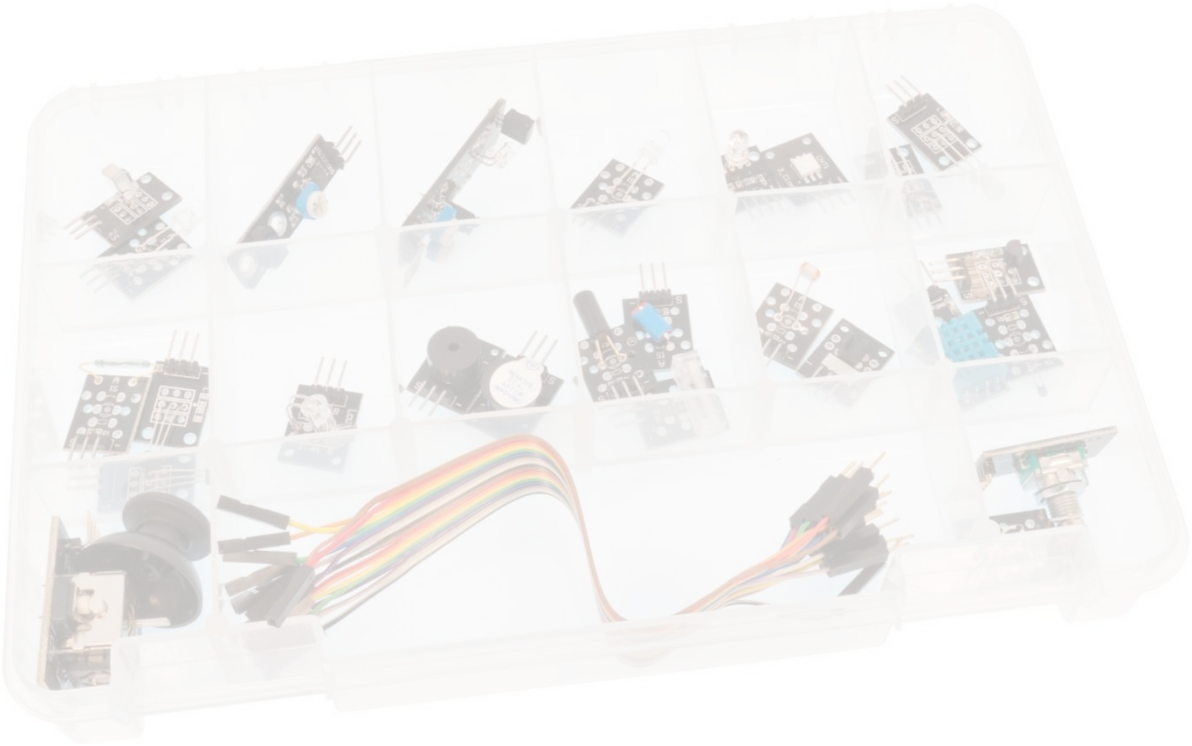
Sensor kit - 25 modules

Suitable for experiments with Arduino & Raspberry Pi



www.electrokit.com

Analog Joystick Module

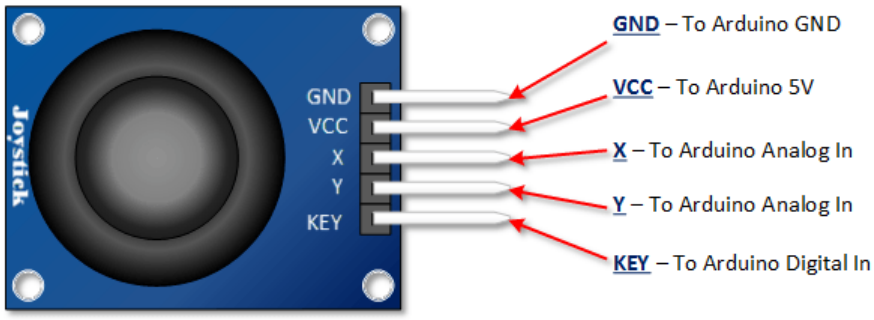
The PS2 style joystick is a thumb operated device, that when put to creative use, offers a convenient way of getting operator input. It fundamentally consists of two potentiometers and a push button switch.

The two potentiometers indicate which direction the potentiometer is being pushed.

The switch sends a low (or ground) when the knob is pressed.

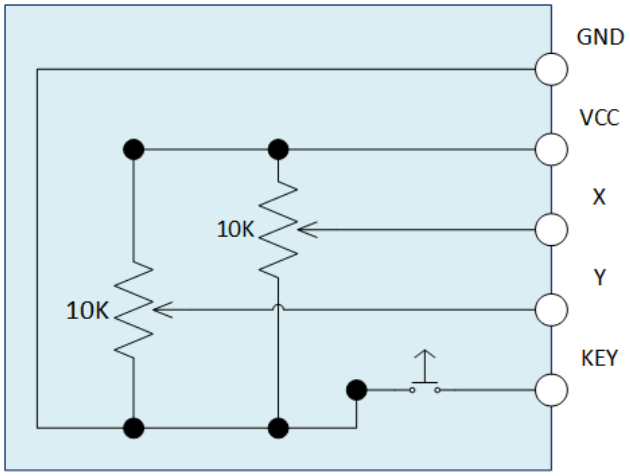
Pinout

This input device interfaces to your Arduino via five pins. Three of which are inputs to your Arduino, while the remaining two supply voltage and ground.



Arduino PS2 Joystick Schematic

As you can see in the schematic below, full deflection of a potentiometer in either direction will provide ground or the supply voltage as an output

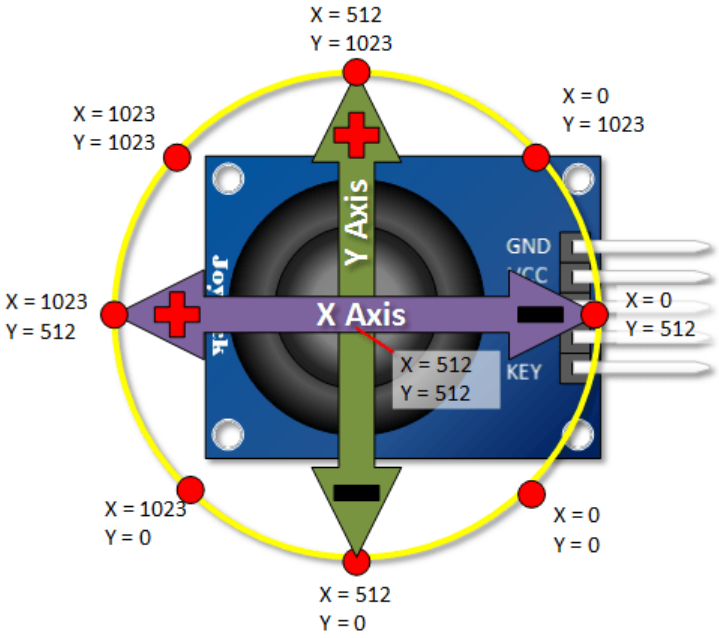


Arduino PS2 Joystick Output Orientation

In order to put this thumb control to use, you are going to want to understand which direction is X and which direction is Y. You will also need to decipher the direction it is being pushed in either the X or the Y direction.

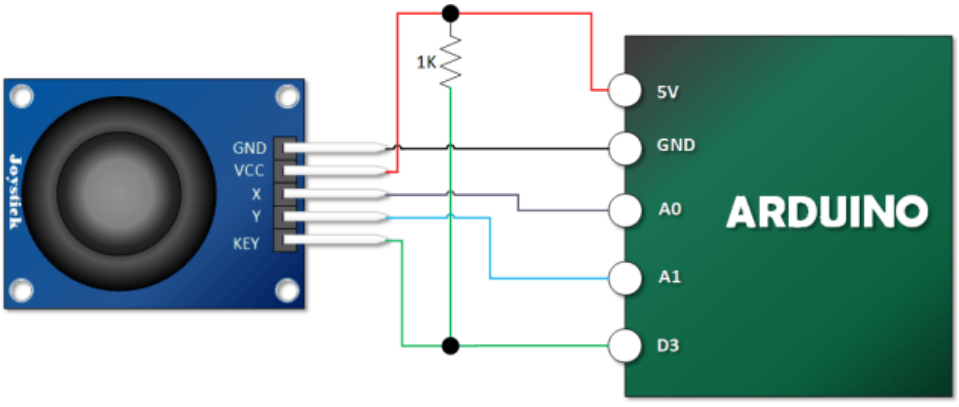
In this tutorial we are using analog inputs to measure the joystick position. The analog inputs provided indications that range between 0 and 1023.

The graphic below shows the X and Y directions and also gives an indication of how the outputs will respond when the joystick is pushed in various directions.



Connection to Arduino

Note that I use a pull up resistor between the key switch and the digital input. Once you move beyond experimentation, I highly recommend some sort of software or hardware debounce for this switch as well.



Arduino Example Sketch

The following Arduino Sketch will read both joystick axis and the push button and output the values in serial monitor.

```
int Xin= A0; // X Input Pin
int Yin = A1; // Y Input Pin
int KEYin = 3; // Push Button

void setup ()
{
  pinMode (KEYin, INPUT);
  Serial.begin (9600);
}
void loop ()
{
  int xVal, yVal, buttonVal;

  xVal = analogRead (Xin);
  yVal = analogRead (Yin);
  buttonVal = digitalRead (KEYin);

  Serial.print("X = ");
  Serial.println (xVal, DEC);

  Serial.print ("Y = ");
  Serial.println (yVal, DEC);

  Serial.print("Button is ");
  if (buttonVal == HIGH){
    Serial.println ("not pressed");
  }
  else{
    Serial.println ("PRESSED");
  }

  delay (500);
}
```

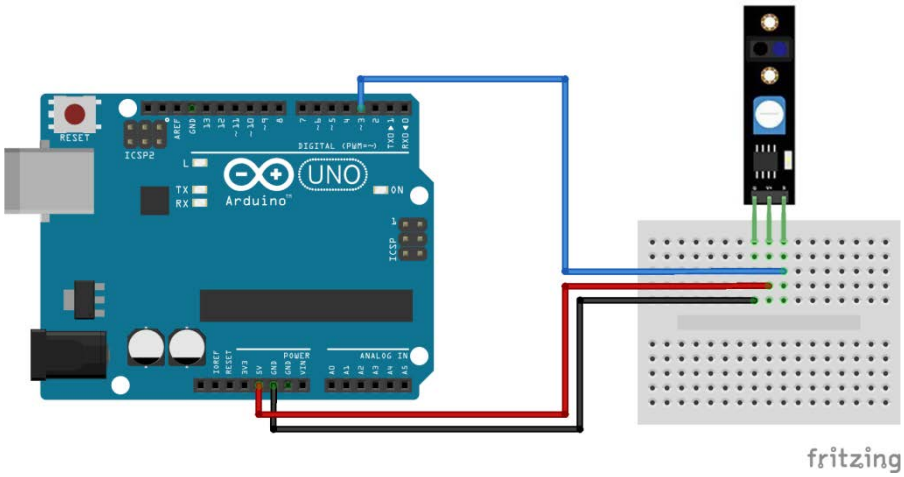
Line Follow Sensor Module

The line follow sensor module has a reflective IR sensor for determining if the surface is light or dark. The sensor can also be used to sense if an object is close or far away. It is used in robots to follow the line or keep inside the restricted area.

If no object is in close proximity to the sensor or the surface is dark, the output is high. If a reflective object is near or the surface is light, the output will go high.

Pinout and connection to Arduino

The sensor connects to the Arduino with three wires. Connect VCC to 5v, GND to GND and OUT to pin 3.

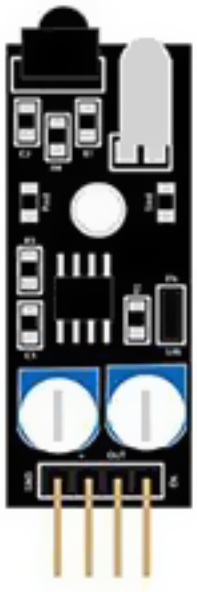


Arduino Example Sketch

The following Arduino Sketch will light the LED when object is near or surface is light.

```
int led = 13;//LED pin
int sensor = 3; //sensor pin
int val; //numeric variable

void setup()
{
        pinMode(led, OUTPUT); //set LED pin as output
        pinMode(sensor, INPUT); //set sensor pin as input
}

void loop()
{
        val = digitalRead(sensor); //Read the sensor
        if(val == HIGH)    { digitalWrite(Led, HIGH); }
        else               { digitalWrite(Led, LOW); }
}
```

Collision Detection Module

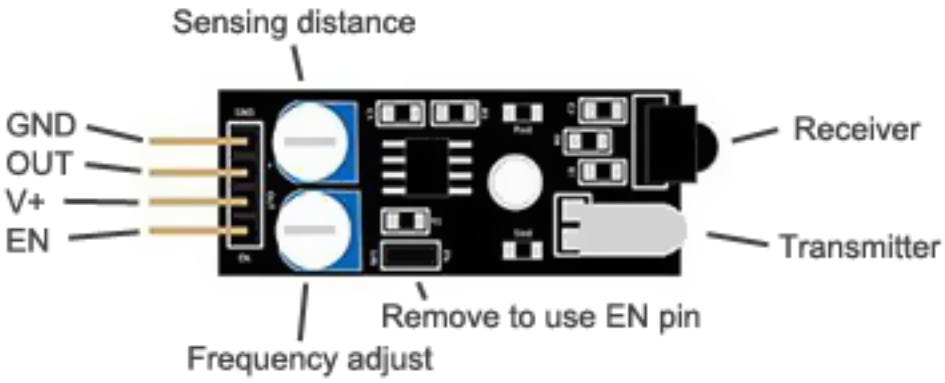


Infrared obstacle avoidance sensor is designed for wheeled robot obstacle avoidance with adjustable detection distance. The module consists of one infrared emitter and one detector. When there is an obstacle in front of the sensor, the infrared light from the emitter is reflected back to the receiver. The signal is squared up by a comparator to produce a digital signal. When there is no obstacle, the output is high. When obstacle is in range, the output is low. The sensitivity can be adjusted by means of potentiometer knob. Several sensors can be connected in parallel. Enable pin can be used for individual control. Remove jumper to use EN input.

