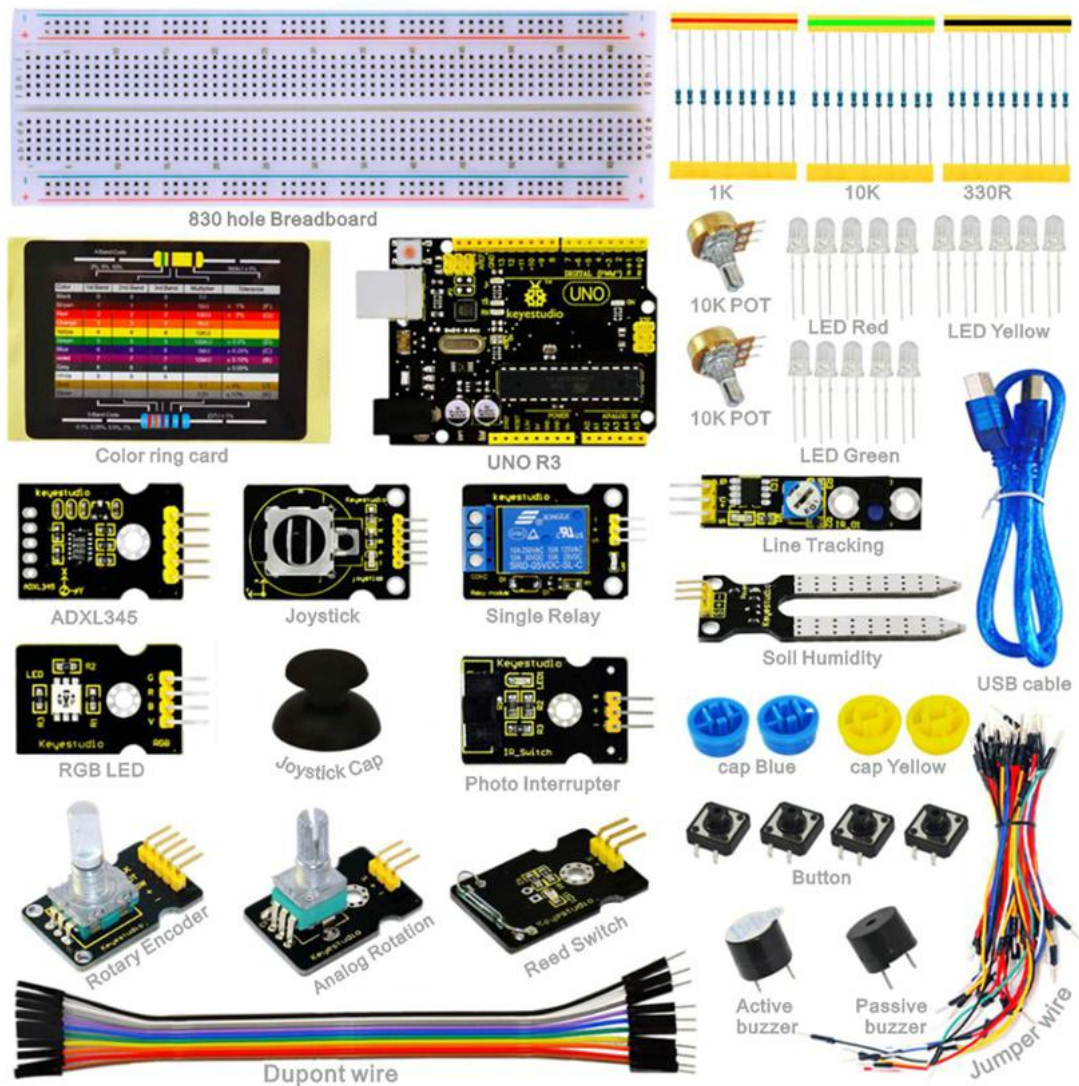


# keyestudio

## keyestudio Sensor Kit for ARDUINO starters- K4



# keyestudio

---

## Catalog

1. Introduction.....	1
2. Component list.....	1
3. Project details:.....	5
Project 1: Hello World.....	5
Project 2: LED blinking.....	7
Project 3: PWM.....	9
Project 4: Traffic light.....	14
Project 5: LED chasing effect.....	16
Project 6: Button-controlled LED.....	18
Project 7: Responder experiment.....	20
Project 8: Active buzzer.....	24
Project 9: Passive buzzer.....	26
Project 10: Joystick module.....	28
Project 11: Photo interrupter module.....	30
Project 12: 5V Relay Module.....	32
Project 13: ADXL345 Three Axis Acceleration Module.....	34
Project 14: Rotary Encoder module.....	38
Project 15: Analog Rotation Sensor.....	41
Project 16: RGB LED module.....	43
Project 17: keyestudio Reed Switch Module.....	45
Project 18: Soil Humidity Sensor.....	47
Project 19: Line Tracking Sensor.....	49

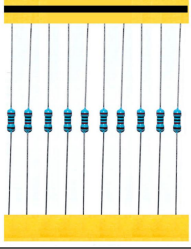

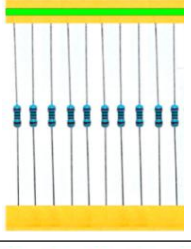
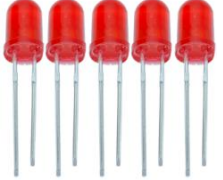
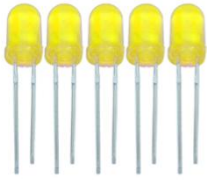
# keystudio

---

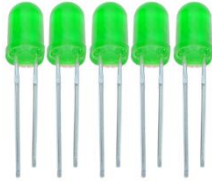






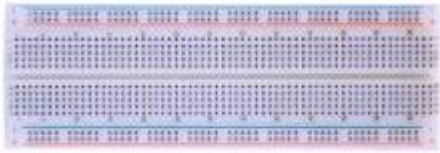


## 1. Introduction

This keystudio Sensor Kit is an Arduino starter learning kit developed by Keyes. We provide detailed tutorials for each project or sensor module, including connection diagrams and sample codes, with which you will find it easy for you to complete every experiments. Besides, you can also find video tutorials of this kit on our official website.


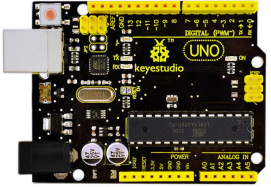





## 2. Component list

No.	Product Name	Quantity	Picture
1	Resistor 330R	10	
2	Resistor 1K	10	
3	Resistor 10K	10	
4	LED Red	5	
5	LED Yellow	5	








# keyestudio

6	LED Green	5	
7	Potentiometer 10K	2	
8	Passive buzzer	1	
9	Active buzzer	1	
10	Button	4	
11	Button cap Blue	2	
12	Button cap Yellow	2	
13	830-hole Breadboard	1	
14	Jumper wire	1*65	
15	M-F Dupont wire 20cm	10	

# keyestudio

16	Resistor color code card	1	
17	KEYESTUDIO UNO	1	
18	USB cable 0.5m	1	
19	Joystick Module	1	
20	Photo Interrupter Module	1	
21	Single Relay Module	1	
22	ADXL345 Three Axis Acceleration Module	1	

# keyestudio

23	Rotary Encoder Module	1	
24	Analog Rotation Sensor module	1	
25	RGB LED module	1	
26	Reed Switch Module	1	
27	Soil Humidity Sensor module	1	
28	Line Tracking Sensor module	1	
29	Component box	1	

# keystudio

---

## 3. Project details:

### Project 1: Hello World

#### Introduction

As for starters, we will begin with something simple. In this project, you only need an Arduino and a USB cable to start the "Hello World!" experiment. This is a communication test of your Arduino and PC, also a primer project for you to have your first try of the Arduino world!

#### Hardware required

Arduino board \*1

USB cable \*1

#### Sample program

After installing driver for Arduino, let's open Arduino software and compile code that enables Arduino to print "Hello World!" under your instruction. Of course, you can compile code for Arduino to continuously echo "Hello World!" without instruction. A simple If () statement will do the instruction trick. With the onboard LED connected to pin 13, we can instruct the LED to blink first when Arduino gets an instruction and then print "Hello World!".

```
////////////////////////////////////
```

```
int val;//define variable val
```

```
int ledpin=13;// define digital interface 13
```

```
void setup()
```

```
{
```

```
    Serial.begin(9600);// set the baud rate at 9600 to match the software set up. When connected to a specific device, (e.g. bluetooth), the baud rate needs to be the same with it.
```

```
    pinMode(ledpin,OUTPUT);// initialize digital pin 13 as output. When using I/O ports on an Arduino, this kind of set up is always needed.
```

```
}
```

```
void loop()
```

```
{
```

```
    val=Serial.read();// read the instruction or character from PC to Arduino, and assign them to Val.
```

```
    if(val=='R')// determine if the instruction or character received is "R".
```

```
    { // if it's "R",
```

```
        digitalWrite(ledpin,HIGH);// set the LED on digital pin 13 on.
```

```
        delay(500);
```

```
        digitalWrite(ledpin,LOW);// set the LED on digital pin 13 off.
```

```
        delay(500);
```

```
        Serial.println("Hello World!");// display"Hello World! "string.
```



# keystudio

```
}  
}  
////////////////////////////////////
```

## Result

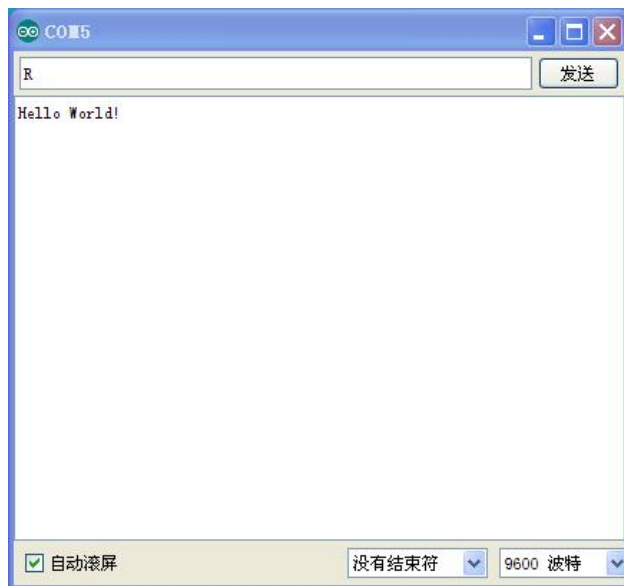


Click serial port monitor, Input R, LED 13 will blink once, PC will receive information from Arduino: Hello World



# keyestudio

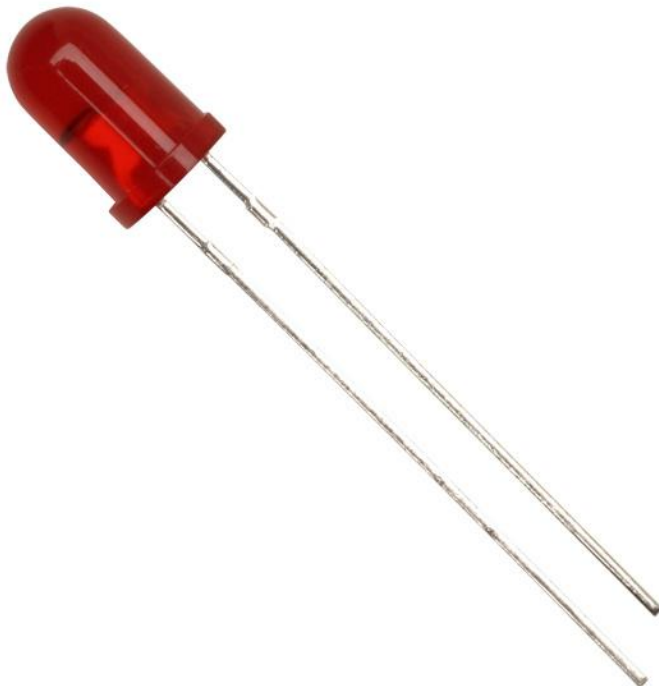
---



After you choosing the right port, the experiment should be easy for you!

