Free your innovation

Freenove is an open-source electronics platform.
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

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 Modular Starter Kit for Arduino, which includes the following folders and files:

- Datasheet Datasheet of electronic components and modules
- Libraries Library files of Arduino Software
- Sketches Code of Arduino Software
- Readme.txt Instructions

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

Copyright

Contents

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.

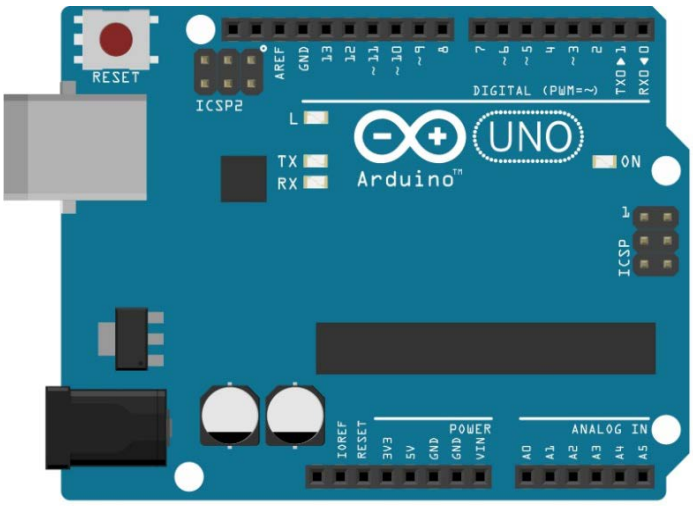
If you have certain understanding of C/C++ language, and are very interested in electronic modules, this book is very suitable for you. Because this book will gradually introduce programming knowledge of Arduino, the using method of various sensor modules and the principle these modules works on. And we arrange some practical application for these module. And the supporting kit, Freenove Modular Starter Kit for Arduino, contains all electronic components and modules needed to complete all projects in this book. After completing all projects, you can also use these modules to achieve your own creativity such as: smart home, smart car or robot and so on. Additionally, you can also ask us for fast and free technical support at any time about this book and its supporting kit. It is especially suitable for beginners.

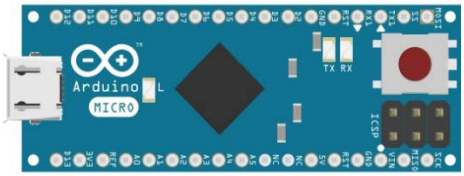
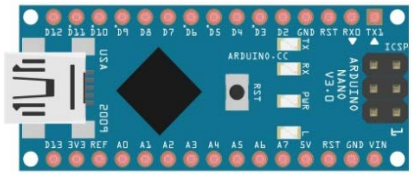
Arduino Board

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.

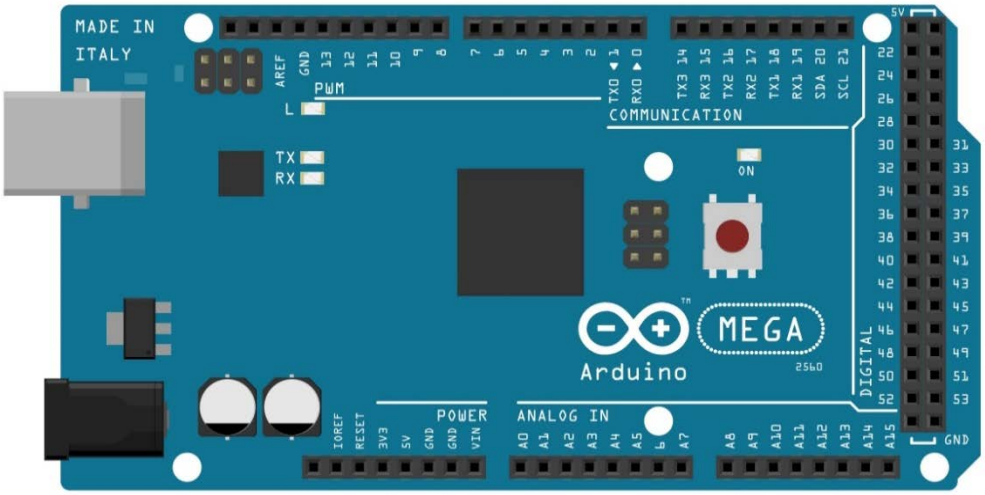
Currently, Arduino Board has several models, and the code between boards of different types is universal (some boards may not be completely compatible because of the differences in hardware). Popular boards include:

UNO

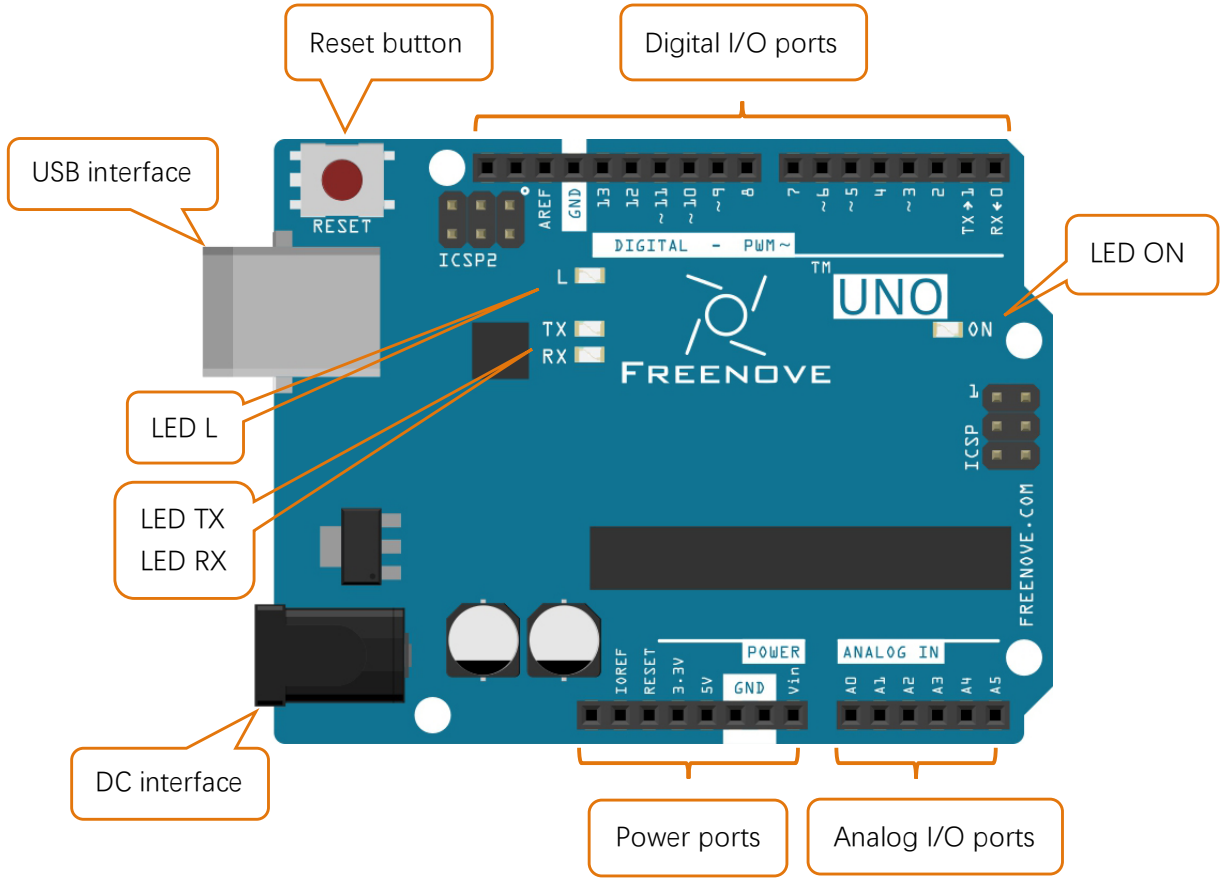


MICRO	NANO
	

MEGA



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:



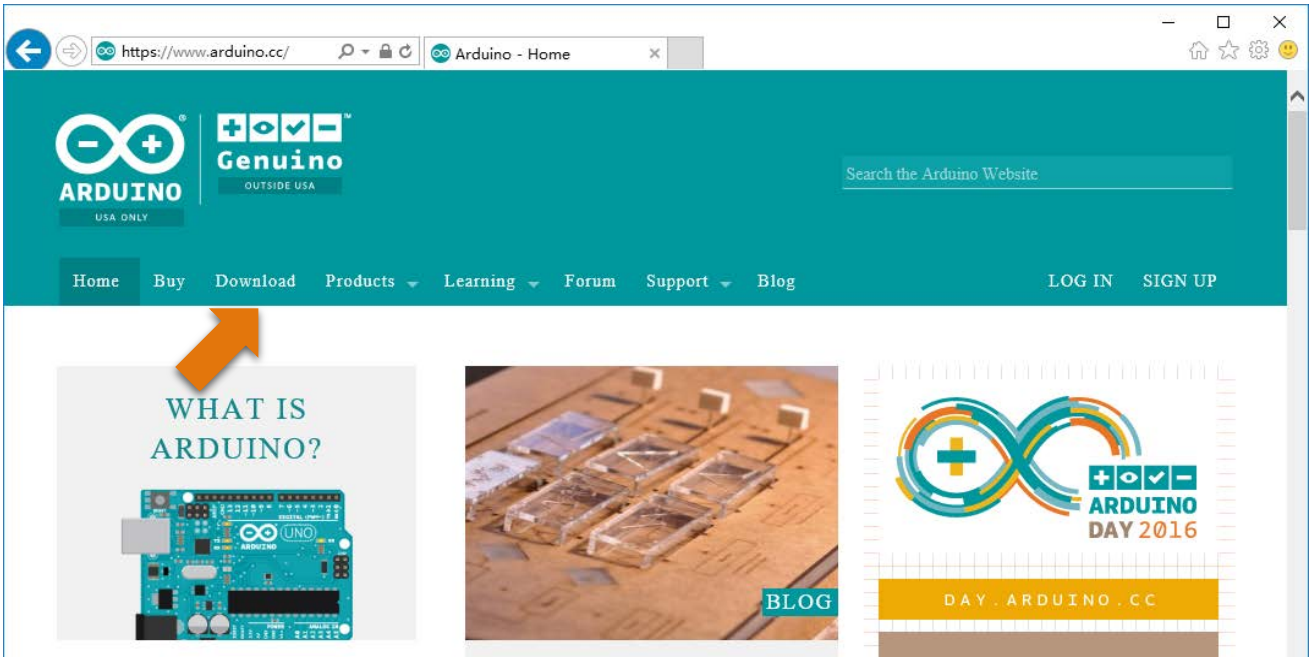
- Digital I/O ports is used to connect to other components or modules, to receive an input signal, or to send a control signal. Usually, we name it by adding a "D" in front of the number, such as D13.
- USB interface is used to provide power, upload code or communicate with PC.
- LED L is connected to 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).

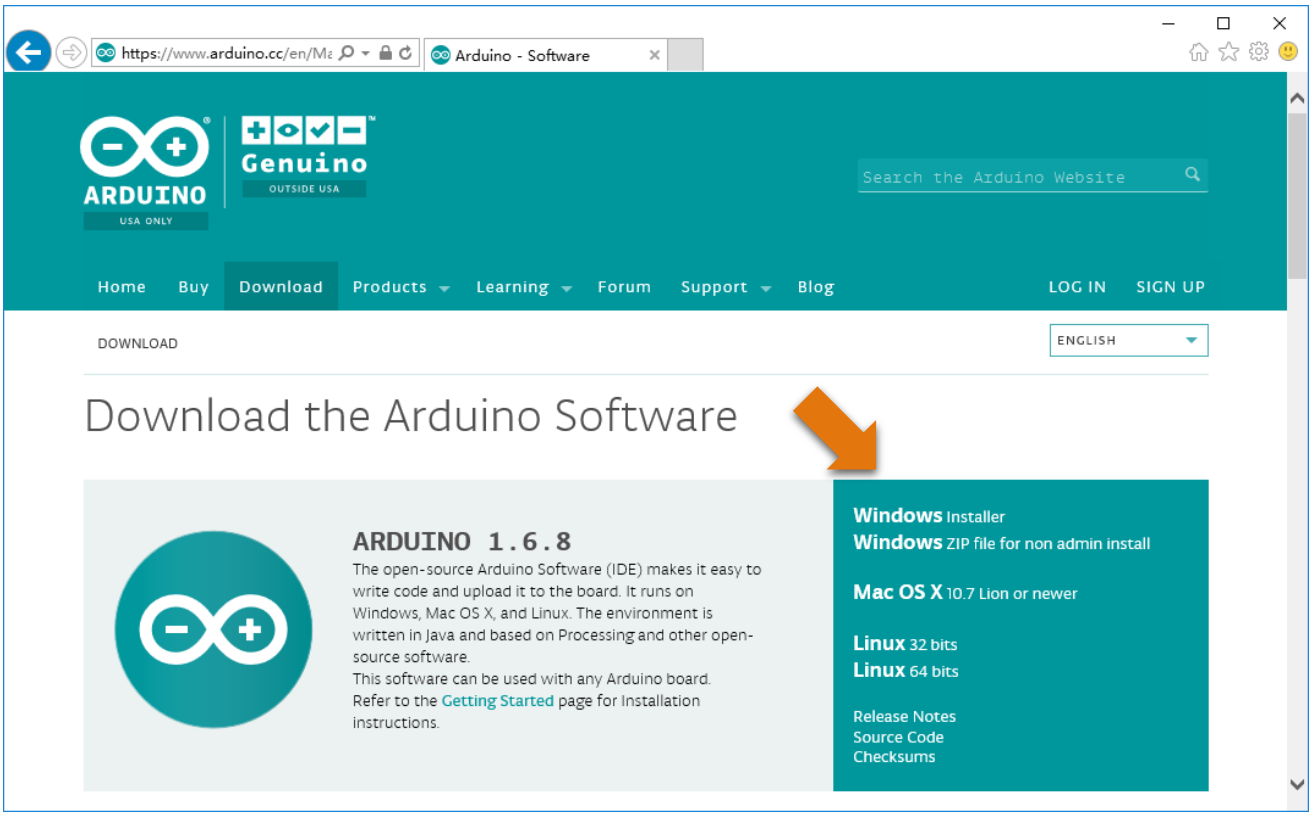
Arduino Software

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.

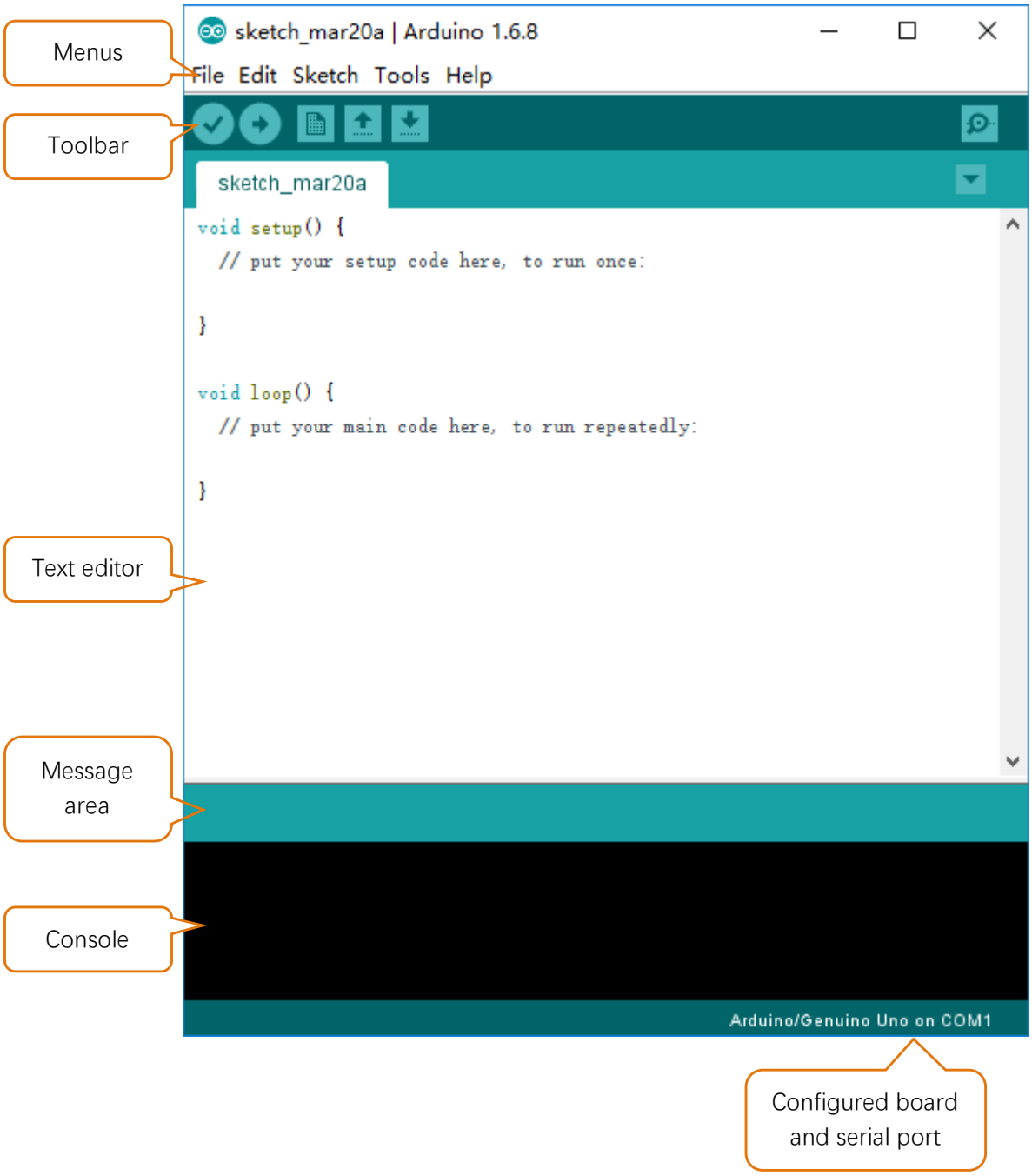


Download corresponding installation program according to your operating system. If you are a Windows user, please select the "Windows Installer" to download to install the driver correctly.



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.

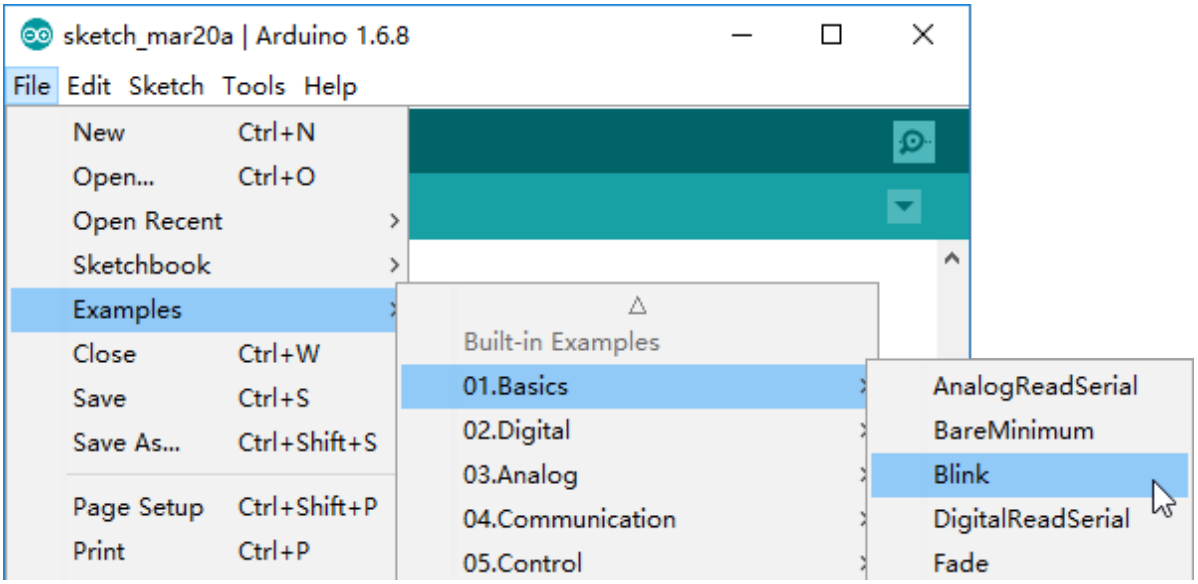
Save
Saves your sketch.

Serial Monitor
Opens the serial monitor.

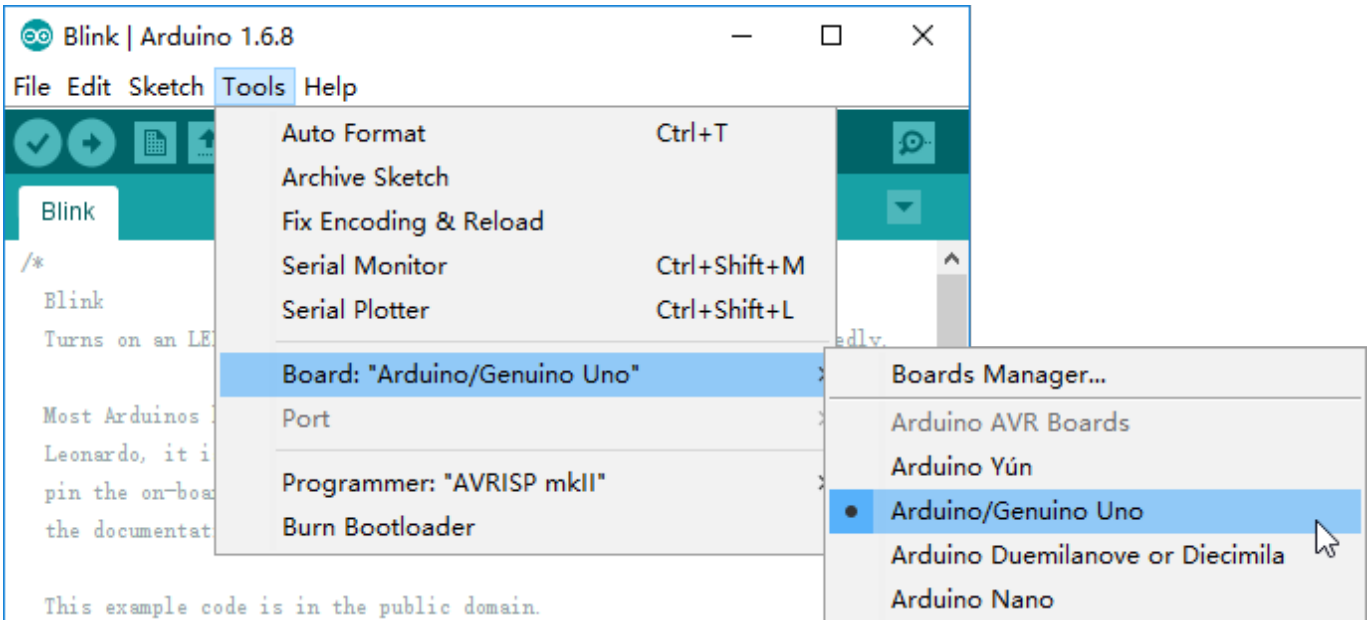
Additional commands are found within the five menus: File, Edit, Sketch, Tools, Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

First Use

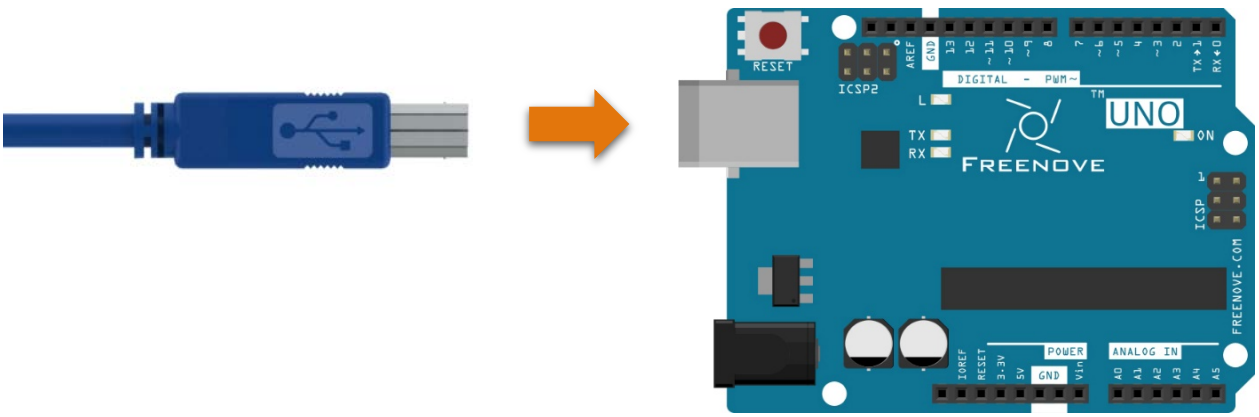
Open the exmple sketch "Blink" with Arduino Software.



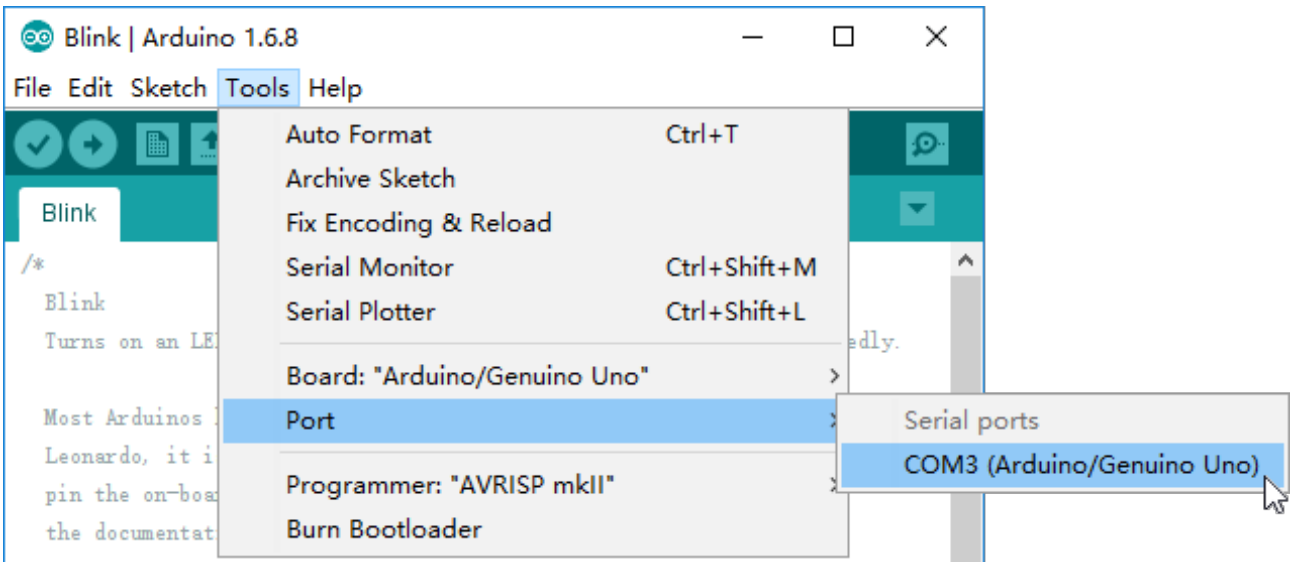
Select board "Arduino/Genuino Uno".