Operating Voltage	5V
Maximum Current	20mA / 5VDC
Detection distance	2 - 4cm
Detection angle	35°
Dimensions	28mm x 23mm

Pinout and Connection to Arduino

Connect V+ to 5V. Connect output to pin 8 on the Arduino and Ground (-) to GND. To use EN pin, remove jumper and connect to an output on Arduino.



Arduino Example Sketch

The following Arduino Sketch will light the LED when an obstacle is detected.

```
int ledPin = 13;
int sensorPin = 3;
int val;

void setup ()
{
  pinMode (ledPin, OUTPUT);
  pinMode (sensorPin, INPUT);
}

void loop ()
{
  val = digitalRead(sensorPin);
  if (val == HIGH)
  {
    digitalWrite(ledPin, LOW);
  }
  else
  {
    digitalWrite(ledPin, HIGH);
  }
}
```

Active Buzzer Module

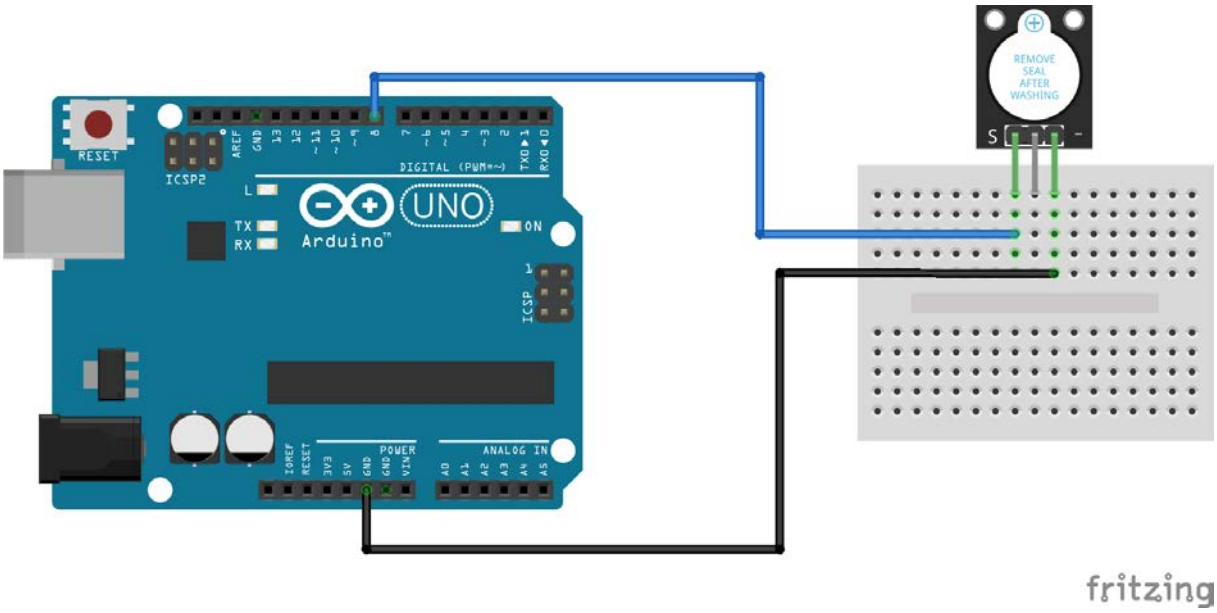
Active Buzzer Arduino module produces a single-tone sound when signal is high. To produce different tones use the Passive Buzzer module.

The Active Buzzer module consists of a piezoelectric buzzer with a built-in oscillator. It generates a sound of approximately 2.5 kHz when signal is high.

Operating Voltage	3.5V ~ 5.5V
Maximum Current	30mA / 5VDC
Resonance Frequency	2500Hz ± 300Hz
Minimum Sound Output	85dB @ 10cm
Working Temperature	-20°C ~ 70°C [-4°F ~ 158°F]
Storage Temperature	-30°C ~ 105°C [-22°F ~ 221°F]
Dimensions	18.5mm x 15mm [0.728in x 0.591in]

Pinout and Connection to Arduino

Connect signal (S) to pin 8 on the Arduino and Ground (-) to GND. Be aware that some boards are wrongly labeled, try inverting the cables if you can't hear any sound when running the sketch.



Arduino Example Sketch

The following Arduino Sketch will continually turn the buzzer on and off, generating a series of short high-pitched beeps.

```
int buzzerPin = 8;

void setup ()
{
  pinMode (buzzerPin, OUTPUT);
}

void loop ()
{
  digitalWrite (buzzerPin, HIGH);
  delay (500);
  digitalWrite (buzzerPin, LOW);
  delay (500);
}
```

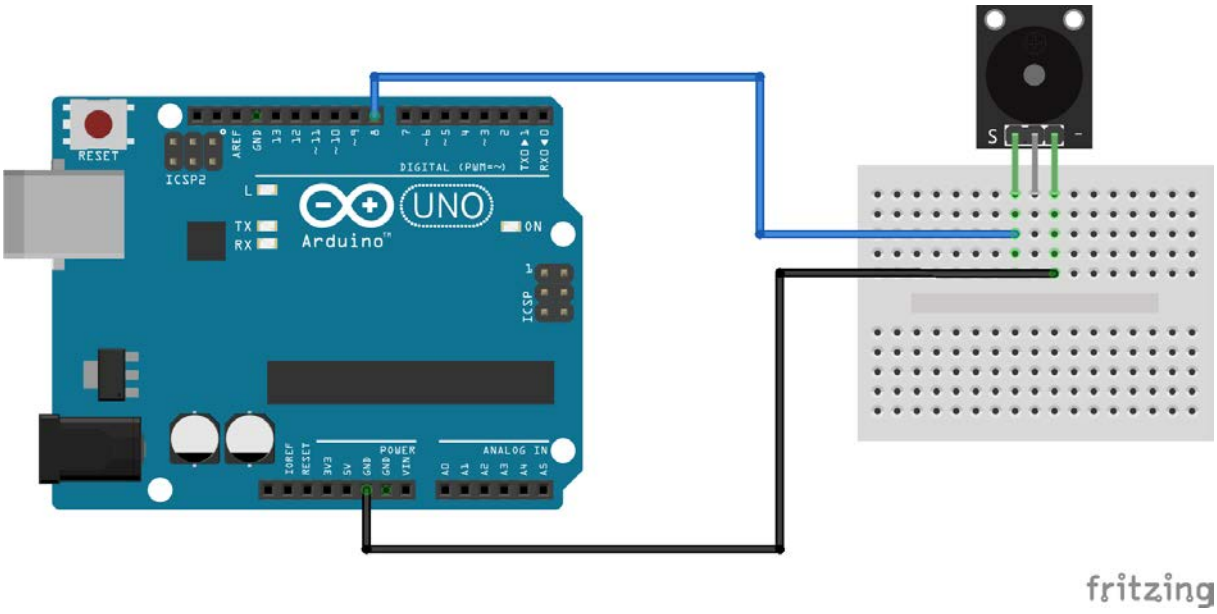
Passive Piezoelectric Buzzer Module

The Passive Buzzer Module consists of a passive piezoelectric buzzer. It can generate tones between 1.5 to 2.5 kHz by switching it on and off at different frequencies either using delays or PWM.

Operating Voltage	1.5 - 15V
Frequency range	1.5 – 2.5kHz
Dimensions	18.5mm x 15mm [0.728in x 0.591in]

Pinout and Connection to Arduino

Connect signal (S) to pin 8 on the Arduino and ground (-) to GND. The middle pin is not used.



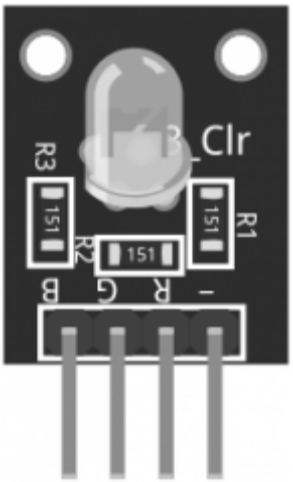
Arduino Example Sketch

The following Arduino sketch will generate two different tones by turning on and off the KY-006 buzzer at different frequencies using a delay.

```
int buzzer = 8; // set the buzzer control digital IO pin

void setup() {
        pinMode(buzzer, OUTPUT); // set pin 8 as output
}

void loop() {
        for (int i = 0; i < 80; i++) {  // make a sound
                digitalWrite(buzzer, HIGH); // send high signal to buzzer
                delay(1); // delay 1ms
                digitalWrite(buzzer, LOW); // send low signal to buzzer
                delay(1);
        }
        delay(50);
        for (int j = 0; j < 100; j++) { //make another sound
                digitalWrite(buzzer, HIGH);
                delay(2); // delay 2ms
                digitalWrite(buzzer, LOW);
                delay(2);
        }
        delay(100);
}
```

RGB LED Module

RGB full color LED Module for Arduino emits a range of colors by mixing red, green and blue. The amount of each primary color is adjusted using PWM.



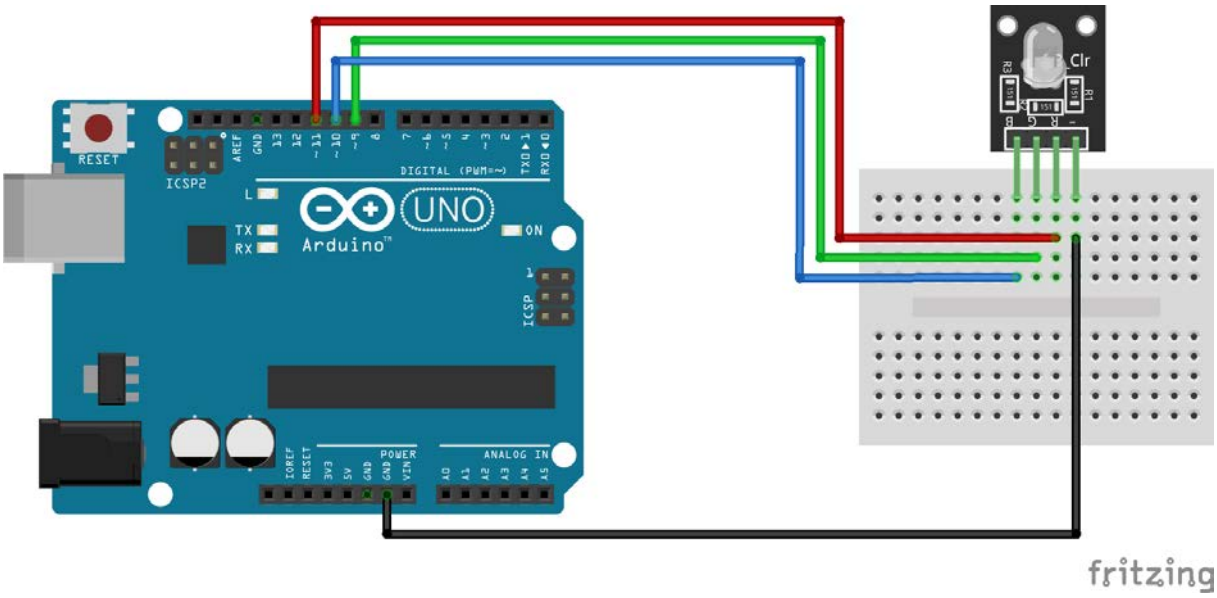
This module consists of a 5mm RGB LED and three 150Ω limiting resistors to prevent burnout. Adjusting the PWM signal on each color pin will result on different colors.

Operating Voltage	5V
LED drive mode	Common cathode
LED diameter	5 mm
Operating Voltage	5V

Pinout and Connection to Arduino

Connect the red pin (R) on the KY-016 to pin 11 on the Arduino. Blue (B) to pin 10, green (G) to pin 9 and ground (-) to GND. Notice that you do not need to use limiting resistors since they are already included on the board.

Module	Arduino	KY-016
R	Pin 11	R
B	Pin 10	B
G	Pin 9	G
-	GND	-



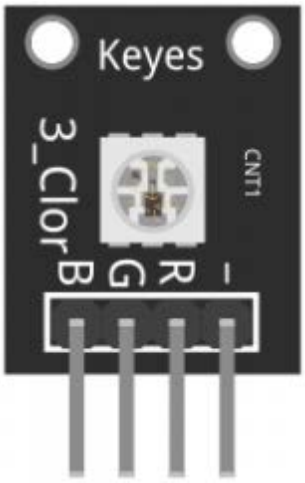
Arduino Example Sketch

The following Arduino sketch will cycle through various colors by changing the PWM value on each of the three primary colors.

