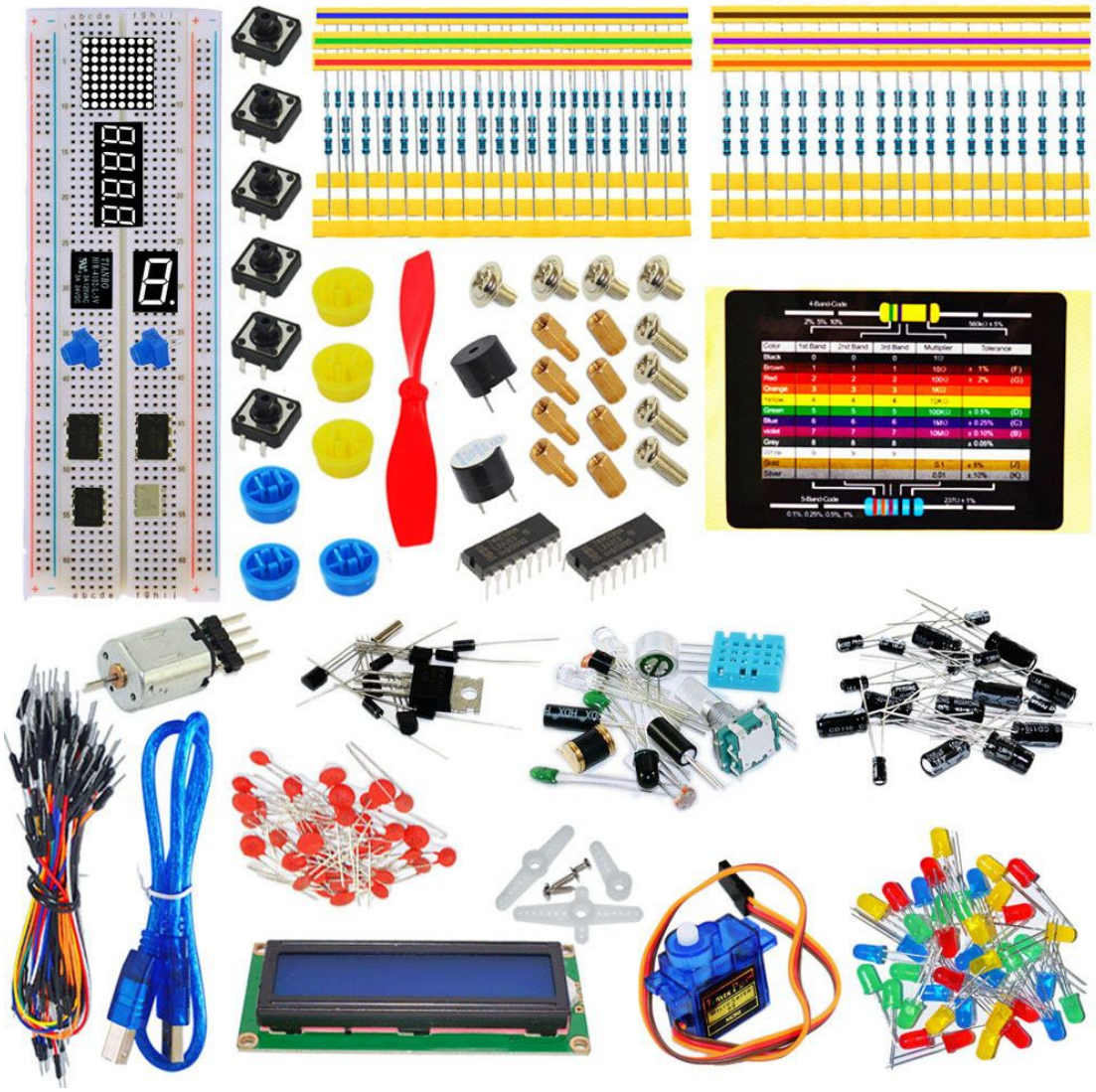
keyestudio

ARDUINO maker learning kit



www.keyestudio.cc

Catalog

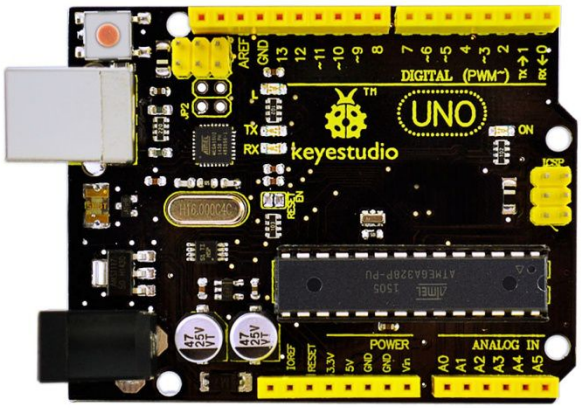
1. Introduction

Want to have enormous fun? Want to DIY some projects? Want to be more creative and more imaginative? Want your child to learn science while having fun? As long as you are willing to create, dare to experience new things, have a passion for scientific experiments, this maker kit is your best tailored choice!

Maker learning kit is a DIY kit for scientific experiments based on ARDUINO. Together with controller, sensors, electronic components, you can build different DIY projects. It can not only enhance operational ability of teenagers, but also develop their imagination and creativity.

Children who are into DIY can learn electronics, physics, science knowledge and software programming while playing; teachers can use it to achieve innovative teaching; makers can use it for design verification of product prototype.

Keyestudio UNO R3 board



Overview

Keyestudio Uno r3 is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Uno r3 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 programmed as a USB-to-serial converter.

Parameters

1. Microcontroller core: AVRmega328P-PU (Processing speed 20MIPS)
2. Working voltage: +5V
3. External input voltage：+7V~+12V（suggest）
4. External input voltage (extremum): +6V≤ Vin ≤ ＋20V
5. Digital signal I/O interface: 14 (6 PWM input interface)
6. Analog signal input interface: 6
7. DC I/O interface current: 40mA
8. Flash capacity: 32KB(other 2k used in hootloader)
9. SRAM static storage capacity
10. EEPROM storage capacity: 512bytes
11. Clock frequency：16MHZ

user

1 | Download the Arduino environment

Get the latest version from the download page.

When the download finishes, unzip the downloaded file. Make sure to preserve the folder structure. Double-click the folder to open it. There should be a few files and sub-folders inside.

2 | Connect the board

The Arduino Uno, Mega, Duemilanove and Arduino Nano automatically draw power from either the USB connection to the computer or an external power supply. If you're using an Arduino Diecimila, you'll need to make sure that the board is configured to draw power from the USB connection. The power source is selected with a jumper, a small piece of plastic that fits onto two of the three pins between the USB and power jacks. Check that it's on the two pins closest to the USB port.

Connect the Arduino board to your computer using the USB cable. The green power LED (labelled PWR) should go on.

3 | Install the drivers

Installing drivers for the Arduino Uno or Arduino Mega 2560 with Windows 7, Vista, or XP:

Plug in your board and wait for Windows to begin it's driver installation process. After a few moments, the process will fail, despite its best efforts

Click on the Start Menu, and open up the Control Panel.

While in the Control Panel, navigate to System and Security. Next, click on System. Once the System window is up, open the Device Manager.

Look under Ports (COM & LPT). You should see an open port named "Arduino UNO (COMxx)".

If there is no COM & LPT section, look under "Other Devices" for "Unknown Device".

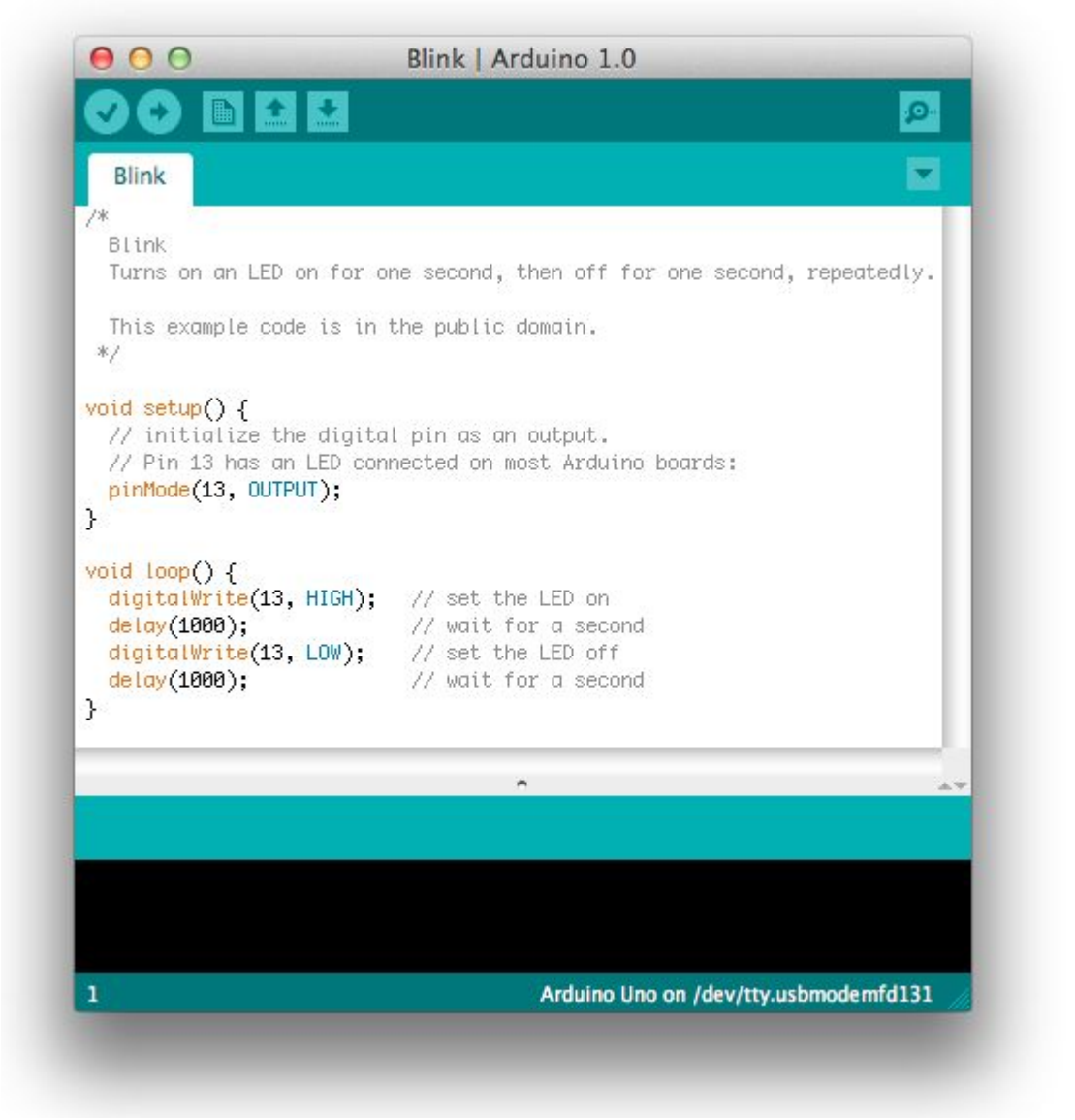
Right click on the "Arduino UNO (COmxx)" port and choose the "Update Driver Software" option.

Next, choose the "Browse my computer for Driver software" option.

Finally, navigate to and select the driver file named "arduino.inf", located in the "Drivers" folder of the Arduino Software download (not the "FTDI USB Drivers" sub-directory). If you are using an old version of the IDE (1.0.3 or older), choose the Uno driver file named "Arduino UNO.inf" Windows will finish up the driver installation from there.
See also: step-by-step screenshots for installing the Uno under Windows XP.
Installing drivers for the Arduino Duemilanove, Nano, or Diecimila with Windows7, Vista, or XP:
When you connect the board, Windows should initiate the driver installation process (if you haven't used the computer with an Arduino board before).
On Windows Vista, the driver should be automatically downloaded and installed. (Really, it works!)
On Windows XP, the Add New Hardware wizard will open:
When asked Can Windows connect to Windows Update to search for software? select No, not this time. Click next.
Select Install from a list or specified location (Advanced) and click next.
Make sure that Search for the best driver in these locations is checked; uncheck Search removable media; check Include this location in the search and browse to the drivers/FTDI USB Drivers directory of the Arduino distribution. (The latest version of the drivers can be found on the FTDI website.) Click next.
The wizard will search for the driver and then tell you that a "USB Serial Converter" was found. Click finish.
The new hardware wizard will appear again. Go through the same steps and select the same options and location to search. This time, a "USB Serial Port" will be found.
You can check that the drivers have been installed by opening the Windows Device Mananger (in the Hardware tab of System control panel). Look for a "USB Serial Port" in the Ports section; that's the Arduino board.

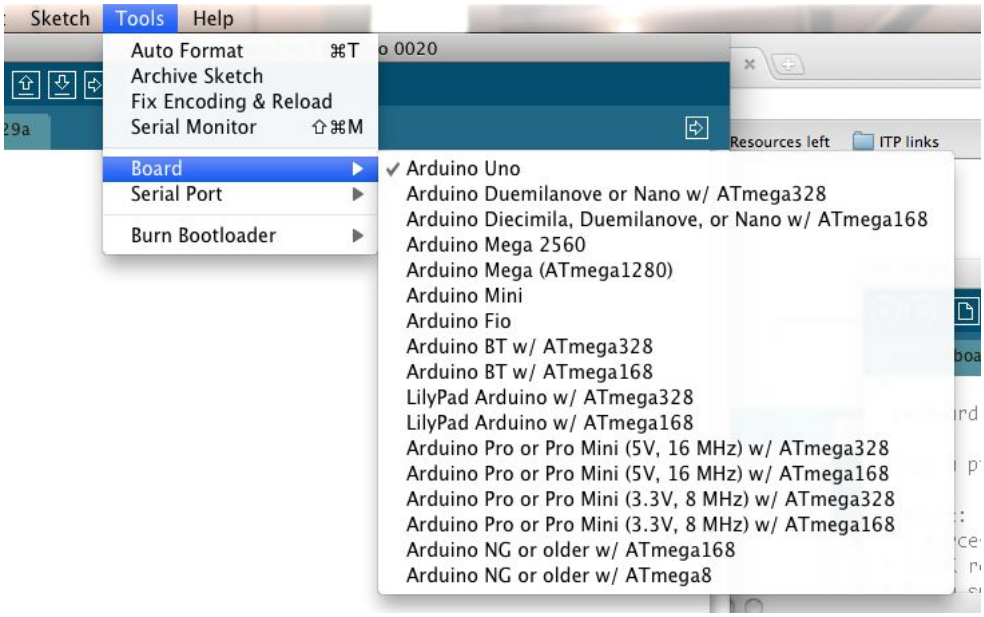
4 | Launch the Arduino application
Double-click the Arduino application. (Note: if the Arduino software loads in the wrong language, you can change it in the preferences dialog. See the environment page for details.)

5 | Open the blink example
Open the LED blink example sketch: File > Examples > 1.Basics > Blink.



6 | Select your board

You'll need to select the entry in the Tools > Board menu that corresponds to your Arduino.



Selecting an Arduino Uno

For Duemilanove Arduino boards with an ATmega328 (check the text on the chip on the board), select Arduino Duemilanove or Nano w/ ATmega328. Previously, Arduino boards came with an ATmega168; for those, select Arduino Diecimila, Duemilanove, or Nano w/ ATmega168. (Details of the board menu entries are available on the environment page.)

7 | Select your serial port

Select the serial device of the Arduino board from the Tools | Serial Port menu. This is likely to be COM3 or higher (COM1and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu; the entry that disappears should be the Arduino board. Reconnect the board and select that serial port.

8 | Upload the program

Now, simply click the "Upload" button in the environment. Wait a few seconds - you should see the RX and TX leds on the board flashing. If the upload is successful, the message "Done uploading." will appear in the status bar. (Note: If you have an Arduino Mini, NG, or other board, you'll need to physically present the reset button on the board immediately before pressing the upload button.)

A few seconds after the upload finishes, you should see the pin 13 (L) LED on the board start to blink (in orange). If it does, congratulations! You've gotten Arduino up-and-running.

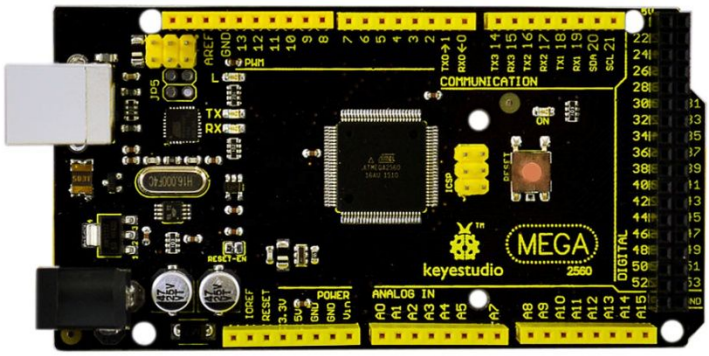
If you have problems, please see the troubleshooting suggestions.

You might also want to look at:

the examples for using various sensors and actuators

the reference for the Arduino language

Keyestudio Mega 2560 R3 board



Introduction

Keyestudio Mega (core to ATmega2560) is a development board (used with16MHz crystal oscillator) of microcontroller. There 54 groups of I/O (input/output) digital ends (of which 14 group to do PWM output), 16 groups of simulation analogy input ends and 4 groups of UART (hardwareserial ports) in it . Because its bootloader, process can be downloaded directly with the USB and you don't need to use other external programmer. And its power can be supplied by the USB, or the AC-to-DC adapter and battery can be also as an external power supply.

Opening source code and using C language developed status in Java concept (cross platform) make a rapid growth of Arduino peripheral module and application. The main reason to attract Artist to use Arduino is that they can quickly use all kinds of software communication such as Arduino language and Flash or Processing and so on. and make multimedia interactive works. Development interface of Arduino IDE is based on the principle of opening source code, which you can download freely used in the thematic making, school teaching, television controlling, interactive works and so on.

Design of Power Supply

There are two choices (direct power supply trough USB or external power supply) for the power supply system of Arduino Mega, and they can be Automatic switched. External power supply can be AC-to-DC adapter or battery. Lit rang of voltage of this control board is 6V~12V, but if

the supplied voltage is greater than 12V, the voltage stabilizing device will be likely overheated and overheat protection and damaging Arduino MEGA will be more likely to occur. So we suggest the power supply should be 6.5~12V in operation and recommended supply is 7.5 or 9V.

Summary

Microcontroller:	ATmega2560
Operating Voltage:	5V
Input Voltage (recommended):	7-12V
Input Voltage (limits):	6-20V
Digital I/O Pins:	54 (of which 15 provide PWM output)
Analog Input Pins:	16
DC Current per I/O Pin:	40 mA
DC Current for 3.3V Pin:	50 mA
Flash Memory:	256 KB of which 8 KB used by bootloader
SRAM:	8 KB
EEPROM:	4 KB
Clock Speed:	16 MHz

Procedure for Installing Arduino Driver

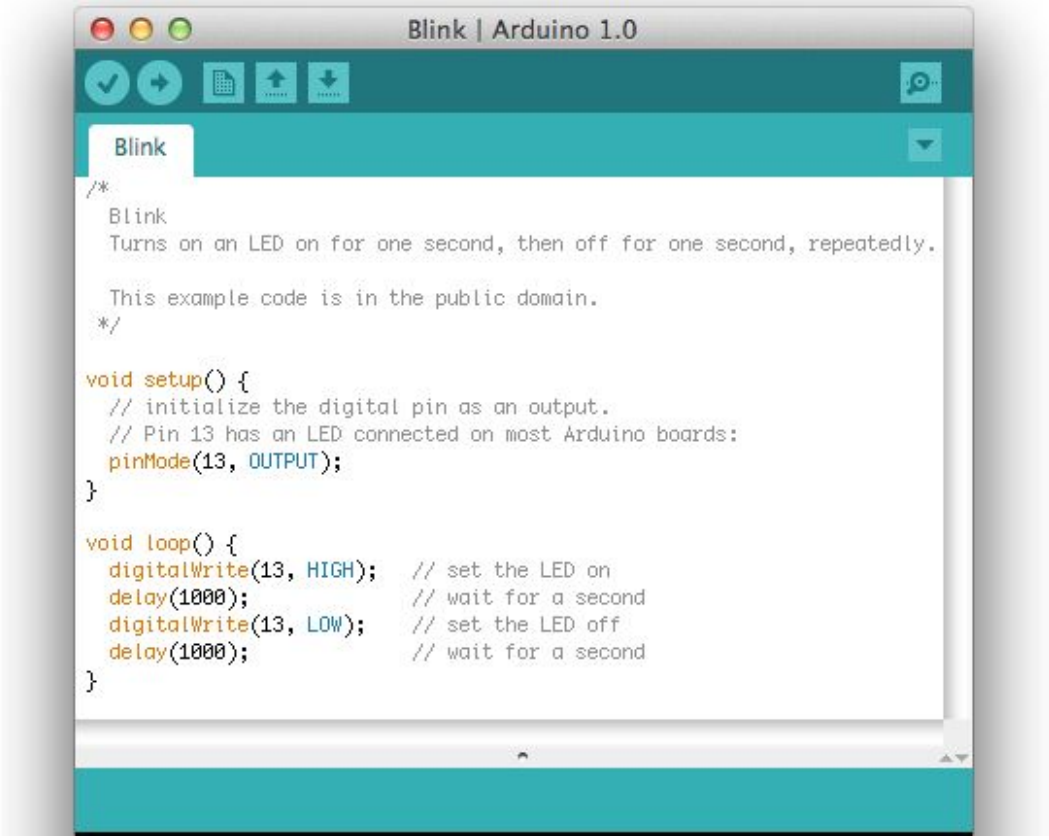
To download the Arduino developing software on the web address:
http:/arduino.cc/en/Main/Software. The downloaded file is arduino-1.0.zip,
a compressed folder, to decompress it to the hard disk.
When 2560R3 Developing Board is connected to the Windows through the
USB line, Windows will prompt a new USB device is found, then it will lead us into the "found new hardware wizard" window.

The next step is to install 2560R3 driver required, selecting the option of "install from a list or specific location (Senior)" and click "next" button:
To put the driver into the driver directory of Arduino 1.0 installation directory, and we
need to specify this directory to be the searched directory when installing the driver.

Click "next" button, Windows begins to find and install Arduino driving procedure.
If all goes well, we will see the success interface as follows:

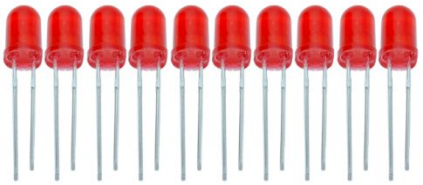
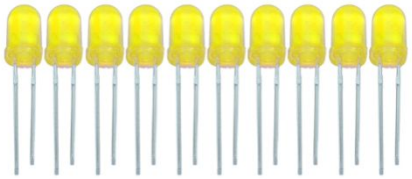
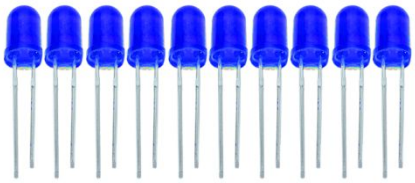
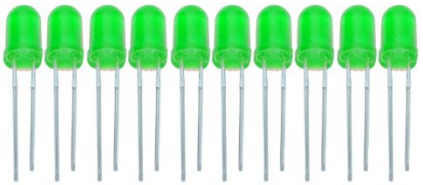
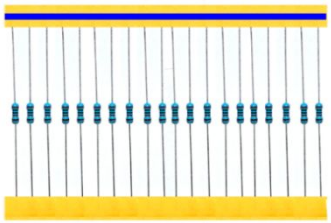
After the installation of Arduino driver is successful, we can find the corresponding Arduino serial port in the Windows device manager:

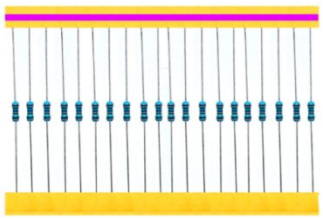
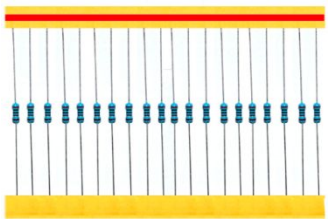
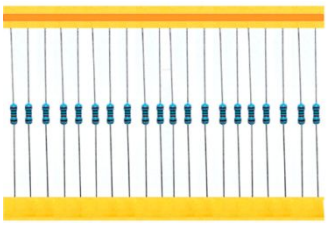
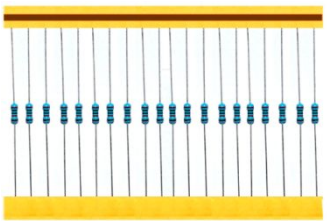
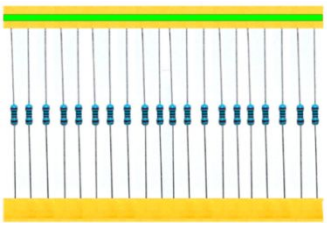
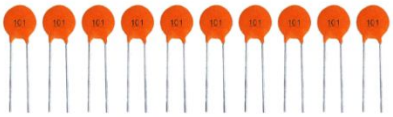
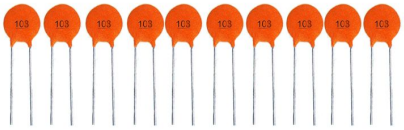
Well, the next is to test driver installation.
Testing code:

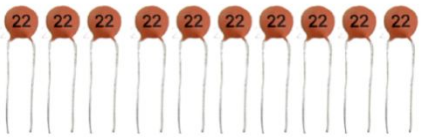
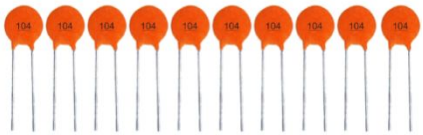
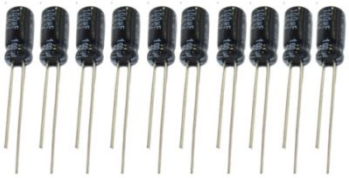
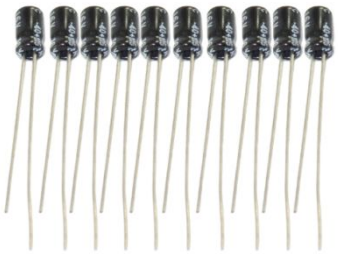


Copy the code above to Arduino status , select the model 2560 and port, and then upload the code. To wait a moment and the results came out, then you will see the LED flashing at D13 of your 2560r3 board and the time interval is 1s, and then we know that is ok.

2. Component list

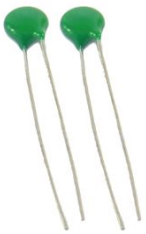
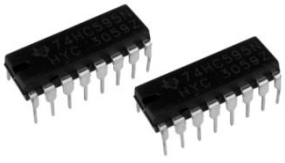
No.	Product Name	Quantity	Picture
1	LED - Red	10	
2	LED - Yellow	10	
3	LED - Blue	10	
4	LED - Green	10	
5	LED - RGB	2	
6	220 Ω resistor	20	

7	100K Ω resistor	20	
8	1K Ω resistor	20	
9	4.7K Ω resistor	20	
10	47K Ω resistor	20	
11	10K Ω resistor	20	
12	101 ceramic capacitor	10	
13	103 ceramic capacitor	10	

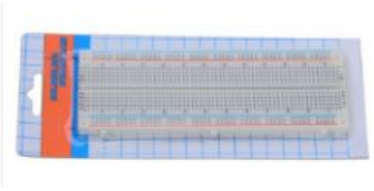
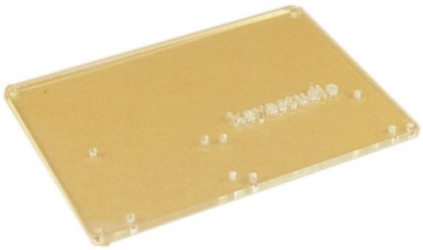
14	22 ceramic capacitor	10	
15	104 ceramic capacitor	10	
16	100uf16V electrolytic capacitor	10	
17	10uf16V electrolytic capacitor	10	
18	Button	6	
19	Yellow round cap	3	
20	Blue round cap	3	
21	4007 diode	5	
22	8050 Transistor	2	

23	8550 Transistor	2	
24	1-digit 7-seg LED (small)	1	
25	4-digit 7-seg LED (small)	1	
26	LED Matrix (small)	1	
27	5V Relay	1	
28	MOS (metal oxide semiconductor) tube	1	
29	Crystal oscillator	1	

30	801S sensor	1	
31	Highly sensitive MIC	1	
32	Rotary encoder	1	
33	DHT11 temperature and humidity	1	
34	LM35 temperature sensor	1	
35	Flame sensor	1	
36	Ball tilt sensor	2	

37	103 thermistor	2	
38	Photoresistor	2	
39	103 variable resistor	2	
40	4N35	1	
41	NE555P	1	
42	DS1302	1	
43	595 IC	2	
44	Active buzzer	1	

45	Passive buzzer	1	
46	Pin header 1*16	1	
47	Fan leaf	1	
48	Fan motor	1	
49	9G servo motor	1	
50	1602 LCD	1	
51	USB cable 100mm	1	
52	Jumper Wire 1*65	1	

53	830-hole Breadboard	1	
54	Retaining screws	1	
55	Acrylic fixed platform	1	

3. Project list

Project 1: Hello World
Project 2: LED blinking
Project 3: PWM
Project 4: Traffic light
Project 5: LED chasing effect
Project 6: Button-controlled LED
Project 7: Responder
Project 8: Active buzzer
Project 9: Passive buzzer
Project 10: RGB LED
Project 11: Analog value reading
Project 12: Photo resistor
Project 13: Flame sensor
Project 14: Analog temperature (thermistor)
Project 15: LM35 temperature sensor
Project 16: Temperature-controlled cup
Project 17: Temperature and humidity sensor

Project 18: Tilt switch
Project 19: Magical Light Cup
Project 20: Vibration switch
Project 21: Sound-control light
Project 22: Voltmeter
Project 23: 74HC595
Project 24: 1-digit LED segment display
Project 25: 4-digit LED segment display
Project 26: 8*8 LED matrix
Project 27: 1602 LCD
Project 28: 9g servo control
Project 29: Rotary encoder
Project 30: 5V Relay
Project 31: DS1302 clock
Project 32: Mos tube driving motor
Project 33: 4N35
Project 34: NE555 timer

4. 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

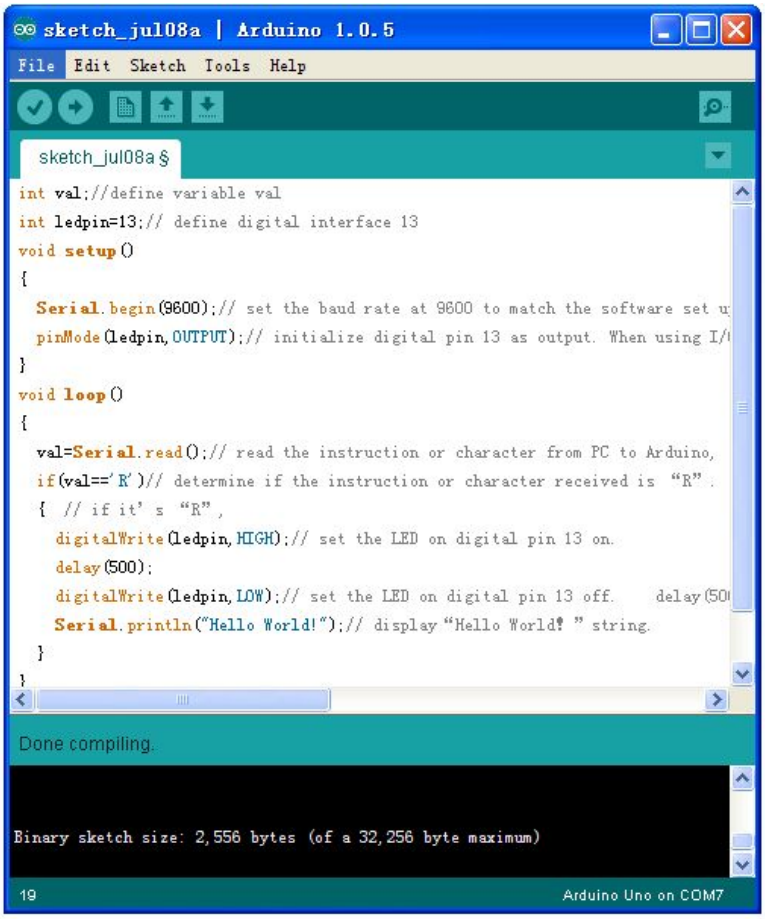
Arduino board *1
USB cable *1

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

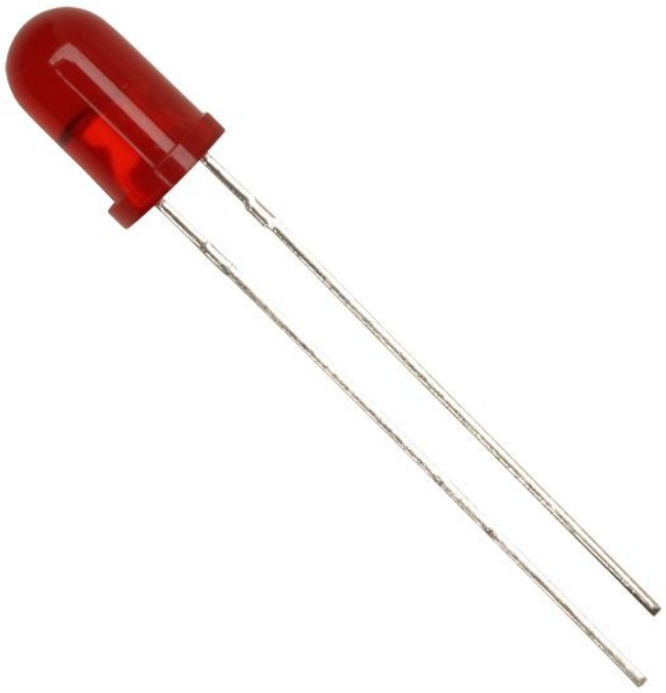


Click serial port monitor,Input R,LED 13 will blink once,PC will receive information from Arduino: Hello World

After you choosing 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:

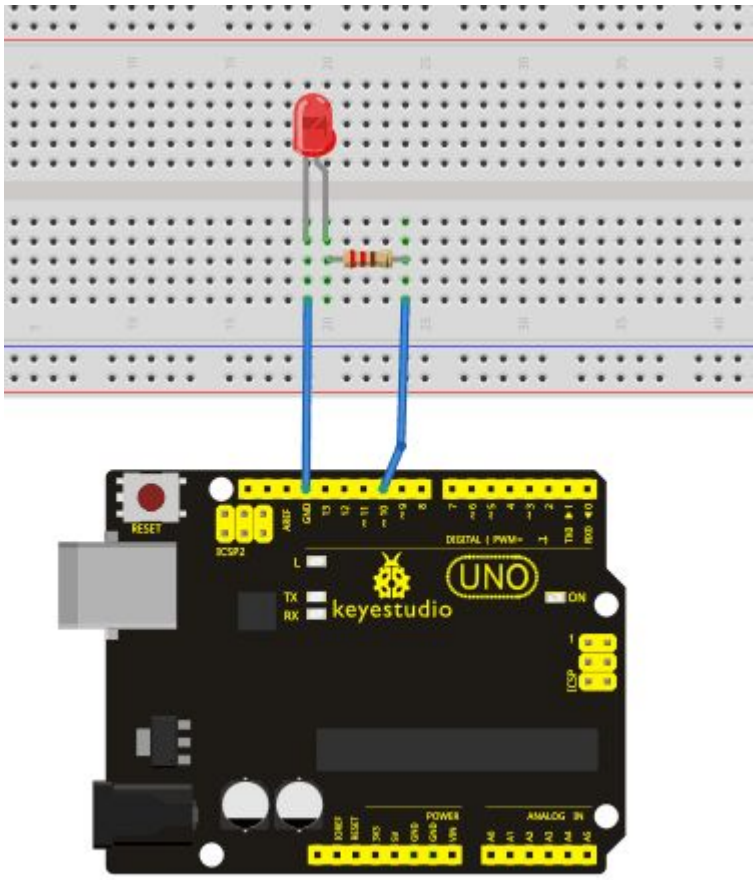
Hardware required

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

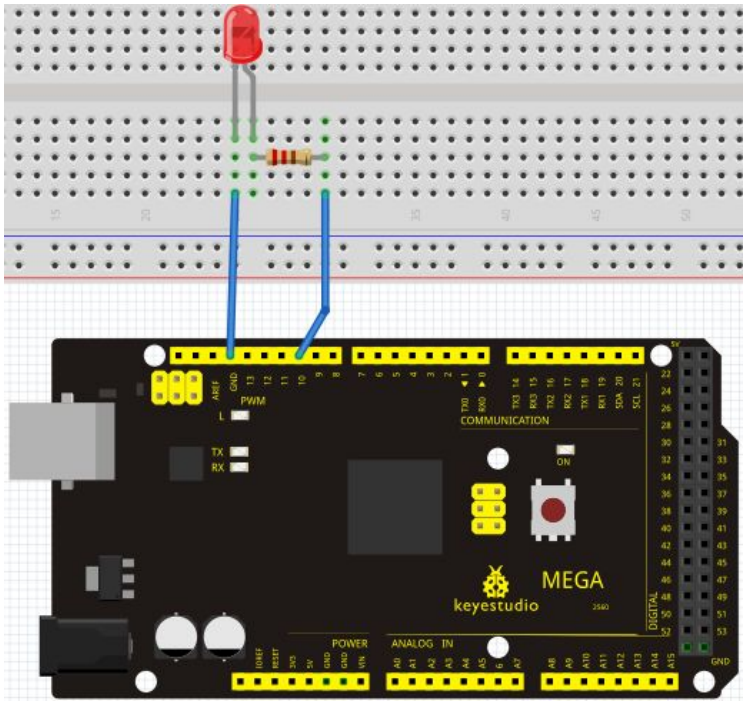
Circuit connection

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 for Uno R3:



Connection for 2560 R3:



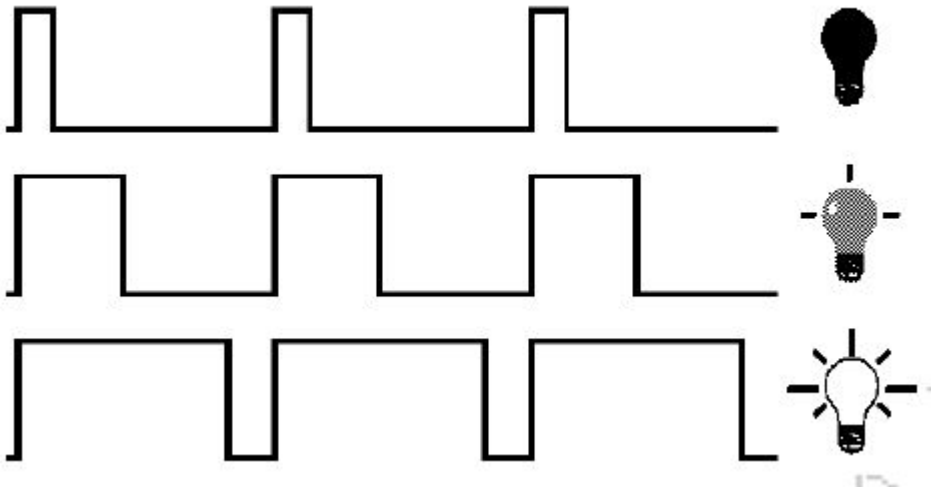
Sample program

```
//////////////////////////////////////////////////////////
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

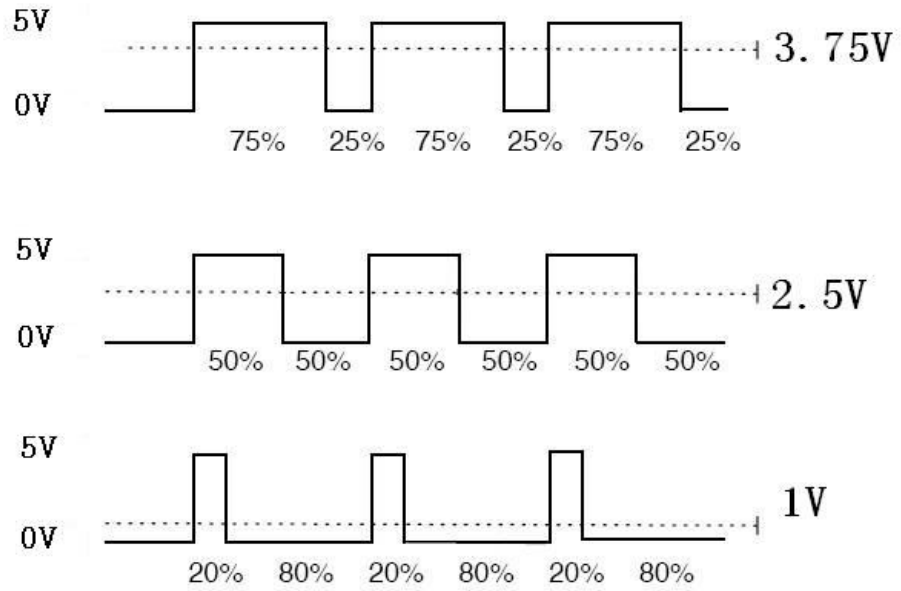
After downloading this program, in the experiment, you will see the LED connected to pin 10 turning on and off, with an interval approximately one second.
The blinking LED experiment is now completed. Thank you!
****

Project 3: PWM



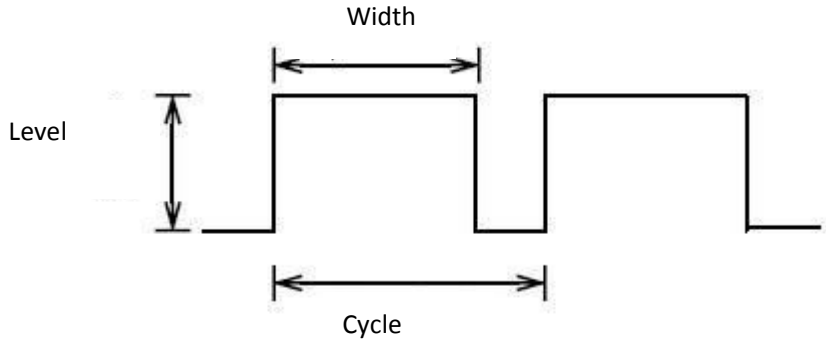
Introduction

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



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

The following are the three basic parameters of PMW:



1. The amplitude of pulse width (minimum / maximum)
2. The pulse period (The reciprocal of pulse frequency in 1 second)

3. The voltage level（such as：0V-5V）

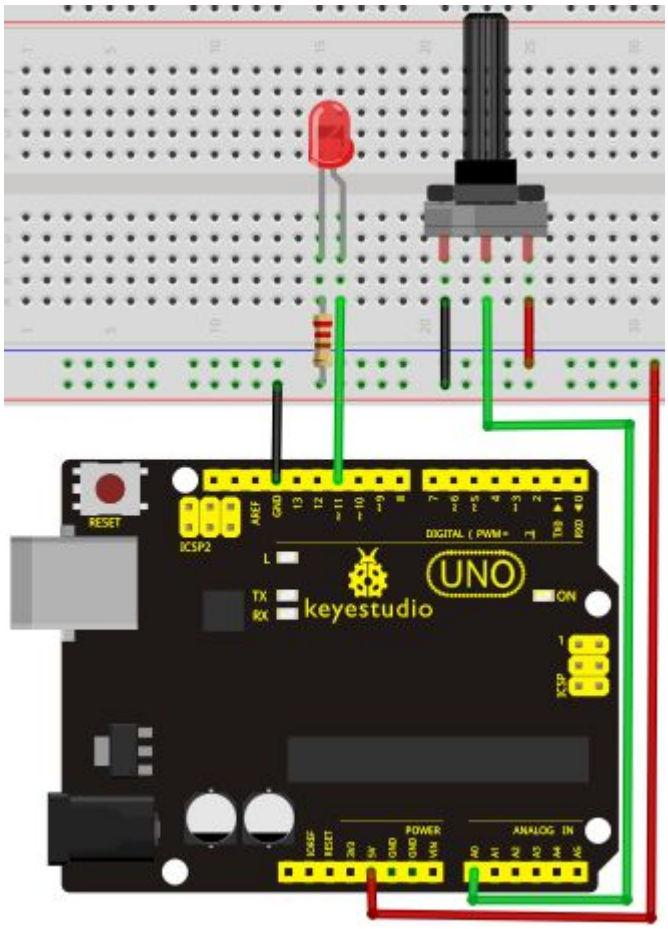
There are 6 PMW interfaces on Arduino, namely digital pin 3, 5, 6, 9, 10, and 11. In previous experiments, we have done "button-controlled LED", using digital signal to control digital pin, also one about potentiometer. This time, we will use a potentiometer to control the brightness of the LED.

Hardware required

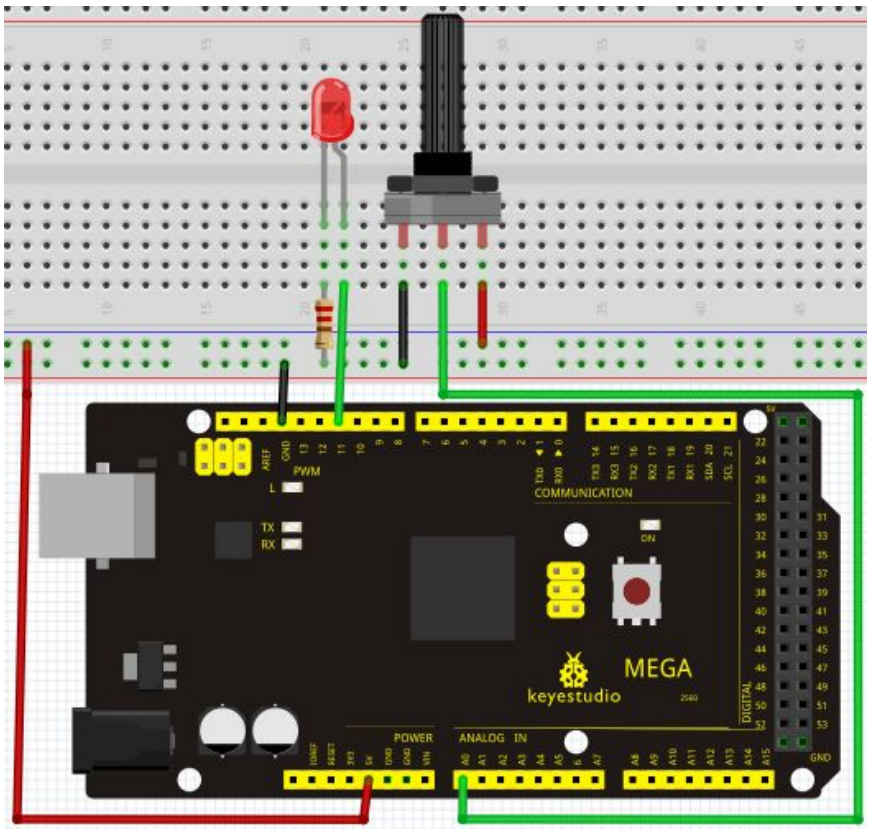
Potentiometer module*1
Red M5 LED*1
220Ω resistor
Breadboard*1
Breadboard jumper wires

Circuit connection

The input of potentiometer is analog, so we connect it to analog port, and LED to PWM port. Different PWM signal can regulate the brightness of the LED.
Connection for R3:



Connection for 2560 R3:



Sample program

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

```
//////////////////////////////////////////////////////////
int potpin=0;// initialize analog pin 0
int ledpin=11;//initialize digital pin 11（PWM output）
int val=0;// Temporarily store variables' value from the sensor
void setup()
{
pinMode(ledpin,OUTPUT);// define digital pin 11 as “output”
Serial.begin(9600);// set baud rate at 9600
// attention: for analog ports, they are automatically set up as “input”
}
void loop()
{
```

```
val=analogRead(potpin);// read the analog value from the sensor and assign it to val
Serial.println(val);// display value of val
analogWrite(ledpin,val/4);// turn on LED and set up brightness（maximum output of PWM is 255）
delay(10);// wait for 0.01 second
}
//////////////////////////////////////////////////////
```

Result

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

****

Project 4: Traffic light

Introduction

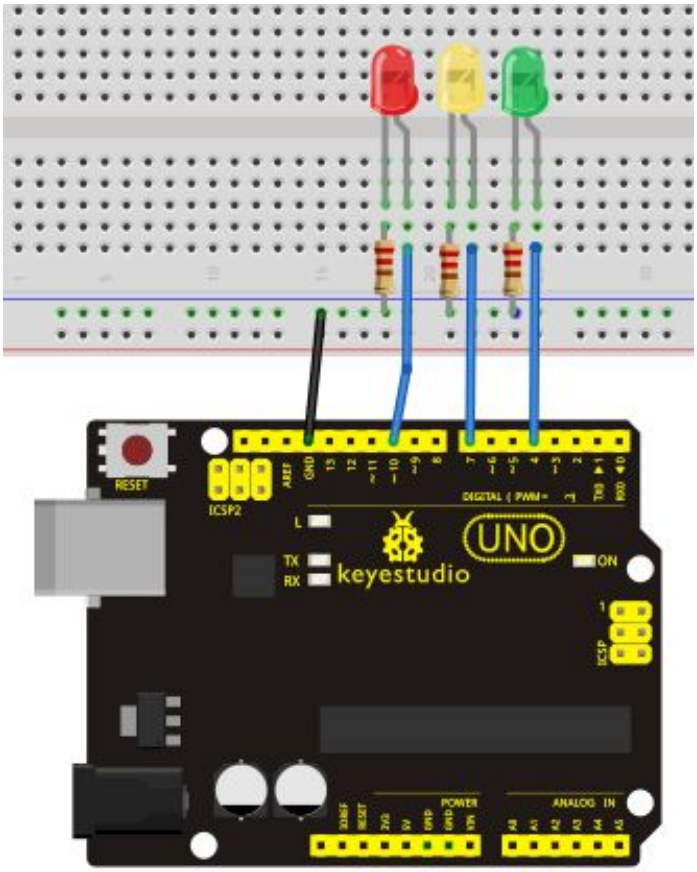
In the previous program, we have done the LED blinking experiment with one LED. Now, it's time to up the stakes and do a bit more complicated experiment-traffic lights. Actually, these two experiments are similar. While in this traffic lights experiment, we use 3 LEDs with different color other than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

Hardware required

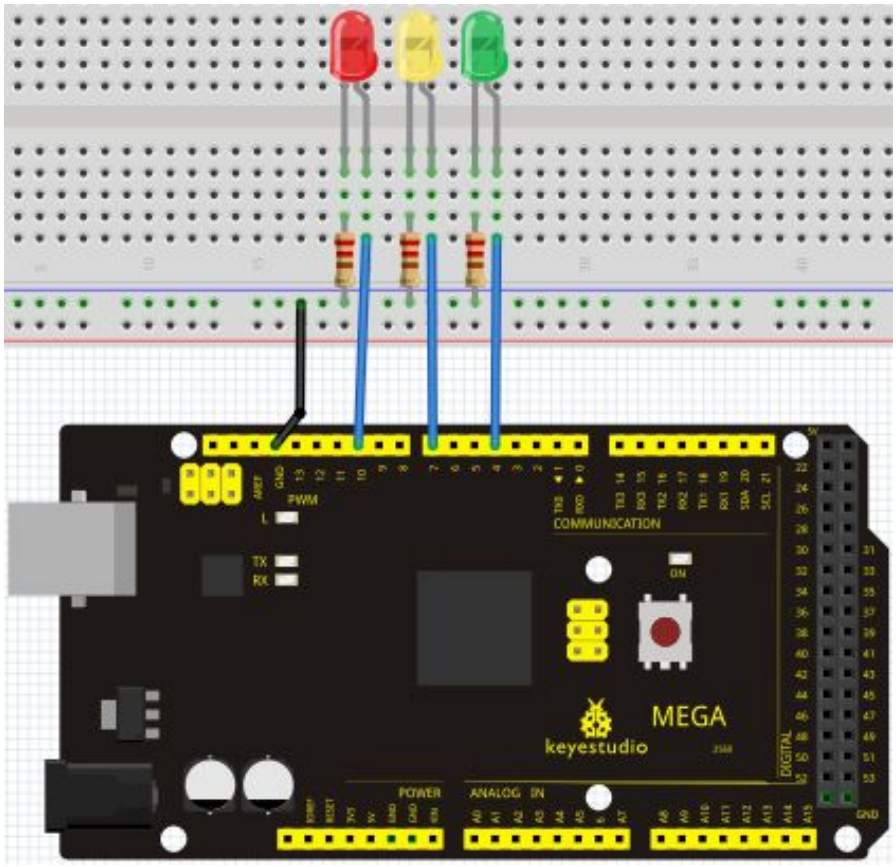
Arduino board *1
USB cable *1
Red M5 LED*1
Yellow M5 LED*1
Green M5 LED*1
220Ω resistor *3
Breadboard*1
Breadboard jumper wires

Circuit connection

Connection for R3:



Connection for 2560 R3:



Sample program

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.

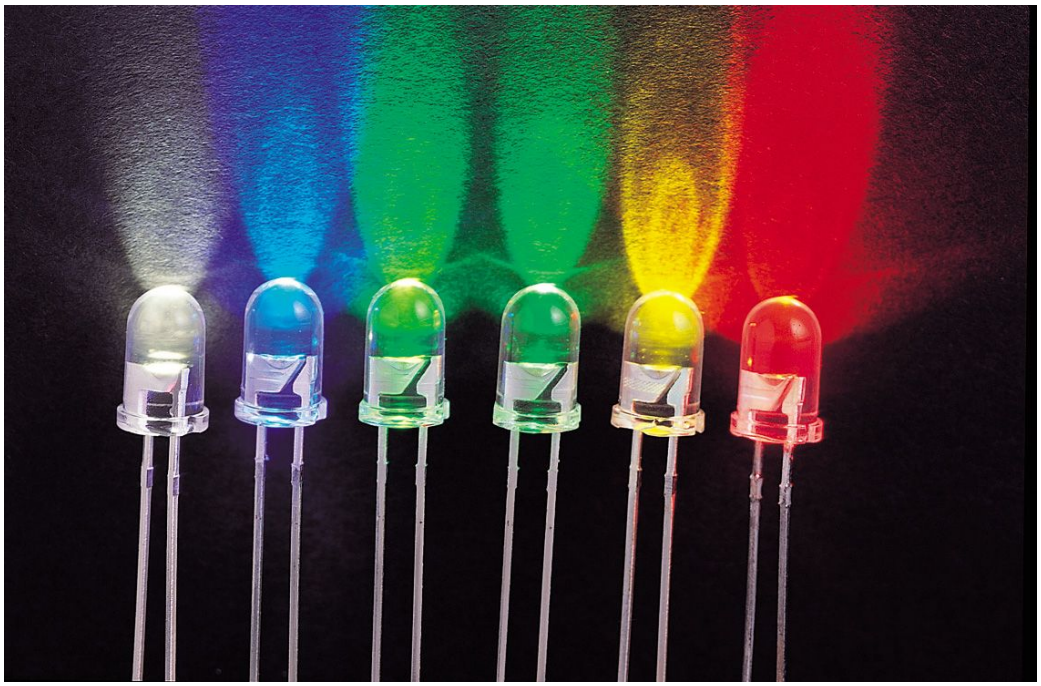
```
//////////////////////////////////////////////////////////
int redled =10; // initialize digital pin 8.
int yellowled =7; // initialize digital pin 7.
int greenled =4; // initialize digital pin 4.
void setup()
{
pinMode(redled, OUTPUT);// set the pin with red LED as “output”
pinMode(yellowled, OUTPUT); // 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 5: LED chasing effect



Introduction

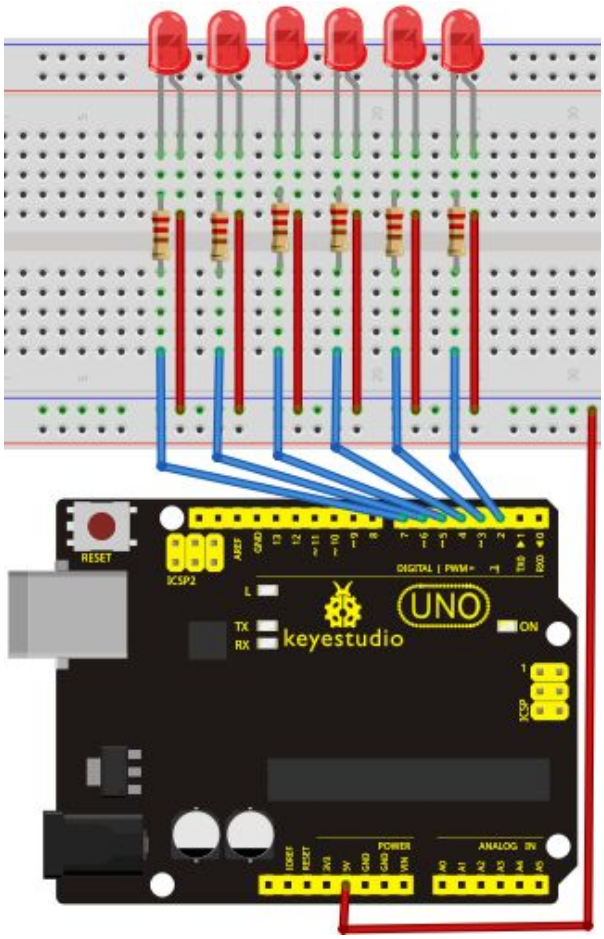
We often see billboards composed of colorful LEDs. They are constantly changing to form various effects. In this experiment, we compile a program to simulate chase effect.

Hardware required

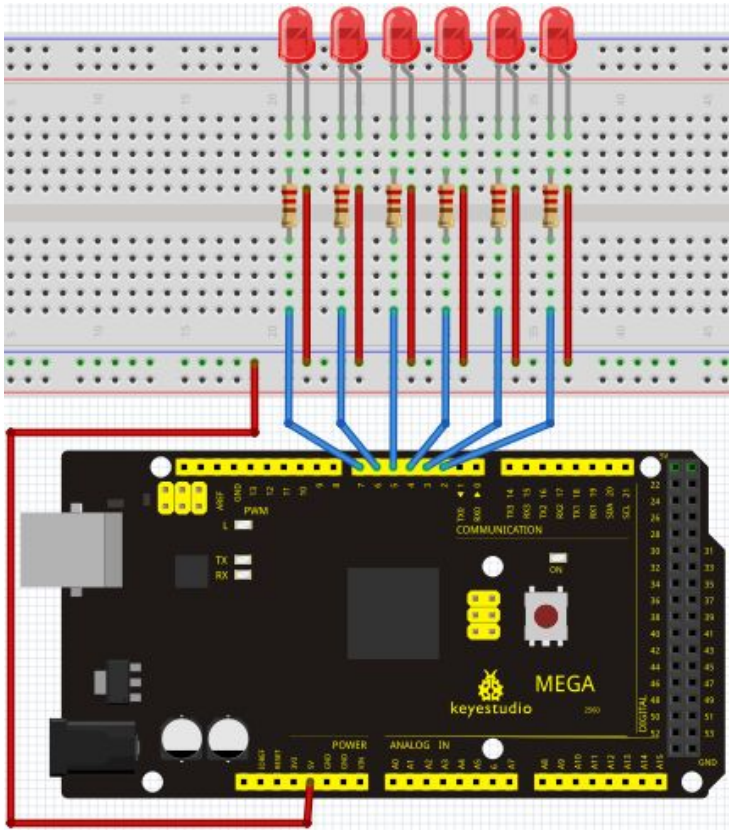
Led *6
220Ω resistor *6
Breadboard jumper wires

Circuit connection

Connection for R3:



Connection for 2560 R3:



Sample program

```
//////////////////////////////////////////////////////////
int BASE = 2 ;    // the I/O pin for the first LED
int NUM = 6;      // 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 6: 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.

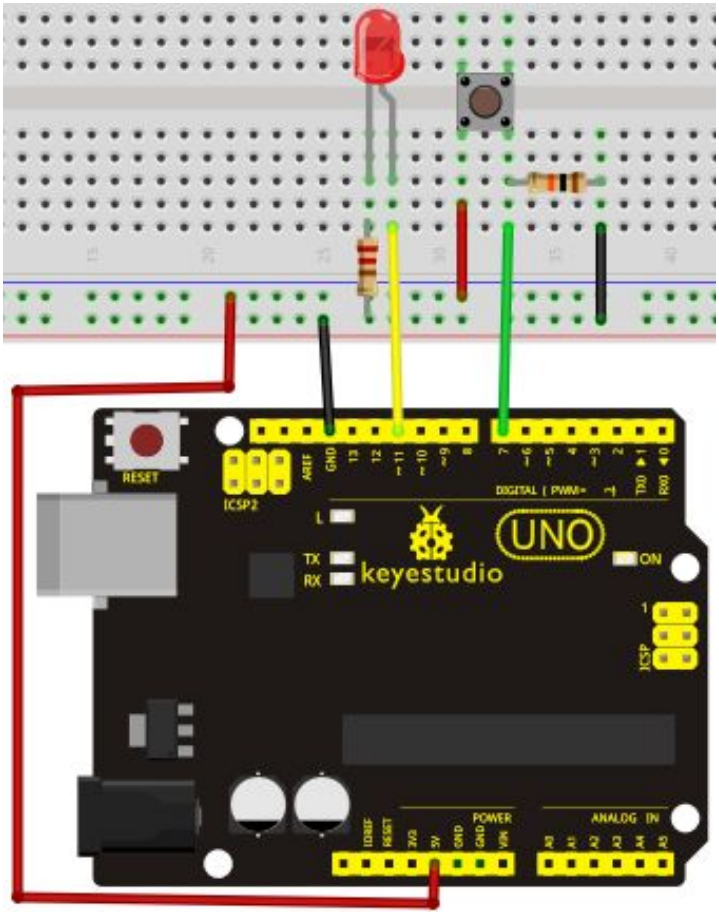
Hardware required

Button switch*1
Red M5 LED*1
220Ω resistor*1
10KΩ resistor*1
Breadboard*1

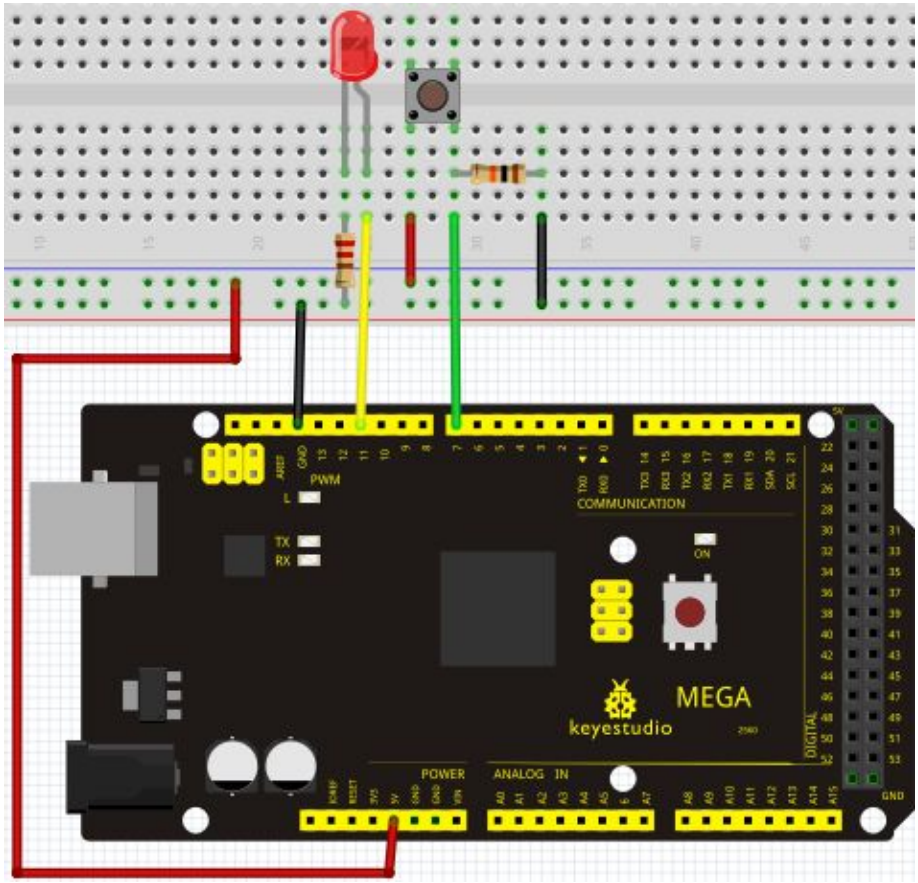
Breadboard jumper wires

Circuit connection

Connection for R3:



Connection for 2560 R3:



Sample program

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.

****

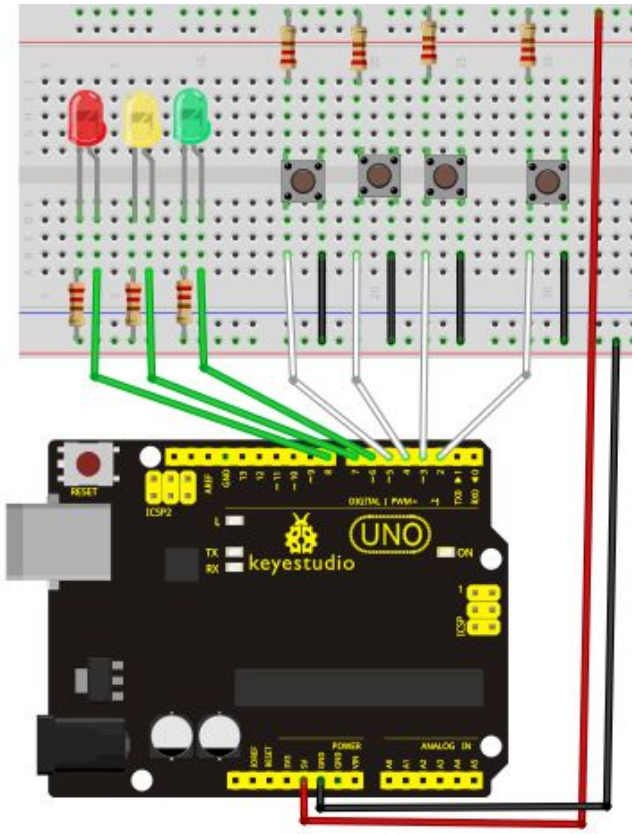
Project 7: Responder experiment

Introduction

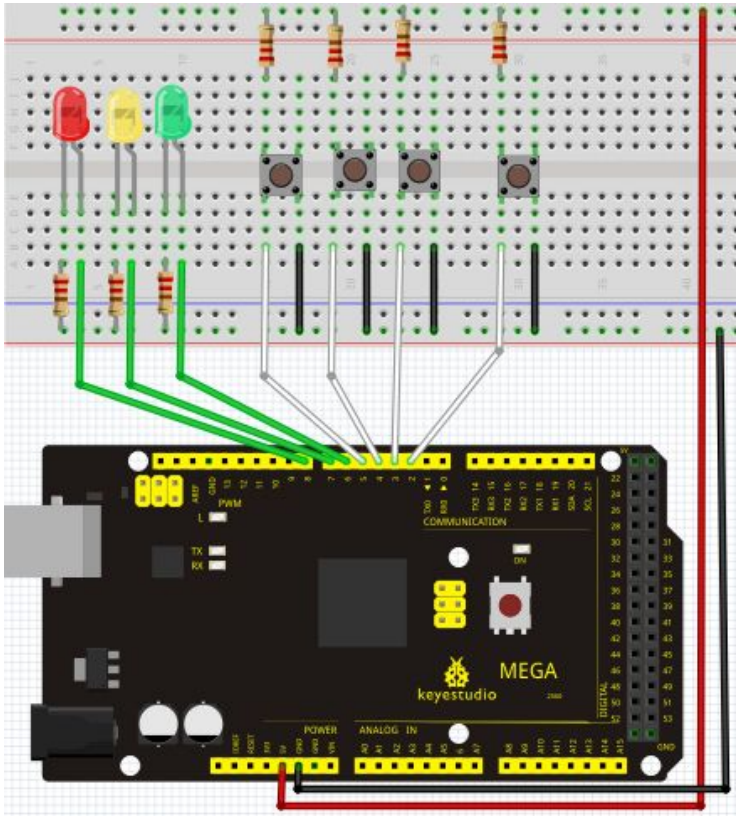
After completing all the previous experiments, we believe you will find this one easy. In this program, we have 3 buttons and a reset button controlling the corresponding 3 LEDs, using 7 digital I/O pins.

Circuit connection

Connection for R3:



Connection for 2560 R3:



Sample program

```
//////////////////////////////////////////////////////////
int redled=8;          // set red LED as “output”
int yellowled=7;    // set yellow LED as “output”
int greenled=6;      // set green LED as “output”
int redpin=5;          // initialize pin for red button
int yellowpin=4;    // initialize pin for yellow button
int greenpin=3;      // initialize pin for green button
int restpin=2;      // initialize pin for reset button
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()    // repeatedly read pins for buttons
{
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()// execute the code until red light is on; end cycle when reset button is pressed
{
   while(digitalRead(restpin)==1)
   {
     digitalWrite(redled,HIGH);
     digitalWrite(greenled,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(greenled,LOW);
  digitalWrite(yellowled,HIGH);
  }
  clear_led();
}
void GREEN_YES()// execute the code until green light is on; end cycle when reset button is pressed
{
  while(digitalRead(restpin)==1)
  {
  digitalWrite(redled,LOW);
  digitalWrite(greenled,HIGH);
  digitalWrite(yellowled,LOW);
  }
  clear_led();
}
void clear_led()// all LED off
{
  digitalWrite(redled,LOW);
  digitalWrite(greenled,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 8: Active buzzer

Introduction

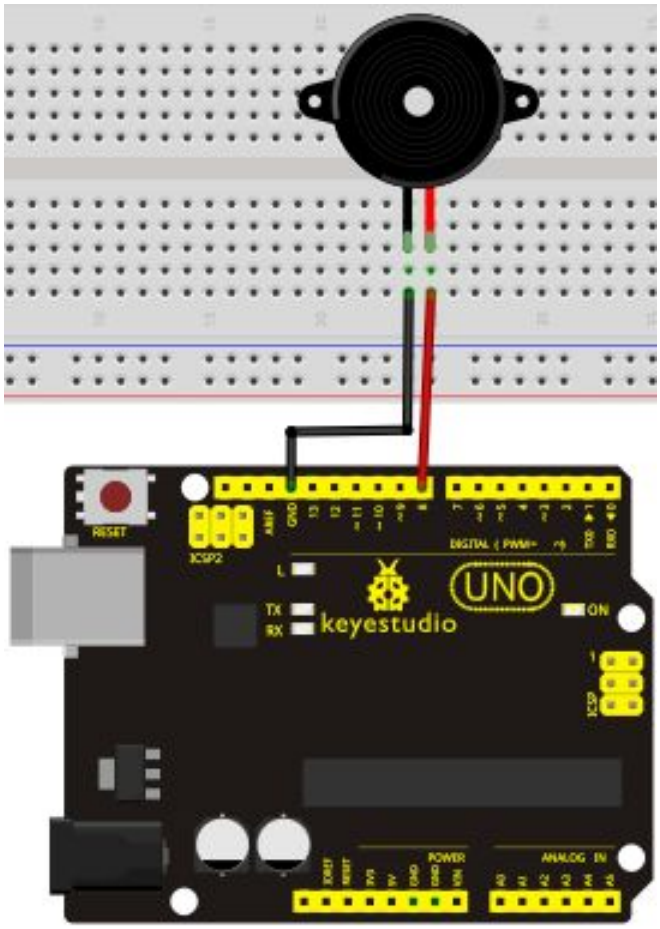
Arduino enables us to make many interesting interactive projects, many of which we have done consists of a LED. They are light-related. While this time, the circuit will produce sound. The sound experiment is usually done with a buzzer or a speaker, while buzzer is simpler and easier to use. The buzzer 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 fun and simple.