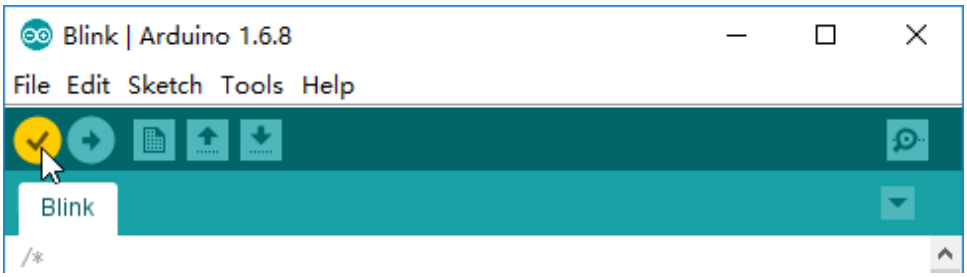
Connect Freenove UNO to computer with USB cable.



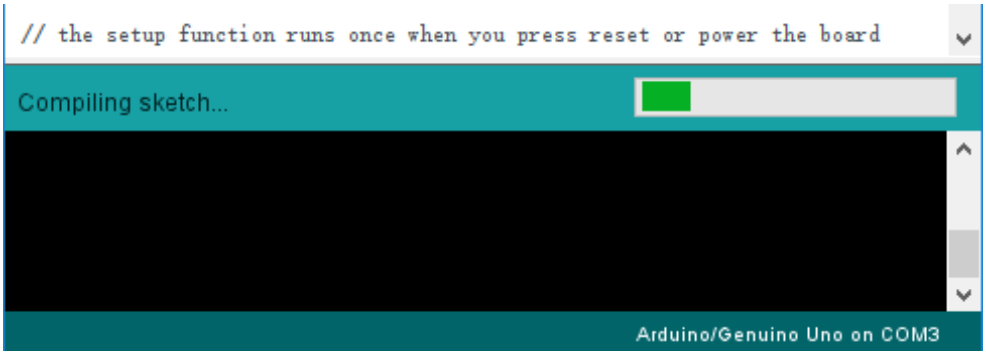
Select the serial port. Your serial number may be different from the following figure. If it is not detected immediately, please wait for a while, then click "Tools" to check again.



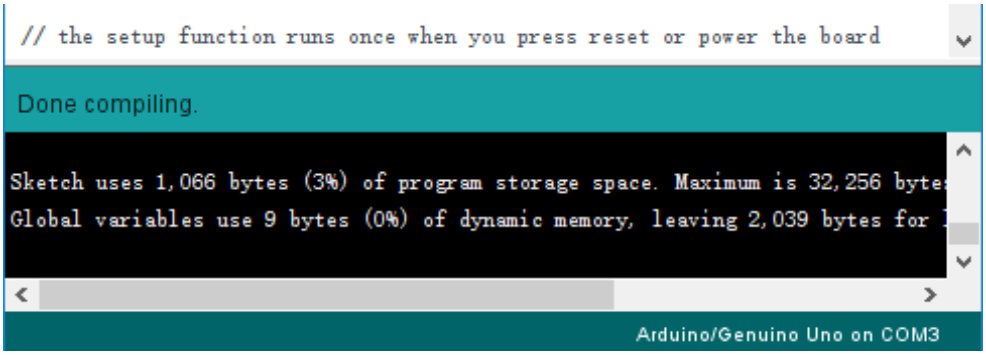
Click "Verify" button.



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.

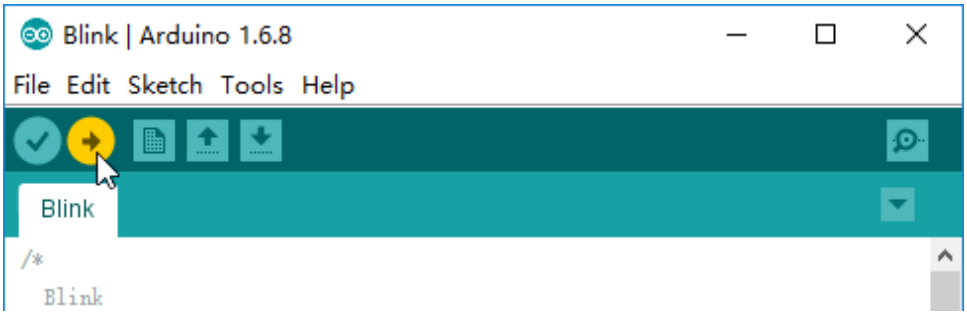
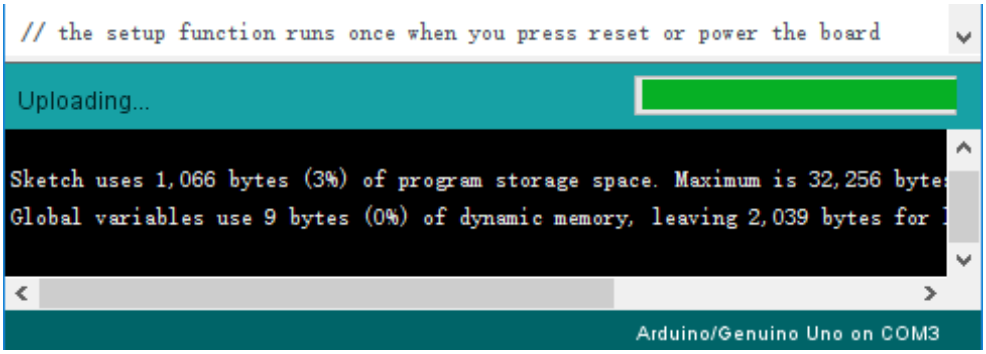
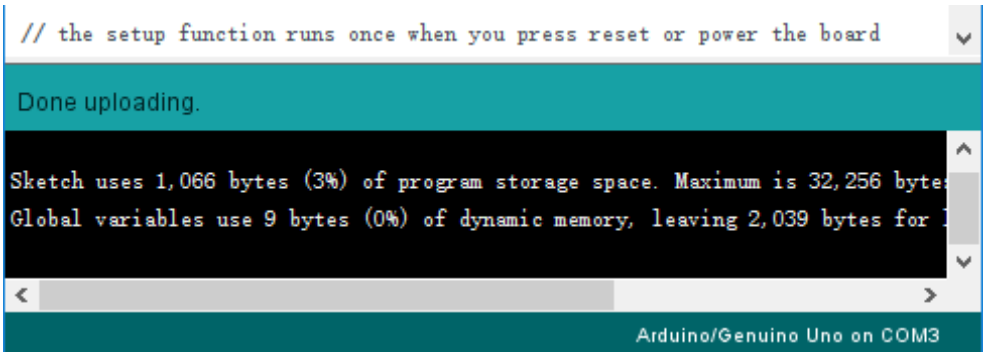


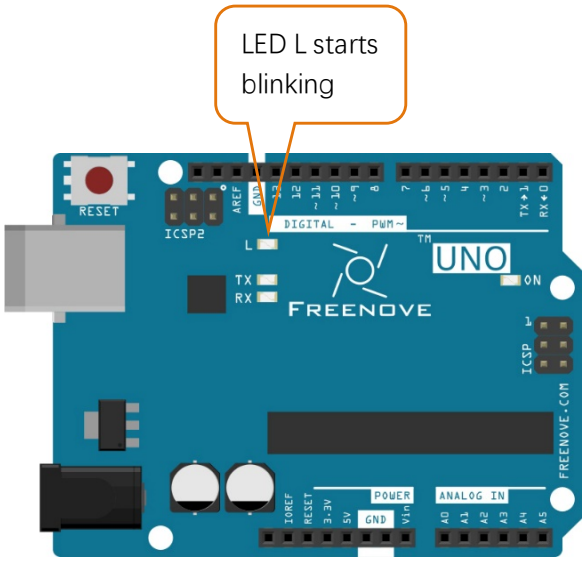
Figure below shows code are uploading.



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!



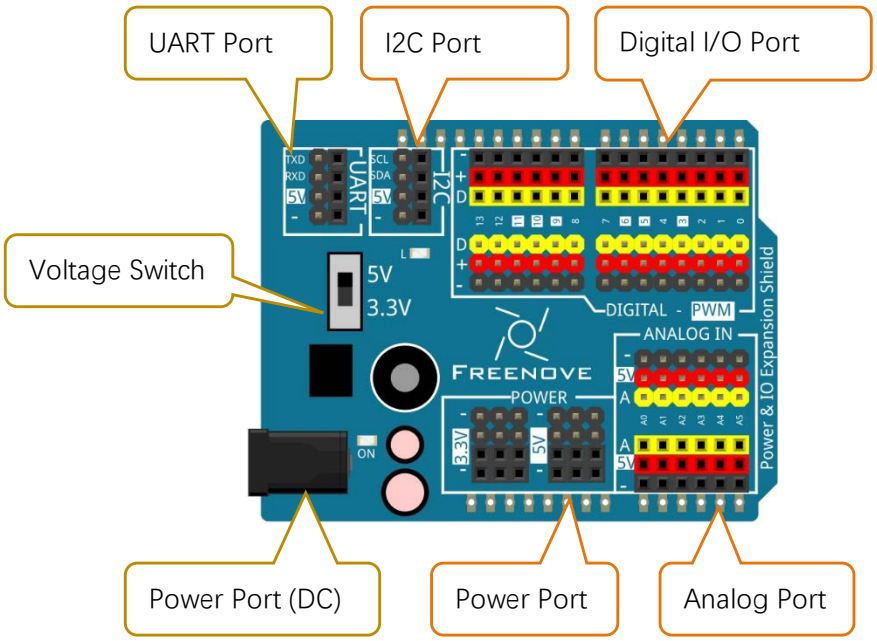
So far, we have completed the first use. I believe you have felt the joy of it. Next, we will carry out a series of projects, from easy to difficult, taking you to learn the Arduino programming and the building of electronic circuit.

Freenove Power & I/O Expansion Shield

Power & I / O expansion shield is appli cable to the Arduino uno board including the compatible board The output power of Power supply is no more than 5V*3A. Silk screen printing notes are as follows:

Mark	Describtion
“+ ”	Digital port VCC of expansion board. select 5V or 3.3V through transferring the switch. There are two situations: 1. The DC port on Expansion Shield is directly connected to power supply: The power of these pins comes from the expansion shield. 2. The DC port is not directly connected to power supply: the power of these pins comes from UNO.
“- ”	It is connected to GND of UNO
“5V ”	+5V voltage supplied by power UNO
“3.3V ”	+3.3V voltage supplied by power UNO

Additionaly, the shield expends the I2C and UART interface. Each set of these interfaces composed of a male type（pin） and female type（slot）, which greatly facilitates the connection between the shield and the peripheral modules. The diagram of the shield is shown below:



Later, we will use this shield to do some experiment.

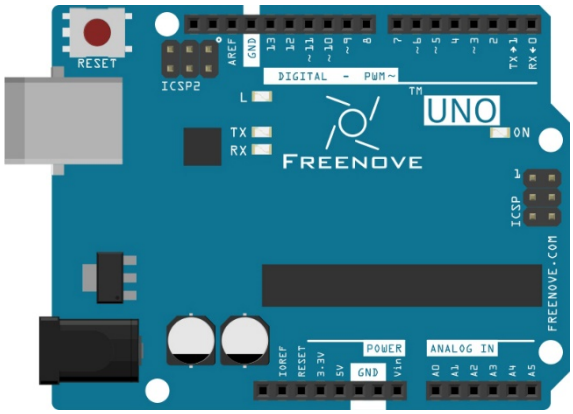
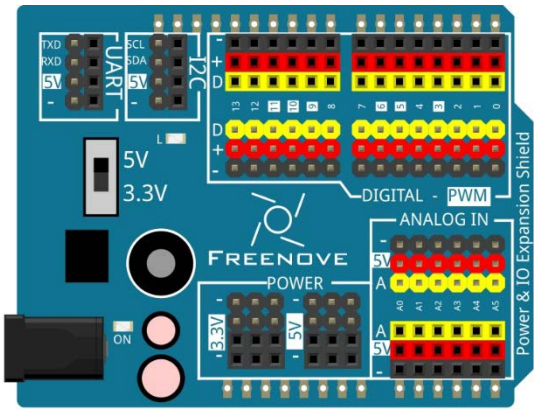
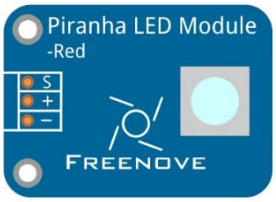
Chapter 1 LED Blink

We have made LED on the UNO board blink. Now, we will make the external LED module flash, and understand the basic components of the electronic system and the basic operation principle of the Arduino program.

Project 1.1 Piranha LED Module Blink

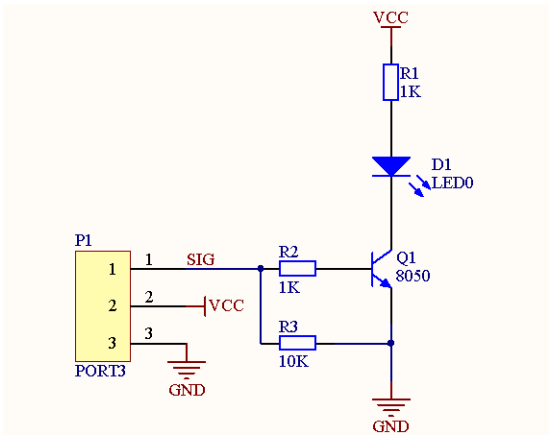
Repeat the previous experiment to make the external LED module blink.

Component List

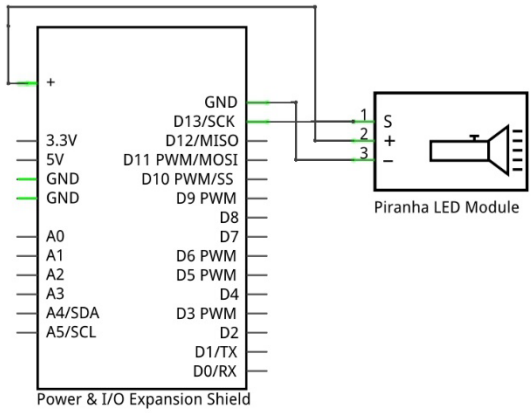
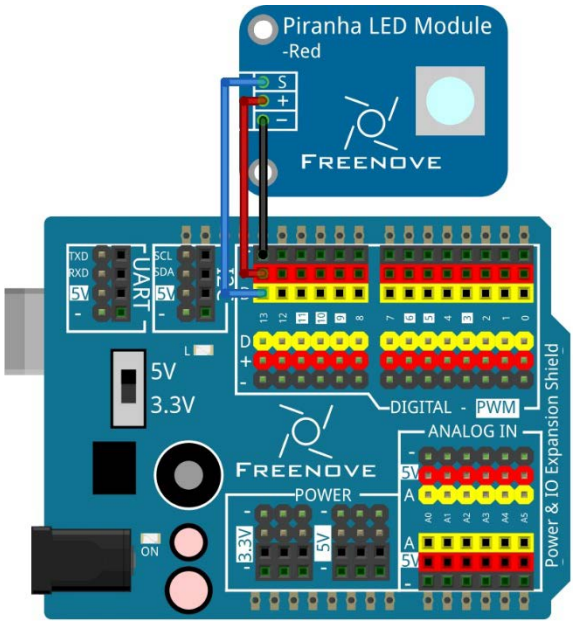
Freenove UNO x1 	Freenove Power & I/O Expansion Shieldx1 
USB cable x1 	Piranha LED Module x1 
Jumper M/F x3 	

Component Knowledge

The circuit diagram of LED Module Piranha is shown below. It is helpful for us to analyze the hardware.

Circuit diagram：	Analyze：
	According to the given circuit diagram, it can be known: when the signal pin SIG is high, the transistor Q1 gets connected, and LED gets on. On the contrary, when the signal pin SIG is low, the LED goes out.

Circuit

Schematic diagram：	Hardware connection：
	

Sketch

Sketch 1.1.1 Blink

Use the previous sketch "Blink" again.

```
1  int ledPin=13; //Connect LED to Pin13
2  void setup() {
3    // initialize digital ledPin as an output.
```

```
4     pinMode(ledPin, OUTPUT);
5   }
6
7   // the loop function runs over and over again forever
8   void loop() {
9     digitalWrite(ledPin, HIGH);   // turn the LED on (HIGH is the voltage level)
10    delay(1000);                  // wait for a second
11    digitalWrite(ledPin, LOW);    // turn the LED off by making the voltage LOW
12    delay(1000);                  // wait for a second
13  }
```

Verify and upload the code to UNO, then the Piranha LED Module starts flashing.

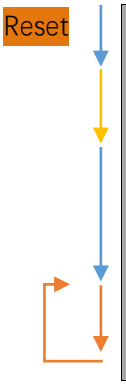
Analysing the Code

The Arduino codes usually contain two basic functions: setup void () and loop void ().
After Arduino board is reset, the setup () function will be executed firstly, then the loop () function will be executed.

Setup () is generally used to write codes to initialize the hardware. And Loop () function is used to write codes to achieve certain functions. Loop () function will be executed repetitively. When the execution reaches to 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 codes and pressing the reset button will make the control panel be 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  }
```

First, define a variable ledPin.

```
int ledPin=13; //Connect LED to Pin13
```

Set ledPin to output mode in setup() function.

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

Finally, set the ledPin output high level to turn LED on and last for one second. Then, set the ledPin output low level to turn LED off for another one second. Repeat this operating constantly.

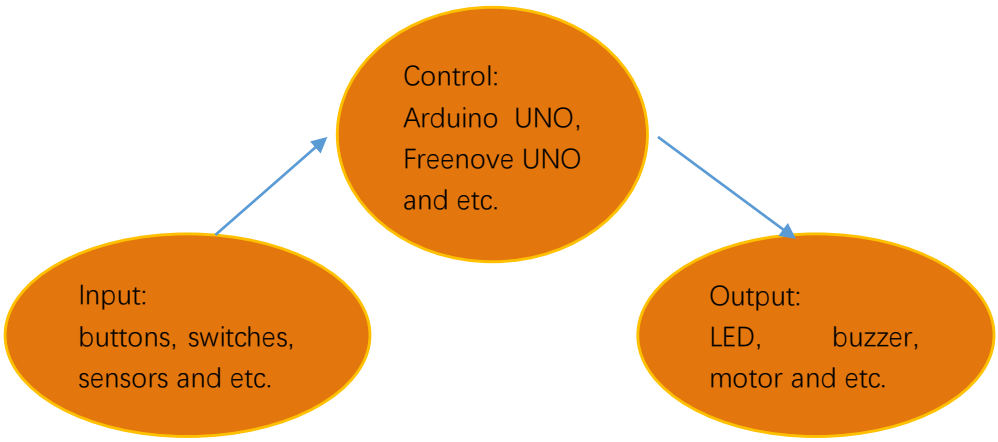
```
  digitalWrite(ledPin, HIGH);   // Turn the LED on (HIGH is the voltage level)
  delay(1000);                  // Wait for a second
  digitalWrite(ledPin, LOW);    // Turn the LED off by making the voltage LOW
  delay(1000);                  // Wait for a second
```

pinMode(pin, mode)
Configures the specified pin to behave either as an input or an output.
digitalWrite (pin, value)
Write a HIGH or a LOW value to a digital pin.

For more detail of Arduino function, variable type, key words and other information, please visit Arduino official website.
http://www.arduino.cc/en/Reference/HomePage

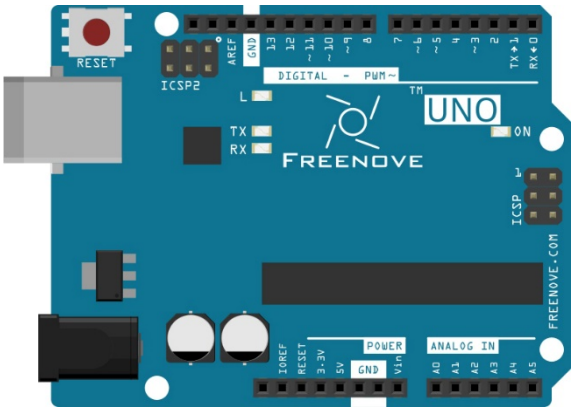
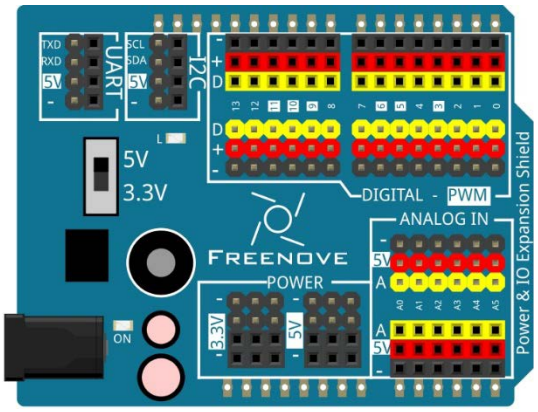
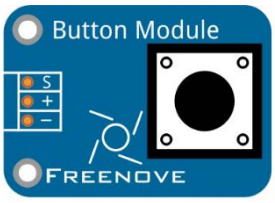
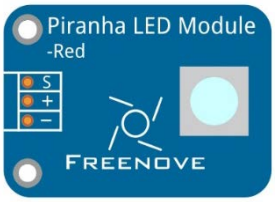
Project 1.2 Control Piranha LED Module Via Button Module

Usually, there are three essential parts in a complete automatic control device: INPUT, OUTPUT, and CONTROL. In the last section, the LED module is the output part and UNO Freenove is the control part. In practical applications, we not only just let the LED lights flash, but make the device sense the surrounding environment, receive instructions and then make the appropriate action such as lights the LED, make a buzzer beep and so on.



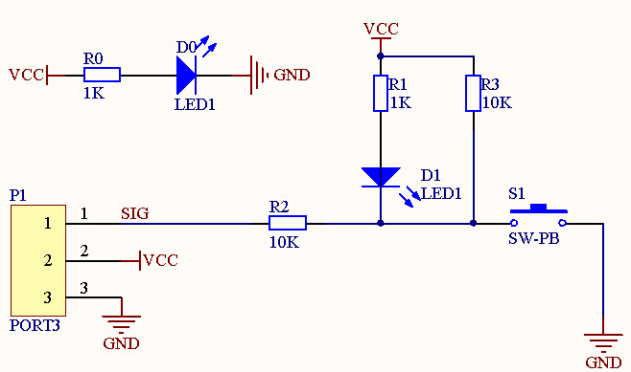
Next, we will build a simple control system to control LED through a button.

Component List

Freenove UNO x1 	Freenove Power & I/O Expansion Shieldx1 
USB cable x1 	Button Module x1 
Jumper M/Fx6 	Piranha LED Module x1 

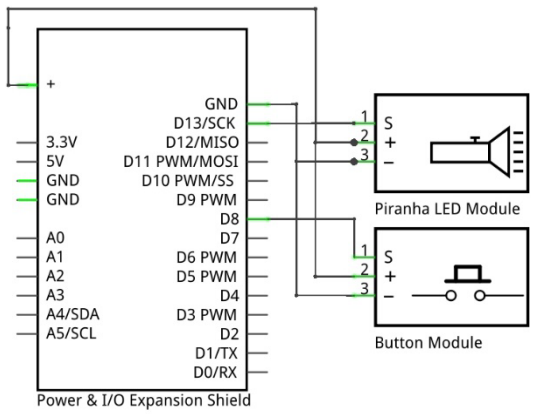
Component Knowledge

The circuit diagram of Module Button is shown below.

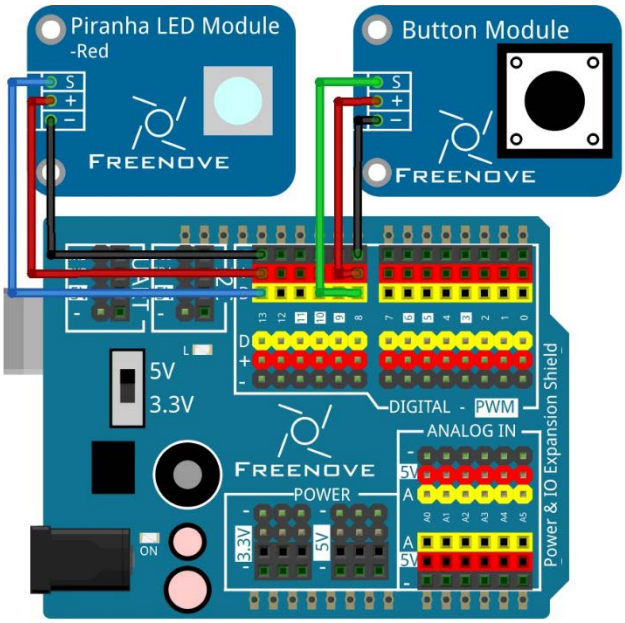
Circuit diagram： 	Analysis： According to the circuit, when the Button is not pressed, the SIG pin is in high level. When the button is pressed, the SIG is in low level.

Circuit

Schematic diagram：



Hardware connection：



Sketch

Sketch 1.2.1 Button_LED

First, define the Button pin and the LED pin, and set corresponding input and output mode of different pins. Then in the loop () function, detect if the button is pressed. If it is pressed, LED will be turned on, otherwise LED is turned off.

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

Verify and upload the code to UNO, then press the button, LED is turned on, and release the button, LED is turned off.

digitalRead(pin)

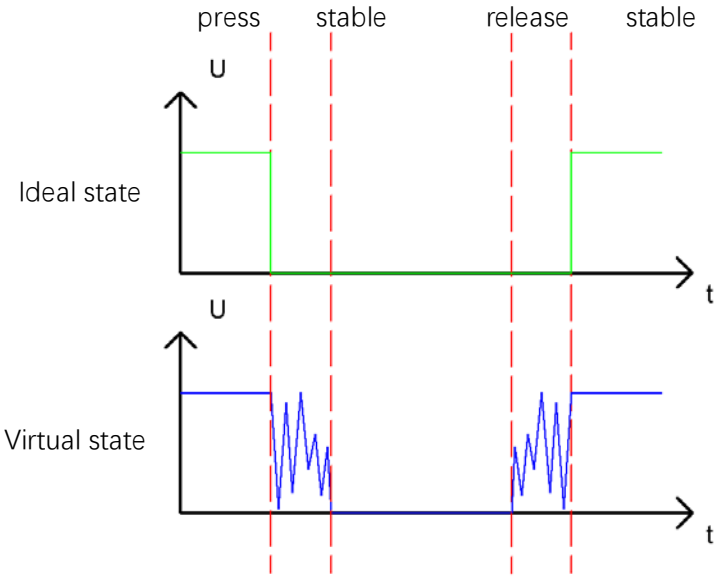
Get the level state of the port pin, and the return value is HIGH (high level) or LOW (low level).

Project 1.3 DIY MINI Table Lamp

We will also use a button, LED and UNO to make a MINI table lamp. But the function is different: Press the button, the LED will be turned on, and press the button again, the LED goes out.
First, let us learn some knowledge about the button.

Eliminating Buffeting of Push Button

When a Push Button is pressed, it will not chang from one state to another state immediately. Due to mechanical vibration, there will be a continuous buffeting before it becomes another state. And the releasing-situation is similar with that process.



Therefore, if we directly detect the state of Push Button, there may be multiple pressing and releasing action in one pressing process. The buffeting will mislead the high-speed operation of the microcontroller to cause a lot of false judgments. So we need to eliminate the impact of buffeting. Our solution is: to judge the state of the button several times. Only when the button state is stable after a period of time, can it indicate that the button is pressed down.
This project needs the same components and circuits with the last section.

Sketch

Now, we began to write the code of MINI table lamp. We will make a concrete analysis of the code later.
Sketch 1.3.1 MINI_tableLamp

```
1  int ledPin = 13;//Define the pin of LED
2  int buttonPin = 8;//Define the pin of button
3  int ledState = LOW;//Record the state of LED; the initial state is OFF
```

```
int buttonState = HIGH;// Record the state of botton; the initial state is HIGH
int lastButtonState = HIGH;//Record the last state of button
long lastChangeTime = 0;// record time point of state chang
long captureTime = 50;// set the time needed to get stable state
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
  digitalWrite(ledPin, ledState);// Set the initial state of LED
}
void loop() {
  int reading = digitalRead(buttonPin);// Read the current state of button
  if (reading != lastButtonState) {
// If the button state has changed, record the time point.
    lastChangeTime = millis();
  }
  // If changing-state of the button last beyond the time we set, we considered that
the current button state is an effective change rather than a buffeting.
  if (millis() - lastChangeTime > captureTime) {
    // Determine if the state is changed. If it is, update and save the state of button.
    if (reading != buttonState) {
      buttonState = reading;
 // If the state is high, the action is pressing. If it is low, the action is releasing.
      if (buttonState == LOW) {
        ledState = !ledState;
      }
    }
  }
  digitalWrite(ledPin, ledState);// In each cycle, the state of the LED will be updated.
  lastButtonState = reading;// Update the last Button State ,and then start the next
round of reading.
}
```

Verify and upload the code to UNO, then press the button, the LED will be turned on, and press the button again, the LED goes out.

Analysing the Code

This code focuses on eliminating the buffeting of button. We define several variables to save the state of LED and button. Then read the button state in loop () constantly, and determine whether the state has changed. If it is, record this time point.

