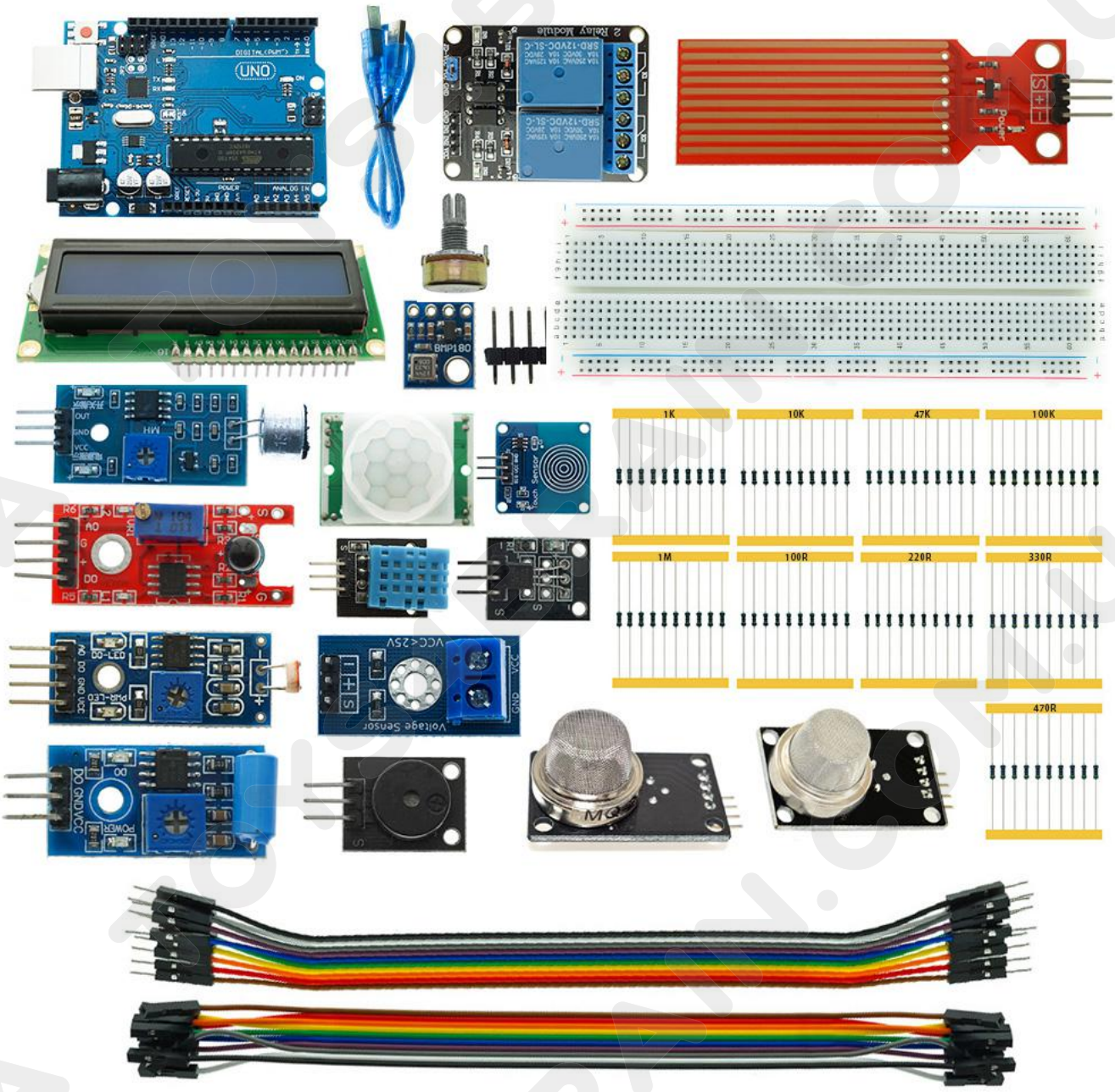
ARDUINO Smart Home Suite Tutorial

V1.0.19.9.24

Smart Home Kit List



Content

Section 1 DHT11 Temperature and Humidity Sensor

Introduction:

The digital temperature and humidity sensor DHT11 contain a chip that can perform analog-to-digital conversion and send out digital signals with temperature and humidity. It is compatible with any MCU and is ideal for those who need some basic data recording. It is very popular for electronics enthusiasts, because it is very cheap but still offers excellent performance.

In this course, we will first understand some background knowledge of humidity and then explain how DHT11 measures humidity. After that, we will show you how to connect DHT11 to ARDUINO and provide some sample code so that you can use DHT11 in your own project.

Preparation of work

Hardware

ARDUINO board x 1

DHT11 sensor x 1

LCD1602 x 1

10k resistor x 1

Breadboard x 1

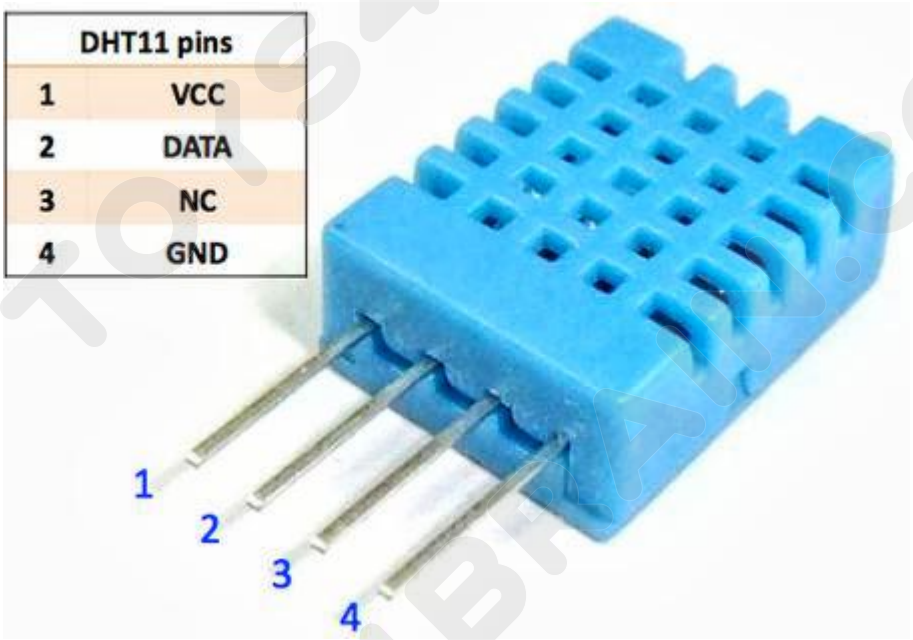
Jumper wire

USB cable x 1

PC x 1

About DHT11

DHT11 is a basic low-cost digital temperature and humidity sensor , It uses a capacitive humidity sensor and a thermistor to measure ambient air and send a digital signal on the data pin (no analog input pin required)。It's easy to use but need more patient to get the data.



Only 3 pins can be used：VCC，GND & DATA. The communication process begins from data line transmits start signal to DHT11 and the sensor receives signal and response. The host then receives responding signal and 40 bits humidity data. (8-bit humidity integer + 8-bit humidity decimal + 8-bit temperature integer + 8-bit temperature decimal + 8-bit checksum).

Product parameter relative humidity:

Resolution: 16Bit repeatability: ±1% RH

Accuracy: 25 ° C ± 5% RH

Interchangeability: Fully interchangeable response time: 1 / e (63%) at 25 ° C for 6 s

1m / s air 6s hysteresis: <±0.3%RH

Long-term stability: <±0.5% RH / year temperature:

Resolution: 16Bit repeatability: ±0.2°C range: at 25°C±2°C

Response time: 1 / e (63%) 10S electrical characteristics Power supply: DC 3.5~5.5V

Supply current: measuring 0.3mA standby 60μA

Sampling period: 2 seconds or more Pin Description:

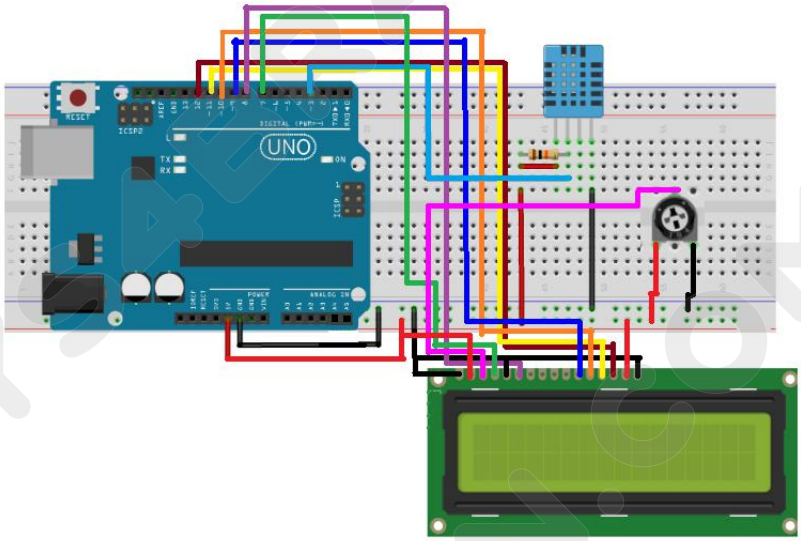
1.VDD power supply 3.5~5.5V DC

2.DATA serial data, single bus

3.NC, empty needle

4.GND ground wire, negative power supply

Example wiring diagram:



Sample code:

```
/*
  * LCD RS pin to digital pin 12
  * LCD Enable pin to digital pin 11
  * LCD D4 pin to digital pin 5
  * LCD D5 pin to digital pin 4
  * LCD D6 pin to digital pin 3
  * LCD D7 pin to digital pin 2
  * LCD R/W pin to ground
```

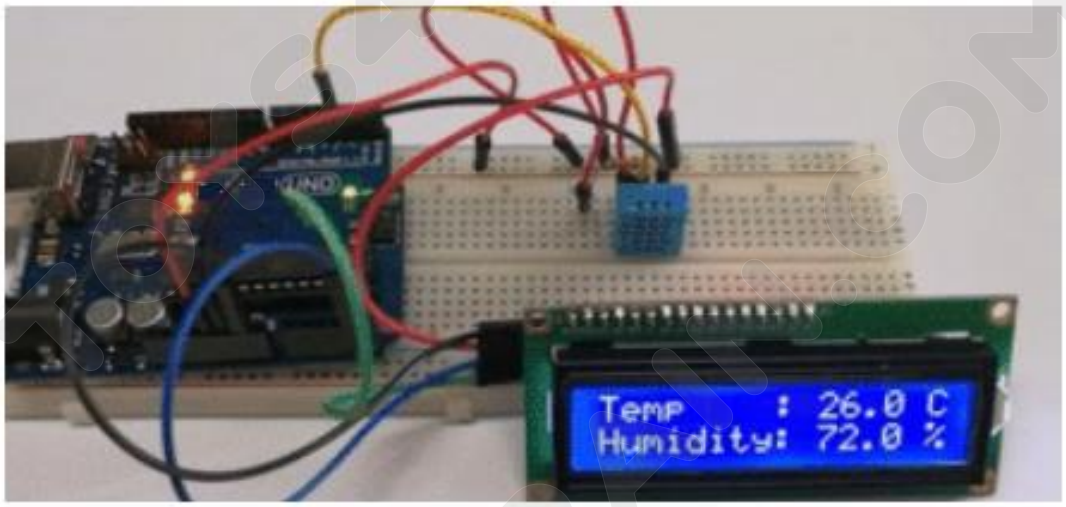
```
  * 10K resistor:
  * ends to +5V and ground
  * wiper to LCD VO pin (pin 3)
*/
#include <LiquidCrystal.h>
#include <dht11.h>
#define DHT11PIN 8
dht11 DHT11;
// initialize the library with the numbers of the interface pins
Liqui dCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
   pinMode(DHT11PIN,OUTPUT);
   // set up the LCD's number of columns and rows:
   lcd.begin(16, 2);
}

void loop() {
   int chk = DHT11.read(DHT11PIN);
   lcd.setCursor(0, 0);
   lcd.print("Tep: ");
   lcd.print((float)DHT11.temperature, 2);
   lcd.print("C");
   // set the cursor to column 0, line 1
   // (note: line 1 is the second row, since counting begins with 0):
   lcd.setCursor(0, 1);
   // print the number of seconds since reset:
   lcd.print("Hum: ");
   lcd.print((float)DHT11.humidity, 2);
   lcd.print("%");
   delay(200);
}
```

Result

After a few seconds of uploading, you will now read the current humidity and temperature values on LCD.

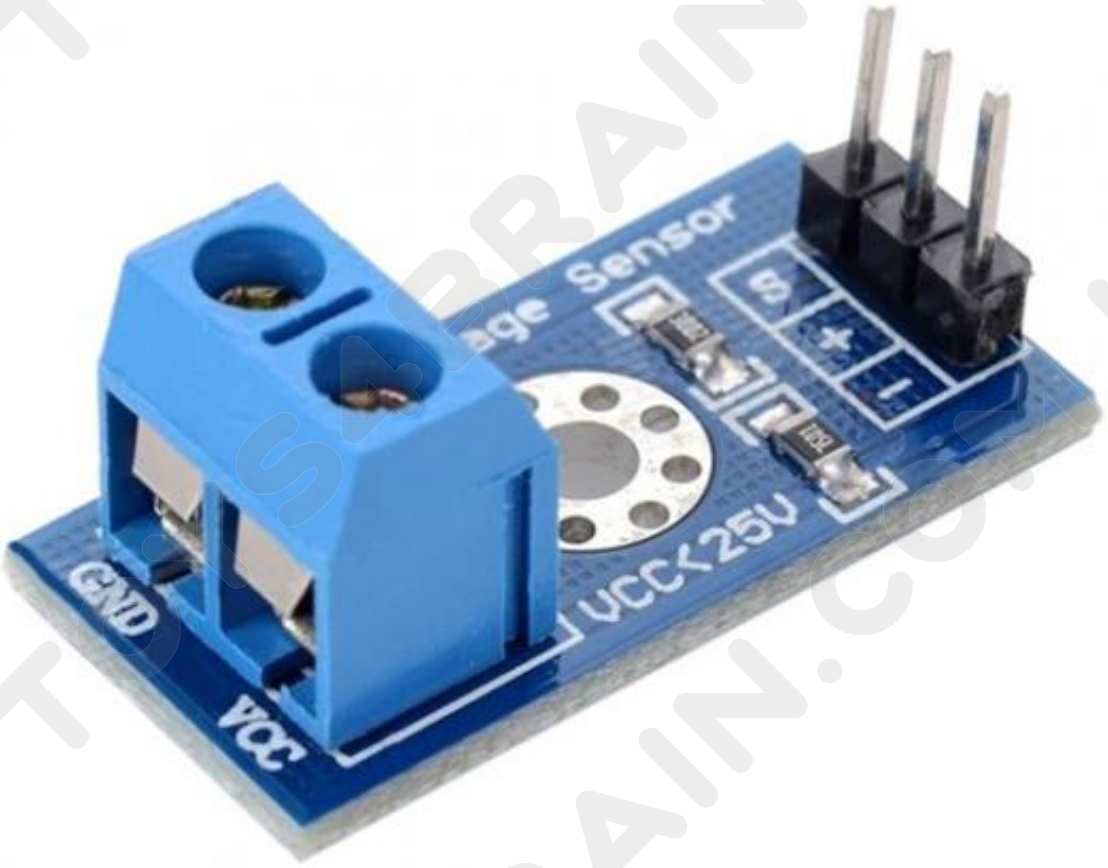


Section 2 ARDUINO & Voltage sensor module

Introduction:

In this project, we will learn how to use ARDUINO to measure voltage by connecting voltage senor with ARDUINO board. With this voltage sensor interface, you can measure voltages up to 25V.

Warning: If using the same voltage sensor module, make sure its input voltage (voltage to be measured) is limited to 25V.



Preparation Work

Hardware

1、ARDUINO R3 x 1

2、Voltage sensor module x 1

3、800 tie bread board x 1

4、Jumper wire

5、USB cable x 1

6、PC x 1

Overview of voltage detection sensors

A simple and very useful module that uses a voltage divider to reduce any input voltage by five times. This allows you to use the microcontroller's analog input to monitor voltages much higher than it can sense.， For example, using the 0-5V analog input range, you can measure voltage up to 25V. The module also includes convenient screw terminals for easy and safe connection of wires.

If you want to measure external voltage with ARDUINO, you must use the analog input pin of ARDUINO board. If you recall the ARDUINO analog pins, their input voltage is limited to 5V, which means you can directly measure up to 5V using its analog input pins.

But what if you want to measure voltage greater than 5V? You cannot use the ARDUINO analog input pins directly because you might burn the ATmega328P IC on ARDUINO ARDUINO board (or the related microcontroller IC, depending on the ARDUINO board you are using).

This is the rescue voltage sensor module. Using this voltage sensor module, you can measure up to 25V of voltage.

Voltage sensor pin

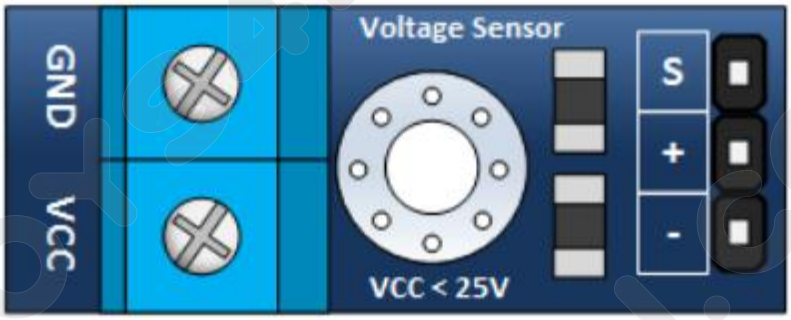
Before going into the details of the voltage sensor functions and schematics, let me outline the available pins for voltage sensor module.

Basically, the 25V voltage sensor is same as voltage sensor used here, with a total of 5 pins. Two of them are located on double-pin screw terminal and three are male head row pins.

The screw terminal pins are labeled VCC and GND and they must be connected to an external voltage source, the voltage to be measured.

The other three male pin headers are labeled S, + and -. The S pin is “Sense” pin and must be connected to analog input pin of ARDUINO. The "-" pin must be connected to GND. Pin marked "+" does not need to be connected to anything (it is the N/C pin).

The figure below shows the pins of voltage sensor module.



Input

GND: - Connect to the low end of the voltage being measured.

Warning!: This is the same electrical point as your ARDUINO ground.

VCC: being used to connect high voltage side of the voltage being measured.

Output

S: This is connected to your ARDUINO analog input.

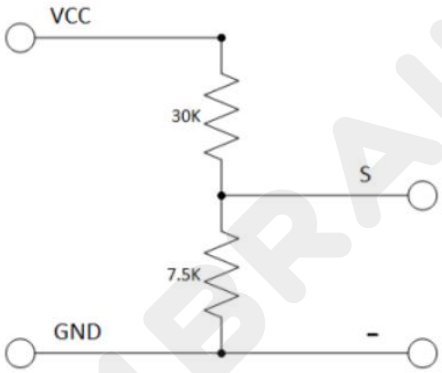
-: This connects to your ARDUINO ground.

+: This is not connected.

Voltage sensor schematic diagram

Now, let's talk about the important thing about voltage sensor: Schematic diagram. The voltage sensor is basically a voltage divider consisting of two resistors with a resistance of 30KΩ and 7.5KΩ, which is a 5:1 voltage divider.

The figure below shows the schematic of the voltage sensor module with an input voltage limit of 25V.



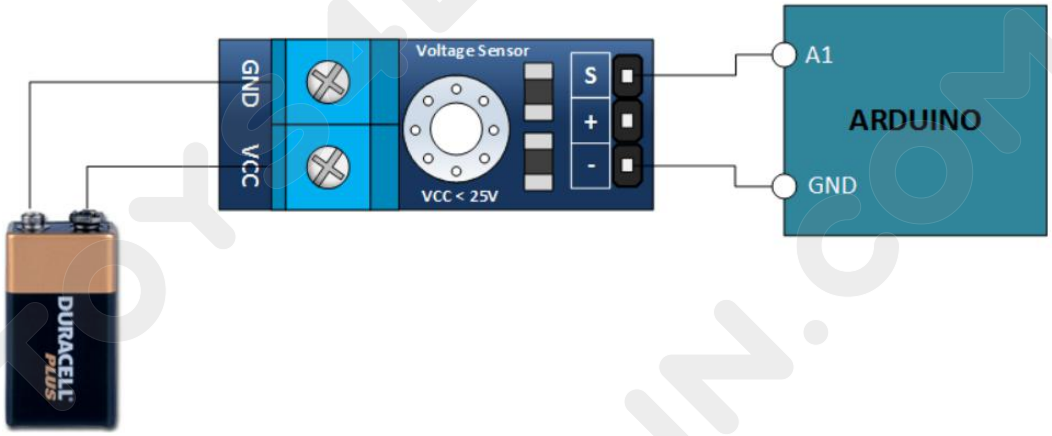
Make your own voltage sensor:

Using a pre-built voltage sensor module is very simple, and if you don't have one, you can easily build one yourself. You only need two resistors.

If you want to make your own voltage sensor, like this one that can measure up to 25V, then you must have resistors of 30KΩ and 7.5KΩ.

Example:

Find a 9-volt battery (or any DC device with a voltage of 0-25v) and connect it to your voltage sensor module and ARDUINO, as shown below.



Code:

Once this is done, connect the Ardunio board to your computer with a USB cable. The green power LED (labeled PWR) should be turned on. Open ARDUINO IDE and select the appropriate board type and port type for your project. Then load the following sketch onto ARDUINO.

```
int analogInput = A1;
float vout = 0.0;
float vin = 0.0;
float R1 = 30000.0; //
float R2 = 7500.0; //

int value = 0;
void setup(){
    pinMode(analogInput, INPUT);
    Serial.begin(9600);
    Serial.print("DC VOLTMETER");
}
void loop(){
    // read the value at analog input
    value = analogRead(analogInput);
    vout = (value * 5.0) / 1024.0; // see text
```

```
    vin = vout / (R2/(R1+R2));

Serial.print("INPUT V= ");
Serial.println(vin,2);
delay(500);
}
```

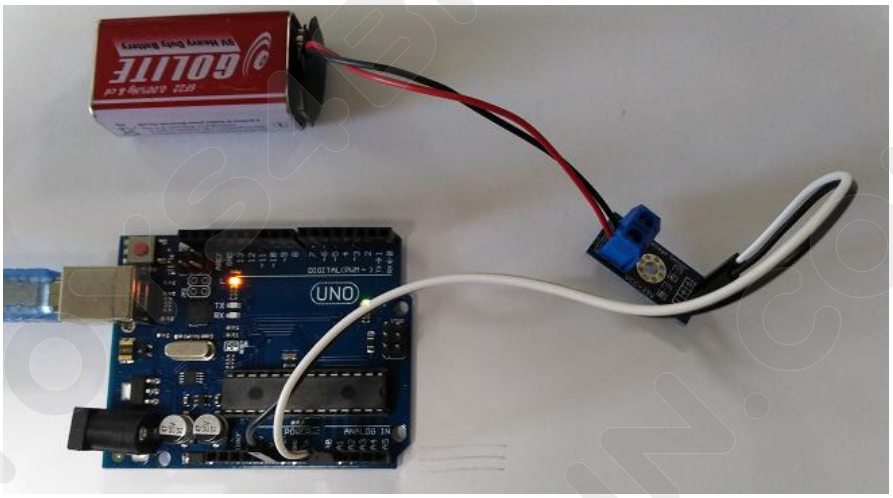
Working principle:

Since the voltage sensor module is basically a voltage divider circuit, you can use the formula to calculate the input voltage.

Vin = Vout *(R2 /(R1 + R2))

Here R1 = 30000, R2 = 7500, and Vout can be calculated from the ARDUINO's analog input by using Vout = (analog value * 5/1024).

Picture of real products:



Result:

The following is the result of using a 9v dry battery. I measured it with a multimeter, which is actually 8.05v. You can adjust the values in above code, the resistance value is the value in the ideal world, but you will find that the resistance has tolerances, which may vary. You can use a multimeter to measure resistance, for low-cost measurement methods such as 9v batteries

or 12v DC power supplies.

(Declaration: The 9V battery used in experiment was tested after a period of time, the original voltage is 8.9V)

Section 3 MQ-2 Gas Sensor

Introduction:

In this project, we will discuss how to build a smoke sensor circuit by using an ARDUINO board.

The smoke sensor we will use is MQ-2. This is a sensor that is sensitive not only to smoke but also to flammable gases.

The MQ-2 smoke sensor reports smoke through its output voltage level. The more smoke, the greater output voltage. Instead, the less smoke it exposes, the lower the output voltage.

Preparation work

Hardware

1、 ARDUINO R3 x 1

2、I2C LCD 1602 display x 1

3、F / M jumper wire

4、USB cable x 1

5、PC x 1

About MQ2 Gas sensor

The MQ-2 smoke sensor is sensitive to smoke and the following flammable gases:

LPG

Butane

Propane

Methane

alcohol

hydrogen

The resistance of the sensor varies depending on the type of gas.

The smoke sensor has a built-in potentiometer that adjusts sensitivity of sensor based on the accuracy of gas you are testing.

Features

1. Wide range of detection

2. High sensitivity and fast response

3. Long life and stability

4. The drive circuit is simple

Due to its fast response time and high sensitivity, measurements can be taken as quickly as possible. A potentiometer can be used to adjust the sensor sensitivity.