\*\*\*\*\*

## Project 2: LED blinking



### Introduction

Blinking LED experiment is quite simple. In the "Hello World!" program, we have come across LED. This time, we are going to connect an LED to one of the digital pins rather than using

# keystudio

---

LED13, which is soldered to the board. Except an Arduino and an USB cable, we will need extra parts as below:

## Hardware required

Red M5 LED\*1

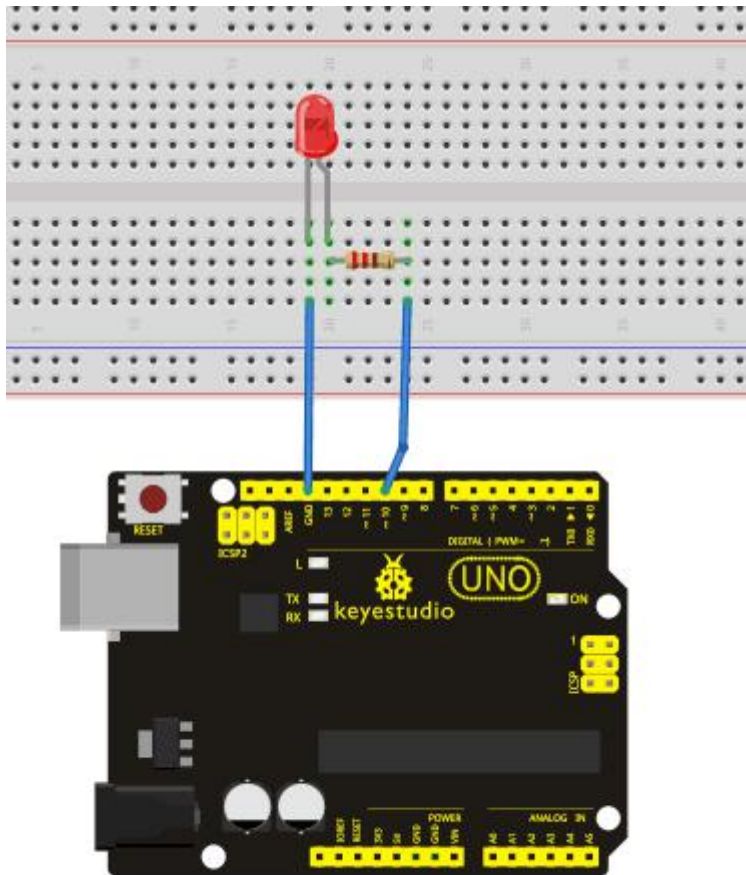
220 $\Omega$  resistor\*1

Breadboard\*1

Breadboard jumper wires

## Circuit connection

We follow below diagram from the experimental schematic link. Here we use digital pin 10. We connect LED to a 220 ohm resistor to avoid high current damaging the LED.



# keystudio

---

## Sample program

```
////////////////////////////////////
int ledPin = 10; // define digital pin 10.
void setup()
{
  pinMode(ledPin, OUTPUT); // define pin with LED connected as output.
}
void loop()
{
  digitalWrite(ledPin, HIGH); // set the LED on.
  delay(1000); // wait for a second.
  digitalWrite(ledPin, LOW); // set the LED off.
  delay(1000); // wait for a second
}
////////////////////////////////////
```

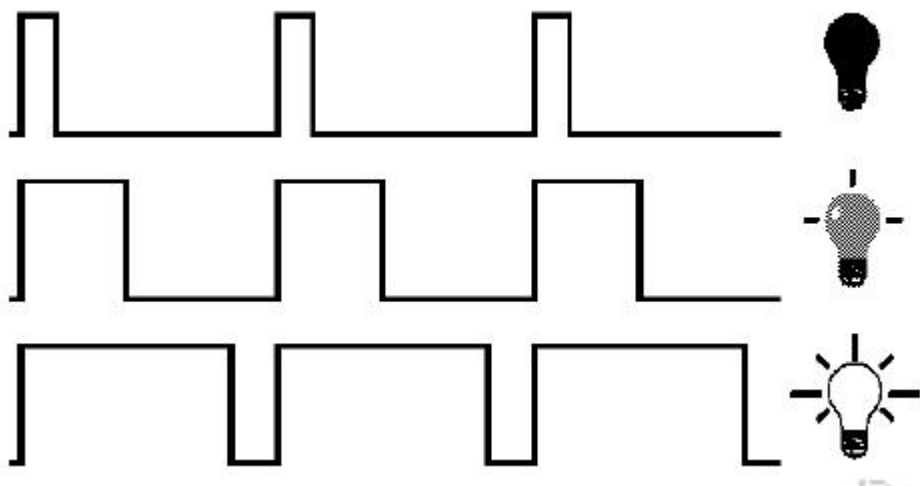
## Result

After downloading this program, in the experiment, you will see the LED connected to pin 10 turning on and off, with an interval approximately one second.

The LED blinking experiment is now complete. Thank you!

\*\*\*\*\*

## Project 3: PWM

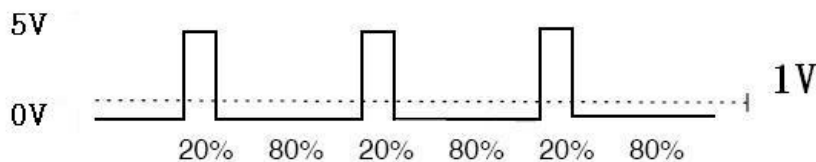
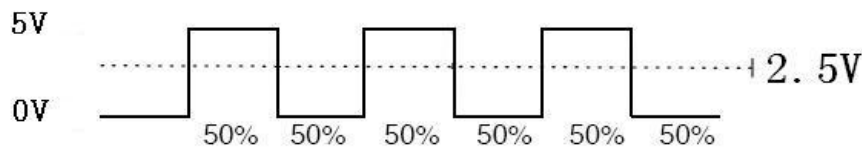
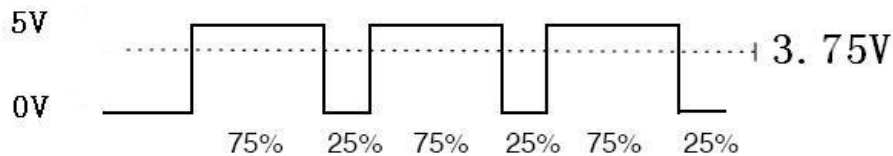


## Introduction

PWM, short for Pulse Width Modulation, is a technique used to encode analog signal level into digital ones. A computer cannot output analog voltage but only digital voltage values such as 0V or 5V. So we use a high resolution counter to encode a specific analog signal level by modulating

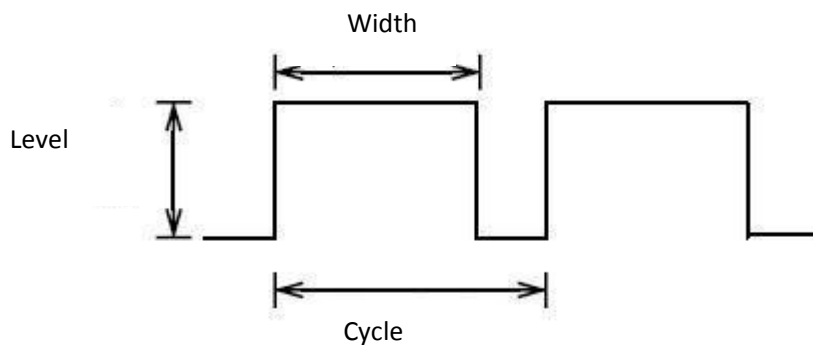
# keyestudio

the duty cycle of PMW. The PWM signal is also digitalized because in any given moment, fully on DC power supply is either 5V (ON), or 0V (OFF). The voltage or current is fed to the analog load (the device that uses the power) by repeated pulse sequence being ON or OFF. Being on, the current is fed to the load; being off, it's not. With adequate bandwidth, any analog value can be encoded using PWM. The output voltage value is calculated via the on and off time. Output voltage = (turn on time/pulse time) \* maximum voltage value



PWM has many applications: lamp brightness regulating, motor speed regulating, sound making, etc.

The following are the three basic parameters of PMW:



1. The amplitude of pulse width (minimum / maximum)
2. The pulse period (The reciprocal of pulse frequency in 1 second)
3. The voltage level (such as: 0V-5V)

There are 6 PMW interfaces on Arduino, namely digital pin 3, 5, 6, 9, 10, and 11. In previous experiments, we have done "button-controlled LED", using digital signal to control digital pin, also one about potentiometer. This time, we will use a potentiometer to control the brightness of the LED.

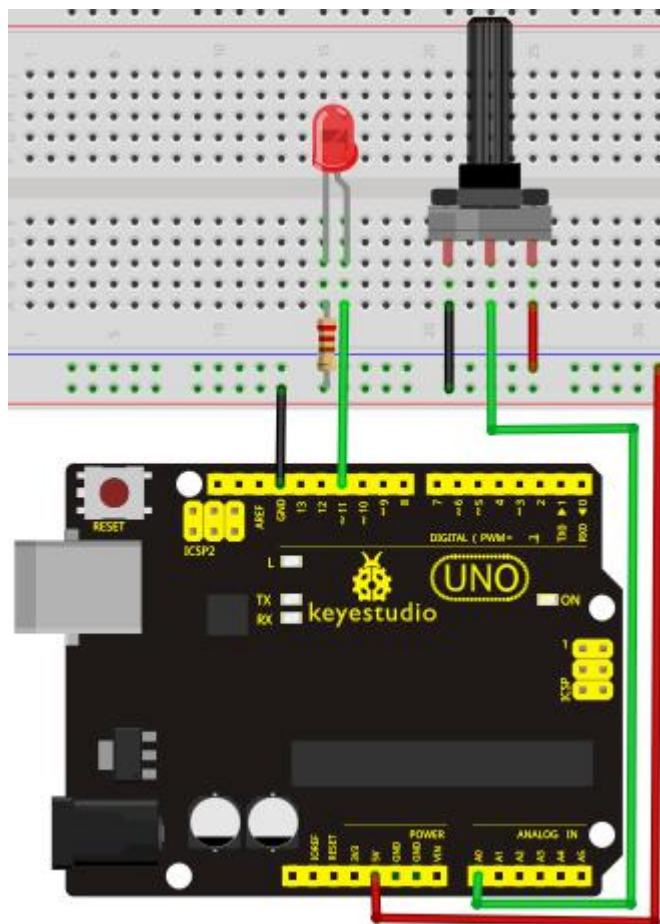
# keyestudio

## Hardware required

Potentiometer module\*1  
Red M5 LED\*1  
220Ω resistor  
Breadboard\*1  
Breadboard jumper wires

## Circuit connection

The input of potentiometer is analog, so we connect it to analog port, and LED to PWM port. Different PWM signal can regulate the brightness of the LED.