```
  int reading = digitalRead(buttonPin);// Read the current button state.
  if (reading != lastButtonState) {// If the state is changed, record the time point.
    lastChangeTime = millis();
  }
```

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

Then according to just recorded time point, judge the duration of the button state change. If the duration exceeds captureTime (buffeting time) we set, it indicates that the state of the button has changed. During that time, the loop () is still detecting the state of the button, so if there is a change, the time point of change will be updated. Then duration will be judged again until the duration of there is a stable state exceeds the time we set.

```
if (millis() - lastChangeTime > captureTime){
    if (reading != buttonState) {
      buttonState = reading;
.....
```

Finally, judge the state of Button. And if it is low level, the changing state indicates that the button is pressed, if the state is high level, then the button is released. Here, we change the status of the LED variable, and then update the state of LED.

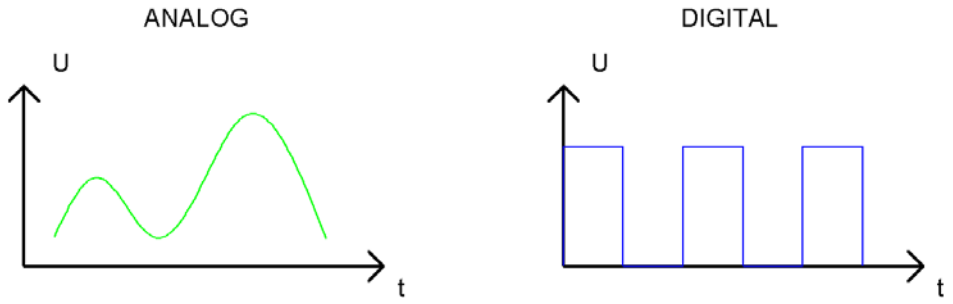
```
      if (buttonState == LOW) {
        ledState = !ledState;
      }
```

Chapter 2 Digital Signals and Analog Signals

In the previous study, we have known that the button has two states: pressed and released, and LED has light-on/off state, then how to enter a middle state? How to output an intermediate state to let LED "semi bright"? That's what we're going to learn.

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 appea a sudden change directly from 0C° to 10C°, 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.

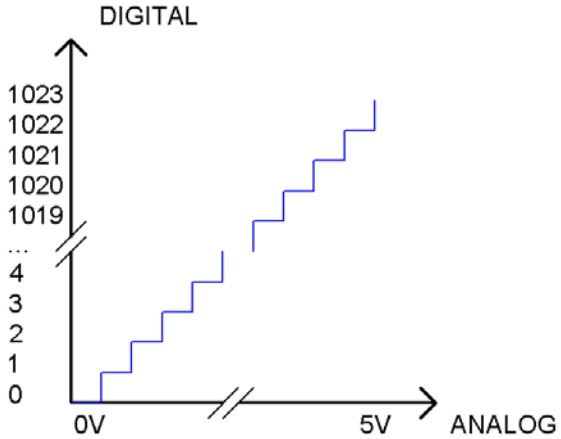


In practical application, we often use binary signal as digital signal, that is 0 and 1. The binary signal only has two forms (0 or 1), so it has strong stability. And digital signal and analog signal can be converted to each other.

Project 2.1 Display the Read voltage value of potentiometer through Serial

ADC

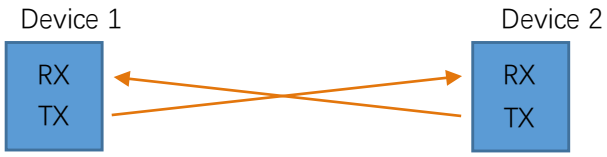
ADC, Analog-to-Digital Converter, is a device used to convert analog to digital. Freenove UNO has a 10 bits ADC, that means the resolution is 2^10=1024, and the range (here is 5V) will be divided equally into 1024 parts. The analog of each section corresponds to one ADC value. So the more bits ADC has, 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.
Analog pins A0-A5 of UNO Freenove are the ADC analog input pins.

Serial communication

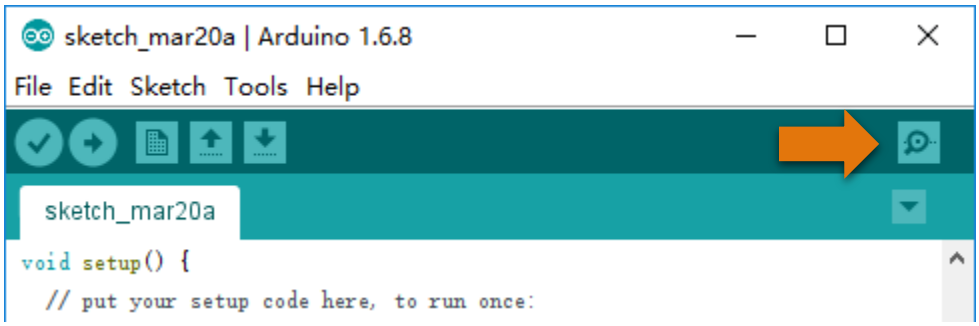
Serial communication generally refers to the Universal Asynchronous Receiver/Transmitter (UART). And it is commonly used in electronic circuit communication. It has two communication lines, one is responsible for sending data (TX line) and another for receiving data (RX line). The serial communication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



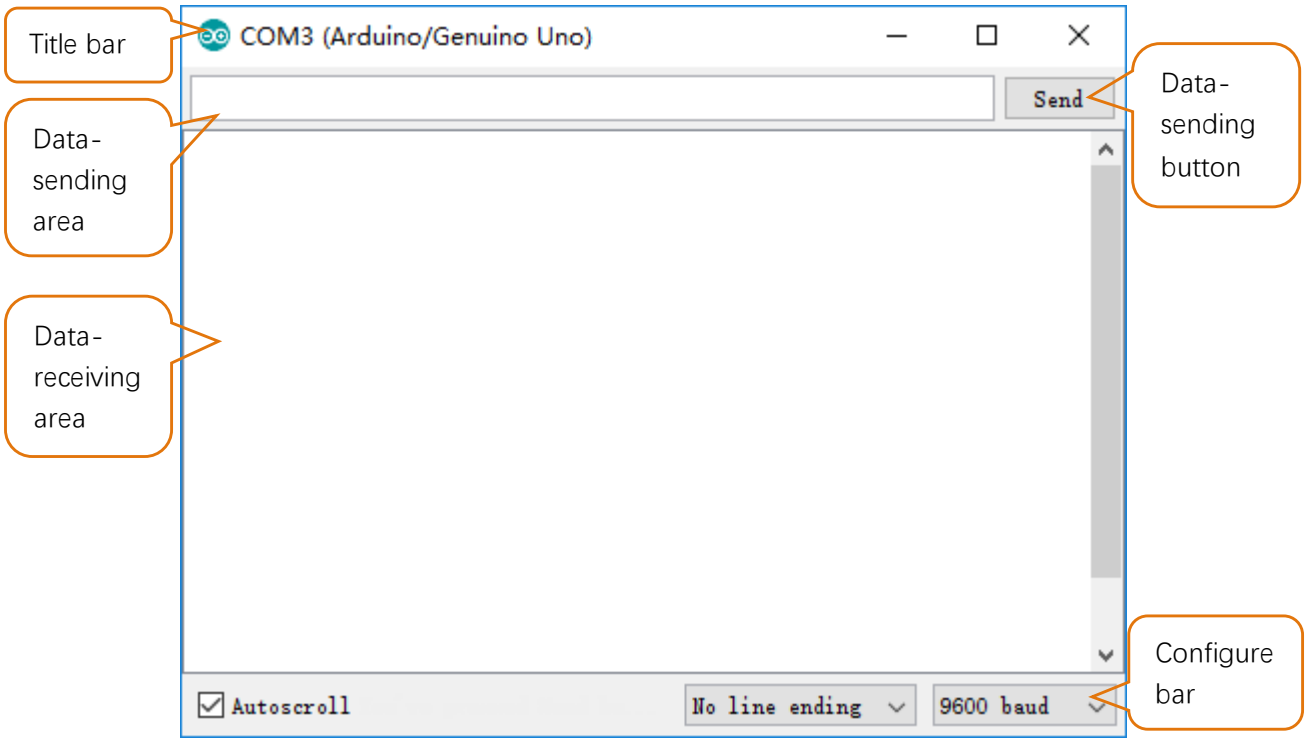
RX and TX pins of UNO are Digital I/O 0 and 1 respectively.
Before serial communication starts, the baud rate in both sides must be the same. Only use the same baud rate can the communication between devices be normal. The commonly used baud rates are 9600 and 115200.

Serial ports on Freenove UNO

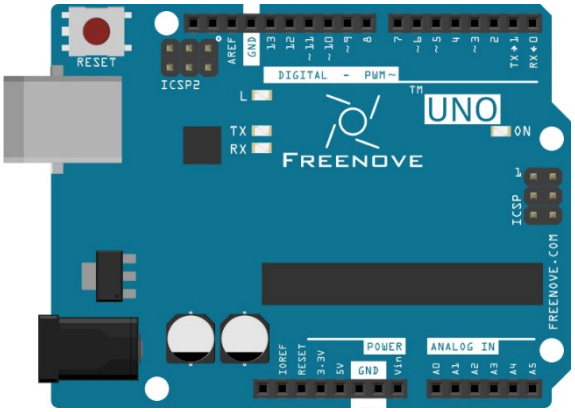
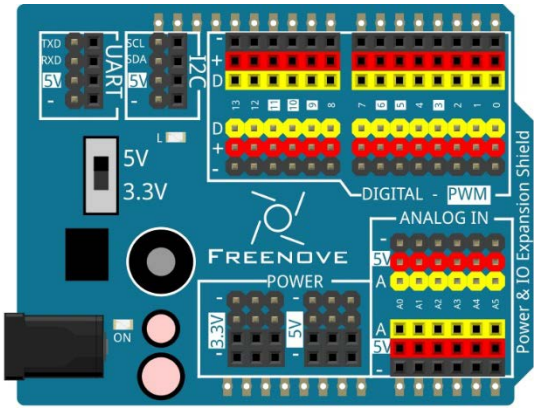
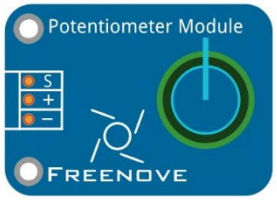
Freenove UNO has integrated USB to serial. When we use USB cable to connect Freenove UNO to computer, the two devices can communicate with each orther. Arduino Software uploads code for Freenove UNO through serial connection. Computer identifies serial devices connected to your computer as COMx. We can 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".



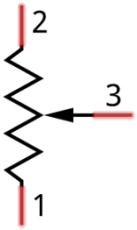
Component List

Freenove UNO x1 	Freenove Power & I/O Expansion Shieldx1 
USB cable x1 	Potentiometer Module x1 
Jumper M/F x3 	

Component Knowledge

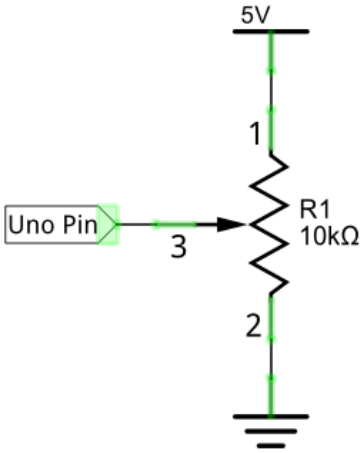
Potentiometer

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.

Linear potentiometer	Rotary potentiometer	Circuit diagram
		

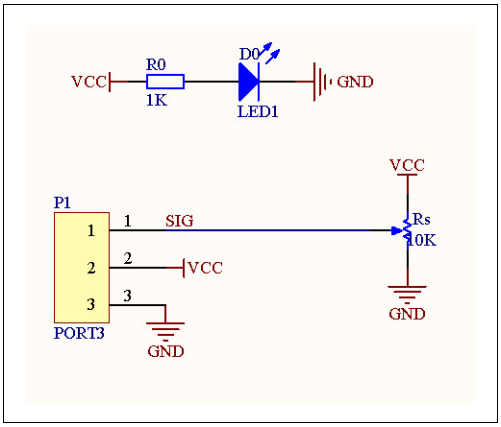
Pin 1 and pin 2 of the potentiometer is connected to two ends of a resistor body respectively, and pins 3 is connected to a brush. When the brush moves from pin 1 to pin 2, resistance between pin 1 and pin 3 will increase up to max resistance of the resistor body linearly, and resistance between pin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

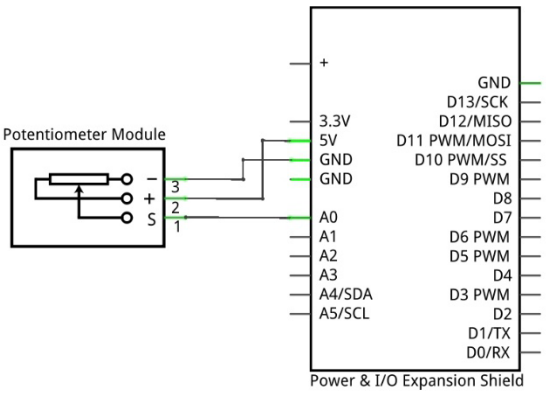
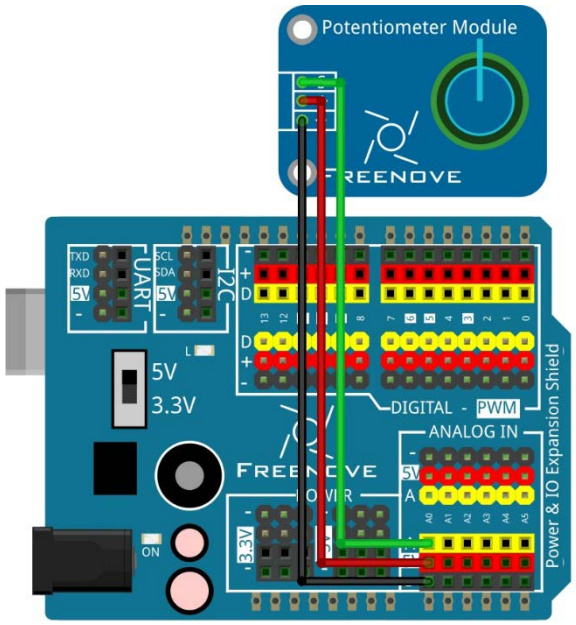


Next we will read the ADC value of the potentiometer and use serial port to print the value out.

The circuit diagram of Module Potentiometer is shown below.



Circuit

Schematic diagram	Hardware connection
	

Sketch

Now let 's write the code to read the analog voltage of A0 port and use serial to print it out. Detailed analysis of the code will be carried out later.

Sketch 2.1.1 AnalogIn_Serial

```
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 value of A0 port to digital
11    voltage = adcValue * (5.0 / 1023.0);// Calculate voltage according to digital value
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  }
```

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

```
    adcValue = analogRead(A0);
```

Serial Class

Serial class is an integration class of IDE Arduino, including the serial port setting and serial reading and writting operations. There are a few common member functions of the class below:

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

Serial.print(val)： Send strings, the parameter is the content needed to send;

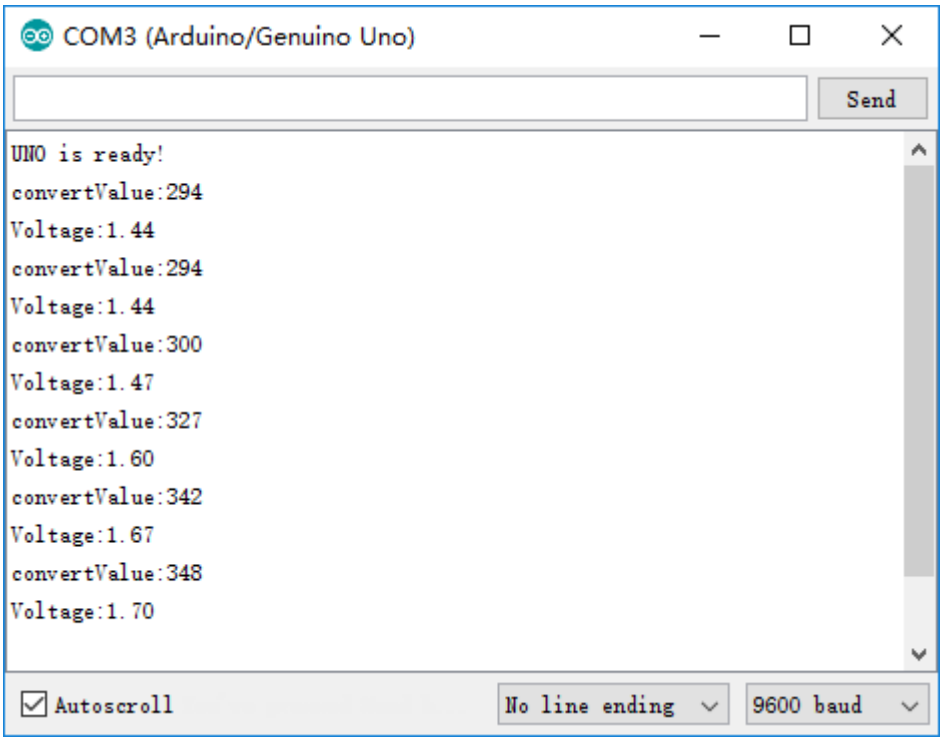
Serial.println(val)： After sending string, send newlines.

For more details of Serial class, please refer to

https://www.arduino.cc/en/Reference/Serial

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

Shift the rotatary potentiometer, and you will see the changes of the voltage value.



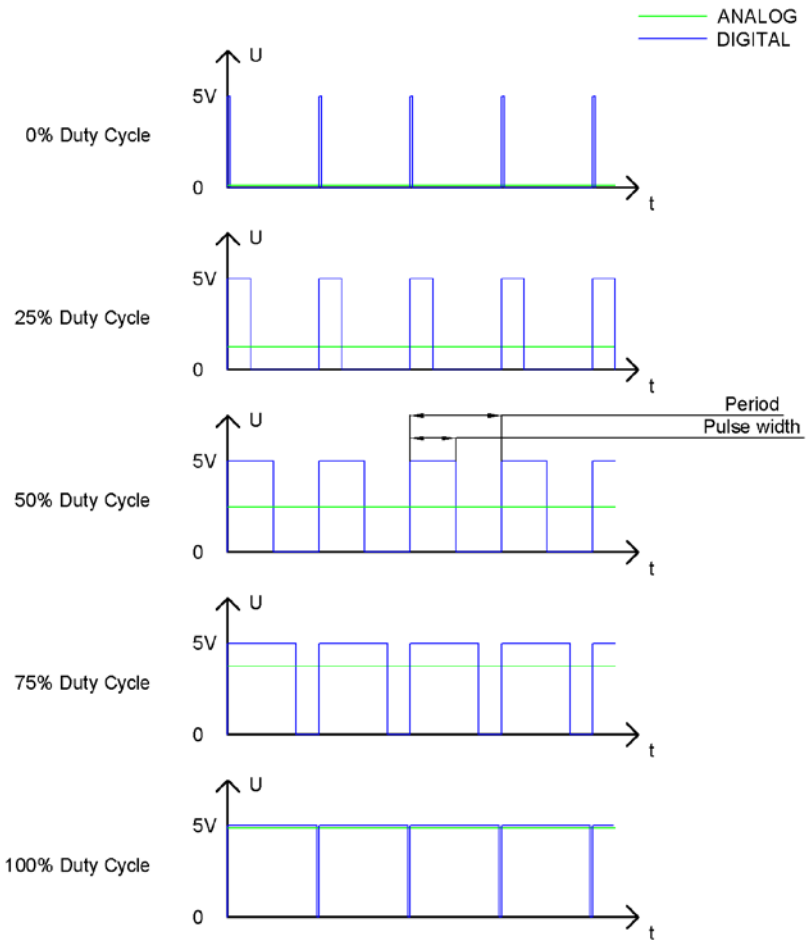
Project 2.2 Breathing LED

We have learned how to read and use the analog value before. Now we will continue to learn how to output analog.

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 make 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 that last for a while 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). The time of high level outputting 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 will be. The following figures show how the analog signals voltage vary between 0V-5V (high level is 5V) corresponding to the pulse width 0%-100%:



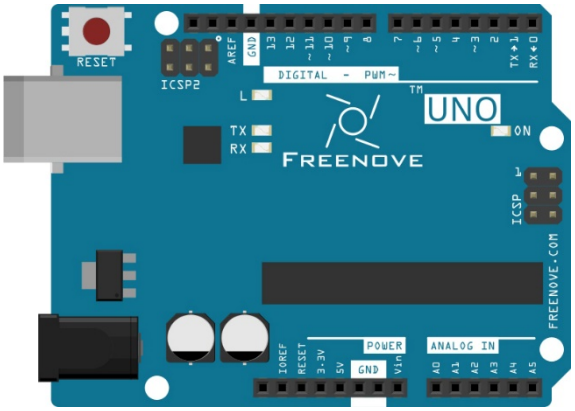
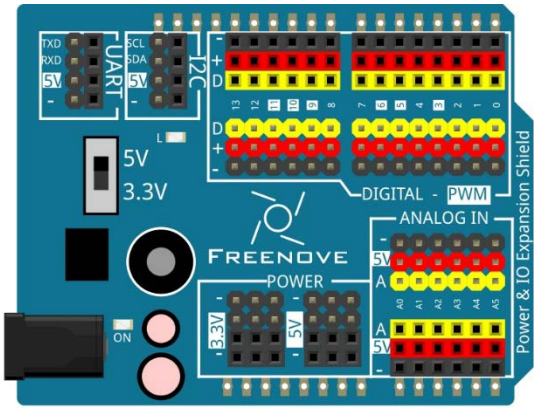
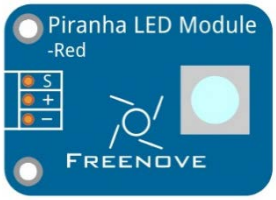
The larger PWM duty cycle is, the lager the output power will be. So we can use PWM to control the brightness of LED, the speed of DC motor and so on.

It is evident from the above that PWM is not real analog, and the effective value of the voltage is equivalent to the corresponding analog. so, we can control the output power of the LED and other output modules to achieve different effects.

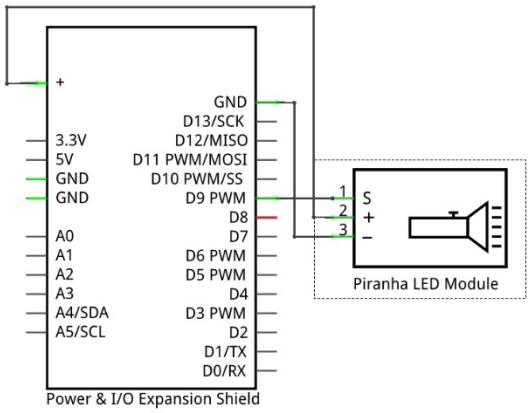
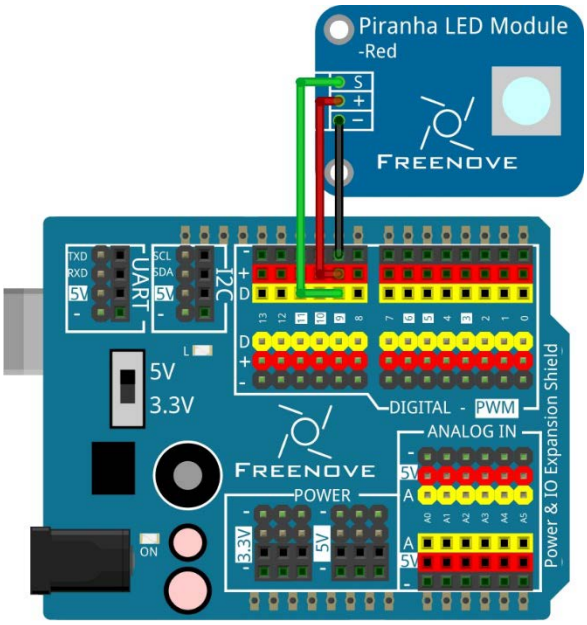
On the Freenove UNO, I/O port 3, 5, 6, 9, 10 and 11 can output PWM with 8 bits accuracy, so the 100% duty cycle is devided into 2^8=256 shares.

Next, we will make LED from off to on, and then from on to off gradually, which is called "Breathing LED".

Component List

Freenove UNO x1 	Freenove Power & I/O Expansion Shieldx1 
USB cable x1 	Piranha LED Module x1 
Jumper M/F x3 	

Circuit

Schematic diagram	Hardware connection
	

Sketch

Now write the code for the Breathing LED, and detailed analysis of the code will be carried out later.

Sketch 2.2.1 Breathing_LED

