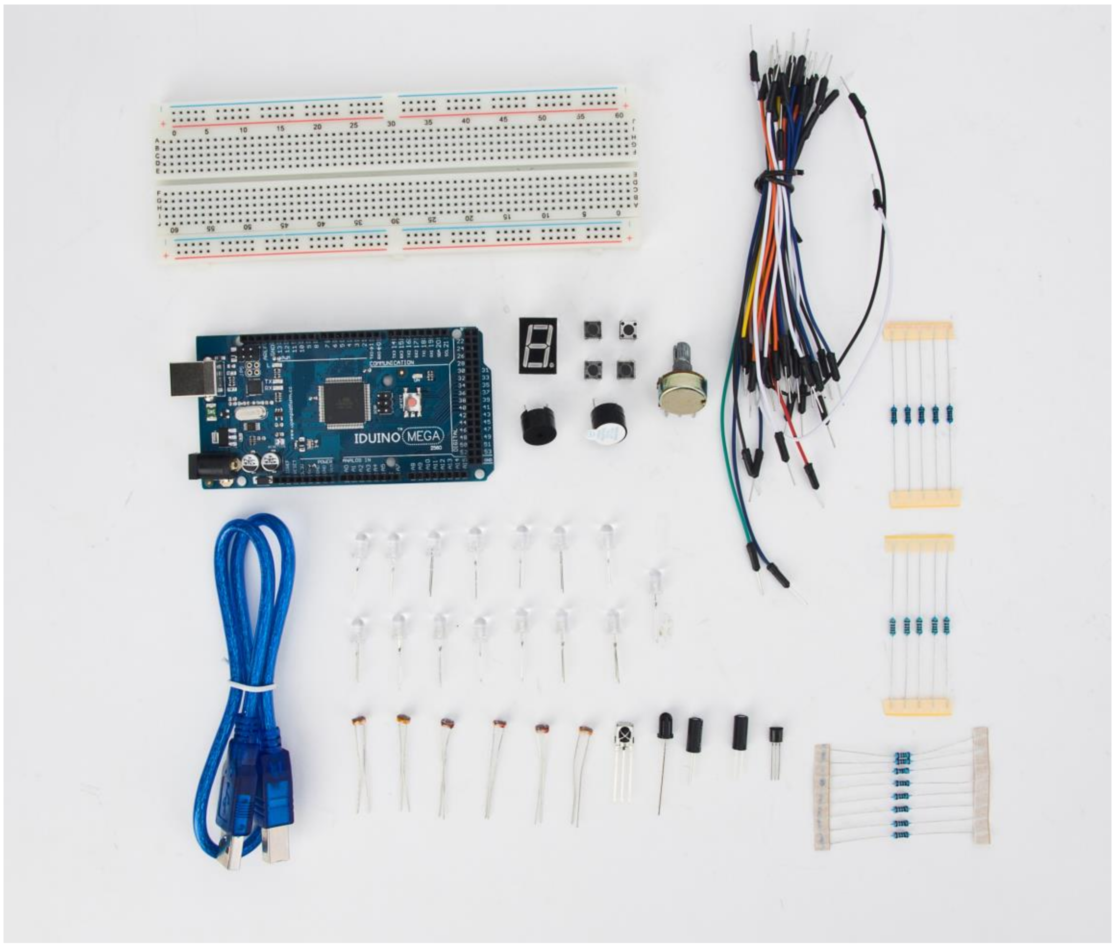
User Manual

For IDUINO Starter Kit(KTS016)



www.openplatform.cc

1. Overview

1.1 what is Arduino?

Arduino is an open-source prototyping platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

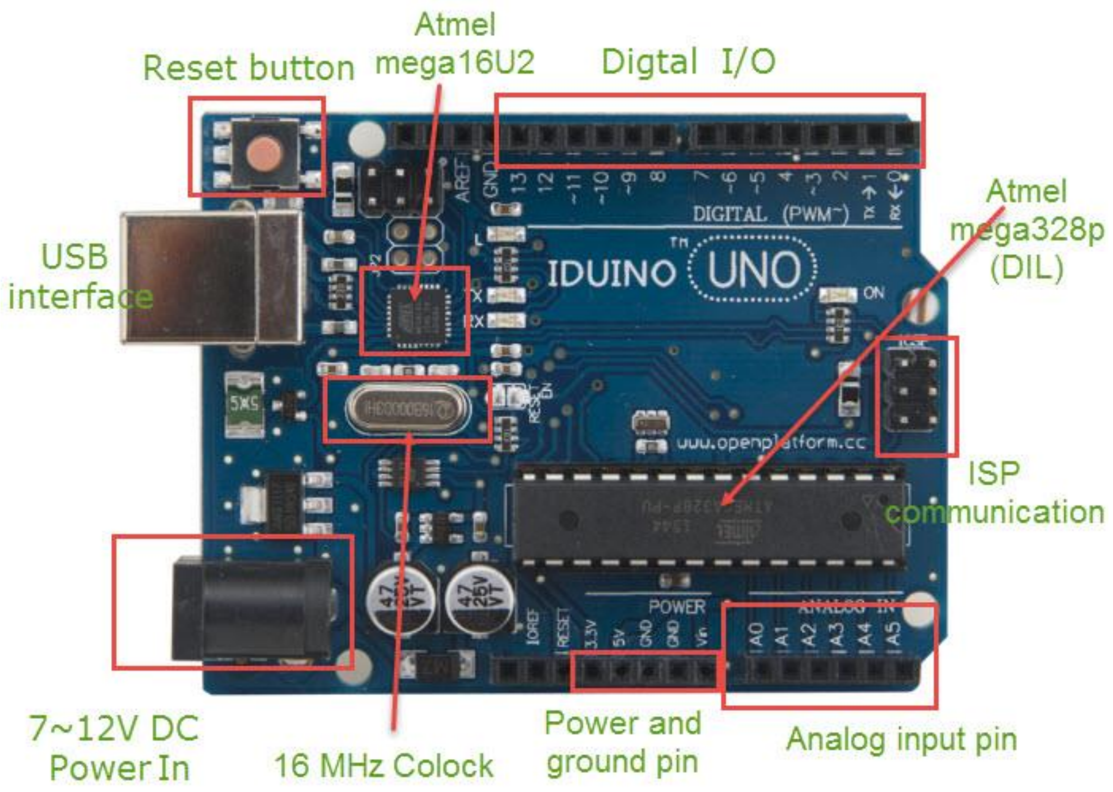
The official website is www.arduino.cc and www.arduino.org.

1.2 what is IDUINO ?

Because of the arduino technology is totally opensource, so anyone can use this facility to create more valuable products.

IDUINO is a series of Ardunio opensource products collection, which includes not only motherboard, but hundreds of sensors and modules used for Arduino board, and many kinds of Arduino Starter Kit, many kinds of Arduino projects, many kinds of car chassis , expansion board, accessories , Arduino DIY 3D Printer.

IDUINO are more focused on manufacturing and constructing Arduino project system.

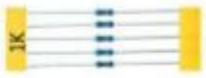


1.3 What's the difference between Arduino and IDUINO?

For the development board, IDUINO is just a different brand comparing with the Arduino development board.

For other categories, IDUINO's quantity exceeds Arduino a lot. IDUINO are more focused on manufacturing and constructing Arduino project system.

2. Content list

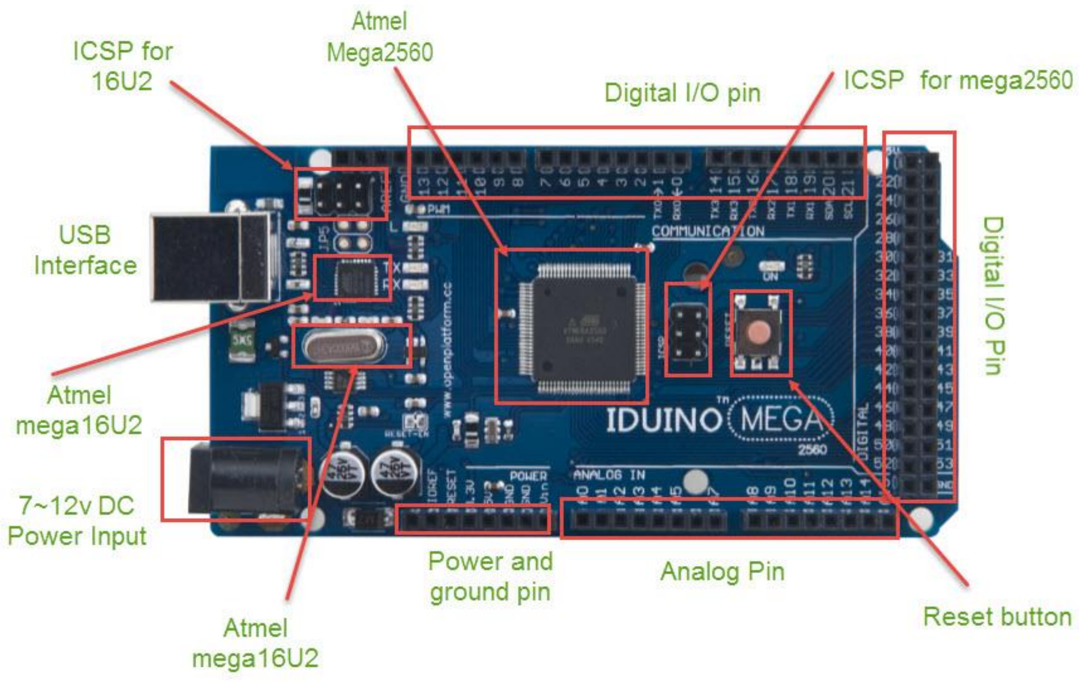
	Name	Quantity	Picture
1	Iduino Mega2560	1	
2	LED with different color	15	
3	220Ω resistor	8	
4	1KΩ resistor	5	
5	10KΩ resistor	5	
6	830 hole breadboard	1	
7	Small 4-pin key switch	4	
8	Active buzzer	1	
9	Passive buzzer	1	
10	Flame sensor	1	
11	LM35 temp sensor	1	
12	Ball tilt switch	2	

13	Photosensitive resistor	3	
14	1-bit 7-seg digital tube	1	
15	Breadboard jumper	30	
16	USB cable	1	

3. IDUINO Mega 2560

The Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

Pinout:

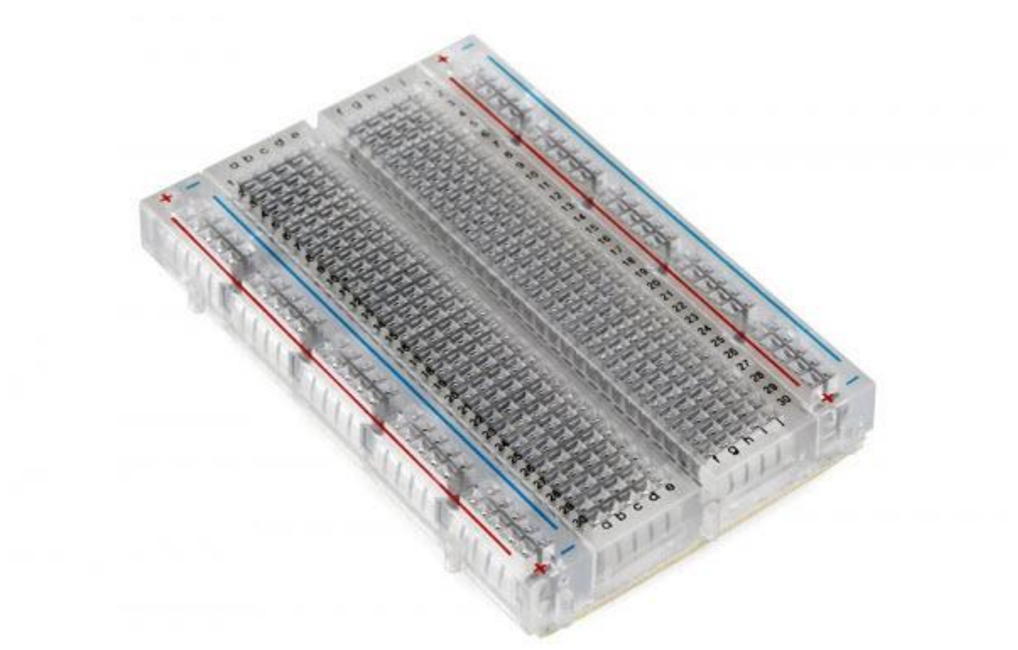


Specifications:

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB

EEPROM	4 KB
Clock Speed	16 MHz
Length	101.52 mm
Width	53.3 mm
Weight	37 g

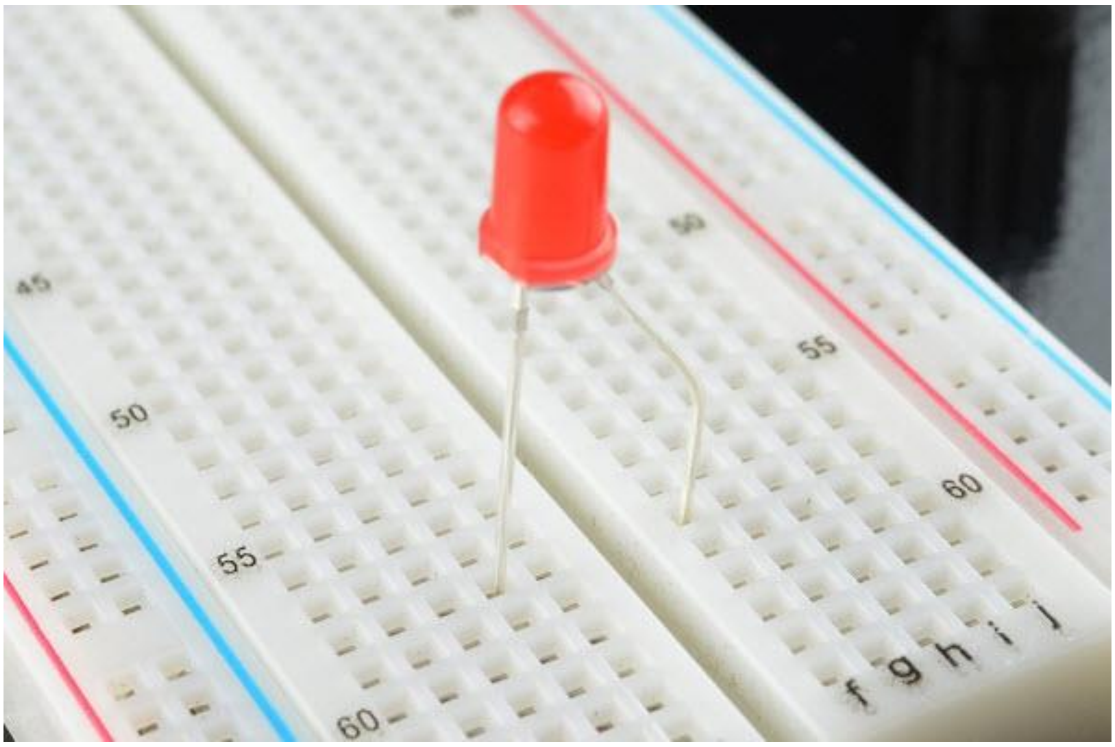
4. chapter 1: Using Breadboard

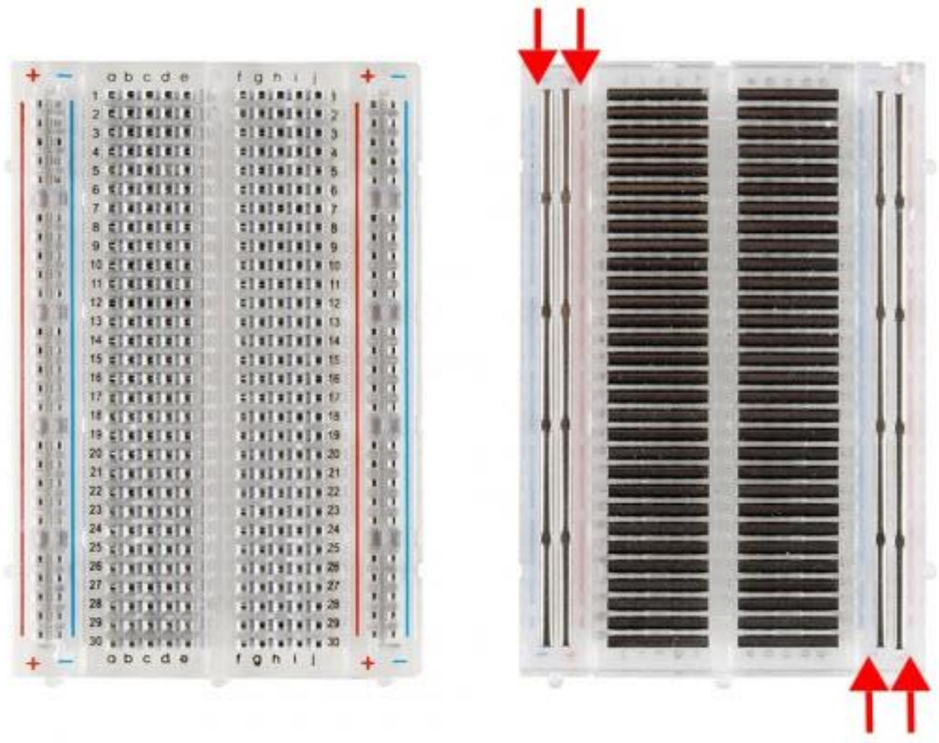


Introduction

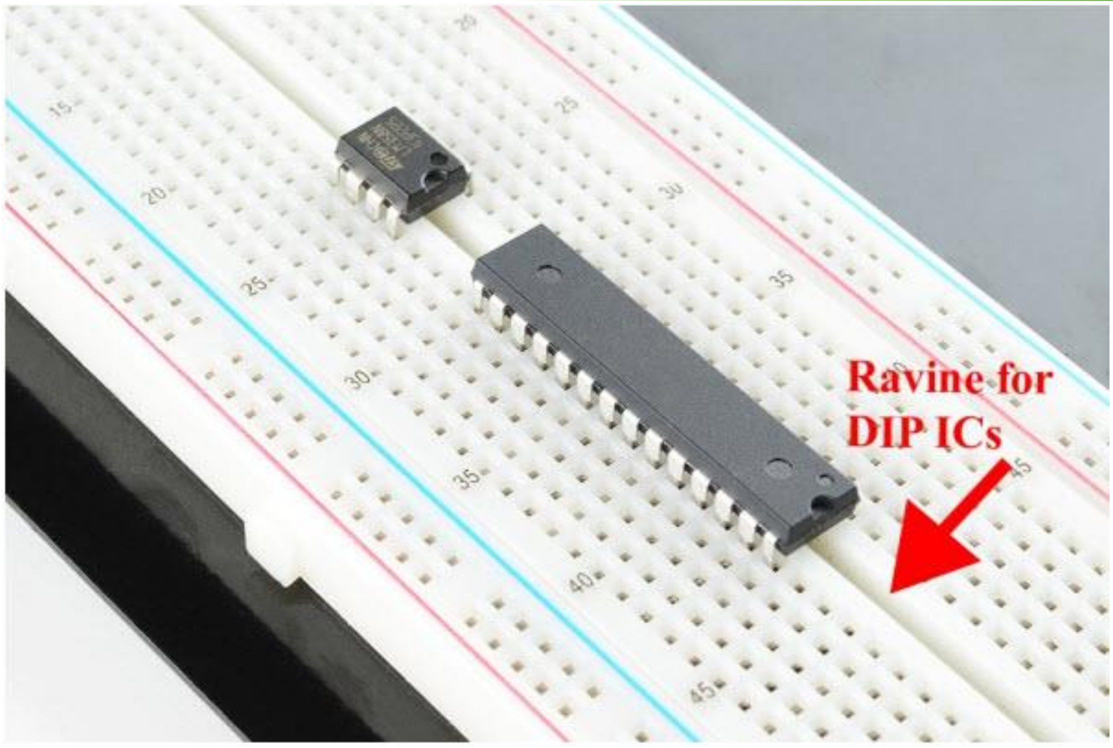
Breadboards are one of the most fundamental pieces when learning how to build circuits. In this tutorial, you will learn a little bit about what breadboards are, why they are called breadboards, and how to use one. Once you are done you should have a basic understanding of how breadboards work and be able to build a basic circuit on a breadboard.

Now that we've seen how the connections in a breadboard are made, let's look at a larger, more typical breadboard. Aside from horizontal rows, breadboards usually have what are called power rails that run vertically along the sides.

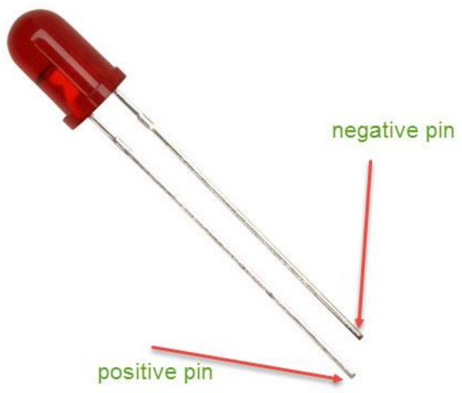




And these DIP chips (salsa anyone?) have legs that come out of both sides and fit perfectly over that ravine. Since each leg on the IC is unique, we don't want both sides to be connected to each other. That is where the separation in the middle of the board comes in handy. Thus, we can connect components to each side of the IC without interfering with the functionality of the leg on the opposite side.



5. chapter 2: LED blinking



Introduction

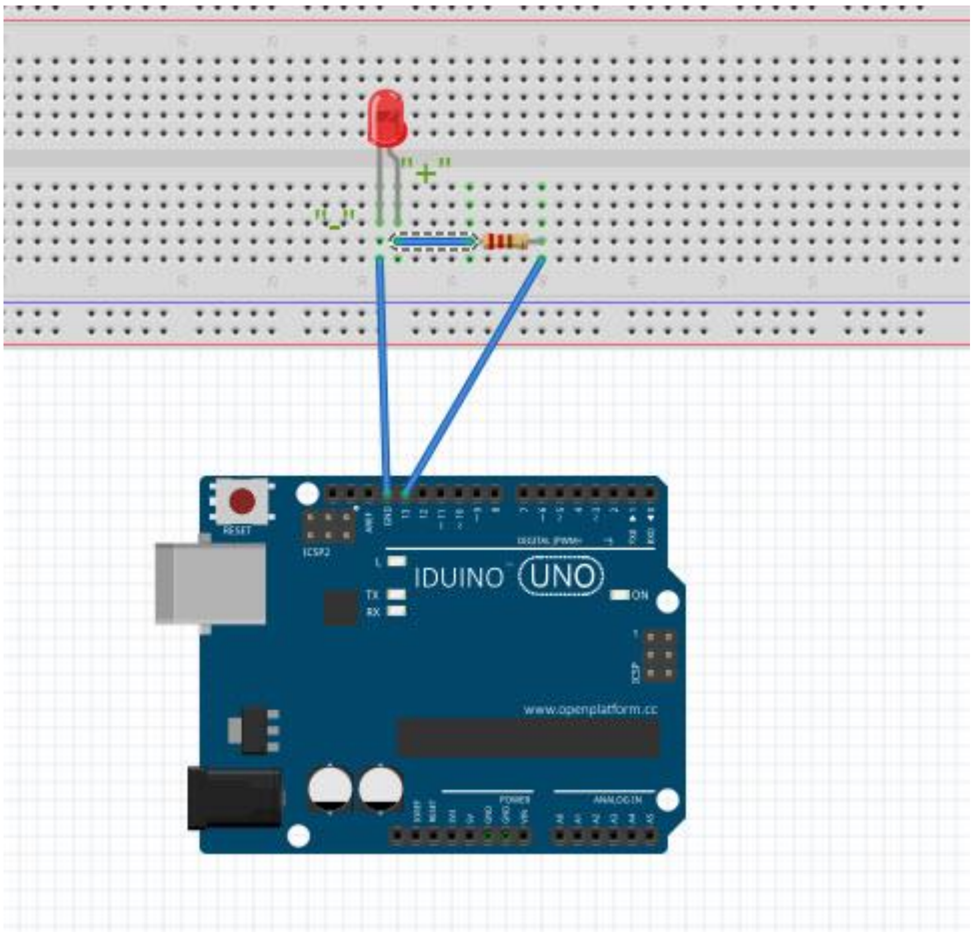
Blinking LED experiment is quite simple. In the "Hello World!" program, we have come across LED. This time, we are going to connect an LED to one of the digital pins rather than using LED13, which is soldered to the board. Except an Arduino and an USB cable, we will need extra parts as below:

Hardware required

1. Red M5 LED*1
2. 220Ω resistor*1
3. Breadboard*1
4. Breadboard jumper wires* several

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

Circuit connection:



Sample program

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

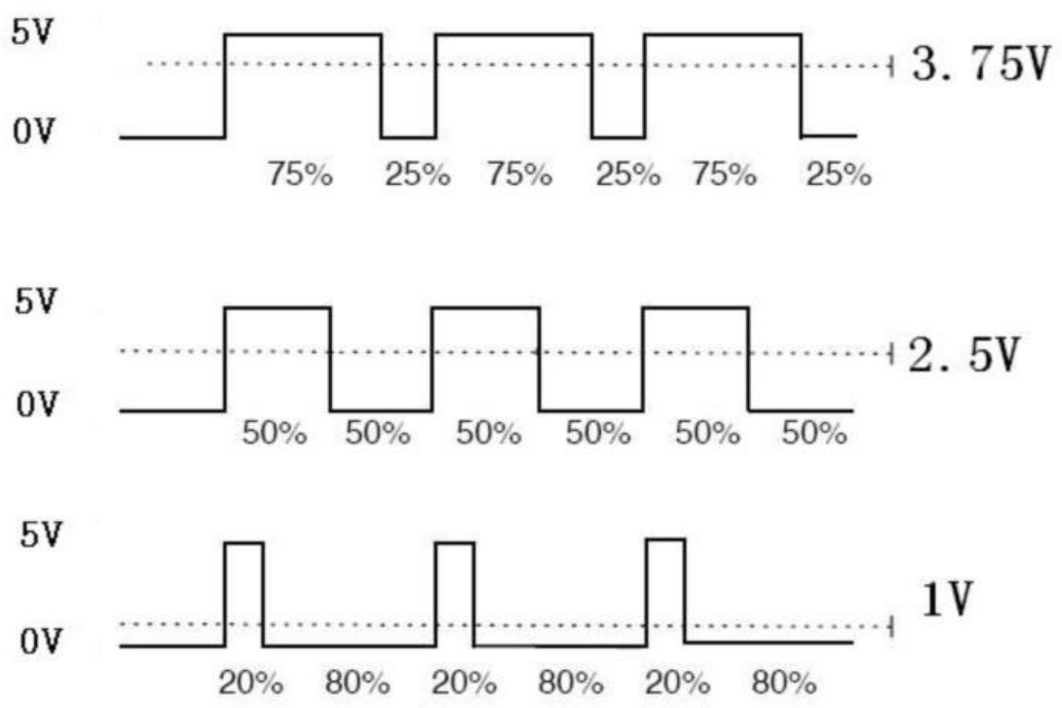
Result

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

6. Chapter3: PWM gradational LED

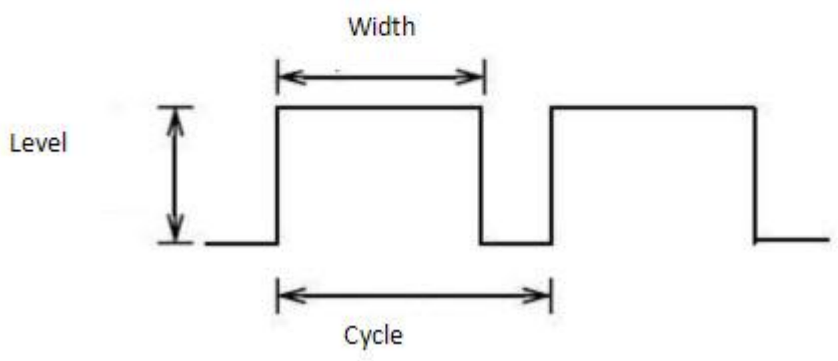
Introduction

PWM, short for Pulse Width Modulation, is a technique used to encode analog signal level into digital ones. A computer cannot output analog voltage but only digital voltage values such as 0V or 5V. So we use a high resolution counter to encode a specific analog signal level by modulating the duty cycle of PMW. The PWM signal is also digitalized because in any given moment, fully on DC power supply is either 5V (ON), or 0V (OFF). The voltage or current is fed to the analog load (the device that uses the power) by repeated pulse sequence being ON or OFF. Being on, the current is fed to the load; being off, it's not. With adequate bandwidth, any analog value can be encoded using PWM. The output voltage value is calculated via the on and off time. Output voltage = (turn on time/pulse time) * maximum voltage value.



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

The following are the three basic parameters of PMW:



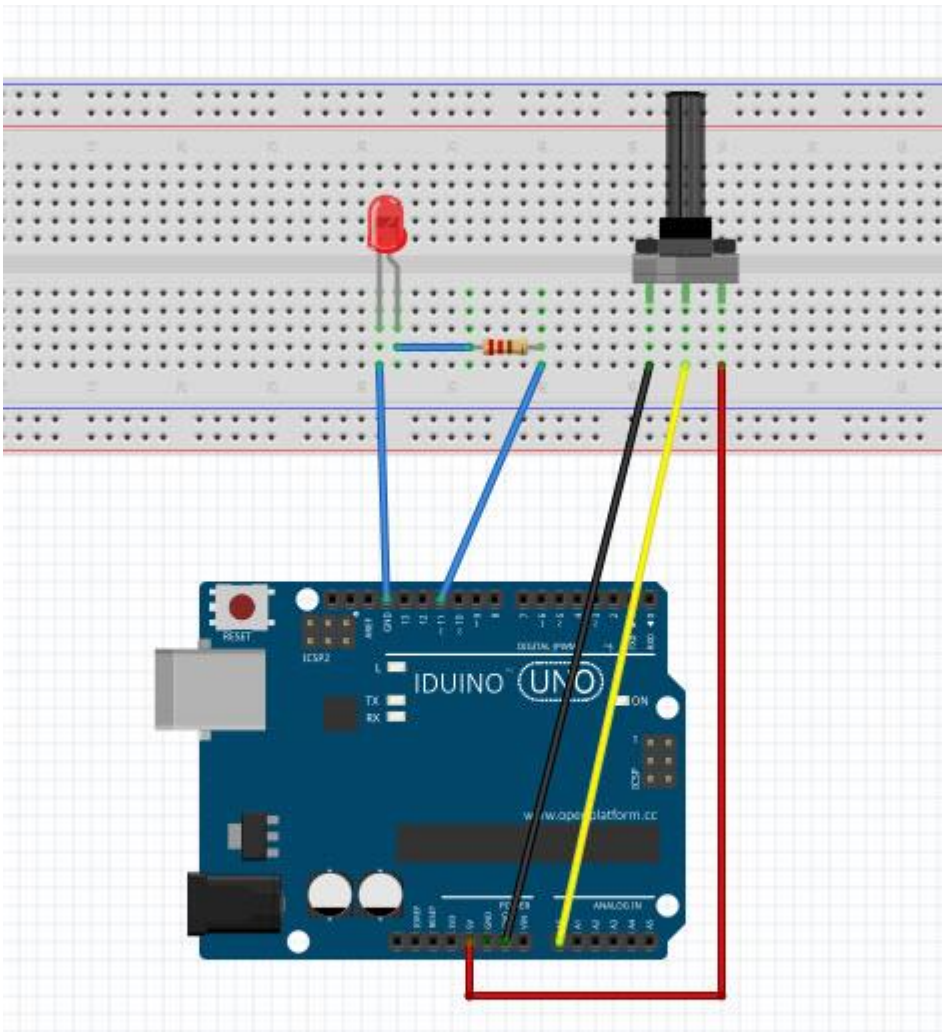
1. The amplitude of pulse width (minimum / maximum)
2. The pulse period (The reciprocal of pulse frequency in 1 second)
3. The voltage level(such as：0V-5V）

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

Hardware required

- Variable resistor*1
- Red M5 LED*1
- 220Ω resistor
- Breadboard*1
- Breadboard jumper wires

Circuit connection



Sample program

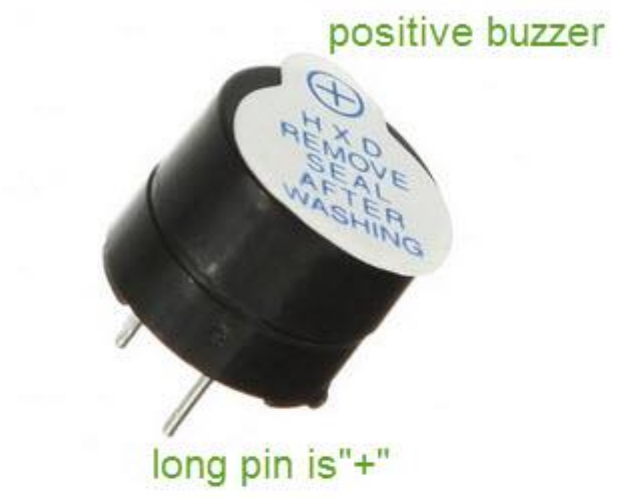
In the program compiling process, we will use the analogWrite (PWM interface, analog value) function. In this experiment, we will read the analog value of the potentiometer and assign the value to PWM port, so there will be corresponding change to the brightness of the LED. One final part will be displaying the analog value on the screen. You can consider this as the "analog value reading" project adding the PWM analog value assigning part. Below is a sample program for your reference.

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

Result

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

7. Chapter4: Active buzzer



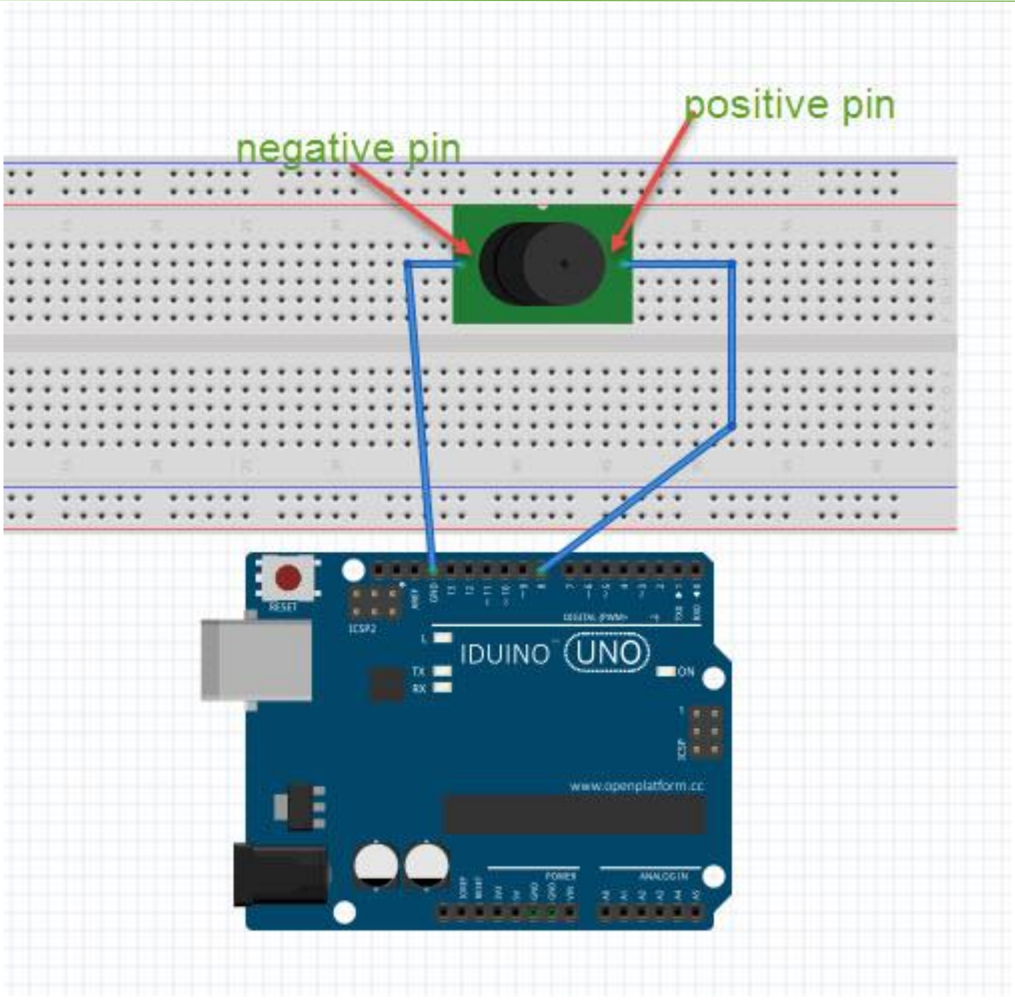
Introduction

Active buzzer is widely used on computer, printer, alarm, electronic toy, telephone, timer etc as a sound making element. It has an inner vibration source. Simply connect it with 5V power supply, it can buzz continuously.

Hardware required

- Buzzer*1
- Key *1
- Breadboard*1
- Breadboard jumper wires

Circuit connection



Example code

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

Result

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

8. Chapter5: Photosensitive Resistor

Introduction

After completing all the previous experiments, we acquired some basic understanding and knowledge about Arduino application. We have learned digital input and output, analog input and PWM. Now, we can begin the learning of sensors applications.
Photo resistor (Photovaristor) is a resistor whose resistance varies according to different incident light strength. It's made based on the photoelectric effect of semiconductor. If the incident light is intense, its resistance reduces; if the incident light is weak, the resistance increases. Photovaristor is commonly applied in the measurement of light, light control and photovoltaic conversion (convert the change of light into the change of electricity).

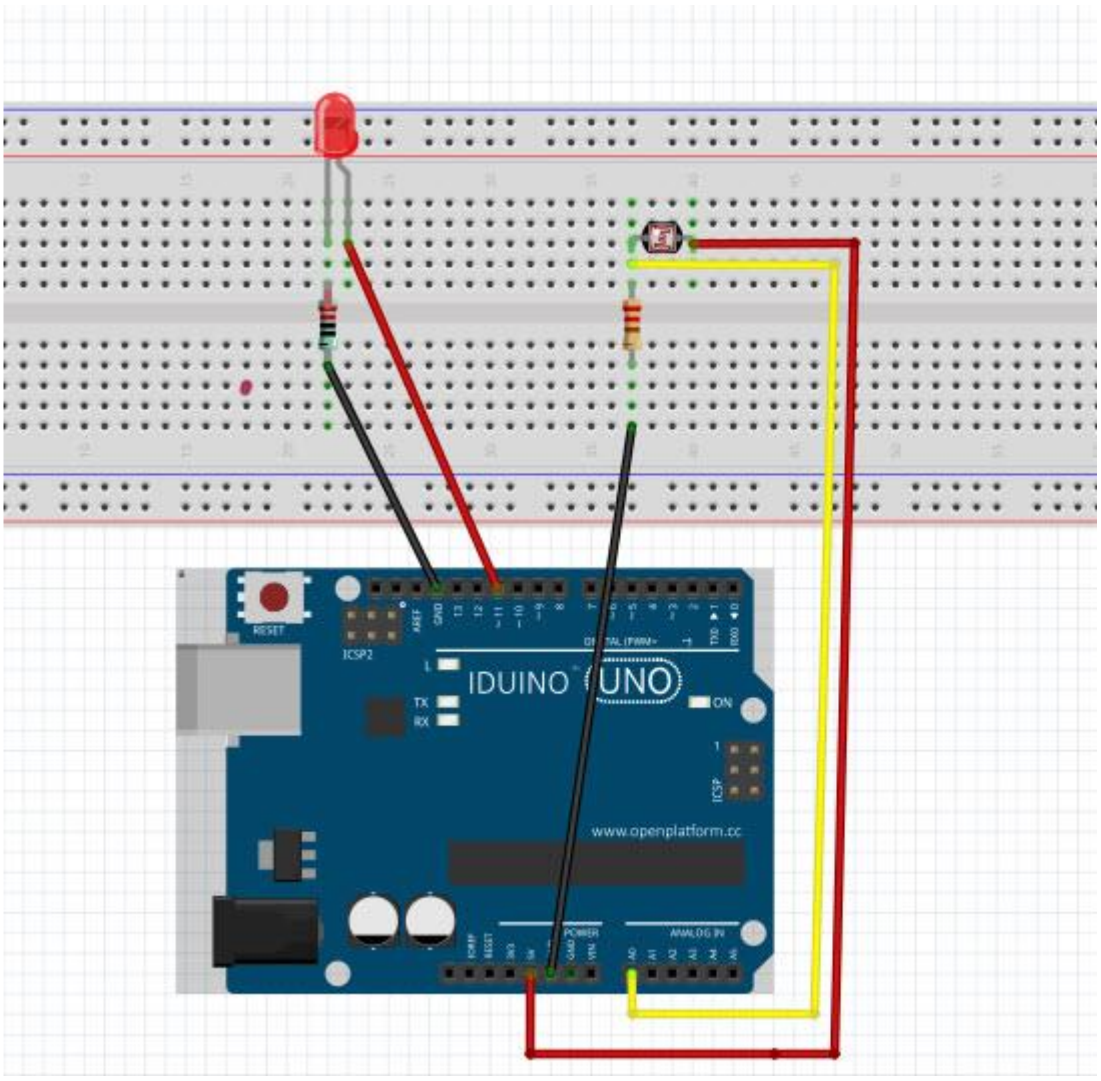
Photo resistor is also being widely applied to various light control circuit, such as light control and adjustment, optical switches etc.

We will start with a relatively simple experiment regarding photovaristor application. Photovaristor is an element that changes its resistance as light strenth changes. So we will need to read the analog values. We can refer to the PWM experiment, replacing the potentiometer with photovaristor. When there is change in light strength, there will be corresponding change on the LED.

Hardware required

- Photo resistor*1
- Red M5 LED*1
- 10KΩresistor*1
- 220Ωresistor*1
- Bread board*1
- Bread board jumper wires

Circuit connection



Example code

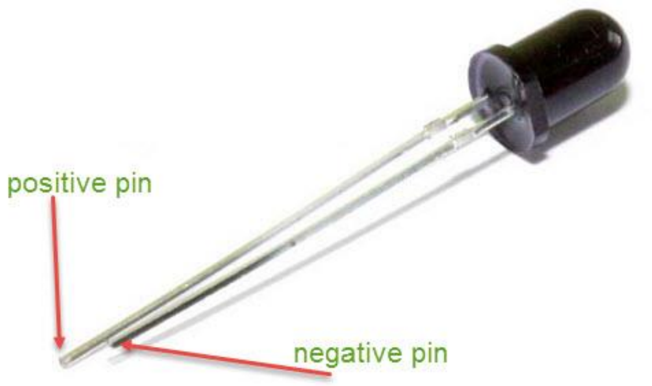
*******code begin*******

```
/////////////////////////////////////////////////////////////
int potpin=0;// initialize analog pin 0, connected with photovaristor
int ledpin=11;// initialize digital pin 11, output regulating the brightness of LED
int val=0;// initialize variable va
void setup()
{
pinMode(ledpin,OUTPUT);// set digital pin 11 as "output"
Serial.begin(9600);// set baud rate at "9600"
}
void loop()
{
val=analogRead(potpin);// read the analog value of the sensor and assign it to val
Serial.println(val);// display the value of val
analogWrite(ledpin,val);// turn on the LED and set up brightness（maximum output
value 255）
delay(10);// wait for 0.01
}
/////////////////////////////////////////////////////////////
*******code End*******
```

Result

After downloading the program, you can change the light strength around the photovaristor and see corresponding brightness change of the LED. Photovaristors has various applications in our everyday life. You can make other interesting interactive projects base on this one.

9. Chapter6: Flame sensor



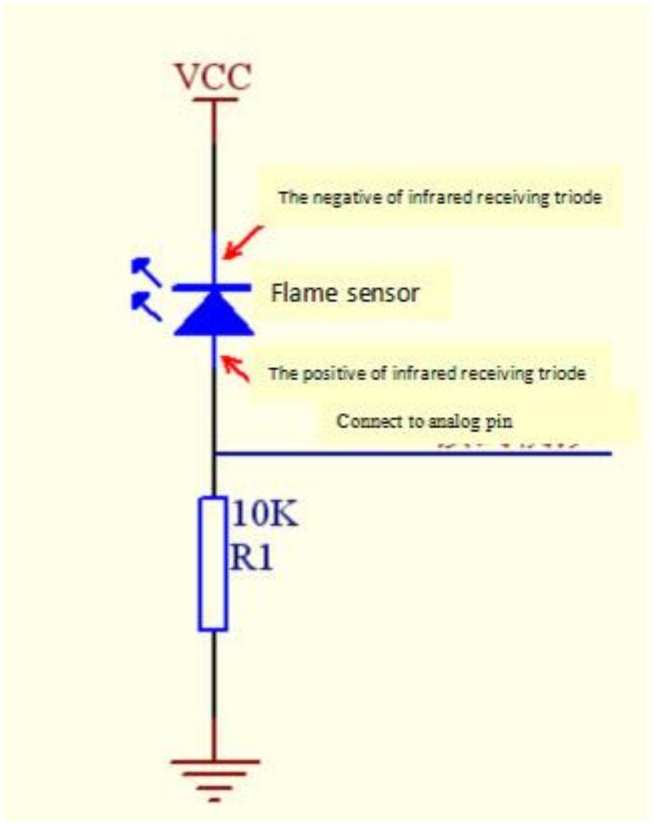
Introduction

Flame sensor (Infrared receiving triode) is specially used on robots to find the fire source. This sensor is of high sensitivity to flame. Below is a photo of it.

Working principle:

Flame sensor is made based on the principle that infrared ray is highly sensitive to flame. It has a specially designed infrared receiving tube to detect fire, and then convert the flame brightness to fluctuating level signal. The signals are then input into the central processor and be dealt with accordingly.
Sensor connection

The shorter lead of the receiving triode is for negative, the other one for positive. Connect negative to 5V pin, positive to resistor; connect the other end of the resistor to GND, connect one end of a jumper wire to a clip which is electrically connected to sensor positive, the other end to analog pin. As shown below:


VCC
The negative of infrared receiving triode
Flame sensor
The positive of infrared receiving triode
Connect to analog pin
10K
R1

Hardware required

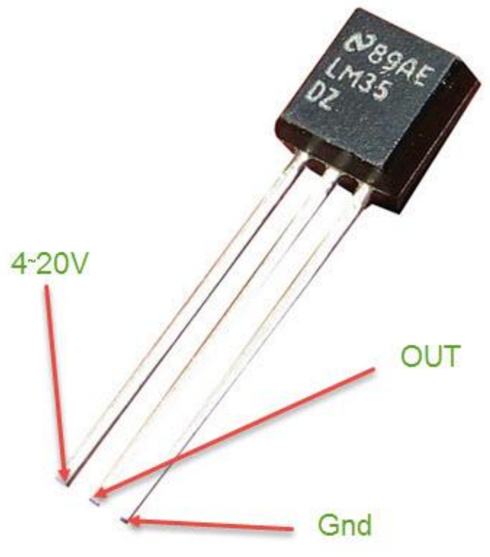
- Flame sensor *1
- Buzzer *1

- 10K resistor *1
- Breadboard jumper wires

Example Code

```
*******code End*******
int flame=0;// select analog pin 0 for the sensor
int Beep=9;// select digital pin 9 for the buzzer
int val=0;// initialize variable
void setup()
{
  pinMode(Beep,OUTPUT);// set LED pin as “output”
pinMode(flame,INPUT);// set buzzer pin as “input”
Serial.begin(9600);// set baud rate at “9600”
}
void loop()
{
  val=analogRead(flame);// read the analog value of the sensor
  Serial.println(val);// output and display the analog value
  if(val>=600)// when the analog value is larger than 600, the buzzer will buzz
  {
    digitalWrite(Beep,HIGH);
    }else
    {
        digitalWrite(Beep,LOW);
      }
    delay(500);
}
*******code End*******
```

10. Chapter7: LM35 Temperature Sensor



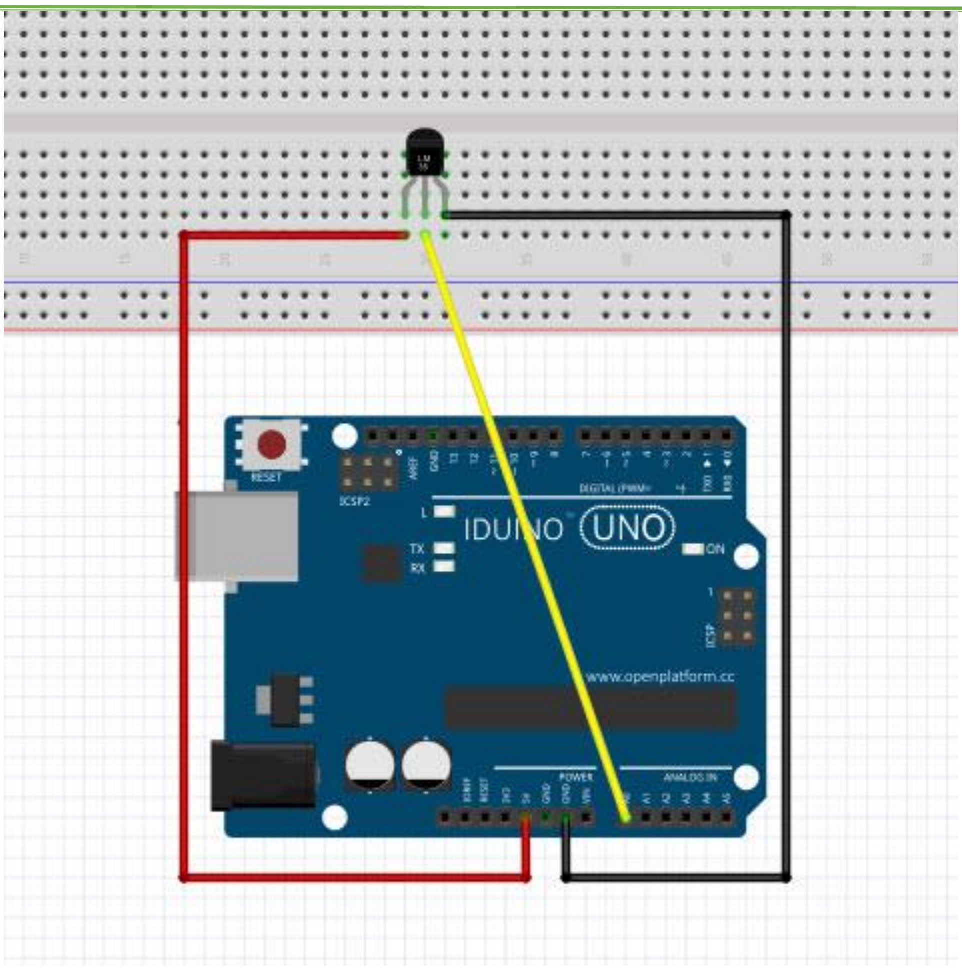
Introduction

LM35 is a common and easy-to-use temperature sensor. It does not require other hardware. You just need an analog port to make it work. The difficulty lies in compiling the code to convert the analog value it reads to celsius temperature.

Hardware required

- LM35*1
- Breadboard*1
- Breadboard jumper wires

Circuit connection



Example code

```
*******code begin*******
int potPin = 0; // initialize analog pin 0 for LM35 temperature sensor
void setup()
{
Serial.begin(9600);// set baud rate at"9600"
}
void loop()
{
int val;// define variable
int dat;// define variable
val=analogRead(0);// read the analog value of the sensor and assign it to val
dat=(125*val)>>8;// temperature calculation formula
Serial.print("Tep:");// output and display characters beginning with Tep
Serial.print(dat);// output and display value of dat
Serial.println("C");// display "C" characters
delay(500);// wait for 0.5 second
```

}
*******code End*******

Result

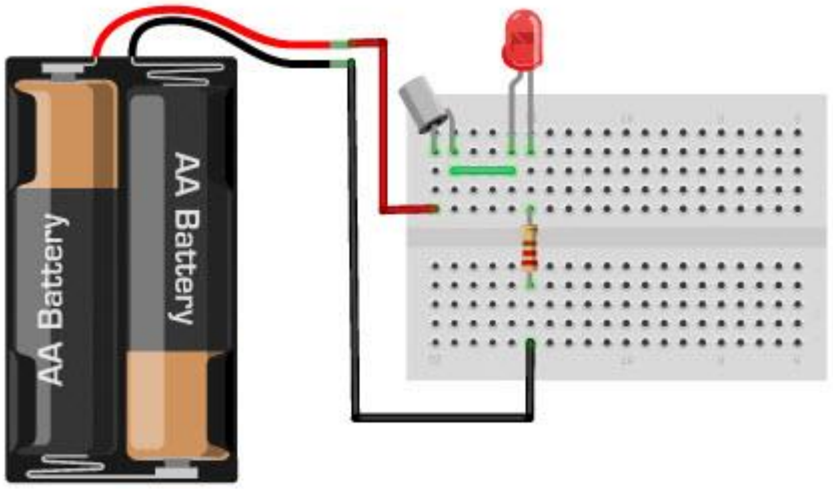
After downloading the program, you can open the monitoring window to see current temperature.

11. Chapter8: Tilt sensor switch

Tilt sensors allow you to detect orientation or inclination. They are small, inexpensive, low-power and easy-to-use. If used properly, they will not wear out. Their simplicitiy makes them popular for toys, gadgets and appliances. Sometimes they are referred to as "mercury switches", "tilt switches" or "rolling ball sensors" for obvious reasons.

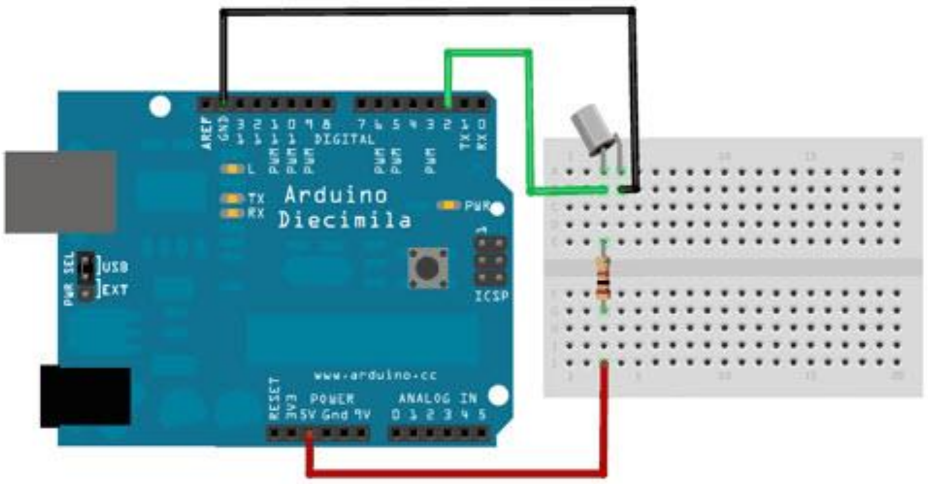
Simple Tilt-Activated LED

This is the most basic way of connecting to a tilt switch, but can be handy while one is learning about them. Simply connect it in series with an LED, resistor and battery. Tilt to turn on and off.



Reading Switch State with a Microcontroller

Note that the layout above shows a 10K pullup resistor but for the code I use the 'built-in' pullup resistor that you can turn on by setting an input pin to HIGH output (its quite neat!) If you use the internal pull-up you can skip the external one.

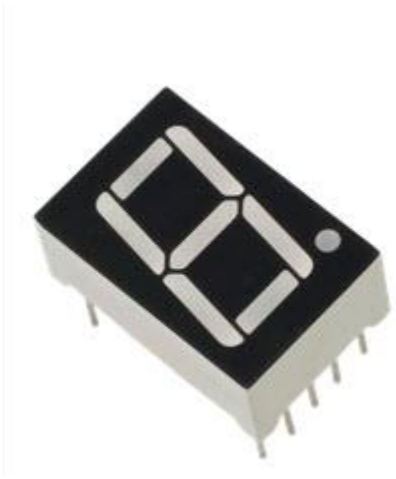


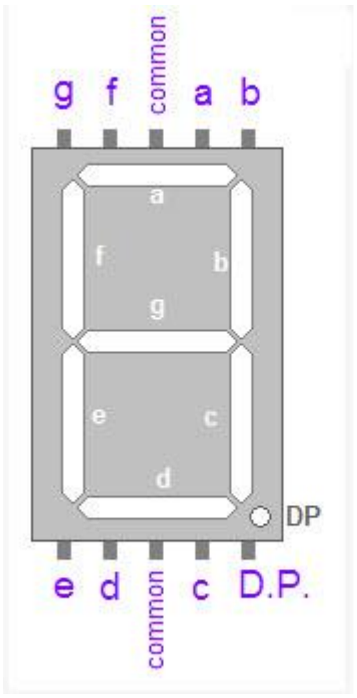
Example code

*******code begin*******

```
int inPin = 2;             // the number of the input pin
int outPin = 13;            // the number of the output pin

int LEDstate = HIGH;          // the current state of the output pin
int reading;                    // the current reading from the input pin
int previous = LOW;        // the previous reading from the input pin

// the follow variables are long's because the time, measured in miliseconds,
// will quickly become a bigger number than can be stored in an int.
long time = 0;                 // the last time the output pin was toggled
long debounce = 50;      // the debounce time, increase if the output flickers

void setup()
{
  pinMode(inPin, INPUT);
  digitalWrite(inPin, HIGH);     // turn on the built in pull-up resistor
  pinMode(outPin, OUTPUT);
}

void loop()
{
  int switchstate;

  reading = digitalRead(inPin);

  // If the switch changed, due to bounce or pressing...
  if (reading != previous) {
    // reset the debouncing timer
    time = millis();
  }

  if ((millis() - time) > debounce) {
     // whatever the switch is at, its been there for a long time
     // so lets settle on it!
     switchstate = reading;

     // Now invert the output on the pin13 LED
    if (switchstate == HIGH)
      LEDstate = LOW;
    else
      LEDstate = HIGH;
  }
  digitalWrite(outPin, LEDstate);

  // Save the last reading so we keep a running tally
  previous = reading;
}
*******code End*******
```

12. Chapter9: 1-bit digit 7-segment display



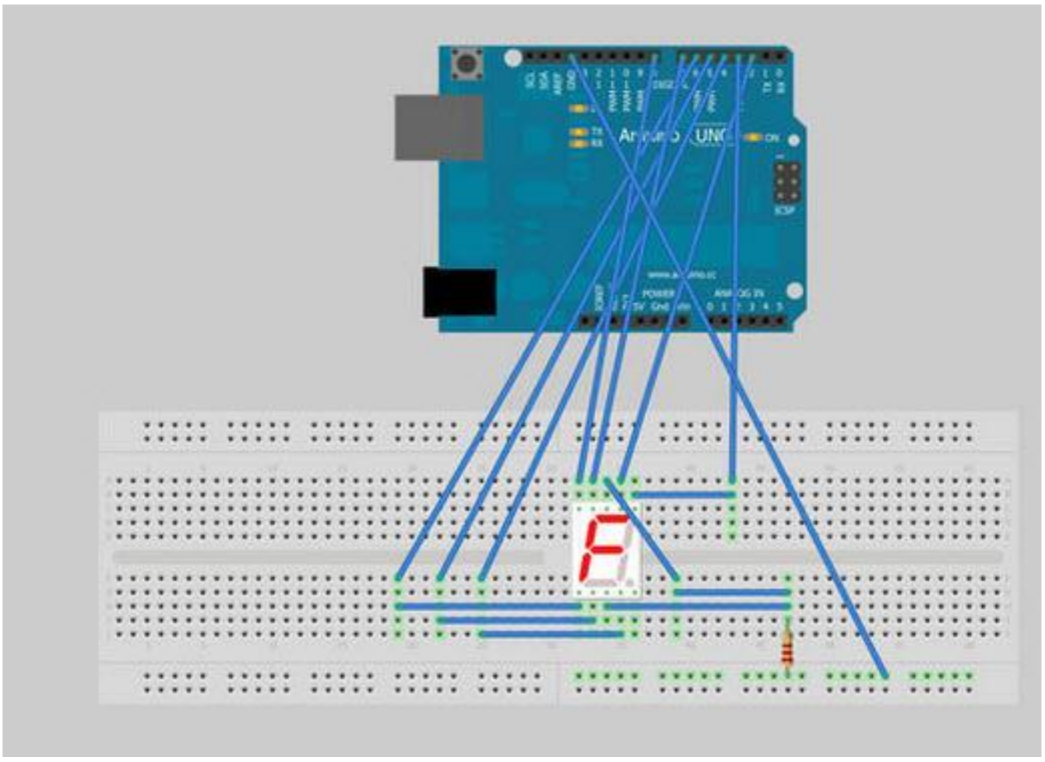


Introduction

LED segment displays are common for displaying numerical information. It's widely applied on displays of electromagnetic oven, full automatic washing machine, water temperature display, electronic clock etc. It is necessary that we learn how it works. LED segment display is a semiconductor light-emitting device. Its basic unit is a light-emitting diode (LED). LED segment display can be divided into 7-segment display and 8-segment display according to the number of segments. 8-segment display has one more LED unit (for decimal point display) than 7-segment one. In this experiment, we use a 8-segment display. According to the wiring method of LED units, LED segment displays can be divided into display with common anode and display with common cathode. Common anode display refers to the one that combine all the anodes of LED units into one common anode (COM).

For the common anode display, connect the common anode (COM) to +5V. When the cathode level of a certain segment is low, the segment is on; when the cathode level of a certain segment is high, the segment is off. For the common cathode display, connect the common cathode (COM) to GND. When the anode level of a certain segment is high, the segment is on; when the anode level of a certain segment is low, the segment is off.

Circuit connection



Example code

*******code begin*******
```
// Define the LED digit patters, from 0 - 9
// Note that these patterns are for common cathode displays
// For common anode displays, change the 1's to 0's and 0's to 1's
// 1 = LED on, 0 = LED off, in this order:
//                                              Arduino pin: 2,3,4,5,6,7,8
byte seven_seg_digits[10][7] = { { 1,1,1,1,1,1,0 },    // = 0

{ 0,1,1,0,0,0,0 },    // = 1

{ 1,1,0,1,1,0,1 },    // = 2

{ 1,1,1,1,0,0,1 },    // = 3

{ 0,1,1,0,0,1,1 },    // = 4

{ 1,0,1,1,0,1,1 },    // = 5
{ 1,0,1,1,1,1,1 },    // = 6

{ 1,1,1,0,0,0,0 },    // = 7

{ 1,1,1,1,1,1,1 },    // = 8

{ 1,1,1,0,0,1,1 }      // = 9
                                                                          };

void setup() {
    pinMode(2, OUTPUT);
    pinMode(3, OUTPUT);
    pinMode(4, OUTPUT);
    pinMode(5, OUTPUT);
    pinMode(6, OUTPUT);
    pinMode(7, OUTPUT);
    pinMode(8, OUTPUT);
    pinMode(9, OUTPUT);
    writeDot(0);    // start with the "dot" off
}

void writeDot(byte dot) {
    digitalWrite(9, dot);
}
void sevenSegWrite(byte digit) {
    byte pin = 2;
    for (byte segCount = 0; segCount < 7; ++segCount) {
        digitalWrite(pin, seven_seg_digits[digit][segCount]);
        ++pin;
    }
}

void loop() {
    for (byte count = 10; count > 0; --count) {
      delay(1000);
      sevenSegWrite(count - 1);
    }
    delay(4000);
}
```

*******code End*******