FREENOVE

FREE YOUR INNOVATION

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

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

Chapter 0 Procesing

Processing software is used to write programs that can run on computers. Processing software is free and open source, and runs on the Mac, Windows, and GNU/Linux platforms, which is the same with Arduino software. In fact, the development of Arduino software is based on Processing software, and they still have similar interface.

Programs written using Processing are also called sketches, and Java is the default language. Java language and C++ language has many similarities, so the readers who have learned our basic tutorial are able to understand and write simple Processing sketches quickly.

This tutorial will introduce how to install and use processing software on Raspberry Pi through some electronic circuit projects. Chapters and sequence is similar to C and python tutorial. Equally, detailed description and explanation is arranged for the skect in each project. Our elaborate electronic circuits and interactive project with Processing are attached in the end, including virtual instruments, games (2D and 3D versions), etc.

Install Processing Software

Processing software / Processing Development Environment (PDE) makes it easy to write Processing programs. First install Processing software: type following command in the terminal to start installation:

```
curl https://processing.org/download/install-arm.sh | sudo sh
```
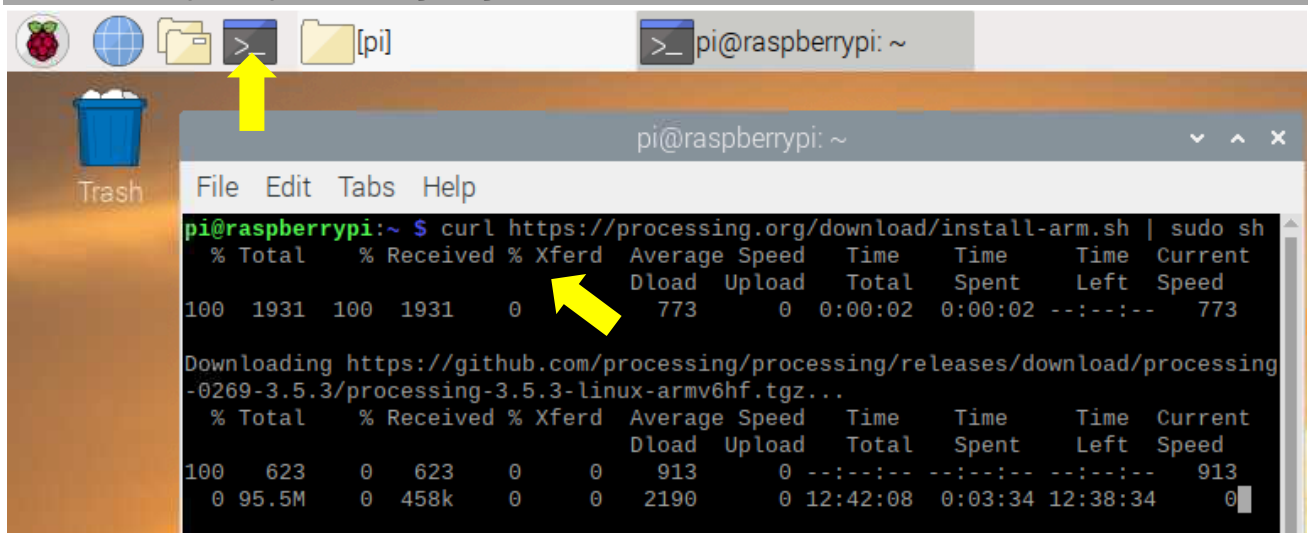

Ensures that your RPi always has the Internet to access in the installation process.
You can also download and install the software by visiting the official website https://processing.org/.

After the installation is completed, you can enter the "processing" to open processing software in any directory of the terminal, or open the software processing in the start menu of the system, as shown below:
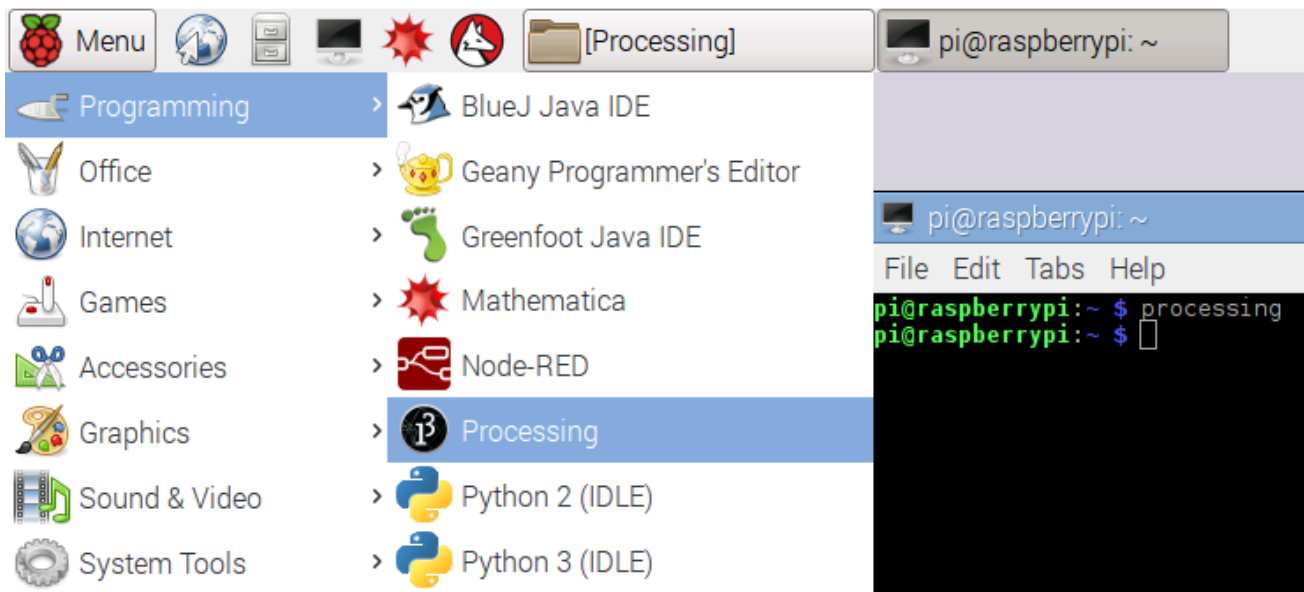
Interface of processing software is shown below:
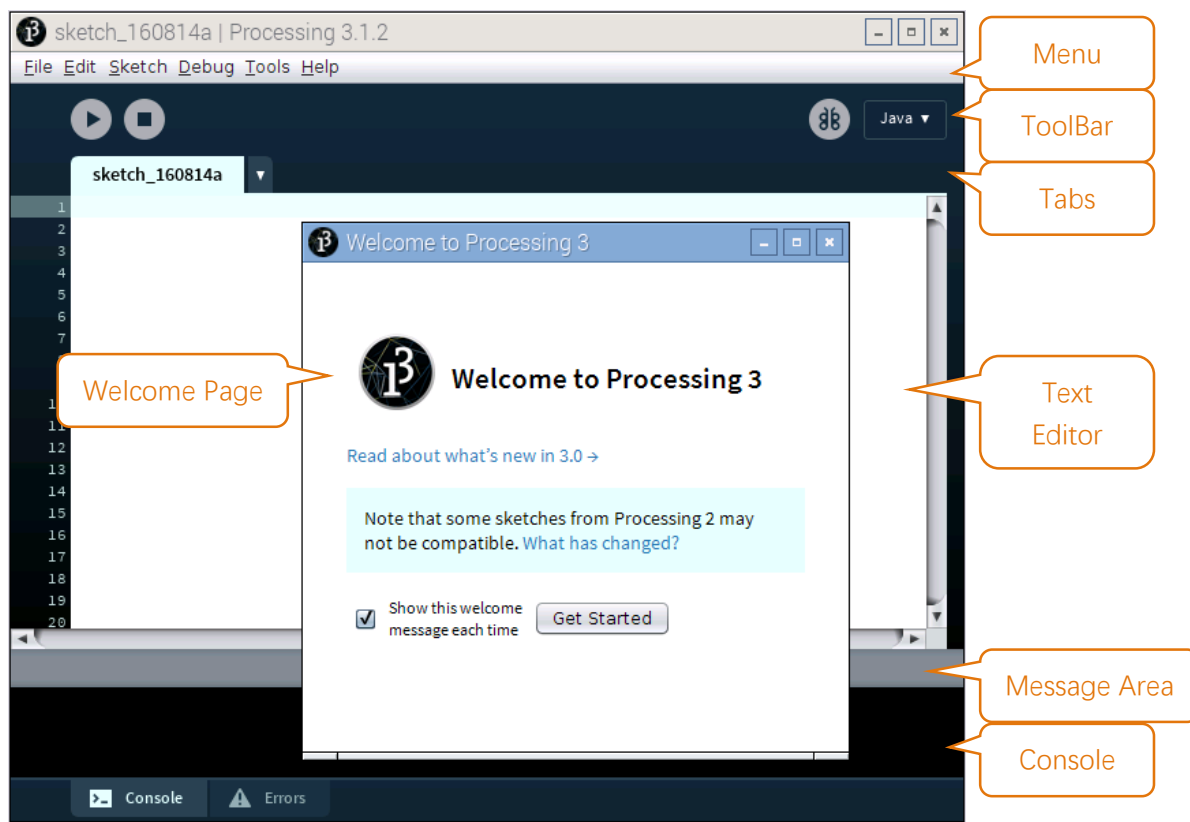
You're now running the Processing Development Environment (or PDE). There's not much to it; the large area is the Text Editor, and there's a row of buttons across the top; this is the toolbar. Below the editor is the Message Area, and below that is the Console. The Message Area is used for one line messages, and the Console is used for more technical details.
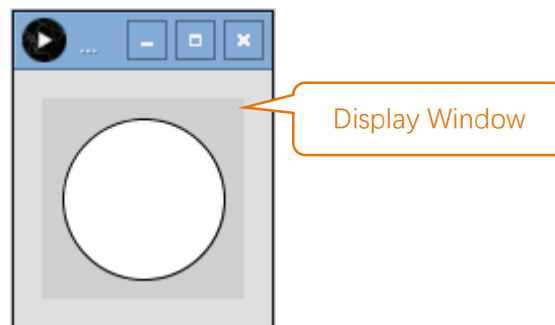
First Use

In the editor, type the following:

1	ellipse(50, 50, 80, 80);

This line of code means "draw an ellipse, with the center 50 pixels over from the left and 50 pixels down from the top, with a width and height of 80 pixels." Click the Run button (the triangle button in the Toolbar).
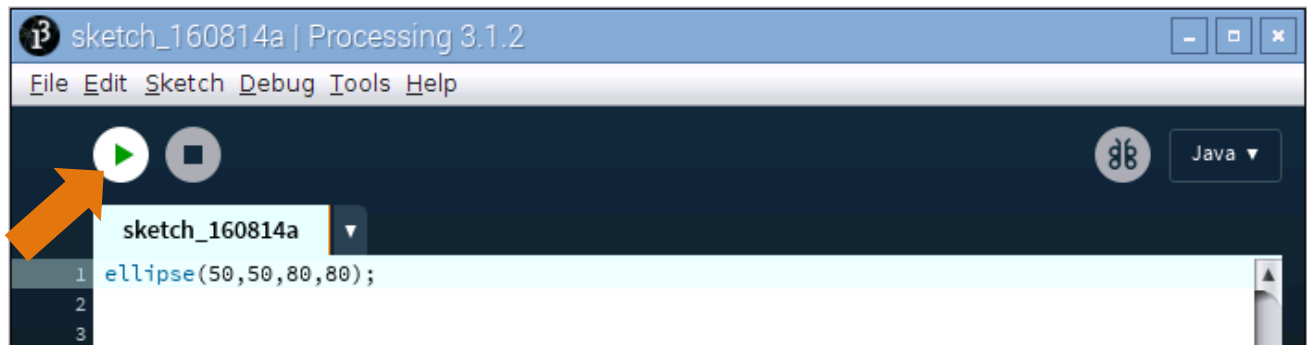
If you've typed everything correctly, you'll see a circle on your screen.

Click on "Stop" (the rectangle button in the Toolbar) or "Close" on Display Window to stop running the program.

If you didn't type it correctly, the Message Area will turn red and complain about an error. If this happens, make sure that you've copied the example code exactly: the numbers should be contained within parentheses and have commas between each of them, and the line should end with a semicolon.

You can export this sketch to an application to run it directly without opening the Processing.