## Sample program

In the program compiling process, we will use the `analogWrite` (PWM interface, analog value) function. In this experiment, we will read the analog value of the potentiometer and assign the value to PWM port, so there will be corresponding change to the brightness of the LED. One final part will be displaying the analog value on the screen. You can consider this as the "analog value

# keyestudio

---

reading" project adding the PWM analog value assigning part. Below is a sample program for your reference.

```
////////////////////////////////////
int potpin=0;// initialize analog pin 0
int ledpin=11;//initialize digital pin 11 (PWM output)
int val=0;// Temporarily store variables' value from the sensor
void setup()
{
  pinMode(ledpin,OUTPUT);// define digital pin 11 as "output"
  Serial.begin(9600);// set baud rate at 9600
  // attention: for analog ports, they are automatically set up as "input"
}
void loop()
{
  val=analogRead(potpin);// read the analog value from the sensor and assign it to val
  Serial.println(val);// display value of val
  analogWrite(ledpin,val/4);// turn on LED and set up brightness (maximum output of PWM is 255)
  delay(10);// wait for 0.01 second
}
////////////////////////////////////
```

# keyestudio



## Result

After downloading the program, when we rotate the potentiometer knob, we can see changes of the displaying value, also obvious change of the LED brightness on the breadboard.

\*\*\*\*\*



## Project 4: Traffic light



### Introduction

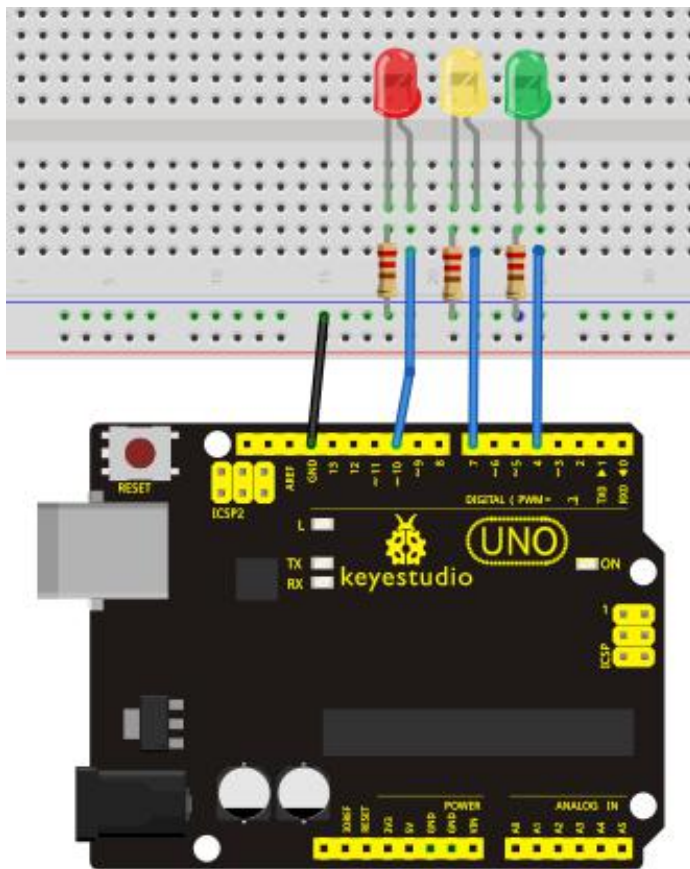
In the previous program, we have done the LED blinking experiment with one LED. Now, it's time to up the stakes and do a bit more complicated experiment-traffic lights. Actually, these two experiments are similar. While in this traffic lights experiment, we use 3 LEDs with different color other than 1 LED.

### Hardware required

- Arduino board \*1
- USB cable \*1
- Red M5 LED\*1
- Yellow M5 LED\*1
- Green M5 LED\*1
- 220Ω resistor \*3
- Breadboard\*1
- Breadboard jumper wires

### Circuit connection

# keyestudio



## Sample program

Since it is a simulation of traffic lights, the blinking time of each LED should be the same with those in traffic lights system. In this program, we use Arduino delay () function to control delay time, which is much simpler than C language.

```
////////////////////////////////////////
int redled =10; // initialize digital pin 8.
int yellowled =7; // initialize digital pin 7.
int greenled =4; // initialize digital pin 4.
void setup()
{
  pinMode(redled, OUTPUT); // set the pin with red LED as "output"
  pinMode(yellowled, OUTPUT); // set the pin with yellow LED as "output"
  pinMode(greenled, OUTPUT); // set the pin with green LED as "output"
}
void loop()
{
  digitalWrite(greenled, HIGH); /// turn on green LED
  delay(5000); // wait 5 seconds
  digitalWrite(greenled, LOW); // turn off green LED
```

# keystudio

---

```
for(int i=0;i<3;i++)// blinks for 3 times
{
  delay(500);// wait 0.5 second
  digitalWrite(yellowled, HIGH);// turn on yellow LED
  delay(500);// wait 0.5 second
  digitalWrite(yellowled, LOW);// turn off yellow LED
}
delay(500);// wait 0.5 second
digitalWrite(redled, HIGH);// turn on red LED
delay(5000);// wait 5 second
digitalWrite(redled, LOW);// turn off red LED
}
////////////////////////////////////
```

## Result

When the uploading process is completed, we can see traffic lights of our own design.

Note: this circuit design is very similar with the one in LED chase effect.

The green light will be on for 5 seconds, and then off., followed by the yellow light blinking for 3 times, and then the red light on for 5 seconds, forming a cycle. Cycle then repeats.

Experiment is now completed, thank you.

\*\*\*\*\*

## Project 5: LED chasing effect



## Introduction

# keyestudio

We often see billboards composed of colorful LEDs. They are constantly changing to form various effects. In this experiment, we compile a program to simulate chase effect.

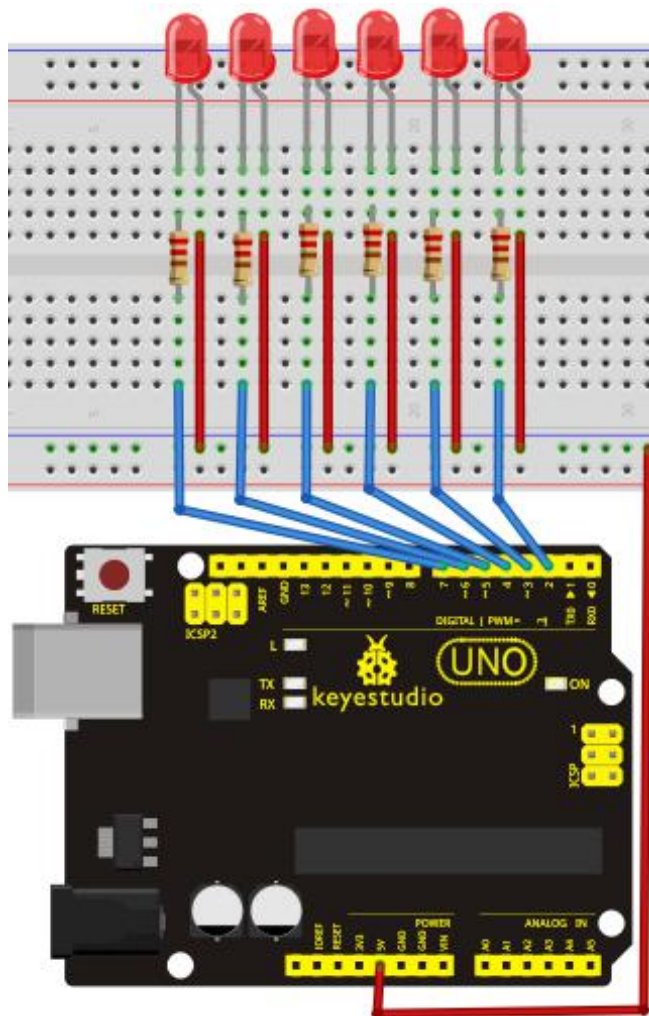
## Hardware required

Led \*6

220Ω resistor \*6

Breadboard jumper wires

## Circuit connection



## Sample program

```
////////////////////////////////////  
int BASE = 2 ; // the I/O pin for the first LED  
int NUM = 6; // number of LEDs  
  
void setup()  
{
```

# keyestudio

---

```
for (int i = BASE; i < BASE + NUM; i ++)  
{  
    pinMode(i, OUTPUT);    // set I/O pins as output  
}  
}  
  
void loop()  
{  
    for (int i = BASE; i < BASE + NUM; i ++)  
    {  
        digitalWrite(i, LOW);    // set I/O pins as “low”, turn off LEDs one by one.  
        delay(200);              // delay  
    }  
    for (int i = BASE; i < BASE + NUM; i ++)  
    {  
        digitalWrite(i, HIGH);   // set I/O pins as “high”, turn on LEDs one by one  
        delay(200);              // delay  
    }  
}  
////////////////////////////////////
```

## Result

You can see the LEDs blink by sequence.

\*\*\*\*\*

## Project 6: Button-controlled LED



# keyestudio

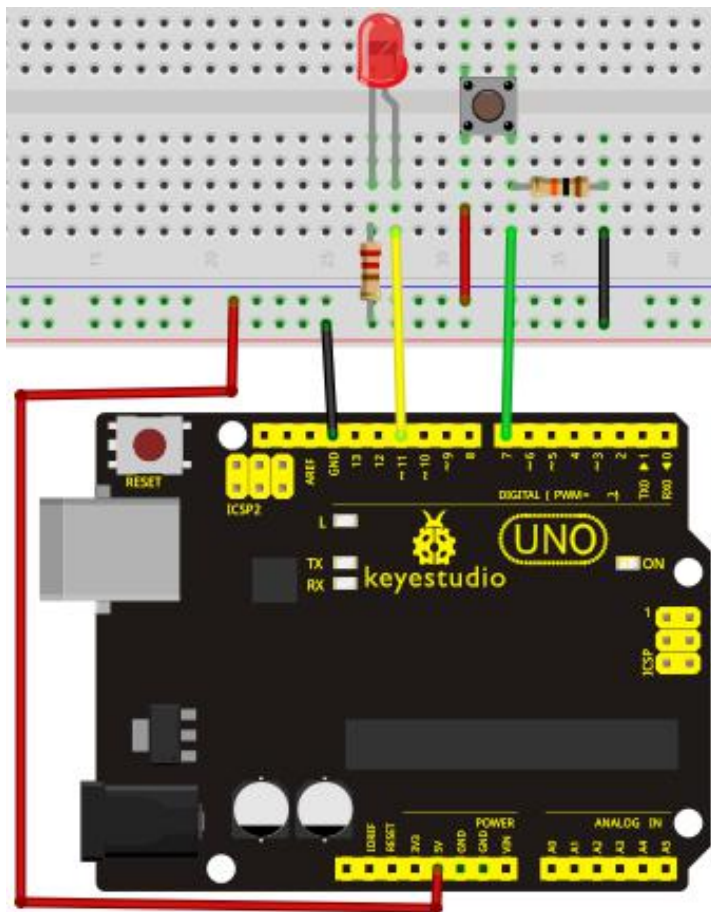
## Introduction

I/O port means interface for INPUT and OUTPUT. Up until now, we have only used its OUTPUT function. In this experiment, we will try to use the input function, which is to read the output value of device connecting to it. We use 1 button and 1 LED using both input and output to give you a better understanding of the I/O function. Button switches, familiar to most of us, are a switch value (digital value) component. When it's pressed, the circuit is in closed (conducting) state.