Hardware required

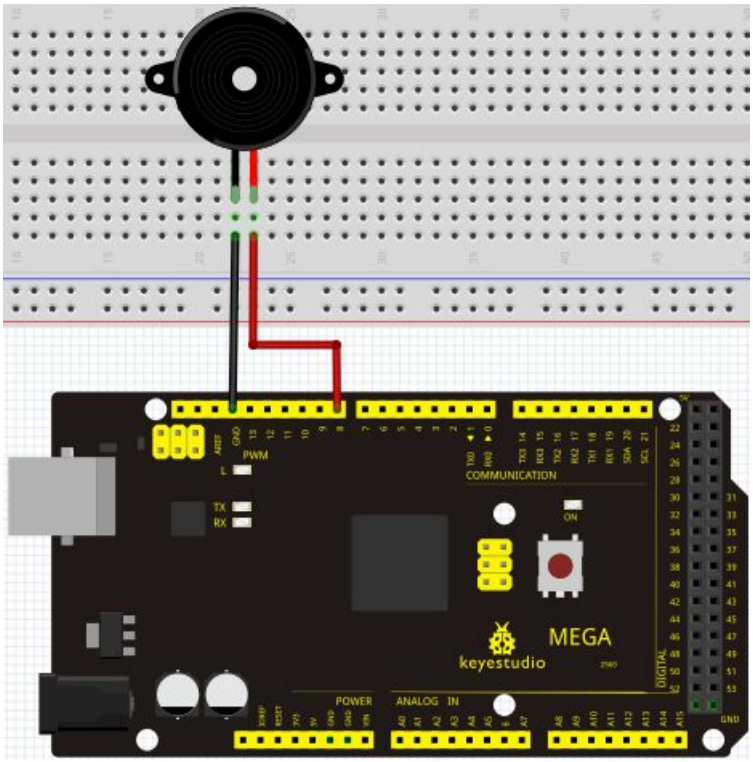
Buzzer*1
Key *1
Breadboard*1
Breadboard jumper wires

Circuit connection

Connection for R3:



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 program

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

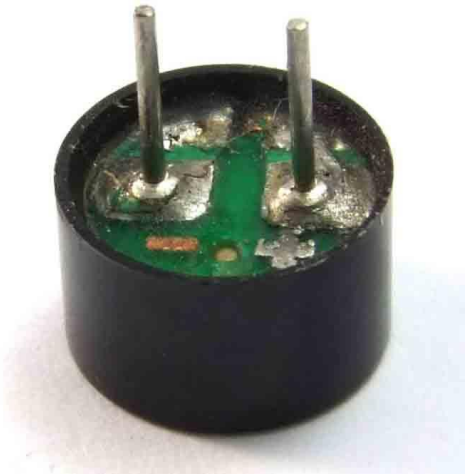
```
//////////////////////////////////////////////////////////
int buzzer=8;// initialize digital IO pin that controls the buzzer
void setup()
{
   pinMode(buzzer,OUTPUT);// set pin mode as “output”
}
void loop()
{
digitalWrite(buzzer, HIGH); // produce sound
}
//////////////////////////////////////////////////////////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

Result

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

****

Project 9: Passive buzzer



Introduction

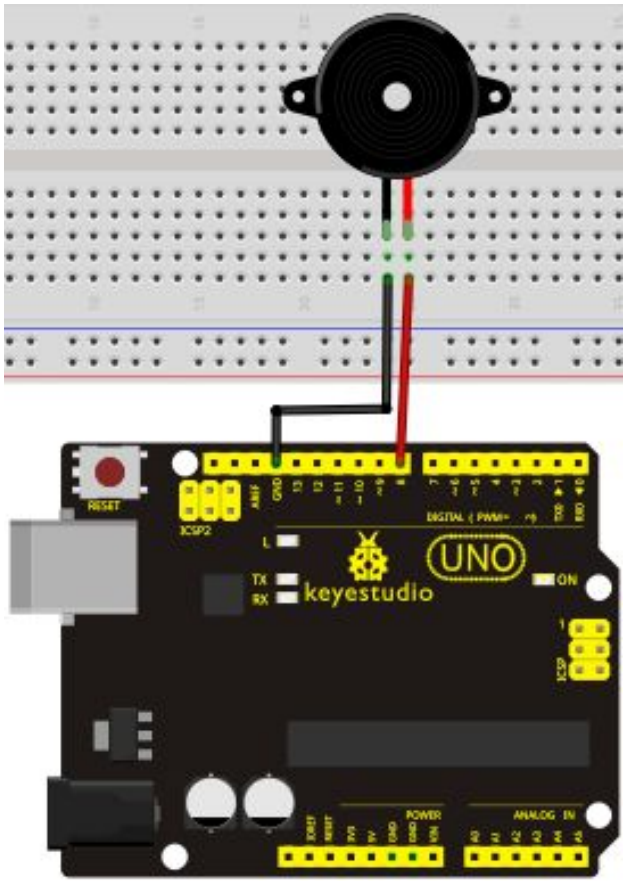
We can use Arduino to make many interactive works of which the most commonly used is acoustic-optic display. All the previous experiment has something to do with LED. However, the circuit in this experiment can produce sound. Normally, the experiment is done with a buzzer or a speaker while buzzer is 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

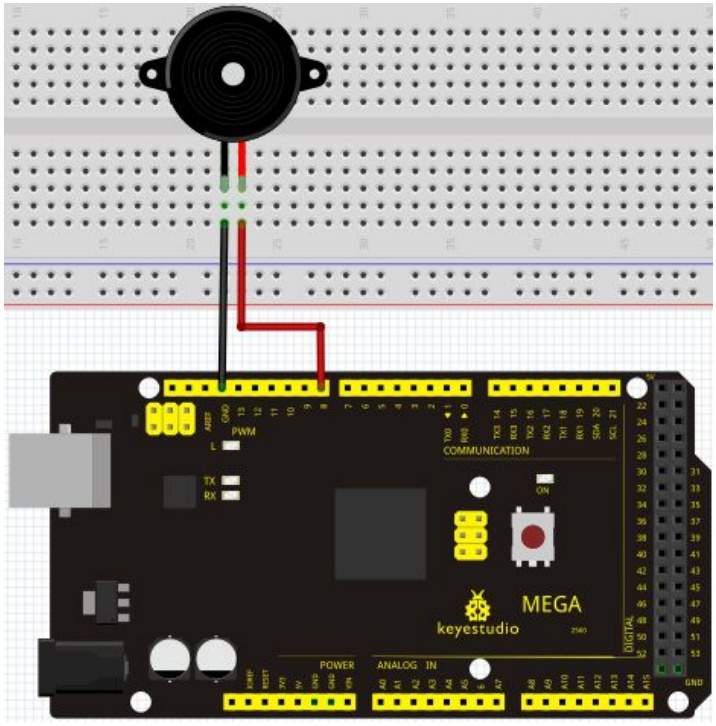
Passive buzzer*1
Key *1
Breadboard*1
Breadboard jumper wires

Circuit connection

Connection for R3:



Connection for 2560 R3:



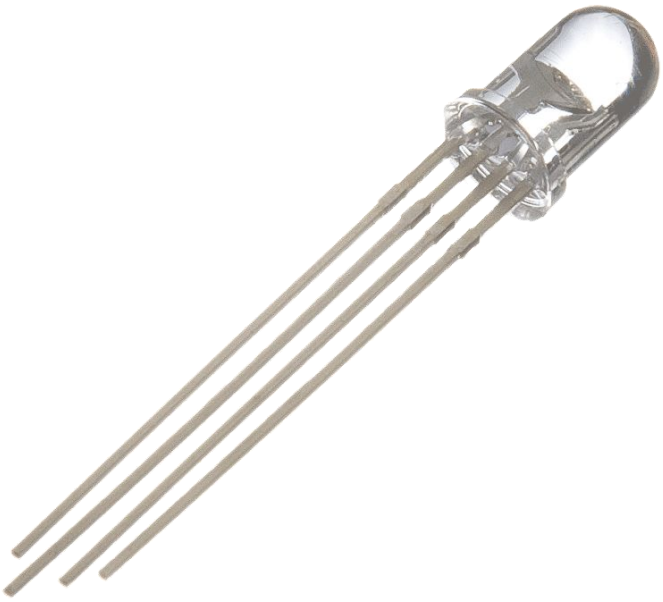
Sample program

```
//////////////////////////////////////////////////////////
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
}
}
}
//////////////////////////////////////////////////////////
```

After downloading the program, buzzer experiment is finished.

****

Project 10: RGB LED

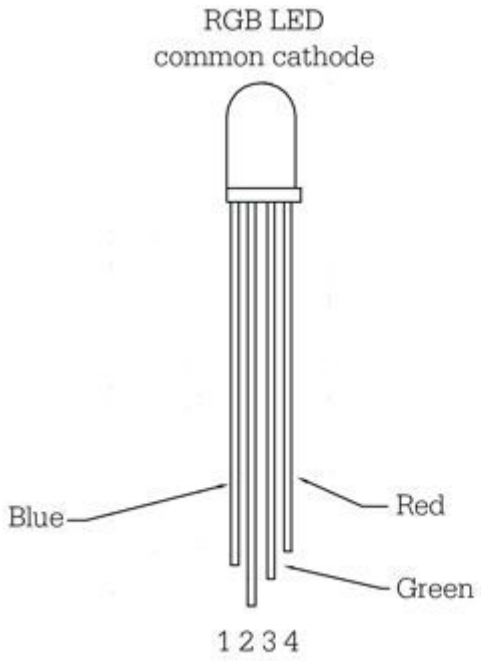


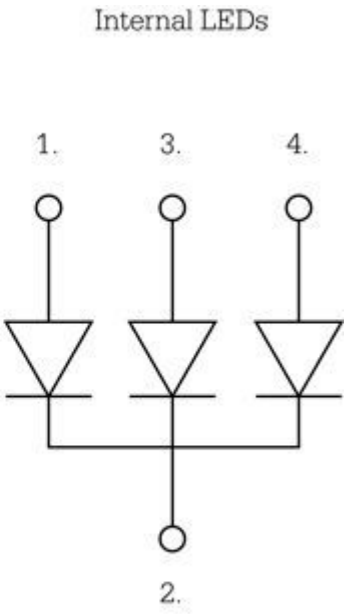
Introduction

Tricolor principle to display various colors

PWM controlling ports to display full color

Can be driven directly by Arduino PWM interfaces



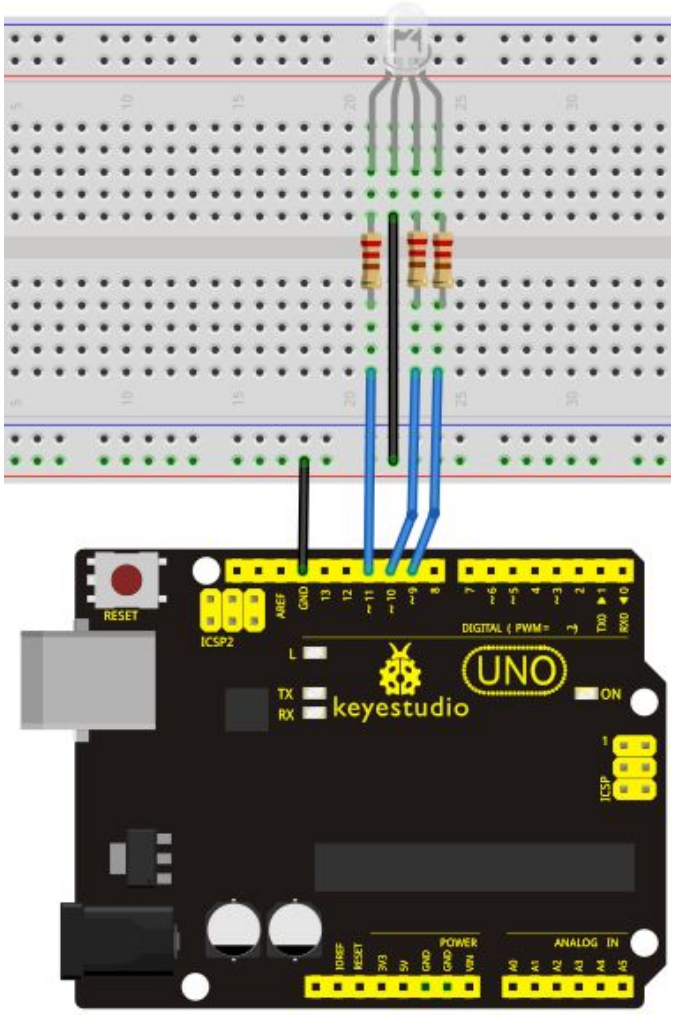


Hardware Required

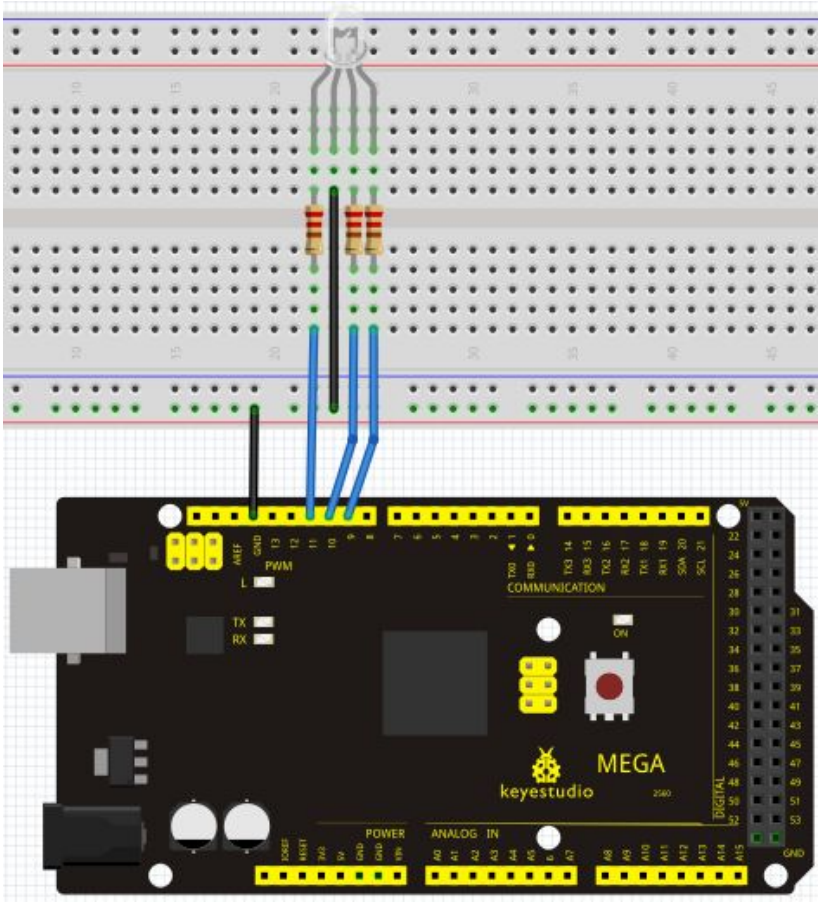
Arduino controller * 1
USB cable * 1
RGB LED * 1

Circuit connection

Connection for R3:



Connection for 2560 R3:



Sample program

```
//////////////////////////////////////////////////////////////
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 11: 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

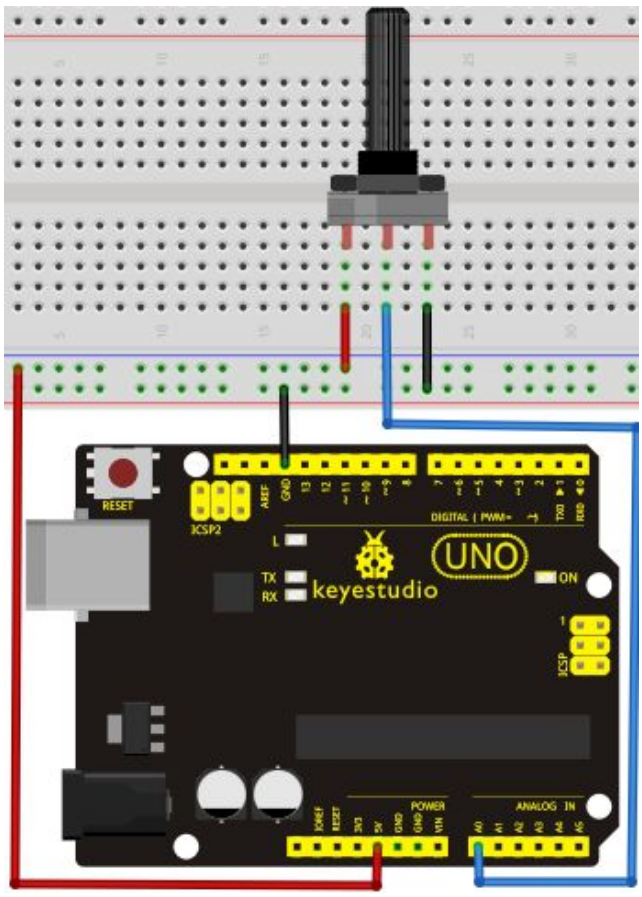
Potentiometer *1
Breadboard*1
Breadboard jumper wires * several

Circuit connection

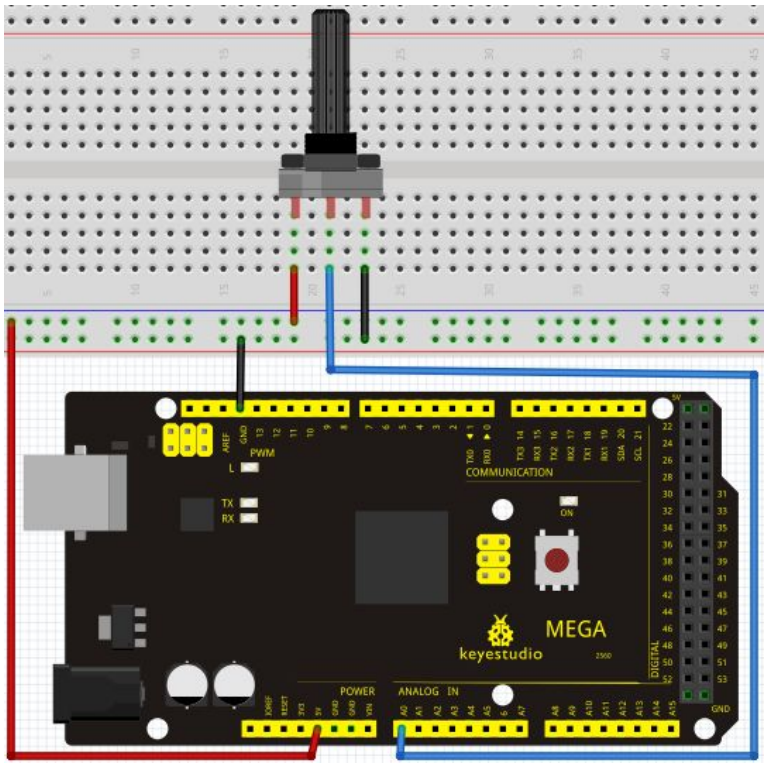
In this experiment, we will convert the resistance value of the potentiometer to analog ones and display it on the screen. This is an application we need to master well for our future experiments.

Connection circuit as below:

Connection for R3:



Connection for 2560 R3:



We use the analog interface 0.
The analog interface we use here is interface 0.

Sample program

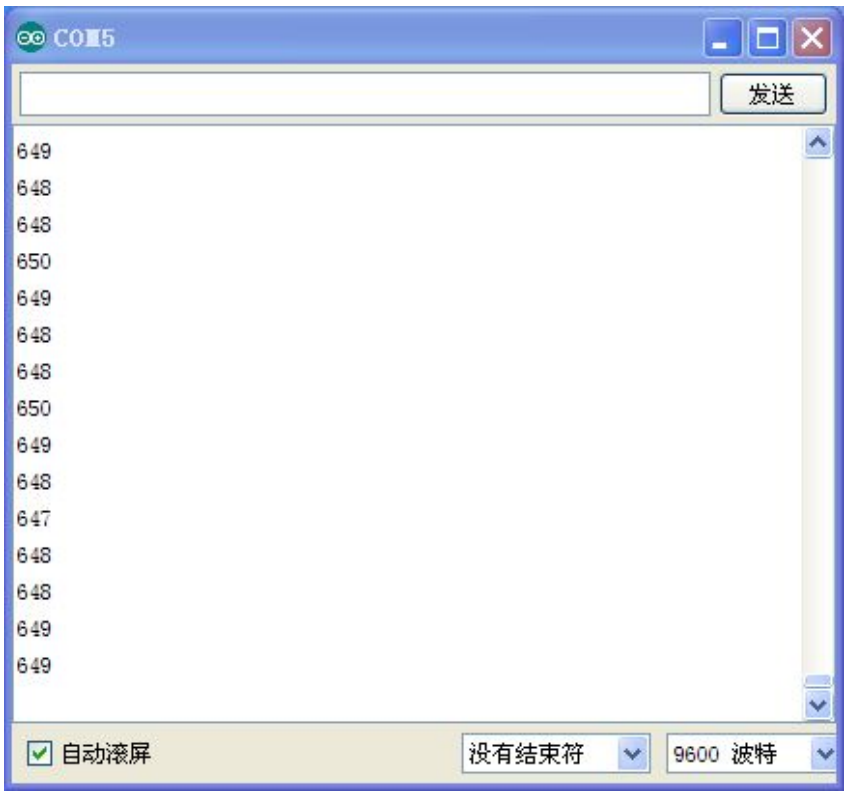
The program compiling is simple. An analogRead () Statement can read the value of the interface. The A/D acquisition of Arduino 328 is in 10 bits, so the value it reads is among 0 to 1023. One difficulty in this project is to display the value on the screen, which is actually easy to learn. First, we 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
}
//////////////////////////////////////////////////////////
```

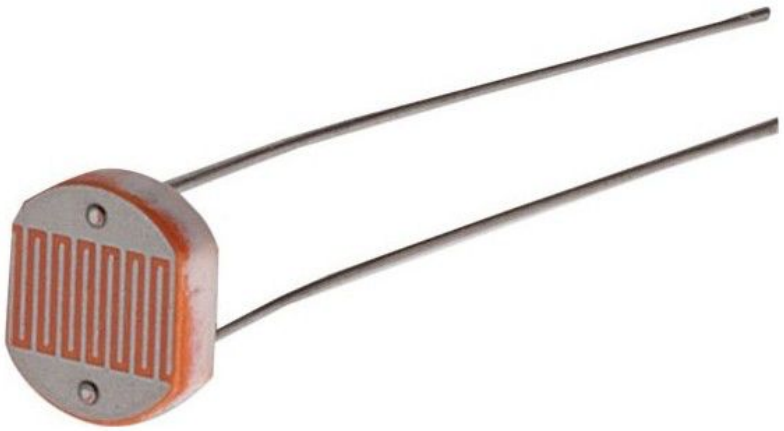
Result

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.



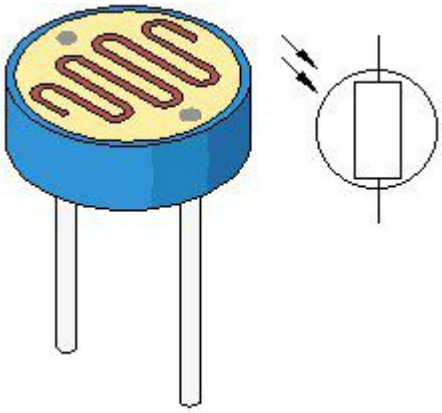
When you rotate the potentiometer knob, you can see the displayed value changes. The reading of analog value is a very common function since most sensors output analog value. After calculation, we can have the corresponding value we need.
The experiment is now completed, thank you.
****

Project 12: 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.



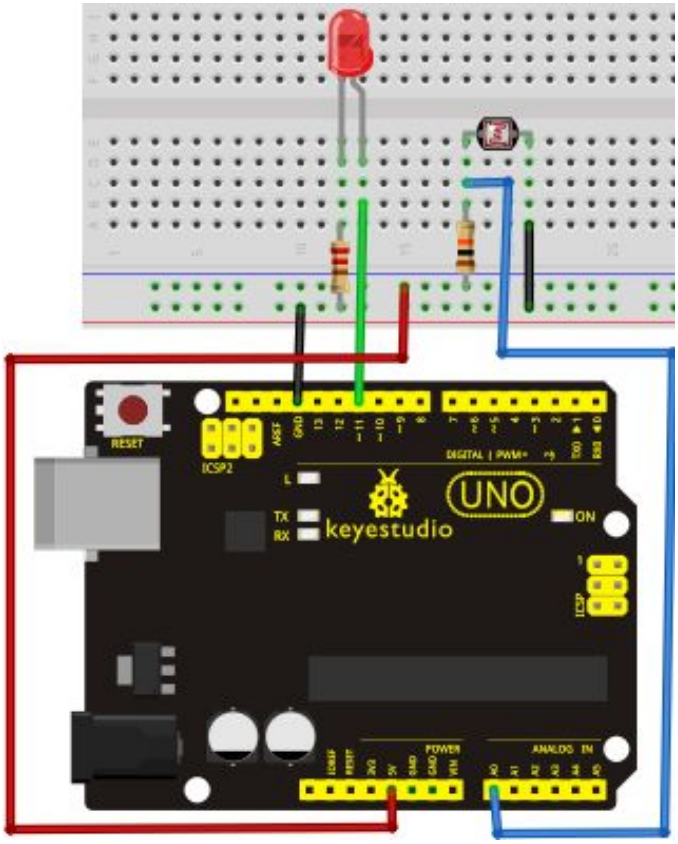
We will start with a relatively simple experiment regarding photovaristor application. Photovaristor is an element that changes its resistance as light strenth changes. So we will need to read the analog values. We can refer to the PWM experiment, replacing the potentiometer with photovaristor. When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

Hardware required

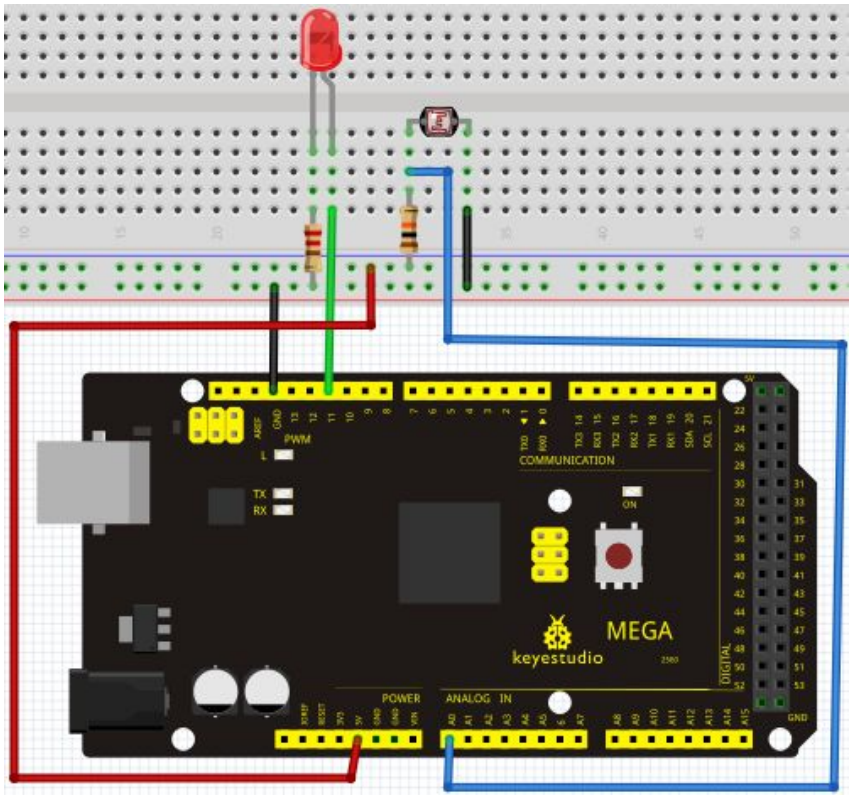
Photo resistor*1
Red M5 LED*1
10KΩresistor*1
220Ωresistor*1
Bread board*1
Bread board jumper wires

Circuit connection

Connection for R3:



Connection for 2560 R3:



Sample program

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 13: Flame sensor

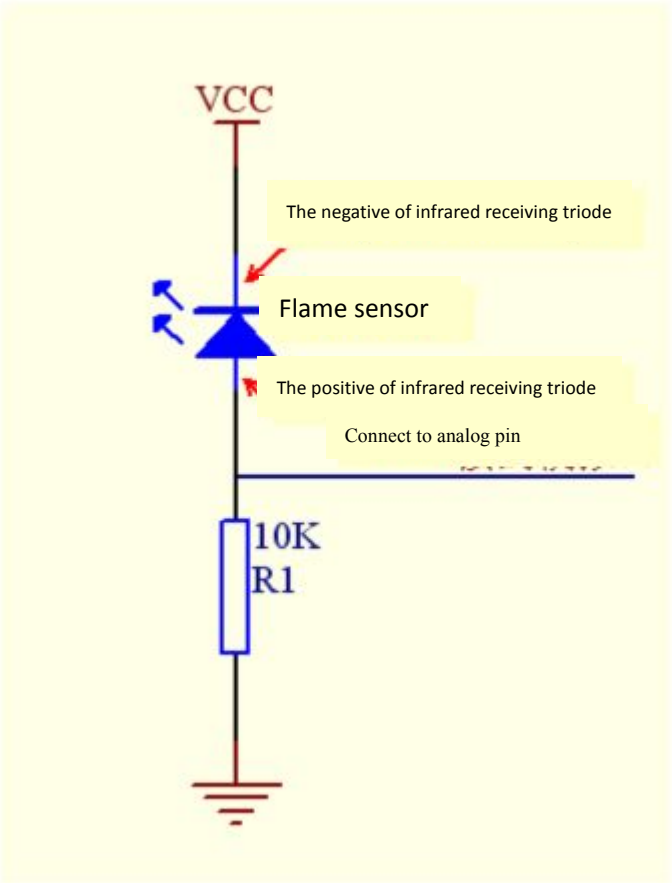
Introduction

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:



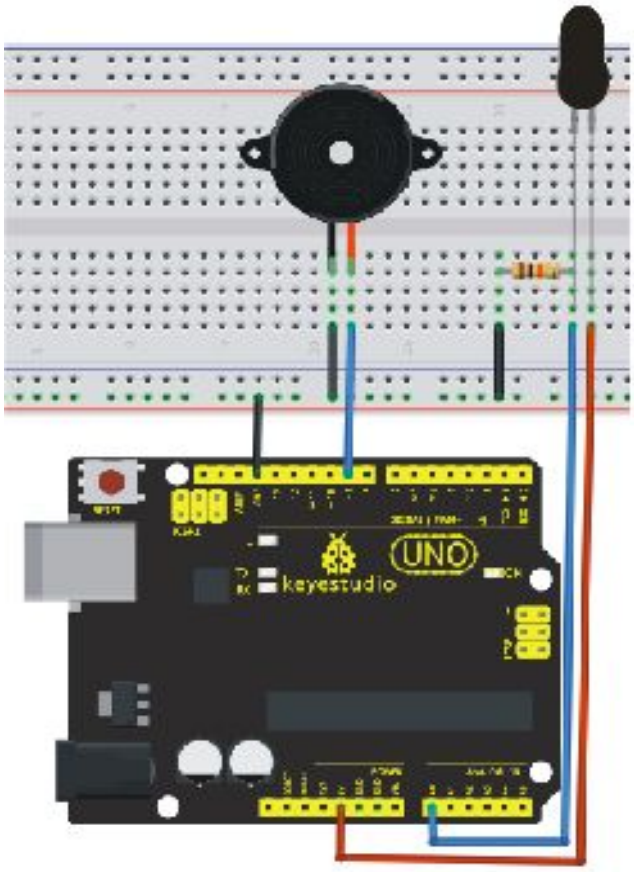
Hardware required

Flame sensor *1
Buzzer *1
10K resistor *1
Breadboard jumper wires

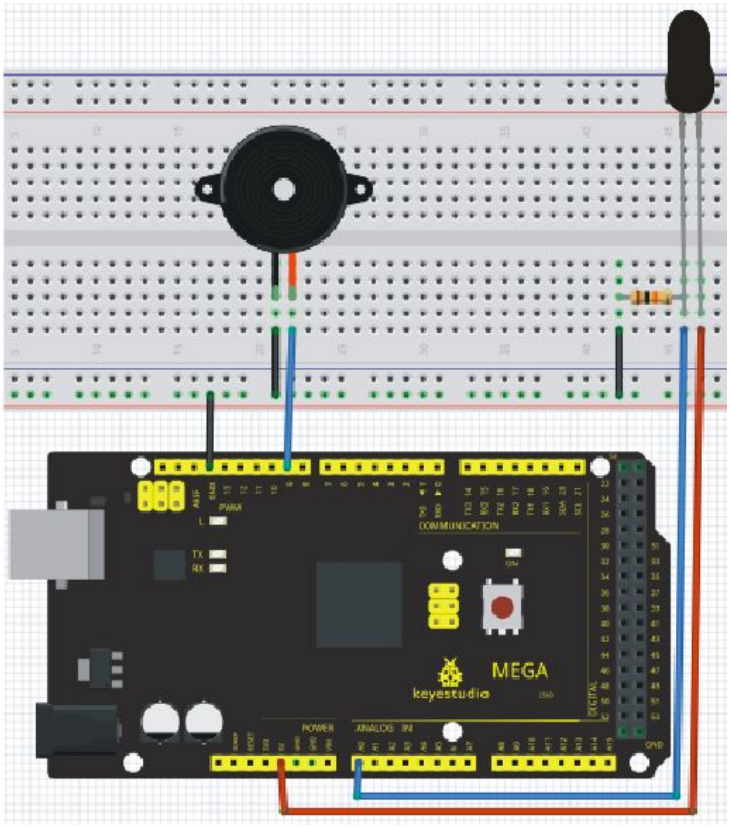
Experiment connection

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.
Connection for R3:



Connection for 2560 R3:



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 program

```
//////////////////////////////////////////////////////////
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 14: Analog temperature (thermistor)

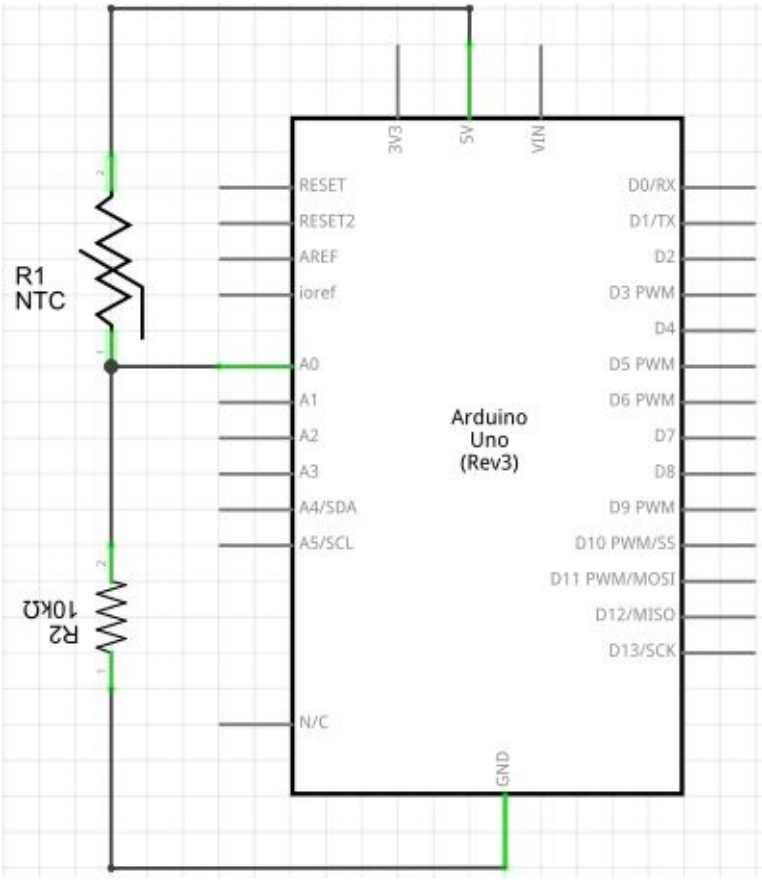
Introduction

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

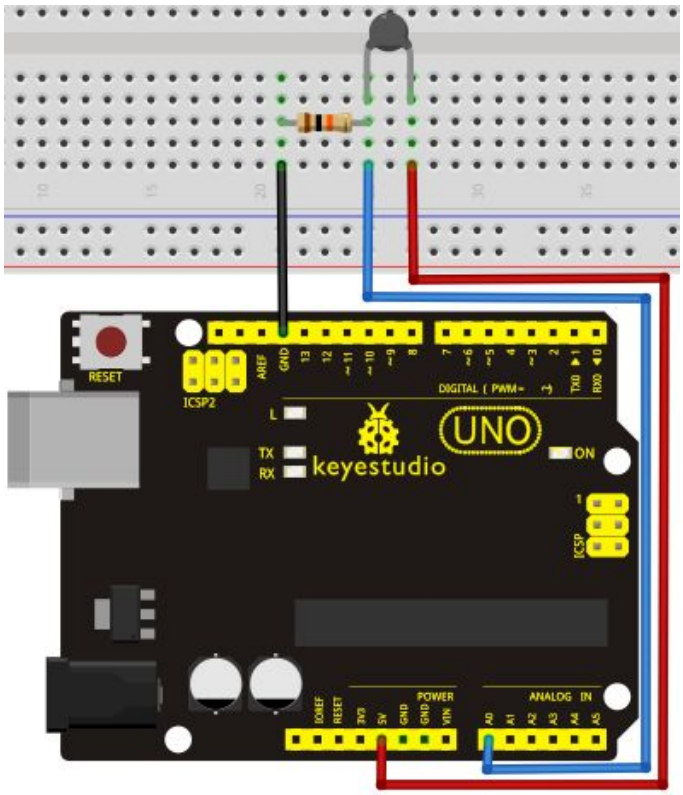
Thermistor *1
10KΩ resistor *1
Breadboard *1
Breadboard jumper wires * several

Schematic diagram

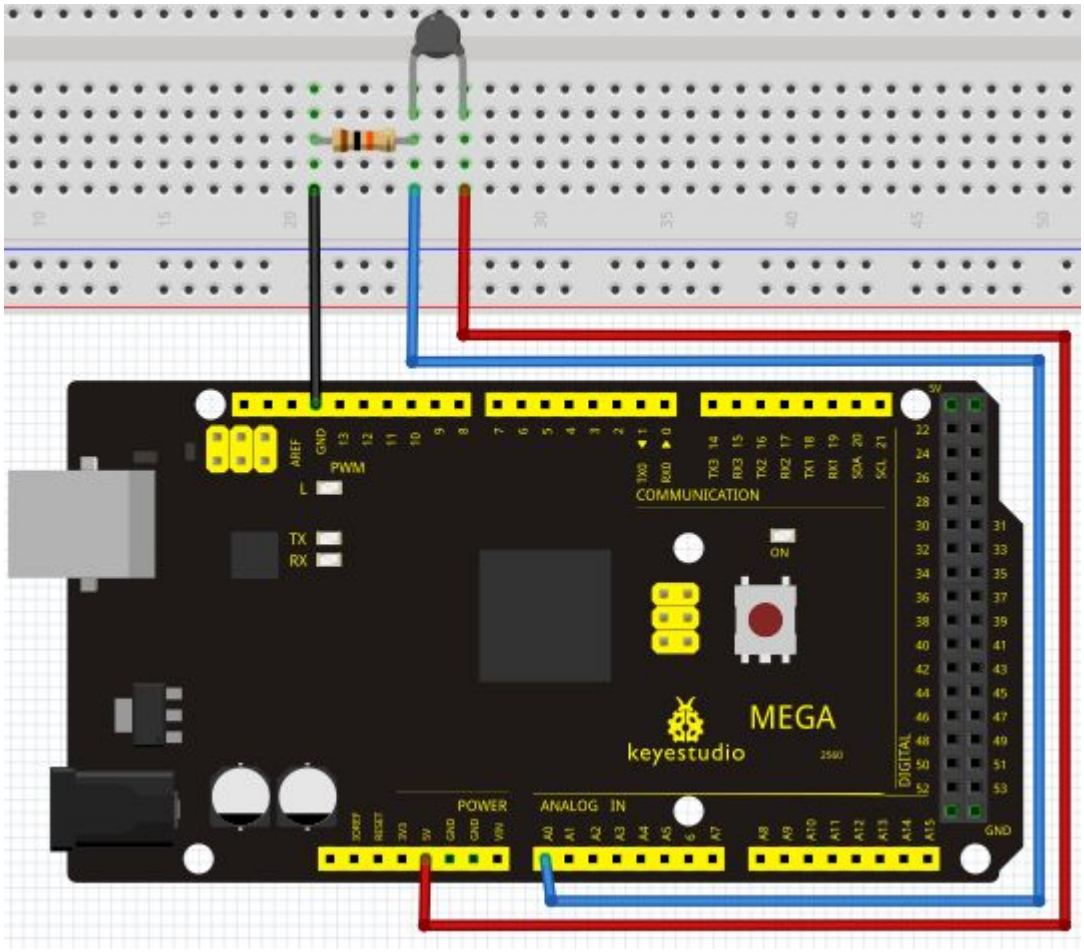


Circuit connection

Connection for R3:



Connection for 2560 R3:



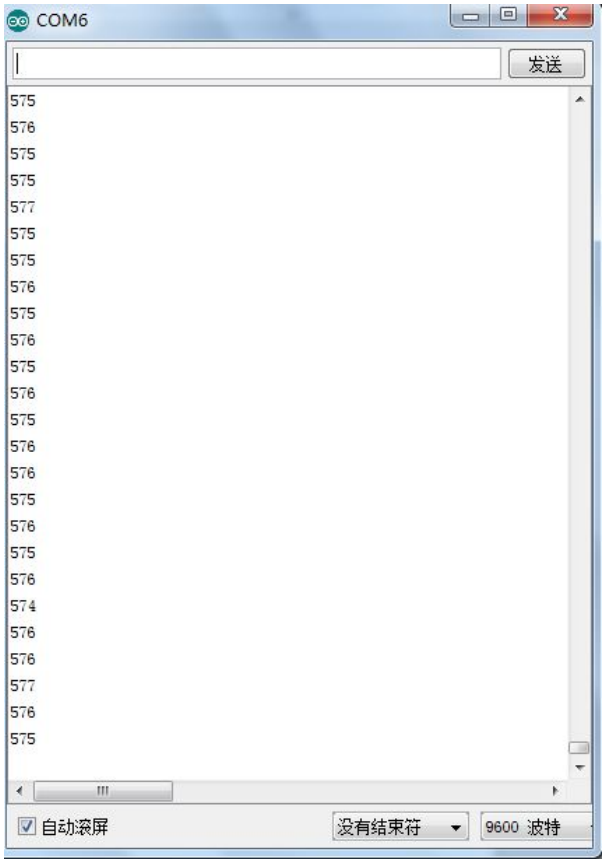
Sample program

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

void setup()