To export the sketch to the application, you must first save it.
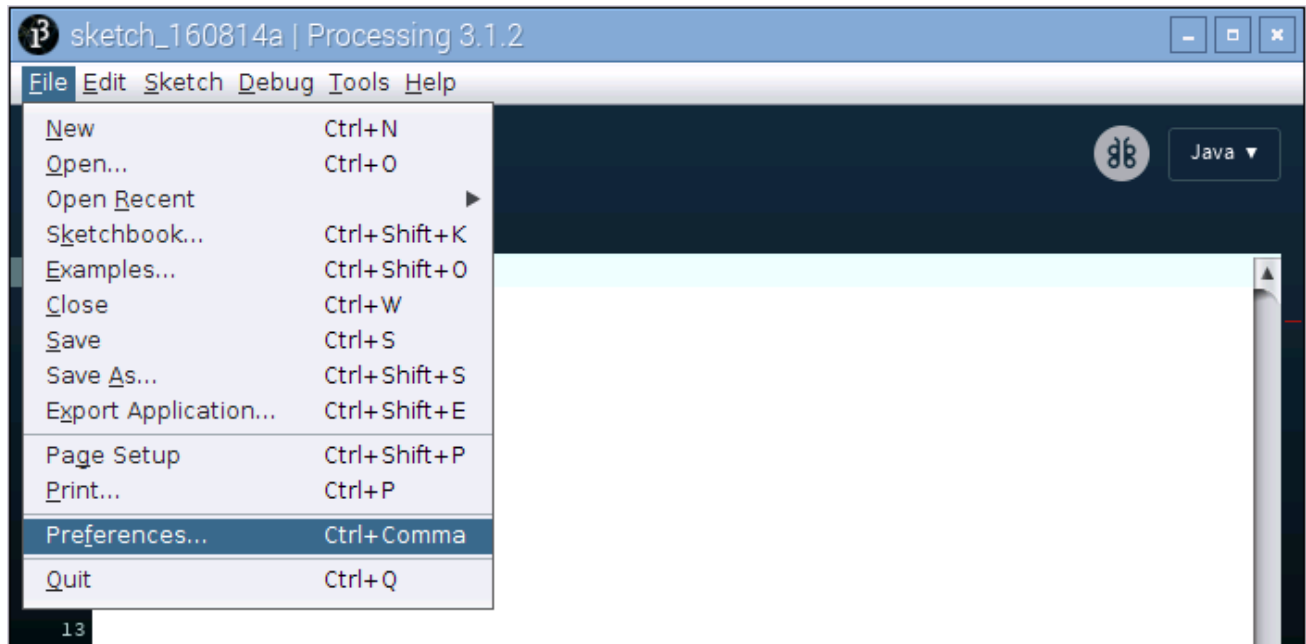
So far, we have completed the first use. I believe you have felt the joy of it.
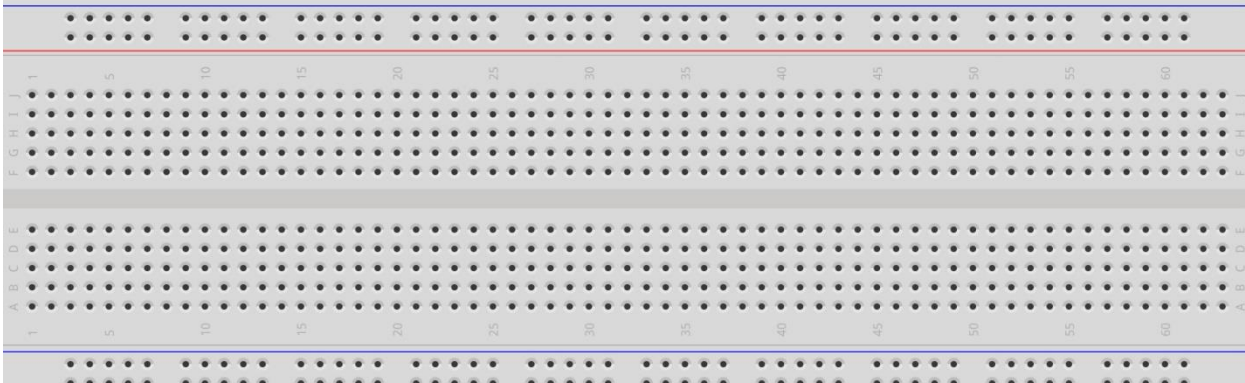
Chapter 1 LED

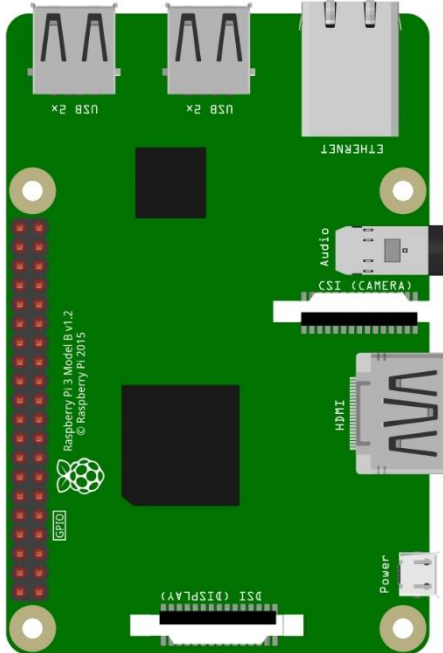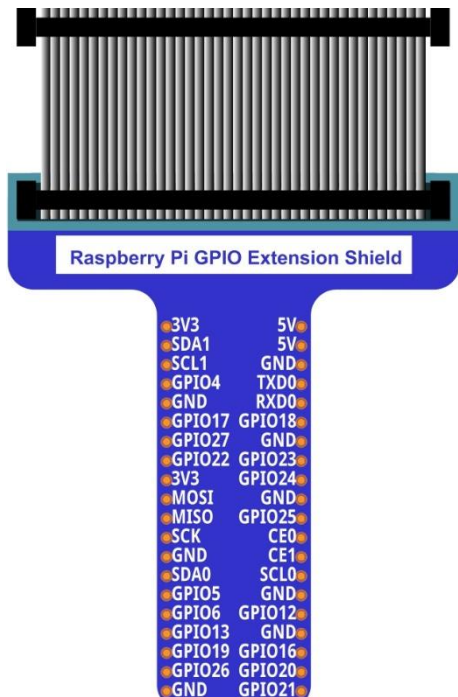We will still start from Blink LED in this chapter, and also learn the usage of some commonly used functions of Processing Software.

Project 1.1 Blink

In this project, we will make a Blink LED and let Display window of Processing Blink at the same time.

Component List

Raspberry Pi x1	GPIO Extension Board & Wire x1

BreadBoard x1

LED x1	Resistor 220Ω x1	Jumper Wire M/M x2

In the components list, Raspberry Pi, GPIO Extension Shield and Breadboard are necessary for each experiment. They will be listed only in text form.

Circuit

Disconnect RPi from GPIO Extension Shield first. Then build the circuit according to the circuit diagram and the hardware connection diagram. After the circuit is built and confirmed, connect RPi to GPIO Extension Shield. In addition, short circuit (especially 5V and GND, 3.3V and GND) should be avoid, because short circuit may cause anormal circuit work, or even damage to PRi.

Schematic diagram
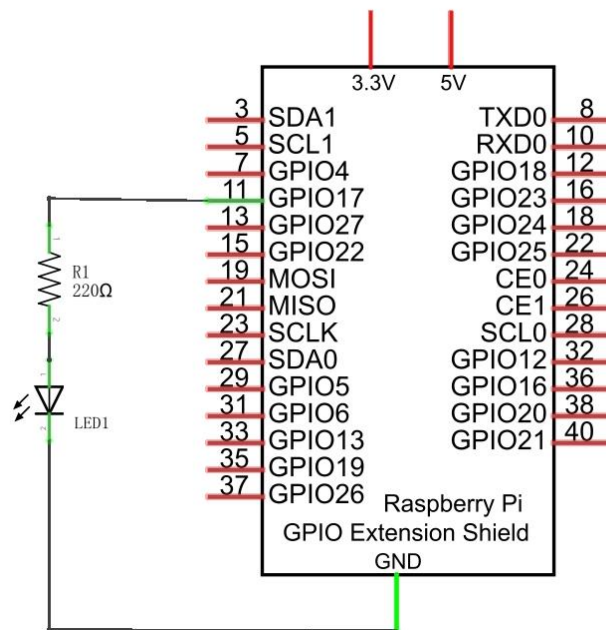
Hardware connection
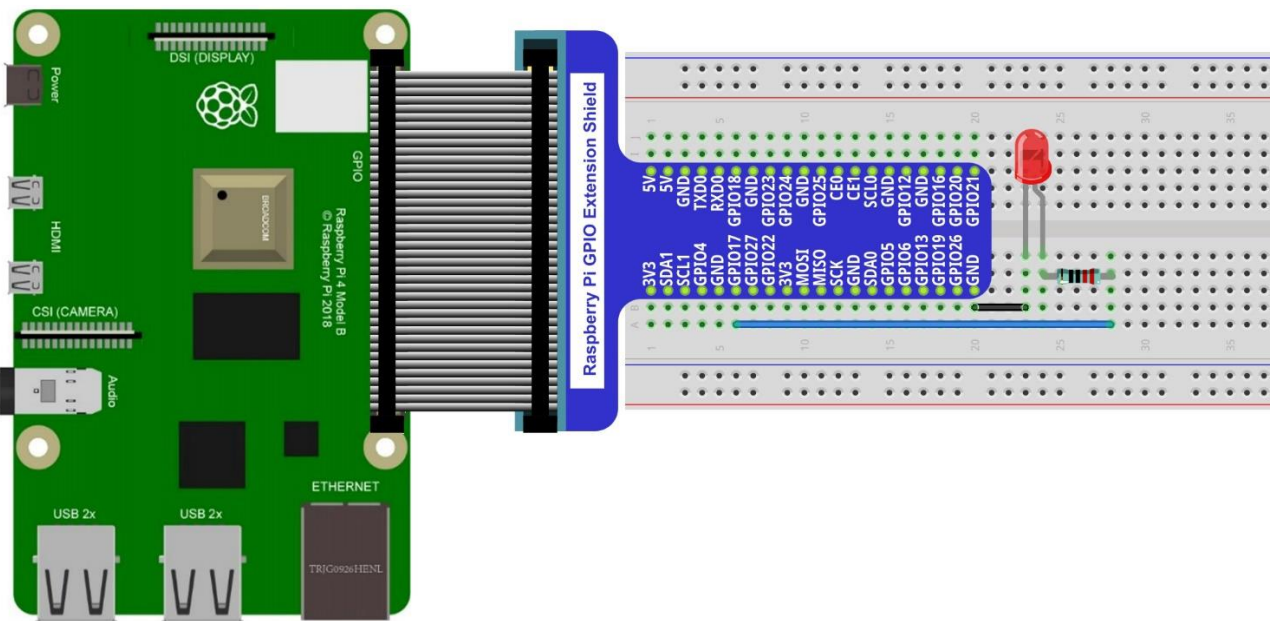
Because Numbering of GPIO Extension Shield is the same as RPi GPIO, latter Hardware connection diagram will only show the part of breadboard and GPIO Extension Shield.

Sketch

Sketch 1.1.1 Blink

Because the resource folder name is too long. For convenience, the folder will be named to "Freenove_Kit".
If you have already renamed it, skip this command. Assume the absolute path is "/ home / pi" or "~ /". Execute the following command in the user directory.

```
mv Freenove_Ultimate_Starter_Kit_for_Raspberry_Pi/ Freenove_Kit/
```

First observe the running result of the sketch, and then analyze the code.
1. Use Processing to open the file Sketch_01_1_1_Blink. (Following is only one line of command. There is a Space after processing.)

```
Processing
~/Freenove_Kit/Processing/Sketches/Sketch_01_1_1_Blink/Sketch_01_1_1_Blink.pde
```
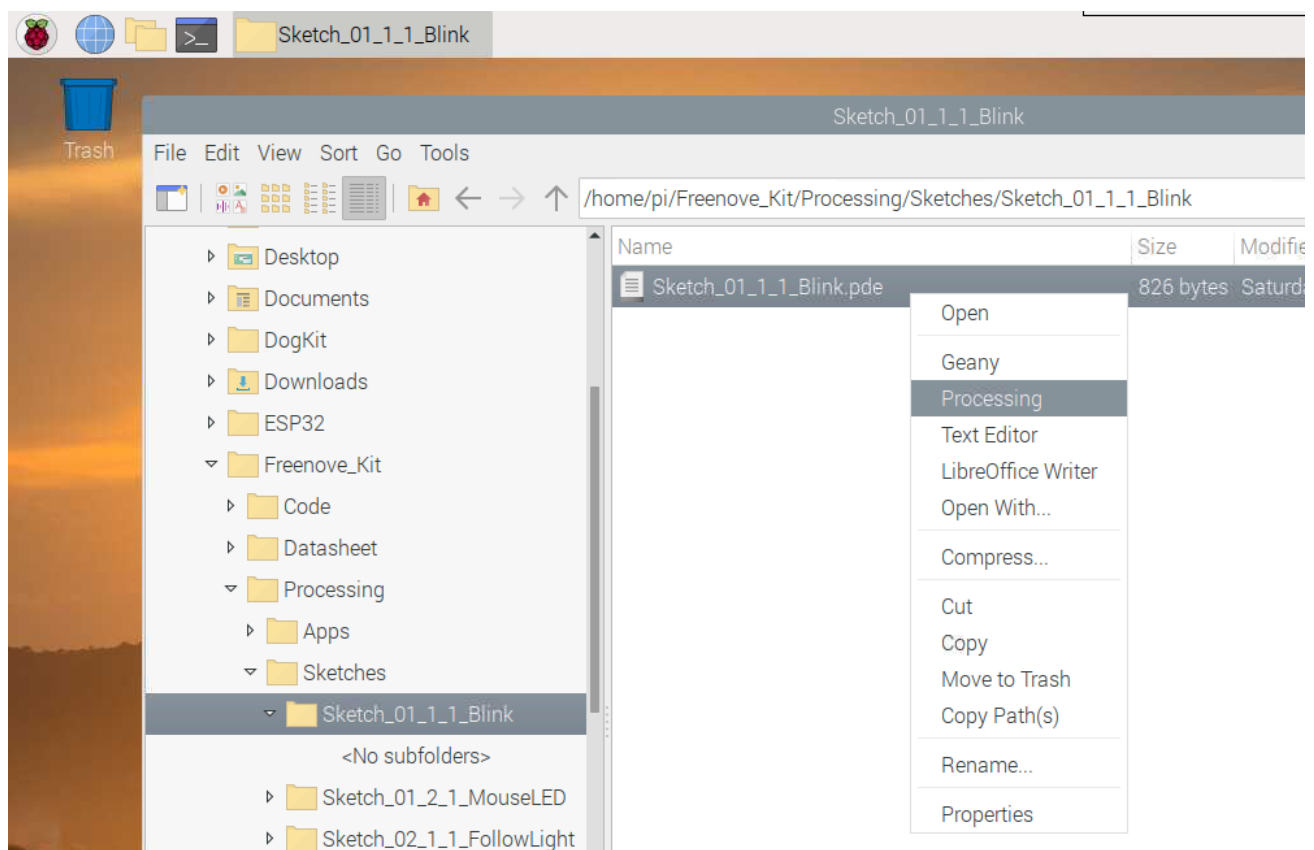
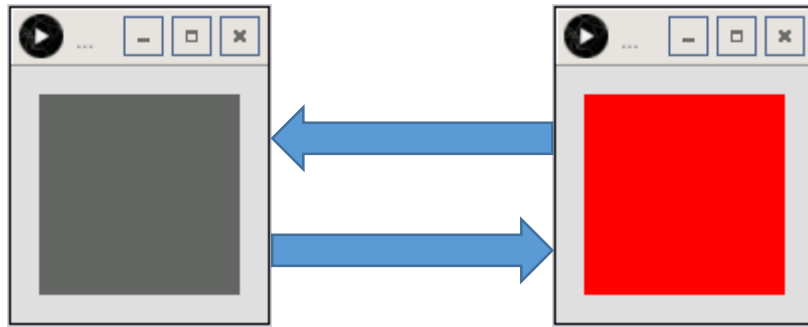2. Click on "RUN" to run the code.

