TA0602 UNO Basic Starter Kit for Arduino Projects

Getting Started

1. Download the Arduino Environment (IDE) and install or unzip/extract the application directory.
2. Connect the Uno board to one of your computer's USB port.
3. Install the drivers (If the computer does not automatically download and install the necessary USB drivers, point the hardware setup to the "drivers" directory of the Arduino IDE application.)
4. Launch the Arduino IDE application
5. Open a sketch example such as "Blink"
6. Select your Board from the Tools menu.
7. Select the Serial Port used by the board
8. Upload the sketch to the board

Project Details

Project 1: Hello World

Introduction:

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

Hardware Required:
1. Arduino board x1
2. USB cable x1

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

```
int val;//define variable val
int ledpin=13;// define digital interface 13
void setup()
{
  Serial.begin(9600);// set the baud rate at 9600 to match the software
set up. When connected to a specific device, (e.g. bluetooth), the baud
rate needs to be the same with it.
  pinMode(ledpin,OUTPUT);// initialize digital pin 13 as output. When
using I/O ports on an Arduino, this kind of set up is always needed.
}
void loop()
{
  val=Serial.read();// read the instruction or character from PC to
Arduino, and assign them to Val.
  if(val=='R')// determine if the instruction or character received is
"R".
```

```
  {  // if it's "R",
    digitalWrite(ledpin,HIGH);// set the LED on digital pin 13 on.
    delay(500);
    digitalWrite(ledpin,LOW);// set the LED on digital pin 13 off.
delay(500);
    Serial.println("Hello World!");// display"Hello World！"string.
  }
}
```

Result:
Click serial port monitor,Input R,LED 13 will blink once,PC will receive information from Arduino: Hello World
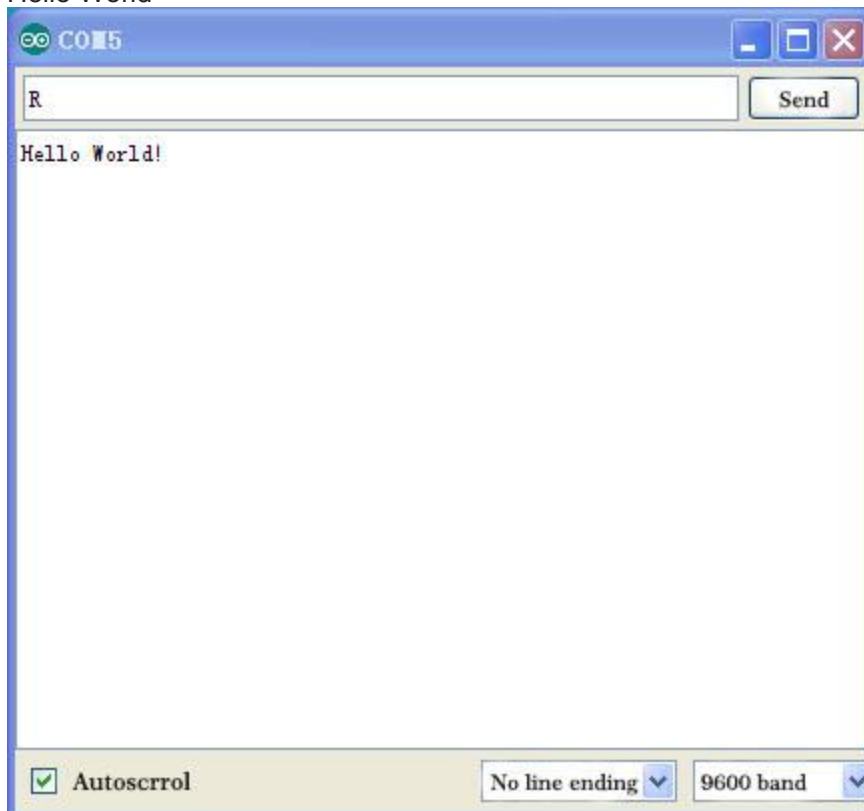
After you choose the right port, the experiment should be easy for you!

# Project 2: LED Blinking

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

Hardware Required:
1.Uno board *1
2.USB Cable *1
3. Red M5 LED*1
4. 220Ω resistor*1
5 Breadboard*1
6. 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.

Connection Diagram:
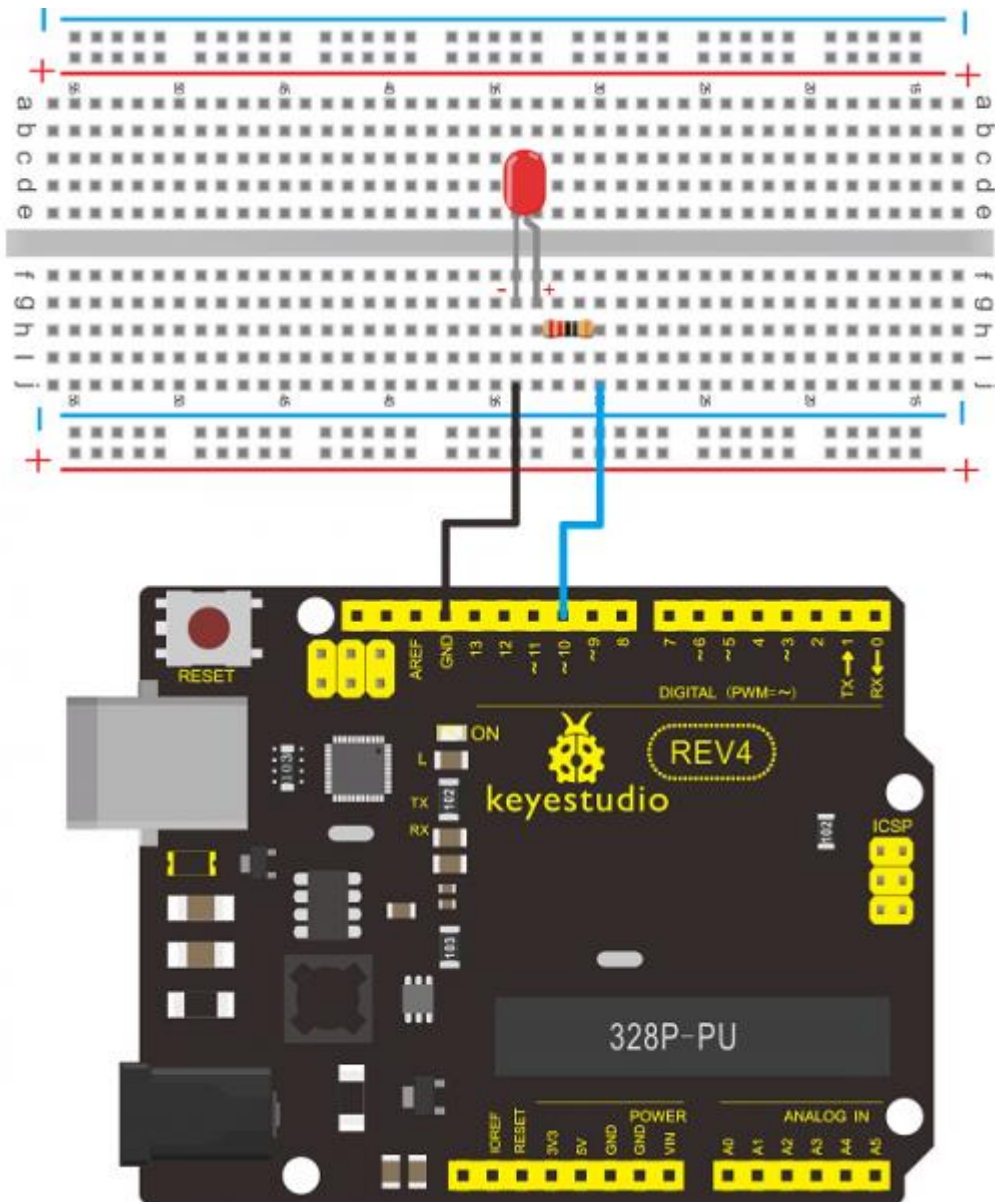
Sample Code:

According to the above circuit, you can start compiling the program, turning the LED 1 second on and 1 second off. This program is simple and similar to one that comes with Arduino except it's connected to digital pin 10.

```
int ledPin = 10; // define digital pin 10.
void setup()
{
pinMode(ledPin, OUTPUT);// define pin with LED connected as output.
}
void loop()
{
```

```
digitalWrite(ledPin, HIGH); // set the LED on.
delay(1000); // wait for a second.
digitalWrite(ledPin, LOW); // set the LED off.
delay(1000); // wait for a second
}
```

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

Project 3 : Traffic Light

Introduction:

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

Hardware Required:
1. REV4 board *1
2. USB cable *1
3. Red M5 LED*1
4. Yellow M5 LED*1
5. Green M5 LED*1
6. 220Ω resistor *3
7. Breadboard*1
8. Breadboard jumper wires* several

Connection Diagram:

With the above parts ready, we can get started. We can refer to the LED blinking experiment for this one. Below is a sample schematic diagram. Pins we use are digital pin 10, 7 and 4.
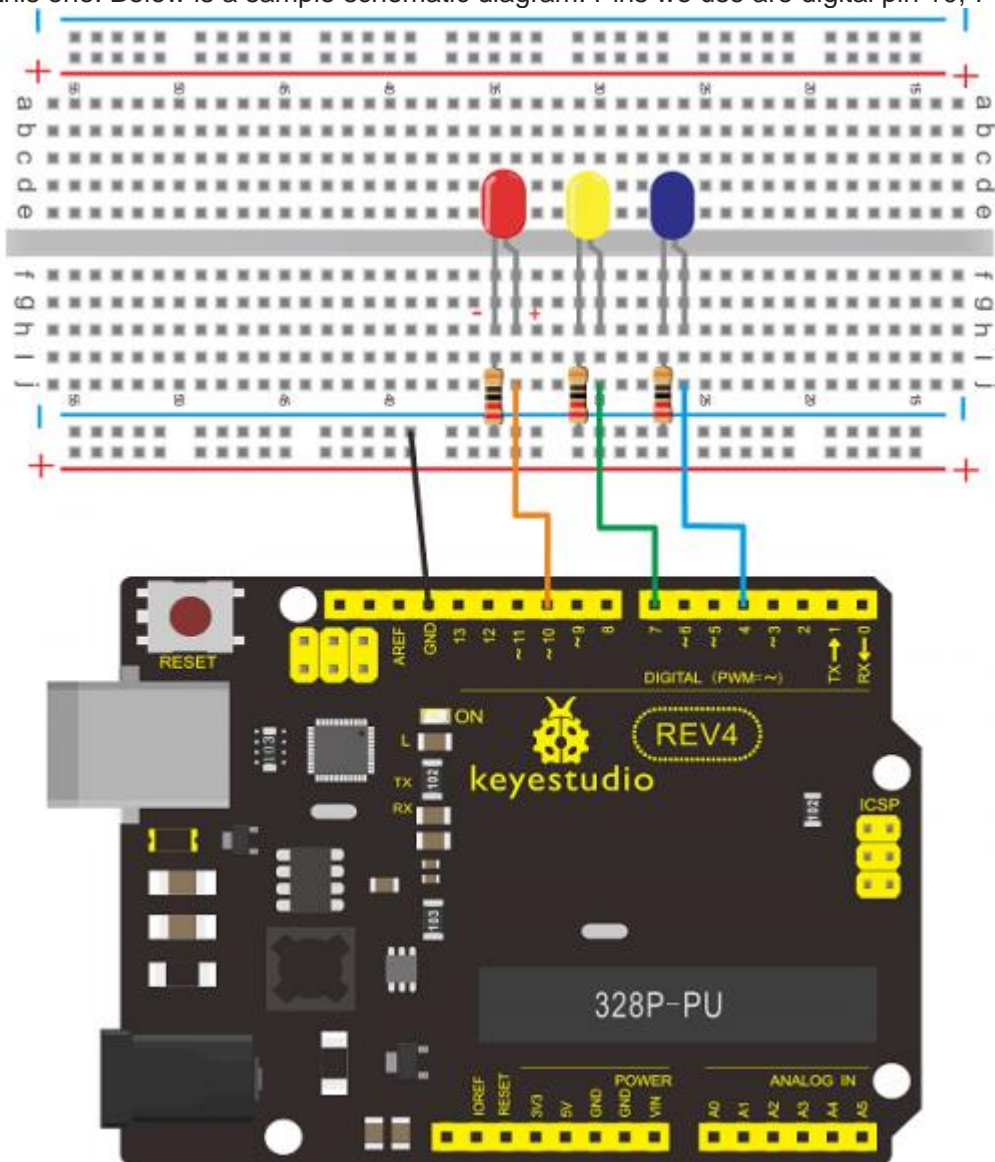
Sample Code:
Since it is a simulation of traffic lights, the blinking time of each LED should be the same with those in traffic lights system. In this program, we use Arduino delay () function to control delay time, which is much simpler than C language.

```
int redled =10; // initialize digital pin 8.
int yellowled =7; // initialize digital pin 7.
int greenled =4; // initialize digital pin 4.
void setup()
{
pinMode(redled, OUTPUT);// set the pin with red LED as "output"
pinMode(yellowled, OUTPUT); // set the pin with yellow LED as "output"
pinMode(greenled, OUTPUT); // set the pin with green LED as "output"
}
void loop()
```

```
{
digitalWrite(greenled, HIGH);//// turn on green LED

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

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

Note: this circuit design is very similar with the one in LED chase effect.
The green light will be on for 5 seconds, and then off., followed by the yellow light blinking for 3 times, and then the red light on for 5 seconds, forming a cycle. Cycle then repeats.
Experiment is now completed, thank you.

# Project 4: LED Chasing Effect

Introduction:

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

Hardware Required:

1. REV4 board *1
2. USB Cable *1
3. Red M5 LED*2 4.Yellow M5 LED*2
5. Blue M5 LED*2
6. 220Ω Resistor *6
7. Breadboard*1
8. Breadboard jumper wires* several

Connection Diagram:

According to the wiring method of diodes, six LEDs are connected to digital 2~7 pins. As shown in fig.: LED chase effect wiring circuit:
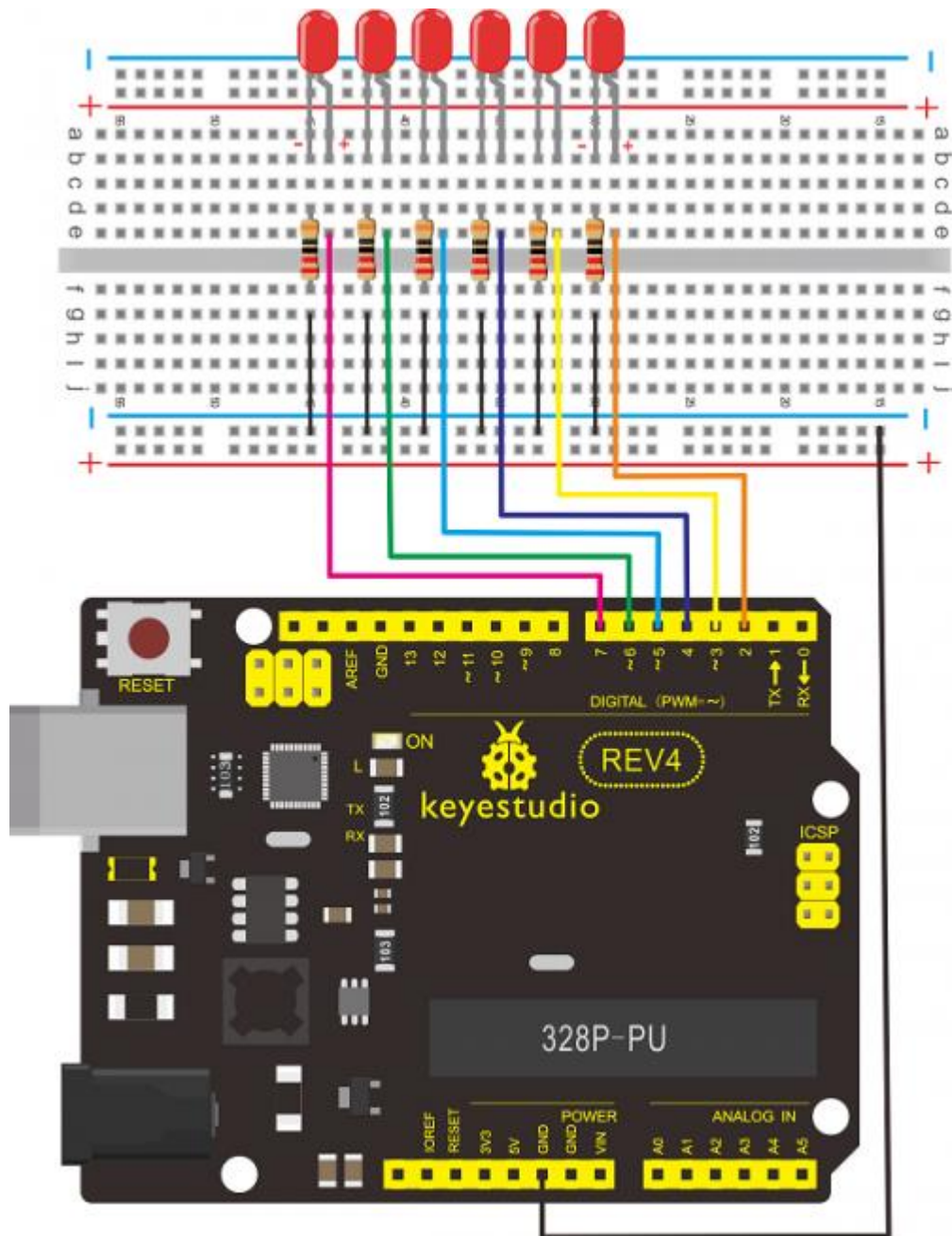
Sample Code:

```
int BASE = 2 ;  // the I/O pin for the first LED
int NUM = 8;    // number of LEDs

void setup()
{
   for (int i = BASE; i < BASE + NUM; i ++)
   {
     pinMode(i, OUTPUT);   // set I/O pins as output
   }
```

```
}

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

Result:
You can see the LEDs blink by sequence.

Project 5: Button-controlled LED


Introduction:

I/O port means interface for INPUT and OUTPUT. Up until now, we have only used its OUTPUT function. In this experiment, we will try to use the input function, which is to read the output value of device connecting to it. We use 1 button and 1 LED using both input and output to give you a better understanding of the I/O function. Button switches, familiar to most of us, are a switch value (digital value) component. When it's pressed, the circuit is in closed (conducting) state.
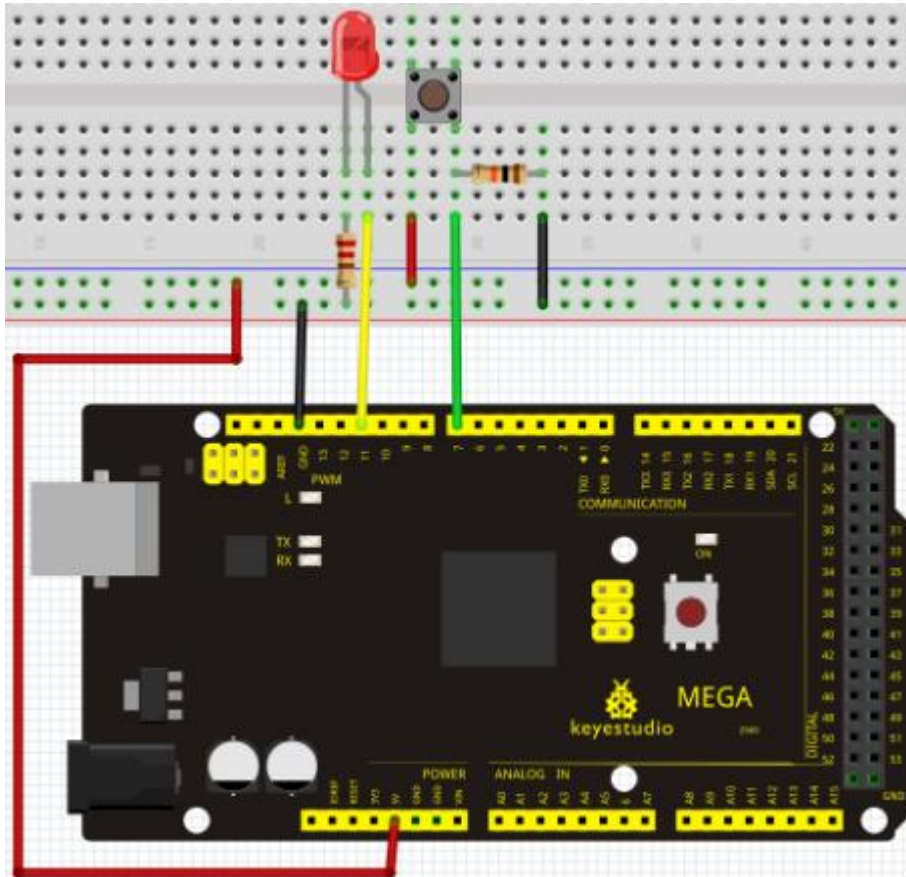
Hardware Required:
1. Button switch*1
2. Red M5 LED*1
3. 220Ω resistor*1
4. 10KΩ resistor*1

5. Breadboard*1
6. Breadboard jumper wires*several

Connection for REV4:
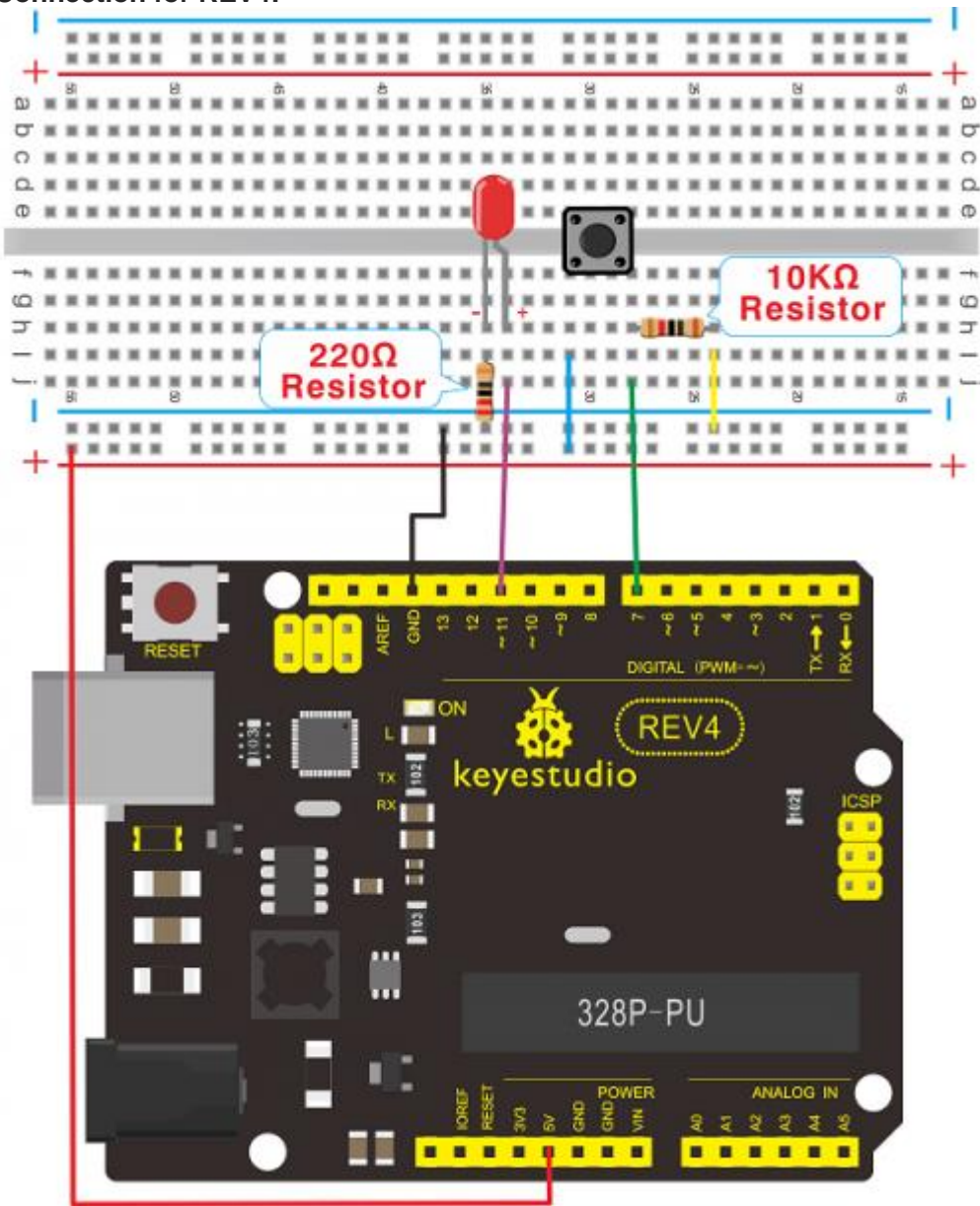
Connection for 2560 R3:

Sample Code:

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

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

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

```
int ledpin=11;// initialize pin 11
int inpin=7;// initialize pin 7
int val;// define val
void setup()
{
pinMode(ledpin,OUTPUT);// set LED pin as "output"
pinMode(inpin,INPUT);// set button pin as "input"

}
void loop()
{
val=digitalRead(inpin);// read the level value of pin 7 and assign if
to val
if(val==LOW)// check if the button is pressed, if yes, turn on the LED
```

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

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

Project 6: Responder experiment

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

Hardware required:
1. REV4 board *1
2. USB Cable *1
3. Button switch*4
4. Red M5 LED*1
5. Yellow M5 LED*1
6. Blue M5 LED*1
7. 220Ω resistor*3
8. 10KΩ resistor*4
9.Breadboard*1
10.Breadboard jumper wires* several