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

void loop()
{
  for(val = 255; val > 0; val--)
  {
    analogWrite(redpin, val);  //set PWM value for red
    analogWrite(bluepin, 255 - val); //set PWM value for blue
    analogWrite(greenpin, 128 - val); //set PWM value for green
    Serial.println(val); //print current value
    delay(1);
  }
  for(val = 0; val < 255; val++)
  {
    analogWrite(redpin, val);
    analogWrite(bluepin, 255 - val);
    analogWrite(greenpin, 128 - val);
    Serial.println(val);
    delay(5);
  }
}
```

RGB LED SMD Module

RGB full color LED Module for Arduino emits a range of colors by mixing red, green and blue. The amount of each primary color is adjusted using PWM.



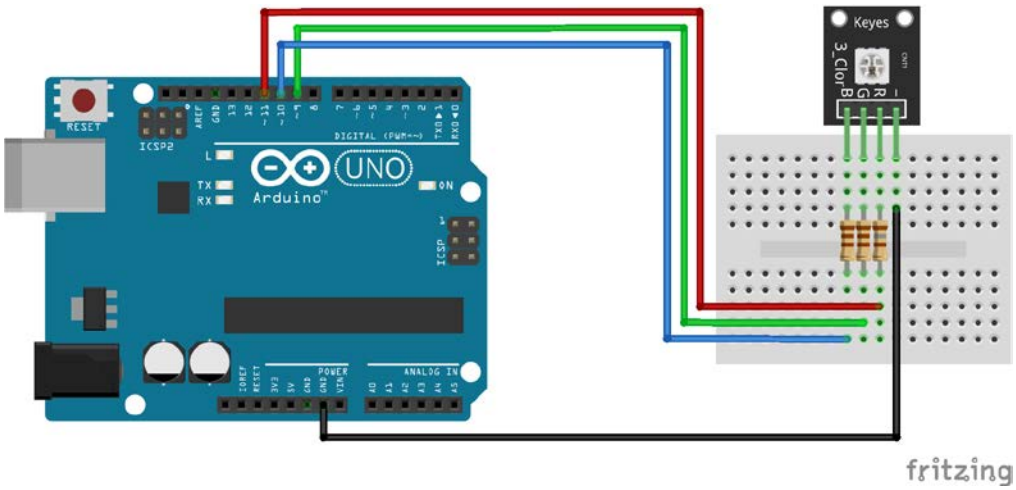
The module consists of a 5050 SMD LED. It's compatible with popular electronics platforms like Arduino, Raspberry Pi and ESP8266.

Operating Voltage	5V max Red 1.8V ~2.4V Green 2.8V ~ 3.6V Blue 2.8V ~ 3.6V
Forward Current	20mA ~ 30mA
Operating Temperature	-25°C to 85°C [-13°F ~ 185°F]
Dimensions	18.5mm x 15mm [0.728in x 0.591in]

Pinout and Connection to Arduino

You need to use resistors to prevent burnout!

Module	Breadboard	Arduino
R	180Ω resistor	Pin 9
G	100Ω resistor	Pin 10
B	100Ω resistor	Pin 11
-	GND	GND

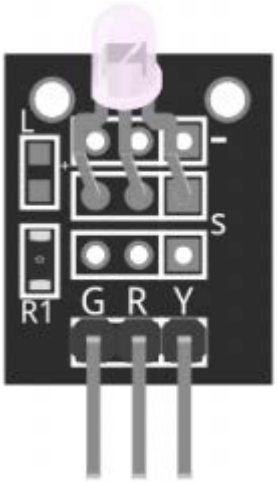


Arduino Example Sketch

The following Arduino sketch will cycle through various colors by changing the PWM value on each of the three primary colors.

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

void loop()
{
  for(val = 255; val > 0; val--)
  {
    analogWrite(redpin, val);  //set PWM value for red
    analogWrite(bluepin, 255 - val); //set PWM value for blue
    analogWrite(greenpin, 128 - val); //set PWM value for green
    Serial.println(val); //print current value
    delay(1);
  }
  for(val = 0; val < 255; val++)
  {
    analogWrite(redpin, val);
    analogWrite(bluepin, 255 - val);
    analogWrite(greenpin, 128 - val);
    Serial.println(val);
    delay(1);
  }
}
```

Two color LED module



Two color LED module for Arduino, emits red and green light. You can adjust the amount of each color using PWM.

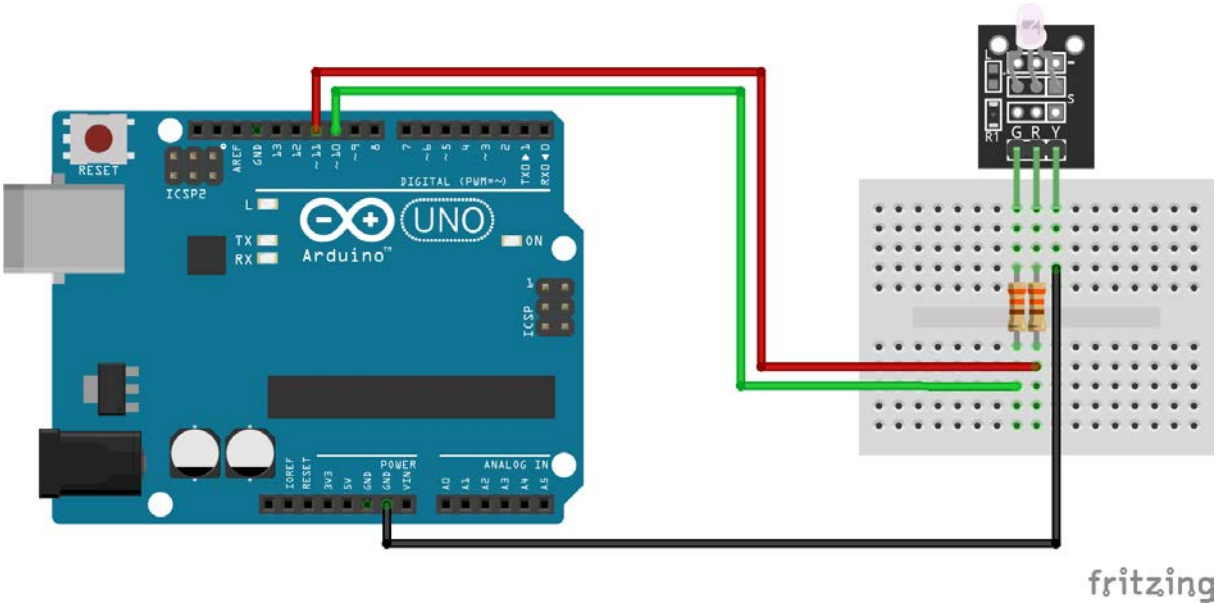
This module consist of a common cathode 3mm or 5mm red/green LED. Since operating voltage is 2.0v ~2.5V, you'll need to use limiting resistors to prevent burnout when connecting to the Arduino.

Operating Voltage	2.0v ~ 2.5v
Using Current	10mA
Diameter	3mm / 5mm
Package Type	Diffusion
Color	Red + Green
Beam Angle	150
Wavelength	571nm + 644nm
Luminosity Intensity	20-40; 40-80

Pinout and Connection to Arduino

We'll use a couple of 330Ω resistors to limit the current from the Arduino and prevent burning the LED.

Module	Breadboard	Arduino
G	330Ω resistor	Pin 10
R	330Ω resistor	Pin 11
Y		GND

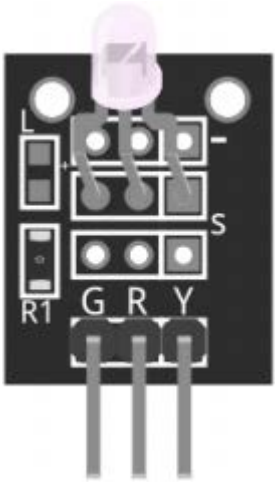


Arduino Example Sketch

The following Arduino sketch will gradually alternate between red and green color.

```
int redpin = 11; // pin for red signal
int greenpin = 10; // pin for green signal
int val;

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

void loop() {
        for(val = 255; val > 0; val--) {
                analogWrite(redpin, val); //dim red
                analogWrite(greenpin, 255 - val); // brighten green
                delay(15);
        }
        for(val = 0; val < 255; val++) {
                analogWrite(redpin, val); //brighten red
                analogWrite(greenpin, 255 - val); //dim green
                delay(15);
        }
}
```

Two color LED module



Two color LED module for Arduino, emits red and green light. You can adjust the amount of each color using PWM.

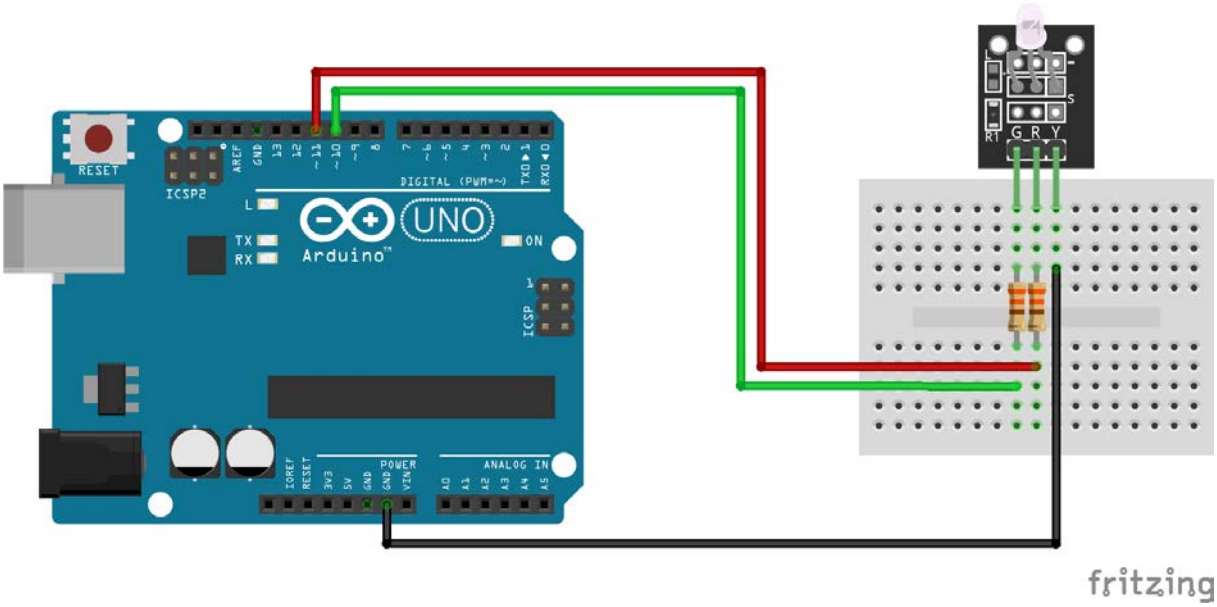
This module consist of a common cathode 3mm or 5mm red/green LED. Since operating voltage is 2.0v ~2.5V, you'll need to use limiting resistors to prevent burnout when connecting to the Arduino.

Operating Voltage	2.0v ~ 2.5v
Using Current	10mA
Diameter	3mm / 5mm
Package Type	Diffusion
Color	Red + Green
Beam Angle	150
Wavelength	571nm + 644nm
Luminosity Intensity	20-40; 40-80

Pinout and Connection to Arduino

We'll use a couple of 330Ω resistors to limit the current from the Arduino and prevent burning the LED.

Module	Breadboard	Arduino
G	330Ω resistor	Pin 10
R	330Ω resistor	Pin 11
Y		GND

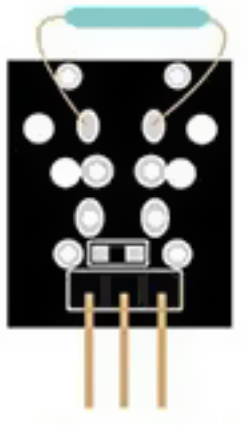


Arduino Example Sketch

The following Arduino sketch will gradually alternate between red and green color.

```
int redpin = 11; // pin for red signal
int greenpin = 10; // pin for green signal
int val;

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

void loop() {
        for(val = 255; val > 0; val--) {
                analogWrite(redpin, val); //dim red
                analogWrite(greenpin, 255 - val); // brighten green
                delay(15);
        }
        for(val = 0; val < 255; val++) {
                analogWrite(redpin, val); //brighten red
                analogWrite(greenpin, 255 - val); //dim green
                delay(15);
        }
}
```

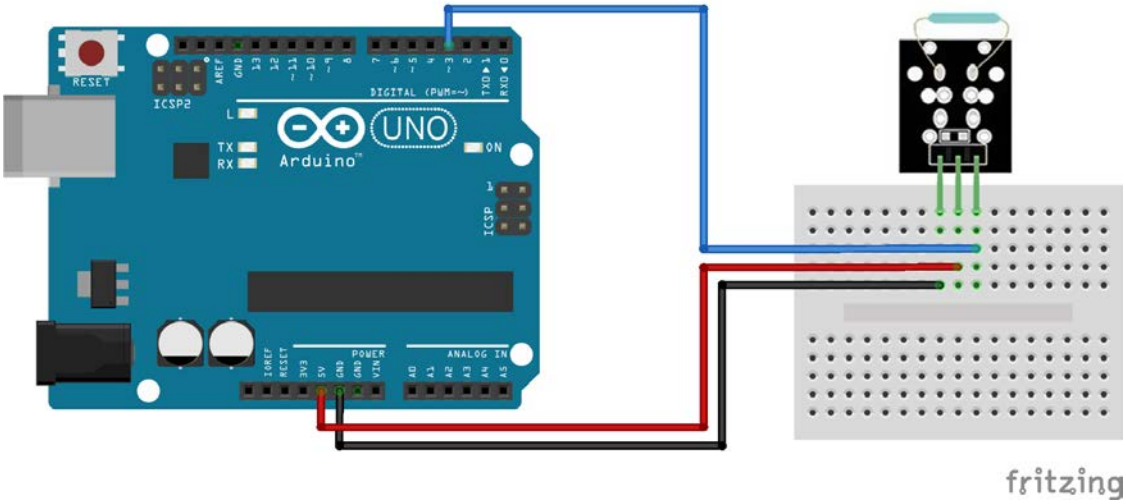
Reed Switch Sensor Module



The reed switch sensor will detect when a magnet is in close range. It works like a regular switch, but operated by magnets. When a magnet is close, the switch also closes. When not magnet is close, the switch will open again.

Pinout and Connection to Arduino

Connect the Power line (middle) and ground (-) to +5 and GND respectively. Connect signal (DO) to pin 3 on the Arduino.