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
```

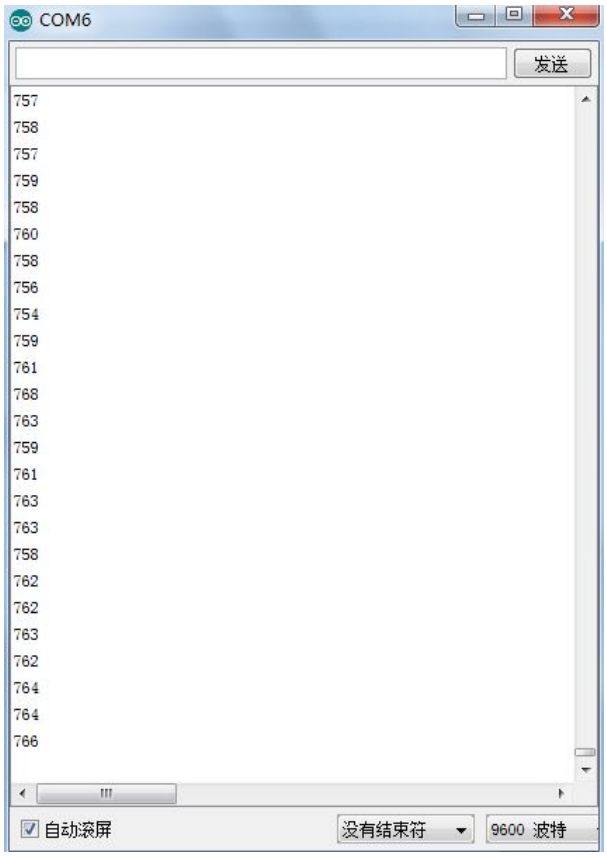
```
Serial.println(val,DEC);//Print the value to serial port
delay(100);
}
////////////////////////////////////////////////////////////
```

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.



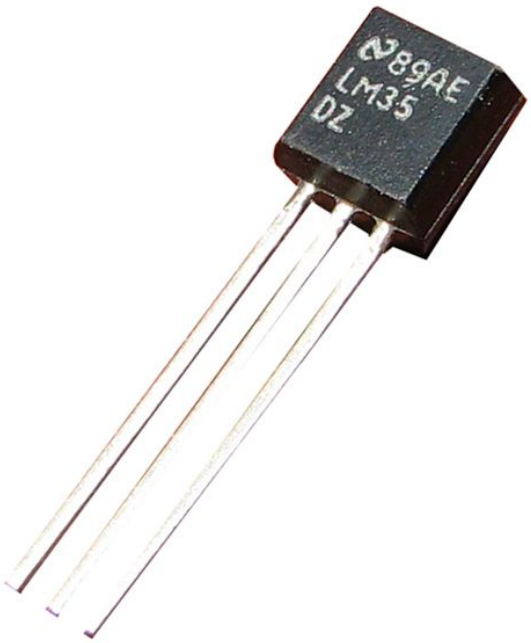
Pic 1



Pic 2

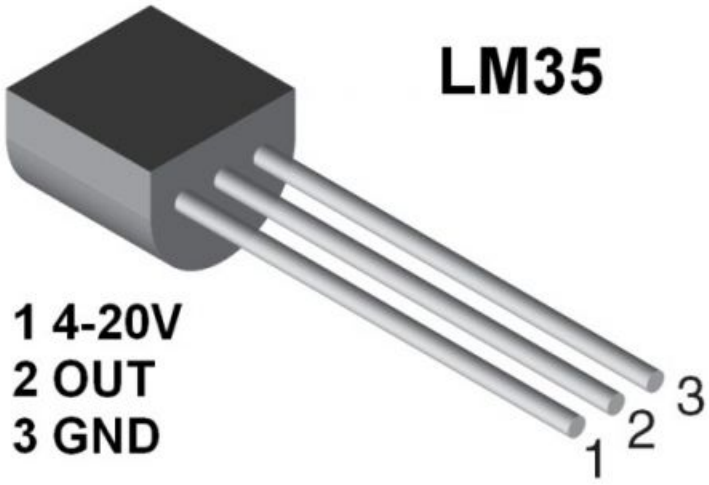
****

Project 15: LM35 temperature sensor



Introduction

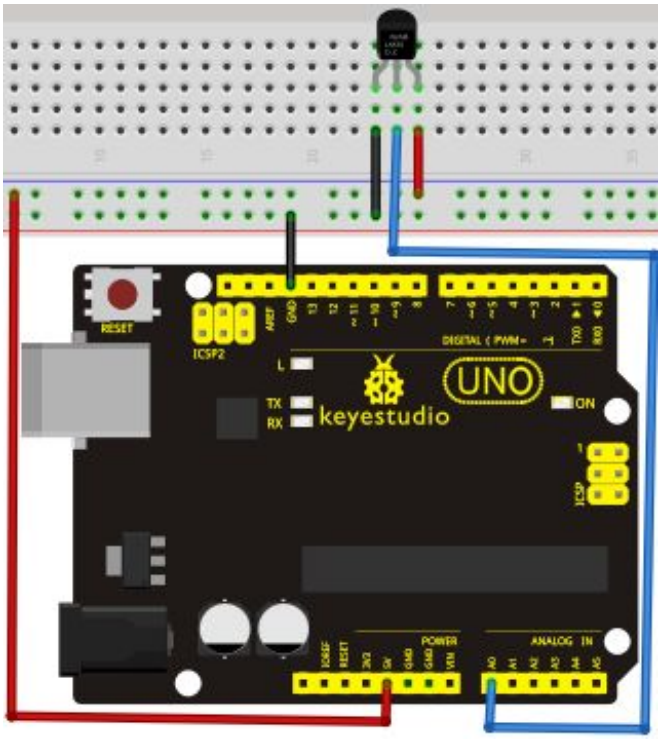
LM35 is a common and easy-to-use temperature sensor. It does not require other hardware. You just need an analog port to make it work. The difficulty lies in compiling the code to convert the analog value it reads to celsius temperature.



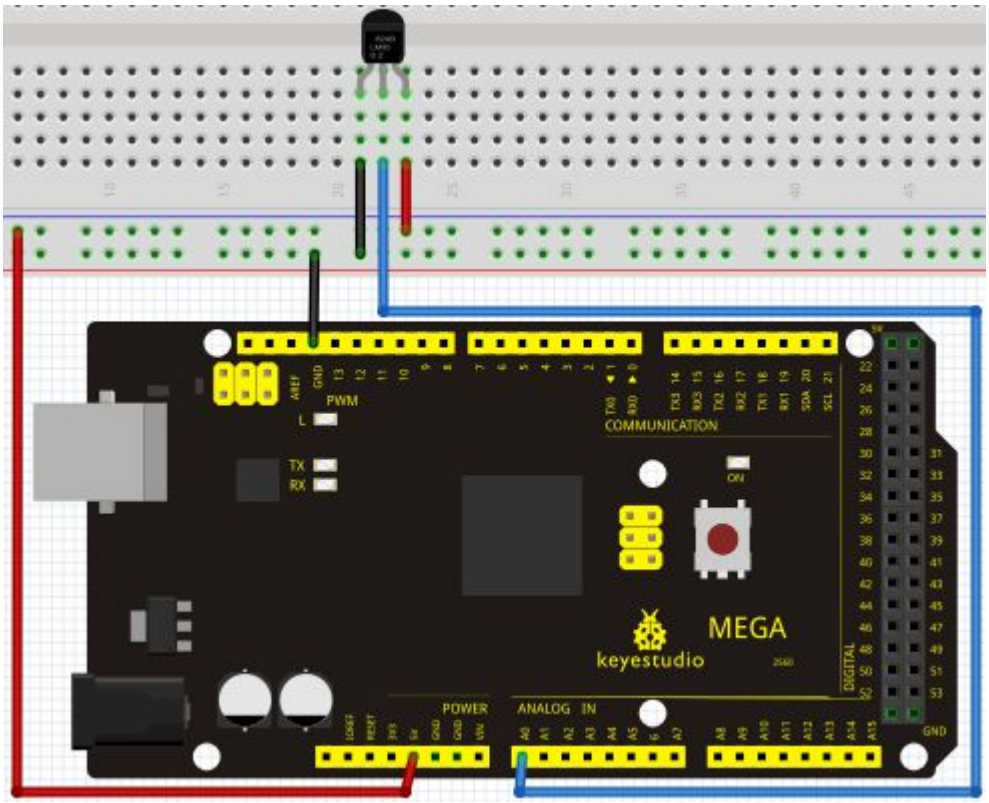
Hardware required

LM35*1
Breadboard*1
Breadboard jumper wires

Connection for R3:



Connection for 2560 R3:

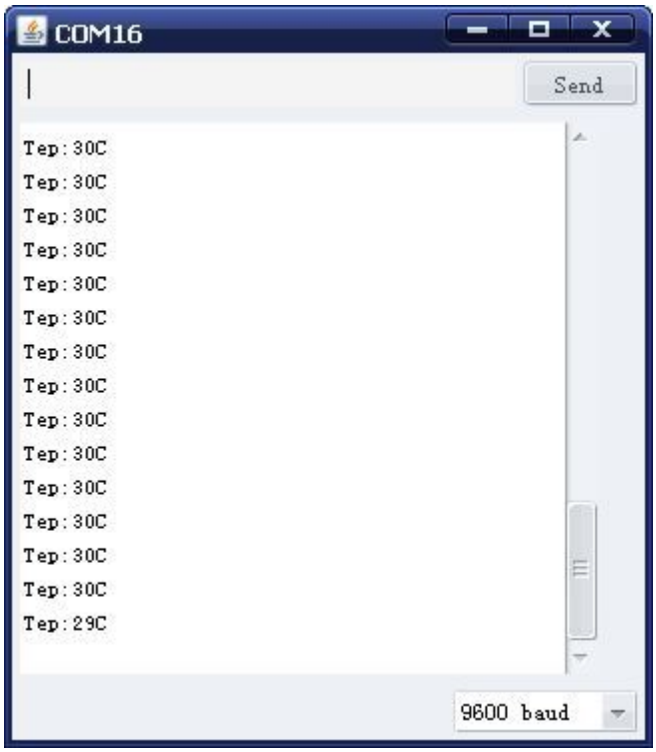


Sample program

```
//////////////////////////////////////////////////////////
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 16: Temperature-controlled cup

Introduction

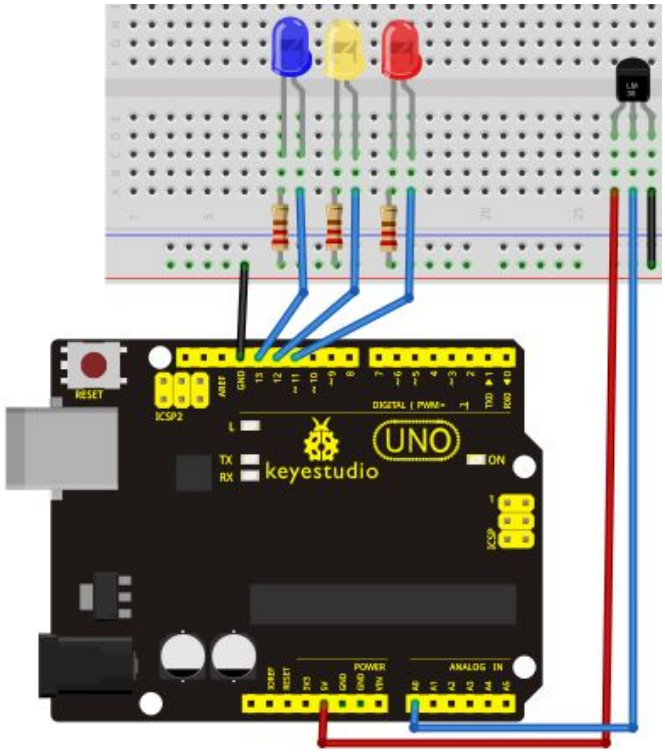
Today, we will use Arduino to make a temperature-controlled cup. First, let's design the circuit. When the LM35 temperature sensor senses different temperature, different LED will be turned on representing the temperature.

Hardware required

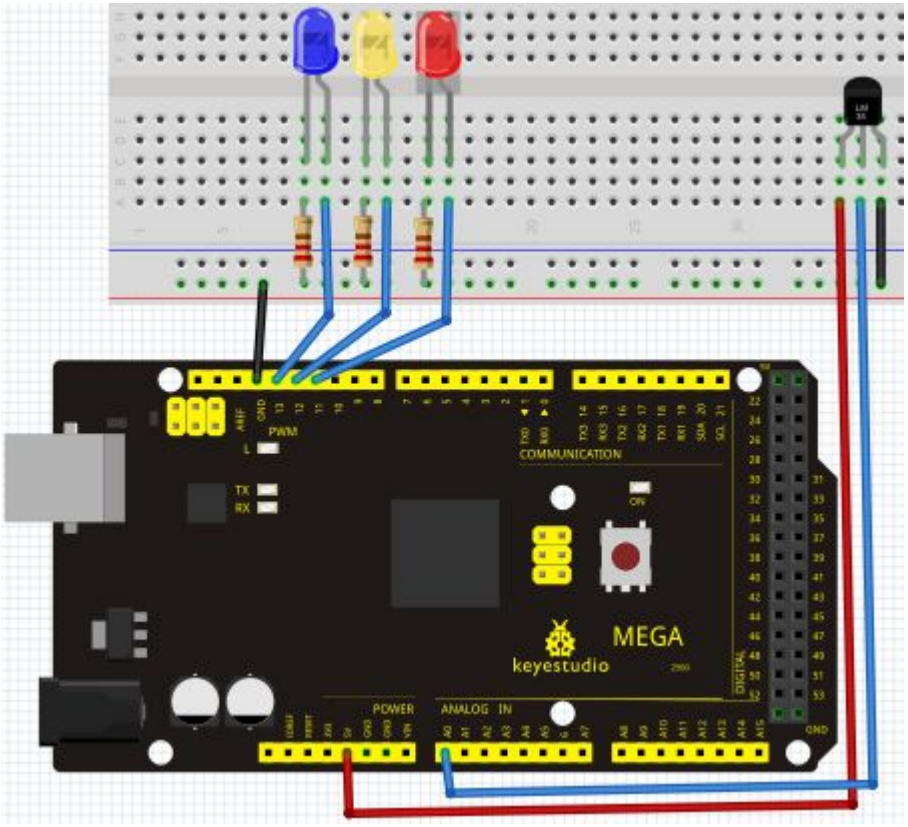
Arduino controller *1
Breadboard *1
Breadboard jumper wires * several
Red, yellow & blue LED *3 (one for each color)
220Ω resistor*3
LM35 temperature sensor *1
USB cable *1

Circuit connection

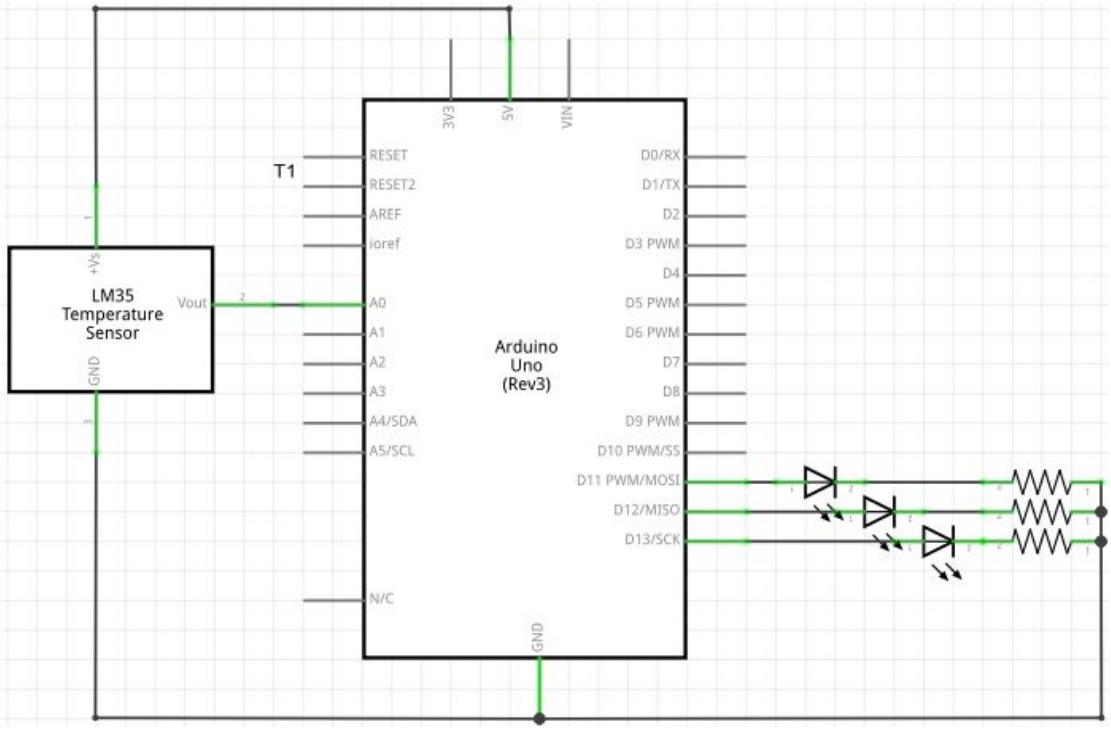
Connection for R3:



Connection for 2560 R3:



Schematic diagram:



Sample program

```
//////////////////////////////////////////////////////////
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.

****

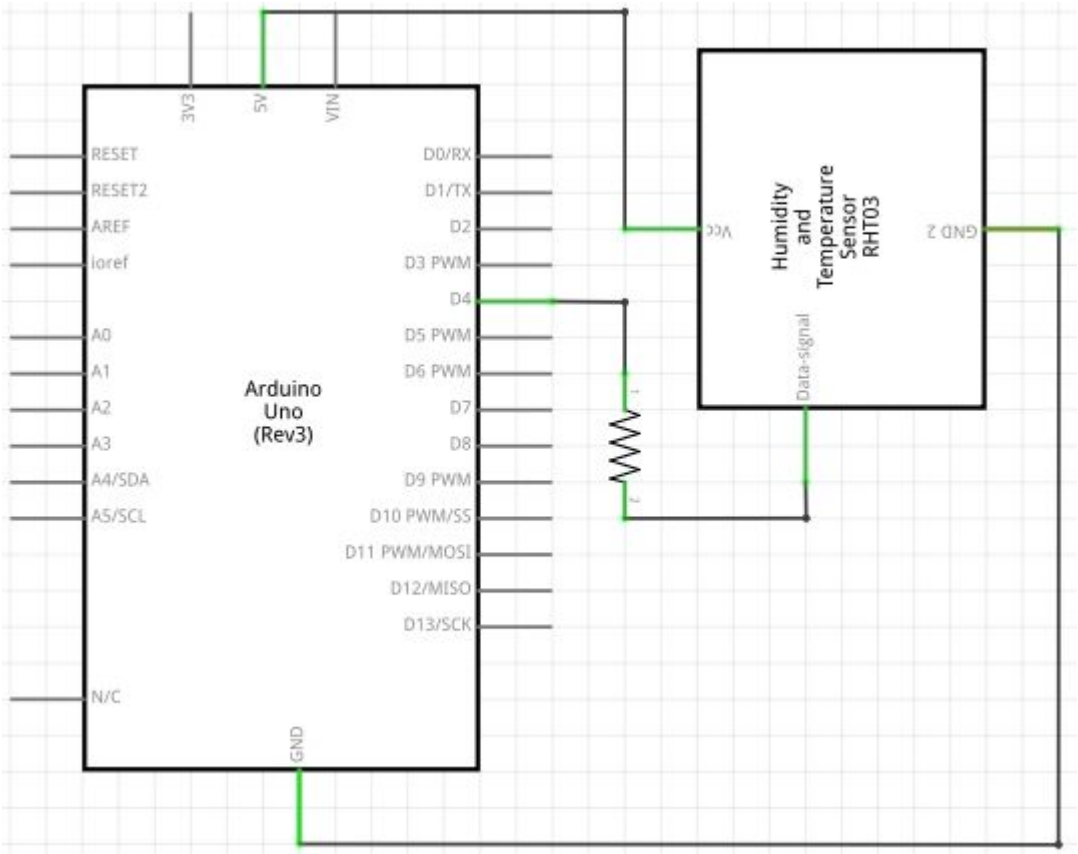
Project 17: DHT11 Temperature and Humidity Sensor

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.

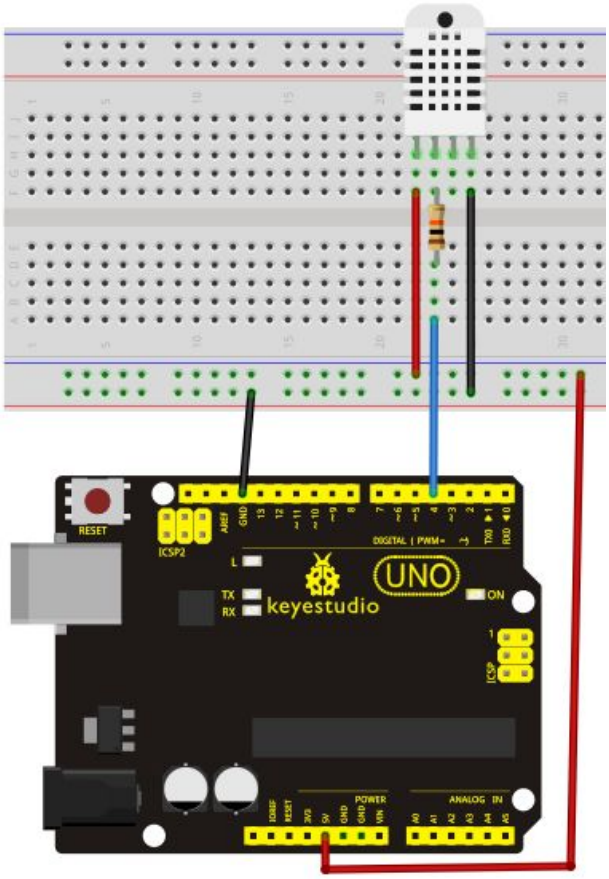
Hardware required

Temperature and humidity unit *1
10K resistor *1
Breadboard *1
Breadboard jumper wires *several
Schematic diagram:

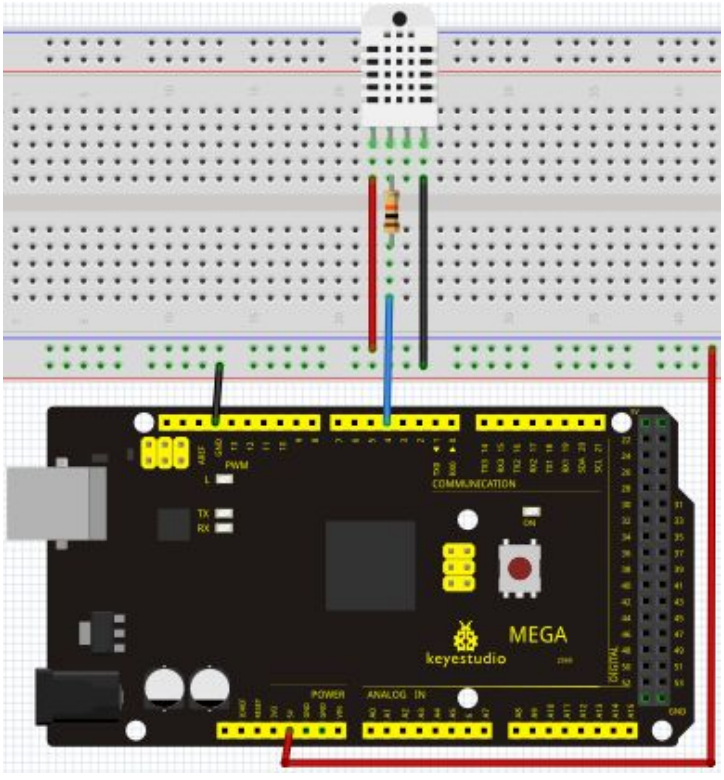


Circuit connection

Connection for R3:



Connection for 2560 R3:



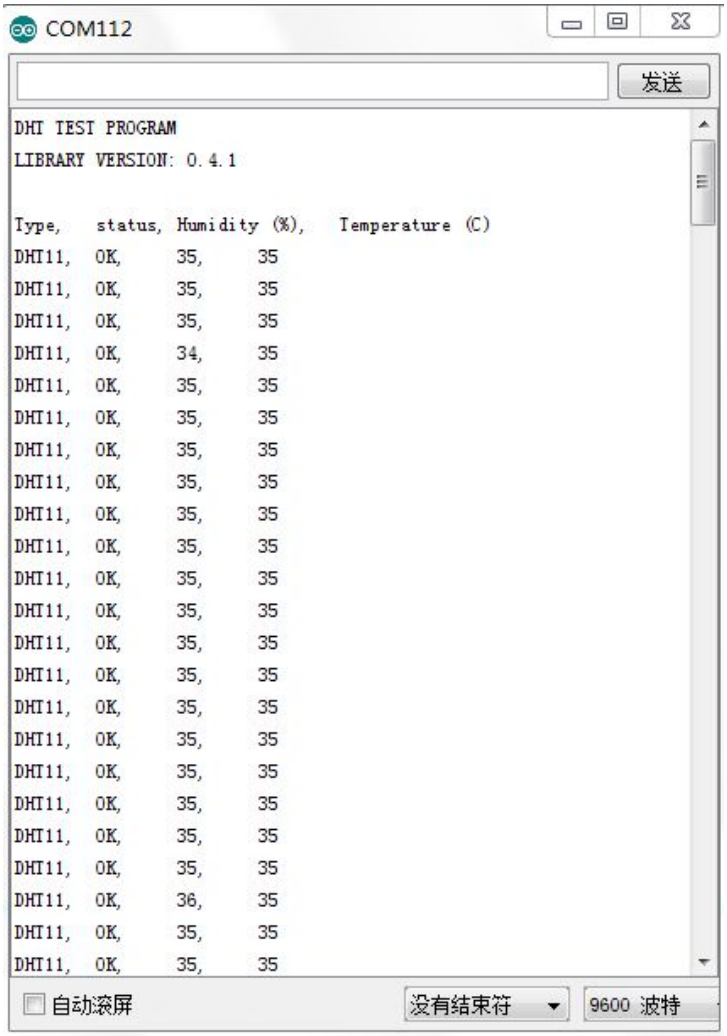
Sample Program

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, 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 18: Tilt switch

Introduction

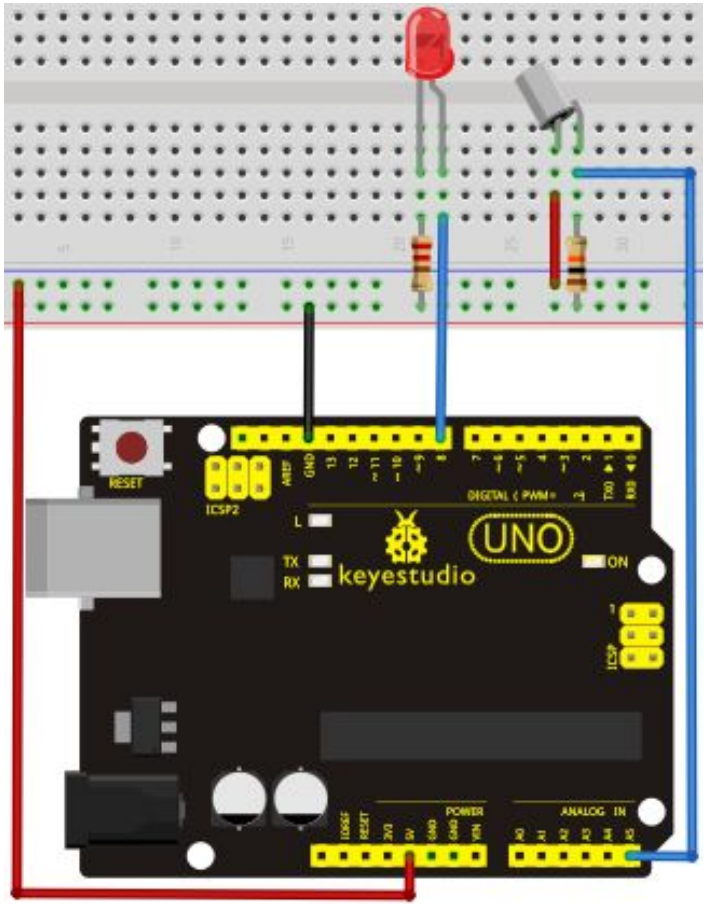
Tilt switch controls the ON and OFF of an LED.

Hardware required

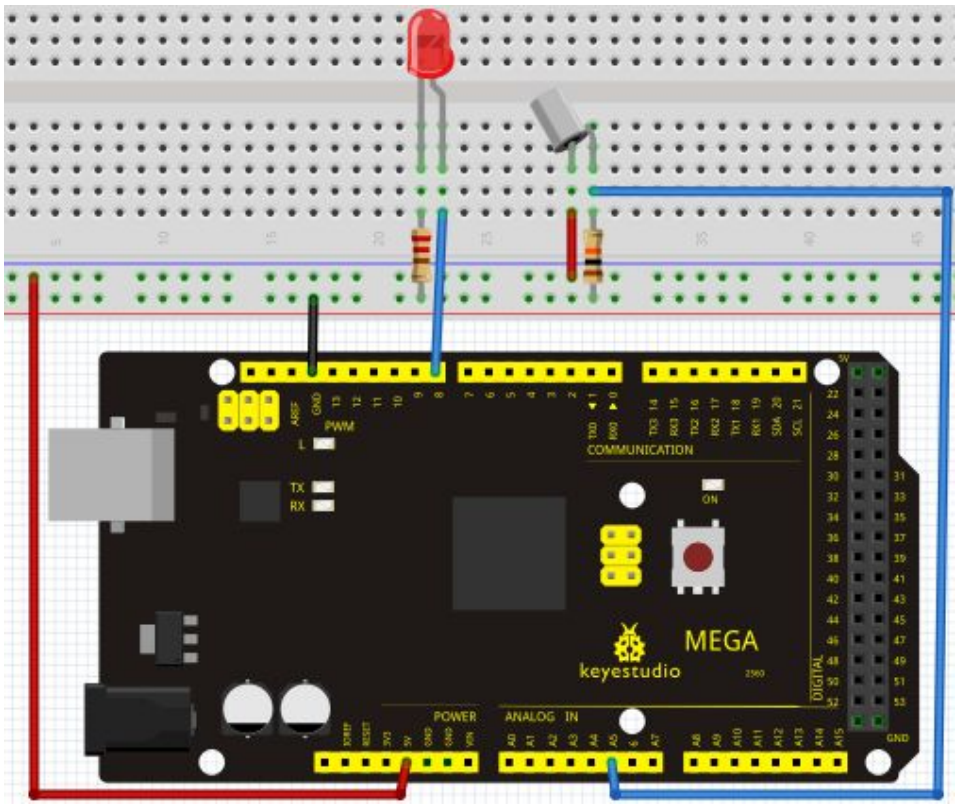
Ball switch*1
Led *1
220Ω resistor*1
Breadboard jumper wires

Circuit connection

Connection for R3:



Connection for 2560 R3:



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 program

```
//////////////////////////////////////////////////////////
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 19: Magical Light Cup

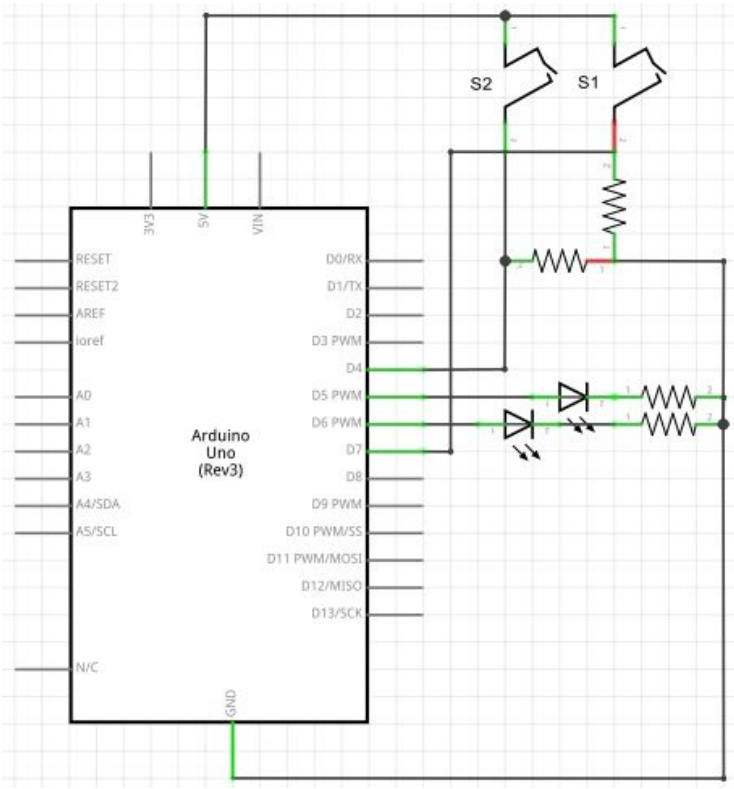
Introduction

Magical light cup module is a product developed by KEYES that can interact with ARDUINO.
The principle is to use PWM to regulate light brightness of the two modules.
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

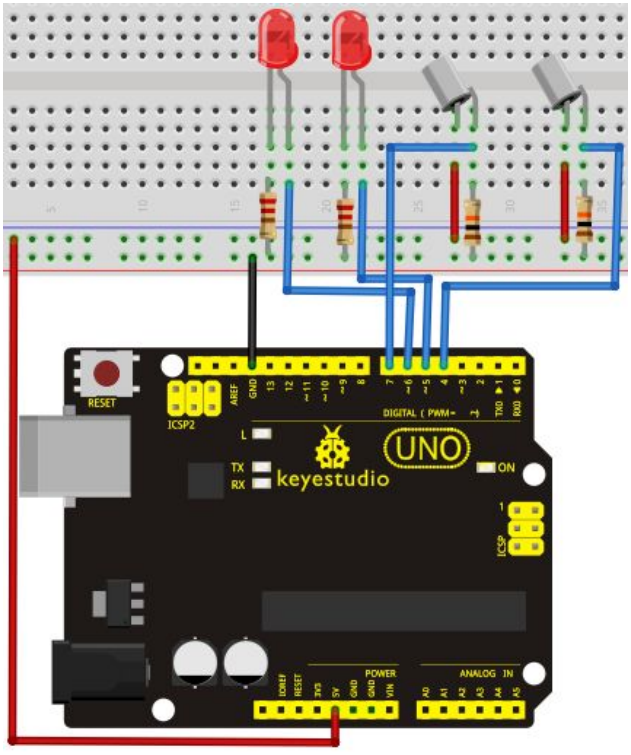
Ball tilt switch *2
Led *2
220Ω resistor *2
10KΩ resistor *2
Breadboard *1
Breadboard jumper wires *several

Schematic diagram

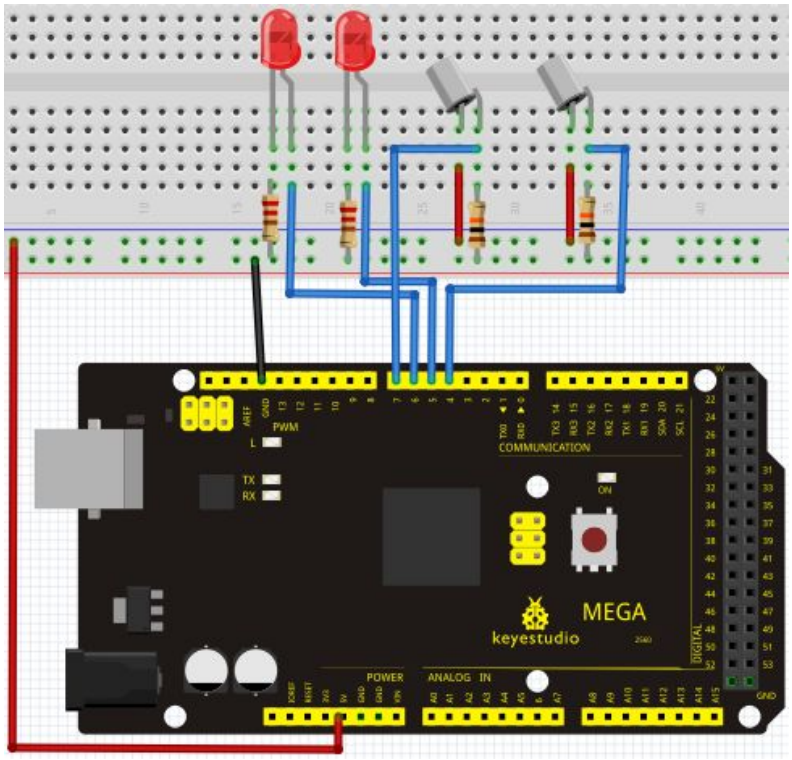


Circuit connection

Connection for R3:



Connection for 2560 R3:



Sample program

```
//////////////////////////////////////////////////////////
int LedPinA = 5;
int LedPinB = 6;
int ButtonPinA = 7;
int ButtonPinB = 4;

void setup()
{
   pinMode(LedPinA, OUTPUT);
   pinMode(LedPinB, OUTPUT);
   pinMode(ButtonPinA, INPUT);
   pinMode(ButtonPinB, INPUT);
}
void loop()
{

    if(digitalRead(ButtonPinA)==HIGH) //Read sensor value
       {
           digitalWrite(LedPinA, HIGH);      // Turn on LED when the sensor is tilted
       }
```

```
    else
      {
        digitalWrite(LedPinA, LOW);       // Turn off LED when the sensor is not triggered
      }
        if(digitalRead(ButtonPinB)==LOW) //Read sensor value
      {
        digitalWrite(LedPinB, HIGH);      // Turn on LED when the sensor is tilted
      }
    else
      {
        digitalWrite(LedPinB, LOW);       // Turn off LED when the sensor is not triggered
      }
}

//////////////////////////////////////////////////////
```

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 20: Vibration switch

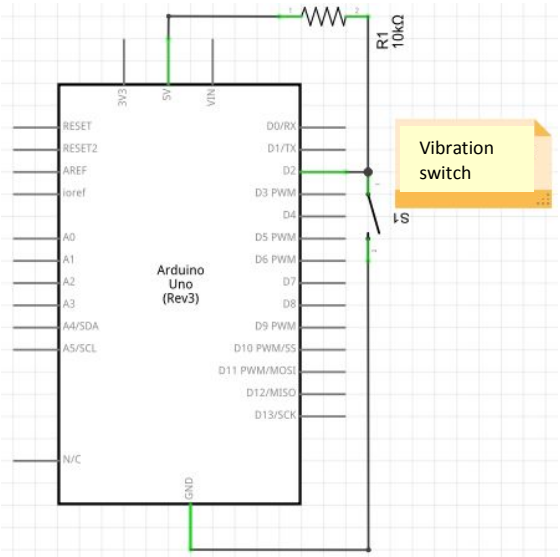
Introduction

Vibration switch, the correct name should be vibration sensor. It is a electronic switch sensing the intensity of a vibration and transfer the result to the circuit device, and activate the circuit to start working.

