

Preface

About SunFounder

SunFounder is a company focused on STEM education with products like opensource robots, aircraft models, and smart devices distributed globally. In SunFounder, we strive to help elementary and middle school students as well as hobbyists, through STEM education, strengthen their hands-on practices and problem-solving abilities. In this way, we hope to disseminate knowledge and provide skill training in a full-of-joy way, thus fostering your interest in programming and making, and exposing you to a fascinating world of science and engineering. To embrace the future of artificial intelligence, it is urgent and meaningful to learn abundant STEM knowledge.

About the Vincent Kit

This is a kit suitable for beginners and knowledgable Arduino learners. The kit uses Mega2560 as the main control board, containing 64 commonly used input and output components, modules and so many basic components (such as resistors, capacitors, transistors, etc.), and it can provide powerful help for you to finish your programming projects.

Free Support



If you have any **TECHNICAL questions**, add a topic under **FORUM** section on our website and we'll reply as soon as possible.



For **NON-TECH questions** like order and shipment issues, please **send an email to service@sunfounder.com**. You're also welcomed to share your projects on FORUM

Contents

Components list.....	1
Introduction.....	9
Part 1: Basic.....	10
1.1 Arduino IDE.....	10
1.2 Digital Write.....	21
1.3 Analog Write.....	29
1.4 Digital Read.....	35
1.5 Analog Read.....	40
1.6 Digital Input Control Output.....	45
1.7 Analog Input Control Output.....	50
1.8 Serial Read.....	55
1.9 Digital Input Pull-Up.....	63
1.10 State Change Detection.....	68
Part 2: Component.....	81
2.1 Common Component.....	81
2.2 LED.....	89
2.3 RGB LED.....	95
2.4 LED Bar Graph.....	102
2.5 7-Segment Display.....	108

2.6 74HC595.....	118
2.7 4-Digital 7-Segment Display.....	125
2.8 LED Matrix Module.....	134
2.9 I2C LCD1602 Module.....	144
2.10 Active Buzzer.....	152
2.11 Passive Buzzer.....	157
2.12 Servo.....	166
2.13 Motor.....	173
2.14 Stepper Motor.....	184
2.15 Button.....	193
2.16 Slide Switch.....	199
2.17 Tilt Switch.....	204
2.18 Touch Switch Module.....	209
2.19 Keypad.....	214
2.20 IR Receiver Module.....	221
2.21 Relay Module.....	227
2.22 Potentiometer.....	233
2.23 Joystick Module.....	238
2.24 MPR121 Module.....	244
2.25 Rotary Encoder Module.....	251
2.26 Photoresistor.....	258
2.27 Thermistor.....	262

2.28 Sound Sensor Module.....	267
2.29 Water Sensor Module.....	272
2.30 IR Obstacle Avoidance Sensor.....	277
2.31 PIR Module.....	281
2.32 DHT11 Module.....	287
2.33 Ultrasonic Module.....	293
2.34 MPU6050 Module.....	300
2.35 RFID-RC522 Module.....	309
Part 3: Example.....	317
3.1 Reversing Aid.....	317
3.2 Pedestrian Crossing Button.....	322
3.3 Overheat Monitor.....	328
3.4 Guess Number.....	336
3.5 Access Control System.....	342
Part 4: Appendix.....	349
4.1 Add Libraries.....	349

Components list



RGB LED

1 PCS



LED (Red)

5 PCS



LED (Green)

5 PCS



LED(Blue)

5 PCS



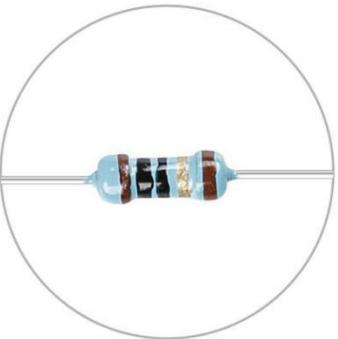
LED (Yellow)

5 PCS



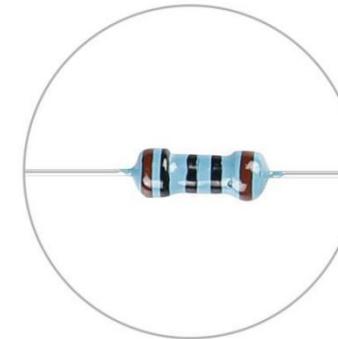
LED (White)

5 PCS



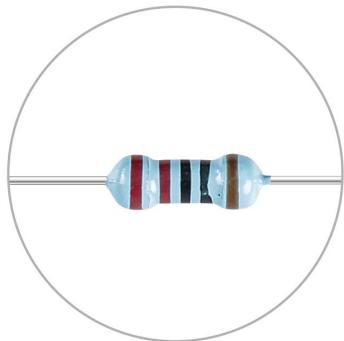
Resistor(10Ω)

10 PCS

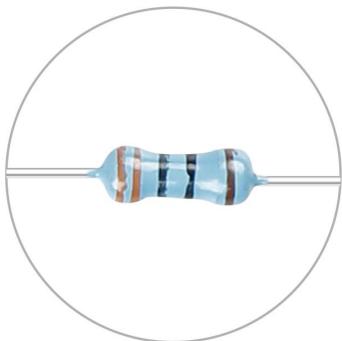


Resistor(100Ω)

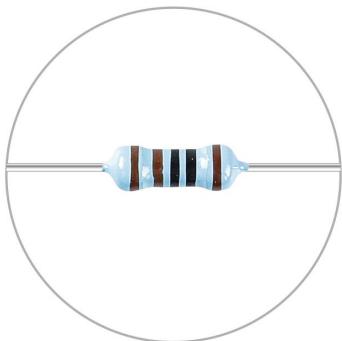
10 PCS



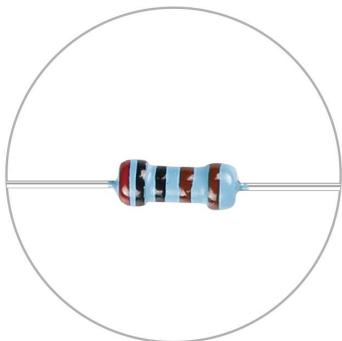
Resistor(220Ω)
30 PCS



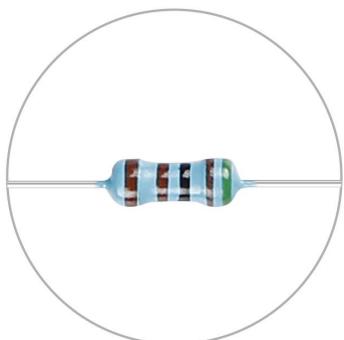
Resistor(330Ω)
10 PCS



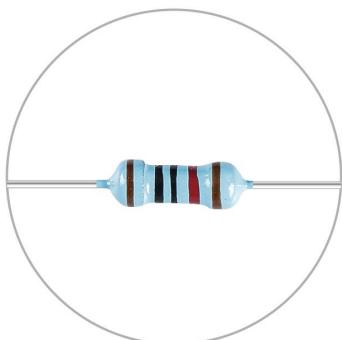
Resistor(1KΩ)
10 PCS



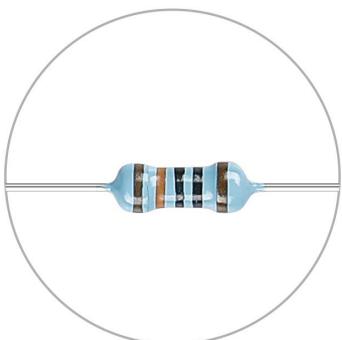
Resistor(2KΩ)
10 PCS



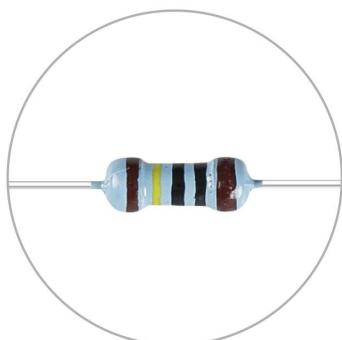
Resistor(5.1KΩ)
10 PCS



Resistor(10KΩ)
10 PCS



Resistor(100KΩ)
10 PCS



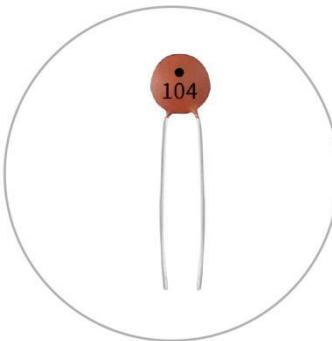
Resistor(1MΩ)
10 PCS



1N4007
1 PCS



22PF Capacitor
5 PCS



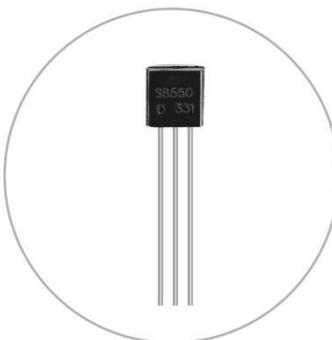
100nF Capacitor
5 PCS



Thermistor
1 PCS



Photoresistor
2 PCS



S8550 Transistor
1 PCS



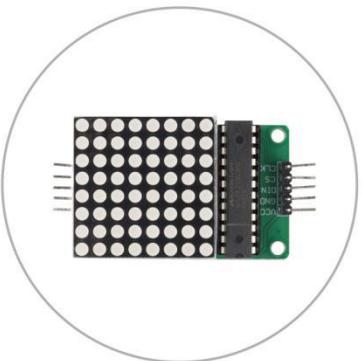
7-Segment Display
1 PCS



**4-Digit 7-
Segment Display**
1 PCS



1602 LCD
1 PCS



**MAX7219 LED
Matrix Module**
1 PCS



74HC595
1 PCS



ULN2003
1 PCS



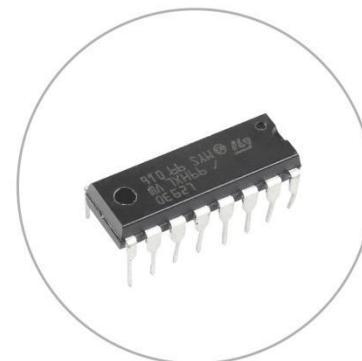
Passive Buzzer
1 PCS



Servo
1 PCS



Active buzzer
1 PCS



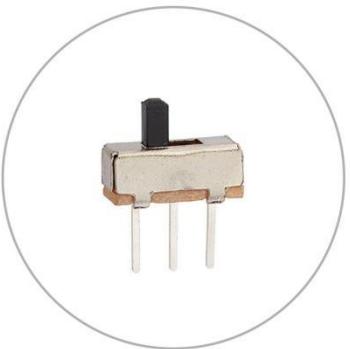
L293D
1 PCS



Stepper Motor
1 PCS



Motor
1 PCS



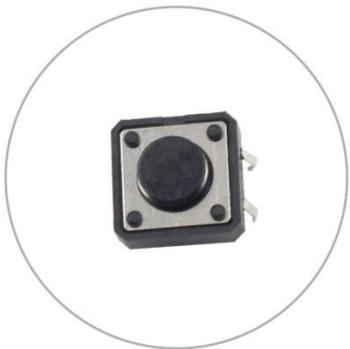
Slider Switch
1 PCS



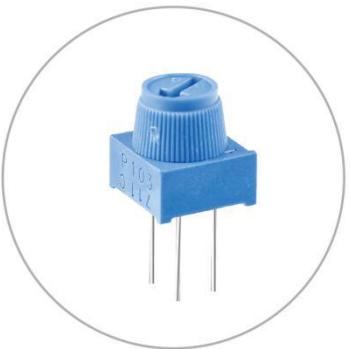
Relay Module
1 PCS



Power Supply Module
1 PCS



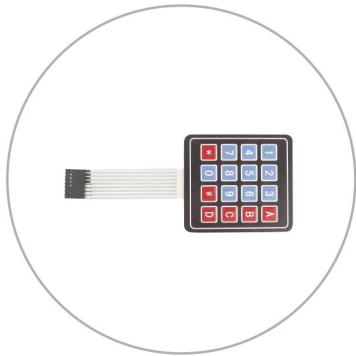
Button
5 PCS



Potentiometer
2 PCS



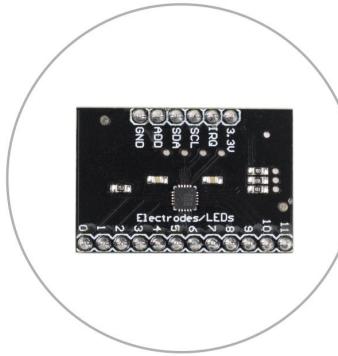
Joystick Module
1 PCS



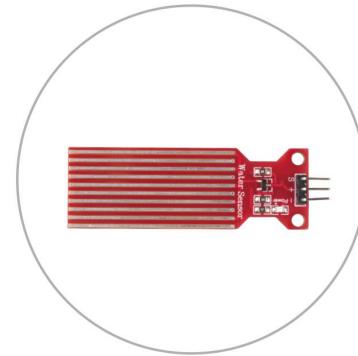
Keypad Module
1 PCS



Touch sensor
1 PCS



MPR121
1 PCS



Water Level Module
1 PCS



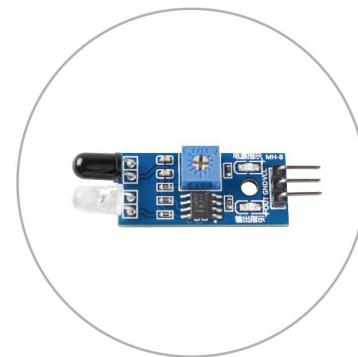
**Rotary
Encoder Module**
1 PCS



**Sound
Sensor Module**
1 PCS



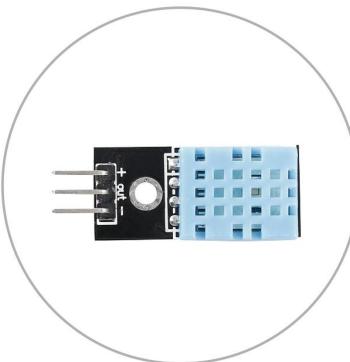
Tilt Switch
1 PCS



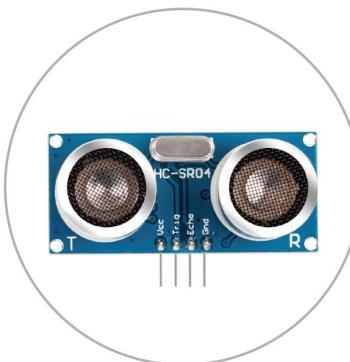
**Infrared Obstacle
Avoidance Module**
1 PCS



PIR Motion Sensor
1 PCS



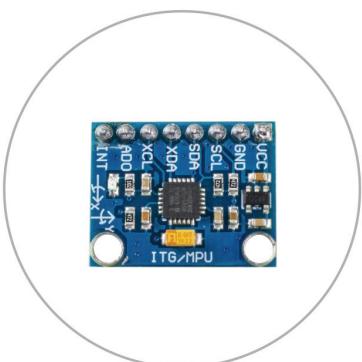
DHT11 Module
1 PCS



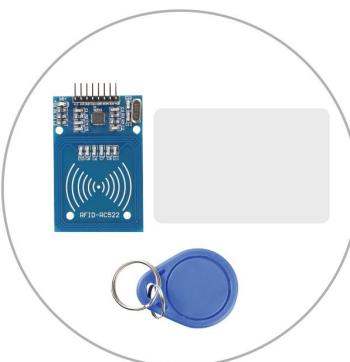
Ultrasonic Sensor
1 PCS



IR Receiver
1 PCS



MPU6050 Module
1 PCS



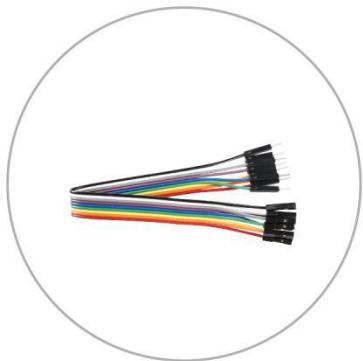
RFID
1 PCS



IR Remote Controller
1 PCS



USB Cable
1 PCS



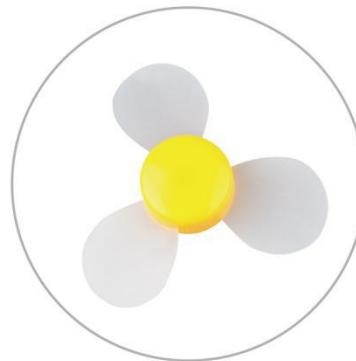
Dupont Wire
20 PCS



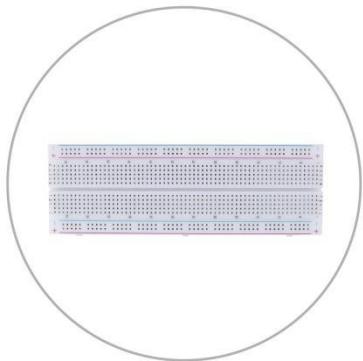
Jumper Wires
65 PCS



**Mega 2560
Development Board**
1 PCS



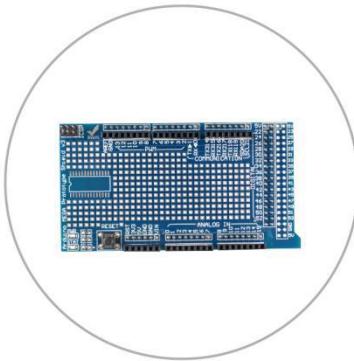
FAN
1 PCS



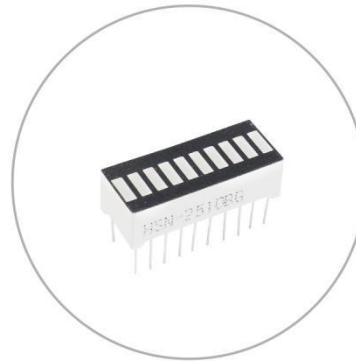
Breadboard
1 PCS



9VBattery Clip
1 PCS



**Mega2560
Protoshield**
1 PCS



LED Bar Graph
1 PCS

Introduction

This book, as a tutorial or a handbook, can be divided into four parts:

➤ Part 1: Basic

This section includes Arduino-related materials, including IDE installation and basic functions (read, write, serial communication, etc.).

➤ Part 2: Component

This part includes all components and modules of Vincent Kit, and explains their working principles and codes.

If you view the directory carefully, you will find that the use of the output device is introduced first, and then the use of the input device. The input device is also divided into switched input and sensor input.

➤ Part 3: Example

This section includes several sample programs written by Sunfounder that you can refer to when finishing some interesting projects of Vincent Kit.

➤ Part 4: Appendix

This section contains the extension function of Arduino IDE, and currently includes the method of adding libraries.

We look forward to your project and hope that you who have read this document will come to our forum to share the results when you finish the project.

Part 1: Basic

1.1 Arduino IDE

Description

Arduino is an open source platform with simple software and hardware. You can pick it up in short time even if you are a beginner. It provides an integrated development environment (IDE) for code compiling, compatible with multiple control boards. So you can just download the Arduino IDE, upload the sketches (i.e. the code files) to the board, and then you can see relative experimental phenomena. For more information, refer to <http://www.arduino.cc>.

Install Arduino IDE

Here are the installation steps on the windows system.

For other systems, please refer to: [Install Arduino IDE in different and FAQ.pdf](#)

The code in this kit is written based on Arduino, so you need to install the IDE first. Skip it if you have done this. Now go to arduino.cc and click SOFTWARE -> DOWNLOADs. on the page, check the software list on the right side.

HOME STORE SOFTWARE EDU RESOURCES COMMUNITY HELP

ONLINE TOOLS

DOWNLOADS

Download the Arduino IDE



ARDUINO 1.8.8

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

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

Windows Installer, for Windows XP and up
Windows ZIP file for non admin install

Windows app Requires Win 8.1 or 10
[Get](#)

Mac OS X 10.8 Mountain Lion or newer

Linux 32 bits
Linux 64 bits
Linux ARM

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

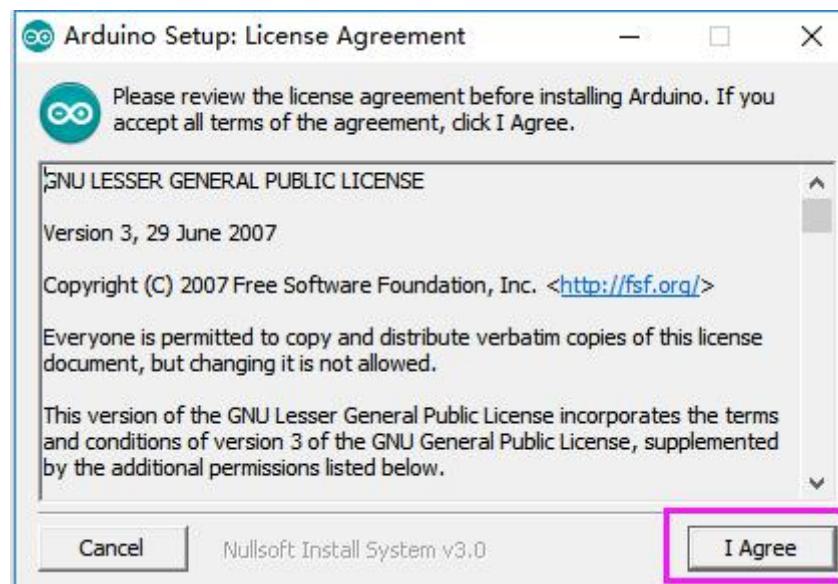
Find the one that suits your operation system and click to download. There are two versions of Arduino for Windows: Installer or ZIP file. You're recommended to download the former.

For Installer File

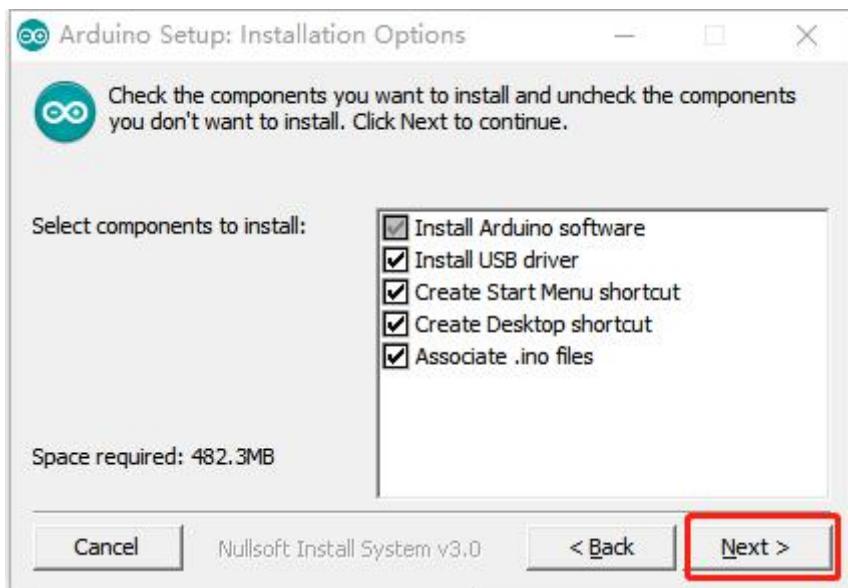
Step 1: Find the .exe file just downloaded.



Step 2: Double click the file and a window will pop up as below. Click **I Agree**.



Step 3: Click Next.

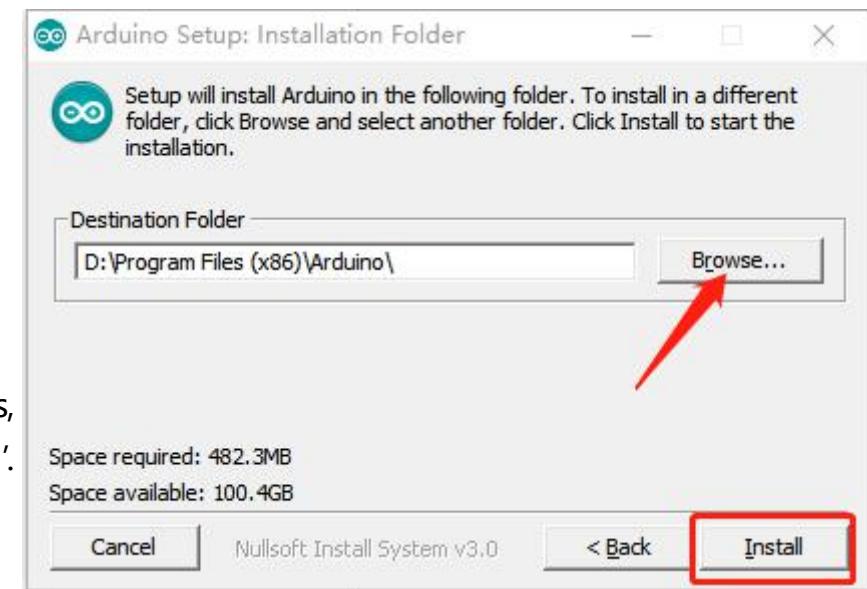


Step 5: Meanwhile, it will prompts install the needed drivers, please select the 'Always trust software from "Arduino LLC"'. After the installation is done, click **Close**.

Note:

The new IDE may prompt errors when you're compiling code under Windows XP. So if your computer is running on XP, you're suggested to install Arduino 1.0.5 or 1.0.6. Also you can upgrade your computer.

Step 4: Select the path to install. By default, it's set in the C disk. You can click **Browse** and choose other paths. Click **OK**. Then click **Install**.

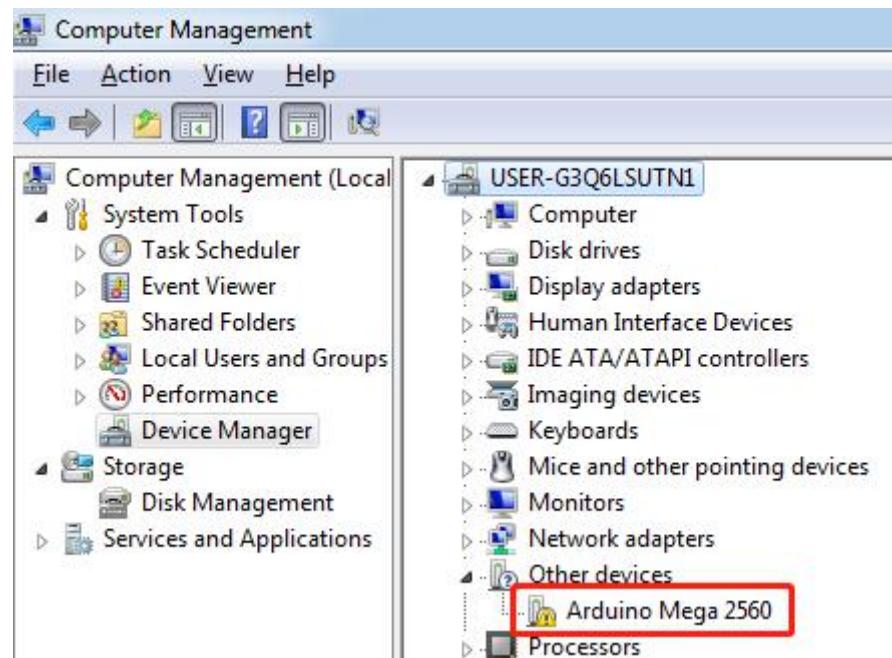


For ZIP File

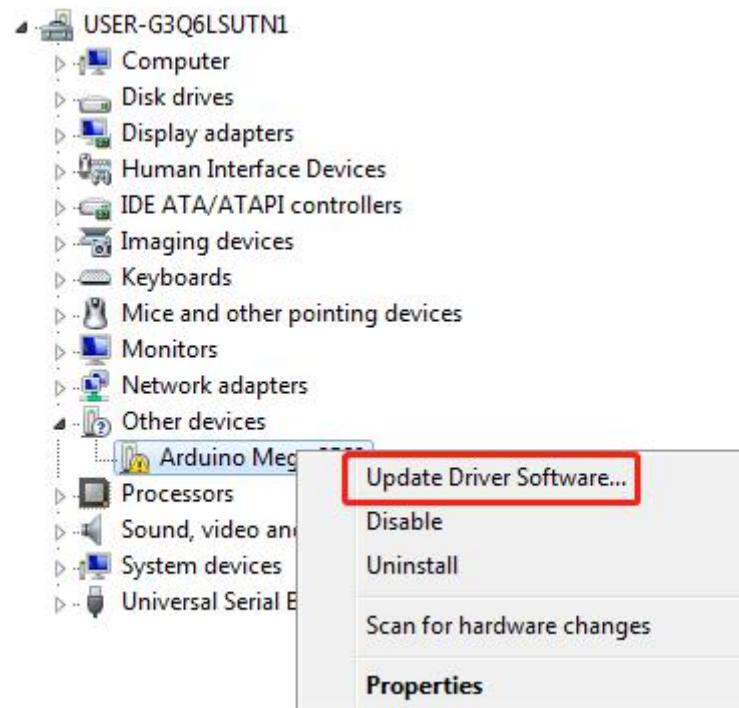
If you download the zip file before, when you connect the MCU to the computer, it may not be recognized. Then you need to install the driver manually. Take the following steps.

Step 1: Plug in the board to the computer with a 5V USB cable. After a while, a prompt message of failed installation will appear.

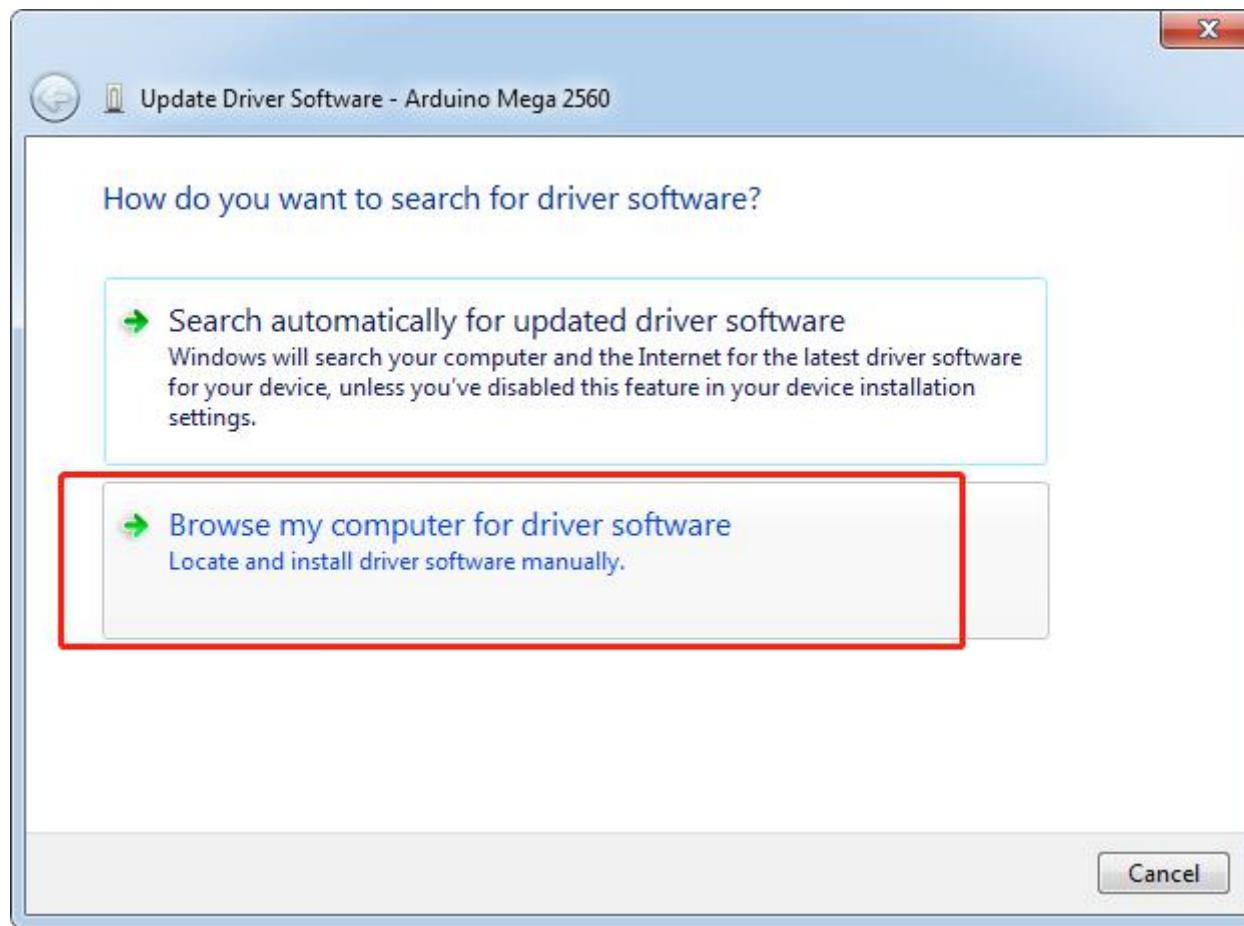
Step 2: Go to the Device Manager. You will find under other devices, Arduino Mega 2560 with an exclamation mark appear, which means the computer did not recognize the board.



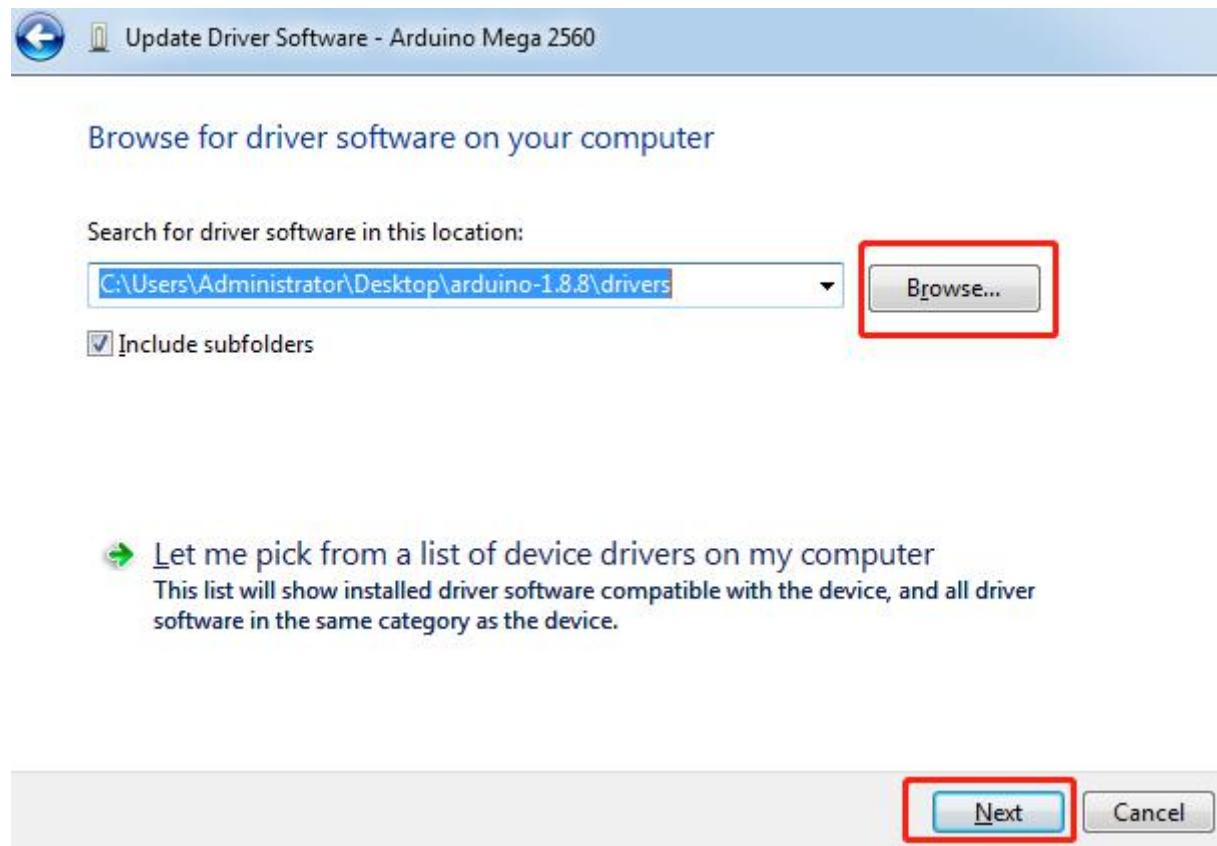
Step 3: Right click on **Arduino Mega 2560** and select **Update Driver Software**.



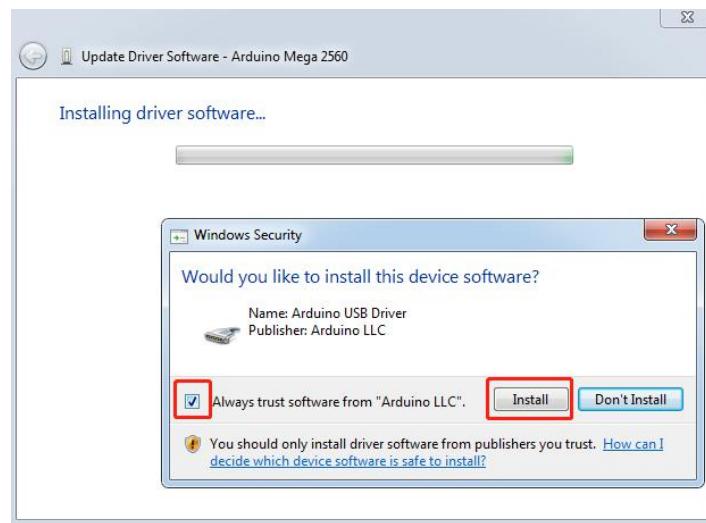
Step 4: Choose the second option, **Browse my computer for Driver software.**



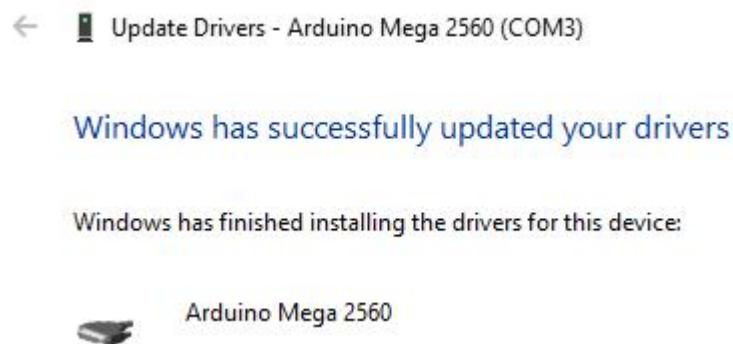
Step 5: A window pops up then. Click **Browse**. Then go to the folder where you just extracted the file. Go to the *drivers* folder and click **OK** -> **Next**.



Step 6: Select 'Always trust software from "Arduino LLC"' then click Install.



It may need a sec. Then the system prompts you the driver has been installed successfully. So the computer can recognize the board now. Click **Close**.



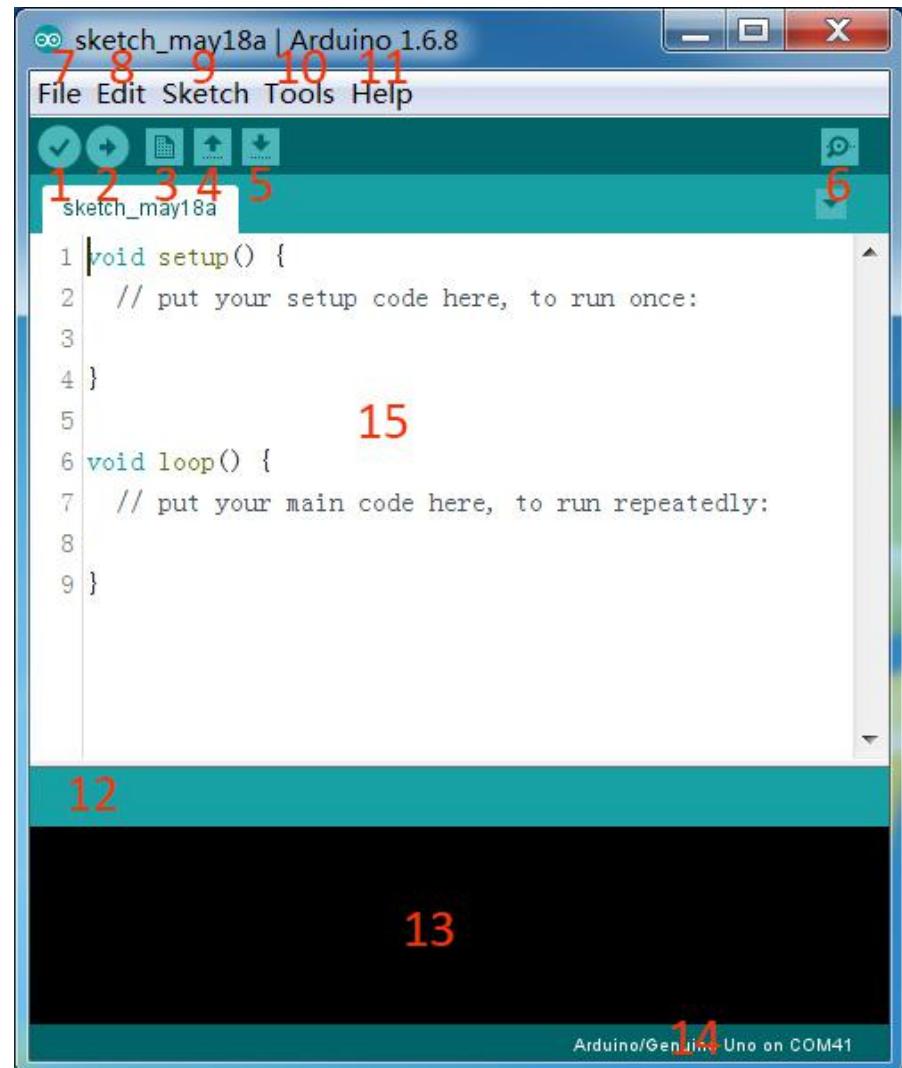
Open the Arduino Software (IDE)

Double-click the Arduino icon (arduino.exe) created by the installation process.



Then the Arduino IDE will appear. Let's check details of the software.

1. **Verify:** Compile your code. Any syntax problem will be prompted with errors.
2. **Upload:** Upload the code to your board. When you click the button, the RX and TX LEDs on the board will flicker fast and won't stop until the upload is done.
3. **New:** Create a new code editing window.
4. **Open:** Open an .ino sketch.
5. **Save:** Save the sketch.
6. **Serial Monitor:** Click the button and a window will appear. It receives the data sent from your control



board. It is very useful for debugging.

7. **File:** Click the menu and a drop-down list will appear, including file creating, opening, saving, closing, some parameter configuring, etc.
8. **Edit:** Click the menu. On the drop-down list, there are some editing operations like Cut, Copy, Paste, Find, and so on, with their corresponding shortcuts.
9. **Sketch:** Includes operations like Verify, Upload, Add files, etc. More important function is Include Library – where you can add libraries.
10. **Tool:** Includes some tools – the most frequently used Board (the board you use) and Port (the port your board is at). Every time you want to upload the code, you need to select or check them.
11. **Help:** If you're a beginner, you may check the options under the menu and get the help you need, including operations in IDE, introduction information, troubleshooting, code explanation, etc.
12. In this message area, no matter when you compile or upload, the summary message will always appear.
13. Detailed messages during compile and upload. For example, the file used lies in which path, the details of error prompts.
14. **Board and Port:** Here you can preview the board and port selected for code upload. You can select them again by **Tools -> Board / Port** if any is incorrect.
15. The editing area of the IDE. You can write code here.

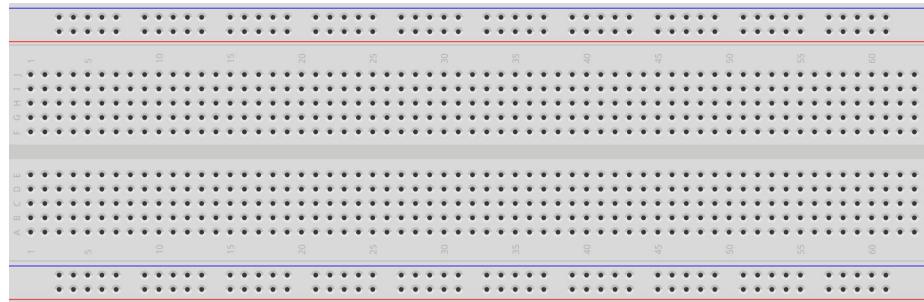
1.2 Digital Write

Overview

The `digitalWrite()` statement here is used to write high level or low level to pins and to let LED and active buzzer [work] or [stop]. In this lesson, we will take LED as an example to introduce the experiment phenomenon.

Components Required

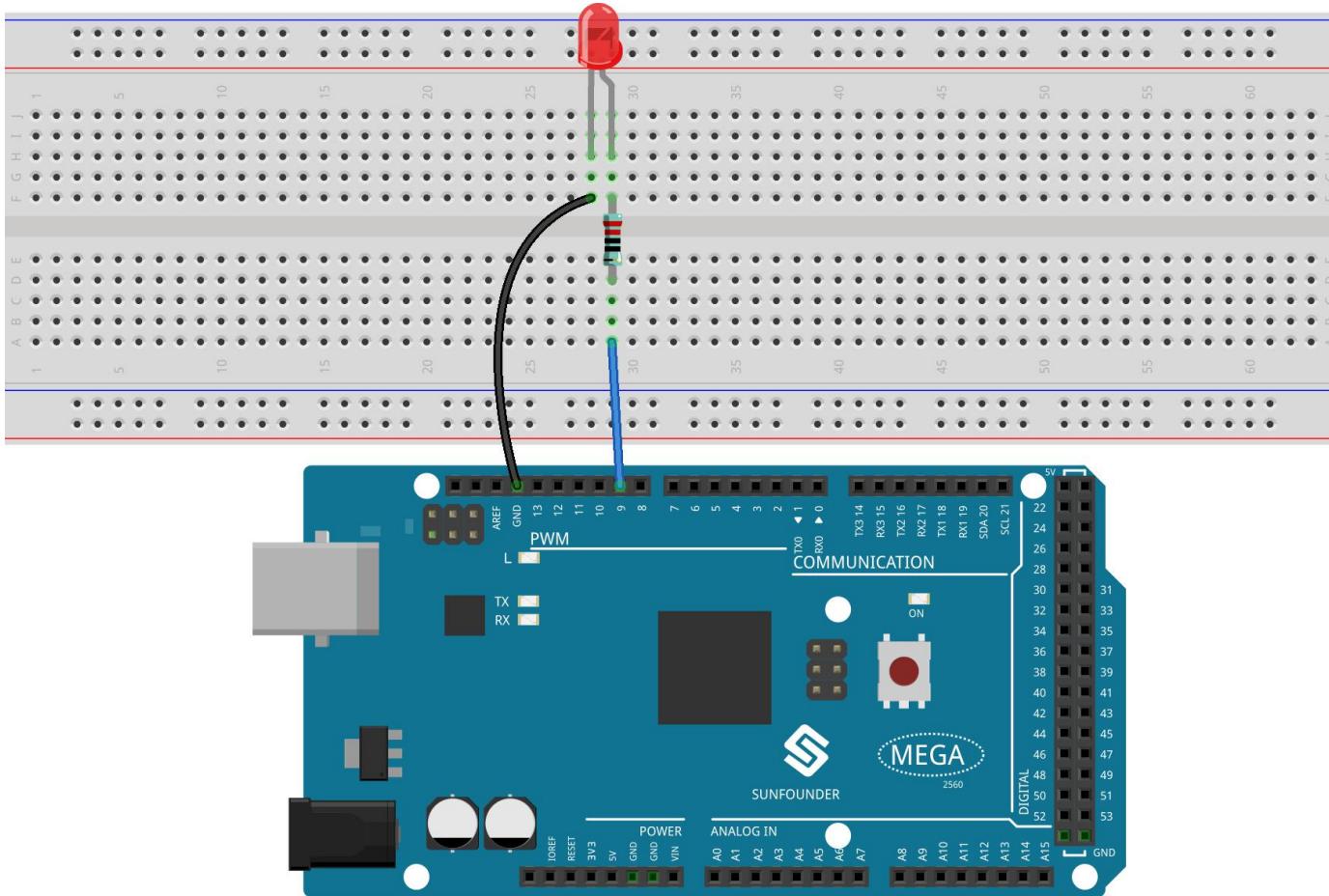
1 * LED	1 * 220ohm Resistor	Several Jumper Wires
		

1 * Mega 2560 Board	1 * Breadboard
	

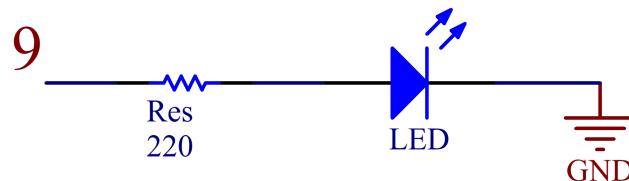
Note: Refer to Part 2 to check details of hardware.

Fritzing Circuit

In this example, we use digital pin 9 to drive the LED. Attach one side of the resistor to the digital pin 9 and the longer pin (anode) of the LED to the other side of the resistor. Connect the shorter pin (cathode) of the LED to GND.



Schematic Diagram



Code

After finishing the circuit connection, connect the Mega2560 board to the computer. Run the Arduino software IDE and type in the following codes.

NOTE: You can also open the 1.2digitalWrite.ino file in the path of sunfoudner_vincent_kit_for_arduino\code\1.2digitalWrite to use the codes.

```
const int ledPin = 9;  
void setup() {  
    pinMode(ledPin, OUTPUT);  
}  
void loop() {  
    digitalWrite(ledPin, HIGH);  
    delay(1000);  
    digitalWrite(ledPin, LOW);  
    delay(1000);  
}
```

Upload the codes to the Mega2560 board, and you can see the blinking of LED.

Code Analysis

Here, we connect the LED to the digital pin 9, so we need to declare an int variable called ledpin at the beginning of the program and assign a value of 9.

```
const int ledPin = 9;
```

Now, initialize the pin in the setup() function, where you need to initialize the pin to OUTPUT mode.

```
pinMode(ledPin, OUTPUT);
```

In loop(), digitalWrite() is used to provide 5V high level signal for ledpin, which will cause voltage difference between LED pins and light LED up.

```
digitalWrite(ledPin, HIGH);
```

If the level signal is changed to LOW, the ledPin's signal will be returned to 0 V to turn LED off.

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

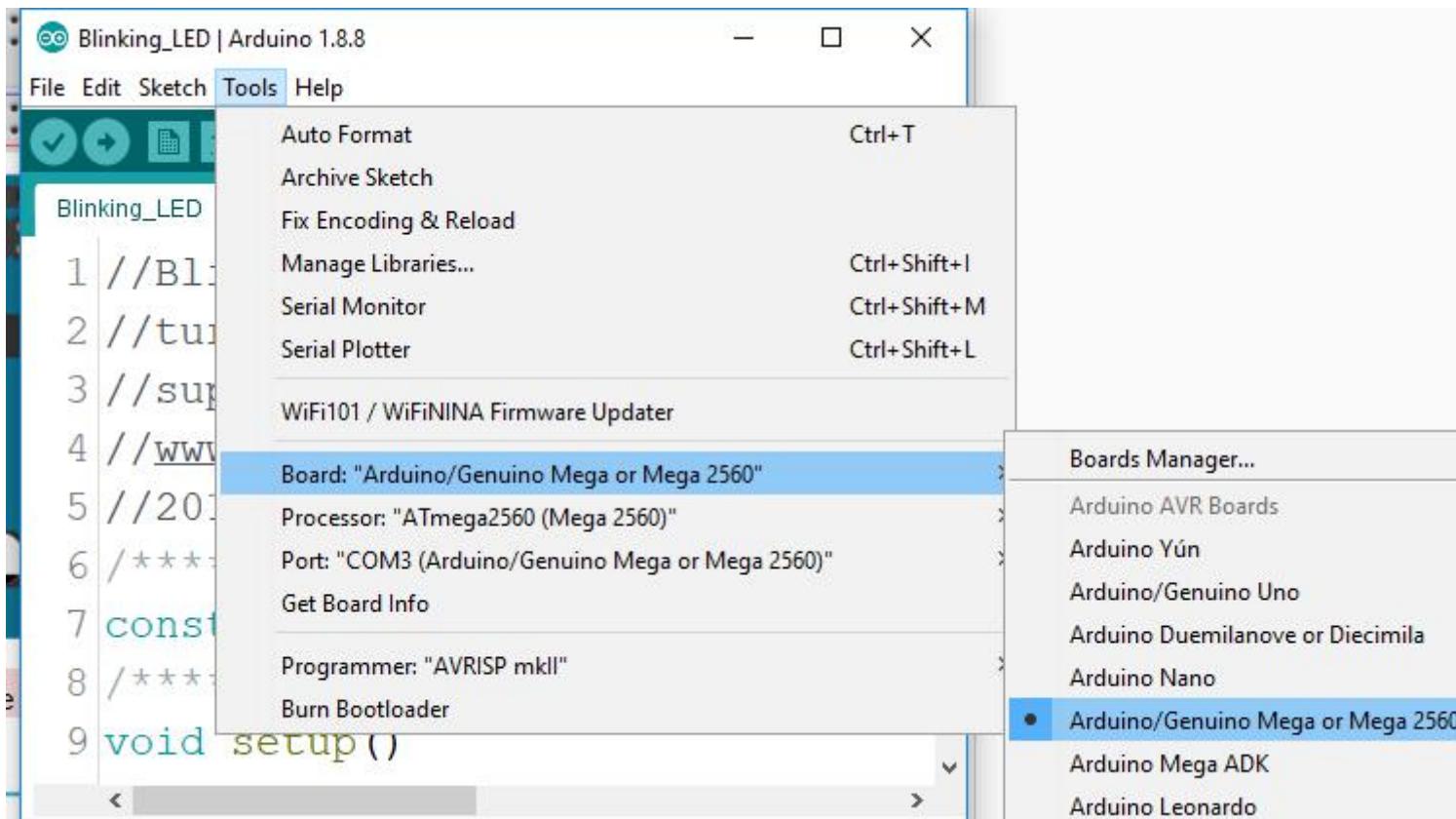
An interval between on and off is required to allow people to see the change, so we use a delay(1000) code to let the controller do nothing for 1000 ms.

```
delay(1000);
```

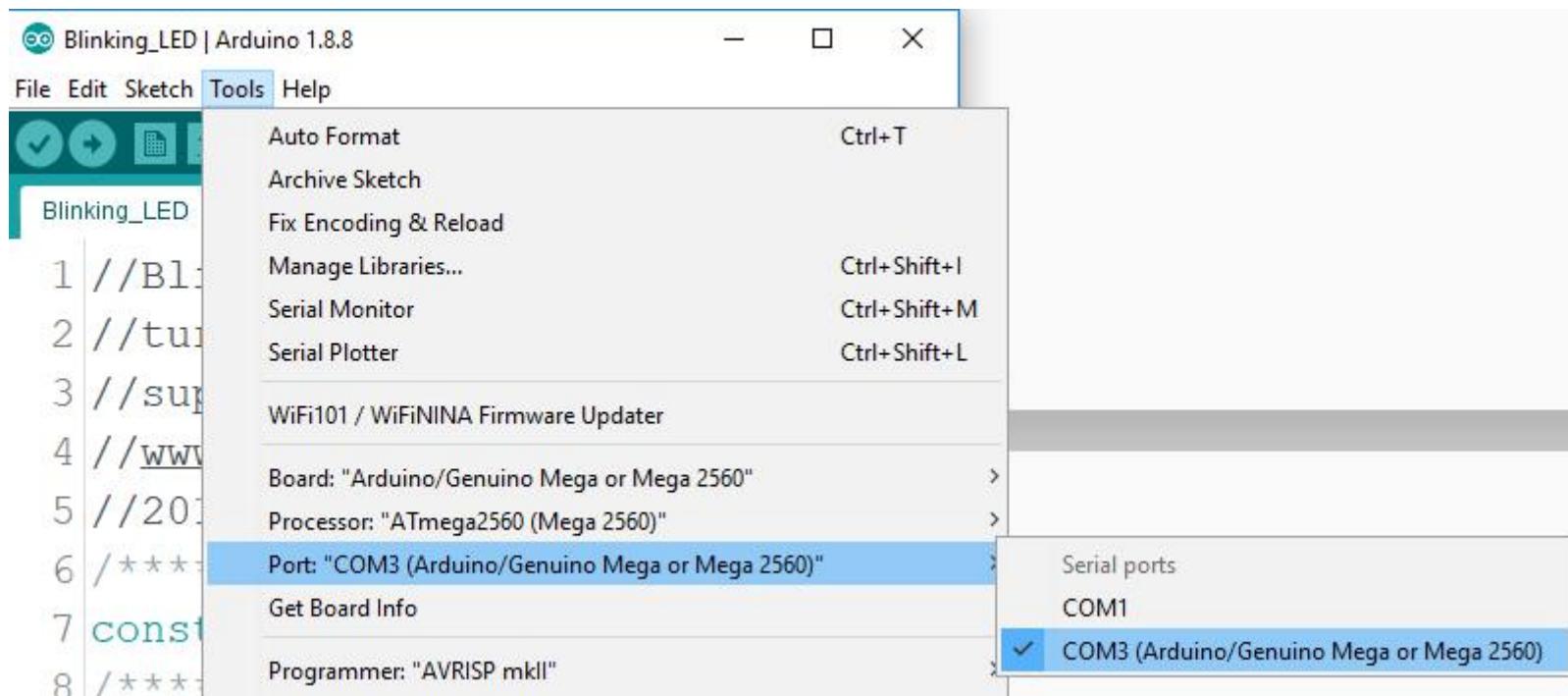
※ How to Upload the Sketch

Before you upload a written code to Arduino, you need first to select Board and Port.

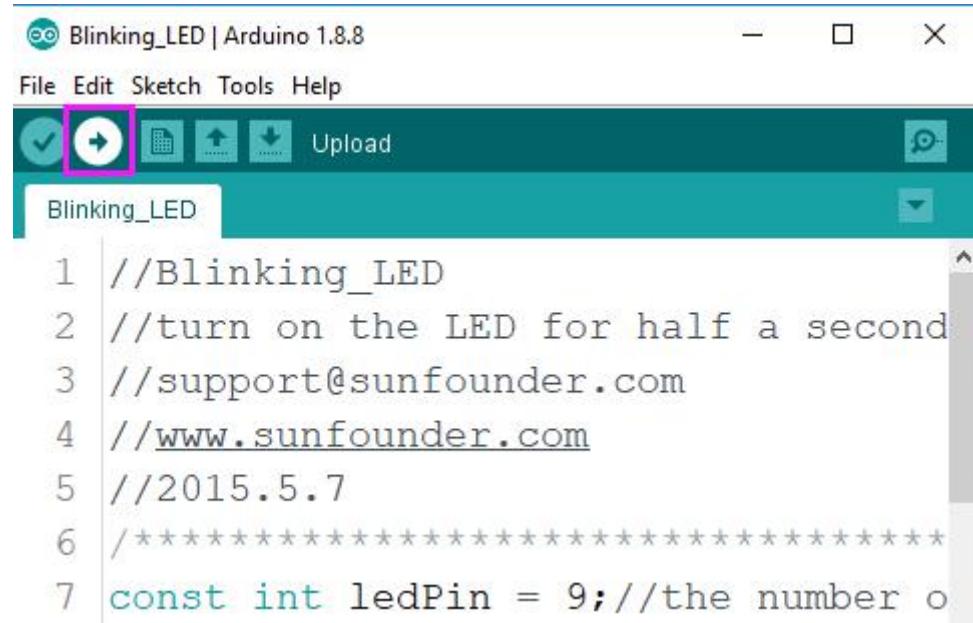
Click Tools ->Board and select Arduino/Genuino Mega or Mega 2560.



Then select **Tools -> Port**. Your port should be different from mine.



Step 4: Upload the sketch to the SunFounder Mega2560 board. Click the **Upload** icon to upload the code to the control board.



Blinking_LED | Arduino 1.8.8

File Edit Sketch Tools Help

Upload

Blinking_LED

```
1 //Blinking_LED
2 //turn on the LED for half a second
3 //support@sunfounder.com
4 //www.sunfounder.com
5 //2015.5.7
6 /*****
7 const int ledPin = 9;//the number o
```

If "Done uploading" appears at the bottom of the window, it means the sketch has been successfully uploaded.

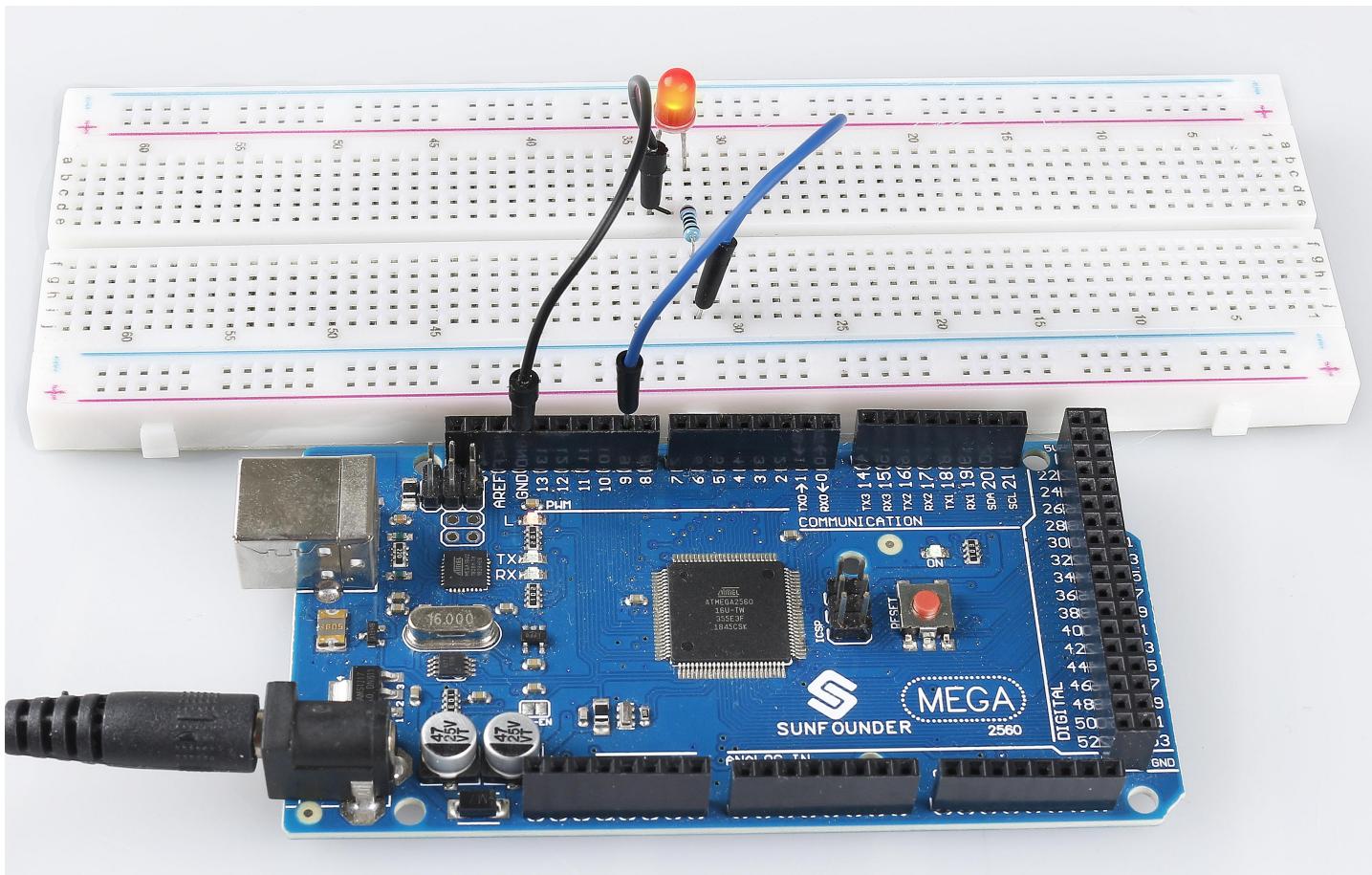


Done uploading.

avrduude done. Thank you.

1 Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM3

Phenomenon Picture

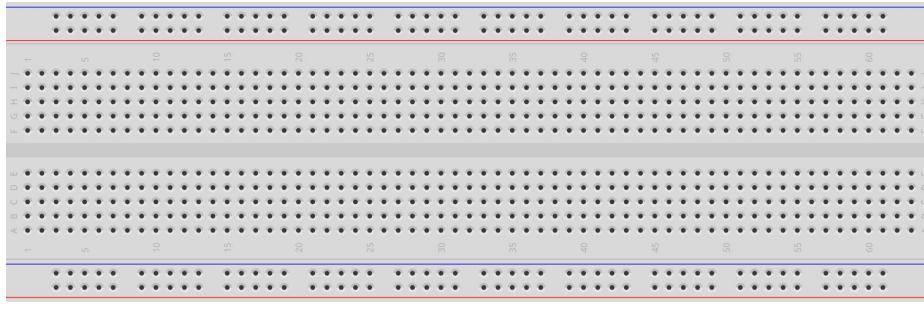


1.3 Analog Write

Overview

You can write the PWM wave to the pin by using `analogWrite()`. This method can be used to adjust the brightness of LED, change the color of RGB, or adjust the motor speed, etc. Here we will take LED as an example to get gradient brightness of LED.

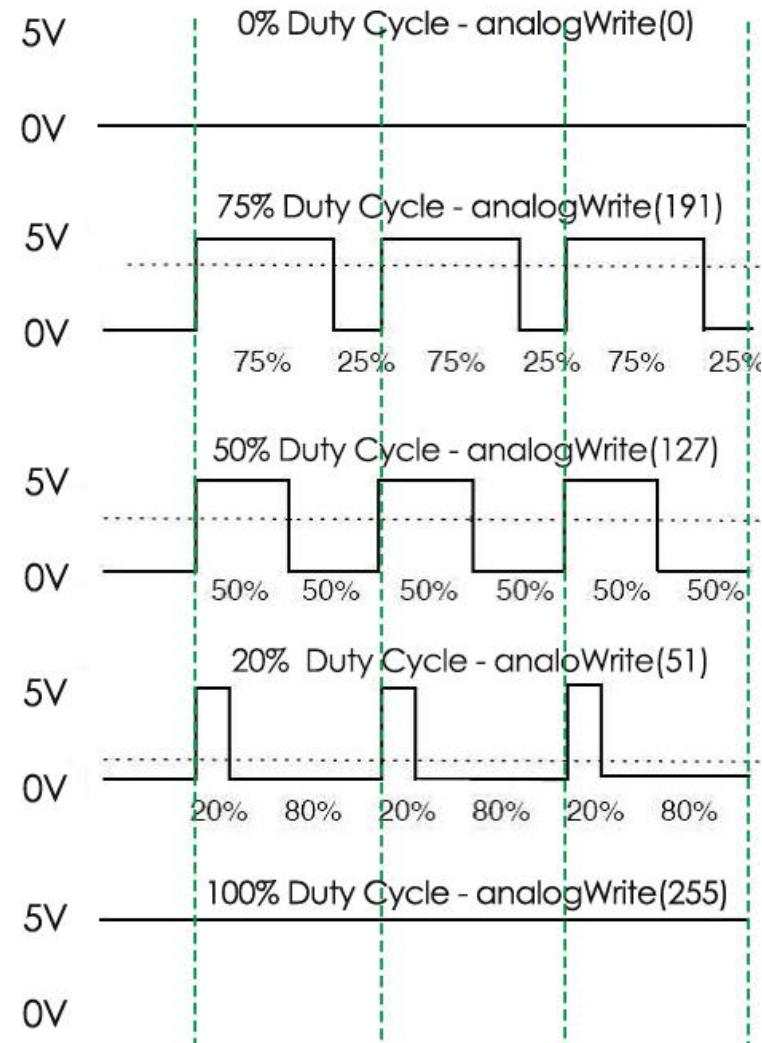
Components Required

1 * LED	1 * 220ohm Resistor	Several Jumper Wires
		
1 * Mega 2560 Board	1 * Breadboard	
		

※ Pulse Width Modulation

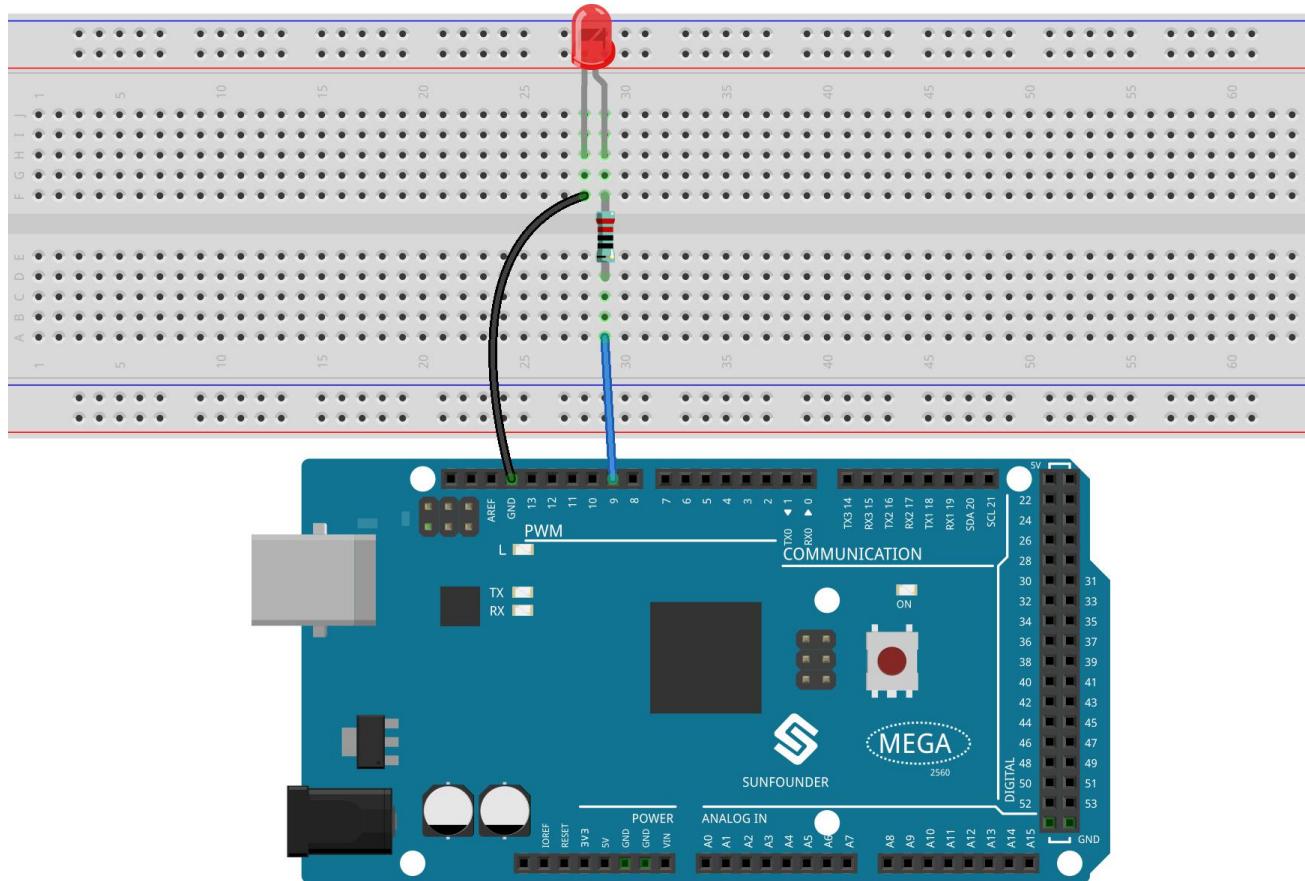
Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages in between full on (5 Volts) and off (0 Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off. The duration of "on time" is called the pulse width. To get varying analog values, you change, or modulate, that pulse width. If you repeat this on-off pattern fast enough with an LED for example, the result is as if the signal is a steady voltage between 0 and 5v controlling the brightness of the LED.

A call to `analogWrite()` is on a scale of 0 - 255, such that `analogWrite(255)` requests a 100% duty cycle (always on), and `analogWrite(127)` is a 50% duty cycle (on half the time) for example.

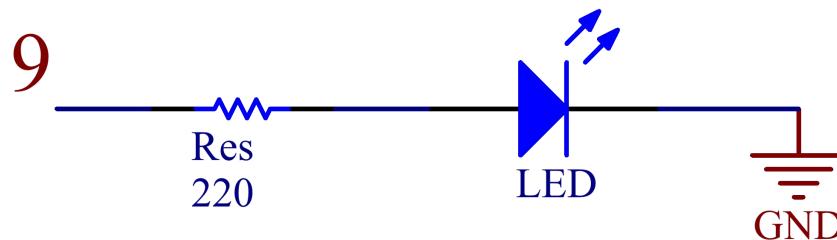


Fritzing Circuit

In this example, we use the PWM pin 9 to drive the LED. Connect one end of the resistor to pin 9. Connect the long pin (anode) of the LED to the other end of the resistor. Connect the short pin (negative, referred to as the cathode) of LED to GND. **NOTE: PWM pins of Mega2560 board are 2 - 13, 44 - 46.**



Schematic Diagram



Code

```
int ledPin = 9;  
void setup() {  
  
}  
void loop() {  
    for (int value = 0; value <= 255; value += 5) {  
        analogWrite(ledPin, value);  
        delay(30);  
    }  
}
```

After uploading the code to the Mega2560 board, you can see that the LED gradually brightens out and turns off gradually.

Code Analysis

Declare pin 9 as ledPin.

```
int ledPin = 9;
```

analogWrite() in loop() assigns ledPin an analog value (PWM wave) between 0 and 255 to change the brightness of LED.

```
analogWrite(ledPin, value);
```

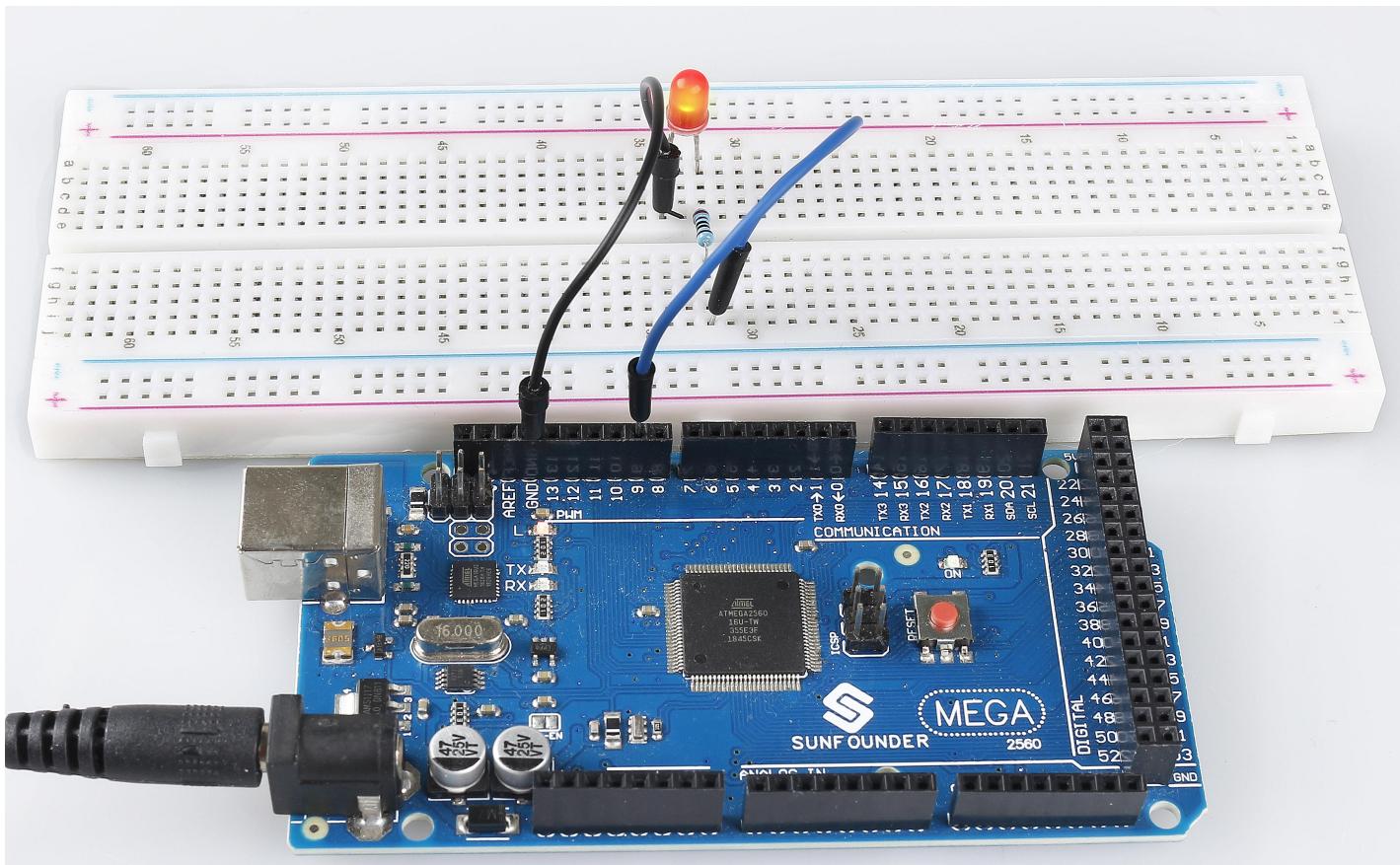
Using a for loop, the value of analogWrite() can be changed step by step between the minimum value (0) and the maximum value (255).

```
for (int value = 0; value <= 255; value += 5) {  
    analogWrite(ledPin, value);  
}
```

In order to see the experimental phenomenon clearly, a delay(30) needs to be added to the for cycle to control the brightness change time.

```
void loop() {  
    for (int value = 0; value <= 255; value += 5) {  
        analogWrite(ledPin, value);  
        delay(30);  
    }  
}
```

Phenomenon Picture



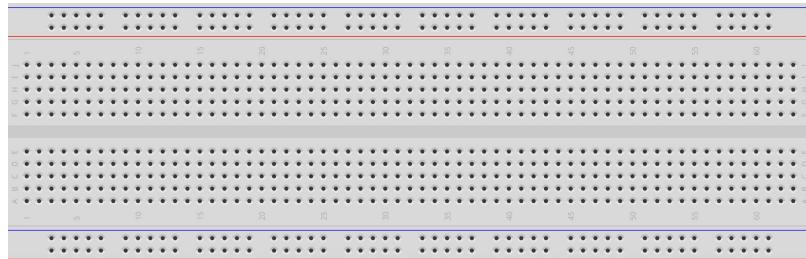
1.4 Digital Read

Overview

You can use the `digitalRead()` command to read the level status from a digital pin. The command is suitable for digital input elements such as Button, Touch sensor, infrared motion sensor, etc. This article will take Button as an example to read the level state.

This example also shows you how to monitor the state of a switch by using USB to establish serial communication between a control board and a computer.

Components Required

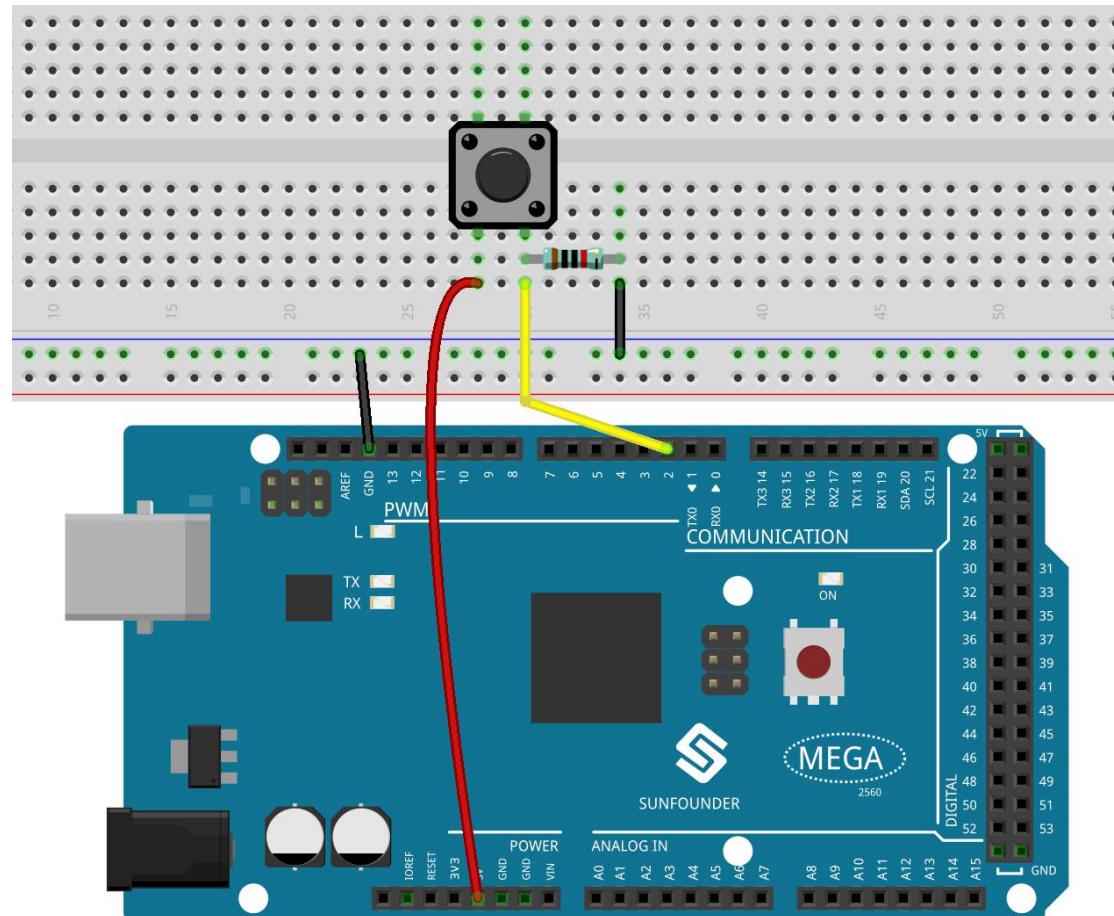
1 * Button		1 * 10k ohm resistor		Several Jumper Wires	
1 * Mega 2560 Board		1 * Breadboard			

Note: Refer to Part 2 to check details of hardware.

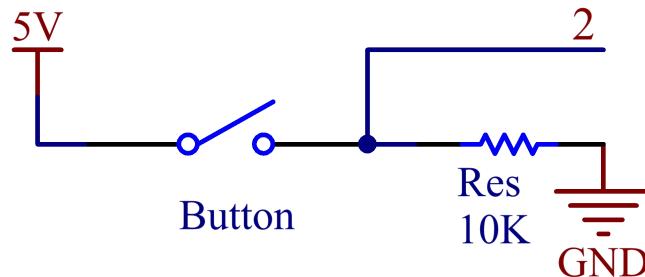
Fritzing Circuit

In this example, we read the signal of the button with the digital pin 2. When the button is not pressed, the digital pin 2 (through the drop-down resistor) is connected to ground to read the low level (0); when the button is pressed, the two pins are connected and when the pin is connected to the 5V power supply, the high level (1) is read.

NOTE : If you disconnect the digital I/O pin from everything, the LED may blink erratically. This is because the input is "floating" - that is, it doesn't have a solid connection to voltage or ground, and it will randomly return either HIGH or LOW. That's why you need a pull-down resistor in the circuit.



Schematic Diagram



Code

```
void setup() {
    Serial.begin(9600);
    pinMode(2, INPUT);
}

void loop() {
    int buttonState = digitalRead(2);
    Serial.println(buttonState);
    delay(1);
}
```

After uploading the code to the Mega2560 board, we can open the serial port monitor to see the reading value of the pin. When you press Button, the serial port monitor will display 「1」 and when Button is released, 「0」 will be displayed.

Code Analysis

Start the serial communication in `setup()` and set the data rate to 9600.

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

You also need to set the status of the digital pin 2 to INPUT to read the output status of Button.

```
pinMode(2, INPUT);
```

Use the `digitalRead()` statement in `loop()` to read the level state of the digital pin 2 and declare a variable to store the state.

```
int buttonState = digitalRead(2);
```

Print the value stored by the variable on the serial port monitor.

```
Serial.println(buttonState);
```

Use `delay()` statements to make printing results easy to observe.

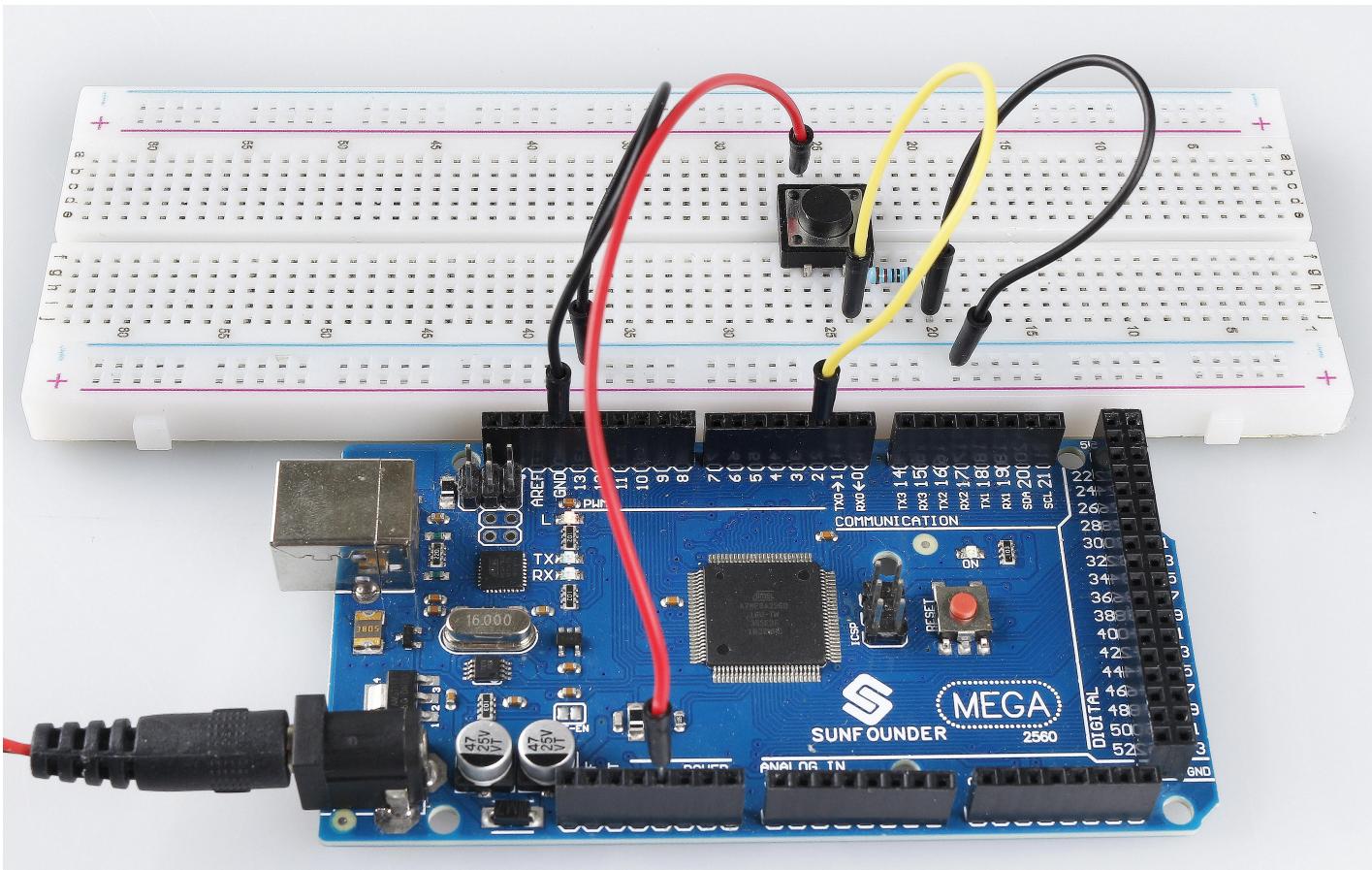
```
delay(1);
```

※ How to turn on Serial Port Monitor



Click the magnifier icon at the top right of the Arduino IDE programming window to open the **Serial Monitor**.

Phenomenon Picture



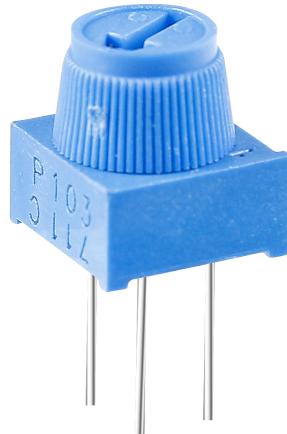
1.5 Analog Read

Overview

You can use the `analogRead()` command to read analog input from the physical world through an analog pin, which is suitable for analog input elements such as potentiometers, photoresistance, water level detection sensors, and so on. This article will take the potentiometer as an example to read the analog value of its output.