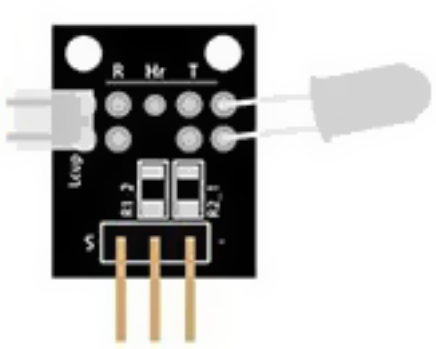


Arduino Example Sketch

The example sketch will light the LED when a magnet is detected.

```
int led = 13;//LED pin
int sensor = 3; //sensor pin
int val; //numeric variable

void setup() {
         pinMode(led, OUTPUT); //set LED pin as output
         pinMode(sensor, INPUT); //set sensor pin as input
}

void loop() {
         val = digitalRead(sensor); //Read the sensor
         if(val == HIGH)    { digitalWrite(Led, HIGH); }
         else               { digitalWrite(Led, LOW); }
}
```

Heartbeat Sensor Module



This project uses bright infrared (IR) LED and a photo transistor to detect the pulse of the finger, a red LED flashes with each pulse.

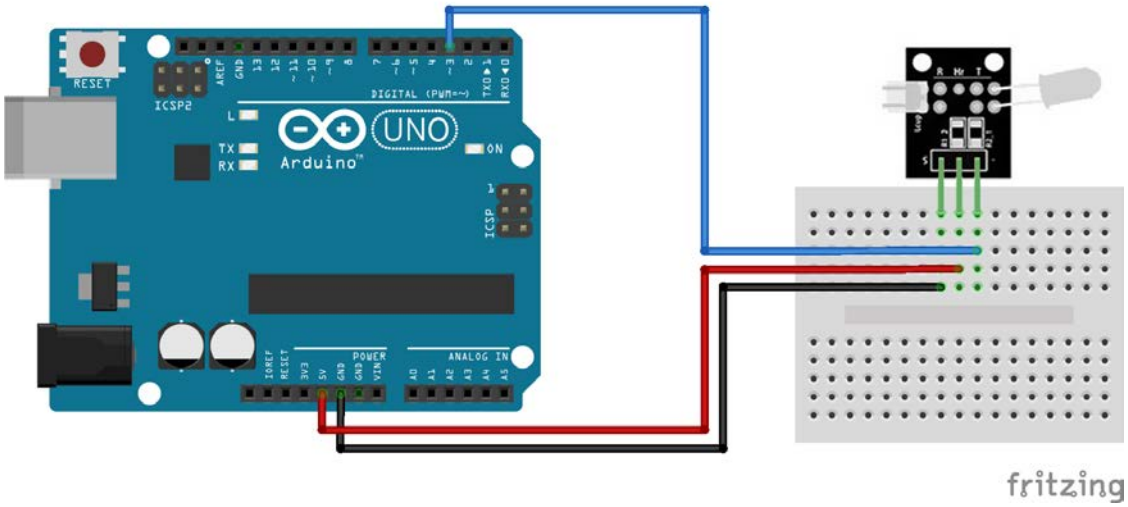
Pulse monitor works as follows: The LED is the light side of the finger, and photo transistor on the other side of the finger, photo transistor used to obtain the flux emitted, when the blood pressure pulse by the finger when the resistance of the photo transistor will be slightly changed.

An important thing is to shield stray light into the phototransistor. For home lighting that is particularly important because the lights at home mostly based 50HZ or 60HZ fluctuate, so faint heartbeat will add considerable noise.

Additional bandpass filtering is most likely required to achieve good data.

Pinout and Connection to Arduino

Connect the Power line (middle) and ground (-) to +5 and GND respectively. Connect signal (DO) to pin 3 on the Arduino.

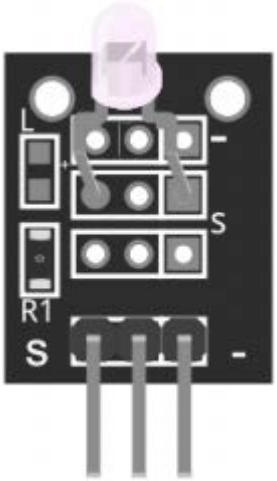


Arduino Example Sketch

The example sketch will output the raw sensor value and a filtered version in serial monitor.

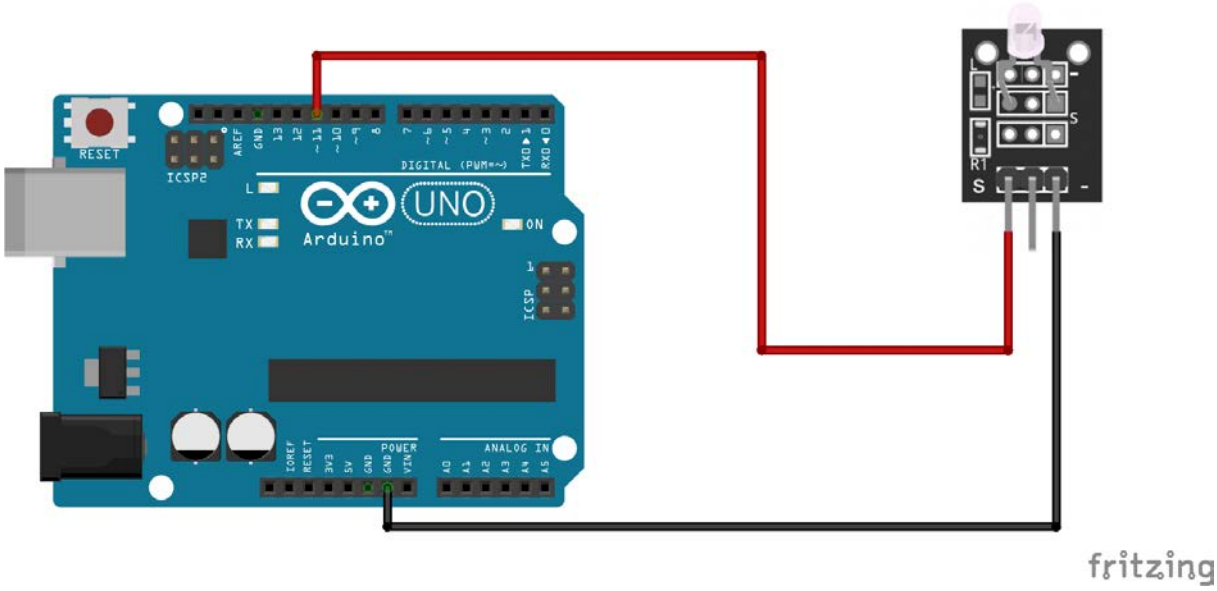
```
int sensorPin = 0;
double alpha = 0.75;
int period = 100;
double change = 0.0;
double minval = 0.0;

void setup ()
{
  Serial.begin (9600);
}

void loop ()
{
    static double oldValue = 0;
    static double oldChange = 0;

    int rawValue = analogRead (sensorPin);
    double value = alpha * oldValue + (1 - alpha) * rawValue;

    Serial.print (rawValue);
    Serial.print (",");
    Serial.println (value);
    oldValue = value;

    delay (period);
}
```

Color Changing LED module



Color Changing LED module for Arduino, emits pulsating, ever changing light. The changes cannot be controlled by the user but runs freely and randomly. This module consist of a color changing LED and a current limiting resistor. The module can be connected directly to the Arduino.

Pinout and Connection to Arduino



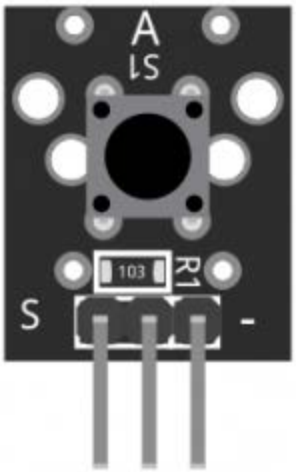
Arduino Example Sketch

The following Arduino sketch will turn the LED on and off every 10 seconds.

```
int ledpin = 11;
int val;

void setup() {
         pinMode(ledpin, OUTPUT);
}

void loop() {
  digitalWrite(ledpin, HIGH);
  delay(10000);
  digitalWrite(ledpin, LOW);
  delay(10000);
}
```

Key Switch Module

The key switch module is a push button that will output a high signal when pressed.

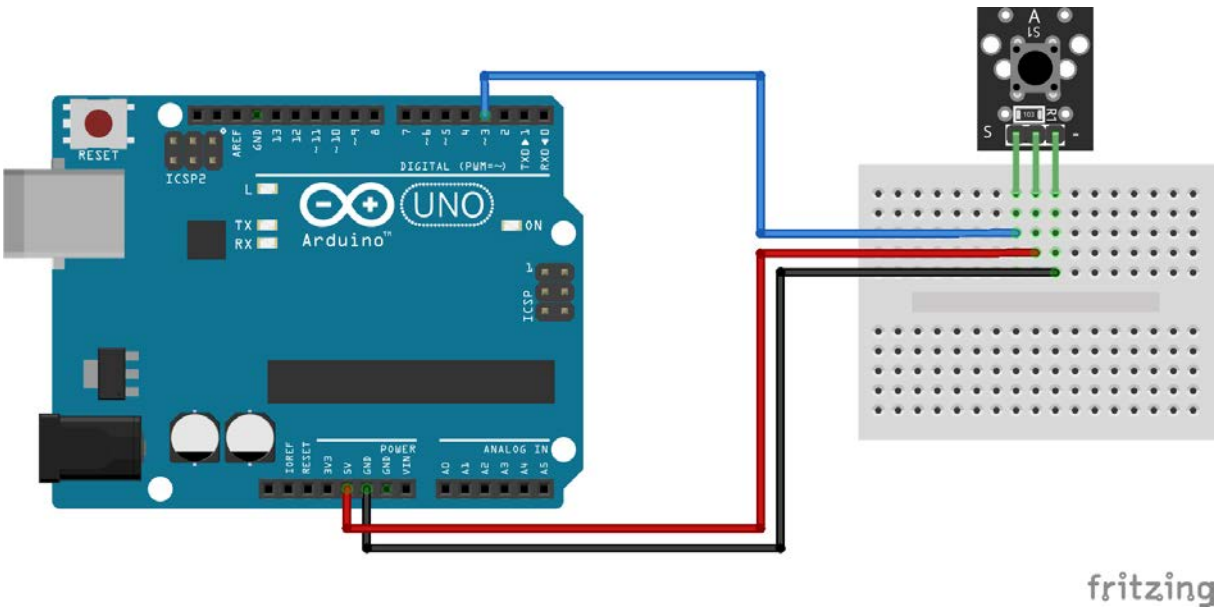


The module consists of a tactile push button switch and a pull-up resistor. It's compatible with popular electronics platforms like Arduino, Raspberry Pi and Esp8266.

Rating	50mA 12VC
Environment temperature	-25°C to 105°C [-13°F to 221°F]
Electrically Life	100,000 cycles
Operating Force	180/230(±20gf)
Dimensions	18.5mm x 15mm [0.728in x 0.591in]

Pinout and Connection to Arduino

Connect the power line (middle) and ground to +5V and GND respectively. Connect signal (S) to pin 3 on the arduino.



Arduino Example Sketch

The following sketch will turn on Arduino's pin 13 LED when the button is pressed.

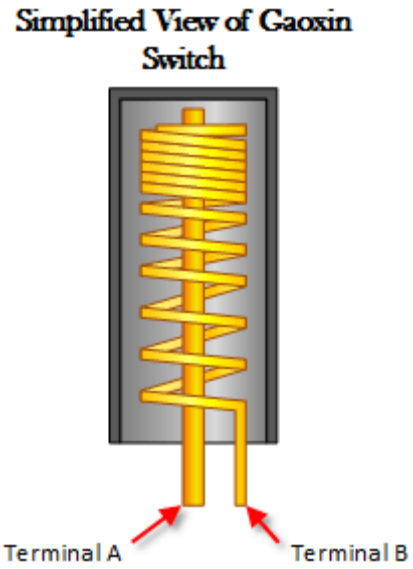
```
int led = 13; //Define the LED pin
int buttonpin = 3; //Define the push button pin
int val; //Define a numeric variable

void setup()
{
          pinMode(led,OUTPUT);
          pinMode(buttonpin,INPUT);
}

void loop()
{
          val = digitalRead(buttonpin); // check the state of the button
          if(val==HIGH) // if button is pressed, turn LED on
          {
                    digitalWrite(led,HIGH);
          }
          else
          {
                    digitalWrite(led,LOW);
          }
}
```

Vibration Shake Sensor

Based on the Gaoxin SW-18010P vibration switch, the Vibration Sensor allows you to use an Arduino to detect impacts, shocks or shaking.

When the switch detects a jolt, the output of the module is sent low.

Vibration Switch Functional Description



The switch primarily consists of a terminal that forms a center post and a second terminal that is a spring that surrounds the center post.

When a sufficient force is transferred to the switch, the terminal consisting of the spring moves and shorts both terminals together.

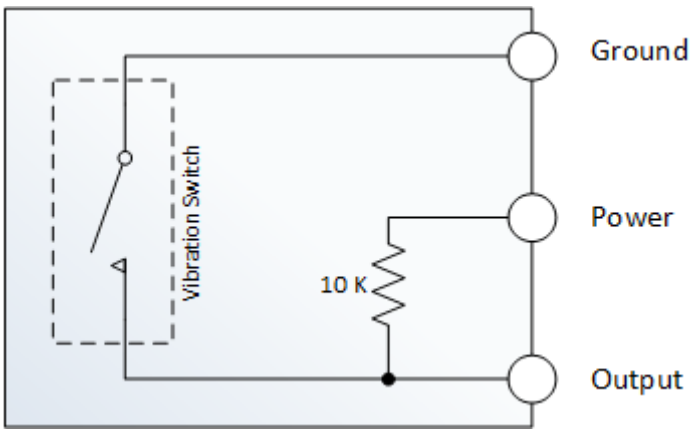
The connection between the terminals is momentary and will require a little thought as you implement it in your Arduino project.

Positioning of the switch is also important. Generally speaking the switch should be physically located as close as possible to the area being monitored. Otherwise, the vibration being detected may be dampened by other structural components in your project.

An exception to this rule may be where you find that the switch is too sensitive for your application. In this case, moving the switch further away from the area of interest may make it less sensitive.

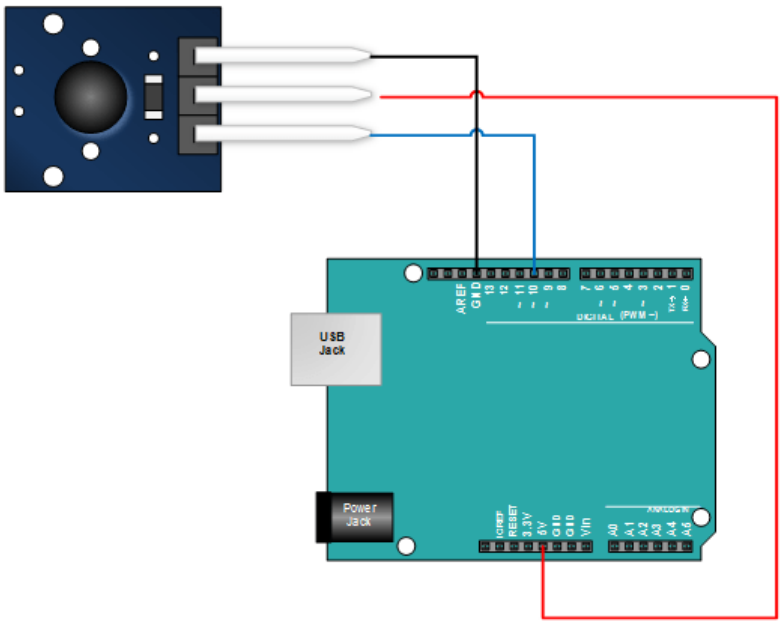
Module Schematic

As the schematic below shows, the module is nothing more than the switch and a pull up resistor. In fact, you could just as easily build your own with the Gaoxin switch alone.



Connect Your Arduino to the Sensor Module

Use the diagram below. You will only need three connections.



Copy Paste and Upload the Tutorial Sketch

In the sketch below:

- We read the input from the shock sensor
- If we detect a shock and we record the time the shock was detected. (see *lastShockTime*)
- If we were not already in an alarm state (see *bAlarm*) when a shock is detected, we set an alarm state and indicate that we're in an alarm state by sending an output to the serial monitor
- We exit the alarm state when the following conditions are satisfied
  - There is a high value measured at the sensor output
  - The difference between the high measurement and the last low measurement is greater then 250 mS (set by *shockAlarmTime*)
  - We are in an alarm state (see *bAlarm*)
- When we exit the alarm state, we set *bAlarm* to false and we send an output to the serial monitor

```
int shockPin = 10; // Use Pin 10 as our Input
int shockVal = HIGH; // This is where we record our shock measurement
boolean bAlarm = false;

unsigned long lastShockTime; // Record the time that we measured a shock

int shockAlarmTime = 250; // Number of milli seconds to keep the shock alarm high


void setup ()
{
  Serial.begin(9600);
  pinMode (shockPin, INPUT) ; // input from the sensor
}
void loop ()
{
  shockVal = digitalRead (shockPin) ; // read the value from our sensor

  if (shockVal == LOW) // If we're in an alarm state
  {
    lastShockTime = millis(); // record the time of the shock
    // The following is so you don't scroll on the output screen
    if (!bAlarm){
      Serial.println("Shock Alarm");
      bAlarm = true;
    }
  }
  else
  {
    if( (millis()-lastShockTime) > shockAlarmTime  &&  bAlarm){
      Serial.println("no alarm");
      bAlarm = false;
    }
  }
}
```

Run the Sketch and Verify the Output

After a successful upload, open your Arduino's serial monitor. With the serial monitor open, gently tap on the sensor.

When the Arduino registers a shock, it will be indicated on the serial monitor with a '**Shock Alarm**'.

After a period of time without registering a shock (set by *shockAlarmTime*), the serial monitor will indicate '**no alarm**'.

Rotary Encoder

The rotary encoder is a rotary input device (*as in knob*) that provides an indication of how much the knob has been rotated AND what direction it is rotating in. It's a great device for stepper and servo motor control. You could also use it to control devices like digital potentiometers.

Rotary Encoder Basics

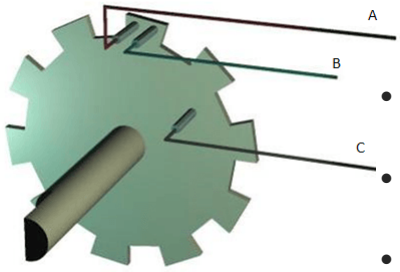
A rotary encoder has a fixed number of positions per revolution. These positions are easily felt as small "clicks" you turn the encoder. On one side of the switch there are three pins. They are normally referred to as A, B and C.

Inside the encoder there are two switches. Once switch connects pin A to pin C and the other switch connects pin B to C. In each encoder position, both switches are either opened or closed. Each click causes these switches to change states as follows:

If both switches are closed, turning the encoder either clockwise or counterclockwise one position will cause both switches to open

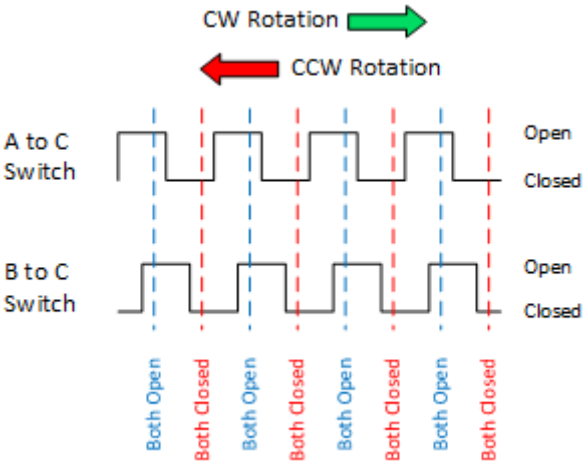
If both switches are open, turning the encoder either clockwise or counterclockwise one position will cause both switches to close.

The illustration below is representative of how the switch is constructed.



As you can see, the angular position of the A terminal and the B terminal is such that:

- **Rotating the switch clockwise** will cause the switch connecting A and C to change states first.
- **Rotating the switch counterclockwise** will cause the switch connecting B and C to change states first.
-

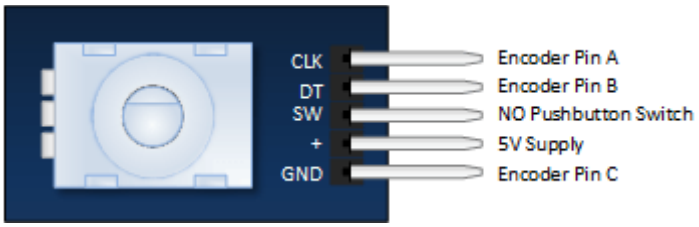


If we were to represent the opening and closing of the switches as wave forms, it would look something like this. Essentially, determining which switch changed states first is how the direction of rotation is determined.

If A changed states first, the switch is rotating in a clockwise direction.

If B changed states first, the switch is rotating in a counter clockwise direction.

Pinout



The module is designed so that a low is output when the switches are closed and a high when the switches are open.

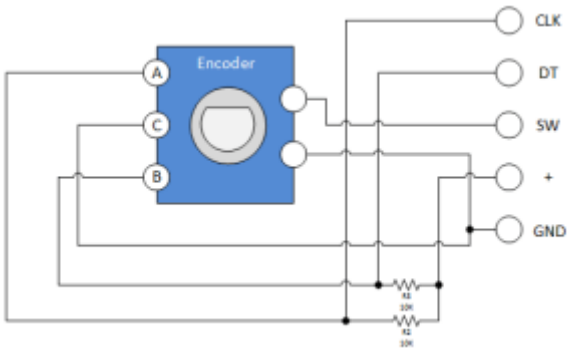
The low is generated by placing a ground at Pin C and passing it to the CLK and DT pins when switches are closed.

The high is generated with a 5V supply input and pullup resistors, such that CLK and DT are both high when switches are open.

Not previously mentioned is the existence of push button switch that is integral to the encoder. If you push on the shaft, a normally open switch will close. The feature is useful if you want to change switch function. For example, you may wish to have the ability to between coarse and fine adjustments.

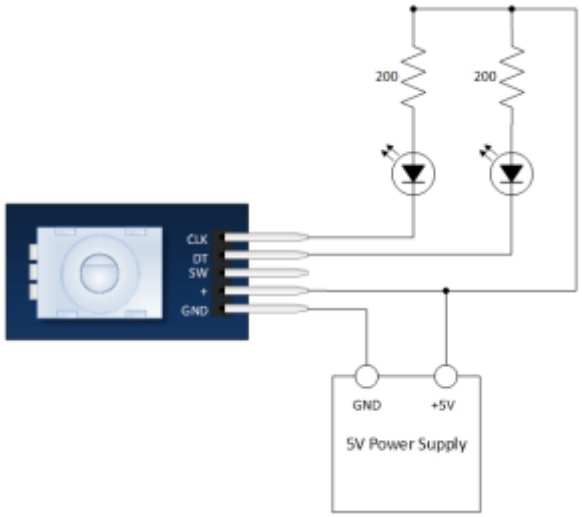
Rotary Encoder Schematic

A schematic for this module is provided below.



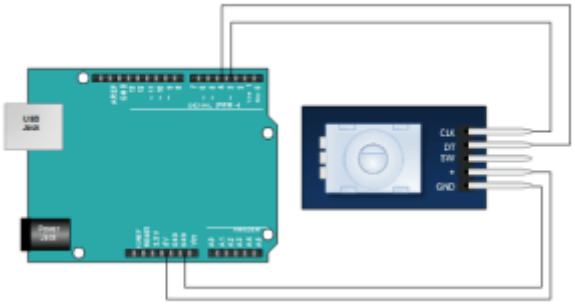
R2 and R3 in the schematic are pull up resistors.

Successfully implementing the Rotary Encoder into any project requires a clear understanding of everything that has been discussed thus far. If you're still a little fuzzy, you may wish to throw together the evaluation circuit illustrated below:



VERY SLOWLY rotate then encoder shaft both clockwise and counter clockwise. Notice which LEDs change state first with rotation.

Module Connection to the Arduino



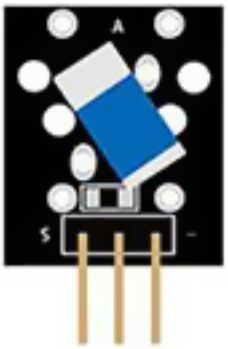
Pretty straight forward… All you need to do is connect four wires to the module.

The Arduino Sketch

This is a simple sketch that shows how to count the encoder position and how to determine direction of rotation. It has no switch de-bounce, nor does it use interrupts. A fully developed application might need to incorporate these in order to make it robust.

```
 int pinA = 3;  // Connected to CLK
 int pinB = 4;  // Connected to DT
 int encoderPosCount = 0;
 int pinALast;
 int aVal;
 boolean bCW;

 void setup() {
   pinMode (pinA,INPUT);
   pinMode (pinB,INPUT);
   /* Read Pin A
   Whatever state it's in will reflect the last position
   */
   pinALast = digitalRead(pinA);
   Serial.begin (9600);
 }

 void loop() {
   aVal = digitalRead(pinA);
   if (aVal != pinALast){ // Means the knob is rotating
     // if the knob is rotating, we need to determine direction
     // We do that by reading pin B.
     if (digitalRead(pinB) != aVal) {  // Means pin A Changed first - We're Rotating
Clockwise
       encoderPosCount ++;
       bCW = true;
     } else {// Otherwise B changed first and we're moving CCW
       bCW = false;
       encoderPosCount--;
     }
     Serial.print ("Rotated: ");
     if (bCW){
       Serial.println ("clockwise");
     }else{
       Serial.println("counterclockwise");
     }
     Serial.print("Encoder Position: ");
     Serial.println(encoderPosCount);

   }
   pinALast = aVal;
 }
```

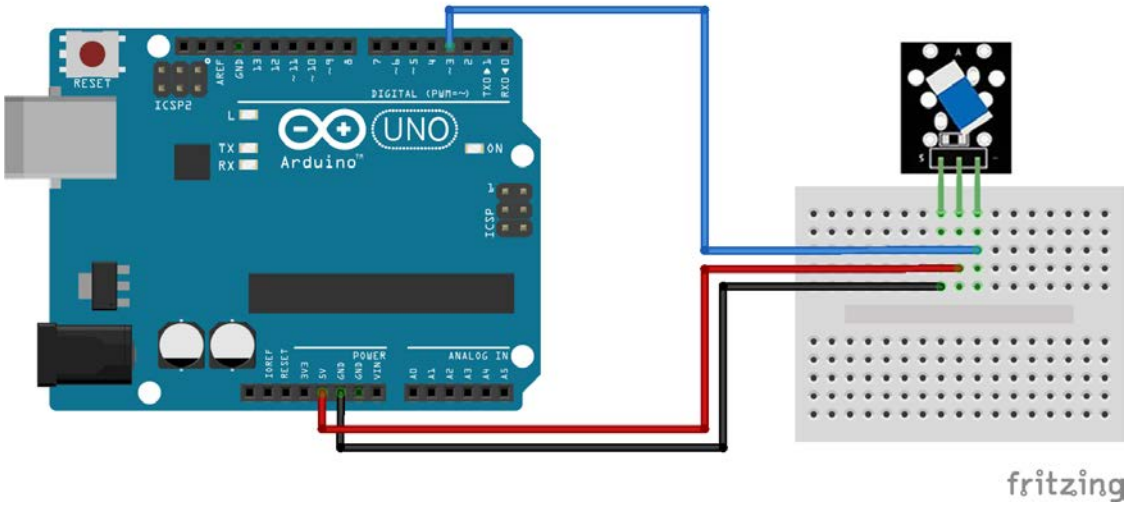
Mini Tilt Sensor Module



The tilt sensor will detect when the module is tilted. It works like a regular switch and has a little metal ball inside. When the sensor is angled, the ball will move and open the circuit. When the sensor is lying flat, the ball will rest and close the circuit.

Pinout and Connection to Arduino

Connect the Power line (middle) and ground (-) to +5 and GND respectively. Connect signal (DO) to pin 3 on the Arduino.



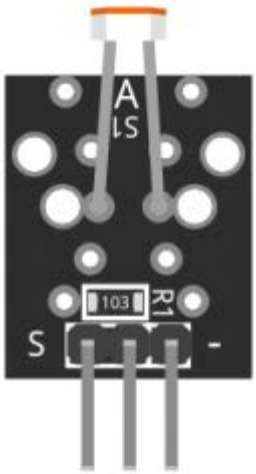
Arduino Example Sketch

The example sketch will light the LED when module is tilted.

```
int led = 13;//LED pin
int sensor = 3; //sensor pin
int val; //numeric variable

void setup() {
         pinMode(led, OUTPUT); //set LED pin as output
         pinMode(sensor, INPUT); //set sensor pin as input
}

void loop() {
         val = digitalRead(sensor); //Read the sensor
         if(val == HIGH)    { digitalWrite(Led, HIGH); }
         else               { digitalWrite(Led, LOW); }
}
```

Photoresistor Module

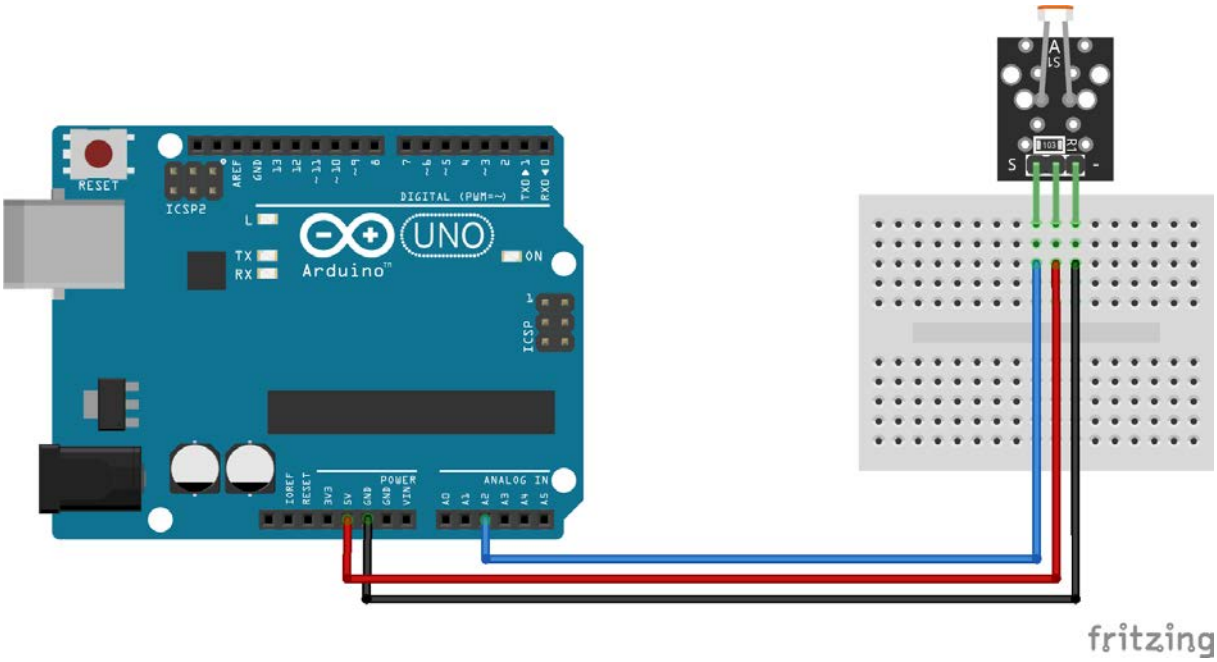
The photoresistor module is used to measure light intensity. It can determine the presence or absence of light.



This module consists of a photoresistor and a 10 kΩ in-line resistor. The photoresistors resistance will decrease in the presence of light and increase in the absence of it. The output is analog and determines the intensity of light.

Pinout and Connection to Arduino

Connect the Power line (middle) and ground (-) to +5 and GND respectively. Connect signal (S) to pin A2 on the Arduino.



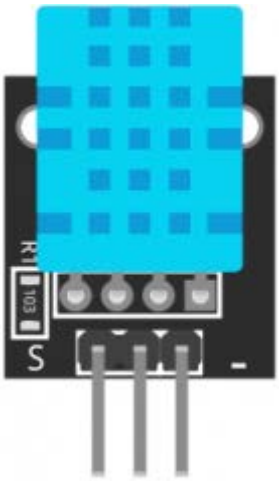
Arduino Example Sketch

The following Arduino sketch will output readings from the photoresistor, cover the module with your hand to prevent light on it and the output values will be low, point a light to the sensor and the values will be high.

```
int sensorPin = 2; //define analog pin 2
int value = 0;

void setup() {
        Serial.begin(9600);
}

void loop() {
        value = analogRead(sensorPin);
        Serial.println(value, DEC); // light intensity
                                    // high values for bright environment
                                    // low values for dark environment
        delay(100);
}
```

Temperature and Humidity Sensor Module

This temperature and humidity sensor module provides a digital serial interface to measure humidity and temperature.

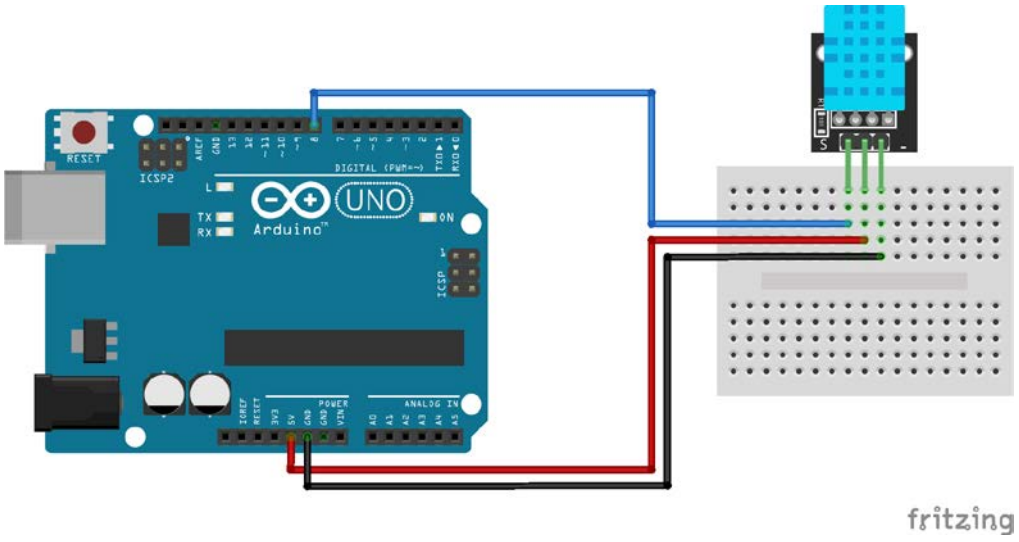


The module consists of a DHT11 digital humidity and temperature sensor and a 1 kΩ resistor. The DHT11 uses an internal thermistor and a capacitive humidity sensor to determine environment conditions, an internal chip is responsible for converting readings to a serial digital signal.

Operating Voltage	3.3V to 5.5V
Humidity range	20% to 90% RH
Humidity accuracy	±5% RH
Humidity resolution	1% RH
Temperature range	0ºC to 50ºC [32ºF to 122ºF]
Temperature accuracy	±2ºC
Temperature resolution	1ºC
Signal transmission range	20m

Pinout and Connection to Arduino

Connect the Power line (middle) and ground (-) to +5 and GND respectively. Connect signal (S) to pin 8 on the Arduino.



Arduino Example Sketch

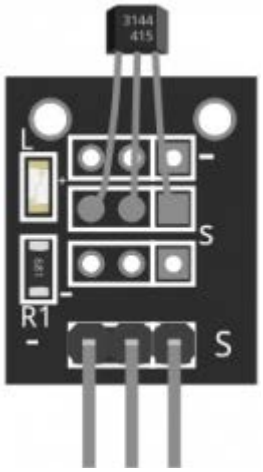
The following sketch uses pin 8 on the Arduino to serially send and receive data from the sensor. Serial communication is archived by sending specific high/low signals to the sensor and waiting for a response. Temperature and humidity data is read bit by bit and returned as an array of bytes.

```
int DHpin = 8; // input/output pin
byte dat[5];
byte read_data()
{
  byte data;
    for(int i = 0; i < 8; i++)
    {
      if(digitalRead(DHpin) == LOW)
      {
        while(digitalRead(DHpin) == LOW); // wait 50us;
        delayMicroseconds(30); // Duration of high level determine whether data is 0 or 1
        if(digitalRead(DHpin) == HIGH)
          data |= (1<<(7 - i)); //High in the former, low in the post;
        while(digitalRead(DHpin) == HIGH); //Data '1', waiting for next bit
      }
    }
  return data;
}

void start_test()
{
  digitalWrite(DHpin, LOW); //Pull down the bus to send the start signal;
  delay(30); //The delay is greater than 18 ms so that DHT 11 can detect the start signal;
  digitalWrite(DHpin, HIGH);
  delayMicroseconds(40); //Wait for DHT11 to respond;
  pinMode(DHpin, INPUT);
  while(digitalRead(DHpin) == HIGH);
  delayMicroseconds(80); //The DHT11 responds by pulling the bus low for 80us;
  if(digitalRead(DHpin) == LOW);
  delayMicroseconds(80); //DHT11 pulled up after the bus 80us to start sending data;
  for(int i = 0; i < 4; i++) //Receiving data, check bits are not considered;
    dat[i] = read_data();
  pinMode(DHpin, OUTPUT);
  digitalWrite(DHpin, HIGH); //After release of bus, wait for host to start next signal
}

void setup()
{
  Serial.begin(9600);
  pinMode(DHpin, OUTPUT);
}

void loop()
{
  start_test();
  Serial.print("Current humdity = ");
  Serial.print(dat[0], DEC); //Displays the integer bits of humidity;
  Serial.print('.');
  Serial.print(dat[1], DEC); //Displays the decimal places of the humidity;
  Serial.println('%');
  Serial.print("Current temperature = ");
  Serial.print(dat[2], DEC); //Displays the integer bits of temperature;
  Serial.print('.');
  Serial.print(dat[3], DEC); //Displays the decimal places of the temperature;
  Serial.println('C');
  delay(700);
}
```

Hall Magnetic Sensor Module

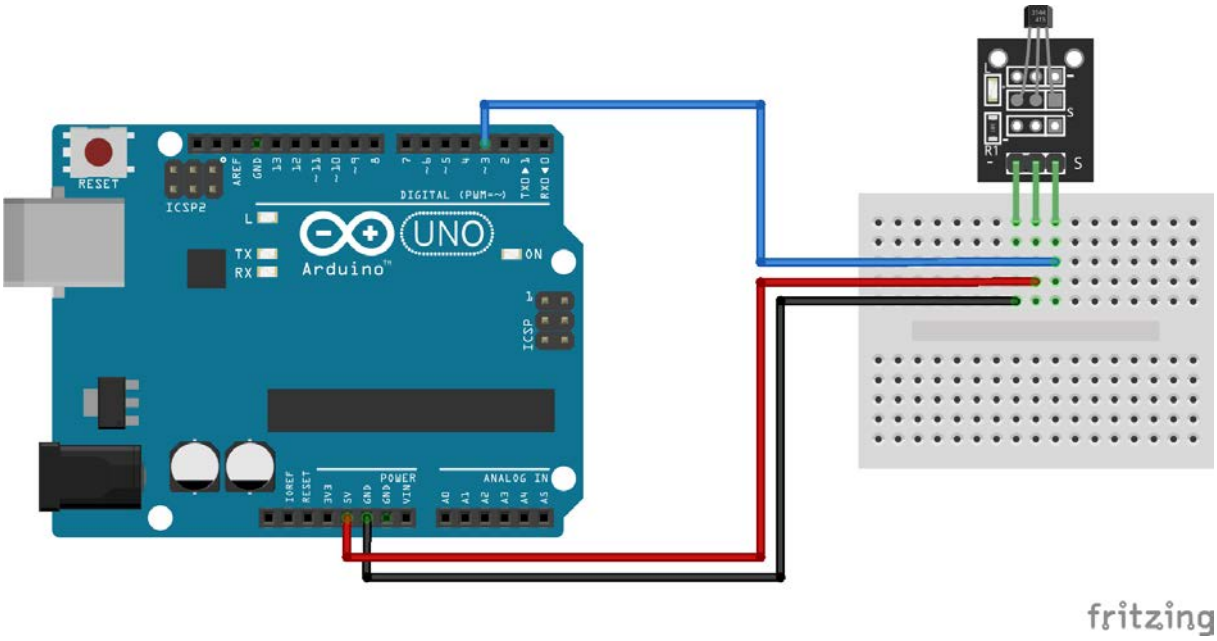
Hall Magnetic Sensor Module is a switch that will turn on/off in the presence of a magnetic field.

The sensor consists of a 3144EUA-S sensitive Hall-effect switch for high-temperature operation, a 680Ω resistor and a LED. It's Compatible with popular electronics platforms like Arduino and Raspberry Pi.



Pinout and Connection to Arduino

Connect the Power line (middle) and ground (-) to +5 and GND respectively. Connect signal (s) to pin 3 on the Arduino.

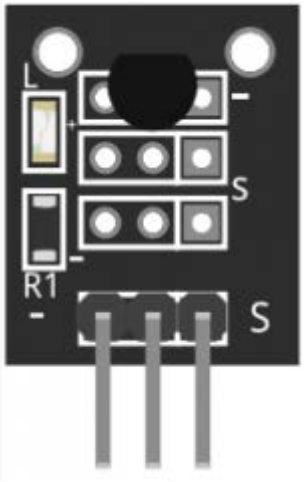


Arduino Example Sketch

The example sketch will light up the LED on pin 13 when a magnetic field is detected.

```
int led = 13;//LED pin
int sensor = 3; //sensor pin
int val; //numeric variable

void setup()
{
         pinMode(led, OUTPUT); //set LED pin as output
         pinMode(sensor, INPUT); //set sensor pin as input
}

void loop()
{
         val = digitalRead(sensor); //Read the sensor
         if(val == HIGH) //when magnetic field is detected, turn led on
         {
                  digitalWrite(Led, HIGH);
         }
         else
         {
                  digitalWrite(Led, LOW);
         }
}
```

Temperature Sensor Module



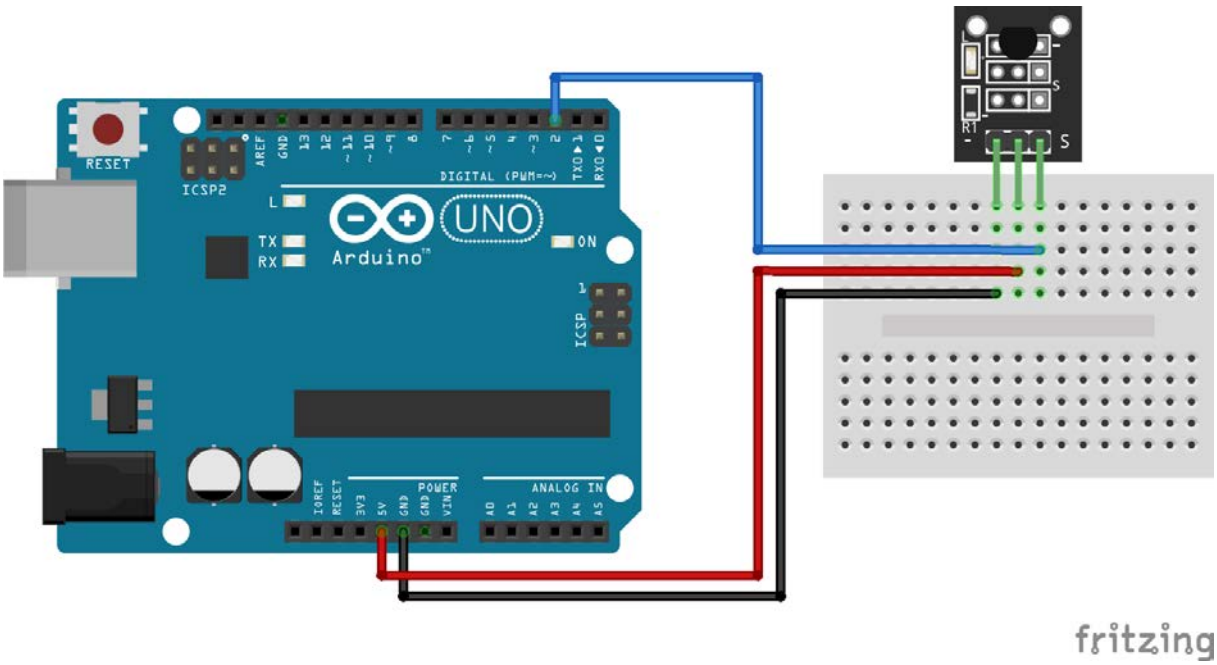
The temperature sensor module allows ambient temperature measurement using digital serial bus.

Temperature Sensor Module consists of a DS18B20 single-bus digital temperature sensor, a LED and a resistor. It's compatible with popular electronics platforms like Arduino, Raspberry Pi and Esp8266.

Operating Voltage	3.0V to 5.5V
Temperature	-55°C to 125°C [-57°F to 257°F]
Measurement Accuracy	±0.5°C
Dimensions	18.5mm x 15mm [0.728in x 0.591in]

Pinout and Connection to Arduino

Connect the Power line (middle) and ground (-) to +5 and GND respectively. Connect signal (S) to pin 2 on the Arduino.



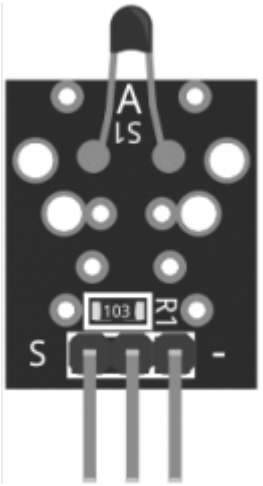
Arduino Example Sketch

The following Arduino sketch will use the OneWire library to serially communicate with the sensor and output the temperature via the serial monitor.

```
#include <OneWire.h>
#include <DallasTemperature.h>

// Data wire is plugged into pin 2 on the Arduino
#define ONE_WIRE_BUS 2

// Setup a oneWire instance to communicate with any OneWire devices
// (not just Maxim/Dallas temperature ICs)
OneWire oneWire(ONE_WIRE_BUS);
// Pass our oneWire reference to Dallas Temperature.
DallasTemperature sensors(&oneWire);

void setup(void)
{
  // start serial port
  Serial.begin(9600);
  Serial.println("Dallas Temperature IC Control Library Demo");
  // Start up the library
  sensors.begin();          // IC Default 9 bit.
                            // If you have troubles consider upping it 12.
                            // Ups the delay giving the IC more time to process
                            // the temperature measurement
}

void loop(void)
{
  // call sensors.requestTemperatures() to issue a global temperature
  // request to all devices on the bus
  Serial.print("Requesting temperatures...");
  sensors.requestTemperatures(); // Send the command to get temperatures
  Serial.println("DONE");

  Serial.print("Temperature for Device 1 is: ");
  Serial.print(sensors.getTempCByIndex(0));
  // Why "byIndex"?
  // You can have more than one IC on the same bus.
  // 0 refers to the first IC on the wire
}
```

Analog Temperature Sensor module

Analog Temperature Sensor module measures ambient temperature based on resistance of the thermistor.

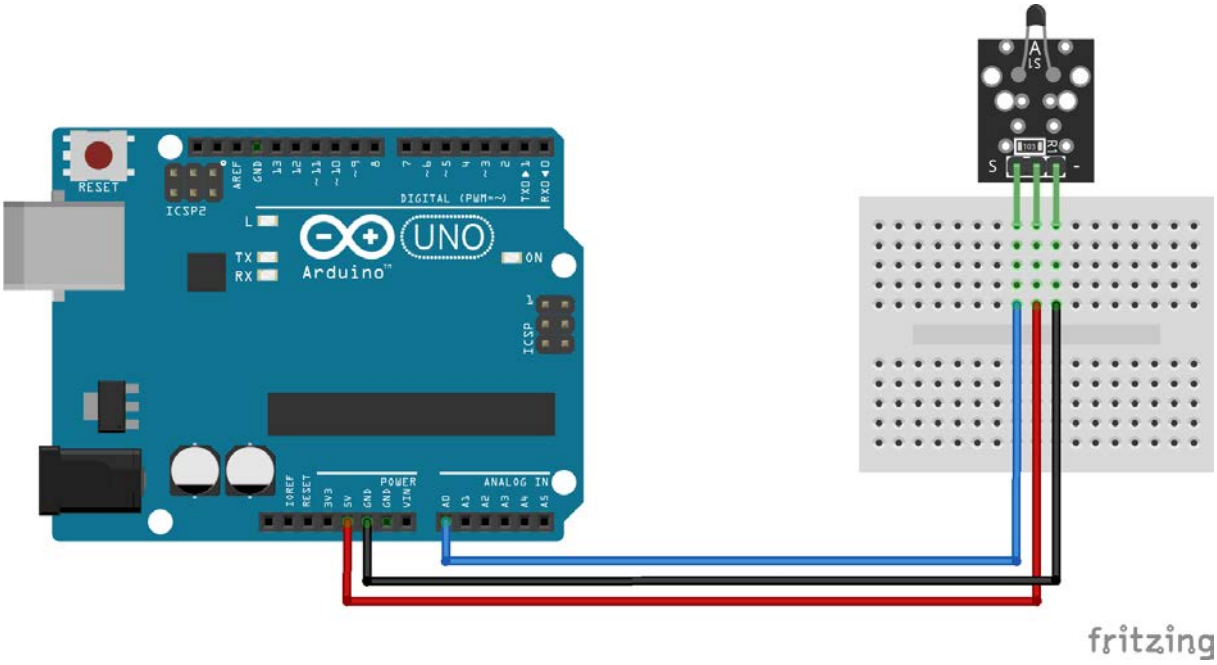


The Sensor module consists of a NTC thermistor and a 10 kΩ resistor. The resistance of the thermistor varies with surrounding temperature. We'll use the Steinhart–Hart equation to derive precise temperature of the thermistor.

Operating Voltage	5V
Temperature	-55°C to 125°C [-67°F to 257°F]
Accuracy	±0.5°C

Pinout and Connection to Arduino

Connect board's power line (middle) and ground (-) to 5V and GND respectively. Connect signal (S) to pin A0 on the Arduino.



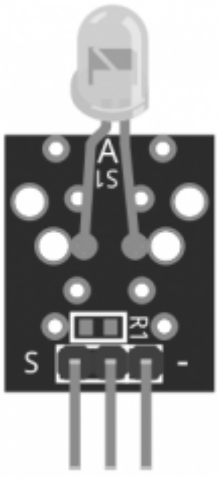
Arduino Example Sketch

The following Arduino Sketch will derive the temperature of the thermistor using the Steinhart-Hart equation implemented in the function Thermister.

```
#include <math.h>

double Thermister(int RawADC) {
          double Temp;
          Temp = log(((10240000/RawADC) - 10000));
          Temp = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * Temp * Temp ))* Temp
);
          Temp = Temp - 273.15; // Convert Kelvin to Celcius
          return Temp;
}

void setup() {
          Serial.begin(9600);
}

void loop() {
          Serial.print(Thermister(analogRead(0))); //read pin A0
          Serial.println("c");
          delay(500);
}
```

Infrared Transmitter Module

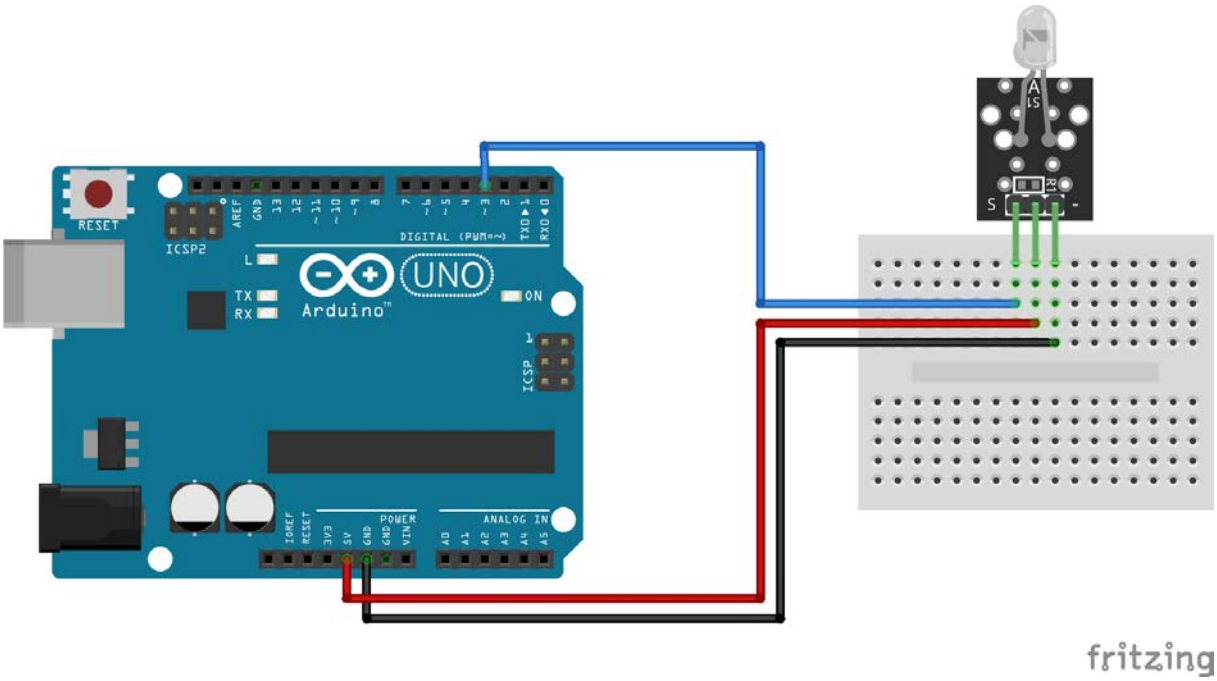
The Infrared Transmitter Module consists of just a 5mm IR LED. It works together with any 38 kHz receiver and almost any device using regular IR remote control. It's compatible with popular electronics platforms like Arduino, Teensy, Raspberry Pi and ESP8266.



Operating Voltage	5V
Forward Current	30 ~ 60 mA
Power Consumption	90mW
Operating Temperature	-25°C to 80°C [-13°F to 176°F]
Dimensions	18.5mm x 15mm [0.728in x 0.591in]

Pinout and Connection to Arduino

Connect the Power line (middle) and ground (-) to +5 and GND respectively. Connect signal (S) to pin 3 on the Arduino UNO or pin 9 on the Arduino Mega. The pin number for the infrared transmitter is determined by the IRremote library, check the download section below for more info



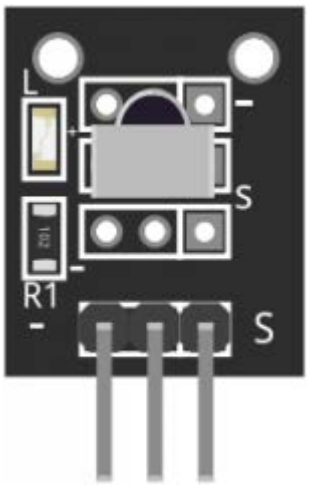
Arduino Example Sketch

The following Arduino sketch uses the IRremote library to serially send infra-red signals. The output pin is determined by the library and it depends on the board you are using, check the IRremote library documentation for supported boards. You'll need an IR receiver like to process the signal.

```
#include <IRremote.h>
IRsend irsend;

void setup()
{
 Serial.begin(9600);
}

void loop()
{
   for (int i = 0; i < 50; i++) {
     irsend.sendSony(0xa90, 12); // Sony TV power code
     delay(40);
   }
}
```

38 kHz IR Receiver Module

IR receiver module reacts to 38 kHz modulated infrared light.

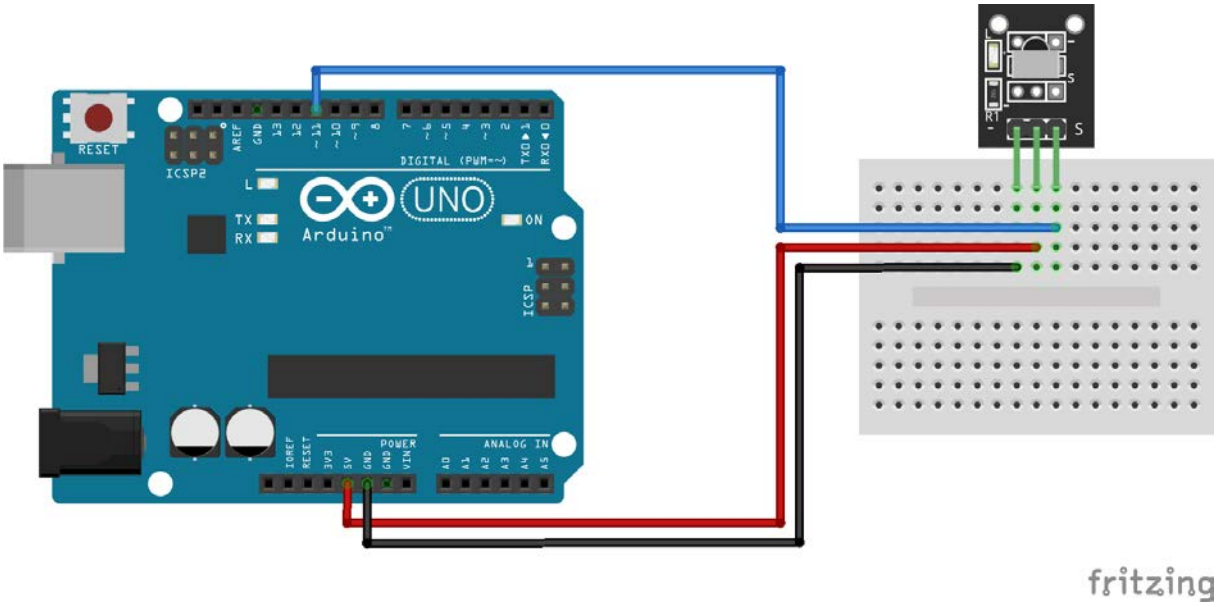


This module consists of a 38 kHz IR receiver, a 1kΩ resistor and an LED. It works together with a 38 kHz IR transmitter or any regular IR remote control. It's compatible with popular electronic platforms like Arduino, Raspberry Pi and ESP8266.

Operating Voltage	2.7 to 5.5V
Operating Current	0.4 to 1.5mA
Reception Distance	18m
Reception Angle	±45º
Carrier Frequency	38KHz
Low Level Voltage	0.4V
High Level Voltage	4.5V
Ambient Light Filter	up to 500LUX

Pinout and Connection to Arduino

Connect the Power line (middle) and ground (-) to +5 and GND respectively. Connect signal (S) to pin 11 on the Arduino. Line un IR receiver and transmitter.



Arduino Example Sketch

The following Arduino sketch uses the IRremote library to receive and process infra-red signals. Use the IR transmitter module to serially send data to this module.

```
#include <IRremote.h>

int RECV_PIN = 11; // define input pin on Arduino
IRrecv irrecv(RECV_PIN);
decode_results results; // decode_results class is defined in IRremote.h

void setup() {
         Serial.begin(9600);
         irrecv.enableIRIn(); // Start the receiver
}

void loop() {
         if (irrecv.decode(&results)) {
                  Serial.println(results.value, HEX);
                  irrecv.resume(); // Receive the next value
         }
         delay (100); // small delay to prevent reading errors
}
```

Vibration Shake Sensor

The Vibration Sensor allows you to use an Arduino to detect impacts, shocks or shaking.

When the switch detects a jolt, the output of the module is sent low.

Functional Description

The switch primarily consists of a terminal that forms a center post and a second terminal that is a spring that surrounds the center post.

When a sufficient force is transferred to the switch, the terminal consisting of the spring moves and shorts both terminals together.

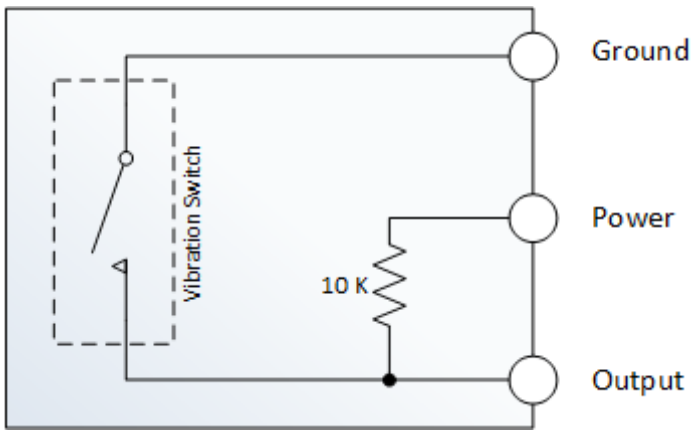
The connection between the terminals is momentary and will require a little thought as you implement it in your Arduino project.

Positioning of the switch is also important. Generally speaking the switch should be physically located as close as possible to the area being monitored. Otherwise, the vibration being detected may be dampened by other structural components in your project.

An exception to this rule may be where you find that the switch is too sensitive for your application. In this case, moving the switch further away from the area of interest may make it less sensitive.

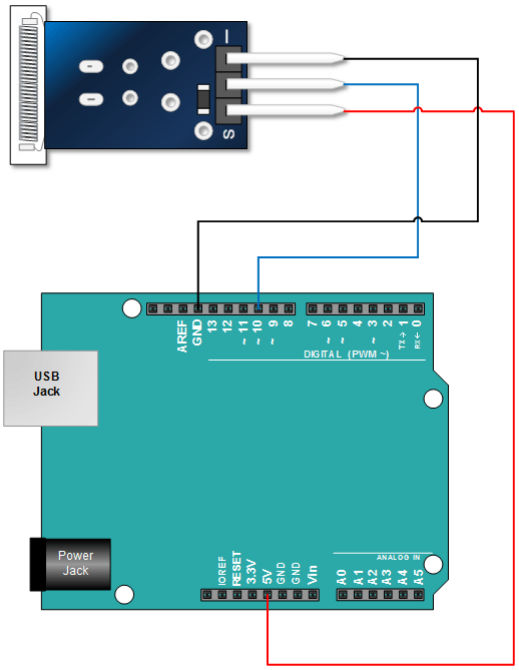
Module Schematic

As the schematic below shows, the module is nothing more than the switch and a pull up resistor. In fact, you could just as easily build your own with the Gaoxin switch alone.



Pinout and Connection to Arduino

Use the diagram below. You will only need three connections. Connect + to 5V, - to Ground and S to pin 10 on Arduino.



Arduino Example Sketch

In the sketch below:

- We read the input from the shock sensor
- If we detect a shock and we record the time the shock was detected. (see *lastShockTime*)
- If we were not already in an alarm state (see *bAlarm*) when a shock is detected, we set an alarm state and indicate that we're in an alarm state by sending an output to the serial monitor
- We exit the alarm state when the following conditions are satisfied
  - There is a high value measured at the sensor output
  - The difference between the high measurement and the last low measurement is greater than 250 mS (set by *shockAlarmTime*)
  - We are in an alarm state (see *bAlarm*)
- When we exit the alarm state, we set *bAlarm* to false and we send an output to the serial monitor

```
int shockPin = 10; // Use Pin 10 as our Input
int shockVal = HIGH; // This is where we record our shock measurement
boolean bAlarm = false;

unsigned long lastShockTime; // Record the time that we measured a shock

int shockAlarmTime = 250; // Number of milli seconds to keep the shock alarm high


void setup ()
{
  Serial.begin(9600);
  pinMode (shockPin, INPUT) ; // input from the sensor
}
void loop ()
{
  shockVal = digitalRead (shockPin) ; // read the value from our sensor

  if (shockVal == LOW) // If we're in an alarm state
  {
    lastShockTime = millis(); // record the time of the shock
    // The following is so you don't scroll on the output screen
    if (!bAlarm){
      Serial.println("Shock Alarm");
      bAlarm = true;
    }
  }
  else
  {
    if( (millis()-lastShockTime) > shockAlarmTime  &&  bAlarm){
      Serial.println("no alarm");
      bAlarm = false;
    }
  }
}
```

Run the Sketch and Verify the Output

After a successful upload, open your Arduino's serial monitor. With the serial monitor open, gently tap on the sensor.

When the Arduino registers a shock, it will be indicated on the serial monitor with a '**Shock Alarm**'.

After a period of time without registering a shock (set by *shockAlarmTime*), the serial monitor will indicate '**no alarm**'.

Photo interrupter Module

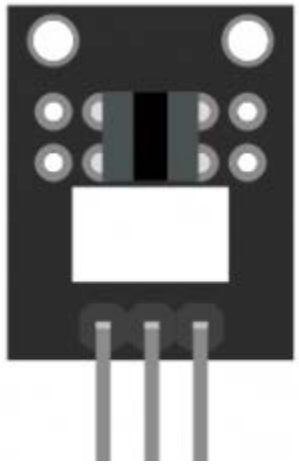


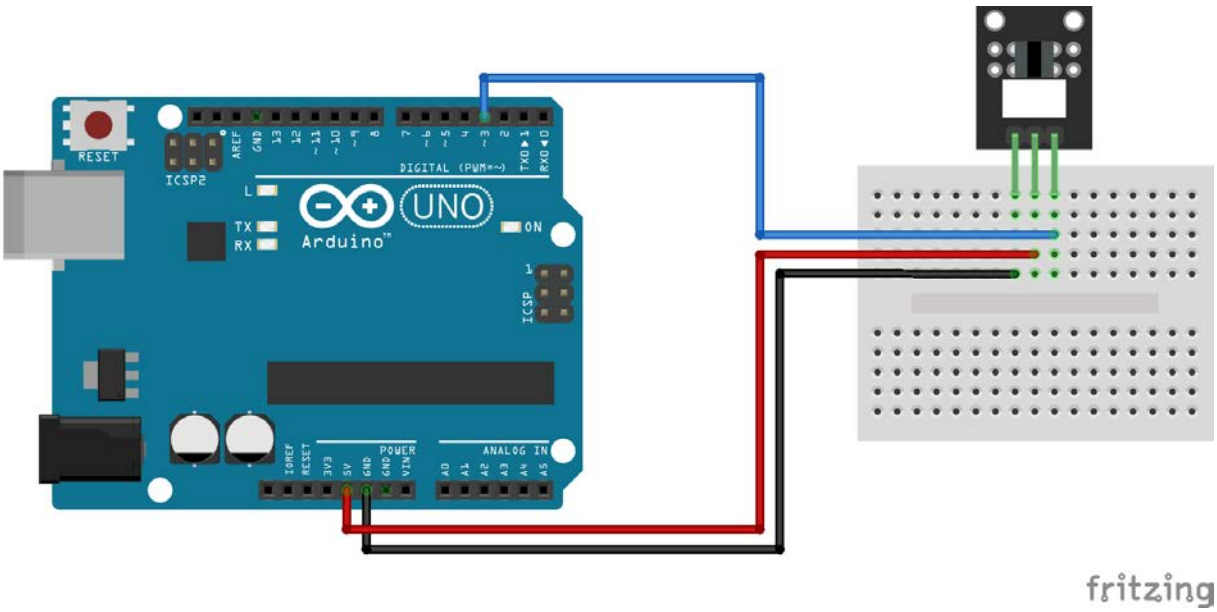
Photo Interrupter Module for Arduino, will trigger a signal when light between the sensor's gap is blocked.

The photo Interrupter module consists of an optical emitter/detector in the front and two resistors (1 kΩ and 33 Ω) in the back. The sensor uses a beam of light between de emitter and detector to check if the path between both is being blocked by an opaque object.

Operating Voltage	3.3 – 5V
Dimensions	18.5mm x 15mm [0.728in x 0.591in]

Pinout and Connection to Arduino

Connect the power line (middle) and ground (left) to +5V and GND respectively. Connect signal (S) to pin 3 on the Arduino.



Arduino Example Sketch

The following sketch will light up the LED (pin 13) on the Arduino when there's an object blocking the beam of light between the sensors gap.

```
int Led = 13; // define LED pin
int buttonpin = 3; // define photo interrupter signal pin
int val; //define a numeric variable

void setup()
{
        pinMode(Led, OUTPUT); // LED pin as output
        pinMode(buttonpin, INPUT); //photo interrupter pin as input
}

void loop()
{
        val=digitalRead(buttonpin); //read the value of the sensor
        if(val == HIGH) // turn on LED when sensor is blocked
        {
                digitalWrite(Led,HIGH);
        }
        else
        {
                digitalWrite(Led,LOW);
        }
}
```