You can also open it as below.
Click Raspberry Pi file manerger. Find the file under path below:
/home/pi/Freenove_Kit/Processing/Sketches/Sketch_01_1_1_Blink
Then right click and choose Processing.

After the program is executed, LED will start Blinking and background of Display window will change with the change of LED state.

The following is program code:

```
1    import processing.io.*;
2
3    int ledPin = 17;    //define ledPin
4    boolean ledState = false;    //define ledState
5
6    void setup() {
7      size(100, 100);
8      frameRate(1);         //set frame rate
9      GPIO.pinMode(ledPin, GPIO.OUTPUT);    //set the ledPin to output mode
10   }
11
12   void draw() {
13     ledState = !ledState;
14     if (ledState) {
15       GPIO.digitalWrite(ledPin, GPIO.HIGH);    //led on
16       background(255, 0, 0); //set the fill color of led on
17     } else {
18       GPIO.digitalWrite(ledPin, GPIO.LOW);    //led off
19       background(102); //set the fill color of led off
20     }
21   }
```

Processing code usually have two functions: setup() and draw(), where the function setup()is only executed once, but the function draw() will be executed circularly. In the function setup(), size(100, 100) specifies the size of the Display Window to 100x100pixl. FrameRate(1) specifies the refresh rate of Display Window to once per second, namely, the draw() function will be executed once per second. GPIO.pinMode (ledPin, GPIO.OUTPUT) is used to set ledPin to output mode.

```
void setup() {
  size(100, 100);
  frameRate(1);         //set frame rate
  GPIO.pinMode(ledPin, GPIO.OUTPUT);    //set the ledPin to output mode
}
```

In draw() function, each execution will invert the variable "ledState". When "ledState" is true, LED is turned on, and the background color of display window is set to red. And when the "ledState" is false, the LED will be turned off and the background color of display window is set to gray. Since the function draw() is executed once per second, the background color of Display Window and the state of LED will also change once per second. Such cycle repeats itself to achieve the effect of blink.

```
void draw() {
  ledState = !ledState;
  if (ledState) {
    GPIO.digitalWrite(ledPin, GPIO.HIGH);    //led on
    background(255, 0, 0); //set the fill color of led on
  } else {
    GPIO.digitalWrite(ledPin, GPIO.LOW);     //led off
    background(102); //set the fill color of led off
  }
}
```

The following is simple description of some functions:

setup()
The setup() function is run once, when the program starts.
draw()
Called directly after setup(), the draw() function continuously executes the lines of code contained inside its block until the program is stopped or noLoop() is called. draw() is called automatically and should never be called explicitly.
size()
Defines the dimension of the display window width and height in units of pixels
framerate()
Specifies the number of frames to be displayed every second
background()
Set the color used for the background of the Processing window.
GPIO.pinMode()
Configures a pin to act either as input or output
GPIO.digitalWrite()
Sets an output pin to be either high or low

All functions used in this code can be found in the Reference of Processing Software, in which built-in functions are described in details, and there are some sample programs. It is recommended for beginners to view more usage and functions of the function. The localization of Reference can be opened by the following steps: click the menu bar "Help"➔"Reference".
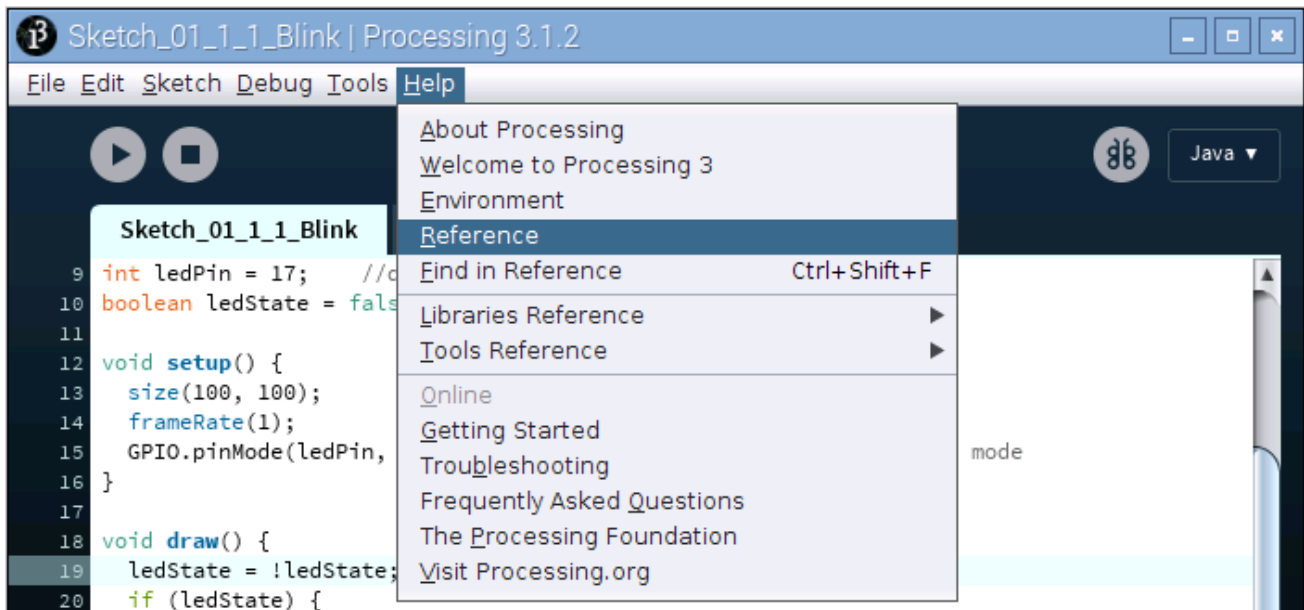
Then the following page will be displayed in the web browser:

Or directly access to the official website for reference:http://processing.org/reference/

Project 1.2 MouseLED

In this project, we will use the mouse to control the state of LED.

The components and circuits of this project are the same as the last section.

Sketch

Sketch 1.2.1 MouseLED

First observe the running result of the sketch, and then analyze the code.

1. Use Processing to open the file Sketch_01_2_1_MouseLED.

```
processing
~/Freenove_Kit/Processing/Sketches/Sketch_01_2_1_MouseLED/Sketch_01_2_1_MouseL
ED.pde
```

2. Click on "RUN" to run the code.

After the program is executed, the LED is under off state, and background color of Display window is gray. Click on Display Window with the mouse, then LED is turned on and Display window background color become red. Click on the Display Window again, then the LED is turned off and the background color become gray, as shown below.
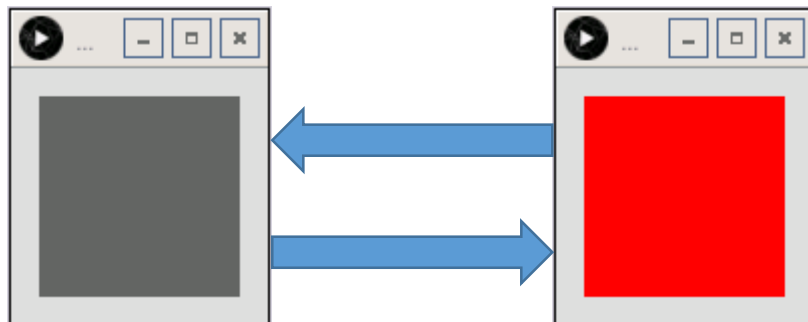
The following is program code:

```
1     import processing.io.*;
2
3     int ledPin = 17;
4     boolean ledState = false;
5     void setup() {
6       size(100, 100);
7       GPIO.pinMode(ledPin, GPIO.OUTPUT);
8       background(102);
9     }
10
11    void draw() {
12      if (ledState) {
13        GPIO.digitalWrite(ledPin, GPIO.HIGH);
14        background(255,0,0);
15      } else {
```

```
16        GPIO.digitalWrite(ledPin, GPIO.LOW);
17        background(102);
18      }
19    }
20
21    void mouseClicked() { //if the mouse Clicked
22      ledState = !ledState;  //Change the led State
23    }
```

The function mouseClicked() is used in this code. The function is used to capture the mouse click events, which is executed when the mouse is clicked on. We can change the state of the variable "ledState" in this function, to realize controlling LED through clicking on the mouse.

```
void mouseClicked() { //if the mouse Clicked
  ledState = !ledState;  //Change the led State
}
```

Chapter 2 LEDBar Graph

We have learned how to control a LED, and next we will learn how to control a number of LED.

Project 2.1 FollowLight

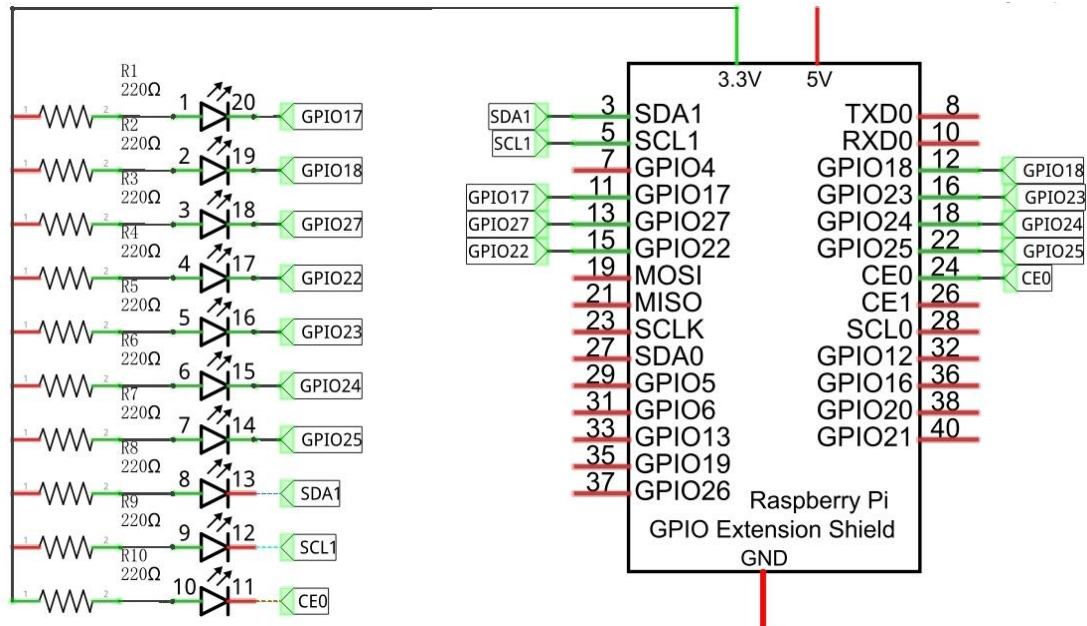In this project, we will use the mouse to control the LEDBar Graph

Component List

Raspberry Pi x1 GPIO Extension Board & Wire x1 BreadBoard x1	LED bar graph x1	Resistor 220Ω x10
Jumper M/M x11		

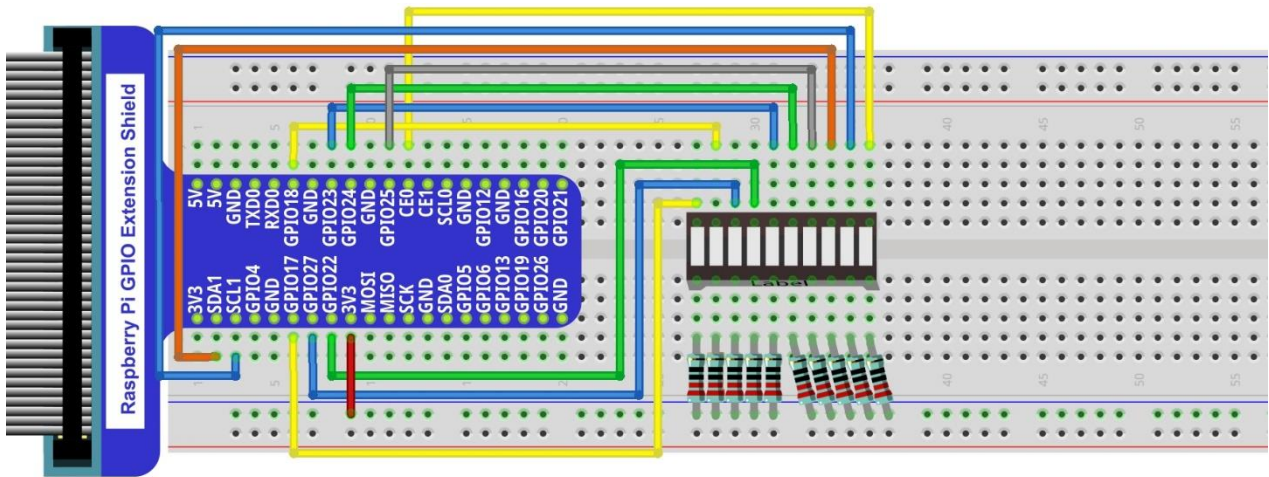# Circuit

The network label is used in the circuit diagram below, and the pins with the same network label are connected togather.