Circuit connection:
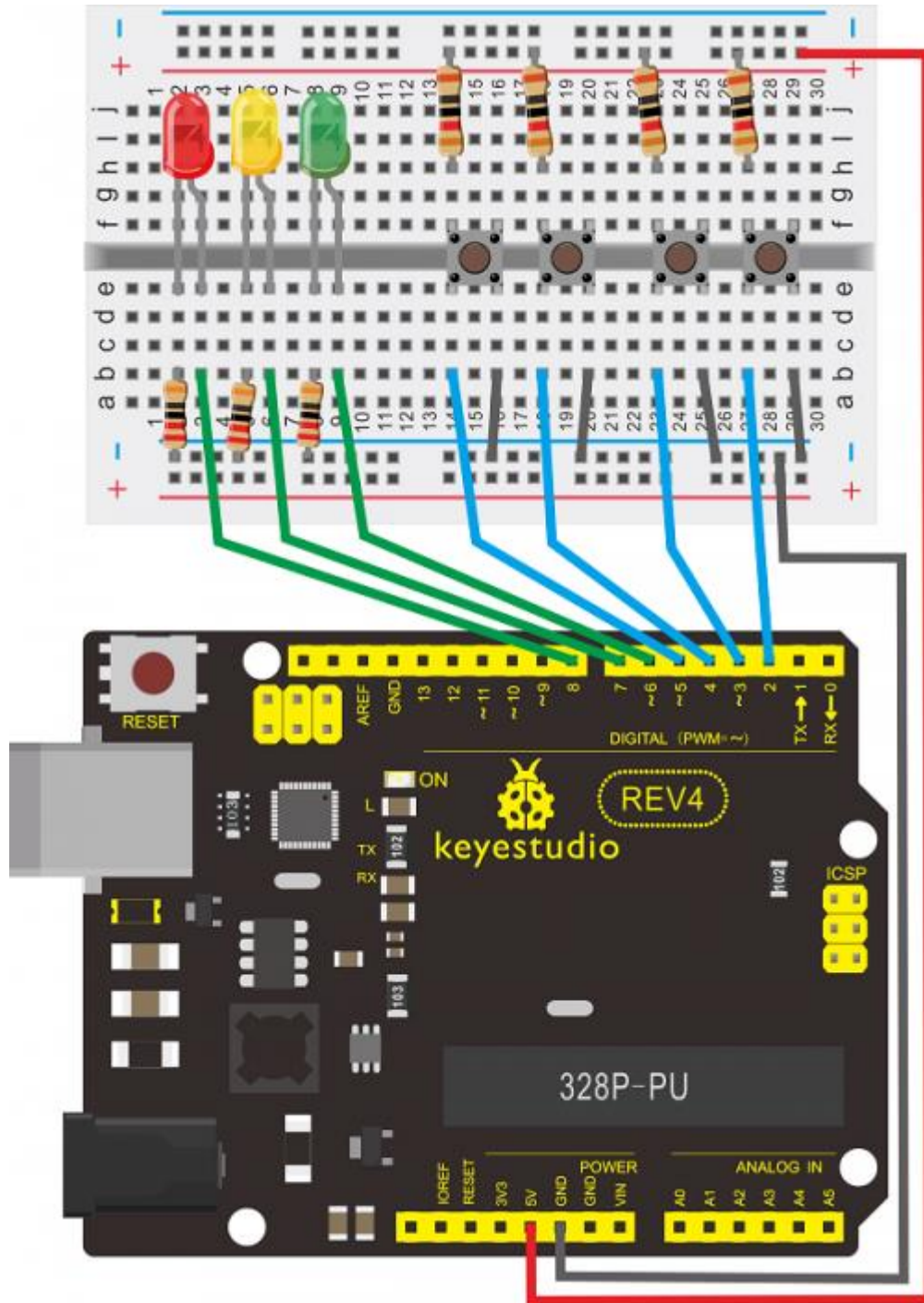
Sample Code:

```
int redled=8;      // set red LED as "output"
int yellowled=7;   // set yellow LED as "output"
int blueled=6;     // set blue LED as "output"
int redpin=5;      // initialize pin for red button
int yellowpin=4;   // initialize pin for yellow button
int bluepin=3;     // initialize pin for blue button
int restpin=2;     // initialize pin for reset button
```

```
int red;
int yellow;
int blue;
void setup()
{
pinMode(redled,OUTPUT);
pinMode(yellowled,OUTPUT);
pinMode(blueled,OUTPUT);
pinMode(redpin,INPUT);
pinMode(yellowpin,INPUT);
pinMode(bluepin,INPUT);
}
void loop()  // repeatedly read pins for buttons
{
red=digitalRead(redpin);
yellow=digitalRead(yellowpin);
blue=digitalRead(bluepin);
if(red==LOW)RED_YES();
if(yellow==LOW)YELLOW_YES();
if(blue==LOW)BLUE_YES();
}

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

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

Result:
Whichever button is pressed first, the corresponding LED will be on!
Then press the REST button to reset.
After the above process, we have built our own simple responder.

Project 7: 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.
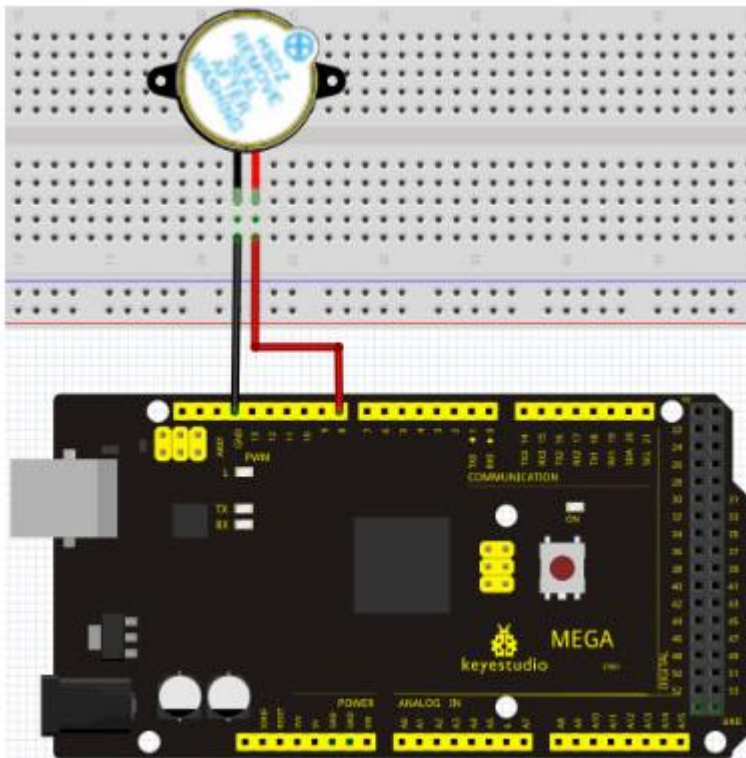
Hardware Required:
1. REV4 board *1
2.USB Cable *1
3.Active Buzzer*1
4.Breadboard*1
5.Breadboard jumper wires* several

Connection for REV4:
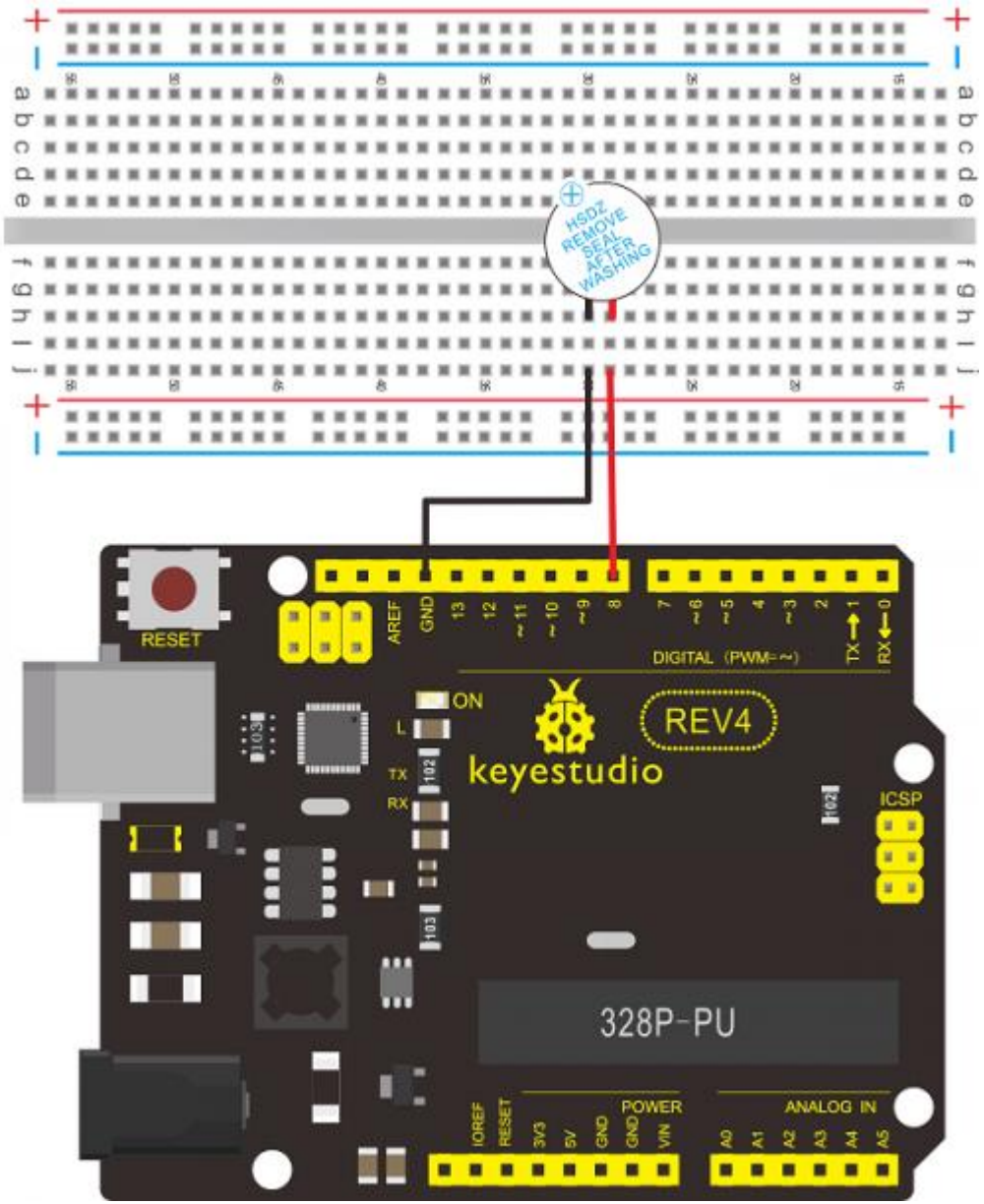
Connection for 2560 R3:

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

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

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

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

Project 8: Passive Buzzer
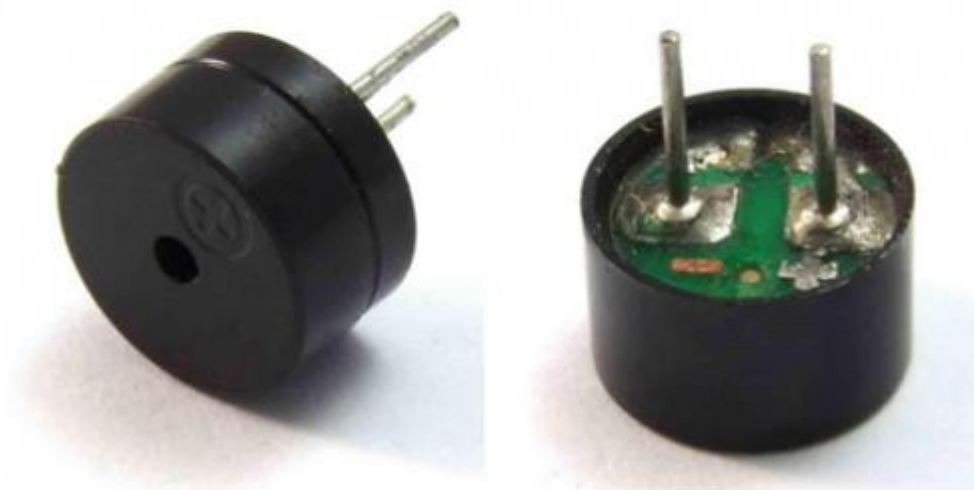
Introduction:

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

Hardware Required:
1. REV4 board *1
2.USB Cable *1
3.Passive buzzer*1
4.Breadboard*1
5.Breadboard jumper wires* several

Circuit connection:
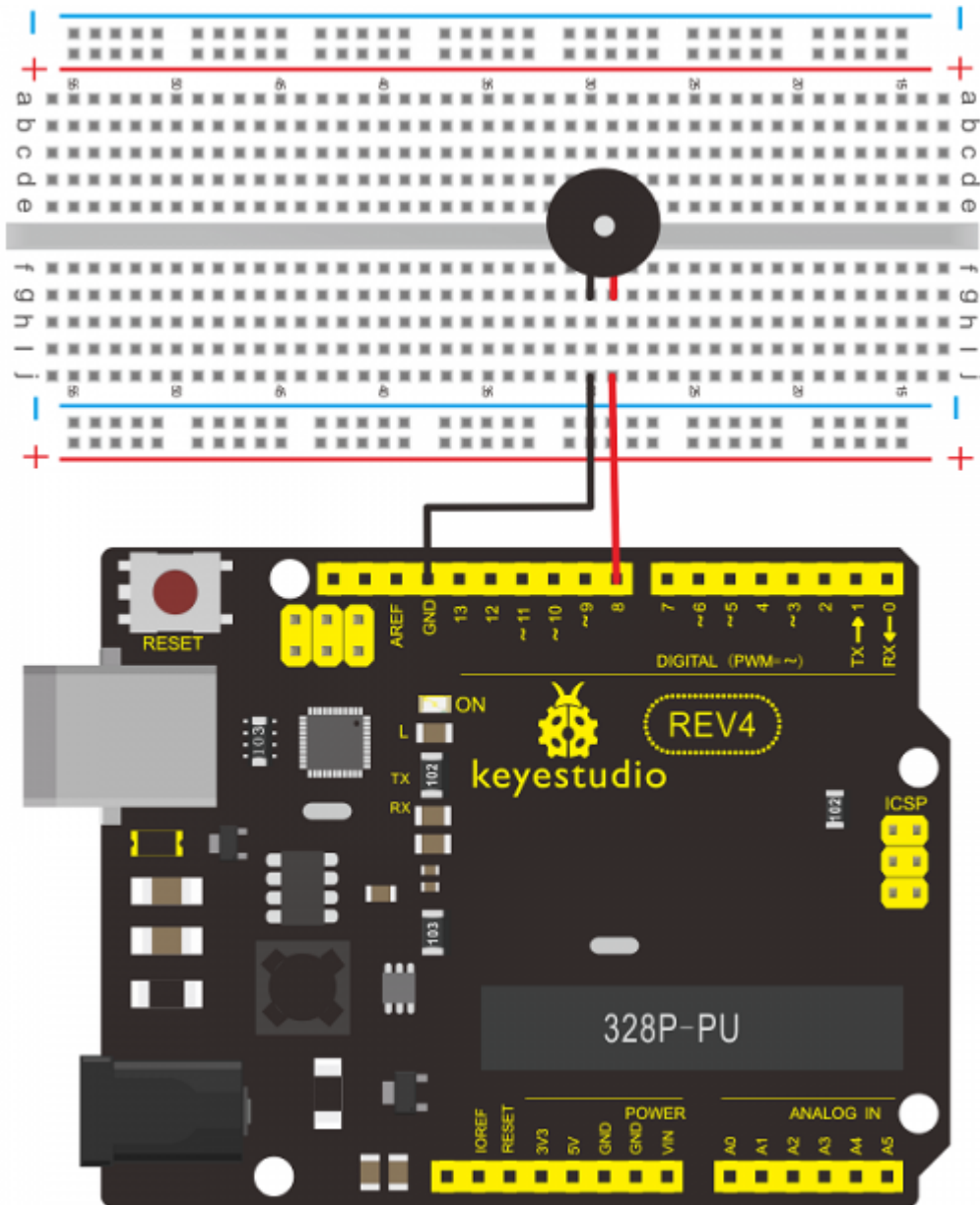
Sample Code:

```
int buzzer=8;// select digital IO pin for the buzzer
void setup()
{
pinMode(buzzer,OUTPUT);// set digital IO pin pattern, OUTPUT to be
output
}
void loop()
{ unsigned char i,j;//define variable
while(1)
{ for(i=0;i<80;i++)// output a frequency sound
{ digitalWrite(buzzer,HIGH);// sound
delay(1);//delay1ms
```