```
1  // set pin numbers:
2  int ledPin = 9;             // 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() function circularly
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”  (number of cycles) is from 0 to 255
17     analogWrite(ledPin, i);         // i from 0 to 255 is consistent with duty ratio
18 from 0% to 100%
19     delay(delayMs);                 // Adjust change rate of LED brightness
20   }
```

```
21    for (int i = 255; i >= 0; i--) {  //  “i”  (number of cycles) is from 255 to 0
22      analogWrite(ledPin, i);      // i from 0 to 255 is consistent with duty ratio from
23  0% to 100%
24      delay(delayMs);           // Adjust change rate of LED brightness
25    }
26  }
```

Verify and upload the code to UNO, then you can see the LED from off to on, then form on to off gradually. This kind of chang will loop continuously.

analogWrite(pin, value)

Function: set the pin to output PWM with duty cycle “value ”
pin：the pin to write to.
value：the duty cycle: between 0 (always off) and 255 (always on).

We defined a subfunction void breath (int ledPin, int delayMs) above. Then through two cycles, the PWM duty cycle changes from 0% to 100%, and then from 100% to 0% circularly. In the for cycle function, we use delay (ms) function to control the change rate in brightness. You can try to modify the second parameter of this subfunction to adjust the change rate of LED ’ brightness.

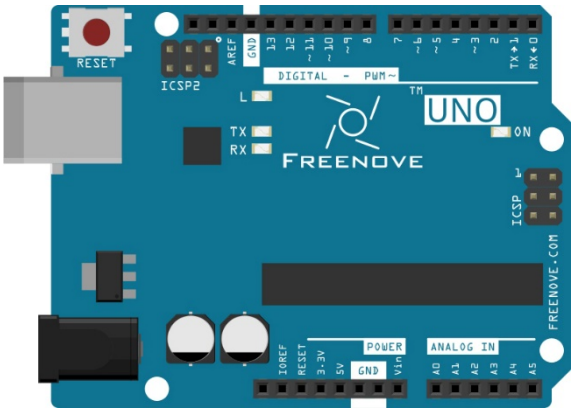
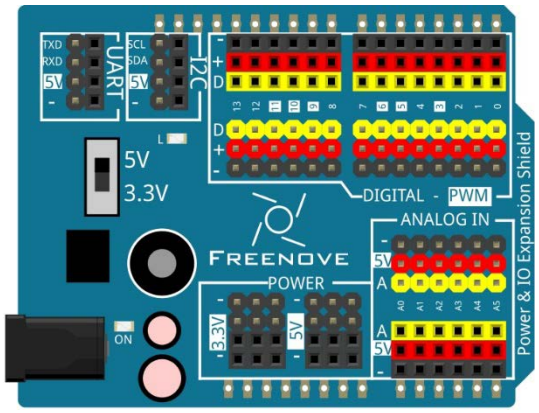
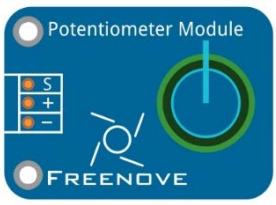
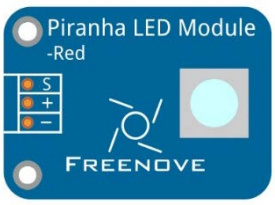
Chapter 3 Potentiometer&LED

We have studied analog input and PWM output. Now we can combine these two things, and try to make a small table lamp with adjustable brightness.

Project 3.1 MINI Table Lamp with Adjustable Brightness

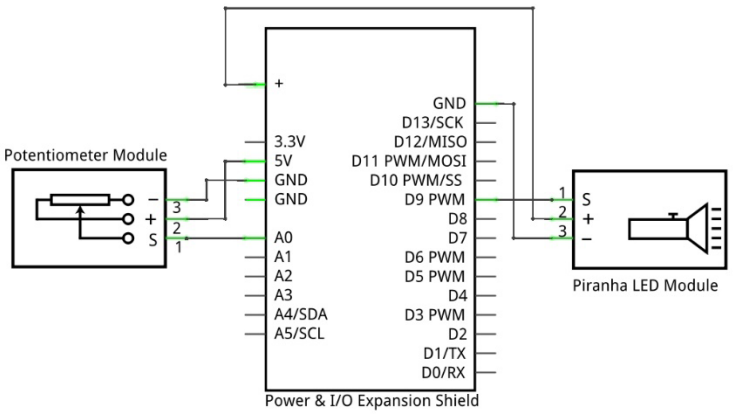
Let ' s combine Potentiometer Module and Piranha LED Module to make a MINI lamp with adjustable brightness.

Component List

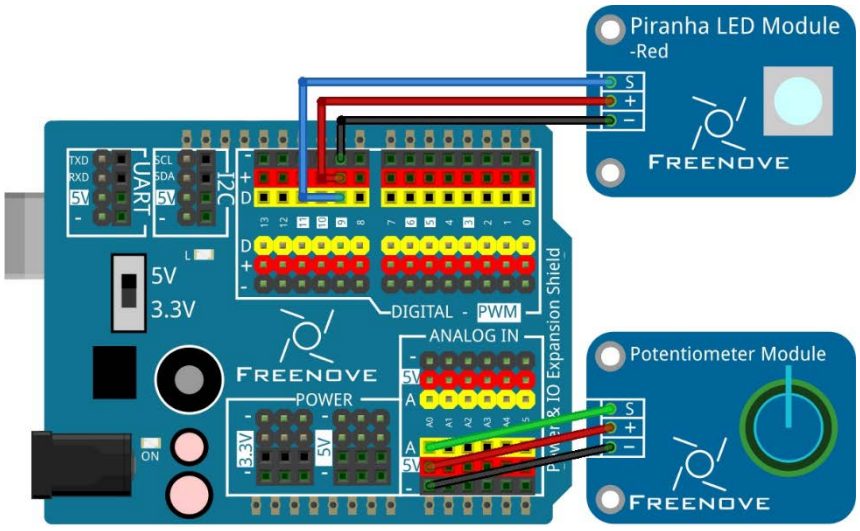
Freenove UNO x1 	Freenove Power & I/O Expansion Shieldx1 	
USB cable x1 	Potentiometer Module x1 	Piranha LED Module x1 
Jumper M/F x6 		

Circuit

Schematic diagram



Circuit connection



Sketch

Now let ’s write the code to adjust the brightness of the MINI lamp. And detailed analysis of the code will be carried out later.

Sketch 3.1.1 adjustable_LED

```
1  int adcValue;     // 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);            // initialize the LED pin as output
6  }
7
8  void loop() {
9    adcValue = analogRead(A0);          // Convert analog of A0 port to digital
```

```
10    // Map the converted digital to the range 0-255 as the 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 pin A0, and map it to the PWM duty cycle of LED pin port to control brightness of the LED.

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

This function is used to map a value from one range to another range, which will return a new value whose percent in the rang of toLow-toHigh is equat to the percent of "value" accounting for in range of fromLow-fromHigh.

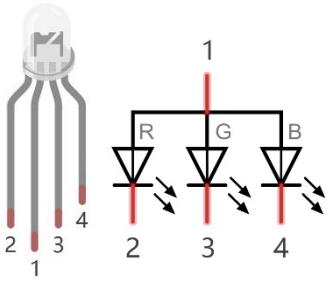
Verify and upload the code to UNO. Then turn the potentiometer， and you will see the brightness of the LED changes.

Chapter 4 RGBLED

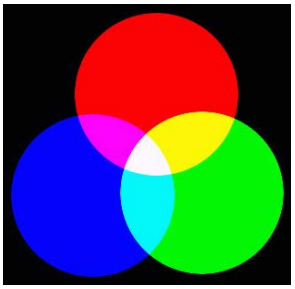
In this chapter, we will learn about a colorful LED.

Project 4.1 Colorful LED

In this project, we will use RGBLED Module to make a colorful 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 positive or negative port. The RGB LED with common positive port and its symbol are shown below. We can make RGB LED emit various colors of light by controlling these 3 LEDs to emit light with different brightness,



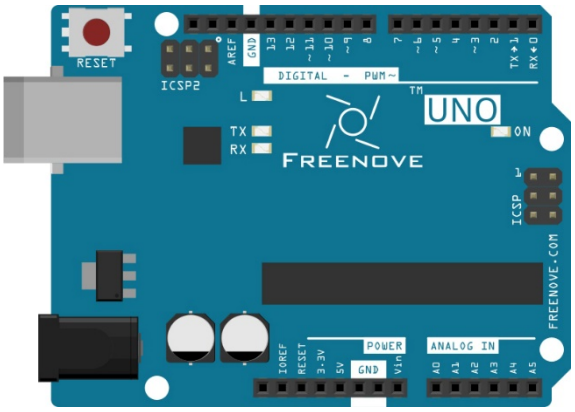
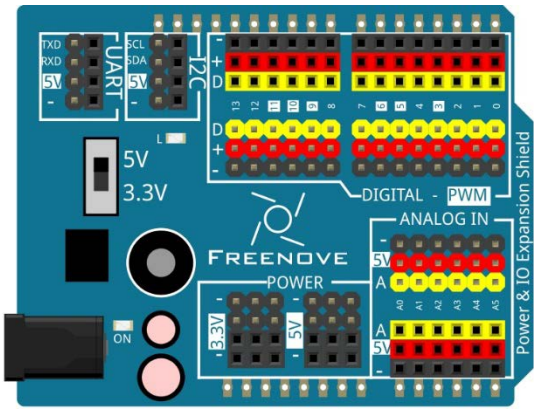
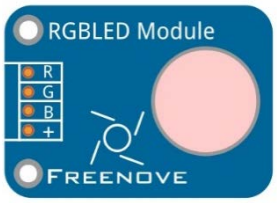
Red, green, and blue light are called 3 primary colors. When you combine these three primary-color light with different brightness, it can produce almost all kinds of visible lights. Computer screens, single pixel of cell phone screen, neon, and etc. are working under this principle.



RGB

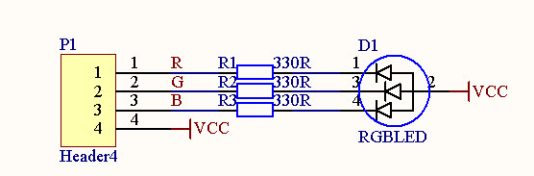
We have known before that UNO can control LED to emit total 256(0-255) different kinds of brightness by PWM. So, through combinating three different colors of LED, RGB LED can emit 256^3=16777216 (about 16Million) colors.

Component List

Freenove UNO x1 	Freenove Power & I/O Expansion Shieldx1 
USB cable x1 	RGBLED Module x1 
Jumper M/F x4 	

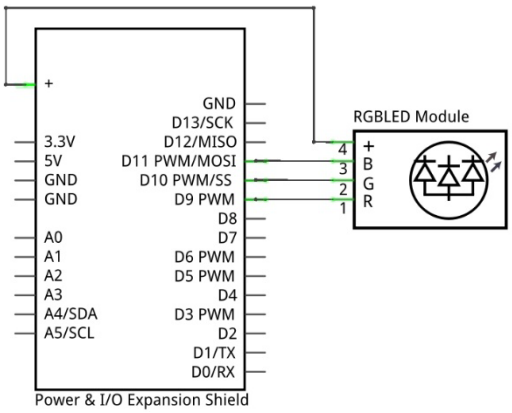
Component Knowledge

The circuit diagram of Module RGBLED is shown below.

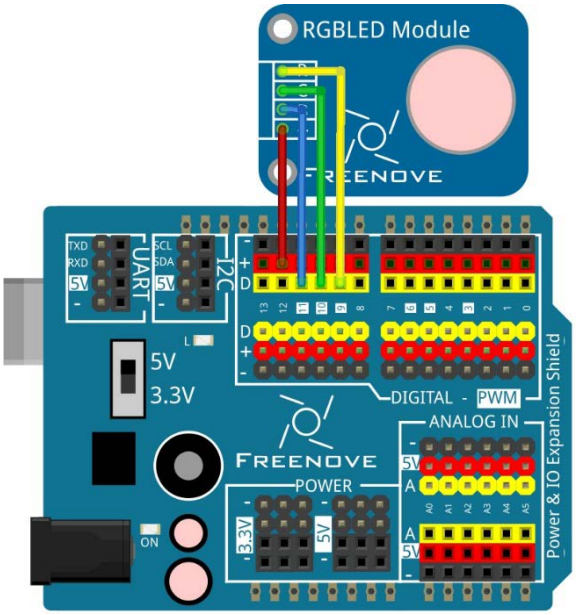


Circuit

Schematic diagram



Circuit connection



Sketch

Now let 's write the code for colorful LED. And detailed analysis of the code will be carried out later.

Sketch 4.1.1 Colorful_LED

```
1  // set pin numbers:
2  int ledPinR = 9;   // the number of the LED R pin
3  int ledPinG = 10;   // the number of the LED G pin
4  int ledPinB = 11;    // 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 //Generates 3 random numbers between 0-255 as 3 duty cycles of the PWM outputed by
15 ledPin.
16   rgbLedDisplay(random(256), random(256), random(256));
17   delay(500);
18 }
19
20 void rgbLedDisplay(int red, int green, int blue) {
```

```
21      // set 3 duty cycles of the PWM outputed by ledPin.
22      analogWrite(ledPinR, constrain(red, 0, 255));
23      analogWrite(ledPinG, constrain(green, 0, 255));
24      analogWrite(ledPinB, constrain(blue, 0, 255));
    }
```

In the code, we generate three random numbers, 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 with different colors and brightness.

random(min, max)
random (min, max) function is used to generate random numbers, 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).
constrain(x, a, b)
Constrain a number to be within a range.

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

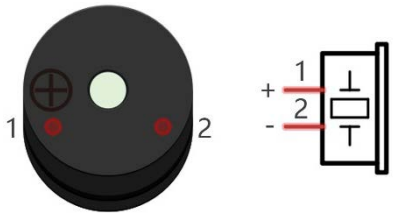
Chapter 5 Buzzer

In this chapter, we'll learn a sounding module, Passive Buzzer Module.

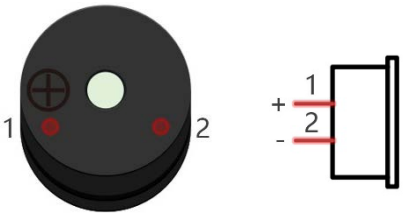
Project 5.1 Alertor

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



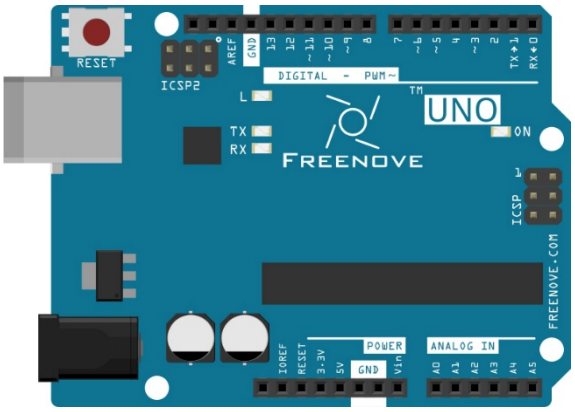
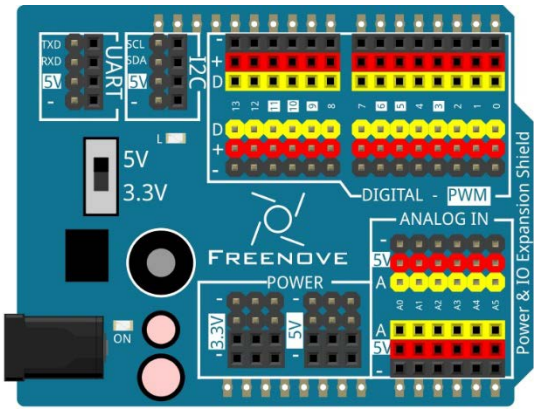
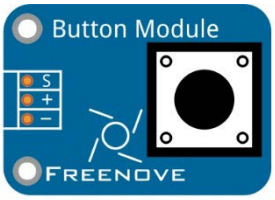
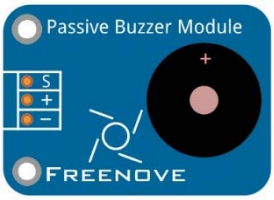
Passive buzzer



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. So we can use PWM with different frequency to make a sound with different frequency.

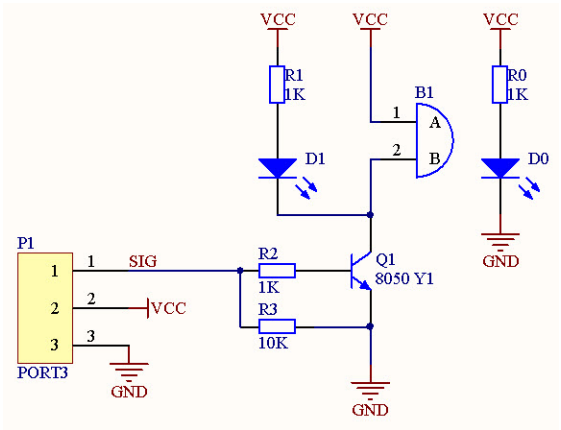
In this project, we use Passive buzzer. Its resonant frequency is 2kHz, which means the passive buzzer is loudest when its frequency is 2kHz.

Component List

Freenove UNO x1 	Freenove Power & I/O Expansion Shieldx1 	
USB cable x1 	Button Module x1 	Passive Buzzer Module x1 
Jumper M/F x6 		

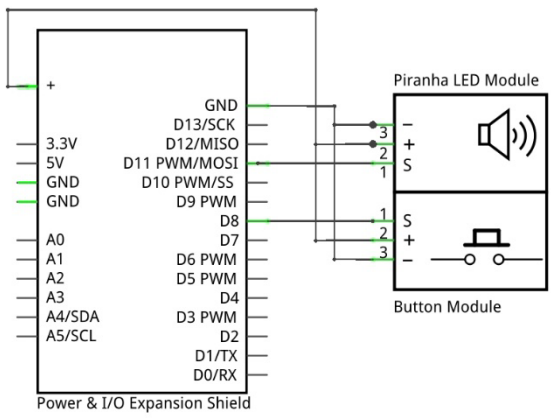
Component Knowledge

The circuit diagram of Buzzer Module Passive is shown below.

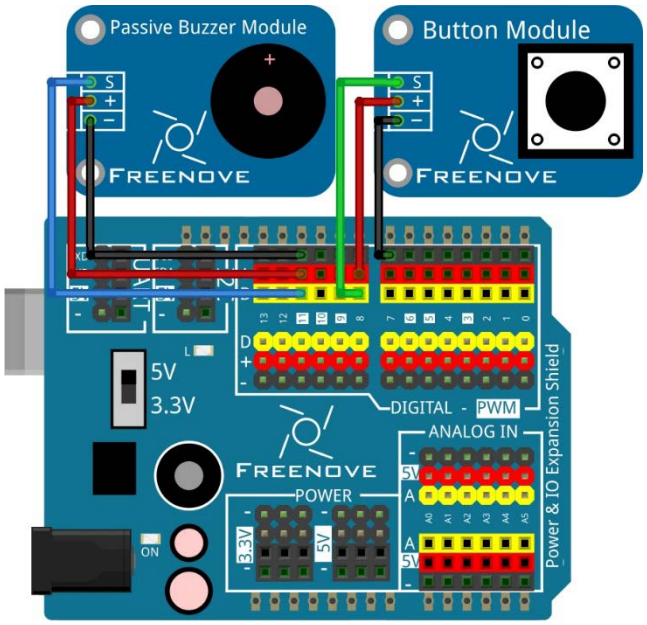
Circuit diagram 	Analysis: According to the circuit diagram, we can see that the buzzer is driven by a NPN transistor. When the SIG pin is in high level, the transistor is turned on, and there is current flowing though the buzzer. When the SIG pin is in low level, the transistor is turned off, and there is no current though the buzzer.

Circuit

Schematic diagram



Circuit connection



Sketch

Write the code. When we press the button, the buzzer starts alarming; and release it, the alarming buzzer stops.

Sketch 5.1.1 alerter

```
int buttonPin = 8;   // the number of the push button pin
int buzzerPin = 11;    // the number of the buzzer pin
float sinVal;         // Define a variable to save the sine value
int toneVal;          // Define a variable to save the sound frequency

void setup() {
  pinMode(buttonPin, INPUT);  // Set Push Button pin to input mode
  pinMode(buzzerPin, OUTPUT); // Set Buzzer pin to output mode
}

void loop() {
  if (digitalRead(buttonPin) == LOW)// when the buttonPin is low, the button is pressed.
    alert();//start to alarm
  else                    // when the buttonPin is high, the button is not pressed.
    noTone (buzzerPin);     // turn off Buzzer
}
void alert() {
  for (int x = 0; x < 360; x++) {     // X from 0 degree->360 degree
    sinVal = sin(x * (PI / 180));     // Calculate the sine of x
```

```
20      toneVal = 2000 + sinVal * 500;    // Calculate sound frequency according to the sine of x
21      tone(buzzerPin, toneVal);         // Output sound frequency to buzzerPin
22      delay(1);
23    }
24  }
```

In the code, we detect the state of Push Button. When the button is pressed, the buzzer alarm function will be executed. Otherwise, the buzzer will not alarm. Sound frequency of the alarm can be fitted as a sine curve. In the subfunction alert (), calculate the sine value of each point, and superimposed on the resonant frequency, then use tone () to output square wave of the frequency.

tone(pin, frequency) / tone(pin, frequency,duration)
Generates a square wave of the specified frequency (and 50% duty cycle) on a pin.A duration can be specified, otherwise the wave continues until a call to noTone().
noTone(pin)
Stops the generation of a square wave triggered by tone(). Has no effect if no tone is being generated.

Verify and upload the code to UNO. When you press the button, the buzzer starts alarming; and release it, the alarming buzzer stops.

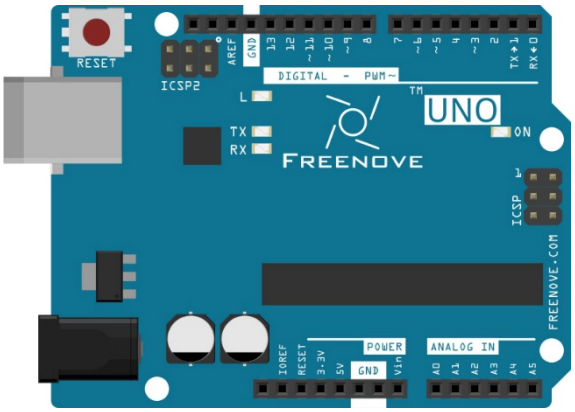
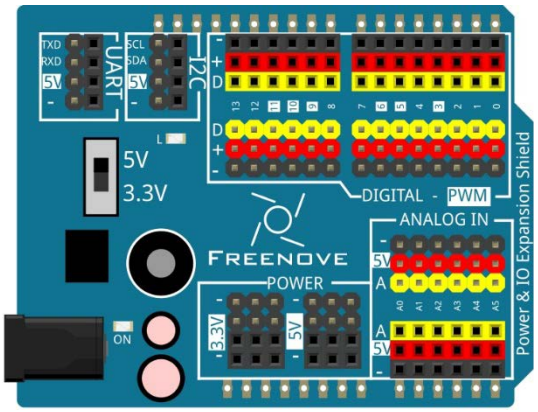
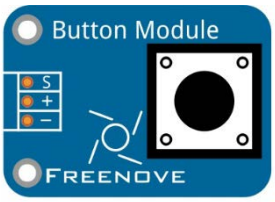
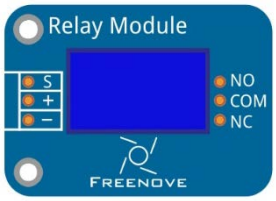
Chapter 6 Relay Module

In this chapter, we will learn a kind of special switch module, Relay Module.

Project 6.1 Relay & Motor

In this project, we will use the button to control the Relay Module, and drive the motor to make a small fan.

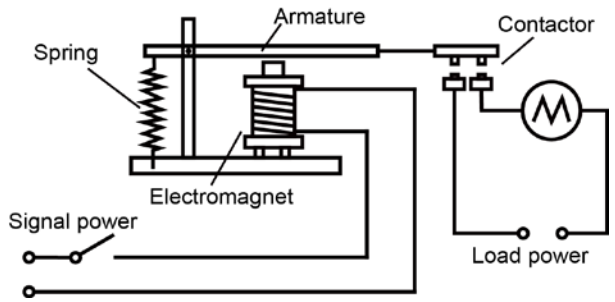
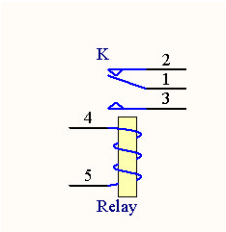
Component List

Freenove UNO x1 	Freenove Power & I/O Expansion Shieldx1 	
USB cable x1 	Button Module x1 	Relay Module x1 
Jumper M/F x7 		
Motor x1 	9V Battery and plug x1 	

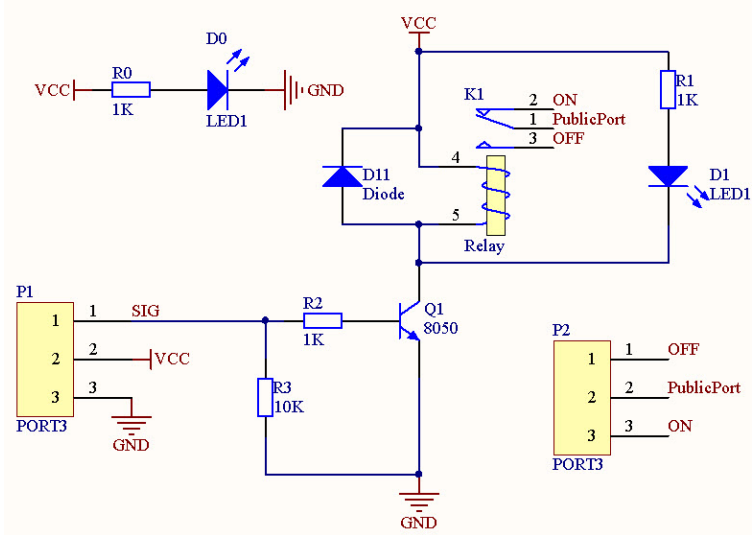
Circuit Knowledge

Relay is a safe switch which can use low power circuit to controll high power circuit. It consists of electromagnet and contacts. The electromagnet is controlled by low power circuit and contacts is used in high

power circuit. When the electromagnet is energized, it will attract contacts. The internal schematic diagram and circuit symbol of common relay are shown below:

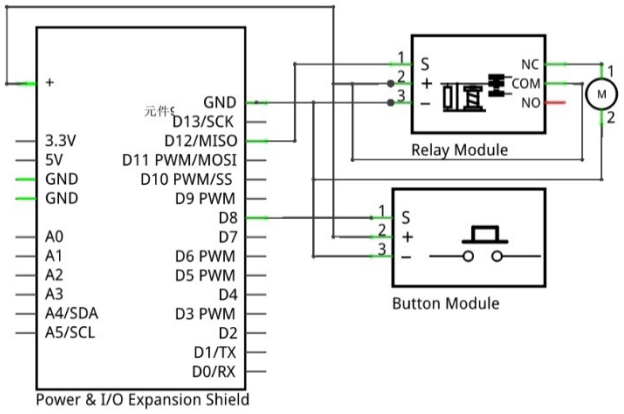
Diagram	Symbol
	

The circuit diagram of Relay Module is shown below:

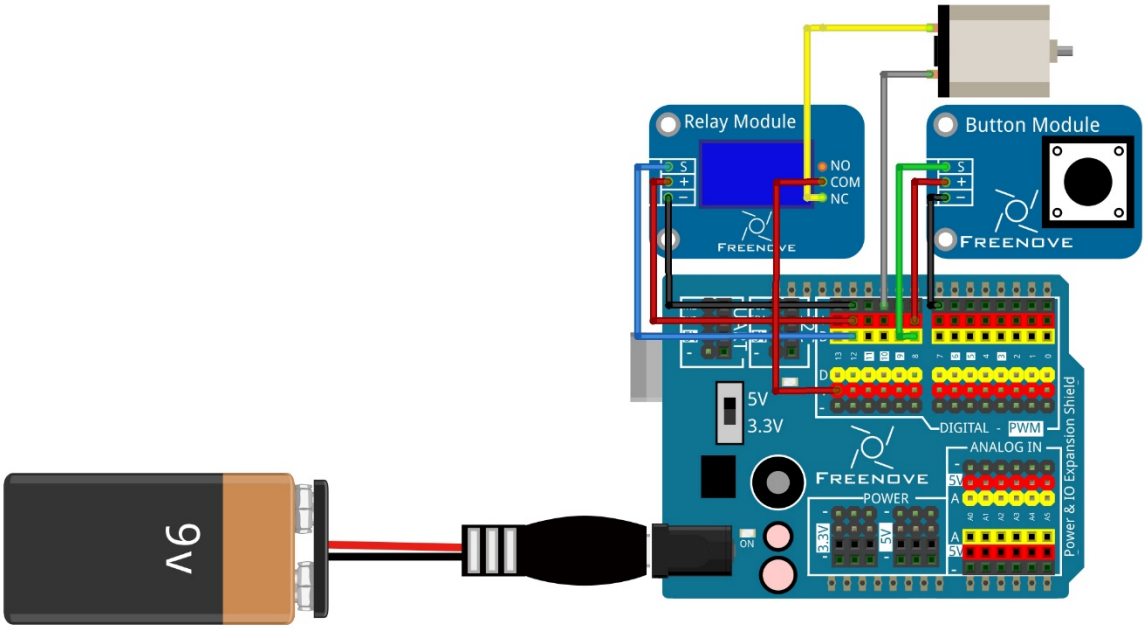
Circuit diagram	Analysis
	According to the circuit diagram, we can see that when the SIG is in high level, triode is turned on, relay contact will be attracted to connect, otherwise, relay contacts will be disconnected.

Circuit

Diagram



Circuit connection



Note: in this project, you need connect the 9V battery to expansion shield and use large power (3A) voltage stabilizing circuit of the expansion shield. When using high power devices such as motors, do not directly use the UNO supply power for them to avoid damage to it.

Sketch

Write the code to realize: Press the button, then the relay contact gets connected and the motor starts to rotate. Press the button again, then the relay is disconnected and the motor stops rotating. This code principle is consistent with the code for MINI table lamp logically.

Sketch 6.1.1 MINI_Fan

```
1  int relayPin = 12;     // the number of the relay pin
2  int buttonPin = 8;    // the number of the push button pin
3
```

```
int buttonState = HIGH; // Record state of the button, the initial state is high level
int relayState = LOW;   // Record state of the relay, the initial state is low level
int lastButtonState = HIGH; // Record the last button state
long lastChangeTime = 0;    // Record time point of state change

void setup() {
  pinMode(buttonPin, INPUT);  // Set Push Button pin to input mode
  pinMode(relayPin, OUTPUT);  // Set Relay pin to output mode
  digitalWrite(relayPin, relayState); // Set initial state of the relay to "close"
}

void loop() {
  int nowButtonState = digitalRead(buttonPin);// Read the current state of the button pin
  // If the button pin state changes, record the time point
  if (nowButtonState != lastButtonState) {
    lastChangeTime = millis();
  }
  // If the state of button has been stable for some time after changing, then we think it has skiped the buffeting period.
  if (millis() - lastChangeTime > 10) {
    if (buttonState != nowButtonState) {  // Confirm that the state of button has changed.
      buttonState = nowButtonState;
      if (buttonState == LOW) {     // If buttonState is low, the button is pressed.
        relayState = !relayState;           // Reverse the state of relay
        digitalWrite(relayPin, relayState); // Update the state of relay
      }
    }
  }
  lastButtonState = nowButtonState; // Save the last state of the button
}
```

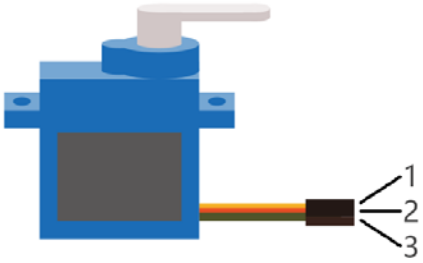
Verify and upload the code to UNO. Every time the Button Push is pressed, the state of the relay and motor will be changed once.

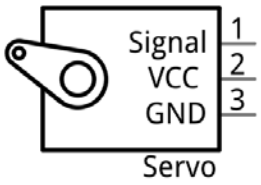
Chapter 7 Servo

In this chapter, we will learn a new "motor", Servo.

Project 7.1 Servo Knob

Servo motor is an auto-control system, consisting of DC motor, reduction gear, sensor and control circuit. Usually, it can rotate in the range of 0-180 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).



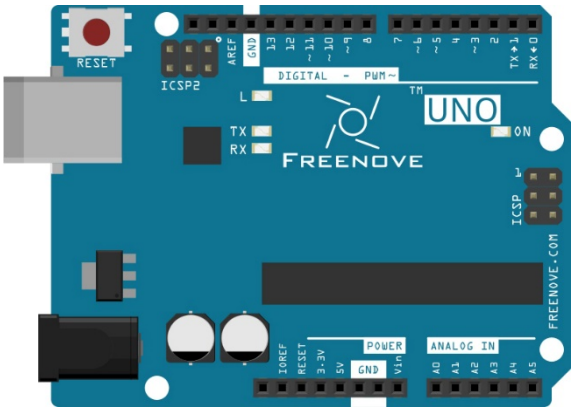
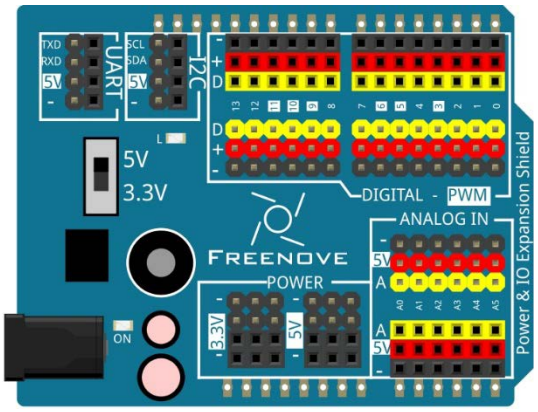
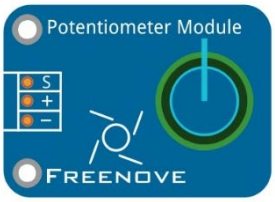
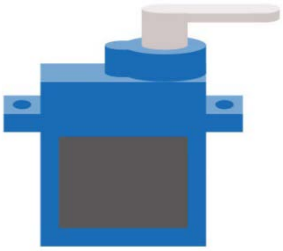


We use 50Hz PWM signal with a duty cycle in a certain range to drive the servo. The lasting time 0.5ms-2.5ms of PWM single cycle high level corresponds to the servo angle 0-180 degrees linearly. Part of the corresponding values are as follows:

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.
In this project, we use potentiometer to control the rotation angle of the servo.

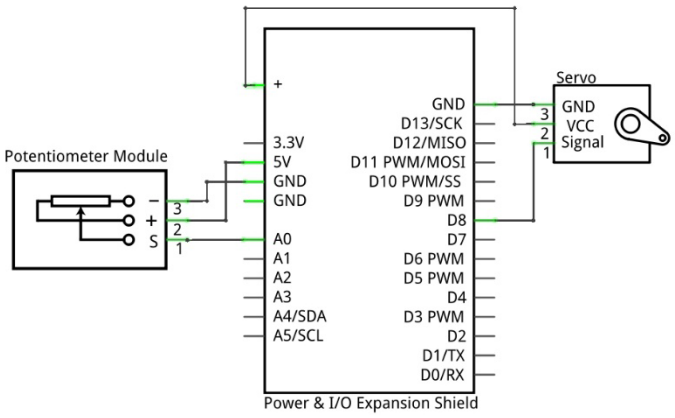
Component List

Freenove UNO x1 	Freenove Power & I/O Expansion Shieldx1 
USB cable x1 	Potentiometer Module x1 
Jumper M/F x3 	Servo x1 

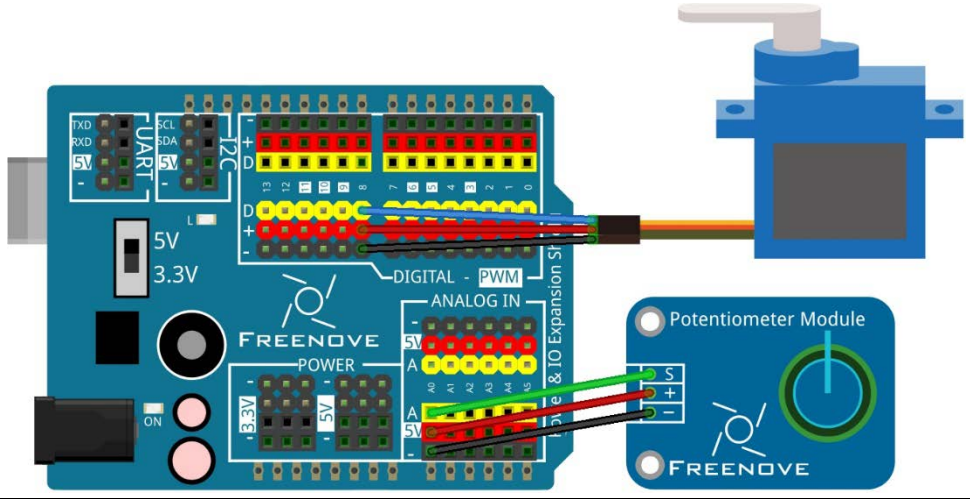
Circuit

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.

Schematic diagram



Circuit connection



Sketch

Write the code to get the ADC value of the potentiometer, and convert it to angle value of servo.

Sketch 7.1.1 Knob

```
#include <Servo.h>

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

int servoPin = 8;           // define the pin of servo signal line
int potPin = A0;            // analog pin used to connect the potentiometer
int potVal;                 // variable to read the potValue from the analog pin

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  }
```

Use the Servo library to control Servo. Reference of the Servo library is shown below:

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

The Servo library provides Servo class to control Servo, which is different from the previous Serial class. And Servo class has to be instantiated before being used:

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

The code above defines a Servo type object, 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, we can refer to the functions, such as the initialization of the servo:

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

Initialize the servo, then we can control the servo to rotate to a specific angle:

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

In this code, we get the ADC value of theA0 pin, and convert it to angle value of servo.

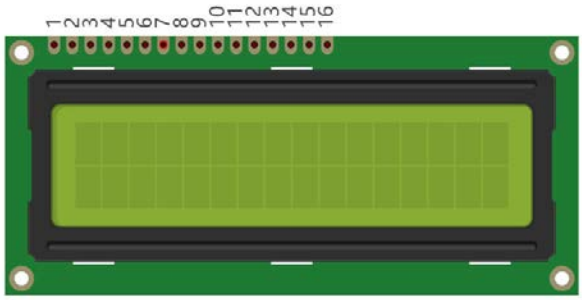
Verify and upload the code to UNO. Turn the potentiometer, then you can see servo roates to corresponding angle.

Chapter 8 LCD1602

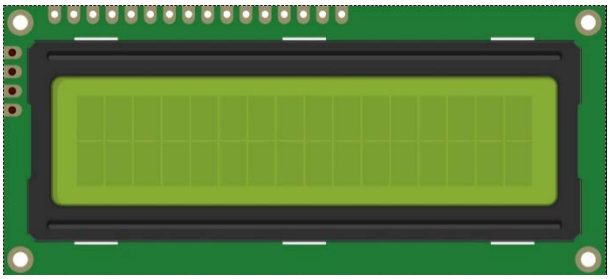
In this chapter, we will learn a display that can display characters like serial port monitoring window, LCD1602.

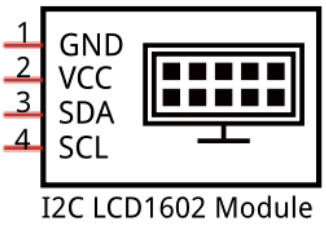
Project 8.1 I2C LCD1602

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:



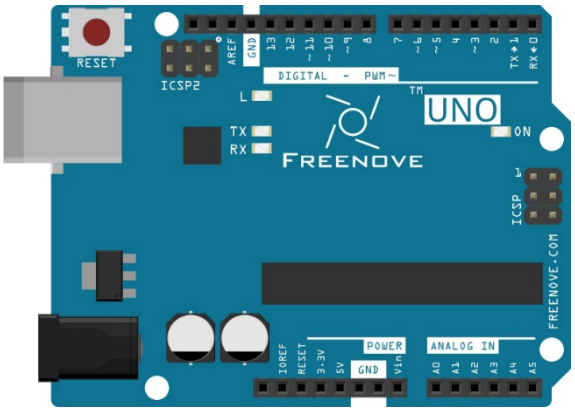
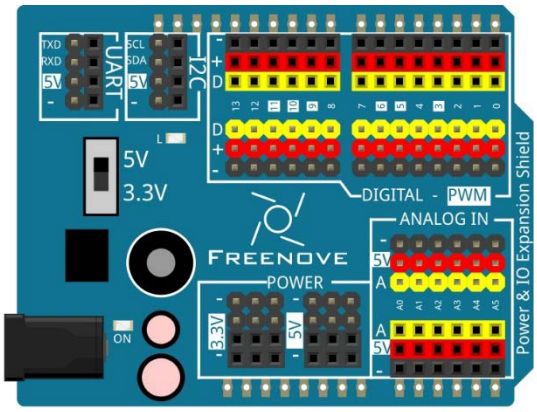
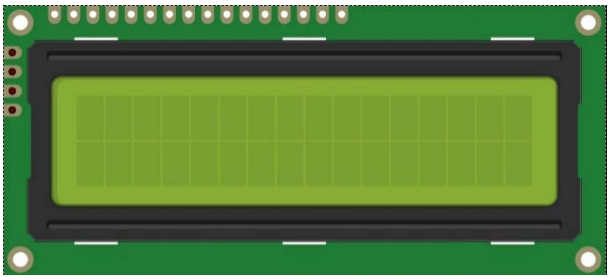
I2C LCD1602 integrates a I2C interface, which connects the serial-input &parallel-output module to LCD1602. We just use 4 lines to the operate LCD1602 easily.



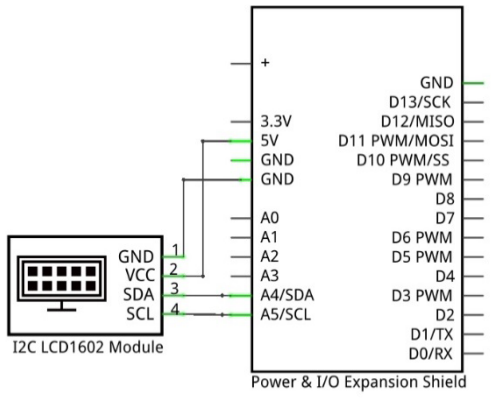
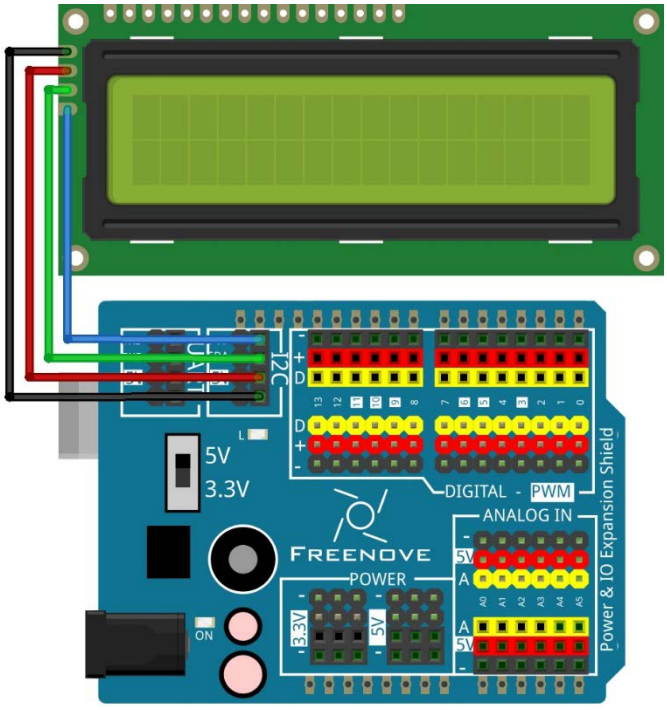


I2C (Inter – Integrated Circuit) is a two-wire serial communication, which is used to connect micro controller and its peripheral equipment. Device that use I2C communication are all connected to the serial data (SDA) line and a serial clock (SCL) line (called I2C bus). Each device among them has a unique address and can be a transmitter or receiver. Additonally, they can communicate with devices connected to the Bus.
Next, let ' s try to use the LCD1602 I2C module to display characters.

Component List

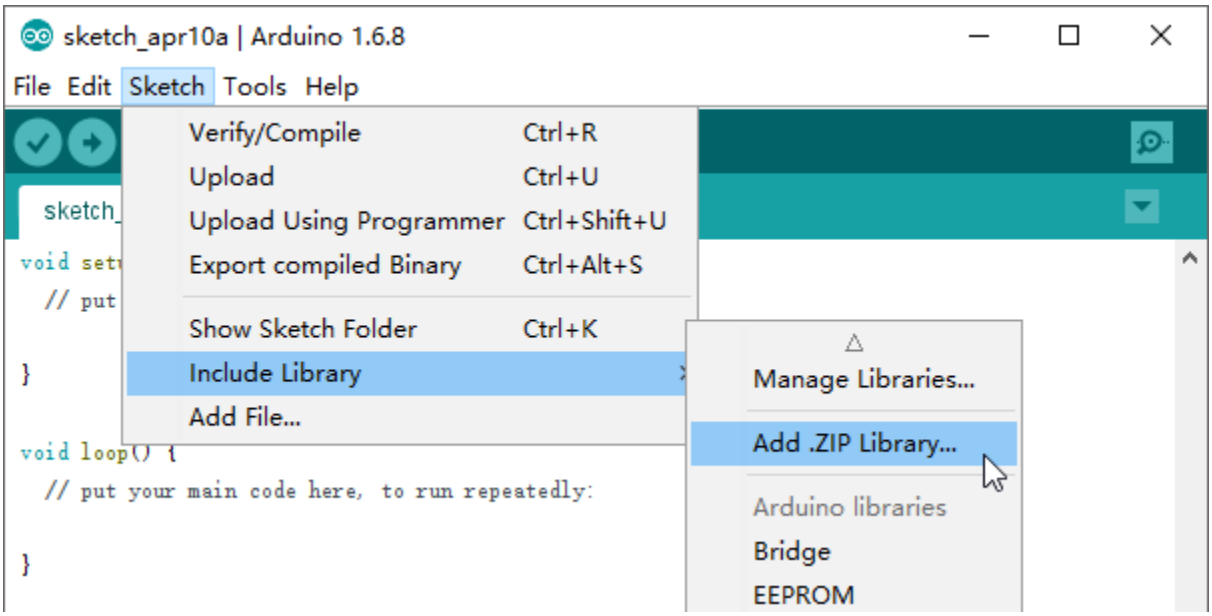
Freenove UNO x1 	Freenove Power & I/O Expansion Shieldx1 
USB cable x1 	I2CLCD1602 Module x1 
Jumper M/F x4 	

Circuit

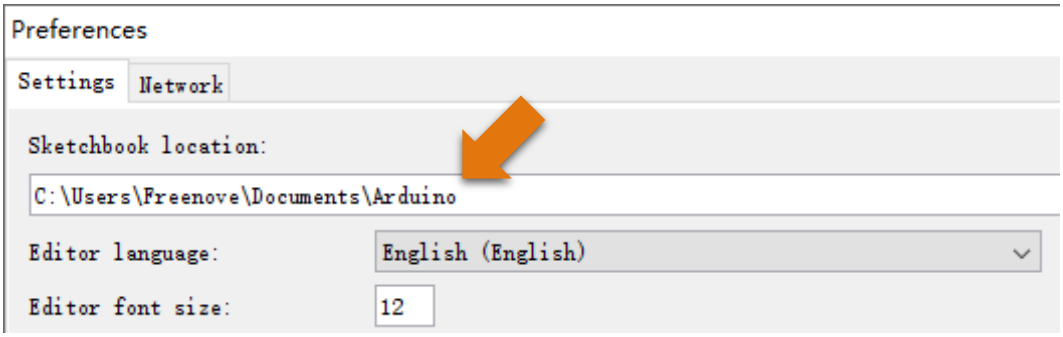
Schematic diagram 	Circuit connection 

Sketch

Before starting to write the code, we need to import the necessary library file. These library files are contributed by other open source developers, which is not previously installed in Arduino Software. Click on the Add.ZIP Library and add the Project8.1 I2C LCD1602\ LiquidCrystal_I2C.zip from the Libraries folder.



When these libraries are addeded, we can find them under Sketchbook location in the File-Preference window. We can view the source code of these library files and learn their specific usage.



Now let 's start to write code to use LCD1602 to display static characters and dynamic variables.

Sketch 8.1.1 I2C_LCD1602

```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2); // set the LCD address to 0x27 for a 16 chars and 2
line display
int counter = 0;
void setup()
{
  lcd.init();                    // initialize the lcd
  lcd.backlight();               // Turn on backlight
  startingAnimation();           // subfunction of Boot animation
}

```

```
12  void loop()
13  {
14    lcd.setCursor(0, 1);  // Set the cursor to the first column and the second row
15    lcd.print(counter);  // Print variable
16    delay(1000);
17    counter++;          // counter=counter+1
18  }
19  void startingAnimation() {
20    for (int i = 0; i < 16; i++) {
21      lcd.scrollDisplayRight();   // shift the screen right 1 character
22    }
23    lcd.setCursor(0, 0);     // Set the cursor to the end of the first row.
24    lcd.print("Freenove");
25    for (int i = 0; i < 16; i++) {
26      lcd.scrollDisplayLeft();  // shift the screen left 1 character
27      delay(300);
28    }
29  }
```

Following are the I2C Wire library and LiquidCrystal_I2C library used for controlling LCD:

```
    #include <Wire.h>
    #include <LiquidCrystal_I2C.h>
```

LiquidCrystal_I2C library provides LiquidCrystal_I2C class that controls LCD1602. When we instantiate a LiquidCrystal_I2C object, we can input some parameters. And these parameters are the row/column numbers of the I2C addresses and screen that connect to LCD1602：

```
    LiquidCrystal_I2C lcd(0x27, 16, 2); // set the LCD address to 0x27 for a 16 chars and
    2 line display
```

First, initialize the LCD and turn on LCD backlight.

```
      lcd.init();                      // initialize the lcd
      lcd.backlight();
```

Then display a boot animation, making a string of characters scroll from the right to the left. First, shifta blank screen right 16 characters, so (0,0) is moved out of the screen the. Then write strings d (0,0) (at this time, it is outside the screen, so it is not visible), and then shift the screen left 16 characters.

```
    void startingAnimation() {
      for (int i = 0; i < 16; i++) {
        lcd.scrollDisplayRight();
      }
      lcd.setCursor(0, 0);     // Set the cursor to end of the first line.
      lcd.print("Freenove");
      for (int i = 0; i < 16; i++) {
        lcd.scrollDisplayLeft();
        delay(300);
      }
    }
```

Finally, in the loop (), the variable will be printed once every second once every one second at first column,

second row.

```
    lcd.setCursor(0, 1);
    lcd.print(counter);
    delay(1000);
    counter++;
```

LiquidCrystal_I2C Class

LiquidCrystal_I2C class can control common LCD screen. First, we need instantiate an object of LiquidCrystal_I2C type, for example:

```
LiquidCrystal_I2C lcd(0x27, 16, 2);
```

When an object is instantiated, a constructed function of the class is called a constructor. In the constructor function, we need to fill in the I2C address of the LCD module, as well as the number of columns and rows of the LCD module. The number of columns and rows can also be set in the lcd.begin ().

The functions used in the LiquidCrystal_I2C class are as follows:

Lcd.setCursor (col, row): set the coordinates of the to-be-printed character. The parameters are the numbers of columns and rows of the characters (start from 0, the number 0 represents first row or first line).

