Basic Learning Kit for Arduino

CONTENT

1. Product Introduction

This is an Arduino basic learning kit, equipped with a lot of electronic components, test code and 21 lessons of interactive tutorials. Let you master the basic knowledge of Arduino, and to get the ability to develop SCM system when playing. It doesn't matter even if you do not have a basic knowledge to write the program, this basic learning kit provides you with Arduino programming learning data to help you from zero.

2. Component List

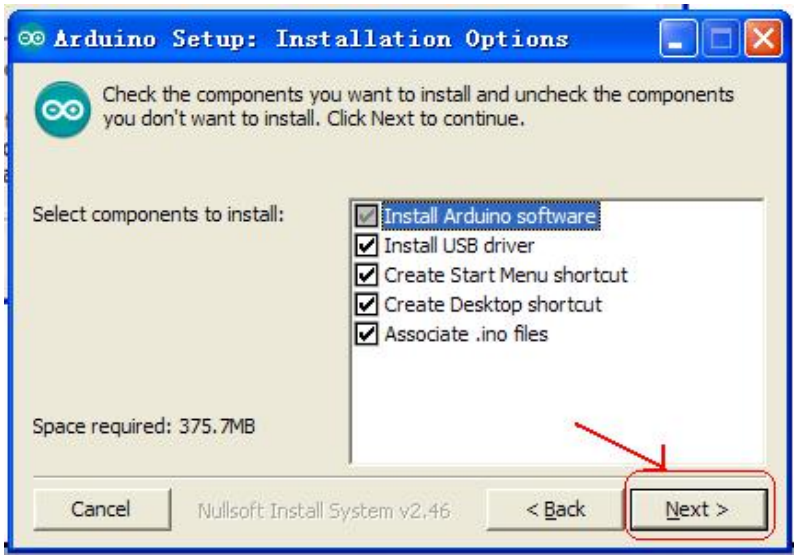
Product Name	Specification & Model	Quantity
LED	F5-White to blue-short end	5
LED	F5-White to red-short end	5
LED	F5-White to yellow-short end	5
Resistor	Carbon film color ring 1/4W 1% 220Ω	8
Photo-resistor	5516 bright resistor 5-10KΩ dark resistor 0.2MΩ	3
Resistor	Carbon film color ring 1/4W 1% 1K braid	5
Resistor	Carbon film color ring 1/4W 1% 10K braid	5
Adjustable Potentiometer	16MM single linkage B50K	1
IC	74HC595 DIP	1
Touch Button	12*12*7.3MM Plugin	4
Sensor	LM35DZ	1
Sensor	IR Receiver 5MM (flame)	1
Sensor	IR Receiver VS1838B	1
Ball Switch	HDX-2801 same pin	2
Dot Matrix	3.7MM 8*8, 16 pins	1
Passive Buzzer	12*8.5MM 5V Ordinary Fission 2K	1
Active Buzzer	12*9.5MM 5V Ordinary Fission 2300Hz	1
LED Segment Display	Four-bit, 0.56 inch, COM Cathode	1
LED Segment Display	One-bit, 0.56 inch, COM Cathode	2
Breadboard	ZY-102 830 holes (no packaging)	1
Remote Controller	KLP-2, 21Keys, 86*40*6.5MM , White	1
Battery Case	6-cell with 15CM wire	1
Sensor	RGB LED plugin（black）	1
Development Board	UNO R3	1
Breadboard Wire	30 breadboard connection wires	1
USB Cable	AM/BM transparent blue OD:5.0 L=50cm	1

3. Arduino IDE and Driver Installation

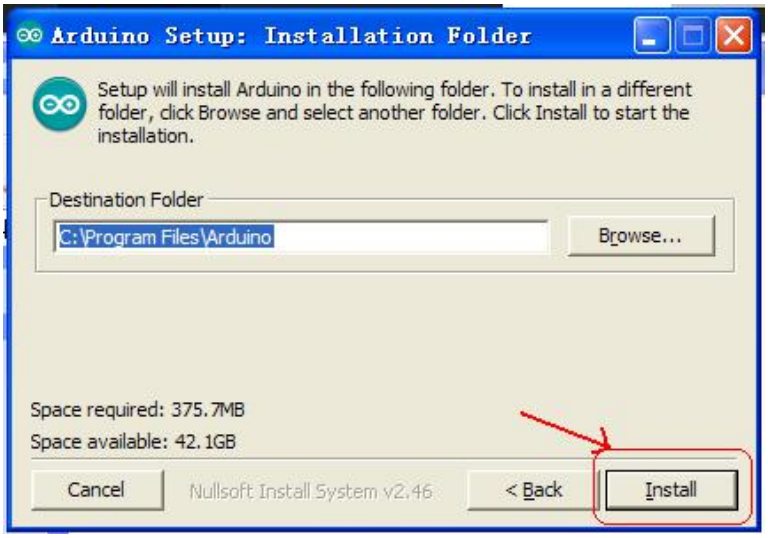
When getting the Arduino development board, first of all you have to install the Arduino IDE and the driver, and all relevant files can be found on the official website. The following link includes various systems, various versions of the Arduino IDE and drivers whatever you choose.
https://www.arduino.cc/en/Main/OldSoftwareReleases#1.5.x

Here we first introduce the installation method of Arduino IDE-1.5.6 version in the Windows system.The file downloaded is an arduino-1.5.6-r2-windows.zip compression folder, please unzip it to the hard disk. Double-click Arduino-1.5.6 .exe file.
Please refer to the following setup figures:

Click “I Agree”

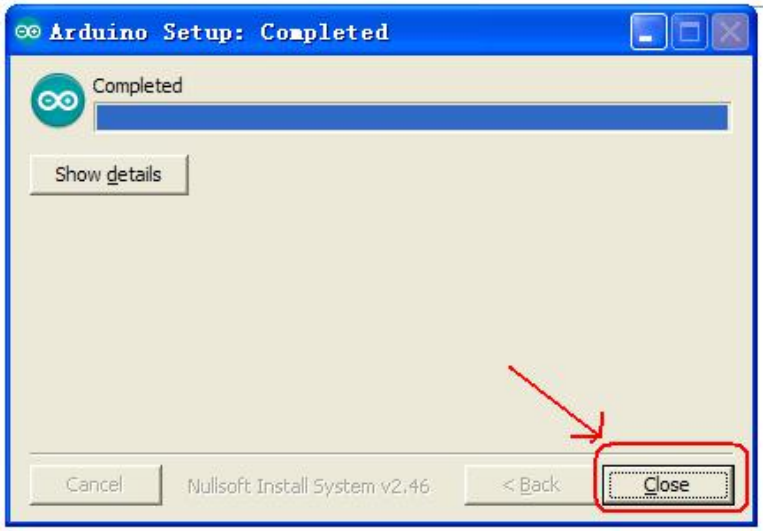


Then, click “Next”

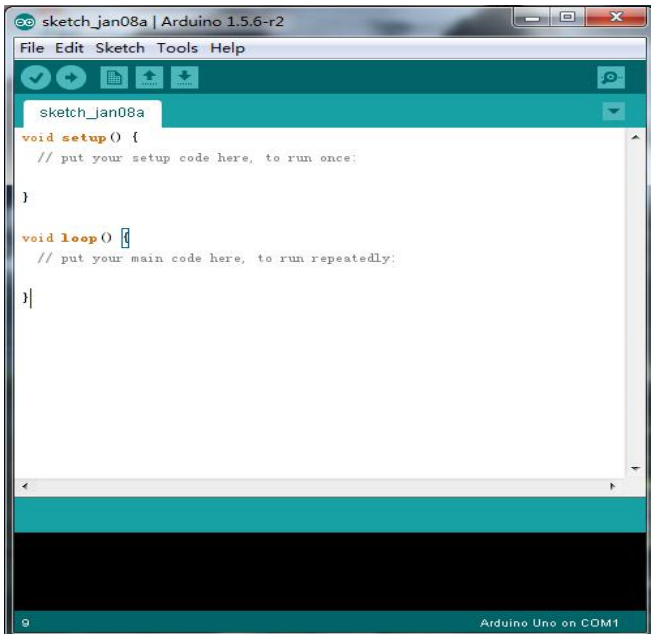


Next, click “Install”.

Click “Close” after completing the installation.



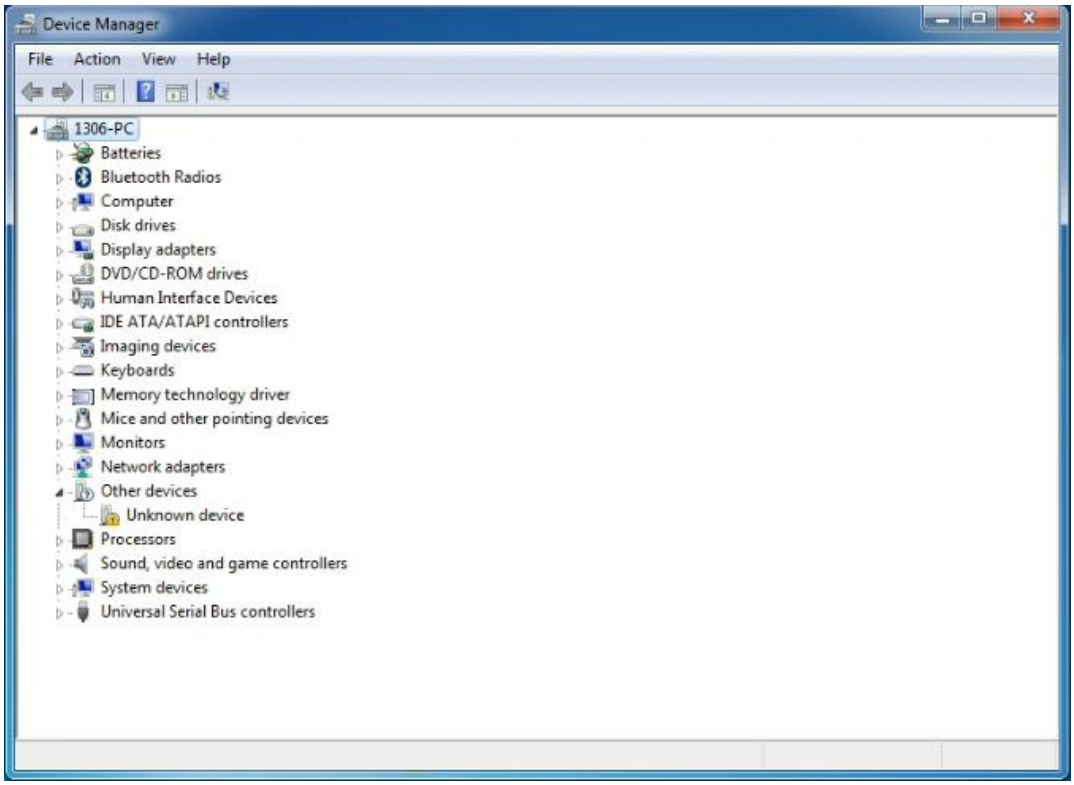
The figure below shows the successful installation of 1.5.6 version:



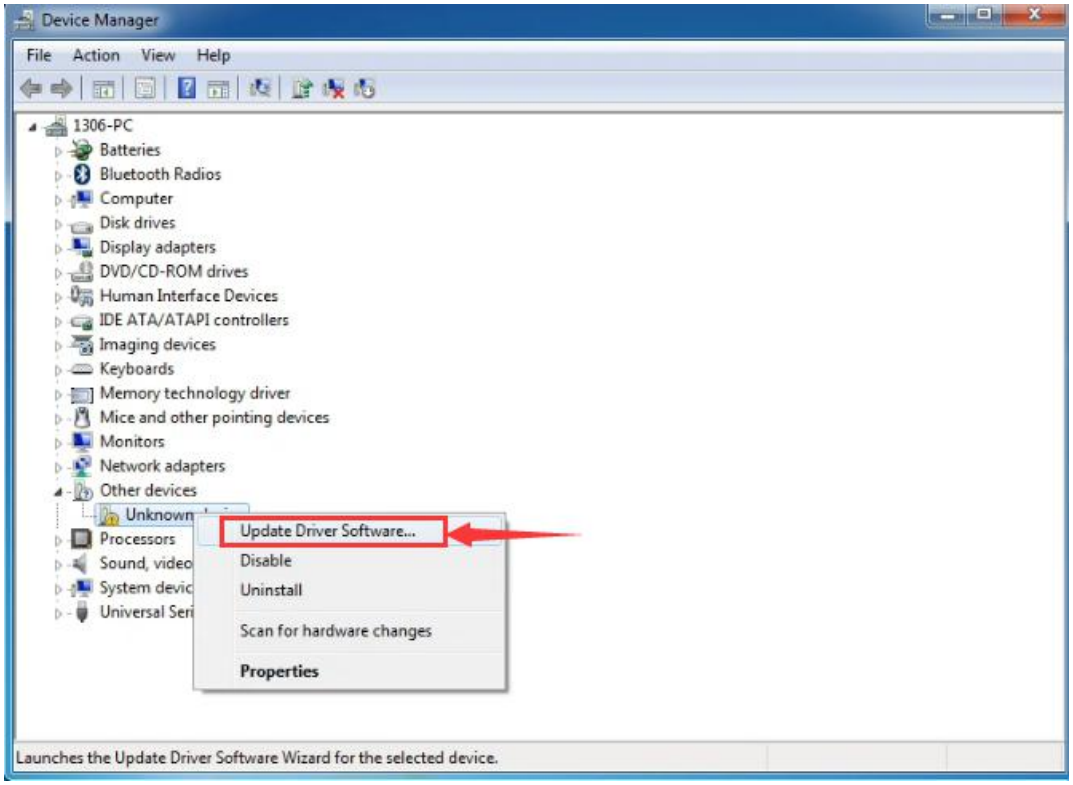
Next, we will introduce the Arduino board driver installation.

Due to different systems, the driver installation method also has some slight differences. Let's move on to the driver installation method in the WIN 7 system.

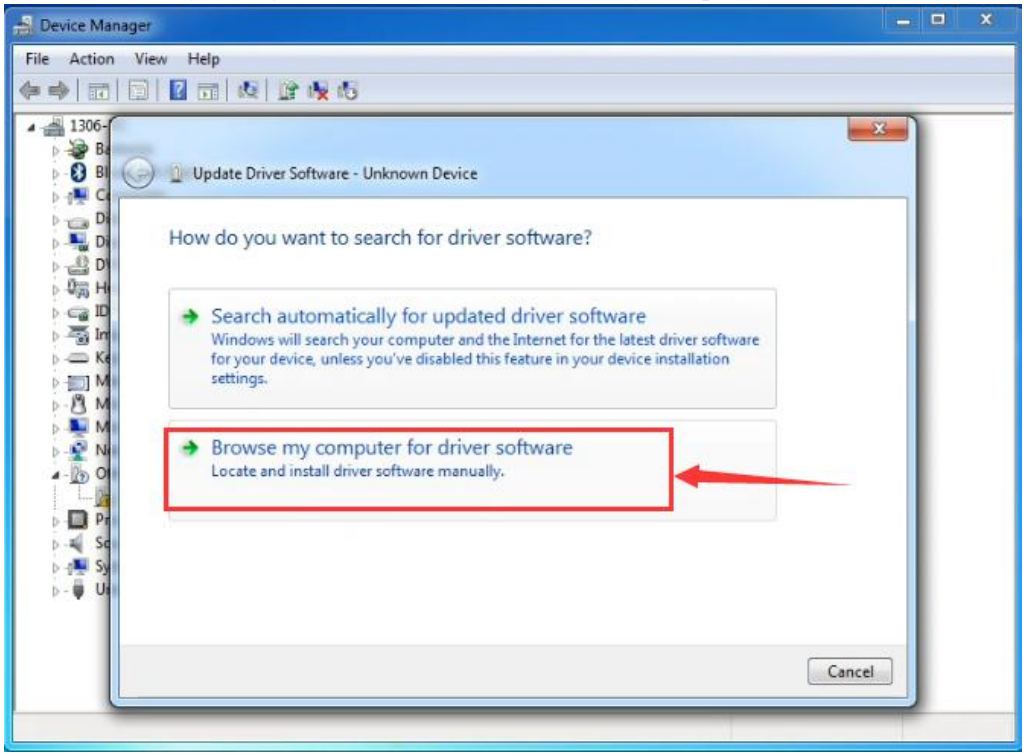
When Arduino uno board is connected to the computer at the first time, right click on the Computer - Properties - Device Manager, as the figure shown below：



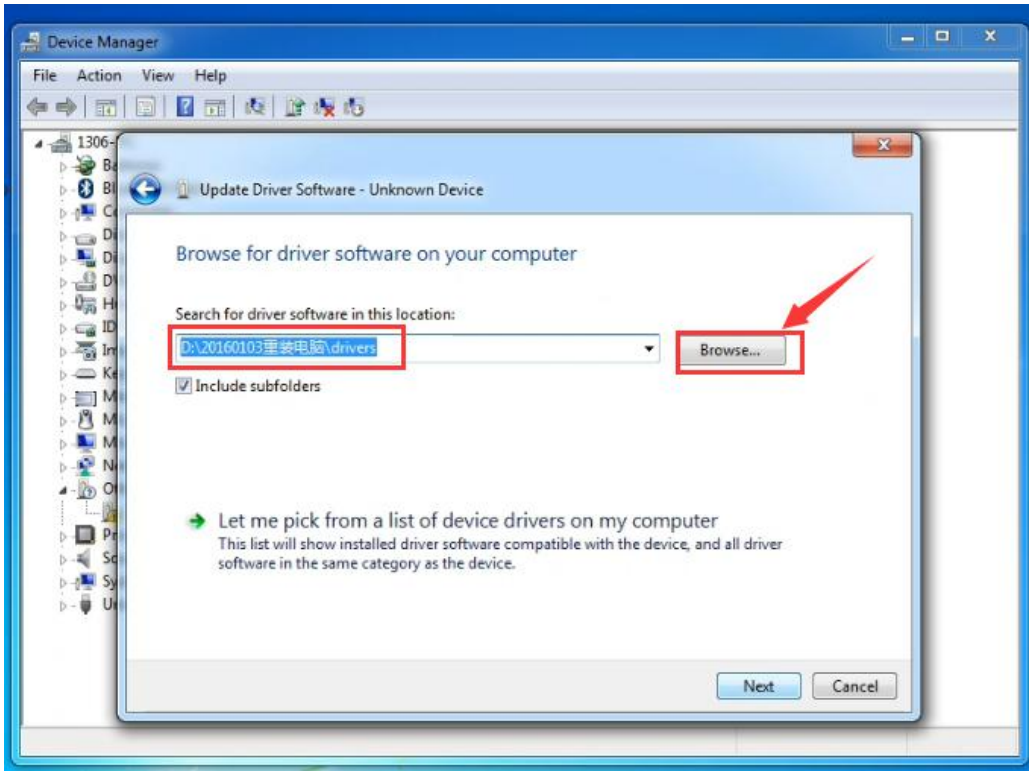
Click "Unknown device" to install driver, as shown below:



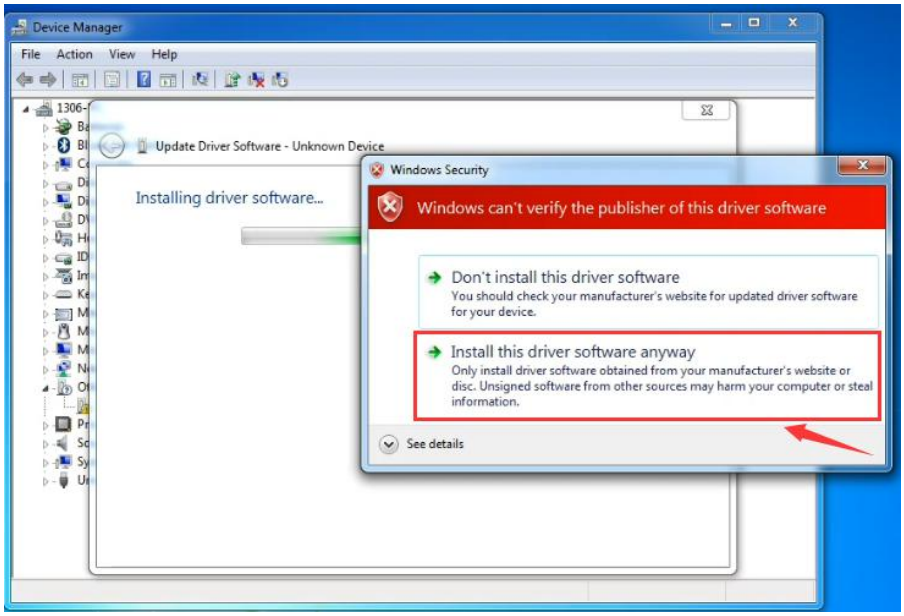
Then followed by the figure as below, select “Browse my computer for driver software”



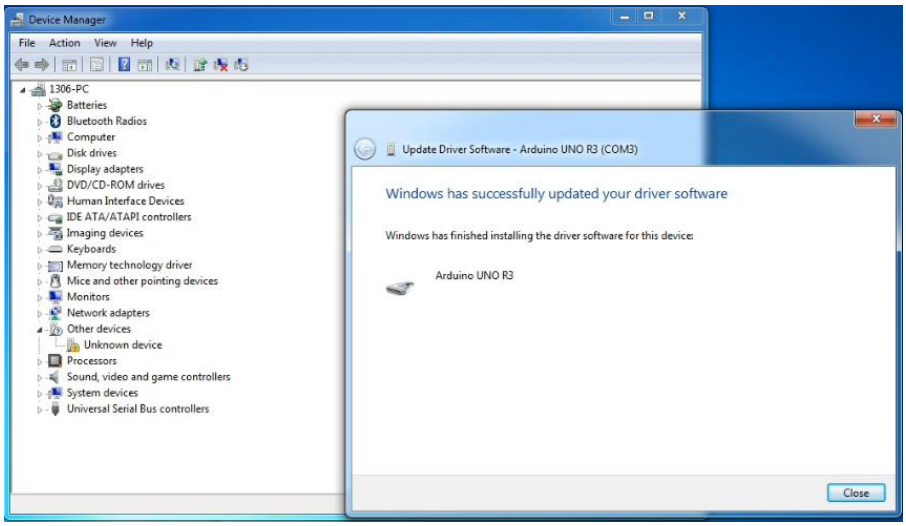
Next, find the drivers folder of Arduino installation location.



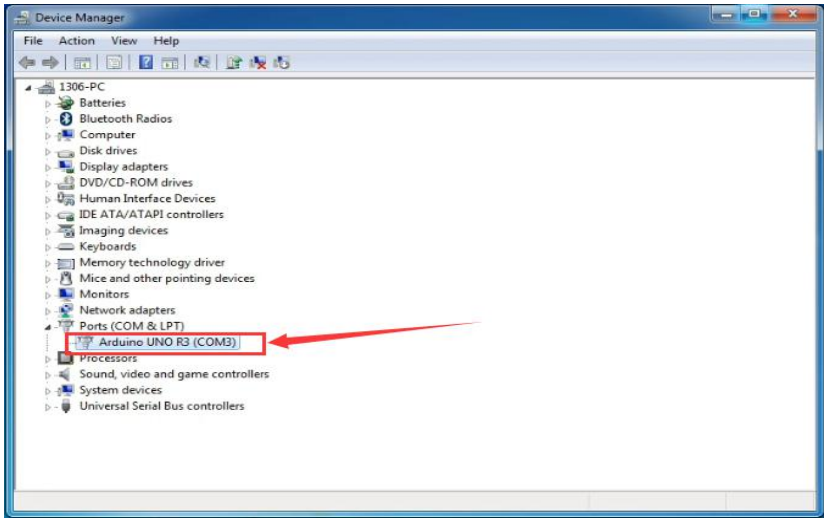
And click “Next”, select “Install this software anyway” as shown below, to install the driver.