```
digitalWrite(buzzer,LOW);//not sound
delay(1);//ms delay
}
for(i=0;i<100;i++)// output a frequency sound
{ digitalWrite(buzzer,HIGH);// sound
digitalWrite(buzzer,LOW);//not sound
delay(2);//2ms delay
}
}
}
```
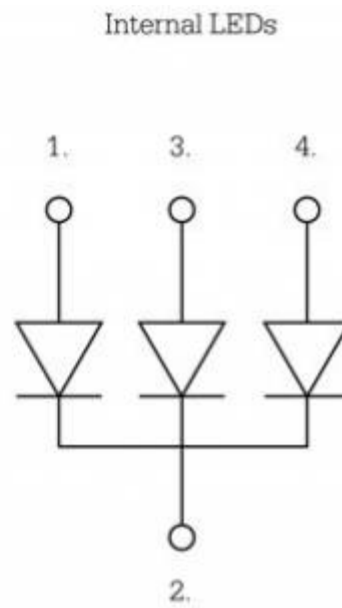
Result:
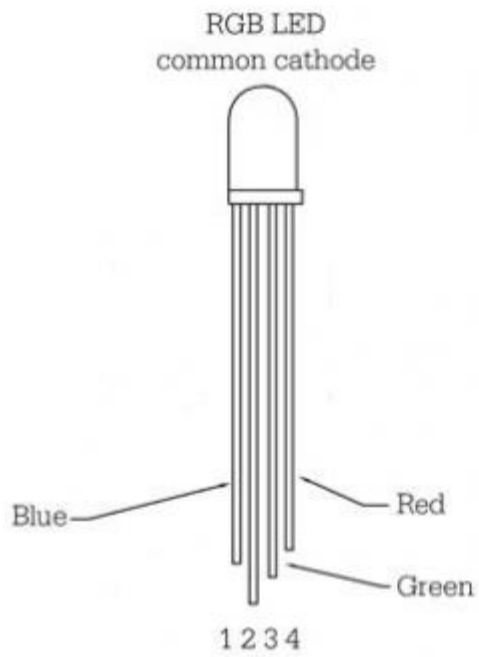After downloading the program, the buzzer experiment is completed.

Project 9: RGB LED

Introduction:
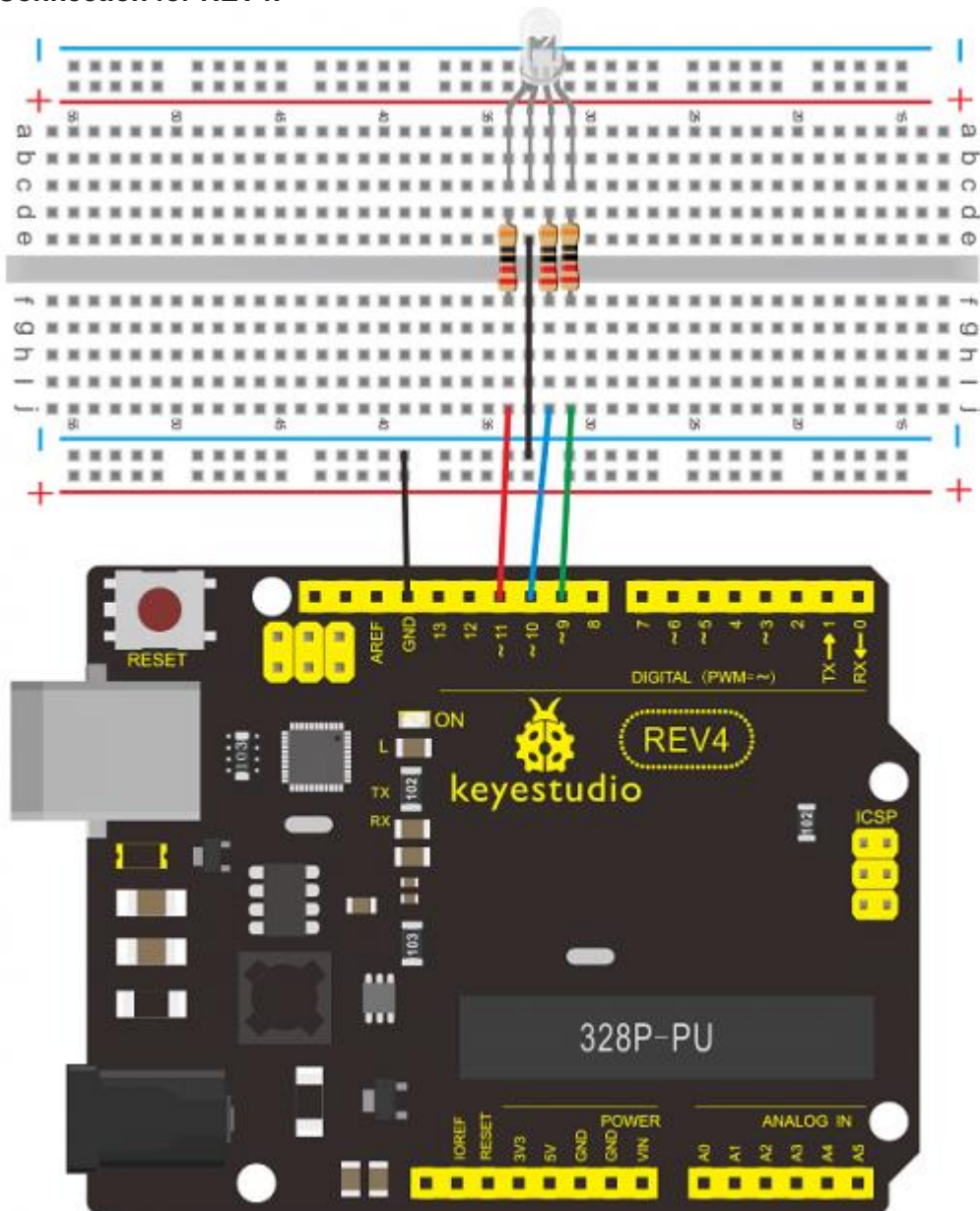Tricolor principle to display various colors
PWM controlling ports to display full color
Can be driven directly by Arduino PWM interfaces

RGB LED
common cathode

Internal LEDs

1.   3.   4.

Blue

Red

Green

1 2 3 4

2.

Hardware Required:
1.REV4 board *1
2.USB Cable *1
3.RGB LED *1
4.Breadboard*1
5.Breadboard jumper wires* several

Connection for REV4:

Sample Code:

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

int val;

void setup() {
  pinMode(redpin, OUTPUT);
  pinMode(bluepin, OUTPUT);
  pinMode(greenpin, OUTPUT);
  Serial.begin(9600);
}
```

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

Result:

Directly copy the above code into arduino IDE, and click upload , wait a few seconds, you can see a full-color LED

# Project 10: Analog Value Reading

Introduction:
In this experiment, we will begin the learning of analog I/O interfaces. On an Arduino, there are 6 analog interfaces numbered from 0 to 5. These 6 interfaces can also be used as digital ones numbered as 14-19. After a brief introduction, let's begin our project. Potentiometer used here is a typical output component of analog value that is familiar to us.

Hardware Required:
1. REV4 board *1
2.USB Cable *1
3.Potentiometer *1
4.Breadboard*1
5.Breadboard jumper wires* several

Connection:
In this experiment, we will convert the resistance value of the potentiometer to analog ones and display it on the screen. This is an application we need to master well for our future experiments.
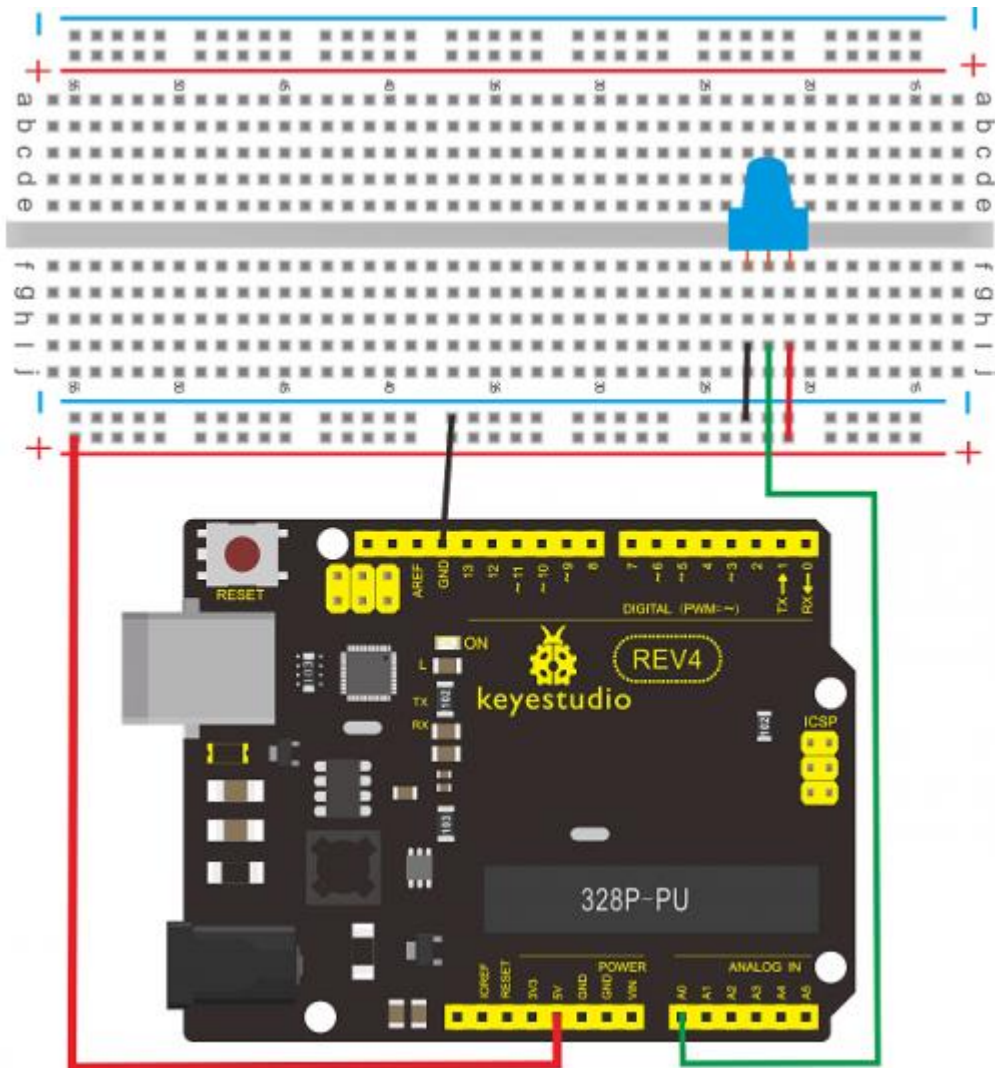
We use the analog interface 0. The analog interface we use here is interface 0.

Sample Code:

The program compiling is simple. An analogRead () Statement can read the value of the interface. The A/D acquisition of Arduino 328 is in 10 bits, so the value it reads is among 0 to 1023. One difficulty in this project is to display the value on the screen, which is actually easy to learn. First, we need to set the baud rate in voidsetup (). Displaying the value is a communication between Arduino and PC, so the baud rate of the Arduino should match the the one in the PC's software set up. Otherwise, the display will be messy codes or no display at all. In the lower right corner of the Arduino software monitor window, there is a button for baud rate set up. The set up here needs to match the one in the program. The statement in the program is Serial.begin(); enclosed is the baud rate value, followed by statement for displaying. You can either use Serial.print() or Serial.println() statement.

```
int potpin=0;// initialize analog pin 0
int ledpin=13;// initialize digital pin 13
int val=0;// define val, assign initial value 0
void setup()
{
pinMode(ledpin,OUTPUT);// set digital pin as "output"
Serial.begin(9600);// set baud rate at 9600
```

```
}
void loop()
{
digitalWrite(ledpin,HIGH);// turn on the LED on pin 13
delay(50);// wait for 0.05 second
digitalWrite(ledpin,LOW);// turn off the LED on pin 13
delay(50);// wait for 0.05 second
val=analogRead(potpin);// read the analog value of analog pin 0, and
assign it to val
Serial.println(val);// display val's value
}
```

Result:
The sample program uses the built-in LED connected to pin 13. Each time the device reads a value, the LED blinks.
Below is the analog value it reads.
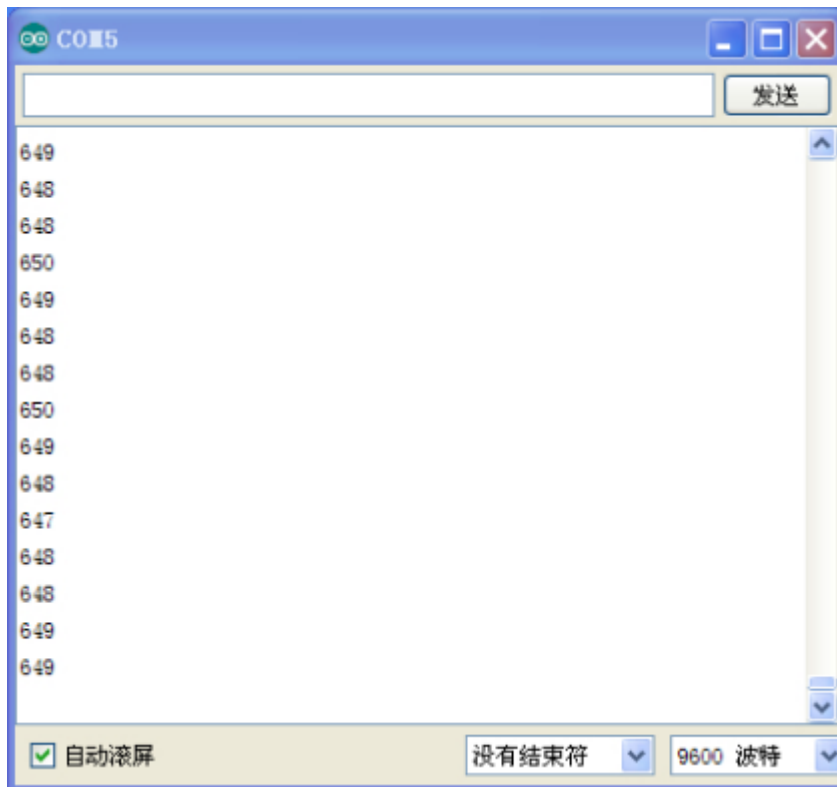
When you rotate the potentiometer knob, you can see the displayed value changes. The reading of analog value is a very common function since most sensors output analog value. After calculation, we can have the corresponding value we need. The experiment is now completed, thank you.