Lcd.print (data): print characters. Characters will be printed on the coordinates set before. If you do not set the coordinates, the string will be printed behind the last printed character.

Verify and upload the code, then observe the LCD screen. If the display is not clear or there is no display, adjust the potentiometer on the back of I2C module to adjust the screen contrast until the character is clearly displayed on the LCD.

You can use the LCD I2C to replace the serial port as a mobile screen when you print the data latter,.

Chapter 9 IR Remote

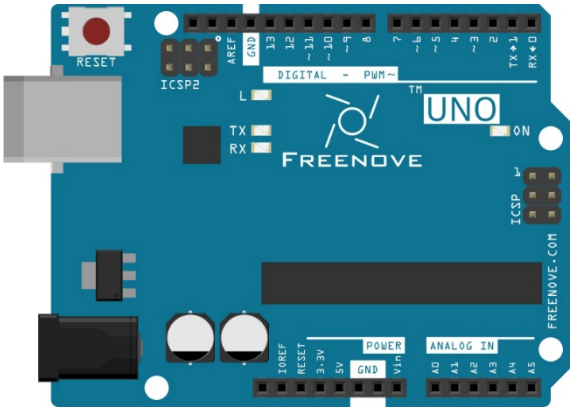
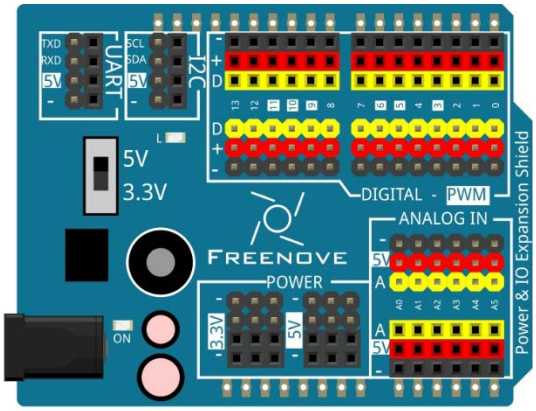
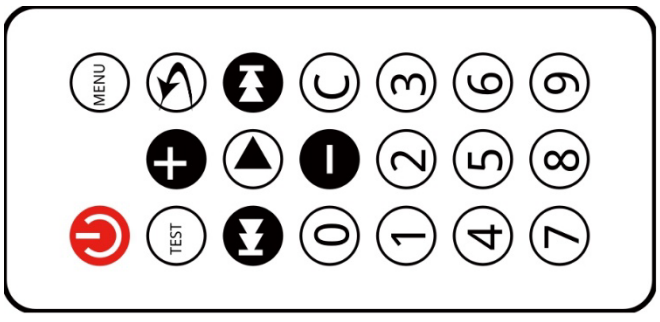
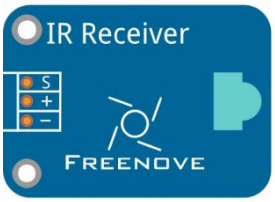
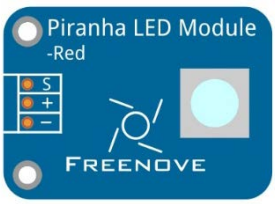
In this chapter, we will learn IR Remote, which is a remote control commonly used in house.

Project 9.1 IR Remote

IR Remote is a very common remote control, for example, the remote control of the TV and air conditioner are IR Remote. Through this experiment, we will learn how to receive the signal of IR Remote and use it to control some devices.

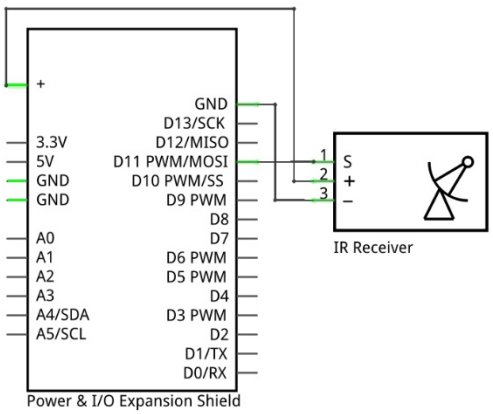
First, we will use serial port to display the received data and see how it works. Next, we make a lamp controlled by IR Remote. Then you can use the IR Remote as a remote switch and control any related equipments as you want.

Component List

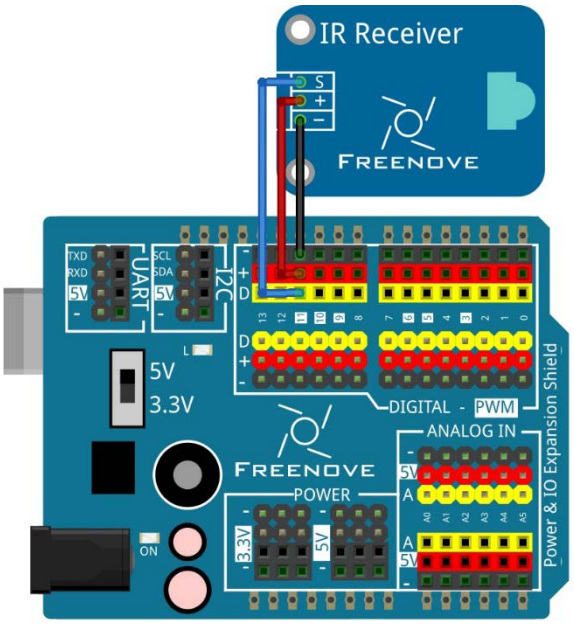
<table>
<tr><td>Freenove UNO x1
  </td><td colspan="2">Freenove Power & I/O Expansion Shieldx1
  </td></tr>
<tr><td>IR Remote Controller x1
  </td><td>IR Receiver x1
  </td><td>Piranha LED Module x1
  </td></tr>
<tr><td>USB cable x1
  </td><td colspan="2">Jumper M/F x6
  </td></tr>
</table>

Circuit

Schematic diagram



Circuit connection



Sketch

Before writing the code, we need to import the corresponding library file IRremote.zip for Receiver Module IR.

Sketch 9.1.1 IRremote_Recv

```
#include <IRremote.h> // Include library file
int RECV_PIN = 11;// Infrared receiving pin
IRrecv irrecv(RECV_PIN);// Create a class object used to receive class
decode_results results;// Create a decoding results class object
void setup()
{
  Serial.begin(9600);// Configurate serial port
  irrecv.enableIRIn(); // Start the receiver
}
void loop() {
  //Waiting for decoding
  if (irrecv.decode(&results)) {
    Serial.println(results.value, HEX);// Print out the decoded results
    irrecv.resume(); // Receive the next value
  }
  delay(100);
}
```

First, include the header file of library.

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

Then define the signal pin of IR Receiver Module and instantiate two objects for the IRrecv (received class) and decode_results (decoded result class) respectively.

```
int RECV_PIN = 11;
IRrecv irrecv(RECV_PIN);
decode_results results;
```

Configurate the serial port in setup () and open the infrared receiver. Finally, determine whether the decoding succeeds in the loop. If it is successful, print out the decoded results in the form of sixteen decimal (HEX).

```
  if (irrecv.decode(&results)) {
    Serial.println(results.value, HEX);
    irrecv.resume(); // Receive the next value
  }
```

IRrecv Class

This is an infrared remote control receiving class. We need instantiate a class object before use. And it has two constructor functions:

IRrecv (recvpin int) – the parameter is the constructor function of infrared receiving pin

IRrecv (recvpin int, blinkpin int) –the parameters are the constructor functions of infrared receiving pin and the pin of the indicator light

The followings are two member functions:

int decode (*results decode_results) - the decoding function, The results are stored in the decode_results class objects.

void enableIRIn () - receive enable function. Start the infrared receiver after it is called

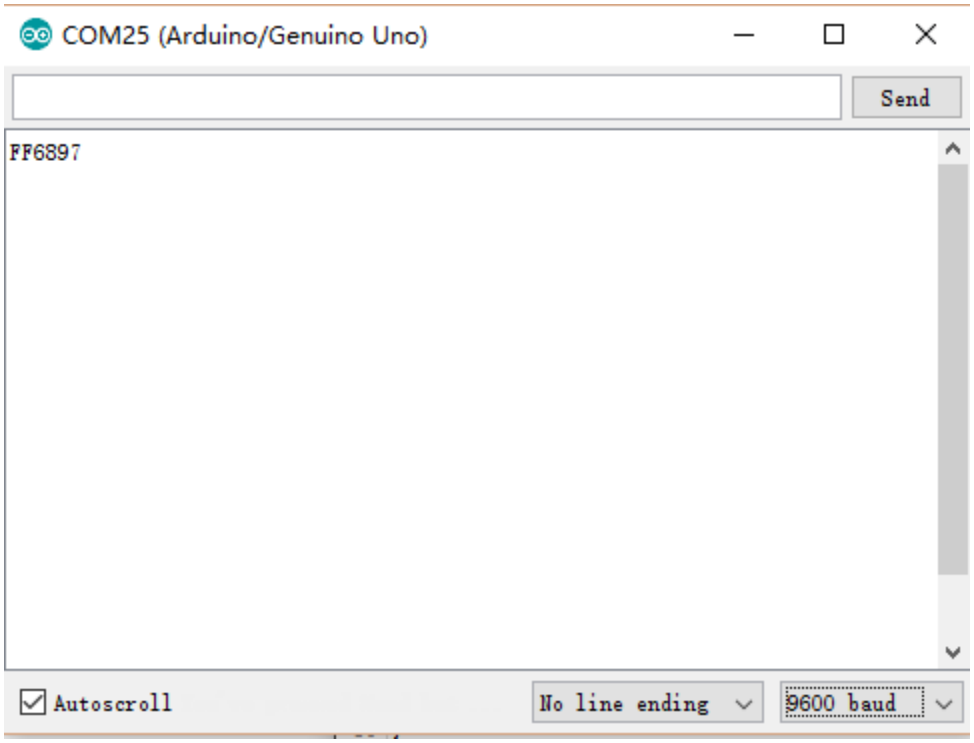
decode_results Class

This is one class of member variables. Save the decoding results and other related information. The following is some member variables.

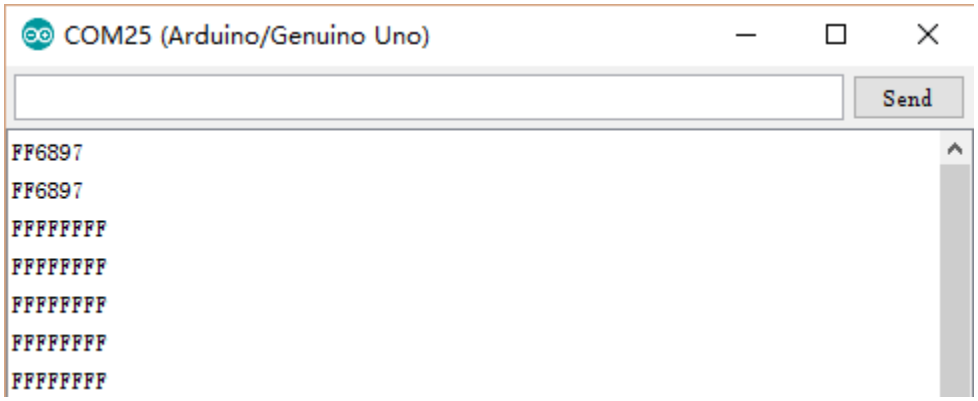
```
unsigned int    address;   // Used by Panasonic & Sharp [16-bits]
unsigned long   value;     // Decoded value [max 32-bits]
  int           bits;      // Number of bits in decoded value
```

For more details about IRremote library, please refer to IRremote.h.

Verify and upload the code to the UNO. Open the serial port monitor and make MINI infrared remote control towards the IR receiver module receiver. Then, when you press any key, serial port will print out the decoding result. And each key-pressing corresponds to different code. For example, if the "0" on the remote control is pressed, the decoding results will be: FF6897.



If you keep pressing the "0", it means to repeat the "0", and the code will be FFFFFFFF.



If the remote control is not placed towards the IR receiver module receiver, or there are obstacles or interference, the decoding will not be successful. The correct decoding result should be a 6 digits ' hexadecimal number, which is in the same with two former number of a remote control key codes.

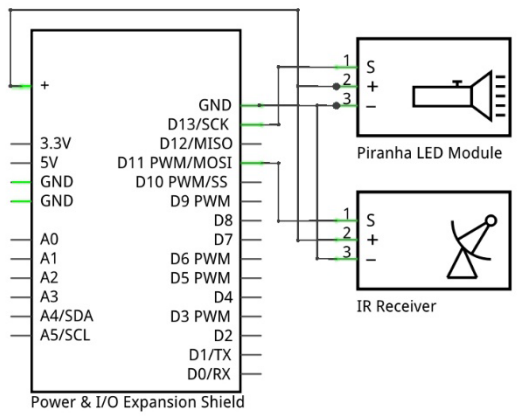
After learning the use of infrared remote control, let ' s began to make a remote control LED.

Project 9.2 Remote control lamp

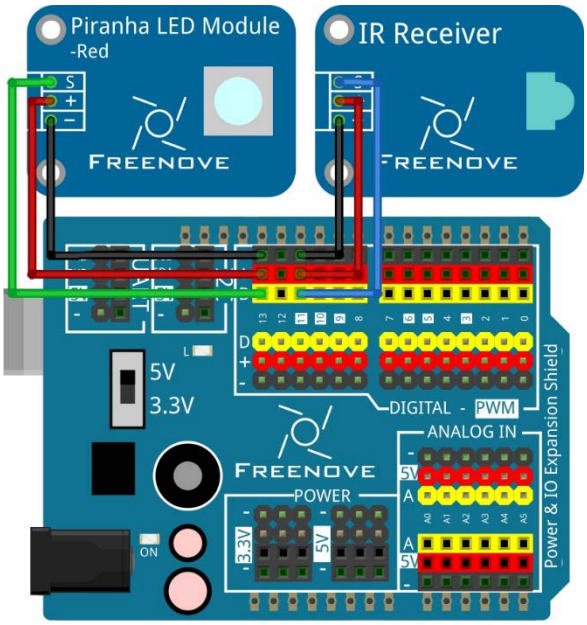
The components list is the same with the last section. We just need add a Piranha LED Module in the circuit above.

Circuit

Schematic diagram



Circuit connection



Sketch

Write the code: if the "0" on the remote control is pressed, the LED will be turned off or on.

Sketch 9.2.1 IRremote_LED

```
#include <IRremote.h> // include library file
int RECV_PIN = 11;// define infrared receiving pin
int ledPin = 13;// define LED pin
boolean ledState = false;// define the LED state
IRrecv irrecv(RECV_PIN);// Create a object for receiver class
decode_results results;// Create a object for decoding result class
void setup()
{
  pinMode(ledPin, OUTPUT);// set ledPin to output mode
  digitalWrite(ledPin, ledState);// update the original LED sate
  Serial.begin(9600);// Configure serial port
  irrecv.enableIRIn(); // Start the receiver
}

void loop() {
  //wait for decoding
  if (irrecv.decode(&results)) {
    Serial.println(results.value, HEX);// print out decoding result
    if (results.value == 0xFF6897) {// if "0"is pressed, chang the LED state.
      ledState = !ledState;//reverse the state
```

```
21          digitalWrite(ledPin, ledState);//update the LED state
22        }
23        irrecv.resume(); // Receive the next value
24      }
25      delay(100);
26  }
```

In the code, the configuration process of decoding is the same with the last code. We only need judge if decoding result is FF6897 after the decoding is successful. If it is, it indicates that the key "0" is pressed. Then we reverse the LED state. If the original sate is on, the new state is off.

```
        if (results.value == 0xFF6897) {// if "0"is pressed, chang the LED state.
          ledState = !ledState;// reverse the state
          digitalWrite(ledPin, ledState);// update the LED state
        }
```

Verify and upload the code to UNO. Then make the MINI infrared remote control towards the IR Receiver Module receiver. If key "0" is pressed, LED will be turned off or on.

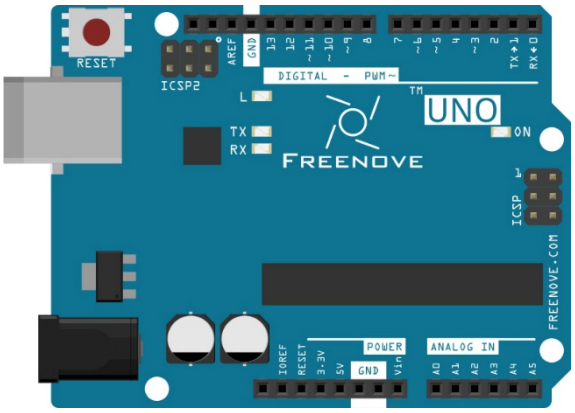
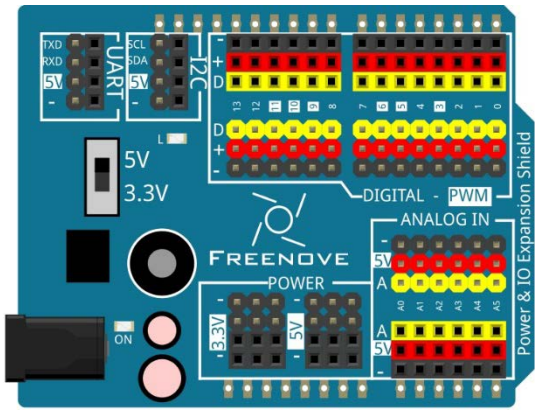
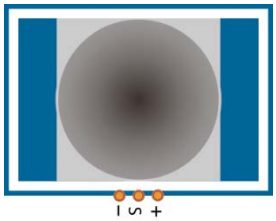
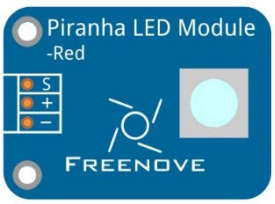
Chapter 10 Infrared Motion Sensor

In this chapter, we will learn a widely used sensor, Infrared Motion Sensor.

Project 10.1 Sense LED

In this project, we will make a sense LED, with the human body infrared pyroelectric sensors.
When someone get close to the LED, it will light automatically. On the contrary, it will be out.
This infrared motion sensor is a kind of sensor which can detect the infrared emitted by human and animals.

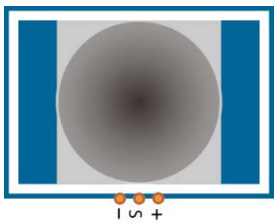
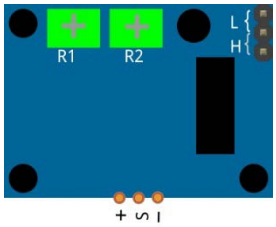
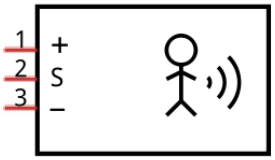
Component List

<table>
<tr><td>Freenove UNO x1
  </td><td>面包板(bread board)x1
  </td><td></td></tr>
<tr><td>USB cable x1
  </td><td>HC SR501 x1
  </td><td>Piranha LED Module x1
  </td></tr>
<tr><td>Jumper M/F x6
  </td><td></td><td></td></tr>
</table>

Component Knowledge

Component knowledge

The following is the diagram of infrared Motion sensor（HC SR-501）:

Top	Bottom	Schematic
		 Infrared Motion Sensor(HC SR501)

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 canbe is adjusted by potentiometer R1.
3. According to the position of jumper cap, you can choose non-repeatable trigger mode or repeatable mode.

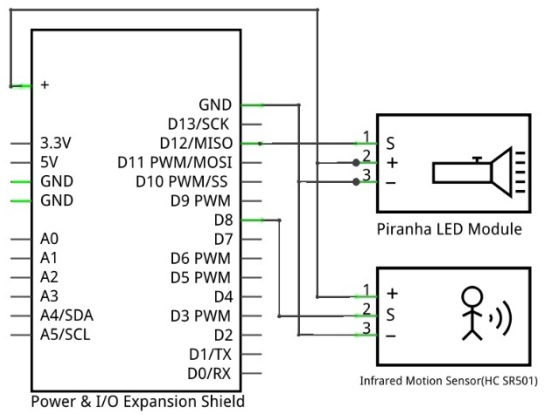
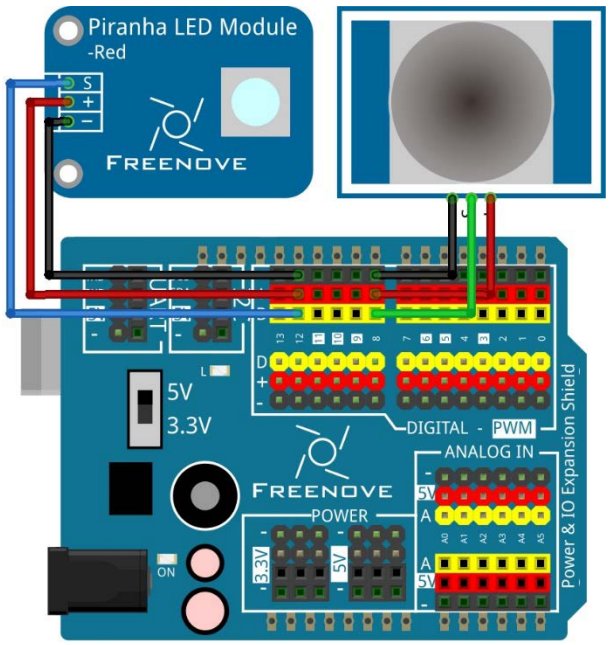
L: non-repeatable trigger mode. The module output high level after sensing body, then when delay time is over, the module will output low level. And during high level time, the sensor don't sense body anymore.

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.

We can regard this sensor as a simple inductive switch when in use.

Circuit

Schematic diagram	Circuit connection
	

Sketch

This code is the same as the one for key control switch because we can take infrared motion sensor as a switch. When someone gets close to the sensor, the model will output high level. And LED will be turned on when pin in high level is detected by UNO. Otherwise, the LED will be turned off.

Sketch 10.1.1 sense_Lamp

```
int ledPin = 12;// define pin for LED
int sensorPin = 8; // define input pin for the sensor
void setup() {
  pinMode(sensorPin, INPUT); // define input and output
  pinMode(ledPin, OUTPUT);
}
void loop() {
  if (digitalRead(sensorPin))// if sensorPin is in high level, turn LED on.
    digitalWrite(ledPin, HIGH);
  else//     turn off LED
    digitalWrite(ledPin, LOW);
}
```

Set sensor pin as input and LED pin as output in the code. When sensor pin in high level, turn LED on, otherwise LED off.

Verify and upload the code to UNO. Wait for initialization of sensor. Then try to get close to and away from the sensor, and observe whether the LED will be turned on or turned off automatically.

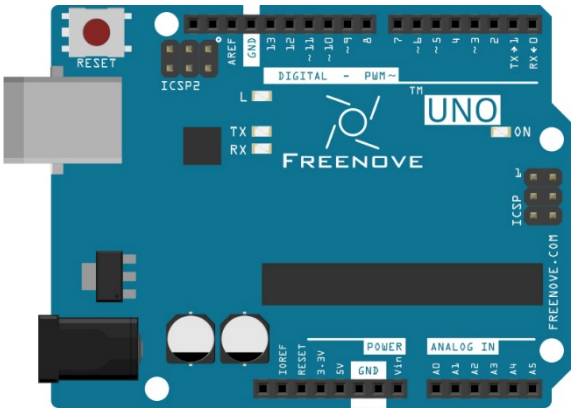
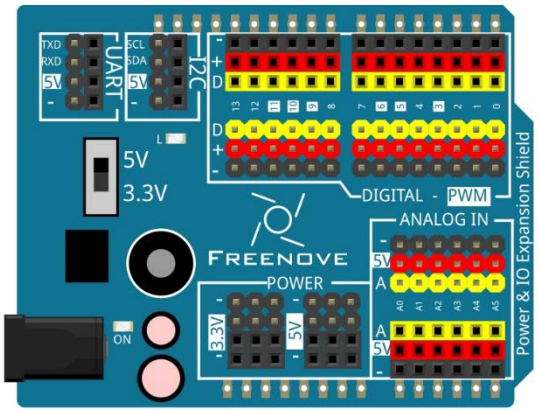
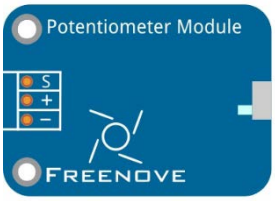
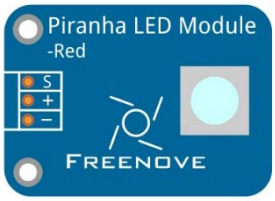
Chapter 11 Sound Sensor

In this chapter, we will learn a sound-sensing sensor, Sound Sensor.

Project 11.1 Sound Control Lamp

We have done a sense light, and in this project we will use sound sensor to do a sound control lamp.

Component List

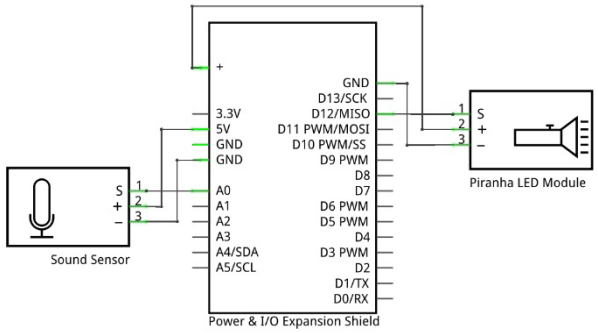
Freenove UNO x1 	Freenove Power & I/O Expansion Shieldx1 
USB cable x1 	Sound Sensor x1 
Jumper M/F x6 	Piranha LED Module x1 

Component Knowledge

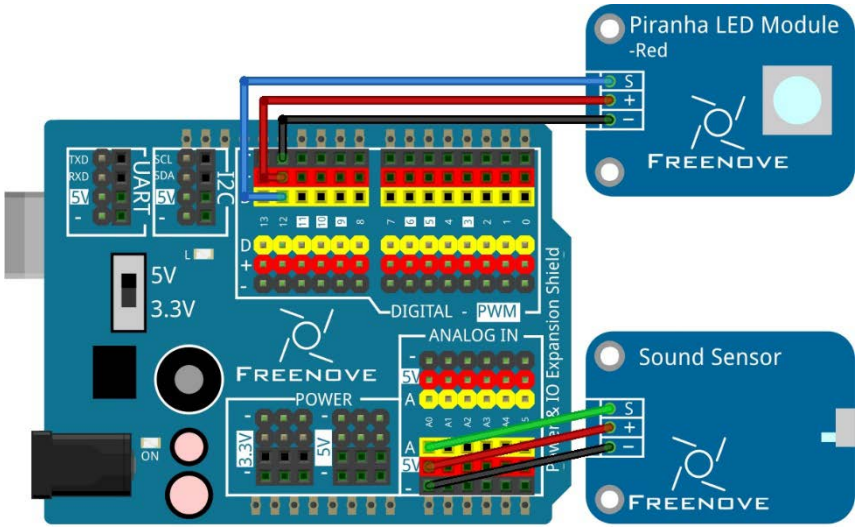
Sound sensor is a kind of sensor which can sense sound around and output analog signal.
In the static state and powered by 5v, the output voltage of sensor is 2.5V. With changing the loudness of the sound, the output voltage changes in range from 0V to 3.5V. The conversion digital of voltage read by UNO is in range from 0 to 716. You can use serial port to print it to check out.