标准工作条件

符号	参数名称	技术条件	备注
$V_{C.}$	电路电压	5V±0.1	AC或DC
$V_{H.}$	加热电压	5V±0.1	AC或DC
$R_{L.}$	负载电阻	可调整的	
$R_{H.}$	加热器阻力	33Kohm±5%	室内温度
$P_{H.}$	供暖消耗	不到800mW	

环境条件

符号	参数名称	技术条件	备注
$T_{O.}$	工作温度	-20℃-50℃	
$T_{S.}$	储存温度	-20℃-70℃	
RH	相对湿度	<95%	
O_2	氧气浓度	21%（标准条件）氧气浓度会影响灵敏度	最低值为2%

灵敏度特征

符号	参数名称	技术条件	备注
$R_{S.}$	传感器电阻	3Kohm-30Kohm（1000ppm异丁烷）	检测浓度范围： 200ppm-5000ppm LPG和丙烷 300ppm-5000ppm丁烷 5000ppm-20000ppm甲烷 300ppm-5000ppm H_2 100ppm-2000ppm酒精
α（3000ppm / 1000ppm异丁烷）	浓度斜率	≤0.6	
标准检测条件	温度：20°C±2°C VC：5V±0.1 湿度：65%±5%VH：5V±0.1		
预热时间	超过24小时		

Working principle:

MQ2 has an electrochemical sensor that changes its resistance to different concentrations of various gases. The sensor is connected in series with a variable resistor to form a voltage divider circuit for varying sensitivity. When one of these gaseous elements comes into contact with the sensor after heating, the resistance of the sensor changes. A change in resistance changes the voltage across the sensor, which can be read by the microcontroller. By knowing the reference voltage and resistance of another resistor, the voltage value can be used to find resistance of the sensor. Sensor has different sensitivities to different types of gases. For different types of gases, the sensitivity characteristic curve is shown below.

The voltage output from sensor varies with the level of smoke/gas present in the atmosphere. The sensor outputs a voltage proportional to the smoke/gas concentration.

In other words, the relationship between voltage and gas concentration is as follows:

Larger gas concentration, larger output voltage

Lower gas concentration, lower output voltage

Working Mechanism：

The output can be an analog signal (A0) that can be read using ARDUINO's analog input or digital output (D0) and can be read using ARDUINO's digital input.

Attention：

The sensor value only reflects the approximate trend of the gas concentration within the allowable error range, it does not represent the exact gas concentration. Detecting certain components in the air often requires more accurate and expensive instruments than a single gas sensor can do. We do not recommend the use of this gas sensor if your project aims to obtain gas concentrations at very precise levels.

Gas Detection: Basic example

In this example, the sensor is connected to the AO pin. Displays the voltage read from the sensor. This value can be used as a threshold to detect increase/decrease in gas concentration.

```
void setup() {
    Serial.begin(9600);
}void loop() {
    float sensor_volt;
    float sensorValue;
    sensorValue = analogRead(A0);
    sensor_volt = sensorValue/1024*5.0;
    Serial.print("sensor_volt = ");
    Serial.print(sensor_volt);
    Serial.println("V");
    delay(1000);
}
```

Measurement Approximation:

These examples illustrate the way to understand the approximate concentration of a gas. According to the MQx sensor data sheet, these equations were tested under standard conditions and not calibrated. It may vary depending on changes in temperature or humidity. Keep the gas sensor in a clean air environment. Upload the code below.

```
void setup() {
      Serial.begin(9600);
}void loop() {
      float sensor_volt;
      float RS_air; //    Get the value of RS through clear air
      float R0;    // Get the value of R0 in H2
     float sensorValue;
     /*--- Get average data by testing 100 times ---*/
      for(int x = 0 ; x < 100 ; x++)
      {
            sensorValue = sensorValue + analogRead(A0);
      }
      sensorValue = sensorValue/100.0;
      /*-----------------------------------------*/
      sensor_volt = sensorValue/1024*5.0;
      RS_air = (5.0-sensor_volt)/sensor_volt; //
      R0 = RS_air/9.8; // Clear the ratio of RS/R0 in the air from Graph (found using WebPlotDigitizer) to 9.8
      Serial.print("sensor_volt = ");
      Serial.print(sensor_volt);
      Serial.println("V");
      Serial.print("R0 = ");
      Serial.println(R0);
      delay(1000);
}
```

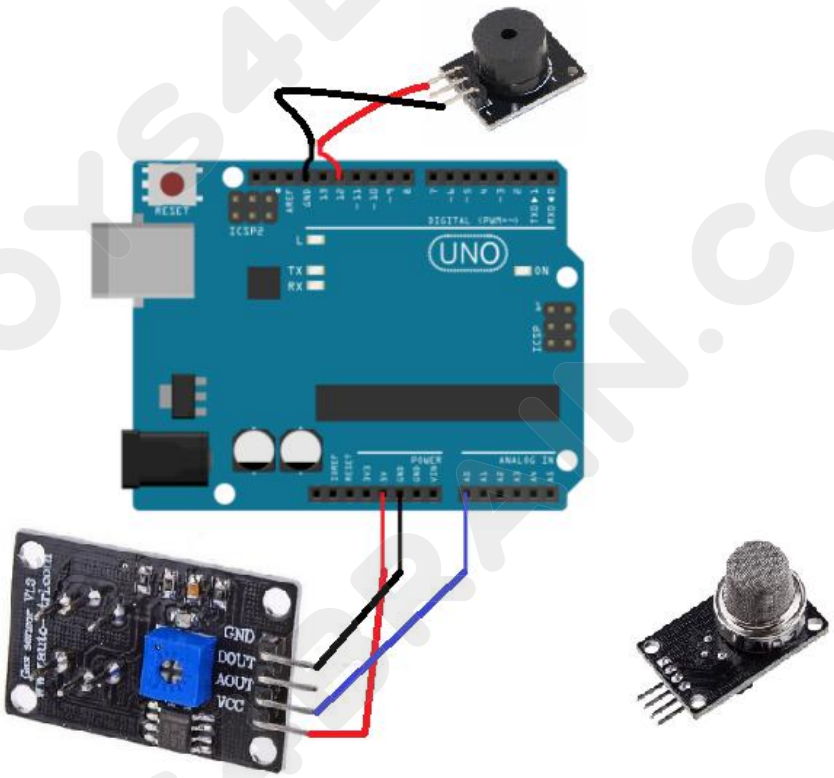
Then, open the ARDUINO IDE's serial monitor. Note down the value of R0, which will be used in the next program. Write R0 after the reading is stable. Replace R0 with the R0 value tested above. Expose the sensor to any of the gases listed above.

```
void setup() {
      Serial.begin(9600);
}void loop() {
      float sensor_volt;
```

```
    float RS_gas; // Get value of RS in a GAS
    float ratio; // Get ratio RS_GAS/RS_air
    int sensorValue = analogRead(A0);
    sensor_volt=(float)sensorValue/1024*5.0;
    RS_gas = (5.0-sensor_volt)/sensor_volt; // omit *RL
        /*-Replace the name "R0" with the value of R0 in the demo of First
Test -*/
    ratio = RS_gas/R0;   // ratio = RS/R0
        /*-----------------------------------------------------------------*/
    Serial.print("sensor_volt = ");
    Serial.println(sensor_volt);
    Serial.print("RS_ratio = ");
    Serial.println(RS_gas);
    Serial.print("Rs/R0 = ");
    Serial.println(ratio);
    Serial.print("\n\n");
    delay(1000);
}
```

Example：ARDUINO MQ-2 Smoke detector

The circuit we will connection as follows:



Therefore, in order to power smoke sensor, we connected smoke sensor pin 2

to 5V terminal of ARDUINO and terminal 3 to the GND terminal. This requires the smoke sensor to provide 5 volts.

The output of the sensor enters the analog pin A0 of ARDUINO. Through this connection, ARDUINO can read the sensor's analog voltage output. The ARDUINO board has a built-in analog-to-digital converter, so analog values can be read without any external ADC chip.

Depending on the value read by ARDUINO, the operation that the circuit will take is determined. We will explain in our code that if voltage output from sensor is above a certain threshold, the buzzer will alarm to warn users that smoke has been detected.

ARDUINO MQ-2 Smoke sensor circuit code

Since we have just looked through the schematic diagram of the smoke sensor circuit, all we need to know is the code that needed to upload to ARDUINO so that this smoke alarm works.

The code we need to upload is shown below.

```
/* Apply MQ-2 smoke sensor circuit code built with ARDUINO board*/
const int sensorPin= 0;
const int buzzerPin= 12;
int smoke_level;

void setup() {
Serial.begin(115200); //Set the baud rate of data transmission in bits/ second
pinMode(sensorPin, INPUT);//The smoke sensor will be the input of ARDUINO
pinMode(buzzerPin, OUTPUT);// buzzer is used for output in circuit
}
void loop() {
smoke_level= analogRead(sensorPin); // ARDUINO reads the smoke sensor value
Serial.println(smoke_level); //Print for debugging purposes only, to see what the sensor is looking at
if(smoke_level > 200){ //If the smoke concentration is greater than 200, the buzzer will turn on digitalWrite(buzzerPin, HIGH);

}
```

```
else
{
digitalWrite(buzzerPin, LOW);
}
}
```

The first code block declares and initializes three variables. Sensor Pin represents smoke sensor. It is initialized to 0 because it will connect to the analog pin A0 of ARDUINO board. The next variable buzzerPin represents the pin of the buzzer anode connection; it is initialized to 12 because it will be connected to the digital pin D12 of the ARDUINO board. The variable smoke_level represents the amount of smoke drawn by the smoke sensor.

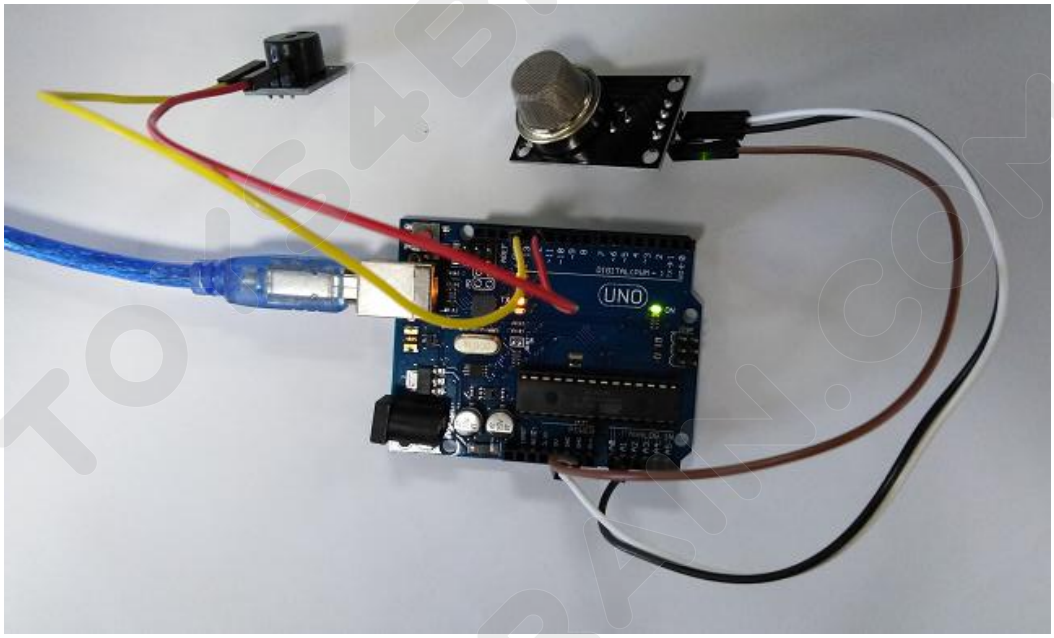
The next code block defines the baud rate of the circuit, as well as its input and output. Sensor Pin is the smoke sensor pin that is used as an input to the circuit. The sensor is input into ARDUINO so that ARDUINO can read and process the value. BuzzerPin is used as the output. If the smoke level is above a certain threshold, the output of the circuit, the buzzer will start.

The next code block reads value from sensorPin (smoke sensor) using the analogRead() function. This will be a value from 0 to 1023. 0 means smoke free, and 1023 represents absolute maximum and highest level of smoke. So the variable smoke_level represents the smoke level, ranging from 0 to 1023. We printed a line for debugging purposes to print this value so that you can see the value returned from this function. In our code, we set it to say that if the smoke level is above 200, we will trigger the buzzer to emit a sound by sending digital pin D12 to a high level. So 200 is our threshold. If the smoke concentration is lower than this, the buzzer will not start.

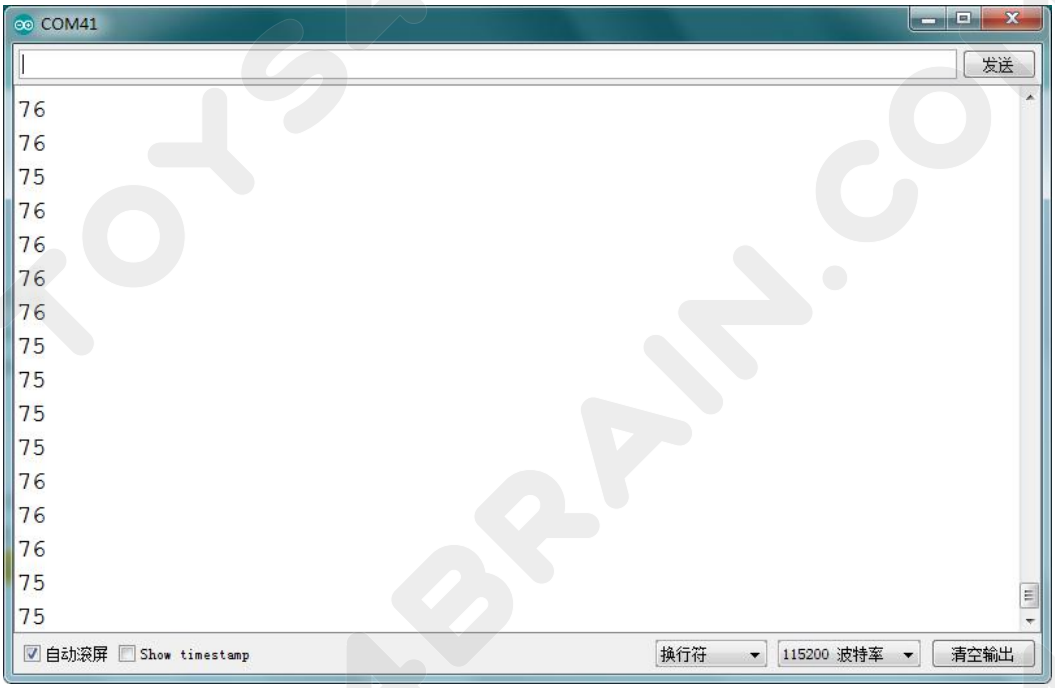
The last piece of code is the loop() function. This is part of the code that repeats over and over in an infinite loop. This means that our code always checks what the smoke_level is so that it can know if the buzzer is triggered.

Physical Picture:

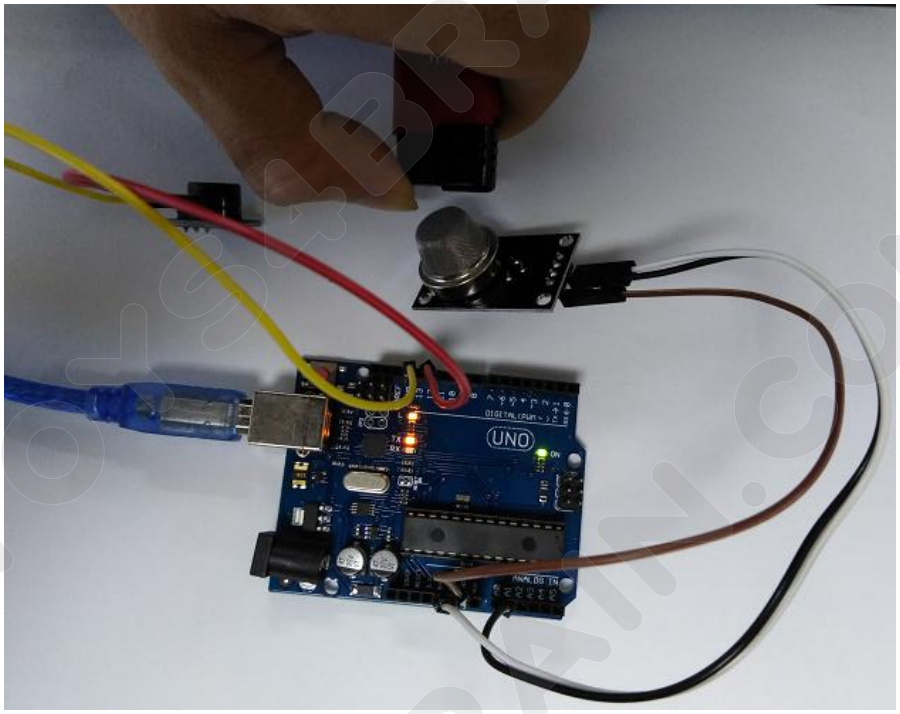
(1) No gas detected:



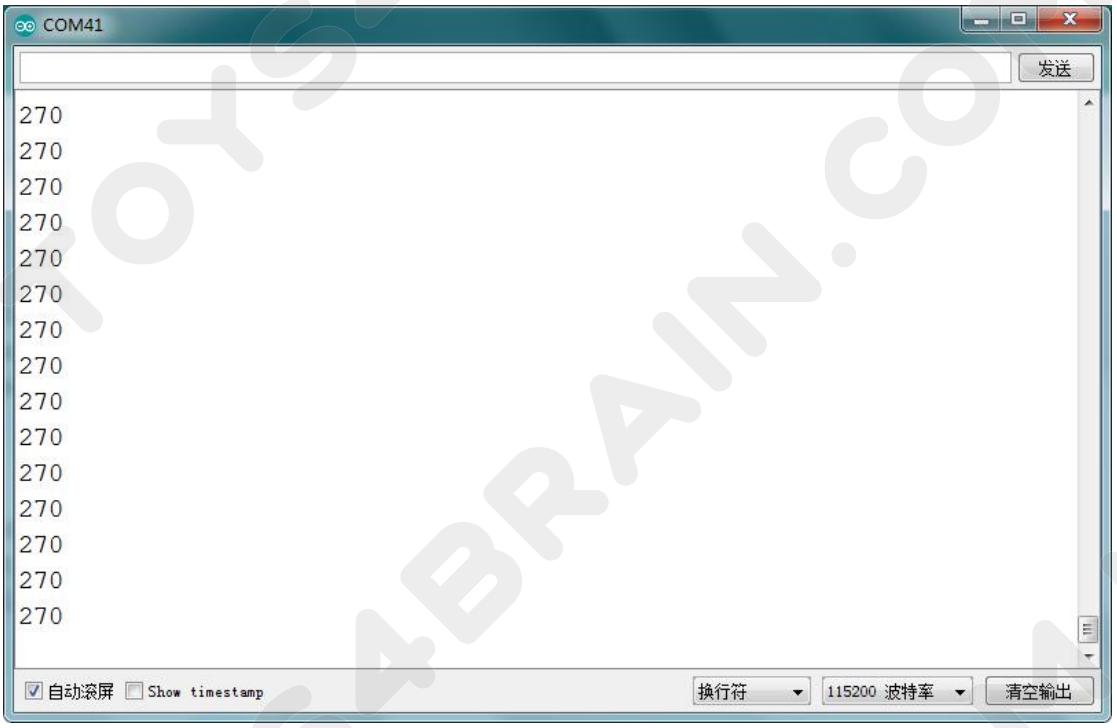
The serial monitor shows:



(2) Gas detected:



The serial monitor shows:



Section 4 MQ-5 Gas Sensor

Introduction

In this project, we will show what the MQ-5 sensor is and how to use it on the ARDUINO board.

Work preparation

Hardware

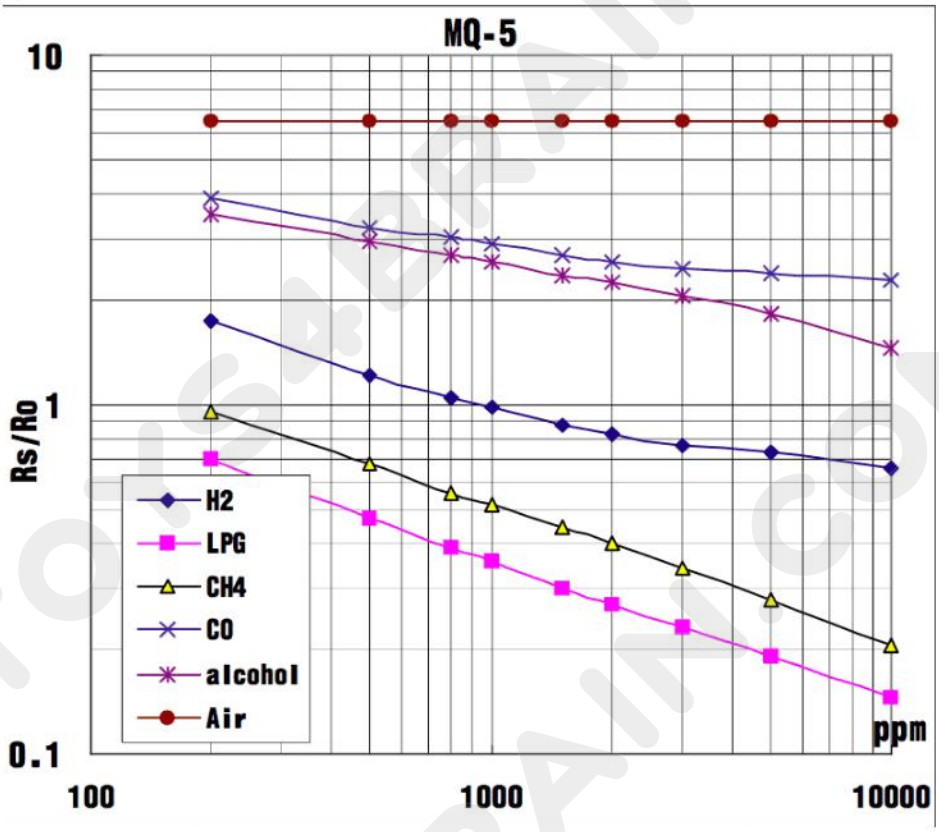
1. UNO R3 x 1
2. MQ-5 sensor x 1
3. jumper
4. USB cable x 1
5. PC x 1

About MQ-5 Gas Sensor

The gas sensing material used in the MQ-5 gas sensor is SnO2, which has low electrical conductivity in clean air. When combustible gas exists in the environment where the sensor is located, the electrical conductivity of sensor increases with the concentration of combustible gas in the air. By using simple circuit, changes in electrical conductivity can be converted into an output signal corresponding to the gas concentration.

The MQ-5 gas sensor is highly sensitive to propane, propane and methane and is able to detect methane and propane as well. The sensor can detect a variety of combustible gases, especially natural gas, making it a low-cost sensor for a variety of applications.

In this tutorial, we use the MQ5 gas sensor module (widely used in the market). The module has two output possibilities - analog output (A0) and digital output (D0). The analog output can be used to detect gas leaks and measure gas leakage in specific units (eg, ppm) (by properly calculating the sensor output within the program). The digital output can be used to detect gas leaks that trigger an alarm system (such as an audible alarm or SMS activation). Digital output only provides two possible outputs - high and low (so it is more suitable for detecting gas leaks than for measuring gas presence).



规格

项目	参数	敏	典型	马克斯	单元
VCC	工作电压	4.9	五	5.1	V
PH	供暖消耗	0.5	-	800	毫瓦
RL	负载电阻		可调整的		
RH	加热器阻力	-	31±10%	-	Ω
卢比	感应阻力	10	-	60	千欧
范围	检测浓度	200	-	10000	PPM

- 响应时间（tres）：≤10S
- 恢复时间（TREC）：≤30S

Feature:

Low cost

High sensitivity to LPG and natural gas

Flat response

Stable longevity

Digital and analog output

On-board LED indicator

Applications: Household gas leak detector, industrial combustible gas detector, portable gas detector, gas leak alarms, gas detector

Note:

The sensor value only reflects approximate trend of the gas concentration within a allowable error range, it does not represent the exact gas concentration. Detecting certain components in the air often requires more accurate and expensive instruments than a single gas sensor can do. We do not recommend the use of this gas sensor if your project is designed to achieve gas concentrations at very precise levels.

Example: Connecting MQ5 Gas Sensor Module to ARDUINO Using a Digital Output Pin

This is very simple. Connect the MQ5 module D0 pin to any digital pins of ARDUINO. Let's connect D0 to ARDUINO's pin 7. Now we need to supply the power (Vcc) and complete the circuit by grounding (Gnd). Please refer to the circuit diagram given below. Get the +5V connection from ARDUINO and connect it to MQ5 module's Vcc. Finally, connect the GND pin of the MQ5 module to the GND of the ARDUINO. That's all, we have completed the circuit.

Circuit diagram of the interface between MQ5 and ARDUINO (digital output)

MQ-5 Sensor	UNO R3
VCC	+ 5V
GND	GND
D0	D7

Note: - The MQ5 sensor has a warm-up requirement. We recommend keeping the sensor energized (from ARDUINO) for about 15 minutes before applying gas to it.