## Hardware required

Button switch\*1  
Red M5 LED\*1  
220Ω resistor\*1  
10KΩ resistor\*1  
Breadboard\*1  
Breadboard jumper wires

## Circuit connection



## Sample program



# keyestudio

---

Now, let's begin the compiling. When the button is pressed, the LED will be on. After the previous study, the coding should be easy for you. In this program, we add a statement of judgment. Here, we use an if () statement.

Arduino IDE is based on C language, so statements of C language such as while, switch etc. can certainly be used for Arduino program.

When we press the button, pin 7 will output high level. We can program pin 11 to output high level and turn on the LED. When pin 7 outputs low level, pin 11 also outputs low level and the LED remains off.

```
////////////////////////////////////
int ledpin=11;// initialize pin 11
int inpin=7;// initialize pin 7
int val;// define val
void setup()
{
  pinMode(ledpin,OUTPUT);// set LED pin as "output"
  pinMode(inpin,INPUT);// set button pin as "input"
}
void loop()
{
  val=digitalRead(inpin);// read the level value of pin 7 and assign it to val
  if(val==LOW)// check if the button is pressed, if yes, turn on the LED
  { digitalWrite(ledpin,LOW);}
  else
  { digitalWrite(ledpin,HIGH);}
}
////////////////////////////////////
```

## Result

When the button is pressed, LED is on, otherwise, LED remains off. After the above process, the button controlled LED experiment is completed. The simple principle of this experiment is widely used in a variety of circuit and electric appliances. You can easily come across it in your every day life. One typical example is when you press a certain key of your phone, the backlight will be on.

\*\*\*\*\*

## Project 7: Responder experiment

### Introduction

After completing all the previous experiments, we believe you will find this one easy. In this program, we have 3 buttons and a reset button controlling the corresponding 3 LEDs, using 7 digital I/O pins.

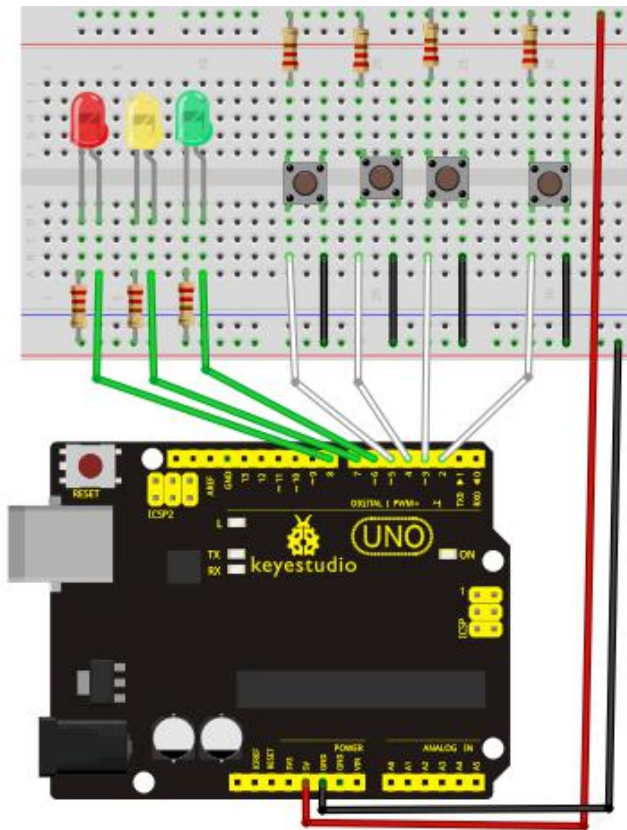


# keyestudio

## Hardware required

Button switch\*4  
Red M5 LED\*1  
Yellow M5 LED\*1  
Green M5 LED\*1  
220Ω resistor\*3  
10KΩ resistor\*4  
Breadboard\*1  
Breadboard jumper wires

## Circuit connection



## Sample program

```
//////////////////////////////////////////  
int redled=8;      // set red LED as “output”  
int yellowled=7;   // set yellow LED as “output”  
int greenled=6;    // set green LED as “output”
```

# keyestudio

---

```
int redpin=5;      // initialize pin for red button
int yellowpin=4;   // initialize pin for yellow button
int greenpin=3;    // initialize pin for green button
int restpin=2;     // initialize pin for reset button
int red;
int yellow;
int green;
void setup()
{
  pinMode(redled,OUTPUT);
  pinMode(yellowled,OUTPUT);
  pinMode(greenled,OUTPUT);
  pinMode(redpin,INPUT);
  pinMode(yellowpin,INPUT);
  pinMode(greenpin,INPUT);
}
void loop()  // repeatedly read pins for buttons
{
  red=digitalRead(redpin);
  yellow=digitalRead(yellowpin);
  green=digitalRead(greenpin);
  if(red==LOW)RED_YES();
  if(yellow==LOW)YELLOW_YES();
  if(green==LOW)GREEN_YES();
}

void RED_YES()// execute the code until red light is on; end cycle when reset button is pressed
{
  while(digitalRead(restpin)==1)
  {
    digitalWrite(redled,HIGH);
    digitalWrite(greenled,LOW);
    digitalWrite(yellowled,LOW);
  }
  clear_led();
}

void YELLOW_YES()// execute the code until yellow light is on; end cycle when reset button is pressed
{
  while(digitalRead(restpin)==1)
  {
    digitalWrite(redled,LOW);
    digitalWrite(greenled,LOW);
```

# keystudio

---

```
    digitalWrite(yellowled,HIGH);
  }
  clear_led();
}
void GREEN_YES()// execute the code until green light is on; end cycle when reset button is
pressed
{
  while(digitalRead(restpin)==1)
  {
    digitalWrite(redled,LOW);
    digitalWrite(greenled,HIGH);
    digitalWrite(yellowled,LOW);
  }
  clear_led();
}
void clear_led()// all LED off
{
  digitalWrite(redled,LOW);
  digitalWrite(greenled,LOW);
  digitalWrite(yellowled,LOW);
}
////////////////////////////////////
```

## Result

Whichever button is pressed first, the corresponding LED will be on!

Then press the REST button to reset.

After the above process, we have built our own simple responder.

\*\*\*\*\*

## Project 8: Active buzzer



### Introduction

Arduino enables us to make many interesting interactive projects, many of which we have done consists of a LED. They are light-related. While this time, the circuit will produce sound. The sound experiment is usually done with a buzzer or a speaker, while buzzer is simpler and easier to use. The buzzer we introduced here is a passive buzzer. It cannot be actuated by itself, but by external pulse frequencies. Different frequencies produce different sounds. We can use Arduino to code the melody of a song, which is actually fun and simple.

### Hardware required

Buzzer\*1

Key \*1

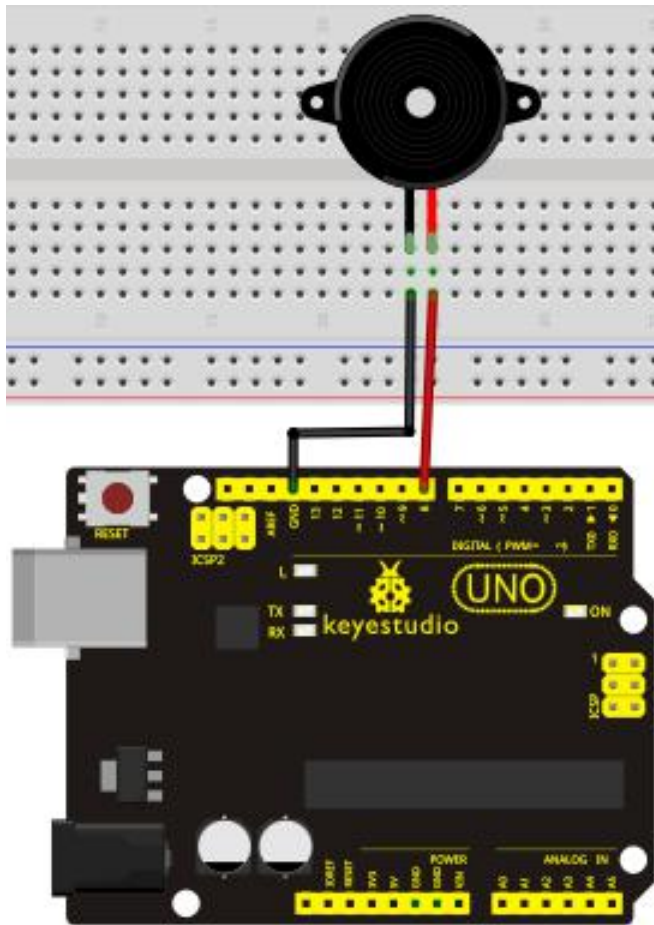
Breadboard\*1

Breadboard jumper wires

### Circuit connection

# keyestudio

---



When connecting the circuit, pay attention to the positive & the negative poles of the buzzer. In the photo, you can see there are red and black lines. When the circuit is finished, you can begin programming.

## Sample program

Program is simple. You control the buzzer by outputting high/low level.

```
////////////////////////////////////
int buzzer=8;// initialize digital IO pin that controls the buzzer
void setup()
{
    pinMode(buzzer,OUTPUT);// set pin mode as “output”
}
void loop()
{
    digitalWrite(buzzer, HIGH); // produce sound
}
////////////////////////////////////
```

# keystudio

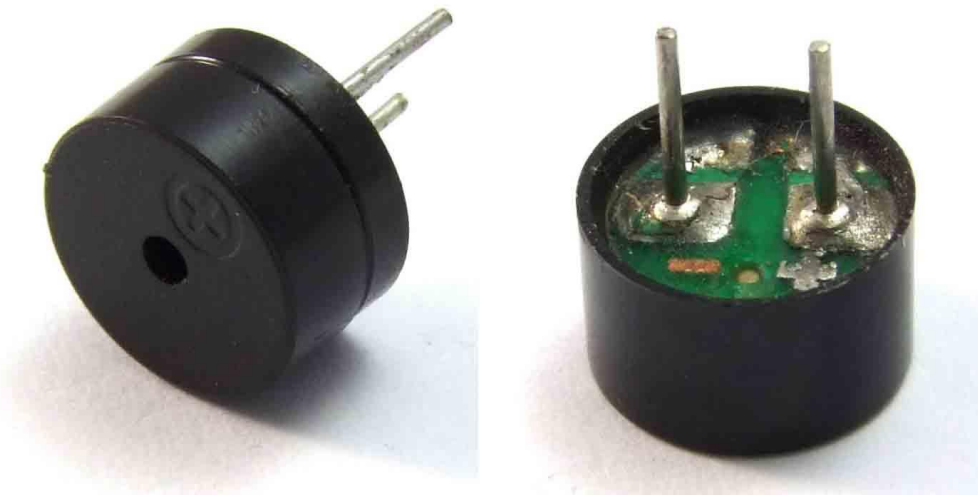
---

## Result

After downloading the program, the buzzer experiment is completed. You can see the buzzer is ringing.

