



# FREENOVE

**FREE YOUR INNOVATION**

Freenove is an open-source electronics platform.

[www.freenove.com](http://www.freenove.com)

## Warning

When you purchase or use this product, please note the following:

- This product contains small parts. Swallowing or improper operation them can cause serious infections and death. Seek immediate medical attention when the accident happened.
- Do not allow children under 3 years old to play with or near this product. Please place this product in where children under 3 years of age cannot reach.
- Do not allow children lack of ability of safe to use this product alone without parental care.
- Never use this product and its parts near any AC electrical outlet or other circuits to avoid the potential risk of electric shock.
- Never use this product near any liquid and fire.
- Keep conductive materials away from this product.
- Never store or use this product in any extreme environments such as extreme hot or cold, high humidity and etc.
- Remember to turn off circuits when not in use this product or when left.
- Do not touch any moving and rotating parts of this product while they are operating.
- Some parts of this product may become warm to touch when used in certain circuit designs. This is normal. Improper operation may cause excessively overheating.
- Using this product not in accordance with the specification may cause damage to the product.

## About

Freenove is an open-source electronics platform. Freenove is committed to helping customer quickly realize the creative idea and product prototypes, making it easy to get started for enthusiasts of programing and electronics and launching innovative open source products. Our services include:

- Electronic components and modules
- Learning kits for Arduino
- Learning kits for Raspberry Pi
- Learning kits for Technology
- Robot kits
- Auxiliary tools for creations

Our code and circuit are open source. You can obtain the details and the latest information through visiting the following web sites:

<http://www.freenove.com>

<https://github.com/freenove>

Your comments and suggestions are warmly welcomed, please send them to the following email address:  
[support@freenove.com](mailto:support@freenove.com)

## References

You can download the sketches and references used in this product in the following websites:

<http://www.freenove.com>

<https://github.com/freenove>

If you have any difficulties, you can send email to technical support for help.

The references for this product is named Freenove Ultimate Starter Kit for Arduino, which includes the following folders and files:

- Datasheet      Datasheet for electronic components and modules
- Libraries      Libraries for Arduino sketches
- Processing      Sketches for Interacting with Processing projects
- Sketches      Sketches for Arduino projects
- Readme.txt      Instructions
- Tutorial.pdf      Tutorial for Arduino
- Processing.pdf      Tutorial for Interacting with Processing

## Support

Freenove provides free and quick technical support, including but not limited to:

- Quality problems of products
- Problems in using products
- Questions for learning and technology
- Opinions and suggestions
- Ideas and thoughts

Please send email to:

[support@freenove.com](mailto:support@freenove.com)

On working day, we usually reply to you within 24 hours.

## Copyright

Freenove reserves all rights to this book. No copies or plagiarizations are allowed for the purpose of commercial use.

The code and circuit involved in this product are released as Creative Commons Attribution ShareAlike 3.0. This means you can use them on your own derived works, in part or completely, as long as you also adopt the same license. Freenove brand and Freenove logo are copyright of Freenove Creative Technology Co., Ltd and cannot be used without formal permission.



## Contents

Preface .....	1
Arduino Board.....	1
Arduino Software .....	4
First Use.....	7
Chapter 1 LED Blink.....	11
Project 1.1 Control LED by Manual Button .....	11
Project 1.2 Control LED by Arduino.....	18
Chapter 2 Two LEDs Blink.....	24
Project 2.1 Two LEDs Blink.....	24
Chapter 3 LED bar graph.....	31
Project 3.1 LED bar graph Display.....	31
Chapter 4 LED Blink Smoothly .....	38
Project 4.1 LEDs Emit Different Brightness.....	38
Project 4.2 LED Blink Smoothly .....	43
Chapter 5 Control LED Through Push Button .....	45
Project 5.1 Control LED Through Push Button.....	45
Project 5.2 Change LED State by Push Button.....	49
Chapter 6 Serial.....	51
Project 6.1 Send data through Serial.....	51
Project 6.2 Receive Data through Serial Port .....	56
Project 6.3 Application of Serial.....	60
Chapter 7 ADC .....	63
Project 7.1 ADC.....	63
Project 7.2 Control LED by Potentiometer .....	68
Project 7.3 Control LED through Photoresistor .....	71
Chapter 8 RGB LED .....	75
Project 8.1 Control RGB LED through Potentiometer.....	75
Project 8.2 Colorful LED .....	79
Chapter 9 Buzzer .....	82
Project 9.1 Active Buzzer.....	82
Project 9.2 Passive Buzzer.....	87
Chapter 10 Motor .....	90
Project 10.1 Control Motor by Relay .....	90
Project 10.2 Control Motor by L293D .....	97
Chapter 11 Servo.....	103
Project 11.1 Servo Sweep.....	103
Project 11.2 Control Servo by Potentiometer .....	107
Chapter 12 Temperature Sensor.....	110
Project 12.1 Detect the Temperature .....	110
Chapter 13 Joystick.....	114
Project 13.1 Joystick .....	114

---

Chapter 14 Acceleration sensor .....	118
Project 14.1 Acceleration Detection .....	118
Chapter 15 LED Matrix .....	125
Project 15.1 74HC595.....	125
Project 15.2 LED Matrix.....	131
Chapter 16 LCD1602.....	140
Project 16.1 Display the string on LCD1602 .....	140
Project 16.2 LCD1602 Clock .....	145
Chapter 17 Digital Display.....	154
Project 17.1 1-digit 7-segment Display .....	154
Project 17.2 4-digit 7-segment Display .....	159
Chapter 18 Stepper Motor.....	167
Project 18.1 Drive Stepper Motor.....	167
Chapter 19 Keypad .....	173
Project 19.1 Get Input Characters .....	173
Project 19.2 Combination Lock .....	178
Chapter 20 Vibration Switch.....	183
Project 20.1 Detect Vibration.....	183
Chapter 21 Infrared Remote.....	188
Project 21.1 Infrared Remote Control .....	188
Project 21.2 Control LED through Infrared Remote .....	192
Chapter 22 Temperature & Humidity Sensor.....	196
Project 22.1 Temperature & Humidity Sensor .....	196
Chapter 23 Infrared Motion Sensor .....	201
Project 23.1 Infrared Motion Sensor .....	201
Chapter 24 Ultrasonic Ranging.....	205
Project 24.1 Ultrasonic Ranging .....	205
Chapter 25 Solder Circuit Board .....	213
Project 25.1 Solder a Buzzer .....	213
Project 25.2 Solder a Flowing Water Light .....	217
Other Components.....	222
What's next?.....	224
Appendix.....	225
ASCII table .....	225
Resistor color code .....	226

# Preface

If you want to creat somethings fun or make your great ideas real, using this product will be a good start. Maybe you have heard of Arduino before. If not, it doesn't matter. You can easily create dozens of interesting projects, and gradually realize the fun of using Arduino to complete creative works by reference to this book.

Arduino is an open-source electronics platform based on easy-to-use hardware and software. It's intended for anyone making interactive projects. Arduino has a lot of fans in the world. They contributed a lot of high quality open source code and circuit, so we can use these free code and the circuit to realize our own creativity more efficiently.

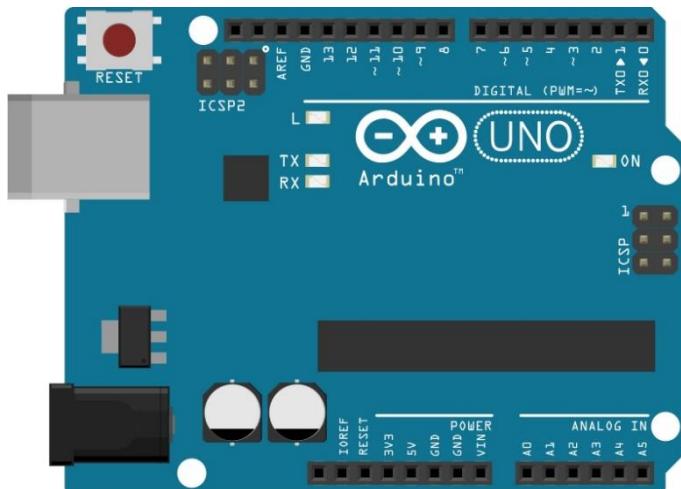
Usually, an Arduino project consists of code and circuit. If you never dealt with code and circuit before, don't worry. Because this book will gradually introduce C programming language and basic knowledge of electronic circuit from easy to difficult. And this product contains all electronic components and modules needed to complete these projects. You can also ask us for fast and free technical support at any time. It is especially suitable for beginners.

## Arduino Board

Arduino Board is a circuit board, which integrates micro controller, input, output interface and etc. Arduino Board can use the sensor to sense the environment and receive user's operation to control LED, motor rotation, etc. We just need to assembly circuit and write the code.

Currently, Arduino Board has several models, and the code between boards of different types is universal (some boards may not be completely compatible because of the differences in hardware). Popular boards include:

UNO





The board used in this book is Freenove UNO, and it is fully compatible to Arduino UNO. Diagram of Freenove UNO board is shown below:



- Digital I/O ports is used to connect to other components or modules, to receive an input signal, or to send a control signal. Usually, we name it by adding a "D" in front of the number, such as D13.
- USB interface is used to provide power, upload code or communicate with PC.
- LED L is connected to digital I/O port 13 (D13).
- LED TX, RX is used to indicate the state of the serial communication.
- DC interface is connected DC power to provide power for the board.
- Power ports can provide power for electronic components and modules.
- Analog I/O ports can be used to measure analog signals.
- LED ON is used to indicate the power state.

Freenove UNO is the most suitable board to complete the experiments of this book. You can also choose to use Arduino UNO, Arduino MICRO, Arduino NANO, Arduino MEGA (or other boards compatible to them).

## Arduino Software

Arduino Software (IDE) is used to write and upload the code for Arduino Board.

First, install Arduino Software (IDE): visit <https://www.arduino.cc>, click "Download" to enter the download page.



Download corresponding installation program according to your operating system. If you are a Windows user, please select the "Windows Installer" to download to install the driver correctly.



After the download completes, run the installer. For Windows users, there may pop up a installation dialog box of driver during the installation process. When it is popped up, please allow the installation.

After installation is complete, an Arduino Software shortcut will be generated in the desktop. Run the Arduino Software.



The interface of Arduino Software is as follows:



---

Programs written using Arduino Software (IDE) are called **sketches**. These sketches are written in the text editor and are saved with the file extension **.ino**. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom righthand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.



Verify

Checks your code for errors compiling it.



Upload

Compiles your code and uploads it to the configured board.



New

Creates a new sketch.



Open

Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content.



Save

Saves your sketch.



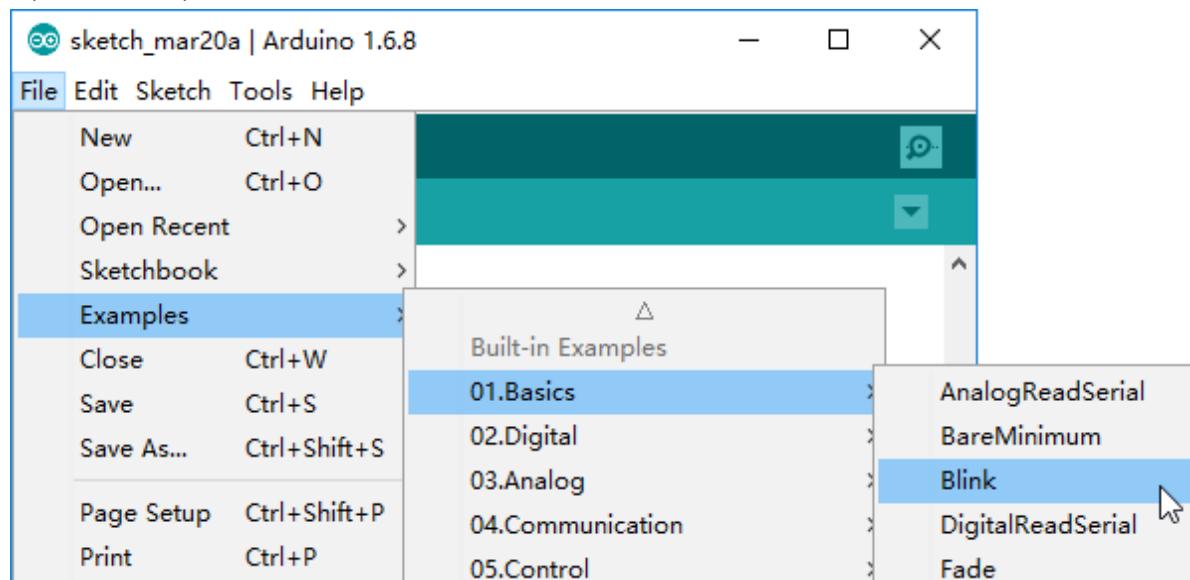
Serial Monitor

Opens the serial monitor.

Additional commands are found within the five menus: File, Edit, Sketch, Tools, Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

## First Use

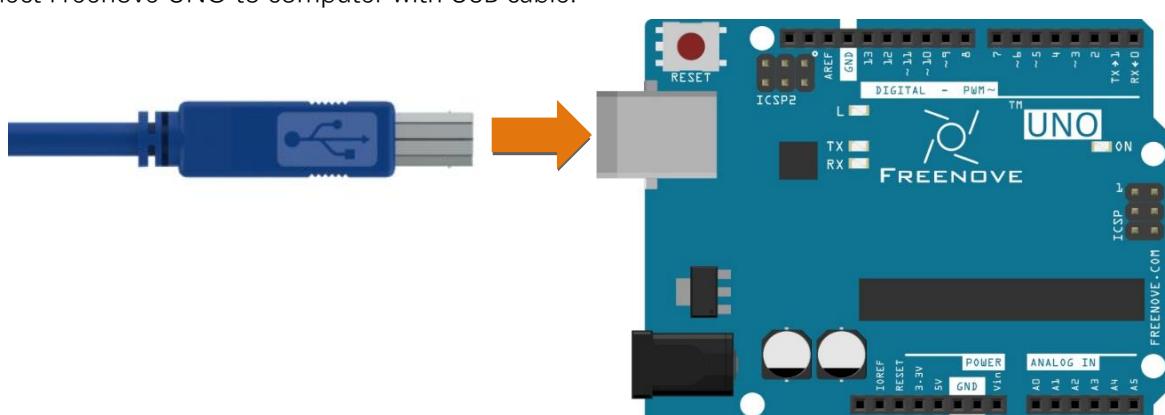
Open the example sketch "Blink" with Arduino Software.



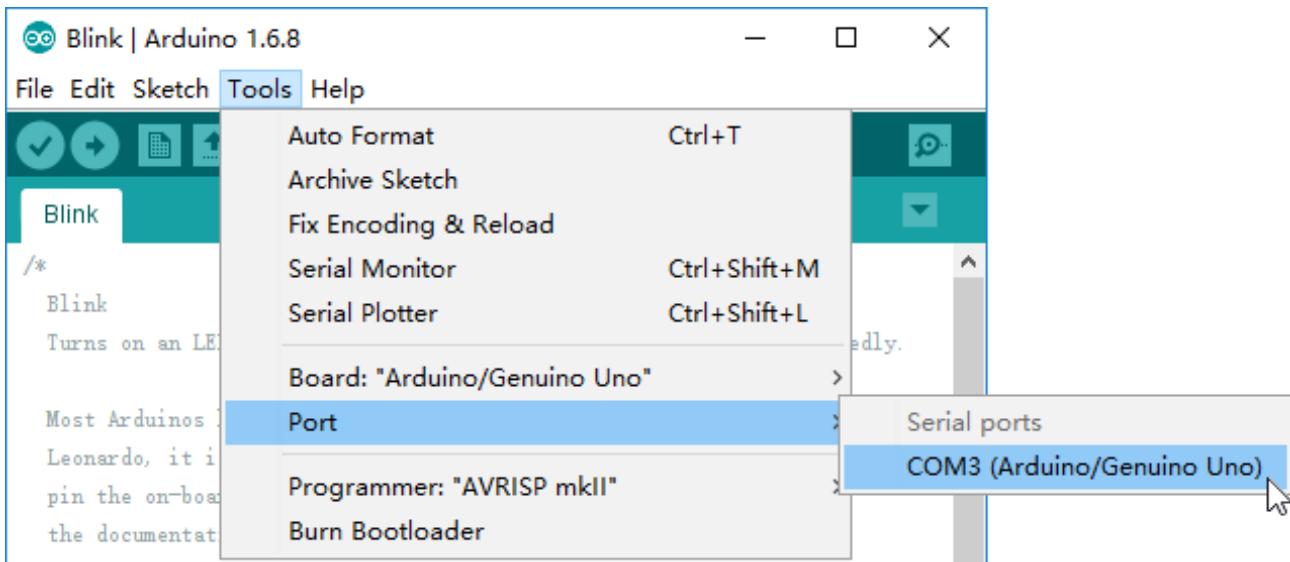
Select board "Arduino/Genuino Uno".



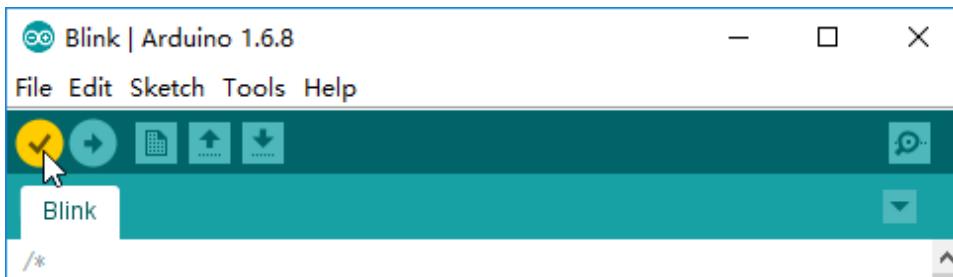
Connect Freenove UNO to computer with USB cable.



Select the serial port. Your serial number may be different from the following figure. If it is not detected immediately, please wait for a while, then click "Tools" to check again.



Click "Verify" button.



The following figure shows the code is being compiled.

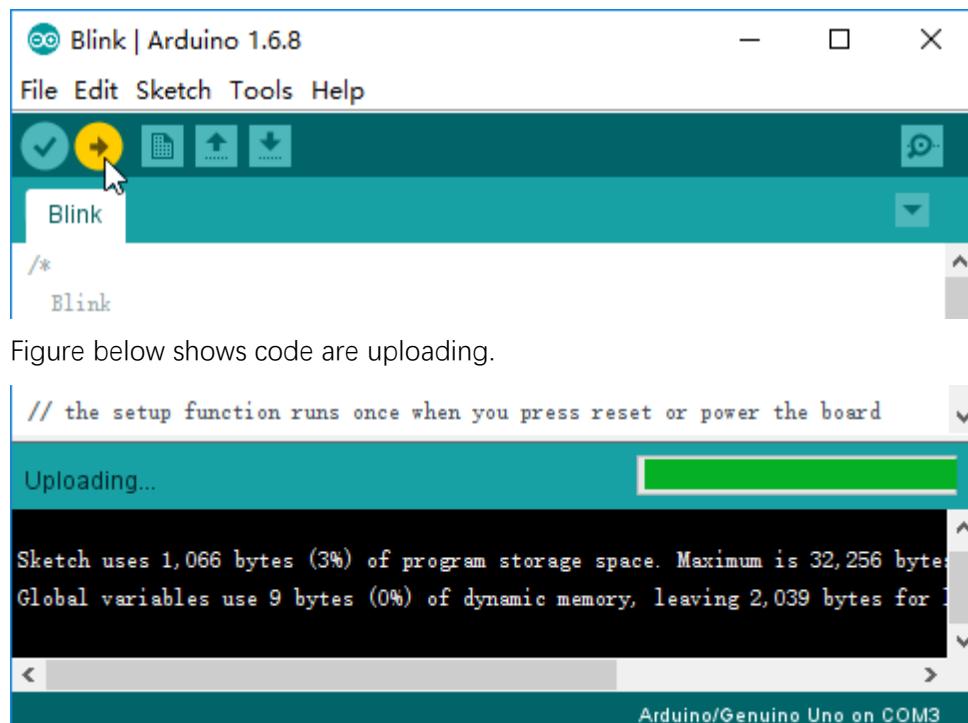


Wait a moment for the compiling to be completed. Figure below shows the code size and percentage of space occupation.

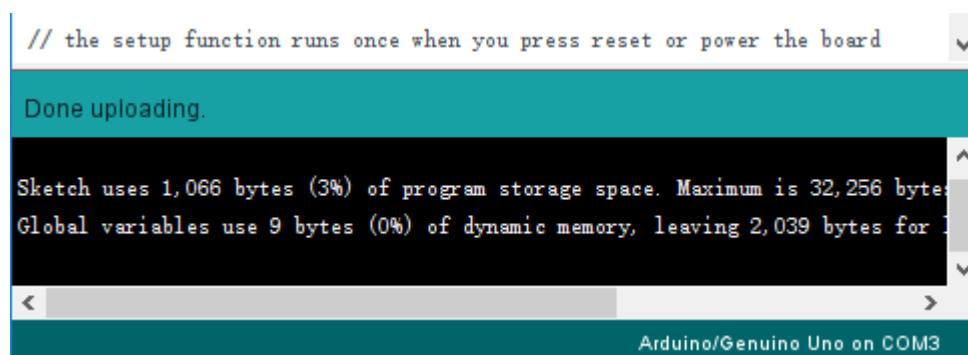


Usually, when we write code, if it has a syntax error, the interface will prompt the error message. Then the compiling can't be completed.

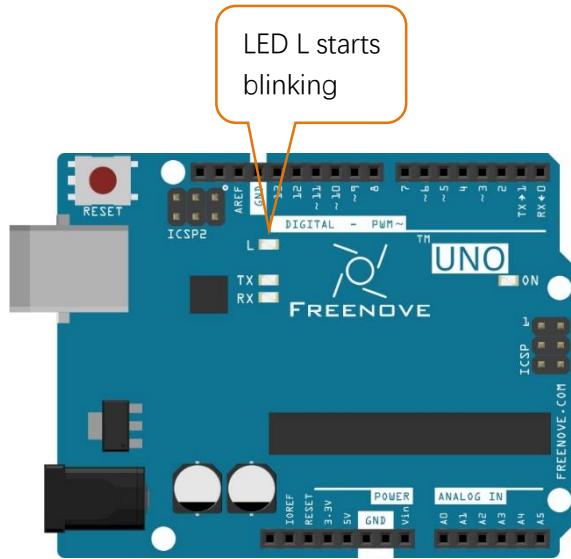
Click "Upload" button.



Wait a moment, then the uploading is completed.



After that, we will see the LED marked with "L" on Freenove UNO starts blinking. It indicates that the code is running now!



So far, we have completed the first use. I believe you have felt the joy of it. Next, we will carry out a series of projects, from easy to difficult, taking you to learn the Arduino programming and the building of electronic circuit.

# Chapter 1 LED Blink

We have previously tried to make the LED marked with "L" blink on Freenove UNO board. Now let us use electronic components and code to reproduce the phenomenon, and try to understand the principle among them.

## Project 1.1 Control LED by Manual Button

First, try using the push button to make the LED blink manually.

### Component list

Freenove UNO x1	Breadboard x1		
			
USB cable x1			
			
Jumper M/M x2	LED x1	Resistor 220Ω x1	Push button x1
			

## Circuit knowledge

### Power supply

Power supply provide energy for the circuit, and it is divided into DC power and AC power.

Voltage and current of DC power supply remains the same over time, such as battery, power adapter.

Voltage and current of AC power supply evolves periodically with time. Its basic form is sinusoidal voltage(current). It is suitable for long-distance transmission of electric energy. And it is used to supply power to homes.



Generally, electronic circuits use DC. Home appliances have rectifiers to convert AC into DC before they use.

Battery or battery pack can be represented by the following symbols:



The positive and negative poles of the power supply can not be directly connected , otherwise it may scald you and cause damage to the battery.

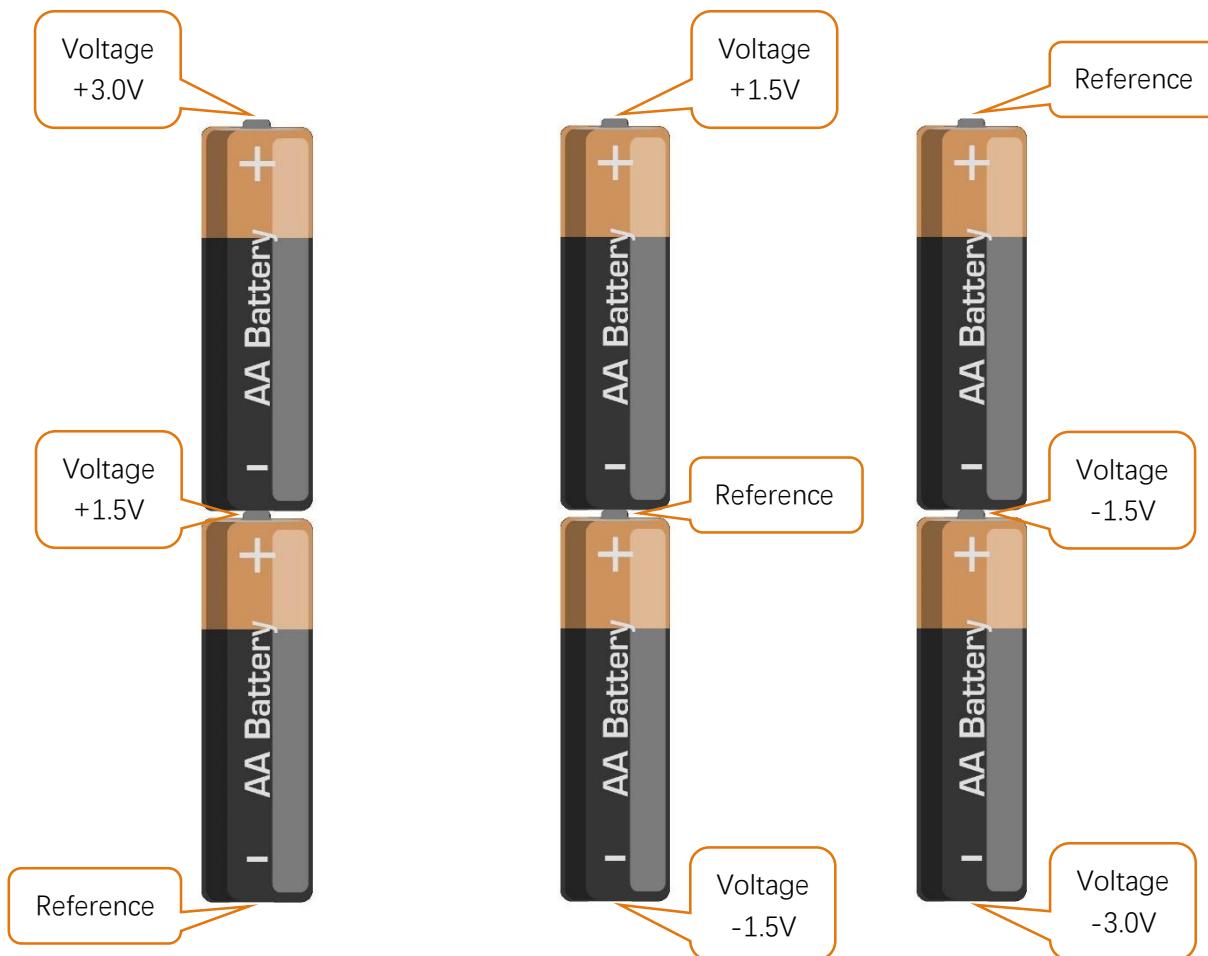
### Voltage

The unit of voltage(U) is volt(V).  $1\text{kV}=1000\text{V}$ ,  $1\text{V}=1000\text{mV}$ ,  $1\text{mV}=1000\mu\text{V}$ .

Voltage is relative. As to a dry battery marked with "1.5V", it's positive (+) voltage is 1.5V higher than the negative (-) voltage. If you specify the negative as reference (0V), the positive voltage will be +1.5V.



When two dry batteries are connected in series, the voltage of each point is as follows:



In practical circuits, we usually specify negative as reference voltage (0V), which is called "Ground". The positive is usually called "VCC". The positive and negative of power supply is usually represented by two following symbols:



## Current

The unit of current( $I$ ) is ampere(A).  $1A=1000mA$ ,  $1mA=1000\mu A$ .

Closed loop consisting of electronic components is necessary for current.

In the figure below: the left is a loop circuit, so current flows through the circuit. The right is not a loop circuit, so there is no current.

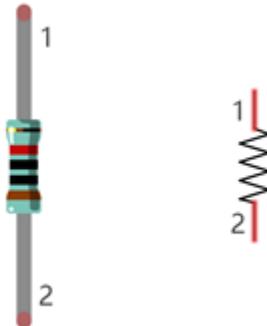


## Resistor

The unit of resistance( $R$ ) is ohm( $\Omega$ ).  $1M\Omega=1000k\Omega$ ,  $1k\Omega=1000\Omega$ .

Resistor is an electrical component that limits or regulates the flow of current in an electronic circuit.

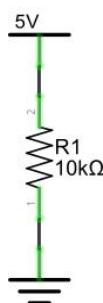
The left is the appearance of resistor. and the right is the symbol of resistor represented in circuit.



Color rings attached to the resistor is used to indicate its resistance. For more details of resistor color code, please refer to the appendix of this book.

With the same voltage there will be less current with more resistance. And the links among current, voltage and resistance can be expressed by the formula below:  $I=U/R$ .

In the following diagram, the current through R1 is:  $I=U/R=5V/10k\Omega=0.0005A=0.5mA$ .



Do not connect the two poles of power supply with low resistance, which will make the current too high to damage electronic components.

## Component knowledge

Let us learn about the basic features of components to use them better.

### Jumper

Jumper is a kind of wire. It is designed to connect the components together with its two terminals by inserting it onto Breadboard or Freenove UNO.

Jumpers have male end (pin) and female end (slot), so jumpers can be divided into the following 3 types.

Jumper M/M



Jumper F/F



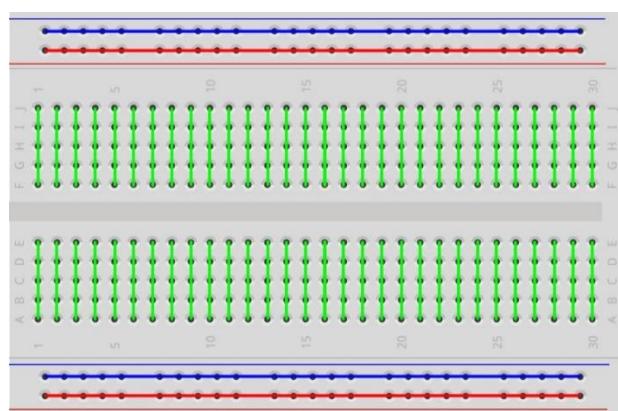
Jumper F/M



### Breadboard

There are many small holes on breadboard to connect Jumper.

Some small holes are connected inside breadboard. The following figure shows the inner links among those holes.



### Push button

Push button has 4 pins. Two pins on the left is connected, and the right is similar as the left, which is shown in the below:



When the push button is pressed, the circuit is turned on.

## LED

LED is a kind of diode. LED will shine only if the long pin of LED is connected to the positive electrode and the short pin is connected to negative electrode.

This is also the features of the common diode. Diode works only if the voltage of its positive electrode is higher than its negative electrode.

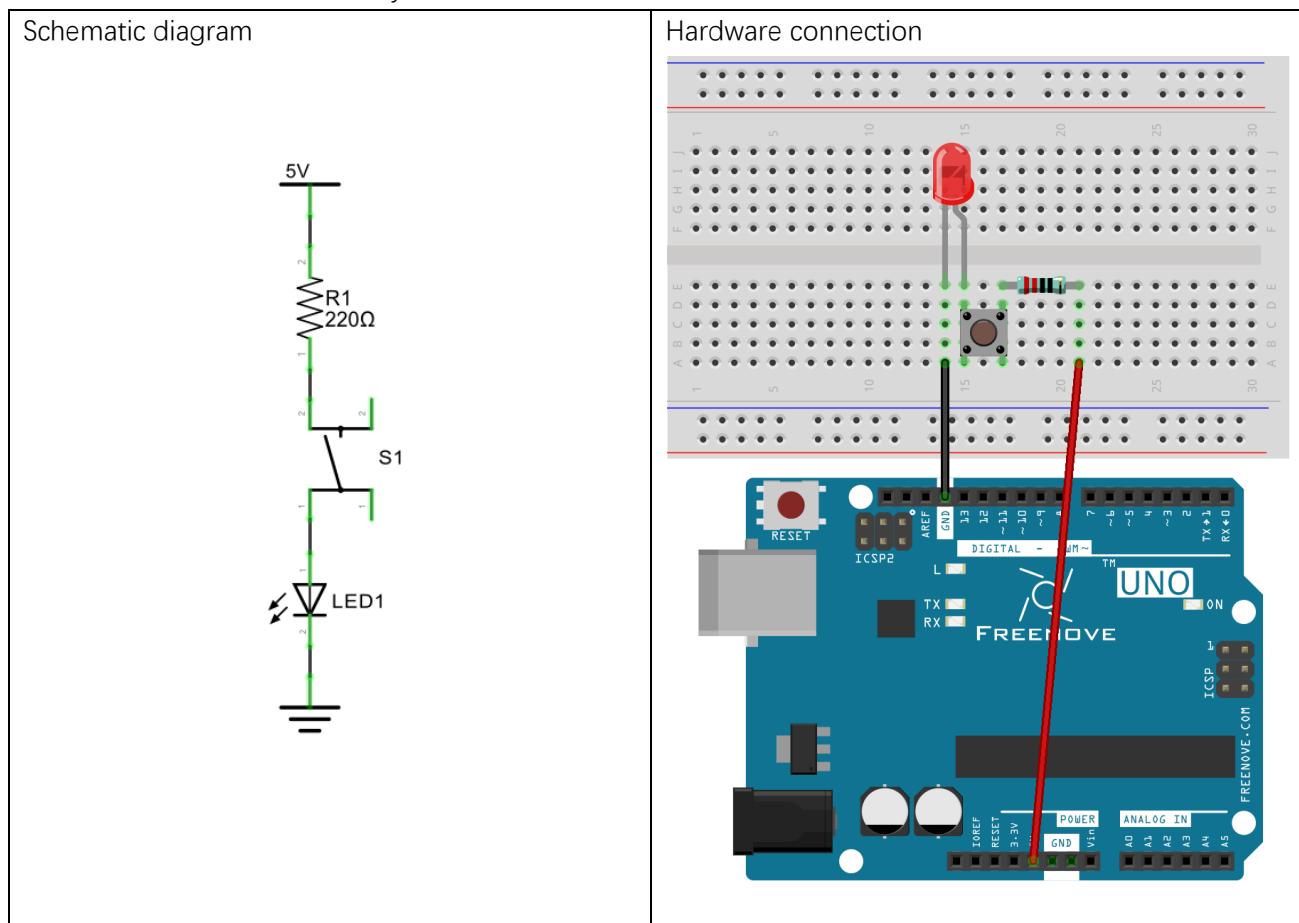


The LED can not be directly connected to power supply, which can damage component. A resistor with certain resistance must be connected in series in the circuit of LED.

## Circuit

In this experiment, the LED is controlled by push button, and Freenove UNO here only plays the role of power supply in the circuit.

Firstly, connect components with jumpers according to "hardware connection". Secondly, check the connection to confirm there is no mistakes. Finally, connect Freenove UNO board to computer with USB cable to avoid that touch on wires may cause short-circuit fault.



LED lights up when you press the push button, and it get off when you release the button.



## Project 1.2 Control LED by Arduino

Now, try using Arduino to make LED blink through programming.

### Component list

Components are basically the same with those in last section. Push button is no more needed.

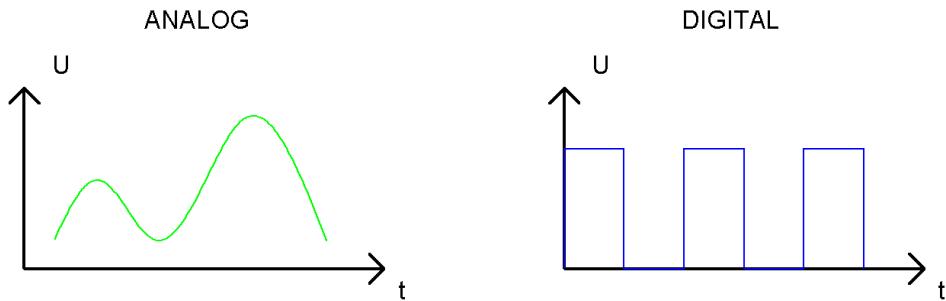
### Circuit knowledge

#### Analog signal and Digital signal

The analog signal is a continuous signal in time and value. On the contrary, digital signal is a discrete signal in time and value.

Most signals in life are analog signals, for example, the temperature in one day is continuously changing, and will not appear a sudden change directly from 0°C to 10°C, while the digital signal is a jump change, which can be directly from 1 to 0.

Their difference can be illustrated by the following figure.



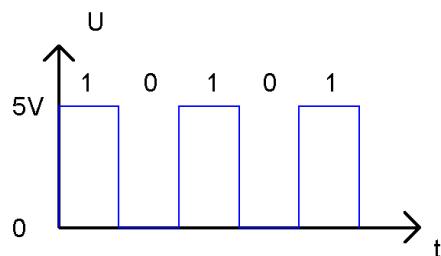
In practical application, the frequently used digital signal is binary signal, that is 0 and 1. The binary signal only has two forms (0 or 1), so it has strong stability. And digital signal and analog signal can be converted to each other.

#### Low level and high level

In circuit, the form of binary (0 and 1) is present as low level and high level.

Low level is generally equal to ground voltage(0V). High level is generally equal to the operating voltage of components.

The low level of Freenove UNO is 0V and high level is 5V, as shown below. When IO port on Freenove UNO outputs high level, components of small power can be directly lit, like LED.



## Code knowledge

Before start writing code, we should learn about the basic programming knowledge.

### Comments

Comments are the words used to explain for the sketches, and won't affect the running of code.

There are two ways to use comments of sketches.

#### 1. Symbol "://"

"// will comment out the content behind it in current line:

```
1 // this is a comment area in this line.
```

The content in front of "//" will not be affected.

```
1 delay(1000); // wait for a second
```

#### 2. Symbol "/\*"and "\*/"

"/" and "/\*" will comment out the content between them:

```
1 /* this is comment area. */
```

"/" and "/\*" can comment out multiple lines.:

```
1 /*
2      this is a comment line.
3      this is a comment line.
4 */
```

### Data type

When programming, we often use digital, characters and other data. C language has several basic data types as follows:

int: A number that does not have a fractional part, an integer, such as 0, 12, -1;

float: A number that has a fractional part, such as 0.1, -1.2;

char: It means character, such as 'a', '@', '0';

For more date types, please visit the website: <https://www.arduino.cc-Learning-Reference-Data Types>.

### Constant

Constant is a kind of data that can not be changed, such as int type 0, 1, float type 0.1, -0.1, char type 'a', 'B'.

### Variable

A variable is a kind of data that can be changed. It consists of a name, a value, and a type. Variables need to be defined before use, such as:

```
1 int i;
```

"int" indicates the type, ";" indicates the end of the statement. The statement is usually written in one single line; and these statements form the code.

After declaration of the variable, you can use it. The following is an assignment to a variable:

```
1 i = 0; // after the execution, the value of i is 0
```

"=" is used to passes the value of a variable or constant on the right side to the variable on the left.

A certain number of variables can be declared in one statement, and a variable can be assigned multiple times. Also, the value of a variable can be passed to other variables. For example:

```

1 int i, j;
2 i = 0;           // after the execution, the value of i is 0
3 i = 1;           // after the execution, the value of i is 1
4 j = i;           // after the execution, the value of j is 1

```

### Function

Function is a collection of statements with a sequence of order, which perform a defined task. Let's define a function void blink() as follows:

```

1 void blink() {
2     digitalWrite(13, HIGH);
3     delay(1000);
4     digitalWrite(13, LOW);
5     delay(1000);
6 }

```

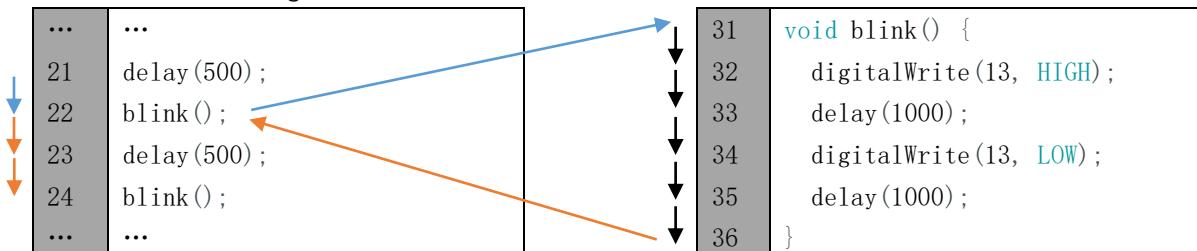
"void" indicates that the function does not return a value (the chapter 4 will detail the return value of functions); "()" its inside is parameters of a function (the chapter 2 will detail the parameters of the function). No content inside it indicates that this function has no parameters;

"{}" contains the entire code of the function.

After the function is defined, it is necessary to be called before it is executed. Let's call the function void blink(), as shown below.

```
1 blink();
```

When the code is executed to a statement calling the function, the function will be executed. After execution of the function is finished, it will go back to the statement and execute the next statement.



Some functions have one or more parameters. When you call such functions, you need to write parameters inside "()":

```

1 digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
2 delay(1000);           // wait for a second

```

## Circuit

Now, let's use IO port of UNO Freenove to provide power for the LED. D13 pin of Freenove UNO is the digital pin. It can output high level or low level. In this way, Freenove UNO can control the state of LED.

Schematic diagram



Hardware connection



## Sketch

### Sketch 1.2.1

In order to make the LED blink, we need to make the 13 pin (D13) of Freenove UNO output high and low level alternately.

We highly recommend you type the code in person instead of copying and pasting, by this method, you can develop your coding skills and get more knowledge.

```

1 // the setup function runs once when you press reset or power the board
2 void setup() {
3     // initialize digital pin 13 as an output
4     pinMode(13, OUTPUT);
5 }
6
7 // the loop function runs over and over again forever
8 void loop() {
9     digitalWrite(13, HIGH);    // turn the LED on (HIGH is the voltage level)
10    delay(1000);           // wait for a second
11    digitalWrite(13, LOW);   // turn the LED off by making the voltage LOW
12    delay(1000);           // wait for a second
13 }
```

The Arduino code usually contain two basic functions: void setup() and void loop().

After Arduino board is **reset**, the setup() function will be executed firstly, then the loop() function will be executed.

setup() function is generally used to write code to initialize the hardware. And loop() function is used to write code to achieve certain functions. loop() function is executed repeatedly. When the execution reaches the end of loop(), it will jump to the beginning of loop() to run again.

### Reset

Reset operation will lead the code to be execute from the beginning. Switching on the power, finishing uploading the code and pressing the reset button will trigger reset operation.



```

Reset
1 // the setup function runs once when you press reset or power the board
2 void setup() {
...
5 }
6
7 // the loop function runs over and over again forever
8 void loop() {
...
13 }
```

In the setup () function, first, we set the 13 pin of UNO board as output mode, which can make the port output high level or low level.

```
3 // initialize digital pin 13 as an output.  
4 pinMode(13, OUTPUT);
```

Then, in the loop () function, set the 13 pin of UNO to output high level to make LED light up.

```
9 digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
```

Wait for 1000ms, that is 1s. Delay () function is used to make control board wait for a moment before executing the next statement. The parameter indicates the number of milliseconds to wait for.

```
10 delay(1000); // wait for a second
```

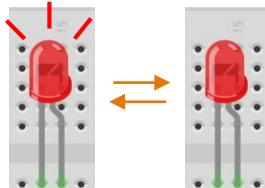
Then set the 13 pin to output low level, and LED light off. One second later, the execution of loop () function will be completed.

```
11 digitalWrite(13, LOW); // turn the LED off by making the voltage LOW  
12 delay(1000); // wait for a second
```

The loop() function is constantly being executed, so LED will keep blinking.

The function called above is standard function of the Arduino, which has been defined in the Arduino Software. Those functions can be called directly. We will introduce more common standard functions in later chapters. For more standard functions and the specific use method, please visit <https://www.arduino.cc-Learning-Reference-Functions>.

Verify and upload the code, then the LED starts blinking.



# Chapter 2 Two LEDs Blink

In last chapter, we have already written code to make 1 LED blink on Freenove UNO board. And now, we will try to make 2 LEDs blink for further programming study.

## Project 2.1 Two LEDs Blink

Now, try to make two LEDs blink through Freenove UNO board.

### Component list

Freenove UNO x1	Breadboard x1
	
USB cable x1	LED x2
	
Jumper M/M x3	Resistor 220Ω x2
	

## Code knowledge

In the last chapter, we have simple understanding of programming. Now let's learn more about the basic programming knowledge.

### Parameters of function

In the last chapter, we have used a function with a parameter, such as:

```
1 delay(1000); // wait for a second
```

Next, we will define a function with a parameter as below:

```
1 void functionA(int i) {
2     i = i + 1;
3 }
```

"i" is the parameter of this function. "int" is the type of i. When calling this function, it is necessary to enter the parameter of int type:

```
1 functionA(1);
```

The input parameter will be assigned to "i" and involved in the calculation of the functionA(int i):



A function can define more than one parameter and the type of the parameters can be different:

```
1 void functionB(int i, char j) {
2     char k = 'a';
3     i = i + 1;
4     k = j;
5 }
```

### Boolean data type

Date of Boolean type can only be assigned to "true" or "false".

"true" generally represents a certain relationship which is tenable and correct, and "false" is the opposite.

```
1 boolean isTrue;
2 isTrue = true; // after the execution, "isTrue" is assigned to true.
3 isTrue = false; // after the execution, "isTrue" is assigned to false.
```

In the code, the number 0 can be considered to be false, and nonzero numbers can be considered true.

### Logical operator

The logic operators have "&&" (and), "||" (or), "!" (non), and the calculation object of them are boolean type. The result of logic operation is as follows:

&&	true	false
true	true	false
false	false	false

	true	false
true	true	true
false	false	false

!	!
true	false
false	true

For example :

```

1 boolean isTrue;
2 isTrue = true && false; // after the execution, "isTrue" is assigned to false.
3 isTrue = true || false; // after the execution, "isTrue" is assigned to true.
4 isTrue = !true;        // after the execution, "isTrue" is assigned to false.

```

### Relation operator

Relational operator is used to judge whether the relationship of the two amount is tenable and correct. If the relationship is tenable, the result is true. Otherwise, the result is false.

For example, the results of "1>2" is true and the result of "1<2" is false:

```

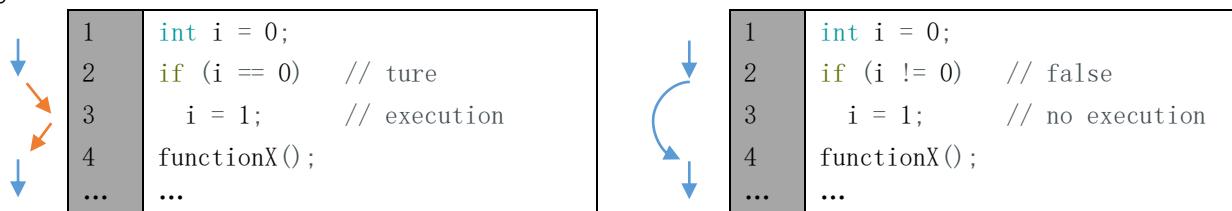
1 boolean isTrue;
2 isTrue = 1 < 2;           // after the execution, "isTrue" is true.
3 isTrue = 1 > 2;           // after the execution, "isTrue" is false.

```

There are other relational operators such as "==" (equal to), ">=" (greater than or equal to), "<=" (less than or equal to) and "!=" (not equal to).

### Conditional statement

Conditional statements are used to decide whether or not to execute the program based on the result of judgment statement.



When there are many statements needed to be executed, we can put them into "{}":



Only the section of code in which conditions are tenable will be executed:



In addition, it can judge multiple condition.



```

1 int i = 1;
2 if (i == 0)      // false
3     i = 1;        // no execution
4 else if (i == 1)// ture
5     i = 2;        // execution
6 else if (i == 2)
7     i = 3;        // no execution
8 else
9     i = 4;        // no execution
10 functionX();
...

```



```

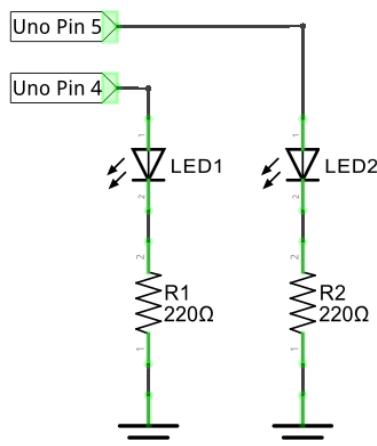
1 int i = 3;
2 if (i == 0)      // false
3     i = 1;        // no execution
4 else if (i == 1)// false
5     i = 2;        // no execution
6 else if (i == 2)// false
7     i = 3;        // no execution
8 else
9     i = 4;        // execution
10 functionX();
...

```

## Circuit

Use D4 and D5 pin of Freenove UNO to drive these two LEDs respectively.

Schematic diagram



Hardware connection



## Sketch

In order to show the difference between use of function and no use of function, we will write two different sketches to make two LEDs blink.

### Sketch 2.1.1

At first, use sketch without function to make two LEDs blink alternatively.

```

1 // set pin numbers:
2 int led1Pin = 4;           // the number of the LED1 pin
3 int led2Pin = 5;           // the number of the LED2 pin
4
5 void setup() {
6     // initialize the LED pin as an output:
7     pinMode(led1Pin, OUTPUT);
8     pinMode(led2Pin, OUTPUT);
9 }
10
11 void loop() {
12     digitalWrite(led1Pin, HIGH);    // turn the LED1 on
13     digitalWrite(led2Pin, LOW);    // turn the LED2 off
14     delay(1000);                 // wait for a second
15
16     digitalWrite(led1Pin, LOW);    // turn the LED1 off
17     digitalWrite(led2Pin, HIGH);   // turn the LED2 on
18     delay(1000);                 // wait for a second
19 }
```

This section of code is similar with the last section. The difference is the amount of LEDs are two. And two LED blink alternatively.

#### Variable scope

In the 2,3 rows of code above, we define two variables to store the pin number. These two variables defined outside of the function are called "Global variable", which can be called by all other functions. Variables defined within a function is called "local variable", which can be called only by the current function. When local variables and global variables have same names, the variable names represents local variable in the current function.

Verify and upload the code, then you will see the two LEDs blink alternatively.



### Sketch 2.1.2

In the last sketch, we can see that the following two sections of the code are similar, so we will use one function to replace them to simplify code.

```
12  digitalWrite(led1Pin, HIGH); // turn the LED1 on
13  digitalWrite(led2Pin, LOW); // turn the LED2 off
14  delay(1000); // wait for a second
```

```
16  digitalWrite(led1Pin, LOW); // turn the LED1 off
17  digitalWrite(led2Pin, HIGH); // turn the LED2 on
18  delay(1000); // wait for a second
```

Now, we will use the function to improve the above code.

```
1 // set pin numbers:
2 int led1Pin = 4; // the number of the LED1 pin
3 int led2Pin = 5; // the number of the LED2 pin
4
5 void setup() {
6     // initialize the LED pin as an output:
7     pinMode(led1Pin, OUTPUT);
8     pinMode(led2Pin, OUTPUT);
9 }
10
11 void loop() {
12     setLed(HIGH, LOW); // set LED1 on, and LED2 off.
13     setLed(LOW, HIGH); // set LED1 off, and LED2 on.
14 }
15
16 void setLed(int led1, int led2) {
17     digitalWrite(led1Pin, led1); // the state of LED1 is determined by variable led1.
18     digitalWrite(led2Pin, led2); // the state of LED2 is determined by variable led2.
19     delay(1000); // wait for a second
20 }
```

In the sketch above, we integrate the 2 LED statements into one function void setLed(int led1, int led2), and control two LEDs through the parameters led1 and led2.

```
16 void setLed(int led1, int led2) {
17     digitalWrite(led1Pin, led1); // the state of LED1 is determined by variable led1.
18     digitalWrite(led2Pin, led2); // the state of LED2 is determined by variable led2.
19     delay(1000); // wait for a second
20 }
```

When the function above is called, we will control the two LEDs by using different parameters as below.

```
12 setLed(HIGH, LOW); // set LED1 on, and LED2 off.
13 setLed(LOW, HIGH); // set LED1 off, and LED2 on.
```

Verify and upload the code, then you will see the two LEDs blink alternatively.

**HIGH and LOW**

The macro is an identifier that represents a number, a statement, or a piece of code. HIGH and LOW are two macros. HIGH and LOW is defined in Arduino Software as below:

```
#define HIGH 1  
#define LOW 0
```

In the code, a macro is used as the content defined by itself. For example, setLed (HIGH, LOW) is equivalent to setLed (1, 0).

Using macros can enhance the readability of code and simplify code, such as INPUT, OUTPUT.

**Sketch 2.1.3**

In the last section of code, we used a function that integrates two similar paragraph of code. And We control two LEDs by using two parameters. As the two LEDs are always blinking alternatively, so let's try to use one parameter to control these two LEDs, which is achieved by conditional statements.

Now, we'll use conditional statement to improve the code above.

```
1 // set pin numbers:  
2 int led1Pin = 4;           // the number of the LED1 pin  
3 int led2Pin = 5;           // the number of the LED2 pin  
4  
5 void setup() {  
6     // initialize the LED pin as an output:  
7     pinMode(led1Pin, OUTPUT);  
8     pinMode(led2Pin, OUTPUT);  
9 }  
10  
11 void loop() {  
12     setLed1(HIGH);        // set LED1 on, and LED2 off.  
13     setLed1(LOW);         // set LED1 off, and LED2 on.  
14 }  
15  
16 void setLed1(int led1) {  
17     digitalWrite(led1Pin, led1); // the state of LED1 is determined by variable led1.  
18  
19     if (led1 == HIGH)       // the state of LED2 is determined by variable led1.  
20         digitalWrite(led2Pin, LOW); // if LED1 is turned on, LED2 will be turned off.  
21     else  
22         digitalWrite(led2Pin, HIGH); // if LED1 is turned off, LED2 will be turned on.  
23  
24     delay(1000);           // wait for a second  
25 }
```

Here, we rewrite the function so that we only need to set the state of LED1. And the state of LED2 can be set automatically.

Verify and upload the code, then two LEDs blink alternatively.

# Chapter 3 LED bar graph

We have learned previously how to control 1 or 2 LED through Sketch on Freenove UNO board and learn some basic knowledge of programming. Now let's try to control 10 LEDs and learn how to simplify the code.

## Project 3.1 LED bar graph Display

Let us use Freenove UNO to control a bar graph LED consist of 10 LEDs.

### Component list

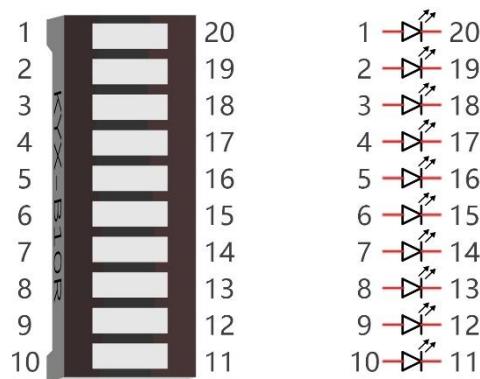
Freenove UNO x1	Breadboard x1
	
USB cable x1	LED bar graph x1
	
Jumper M/M x11	Resistor 220Ω x10
	

## Component knowledge

Let us learn about the basic features of components to use them better.

### LED bar graph

LED bar graph is a component Integration consist of 10 LEDs. There are two rows of pins at its bottom. At the bottom of the LED bar graph, there are two rows of pins, corresponding to positive and negative pole separately. If the LED bar graph can not work in the circuit, it was probably because the connection between positive and negative pole is wrong. Please try to reverse the LED bar graph connection.



## Code knowledge

This section will use new code knowledge.

### Array

Array is used to record a set of variables. An array is defined below:

```
1 int a[10];
```

"int" is the type of the array and "10" is an element of the array. This array can store 10 int types of elements as below.

```
1 int a[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
```

Or there is another form that the number of elements is the size of the array:

```
1 int a[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
```

We can reference elements of an array as below :

```
1 int i, j;
2 i = a[0];
3 j = a[1];
4 a[0] = 0;
```

Among them, "[]" is the array index, with a[0] as the first elements in the array.

For example, now we define an arry b[] below:

```
1 int b[] = {5, 6, 7, 8};
```

The value of each element in array b[] is as follows:

b[0]	b[1]	b[2]	b[3]
5	6	7	8

This is just the use of one-dimensional array. And there are two-dimensional arrays, three-dimensional arrays, and multi-dimensional arrays. Readers interested of this part can develop your own learning.

## Loop

The loop statement is used to perform repetitive work such as the initialization to all the elements of an array.

```
1 while(expression)
2     functionX();
```

When there is more than one statement to be executed, the form is as follows:

```
1 while(expression) {
2     functionX();
3     functionY();
4 }
```

The first step of the execution is judgment expression inside "()". If the result is false, the statements inside "{}" will not be executed; if result is true, the statements will be executed.

```
1 int i = 0;
2 while (i < 2)
3     i = i + 1;
4 i = 5;
```

First time: i<2, i=0 is tenable, execute i=i+1, then i=1;

Second time: i<2, i=1 is tenable, execute i=i+1, then i=2;

Third time: i<2, i=2 is not tenable, execution of loop statements is completed. Statement i=5 will be executed next.

"do while" and "while" is similar. The difference is that the loop statements of "do while" is executed before judging expression. The result of the judgment will decide whether or not go on the next execution:

```
1 do {
2     functionX();
3 } while (expression);
```

"for" is another loop satement, and its form is as follows:

```
1 for (expression1; expression2; expression 3)
2     functionX();
```

When there is more than one statement to be executed, the form is as follows:

```
1 for (expression 1; expression 2; expression 3) {
2     functionX();
3     functionY();
4 }
```

Expression 1 is generally used to initialize variables; expression 2 is judgement which is used to decide wether or not to execute loop statements; the expression 3 is generally used to change the value of variables.

For example:

```

1 int i = 0, j = 0;
2 for (i = 0; i < 2; i++)
3     j++;
4 i = 5;

```

First:  $i=0$ ,  $i < 2$  is tenable, execute  $j++$ , and execute  $i++$ , then  $i=1$ ,  $j=1$ ;

Second:  $i=1$ ,  $i < 2$  is tenable, execute  $j++$ , and execute  $i++$ , then  $i=2$ ,  $j=2$ ;

Third times:  $i < 2$  is tenable,  $i=2$  is not tenable. The execution of loop statements is completed. Statement  $i=5$  will be executed next.

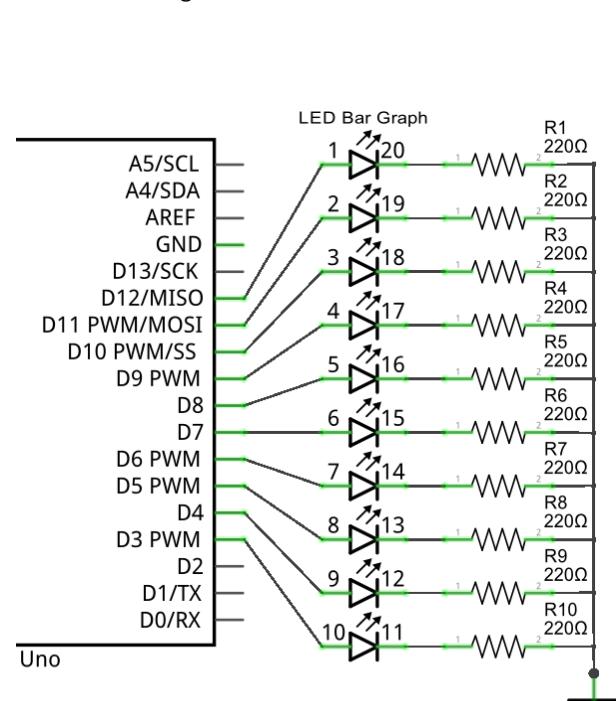
### Operator $++$ , $--$

" $i++$ " is equivalent to " $i=i+1$ ". And " $i--$ " equivalent to " $i=i-1$ ".