Code:

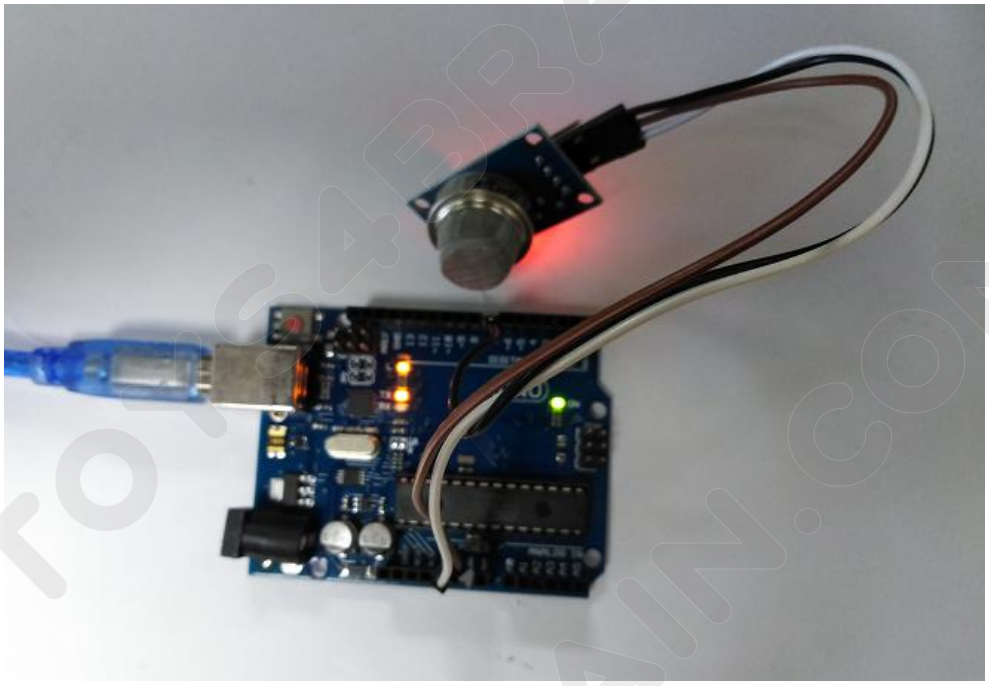
```
int sensor=7;
int gas_value;
void setup()
{
pinMode(sensor,INPUT);
Serial.begin(9600);
}

void loop()
{
gas_value=digitalRead(sensor);
Serial.println(gas_value);
}
```

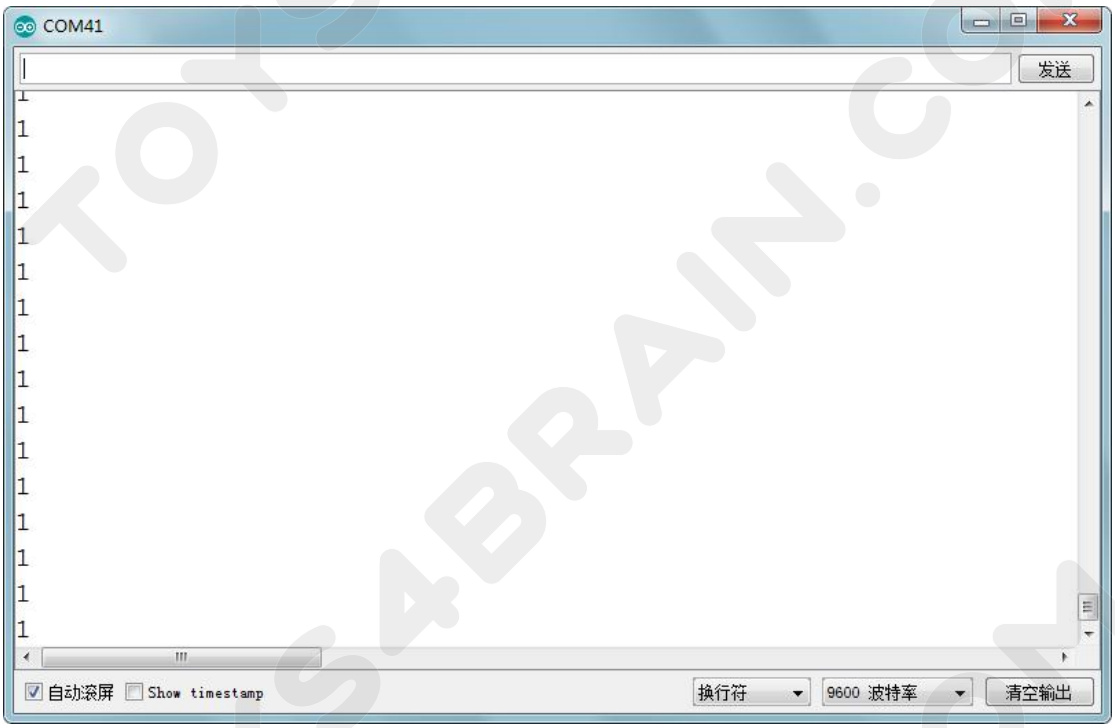
Note:

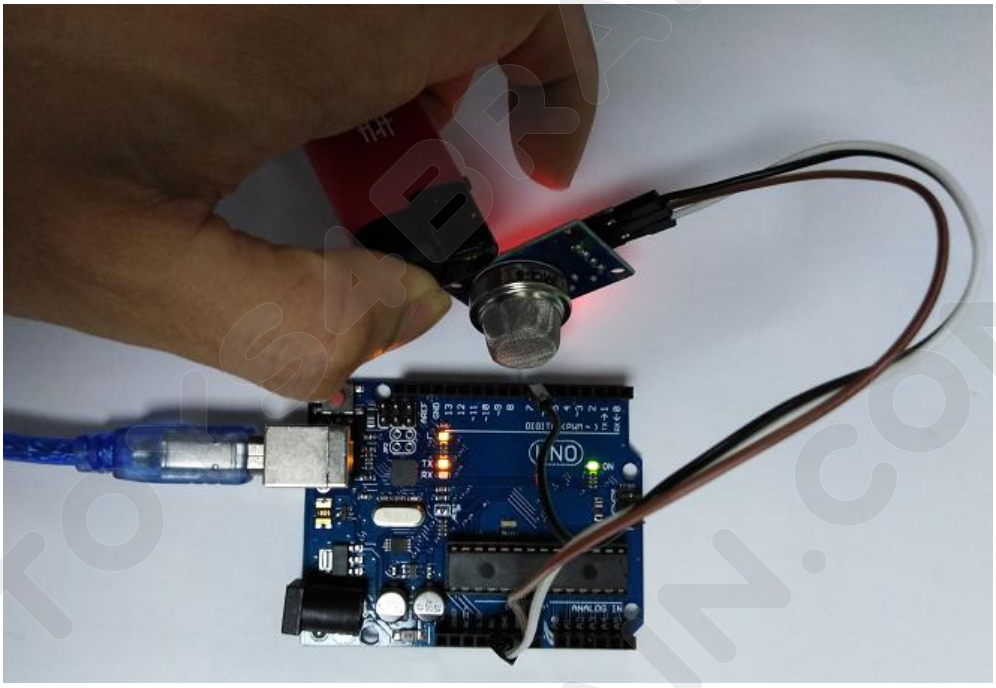
To apply a “gas leak” to the MQ5 sensor, simply use a cigarette or a cigarette lighter! Gently press the cigarette lighter's trigger switch (mild enough to allow gas leaks without triggering sparks) to allow continuous gas leakage and place the lighter near MQ5 sensor.

Picture of real product：

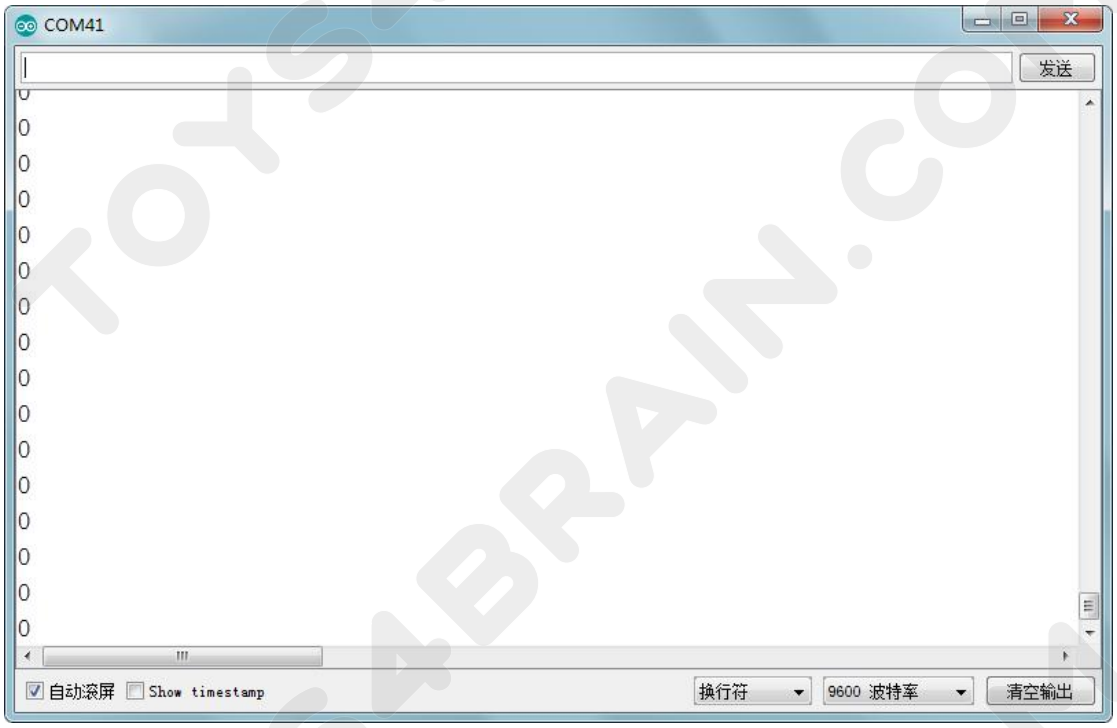


The screenshot below shows the ARDUINO's serial monitor reading before applying a gas leak and after applying gas leak. Before applying the gas leak, MQ5 only captures the atmospheric air concentration (our digital output pin is high and ARDUINO is measured at 1, as shown on the serial monitor).





When we apply "gas leak", the heating element inside MQ5 heats up and the output voltage changes (our D0 pin goes low and is measured as 0 by ARDUINO, as shown in the serial monitor output screenshot)



Section 5 MQ-7 Gas Sensor

Introduction

In this project, we will show what an mq-5 sensor is and how to use it on an ARDUINO board

Work Preparation

Hardware

1. UNO R3 x 1
2. MQ-7 sensor x 1
3. Jumper
4. USB cable x 1
5. PC x 1

About MQ-7 Gas Sensor

Description

MQ-7 combustible gas semiconductor sensor

The sensitive material of the MQ-7 gas sensor is SnO2, which has a low electrical conductivity in clean air. It is tested by circulating high and low temperature methods and detects CO at low temperatures (heating 1.5V). With the increase of gas concentration, the conductivity of the sensor is higher. When heated at a high temperature (5.0V), it removes other gases adsorbed at low temperatures. Please use a simple circuit to convert electrical conductivity changes to output signal corresponding to the gas concentration.

The MQ-7 gas sensor is highly sensitive to carbon monoxide. The sensor can be used to detect different gases containing CO with low cost and is suitable for different applications.

Feature:

Wide range of high sensitivity to combustible gases

High sensitivity to natural gas

fast reaction

Wide detection range

Stable performance, long life and low cost

Simple drive circuit

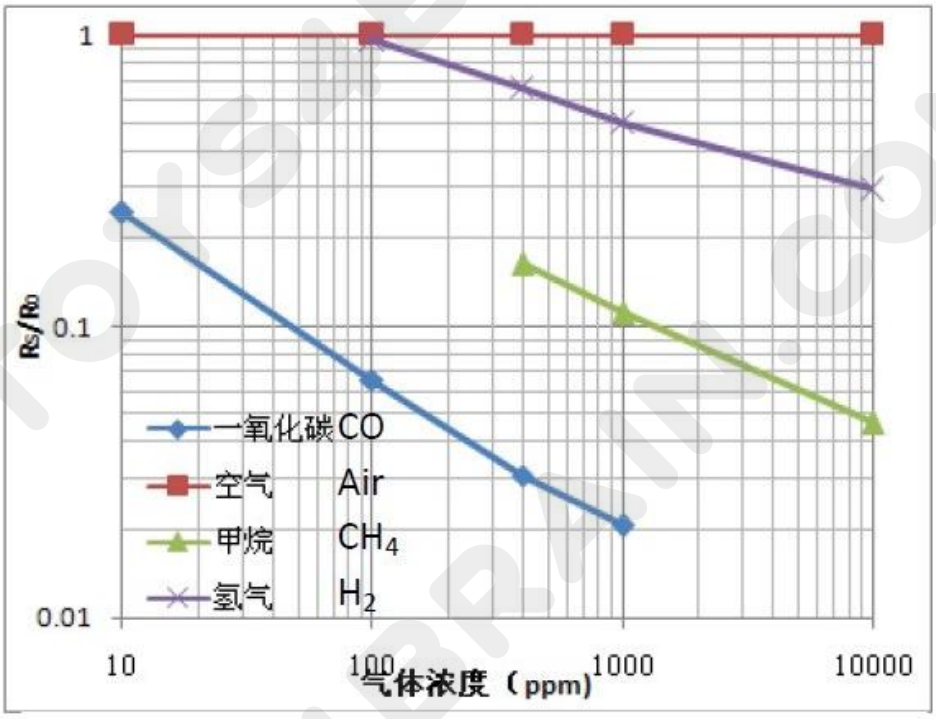
Technical parameters:

产品型号			MQ-7
产品类型			半导体气敏元件
标准封装			塑封
检测气体			一氧化碳
检测浓度			10-1000ppmCO
标准电路条件	回路电压	V_c	≤10V DC
	加热电压	V_H	5.0V±0.2V ACorDC（高） 1.5V±0.1V ACorDC（低）
	加热时间	T_L	60±1S（高）90±1S（低）
	负载电阻	R_L	可调
标准测试条件下元件特性	加热电阻	R_H	31Ω±3Ω（室温）
	加热功耗	P_H	≤350mW
	敏感体表面电阻	R_s	2KΩ-20KΩ（in100ppmCO）
	灵敏度	S	Rs(in air)/Rs(100ppmCO)≥5
	浓度斜率	α	≤0.6(R_{300ppm}/R_{100ppm} CO)
标准测试条件	温度、湿度		20℃±2℃；65%±5%RH
	标准测试电路		Vc:5.0V±0.1V； V_H（高）：5.0V±0.1V； V_H（低）：1.5V±0.1V
	预热时间		不少于48小时

敏感体功耗（Ps）值可用计算下式：$Ps=Vc^2\times Rs/(Rs+R_L)^2$

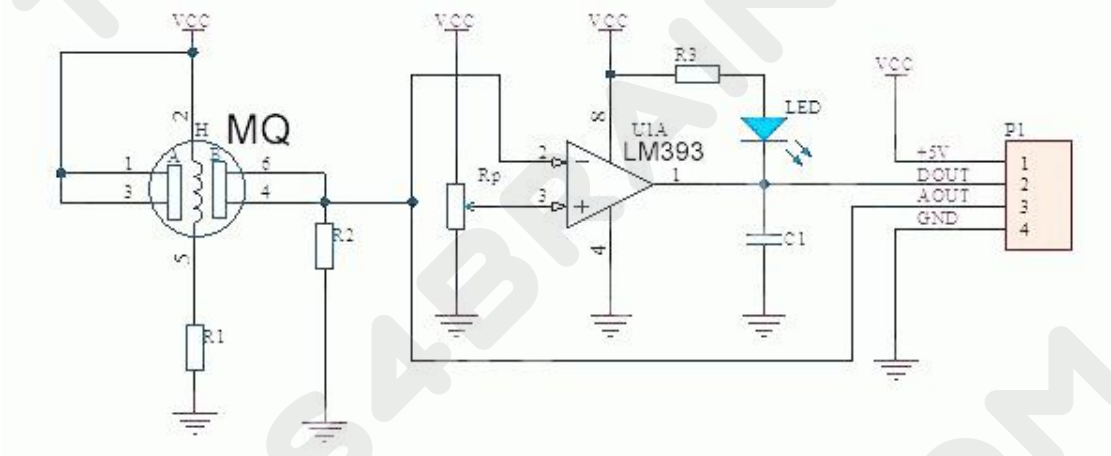
MQ-7 carbon monoxide sensor Introduction

According to its data sheet, the MQ-7 carbon monoxide sensor detects 20 to 2000 ppm of CO in the air. This is its sensitivity characteristic curve:



This is a diagram of Rs / R0 versus gas concentration (ppm). Rs is the resistance of the sensor in the target gas, and R0 is the resistance in the clean air. We will use this diagram later when we create code.

This wire splitter is more convenient because it converts resistance changes into voltage changes. Here's a schematic of it:



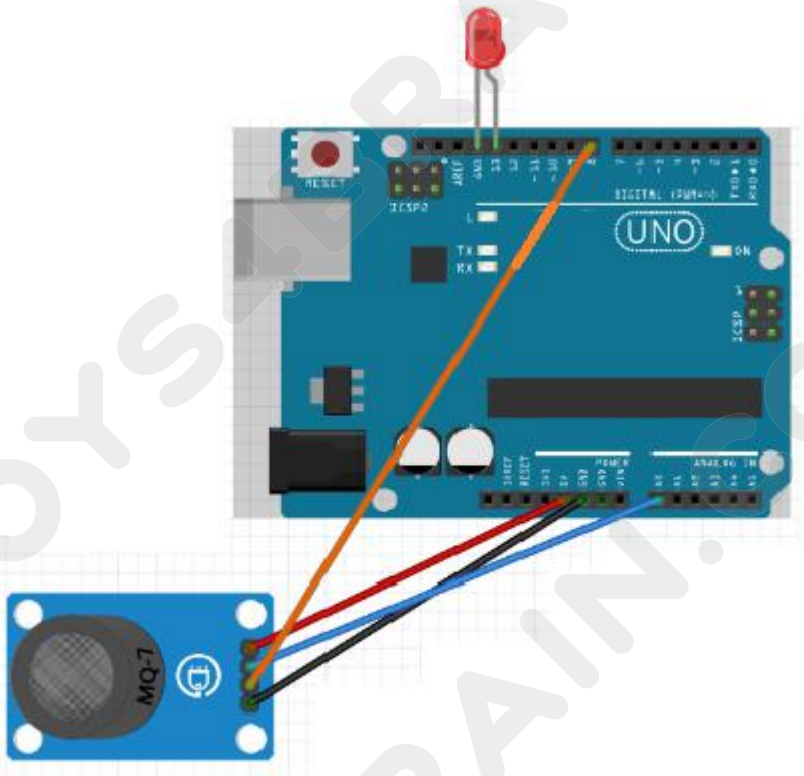
There are two ways to read the output of MQ-7. One is through the DOUT pin, which gives a high level when the concentration threshold is reached, otherwise it is low. The threshold can be changed by adjusting the trimmer on distribution board, which is Rp in the schematic diagram.

At the same time, the AOUT pin provides a varying voltage representative of the CO concentration. If we look at the above characteristic curve, we can convert the voltage reading to ppm. The relationship between ppm concentration and RS / R0 is:

$$ppm = 100 * / fraclog(40)log(0.09)^{/fracRSR0}$$

Example:

Connect the MQ 7 Sensor and ARDUINO Board as shown below:



Code：

```
const int AOUTpin=0;    //The AOUT pin of CO sensor enters ARDUINO's analog pin A0
const int DOUTpin=8;    //The DOUT pin of CO sensor enters ARDUINO's digital pin A8
const int ledPin=13;    //The LED's anode is connected to ARDUINO's digital pin D13

int limit;
int value;

void setup() {
Serial.begin(115200);    //Set baud rate
pinMode(DOUTpin, INPUT);    //Set the pin as the input for ARDUINO
pinMode(ledPin, OUTPUT);    //Set the pin as the onput for ARDUINO
}

void loop()
{
value= analogRead(AOUTpin);    //Read the simulation value from the CO sensor OUT pin
limit= digitalRead(DOUTpin);    //Read the numeric value from DOUT pin of CO sensor
Serial.print("CO value: ");
Serial.println(value);    //Print CO value
Serial.print("Limit: ");
Serial.print(limit);    //Print to LOW or HIGH (above or below) limit
delay(100);
if (limit == HIGH){
digitalWrite(ledPin, HIGH);    //If the limit is reached, LED is turned on as a status indicator
}
else{
digitalWrite(ledPin, LOW);    //If the threshold is not reached, the LED remains off}
}
```

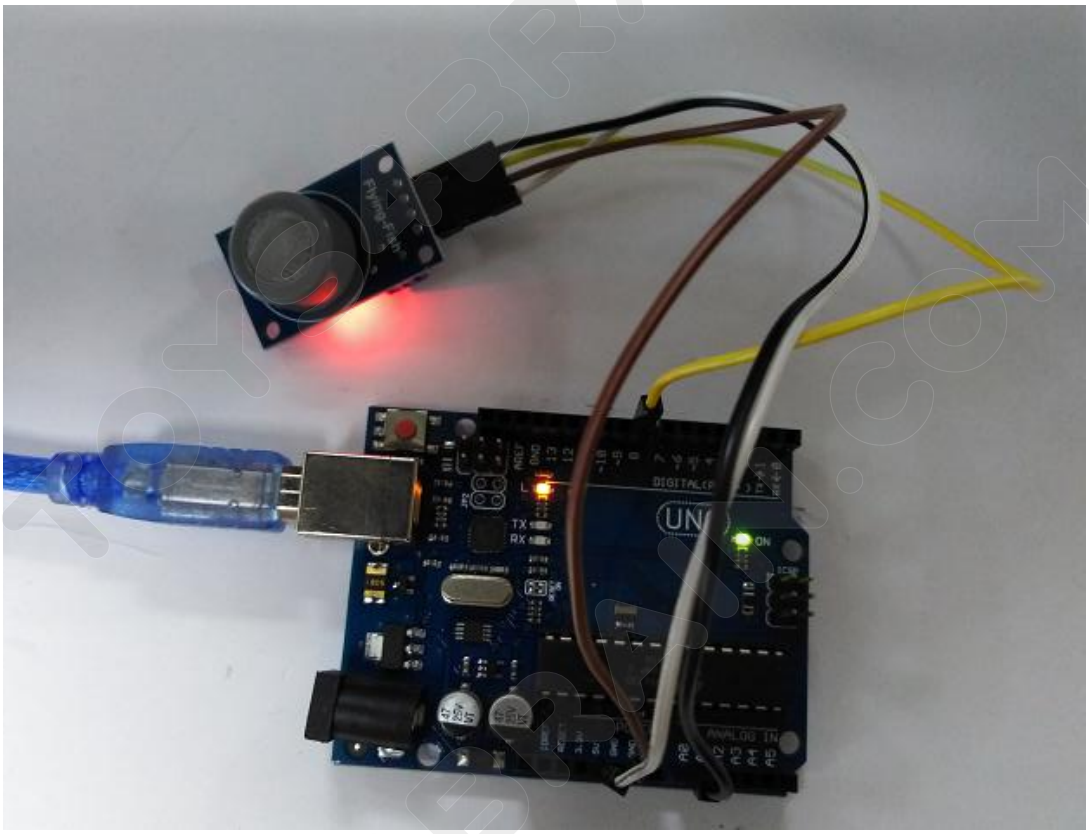
The first code block defines all pin connections for the sensor and LED. Since AOUTpin is connected to analog pin A0, it is initialized to 0. Since the DOUT pin is connected to digital pin D8, it is initialized to 8. Since the LED is connected to digital pin D13, it is initialized to 13. Two variables, limits and values, are also declared. These will be used to store the values of analog pin AOUT and digital pin DOUT.

The next code block sets the baud rate and declares DOUTpin as an input and ledPin as an output. This is because the sensor is an input to the ARDUINO to read and process the sensor values. If the sensor detects alcohol, the LED output will act as an indicator.

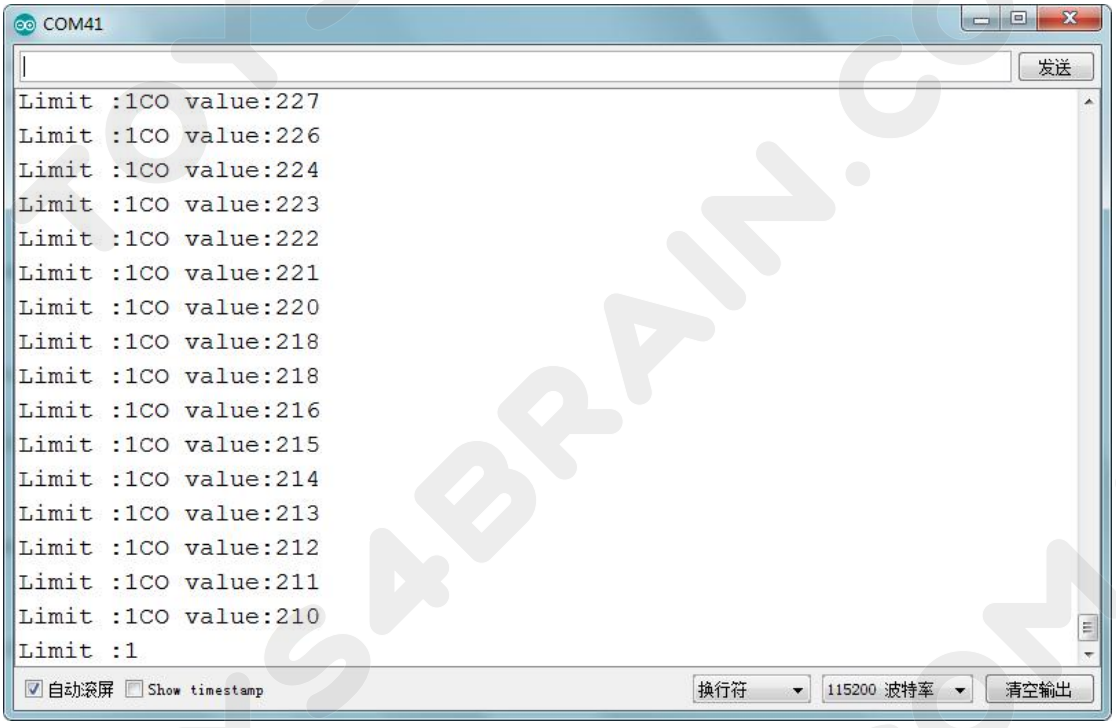
The next code block reads sensor pin AOUT and stores the value in an integer value. It also reads the sensor pin DOUT and stores the value in an integer limit. Then we print the alcohol value, which will be a value from 0 (no alcohol detected) to 1023 (the maximum carbon monoxide level that can be read). We will print out the limit of HIGH or LOW. If detected CO is below threshold level, the returned limit will be low. If detected CO is above threshold, the returned limit will be HIGH.