Project 11: Photo 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.
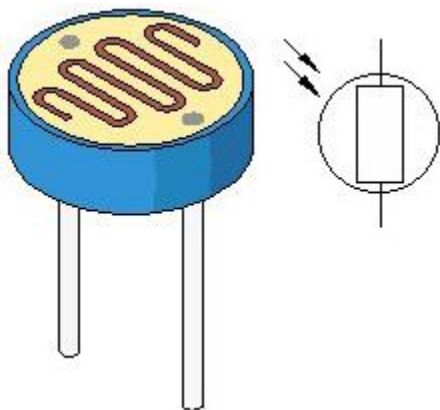
We will start with a relatively simple experiment regarding photovaristor application. Photovaristor is an element that changes its resistance as light strenth changes. So we will need to read the analog values. We can refer to the PWM experiment, replacing the potentiometer with photovaristor. When there is change in light strength, there will be corresponding change on the LED.

Hardware Required:
1.REV4 board *1
2.USB Cable *1
3..Photo resistor*1
4.Red M5 LED*1
5.10KΩresistor*1
6.220Ωresistor*1
7.Breadboard*1
8.Breadboard jumper wires* several

Circuit connection:
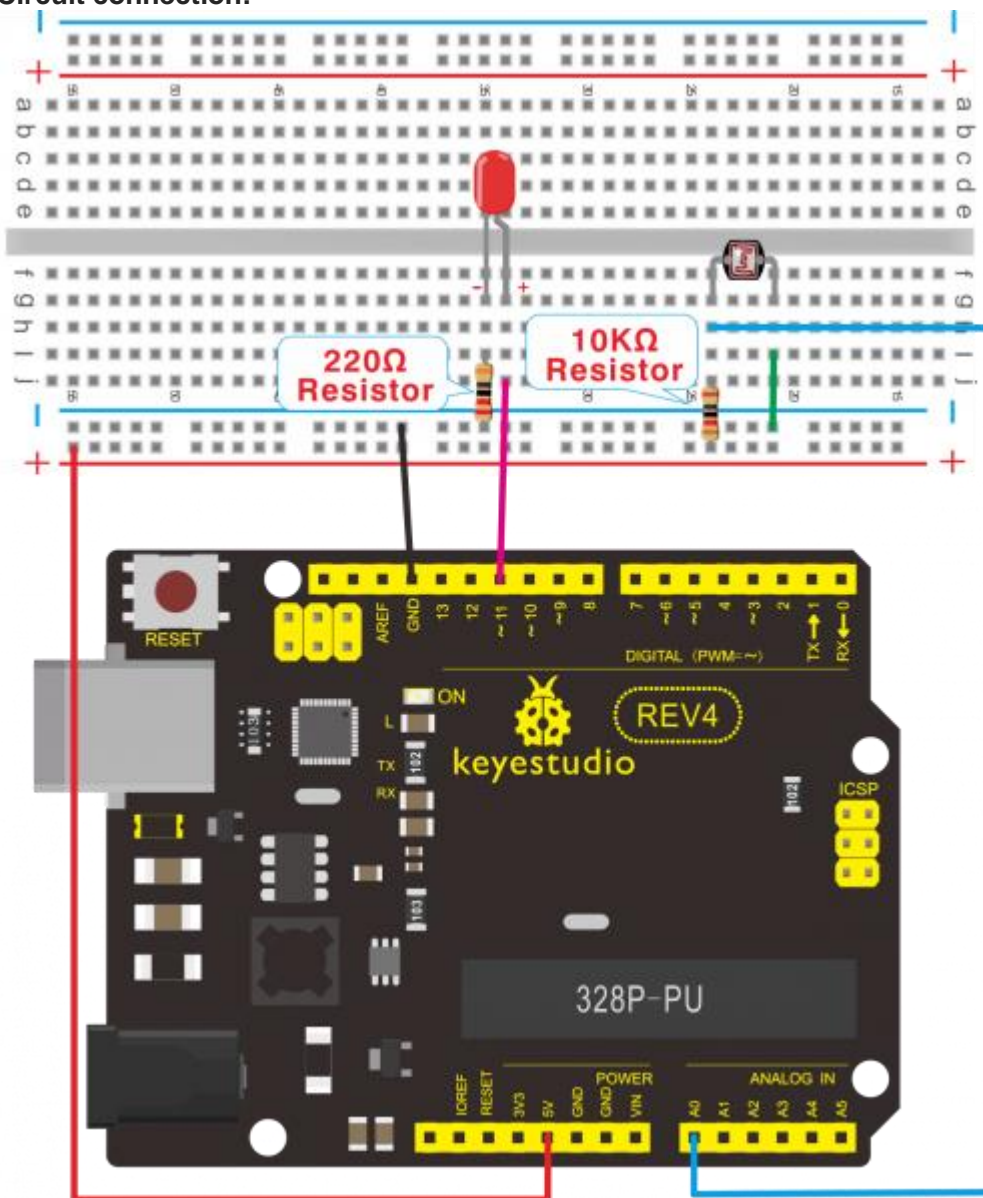
Sample Code:
After the connection, let's begin the program compiling. The program is similar to the one of PWM. For change detail, please refer to the sample program below.

```
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
}
```

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.

Project 12: 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:
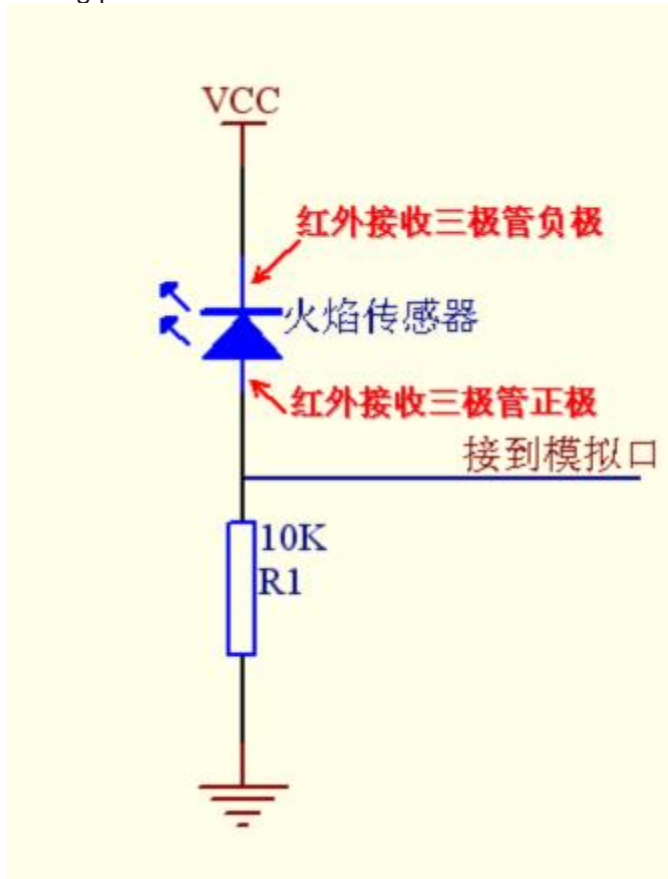

Hardware Required:
1.REV4 board *1
2.USB Cable *1
3.Flame sensor *1
4.Buzzer *1
5.10K resistor *1
6.Breadboard*1
7.Breadboard jumper wires* several

Circuit connection:
1）Connecting buzzer:
Connect the controller board, prototype board, breadboard and USB cable according to the Arduino tutorial. Connect the buzzer to digital pin 8.

2）Connecting flame sensor:
Connect the sensor to analog pin 0.
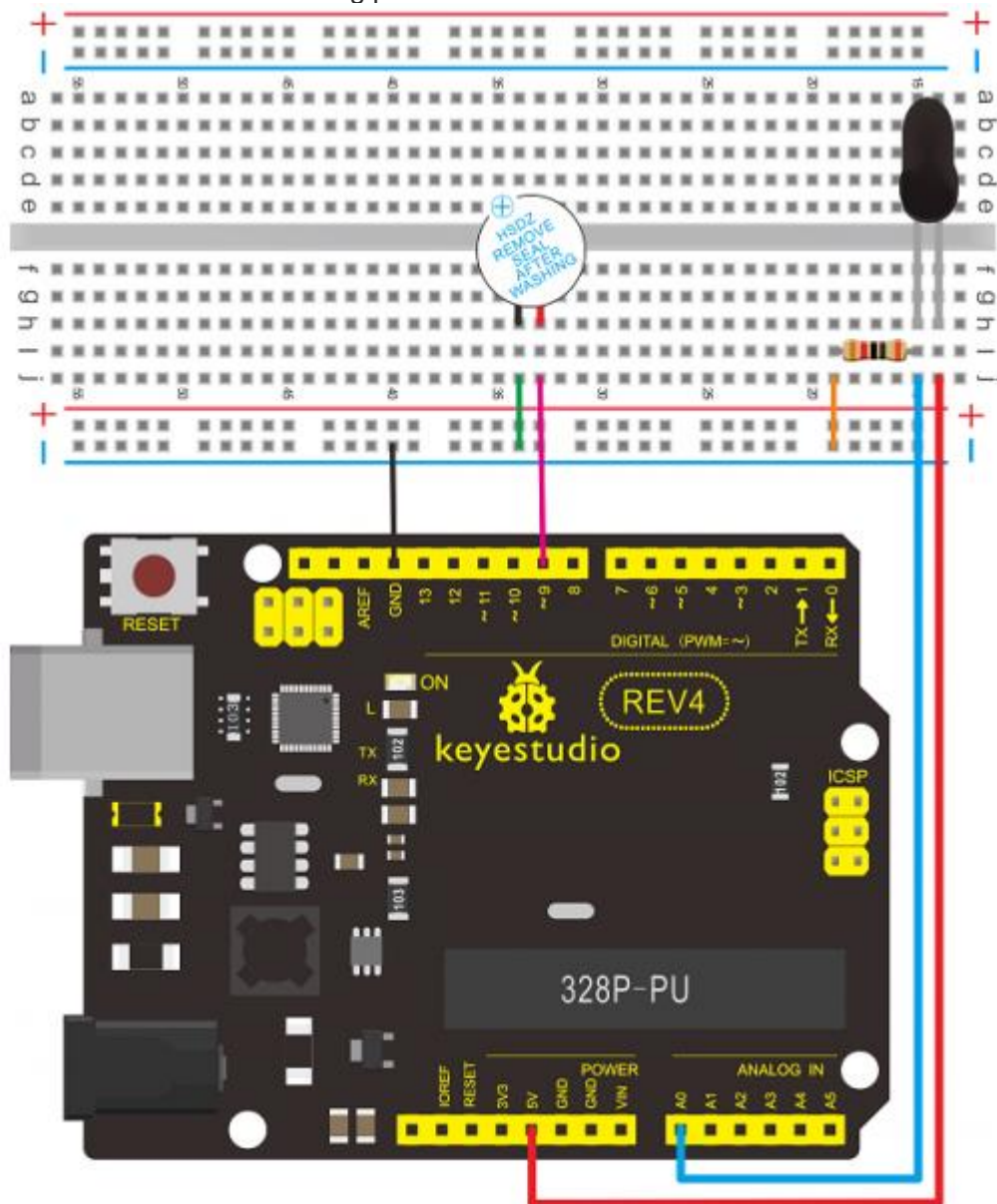
Experiment principle:
When it's approaching a fire, the voltage value the analog port reads differs. If you use a multimeter, you can know when there is no fire approaching, the voltage it reads is around 0.3V; when there is fire approaching, the voltage it reads is around 1.0V, tthe nearer the fire, the higher the voltage. So in the beginning of the program, you can initialize voltage value i (no fire value); Then, continuously read the analog voltage value j and obtain difference value k=j-i; compare k with 0.6V (123 in binary) to determine whether or not there is a fire approaching; if yes, the buzzer will buzz.

Sample Code:

```
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);
}
```

Result:
This program can simulate an alarm when there is a fire. Everything is normal when there is no fire; when there is, the alarm will be set off immediately.

Project 13: Analog temperature (thermistor)

Introduction:
Thermistor is a temperature measuring component base on the principle that a conductor changes in resistance with a change in its body temperature. As a result, it requires the temperature coefficient and the resistivity of the conductor to be as large and stable as possible. It is best that the resistance is in linear relationship with temperature. And it should also have stable physical and chemical properties in a wide range. Currently, the most used thermal resistance materials are platinum, nickel and copper.