Hardware required

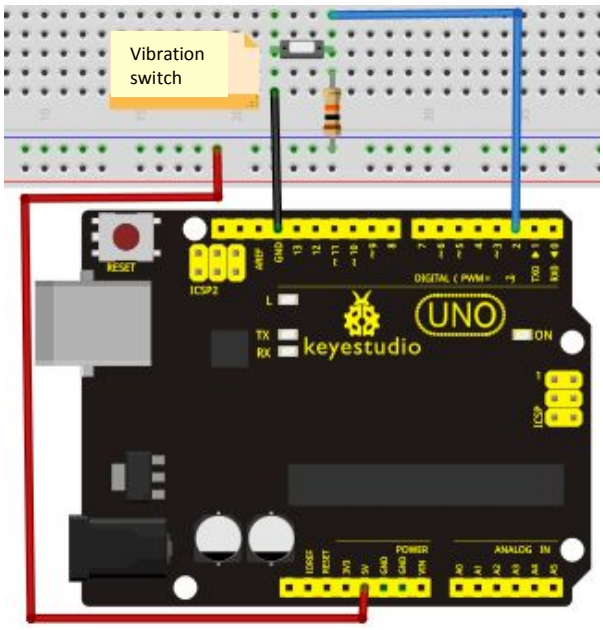
Arduino board *1
USB cable *1
Vibration Sensor*1
10KΩ resistor *3
Breadboard*1
Breadboard jumper wires* several

Schematic diagram

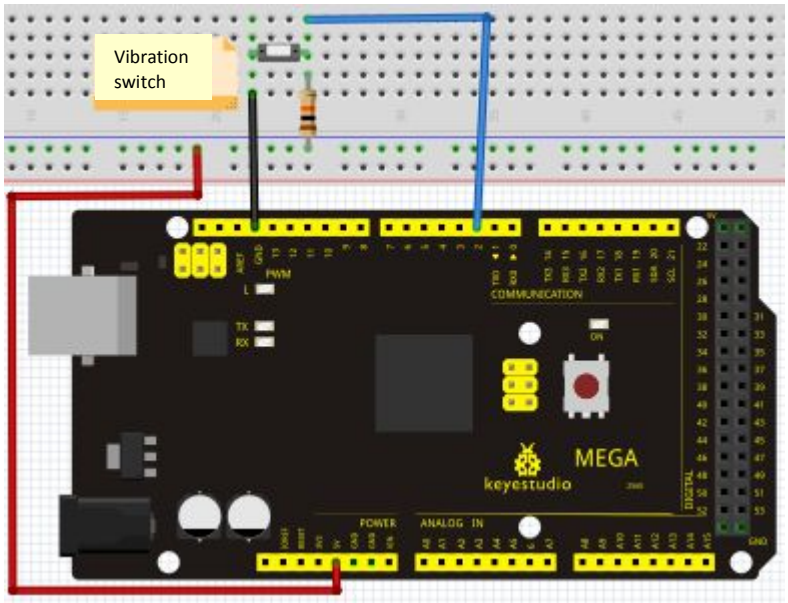


Circuit connection

Connection for R3:



Connection for 2560 R3:



Sample program

```
//////////////////////////////////////////////////////////
#define SensorLED         13
#define SensorINPUT       2
unsigned char state = 0;
void setup()
{
   pinMode(SensorLED, OUTPUT);
   pinMode(SensorINPUT, INPUT);
attachInterrupt(0, blink, FALLING);//D2 as external interruption 0, when there is falling trigger and call blink function

}
void loop()
{
        if(state!=0)
        {
           digitalWrite(SensorLED,HIGH);
           delay(3000);
           state = 0;
        }
        else
           digitalWrite(SensorLED,LOW);
}

void blink()// digital input of the sensor falling, triggering interruption function
```

```
{
   state++;
}
//////////////////////////////////////////////////////////
```

Result

Touch the sensor with your hand, the D13 indicator light on Arduino will be on for 3 seconds and then be out.

****

Project 21: Sound-control light

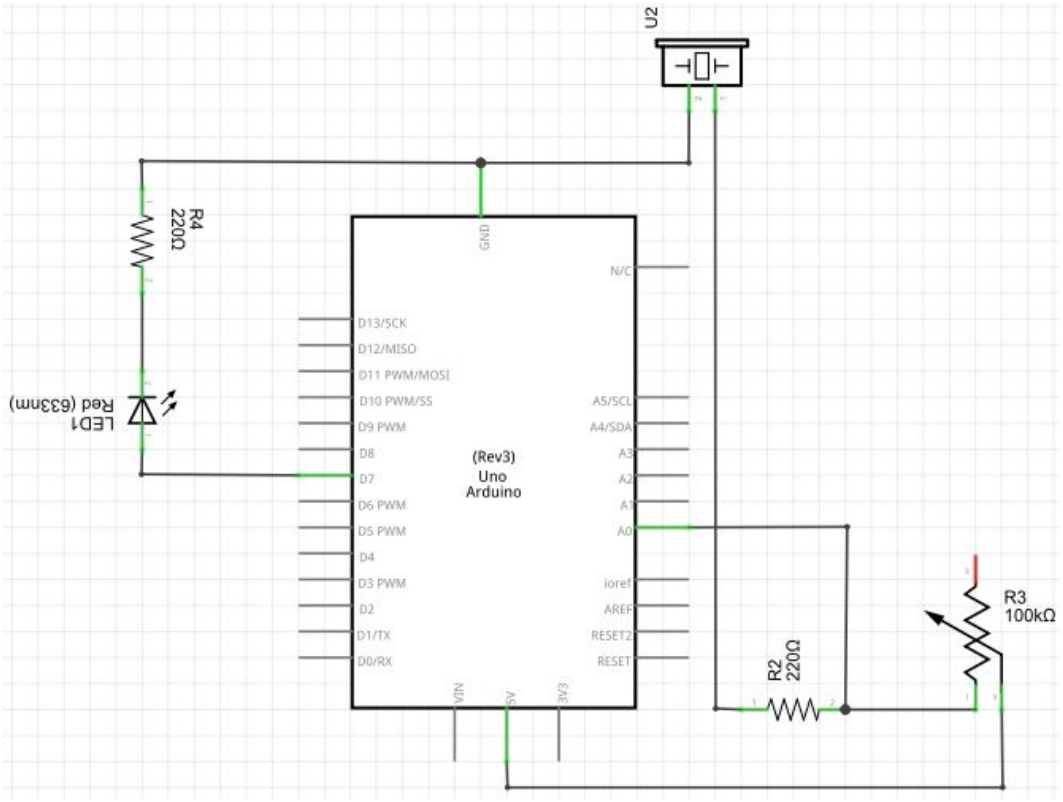
Introduction

In this experiment, we use sound passing through MIC to control the on and off of the light.

Hardware required

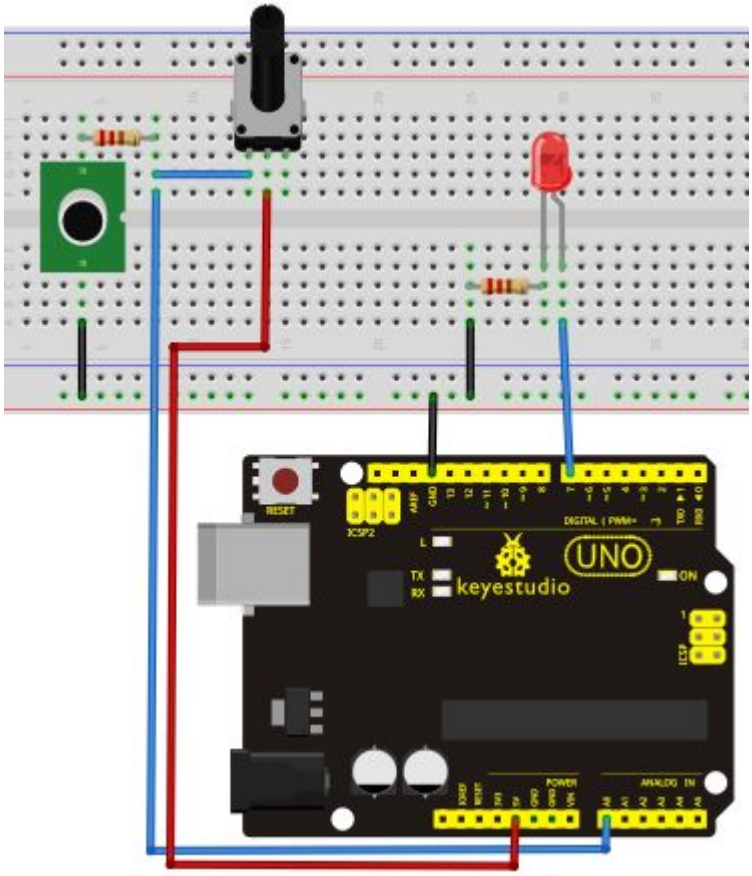
Potentiometer module *1
Red M5 LED *1
220Ω resistor *2
Breadboard *1
Breadboard jumper wires *several
MIC *1

Schematic diagram

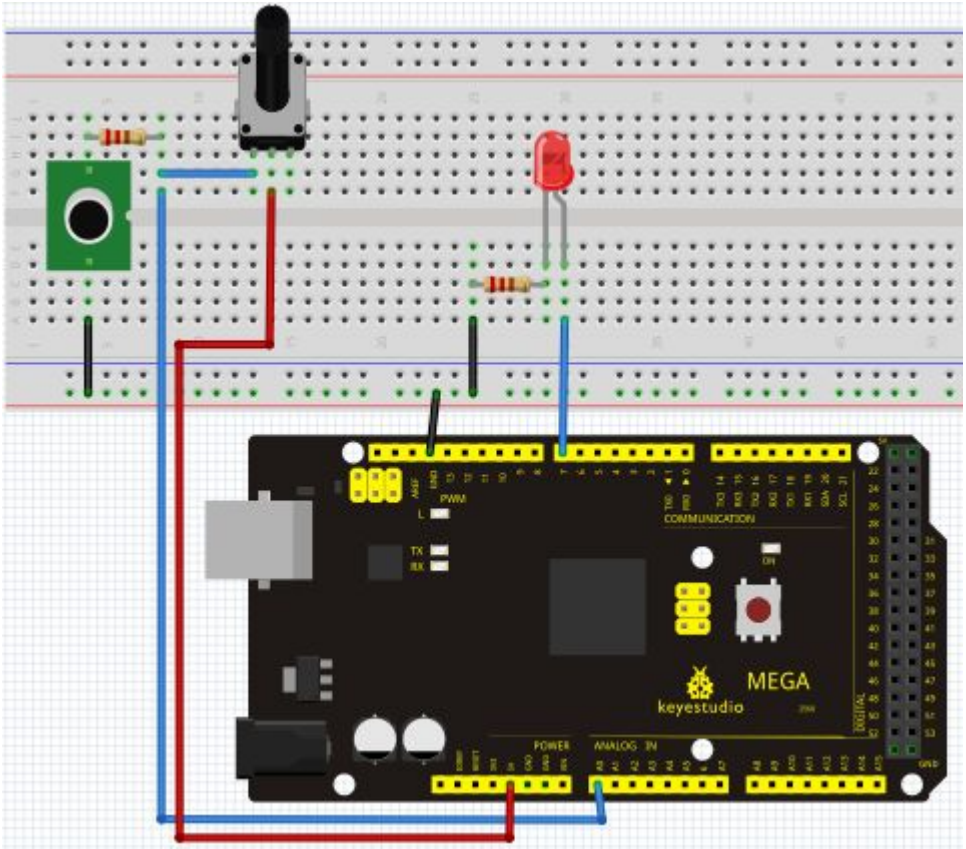


Circuit connection

Connection for R3:



Connection for 2560 R3:



The circuit connection for this experiment is relatively simple. It has no processing of the signal from the MIC, so signal is weak and insensitive. Instead of sound signal, we blow air to the MIC.

Sample program

```
//////////////////////////////////////////////////////////
int LEDpin = 7;                         // set pin for LED
void setup() {
   Serial.begin(9600);
   pinMode(LEDpin,OUTPUT);
}
void loop() {

   int Soundvalue = analogRead(A0);    // read the input analog value
   Serial.println(Soundvalue);
   if(Soundvalue>300)
   {
    digitalWrite(LEDpin,HIGH);          // when the analog value is bigger than the set value, turn
```

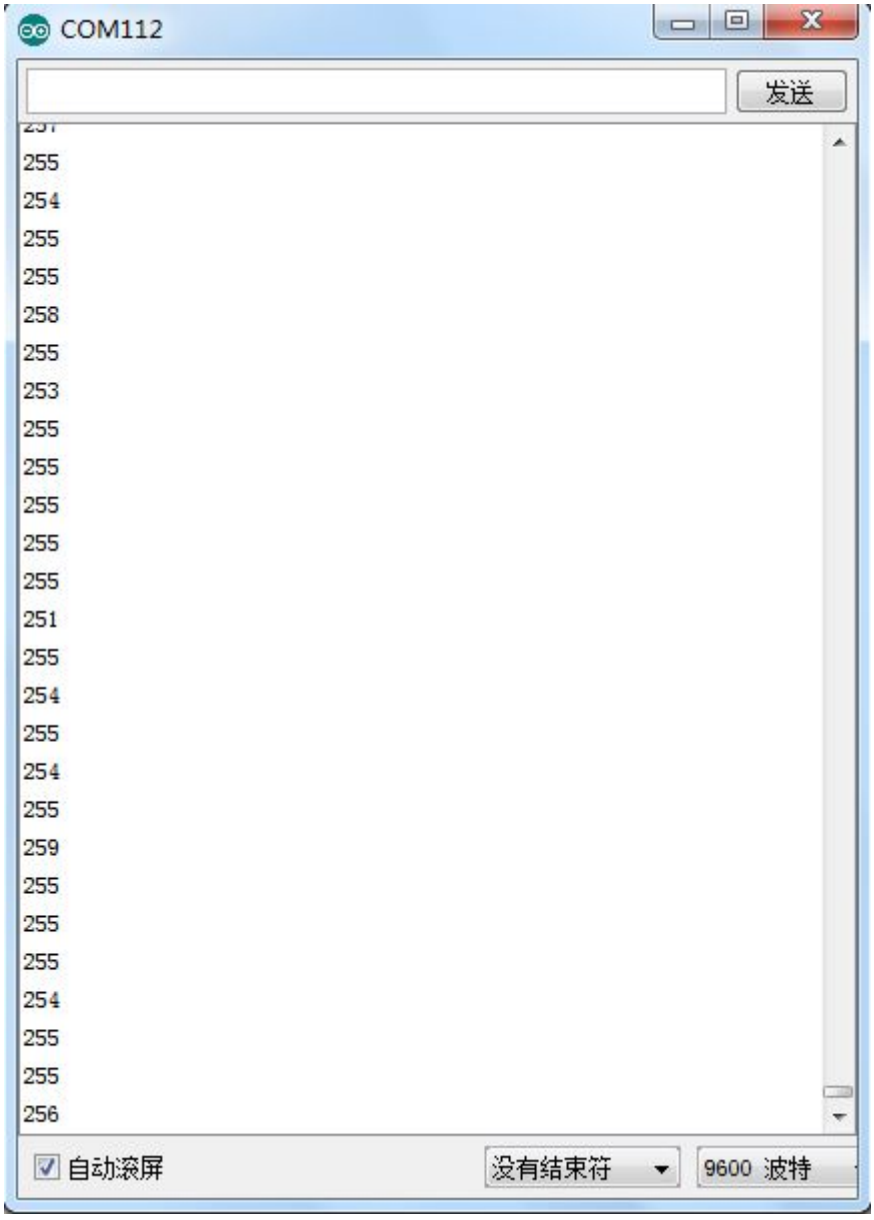
```
on the LED
     for(int i=0;i<5;i++){
     delay(1000);                              // wait for 5s
     }
   }
   else{
     digitalWrite(LEDpin,LOW);            // turn off the LED
       }
}
```

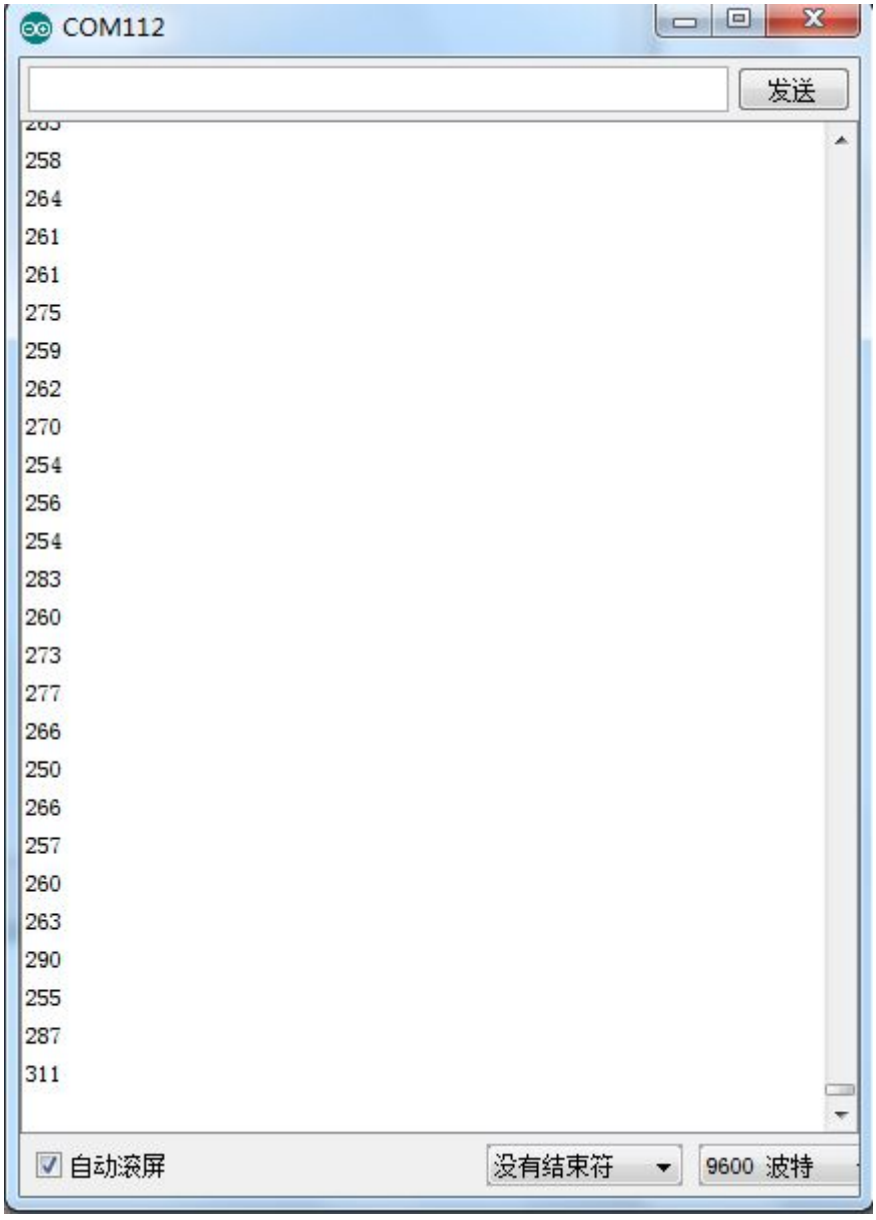
Program description: By rotating the potentiometer, the analog value of A0 changes; After adjusting the potentiometer, blow air into the MIC, and observe data in the serial monitor. For example, the displayed data is less than 300 before blowing; after blowing, data is more than 300. Setup code if (Soundvalue > 300), control the on and off of the light; the on time of the light is controlled by the code for(int i=0;i<5;i++) { delay(1000); }, so the light is on 5*1s.

//

Result

Connect the wire according to the connection diagram. When the light is out, data as shown in serial port monitor as pic 1; when blowing air to the MIC, light is on for 5 seconds and data as shown in pic 2.


COM112
发送
255
254
255
255
258
255
253
255
255
255
255
255
255
251
255
254
255
254
255
259
255
255
255
254
255
255
256
自动滚屏
没有结束符
9600 波特


COM112
发送
258
264
261
261
275
259
262
270
254
256
254
283
260
273
277
266
250
266
257
260
263
290
255
287
311
自动滚屏
没有结束符
9600 波特

****

Project 22: Voltmeter

Introduction

Today, let's learn how to use Arduino serial communication and analog ports. We have introduced serial ports before. It can measure voltage 0-5V, and return corresponding 0-1023 value. Today, we will use Arduino analog to make a 0-5V voltmeter.

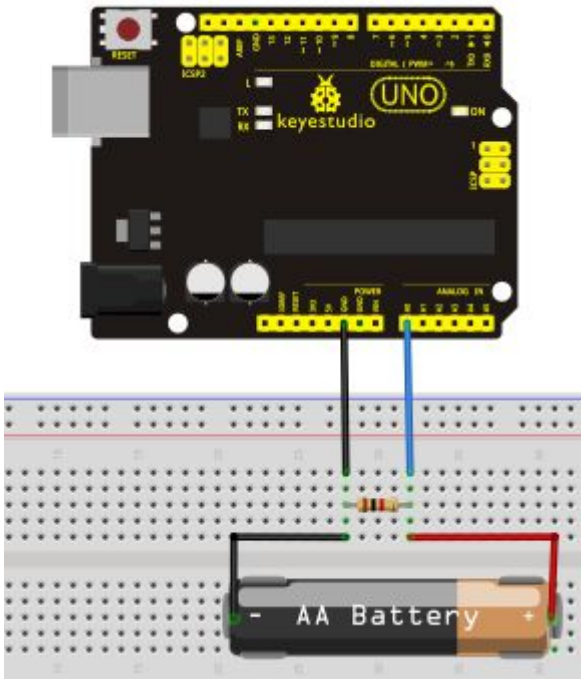
Note: in this experiment, there is no complex protective circuit. So please do not use more that 2 cells of AA batteries. Besides, do not use it to measure lithium battery or other power supply!

Hardware required

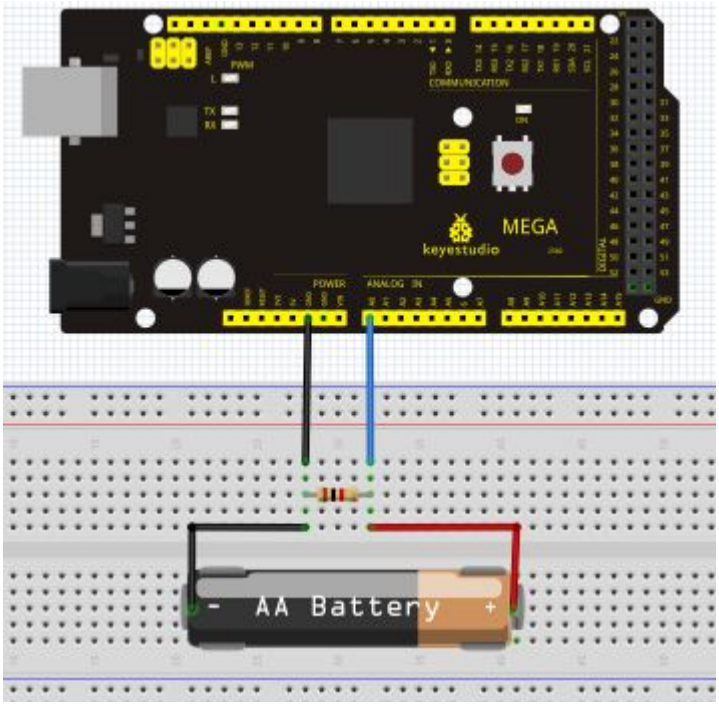
Controller board *1
1KΩ resistor *1
USB cable *1
Breadboard *1
Breadboard jumper wires * several

Circuit connection

Connection for R3:



Connection for 2560 R3:



Sample program

```
//////////////////////////////////////////////////////////
 float temp;    // create a floating-point type variable temp as storage space for storing data
  void setup()
  {
 Serial.begin(9600);      // use 9600 baud rate to have serial communication
}
void loop()
{
 int V1 = analogRead(A0);
// Read the voltage data from A0 port and store it in the newly created integer variables V1; measurement range of voltage from analog port is 0 to 5V; return value of 0-1023.
float vol = V1*(5.0 / 1024.0);
// We convert V1 value into actual voltage value and store it into floating-point variable vol
if (vol == temp)
  // The judgment here is used to filter repeated data; only voltage value that is differ than the last one will be output.
{
temp = vol; // After comparison is completed, store the value in variable temp
 }
else
{
Serial.print(vol);     // Serial port outputs voltage value, in the same line
```

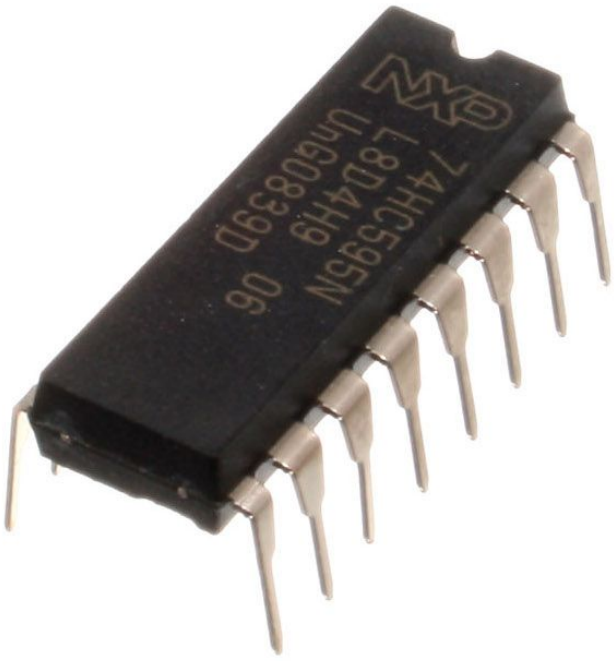
```
Serial.println(" V"); // Serial port outputs character V, and begin a new line
  temp = vol;
delay(1000);   // Wait 1 second after the output is complete for controlling the data refresh rate.
 }
}
//////////////////////////////////////////////////////////
```

Result

Click and open the serial port monitor; use the red line to measure battery positive pole, black line for negative pole. Serial monitor will refresh the voltage at 1 time/second. It is normal if there is fluctuation between two voltage values because it is, after all, a low accuracy test.

****

Project 23: 74HC595



Introduction

To put it simply, 74HC595 is a combination of 8-digit shifting register, memorizer and equipped with tri-state output. Here, we use it to control 8 LEDs. You may wonder why use a 74HC595 to control LED? Well, think about how many I/O it takes for an Arduino to control 8 LEDs? Yes, 8. For an Arduino 168, it has only 20 I/O including analog ports. So, to save port resources, we use 74HC595 to reduce the number of ports it needs. Using 74HC595 enables us to use 3 digital

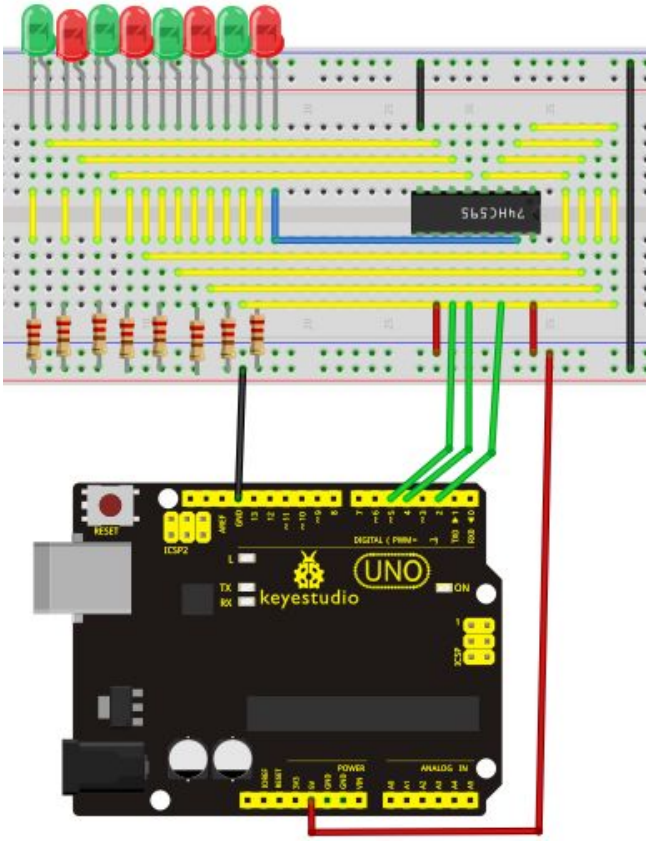
I/O port to control 8 LEDs!

Hardware required

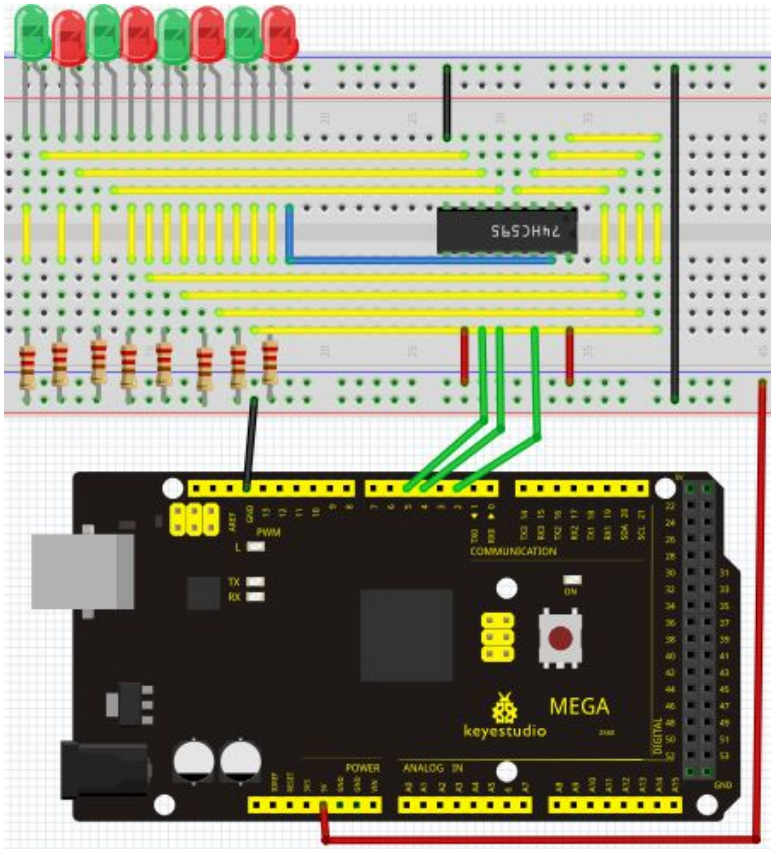
74HC595 chip*1
Red M5 LED*4
Green M5 LED*4
220Ω resistor*8
Breadboard*1
Breadboard jumper wires
Note: for pin 13 OE port of 74HC595, it needs to be connected to GND

Circuit connection

Connection for R3:



Connection for 2560 R3:



The circuit may seem completed, but once you give it a good look, you will find it easy!

Sample program

```
//////////////////////////////////////////////////////////
int data = 2;// set pin 14 of 74HC595as data input pin SI
int clock = 5;// set pin 11 of 74hc595 as clock pin SCK
int latch = 4;// set pin 12 of 74hc595 as output latch 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);
```

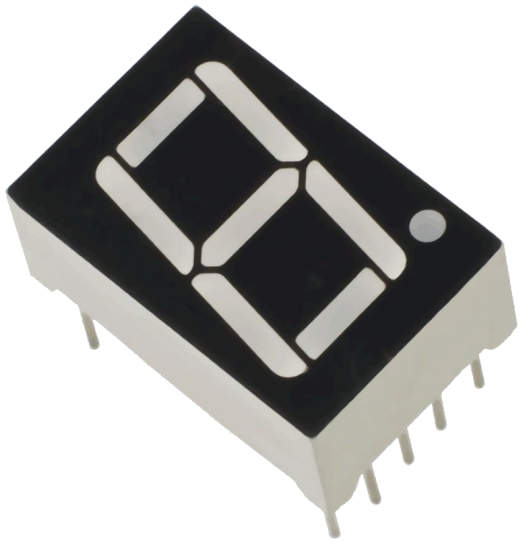
```
delay(500);
}
}
void updateLEDs(int value)
{
digitalWrite(latch, LOW);//
shiftOut(data, clock, MSBFIRST, ~value);// serial data “output”, high level first
digitalWrite(latch, HIGH);// latch
}
////////////////////////////////////////////////////////
```

Result

After downloading the program, you can see 8 LEDs displaying 8-bit binary number.

****

Project 24: 1-digit LED segment display

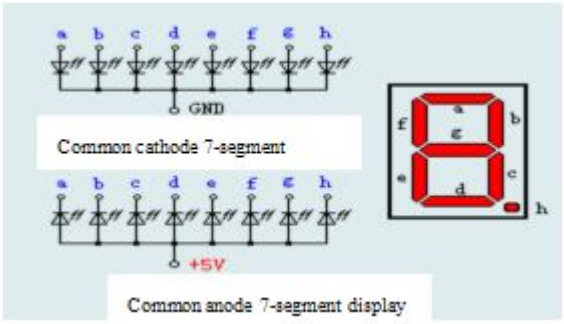


Introduction

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



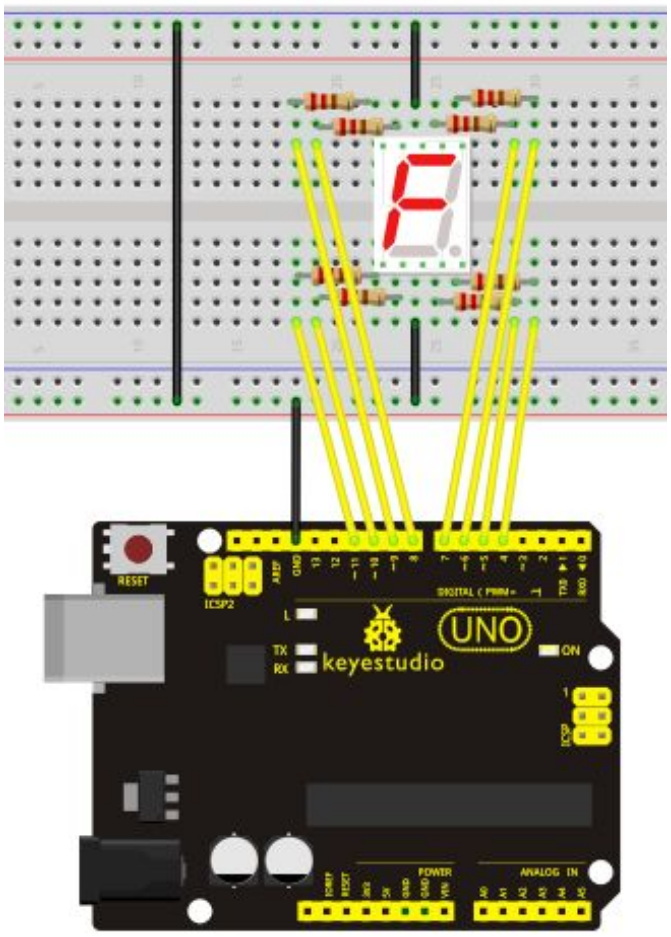
Each segment of the display consists of an LED. So when you use it, you also need use a current-limiting resistor. Otherwise, LED will be burnt out. In this experiment, we use a common cathode display. As we mentioned above, for common cathode display, connect the common cathode (COM) to GND. When the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

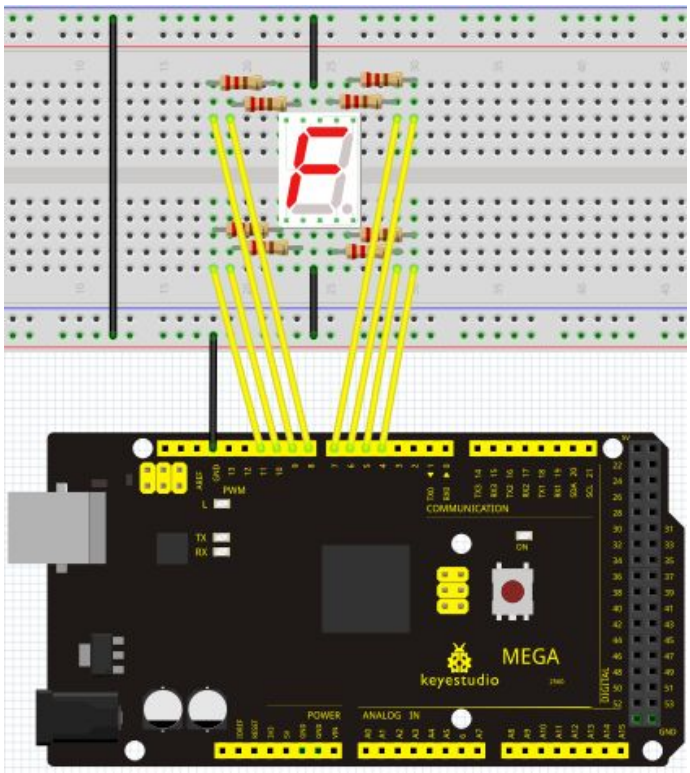
Eight-segment display*1
220Ω resistor*8
Breadboard*1
Breadboard jumper wires*several

Circuit connection

Connection for uno R3:



Connection for 2560 R3:



Sample program

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
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.