If the value is HIGH, the LED will illuminate. If the value is lower, the LED will remain off.

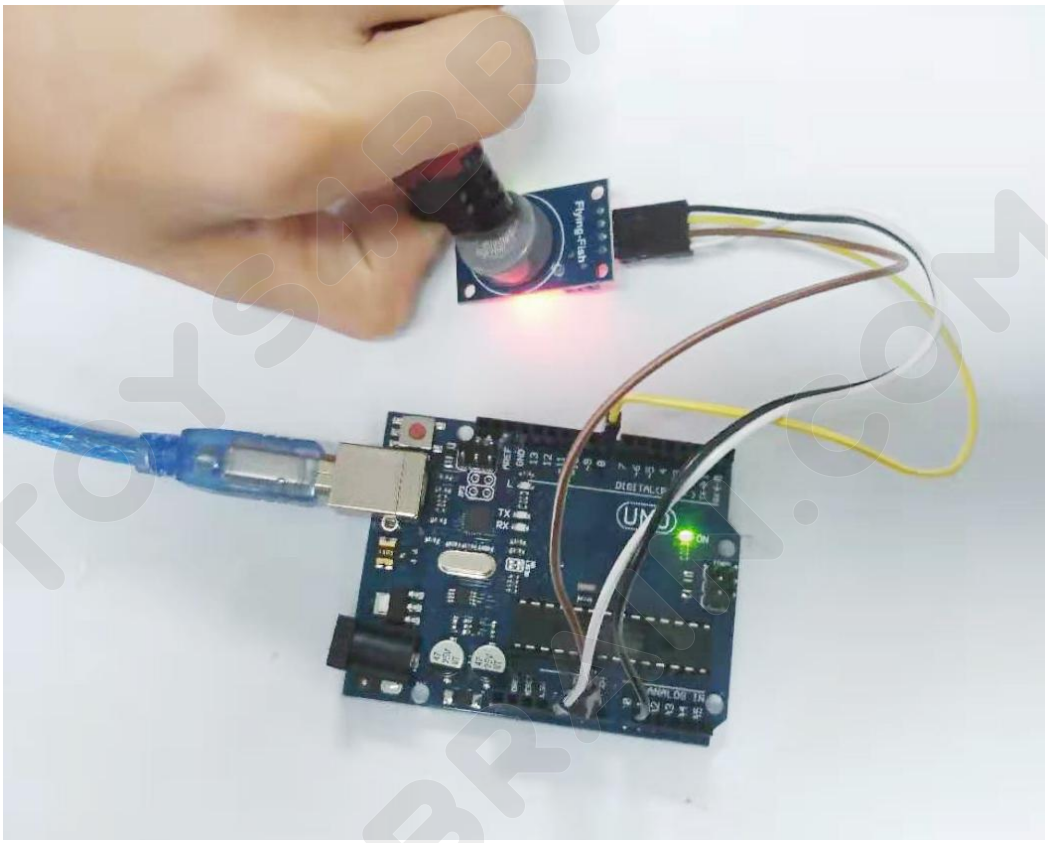
Picture of real product:



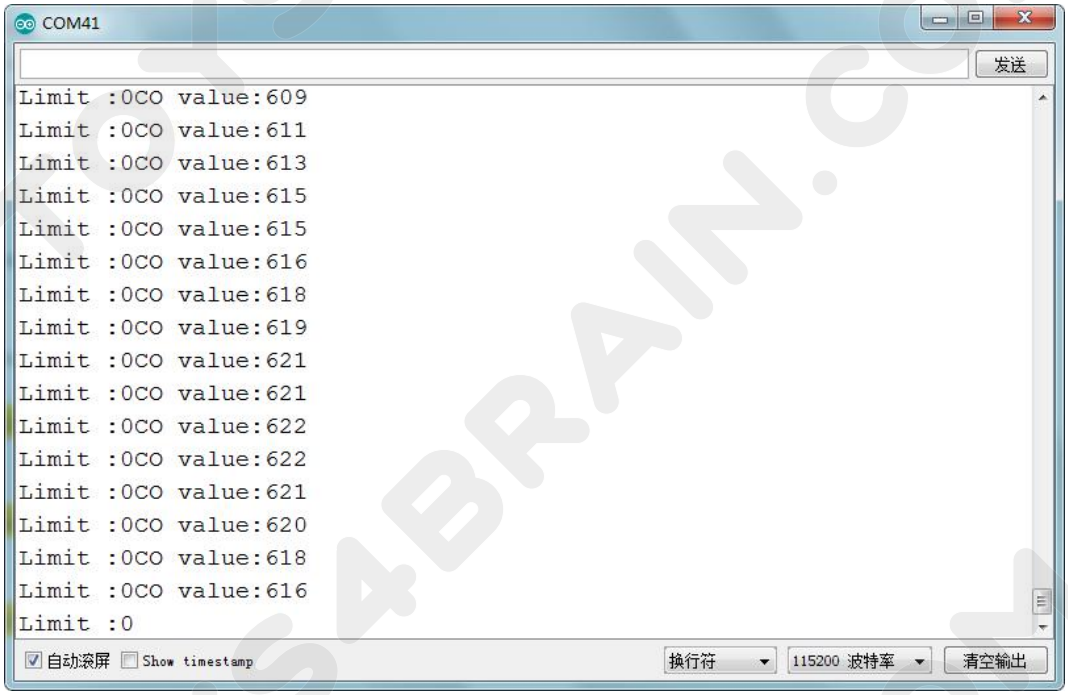
The serial monitor shows:



When carbon monoxide is detected, the LED off



The serial monitor shows:



Section 6 Barometric Pressure Sensor BMP 180

Introduction

In this tutorial, we will use BMP180 and ARDUINO to design a barometric measurement system. First, in order to interface BMP180 with ARDUINO, we need to download a library designed specifically for BMP180. The library is located at:

https : //github.com/adafruit/Adafruit-BMP085-Library After attaching the library, we can invoke special functions that will simplify working with the use of the BMP180 sensor.

Work preparation

Hardware

1. UNO R3 x 1
2. air pressure sensor BMP 180 x 1
3. F / M jumper
4. USB cable x 1
5. PC x 1

About the I2C LCD 1602 display

The pressure sensor is a BMP-180 based digital air pressure sensor module that is compatible with the old BMP-085 digital pressure sensor, with lower power consumption, smaller size and higher accuracy. The BMP180 combines air pressure, temperature and altitude. I2C allows easy connection to any microcontroller. Onboard 3.3v LDO regulator makes the plate fully compatible with 5V power supply. The BMP-180 can measure pressure range from 300 to 1100 hPa (+ 9000 m to -500 m related to sea level) in pre-resolution mode with accuracy down to 0.02 hPa (0.17 m). The BMP-180 is an improved replacement for the BMP-085 sensor. The BMP-180 uses piezoresistive technology to achieve high precision, linearity, EMC robustness and stability over a longer period of time.

Feature:

Using a digital barometer such as this to measure the absolute pressure of environment has some interesting applications. By converting the measured pressure to altitude, you will get a reliable sensor to determine the height of the robot, aircraft or projectile!

With a sensor as powerful as the BMP180, you can achieve 1m accuracy, and the noise in ultra-high resolution noise is only 17cm.

The device will operate at a current of only 0.3uA, which means that battery-powered applications require low current consumption.

The BMP180 is fully calibrated and ready for immediate use.

When the device is running through I2C, we add an optional I2C pull-up circuit that can be enabled through the PU (pull-up) jumper on board for easy operation on the breadboard.

The device USES I2C to provide a 16-bit value of pressure and temperature, which is calculated in conjunction with the calibration data in the device to provide a temperature compensation height.

Product specifications:

1.8V / 5V supply voltage

Low power consumption – 0.5uA at 1Hz

I2C interface

Maximum I2C speed: 3.5MHz

Very low noise – up to 0.02hPa (17cm)

Full calibration

Pressure range: 300hPa to 1100hPa (9000m to -500m)

Weight: 1.18 grams

Size: 21mm x 18mm

How does BMP180 work?

The BMP180 consists of a piezoresistive sensor, an analog-to-digital converter and a control unit with an E2PROM and serial I2C interface. The BMP180 provides uncompensated pressure and temperature values. The microcontroller sends an initiation sequence to start a pressure or temperature measurement. After the conversion time, the result values (pressure or temperature, respectively) can be read through the I2C interface. To calculate the temperature in °C and calculate the pressure in hPa, calibration data must be used. These constants can be read from the BMP180 E2PROM via the I2C interface during software initialization. The sampling rate can be increased to 128 samples per second (standard mode) for dynamic measurements. In this case, it is sufficient to measure the temperature once per second and use this value for all pressure measurements during the same time period.

Application

Temperature monitoring

Pressure monitoring

Highly monitored

3D navigation in complex indoor spaces (used with accelerometer)

Connect the wires to the board by using any methods you like to establish a connection to the board. In this example, we will solder a five-pin male-to-male row and connect the BMP180 to ARDUINO using a male/female jumper.

Solder the 5-pin long male-female connector to the board. You can solder it to either side; the bottom is more useful for breadboard, and the top is more useful for jumpers.

Please note that the BMP180 is sensitive to moisture. After soldering, do not remove the flux by flushing the board with water or other liquids.

Connecting it to the ARDUINO In the following sections, we will see how to connect the breakout board to the UNO R3. There are 5 pins on the branch board as described below. The plate has a 3.3V regulator. Therefore, you can use either 5V or 3.3V as the power supply voltage.

Measuring weather and altitude

BMP180 is designed to accurately measure atmospheric pressure. Atmospheric pressure varies with weather and altitude; you can use this sensor to measure both. How is this done:

What is atmospheric pressure?

Pressure is defined as the force that "presses" the area. Common pressure unit is pounds per square inch (psi). One pound, one square inch, equals one pound per square inch. The SI unit is Newton per square meter and is called Pascal (Pa).

Pressure (gravity, tension, etc.) can be measured in many situations, but now we are interested in atmospheric pressure, which is the force of air around you acting on all objects. The weight of gases in the atmosphere produces

atmospheric pressure. Normally people won't notice that air weighs anything, but if you pick up an inch of air from the sea level to the top of the atmosphere, the air weighs about 14.7 pounds. (a centimeter-wide column of air weighs about a kilogram.) This weight presses on the footprint of the column, producing an atmospheric pressure that we can measure using sensors such as BMP180.

Since the inch-wide air column weighs about 14.7 pounds and the pressure is 1 square inch, the resulting average sea level pressure is about 14.7 pounds per square inch (psi) or 101,325 pascals. This will drop by about 4% for every 1000 feet (or 300 meters). The higher you get, the less pressure you see, because the column to the top of the atmosphere is much shorter and therefore lighter. It's useful to know this because by measuring the pressure and doing some math, you can determine your height.

Interesting fact: The pressure at 12,500 feet (3810 meters) is only half of the sea level pressure. In other words, half of the air mass is below 12,500 feet, and the air density at 12,500 feet is half of the sea level. No wonder you have difficulty breathing there.

BMP180 outputs absolute pressure in Pascal (Pa). The pressure of a Pascal is small, about equal to the pressure of a piece of paper on a table. You will often see measurements in hPa (1 hPa = 100 Pa) or kPa (1 kPa = 1000 Pa). The ARDUINO library we provide outputs floating point values in hPa, which also happen to be equal to 1 millibar (mbar).

Here are some conversions to other pressure units:

1 hPa = 100 Pa = 1 mbar = 0.001 bar

1 hPa = 0.75006168 torr

1 hPa = 0.01450377 psi (psi)

1 hPa = 0.02953337 inHg

1 pa = 0.00098692 atmosphere (standard atmosphere)

Temperature influence

Since temperature affects the density of the gas, density affects the quality of the gas, and the mass affects pressure (rotation), so atmospheric pressure changes drastically with temperature. The pilot called it "density height", which is easier to take off in cold weather than hot weather because air is denser and has more aerodynamic effects.

To compensate for the temperature, the BMP180 includes a fairly good temperature sensor and a pressure sensor. To perform a pressure reading, you first obtain a temperature reading and then combine it with the original pressure reading to arrive at the final temperature compensated pressure measurement. (Don't worry, the ARDUINO library makes all of this very easy.)

Measuring absolute pressure

As we just mentioned, if your application needs to measure absolute pressure,

all you have to do is take a temperature reading and then perform the pressure reading (see the sample sketch for details). The final pressure reading will be in hPa = mbar. You can use the above conversion factor to convert it to other units if needed.

Please note that the absolute pressure of the atmosphere will vary with your altitude and current weather patterns, both of which are useful measurements.

Weather observation

The pressure at any given location on the earth (or anywhere in the atmosphere) is not constant. The complex interaction between the spin of the Earth, the tilt of shaft and many other factors can lead to higher and lower pressure areas of motion, which in turn lead to weather changes we see every day. By observing changes in pressure, you can predict short-term changes in the weather. For example, pressure reducing usually means wet weather or an imminent storm (low pressure systems are entering). An increase in pressure usually means that the weather is fine (the high pressure system is passing).

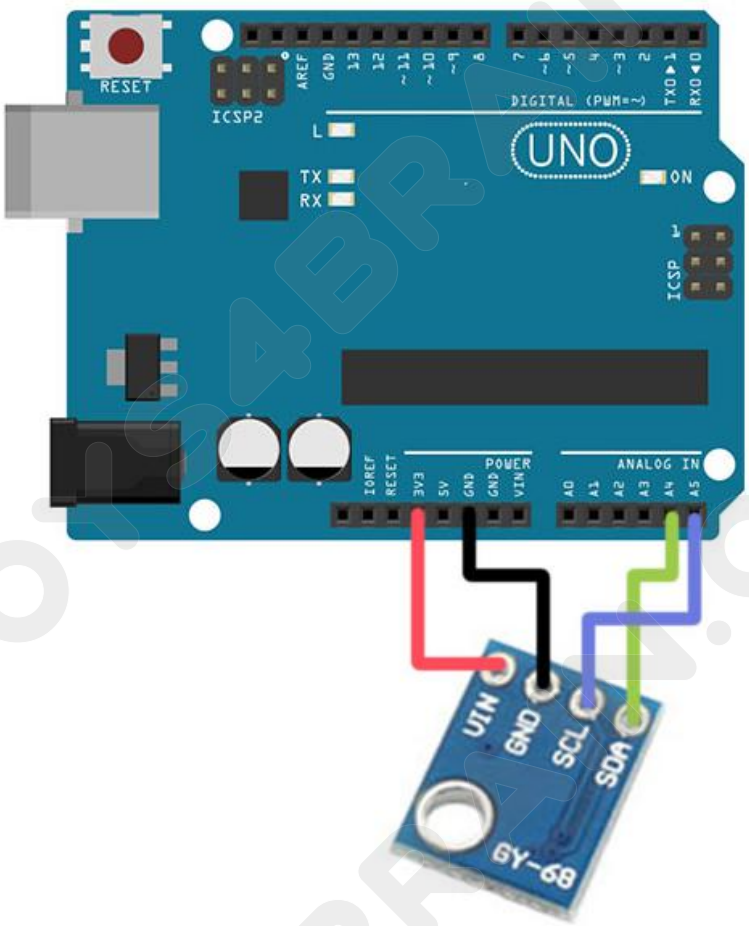
But keep in mind that atmospheric pressure also varies with altitude. The absolute pressure of Denver (5280' above sea level) will always be lower r than that in San Francisco (52' above sea level). If the weather station only reports its absolute pressure, it will be difficult to directly compare pressure measurements from one location to another (large-scale weather forecasts depend on as many weather station measurements as possible).

To solve this problem, weather station always eliminates the effects of altitude from reported pressure readings by mathematically adding an equivalent fixed pressure to make it appear as if it were being acquired at sea level. When you do this, San Francisco's readings are always higher than Denver, because of

weather conditions, not because of altitude.

To do this, there is a function called seaLevel(P,A) in the library. This requires absolute pressure (P) in hPa, the current height of the station in meters (A), and the effect of height can be removed from the pressure. You can use the output of this feature to directly compare your weather readings to other weather stations around the world.

Example:



Pin description

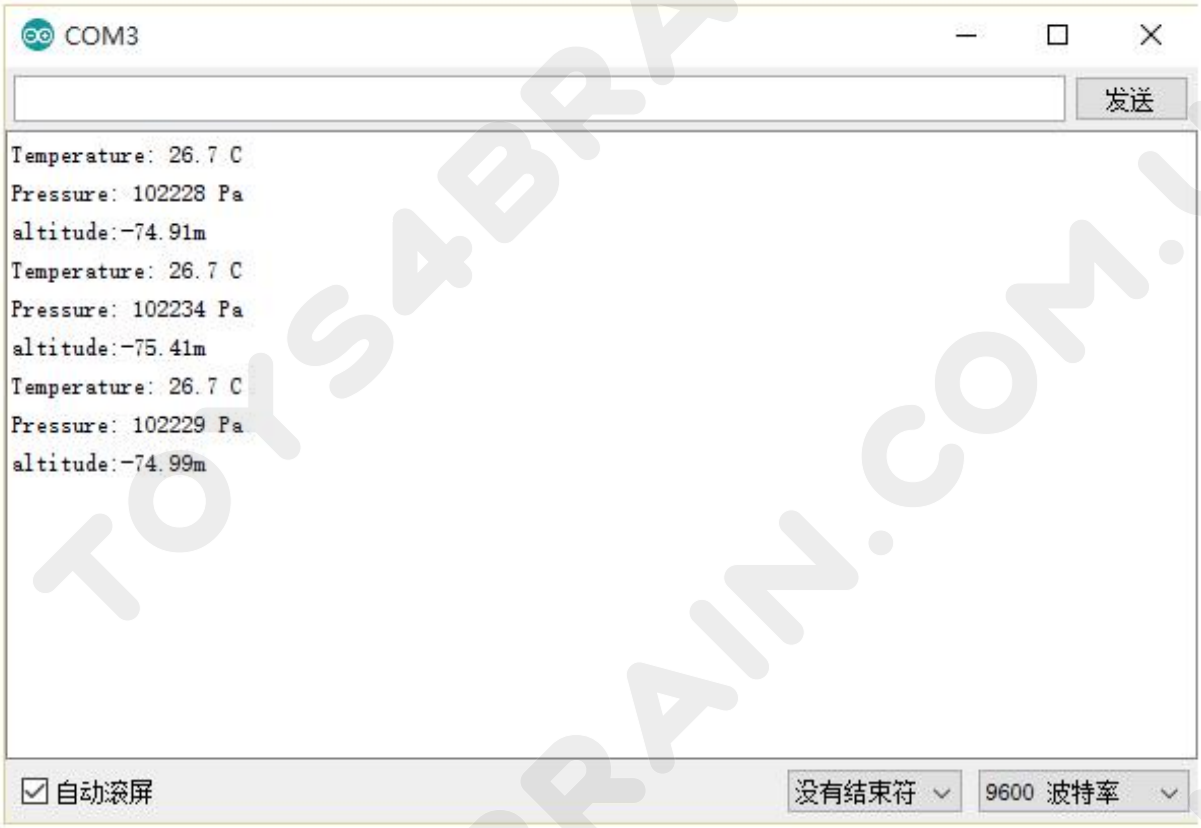
VIN power input 3.3v

GND ground

SCL clock line

SAD data line

Experimental results:



Section 7 Flame sensor module

Introduction

In this lesson we will show you what a flame sensor is and how it works. You can learn how to use a flame sensor on the UNO R3 board.

Flame sensor is an electronic device capable of sensing/detecting fire or high temperature areas. When it senses a fire, it signals through an LED on the top. These types of sensors are typically used for short distances. They were able to detect fires as high as three feet. Because of its good results and cost-effectiveness, flame sensor has become the most common equipment in market today.

There are two types of flame sensors on the market, one with three pins and the other with four pins. Both sensors can be easily connected to any microcontroller. In this tutorial we will use a four-pin flame sensor. You will see the complete wiring diagram of the flame sensor and ARDUINO interface, as well as the complete ARDUINO source code and its description.

Work preparation

Hardware

UNO R3 board x 1

Flame sensor x 1

F / M jumper

USB cable x 1

PC x 1

About the flame sensor

The flame sensor can detect flames at wavelengths ranging from 760 to 1100 nanometre. A small fire like a light fire can be detected at about 0.8m. With a detection Angle of about 60 degrees, the sensor is particularly sensitive to flame spectrum. The module includes an infrared sensor, potentiometer, operational amplifier circuit and an led indicator. The on-board LM393 operational amplifier is used as a comparator to adjust the sensitivity level. The sensor has digital and analog outputs, and the sensitivity can be adjusted by a blue potentiometer.

Feature

The operating voltage is 3.3 – 5V.

It provides us with analog and digital outputs.

It has an indicator light to indicate if a flame has been detected.

The threshold can be changed by rotating the top of the potentiometer.

The flame detection distance can trigger a lighter flame test within 0.8m. If the flame intensity is high, the detection distance will increase.

The detection angle of the flame sensor module is approximately 60 degrees.

It has both analog and digital outputs. The analog output gives us a real-time voltage output signal on the thermal resistance, while the digital output allows us to set the threshold through a potentiometer. In our tutorial, we will use both of these outputs and see how the sensor works. In many projects, flame sensors can be used to alert when a fire is detected, for safety purposes and more.

Working principle

Flame sensor is very sensitive to flames and other light.

Its analog output provides a real-time output voltage across the thermal resistance.

When the temperature reaches a certain threshold, the output of high and low

signal threshold can be adjusted by potentiometer, whose task is digital output.

Flame sensor Pin as follows:

A0: This is an analog pin that will be connected to the analog pin of ARDUINO.

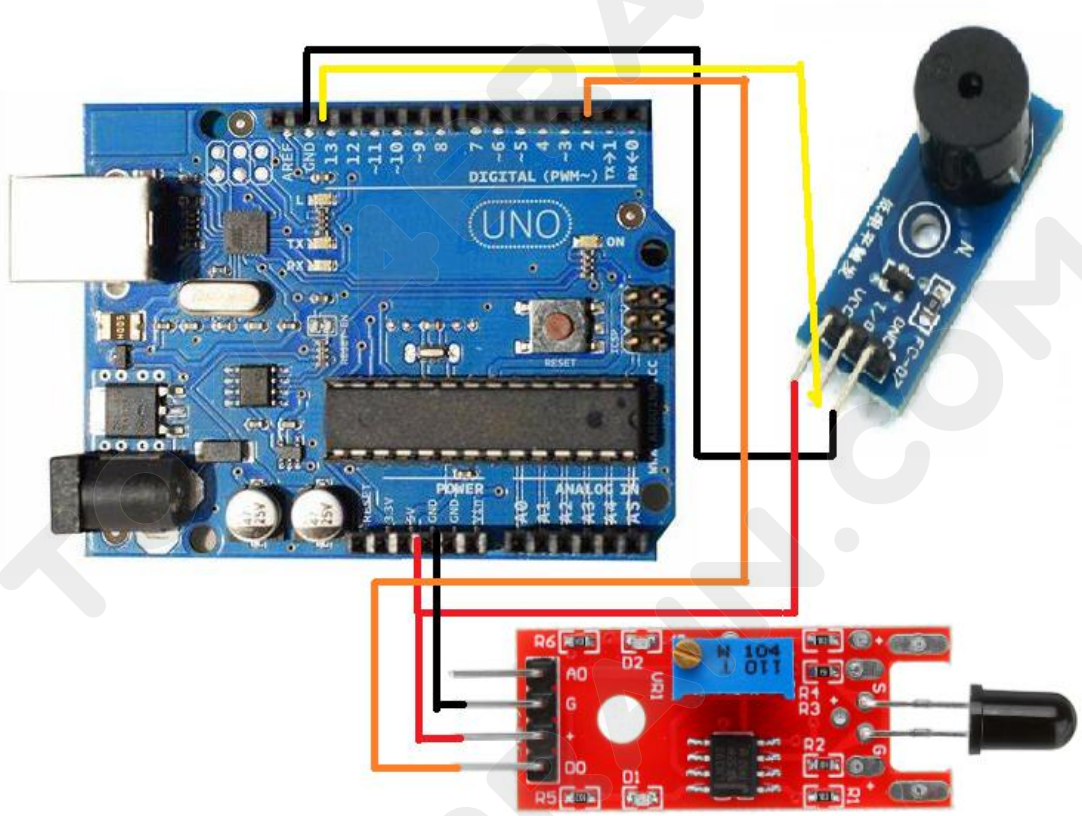
G / GND: This is the ground pin that will connect to the ground of ARDUINO.

+ / VCC: This is the sensor's input voltage pin, which will connect to the +5V of ARDUINO.

D0: This is a digital pin that will be connected to the digital pin of ARDUINO.