Hardware Required:
1.REV4 board *1
2.USB Cable *1
3.Thermistor *1
7.10KΩ resistor *1
5.Breadboard*1
6.Breadboard jumper wires* several

Circuit connection:
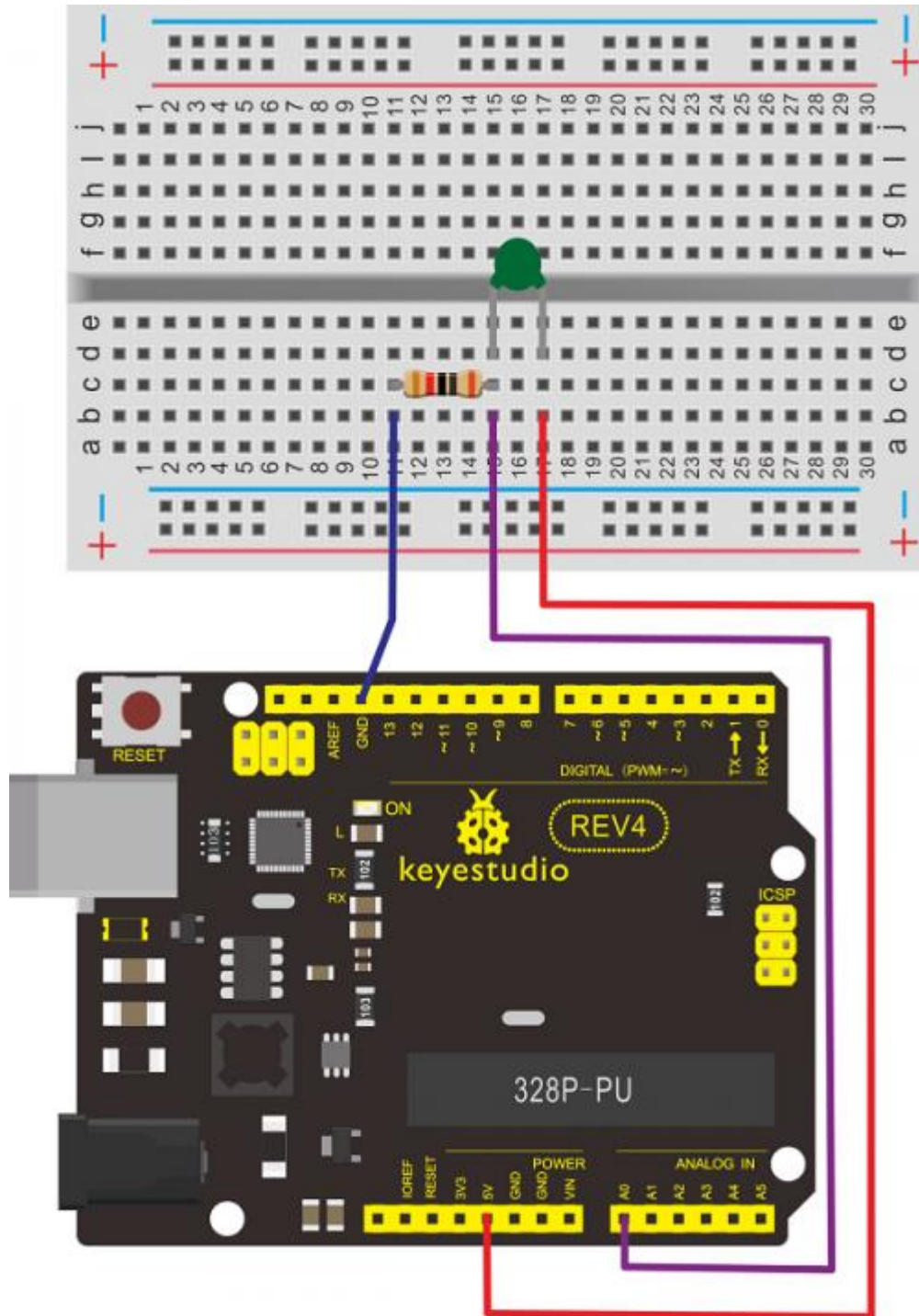
Sample Code:

```
int pin = 7;  //attach to the third pin of NE555
unsigned long duration;  //the variable to store the length of the
pulse

void setup()
{
  Serial.begin(9600); //Set serial baud rate to 9600 bps
```

```
}
void loop()
{
int val;
val=analogRead(0);//Read rotation sensor value from analog 0
Serial.println(val,DEC);//Print the value to serial port
delay(100);
}
```

Result:
Shown in pic 1 is data displayed by serial port monitor in room temperature. After the temperature is changed (thermistor wrapped and put into hot water), the data changes as shown in pic 2.
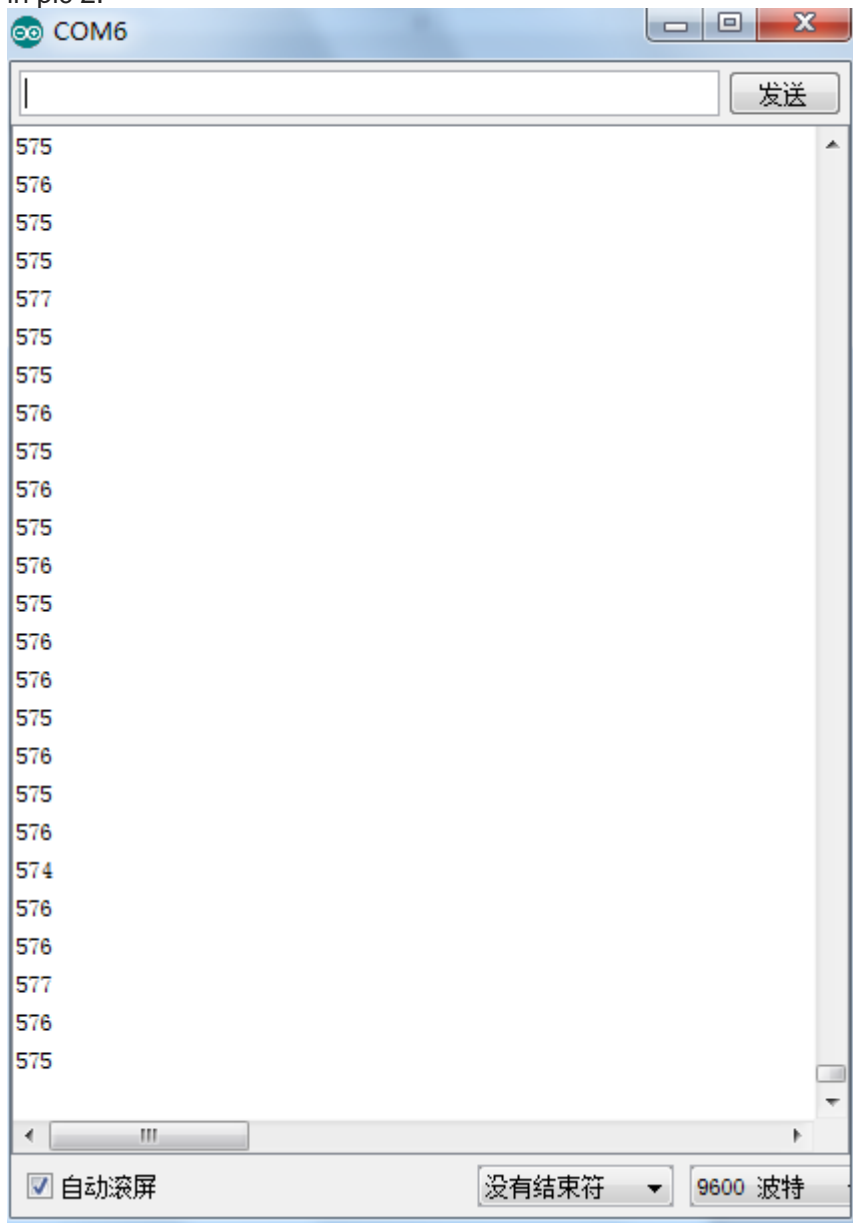
Pic 1
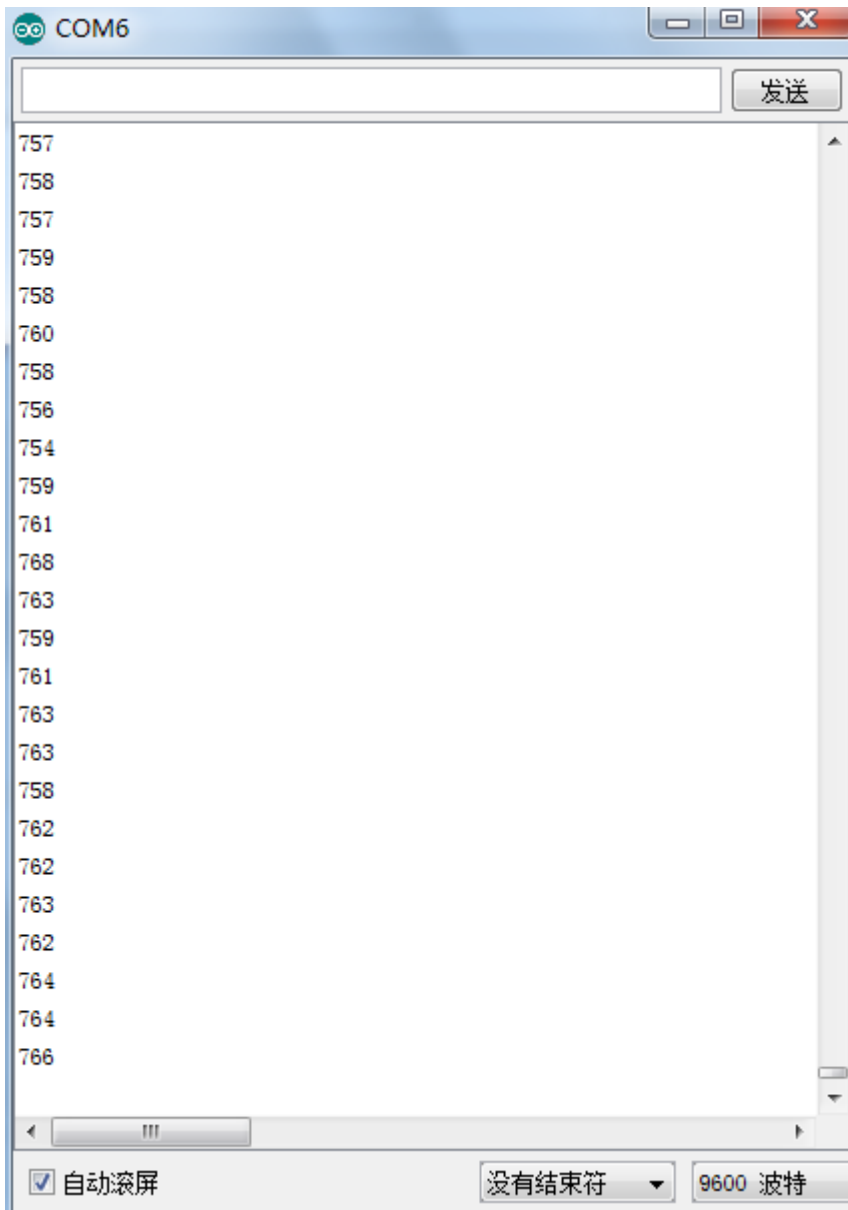
Pic 2

Project 14: LM35 Temperature Sensor
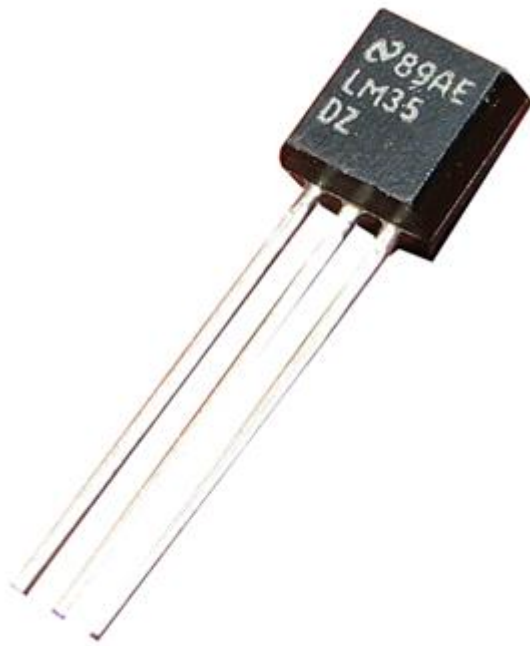
Introduction:
LM35 is a common and easy-to-use temperature sensor. It does not require other hardware. You just need an analog port to make it work. The difficulty lies in compiling the code to convert the analog value it reads to celsius temperature.

LM35

1 4-20V
2 OUT
3 GND

Hardware Required:
1. REV4 board *1
2.USB Cable *1
3.LM35*1

4.Breadboard*1
5.Breadboard jumper wires* several

Circuit connection:

Sample Code:

```
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
```
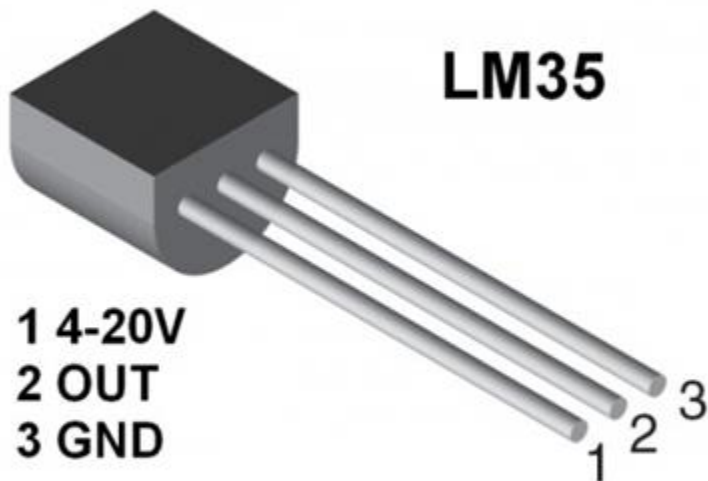
```
dat=(125*val)>>8;// temperature calculation formula
Serial.print("Tep:");// output and display characters beginning with
Tep
Serial.print(dat);// output and display value of dat
Serial.println("C");// display "C" characters
delay(500);// wait for 0.5 second
}
```

Result:
After downloading the program, you can open the monitoring window to see current temperature.
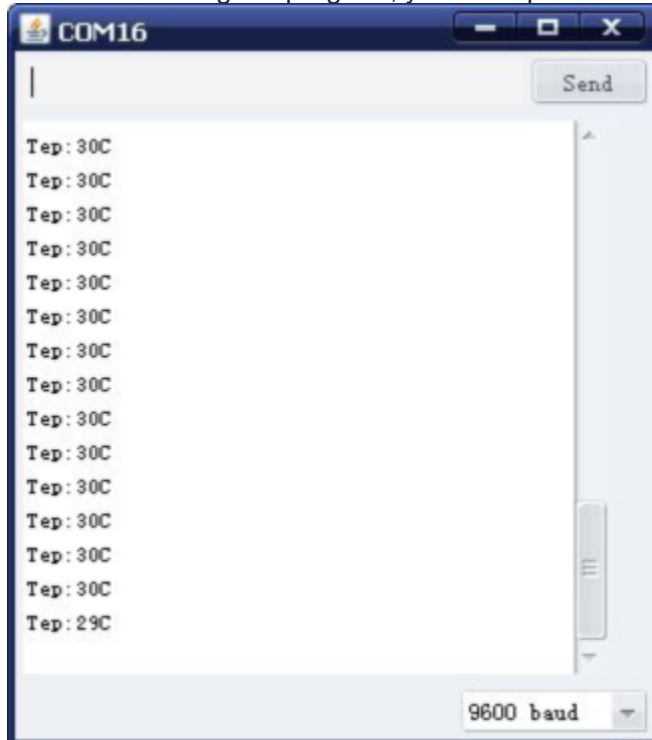
Project 15: A cup with temperature indicator

Introduction:
Today, we will use Arduino to make a temperature-indicated cup. First, let's design the circuit.
When the LM35 temperature sensor senses different temperature, different LED will be turned on representing the temperature.