Circuit

Schematic diagram



Circuit connection



Sketch

The sound is created by the vibration, so the output wave form of the Sound Sensor is not continuous high level or low level, but vibrational curve with peaks and valleys. And the louder the sound, the higher the peaks, and the higher or lower the output voltage. Therefore, when the sound we collect is loud enough, turn LED on lasting for durationTime, just like infrared motion sensor. During this time, if the sound is still loud enough, then we should turn LED on continuously. Otherwise, turn LED off after the durationtime. Next, let 's write codes according to these.

Sketch 11.1.1 sound_Lamp

```
1  #define sensorPin A0// define input pin for sensor
2  int ledPin = 12;// define pin for LED
3  unsigned long updateTime;// Define a variable to save the time point for collection of
4  sound
5  unsigned long durationTime = 3000;//define the duration after collection of sound
6
7  void setup() {
```

```
8     pinMode(ledPin, OUTPUT);
9   }
10  void loop() {
11    if (analogRead(sensorPin) > 600) {// If the sound is loud enough, update the
12  collection time point, and turn LED on.
13      updateTime = millis();
14      digitalWrite(ledPin, HIGH);
15    }
16    if ((millis() - updateTime) > durationTime) {//if the duration is over, turn LED off.
17      digitalWrite(ledPin, LOW);
18    }
19  }
```

Verify and upload the code to UNO. The LED will be turned if you speak or clap toward the loudspeaker. If there is no sound after a while, the LED will be turned off.

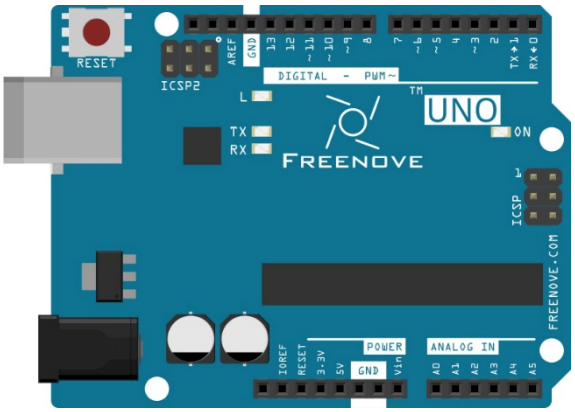
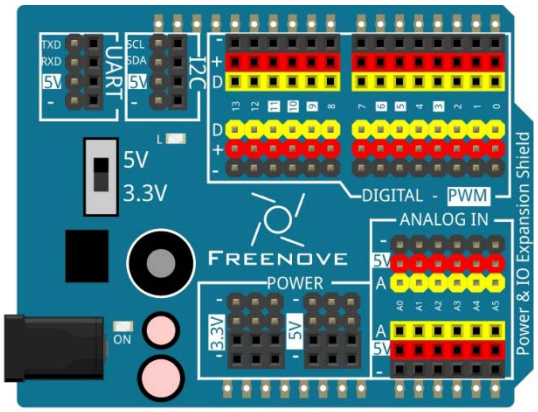
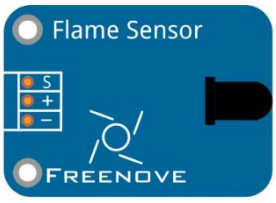
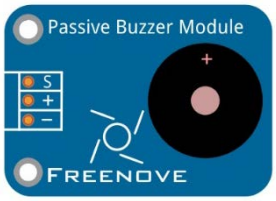
Chapter 12 Flame Sensor

In this chapter, we will learn a sensor that can detects the flame, Sensor Flame.

Project 12.1 Fire source monitor

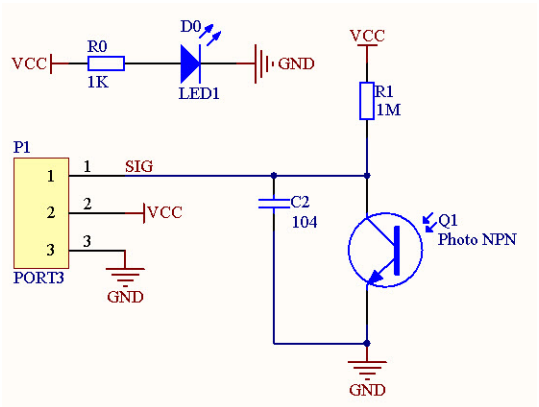
In this project, we will make a fire source monitor. When the fire source is detected, the buzzer will alarm.

Component List

<table>
<tr><td>Freenove UNO x1
  </td><td colspan="2">Freenove Power & I/O Expansion Shieldx1
  </td></tr>
<tr><td>USB cable x1
  </td><td rowspan="2">Flame Sensor x1
  </td><td rowspan="2">Passive Buzzer Module x1
  </td></tr>
<tr><td>Jumper M/F x6
  </td></tr>
</table>

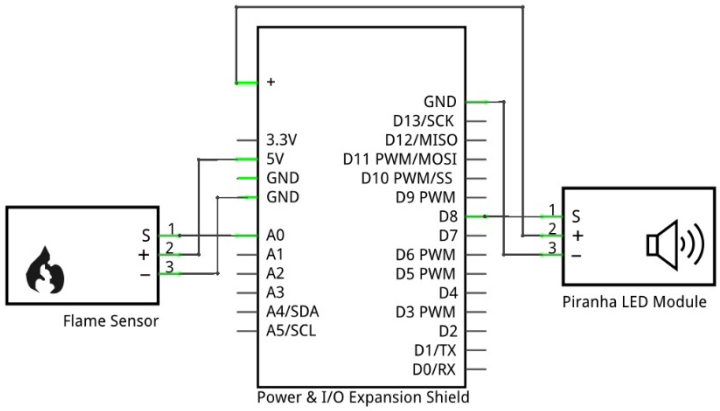
Component Knowledge

The circuit diagram of Flame Sensor is shown below:

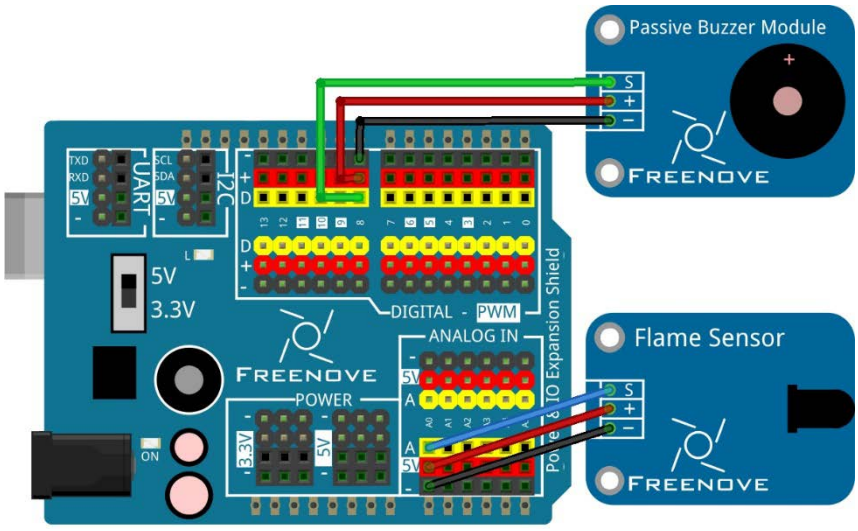
Circuit diagram	Analysis
	According to the circuit diagram, we can know that when the infrared (wavelength 940nm) emitted by the flame is received by the sensor, the infrared transistor Q1 turns on, then the voltage of output pin SIG is pulled down. The higher the flame intensity is, the more close to 0 the voltage of SIG pin will be.

Circuit

Schematic diagram



Circuit connection



Sketch

Write code to detect the ADC value of the flame sensor, and when the flame is detected, the ADC value will be reduced, then the buzzer alarms.

Sketch 12.1.1 flame_ detection

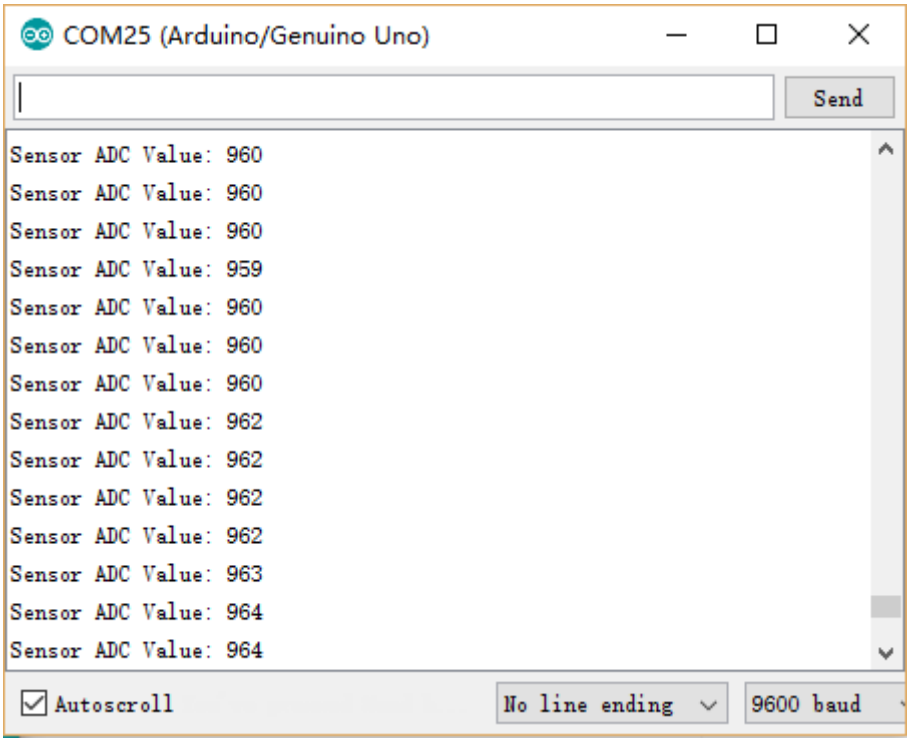
```
#define sensorPin A0 // define the sensor pin
int buzzerPin = 8;// define the buzzer pin
int flameThreshold = 500;// Define a threshold, when the sensor data is below it, the buzzer alarms.
void setup() {
  pinMode(buzzerPin, OUTPUT); // set buzzer pin to output mode
  Serial.begin(9600);//configure serial port
}

void loop() {
  int reading = analogRead(sensorPin);// read voltage of sensor
  Serial.print("Sensor ADC Value: ");// print ADC value of the sensor
  Serial.println(reading);
  if (reading < flameThreshold) { // if thesensor detects theflame,  the buzzer will alarm。
    alert(); //alarm
  }
  else
    noTone(buzzerPin);// stop alarming
}
void alert() {
  float sinVal;         // Define a variable to save the sine value
  int toneVal;          // Define a variable to save the sound frequency
  for (int x = 0; x < 360; x++) {       // xfrom 0° to 360°
    sinVal = sin(x * (PI / 180));       // calculat sine value of x
    toneVal = 2000 + sinVal * 500;      // calculate the sound frequency according to the sine value of X
    tone(buzzerPin, toneVal);           // output sound frequency to buzzerPin
    delay(1);
  }
}
```

In the code, read the sensor ADC value to determine whether the value is less than the set value flameThreshold constantly. If it is less than the value, it indicates that the flame is detected. Then call the alarm function to alarm. At the same time, use the serial port to print out the data to facilitate observation of sensor data, and set a more appropriate threshold according to the data (the threshold used in different environment may be different).

Verify and upload the code to the UNO. Trigger the lighter (attention to safety), and move the lighter close

to the receiver of the flame sensor. Then the buzzer began to alarm. Observe the data in the serial monitor.



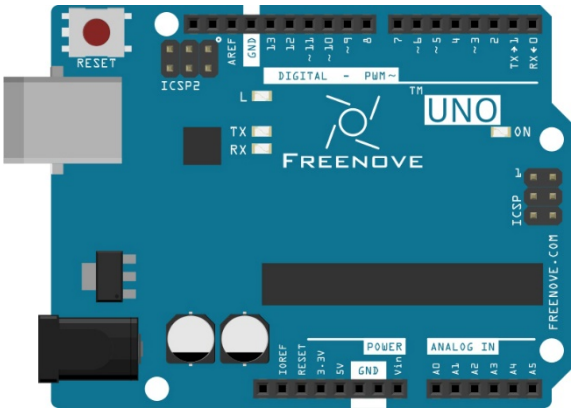
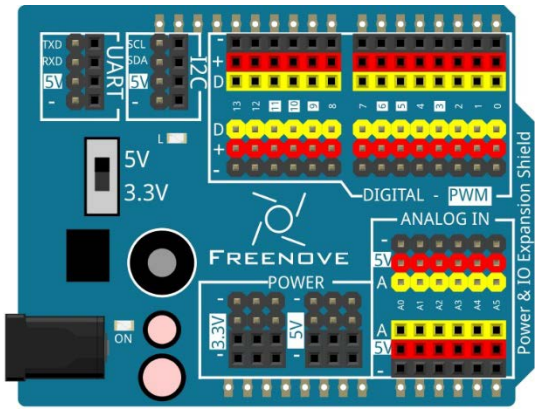
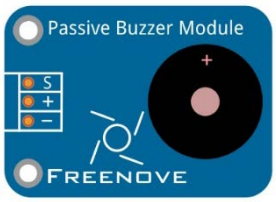
Chapter 13 Combustible Gas Sensor

We have introduced the Flame Sensor which can detect the flame. In this chapter, we will introduce a sensor that can detect the combustible gas, Sensor MQ-2 - Gas.

Project 13.1 Gas monitor

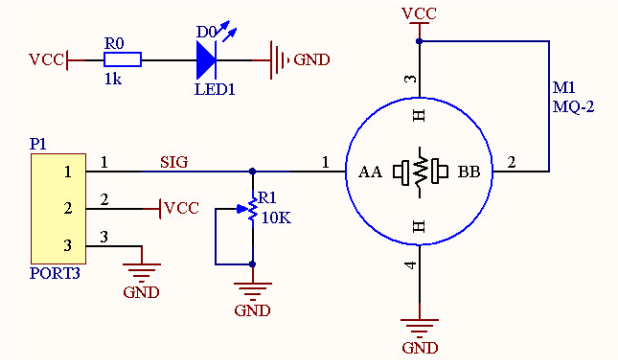
In this project, we will use combustible gas sensor MQ-2 to make a gas monitoring device.

Component List

<table>
<tr><td>Freenove UNO x1
  </td><td colspan="2">Freenove Power & I/O Expansion Shieldx1
  </td></tr>
<tr><td>USB cable x1
  </td><td>Gas Sensor x1
  </td><td>Passive Buzzer Module x1
  </td></tr>
<tr><td>Jumper M/F x6
  </td><td colspan="2"></td></tr>
</table>

Component Knowledge

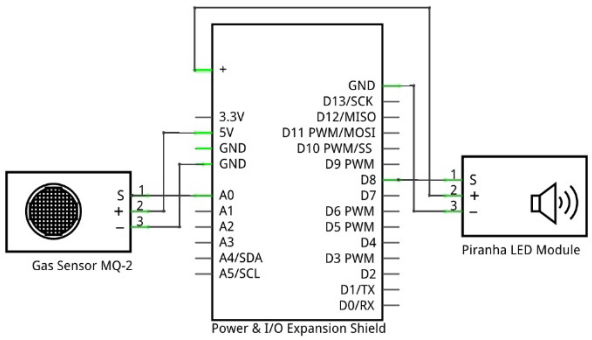
MQ-2 is a gas sensor that can be used to monitor the gas leakage of home and factory, which is suitable for the detection of liquefied gas, propane, methane, alcohol, hydrogen, smoke and so on. The circuit diagram of Sensor Gas is shown below:

Circuit diagram	Analysis：
	SIG is signal output pin for analog signal. The denser the detected gas is, the higher the output voltage. Potentiometer R1 is used to adjust output sensitivity and output voltage range of SIG pin. In addition, the oxygen content in the air will also affect the output of the sensor.

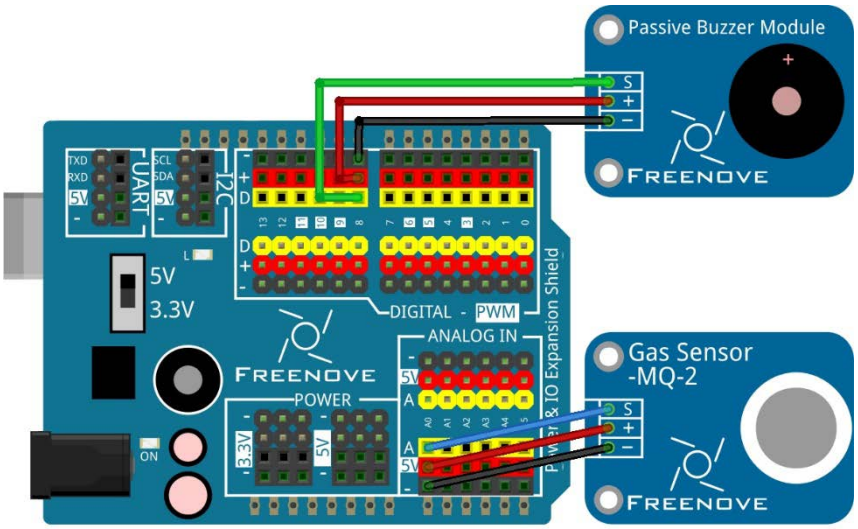
Circuit

The circuit is similar to the last chapter. We just need replace Flame Sensor with Gas Sensor.

Schematic diagram



Circuit connection



Sketch

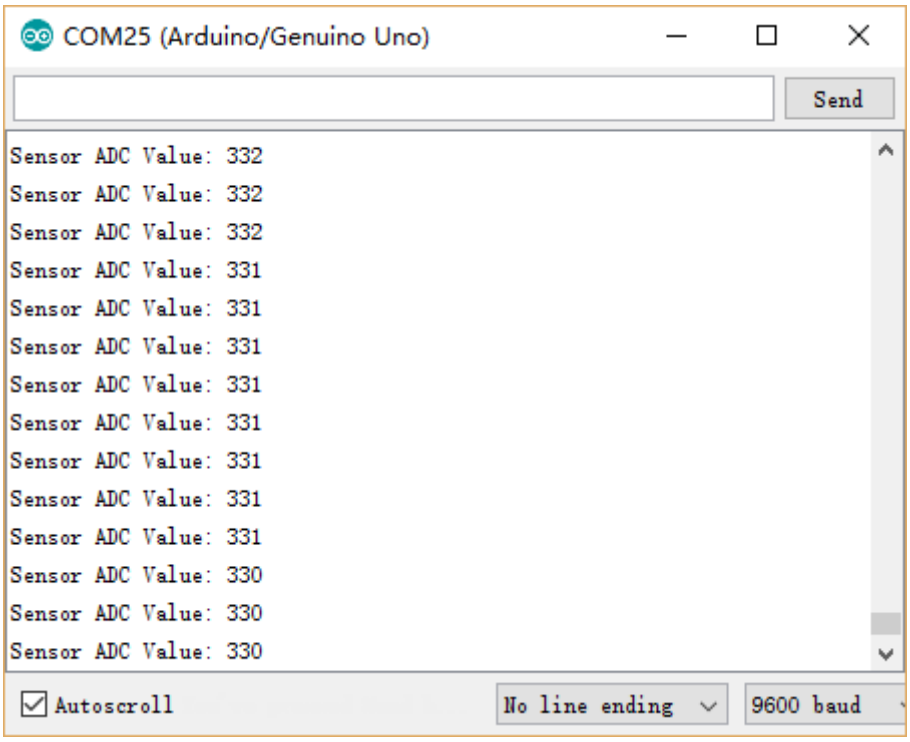
The code is the same with the fire monitor in logic. What 's different is that when the flame sensor detects the flame, the output voltage decreases, but when gas sensor detects combustible gas, the output voltage increases.

Sketch 13.1.1 gas_ detection

```
#define sensorPin A0 //define the pin for sensor
int buzzerPin = 8;// define the pin for buzzer
int gasThreshold = 700;// Define a threshold. When the sensor data is higher than it,
buzzer starts alarm.
void setup() {
  pinMode(buzzerPin, OUTPUT); // set buzzer pin to output mode
  Serial.begin(9600);//Configure the serial port
}

void loop() {
  int reading = analogRead(sensorPin);// read the voltage of sensor
  Serial.print("Sensor ADC Value: ");// print ADC value of sensor
  Serial.println(reading);
  if (reading > gasThreshold) { // If the gas detected is too dense, the buzzer starts
salarms.
    alert(); //alarm
  }
  else
    noTone(buzzerPin);//stop alarming
}
void alert() {
  float sinVal;         //define a variable to save the sine value
  int toneVal;          //define a variable to save the sound frequency
  for (int x = 0; x < 360; x++) {       // x is from 0° to 360°
    sinVal = sin(x * (PI / 180));       //calculat sine value of x
    toneVal = 2000 + sinVal * 500;      //calculat sound frequency according to sine
value of x
    tone(buzzerPin, toneVal);       //buzzerPin output sound frequency to buzzerPin
    delay(1);
  }
}
```

Verify and upload the code to UNO. Use lighter to release gas near the Gas Sensor (attention to safety), then the buzzer will alarm. Observe the data chang in the serial monitor window.



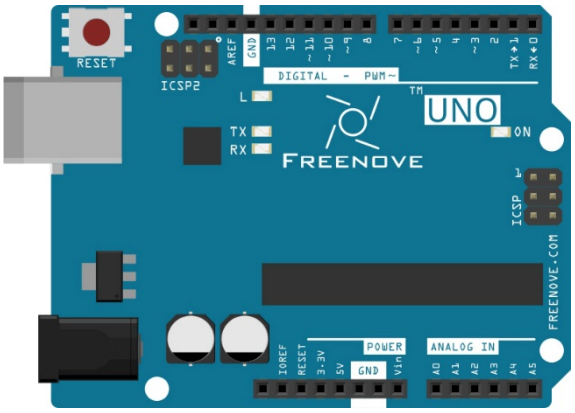
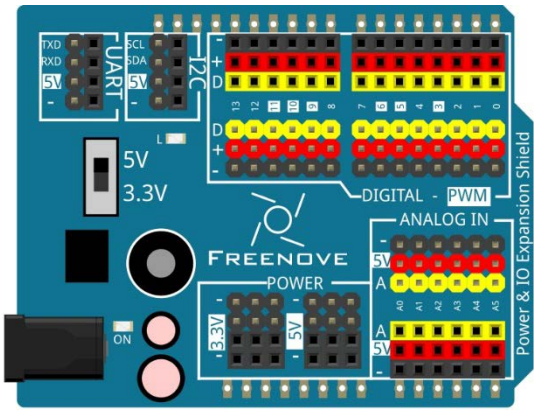
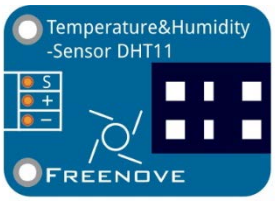
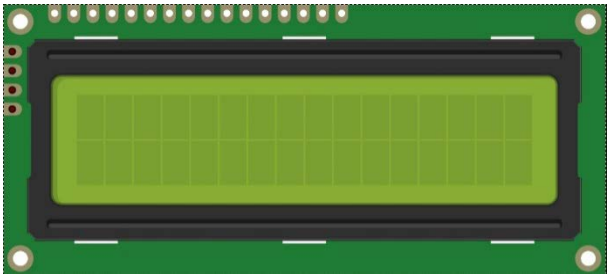
Chapter 14 Temperature & Humidity Tester

In this chapter, we will learn a commonly used sensor, Temperature & Humidity Sensor.

Project 14.1 LCD1602 And DHT11

Temperature & Humidity Sensor is important for our life, which reminds us of warming and water replenishment. In this experiment, we will read the temperature and humidity of the sensor, display the data through LCD1602.

Component List

Freenove UNO x1 	Freenove Power & I/O Expansion Shieldx1 
USB cable x1 	Jumper M/F x7 
Temperature & Humidity Sensor x1 	I2CLCD1602 Module x1 

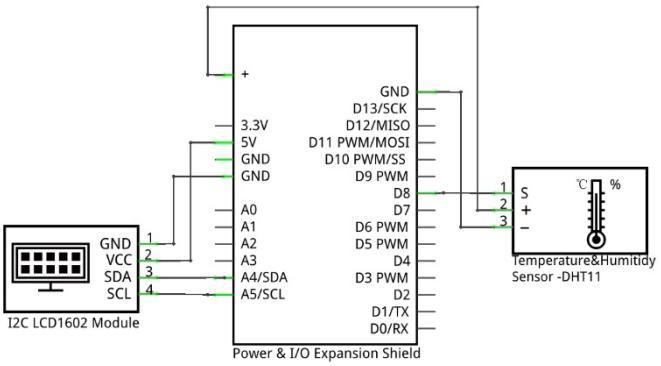
Component Knowledge

Temperature & Humidity Sensor DHT11 is a compound temperature & humidity sensor, and the output digital signal has been calibrated inside.