Example:

Flame sensor digital (D0) output



Sample code:

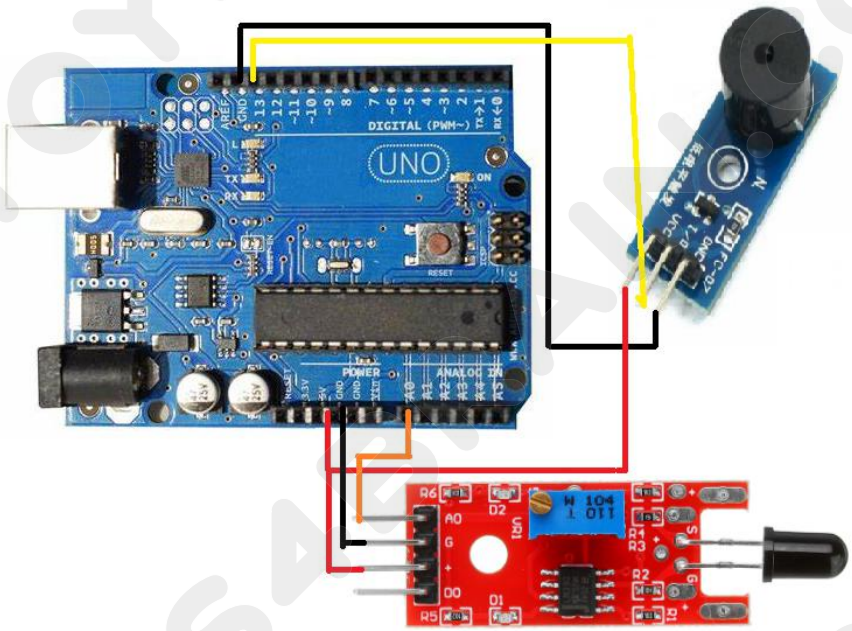
```
Int Buzzer = 13; // Use the buzzer to alert
Int FlamePin = 2; // This is for the input pin
Int Flame = HIGH; //High when the flame is exposed

void setup() {
   pinMode(Buzzer, OUTPUT);
   pinMode(FlamePin, INPUT);
   Serial.begin(9600);
}

void loop() {
   Flame = digitalRead(FlamePin);
   if (Flame== HIGH)
   { Serial.println("HIGH FLAME");
      digitalWrite(Buzzer, HIGH);
   }
   else
   { Serial.println("No flame");
      digitalWrite(Buzzer, LOW);
   }
}
```

Example:

Flame sensor digital (A0) output



Example code:

```
Const int analogPin = A0; // Flame sensor (A0) to ARDUINO analog input pin A0
Const int BuzzerPin = 13; // buzzer output pin
Const int threshold = 400; // flame level threshold (you can change the value as needed)
Void setup() {
   pinMode(BuzzerPin, OUTPUT);
   / / Initialize serial communication: Serial.begin (9600);
}
Void loop() {
   / / Read the value of flame sensor:
   Int analogValue = analogRead(analogPin);
Serial.println(analogValue); // Serial print FLAME sensor value
   If (analogValue > threshold) {
digitalWrite(BuzzerPin, HIGH);
  Serial.print("High FLAME");
   }
   Else if (analogValue = threshold)
{ Serial.print("Low FLAME");
     digitalWrite(BuzzerPin, HIGH);
     Delay(400);
     digitalWrite(BuzzerPin, LOW);
   }
   Else {
digitalWrite(BuzzerPin, LOW);
Serial.print("No flame");
   }
   Delay(1);
}
```

Section 8 Water Level Sensor Module

Introduction

Water sensor bricks are designed for water detection and can be widely used to detect rainfall, water levels and even liquid leaks.

In this course, we will learn what a water sensor is and how it can be used with the Uno R3 board. It comes to detect water.

Ready to work

Hardware

UNO board x 1

Water sensor x 1

Red LED x 1

Breadboard x 1

Jumper

USB cable x 1

PC x 1

About water sensor

Water sensor is an easy to use, small size, light weight, high water cost, droplet identification and detection sensor. Compared with domestic and foreign products, it is not only small in size, powerful in function, but also ingenious in design, with the following characteristics:

The water level is determined by a series of exposed parallel lines to measure the size of the droplet/water.

The water quantity can be easily changed into the analog quantity signal, and the output analog quantity can be directly used for the program function, so as

to realize the water level alarm function.

Low power consumption and high sensitivity are the most important features of this module.

Compatible with SainSmart UNO SainSmart mega2560 SainSmart ADK.

Working principle

The sensor works by connecting a series of exposed traces to the ground and interleaving them between the ground trace and the sensor trace. Sensor walk line with 1 m Ω weak pull-up resistor. The resistance will pull up the sensor wiring value until the water drops short-circuit the sensor wiring and grounding wiring. This circuit will work with your ARDUINO digital I/O pins, or you can use it with analog pins to detect water-induced contact between grounding and sensor wiring.

Used as the water level in the tank

To use it as a water level detector in sediments, sensor must be installed inside the tank where the water level is to be controlled. The position of the sensor must ensure that the parallel lines are perpendicular to the sensor level. When the sensor is immersed in water, the pin S will give us a larger value.

Used as a rain detector

To use this sensor to detect whether it is raining, it must be placed horizontally so that the raindrop falls on the sensor, because the raindrop forms on the water film on the sensor surface by increasing the value of pin S, thus it can be inferred whether it is raining.

Product specifications:

Working voltage: 5V

Interface: analog

Detection width: 40mm × 16mm

Working temperature: 10 ° C - 30 ° C

ARDUINO compatible interface

Low power consumption

High sensitivity

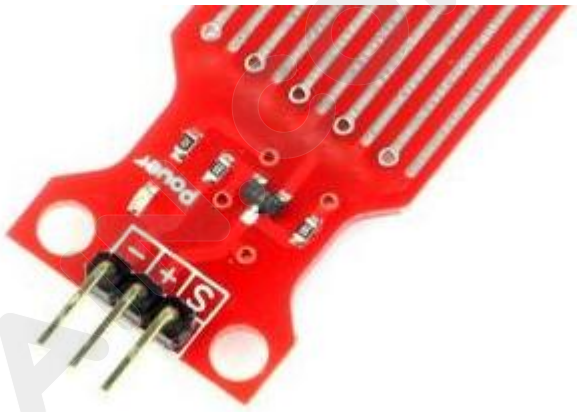
Output voltage signal: 0-4.2V

Pin Definition:

“S” stands for signal input

“+” stands for power supply

"-" stands for GND



Two of the pins are used for power supply, one is connected to +5V of ARDUINO, and the other pin is connected to ground terminal of the ARDUINO. The Pin S is the signal pin. The pin outputs an analog voltage signal proportional to the number of sensors covered by the liquid. The pin is connected to a simulation pin on the ARDUINO board to be read.

Application:

Rainfall detection

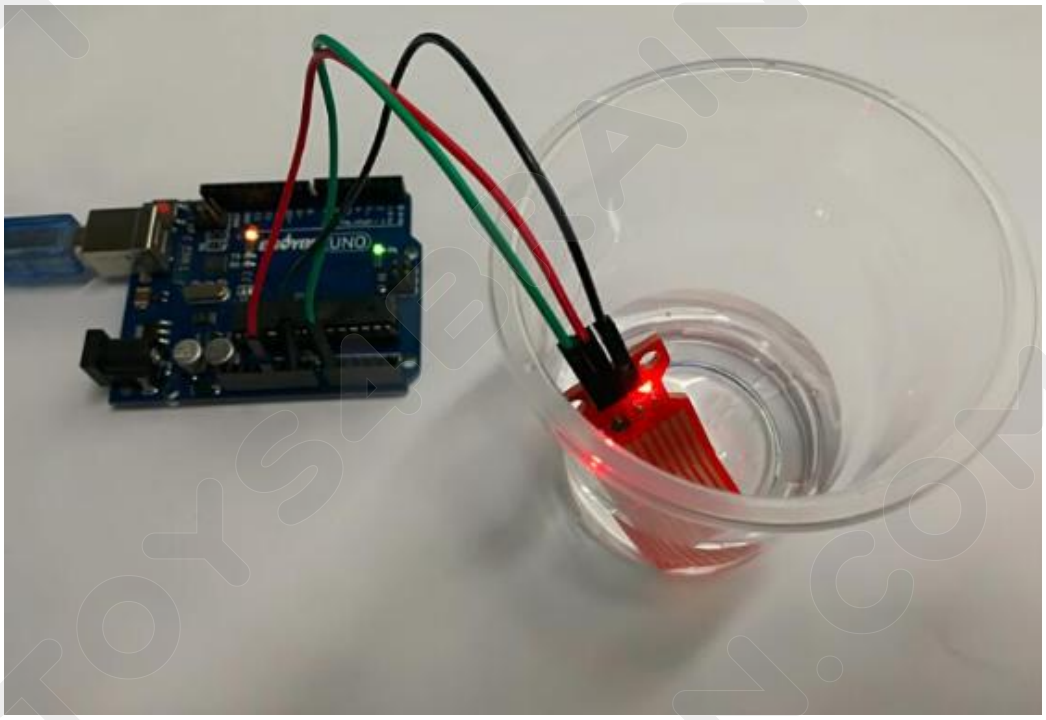
Leakage

Tank overflow detector

Example:

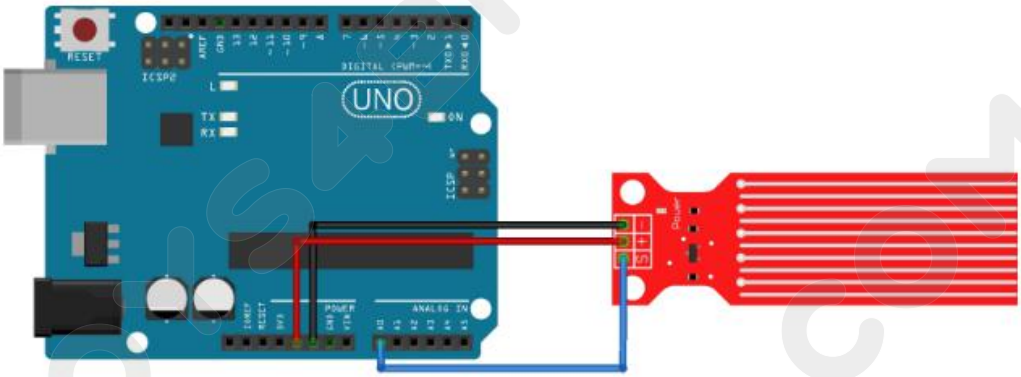
ARDUINO liquid level detector

In this example, we will show how to use a water sensor to detect the amount of water in a tank. We will use the S pin as the analog input connected with ARDUINO, and the value read will be higher depending on whether the sensor surface is covered by water. This is because considering that the water we use in the sediments is not pure water (H2O), water therefore ACTS as a conductor, because if water is not conductive. But we rarely use this type of sensor to measure the water level of purified water in the tank.



Connection Diagram

Construct the circuit as follows:



Here, we connect the signal pin (S) to analog pin A0. This allows the ARDUINO board to read analog voltage values.

Code program

Once this is done, connect the ARDUINO board to your computer with a USB cable. The green power LED (marked PWR) should illuminate. Open the ARDUINO IDE and select the appropriate board type and port type for your project. Then load the following sketch onto your ARDUINO.

```
const int analogInPin = A0;

int sensorValue = 0;

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

```
void loop()
 {
sensorValue = analogRead(analogInPin);
Serial.print("Sensor = " );
Serial.print(sensorValue*100/1024);
Serial.println("%"); delay(1000);
}
```

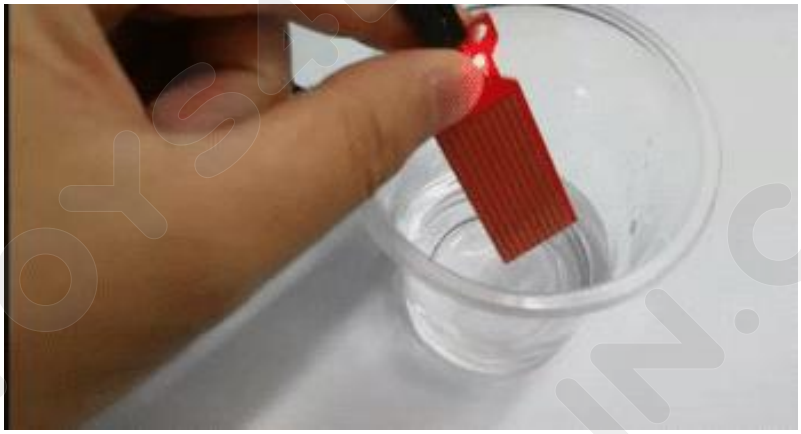
In the first paragraph of this code, we declare the AnalogInPin variable and initialize it to 0. Then declare a sensor Value variable that will hold the value of the analog value output from the sensor as our sensor reading, representing the liquid level.

In the setup () function, we set the baud rate and enter sensor Pin because it is the input value of the ARDUINO board to be read.

In our loop () function (which is code over and over again), we read the value from sensorPin and store it in the variable sensor Value. We then output this value to the serial monitor for reading. The code keeps running this code over and over again, so it keeps gaining new understanding. We give a 100ms delay between each read to create a small pause between the two.

Operation result:

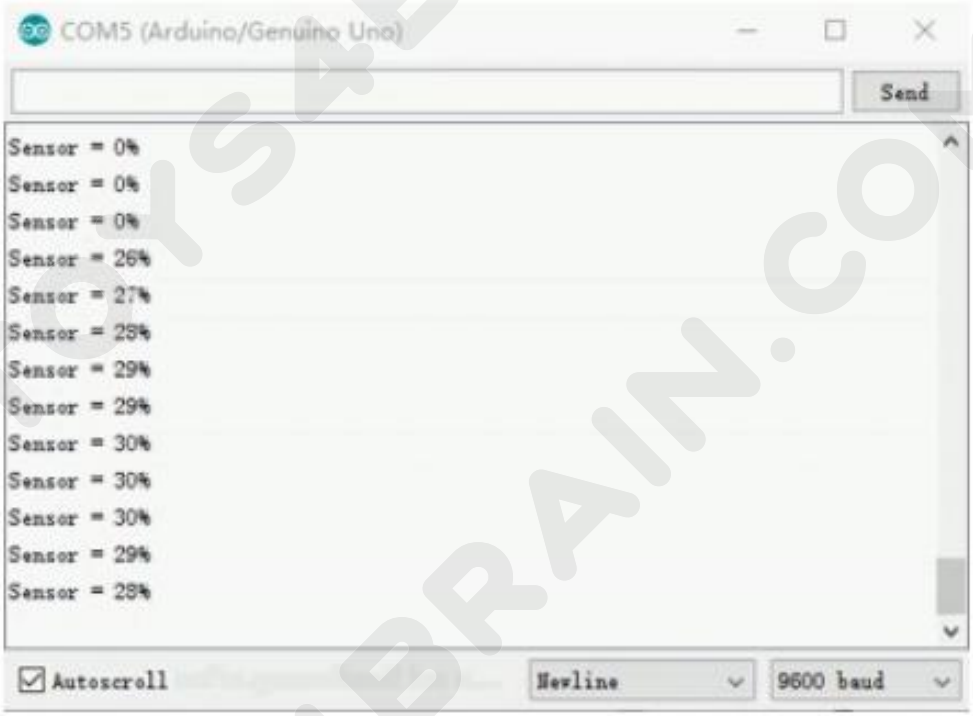
After the upload is completed for a few seconds, the sensor is ready to immerse in the water.



The serial monitor shows:

Example of sensor immersion in water:

The serial monitor shows:



Section 9 Digital Touch Sensor Module

Introduction

We need to use switches to control electronic devices or appliances. Sometimes, when we use electronic switches with wet hands and then touch electronic or electrical loads, the electrical switches generate vibrations compared to ordinary switches, which may be required for some projects.

In this course, we will show you what a digital touch sensor module is and how to use it on an ARDUINO board.

Hardware

UNO R3 board x 1

Breadboard x 1

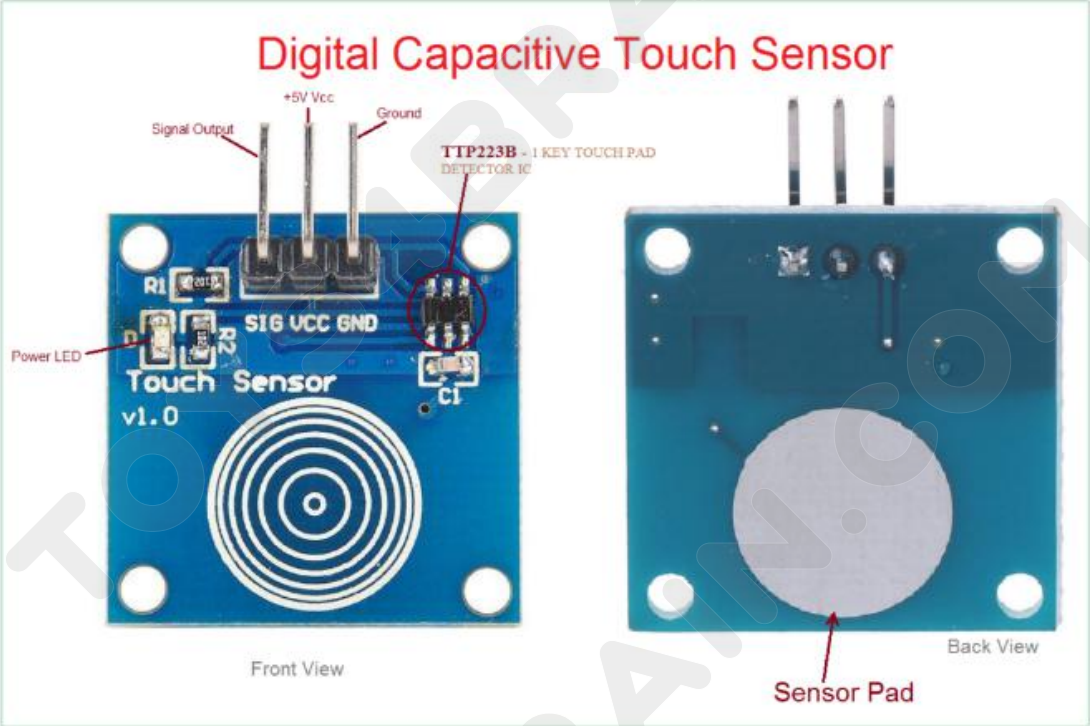
Digital touch sensor module x 1

Jumper

USB cable x 1

PC x 1

About the digital touch sensor module



Overview:

The module is based on a touch-sensing IC (TTP223B) capacitive touch switch module.

In the normal state, the module output and the power consumption are low; when a finger touches the corresponding position, module outputs high level. If it is not touched for 12 seconds, it switches to the low power mode.

Jog type: initial state is low, high touch, no touch is low (touch like button function)

The module can be mounted on non-metallic materials such as surface plastics, glass, and the like. In addition to the thin paper (non-metallic) that covers module surface, as long as the location of touch is correct , you can make hidden in the walls, desktops and other parts of buttons.

Specification:

Control interface: There are three pins (GND, VCC, SIG), GND is grounded, VCC is the power supply, SIG digital signal output pin;

Power indicator: Green LED, the power is flashing on the right side;

Touch area: Similar to the fingerprint icon inside the area, you can touch the trigger finger.

Positioning hole: The diameter of the four M2 screw locating holes is 2.2mm. Module positioning is easy to install and realize the module combination.

Features:

Low power consumption

Power supply is 2~5.5V DC

Operating current (Vcc = 3V): 1.5 – 3.0μA

Operating current (VDD = 3V): 3.5 –7.0μA

Can replace traditional buttons

Four M2 screw locating holes for easy installation

Response time: low power mode: 220ms; fast mode: 60ms

Application:

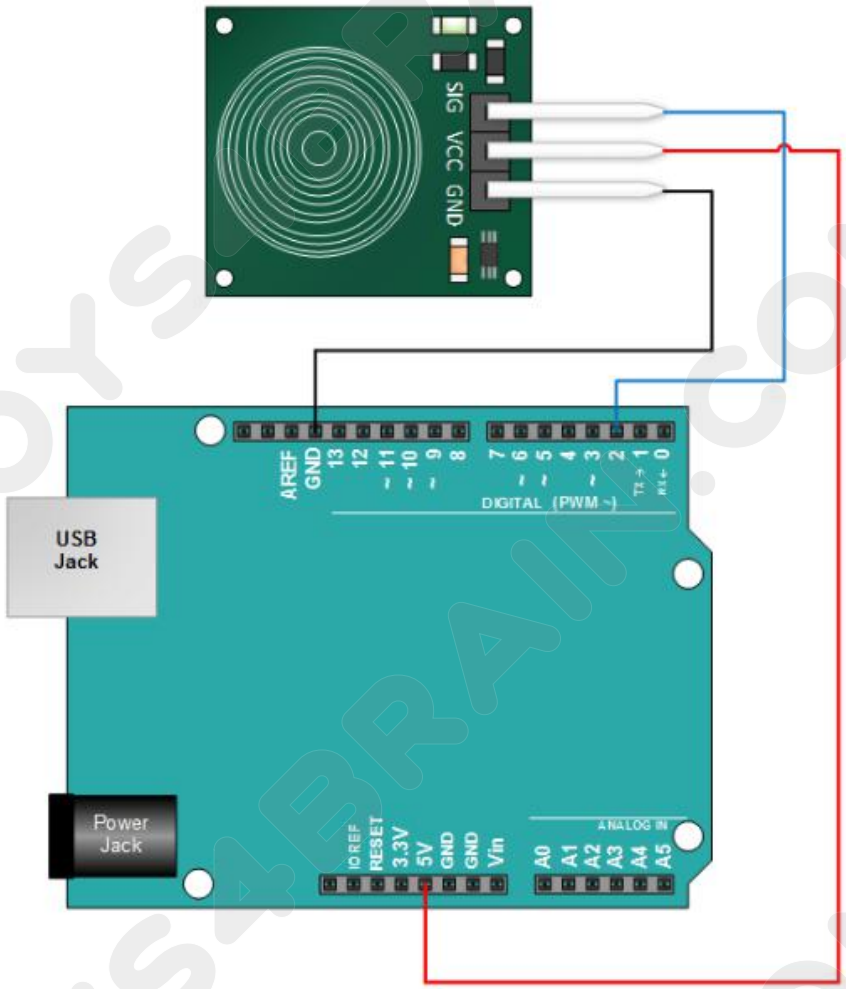
Waterproof electrical products

Button replacement

Consumer products

Example:

Connect the Vcc pin of sensor's breakout board to the +5V pin of ARDUINO and connect GND to GND. Connect the signal (SIG) pin to the ARDUINO digital pin D2.



Sample code:

```
#define ctsPin 2 // Capacitive touch sensor pins
int ledPin = 13; //LED pins
void setup()
 {
   Serial.begin(9600);

   pinMode(ledPin, OUTPUT);

   pinMode(ctsPin, INPUT);
}
```

```
void loop()
{
   int ctsValue = digitalRead(ctsPin);
   if (ctsValue == HIGH)
{
      digitalWrite(ledPin, HIGH);
      Serial.println("TOUCHED");
   }
   else
{
      digitalWrite(ledPin,LOW);
      Serial.println("not touched");
   }
   delay(500);