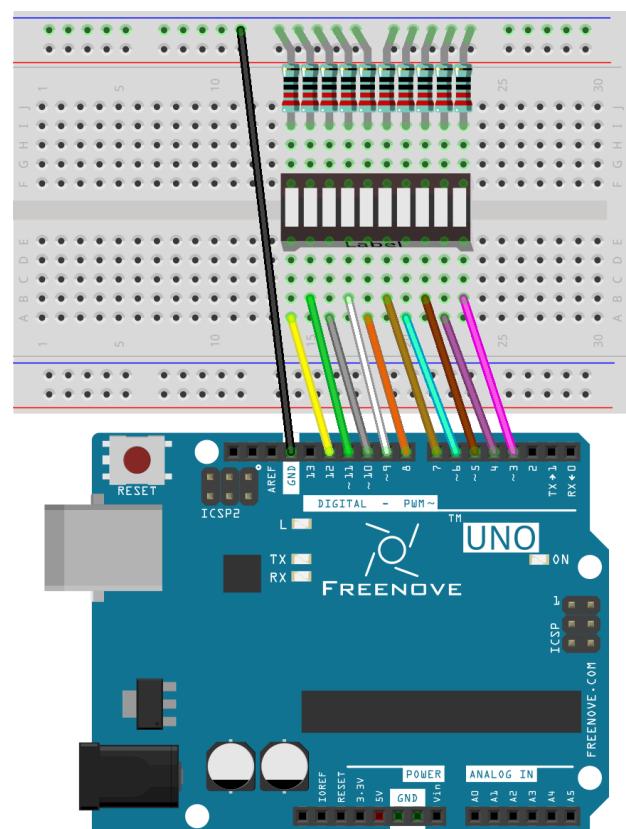
## Circuit

Let us use D3, D4, D5, D6, D7, D8, D9, D10, D11, D12 pin of Freenove UNO board to drive LED bar graph.

Schematic diagram



Hardware connection



## Sketch

Now let us complete the sketch to control LED bar graph.

### Sketch 3.1.1

First, write a sketch which can achieve the LED light water.

```
1 const int ledCount = 10;      // the number of LEDs in the bar graph
2
3 // an array of pin numbers to which LEDs are attached
4 int ledPins[] = { 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };
5
6 void setup() {
7     // loop over the pin array and set them all to output:
8     for (int i = 0; i < ledCount; i++) {
9         pinMode(ledPins[i], OUTPUT);
10    }
11 }
12
13 void loop() {
14     // the ith LED of LED bar graph will light up in turn
15     for (int i = 0; i < ledCount; i++) {
16         barGraphDisplay(i);
17     }
18 }
19
20 void barGraphDisplay(int ledOn) {
21     // make the "ledOn"th LED of bar graph LED on and the others off
22     for (int i = 0; i < ledCount; i++) {
23         if (i == ledOn)
24             digitalWrite(ledPins[i], HIGH);
25         else
26             digitalWrite(ledPins[i], LOW);
27     }
28     delay(100);
29 }
```

Firstly, let us define a read-only variable to record the number of LEDs as the number of times in the loop.

```
1 const int ledCount = 10;      // the number of LEDs in the bar graph
```

#### Read-only variable

"const" keyword is used to define read-only variables, which is called in the same way with common variable. But read-only variables can be assigned only once.

Then we define an array used to store the number of pins connected to LED bar graph. So it is convenient to manipulate arrays to modify the pin number.

```
3 // an array of pin numbers to which LEDs are attached
4 int ledPins[] = { 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };
```

Use loop statement to set the pins to output mode in function setup().

```
6 void setup() {
7     // loop over the pin array and set them all to output:
8     for (int i = 0; i < ledCount; i++) {
9         pinMode(ledPins[i], OUTPUT);
10    }
11 }
```

Define a function to open a certain LED on the LED bar graph and close the other LEDs.

```
20 void barGraphDisplay(int ledOn) {
21     // make the "ledOn"th LED of LED bar graph on and the others off
22     for (int i = 0; i < ledCount; i++) {
23         if (i == ledOn)
24             digitalWrite(ledPins[i], HIGH);
25         else
26             digitalWrite(ledPins[i], LOW);
27     }
28     delay(100);
29 }
```

Finally, when the above function is called cyclically, there will be the formation of LED water effect in LED bar graph.

```
13 void loop() {
14     // make the "i"th LED of LED bar graph on and the others off in turn
15     for (int i = 0; i < ledCount; i++) {
16         barGraphDisplay(i);
17     }
18 }
```

Verify and upload the code, then you will see the light water on the LED bar graph.



### Sketch 3.1.2

Then modify the code to create a reciprocating LED light water.

```
1 const int ledCount = 10;      // the number of LEDs in the bar graph
2
3 // an array of pin numbers to which LEDs are attached
4 int ledPins[] = { 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };
5
6 void setup() {
7     // loop over the pin array and set them all to output:
8     for (int i = 0; i < ledCount; i++) {
9         pinMode(ledPins[i], OUTPUT);
10    }
11 }
12
13 void loop() {
14     // makes the "i"th LED of LED bar graph bright in turn
15     for (int i = 0; i < ledCount; i++) {
16         barGraphDisplay(i);
17     }
18     // makes the "i"th LED of LED bar graph bright in reverse order
19     for (int i = ledCount; i > 0; i--) {
20         barGraphDisplay(i - 1);
21     }
22 }
23
24 void barGraphDisplay(int ledOn) {
25     // make the "ledOn"th LED of LED bar graph on and the others off
26     for (int i = 0; i < ledCount; i++) {
27         if (i == ledOn)
28             digitalWrite(ledPins[i], HIGH);
29         else
30             digitalWrite(ledPins[i], LOW);
31     }
32     delay(100);
33 }
```

We have modified the code inside the function loop() and make the LED bar graph light in order, and then light in reverse order cyclically.

Verify and upload the code, then you will see the reciprocating LED light water on LED bar graph.



# Chapter 4 LED Blink Smoothly

In the previous chapter, we have used Sketch on the Freenove UNO board to control up to 10 LED blink and learned some basic knowledge of programming. Now, let us try to make LED emit different brightness of light.

## Project 4.1 LEDs Emit Different Brightness

Now, let us use Freenove UNO board to make 4 LED emit different brightness of light.

### Component list

Freenove UNO x1	Breadboard x1
	
USB cable x1	LED x4
	
Jumper M/M x5	Resistor 220Ω x4
	

## Circuit knowledge

At first, let us learn the knowledge how to use the circuit to make LED emit different brightness of light,

### PWM

PWM, namely Width Modulation Pulse, is a very effective technique for using digital signals to control analog circuits. The common processors can not directly output analog signals. PWM technology makes it very convenient to achieve this purpose.

PWM technology uses digital pins to send certain frequency of square waves, that is, the output of high level and low level last for a period alternately. The total time for each set of high level and low level is generally fixed, which is called period (the reciprocal of the period is frequency). Output high time is generally called pulse width, and the percentage of pulse width is called duty cycle.

The longer the output of high level last, the larger the duty cycle and the larger the corresponding voltage in analog signal. The following figures show how the analog signal voltage vary between 0V-5V (high level is 5V) corresponding to the pulse width 0%-100%:



The larger PWM duty ratio, the larger the output power. So we can use PWM to control the brightness of LED, the speed of DC motor and so on.

## Code knowledge

We will use new code knowledge in this section.

### Return value of function

We have learned and used the function without return value, now we will learn how to use the function with return value. A function with return value is shown as follow:

```

1 int sum(int i, int j) {
2     int k = i + j;
3     return k;
4 }
```

"int" declare the type of return value of the function sum(int i, int j). If the type of the return value is void, the function does not return a value.

One function can only return one value. It is necessary to use the return statement to return the value of function.

When the return statement is executed, the function will return immediately regardless of code behind return statement in this function.

A function with return value is called as follows:

```

1 int a = 1, b = 2, c = 0;
2 c = sum(1, 2);           // after the execution the value of c is 3
```

A function with a return value can also be used as a parameter of functions, for example:

```
1 delay(sum(100, 200));
```

It is equivalent to the following code:

```
1 delay(300);
```

### return

We have learned the role of the return statement in a function with a return value. It can also be used in functions without return value, and there is no data behind the return keyword:

```
1 return;
```

In this case, when the return statement is executed, the function will immediately end its execution rather than return in the end of the function. For example:



## Circuit

Use D5, D6, D9, D10 pin on Freenove UNO board to drive 4 LEDs.



## Sketch

### Sketch 4.1.1

Now let us use sketch to make 4 LEDs emit different brightness of light. We will transmit signal to make the 4 ports connected to LEDs output the PWM waves with duty cycle of 2%, 10%, 50%, and 100% to let the LED emit different brightness of the light.

```

1 // set pin numbers:
2     int ledPin1 = 5,           // the number of the LED1 pin
3         ledPin2 = 6,           // the number of the LED2 pin
4         ledPin3 = 9,           // the number of the LED3 pin
5         ledPin4 = 10;          // the number of the LED4 pin
6

```

```

7 void setup() {
8     // initialize the LED pin as an output:
9     pinMode(ledPin1, OUTPUT);
10    pinMode(ledPin2, OUTPUT);
11    pinMode(ledPin3, OUTPUT);
12    pinMode(ledPin4, OUTPUT);
13 }
14
15 void loop()
16 {
17     // set the ports output PWM waves with different duty cycle
18     analogWrite(ledPin1, map(2, 0, 100, 0, 255));
19     analogWrite(ledPin2, map(10, 0, 100, 0, 255));
20     analogWrite(ledPin3, map(50, 0, 100, 0, 255));
21     analogWrite(ledPin4, map(100, 0, 100, 0, 255));
22 }
```

After the initialization of the 4 ports, we set the ports to output PWM waves with different duty cycle. Take ledPin1 as an example, firstly map 2% to the 0-255 range, and then output the PWM wave with duty cycle of 2%,

1	analogWrite(ledPin1, map(2, 0, 100, 0, 255));
---	-----------------------------------------------

#### analogWrite(pin, value)

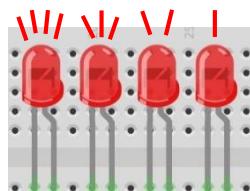
Arduino Software provides the function `analogWrite (pin, value)` which can make ports directly output PWM waves. Only the digital pin marked "˜" symbol on The UNO board can use this function to output PWM waves. In the function `analogWrite(pin, value)`, the parameter "pin" specifies the port used to output PWM wave. The range of value is 0-255, which represent the duty ratio 0%-100%.

In order to use this function, we need to set the port to output mode.

#### map(value, fromLow, fromHigh, toLow, toHigh)

This function is used to remap a value, which will return a new value whose percent in the rang of toLow-toHigh is equat to the percent of "value" accouting for in range of fromLow-fromHigh. Such as 1 is in the range 0-1 in the maximum and the maximum value in the scope of 0-2 is 2, that is, the result value map (1, 0, 1, 0, 2) is 2.

Verify and upload the code, then you will see the 4 LEDs emit light with different brightness.



## Project 4.2 LED Blink Smoothly

We will learn how to make a LED blink smoothly, that is, breathing light.

### Component list

The Component list is basically the same as those in last section. And we need to get rid of a few LEDs and resistors.

### Circuit

Remove some LEDs and resistors connected to D6, D9, D10 pin on Freenove UNO board in circuit of the last section.



## Sketch

### Sketch 4.2.1

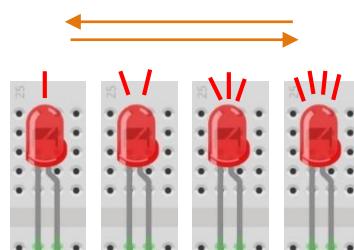
Now complete sketch to make brightness of LED change from dark to bright, and then from the bright to dark. That is to make the duty cycle of the PWM wave change from 0%-100%, and then from 100%-0% cyclically.

```

1 // set pin numbers:
2 int ledPin = 5;           // the number of the LED pin
3
4 void setup() {
5     // initialize the LED pin as an output:
6     pinMode(ledPin, OUTPUT);
7 }
8
9 void loop() {
10    // call breath() cyclically
11    breath(ledPin, 6);
12    delay(500);
13 }
14
15 void breath(int ledPin, int delayMs) {
16     for (int i = 0; i <= 255; i++) { // "i" change from 0 to 255
17         analogWrite(ledPin, i);      // corresponding duty cycle change from 0%-100%
18         delay(delayMs);          // adjust the rate of change of brightness
19     }
20     for (int i = 255; i >= 0; i--) { // "i" change from 255 to 0
21         analogWrite(ledPin, i);      // corresponding duty cycle change from 0%-100%
22         delay(delayMs);          // adjust the rate of change in brightness
23     }
24 }
```

Through two cycles, the duty cycle of the PWM wave changes from 0% to 100%, and then from 100% to 0% cyclically. `delay(ms)` function is used to control the rate of change in the "for" cycle, and you can try to modify the parameters to modify the rate of change in brightness.

Verify and upload the code, then you will see that the brightness of the LED changes from dark to light, and from the light to dark cyclically.



# Chapter 5 Control LED Through Push Button

From previous chapter, we have used Freenove UNO to output signals to make 10 LEDs flash, and make one LED emit different brightness. Now, let's learn how to get the input signal.

## Project 5.1 Control LED Through Push Button

We will use the Freenove UNO to get the status of the push button, and show that through LED.

### Component list

Freenove UNO x1	Breadboard x1
	
USB cable x1	LED x1 Resistor 220Ω x1 Resistor 10kΩ x2 Push button x1
Jumper M/M x4	   

## Circuit knowledge

### Connection of Push Button

In Chapter 1, we connect push button directly to power line of the circuit to control the LED to light on or off. In digital circuits, we need to use the push button as the input signal. The recommended connection is as follows:



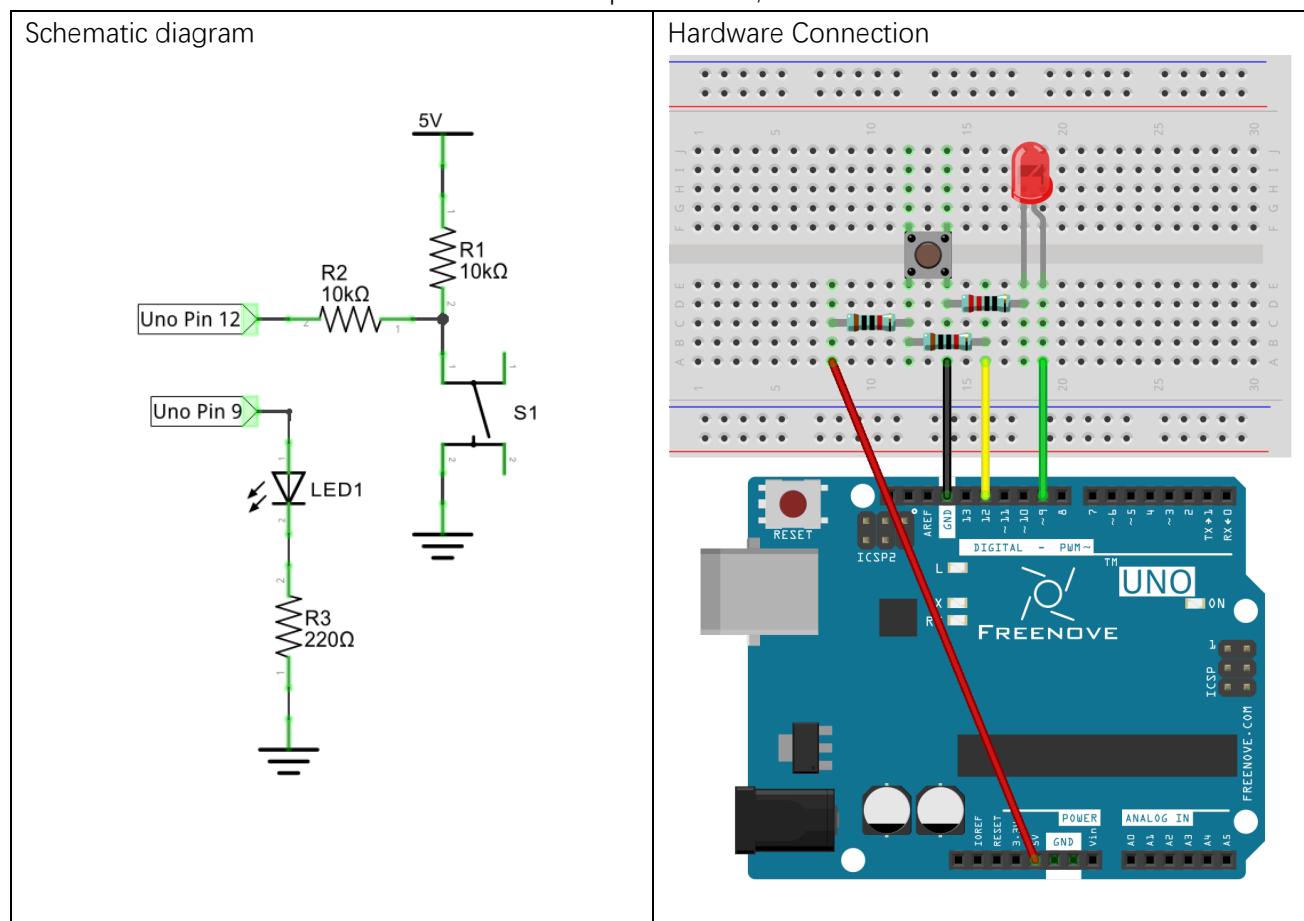
In the above circuit diagram, when the button is not pressed, 5V (high level) will be detected by UNO port; and 0V (low level) when the button is pressed. The role of Resistor R2 here is to prevent the port from being set to output high level by accident, which could be connected directly to the cathode and cause a short circuit when the button is pressed.

The following diagram shows another connection, in which the level detected by UNO port is opposite to above diagram, when the button is pressed or not.



## Circuit

Use D12 of Freenove UNO to detect the status of push button, and D9 to drive LED.



## Sketch

### Sketch 5.1.1

Now, write code to detect the state of push button, and show that through LED.

```

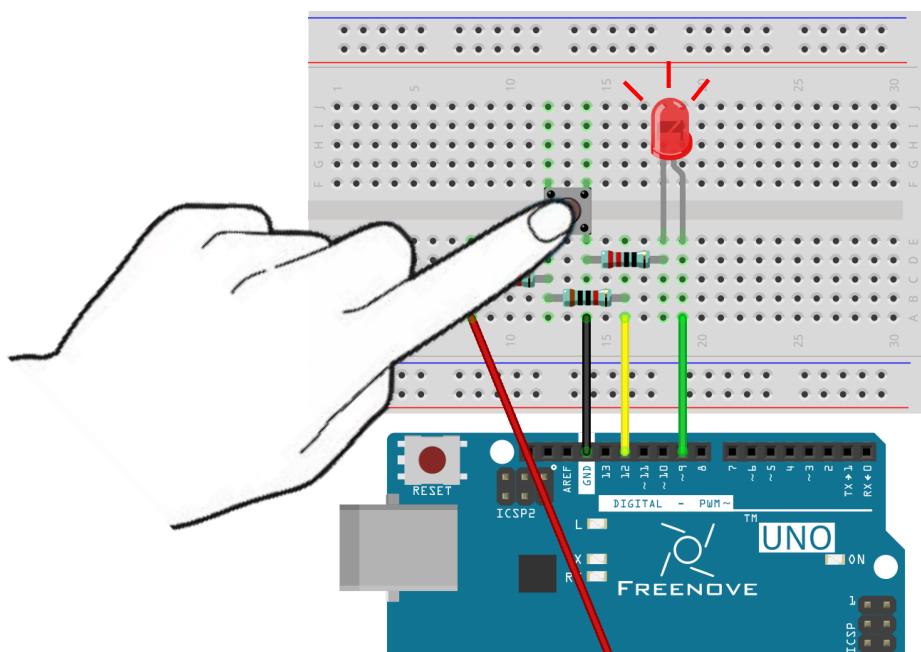
1 int buttonPin = 12; // the number of the push button pin
2 int ledPin = 9; // the number of the LED pin
3
4 void setup() {
5     pinMode(buttonPin, INPUT); // set push button pin into input mode
6     pinMode(ledPin, OUTPUT); // set LED pin into output mode
7 }
8
9 void loop() {
10    if (digitalRead(buttonPin) == HIGH) // if the button is not pressed
11        digitalWrite(ledPin, LOW); // switch off LED
12    else // if the button is pressed
13        digitalWrite(ledPin, HIGH); // switch on LED
14 }
```

After the port is initialized, the LED will be switched on or off in accordance with the state of the pin connected to push button.

#### **digitalRead(pin)**

Arduino Software provides a function `digitalRead(pin)` to obtain the state of the port pin. The return value is HIGH or LOW, that is, high level or low level.

Verify and upload the code, press the button, then LED lights up; loosen the button, then LED lights off.



## Project 5.2 Change LED State by Push Button

In the last section, we have finished the experiment that LED lights on when push button is pressed, and off as soon as it's released. Now, let's try something new: each time you pressed the button down, the state of LED will be changed.

### Component List

Same with the last section.

### Circuit knowledge

#### Debounce for push button

When push button is pressed, the potential won't transfer from one state to another state ideally. In fact, it will occur a continuous bounce process before it becomes final stable state.



If you detect the state of push button, there may be repeated "press" and "release" detected during one process of pressing. So it is necessary to eliminate the influence caused by the bounce. Here is the most direct and effective way: when the changed signals are received by Freenove UNO for the first time, the program does not operate immediately, just waits for a certain time to skip the bounce process and confirm the final state of push button.

### Circuit

Same with the last section.

## Sketch

### Sketch 5.2.1

Now, write the code to detect the state of push button. Every time you pressed, the state of LED will be changed.

```
1 int buttonPin = 12; // the number of the push button pin
2 int ledPin = 9;      // the number of the LED pin
3 boolean isLighting = false; // define a variable to save the state of LED
4
5 void setup() {
6     pinMode(buttonPin, INPUT); // set push button pin into input mode
7     pinMode(ledPin, OUTPUT); // set LED pin into output mode
8 }
9
10 void loop() {
11     if (digitalRead(buttonPin) == LOW) { // if the button is pressed
12         delay(10); // delay for a certain time to skip the bounce
13         if (digitalRead(buttonPin) == LOW) { // confirm again if the button is pressed
14             reverseLED(); // reverse LED
15             while (digitalRead(buttonPin) == LOW); // wait for releasing
16             delay(10); // delay for a certain time to skip bounce when the button is released
17         }
18     }
19 }
20
21 void reverseLED() {
22     if (isLighting) { // if LED is lighting,
23         digitalWrite(ledPin, LOW); // switch off LED
24         isLighting = false; // store the state of LED
25     }
26     else { // if LED is off,
27         digitalWrite(ledPin, HIGH); // switch LED
28         isLighting = true; // store the state of LED
29     }
30 }
```

We use a function `reverseLED ()` to change the state of the LED.

When judging the push button state, if it is detected "pressed down", wait for a certain time to detect again to eliminate the effect of bounce. When confirmed, it starts to wait for the push button to be released, and waits for a certain time to eliminate the effect of bounce after it is released.

Verify and upload the code, then each time you press the button, LED changes its state accordingly.

### Sketch 5.2.2

Beginners can upload Sketch\_5.2.2 to try not dealing with the push button bounce and see what happens.

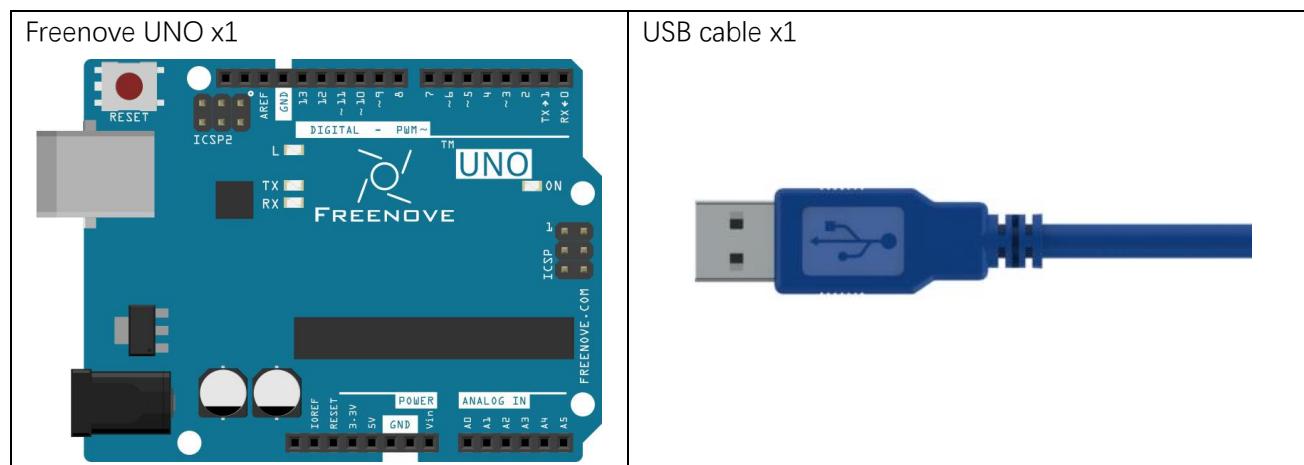
# Chapter 6 Serial

Earlier, we already try to output signals to LED, and get the input signal of push button. Now, we can try a more advanced means of communication—serial communication.

## Project 6.1 Send data through Serial

We will use the serial port on Freenove UNO to send data to computer.

### Component list



## Code knowledge

### Bit and Byte

As mentioned earlier, the computer uses a binary signal. A binary signal is called 1 bit, and 8 bits organized in order is called 1 byte. Byte is the most basic unit of storage and communications for processors. 1 byte can represent a  $2^8=256$  numbers, that is, 0-255. For example:

As to binary number 10010110, "0" usually present the lowest in code.

Sequence	7	6	5	4	3	2	1	0
Number	1	0	0	1	0	1	1	0

When a binary number need to be converted to decimal number, first, the nth number of it need be multiplied by n power of 2, then sum all multiplicative results. Take 10010110 as an example:

$$1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 150$$

We can make decimal number divided by 2 to convert it to binary number. Quotient continuously obtained divided by 2 until quotient is zero. Arrange all remainders from right to left in a line. Then we complete the conversion. For example:

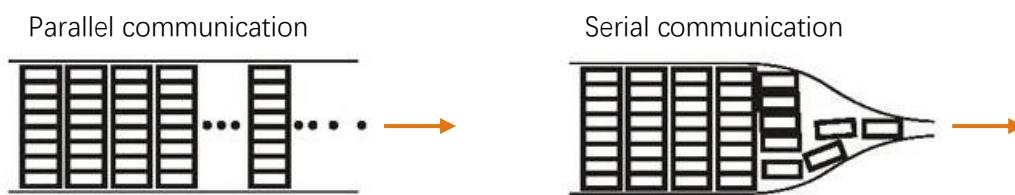
		Remainder	Sequence
2	150	..... 0	0
2	75	..... 1	1
2	37	..... 1	2
2	18	..... 0	3
2	9	..... 1	4
2	4	..... 0	5
2	2	..... 0	6
2	1	..... 1	7
	0		

The result is 10010110.

## Circuit knowledge

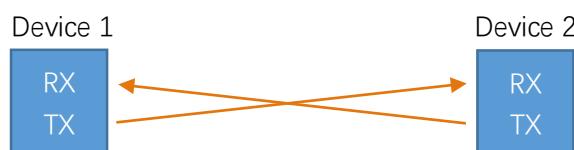
### Serial and parallel communication

Serial communication uses one data cable to transfer data one bit by another in turn. Parallel communication means that the data is transmitted simultaneously on multiple cables. Serial communication takes only a few cables to exchange information between systems, especially suitable for computers to computer, long distance communication between computers and peripherals. Parallel communication is faster, but with more cables and the high cost, so it is not appropriate for long distance communication.



### Serial communication

Serial communication generally refers to the Universal Asynchronous Receiver/Transmitter (UART). And it is commonly used in electronic circuit communication. It has two communication lines, one is responsible for sending data (TX line) and another for receiving data (RX line). The serial communication connections two devices use is as follows:



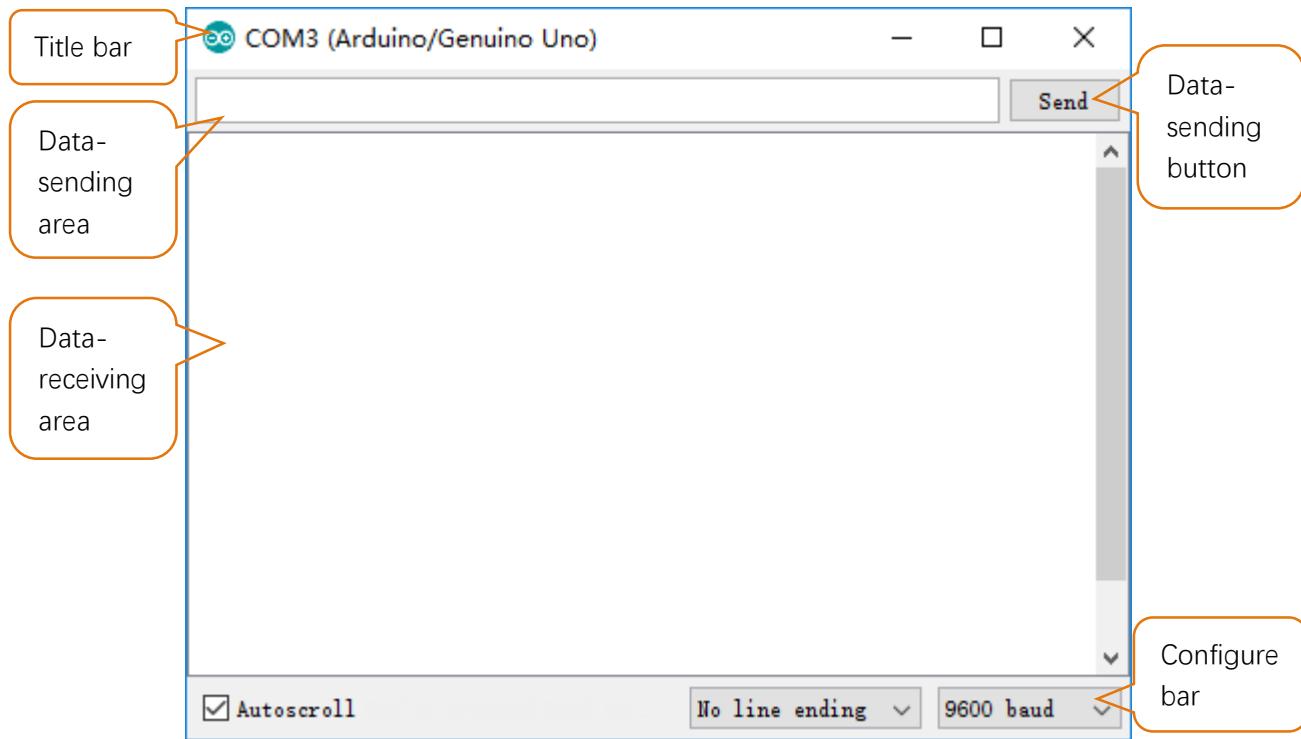
Before serial communication starts, the baud rate in both sides must be the same. Only use the same baud rate can the communication between devices be normal. The baud rates commonly used is 9600 and 115200.

### Serial port on Arduino

Freenove UNO has integrated USB to serial transfer, could communicates with computer when USB cable get connected to it. Arduino Software also uploads code for Freenove UNO through the serial connection. Computer identifies serial devices connected to your computer as COMx. We can use the Serial Monitor window of Arduino Software to communicate with Freenove UNO, connect Freenove UNO to computer through the USB cable, choose the right device, and then click the Serial Monitor icon to open the Serial Monitor window.



Interface of Serial Monitor window is as follows. If you can't open it, make sure Freenove UNO had been connected to the computer, and choose the right serial port in the menu bar "Tools".



## Circuit

Connect Freenove UNO to the computer with USB cable.



## Sketch

### Sketch 6.1.1

Now, write code to send some text to the Serial Monitor window

```
1 int counter = 0; // define a variable as a data sending to serial port
2
```

```

3 void setup() {
4     Serial.begin(9600);           // initialize the serial port, set the baud rate to 9600
5     Serial.println("UNO is ready!"); // print the string "UNO is ready!"
6 }
7
8 void loop() {
9     // print variable counter value to serial
10    Serial.print("counter:");
11    Serial.println(counter);      // print the variable counter value
12    delay(500);                // wait 500ms to avoid cycling too fast
13    counter++;                 // variable counter increases 1
14 }

```

setup() function initializes the serial port and send the string "UNO is ready!".

Then continuously sends variable counter values in the loop () function.

### Serial class

Class is a C++ language concept. Arduino Software supports C++ language, which is a language extension. We don't explain specifically the concept here, only describes how to use it. If you are interested, you can learn that by yourself. Serial is a class name, which contains variables and functions. You can use the "." operational character to visit class variables and functions, such as:

Serial.begin(speed): Initialize serial port, the parameter is the serial port baud rate;

Serial.print(val): Send string, the parameter here is what you want to send;

Serial.println(val): Send newline behind string.

Verify and upload the code, open the Serial Monitor, then you'll see data sent from UNO.

If it is not displayed correctly, check the Serial Monitor whether configuration in the lower right corner of the window is correct.





## Project 6.2 Receive Data through Serial Port

From last section, we use Serial port on Freenove UNO to send data to a computer, now we will use that to receive data from computer.

### Component list

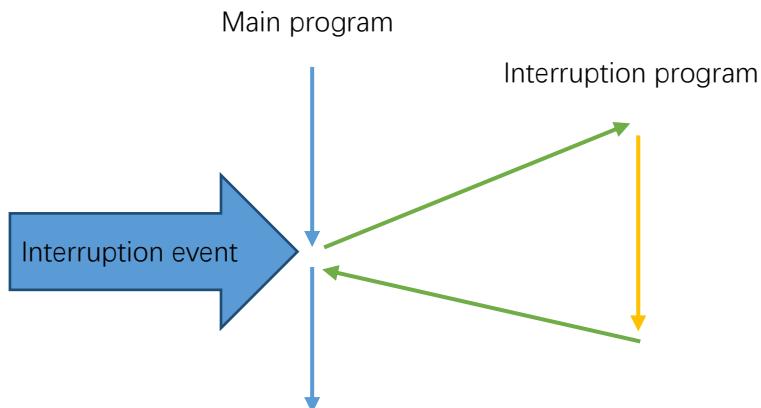
Same with last section.

### Code knowledge

#### Interruption

Interruption is controller's response to an event. The event causing an interruption is interruption source. We'll illustrate this interruption concept. If you're watching TV, while water is heating, then you have to check whether the water is boiling or not from time to time, so you can't concentrate on watching TV. But interruption can work as warning device for kettle, and before the water is boiling, you can focus on watching TV until a beep warning comes out, then go to handle it.

Advantages of interruption here: Processor won't need to check whether the event has happened every now and then., but when the event occurs, it informs the controller immediately. When an interruption occurs, the processor will jump to the interrupt function to handle interruption events, then return to the interruption place after finishing it and go on this program.



### Circuit

Same with last section.

## Sketch

### Sketch 6.2.1

Now, write code to receive the characters from Serial Monitor window, and send it back.

```
1  char inChar;      // define a variable to store characters received from serial port
2
3  void setup() {
4      Serial.begin(9600);          // initialize serial port, set baud rate to 9600
5      Serial.println("UNO is ready!"); // print the string "UNO is ready!"
6  }
7
8  void loop() {
9      if (Serial.available()) {    // judge whether data has been received
10         inChar = Serial.read();   // read one character
11         Serial.print("UNO received:"); // print the string "UNO received:"
12         Serial.println(inChar);    // print the received character
13     }
14 }
```

In the setup() function, we initialize the serial port and send the string "UNO is ready!". Then, the loop() function will continuously detect whether there are data needs to be read. If so, read the character and send it back.

#### Serial Class

Serial.available(): return bytes of data that need to be read by serial port;

Serial.read(): return 1 byte of data that need to be read by serial port.

Verify and upload the code, open the Serial Monitor, write character in sending area, click Send button, then you'll see information returned from UNO.



**char type**

char type variable can represent a character, but char type cannot store characters directly. It stores numbers to replace characters. char type occupies 1-byte store area, and use a value 0-127 to correspond to 128 characters. The corresponding relation between number and character is ruled by ASCII table. For more details of ASCII table, please refer to the appendix of this book.

Example: Define char aChar = 'a', bChar = '0', then the decimal value of aChar is 97, bChar will be 48.

**Sketch 6.2.2**

When serial port receives data, it can trigger an interrupt event, and enter into the interrupt handling function. Now we use an interrupt to receive information from Serial Monitor window, and sends it back. To illustrate the interrupt does not influence the program's running, we will constantly send changing number in loop () function.

```

1  char inChar;      // define a variable to store character received from serial port
2  int counter = 0; // define a variable as the data sent to Serial port
3
4  void setup() {
5      Serial.begin(9600);           // initialize serial port and set baud rate to 9600
6      Serial.println("UNO is ready!"); // print the string "UNO is ready!"
7  }
8
9  void loop() {
10     // Print value of variable counter to serial
11     Serial.print("counter:");
12     Serial.println(counter);    // print the value of variable "counter"
13     delay(1000);              // wait 1000ms to avoid cycling too fast
14     counter++;                // variable "counter" increases 1
15 }
16
17 void serialEvent() {
18     if (Serial.available()) {    // judge whether the data has been received
19         inChar = Serial.read(); // read one character
20         Serial.print("UNO received:");
21         Serial.println(inChar); // print the received character
22     }
23 }
```

void serialEvent () function here is the serial port interrupt function. When serial receives data, processor will jump to this function, and return to the original interrupt place to proceed after execution. So loop () function's running is not affected.

Verify and upload the code, open the Serial Monitor, then you'll see the number constantly sent from UNO. Fill character in the sending area, and click the Send button, then you'll see the string returned from UNO.

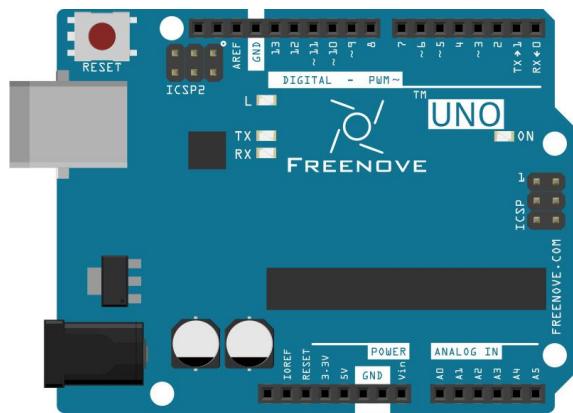


## Project 6.3 Application of Serial

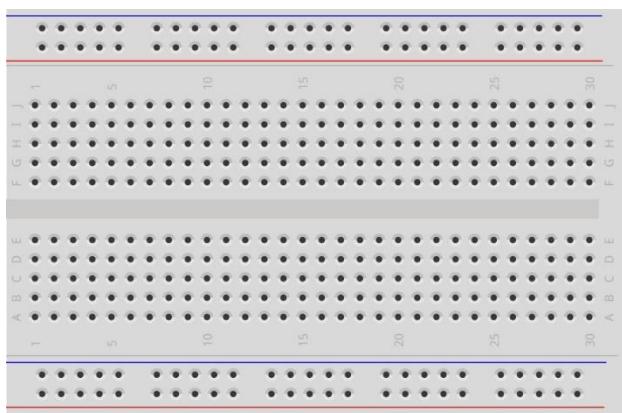
We will use the serial port on Freenove UNO to control one LED.

### Component list

Freenove UNO x1



Breadboard x1



USB cable x1



Jumper M/M x2



LED x1

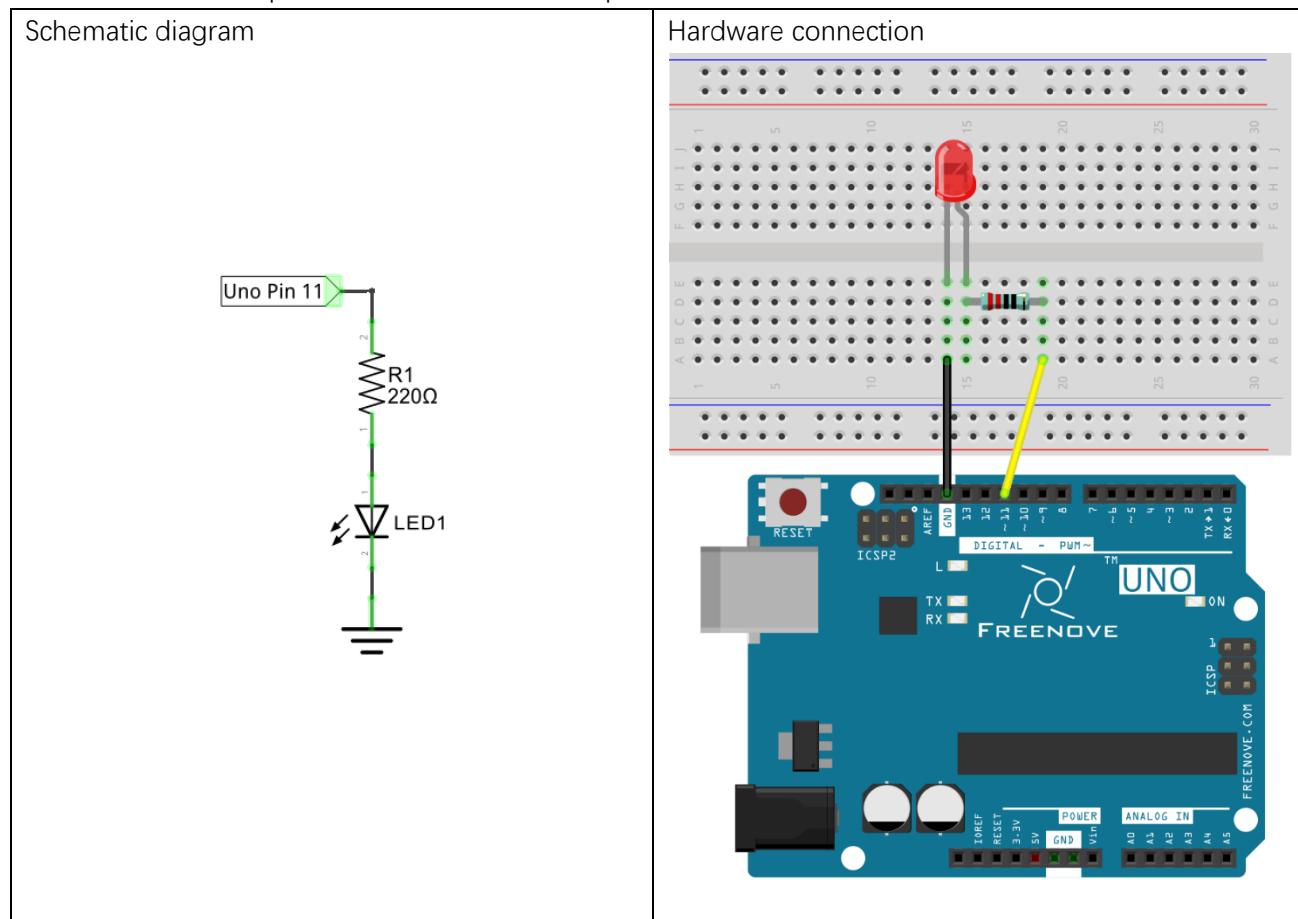


Resistor 220Ω x1



## Circuit

Here we will use D11 pin of Freenove UNO to output PWM to drive 1 LED.



## Sketch

### Sketch 6.3.1

Code is basically the same with Sketch 6.2.1. But after receiving the data, Freenove UNO will convert it into PWM duty cycle from output port.

```

1 int inInt;           // define a variable to store the data received from serial
2 int counter = 0;    // define a variable as the data sending to serial
3 int ledPin = 11;    // the number of the LED pin
4
5 void setup() {
6     pinMode(ledPin, OUTPUT);          // initialize the LED pin as an output
7     Serial.begin(9600);             // initialize serial port, set baud rate to 9600
8     Serial.println("UNO is ready!"); // print the string "UNO is ready!"
9 }
10

```

```
11 void loop() {  
12     if (Serial.available()) {          // judge whether the data has been received  
13         inInt = Serial.parseInt();      // read an integer  
14         Serial.print("UNO received:"); // print the string "UNO received:"  
15         Serial.println(inInt);        // print the received character  
16         // convert the received integer into PWM duty cycle of ledPin port  
17         analogWrite(ledPin, constrain(inInt, 0, 255));  
18     }  
19 }
```

When serial receives data, it converts the data into PWM duty cycle of output port, and then causes LED to emit light with corresponding brightness.

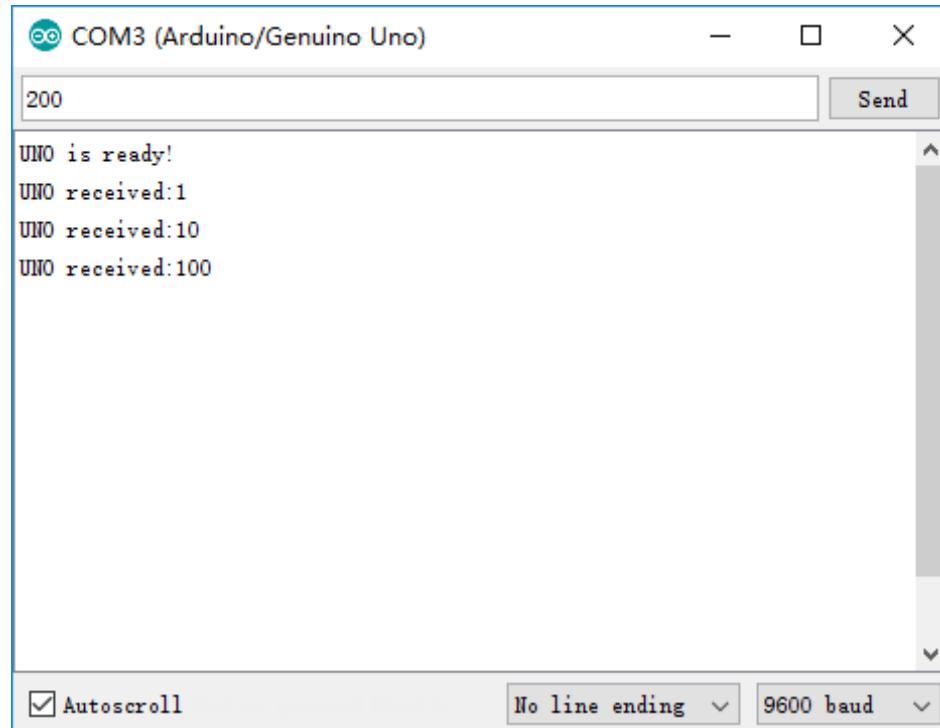
**Serial Class**

Serial.parseInt(): Receive an int type number as the return value.

**constrain(x, a, b)**

Limit x between a and b, if  $x < a$ , return a; if  $x > b$ , return b.

Verify and upload the code, open the Serial Monitor, and put a number in the range 0-255 into the sending area and click the Send button. Then you'll see information returned from UNO, meanwhile, LED can emit light with different brightness according to the number you send.



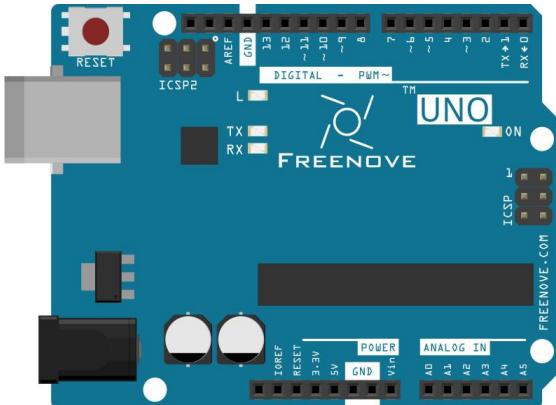
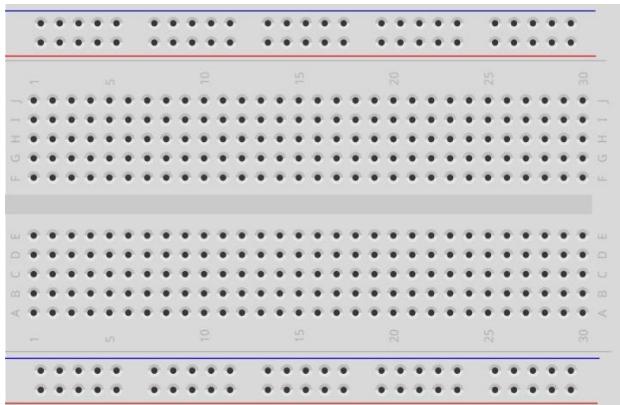
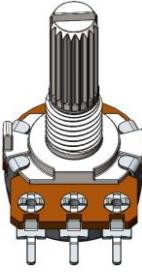
# Chapter 7 ADC

Before, we have learned the digital ports of Freenove UNO, and tried the output and input signals. Now, let's learn how to use analog ports.

## Project 7.1 ADC

ADC is used to convert analog signals into digital signals. Control chip on Freenove has integrated this function. Now let us try to use this function to convert analog signals into digital signals.

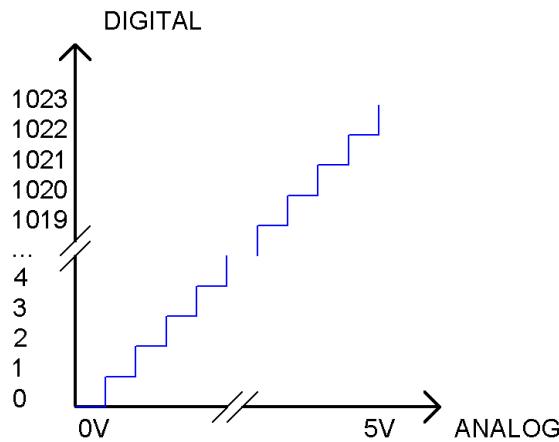
### Component list

Freenove UNO x1	Breadboard x1
	
USB cable x1	Rotary potentiometer x1
	
Jumper M/M x3	

## Circuit knowledge

### ADC

ADC, Analog-to-Digital Converter, is a device used to convert analog to digital. Freenove UNO has a 10 bits ADC, that means the resolution is  $2^{10}=1024$ , and the range (here is 5V) will be divided equally into 1024 parts. The analog of each section corresponds to one ADC value. So the more bits ADC has, the denser the partition of analog will be, also the higher precision of the conversion will be.



Subsection 1: the analog in rang of 0V-5/1024V corresponds to digital 0;

Subsection 2: the analog in rang of 5 /1024V-2\*5/1024V corresponds to digital 1;

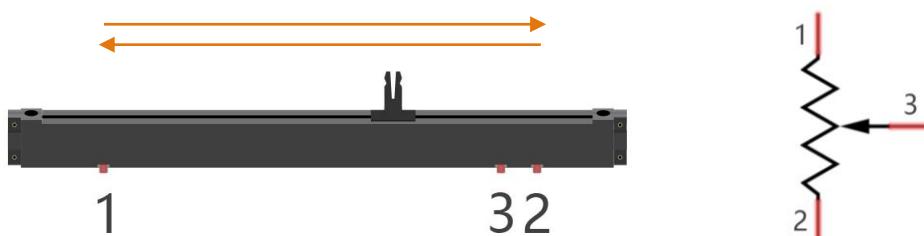
...

The following analog will be divided accordingly.

## Component knowledge

### Potentiometer

Potentiometer is a resistive component with three terminal parts and its resistance can be adjusted in accordance with according to a certain change rule. Potentiometer is often made up by resistance and removable brush. When the brush moves along the resistor body, there will be resistance or voltage that has a certain relationship with displacement on the output side (3). Figure shown below is the linear sliding potentiometer and its symbol.



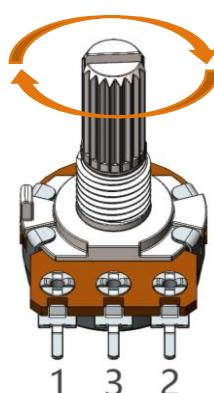
Pin 1 and pin 2 of the potentiometer is connected to two ends of a resistor body respectively, and pins 3 is connected to a brush. When the brush moves from pin 1 to pin 2, resistance between pin 1 and pin 3 will increase up to max resistance of the resistor body linearly, and resistance between pin 2 and pin 3 will decrease down to 0 linearly.

In the circuit, the two sides of resistor body are often connected to the positive and negative electrodes of a power respectively. When you slide the brush of pin 3, you can get a certain voltage in the range of negative voltage to positive voltage of the power supply.



### Rotary potentiometer

Rotary potentiometer and linear potentiometer have similar function; the only difference is: the resistance is adjusted through rotating the potentiometer.



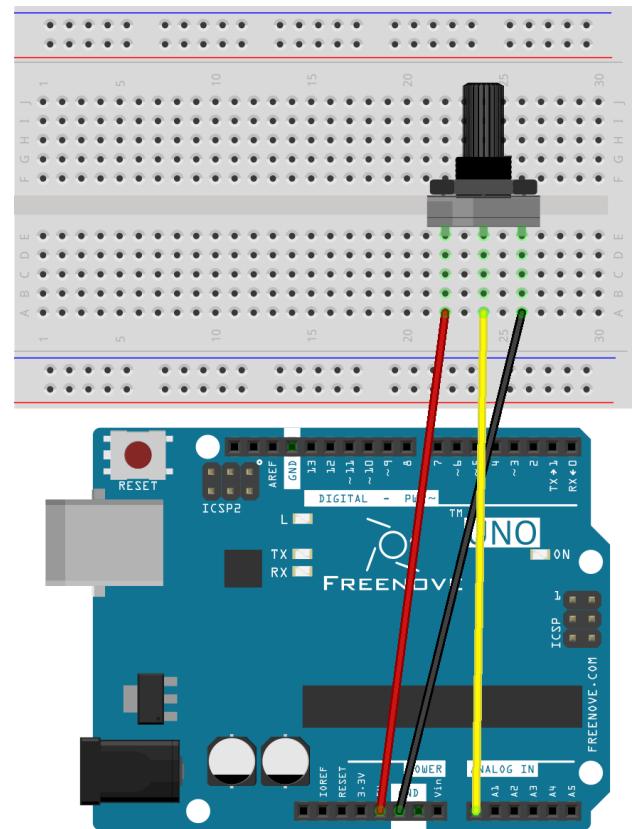
## Circuit

Use A0 port on Freenove UNO to detect the voltage of rotary potentiometer.

Schematic diagram



Hardware connection



## Sketch

### Sketch 7.1.1

Now, write code to detect the voltage of rotary potentiometer, and send the data to Serial Monitor window of Arduino Software through serial port.

```
1 int adcValue; // Define a variable to save ADC value
2 float voltage; // Define a variable to save the calculated voltage value
3
4 void setup() {
5     Serial.begin(9600); // Initialize the serial port and set the baud rate to 9600
6     Serial.println("UNO is ready!"); // Print the string "UNO is ready!"
7 }
8
9 void loop() {
10    adcValue = analogRead(A0); // Convert analog of A0 port to digital
11    voltage = adcValue * (5.0 / 1023.0); // Calculate voltage according to digital
12    // Send the result to computer through serial
13    Serial.print("convertValue:");
14    Serial.println(adcValue);
15    Serial.print("Voltage:");
16    Serial.println(voltage);
17    delay(500);
18 }
```

From the code, we get the ADC value of A0 pin, then convert it into voltage and sent to the serial port.

Verify and upload the code, open the Serial Monitor, then you will see the original ADC value and converted voltage sent from UNO.

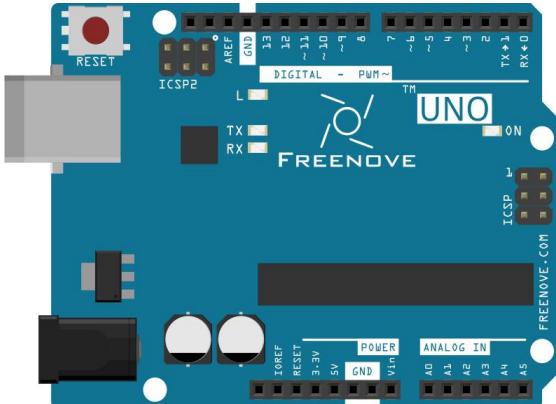
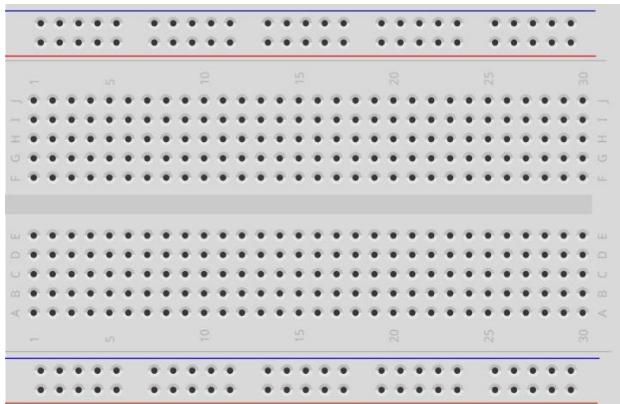
Turn the rotary potentiometer shaft, and you can see the voltage change.



## Project 7.2 Control LED by Potentiometer

In the last section, we have finished reading ADC value and convert it into voltage. Now, we will try to use potentiometer to control the brightness of LED.

### Component list

Freenove UNO x1	Breadboard x1
	
USB cable x1	Rotary potentiometer x1
	
Jumper M/M x5	LED x1
	Resistor 220Ω x1
	

## Circuit

Use A0 on Freenove UNO to detect the voltage of rotary potentiometer, and use D9 to control one LED.



## Sketch

### Sketch 7.2.1

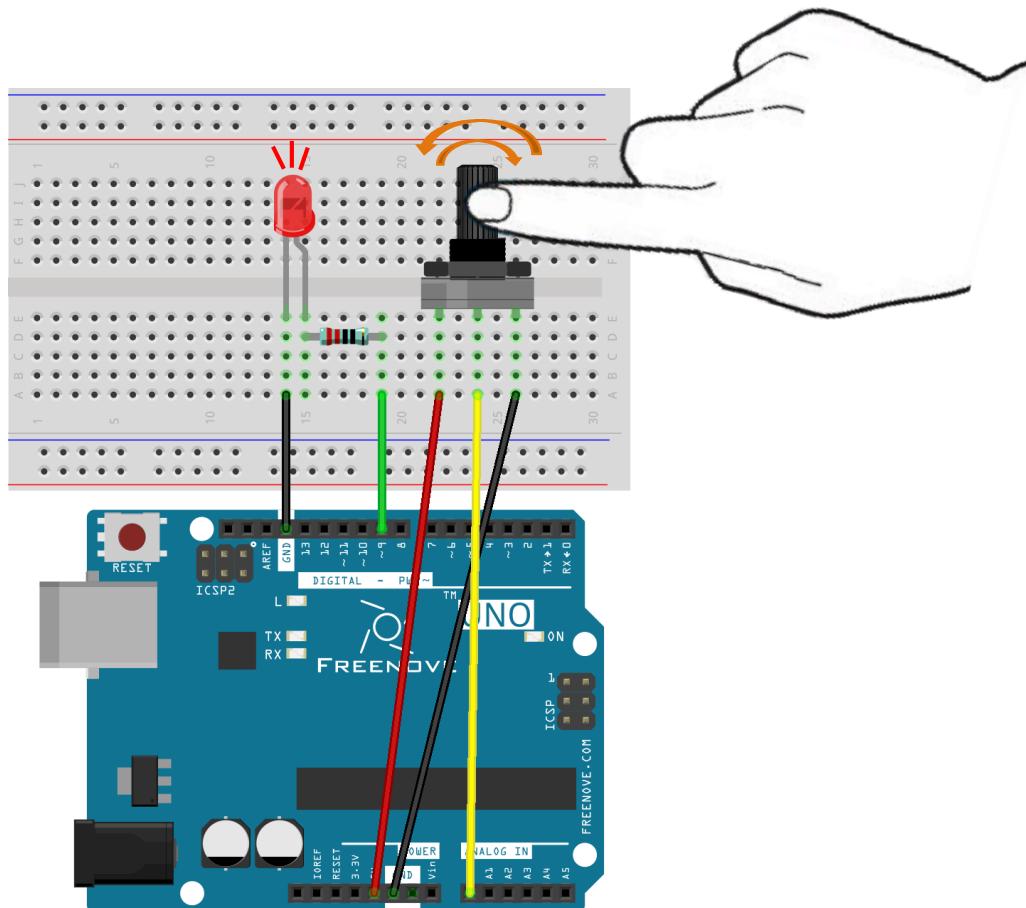
Now, write the code to detect the voltage of rotary potentiometer, and control LED to emit light with different brightness according to that.

```

1 int adcValue; // Define a variable to save the ADC value
2 int ledPin = 9; // Use D9 on Freenove UNO to control the LED
3
4 void setup() {
5     pinMode(ledPin, OUTPUT); // Initialize the LED pin as an output
6 }
7
8 void loop() {
9     adcValue = analogRead(A0); // Convert the analog of A0 port to digital
10    // Map analog to the 0-255 range, and works as PWM duty cycle of ledPin port
11    analogWrite(ledPin, map(adcValue, 0, 1023, 0, 255));
12 }
```

In the code, we get the ADC value of A0 pin and map it to PWM duty cycle of LED pin port. According to different LED brightness, we can see the voltage changes easily.