\*\*\*\*\*

## Project 9: Passive buzzer



## Introduction

We can use Arduino to make many interactive works of which the most commonly used is acoustic-optic display. All the previous experiment has something to do with LED. However, the circuit in this experiment can produce sound. Normally, the experiment is done with a buzzer or a speaker while buzzer is simpler and easier to use. The buzzer we introduced here is a passive buzzer. It cannot be actuated by itself, but by external pulse frequencies. Different frequencies produce different sounds. We can use Arduino to code the melody of a song, which is actually quite fun and simple.

## Hardware required

Passive buzzer\*1

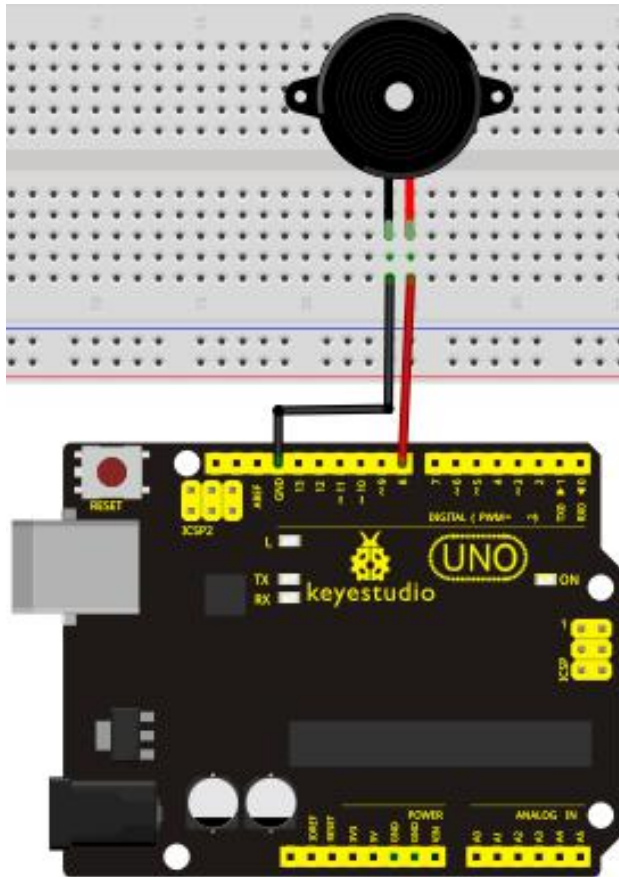
Key \*1

Breadboard\*1

Breadboard jumper wires

# keyestudio

---



## Sample program

```
////////////////////////////////////
int buzzer=8;// select digital IO pin for the buzzer
void setup()
{
  pinMode(buzzer,OUTPUT);// set digital IO pin pattern, OUTPUT to be output
}
void loop()
{ unsigned char i,j;//define variable
  while(1)
  { for(i=0;i<80;i++)// output a frequency sound
    { digitalWrite(buzzer,HIGH);// sound
      delay(1);//delay 1ms
      digitalWrite(buzzer,LOW);//not sound
      delay(1);//ms delay
    }
    for(i=0;i<100;i++)// output a frequency sound
    { digitalWrite(buzzer,HIGH);// sound
      digitalWrite(buzzer,LOW);//not sound
      delay(2);//2ms delay
```



# keystudio

---

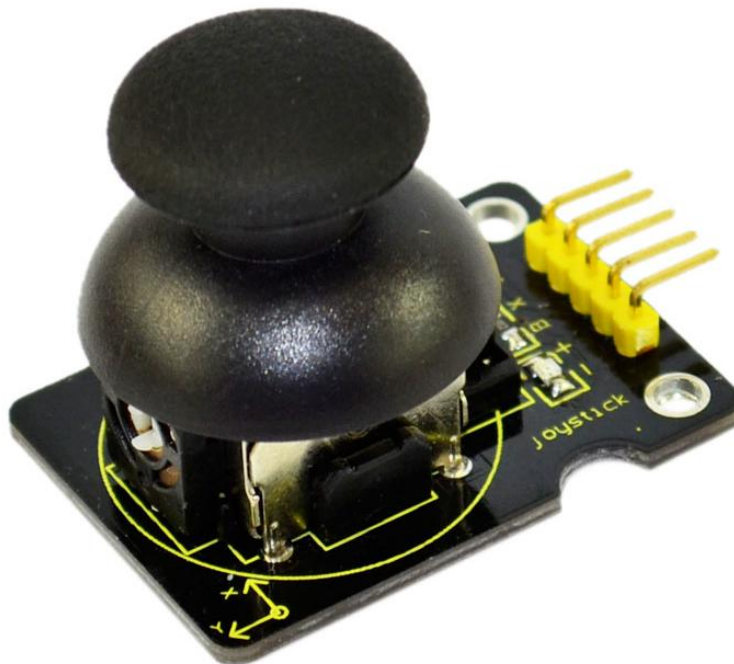
```
}  
}  
}
```

```
////////////////////////////////////
```

After downloading the program, buzzer experiment is finished.

```
*****
```

## Project 10: Joystick module



### Introduction

Lots of robot projects need joystick. This module provides an affordable solution. By simply connecting to two analog inputs, the robot is at your commands with X, Y control. It also has a switch that is connected to a digital pin. This joystick module can be easily connected to Arduino by IO Shield. This module is for Arduino(V5) with cables supplied.

### Specification

Supply Voltage: 3.3V to 5V

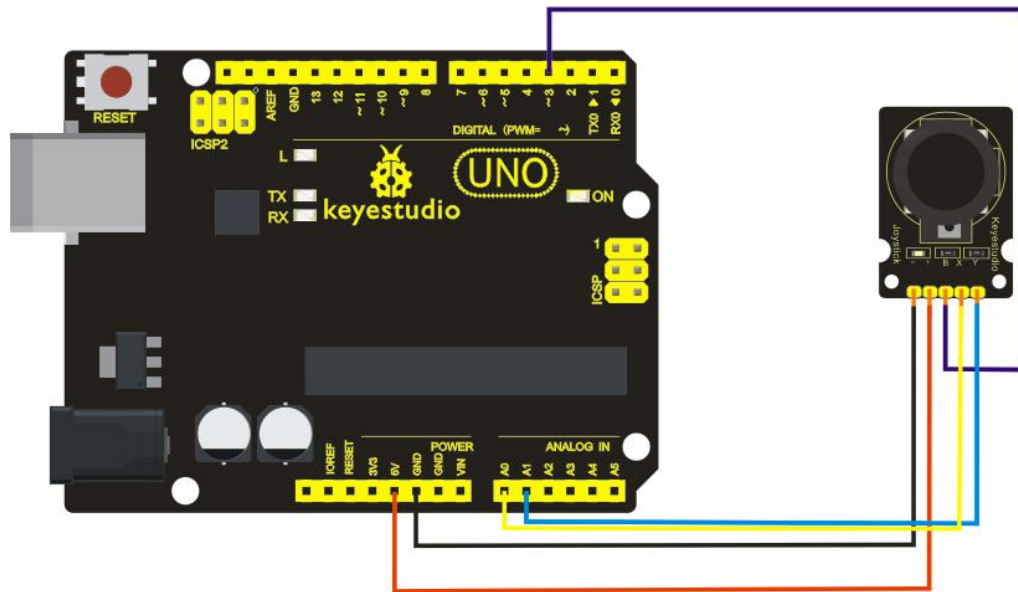
Interface: Analog x2, Digital x1

Size: 40\*28mm

Weight: 12g

# keyestudio

## Connection Diagram



## Sample Code

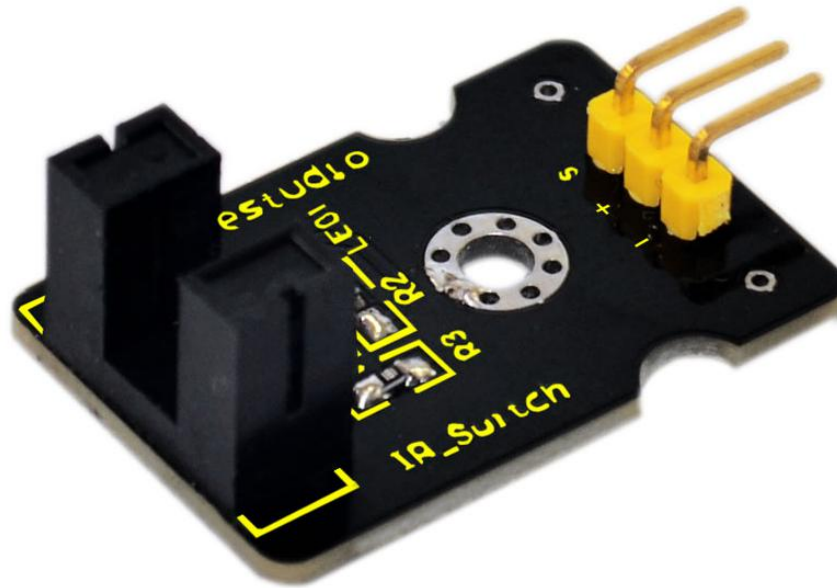
```
int JoyStick_X = 0; //x
int JoyStick_Y = 1; //y
int JoyStick_Z = 3; //key
void setup()
{
  pinMode(JoyStick_Z, INPUT);
  Serial.begin(9600); // 9600 bps
}
void loop()
{
  int x,y,z;
  x=analogRead(JoyStick_X);
  y=analogRead(JoyStick_Y);
  z=digitalRead(JoyStick_Z);
  Serial.print(x,DEC);
  Serial.print(",");
  Serial.print(y,DEC);
  Serial.print(",");
  Serial.println(z,DEC);
  delay(100);
}
```

\*\*\*\*\*

# keyestudio

---

## Project 11: Photo interrupter module



### Introduction

Upright part of this sensor is an infrared emitter and on the other side, it's a shielded infrared detector. By emitting a beam of infrared light from one end to other end, the sensor can detect an object when it passes through the beam. It is used for many applications including optical limit switches, pellet dispensing, general object detection, etc.