****

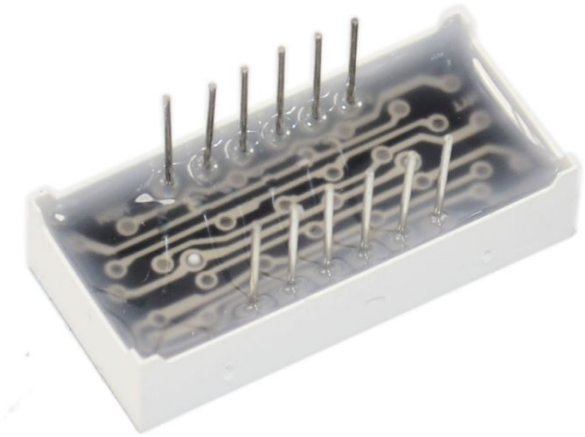
Project 25: 4-digit LED segment display

Introduction

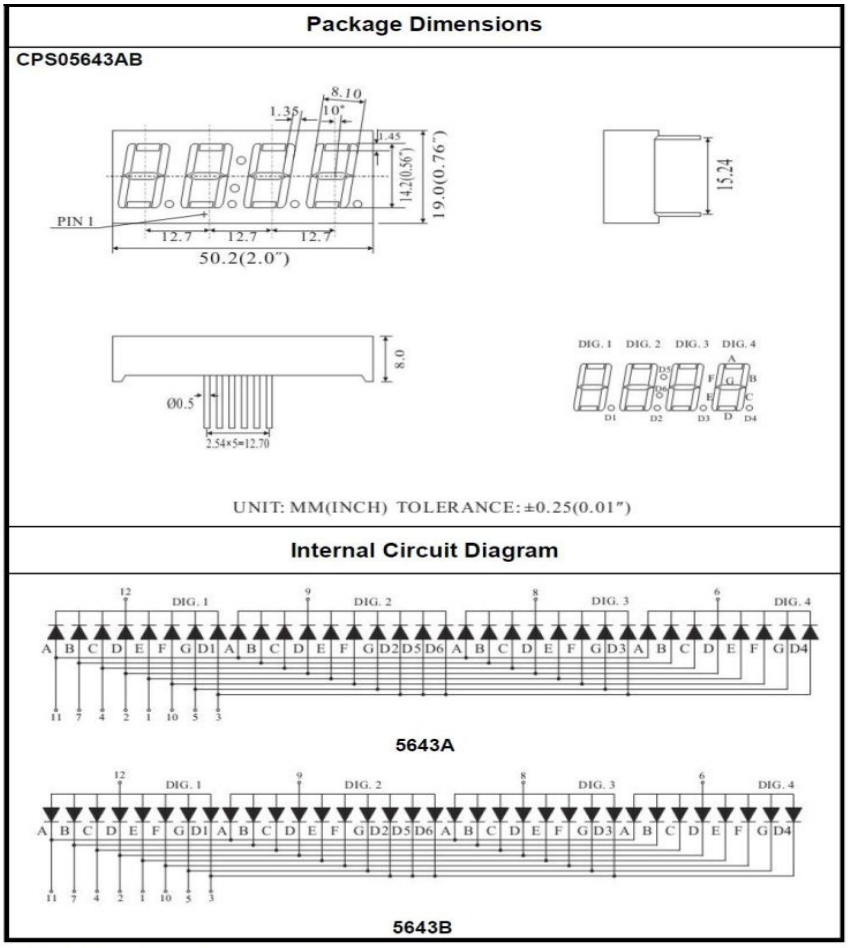
In this experiment, we use an Arduino to drive a common anode, 4-digit, 7-segment LED display. For LED display, current-limiting resistors are indispensable. There are two wiring method for Current-limiting resistor. One is to connect one resistor for each anode, 4 in totals for d1-d4 anode. An advantage for this method is that it requires fewer resistors, only 4. But it cannot maintain consistent brightness, 1 the brightest, 8, the least bright. Another method is to connect one resistor to each pin. It guarantees consistent brightness, but requires more resistors. In this experiment, we use 8 220Ω resistors (we use 220Ω resistors because no 100Ω resistor available. If you use 100Ω, the displaying will be brighter).

Circuit connection

For 4-digit displays, there are 12 pins in total. When you place the decimal point downward , the pin on the lower left part is refer to as 1, the upper left part 12.

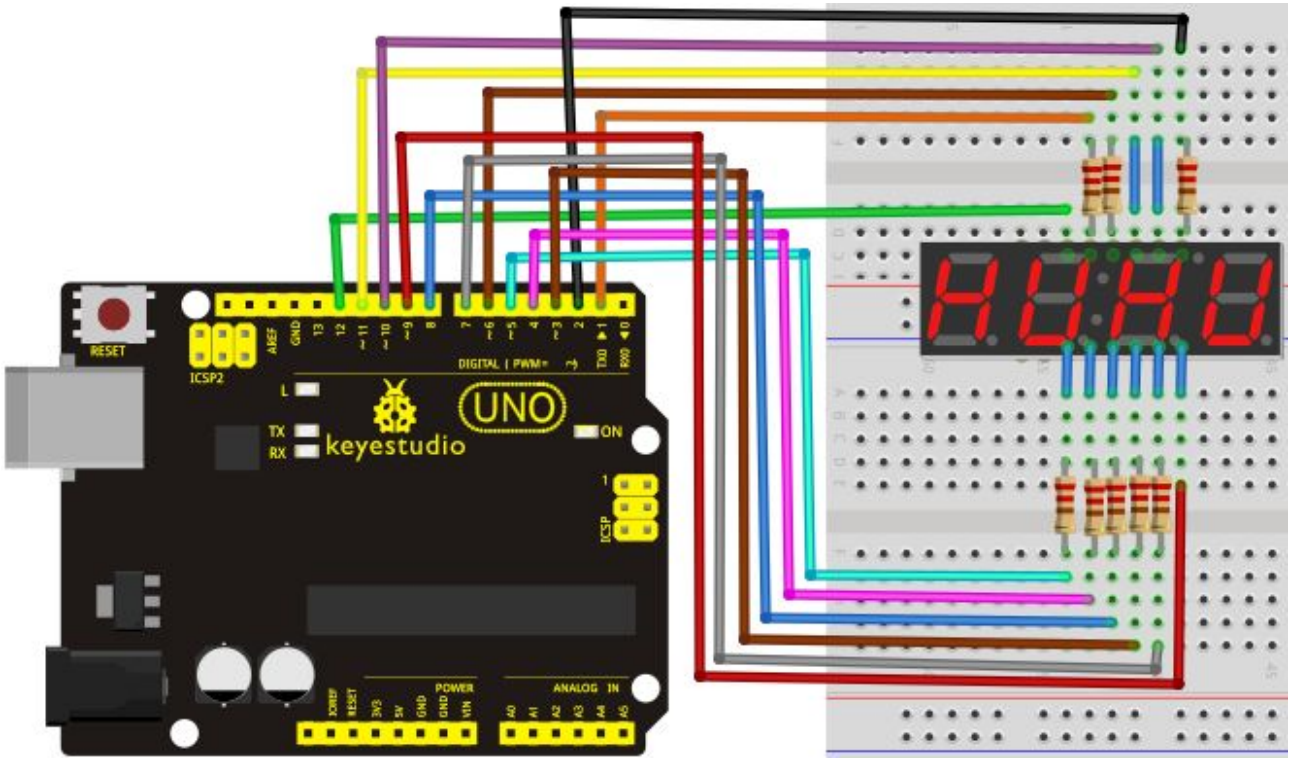


Manual for LED segment display

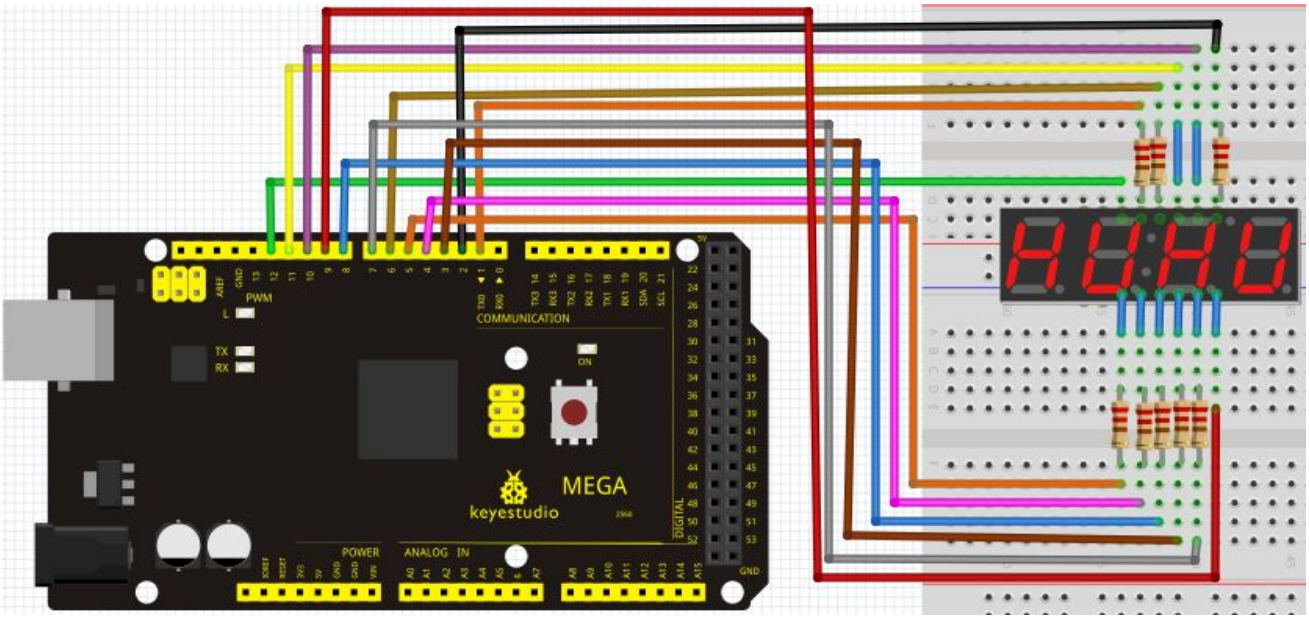


Four Digits Displays Series

Connection for R3:



Connection for 2560 R3:



Sample Program

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

      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);
```

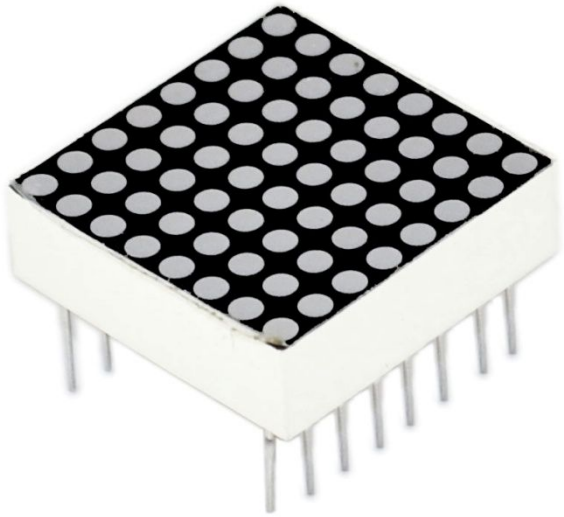
```
    digitalWrite(f, LOW);
    digitalWrite(g, LOW);
    digitalWrite(dp,LOW);
}
void Num_8()
{
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, HIGH);
    digitalWrite(f, HIGH);
    digitalWrite(g, HIGH);
    digitalWrite(dp,LOW);
}
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 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)//    take x as coordinate and display number
{
   WeiXuan(x);
   pickNumber(Number);
   delay(1);
   Clear() ; // clear the screen
}
//////////////////////////////////////////////////////////
```

Result

Download the above code to the controller board and see the result.
The experiment result displays 1234 on the display.
Note: if it's not displaying correctly, check the wiring.
Thank you.
****

Project 26: 8*8 LED matrix



Introduction

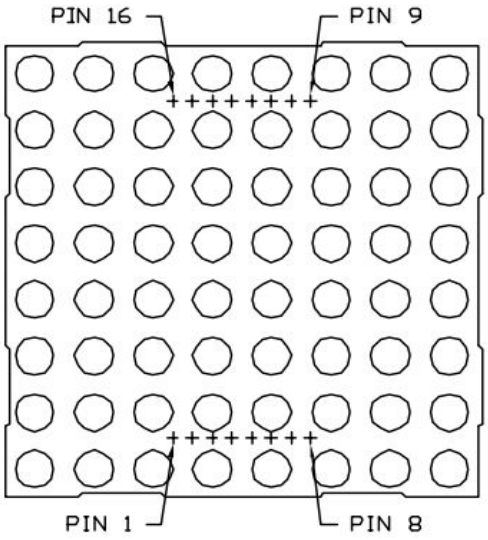
With low-voltage scanning, LED dot-matrix displays have advantages such as power saving, long service life, low cost, high brightness, wide angle of view, long visual range, waterproof, and numerous specifications. LED dot-matrix displays can meet the needs of different applications and thus have a broad development prospect. This time, we will conduct an LED dot-matrix experiment to experience its charm firsthand.

Hardware required

1 * Uno board
1 * 8*8 dot-matrix
8 * Resistor (220Ω)
1 * Breadboard
2 * 74HC595
1 * USB cable
Jumper wires

Circuit connection

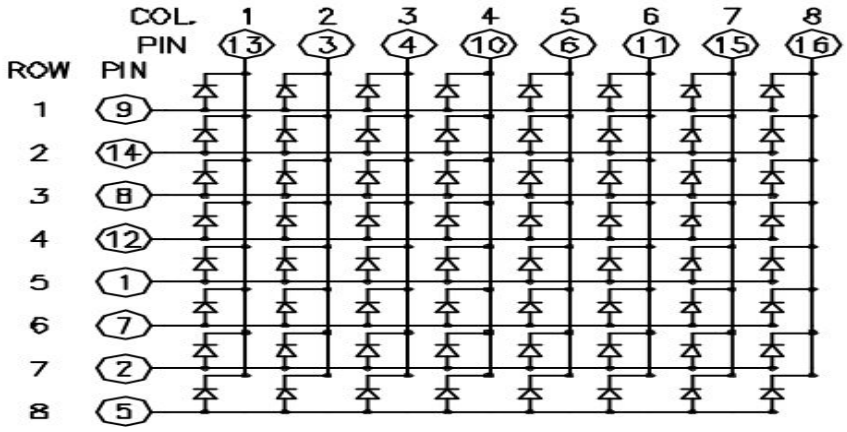
The external view of a dot-matrix is shown as follows:



The display principle of the 8*8 dot-matrix:

The 8*8 dot-matrix is made up of sixty-four LEDs, and each LED is placed at the cross point of a row and a column. When the electrical level of a certain row is 1 and the electrical level of a certain column is 0, the corresponding LED will light up. If you want to light the LED on the first dot, you should set pin 9 to high level and pin 13 to low level. If you want to light LEDs on the first row, you should set pin 9 to high level and pins 13, 3, 4, 10, 6, 11, 15 and 16 to low level. If you want to light the LEDs on the first column, set pin 13 to low level and pins 9, 14, 8, 12, 1, 7, 2 and 5 to high level.

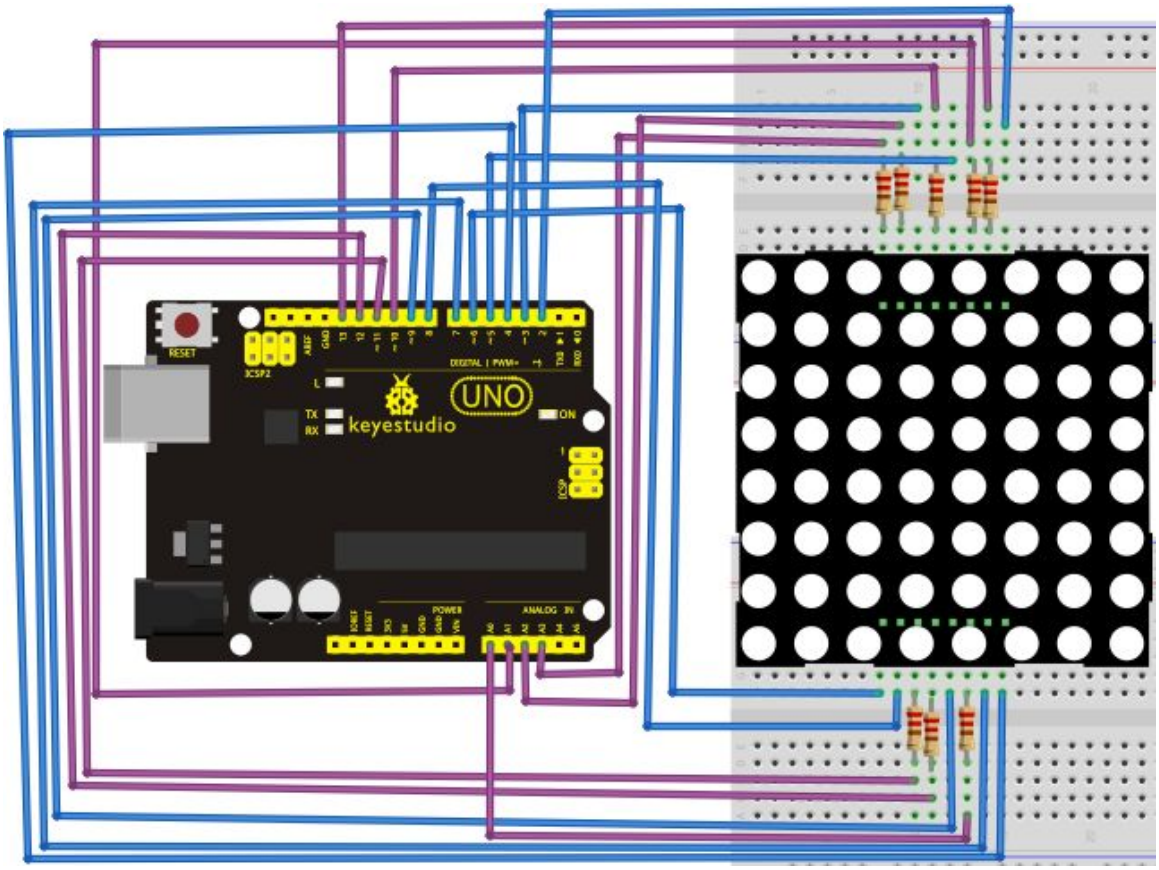
The internal view of a dot-matrix is shown as follows:



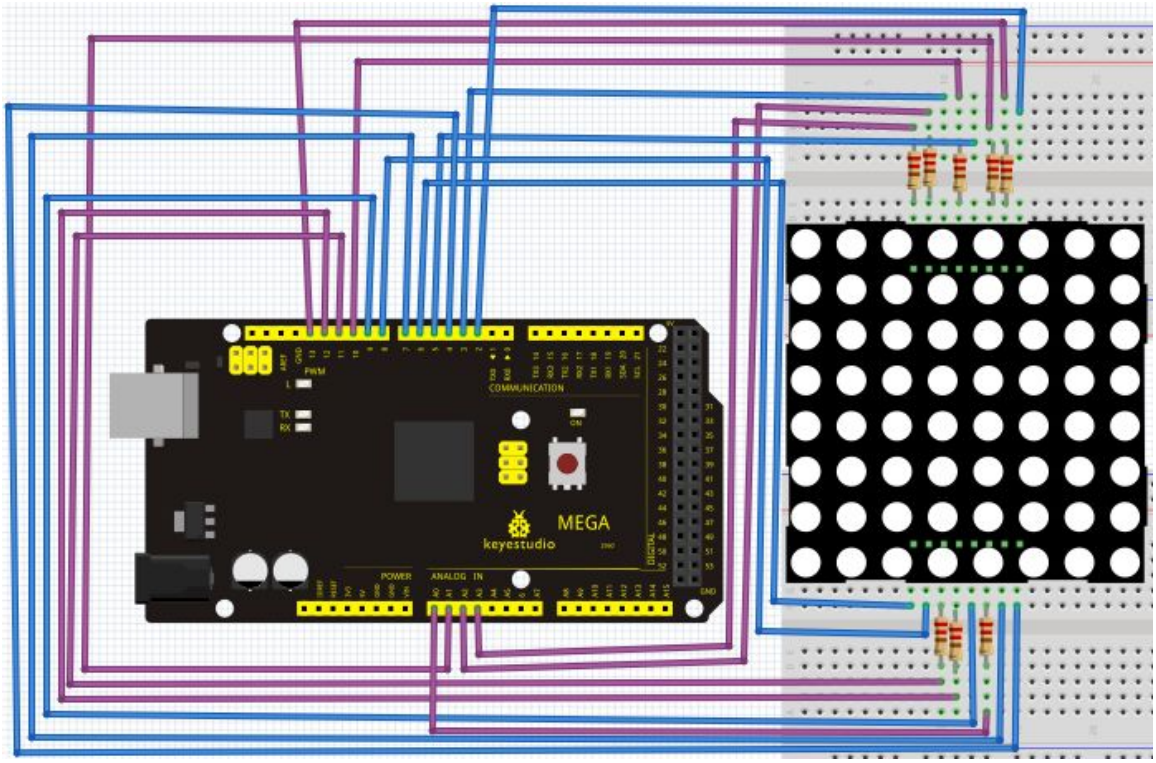
The principle of 74HC595 has been previously illustrated. One chip is used to control the rows of the dot-matrix while the other chip is used to control the columns.

Connect circuit as shown in the following diagram:

Connection for R3:



Connection for 2560 R3:



Sample program for displaying "0"

```
//////////////////////////////////////////////////////////
// set an array to store character of "0"
unsigned char Text[]={0x00,0x1c,0x22,0x22,0x22,0x22,0x22,0x1c};
void Draw_point(unsigned char x,unsigned char y)// point drawing function
{ clear_();
    digitalWrite(x+2, HIGH);
    digitalWrite(y+10, LOW);
    delay(1);
}
void show_num(void)// display function, call point drawing 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 screen
{for(int i=2;i<10;i++)
  digitalWrite(i, LOW);
  for(int i=0;i<8;i++)
  digitalWrite(i+10, HIGH);
}
```

//

Result

Burn the program into Uno board The dot-matrix will display 0.

****

Project 27: 1602 LCD

Introduction

In this experiment, we use an Arduino to drive the 1602 LCD.
1602 LCD has wide applications. In the beginning, 1602 LCD uses a HD44780 controller. Now, almost all 1602 LCD module uses a compatible IC, so their features are basically the same.

1602LCD main parameters:

- Display capacity: 16 * 2 characters.
- Chip operating voltage: 4.5 ~ 5.5V.
- Working current: 2.0mA (5.0V).
- Optimum working voltage of the module is 5.0V.
- Character size: 2.95 * 4.35 (W * H) mm.

Pin description of 1602 LCD

No.	Mark	Pin description	No.	Mark	Pin description
1	VSS	Power GND	9	D2	Date I/O
2	VDD	Power positive	10	D3	Date I/O
3	VL	LCD voltage bias signal	11	D4	Date I/O
4	RS	Select data/command(V/L)	12	D5	Date I/O

5	R/W	Select read/write(H/L)	13	D6	Date I/O
6	E	Enable signal	14	D7	Date I/O
7	D0	Date I/O	15	BLA	Back light power positive
8	D1	Date I/O	16	BLK	Back light power negative

Interface description:

1. two power sources, one for module power, another one for back light, generally use 5V. In this project, we use 3.3V for back light.
2. VL is the pin for adjusting contrast ratio; it usually connects a potentiometer(no more than 5KΩ) in series for its adjustment. In this experiment, we use a 1KΩ resistor. For its connection, it has 2 methods, namely high potential and low potential. Here, we use low potential method; connect the resistor and then the GND.
3. RS is a very common pin in LCD. It's a selecting pin for command/data. When the pin is in high level, it's in data mode; when it's in low level, it's in command mode.
4. RW pin is also very common in LCD. It's a selecting pin for read/write. When the pin is in high level, it's in read operation; when it's in low level, it's in write operation.
5. E pin is also very common in LCD. Usually, when the signal in the bus is stabilized, it sends out a positive pulse requiring read operation. When this pin is in high level, the bus is not allowed to have any change.
6. D0-D7 is 8-bit bidirectional parallel bus, used for command and data transmission.
7. BLA is anode for back light; BLK, cathode for back light.

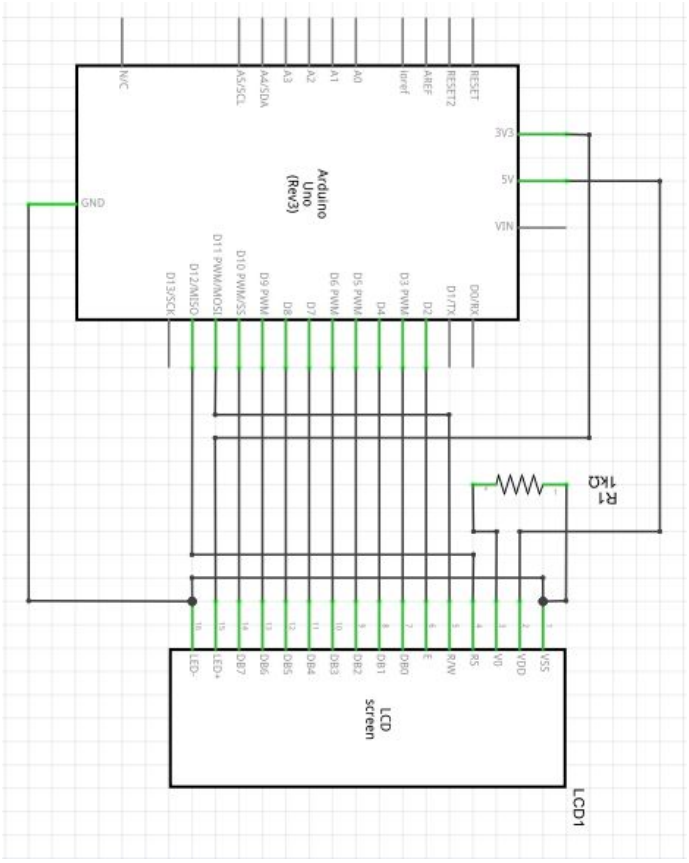
4 basic operations of 1602LCD:

Read status	input	RS=L, R/W=H, E=H	output	D0-D7=status word
Write command	input	RS=L, R/W=H, D0-D7=command code, E=high pulse	output	none
Read data	input	RS=H, R/W=H, E=H	output	D0-D7=data
Write data	input	RS=H, R/W=L, D0-D7=data, E=high pulse	output	none

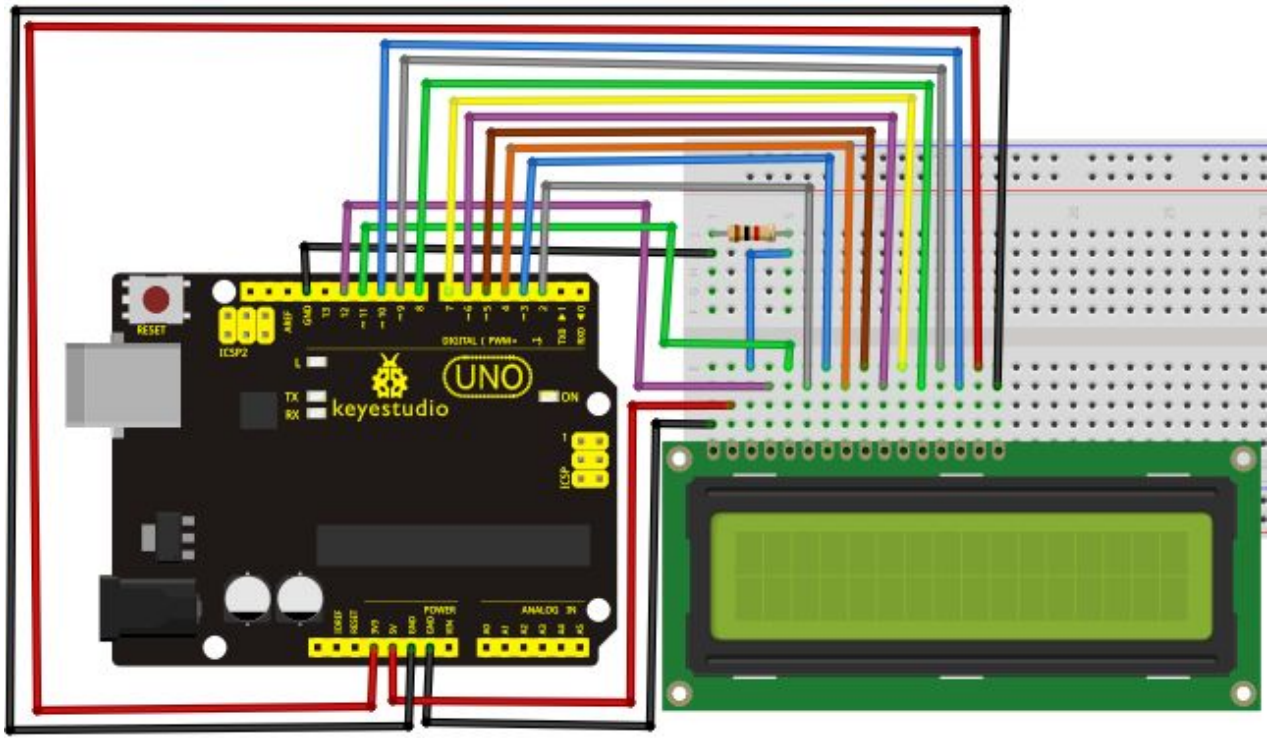
Connection & sample program

1602 can directly communicate with Arduino. According to the product manual, it has two connection methods, namely 8-bit connection and 4-bit connection.

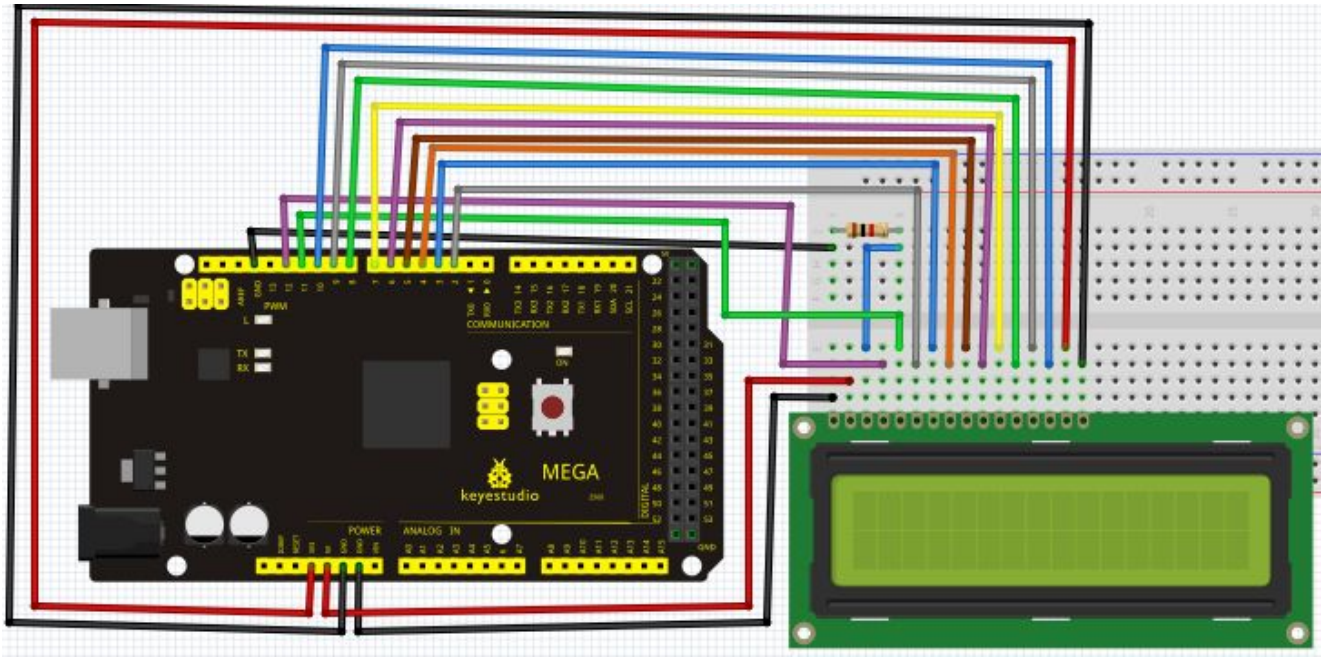
8-bit connection method:



Connection for R3:



Connection for 2560 R3:



Sample code A:

```
//////////////////////////////////////////////////////////
int DI = 12;
int RW = 11;
int DB[] = {3, 4, 5, 6, 7, 8, 9, 10};// use array to select pin for bus
int Enable = 2;

void LcdCommandWrite(int value) {
// define all pins
int i = 0;
for (i=DB[0]; i <= DI; i++) // assign value for bus
{
   digitalWrite(i,value & 01);// for 1602 LCD, it uses D7-D0( not D0-D7) for signal identification;
here, it's used for signal inversion.
   value >>= 1;
}
digitalWrite(Enable,LOW);
delayMicroseconds(1);
digitalWrite(Enable,HIGH);
delayMicroseconds(1);  // wait for 1ms
digitalWrite(Enable,LOW);
delayMicroseconds(1);  // wait for 1ms
}
```

```
void LcdDataWrite(int value) {
// initialize all pins
int i = 0;
digitalWrite(DI, HIGH);
digitalWrite(RW, LOW);
for (i=DB[0]; i <= DB[7]; i++) {
   digitalWrite(i,value & 01);
   value >>= 1;
}
digitalWrite(Enable,LOW);
delayMicroseconds(1);
digitalWrite(Enable,HIGH);
delayMicroseconds(1);
digitalWrite(Enable,LOW);
delayMicroseconds(1);  // wait for 1ms
}

void setup (void) {
int i = 0;
for (i=Enable; i <= DI; i++) {
   pinMode(i,OUTPUT);
}
delay(100);
// initialize LCD after a brief pause
// for LCD control
LcdCommandWrite(0x38);  // select as 8-bit interface, 2-line display, 5x7 character size
delay(64);
LcdCommandWrite(0x38);  // select as 8-bit interface, 2-line display, 5x7 character size
delay(50);
LcdCommandWrite(0x38);  // select as 8-bit interface, 2-line display, 5x7 character size
delay(20);
LcdCommandWrite(0x06);  // set input mode
                        // auto-increment, no display of shifting
delay(20);
LcdCommandWrite(0x0E);  // display setup
                        // turn on the monitor, cursor on, no flickering
delay(20);
LcdCommandWrite(0x01);  // clear the scree, cursor position returns to 0
delay(100);
LcdCommandWrite(0x80);  //    display setup
                        //    turn on the monitor, cursor on, no flickering
```

```
delay(20);
}

void loop (void) {
  LcdCommandWrite(0x01);  // clear the scree, cursor position returns to 0
  delay(10);
  LcdCommandWrite(0x80+3);
  delay(10);
  // write in welcome message
  LcdDataWrite('W');
  LcdDataWrite('e');
  LcdDataWrite('l');
  LcdDataWrite('c');
  LcdDataWrite('o');
  LcdDataWrite('m');
  LcdDataWrite('e');
  LcdDataWrite(' ');
  LcdDataWrite('t');
  LcdDataWrite('o');
  delay(10);
  LcdCommandWrite(0xc0+1);  // set cursor position at second line, second position
  delay(10);
  LcdDataWrite('g');
  LcdDataWrite('e');
  LcdDataWrite('e');
  LcdDataWrite('k');
  LcdDataWrite('-');
  LcdDataWrite('w');
  LcdDataWrite('o');
  LcdDataWrite('r');
  LcdDataWrite('k');
  LcdDataWrite('s');
  LcdDataWrite('h');
  LcdDataWrite('o');
  LcdDataWrite('p');
  delay(5000);
  LcdCommandWrite(0x01);  // clear the screen, cursor returns to 0
  delay(10);
  LcdDataWrite('I');
  LcdDataWrite(' ');
  LcdDataWrite('a');
  LcdDataWrite('m');
```

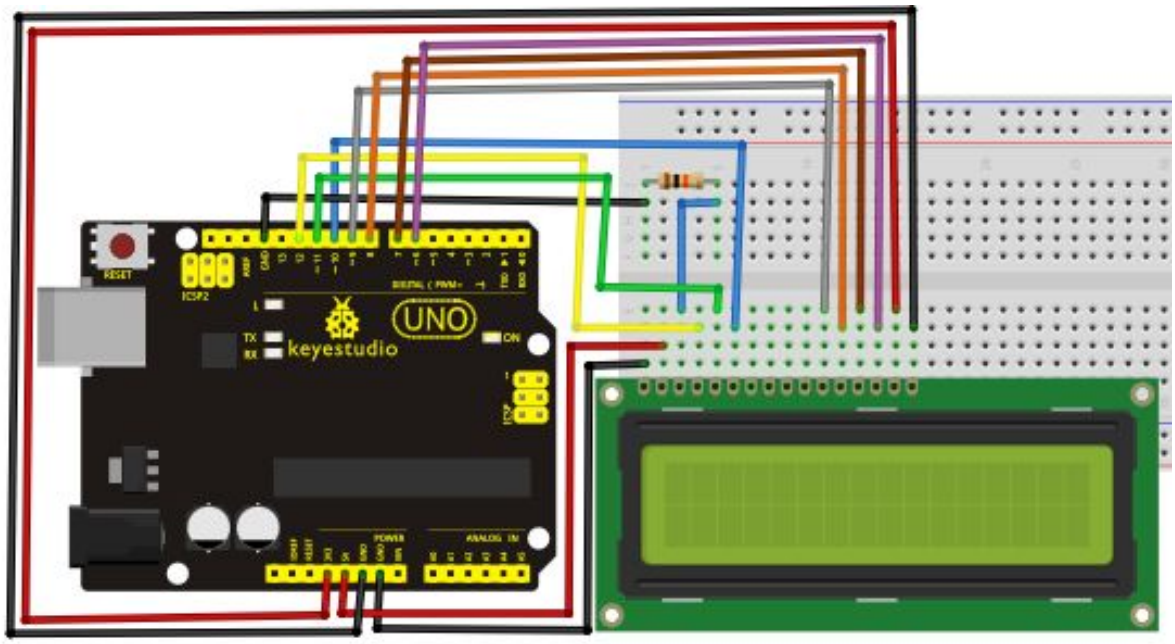
```
  LcdDataWrite(' ');
  LcdDataWrite('h');
  LcdDataWrite('o');
  LcdDataWrite('n');
  LcdDataWrite('g');
  LcdDataWrite('y');
  LcdDataWrite('i');
  delay(3000);
   LcdCommandWrite(0x02); // set mode as new characters replay old ones, where there is no new ones remain the same
  delay(10);
  LcdCommandWrite(0x80+5); // set cursor position at first line, sixth position
  delay(10);
  LcdDataWrite('t');
  LcdDataWrite('h');
  LcdDataWrite('e');
  LcdDataWrite(' ');
  LcdDataWrite('a');
  LcdDataWrite('d');
  LcdDataWrite('m');
  LcdDataWrite('i');
  LcdDataWrite('n');
  delay(5000);
}
//////////////////////////////////////////////////////////
```

4-bit connection method:

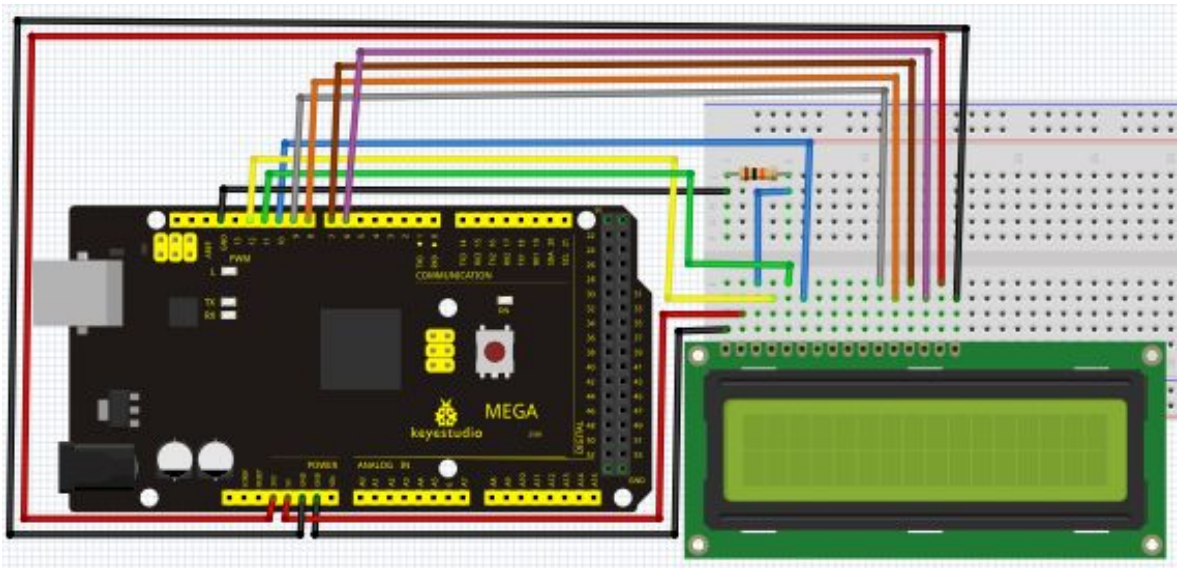
When using this module, 8-bit connection uses all the digital pins of the Arduino, leaving no pin for sensors. What then? We can use 4-bit connection.

Connection circuit:

Connection for R3:



Connection for 2560 R3:



After the connection, upload below code to the controller board and see how it goes.

Sample code B:

```
//////////////////////////////////////////////////////////
int LCD1602_RS=12;
int LCD1602_RW=11;
int LCD1602_EN=10;
int DB[] = { 6, 7, 8, 9};
char str1[]="Welcome to";
char str2[]="geek-workshop";
```

```
char str3[]="this is the";
char str4[]="4-bit interface";

void LCD_Command_Write(int command)
{
int i,temp;
digitalWrite( LCD1602_RS,LOW);
digitalWrite( LCD1602_RW,LOW);
digitalWrite( LCD1602_EN,LOW);

temp=command & 0xf0;
for (i=DB[0]; i <= 9; i++)
{
   digitalWrite(i,temp & 0x80);
   temp <<= 1;
}

digitalWrite( LCD1602_EN,HIGH);
delayMicroseconds(1);
digitalWrite( LCD1602_EN,LOW);

temp=(command & 0x0f)<<4;
for (i=DB[0]; i <= 10; i++)
{
   digitalWrite(i,temp & 0x80);
   temp <<= 1;
}

digitalWrite( LCD1602_EN,HIGH);
delayMicroseconds(1);
digitalWrite( LCD1602_EN,LOW);
}

void LCD_Data_Write(int dat)
{
int i=0,temp;
digitalWrite( LCD1602_RS,HIGH);
digitalWrite( LCD1602_RW,LOW);
digitalWrite( LCD1602_EN,LOW);

temp=dat & 0xf0;
for (i=DB[0]; i <= 9; i++)
```

```
{
   digitalWrite(i,temp & 0x80);
   temp <<= 1;
}

digitalWrite( LCD1602_EN,HIGH);
delayMicroseconds(1);
digitalWrite( LCD1602_EN,LOW);

temp=(dat & 0x0f)<<4;
for (i=DB[0]; i <= 10; i++)
{
   digitalWrite(i,temp & 0x80);
   temp <<= 1;
}

digitalWrite( LCD1602_EN,HIGH);
delayMicroseconds(1);
digitalWrite( LCD1602_EN,LOW);
}

void LCD_SET_XY( int x, int y )
{
 int address;
 if (y ==0)      address = 0x80 + x;
 else            address = 0xC0 + x;
 LCD_Command_Write(address);
}

void LCD_Write_Char( int x,int y,int dat)
{
 LCD_SET_XY( x, y );
 LCD_Data_Write(dat);
}

void LCD_Write_String(int X,int Y,char *s)
{
    LCD_SET_XY( X, Y );      // address setup
    while (*s)               // write character string
    {
     LCD_Data_Write(*s);
     s ++;
```

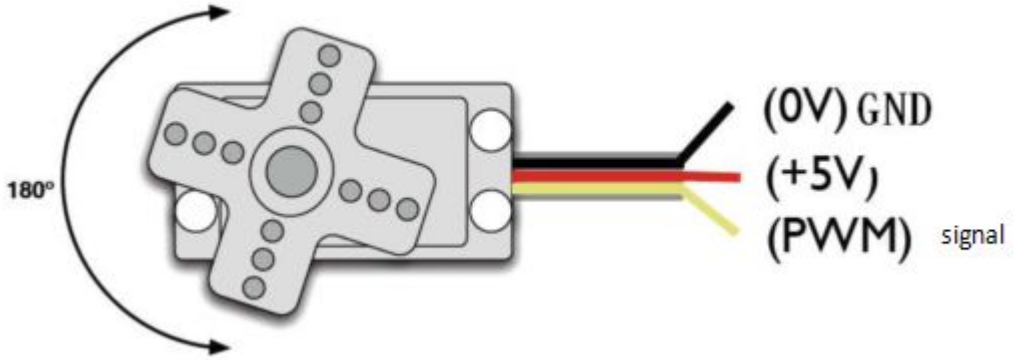
```
    }
}

void setup (void)
{
 int i = 0;
 for (i=6; i <= 12; i++)
   {
    pinMode(i,OUTPUT);
   }
 delay(100);
 LCD_Command_Write(0x28);// 4 wires, 2 lines 5x7
 delay(50);
 LCD_Command_Write(0x06);
 delay(50);
 LCD_Command_Write(0x0c);
 delay(50);
 LCD_Command_Write(0x80);
 delay(50);
 LCD_Command_Write(0x01);
 delay(50);

}

void loop (void)
{
  LCD_Command_Write(0x01);
  delay(50);
  LCD_Write_String(3,0,str1);// line 1, start at the fourth address
  delay(50);
  LCD_Write_String(1,1,str2);// line 2, start at the second address
  delay(5000);
  LCD_Command_Write(0x01);
  delay(50);
  LCD_Write_String(0,0,str3);
  delay(50);
  LCD_Write_String(0,1,str4);
  delay(5000);

}
////////////////////////////////////////////////////////////
**********************************************************************************
```

Project 28: 9g servo control

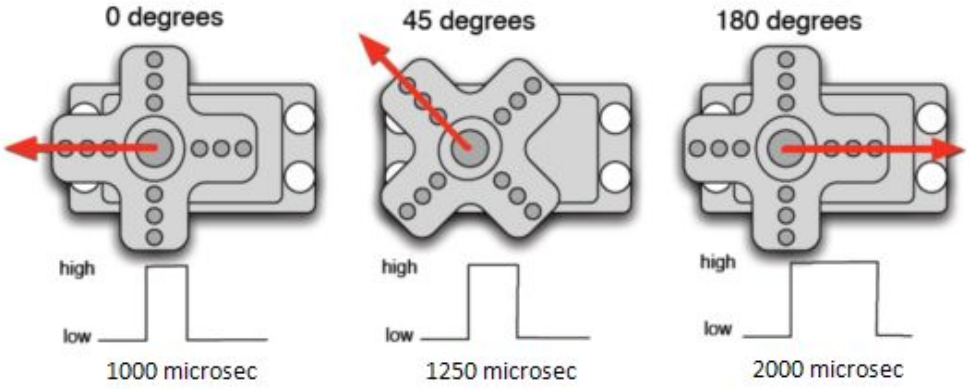
Introduction

Servomotor is a position control rotary actuator. It mainly consists of housing, circuit board, core-less motor, gear and position sensor. The receiver or MCU outputs a signal to the servomotor. The motor has a built-in reference circuit that gives out reference signal, cycle of 20ms and width of 1.5ms. The motor compares the acquired DC bias voltage to the voltage of the potentiometer and outputs a voltage difference. The IC on the circuit board will decide the rotate direction accordingly and drive the core-less motor. The gear then pass the force to the shaft. The sensor will determine if it has reached the commanded position according to the feedback signal. Servomotors are used in control systems that requires to have and maintain different angles. When the motor speed is definite, the gear will cause the potentiometer to rotate. When the voltage difference reduces to zero, the motor stops. Normally, the rotation angle range is among 0-180 degrees.

Servomotor comes with many specifications. But all of them have three connection wires, distinguished by brown, red, orange colors(different brand may have different color). Brown one is for GND, red one for power positive, orange one for signal
line.



The rotate angle of the servo motor is controlled by regulating the duty cycle of the PWM(Pulse-Width Modulation) signal. The standard cycle of the PWM signal is 20ms（50Hz）. Theoretically, the width is distributed between 1ms-2ms, but in fact, it's between 0.5ms-2.5ms. The width corresponds the rotate angle from 0° to 180°. But note that for different brand motor, the same signal may have different rotate angle.



After some basic knowledge, let's learn how to control a servomotor. For this experiment, you only need a servomotor and several jumper wires.

Hardware required

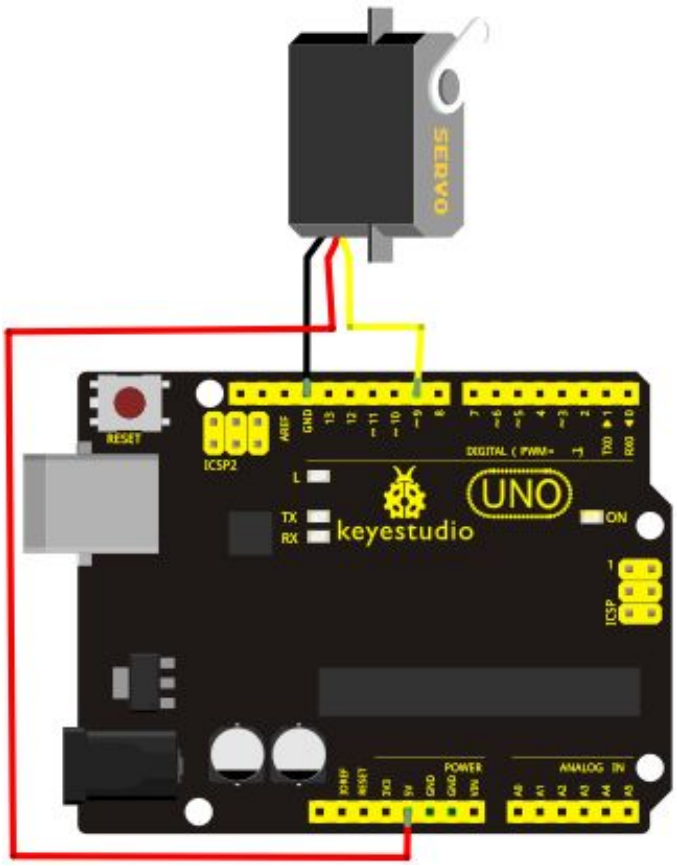
9G servo motor*1
Breadboard jumper wire*several

Connection & sample program

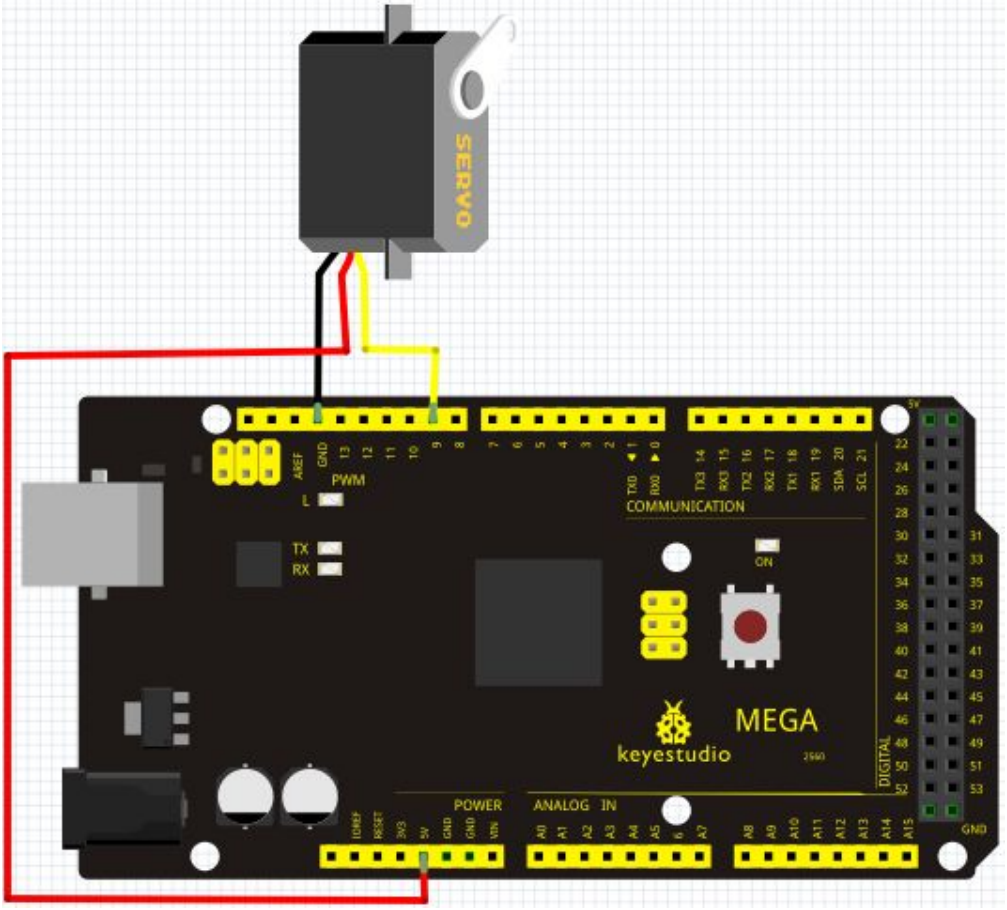
There are two ways to control a servomotor with Arduino. One is to use a common digital sensor port of Arduino to produce square wave with different duty cycle to simulate PWM signal and use that signal to control the positioning of the motor. Another way is to directly use the Servo function of the Arduino to control the motor. In this way, the program will be easier but it can only control two-contact motor because for the servo function, only digital pin 9 ang 10 can be used. The Arduino drive capacity is limited. So if you need to control more than one motor, you will need external power.

Method 1

Connection for R3:



Connection for 2560 R3:



Connect the motor to digital pin 9.

Compile a program to control the motor to rotate to the commanded angle input by the user and display the angle on the screen.

Sample program A

```
//////////////////////////////////////////////////////////
int servopin=9;// select digital pin 9 for servomotor signal line
int myangle;// initialize angle variable
int pulsewidth;// initialize width variable
int val;
void servopulse(int servopin,int myangle)// define a servo pulse function
{
pulsewidth=(myangle*11)+500;// convert angle to 500-2480 pulse width
digitalWrite(servopin,HIGH);// set the level of servo pin as “high”
delayMicroseconds(pulsewidth);// delay microsecond of pulse width
digitalWrite(servopin,LOW);// set the level of servo pin as “low”
delay(20-pulsewidth/1000);
}
void setup()
{
```

```
pinMode(servopin,OUTPUT);// set servo pin as “output”
Serial.begin(9600);// connect to serial port, set baud rate at “9600”
Serial.println("servo=o_seral_simple ready" ) ;
}
void loop()// convert number 0 to 9 to corresponding 0-180 degree angle, LED blinks
corresponding number of time
{
val=Serial.read();// read serial port value
if(val>'0'&&val<='9')
{
val=val-'0';// convert characteristic quantity to numerical variable
val=val*(180/9);// convert number to angle
Serial.print("moving servo to ");
Serial.print(val,DEC);
Serial.println();
for(int i=0;i<=50;i++) // giving the servo time to rotate to commanded position
{
servopulse(servopin,val);// use the pulse function
}
}
}
////////////////////////////////////////////////////////////
```

Method 2:

Let's first take a look at the Arduino built-in servo function and some of its common statements.

1. attach（interface）——select pin for servo, can only use pin 9 or 10.
2. write（angle）——used to control the rotate angle of the servo, can set the angle among 0 degree to 180 degree.
3. read（）——used to read the angle of the servo, consider it a function to read the value in the write() function.
4、attached（）——determine whether the parameter of the servo is sent to the servo pin.
5、detach（）—— disconnect the servo and the pin, and the pin(digital pin 9 or 10) can be used for PWM port.

Note: Note: the written form of the above statements are " servo variable name. specific statement ()", e.g. myservo. Attach (9).

Still, connect the servo to pin 9.

Sample program B：

```
////////////////////////////////////////////////////////////
#include <Servo.h>// define a header file. Special attention here, you can call the servo function
directly from Arduino's software menu
bar Sketch>Importlibrary>Servo, or input   #include <Servo.h>. Make sure there is a space
```

between #include and <Servo.h>. Otherwise, it will cause compile error.

```
Servo myservo;// define servo variable name
void setup()
{
myservo.attach(9);// select servo pin(9 or 10)
}
void loop()
{
myservo.write(90);// set rotate angle of the motor
}
```

//

Above are the two methods to control the servo. You can choose either one according to your liking or actual need.

****

Project 29: Rotary Encoder

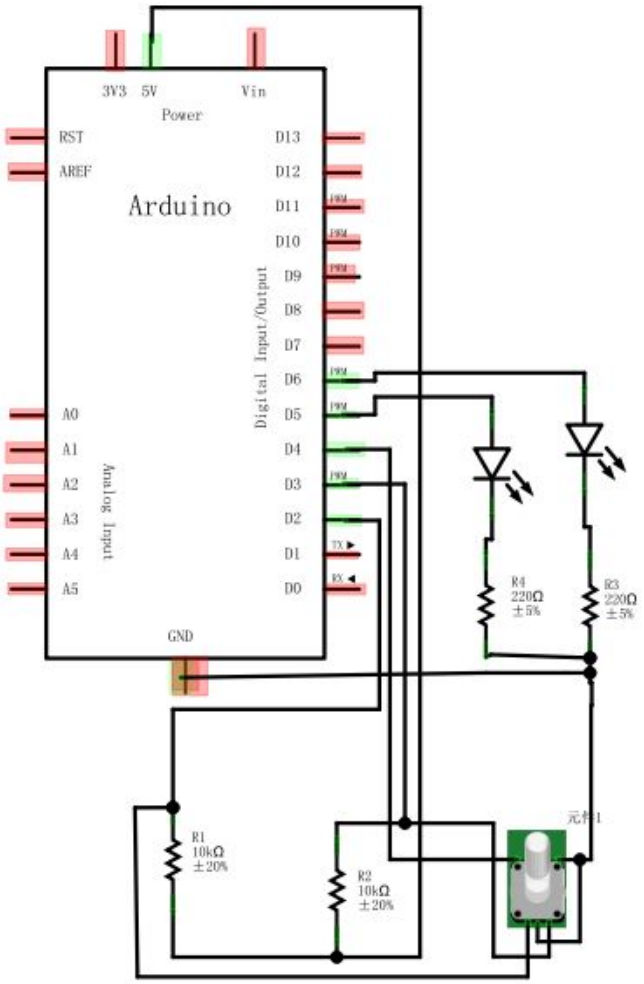
Introduction

The rotary encoder can count the pulse outputting times during the process of its rotation in positive and reverse direction by rotating. This rotating counting is unlimited, not like potential counting. It can be restored to initial state to count from 0 with the button on rotary encoder.

Hardware required

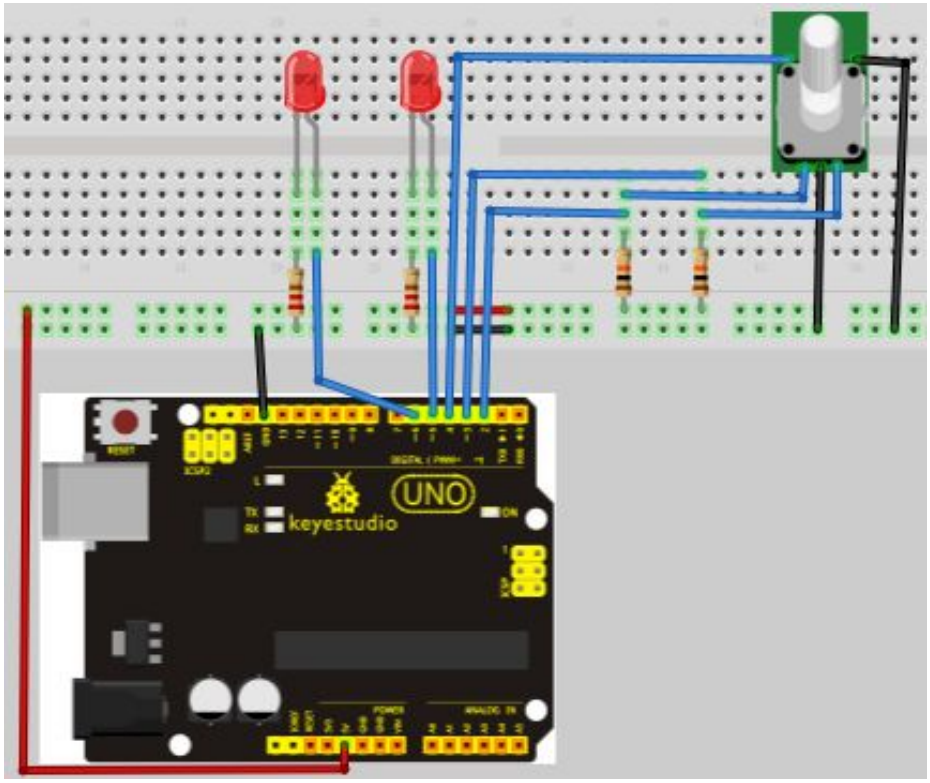
1 * Uno board
1 * Breadboard
1 * USB cable
1 * Rotar Encoder
2 * Red M5 LED
2 * 220Ω resistor
2 * 10KΩ resistor
Jumper wires

Schematic diagram

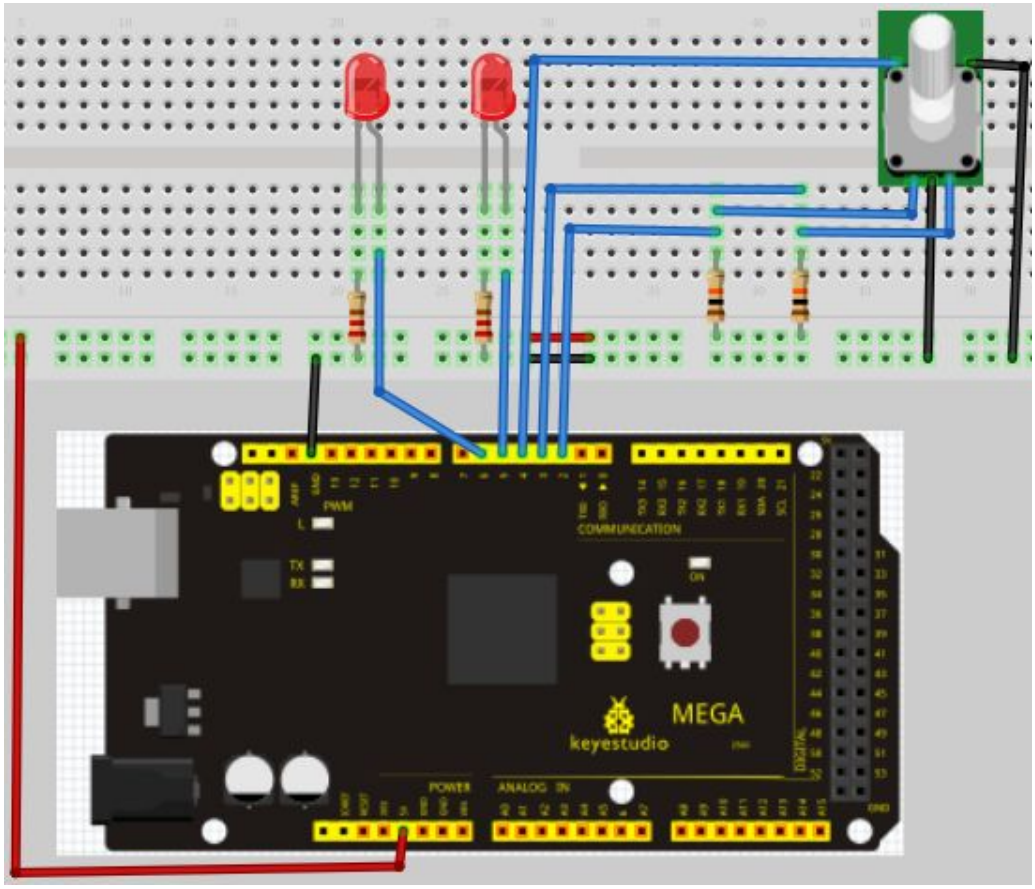


Circuit connection

Connection for uno R3:



Connection for 2560 R3:



Sample program

```
//////////////////////////////////////////////////////////
const int interruptA = 0;
const int interruptB = 1;
int CLK = 2;          // PIN2
int DAT = 3;          // PIN3
int BUTTON = 4;    // PIN4
int LED1 = 5;        // PIN5
int LED2 = 6;        // PIN6
int COUNT = 0;

void setup()
  {
    attachInterrupt(interruptA, RoteStateChanged, FALLING);
  // attachInterrupt(interruptB, buttonState, FALLING);
    pinMode(CLK, INPUT);
    digitalWrite(2, HIGH);    // Pull High Restance
    pinMode(DAT, INPUT);
    digitalWrite(3, HIGH);    // Pull High Restance
    pinMode(BUTTON, INPUT);
    digitalWrite(4, HIGH);    // Pull High Restance
    pinMode(LED1, OUTPUT);
    pinMode(LED2, OUTPUT);
      Serial.begin(9600);
  }

void loop()
{
   if    (!(digitalRead(BUTTON)))
        {
          COUNT = 0;
          Serial.println("STOP COUNT = 0");
          digitalWrite(LED1, LOW);
          digitalWrite(LED2, LOW);
          delay (2000);
        }
          Serial.println(COUNT);
}

//-----------------------------------------
void RoteStateChanged() //When CLK    FALLING READ DAT
{
```

```
 if   (digitalRead(DAT)) // When DAT = HIGH IS FORWARD
    {
     COUNT++;
     digitalWrite(LED1, HIGH);
     digitalWrite(LED2, LOW);
     delay(20);
    }
 else                          // When DAT = LOW IS BackRote
    {
     COUNT--;
     digitalWrite(LED2, HIGH);
     digitalWrite(LED1, LOW);
     delay(20);
    }
}
//////////////////////////////////////////////////////
```

Result

Rotate the encoder, you can control the on and off of the two LEDs.

****

Project 30: 5V relay

Introduction

Relay is an automatic switch element with isolation function. It's widely used in remote control, remote sensing, communication, automatic control, mechatronics and electronic devices. It is one of the most important control elements. In summary, it has below functions:

1)Expanding the control range: for example, when the control signal of the multicontact relay reaches a certain value, it can form different contact ways. At the same time switch on or off of multiple circuits.

2)Amplification: for example, sensitive relay and intermediate relay can control circuits of large power using a small amount of controlled quantity.

3)Synthetic signal: for example, when input more than one specific control signals to multiwound relay, achieve predetermined control effect after comparison and synthesis.

4)Automatic, remote control and monitoring: for example, combining relay on the automatic device to other electric equipment can realize automatic operation.

Matters needing attention

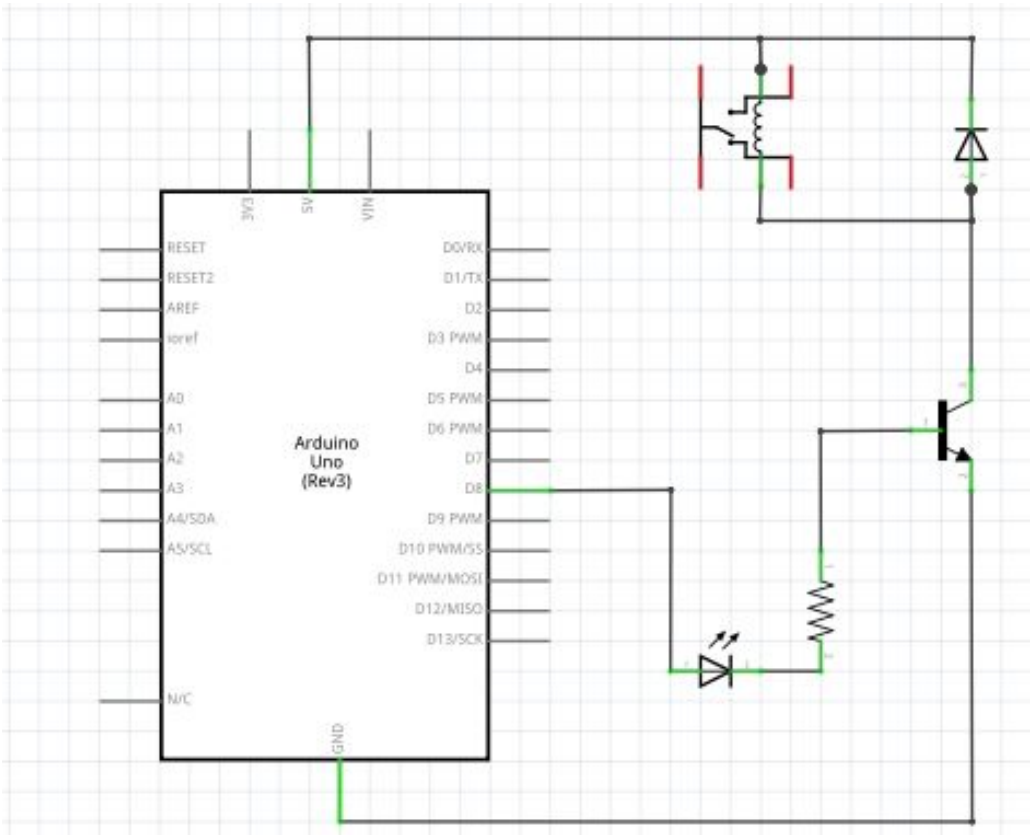
a. rated working voltage: refers to the normal working voltage the relay coil needs,

b. Voltage control is the control circuit. It can be AC or DC voltage according to different models.
c. DC resistance: refers to the DC resistance of relay coil, multimeter.
d. Attract current: refers to the minimum current that the relay needs to generate suction action. During normal use, given current must be slightly larger than the operating current to guarantee stable operation of relay. Normally, the working voltage is 1.5 times of the coil's. It cannot be more than the rated working voltage or coil will be burnt out due to high current.
e. Release current: Refers to the maximum current for the relay to gennerate release action. When the suction-state current reduces to a certain point, relay will return to its unenergized release state. The current here is far less than the suction current.
f. Contact switch voltage and current: refers to the load voltage and current the relay allows. This determines the magnitude of control voltage and current. Use current no high than these values or relay contact will be damaged.

Hardware required

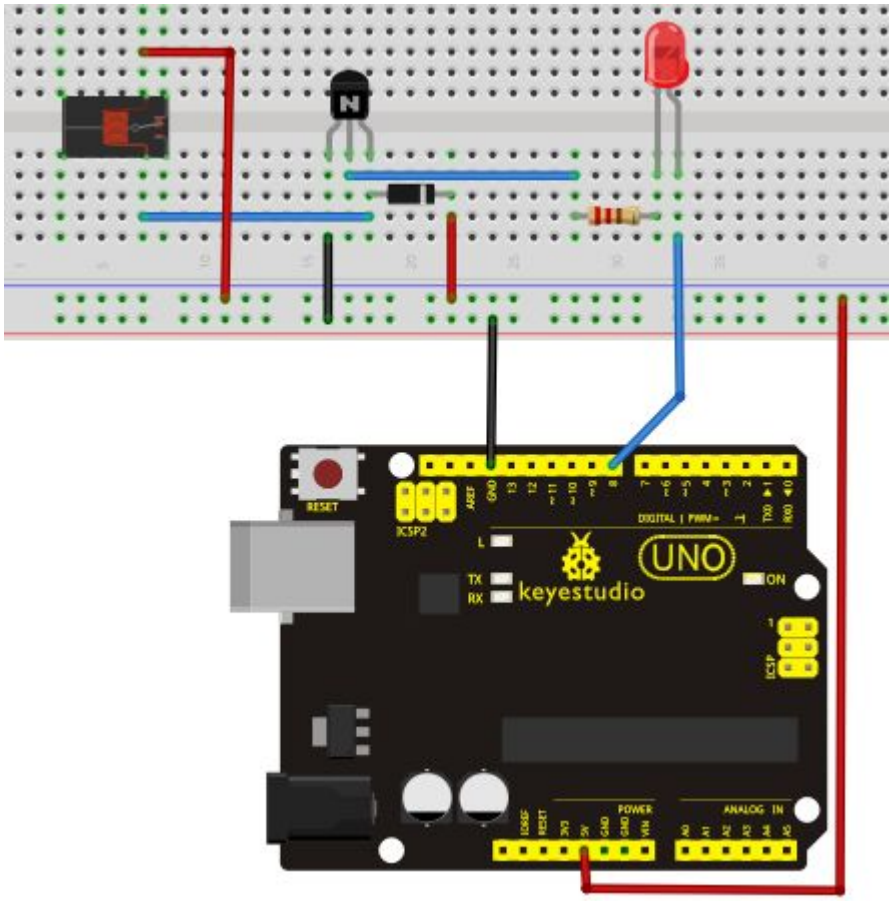
Uno board *1
Red M5 LED *1
220Ω resistor *1
8050 NPN dynatron *1
4007 diode *1
5V relay *1
Breadboard *1
USB cable *1
Jumper wires

Schematic diagram

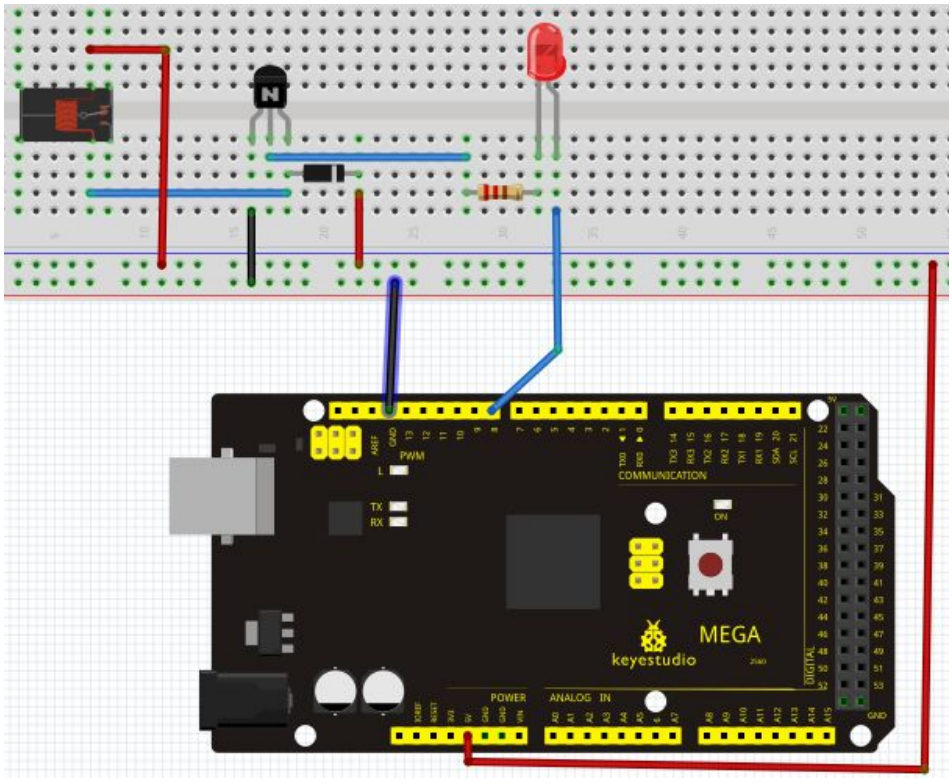


Circuit connection

Connection for R3:



Connection for 2560 R3:



Sample program