Verify and upload the code, turn the rotary potentiometer shaft, then you will see the LED brightness changes.



## Project 7.3 Control LED through Photoresistor

In the last section, we have finished reading ADC value and converted it into LED brightness. There are many components, especially the sensor whose output is analog. Now, we will try to use photoresistor to measure the brightness of light.

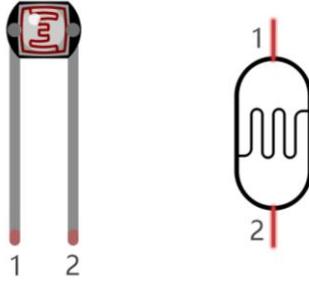
### Component list

Freenove UNO x1	Breadboard x1
	
USB cable x1	Photoresistor x1
	LED x1
Jumper M/M x5	Resistor 10kΩ x1
	Resistor 220Ω x1

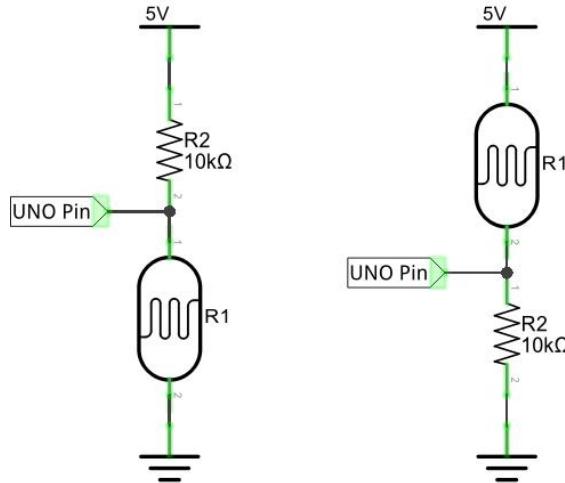
## Component knowledge

### Photoresistor

Photoresistor is a light sensitive resistor. When the strength that light casts onto the photoresistor surface is not the same, resistance of photoresistor will change. With this feature, we can use photoresistor to detect light intensity. Photoresistor and symbol are as follows.



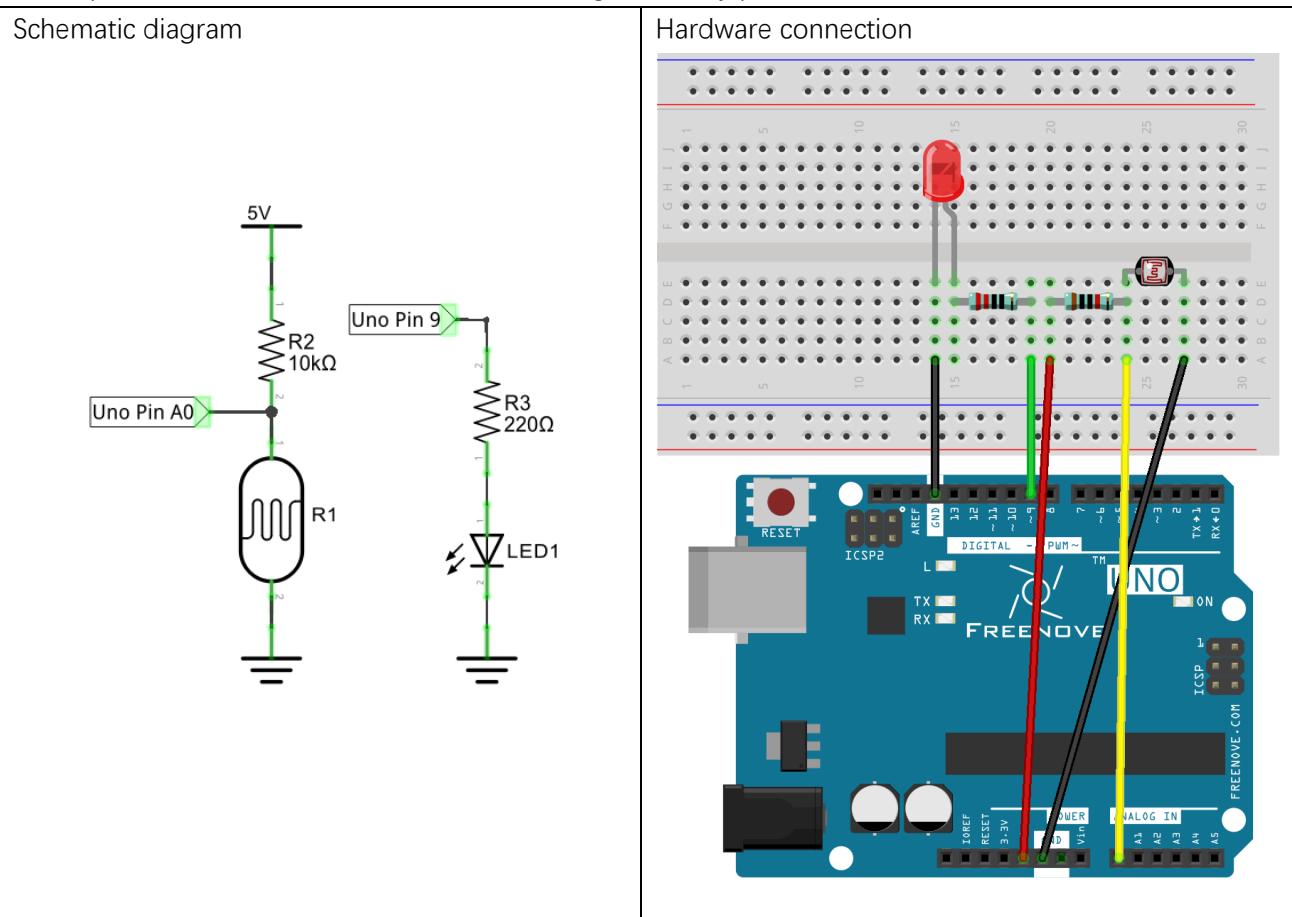
The circuit below is often used to detect the change of photoresistor resistance:



In the above circuit, when photoresistor resistance changes due to light intensity, voltage between photoresistor and resistor R1 will change, so light's intensity can be obtained by measuring the voltage.

## Circuit

Use A0 port on Freenove UNO to detect the voltage of rotary potentiometer, and use D9 to control one LED.



## Sketch

### Sketch 7.3.1

Now, write code to detect the voltage of rotary potentiometer, and control LED to emit light with different brightness according to that.

```

1 int convertValue; // Define a variable to save the ADC value
2 int ledPin = 9; // The number of the LED pin
3
4 void setup() {
5     pinMode(ledPin, OUTPUT); // Set ledPin into output mode
6 }
7
8 void loop() {
9     convertValue = analogRead(A0); // Read analog voltage value of A0 port, and save
10    // Map analog to the 0~255 range, and works as ledPin duty cycle setting
11    analogWrite(ledPin, map(convertValue, 0, 1023, 0, 255));
12 }
```

In the code, we get the ADC value of A0 pin, map that to PWM duty cycle of LED pin port. According to the LED brightness, we can see the voltage changes easily.

Verify and upload the code, cover photoresistor with your hand, then you can see the LED brightness changes.



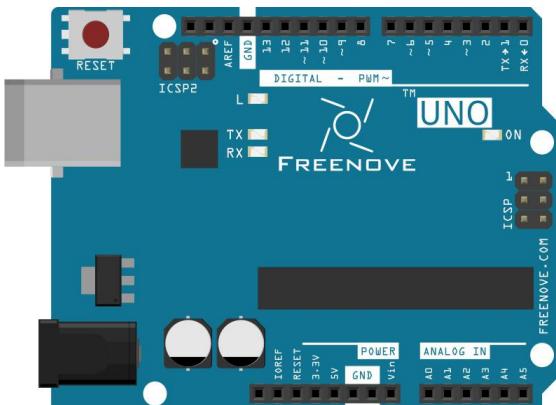
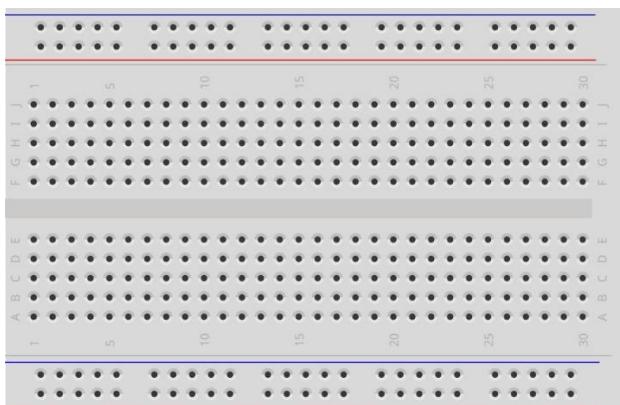
# Chapter 8 RGB LED

Before, we have learned to apply the analog port and ADC of Freenove UNO. Now, we'll use ADC to control RGB LED.

## Project 8.1 Control RGB LED through Potentiometer

RGB LED has three different-color LED inside, and we will use 3 potentiometers to control these 3 LEDs to emit light with different brightness, and observe what will happen.

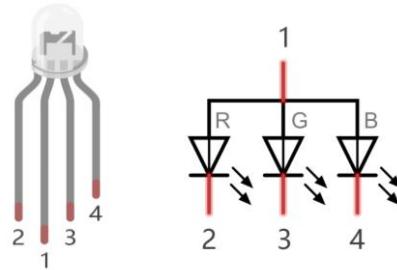
### Component list

Freenove UNO x1	Breadboard x1
	
USB cable x1	Rotary potentiometer x3
	
Jumper M/M x15	RGB LED x1
	
	Resistor 220Ω x3
	

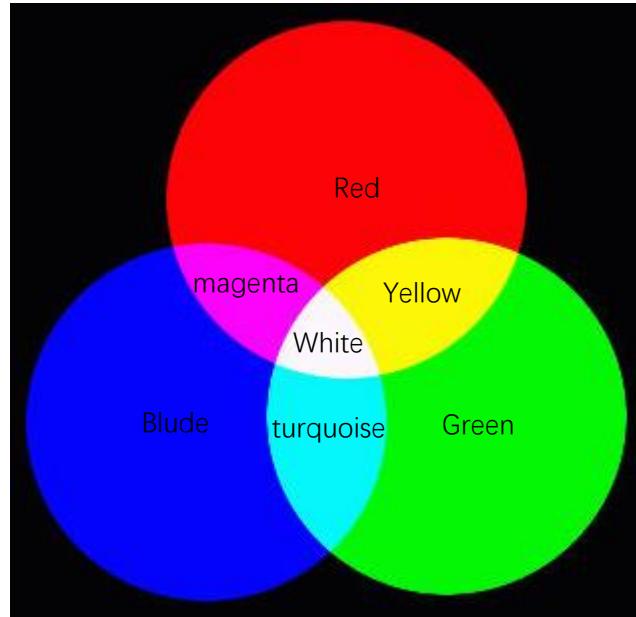
## Component knowledge

### RGB LED

RGB LED has integrated 3 LEDs that can respectively emit red, green and blue light, and it has 4 pins. The long pin (1) is the common port, that is, 3 LED's cathode or anode. The RGB LED with common anode and its symbol are as follows. By controlling these 3 LED to emit light with different brightness, we can make RGB LED emit various colors of light.



Red, green, and blue light are called tricolor light. When you combine these three primary-color light with different brightness, it can produce almost all visible light. Computer screens, single pixel of cell phone screen, neon, and etc. are working through this principle.



We know from previous section that, UNO controls LED to emit a total 256(0-255) different brightness by PWM. So, through the combination of three different colors of LED, RGB LED can emit  $256^3 = 16777216$  Colors, 16Million colors.

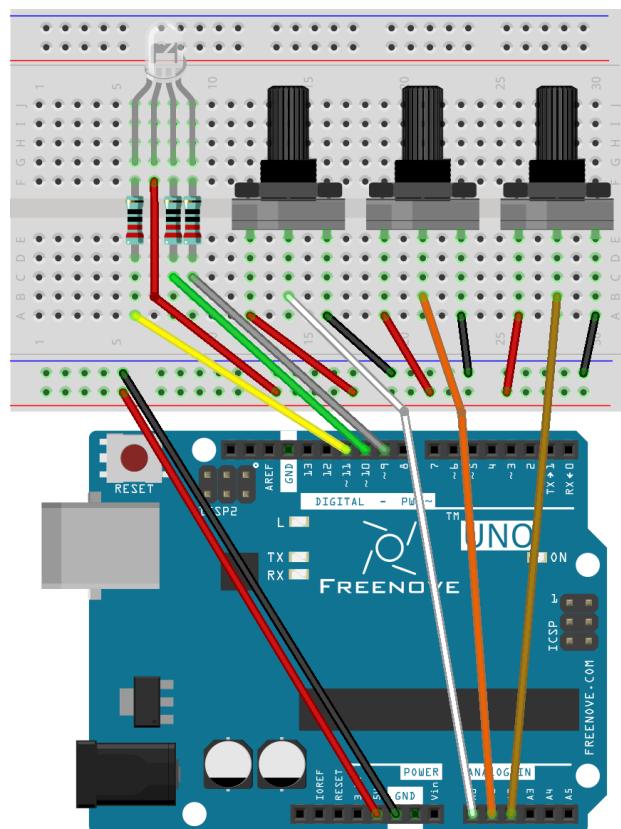
## Circuit

Use A0, A1, A2 port of Freenove UNO to detect the voltage of rotary potentiometer, and control RGB LED by D9, D10, D11.

Schematic diagram



Hardware connection



## Sketch

### Sketch 8.1.1

Now, write code to detect the voltage of these three rotary potentiometers, and convert that into PWM duty cycle to control 3 LED inside the RGB LED.

```
1 // set pin numbers:  
2 int ledPinR = 11; // the number of the LED R pin  
3 int ledPinG = 10; // the number of the LED G pin  
4 int ledPinB = 9; // the number of the LED B pin  
5  
6 void setup() {  
7     // initialize the LED pin as an output:  
8     pinMode(ledPinR, OUTPUT);  
9     pinMode(ledPinG, OUTPUT);  
10    pinMode(ledPinB, OUTPUT);  
11}  
12  
13 void loop() {  
14     int adcValue; // Define a variable to save the ADC value  
15     // Convert analog of A0 port into digital, and work as PWM duty cycle of ledPinR port  
16     adcValue = analogRead(A0);  
17     analogWrite(ledPinR, map(adcValue, 0, 1023, 0, 255));  
18     // Convert analog of A1 port into digital, and work as PWM duty cycle of ledPinG port  
19     adcValue = analogRead(A1);  
20     analogWrite(ledPinG, map(adcValue, 0, 1023, 0, 255));  
21     // Convert analog of A2 port into digital, and work as PWM duty cycle of ledPinB port  
22     adcValue = analogRead(A2);  
23     analogWrite(ledPinB, map(adcValue, 0, 1023, 0, 255));  
24 }
```

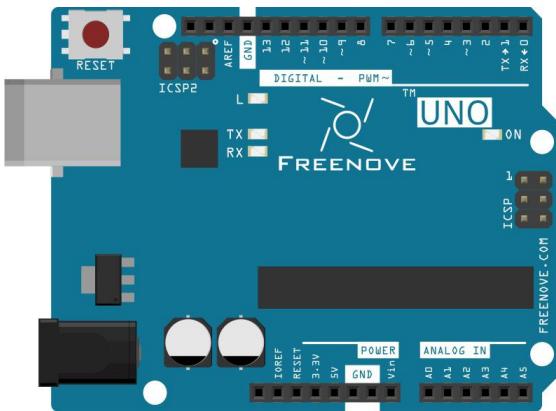
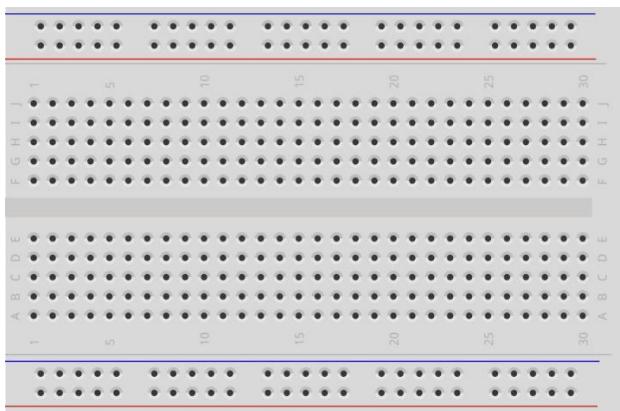
In the code, we get the voltage of three rotary potentiometers, and convert that into PWM duty cycle to control these three LED of the RGB LED to emit light with different brightness.

Verify and upload the code, turn the three rotary potentiometer shaft, then you can see the LED light changes in color and brightness.

## Project 8.2 Colorful LED

From previous section, we have finished controlling the RGB LED to emit light with different color and brightness through three potentiometers. Now, we will try to make RGB LED emit colorful light automatically.

### Component list

Freenove UNO x1	Breadboard x1
	
USB cable x1	RGB LED x1
	
Jumper M/M x4	Resistor 220Ω x3
	

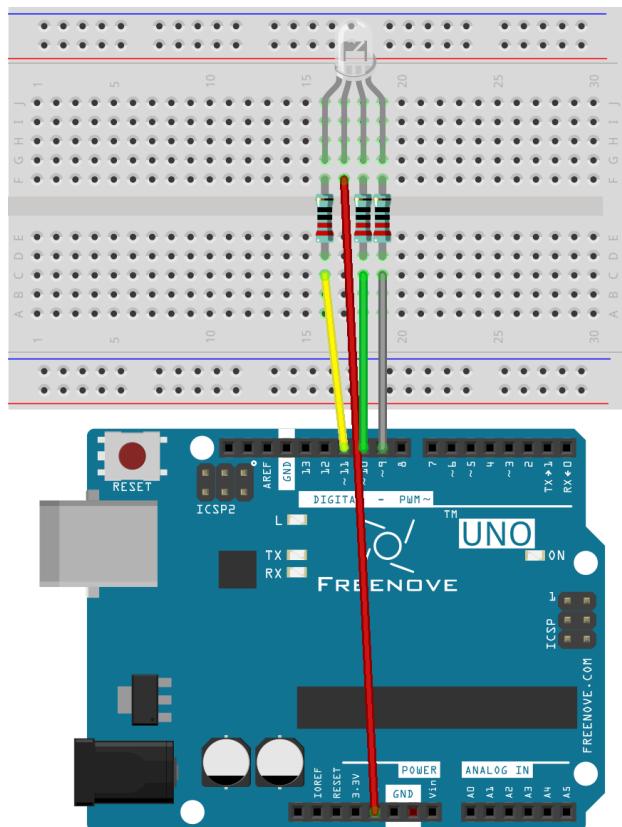
## Circuit

Use D9, D10, D11 of Freenove UNO to control RGB LED.

Schematic diagram



Hardware connection



## Sketch

### Sketch 8.2.1

Now, write code to generate three random number, and convert that into PWM duty cycle to control these three LED of RGB LED.

```
1 // set pin numbers:  
2 int ledPinR = 11; // the number of the LED R pin  
3 int ledPinG = 10; // the number of the LED G pin  
4 int ledPinB = 9; // the number of the LED B pin  
5  
6 void setup() {  
7     // initialize the LED pin as an output:  
8     pinMode(ledPinR, OUTPUT);  
9     pinMode(ledPinG, OUTPUT);  
10    pinMode(ledPinB, OUTPUT);  
11}  
12  
13 void loop() {  
14     // Generate three random numbers between 0-255 as the output PWM duty cycle of ledPin  
15     rgbLedDisplay(random(256), random(256), random(256));  
16     delay(500);  
17 }  
18  
19 void rgbLedDisplay(int red, int green, int blue) {  
20     // Set three ledPin to output the PWM duty cycle  
21     analogWrite(ledPinR, constrain(red, 0, 255));  
22     analogWrite(ledPinG, constrain(green, 0, 255));  
23     analogWrite(ledPinB, constrain(blue, 0, 255));  
24 }
```

In the code, we create three random number, and convert that into PWM duty cycle, controlling these three LED of RGB LED to emit light with different brightness. At regular intervals, a new random number will be created, so RGB LED will start flashing light with different colors and brightness.

#### random(min, max)

random (min, max) function is used to generate random number, and it will return a random value in the range (min, Max-1).

You can also use random (max) function, the function set the minimum value into 0 by default, and returns a random value in the range (0, Max-1).

Verify and upload the code, then RGB LED starts flashing with different colors and brightness.

# Chapter 9 Buzzer

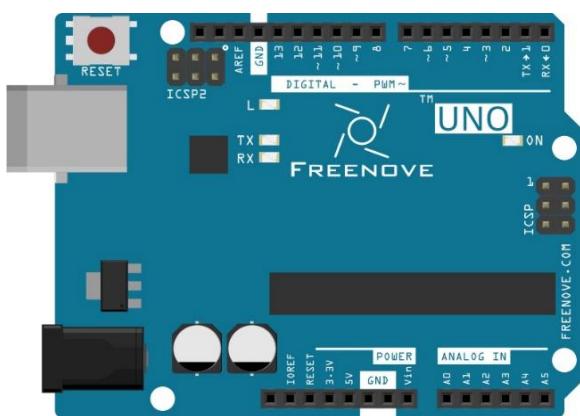
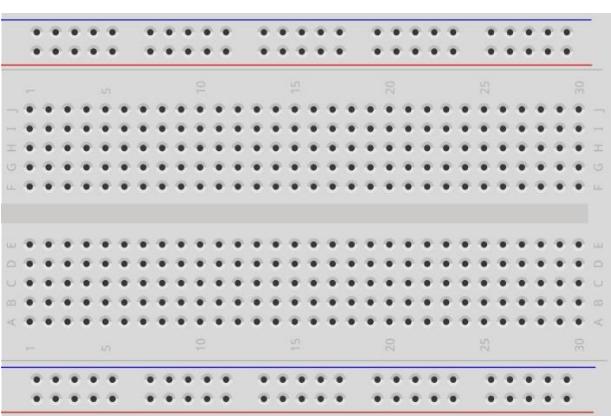
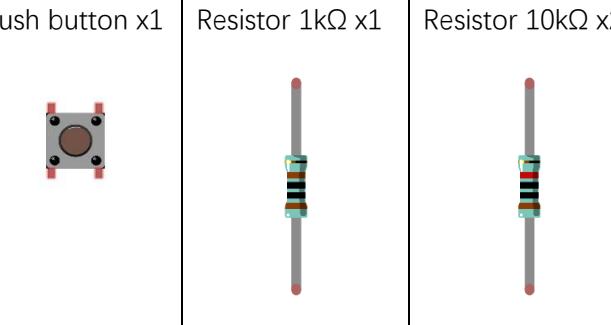
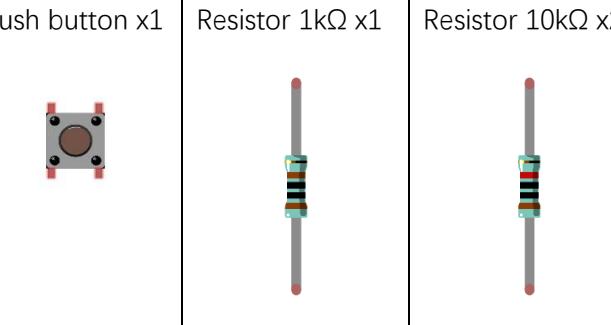
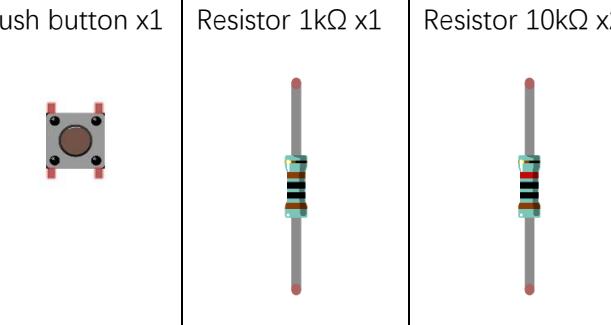
Before, we have used Freenove UNO and basic electronic components to carry out a series of interesting projects. Now, let's learn how to use some integrated electronic components and modules. These modules are usually integrated with a number of electronic components, so it has special features and uses.

In this chapter, we'll use a sounding module, buzzer. It has two types: active and passive buzzer.

## Project 9.1 Active Buzzer

First, let's study some knowledge about active buzzer.

### Component list

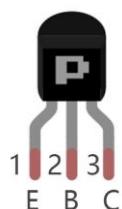
Freenove UNO x1		Breadboard x1	
USB cable x1		Jumper M/M x6	
NPN transistor x1		Push button x1	
Active buzzer x1		Resistor 1kΩ x1	
		Resistor 10kΩ x2	

## Component knowledge

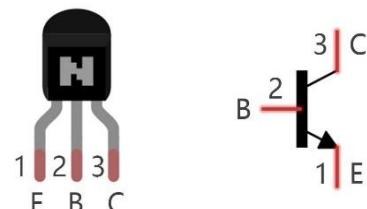
### Transistor

Transistor, the full name: semiconductor transistor, is a semiconductor device that controls current. Transistor can be used to amplify weak signal, or works as a switch. It has three electrodes(PINs): base (b), collector (c) and emitter (e). When there is current passing between "be", "ce" will allow several-fold current (transistor magnification) pass, at this point, transistor works in the amplifying area. When current between "be" exceeds a certain value, "ce" will not allow current to increase any longer, at this point, transistor works in the saturation area. Transistor has two types shown below: PNP and NPN,

PNP transistor



NPN transistor



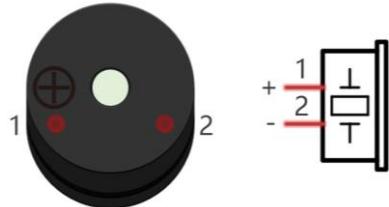
According to the transistor's characteristics, it is often used as a switch in digital circuits. For micro-controller's capacity of output current is very weak, we will use transistor to amplify current and drive large-current components.

The transistor marked 8550 is PNP, and 8050 is NPN.

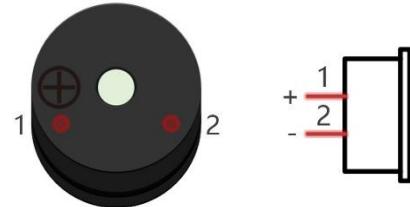
### Buzzer

Buzzer is a sounding component, which is widely used in electronic devices such as calculator, electronic warning clock, alarm. Buzzer has active and passive type. Active buzzer has oscillator inside, and it will make a sound as long as it is supplied with power. Passive buzzer requires external oscillator signal (generally use PWM with different frequency) to make a sound.

Active buzzer



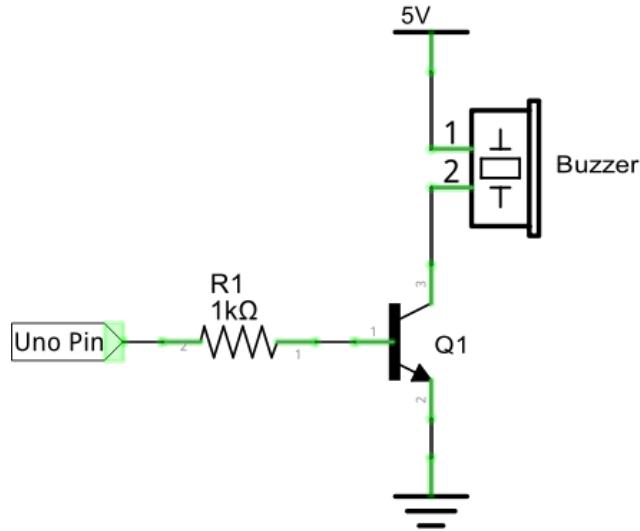
Passive buzzer



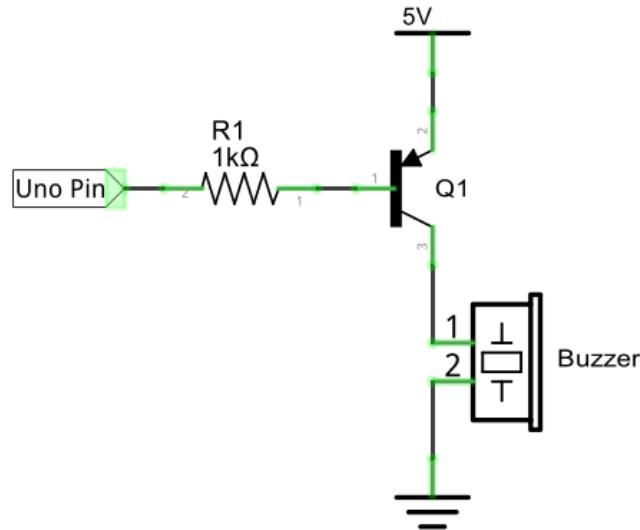
Active buzzer is easy to use. Generally, it can only make a specific frequency of sound. Passive buzzer requires an external circuit to make a sound, but it can be controlled to make a sound with different frequency. The resonant frequency of the passive buzzer is 2kHz, which means the passive buzzer is loudest when its resonant frequency is 2kHz.

Buzzer needs larger current when it works. But generally, microcontroller port cannot provide enough current for that. In order to control buzzer by UNO, transistor can be used to drive a buzzer indirectly.

When use NPN transistor to drive buzzer, we often adopt the following method. If UNO pin outputs high level, current will flow through R1, the transistor gets conducted, and the buzzer make a sound. If UNO pin outputs low level, no current flows through R1, the transistor will not be conducted, and buzzer will not sound.

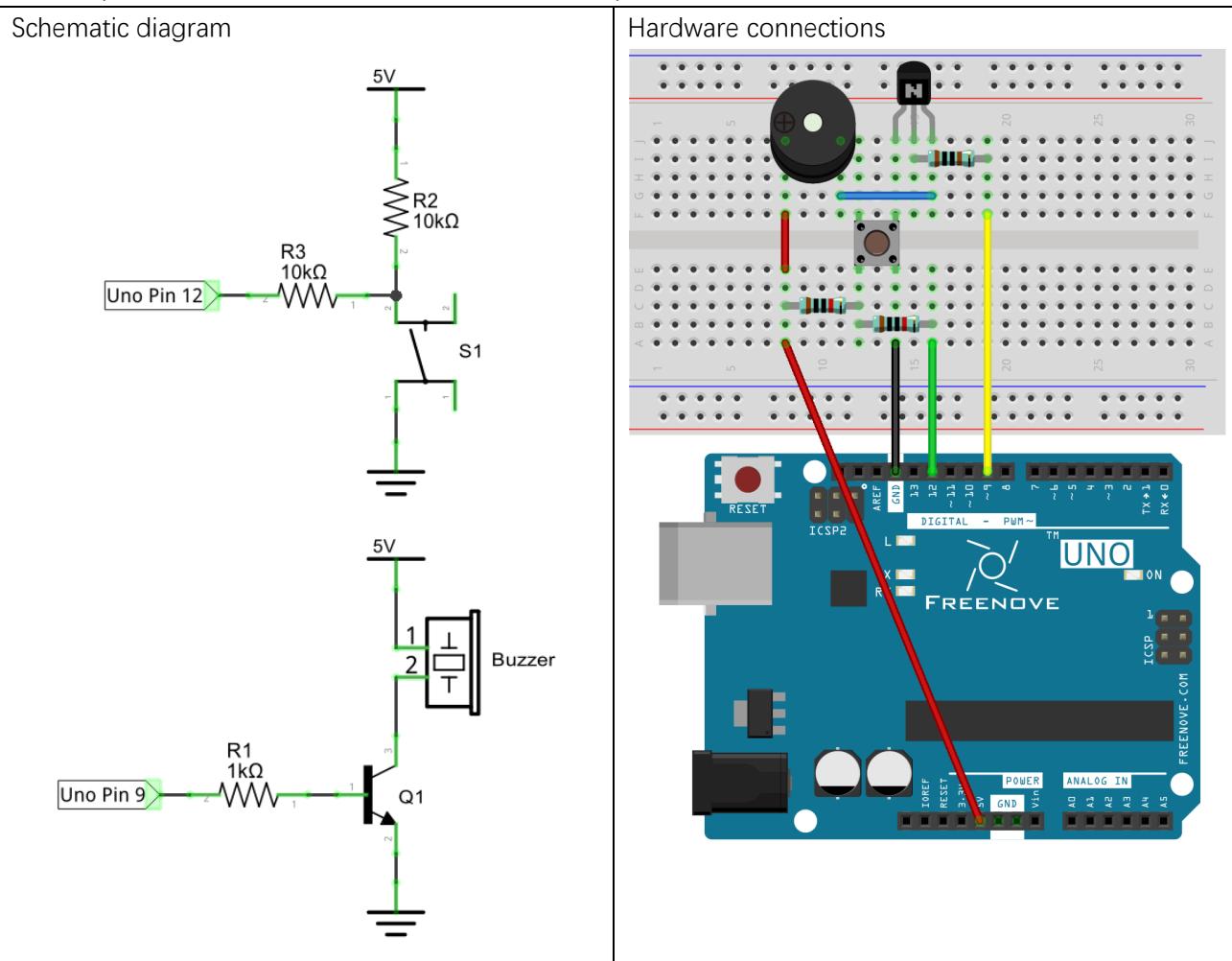


When use PNP transistor to drive buzzer, we often adopt the following method. If UNO pin outputs low level, current will flow through R1, the transistor gets conducted, buzzer make a sound. If UNO pin outputs high level, no current flows through R1, the transistor will not be conducted, and buzzer will not sound.



## Circuit

Use D12 port of Freenove UNO to detect the state of push button, and D9 to drive active buzzer.



## Sketch

### Sketch 9.1.1

Now, write code to detect the state of push button, and drive active buzzer to make a sound when it is pressed.

```

1 int buttonPin = 12; // the number of the push button pin
2 int buzzerPin = 9; // the number of the buzzer pin
3
4 void setup() {
5     pinMode(buttonPin, INPUT); // Set push button pin into input mode
6     pinMode(buzzerPin, OUTPUT); // Set Buzzer pin into output mode
7 }
8
9 void loop() {
10    if (digitalRead(buttonPin) == HIGH)// If the pin is high level, the button is not pressed.
11        digitalWrite(buzzerPin, LOW); // Turn off Buzzer
12    else // The button is pressed, if the pin is low level
13        digitalWrite(buzzerPin, HIGH); // Turn on Buzzer
14 }
```

In the code, we check the state of push button. When it is pressed, the output high level controls transistor to get conducted, and drives active buzzer to make a sound.

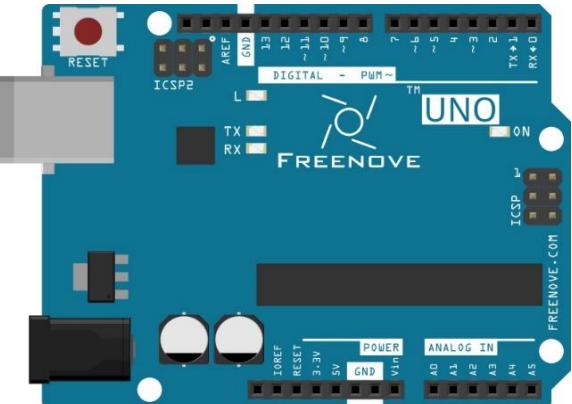
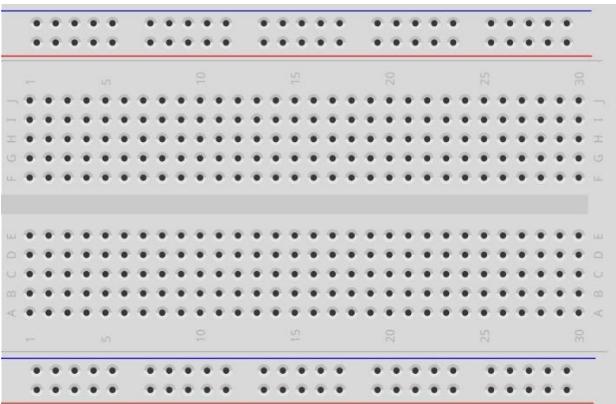
Verify and upload the code, press the push button, then the active buzzer will make a sound.



## Project 9.2 Passive Buzzer

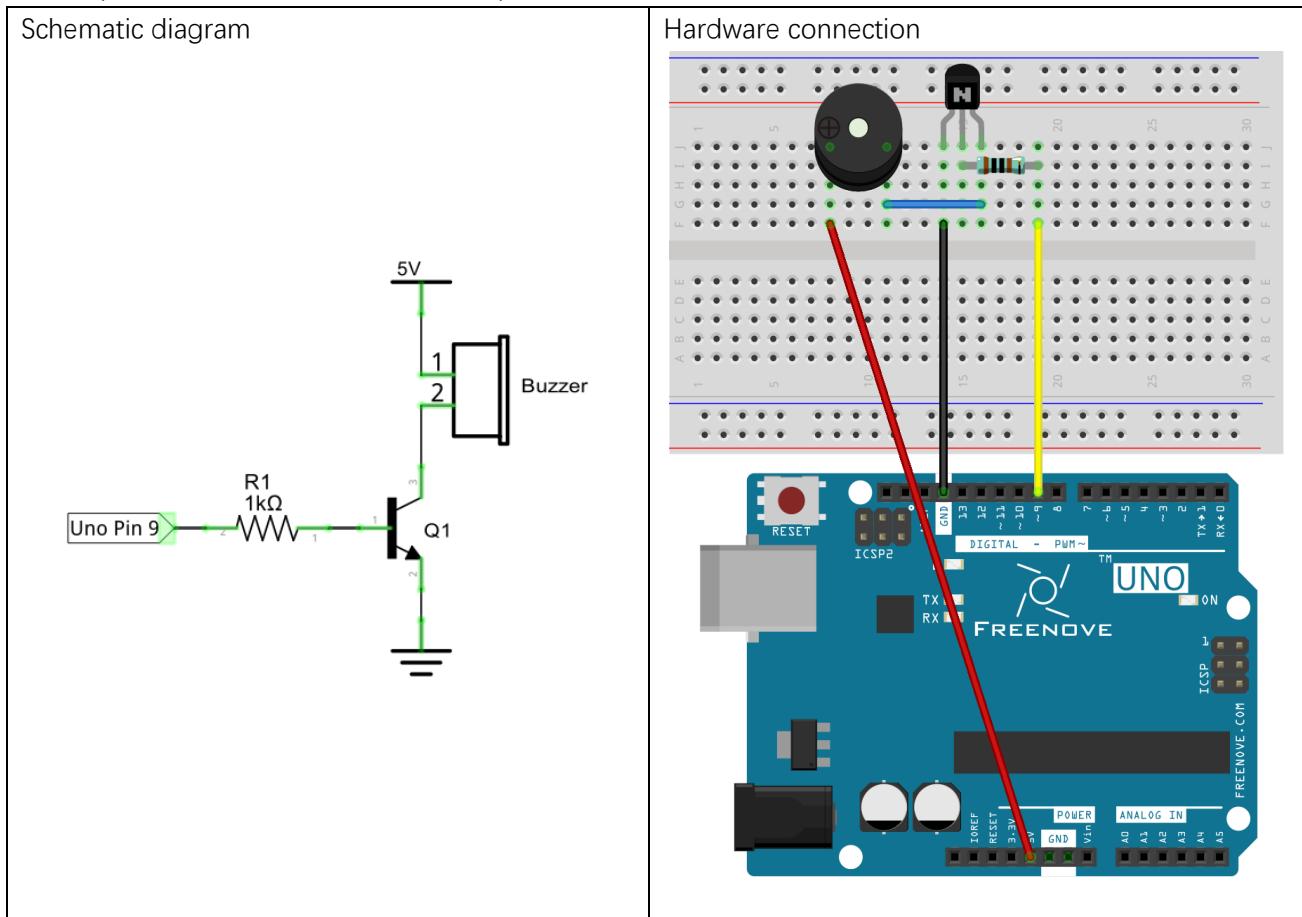
In the last section, we have finished using transistor to driven active buzzer to beep. Now, we will try to use passive buzzer to make a sound with different frequency.

### Component list

Freenove UNO x1	Breadboard x1	
		
USB cable x1	NPN transistor x1	
		Passive buzzer x1
Jumper M/M x4		Resistor 1kΩ x1
		

## Circuit

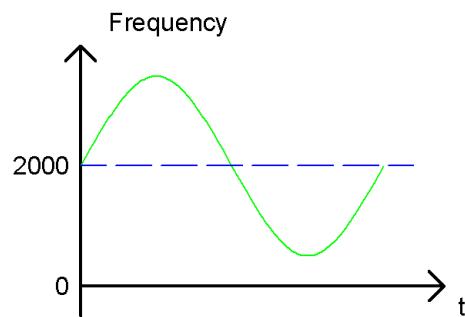
Use D9 port of Freenove UNO to drive a passive buzzer.



## Sketch

### Sketch 9.2.1

Now, write code to drive a passive buzzer to make a warning sound. The frequency of that conforms roughly to following sine curve over time:



Output PWM wave with different frequency to the port, which is connected to transistor, to drive buzzer to make a sound with different frequency.

```
1 int buzzerPin = 9;      // the number of the buzzer pin
2 float sinVal;          // Define a variable to save sine value
3 int toneVal;           // Define a variable to save sound frequency
4
5 void setup() {
6     pinMode(buzzerPin, OUTPUT); // Set Buzzer pin to output mode
7 }
8
9 void loop() {
10    for (int x = 0; x < 360; x++) {           // X from 0 degree->360 degree
11        sinVal = sin(x * (PI / 180));         // Calculate the sine of x
12        toneVal = 2000 + sinVal * 500;         // Calculate sound frequency according to the sine of x
13        tone(buzzerPin, toneVal);             // Output sound frequency to buzzerPin
14        delay(1);
15    }
16 }
```

In the code, use one loop to control the sound frequency, varying according to sine curve in the range  $2000 \pm 500$ .

```
10 for (int x = 0; x < 360; x++) {           // X from 0 degree->360 degree
11    sinVal = sin(x * (PI / 180));         // Calculate the sine of x
12    toneVal = 2000 + sinVal * 500;         // Calculate sound frequency according to the sine of x
13    tone(buzzerPin, toneVal);             // Output sound frequency to buzzerPin
14    delay(1);
15 }
```

The parameter of `sin()` function is radian, so we should convert unit of  $\pi$  from angle to radian first before using it.

### tone(pin, frequency)

Generates a square wave of the specified frequency (and 50% duty cycle) on a pin.

Verify and upload the code, passive buzzer starts making a warning sound.

You can try using PNP transistor to complete the project of this chapter again.

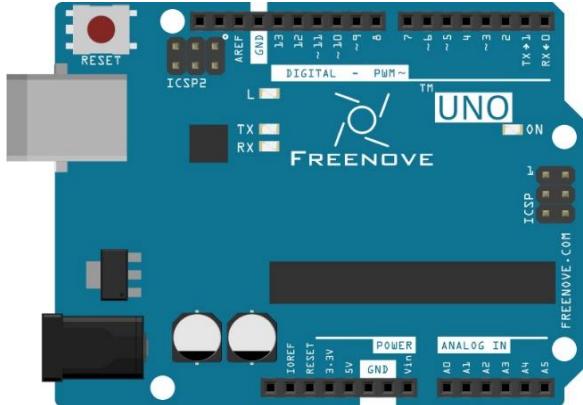
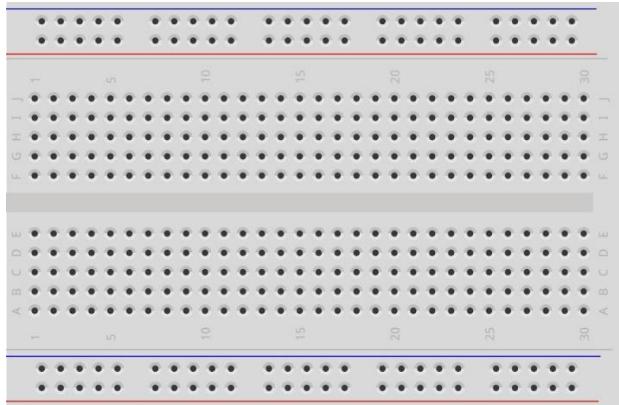
# Chapter 10 Motor

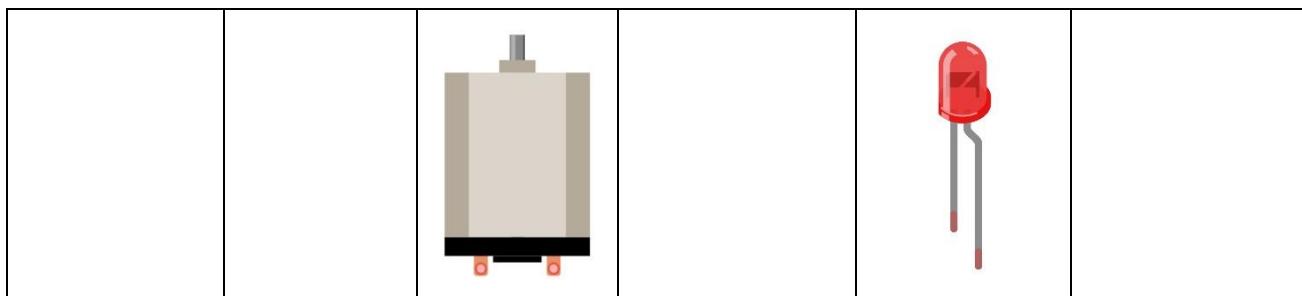
Before, we have done a series of interesting projects with Freenove UNO and basic electronic components. Now, let's study some movable electronic modules. In this chapter, we will learn to control the motor.

## Project 10.1 Control Motor by Relay

First, use relay to control the Motor.

### Component list

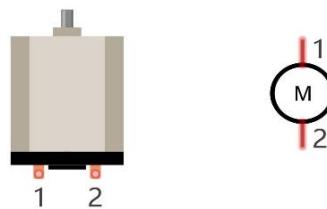
Freenove UNO x1 	Breadboard x1 			
USB cable x1 	Jumper M/M x8 			
AA Battery holder x1 	Resistor 10kΩ x2 	Resistor 1kΩ x1 	Resistor 220Ω x1 	
NPN transistor x1 	Relay x1 	Motor x1	Push button x1 	LED x1
				Diode x1 



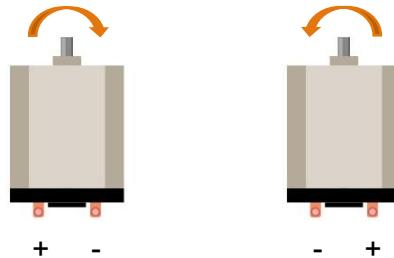
## Component knowledge

### Motor

Motor is a device that converts electrical energy into mechanical energy. Motor consists of two parts: stator and rotor. When motor works, the stationary part is stator, and the rotating part is rotor. Stator is usually the outer case of motor, and it has terminals to connect to the power. Rotor is usually the shaft of motor, and can drive other mechanical devices to run. Diagram below is a small DC motor with two pins.



When motor get connected to the power supply, it will rotate in one direction. Reverse the polarity of power supply, then motor rotates in opposite direction.



### Capacitor

The unit of capacitance(C) is farad (F).  $1F = 1000000\mu F$ ,  $1\mu F = 1000000pF$ .

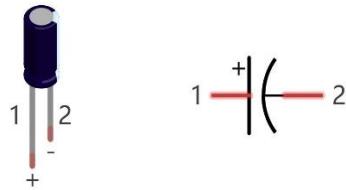
Capacitor is an energy storage device, with a certain capacitance. When capacitor voltage increases, the capacitor will be charged. And capacitor will discharge when the voltage drops. So capacitor voltage of both ends is not transient. According to this characteristic, capacitor is often used to stabilize the voltage of power supply, and filters the signal. Capacitor with large capacity can filter out low frequency signals, and small-capacity capacitor can filter out high frequency signals.

The capacitor has a non-polar capacitor and a polar capacitor. Generally, non-polar capacitor has small capacitance, and a ceramic non-polar capacitor is shown below.



For polar capacitor, it usually has larger capacitance, and an electrolytic polar capacitor of that is shown

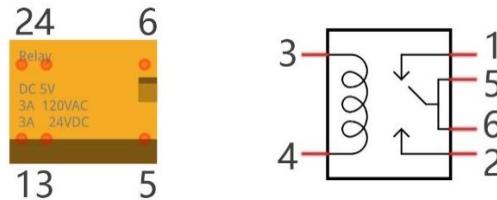
below:



When the motor rotates, it will generate noise. As the contact of coil connects and disconnects the electrode constantly, it will cause the supply voltage unstable. Thus, a small capacitor is often connected to motor to reduce the impact on power supply from motor.

### Relay

Relay is an electrical switch, which consists of coils and contacts. When the coil is energized, it will form the electromagnet, attracting contacts to close. Coils and contacts are independent of each other, so other independent circuit can be controlled by relay, through using a small current to control the circuit with large current. Diagram below is a relay with control voltage of 5V.



Pin 5 and pin 6 are connected to each other inside. When the coil pin 3 and 4 get connected to 5V power supply, pin 1 will be disconnected to pin 5&6 and pin 2 will be connected to pin 5&6. So pin 1 is called close end, pin 2 is called open end.

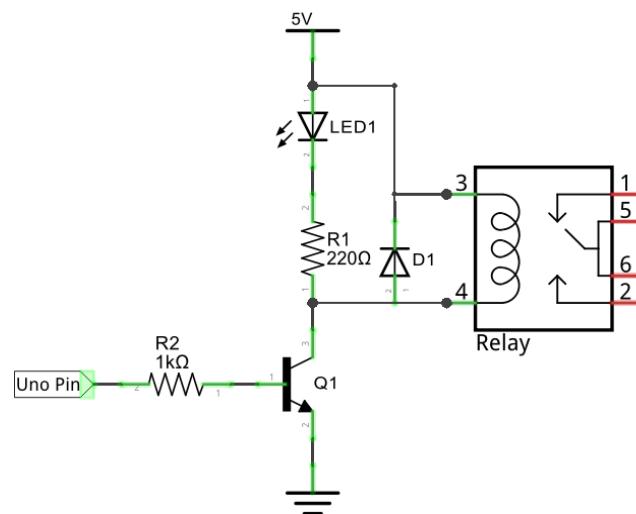
### Inductor

The unit of inductance(L) is the henry (H).  $1H=1000mH$ ,  $1mH=1000\mu H$ .

Inductor is an energy storage device that converts electrical energy into magnetic energy. Generally, it consists of winding coil, with a certain amount of inductance. Inductor will hinder the changing current passing through the inductor. When the current passing through inductor increases, it will attempt to hinder the increasing trend of current; and when the current passing through the inductor decreases, it will attempt to hinder the decreasing trend of current. So the current passing through inductor is not transient.



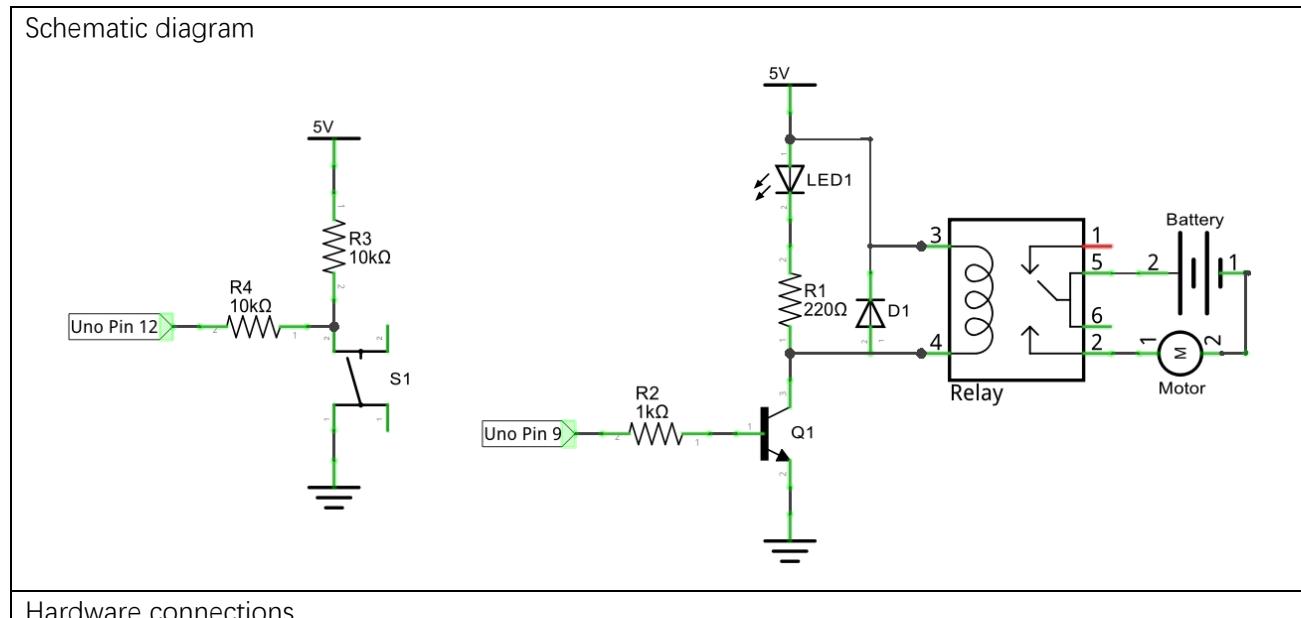
The reference circuit for relay is as follows. The coil of relay can be equivalent to inductor, when the transistor disconnects power supply of the relay, the current in the coil of the relay can't stop immediately, causing an impact on power supply. So a parallel diode will get connected to both ends of relay coil pin in reversing direction, then the current will pass through diode, avoiding the impact on power supply.



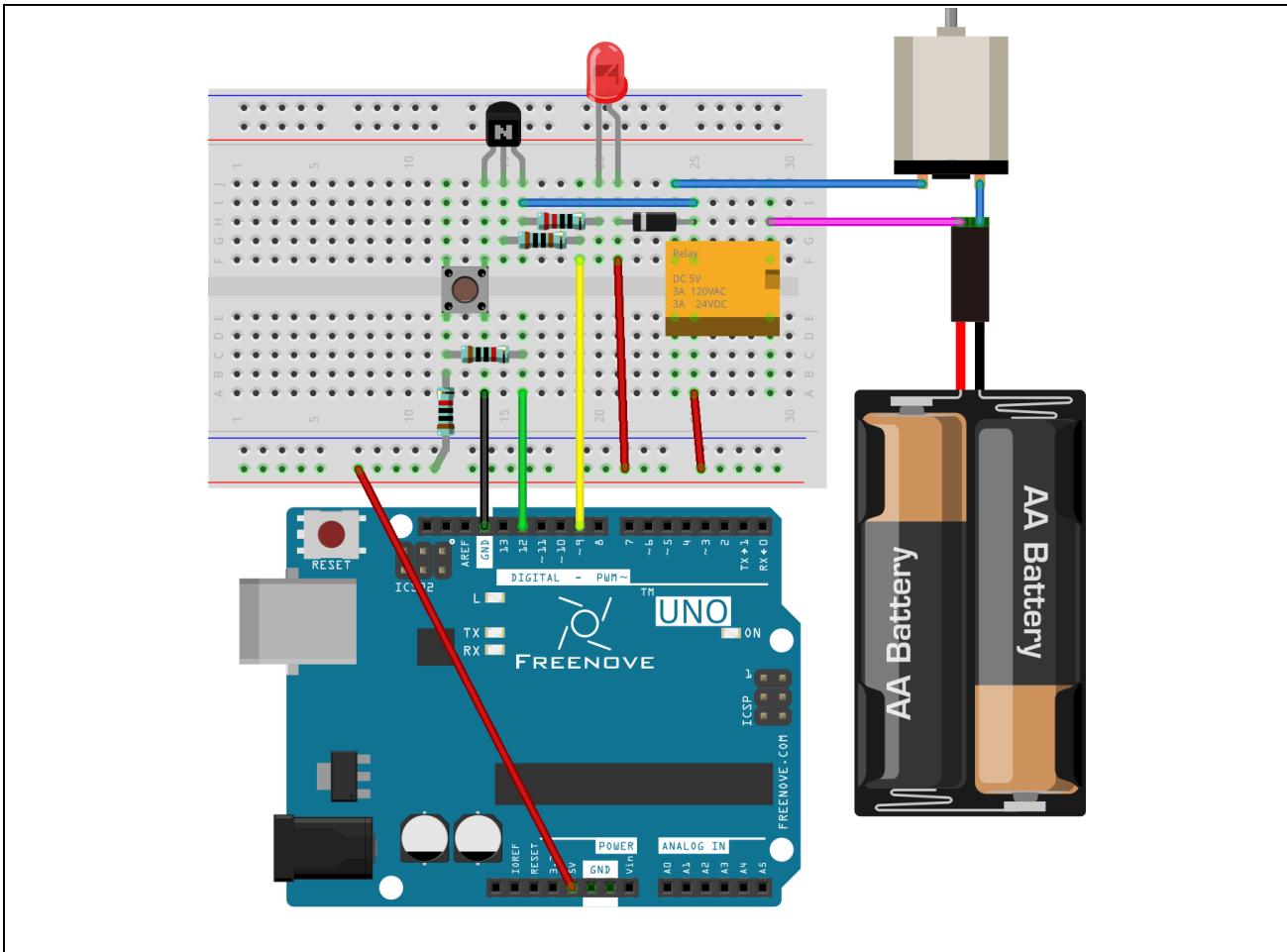
## Circuit

Use D12 of Freenove UNO to detect the state of push button, and D9 to control the relay. As motor running needs larger power, we will use two AA batteries to supply power for the motor alone.

Schematic diagram



Hardware connections



## Sketch

### Sketch 10.1.1

Now, write code to detect the state of push button. Each time you press the button, the switching status of relay will change. So we control the motor to rotate or stop in this way.

```

1 int relayPin = 9;      // the number of the relay pin
2 int buttonPin = 12;    // the number of the push button pin
3
4 int buttonState = HIGH; // Record button state, and initial the state to high level
5 int relayState = LOW;   // Record relay state, and initial the state to low level
6 int lastButtonState = HIGH; // Record the button state of last detection
7 long lastChangeTime = 0; // Record the time point for button state change
8
9 void setup() {
10     pinMode(buttonPin, INPUT); // Set push button pin into input mode
11     pinMode(relayPin, OUTPUT); // Set relay pin into output mode
12     digitalWrite(relayPin, relayState); // Set the initial state of relay into "off"
13     Serial.begin(9600);           // Initialize serial port, and set baud rate to 9600
14 }
```

```

15
16 void loop() {
17     int nowButtonState = digitalRead(buttonPin); // Read current state of button pin
18     // If button pin state has changed, record the time point
19     if (nowButtonState != lastButtonState) {
20         lastChangeTime = millis();
21     }
22     // If button state changes, and stays stable for a while, then it should have skipped the bounce area
23     if (millis() - lastChangeTime > 10) {
24         if (buttonState != nowButtonState) { // Confirm button state has changed
25             buttonState = nowButtonState;
26             if (buttonState == LOW) { // Low level indicates the button is pressed
27                 relayState = !relayState; // Reverse relay state
28                 digitalWrite(relayPin, relayState); // Update relay state
29                 Serial.println("Button is Pressed!");
30             }
31             else { // High level indicates the button is released
32                 Serial.println("Button is Released!");
33             }
34         }
35     }
36     lastButtonState = nowButtonState; // Save the state of last button
37 }
```

In this code, we used a new method to detect button state. In the loop() function, the level state of button pin is detected constantly. When the level is changed, record this time. If the level has not changed after a while, it will be considered bounce area has already been skipped. Then, judge whether the button is pressed or released according to button pin state.

First, define two variables to record the state of button and relay.

```

4 int buttonState = HIGH; // Record button state, initial the state into high level
5 int relayState = LOW; // Record relay state, initial the state into low level
```

Define a variable to record button pin state of last detection.

```
6 int lastButtonState = HIGH; // Record the button state of last detection
```

Define a variable to record the time of the last button pin changes.

```
7 long lastChangeTime = 0; // Record the time point for button state change
```

In the loop() function, the detected pin state of button will be compared with the last detected state. If it changes, record this time.

```

16 void loop() {
17     int nowButtonState = digitalRead(buttonPin); // Read current state of button pin
18     // If the state of button pin has changed, record the time point
19     if (nowButtonState != lastButtonState) {
20         lastChangeTime = millis();
21     }
```

```

...
36     lastButtonState = nowButtonState; // Save the state of last button
37 }
```

If the level stays unchanged over a period of time, it is considered bounce area has already been skipped.

```

23     if (millis() - lastChangeTime > 10) {
...
35 }
```

After the pin state stays stable, the changed state of button is confirmed, then it will be recorded for the next comparison.

```

24     if (buttonState != nowButtonState) { // Confirm button state has changed
25         buttonState = nowButtonState;
...
34 }
```

Judge whether the button is pressed or released according to button pin level, print button information to serial port, and reverse relay when the button is pressed.

```

26     if (buttonState == LOW) { // Low level indicates the button is pressed
27         relayState = !relayState; // Reverse relay state
28         digitalWrite(relayPin, relayState); // Update relay state
29         Serial.println("Button is Pressed!");
30     }
31     else { // High level indicates the button is released
32         Serial.println("Button is Released!");
33 }
```

This button detecting method does not put program into the state of delay waiting, you can increase the efficiency of code execution.

### millis()

Returns the number of milliseconds since the Arduino board began running the current program.

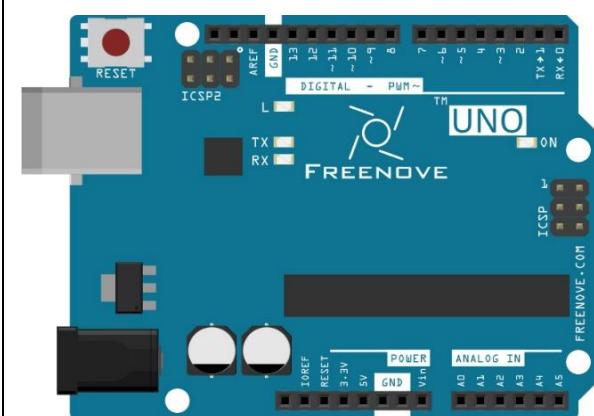
Verify and upload the code, every time you press the push button, the state of relay and motor changes once.

## Project 10.2 Control Motor by L293D

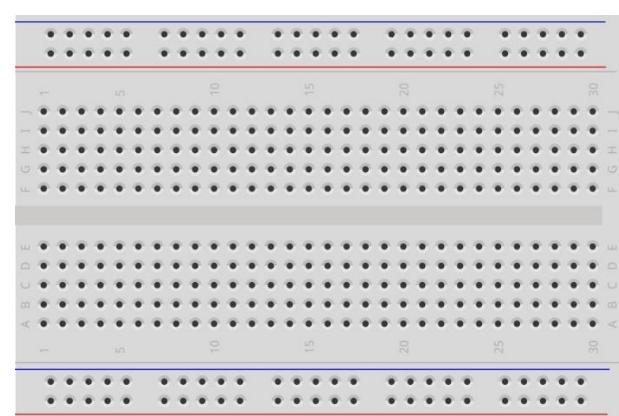
Now, we will use dedicated chip L293D to control the Motor.

### Component list

Freenove UNO x1



Breadboard x1



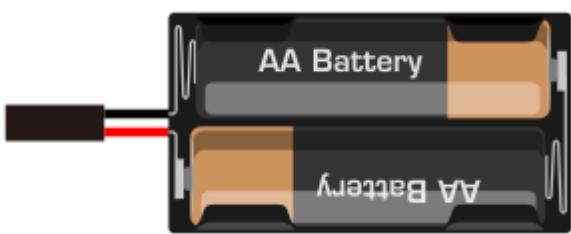
USB cable x1



Jumper M/M x10



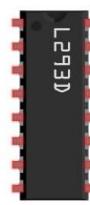
AA Battery holder x1



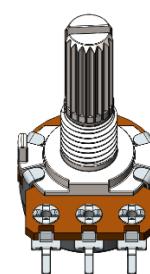
Motor x1



L293D x1



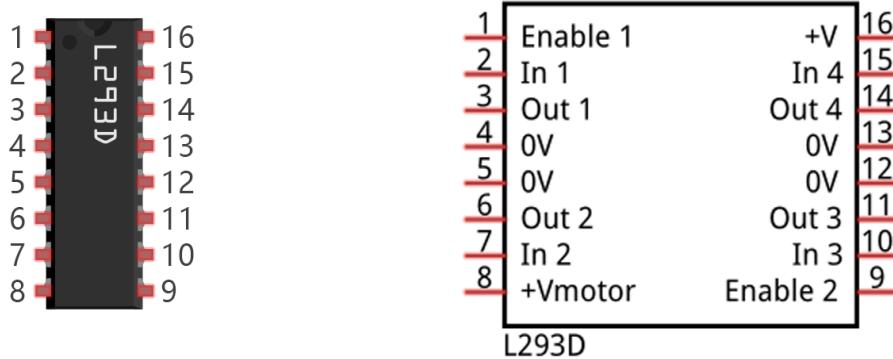
Rotary potentiometer x1



## Component knowledge

### L293D

L293D is a chip integrated with 4-channel motor drive. You can drive a unidirectional motor with 4 ports or a bi-directional motor with 2 port or a stepper motor.



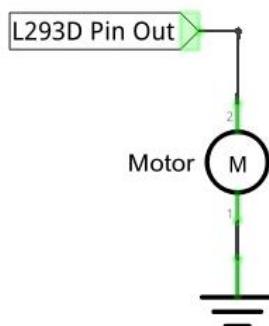
Port description of L293D module is as follows:

Pin name	Pin number	Description
In x	2, 7, 10, 15	Channel x digital signal input pin
Out x	3, 6, 11, 14	Channel x output pin, input high or low level according to In x pin, get connected to +Vmotor or 0V
Enable1	1	Channel 1 and channel 2 enable pin, high level enable
Enable2	9	Channel 3 and channel 4 enable pin, high level enable
0V	4, 5, 12, 13	Power cathode (GND)
+V	16	Positive electrode (VCC) of power supply, supply voltage 4.5~36V
+Vmotor	8	Positive electrode of load power supply, provide power supply for the Out pin x, the supply voltage is +V~36V

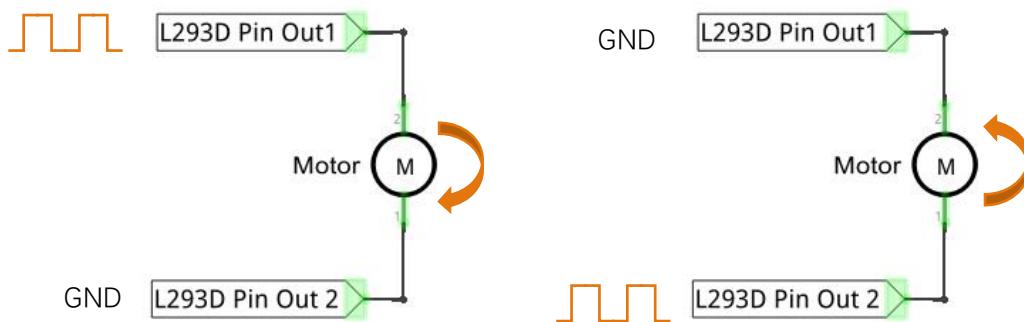
For more details, please see datasheet.

When using L293D to drive DC motor, there are usually two kinds of connection.

Following connection uses one channel, and it can control motor speed through PWM, but the motor can only rotate in one direction.



Following connection uses two channels: one channel outputs PWM wave, and another channel connects GND, so you can control the speed of motor. When these two channel signals are exchanged, the current direction of the motor can be reversed, and the motor will rotate in reverse direction. This can not only control the speed of motor, but also can control the steering of motor.

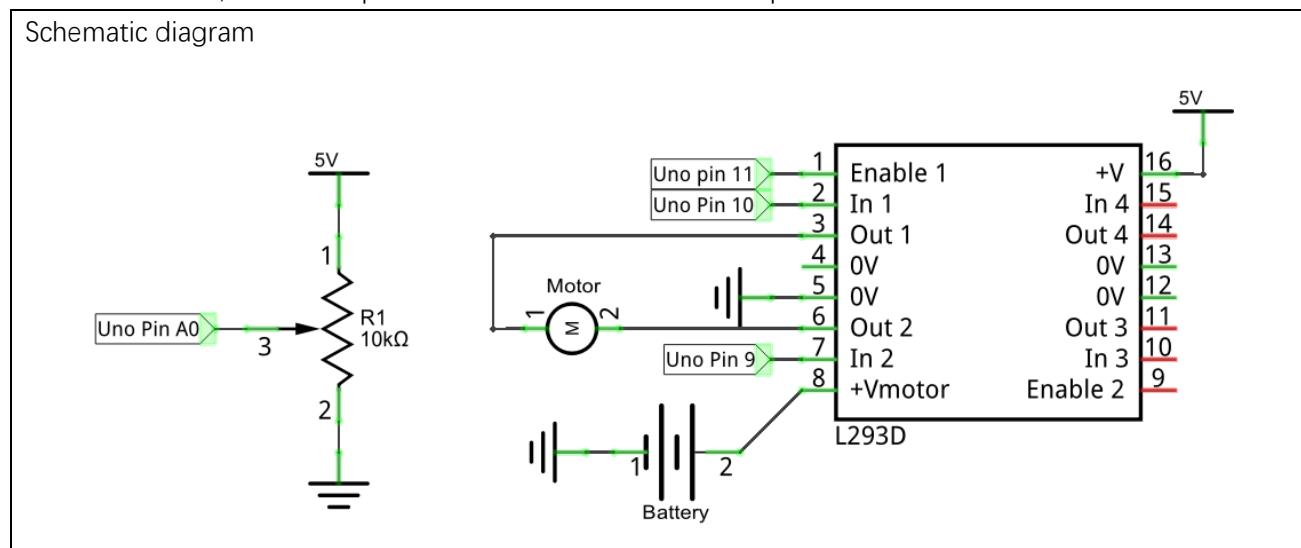


In actual use, motor is usually connected to the channel 1 and 2, output different level to in1 and in2 to control the rotation direction of the motor, and output PWM wave to Enable1 port to control the motor rotation speed. Or, get motor connected to the channel 3 and 4, output different level to in3 and in4 to control the motor's rotation direction, and output PWM wave to Enable2 pin to control the motor rotation speed.

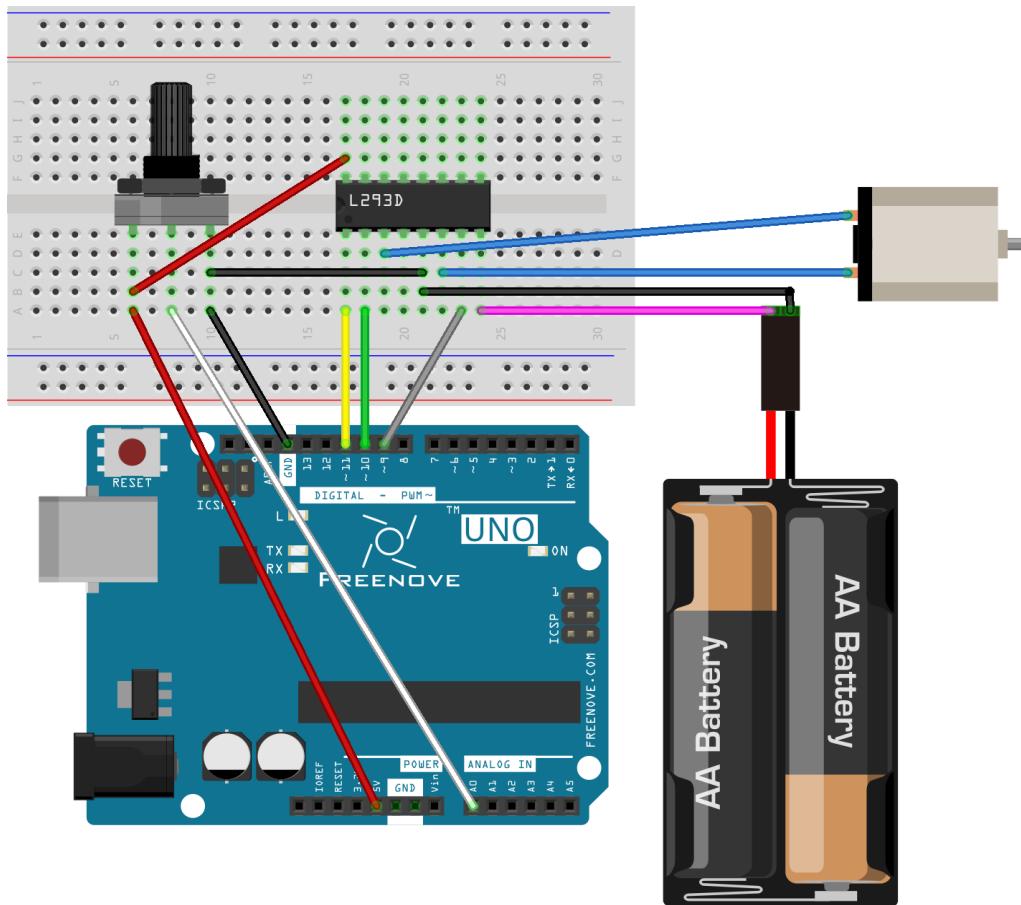
## Circuit

Use A0 of Freenove UNO to detect the voltage of rotary potentiometer; D9, D10 to control the motor's rotation direction, D11 to output PWM wave to control motor speed.

Schematic diagram



### Hardware connection



### Sketch

#### Sketch 10.2.1

Now, write the code to control speed and rotation direction of motor through rotary potentiometer. When the potentiometer stays in the middle position, motor speed will be minimum, and when deviates intermediate position, the speed will increase. Also, if the potentiometer deviates from the middle position potentiometer clockwise or counterclockwise, the rotation direction of the motor is different.

```
1 int in1Pin = 10;      // Define L293D channel 1 pin
2 int in2Pin = 9;       // Define L293D channel 2 pin
3 int enable1Pin = 11; // Define L293D enable 1 pin
4
5 boolean rotationDir; // Define a variable to save the motor's rotation direction, true and false are
6 // represented by positive rotation and reverse rotation.
7 int rotationSpeed;   // Define a variable to save the motor rotation speed
8
9 void setup() {
10 // Initialize the pin into an output mode:
11 pinMode(in1Pin, OUTPUT);
12 pinMode(in2Pin, OUTPUT);
13 pinMode(enable1Pin, OUTPUT);
14 }
15
16 void loop() {
17 int potenVal = analogRead(A0); // Convert the voltage of rotary potentiometer into digital
18 // Compare the number with value 512, if more than 512, clockwise rotates, otherwise, counter
19 // clockwise rotates
20 rotationSpeed = potenVal - 512;
21 if (potenVal > 512)
22     rotationDir = true;
23 else
24     rotationDir = false;
25 // Calculate the motor speed, the far number of deviation from the middle value 512, the faster the
26 // control speed will be
27 rotationSpeed = abs(potenVal - 512);
28 // Control the steering and speed of the motor
29 driveMotor(rotationDir, map(rotationSpeed, 0, 512, 0, 255));
30 }
31
32 void driveMotor(boolean dir, int spd) {
33 // Control motor rotation direction
34 if (rotationDir) {
35     digitalWrite(in1Pin, HIGH);
36     digitalWrite(in2Pin, LOW);
37 }
38 else {
39     digitalWrite(in1Pin, LOW);
40     digitalWrite(in2Pin, HIGH);
41 }
42 // Control motor rotation speed
43 analogWrite(enable1Pin, constrain(spd, 0, 255));
44 }
```

In the code, we write a function to control the motor, and control the speed and steering through two parameters.

```

29 void driveMotor(boolean dir, int spd) {
30     // Control motor rotation direction
31     if (rotationDir) {
32         digitalWrite(in1Pin, HIGH);
33         digitalWrite(in2Pin, LOW);
34     }
35     else {
36         digitalWrite(in1Pin, LOW);
37         digitalWrite(in2Pin, HIGH);
38     }
39     // Control motor rotation speed
40     analogWrite(enable1Pin, constrain(spd, 0, 255));
41 }
```

In the loop () function, detect the digital value of rotary potentiometer, and convert that into the motor speed and steering through calculation.

```

15 void loop() {
16     int potenVal = analogRead(A0); // Convert the voltage of rotary potentiometer into digital
17     // Compare the digital number with middle value 512, if more than 512, clockwise rotates, otherwise,
18     counter clockwise rotates
19     rotationSpeed = potenVal - 512;
20     if (potenVal > 512)
21         rotationDir = true;
22     else
23         rotationDir = false;
24     // Calculate the motor speed, the far number of deviation from the middle value 512, the faster the
25     control speed will be
26     rotationSpeed = abs(potenVal - 512);
27     // Control the steering and speed of the motor
28     driveMotor(rotationDir, map(rotationSpeed, 0, 512, 0, 255));
29 }
```

**abs(x)**

Computes the absolute value of a number.

Verify and upload the code, turn the shaft of rotary potentiometer, then you can see the change of the motor speed and direction.

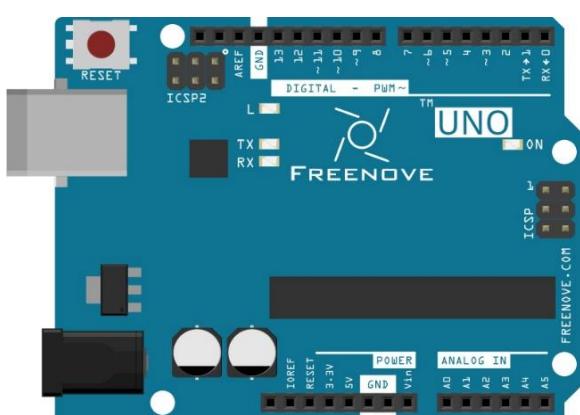
# Chapter 11 Servo

Before, we have used Freenove UNO and L293D module to control the motor speed and steering. Now, we will use another motor, servo, which can rotate to a certain angle.

## Project 11.1 Servo Sweep

First, let's get the servo to rotate.

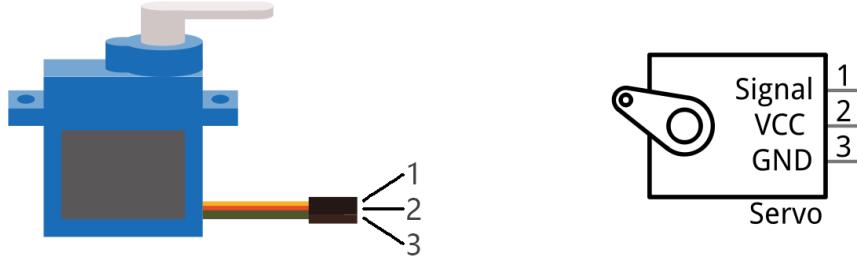
### Component list

Freenove UNO x1	USB cable x1
	
Jumper M/M x3	Servo x1

## Component knowledge

### Servo

Servo is an auto-control system, consisting of DC motor, reduction gear, sensor and control circuit. Usually, it can rotate in the range of 180 degrees. Servo can output larger torque and is widely used in model airplane, robot and so on. It has three lines, including two for electric power line positive (2-VCC, red), negative (3-GND, brown), and the signal line (1-Signal, orange).



We use 50Hz PWM signal with a duty cycle in a certain range to drive the servo. The lasting time 0.5ms-2.5ms of PWM single cycle high level corresponds to the servo angle 0 degrees - 180 degree linearly. Part of the corresponding values are as follows:

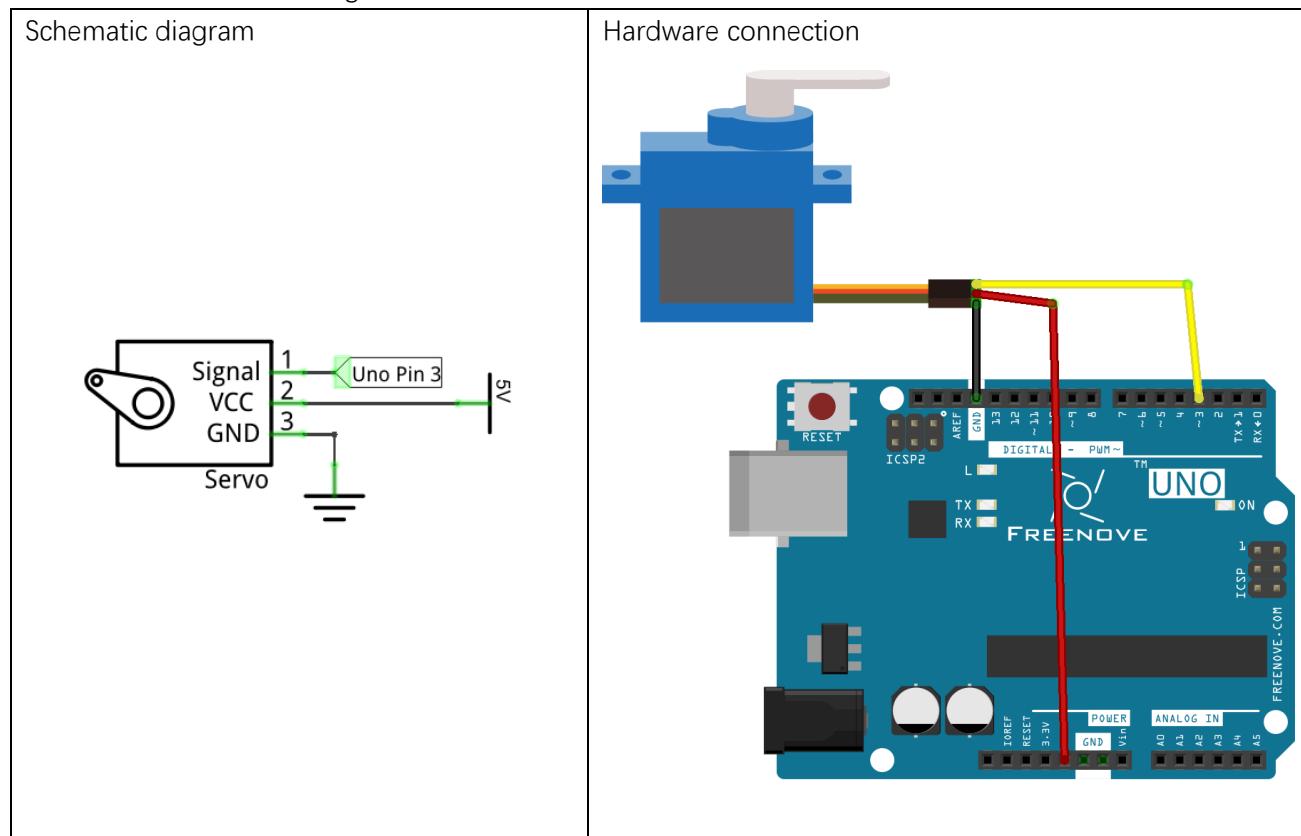
High level time	Servo angle
0.5ms	0 degree
1ms	45 degree
1.5ms	90 degree
2ms	135 degree
2.5ms	180 degree

When you change the servo signal, servo will rotate to the designated position.

## Circuit

Use D3 of Freenove UNO to drive the servo.

Pay attention to the color of servo lead wire: VCC (red), GND (brown), and signal line (orange). The wrong connection can cause damage to servo.



## Sketch

### Sketch 11.1.1

Now, write the code to control servo, making it sweep in the motion range continuously.

```

1 #include <Servo.h>
2
3 Servo myservo; // create servo object to control a servo
4
5 int pos = 0; // variable to store the servo position
6 int servoPin = 3; // define the pin of servo signal line
7
8 void setup() {
9     myservo.attach(servoPin); // attaches the servo on servoPin to the servo object
10 }
11

```

```

12 void loop() {
13     for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
14         // in steps of 1 degree
15         myservo.write(pos);           // tell servo to go to position in variable "pos"
16         delay(15);                  // waits 15ms for the servo to reach the position
17     }
18     for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
19         myservo.write(pos);           // tell servo to go to position in variable "pos"
20         delay(15);                  // waits 15ms for the servo to reach the position
21     }
22 }
```

Servo uses the Servo library, like the following reference to Servo library:

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

Servo library provides the Servo class that controls it. Different from previous Serial class, the Servo class must be instantiated before you use:

```
3 Servo myservo;           // create servo object to control a servo
```

The code above defines an object of Servo type, myservo.

### Servo Class

Servo class must be instantiated when used, that is, define an object of Servo type, for example:

```
Servo myservo;
```

Most Arduino control board can define 12 objects of Servo type, namely, it can control up to 12 servos.

The function commonly used in the servo class is as follows:

`myservo.attach(pin):` Initialize the servo, the parameter is the port connected to servo signal line;

`myservo.write(angle):` Control servo to rotate to the specified angle; parameter here is to specify the angle.

After the Servo object is defined, it can refer to the function, such as initializing the servo:

```
9 myservo.attach(servoPin); // attaches the servo on servoPin to the servo object
```

After initializing the servo, you can control the servo to rotate to a specific angle:

```
15 myservo.write(pos);           // tell servo to go to position in variable "pos"
```

In the `loop ()` function, we use the loop to control the servo to rotate from 0 degrees to 180 degrees, and then from 180 degrees to 0 degrees, then repeat the cycle all the time.

Verify and upload the code, the servo starts to sweep continuously.

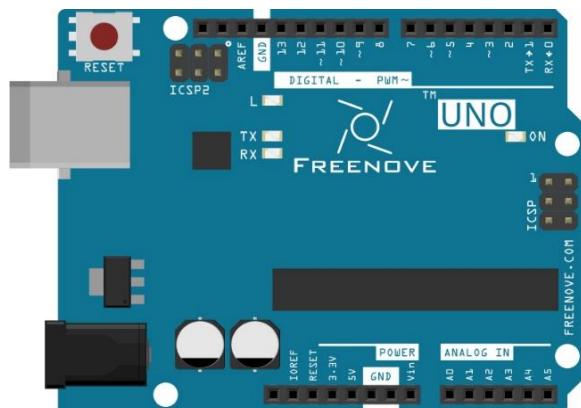


## Project 11.2 Control Servo by Potentiometer

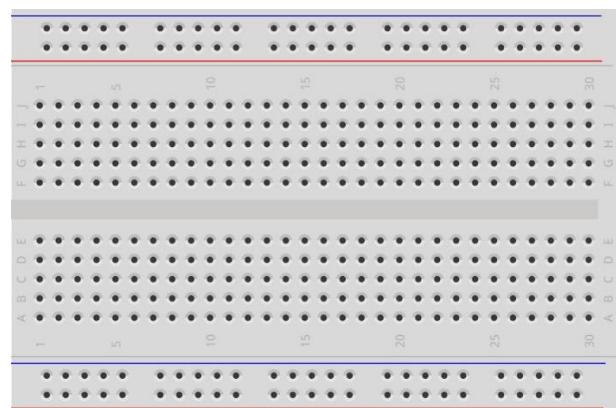
In the last section, we've made the servo sweep continuously. Now, we will use potentiometer to control the servo angle.

### Component list

Freenove UNO x1



Breadboard x1



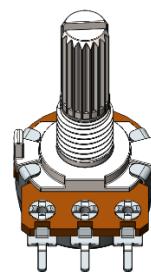
USB cable x1



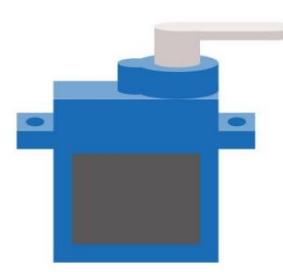
Jumper M/M x6



Rotary potentiometer x1

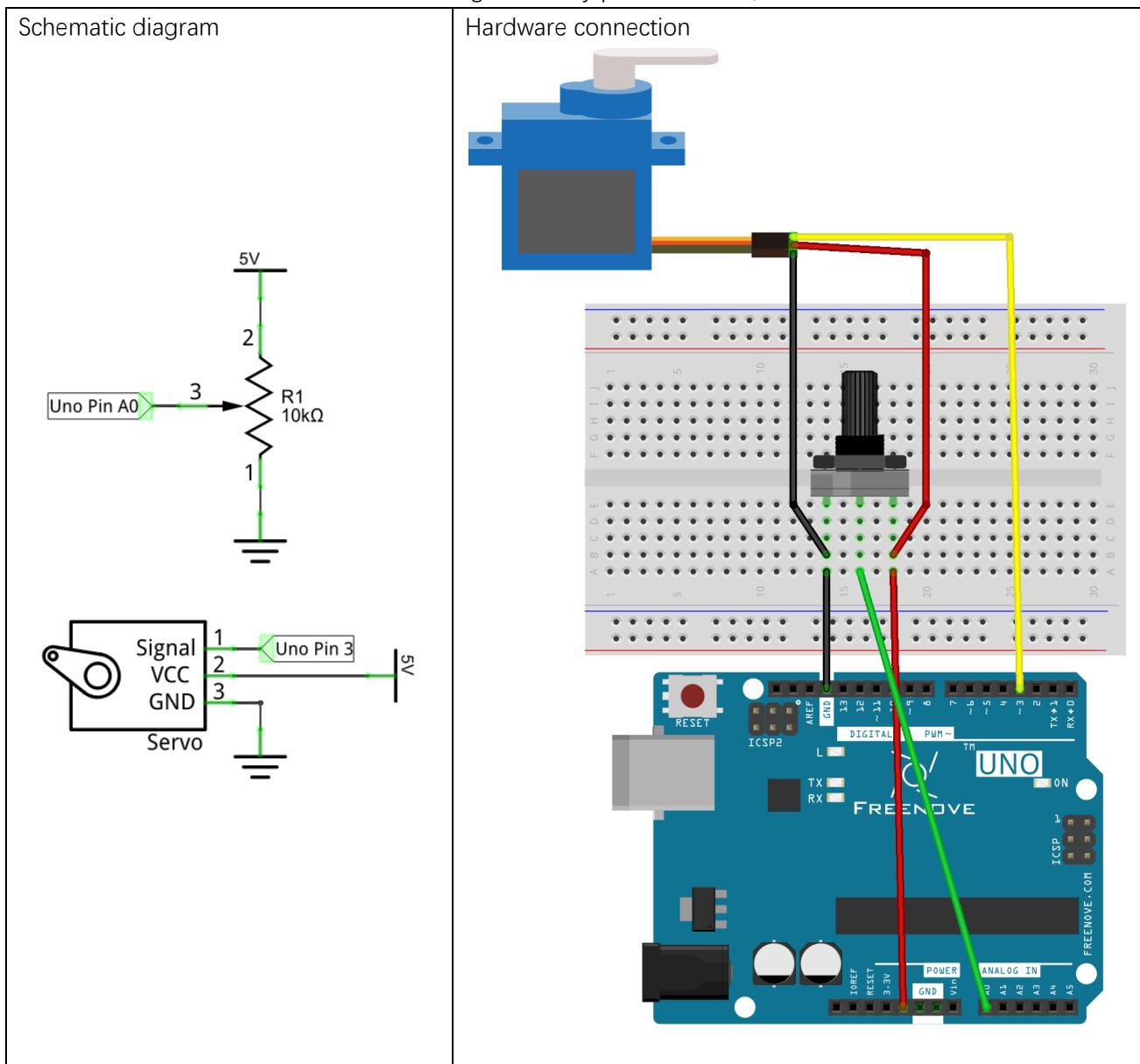


Servo x1



## Circuit

Use A0 of Freenove UNO to detect the voltage of rotary potentiometer, and D3 to drive the servo.



## Sketch

### Sketch 11.2.1

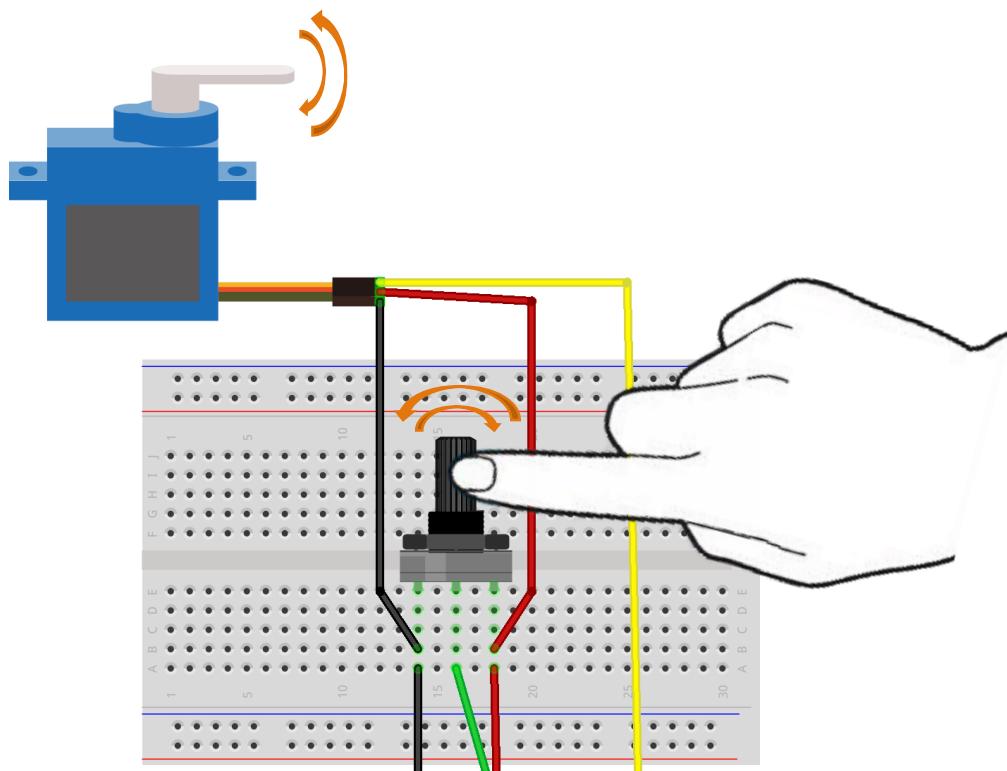
Now, write the code to detect the voltage of rotary potentiometer, and control servo to rotate to a different angle according to that.

```

1 #include <Servo.h>
2
3 Servo myservo;           // create servo object to control a servo
4
5 int servoPin = 3;        // define the pin of servo signal line
6 int potPin = 0;           // analog pin used to connect the potentiometer
7 int potVal;              // variable to read the potValue from the analog pin
8
9 void setup() {
10   myservo.attach(servoPin); // attaches the servo on servoPin to the servo object
11 }
12
13 void loop() {
14   potVal = analogRead(potPin);      // reads the potValue of the potentiometer
15   potVal = map(potVal, 0, 1023, 0, 180); // scale it to use it with the servo
16   myservo.write(potVal);           // sets the servo position
17   delay(15);                    // waits for the servo to get there
18 }
```

In the code, we obtain the ADC value of A0 pin, and map it to the servo angle.

Verify and upload the code, turn the potentiometer shaft, then the servo will rotate to the corresponding angle.



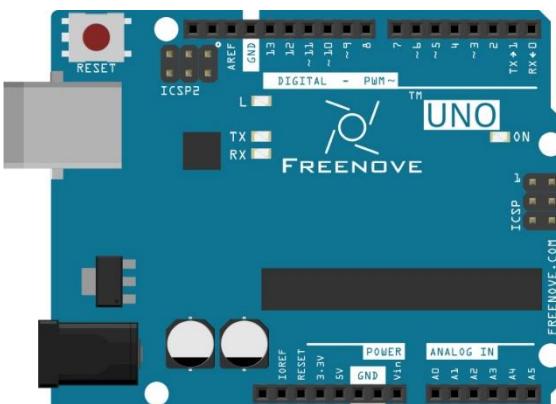
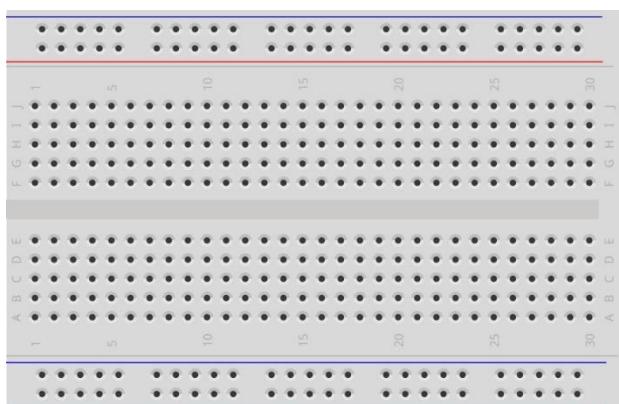
# Chapter 12 Temperature Sensor

Before, we have used Freenove UNO and photoresistor to detect the intensity of light. Now, we will learn to use the temperature sensor.

## Project 12.1 Detect the Temperature

We will use a thermistor to detect the ambient temperature.

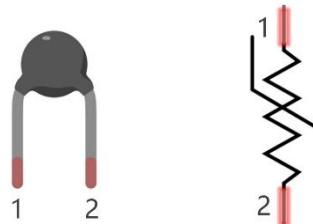
### Component list

Freenove UNO x1	Breadboard x1
	
USB cable x1	Thermistor x1
	
Jumper M/M x3	Resistor 10kΩ x1
	

## Component knowledge

### Thermistor

Thermistor is a temperature sensitive resistor. When the temperature changes, resistance of thermistor will change. With this feature, we can use thermistor to detect temperature intensity. Thermistor and symbol are as follows.



The relationship between resistance value and temperature of thermistor is:

$$R_t = R \cdot \exp[B \cdot (1/T_2 - 1/T_1)]$$

Where:

$R_t$  is the thermistor resistance under  $T_2$  temperature;

$R$  is the nominal resistance of thermistor under  $T_1$  temperature;

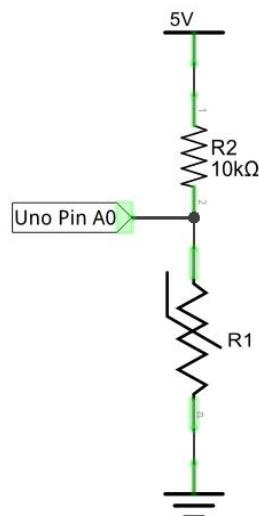
$\exp[n]$  is nth power of E;

B is for thermal index;

$T_1, T_2$  is Kelvin temperature (absolute temperature). Kelvin temperature = 273.15 + Celsius temperature.

Parameters of the thermistor we use is:  $B=3950$ ,  $R=10k$ ,  $T_1=25$ .

The circuit connection method of the thermistor is similar to photoresistor, like the following method:

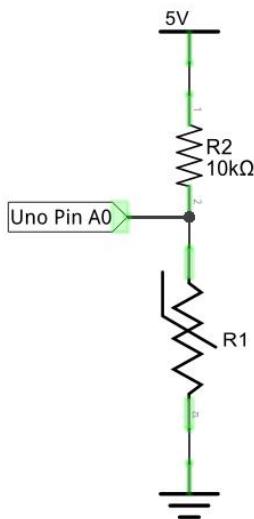


We can use the value measured by the analog pin of UNO to obtain resistance value of thermistor, and then can use the formula to obtain the temperature value.

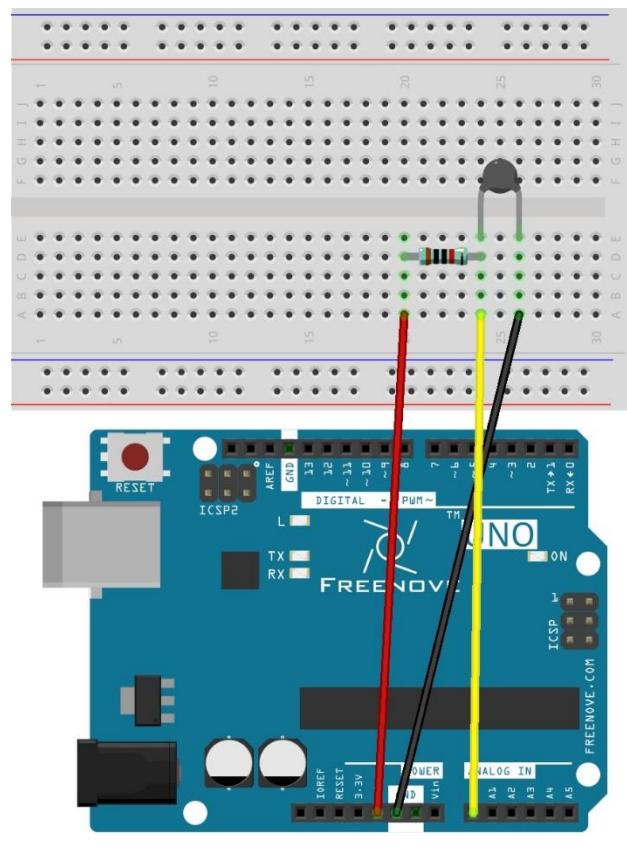
## Circuit

Use A0 port on Freenove UNO to detect the voltage of thermistor.

Schematic diagram



Hardware connection



## Sketch

### Sketch 12.1.1

Now, write the code to detect the voltage value of thermistor, calculate the temperature value, and send it to Serial Monitor window.

```

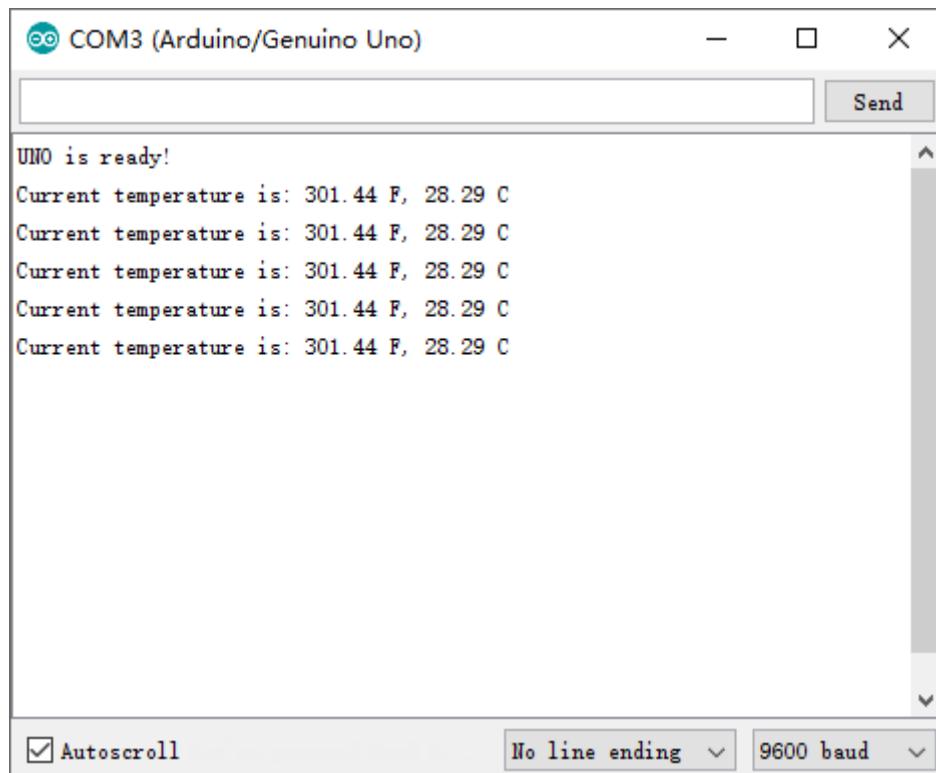
1 void setup() {
2     Serial.begin(9600);           // Initialize the serial port, set the baud rate
3     into 9600
4     Serial.println("UNO is ready!"); // Print the string "UNO is ready!"
5 }
6
7 void loop() {
8     // Convert analog value of A0 port into digital value
9     int adcVal = analogRead(A0);

```

```
9 // Calculate voltage
10 float v = adcVal * 5.0 / 1024;
11 // Calculate resistance value of thermistor
12 float Rt = 10 * v / (5 - v);
13 // Calculate temperature (Kelvin)
14 float tempK = 1 / (log(Rt / 10) / 3950 + 1 / (273.15 + 25));
15 // Calculate temperature (Celsius)
16 float tempC = tempK - 273.15;
17
18 // Send the result to computer through serial port
19 Serial.print("Current temperature is: ");
20 Serial.print(tempK);
21 Serial.print(" K, ");
22 Serial.print(tempC);
23 Serial.println(" C");
24 delay(500);
25 }
```

In the code, we obtain the ADC value of A0 pin, and convert it into temperature value, then send it to the serial port.

Verify and upload the code, open the Monitor Serial, then you will see the temperature value sent from UNO board.



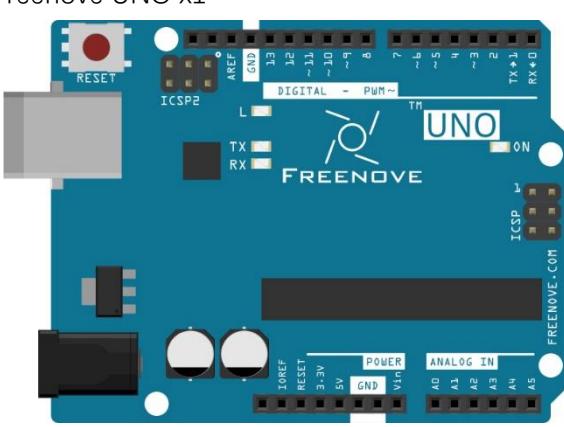
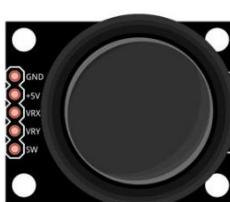
# Chapter 13 Joystick

In the previous chapter, we have learned how to use rotary potentiometer. Now, let's learn a new electronic module Joystick which working on the same principle as rotary potentiometer.

## Project 13.1 Joystick

We will use the serial port to get Joystick data.

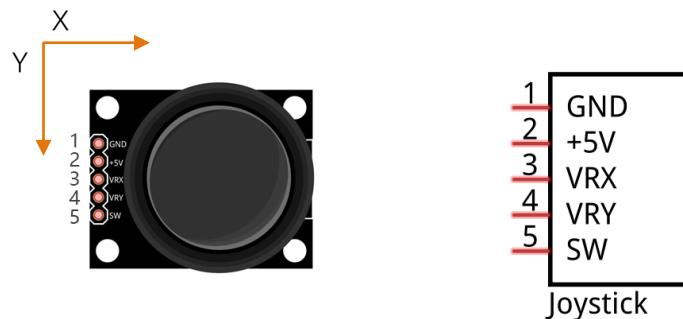
## Component list

Freenove UNO x1	Joystick x1
 A blue Freenove Arduino Uno R3 board with various pins, connectors, and a microcontroller chip.	 A black circular joystick module with four pins labeled GND, 5V, VRY, and SW.
USB cable x1	Jumper F/M x5

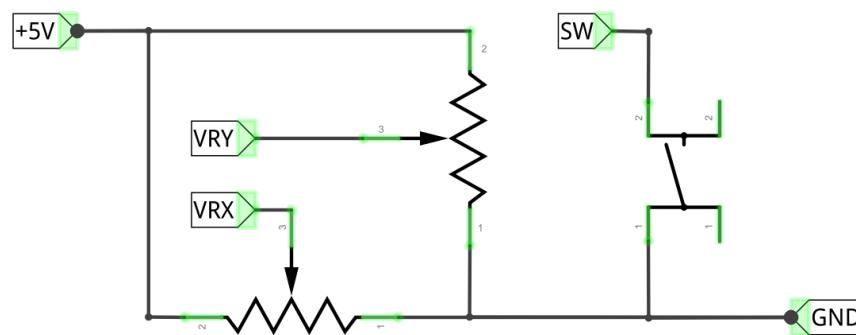
## Component knowledge

### Joystick

Joystick is a kind of sensor used with your fingers, which is widely used in gamepad and remote controller. It can shift in direction Y or direction X at the same time. And it can also be pressed in direction Z.

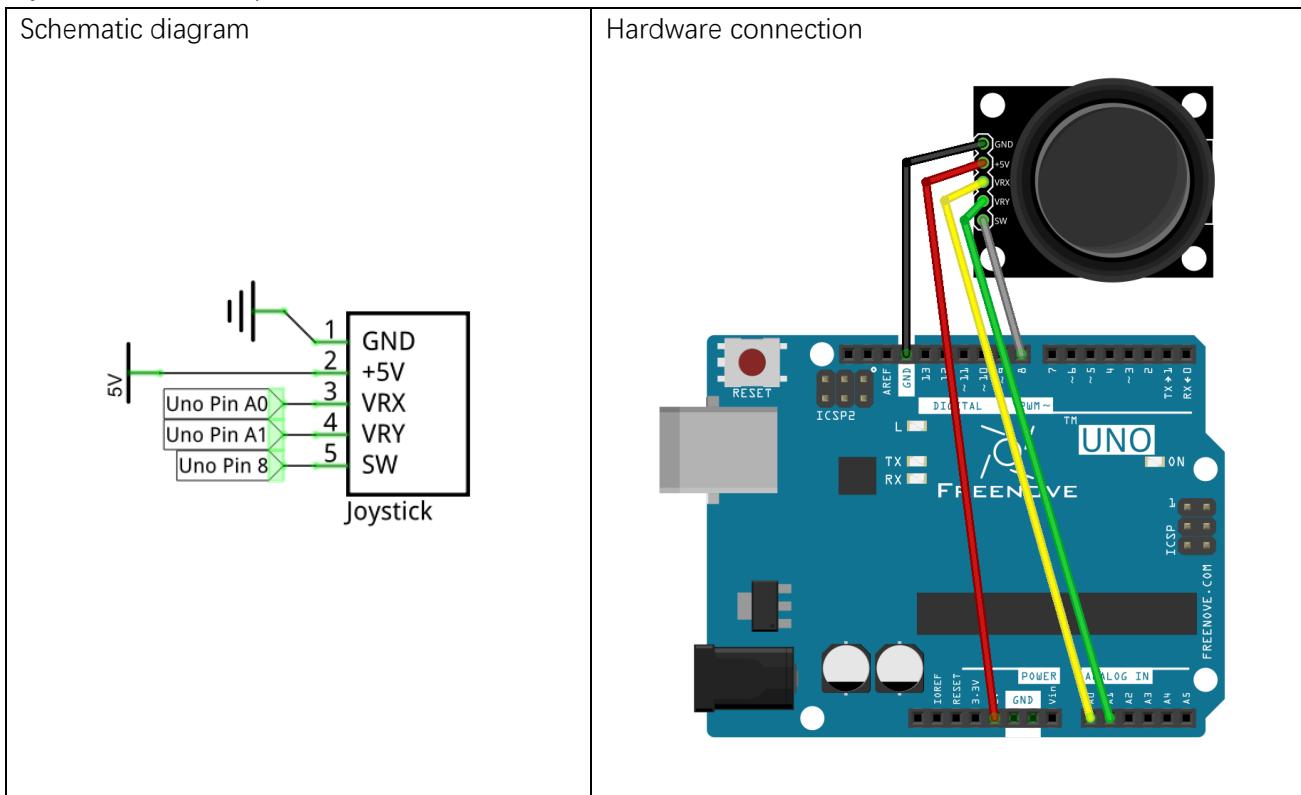


Two rotary potentiometers inside the joystick are set to detect the shift direction of finger, and a push button in vertical direction is set to detect the action of pressing.



## Circuit

Use A0 and A1 port on Freenove UNO board to detect the voltage value of two rotary potentiometers inside Joystick, and use D8 port to detect the vertical button.



## Sketch

### Sketch 13.1.1

Now write the sketch to detect the voltage value of these two rotary potentiometers and the state of the button in vertical direction, then sent the data to Serial Monitor window.

```

1 int xAxisPin = 0;           // define X pin of Joystick
2 int yAxisPin = 1;           // define Y pin of Joystick
3 int zAxisPin = 8;           // define Z pin of Joystick
4 int xVal, yVal, zVal;       // define 3 variables to store the values of 3 direction
5
6 void setup() {
7     pinMode(zAxisPin, INPUT_PULLUP); // initialize the port to pull-up input
8     Serial.begin(9600);           // initialize the serial port with baud rate 9600
9     Serial.println("UNO is ready!"); // print the string "UNO is ready!"
10 }
11

```

```

12 void loop() {
13     // read analog value in XY axis
14     xVal = analogRead(xAxisPin);
15     yVal = analogRead(yAxisPin);
16     // read digital value of switch in Z axis
17     zVal = digitalRead(zAxisPin);
18     //print the data read above
19     Serial.print("X : ");
20     Serial.print(xVal);
21     Serial.print("\t Y : ");
22     Serial.print(yVal);
23     Serial.print("\t Z : ");
24     Serial.println(zVal);
25     delay(200);
26 }
```

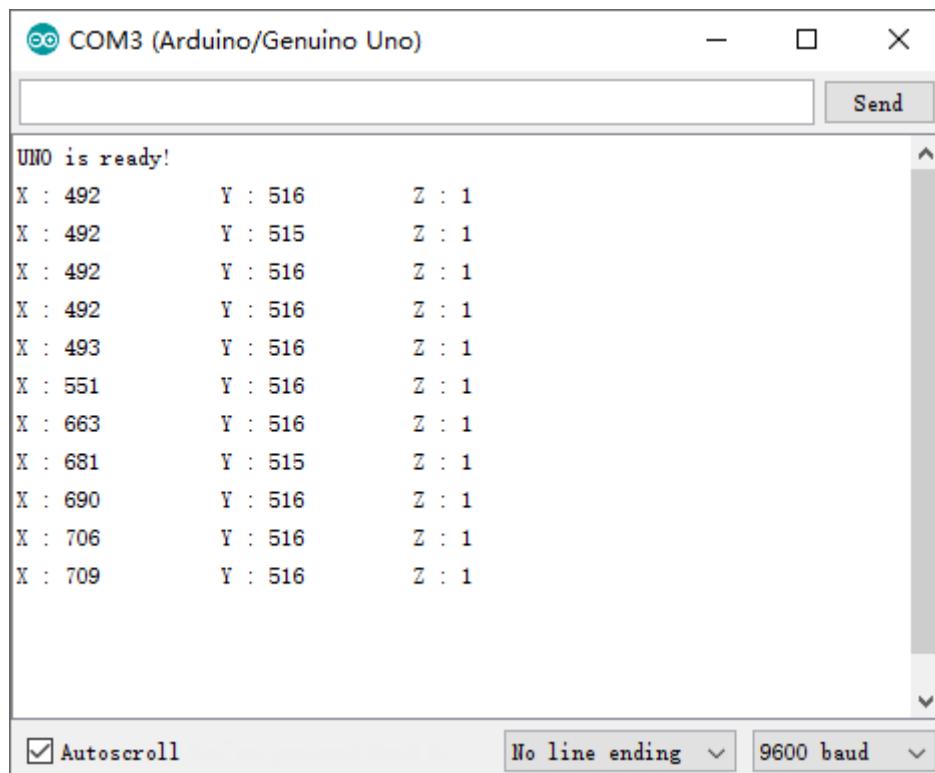
In the code, we get the ADC value of A0, A1 pin and the state of button, then sent the data to serial port.

### INPUT\_PULLUP

Set the port to INPUT\_PULLUP mode, which is equivalent to configuring the port to INPUT mode, then connect a resistor with high resistance value to VCC behind the port.

Push button of joystick is left hanging when it is not pressed (connected to no circuits with certain voltage value). The results of push button port read by UNO are not fixed. So we can set this port for INPUT\_PULLUP mode. Then when the push button is not pressed, the state of the port is high. But if it is pressed, the state turns into low level.

Verify and upload the code, open the Serial Monitor, then you can see the Joystick state value sent by UNO board. Shift and press the rocker of joystick with your finger, then you can see the change of value.



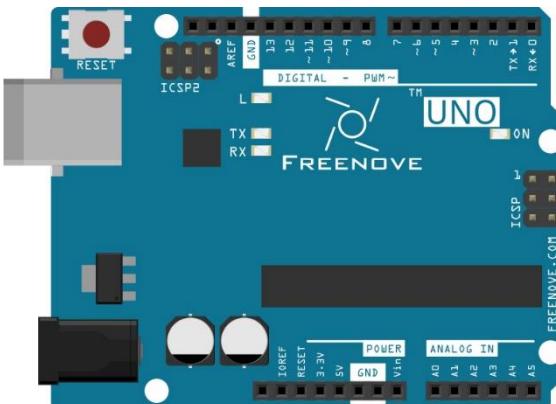
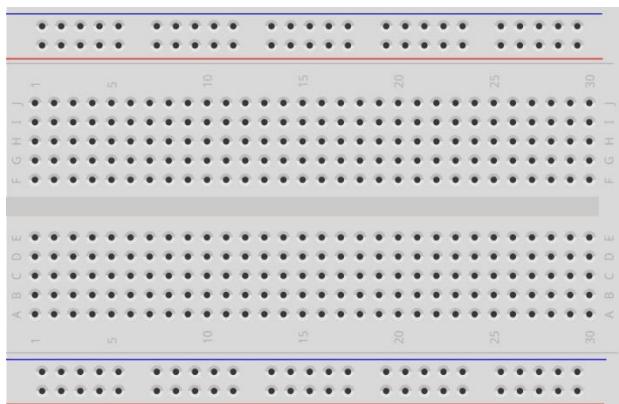
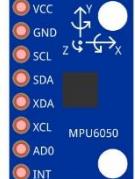
# Chapter 14 Acceleration sensor

In the previous chapter, we have learned sensors which are used to detect light or temperature. Now we will learn the sensor which can detect acceleration.

## Project 14.1 Acceleration Detection

We will use serial port to get the data of acceleration sensor MPU6050 module.

### Component list

Freenove UNO x1	Breadboard x1
	
USB cable x1	MPU6050 module x1
	
Jumper M/M x4	
	

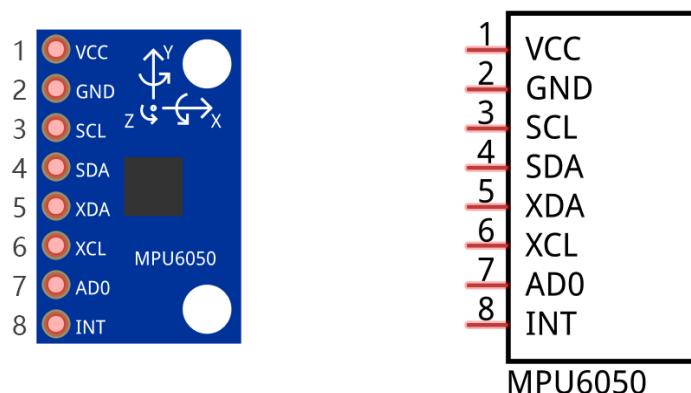
## Component knowledge

### I2C communication

I2C(Inter-Integrated Circuit) is a two-wire serial communication mode, which can be used to connection of micro controller and its peripheral equipment. Devices using I2C communication must be connected to the serial data (SDA) line, and serial clock (SCL) line (called I2C bus). Each device has a unique address and can be used as a transmitter or receiver to communicate with devices connected to the bus.

### MPU6050

MPU6050 is a sensor which integrates 3 axis accelerometer, 3 axes angular accelerometer (called gyroscope) and 1 digital attitude processor (DMP). The range of accelerometer and gyroscope of MPU6050 can be changed. A digital temperature sensor with wide range and high precision is integrated within it for temperature compensation, and the temperature value can be also read out. The MPU6050 module follows I2C communication protocol and the default address is 0x68.



The port description of the MPU6050 module is as follows:

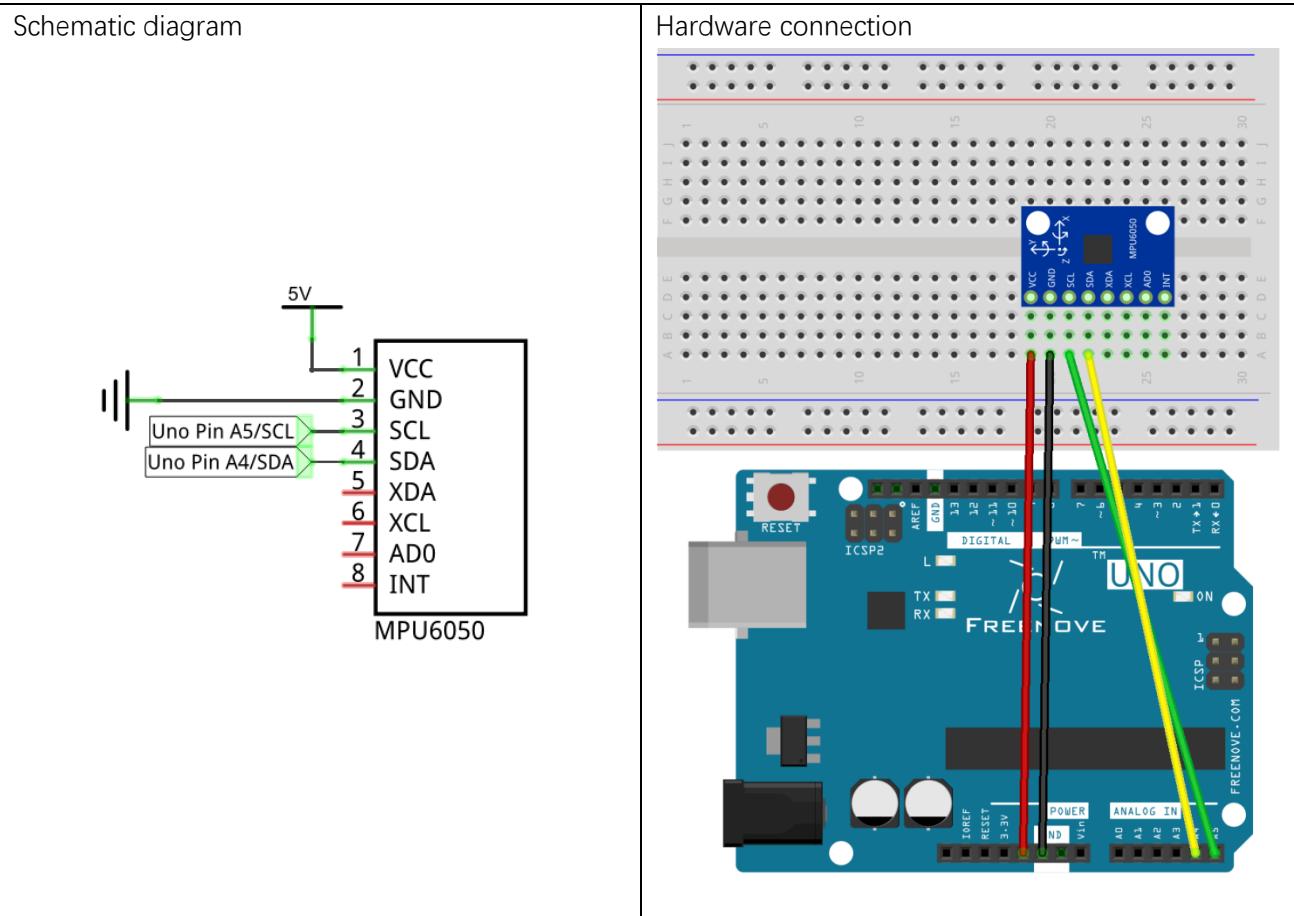
Pin name	Pin number	Description
VCC	1	Positive pole of power supply with voltage 5V
GND	2	Negative pole of power supply
SCL	3	I2C communication clock pin
SDA	4	I2C communication data pin
XDA	5	I2C host data pin which can be connected to other devices.
XCL	6	I2C host clock pin which can be connected to other devices.
AD0	7	I2C address bit control pin. Low level: the device address is 0x68 High level: the device address is 0x69
INT	8	Output interrupt pin

For more detail, please refer to dataheet.

MPU6050 is widely used in the field of balancing vehicles, aircraft and others which need to control the attitude.

## Circuit

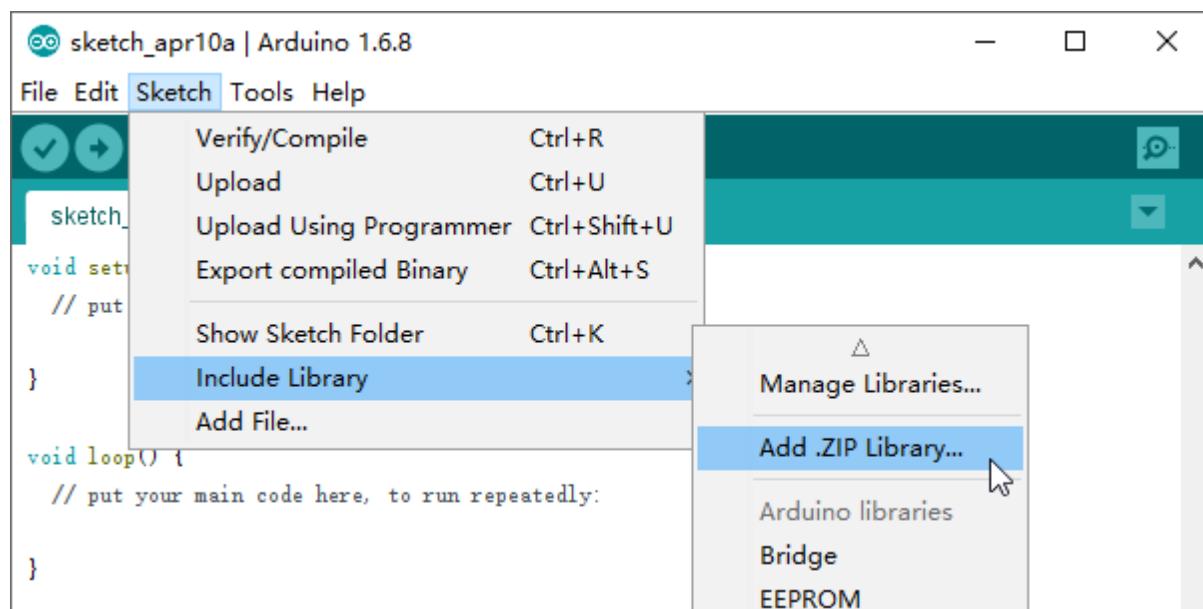
Use A4/SDA, A5/SCL port on Freenove UNO board to communicate with MPU6050 module.



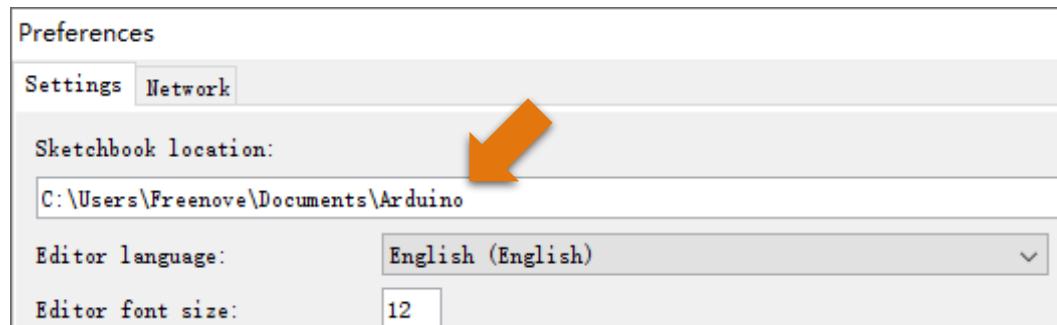
## Sketch

### Sketch 14.1.1

Before writing code, we need to import the library file you want to use. These library files are contributed by other open source developers, which is not previously installed in Arduino Software. Click Add.ZIP Library, and then add the I2Cdev.zip and MPU6050.zip under Libraries folder.



When these libraries are added, you can locate them in the libraries under Sketchbook location in the File-Preference window. You can view the source code of these library files to understand their specific usage.



Now write sketch to communicate with the MPU6050 module and send the captured data to Serial Monitor window.

```

1 // Reference the library to be used by MPU6050
2 #include "Wire.h"
3 #include "I2Cdev.h"
4 #include "MPU6050.h"
5
6 MPU6050 accelgyro;           // Construct a MPU6050 object using the default address
7 int16_t ax, ay, az;          // define acceleration values of 3 axes
8 int16_t gx, gy, gz;          // Define variables to save the values in 3 axes of gyroscope

```

```
9 #define LED_PIN 13           // the number of the LED pin
10 bool blinkState = false;    // Define a variable to save the state of LED
11
12 void setup() {
13     Serial.begin(9600);      // initialize the serial port, and baud rate is set to 9600
14     Serial.println("Initializing I2C devices..."); 
15     Wire.begin();            // initialize I2C
16     accelgyro.initialize();   // initialize MPU6050
17     Serial.println("Testing device connections..."); 
18     // verify connection
19     if (accelgyro.testConnection()) {
20         Serial.println("MPU6050 connection successful");
21     }
22     else {
23         Serial.println("MPU6050 connection failed");
24         while (1);
25     }
26     // when you need to calibrate the gravity acceleration, you can set the offset here and
27     // eliminate the note
28     // accelgyro.setXAccelOffset(-1200);
29     // accelgyro.setYAccelOffset(-2500);
30     // accelgyro.setZAccelOffset(1988);
31     Serial.print("X.Y.Z offset :\t");
32     Serial.print(accelgyro.getXAccelOffset()); Serial.print("\t");
33     Serial.print(accelgyro.getYAccelOffset()); Serial.print("\t");
34     Serial.print(accelgyro.getZAccelOffset()); Serial.print("\n");
35     // initialize LED port
36     pinMode(LED_PIN, OUTPUT);
37 }
38
39 void loop() {
40     // read raw accel/gyro measurements from device
41     accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
42     // display tab-separated accel/gyro x/y/z values
43     Serial.print("a/g:\t");
44     Serial.print(ax); Serial.print("\t");
45     Serial.print(ay); Serial.print("\t");
46     Serial.print(az); Serial.print("\t");
47     Serial.print(gx); Serial.print("\t\t");
48     Serial.print(gy); Serial.print("\t\t");
49     Serial.println(gz);
50     // converted acceleration unit to g and the gyroscope unit to dps (degree per second)
51     // according to the sensitivity
52     Serial.print("a/g:\t");
```

```

51 Serial.print((float)ax / 16384); Serial.print("g\t");
52 Serial.print((float)ay / 16384); Serial.print("g\t");
53 Serial.print((float)az / 16384); Serial.print("g\t");
54 Serial.print((float)gx / 131); Serial.print("d/s \t");
55 Serial.print((float)gy / 131); Serial.print("d/s \t");
56 Serial.print((float)gz / 131); Serial.print("d/s \n");
57 delay(300);
58 // blink LED to indicate activity
59 blinkState = !blinkState;
60 digitalWrite(LED_PIN, blinkState);
61 }

```

We reference the libraries designed for the I2C bus and the MPU6050 to manipulate the MPU6050.

```

2 #include "Wire.h"
3 #include "I2Cdev.h"
4 #include "MPU6050.h"

```

MPU6050 library provides MPU6050 class to manipulate the MPU6050, and it is necessary to instantiate an object of class before using it.

```

6 MPU6050 accelgyro;           // Construct a MPU6050 object using the default address

```

First initialize the I2C bus, and then initialize the MPU6050.

```

15 Wire.begin();           // initialize I2C
16 accelgyro.initialize(); // initialize MPU6050

```

Then do a test to confirm whether MPU6050 is connected to the I2C bus and print the related information in serial port.

```

19 if (accelgyro.testConnection()) {
20     Serial.println("MPU6050 connection successful");
21 }
22 else {
23     Serial.println("MPU6050 connection failed");
24     while (1);
25 }

```

If you want to make the results more close to the actual situation, you can adjust the offset of MPU6050 before using it. You can refer to the MPU6050 library files for more details about setting offset. If there are no strict requirements, this step can also be ignored.

```

26 // when you need to calibrate the gravity acceleration, you can set the offset here and
27 // eliminate the note
28 // accelgyro.setXAccelOffset(-1200);
29 // accelgyro.setYAccelOffset(-2500);
30 // accelgyro.setZAccelOffset(1988);

```

We can also read out the value of the offset which is already set through following code:

```

30 Serial.print("X.Y.Z offset :\t");
31 Serial.print(accelgyro.getXAccelOffset()); Serial.print("\t");
32 Serial.print(accelgyro.getYAccelOffset()); Serial.print("\t");
33 Serial.print(accelgyro.getZAccelOffset()); Serial.print("\n");

```

Read the values of 3 accelerations and 3 angular accelerations in the loop () function,

```
40     accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
```

Then, convert the data and send them to the serial port. For the conversion from the raw data unit of MPU6050 to the standard unit, please refer to dataheet.

### & Operator

The function of the operator "&" is to get the address. We know that the parameter of the function is used to pass the value to the function body. When a function is called, the value of variables that works as parameters does not change.

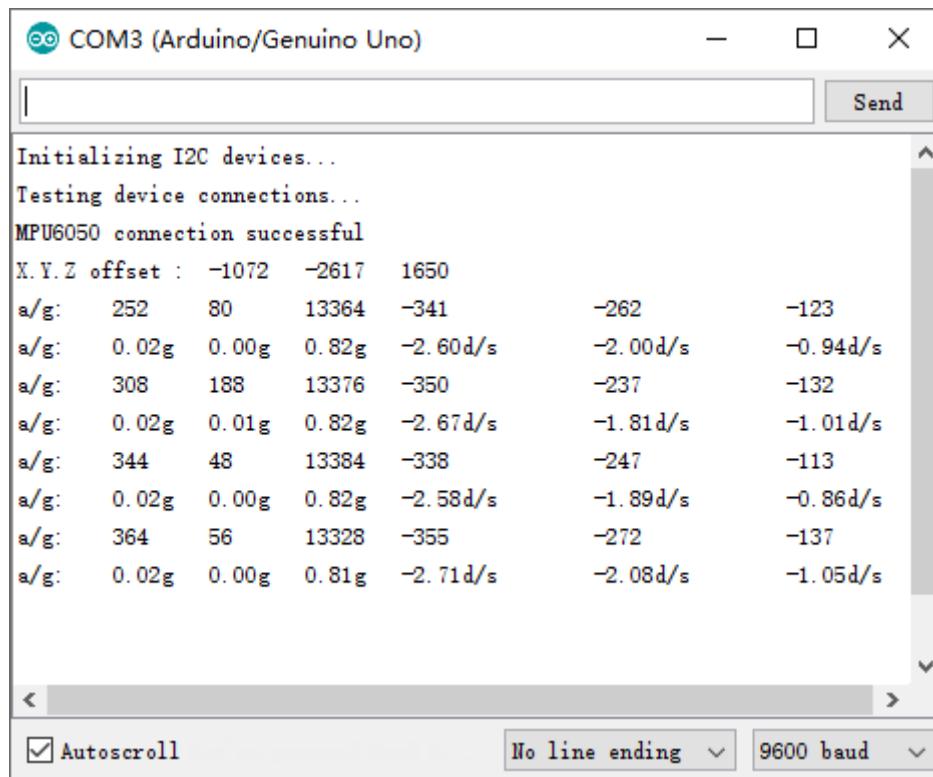
If the parameters of function are defined as pointer type, then when the function is called, the variable as function parameter will be passed to the function body and participate in the operation of the function body. And the value will be changed. It is equivalent to that the function indirectly return more value.

A pointer type variable points to an address. When we define it, we need to add "\*" in front of it, for example:

```
int *a;
```

When the function's parameter is pointer type, and the common variable works as parameter of the function, the & operator need to be added in front of the parameter.

Verify and upload the code, open the Serial Monitor, then you can see the value of MPU6050 in original state and converted state, which is sent by UNO board. Rotate and move the MPU6050 module, then you can see the change of values.



Data sent by this code may be too much for the users not familiar with acceleration. You can choose to upload sketch 14.1.2, which only send three direction acceleration values to the serial port. And it will be relatively easy to observe change of numbers, when you rotate and move this module,

# Chapter 15 LED Matrix

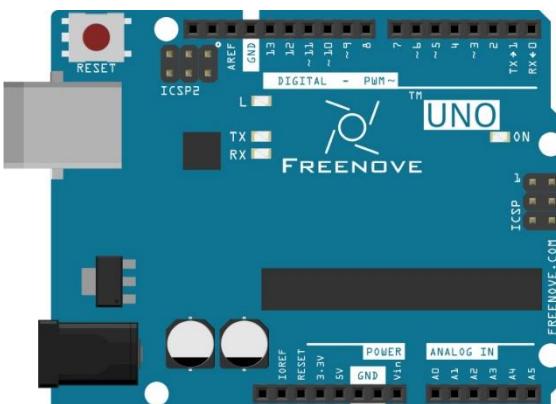
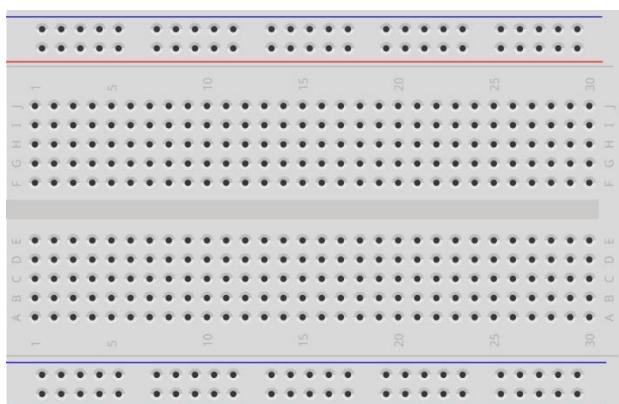
In the previous chapter, we have learned how to use some modules and sensors and shows some information on the computer through serial port. Now let us learn some modules which can output images and text.

In this chapter, we will learn how to use the LED matrix to output characters and images.

## Project 15.1 74HC595

Firstly, let us learn how to use the 74HC595 chip, which is very helpful for us to control the LED matrix.

### Component list

Freenove UNO x1	Breadboard x1
	
USB cable x1	74HC595 x1
	
Jumper M/M x17	LED bar graph x1
	
	Resistor 220Ω x8
	

## Code knowledge

### Hexadecimal

The conversion between binary and decimal system has been mentioned before. When you write the code, the number is decimal by default. Hexadecimal numbers need to add the 0x prefix in the code, such as 0x01. One Hexadecimal bit can present one number between 0-15. In order to facilitate writing, the numbers greater than 9 are written into the letter A-F (case-insensitive) such as 0x2A. The corresponding relationship is as follows:

Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Represent	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Conversion between hexadecimal and decimal system is similar to the conversion between hexadecimal and binary such as the sixteen digit 0x12:

Sequence	1	0
Number	1	2

When a hexadecimal number need to be converted to decimal number, first, the nth number of it need be multiplied by n power of 16, then sum all multiplicative results. Take 0x12 as an example:

$$1 * 16^1 + 2 * 16^0 = 18$$

When decimal number is converted to hexadecimal number, decimal number is divided by 16. Then we will get quotient and remainder, and quotient obtained will be continuously divided by 16 until quotient is zero. Arrange all remainders from right to left in a line. Then we complete the conversion. For example:

16	18	.....	2	0
16	1	.....	1	1

The result is of the conversion 0x12.

When you write code, sometimes it is convenient to use hexadecimal, especially involving bit operation, because 1 hexadecimal number can be expressed by 4 binary number ( $2^4=16$ ). The corresponding relationship between 4 bit binary numbers and 1 hexadecimal number is shown as follows:

4 bit binary	0000	0001	0010	0011	0100	0101	0110	0111
1 figure of hexadecimal	0	1	2	3	4	5	6	7

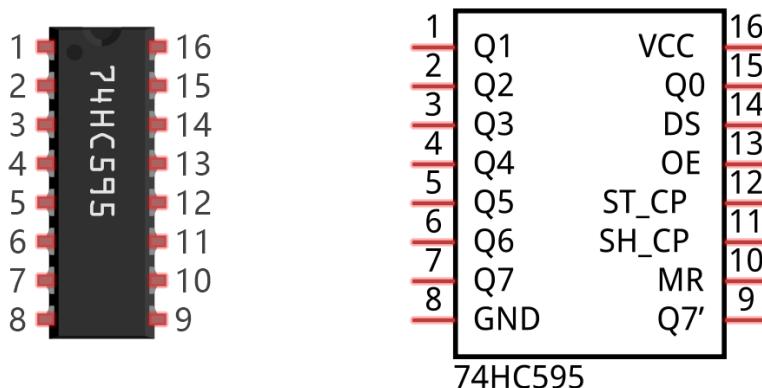
4 of bit binary	1000	1001	1010	1011	1100	1101	1110	1111
1 figure of hexadecimal	8	9	A	B	C	D	E	F

For example, binary 00010010 is corresponding to hexadecimal 0x12.

## Component knowledge

### 74HC595

74HC595 chip is used to convert serial data into parallel data. 74HC595 can convert the serial data of one byte to 8 bits, and send its corresponding level to the corresponding 8 ports. With this feature, 74HC595 can be used to expand the IO port of Arduino board. At least 3 ports on the Arduino board are need to control 8 ports of 74HC595.



The ports of 74HC595 are described as follows:

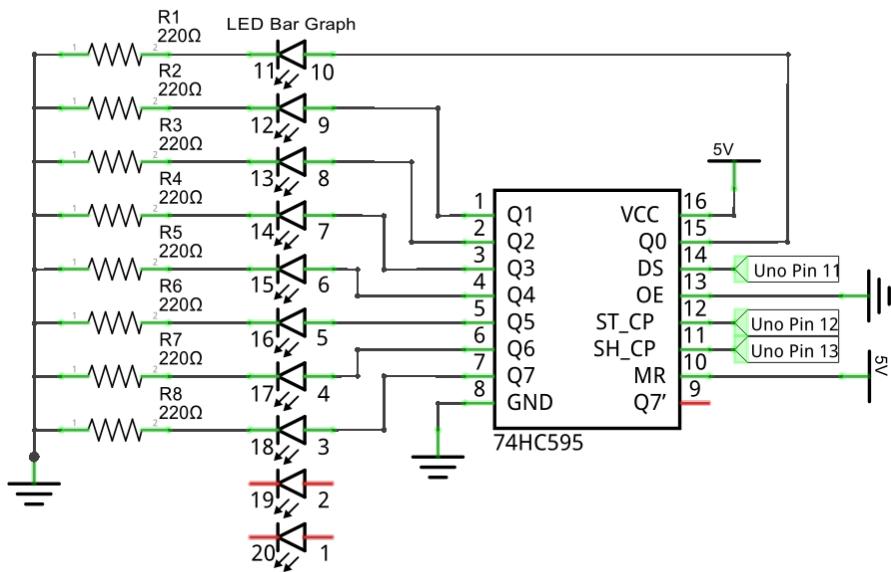
Pin name	Pin number	Description
Q0-Q7	15, 1-7	Parallel data output
VCC	16	The positive electrode of power supply, the voltage is 2~6V
GND	8	The negative electrode of power supply
DS	14	Serial data Input
OE	13	Enable output, When this pin is in high level, Q0-Q7 is in high resistance state When this pin is in low level, Q0-Q7 is in output mode
ST_CP	12	Parallel update output: when its electrical level is rising, it will update the parallel data output.
SH_CP	11	Serial shift clock: when its electrical level is rising, serial data input register will do a shift.
MR	10	Remove shift register: When this pin is in low level, the content in shift register will be cleared .
Q7'	9	Serial data output: it can be connected to more 74HC595 in series.

For more detail, please refer to the dataheet.

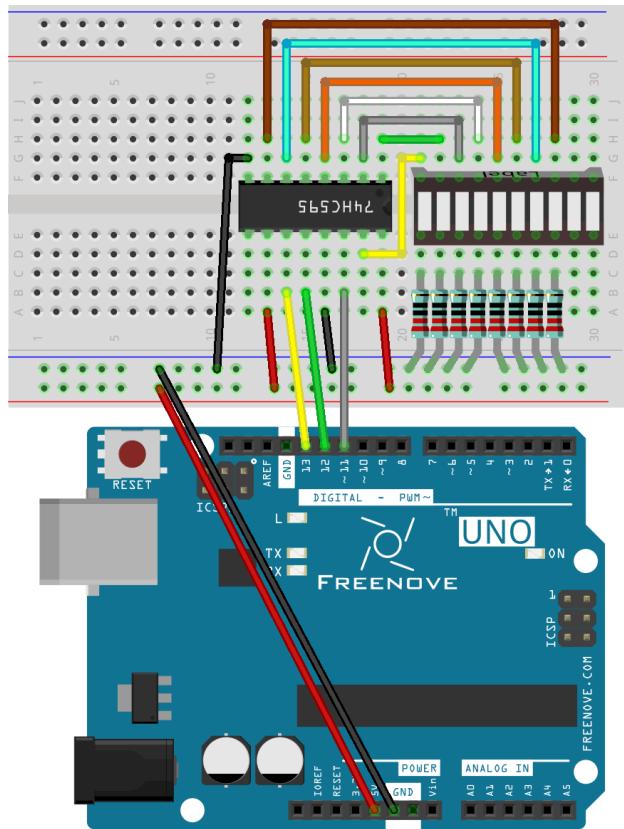
## Circuit

Use D11, D12, D13 pin on Freenove UNO to control the 74HC595, and connect it to the 8 LEDs of LED bar graph.

Schematic diagram



Hardware connection



## Sketch

### Sketch 15.1.1

Now write code to control the 8 LEDs of LED bar graph through 74HC595.

```
1 int latchPin = 12;           // Pin connected to ST_CP of 74HC595 (Pin12)
2 int clockPin = 13;           // Pin connected to SH_CP of 74HC595 (Pin11)
3 int dataPin = 11;            // Pin connected to DS of 74HC595 (Pin14)
4
5 void setup() {
6     // set pins to output
7     pinMode(latchPin, OUTPUT);
8     pinMode(clockPin, OUTPUT);
9     pinMode(dataPin, OUTPUT);
10 }
11
12 void loop() {
13     // Define a one-byte variable to use the 8 bits to represent the state of 8 LEDs of LED
14     // bar graph.
15     // This variable is assigned to 0x01, that is binary 00000001, which indicates only one
16     // LED light on.
17     byte x = 0x01;
18     for (int j = 0; j < 8; j++) {
19         // Output low level to latchPin
20         digitalWrite(latchPin, LOW);
21         // Send serial data to 74HC595
22         shiftOut(dataPin, clockPin, LSBFIRST, x);
23         // Output high level to latchPin, and 74HC595 will update the data to the parallel
24         // output port.
25         digitalWrite(latchPin, HIGH);
26         // make the variable move one bit to left once, then the bright LED move one step to
27         // the left once.
28         x <<= 1;
29         delay(100);
30     }
31 }
```

In the code, we configure three pins to control the 74HC595. And define a one-byte variable to control the state of 8 LEDs through the 8 bits of the variable. The LED lights on when the corresponding bit is 1. If the variable is assigned to 0x01, that is 00000001 in binary, there will be only one LED on.

```
15 byte x = 0x01;
```

In each loop, the x is sent to 74HC595. The sending process is as follows:

```

17 // Output low level to latchPin
18 digitalWrite(latchPin, LOW);
19 // Send serial data to 74HC595
20 shiftOut(dataPin, clockPin, LSBFIRST, x);
21 // Output high level to latchPin, and 74HC595 will update the data to the parallel
22 // output port.
23   digitalWrite(latchPin, HIGH);

```

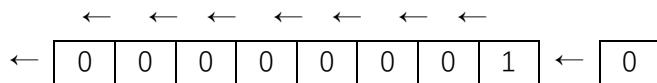
The x will be shift 1 bit to left in each cycle, which makes the bright LED of the 8 LEDs move one bit.

```
24 x <<= 1;
```

### << operator

"<<" is the left shift operator, which can make all bits of 1 byte shift by several bits to the left (high) direction and add 0 on the right (low). For example, shift binary 00000001 by 1 bit to left:

```
byte x = 1 << 1;
```

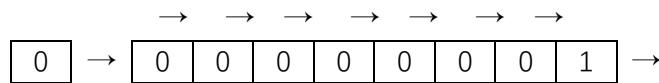


The result of x is 2 (binary 00000010).



There is another similar operator " >> ". For example, shift binary 00000001 by 1 bit to right:

```
byte x = 1 >> 1;
```

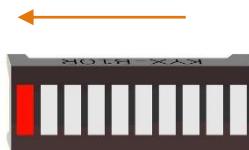


The result of x is 0 (00000000).



X <<= 1 is equivalent to x = x << 1 and x >>= 1 is equivalent to x = x >> 1

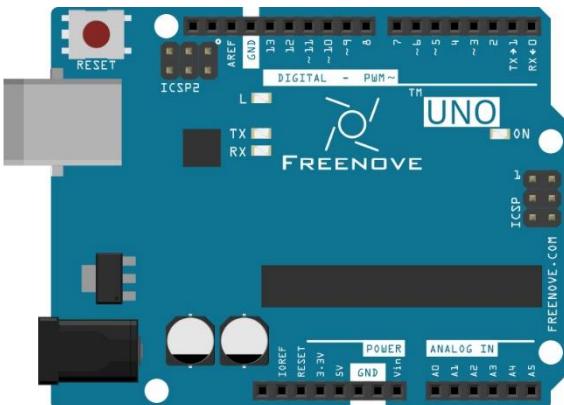
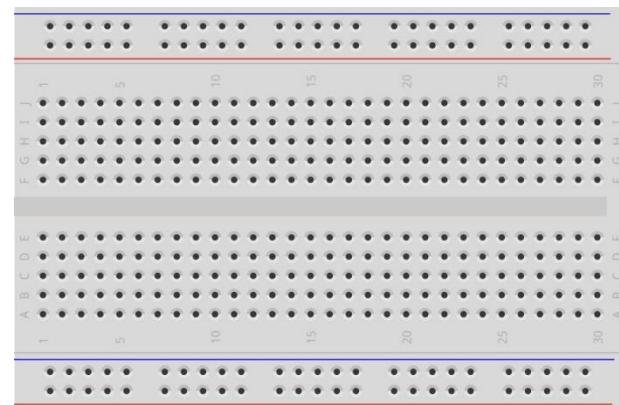
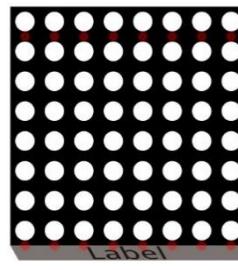
Verify and upload the code, then you will see the LED bar graph with the effect of flowing water.



## Project 15.2 LED Matrix

In the last section, we have used 74HC595 to control 8 LEDs of the LED bar graph. Now let's use 74HC595 to control LED matrix.

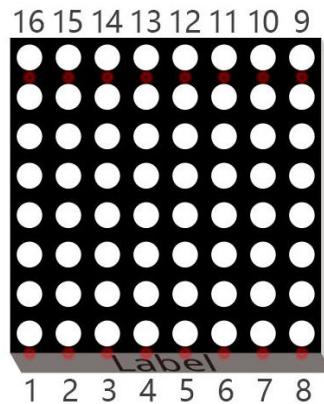
### Component list

Freenove UNO x1	Breadboard x1		
			
USB cable x1	LED matrix x1	74HC595 x1	R220 x8
			
Jumper M/M x33			

## Component knowledge

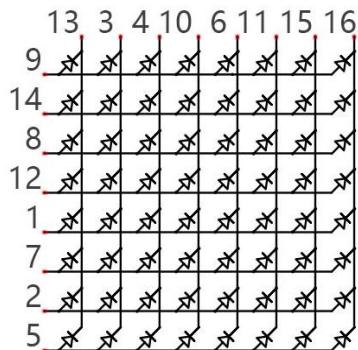
### LED matrix

LED matrix is a rectangular display module that consists of several LEDs. The following is an 8\*8 monochrome LED matrix with 64 LEDs (8 rows and 8 columns).



In order to facilitate the operation and save the ports, positive pole of LEDs in each row and negative pole of LEDs in each column are respectively connected together inside LED matrix module, which is called Common Anode. There is another form. Negative pole of LEDs in each row and positive pole of LEDs in each column are respectively connected together, which is called Common Cathode.

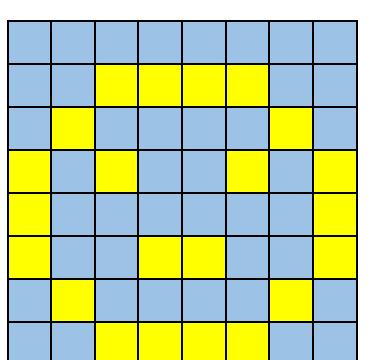
Connection mode of common anode



Connection mode of common cathode



Let us learn how connection mode of common anode works. Choose 16 ports on UNO board to connect to the 16 ports of LED Matrix. Configured one port in columns for low level, which make the column of the port selected. Then configure the eight ports in row to display content in the selected column. Delay for a moment. And then select the next column and outputs the corresponding content. This kind of operation to column is called scan. If you want to display the following image of a smiling face, you can display it in 8 columns, and each column is represented by one byte.



1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	0
0	1	0	0	0	0	1	0
1	0	1	0	0	1	0	1
1	0	0	0	0	0	0	1
1	0	0	1	1	0	0	1
0	1	0	0	0	0	1	0
0	0	1	1	1	1	0	0

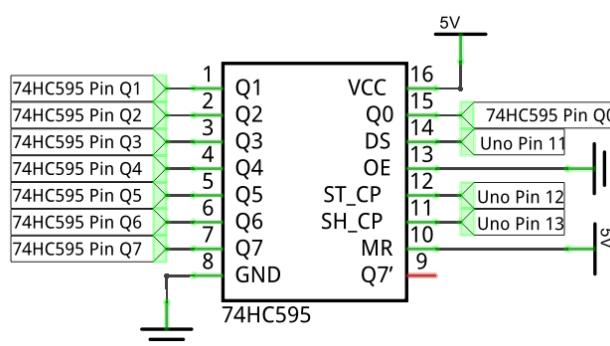
Column	Binary	Hexadecimal
1	0001 1100	0x1c
2	0010 0010	0x22
3	0101 0001	0x51
4	0100 0101	0x45
5	0100 0101	0x45
6	0101 0001	0x51
7	0010 0010	0x22
8	0001 1100	0x1c

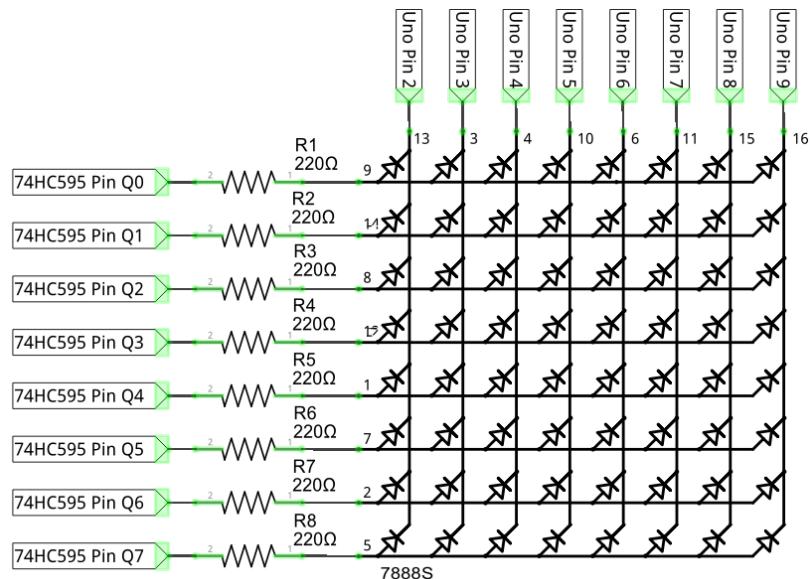
Scanning rows is another display way of dot matrix. It can save the controller ports, but it still needs 16 ports. So we use one 74HC595 to save some ports. You can use more 74HC595 to save ports where there are more ports needed.

## Circuit

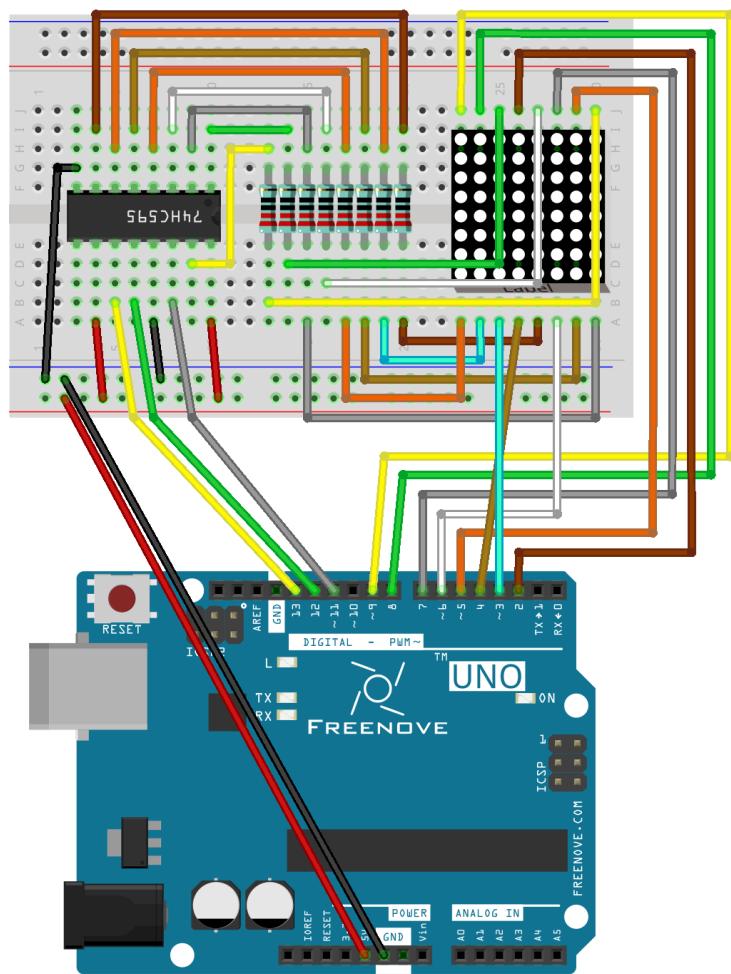
Use D11, D12, D13 pin on Freenove UNO board to control the 74HC595. And connect 74HC595 to the 8 anode pin of Matrix LED, in the meanwhile, connect 8 digital port on UNO board are to the 8 cathode pins of LED Matrix.

Schematic diagram





Hardware connection



## Sketch

### Sketch 15.2.1

Now write the code to drive LED dot matrix to display static and dynamic images, in fact, the dynamic image is formed by continuous static image.

```
1 int latchPin = 12;           // Pin connected to ST_CP of 74HC595 (Pin12)
2 int clockPin = 13;          // Pin connected to SH_CP of 74HC595 (Pin11)
3 int dataPin = 11;           // Pin connected to DS of 74HC595 (Pin14)
4 int LEDPin[] = {2, 3, 4, 5, 6, 7, 8, 9};    // column pin (cathode) of LED Matrix
5
6 // Define the pattern data for a smiling face
7 const int smilingFace[] = {
8     0x1C, 0x22, 0x51, 0x45, 0x45, 0x51, 0x22, 0x1C
9 };
10 // Define the data of numbers and letters, and save them in flash area
11 const int data[] PROGMEM = {
12     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // " "
13     0x00, 0x00, 0x21, 0x7F, 0x01, 0x00, 0x00, 0x00, // "1"
14     0x00, 0x00, 0x23, 0x45, 0x49, 0x31, 0x00, 0x00, // "2"
15     0x00, 0x00, 0x22, 0x49, 0x49, 0x36, 0x00, 0x00, // "3"
16     0x00, 0x00, 0x0E, 0x32, 0x7F, 0x02, 0x00, 0x00, // "4"
17     0x00, 0x00, 0x79, 0x49, 0x49, 0x46, 0x00, 0x00, // "5"
18     0x00, 0x00, 0x3E, 0x49, 0x49, 0x26, 0x00, 0x00, // "6"
19     0x00, 0x00, 0x60, 0x47, 0x48, 0x70, 0x00, 0x00, // "7"
20     0x00, 0x00, 0x36, 0x49, 0x49, 0x36, 0x00, 0x00, // "8"
21     0x00, 0x00, 0x32, 0x49, 0x49, 0x3E, 0x00, 0x00, // "9"
22     0x00, 0x00, 0x3E, 0x41, 0x41, 0x3E, 0x00, 0x00, // "0"
23     0x00, 0x00, 0x3F, 0x44, 0x44, 0x3F, 0x00, 0x00, // "A"
24     0x00, 0x00, 0x7F, 0x49, 0x49, 0x36, 0x00, 0x00, // "B"
25     0x00, 0x00, 0x3E, 0x41, 0x41, 0x22, 0x00, 0x00, // "C"
26     0x00, 0x00, 0x7F, 0x41, 0x41, 0x3E, 0x00, 0x00, // "D"
27     0x00, 0x00, 0x7F, 0x49, 0x49, 0x41, 0x00, 0x00, // "E"
28     0x00, 0x00, 0x7F, 0x48, 0x48, 0x40, 0x00, 0x00 // "F"
29 };
30
31 void setup() {
32     // set pins to output
33     pinMode(latchPin, OUTPUT);
34     pinMode(clockPin, OUTPUT);
35     pinMode(dataPin, OUTPUT);
36     for (int i = 0; i < 8; i++) {
37         pinMode(LEDPin[i], OUTPUT);
38     }
}
```

```
39  }
40
41 void loop() {
42     // Define a one-byte variable (8 bits) which is used to represent the selected state of
43     // 8 column.
44     int cols;
45     // Display the static smiling pattern
46     for (int j = 0; j < 500; j++) { // repeat 500 times
47         cols = 0x01; // Assign 0x01(binary 00000001) to the variable, which represents the
48         // first column is selected.
49         for (int i = 0; i < 8; i++) { // display 8 column data by scaning
50             matrixColsVal(cols); // select this column
51             matrixRowsVal(smilingFace[i]); // display the data in this column
52             delay(1); // display them for a period of time
53             matrixRowsVal(0x00); // clear the data of this column
54             cols <<= 1; // shift "cols" 1 bit left to select the next column
55         }
56     }
57     // Display the dynamic patterns of numbers and letters
58     for (int i = 0; i < 128; i++) { // "space, 0-9, A-F" 16 letters , each letter hold 8
59     // columns, total 136 columns. Firstly, display space ,then we need shift 128 times (136-8)
60     for (int k = 0; k < 10; k++) { // repeat image of each frame 10 times.
61         cols = 0x01; // Assign binary 00000001. Means the first column is selected.
62         for (int j = i; j < 8 + i; j++) { // display image of each frame
63             matrixColsVal(cols); // select this column
64             matrixRowsVal(pgm_read_word_near(data + j)); // display the data in this column
65             delay(1); // display them for a period of time
66             matrixRowsVal(0x00); // close the data of this column
67             cols <<= 1; // shift "cols" 1 bit left to select the next column
68     }
69 }
70 void matrixRowsVal(int value) {
71     // make latchPin output low level
72     digitalWrite(latchPin, LOW);
73     // Send serial data to 74HC595
74     shiftOut(dataPin, clockPin, LSBFIRST, value);
75     // make latchPin output high level, then 74HC595 will update the data to parallel
76     // output
77     digitalWrite(latchPin, HIGH);
78 }
```

```

79 void matrixColsVal(byte value) {
80     byte cols = 0x01;
81     // Output the column data to the corresponding port.
82     for (int i = 0; i < 8; i++) {
83         digitalWrite(LEDPin[i], ((value & cols) == cols) ? LOW : HIGH);
84         cols <= 1;
85     }
86 }
```

In the code, use an array to define the column pins of LED Matrix.

```
4 int LEDPin[] = {2, 3, 4, 5, 6, 7, 8, 9}; // Column pins (cathode) of LED Matrix
```

Use another array to define 8 column datas of a smiling face.

```

7 const int smilingFace[] = {
8     0x1C, 0x22, 0x51, 0x45, 0x45, 0x51, 0x22, 0x1C
9 };
```

Use another array to define some numbers and letters, and every eight elements of the array represent a dot matrix pattern data of a number or a letter.

```

11 const int data[] PROGMEM = {
12     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // " "
13     0x00, 0x00, 0x21, 0x7F, 0x01, 0x00, 0x00, 0x00, // "1"
14     0x00, 0x00, 0x23, 0x45, 0x49, 0x31, 0x00, 0x00, // "2"
15     0x00, 0x00, 0x22, 0x49, 0x49, 0x36, 0x00, 0x00, // "3"
16     0x00, 0x00, 0x0E, 0x32, 0x7F, 0x02, 0x00, 0x00, // "4"
17     0x00, 0x00, 0x79, 0x49, 0x49, 0x46, 0x00, 0x00, // "5"
18     0x00, 0x00, 0x3E, 0x49, 0x49, 0x26, 0x00, 0x00, // "6"
19     0x00, 0x00, 0x60, 0x47, 0x48, 0x70, 0x00, 0x00, // "7"
20     0x00, 0x00, 0x36, 0x49, 0x49, 0x36, 0x00, 0x00, // "8"
21     0x00, 0x00, 0x32, 0x49, 0x49, 0x3E, 0x00, 0x00, // "9"
22     0x00, 0x00, 0x3E, 0x41, 0x41, 0x3E, 0x00, 0x00, // "0"
23     0x00, 0x00, 0x3F, 0x44, 0x44, 0x3F, 0x00, 0x00, // "A"
24     0x00, 0x00, 0x7F, 0x49, 0x49, 0x36, 0x00, 0x00, // "B"
25     0x00, 0x00, 0x3E, 0x41, 0x41, 0x22, 0x00, 0x00, // "C"
26     0x00, 0x00, 0x7F, 0x41, 0x41, 0x3E, 0x00, 0x00, // "D"
27     0x00, 0x00, 0x7F, 0x49, 0x49, 0x41, 0x00, 0x00, // "E"
28     0x00, 0x00, 0x7F, 0x48, 0x48, 0x40, 0x00, 0x00 // "F"
29 };
```

### PROGMEM keyword

Microprocessors generally have two storage areas, namely ROM and RAM. ROM is used to store code. And these stored data will not change with the execution of code until the code are uploaded. RAM is used to store data, for example, the variables we defined are stored here. The stored data will change in real time with the execution of the code. Generally, capacity of RAM is small. So we can use PROGMEM keyword to save the data that don't change in ROM.

Define two functions, one of them uses UNO port to select the column.

```

79 void matrixColsVal(byte value) {
80     byte cols = 0x01;
81     // output the column data to the corresponding port.
82     for (int i = 0; i < 8; i++) {
83         digitalWrite(LEDPin[i], ((value & cols) == cols) ? LOW : HIGH);
84         cols <<= 1;
85     }
86 }
```

Another one uses 74HC595 to write data of the column.

```

70 void matrixRowsVal(int value) {
71     // ouput low level to latchPin
72     digitalWrite(latchPin, LOW);
73     // send serial data to 74HC595
74     shiftOut(dataPin, clockPin, LSBFIRST, value);
75     // output high level to latchPin, then 74HC595 will update the data to parallel output
76     // port
77     digitalWrite(latchPin, HIGH);
78 }
```

### ? : operator

"? :" operator is similar to conditional statements. When the expression in front of "?" is tenable, the statement in front of ":" will be executed. When the expression is not tenable, the statement behind ":" will be executed. For example:

```
int a = (1 > 0) ? 2: 3;
```

Because  $1 > 0$  is tenable, so "a" will be assigned to 2.

### Bitwise logical operation

There are many bitwise logical operators such as and (&), or (|), xor (^), negate (~). The result of exclusive or (^) is true only when two corresponding bit is not equal.

&, | and ^ is used to operate the corresponding bit of two numbers. Such as:

```
byte a = 1 & 2;
```

"a" will be assigned to 0. The calculation procedure is as follows:

$$\begin{array}{r} 1(00000001) \\ \& 2(00000010) \\ \hline 0(00000000) \end{array}$$

Negate (~) is used to negate a number, for example:

```
byte a = ~15;
```

"a" will be assigned to 240. The calculation procedure is as follows:

$$\begin{array}{r} \sim 15(00001111) \\ \hline 240(11110000) \end{array}$$

In the loop () function, firstly, show the static smile pattern. Selecte one of the 8 columns circularly in turn to display each the result. Repeat 500 times the process, then we can see a static smile pattern.

```

45   for (int j = 0; j < 500; j++) { //repeat 10 times
46     cols = 0x01; // variable cols is assigned to 0x01(binary 00000001), then the first
        column is selected
47     for (int i = 0; i < 8; i++) { // display the data in 8 column data by scaning
48       matrixColsVal(cols); // select this column
49       matrixRowsVal(smilingFace[i]); // display data in this column
50       delay(1); // display the data for a peroid of time
51       matrixRowsVal(0x00); // close the data of this column
52       cols <= 1; // shift"cols" 1 bit left to select the next column
53     }
54   }

```

Then display the dynamic pattern of the numbers and letters. We have defined space, 0-9, A-F, total of 16 characters (136 columns) in an array, among which 8 adjacent rows of data form one frame. Shift one column once. There are for 128 frames of image from the first frame (1-8) to the last frame (128-136 column). Each frame image is displayed 10 times, then display the next frame. Repeat the process above, then we can see the pattern of scrolling numbers and letters.

```

56   for (int i = 0; i < 128; i++) { // "space, 0-9, A-F"16 letters ,each letter hold 8
      columns, total 136 columns. Firstly, display space ,then we need shift 128 times (136-8)
57     for (int k = 0; k < 10; k++) { // repeat image of each frame 10 times.
58       cols = 0x01; // Assign binary 00000001. Means the first column is selected.
59       for (int j = i; j < 8 + i; j++) { // display image of each frame
60         matrixColsVal(cols); // select this column
61         matrixRowsVal(pgm_read_word_near(data + j)); // display the data in this column
62         delay(1); // display them for a period of time
63         matrixRowsVal(0x00); // close the data of this column
64         cols <= 1; // shift"cols" 1 bit left to select the next column
65       }
66     }
67   }
68 }

```

Verify and upload the code, then LED Matrix begins to display the static smile pattern. A few seconds later, LED Matrix will display the scrolling number 0-9 and the letter A-F.

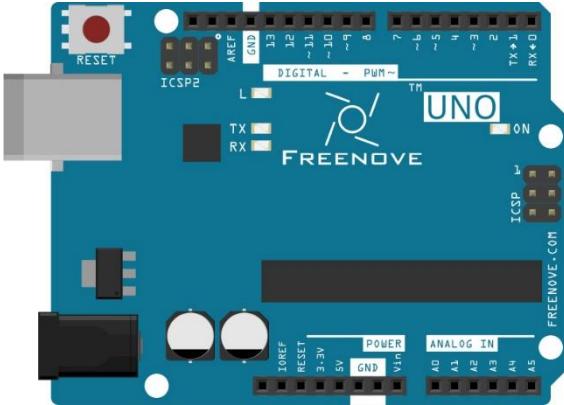
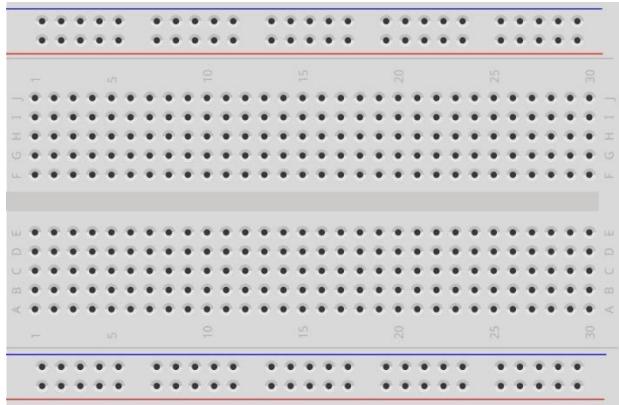
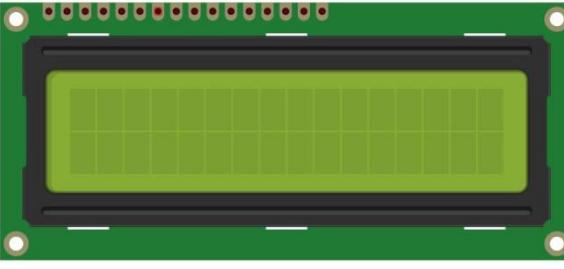
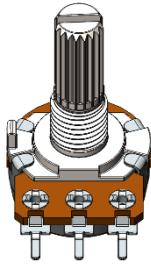
# Chapter 16 LCD1602

In the previous chapter, we have used the LED matrix to display images and characters. Now, let us use a screen module LCD1602 with a higher resolution to display more content.

## Project 16.1 Display the string on LCD1602

Firstly, use LCD1602 to display some strings.

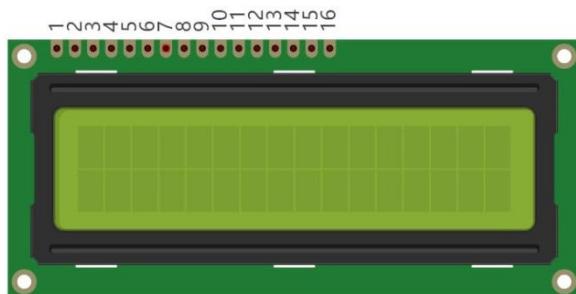
### Component list

Freenove UNO x1	Breadboard x1
	
USB cable x1	Jumper M/M x16
	
LCD1602 x1	Rotary potentiometer x1
	

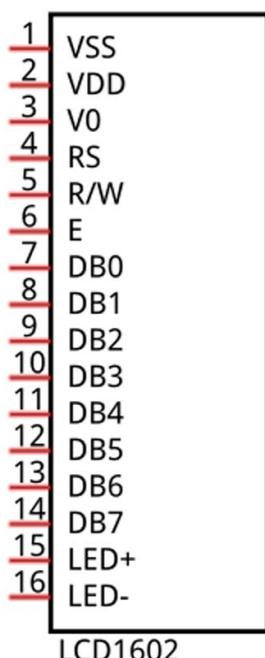
## Component knowledge

### LCD1602

LCD1602 can display 2 lines of characters in 16 columns. It can display numbers, letters, symbols, ASCII code and so on. As shown in the following is a monochrome LCD1602 display screen:



Circuit symbol and pin descriptions of LCD1602 are shown as follows:



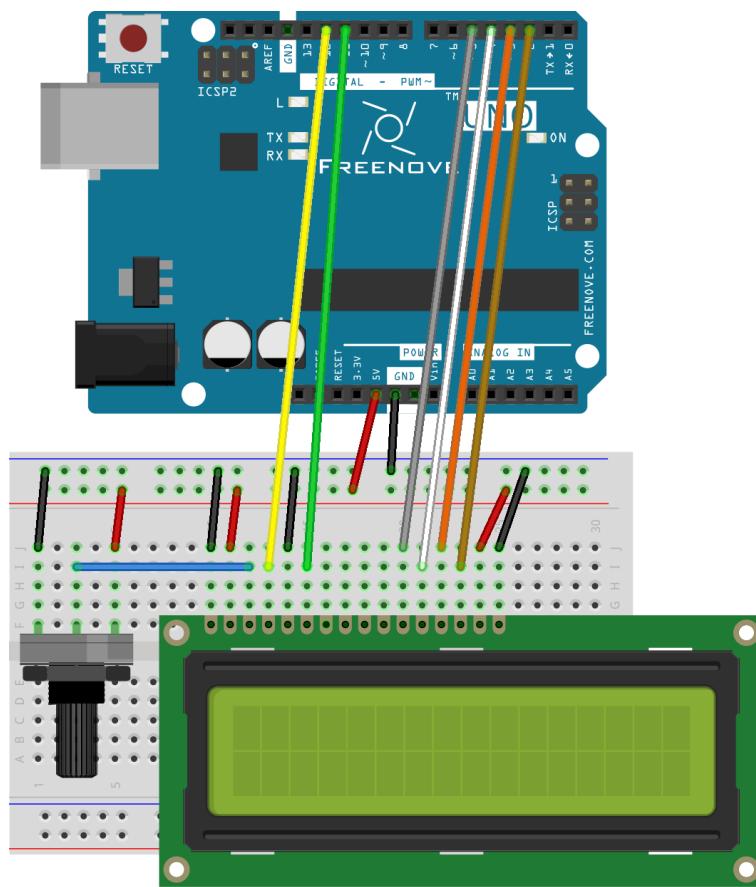
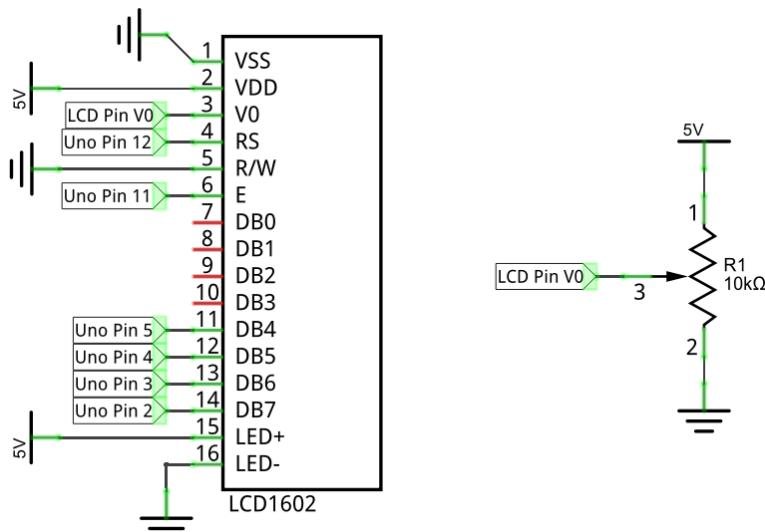
Pin name	Pin number	Description
VSS	1	Negative electrode of power supply
VDD	2	Positive electrode of power supply, the voltage is 5V
VO	3	Contrast and adjust the display effect
RS	4	Data/Command selection
RW	5	Read/Write selection
E	6	Enable pin
D0-D7	7-14	Data pin
LED+	15	Positive electrode of backlight LED, the voltage is 5V
LED-	16	Negative electrode of backlight LED

For more details, please refer to the dataheet.

## Circuit

The connection of Freenove UNO board and LCD1602 is shown below. And use a rotary potentiometer to adjust the contrast of LCD1602.

Schematic diagram



## Sketch

### Sketch 16.1.1

Now write code to make LCD1602 display a string and changing numbers.

```
1 #include <LiquidCrystal.h>
2
3 // initialize the library with the numbers of the interface pins
4 LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
5
6 void setup() {
7     // set up the LCD's number of columns and rows:
8     lcd.begin(16, 2);
9     // Print a message to the LCD
10    lcd.print("hello, world!");
11 }
12
13 void loop() {
14     // set the cursor to column 0, line 1
15     // (note: line 1 is the second row, since counting begins with 0):
16     lcd.setCursor(0, 1);
17     // print the number of seconds since reset:
18     lcd.print("Counter:");
19     lcd.print(millis() / 1000);
20 }
```

Use LiquidCrystal library to control the LCD:

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

LiquidCrystal library provides a specific class used to control LCD1602. We instantiate a LiquidCrystal object, and we can input some parameters which is related to pins of the LCD1602:

```
4 LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
```

Initialize the LCD1602 in the setup () function, and the parameters are the number of columns and rows of characters which LCD can display:

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

And then print a string:

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

Print a changing number in the loop () function:

```
13 void loop() {
14     // set the cursor to column 0, line 1
15     // (note: line 1 is the second row, since counting begins with 0):
16     lcd.setCursor(0, 1);
17     // print the number of seconds since reset:
18     lcd.print("Counter:");
19     lcd.print(millis() / 1000);
20 }
```

Before printing characters, we need to set the coordinate of the printed character, that is, in which line and which column:

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

If we don't set the coordinate, the string will be printed behind the last printed character.

### LiquidCrystal class

The LiquidCrystal class can manipulate common LCD screens. The first step is defining an object of LiquidCrystal, for example:

```
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
```

when instantiate the object, we need to write the pins connected to the LCD. There are many different ways for instantiation LiquidCrystal class. The way we use is shown below:

```
LiquidCrystal(rs, enable, d4, d5, d6, d7);
```

The common function of LiquidCrystal is shown below :

`lcd.begin(cols, rows)`: Initialize the LCD, and the parameters are the number of columns and rows of characters which LCD can display;

`lcd.setCursor(col, row)`: set the coordinate of the printed character, and the parameters is the row and the column (starting from 0, the number 0 represents first line or column);

`lcd.print(data)`: print the number and letters in the coordinate we set before. If we haven't set the coordinate, the string will be printed behind the last printed character.

Verify and upload the code, then you will see the LCD1602 display the string and changing number.

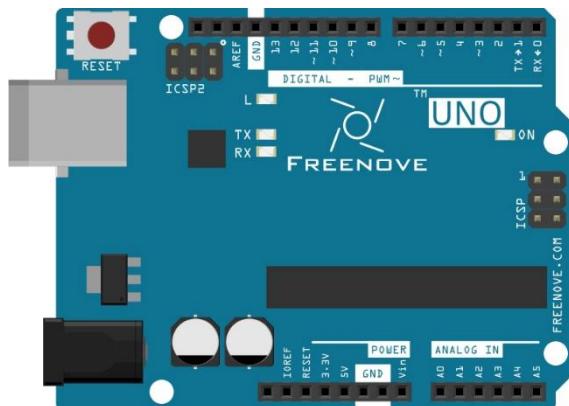
If the display of character is not normal, shift the rotary potentiometer to adjust the contrast.

## Project 16.2 LCD1602 Clock

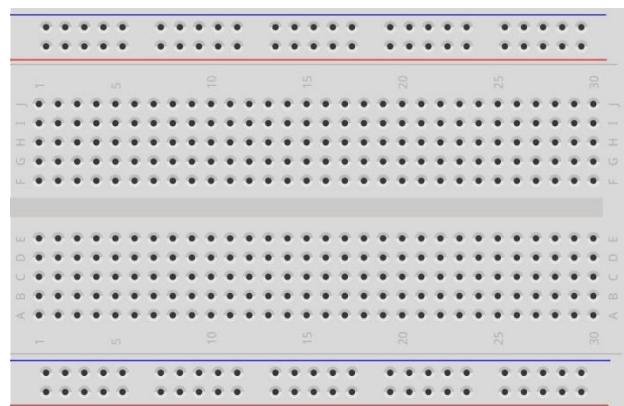
In the last chapter, we have used LCD1602 to display some strings, and now let us use LCD1206 to make a clock.

### Component list

Freenove UNO x1



Breadboard x1



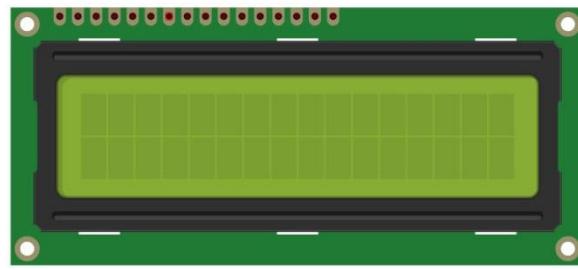
USB cable x1



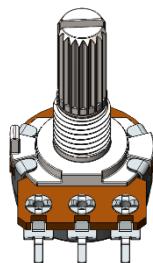
Jumper M/M x18



LCD1602 x1



Rotary potentiometer x1



Thermistor x1



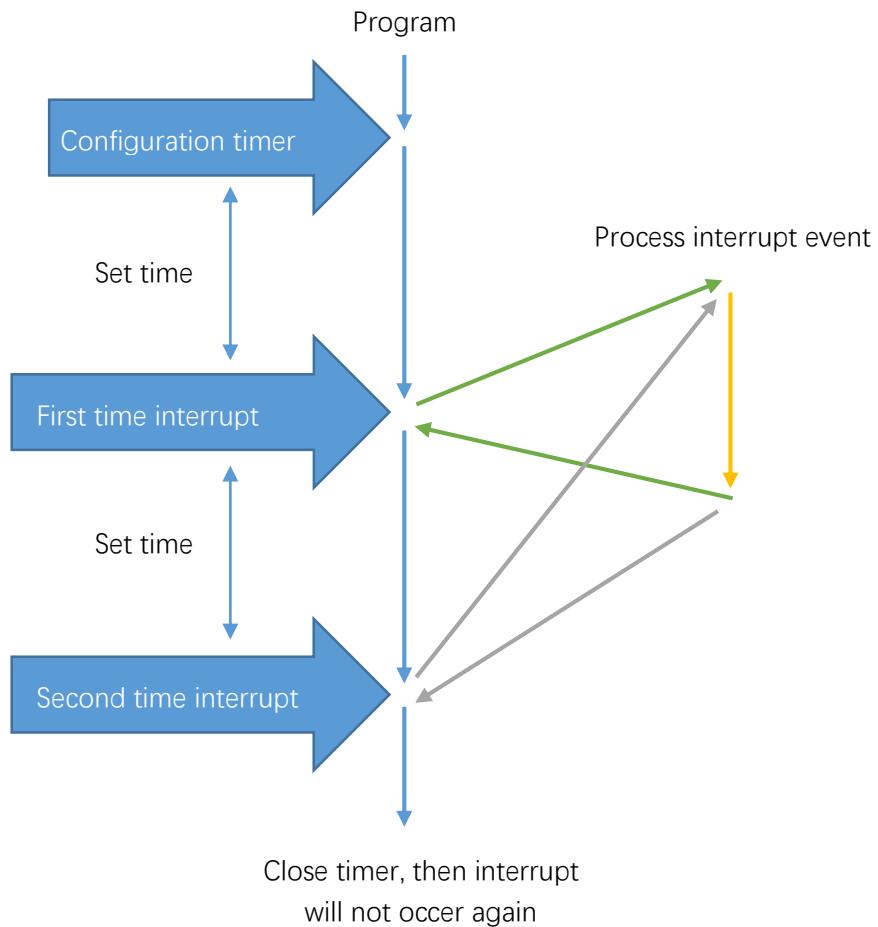
Resistor 10kΩ x1



## Code knowledge

### Timer

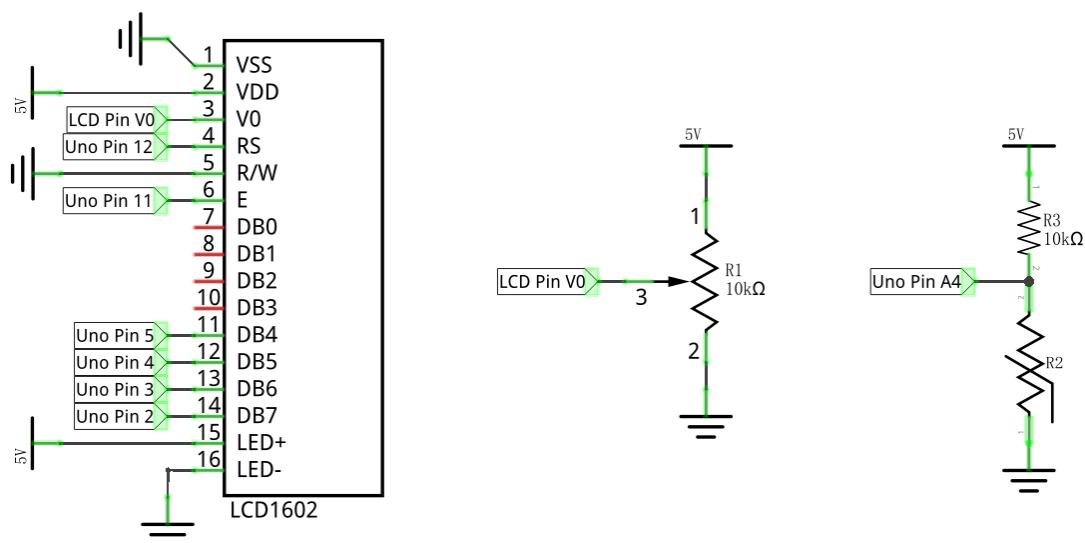
Timer can be set to produce an interrupt after a period of time. When the timer interrupt occurs, the processor will jump to the interrupt function to process the interrupt event. And after completion the processing, execution will return to the interrupted place to go on. If you don't close the timer, interrupt will occur at intervals you set.



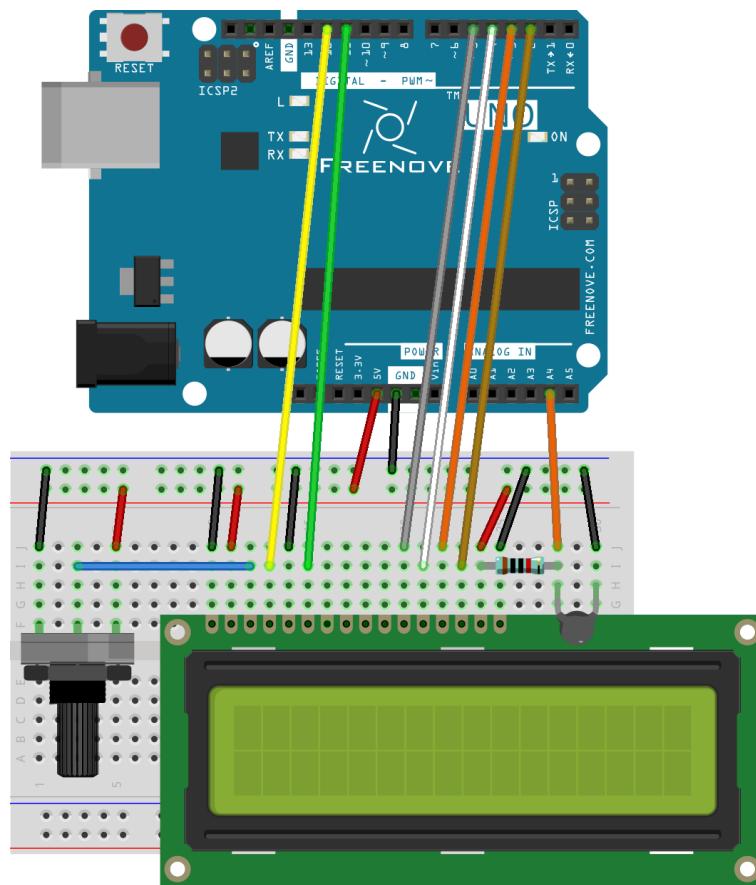
## Circuit

The connection between Freenove UNO board and LCD1602 is shown below, and a rotary potentiometer is used to adjust the contrast of LCD1602. A4 port is used to detect the voltage of thermistor.

Schematic diagram



Hardware connection



## Sketch

### Sketch 16.2.1

Before starting to write code, we need to import the library file we need. In the Arduino Software menu bar, click on the Add.ZIP Library, and add the FlexiTimer2.zip under the Libraries folder. The FlexiTimer2 library can help manipulate the timer.

Now write code to make LCD1602 display the time and temperature, and the time can be modified through the serial port.

```
1 #include <LiquidCrystal.h>      // Contains LiquidCrystal Library
2 #include <FlexiTimer2.h>        // Contains FlexiTimer2 Library
3
4 // initialize the library with the numbers of the interface pins
5 LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
6
7 int tempPin = 4;                // define the pin of temperature sensor
8 float tempVal;                 // define a variable to store temperature value
9 int hour, minute, second;       // define variables stored record time
10
11 void setup() {
12     lcd.begin(16, 2);          // set up the LCD's number of columns and rows
13     startingAnimation();       // display a dynamic start screen
14     FlexiTimer2::set(1000, timerInt); // configure the timer and interrupt function
15     FlexiTimer2::start();       // start timer
16     Serial.begin(9600);        // initialize serial port with baud rate 9600
17     Serial.println("UNO is ready!"); // print the string "UNO is ready!"
18     Serial.println("Input hour, minute, second to set time.");
19 }
20
21 void loop() {
22     // Get temperature
23     tempVal = getTemp();
24     if (second >= 60) {         // when seconds is equal to 60, minutes plus 1
25         second = 0;
26         minute++;
27         if (minute >= 60) {     // when minutes is equal to 60, hours plus 1
28             minute = 0;
29             hour++;
30             if (hour >= 24) {    // when hours is equal to 24, hours turn to zero
31                 hour = 0;
32             }
33         }
34     }
35     lcdDisplay();              // display temperature and time information on LCD
```

```
36     delay(200);
37 }
38
39 void startingAnimation() {
40     for (int i = 0; i < 16; i++) {
41         lcd.scrollDisplayRight();
42     }
43     lcd.print("starting... ");
44     for (int i = 0; i < 16; i++) {
45         lcd.scrollDisplayLeft();
46         delay(300);
47     }
48     lcd.clear();
49 }
50
51 // the timer interrupt function of FlexiTimer2 is executed every 1s
52 void timerInt() {
53     second++;      // second plus 1
54 }
55
56 // serial port interrupt function
57 void serialEvent() {
58     int inInt[3]; // define an array to save the received serial data
59     while (Serial.available()) {
60         for (int i = 0; i < 3; i++) {
61             inInt[i] = Serial.parseInt(); // receive 3 integer data
62         }
63         // print the received data for confirmation
64         Serial.print("Your input is: ");
65         Serial.print(inInt[0]);
66         Serial.print(",");
67         Serial.print(inInt[1]);
68         Serial.print(",");
69         Serial.println(inInt[2]);
70         // use received data to adjust time
71         hour = inInt[0];
72         minute = inInt[1];
73         second = inInt[2];
74         // print the modified time
75         Serial.print("Time now is: ");
76         Serial.print(hour / 10);
77         Serial.print(hour % 10);
78         Serial.print(':');
79         Serial.print(minute / 10);
```

```

80     Serial.print(minute % 10);
81     Serial.print(':' );
82     Serial.print(second / 10);
83     Serial.println(second % 10);
84 }
85 }
86
87 // function used by LCD1602 to display time and temperature
88 void lcdDisplay() {
89     lcd.setCursor(0, 0); // set the cursor to (0,0) (first column, first row).
90     lcd.print("TEMP: "); // display temperature information
91     lcd.print(tempVal);
92     lcd.print("C");
93     lcd.setCursor(0, 1); // set the cursor to (0,1) (first column, second row)
94     lcd.print("TIME: "); // display time information
95     lcd.print(hour / 10);
96     lcd.print(hour % 10);
97     lcd.print(':' );
98     lcd.print(minute / 10);
99     lcd.print(minute % 10);
100    lcd.print(':' );
101    lcd.print(second / 10);
102    lcd.print(second % 10);
103 }
```

In the code, we define 3 variables to represent time: second, minute, hour.

9	<code>int hour, minute, second; // define variables to store hour, minute, seconds</code>
---	-------------------------------------------------------------------------------------------

Defines a timer with cycle of 1000 millisecond (1 second). And each interrupt makes the second plus 1. When setting the timer, you need to define a function and pass the function name that works as parameter to FlexiTimer2::set () function.

14	<code>FlexiTimer2::set(1000, timerInt); // configure timer and timer interrupt function</code>
15	<code>FlexiTimer2::start(); // start timer</code>

After every interrupt, the second plus 1.

52	<code>void timerInt() {</code>
53	<code>    sec++; // second plus 1</code>
54	<code>}</code>

### ):: operator

"::" is scope operator. The function behind "::" belongs to the scope of the front. If we want to call the function defined in the FlexiTimer2 scope outside, we need to use the "::". It can be global scope operator, class scope operator and namespace scope operator. It is a namespace scope operator here. Because functions of FlexiTimer2 library is defined in the namespace of FlexiTimer2, so we can find them in FlexiTimer2 library file.

In the loop () function, the information on the LCD display will be refreshed at set intervals.

```

21 void loop() {
...
35     lcdDisplay();           // display temperature and time information on LCD
36     delay(200);
37 }
```

In the loop function, we need to control the second, minute, hour. When the second increases to 60, the minute adds 1, and reset the second to zero; when the minute increases to 60, the hour adds 1, and reset the minute to zero; when the hour increases to 24, reset it to zero.

```

24 if (second >= 60) {      // when the second increases to 60, the minute adds 1
25     second = 0;
26     minute++;
27     if (minute >= 60) {    //when the minute increases to 60, the hour adds 1
28         minute = 0;
29         hour++;
30         if (hour >= 24) {    // when the hour increases to 24, turn it to zero
31             hour = 0;
32         }
33     }
34 }
```

We define a function lcdDisplay () to refresh the information on LCD display. In this function, use two bit to display the hour, minute, second on the LCD. Such as hour/ 10 is the units, hour% 10 is the tens.

```

88 void lcdDisplay() {
89     lcd.setCursor(0, 0); // set the cursor to (0,0) (first column,first row).
90     lcd.print("TEMP: "); // display temperature information
91     lcd.print(tempVal);
92     lcd.print("C");
93     lcd.setCursor(0, 1); // set the cursor to (0,1) (first column,second row)
94     lcd.print("TIME: "); // display time information
95     lcd.print(hour / 10);
96     lcd.print(hour % 10);
97     lcd.print(':' );
98     lcd.print(minute / 10);
99     lcd.print(minute % 10);
100    lcd.print(':' );
101    lcd.print(second / 10);
102    lcd.print(second % 10);
103 }
```

Serial port interrupt function is used to receive the data sent by computer to adjust the time, and return the data for confirmation.

```

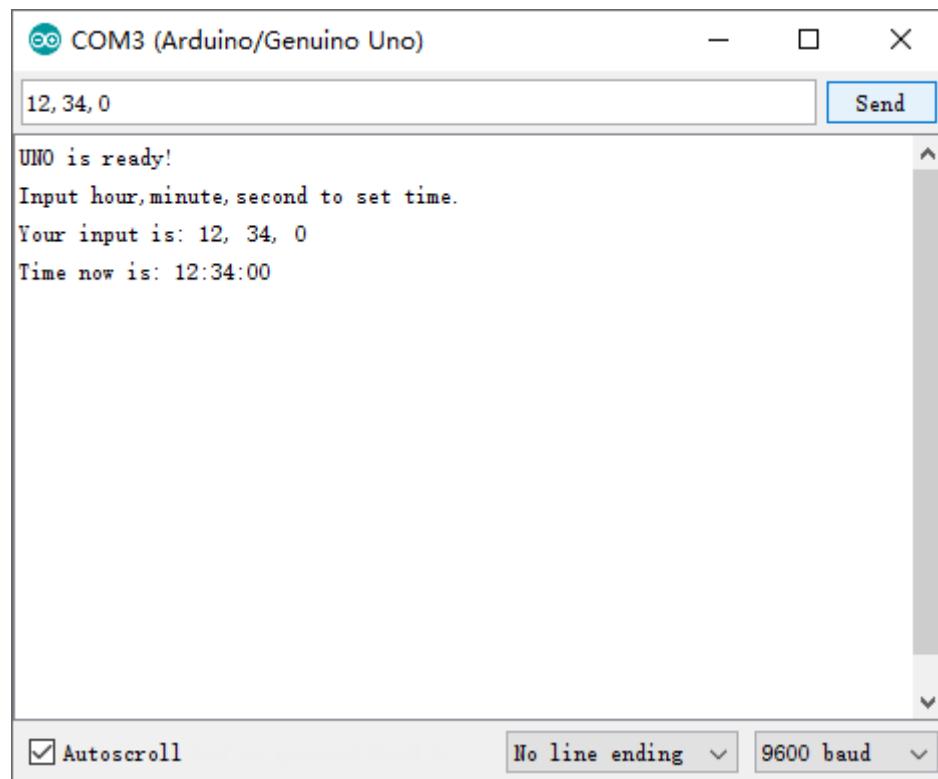
57 void serialEvent() {
58     int inInt[3]; // define an array to save the received serial data
59     while (Serial.available()) {
60         for (int i = 0; i < 3; i++) {
61             inInt[i] = Serial.parseInt(); // receive 3 integer data
62         }
63         // print the received data for confirmation
64         Serial.print("Your input is: ");
65         Serial.print(inInt[0]);
66         Serial.print(",");
67         Serial.print(inInt[1]);
68         Serial.print(",");
69         Serial.println(inInt[2]);
70         // use the received data to adjust time
71         hour = inInt[0];
72         minute = inInt[1];
73         second = inInt[2];
74         // print the modified time
75         Serial.print("Time now is: ");
76         Serial.print(hour / 10);
77         Serial.print(hour % 10);
78         Serial.print(':');
79         Serial.print(minute / 10);
80         Serial.print(minute % 10);
81         Serial.print(':');
82         Serial.print(second / 10);
83         Serial.println(second % 10);
84     }
85 }
```

We also define a function that displays a scrolling string when the UNO has been just started.

```

39 void startingAnimation() {
40     for (int i = 0; i < 16; i++) {
41         lcd.scrollDisplayRight();
42     }
43     lcd.print("starting... ");
44     for (int i = 0; i < 16; i++) {
45         lcd.scrollDisplayLeft();
46         delay(300);
47     }
48     lcd.clear();
49 }
```

Verify and upload the code. The LCD screen will display a scrolling string first, and then displays the temperature and time. We can open Monitor Serial and enter time in the sending area, then click the Send button to set the time.



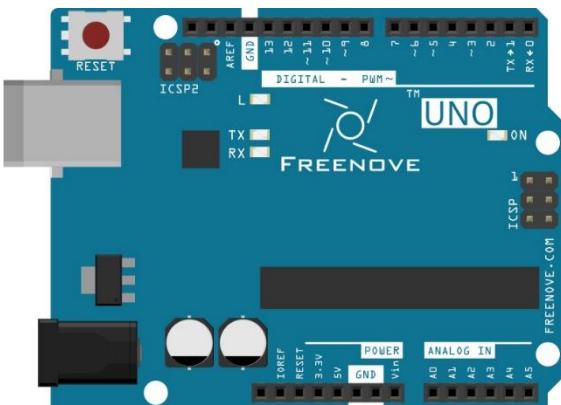
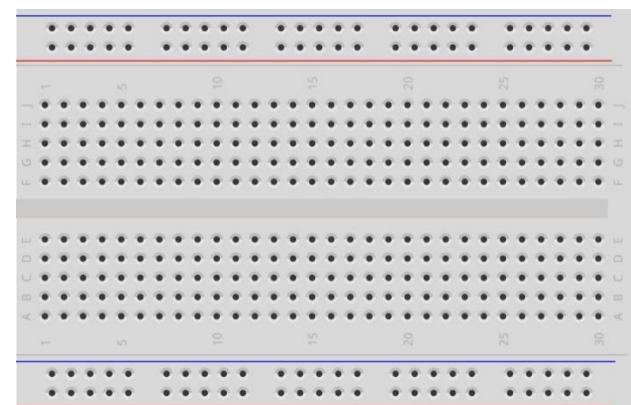
# Chapter 17 Digital Display

Digital display is a kind of device that can display one or several digits. We will learn how to use it in this chapter.

## Project 17.1 1-digit 7-segment Display

First, try to use the digit display that can display 1-digit number.

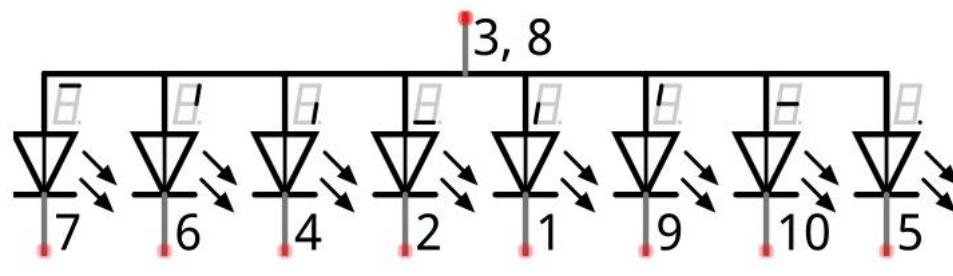
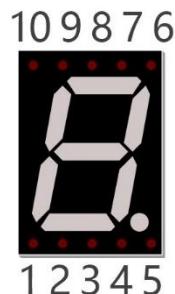
### Component list

Freenove UNO x1	Breadboard x1
	
USB cable x1	74HC595 x1
	Resistor 220Ω x8
Jumper M/M x18	1-digit 7-segment display x1
	

## Component knowledge

### 1-digit 7-segment display

1-digit 7-segment display is a kind of electron component that can display numbers. It has a figure of "8" and a decimal point, which consists of 8 LEDs. It can display numbers of 0~9 by lighting some of its LED segment. There are two types of LED common port inside, that is, common cathode port and common anode port. Common anode 1-digit 7-segment display is as follows:



If you want to display 0, you can light the following LEDs that connected with pin 7, 6, 4, 2, 1, 9.



If we use a byte to show the state of the LEDs that connected to pin 5, 10, 9, 1, 2, 4, 6, 7, we can make 0 represent the state of on and 1 for off. Then the number 0 can be expressed as a binary number 11000000, namely hex 0xc0.

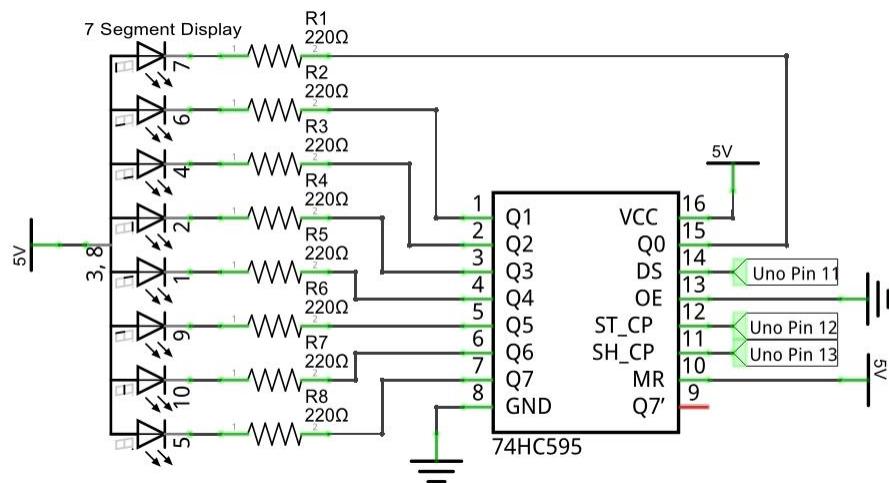
The numbers and letters that can be display are shown below:

Number/Letter	Binary number	Hexadecimal number
0	11000000	0xc0
1	11111001	0xf9
2	10100100	0xa4
3	10110000	0xb0
4	10011001	0x99
5	10010010	0x92
6	10000010	0x82
7	11111000	0xf8
8	10000000	0x80
9	10010000	0x90
A	10001000	0x88
b	10000011	0x83
C	11000110	0xc6
d	10100001	0xa1
E	10000110	0x86
F	10001110	0x8e

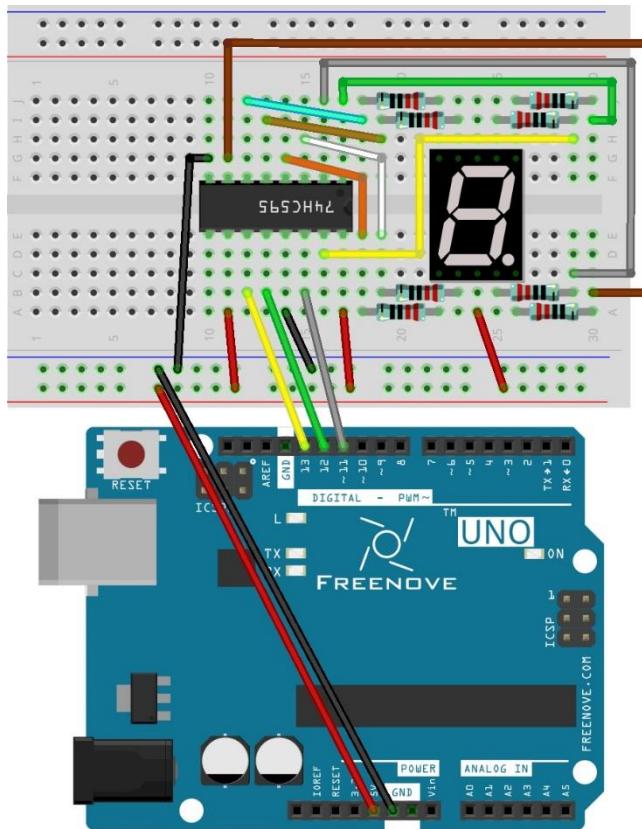
## Circuit

Use pin D11, D12, D13 on Freenove UNO to control the 74HC595, and connect it to the 1-digit 7-segment display.

Schematic diagram



Hardware connection



## Sketch

### Sketch 17.1.1

Now write code to control the 1-digit 7-segment display through 74HC595.

```

1 int latchPin = 12;           // Pin connected to ST_CP of 74HC595 (Pin12)
2 int clockPin = 13;          // Pin connected to SH_CP of 74HC595 (Pin11)
3 int dataPin = 11;           // Pin connected to DS of 74HC595 (Pin14)
4
5 // Define the encoding of characters 0-F of the common-anode 7-segment Display
6 byte num[] = {0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xf8, 0x80, 0x90, 0x88, 0x83,
7 0xc6, 0xa1, 0x86, 0x8e};
8
9 void setup() {
10    // set pins to output
11    pinMode(latchPin, OUTPUT);
12    pinMode(clockPin, OUTPUT);
13    pinMode(dataPin, OUTPUT);
14 }
15
16 void loop() {
17    // Cycling display 0-F
18    for (int i = 0; i <= 0x0f; i++) {
19        // Output low level to latchPin
20        digitalWrite(latchPin, LOW);
21        // Send serial data to 74HC595
22        shiftOut(dataPin, clockPin, MSBFIRST, num[i]);
23        // Output high level to latchPin, and 74HC595 will update the data to the parallel
24        // output port.
25        digitalWrite(latchPin, HIGH);
26        delay(1000);
27    }
28 }
```

We define an array to save and display the encoding of character 0-F in this code.

```
6 byte num[] = {0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xf8, 0x80, 0x90, 0x88, 0x83,
7 0xc6, 0xa1, 0x86, 0x8e};
```

Initialize the pin connected to 74HC595 in setup() function.

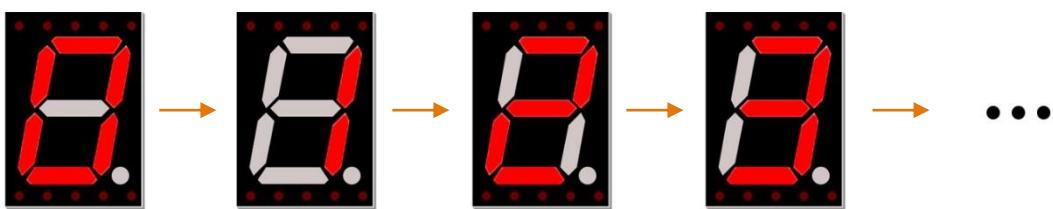
```

8 void setup() {
9    // set pins to output
10   pinMode(latchPin, OUTPUT);
11   pinMode(clockPin, OUTPUT);
12   pinMode(dataPin, OUTPUT);
13 }
```

Then in loop() function, send the encoding of 0-F to 74HC595 circularly.

```
17  for (int i = 0; i <= 0x0f; i++) {  
18      // Output low level to latchPin  
19      digitalWrite(latchPin, LOW);  
20      // Send serial data to 74HC595  
21      shiftOut(dataPin, clockPin, MSBFIRST, num[i]);  
22      // Output high level to latchPin, and 74HC595 will update the data to the parallel  
23      output port.  
24      digitalWrite(latchPin, HIGH);  
25      delay(1000);  
}
```

Verify and upload the code, then you will see the 1-digit 7-segment display show 0-F in a circular manner.

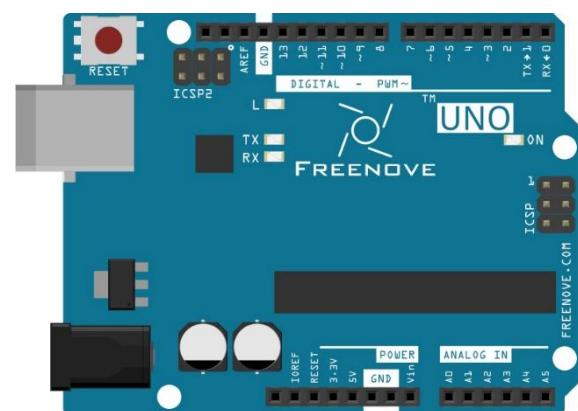


## Project 17.2 4-digit 7-segment Display

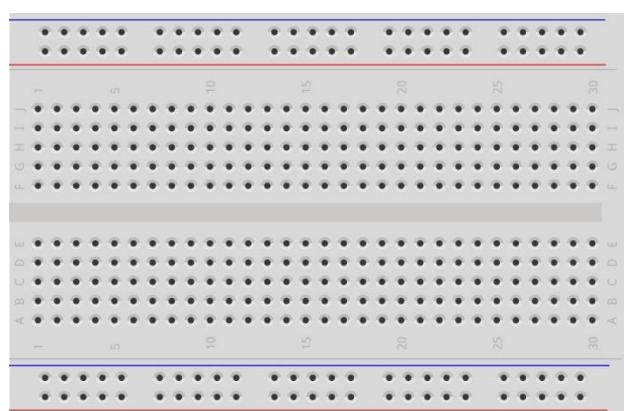
Now, try to use digit display that can display 4-digit numbers.

### Component list

Freenove UNO x1



Breadboard x1



USB cable x1



Jumper M/M x21



4-digit 7-segment display x1



74HC595 x1



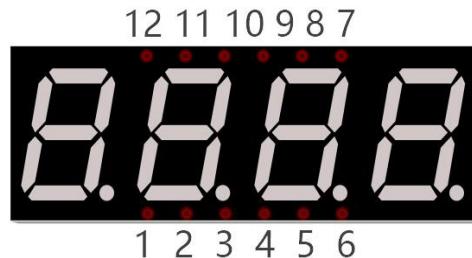
Resistor 220Ω x8



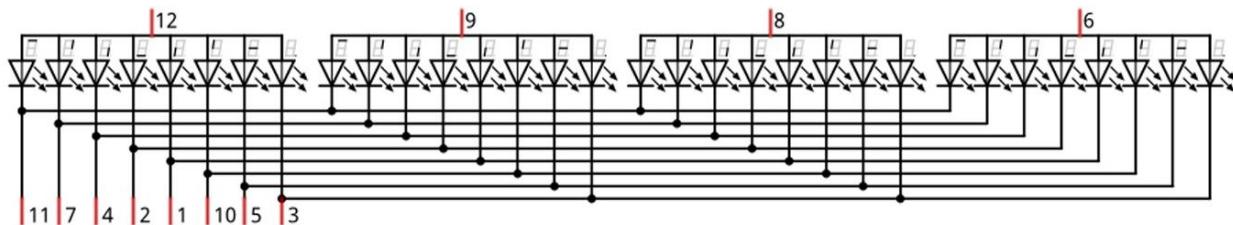
## Component knowledge

### 4-digit 7-segment display

4-digit 7-segment display is a component that integrates four 1-digit 7-segment display device, so it can display more numbers. There are also two types of LED common port inside, that is, common cathode port and common anode port. Common anode 4-digit 7-segment display is as follows.



The internal circuit is shown below, and eight LED cathode pins of all 1-digit 7-segment display are connected together.

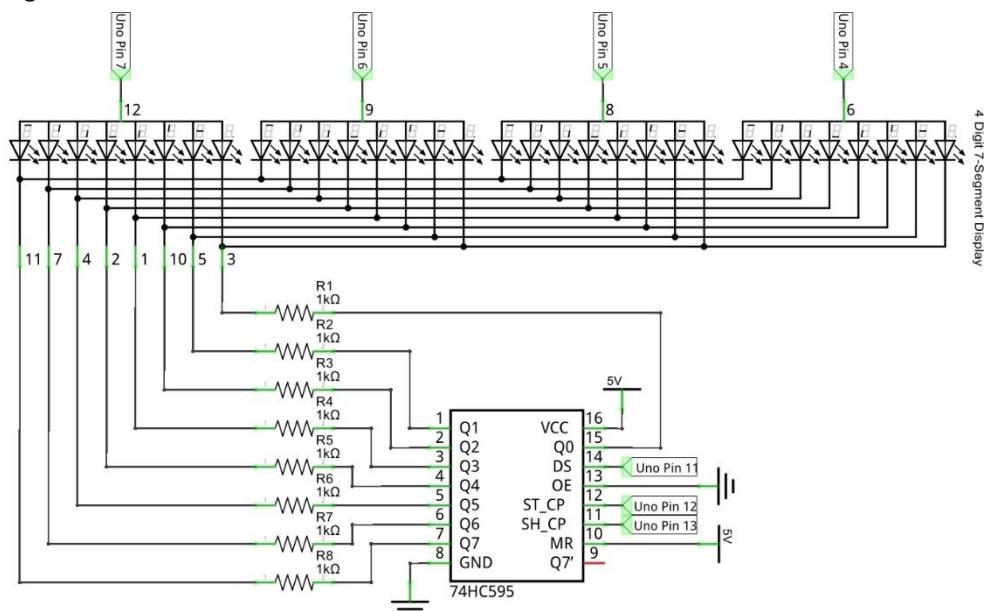


The display method of 4-digit 7-segment display is the same with LED Matrix. First, output high level to one of the common port, then output the content to the eight LED cathode when operating. Display the content of other 7-segment display in turn in this way. We can see all 4 digits at the same time when this process is fast enough.

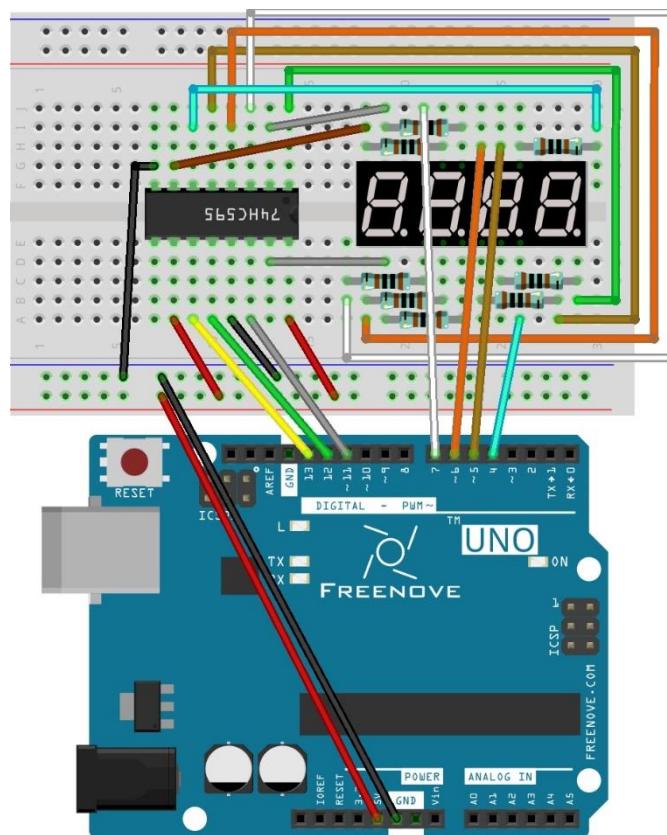
## Circuit

Use pin D11, D12, D13 on Freenove UNO to control the 74HC595, and connect it to the 4-digit 7-segment display. Use D7, D6, D5, D4 to control the 4 common ports.

Schematic diagram



Hardware connection



## Sketch

### Sketch 17.2.1

Now, write code to control 4-digit 7-segment display to display 4 numbers.

```
1 int latchPin = 12;           // Pin connected to ST_CP of 74HC595 (Pin12)
2 int clockPin = 13;           // Pin connected to SH_CP of 74HC595 (Pin11)
3 int dataPin = 11;            // Pin connected to DS of 74HC595 (Pin14)
4 int comPin[] = {7, 6, 5, 4};// Common pin (anode) of 4 digit 7-segment display
5
6 // Define the encoding of characters 0-F of the common-anode 7-Segment Display
7 byte num[] = {0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xf8, 0x80, 0x90, 0x88, 0x83,
8 0xc6, 0xa1, 0x86, 0x8e};
9
10 void setup() {
11     // set pins to output
12     pinMode(latchPin, OUTPUT);
13     pinMode(clockPin, OUTPUT);
14     pinMode(dataPin, OUTPUT);
15     for (int i = 0; i < 4; i++) {
16         pinMode(comPin[i], OUTPUT);
17     }
18 }
19
20 void loop() {
21     for (int i = 0; i < 4; i++) {
22         // Select a single 7-segment display
23         chooseCommon(i);
24         // Send data to 74HC595
25         writeData(num[i]);
26         delay(5);
27         // Clear the display content
28         writeData(0xff);
29     }
30 }
31
32 void chooseCommon(byte com) {
33     // Close all single 7-segment display
34     for (int i = 0; i < 4; i++) {
35         digitalWrite(comPin[i], LOW);
36     }
37     // Open the selected single 7-segment display
38     digitalWrite(comPin[com], HIGH);
39 }
```

```

39
40 void writeData(int value) {
41     // Make latchPin output low level
42     digitalWrite(latchPin, LOW);
43     // Send serial data to 74HC595
44     shiftOut(dataPin, clockPin, LSBFIRST, value);
45     // Make latchPin output high level, then 74HC595 will update the data to parallel
46     // output
47     digitalWrite(latchPin, HIGH);
}

```

Besides the similarity with the previous section, the difference is that this code is to output content to the four 7-segment display continuously. Write a function to select a common port.

```

31 void chooseCommon(byte com) {
32     // Close all the single 7-segment display
33     for (int i = 0; i < 4; i++) {
34         digitalWrite(comPin[i], LOW);
35     }
36     // Open the selected single 7-segment display
37     digitalWrite(comPin[com], HIGH);
38 }

```

Write a function to send data to 74 HC595

```

40 void writeData(int value) {
41     // Make latchPin output low level
42     digitalWrite(latchPin, LOW);
43     // Send serial data to 74HC595
44     shiftOut(dataPin, clockPin, LSBFIRST, value);
45     // Make latchPin output high level, then 74HC595 will update the data to parallel
46     // output
47     digitalWrite(latchPin, HIGH);
}

```

First select a commom port and then output the content, which will be displayed by the 7-segment display connected to common port, to 74HC595 when operating. Clear the display content after a period of time to avoid ghosting phenomenon.

```

21     // Select a single 7-segment display
22     chooseCommon(i);
23     // Send data to 74HC595
24     writeData(num[i]);
25     delay(5);
26     // Clear the display content
27     writeData(0xff);

```

Use cycle command in loop() function to output content to the four 7-segment display.

```
20   for (int i = 0; i < 4; i++) {  
...  
28 }
```

Repeat this operation continuously, and you can see all 4 digits.

Verify and upload the code, then you will see number 0123 shown by-4 digit 7-segment display.



### Sketch 17.2.2

Now write code to control 4-digit 7-segment display to display dynamic numbers.

```
1 #include <FlexiTimer2.h>      // Contains FlexiTimer2 Library
2
3 int latchPin = 12;           // Pin connected to ST_CP of 74HC595 (Pin12)
4 int clockPin = 13;           // Pin connected to SH_CP of 74HC595 (Pin11)
5 int dataPin = 11;            // Pin connected to DS of 74HC595 (Pin14)
6 int comPin[] = {7, 6, 5, 4};// Common pin (anode) of 4 digit 7-segment display
7
8 int second = 0;              // Define variables stored record time
9
10 // Define the encoding of characters 0-F for the common-anode 7-Segment Display
11 byte num[] = {0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xf8, 0x80, 0x90, 0x88, 0x83,
12 0xc6, 0xa1, 0x86, 0x8e} ;
13
14 void setup() {
15     // Set pins to output
16     pinMode(latchPin, OUTPUT);
17     pinMode(clockPin, OUTPUT);
18     pinMode(dataPin, OUTPUT);
19     for (int i = 0; i < 4; i++) {
20         pinMode(comPin[i], OUTPUT);
21     }
22     FlexiTimer2::set(1000, timerInt); // configure the timer and interrupt function
23     FlexiTimer2::start();          // start timer
24 }
25
26 void loop() {
27     // Calculate the seconds of each digit number
28     byte bit[4];
29     bit[0] = second % 10;
30     bit[1] = second / 10 % 10;
31     bit[2] = second / 100 % 10;
32     bit[3] = second / 1000 % 10;
33     // Show seconds
34     for (int i = 0; i < 4; i++) {
35         // Select a single 7-segment display
36         chooseCommon(i);
37         // Send data to 74HC595
38         writeData(num[bit[3 - i]]);
39         delay(5);
40         // Clear the display content
41         writeData(0xff);
42     }
43 }
```

```

42 }
43
44 // the timer interrupt function of FlexiTimer2 is executed every 1s
45 void timerInt() {
46     second++;      // second plus 1
47 }
48
49 void chooseCommon(byte com) {
50     // Close all single 7-segment display
51     for (int i = 0; i < 4; i++) {
52         digitalWrite(comPin[i], LOW);
53     }
54     // Open the selected single 7-segment display
55     digitalWrite(comPin[com], HIGH);
56 }
57
58 void writeData(int value) {
59     // Make latchPin output low level
60     digitalWrite(latchPin, LOW);
61     // Send serial data to 74HC595
62     shiftOut(dataPin, clockPin, LSBFIRST, value);
63     // Make latchPin output high level, then 74HC595 will update the data to parallel
64     // output
65     digitalWrite(latchPin, HIGH);
}

```

Frist set a timer with the cycle of 1s

21	FlexiTimer2::set(1000, timerInt); // configure the timer and interrupt function
22	FlexiTimer2::start(); // start timer

In the timer interrupt function, make the variable second plus 1

45	void timerInt() {
46	second++; // second plus 1
47	}

In function loop(), frist calculate the unit of variable second, then decade, hundreds and kilobit for single display.

27	byte bit[4];
28	bit[0] = second % 10;
29	bit[1] = second / 10 % 10;
30	bit[2] = second / 100 % 10;
31	bit[3] = second / 1000 % 10;

Then display the value of each bit.

37	writeData(num[bit[3 - i]]);
----	-----------------------------

Verify and upload the code, then you will see the dymaic number shown by 4 digit 7-segment display.

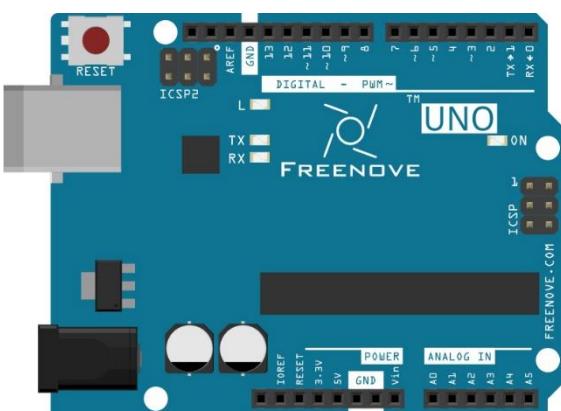
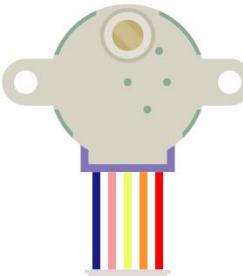
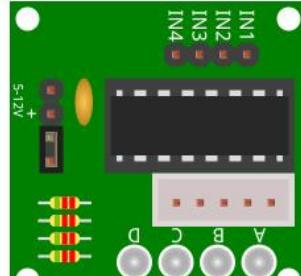
# Chapter 18 Stepper Motor

Stepper motor is a kind of motor that can rotate a certain angle at a time. With that, we can achieve a higher precision mechanical movement easily.

## Project 18.1 Drive Stepper Motor

Now, try to drive a stepper motor.

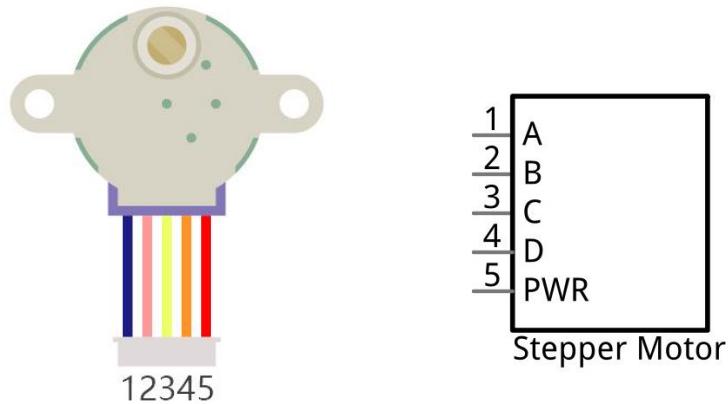
### Component list

Freenove UNO x1	Stepper motor x1	ULN2003 stepper motor driver x1
 A blue Freenove UNO R3 microcontroller board with various pins, a USB port, and a FTDI chip.	 A grey cylindrical stepper motor with four lead wires (red, yellow, green, blue) attached to its rear.	 A green printed circuit board (PCB) featuring a ULN2003 integrated circuit and several connection pins labeled IN1, IN2, IN3, IN4, A, B, C, D, and VDD.

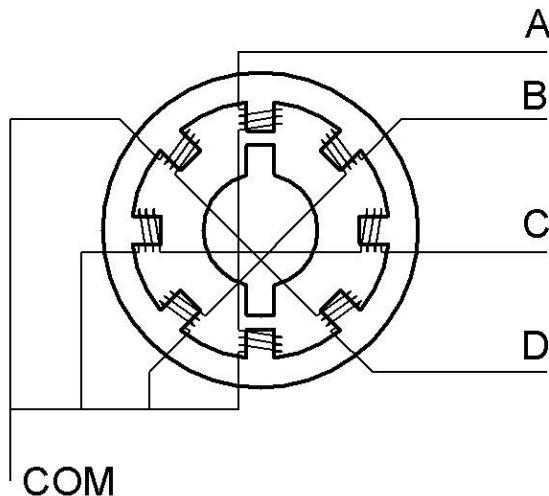
## Component knowledge

### Stepper motor

Stepper motor is a kind of open-loop control motor that can transform electrical pulse signal into angular displacement or linear displacement. In the case of unoverlaod, motor speed and stop position only depends on the frequency of pulse signal and pulse number, and not affected by load change. Schematic of small four phase reduction step motor is as follows:

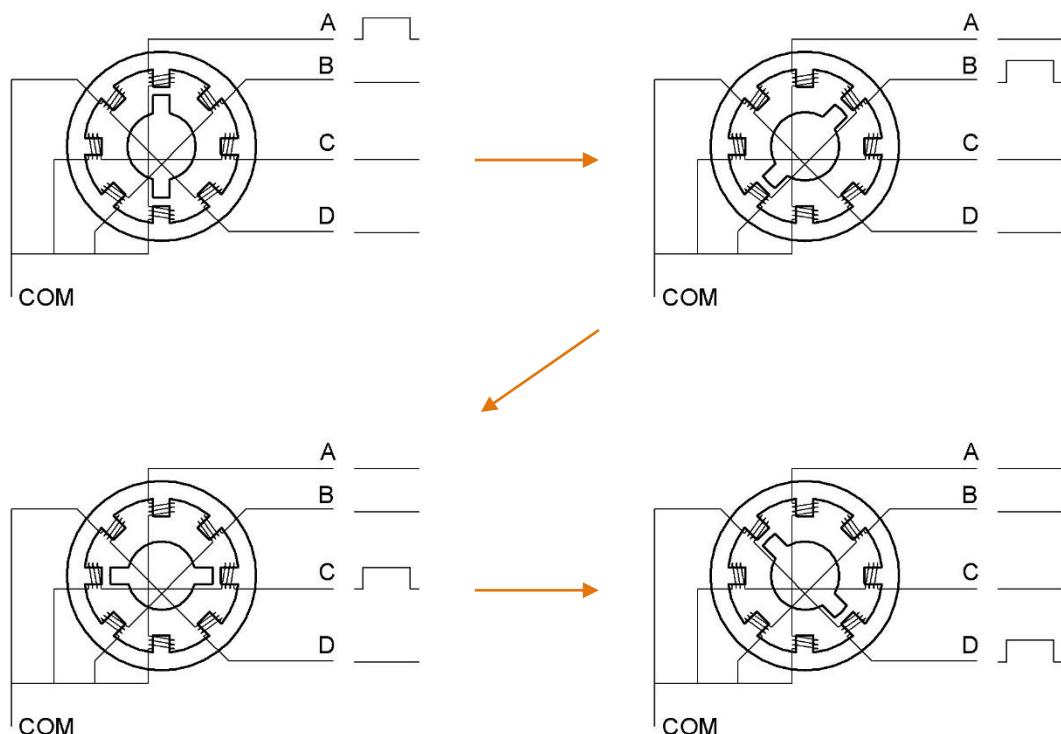


The internal structure principle of 4 phase stepper motor can be simply shown as follows:



Outside of the motor is the motor stator and inside of the motor is the motor rotor. There are several coils on the stator (usually an integral multiple of phase number) that can turn into electromagnet, when it is electrified, to attract the protruding part of the rotor (usually in the form of iron or permanent magnet). So, the motor can rotate through connecting the coils on the stator to power supply orderly.

A common drive process is as follows:



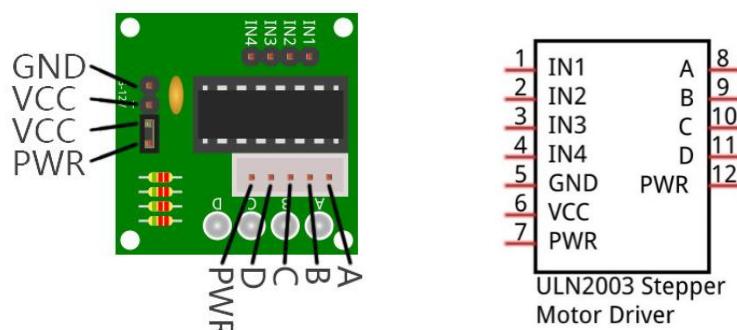
In the above process, when the stepper motor rotates each time, it is called a step. We can control the step motor's rotation angle by controlling rotating steps and control the step motor's rotation speed by controlling the time of each step.

Stepper motor also has other control mode, for example, connect A phase, and then connect AB phase at the same time, now the stator is in the middle of AB phase and it only goes a half step. We can use this way to improve the stepper motor's stability and reduce noise at the same time. Readers who are interested can learn this method yourself by searching relative knowledge.

The stepper motor's stator we used has 32 magnetic poles and one rotation circle needs 32 steps. The stepper motor's output shaft is connected with the reduction gear group and reduction ratio is 1/64. So one rotation cycle needs  $32 \times 64$  steps.

### ULN2003 stepper motor driver

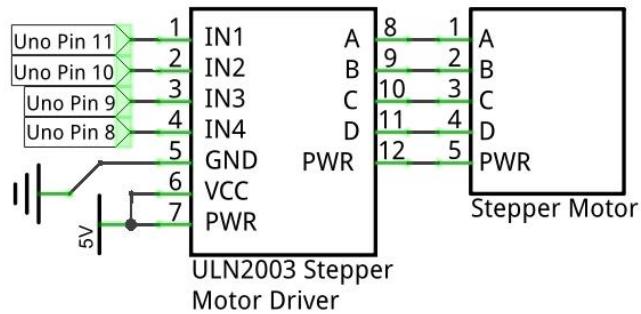
ULN2003 stepper motor driver is used to transfer the weaker signal into stronger control signal that can drive the stepper motor. The input signal IN1-IN4 is correspond to the output signal A-D and the circuit board is integrated with 4 LEDs to indicate the state of each signal. PWR interface can be used for the stepper motor's supply separately, and PWR and VCC is connected by jumper by default.



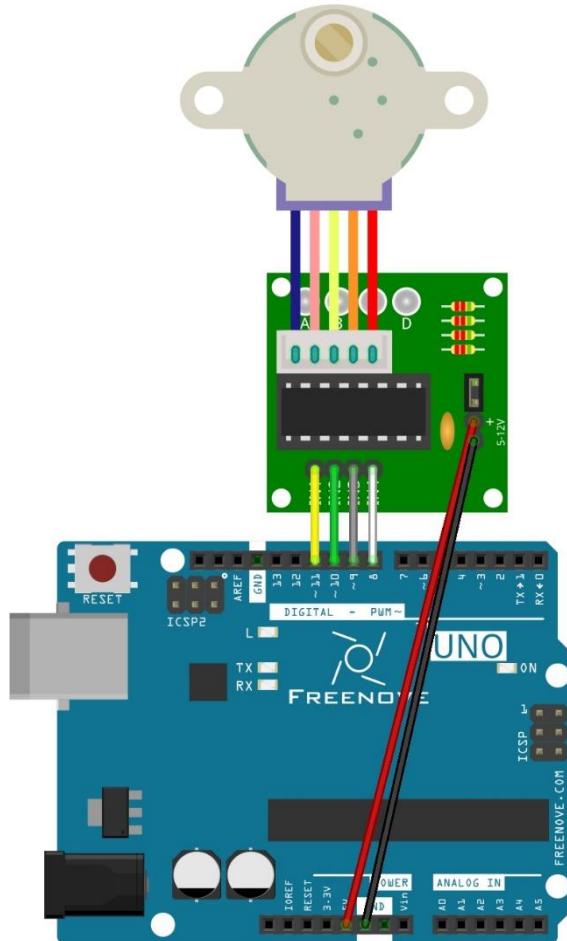
## Circuit

Use pin D11, D10, D9, D8 on Freenove UNO to control the ULN2003 stepper motor driver, and connect it to the stepper motor.

Schematic diagram



Hardware connection



## Sketch

### Sketch 18.1.1

Now write code to control the stepper motor through ULN2003 stepper motor driver.

```
1 // Connect the port of the stepper motor driver
2 int outPorts[] = {11, 10, 9, 8};
3
4 void setup() {
5     // set pins to output
6     for (int i = 0; i < 4; i++) {
7         pinMode(outPorts[i], OUTPUT);
8     }
9 }
10
11 void loop()
12 {
13     // Rotate a full turn
14     moveSteps(true, 32 * 64, 2);
15     delay(1000);
16     // Rotate a full turn towards another direction
17     moveSteps(false, 32 * 64, 2);
18     delay(1000);
19 }
20
21 void moveSteps(bool dir, int steps, byte ms) {
22     for (int i = 0; i < steps; i++) {
23         moveOneStep(dir); // Rotate a step
24         delay(ms); // Control the speed
25     }
26 }
27
28 void moveOneStep(bool dir) {
29     // Define a variable, use four low bit to indicate the state of port
30     static byte out = 0x01;
31     // Decide the shift direction according to the rotation direction
32     if (dir) { // ring shift left
33         out != 0x08 ? out = out << 1 : out = 0x01;
34     }
35     else { // ring shift right
36         out != 0x01 ? out = out >> 1 : out = 0x08;
37     }
38     // Output singal to each port
39     for (int i = 0; i < 4; i++) {
```

```

40     digitalWrite(outPorts[i], (out & (0x01 << i)) ? HIGH : LOW);
41 }
42 }
```

In the code, we define a function to make the motor rotate a step. And the parameter determines the rotation direction of the stepper motor.

```

28 void moveOneStep(bool dir) {
...
42 }
```

A variable is defined in this function and we use four low bits to show the state of 4 ports. These ports are connected in order so this variable can be assigned to 0x01 and we can use the shifting method to change bit of the connected port.

```

29 // Define a variable, use four low bit to indicate the state of port
30 static byte out = 0x01;
31 // Decide the shift direction according to the rotation direction
32 if (dir) { // ring shift left
33     out != 0x08 ? out = out << 1 : out = 0x01;
34 }
35 else { // ring shift right
36     out != 0x01 ? out = out >> 1 : out = 0x08;
37 }
```

Then change the state of the port according to the above variables.

```

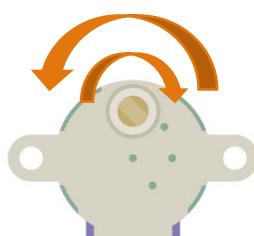
38 // Output signal to each port
39 for (int i = 0; i < 4; i++) {
40     digitalWrite(outPorts[i], (out & (0x01 << i)) ? HIGH : LOW);
41 }
```

We define a function to control the step motor to rotate several steps and control the direction and speed through parameters. Call it directly in the loop () function.

```

21 void moveSteps(bool dir, int steps, byte ms) {
22     for (int i = 0; i < steps; i++) {
23         moveOneStep(dir); // Rotate one step
24         delay(ms); // Control the speed
25     }
26 }
```

Verify and upload the code, and you will see the step motor rotate a full turn, then repeat this process in reverse direction.



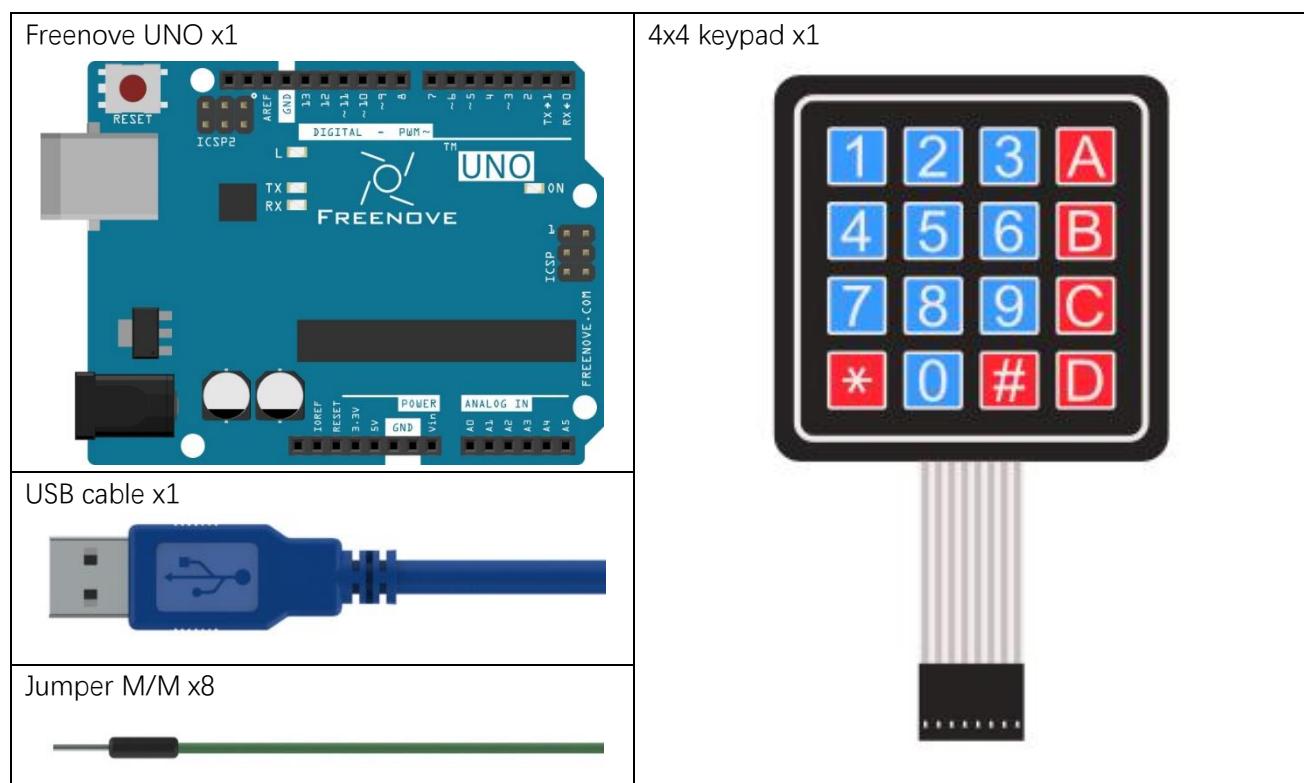
# Chapter 19 Keypad

We've already learned and used the push button before, now let's try a device integrated with many push buttons, keypad.

## Project 19.1 Get Input Characters

First, try to understand the keypad, and get the input characters.

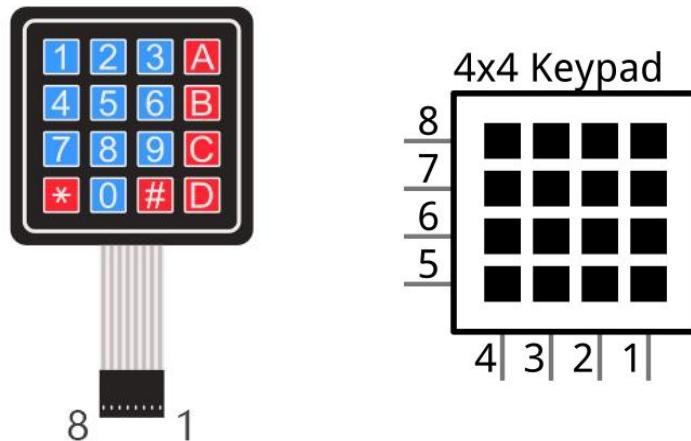
### Component list



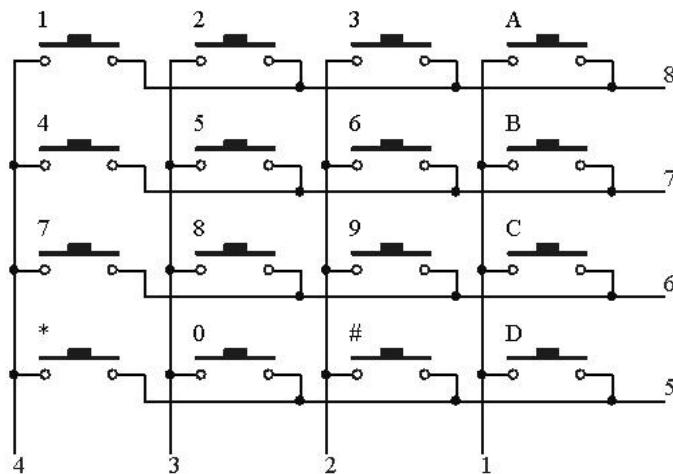
## Component knowledge

### 4x4 keypad

Keypad is a kind of component integrated with multiple keys. The following is a 4x4 keypad, it integrates 16 keys:



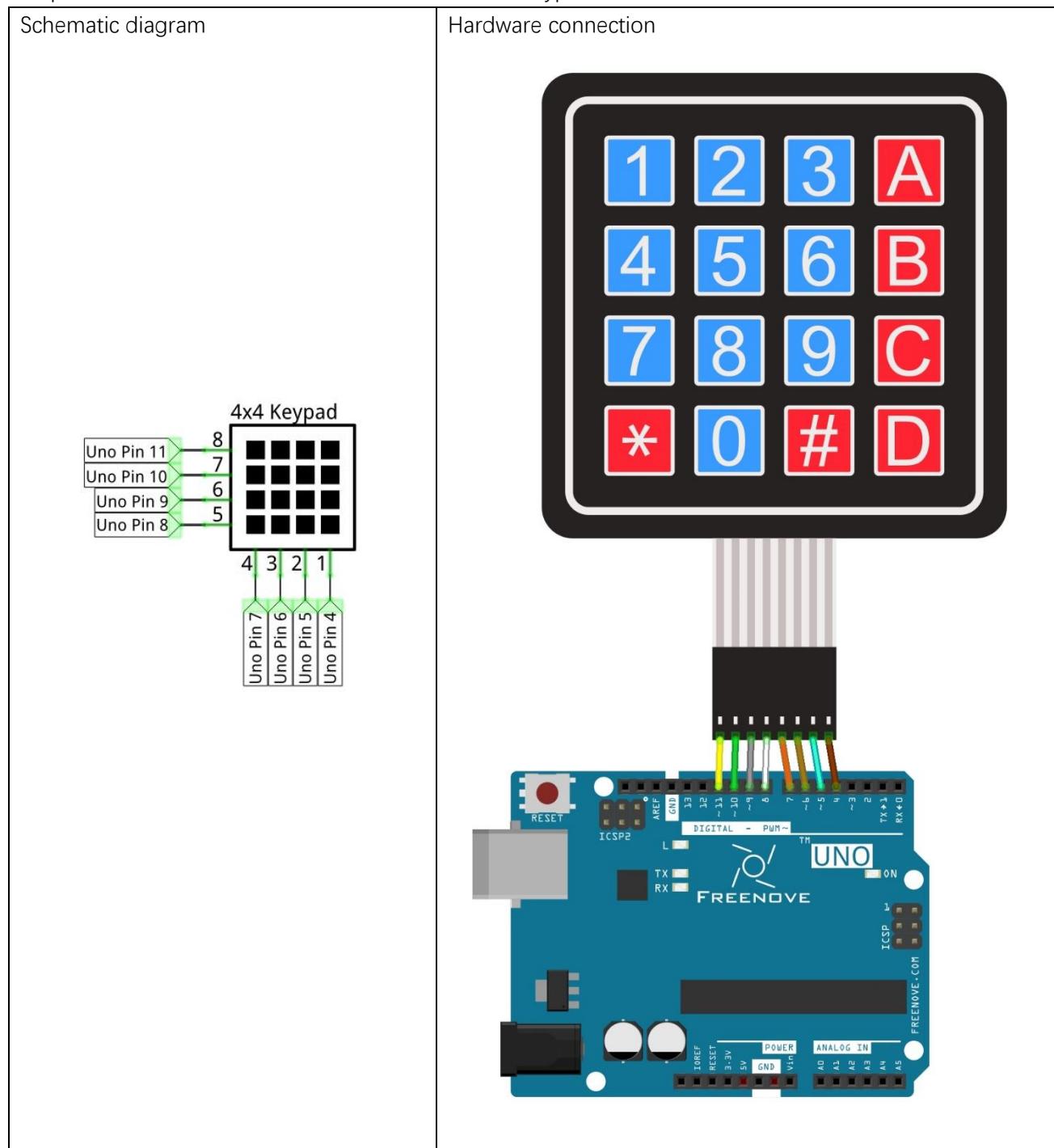
The internal circuit of 4x4 keypad is as follows, such a connection can reduce the occupation of processor port.



Detect whether the key of each column or row is pressed or not contiously during the practical use. For example, make column 1 low level and detect the state of row 5,6,7,8 to judge whether key A, B, C, D are pressed or not. Then detect other three columns in turn. Detect this in cycle way, and you will get the state of all keys.

## Circuit

Use pin D11 - D4 on Freenove UNO to connect 4x4 keypad.



## Sketch

### Sketch 19.1.1

Before writing code, we need to import Keypad.zip library. This library can facilitate our operation of keypad. Now write the code to obtain the keypad characters, and sent them to the serial port.

```
1 #include <Keypad.h>
2
3 // define the symbols on the buttons of the keypad
4 char keys[4][4] = {
5     {'1', '2', '3', 'A'},
6     {'4', '5', '6', 'B'},
7     {'7', '8', '9', 'C'},
8     {'*', '0', '#', 'D'}
9 };
10
11 byte rowPins[4] = {11, 10, 9, 8}; // connect to the row pinouts of the keypad
12 byte colPins[4] = {7, 6, 5, 4}; // connect to the column pinouts of the keypad
13
14 // initialize an instance of class Keypad
15 Keypad myKeypad = Keypad(makeKeymap(keys), rowPins, colPins, 4, 4);
16
17 void setup() {
18     Serial.begin(9600); // Initialize the serial port and set the baud rate to 9600
19     Serial.println("UNO is ready!"); // Print the string "UNO is ready!"
20 }
21
22 void loop() {
23     // Get the character input
24     char keyPressed = myKeypad.getKey();
25     // If there is a character input, sent it to the serial port
26     if (keyPressed) {
27         Serial.println(keyPressed);
28     }
29 }
```

In the code, we use the Keypad class provided by the Keypad library to operate keypad. The following is to instantiate a keypad object, and the last two parameter represents the number of the row and column of the keypad.

6	Keypad myKeypad = Keypad(makeKeymap(keys), rowPins, colPins, 4, 4);
---	---------------------------------------------------------------------

The two-dimensional array records the keypad characters, and these characters can be returned when press the keyboard

```
4   char keys[4][4] = {
5     {'1', '2', '3', 'A'},
6     {'4', '5', '6', 'B'},
7     {'7', '8', '9', 'C'},
8     {'*', '0', '#', 'D'}
9 }
```

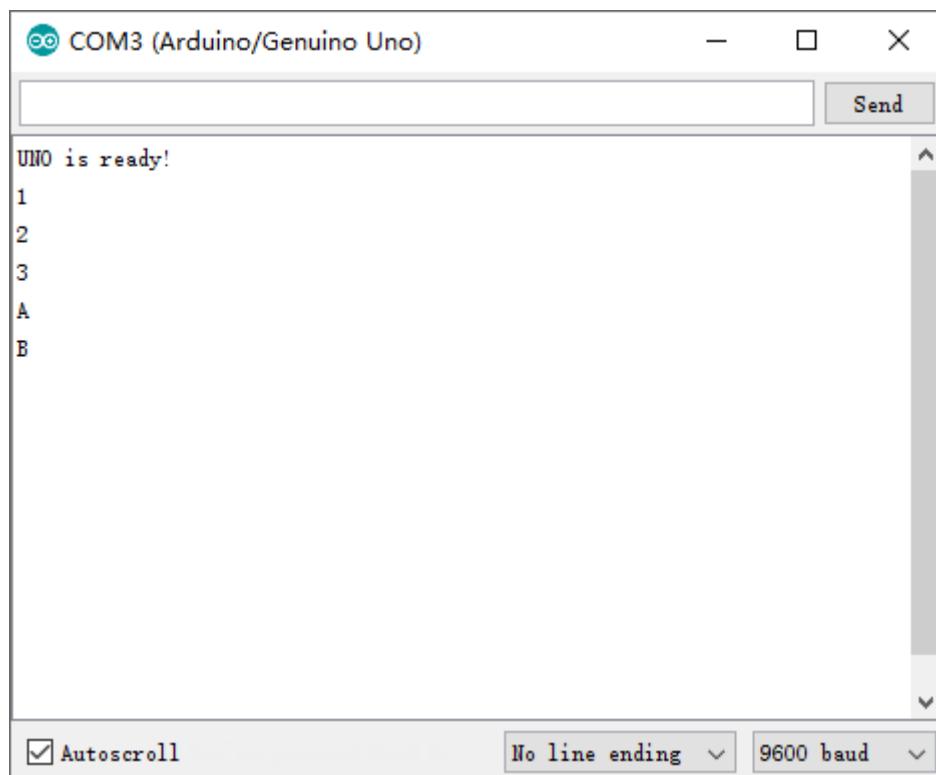
This two array record the row and column's connection pins of keypad.

```
11 byte rowPins[4] = {11, 10, 9, 8}; // connect to the row pinouts of the keypad
12 byte colPins[4] = {7, 6, 5, 4}; // connect to the column pinouts of the keypad
```

Send the input that get from the keyboard to the computer via the serial port in function loop().

```
22 void loop() {
23   // Get the character input
24   char keyPressed = myKeypad.getKey();
25   // If there is a character input, sent it to the serial port
26   if (keyPressed) {
27     Serial.println(keyPressed);
28   }
29 }
```

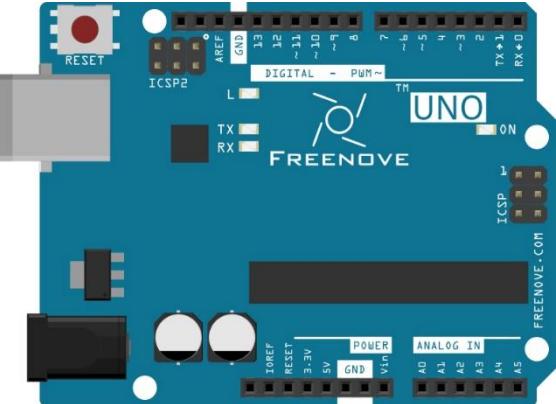
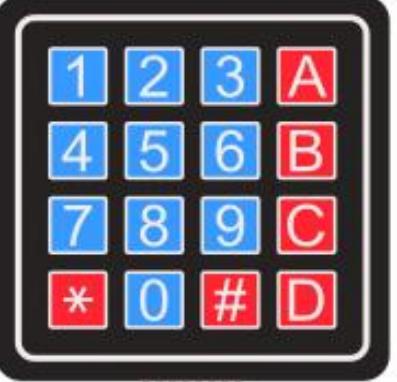
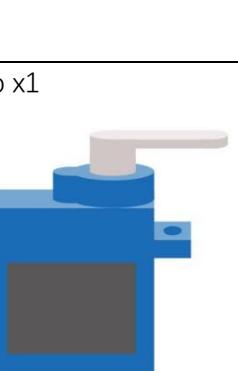
Verify and upload the code, open the Serial Monitor and press the keypad, then you will see the character you press is printed out.



## Project 19.2 Combination Lock

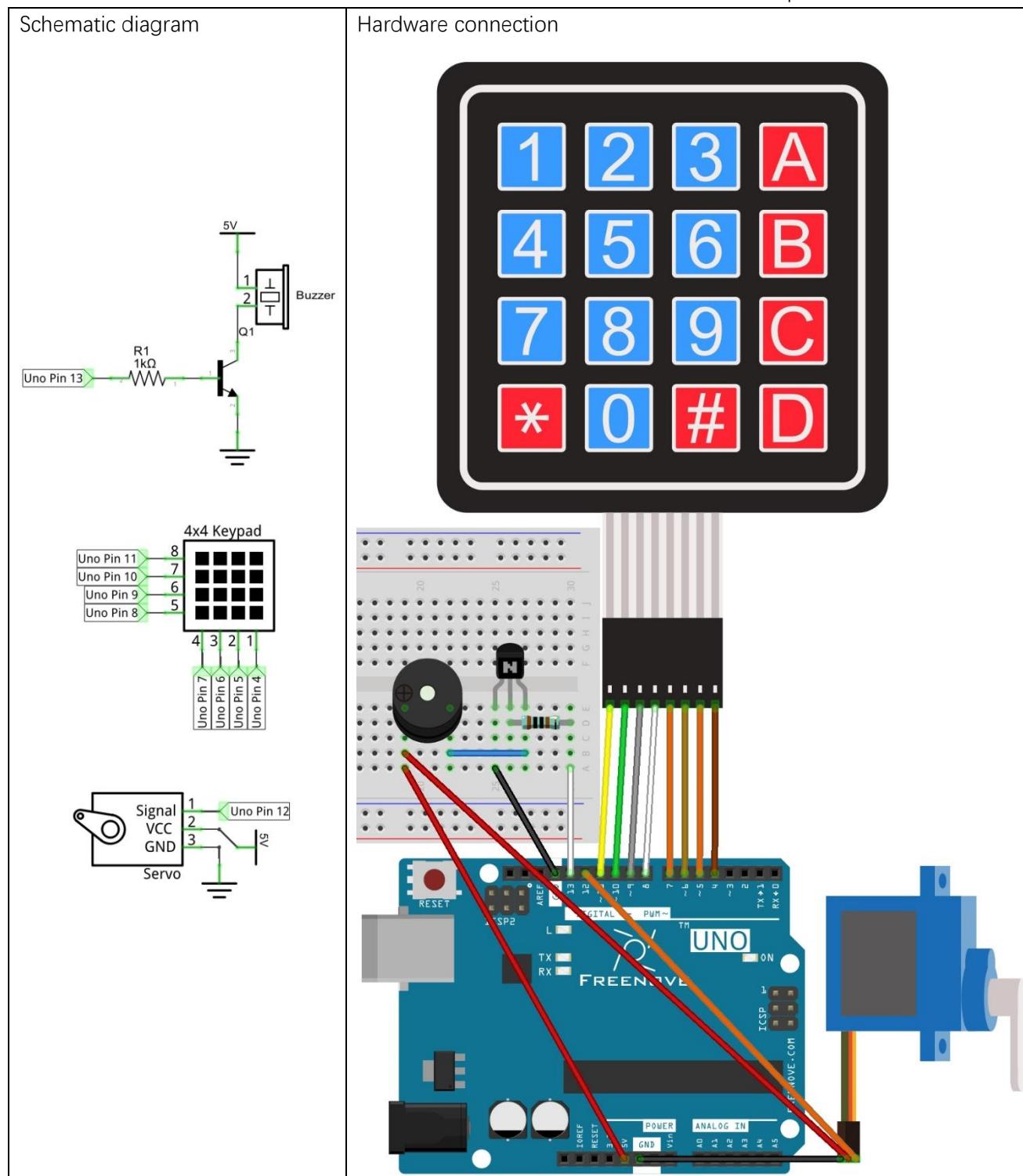
Now, let's try to use keypad to make a combination lock.

### Component list

Freenove UNO x1	4x4 keypad x1		
			
USB cable x1			
			
Jumper M/M x15			
NPN transistor x1	Active buzzer x1	Resistor 1kΩ x1	Servo x1
			

## Circuit

Use pin D11 - D4 on Freenove UNO to connect 4x4 keypad, and use D13 to drive buzzer, D12 to drive servo. Servo can be used to drive the mechanical switch and turn on the switch when the password is correct.



## Sketch

### Sketch 19.2.1

Now write the code to obtain the keypad characters, and compare it with the preset password. If the input is correct, drive servo moves, otherwise make an input error alarm.

```
1 #include <Keypad.h>
2 #include <Servo.h>
3
4 // define the symbols on the buttons of the keypad
5 char keys[4][4] = {
6     {'1', '2', '3', 'A'},
7     {'4', '5', '6', 'B'},
8     {'7', '8', '9', 'C'},
9     {'*', '0', '#', 'D'}
10};
11
12 byte rowPins[4] = {11, 10, 9, 8}; // connect to the row pinouts of the keypad
13 byte colPins[4] = {7, 6, 5, 4}; // connect to the column pinouts of the keypad
14
15 // initialize an instance of class NewKeypad
16 Keypad myKeypad = Keypad(makeKeymap(keys), rowPins, colPins, 4, 4);
17
18 Servo myservo; // Create servo object to control a servo
19 int servoPin = 12; // Define the servo pin
20 int buzzerPin = 13; // Define the buzzer pin
21
22 char passWord[] = {'1', '2', '3', '4'}; // Save the correct password
23
24 void setup() {
25     myservo.attach(servoPin); // Attaches the servo on servoPin to the servo object
26     myservo.write(45); // Let the steering gear move to the start position
27     pinMode(buzzerPin, OUTPUT);
28 }
29
30 void loop() {
31     static char keyIn[4]; // Save the input character
32     static byte keyInNum = 0; // Save the number of input characters
33     char keyPressed = myKeypad.getKey(); // Get the character input
34     // Handle the input characters
35     if (keyPressed) {
36         // Make a prompt tone each time press the key
37         digitalWrite(buzzerPin, HIGH);
38         delay(100);
39 }
```

```

39   digitalWrite(buzzerPin, LOW);
40   // Save the input characters
41   keyIn[keyInNum++] = keyPressed;
42   // Judge the correctness after input
43   if (keyInNum == 4) {
44       bool isRight = true;           // Save password is correct or not
45       for (int i = 0; i < 4; i++) { // Judge each character of the password is correct
or not
46           if (keyIn[i] != passWord[i])
47               isRight = false;        // Mark wrong passageword if there is any wrong
character
48       }
49       if (isRight) {              // If the input password is right
50           myservo.write(135);    // Open the switch
51           delay(2000);         // Delay a period of time
52           myservo.write(45);    // Close the switch
53       }
54       else {                   // If the input password is wrong
55           digitalWrite(buzzerPin, HIGH); // Make a wrong password prompt tone
56           delay(500);
57           digitalWrite(buzzerPin, LOW);
58       }
59       keyInNum = 0; // Reset the number of the input characters to 0
60   }
61 }
62 }
```

Based on the last section, this code increases the comparison function, the according action will be different when the password is right or wrong.

We first define the correct password with four characters.

```
22 char passWord[] = {'1', '2', '3', '4'}; // save the right password
```

Make a prompt sound every time when the key is pressed and save the pressed characters.

```

35   if (keyPressed) {
36       // Make a prompt tone each time press the key.
37       digitalWrite(buzzerPin, HIGH);
38       delay(100);
39       digitalWrite(buzzerPin, LOW);
40       // Save the input characters
41       keyIn[keyInNum++] = keyPressed;
...
61 }
```

Compare with the preset password after 4 characters are input and adopt the corresponding operation.

```
43     if (keyInNum == 4) {  
44         bool isRight = true;           // Sav password is correct or not  
45         for (int i = 0; i < 4; i++) {   // Judge each character of the password is correct  
or not  
46             if (keyIn[i] != passWord[i])  
47                 isRight = false;        // Mark wrong passageword if there is any wrong  
character.  
48         }  
49         if (isRight) {               // If the input password is right  
50             myservo.write(135);       // Open the switch  
51             delay(2000);           // Delay a period of time  
52             myservo.write(45);       // Close the switch  
53         }  
54         else {                   // If the input password is wrong  
55             digitalWrite(buzzerPin, HIGH); // Make a wrong password prompt tone  
56             delay(500);  
57             digitalWrite(buzzerPin, LOW);  
58         }  
59         keyInNum = 0; // Reset the number of the input characters to 0.  
60     }
```

Verify and upload the code, and press the keypad to input password with 4 characters. If the input is correct, the servo will move to a certain degree, then return to the original position. If the input is error, an input error alarm will be generated.

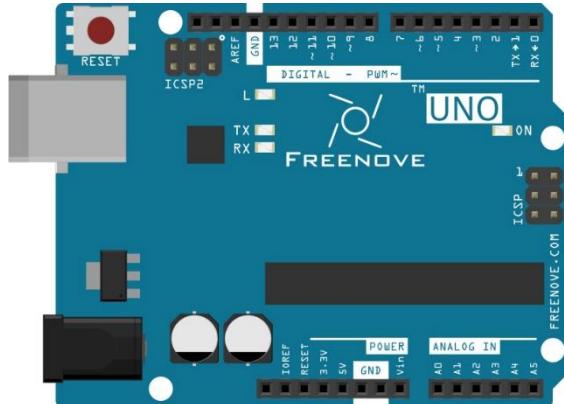
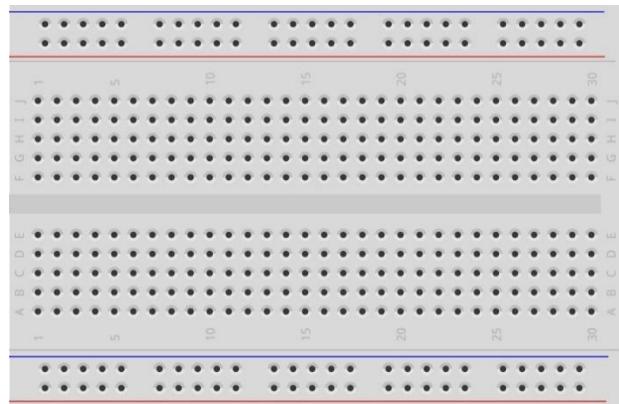
# Chapter 20 Vibration Switch

Vibration switch is a component that can detect vibration. We will use this sensor later.

## Project 20.1 Detect Vibration

Now let's try to use vibration switch to detect vibration.

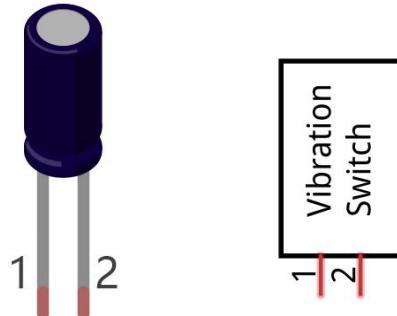
### Component list

Freenove UNO x1		Breadboard x1	
USB cable x1		Jumper F/M x2	
NPN transistorx1		Jumper M/M x4	
Active buzzer x1		Resistor 1kΩ x1	
Vibration switch x1			

## Component knowledge

### Vibration switch

Vibration switch is a kind of sensor that can detect vibration. When the vibration amplitude is greater than the critical value, two pins of vibration switch will be switched on.



The internal circuit of the Vibration switch is as follows, one of the pin is connected with metal bar and another pin is connected with spring. The spring will swing when the vibration occurs. When the vibration is strong enough, the spring will contact with the metal bar to make the two pins connected with each other.



## Circuit knowledge

### Digital pins with interrupts

Some of the Arduino digital pins can be configured to interrupt mode, which can trigger interrupt event and execute interrupt function.

Pins D2 and D3 of UNO can be configured to interrupt mode. Conditions that trigger interrupt can be configured to:

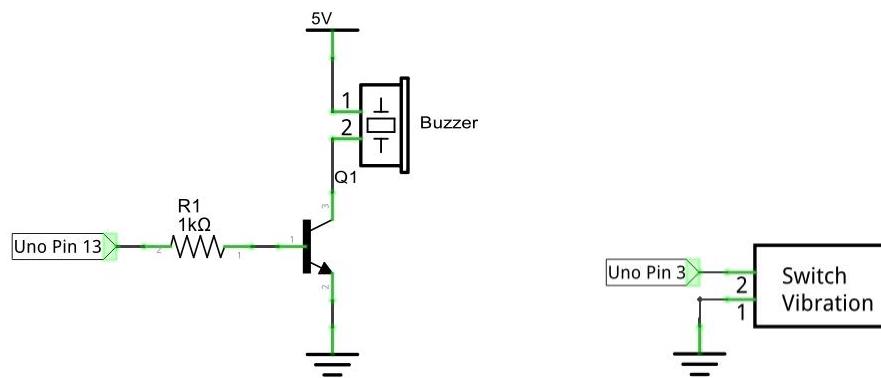
Condition	Function
LOW	to trigger the interrupt whenever the pin is low
CHANGE	to trigger the interrupt whenever the pin changes value
RISING	to trigger when the pin goes from low to high
FALLING	to trigger for when the pin goes from high to low

Interrupts are useful for making things happen automatically in microcontroller programs, and can help solve timing problems. Good tasks for using an interrupt may include reading a rotary encoder, or monitoring user input.

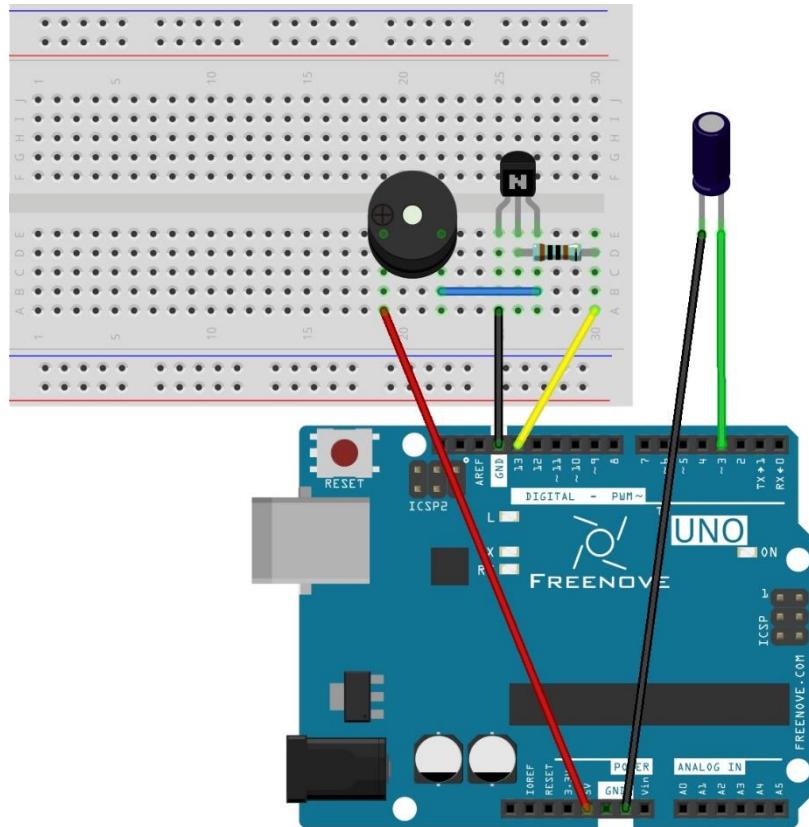
## Circuit

Use D3 pin on Freenove UNO to connect vibration switch, and D13 pin to drive buzzer.

Schematic diagram



Hardware connection



## Sketch

### Sketch 20.1.1

Now write code to detect whether vibration switch is conducted and make a buzzer sound when it is conducted.

```

1 int buzzerPin = 13; // Define the buzzer pin
2 int switchPin = 3; // Define the vibration switch pin, which can cause interrupt
3
4 bool isVibrate = false;
5
6 void setup() {
7     pinMode(buzzerPin, OUTPUT); // Set the buzzer pin to output mode
8     pinMode(switchPin, INPUT_PULLUP); // Set the vibration switch pin to pull up input mode
9     // Set interrupt, if vibration switch pin change from high level to low level, vibrate
function will be called
10    attachInterrupt(digitalPinToInterrupt(switchPin), vibrate, FALLING);
11 }
12
13 void loop() {
14     if (isVibrate) { // If the interrupt function is triggered
15         isVibrate = false; // Marked no trigger again
16         digitalWrite(buzzerPin, HIGH); // Open the buzzer
17         delay(50); // Delay for a period of time
18     }
19     else // Else close the buzzer
20         digitalWrite(buzzerPin, LOW);
21 }
22
23 void vibrate() {
24     isVibrate = true; // Marked as the trigger
25 }
```

This code configures the pin that connected to vibration switch to be triggered by falling edge, that is, FALLING mode.

10	attachInterrupt(digitalPinToInterrupt(switchPin), vibrate, FALLING);
----	----------------------------------------------------------------------

attachInterrupt(interrupt, ISR, mode);
----------------------------------------

This function is used to configure the digital pin's interrupt mode. Each parameter of function is as follows:

interrupt: the number of the interrupt (int). Normally you should use digitalPinToInterruption(pin) to translate the actual digital pin to the specific interrupt number.

ISR: the ISR to call when the interrupt occurs; this function must take no parameters and return nothing.

This function is sometimes referred to as an interrupt service routine.

mode: defines when the interrupt should be triggered.

And we configure the pins that connected with vibration switch into pull up input mode. This way can ensure a high level of vibration switch when not connected, and low level when connected. So it can cause interrupt.

```
8     pinMode(switchPin, INPUT_PULLUP); // Set the vibration switch pin to pull up input mode
```

The following is the interrupt function, and it will be executed when the interrupt is triggered.

```
23 void vibrate() {  
24     isVibrate = true; // Marked as the trigger  
25 }
```

Interrupt function should keep short, so we use a variable "isVibrate" to record whether the interrupt is triggered, and dispose it in the loop() function. And the buzzer will be connected if the interrupt is triggered.

```
14 if (isVibrate) { // If the interrupt function is triggered.  
15     isVibrate = false; // Marked no trigger again  
16  
17     digitalWrite(buzzerPin, HIGH); // Open the buzzer  
18     delay(50); // Delay for a period of time  
19 }  
20 else // Else close the buzzer  
21     digitalWrite(buzzerPin, LOW);
```

Verify and upload the code and tap on the vibration switch, then the buzzer will make a sound.

# Chapter 21 Infrared Remote

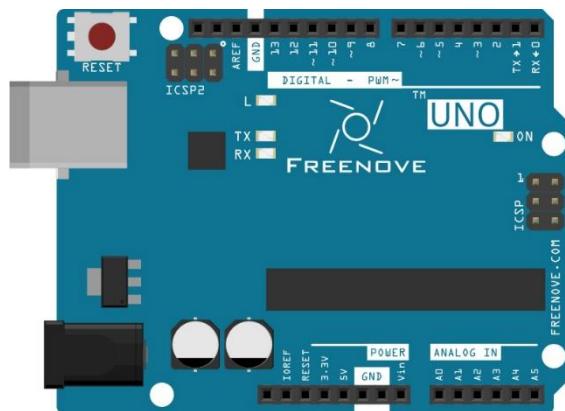
In this chapter, we'll learn how to use an infrared remote control, and control a LED.

## Project 21.1 Infrared Remote Control

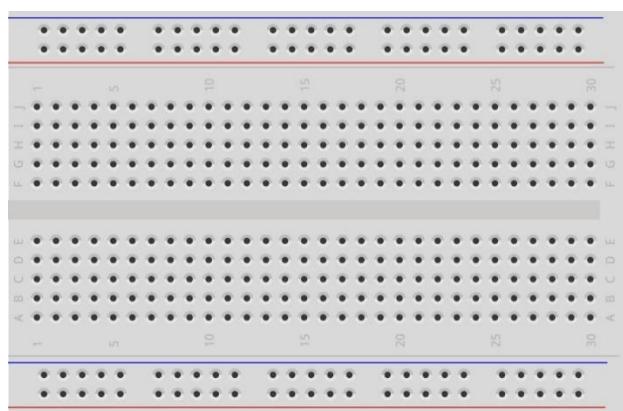
First, we need to understand how infrared remote control works, then get the command sent from infrared remote control.

### Component list

Freenove UNO x1



Breadboard x1



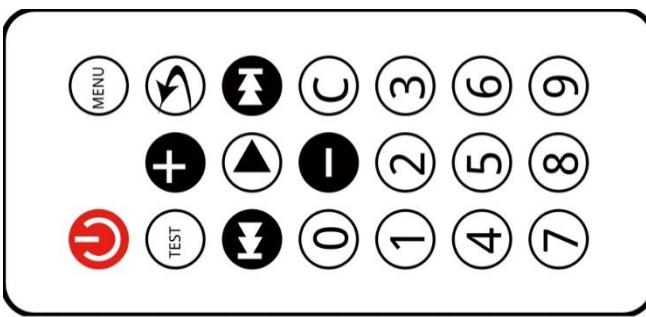
USB cable x1



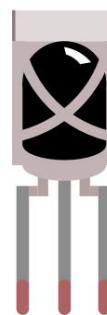
Jumper M/M x4



Infrared remote x1



Infrared receiver x1



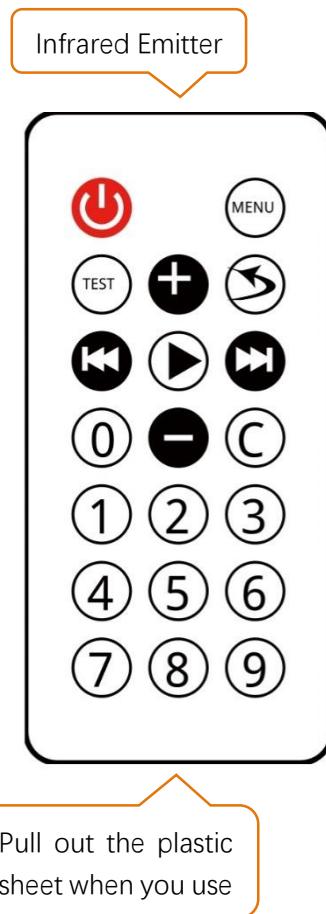
Resistor 10kΩ x1



## Component knowledge

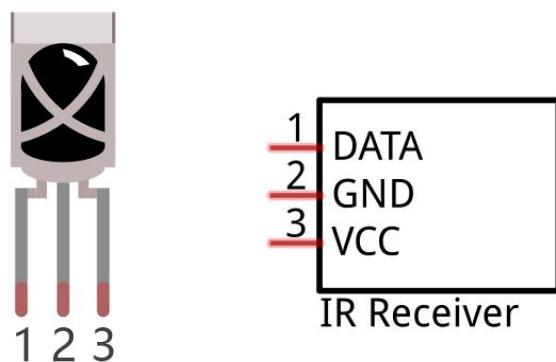
### Infrared remote

Infrared(IR) remote control is a device with a certain number of buttons. Pressing down different buttons will make the infrared emission tube, which is located in the front of the remote control, send infrared with different encoding. Infrared remote control technology is widely used, such as TV, air conditioning, etc. Thus, it makes it possible for you to switch TV programs and adjust the temperature of the air conditioning away from them. The remote control we use is shown below:



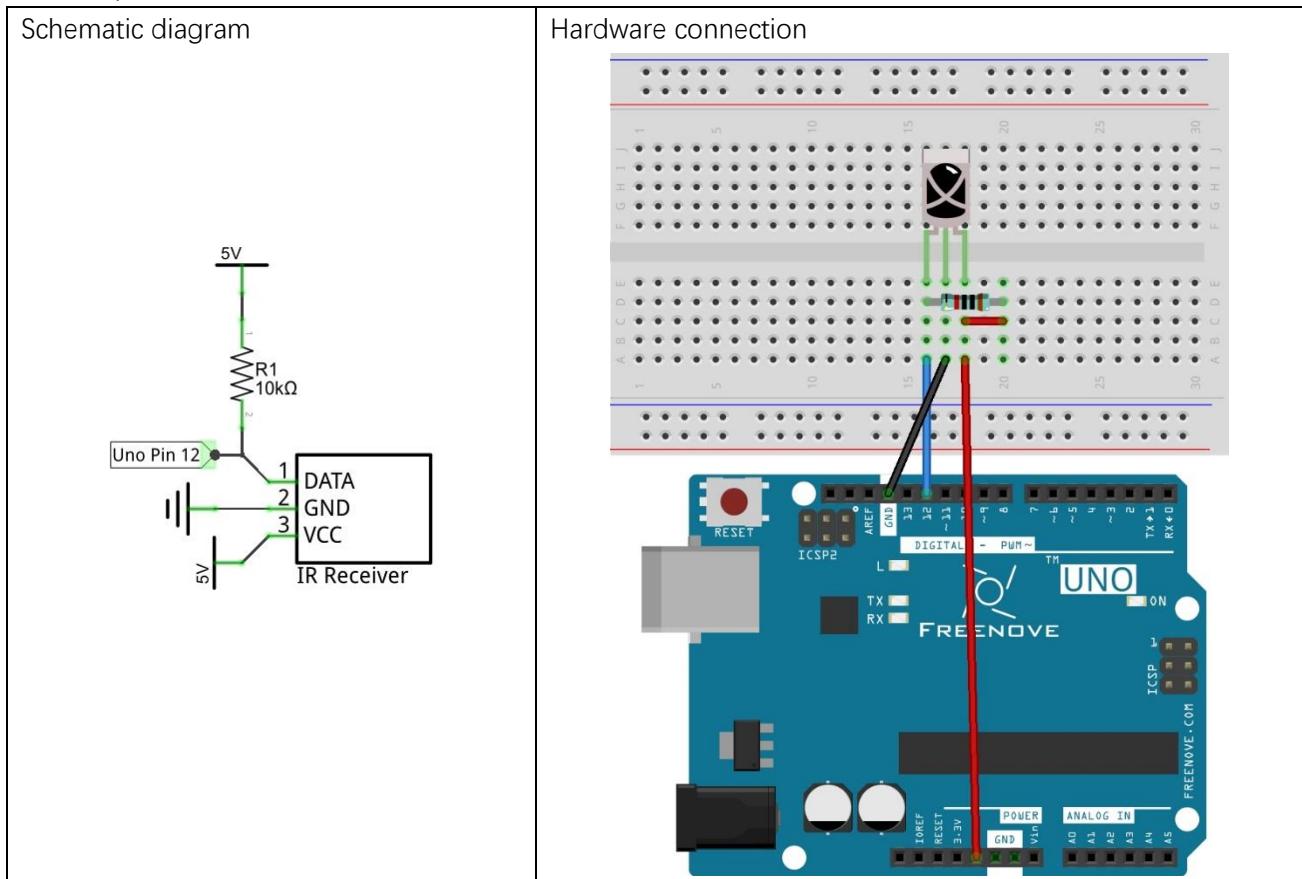
### Infrared receiver

Infrared(IR) receiver is a component which can receive the infrared light, so we can use it to detect the signal emitted by the infrared remote control. DATA pin here outputs the received infrared signal.



## Circuit

Use D12 pin on Freenove UNO to connect IR receiver.



## Sketch

### Sketch 21.1.1

Before writing code, we need to import the `IRremote.zip` library first. Then you can control IR receiver more conveniently with this library.

Now, write code to get the command sent from IR remote control, and sent it to the serial port.

```

1 #include <IRremote.h>
2
3 int RECV_PIN = 12;           // Infrared receiving pin
4 IRrecv irrecv(RECV_PIN);    // Create a class object used to receive class
5 decode_results results;     // Create a decoding results class object
6
7 void setup()
8 {
9     Serial.begin(9600); // Initialize the serial port and set the baud rate to 9600
10    Serial.println("UNO is ready!"); // Print the string "UNO is ready!"
11    irrecv.enableIRIn(); // Start the receiver
12 }

```

```

13
14 void loop() {
15     if (irrecv.decode(&results)) { // Waiting for decoding
16         Serial.println(results.value, HEX); // Print out the decoded results
17         irrecv.resume(); // Receive the next value
18     }
19     delay(100);
20 }
```

We use the IRrecv class provided by the IRemote library to control IR receiver in this code. As shown below, instantiate one IRrecv object, and the parameter represents the pin connected to the IR receiver.

```
4 IRrecv irrecv(RECV_PIN); // Create a class object used to receive class
```

decode\_results class provided by the IRemote library is used to save the results of IR control decoding.

```
5 decode_results results; // Create a decoding results class object
```

Start the signal receiving in the setup() function

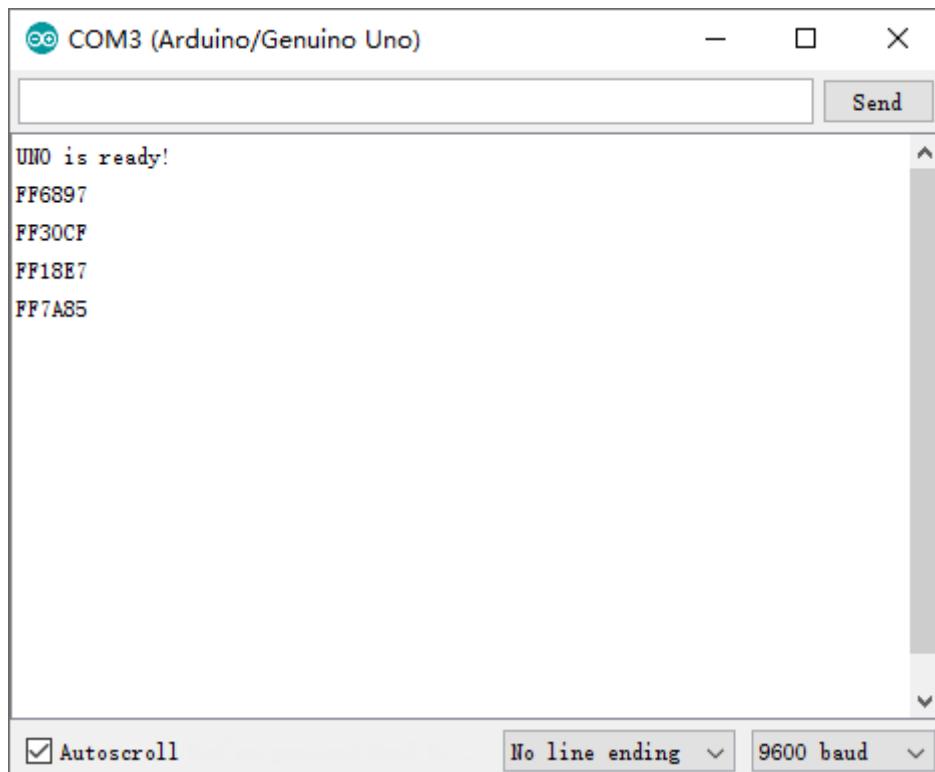
```
11 irrecv.enableIRIn(); // Start the receiver
```

In the loop() function, decode the received signal, and sent it to computer through the serial port.

```

14 void loop() {
15     if (irrecv.decode(&results)) { // Waiting for decoding
16         Serial.println(results.value, HEX); // Print out the decoded results
17         irrecv.resume(); // Receive the next value
18     }
19     delay(100);
20 }
```

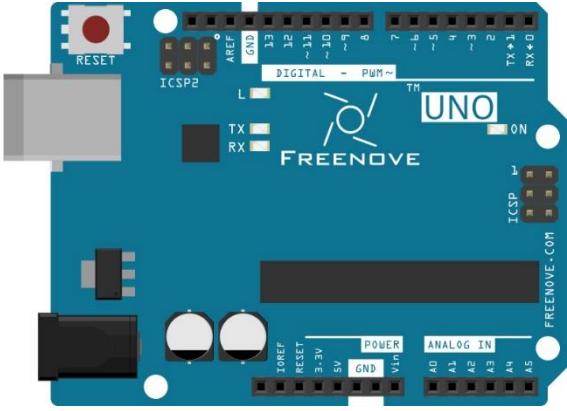
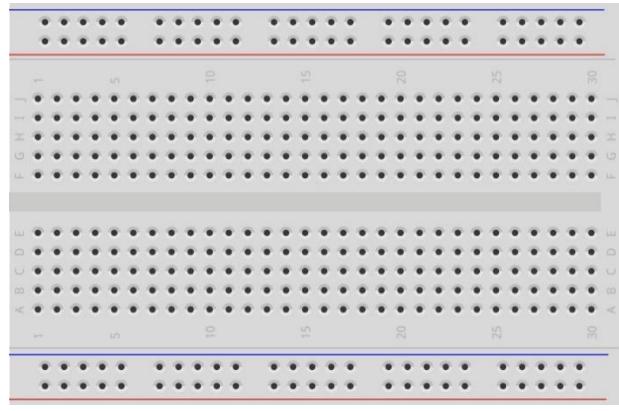
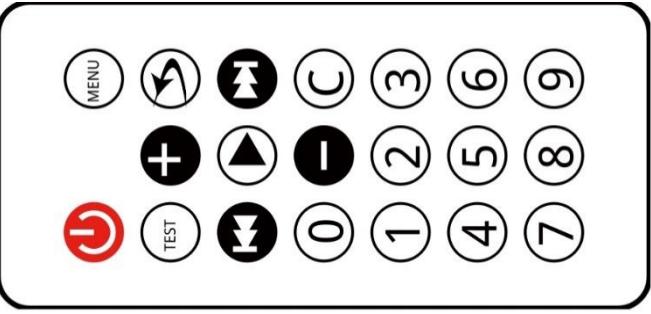
Verify and upload the code, open the Serial Monitor, and press the IR control button, then you can see the corresponding code will be printed out.



## Project 21.2 Control LED through Infrared Remote

Now, let's try to control a LED through infrared remote.

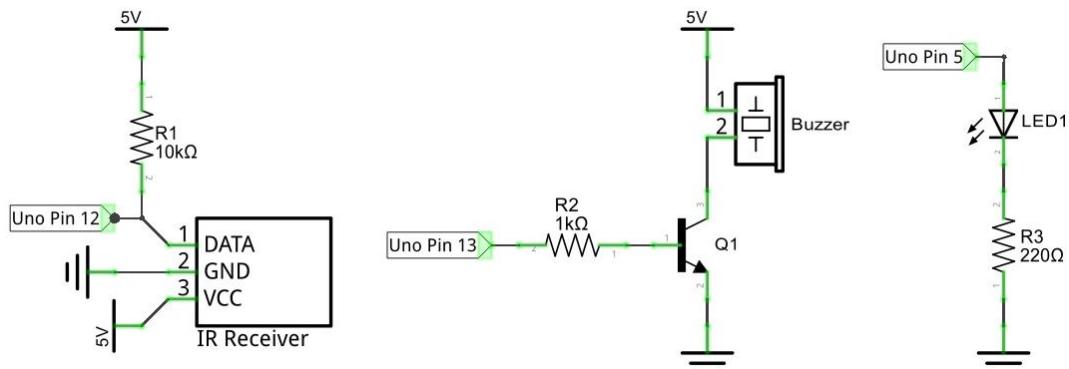
### Component list

Freenove UNO x1	Breadboard x1
	
USB cable x1	Infrared remote x1
	
Jumper M/M x10	
	
NPN transistor x1	LED x1
	
Active buzzer x1	Resistor 220Ω x1
	
Infrared receiver x1	Resistor 1kΩ x1
	
	Resistor 10kΩ x1
	

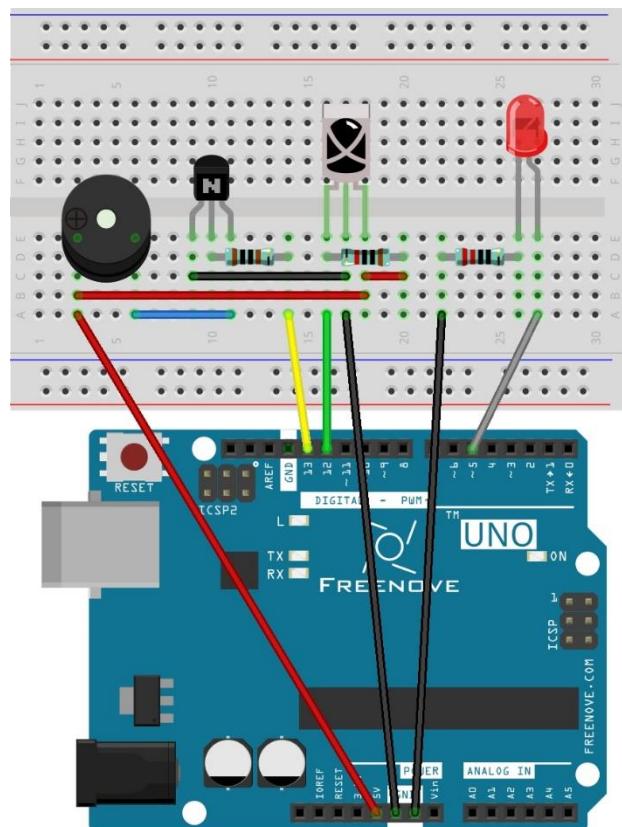
## Circuit

Connect D12 pin of Freenove UNO to IR receiver to simulate a desk lamp. And drive buzzer through D13, drive LED through D5.

Schematic diagram



Hardware connection



## Sketch

### Sketch 21.2.1

Now, write code to get the commands sent from IR remote, and control the LED light on/off or emit light with different brightness. and control the buzzer to generate a confirmation sound when it receives the command.

```
1 #include <IRremote.h>
2
3 int RECV_PIN = 12;           // Infrared receiving pin
4 IRrecv irrecv(RECV_PIN);    // Create a class object used to receive class
5 decode_results results;    // Create a decoding results class object
6
7 int ledPin = 5;             // the number of the LED pin
8 int buzzerPin = 13;         // the number of the buzzer pin
9
10 void setup()
11 {
12     Serial.begin(9600); // Initialize the serial port and set the baud rate to 9600
13     Serial.println("UNO is ready!"); // Print the string "UNO is ready!"
14     irrecv.enableIRIn(); // Start the receiver
15     pinMode(ledPin, OUTPUT); // set LED pin into output mode
16     pinMode(buzzerPin, OUTPUT); // set buzzer pin into output mode
17 }
18
19 void loop() {
20     if (irrecv.decode(&results)) { // Waiting for decoding
21         Serial.println(results.value, HEX); // Print out the decoded results
22         handleControl(results.value); // Handle the commands from remote control
23         irrecv.resume(); // Receive the next value
24     }
25 }
26
27 void handleControl(unsigned long value) {
28     // Make a sound when it receives commands
29     digitalWrite(buzzerPin, HIGH);
30     delay(100);
31     digitalWrite(buzzerPin, LOW);
32     // Handle the commands
33     switch (value) {
34         case 0xFF6897:           // Receive the number '0'
35             analogWrite(ledPin, 0); // Turn off LED
36             break;
37         case 0xFF30CF:           // Receive the number '1'
38             analogWrite(ledPin, 7); // Dimmest brightness
```

```

39     break;
40     case 0xFF18E7:           // Receive the number '2'
41         analogWrite(ledPin, 63); // Medium brightness
42         break;
43     case 0xFF7A85:           // Receive the number '3'
44         analogWrite(ledPin, 255); // Strongest brightness
45         break;
46     }
47 }
```

Based on the last section, we add some new function: control LED and buzzer.

We define a function that is used to handle the received commands. When this function is executed, make the buzzer beep first, then output PWM signals with different duty cycle to the pin connected LED according to received the commands

```

27 void handleControl(unsigned long value) {
28     // Make a sound when it receives commands
29     digitalWrite(buzzerPin, HIGH);
30     delay(100);
31     digitalWrite(buzzerPin, LOW);
32     // Handle the commands
33     switch (value) {
34         case 0xFF6897:           // Receive the number '0'
35             analogWrite(ledPin, 0); // Turn off LED.
36             break;
37         case 0xFF30CF:           // Receive the number '1'
38             analogWrite(ledPin, 7); // Dimmest brightness.
39             break;
40         case 0xFF18E7:           // Receive the number '2'
41             analogWrite(ledPin, 63); // Medium brightness.
42             break;
43         case 0xFF7A85:           // Receive the number '3'
44             analogWrite(ledPin, 255); // Strongest brightness.
45             break;
46     }
47 }
```

Each time when the command is received, the function above will be called in the loop() function.

```

19 void loop() {
20     if (irrecv.decode(&results)) { // Waiting for decoding
21         Serial.println(results.value, HEX); // Print out the decoded results
22         handleControl(results.value); // Handle the commands from remote control
23         irrecv.resume(); // Receive the next value
24     }
25 }
```

Verify and upload the code, press the button '0', '1', '2', '3' on IR remote, then you can see LED will emit light with different brightness, and the buzzer will start beeping when it receives the signal.

# Chapter 22 Temperature & Humidity Sensor

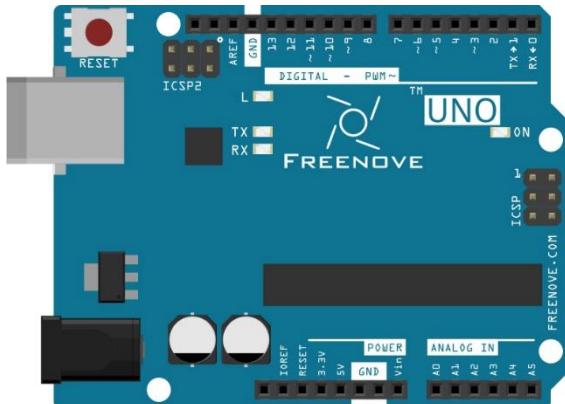
In this chapter, we will learn how to use a sensor that can detect temperature and humidity.

## Project 22.1 Temperature & Humidity Sensor

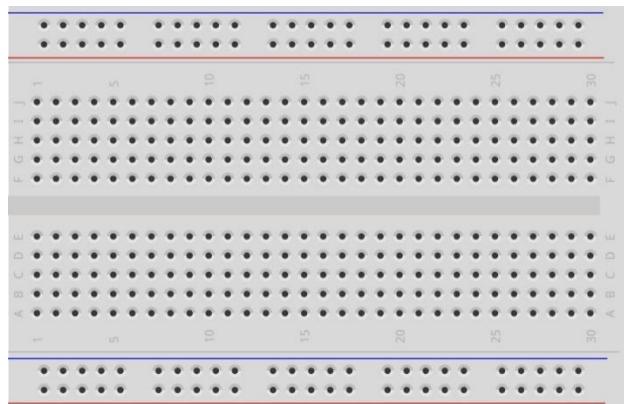
Now, let's try to get the temperature and humidity value of the current environment.

### Component list

Freenove UNO x1



Breadboard x1



USB cable x1



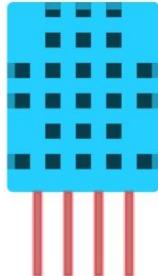
Jumper M/M x4



Resistor 10kΩ x1



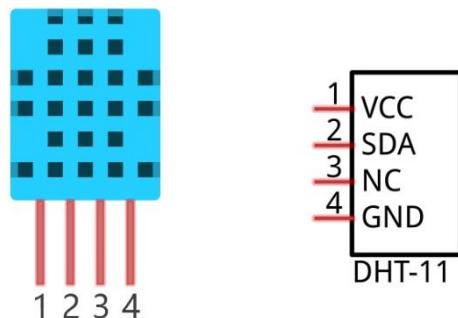
DHT11 x1



## Component knowledge

### DHT11

DHT11 is a compound temperature & humidity sensor, and the output digital signal has been calibrated inside.

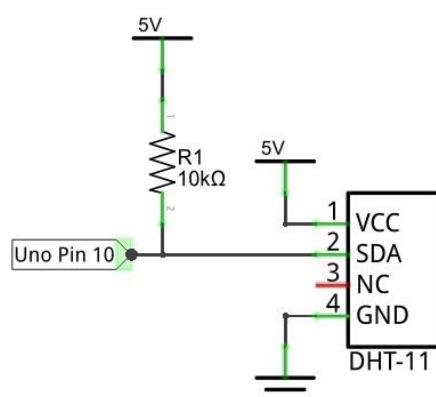


DHT11 uses customized single-line communication protocol, so we can use the library to read data more conveniently.

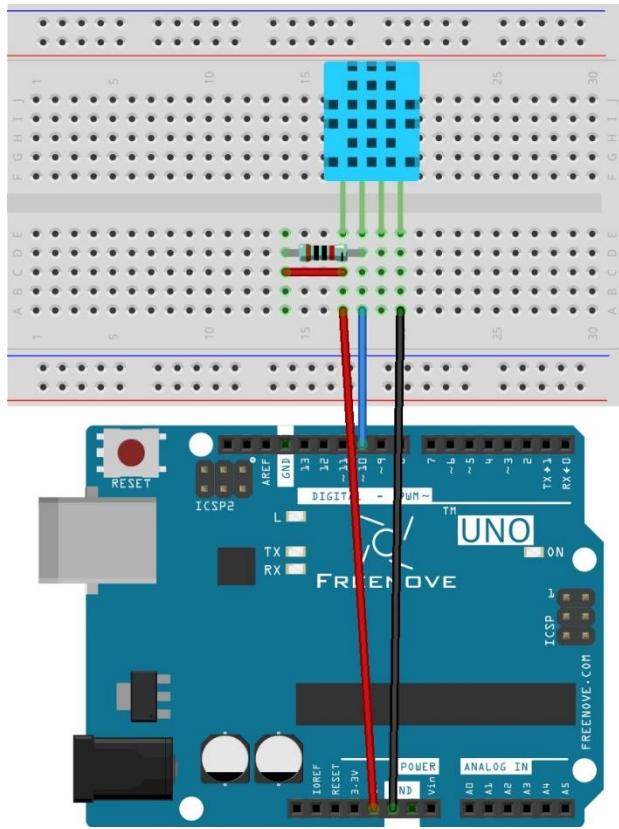
## Circuit

Use D10 pin on Freenove UNO to connect DHT11.

Schematic diagram



## Hardware connection



## Sketch

## Sketch 22.1.1

Before writing the code, we need to import the DHT.zip library first. This library can make it convenient for us to control DHT11.

Now, write code to get the temperature and humidity data measured by DHT11, and sent it to the serial port.

```

1 #include <dht.h>
2
3 dht DHT;           // create dht object
4 int dhtPin = 10;   // the number of the DHT11 sensor pin
5
6 void setup() {
7     Serial.begin(9600); // Initialize the serial port and set the baud rate to 9600
8     Serial.println("UNO is ready!"); // Print the string "UNO is ready!"
9 }
10

```

```
11 void loop() {  
12     // read DHT11 and judge the state according to the return value  
13     int chk = DHT.read11(dhtPin);  
14     switch (chk)  
15     {  
16         case DHTLIB_OK: // When read data successfully, print temperature and humidity data  
17             Serial.print("Humidity: ");  
18             Serial.print(DHT.humidity);  
19             Serial.print("%, Temperature: ");  
20             Serial.print(DHT.temperature);  
21             Serial.println("C");  
22             break;  
23         case DHTLIB_ERROR_CHECKSUM: // Checksum error  
24             Serial.println("Checksum error");  
25             break;  
26         case DHTLIB_ERROR_TIMEOUT: // Time out error  
27             Serial.println("Time out error");  
28             break;  
29         default: // Unknown error  
30             Serial.println("Unknown error");  
31             break;  
32     }  
33     delay(1000);  
34 }
```

In the code, we use the dht class provided by the DHT library to control DHT11. The following is a DHT object. As shown below, instantiate one dht object.

```
3 dht DHT; // create dht object
```

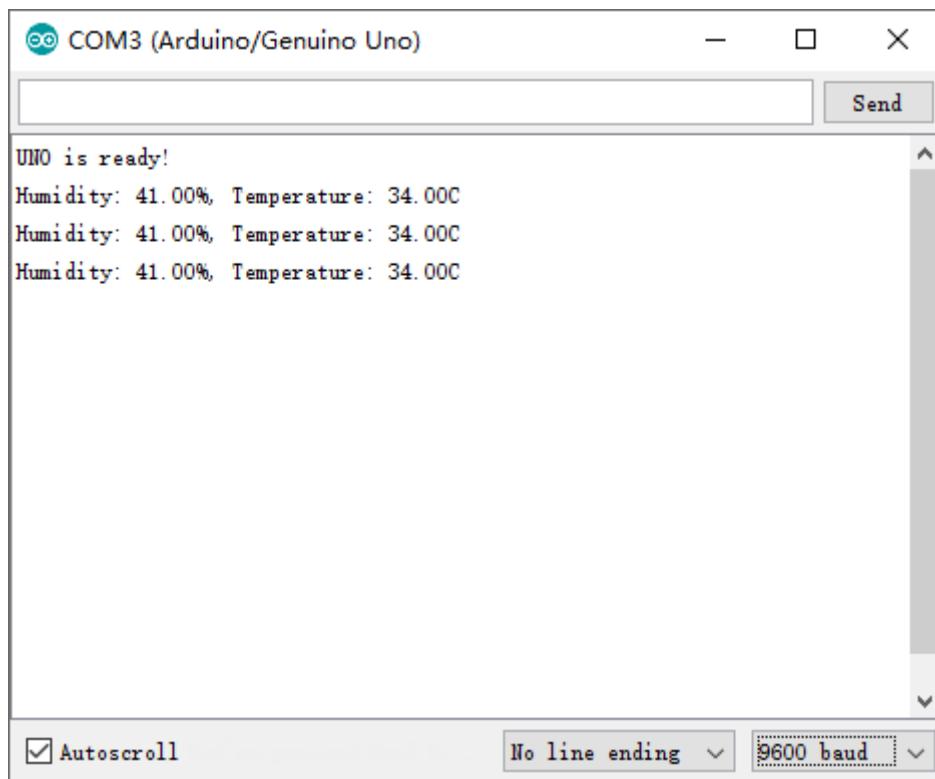
Read DHT11 data and send the result to the serial port in the loop() function.

```
12     // read DHT11 and judge the state according to the return value  
13     int chk = DHT.read11(dhtPin);  
14     switch (chk)  
15     {  
16         case DHTLIB_OK: // When read data successfully print temperature and humidity data  
17             Serial.print("Humidity: ");  
18             Serial.print(DHT.humidity);  
19             Serial.print("%, Temperature: ");  
20             Serial.print(DHT.temperature);  
21             Serial.println("C");  
22             break;  
...  
32 }
```

Send the failure reason when fail to read.

```
23     case DHTLIB_ERROR_CHECKSUM: // Checksum error
24         Serial.println("Checksum error");
25         break;
26     case DHTLIB_ERROR_TIMEOUT: // Time out error
27         Serial.println("Time out error");
28         break;
29     default:                  // Unknown error
30         Serial.println("Unknown error");
31         break;
```

Verify and upload the code, open the Serial Monitor, then you will see the temperature and humidity data sending from UNO.



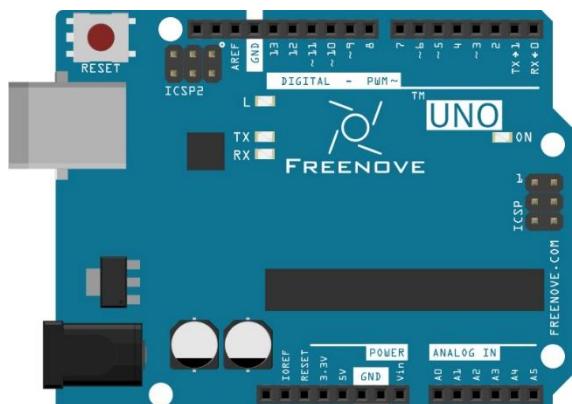
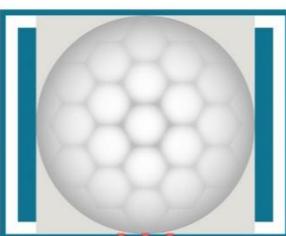
# Chapter 23 Infrared Motion Sensor

Infrared Motion Sensor is an integrated sensor that can sense the motion of the human body. Now let's try to use it.

## Project 23.1 Infrared Motion Sensor

Now, we'll use Infrared Motion Sensor to detect human motion.

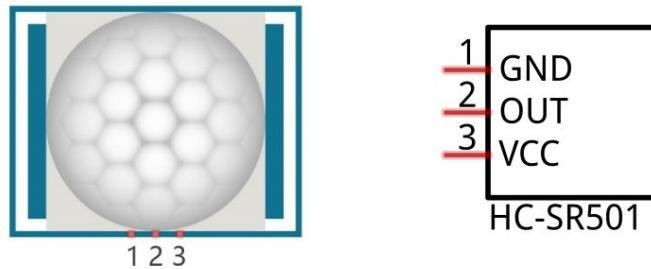
### Component list

Freenove UNO x1	USB cable x1
 The image shows the Freenove UNO R3 microcontroller board. It is a blue PCB with various components and pins. Key features include a central ATmega328P microcontroller, a USB port, a power jack, and numerous pins labeled with their respective functions such as AREF, GND, A0-A5, and digital pins 0-13. The FREENOVE logo is printed on the board.	 A standard blue USB cable with a grey plastic housing and metal contacts at the ends.
Jumper F/M x3	Infrared motion sensor x1
	 A white, hemispherical infrared motion sensor module. It has a central circular sensor array and two vertical black bars extending from its sides.

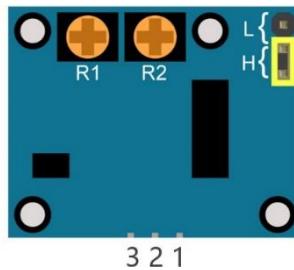
## Component knowledge

### Infrared motion sensor

Infrared Motion Sensor is an integrated sensor that can sense the motion of the human body. It can detect whether someone is on motion in the sensing range by detecting the infrared light emitted by human body. Here is the HC-SR501 infrared motion sensor:



The back of HC-SR501 infrared motion sensor is shown below:



Description:

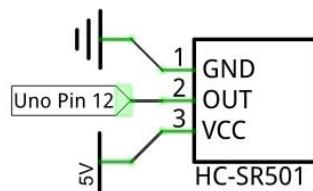
1. Working voltage: 5v-20v(DC), static current: 65uA.
2. Automatic trigger. When the body enter into the active area of sensor, the module will output high level (3.3V. Though the high level of UNO is 5v, 3.3v is identified as high level here). When body leave out, it will output high level lasting for time T, then output low level (0V). Delay time T is adjusted by potentiometer R1.
3. According to the position of jumper cap, you can choose non-repeatable trigger mode or repeatable mode:
  - L: non-repeatable trigger mode. The module output high level after sensing body, then when delay time is over, the module will output low level. And during high level time, the sensor doesn't sense body anymore.
  - H: repeatable trigger mode. The distinction from L is that it can sense body until body leaves during the period of outputting high level. And then it starts to time and output low level after delaying T time.
4. Induction block time: the induction will stay in block condition and do not induce external signal at a little time (less than delay time) after outputting high level or low level
5. Initialization time: the module needs about 1 minute to initialize after powered on. During this period, it will output high or low level alternately.
6. In consideration of feature of this sensor, when body get close or away from edgewise side, the sensor will work with high sensitively. When body get close or away in vertical direction, the sensor can't work well, which should get your attention. Sensing distance is adjusted by potentiometer R2.

We can regard this sensor as a simple inductive switch when in use.

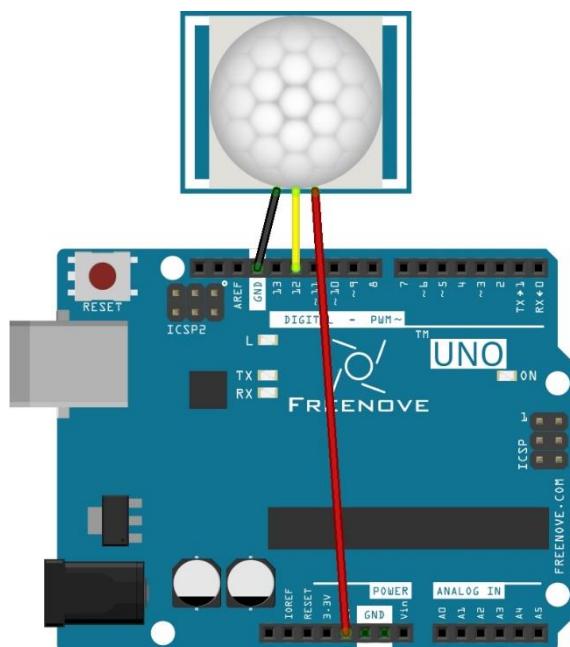
## Circuit

Use D12 pin on Freenove UNO to connect OUT pin of HC-SR501 infrared motion sensor.

Schematic diagram



Hardware connection



## Sketch

### Sketch 23.1.1

Now, write code to get the results measured by the HC-SR501 infrared motion sensor, and display that through LED.

```
1 int sensorPin = 12; // the number of the infrared motion sensor pin
2 int ledPin = 13;    // the number of the LED pin
3
4 void setup() {
5     pinMode(sensorPin, INPUT); // initialize the sensor pin as input
6     pinMode(ledPin, OUTPUT); // initialize the LED pin as output
7 }
8
9 void loop() {
10    // Turn on or off LED according to Infrared Motion Sensor
11    digitalWrite(ledPin, digitalRead(sensorPin));
12    delay(1000);           // wait for a second
13 }
```

This code is relatively simple. We have obtained the sensor's output signal, and control a LED according to it.

```
11    digitalWrite(ledPin, digitalRead(sensorPin));
```

Verify and upload the code, and put the sensor on a stationary table and wait for about a minute. Then try to get close to or away from the sensor, and observe whether the LED will be turned on or turned off automatically.

You can turn the potentiometer on the sensor to adjust the detection effect, or use different modes through changing jumper.

Apart from that, you can use this sensor to control some other modules to achieve different functions by re-editing the code, such as the induction lamp, induction door.

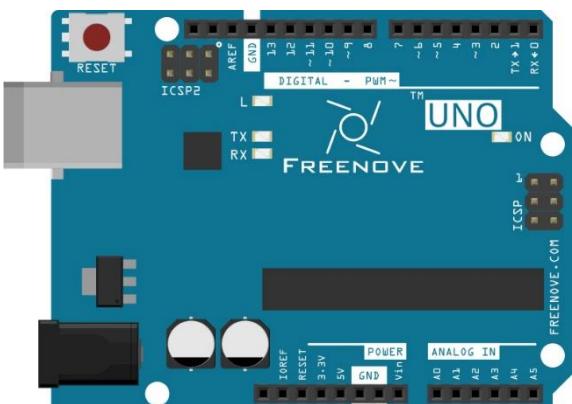
# Chapter 24 Ultrasonic Ranging

In this chapter, we learn a module which use ultrasonic to measure distance, HC-SR04 ultrasonic ranging module.

## Project 24.1 Ultrasonic Ranging

In this project, we use ultrasonic ranging module to measure distance, and print out the data in the serial port.

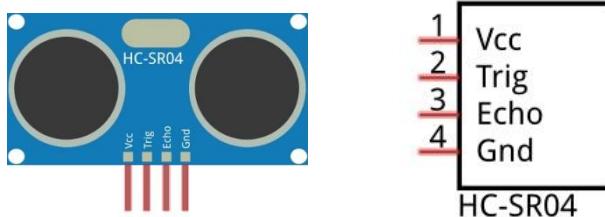
## Component List

Freenove UNO x1	USB cable x1
	
Jumper F/M x4	
Ultrasonic ranging module x1	

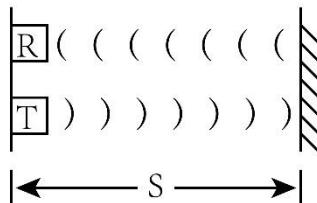
## Component Knowledge

### Ultrasonic ranging module

Ultrasonic ranging module use the principle that ultrasonic will reflect when it encounters obstacles. Ultrasonic module integrates a transmitter and a receiver. The transmitter is used to convert electrical signals (electrical energy) into sound waves (mechanical energy) and the function of the receiver is opposite.



Start counting the time when ultrasonic is transmitted. And when ultrasonic encounters an obstacle, it will reflect back. The counting will end after ultrasonic is received, and the time difference is the total time of ultrasonic from transmitting to receiving. Because the speed of sound in air is constant, and is about  $v=340\text{m/s}$ . So we can calculate the distance between the module and the obstacle:  $S=vt/2$ .



Pin description:

Pin name	Pin number	Description
Vcc	1	Positive electrode of power supply, the voltage is 5V
Trig	2	Trigger pin
Echo	3	Echo pin
Gnd	4	Negative electrode of power supply

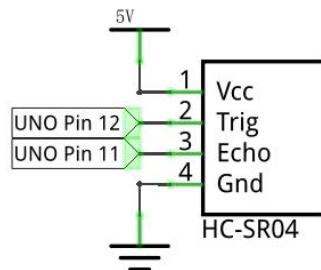
Instructions for use:

Output a high-level pulse in Trig pin lasting for least 10us. Then the the module begins to transmit ultrasonic. At the same time, the Echo pin will be pulled up. When the module receives the returned ultrasonic, the Echo pin will be pulled down. The duration of high level in Echo pin is the total time of the ultrasonic from transmitting to receiving,  $s=vt/2$ , which is used to operate the HC-SR04 in Arduino.

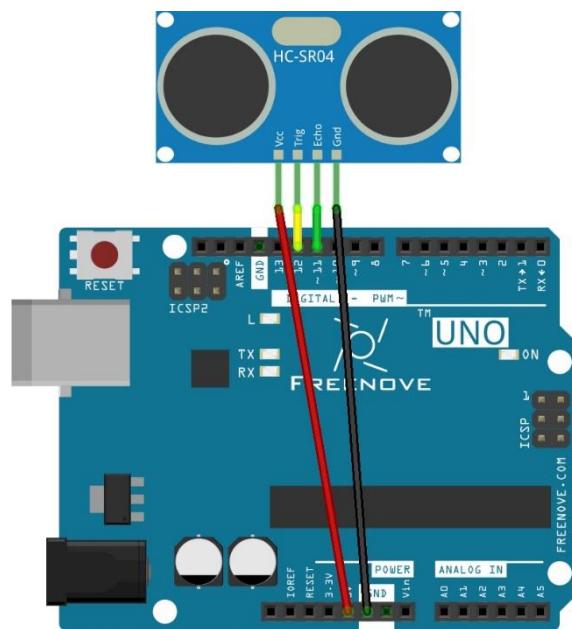
## Circuit

The connection of Freenove UNO board and HC-SR04 is shown below.

Schematic diagram



Hardware connection



## Sketch

### Sketch 24.1.1

First, we use the HC-SR04 communication protocol to operate the module, get the range of time, and calculate the distance.

```
1 #define trigPin 12 // define TrigPin
2 #define echoPin 11 // define EchoPin.
3 #define MAX_DISTANCE 200 // Maximum sensor distance is rated at 400-500cm.
4 // define the timeOut according to the maximum range. timeOut= 2*MAX_DISTANCE /100 /340
5 *1000000 = MAX_DISTANCE*58.8
6 float timeOut = MAX_DISTANCE * 60;
7 int soundVelocity = 340; // define sound speed=340m/s
8
9 void setup() {
10     pinMode(trigPin, OUTPUT); // set trigPin to output mode
11     pinMode(echoPin, INPUT); // set echoPin to input mode
12     Serial.begin(9600); // Open serial monitor at 9600 baud to see ping results.
13 }
14
15 void loop() {
16     delay(100); // Wait 100ms between pings (about 20 pings/sec). 29ms should be the
17     shortest delay between pings.
18     Serial.print("Ping: ");
19     Serial.print(getSonar()); // Send ping, get distance in cm and print result (0 =
20     outside set distance range)
21     Serial.println("cm");
22 }
23
24 float getSonar() {
25     unsigned long pingTime;
26     float distance;
27     digitalWrite(trigPin, HIGH); // make trigPin output high level lasting for 10µs to
28     trigger HC_SR04,
29     delayMicroseconds(10);
30     digitalWrite(trigPin, LOW);
31     pingTime = pulseIn(echoPin, HIGH, timeOut); // Wait HC-SR04 returning to the high level
32     and measure out this waiting time
33     distance = (float)pingTime * soundVelocity / 2 / 10000; // calculate the distance
34     according to the time
35     return distance; // return the distance value
36 }
```

First, define the pins and the maximum measurement distance.

```
1 #define trigPin 12 // define TrigPin  
2 #define echoPin 11 // define EchoPin.  
3 #define MAX_DISTANCE 200 // Maximum sensor distance is rated at 400-500cm.
```

If the module does not return high level, we can not wait forever. So we need to calculate the lasting time over maximum distance, that is, time Out.  $\text{timOut} = 2 * \text{MAX\_DISTANCE} / 100 / 340 * 1000000$ . The constant part behind is approximately equal to 58.8.

```
5 float timeOut = MAX_DISTANCE * 60;
```

Then, in the setup (), set the pin to input or output, and set the serial port. In the loop(). We continue to use serial to print the value of subfunction getSonar (), which is used to return the measured distance of the HC\_SR04. Make trigPin output a high level lasting for at least 10 $\mu$ s, according to the communication protocol.

```
24 digitalWrite(trigPin, HIGH); // make trigPin output high level lasting for 10 $\mu$ s to  
triger HC_SR04,  
25 delayMicroseconds(10);  
26 digitalWrite(trigPin, LOW);
```

Then the echoPin of HC\_SR04 will output a pulse. Time of the pulse is the total time of ultrasonic from transmitting to receiving. We use the pulseIn () function to return the time, and set the timeout.

```
27 pingTime = pulseIn(echoPin, HIGH, timeOut); // Wait HC-SR04 returning to the high level  
and measure out this waitting time
```

Calculate the distance according to the time and return the value.

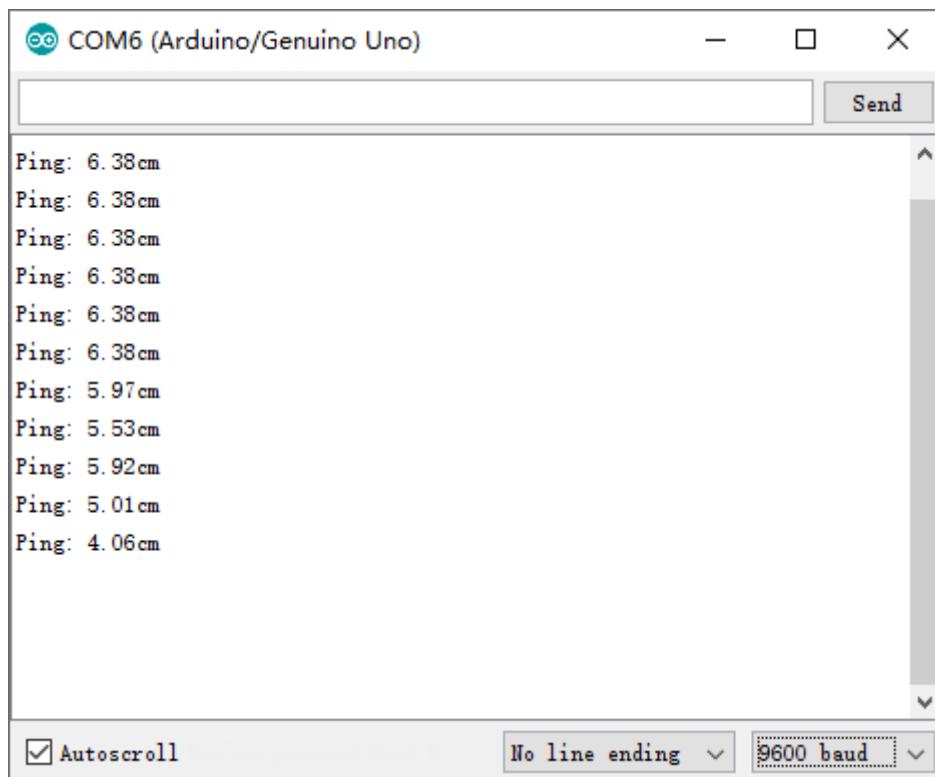
```
28 distance = (float)pingTime * soundVelocity / 2 / 10000; // calculate the distance  
according to the time  
29 return distance; // return the distance value
```

Finally, the code above will be called in the loop ().

```
pulseIn(pin, value) / pulseIn(pin, value, timeout)
```

Return the length of the pulse (in microseconds) or 0 if no pulse is completed before the timeout (unsigned long).

Verify and upload the code to the UNO. Open the serial port monitoring window. Turn the HC-SR04 probe towards the object plane, and observe the data in the serial port monitoring window.



### Sketch 24.1.2

We can also use the library function to directly obtain the measuring distance. We need import the file NewPing.zip before using the library.

```

1 #include <NewPing.h>
2 #define trigPin 12 // define TrigPin
3 #define echoPin 11 // define EchoPin.
4 #define MAX_DISTANCE 200 // Maximum sensor distance is rated at 400-500cm.
5
6 NewPing sonar(trigPin, echoPin, MAX_DISTANCE); // NewPing setup of pins and maximum
distance.
7
8 void setup() {
9     Serial.begin(9600); // Open serial monitor at 9600 baud to see ping results.
10 }
11
12 void loop() {
13     delay(50); // Wait 50ms between pings (about 20 pings/sec). 29ms should be the shortest
delay between pings.
14     Serial.print("Ping: ");
15     Serial.print(sonar.ping_cm()); // Send ping, get distance in cm and print result (0 =
outside set distance range)
16     Serial.println("cm");
17 }
```

First, include the header file of library, and then define the HC SR04 pin and the maximum measurement distance. And, write these parameters when we define the NewPing class objects.

```

1 #include <NewPing.h>
2 #define trigPin 12 // define TrigPin
3 #define echoPin 11 // define EchoPin.
4 #define MAX_DISTANCE 200 // Maximum sensor distance is rated at 400-500cm.
```

And then, in the loop (), use sonar.ping\_cm () to obtain the the ultrasonic module detection distance with unit of centimeter. And print the distance out. When the distance exceeds range of 2cm~200cm, the printed data is zero.

#### NewPing Class

NewPing class can be used for SR04, SRF05, SRF06 and other sensors. An object that needs to be instantiated when the class is used. The three parameters of the constructor function are: trigger pin, echo pin and maximum measurement distance.

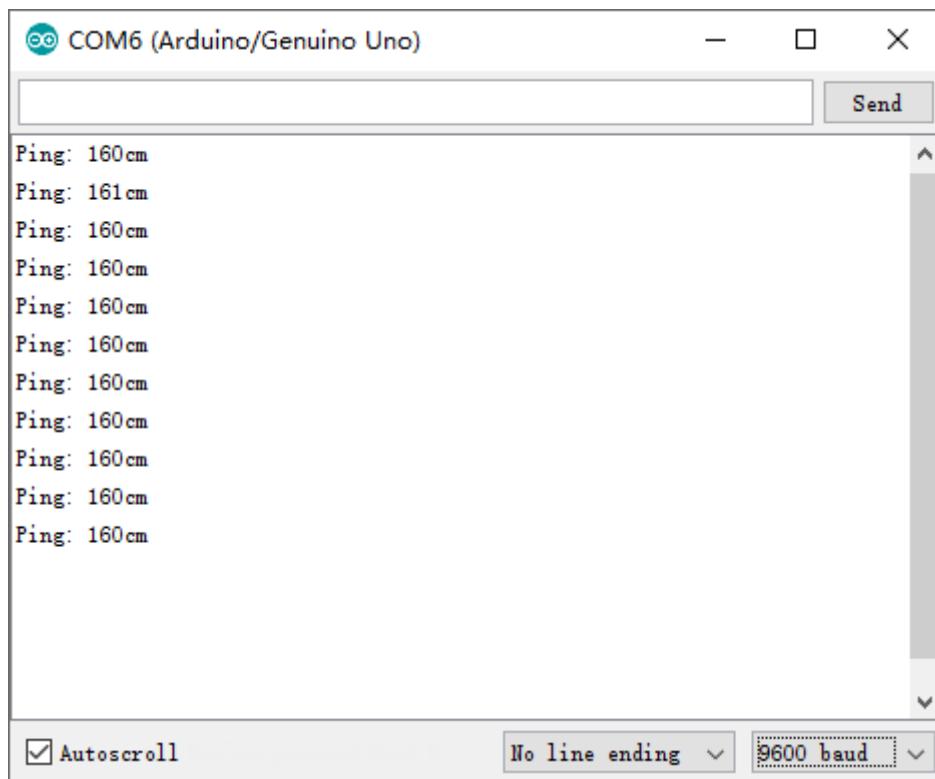
`sonar.ping()` - Send a ping and get the echo time (in microseconds) as a result.

`sonar.ping_in()` - Send a ping and get the distance in whole inches.

`sonar.ping_cm()` - Send a ping and get the distance in whole centimeters.

For more details, please refer to the NewPing.h in the NewPing library.

Verify and uploaded the code to UNO. Open the serial port monitoring window. When you make ultrasonic probe toward a plane of an object, you can observe the distance changes.



# Chapter 25 Solder Circuit Board

From previous chapters, we have learned the knowledge of electronic circuit and component, and build a variety of circuits. Now, we will take a further step, making a piece of circuit board on your own.

We will use the general board to solder the circuit and components. And when this chapter is over, it's our hope to help you master the idea of how to design your own circuit, build and print circuit boards.

To finish this chapter, you need to prepare the necessary soldering equipments, including electric iron and solder. We have prepared the general board for you, please pay attention to safety when you operate these experiments.

## Project 25.1 Solder a Buzzer

We have tried to use the buzzer from previous chapter, and now we will solder a circuit that when the button is pressed, the buzzer sounds. This circuit doesn't need Arduino, and cost no electricity power when the button is not pressed.

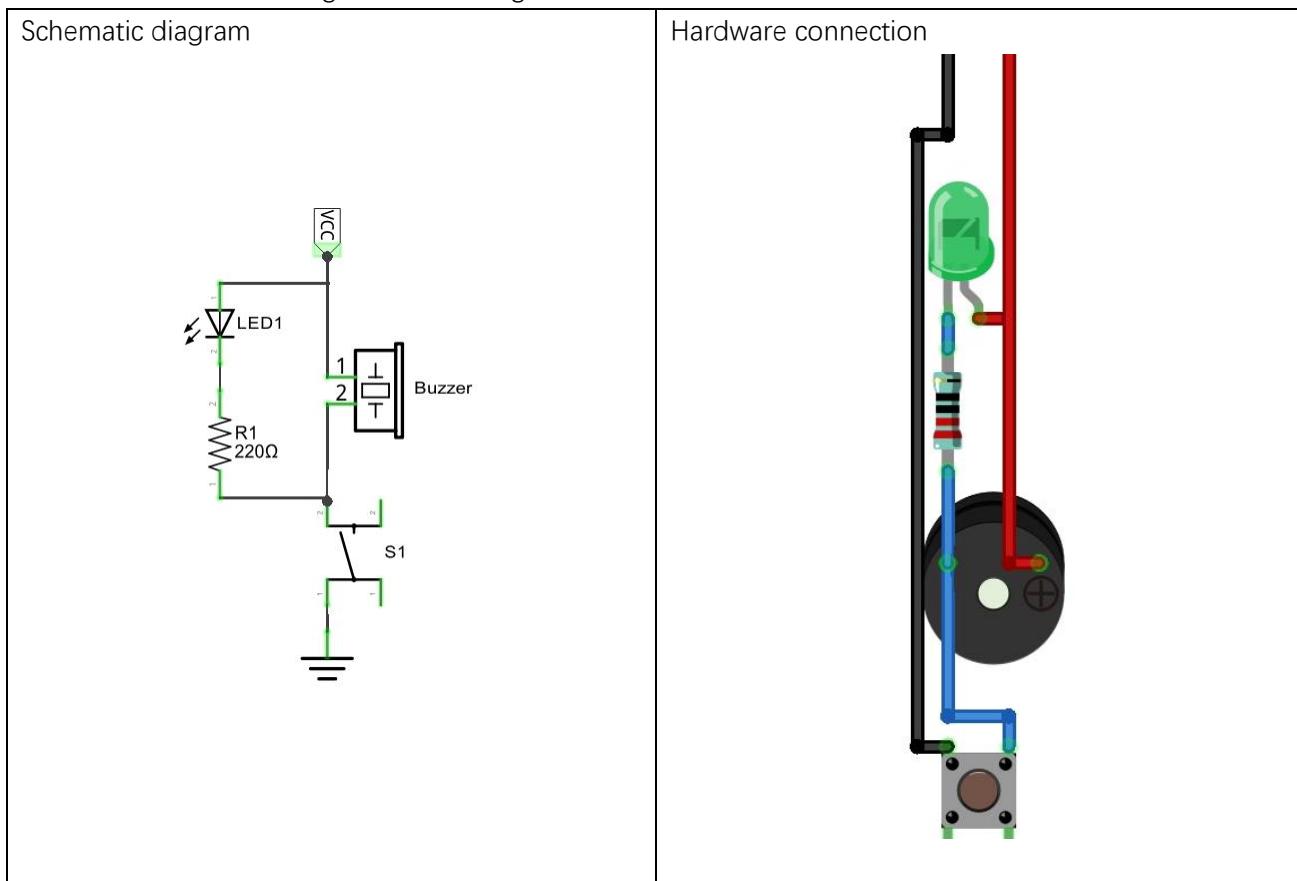
You can install it on your bike, or any other place you need.

## Component list

Pin header x2	LED x1	Resistor 220Ω x1	Active buzzer x1	Push button x1
				

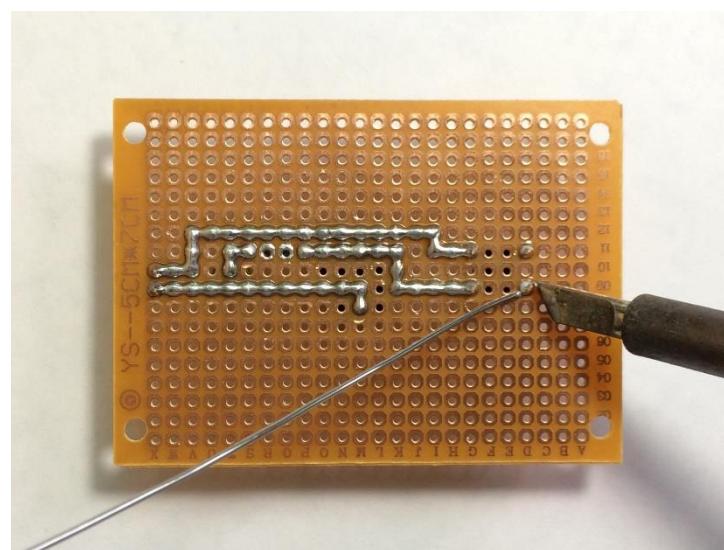
## Circuit

We will solder the following circuit on the general board.

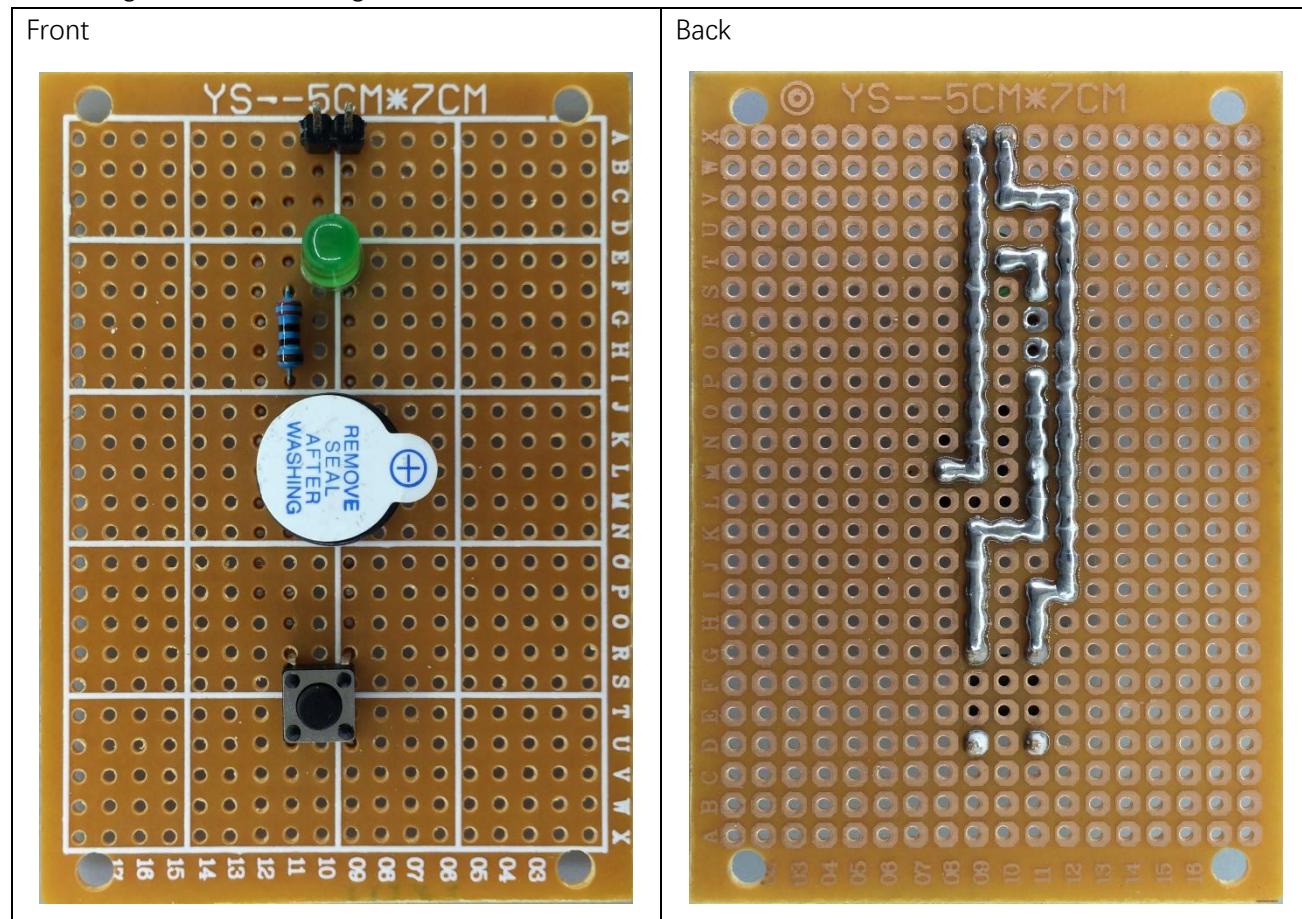


## Solder the circuit

Insert the components in the general board and solder the circuit on the back.

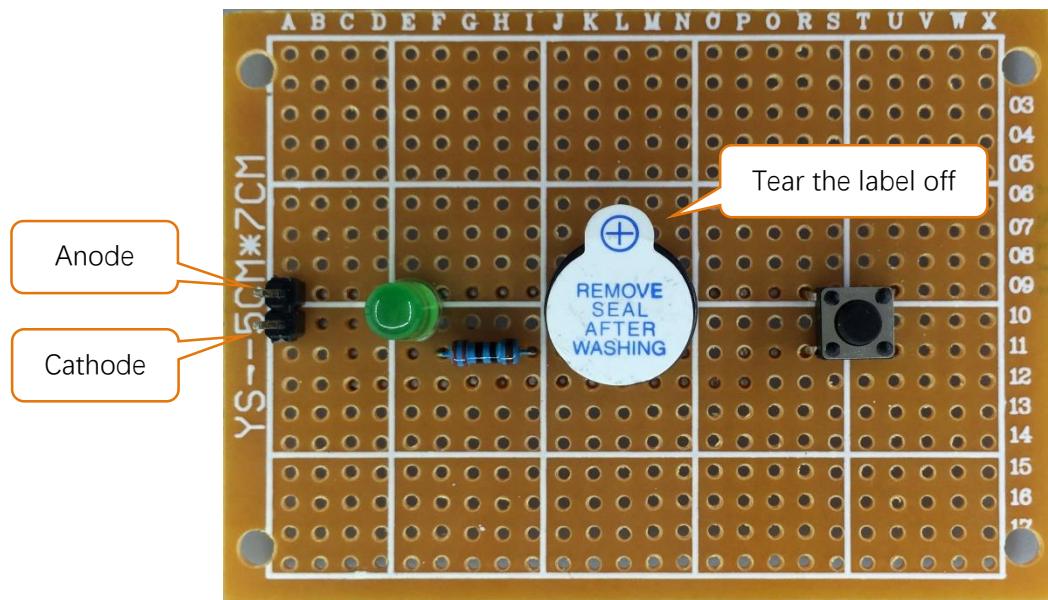


Effect diagram after soldering:



## Test circuit

Connect the circuit board to power supply (3~5V). You can use UNO board or battery box as the power supply.



Press the push button after connecting the power, and then the buzzer will make a sound.

## Project 25.2 Solder a Flowing Water Light

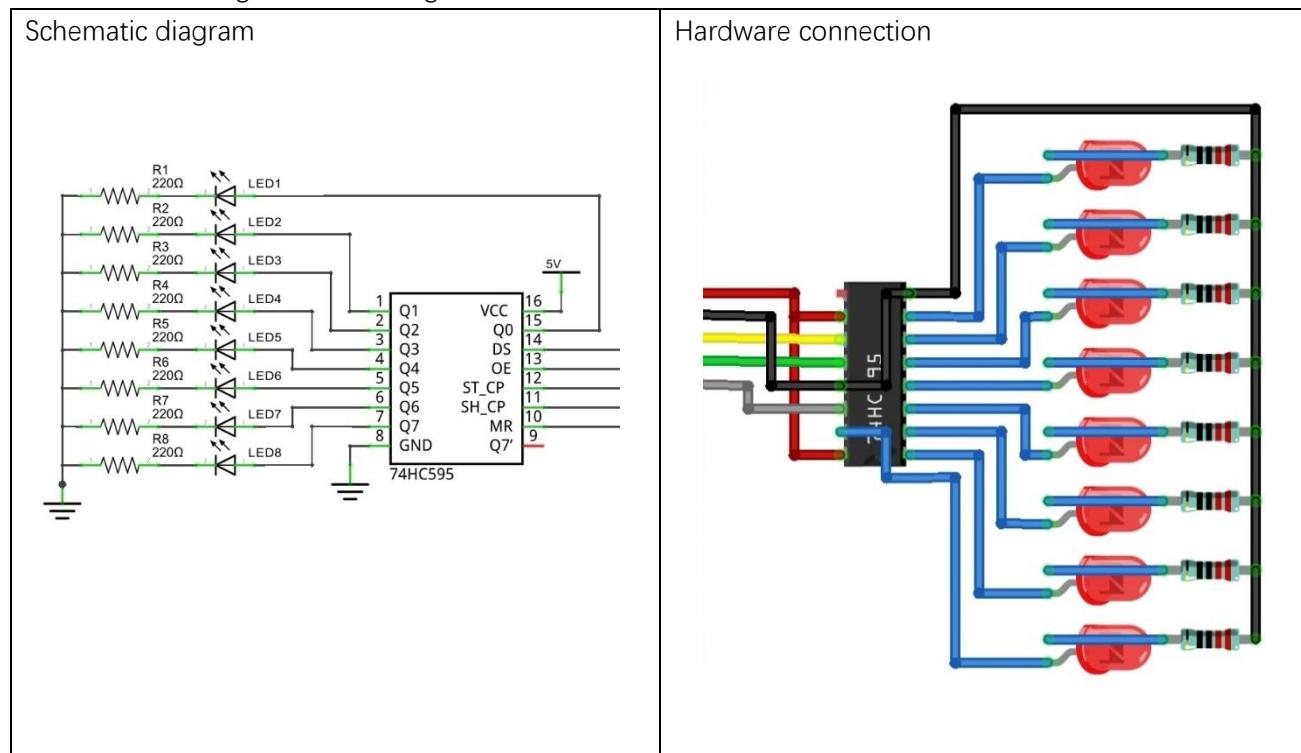
From previous chapter, we have learned to make a flowing water light with LED. Now, we will solder a circuit board, and use the improved code to make a more interesting flowing water light.

### Component list

Pin header x5	Resistor 220Ω x8	LED x8	74HC595 x1

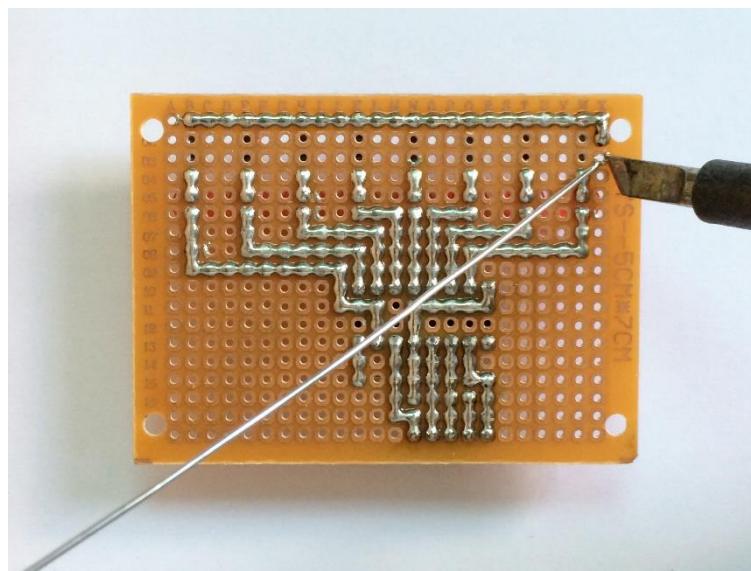
### Circuit

Solder the following circuit on the general board.

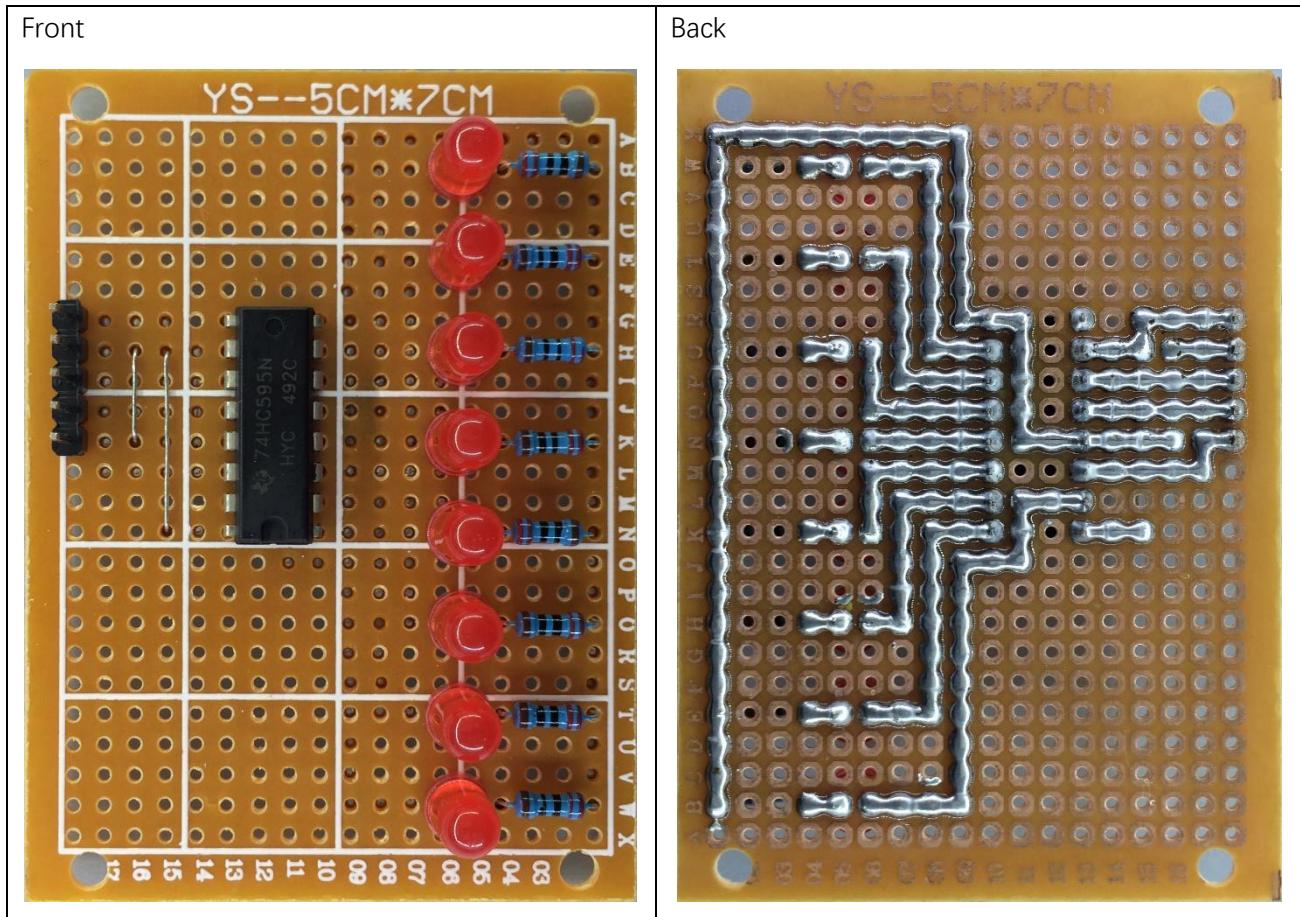


## Solder the circuit

Insert the components in the general board, and solder the circuit on the back.

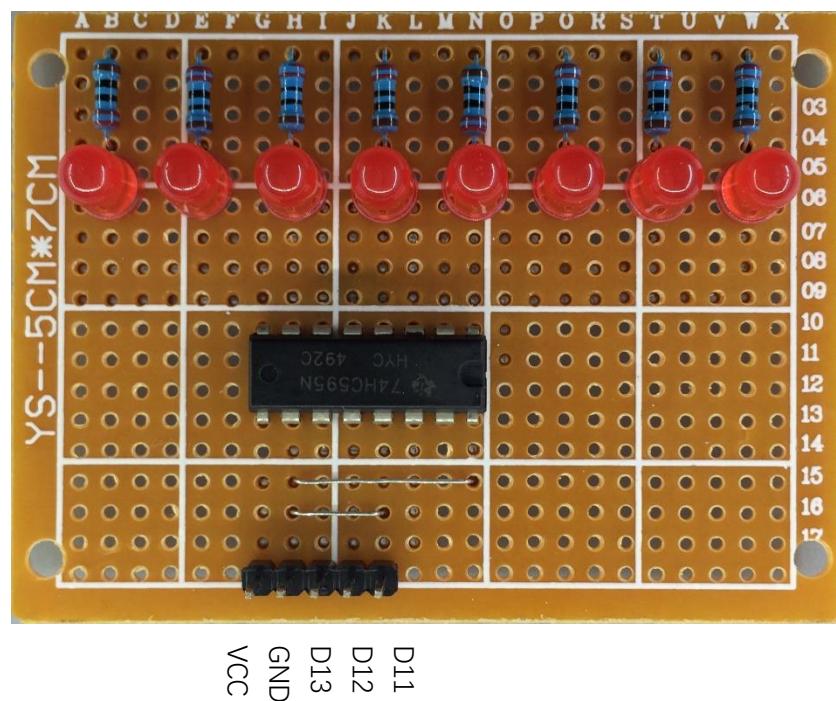


Effect diagram after soldering:



## Connect the circuit

Connenct the board to UNO with jumper wire in the following way.



## Sketch

### Sketch 25.2.1

Now, let's write code to make the dropping-rain effect happen on our board.

```
1 int latchPin = 12;           // Pin connected to ST_CP of 74HC595 (Pin12)
2 int clockPin = 13;           // Pin connected to SH_CP of 74HC595 (Pin11)
3 int dataPin = 11;            // Pin connected to DS of 74HC595 (Pin14)
4
5 void setup() {
6     // set pins to output
7     pinMode(latchPin, OUTPUT);
8     pinMode(clockPin, OUTPUT);
9     pinMode(dataPin, OUTPUT);
10 }
11
12 void loop() {
13     // Define an array to save the pulse width of LED. Output the signal to the 8 adjacent
14     // LEDs in order, and then it produces the dropping-rain effect
```

```

14   const byte pulse[] = {0, 0, 0, 0, 0, 0, 0, 0, 64, 48 ,32, 16, 8, 4, 2, 1, 0, 0, 0, 0,
15   // Define a variable to select 8 contiguous data in the array sequentially
16   static byte offset;
17   // Define a variable to control the speed
18   static unsigned int counter;
19   if (counter++ % 8 == 0)           // Reduce the self-increasing speed of offset
20       offset < 15 ? offset++ : offset = 0;// offset increases
21   // Out put PWM wave
22   for (int i = 0; i < 64; i++) { // The cycle of PWM is 64 cycles
23       byte data = 0;           // Define a variable to represent the output state of
this loop
24       for (int j = 0; j < 8; j++) // Calculate the output state of this loop
25   {
26       if (i < pulse[j + offset]) // Calculate the LED state according to the pulse width
27   {
28           data |= 0x01 << j;    // Represent the LED state with the corresponding bit
of a variable
29       }
30   }
31   // Send the state of LED to 74HC595
32   writeData(data);
33   }
34   }
35
36 void writeData(int value) {
37     // Make latchPin output low level
38     digitalWrite(latchPin, LOW);
39     // Send serial data to 74HC595
40     shiftOut(dataPin, clockPin, MSBFIRST, value);
41     // Make latchPin output high level, then 74HC595 will update the data to parallel
output
42     digitalWrite(latchPin, HIGH);
43 }
```

First, we define an array with 24 numbers that each number represents the brightness of LED, that is, the pulse width of PWM. 8 LEDs output the brightness of 8 adjacent numbers each time.

14	const byte pulse[] = {0, 0, 0, 0, 0, 0, 0, 0, 64, 48 ,32, 16, 8, 4, 2, 1, 0, 0, 0,
	0, 0, 0, 0};

Define a variable to select 8 adjacent numbers in an array in turn.

16	static byte offset;
----	---------------------

Define a variable to control the speed of the raindrops.

18	static unsigned int counter;
----	------------------------------

Reduce the auto-increment speed of the variable offset by the following code.

```
19   if (counter++ % 8 == 0)           // Reduce the self-increasing speed of offset  
20     offset < 15 ? offset++ : offset = 0;// offset increases
```

We use software to output PWM waveform. Define the cycle of PWM to be 64 cycles and determine the pulse width of LED (that is brightness) according to the selected eight numbers of the array in each cycle.

```
22   for (int i = 0; i < 64; i++) { // The cycle of PWM is 64 cycles  
23     byte data = 0;             // Define a variable to represent the output state of  
this loop.  
24     for (int j = 0; j < 8; j++) // Calculate the output state of this loop  
25     {  
26       if (i < pulse[j + offset]) // Calculate the LED state according to the pulse width  
27       {  
28         data |= 0x01 << j;    // Represent the LED state with the corresponding bit  
of variable  
29       }  
30     }
```

Due to the change of the variable offset, the LED will output the brightness that the eight adjacent numbers represents in the array, and form the dropping-rain effect.

Verify and upload the code, then you will see the dropping-rain effect that LED forms.

# Other Components

This kit also includes other common components that can help you realize your ideas. Now let's introduce components not mentioned in the previous section.

## Component knowledge

### Toggle switch

Like push button, toggle switch is also a kind of switching devices. The difference is that toggle switch is suitable for the long-time open or close circuit.



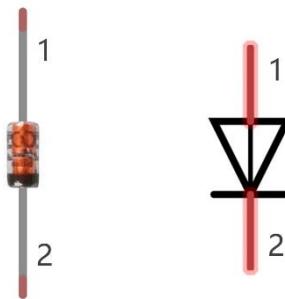
When move the lever to the left, pin 1, 2 gets conducted, and pin 2, 3 disconnected;

When move the lever is to the right, pin 2,3 gets conducted, and pin 1,2 disconnected;

### Switch diode

Diodes has several types. We have used 1N4001 before which is a common rectifier diode and commonly used in ac rectifier.

Here is 1N4148, which is a kind of high-speed switching diodes and is characterized by a relatively rapid switching.



For the switching diode, the changing time from conduction to cut off or from cut off to conduction is shorter than general diode and it is mainly used in electronic computer, pulse and switching circuits.

### 9V battery cable

A 9V battery cable can connect a 9 V battery, which can supply power for UNO board.

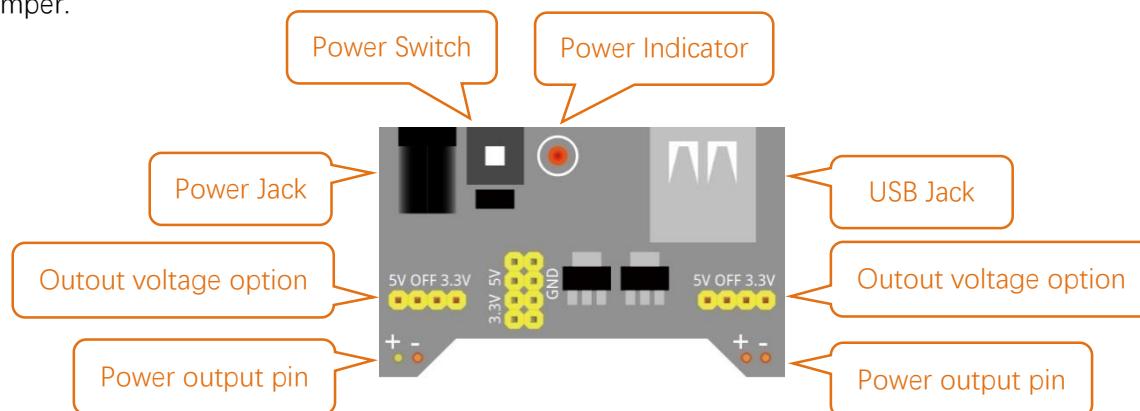


The installation of 9V battery cable is as follows:

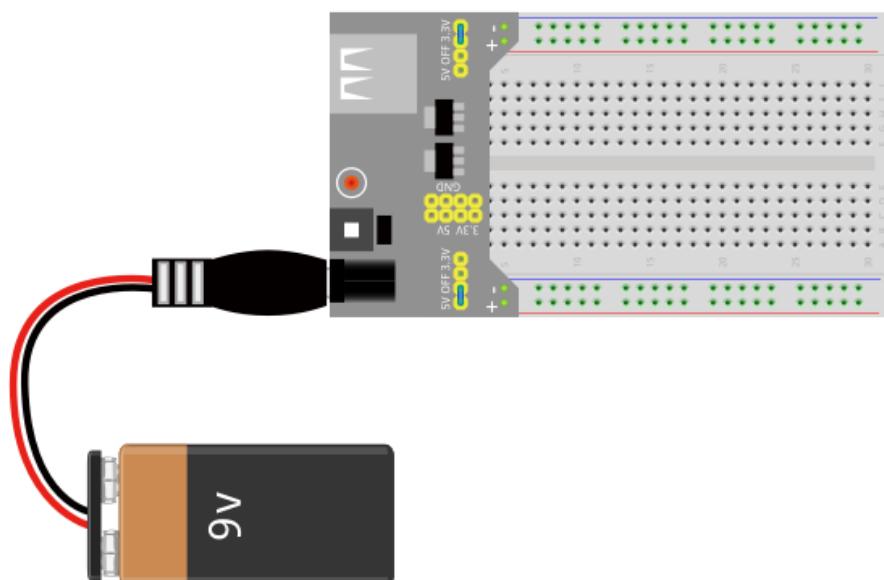


#### Power supply module for breadboard

The following is the power supply module for breadboard, this module can provide the breadboard with two-channel power supply separately, each power supply can be configured to 3.3 V or 5 V separately through the jumper.



We can build the circuit conveniently by using this module. You only need to provide power supply for this module, and then insert it on the the breadboard.





## What's next?

Thanks for your reading.

This book is all over here. If you find any mistakes, missions or you have other ideas and questions about contents of this book or the kit and ect, please feel free to contact us, and we will check and correct it as soon as possible.

If you want to learn more about Arduino, Raspberry Pi, smart cars, robots and orther interesting products in science and technology, please continue to focus on our website. We will continue to launch cost-effective, innovative and exciting products.

Thank you again for choosing Freenove products.

# Appendix

## ASCII table

<b>Dec</b>	<b>Hex</b>	<b>Char</b>	<b>Dec</b>	<b>Hex</b>	<b>Char</b>	<b>Dec</b>	<b>Hex</b>	<b>Char</b>	<b>Dec</b>	<b>Hex</b>	<b>Char</b>
0	00	Null	32	20	Space	64	40	Ø	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	\$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	Ø	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	:	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

## Resistor color code

The diagram illustrates three methods for resistor color coding:

- 4-Band-Code:** Shows a resistor with four bands. The first three bands (Red, Green, Blue) represent the resistance value (220 ohms). The fourth band (Gold) represents the tolerance (± 5%).
- 5-Band-Code:** Shows a resistor with five bands. The first four bands (Red, Green, Blue, Gold) represent the resistance value (220 ohms). The fifth band (Silver) represents the tolerance (± 1%).
- 5-Band-Code with Tolerance:** Shows a resistor with five bands. The first four bands (Red, Green, Blue, Gold) represent the resistance value (220 ohms). The fifth band (Silver) represents the tolerance (± 1%).

**Color Chart:**

COLOR	1 <sup>ST</sup> BAND	2 <sup>ND</sup> BAND	3 <sup>RD</sup> BAND	MULTIPLIER	TOLERANCE
Black	0	0	0	1Ω	
Brown	1	1	1	10Ω	± 1% (F)
Red	2	2	2	100Ω	± 2% (G)
Orange	3	3	3	1KΩ	
Yellow	4	4	4	10KΩ	
Green	5	5	5	100KΩ	± 0.5% (D)
Blue	6	6	6	1MΩ	± 0.25% (C)
Violet	7	7	7	10MΩ	± 0.10% (B)
Grey	8	8	8		± 0.05%
White	9	9	9		
Gold				0.1Ω	± 5% (J)
Silver				0.01Ω	± 10% (K)