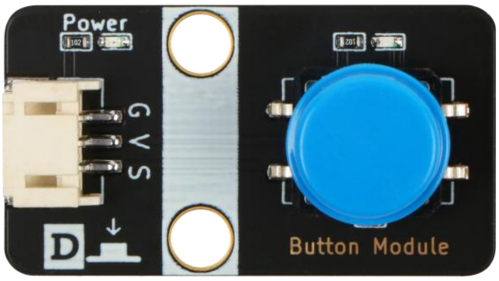
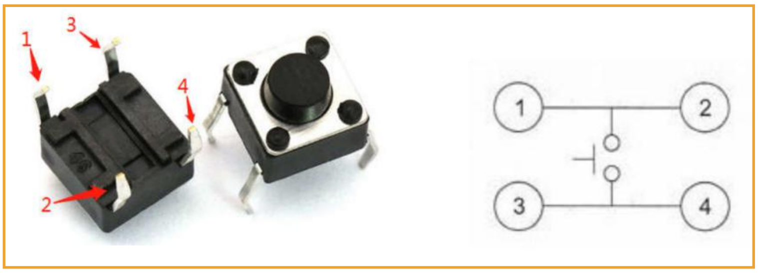
Key Module Experiment

Module Introduction

The key is also known as the light touch switch. When used, the pressure is applied to the switch operation direction to meet the operation force conditions, and the switch is closed and connected. When the pressure is removed, the switch will be disconnected. The internal structure of the switch is realized by the force change of the metal spring piece. As shown in the following four pin circuit diagram, we can clearly find that under normal circumstances, pins 1 and 2, 3 and 4 of the key are connected; When we press the button, the four feet are connected to each other; when we release the button, the four feet return to normal state. The digital I/O port means input and output interface. In the previous LED lamp experiments, we only use the OUTPUT function of GPIO. Now let's try to use the INPUT function of I / O in Arduino, that is to read the output value from the external device in this experiment. We use buttons and LED lights to complete the experiment using INPUT and OUTPUT as a combination. Our key module outputs low level when the key is pressed and high level when released.



Physical Picture



Four Leg Circuit Diagram

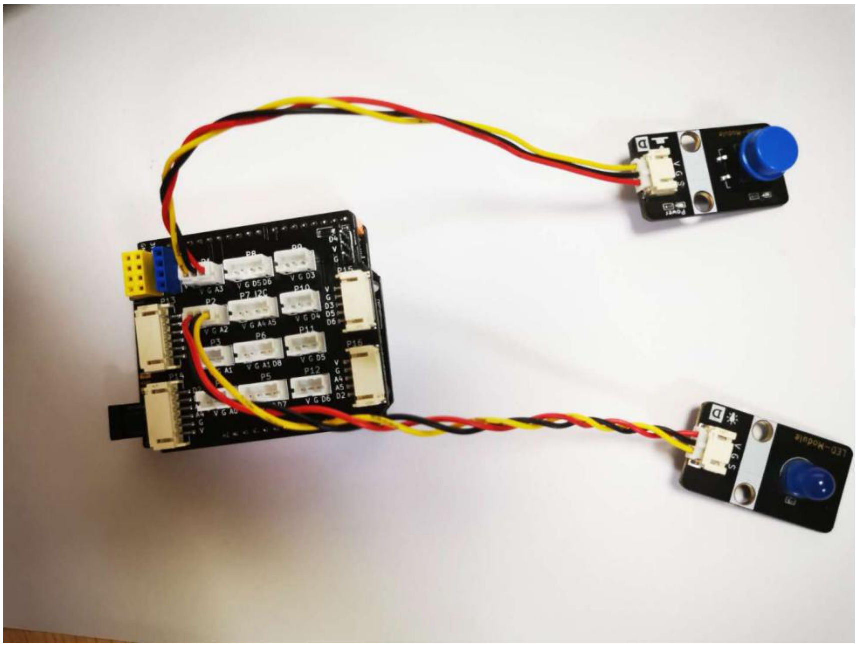
Purpose of the Experiment

Control the LED light on and off through ph2.0 key module, and understand the working principle of the key module.

Device List

- BLE-UNO Main Board: 1
- Expansion Board of PH2.0 Sensor: 1
- USB Data Wire: 1
- LED Light Module: 1
- Key Module: 1
- 3pin Wire Jumper: 2