Schematic diagram
Hardware connection

In this circuit, the cathode of LED is connected to GPIO, which is the different from the front circuit. So, LED will be turned on when GPIO output low level in the program.

Sketch

Sketch 2.1.1 FollowLight

First observe the running result of the sketch, and then analyze the code.
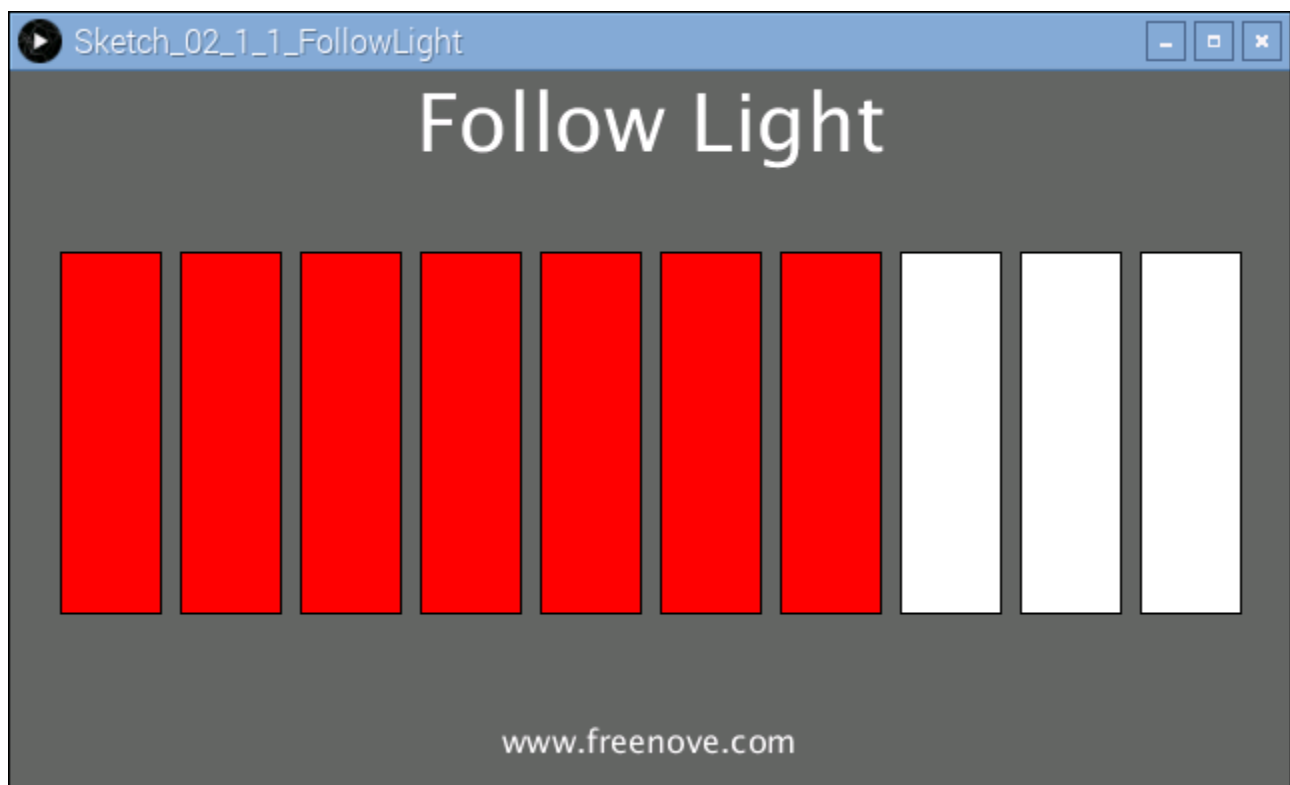
1.   Use Processing to open the file Sketch_02_1_1_FollowLight.

```
processing
~/Freenove_Kit/Processing/Sketches/Sketch_02_1_1_FollowLight/Sketch_02_1_1_Foll
owLight.pde
```

2.   Click on "RUN" to run the code.

After the program is executed, slide the mouse in the Display Window, then the state of LEDBar Graph will be changed, as shown below.

The following is program code:

```
1   import processing.io.*;
2
3   int leds[]={17, 18, 27, 22, 23, 24, 25, 2, 3, 8}; //define ledPins
4
5   void setup() {
6     size(640, 360);  //display window size
7     for (int i=0; i<10; i++) {  //set led Pins to output mode
8       GPIO.pinMode(leds[i], GPIO.OUTPUT);
9     }
10    background(102);
```

```
11      textAlign(CENTER);    //set the text centered
12      textSize(40);         //set text size
13      text("Follow Light", width / 2, 40);    //title
14      textSize(16);
15      text("www.freenove.com", width / 2, height - 20);    //site
16    }
17
18    void draw() {
19      for (int i=0; i<10; i++) {    //draw 10 rectanglar box
20        if (mouseX>(25+60*i)) {    //if the mouse cursor on the right of rectanglar box
21          fill(255, 0, 0);         //fill the rectanglar box in red color
22          GPIO.digitalWrite(leds[i], GPIO.LOW);  //turn on the corresponding led
23        } else {
24          fill(255, 255, 255);  //else fill the rectanglar box in white color
25          GPIO.digitalWrite(leds[i], GPIO.HIGH);  //and turn off the led
26        }
27        rect(25+60*i, 90, 50, 180);    //draw a rectanglar box
28      }
29    }
```

In the function draw(), we draw 10 rectangles to represent 10 LEDs of LEDBar Graph. We make rectangles on the left of mouse filled with red, corresponding LEDs turned on. And make We make rectangles on the right of mouse filled with red, corresponding LEDs turned off. In this way, when slide the mouse to right, the more LEDs on the left of mouse will be turned on. When to the left, the reverse is the case.

```
void draw() {
  for (int i=0; i<10; i++) {    //draw 10 rectanglar box
    if (mouseX>(25+60*i)) {    //if the mouse cursor on the right of rectanglar box
      fill(255, 0, 0);         //fill the rectanglar box in red color
      GPIO.digitalWrite(leds[i], GPIO.LOW);  //turn on the corresponding led
    } else {
      fill(255, 255, 255);  //else fill the rectanglar box in white color
      GPIO.digitalWrite(leds[i], GPIO.HIGH);  //and turn off the led
    }
    rect(25+60*i, 90, 50, 180);    //draw a rectanglar box
  }
}
```

Chapter 3 PWM

In this chapter, we will learn how to use PWM.
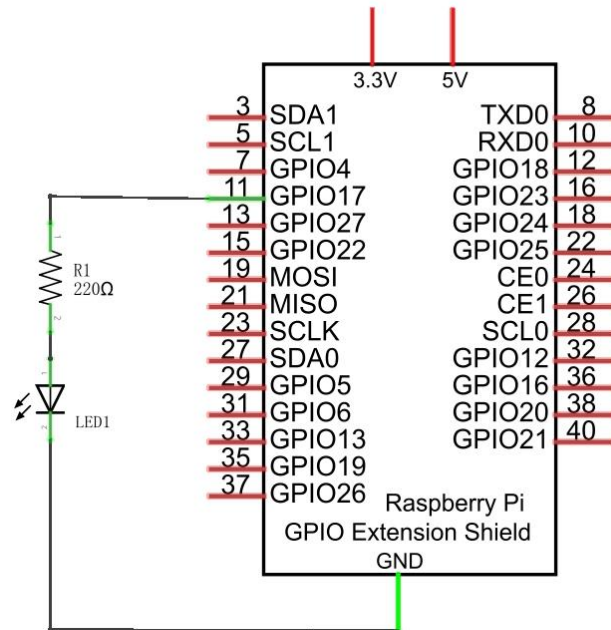
Project 3.1 BreathingLED

In this project, we will make a breathing LED, and the Display Window will show a breathing LED pattern and a progress bar, at the same time.
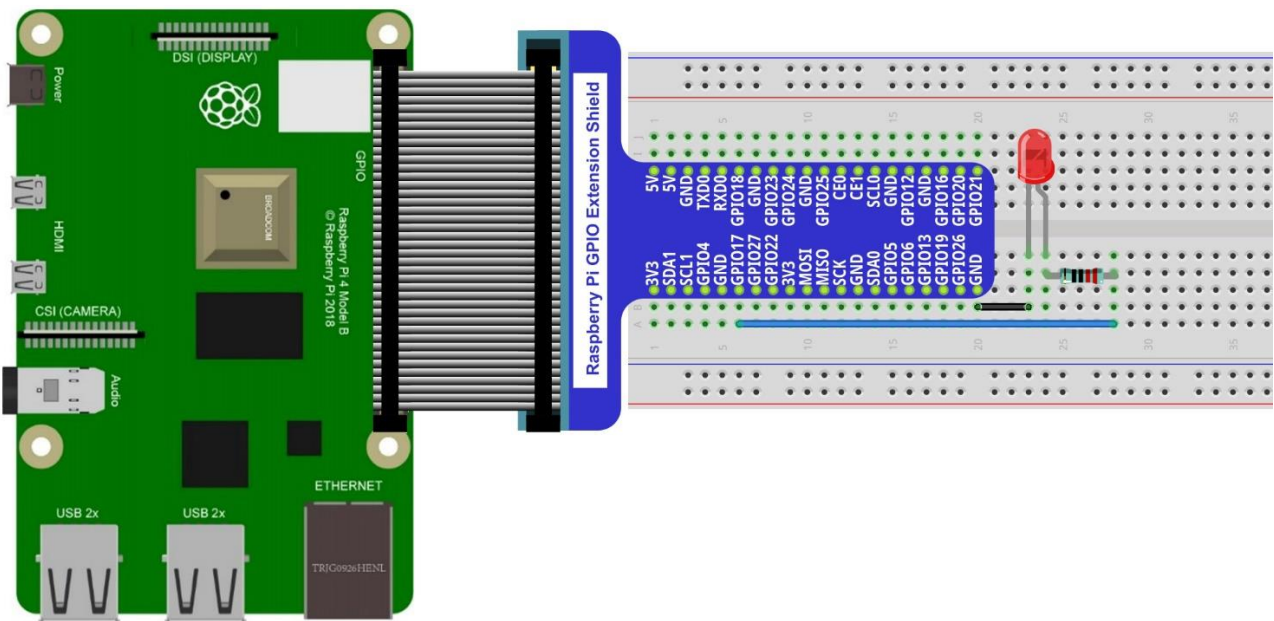
Component List

Raspberry Pi x1 GPIO Extension Board & Wire x1 BreadBoard x1	LED x1	Resistor 220Ω x1
Jumper M/M x2		

Circuit

| Schematic diagram |
|---|
| |
| Hardware connection |
| |

Sketch

First observe the running result of the sketch, and then analyze the code.

1. Use Processing to open the file Sketch_03_1_1_BreathingLED.

```
processing
~/Freenove_Kit/Processing/Sketches/Sketch_03_1_1_BreathingLED/Sketch_03_1_1_Bre
athingLED.pde
```

2. Click on "RUN" to run the code.

After the program is executed, the LED in circuit will be brightened gradually, and the color of LED pattern in Display Window will be deepen gradually at the same time. The progress bar under the patten shows the percentage of completion, and clicking on the inside of window with the mouse can change the progress.
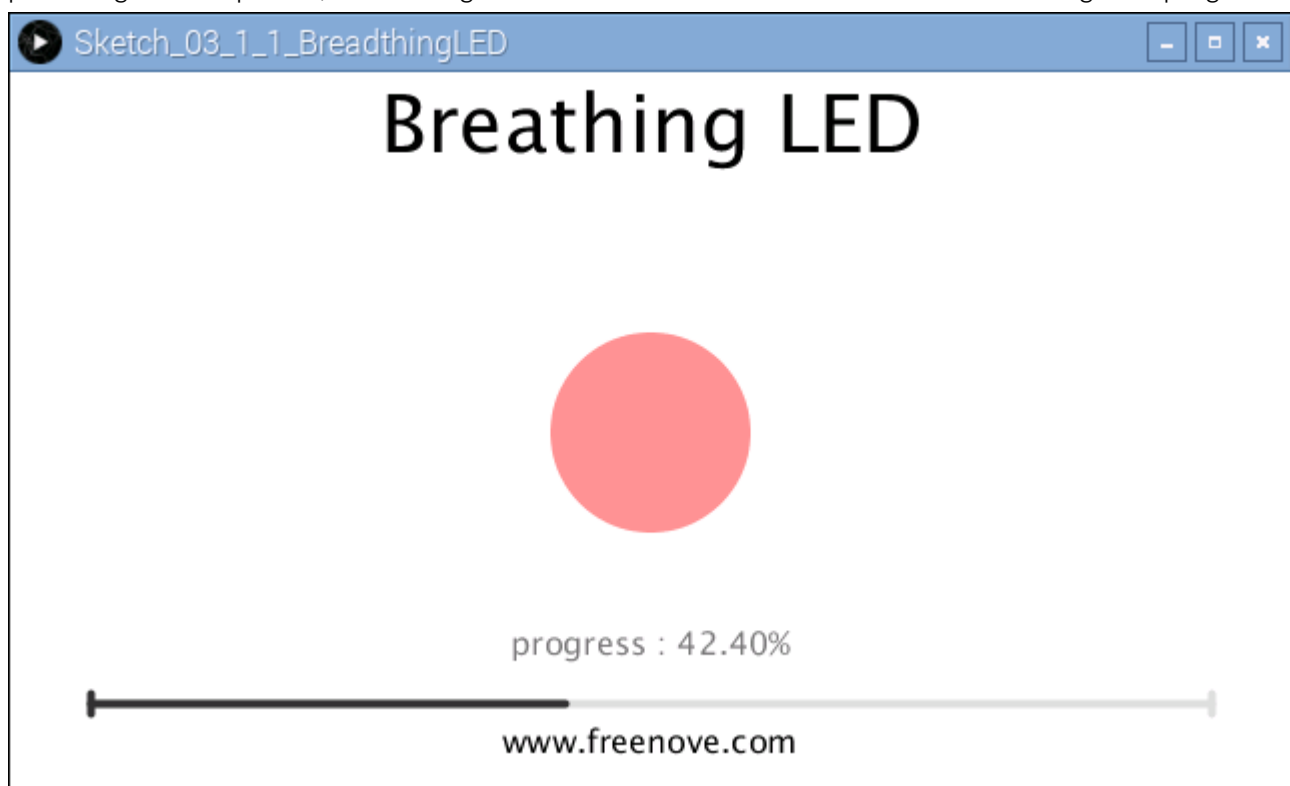
The following is program code:

```
1    import processing.io.*;
2
3    int ledPin = 17;      //led Pin
4    int borderSize = 40;  //
5    float t = 0.0;         //progress percent
6    float tStep = 0.004;    // speed
7    SOFTPWM p = new SOFTPWM(ledPin, 10, 100);    //Create a PWM pin,initialize the duty cycle
8    and period
9    void setup() {
10     size(640, 360);  //display window size
```

```
11        strokeWeight(4);   //stroke Weight
12    }
13
14    void draw() {
15       // Show static value when mouse is pressed, animate otherwise
16       if (mousePressed) {
17          int a = constrain(mouseX, borderSize, width - borderSize);
18          t = map(a, borderSize, width - borderSize, 0.0, 1.0);
19       } else {
20          t += tStep;
21          if (t > 1.0) t = 0.0;
22       }
23       p.softPwmWrite((int)(t*100)); //wirte the duty cycle according to t
24       background(255);   //A white background
25       titleAndSiteInfo();   //title and Site infomation
26
27       fill(255, 255-t*255, 255-t*255);   //cycle
28       ellipse(width/2, height/2, 100, 100);
29
30       pushMatrix();
31       translate(borderSize, height - 45);
32       int barLength = width - 2*borderSize;
33
34       barBgStyle();   //progressbar bg
35       line(0, 0, barLength, 0);
36       line(barLength, -5, barLength, 5);
37
38       barStyle();   //progressbar
39       line(0, -5, 0, 5);
40       line(0, 0, t*barLength, 0);
41
42       barLabelStyle();    //progressbar label
43       text("progress : "+nf(t*100,2,2),barLength/2,-25);
44       popMatrix();
45    }
46
47    void titleAndSiteInfo() {
48       fill(0);
49       textAlign(CENTER);    //set the text centered
50       textSize(40);          //set text size
51       text("Breathing Light", width / 2, 40);    //title
52       textSize(16);
53       text("www.freenove.com", width / 2, height - 20);    //site
54    }
```

```
55   void barBgStyle() {
56     stroke(220);
57     noFill();
58   }
59
60   void barStyle() {
61     stroke(50);
62     noFill();
63   }
64
65   void barLabelStyle() {
66     noStroke();
67     fill(120);
68   }
```

First, use SOFTPWM class to create a PWM pin, which is used to control the brightness of LED. Then define a variable "t" and variable "tStep" to control the PWM duty cycle and add-self rate.

```
float t = 0.0;        //progress percent
float tStep = 0.004;    // speed
SOFTPWM p = new SOFTPWM(ledPin, 10, 100);
```

In the function draw, if there is a click, the coordinate in X direction of mouse will be mapped into the duty cycle "t", otherwise, duty cycle "t" will be increased gradually. Then output PWM with the duty cycle.

```
if (mousePressed) {
    int a = constrain(mouseX, borderSize, width - borderSize);
    t = map(a, borderSize, width - borderSize, 0.0, 1.0);
} else {
    t += tStep;
    if (t > 1.0) t = 0.0;
}
p.softPwmWrite((int)(t*100)); //wirte the duty cycle according to t
```

The next code is designed to draw a circle filled colors with different depth according to the "t" value, which is used to simulate the of LED with different brightness.

```
fill(255, 255-t*255, 255-t*255);  //cycle
ellipse(width/2, height/2, 100, 100);
```

The last code is designed to draw the progress bar and the percentage of the progress.

```
barBgStyle();  //progressbar bg
line(0, 0, barLength, 0);
line(barLength, -5, barLength, 5);

barStyle();  //progressbar
line(0, -5, 0, 5);
```
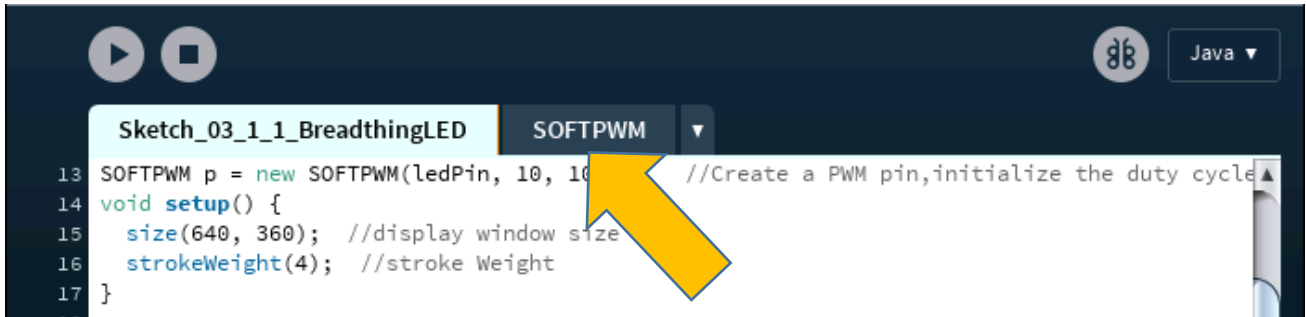
```
    line(0, 0, t*barLength, 0);


    barLabelStyle();      //progressbar label
    text("progress : "+nf(t*100,2,2),barLength/2,-25);
```

In processing software, you will see a tag page "SOFTPWM" in addition to above code.

The file contains some information about the SOFTPWM class.

class SOFTPWM

```
public SOFTPWM(int iPin, int dc, int pwmRange):
```
Constructor, used to create a PWM pin, set the pwmRange and initial duty cycle. The minimum of pwmRange is 0.1ms. So pwmRange=100 means that the PWM cycle is 0.1ms*100=10ms.
```
public void softPwmWrite(int value)
```
Set PMW duty cycle.
```
public void softPwmStop()
```
Stop outputting PMW.

Chapter 4 RGBLED

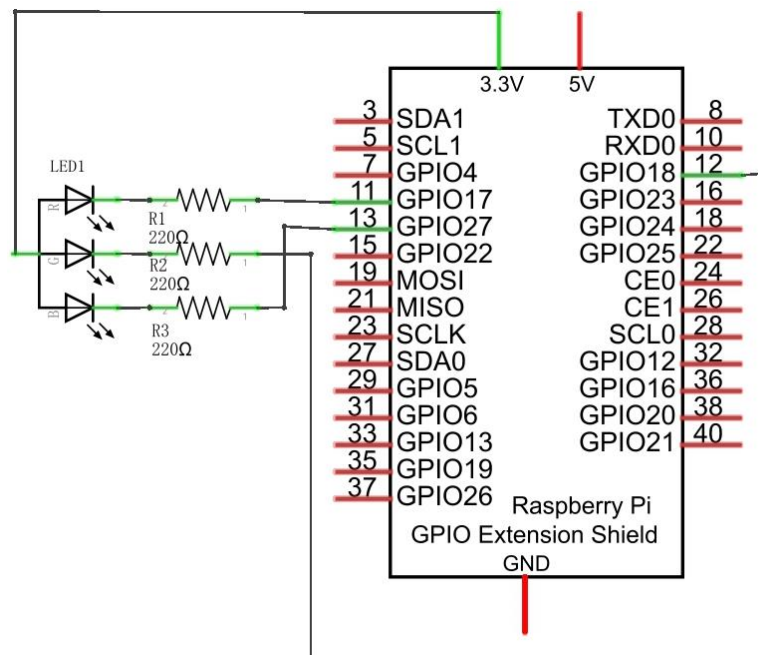In this chapter, we will learn how to use RGBLED.

Project 4.1 ColorfulLED

This project will make a ColorfulLED, namely, use Processing to control the color of RGBLED.
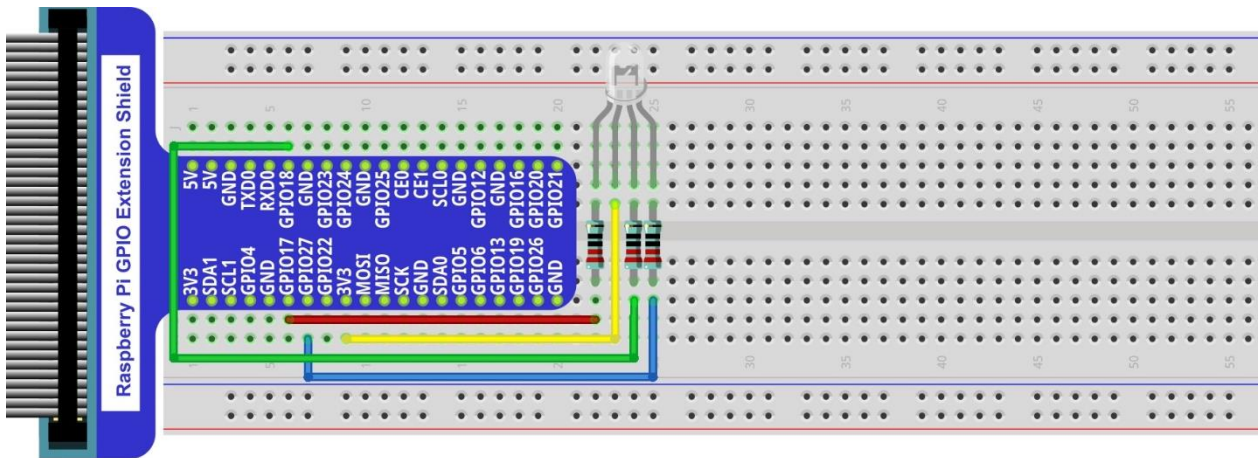
Component List

Raspberry Pi x1	RGBLED x1	Resistor 220Ω x3
GPIO Extension Board & Wire x1 BreadBoard x1		
Jumper M/M x4 		

Circuit

Schematic diagram

| Hardware connection |
| --- |
| |

Sketch

Sketch 4.1.1 ColorfulLED

First observe the running result of the sketch, and then analyze the code.

1. Use Processing to open the file Sketch_03_1_1_BreathingLED.

```
processing
~/Freenove_Kit/Processing/Sketches/Sketch_04_1_1_ColorfulLED/Sketch_04_1_1_Colo
rfulLED.pde
```

2. Click on "RUN" to run the code.

After the program is executed, RGBLED is under off state. And in Display Window, the pattern used to simulate LED is in black. Red, Green and Blue progress are in 0. By using mouse to click on and drag any progress bar, you can set the PWM duty cycle of color channels, and then RGBLED used in the circuit will show corresponding color. At the same time, the pattern in Display Window will show the same color.
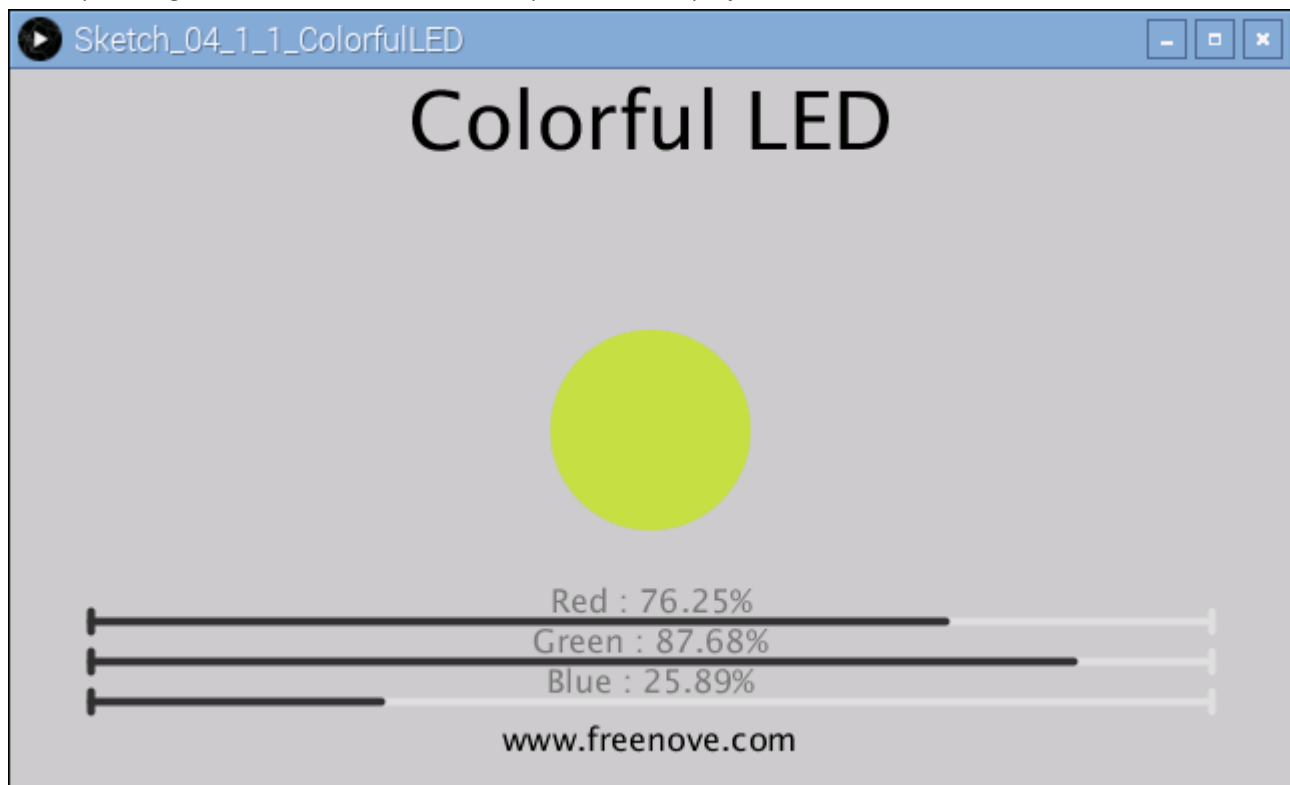
This project contains a lot of code files, and the core code is contained in the file Sketch_04_1_1_ColorfulLED. The other files only contain some custom classes.