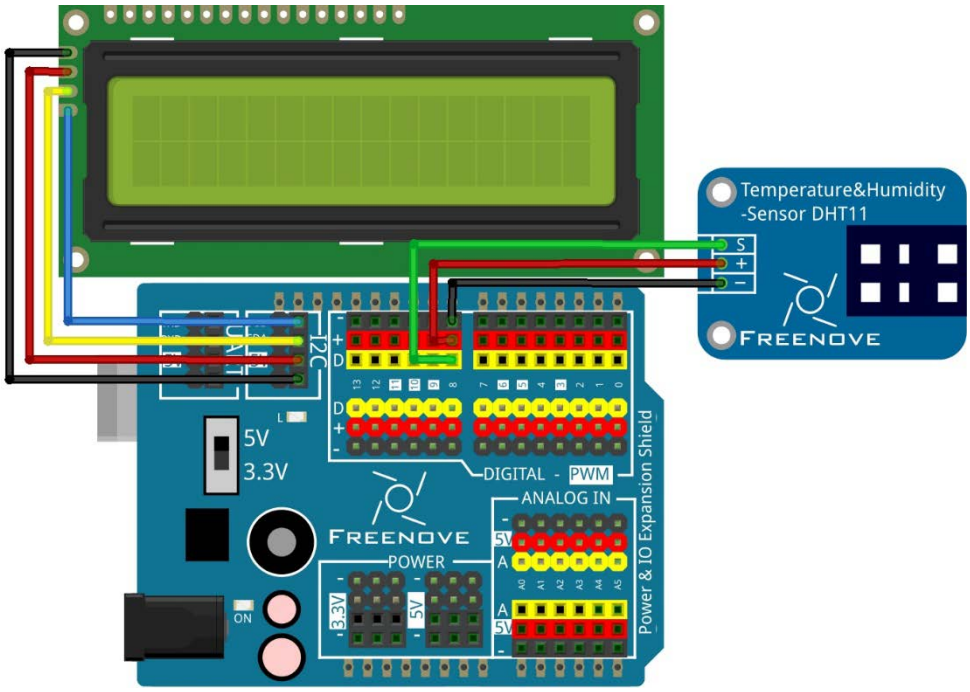
It has 1S's initialization time after powered up. When using DHT11 to read the Arduino, it is convenient to read the temperature and humidity data by using DHT library files.

Circuit

Schematic diagram



Circuit connection



Sketch

First, you should import the library file DHT.zip for DHT11 sensor.

When you write code, include the header files of DHT and LCD1602. Use DHT class object, get the relevant parameters of the DHT11 sensor, and display the parameters on the LCD1602 screen.

Sketch 14.1.1 hygrothermograph

```
1  #include <LiquidCrystal_I2C.h> // include header file of I2C LCD1602
2  #include <dht.h> // include DHT library file
```

```
dht DHT;// create DHT object
#define DHT11_PIN 8 // define pin for sensor
LiquidCrystal_I2C lcd(0x27, 16, 2); // set the LCD address to 0x27 for a 16 chars and 2
line display
void setup()
{
  lcd.init();  // initialize LCD
  lcd.backlight();//Turn on LCD backlight
}
void loop()
{
  int chk = DHT.read11(DHT11_PIN); // read DHT11 and judge the state according to the
return value
  lcd.setCursor(0, 0); // Set cursor (0,0)
  lcd.clear();         // clear all contents on LCD
  switch (chk)
  {
    case DHTLIB_OK:// if DHT state is OK, print temperature and humidity data
      lcd.print("H:");
      lcd.print(DHT.humidity);// display humidity data
      lcd.print("%");
      lcd.setCursor(0, 1);
      lcd.print("T:");
      lcd.print(DHT.temperature);// display temperature data
      lcd.print("C");
      lcd.setCursor(14, 0);
      lcd.print("OK");
      break;
    case DHTLIB_ERROR_CHECKSUM:// data verification and error
      lcd.print("Checksum error");
      break;
    case DHTLIB_ERROR_TIMEOUT:// connection timeouts
      lcd.print("Time out error");
      break;
    default:// other erro
      lcd.print("Unknown error");
      break;
  }
  delay(1000);
}
```

In the code, the first line includes two header files, and then create dht and LiquidCrystal_I2C class objects to use these two devices. Use DHT.read11()to read the DHT11 information, judge whether the normal sensor according to the return value of the function. If the data is normal, display the data on the screen, otherwise

display the error type.

```
int chk = DHT.read11(DHT11_PIN); //read the DHT11, judge whether the state according to the return value
......
switch (chk)
  {
    case DHTLIB_OK:// if DHT state is OK, print temperature and humidity data
      ......
    case DHTLIB_ERROR_CHECKSUM:// data verification and error
      ......
    case DHTLIB_ERROR_TIMEOUT:// connection timeouts
      ......
    default:// other erro
      ......
  }
```

For the return value type and function description, you can find the corresponding meaning in the library file DHT.h

dht class

This is a library for Temperature Humidity & Sensor of DHT type. We need instantiate an object of a DHT class to use it.
dht DHT;
This is one of the member functions:
int read11 (pin uint8_t): read the temperature and humidity data of DHT11.
for more details, please refer to dht.h in the DHT library.

Verify and upload the code to UNO. After the initialization, LCD1602 will display the current humidity and temperature data correctly.

Chapter 15 Analog Hall Sensor

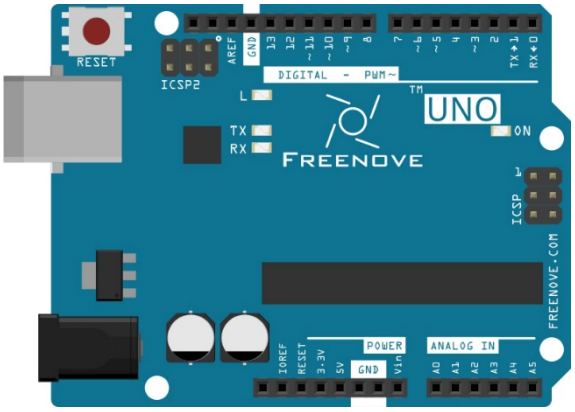
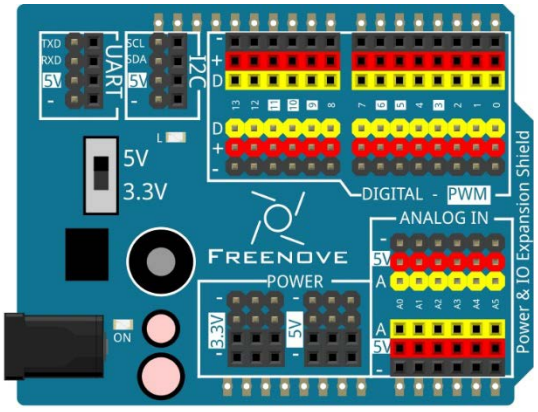
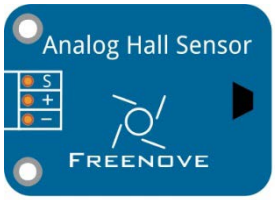
In this chapter, we will learn a sensor that can detect the magnetic field, Analog Hall Sensor.

Project 15.1 tester for Magnetic induction strength

Hall Sensor Analog is a linear-changing sensor according to the magnetic field strengh. Hall sensors are widely used in high precision measurement, industrial automation technology, detection technology and information processing, etc.

In this project, we will use Analog Hall Sensor to measure the magnetic induction intensity of magnetic field, and print it out through the serial port.

Component List

Freenove UNO x1 	Freenove Power & I/O Expansion Shieldx1 	
USB cable x1 	magnet x1 	Analog Hall Sensor x1 
Jumper M/F x3 		

Component Knowledge

S49E used by Analog Hall Sensor is a linear Holzer sensor, whose output voltage is proportional to the intensity of the magnetic field. When the magnetic field intensity is 0, the output voltage is half of power supply. As is shown below, when the S pole of magnet is close to the side with silk screen (or the N pole close to the side without silk screen) of sensor, the output voltage will increase. Otherwise, the output voltage will decrease.

The S pole of magnet is close to the side with silk screen (or the N pole close to the side without silk screen) of S49E. Vout>1/2VCC	The N pole of magnet is close to the side with silk screen (or the S pole is close to the side without silk screen) of S49E. Vout<1/2VCC
	

Technical specs:

Working voltage: 5V, 4.2mA

Sensitivity: 1.8mv/G (1.6~2.0)

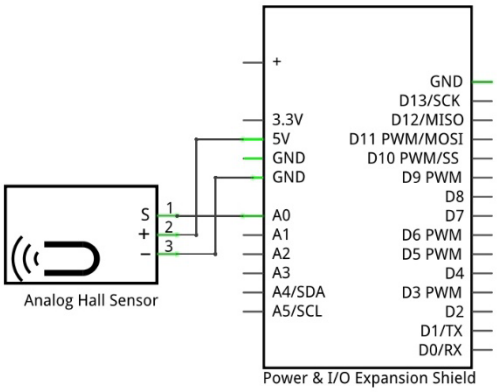
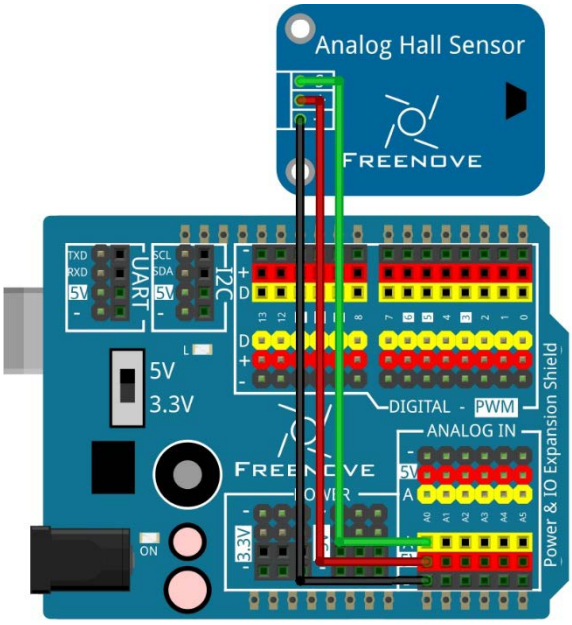
Range: 1500G.

Output range: 0.86V~4.21V

Output voltage(Under static state): 2.5V (2.25V~2.75V).--determinated by temperature and magnetic field under different conditions

For more details, please refer to datasheet.

Circuit

Schematic diagram	Circuit connection
	

Sketch

Because the sensor outputs analog voltage. We need make the sensor connected to the Analog pin and convert analog voltage to the digital value through ADC converter. And calculatethe voltage of the sensor to get the magnetic induction intensity according to the sensitivity.

Sketch 15.1.1 magnetometer

```
#define sensorPin A0 // define the pin for sensor
const float zeroVolt=2.5;// Set the voltage (affected by temperature, magnetic field and other factors) in the static state.
float B, sVolt;// define two variables to save magnetic induction intensity and output voltage
void setup() {
  Serial.begin(9600);
}
void loop() {
  int reading = analogRead(A0);// read sensor voltage and convert it to digital
  sVolt = (float)reading / 1023 * 5;// calculate output voltage of the sensor
  B = (sVolt - zeroVolt) / 1.8 * 1000;// calculate the magnetic induction intensity according to zero-point voltage(when magnetic intensity is zero) and sensitivity
  // display contents above in the serial port
  Serial.print("sVolt:"); Serial.print(sVolt); Serial.print("V\t");
  Serial.print("B:"); Serial.print(B); Serial.print("G\t"); Serial.println();
  delay(100);
}
```

Due to the effects of temperature and magnetic field, the zero-point voltage may be different in different environments. So it is necessary to define the zeroVolt constant to debug. And you need to set the appropriate zero-point according to the output voltage when there is no magnet close to the sensor.

```
const float zeroVolt=2.58;// Set the voltage (affected by temperature, magnetic field and other factors) in the static state.
```

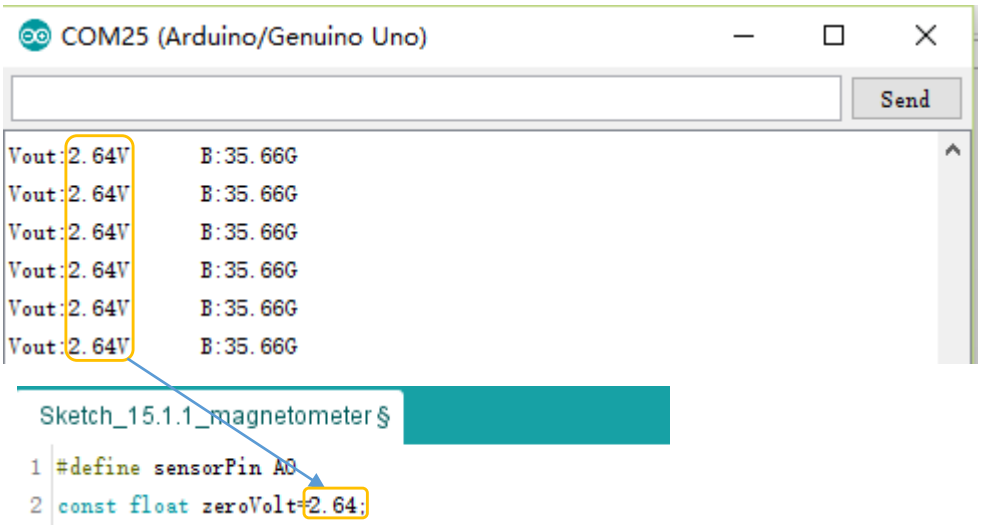
In loop (), first, read the ADC value of the sensor, then converted it to voltage value. The voltage value sVolt in the static state can be used as value of constant zeroVolt.

```
  int reading = analogRead(A0);// read sensor voltage and convert it to digital
  sVolt = (float)reading / 1023 * 5;// calculate output voltage of the sensor
```

Then calculate the magnetic induction intensity according to the sensitivity of 1.8mV/G. And print these information out though serial port.

```
B = (sVolt - zeroVolt) / 1.8 * 1000;// calculate the magnetic induction intensity according to zero-point voltage(when magnetic intensity is zero) and sensitivity
……
```

Verify and upload the code to UNO. Open the serial port monitoring window, and ensure that the sensor around does not magnetic field interference. Then observation sVolt, and assign the value to zeroVolt in the code. Verify and upload the code to UNO agaian.



Open the serial port monitoring window again. Then move the magnet close to the sensor, and observe the changes of the voltage and magnetic field data.

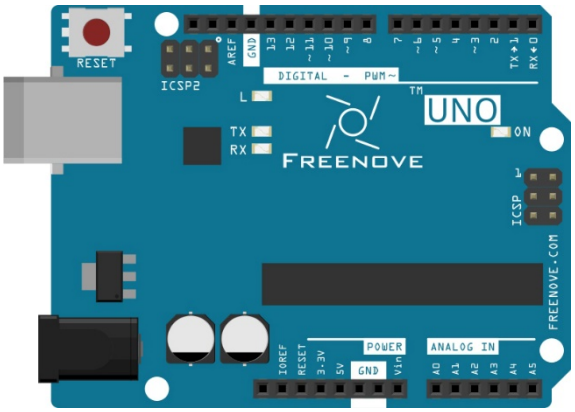
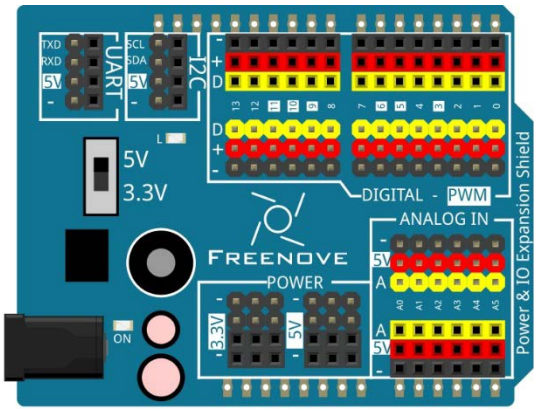
Chapter 16 Ultrasonic Ranging

In this chapter, we learn a module which use ultrasonic to measure distance, HC SR04.

Project 16.1 Ultrasonic Ranging

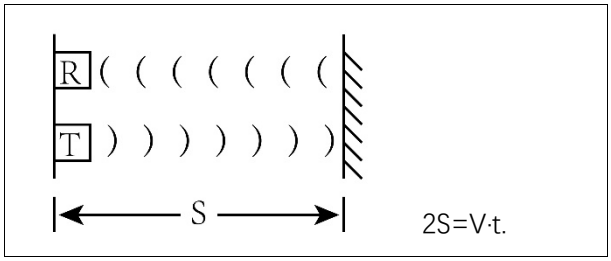
In this project, we use ultrasonic ranging module to measure distance, and print out the data in the serial port (or display it on LCD1602).

Component List

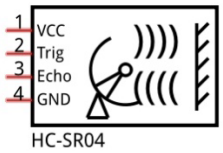
Freenove UNO x1 	Freenove Power & I/O Expansion Shieldx1 
USB cable x1 	Ultrasonic Ranging module x1 
Jumper M/F x4 	

Component Knowledge

Ultrasonic ranging module use the principle that ultrasonic will reflect when it encounters obstacles. 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=340m/s. So we can calculate the distance between the modle and the obstacle: s=vt/2.



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. The object picture and the diagram of HC SR04 ultrasonic module are shown below:



Pin description:

VCC	power supply pin
Trig	triger pin
Echo	Echo pin
GND	GND

Technical specs:

Working voltage: 5V

Working current: 12mA

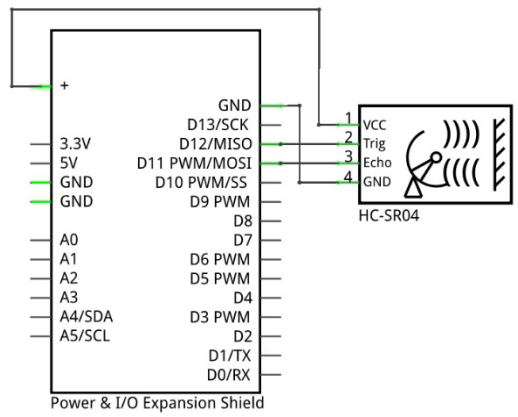
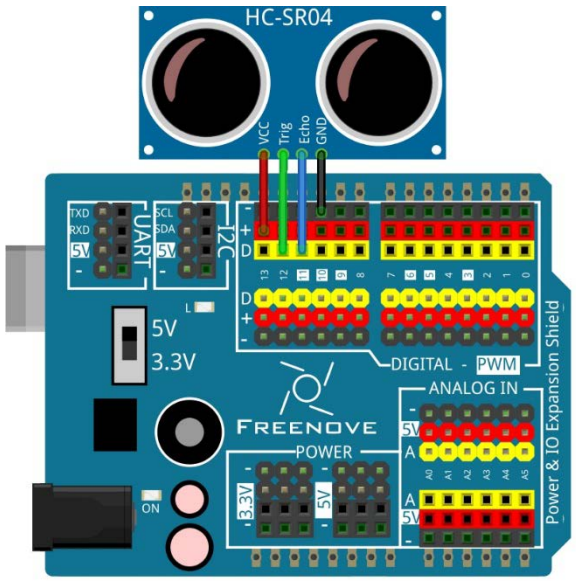
Minimum measuring distance: 2cm

Maximum measuring distance: 200cm

Size: 45mm*20mm*15mm

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 SR-04 HC in Arduino. We can also use the library function to directly obtain the measuring distance. The library is developed by third party enthusiasts, and we also need import the file NewPing.zip before using the library.

Circuit

Schematic diagram	Circuit connection
	

Sketch

First, we use the HC_SR04 communication protocol to operate the module, get the range of time, and calculate the distance.

Sketch 16.1.1 UltrasonicRanging_01_ Communication

```
#define trigPin  12  // define TrigPin
#define echoPin     11  // define EchoPin.
#define MAX_DISTANCE 200 //  Maximum sensor distance is rated at 400-500cm.
/define the timeOut according to the maximum range. timeOut= 2*MAX_DISTANCE /100 /340
  *1000000 = MAX_DISTANCE*58.8
float timeOut = MAX_DISTANCE * 60;
int soundVelocity = 340;  //define sound speed=340m/s
void setup() {
  pinMode(trigPin,OUTPUT); // set trigPin to output mode
  pinMode(echoPin,INPUT);// set echoPin to input mode
  Serial.begin(115200); // Open serial monitor at 115200 baud to see ping results.
}
void loop() {
  delay(100);                          // Wait 50ms between pings (about 20 pings/sec).
29ms should be the shortest delay between pings.
  Serial.print("Ping: ");
  Serial.print(getSonar()); // Send ping, get distance in cm and print result (0 =
outside set distance range)
  Serial.println("cm");
}
```

```
21  float getSonar() {
22    unsigned long pingTime;
23    float distance;
24    digitalWrite(trigPin, HIGH); // make trigPin output high level lasting for 10μs to
25  triger HC_SR04,
26    delayMicroseconds(10);
27    digitalWrite(trigPin, LOW);
28    pingTime = pulseIn(echoPin, HIGH, timeOut);// Wait  HC_SR04 returning to the high
29  level and measure out this waitting time
30    distance = (float)pingTime * soundVelocity / 2 / 10000; // calculate the distance
31  according to the time
32    return distance;// return the distance value
    }
```

First, define the pins and the maximum measurement distance.

```
#define trigPin  12  // define TrigPin
#define echoPin     11  // define EchoPin.
#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. timOut= 2*MAX_DISTANCE/100/340*1000000. The constant part behind is approximately equal to 58.8.

```
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 HC_SR04. Make trigPin output a high level lasting for at least 10μS, according to the communication protocol.

```
  digitalWrite(trigPin, HIGH); // make trigPin output high level lasting for 10μs to
triger HC_SR04,
  delayMicroseconds(10);
  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.

```
pingTime = pulseIn(echoPin, HIGH, timeOut);
```

Calculate the distance according to the time and return the value.

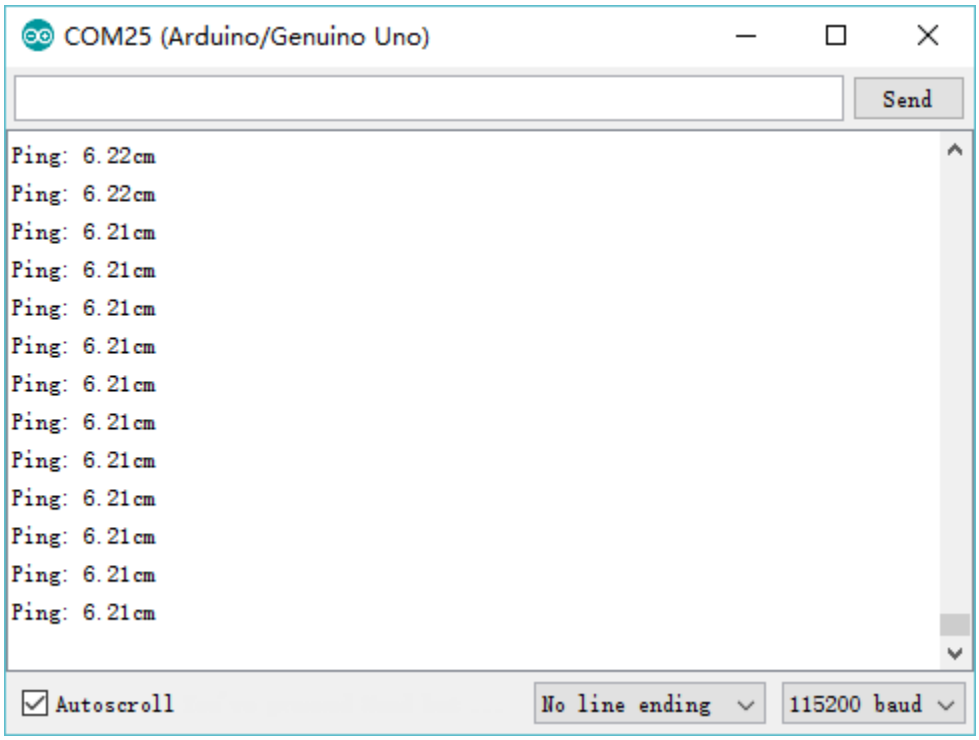
```
  distance = (float)pingTime * soundVelocity / 2 / 10000; // calculate the distance
according to the time
  return distance;// return the distance value
```

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.



Of course, we can also use the library function to operate the HC_SR04 ultrasonic ranging module.

Sketch 16.1.2 UltrasonicRanging_02_Ping

```
#include <NewPing.h>
#define trigPin  12  // define TrigPin
#define echoPin     11  // define EchoPin.
#define MAX_DISTANCE 200 //  Maximum sensor distance is rated at 400-500cm.

NewPing sonar(trigPin, echoPin, MAX_DISTANCE); // NewPing setup of pins and maximum distance.

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

void loop() {
  delay(50);                          // Wait 50ms between pings (about 20 pings/sec). 29ms should be the shortest delay between pings.
  Serial.print("Ping: ");
  Serial.print(sonar.ping_cm()); // Send ping, get distance in cm and print result (0 = outside set distance range)
  Serial.println("cm");
}
```

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.

```
#include <NewPing.h>
```

```
#define trigPin  12
#define echoPin     11
#define MAX_DISTANCE 200
NewPing sonar(trigPin, echoPin, MAX_DISTANCE);
```

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.

NewPing sonar(trigger_pin, echo_pin [, max_cm_distance])

Some member function:

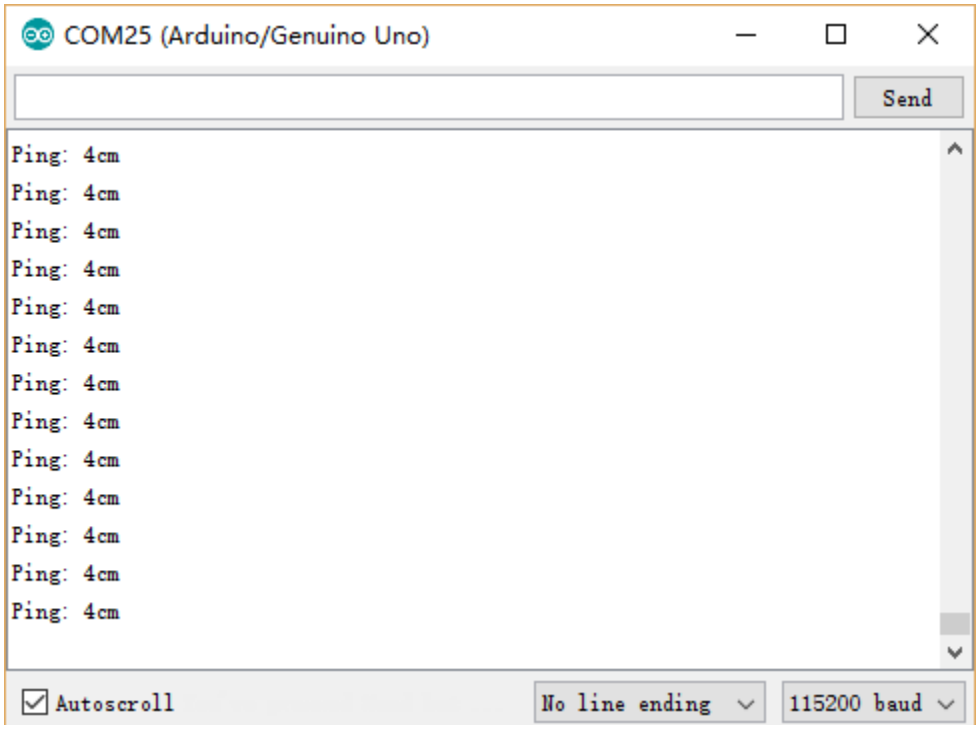
sonar.ping() - Send a ping and get the echo time (in microseconds) as a result.

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 observate the distance changes.



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.