Physical Wiring Diagram



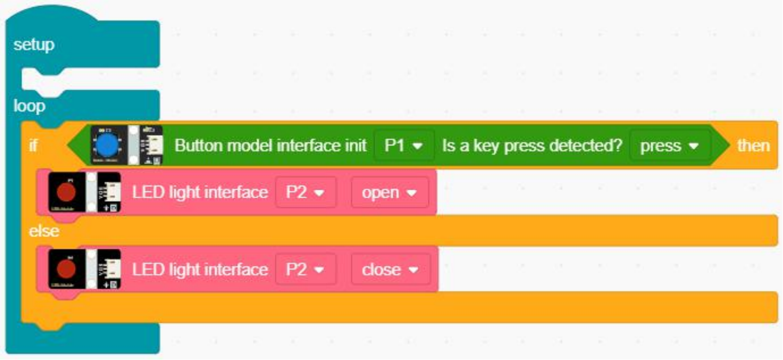
Arduino Program Code

```
int led_out = A2 ;    //GPIO A2    LED pin
int keypad_pin = A3; //GPIO A3 key pin
int value;
void setup()
{
    pinMode(led_out,OUTPUT);                       // init led pin output
    pinMode(keypad_pin,INPUT);                 // init key pin input
}
void loop()
{
        value = digitalRead(keypad_pin);     // read key pad pin vaule
        if( value == LOW)
        {
            digitalWrite(led_out,HIGH);         // if key value is down    turn on LED
        }
        else
        {
            digitalWrite(led_out,LOW);        // if key value is down    turn off LED
        }
}
```

MagicBlock Program

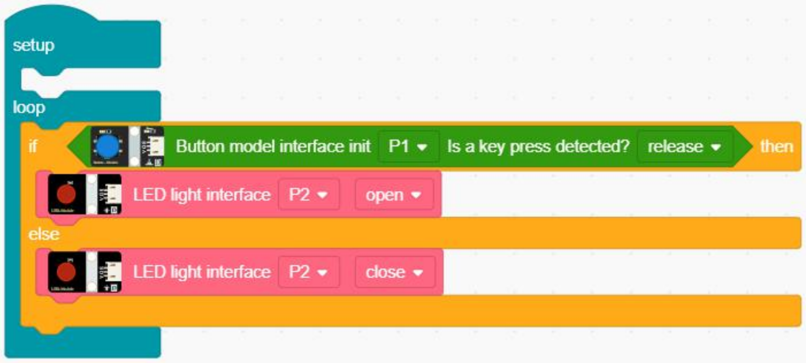
MagicBlock program has three situations for the keys:

1. When the key is pressed, as shown in the figure



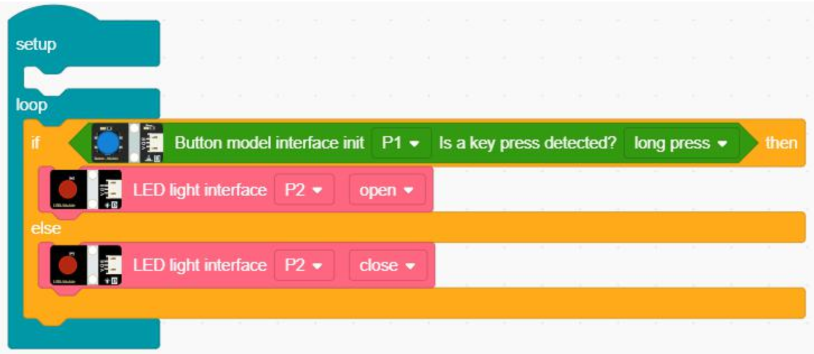
```
setup
loop
  if  Button model interface init  P1 ▾  Is a key press detected?  press ▾  then
    LED light interface  P2 ▾  open ▾
  else
    LED light interface  P2 ▾  close ▾
```

2. When the key is released, as shown in the figure



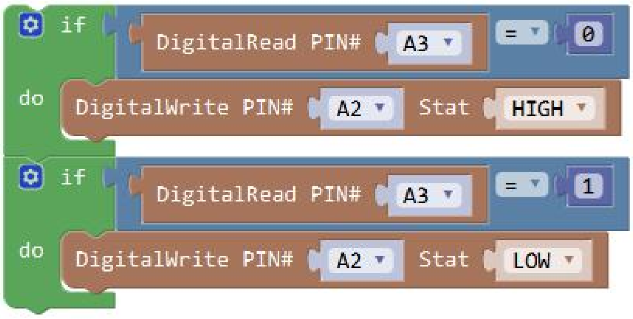
```
setup
loop
  if  Button model interface init  P1 ▾  Is a key press detected?  release ▾  then
    LED light interface  P2 ▾  open ▾
  else
    LED light interface  P2 ▾  close ▾
```

3. When the key is pressed for long time, as shown in the figure



```
setup
loop
  if  Button model interface init  P1 ▾  Is a key press detected?  long press ▾  then
    LED light interface  P2 ▾  open ▾
  else
    LED light interface  P2 ▾  close ▾
```

Mixly Program



```
if  DigitalRead PIN# A3 = 0
do  DigitalWrite PIN# A2 Stat HIGH
if  DigitalRead PIN# A3 = 1
do  DigitalWrite PIN# A2 Stat LOW
```

Experimental Conclusion

After the device is connected with the cable, burn the above program to the motherboard, power on the motherboard, press the key to make the LED light up, and release it to turn off. By pressing and releasing the button, the signal port that triggers the button outputs the high and low level. By judging the high and low level status of the signal port, the LED light is on and off. When our key module is pressed, it will output low level, otherwise it will output high level to achieve the purpose.