Click “Close” after successfully installing the driver, as the figure shown below:



In this way, the driver is installed well. Right click on the Computer - Properties - Device Manager, you can check the Arduino port as shown below:



4. Experimental Courses

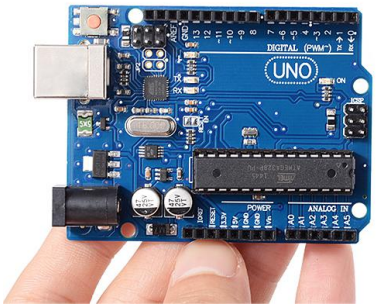
Lesson 1: Hello World

Introduction

First of all, we only need an UNO R3 and a download cable without other auxiliary components to display "Hello World!". This experiment allows Uno R3 to communicate with the PC, which is an introductory test hoped to lead everyone into the world of Arduino.

Hardware Required

UNO R3 * 1



USB Cable*1

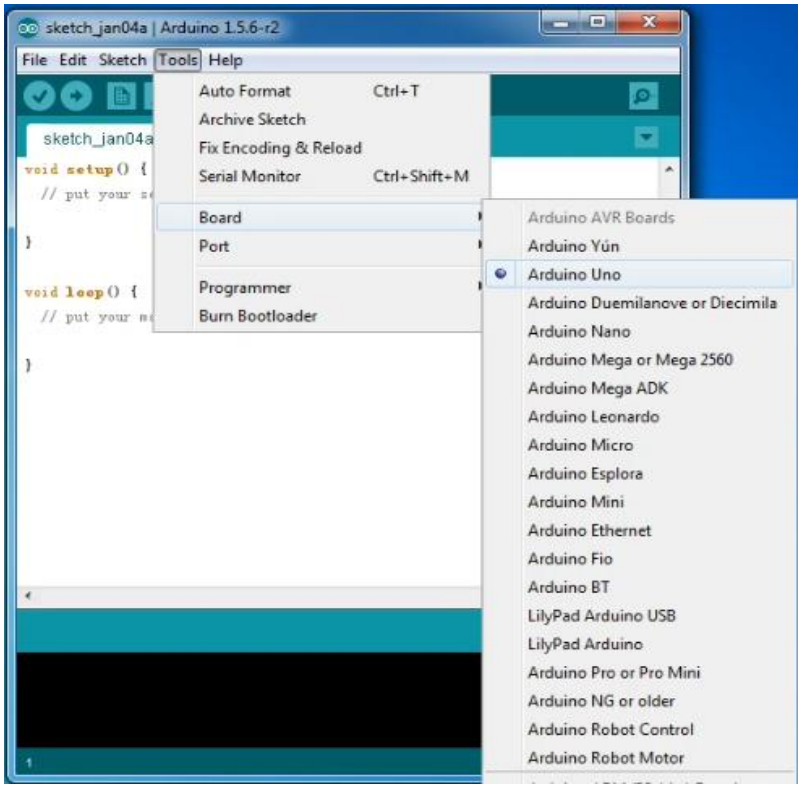
Sample Code

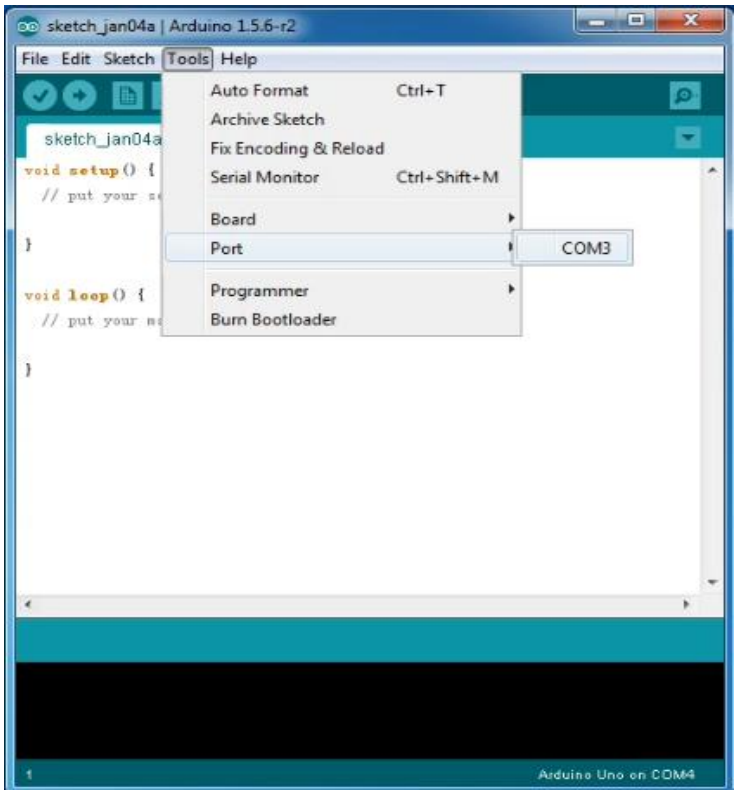
```
int val;
int ledpin=13;
void setup()
{
   Serial.begin(9600);
   pinMode(ledpin,OUTPUT);
}
void loop()
{
   val=Serial.read();
   if(val=='R')
   {
      digitalWrite(ledpin,HIGH);
      delay(500);
      digitalWrite(ledpin,LOW);
      delay(500);
```

```
        Serial.println("Hello World!");
    }
}
```

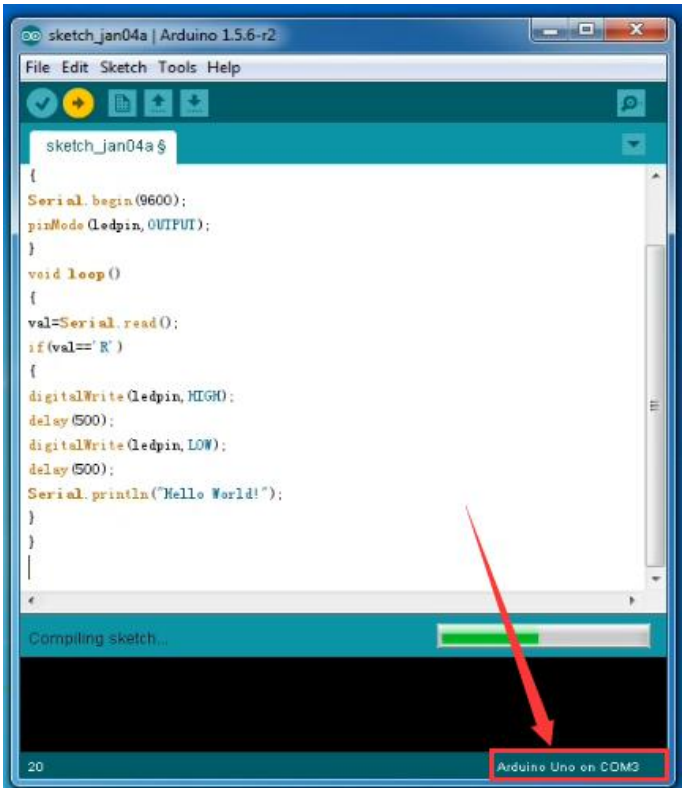
Test Result

First, open the Arduino IDE, set the Board and COM Port as the figure shown below:

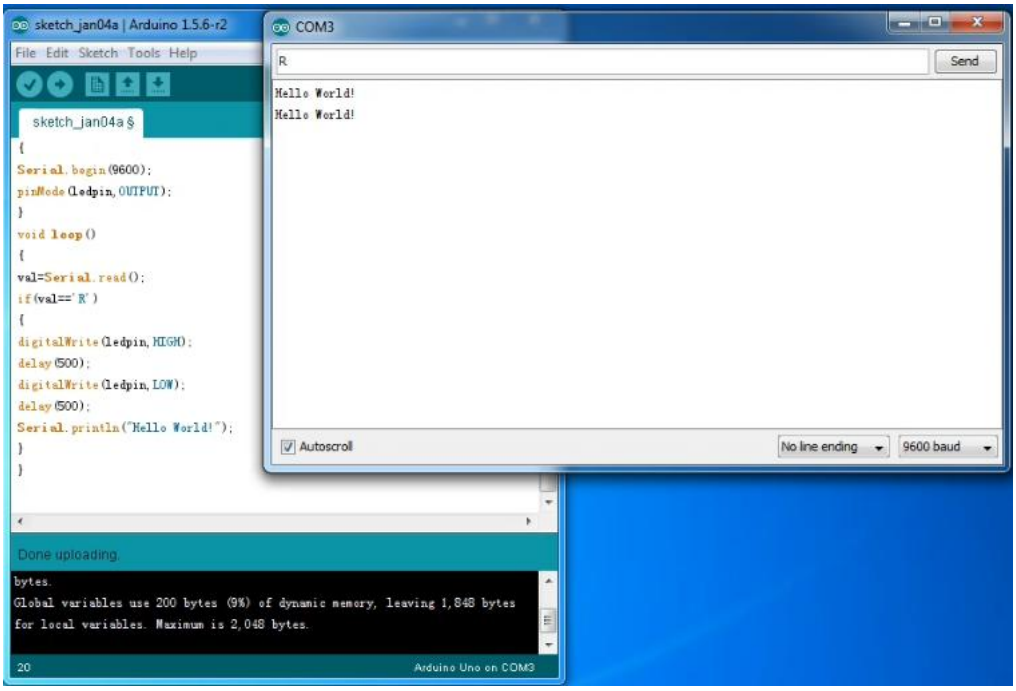




After setting well the Uno board, the lower right displays the same content as the device manager as the figure shown below:



Then, Click  on the compiler to check the compiling mistakes; Click  to upload the program to Arduino board. Uploading is successful, then open serial monitor on the upper right and enter an "R", set the baud rate of 9600, then click "Send". Arduino UNO board's D13 LED flashes once, and the serial monitor shows Hello World! as shown below:



Congratulations！ The first programming is finished successfully！

Lesson 2: LED Blinking

Introduction

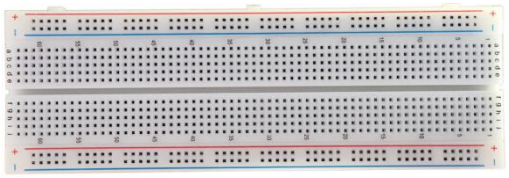
LED small light experiment is one of the more basic experiments. In the last lesson"Hello World!", we has used Arduino built-in LED, this time we use other I/O port and external LED light to complete the experiment.

Hardware Required

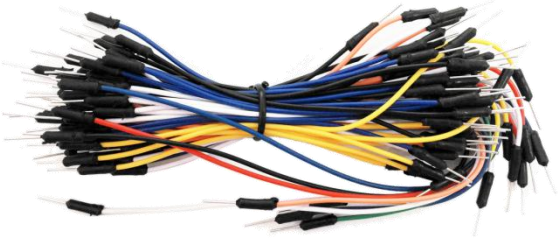
Red M5 Direct Plug-in LED*1

220Ω Direct Plug-in Resistance*1

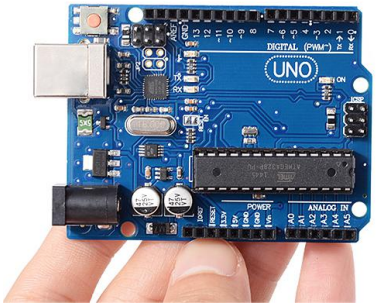
Breadboard*1



Breadboard Jumper Wires*1 bunch



UNO R3 * 1

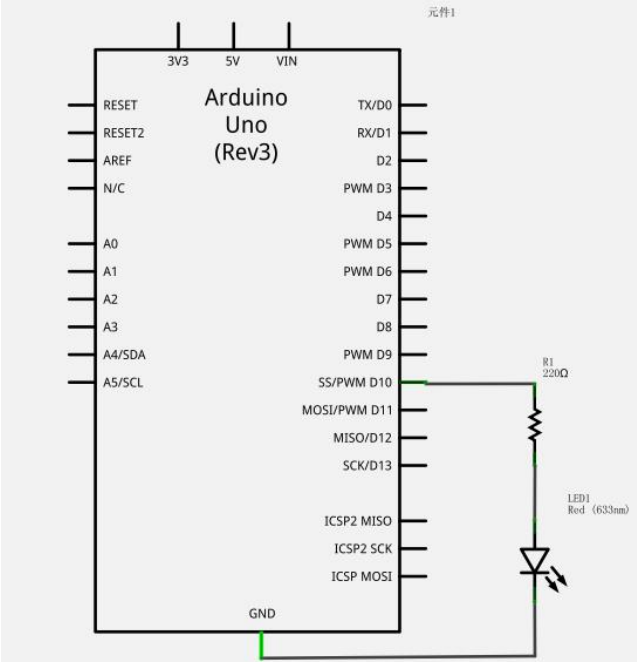


USB Cable*1

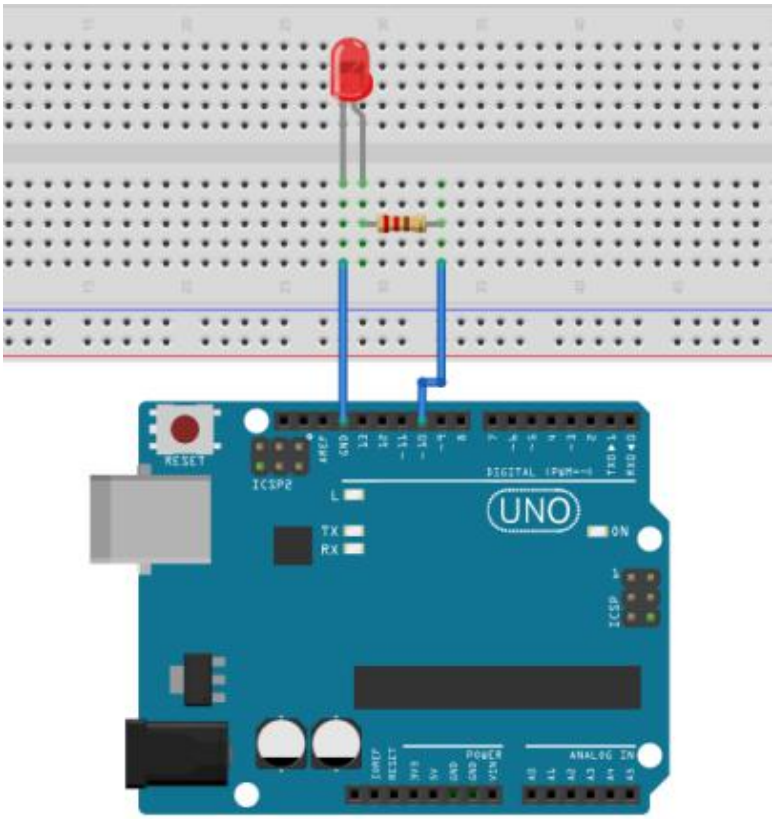
Schematic & Wiring Diagram

The next step is to link the physical map according to the following schematic diagram of small light experiment, here using the digit 10 interface. When using a light-emitting diode (LED), it needs to connect a current-limiting resistor with 220Ω resistor, otherwise the current is too large to burn the light-emitting diode.

Schematic Diagram of Small Light Experiment:



Physical Connection Diagram:



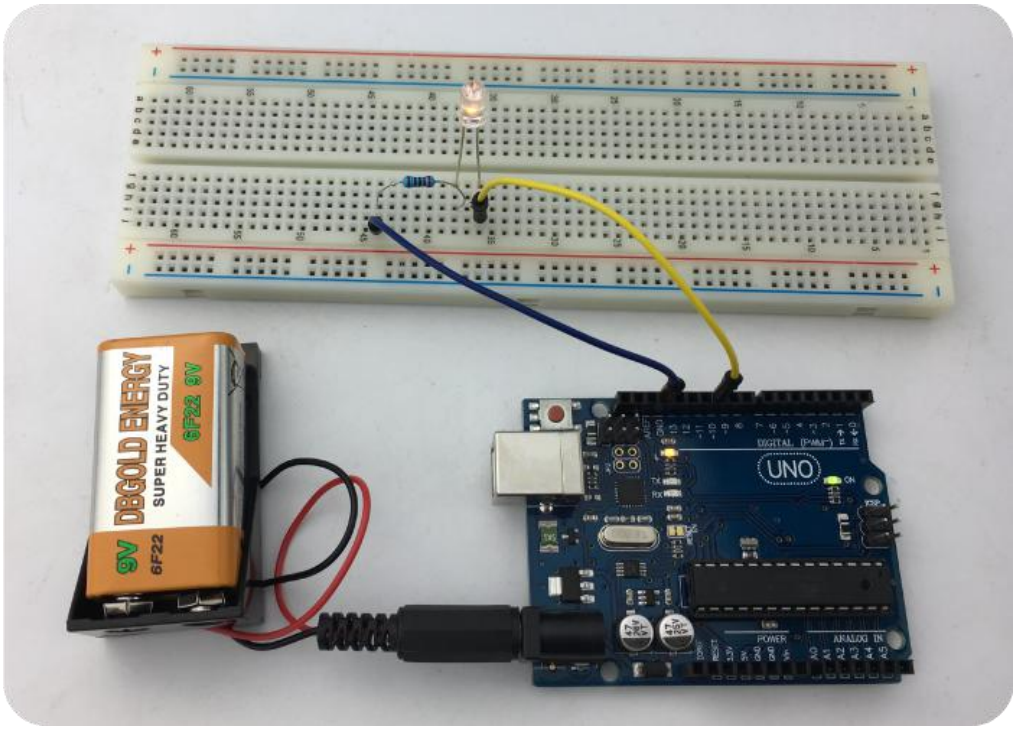
Sample Code

```
int ledPin = 10; //define digit 10 interface
void setup()
{
pinMode(ledPin, OUTPUT);//define the light port as output interface
}
void loop()
{
digitalWrite(ledPin, HIGH); //light up small light
delay(1000); //delay 1 second
digitalWrite(ledPin, LOW); //put out small light
delay(1000); // delay 1 second
}
```

Test Result

Uploading the program, you can see external small light in digit 10 port flashing.

The experimental phenomenon shows that the LED continues to blink with an interval of about one second.



The LED blinking test is now completed. Thank you!

Lesson 3: Flowing Water Light Effect

Introduction

In life we often can see some billboards composed of various colors of LED lights, in which the LED lights constantly change to form various effects. This experiment uses the LED light program to simulate the effect of advertisement light.

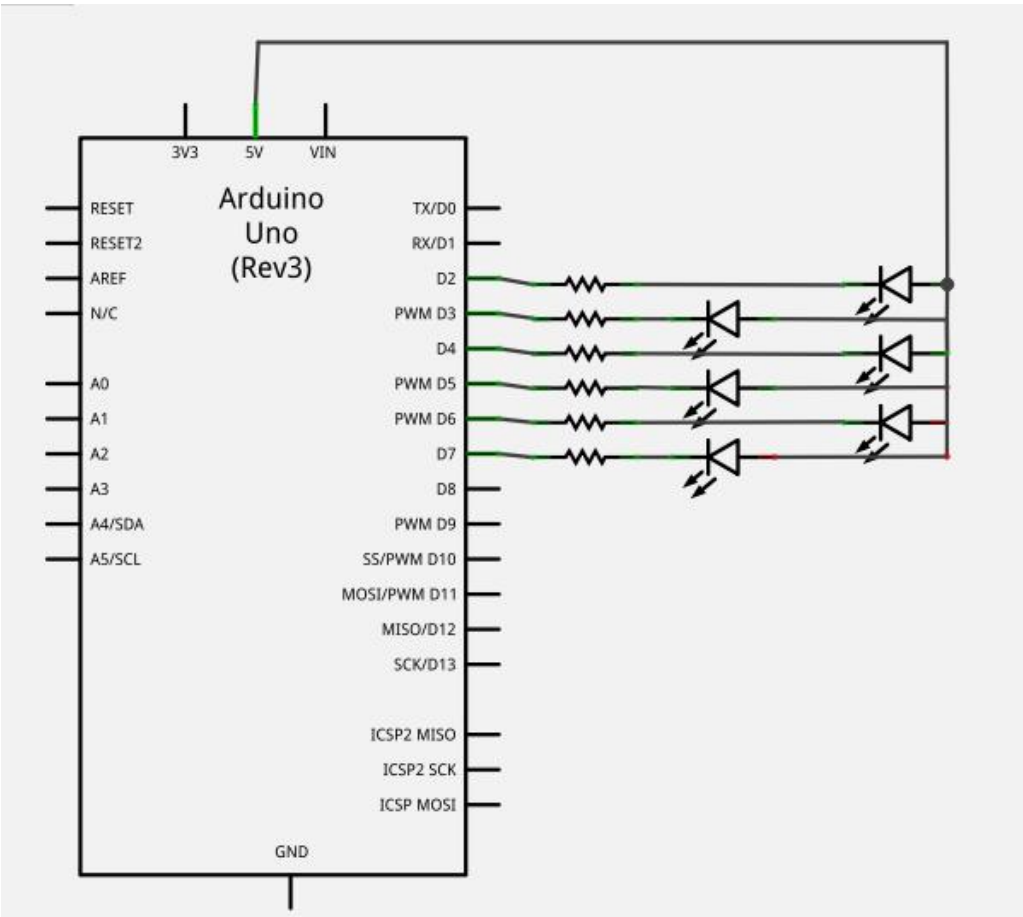
Hardware Required

LED lights*6
220 Ω Resistance*6
Colorful Breadboard Jump Wire*Several

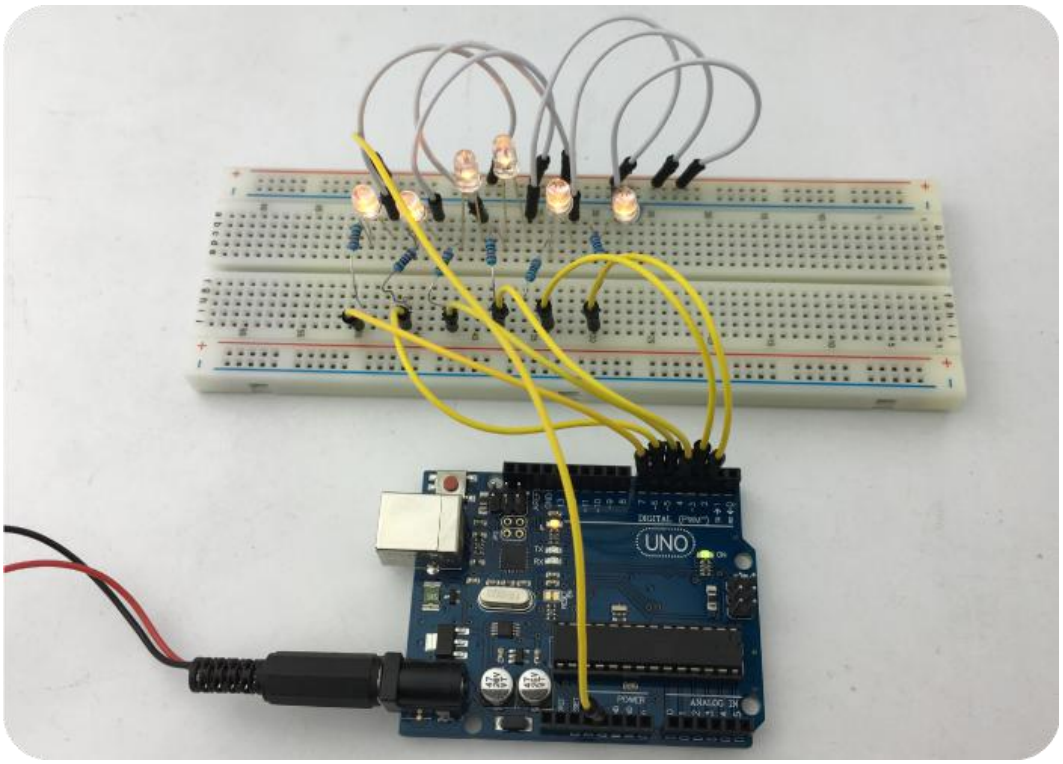
Schematic & Wiring Diagram

According to the wiring diagram of diode, six LED lights are successively connected to the digit 2 to 7 pin. The wiring of flowing water light experiment as the figure shown below:

Connection Schematic Diagram:



Physical Connection Diagram:



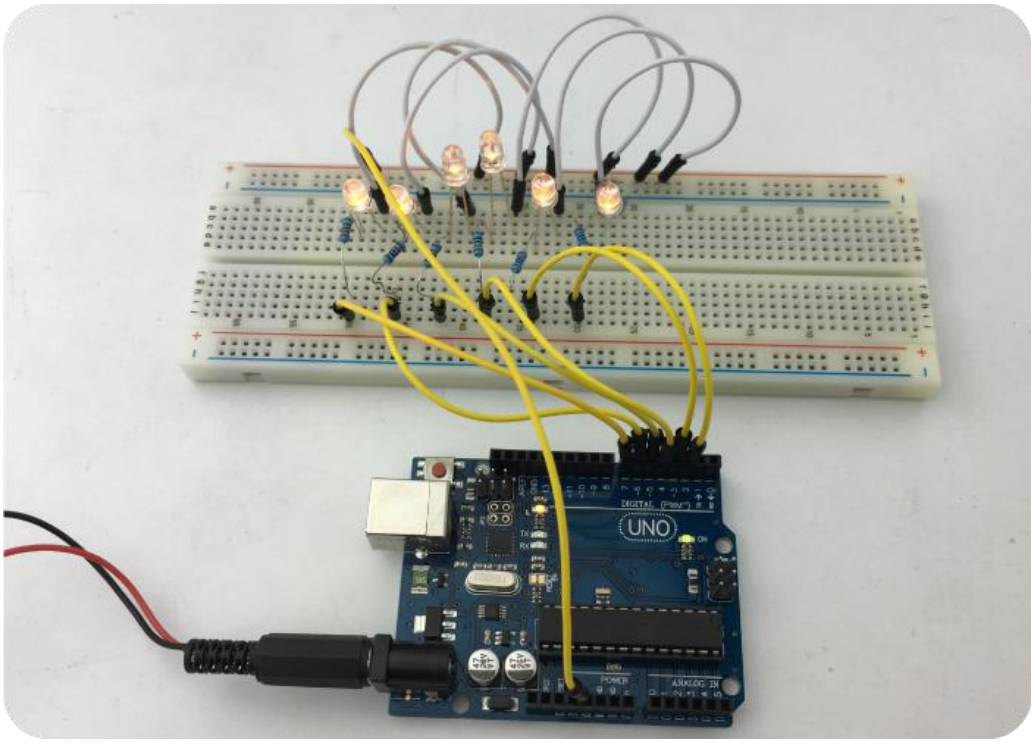
Sample Code

```
int BASE = 2 ;    //the first LED I/O port
int NUM = 6;      //total number of LED

void setup()
{
    for (int i = BASE; i < BASE + NUM; i ++)
    {
       pinMode(i, OUTPUT);      //set the digital I/O pin to output
    }
}

void loop()
{
    for (int i = BASE; i < BASE + NUM; i ++)
    {
       digitalWrite(i, LOW);         //set the digital I/O port output “LOW”, gradually turning off the light.
       delay(200);               //delay
    }
    for (int i = BASE; i < BASE + NUM; i ++)
```

```
    {
      digitalWrite(i, HIGH);        //set the digital I/O port output “HIGH”, gradually turning on the light.
      delay(200);                   //delay
    }
}
```

Test Result

You can see the LED light is flashing after programming, as the figure shown below:



Lesson 4: Traffic Light

Introduction

we have finished a small light control experiment before, and this time we'll do a little bit more complicated traffic light test. This test actually expands one small light into three small lights with color, so as to achieve the analog traffic light.

Hardware Required

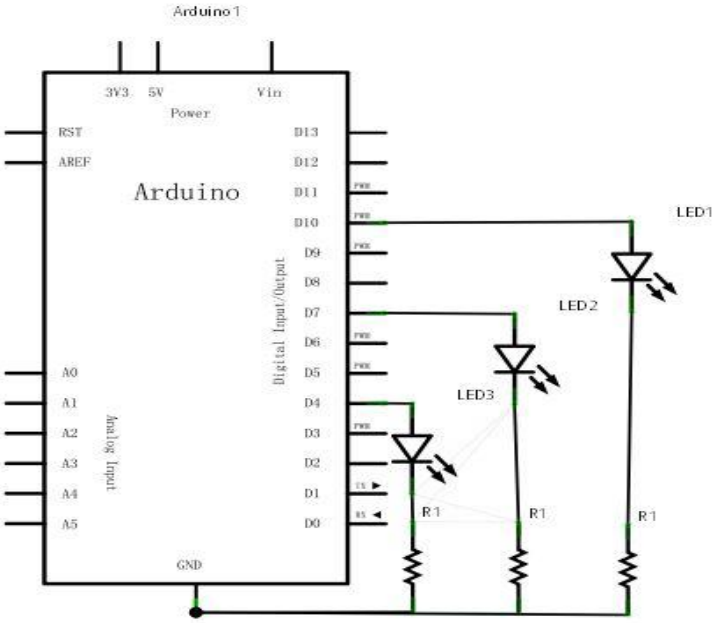
Red M5 Direct Plug-in LED*1
Yellow M5 Direct Plug-in LED*1
Green M5 Direct Plug-in LED*1

220Ω Resistance*3
Breadboard*1
Breadboard Jumper Wires*1 bunch

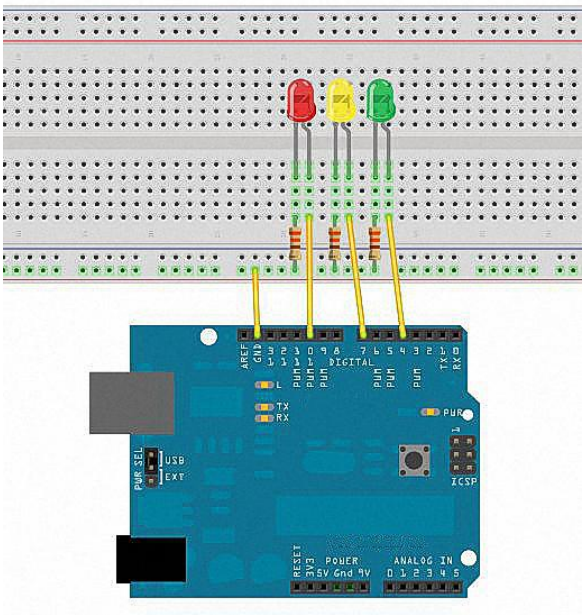
Schematic & Wiring Diagram

We can start the test after preparing those components needed, getting prompted from the former LED flashing experiment. The referential schematic diagram we provided is as below, respectively using the digit 10, 7, 4 interfaces.

Connection Schematic Diagram:



Wiring Diagram:



Sample Code

```
int redled =10; //define digit 10 interface
int yellowled =7; //define digit 7 interface
int greenled =4; //define digit 4 interface
void setup()
{
pinMode(redled, OUTPUT);//define the red light interface as the output interface
pinMode(yellowled, OUTPUT); //define the yellow light interface as the output interface
pinMode(greenled, OUTPUT); //define the green light interface as the output interface
}
void loop()
{
digitalWrite(greenled, HIGH);////turn on green light
delay(5000);//delay 5 seconds
digitalWrite(greenled, LOW); //put out green light
for(int i=0;i<3;i++)//flicker for three times, yellow light blinking effect.
{
delay(500);//delay 0.5 second
digitalWrite(yellowled, HIGH);//turn on yellow light
delay(500);//delay 0.5 second
digitalWrite(yellowled, LOW);//put out yellow light
}
delay(500);//delay 0.5 second
digitalWrite(redled, HIGH);//turn on red light
delay(5000);//delay 0.5 second
digitalWrite(redled, LOW);//put out red light
}
```

Test Result

After programming, you can see you own designed traffic light. Green light is on for five seconds then goes out; yellow light flashes cyclically three times; and red light is on for five seconds, repeating in turn.

Lesson 5: Button Controlled LED

Introduction

I/O port means INPUT interface and OUTPUT interface. Until now, we only apply the output function of Arduino I/O port in the small lights experiments. As for this lesson, we try to apply the input function of Arduino I/O port, namely reading the output value of external devices. We can use a button and an LED light to finish the test combined input and output functions, letting

you simply know the I/O functions. Button switch belongs to the switching (digital) element, pressing down for the closed (conduction)state.

Hardware Required

Button switch*1

Red M5 Direct Plug-in LED*1

220Ω Resistance*1

10KΩ Resistance*1

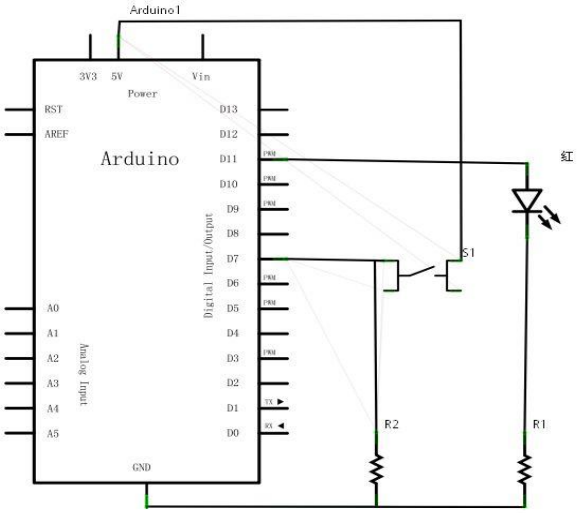
Breadboard*1

Breadboard Jumper Wires*1 bunch

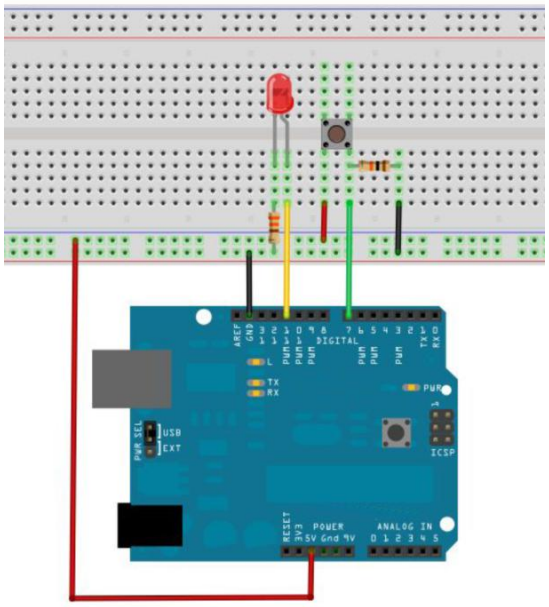
Schematic & Wiring Diagram

Connect the button switch to digit 7 interface; connect red LED light to digit 11 interface.

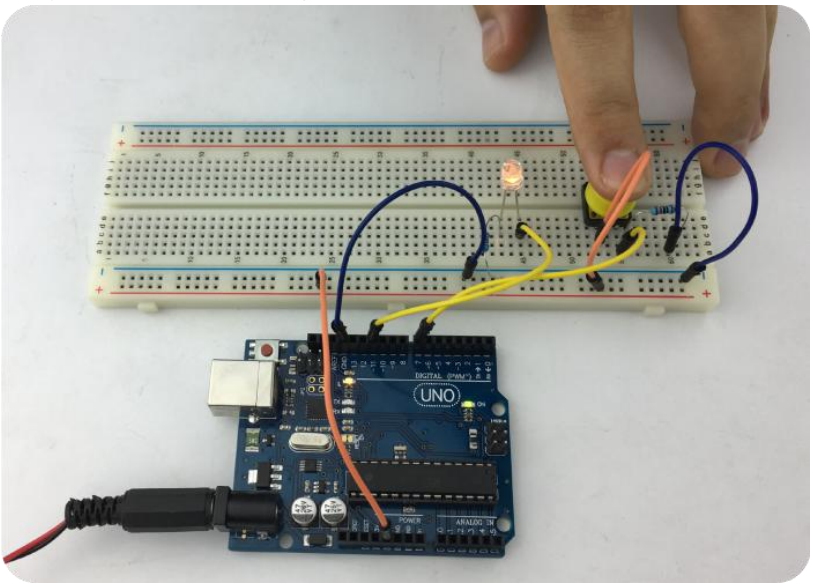
Connection Schematic Diagram:



Wiring Diagram:



Physical Connection Diagram:



Sample Code

```
int ledpin=11;//define digit 11 interface
int inpin=7;//define digit 7 interface
int val;//define variable val
void setup()
{
pinMode(ledpin,OUTPUT);//define the small lamp interface as the output interface
pinMode(inpin,INPUT);//define the button interface as input interface
}
void loop()
{
val=digitalRead(inpin);//read the digit 7 port level value assigning to val
if(val==LOW)//check the button pressed down or not, the small lamp lights up if pressed down.

{ digitalWrite(ledpin,LOW);}
else
{ digitalWrite(ledpin,HIGH);}
}
```

Test Result

when the button is pressed down, the LED will be on, otherwise the LED is off.

After downloading the program, this experiment combined small light and button is finished. The test principle is very simple and is widely used in various kinds of circuits and electric appliances. Take a typical application as an example, when pressing down any buttons of mobile phone,its backlight will light up. You can also control the 220V light by using the LED as a relay.

Lesson 6: Responder

Introduction

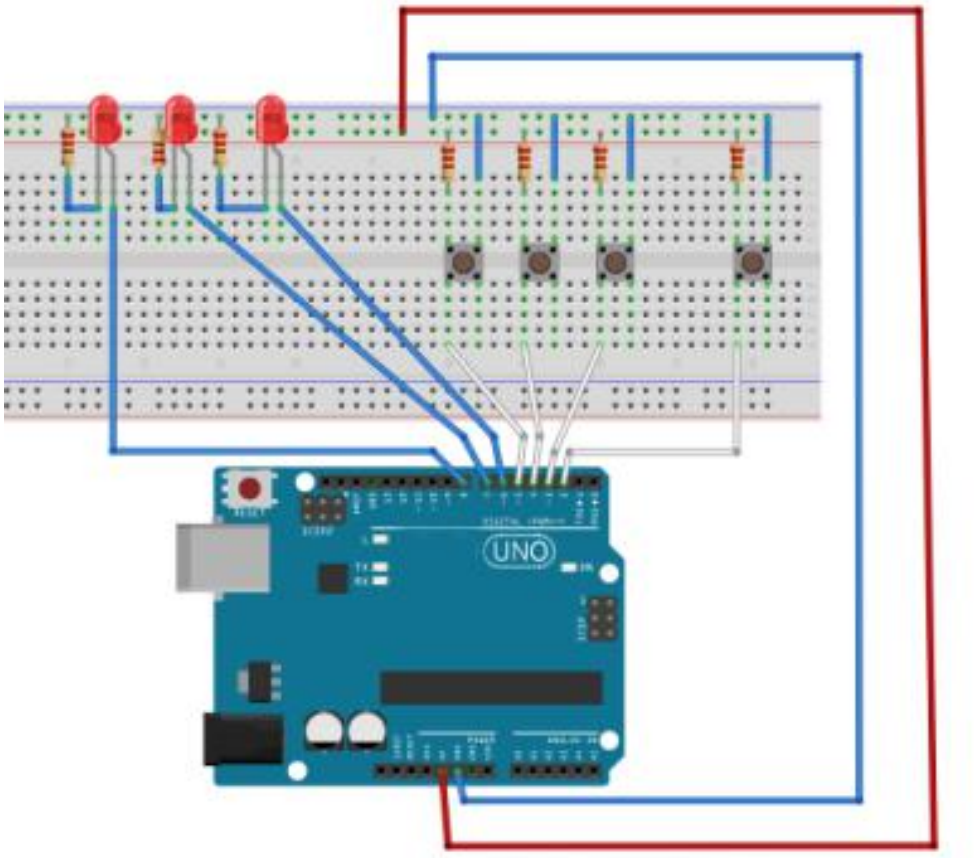
In this lesson, we make a further study based on the last test of button controlled LED, using three buttons to match with three lights. Besides, add more a reset button, using total 7 digital I/O ports. We use four buttons to control three LEDs, making a responder.

Hardware Required

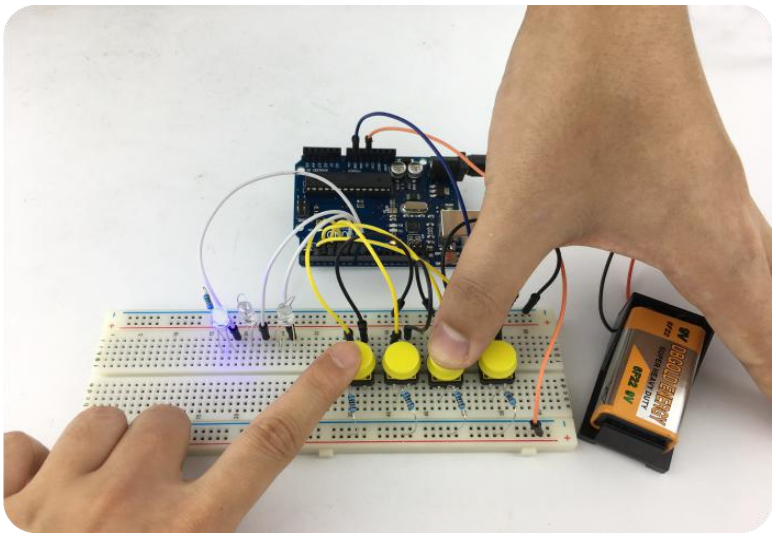
Button switch*4
Red M5 Direct Plug-in LED*3
220Ω Resistance*3
10KΩ Resistance*4
Breadboard*1
Breadboard Jumper Wires*1 bunch

Wiring Diagram

Connection Diagram:



Physical Connection Diagram:



Sample Code

```
int redled=8;          //red LED output
int yellowled=7;    //yellow LED output
int greenled=6;      //green LED output
int redpin=5;          //red button pin
int yellowpin=4;    //yellow button pin
int greenpin=3;      //green button pin
int restpin=2;      //define reset button pin
int red;
int yellow;
int green;
void setup()
{
pinMode(redled,OUTPUT);
pinMode(yellowled,OUTPUT);
pinMode(greenled,OUTPUT);
pinMode(redpin,INPUT);
pinMode(yellowpin,INPUT);
pinMode(greenpin,INPUT);
}
void loop()    //button loop scanning。
{
red=digitalRead(redpin);
yellow=digitalRead(yellowpin);
green=digitalRead(greenpin);
if(red==LOW)RED_YES();
if(yellow==LOW)YELLOW_YES();
if(green==LOW)GREEN_YES();
}
```

```
void RED_YES()//keep red led lighting until the reset button pressed down to end the loop.
{
  while(digitalRead(restpin)==1)
  {
   digitalWrite(redled,HIGH);
   digitalWrite(greenled,LOW);
   digitalWrite(yellowled,LOW);
  }
  clear_led();
}
void YELLOW_YES()//keep yellow led lighting until the reset button pressed down to end the loop.
{
  while(digitalRead(restpin)==1)
  {
  digitalWrite(redled,LOW);
  digitalWrite(greenled,LOW);
  digitalWrite(yellowled,HIGH);
  }
  clear_led();
}
void GREEN_YES()//keep green led lighting until the reset button pressed down to end the loop.

{
  while(digitalRead(restpin)==1)
  {
  digitalWrite(redled,LOW);
  digitalWrite(greenled,HIGH);
  digitalWrite(yellowled,LOW);
  }
  clear_led();
}
void clear_led()//clear LED
{
  digitalWrite(redled,LOW);
  digitalWrite(greenled,LOW);
  digitalWrite(yellowled,LOW);
}
```

Test Result

The light will be on if its button is pressed down first. Then press down the REST button to reset it. Thus the responder is finished after programming.

Lesson 7: Buzzer Test

Introduction

Using Arduino can finish many interactive works. The most common seen and used is the sound and light show. We have used LED in the previous experiment, so this lesson is to make the circuit sound. The most common sounding components are buzzers and speakers, and we use the easier buzzer to finish the test.

Hardware Required

Buzzer*1----------mark '+' connected to digit 8 pin.

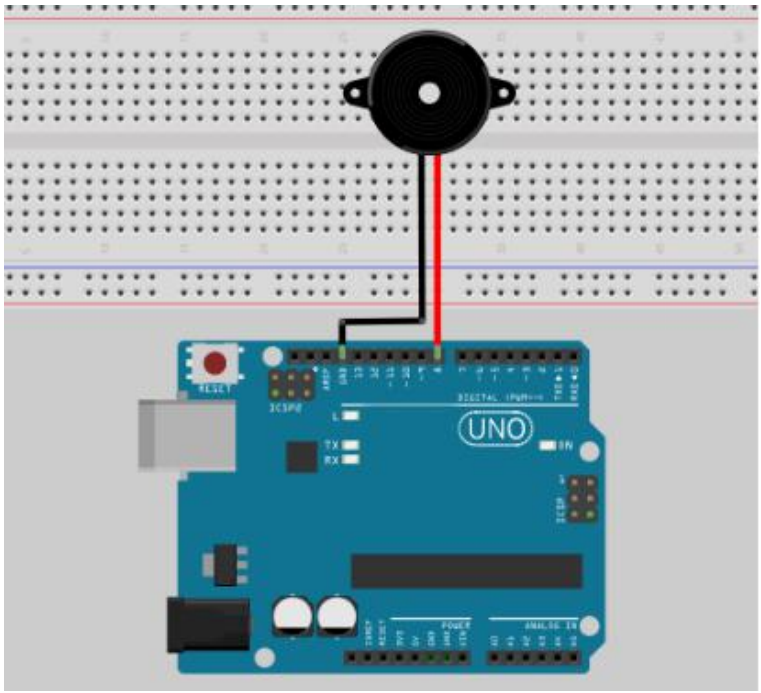
Button*1

Breadboard*1

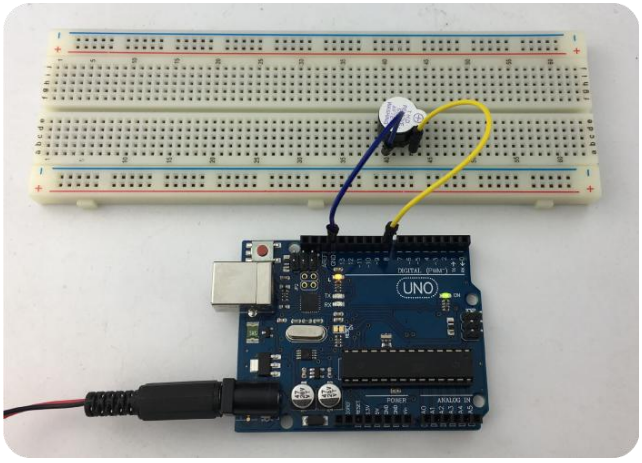
Breadboard Jumper Wires*1 bunch

Wiring Diagram

Connection diagram：



Physical connection diagram:



When connecting the circuit, note that the buzzer has positive and negative ends. It can be seen that the buzzer has red and black wiring in the physical map.

Sample Code

```
int buzzer=8;//set the digit IO pin of controlling the buzzer
void setup()
{
  pinMode(buzzer,OUTPUT);//set the digit IO pin mode, OUTPUT means output.
}
void loop()
{
  unsigned char i,j;//define variable
  while(1)
  {
    for(i=0;i<80;i++)//output a frequency of sound.
    {
      digitalWrite(buzzer,HIGH);//make a sound
      delay(1);//delay 1ms
      digitalWrite(buzzer,LOW);//make no sound
      delay(1);//delay ms
    }
    for(i=0;i<100;i++)//output another frequency sound
    {
      digitalWrite(buzzer,HIGH);//make a sound
      delay(2);//delay 2ms
      digitalWrite(buzzer,LOW);//make no sound
      delay(2);//delay 2ms
    }
  }
}
```

Test Result

The test is completed after downloading the program. You can hear the buzzer is always ringing.

Lesson 8: Analog Value Reading

Introduction

In this lesson, we begin to learn about the use of analog I/O interface. Arduino has analog port 0 to 5 in total 6 analog ports, which can also be used as interface function reuse. Apart from analog interface function, these 6 ports can be used as a digital interface, numbered as the number 14 to number 19. The experiment will use a typical component of analog value output, namely potentiometer.

Hardware Required

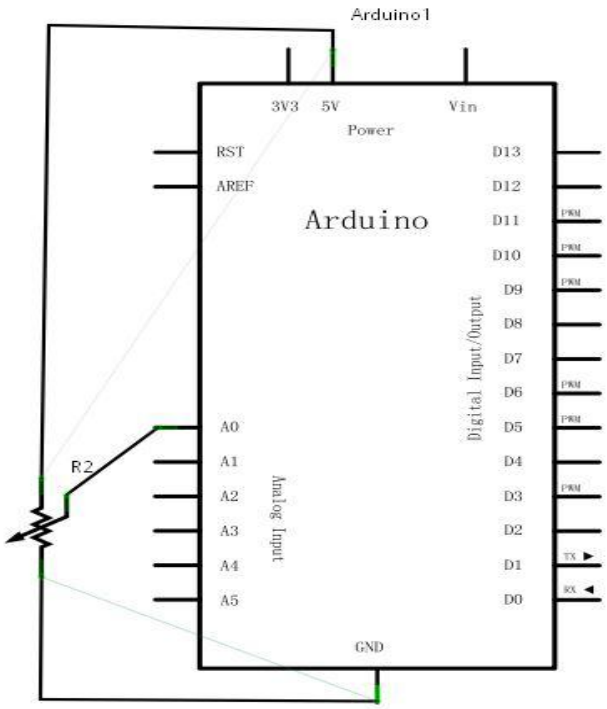
Potentiometer*1

Breadboard*1

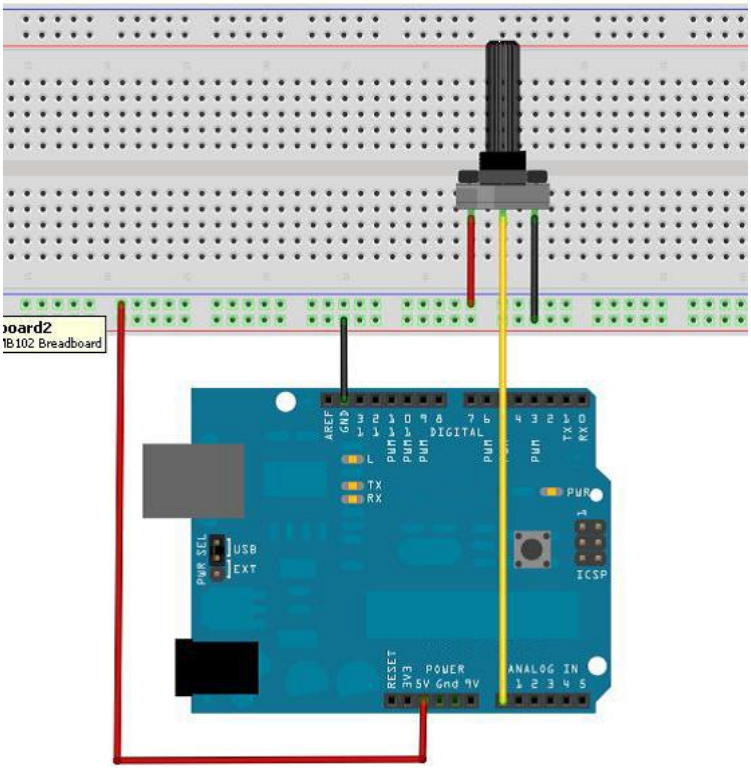
Breadboard Jumper Wires*1 bunch

Schematic & Wiring Diagram

Schematic Diagram：



Connection Diagram：

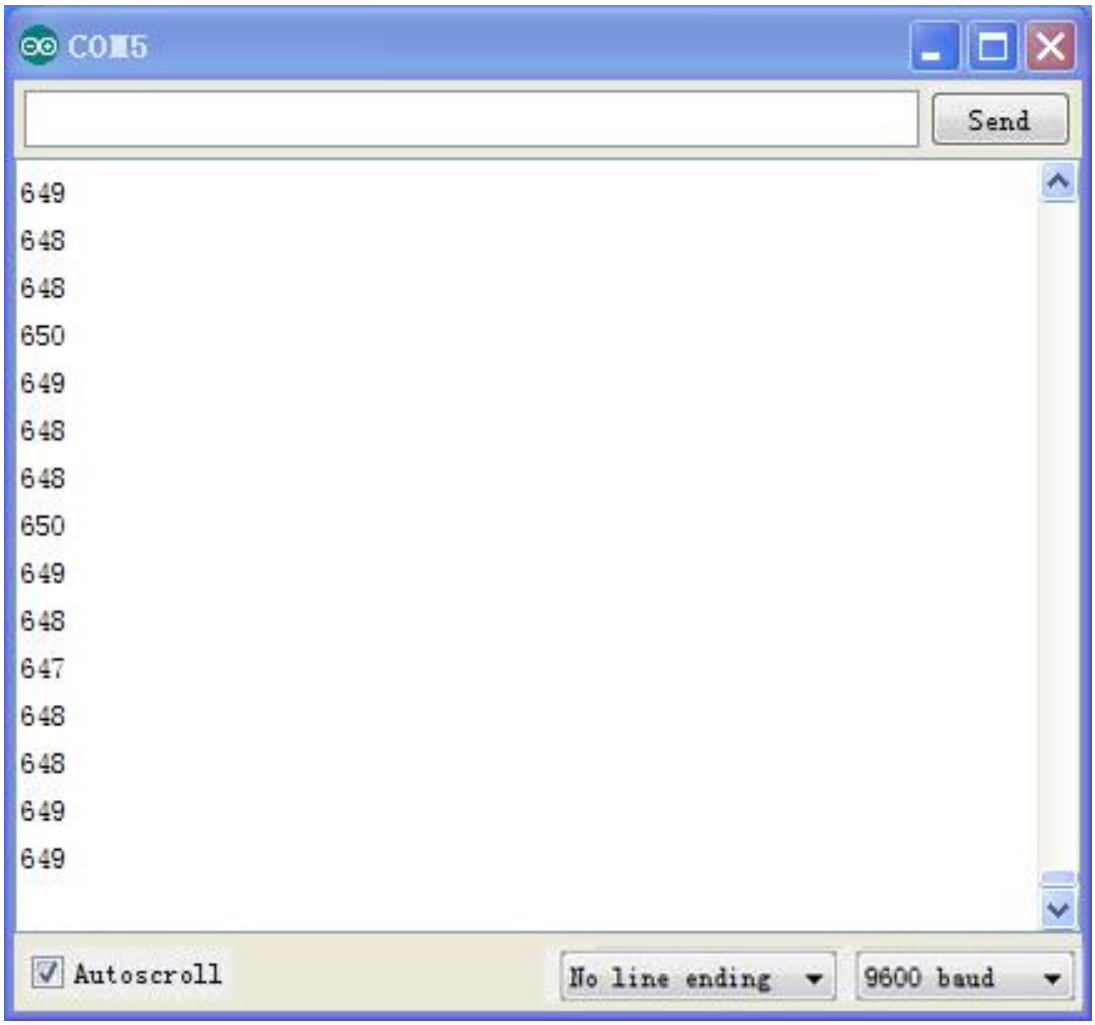


Sample Code

```
int potpin=0;//define analog port 0
int ledpin=13;//define analog port 13
int val=0;//define variable val,and assign initial value 0
void setup()
{
pinMode(ledpin,OUTPUT);//define the digital interface as output interface.
Serial.begin(9600);//set the baud rate of 9600
}
void loop()
{
digitalWrite(ledpin,HIGH);//turn on the LED in digital interface 13
delay(50);//delay 0.05 second
digitalWrite(ledpin,LOW);//turn off the LED in digital interface 13
delay(50);//delay 0.05 second
val=analogRead(potpin);//read the value of analog port 0,and assign it to val.
Serial.println(val);//printing the value of val
}
```

Test Result

The analog value is read as the figure shown below. You can see the value changes on the display when rotating the potentiometer knob.



Lesson 9: Digital Voltmeter

Introduction

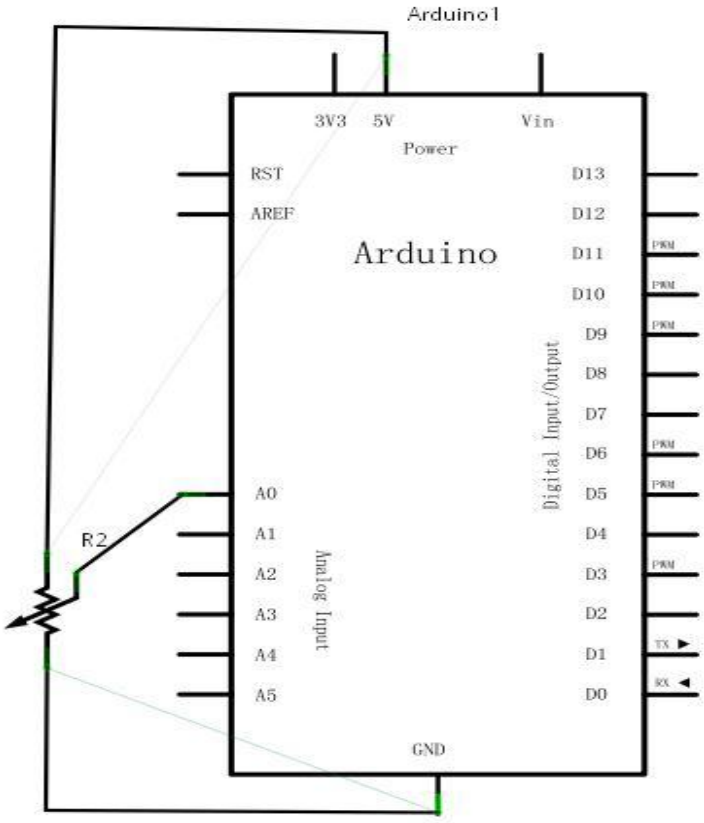
It is similar to the previous experiment, but this time the data needs to be calculated, getting the volt value of analog port A0.

Hardware Required

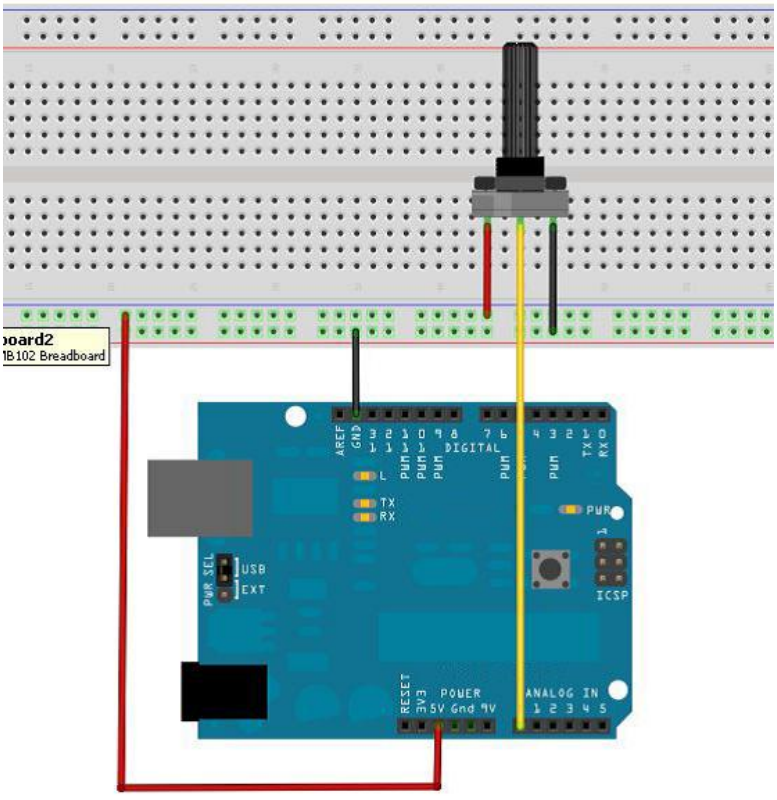
Potentiometer*1
Breadboard*1
Breadboard Jumper Wires*1 bunch

Schematic & Wiring Diagram

Schematic Diagram：



Connection Diagram：

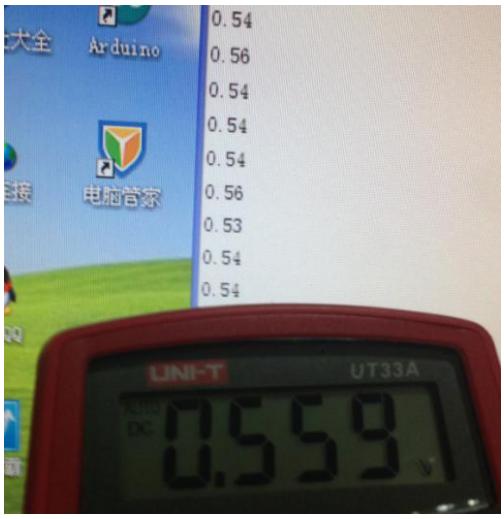


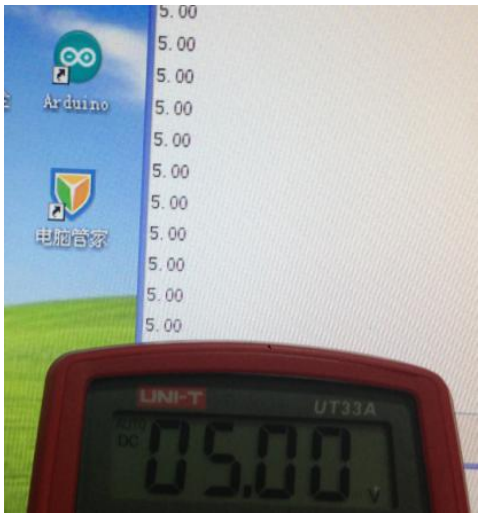
Sample Code

```
int potpin=0;      //define analog interface 0
int ledpin=13;    //define digital interface 13
int val=0;           //define variable val,and assign initial value 0
int v;
void setup()
{
   pinMode(ledpin,OUTPUT);    //define digital interface as output port
   Serial.begin(9600);          //set baud rate of 9600
}
void loop()
{
   digitalWrite(ledpin,HIGH);//turn on LED in digital interface 13
   delay(50);                          //delay 0.05 second
   digitalWrite(ledpin,LOW);//turn off LED in digital interface 13
   delay(50);                          //delay 0.05 second

   val=analogRead(potpin);    //read the value of analog port 0 and assign to val.
   v=map(val,0,1023,0,500);
//function description map(x,Amin,Amax,Bmin,Bmax)
   Return value long type
   Serial.println((float)v/100.00);        //display the value of v
}
```

Test Result

After programming, open the serial monitor, you can see the volt value of analog port A0 read as the figure shown below. Please use a voltmeter to measure the voltage of input port A0.





When you rotate the potentiometer knob, you can see the value on the display changes. So the voltmeter is rather available. The test is now completed. Thank you!

Lesson 10: Light-Controlled Sound

Introduction

The test makes use of light intensity to control the sound frequency of buzzer. The effect is very obvious that the greater the light intensity, the more rapid the sound. This lesson is quite easy, interesting and useful. Since the circuit is very common, hope you can learn more about it by analogy.

Hardware Required

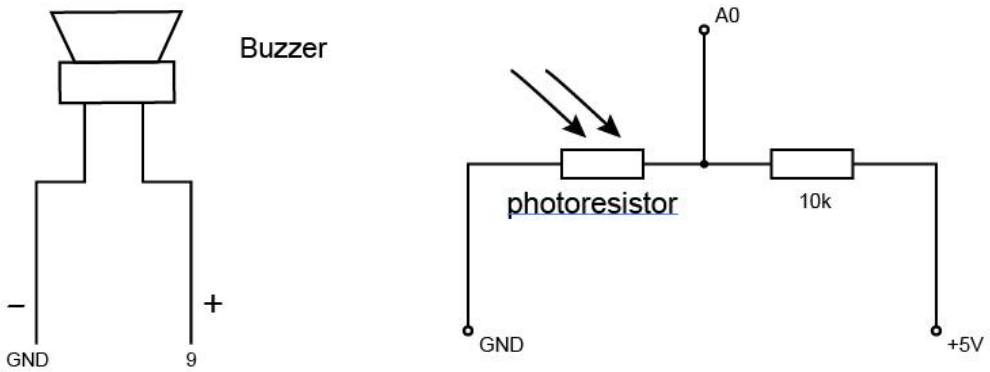
Photo-resistance*1

Buzzer *1

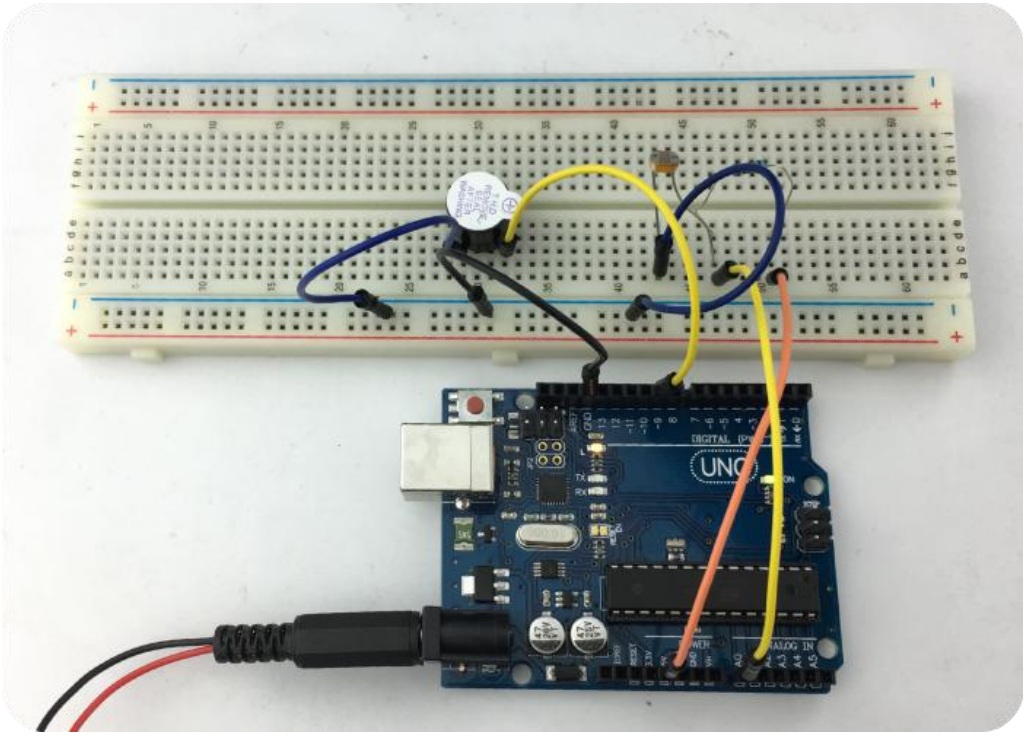
Colorful Breadboard Jumper Wires* Several

Wiring Diagram

Connection Diagram:



Circuit Connection Diagram:



Sample Code

```
int buzzer = 9; // define the output pin 9 of buzzer
int R_guangming = 0;//define the output pin of photo-resistance
int val;
void setup()
{
pinMode(buzzer,OUTPUT);//set the pin connected to buzzer as output
}
void voice_out(int del)//sound frequency to control function
{
  delay(del);// change the frequency by changing the delay.
  digitalWrite(buzzer,HIGH);
  delay(del);
  digitalWrite(buzzer,LOW);
}
void loop()
{
   val=analogRead(R_guangming);     //read the value of analog port , and assign it to val.
if(val<700)
     {
       voice_out(val); // convey the val value read to frequency to control the function
     }
}
```

Test Result

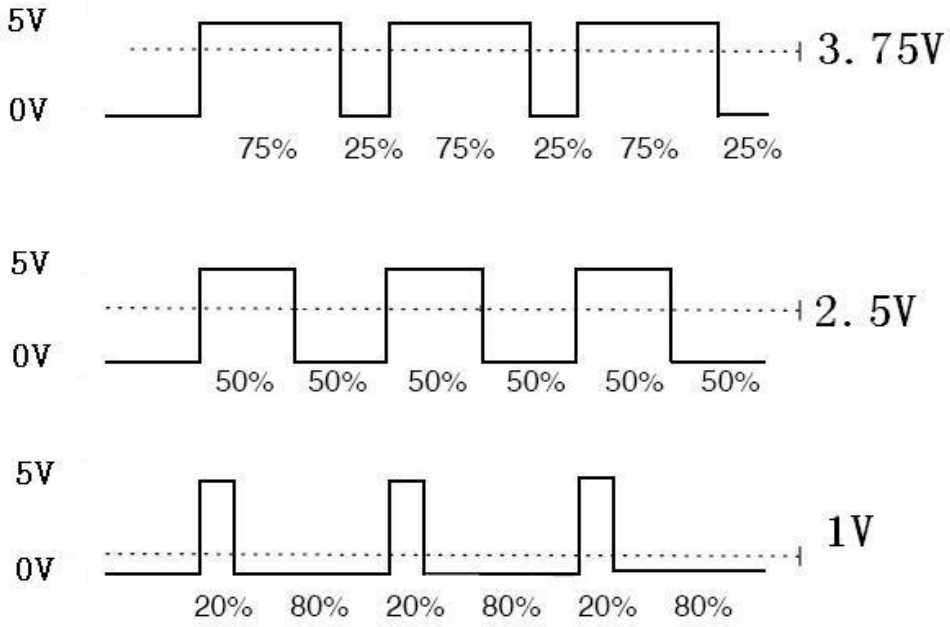
After uploading the program to the test board, you can use a flashlight or other light objects to illuminate the photo-resistor, and you can hear the buzzer sound frequency changes significantly when lighting. Mastering the program, you can design you own experiment, and can also use a photo-resistor to control led light's brightness.

Lesson 11: PWM Light Modulation

Introduction

Pulse Width Modulation (PWM) is a method of digital encoding for analog signal levels. Since the computer can not output analog voltage but can output 0 or 5V digital voltage value, so we use the high-resolution counter to encode a specific analog signal level by means of modulating the duty ratio of square wave.