The following is program code:

```
1    import processing.io.*;
2
3    int bluePin = 17;     //blue Pin
4    int greenPin = 18;  //green Pin
5    int redPin = 27;     //red Pin
6    int borderSize = 40;  //picture border size
7    //Create a PWM pin,initialize the duty cycle and period
8    SOFTPWM pRed = new SOFTPWM(redPin, 100, 100);
9    SOFTPWM pGreen = new SOFTPWM(greenPin, 100, 100);
10   SOFTPWM pBlue = new SOFTPWM(bluePin, 100, 100);
11   //instantiate three ProgressBar Object
12   ProgressBar rBar, gBar, bBar;
13   boolean rMouse = false, gMouse = false, bMouse = false;
14   void setup() {
15     size(640, 360);  //display window size
16     strokeWeight(4);  //stroke Weight
17     //define the ProgressBar length
18     int barLength = width - 2*borderSize;
19     //Create ProgressBar Object
20     rBar = new ProgressBar(borderSize, height - 85, barLength);
21     gBar = new ProgressBar(borderSize, height - 65, barLength);
22     bBar = new ProgressBar(borderSize, height - 45, barLength);
23     //Set ProgressBar's title
24     rBar.setTitle("Red");gBar.setTitle("Green");bBar.setTitle("Blue");
25   }
26
27   void draw() {
28     background(200);  //A white background
29     titleAndSiteInfo();  //title and Site infomation
30
31     fill(rBar.progress*255, gBar.progress*255, bBar.progress*255);  //cycle color
32     ellipse(width/2, height/2, 100, 100);  //show cycle
33
34     rBar.create();  //Show progressBar
35     gBar.create();
36     bBar.create();
37   }
38
39   void mousePressed() {
40     if ( (mouseY< rBar.y+5) && (mouseY>rBar.y-5) ) {
41       rMouse = true;
42     } else if ( (mouseY< gBar.y+5) && (mouseY>gBar.y-5) ) {
43       gMouse = true;
```

```
44        } else if ( (mouseY< bBar.y+5) && (mouseY>bBar.y-5) ) {
45          bMouse = true;
46        }
47      }
48      void mouseReleased() {
49        rMouse = false;
50        bMouse = false;
51        gMouse = false;
52      }
53      void mouseDragged() {
54        int a = constrain(mouseX, borderSize, width - borderSize);
55        float t = map(a, borderSize, width - borderSize, 0.0, 1.0);
56        if (rMouse) {
57          pRed.softPwmWrite((int)(100-t*100)); //wirte the duty cycle according to t
58          rBar.setProgress(t);
59        } else if (gMouse) {
60          pGreen.softPwmWrite((int)(100-t*100)); //wirte the duty cycle according to t
61          gBar.setProgress(t);
62        } else if (bMouse) {
63          pBlue.softPwmWrite((int)(100-t*100)); //wirte the duty cycle according to t
64          bBar.setProgress(t);
65        }
66      }
67
68      void titleAndSiteInfo() {
69        fill(0);
70        textAlign(CENTER);    //set the text centered
71        textSize(40);         //set text size
72        text("Colorful LED", width / 2, 40);    //title
73        textSize(16);
74        text("www.freenove.com", width / 2, height - 20);    //site
75      }
```

In the code, first create three PWM pins and three progress bars to control RGBLED.

```
SOFTPWM pRed = new SOFTPWM(redPin, 100, 100);
SOFTPWM pGreen = new SOFTPWM(greenPin, 100, 100);
SOFTPWM pBlue = new SOFTPWM(bluePin, 100, 100);
//instantiate three ProgressBar Object
ProgressBar rBar, gBar, bBar;
```

And then in function setup(), define position and length of progress bar ccording to the size of Display Window, and set the name of each progress bar.

```
void setup() {
    size(640, 360);  //display window size
```

```
        strokeWeight(4);  //stroke Weight
        //define the ProgressBar length
        int barLength = width - 2*borderSize;
        //Create ProgressBar Object
        rBar = new ProgressBar(borderSize, height - 85, barLength);
        gBar = new ProgressBar(borderSize, height - 65, barLength);
        bBar = new ProgressBar(borderSize, height - 45, barLength);
        //Set ProgressBar's title
        rBar.setTitle("Red");gBar.setTitle("Green");bBar.setTitle("Blue");
    }
```

In function draw(), first set background, header and other basic information. Then draw a circle and set its color according to the duty cycle of three channel of RGB. Finally draw three progress bars.

```
    void draw() {
        background(200);  //A white background
        titleAndSiteInfo();  //title and Site infomation

        fill(rBar.progress*255, gBar.progress*255, bBar.progress*255);  //cycle color
        ellipse(width/2, height/2, 100, 100);  //show cycle

        rBar.create();  //Show progressBar
        gBar.create();
        bBar.create();
    }
```

System function mousePressed(), mouseReleased() and mouseDragged() are used to determine whether the mouse drag the progress bar and set the schedule. If the mouse button is pressed in a progress bar, then the mousePressed () the progress of a flag *Mouse is set to true, mouseDragged (mouseX), in the mapping progress value set at the same time, the progress of the corresponding schedule and PWM. When the mouse is released, empty all the flags in the mouseReleased ().

```
    void mousePressed() {
        if ( (mouseY< rBar.y+5) && (mouseY>rBar.y-5) ) {
            rMouse = true;
        } else if ( (mouseY< gBar.y+5) && (mouseY>gBar.y-5) ) {
            gMouse = true;
        } else if ( (mouseY< bBar.y+5) && (mouseY>bBar.y-5) ) {
            bMouse = true;
        }
    }
    void mouseReleased() {
        rMouse = false;
        bMouse = false;
        gMouse = false;
    }
```

```
void mouseDragged() {
  int a = constrain(mouseX, borderSize, width - borderSize);
  float t = map(a, borderSize, width - borderSize, 0.0, 1.0);
  if (rMouse) {
    pRed.softPwmWrite((int)(100-t*100)); //wirte the duty cycle according to t
    rBar.setProgress(t);
  } else if (gMouse) {
    pGreen.softPwmWrite((int)(100-t*100)); //wirte the duty cycle according to t
    gBar.setProgress(t);
  } else if (bMouse) {
    pBlue.softPwmWrite((int)(100-t*100)); //wirte the duty cycle according to t
    bBar.setProgress(t);
  }
}
```

About class ProgressBar:

class ProgressBar
This is a custom class that is used to create a progress bar.
`public ProgressBar(int ix, int iy, int barlen)`
Constructor used create ProgressBar, the parameters for coordinates X , Y and length of ProgressBar.
`public void setTitle(String str)`
Used to set the name of progress bar, which will be displayed in middle of the progress bar.
`public void setProgress(float pgress)`
Used to set the progress of progress bar. The parameter: 0<pgress<1.0.
`public void create() & public void create(float pgress)`
Used to draw progress bar.

Chapter 5 Buzzer

In this chapter we will learn how to use buzzer.

Project 5.1 ActiveBuzzer

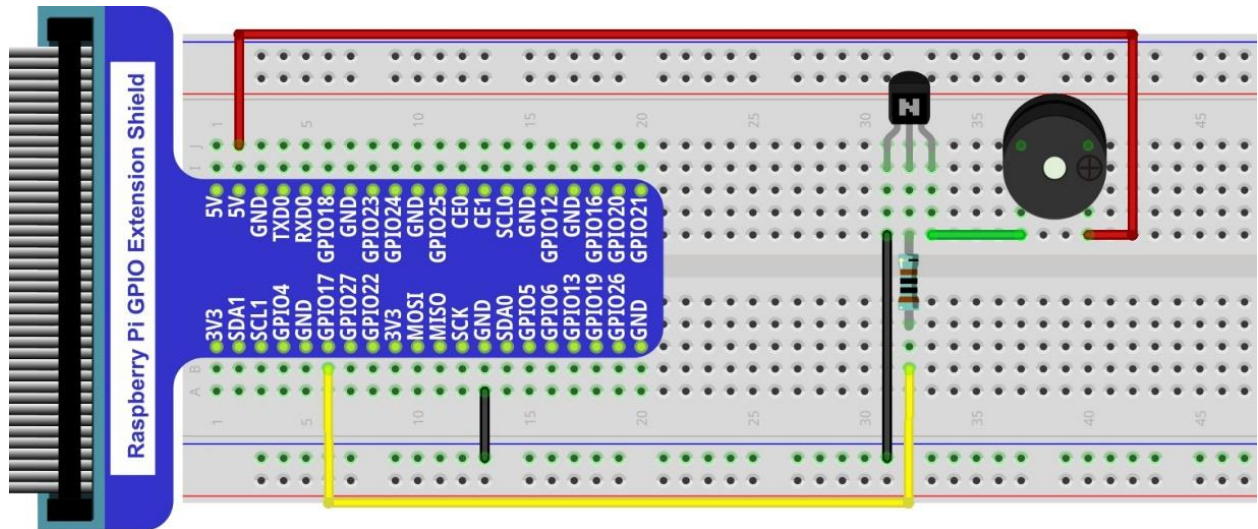This project will use the mouse to control an active buzzer.

Component List

Raspberry Pi x1 GPIO Extension Board & Wire x1 BreadBoard x1		Jumper M/M x7			
NPN transistorx1	Active buzzer x1	Push button x1	Resistor 1kΩ x1	Resistor 10kΩ x2

Circuit
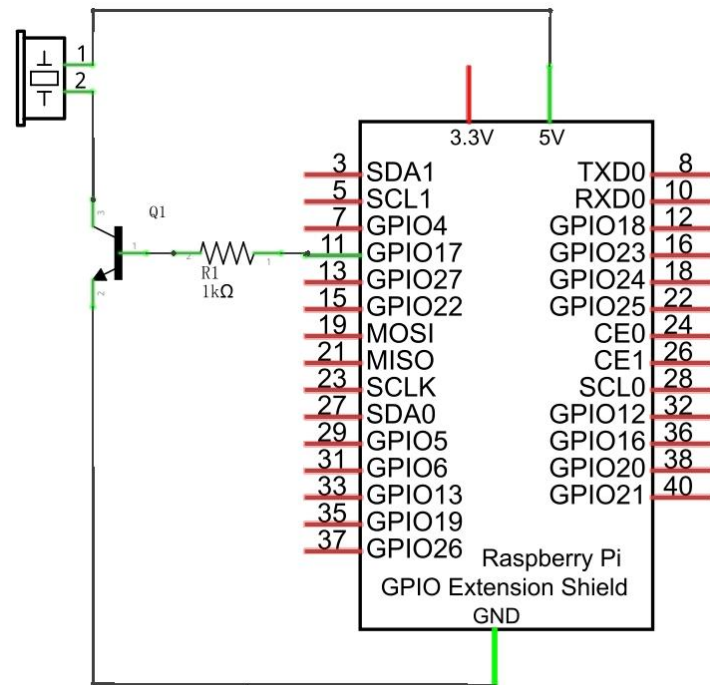
Schematic diagram
Hardware connection

Note: in this circuit, the power supply for buzzer is 5V, and pull-up resistor of the button connected to the power 3.3V. The buzzer can work when connected to power 3.3V, but it will reduce the loudness.

Sketch

Sketch 5.1.1 ActiveBuzzer

First observe the running result of the sketch, and then analyze the code.

1. Use Processing to open the file Sketch_05_1_1_ActiveBuzzer.

```
processing
~/Freenove_Kit/Processing/Sketches/Sketch_05_1_1_ActiveBuzzer/Sketch_05_1_1_Act
iveBuzzer.pde
```

2. Click on "RUN" to run the code.

After the program is executed, use the mouse to click on any position of the Display Window, then Active Buzzer begins to sound and arc graphics(Schematic of sounding) will appear next to the buzzer pattern in Display Window. Click the mouse again, then Active Buzzer stops sounding and arc graphics disappeare.