Hardware Required:
1.REV4 board *1
2.USB Cable *1
3.Red M5 LED*1
4.Yellow M5 LED*1
5.Blue M5 LED*1
6.220Ω Resistor *3
7.LM35 temperature sensor *1
8.Breadboard*1
9. Breadboard jumper wires* several

Circuit connection:

Sample Code:

```
void setup() {
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
}
void loop() {
  int vol = analogRead(A0) * (5.0 / 1023.0*100);   // read temperature
value of LM35
if (vol<=31)                              // low temperature area and LED setup
{
  digitalWrite(13, HIGH);
  digitalWrite(12, LOW);
  digitalWrite(11, LOW);
```

```
}
else if (vol>=32 && vol<=40)
 {
   digitalWrite(13, LOW);

  digitalWrite(12, HIGH);
  digitalWrite(11, LOW);
}
else if (vol>=41)                              //  low temperature
area and LED setup

{
   digitalWrite(13, LOW);
  digitalWrite(12, LOW);
  digitalWrite(11, HIGH);
}
}
```

Result:
Corresponding LED will be turned on in accordance with corresponding temperature range.

Project 16: DHT11 Temperature and Humidity Sensor
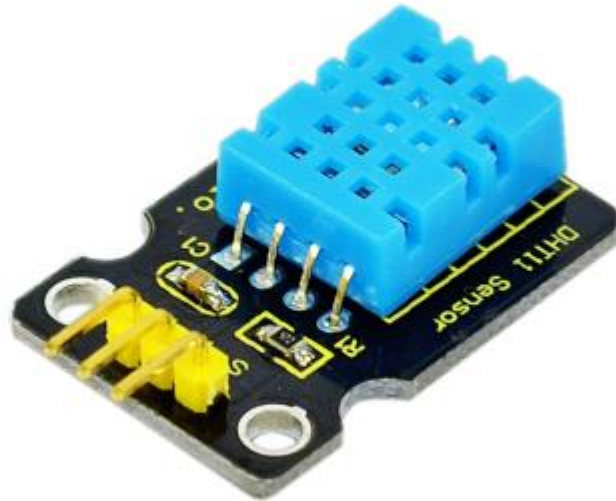
Introduction:
This DHT11 Temperature and Humidity Sensor features calibrated digital signal output with the temperature and humidity sensor complex. Its technology ensures high reliability and excellent long-term stability. A high-performance 8-bit microcontroller is connected. This sensor includes a resistive element and a sense of wet NTC temperature measuring devices. It has excellent quality, fast response, anti-interference ability and high cost performance advantages.
Each DHT11 sensor features extremely accurate calibration data of humidity calibration chamber. The calibration coefficients stored in the OTP program memory, internal sensors detect signals in the process, and we should call these calibration coefficients. The single-wire serial interface system is integrated to make it quick and easy. Qualities of small size, low power, and 20-meter signal transmission distance make it a wide applied application and even the most demanding one. Convenient connection, special packages can be provided according to users need.

Hardware required:
1. REV4 board *1
2.USB Cable *1
3.Temperature and humidity unit *1
4.10K resistor *1
5.Breadboard*1
6. Breadboard jumper wires* several

Circuit connection:
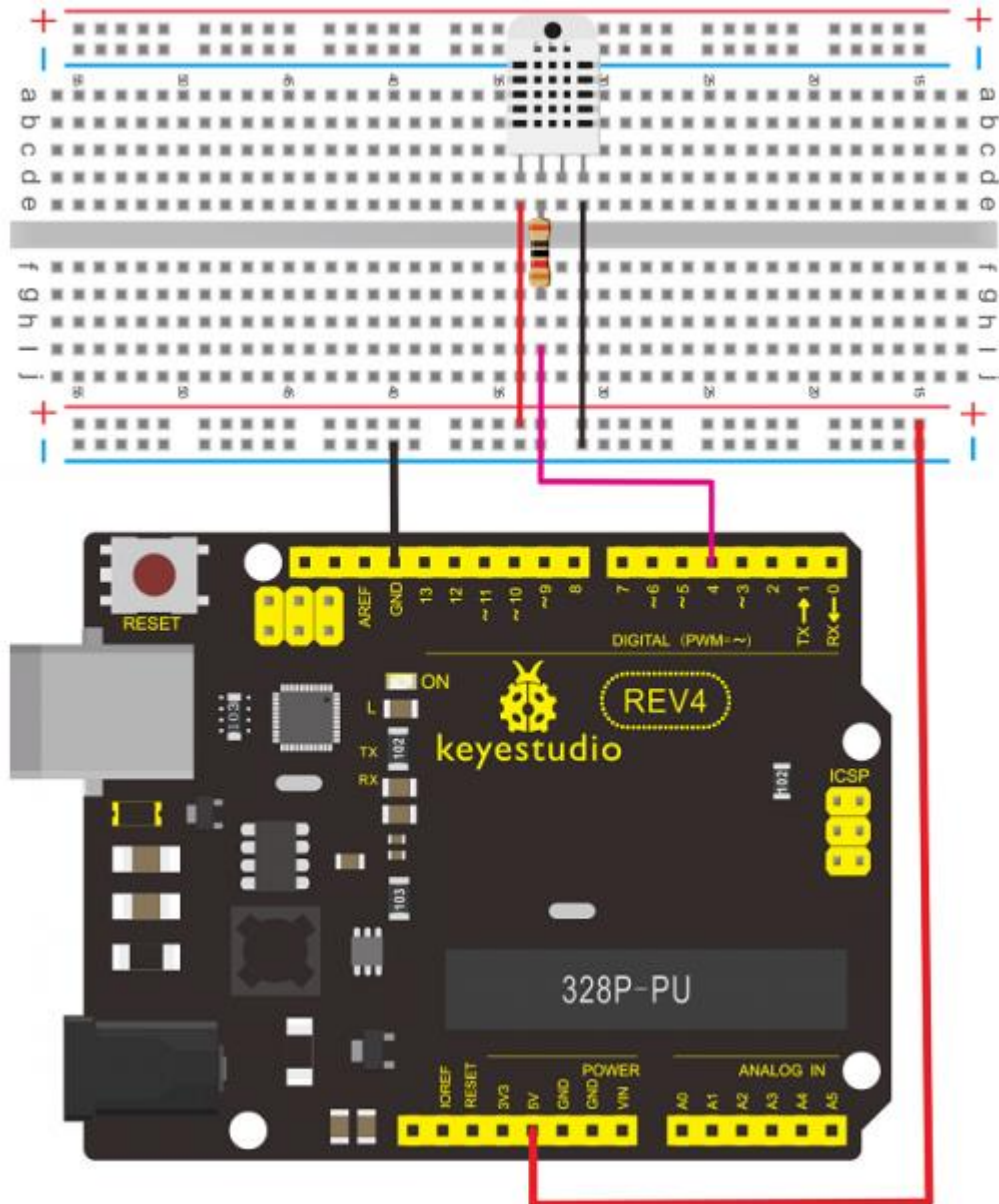
Sample Code :
Please download the DHT11Lib firstly.Or,see the website

```
#include <dht11.h>
dht11 DHT;
#define DHT11_PIN 4

void setup(){
  Serial.begin(9600);
  Serial.println("DHT TEST PROGRAM ");
  Serial.print("LIBRARY VERSION: ");
  Serial.println(DHT11LIB_VERSION);
  Serial.println();
  Serial.println("Type,\tstatus,\tHumidity (%),\tTemperature (C)");
```

```
  }

void loop(){
  int chk;
  Serial.print("DHT11, \t");
  chk = DHT.read(DHT11_PIN);    // READ DATA
  switch (chk){
    case DHTLIB_OK:
                Serial.print("OK,\t");
                break;
    case DHTLIB_ERROR_CHECKSUM:
                Serial.print("Checksum error,\t");
                break;
    case DHTLIB_ERROR_TIMEOUT:
                Serial.print("Time out error,\t");
                break;
    default:
                Serial.print("Unknown error,\t");
                break;
  }
 // DISPLAT DATA
  Serial.print(DHT.humidity,1);
  Serial.print(",\t");
  Serial.println(DHT.temperature,1);

  delay(1000);
}
```

Result:


```
COM112                              ─  ▫  ✕

[                              ]  发送

DHT TEST PROGRAM                           ▲
LIBRARY VERSION: 0.4.1
                                           ▤

Type,   status, Humidity (%),   Temperature (C)
DHT11,  OK,     35,     35
DHT11,  OK,     35,     35
DHT11,  OK,     35,     35
DHT11,  OK,     34,     35
DHT11,  OK,     35,     35
DHT11,  OK,     35,     35
DHT11,  OK,     35,     35
DHT11,  OK,     35,     35
DHT11,  OK,     35,     35
DHT11,  OK,     35,     35
DHT11,  OK,     35,     35
DHT11,  OK,     35,     35
DHT11,  OK,     35,     35
DHT11,  OK,     35,     35
DHT11,  OK,     35,     35
DHT11,  OK,     35,     35
DHT11,  OK,     35,     35
DHT11,  OK,     35,     35
DHT11,  OK,     35,     35
DHT11,  OK,     36,     35
DHT11,  OK,     35,     35
DHT11,  OK,     35,     35     ▼

☐ 自动滚屏        没有结束符  ▼  9600 波特
```
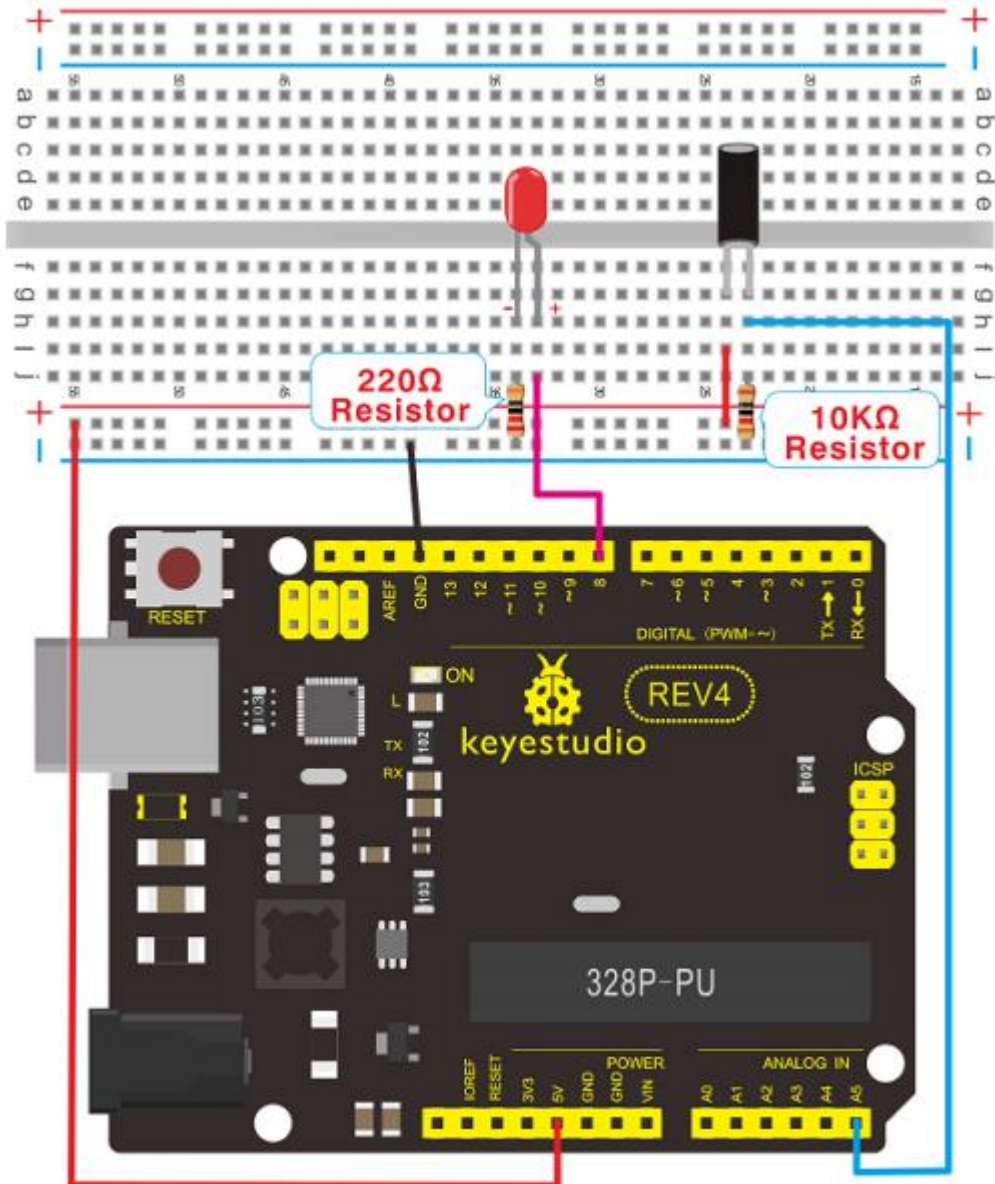
Project 17: Tilt Switch

Introduction:
Tilt switch controlling the ON and OFF of LED.

Hardware Required:
1.REV4 board *1
2.USB Cable *1
3.Red M5 LED*1
4.Ball switch*1
5.220Ω Resistor *1
6.10kΩ resistor*1
7.Breadboard*1
8.Breadboard jumper wires* several

Circuit connection:

Connect the controller board, shield, breadboard and USB cable according to Arduino tutorial. Connect the LED to digital pin 8, ball switch to analog pin 5.

Experiment Principle:
When one end of the switch is below horizontal position, the switch is on. The voltage of the analog port is about 5V (1023 in binary). The LED will be on. When the other end of the switch is below horizontal position, the switch is off. The voltage of the analog port is about 0V (0 in binary). The LED will be off. In the program, we determine whether the switch is on or off according to the voltage value of the analog port, whether it's above 2.5V (512 in binary) or not.