}
```

Result:

After uploaded, if you touch the metal surface of transducer with a finger or metal object, UNO's on-board red LED indicator will on. Open the Serial Monitor at a baud rate of 9600 and you will see the following content:

Section 10 Vibration Sensor Module

Introduction

Many applications can be created by measuring vibration levels, but detecting vibration accurately is a hard work. This article describes the vibration sensor SW-420 and ARDUINO interfaces, which can help you reduce the amount of vibration measurement work.

Ready to work

Hardware

UNO R3 board x 1

Breadboard x 1

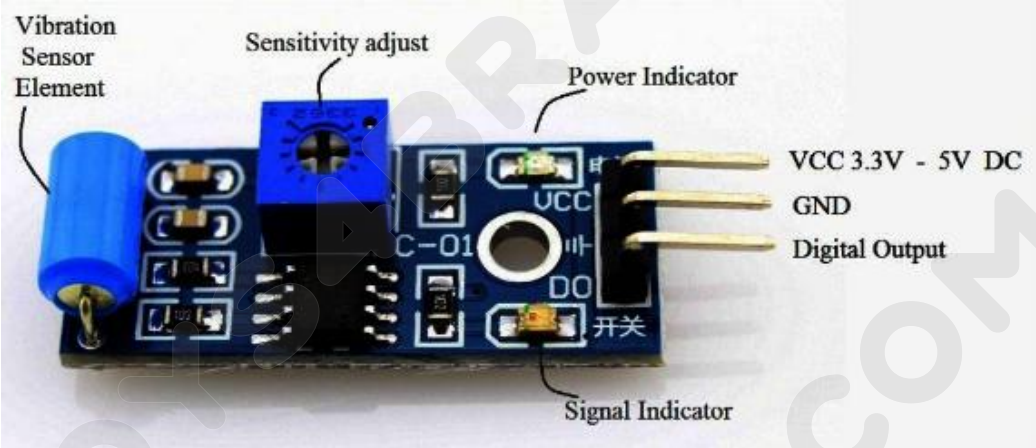
Vibration sensor module x 1

Jumper

USB cable x 1

PC x 1

About the vibration sensor module



The vibration sensor module has a SW-420 vibration sensor that integrates adjustable sensitivity through an on-board potentiometer. There are also LED indicators for power and digital output status on board. It has a simple and intuitive 3-pin interface, VCC, GND and DO (digital output). It supports 3.3V or 5V power supply.

The vibration sensor module is compatible with any micro-controller with digital inputs, thus, it is compatible with any popular micro-controllers such as PIC, ARDUINO and Raspberry Pi. The direct interface is critical to the use of this sensor.

When there is no vibration, the DO pin will be low and the indicator LED will illuminate.

Module function:

Use the SW-420 normally closed vibration sensor produced by our company.

Comparator output, clean signal, waveform, drive capability over 15mA

Working voltage 3.3V-5V

Output form: digital switch output (0 and 1)

Fixed bolt holes for easy installation

Wide voltage LM393 comparator

Use:

This vibration sensor can be used to detect vibration at any angle. On-board potentiometer adjusts the vibration threshold values. Output logic high level when the module is not triggered, and the logic low level is output when triggered.

Application ideas:

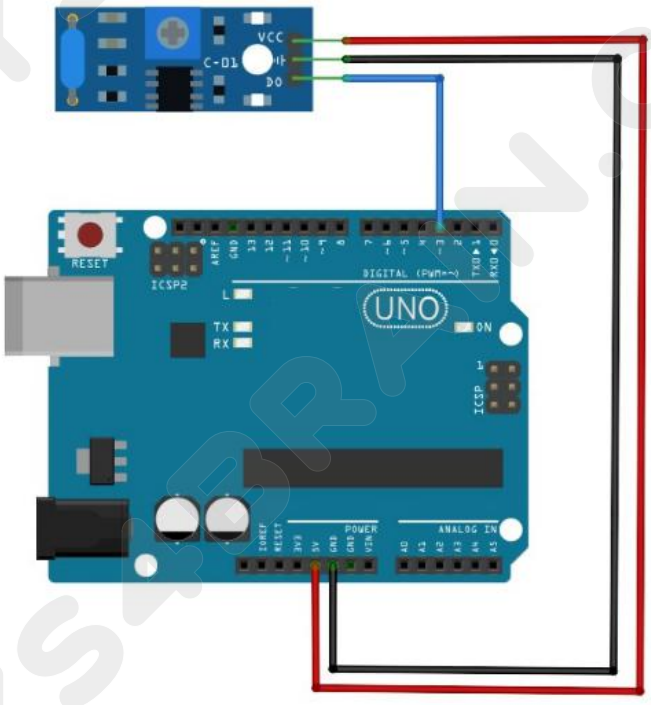
• Automatic alert

• Motion detection

• Vibration detection application function

Example:

Connect the Vcc pin of the sensor board to the 5V pin of the ARDUINO board, connect the Gnd pin to the Gnd pin of the ARDUINO, and connect the DO output signal pin of the sensor board to the ARDUINO digital pin D3. Do some calibration and adjust the sensitivity threshold, then upload the following sketch to the ARDUINO board.



In this project, we will connect the ARDUINO with the vibration sensor and LED. When no vibration is detected, the output of the vibration sensor is 0 (low voltage), otherwise its output is 1 (high voltage). If the ARDUINO gets 0 (no vibration) from the vibration sensor, it turns on the green LED and turns off the red LED. If the ARDUINO gets 1 from the vibration sensor, it will turn on the red LED and turn off the green LED (please choose the correct board type and the correct port for your ARDUINO IDE)

Sample code:

```
int vibr_pin=3;
int LED_Pin=13;
void setup()
{
   pinMode(vibr_pin,INPUT);
   pinMode(LED_Pin,OUTPUT);
}

void loop() {
   int val;
   val=digitalRead(vibr_pin);
   if(val==1)
   {
      digitalWrite(LED_Pin,HIGH);
      delay(1000);
      digitalWrite(LED_Pin,LOW);
      delay(1000);
    }
    else
    digitalWrite(LED_Pin,LOW);
}
```

The ARDUINO value reads and serially prints the code vibration value. When the measured value is greater than 1000, this code turns on the onboard LED, which you can adjust as needed.

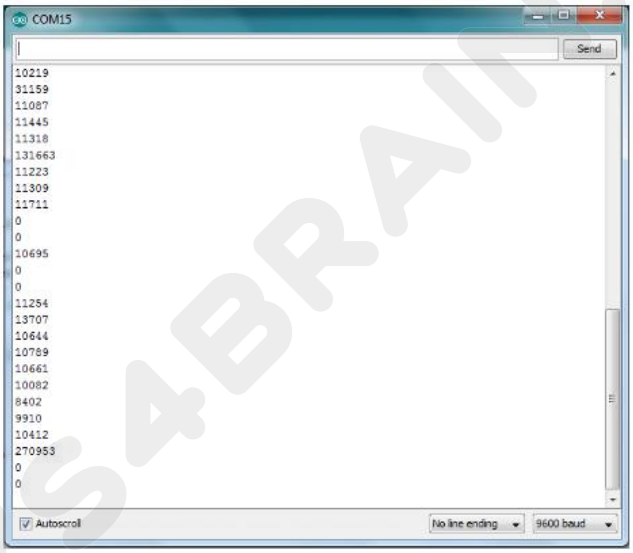
```
int LED_Pin = 13;
```

```
int vibr_Pin =3;

void setup(){
   pinMode(LED_Pin, OUTPUT);
   pinMode(vibr_Pin, INPUT); //set vibr_Pin input for measurment
   Serial.begin(9600); //init serial 9600
  // Serial.println("----------------------Vibration demo----------------------");
}
void loop(){
   long measurement =TP_init();
   delay(50);
  // Serial.print("measurment = ");
   Serial.println(measurement);
   if (measurement > 1000){
      digitalWrite(LED_Pin, HIGH);
   }
   else{
      digitalWrite(LED_Pin, LOW);
   }
}

long TP_init(){
   delay(10);
   long measurement=pulseIn (vibr_Pin, HIGH);    //wait for the pin to get HIGH
and returns measurement
   return measurement;
}
```

Finally, you should open the ARDUINO IDE serial monitor at a baud rate of 9600 and you will see:



Section 11 Sound Detection Sensor Module

Introduction

The sound detection sensor is a small board that combines a microphone and some processing circuits, and is capable of detecting sounds volume. The sensor can be used for a variety of purposes from industry to simple hobbies or play.

In this course, we will guide you the way to connect and use of this sound detection module. It will check how the circuit works, explain some details about getting the best performance from sound sensor, and then introduce some projects that demonstrate how to use it.

Preparation Work

Hardware

UNO R3 board x 1

Sound detection sensor x 1

Breadboard x 1

Jumper

USB cable x 1

PC x 1

Overview of sound detection sensors

The sound detection sensor module has a built-in capacitive electret microphone that is highly sensitive to sound. The sound wave vibrates the electret film, resulting in a corresponding change voltage, so it can detect the intensity of the sound in the surrounding environment. Since the change is very weak, it needs to be enlarged. We use the LM393 as a power amplifier here. You can adjust sensitivity by adjusting the potentiometer. When sound level exceeds the set point, LED indicator on sensor module will illuminate and the output will be sent low.

Note: This sound sensor is used to detect whether there is sound around, it is unable to recognize frequency or volume. Please do not use this module to collect sound signals.

ARDUINO sound detection sensor pin output

The following figure details the controls, pin assignments, and other key components.

When it comes to sensitivity, it's good. I mean:

When the sensitivity is low, more sound is needed to trigger device

Less sensitivity is required when triggering device

Parameter value:

ARDUINO + 5 V ------DC

ARDUINO G ----------GND

D0 is connected to the digital input pin

A0 is connected to the analog input pin

When the power is turned on, power indicator lights up.

Sound detection indicator lights up when a sound is detected

Potentiometer CW = more sensitive

CCW = low sensitivity

It has four pins that need to be connected to the ARDUINO. The top (if you look at the picture above) is AO. This should be connected to analog input 0 on the ARDUINO (A0). The next one is GND, it is grounded, VCC is connected to +5V, the last one is DO, DO is the digital output of the module and should be connected to the digital pin 2 of the ARDUINO.

At the top of the sound sensor is a small flat head screw that you can turn to adjust sensitivity and analog output. To calibrate the sound sensor, you can make some noise to keep it turning until you see the sensor LED indicator starts to flash in rhythm.

Example:

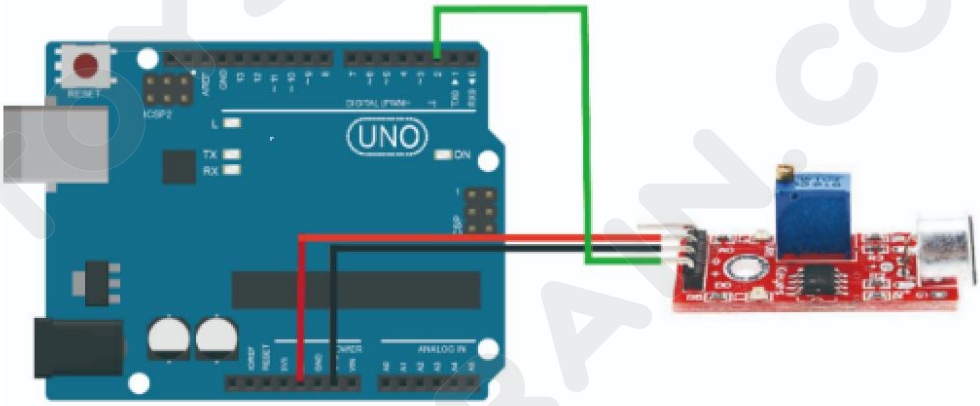
Digital detection sound sensitive lamp

In this example, we connect the sound detection sensor module to ARDUINO digital pin in order to control on-board LED, so that LED will illuminate each time the sensor detects sound.

Note: The sensitivity of sound detection sensor is adjustable – you can adjust it with a potentiometer.

Connection

Here, we use D2 as a digital pin to connect with sound sensor, and build circuit as follows:



Code program:

Once above operation is done, connect ARDUINO board to your computer with a USB cable. The green power LED (labeled PWR) should illuminate. Open ARDUINO IDE and select appropriate board type and port type for your project.

Then load the following sketch onto your ARDUINO.

```
int digit_sensor = 2; // select the input pin for the potentiometer
int ledPin = 13; // select the pin for the LED
  int digitValue ; //    value from the digit input pin
void setup ()
{
pinMode (ledPin, OUTPUT);
pinMode (digit_sensor, INPUT);
Serial.begin (9600);
}
  void loop ()
  {
```

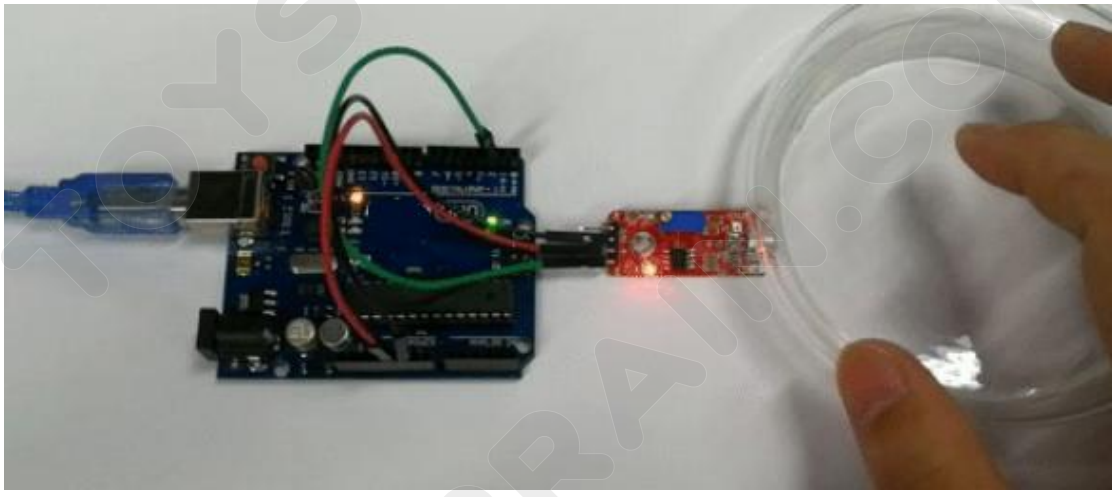
```
digitValue=digitalRead(digit_sensor);
if (digitValue==LOW)
{
digitalWrite (ledPin, HIGH); delay(50);
}
else
  {
digitalWrite (ledPin, LOW); delay(10);
}

}
```

This is the code that causes LED to flash with sound. You must set its threshold to be smart enough to make the LED blink, and you can also read the value of the sound intensity on the serial monitor.

Operation result:

After copying and uploading this code, the LED connected to pin 13 on Uno board will illuminate when volume reaches a certain value. If the sound doesn't feel good, try turning the potentiometer to change threshold or sensor sensitivity.



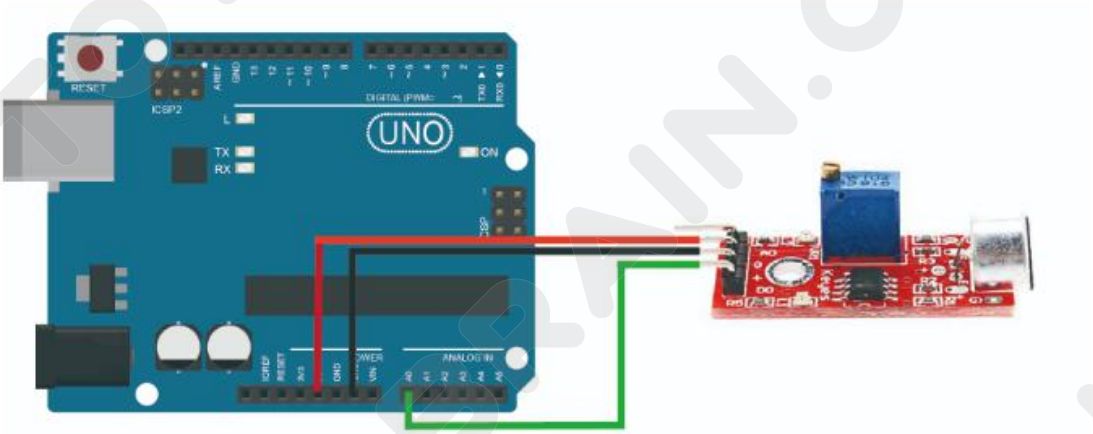
Note: If the LED is not lit or is always on, you need to change the sensitivity of the sensor by turning the potentiometer.

Analog detection sound sensor light

In this example, we will demonstrate how to use analog pins to detect sound. Microphone sensor is able to detect sound intensity of surrounding environment and LED will illuminate if it exceeds a certain threshold.

Connection

Here, we use A0 as an analog pin to connect with sound sensor and build circuit as following:



Code program:

Once this is done, connect ARDUINO board to your computer with a USB cable. The green power LED (labeled PWR) should illuminate. Open the ARDUINO IDE and select the appropriate board type and port type for your project. Then load the following sketch onto your ARDUINO.

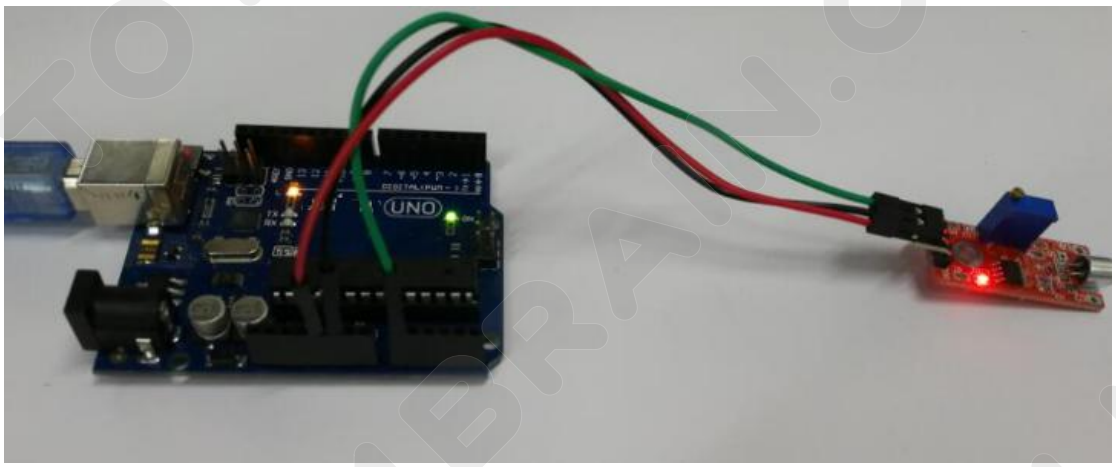
```
const int ledPin = 13; //pin 13 built-in led const
int soundPin = A0; //sound sensor attach to A0
int threshold = 600; //Set minimum threshold for LED lit
void setup()
  {
pinMode(ledPin,OUTPUT);//set pin13 as OUTPUT
Serial.begin(9600); //initialize serial
}
void loop()
{
```

```
int value = analogRead(soundPin);//read the value of A0
Serial.println(value);//print the value
if(value > threshold) //如果该值大于 600
 {
digitalWrite(ledPin,HIGH);//turn on the led
delay(200);//delay 200ms }
else
 {
 digitalWrite(ledPin,LOW);//turn off the led
}
delay(1000);
}
```

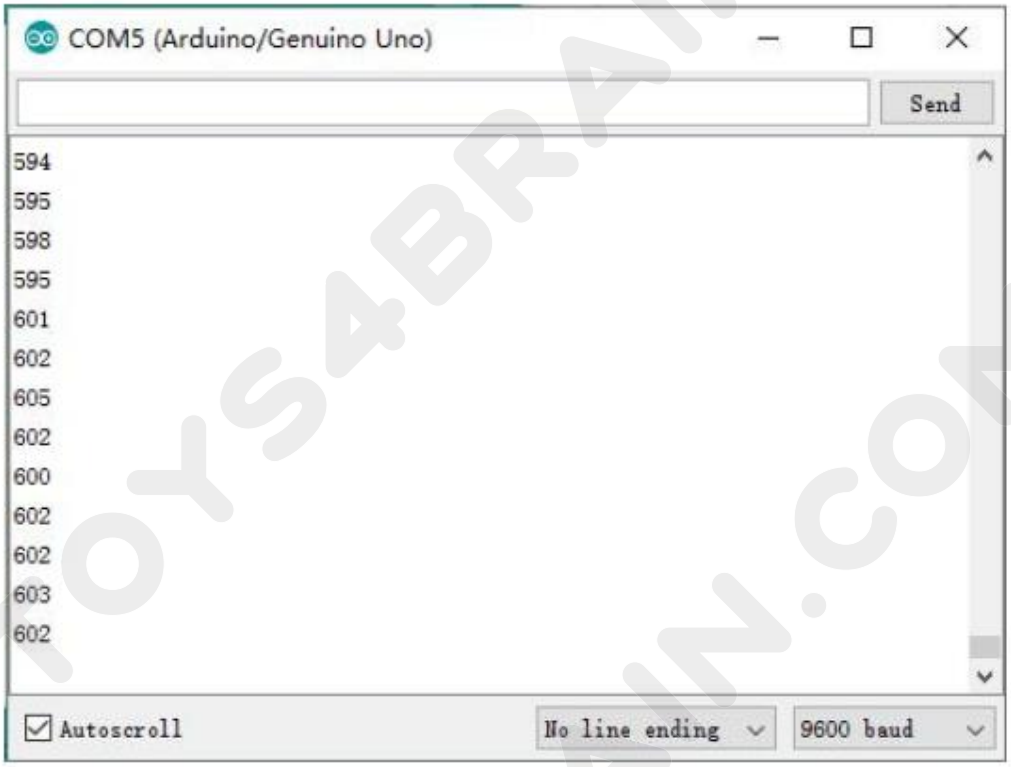
This is the code that causes the LED to flash with the sound. You must set the threshold to be smart enough to make the LED blink, and you can also see the value of the sound intensity on the serial monitor.

Operation result

After copying and uploading this code, the LED connected to pin 13 on the Uno board will illuminate when the volume reaches a certain value. If the sound doesn't feel good, try turning the potentiometer to change threshold value or sensor sensitivity.



You can open the serial monitor by going to Tools > Serial Monitor or by pressing the magnifying glass button in ARDUINO software window.

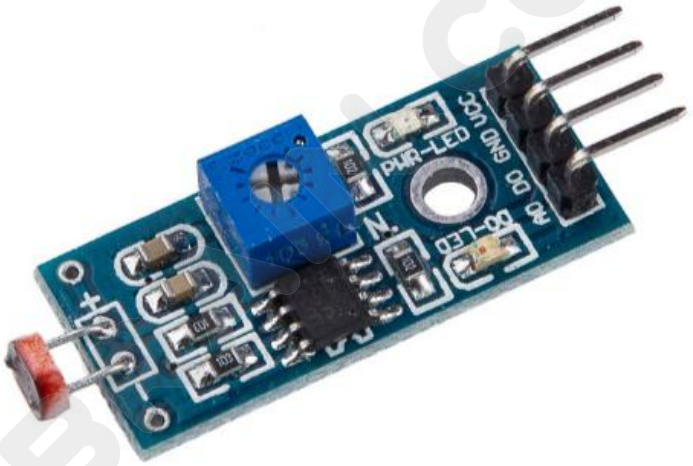


The analog and digital values of the sound sensor module are printed. When noise occurs, the analog value should rise and remain stable when quiet again. Now, in the code, there is an "int threshold = 600;" line that needs to be changed to a value that is very close but higher than the serial monitor (when your surrounding is quiet). For example, if you see an analog value of 500, you should change the threshold to 503 or 505. When a sound is made, the analog value will rise and exceed threshold value. In this circumstance, your LED will illuminate. When quiet again, analog value will return to 503 and the LED will go out again.

Section 12 Photosensitive Sensor Module

Introduction

In this course, we will show how to use a photoresistor with Osoyoo UNO. We will monitor the output of the photoresistor and let ARDUINO know weather it is light or dark. When the light is below a certain level, the ARDUINO turns on an LED.



Preparation

Hardware

UNO R3 board x 1

Breadboard x 1

Photosensitive resistor module x 1

LED x 1

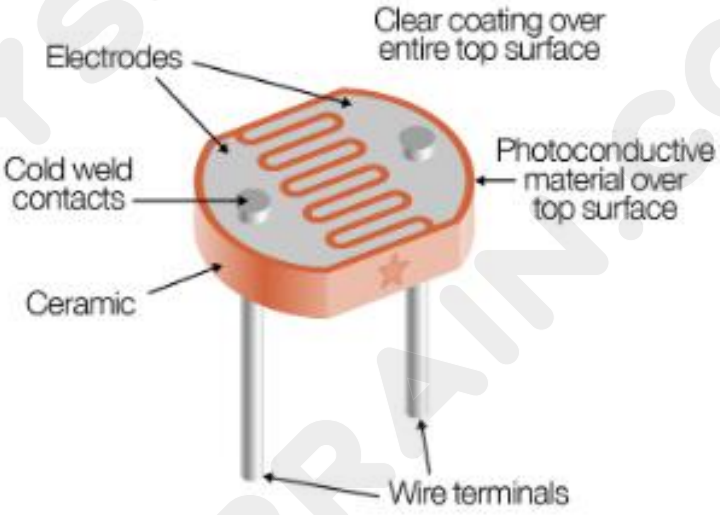
M / M jumper

USB cable x 1

PC x 1

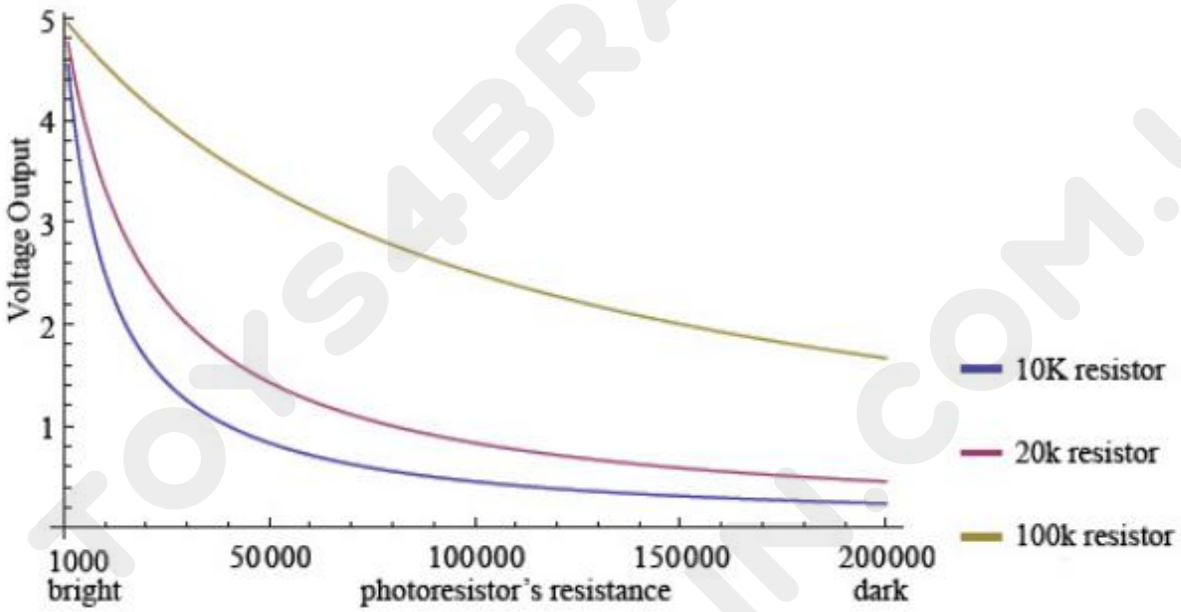
About photoresistor

Photocells are sensors that detect light. They are small, inexpensive, lower power, easy to use and not wear out. Therefore, they often appear in toys, gadgets and household appliances. They are commonly referred to as CdS batteries (which are made of cadmium sulfide), photoresistors (LDR) and photoresistors.



A photocell is basically a resistor that changes its resistance (in ohms) depending on how much light is shining on the curved surface. In the dark, photoresistor resistance value can be reach as high as MΩ. However, when light is weak, the resistance of the photoresistor may be as low as several hundred ohms. They are very low cost and are easily available in many sizes and sizes, but they are very inaccurate. The action of each photocell sensor will be different from other photocells, even if they are from the same batch. The difference can be very large, up to 50% or higher! Therefore, they should not be used to determine the exact illumination of a lux or millicandela. Instead, you can expect to only determine the basic lighting changes.

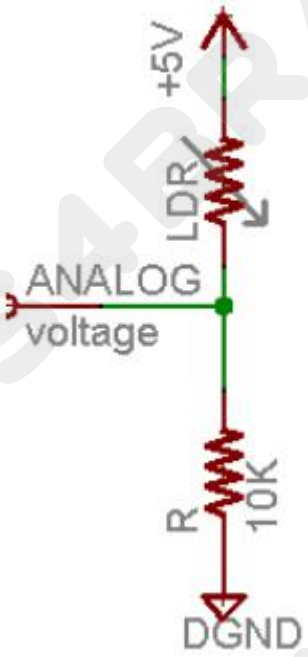
This figure shows the resistance of the sensor at different illumination levels:



Connection

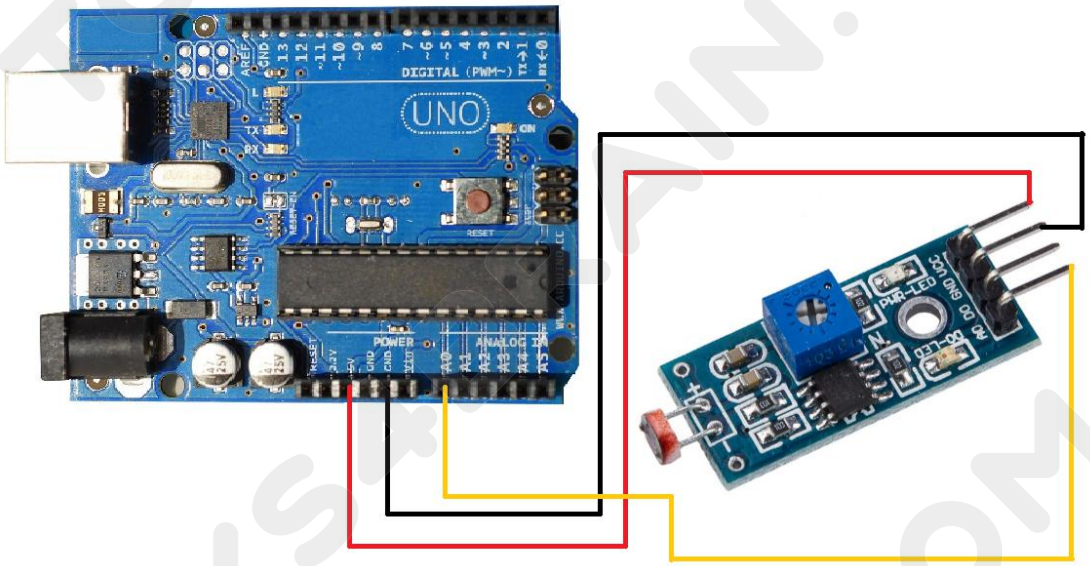
Connect the components as shown below. Connect the LED to pin 9 of the ARDUINO. The 200 ohm resistor is a current limiting resistor. One lead of the photoresistor is connected to 5V, and the other lead is connected to one lead of the 10k ohm resistor. The other lead of the 10k ohm resistor is grounded. This forms a voltage divider whose output is connected to the A0 pin of the

ARDUINO.



As the light impinging on the photoresistor becomes stronger, the resistance decreases and the voltage output of the voltage divider increases. When the incident light is weak, the opposite happens.

Illustration:



Upload a sketch

Once this is done, connect the ARDUINO board to your computer with a USB cable. The green power indicator (labeled PWR) should illuminate.

Code:

```
Int photocellPin = A0; //Select input pin of the photoresistor
Int ledPin = 13; // select LED pin
Int val = 0; // variable to store the value from sensor
Void setup()
{
Serial.begin(9600); //Set the baud rate to 9600, making sure it is the same as your software settings
pinMode(ledPin, OUTPUT); // declare ledPin as output
pinMode (photocellPin, INPUT); / / declare ledPin as output
}
Void loop()
{ val = analogRead(photocellPin); // Read the value from sensor
Serial.println(val); //Serial number will print out the light value
If(val<=512) // The point at which the LED status changes
  {
digitalWrite(ledPin, HIGH); // turn on LED

}
Else
{
digitalWrite(ledPin, LOW); //turn off the LED
}
}
```

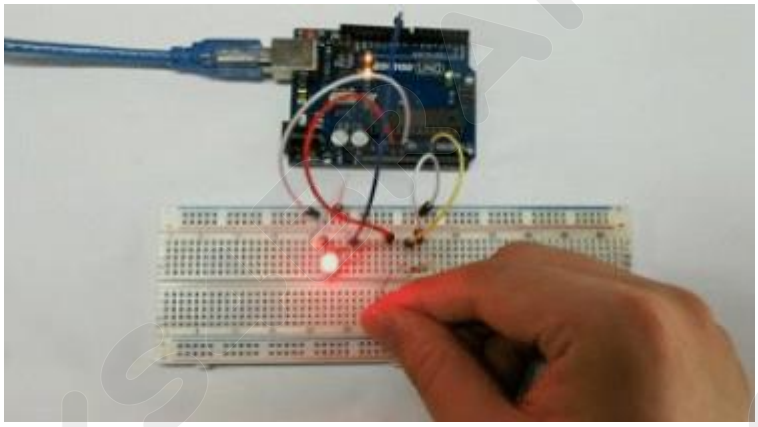
Compile and Upload

Open the ARDUINO IDE and select the appropriate board type and port type for ARDUINO board.

Once you have compiled this sketch, simply click on the "Upload" button in your environment. Wait a few seconds – you should see the RX and TX on-board LEDs flash. If the upload is successful, the message "Complete Upload" will be displayed in the status bar.

Result

If the room is illuminated, LED indicator should be off. Try covering the photoresistor with your hand to turn it on. Remove your hands and watch them off again.



At the same time, turn on serial monitor and you will get the following output data:

Note:

When using a serial monitor, make sure the baud rate setting is the same as the sketch definition.

Section 13 Passive Buzzer Module Experiment

Introduction

In this lesson we will demonstrate how to use a passive buzzer to make a sound.

About buzzer

As a whole structure of electronic buzzer, buzzer powered by DC power supply is widely used in computers, printers, copiers, alarms, electronic toys, automotive electronics, telephones, timers and other electronic devices for voice equipment. product. The buzzer can be divided into an active buzzer and a passive buzzer (see the figure below). Place two buzzer pin face up, the buzzer with green board is passive buzzer, and the other is an active buzzer with black tape.

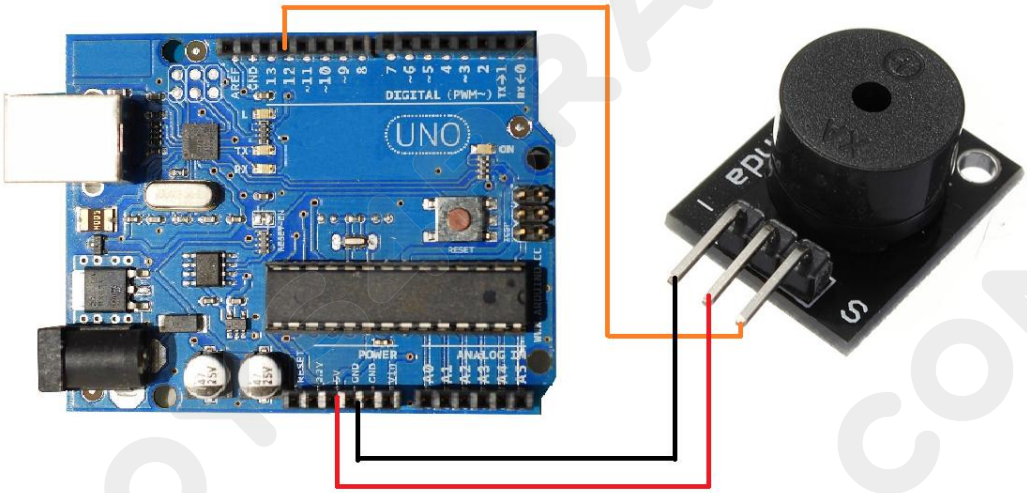
The difference between an active buzzer and a passive buzzer is:

The active buzzer has a built-in source of oscillation, so it produces sound when it is energized. However, passive buzzers do not have such a source, so if a DC signal is used, it will not beep. In contrast, you need to drive it with a square wave between 2K and 5K. Active buzzers are typically more expensive than passive buzzers due to the presence of multiple built-in oscillating circuits. In this course we will use a passive buzzer.

Note:

The active buzzer has a built-in oscillator source, so a beep will sound as soon as it is energized, but it can only beep at a fixed frequency.

Illustration:



Upload Sketch

Once this is done, connect the ARDUINO board to your computer with a USB cable. The green power indicator (labeled PWR) should illuminate.

Code:

```
int buzzer = 12;//the pin of the active buzzer
void setup() {
pinMode(buzzer,OUTPUT);//initialize the buzzer pin as an output
}
```

```
void loop() {
unsigned char i;
while(1)
{ //output an frequency
for(i=0;i<80;i++) {
digitalWrite(buzzer,HIGH);
delay(1);//wait for 1ms
digitalWrite(buzzer,LOW);
delay(1);//wait for 1ms
} //output another frequency
for(i=0;i<100;i++) {
digitalWrite(buzzer,HIGH);
delay(2);//wait for 2ms
digitalWrite(buzzer,LOW);
delay(2);//wait for 2ms
}
}
}
```

Once you have compiled this sketch, simply click on the "Upload" button in your environment. Wait a few seconds – you should see the on-board RX and TX LEDs flash. If upload succeed, the message "Complete upload." will appear in status bar.

Result

You should hear a beep after a few seconds of uploading.

Note Again:

The active buzzer has a built-in oscillator source, so a beep will sound as soon as it is energized, but it can only beep at a fixed frequency.

Section 14 PIR Motion Sensor Experiment

Introduction

A passive infrared sensor (PIR motion sensor) is an electronic sensor that measures infrared (IR) light emitted from objects in its field of view. They are most commonly used in PIR based motion detectors. Therefore, it can detect motion based on changes in infrared light in the environment. It is desirable to detect if inspector has moved in or out of sensor range. In this course, we will learn how PIR sensor works and how it can be used with ARDUINO development board to detect motion.

Work Preparation

Hardware

UNO R3 board x 1

PIR motion sensor x 1

Relay x 1

Breadboard x 1

Jumper

USB cable x 1

PC x 1

Overview of PIR motion sensor:

The PIR motion sensor allows you to sense motion and is almost always used to detect if a human has entered or left the sensor range. They are small, inexpensive, low power, easy to use and not subject to wear. Therefore, they are usually found in appliances and gadgets used in homes or businesses. They are often called PIR, "passive infrared", "electric heating" or "IR motion" sensors.

Using this ARDUINO motion sensor to build an anti-theft alarm system, home automation system or any other simple gadgets that prevent people from entering the room!

Specifications:

Working voltage: 4.5V to 20V

Output: High: 3.3V, Low: 0V

Detection angle: about 120 degrees

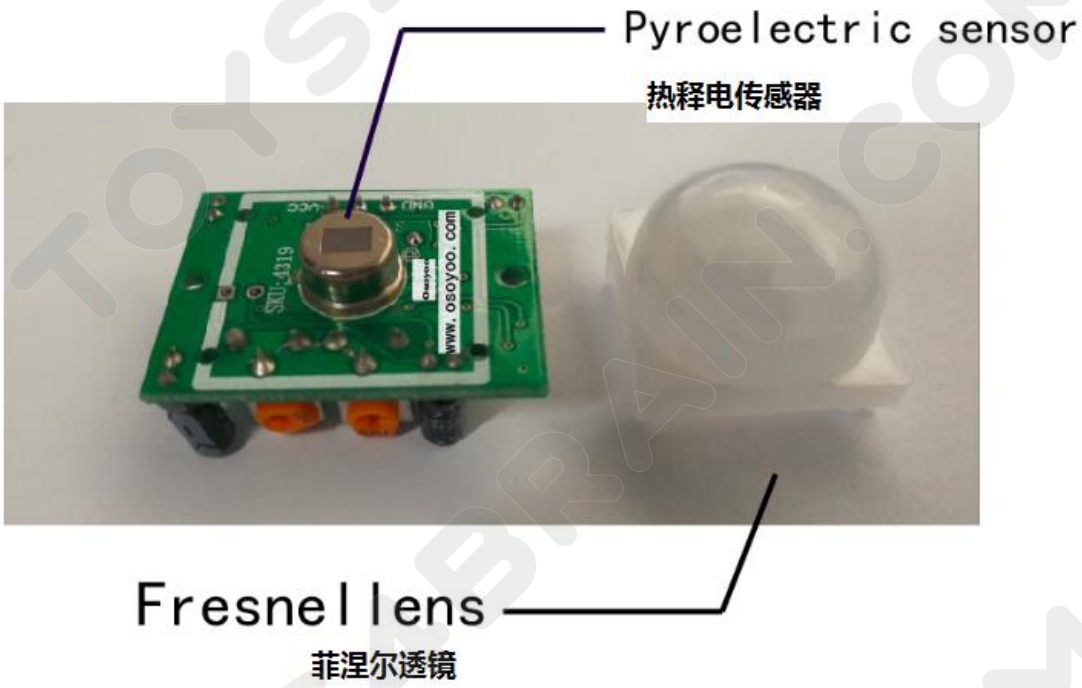
Range: adjustable, up to 7m

Trigger mode: L non-repeatable trigger / H repeatable trigger (default)

Residence time: (holding time) adjustable between 5 and 300 seconds. It can be further increased by increasing the value of the CY1-Timing capacitor on IC pin 4.

Working temperature: -20 – +80 degrees Celsius

Lens: 11 mm high and 23 mm in diameter.

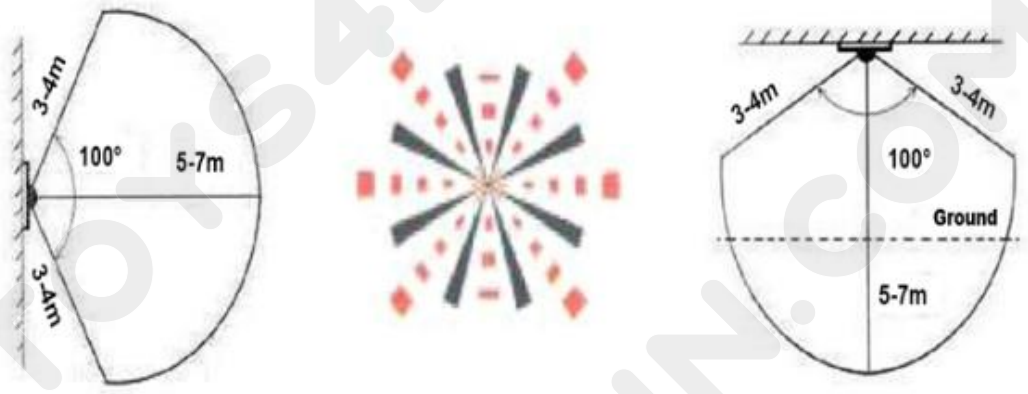


The PIR is basically made of a pyroelectric sensor (you can see above is a circular metal with a rectangular crystal in the center) that is able to detect infrared radiation level. Everything gives off some low levels of radiation, and the hotter item is, the more radiation it emits. In fact, the sensor in motion detector is divided in half, that is the reason why we want to detect motion change rather than average IR level. Connecting the two halves to offset each other. If half of the infrared radiation is more or less than the other half, the output will swing high or swing low.

Afterwards, place the sensor behind a polygon mirror (Fresnel lens) that

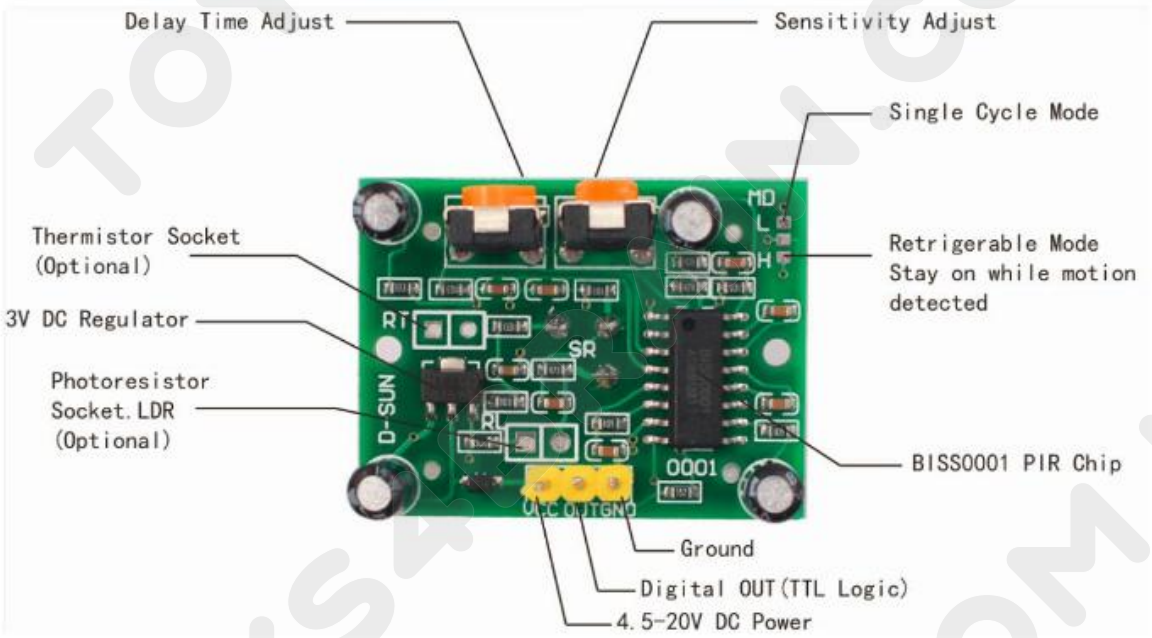
"divides" the world's field of view into smaller cones with improved visibility and intermediate areas of reduced visibility, expanding useful observations and detect angle significantly.

PIR sensing angle diagram:



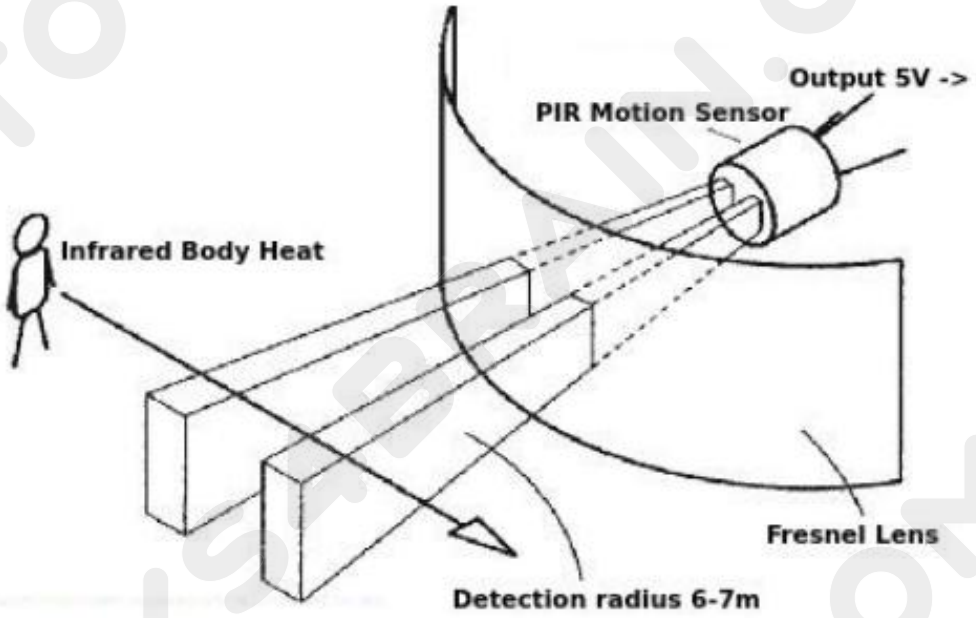
In addition to the pyroelectric sensor, there are a bunch of support circuits, resistors and capacitors. It seems that most small enthusiasts use the BISS0001 sensor (Micropower PIR Motion Detector IC), which is undoubtedly a very inexpensive chip. The chip receives output of sensors and performs some small processing to send a digital output pulse from the analog sensor.

Our PIR motion sensor is as follow:

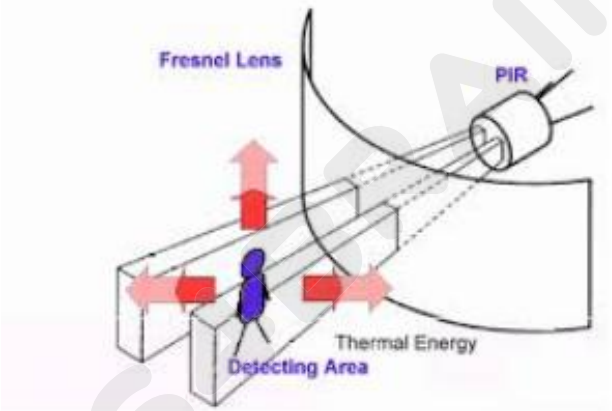


How does it work?

Here we are using a PIR motion sensor. PIR stands for Passive Infrared. The motion sensor is composed of a Fresnel lens, an infrared detector and an auxiliary detection circuit. The lens on the sensor focuses all of infrared radiation around it to the infrared detector. Our body produces infrared heat and is therefore absorbed by motion sensors. The sensor immediately outputs 5V signal for one minute when it detects someone. It provides an initial detection range of approximately 6-7 m and is highly sensitive.



When the PIR motion sensor detects someone, it outputs 5V signal to ARDUINO. Therefore, the interruption of ARDUINO is triggered. We defined the actions that ARDUINO should take when it detects an intruder.



Sensitivity adjustment

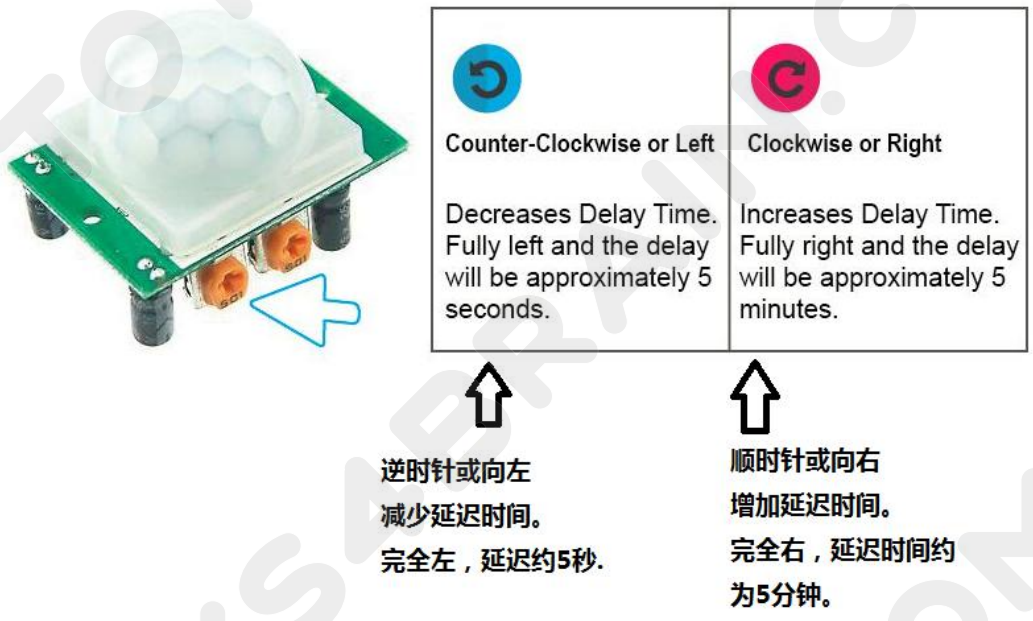
As mentioned earlier, the adjustable range is approximately 3 to 7 meters. The figure below shows this adjustment. You can click to enlarge the illustration.

反方向-顺时针或向左
增加灵敏度。
完全向左，射程将近
7米.

顺时针或向右
右移，范围约为3米

Delay time adjustment

Delay adjustment determines the amount of time that output of PIR sensor module will remain high after motion is detected. The range is from about 5 seconds to 5 minutes. The figure below shows this adjustment.



逆时针或向左
减少延迟时间。
完全左，延迟约5秒.

顺时针或向右
增加延迟时间。
完全右，延迟时间约
为5分钟。

Start with detecting the last action.

5 seconds off after delay is completed – important

After time delay is complete, the output of this device will go low (or off) for about 5 seconds. In other words, all motion detection is blocked in these three seconds.

Example:

Suppose you are in one-shot mode (see below) and your time delay is set to 5 seconds.

The PIR will detect the motion and set it to high level for 5 seconds.

After five seconds, the PIR sets its output low for about 3 seconds.

Within three seconds, the PIR will not detect motion.

After three seconds, the PIR will detect motion again, and the detected motion will again set output at high level and remain on, it depends on delay time adjustment and trigger mode selection.

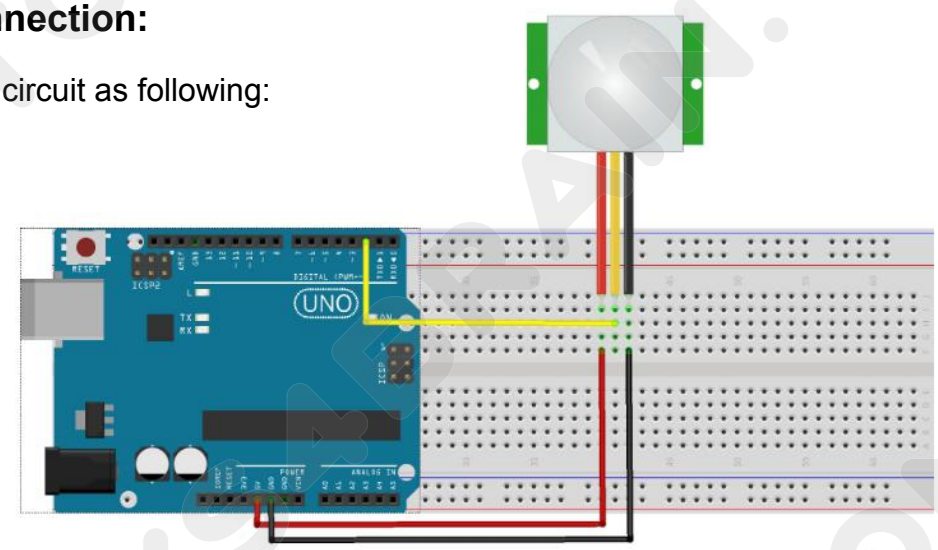
Illustration:

PIR motion sensor controls LED

In this project, you will create a simple circuit by using ARDUINO and PIR motion sensors. When the movement is detected, LED indicator will be on.

Connection:

Built circuit as following:



Connecting PIR sensor to the microcontroller is really easy. PIR is used as a digital output, so all you need to do is monitor the pin flips high-level(detected) or flip low-level(not detected). Powering and grounding the PIR with 5V. Then connect the output with digital pin. In this example we will use pin 2.

Code:

After the above operation is completed, connect the ARDUINO board to your computer with a USB cable. The green power LED (labeled PWR) should illuminate. Open the ARDUINO IDE and select the appropriate board type and port type for your project. Then load the following sketch onto your ARDUINO.

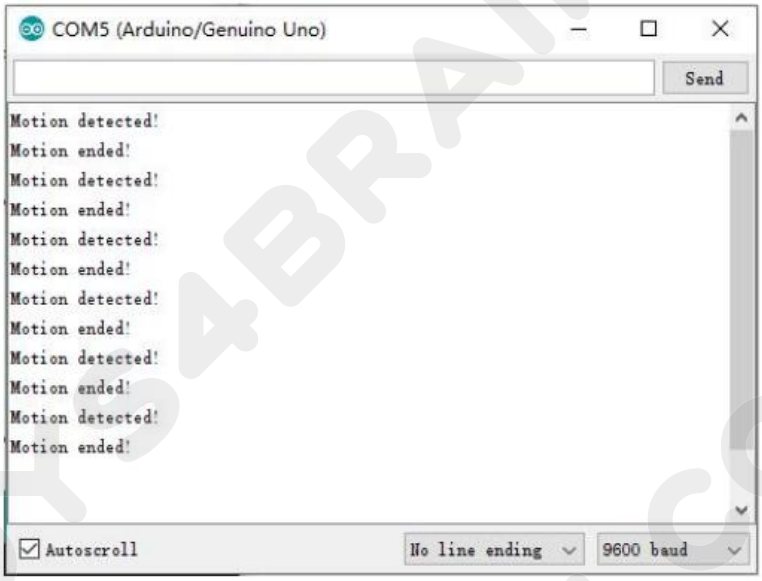
```
int ledPin = 13; // choose the pin for the LED
int inputPin = 2; // choose the input pin (for PIR sensor)
int pirState = LOW; // we start, assuming no motion detected
int val = 0; // variable for reading the pin status
void setup()
{
pinMode(ledPin, OUTPUT); // Declare LED as output
pinMode(inputPin, INPUT); // Declare the sensor as an input
Serial.begin(9600);
}
void loop()
{ val = digitalRead(inputPin); //Read input value
  if (val == HIGH)
{ // check if the input is HIGH
digitalWrite(ledPin, HIGH); // turn LED ON
if (pirState == LOW)
  { // we have just turned on
Serial.println("Motion detected!"); // We only want to print on the output change,
```