### Specification

Supply Voltage: 3.3V to 5V

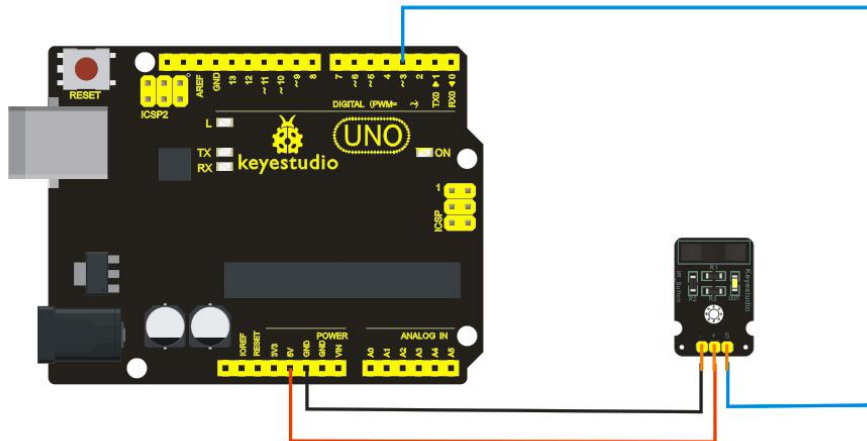
Interface: Digital

Size: 30\*20mm

Weight: 3g

### Connection Diagram

# keyestudio



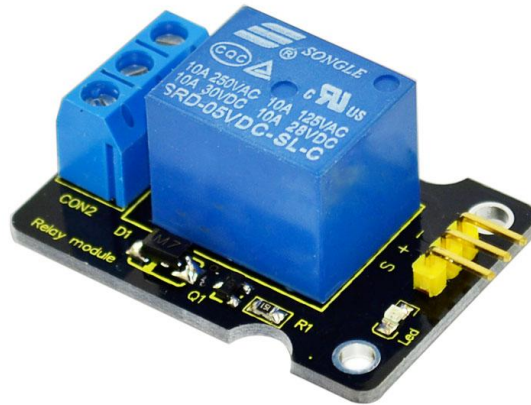
## Sample code

// photo interrupter module

```
int Led = 13 ;// define LED Interface
int buttonpin = 3; // define the photo interrupter sensor interface
int val ;// define numeric variables val
void setup ()
{
  pinMode (Led, OUTPUT) ;// define LED as output interface
  pinMode (buttonpin, INPUT) ;// define the photo interrupter sensor output interface
}
void loop ()
{
  val = digitalRead (buttonpin) ;// digital interface will be assigned a value of 3 to read val
  if (val == HIGH) // When the light sensor detects a signal is interrupted, LED flashes
  {
    digitalWrite (Led, HIGH);
  }
  else
  {
    digitalWrite (Led, LOW);
  }
}
```

\*\*\*\*\*

## Project 12: 5V Relay Module



### Introduction

This single relay module can be used in interactive projects. This module uses SONGLE 5v high-quality relay. It can also be used to control lighting, electrical and other equipment. The modular design makes it easy to expand with the Arduino board (not included). The Relay output is by a light-emitting diode. It can be controlled through digital IO port, such as solenoid valves, lamps, motors and other high current or high voltage devices.

### Specification

Type: Digital

Rated current: 10A (NO) 5A (NC)

Maximum switching voltage: 150VAC 24VDC

Digital interface

Control signal: TTL level

Rated load: 8A 150VAC (NO) 10A 24VDC (NO), 5A 250VAC (NO/NC) 5A 24VDC (NO/NC)

Maximum switching power: AC1200VA DC240W (NO) AC625VA DC120W (NC)

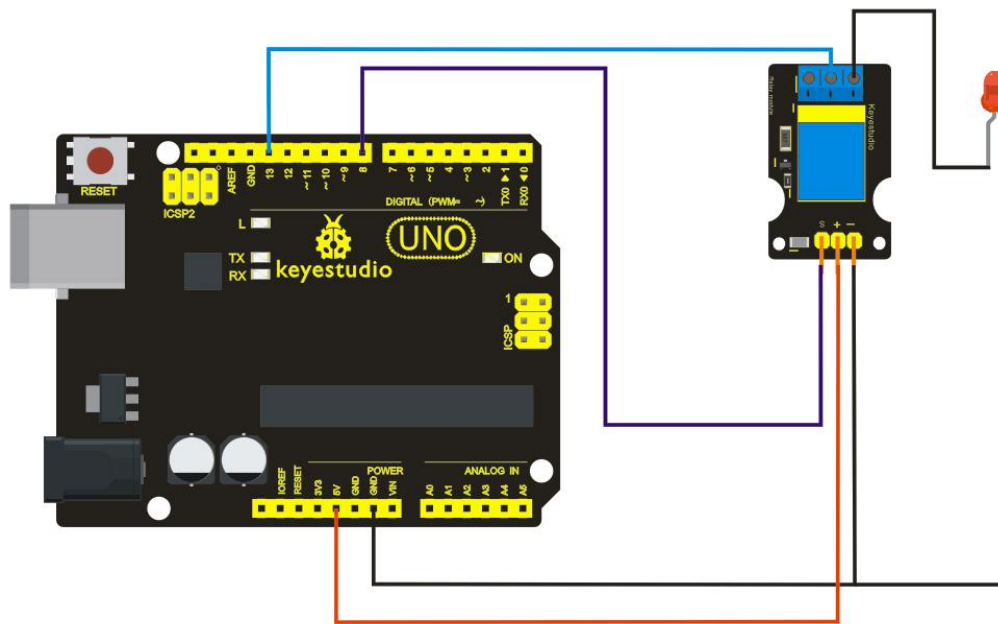
Contact action time: 10ms

Size: 40\*28mm

Weight: 15g

### Connection Diagram

# keyestudio



## Sample Code

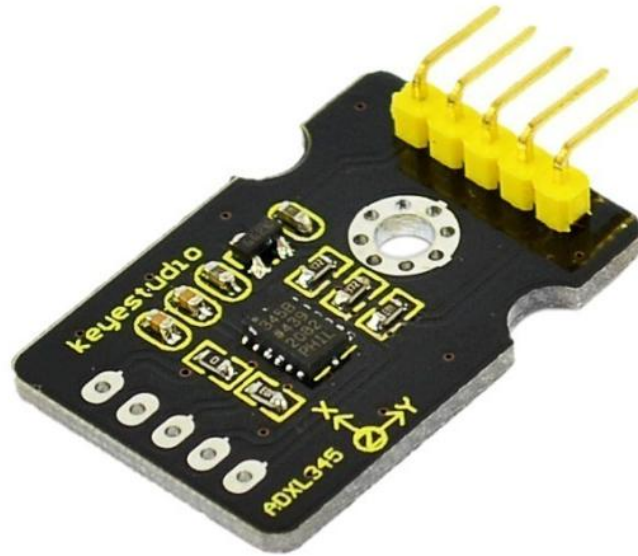
```
int Relay = 8;
void setup()
{
  pinMode(13, OUTPUT);    //Set Pin13 as output
  digitalWrite(13, HIGH); //Set Pin13 High
  pinMode(Relay, OUTPUT); //Set Pin3 as output
}
void loop()
{
  digitalWrite(Relay, HIGH); //Turn off relay
  delay(2000);
  digitalWrite(Relay, LOW);  //Turn on relay
  delay(2000);
}
```

\*\*\*\*\*

# keyestudio

---

## Project 13: ADXL345 Three Axis Acceleration Module



### Introduction

The ADXL345 is a small, thin, low power, 3-axis MEMS accelerometer with high resolution (13-bit) measurement at up to  $\pm 16$  g. Digital output data is formatted as 16-bit twos complement and is accessible through either a SPI (3- or 4-wire) or I2C digital interface.

The ADXL345 is well suited to measure the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion or shock. Its high resolution (4 mg/LSB) enables measurement of inclination changes less than 1.0 degrees;.

### Specification

2.0-3.6VDC Supply Voltage

Ultra Low Power: 40uA in measurement mode, 0.1uA in standby@ 2.5V

Tap/Double Tap Detection

Free-Fall Detection

SPI and I2C interfaces

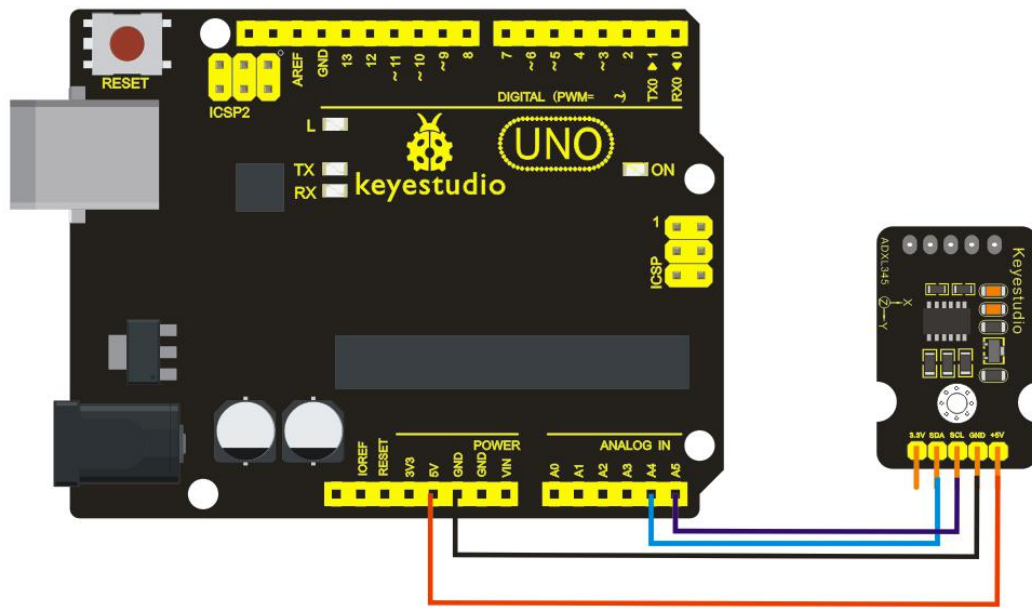
Size: 30\*20mm

Weight: 3g

### Connection Diagram



# keyestudio



## Sample Code

/\*

The circuit:

VCC: 5V

GND: ground

SCL: UNO SCL

SDA: UNO SDA

This example code is in the public domain.