As long as the bandwidth is sufficient, any analog value can be encoded using PWM. The output voltage value is calculated by the time of on and off. Output voltage = (on time / pulse time) * Maximum voltage value.



PWM is widely used in dimming lamps, motor speed control, sound production and so on.

Arduino controller has six PWM interfaces, respectively, digital interface 3,5,6,9,10,11. In the previous test, we have done a button controlled LED experiment, which is using digital signal to control digital interface. This time we come to complete a experiment of using potentiometer to control small lights.

Hardware Required

Potentiometer Module*1

Red M5 Direct Plug-in LED*1

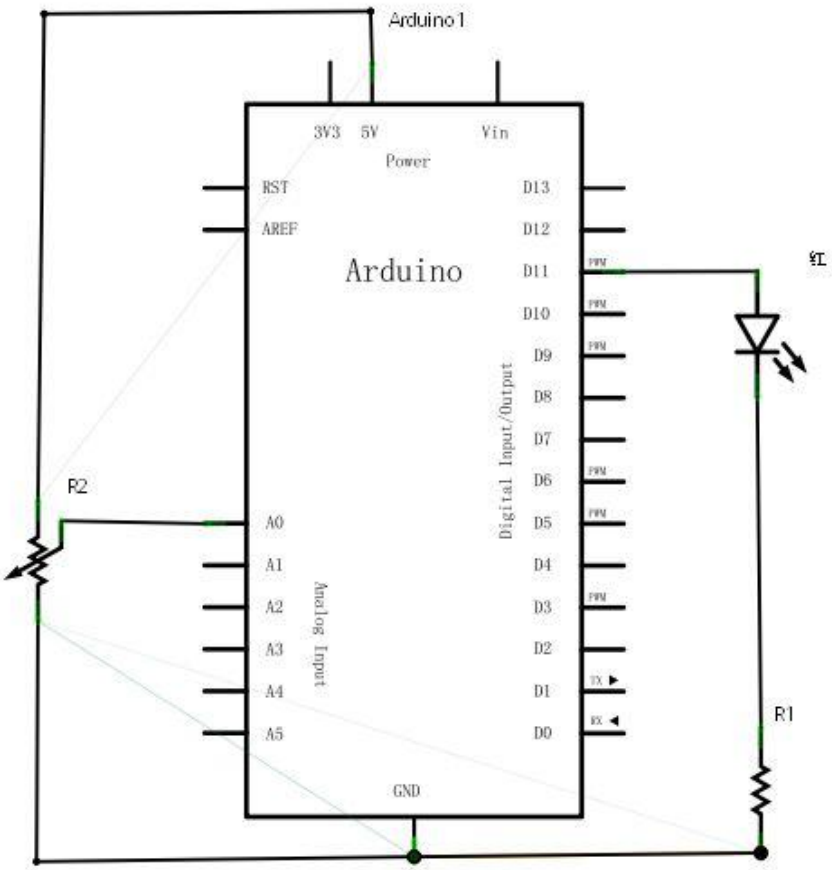
220Ω Direct Plug-in Resistor*1

Breadboard*1

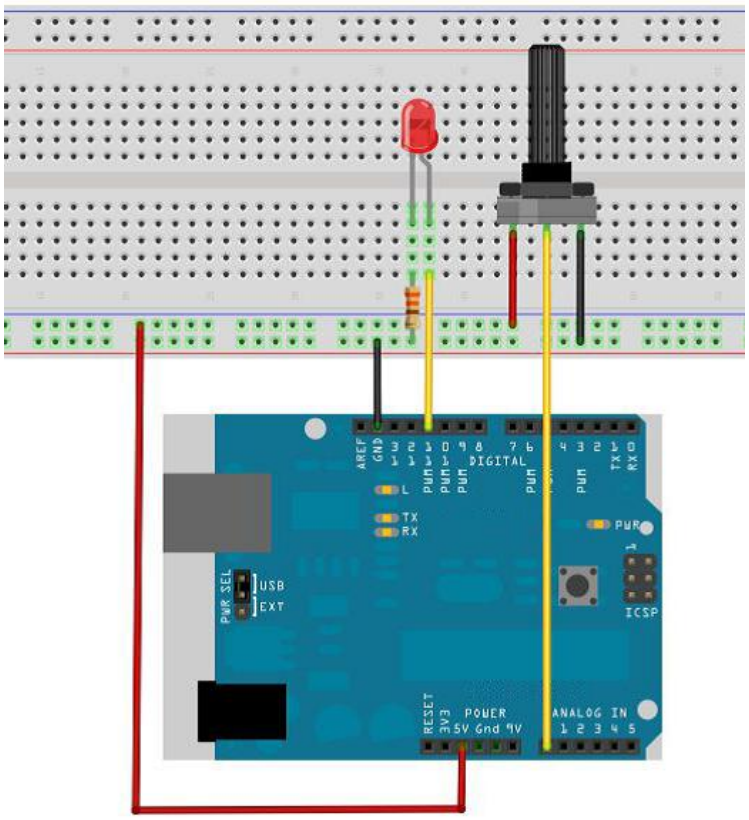
Breadboard Jumper Wires*1 bunch

Schematic & Wiring Diagram

Schematic Diagram:



Wiring Diagram:



Sample Code

```
int potpin=0;//define analog port 0
int ledpin=11;//define digital port 11（PWM output）
int val=0;// store the variable value from sensor
void setup()
{
pinMode(ledpin,OUTPUT);//define digit 11 interface as output
Serial.begin(9600);//set baud rate of 9600
//notice：analog interface is automatically set to input
}
void loop()
{
val=analogRead(potpin);// read analog value of sensor and assign to val
Serial.println(val);//display val variable
analogWrite(ledpin,val/4);// open LED and set its brightness（PWM output max value is 255）
delay(10);//delay 0.01 second
}
```

Test Result

After uploading the program, when rotating the knob of potentiometer, you can not only see the value changes on the serial monitor but also the brightness changes of LED light on breadboard.

Lesson 12: Photosensitive Light

Introduction

After learning the above experiments, you may have an understanding of Arduino applications. Mastering the basic digital input and output, analog input and PWM generation, you can start to learn some sensor applications.

Photovaristor, known as photo-resistor, is using photoelectric effect of semiconductor to make a resistance value which varies from the intensity of the incident light. When incident light is strong, the resistance decreases; and when the incident light is weak, the resistance increases.

Photovaristors are generally used for light measurement, light control and photoelectric conversion (converting light changes to electrical changes), also be widely used in various light controlling circuits, such as lighting control, photoswitch and so on.

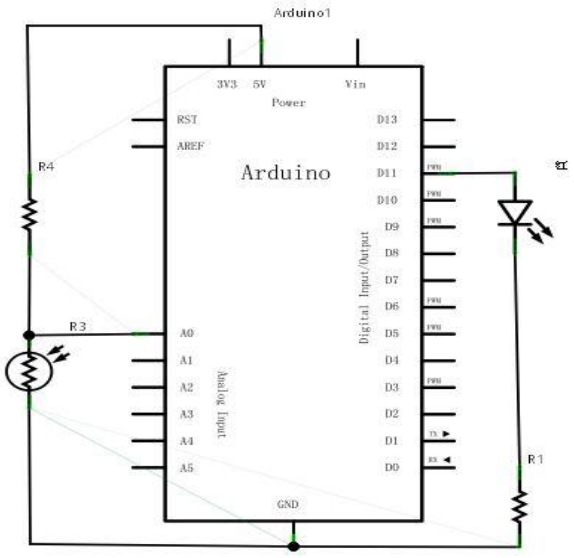
Next, we will do a simple test of photo-resistor application. Since the resistance value of photo-resistor component can vary from the intensity of light, it naturally need the analog port to read the analog value. This experiment can learn from the PWM interface experiment, just using photo-resistor instead of potentiometer to achieve that LED light brightness varies from the light intensity.

Hardware Required

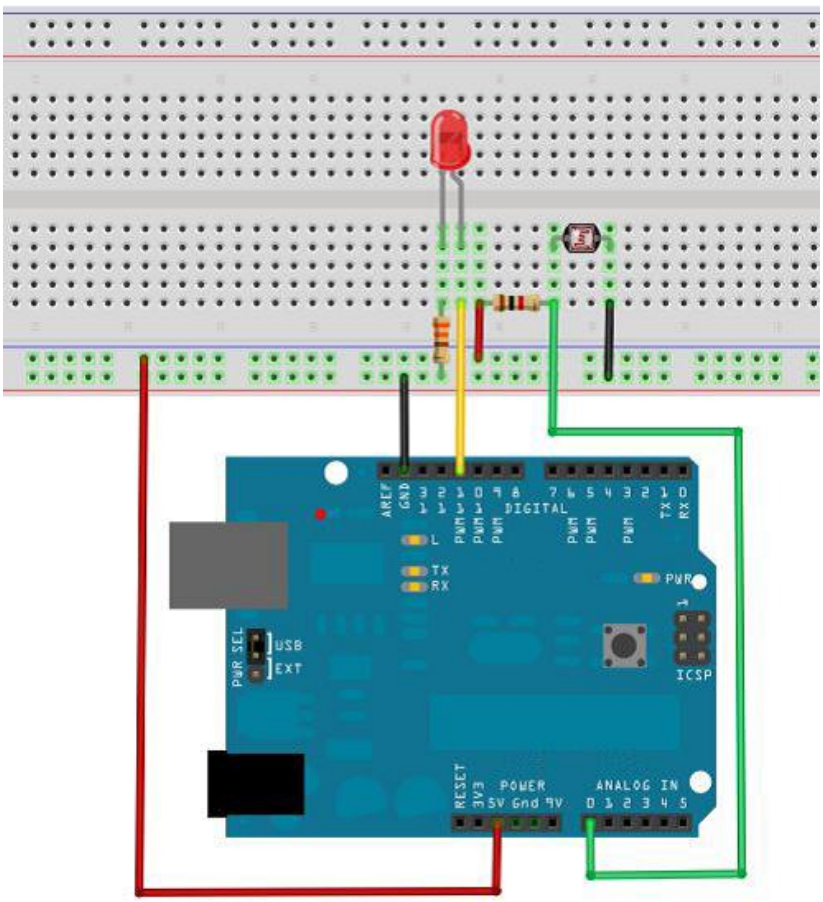
Photo-resistor*1
Red M5 Direct Plug-in LED*1
10KΩ Direct Plug-in Resistance*1
220Ω Direct Plug-in Resistance*1
Breadboard*1
Breadboard Jumper Wires*1 bunch

Schematic & Wiring Diagram

Schematic Diagram:



Connection Diagram:



Sample Code

```
int potpin=0;//define analog interface 0 connected to the photovaristor
int ledpin=11;//define digit 11 interface output PWM to adjust LED brightness

int val=0;//define variable val
void setup()
{
pinMode(ledpin,OUTPUT);//define digital port 11 as output
Serial.begin(9600);//set the baud rate of 9600
}
void loop()
{
val=analogRead(potpin);//read the analog value of sensor and assign it to val
Serial.println(val);//display val variable value
analogWrite(ledpin,val/4);//open LED and set its brightness（PWM output max value is 255）
delay(10);//delay 0.01 second
}
```

Test Result

After uploading the program, you can see the small light has a corresponding change when changing the environmental light intensity of photovaristor. There are numerous applications of phtovaristor in our daily life. It is hoped that you can make more better interactive works after studying this lesson.

Lesson 13: LM35 Temperature Sensor

Introduction

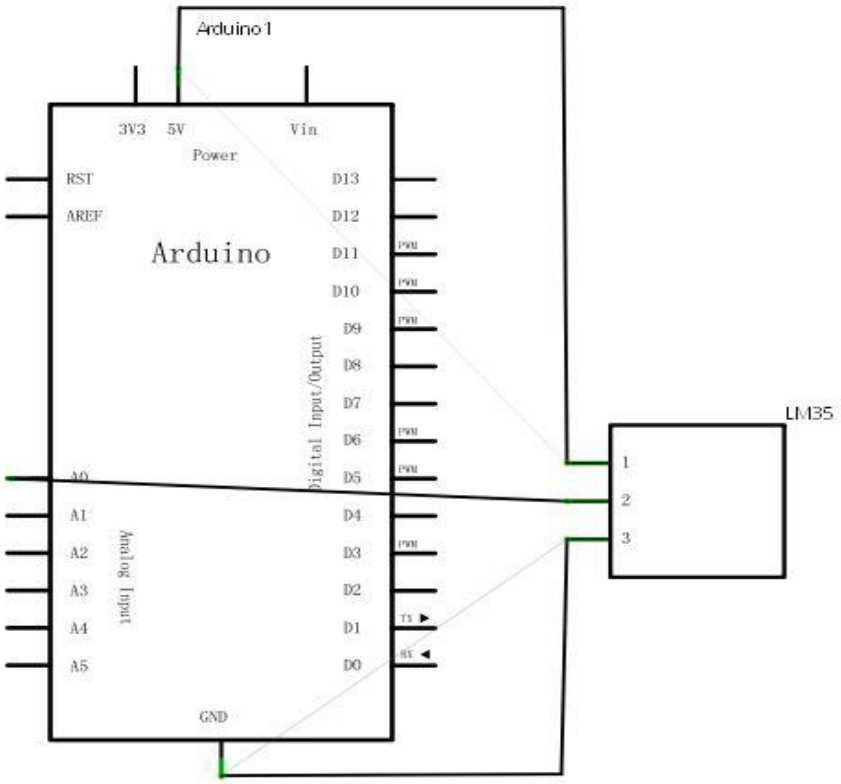
LM35 is a common and easy-to-use temperature sensor component. It only need a LM35 component and an analog interface in applications. However, it is algorithmically difficult to convert the analog value into the actual temperature.

Hardware Required

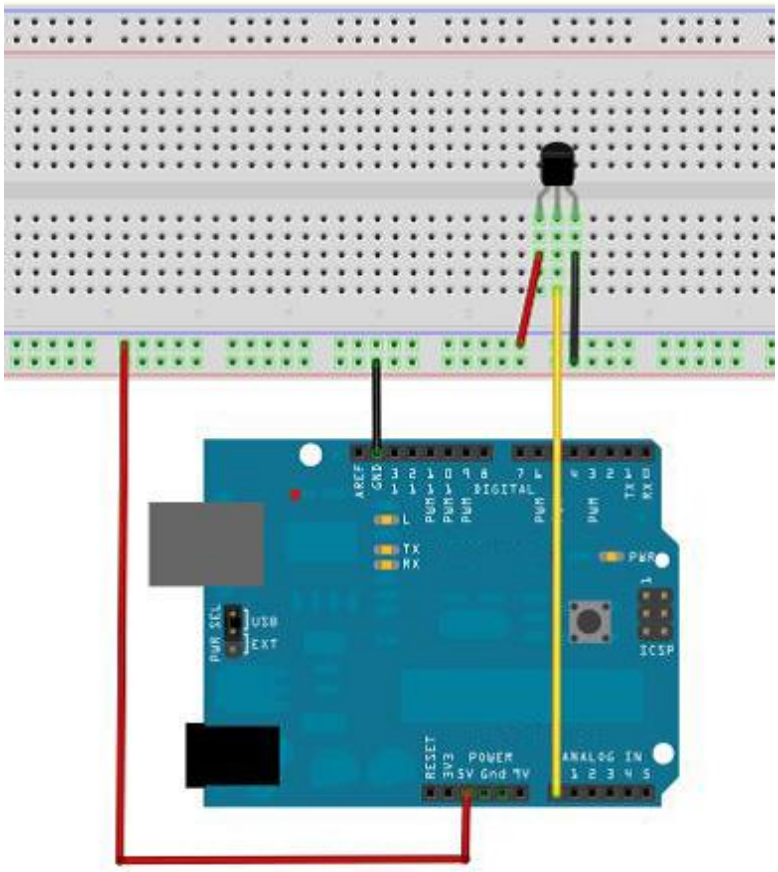
Direct Plug-in LM35 Sensor*1
Breadboard*1
Breadboard Jumper Wires*1 bunch

Schematic & Wiring Diagram

Schematic Diagram:



Connection Diagram:

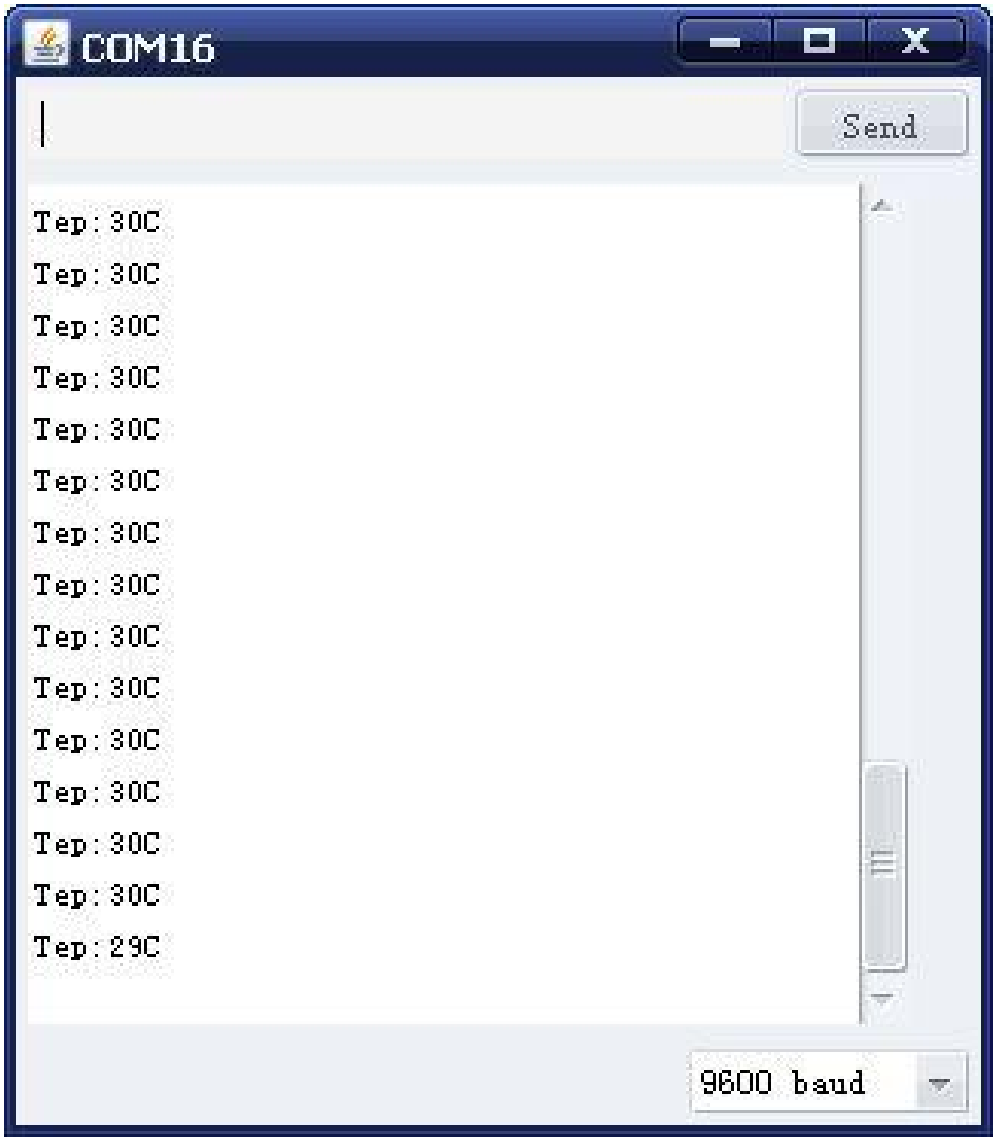


Sample Code

```
int potPin = 0; //connect analog port 0 to LM35 temperature sensor
void setup()
{
Serial.begin(9600);//set the baud rate of 9600
}
void loop()
{
int val;//define variable
int dat;//define variable
val=analogRead(0);// read the analog value of sensor and assign it to val
dat=(125*val)>>8;//temperature calculation formula
Serial.print("Tep:");//original output shows the Tep strings representing the temperature

Serial.print(dat);//outputs display dat value
Serial.println("C");//original output shows C strings
delay(500);//delay 0.5 second
}
```

Test Result

After uploading the program, open the serial monitor to get the current temperature as the figure shown below.



Lesson 14: Tilt Switch

Introduction

Using tilt switch to control the LED light's on or off

Hardware Required

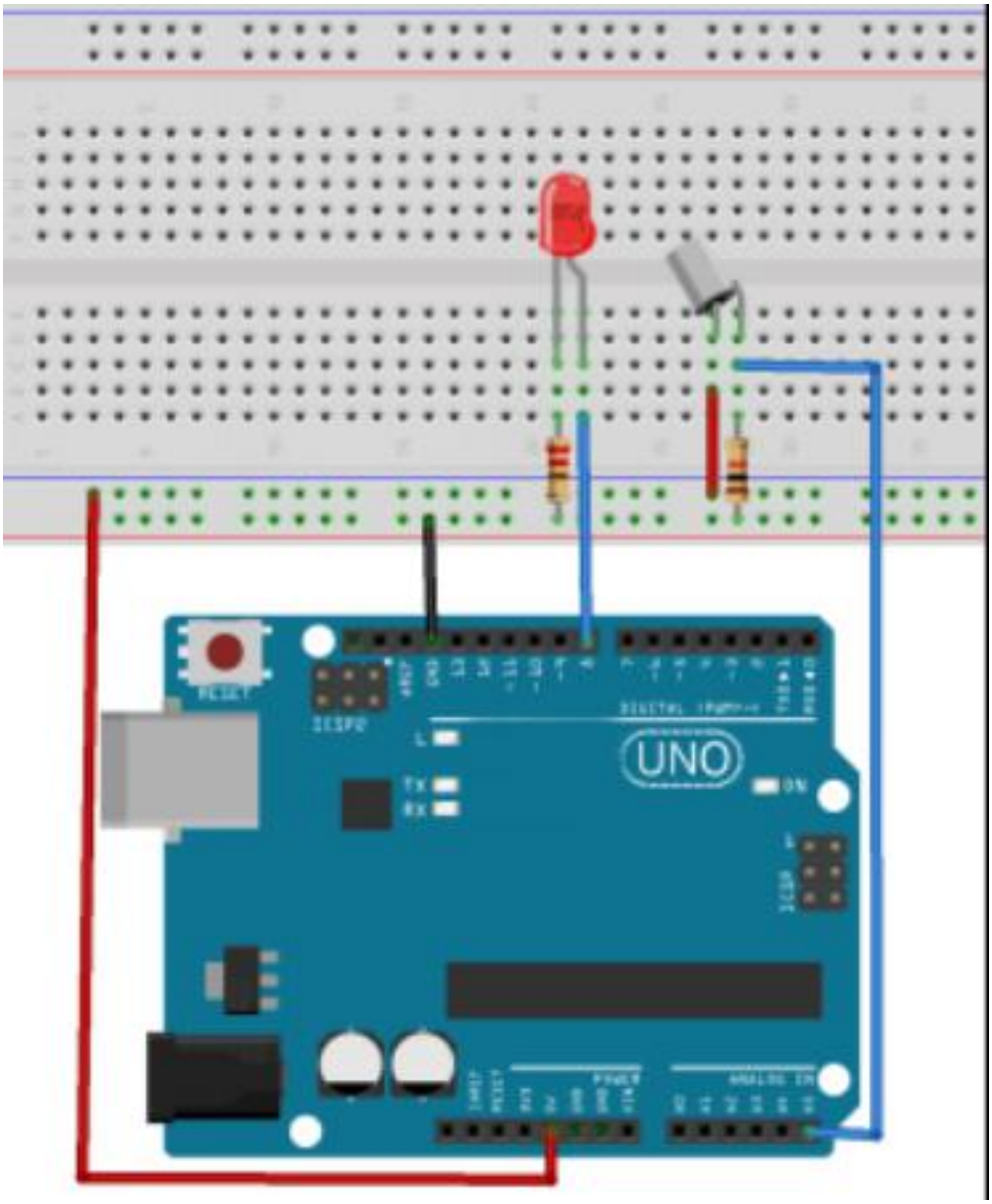
Ball Switch *1

Led Light *1

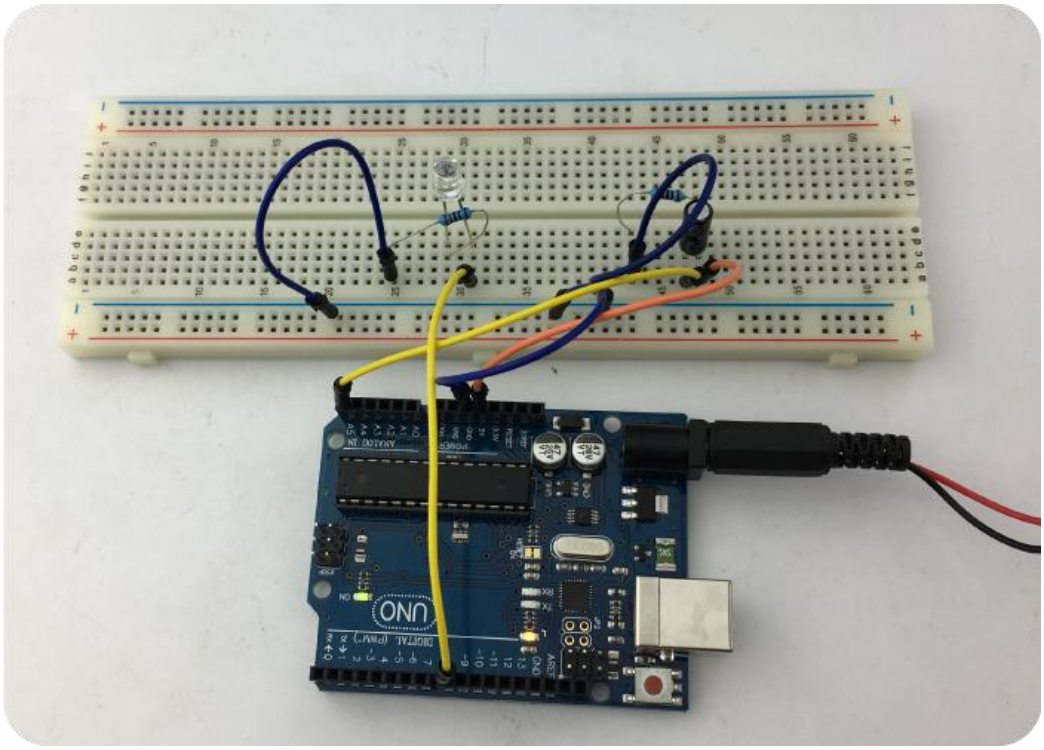
220Ω Resistor *1

Colorful Breadboard Jumper Wires*Several

Wiring Diagram



Physical Connection Diagram:

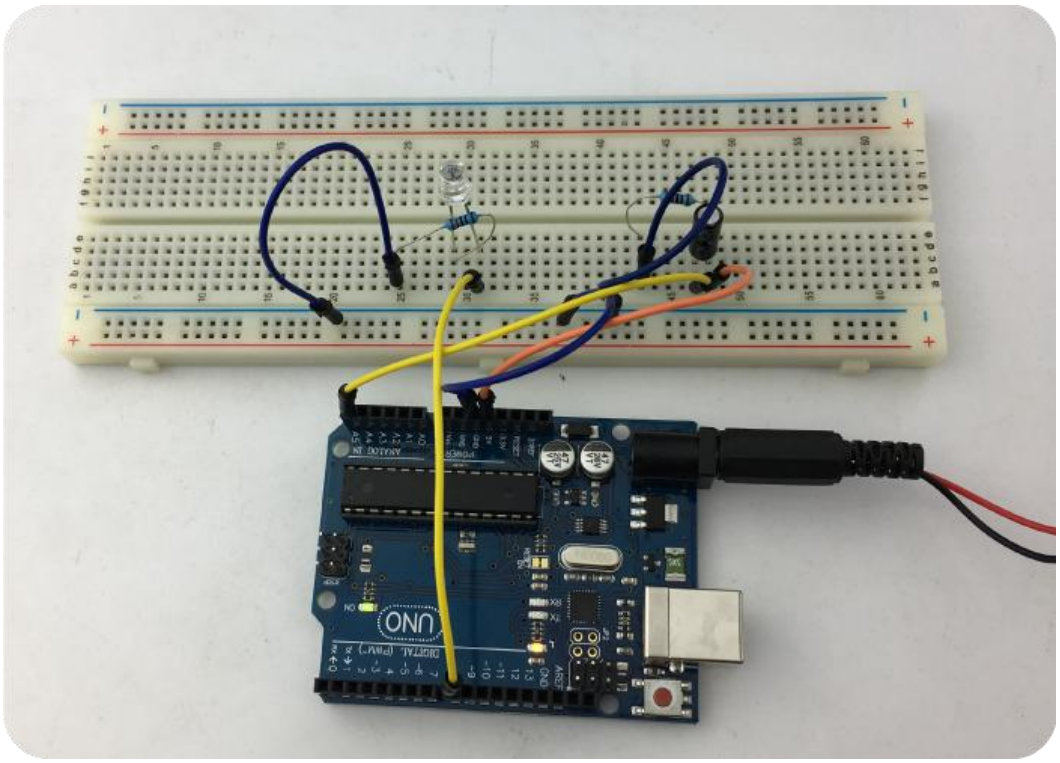


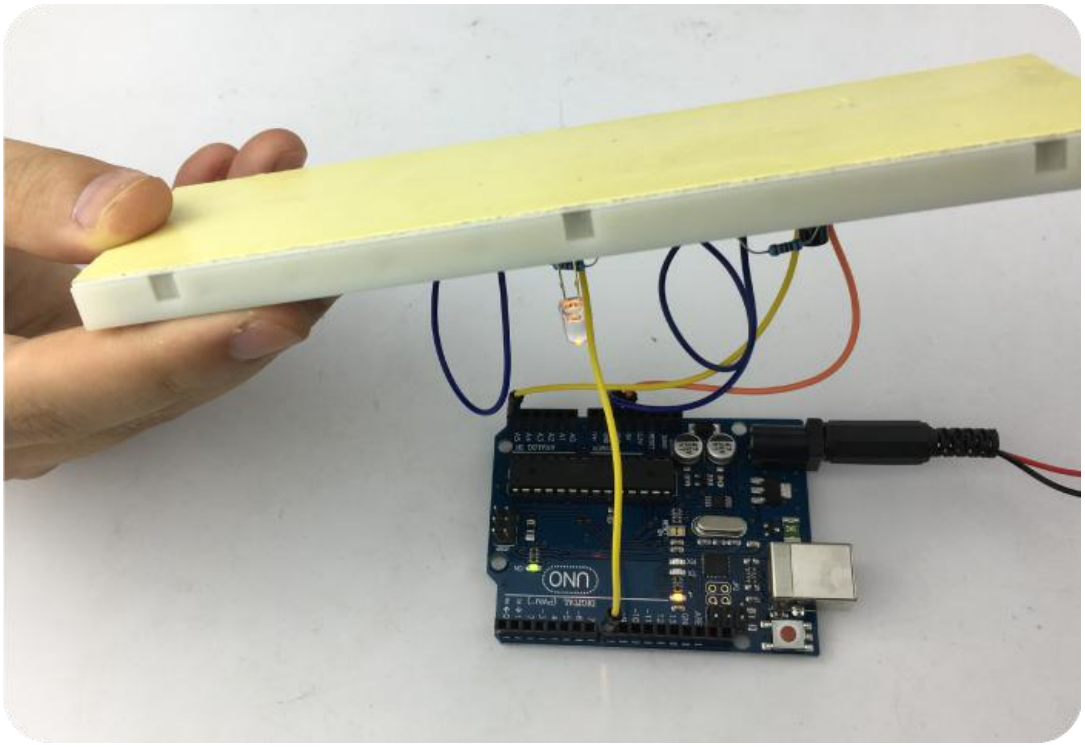
Sample Code

```
 void setup()
{
   pinMode(8,OUTPUT);//set the digit 8 pin as output mode
}
void loop()
{
int i;//define i
while(1)
{
   i=analogRead(5);//read the volt value of analog port 5
   if(i>512)//if greater than 512（2.5V）
   {
      digitalWrite(8,LOW);//turn on led light
   }
   else//otherwise
   {
      digitalWrite(8,HIGH);//put out led light
   }
 }
}
```

Test Result

With the breadboard in hand, tilt it to a certain extent, the LED light is on. If not tilted, the led is off. You can refer to the two figures below:





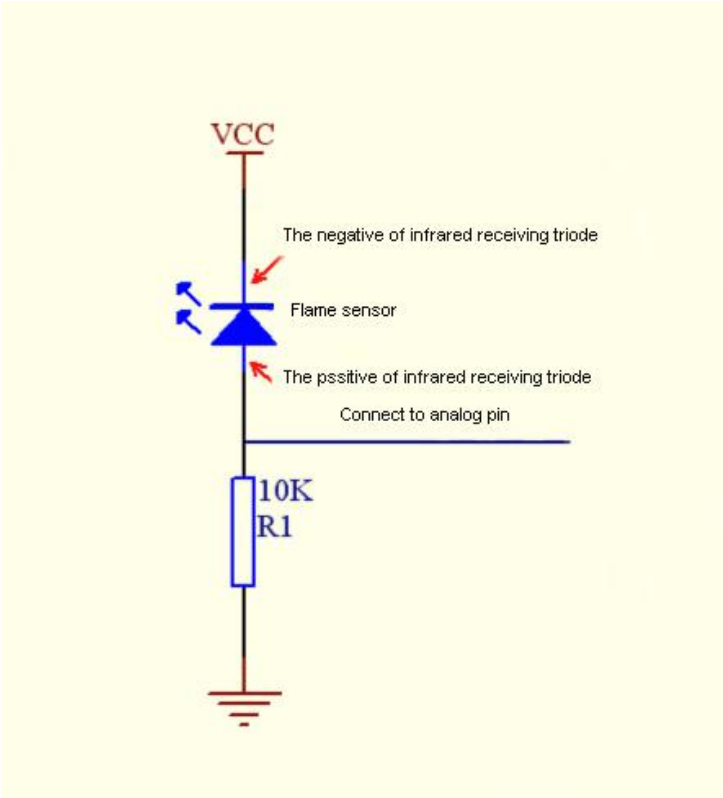
Lesson 15: Flame Alarm

Introduction

Flame sensor (namely infrared receiving triode) is specifically designed for robot to search for fire source. This sensor is particularly sensitive to the flame.

The short lead terminals of the IR receiving triode are negative end and its long lead terminal is positive end. First, connect the negative end to 5V port; then connect the positive end to the 10K resistor. The other end of the resistor is connected to the GND port. Finally, connect one end of jumper wire to the positive end of the flame sensor, and the other end is connected to the analog port.

Wiring as the figure shown below:



This lesson is to simulate flame alarm by using a flame sensor.

Hardware Required

Flame Sensor *1

LED* 1

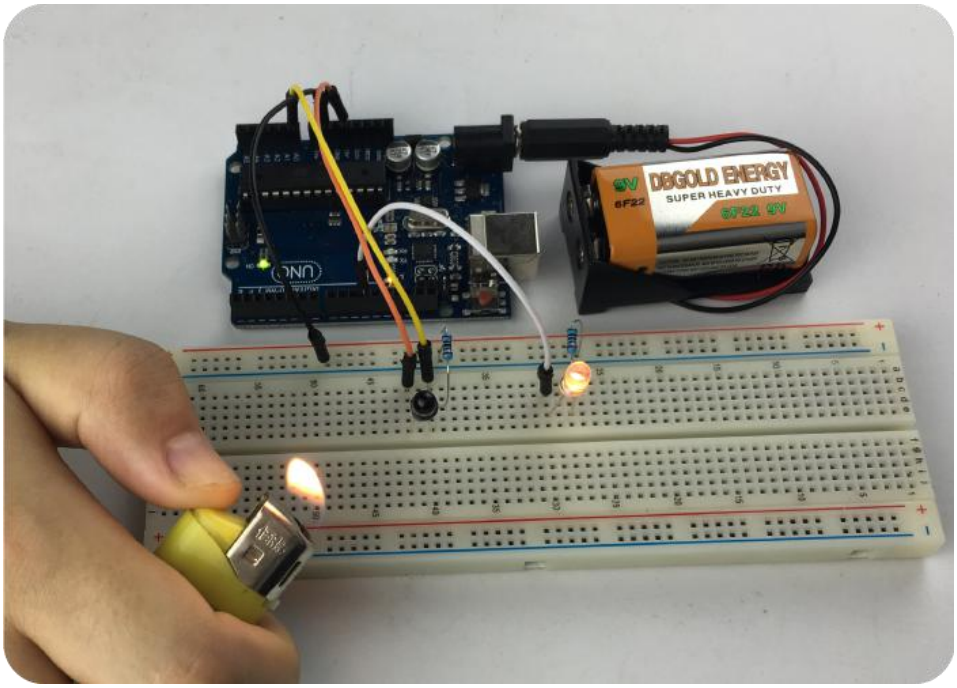
10K Resistor*1

220Ω Resistor*1

Colorful Breadboard Jumper Wires*Several

Wiring Diagram

Remove the flame sensor from the test box. According to the wiring description in the first part, connect flame sensor to the analog port A0； and the LED is connected to pin D11.



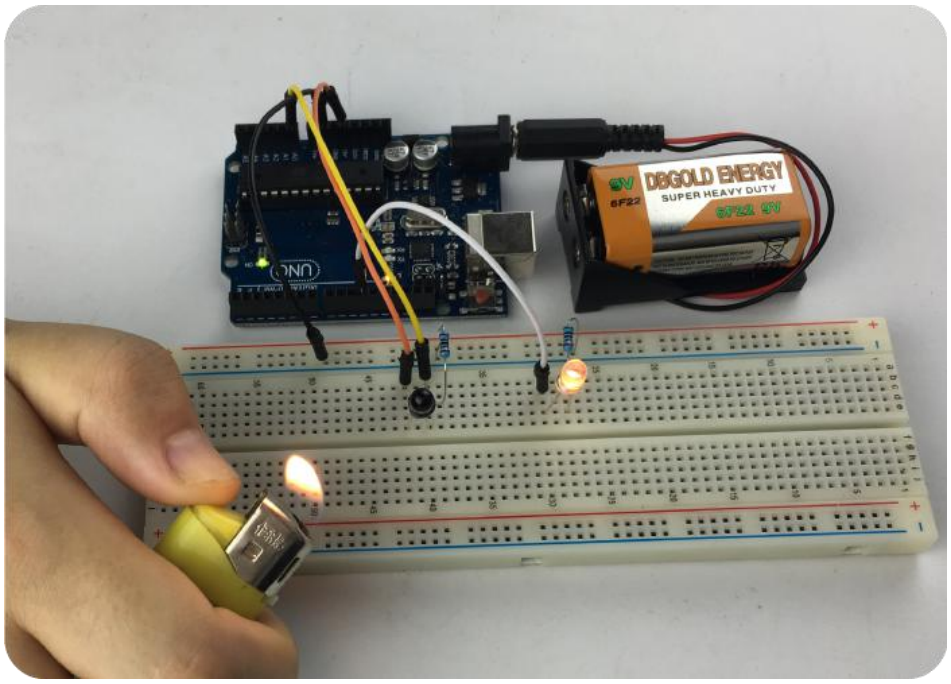
Sample Code

```
int flame=A0;//define flame interface as analog port A0
  int led=11;//define led interface as D11
  int val=0;//define digital variable
  void setup()
{
    pinMode(led,OUTPUT);//set LED as output port
  pinMode(flame,INPUT);//set the flame as input port
  Serial.begin(9600);//set the baud rate of 9600
  }
void loop()
{
    val=analogRead(flame);//read the analog value of flame sensor
    Serial.println(val);//output the analog value and print it out
    if(val>600)//if analog value is greater than 600, the led is on.
    {
      digitalWrite(led,HIGH);
     }else
     {
        digitalWrite(led,LOW);
      }
}
```

Test Result:

This program simulates the phenomenon of alarm in case of flame. In the test, you can see

the LED is on when there is a flame.



Lesson 16: One-bit LED Segment Display

Introduction

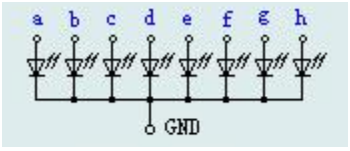
LED segment display is a common component displaying the digit, you can find it in our daily life like induction cooker, automatic washing machine, solar water temperature display, electronic clock and so on. So it is really important to know the display rule of LED segment display. LED segment display is a semiconductor display component whose basic unit is a light emitting diode (LED). It is divided into seven-segment display and eight-segment display. Compared with seven-seg display, the eight-segment display just has one more light-emitting diode unit (more than a decimal point display).

According to the connection method of light-emitting diode unit, it can be divided into common anode display and common cathode display. Common anode display means that the anode of all light-emitting diodes are connected together to form a common anode (COM). And it should connect the COM to +5V when using the common anode display. When the cathode of light-emitting diode for one numeric field is in low level, the corresponding numeric field is lighted up. If it is in a high level, the corresponding numeric field is off.

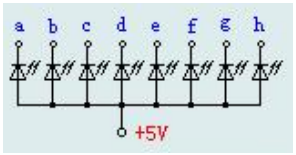
Common cathode display means that the cathode of all light-emitting diodes are connected together to form a common cathode (COM). And it should connect the COM to ground(GND) when using the common cathode display. When the anode of light-emitting diode for one numeric field is in high level, the corresponding numeric field is lighted up. If it is in a low level, the corresponding numeric field is off.

Each segment of the digital display is composed of a light-emitting diode, so it needs to

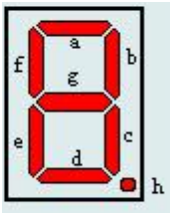
connect the current limiting resistor, otherwise the current is too large to burn the LED. This test will use a COM cathode eight-segment display.



COM Cathode Eight-segment Display



COM Anode Eight-segment Display



Eight-segment LED Display

Hardware Required

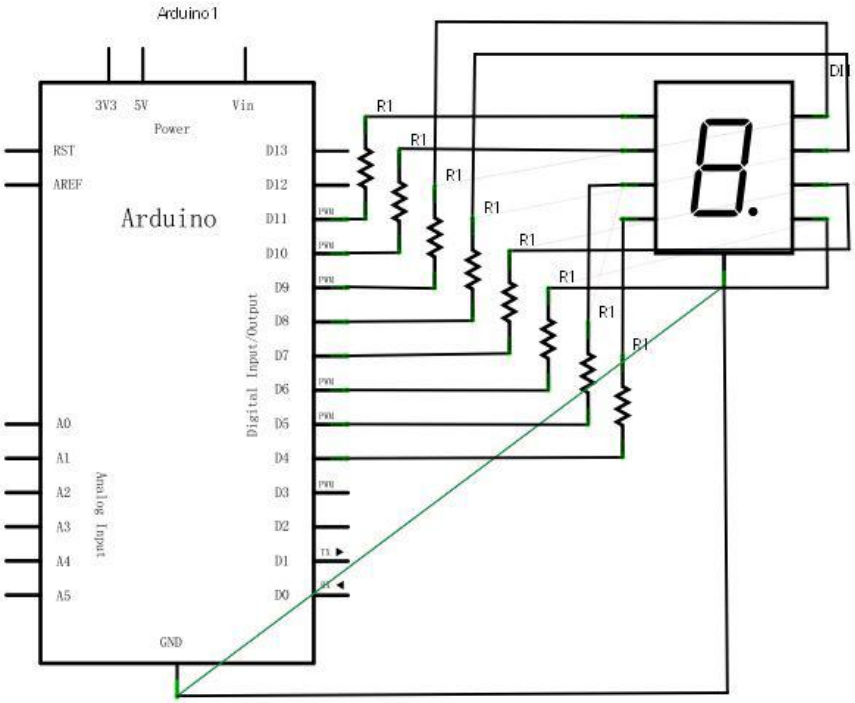
Eight-segment LED Display*1

220Ω Direct Plug-in Resistor *8

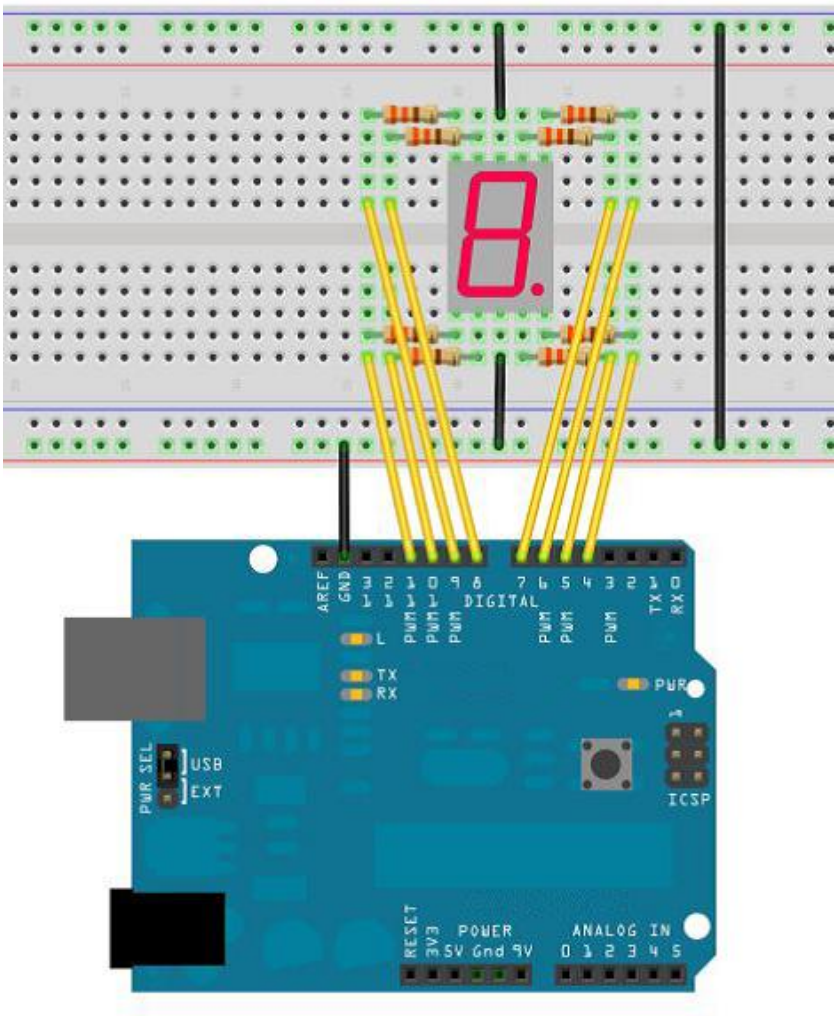
Colorful Breadboard Jumper Wires*1 bunch

Schematic & Wiring Diagram

Schematic Diagram:



Wiring Diagram:



Sample Code

```
//set the digital IO pin of segment control
int a=7;//connect the digital port 7 to a segment of LED display
int b=6;// connect the digital port 6 to b segment
int c=5;// connect the digital port 5 to c segment
int d=10;// connect the digital port 10 to d segment
int e=11;// connect the digital port 11 to e segment
int f=8;// connect the digital port 8 to f segment
int g=9;// connect the digital port 9 to g segment
int dp=4;// connect the digital port 6 to dp segment
void digital_0(void) //display digit 5
{
unsigned char j;
digitalWrite(a,HIGH);
digitalWrite(b,HIGH);
digitalWrite(c,HIGH);
```

```
digitalWrite(d,HIGH);
digitalWrite(e,HIGH);
digitalWrite(f,HIGH);
digitalWrite(g,LOW);
digitalWrite(dp,LOW);
}
void digital_1(void) //display digit 1
{
unsigned char j;
digitalWrite(c,HIGH);//digital port 5 pin of high level, to light up c segment
digitalWrite(b,HIGH);//light up b segment
for(j=7;j<=11;j++)//put out the balance segment
digitalWrite(j,LOW);
digitalWrite(dp,LOW);//put out decimal point DP segment
}
void digital_2(void) //display digit 2
{
unsigned char j;
digitalWrite(b,HIGH);
digitalWrite(a,HIGH);
for(j=9;j<=11;j++)
digitalWrite(j,HIGH);
digitalWrite(dp,LOW);
digitalWrite(c,LOW);
digitalWrite(f,LOW);
}
void digital_3(void) //display digit 3
{
digitalWrite(g,HIGH);
digitalWrite(a,HIGH);
digitalWrite(b,HIGH);
digitalWrite(c,HIGH);
digitalWrite(d,HIGH);
digitalWrite(dp,LOW);
digitalWrite(f,LOW);
digitalWrite(e,LOW);
}
void digital_4(void) //display digit 4
{
digitalWrite(c,HIGH);
digitalWrite(b,HIGH);
digitalWrite(f,HIGH);
digitalWrite(g,HIGH);
digitalWrite(dp,LOW);
```

```
digitalWrite(a,LOW);
digitalWrite(e,LOW);
digitalWrite(d,LOW);
}
void digital_5(void) //display digit 5
{
unsigned char j;
digitalWrite(a,HIGH);
digitalWrite(b, LOW);
digitalWrite(c,HIGH);
digitalWrite(d,HIGH);
digitalWrite(e, LOW);
digitalWrite(f,HIGH);
digitalWrite(g,HIGH);
digitalWrite(dp,LOW);
}
void digital_6(void) //display digit 6
{
unsigned char j;
for(j=7;j<=11;j++)
digitalWrite(j,HIGH);
digitalWrite(c,HIGH);
digitalWrite(dp,LOW);
digitalWrite(b,LOW);
}
void digital_7(void) //display the digit 7
{
unsigned char j;
for(j=5;j<=7;j++)
digitalWrite(j,HIGH);
digitalWrite(dp,LOW);
for(j=8;j<=11;j++)
digitalWrite(j,LOW);
}
void digital_8(void) //display digit 8
{
unsigned char j;
for(j=5;j<=11;j++)
digitalWrite(j,HIGH);
digitalWrite(dp,LOW);
}
void digital_9(void) //display digit 9
{
unsigned char j;
```

```
digitalWrite(a,HIGH);
digitalWrite(b,HIGH);
digitalWrite(c,HIGH);
digitalWrite(d,HIGH);
digitalWrite(e, LOW);
digitalWrite(f,HIGH);
digitalWrite(g,HIGH);
digitalWrite(dp,LOW);
}
void setup()
{
int i;//define variable
for(i=4;i<=11;i++)
pinMode(i,OUTPUT);//set 4～11pin as output mode
}
void loop()
{
while(1)
{
digital_0();//display digit 0
delay(1000);//delay 1s
digital_1();//display digit 1
delay(1000);//delay 1s
digital_2();//display digit 2
delay(1000); //delay 1s
digital_3();//display digit 3
delay(1000); //delay 1s
digital_4();//display digit 4
delay(1000); //delay 1s
digital_5();//display digit 5
delay(1000); //delay 1s
digital_6();//display digit 6
delay(1000); //delay 1s
digital_7();//display digit 7
delay(1000); //delay 1s
digital_8();//display digit 8
delay(1000); //delay 1s
digital_9();//display digit 9
delay(1000); //delay 1s
}
}
```

Test Result

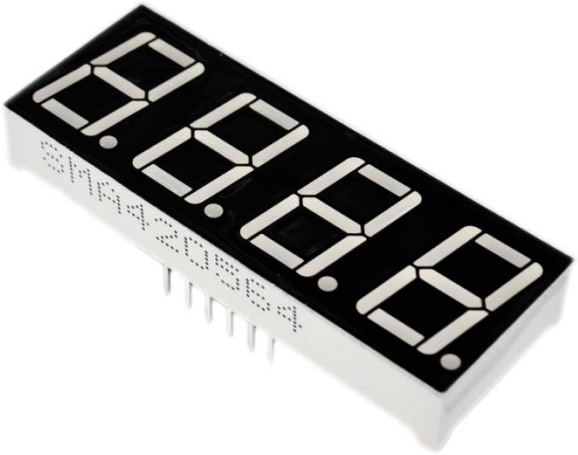
The LED display will circularly display the digit 0-9.

Lesson 17: Four-bit LED Segment Display

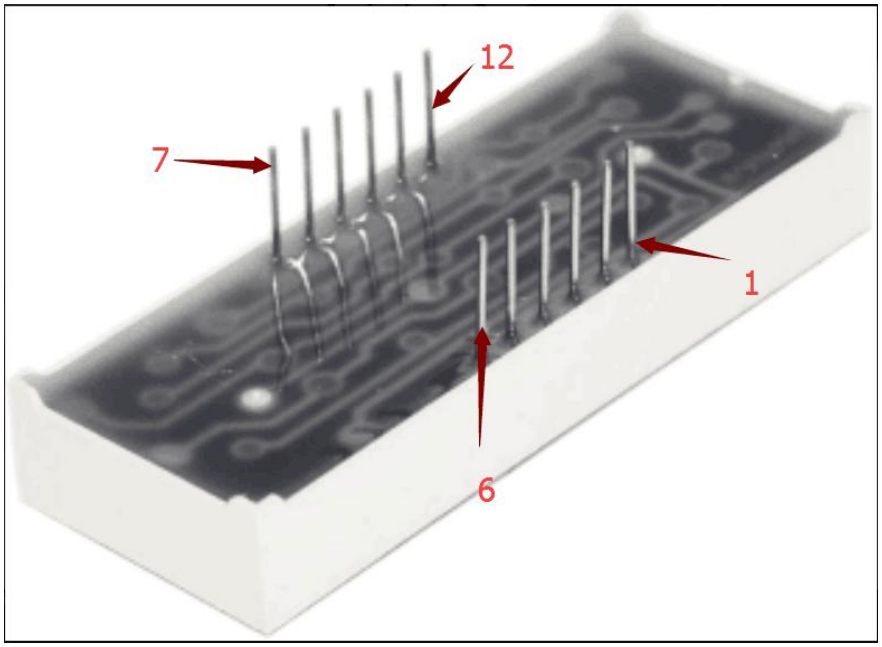
Introduction

In this experiment, we use arduino to drive a common cathode four-bit led segment display. It is very essential to use the current limiting resistor when driving the display. There are two connection methods of current limiting resistor. One is to connect total four current limiting resistors to D1-D4 anode end, using less resistors but the display brightness is uneven from the different digits.Another method is to connect to other eight pins. In this way, the display brightness is even but using more resistors.

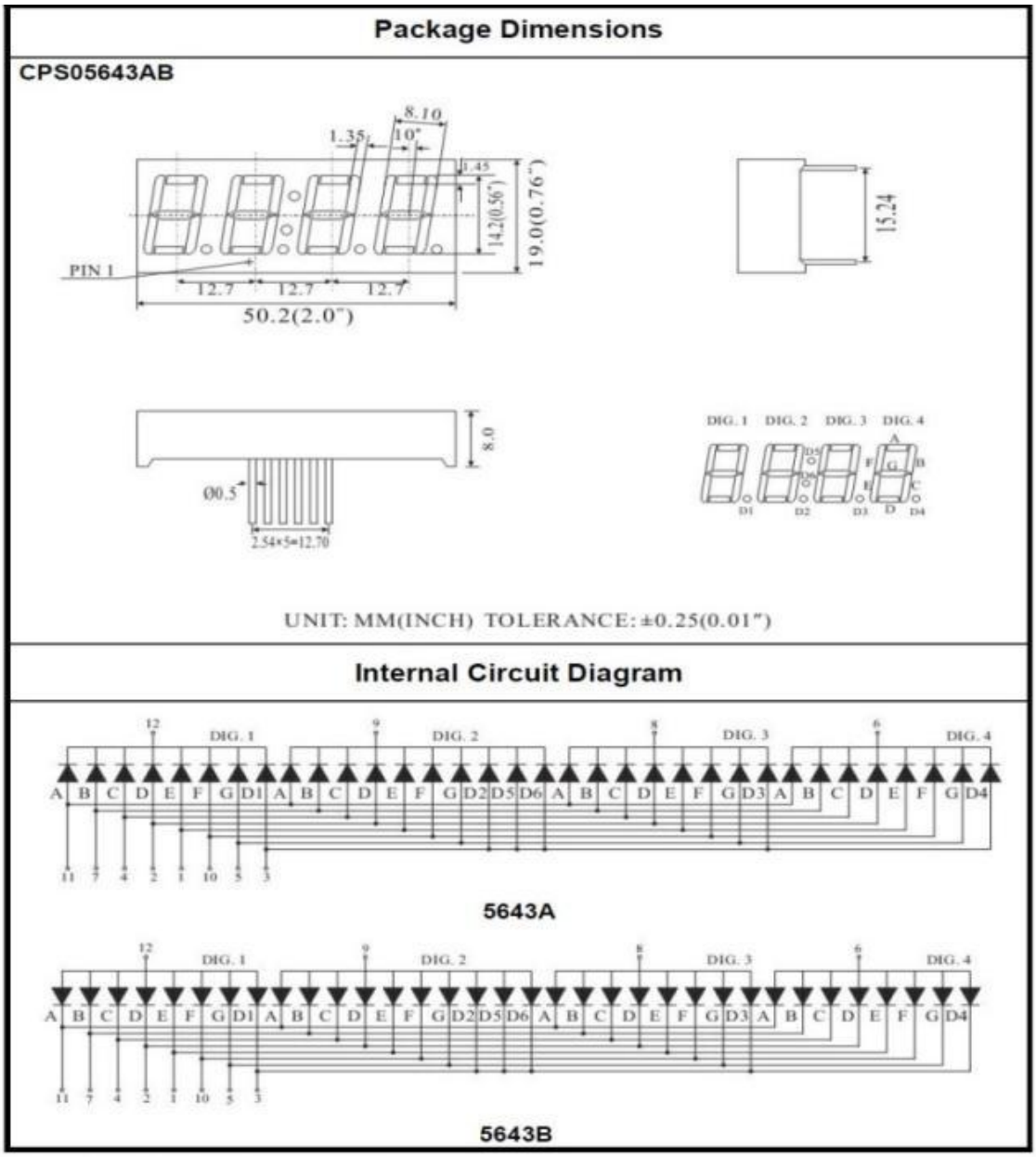
This experiment uses eight 220Ωresistors (we use 220Ωresistor because no 100Ωresistor is available. If you use 100Ω,the display will be brighter).



The four-bit LED display has a total of 12 pins. When the decimal point is placed in front of you, its lower left corner is the digit 1, and other pin's order is for the counterclockwise rotation. Upper left corner is the largest digit 12 pin. Please refer to the following figure:



LED Segment Display Manual:



Four Digits Displays Series

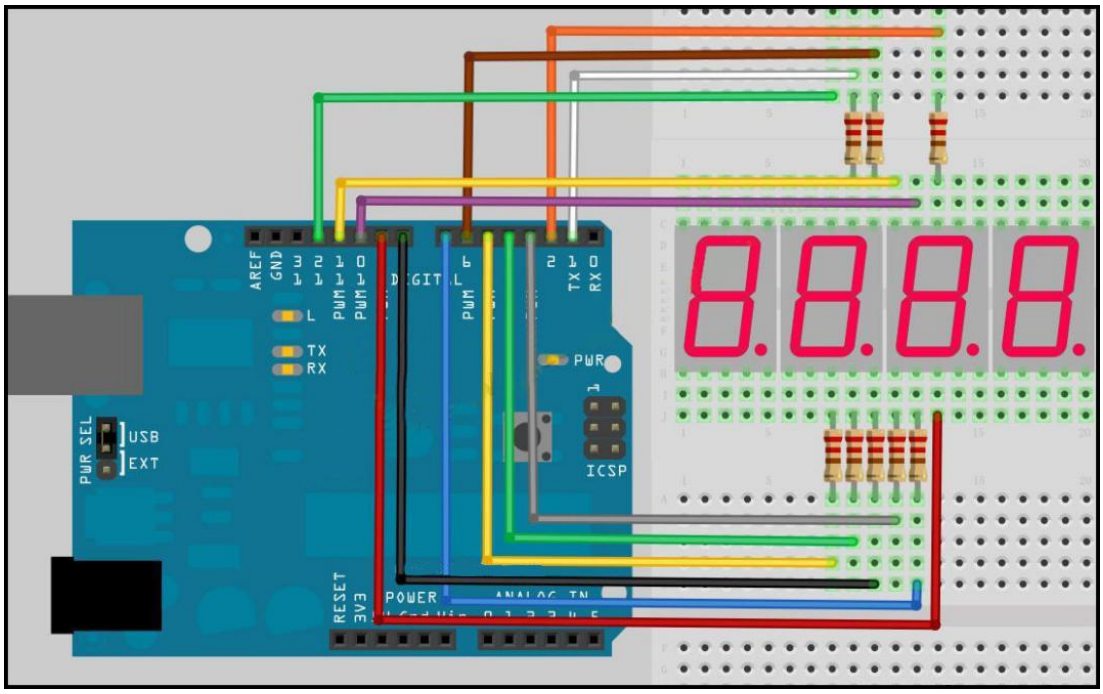
Hardware Required

Four-bit LED Segment Display*1

220Ω Resistor*8

Breadboard Jumper Wire*1 bunch

Wiring Diagram



Sample Code

```
//display 1234
      //set the cathode port
      int a = 1;
      int b = 2;
      int c = 3;
      int d = 4;
      int e = 5;
      int f = 6;
      int g = 7;
      int dp = 8;
      //set the anode port
      int d4 = 9;
      int d3 = 10;
      int d2 = 11;
      int d1 = 12;
      //set the variable
      long n = 1230;
      int x = 100;
      int del = 55;    //the digital value is for the clock fine-tune

      void setup()
```

```
    {
      pinMode(d1, OUTPUT);
      pinMode(d2, OUTPUT);
      pinMode(d3, OUTPUT);
      pinMode(d4, OUTPUT);
      pinMode(a, OUTPUT);
      pinMode(b, OUTPUT);
      pinMode(c, OUTPUT);
      pinMode(d, OUTPUT);
      pinMode(e, OUTPUT);
      pinMode(f, OUTPUT);
      pinMode(g, OUTPUT);
      pinMode(dp, OUTPUT);
    }
////////////////////////////////////////////////////////////
void loop()
{
 Display(1, 1);
 Display(2, 2);
 Display(3, 3);
 Display(4, 4);

}
////////////////////////////////////////////////////////////
void WeiXuan(unsigned char n)//
{
    switch(n)
     {
    case 1:
      digitalWrite(d1,LOW);
      digitalWrite(d2, HIGH);
      digitalWrite(d3, HIGH);
      digitalWrite(d4, HIGH);
     break;
     case 2:
      digitalWrite(d1, HIGH);
      digitalWrite(d2, LOW);
      digitalWrite(d3, HIGH);
      digitalWrite(d4, HIGH);
         break;
      case 3:
         digitalWrite(d1,HIGH);
       digitalWrite(d2, HIGH);
       digitalWrite(d3, LOW);
```

```
            digitalWrite(d4, HIGH);
              break;
           case 4:
            digitalWrite(d1, HIGH);
            digitalWrite(d2, HIGH);
            digitalWrite(d3, HIGH);
            digitalWrite(d4, LOW);
              break;
              default :
                   digitalWrite(d1, HIGH);
            digitalWrite(d2, HIGH);
            digitalWrite(d3, HIGH);
            digitalWrite(d4, HIGH);
              break;
          }
}
void Num_0()
{
   digitalWrite(a, HIGH);
   digitalWrite(b, HIGH);
   digitalWrite(c, HIGH);
   digitalWrite(d, HIGH);
   digitalWrite(e, HIGH);
   digitalWrite(f, HIGH);
   digitalWrite(g, LOW);
   digitalWrite(dp,LOW);
}
void Num_1()
{
   digitalWrite(a, LOW);
   digitalWrite(b, HIGH);
   digitalWrite(c, HIGH);
   digitalWrite(d, LOW);
   digitalWrite(e, LOW);
   digitalWrite(f, LOW);
   digitalWrite(g, LOW);
   digitalWrite(dp,LOW);
}
void Num_2()
{
   digitalWrite(a, HIGH);
   digitalWrite(b, HIGH);
   digitalWrite(c, LOW);
   digitalWrite(d, HIGH);
```

```
  digitalWrite(e, HIGH);
  digitalWrite(f, LOW);
  digitalWrite(g, HIGH);
  digitalWrite(dp,LOW);
}
void Num_3()
{
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, LOW);
  digitalWrite(f, LOW);
  digitalWrite(g, HIGH);
  digitalWrite(dp,LOW);
}
void Num_4()
{
  digitalWrite(a, LOW);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, LOW);
  digitalWrite(e, LOW);
  digitalWrite(f, HIGH);
  digitalWrite(g, HIGH);
  digitalWrite(dp,LOW);
}
void Num_5()
{
  digitalWrite(a, HIGH);
  digitalWrite(b, LOW);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, LOW);
  digitalWrite(f, HIGH);
  digitalWrite(g, HIGH);
  digitalWrite(dp,LOW);
}
void Num_6()
{
  digitalWrite(a, HIGH);
  digitalWrite(b, LOW);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
```

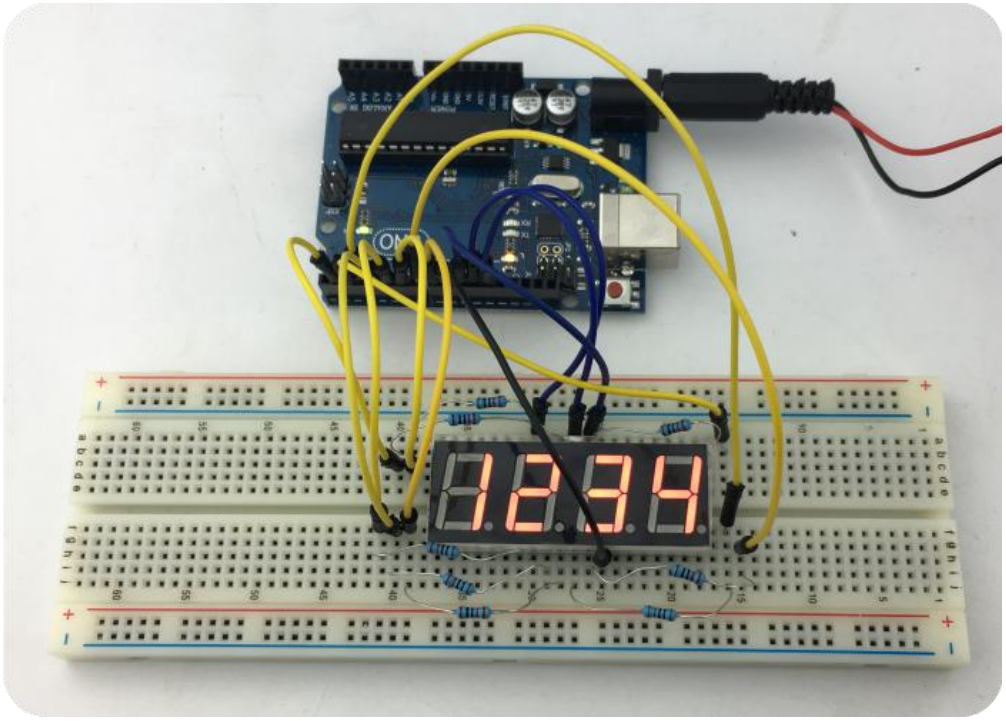
```
    digitalWrite(e, HIGH);
    digitalWrite(f, HIGH);
    digitalWrite(g, HIGH);
    digitalWrite(dp,LOW);
}
void Num_7()
{
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, LOW);
    digitalWrite(e, LOW);
    digitalWrite(f, LOW);
    digitalWrite(g, LOW);
    digitalWrite(dp,LOW);
}
void Num_8()
{
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, HIGH);
    digitalWrite(f, HIGH);
    digitalWrite(g, HIGH);
    digitalWrite(dp,LOW);
}
void Num_9()
{
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, LOW);
    digitalWrite(f, HIGH);
    digitalWrite(g, HIGH);
    digitalWrite(dp,LOW);
}
void Clear()    //clear the screen
{
    digitalWrite(a, LOW);
    digitalWrite(b, LOW);
    digitalWrite(c, LOW);
    digitalWrite(d, LOW);
```

```
  digitalWrite(e, LOW);
  digitalWrite(f, LOW);
  digitalWrite(g, LOW);
  digitalWrite(dp,LOW);
}
void pickNumber(unsigned char n)//select the number
{
  switch(n)
  {
    case 0:Num_0();
    break;
    case 1:Num_1();
    break;
    case 2:Num_2();
    break;
    case 3:Num_3();
    break;
    case 4:Num_4();
    break;
    case 5:Num_5();
    break;
    case 6:Num_6();
    break;
    case 7:Num_7();
    break;
    case 8:Num_8();
    break;
    case 9:Num_9();
    break;
    default:Clear();
    break;
  }
}
void Display(unsigned char x, unsigned char Number)//display x as coordinate
{
  WeiXuan(x);
  pickNumber(Number);
 delay(1);
 Clear() ; //clear the shadow
}
```

Test Result

After downloading the code to control board, the LED display shows the digit 1234 as the

figure shown below:



NOTE: Be careful to connect the wire; please check the circuit when appearing messy code. Thank you!