```
not state pirState = HIGH;
}
}
else
{
  digitalWrite(ledPin, LOW); // turn LED OFF
  if (pirState == HIGH)
{ // we have just turned of
Serial.println("Motion ended!"); // We only want to print on the output change,
not state pirState = LOW;
  }
}
}
```

This code only tracks whether the input of pin 2 is high or low. It also tracks pin status to print a message when the motion starts and stops.

Result:

After a few seconds of uploading, take a look at the ARDUINO's pin 13 LED. You can also open serial monitor and set baud rate to 9600 bps, you might see the following:



The PIR sensor requires a few seconds of immobility and “views” its viewing area. Move until the LED of the 13th pin goes out, then wave your hand.

Section 15 2-channel relay module

Introduction

The relay is an electric switch. Many relays use electromagnets to operate switches mechanically, but other operating principles, such as solid-state relays, are also used. Relays are used where circuits need to be controlled by a single low-power signal, or where multiple circuits must be controlled by a single signal.

In this course, we will show you how the 2-channel relay module works and how it works with the Uno R3 board to control high voltage equipment.

Work preparation

Hardware

UNO R3 board x 1

2-channel relay module x 1

Breadboard x 1

Jumper

USB cable x 1

PC x 1

About 2-channel relay module

This is a 5V 2 channel relay module board that can control various devices and other devices with high current. It can be controlled by the microcontroller directly (Raspberry Pi, Arduino, Arduino, 8051, AVR, PIC, DSP, ARM, ARM, MSP430, TTL logic). Items that are useful for applications, such as microcontroller-based items, remote controls, lights off, and circuits that need to isolate large current and high voltage switches by applying any TTL or CMOS level voltage.

Features:

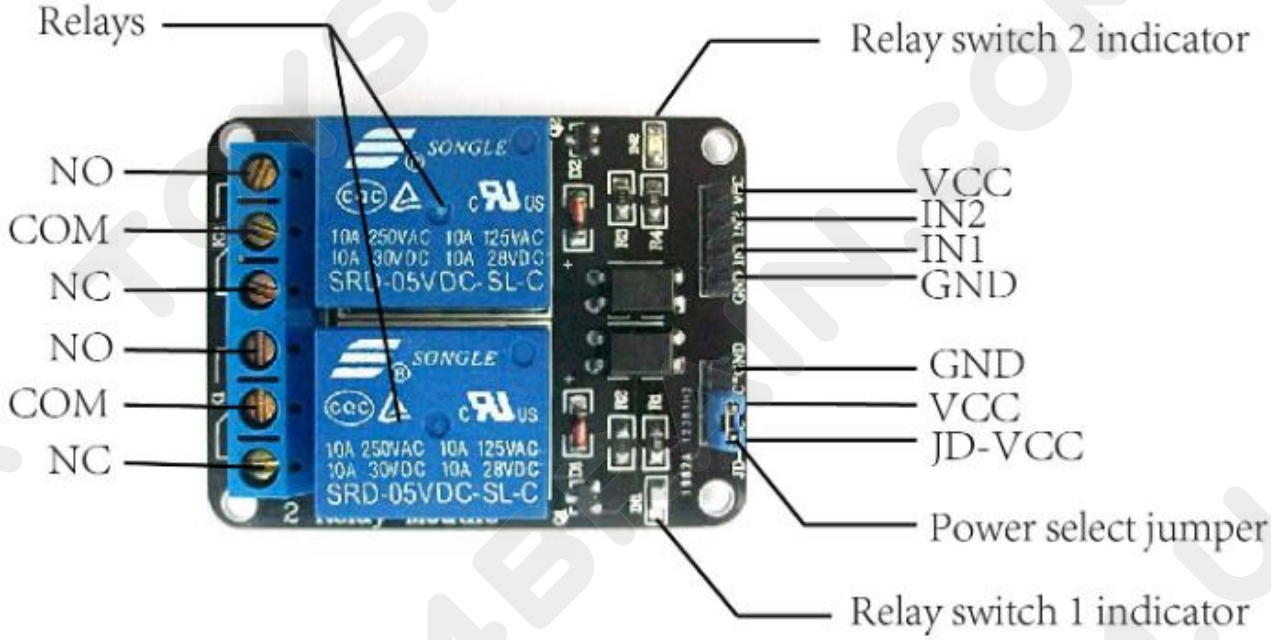
Large current relay, AC250V 10A, DC30V 10A

2 LEDs indicate when the relay is on

Suitable for logic level signals of 3.3v or 5V devices

Photoelectric isolating circuit

PIN:



Input:

It has a 1 x 4 (2.54 mm pitch) pin connector for connecting the power supply (5V and 0V) and controlling 2 relays. The pins are marked on PCB:

GND – Connect 0V to this pin.

IN1 - control relay 1, low level effective! When this input is below about 2.0V, the relay is turned on

IN2 - control relay 2, low level effective! When this input is below about 2.0V, the relay is turned on

VCC - connect 5V to the pin. It is used to power the optocoupler and a second 1×3 (2.54mm spacing) pin row is used to supply 5V voltage to "relay side" of the plate. The connector has a jumper for selecting a 5V signal from the 1×4 pin connector to power the relay. Do not change this jumper for default action!

The pins of the 1×3 pin header are marked on PCB:

JD-VCC – This is the 5V required for relay. There are jumpers on this and adjacent (VCC) pins when delivered.

VCC – This is the 5V VCC available on a 1×4 pin connector

GND – 0V pin connected to a 1×4 pin connector

If opto-isolation is required, an isolated 5V power supply should be used. For normal operation, the jumper between pins 1 and 2 selects 5V signal from 1x4 pin header. This means that both "input side" and "relay side" use the same 5V supply and there is no optical isolation.

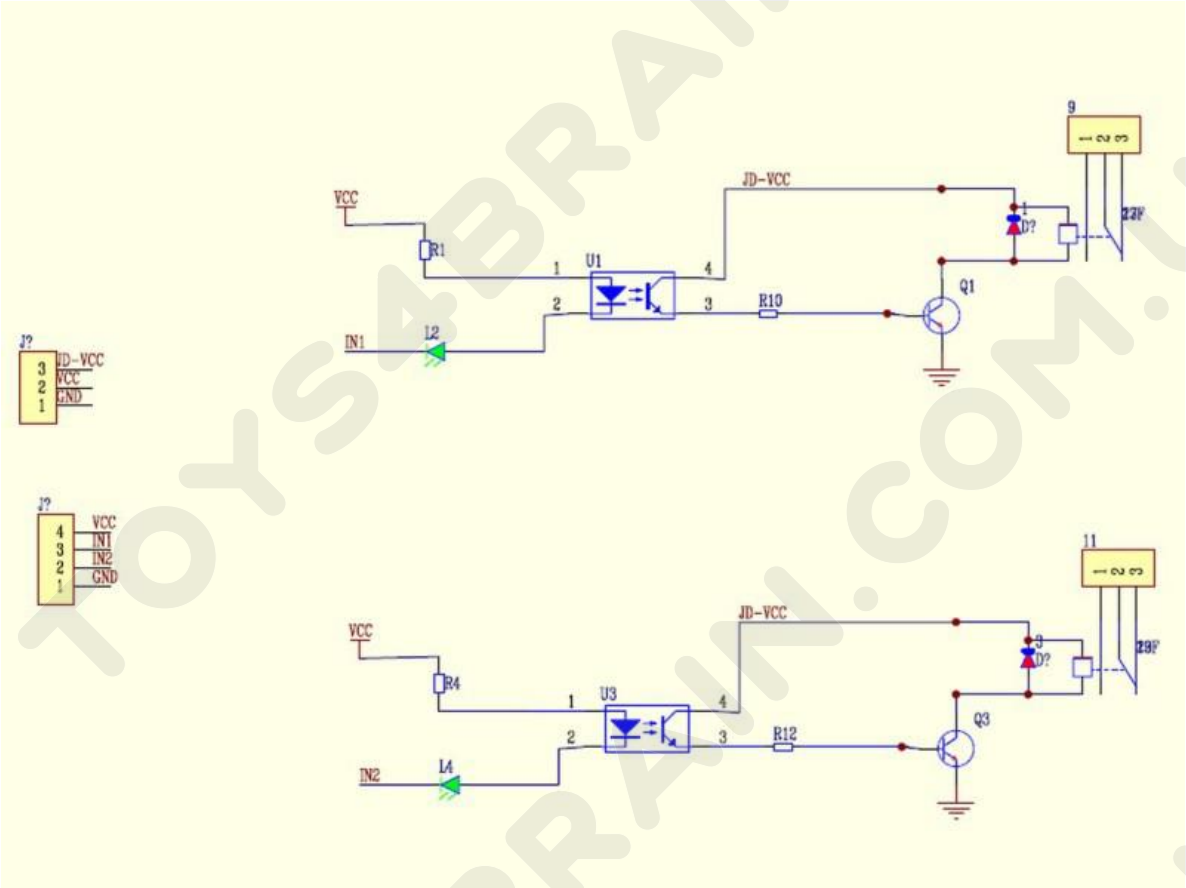
Output:

The 2-channel relay module can be regarded as a series switch: 2 normally open (NO), 2 normally closed (NC) and 2 common pins (COM).

COM- public pin

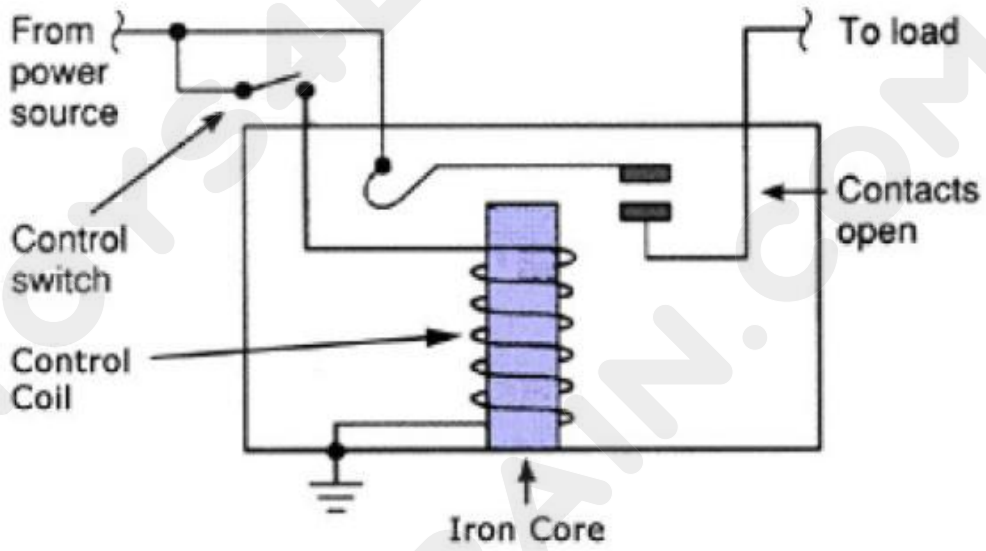
Nc-normally closed, in this case, when INT1 is set to low power, NC is connected to COM, when INT1 is set to high power, NC disconnects and No connects. In this case, when INT1 is set to low power, NO disconnects from COM1. When INT1 is set to high power, NO is connected to NO.

Schematic diagram:



How does the relay work?

By interpreting the diagram given below, you can better understand how the relay works.



Each relay has five parts:

1. Electromagnet -- it is composed of an iron core and a coil. When an electric current passes through it, it becomes magnetic. So it's called an electromagnet.

2. Armature - a removable magnetic strip is called an armature. When current flows through them, the coil is energized, creating a magnetic field that is used to form or destroy normally open (N/O) or normally closed (N/C) points. The armature can be moved by direct current (DC) and alternating current (AC).

3. Spring - when no current flows through electromagnetic coil, the spring pulls the armature apart, so the circuit cannot be completed.

4. Electrical contact group -- there are two contacts:

- Normally open - the relay is connected when activated and disconnected when inactive.
- Normally closed - the relay is not connected when activated and not connected when inactive.

Principle:

This diagram shows the internal cross section of relay. The core is surrounded by a control coil. As shown, the power supply is provided to the electromagnet through a control switch and load contacts. When the current begins to flow through control coil, the electromagnet starts to energize, increasing magnetic field. Therefore, the upper contact arm begins to be attracted by the lower fixed arm, thus closing the contact and causing a short circuit in the load power supply. On the other hand, if the relay has lost power when the contact is closed, the contact will move in the opposite direction and break.

Once the coil current is disconnected, the movable armature will return to its original position by force. The force will be almost half strength of the magnetic force. The force is mainly supplied by two factors. They are both springs and gravity.

Relays are mainly used for two basic operations. One is for low pressure applications and the other is for high pressure. For low voltage applications, it is more desirable to reduce noise throughout the circuit. For high voltage applications, they are primarily used to reduce a phenomenon called arc.

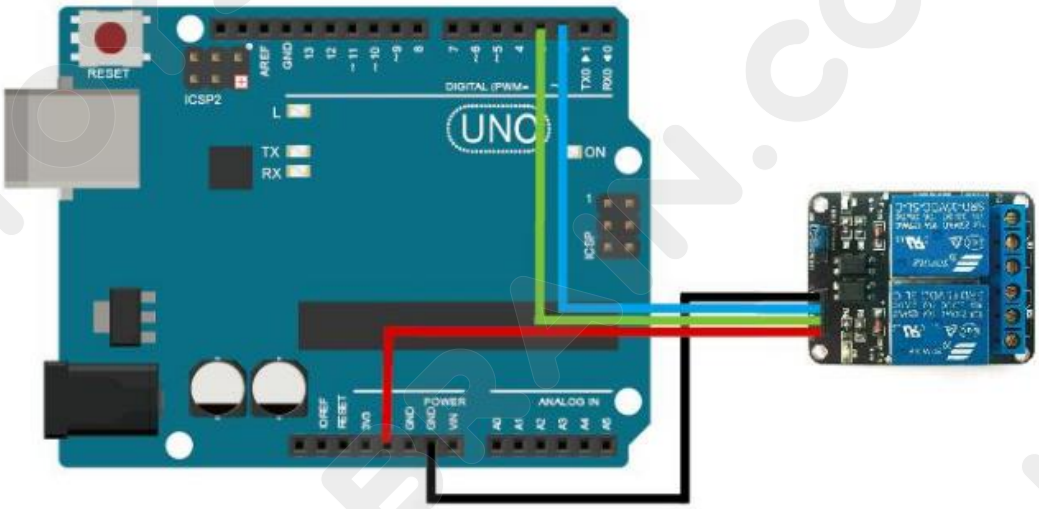
High voltage warning:

Before continuing this course, I would like to warn you that we will use high pressure, and if used improperly or improperly, it may result in serious personal injury or even death. So be careful.

Example:

Use Arduino to control the 2 channel relay

In this example, the LED on the relay lights up when low power is provided to signal terminal of 2-channel relay. Otherwise, it will turn off. If high and low levels are supplied to the signal terminal on a regular basis, you can see that the LED will cycle between on and off.



Code:

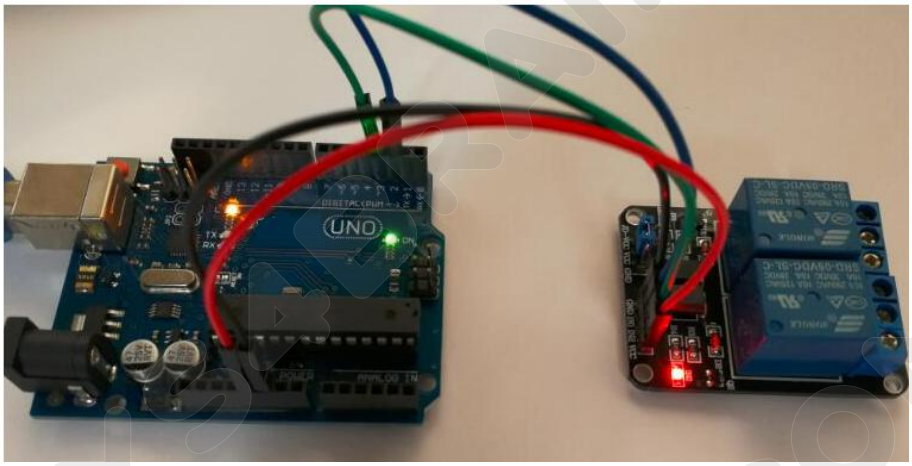
After this is done, connect the Arduino board to computer using a USB cable. The green power LED indicator light (marked PWR) should be on. Open Arduino IDE and select the appropriate board type and port type for your project. Then load following sketch onto your Arduino.

```
//the relays connect to
int IN1 = 2;
int IN2 = 3;
#define ON 0
#define OFF 1
void setup()
{
```

```
relay_init();//initialize the relay
 }
void loop()
{ relay_SetStatus(ON, OFF);//turn on RELAY_1
delay(2000);//delay 2s
relay_SetStatus(OFF, ON);//turn on RELAY_2
delay(2000);//delay 2s
 }
void relay_init(void)//initialize the relay
{ //set all the relays OUTPUT
pinMode(IN1, OUTPUT);
pinMode(IN2, OUTPUT);
relay_SetStatus(OFF, OFF); //turn off all the relay
 } //set the status of relays
 void relay_SetStatus( unsigned char status_1, unsigned char status_2)
{
   digitalWrite(IN1, status_1);
   digitalWrite(IN2, status_2);
 }
```

Operation results

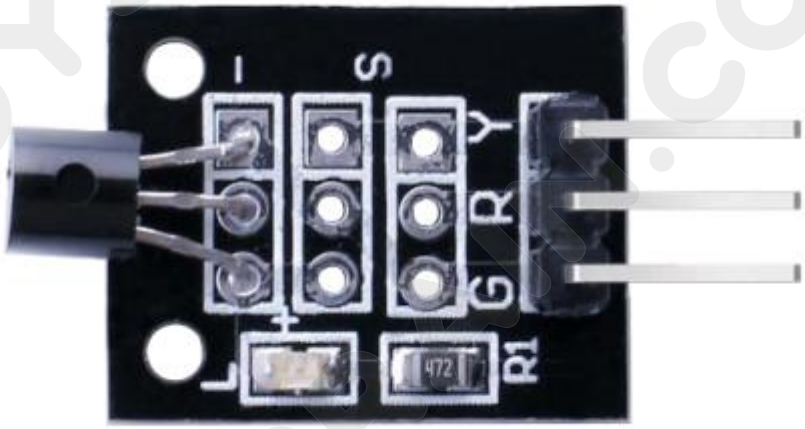
After a few seconds of uploading, you should see the LED cycle between on and off.



Section 16 18b20 Temperature Sensor Module

Introduction:

This module is a temperature sensor with chip DS18B20, unlike other modules NTC-MF523950 temperature sensor (ST1147) or LM35 temperature sensor (SE039).



Specification:

●Chip: DS18B20

● Temperature range: -55 ° C ~ + 125 ° C ● Capacity: +/- 0.5 ° C

●Power supply voltage: 5V DC

Pin configuration:

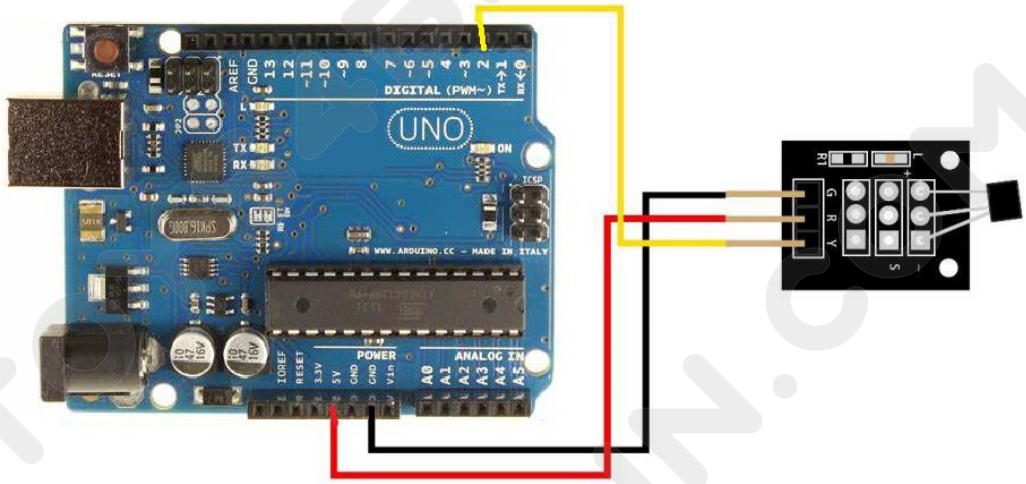
1, "S": analog output pin, real-time output voltage signal

2，“ R”：+ 5V

3，“ G”：GND

Example:

This is a simple code for the DS18B20 temperature module. The wiring is as follows:



Code:

```
// Include the libraries we need
#include <OneWire.h>
#include <DallasTemperature.h>
// Data wire is plugged into port 10 on the Arduino
#define ONE_WIRE_BUS 10
// 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);
/*
* The setup function. We only start the sensors here
2 / 4*/
void setup(void)
{
// start serial port
Serial.begin(9600);
Serial.println("Dallas Temperature IC Control Library Demo");
// Start up the library
sensors.begin();
}
/*
```

```
* Main function, get and show the temperature
*/
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");
// After we got the temperatures, we can print them here.
// We use the function ByIndex, and as an example get the temperature from
the
first sensor only.
Serial.print("Temperature for the device 1 (index 0) is: ");
Serial.println(sensors.getTempCByIndex(0));
}
```

Operation result:

The serial monitor displays the current temperature values as follows: