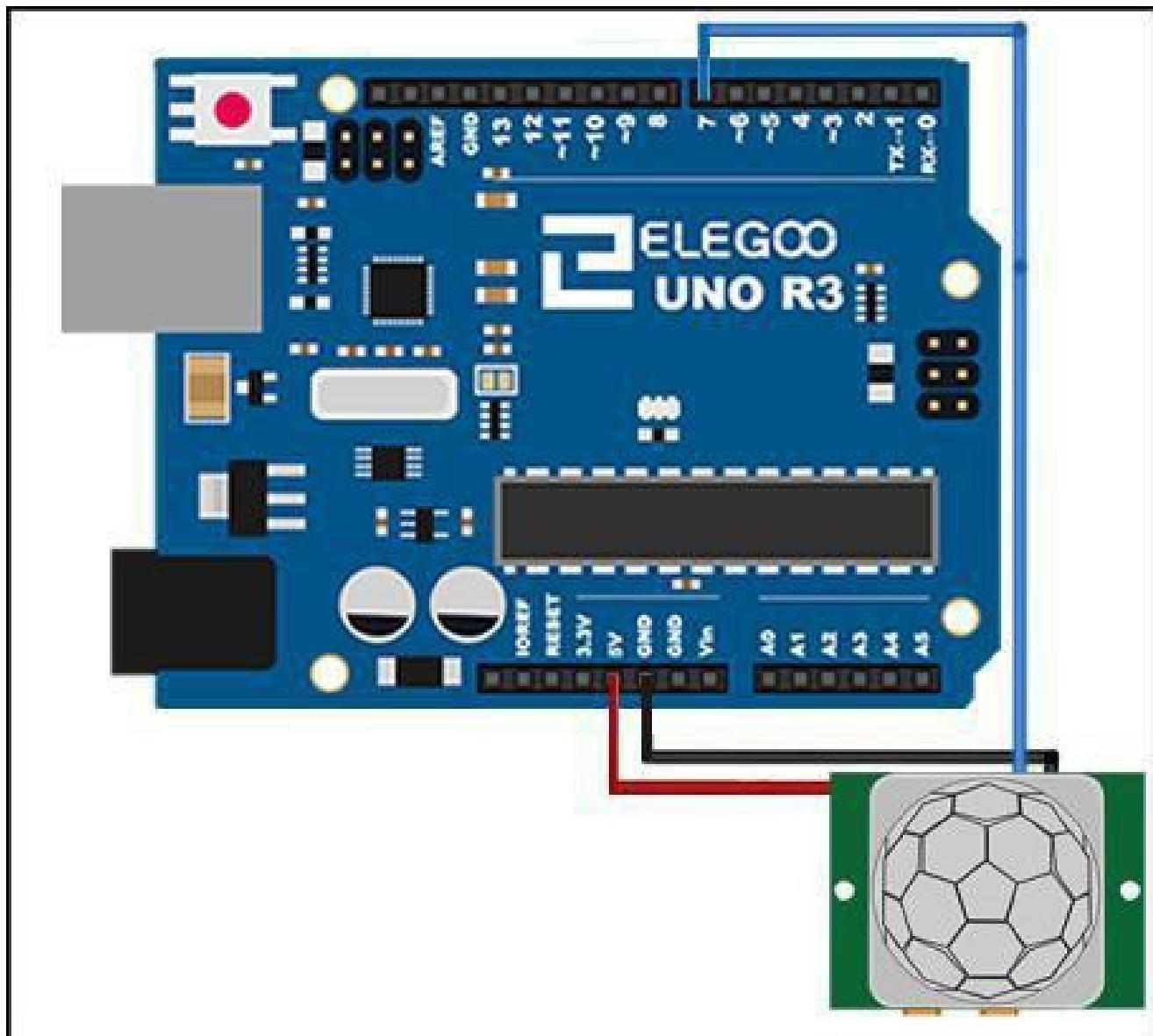
Connection

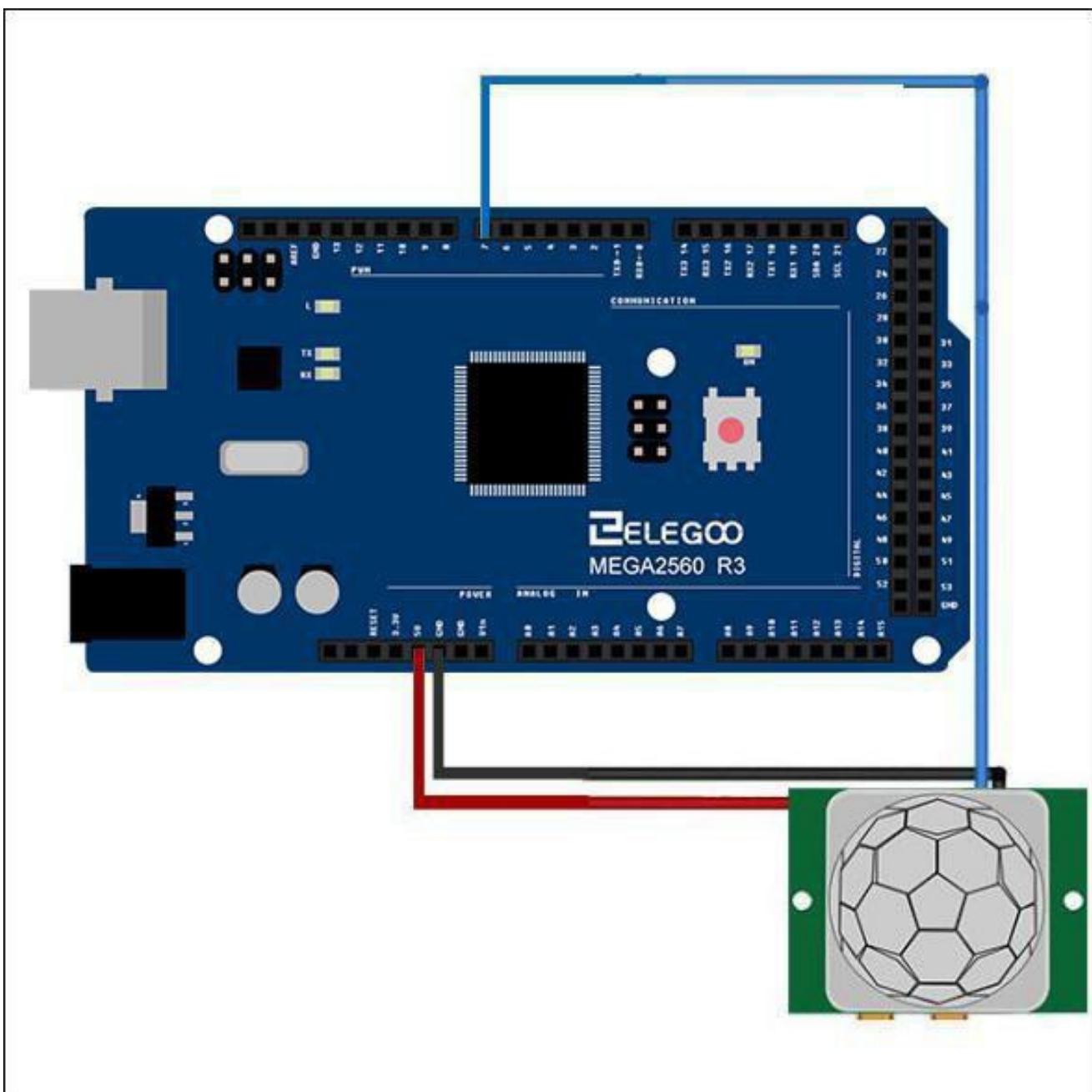
Schematic





Wiring diagram





Connecting PIR sensors to a microcontroller is really simple. The PIR acts as a digital output so all you need to do is listen for the pin to flip high (detected) or low(not detected). It's likely that you'll want retriggering, so be sure to put the jumper in the H position! Power the PIR with 5V and connect ground to ground. Then connect the output to a digital pin. In this example, we'll use pin 2.

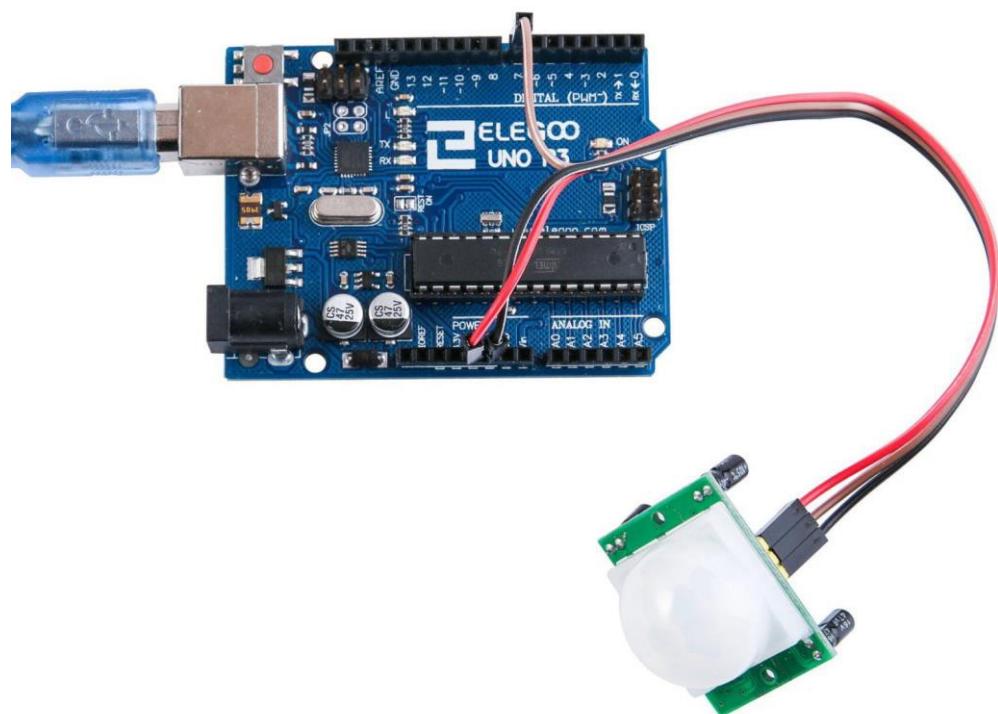
Code

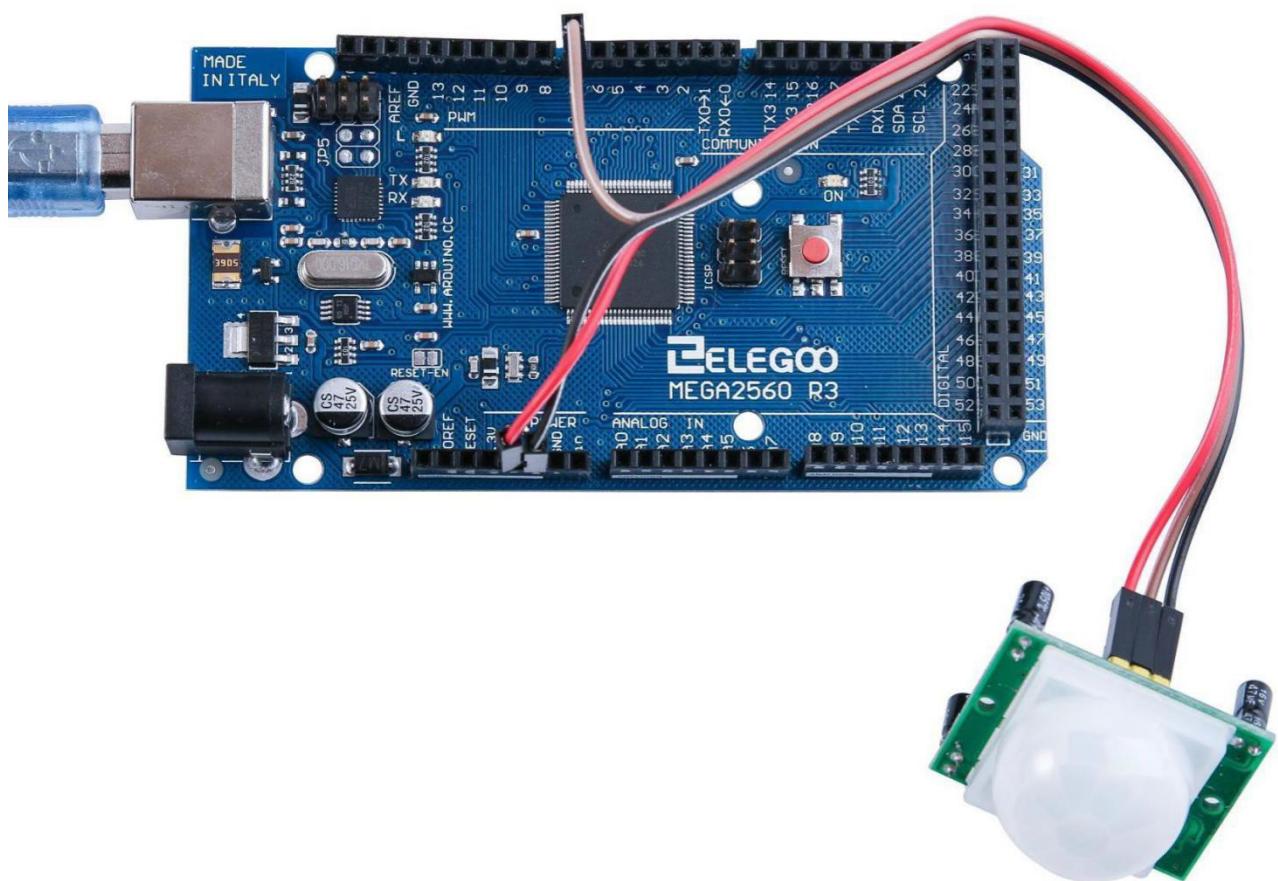
After Wiring, please open the program in the code folder- Lesson 31 HC-SR501 PIR Sensor and click UPLOAD to upload the program. See Lesson 3 for details about program uploading if there are any errors.