Sample Code:

```
void setup()
{
  pinMode(8,OUTPUT);// set digital pin 8 as "output"
}
void loop()
{
```

```
int i;// define variable i
while(1)
{
  i=analogRead(5);// read the voltage value of analog pin 5
  if(i>512)// if larger that 512 (2.5V)
  {
    digitalWrite(8,LOW);// turn on LED
  }
  else// otherwise
  {
    digitalWrite(8,HIGH);// turn off LED
  }
 }
}
```

Result:
Hold the breadboard with your hand. Tilt it to a certain extent, the LED will be on.
If there is no tilt, the LED will be off.
The principle of this experiment can be applied to relay control.
Experiment completed.
Thank you!

Project 18: Magical Light Cup

Introduction:
Magical light cup is a product developed by KEYES that can interact with ARDUINO.
The principle is to use PWM to regulate light brightness of the two components.
Mercury switch provides digital signal, triggering PWM to regulate light brightness. Through the designed program, We can see effect like two cups pouring light to each other.

Hardware Required:
1.REV4 board *1
2.USB Cable *1
3.Red M5 LED*2
4.Ball switch*2
5.220Ω Resistor *2
6.10kΩ resistor*2
7.Breadboard*1
8.Breadboard jumper wires* several

Circuit connection:
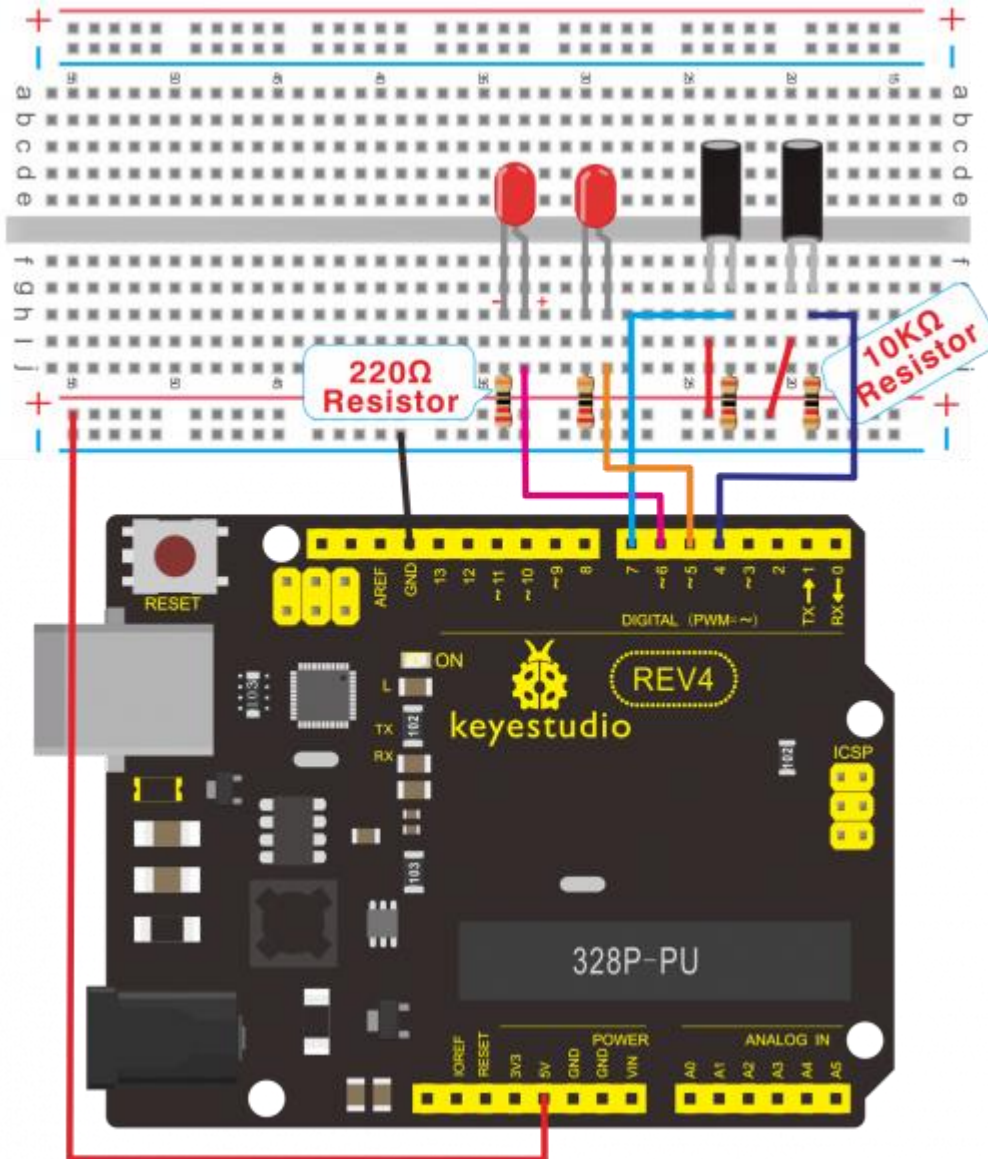
Sample Code:

```
int LedPinA = 5;
int LedPinB = 6;
int ButtonPinA = 7;
int ButtonPinB = 4;
int buttonStateA = 0;
int buttonStateB = 0;
int brightnessA = 0;
int brightnessB= 255;
void setup()
{
Serial.begin(9600);
pinMode(LedPinA, OUTPUT);
pinMode(LedPinB, OUTPUT);
```

```
pinMode(ButtonPinA, INPUT);
pinMode(ButtonPinB, INPUT);
}
void loop()
{
buttonStateA = digitalRead(ButtonPinA);
if (buttonStateA == HIGH && brightnessA != 255)
{
brightnessA ++;
}
if (buttonStateA == LOW && brightnessA != 0)
{
brightnessA --;
}
analogWrite(LedPinB, brightnessA);
Serial.print(brightnessA);
Serial.print("   ");
buttonStateB = digitalRead(ButtonPinB);
if (buttonStateB == HIGH && brightnessB != 0)
{
brightnessB --;
}
if (buttonStateB == LOW && brightnessB != 255)
{
brightnessB++;
}
analogWrite(LedPinA, brightnessB);
Serial.println(brightnessB);
delay(5);
}
```

Result:
Tilt the circuit to one side, A light on, B light out; tilt to the other side, A light out, B light on.

## Project 19: 1-digit LED Segment Display
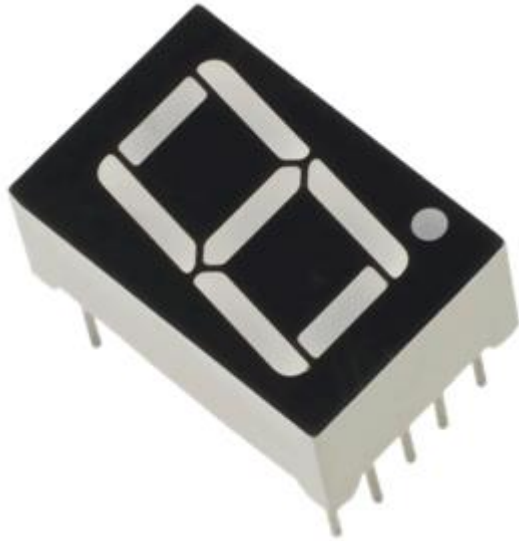
Introduction:

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.

Common cathode 7-segment display
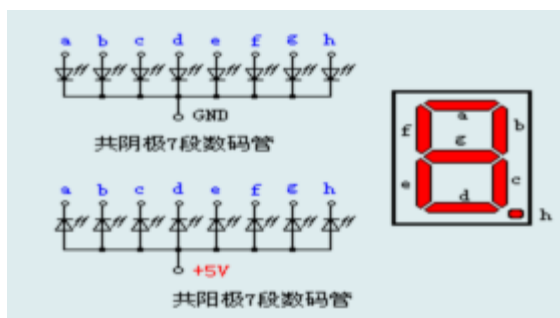

Common anode 7-segment display

Each segment of the display consists of an LED. So when you use it, you also need use a current-limiting resistor. Otherwise, LED will be burnt out. In this experiment, we use a common cathode display. As we mentioned above, for 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.

Hardware Required:
1.REV4 board *1
2.USB Cable *1

3.Eight-segment display*1
4.220Ω Resistor *2
5.Breadboard*1
6. Breadboard jumper wires* several

Circuit connection:
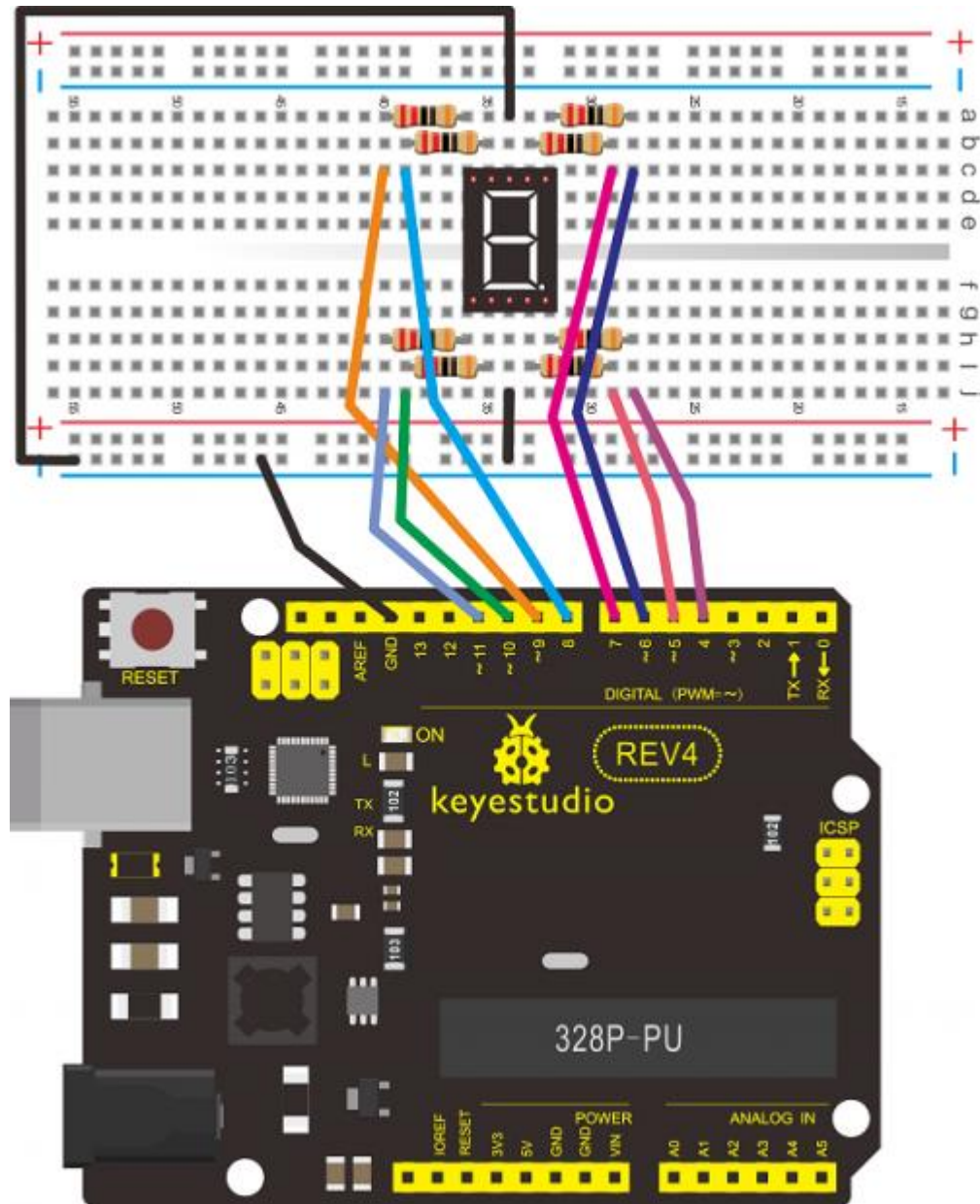
Sample Code:
There are seven segments for numerical display, one for decimal point display. Corresponding segments will be turned on when displaying certain numbers. For example, when displaying number 1, b and c segments will be turned on. We compile a subprogram for each number, and compile the main program to display one number every 2 seconds, cycling display number 0 ~ 9. The displaying time for each number is subject to the delay time, the longer the delay time, the longer the displaying time.

```
// set the IO pin for each segment
int a=7;// set digital pin 7 for segment a
```

```cpp
int b=6;// set digital pin 6 for segment b
int c=5;// set digital pin 5 for segment c
int d=10;// set digital pin 10 for segment d
int e=11;// set digital pin 11 for segment e
int f=8;// set digital pin 8 for segment f
int g=9;// set digital pin 9 for segment g
int dp=4;// set digital pin 4 for segment dp
void digital_0(void) // display number 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 number 1
{
unsigned char j;
digitalWrite(c,HIGH);// set level as "high" for pin 5, turn on segment
c
digitalWrite(b,HIGH);// turn on segment b
for(j=7;j<=11;j++)// turn off other segments
digitalWrite(j,LOW);
digitalWrite(dp,LOW);// turn off segment dp
}
void digital_2(void) // display number 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 number 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 number 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 number 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 number 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 number 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 number 8
{
unsigned char j;
for(j=5;j<=11;j++)
digitalWrite(j,HIGH);
digitalWrite(dp,LOW);
}
void digital_9(void) // display number 5
{
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;// set variable
for(i=4;i<=11;i++)
pinMode(i,OUTPUT);// set pin 4-11as "output"
}
void loop()
{
while(1)
{
digital_0();// display number 0
delay(1000);// wait for 1s
digital_1();// display number 1
delay(1000);// wait for 1s
digital_2();// display number 2
delay(1000); // wait for 1s
digital_3();// display number 3
delay(1000); // wait for 1s
digital_4();// display number 4
delay(1000); // wait for 1s
digital_5();// display number 5
delay(1000); // wait for 1s
digital_6();// display number 6
```

```
delay(1000); // wait for 1s
digital_7();// display number 7
delay(1000); // wait for 1s
digital_8();// display number 8
delay(1000); // wait for 1s
digital_9();// display number 9
delay(1000); // wait for 1s
}
}
```

Result:
LED segment display displays number 0 to 9