\*/

#include <Wire.h>

// Registers for ADXL345

#define ADXL345\_ADDRESS (0xA6 >> 1) // address for device is 8 bit but shift to the  
// right by 1 bit to make it 7 bit because the  
// wire library only takes in 7 bit addresses

#define ADXL345\_REGISTER\_XLSB (0x32)

int accelerometer\_data[3];

// void because this only tells the cip to send data to its output register

// writes data to the slave's buffer

void i2c\_write(int address, byte reg, byte data) {

    // Send output register address

    Wire.beginTransmission(address);

    // Connect to device

# keyestudio

---

```
Wire.write(reg);
// Send data
Wire.write(data); //low byte
Wire.endTransmission();
}

// void because using pointers
// microcontroller reads data from the sensor's input register
void i2c_read(int address, byte reg, int count, byte* data) {
    // Used to read the number of data received
    int i = 0;
    // Send input register address
    Wire.beginTransmission(address);
    // Connect to device
    Wire.write(reg);
    Wire.endTransmission();

    // Connect to device
    Wire.beginTransmission(address);
    // Request data from slave
    // Count stands for number of bytes to request
    Wire.requestFrom(address, count);
    while(Wire.available()) // slave may send less than requested
    {
        char c = Wire.read(); // receive a byte as character
        data[i] = c;
        i++;
    }
    Wire.endTransmission();
}

void init_adxl345() {
    byte data = 0;

    i2c_write(ADXL345_ADDRESS, 0x31, 0x0B); // 13-bit mode  +- 16g
    i2c_write(ADXL345_ADDRESS, 0x2D, 0x08); // Power register

    i2c_write(ADXL345_ADDRESS, 0x1E, 0x00); // x
    i2c_write(ADXL345_ADDRESS, 0x1F, 0x00); // Y
    i2c_write(ADXL345_ADDRESS, 0x20, 0x05); // Z

    // Check to see if it worked!
```

# keyestudio

---

```
i2c_read(ADXL345_ADDRESS, 0X00, 1, &data);
if(data==0xE5)
    Serial.println("it work Success");
else
    Serial.println("it work Fail");
}

void read_adxl345() {
    byte bytes[6];
    memset(bytes,0,6);

    // Read 6 bytes from the ADXL345
    i2c_read(ADXL345_ADDRESS, ADXL345_REGISTER_XLSB, 6, bytes);
    // Unpack data
    for (int i=0;i<3;++i) {
        accelerometer_data[i] = (int)bytes[2*i] + (((int)bytes[2*i + 1]) << 8);
    }
}

// initialise and start everything
void setup() {
    Wire.begin();
    Serial.begin(9600);
    for(int i=0; i<3; ++i) {
        accelerometer_data[i] = 0;
    }
    init_adxl345();
}

void loop() {
    read_adxl345();
    Serial.print("ACCEL: ");
    Serial.print(float(accelerometer_data[0])*3.9/1000); //3.9mg/LSB scale factor in 13-bit mode
    Serial.print("\t");
    Serial.print(float(accelerometer_data[1])*3.9/1000);

    Serial.print("\t");
    Serial.print(float(accelerometer_data[2])*3.9/1000);
    Serial.print("\n");
    delay(100);
}

*****
```

## Project 14: Rotary Encoder module



### Introduction

The rotary encoder can count the pulse outputting times during the process of its rotation in positive and reverse direction by rotating. This rotating counting is unlimited, not like potential counting. It can be restored to initial state to count from 0.

### Specification

Power Supply: 5V

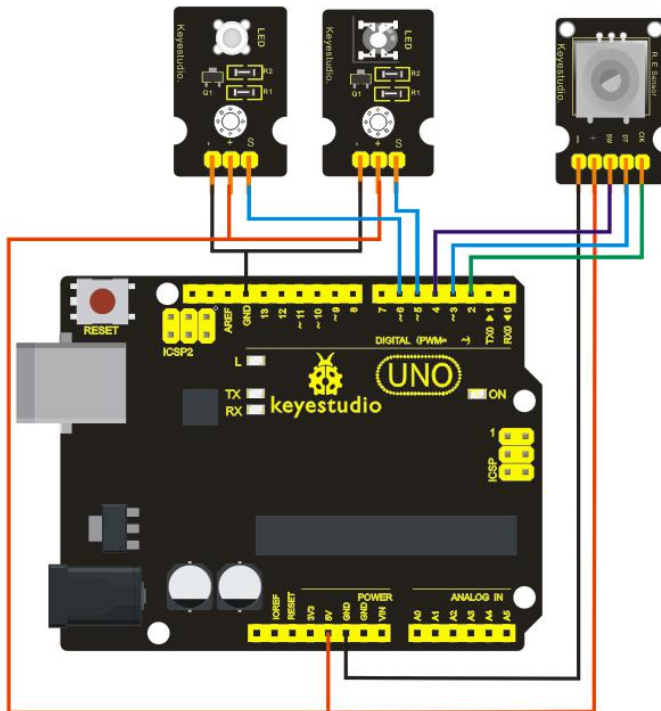
Interface: Digital

Size: 30\*20mm

Weight: 7g

### Connection Diagram

# keyestudio



## Sample Code

```
const int interruptA = 0;
const int interruptB = 1;
int CLK = 2;    // PIN2
int DAT = 3;    // PIN3
int BUTTON = 4; // PIN4
int LED1 = 5;   // PIN5
int LED2 = 6;   // PIN6
int COUNT = 0;

void setup()
{
    attachInterrupt(interruptA, RoteStateChanged, FALLING);
    // attachInterrupt(interruptB, buttonState, FALLING);
    pinMode(CLK, INPUT);
    digitalWrite(2, HIGH); // Pull High Restance
    pinMode(DAT, INPUT);
    digitalWrite(3, HIGH); // Pull High Restance

    pinMode(BUTTON, INPUT);
    digitalWrite(4, HIGH); // Pull High Restance
    pinMode(LED1, OUTPUT);
    pinMode(LED2, OUTPUT);
}
```

# keyestudio

---

```
    Serial.begin(9600);
}

void loop()
{
    if  (!(digitalRead(BUTTON)))
    {
        COUNT = 0;
        Serial.println("STOP COUNT = 0");
        digitalWrite(LED1, LOW);
        digitalWrite(LED2, LOW);
        delay (2000);
    }
    Serial.println(COUNT);
}

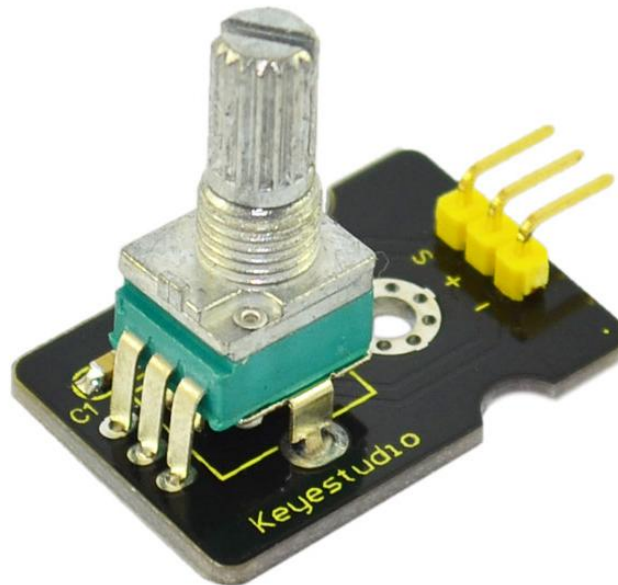
//-----
void RoteStateChanged() //When CLK  FALLING READ DAT
{
    if  (digitalRead(DAT)) // When DAT = HIGH IS FORWARD
    {
        COUNT++;
        digitalWrite(LED1, HIGH);
        digitalWrite(LED2, LOW);
        delay(20);
    }
    else // When DAT = LOW IS BackRote
    {
        COUNT--;
        digitalWrite(LED2, HIGH);
        digitalWrite(LED1, LOW);
        delay(20);
    }
}

*****
```

# keyestudio

---

## Project 15: Analog Rotation Sensor



### Introduction

This analog Rotation Sensor is arduino compatible. It is based on a potentiometer. Its voltage can be subdivided into 1024, easy to be connected to Arduino with our sensor shield. Combined with other sensors, we can make interesting projects by reading the analog value from the IO port.

### Specification

Supply Voltage: 3.3V to 5V

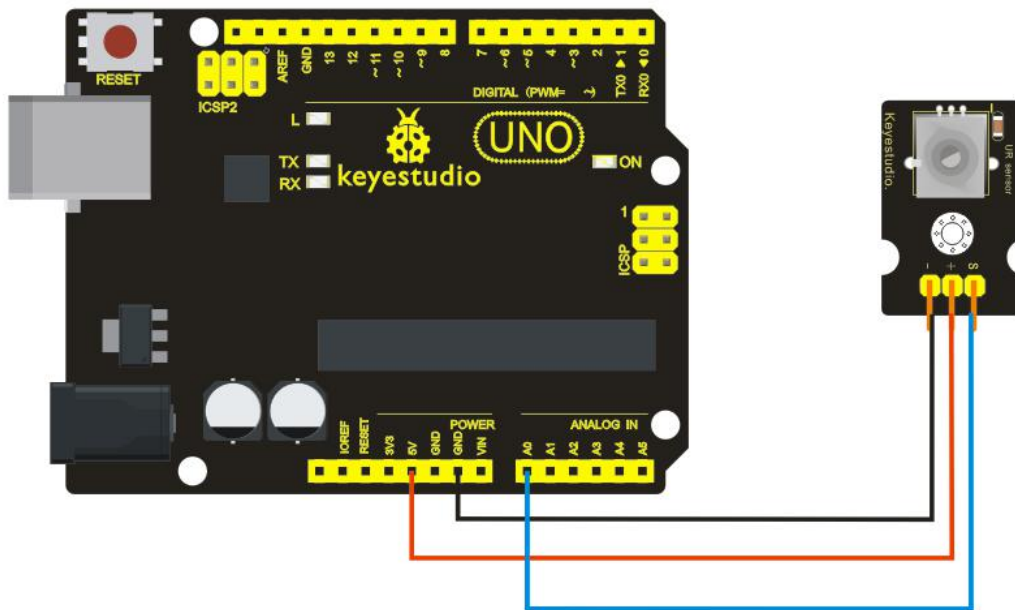
Interface: Analog

Size: 30\*20mm

Weight: 8g

### Connection Diagram

# keyestudio



## Sample Code

```
///Arduino Sample Code
void setup()
{
  Serial.begin(9600); //Set serial baud rate to 9600 bps
}
void loop()
{
  int val;
  val=analogRead(0); //Read rotation sensor value from analog 0
  Serial.println(val,DEC); //Print the value to serial port
  delay(100);
}
```

\*\*\*\*\*



# keyestudio

---

## Project 16: RGB LED module



### Introduction

This is a full-color LED module, which contains 3 basic colors—red, green and blue. They can be seen as separate LED lights. After program, we can turn them on and off by sequence. We can also use PWM analog output to mix the three colors to generate different colors.

### Specification

Color: red, green and blue

Brightness: High

Voltage: 5V

Input: digital level

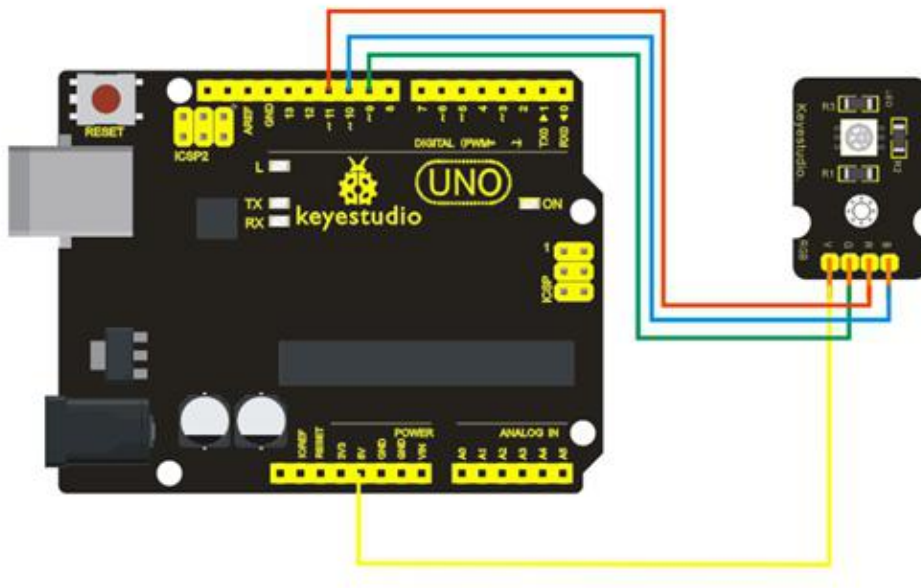
Size: 30 \* 20mm

Weight: 3g

### Connection diagram

# keyestudio

---



## Sample code

```
int redpin = 11; //select the pin for the red LED
int bluepin =10; // select the pin for the blue LED
int greenpin =9; // select the pin for the green LED
int val;

void setup() {
  pinMode(redpin, OUTPUT);
  pinMode(bluepin, OUTPUT);
  pinMode(greenpin, OUTPUT);
}

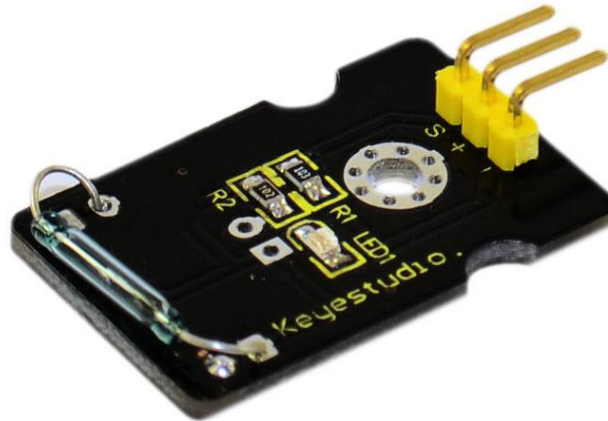
void loop()
{for(val=255; val>0; val--)
  {analogWrite(11, val);
   analogWrite(10, 255-val);
   analogWrite(9, 128-val);
   delay(1);
  }
for(val=0; val<255; val++)
  {analogWrite(11, val);
   analogWrite(10, 255-val);
   analogWrite(9, 128-val);
   delay(1);
  }
}
```

\*\*\*\*\*

# keyestudio

---

## Project 17: keyestudio Reed Switch Module



### Introduction

Reed Switch is a special switch and a main component for reed relay and proximity switch. Reed switch is usually comprised of two soft magnetic material and metal reed contacts which will disconnect itself when there is no magnetic. In addition, some reed switches are also equipped with another reed acting as the third normally-closed contact. These reed contacts are encapsulated in a glass tube full of inert gases(such as nitrogen and helium) or in a vacuum glass tube. The reeds encapsulated in the glass tube are placed in parallel with ends overlapped. Certain amount of space or mutual contactation will be reserved so as to constitute the normally-open or normally-closed contacts of the switch.

Reed switch can be used as sensor for count, limit and other purposes. For instance, a kind of bike-kilometer is constituted by sticking magnetic to the tire and mounting reed switch aside. We can mount reed switch on the door for alarming purpose or as switches.

Reed switch has been widely applied in household appliances, cars, communication, industry, healthcare and security areas. Furthermore, it can also be applied to other sensors and electric devices such as liquidometer, door magnet, reed relay, oil level sensor and proximity sensor(magnetic sensor). It can be used under high-risk environment.

### Specification

Working voltage: DC 3.3V-5V

Working current:  $\geq 20\text{mA}$

Working temperature:  $-10^{\circ}\text{C} \sim +50^{\circ}\text{C}$

Detection distance:  $\leq 10\text{mm}$

IO Interface: 3 wire interface (-/+ /S)

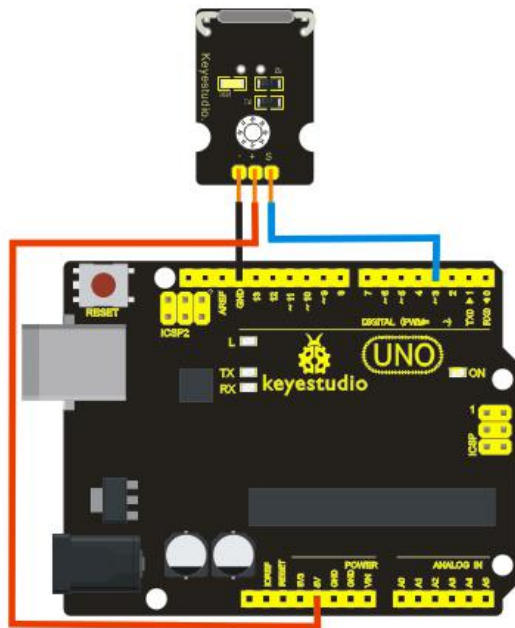
Size: 30\*20mm

Weight: 3g

# keyestudio

---

## Connection diagram:



## Sample code

```
int Led=13;//define LED interface
int buttonpin=3; //define magnetic ring sensor interface
int val;//define digital variable val
void setup()
{
  pinMode(Led,OUTPUT);//define LED as output interface
  pinMode(buttonpin,INPUT);//define magnetic ring sensor as output interface }
void loop()
{
  val=digitalRead(buttonpin);// read and assign the value of digital interface 3 to val

  if(val==HIGH)//When a signal is detected by magnetic ring sensor, LED will flash
  {
    digitalWrite(Led,HIGH);
  }
  else
  {
    digitalWrite(Led,LOW);
  }
}
```

\*\*\*\*\*

## Project 18: Soil Humidity Sensor



### Introduction

This is a simple soil humidity sensor aims to detect the soil humidity. If the soil is in lack of water, the analog value output by the sensor will decrease, otherwise, it will increase. If you use this sensor to make an automatic watering device, it can detect whether your botany is thirsty to prevent it from withering when you go out. Using the sensor with Arduino controller makes your plant more comfortable and your garden smarter.

The soil humidity sensor module is not as complicated as you might think, and if you need to detect the soil in your project , it will be your best choice.

The sensor is set with two probes inserted into the soil, then with the current go through the soil, the sensor will get resistance value by reading the current changes between the two probes and convert such resistance value into moisture content. The higher moisture (less resistance), the higher conductivity the soil has.

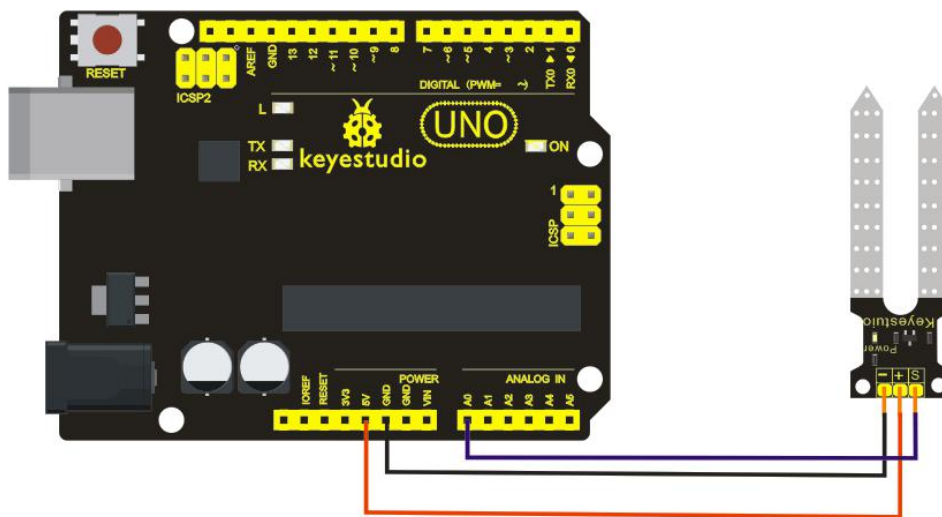
The surface of the sensor have undergone metallization process to prolong its service life. Insert it into the soil and then use the AD converter to read it. With the help of this sensor, the plant can remind you: I need water.

# keyestudio

## Specification

- Power Supply Voltage: 3.3V or 5V
- Working Current:  $\leq 20\text{mA}$
- Output Voltage: 0-2.3V (When the sensor is totally immersed in water, the voltage will be 2.3V) 5V power supply, the higher humidity, the higher the output voltage
- Packaging : Electrostatic bag sealing
- Sensor type: Analog output
- Interface definition: Pin1- signal, pin2- GND, pin3 - VCC
- Service life: About one year (gold-plated surface for enhancing conductivity and corrosion resistance )
- Module size: 20X60mm

## Connection diagram



## Sample code

```
/*
# Example code for the moisture sensor
# Connect the sensor to the A0(Analog 0) pin on the Arduino board

# the sensor value description
# 0 ~300    dry soil
# 300~700   humid soil
# 700~950   in water
*/

void setup(){

  Serial.begin(57600);

}
```

```
void loop(){  
  
    Serial.print("Moisture Sensor Value:");  
    Serial.println(analogRead(0));  
    delay(100);  
  
}
```

\*\*\*\*\*

## Project 19: Line Tracking Sensor



### Introduction

This Line Tracking Sensor can detect white lines in black and black lines in white. The single line-tracking signal provides a stable output signal TTL for a more accurate and more stable line. Multi-channel option can be easily achieved by installing required line-tracking robot sensors.

### Specification

Power supply: +5V

Operating current: <10mA

Operating temperature range: 0°C ~ + 50°C

Output interface: 3-wire interface (1 - signal, 2 - power, 3 - power supply negative)

Output Level: TTL level

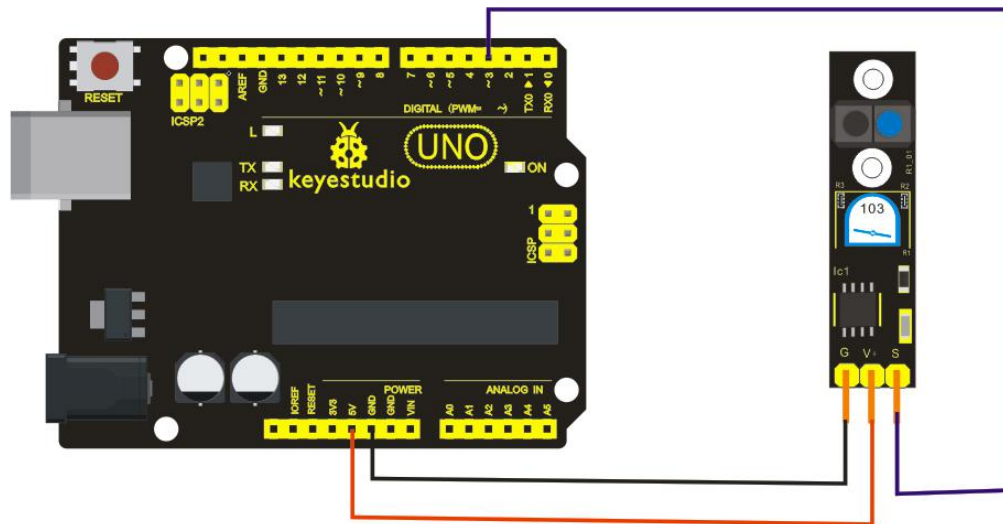
Size: 41.7\*10.7mm

Weight: 3g

# keyestudio

---

## Connection Diagram



## Sample Code

///Arduino Sample Code

```
void setup()
```

```
{
```

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

```
}
```

```
void loop()
```

```
{
```

```
  Serial.println(digitalRead(3)); // print the data from the sensor
```

```
  delay(500);
```

```
}
```

```
*****
```