The sketch simply turns on Your Arduino LED connected to Pin 13 whenever motion is detected.

Be sure to beware of and somehow handle the 1 minute initialization in whatever application you develop.

Example picture





The followings are the code used in this experiment and their explanations:

```
// choose the pin for the LED
int ledPin = 13;

// choose the input pin (for PIR sensor)
int inputPin = 7;

// we start, assuming no motion detected
int pirState = LOW;
// variable for reading the pin status
int val = 0;

void setup() {
    // declare LED as output
    pinMode(ledPin, OUTPUT);
    // declare sensor as input
    pinMode(inputPin, INPUT);
    Serial.begin(9600);
}

void loop(){
    // read input value
    val = digitalRead(inputPin);
    if (val == HIGH) {
        // check if the input is HIGH
        // turn LED ON
        digitalWrite(ledPin, HIGH);
        if (pirState == LOW) {
            // we have just turned on
            Serial.println("Motion detected!");
            // We only want to print on the output change, not state
            pirState = HIGH;
        }
    }
}
```

```
    }
else
{
    // turn LED OFF
    digitalWrite(ledPin, LOW);
    if (pirState == HIGH){
        // we have just turned off
        Serial.println("Motion ended!");
        // We only want to print on the output change, not state
        pirState = LOW;
    }
}
```

Lesson 32 Water Level Detection Sensor Module

Overview

In this lesson, you will learn how to use a water level detection sensor module. This module can perceive the depth of water and the core component is an amplifying circuit which is made up of a transistor and several pectinate PCB routings. When put into the water, these routings will present a resistor that can change along with the change of the water's depth. Then, the signal of water's depth is converted into the electrical signal, and we can know the change of water's depth through the ADC function of UNO R3.

Component Required:

1x Elegoo Uno R3

1x Water lever detection sensor module

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



Component Introduction

Water sensor:

A water sensor brick is designed for water detection, which can be widely used in sensing the rainfall, water level, even the liqueate leakage. The brick is mainly composed of three parts: an electronic brick connector, a $1\text{ M}\Omega$ resistor, and several lines of bare conducting wires.

This sensor works by having a series of exposed traces connected to ground. Interlaced between the grounded traces are the sense traces.

The sensor traces have a weak pull-up resistor of $1\text{ M}\Omega$. The resistor will pull the sensor trace value high until a drop of water shorts the sensor trace to the grounded trace. Believe it or not this circuit will work with the digital I/O pins of your UNO R3

board or you can use it with the analog pins to detect the amount of water induced contact between the grounded and sensor traces.

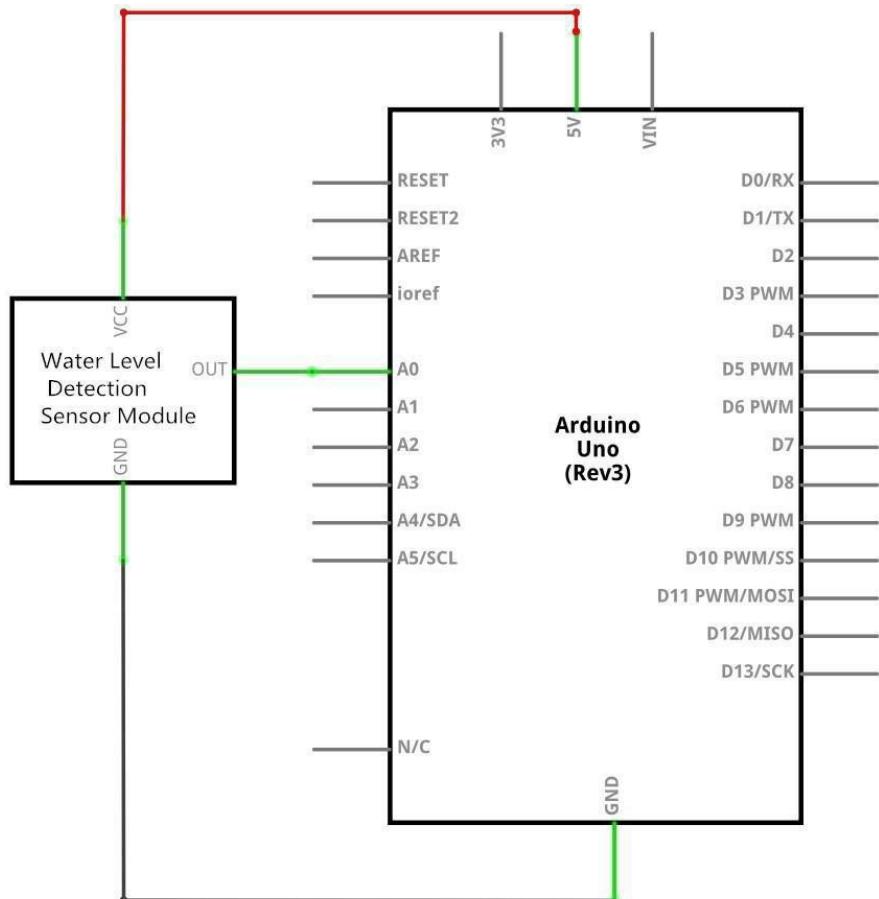
This item can judge the water level through with a series of exposed parallel wires stitch to measure the water droplet/water size. It can easily change the water size to analog signal, and output analog value can directly be used in the program function, then to achieve the function of water level alarm.

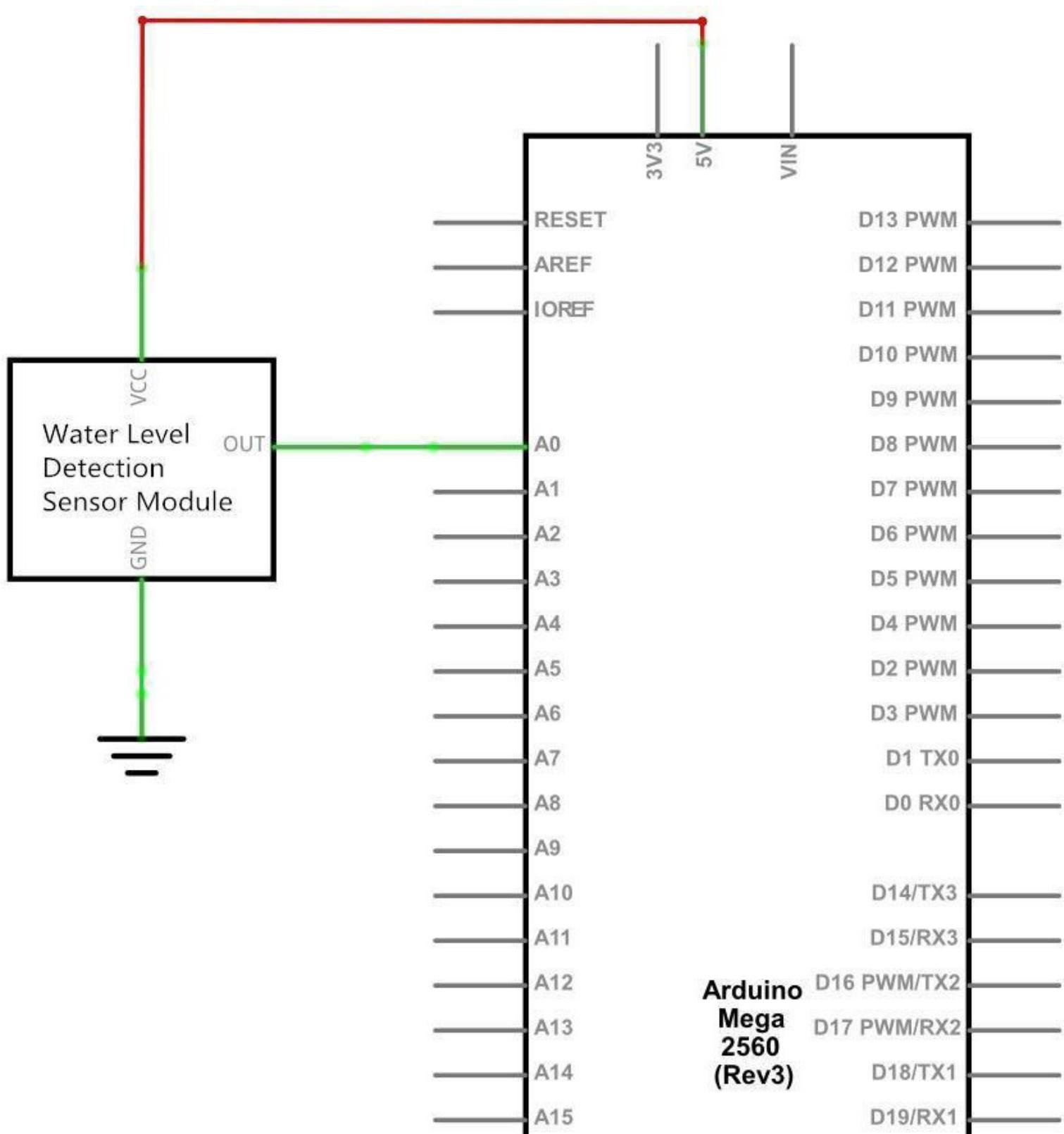
It has low power consumption, and high sensitivity. Features:

1. Working voltage: 5V
2. Working Current: <20ma
3. Interface: Analog
4. Width of detection: 40mm×16mm
5. Working Temperature: 10°C~30°C
6. Output voltage signal: 0~4.2V

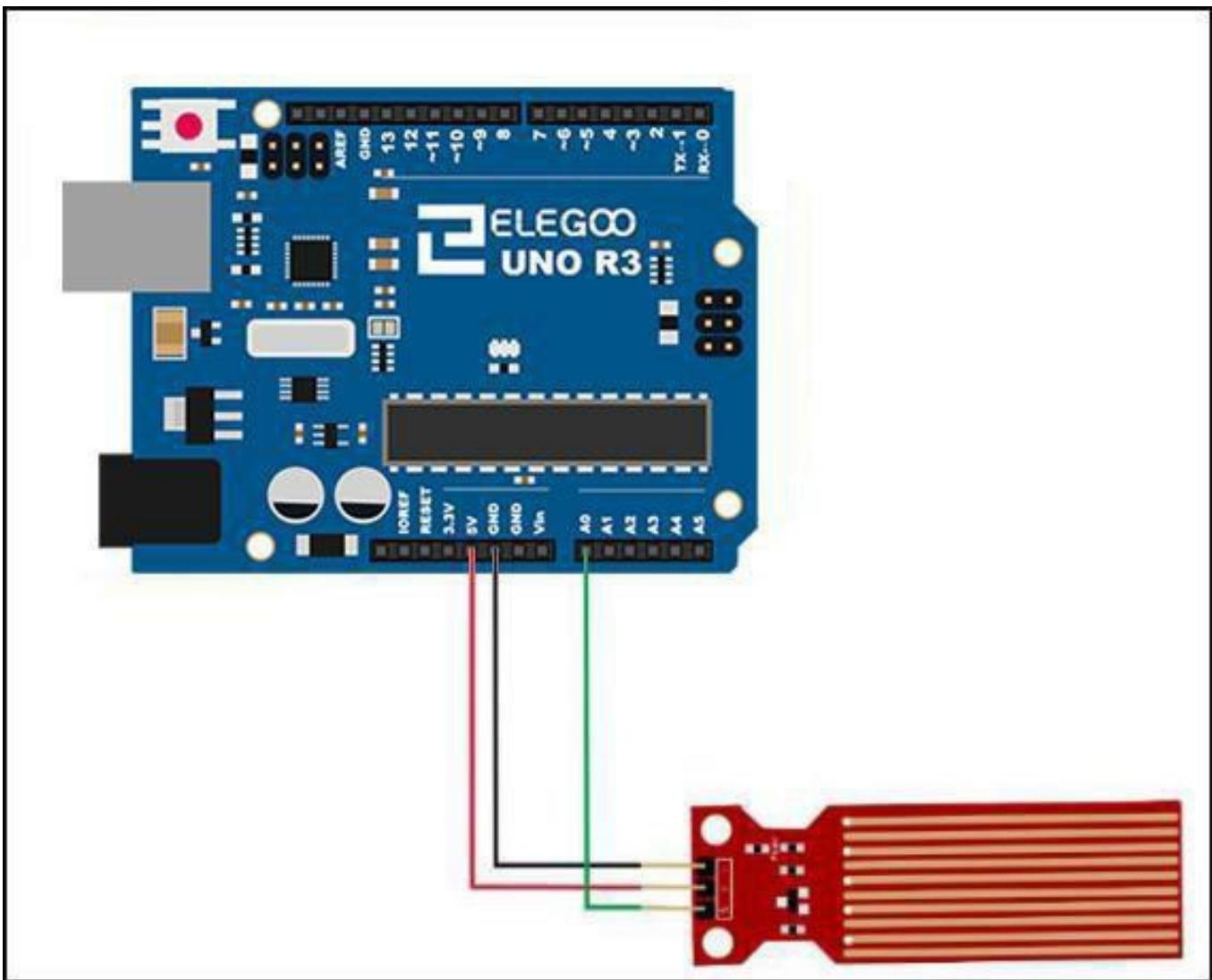
Connection

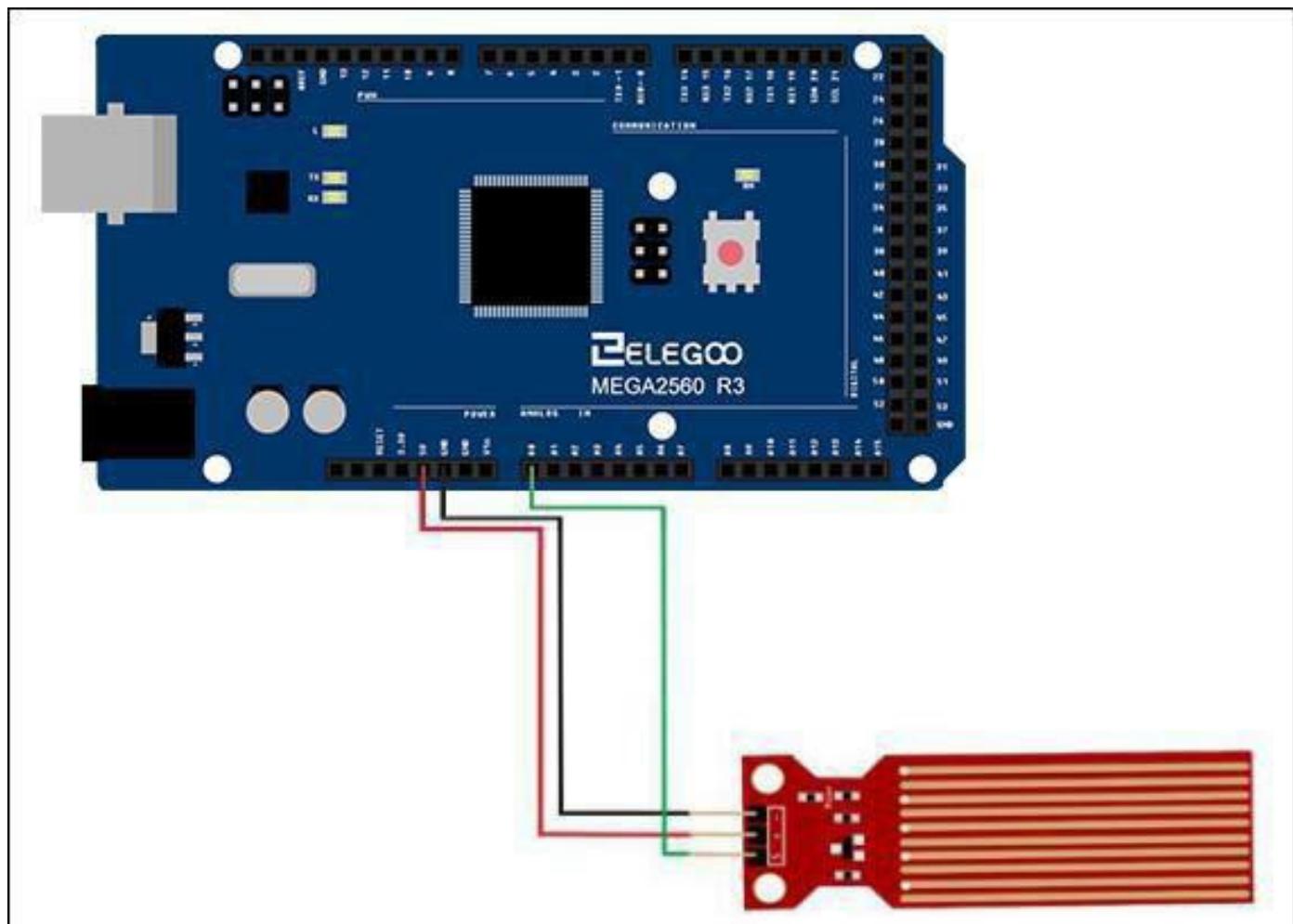
Schematic





Wiring diagram



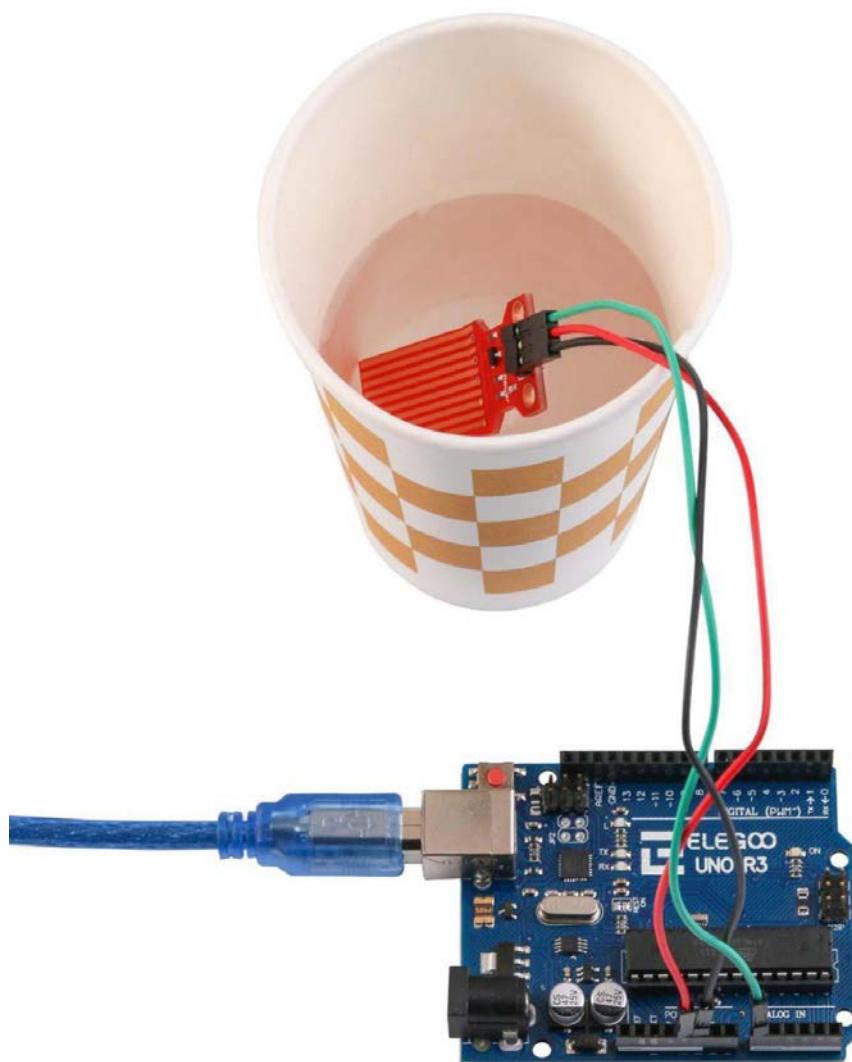


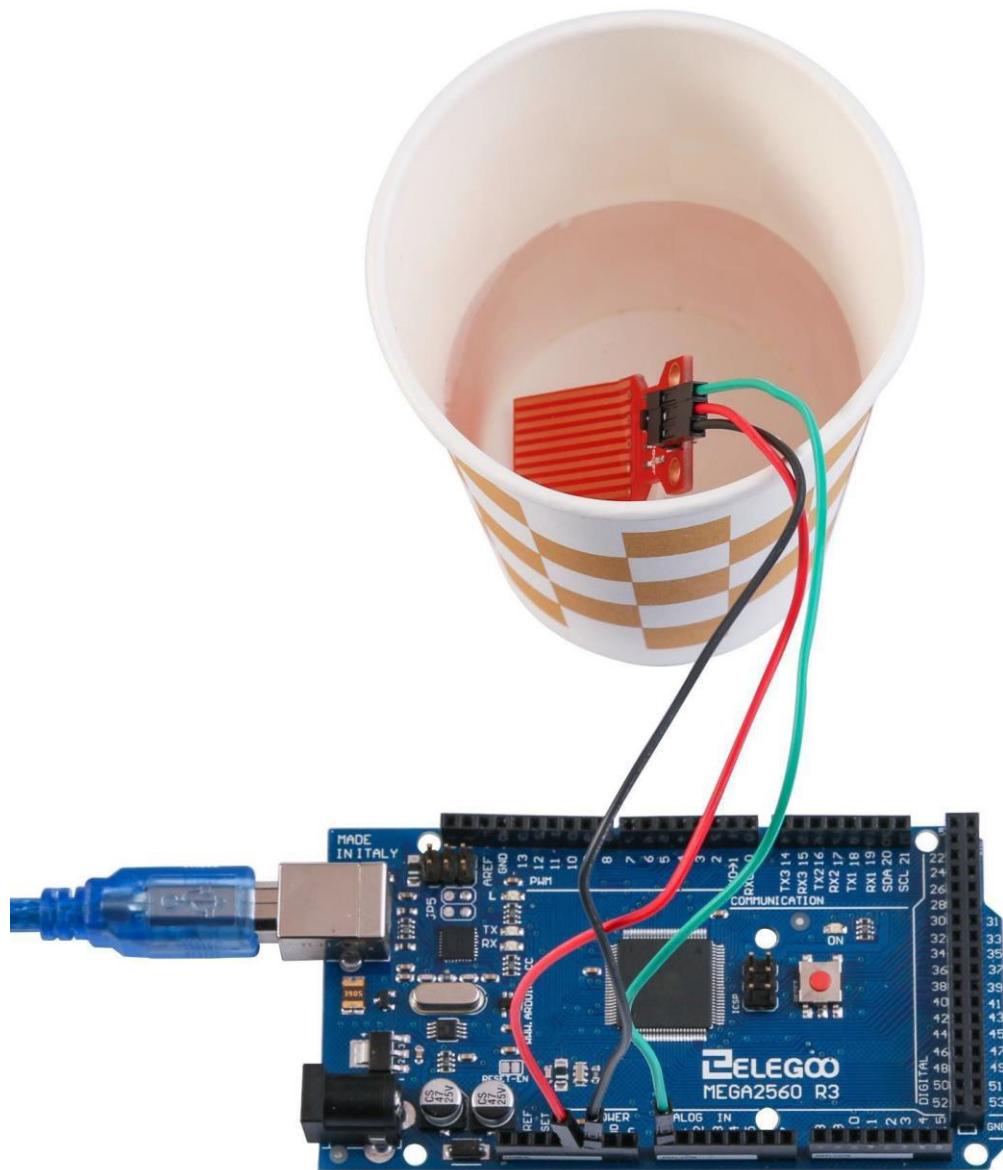
Wiring tips: Power supply (+) is connected to 5V of UNO R3 board, ground electrode (-) is connected to GND. Signal output (S) is connected to the ports (A0-A5) which have function of inputting analog signal in UNO R3 board, random one is OK, but it should define the same demo code as the routine.

Code

After wiring, please open the program in the code folder- Lesson 32 Water Level Detection Sensor Module and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

Example picture





Open the monitor then you can see the data asbelow:

Click the Serial Monitor button to turn on the serial monitor. The basics about the serial monitor are introduced in details in Lesson1.

The screenshot shows the Arduino Serial Monitor window. The title bar says "COM215". The main area displays a series of text lines: "ADCO level is 11", "ADCO level is 22", "ADCO level is 33", "ADCO level is 44", "ADCO level is 55", "ADCO level is 66", "ADCO level is 77", "ADCO level is 88", "ADCO level is 99", "ADCO level is 110", "ADCO level is 127", "ADCO level is 138", "ADCO level is 149", "ADCO level is 160", "ADCO level is 171", "ADCO level is 182", "ADCO level is 193", "ADCO level is 204", "ADCO level is 215", "ADCO level is 226", "ADCO level is 237", "ADCO level is 248", "ADCO level is 259", "ADCO level is 270", "ADCO level is 259", "ADCO level is 248", "ADCO level is 237", "ADCO level is 226", and "ADCO level is 215". At the bottom, there are three buttons: "Autoscroll" (checked), "Newline" (dropdown menu), and "9600 baud" (dropdown menu).

The followings are the code used in this experiment and their explanations:

```
int adc_id = 0;  
int HistoryValue = 0;  
char printBuffer[128];  
  
void setup()  
{  
    Serial.begin(9600);  
}  
  
void loop()  
{  
    int value = analogRead(adc_id); // get adc value  
  
    if(((HistoryValue>=value) && ((HistoryValue - value) > 10)) || ((HistoryValue<value) &&  
        ((value - HistoryValue) > 10)))  
    {  
        sprintf(printBuffer,"ADC%d level is %d\n",adc_id, value);  
  
        Serial.print(printBuffer);  
        HistoryValue = value;  
    }  
}
```

Lesson 33 Real Time Clock Module

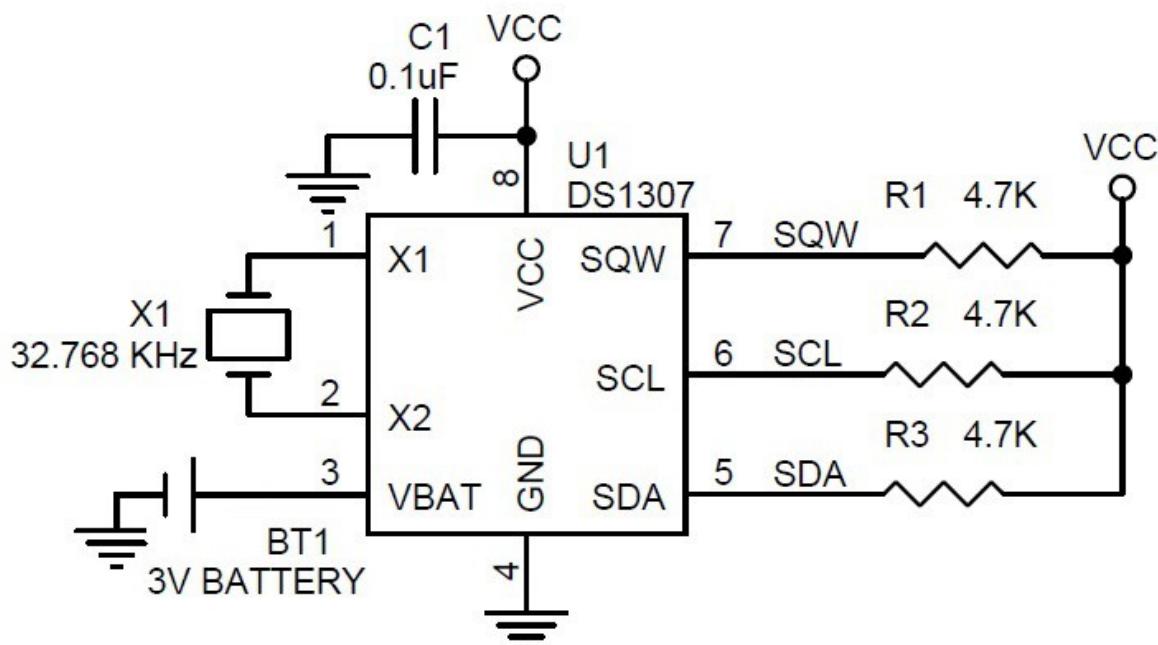
Overview

In this lesson, you will learn how to use the RTC module, The DS1307 real-time clock is a low-power chip. Address and data are transferred serially through an I₂C, which can be used unless being connected to UNO with only three data cables. DS1307 provides seconds, minutes, hours, day, date, month, and year information. Timekeeping operation continues while the part operates from the backup supply.

Component Required:

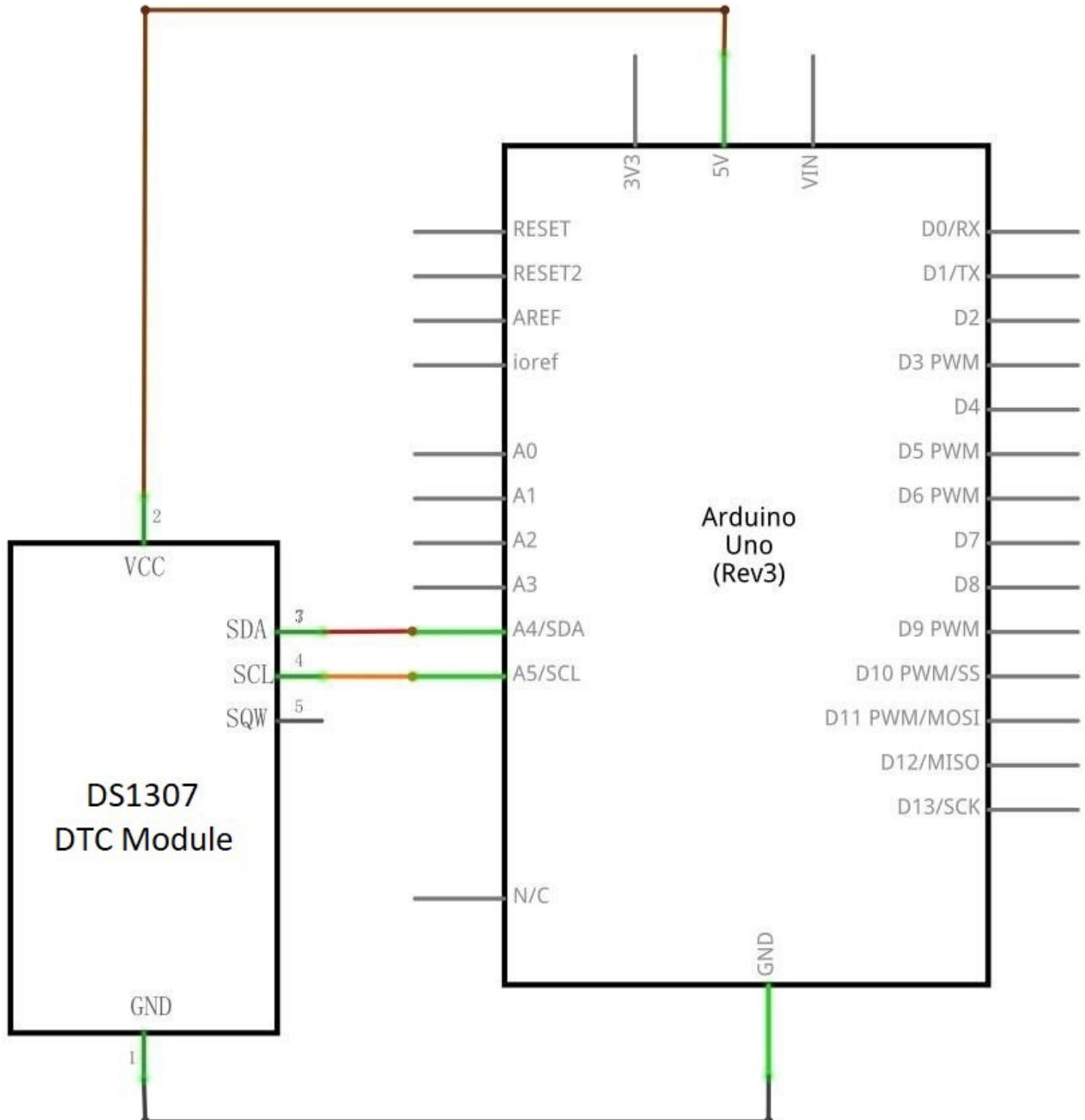
- 1x Elegoo Uno R3
- 1x DS1307 RTC module
- 4x F-M wires (Female to Male DuPont wires)

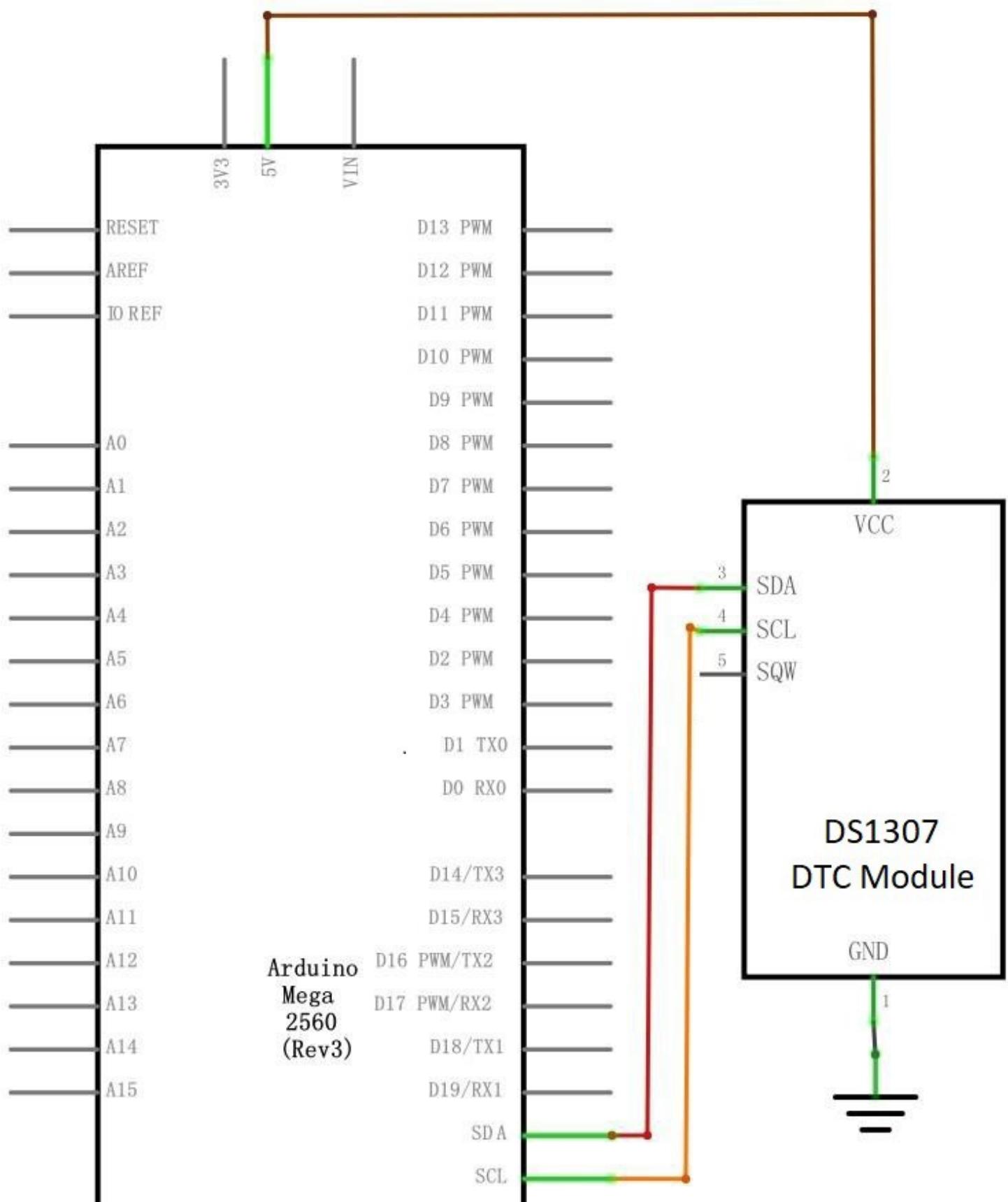
Component Introduction



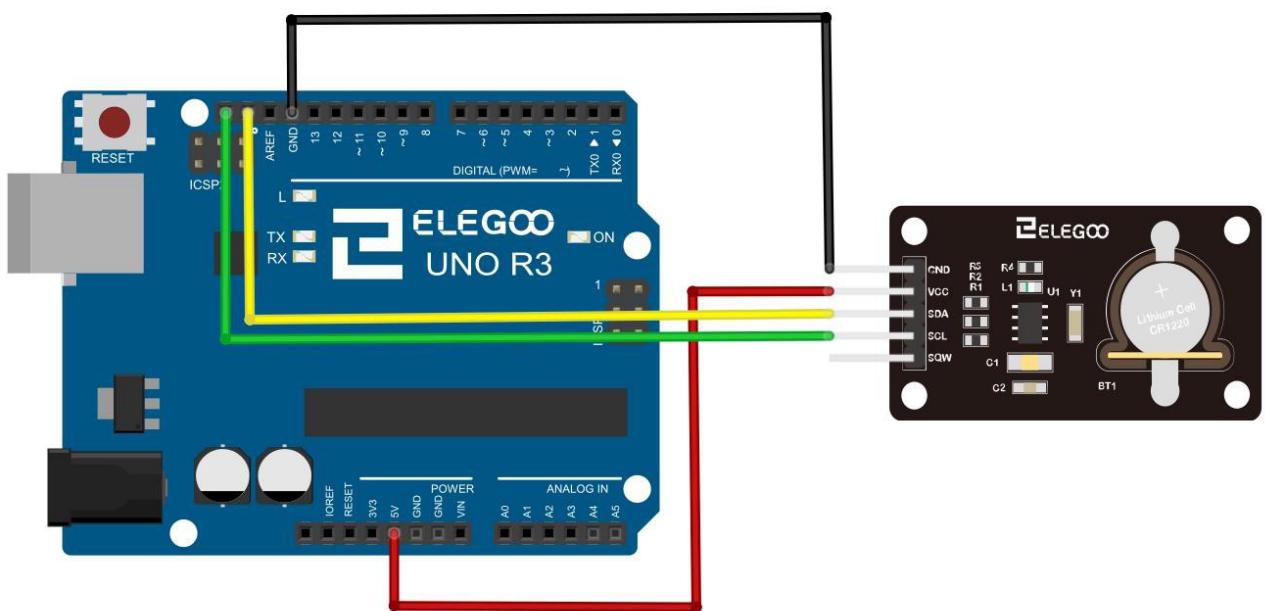
NOTICE: If there is the left version of module in the Kit, don't worry, its function and pin names are the same as the new version. Just follow the wiring diagrams and sketch in the tutorial below to get it working.

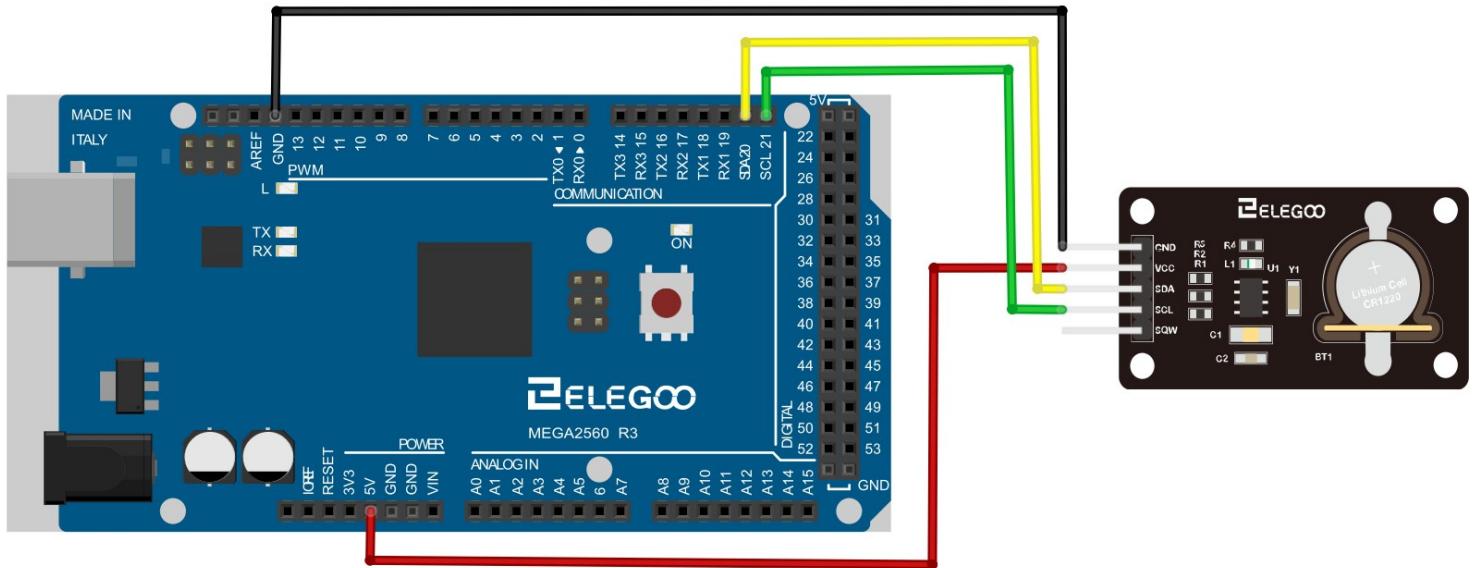
Connection Schematic





Wiring diagram





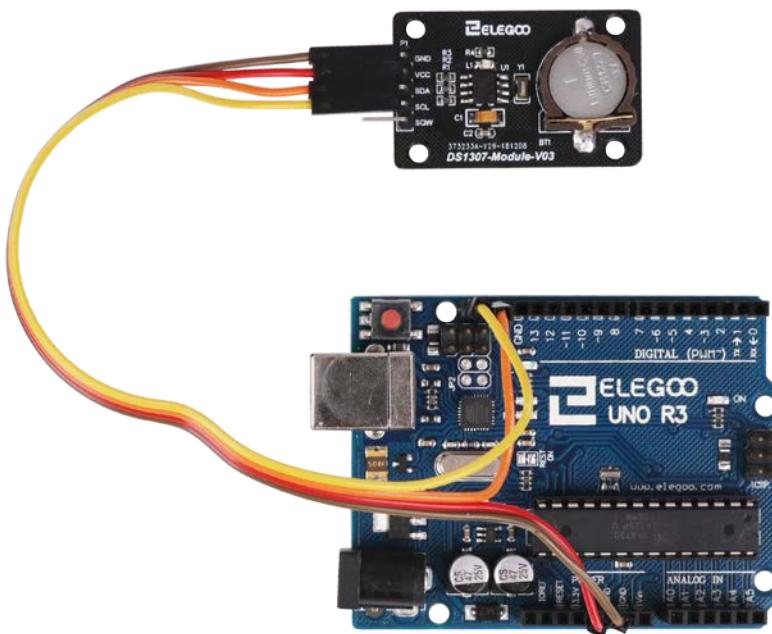
Set up according to the following image.

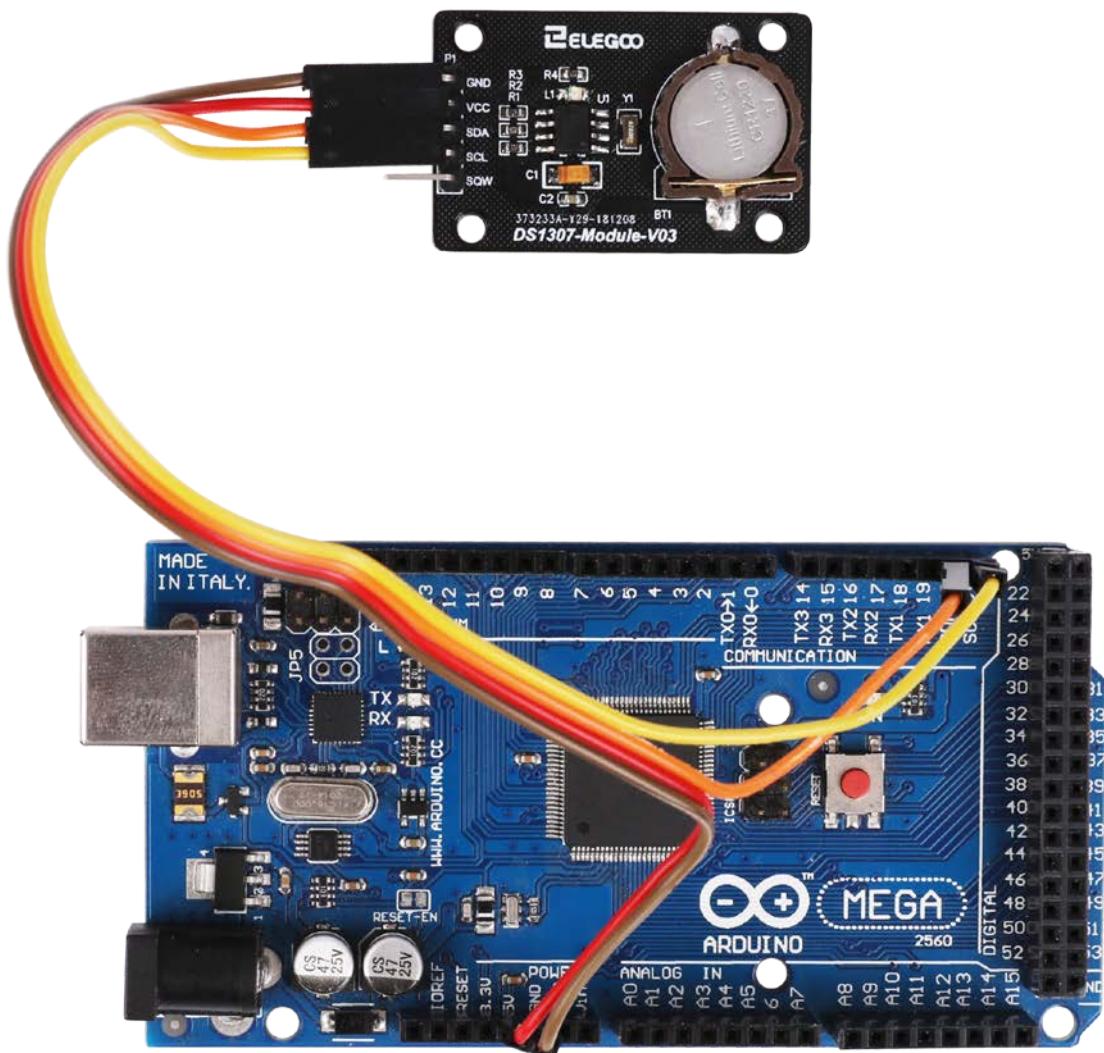
Note:Ignore the SQW pin; you will not need them in this Lesson. Plug the SCL pin into your UNO R3 board SCL port, and the SDA pin into the SDA port. The VCC pin plugs into the 5V port, and the GND plugs into the GND port.

Code

After wiring, please open program in the code folder- Lesson33 Real Time Clock 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 < DS1307 > library or re-install it, if necessary. Otherwise, your code won't work. For details about loading the library file, see Lesson 1.

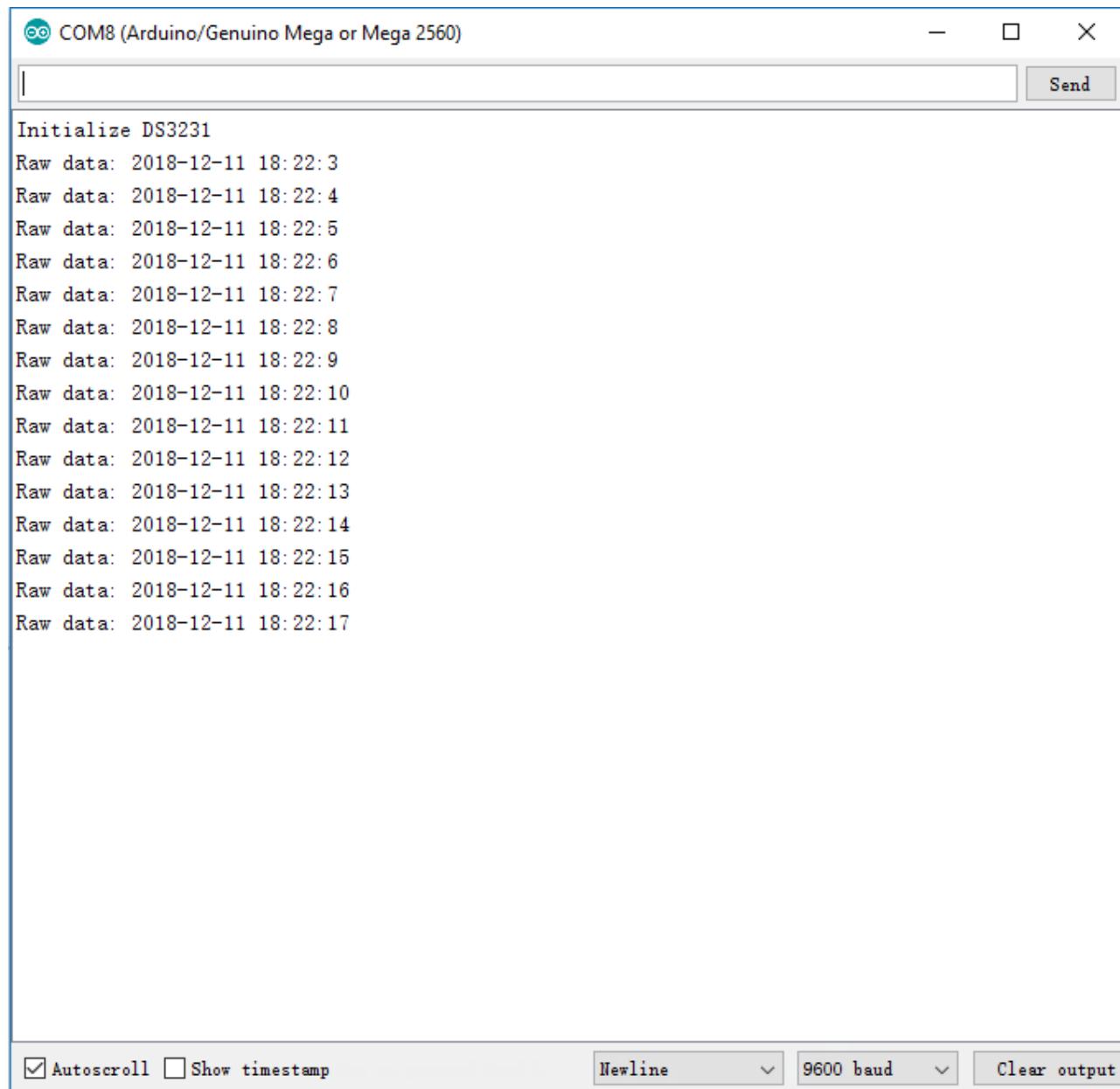
Example picture





Open the monitor then you can see the module can read the time as below:

Click the Serial Monitor button to turn on the serial monitor. The basics about the serial monitor are introduced in details in Lesson 2.



The screenshot shows the Arduino Serial Monitor window titled "COM8 (Arduino/Genuino Mega or Mega 2560)". The window displays a series of timestamped raw data outputs. The first line is "Initialize DS3231". Subsequent lines show "Raw data: 2018-12-11 18:22:3", "Raw data: 2018-12-11 18:22:4", "Raw data: 2018-12-11 18:22:5", "Raw data: 2018-12-11 18:22:6", "Raw data: 2018-12-11 18:22:7", "Raw data: 2018-12-11 18:22:8", "Raw data: 2018-12-11 18:22:9", "Raw data: 2018-12-11 18:22:10", "Raw data: 2018-12-11 18:22:11", "Raw data: 2018-12-11 18:22:12", "Raw data: 2018-12-11 18:22:13", "Raw data: 2018-12-11 18:22:14", "Raw data: 2018-12-11 18:22:15", "Raw data: 2018-12-11 18:22:16", and "Raw data: 2018-12-11 18:22:17". At the bottom of the monitor window, there are three checkboxes: "Autoscroll" (checked), "Show timestamp" (unchecked), and "Clear output". There are also dropdown menus for "Newline" and "9600 baud".

The followings are the code used in this experiment and their explanations:

```
#include <Wire.h>
#include <DS3231.h>

DS3231 clock;
RTCDateTime dt;

void setup()
{
    Serial.begin(9600);

    Serial.println("Initialize DS3231");
    // Initialize DS3231
    clock.begin();

    // Manual setting format YYYY, MM, DD, HH, II, SS
    // clock.setDateTime(2016, 12, 9, 11, 46, 00);

    // Send sketch compiling time to Arduino
    clock.setDateTime(    DATE    ,    TIME    );
/*
    Tips: This command will be executed every time when Arduino restarts.
    Comment this line out to store the memory of DS3231 module
*/
}

void loop()
{
    dt = clock.getDateTime();

    // For leading zero look to DS3231_dateformat example

    Serial.print("Raw data: ");
    Serial.print(dt.year); Serial.print("-");
    Serial.print(dt.month); Serial.print("-");
    Serial.print(dt.day); Serial.print(" ");
    Serial.print(dt.hour); Serial.print(":");
    Serial.print(dt.minute); Serial.print(":");
    Serial.print(dt.second); Serial.println("");

    delay(1000);
}
```

Lesson 34 Keypad Module

Overview

In this project, we will go over how to integrate a keyboard with an UNO R3 board so that the UNO R3 can read the keys being pressed by a user.

Keypads are used in all types of devices, including cell phones, fax machines, microwaves, ovens, door locks, etc. They're practically everywhere. Tons of electronic devices use them for userinput.

So knowing how to connect a keypad to a microcontroller such as an UNO R3 board is very valuable for building many different types of commercial products.

At the end when all is connected properly and programmed, when a key is pressed, it shows up at the Serial Monitor on your computer. Whenever you press a key, it shows up on the Serial Monitor. For simplicity purposes, we start at simply showing the key pressed on the computer.

For this project, the type of keypad we will use is a matrix keypad. This is a keypad that follows an encoding scheme that allows it to have much less output pins than there are keys. For example, the matrix keypad we are using has 16 keys (0-9, A-D, *, #), yet only 8 output pins. With a linear keypad, there would have to be 17 output pins (one for each key and a ground pin) in order to work. The matrix encoding scheme allows for less output pins and thus much less connections that have to make for the keypad to work. In this way, they are more efficient than linear keypads, being that they have less wiring.

Component Required:

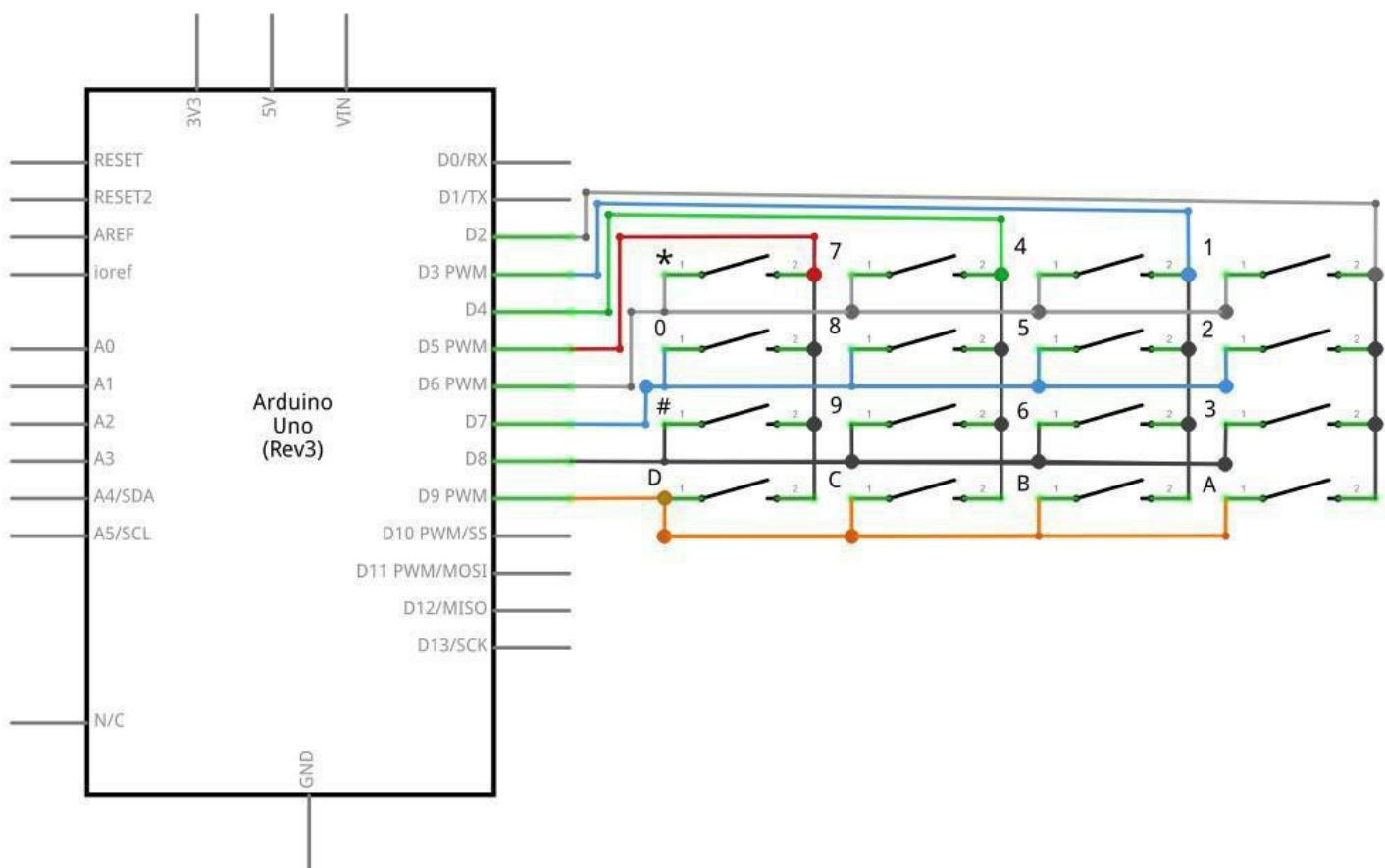
1 x Elegoo Uno R3

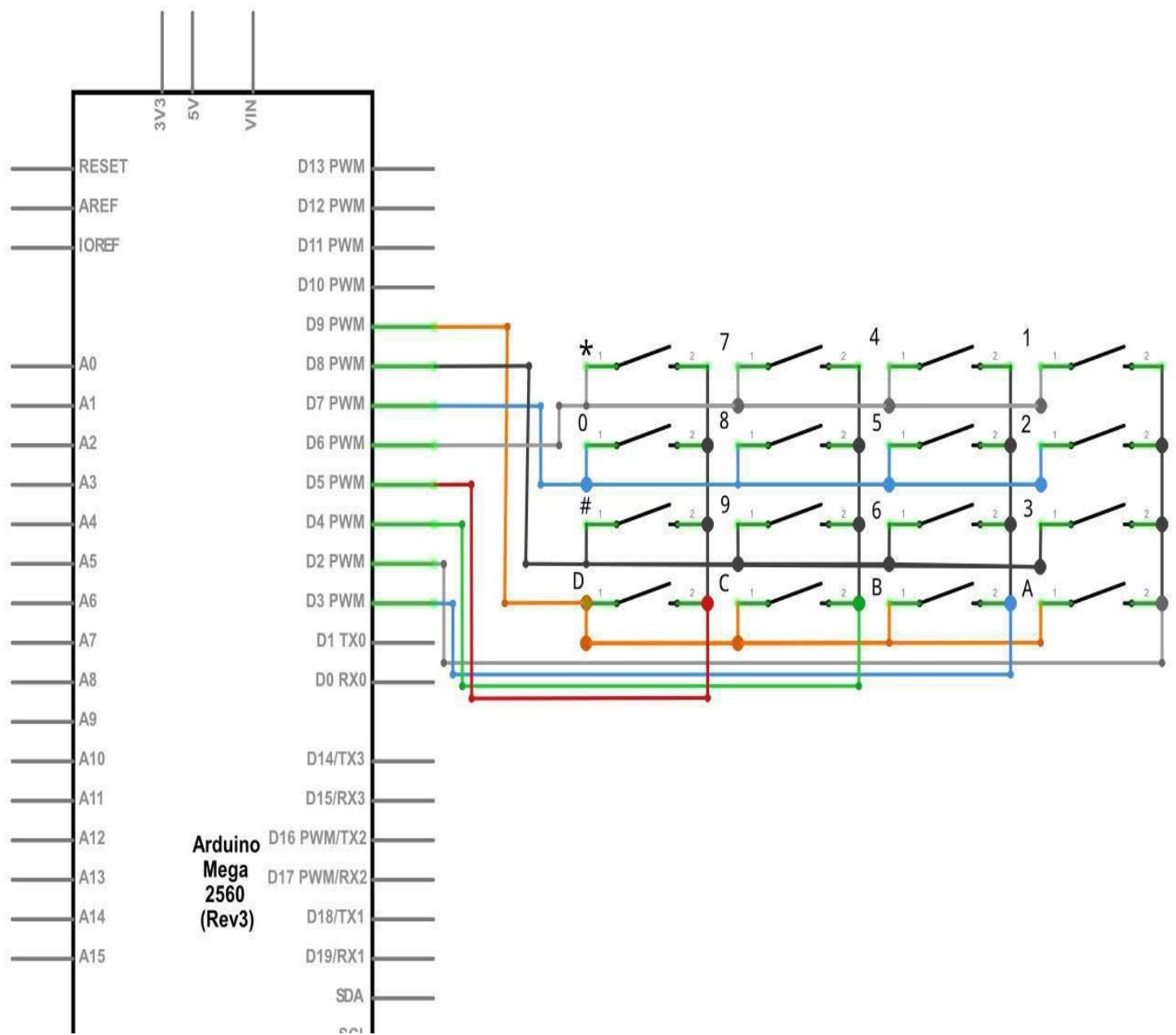
1 x Membrane switch module

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

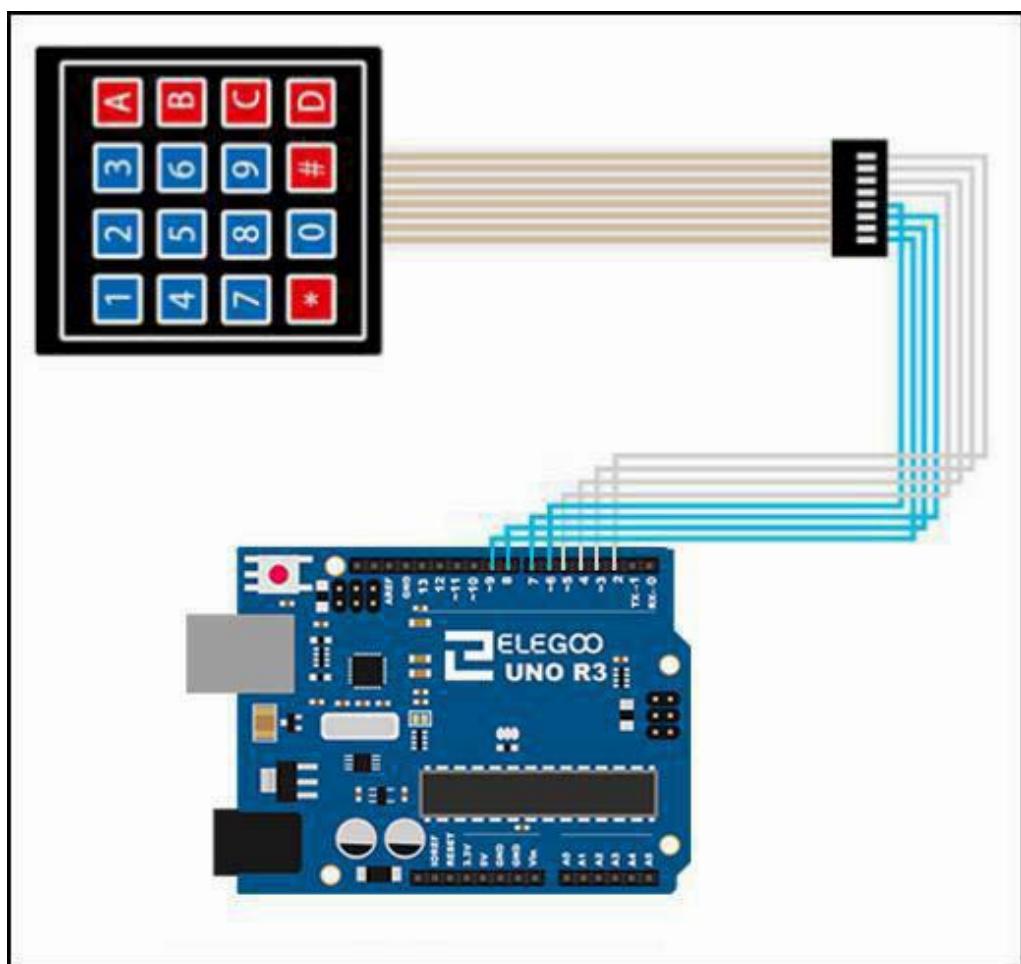


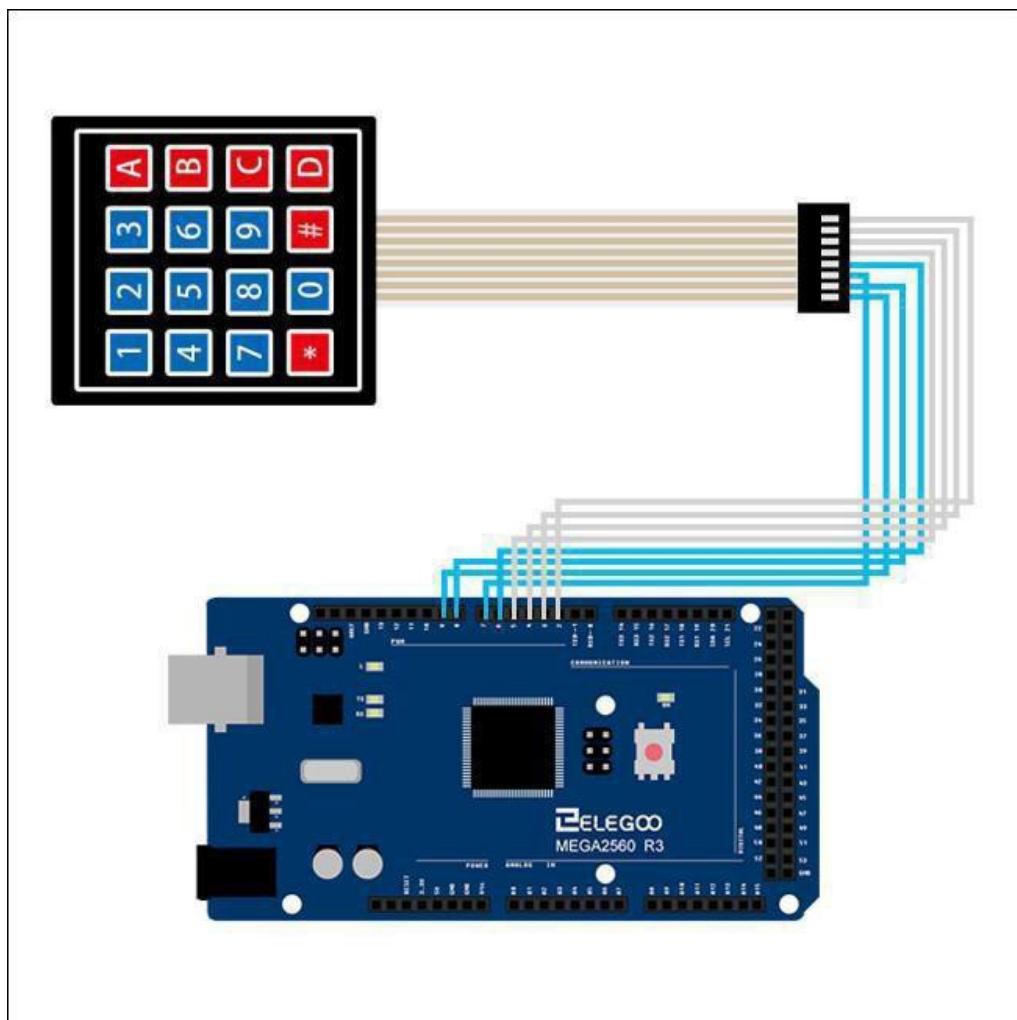
Connection Schematic





Wiring diagram





When connecting the pins to the UNO R3 board, we connect them to the digital output pins, D9-D2. We connect the first pin of the keypad to D9, the second pin to D8, the third pin to D7, the fourth pin to D6, the fifth pin to D5, the sixth pin to D4, the seventh pin to D3, and the eighth pin to D2.

These are the connections in a table:

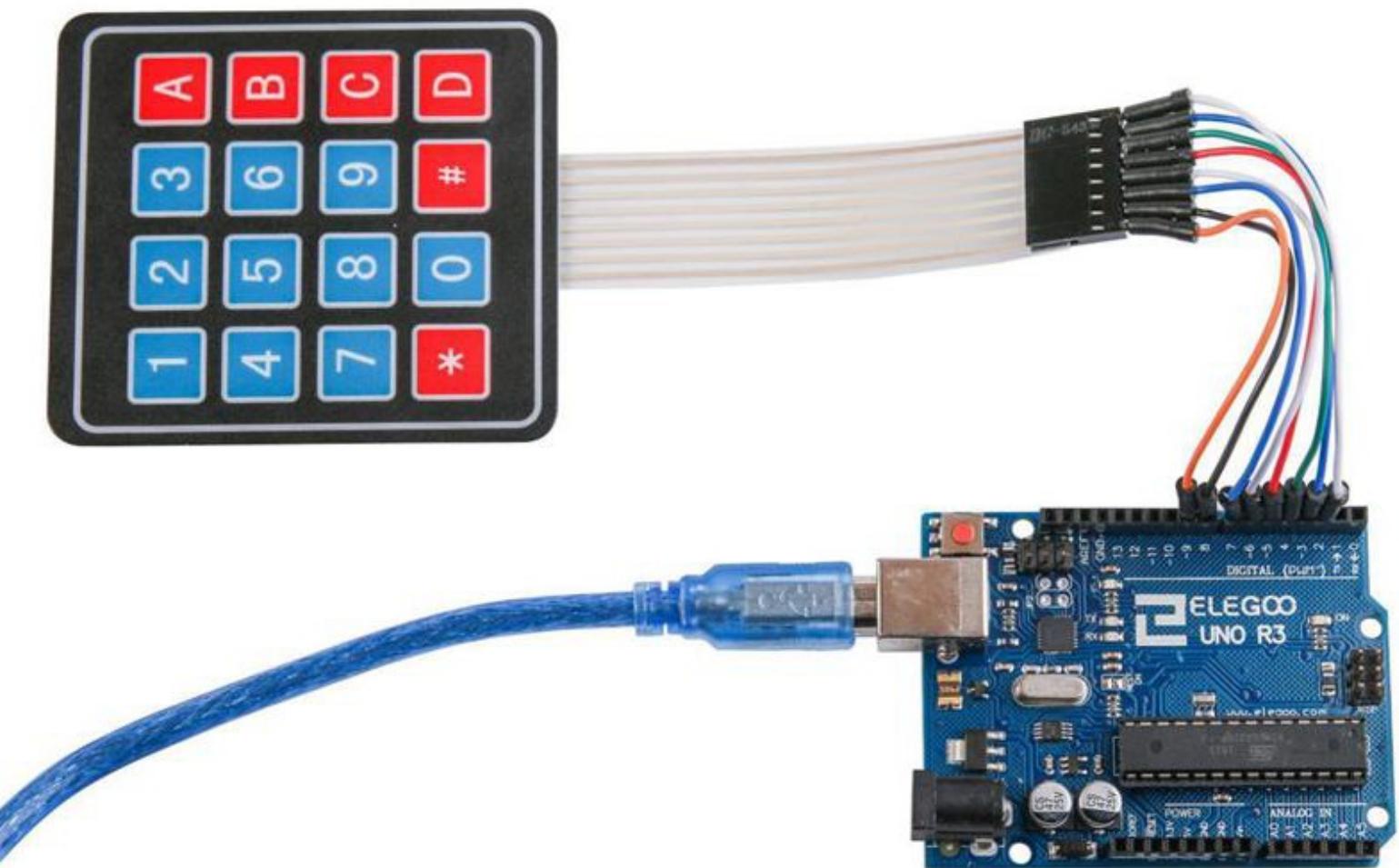
Keypad Pin	Connects to Arduino Pin...
1	D9
2	D8
3	D7
4	D6
5	D5
6	D4
7	D3
8	D2

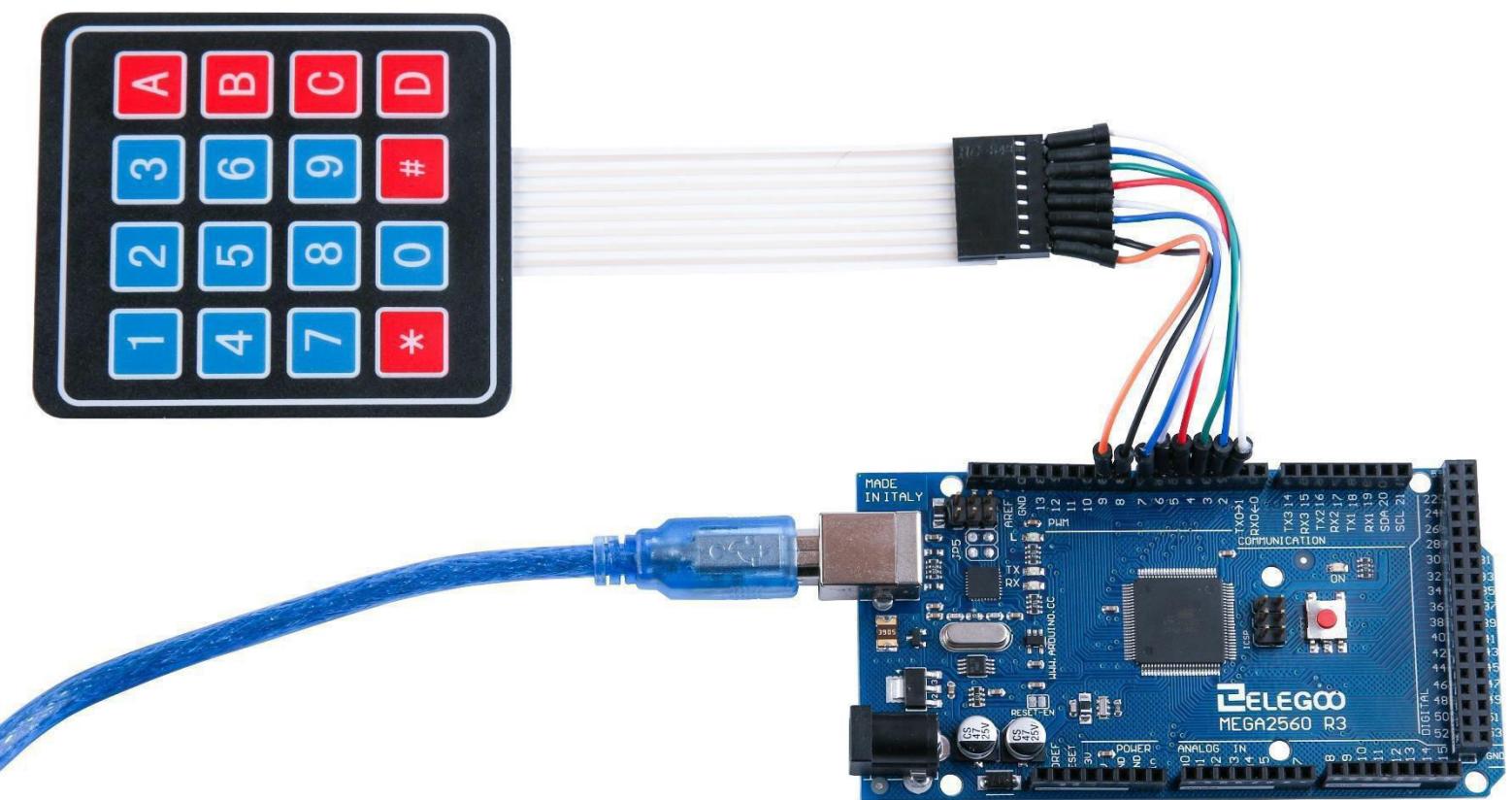
Code

After wiring, please open the program in the code folder- Lesson 34 Keypad 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 < Keypad > library or re-install it, if necessary. Otherwise, your code won't work.

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

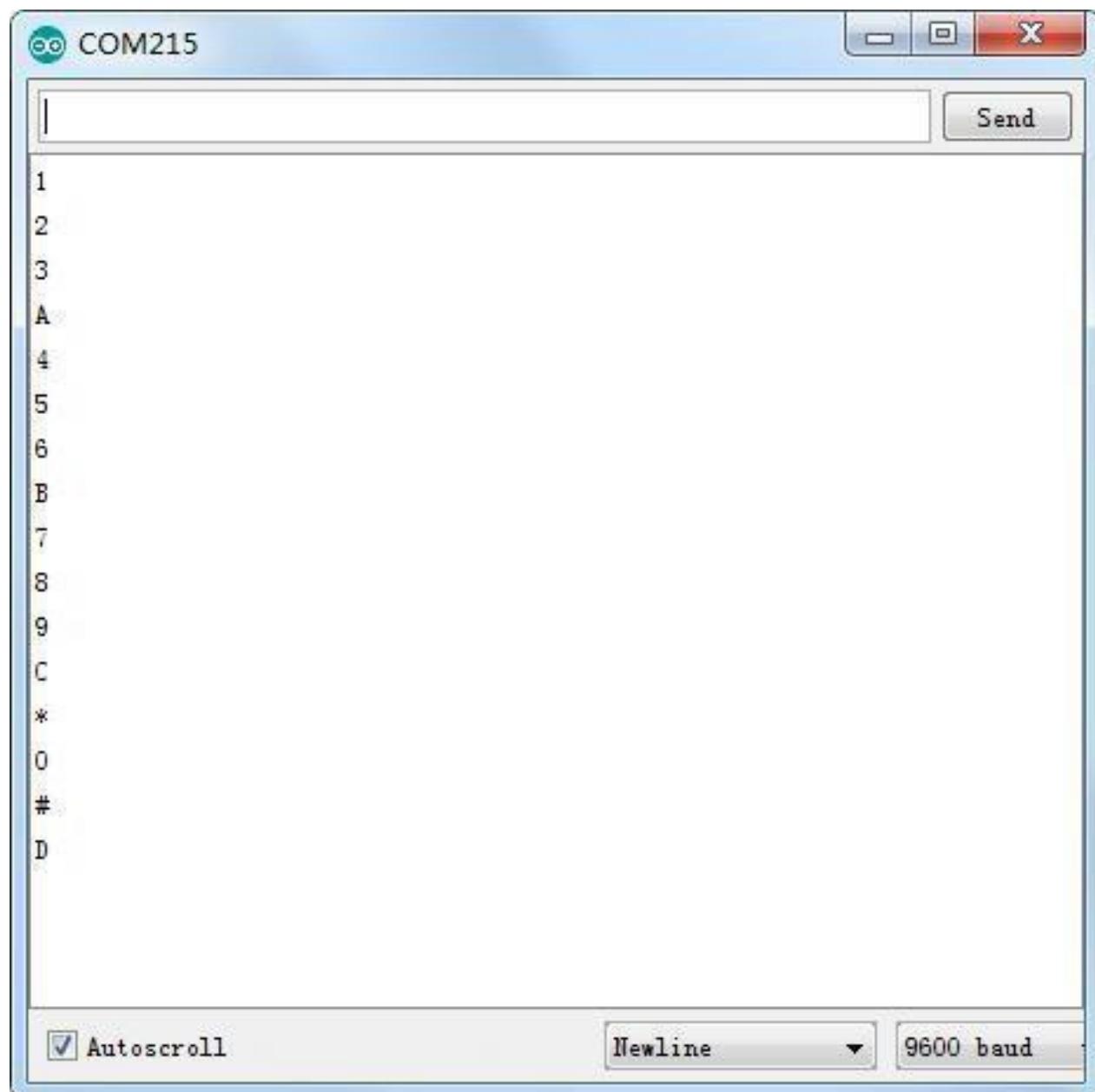
Example picture





With this code, once we press a key on the keypad, it should show up on the serial monitor of the Arduino software once the code is compiled and uploaded to the UNO R3 board.

[Click the Serial Monitor button to turn on the serial monitor.](#) The basics about the serial monitor are introduced in details in [Lesson2](#).



The following is the code needed to use this experiment and explain:

```
#include <Keypad.h>
//four rows
const byte ROWS = 4;
//four columns
const byte COLS = 4;
//define the symbols on the buttons of the keypads
char hexaKeys[ROWS][COLS] = {
    {'1','2','3','A'},
    {'4','5','6','B'},
    {'7','8','9','C'},
    {'*','0','#','D'}
};

//connect to the row pinouts of the keypad
byte rowPins[ROWS] = {9, 8, 7, 6};
//connect to the column pinouts of the keypad
byte colPins[COLS] = {5, 4, 3, 2};
//initialize an instance of class NewKeypad
Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);
void setup(){
    Serial.begin(9600);
}
void loop(){
    char customKey = customKeypad.getKey();

    if (customKey){
        Serial.println(customKey);
    }
}
```

Conclusion

As the creator and editor of this tutorial, we try our very best to teach you how to learn and how to solve problems so that gradually you will be able to handle different issues by yourself. If you'd like to modify the kit or any one of the projects, feel free to release your imagination and innovation.

If you have any questions or problems while building up your projects, feel free to let us know but please also give us some time to response you email. ;) While waiting for our reply, you can also do an online research first, there are many communities you can go to and find out solutions. It's going to be a fulfilling experience by solving problems on your own, which is also a great way of self-teaching. But if you prefer to talk with our customer service, leave us an message at service@elegoo.com or euservice@elegoo.com if you are from Europe and Japan. ;) Thank you for your support and we hope you can have a lot of fun with the kit. ;)