Components Required

1 * potentiometer



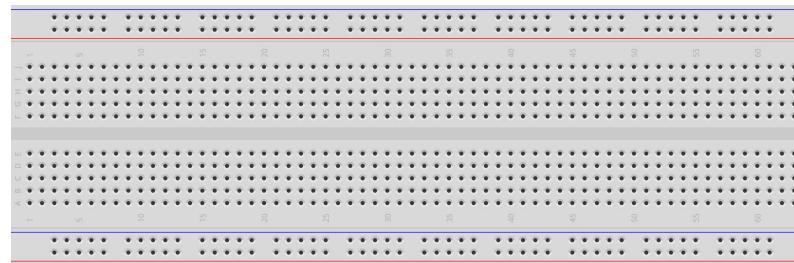
Several Jumper Wires



1 * Mega 2560 Board



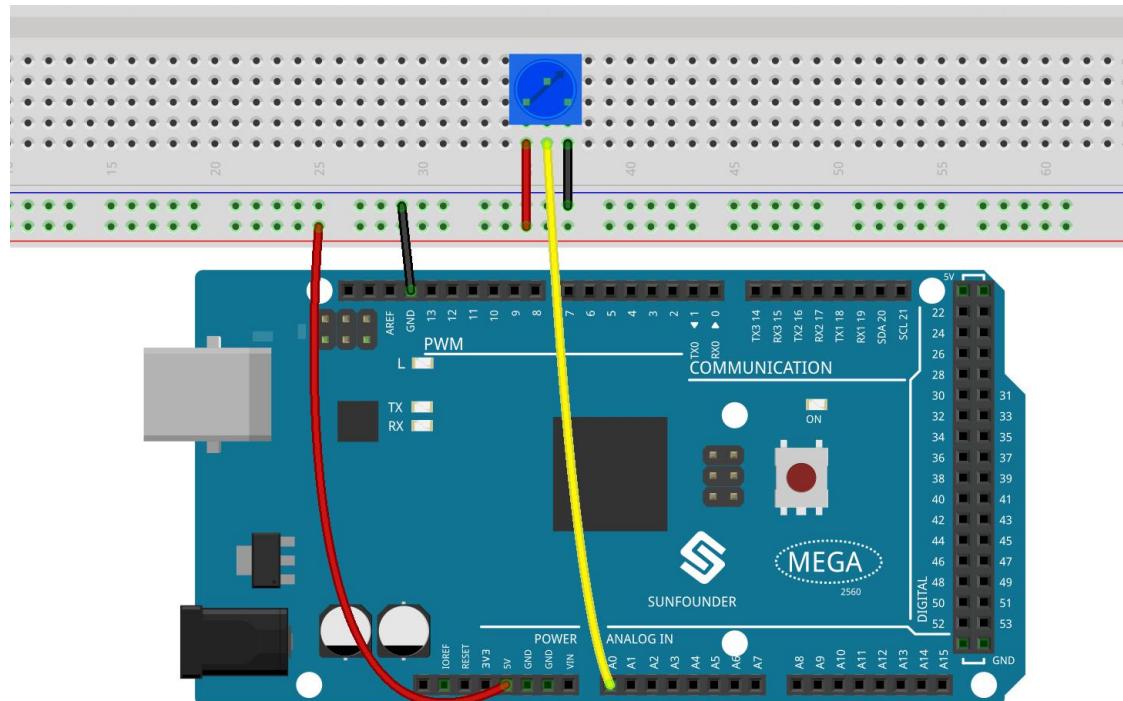
1 * Breadboard



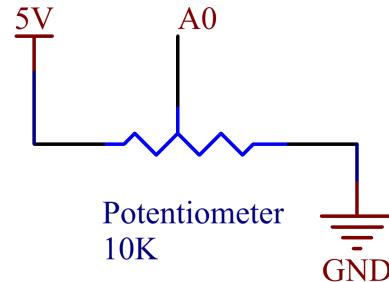
Fritzing Circuit

In this example, we use the analog pin (A0) to read the value of the potentiometer. Connect the pins at both ends of the potentiometer to 5V and GND respectively. Connect the middle pin to A0.

The voltage of the middle pin will be output to the Mega2560 board as an analog value. By rotating the axis of the potentiometer, you can change the voltage on the middle pin, thereby changing the analog value of the pin obtained by A0.



Schematic Diagram



Code

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

void loop() {
    int sensorValue = analogRead(A0);
    Serial.println(sensorValue);
    delay(1);
}
```

After the code is uploaded to the Mega2560 board, you can open the serial port monitor to view the reading value of the pin. When the shaft of the potentiometer is turned, the serial port monitor will print the value that changes between "0" and "1023".

Code Analysis

To enable Arduino IDE to print the value transmitted from electronic component to the Mega2560 board, you need to start serial communication in `setup()` and set the data rate to 9600.

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

Use the `analogRead()` statement in `loop()` to read the level state acquired by analog pin A0 and declare a variable to store the level state.

```
int sensorValue = analogRead(A0);
```

Print the value stored in the variable on the serial monitor.

```
Serial.println(sensorValue);
```

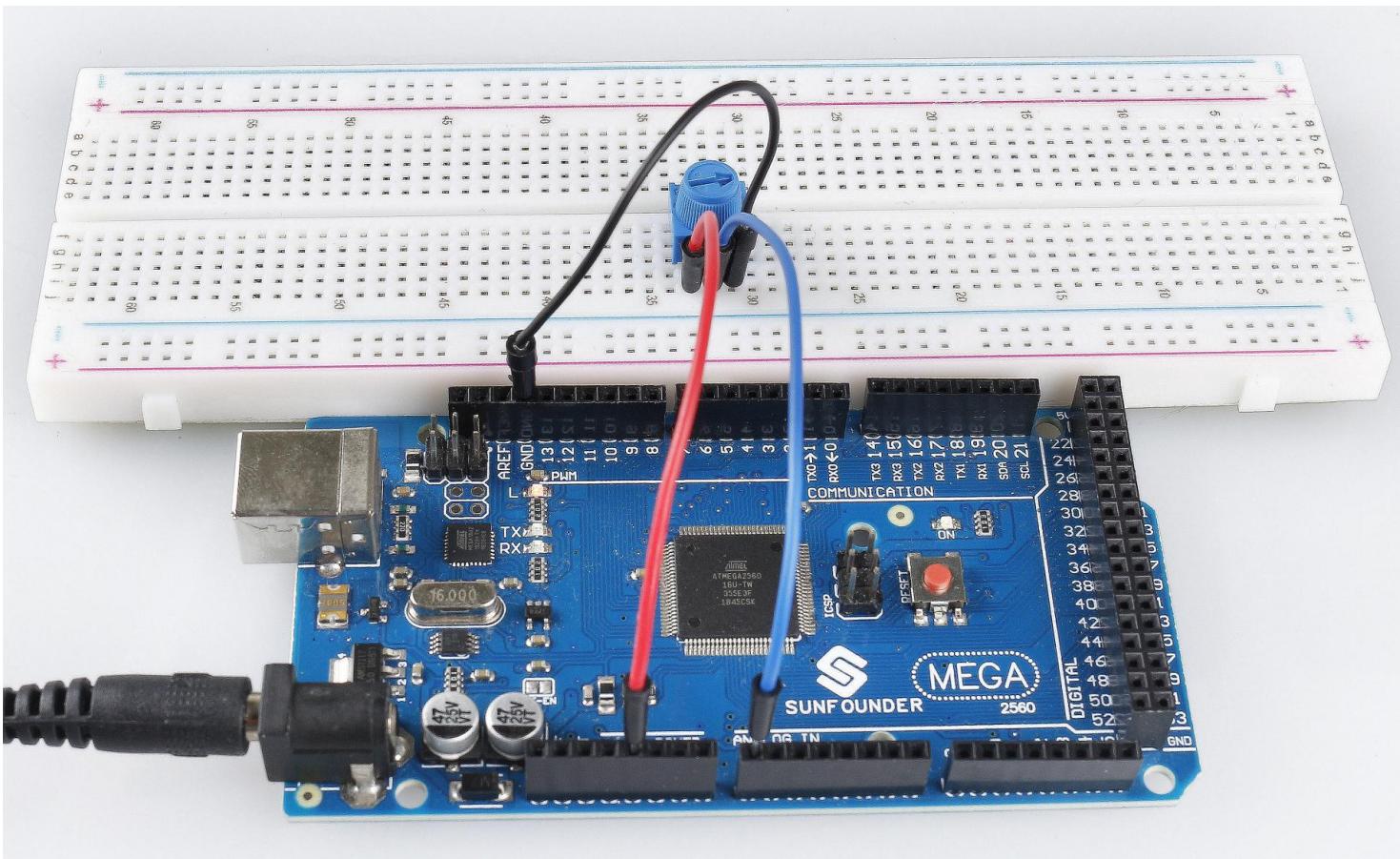
Use `delay()` statements to make printing results easy to observe.

```
delay(1);
```

※ Analog-to-Digital Converter

The Arduino have a circuit inside called an analog-to-digital converter or ADC that reads this changing voltage and converts it to a number between 0 and 1023. When the shaft is turned all the way in one direction, there are 0 volts going to the pin, and the input value is 0. When the shaft is turned all the way in the opposite direction, there are 5 volts going to the pin and the input value is 1023. In between, `analogRead()` returns a number between 0 and 1023 that is proportional to the amount of voltage being applied to the pin.

Phenomenon Picture

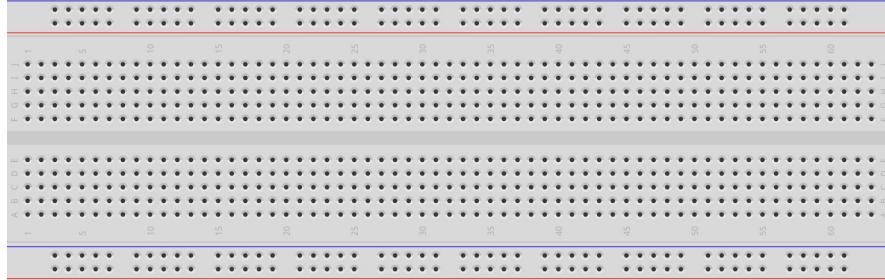


1.6 Digital Input Control Output

Overview

With the understanding of digitalWrite() and digitalRead(), we can build a complete I / O system to control the output device by obtaining the data from the input device. We can use this method to enable digital input components such as Button, Touch sensor, Infrared motion sensor to control digital output devices such as LED, active buzzer. This lesson will take Button and LED as examples to realize button control LED with the condition (if-else).

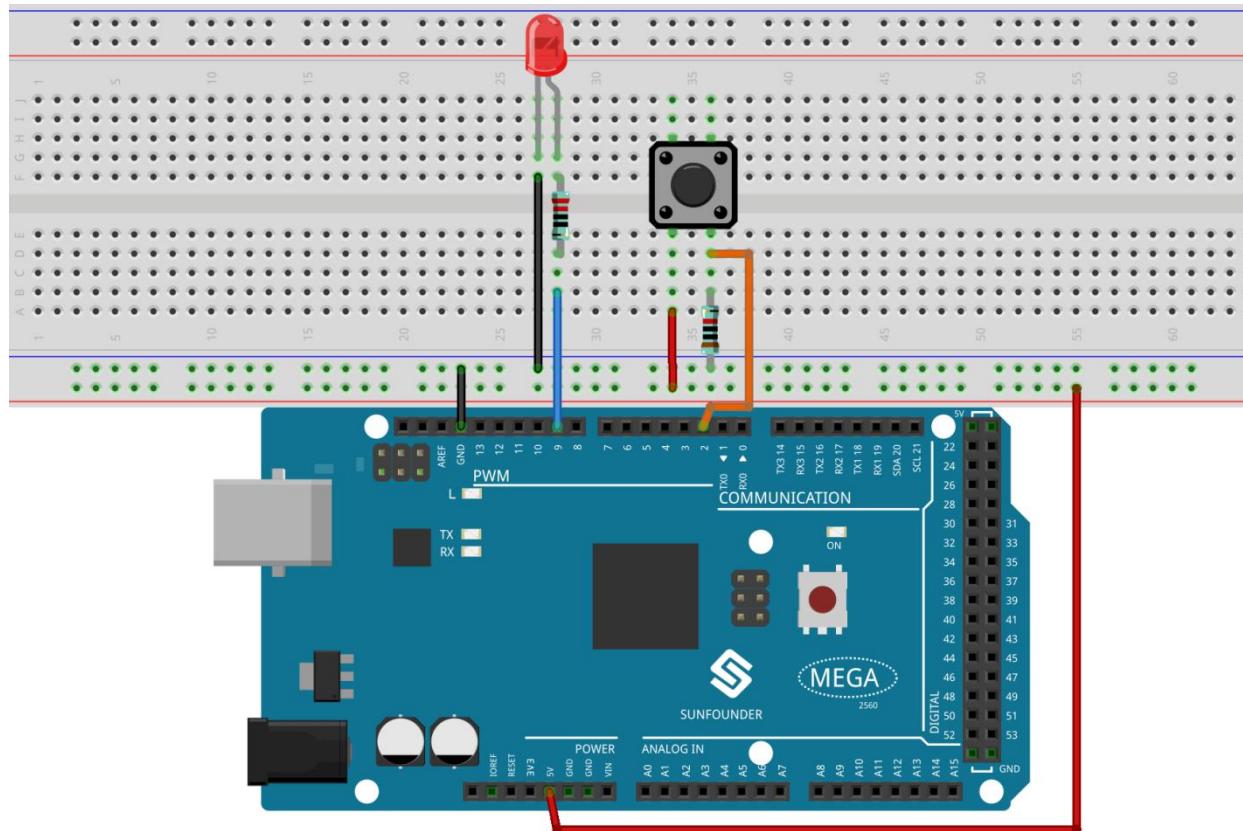
Components Required

1 * Button	1 * LED	1 * 10k ohm resistor
		
Jumper wire		1 * 220ohm Resistor
		
1 * Mega 2560 Board	1 * Breadboard	
		

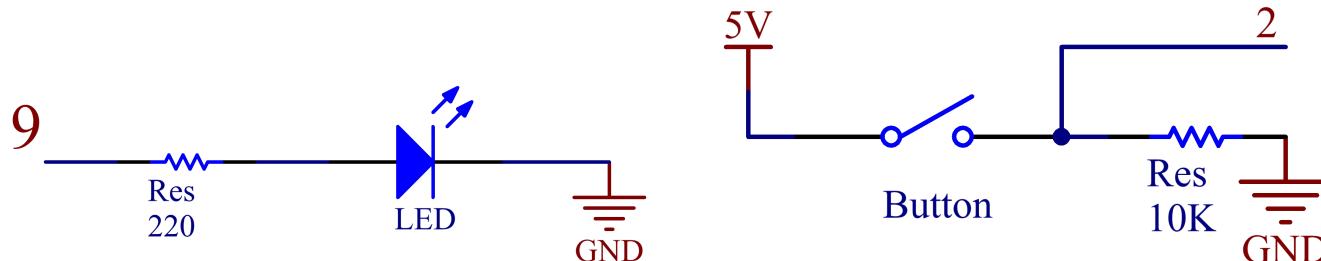
Note: Refer to Part 2 to check details of hardware.

Fritzing Circuit

In this example, we use pin 9 to drive LED. Use digital pin 2 to read the signal of Button. When the button is pressed, the LED lights up.



Schematic Diagram



Code

```
const int buttonPin = 2;  
const int ledPin = 9;  
int buttonState = 0;  
  
void setup() {  
    pinMode(ledPin, OUTPUT);  
    pinMode(buttonPin, INPUT);  
}  
void loop() {  
    buttonState = digitalRead(buttonPin);  
    if (buttonState == HIGH) {  
        digitalWrite(ledPin, HIGH);  
    } else {
```

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

After uploading the code to the Mega2560 board, you can hold down Button to lighten the LED.

Code Analysis

Declare the pins of LED and Button and declare a variable to store the state of button.

```
const int buttonPin = 2;  
const int ledPin = 9;  
int buttonState = 0;
```

Initialize the pin mode in setup().

```
pinMode(ledPin, OUTPUT);  
pinMode(buttonPin, INPUT);
```

Read the status of the Button in loop() and assign it to the variable buttonState.

```
buttonState = digitalRead(buttonPin);
```

Use if condition to judge: if you get high level from a button, light up the LED.

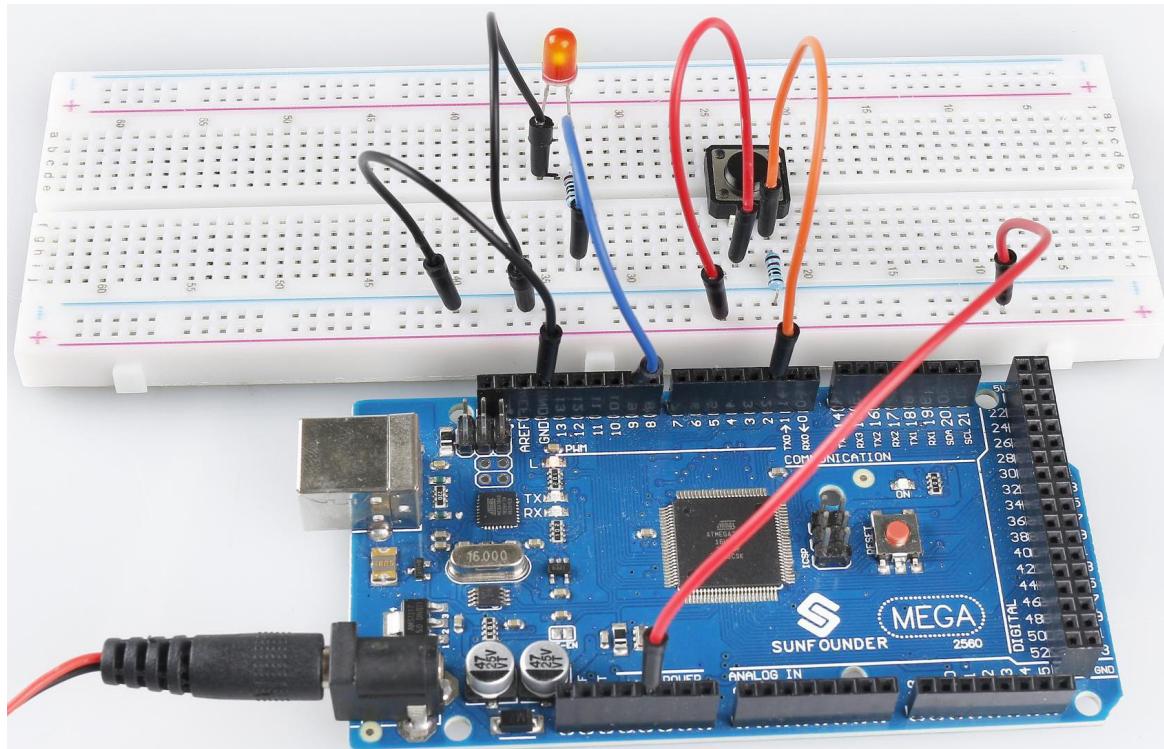
```
if (buttonState == HIGH) {  
    digitalWrite(ledPin, HIGH);
```

}

Otherwise, turn off the LED.

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

Phenomenon Picture

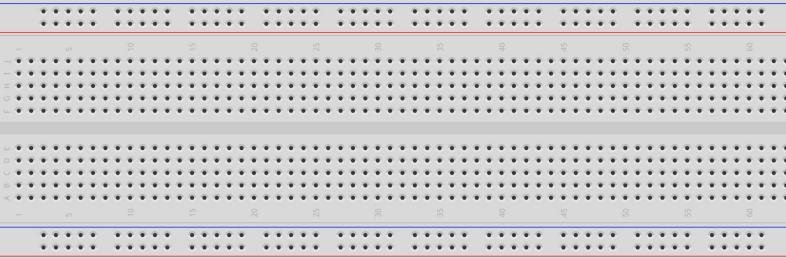


1.7 Analog Input Control Output

Overview

You can install an I/O system by using an analog input/ output device. For example, you can use potentiometer, photoresistor, water level sensor, etc., to control the brightness of LED, the speed of motor, and the like. In this lesson, potentiometer and LED are taken as examples to change the brightness of the LED when the potentiometer is turning.

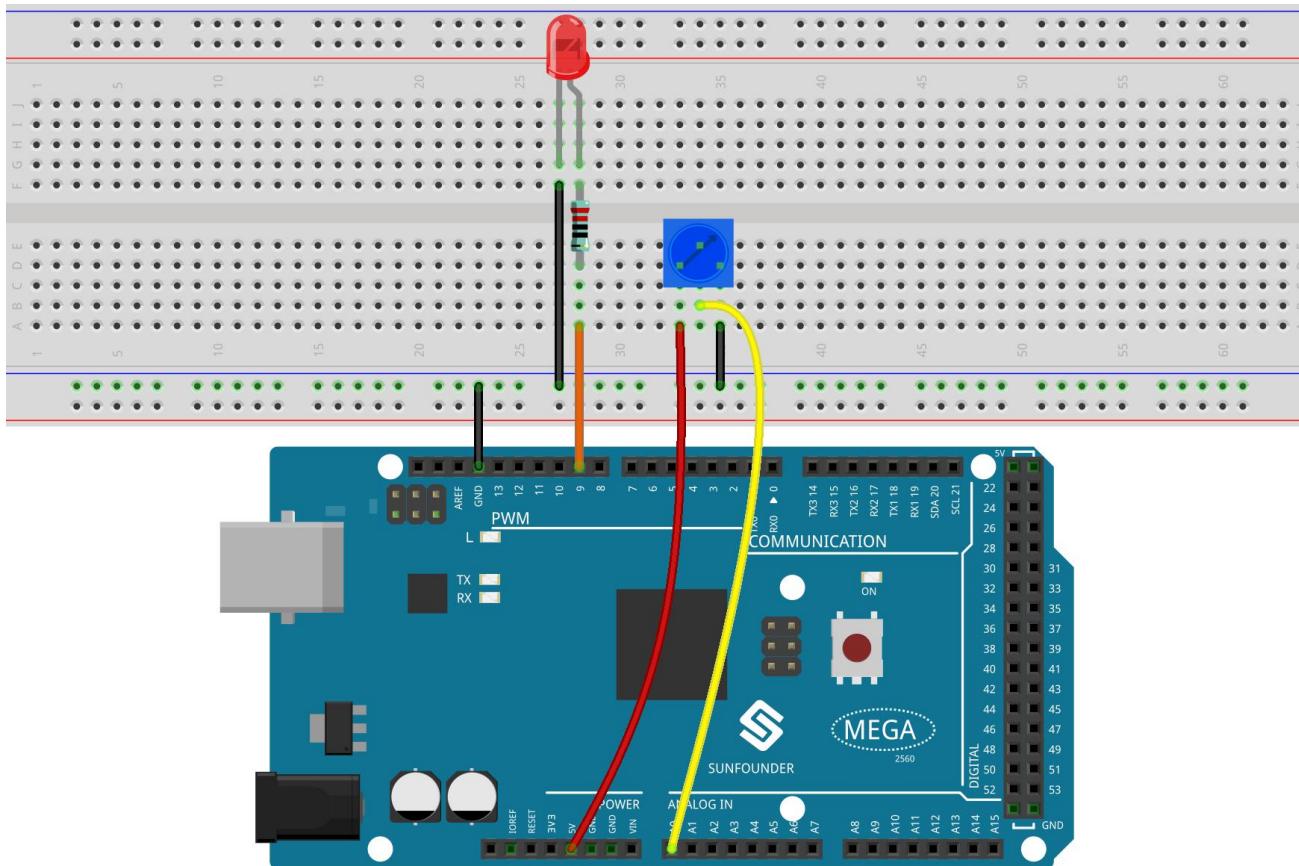
Components Required

1 * Potentiometer	1 * LED	1 * 220 ohm resistor
		
Several Jumper Wires		
1 * Mega 2560 Board		1 * Breadboard
		

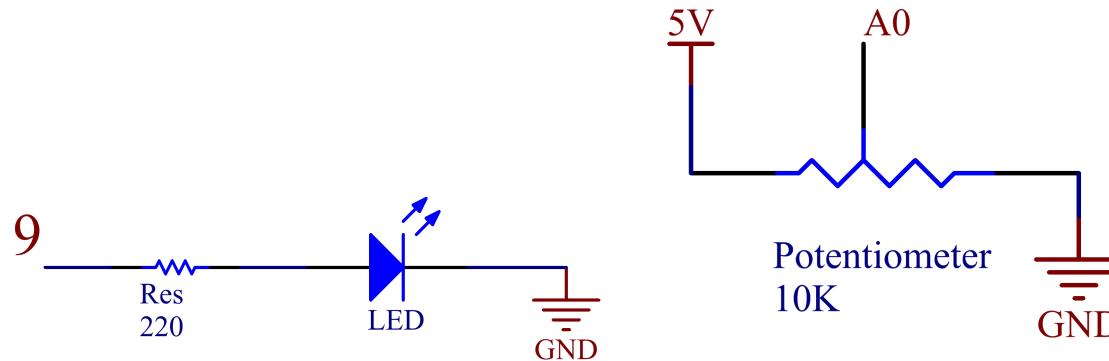
Note: Refer to Part 2 to check details of hardware.

Fritzing Circuit

In this lesson, we use PWM pin 9 to drive LED. The analog pin (A0) is used to read the value of potentiometer. After uploading the code, you'll notice that the brightness of the LED changes as the potentiometer rotates.



Schematic Diagram



Code

```
const int sensorPin = A0;  
const int ledPin = 9;  
void setup() {  
    pinMode(ledPin,OUTPUT);  
}  
  
void loop() {  
    int sensorValue=analogRead(sensorPin);  
    int brightness = map(sensorValue,0,1024,0,255);  
    analogWrite(ledPin,brightness);
```

}

When the codes are uploaded to the Mega2560 board, you can see that the brightness of LED is changing with the turning of the knob of potentiometer.

Code Analysis

Declare the pins of LED and Button.

```
const int sensorPin = A0;  
const int ledPin = 9;
```

In setup(), set the mode of ledPin to OUTPUT.

```
pinMode(ledPin,OUTPUT);
```

Read the readings of potentiometer in loop().

```
int sensorValue=analogRead(sensorPin);
```

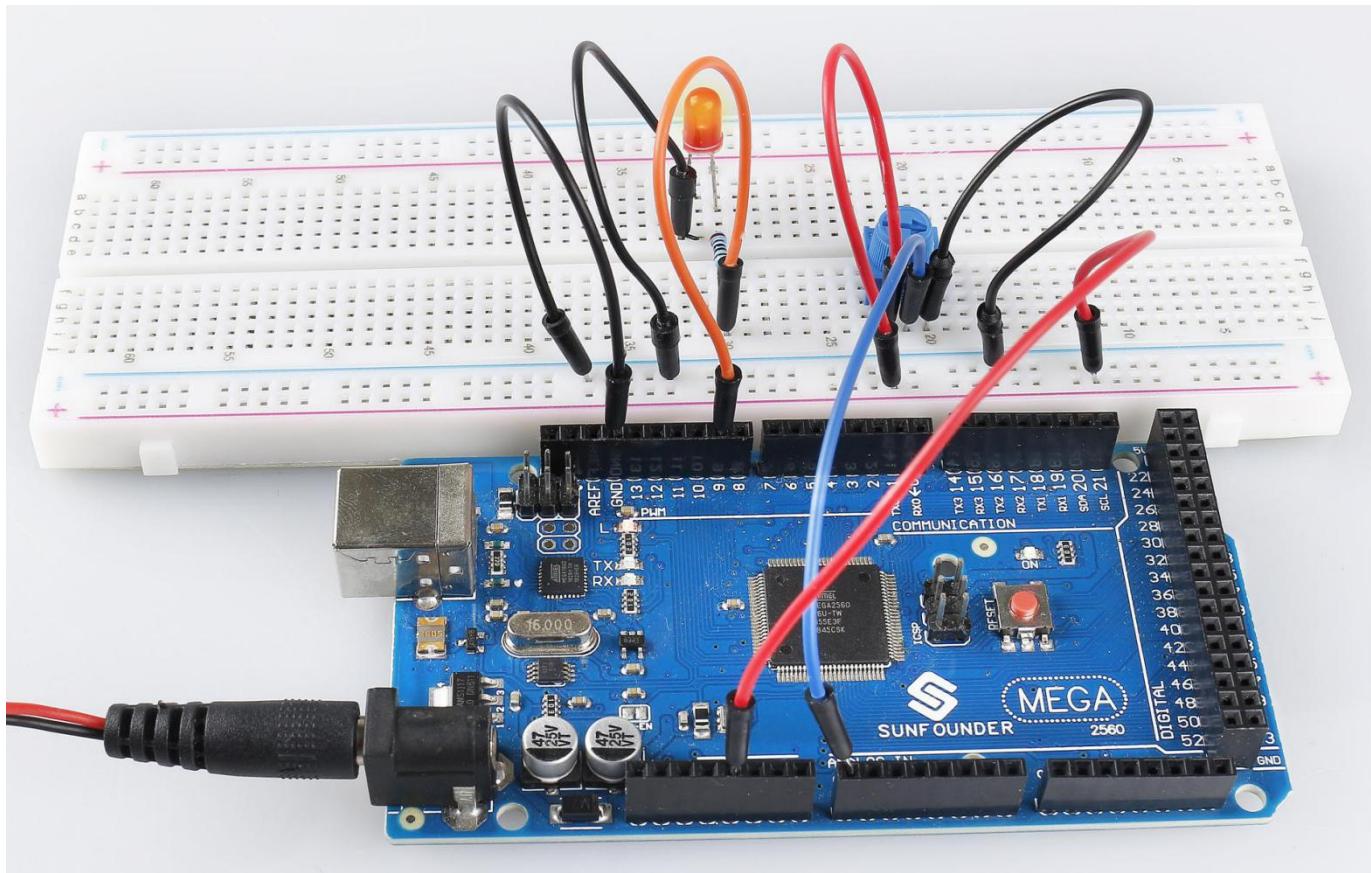
Map the potentiometer reading to the LED brightness value (0-1024 is mapped to 0-255).

```
int brightness = map(sensorValue,0,1024,0,255);
```

Write the brightness value to LED.

```
analogWrite(ledPin,brightness);
```

Phenomenon Picture

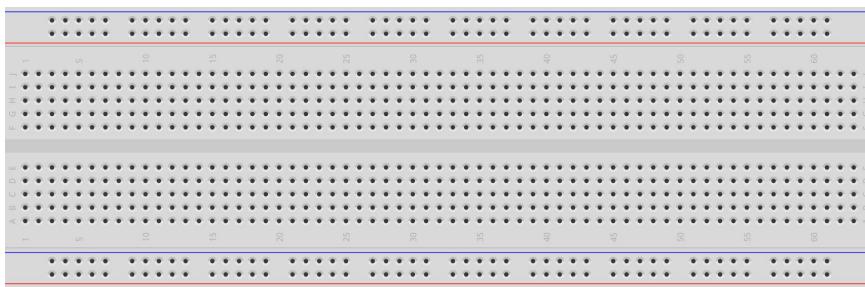


1.8 Serial Read

Overview

In addition to reading data from the electronic components, the Mega2560 board can read the data input in the serial port monitor, and you can use `Serial.read()` as the controller of the circuit experiment. Then we use LED to experiment with the `Serial.Read()` statement to control LED to turn on and off.

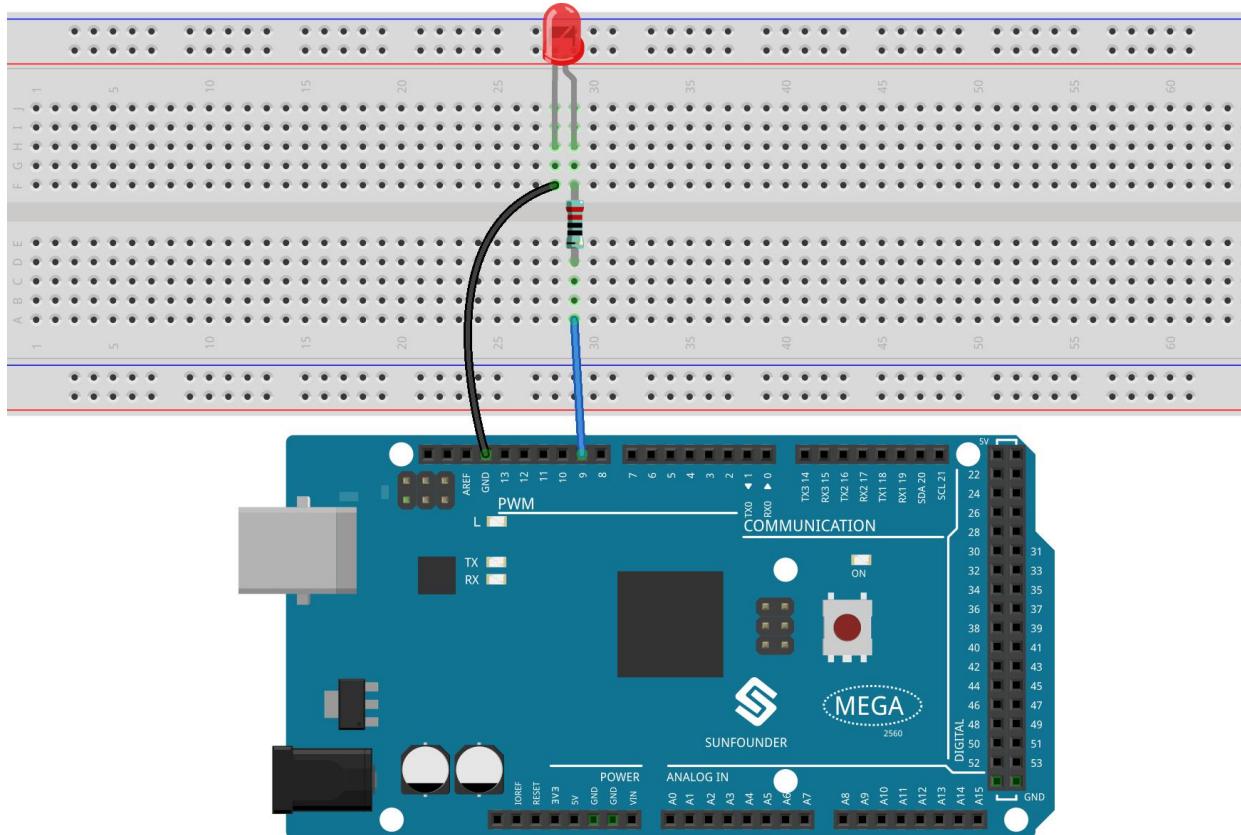
Components Required

1 * LED	1 * 220ohm Resistor	Several Jumper Wires
		
1 * Mega 2560 Board	1 * Breadboard	
		

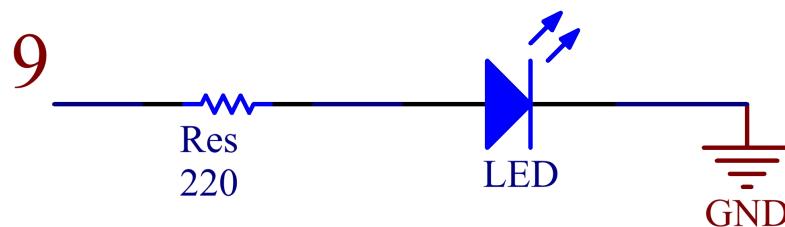
Note: Refer to Part 2 to check details of hardware.

Fritzing Circuit

In this example, we use digital pin 9 to drive LED. When 1 is entered in serial monitor, the LED lights up. When 0 is entered, the LED turns off.



Schematic Diagram

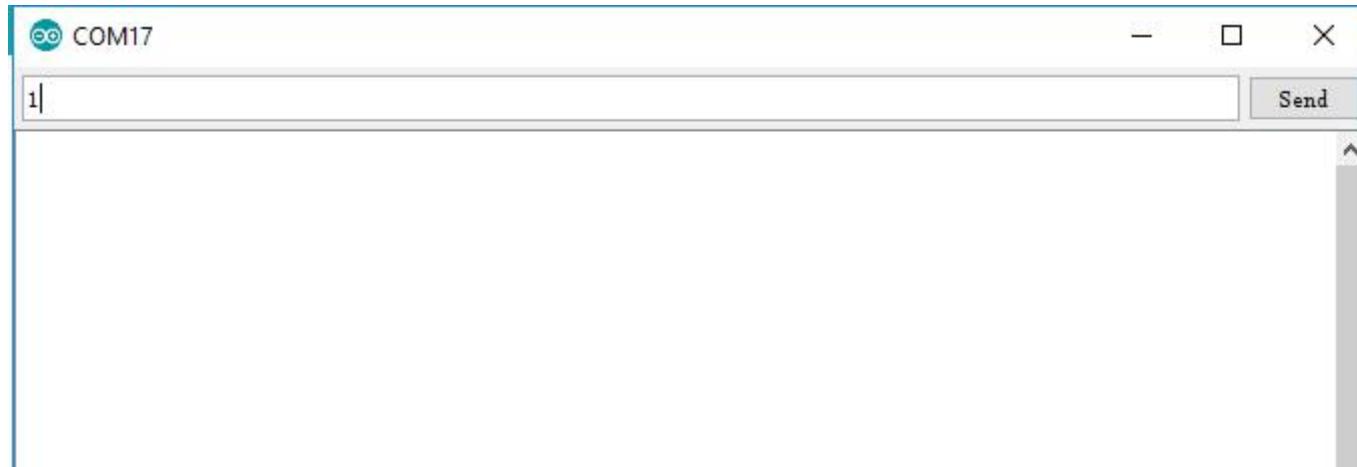


Code

```
const int ledPin = 9;  
int incomingByte = 0;  
  
void setup() {  
    pinMode(ledPin,OUTPUT);  
    Serial.begin(9600);  
}  
  
void loop() {  
    if (Serial.available() > 0) {  
        incomingByte = Serial.read();  
        if(incomingByte=='1'){digitalWrite(ledPin,HIGH);}  
    }  
}
```

```
else if(incomingByte=='0'){digitalWrite(ledPin,LOW);}
}
}
```

After the codes are uploaded to the Mega2560 board, please turn on the serial port monitor. Typing in "1" can make LED turn on and typing in "0" can make it turn off.



Code Analysis

Declare digital pin 9 as ledPin.

```
const int ledPin = 9;
```

Serial.read() reads a single byte of ASCII value, and therefore you need to declare a int type variable, incomingByte to store the acquired data.

```
int incomingByte = 0;
```

Run the serial communication in setup() and set the data rate to 9600.

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

Set ledPin to OUTPUT mode.

```
pinMode(ledPin,OUTPUT);
```

The state of serial port monitor is judged in loop(), and the information processing will be carried out only when the data are received.

```
if (Serial.available() > 0){}
```

Reads the input value in the serial port monitor and stores it to the variable incomingByte.

```
incomingByte = Serial.read();
```

When the character '1' is obtained, the LED is lit; when '0' is obtained, the LED turns off.

```
if(incomingByte=='1'){digitalWrite(ledPin,HIGH);}
else if(incomingByte=='0'){digitalWrite(ledPin,LOW);}
```

NOTE: Serial.read() takes the ASCII value in single character, which means that when you input '1', the obtained value is not the number '1', but the character '1' whose corresponding ASCII value is 49.

※ ASCII chart

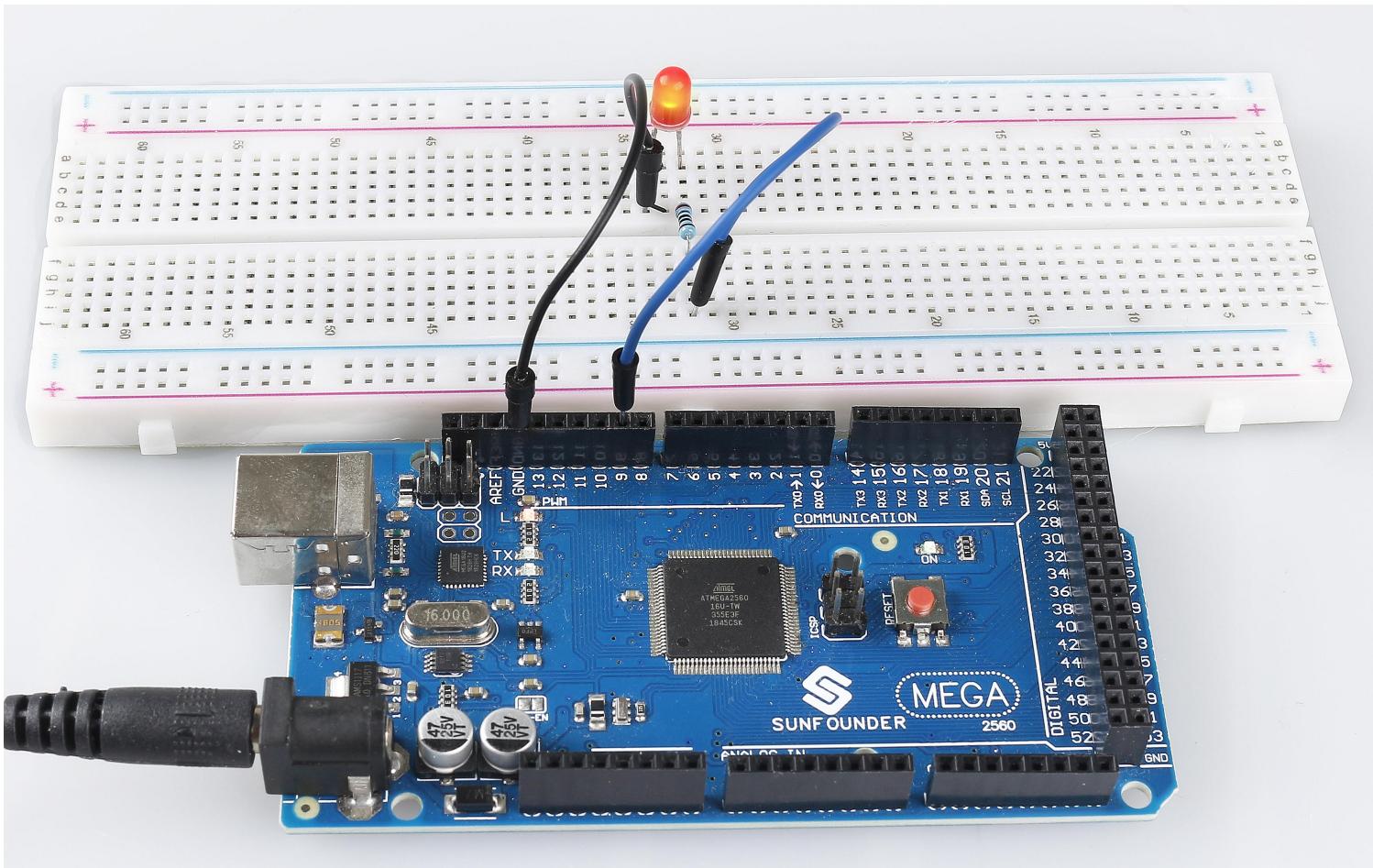
The ASCII (American Standard Code for Information Interchange) encoding dates to the 1960's. It is the standard way that text is encoded numerically.

Note that the first 32 characters (0-31) are non-printing characters, often called control characters. The more useful characters have been labeled.

ASCII Character		ASCII Character		ASCII Character		ASCII Character		ASCII Character		ASCII Character	
0	Null	23		46	.	69	E	92	/	115	s
1		24		47	/	70	F	93]	116	t
2		25		48	0	71	G	94	^	117	u
3		26		49	1	72	H	95	_	118	v
4		27		50	2	73	I	96	,	119	w
5		28		51	3	74	J	97	a	120	x
6		29		52	4	75	K	98	b	121	y
7		30		53	5	76	L	99	c	122	z
8		31		54	6	77	M	100	d	123	{
9	tab	32	space	55	7	78	N	101	e	124	

10	line feed	33	!	56	8	79	O	102	f	125	}
11		34	"	57	9	80	P	103	g	126	'
12		35	#	58	:	81	Q	104	h	127	
13	carriage return	36	\$	59	;	82	R	105	i		
14		37	%	60	<	83	S	106	j		
15		38	&	61	=	84	T	107	k		
16		39	,	62	>	85	U	108	l		
17		40	(63	?	86	V	109	m		
18		41)	64	@	87	W	110	n		
19		42	*	65	A	88	X	111	o		
20		43	+	66	B	89	Y	112	p		
21		44	,	67	C	90	Z	113	q		
22		45	-	68	D	91	[114	r		

Phenomenon Picture



1.9 Digital Input Pull-Up

Overview

When using some switch input devices, some pull-up or pull-down resistors are often used to keep the level of corresponding pins at certain value on the condition that the device is not working. Such as in 1.4 Digital Read, a 10k resistor is used to make the pin be connected to GND under the condition that the button is not pressed down. If we have used a lot of switch input components and want to simplify the circuit, we can set the pin mode to `[INPUT_PULLUP]` in the code so that the pin reads the high level in the suspended state.

Components Required

1 * potentiometer



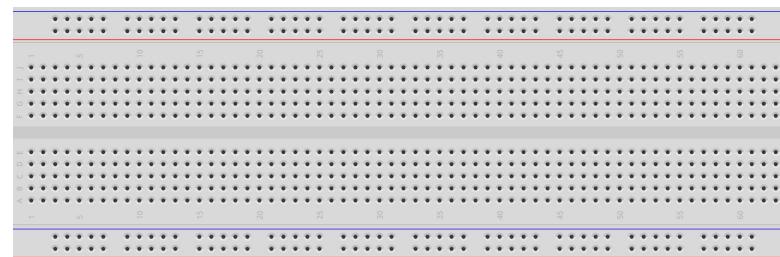
1 * Mega 2560 Board



Several Jumper Wires

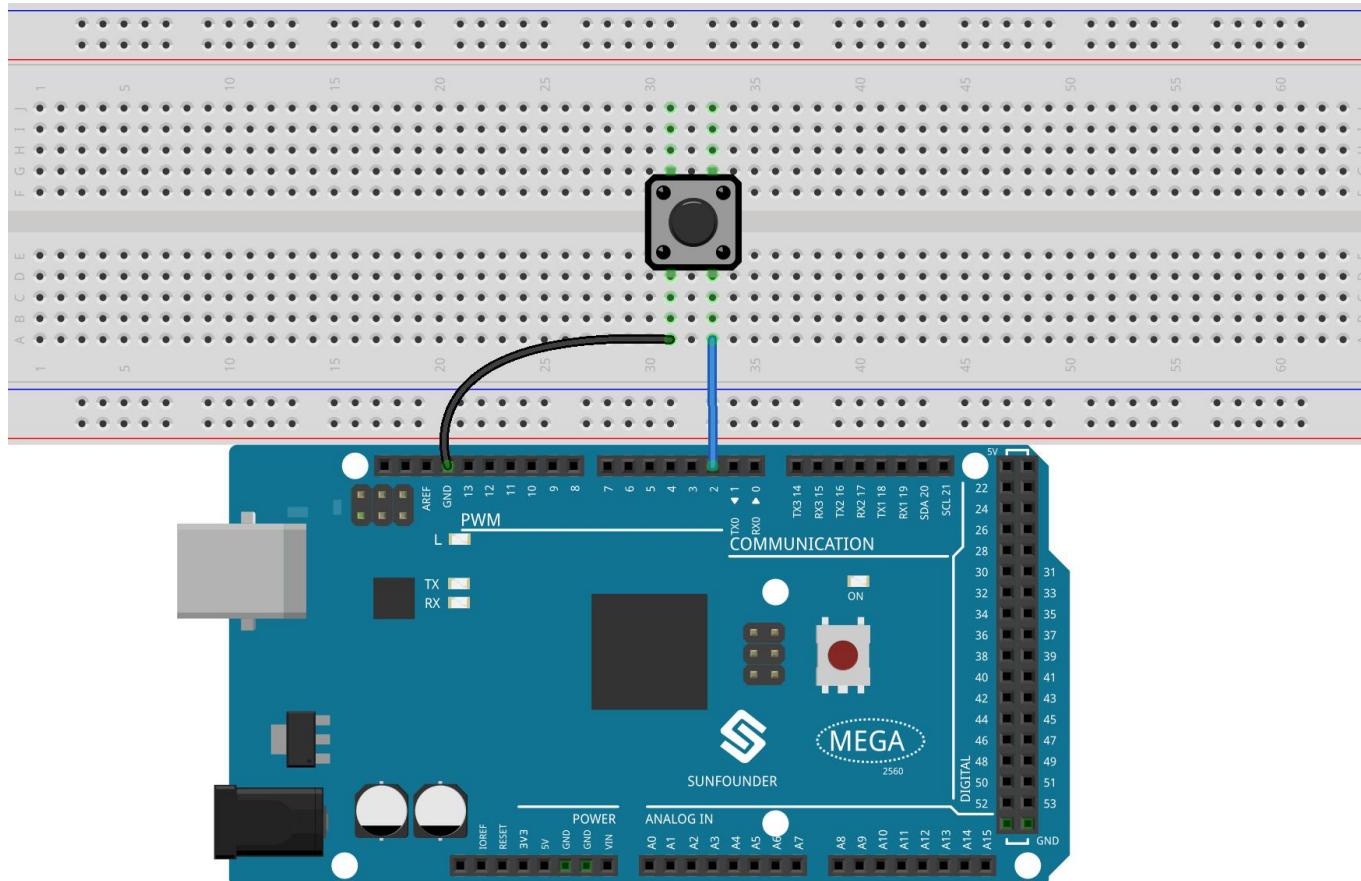


1 * Breadboard

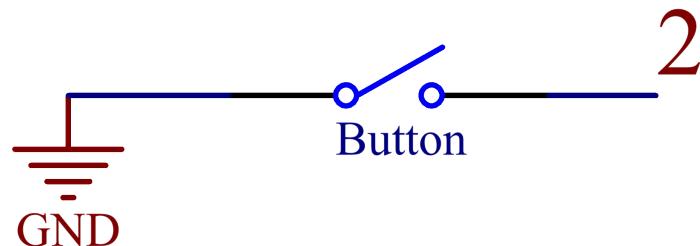


Fritzing Circuit

In this example, we use pin 2 to read the signal of button. The internal pull-up in pin 2 is valid, so if the button isn't pressed, HIGH is read in pin 2; when the button is pressed, LOW is read.



Schematic Diagram



Code

```
void setup() {
    Serial.begin(9600);
    pinMode(2, INPUT_PULLUP);
}

void loop() {
    int buttonState = digitalRead(2);
    Serial.println(buttonState);
    delay(1);
}
```

After the codes are uploaded to the Mega2560 board, you can open the serial port monitor to view the read values of the pin. When the Button is pressed, the serial port monitor will display "0", and the "1" will be displayed when the button is released.

Code Analysis

Run the serial communication in setup() and set the data rate to 9600.

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

Configure pin 2 as an input and enable the internal pull-up resistor.

```
pinMode(2, INPUT_PULLUP);
```

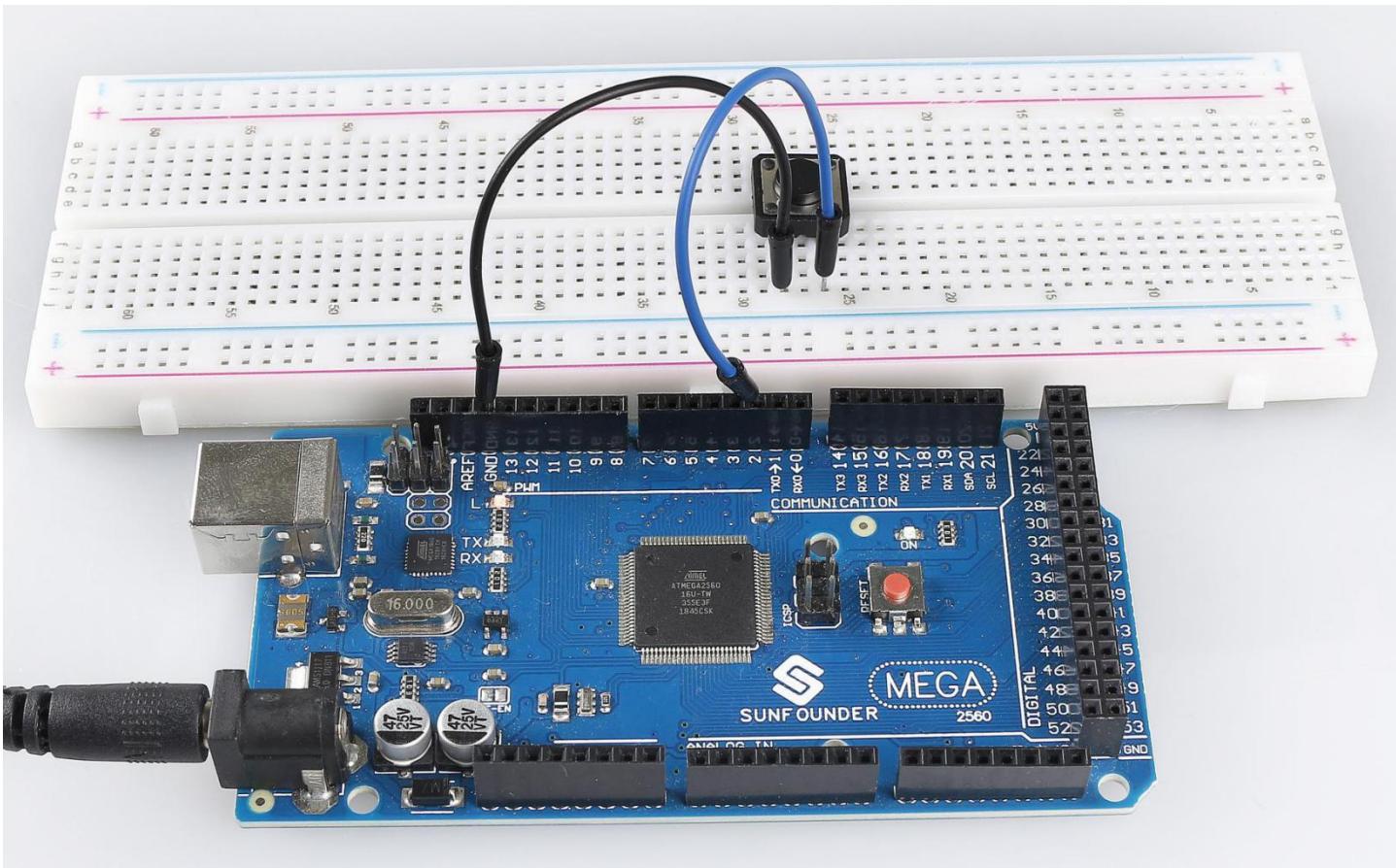
Read the level state from the digital pin 2 by using digitalRead() statement in loop() and declare a variable to store it.

```
int buttonState = digitalRead(2);
```

Print the values stored by variables on the serial port monitor.

```
Serial.println(buttonState);
```

Phenomenon Picture

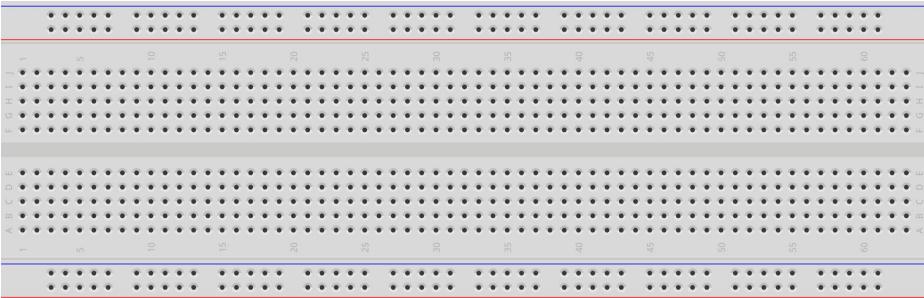


1.10 State Change Detection

Overview

When you use button, you can not only press down the button, light on the LED, release the button, turn off the LED, but also can switch the working state of the LED every time the button is pressed. In order to achieve this effect, you need to know when the state of the button changes from **off** to **on**, that is, "state change detection". In this lesson, we will print the results of state change detection of the button in the serial monitor.

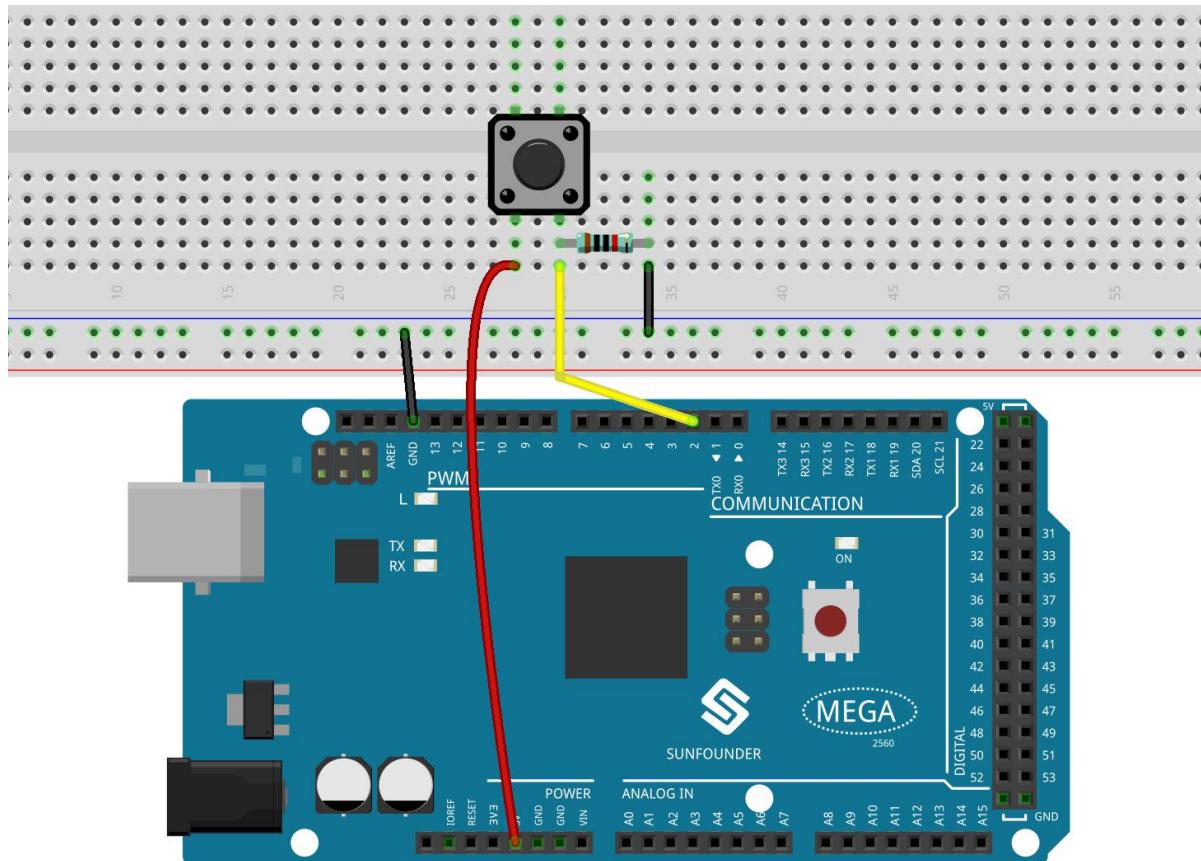
Components Required

1 * Button		1 * 10k ohm resistor		Several Jumper Wires		
1 * Mega 2560 Board		1 * Breadboard				

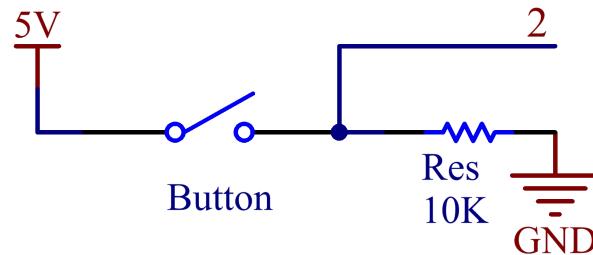
Note: Refer to Part 2 to check details of hardware.

Fritzing Circuit

In this example, we use pin 2 to read the signal of the button.



Schematic Diagram



Code

```
const int buttonPin = 2;  
int detectionState = 0;  
int buttonState = 0;  
int lastButtonState = 0;  
  
void setup() {  
    pinMode(buttonPin, INPUT);  
    Serial.begin(9600);  
}  
  
void loop() {  
    buttonState = digitalRead(buttonPin);  
    if (buttonState != lastButtonState) {
```

```
if (buttonState == HIGH) {  
    detectionState=(detectionState+1)%2;  
    Serial.print("The detection state is:");  
    Serial.println(detectionState);  
}  
delay(50);  
}  
lastButtonState = buttonState;  
}
```

After the codes are uploaded into the Mega2560 board, the output number will switch between 0 and 1 every time you press the button.

Code Analysis

Declare a pin connected to Button.

```
const int buttonPin = 2;
```

Declare a variable called 「detectionState」 to record every state of state change detection.

```
int detectionState = 0;
```

Declare two variables to read the state of the button for state change detection.

```
int buttonState = 0;
```

```
int lastButtonState = 0;
```

In setup(), initialize the pins and then start up the serial monitor.

```
pinMode(buttonPin, INPUT);  
Serial.begin(9600);
```

In loop(), read the value of buttonPin and then assign to the variable buttonState.

```
buttonState = digitalRead(buttonPin);
```

Compare buttonState with lastButtonState, if they are not equal, it indicates that the state is changed. A delay(50) is needed to realize debouncing during the changing detection. After comparison, assign the buttonState to lastButtonState to make the next round of judgment.

```
if (buttonState != lastButtonState) {  
    ...  
    delay(50);  
}  
lastButtonState = buttonState;
```

The state change judgment installed (buttonState != lastButtonState), the further judgment is made to get the condition [Press the button] .

```
if (buttonState == HIGH) {  
    ...  
}
```

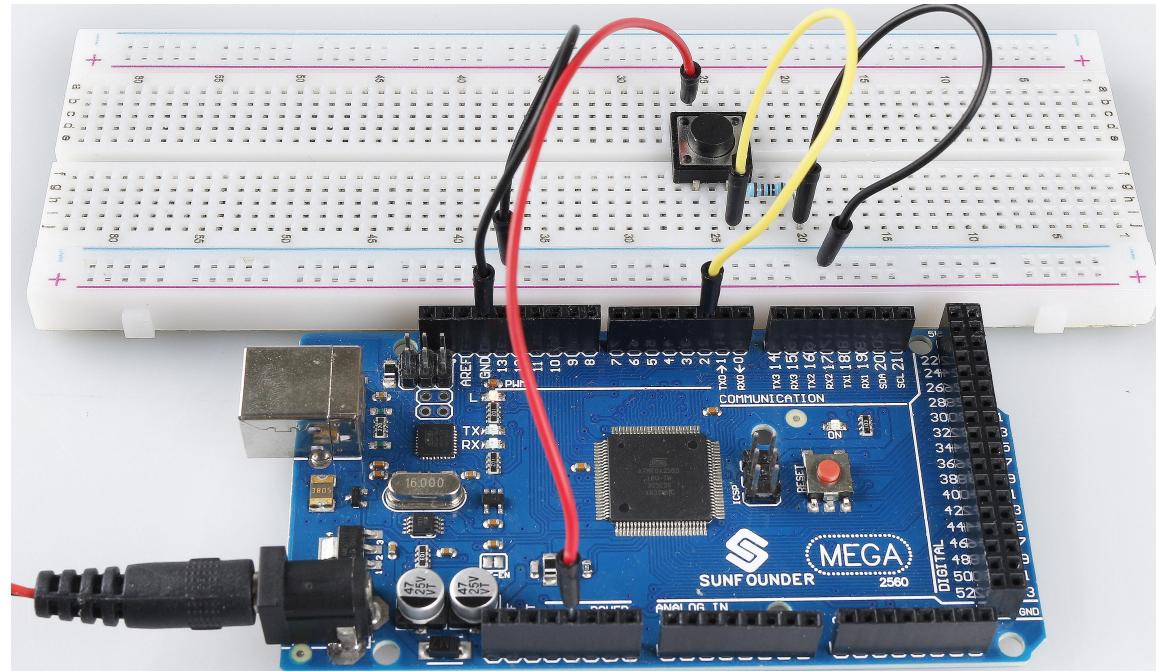
}

Under the state [Press the button] , detectionState is being operated and it switches between 1 and 0.

Meanwhile, the value of detectionState is printed.

```
detectionState=(detectionState+1)%2;  
Serial.print("The detection state is:");  
Serial.println(detectionState);
```

Phenomenon Picture



1.11 Interval

Overview

Sometimes you need to do two things at once. For example you might want to blink an LED while reading a button press. In this case, you can't use `delay()`, because Arduino pauses your program during the `delay()`. If the button is pressed while Arduino is paused waiting for the `delay()` to pass, your program will miss the button press.

This sketch demonstrates how to blink an LED without using `delay()`. It turns the LED on and then makes note of the time. Then, each time through `loop()`, it checks to see if the desired blink time has passed. If it has, it toggles the LED on or off and makes note of the new time. In this way the LED blinks continuously while the sketch execution never lags on a single instruction.

An analogy would be warming up a pizza in your microwave, and also waiting some important email. You put the pizza in the microwave and set it for 10 minutes. The analogy to using `delay()` would be to sit in front of the microwave watching the timer count down from 10 minutes until the timer reaches zero. If the important email arrives during this time you will miss it.

What you would do in real life would be to turn on the pizza, and then check your email, and then maybe do something else (that doesn't take too long!) and every so often you will come back to the microwave to see if the timer has reached zero, indicating that your pizza is done.

Components Required

1 * LED



1 * 220ohm Resistor



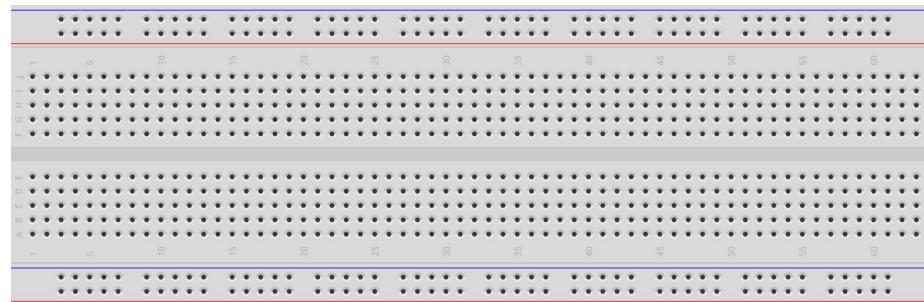
Several Jumper Wires



1 * Mega 2560 Board



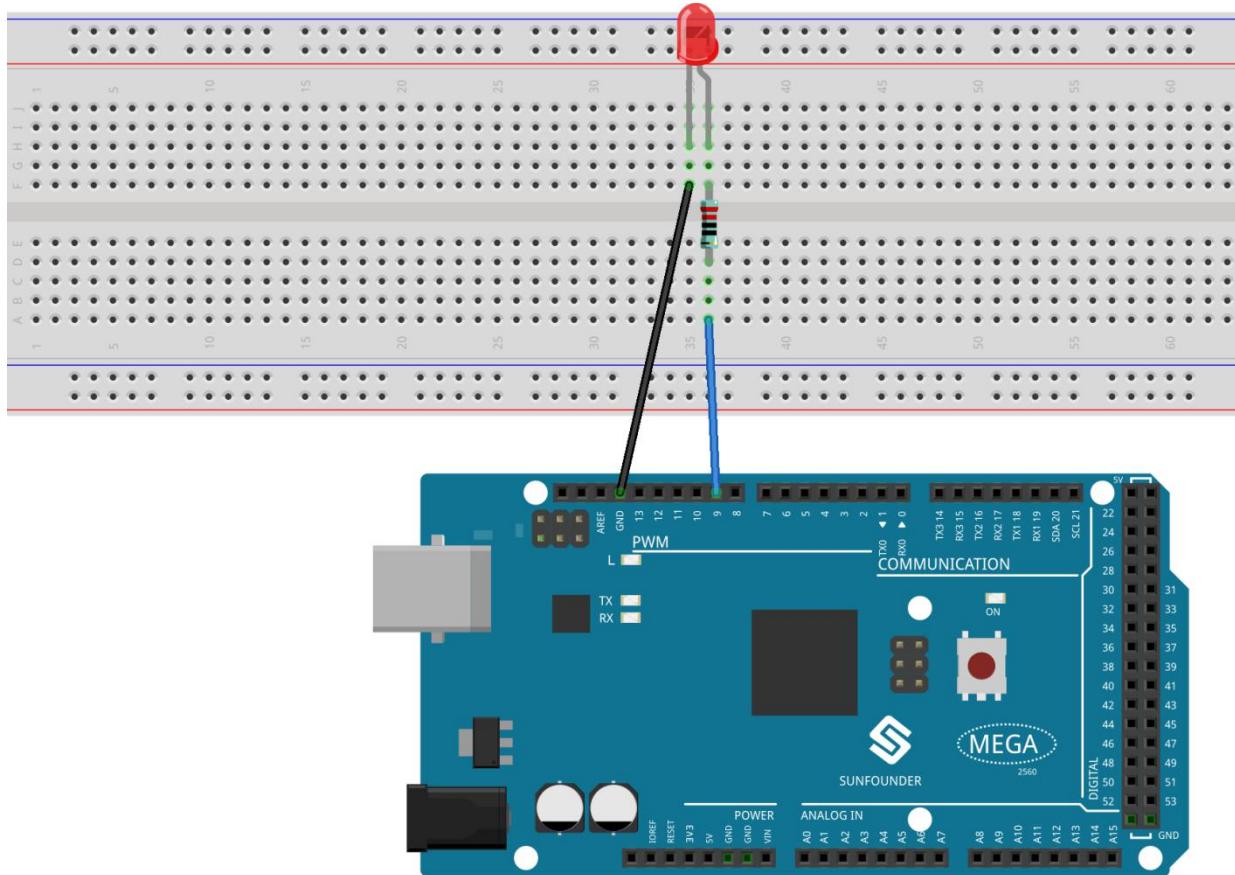
1 * Breadboard



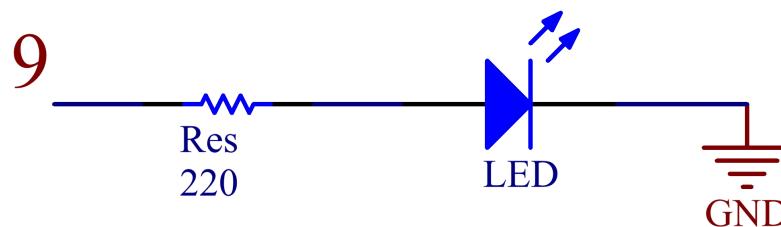
Note: Refer to Part 2 to check details of hardware.

Fritzing Circuit

In this example, we use digital pin 9 to drive the LED, and we attach one side of the resistor to the corresponding digital pins. The longer pin of LED (a positive electrode, referred to as anode) is connected to the other side of the resistor. The shorter pin (a negative electrode, referred to as cathode) of LED is attached to GND.



Schematic Diagram



Code

```
const int ledPin = 9;
int ledState = LOW;
unsigned long previousMillis = 0;
const long interval = 1000;
void setup() {
    pinMode(ledPin, OUTPUT);
}
void loop() {
    unsigned long currentMillis = millis();
    if (currentMillis - previousMillis >= interval) {
        previousMillis = currentMillis; // save the last time you blinked the LED
        if (ledState == LOW) {
```

```
    ledState = HIGH;  
} else {  
    ledState = LOW;  
}  
digitalWrite(ledPin, ledState);  
}  
}
```

When you finish uploading the codes to the Mega2560 board, you can see the LED uploading.

Code Analysis

Declare the digital pin 9 as ledPin.

```
const int ledPin = 9;
```

Set the state of ledState to LOW to turn off the LED.

```
int ledState = LOW;
```

Initial a variable named previousMillis to store previous operating time of microcontroller.

```
unsigned long previousMillis = 0;
```

Set the interval time to 1000ms (milliseconds).

```
const long interval = 1000;
```

Set ledPin to OUTPUT mode.

```
pinMode(ledPin, OUTPUT);
```

In loop(), declare currentMillis to store the current time.

```
unsigned long currentMillis = millis();
```

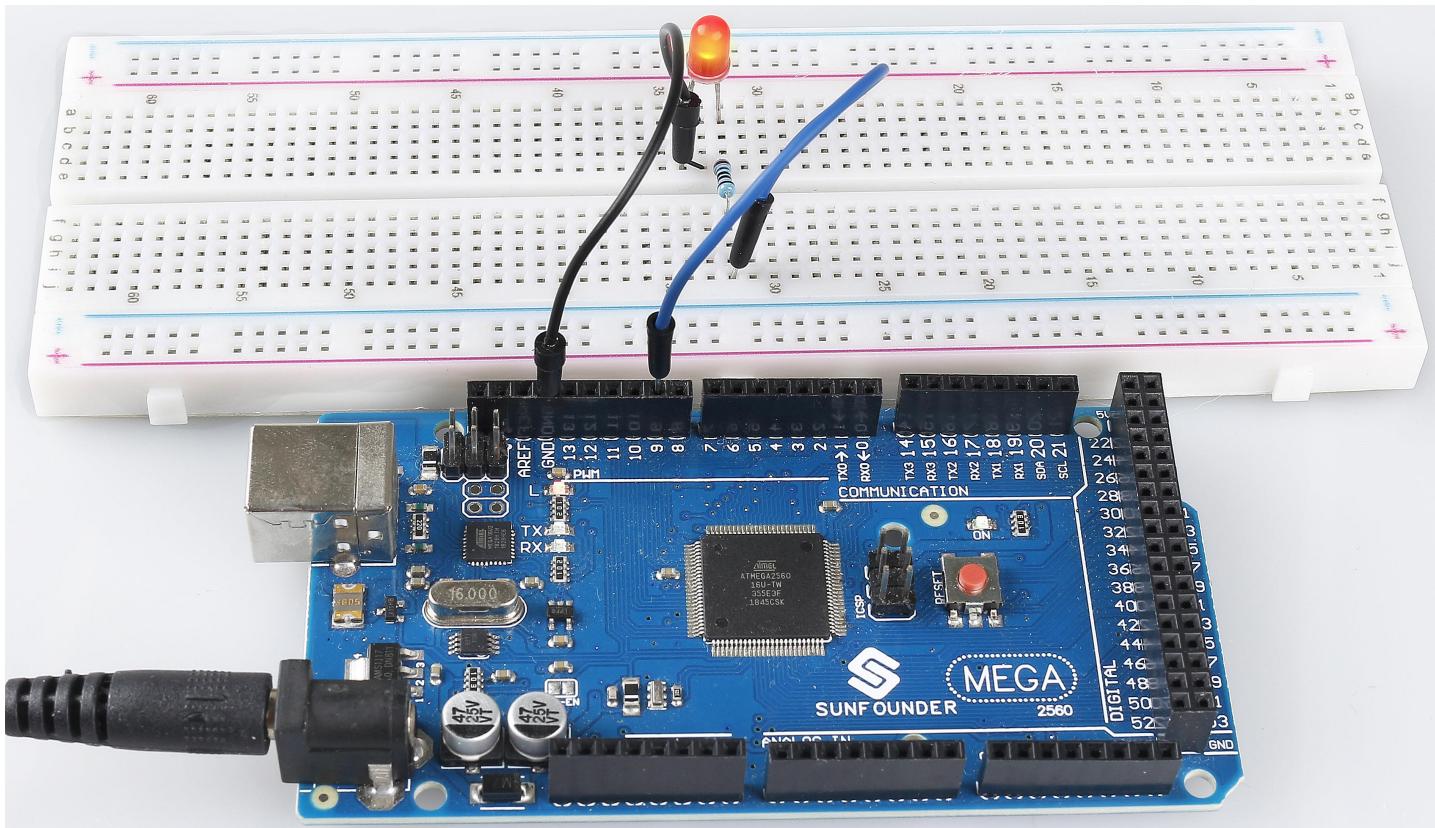
When the interval between the current operating time and last updating time is larger than 1000ms, certain functions are triggered. Meanwhile, update the previousMillis to the current time for the next triggering that is to happen 1 second latter.

```
if (currentMillis - previousMillis >= interval) {  
    previousMillis = currentMillis;// save the last time you blinked the LED  
    //...  
}
```

Here, certain functions executed at intervals are to change the state of LED.