```
//////////////////////////////////////////////////////////
Int relay = 8; / / relay turn-on trigger signal - active high;
Setup (void)
{
PinMode (relay, OUTPUT); / / define port attribute for the output;
}
Loop (void)
{
DigitalWrite (relay, HIGH); / / relay conducted;
Delay (1000);
DigitalWrite (relay, LOW); / / relay switch;
Delay (1000);
}
//////////////////////////////////////////////////////////
```

Result

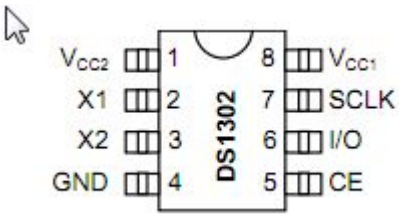
You can have different ways to do the conduction and disconnection process. This is one way for your reference. Here, when S is in high level, relay switches to the on end. LED will be turned on or you can switch to NC end. In the test result, you will see LED turning on and off in 1s interval.

****

Project 31: DS1302 clock

Introduction

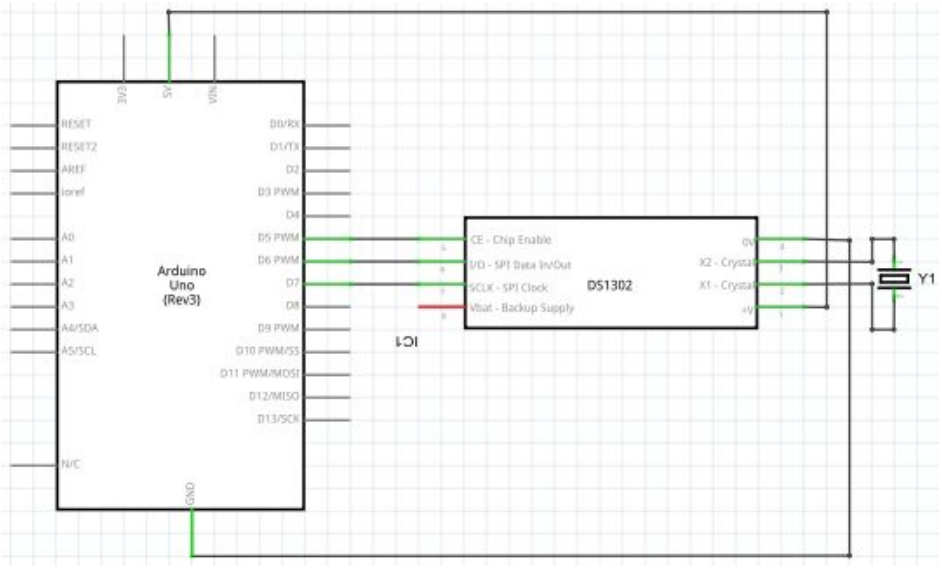
DS1302 is a clock module produced by Maxim. It supports dispaly of year, month, day, hour, minute, second and week. It supports backup battery for continuous charging.You can conviniently connect it to Arduino with three data cables.



Hardware required

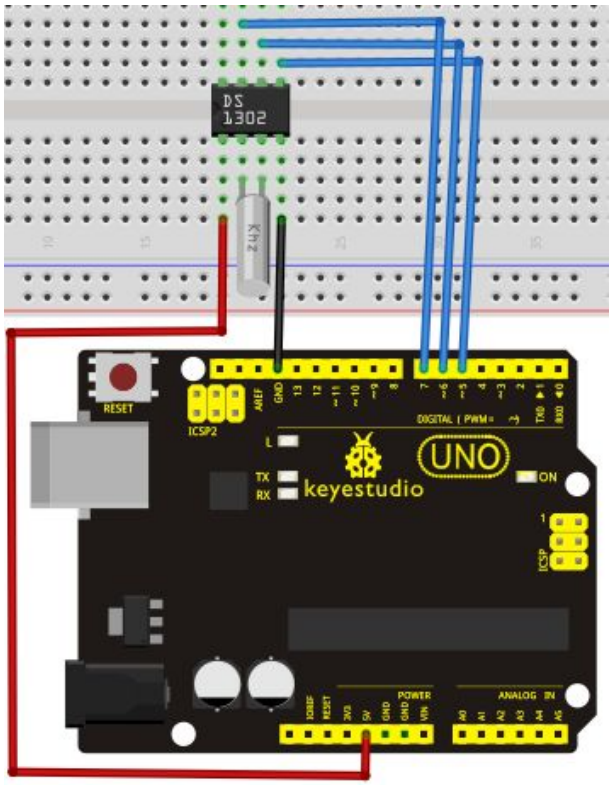
Arduino board x1
USB cable x1
DS1302*1
Crystal Oscillator*1
Breadboard*1
Breadboard jumper wires* several

Schematic diagram

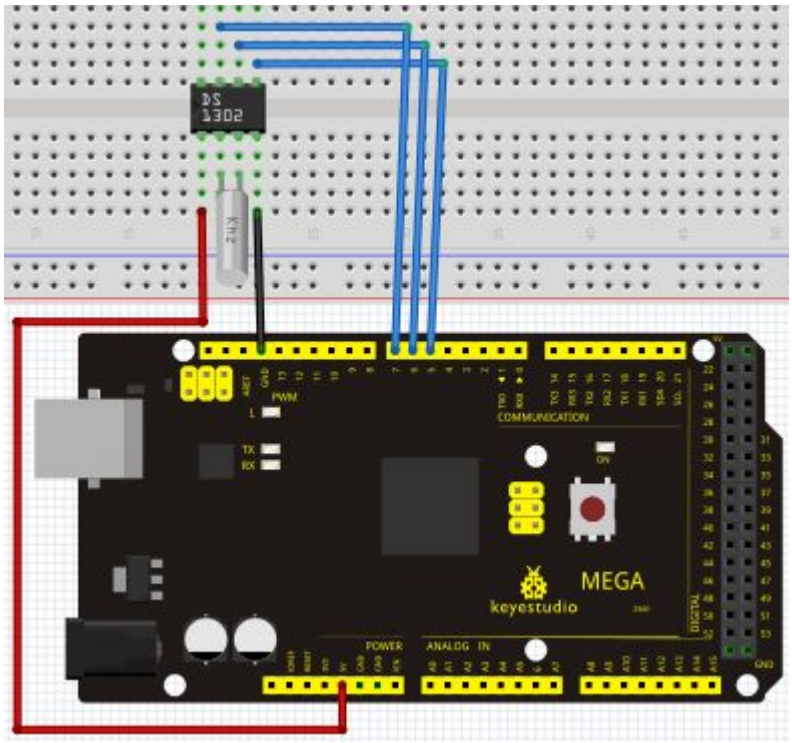


Circuit connection

Connection for R3:



Connection for 2560 R3:



Sample program

//

/*

Example sketch for interfacing with the DS1302 timekeeping chip.

http://quadpoint.org/projects/arduino-ds1302
*/

```
#include <stdio.h>
#include <string.h>
#include <DS1302.h>

/* Set the appropriate digital I/O pin connections */
uint8_t CE_PIN   = 5;
uint8_t IO_PIN   = 6;
uint8_t SCLK_PIN = 7;

/* Create buffers */
char buf[50];
char day[10];

/* Create a DS1302 object */
DS1302 rtc(CE_PIN, IO_PIN, SCLK_PIN);


void print_time()
{
  /* Get the current time and date from the chip */
  Time t = rtc.time();

  /* Name the day of the week */
  memset(day, 0, sizeof(day));    /* clear day buffer */
  switch (t.day) {
    case 1:
      strcpy(day, "Sunday");
      break;
    case 2:
      strcpy(day, "Monday");
      break;
    case 3:
      strcpy(day, "Tuesday");
      break;
    case 4:
```

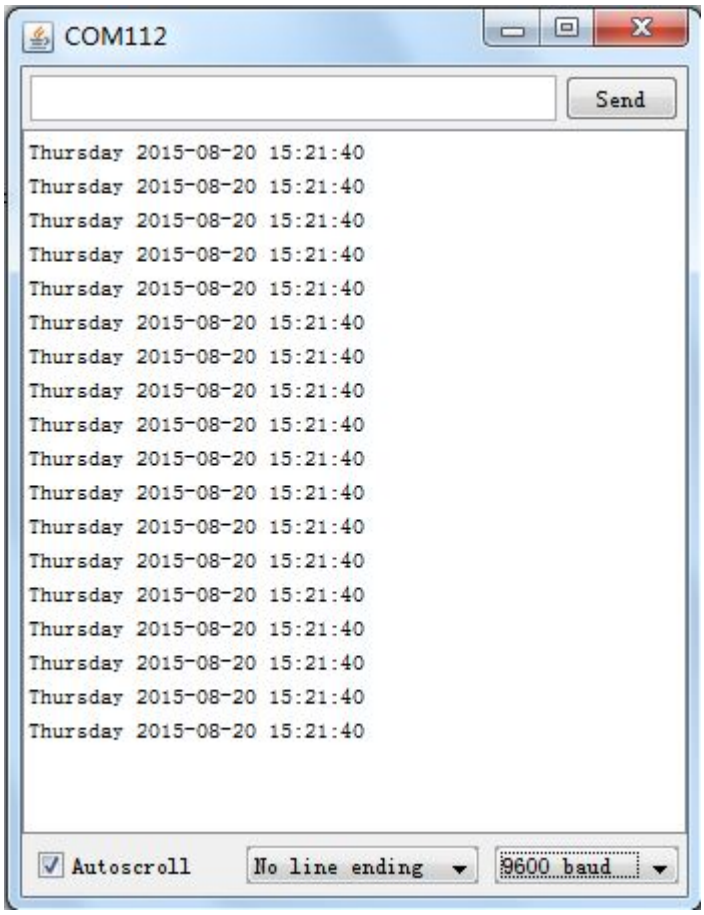
```
            strcpy(day, "Wednesday");
            break;
        case 5:
            strcpy(day, "Thursday");
            break;
        case 6:
            strcpy(day, "Friday");
            break;
        case 7:
            strcpy(day, "Saturday");
            break;
    }

    /* Format the time and date and insert into the temporary buffer */
    snprintf(buf, sizeof(buf), "%s %04d-%02d-%02d %02d:%02d:%02d",
                day,
                t.yr, t.mon, t.date,
                t.hr, t.min, t.sec);

    /* Print the formatted string to serial so we can see the time */
    Serial.println(buf);
}


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

    /* Initialize a new chip by turning off write protection and clearing the
       clock halt flag. These methods needn't always be called. See the DS1302
       datasheet for details. */
    rtc.write_protect(false);
    rtc.halt(false);

    /* Make a new time object to set the date and time */
    /*      Tuesday, May 19, 2009 at 21:16:37.                      */
    Time t(2015,8,20,15,21,40,5);

    /* Set the time and date on the chip */
    rtc.time(t);
}
```

```
/* Loop and print the time every second */
void loop()
{
   print_time();
   delay(1000);

}
```

//

Result

Open Arduino com port debugger, you can see current time. If the time needs adjustment, you only need to input current date and time in the com port, and seperate with a comma.
Format as:
Year, month, day, minute, second, week
Week nomber: Sunday=1，MOnday=2，...Saturday=7
For example, the time now is 15:21:40, Thursday, August 20th, 2015,
Just full in 2015, 8, 20, 15, 21, 40, 5



****

Project 32: Mos tube driving motor

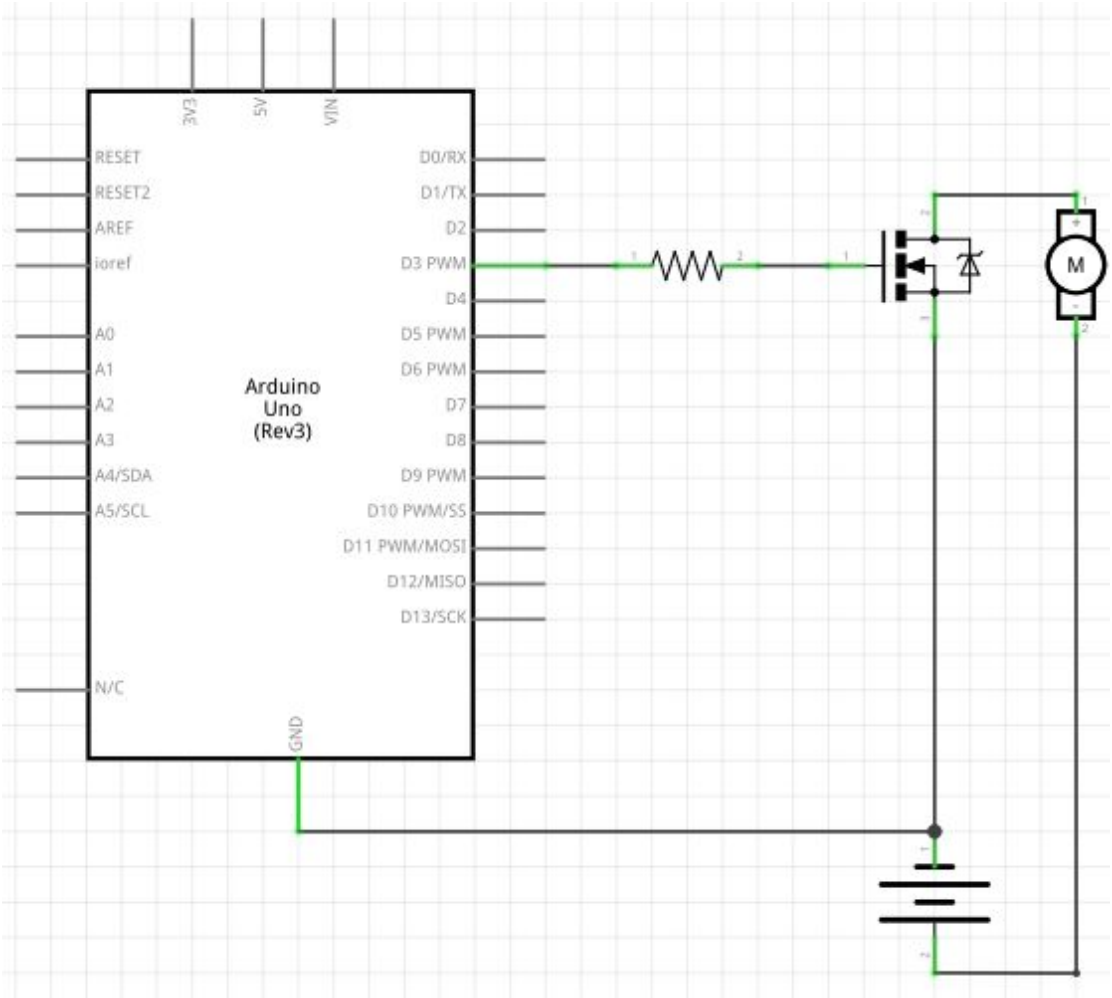
Introduction

In this experiment, we use P-type MOS tube as a switch to control the rotation of the motor. When D3 is input with high level voltage, MOS tube DS is conducted, motor begins to rotate; When D3 is input with low level voltage, MOS tube DS is disconnected, motor stops running.

Hardware required

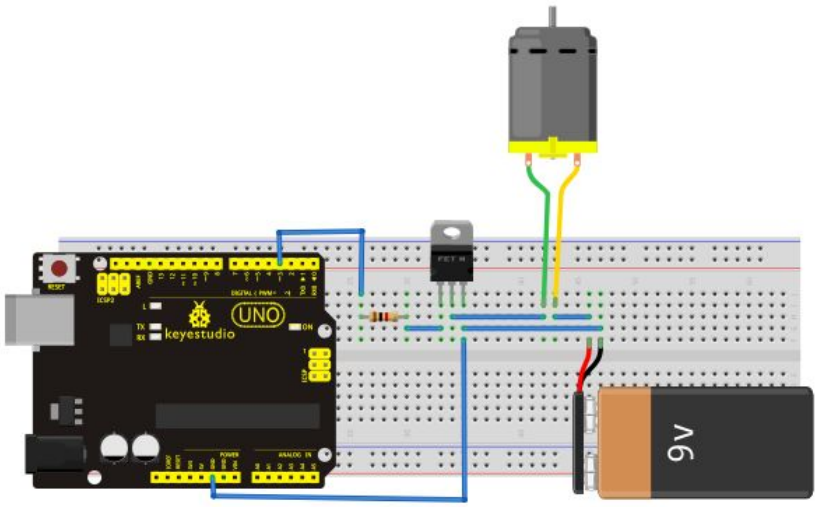
P MOS *1
9V battery *1
1 KΩresistor *1
Motor *1
Bread board *1
Bread board jumper wires *several

Schematic diagram

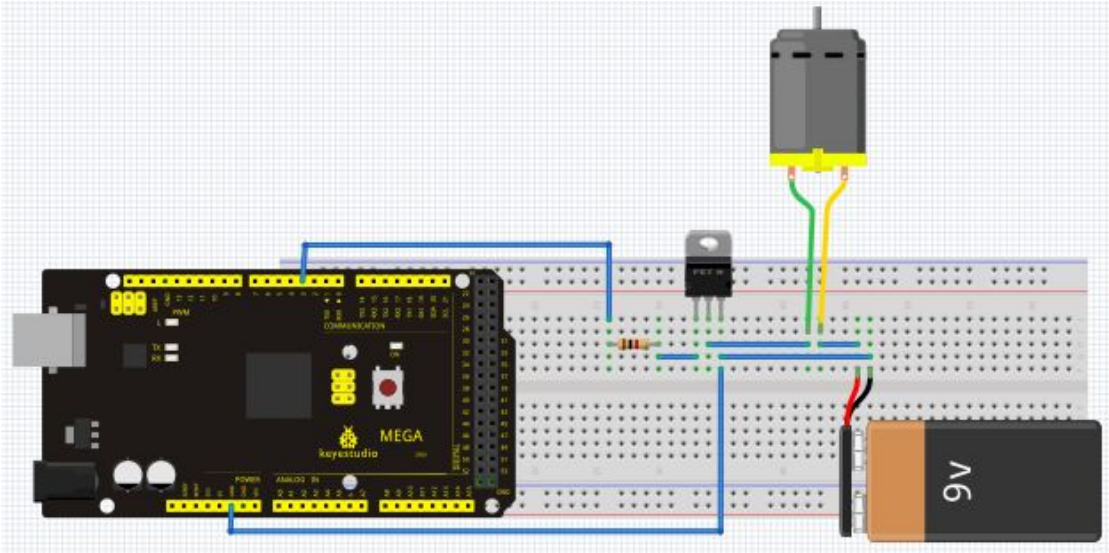


Circuit connection

Connection for R3:



Connection for 2560 R3:



Sample program

```
//////////////////////////////////////////////////////////
// the setup function runs once when you press reset or power the board
void setup() {
   // initialize digital pin 33 as an output.
   pinMode(3, OUTPUT);
}
```

```
// the loop function runs over and over again forever
void loop() {
   digitalWrite(3, HIGH);      // turn the LED on (HIGH is the voltage level)
   delay(2000);                      // wait for a second
   digitalWrite(3, LOW);       // turn the LED off by making the voltage LOW
   delay(3000);                      // wait for a second
}
////////////////////////////////////////////////////////
```

Result

Motor rotates for 2 seconds, and stops for 3 seconds, then cycles on.

Experiment Principle:

The characteristics of NMOS is that when Vgs is greater than a certain value, it will be conducted. It is suitable for use in situation when source electrode is connected to ground (low-end driving), as long as the grid voltage reaches 4V to 10V. When G electrode is connected to digital port 3, s electrode to GND and digital port 3 output is HIGH, Vgs is 5V, meeting the requirement, DS is conducted, motor rotates. When digital port output is LOW, Vgs is 0 v, not meeting the requirements, DS is disconnected, motor stops running.

In this experiment, by controlling the output of digital port 3 to control Vgs voltage and then controling the conduction and disconnection of DS electrode to control the rotation of the motor.

****

Project 33: 4N35

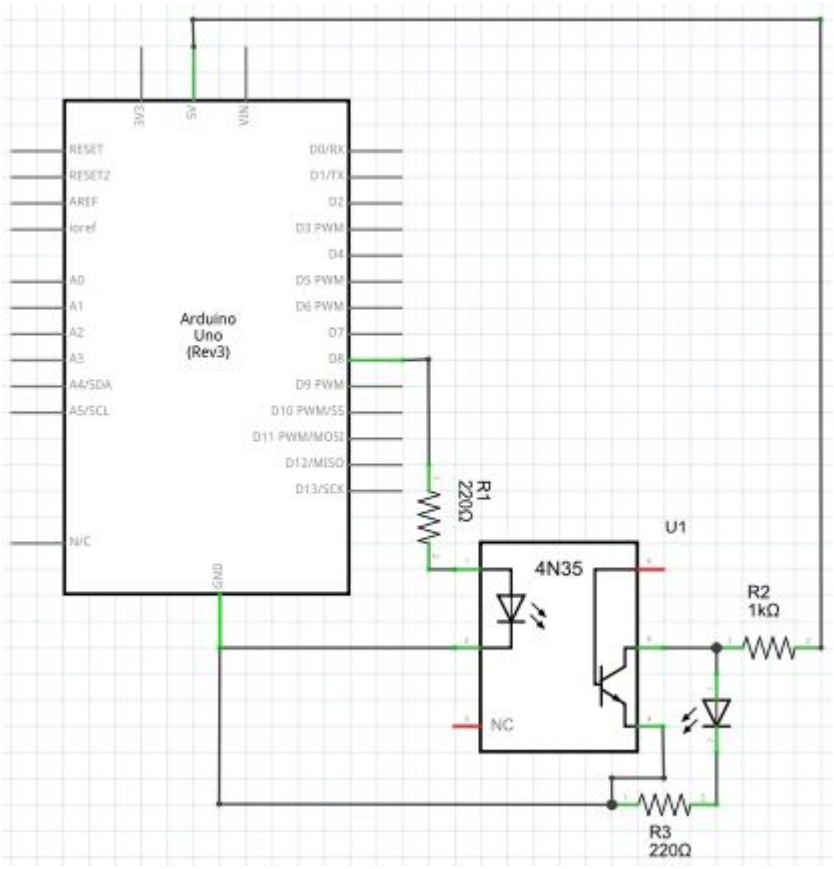
Introduction

4N35 is a general photoelectric coupler, containing a gallium arsenide infrared LED, and use the LED to drive silicon photoelectric transistor diode. The package of 4N35 is 6-pin dual-in-line Package.

Hardware required

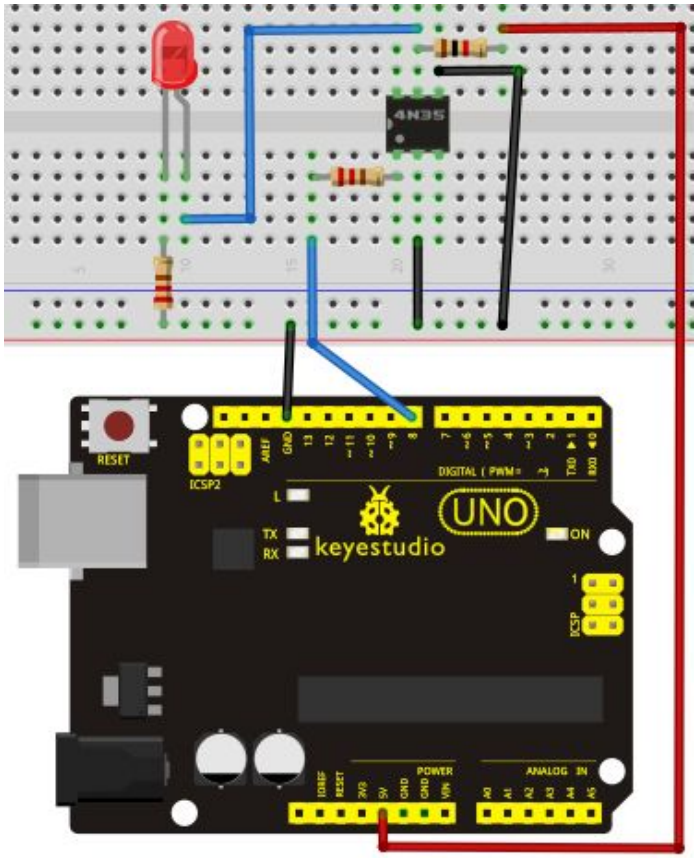
4n35 *1
1KΩ resistor *1
220Ω resistor *2
Red M5 LED *1
Breadboard *1
Breadboard jumper wires *several

Schematic diagram

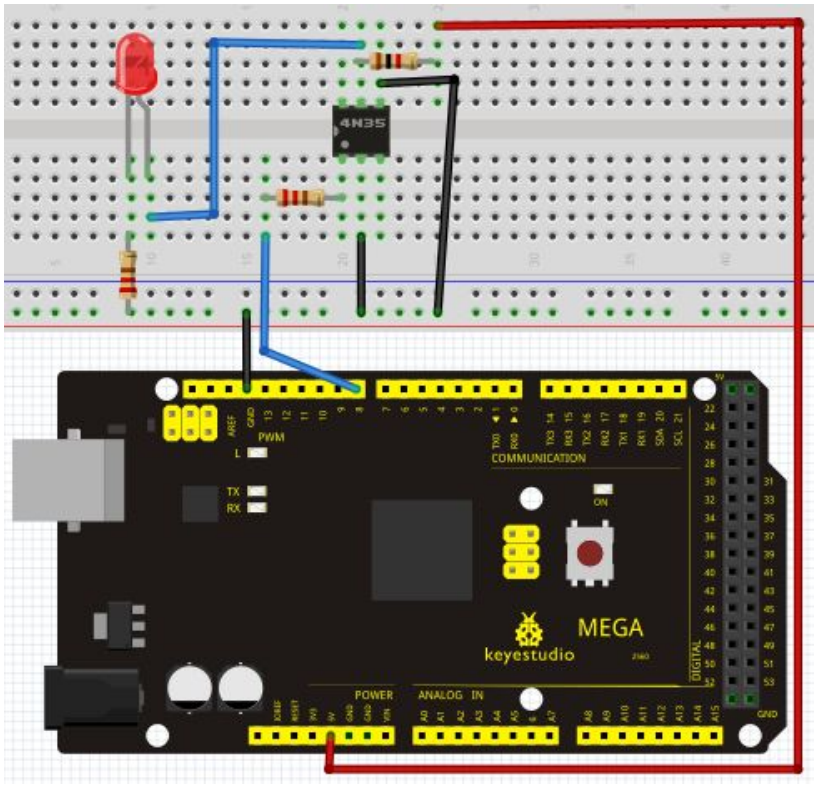


Circuit connection

Connection for R3:



Connection for 2560 R3:



Sample program

```
//////////////////////////////////////////////////////////

int Optocoupler=8;

void setup()

{

  pinMode(Optocoupler,OUTPUT);

}


void loop()

{

   digitalWrite(Optocoupler,LOW);

   delay(1000);

   digitalWrite(Optocoupler,HIGH);

   delay(1000);

}

//////////////////////////////////////////////////////////
```

Result

Red light blinks, on for 1 second and off for 1 second.

****

Project 34: NE555 Timer

Introduction

If you ask anyone in the know to rank the most commonly and widely applied integrated circuits, the famous 555 time base integrated circuit would certainly be at the top of the list. The 555 – a mixed circuit composed of analog and digital circuits – integrates analogue and logical functions into an independent integrated circuit, and hence tremendously expands the application range of analog integrated circuits. The 555 is widely used in various timers, pulse generators, and oscillators. In this experiment, we will use the Uno board to test the frequencies of square waves generated by the 555 oscillating circuit and show them on a serial monitor.

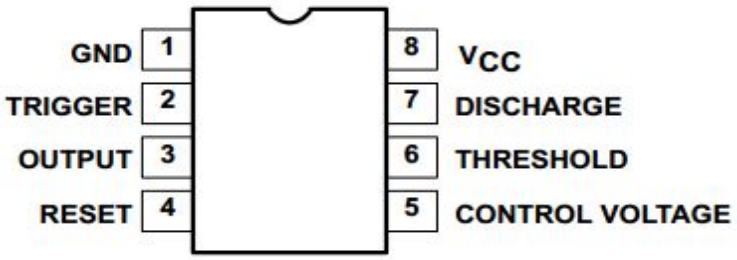
Hardware required

Uno board *1
USB cable *1
Breadboard *1
NE555 *1
104 ceramic capacitor *2
Potentiometer (100KΩ) *1
Resistor (10KΩ) *1
Jumper wires

Circuit connection

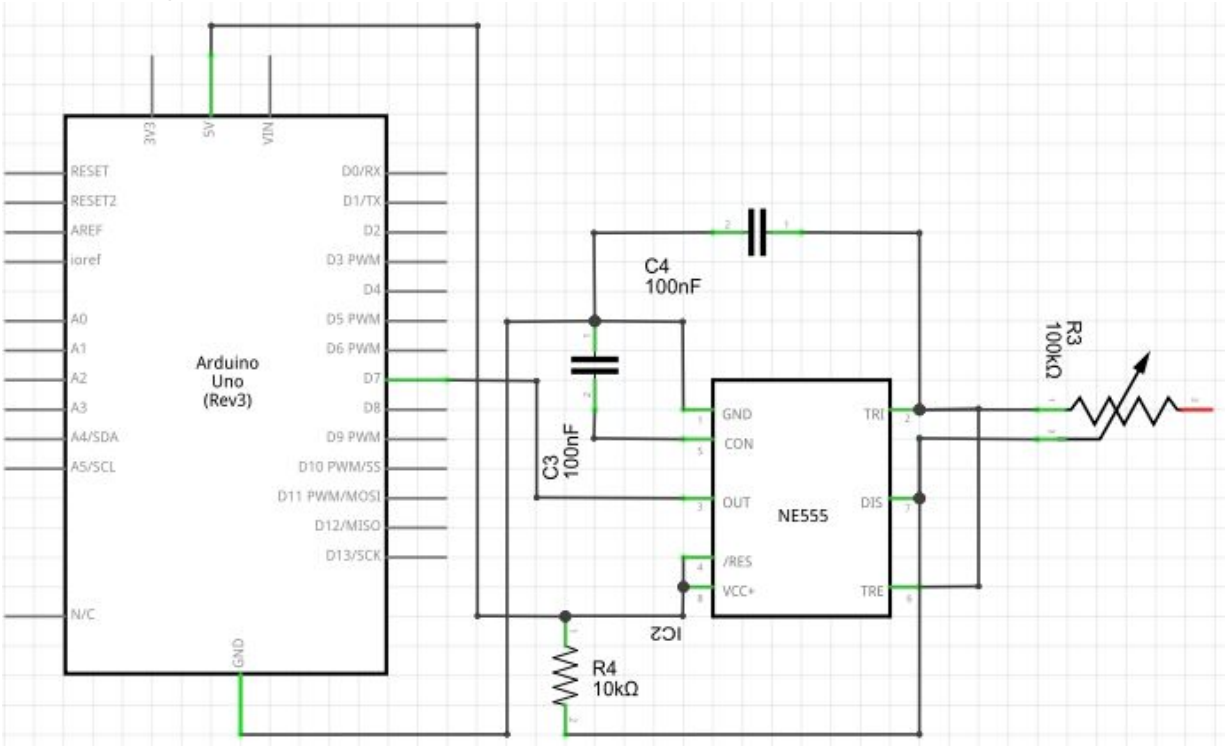
The 555 integrated circuit was originally used as a timer, hence the name 555 time base circuit. It is now widely used in various electronic products because of its reliability, convenience, and low price. The 555 is a complex hybrid circuit with dozens of components such as a divider, comparator, basic R-S trigger, discharge tube, and buffer.

555 chip pins are introduced as follows:

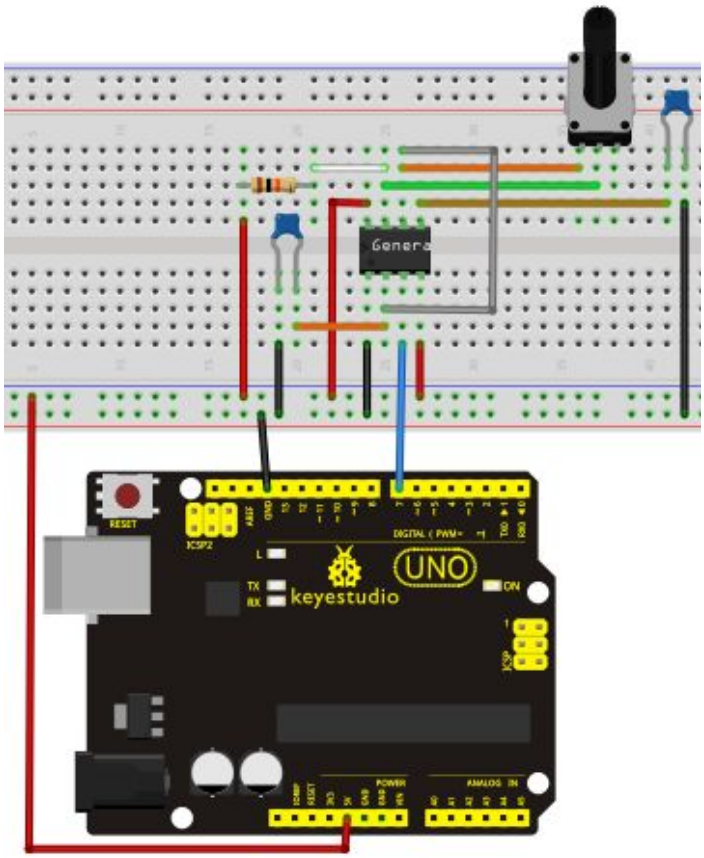


As shown in the picture, the 555 integrated circuit is dual in-line with the 8-pin package. Thus:
Pin 1 (GND): the ground;
Pin 2 (TRIGGER): the input of lower comparator;
Pin 3 (OUTPUT): having two states of 0 and 1 decided by the input electrical level;
Pin 4 (RESET): output low level when supplied a low voltage level;
Pin 5 (CONTROL VOLTAGE): changing the upper and lower level trigger values;
Pin 6 (THRESHOLD): the input of upper comparator;
Pin 7 (DISCHARGE): having two states of suspension and ground connection also decided by input, and the output of the internal discharge tube;
Pin 8 (VCC): the power supply;

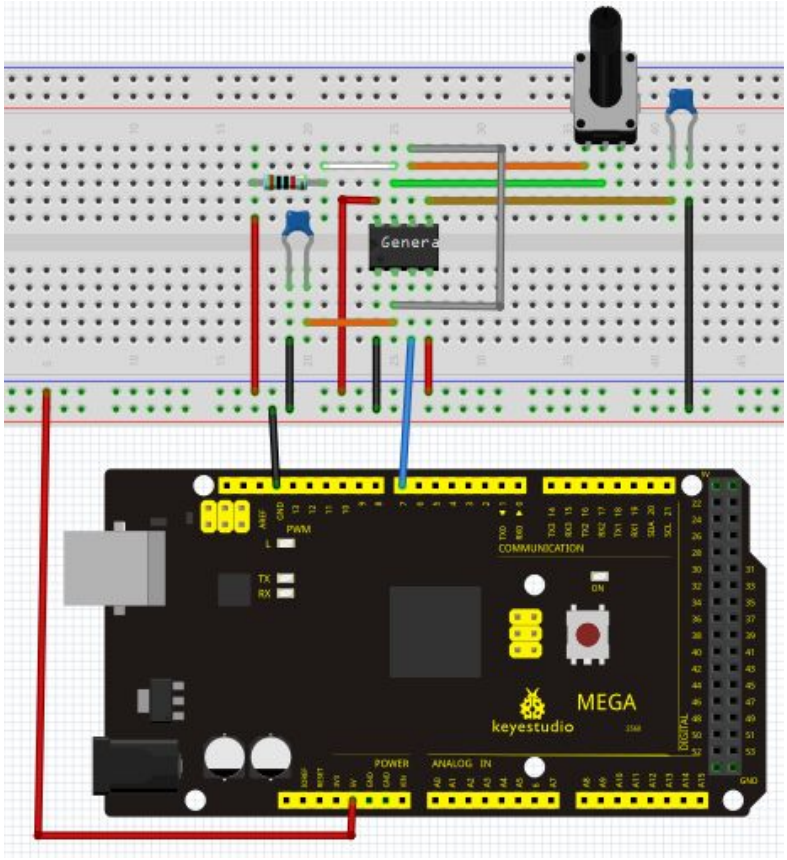
Schematic diagram



Connection for R3:



Connection for 2560 R3:



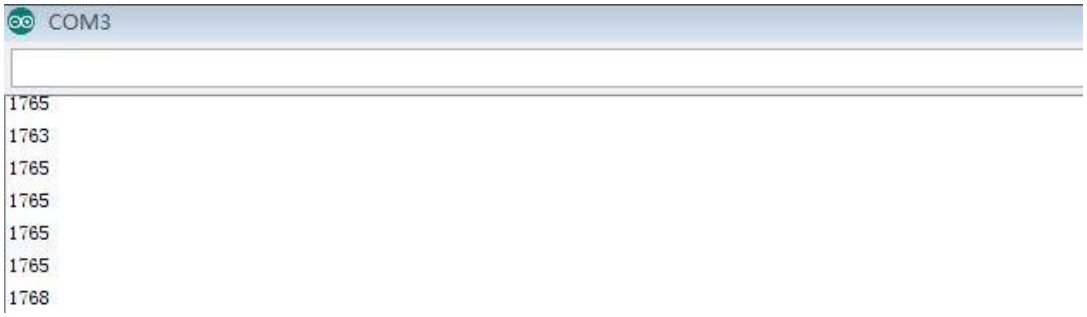
Sample program

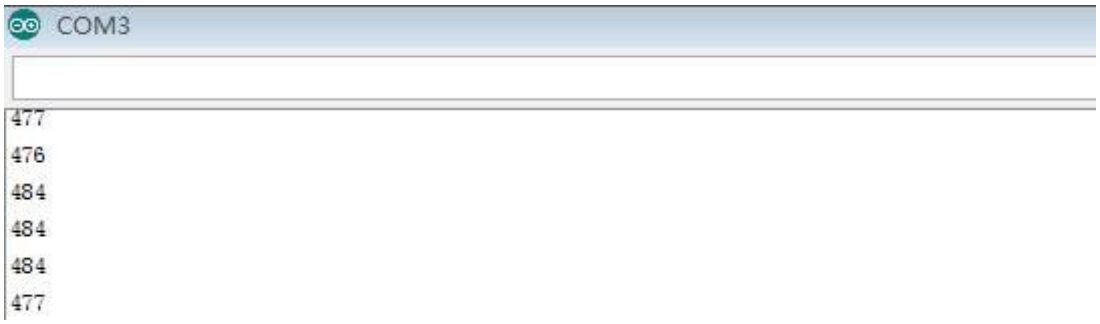
```
  int pin = 7;    //attach to the third pin of NE555
unsigned long duration;    //the variable to store the length of the pulse

void setup()
{ pinMode(pin, INPUT);    //set the pin as an input
   Serial.begin(9600);    // start serial port at 9600 bps:
}

void loop()
{
   duration = pulseIn(pin, HIGH);    //Reads a pulse on pin
   Serial.print(duration);    //print the length of the pulse on the serial monitor
   Serial.println();    //print an blank on serial monitor
   delay(500);    //wait for 500 microseconds
}
```

Result

Burn the program into Uno board

After burning the program, open the serial monitor and you will see the picture shown below. If you rotate the potentiometer, the length of the pulse (in microsecond) displayed will change accordingly.





****