Lesson 18: 74HC595 Test

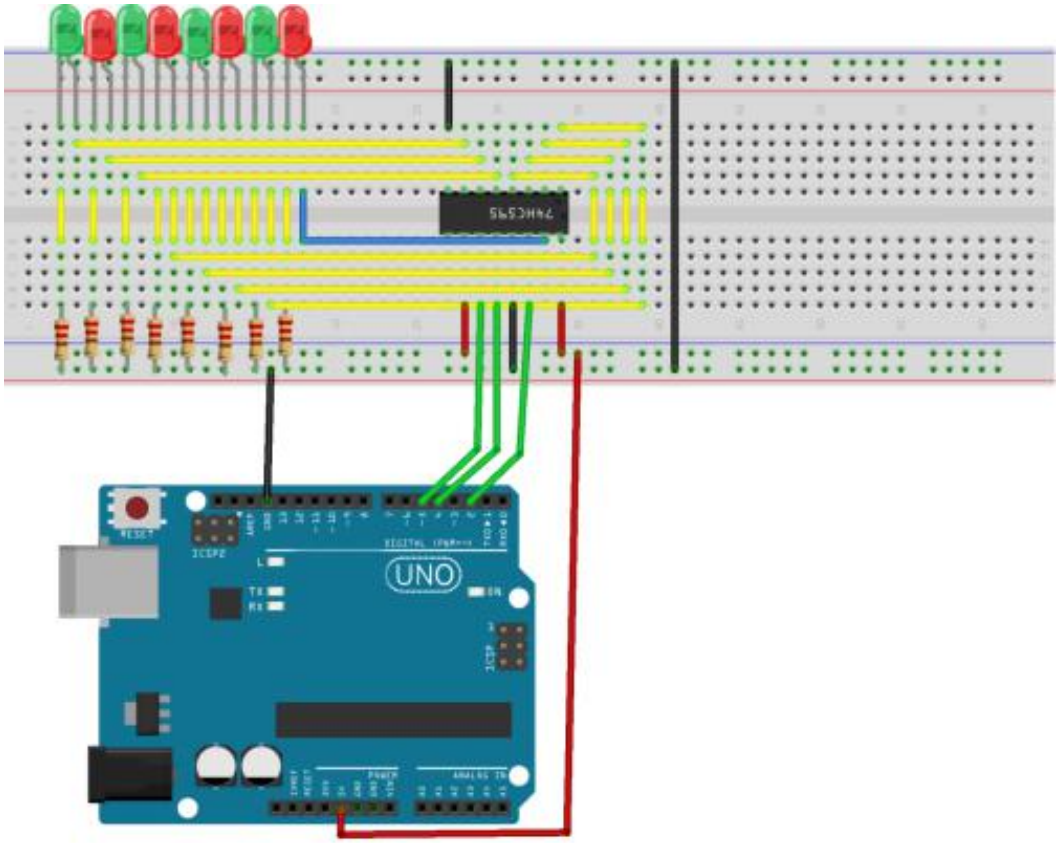
Introduction

The 74HC595 simply comes with an 8-bit shift register and a memory, as well as a tri-state output function. Here we use it to control eight LED lights. So why? Maybe a lot of friends will ask this question. I would like to ask is that if we simply use Arduino to control 8 LED lights, then how many I/O ports we needed? The answer is eight, but our Arduino 168's I/O ports plus the analog interfaces are only total in 20. Since these eight LED lights take up too much ports, so using 74HC595 is to save I/O port. With 74HC595 later we can use three digital I/O ports to control 8 LED lights.

Hardware Required

74HC595 Direct Plug-in Chip*1
Red M5 Direct Plug-in LED*4
Green M5 Direct Plug-in LED*4
220Ω Direct Plug-in Resistance*8
Breadboard*1
Breadboard Jumper Wires*1 bunch

Wiring Diagram



Sample Code

```
int data = 2;//14 pin of 74HC595, data input pin SI
int clock = 5;//11 pin of 74hc595,clock line SCK
int latch = 4;//12 pin of 74hc595, output memory lock line RCK
int ledState = 0;
const int ON = HIGH;
const int OFF = LOW;
void setup()
{
pinMode(data, OUTPUT);
pinMode(clock, OUTPUT);
pinMode(latch, OUTPUT);
}
void loop()
{
for(int i = 0; i < 256; i++)
{
```

```
updateLEDs(i);
delay(500);
}
}
void updateLEDs(int value)
{
digitalWrite(latch, LOW);//
shiftOut(data, clock, MSBFIRST, ~value);//serial data output, high data first.
digitalWrite(latch, HIGH);//latch
}
```

Test Result

After programming, you can see the beautiful picture of 8 LED lights flashing.
It is actually eight LED display octets binary number, cycling to add more one.

Lesson 19: RGB Module

Introduction

There are three colors of LED light on the RGB module, that is, red, green and blue. The driving voltage on each color of LED light is different, so their brightness is also different, and they are combined together to form a variety of colors. We can control three groups of signal output to achieve the full color effect mixed with R, G, B three colors .

Hardware Required

UNO R3 *1
USB Cable*1
RGB Module*1
Dupont Wire*Several

Wiring Diagram

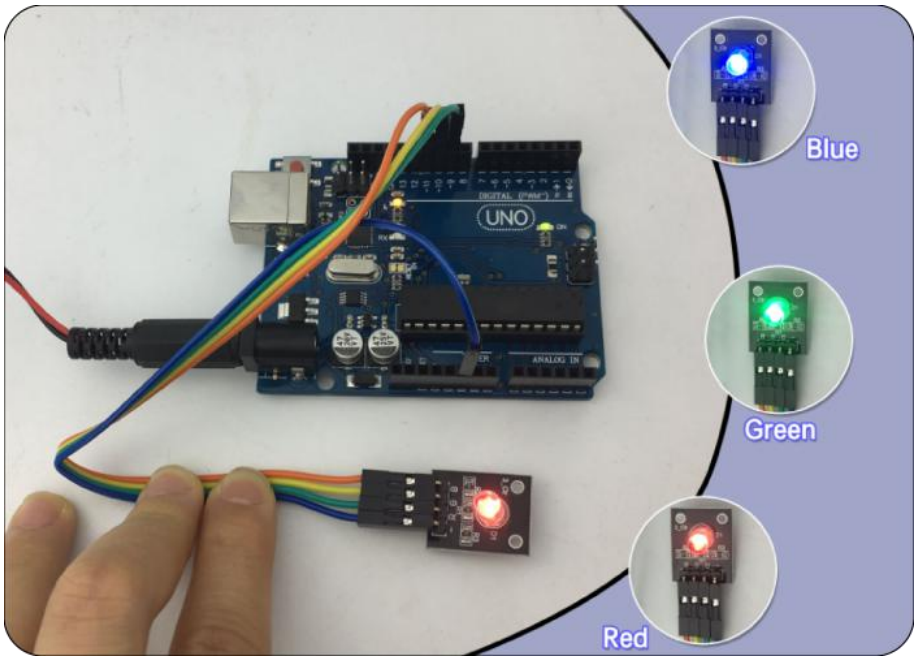
RGB Module	UNO R3
-	GND
R	D9
G	D10
B	D11

Sample Code

```
int redPin = 9;    // Red LED control pin is connected to Arduino's 9 pin
int greenPin = 10;    // Green LED control pin is connected to Arduino's 10 pin
int bluePin = 11;    // Blue LED control pin is connected to Arduino's 11 pin
void setup()
{
 pinMode(redPin, OUTPUT); //set the corresponding 9 pin of redPin as output
 pinMode(greenPin, OUTPUT); //set the corresponding 10 pin of greenPin as output
pinMode(bluePin, OUTPUT); //set the corresponding 11 pin of bluePin as output
}

void loop()    // run over and over again
{
            // Basic colors:
            color(255, 0, 0); // red LED is on
            delay(1000); // delay one second
            color(0,255, 0); //green LED is on
            delay(1000); //delay one second
            color(0, 0, 255); // blue LED is on
            delay(1000); //delay one second

            // Example blended colors:
            color(255,255,0); // yellow
            delay(1000); //delay one second
            color(255,255,255); // white
            delay(1000); //delay one second
            color(128,0,255); // purple
            delay(1000); //delay one second
            color(0,0,0); // t turn off led
            delay(1000); //delay one second
}

void color (unsigned char red, unsigned char green, unsigned char blue)    //color control
function
{
        analogWrite(redPin, red);
        analogWrite(greenPin, green);
        analogWrite(bluePin, blue);
}
```

Test Result

The LED light on the RGB module loops a variety of colors. The figure shown below is the combination of blue and green led:



Lesson 20: IR Remote Controller Decoding

Introduction

The signal sent out by Infrared remote controller is a series of binary pulse codes. In order to avoid interference from other infrared signals in the process of wireless transmission, it is usually firstly modulated in a specific carrier frequency, and then sent out by the infrared emission diode. The infrared receiver will filter out other clutters, only to receive the specific frequency signal and restore it into a binary pulse code, namely, demodulation. This experiment will use an IR remote controller and an IR receiver to decode the infrared remote control.

Hardware Required

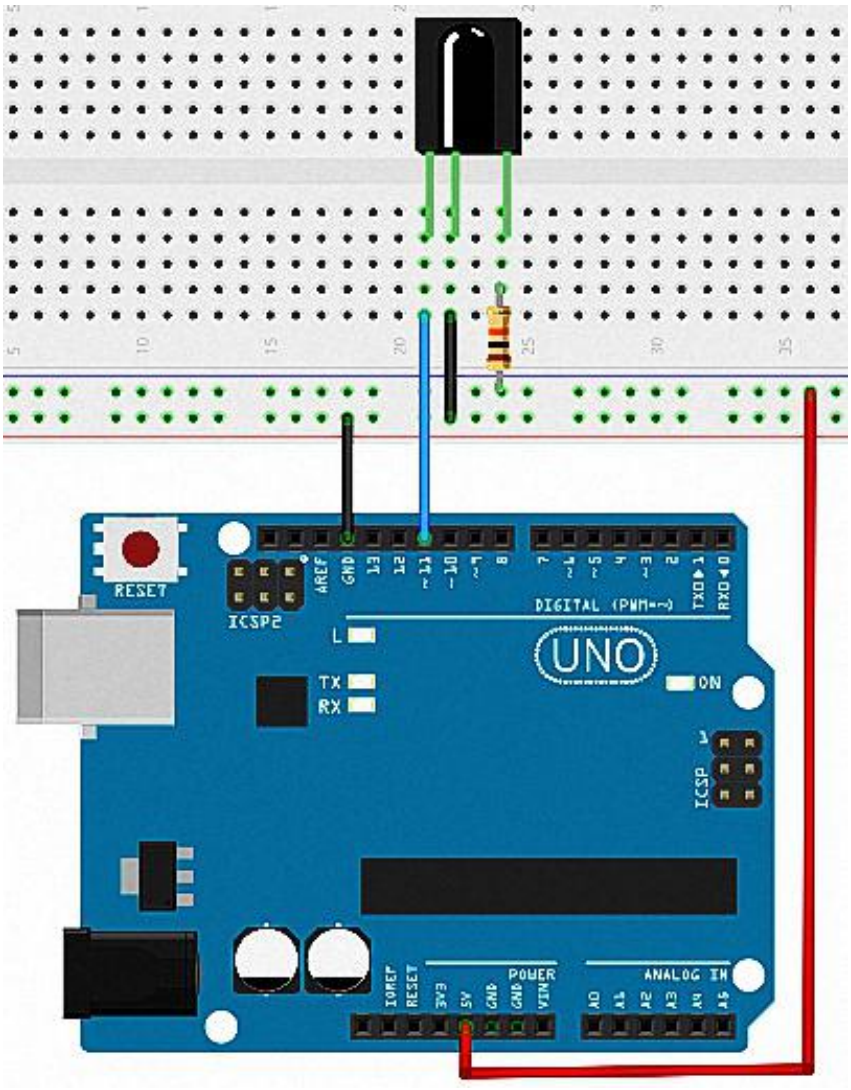
IR Remote Controller*1

IR Receiver*1

10K Resistor*1

Colorful Breadboard Wires* Several

Wiring Diagram



Sample Code

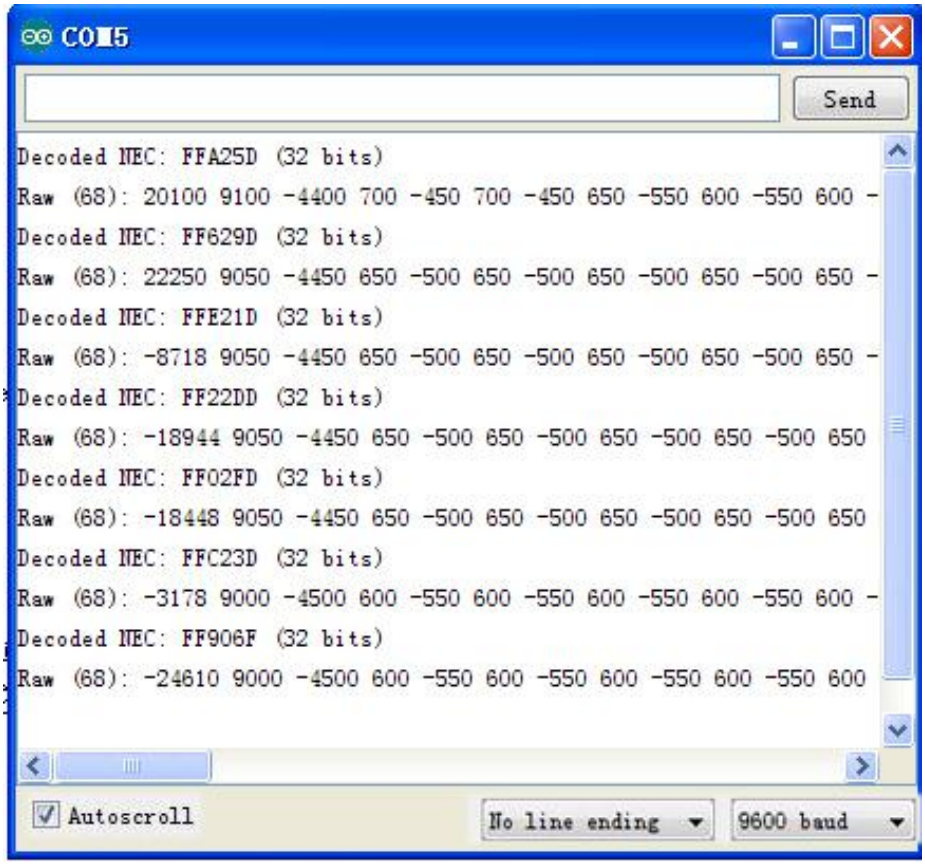
```
#include <IRremote.h>
int RECV_PIN = 11; //define input pin on Arduino
IRrecv irrecv(RECV_PIN);
decode_results results;
void setup()
```

```
{
Serial.begin(9600);
irrecv.enableIRIn(); // Start the receiver
}
void loop() {
if (irrecv.decode(&results)) {
Serial.println(results.value, HEX);
irrecv.resume(); // Receive the next value
}
}
```

Note: in program B, put the IRremote folder into the \Arduino\libraries of compiler installation directory. Otherwise, it will appear the compiling error.
For example：C:\Program Files\Arduino\libraries

Test Result

Decode the code pulse transmitted from the remote controller, and execute the corresponding action according to the decoding result. In this way, you can use the remote controller to control your device, letting it listen to your command.

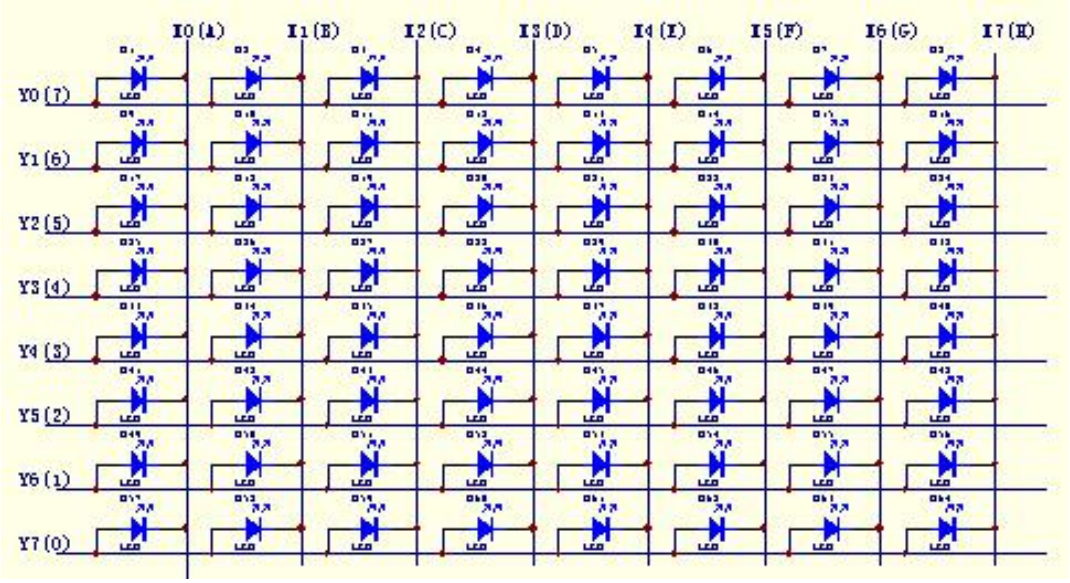


Lesson 21: 8*8 Dot Matrix

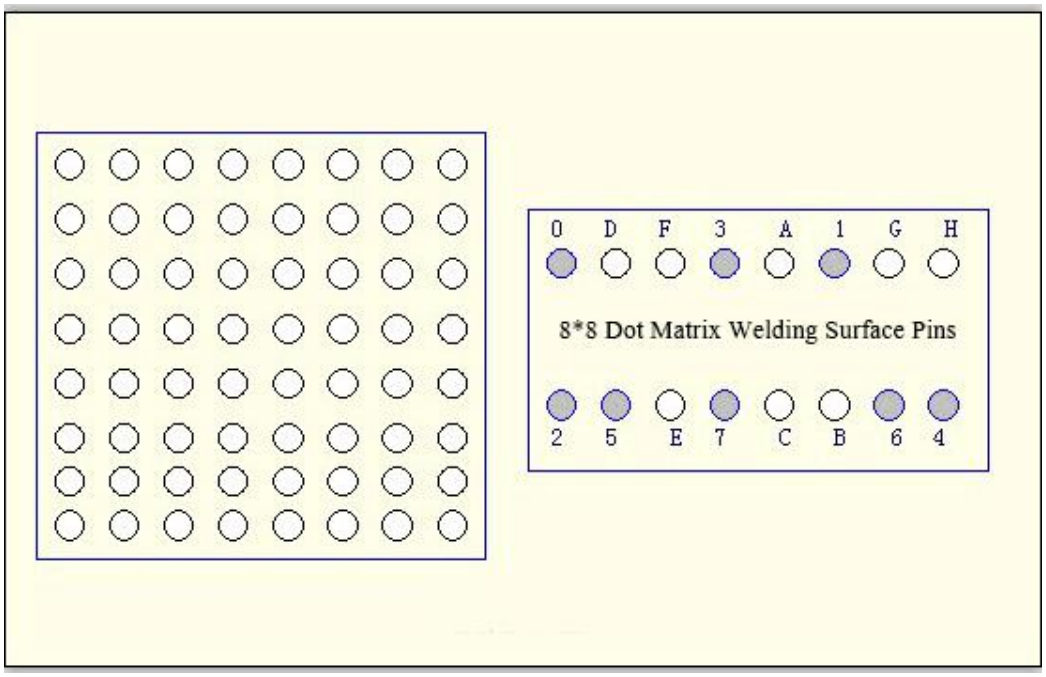
Introduction

It is very common to see the dot matrix in our daily life, such as LED advertising display, elevator display floor, public newspaper station and so on. In this lesson,we will do a simple test about 8*8 dot matrix.

Schematic Diagram of 8*8 Dot Matrix:



Physical Map of 8*8 Dot Matrix:



Hardware Required

Uno R3 *1

USB Cable*1

8*8 Dot Matrix*1

220Ω Direct Plug-in Resistor*8

Dupont Wire*Several

Wiring Diagram

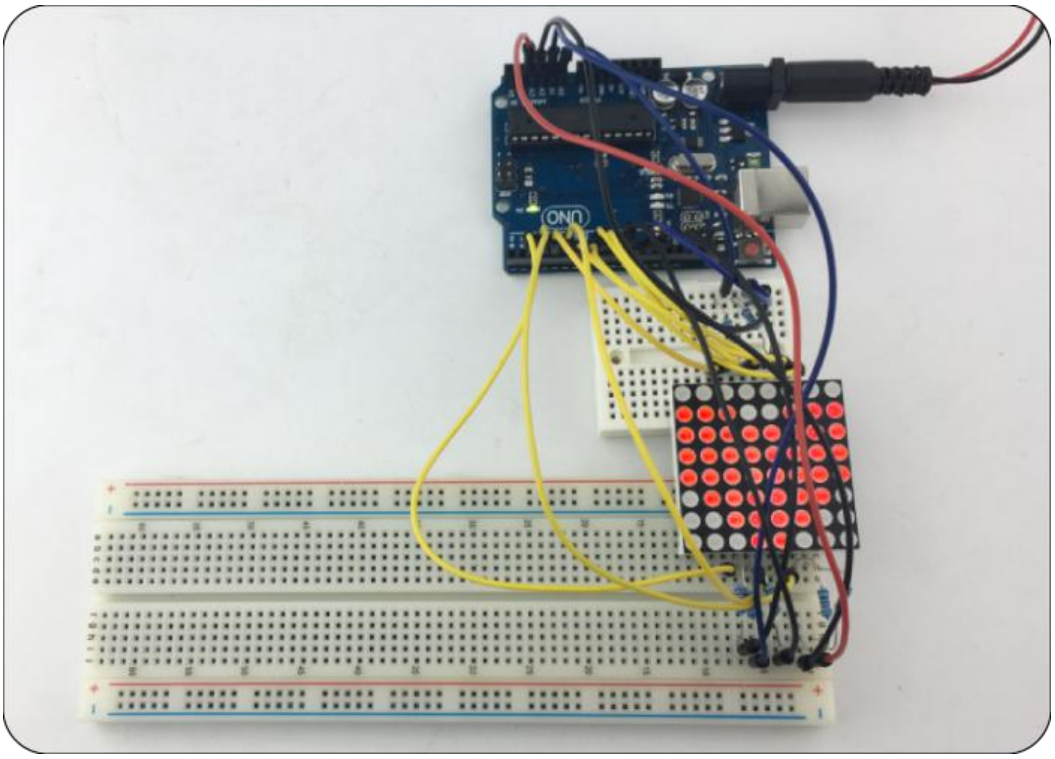
8*8 Dot Matrix	Uno R3
0	D2
1	D3
2	D4
3	D5
4	D6
5	D7
6	D8
7	D9
A	D10
B	D11
C	D12
D	D13
E	A1(D14)
F	A2(D15)
G	A3(D16)
H	A4(D17)

Sample Code

```
//define an array that holds the font of the heart pattern.
unsigned char Text[]={0x78,0x7c,0x7e,0x3f,0x3f,0x7e,0x7c,0x78};
void Draw_point(unsigned char x,unsigned char y)//draw function
{
    clear_();
    digitalWrite(x+2, HIGH);
    digitalWrite(y+10, LOW);
    delayMicroseconds(100);
}
void show_num(void)//display the function, calling the draw function.
{
  unsigned char i,j,data;
  for(i=0;i<8;i++)
  {
     data=Text[i];
     for(j=0;j<8;j++)
     {
        if(data & 0x01)Draw_point(j,i);
        data>>=1;
     }
  }
}
void setup(){
int i = 0 ;
for(i=2;i<18;i++)
 {
    pinMode(i, OUTPUT);
  }
  clear_();
}
void loop()
{
  show_num();
}
void clear_(void)//clear the screen
{
  for(int i=2;i<10;i++)
  digitalWrite(i, LOW);
  for(int i=0;i<8;i++)
  digitalWrite(i+10, HIGH);
}
```

Test Result

After programming and powered on, the 8*8 dot matrix will display the heart pattern as the figure shown below:



Heart Pattern Setting:

●	●	●			●	●	●
●	●	●	●	●	●	●	●
●	●	●	●	●	●	●	●
●	●	●	●	●	●	●	●
	●	●	●	●	●	●	
		●	●	●	●		
			●	●			

Set the binary value according to the above picture and then convert it to a hexadecimal value, setting it in the code.

01111000:78
01111100:7C
01111110:7E
00111111:3F
00111111:3F
01111110:7E
01111100:7C
01111000:78