```
if (ledState == LOW)  
{ledState = HIGH;}  
else  
{ledState = LOW;}  
digitalWrite(ledPin, ledState);
```

Phenomenon Picture



Part 2: Component

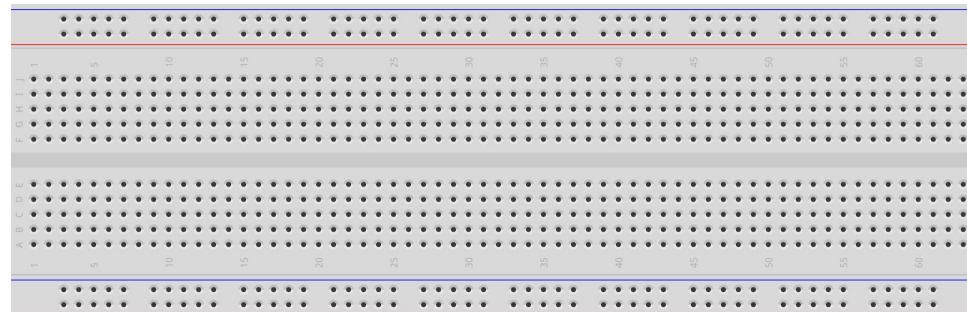
2.1 Common Component

Overview

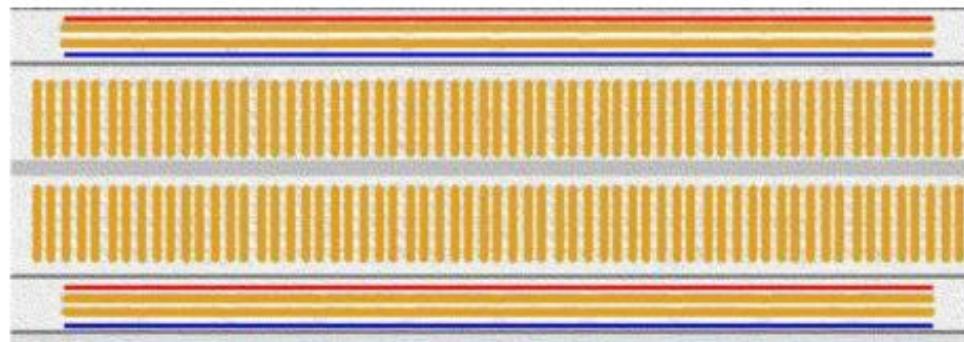
This lesson introduces some common components, including breadboard, jump wires, resistor, capacitor, transistor, diode, Prototype Shield and so on.

Breadboard

A breadboard is a construction base for prototyping of electronics. It is used to build and test circuits quickly before finalizing any circuit design. And it has many holes into which components like ICs and resistors as well as jumper wires mentioned above can be inserted. The breadboard allows you to easily plug in and remove components. So if there are going to be many changes or if you just want to make a circuit quickly, it will be much quicker than soldering up your circuit. Therefore in lots of experiments, it is often used as a hub to connect two or more devices.



This is the internal structure of a full+ breadboard. Although there are holes on the breadboard, **internally some of them are connected with metal strips**. Those holes are to insert pins of devices or wires. There are four long metal strips on the long sides; the blue and red lines are marked just for clear observation. But you can take the blue line as the GND and red one as VCC for convenience. Every five holes in the middle are vertically connected with metal strips internally which don't connect with each other. You can connect them horizontally with wires or components. A groove is made in the middle on the breadboard for IC chips.



Jumper Wires

Wires that connect two terminals are called jumper wires. There are various kinds of jumper wires. Here we focus on those used in breadboard. Among others, they are used to transfer electrical signals from anywhere on the breadboard to the input/output pins of a microcontroller.

Jump wires are fitted by inserting their "end connectors" into the slots provided in the breadboard, beneath whose surface there are a few sets of parallel plates that connect the slots in groups of rows or columns depending on the area.

The "end connectors" are inserted into the breadboard, without soldering, in the particular slots that need to be connected in the specific prototype.

There are three types of jumper wire: Female-to-Female, Male-to-Male, and Male-to-Female. The reason we call it Male-to-Female is because it has the outstanding tip in one end as well as a sunk female end. Male-to-Male means both sides are male and Female-to-Female means both ends are female.



More than one type of them may be used in a project. **The color of the jump wires is different but it doesn't mean their function is different accordingly; it's just designed so to better identify the connection between each circuit.**

Resistor

Resistor is an electronic element that can limit the branch current. A fixed resistor is one whose resistance cannot be changed, while that of a potentiometer or variable resistor can be adjusted.

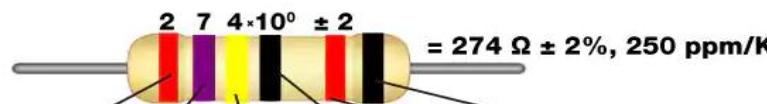
The resistors in this kit are fixed ones. It is essential in the circuit to protect the connected components. The following pictures show a real 220Ω resistor and two generally used circuit symbols for resistor. Ω is the unit of resistance and the larger includes $K\Omega$, $M\Omega$, etc. Their relationship can be shown as follows: $1 M\Omega = 1000 K\Omega$, $1 K\Omega = 1000 \Omega$, which means $1 M\Omega = 1000,000 \Omega = 10^6 \Omega$. Normally, the resistance is marked on it. So if you see these symbols in a circuit, it stands for a resistor.



220Ω

220Ω

6-Band



Color	1st Digit	2nd Digit	3rd Digit
Black	0	0	0
Brown	1	1	1
Red	2	2	2
Orange	3	3	3
Yellow	4	4	4
Green	5	5	5
Blue	6	6	6
Violet	7	7	7
Grey	8	8	8
White	9	9	9
Gold			
Silver			

Multiplier	Tolerance	Temperature Coefficient
1 Ω	± 1%	250 ppm/K
10 Ω	± 2%	100 ppm/K
100 Ω	± 5%	50 ppm/K
1k Ω	± 10%	15 ppm/K
10k Ω	± 20%	25 ppm/K
100k Ω	± 0.5%	20 ppm/K
1M Ω	± 0.25%	10 ppm/K
	± 0.1%	5 ppm/K
0.1 Ω	± 5%	1 ppm/K
0.01 Ω	± 10%	

4-Band $12 \times 10^5 \pm 5\%$ = 1,200 kΩ ± 5%

5-Band $100 \times 10^2 \pm 1\%$ = 10,000 Ω ± 1%

The resistance can be marked directly, in color code, and by character. The resistors offered in this kit are marked by different colors. Namely, the bands on the resistor indicate the resistance.

When using a resistor, we need to know its resistance first. Here are two methods: you can observe the bands on the resistor, or use a multimeter to measure the resistance. You are recommended to use the first method as it is more convenient and faster. If you are not sure about the value, use the multimeter.

As shown in the card, each color stands for a number.

Capacitor



The capacitor is a component that has the capacity to store energy in the form of electrical charge or to produce a potential difference (Static Voltage) between its plates, much like a small rechargeable battery.

Standard Units of Capacitance

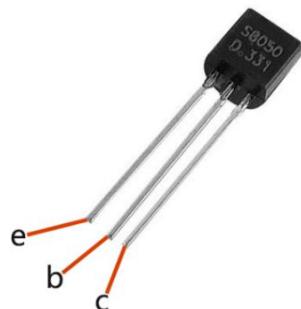
Microfarad (μF) $1\mu\text{F} = 1/1,000,000 = 0.000001 = 10^{-6}\text{ F}$

Nanofarad (nF) $1\text{nF} = 1/1,000,000,000 = 0.000000001 = 10^{-9}\text{F}$

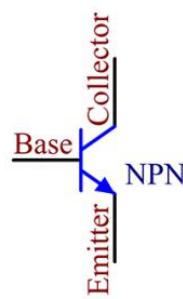
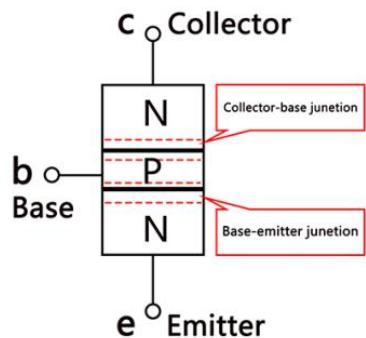
Picofarad (pF) $1\text{pF} = 1/1,000,000,000,000 = 0.000000000001 = 10^{-12}\text{F}$

Here we shown **104 capacitor(10 x 104PF)**. Just like the ring of resistors, the numbers on the capacitors help to read the values once assembled onto the board. The first two digits represent the value and the last digit of the number means the multiplier. Thus 104 represents a power of 10×10 to 4 (in pF) equal to 100 nF.

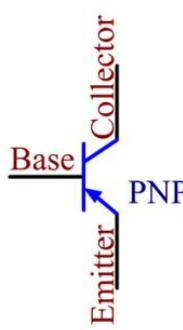
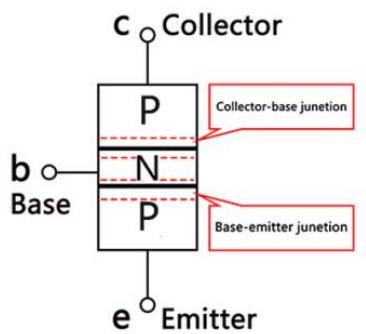
Transistor



Transistor is a semiconductor device that controls current by current. It functions by amplifying weak signal to larger amplitude signal and is also used for non-contact switch. A transistor is a three-layer structure composed of P-type and N-type semiconductors. They form the three regions internally. The thinner in the middle is the base region; the other two are both N-type or P-type ones – the smaller region with intense majority carriers is the emitter region, when the other one is the collector region. This composition enables the transistor to be an amplifier.



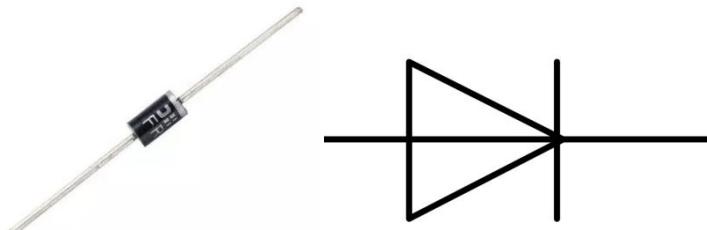
From these three regions, three poles are generated respectively, which are base (b), emitter (e), and collector (c). They form two P-N junctions, namely, the emitter junction and collection junction. The direction of the arrow in the transistor circuit symbol indicates that of the emitter junction. Based on the semiconductor type, transistors can be divided into two groups, the NPN and PNP ones. From the abbreviation, we can tell that the former is made of two N-type semiconductors and one P-type and that the latter is the opposite. See the figure below.



When a High level signal goes through an NPN transistor, it is energized. But a PNP one needs a Low level signal to manage it.

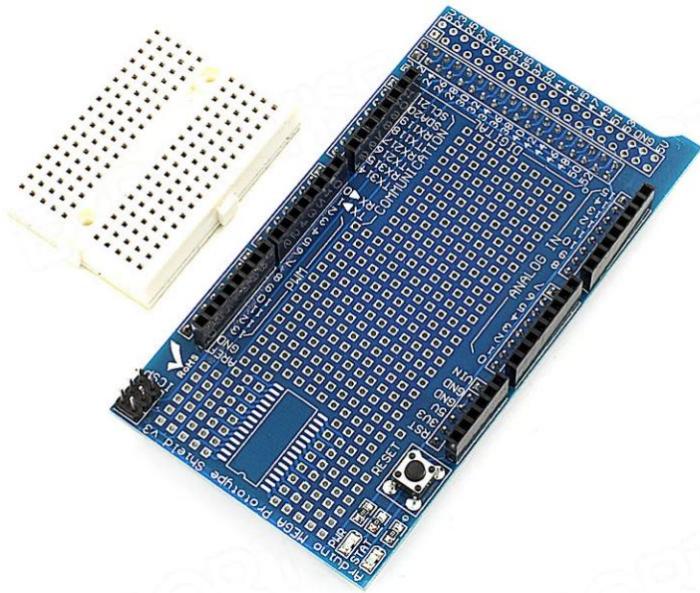
Diode

A diode is a two-terminal component in electronics with a unidirectional flow of current. It offers low resistance in the direction of current flow and offers high resistance in the opposite direction. Diodes are mostly used to prevent damage to components, especially due to electromotive force in circuits which are usually polarized.



The two terminals of a diode are polarized, with the positive end called anode and the negative end called cathode. The cathode is usually made of silver or has a color band. Controlling the direction of current flow is one of the key features of diodes — the current in a diode flows from anode to cathode. The behavior of a diode is similar to the behavior of a check valve. One of the most important characteristics of a diode is the non-linear current voltage. If higher voltage is connected to the anode, then current flows from anode to cathode, and the process is known as forward bias. However, if the higher voltage is connected to the cathode, then the diode does not conduct electricity, and the process is called reverse bias.

Prototype Shield



The Prototype Shield makes it easy for you to design custom circuits. You can solder parts to the prototyping area to create your project, or use it with a Tiny breadboard to quickly test circuit ideas without having to solder. It's got extra connections for all of the Arduino I/O pins, and it's got space to mount through-hole and surface mount integrated circuits. It's a convenient way to make your custom circuit and Arduino into a single module.

Specifications

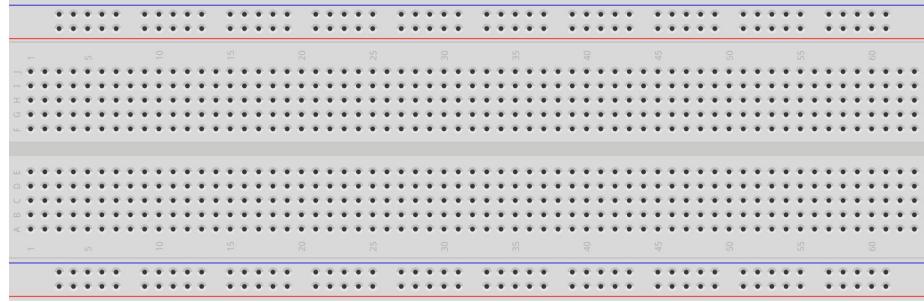
- It can be overlayed on the Sunfounder Mega2560 board directly. A Tiny breadboard is provided, which you can use to do some simple experiments.
- Provide a footprint for SOP28.
- It's got extra connections for all of the Arduino I/O pins
- We offer you pin22-pin53 bonding pad and you can use it to weld the component directly.

2.2 LED

Overview

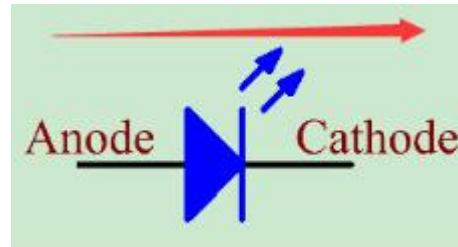
In this lesson, you will learn how to use LED. LED is a kind of common light-emitting device that works according to the principle that the recombination of electron and hole releases energy to give out light. This component is used widely in the current society, such as illumination, panel display, medical device and so on.

Components Required

1 * LED	1 * 220ohm Resistor	Several Jumper Wires
		
1 * Mega 2560 Board	1 * Breadboard	

Component Introduction

Semiconductor light-emitting diode is a type of component which can turn electric energy into light energy via PN junctions. By wavelength, it can be categorized into laser diode, infrared light-emitting diode and visible light-emitting diode which is usually known as light-emitting diode (LED).



Diode has unidirectional conductivity, so the current flow will be as the arrow indicates in figure circuit symbol. You can only provide the anode with a positive power and the cathode with a negative. Thus the LED will light up.

An LED has two pins. The longer one is the anode, and shorter one, the cathode. Pay attention not to connect them inversely. There is fixed forward voltage drop in the LED, so it cannot be connected with the circuit directly because the supply voltage can outweigh this drop and cause the LED to be burnt. The forward voltage of the red, yellow, and green LED is 1.8 V and that of the white one is 2.6 V. Most LEDs can withstand a maximum current of 20 mA, so we need to connect a current limiting resistor in series.

The formula of the resistance value is as follows:

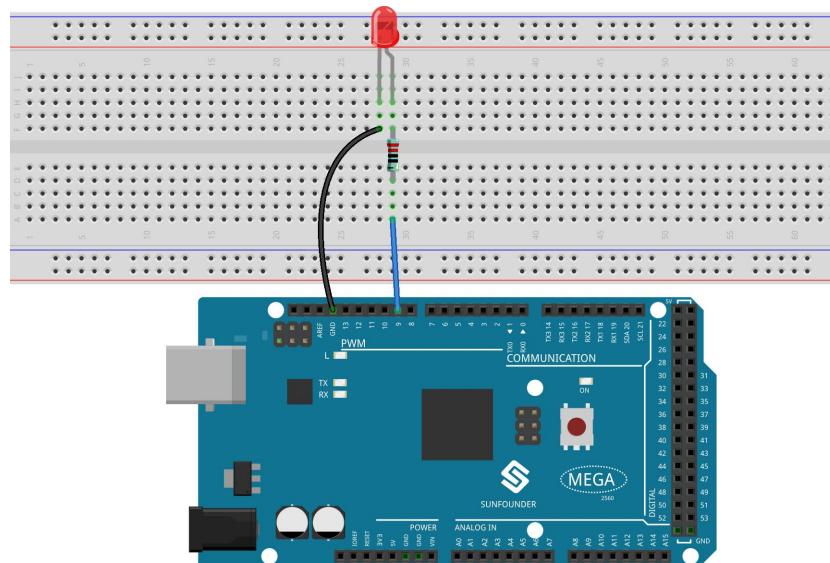
$$R = (V_{\text{supply}} - V_D)/I$$

R stands for the resistance value of the current limiting resistor, V_{supply} for voltage supply, V_D for voltage drop and I for the working current of the LED.

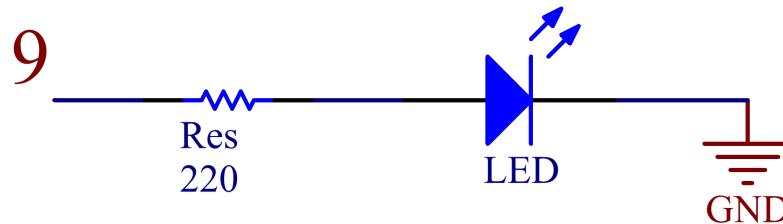
If we provide 5V voltage for the red LED, the minimum resistance of the current limiting resistor should be: $(5V - 1.8V) / 20mA = 160\Omega$. Therefore, you need a 160Ω or larger resistor to protect the LED. **You are recommended to use the 220Ω resistor offered in the kit.**

Fritzing Circuit

In this example, we use pin 9 to drive LED. Insert one side of the resistor in the digital pin 9 and connect the longer pin (a positive electrode, referred to as anode) of the LED with the other side of the resistor. Extend the shorter pin (a negative electrode, referred to as cathode) of the LED to GND.



Schematic Diagram



Code

Example 1:

```
const int ledPin = 9;

void setup() {
    pinMode(ledPin, OUTPUT);
}

void loop() {
    digitalWrite(ledPin, HIGH);
    delay(1000);
    digitalWrite(ledPin, LOW);
    delay(1000);
}
```

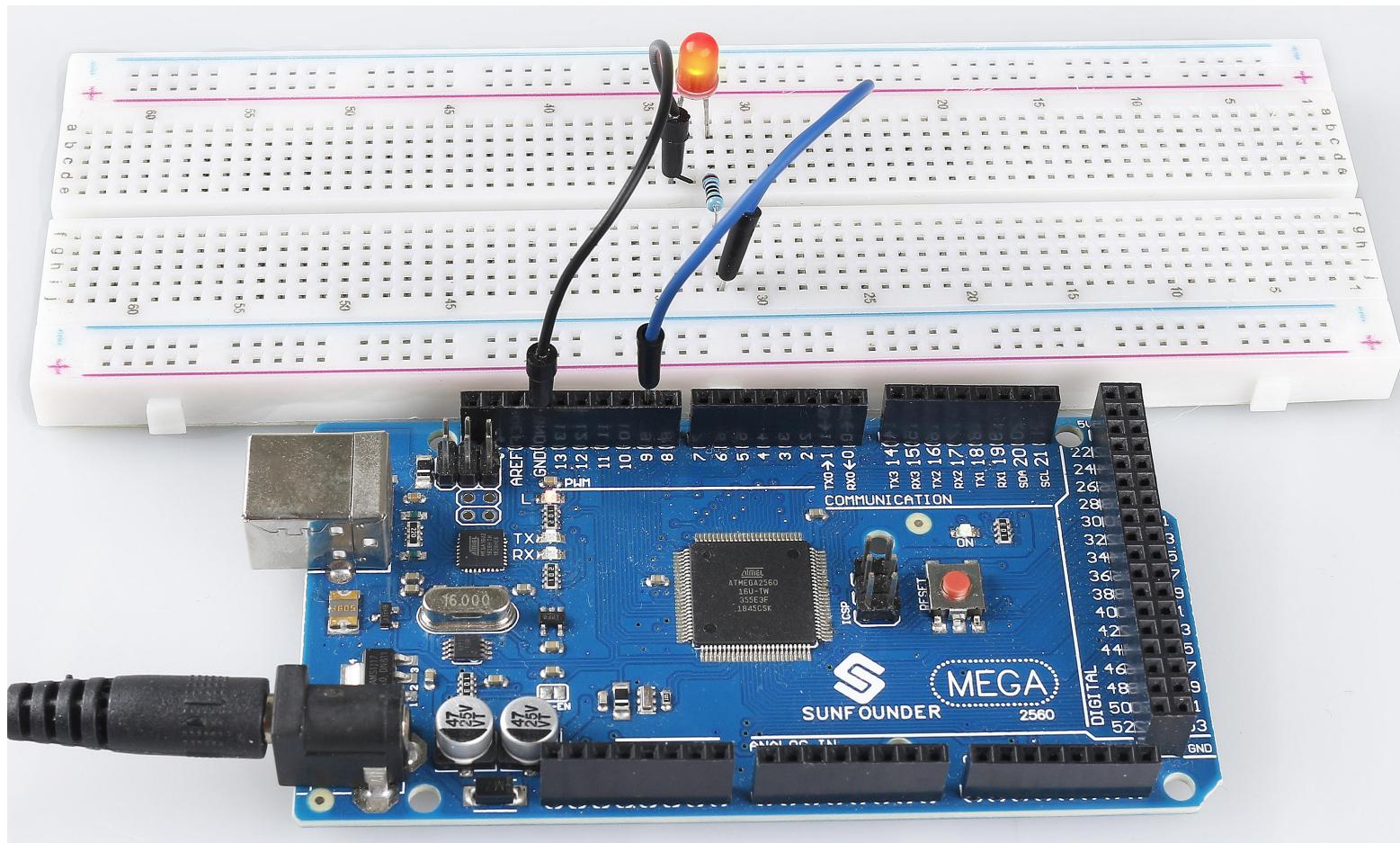
Finished uploading the codes to the Mega2560 board, you can see the LED flashing. Refer to [Part 1-1.2 Digital Write](#) to check the detail code explanation.

Example 2:

```
int ledPin = 9;  
void setup() {  
}  
void loop() {  
    for (int value = 0 ; value <= 255; value += 5) {  
        analogWrite(ledPin, value);  
        delay(30);  
    }  
}
```

After uploading the codes to the Mega2560 board, you can see the LED getting brighter, then turning off, getting brighter, then turning off again...this loop will continue in this way. About the detail code explanation, please refer to [Part 1-1.3 Analog Write](#).

Phenomenon Picture

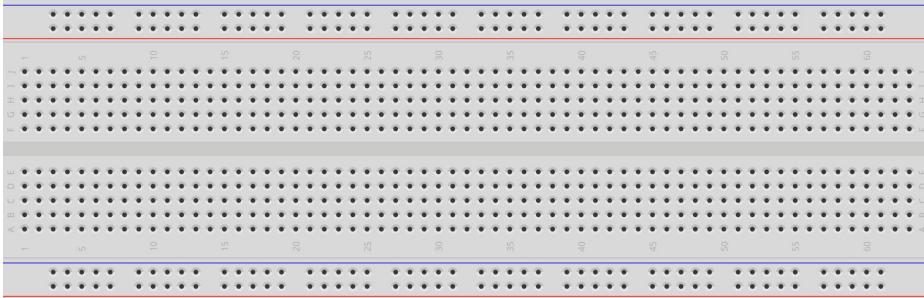


2.3 RGB LED

Overview

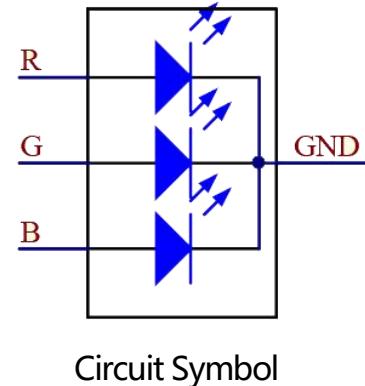
In this lesson, you will learn about how to use RGB LED. A RGB LED packages three LEDs in red, green and blue into one transparent or semitransparent plastic shell. It displays a broad array of colors by changing the input voltage of three pins and adding the three colors together in different ways. As is said in a statistic, RGB LED can create 16,777,216 different colors.

Components Required

1 * RGB LED 	3 * 220 ohm resistor 	Several Jumper Wires 
1 * Mega 2560 Board 	1 * Breadboard 	

Component Introduction

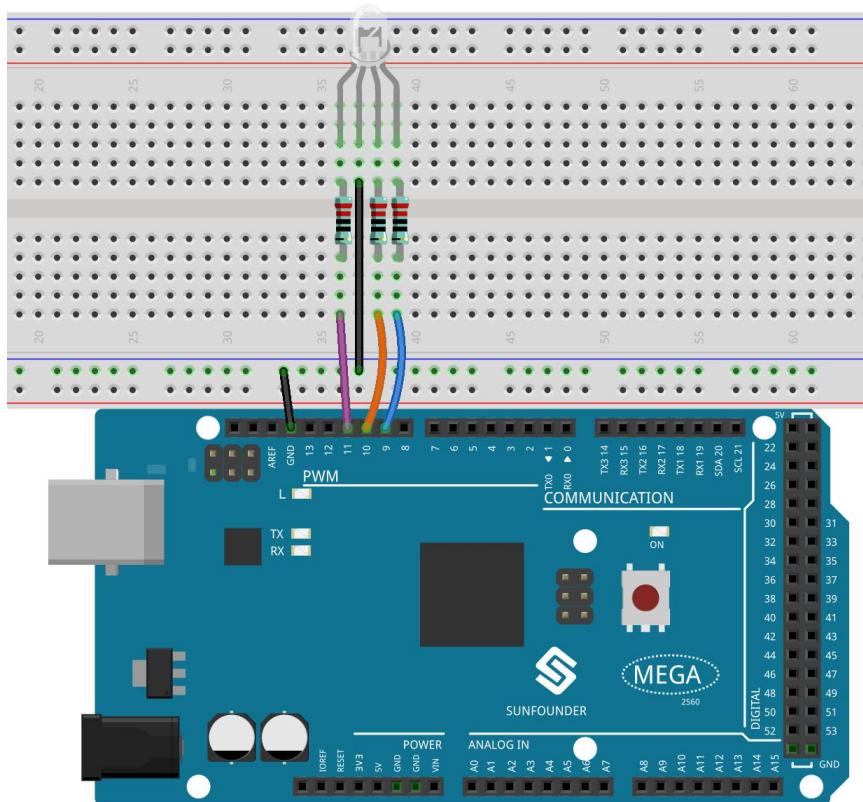
RGB LEDs can be categorized into common anode and common cathode ones. In this experiment, the latter is used. The common cathode, or CC, means to connect the cathodes of the three LEDs. After you connect it with GND and plug in the three pins, the LED will flash the corresponding color.



An RGB LED has 4 pins: the longest one is GND; the others are Red, Green and Blue. Touch its plastic shell and you will find a cut. The pin closest to the cut is the first pin, marked as Red, then GND, Green and Blue in turn.



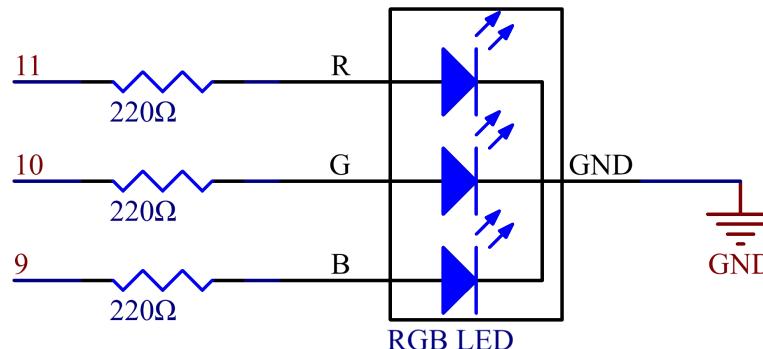
Or you can distinguish them in another way. As GND is the longest one and can be defined directly, you just need to confirm the other three pins. You can test it by giving them a small voltage. The forward voltage drop from the three pins to the GND are respectively 1.8V (red), 2.5V (blue), and 2.3V (green). Thus, when you connect the same current limiting resistor with the three pins and supply them with the same voltage, the red one is the brightest, and then comes the green and the blue one. Therefore, you may need to add a current limiting resistor with different resistances to the three pins for these colors.



Fritzing Circuit

Here we input a value between 0 and 255 to the three pins of the RGB LED to make it display different colors. After connecting the pins of R, G, and B to a current limiting resistor, connect them to the pin 9, pin 10, and pin 11 respectively. The longest pin (GND) of the LED connects to the GND of the Mega 2560. When the three pins are given different PWM values, the RGB LED will display different colors.

Schematic Diagram



Code

NOTE: A small portion of the codes are omitted. Please check the complete codes (`sunfounder_vincent_kit_for_arduino\code\2.3rgbLed\2.3rgbLed.ino`) in Arduino IDE.

```
const int redPin = 11;
const int greenPin = 10;
const int bluePin = 9;
void setup()
{
    pinMode(redPin, OUTPUT); // sets the redPin to be an output
    pinMode(greenPin, OUTPUT); // sets the greenPin to be an output
    pinMode(bluePin, OUTPUT); // sets the bluePin to be an output
```

```
}

void loop() // run over and over again
{
    color(255, 0, 0); // turn the RGB LED red
    delay(1000); // delay for 1 second
    color(0,255, 0); // turn the RGB LED green
    delay(1000); // delay for 1 second
    color(0, 0, 255); // turn the RGB LED blue
    delay(1000); // delay for 1 second
    // ...
}

void color (unsigned char red, unsigned char green, unsigned char blue)// the color generating function
{
    analogWrite(redPin, red);
    analogWrite(greenPin, green);
    analogWrite(bluePin, blue);
}
```

Uploaded the codes into the Mega2560 board, you can see that the color of the RGB LED changes once a second.

Code Analysis

In this example, the function used to assign values to the three pins of RGB is packaged in an independent subfunction color().

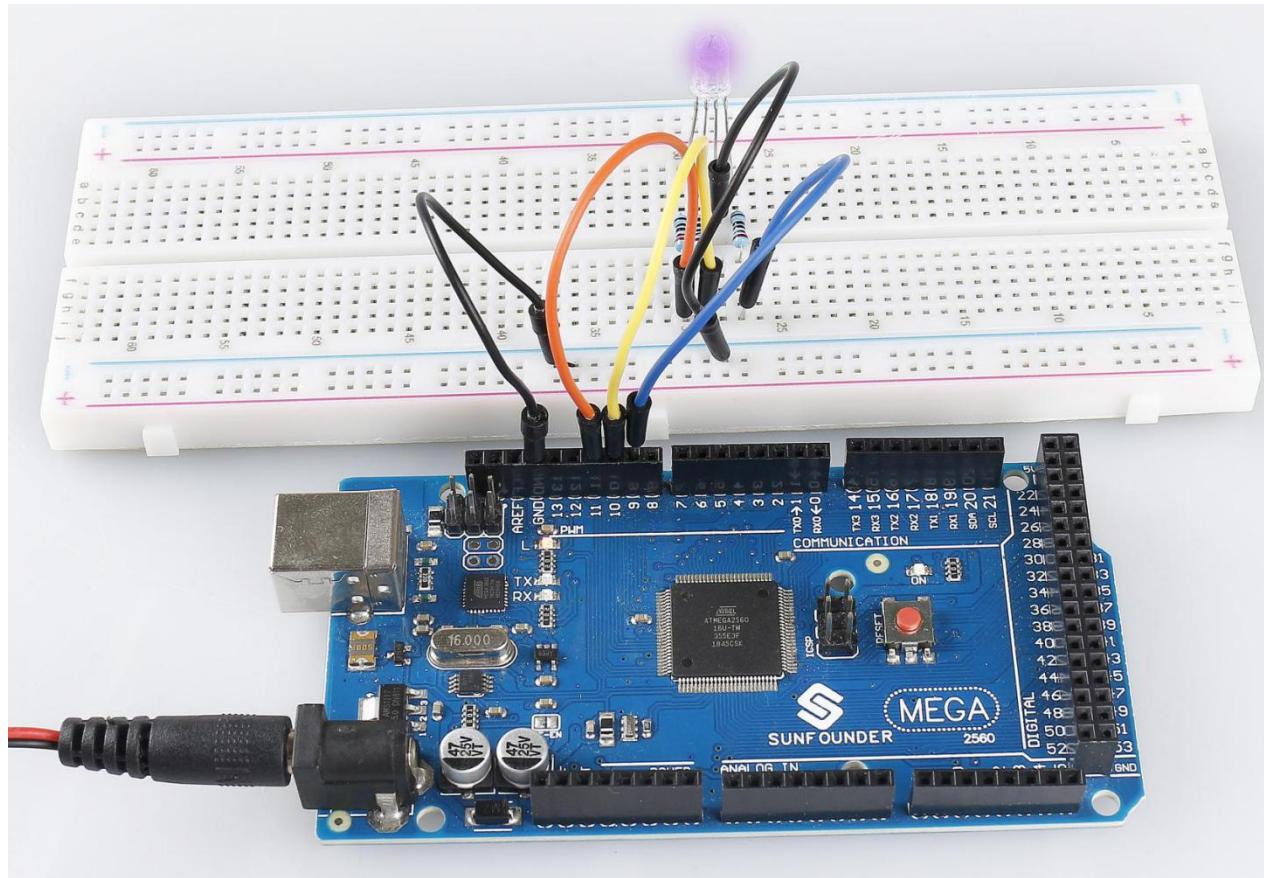
```
void color (unsigned char red, unsigned char green, unsigned char blue)// the color generating function
{
    analogWrite(redPin, red);
    analogWrite(greenPin, green);
    analogWrite(bluePin, blue);
}
```

In loop(), RGB value works as an input argument to call the function color() to realize that the RGB can emit different colors.

```
void loop() // run over and over again
{
    color(255, 0, 0); // turn the RGB LED red
    delay(1000); // delay for 1 second
    color(0,255, 0); // turn the RGB LED green
    delay(1000); // delay for 1 second
    color(0, 0, 255); // turn the RGB LED blue
    delay(1000); // delay for 1 second
    // ...
}
```

}

Phenomenon Picture

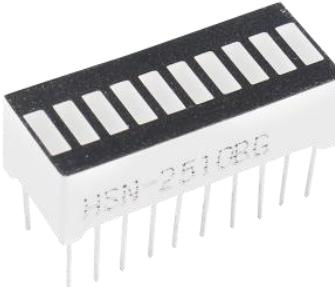


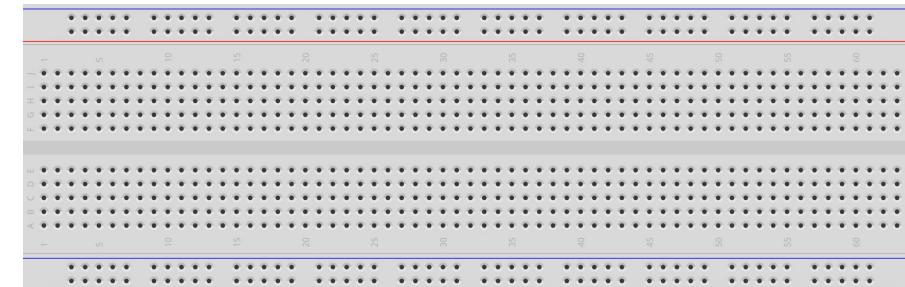
2.4 LED Bar Graph

Overview

In this lesson, you will learn something about LED Bar Graph. Generally, LED Bar Graph works as a battery level indicator, Audio equipment, industrial control panel. If we want, we can also find its other application.

Components Required

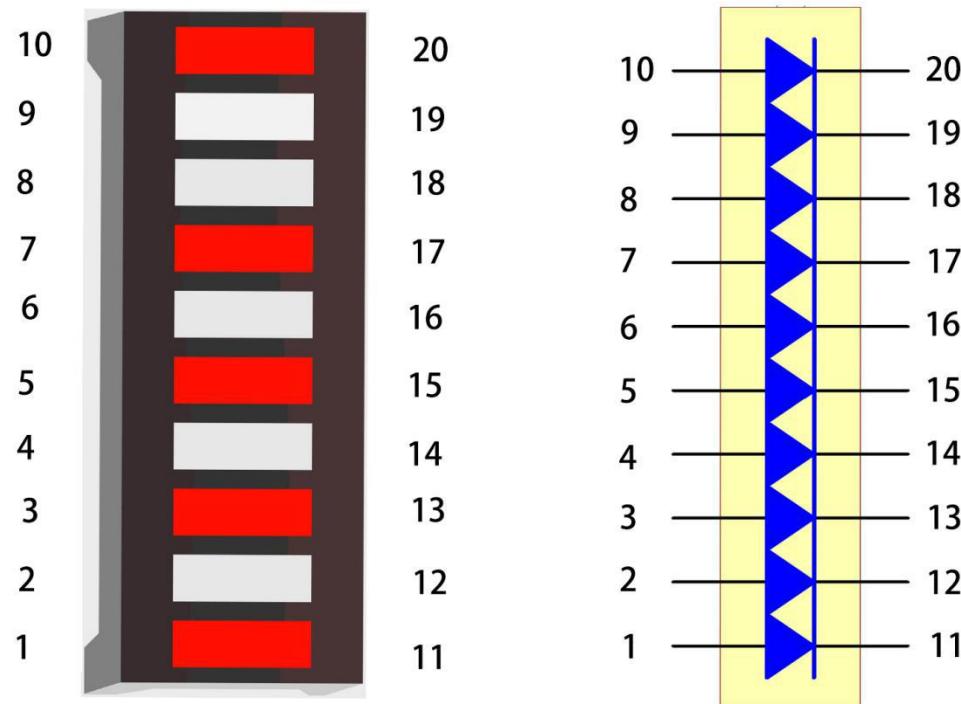
1 * LED Bar Graph	1 * 220ohm Resistor	Several Jumper Wires
		

1 * Mega 2560 Board	1 * Breadboard
	

Component Introduction

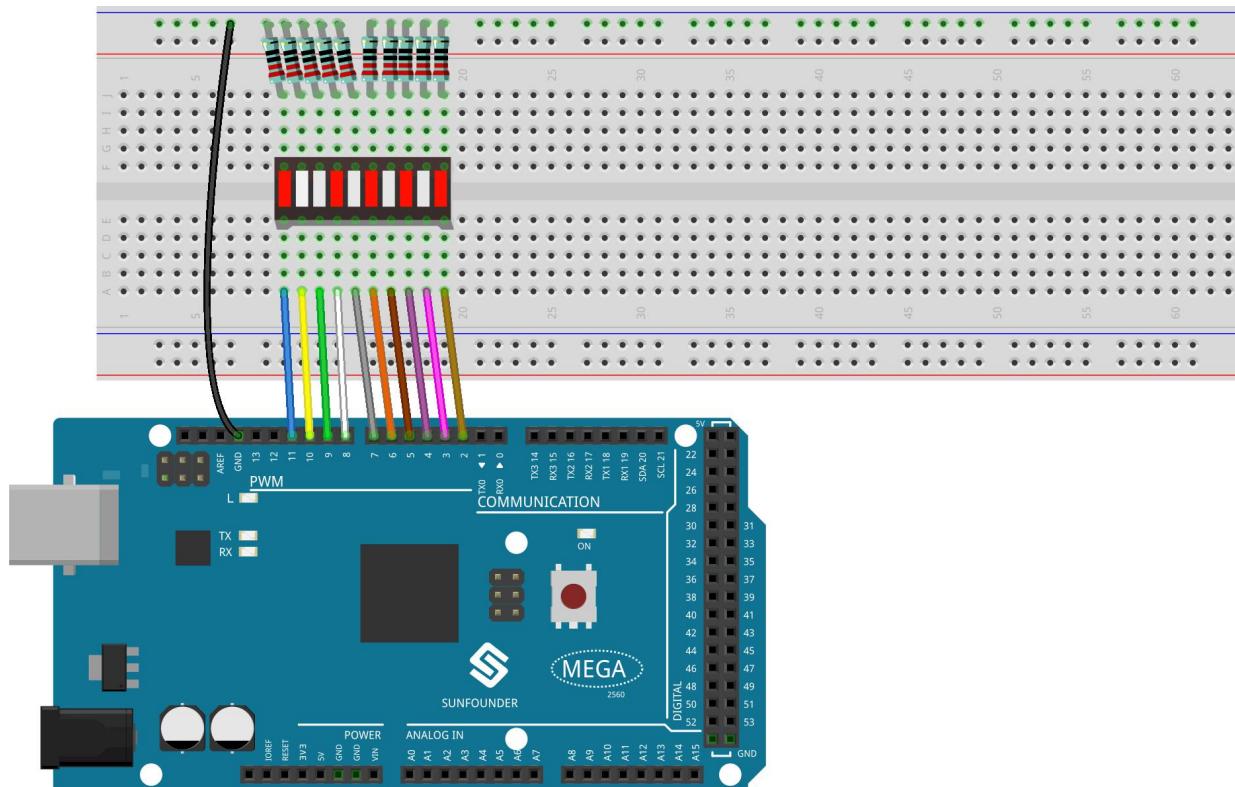
LED Bar Graph is an LED array, which is used to connect with electronic circuit or microcontroller. It's easy to connect LED bar graph with the circuit like as connecting 10 individual LEDs with 10 output pins.

Note: The anode is the side with a label (1-10).

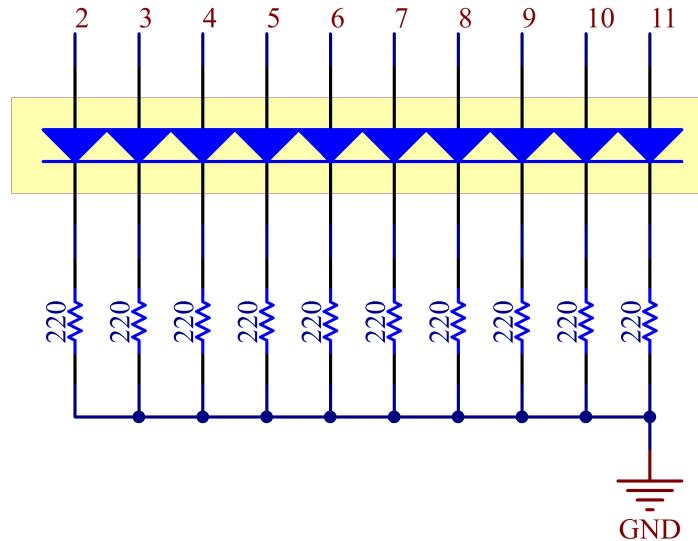


Fritzing Circuit

In this example, we use digital pins 2~11 to drive the LED Bar Graph. LED Bar Graph has ten separate LEDs inside and each LED has two pins. The left pins 1~10 of LED Bar Graph are connected with the digital pins 2~11 respectively; the right side pins 11~20 are separately extended to same side of these 220ohm resistors whose other sides are identically connected to GND.



Schematic Diagram



Code

```
void setup() {
  for(int i=2;i<=11;i++)
  {
    pinMode(i,OUTPUT);
  }
}

void loop()
```

```
for(int i=2;i<=11;i++)  
{  
    digitalWrite(i,HIGH);  
    delay(500);  
    digitalWrite(i,LOW);  
    delay(500);  
}  
}
```

Uploaded the codes to the Mega2560 board, you can see that the LEDs on the LED Bar Graph flash in sequence.

Code Analysis

The codes in setup() use the for loop to initialize pins 2~11 to output mode in turn.

```
for(int i=2;i<=11;i++)  
{  
    pinMode(i,OUTPUT);  
}
```

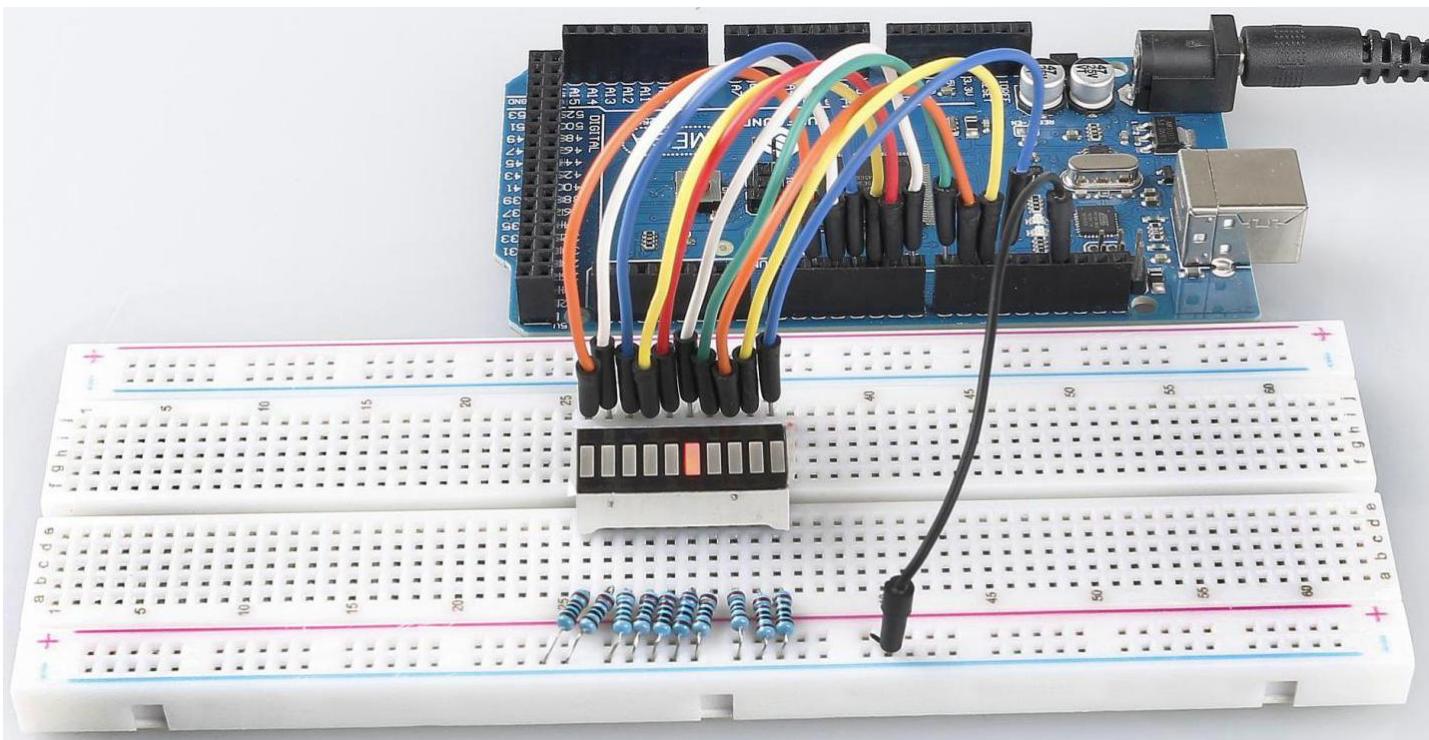
The for loop is used in loop() to make the LED flash(turn on 0.5s, then turn off 0.5s) in sequence.

```
for(int i=2;i<=11;i++)  
{  
    digitalWrite(i,HIGH);
```

```
delay(500);  
digitalWrite(i,LOW);  
delay(500);  
}  
}
```

Refer to [Part 1-1.2 Digital Write](#) for more details about controlling the LED by using digital pins.

Phenomenon Picture

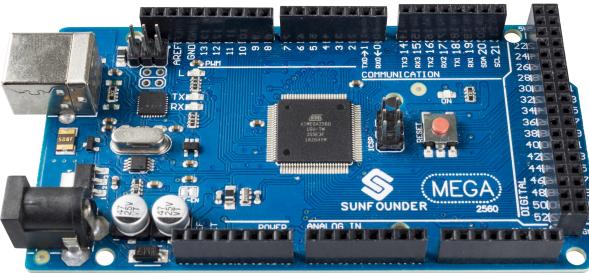
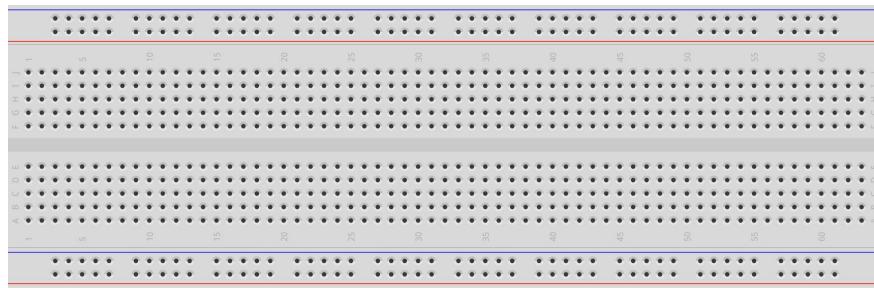


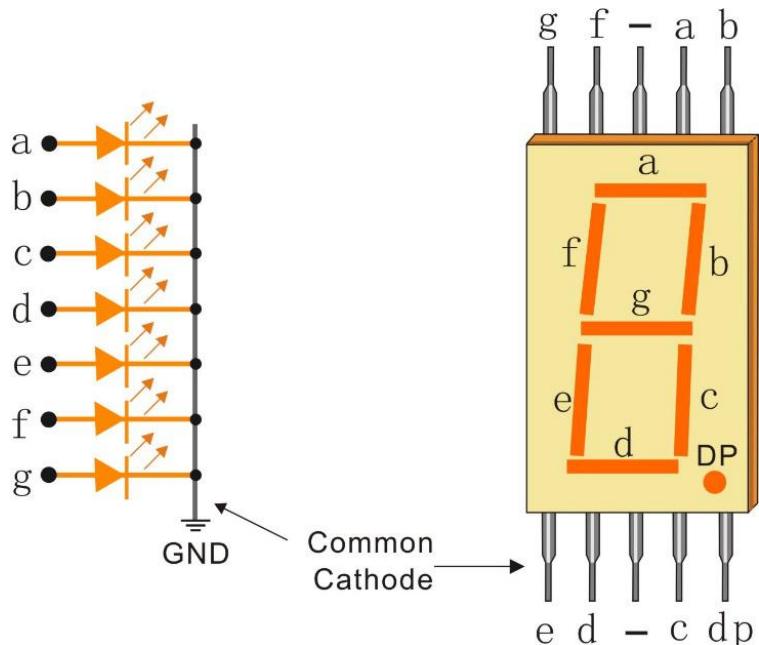
2.5 7-Segment Display

Overview

In this lesson, you will learn something about 7-Segment Display. 7-Segment Display has so many advantages that it is widely used in electrical equipments, especially in household appliances that display numerical information, such as display, air conditioner, water heater, refrigerator and so on. LEDs on the 7-Segment Display emit light by the input of different electrical signals to the different pins of it. The numerical information it can display includes time, date, temperature and so on.

Components Required

1 * 7-Segment Display 	8 * 220 ohm resistor 	Several Jumper Wires 
1 * Mega 2560 Board 	1 * Breadboard 	



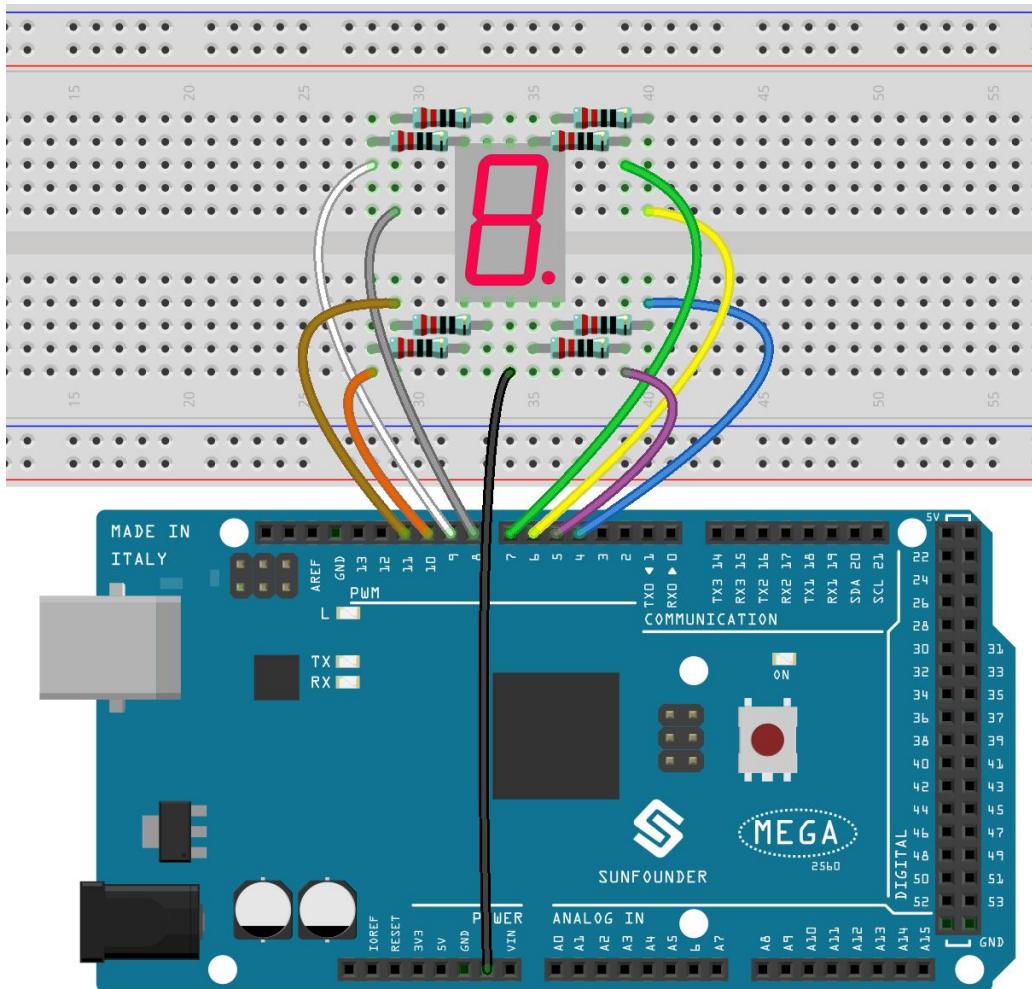
Component Introduction

A 7-segment display is an 8-shaped component which packages 7 LEDs. Each LED is called a segment – when energized, one segment forms part of a numeral to be displayed.

There are two types of pin connection: Common Cathode (CC) and Common Anode (CA). As the name suggests, a CC display has all the cathodes of the 7 LEDs connected when a CA display has all the anodes of the 7 segments connected. In this kit, we use the former.

Each of the LEDs in the display is given a positional segment with one of its connection pins led out from the rectangular plastic package. These LED pins are labeled from "a" through to "g" representing each individual LED. The other LED pins are connected together forming a common pin. So by forward biasing the appropriate pins of the LED segments in a particular order, some segments will brighten and others stay dim, thus showing the corresponding character on the display.

Fritzing Circuit

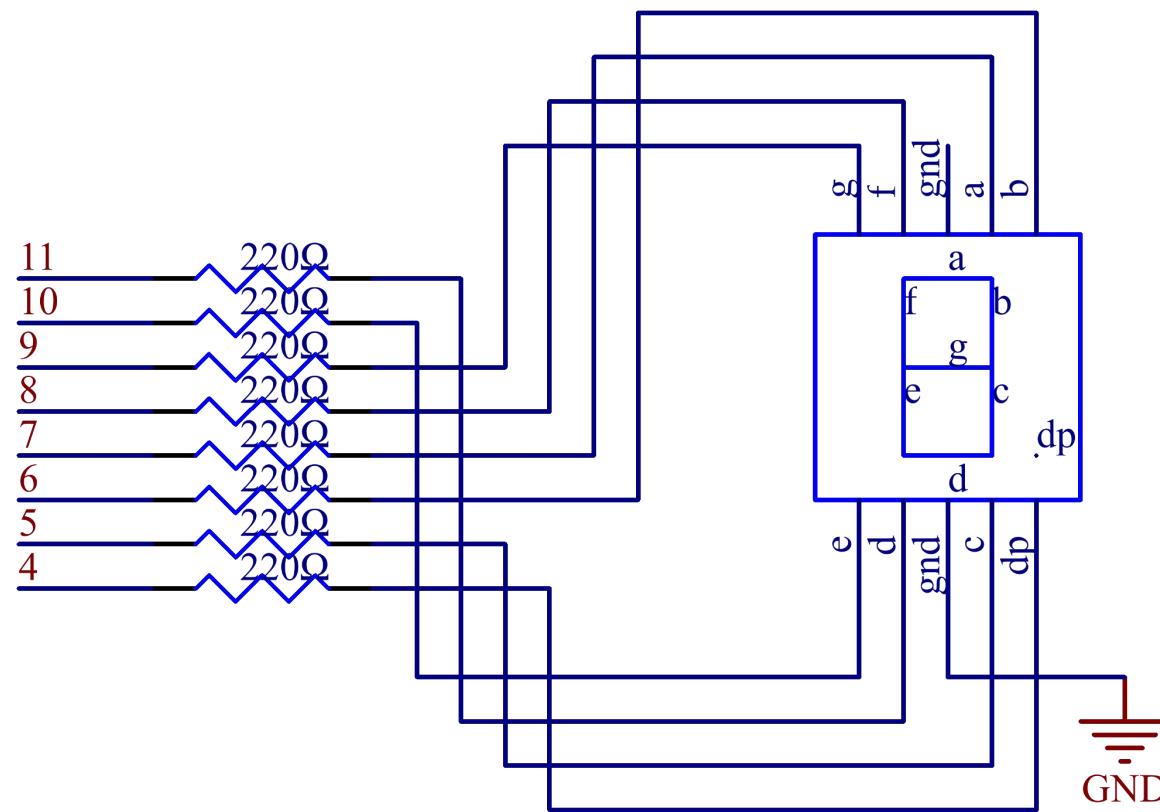


Connect each of pin a-g of the 7-Segment Display to one 220ohm current limiting resistor respectively and then to pin 4–11. GND connects to GND.

The wiring between the 7-segment display and the MEGA2560 board as shown below :

7-Segment	Mega2560 Board
a	7
b	6
c	5
d	11
e	10
f	8
g	9
dp	4
" - "	GND

Schematic Diagram



Code

NOTE: A small portion of the codes are omitted. Please check the complete codes by opening 2.5_7segment.ino in path of sunfounder_vincent_kit_for_arduino\code\2.5_7segment in Arduino IDE.

```
const int a=7; //a of 7-segment attach to digital pin 7
const int b=6; //b of 7-segment attach to digital pin 6
const int c=5; //c of 7-segment attach to digital pin 5
const int d=11;//d of 7-segment attach to digital pin 11
const int e=10;//e of 7-segment attach to digital pin 10
const int f=8;//f of 7-segment attach to digital pin 8
const int g=9;//g of 7-segment attach to digital pin 9
const int dp=4;//dp of 7-segment attach to digital pin 4

void setup()
{
    //loop over thisPin from 4 to 11 and set them all to output
    for(int thisPin = 4;thisPin <= 11;thisPin++)
    {
        pinMode(thisPin,OUTPUT);
    }
    digitalWrite(dp,LOW);//turn the dp of the 7-segment off
```

```
}

void loop()
{
    digital_1();
    delay(1000);
    digital_2();
    delay(1000);
    digital_3();
    delay(1000);
    //...
}

void digital_1(void) //display 1 to the 7-segment
{
    digitalWrite(c,HIGH); //turn the c of the 7-segment on
    digitalWrite(b,HIGH); //turn the b of the 7-segment on
    for(int j = 7;j <= 11;j++)
        digitalWrite(j,LOW);
}

void digital_2(void) //display 2 to the 7-segment
{
```

```
//...
}
void digital_3(void) //diaplay 3 to the 7-segment
{
    //...
}
//...
```

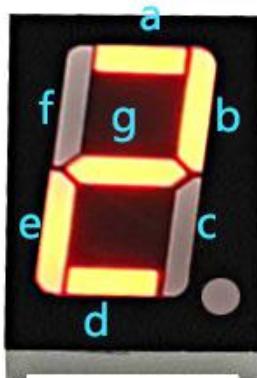
Once upload the codes, you can see the 7-segment display displaying 1, 2, 3, 4, 5, 6, 7, 8, 9, A, b, C, d, E, F in sequence.

Code Analysis

Take the pin numbers on 7-segment as names, and declare the pins on the Mega2560 board.

```
const int a=7; //a of 7-segment attach to digital pin 7
const int b=6; //b of 7-segment attach to digital pin 6
const int c=5; //c of 7-segment attach to digital pin 5
const int d=11;//d of 7-segment attach to digital pin 11
const int e=10;//e of 7-segment attach to digital pin 10
const int f=8;//f of 7-segment attach to digital pin 8
const int g=9;//g of 7-segment attach to digital pin 9
const int dp=4;//dp of 7-segment attach to digital pin 4
```

Install a series of subfunctions to package the level state at each block during the number display of the 7-segment. For example, when the character 「2」 is displayed, the block F and the block c are turn off; the other blocks are lit up.



First we need to know how it looks like when display the numeral **2** on the 7-Segment display. It's actually the segments a, b, d, e and g are power on, which generates the display of **2**. In programming, pins connected to these segments are set High level when c and f are Low level. Here we use a *for()* statement to set these pins as High level respectively (the braces after *for()* are deleted as there is only one line). Connect pin dp to pin 4; it's already defined as LOW in *setup()*.

After running this part, the 7-segment will display **2**. Similarly, the display of other characters are the same. Since the letters b and d in upper case, namely **B** and **D**, would look the same with **8** and **0** on the display, they are displayed in lower case instead.

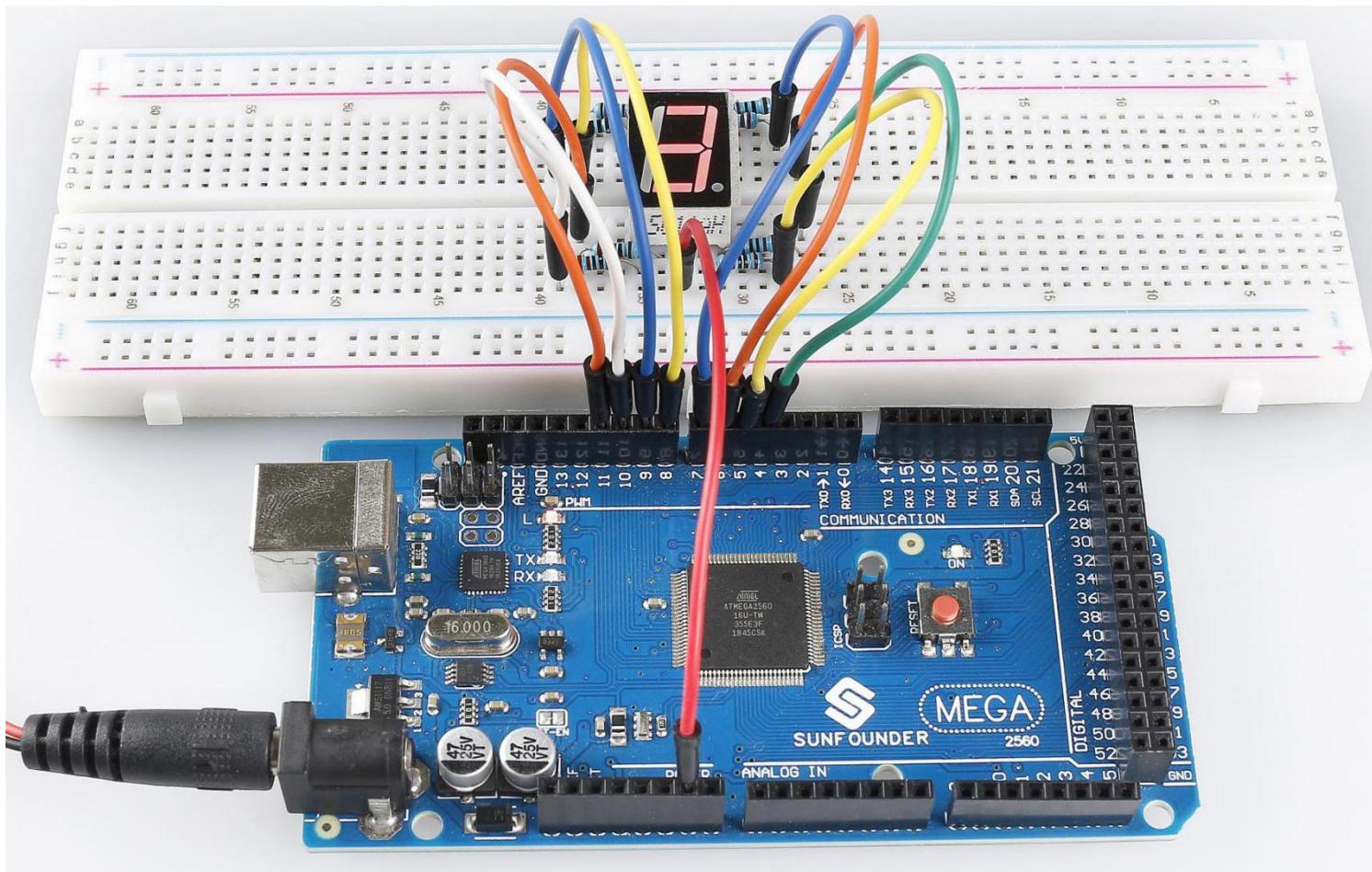
```
...
void digital_2(void) //diaplay 2 to the 7-segment
{
```

```
digitalWrite(b,HIGH);
digitalWrite(a,HIGH);
for(int j = 9;j <= 11;j++)
    digitalWrite(j,LOW);
digitalWrite(c,LOW);
digitalWrite(f,LOW);
}
...
...
```

In loop(), call the function that displays the number.

```
void loop()
{
    digital_1(); //display 1 to the 7-segment
    delay(1000); //wait for a second
    digital_2(); //display 2 to the 7-segment
    delay(1000); //wait for a second
    digital_3(); //display 3 to the 7-segment
    //...
}
```

Phenomenon Picture

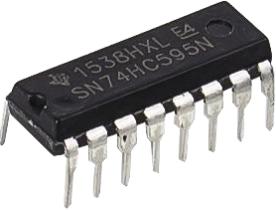
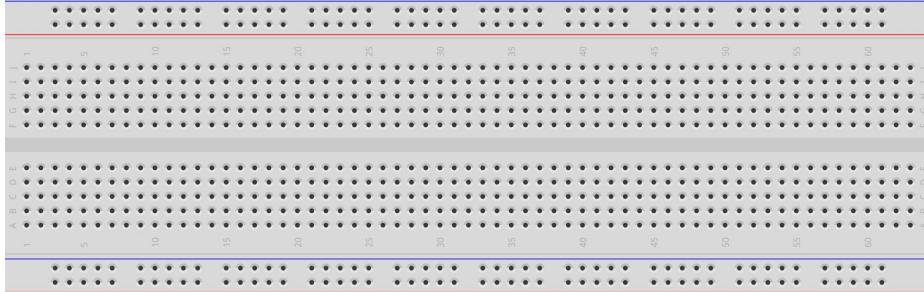


2.6 74HC595

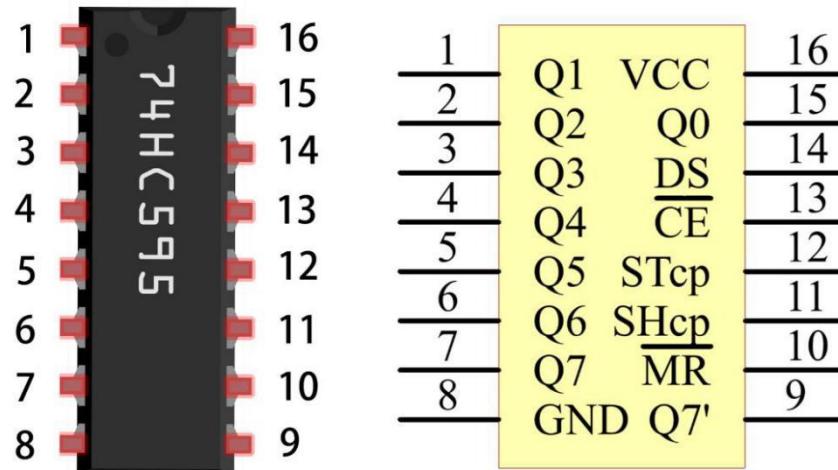
Overview

In this lesson, you will learn how to use 74HC595. 74HC595 consists of an 8-bit shift register and a storage register with three-state parallel outputs. It converts serial input into parallel output so you can save IO ports of an MCU.

Components Required

1 * 74HC595 	8 * LED 	8 * 220 ohm resistor 	Several Jumper Wires 
1 * Mega 2560 Board 		1 * Breadboard 	

Component Introduction



When MR (pin10) is high level and OE (pin13) is low level, data is input in the rising edge of SHcp and goes to the memory register through the rising edge of SHcp. If the two clocks are connected together, the shift register is always one pulse earlier than the memory register. There is a serial shift input pin (Ds), a serial output pin (Q) and an asynchronous reset button (low level) in the memory register. The memory register outputs a Bus with a parallel 8-bit and in three states. When OE is enabled (low level), the data in memory register is output to the bus.

Pins of 74HC595 and their functions:

Q0-Q7: 8-bit parallel data output pins, able to control 8 LEDs or 8 pins of 7-segment display directly.

Q7': Series output pin, connected to DS of another 74HC595 to connect multiple 74HC595s in series

MR: Reset pin, active at low level;

SHcp: Time sequence input of shift register. On the rising edge, the data in shift register moves successively one bit, i.e. data in Q1 moves to Q2, and so forth. While on the falling edge, the data in shift register remain unchanged.

STcp: Time sequence input of storage register. On the rising edge, data in the shift register moves into memory register.

OE: Output enable pin, active at low level.

DS: Serial data input pin

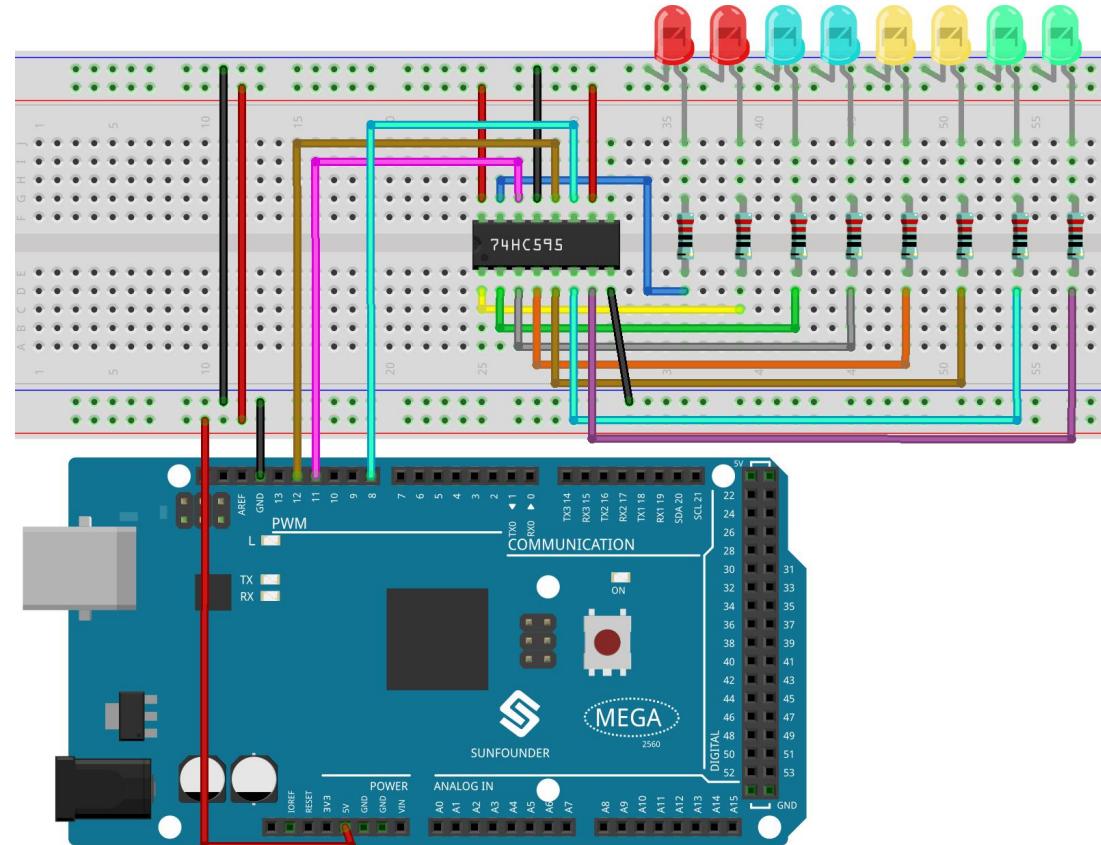
VCC: Positive supply voltage

GND: Ground

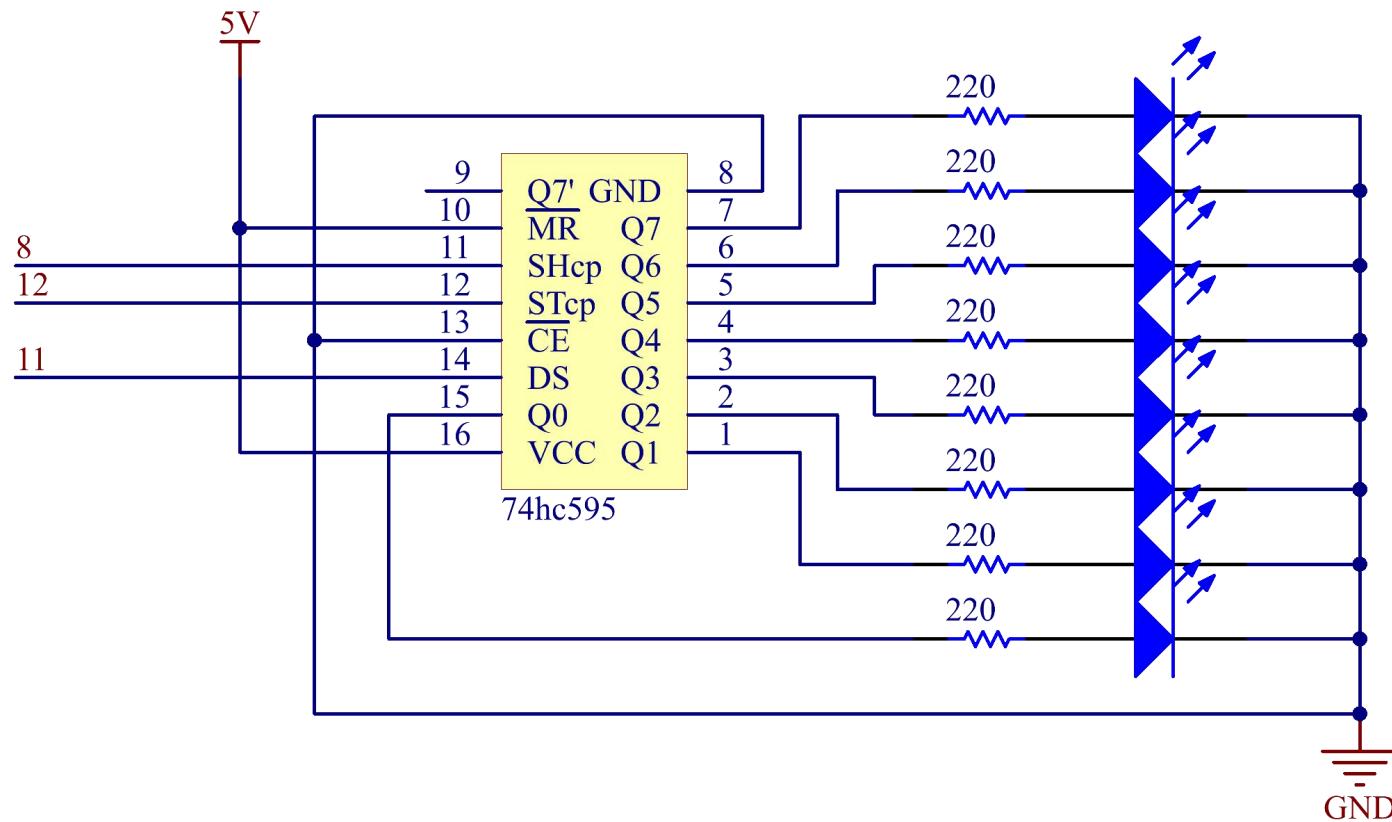
Fritzing Circuit

In this example, we use 74HC595 to control LED. Give each data output pin (Q0-Q7) a 220 ohm resistor then connect them to LED. The wiring diagram is as follows:

74HC595	Mega Board
MR(10)	5V
SHcp(11)	8
STcp(12)	12
OE(13)	GND
DS(14)	11
VCC(16)	5V
GND(8)	GND



Schematic Diagram



Code

```
const int STcp = 12;//Pin connected to ST_CP of 74HC595
const int SHcp = 8;//Pin connected to SH_CP of 74HC595
const int DS = 11;//Pin connected to DS of 74HC595
int datArray[] = {B00000000, B00000001, B00000011, B00000111, B00001111, B00011111, B00111111, B01111111,
B11111111};

void setup()
{
    //set pins to output
    pinMode(STcp,OUTPUT);
    pinMode(SHcp,OUTPUT);
    pinMode(DS,OUTPUT);
}

void loop()
{
    for(int num = 0; num <=8; num++)
    {
        digitalWrite(STcp,LOW); //ground ST_CP and hold low for as long as you are transmitting
        shiftOut(DS,SHcp,MSBFIRST,datArray[num]);
        //return the latch pin high to signal chip that it
    }
}
```

```
//no longer needs to listen for information  
digitalWrite(STcp,HIGH); //pull the ST_CPST_CP to save the data  
delay(500); //wait for a second  
}  
}
```

When you finish uploading the codes to the Mega2560 board, you can see the LEDs turning on one after another.

Code Analysis

Declare an array, store several 8 bit binary numbers that are used to change the working state of the eight LEDs controlled by 74HC595.

```
int dataArray[] = {B00000000, B00000001, B00000011, B00000111, B00001111, B00111111, B01111111,  
B11111111};
```

Set STcp to low level first and then high level. It will generate a rising edge pulse of STcp.

```
digitalWrite(STcp,LOW);
```

shiftOut() is used to shift out a byte of data one bit at a time, which means to shift a byte of data in dataArray[num] to the shifting register with the DS pin. MSBFIRST means to move from high bits.

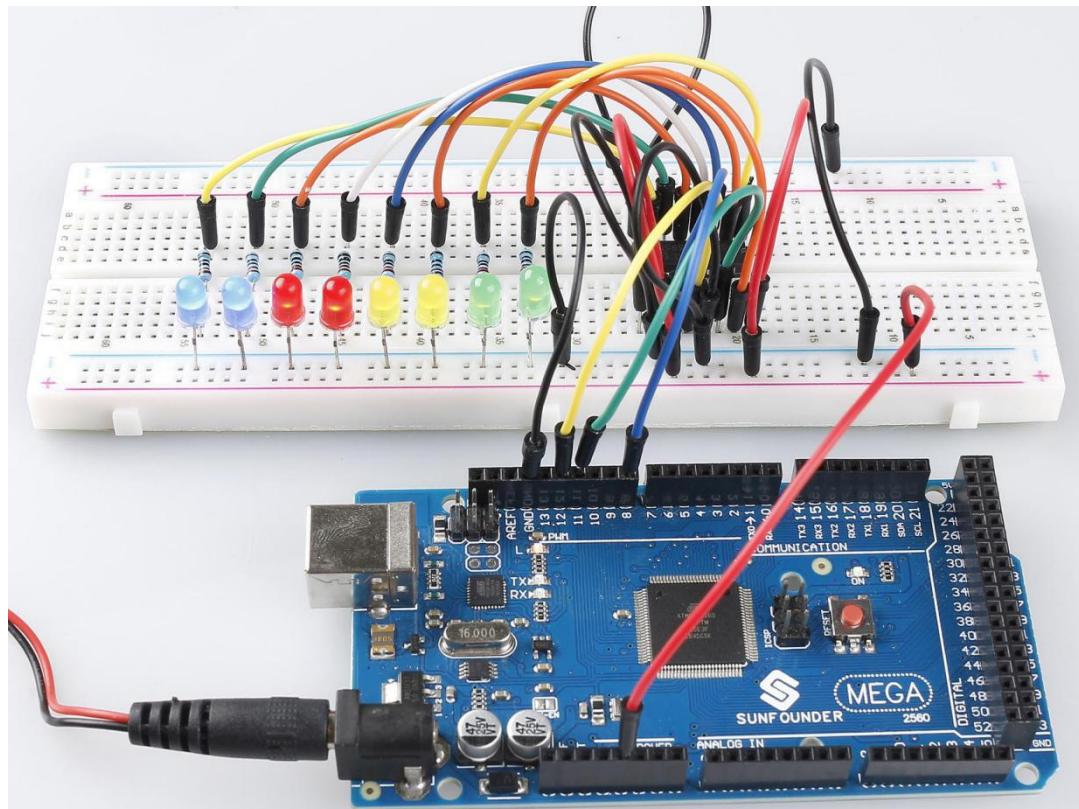
```
shiftOut(DS,SHcp,MSBFIRST,dataArray[num]);
```

After digitalWrite(STcp,HIGH) is run, the STcp will be at the rising edge. At this time, the data in the shift register will be moved to the memory register.

```
digitalWrite(STcp,HIGH);
```

A byte of data will be transferred into the memory register after 8 times. Then the data of memory register are output to the bus (Q0-Q7). For example, shiftout 「B00000001」 will light up the LED controlled by Q0 and turn off the LED controlled by Q1~Q7.

Phenomenon Picture



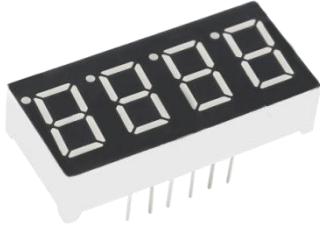
2.7 4-Digital 7-Segment Display

Overview

In this lesson, you will learn about the 4-Digital 7-Segment Display. It consists of four 7-segment displays working together so as to display 4 digit numbers.

Components Required

1 * 4-Digital 7-Segment Display



8 * 220 ohm resistor



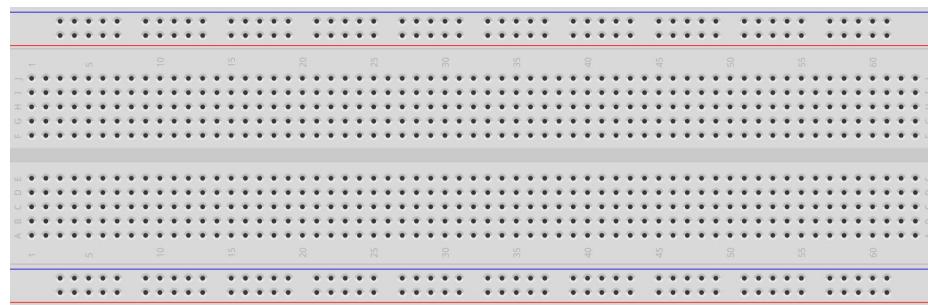
Several Jumper Wires



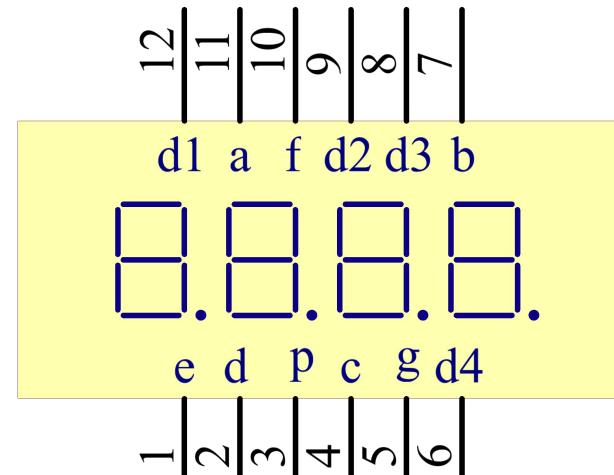
1 * Mega 2560 Board



1 * Breadboard

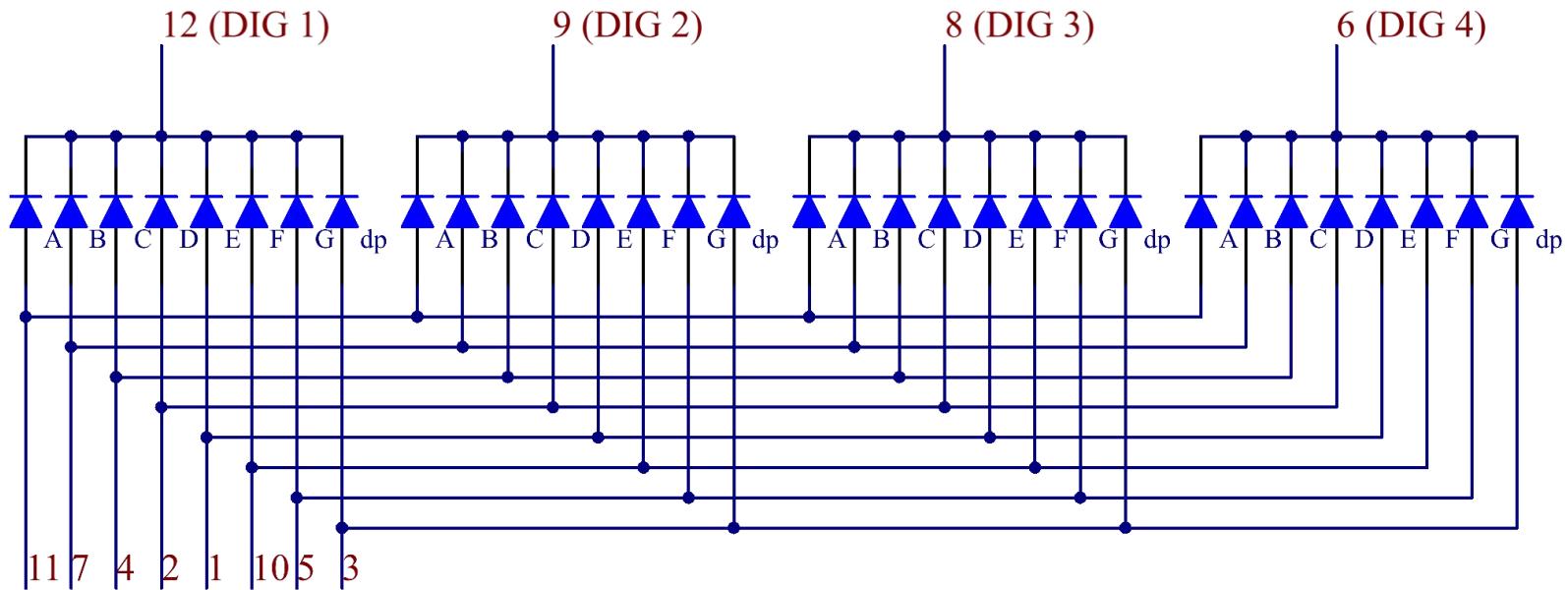


Component Introduction



The 4-digital 7-segment display works independently. It uses the principle of human visual persistence to quickly display the characters of each 7-segment in a loop to form continuous strings.

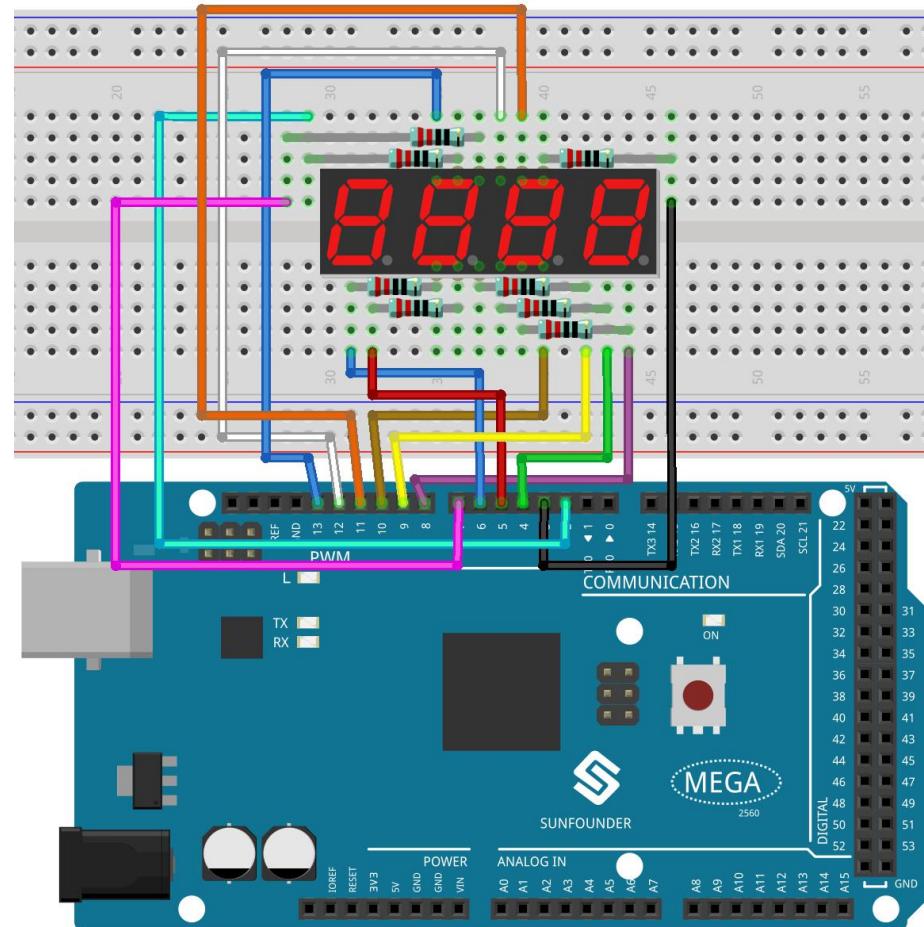
For example, when "1234" is displayed on the display, "1" is displayed on the first 7-segment, and "234" is not displayed. After a period of time, the second 7-segment shows "2", the 1st 3th 4th of 7-segment does not show, and so on, the four digital display show in turn. This process is very short (typically 5ms), and because of the optical afterglow effect and the principle of visual residue, we can see four characters at the same time.



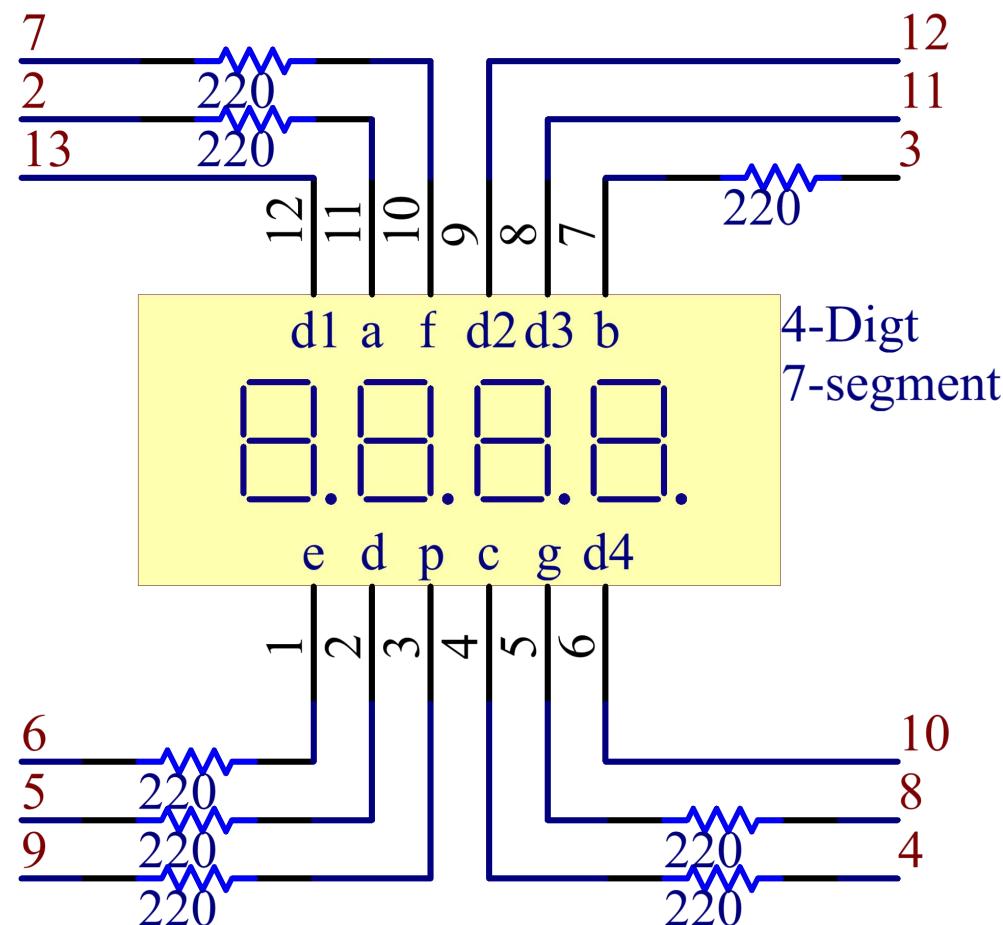
Fritzing Circuit

The wiring between the 4-digit 7-segment display and the Mega 2560 Board board is as shown below:

4-Digit Display	Mega 2560 Board
A(11)	2
B(7)	3
C(4)	4
D(2)	5
E(1)	6
F(10)	7
G(5)	8
dp(3)	9
D1(12)	13
D2(9)	12
D3(8)	11
D4(6)	10



Schematic Diagram

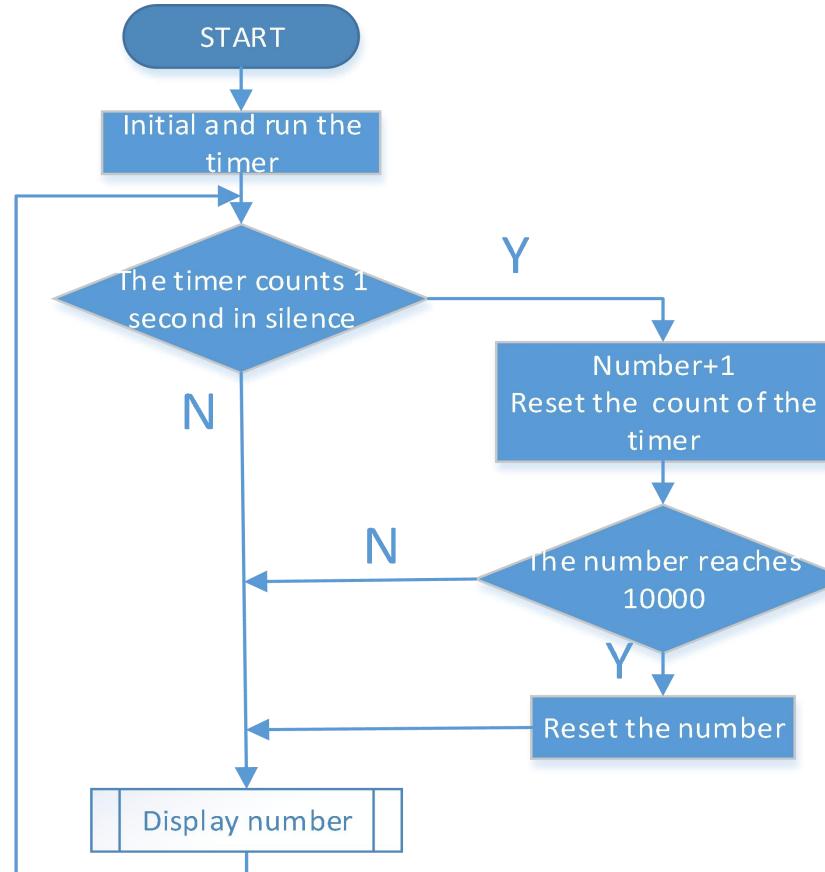


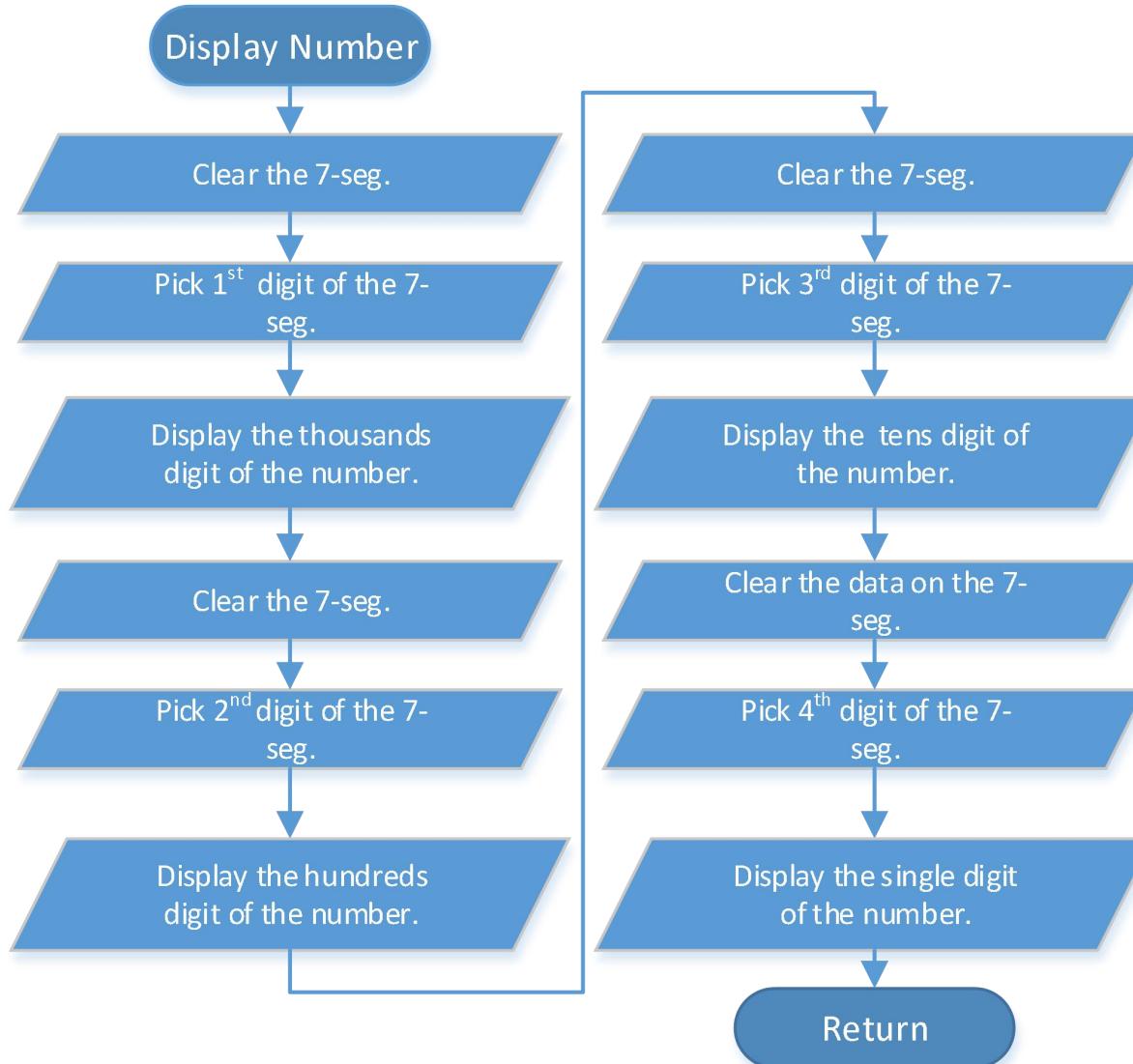
Code

Note: Open 2.7_4digitalSegment in Arduino IDE to check the codes.

The code uses TimerOne.h library. Refer to [Part 4 - 4.1 Add Libraries](#) to import library.

Code Analysis





There are two points needing your attention:

① Because every segment display works independently in the 4-Digital 7-Segment Display, the principle of visual persistence is applied to quickly display every 7 segment character in turn to form a continuous character string.

Refer to [Part 2 - 2.5 7-Segment Display](#) to check the details of the number display of the 4-Digital 7-Segment Display.

② In this example, a library TimerOne.h is used to realize the function of counting.

```
#include "TimerOne.h"
```

Library Functions:

```
void initialize(long microsenconds=1000000)
```

You must call this method first to use any of the other methods. You can optionally specify the timer's period here (in microseconds), by default it is set at 1 second. **Note that this breaks analogWrite() for digital pins 9 and 10 on Arduino.**

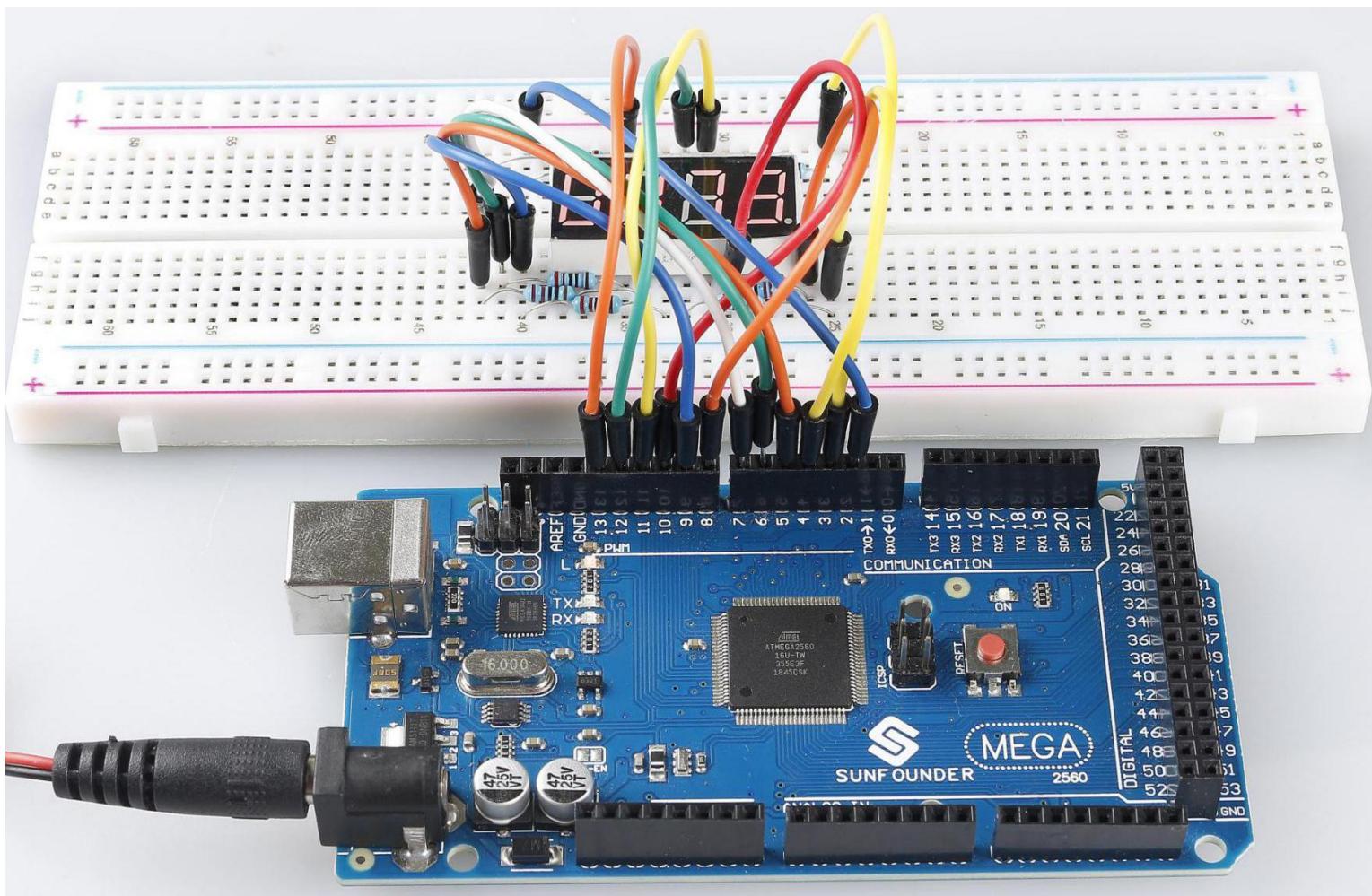
```
void attachInterrupt(void (*isr)(), long microseconds=-1);
```

Calls a function at the specified interval in microseconds. Be careful about trying to execute too complicated of an interrupt at too high of a frequency, or the CPU may never enter the main loop and your program will 'lock up'. Note that you can optionally set the period with this function if you include a value in microseconds as the last parameter when you call it.

```
void detachInterrupt();
```

Disables the attached interrupt.

Phenomenon Picture



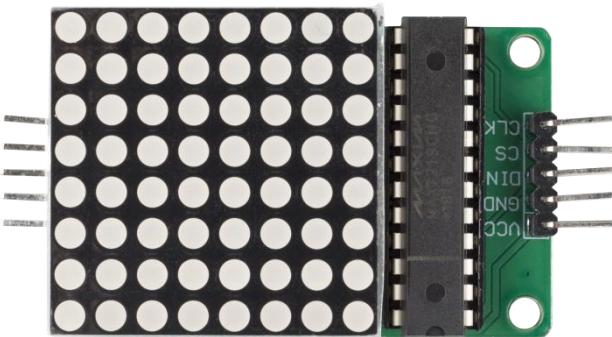
2.8 LED Matrix Module

Overview

In this lesson, you will learn about LED Matrix Module. LED Matrix Module uses the MAX7219 driver to drive the 8 x 8 LED Matrix.

Components Required

1 * MAX7219 LED Matrix Module



1 * Mega 2560 Board

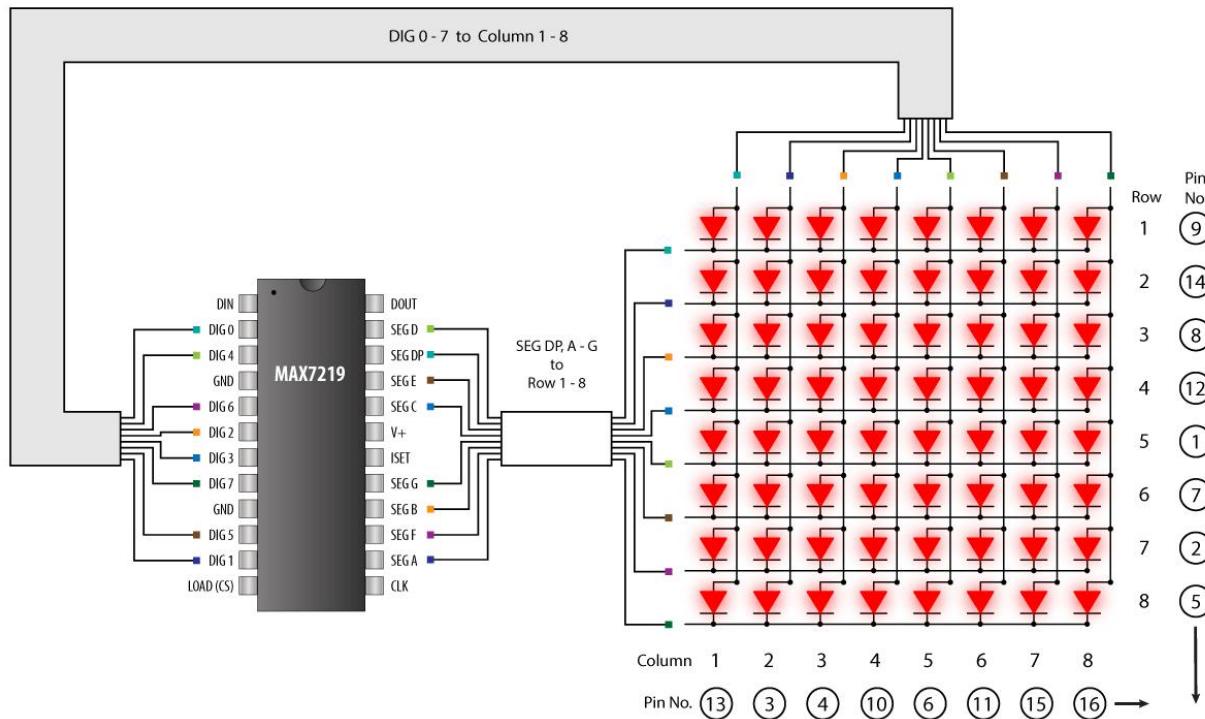


Several Jumper Wires



Component Introduction

The 64 LEDs are driven by 16 output pins of the IC. The maximum number of LEDs light up at the same time is actually eight. The LEDs are arranged as 8×8 set of rows and columns. So the MAX7219 activates each column for a very short period of time and at the same time it also drives each row. So by rapidly switching through the columns and rows the human eye will only notice a continuous light.

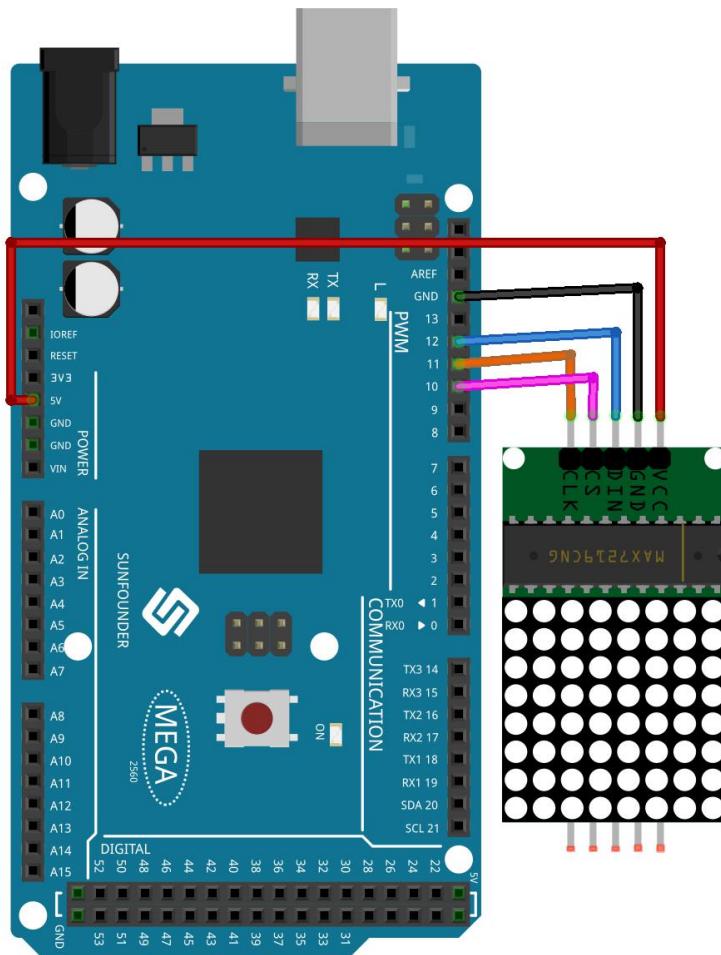


MAX7219

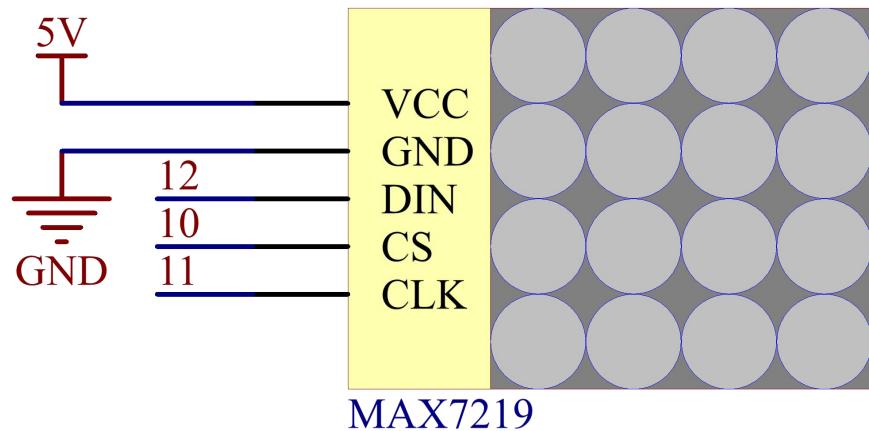
These integrated circuits from Maxim are for driving either 64 individual LED's, or up to 8 digits of 7-segment displays. The drivers implement a SPI compatible slave interface that can be controlled from the Arduino using only 3 of the digital output pins.

Fritzing Circuit

In this example, we get the VCC pin of MAX7219 connected to 5V, GND to ground, DIN to digital pin 12, CS to digital pin 10, CLK to digital pin 11.



Schematic Diagram



Code

The codes use the library LedControl.h, and please refer to [Part 4 - 4.1 Add Libraries](#) to add in the library to Arduino IDE.

```
#include "LedControl.h"
LedControl lc = LedControl(12, 11, 10, 1);
void setup() {
    lc.shutdown(0, false);
    lc.setIntensity(0, 5);
    lc.clearDisplay(0);
}
void loop() {
    displayDot();  displayCol();
    displayRow(); displayPic();
}
void displayDot() {
    lc.clearDisplay(0);
    delay(100);
    for (int row = 0; row < 8; row++){
        for (int col = 0; col < 8; col++){
            lc.setLed(0, row, col, true);
            delay(50);
        }
    }
}
```

```
    }
}

void displayCol() {
    lc.clearDisplay(0);
    delay(100);
    byte data = B01100110;
    for (int col = 0; col < 8; col++) {
        lc.setColumn(0, col, data);
        delay(100);
    }
}

void displayRow() {
    lc.clearDisplay(0);
    delay(100);
    byte data = B10011001;
    for (int row = 0; row < 8; row++) {
        lc.setRow(0, row, data);
        delay(100);
    }
}

void displayPic() {
    lc.clearDisplay(0);
```

```
delay(100);
byte pic[8] = {
    B00000000,
    B01100110,
    B11111111,
    B11111111,
    B01111110,
    B00111100,
    B00011000,
    B00000000};
for (int col = 0; col < 8; col++)
{l.setColumn(0, col, pic[col]);}
delay(2000);
}
```

After the codes are uploaded, you can see that the LEDs turn on in the sequence of a column, a row or a dot or there is a picture appearing on the LED matrix.

Code Analysis

By calling the library LedControl.h, you can easily use the LED matrix.

```
#include "LedControl.h"
```

Library Functions:

LedControl(int dataPin,int clockPin,int csPin,int numDevices)

Create an instance of type LedControl through which we talk to the MAX7219 devices. The initialization of an LedControl takes 4 arguments.

dataPin,clockPin,csPin: The first 3 arguments are the pin-numbers on the Arduino that are connected to the MAX7219. You are free to choose any of the digital IO-pins on the arduino, but since some of the pins are also used for serial communication or have a led attached to them its best to avoid pin 0,1 and 13.

numDevices: The fourth argument is the number of cascaded MAX7219 devices you're using with this LedControl. The library can address up to 8 devices from a single LedControl-variable.

void shutdown(int addr, bool b)

addr: The address of the display to control.

b: If true the device goes into power-down mode. If false device goes into normal operation.

void setIntensity(int addr, int intensity)

The method lets you control brightness in 16 discrete steps. Larger values make the display brighter up to the maximum of 15.

addr: The address of the display to control.

intensity: the brightness of the display. Only values between 0(darkest) and 15(brightest) are valid.

void clearDisplay(int addr)

All LEDs off after this one.

addr: The address of the display to control.

void setLed(int addr, int row, int col, boolean state)

Set the status of a single Led.

addr: The address of the display to control.

row: The row of the Led (0..7).

col: The column of the Led (0..7).

state: If true the led is switched on, if false it is switched off.

void setRow(int addr, int row, byte value)

Set all 8 LEDs in a row to a new state.

addr: The address of the display to control.

row: Row which is to be set (0..7).

value: Each bit set to 1 will light up the corresponding Led.(e.g. B01000000 will light up the 2nd).

void setColumn(int addr, int col, byte value)

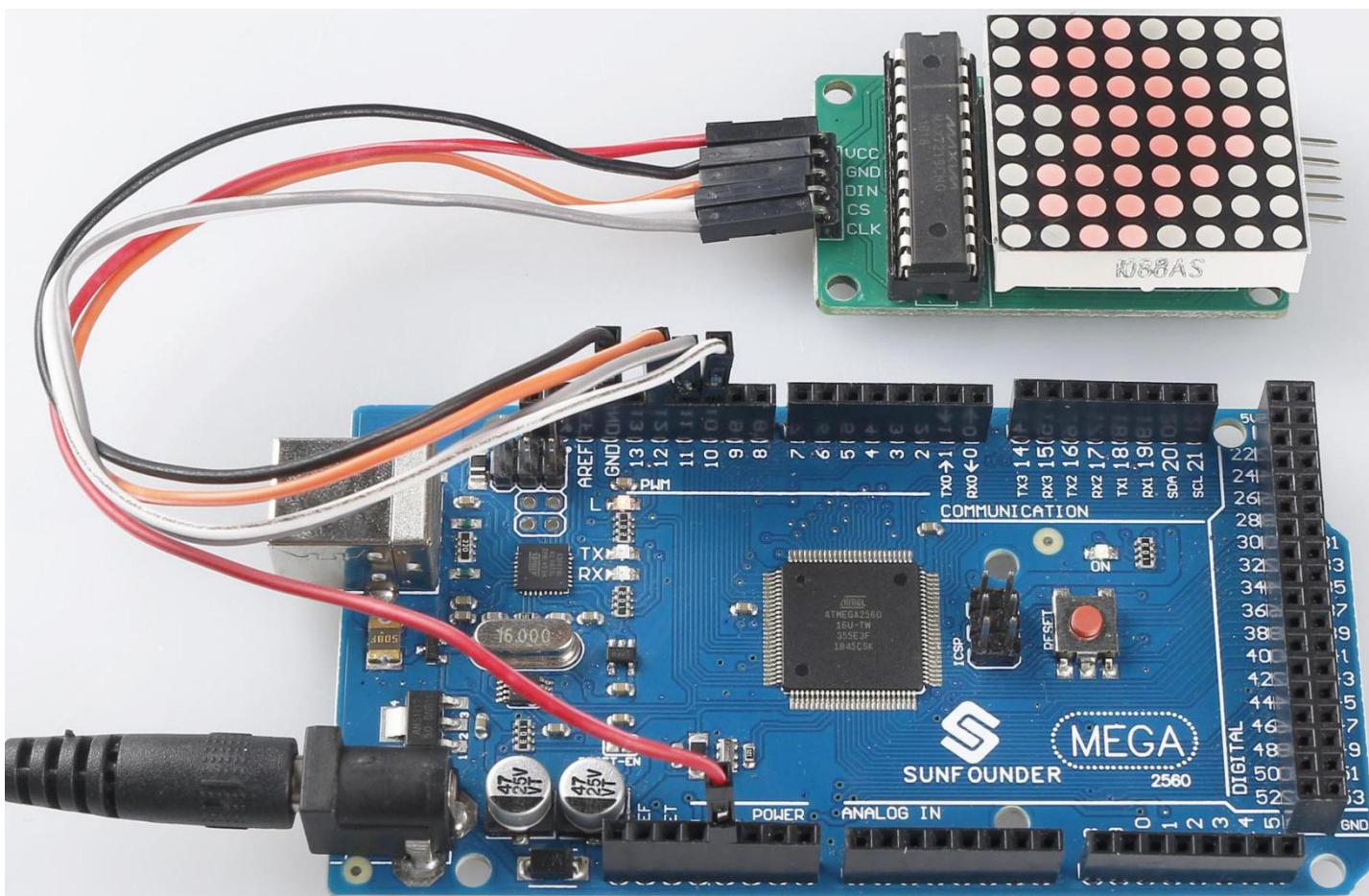
Set all 8 LEDs in a row to a new state.

addr: The address of the display to control.

col: Column which is to be set (0..7).

value: Each bit set to 1 will light up the corresponding Led.(e.g. B01000000 will light up the 2nd).

Phenomenon Picture



2.9 I2C LCD1602 Module

Overview

In this lesson, you will learn about LCD1602. LCD1602, or 1602 character-type liquid crystal display, a kind of dot matrix module to show letters, numbers, characters and so on.

Components Required

1 * I2C LCD1602 Module



1 * Mega 2560 Board



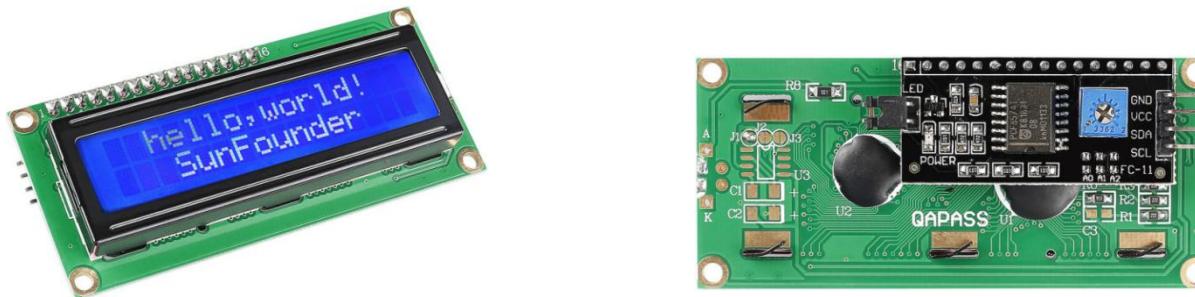
Several Jumper Wires



Component Introduction

It's composed of 5x7 or 5x11 dot matrix positions; each position can display one character. There's a dot pitch between two characters and a space between lines, thus separating characters and lines. The number 1602 means on the display, 2 rows can be showed and 16 characters in each.

As we all know, though LCD and some other displays greatly enrich the man-machine interaction, they share a common weakness. When they are connected to a controller, multiple IOs will be occupied of the controller which has no so many outer ports. Also it restricts other functions of the controller. Therefore, LCD1602 with an I2C bus is developed to solve the problem.

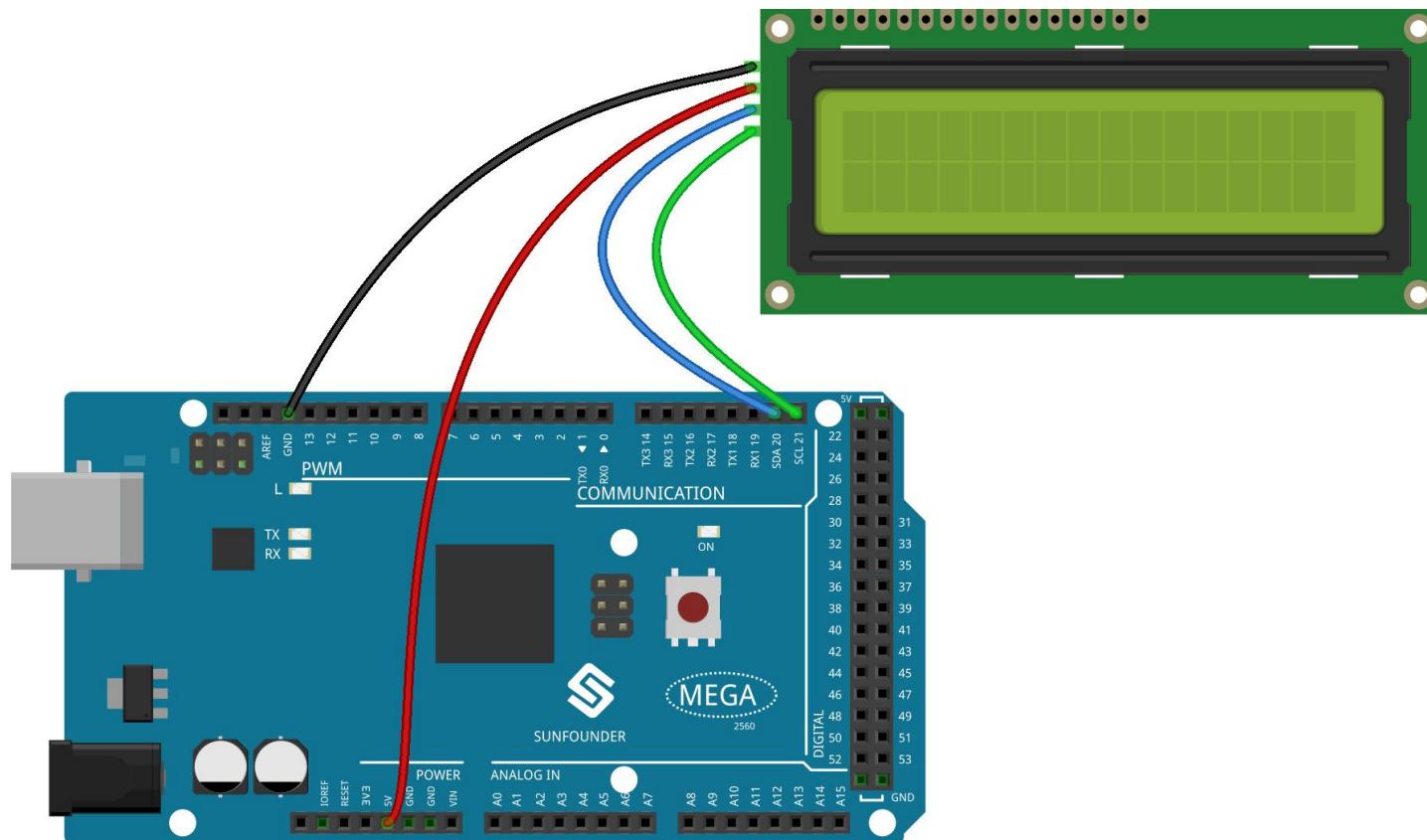


I2C communication

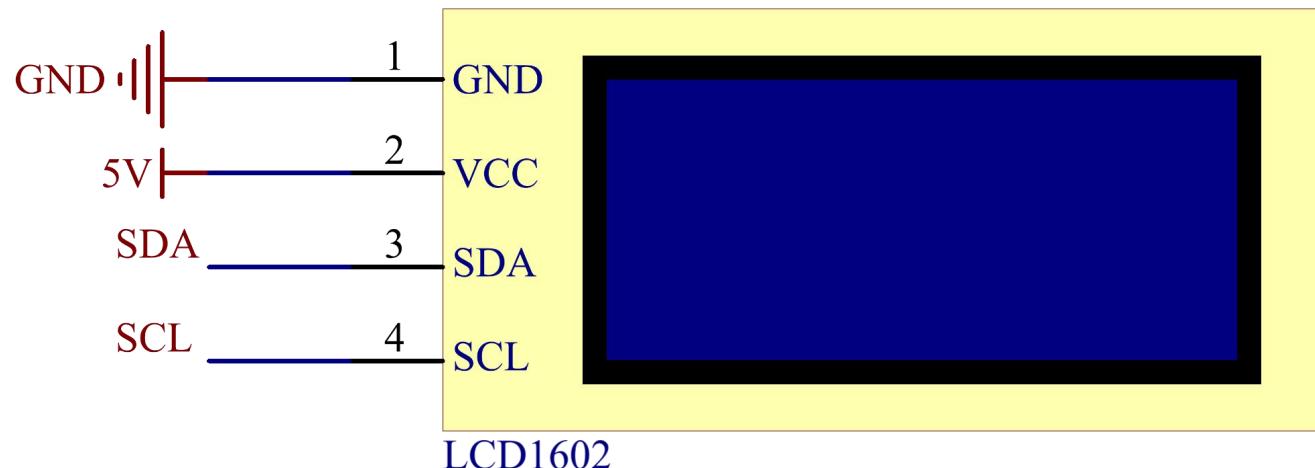
I2C(Inter-Integrated Circuit) bus is a very popular and powerful bus for communication between a master device (or master devices) and a single or multiple slave devices. I2C main controller can be used to control IO expander, various sensors, EEPROM, ADC/DAC and so on. All of these are controlled only by the two pins of host, the serial data (SDA) line and the serial clock line(SCL).

Fritzing Circuit

In this example, we will get the first pin GND of LCD1602 connected to GND, the second pin VCC to 5V, the third pin SDA to the pin SDA 20 and the forth pin SCL to the pin SCL 21.



Schematic Diagram



Note: The SDA and SCL of the Mega2560 board are the pins 20 and 21.

Code

The libraries Wire.h and LiquidCrystal_I2C.h are used in these codes, Wire.h is built in Arduino, but LiquidCrystal_I2C.h needs adding manually. Add Method: Refer to [Part 4 - 4.1 Add Libraries](#).

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2); // set the LCD address to 0x27 for a 16 chars and 2 line display
void setup()
{
```

```
lcd.init();           // initialize the lcd
lcd.backlight();
Serial.begin(9600);
}

void loop()
{
    // when characters arrive over the serial port...
    if (Serial.available()) {
        // wait a bit for the entire message to arrive
        delay(100);
        // clear the screen
        lcd.clear();
        // read all the available characters
        while (Serial.available() > 0) {
            char incomingByte=Serial.read();
            // skip the line-feed(ASCII 10) character
            if(incomingByte==10){break;}
            // display each character to the LCD
            lcd.print(incomingByte);
        }
    }
}
```

Upload the codes to the Mega2560 board, the content that you input in the serial monitor will be printed on the LCD.

Note: About the ASCII code and the character input in the serial monitor, please refer to Part 1-1.8 Serial Read.

Code Analysis

By calling the library LiquidCrystal_I2C.h, you can easily drive the LCD.

```
#include "LiquidCrystal_I2C.h"
```

Library Functions:

LiquidCrystal_I2C(uint8_t lcd_Addr,uint8_t lcd_cols,uint8_t lcd_rows)

Creates a new instance of the LiquidCrystal_I2C class that represents a particular LCD attached to your Arduino board.

lcd_AddR: The address of the LCD defaults to 0x27.

lcd_cols: The LCD1602 has 16 columns.

lcd_rows: The LCD1602 has 2 rows.

void init()

Initialize the lcd.

void backlight()

Turn the (optional) backlight on.

void nobacklight()

Turn the (optional) backlight off.

void display()

Turn the LCD display on.

void nodisplay()

Turn the LCD display off quickly.

void clear()

Clear display, set cursor position to zero.

void setCursor(uint8_t col,uint8_t row)

Set the cursor position to col,row.

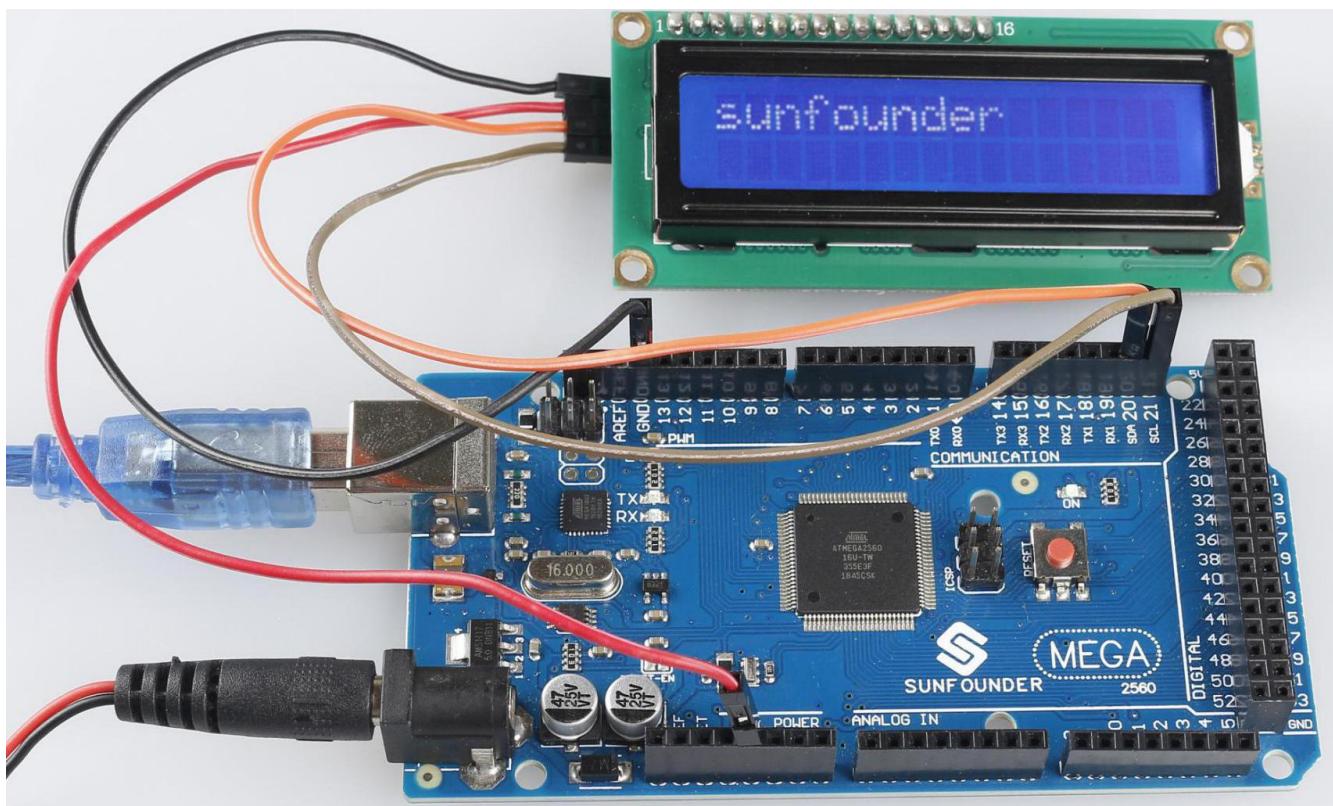
void print(data,BASE)

Prints text to the LCD.

data: The data to print (char, byte, int, long, or string).

BASE (optional): The base in which to print numbers: BIN for binary (base 2), DEC for decimal (base 10), OCT for octal (base 8), HEX for hexadecimal (base 16).

Phenomenon Picture



2.10 Active Buzzer

Overview

In this lesson, you will get to know about active buzzer. As a type of electronic buzzer with an integrated structure, active buzzer is supplied by DC power, widely used in computer, alarm, electronic toy, telephone, timer and other electronic products or voice devices.

Components Required

1 * Active Buzzer



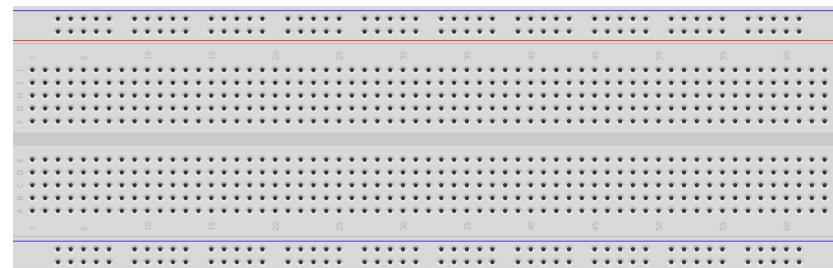
1 * Mega 2560 Board



Several Jumper Wires



1 * Breadboard



Component Introduction

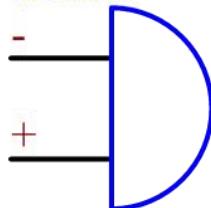
Buzzers can be categorized as active and passive ones (see the following picture). Turn the buzzer so that its pins are facing up, and the buzzer with a green circuit board is a passive buzzer, while the one enclosed with a black tape is an active one.

The difference between an active buzzer and a passive buzzer:



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

Buzzer

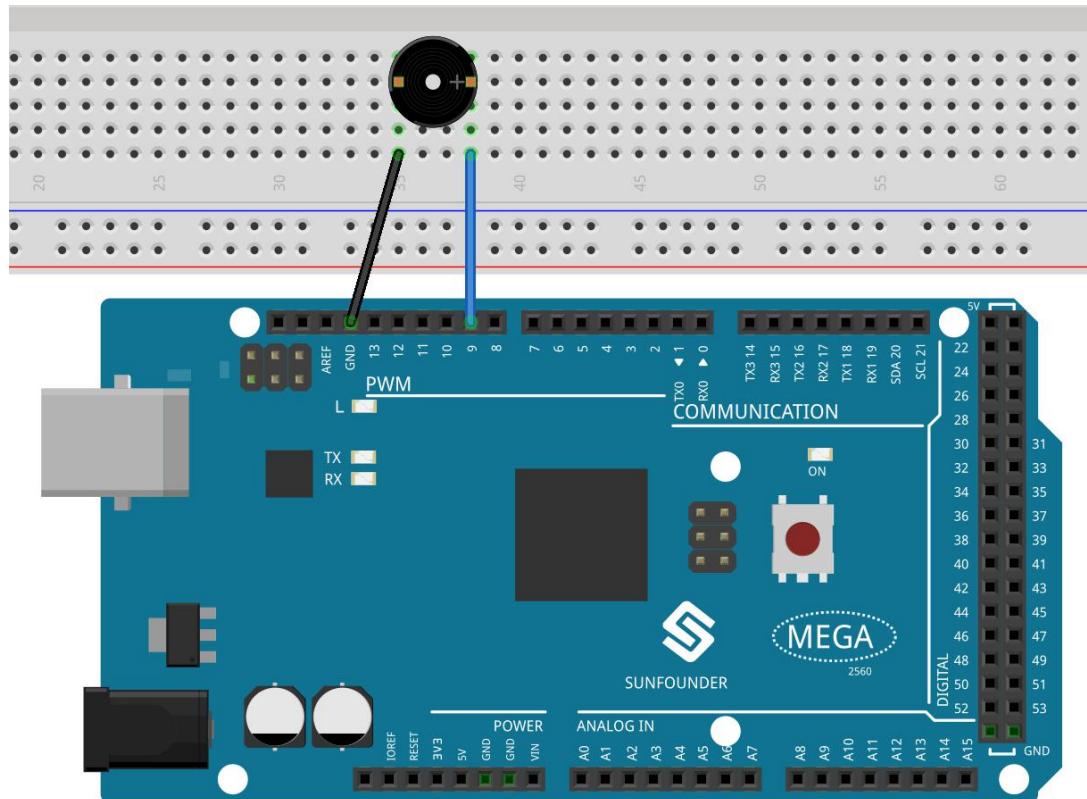


The following is the electrical symbol of a buzzer. It has two pins with positive and negative poles. With a + in the surface represents the anode and the other is the cathode.

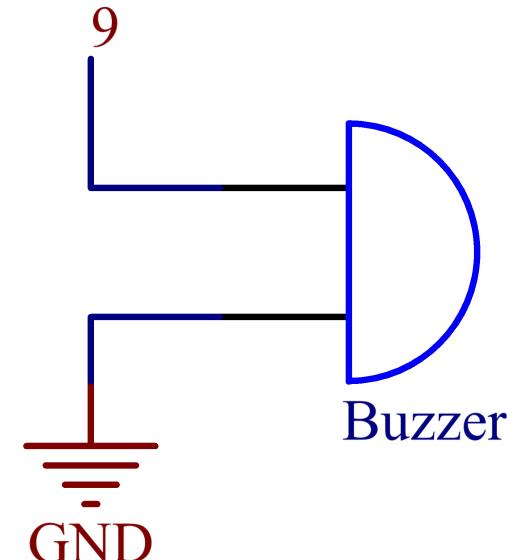
You can check the pins of the buzzer, the longer one is the anode and the shorter one is the cathode. Please don't mix them up when connecting, otherwise the buzzer will not make sound.

Fritzing Circuit

In this example, we use the digital pin 9 to drive the buzzer and extend the cathode of the Buzzer to GND and its anode to the digital pin 9.



Schematic Diagram



Code

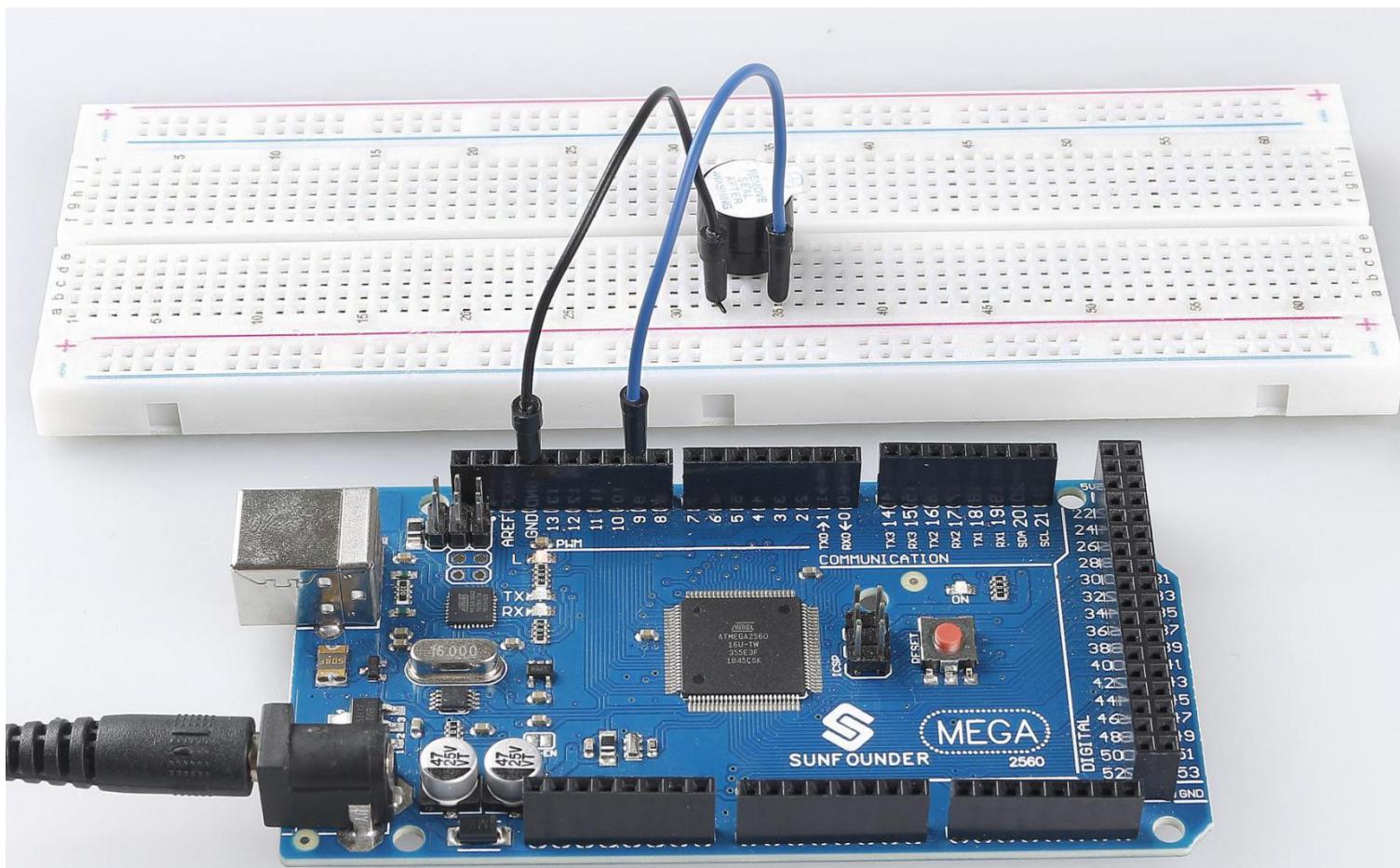
```
const int buzzerPin = 9;

void setup() {
    pinMode(buzzerPin, OUTPUT);
}

void loop() {
    digitalWrite(buzzerPin, HIGH);
    delay(300);
    digitalWrite(buzzerPin, LOW);
    delay(300);
}
```

When you finish uploading the codes to the Mega2560 board, you can hear the beep—beep emitted from the buzzer. If you want to know more about the detail code explanation, please refer to [Part 1-1.2 Digital Write](#).

Phenomenon Picture



2.11 Passive Buzzer

Overview

In this lesson, you will get to know about passive buzzer. As a type of electronic buzzer with an integrated structure, passive buzzer is supplied by DC power, widely used in computer, alarm, electronic toy, telephone, timer and other electronic products or voice devices.

Components Required

1 * Passive Buzzer



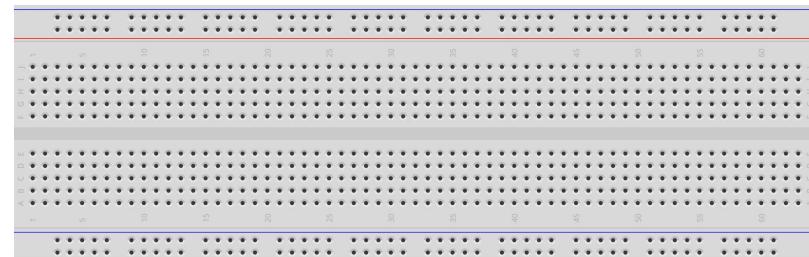
Several Jumper Wires



1 * Mega 2560 Board



1 * Breadboard



Component Introduction

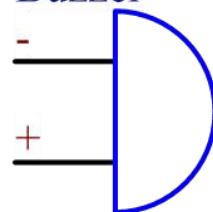
Buzzers can be categorized as active and passive ones (see the following picture). Turn the buzzer so that its pins are facing up, and the buzzer with a green circuit board is a passive buzzer, while the one enclosed with a black tape is an active one.

The difference between an active buzzer and a passive buzzer:



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

Buzzer

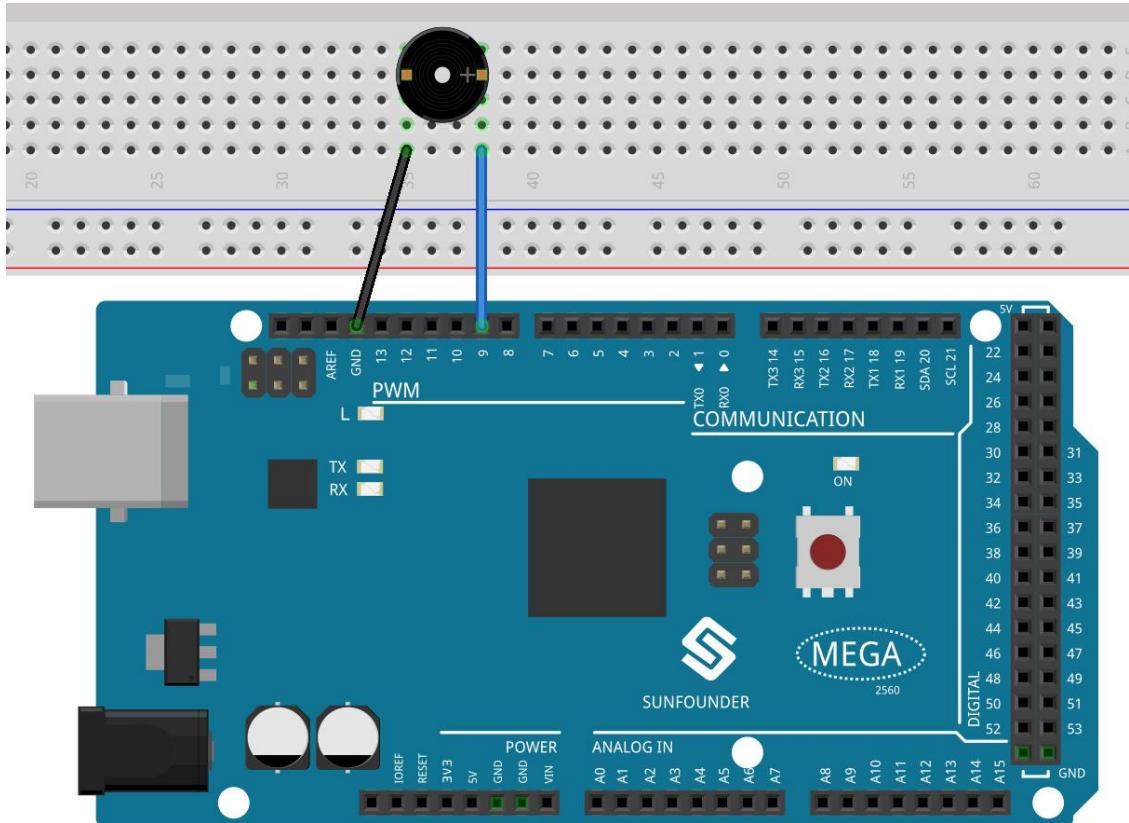


The following is the electrical symbol of a buzzer. It has two pins with positive and negative poles. With a + in the surface represents the anode and the other is the cathode.

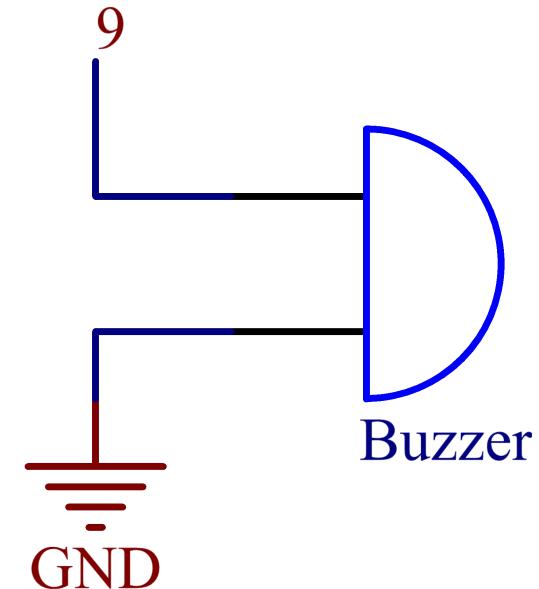
You can check the pins of the buzzer, the longer one is the anode and the shorter one is the cathode. Please don't mix them up when connecting, otherwise the buzzer will not make sound.

Fritzing Circuit

In this example, what we use to drive the buzzer is the pin 9. We get the cathode of the Buzzer to GND, and the anode to the digital pin 9.



Schematic Diagram



Code

```
#include "pitches.h"
int melody[] = {
    NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4
};
int noteDurations[] = {
    4, 8, 8, 4, 4, 4, 4, 4
};
void setup() {
    for (int thisNote = 0; thisNote < 8; thisNote++) {
        int noteDuration = 1000 / noteDurations[thisNote];
        tone(9, melody[thisNote], noteDuration);
        int pauseBetweenNotes = noteDuration * 1.30;
        delay(pauseBetweenNotes);
        noTone(9);
    }
}
void loop() {
    // no need to repeat the melody.
}
```

At the time when you finish uploading the codes to the Mega2560 board, you can hear a melody containing seven notes.

Code Analysis

There are two points needing your attention:

①tone() & noTone(): This function is used to control the sound of the passive buzzer directly and its prototype is as follows:

```
void tone(int pin, unsigned int frequency)
void tone(int pin, unsigned int frequency, unsigned long duration)
```

Generates a square wave of the specified frequency (and 50% duty cycle) on a pin (so as to make the passive buzzer vibrate to make sound). A duration can be specified, otherwise the wave continues until a call to noTone(). The pin can be connected to a piezo buzzer or other speaker to play tones.

Only one tone can be generated at a time. If a tone is already playing on a different pin, the call to tone() will have no effect. If the tone is playing on the same pin, the call will set its frequency.

Use of the tone() function will interfere with PWM output on pins 3 and 11 (on boards other than the Mega).

It is not possible to generate tones lower than 31Hz.

pin: The Arduino pin on which to generate the tone.

frequency: The frequency of the tone in hertz.

duration: The duration of the tone in milliseconds (optional)

```
void noTone(int pin)
```

Stops the generation of a square wave triggered by tone(). Has no effect if no tone is being generated.

pin: The Arduino pin on which to generate the tone.

Having known the two functions, you may grasp the codes—the installation of the array melody[] and the array noteDurations[] is the preparation of the subsequently several times of calling of the function tone() and the changing of tone and duration in the loop for better effect of music play.

②pitches.h: The code uses an extra file, **pitches.h**. This file contains all the pitch values for typical notes. For example, NOTE_C4 is middle C. NOTE_FS4 is F sharp, and so forth. This note table was originally written by Brett Hagman, on whose work the tone() command was based. You may find it useful whenever you want to make musical notes.

```
#include "pitches.h"
```

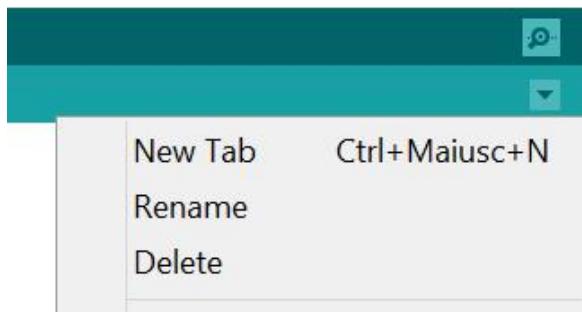
Note: There is already a pitches.h file in this sample program. If we put it together with the main code in one folder, the successive steps of installing pitches.h can be omitted.



```
2.11passiveBuzzer  pitches.h
#include "pitches.h"
int melody[] = {
```

After you open the code 2.11passiveBuzzer, if you cannot open the pitches.h code, you can just install one manually. The steps are as follows:

To make the pitches.h file, either click on the button just below the serial monitor icon and choose "New Tab", or use Ctrl+Shift+N.



Then paste in the following code and save it as **pitches.h**:

```
*****
Public Constants
*****
#define NOTE_B0 31
#define NOTE_C1 33
#define NOTE_CS1 35
#define NOTE_D1 37
#define NOTE_DS1 39
#define NOTE_E1 41
#define NOTE_F1 44
#define NOTE_FS1 46
```

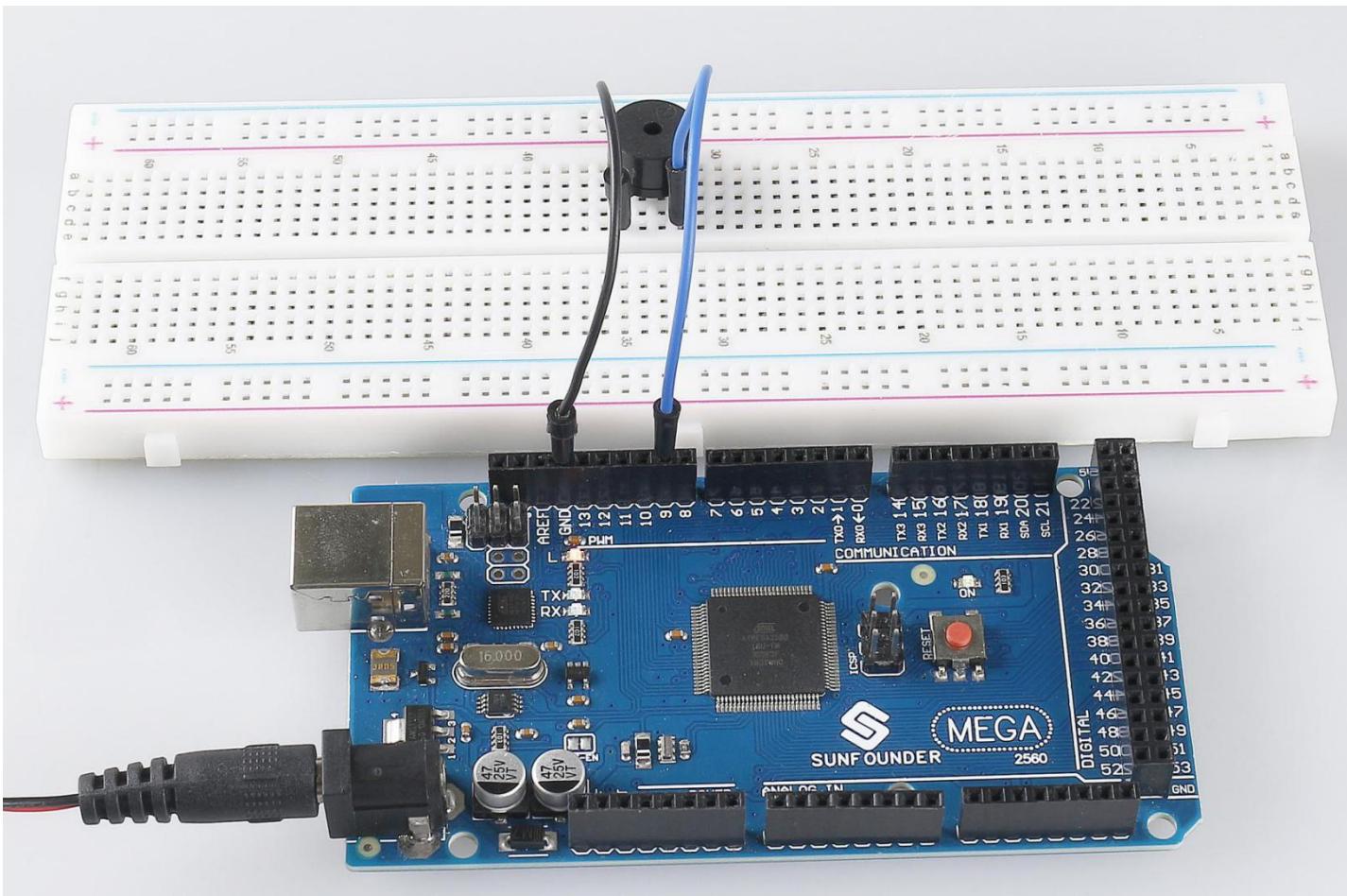
```
#define NOTE_G1 49
#define NOTE_GS1 52
#define NOTE_A1 55
#define NOTE_AS1 58
#define NOTE_B1 62
#define NOTE_C2 65
#define NOTE_CS2 69
#define NOTE_D2 73
#define NOTE_DS2 78
#define NOTE_E2 82
#define NOTE_F2 87
#define NOTE_FS2 93
#define NOTE_G2 98
#define NOTE_GS2 104
#define NOTE_A2 110
#define NOTE_AS2 117
#define NOTE_B2 123
#define NOTE_C3 131
#define NOTE_CS3 139
#define NOTE_D3 147
#define NOTE_DS3 156
#define NOTE_E3 165
```

```
#define NOTE_F3 175
#define NOTE_FS3 185
#define NOTE_G3 196
#define NOTE_GS3 208
#define NOTE_A3 220
#define NOTE_AS3 233
#define NOTE_B3 247
#define NOTE_C4 262
#define NOTE_CS4 277
#define NOTE_D4 294
#define NOTE_DS4 311
#define NOTE_E4 330
#define NOTE_F4 349
#define NOTE_FS4 370
#define NOTE_G4 392
#define NOTE_GS4 415
#define NOTE_A4 440
#define NOTE_AS4 466
#define NOTE_B4 494
#define NOTE_C5 523
#define NOTE_CS5 554
#define NOTE_D5 587
```

```
#define NOTE_DS5 622
#define NOTE_E5 659
#define NOTE_F5 698
#define NOTE_FS5 740
#define NOTE_G5 784
#define NOTE_GS5 831
#define NOTE_A5 880
#define NOTE_AS5 932
#define NOTE_B5 988
#define NOTE_C6 1047
#define NOTE_CS6 1109
#define NOTE_D6 1175
#define NOTE_DS6 1245
#define NOTE_E6 1319
#define NOTE_F6 1397
#define NOTE_FS6 1480
#define NOTE_G6 1568
#define NOTE_GS6 1661
#define NOTE_A6 1760
#define NOTE_AS6 1865
#define NOTE_B6 1976
#define NOTE_C7 2093
```

```
#define NOTE_CS7 2217
#define NOTE_D7 2349
#define NOTE_DS7 2489
#define NOTE_E7 2637
#define NOTE_F7 2794
#define NOTE_FS7 2960
#define NOTE_G7 3136
#define NOTE_GS7 3322
#define NOTE_A7 3520
#define NOTE_AS7 3729
#define NOTE_B7 3951
#define NOTE_C8 4186
#define NOTE_CS8 4435
#define NOTE_D8 4699
#define NOTE_DS8 49
```

Phenomenon Picture



2.12 Servo

Overview

In this lesson, you will learn something about Servo. Servo is a kind of driver whose position (angular) can be adjustable and kept or a rotary actuator that allows for precise control of angular position. Currently, it is widely used in upscale remote control toys, such as airplane, submarine, telerobot and so on.

Components Required

1 * Servo



1 * Mega 2560 Board



Several Jumper Wires



Component Introduction

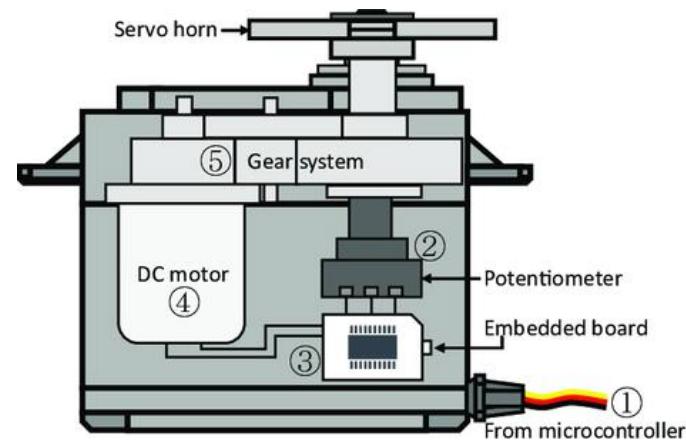


A servo is generally composed of the following parts: case, shaft, gear system, potentiometer, DC motor, and embedded board.

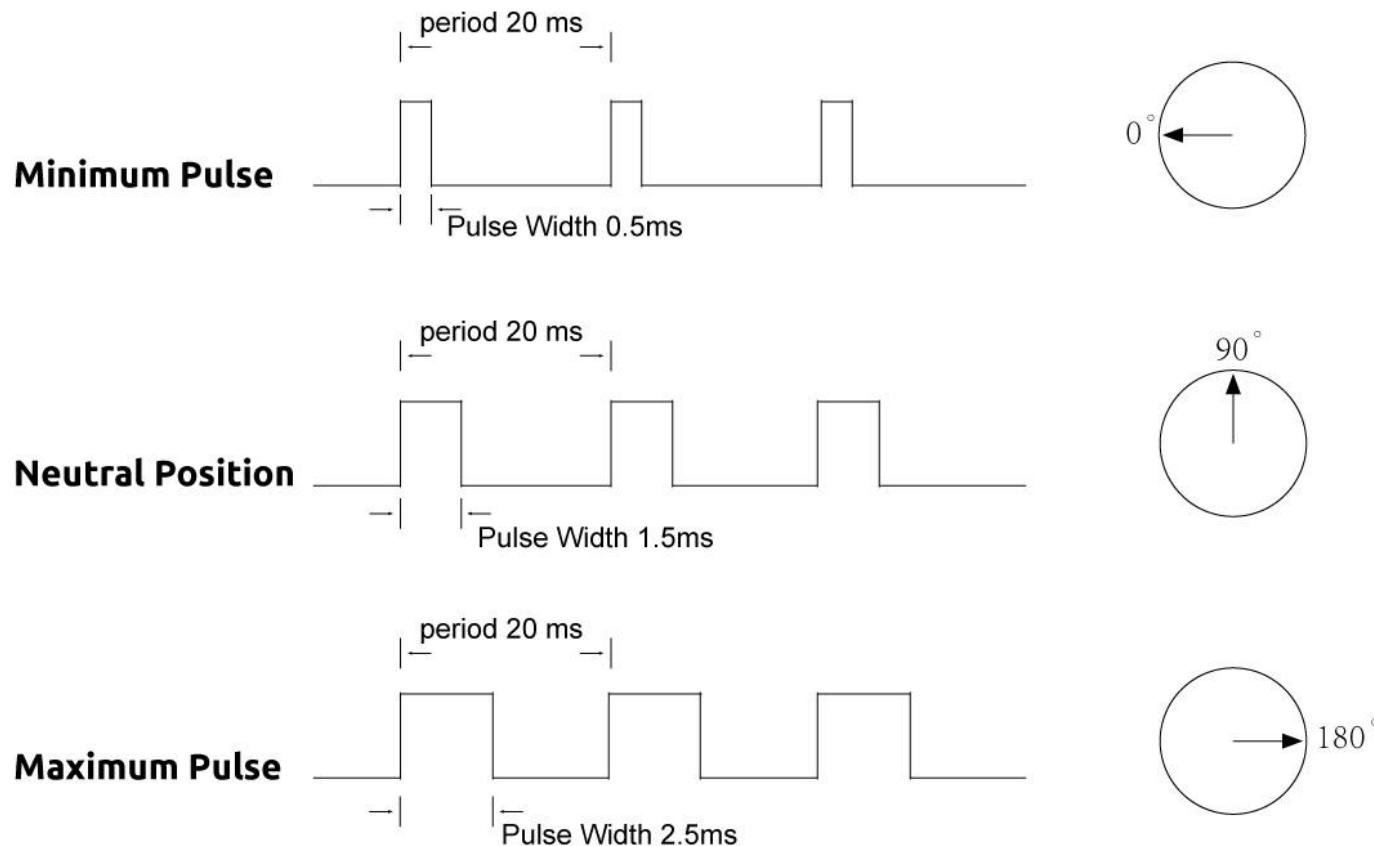
It works like this: The microcontroller sends out PWM signals to the servo, and then the embedded board in the servo receives the signals through the signal pin and controls the motor inside to turn. As a result, the motor drives the gear system and then motivates the shaft after deceleration. The shaft and potentiometer of the servo are connected together. When the shaft rotates, it drives the potentiometer, so the potentiometer outputs a voltage signal to the embedded board. Then the board determines the direction and speed of rotation based on the current position, so it can stop exactly at the right position as defined and hold there.

The angle is determined by the duration of a pulse that is applied to the control wire. This is called Pulse width Modulation. The servo expects to see a pulse every 20 ms. The length of the pulse will determine how far the motor turns. For example, a 1.5ms pulse will make the motor turn to the 90 degree position (neutral position).

When a pulse is sent to a servo that is less than 1.5 ms, the servo rotates to a position and holds its output shaft some number of degrees counterclockwise from the neutral point.

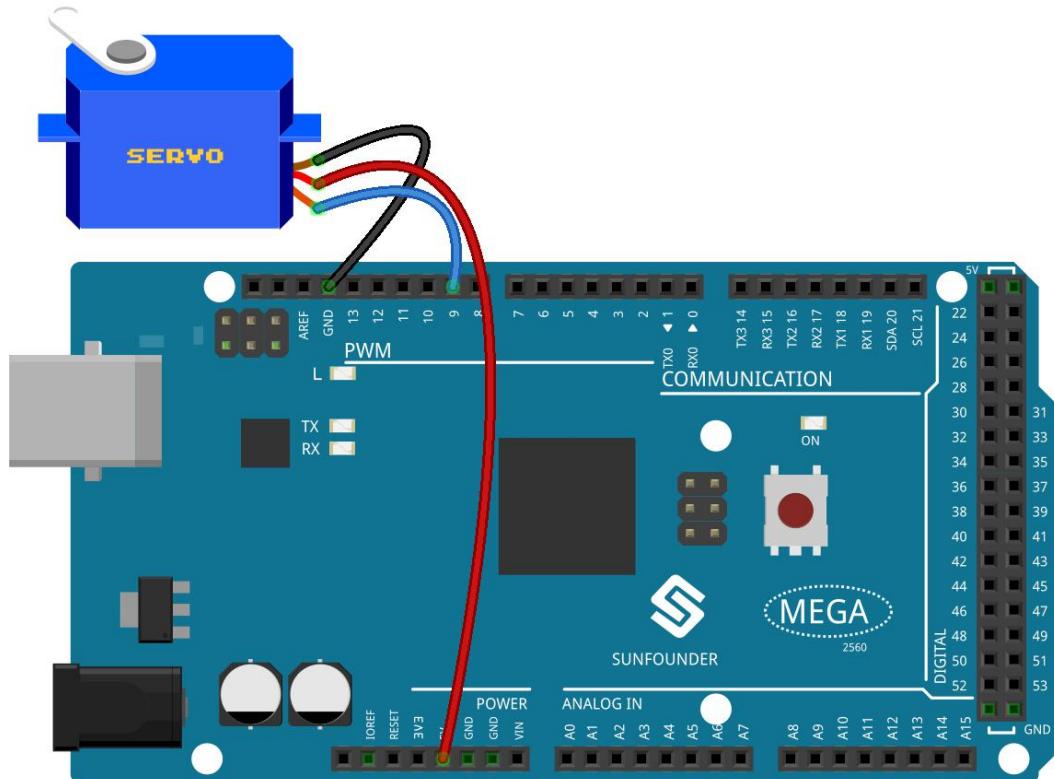


When the pulse is wider than 1.5 ms the opposite occurs. The minimal width and the maximum width of pulse that will command the servo to turn to a valid position are functions of each servo. Generally the minimum pulse will be about 0.5 ms wide and the maximum pulse will be 2.5 ms wide.

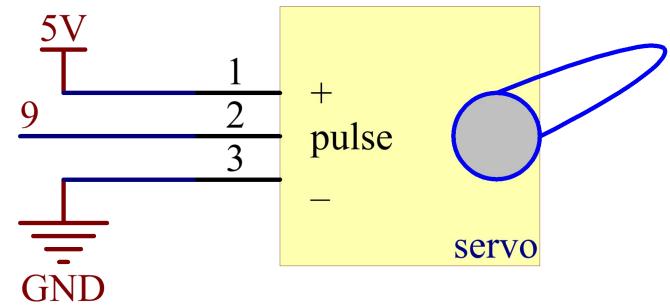


Fritzing Circuit

In this example, we use PWM pin 9 to drive the Servo, and get the orange wire of the servo connected to the PWM pin 9, the red one to 5V, and the brown one to GND.



Schematic Diagram



Code

```
#include <Servo.h>
Servo myservo;//create servo object to control a servo
void setup()
{
    myservo.attach(9);//attaches the servo on pin 9 to servo object
    myservo.write(0);//back to 0 degrees
    delay(1000);//wait for a second
}
void loop()
{
    for (int i = 0; i <= 180; i++){
        myservo.write(i); //write the i angle to the servo
        delay(15); //delay 15ms
    }
    for (int i = 180; i >= 0; i--)
    {
        myservo.write(i); //write the i angle to the servo
        delay(15); //delay 15ms
    }
}
```

Once you finish uploading the codes to the Mega2560 board, you can see the servo arm rotating in the range 0°~180°.

Code Analysis

By calling the library Servo.h, you can drive the servo easily.

```
#include <Servo.h>
```

Library Functions:

Servo

Create **Servo** object to control a servo.

```
uint8_t attach(int pin);
```

Turn a pin into a servo driver. Calls pinMode. Returns 0 on failure.

```
void detach();
```

Release a pin from servo driving.

```
void write(int value);
```

Set the angle of the servo in degrees, 0 to 180.

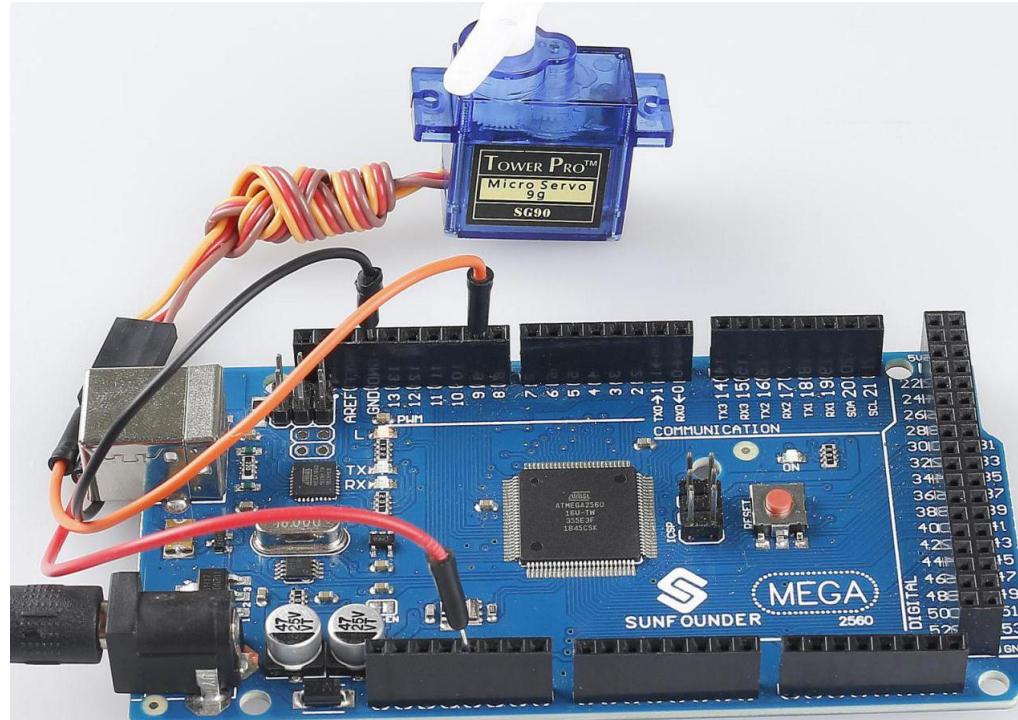
int read();

Return that value set with the last write().

bool attached();

Return 1 if the servo is currently attached.

Phenomenon Picture



2.13 Motor

Overview

In this lesson, you will learn how to use Motor, the working principle of which is that the energized coil is forced to rotate in the magnetic field then the rotor of the motor rotates accordingly on which the pinion gear drives the engine flywheel to rotate.

Components Required

1 * Motor



1 * L293D



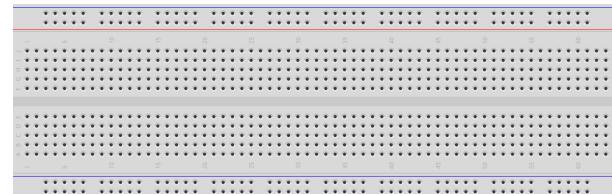
1 * Power Supply Module



1 * Mega 2560 Board



1 * Breadboard



Several Jumper Wires



Component Introduction

This is a 5V DC motor. It will rotate when you give the two terminals of the copper sheet one high and one low level. For convenience, you can weld the pins to it.



Size: 25*20*15MM

Free-run current (3V): 70mA

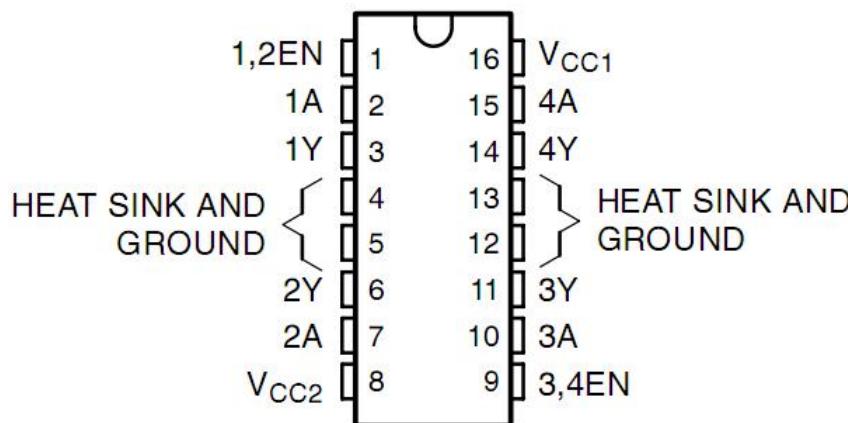
Stall current (3V): 800mA

Operation Voltage: 1-6V

A Free-run speed (3V): 13000RPM

Shaft diameter: 2mm

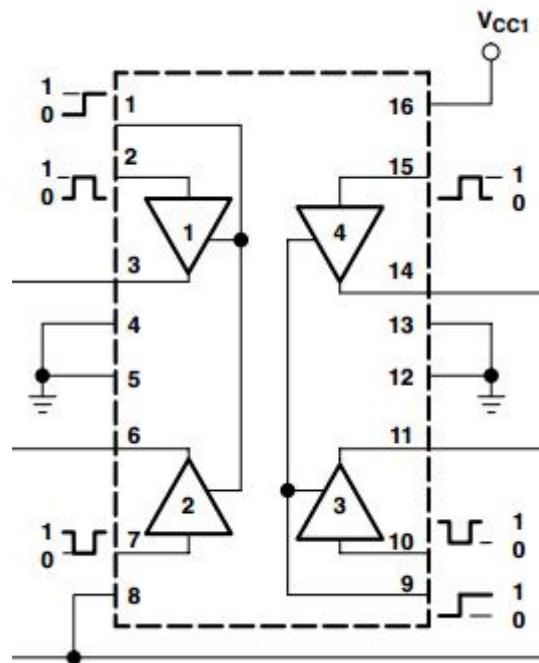
L293D



L293D is a 4-channel motor driver integrated by chip with high voltage and high current. It's designed to connect to standard DTL, TTL logic level, and drive inductive loads (such as relay coils, DC, Stepper Motors) and power switching transistors etc. DC Motors are devices that turn DC electrical energy into mechanical energy. They are widely used in electrical drive for their superior speed regulation performance.

L293D has two pins (Vcc1 and Vcc2) for power supply. Vcc2 is used to supply power for the motor, while Vcc1 to supply for the chip.

The following is the internal structure of L293D. Pin EN is an enable pin and only works with high level; A stands for input and Y for output. You can see the relationship among them at the right bottom. When pin EN is High level, if A is High, Y outputs high level; if A is Low, Y outputs Low level. When pin EN is Low level, the L293D does not work.



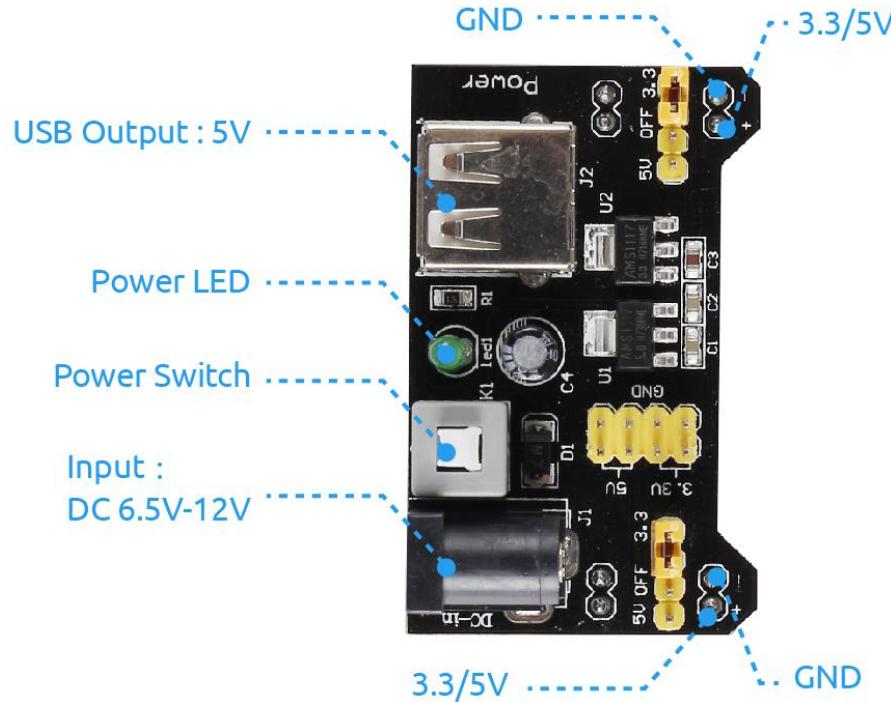
INPUTS†		OUTPUT Y
A	EN	
H	H	H
L	H	L
X	L	Z

H = high level, L = low level, X = irrelevant,
Z = high impedance (off)

Power Supply Module

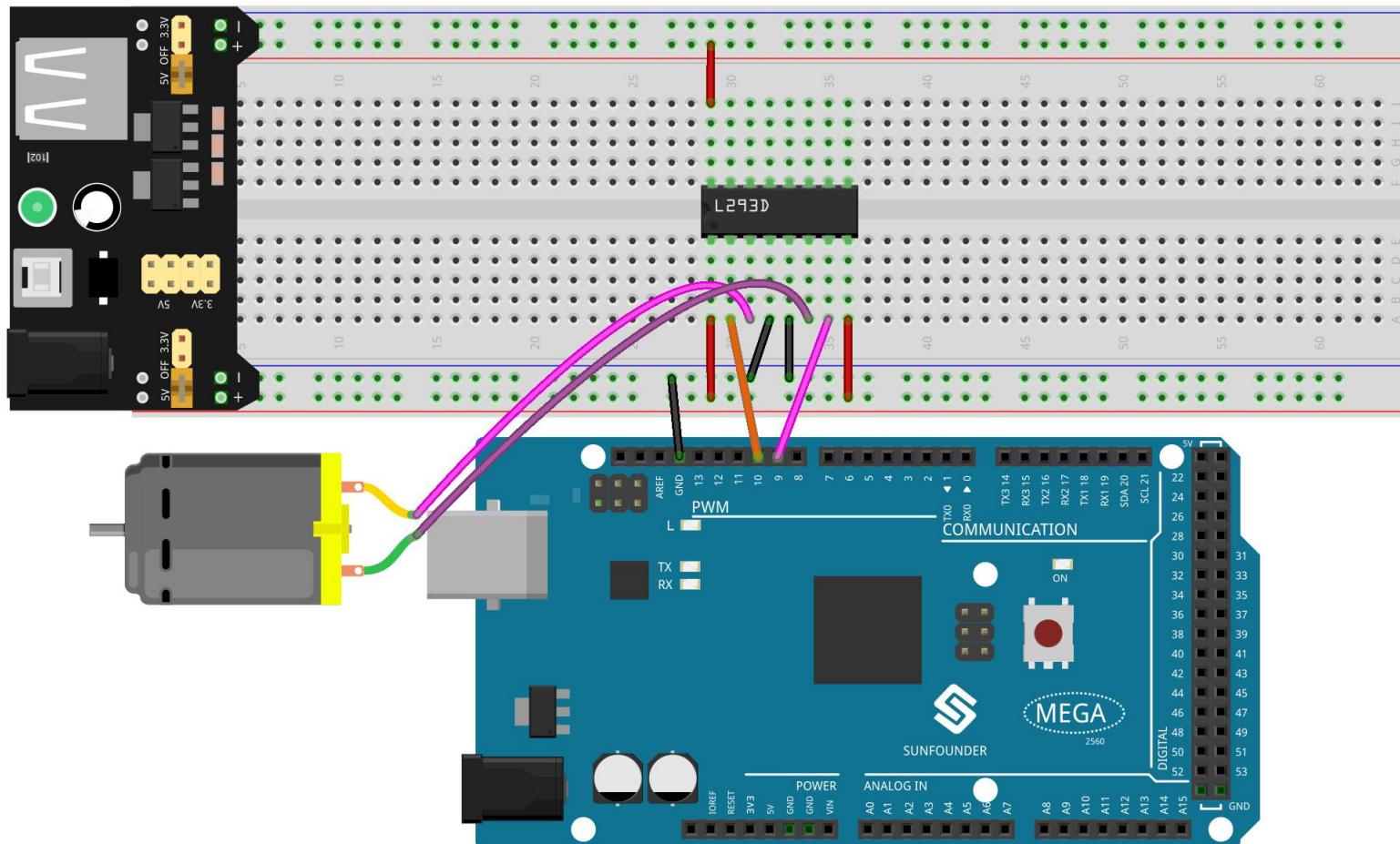
In this experiment, it needs large currents to drive the motor especially when it starts and stops, which will severely interfere with the normal work of Raspberry Pi. Therefore, we separately supply power for the motor by this module to make it run safely and steadily.

You can just plug it in the breadboard to supply power. It provides a voltage of 3.3V and 5V, and you can connect either via a jumper cap included.

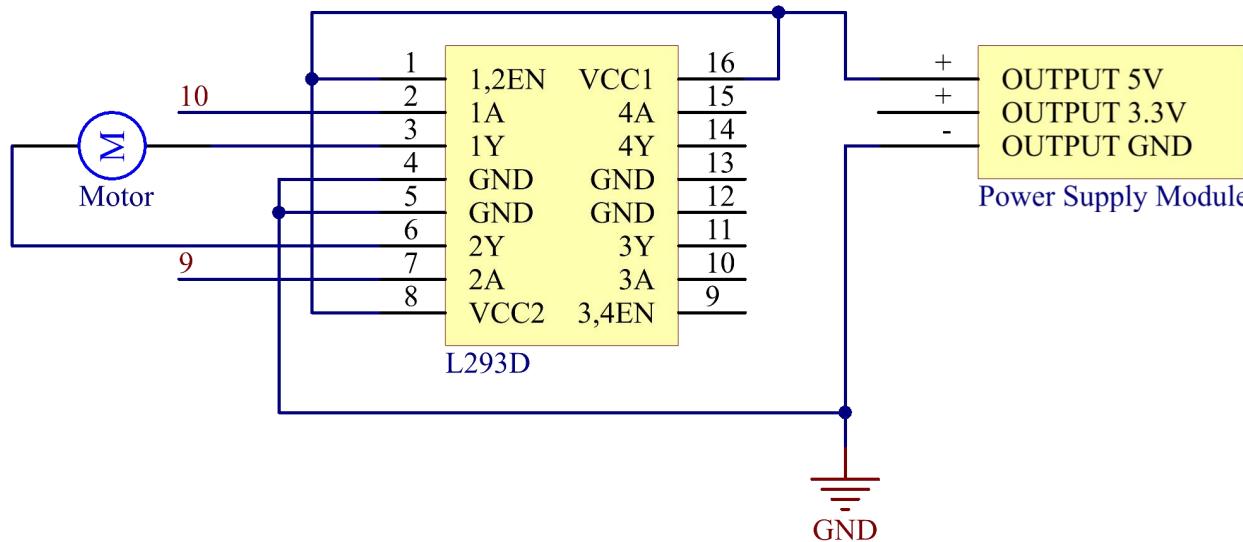


Fritzing Circuit

In this example, we use Power Supply Module to power the anode and cathode of breadboard. GND of Mega 2560 Board is connected to the cathode.



Schematic Diagram



Code

```
const int motor1A=10;
const int motor2A=9;

void setup() {
  pinMode(motor1A,OUTPUT);
  pinMode(motor2A,OUTPUT);
  Serial.begin(9600);
```

```
Serial.println("Please input 'A' or 'B' to select the motor rotate direction.");
}

void loop() {
    if (Serial.available() > 0) {
        int incomingByte = Serial.read();
        switch(incomingByte){
            case 'A':
                clockwise(255);
                Serial.println("The motor rotate clockwise.");
                break;
            case 'B':
                anticlockwise(255);
                Serial.println("The motor rotate anticlockwise.");
                break;
        }
        delay(3000);
        stopMotor();
    }
}

void clockwise(int Speed)
{
```

```
analogWrite(motor1A,0);
analogWrite(motor2A,Speed);
}

void anticlockwise(int Speed)
{
    analogWrite(motor1A,Speed);
    analogWrite(motor2A,0);
}
void stopMotor()
{
    analogWrite(motor1A,0);
    analogWrite(motor2A,0);
}
```

After uploading the codes to the Mega2560 board, you can select the rotating direction of motor by typing 「A」 or 「B」 in the serial monitor.

Code Analysis

The motor can be driven by providing a voltage difference between the copper sheets at both sides of the motor. Therefore, you only need to write 0 for the voltage of one side of the copper sheet and 5V for the other side. Modify the

written analog signal value to adjust the direction and speed.

```
void clockwise(int Speed)
{
    digitalWrite(motor1A,0);
    digitalWrite(motor2A,Speed);
}

void anticlockwise(int Speed)
{
    digitalWrite(motor1A,Speed);
    digitalWrite(motor2A,0);
}
```

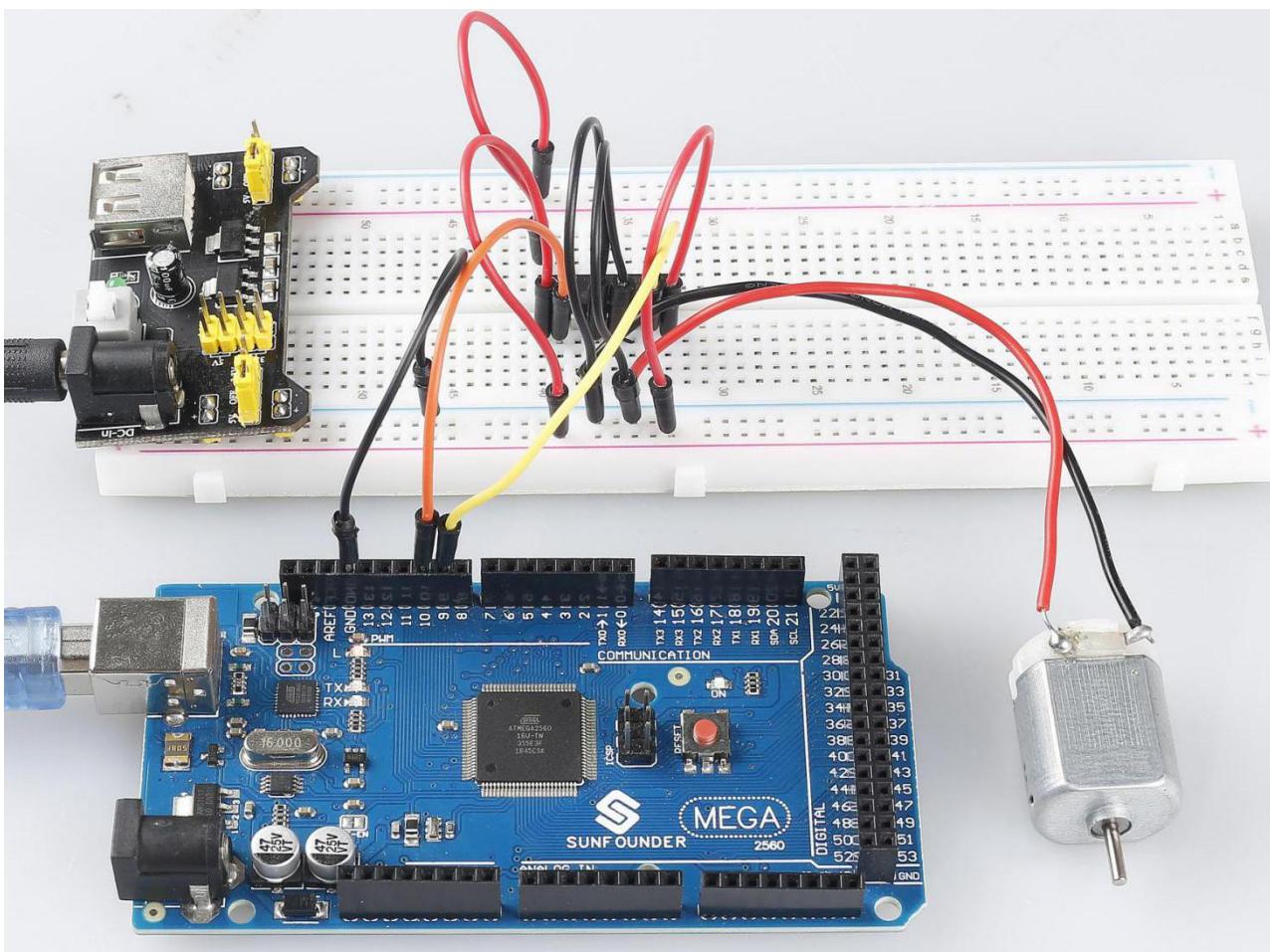
In this example, Serial.Read() is used to control the direction of motor.

When you type 'A' in serial monitor, there calls the clockwise (255) function to make the motor rotate with the speed of 255. Input 'B', and the motor will rotate in reverse direction.

```
void loop() {
    if (Serial.available() > 0) {
        int incomingByte = Serial.read();
        switch(incomingByte){
            case 'A':
```

```
clockwise(255);
Serial.println("The motor rotate clockwise.");
break;
case 'B':
    anticlockwise(255);
    Serial.println("The motor rotate anticlockwise.");
    break;
}
}
delay(3000);
stopMotor();
}
```

Phenomenon Picture



2.14 Stepper Motor

Overview

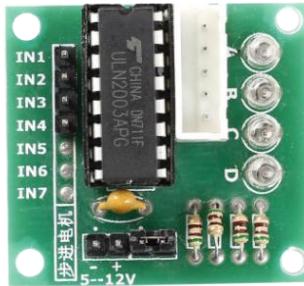
In this lesson, you will learn about Stepper Motor.

Components Required

1 * Stepper Motor



1 * ULN2003



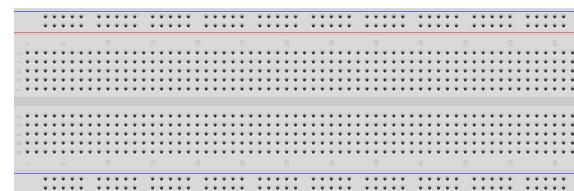
1 * Power Supply Module



1 * Mega 2560 Board



1 * Breadboard



Several Jumper Wires



Component Introduction



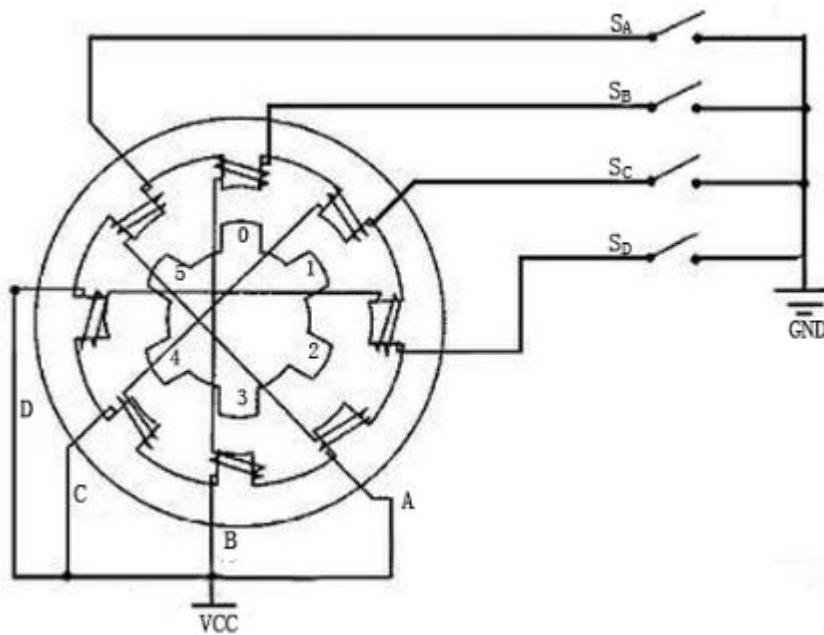
Stepper motor is an open-loop control motor by converting the electric pulse signal into angular displacement or line displacement. It is the main executing component in the modern digital program control system. When the stepper driver receives a pulse signal, it drives the stepper motor to rotate a fixed angle in a set direction. The rotation of stepper driver runs step by step at a fixed angle. The angular displacement can be controlled by changing the number of pulses, thereby achieving the purpose of accurate positioning. At the same time, the speed and acceleration of the rotation of the motor can be controlled by adjusting the pulse frequency so as to achieve the purpose of speed regulation.

There are two types of steppers, unipolars and bipolars, and it is very important to know which type you are working with. In this experiment, we will use a unipolar stepper.

The stepper motor is a four-phase one, which uses a unipolarity DC power supply. As long as you electrify all phase windings of the motor by an appropriate timing sequence, you can make it rotate step by step. The schematic diagram of

a four-phase reactive stepper motor:

In the figure, in the middle of the motor is a rotor – a gear-shaped permanent magnet. Around the rotor, 0 to 5 are teeth. Then more outside, there are 8 magnetic poles, with each two opposite ones connected by coil winding. So they form four pairs from A to D, which is called a phase. It has four lead wires to be connected with switches SA, SB, SC, and SD. Therefore, the four phases are in parallel in the circuit, and the two magnetic poles in one phase are in series.



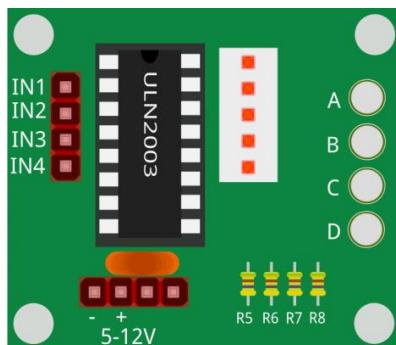
Here's how a 4-phase stepper motor works:

When switch SB is power on, switch SA, SC, and SD is power off, and B-phase magnetic poles align with tooth 0 and 3 of the rotor. At the same time, tooth 1 and 4 generate staggered teeth with C- and D-phase poles. Tooth 2 and 5 generate staggered teeth with D- and A-phase poles. When switch SC is power on, switch SB, SA, and SD is power off, the rotor rotates under magnetic field of C-phase winding and that between tooth 1 and 4. Then tooth 1 and 4 align with the magnetic poles of C-phase winding. While tooth 0 and 3 generate staggered teeth with A- and B-phase poles, and tooth 2 and 5 generate staggered teeth with the magnetic poles of A- and D-phase poles. The similar situation goes on and on. Energize the A, B, C and D phases in turn, and the rotor will rotate in the order of A, B, C and D.

with the magnetic poles of A- and D-phase poles. The similar situation goes on and on. Energize the A, B, C and D phases in turn, and the rotor will rotate in the order of A, B, C and D.

The stator of Stepper Motor we use has 32 magnetic poles, so a circle needs 32 steps. The output shaft of the Stepper Motor is connected with a reduction gear set, and the reduction ratio is 1/64. **So the final output shaft rotates a circle requiring a $32 \times 64 = 2048$ step.**

ULN2003



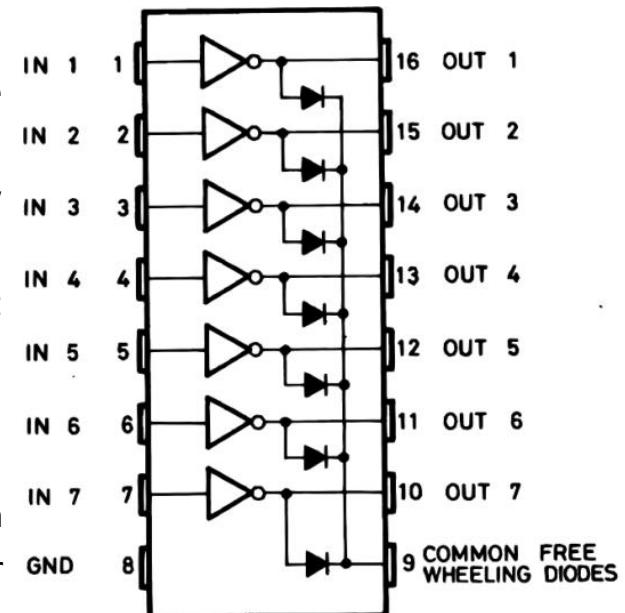
To apply the motor in the circuit, a driver board needs to be used. Stepper Motor Driver-ULN2003 is a 7-channel inverter circuit.

That is, when the input pin is at high level, the output pin of ULN2003 is at low level, and vice versa. If we supply high level to IN1, and low level to IN2, IN3 and IN4, then the output end OUT1 is at low level, and all the other output ends are at high level.

The internal structure of the chip is shown as below.

The stepper motor driver constituted by ULN2003 chip and 4 LEDs is shown as follows. On the board, IN1, IN2, IN3 and IN4 work as input and the four LEDs, A, B, C, D are the indicators of input pin.

In addition, OUT1, OUT2, OUT3 and OUT4 are connected to SA, SB, SC and SD on the stepper motor driver. When the value of IN1 is set to a high level, A lights up;

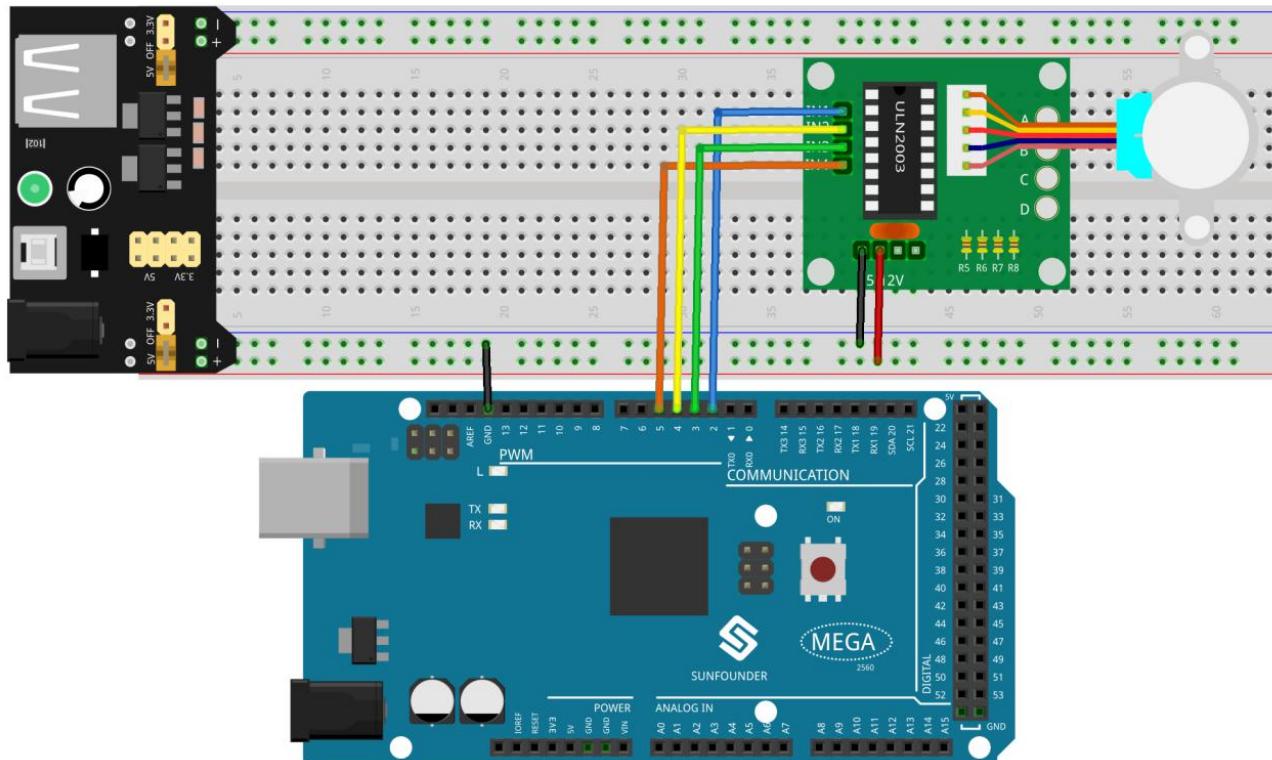


switch SA is power on, and the stepper motor rotates one step. The similar case repeats on and on.

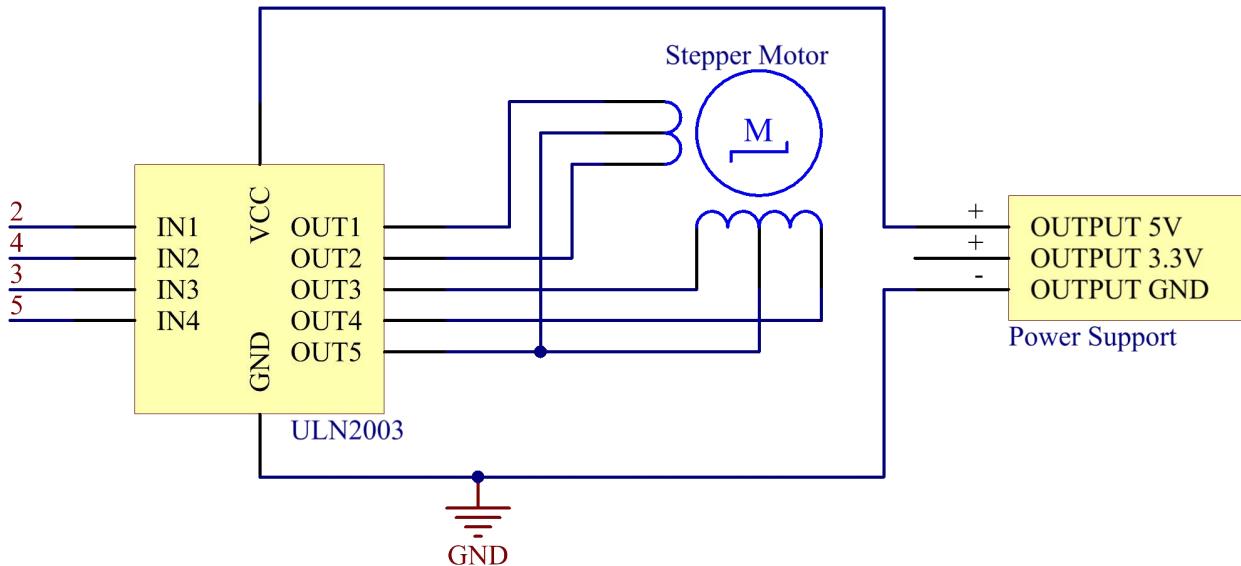
Fritzing Circuit

Power Supply Module is used to power the stepper motor. Get the GND of Mega 2560 Board and GND of ULN2003 connected to the cathode of the breadboard, and connect the VCC of ULN2003 to 5V OUTPUT of Power Supply.

The wiring of ULN2003 and Mega2560 is shown as follows:



Schematic Diagram



Code

```
#include <Stepper.h>
const int stepsPerRevolution = 2048; // change this to fit the number of steps per revolution
const int ratePerMinute = 16; // Adjustable range of 28BYJ-48 stepper is 0~17 rpm

//set steps and the connection with MCU
Stepper stepper(stepsPerRevolution, 2, 3, 4, 5);
```

```
void setup()
{
    stepper.setSpeed(rolePerMinute);
}

void loop()
{
    int val = 2048;
    stepper.step(val); //Turn the motor in val steps
    delay(1000);
}
```

After uploading the codes to the Mega2560 board, you will be able to see that the stepper motor rotates one circle with an interval of a second and each circle takes 3.75s.

Code Analysis

By calling the library Stepper.h, you can easily drive the stepper motor.

```
#include <Stepper.h>
```

Library Functions:

```
Stepper(steps, pin1, pin2, pin3, pin4)
```

Creates a new instance of the Stepper class that represents a particular stepper motor attached to your Arduino board.

steps: the number of steps in one revolution of your motor. If your motor gives the number of degrees per step, divide that number into 360 to get the number of steps (e.g. 360 / 3.6 gives 100 steps). (int)

Note: every circle of the stepper motor takes 2048 steps.

setSpeed(rpm)

Sets the motor speed in rotations per minute. This function doesn't make the motor turn, just sets the speed at which it will when you call step().

rpm: the speed at which the motor should turn in rotations per minute - a positive number. (long)

Note: The stepper motor we use here rotates 17 circles a minute at most.

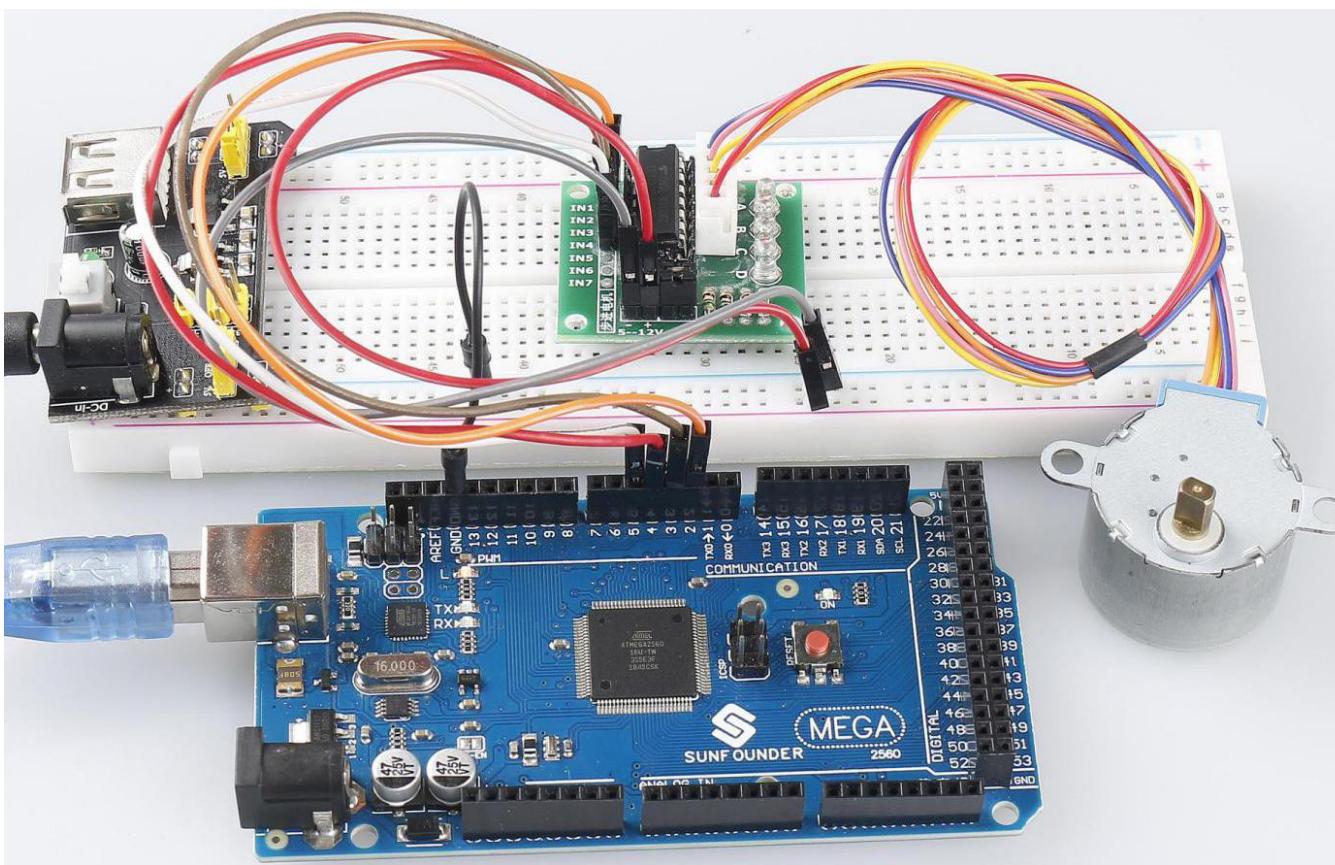
step(steps)

Turns the motor a specific number of steps, at a speed determined by the most recent call to setSpeed().

This function is blocking; that is, it will wait until the motor has finished moving to pass control to the next line in your sketch. For example, if you set the speed to, say, 1 RPM and called step(2048) on a 2048-step motor, this function would take a full minute to run. For better control, keep the speed high and only go a few steps with each call to step().

steps: the number of steps to turn the motor - positive to turn one direction, negative to turn the other. (int)

Phenomenon Picture

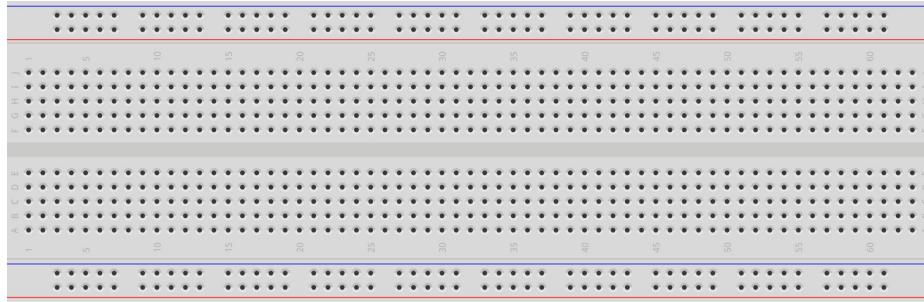


2.15 Button

Overview

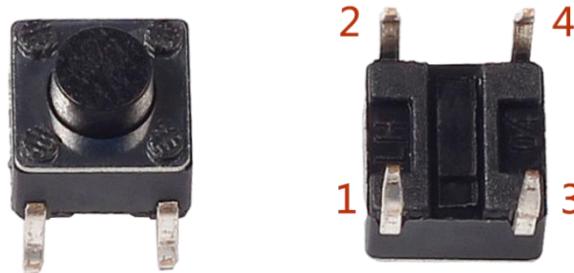
In this lesson, you will learn about Button. Button is a common component used to control electronic devices. It is usually used as a switch to connect or break circuits.

Components Required

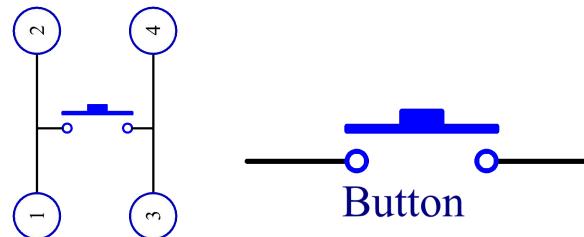
1 * Button		1 * 10k ohm resistor		Several Jumper Wires		
1 * Mega 2560 Board		1 * Breadboard				

Component Introduction

Two pins on the left are connected, and the one on the right is similar to the left, which is shown below:



The symbol shown as below is usually used to represent a button in circuits.

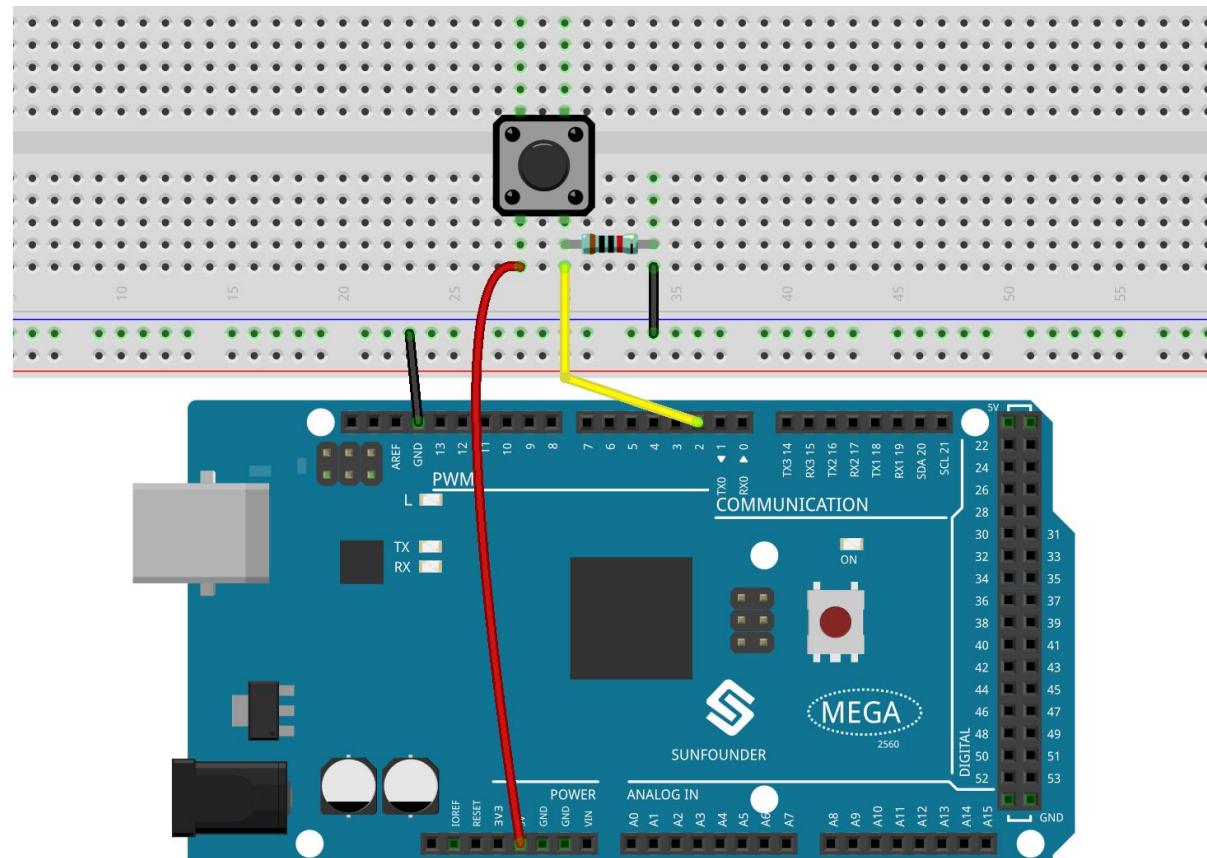


When the button is pressed, the 4 pins are connected, thus closing the circuit.

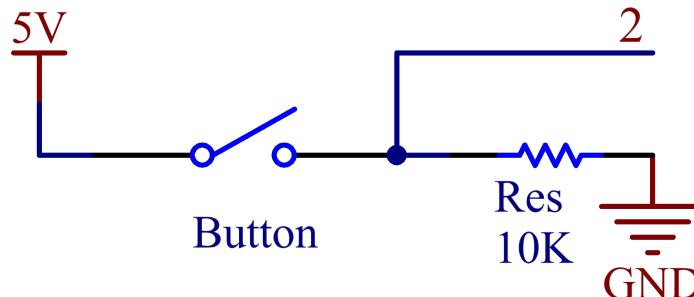
Fritzing Circuit

In this example, we read the signal of the button with the digital pin 2. When the button is not pressed, the digital pin 2 (through the drop-down resistor) is connected to ground to read the low level (0); when the button is pressed, the two pins are connected and when the pin is connected to the 5V power supply, the high level (1) is read.

NOTE : If you disconnect the digital I/O pin from anything, the LED may blink erratically. The input is "floating" or it doesn't have a solid connection to voltage or ground, so it will randomly return either HIGH or LOW. That's why there needs a pull-down resistor in the circuit.



Schematic Diagram



Code

Example 1:

```
void setup() {
    Serial.begin(9600);
    pinMode(2, INPUT);
}

void loop() {
    int buttonState = digitalRead(2);
    Serial.println(buttonState);
    delay(1);
}
```

Uploaded the codes to the Mega2560 board, you can see the readings of the pins on the serial monitor. When you press down the Button, there will display [1] on the serial monitor, and once you release it, there will display [0]. As for the detail code explanation, please refer to [Part 1-1.4 Digital Read](#).

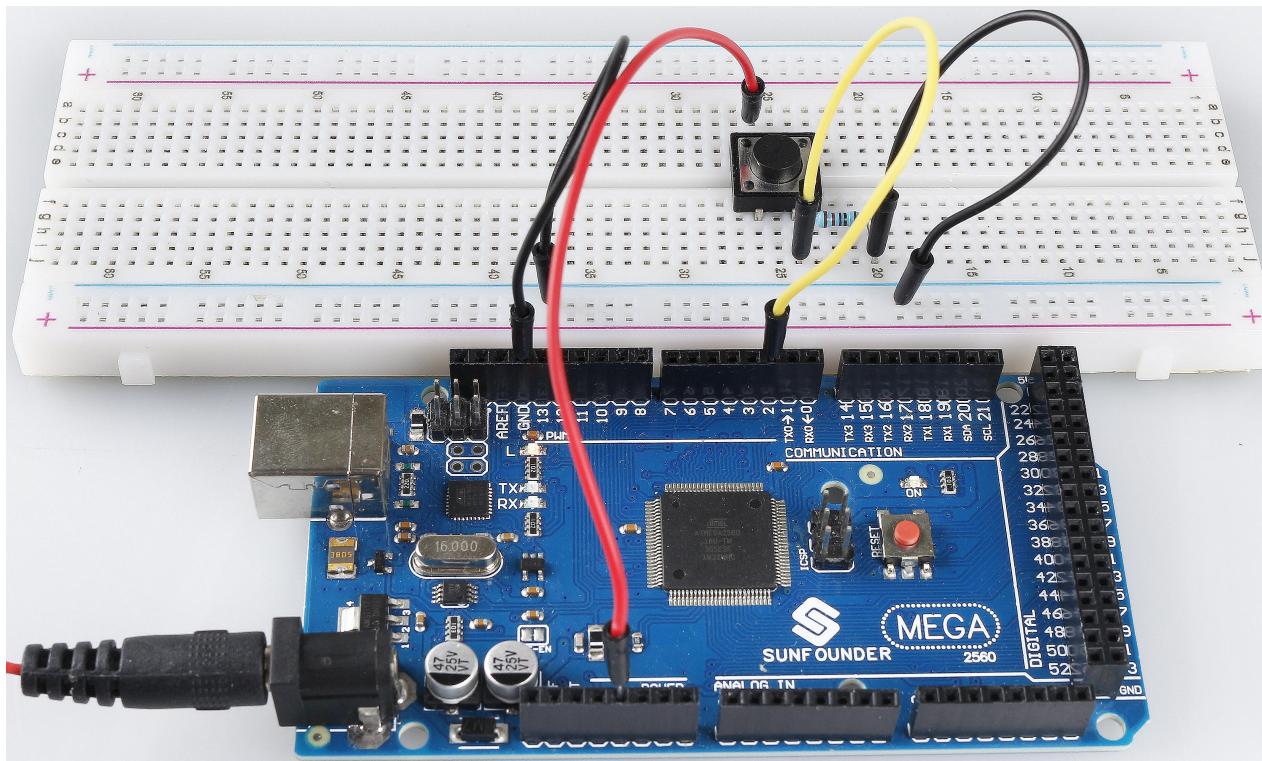
Example 2:

```
const int buttonPin = 2;  
int detectionState = 0;  
int buttonState = 0;  
int lastButtonState = 0;  
  
void setup() {  
    pinMode(buttonPin, INPUT);  
    Serial.begin(9600);  
}  
  
void loop() {  
    buttonState = digitalRead(buttonPin);  
    if (buttonState != lastButtonState) {  
        if (buttonState == HIGH) {  
            detectionState=(detectionState+1)%2;  
            Serial.print("The detection state is:");  
            Serial.println(detectionState);  
        }  
        delay(50);  
    }  
}
```

```
lastButtonState = buttonState;  
}
```

Uploaded the codes to the Mega2560 board, every time you press the button, the output value will switch between 0 and 1. If you want to know more about the code explanation, you can turn to [Part 1-1.10 State Change Detection](#).

Phenomenon Picture

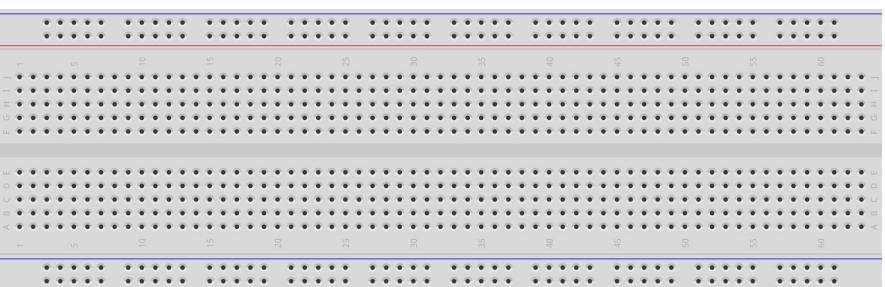


2.16 Slide Switch

Overview

In this lesson, you will get to know something about Switch. A slide switch, just as its name implies, is to slide the switch bar to connect or break the circuit, and further switch circuits. The slide switch is commonly used in low-voltage circuit. It has the features of flexibility and stability, and applies in electric instruments and electric toys widely.

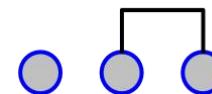
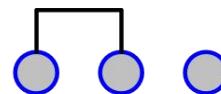
Components Required

1 * Switch	1 * 104 Capacitor	Several Jumper Wires
		
1 * Mega 2560 Board	1 * Breadboard	
		

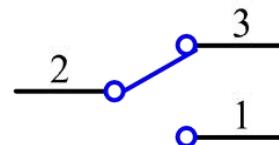
Component Introduction



How it works: Set the middle pin as the fixed one. When you pull the slide to the left, the two pins on the left are connected; when you pull it to the right, the two pins on the right are connected. Thus, it works as a switch connecting or disconnecting circuits. See the figure below:

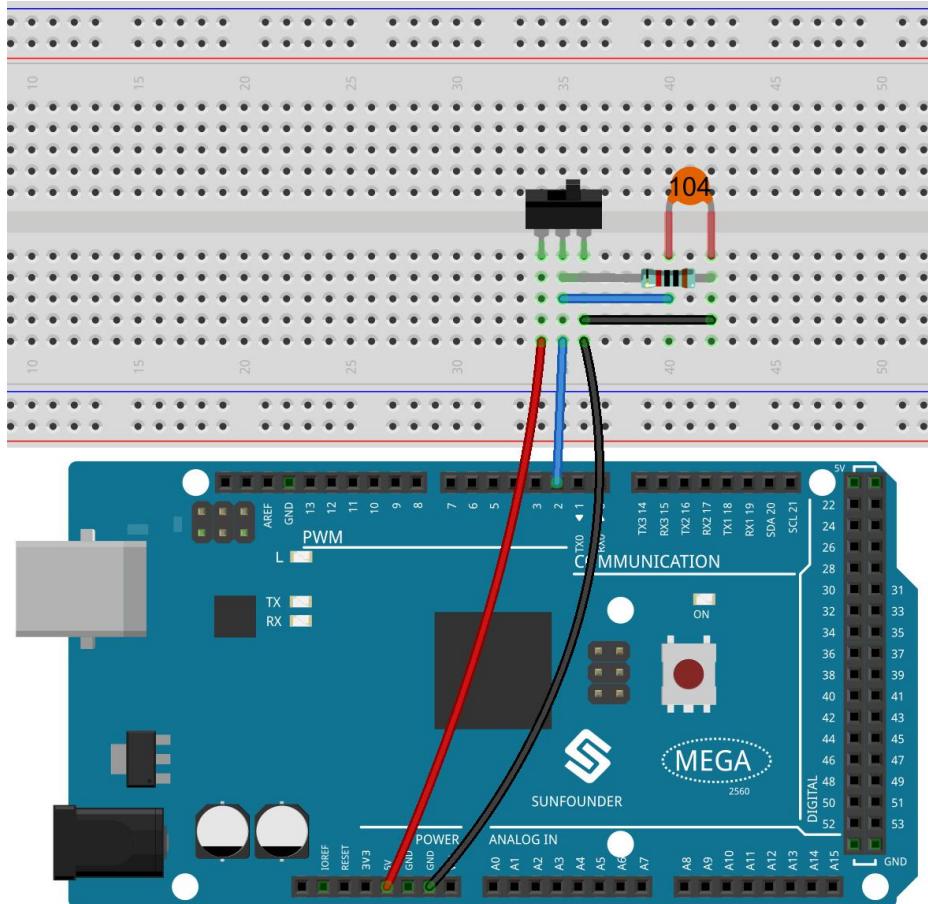


The circuit symbol of the slide switch is shown as below. The pin2 in the figure refers to the middle pin.

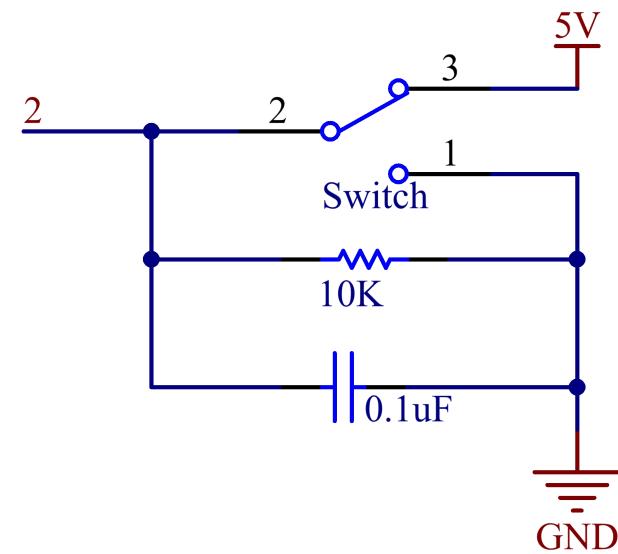


Fritzing Circuit

In this example, digital pin 2 is used to read the signal of Switch. In addition, you need to connect a $10\text{ k}\Omega$ resistor and a 104 capacitor in parallel to form a RC circuit (Resistor - Capacitance circuit) which is set between pin 2 and GND to realize debounce that may arise from your toggle of switch.



Schematic Diagram

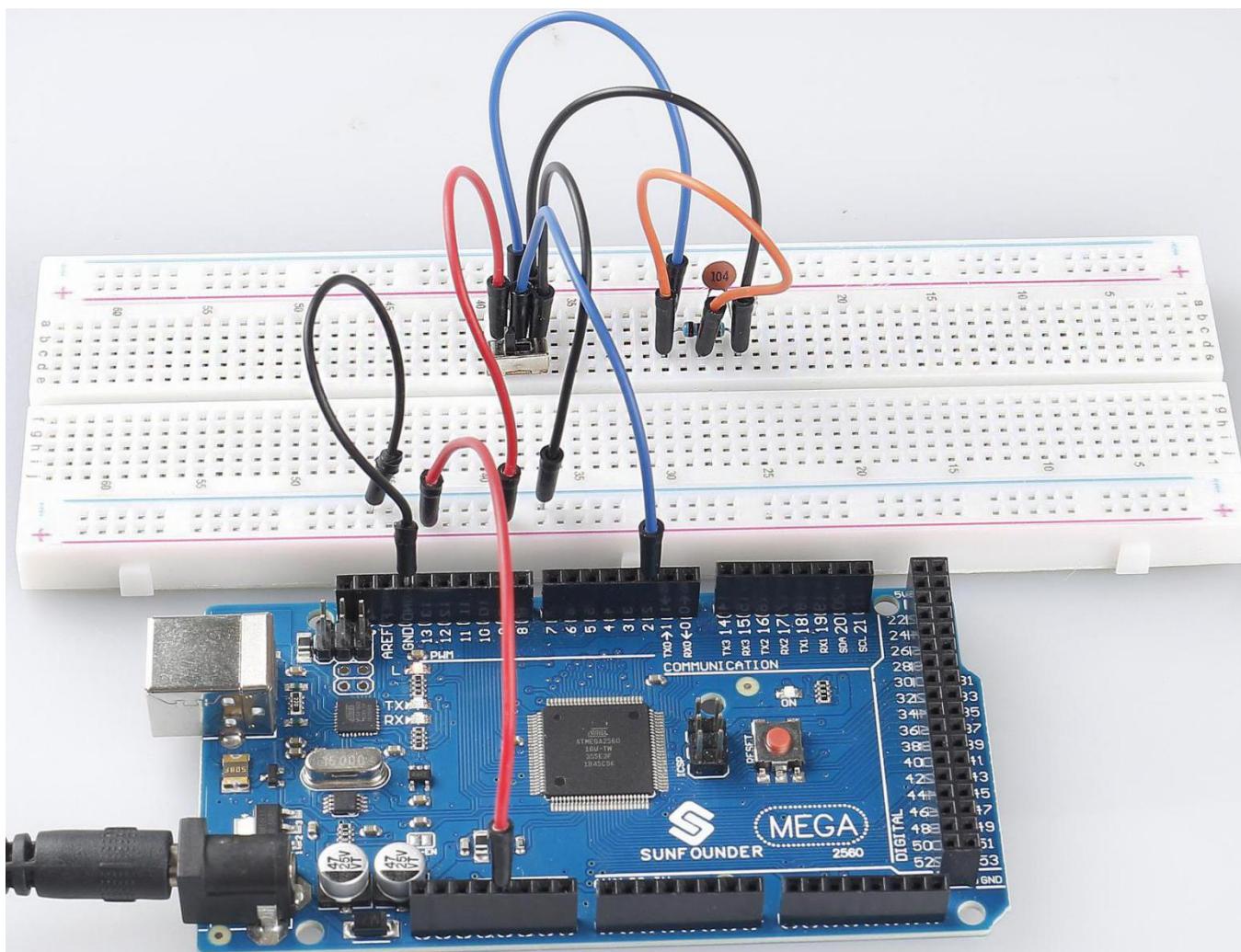


Code

```
void setup() {  
    Serial.begin(9600);  
    pinMode(2, INPUT);  
}  
  
void loop() {  
    int switchState = digitalRead(2);  
    Serial.println(switchState);  
    delay(1);  
}
```

After the codes are uploaded to the Mega2560 board, you can open the serial monitor to check the readings of the pin. When the Switch toggles to the left, the serial monitor displays 「1」 ; when toggles to the right, the serial monitor displays 「0」 . Refer to [Part 1-1.4 Digital Read](#) to check the code explanation.

Phenomenon Picture

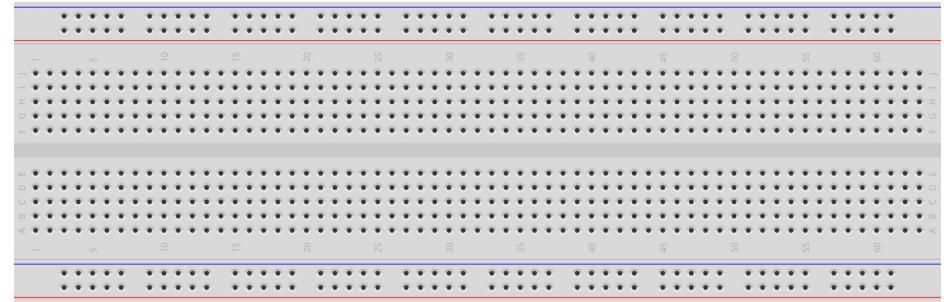


2.17 Tilt Switch

Overview

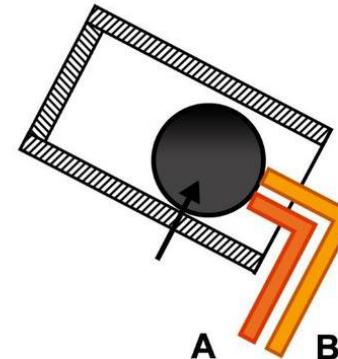
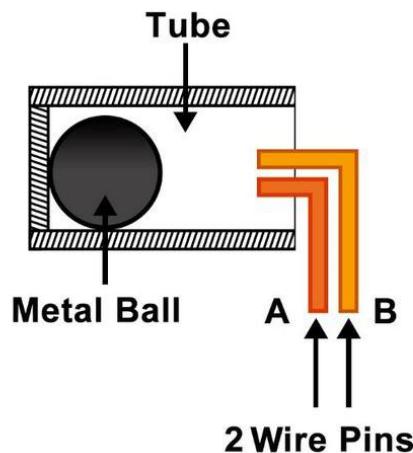
In this lesson, you will learn about tilt switch. Tilt switch can be used to detect whether objects tilt, which is of great value in practical applications. It can be used to judge the tilt of bridges, buildings, transmission line tower and so on, so it has an important guiding function in carrying out maintenance work.

Components Required

1 * Tilt Switch	1 * 10k ohm Resistor	Several Jumper Wires
		
1 * Mega 2560 Board	1 * Breadboard	
		

Component Introduction

The principle is very simple. When the switch is tilted in a certain angle, the ball inside rolls down and touches the two contacts connected to the pins outside, thus triggering circuits. Otherwise the ball will stay away from the contacts, thus breaking the circuits.



Ball Slides Down to Connect Both Contacts



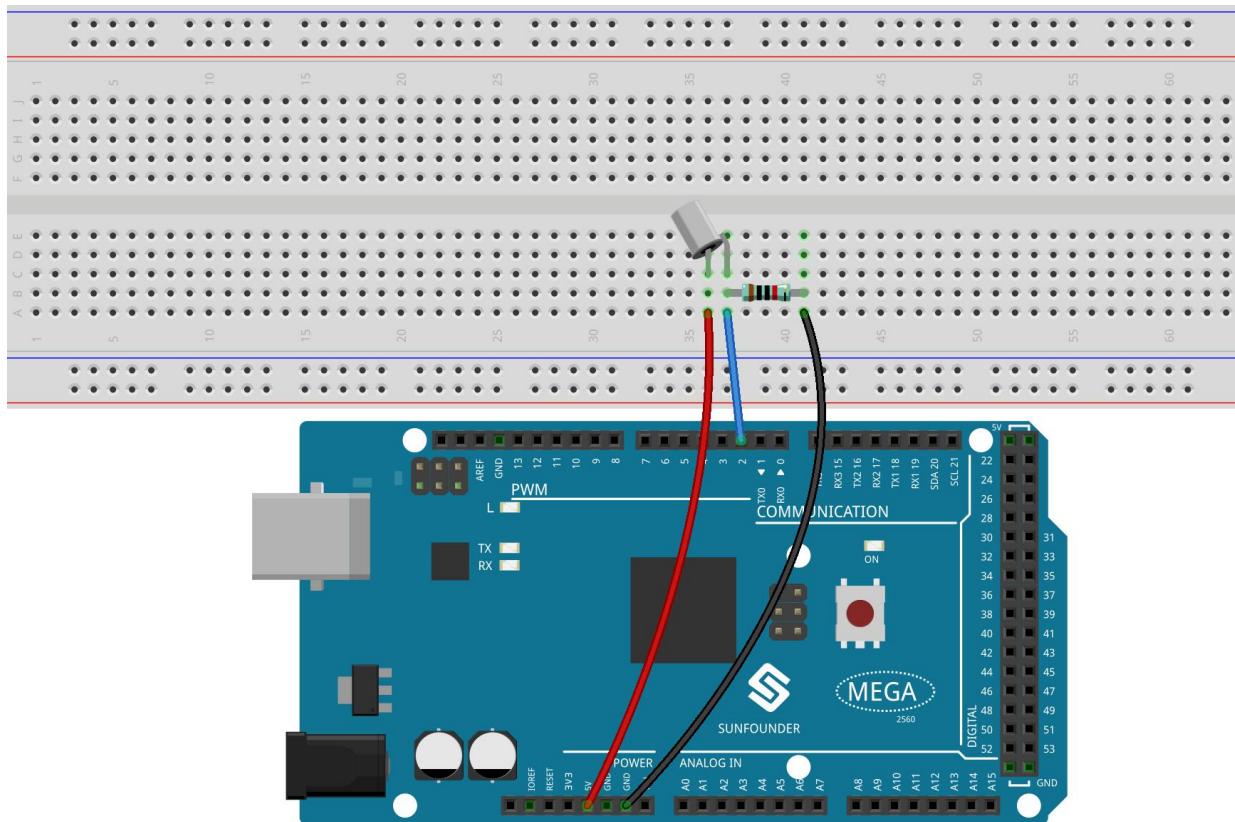
Schematic Equivalent



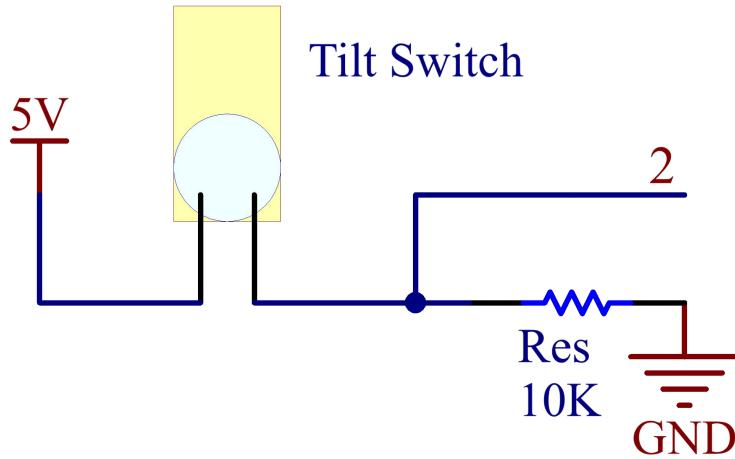
Schematic Equivalent

Fritzing Circuit

In this example, digital pin 2 is used to read the signal of Tilt Switch.



Schematic Diagram



Code

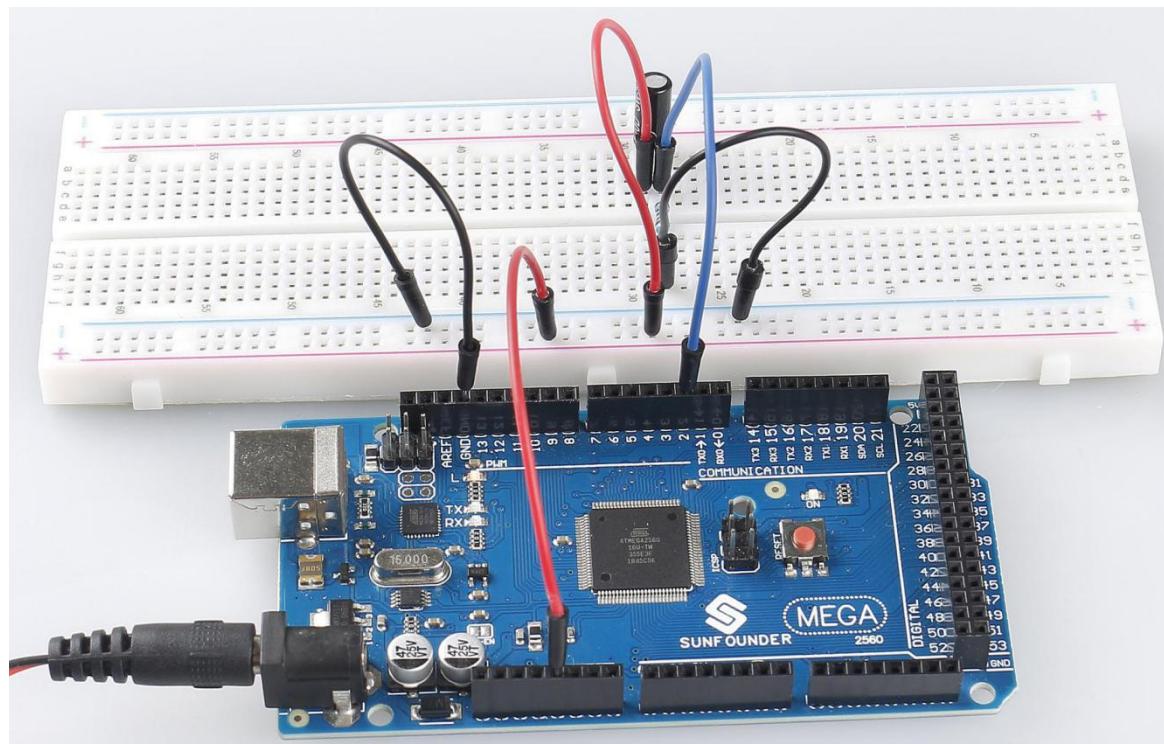
```
void setup() {
    Serial.begin(9600);
    pinMode(2, INPUT);
}

void loop() {
    int tiltState = digitalRead(2);
    Serial.println(tiltState);
```

```
delay(1);  
}
```

After the codes are uploaded to the Mega2560 board, you can open the serial monitor to see the readings of pins, which displays 「1」 or 「0」 when Tilt Switch is vertical (bringing the internal metal ball into contacting with the Wire Pins) or tilted. For detailed explanation of codes, you can turn to [Part 1-1.4 Digital Read](#).

Phenomenon Picture



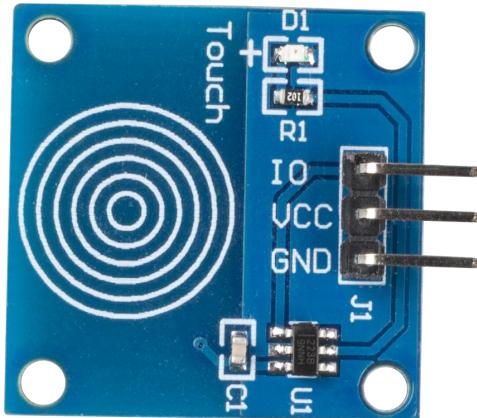
2.18 Touch Switch Module

Overview

In this lesson, you will learn about touch switch module. It can replace the traditional kinds of switch with these advantages: convenient operation, fine touch sense, precise control and least mechanical wear.

Components Required

1 * Touch Switch Module



1 * Mega 2560 Board



Several Jumper Wires



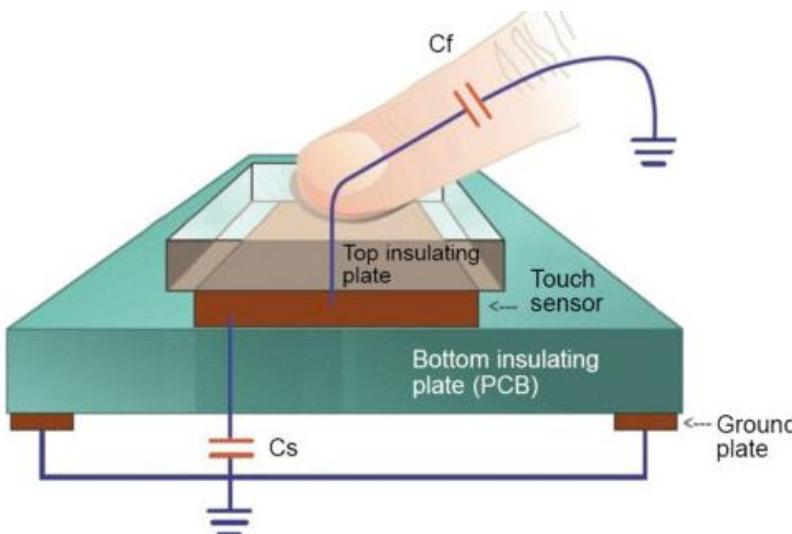
Component Introduction

Touch switch module works by detecting a change in capacitance due to influence of an external object. The touch plate is covered with insulating material, and the user does not come in contact with the electrical circuit.

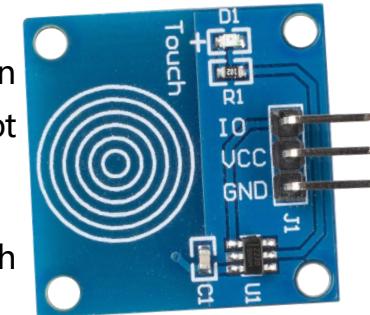
A capacitive touch switch has different layers—top insulating face plate followed by touch plate, another insulating layer and then ground plate.

In practice, a capacitive sensor can be made on a double-sided PCB by regarding one side as the touch sensor and the opposite side as ground plate of the capacitor. When power is applied across these plates, the two plates get charged. In

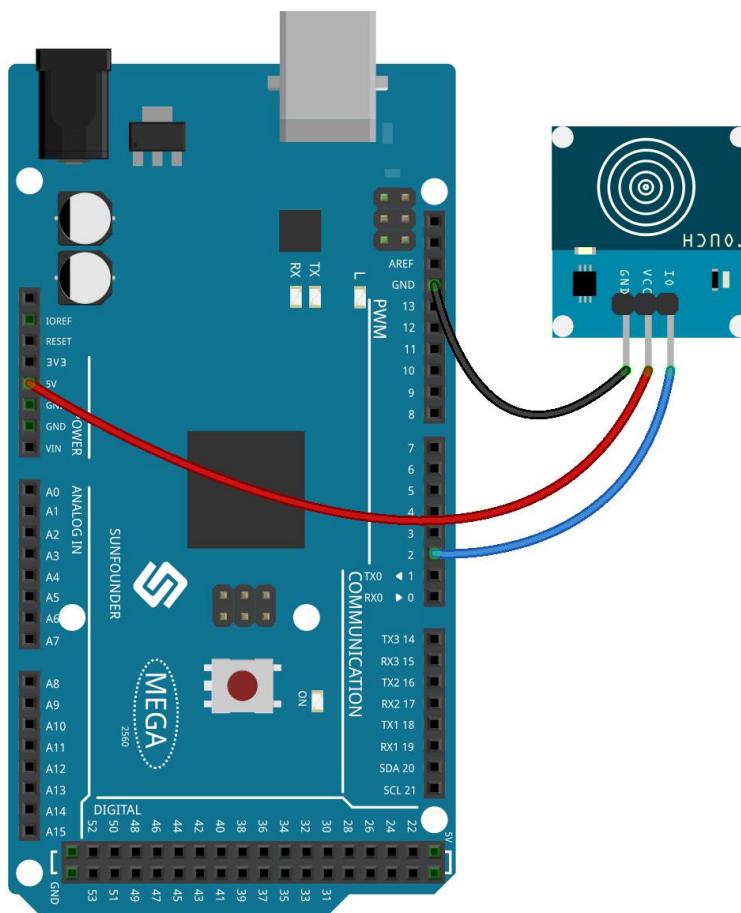
equilibrium state, the plates have the same voltage as the power source.



Cs = Capacitance between touch pad and ground plane
Cf = Additional capacitance induced by the finger

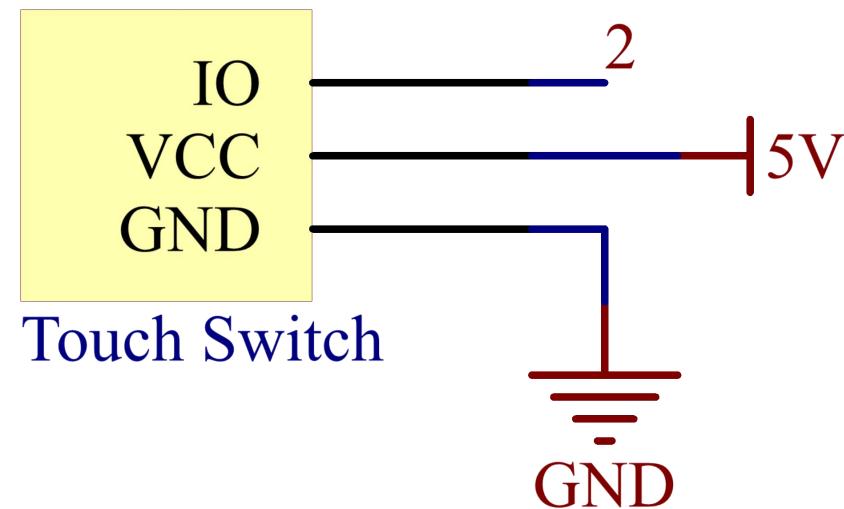


Fritzing Circuit



In this example, pin 2 is used to read the signal of Touch Switch Module.

Schematic Diagram



Touch Switch

Code

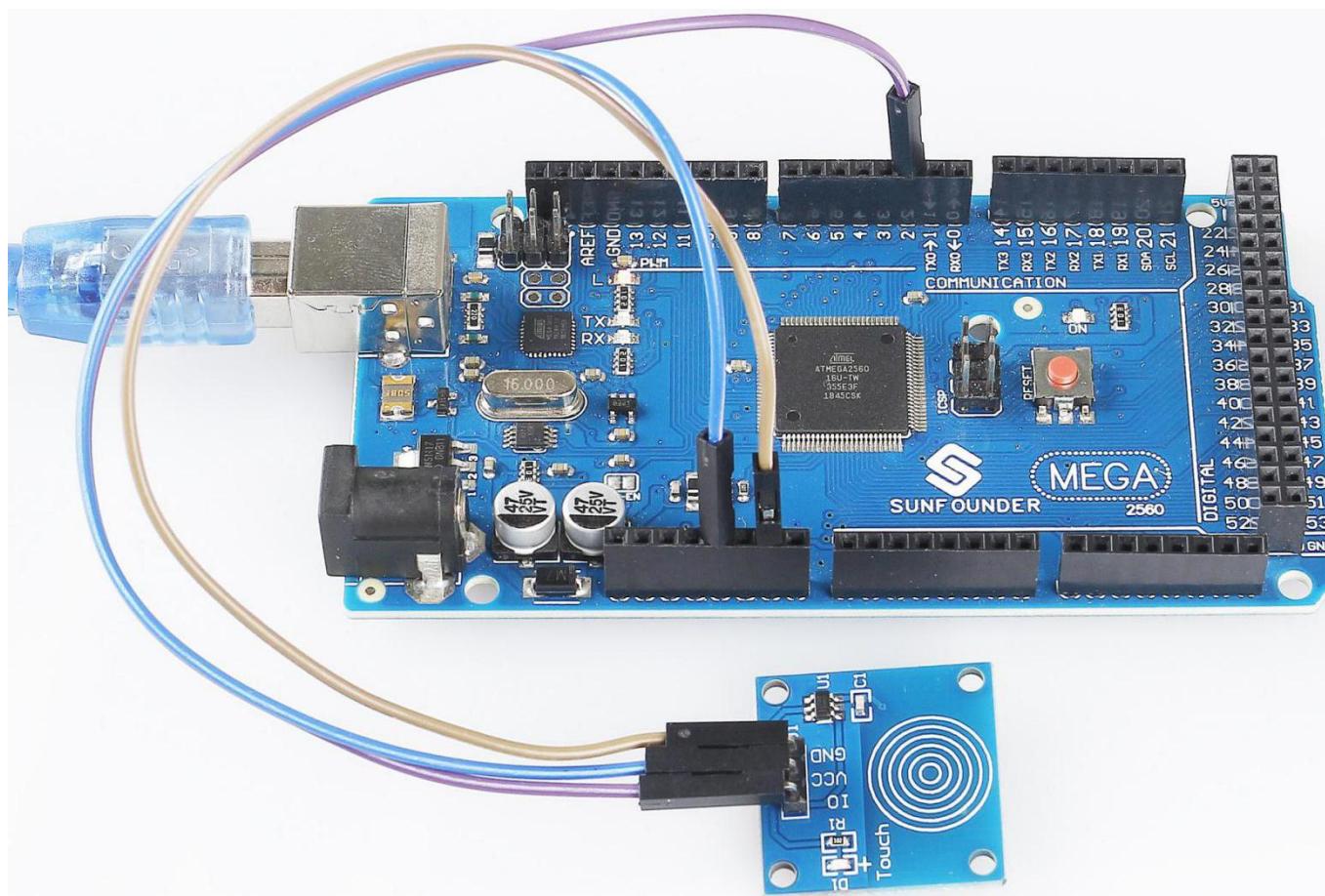
```
void setup() {  
    Serial.begin(9600);  
    pinMode(2, INPUT);  
}  
  
void loop() {  
    int touchState = digitalRead(2);  
    Serial.println(touchState);  
    delay(1);  
}
```

Uploaded the codes to the Mega2560 board, you can see the readings of pins displaying on the serial monitor.

When your finger tip touches the Touch switch module, 「1」 will be displayed on the serial monitor; and when you remove your finger, 「0」 will be displayed. As for the detailed code explanation, you need to turn to

[Part 1-1.4 Digital Read.](#)

Phenomenon Picture



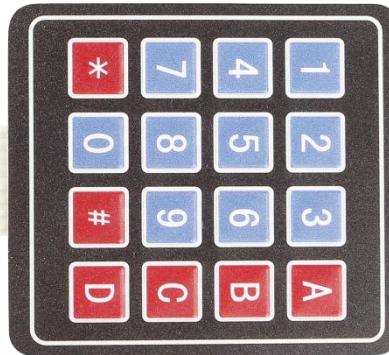
2.19 Keypad

Overview

In this lesson, you will learn to use Keypad. Keypad can be applied into various kinds of devices, including mobile phone, fax machine, microwave oven and so on. It is commonly used in user input.

Components Required

1 * 4x4 Matrix Keypad Module



1 * Mega 2560 Board

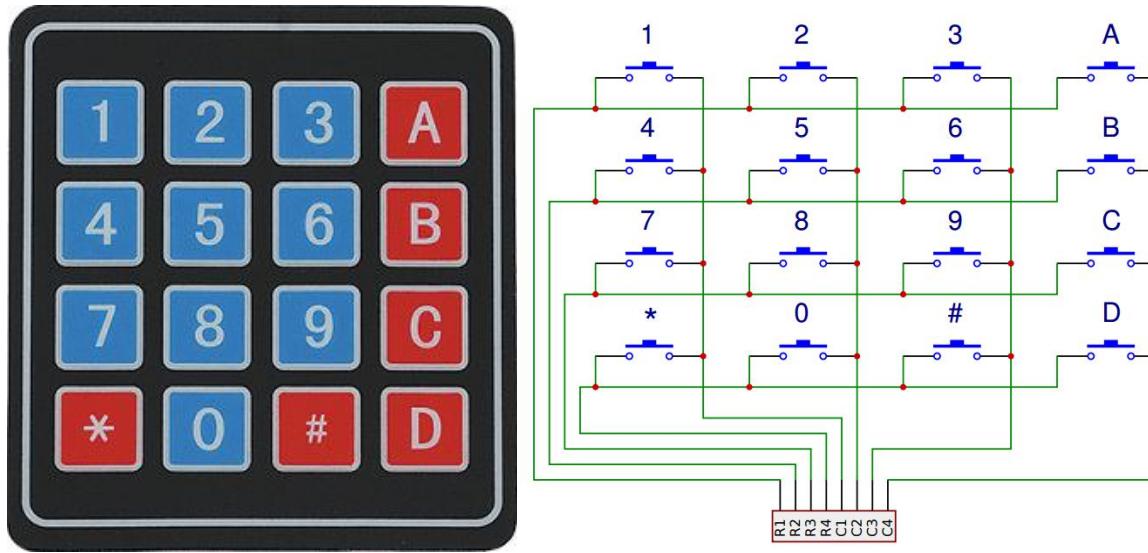


Several Jumper Wires



Component Introduction

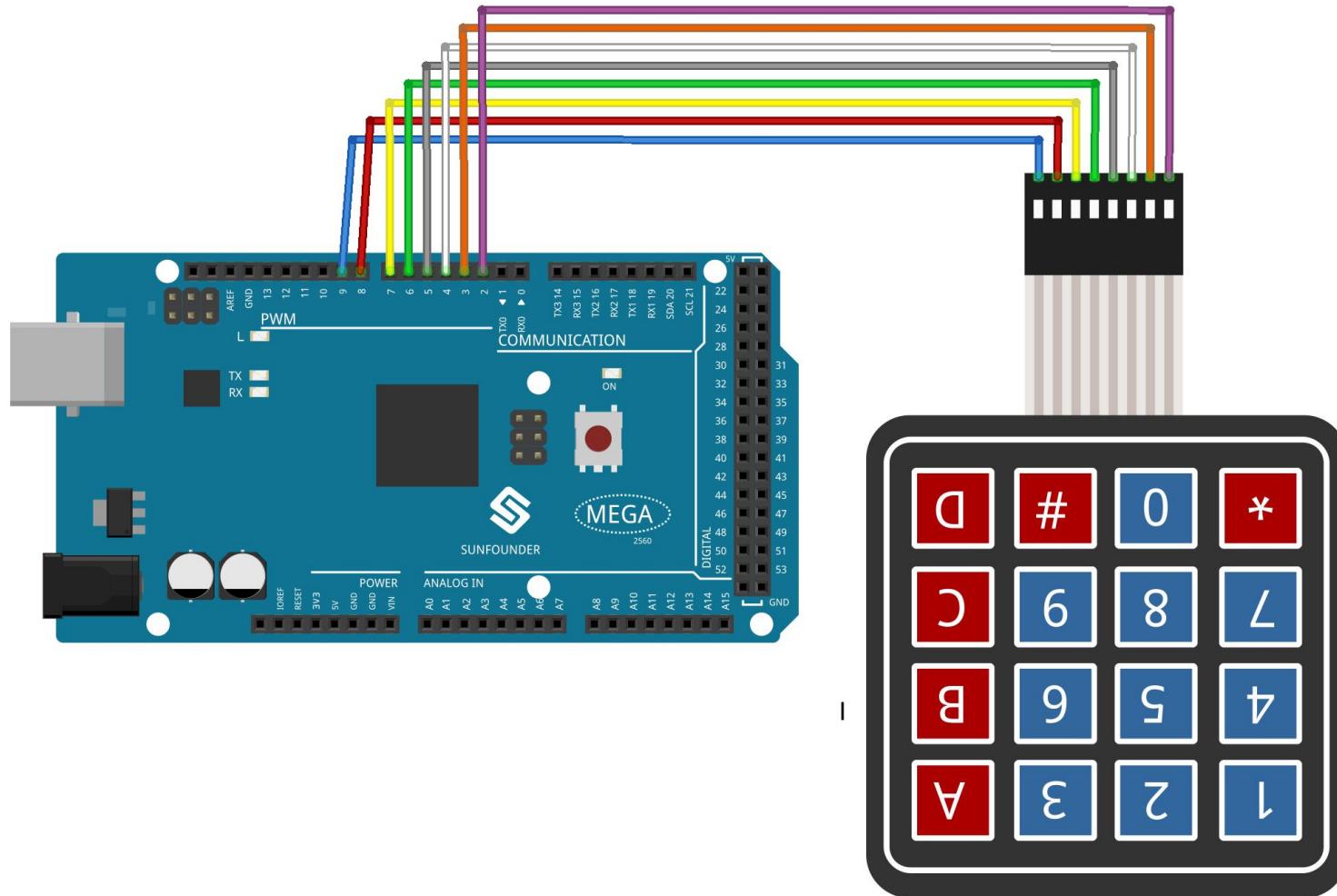
A keypad is a rectangular array of 12 or 16 OFF-(ON) buttons. Their contacts are accessed via a header suitable for connection with a ribbon cable or insertion into a printed circuit board. In some keypads, each button connects with a separate contact in the header, while all the buttons share a common ground.



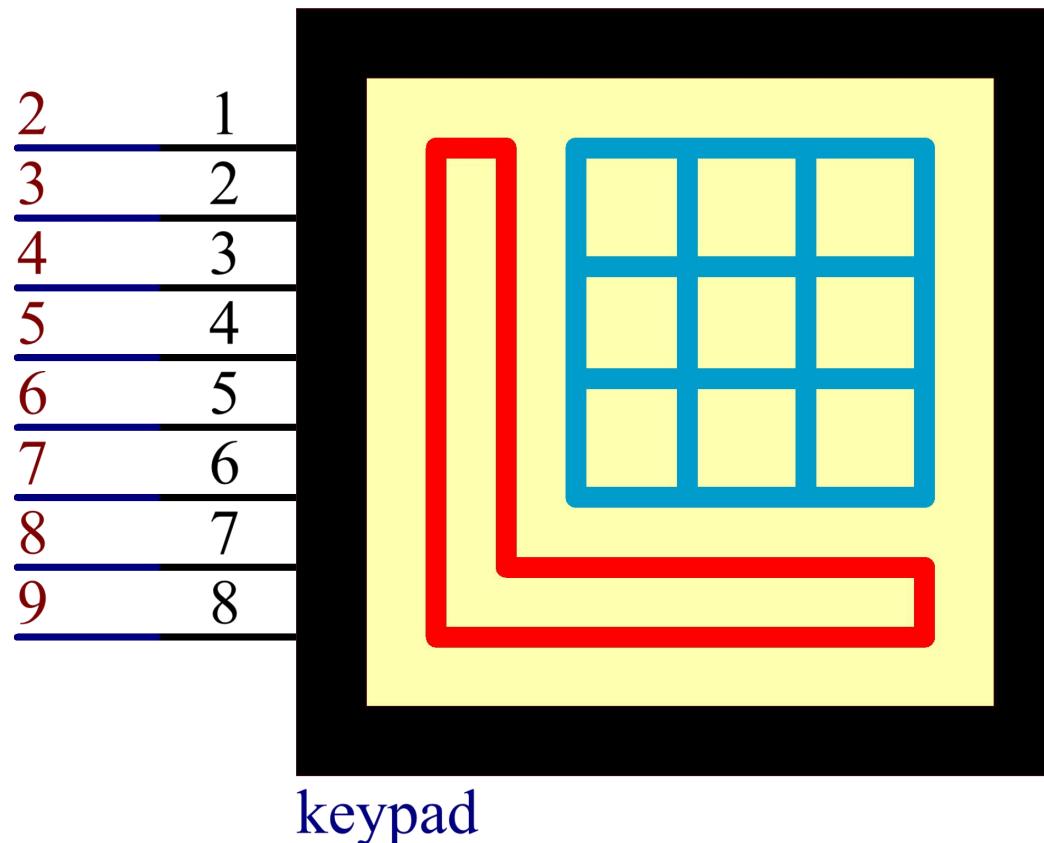
More often, the buttons are matrix encoded, meaning that each of them bridges a unique pair of conductors in a matrix. This configuration is suitable for polling by a microcontroller, which can be programmed to send an output pulse to each of the four horizontal wires in turn. During each pulse, it checks the remaining four vertical wires in sequence, to determine which one, if any, is carrying a signal. Pullup or pulldown resistors should be added to the input wires to prevent the inputs of the microcontroller from behaving unpredictably when no signal is present.

Fritzing Circuit

In this example, we extend the pins 1~8 of Keypad to connect to the digital pins 2~9.



Schematic Diagram



Code

The codes use the library Keypad.h. Please refer to [Part 4 - 4.1 Add Libraries](#) to import the library.

```
#include <Keypad.h>
const byte ROWS = 4; //four rows
const byte COLS = 4; //four columns
//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' }
};
byte rowPins[ROWS] = {2, 3, 4, 5}; //connect to the row pinouts of the keypad
byte colPins[COLS] = {6, 7, 8, 9}; //connect to the column pinouts of the keypad
//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);
    }
}
```

After uploading the codes to the Mega2560 board, on the serial monitor, you can see the value of the key currently pressed on the Keypad.

Code Analysis

By calling the Keypad.h library, you can easily use Keypad.

```
#include <Keypad.h>
```

Library Functions:

Keypad(char *userKeymap, byte *row, byte *col, byte numRows, byte numCols)

Initializes the internal keymap to be equal to userKeymap.

userKeymap: The symbols on the buttons of the keypads.

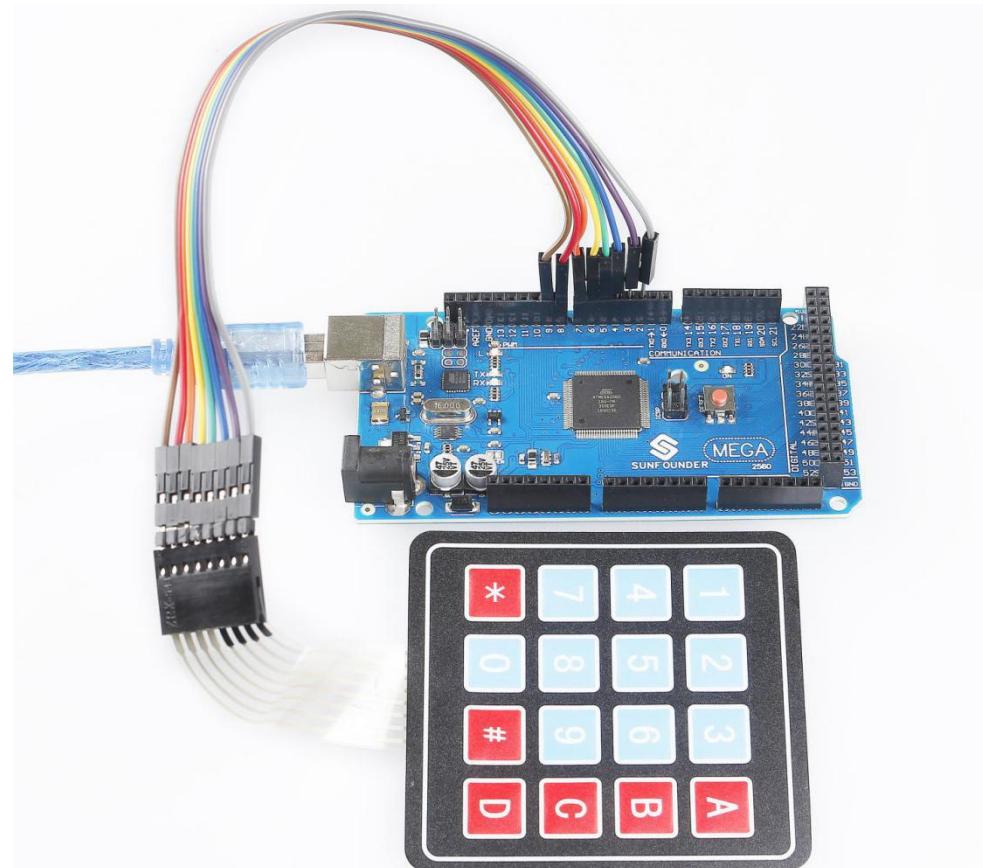
row, col: Pin configuration.

numRows, numCols: Keypad sizes.

`char getKey()`

Returns the key that is pressed, if any. This function is non-blocking.

Phenomenon Picture



2.20 IR Receiver Module

Overview

In this lesson, you will learn to use IR Receiver Module. IR Receiver is a component with photocell that is tuned to receive to infrared light. It is almost always used for remote control detection - every TV and DVD player has one of these in the front to receive 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.

Components Required

1 * IR Receiver Module



1 * IR Remote Controller



1 * Mega 2560 Board



Several Jumper Wires

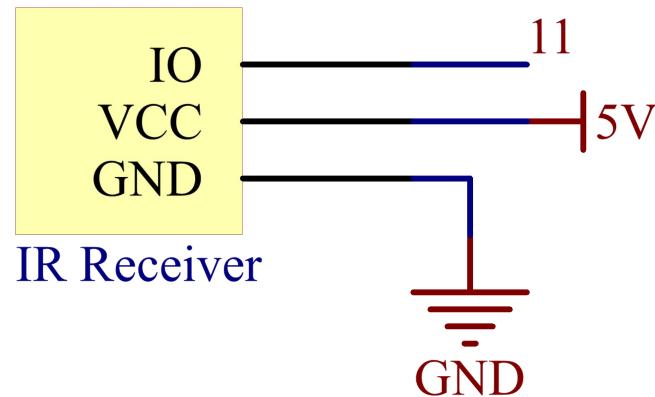


Component Introduction

An infrared-receiver is a component which receives infrared signals and can independently receive infrared ray and output signals compatible with TTL level. It's similar with a normal plastic-packaged transistor in size and is suitable for all kinds of infrared remote control and infrared transmission.

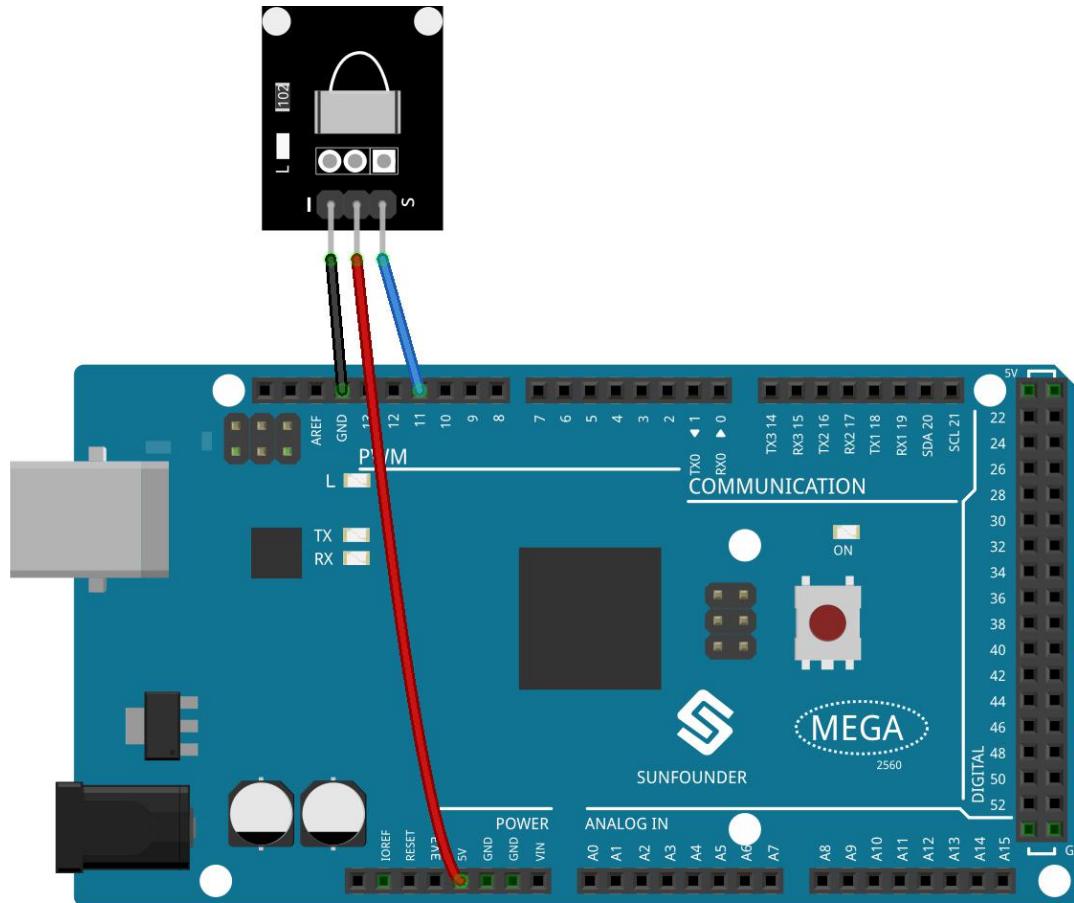


Schematic Diagram



Fritzing Circuit

In this example, we wire up the left pin (-) of IR Receiver Module to GND, the middle pin to 5V, and the right pin (S) to pin 11.



Code

The codes use the library `IRremote.h`, about how to import library, please refer to [Part 4 - 4.1 Add Libraries](#).

```
#include <IRremote.h>
const int recvPin = 11;
IRrecv irrecv(recvPin);
decode_results results;
void setup()
{
    Serial.begin(9600);
    irrecv.enableIRIn(); // Start the receiver
}
void loop() {
    if (irrecv.decode(&results)) {
        //Serial.println(results.value,HEX);
        if (decodeKeyValue(results.value)!="ERROR"){
            Serial.println(decodeKeyValue(results.value));
        }
        irrecv.resume(); // Receive the next value
    }
}
```

After uploading the codes to the Mega2560 board, you can see that the current value of the pressed button of IR Remote Controller displays on the serial monitor.

Code Analysis

There are two important parts to notice in this program.

- ①The code uses an extra file decodeKeyValue.ino to decode the values in class decode_result into key value.The file will be opened together with the main file.
- ②IR Remote function is achieved by calling IRemote.h library related functions.

```
#include <IRremote.h>
```

Library Functions:

IRrecv(int recvpin)

Create IRrecv object to control a IR Receiver module.

decode_result

In this kit, results are usually 8-digit hexadecimal numbers starting with 00FF. You can check decodeKeyValue.ino file in the sample file.

```
void enableIRIn()
```

Initialize the IR receiver module.

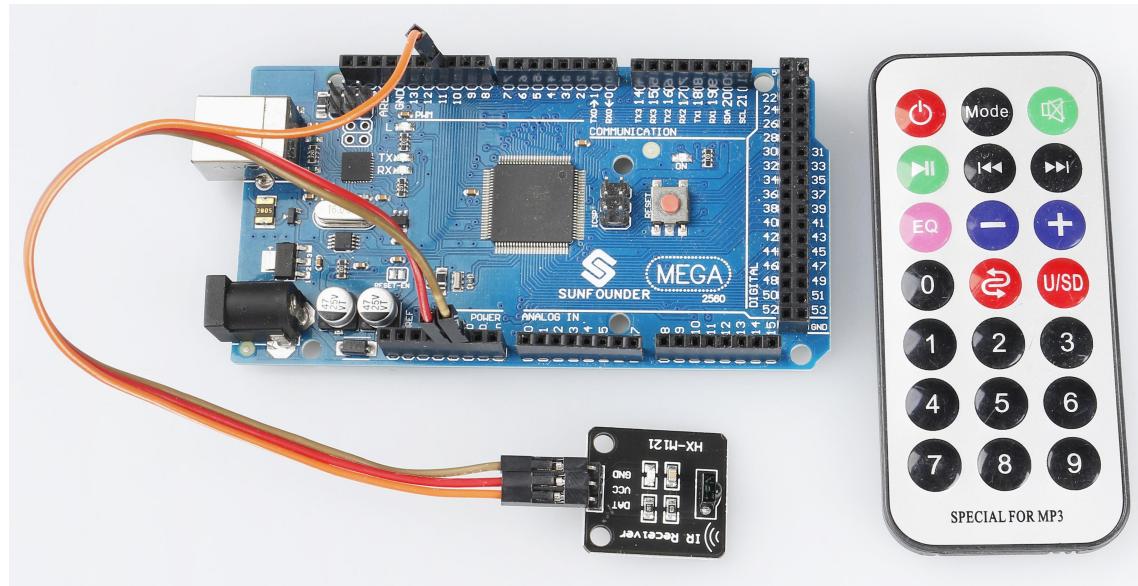
```
int decode(decode_results *results);
```

Decodes the received IR message. Returns 0 if no data ready, 1 if data ready. Results of decoding are stored in `results`.

```
void resume()
```

Restart for receiving an other value.

Phenomenon Picture

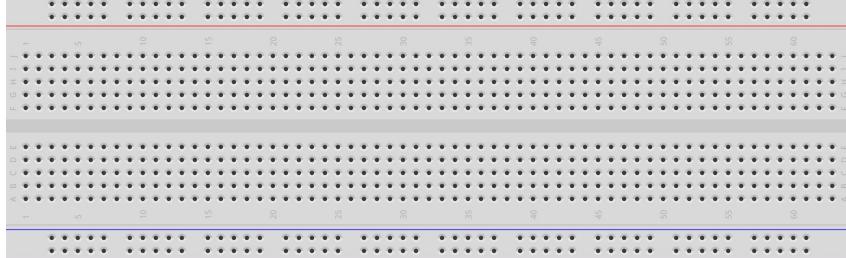


2.21 Relay Module

Overview

In this lesson, you will learn about Relay Module.

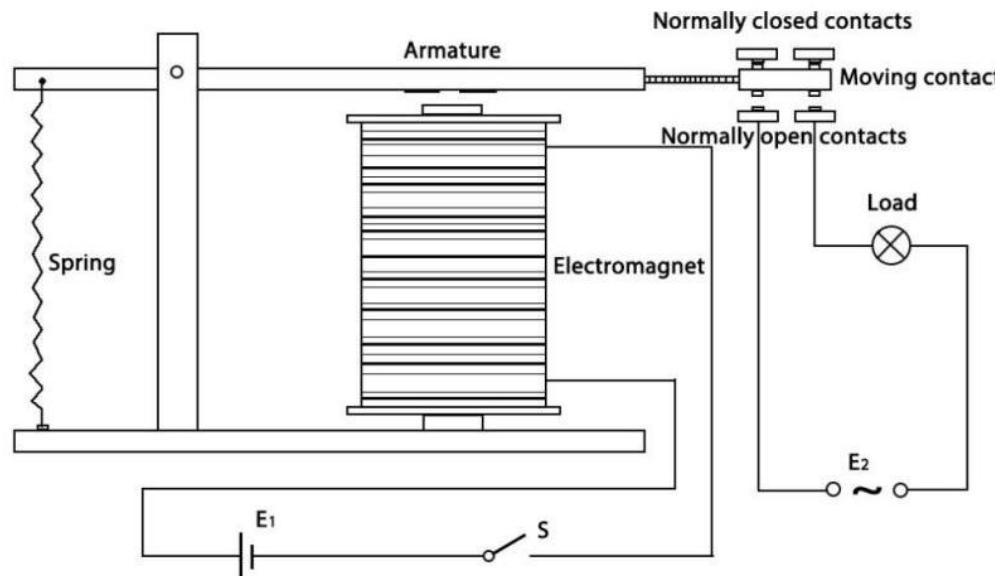
Components Required

1 * Relay Module 	1 * LED 	Breadboard power module 	1 * 220 ohm resistor  Several Jumper Wires 
1 * Mega 2560 Board 	1 * Breadboard 		

Component Introduction

As we may know, relay is a device which is used to provide connection between two or more points or devices in response to the input signal applied. In other words, relays provide isolation between the controller and the device as devices may work on AC as well as on DC. However, they receive signals from a microcontroller which works on DC hence requiring a relay to bridge the gap. Relay is extremely useful when you need to control a large amount of current or voltage with small electrical signal.

There are 5 parts in every relay:



1. **Electromagnet** – It consists of an iron core wound by coil of wires. When electricity is passed through, it becomes magnetic. Therefore, it is called electromagnet.
2. **Armature** – The movable magnetic strip is known as armature. When current flows through them, the coil is energized thus producing a magnetic field which is used to make or break the normally open (N/O) or normally close (N/C) points. And the armature can be moved with direct current (DC) as well as alternating current (AC).
3. **Spring** – When no currents flow through the coil on the electromagnet, the spring pulls the armature away so the circuit cannot be completed.
4. Set of electrical **contacts** – There are two contact points:
 - Normally open - connected when the relay is activated, and disconnected when it is inactive.
 - Normally close – not connected when the relay is activated, and connected when it is inactive.

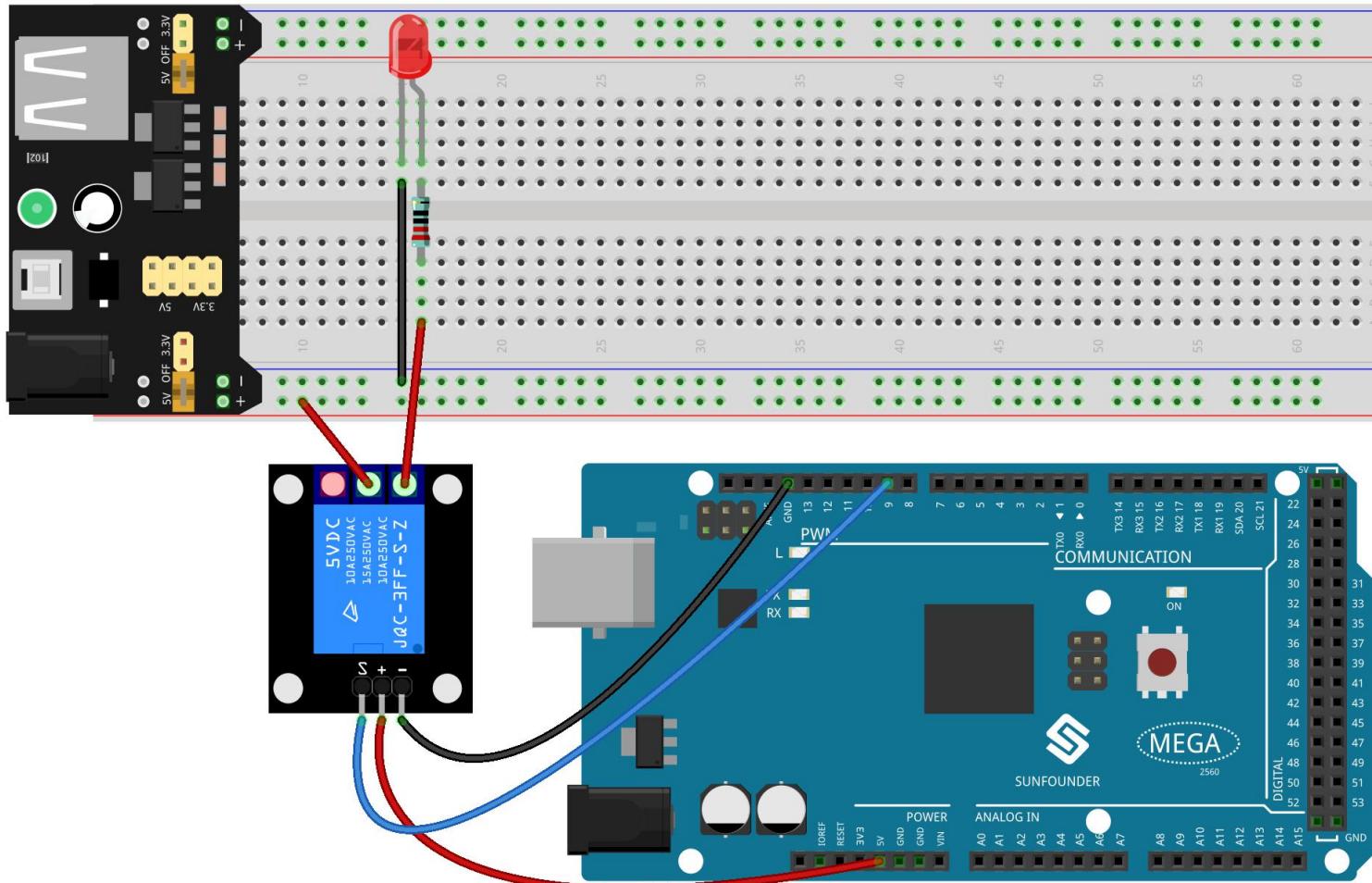
5. Molded frame – Relays are covered with plastic for protection.

Working of Relay

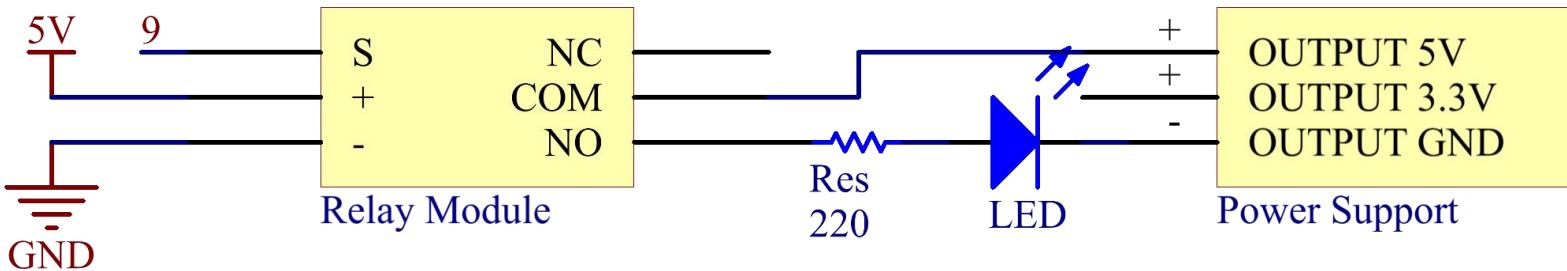
The working principle of relay is simple. When power is supplied to the relay, currents start flowing through the control coil; as a result, the electromagnet starts energizing. Then the armature is attracted to the coil, pulling down the moving contact together thus connecting with the normally open contacts. So the circuit with the load is energized. Then breaking the circuit would be a similar case, as the moving contact will be pulled up to the normally closed contacts under the force of the spring. In this way, the switching on and off of the relay can control the state of a load circuit.

Fritzing Circuit

In this example, we use Power Supply Module to power the load, use LED as an example.



Schematic Diagram



Code

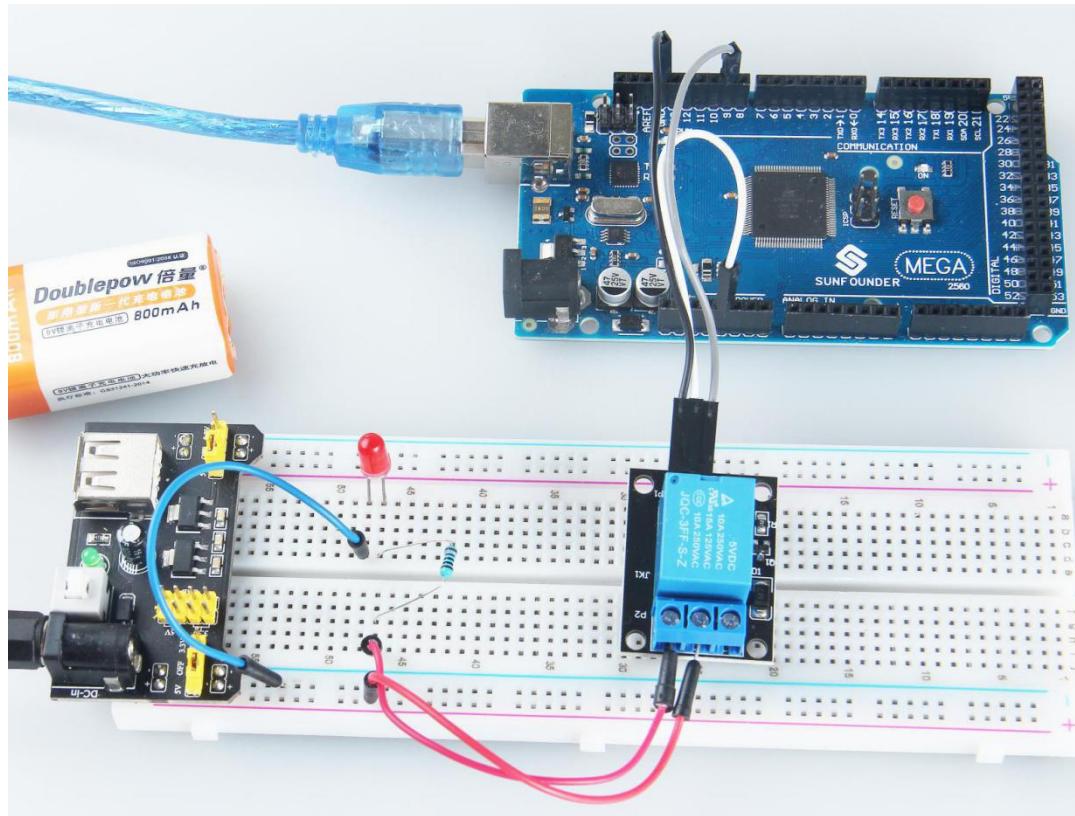
```
const int relayPin = 9;

void setup() {
  pinMode(relayPin, OUTPUT);
}

void loop() {
  digitalWrite(relayPin, HIGH);
  delay(1000);
  digitalWrite(relayPin, LOW);
  delay(1000);
}
```

Once the codes are uploaded to the Mega2560 board, you can see that the Relay Module controls the closing and breaking of the external circuit, which will change its working state a second. For detailed code explanation, refer to [Part 1-1.2 Digital Write](#).

Phenomenon Picture



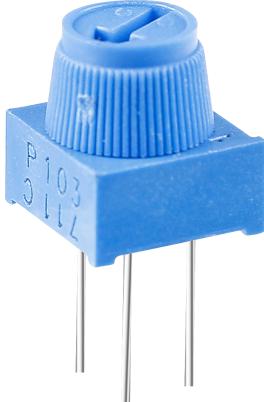
2.22 Potentiometer

Overview

In this lesson, you will learn about Potentiometer. Potentiometer is a resistor component with 3 terminals and its resistance value can be adjusted according to some regular variation.

Components Required

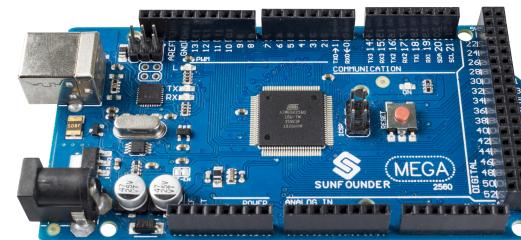
1 * potentiometer



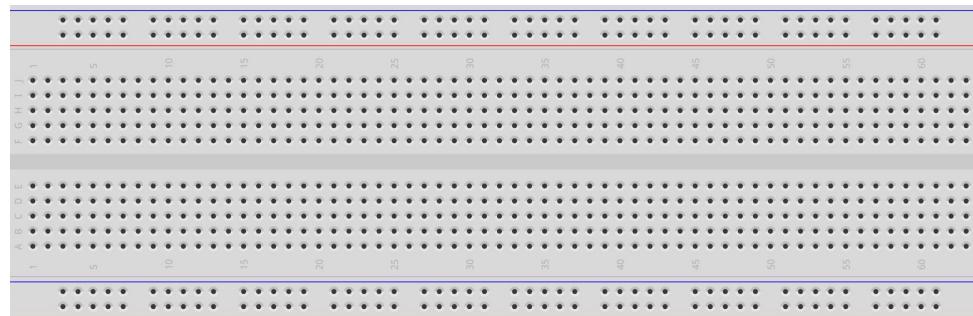
Several Jumper Wires

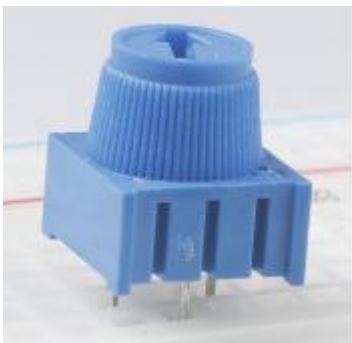


1 * Mega 2560 Board



1 * Breadboard

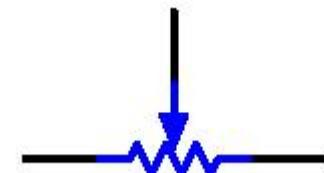




Component Introduction

Potentiometer usually consists of resistor and movable brush. When the brush is moving along the resistor, there is a certain resistance or voltage output depending on the displacement.

The functions of the potentiometer in the circuit are as follows:



1. Serving as a voltage divider

Potentiometer is a continuously adjustable resistor. When you adjust the shaft or sliding handle of the potentiometer, the movable contact will slide on the resistor. At this point, a voltage can be output depending on the voltage applied onto the potentiometer and the angle the movable arm has rotated to or the distance it moves.

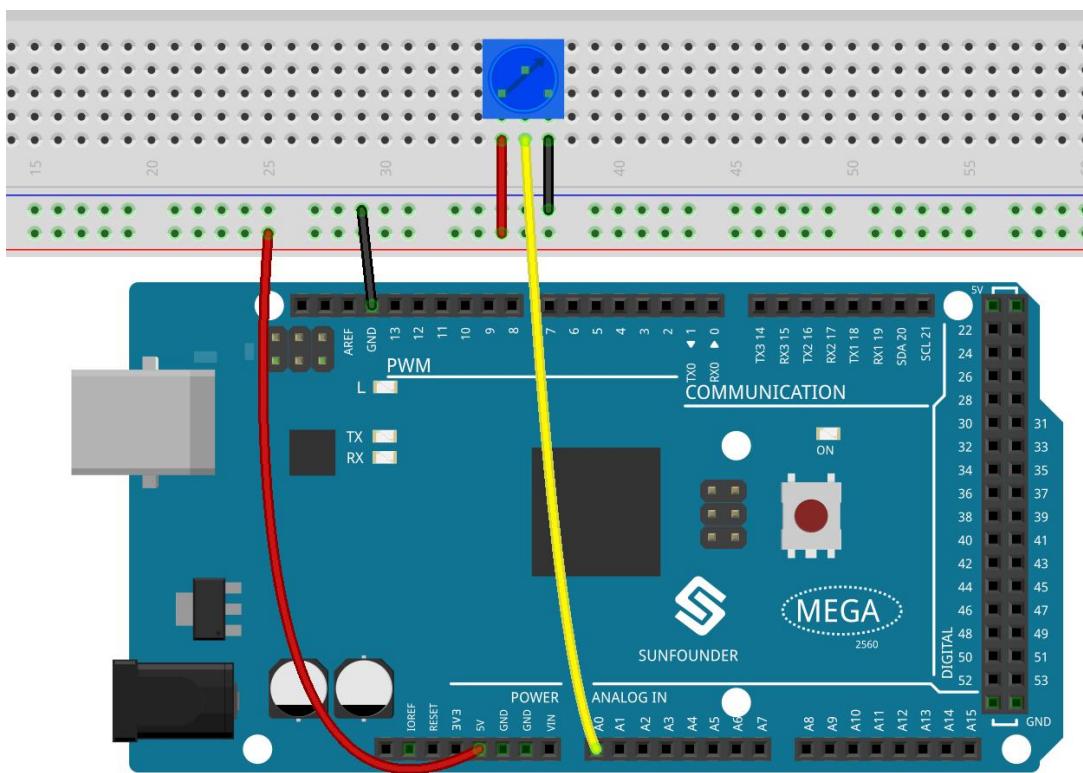
2. Serving as a rheostat

When the potentiometer is used as a rheostat, connect the middle pin and one of the other 2 pins in the circuit. Thus you can get a smoothly and continuously changed resistance value cused by moving contact.

3. Serving as a current controller

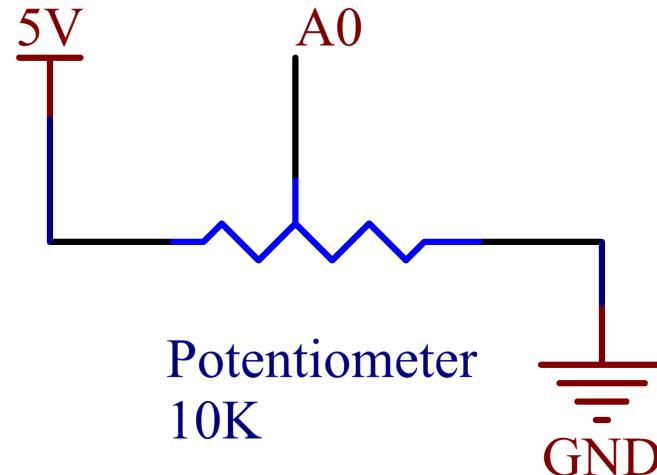
When the potentiometer acts as a current controller, the sliding contact terminal must be connected as one of the output terminals.

Fritzing Circuit



In this example, we use the analog pin (A0) to read the value of the potentiometer. By rotating the axis of the potentiometer, you can change the distribution of resistance among these three pins, changing the voltage on the middle pin. When the resistance between the middle and a outside pin connected to 5V is close to zero (and the resistance between the middle and the other outside pin is close to $10k\Omega$), the voltage at the middle pin is close to 5 V. The reverse operation (the resistance between the middle and a outside pin connected to 5V is close to $10k \Omega$) will make the voltage at the middle pin be close to 0V.

Schematic Diagram



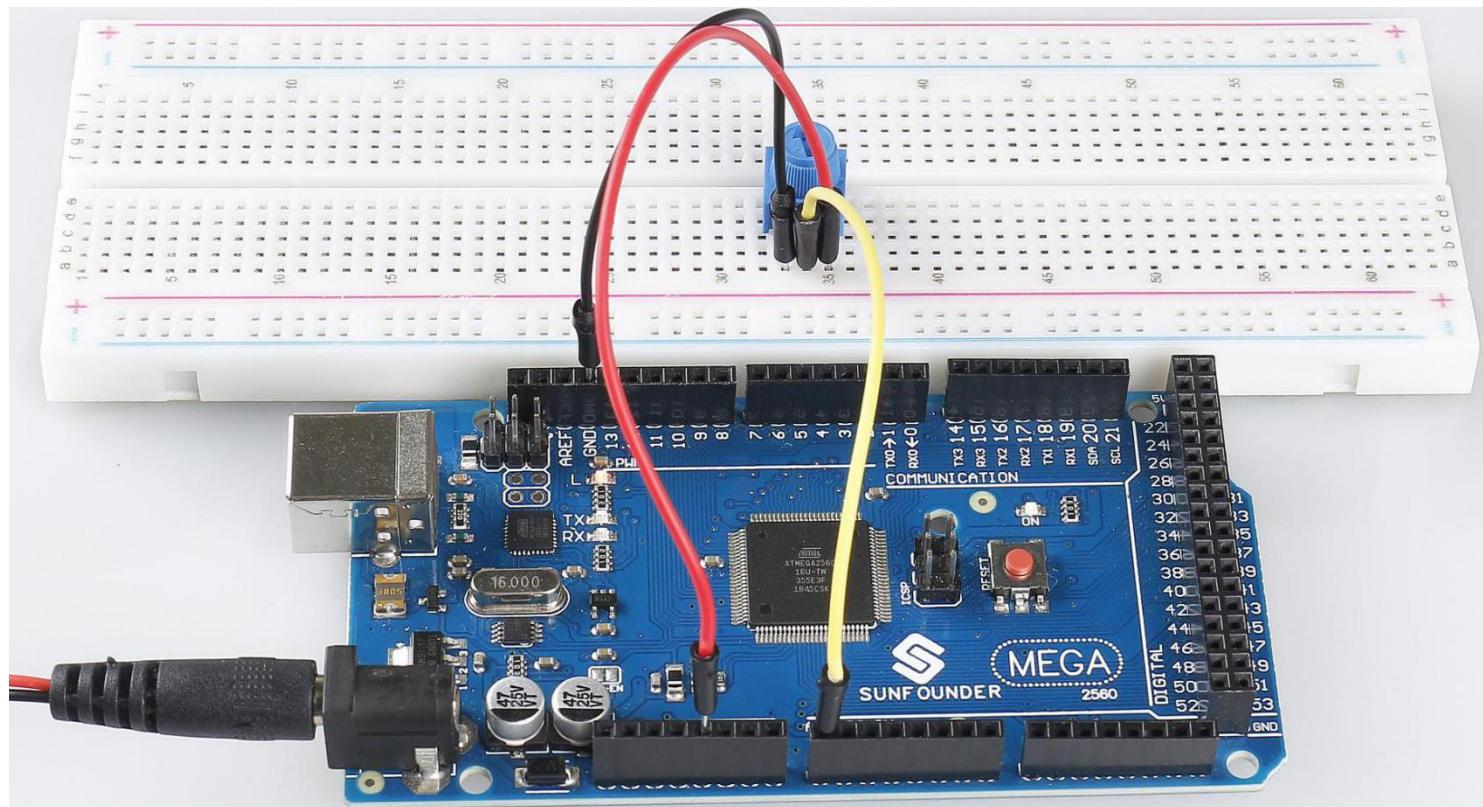
Code

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

void loop() {
  int sensorValue = analogRead(A0);
  Serial.println(sensorValue);
  delay(1);
}
```

After uploading the codes to the Mega2560 board, you can open the serial monitor to see the reading value of the pin. When rotating the axis of the potentiometer, the serial port monitor will print the value 「0」 ~ 「1023」 . For the detailed explanation of code, turn to check [Part 1-1.5 Analog Read](#).

Phenomenon Picture



2.23 Joystick Module

Overview

In this lesson, you will learn something about Joystick. The basic idea of a joystick is to translate the movement of a stick into electronic information that a computer can process. It can be applied to work as the controller of devices, such as robot.

Components Required

1 * Joystick Module



1 * Mega 2560 Board



Several Jumper Wires

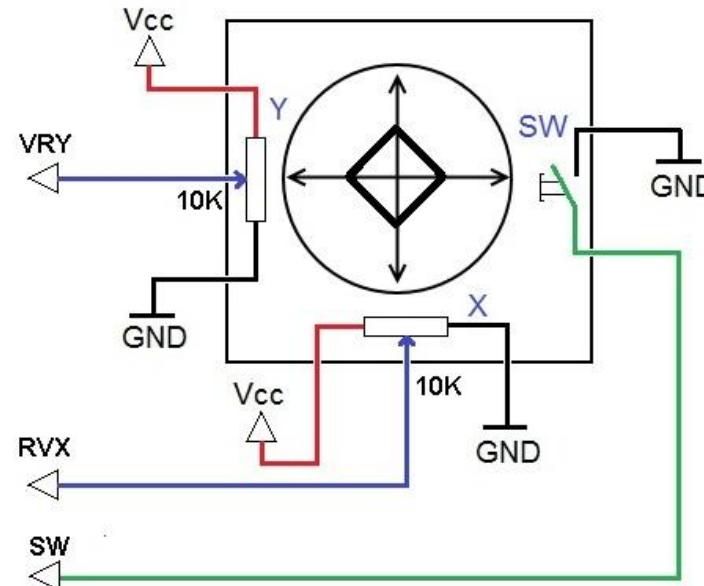


Component Introduction

In order to communicate a full range of motion to the computer, a joystick needs to measure the stick's position on two axes -- the X-axis (left to right) and the Y-axis (up and down). Just as in basic geometry, the X-Y coordinates pinpoint the stick's position exactly.

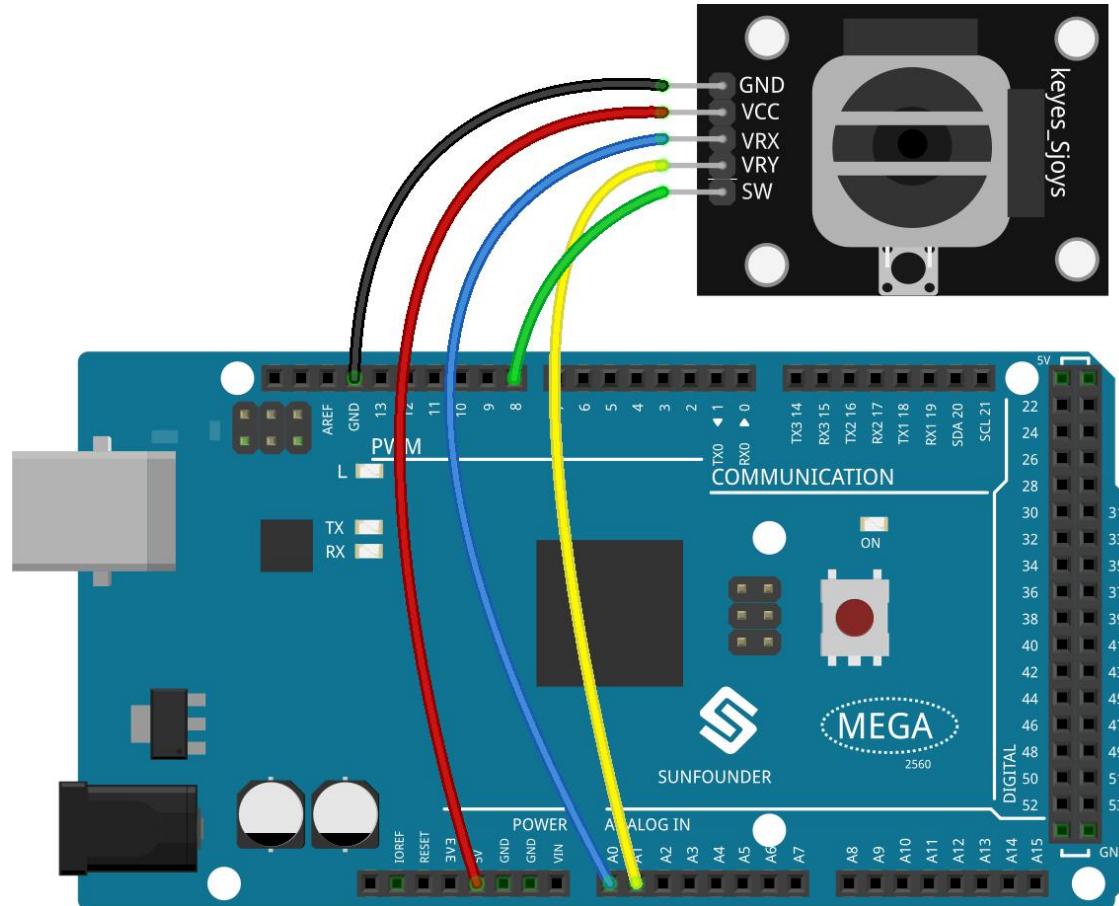
To determine the location of the stick, the joystick control system simply monitors the position of each shaft. The conventional analog joystick design does this with two potentiometers, or variable resistors.

The joystick also has a digital input that is actuated when the joystick is pressed down.

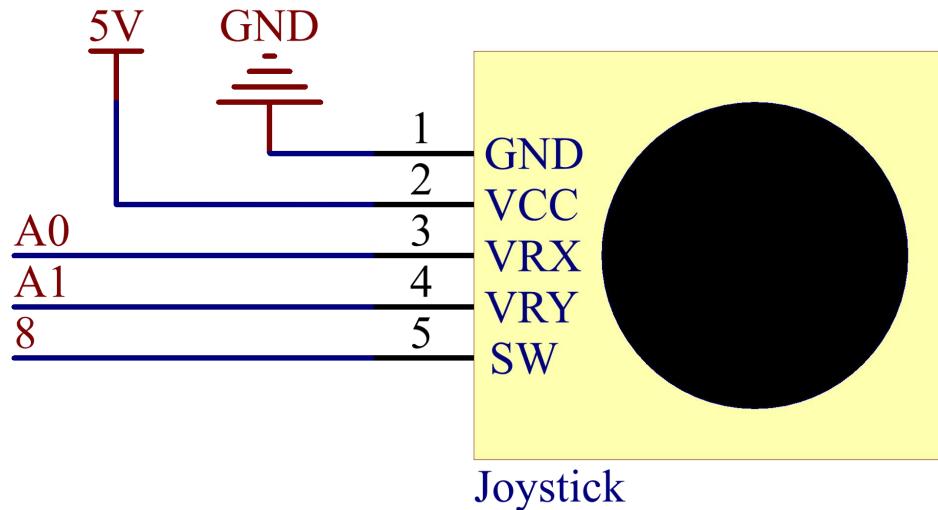


Fritzing Circuit

In this example, we get the GND of the Joystick extended to connect with GND, VCC with 5V, VRX with pin A0. After that, we make VRY connect with pin A1, SW connect with pin 8.



Schematic Diagram



Code

```
const int xPin = A0; //the VRX attach to
const int yPin = A1; //the VRY attach to
const int swPin = 8; //the SW attach to

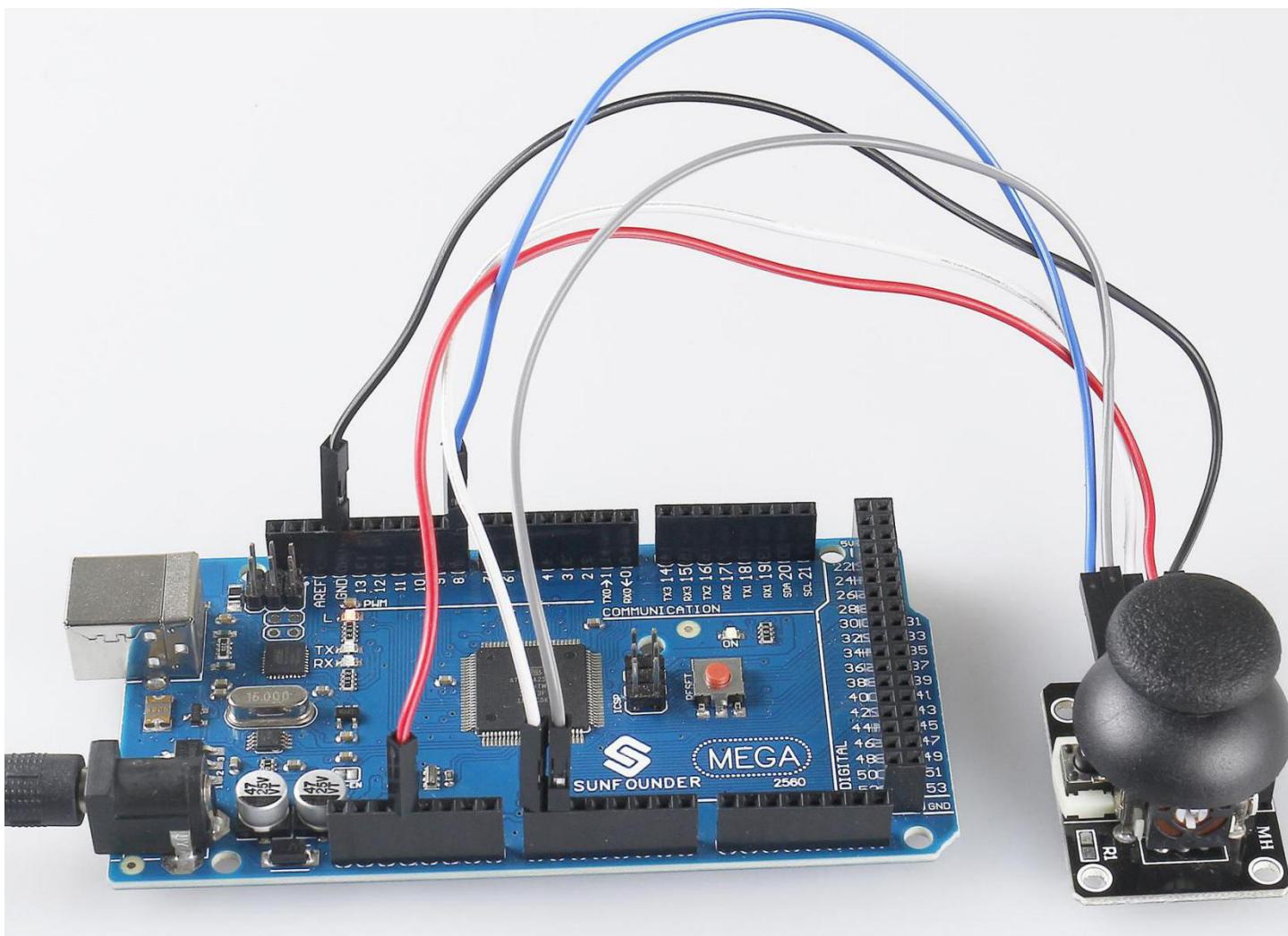
void setup()
{
    pinMode(swPin, INPUT); //set the SW pin to INPUT
```

```
digitalWrite(swPin, HIGH); //And initial value is HIGH
Serial.begin(9600);
}

void loop()
{
    Serial.print("X: ");
    Serial.print(analogRead(xPin), DEC); // print the value of VRX in DEC
    Serial.print("|Y: ");
    Serial.print(analogRead(yPin), DEC); // print the value of VRX in DEC
    Serial.print("|Z: ");
    Serial.println(digitalRead(swPin)); // print the value of SW
    delay(500);
}
```

Uploaded the codes to the Mega2560 board, you can open the serial monitor to see readings on the X-axis and Y-axis of Joystick, as well as the button status of Z-axis. The values of the X-axis and Y-axis are the analog values, which vary within the range [0] ~ [1023]. The Z-axis shows numerical value and the state is either [1] or [0]. Refer to [Part 1-1.5 Analog Read](#) and [1.4 Digital Read](#) to check the code explanation.

Phenomenon Picture



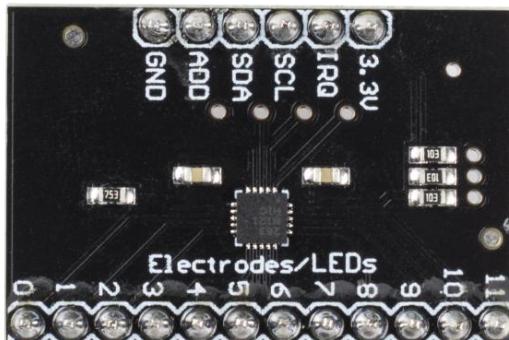
2.24 MPR121 Module

Overview

In this lesson, you will learn how to use MPR121. It's a good option when you want to add a lot of touch switches to your project. The electrode of MPR121 can be extended with a conductor. If you connect a wire to a banana, you can turn the banana into a touch switch, thus realizing projects such as fruit piano.

Components Required

1 * MPR121



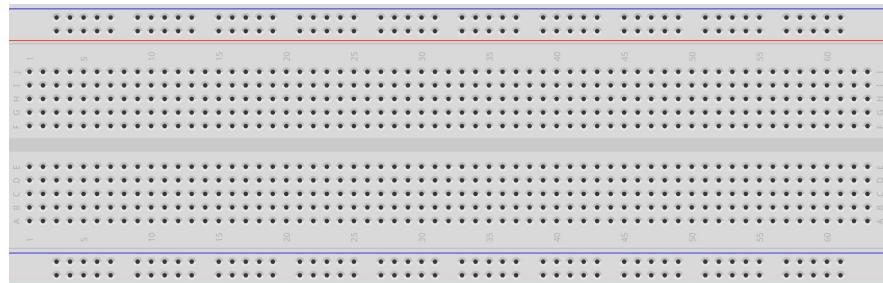
Several Jumper Wires

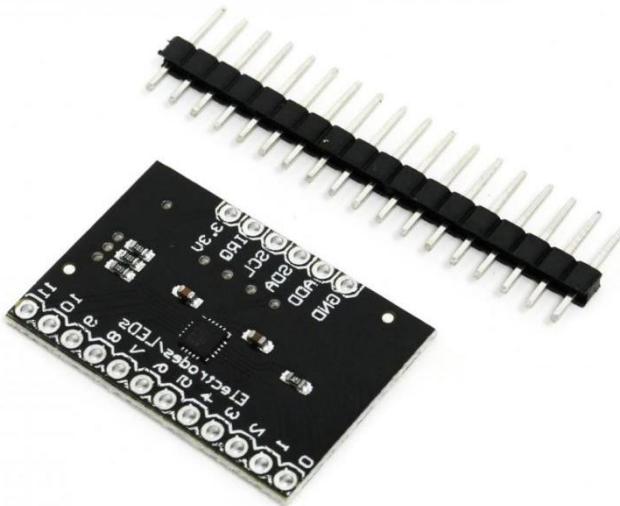


1 * Mega 2560 Board



1 * Breadboard





Component Introduction

Add lots of touch sensors to your next project with this easy-to-use 12-channel capacitive touch sensor breakout board, starring the MPR121. This chip can handle up to 12 individual touch pads.

The MPR121 has support for only I₂C, which can be implemented with nearly any microcontroller. You can select one of 4 addresses with the ADDR pin, for a total of 48 capacitive touch pads on one I₂C 2-wire bus. Using this chip is a lot easier than doing the capacitive sensing with analog inputs: it handles all the filtering for you and can be configured for more/less sensitivity.

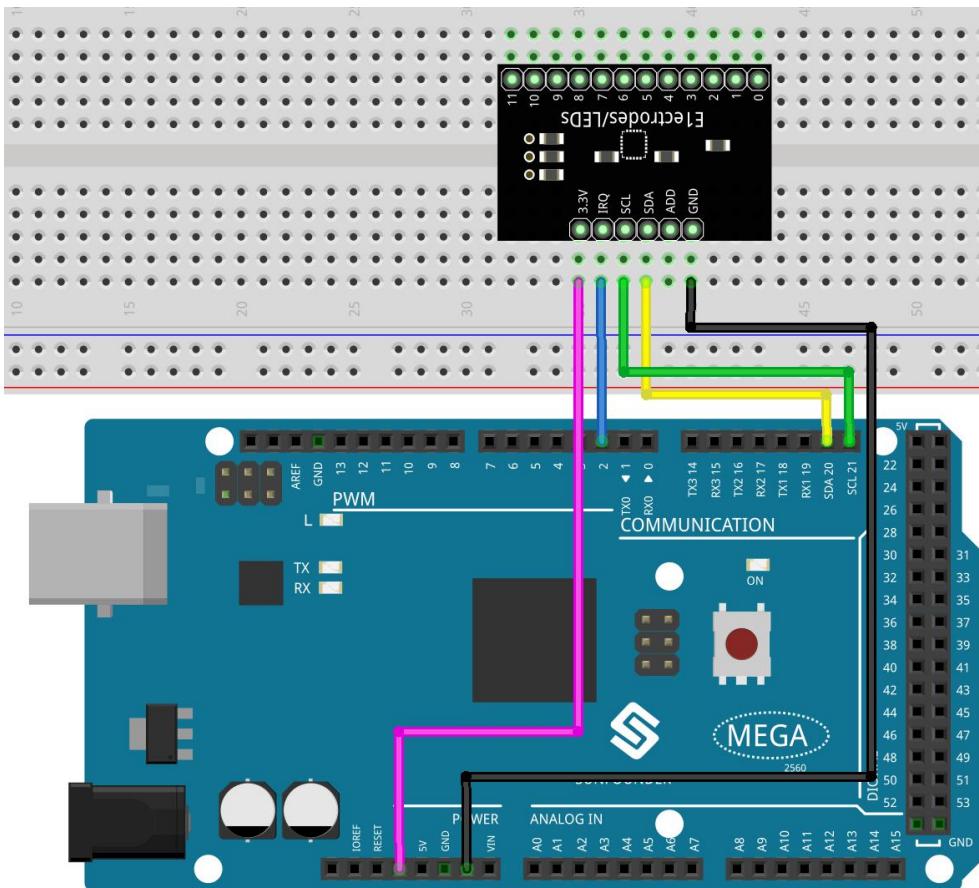
When the MPR121 senses a change, it pulls an interrupt pin LOW. The control board going to check that pin to see if it is LOW during the loop. To do this, this sensor also needs access to another digital pin.

Electrodes

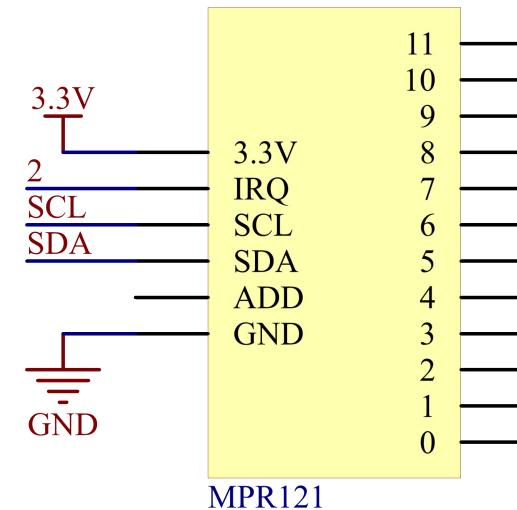
Electrode is a touch sensor. Typically, electrodes can just be some piece of metal, or a wire. But sometimes depending on the length of our wire, or the material the electrode is on, it can make triggering the sensor difficult. For this reason, the MPR121 allows you to configure what is needed to trigger and untrigger an electrode.

Fritzing Circuit

In this example, we insert MPR121 into the breadboard. Get the GND of MPR121 connected to GND, 3.3V to 3V3, IRQ to the digital pin 2, SCL to the pin SCL(21), and SDA to the pin SDA(20). There are 12 electrodes for touch sensing. Note: MPR121 is powered by 3.3V, not 5V.



Schematic Diagram



Code

The codes use the MPR121_JM.h library. Turn to [Part 4 - 4.1 Add Libraries](#) for information on how to import Libraries.

```
#include <MPR121_JM.h>
MPR121 mpr121(2, 0x2A, 0x1F);
boolean touchStates[12];
void setup() {
    mpr121.mpr121_setup();
    Serial.begin(9600);
}
void loop() {
    if (!mpr121.checkInterrupt())
    {
        int touched = mpr121.readTouchInputs();
        for (int i = 0; i < 12; i++) {
            if (touched & (1 << i)) {touchStates[i] = 1;}
            else {touchStates[i] = 0;}
        }
        for (int i = 0; i < 12; i++)
        {Serial.print(touchStates[i]);}
        Serial.println();
    }
}
```

```
 }  
}
```

After uploading the codes to the Mega2560 board, the touch state of pins of MPR121 「1」 and 「0」 will be recorded in a 12 - bit boolean type of array that will be printed on the serial monitor.

Code Analysis

The function of the module is included in the library MPR121_JM.h.

```
#include <MPR121_JM.h>
```

Library Functions:

```
MPR121(int irqpin,uint8_t touThresh,uint8_t relThresh)
```

Creates a new instance of the MPR121.

irqpin: the interrupt request pin.

touThresh: Touch threshold, the electrode is recognized as a threshold of the 「Touch」 state.

relThresh: Release threshold, the electrode is recognized as a threshold of the 「Release」 state.

The range of the electrode data value is 0~255. For typical touch application, the threshold value can be in range 0x05~0x30 for example. The smaller the value, the more sensitive it is. The setting of the threshold is depended on the actual application.Typically the touch threshold is a little bigger than the release threshold to touch debounce and hysteresis.

void mpr121_setup()

Setup MPR121 module.

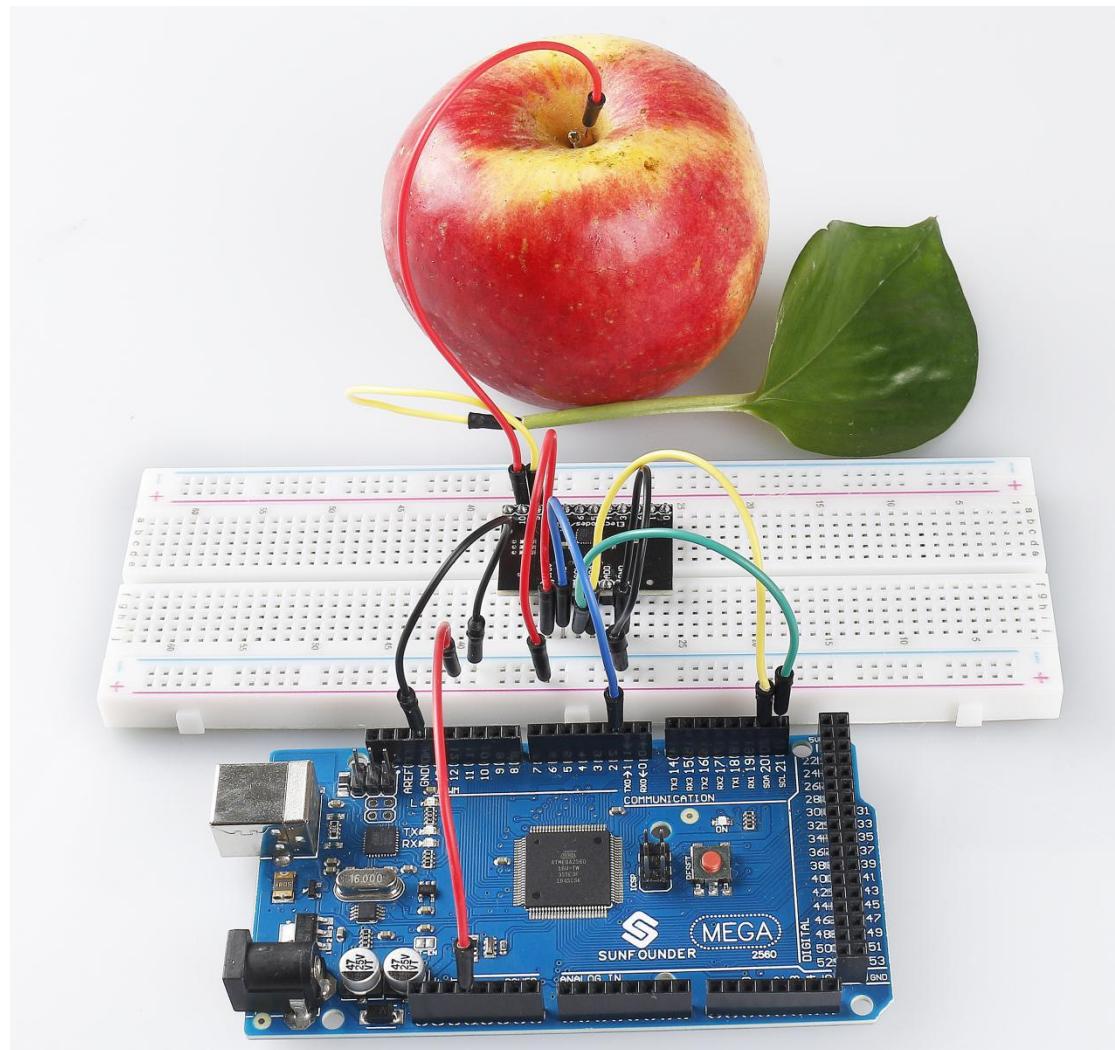
bool checkInterrupt()

Make interrupt judgment, when the electrode state changes, the function returns a Boolean value 「0」 .

uint16_t readTouchInputs()

The touch state of the electrode produces a Boolean value. The function accumulates the Boolean values generated by all the electrodes in sequence into a 12-bit binary number as the return value. If the first and eleventh electrodes are touched, 「100000000010」 is returned.

Phenomenon Picture



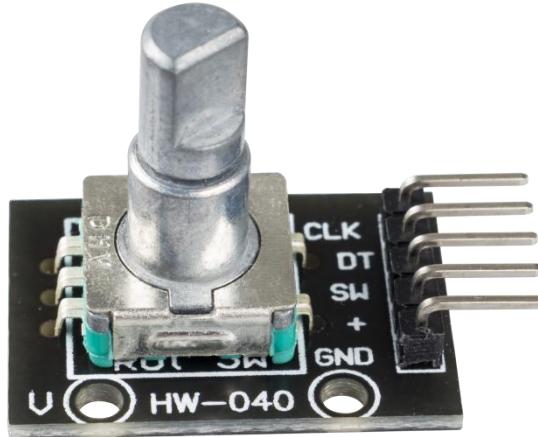
2.25 Rotary Encoder Module

Overview

In this lesson, you will learn about Rotary Encoder. A rotary encoder is an electronic switch with a set of regular pulses in strictly timing sequence. When used with IC, it can achieve increment, decrement, page turning and other operations such as mouse scrolling, menu selection, and so on.

Components Required

1 * Rotary Encoder



1 * Mega 2560 Board



Several Jumper Wires





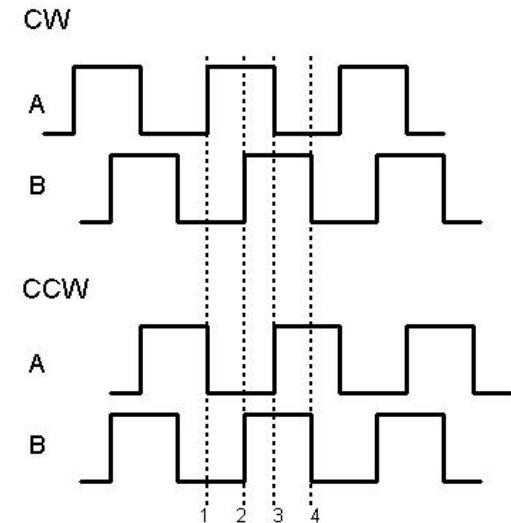
Component Introduction

There are mainly two types of rotary encoders: absolute and incremental (relative) encoders. An incremental one is used in this experiment.

Most rotary encoders have 5 pins with three functions of turning left & right and pressing down. Pin 1 and pin 2 are switch wiring terminals used to press. Pin 4 is generally connected to ground. Pin 3 and pin 5 are first connected to a pull-up resistor and then to VCC and they can generate two-phase square waves whose phase difference is 90° . Usually these

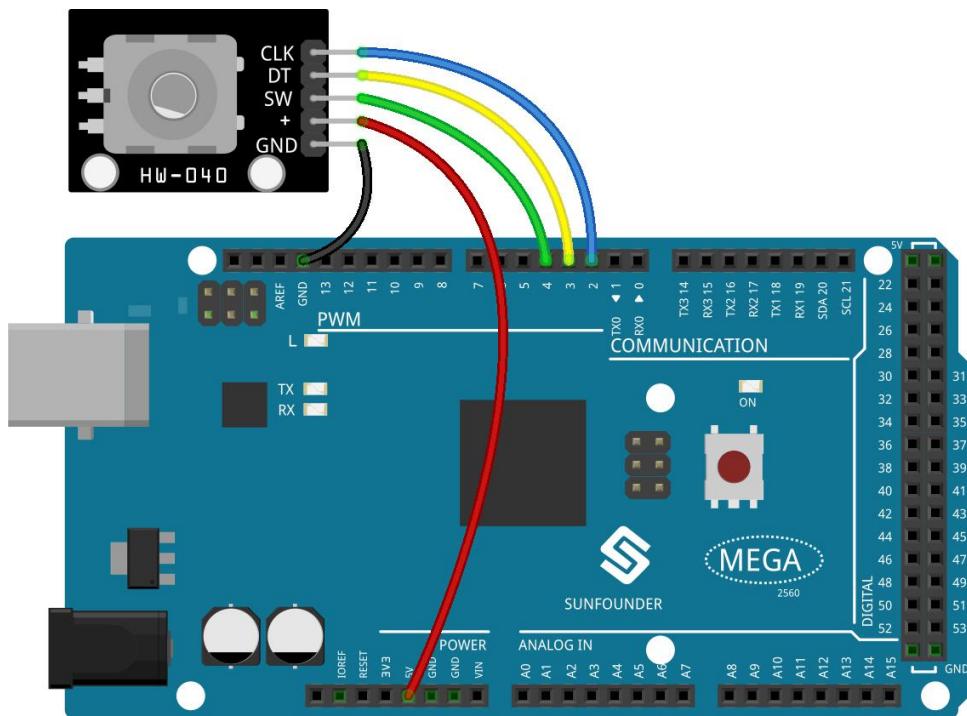
waves are called channel A and channel B as shown below:

As shown on the right, when channel A changes from high level to low level, if channel B is high level, it indicates the rotary encoder spins clockwise (CW); if at that moment channel B is low level, it means spins counterclockwise (CCW). So if we read the value of channel B when channel A is low level, we can know in which direction the rotary encoder rotates.

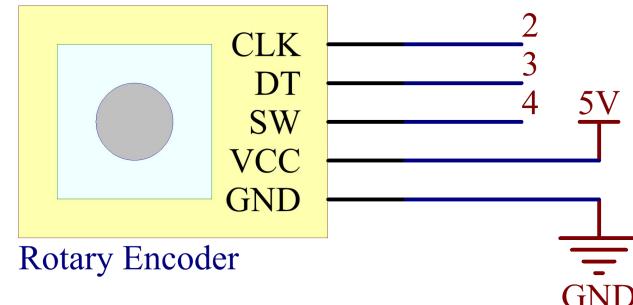


Fritzing Circuit

In this example, we can connect the Rotary Encoder pin directly to the Mega 2560 Board pin, connect the GND of the Rotary Encoder to GND, 「+」 to 5V, SW to digital pin 4, DT to digital pin 3, and CLK to digital pin 2.



Schematic Diagram



Code

```
const int clkPin= 2;//the clk attach to pin2
const int dtPin= 3;//the dt attach to pin3
const int swPin= 4;//the number of the button
int encoderVal = 0;
void setup()
{
    //set clkPin,dtPin,swPin as INPUT
    pinMode(clkPin, INPUT);
    pinMode(dtPin, INPUT);
    pinMode(swPin, INPUT);
    digitalWrite(swPin, HIGH);
    Serial.begin(9600); // initialize serial communications at 9600 bps
}
void loop()
{
    int change = getEncoderTurn();
    encoderVal = encoderVal + change;
    if(digitalRead(swPin) == LOW)//if button pull down
    {
        encoderVal = 0;
    }
}
```

```
}

Serial.println(encoderVal); //print the encoderVal on the serial monitor

}

int getEncoderTurn(void)
{
    static int oldA = HIGH; //set the oldA as HIGH
    static int oldB = HIGH; //set the oldB as HIGH
    int result = 0;
    int newA = digitalRead(dtPin); //read the value of clkPin to newA
    int newB = digitalRead(clkPin); //read the value of dtPin to newB
    if (newA != oldA || newB != oldB) //if the value of clkPin or the dtPin has changed
    {
        // something has changed
        if (oldA == HIGH && newA == LOW)
        {
            result = (oldB * 2 - 1);
        }
    }
    oldA = newA;
    oldB = newB;
    return result;
}
```

You will see the angular displacement of the rotary encoder printed on Serial Monitor. When you turn the rotary encoder clockwise, the angular displacement is increased; when turn it counterclockwise, the displacement is decreased. If you press the switch on the rotary encoder, the readings will return to zero.

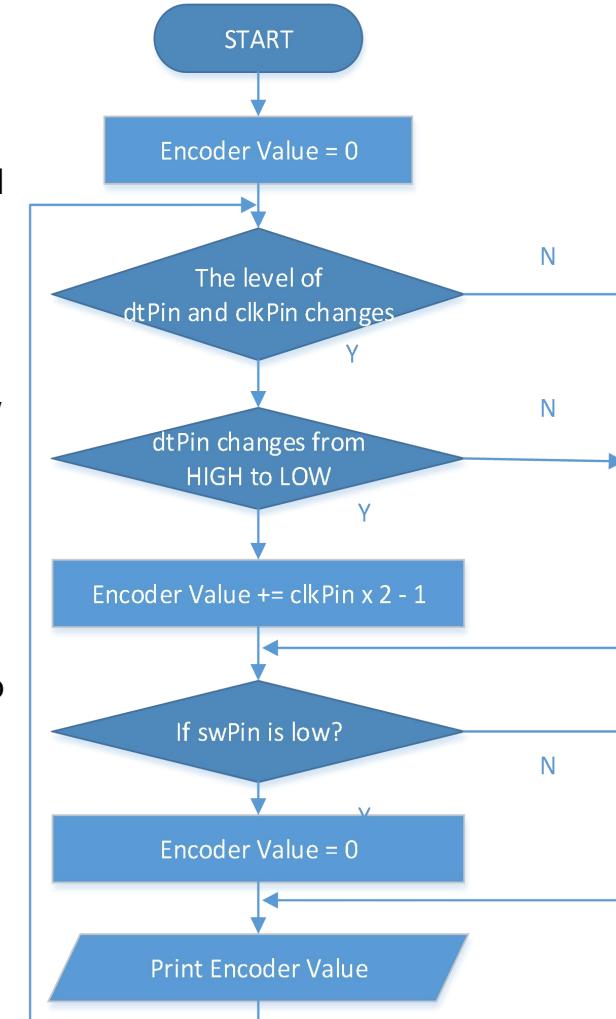
Code Analysis

When Rotary Encoder is used, the following situations of pin level will occur.

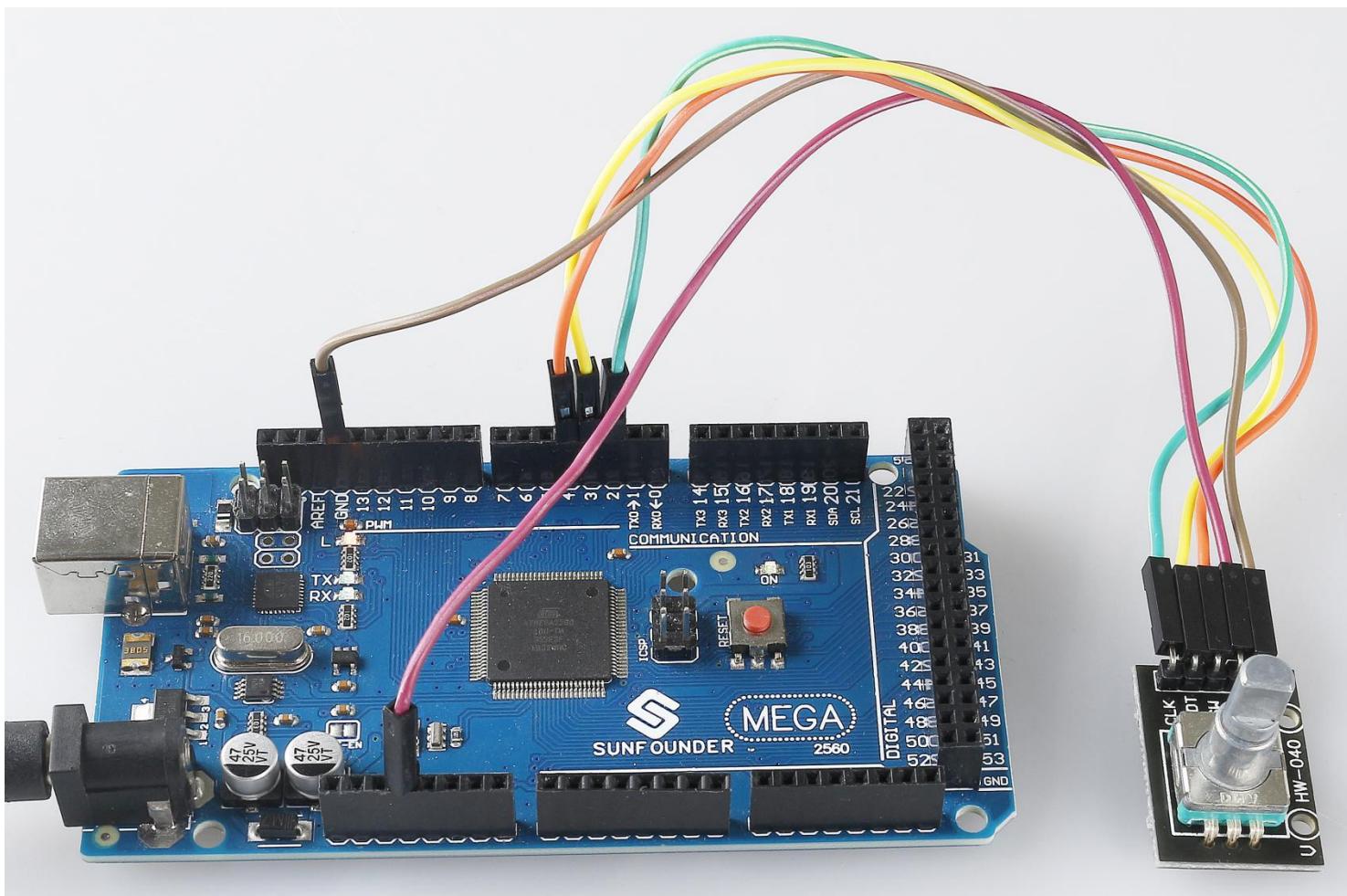
- ① When rotating the shaft, dtPin will go from high level to low level.
- ② clkPin will remain high level when the shaft rotates clockwise, otherwise it goes low level.
- ③ When the shaft is pressed, swPin will have low level.

From this, the program flow is shown on the right.

For detailed analysis of potential state change judgment, please refer to [Part 1-1.10 State Change Detection](#).



Phenomenon Picture

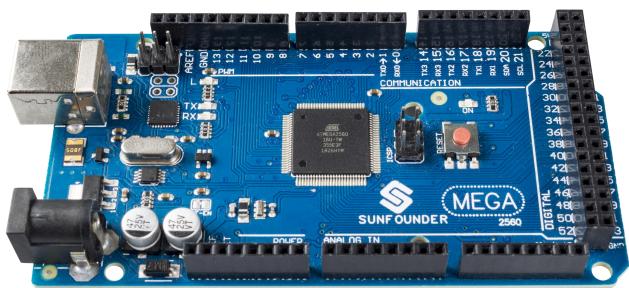
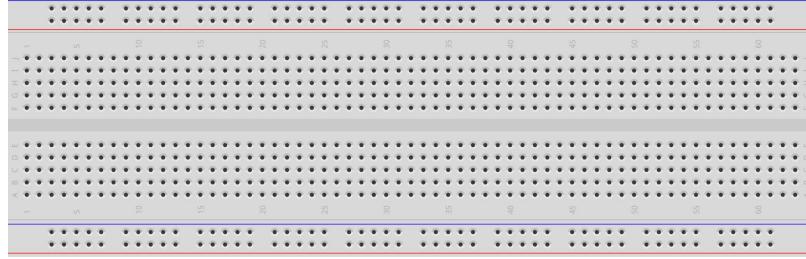


2.26 Photoresistor

Overview

In this lesson, you will learn about Photoresistor. Photoresistor is applied in many electronic goods, such as the camera meter, clock radio, alarm device (as beam detector), small night lights, outdoor clock, solar street lamps and etc. Photoresistor is placed in a street lamp to control when the light is turned on. Ambient light falling on the photoresistor causes street lamps to turn on or off.

Components Required

1 * Photoresistor	1 * 10K ohm resistor	Several Jumper Wires
 A photoresistor component, which is a small cylindrical device with two wires extending from it.	 A 10K ohm resistor component, which is a cylindrical resistor with four color bands on its body.	 A pair of jumper wires, one red and one black, used for connecting components.
1 * Mega 2560 Board	1 * Breadboard	
 A blue Arduino Mega 2560 microcontroller board with various pins, chips, and connectors.	 A breadboard with a grid of 40 columns and 24 rows of holes for connecting components.	



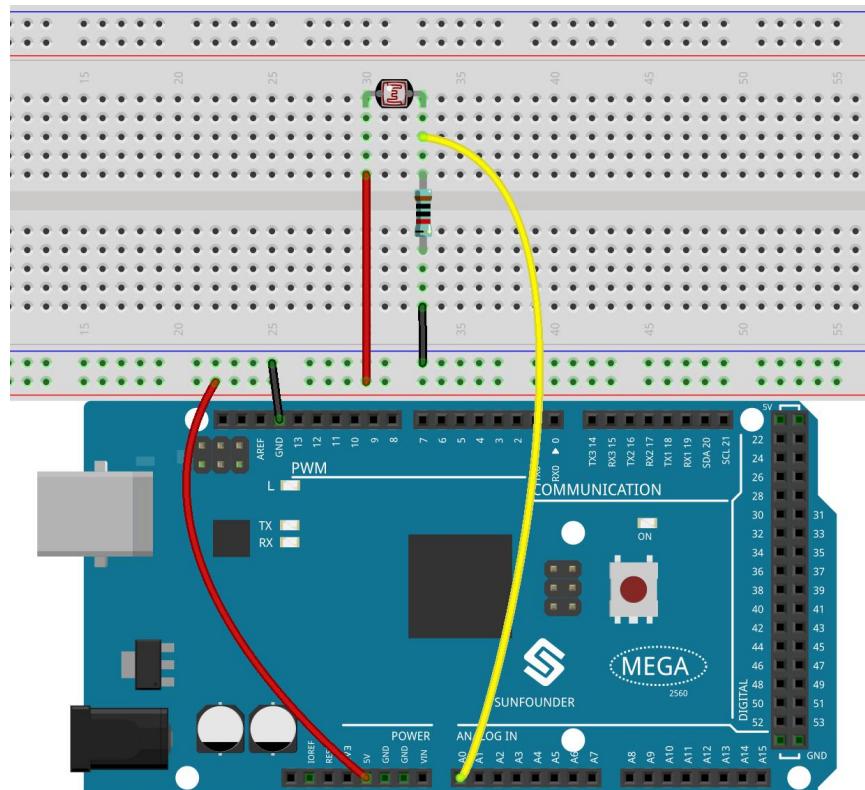
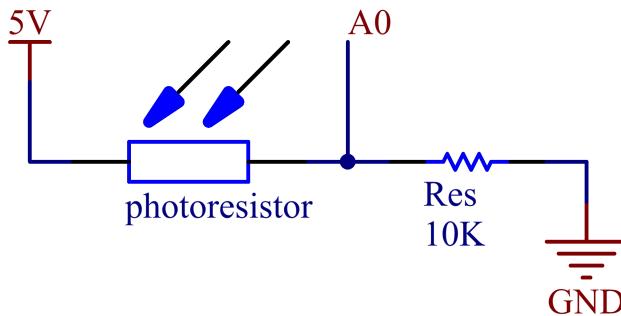
Component Introduction

A photoresistor or photocell is a light-controlled variable resistor. The resistance of a photoresistor decreases with increasing incident light intensity; in other words, it exhibits photo conductivity. A photoresistor can be applied in light-sensitive detector circuits, and light- and darkness-activated switching circuits.

Fritzing Circuit

In this example, we use analog pin (A0) to read the value of photoresistor. One pin of photoresistor is connected to 5V, the other is wired up to A0. Besides, a $10\text{k}\Omega$ resistor is needed before the other pin is connected to GND.

Schematic Diagram



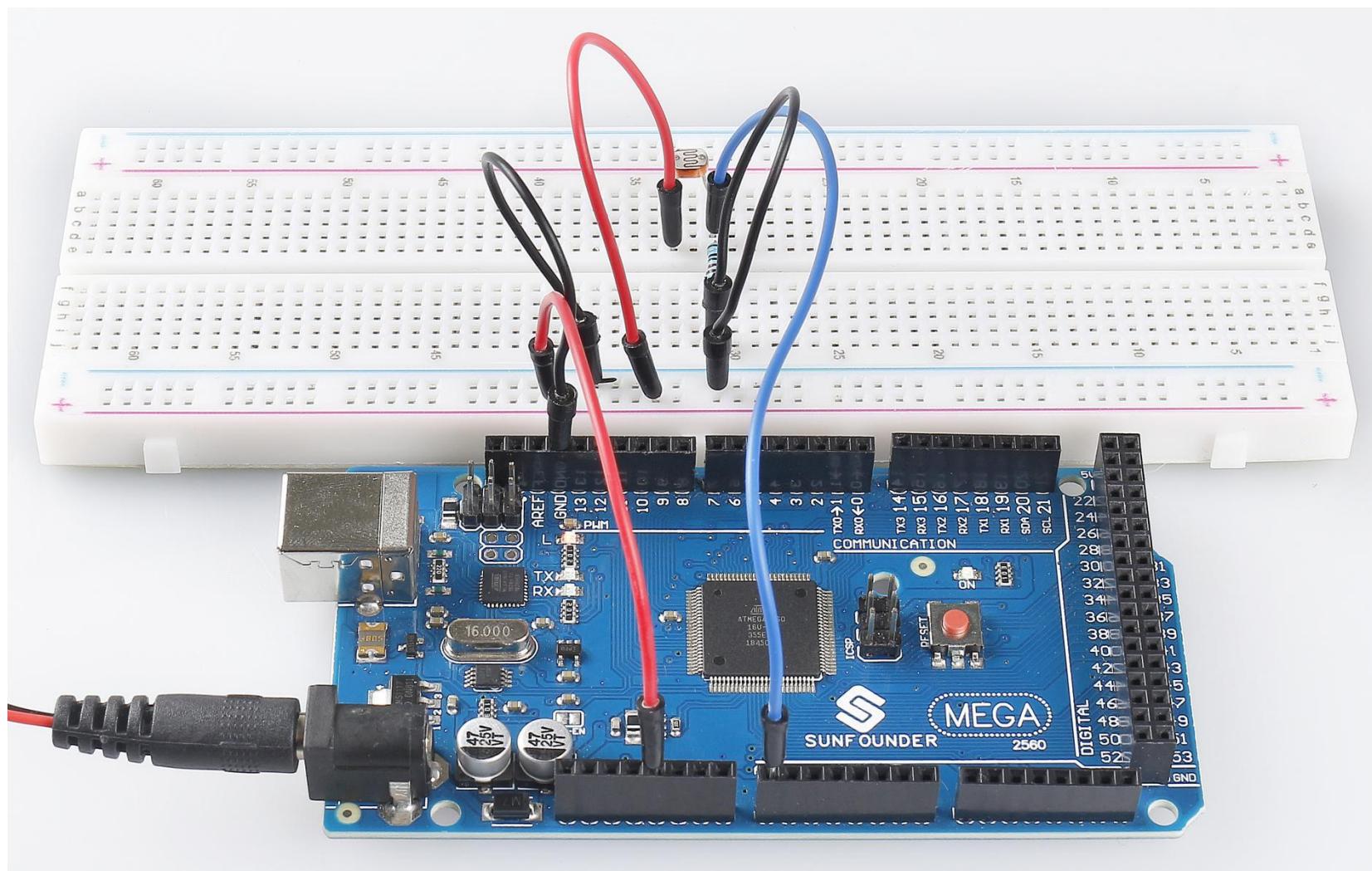
Code

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

void loop() {
    int sensorValue = analogRead(A0);
    Serial.println(sensorValue);
    delay(1);
}
```

After uploading the codes to the Mega2560 board, you can open the serial monitor to see the read value of the pin. When the ambient light becomes stronger, the reading will increase correspondingly, and the pin reading range is [0] ~ [1023]. However, according to the environmental conditions and the characteristics of the photoresistor, the actual reading range may be smaller than the theoretical range. For a detailed explanation of the code, refer to [Part 1-1.5 Analog Read](#).

Phenomenon Picture

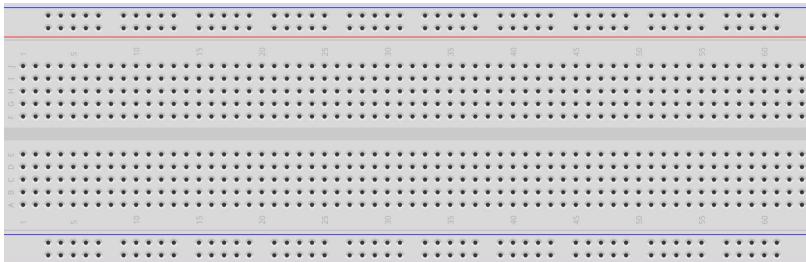


2.27 Thermistor

Overview

In this lesson, you will learn how to use thermistor. Thermistor can be used as electronic circuit components for temperature compensation of instrument circuits. In the current meter, flowmeter, gas analyzer, and other devices. It can also be used for overheating protection, contactless relay, constant temperature, automatic gain control, motor start, time delay, color TV automatic degaussing, fire alarm and temperature compensation.

Components Required

1 * Thermistor	1 * 10K ohm resistor	Several Jumper Wires
		
1 * Mega 2560 Board	1 * Breadboard	
		

Component Introduction

103

Thermistor is a sensitive element, and it has two types: Negative Temperature Coefficient (NTC) and Positive Temperature Coefficient (PTC), also known as NTC and PTC. Its resistance varies significantly with temperature. The resistance of PTC thermistor increases with temperature ,while the condition of NTC is opposite to the former. In this experiment we use NTC.

The principle is that the resistance of the NTC thermistor changes with the temperature of the outer environment. It detects the real-time temperature of the environment. When the temperature gets higher, the resistance of the thermistor decreases. Then the voltage data is converted to digital quantities by the A/D adapter. The temperature in Celsius or Fahrenheit is output via programming.

In this experiment, a thermistor and a 10k pull-up resistor are used. Each thermistor has a normal resistance. Here it is 10k ohm, which is measured under 25 degree Celsius.

Here is the relation between the resistance and temperature:

$$R_T = R_N \exp^{B(1/T_K - 1/T_N)}$$

R_T is the resistance of the NTC thermistor when the temperature is T_K .

R_N is the resistance of the NTC thermistor under the rated temperature T_N . Here, the numerical value of R_N is 10k.

T_K is a Kelvin temperature and the unit is K. Here, the numerical value of T_K is 273.15 + degree Celsius.

T_N is a rated Kelvin temperature; the unit is K too. Here, the numerical value of T_N is 273.15+25.

And **B(beta)**, the material constant of NTC thermistor, is also called heat sensitivity index with a numerical value 3950.

exp is the abbreviation of exponential, and the base number e is a natural number and equals 2.7 approximately.

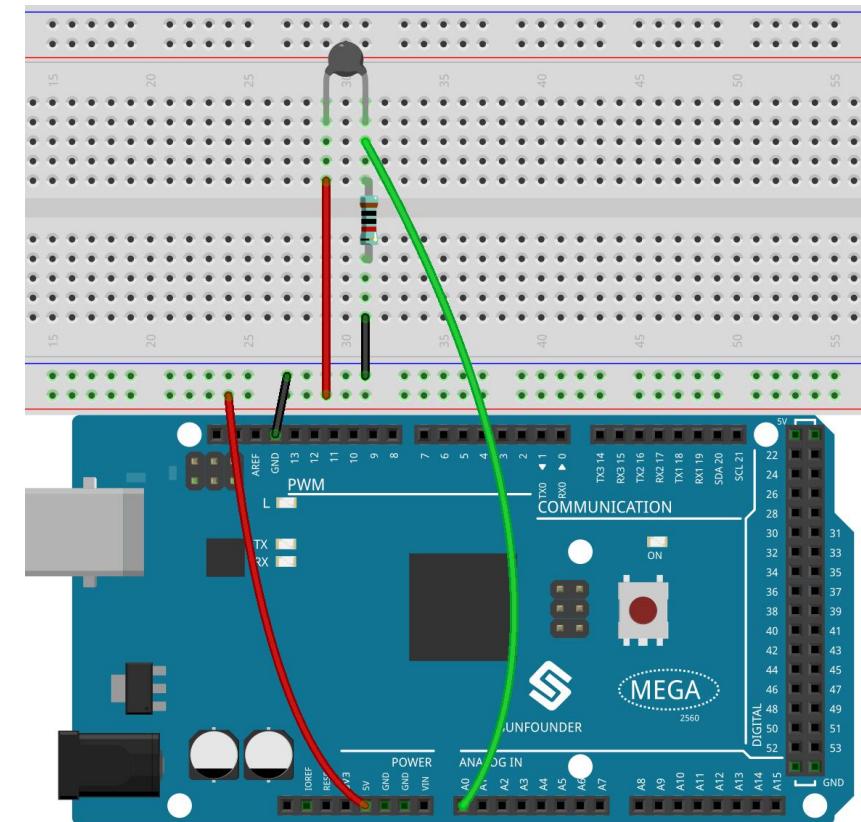
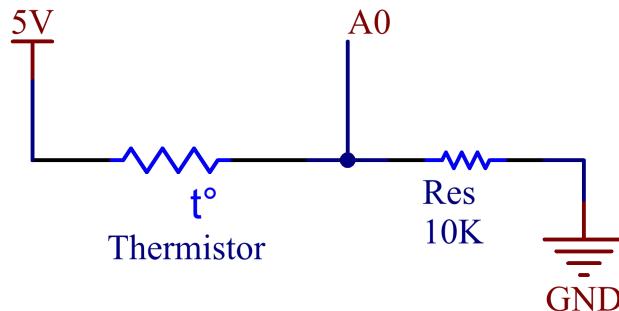
Convert this formula $T_k = 1 / (\ln(R_T/R_N)/B + 1/T_N)$ to get Kelvin temperature that minus 273.15 equals degree Celsius.

This relation is an empirical formula. It is accurate only when the temperature and resistance are within the effective range.

Fritzing Circuit

In this example, we use the analog pin A0 to get the value of Thermistor. One pin of thermistor is connected to 5V, and the other is wired up to A0. At the same time, a 10k Ω resistor is connected with the other pin before connecting to GND.

Schematic Diagram



Code

```
#define analogPin A0 //the thermistor attach to
#define beta 3950 //the beta of the thermistor
#define resistance 10 //the value of the pull-up resistor

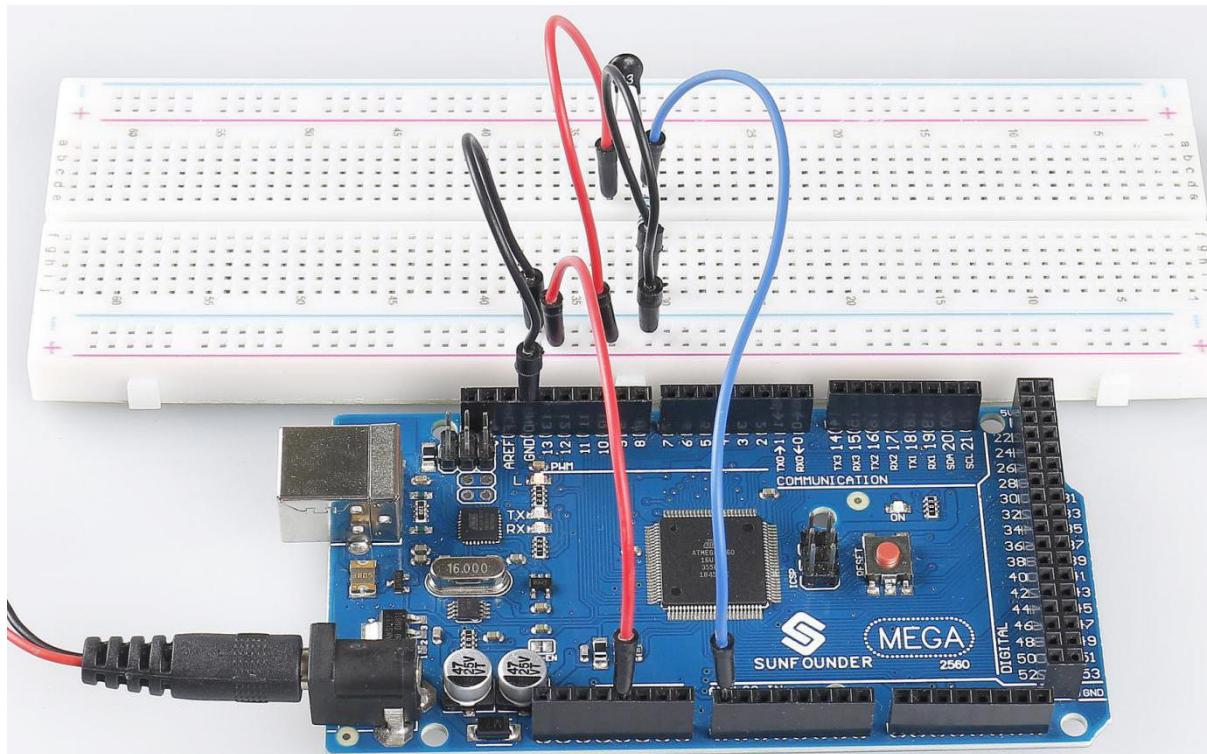
void setup()
{
    Serial.begin(9600);
}

void loop()
{
    //read thermistor value
    long a = analogRead(analogPin);
    //the calculating formula of temperature
    float tempC = beta / (log((1025.0 * 10 / a - 10) / 10) + beta / 298.0) - 273.0;
    float tempF = 1.8 * tempC + 32.0;
    Serial.print("Temp: ");
    Serial.print(tempC);
    Serial.println("degree Celsius");
    Serial.print("Temp: ");
    Serial.print(tempF);
```

```
Serial.println("degree Fahrenheit");
delay(200); //wait for 200 milliseconds
}
```

After uploading the code to the Mega2560 board, you can open the serial monitor to check the current temperature. The Kelvin temperature is calculated according to the formula $T_K=1/(\ln(R_T/R_N)/B+1/T_N)$.

Phenomenon Picture



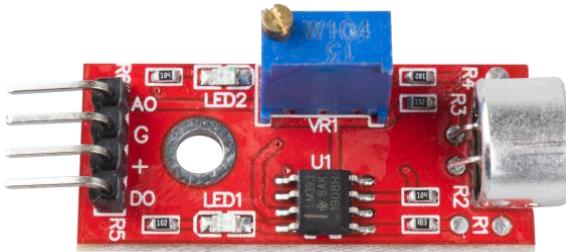
2.28 Sound Sensor Module

Overview

In this lesson, you will learn how to use a sound sensor module. The sound sensor module provides an easy way to detect sound and is generally used for detecting sound intensity.

Components Required

1 * Sound Sensor Module



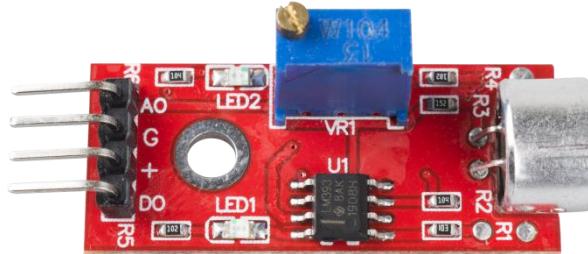
1 * Mega 2560 Board



Several Jumper Wires



Component Introduction



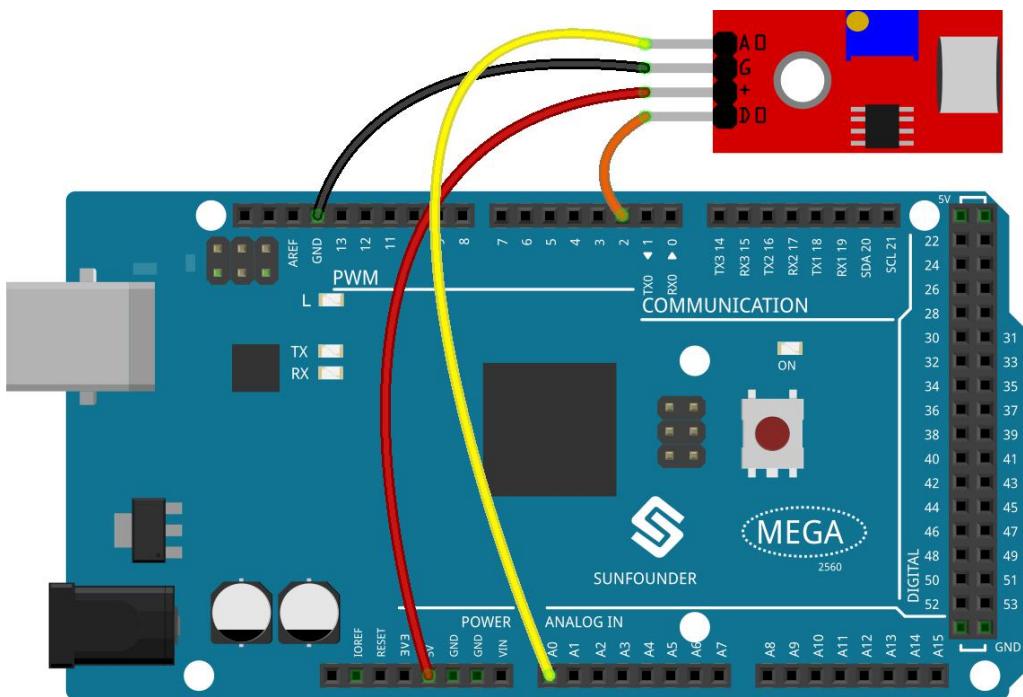
This module can be used for security, switch, and monitoring applications. Its accuracy can be easily adjusted for the convenience of usage.

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 micro-controller then performs necessary processing.

This module has two outputs:

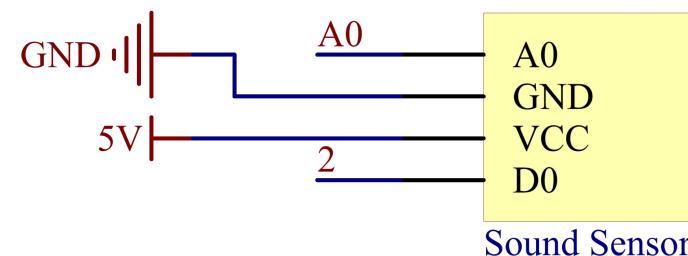
- ① AO: analog output, real-time output voltage signal of microphone.
- ② DO: when the intensity of the sound reaches a certain threshold, the output is a high or low level signal. The threshold sensitivity can be achieved by adjusting the potentiometer.

Fritzing Circuit



In this example, we can directly connect the pin of Sound Sensor Module to the pin of Mega 2560 Board, connect the pin 「G」 of Sound Sensor Module to GND, the pin 「+」 to 5V, AO to analog pin A0, and D0 to digital pin 2.

Schematic Diagram



Code

```
void setup() {  
    pinMode(2,INPUT);  
    Serial.begin(9600);  
}  
void loop() {  
    int sensorDigitalValue= digitalRead(2);  
    int sensorAnalogValue = analogRead(A0);  
    Serial.print("Digital Reading: ");  
    Serial.println(sensorDigitalValue);  
    Serial.print("Analog Reading: ");  
    Serial.println(sensorAnalogValue);  
    Serial.println("");  
    delay(1);  
}
```

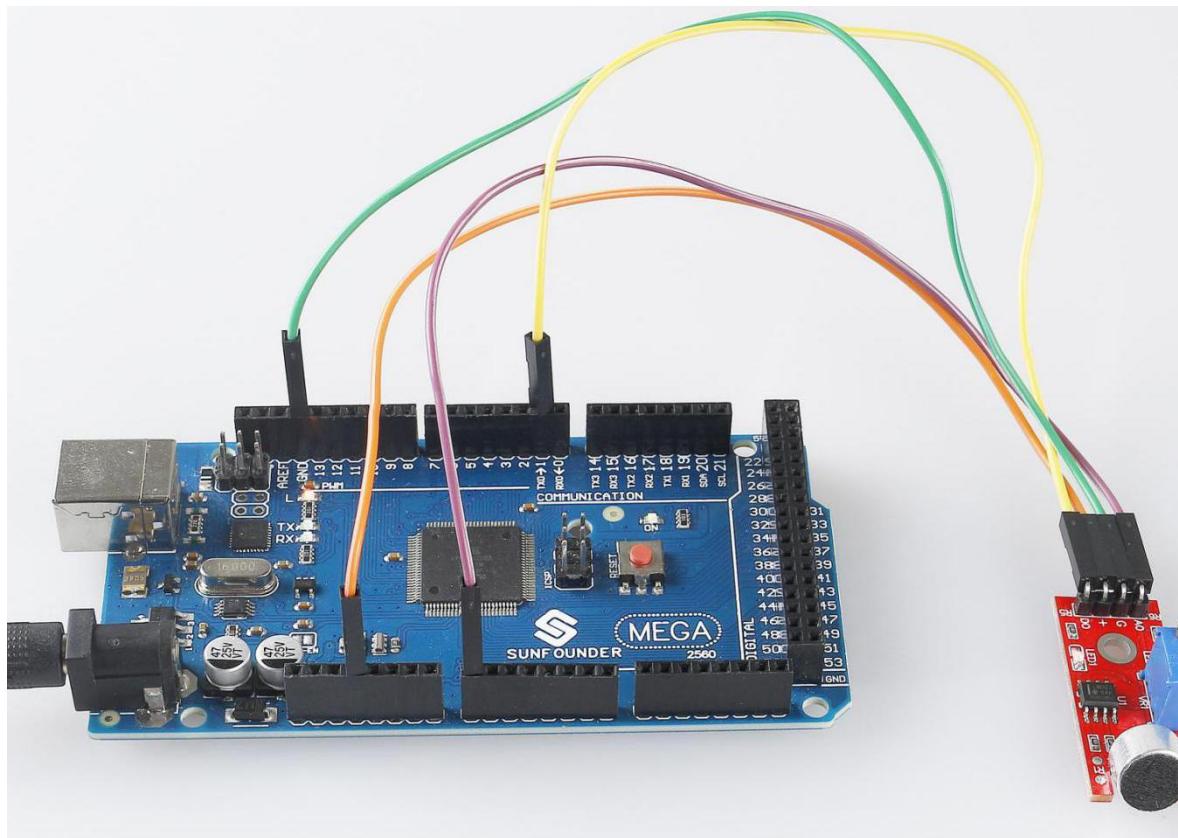
After uploading the code to the Mega2560 board, you can open the serial monitor to see the read value of the pin. When the ambient sound gets louder, the digital reading is 「1」 (adjust the potentiometer of the module to modify the threshold to trigger the high level), and the reading value of the analog pin will change significantly; when the environment is quiet, the digital reading is 「0」 and the analog reading changes smoothly.

The range of analog reading is 「0」 ~ 「1023」, but influenced by the environmental condition and the characteristics of

sound sensor, the actual reading range may be smaller than the theoretical one. If an oscilloscope is used, the changing of analog reading of the sound sensor will be more obvious.

About the detail code explanation, refer to [Part 1-1.5 Analog Read](#) and [Part 1-1.4 Digital Read](#).

Phenomenon Picture



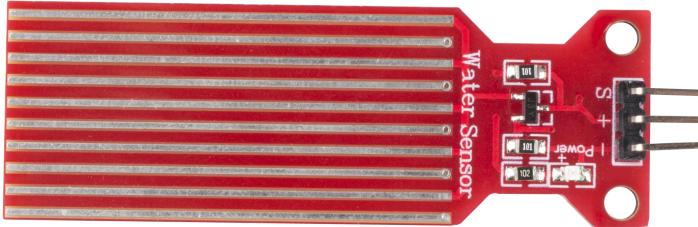
2.29 Water Sensor Module

Overview

In this lesson, you will learn how to use a water sensor module. A water sensor module is designed for water detection, which can be widely used in sensing the rainfall, water level, even the liquid leakage.

Components Required

1 * Water Level Detection Sensor



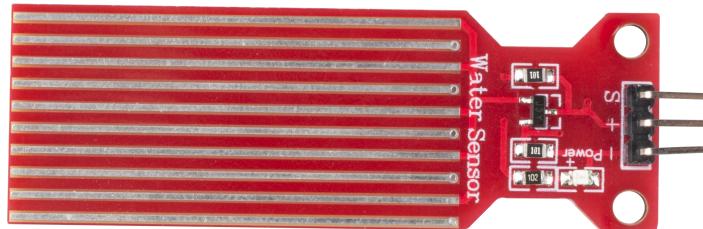
1 * Mega 2560 Board



Several Jumper Wires



Component Introduction



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.

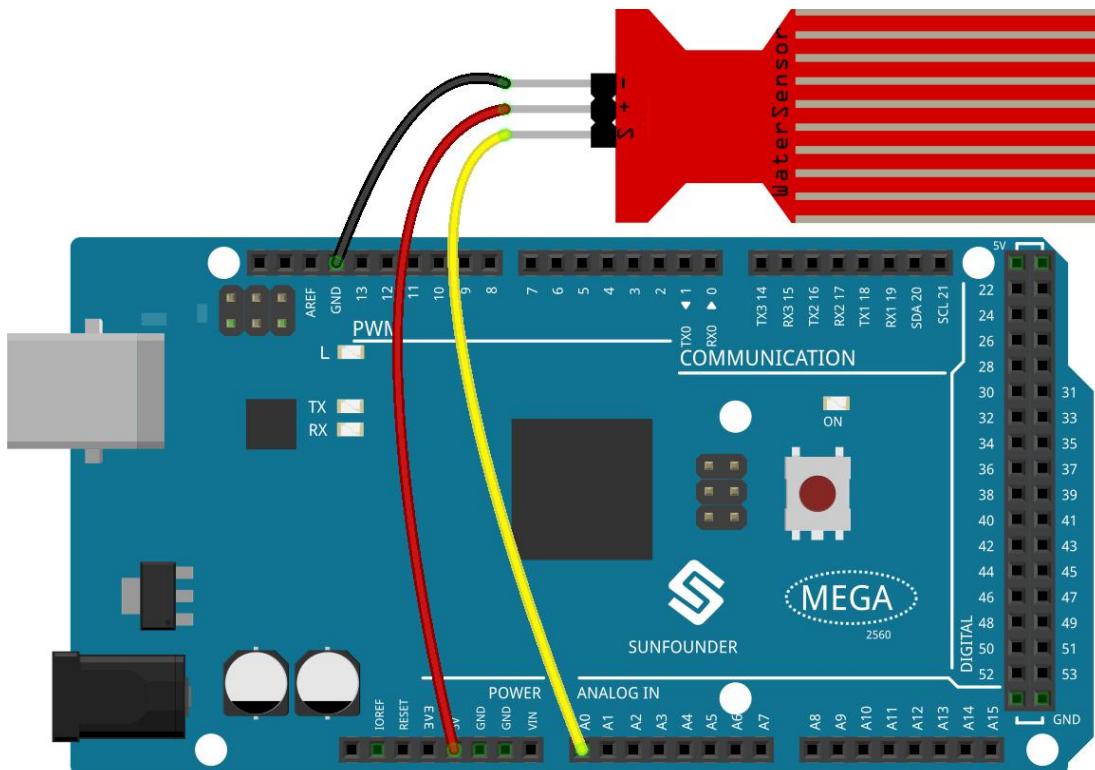
This item can judge the water level through with a series of exposed parallel wires stitch to measure the water droplet/water size.

The core part of water level sensor is an amplification circuit composed of a transistor and several comb-shape PCB cables. When placed in water, the comb-shape cable will change its resistance with the depth of the water and convert the depth signal into an electrical signal, and output analog value can directly be used in the program function, then to achieve the function of water level alarm.

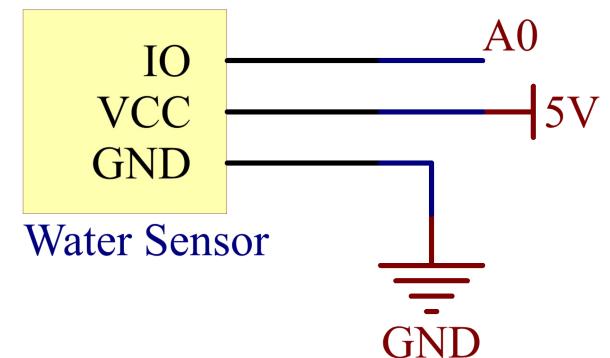
Fritzing Circuit

In this example, we directly connect the pins of Water Sensor Module to pins of Mega 2560 Board. We use analog A0 to

get the value of Water Sensor Module, and get the pin 「S」 of Water Sensor Module to A0, 「-」 to GND, 「+」 to 5V.



Schematic Diagram

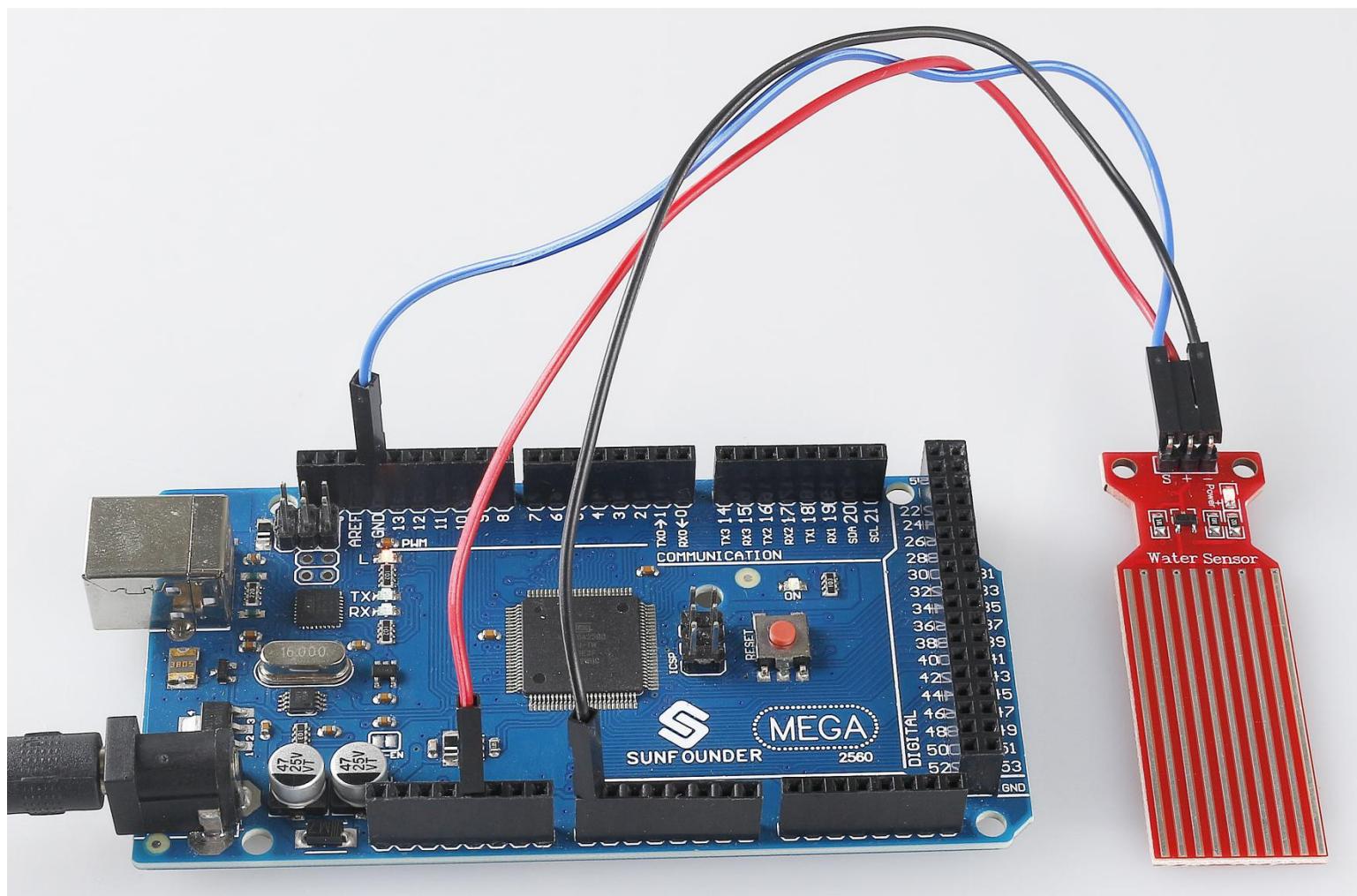


Code

```
void setup() {  
    Serial.begin(9600);  
}  
  
void loop() {  
    int sensorValue = analogRead(A0);  
    Serial.println(sensorValue);  
    delay(1);  
}
```

After uploading the code to the Mega2560 board, you can open the serial monitor to see the read value of the pin. As the water level rises, the readings increase. Readings vary within the range [0] ~ [1023], but influenced by the environmental condition and the characteristics of water level sensor, the actual reading range may be smaller than the theoretical range. Refer to Part 1-1.5 Analog Read to check the detail code explanation.

Phenomenon Picture



2.30 IR Obstacle Avoidance Sensor

Overview

In this lesson, you will learn how to use IR Obstacle Avoidance Sensor. This module is commonly installed on the car and robot to judge the existence of the obstacles ahead. Also it is widely used in hand held device, water faucet and so on.

Components Required

1 * IR Obstacle Avoidance Sensor Module



Several Jumper Wires



1 * Mega 2560 Board

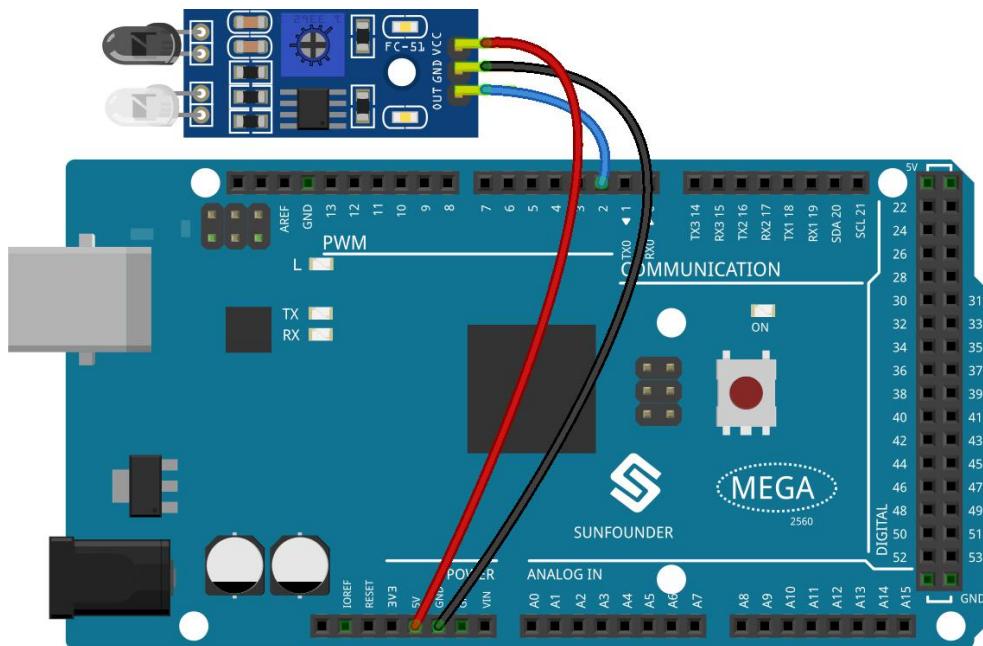


Component Introduction

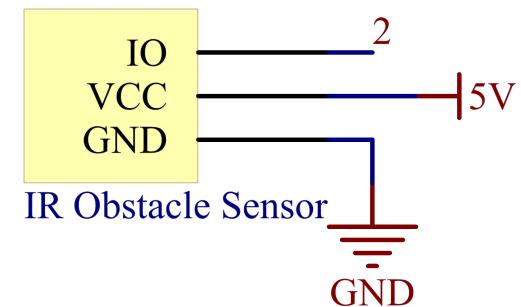
An IR Obstacle Sensor works in accordance with the infrared reflection principle to detect obstacles. When there is no object, the infrared receiver receives no signals; when there is an object ahead which blocks and reflects the infrared light, the infrared receiver will receive signals.

Fritzing Circuit

We can directly connect the pins of IR Obstacle Sensor Module with the pins of Mega 2560 Board. The digital pin 2 is used to read the signal of IR Obstacle Avoidance Sensor Module. We get the VCC of IR Sensor Module connected to 5V, GND to GND, OUT to digital pin 2.



Schematic Diagram

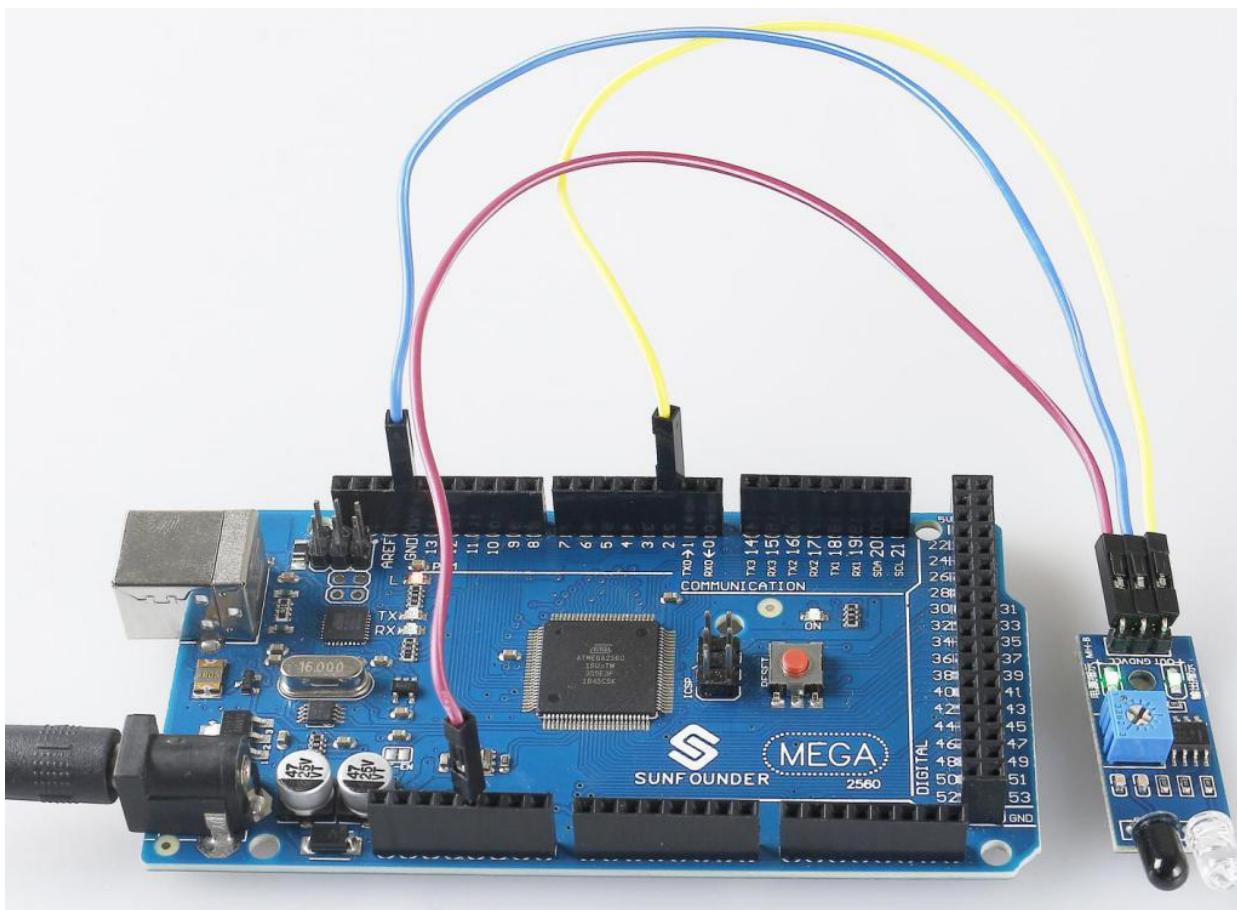


Code

```
void setup() {  
    Serial.begin(9600);  
    pinMode(2, INPUT);  
}  
  
void loop() {  
    Serial.println(digitalRead(2));  
    delay(1);  
}
```

Uploaded the codes to the Mega2560 board, you can see the readings of pins on the serial monitor. When IR Obstacle Avoidance Sensor Module detects there is something covering ahead, there appears 「0」 on the serial monitor; otherwise, 「1」 is displayed. Refer to [Part 1-1.4 Digital Read](#) to check the detail code explanation.

Phenomenon Picture



2.31 PIR Module

Overview

In this lesson, you will learn how to use PIR Module. The PIR sensor detects infrared heat radiation or the presence of organisms that emit infrared heat radiation. This module is widely used in daily life for our intruder alarm and visiting prompt.

Components Required

1 * PIR Module



1 * Mega 2560 Board



Several Jumper Wires





Component Introduction

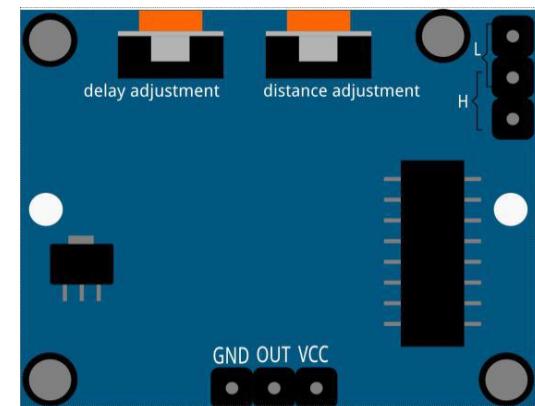
The PIR sensor is split into two slots that are connected to a differential amplifier. Whenever a stationary object is in front of the sensor, the two slots receive the same amount of radiation and the output is zero. Whenever a moving object is in front of the sensor, one of the slots receives more radiation than the other, which makes the output fluctuate high or low. This change in output voltage is a result of detection of motion.

After the sensing module is wired, there is a one-minute initialization. During the initialization, module will output for 0~3 times at intervals. Then the module will be in the standby mode. Please keep the interference of light source and other sources away from the surface of the module so as to avoid the misoperation caused by the interfering signal. Even you'd better use the module without too much wind, because the wind can also interfere with the sensor.

Two trigger modes: (choosing different modes by using the jumper cap).

Distance Adjustment

Turning the knob of the distance adjustment potentiometer clockwise, the range of sensing distance increases, and the maximum sensing distance range is about 0-7 meters. If turn it anticlockwise, the range of sensing distance is reduced, and the minimum sensing distance range is about 0-3 meters.



Delay Adjustment

Rotate the knob of the delay adjustment potentiometer clockwise, you can also see the sensing delay increasing. The maximum of the sensing delay can reach up to 300s. On the contrary, if rotate it anticlockwise, you can shorten the delay with a minimum of 5s.

Two trigger modes: (choosing different modes by using the jumper cap).

- ✧ **H: Repeatable trigger mode**, after sensing the human body, the module outputs high level. During the subsequent delay period, if somebody enters the sensing range, the output will keep being the high level.
- ✧ **L: Non-repeatable trigger mode**, outputs high level when it senses the human body. After the delay, the output will change from high level into low level automatically.

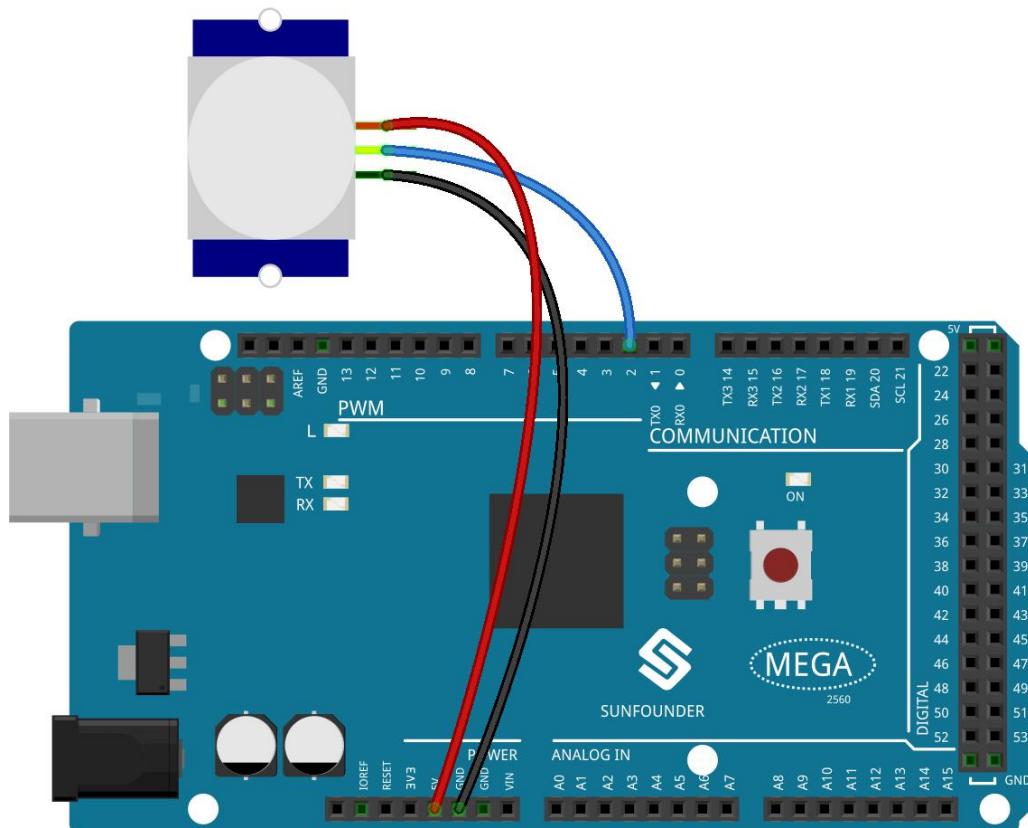
PIR can only be installed indoors, and its false alarm rate has a great relationship with the location and mode of installation.

The correct use should meet the following conditions:

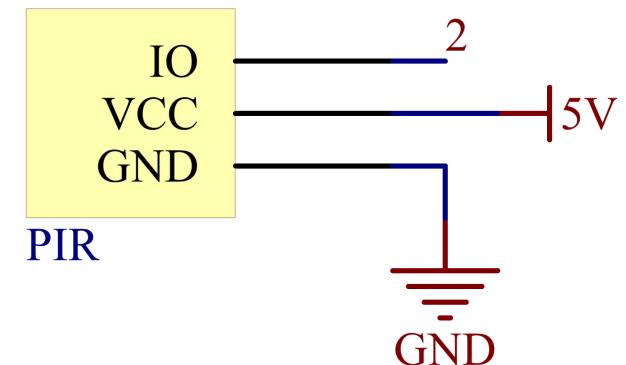
1. It should be 2.0-2.2 meters above the ground.
2. Stay away from air conditioners, refrigerators, stoves and other places where air temperature changes apparently.
3. No screen, furniture, large-scale bonsai or other isolation objects shall be detected within the detection range.
4. Don't face straightly toward the window, otherwise the disturbance of the hot air flow outside the window and the movement of people will cause false alarm.
5. Do not use in areas with strong airflow.
6. The sensitivity of PIR to human body is also closely related to the direction of movement. It is least sensitive to radial movement and most sensitive to the movement in the crosscutting direction.

Fritzing Circuit

In this example, we can connect the pins of Sound Sensor Module to the pins of Mega 2560 Board directly, and we use digital pin 2 to read the signal of PIR Module. Connect the VCC of PIR Module to 5V, GND to GND, and OUT to digital pin 2.
 NOTE: you can remove the PIR cover to see the pin mark.



Schematic Diagram



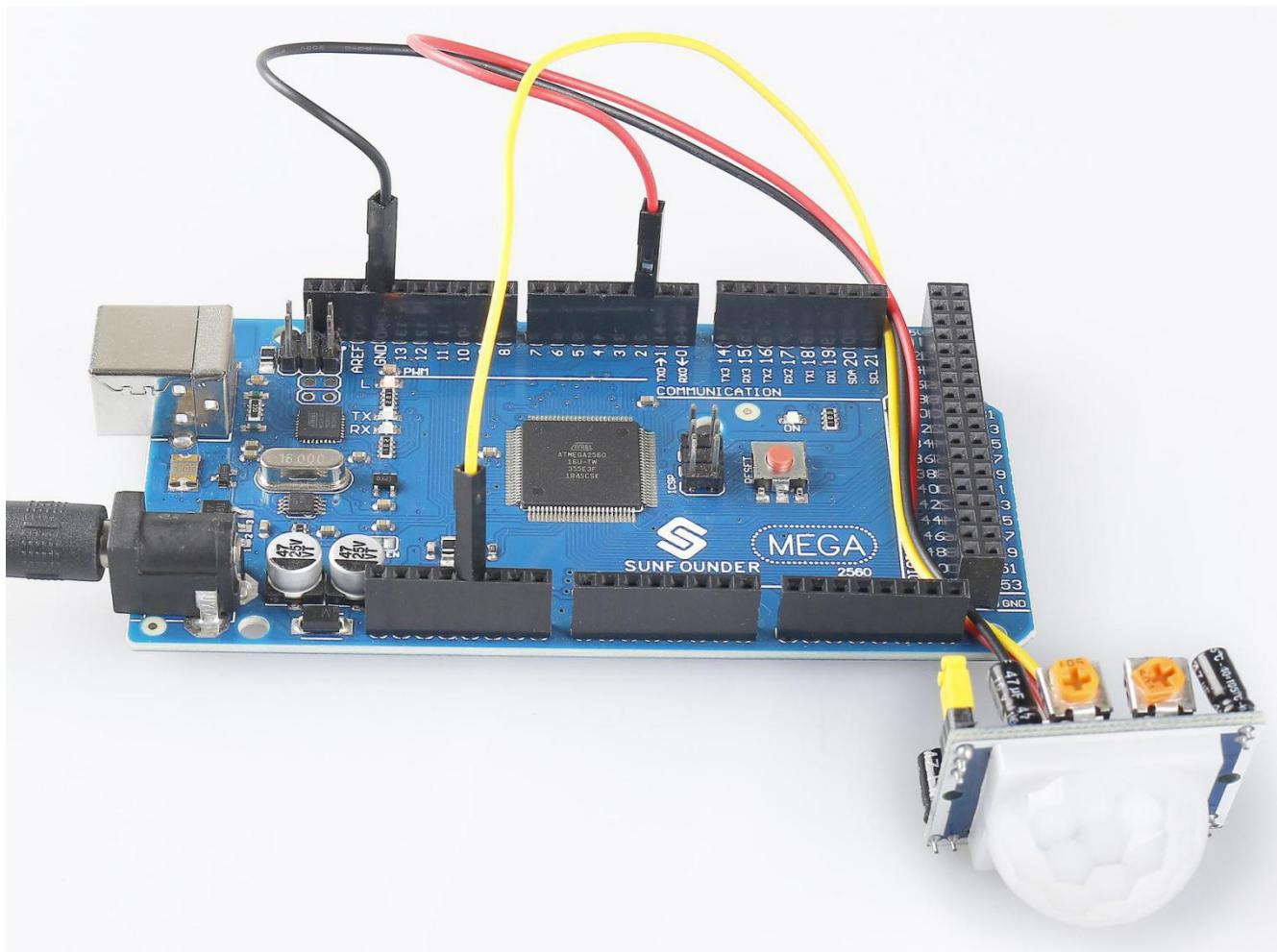
Code

```
void setup() {  
    Serial.begin(9600);  
    pinMode(2, INPUT);  
}  
  
void loop() {  
    Serial.println(digitalRead(2));  
    delay(1);  
}
```

After the codes are uploaded to the Mega2560 board, you can open the serial monitor to see the reading value of the pin. When PIR Module detects activity nearby, the serial monitor will display [1]; otherwise, it will display [0]. Check [Part 1-1.4 Digital Read](#) for more detail code explanation.

There are two potentiometers on the PIR module: one is to adjust **sensitivity** and the other is to adjust the **detection distance**. In order to make the PIR module work better, you need to try to adjust these two potentiometers.

Phenomenon Picture



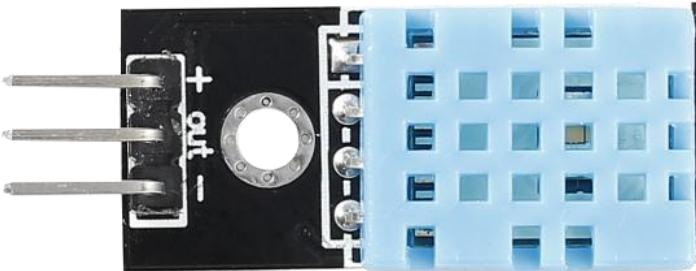
2.32 DHT11 Module

Overview

In this lesson, you will learn how to use DHT11 Module. The DHT11 is a basic, ultra low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins are needed).

Components Required

1 * DHT11 Module



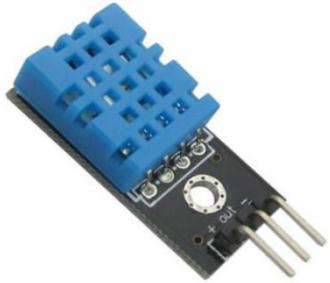
1 * Mega 2560 Board



Several Jumper Wires

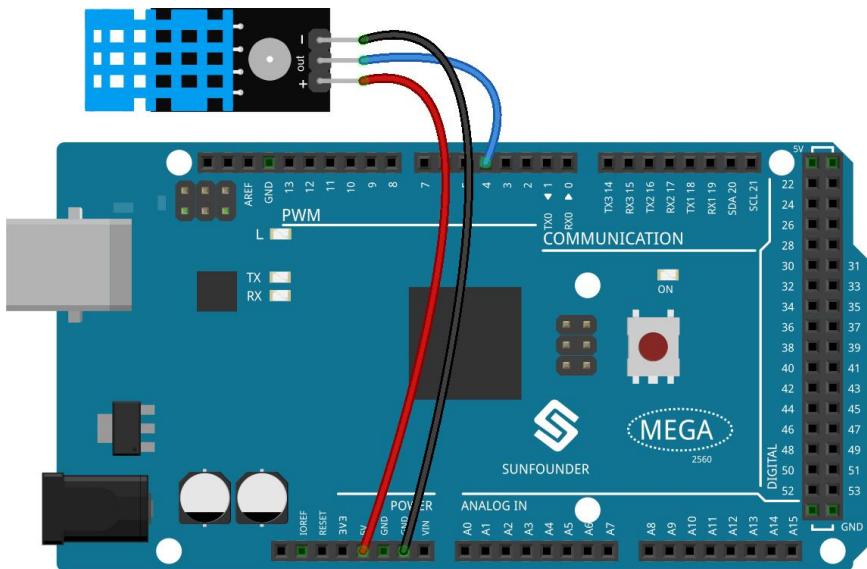


Component Introduction



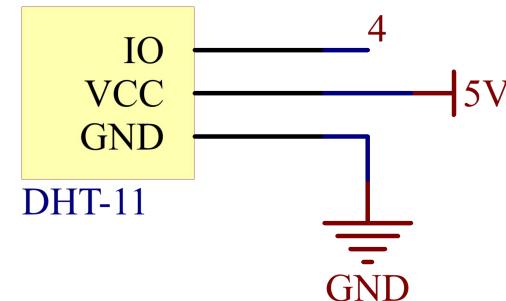
Only three pins are available: VCC, GND, and DATA. The communication process begins with the DATA line sending start signals to DHT11, and DHT11 receives the signals and returns an answer signal. Then the host receives the answer signal and begins to receive 40-bit humiture data (8-bit humidity integer + 8-bit humidity decimal + 8-bit temperature integer + 8-bit temperature decimal + 8-bit checksum). For more information, please refer to DHT11 datasheet.

Fritzing Circuit



In this example, we can directly connect the pins of DHT11 Module to the pins of Mega 2560 Board, and we use pin 4 to read the signal of DHT11 Module. Connect the pin 「+」 of DHT11 Module to 5V, the pin 「-」 to GND, and the pin OUT to pin 4.

Wiring Diagram



Code

The codes use the library dht.h. About how to import library, please refer to [Part 4 - 4.1 Add Libraries](#).

```
#include <dht.h>
dht DHT;

#define DHT11_PIN 4

void setup()
{
    Serial.begin(9600);
    Serial.println("DHT TEST PROGRAM ");
    Serial.print("LIBRARY VERSION: ");
    Serial.println(DHT_LIB_VERSION);
    Serial.println();
    Serial.println("Type,\tstatus,\tHumidity (%),\tTemperature (C)");
}

void loop()
{
    Serial.print("DHT11, \t");
}
```

```
int chk = DHT.read11(DHT11_PIN);
switch (chk)
{
    case DHTLIB_OK:
        Serial.print("OK\t");
        Serial.print(DHT.humidity,1);
        Serial.print("\t");
        Serial.println(DHT.temperature,1);
        delay(1000);
        break;
    case DHTLIB_ERROR_CHECKSUM:
        Serial.println("Checksum error\t");
        break;
    case DHTLIB_ERROR_TIMEOUT:
        Serial.println("Time out error\t");
        break;
    default:
        Serial.println("Unknown error\t");
        break;
}
```

After the codes are uploaded to the Mega2560 board, the serial monitor will continue to output the current temperature and humidity values of the environment.

Code Analysis

The function of the module is included in the library dht.h.

```
#include <dht.h>
```

Library Functions:

dht

Creates a new instance of the dht class that represents a particular DHT-11 module attached to your Arduino board.

```
int read11(uint8_t pin)
```

This function will return CHECK values.

DHTLIB_OK means that DHT-11 is in good condition;

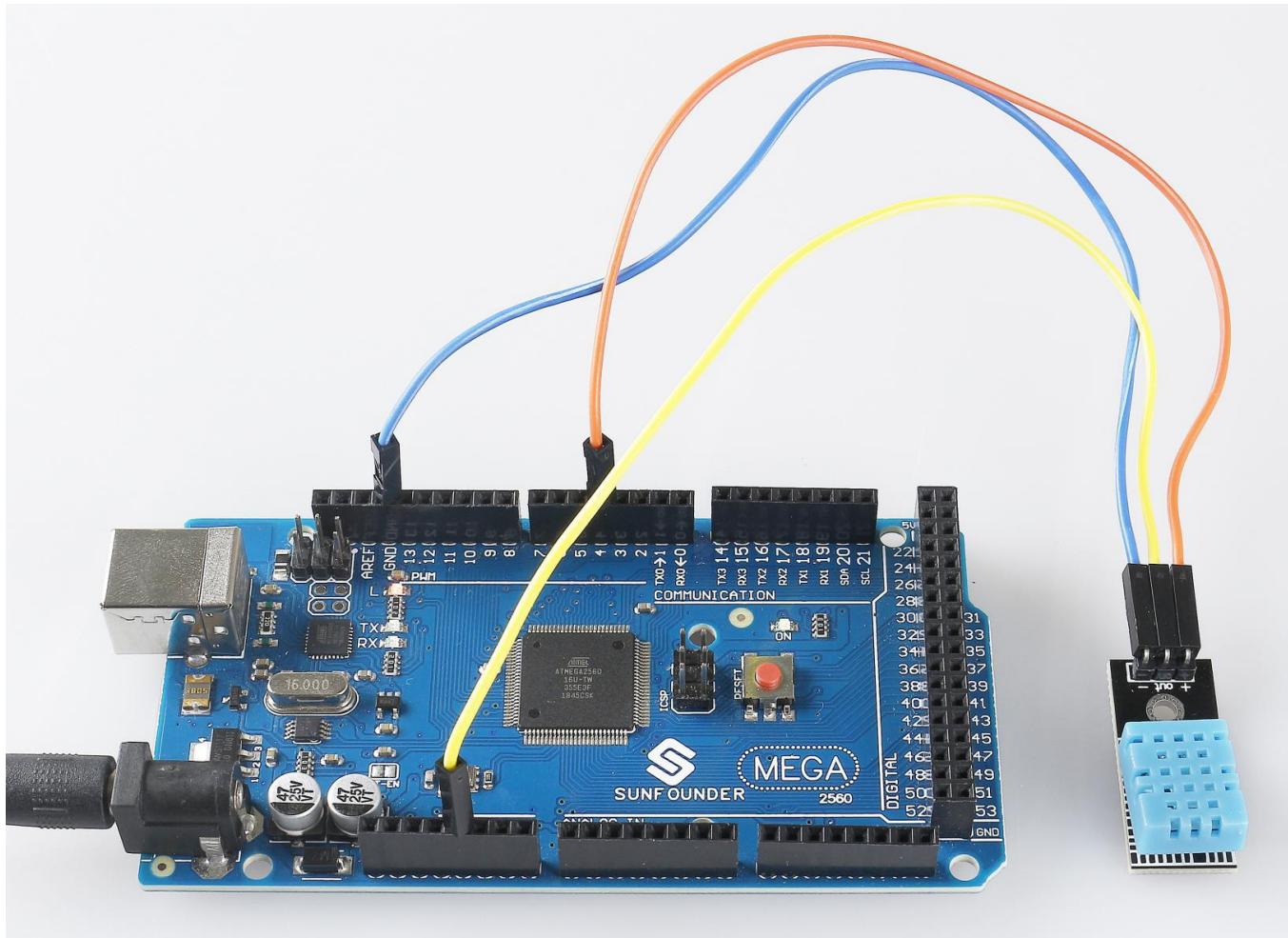
DHTLIB_ERROR_CHECKSUM represents that the value may be abnormal;

DHTLIB_ERROR_TIMEOUT indicates that there is timeout.

The function will store the detected humidity and temperature into the variables with the same name in dht class.

The function should be called and used directly in the main program. (e.g. `Serial.println(DHT.temperature,1);`)

Phenomenon Picture



2.33 Ultrasonic Module

Overview

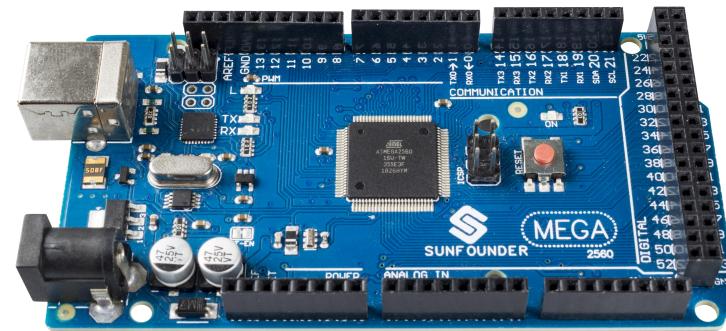
In this lesson, you will learn how to use Ultrasonic module.

Components Required

1 * Ultrasonic Module



1 * Mega 2560 Board



Several Jumper Wires



Component Introduction



Ultrasonic ranging module provides 2cm - 400cm non-contact measurement function. The working principle is that when the MPU sends out orders, the ceramic chip begins to vibrate and then aluminum enclosure vibrates together with it to form ultrasonic wave and emit the wave towards the back of the car. After the emitted ultrasonic wave hits an obstacle, there will be echo wave. This echo wave is received by the same aluminum enclosure and the ceramic chip in the form of vibration. The MPU judges the position of the obstacle by calculating the time difference and azimuth difference of the echo wave. Ultrasonic distance sensor can be widely used in the field of material level (liquid level) monitoring, robot anti-collision, various ultrasonic proximity switches, and intruder alarm.

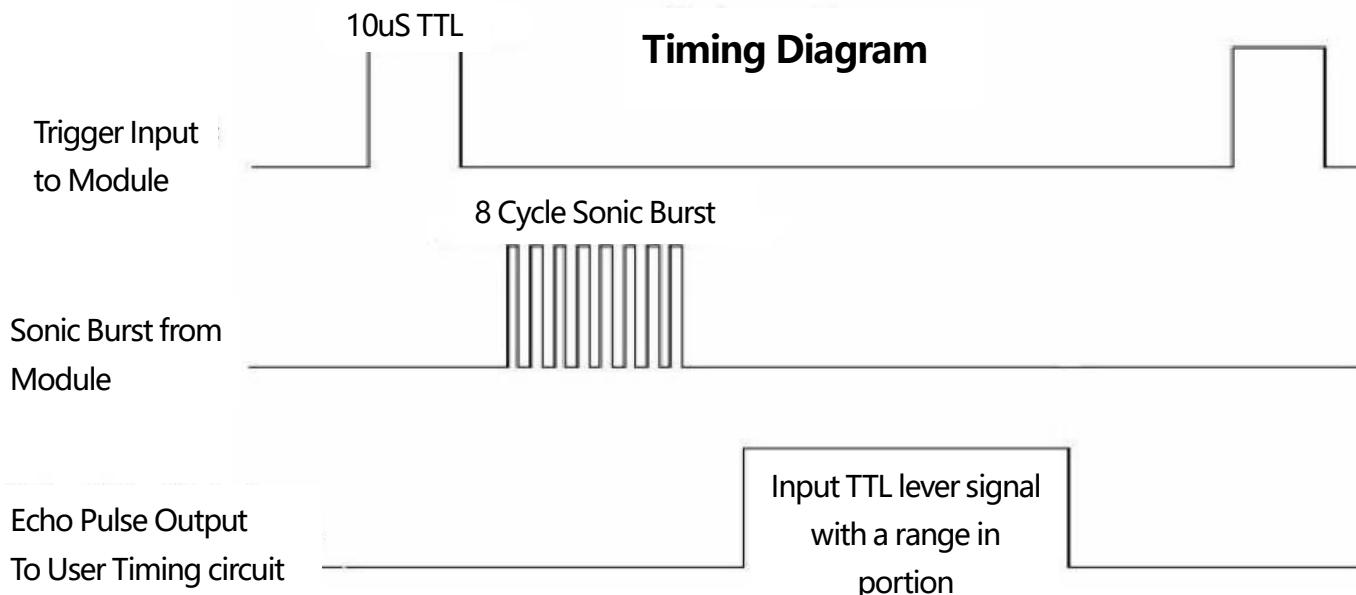
The module includes ultrasonic transmitters, receiver and control circuit. The basic principles are as follows:

- (1) Use an IO flip-flop to process a high level signal of at least 10us;
- (2) The module automatically sends eight 40khz and detects if there is a pulse signal return.
- (3) If the signal returns, passing the high level, the high output IO duration is the time from the transmission of the ultrasonic wave to the return of it. Here, test distance = $(\text{high time} \times \text{sound speed (340 m / s)}) / 2$.

TRIG	Trigger Pulse Input	GND	Ground
ECHO	Echo Pulse Output	VCC	Supply

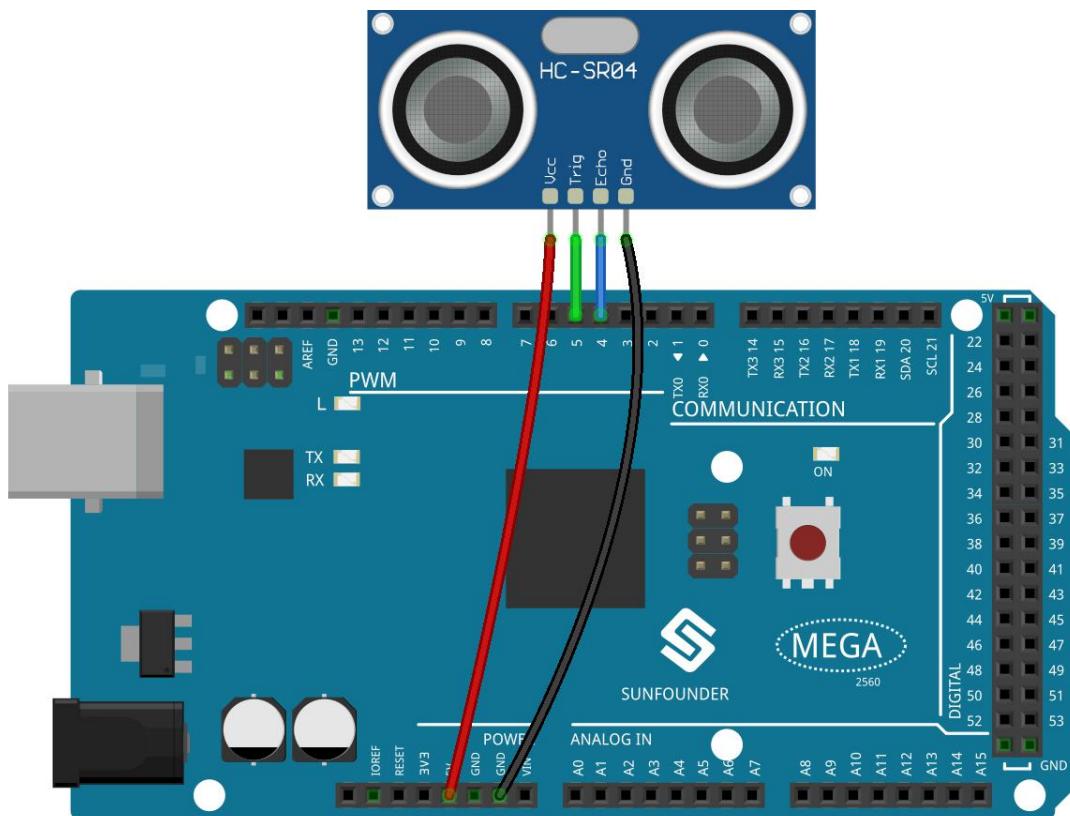
The timing diagram is shown below. You only need to supply a short 10us pulse for the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. You can calculate the range through the time interval between sending trigger signal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

Formula: $us / 58 = \text{centimeters}$ or $us / 148 = \text{inch}$; or: the range = high level time * velocity (340M/S) / 2; you are suggested to use measurement cycle over 60ms in order to prevent signal collisions of trigger signal and the echo signal.



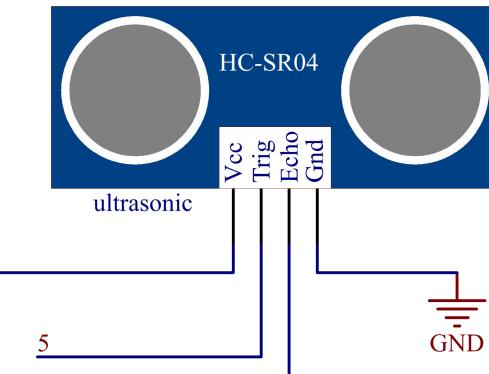
Fritzing Circuit

In this example, we directly connect the pins of Ultrasonic Module with the pins of Mega 2560 Board. And then we get



VCC of the Ultrasonic Module connected to 5V, GND to GND, Trig to the digital pin 5, Echo to the digital pin 4.

Schematic Diagram



Code

```
const int echoPin = 4;
const int trigPin = 5;
void setup(){
    Serial.begin(9600);
    pinMode(echoPin, INPUT);
    pinMode(trigPin, OUTPUT);
    Serial.println("Ultrasonic sensor:");
}

void loop(){
    float distance = readSensorData();
    Serial.print(distance);
    Serial.println(" cm");
    delay(400);
}
float readSensorData(){
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
```

```
digitalWrite(trigPin, LOW);
float distance = pulseIn(echoPin, HIGH)/58.0;
return distance;
}
```

After uploading the codes to the Mega2560 board, the serial monitor will display the distance of obstacles ahead that the ultrasonic sensor has detected.

Code Analysis

About the application of ultrasonic sensor, we can directly check the subfunction.

```
float readSensorData(){// ...}
```

PING is triggered by a HIGH pulse of 2 or more microseconds. (Give a short LOW pulse beforehand to ensure a clean HIGH pulse.)

```
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
```

The echo pin is used to read signal from PING, a HIGH pulse whose duration is the time (in microseconds) from the sending of the ping to the reception of echo of the object.

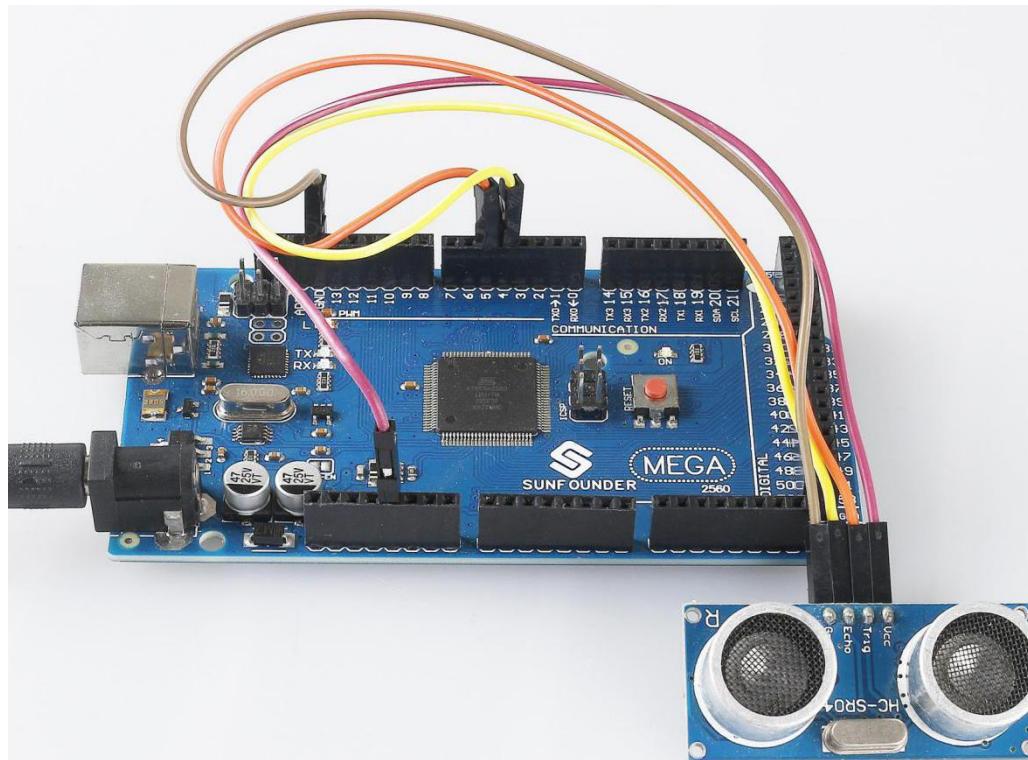
```
microsecond=pulseIn(echoPin, HIGH);
```

The speed of sound is 340 m/s or 29 microseconds per centimeter.

This gives the distance travelled by the ping, outbound and return, so we divide by 2 to get the distance of the obstacle.

```
float distance = microsecond / 29.0 / 2;
```

Phenomenon Picture



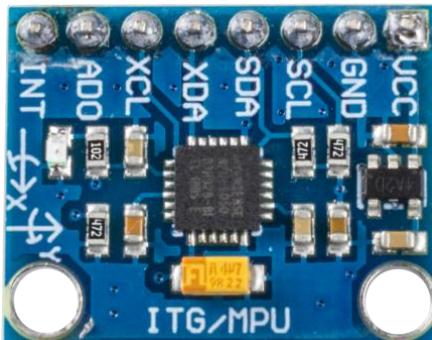
2.34 MPU6050 Module

Overview

In this lesson, you will learn how to use MPU6050. MPU-6050 is a 6-axis (combined 3-axis Gyroscope, 3-axis Accelerometer) motion tracking devices. It is often used for augmented reality and electronic image stabilization (EIS: Electronic Image Stabilization), optical image stabilization (OIS: Optical Image Stabilization), "Zero touch" gesture user interface.

Components Required

1 * MPU6050



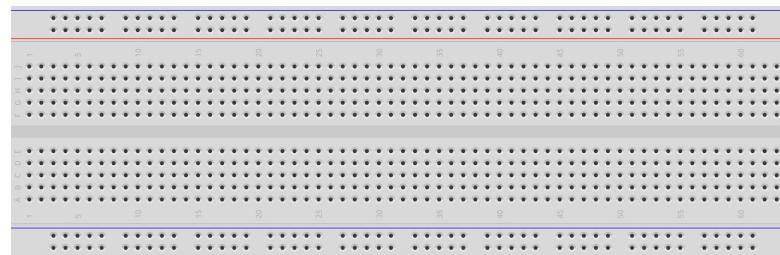
Several Jumper Wires

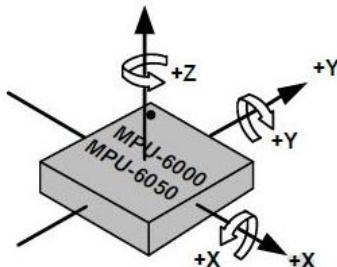


1 * Mega 2560 Board



1 * Breadboard



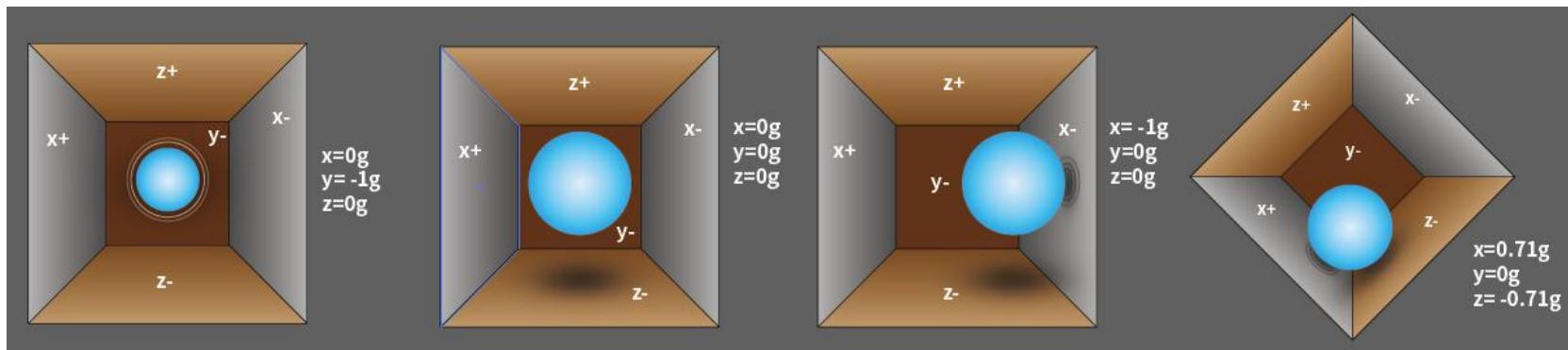


Component Introduction

Its three coordinate systems are defined as follows:

Put MPU6050 flat on the table, assure that the face with label is upward and a dot on this surface is on the top left corner. Then the upright direction upward is the z-axis of the chip. The direction from left to right is regarded as the X-axis. Accordingly the direction from back to front is defined as the Y-axis.

Acceleration



The accelerometer works on the principle of piezo electric effect, the ability of certain materials to generate an electric charge in response to applied mechanical stress.

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.

We can use the MPU6050 to detect its acceleration on each coordinate axis (in the stationary desktop state, the Z-axis acceleration is 1 gravity unit, and the X and Y axes are 0). If it is tilted or in a weightless/overweight condition, the corresponding reading will change.

There are four kinds of measuring ranges that can be selected programmatically: +/-2g, +/-4g, +/-8g, and +/-16g (+/-2g by default) corresponding to each precision. Values range from -32768 to 32767.

The reading of accelerometer is converted to an acceleration value by mapping the reading from the reading range to the measuring range.

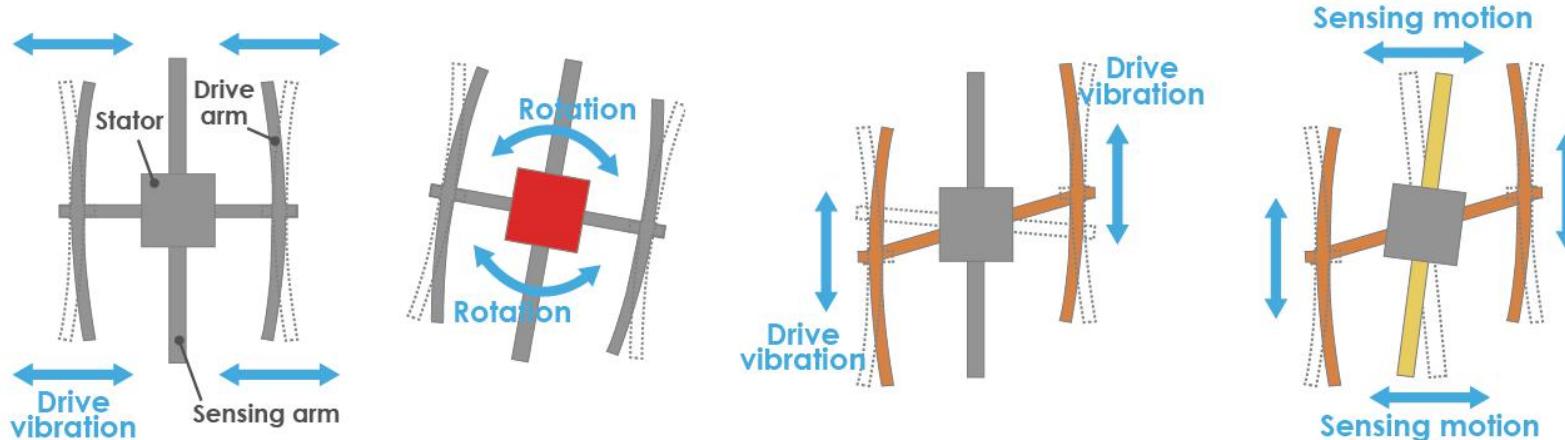
$$\text{Acceleration} = (\text{Accelerometer axis raw data} / 65536 * \text{full scale Acceleration range}) \text{ g}$$

Take the X-axis as an example, when Accelerometer X axis raw data is 16384 and the range is selected as +/-2g:

$$\text{Acceleration along the X axis} = (16384 / 65536 * 4) \text{ g} = 1\text{g}$$

Gyroscopes

Gyroscopes work on the principle of Coriolis acceleration. Imagine that there is a fork like structure, that 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.



1. Normally, a drive arm vibrates in a certain direction.

2. Direction of rotation

3. When the gyro is rotated, the Coriolis force acts on the drive arms, producing vertical vibration.

4. The stationary part bends due to vertical drive arm vibration, producing a sensing motion in the sensing arms.

The Gyroscope also has four kinds of measuring ranges: +/- 250, +/- 500, +/- 1000, +/- 2000 (+/-250degree/s by default). The calculation method and Acceleration are basically consistent.

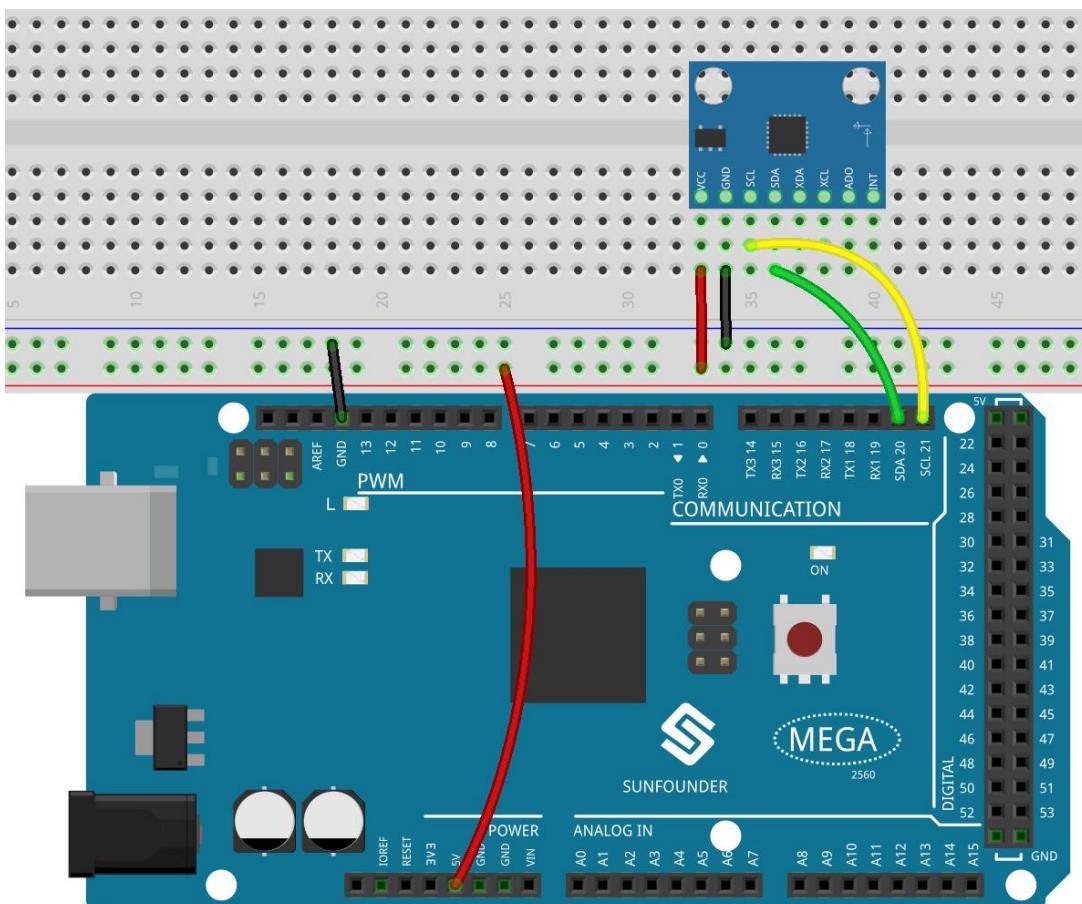
The formula for converting the reading into angular velocity is as follows:

$$\text{Angular velocity} = (\text{Gyroscope axis raw data} / 65536 \times \text{full scale Gyroscope range}) \text{ °/s}$$

The X axis, for example, the Accelerometer X axis raw data is 16384 and ranges + / - 250 ° / s:

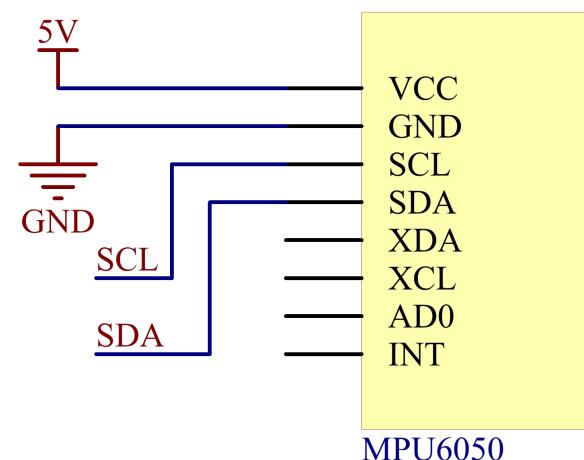
$$\text{Angular velocity along the X axis} = (16384 / 65536 \times 500) \text{ °/s} = 125 \text{ °/s}$$

Fritzing Circuit



In this example, we drive MPU6050 with IIC. We inset MPU6050 into the breadboard; get the VCC connected to 5V, GND to GND, SCL to pin SCL 21, and SDA to the pin SDA 20.

Schematic Diagram



Code

```
#define ACCELE_RANGE 4
#define GYROSC_RANGE 500

#include<Wire.h>
const int MPU_addr = 0x68; // I2C address of the MPU-6050
float AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ;
void setup() {
    Wire.begin();
    Wire.beginTransmission(MPU_addr);
    Wire.write(0x6B); // PWR_MGMT_1 register
    Wire.write(0); // set to zero (wakes up the MPU-6050)
    Wire.endTransmission(true);
    Serial.begin(9600);
}
void loop() {
    Wire.beginTransmission(MPU_addr);
    Wire.write(0x3B); // starting with register 0x3B (ACCEL_XOUT_H)
    Wire.endTransmission(false);
    Wire.requestFrom(MPU_addr, 14, true); // request a total of 14 registers
    AcX = Wire.read() << 8 | Wire.read(); // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
```

```

AcY = Wire.read() << 8 | Wire.read(); // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
AcZ = Wire.read() << 8 | Wire.read(); // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
Tmp = Wire.read() << 8 | Wire.read(); // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
GyX = Wire.read() << 8 | Wire.read(); // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
GyY = Wire.read() << 8 | Wire.read(); // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
GyZ = Wire.read() << 8 | Wire.read(); // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
Serial.print(" AcX = "); Serial.print(AcX / 65536 * ACCELE_RANGE); Serial.print("g ");
Serial.print(" | AcY = "); Serial.print(AcY / 65536 * ACCELE_RANGE); Serial.print("g ");
Serial.print(" | AcZ = "); Serial.print(AcZ/65536 * ACCELE_RANGE); Serial.println("g ");
Serial.print(" GyX = "); Serial.print(GyX / 65536 * GYROSC_RANGE); Serial.print("d/s ");
Serial.print(" | GyY = "); Serial.print(GyY/65536 * GYROSC_RANGE); Serial.print("d/s ");
Serial.print(" |GyZ= "); Serial.print(GyZ/65536*GYROSC_RANGE); Serial.println("d/s \n");
delay(500);
}

```

After uploading the codes to the Mega2560 board, you can open the serial monitor to see the gravity acceleration and angular velocity of MPU6050 in each direction.

Code Analysis

In the stationary desktop state, the Z-axis acceleration is 1 gravity unit, and the X and Y axes are 0.

Before your use, you need to calibrate the module, the methods are as follows:

- ① MPU6050 modules are placed horizontally on the desktop and then you can fix them with clamps or adhesive tape.
- ② Run the sample codes to get the RAW DATA of MPU6050 when it is static.
- ③ Add compensation according to the readings when MPU6050 is static.

Take MPU6050 we use as an example, and the results of compensation are as follows:

```
Serial.print(AcX / 65536 * ACCELE_RANGE - 0.02);
Serial.print(AcY / 65536 * ACCELE_RANGE + 0);
Serial.print(AcZ / 65536 * ACCELE_RANGE + 0.02);
Serial.print(GyX / 65536 * GYROSC_RANGE + 1.70);
Serial.print(GyY / 65536 * GYROSC_RANGE - 1.70);
Serial.print(GyZ / 65536 * GYROSC_RANGE + 0.25);
```

AcX = -0.00g AcY = -0.00g AcZ = 0.99g GyX = 0.14d/s GyY = -0.18d/s GyZ = -0.20d/s	AcX = 0.02g AcY = -0.00g AcZ = 0.98g GyX = -1.72d/s GyY = 1.67d/s GyZ = -0.31d/s
--	---

AcX = 0.00g AcY = 0.00g AcZ = 1.00g GyX = -0.02d/s GyY = -0.20d/s GyZ = -0.06d/s	AcX = 0.02g AcY = 0.00g AcZ = 0.98g GyX = -1.70d/s GyY = 1.63d/s GyZ = -0.29d/s
---	--

AcX = -0.00g AcY = -0.00g AcZ = 1.01g GyX = -0.06d/s GyY = -0.06d/s GyZ = 0.05d/s	AcX = 0.02g AcY = -0.00g AcZ = 0.97g GyX = -1.64d/s GyY = 1.60d/s GyZ = -0.26d/s
--	---

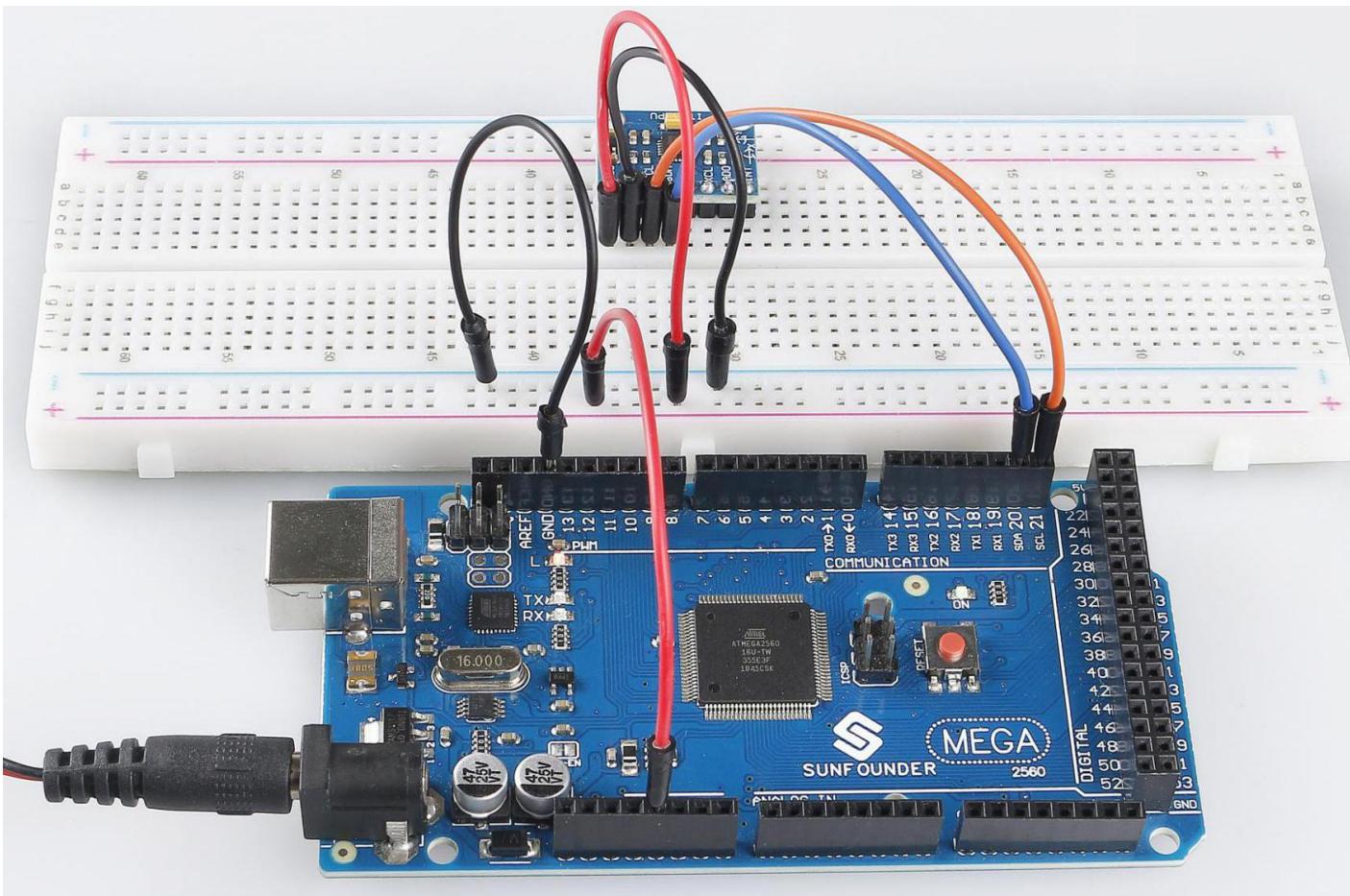
AcX = -0.01g AcY = 0.00g AcZ = 1.00g GyX = 0.04d/s GyY = -0.04d/s GyZ = -0.16d/s	AcX = 0.02g AcY = -0.00g AcZ = 0.98g GyX = -1.82d/s GyY = 1.52d/s GyZ = -0.20d/s
---	---

AcX = -0.01g AcY = 0.00g AcZ = 1.01g GyX = -0.15d/s GyY = -0.20d/s GyZ = -0.06d/s	AcX = 0.02g AcY = -0.00g AcZ = 0.98g GyX = -1.75d/s GyY = 1.75d/s GyZ = -0.39d/s
--	---

Before the calibration

After the calibration

Phenomenon Picture



2.35 RFID-RC522 Module

Overview

In this lesson, you will learn how to use RFID Module. RFID is the abbreviation of Radio Frequency Identification. Its working principle is to carry on the contactless data communication between the reader and the label to achieve the goal of identifying the target. The application of RFID is very extensive, currently the typical applications are animal chips, immobilizer, access control, parking control, production chain automation, material management and so on.

Components Required

- 1 * Card
- 1 * Key
- 1 * RFID



- 1 * Mega 2560 Board



- Several Jumper Wires



Component Introduction

Radio Frequency Identification (RFID) refers to technologies that involve using wireless communication between an object (or tag) and an interrogating device (or reader) to automatically track and identify such objects. The tag transmission range is limited to several meters from the reader. A clear line of sight between the reader and tag is not necessarily required.

Most tags contain at least one integrated circuit (IC) and an antenna. The microchip stores information and is responsible for managing the radio frequency (RF) communication with the reader. Passive tags do not have an independent energy source and depend on an external electromagnetic signal, provided by the reader, to power their operations. Active tags contain an independent energy source, such as a battery. Thus, they may have increased processing, transmission capabilities and range.

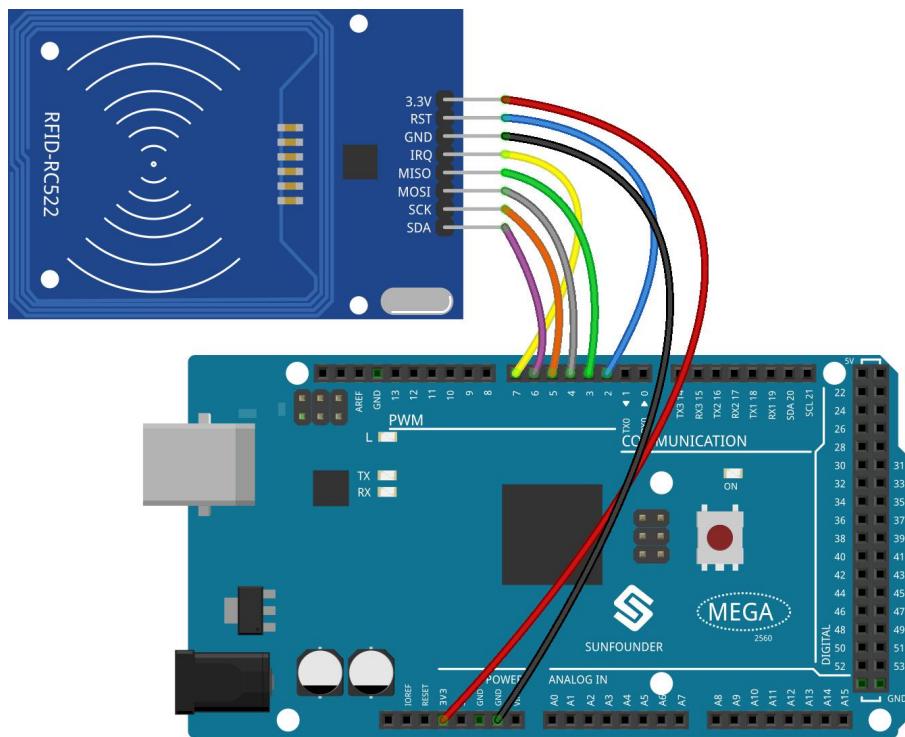
MFRC522

MF RC522 is a highly integrated read and write card chip applied to the 13.56MHz contactless communication. Launched by the NXP Company, it is a low-voltage, low-cost, and small-sized non-contact card chip, a best choice of intelligent instrument and portable hand held device.

The MF RC522 uses advanced modulation and demodulation concept which fully presented in all types of 13.56MHz passive contactless communication methods and protocols. In addition, it supports rapid CRYPTO1 encryption algorithm to verify MIFARE products.

MFRC522 also supports MIFARE series of high-speed non-contact communication, with a two-way data transmission rate up to 424kbit/s. As a new member of the 13.56MHz highly integrated reader card series, MF RC522 is much similar to the existing MF RC500 and MF RC530 but there also exists great differences.

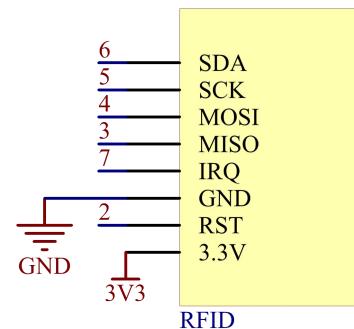
It communicates with the host machine via the serial manner which needs less wiring. You can choose between SPI, I2C and serial UART mode (similar to RS232), which helps reduce the connection, save PCB board space (smaller size), and reduce cost.



Fritzing Circuit

In this example, we insert the RFID into the breadboard. Get the 3.3V of RFID connected to 3.3V, GND to GND, RST to pin 2, SDA to pin 6, SCK to pin 5, MOSI to pin 4, MISO to pin 3 and IRQ to pin 7.

Schematic Diagram



Code

The codes use the rfid1.h library. About how to import the library, please refer to [Part 4 - 4.1 Add Libraries](#).

```
#include"rfid1.h"
RFID1 rfid; //create a variable type of RFID1
uchar serNum[5]; // array to store your ID

void setup()
{
    Serial.begin(9600); //initialize the serial
    rfid.begin(7, 5, 4, 3, 6, 2); //rfid.begin(IRQ_PIN,SCK_PIN,MOSI_PIN,MISO_PIN,SDA_PIN,RST_PIN)
    delay(100); //delay 1 second
    rfid.init(); //initialize the RFID
}

void loop()
{
    uchar status;
    uchar str[MAX_LEN]; //The maximum length of the array is 16

    // Search card, return card types
    status = rfid.request(PICC_REQIDL, str);
    if (status != MI_OK)
```

```
{  
    return;  
}  
Serial.print("Card type: ");  
Serial.println(rfid.readCardType(str));  
//Prevent conflict, return the 4 bytes Serial number of the card  
status = rfid.anticoll(str);  
if (status == MI_OK)  
{  
    Serial.print("The card's number is: ");  
    int IDlen=4;  
    for(int i=0; i<IDlen; i++){  
        Serial.print(str[i],HEX);  
    }  
    Serial.println();  
    Serial.println();  
}  
  
delay(500);  
rfid.halt(); //command the card into sleep mode  
}
```

Uploaded the codes to the Mega2560 board, you can get your RFID card (secret key) close to the RFID Reader. The module will read the card information and then print it on the serial monitor.

Code Analysis

The functions of the module are included in the library rfid1.h.

```
#include <rfid1.h>
```

Library Functions:

RFID1

Create a new instance of the rfid1 class that represents a particular RFID module attached to your Arduino .

```
void begin(IRQ_PIN,SCK_PIN,MOSI_PIN,MISO_PIN,SDA_PIN,RST_PIN)
```

Pin configuration.

IRQ_PIN,SCK_PIN,MOSI_PIN,MISO_PIN: the pins used for the SPI communication.

SDA_PIN: Synchronous data adapter.

RST_PIN: The pins used for reset.

```
void init()
```

Initialize the RFID.

uchar request(uchar reqMode, uchar *TagType);

Search card and read card type, and the function will return the current read status of RFID and return MI_OK if succeeded.

reqMode: Search methods. PICC_REQIDL is defined that 0x26 command bits (Search the cards that does not in the sleep mode in the antenna area).

***TagType:** It is used to store card type, and its value can be 4byte (e.g. 0x0400).

char * readCardType(uchar *TagType)

This function decodes the four-digit hexadecimal number of ***tagType** into the specific card type and returns a string. If passed 0x0400, "MFOne-S50" will be returned.

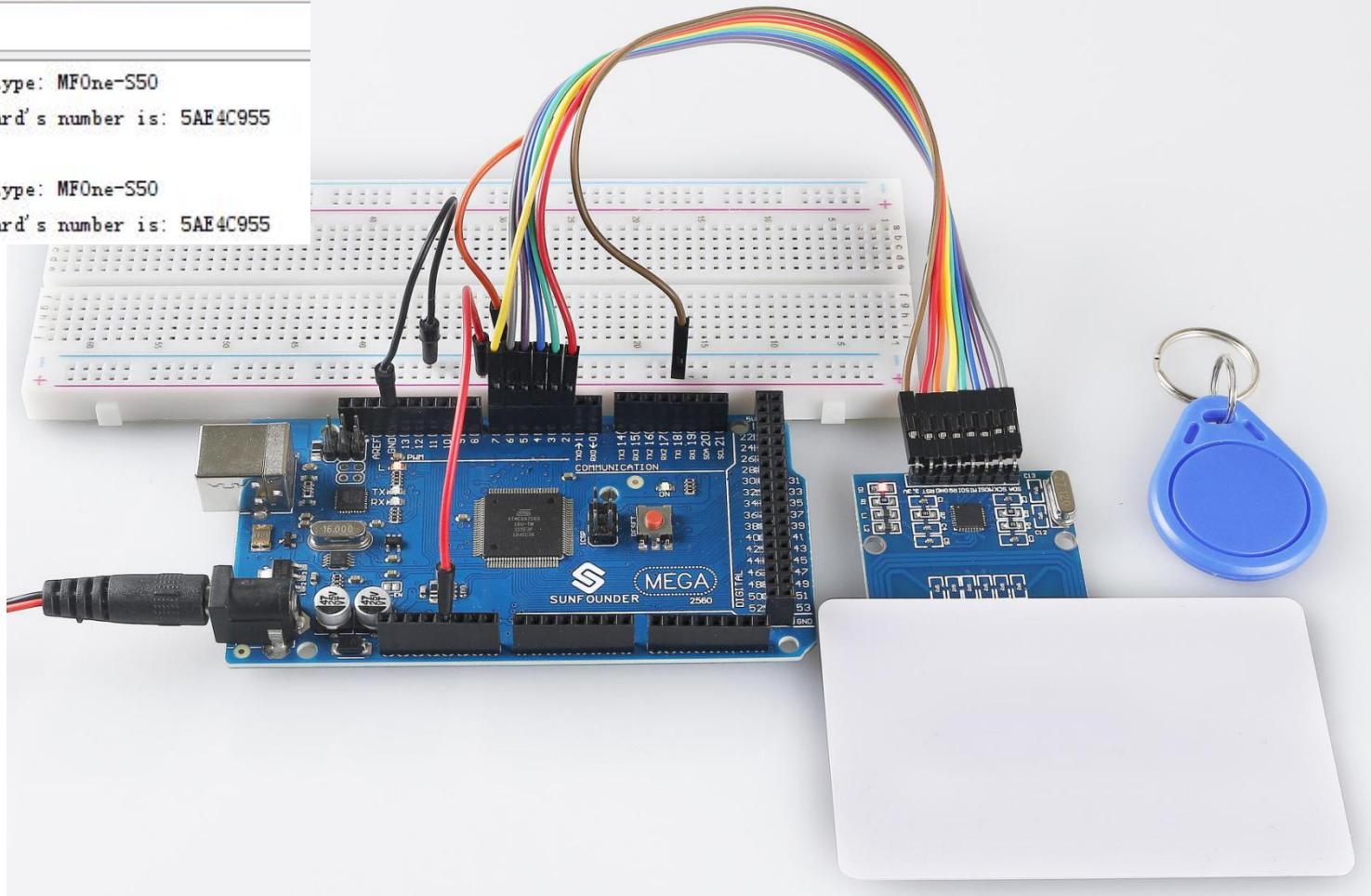
uchar anticoll(uchar *serNum);

Prevent conflict, and read the card serial number. The function will return the current reading status of RFID. It returns MI_OK if succeeded.

***serNum:** It is used to store the card serial number, and return the 4 bytes card serial number. The 5th byte is recheck byte(e.g. e.g. my magnetic card ID is 5AE4C955).

COM19

```
Card type: MFOne-S50  
The card's number is: 5AE4C955  
  
Card type: MFOne-S50  
The card's number is: 5AE4C955
```



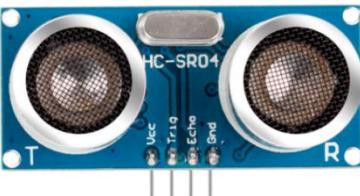
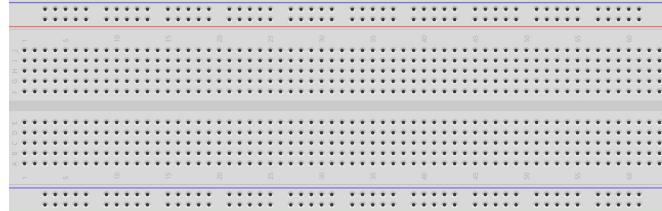
Part 3: Example

3.1 Reversing Aid

Overview

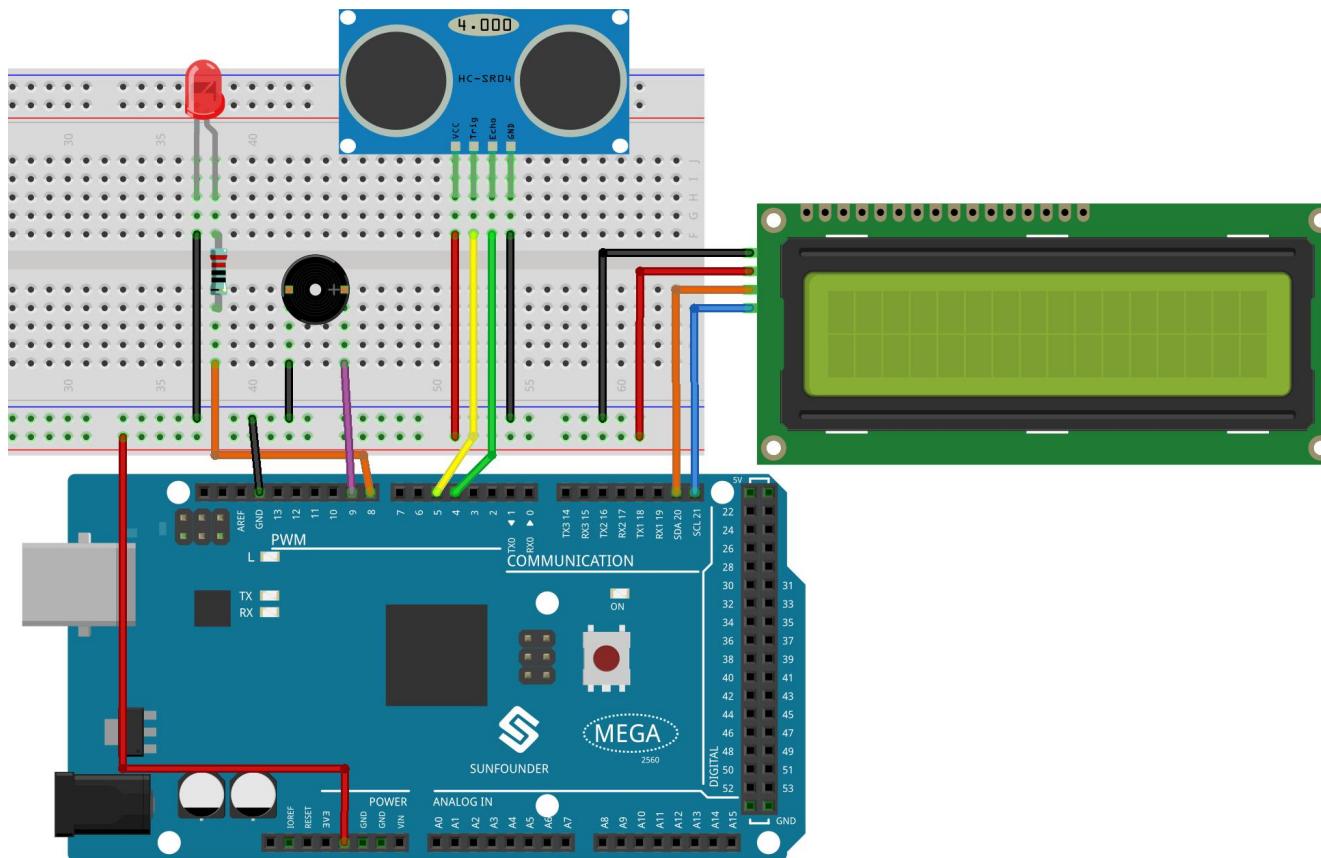
With the development of science and technology, a lot of high-tech products have been installed in cars, among which the reversing assist system is one of them. Here we use ultrasonic sensors, LCD, LED and buzzer to make a simple ultrasonic reversing assist system.

Components Required

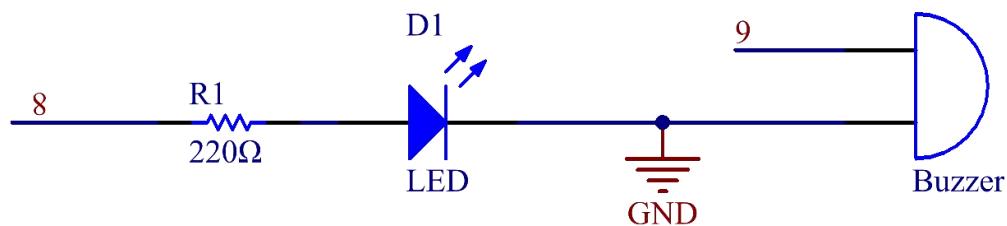
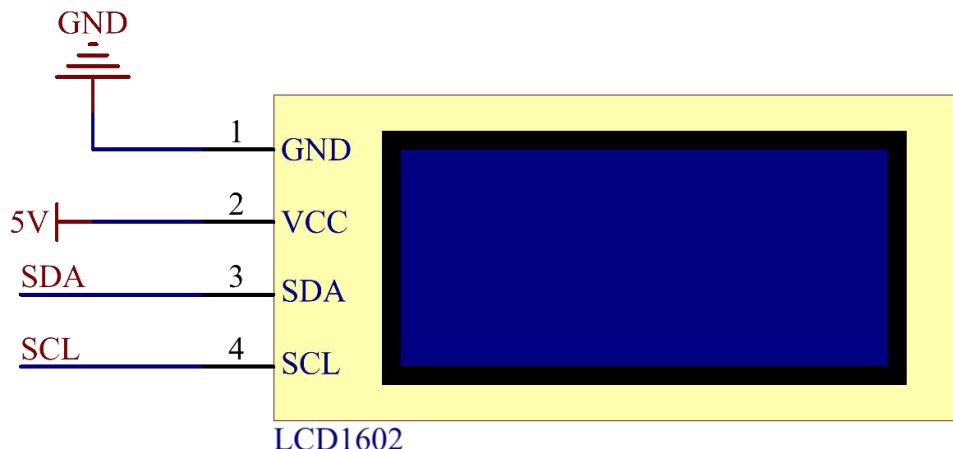
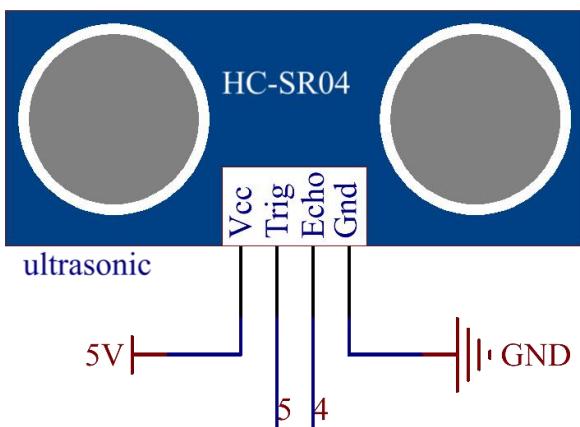
1 * Ultrasonic Module 	1 * Active Buzzer  Several Jumper Wires 	1 * I2C LCD1602 
1 * Mega 2560 Board 	1 * Breadboard 	

Fritzing Circuit

In this example, the wiring is shown below.

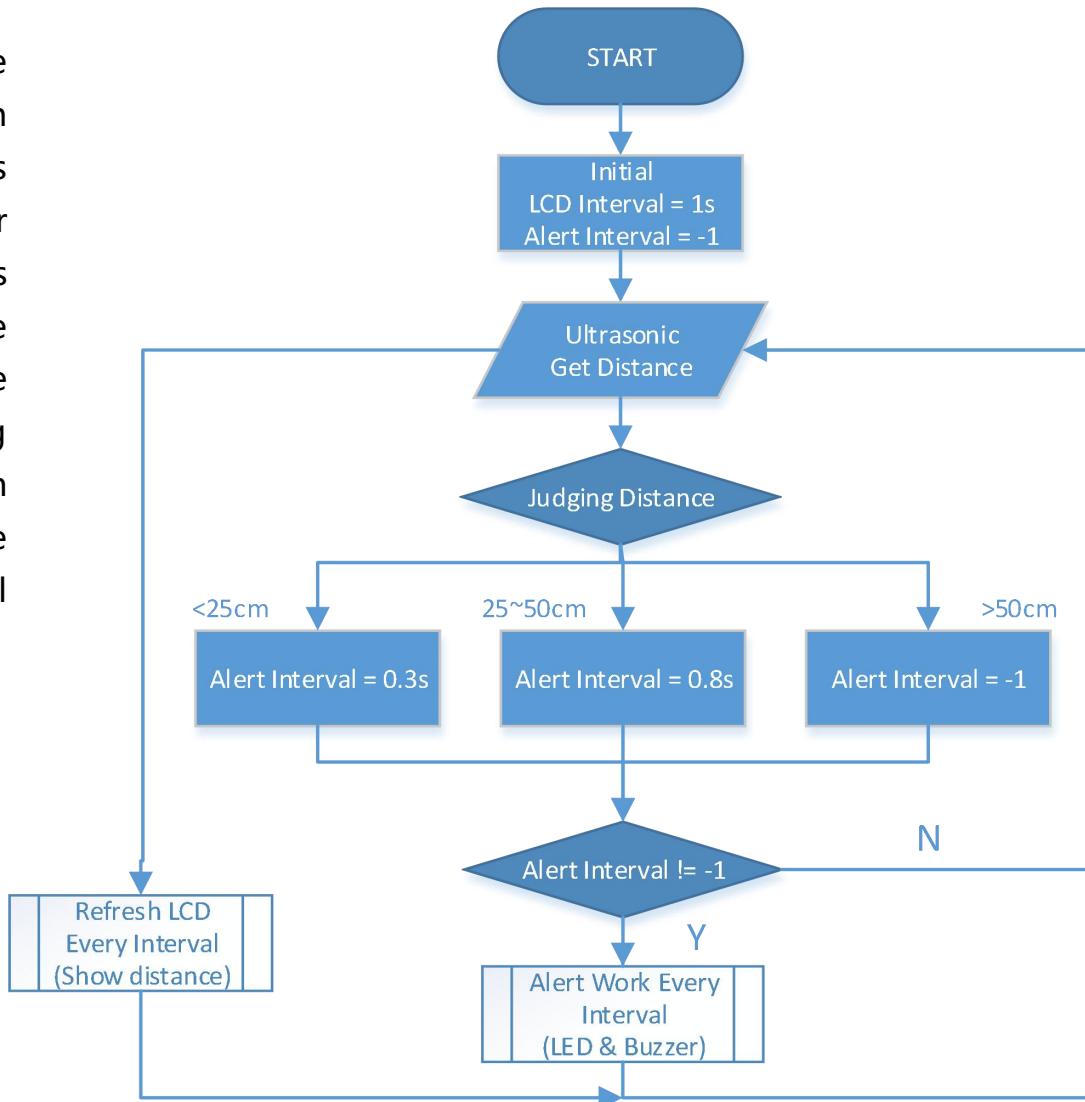


Schematic Diagram

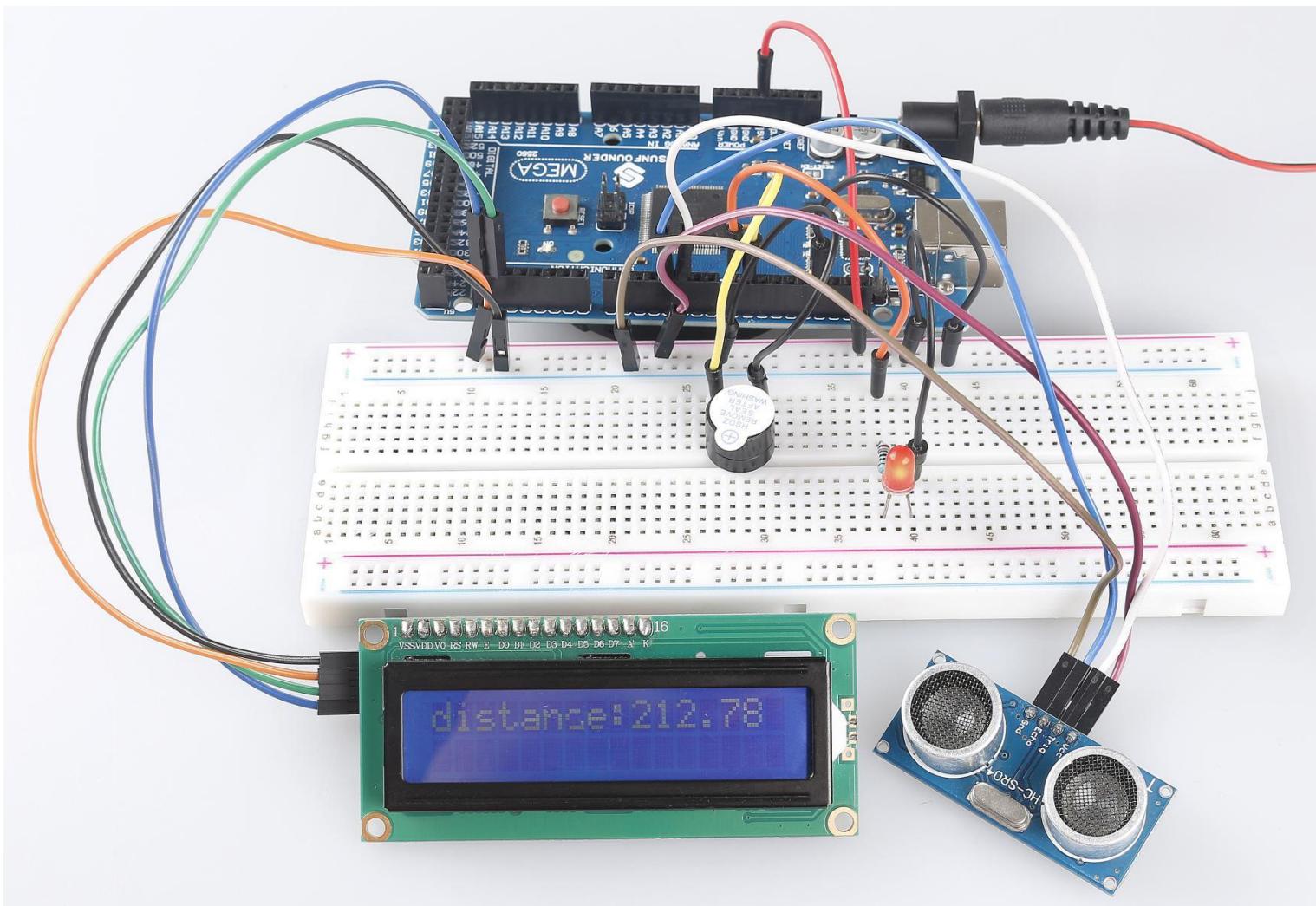


Example Explanation

In this project, we need to avoid the interference between the LCD screen and the alarm system as much as possible (for example, the LED flicker time is too long and the LCD refresh is delayed), so please avoid using the `delay()` statement and use two separate intervals to control the working frequency of the LCD and alarm system respectively. Its workflow is shown in the flow chart. For analysis of Interval function, refer to [Part 1-1.11 Interval](#).



Phenomenon Picture

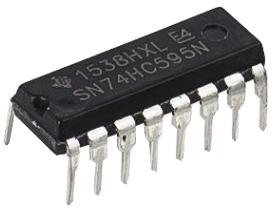
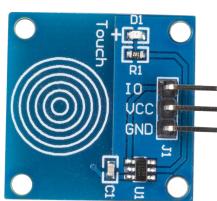
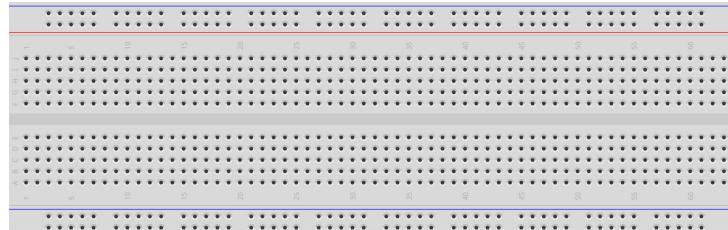


3.2 Pedestrian Crossing Button

Overview

When pedestrians cross the street, they just need to touch the button on the lamppost of the roadside signal lamp, and the green light above the traffic lane will turn into red then pedestrians can pass safely. Thus, the hard situation of citizens crossing the street is comprehensively resolved. At the same time, when there is no pedestrian to press, the light above the lane that is set for vehicles to pass will always be green, thus greatly improving the use efficiency of the road and traffic capacity.

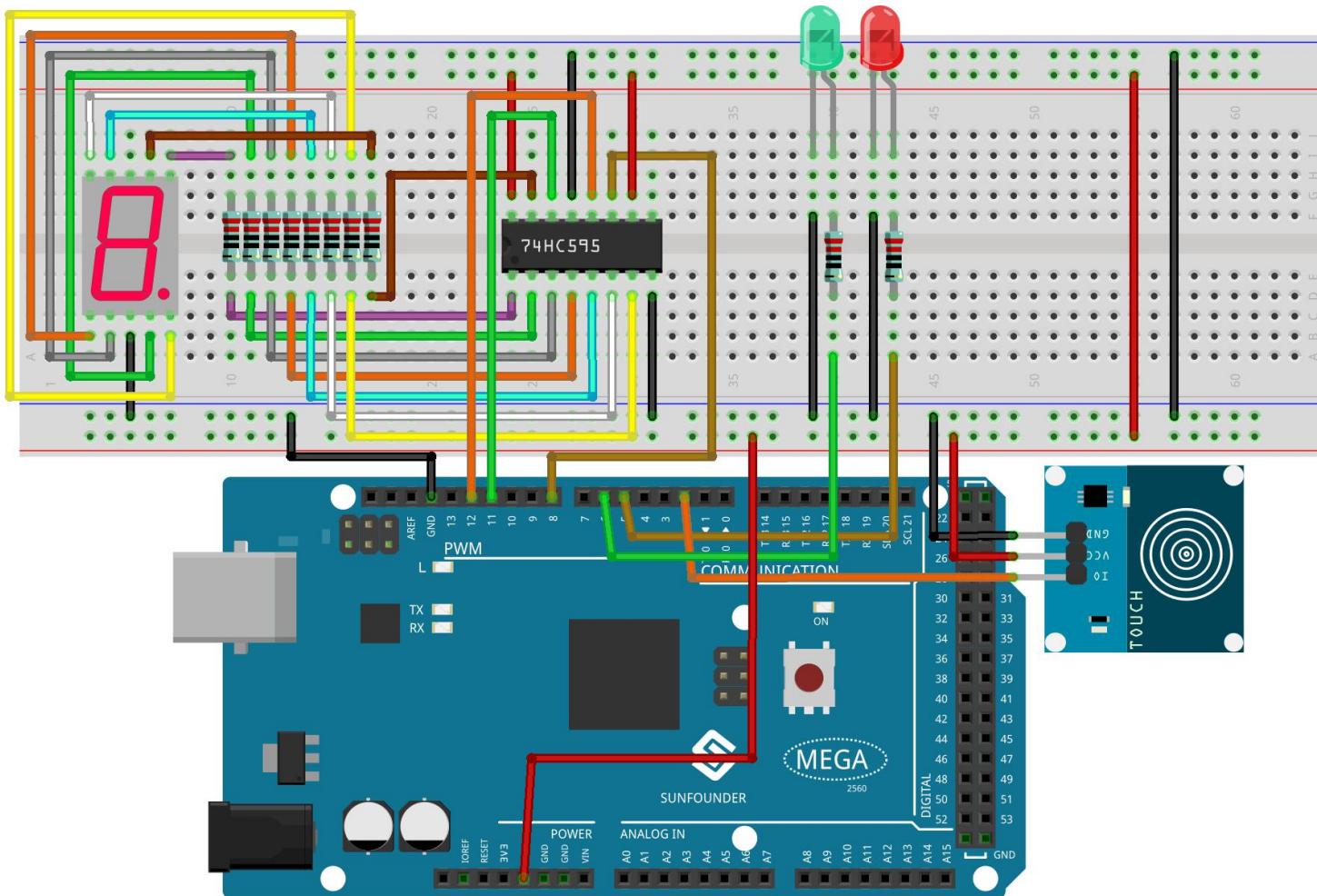
Components Required

1 * 74hc595 	2 * led 	1 * 7-Segment 	1 * Touch 	10 * 220Ω resistor  Several Jumper Wires 
1 * Mega 2560 Board 		1 * Breadboard 		

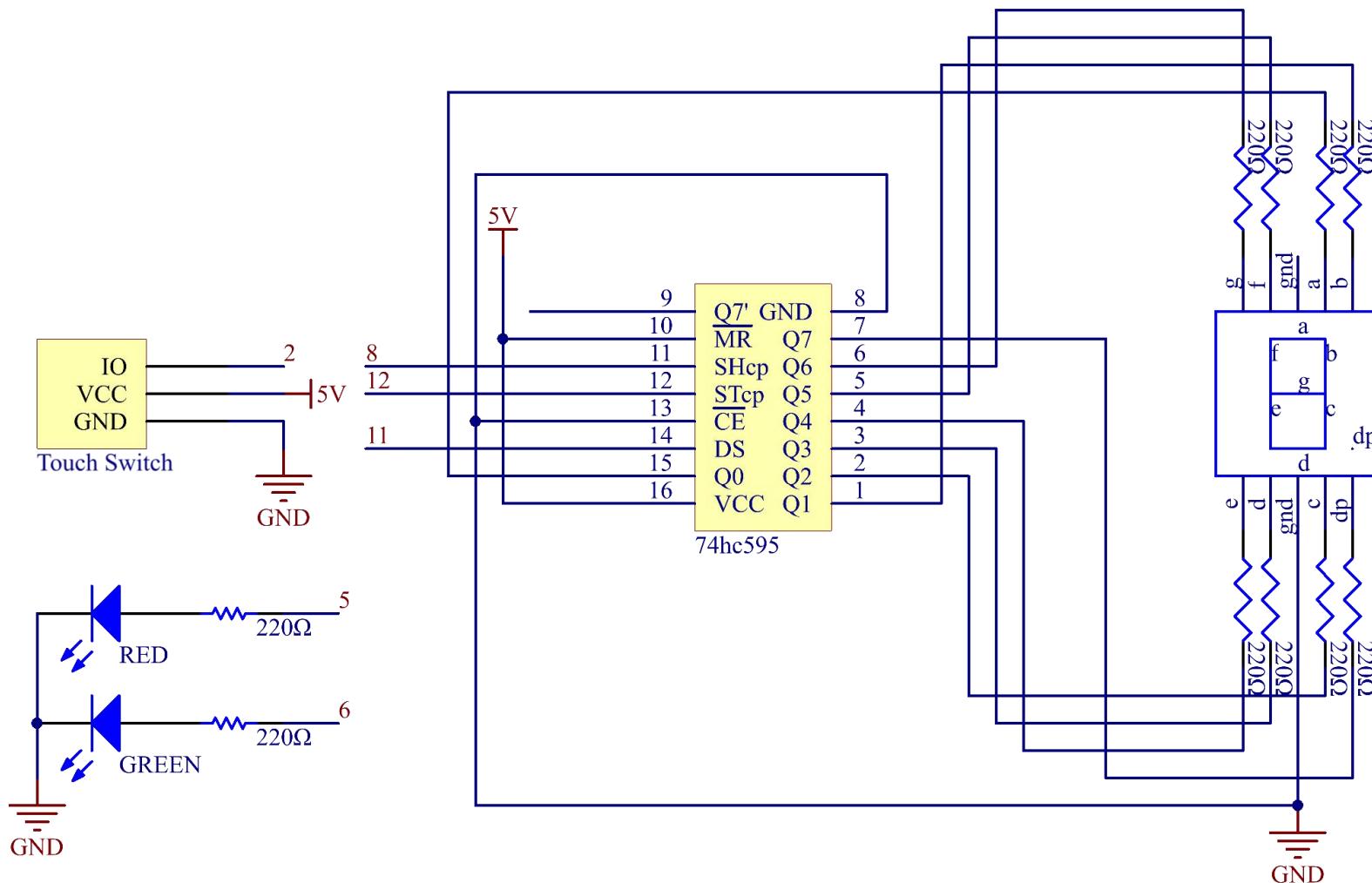
Fritzing Circuit

In this example, 74hc595, 7-Segment, LED, touch sensor are to be connected according to the chart.

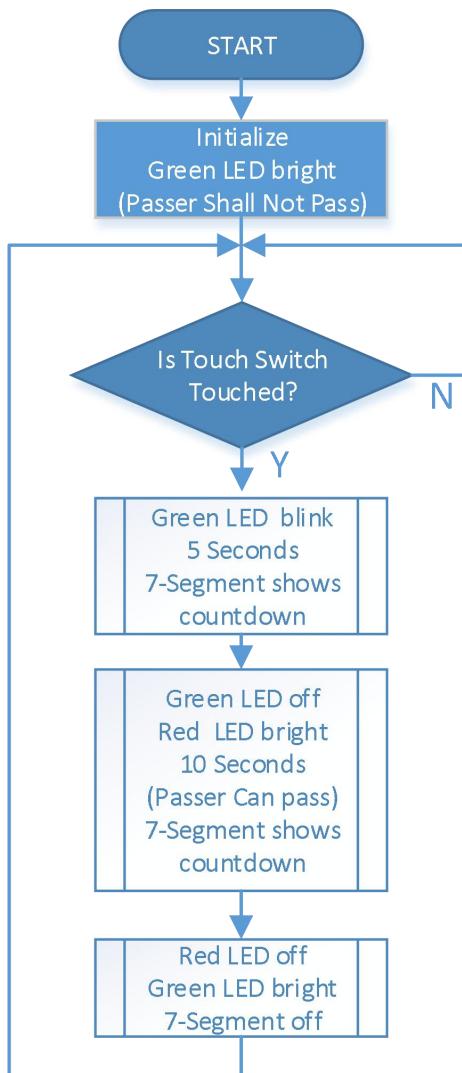
Mega 2560 Board	74hc595	7-Segment
	Q1	b
	Q2	c
	Q3	d
	Q4	e
	Q5	f
	Q6	g
	Q7	dp
GND	GND	
5V	MR	
8	SHcp	
12	STcp	
GND	CE	
11	DS	
	Q0	a
5V	VCC	



Schematic Diagram



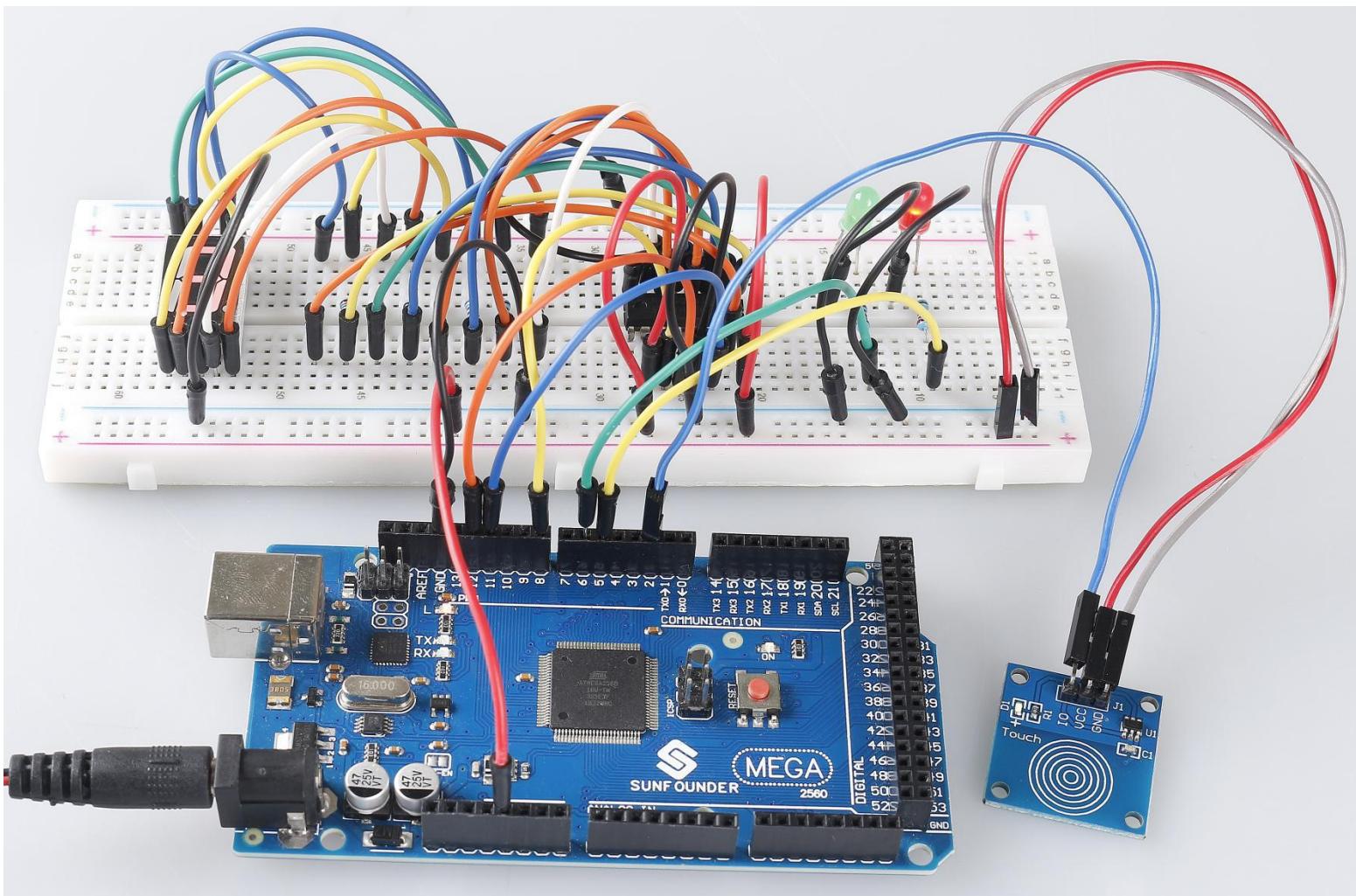
Example Explanation



The workflow of the project is as shown in the flow chart. The function of number display of 7-Segment is realized by writing 8 bit data into 74HC595. When there is a need of displaying 「0」, the pins abcdef of the segment display will be connected to the high level. The pins, g and dp need to be connected to low level to write 「0x3f」 (B00111111) in the codes. The complete codes for number display of 7-Segment are as follows.

Numbers	Common Cathode	
	(DP)GFEDCBA	Hex Code
0	00111111	0x3f
1	00000110	0x06
2	01011011	0x5b
3	01001111	0x4f
4	01100110	0x66
5	01101101	0x6d
6	01111101	0x7d
7	00000111	0x07
8	01111111	0x7f
9	01101111	0x6f

Phenomenon Picture

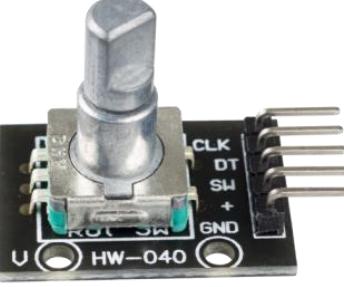


3.3 Overheat Monitor

Overview

You may want to make an overheat monitoring device that applies to various situations. When the temperature of room is above 30°C in summer, the electric fan or the air conditioner will be turned on automatically. If the refrigerator stops to refrigerate, there will emit alarm. When the CPU gets overheated, the water-cooling system turns on. Next, we will use thermistor, relay, button, rotary encoder and LCD to make an intelligent temperature monitoring device whose threshold is adjustable. You can make it suitable for the scene you want by inserting different peripherals into the relay and using a rotary encoder to adjust the high temperature threshold.

Components Required

1 * Rotary Encoder	1 * button	1 * Relay Module	1 * thermistor	1 * led
 A black PCB with a silver cylindrical potentiometer and four metal pins labeled CLK, DT, SW, and GND. Below the potentiometer are two small blue LEDs labeled U and HW-040.	 A black rectangular pushbutton switch with a central black dome and four metal pins labeled U, SW, GND, and VCC.	 A blue PCB with a blue relay component labeled 5VDC, 10A 250VAC, 15A 125VAC, and JQC-3FF-S-Z. It has four metal pins labeled U, P1, T1C, and P2.	 A small black cylindrical component with two thin metal leads, labeled 103.	 A red LED with a clear plastic housing and two metal leads.

1 * 220Ω resistor



2 * 10KΩ resistor



Several Jumper Wires



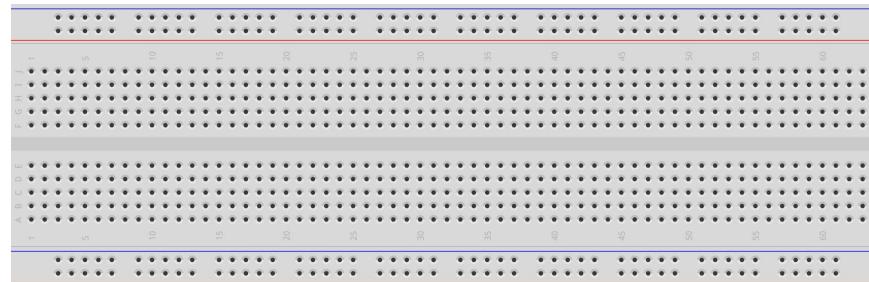
1 * Mega 2560 Board



1 * LCD1602



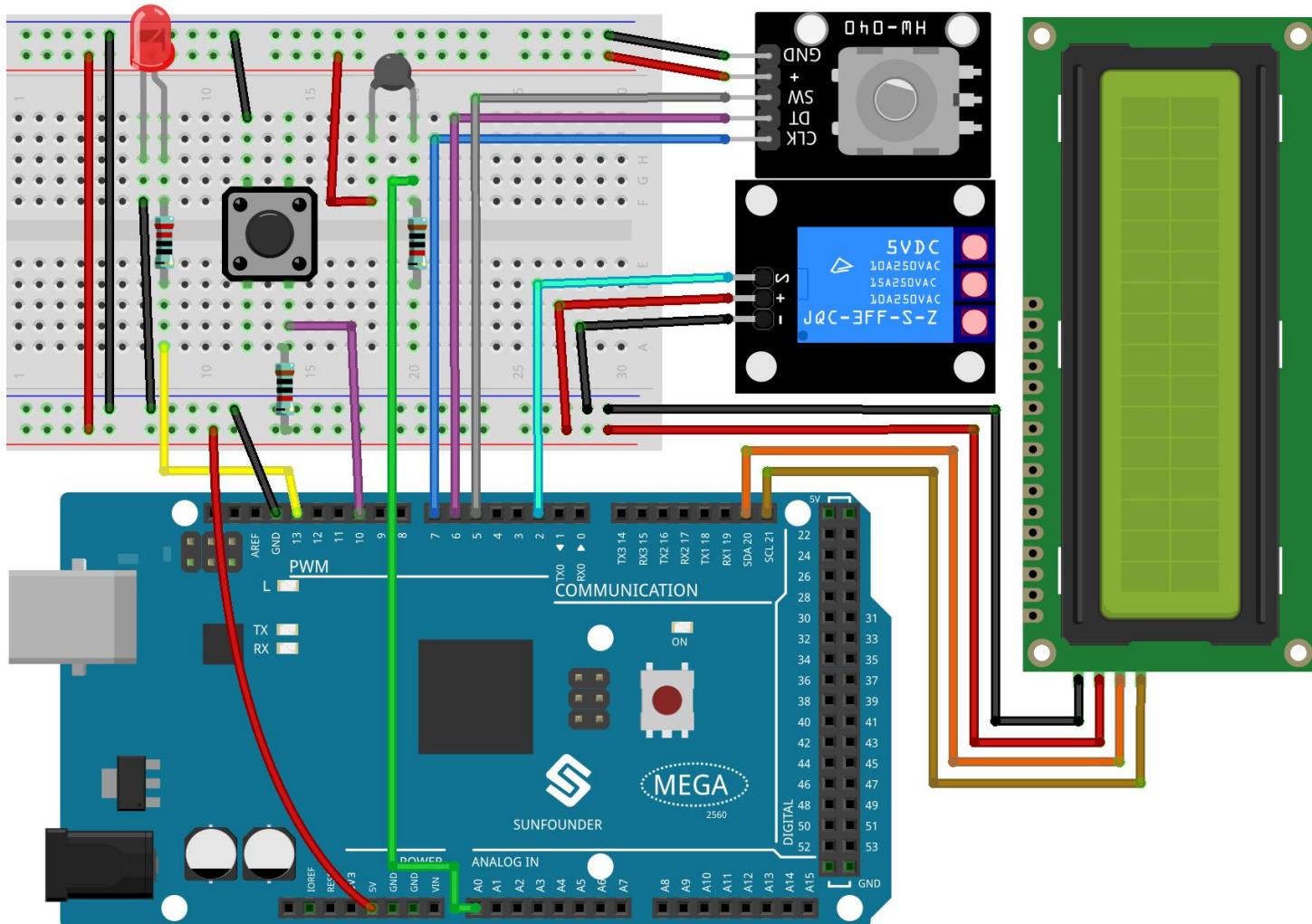
1 * Breadboard



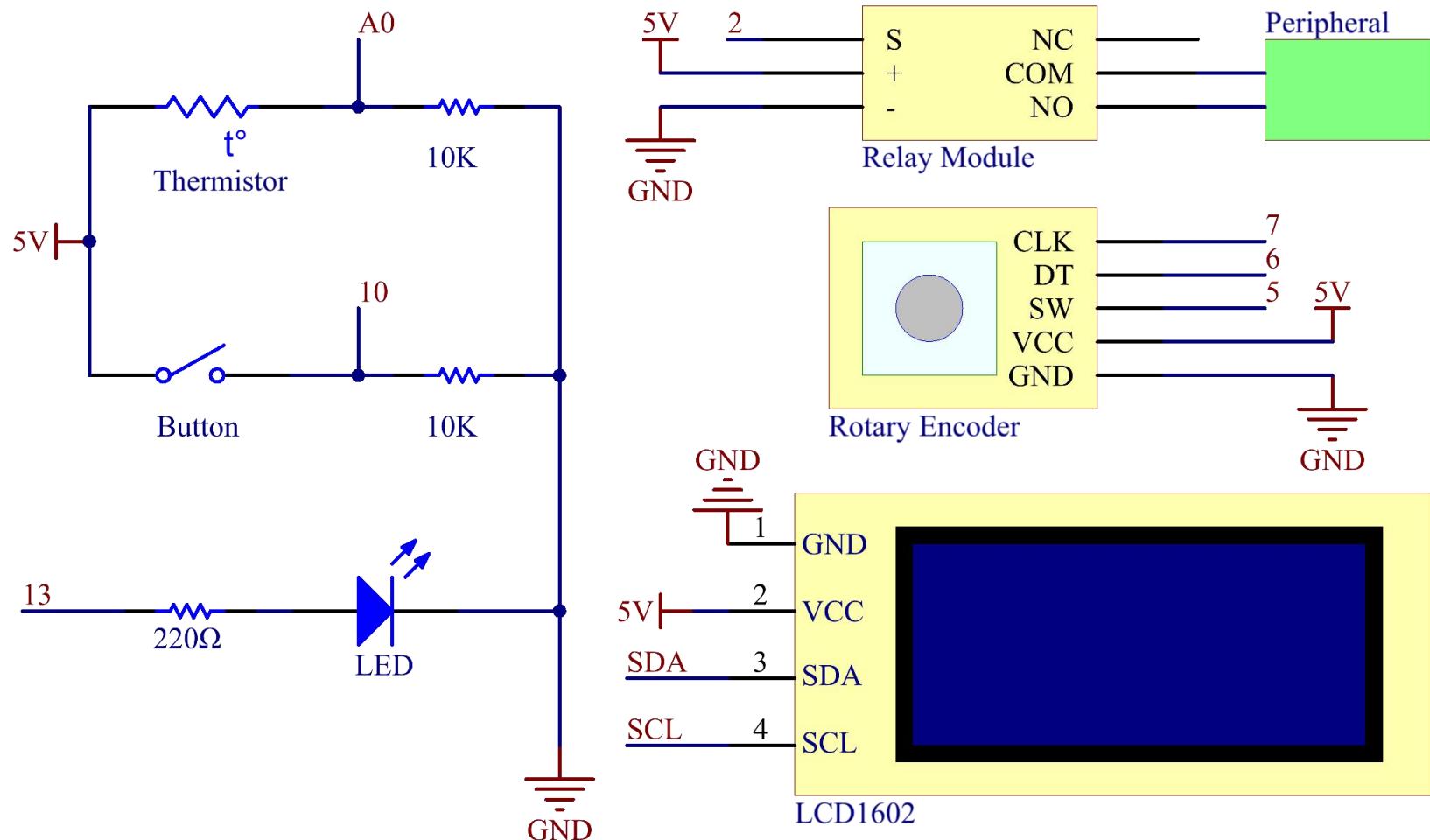
Fritzing Circuit

In this example, the component modules are connected as shown in the table.

Mega 2560 Board	LCD1602	Rotary Encoder	Relay Module	LED	thermistor	button
GND	GND	GND	(-)	(-)	(-)	(-)
5V	VCC	VCC	(+)			
SDA	SDA					
SCL	SCL					
7		CLK				
6		DT				
5		SW				
2			S			
13				(+)		
A0					(+)	
10						(+)

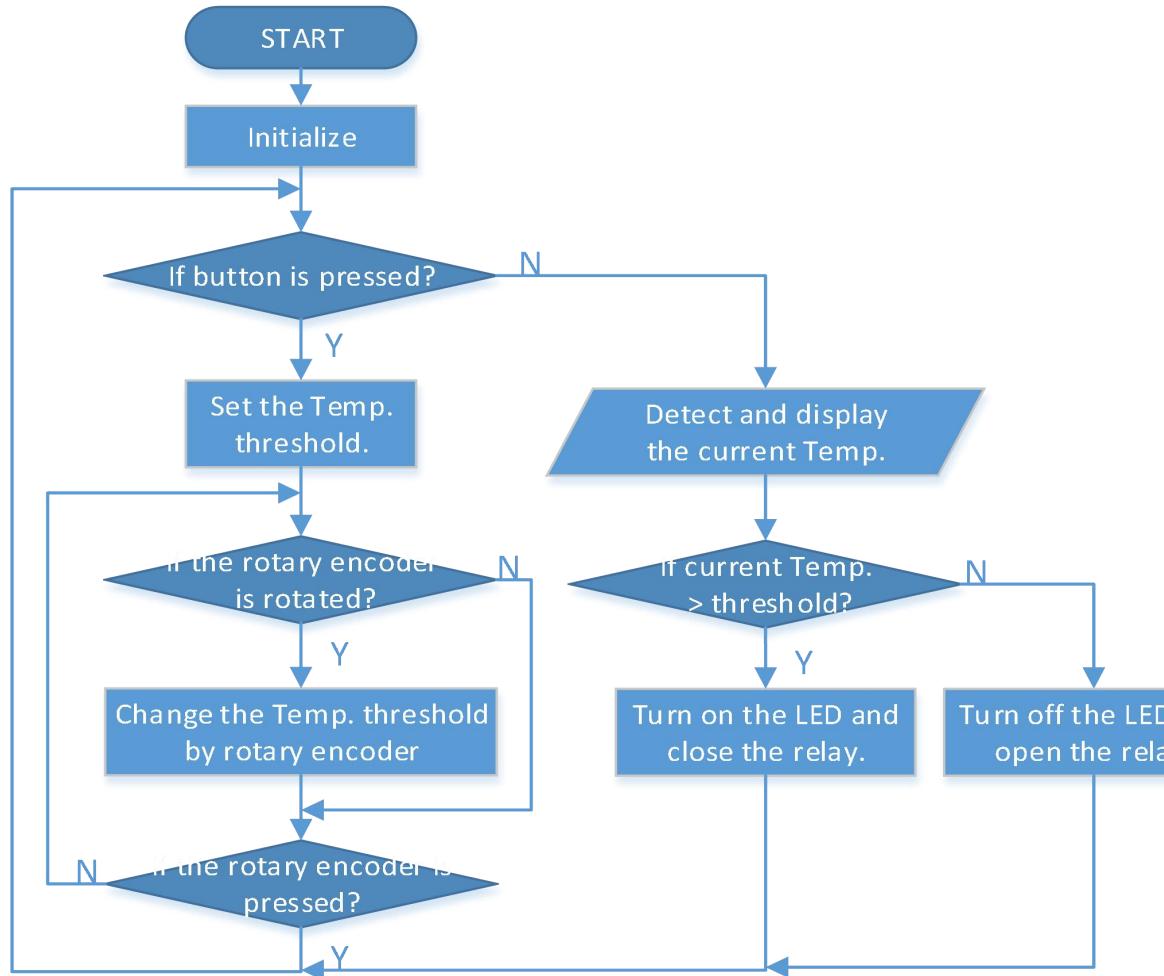


Schematic Diagram



Example Explanation

The flow diagram of the project is as follows:



By using EEPROM.h library, the high temperature threshold is saved in EEPROM to avoid the value reset after the restart of MCU.

Library Functions:

`void write(address,value)`

Write a byte to the EEPROM.

`void Read(address)`

Reads a byte from the EEPROM. Locations that have never been written to have the value of 255.

`void update(address,value)`

Write a byte to the EEPROM. The value is written only if differs from the one already saved at the same address.

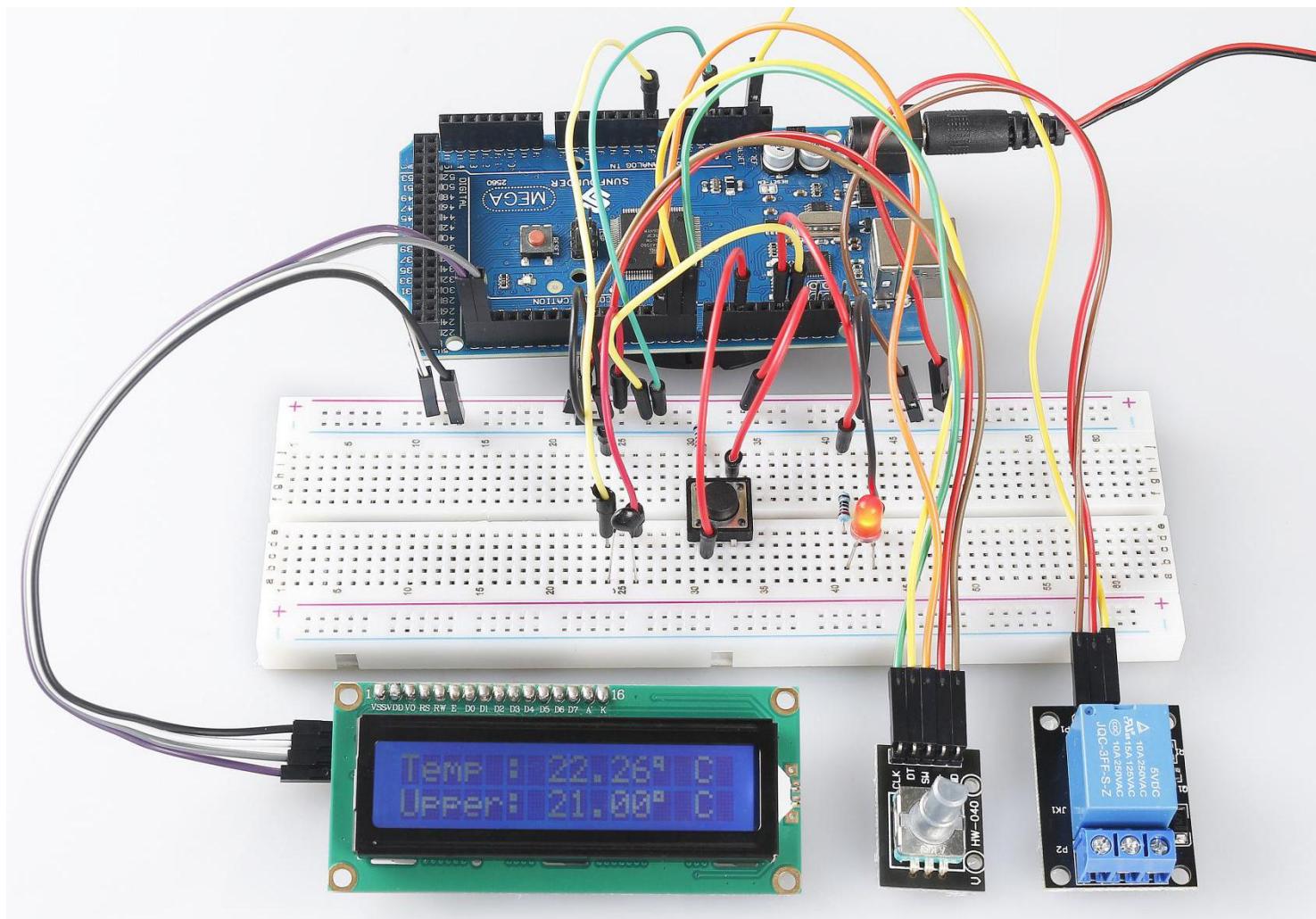
`void put(address,value)`

Write any data type or object to the EEPROM.

`void get(address)`

Read any data type or object from the EEPROM.

Phenomenon Picture



3.4 Guess Number

Overview

Guessing Numbers is a fun party game where you and your friends take turns inputting a number (0~99). The range will be smaller with the inputting of the number till a player answers the riddle correctly. Then the player is defeated and punished. For example, if the lucky number is 51 which the players cannot see, and the player ① inputs 50, the prompt of number range changes to 50~99; if the player ② inputs 70, the range of number can be 50~70; if the player ③ inputs 51, he or she is the unlucky one. Here, we use IR Remote Controller to input numbers and use LCD to output outcomes.

Components Required

1 * LCD1602



1 * IR Remote Controller



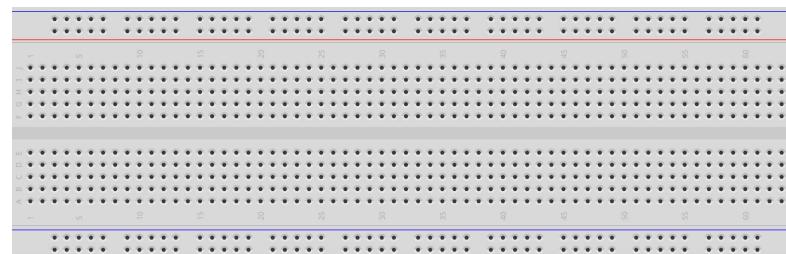
Several Jumper Wires



1 * Mega 2560 Board

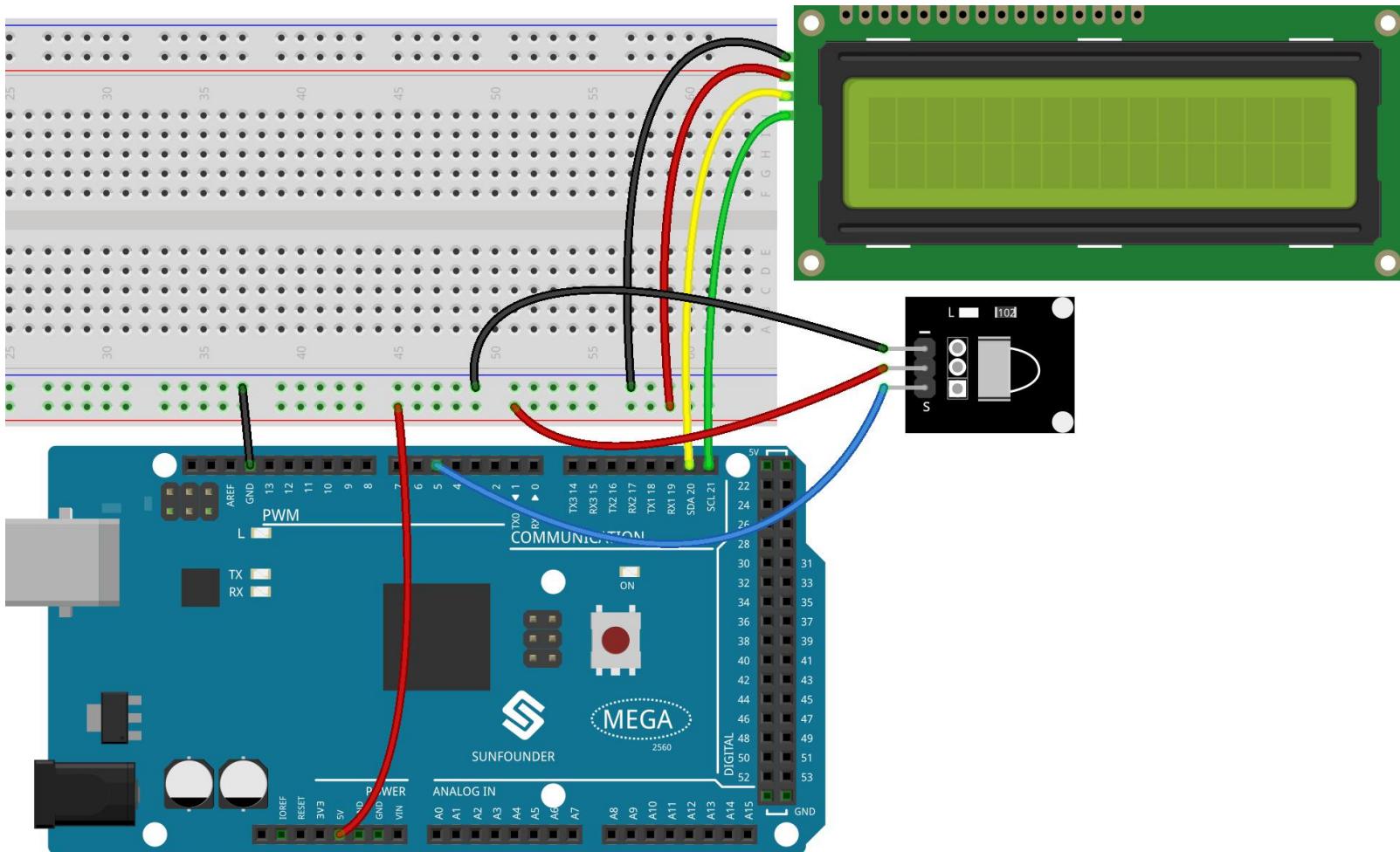


1 * Breadboard

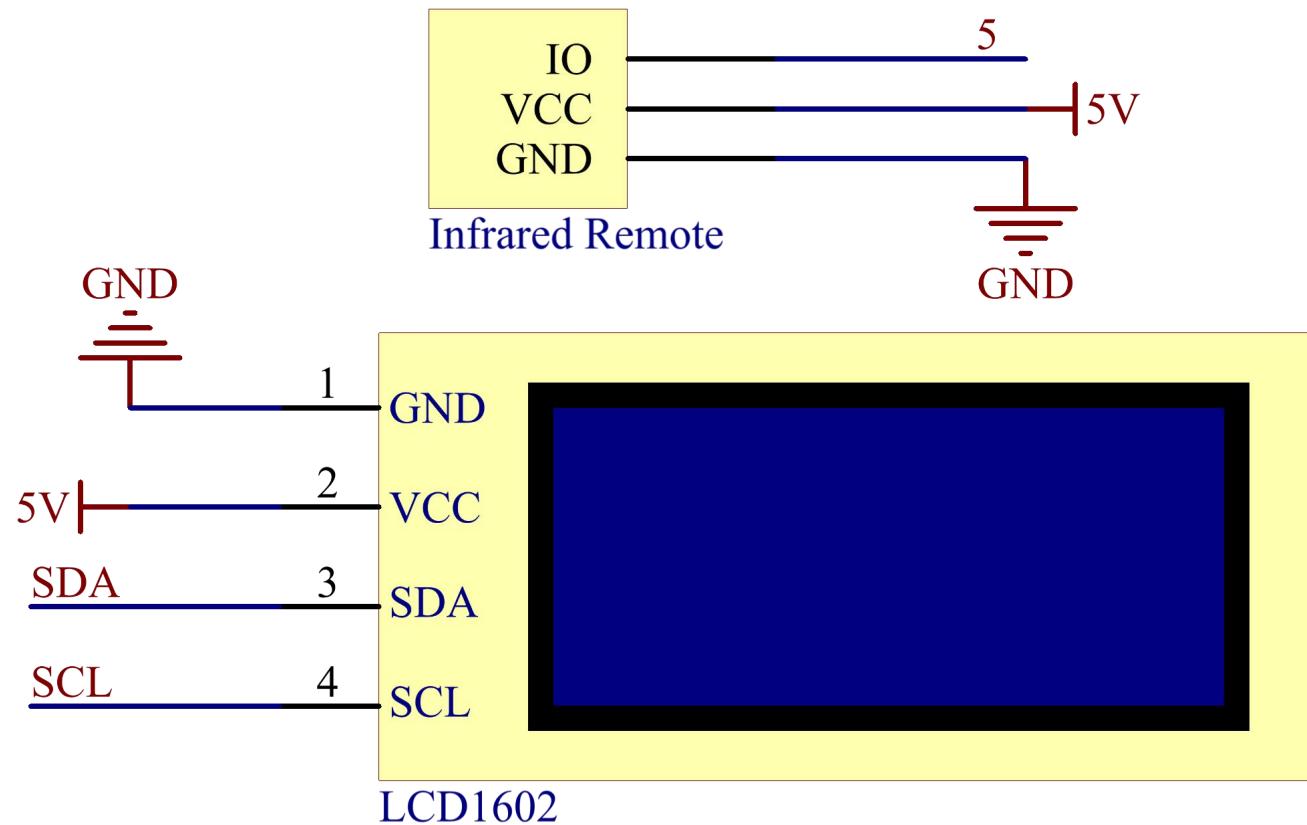


Fritzing Circuit

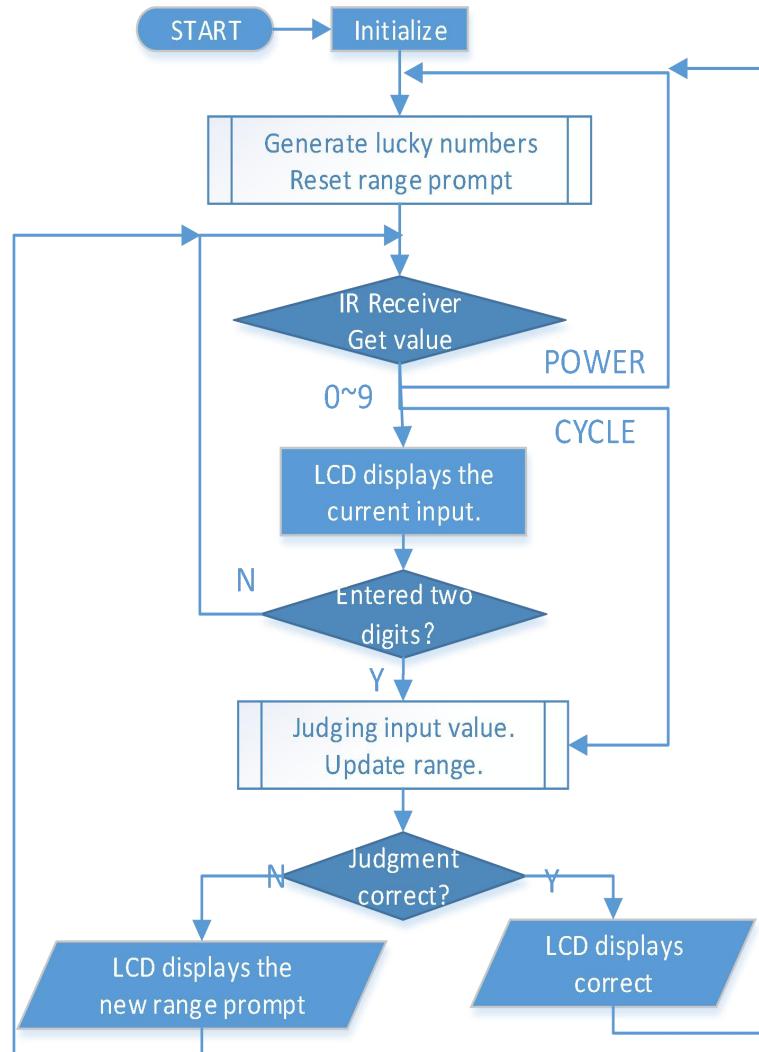
In this example, the wiring of LCD1602 and infrared receiving module is as follows.



Schematic Diagram



Example Explanation

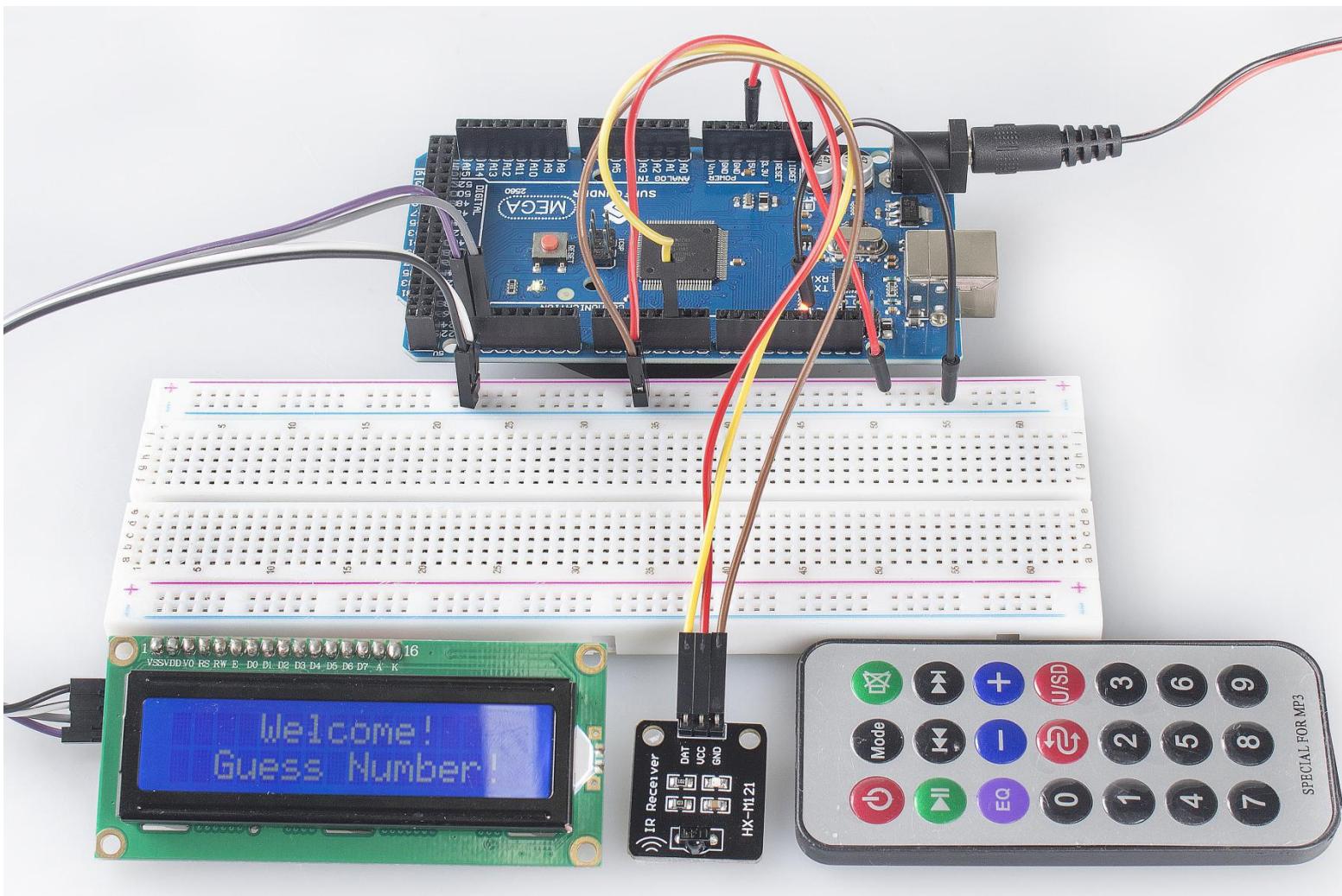


In order to make the number guessing game become vivid and funny, we need to achieve the following functions:

- ① The lucky number will be displayed when we start and reset the game, and the number range prompt is reset to 0 ~ 99.
- ② LCD will display the number being input and the number range prompt.
- ③ After inputting two digits, there appears result judgment automatically.
- ④ If you input a single digit, you can press the CYCLE key (the key at the center of the Controller) to start the result judgment.
- ⑤ If the answer is not guessed, the new number range prompt will be displayed (if the lucky number is 51 and you enter 50, the number range prompt will change to 50~99).
- ⑥ The game is automatically reset after the lucky number is guessed, so that the player can play a new round.
- ⑦ The game can be reset by directly pressing the POWER button (the button in the upper left corner).

In conclusion, the work flow of the project is shown in the flow chart.

Phenomenon Picture



3.5 Access Control System

Overview

The access control system is the system that is used to control the entrance channel, which is developed on the basis of the traditional door lock. The traditional mechanical door lock is only a simple mechanical device, and no matter how reasonable the structural design is, how strong the material is, people can always open it through various means. The key to the entrance and exit (like an office building, a hotel room) is cumbersome. If the key is missed or replaced, the lock is to be replaced with the key. In order to solve these problems, the electronic magnetic card lock and the electronic coded lock are present, which has raised the management level of the access channel to a certain extent, and then the channel management enters into the electronic age.

Components Required

1 * RFID



1 * Keypad



1 * Power Supply Module



1 * I2C LCD1602



1 * ULN2003



1 * Buzzer



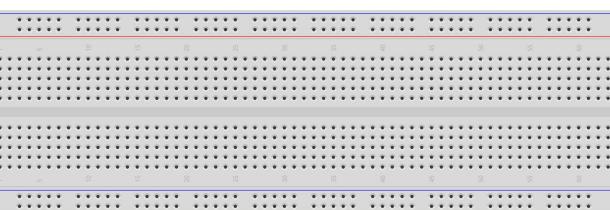
1 * Stepper Motor



1 * Mega 2560 Board



1 * Breadboard



Several Jumper Wires

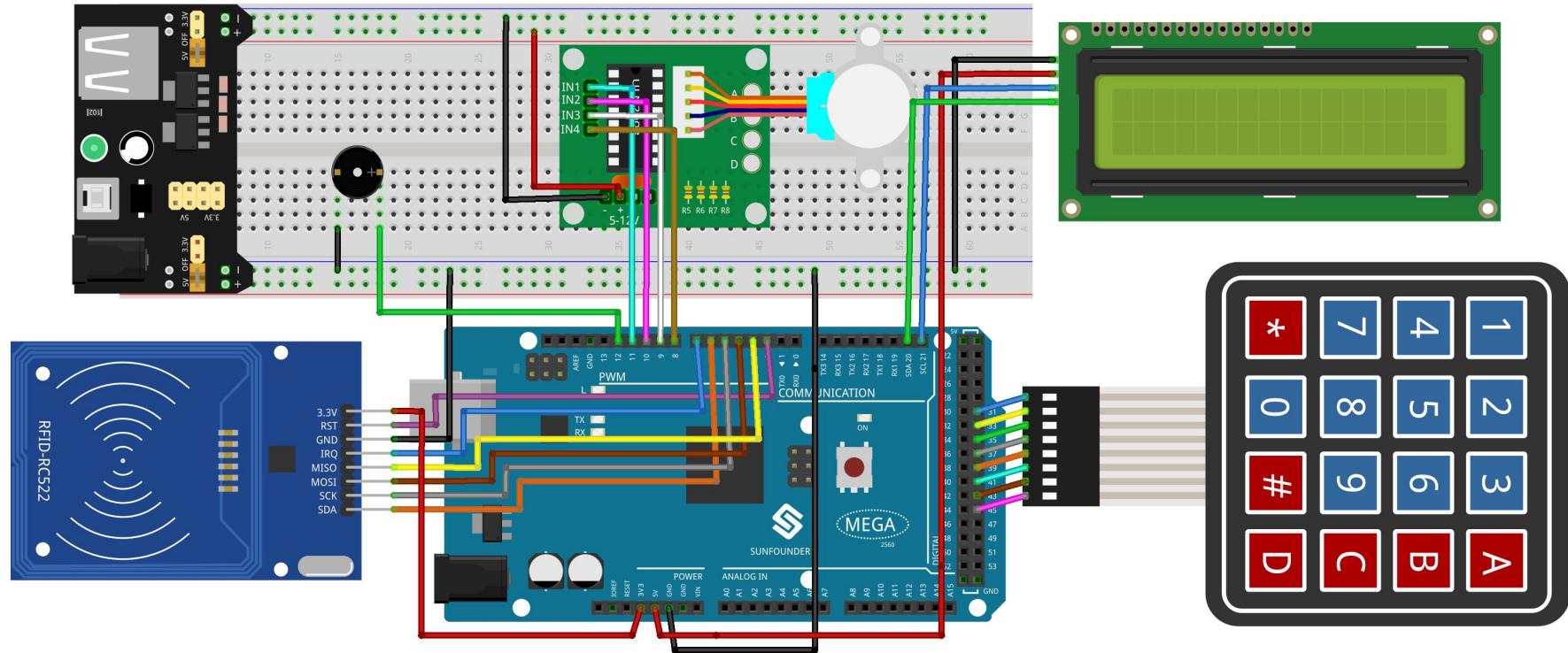


Fritzing Circuit

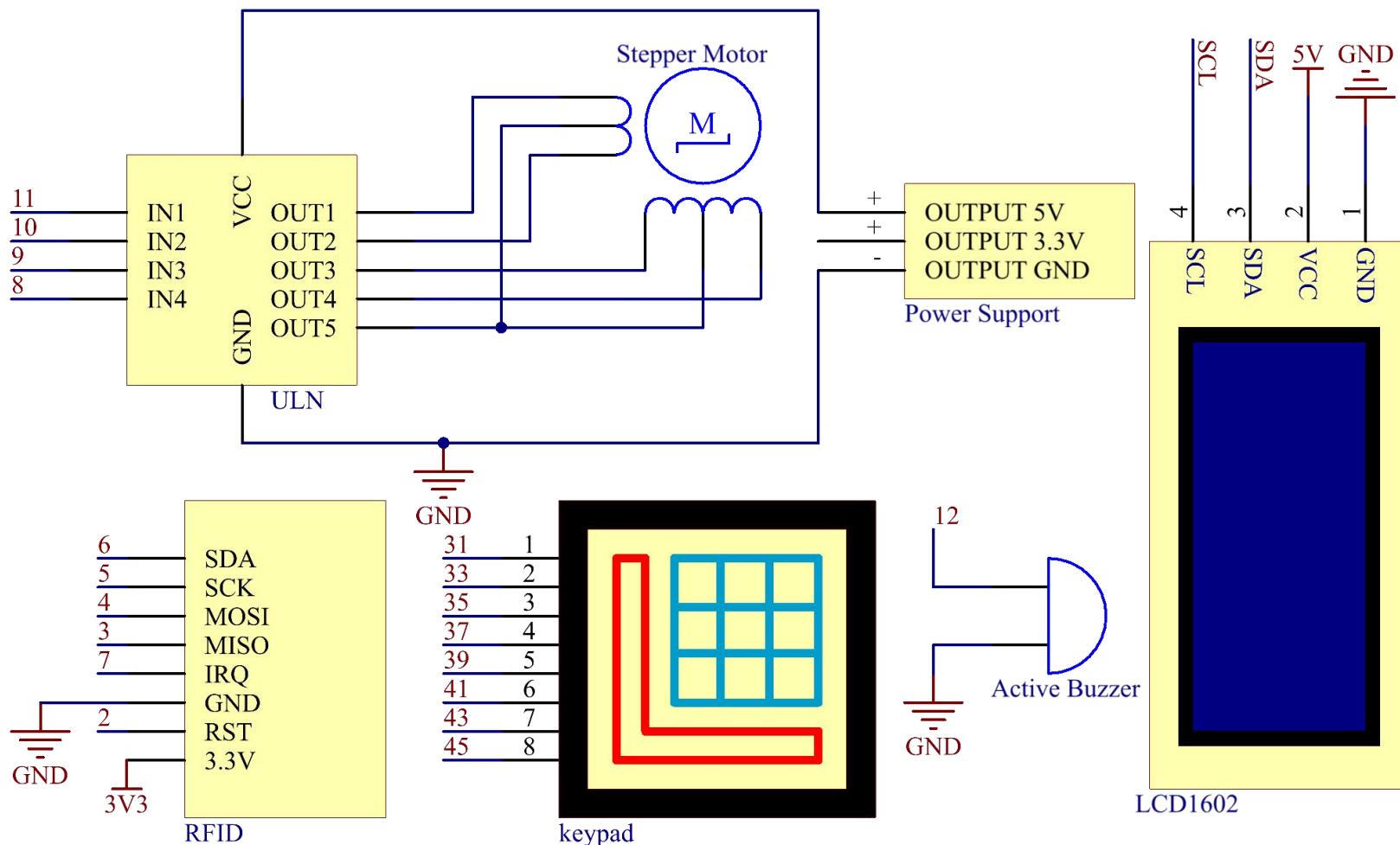
In this example, Power Supply Module is used to power the breadboard. We get the GND of Mega 2560 Board connected to the cathode rail of the breadboard, GND of ULN2003 to the cathode rail of the breadboard, VCC to 5V OUTPUT of Power Supply and the Stepper Motor to OUT1-OUT5 of ULN.

Mega 2560 Board	LCD1602	RFID	Buzzer
GND	GND	GND	(-)
5V	VCC		
SDA	SDA		
SCL	SCL		
3.3V		3.3V	
6		SDA	
5		SCK	
4		MOSI	
3		MISO	
7		IRQ	
2		RST	
12			(+)

Mega 2560 Board	ULN	Keypad
GND	GND	
5V	VCC	
11	IN1	
10	IN2	
9	IN3	
8	IN4	
31		1
33		2
35		3
37		4
39		5
41		6
43		7
45		8



Schematic Diagram



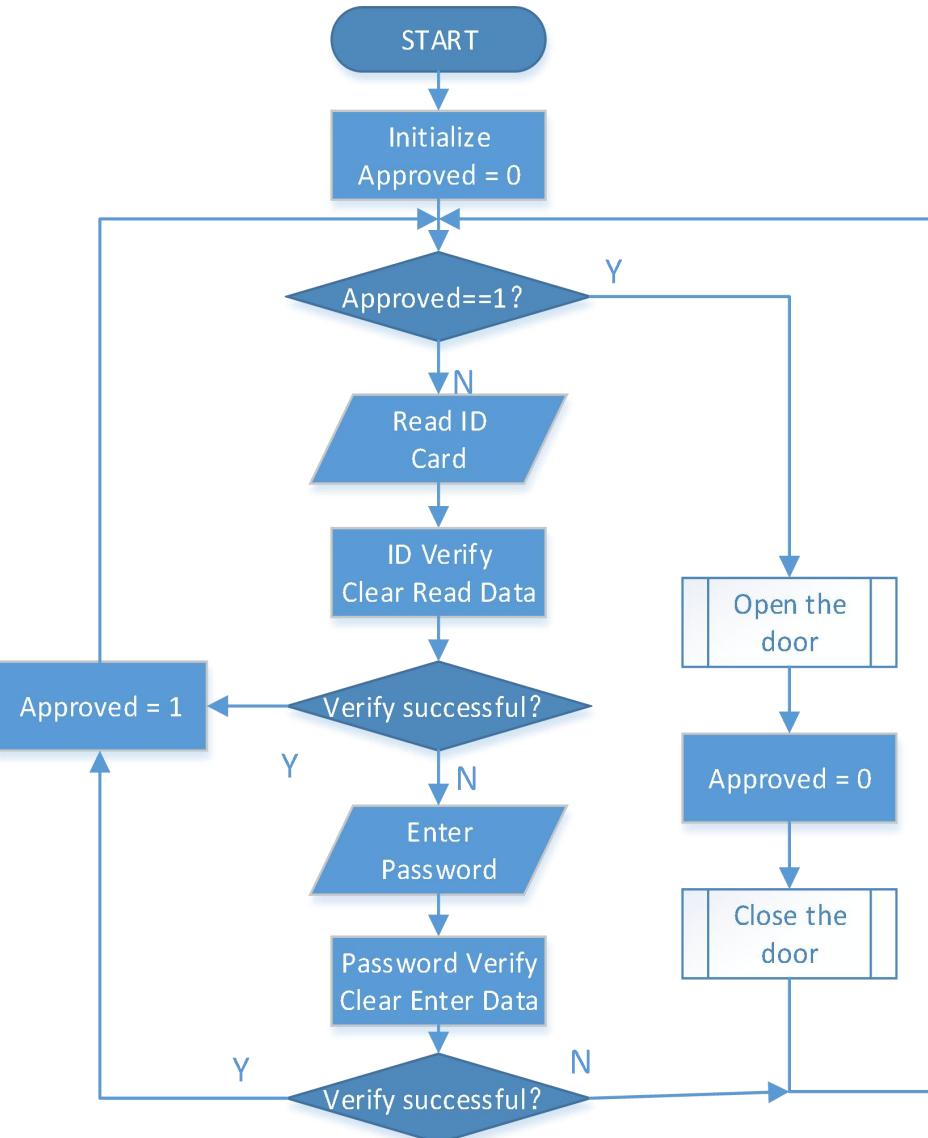
Example Explanation

The workflow of the access control system is shown in the flow chart.

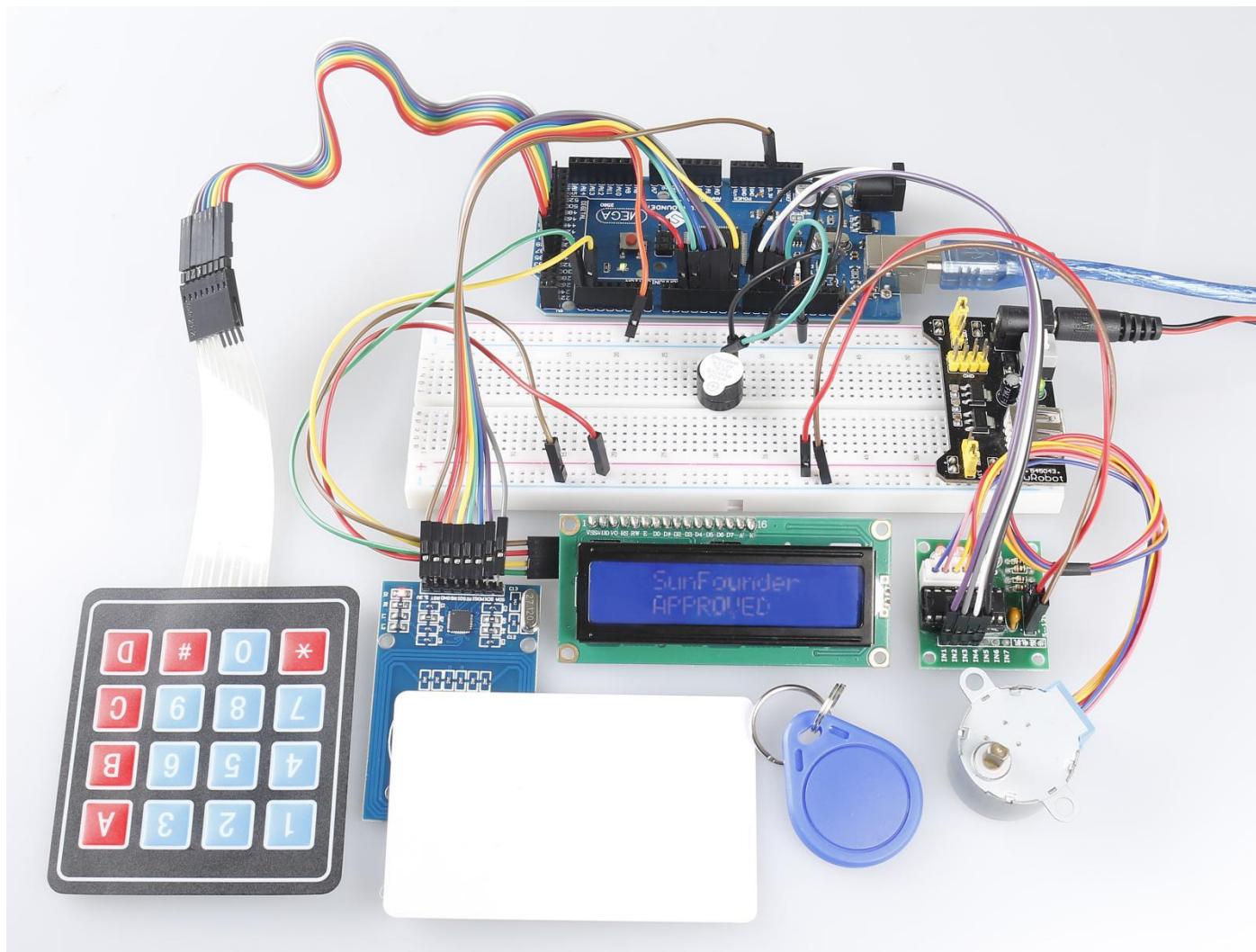
When there is no authorized ID identified (the variable “Approved” equals to 0), the device will perform functions of ID identification and password identification.

If the authorized ID is identified (“Approved” equals to 1), the door will open. After that, the door will be closed a few seconds later and the identified status will be reset (“Approved” equals to 0).

In addition to the core access control function, the project also uses LCD and active buzzer to complete the work of the user interaction system of the access control system.



Phenomenon Picture



Part 4: Appendix

4.1 Add Libraries

What is Library?

A library, gathering some function definitions and header files, usually contains two files: .h (header file, including function statement, Macro definition, constructor definition, etc.) and .cpp (execution file, with function implementation, variable definition, and so on). When you need to use a function in some library, you just need to add a header file (e.g. #include <dht.h>), and then call that function. This can make your code more concise. If you don't want to use the library, you can also write that function definition directly. Though as a result, the code will be long and inconvenient to read.

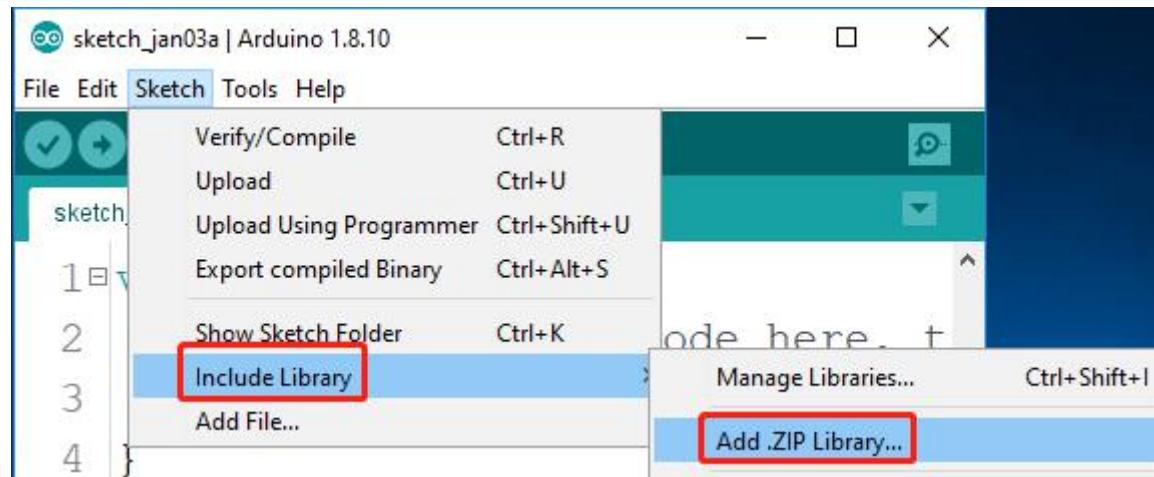
Add libraries

Some libraries are already built in the Arduino IDE, when some others may need to be added. So now let's see how to add one. There are 2 methods for that.

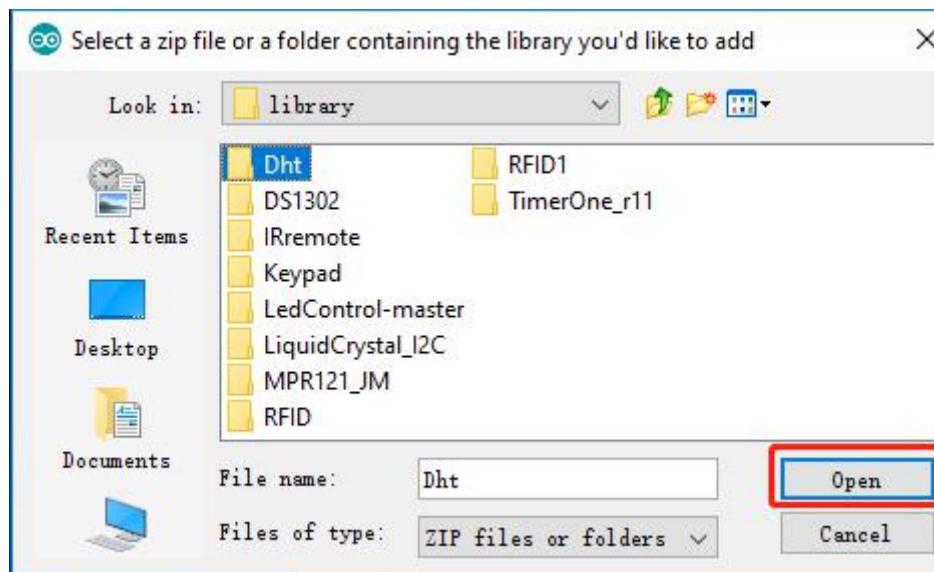
Method 1

Directly import the library in Arduino IDE (take [Dht](#) as an example below). The advantage of this method is easy to understand and operate, but on the other hand, only one library can be imported at a time. So it is inconvenient when you need to add quite a lot of libraries.

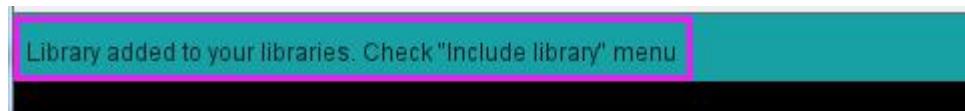
Step 1: Select Sketch -> Include Library -> Add ZIP Library.



Step 2: Find *SunFounder Mega Kit|Library*, Click **Open**.



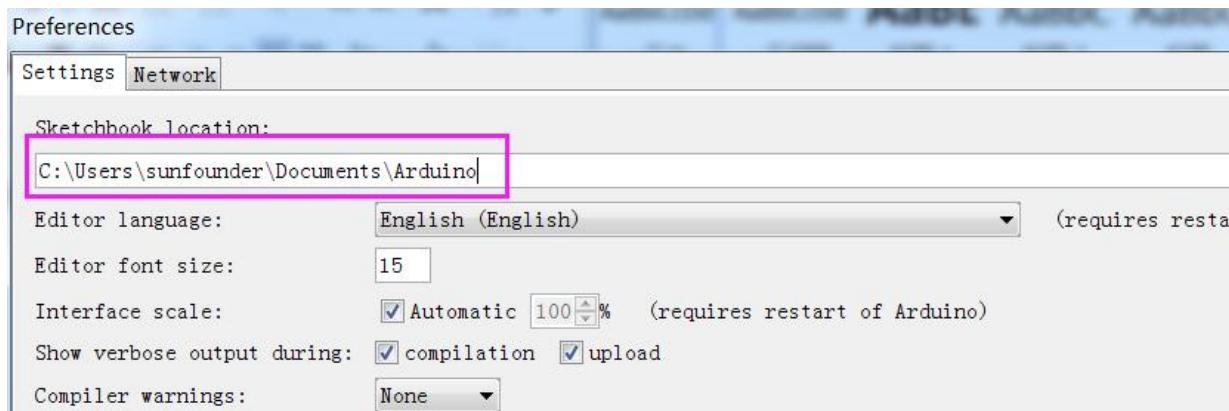
Step 3: When you see “Library added to your libraries. Check “Include library” menu”, it means you have added the library successfully. Please use the same method to add other libraries then.



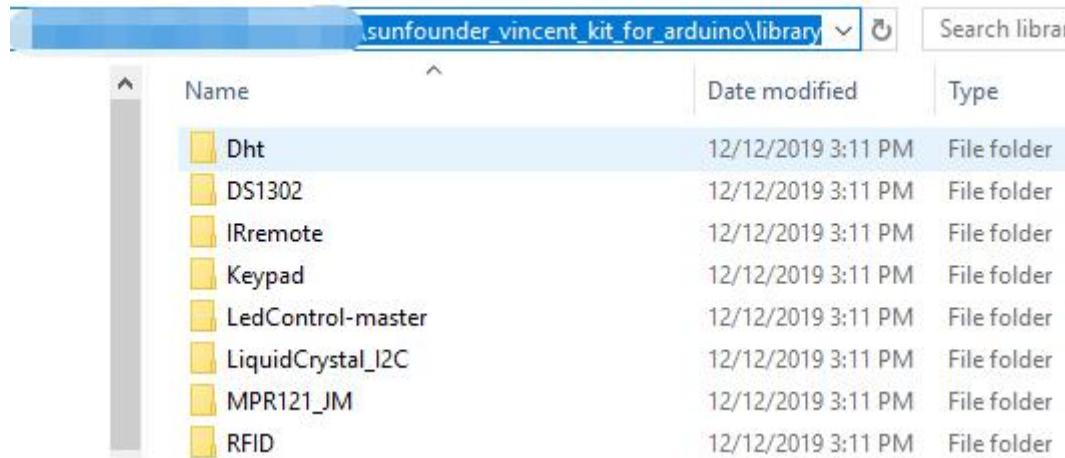
Method 2

Directly copy the library to libraries/Arduino path. This method can copy all libraries and add them at a time, but the drawback is that it is difficult to find libraries/Arduino.

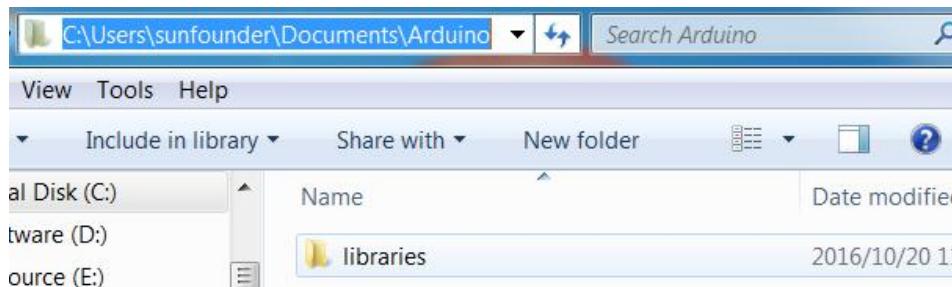
Step 1: Click **File -> Preferences** and on the pop-up window you can see the path of the libraries folder in the text box as shown below.



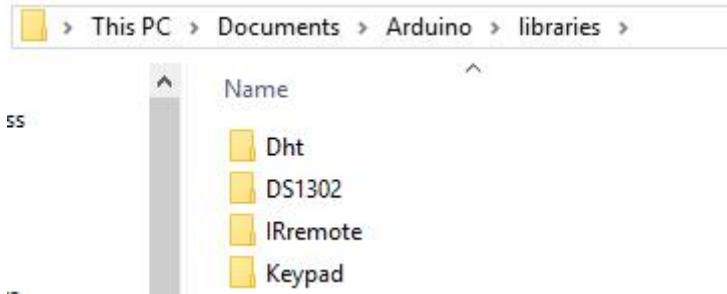
Step 3: Copy all Libraries in the *library* folder.



Step 4: Go to the path above and you will see there is a *libraries* folder, click to open it.



Step 5: Paste all the libraries copied before to the folder. Then you can see them in libraries folder.



Copyright Notice

All contents including but not limited to texts, images, and code in this manual are owned by the SunFounder Company. You should only use it for personal study, investigation, enjoyment, or other non-commercial or nonprofit purposes, under the related regulations and copyrights laws, without infringing the legal rights of the author and relevant right holders. For any individual or organization that uses these for commercial profit without permission, the Company reserves the right to take legal action.