The following is program code:

```
import processing.io.*;

int buzzerPin = 17;
boolean buzzerState = false;
void setup() {
  size(640, 360);
  GPIO.pinMode(buzzerPin, GPIO.OUTPUT);
}
```

```
void draw() {
  background(255);
  titleAndSiteInfo();     //title and site infomation
  drawBuzzer();           //buzzer img
  if (buzzerState) {
    GPIO.digitalWrite(buzzerPin, GPIO.HIGH);
    drawArc();            //Sounds waves img
  } else {
    GPIO.digitalWrite(buzzerPin, GPIO.LOW);
  }
}

void mouseClicked() { //if the mouse Clicked
  buzzerState = !buzzerState;  //Change the buzzer State
}
void drawBuzzer() {
  strokeWeight(1);
  fill(0);
  ellipse(width/2, height/2, 50, 50);
  fill(255);
  ellipse(width/2, height/2, 10, 10);
}
void drawArc() {
  noFill();
  strokeWeight(8);
  for (int i=0; i<3; i++) {
    arc(width/2, height/2, 100*(1+i), 100*(1+i), -PI/4, PI/4, OPEN);
  }
}
void titleAndSiteInfo() {
  fill(0);
  textAlign(CENTER);      //set the text centered
  textSize(40);           //set text size
  text("Active Buzzer", width / 2, 40);     //title
  textSize(16);
  text("www.freenove.com", width / 2, height - 20);    //site
}
```

Code in this project is based on similar logic with the previous "MouseLED". And the difference is that this project needs to draw the buzzer pattern and arc graphics after the buzzer sounding.

App 1 Snake Game

In this chapter, we will play a classic game, snake.

App 1.1 Snake Game

Now, let's create and experience our own game.
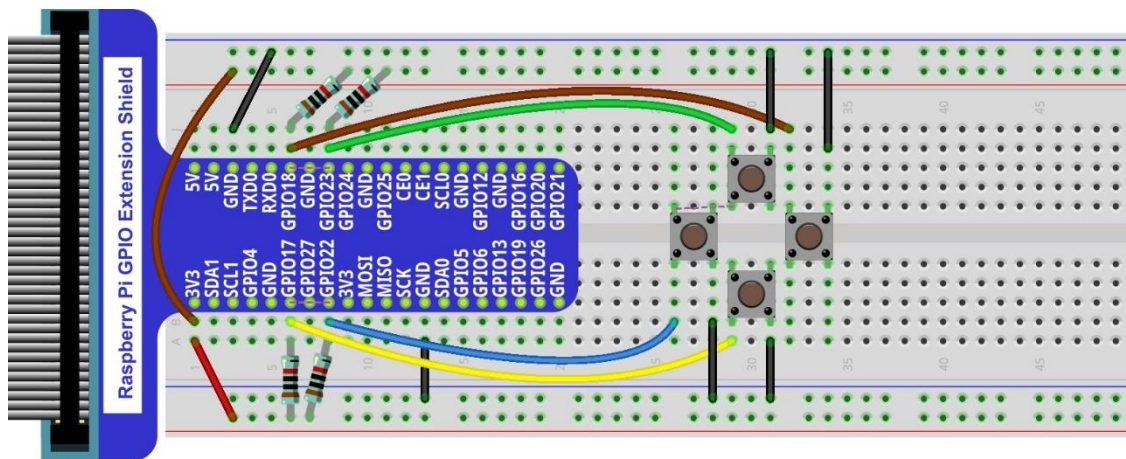
Component List

Raspberry Pi x1 GPIO Extension Board & Wire x1 BreadBoard x1	Resistor 10KΩ x4	Push button x4
Jumper M/M x12		

Circuit

Schematic diagram
Hardware connection

App

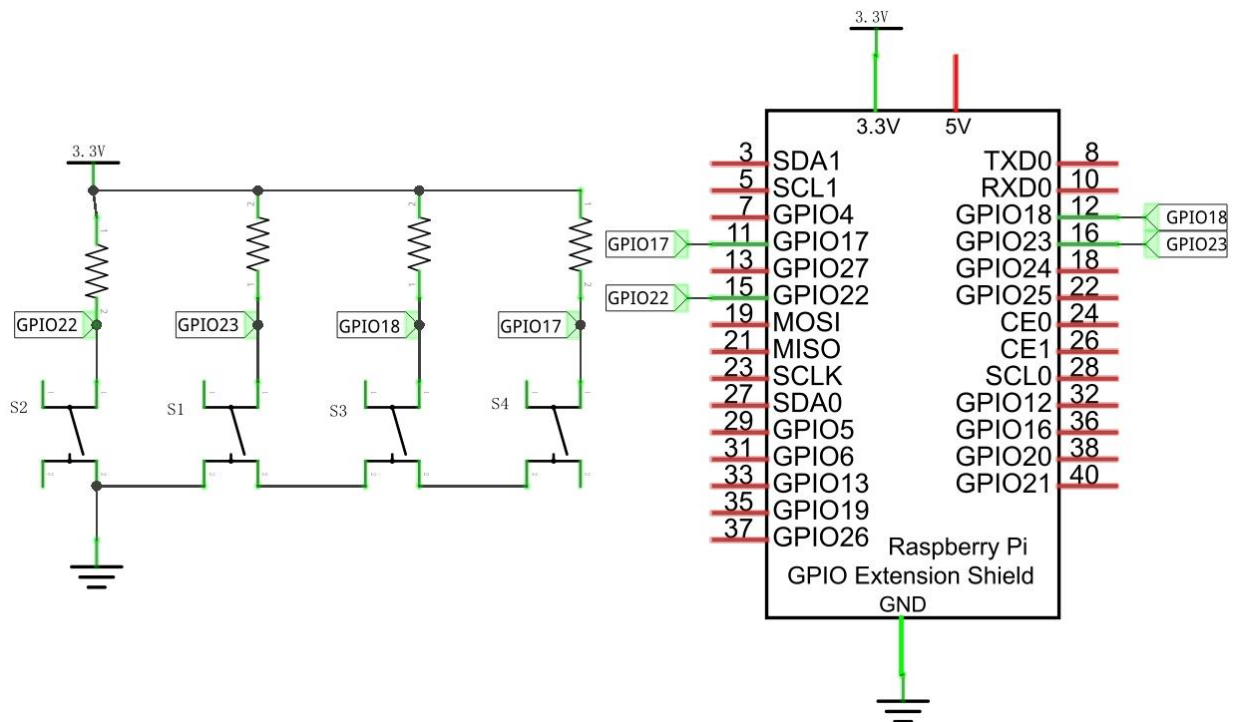1. Use Processing to open the file App_01_1_1_SnakeGame.

```
processing
~/Freenove_Kit/Processing/Apps/App_01_1_1_SnakeGame/App_01_1_1_SnakeGame.pde
```

2. Click on "RUN" to run the code.

After the program is executed, Display Window displays as below.
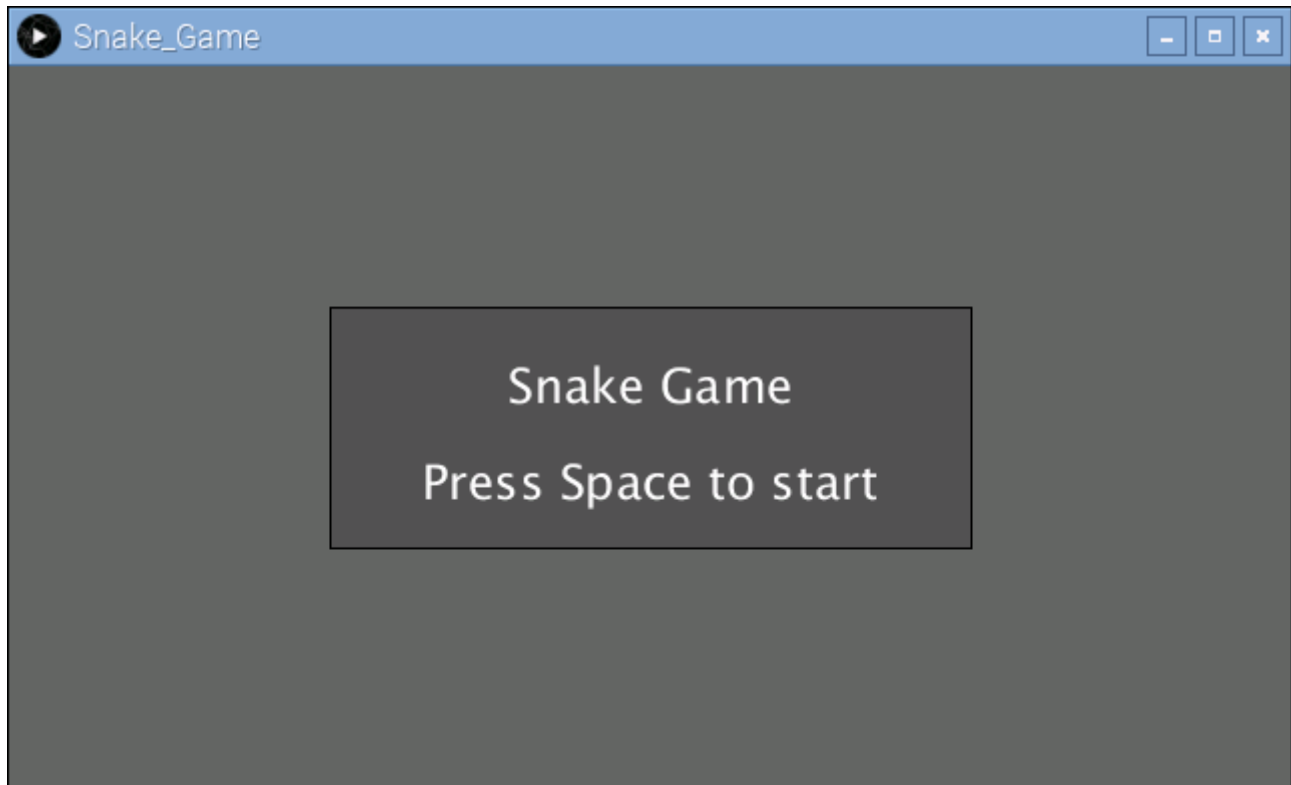
Pressing the space can start the game:
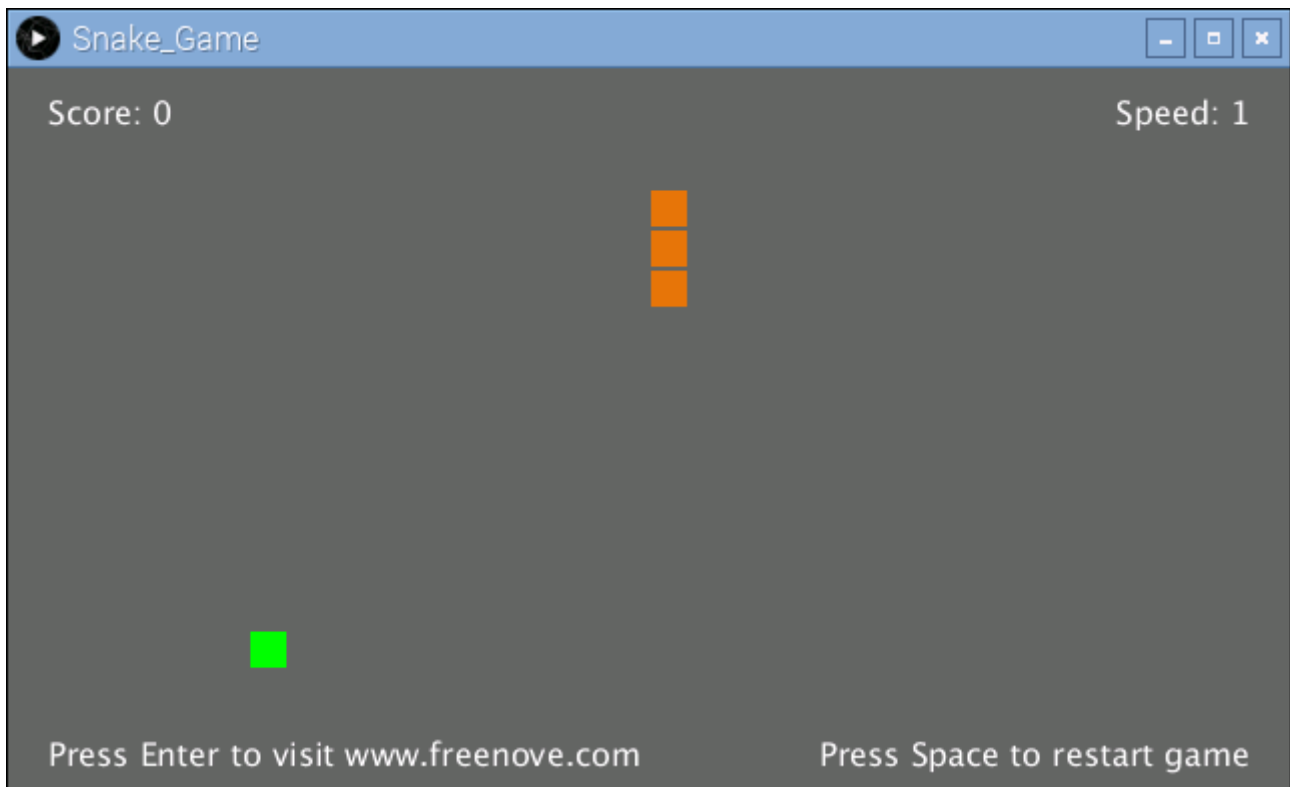
You can control the movement direction of the snake through the four buttons in circuit or four arrow keys on the keyboard. The rules are the same as the classic Snake game:
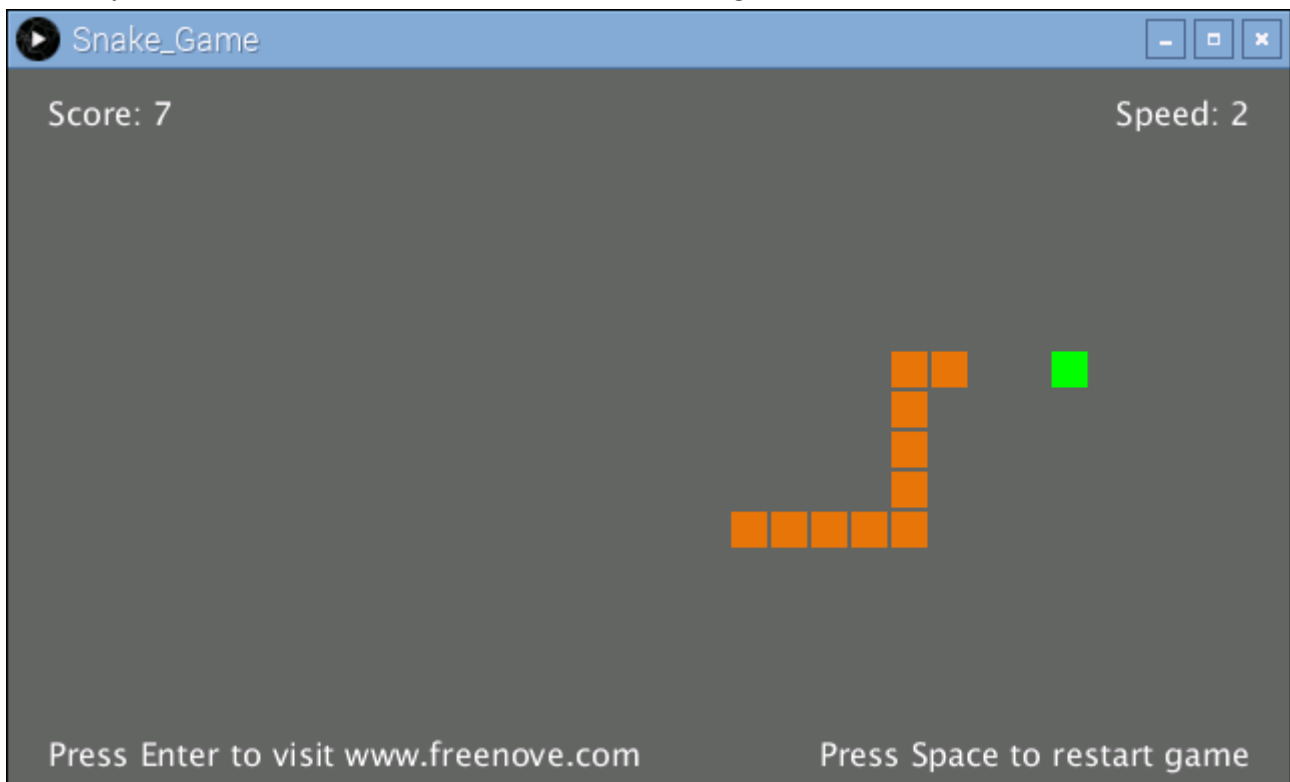
When game is over, pressing the space can restart the game:

You can restart the game by pressing the space bar at any time during the game.

App 2 Tetris Game

In this chapter, we will play a game, tetris game.

App 2.1 Tetris Game

Now, let's create and experience our own game.

Component List

Raspberry Pi x1 GPIO Extension Board & Wire x1 BreadBoard x1	Resistor 10KΩ x4	Push button x4
Jumper M/M x12		

Circuit

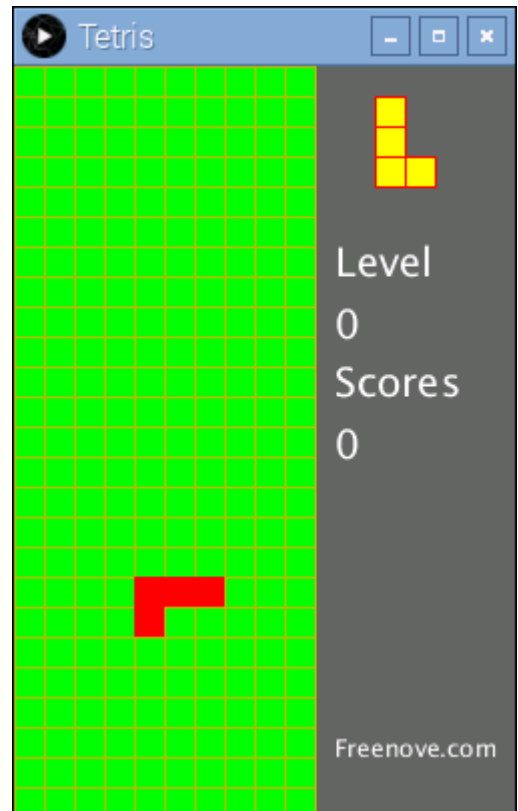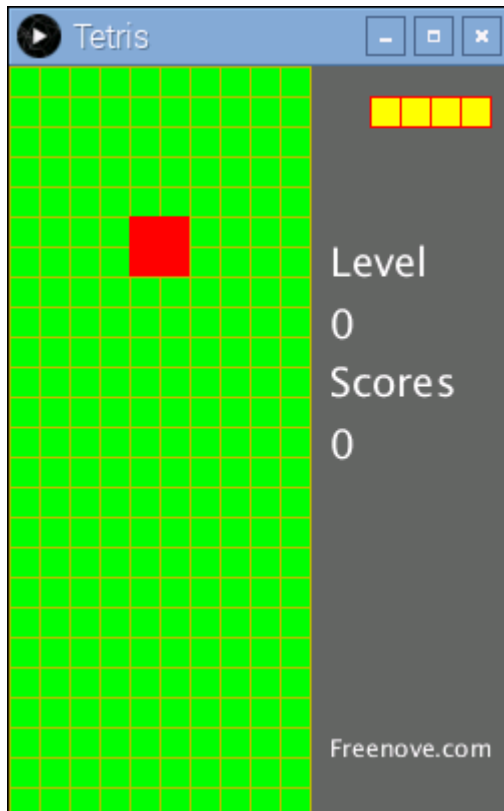| Schematic diagram |
|---|
| |
| Hardware connection |
| |

App

1. Use Processing to open the file App_02_1_1_TetrisGame.

```
processing
~/Freenove_Kit/Processing/Apps/App_02_1_1_TetrisGame/App_02_1_1_TetrisGame.pde
```

2. Click on "RUN" to run the code.

After the program is executed, Display Window displays as below.

The left and right button in the circuit can control the moving of the falling block to left or right. And the button below can accelerate falling of the block. The button above is used for rotating of the block. Four direction keys on keboard can also be used to play the game.
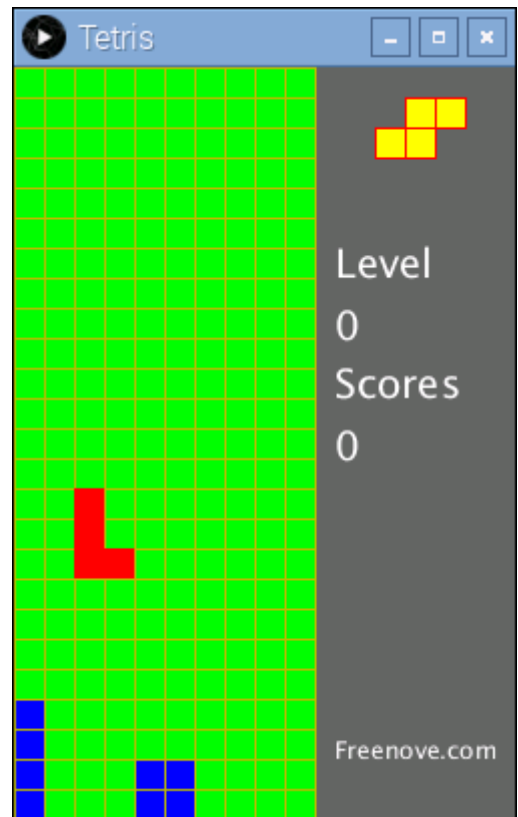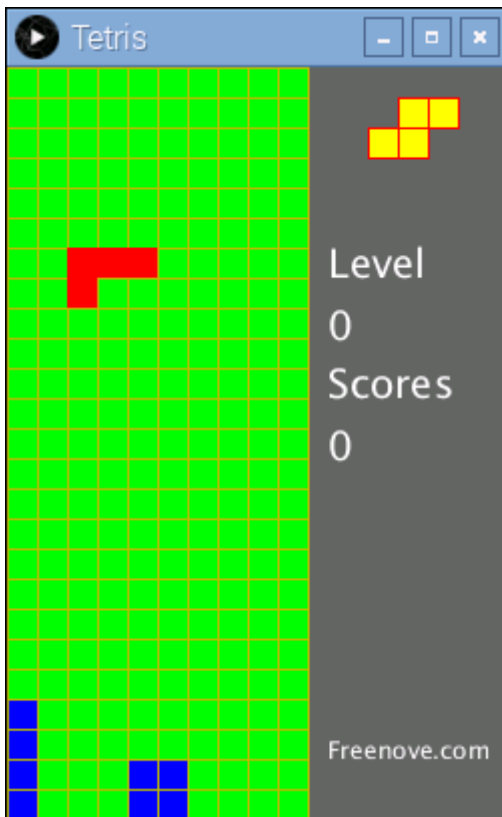
In the process of game, pressing the space bar on the keyboard can pause the game. The right side of the Display Window shows the next upcoming block, the current game speed and the current score. The more lines you eliminate once, the higher the scores. If you eliminate one line once, you will get 10 points. If you eliminate 4 lines once, you will get 70 points.
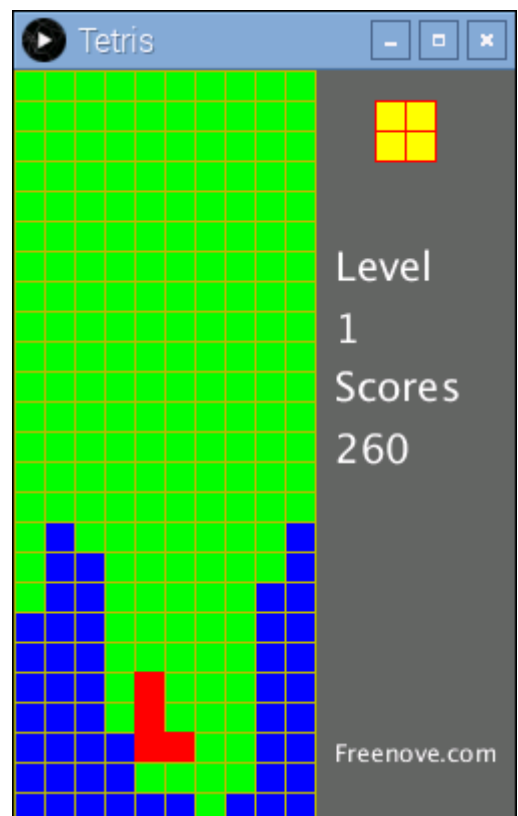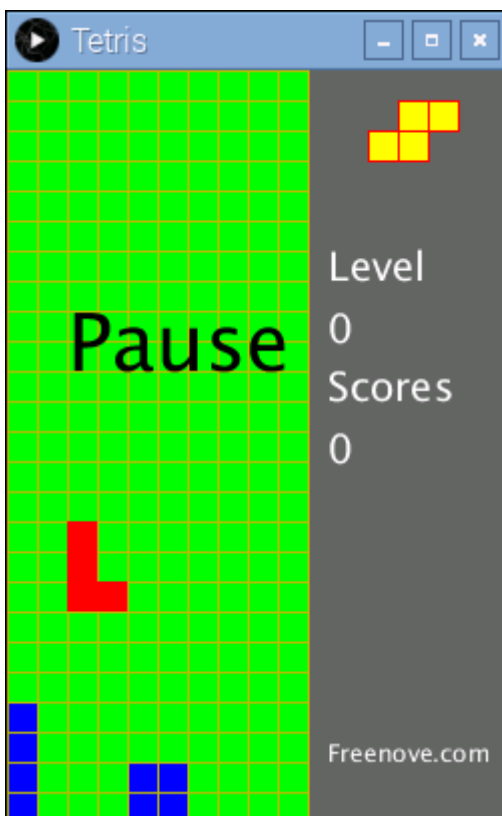
When the blocks are beyond the screen, the game is over. After the game is over, press the space bar to start a new game.
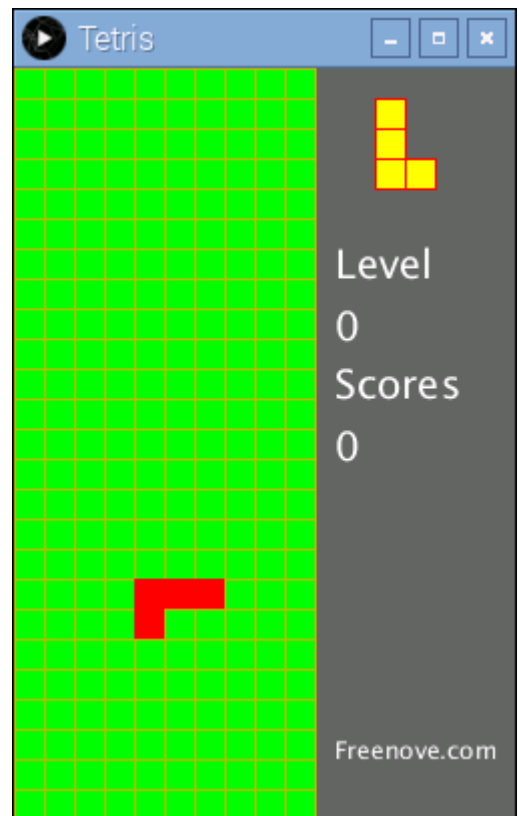
What's next?

Thanks for your reading.

This tutorial is all over here. If you find any mistakes, missions or you have other ideas and questions about contents of this tutorial 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.