Adeept

Adeept Primary Starter Kit for Arduino UNO R3

Count

About Arduino

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

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software, too, is open-source, and it is growing through the contributions of users worldwide.

About Processing

What is processing?

Processing is a programming language, development environment, and online community. Since 2001, Processing has promoted software literacy within the visual arts and visual literacy within technology. Initially created to serve as a software sketchbook and to teach computer programming fundamentals within a visual context, Processing evolved into a development tool for professionals. Today, there are tens of thousands of students, artists, designers, researchers, and hobbyists who use Processing for learning, prototyping, and production.

» Free to download and open source
» Interactive programs with 2D, 3D or PDF output
» OpenGL integration for accelerated 3D
» For GNU/Linux, Mac OS X, and Windows
» Over 100 libraries extend the core software

PROCESSING SOFTWARE

Download Processing:
https://www.processing.org/download/
For more detailed information about Processing IDE, please refer to the following link:
https://www.processing.org/reference/environment/

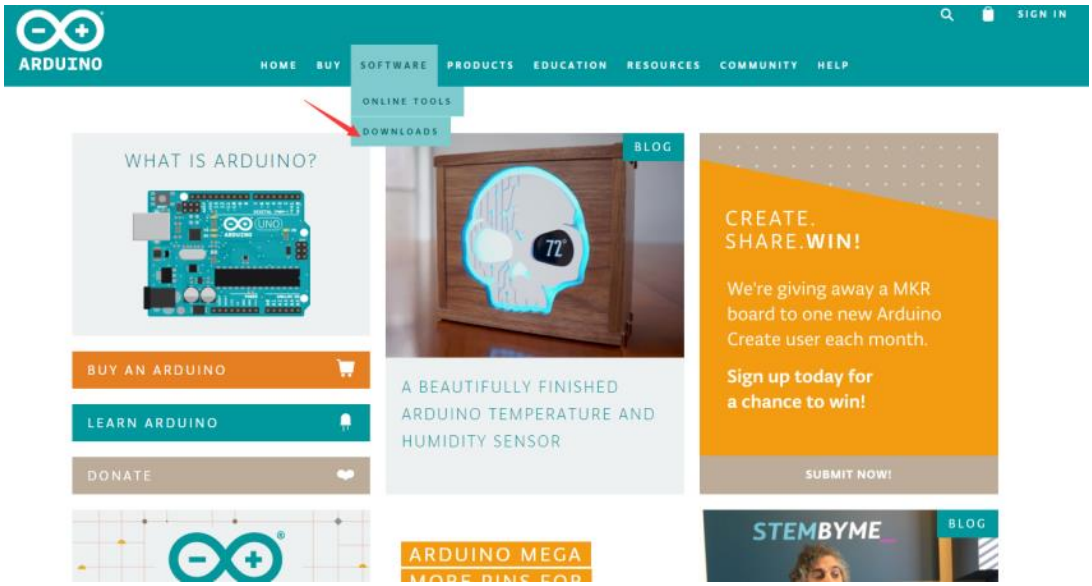
If you have any question when you use it, you can post on our official forum, we will solve it immediately. the forum website is :
http://www.adeept.com/forum

Download

Arduino official website： www.arduino.cc

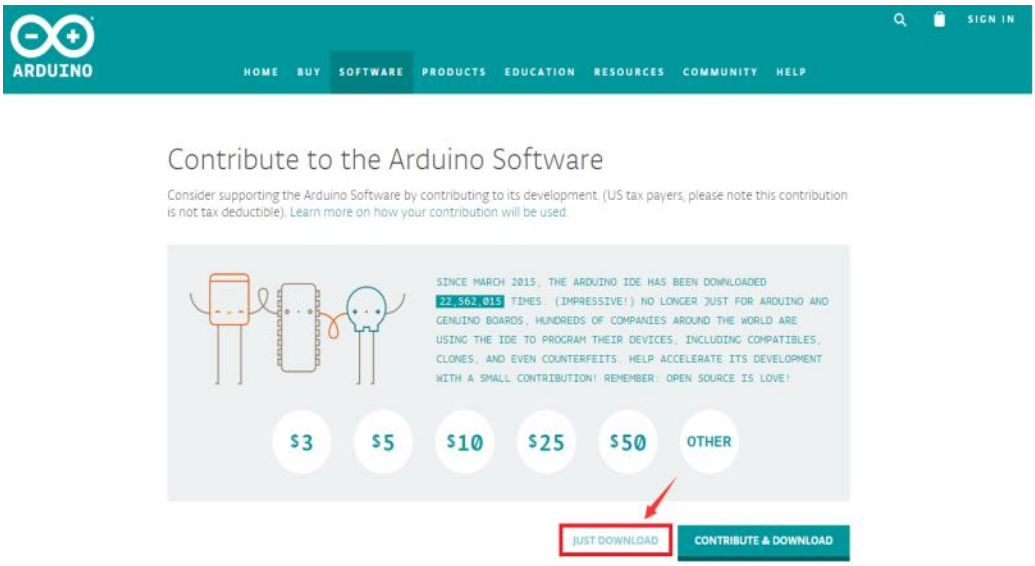
Download steps:

1. Enter Arduino official website and find “DOWNLOADS” and click it



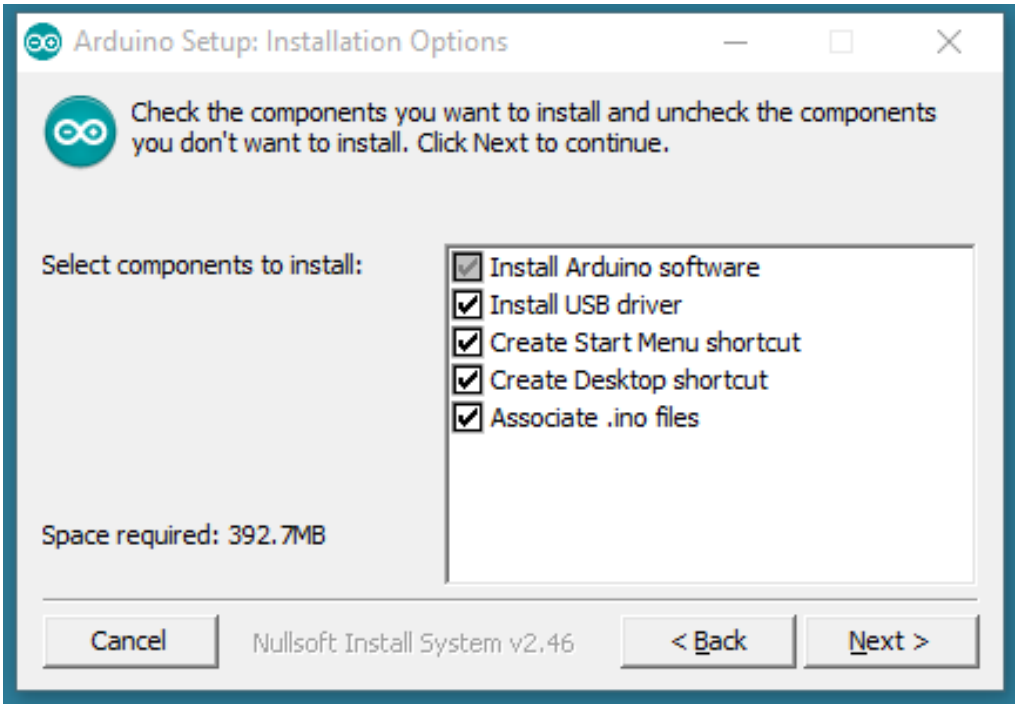
2. According to your need, you can download any kind of system of arduino IDE

3. Enter download page and click “JUST DOWNLOAD”

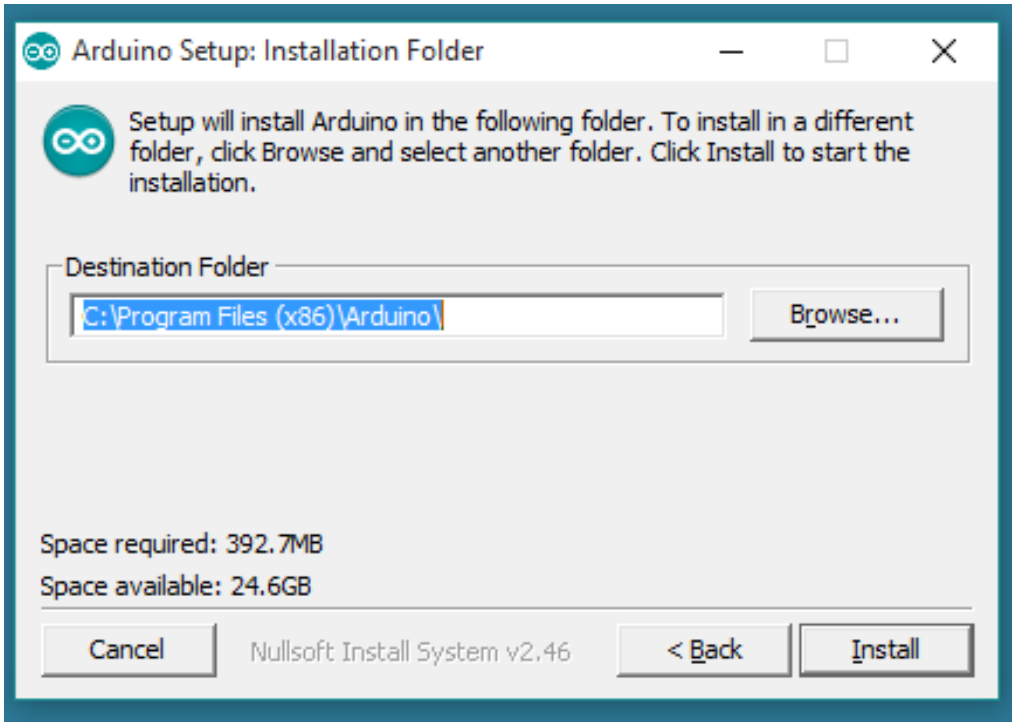


Arduino installation steps:

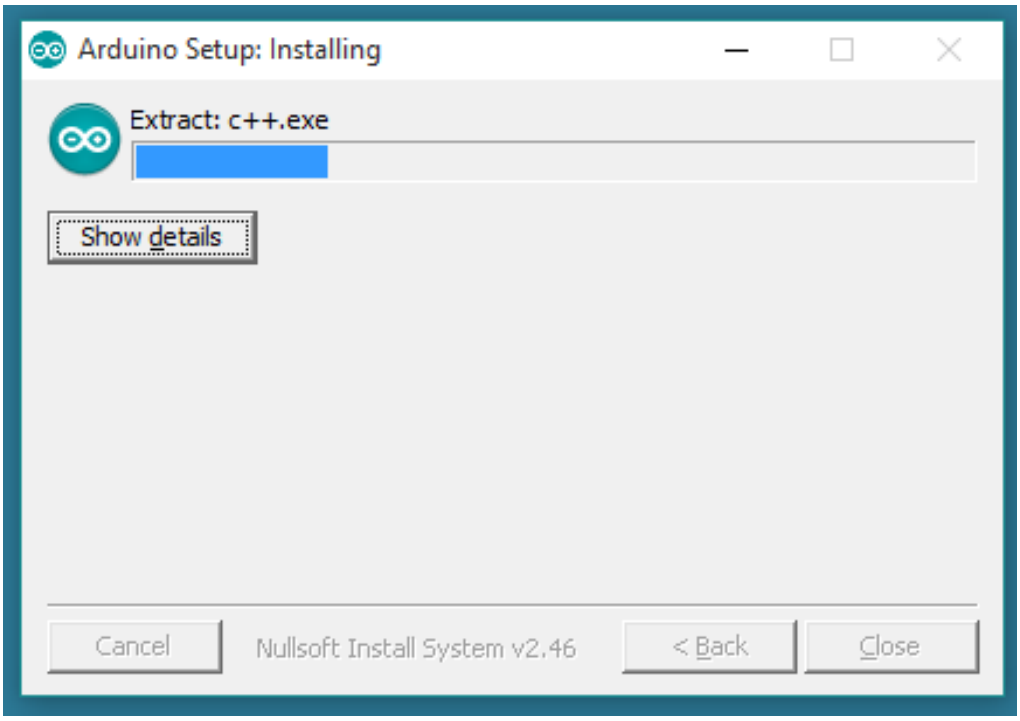
1. When the download finishes, proceed with the installation and please allow the driver installation process when you get a warning from the operating system.



2. Choose the components to install



3. Choose the installation directory (we suggest to keep the default one)



4. The process will extract and install all the required files to execute properly the Arduino Software (IDE)

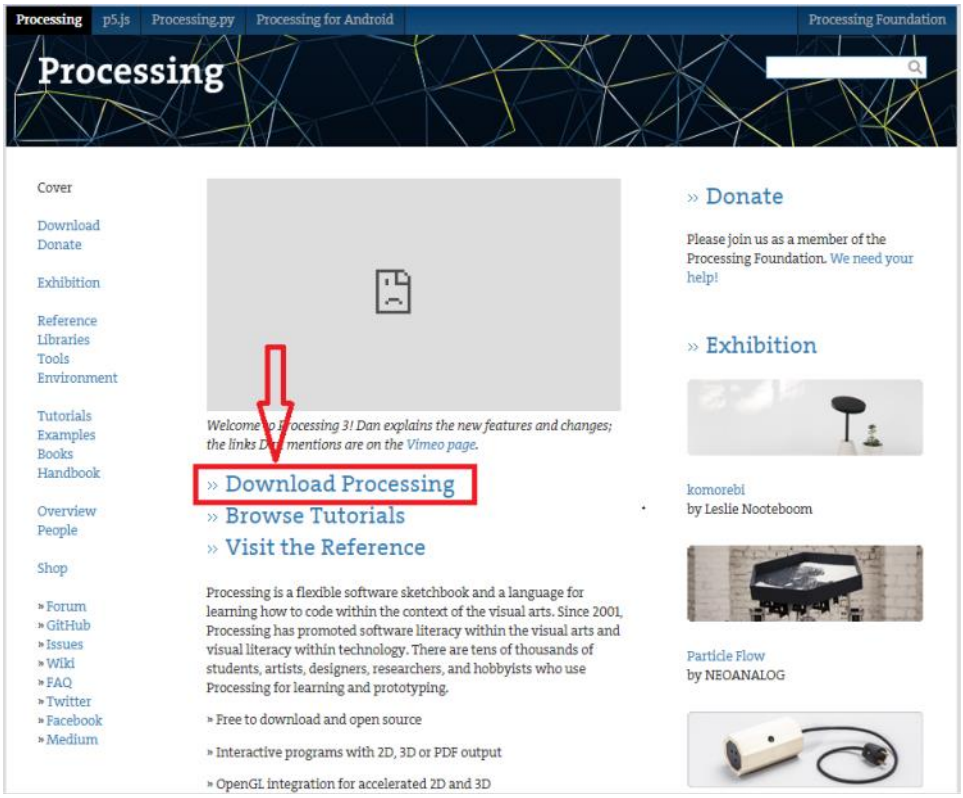
Arduino IDE installation video tutorial address:

https://youtu.be/BsTDVB8B240

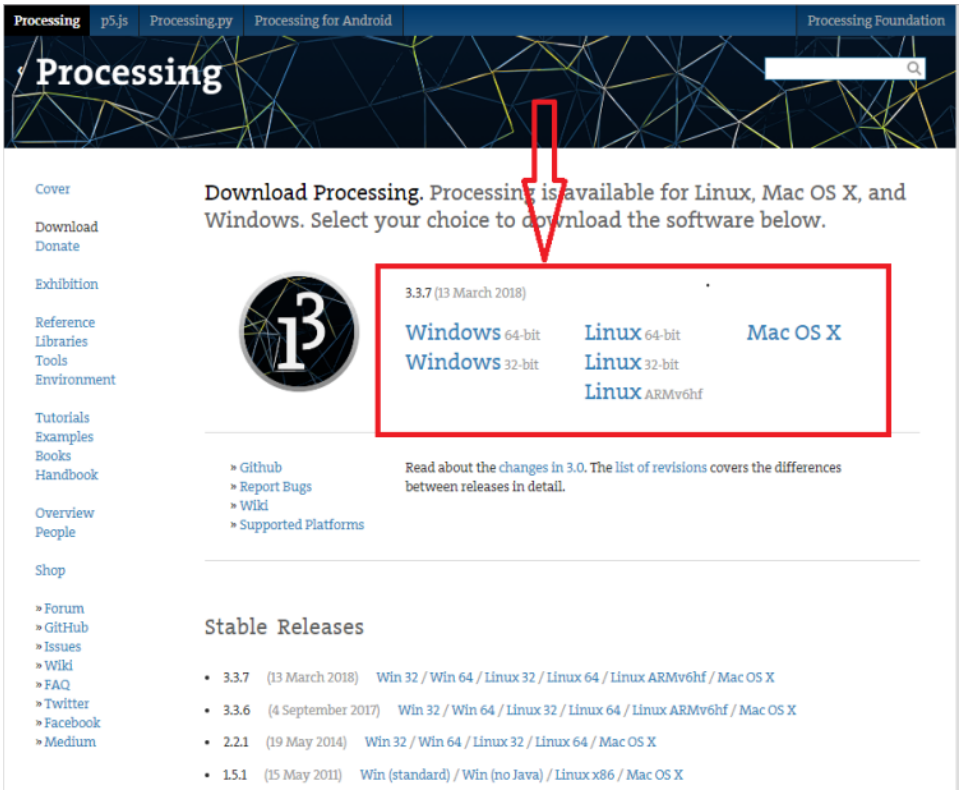
Processing

Processing official website： https://processing.org

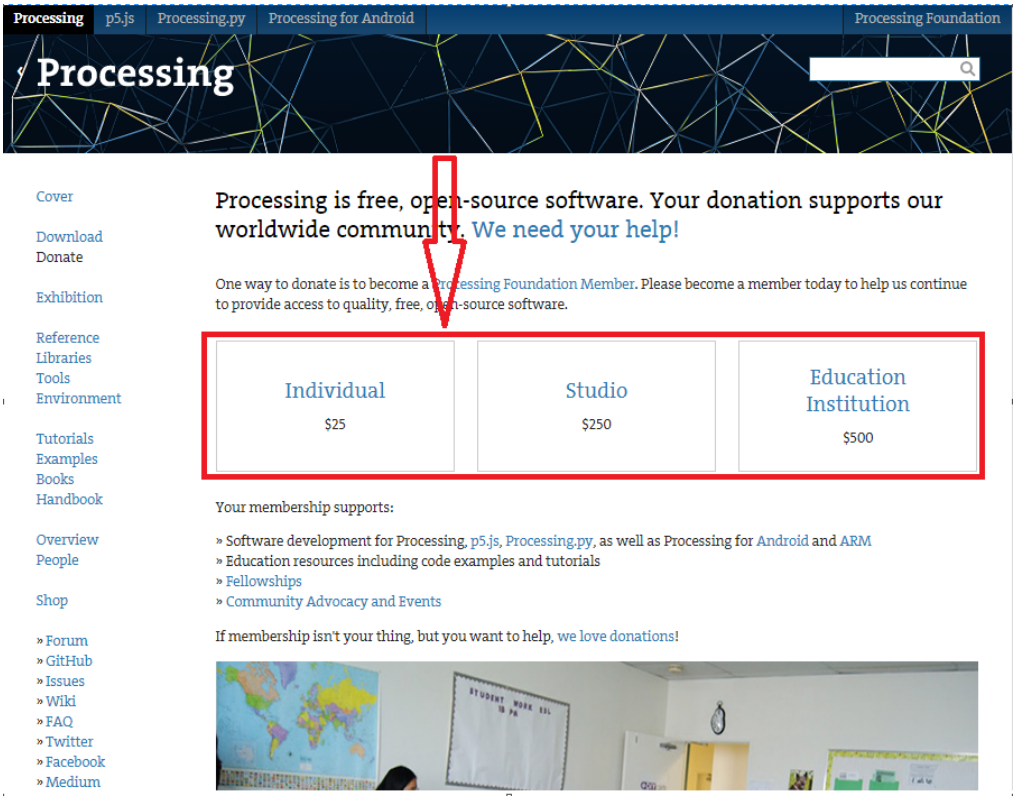
1. Enter Processing official website and find “Download Processing”



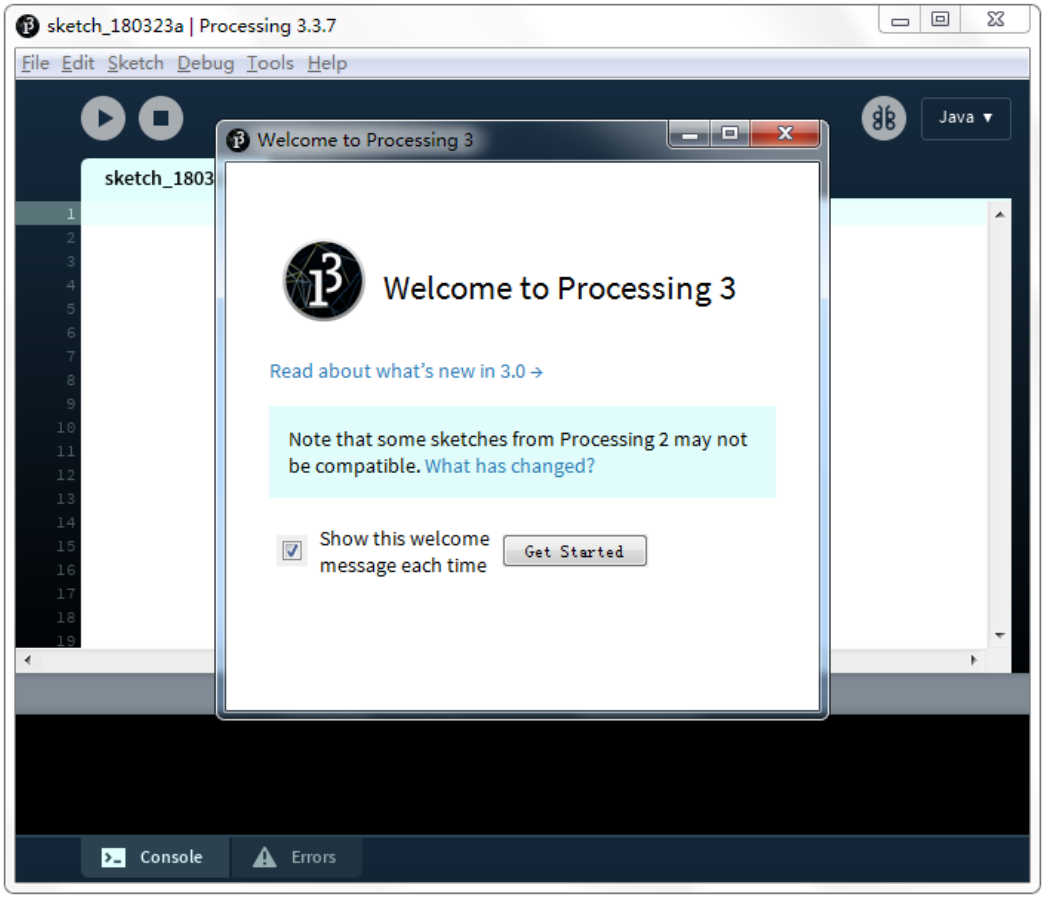
2. Click " Download Processing" and then you can download the version you want according to your computer system



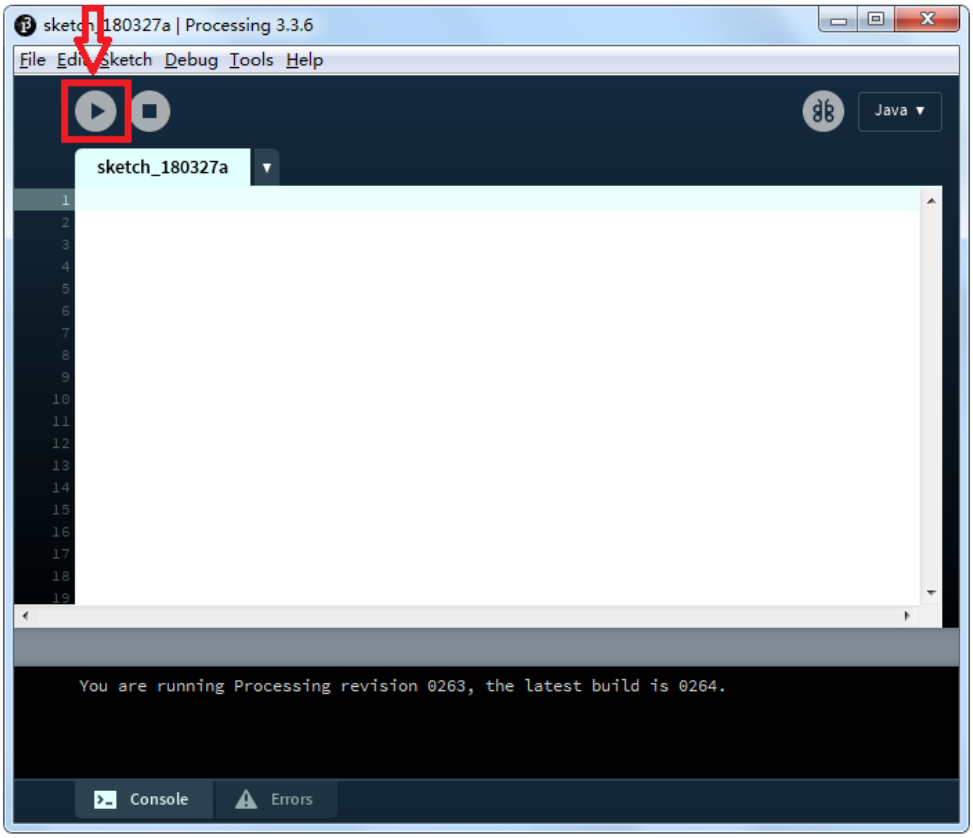
3. Enter the page and download what you need



4. After downloading, you can open the compressed files directly or you can decompress the files to where you want to keep it, and then open processing.exe and use it directly.



5. After coding, click the execution button below, and then you can see the result.



6. In this way, we finished the downloading and use of processing.

Chapter 1 LED

Project 1.1 Blinking LED

Experimental objective:

In this experiment, we will learn how to light and extinguish the LED lamp.

Required materials:

- 1* Arduino UNO
- 1* USB Cable
- 1* 220Ω Resistor
- 1* LED
- 1* Breadboard
- 2* Jumper Wires

Operating principle:

In this lesson, we will program the Arduino's GPIO output high(+5V) and low level(0V), and then make the LED which is connected to the Arduino's GPIO flicker with a certain frequency.

1. What is the LED?

The LED is the abbreviation of light emitting diode. It is usually made of gallium arsenide, gallium phosphide semiconductor materials. The LED has two electrodes, a positive electrode and a negative electrode, it will light only when a forward current passes, and it can be red, blue, green or yellow light, etc. The color of light depends on the materials it was made of.

In general, the drive current for LED is 5-20mA. Therefore, in reality it usually needs an extra resistor for current limitation so as to protect the LED.

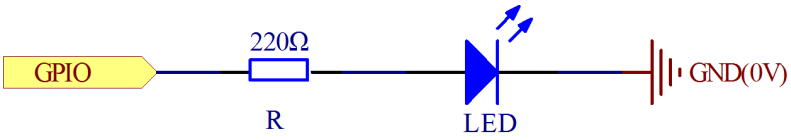
2. What is the resistor?

The main function of the resistor is to limit current. In the circuit, the character 'R' represents resistor, and the unit of resistor is ohm(Ω).

The band resistor is used in this experiment. A band resistor is one whose surface is coated with some particular color through which the resistance can be identified directly.

There are two methods for connecting LED to Arduino's GPIO:

①



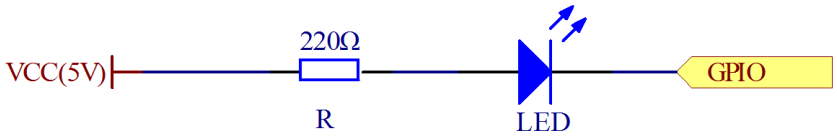
As shown in the schematic diagram above, the anode of LED is connected to Arduino's GPIO via a resistor, and the cathode of LED is connected to the ground(GND). When the GPIO output high level, the LED is on; when the GPIO output low level, the LED is off.

The size of the current-limiting resistor is calculated as follows: 5~20mA current is required to make an LED on, and the output voltage of the Arduino UNO's GPIO is 5V, so we can get the resistance :

R = U / I = 5V / (5~20mA) = 250Ω~1KΩ

Since the LED has a certain resistance, thus we choose a 220ohm resistor.

②



As shown in the schematic diagram above, the anode of LED is connected to VCC(+5V), and the cathode of LED is connected to the Arduino's GPIO. When the GPIO output low level, the LED is on; when the GPIO output high level, the LED is off.

The experiment is based on method ①, we select Arduino's D8 pin to control the LED. When the Arduino's D8 pin is programmed to output high level, then the LED will be on, next delay for the amount of time, and then programmed

the D8 pin to low level to make the LED off. Continue to perform the above process, and then you can get a blinking LED.

Required functions:

- setup()

The setup () function is called when a sketch starts. Use it to initialize variables, pin modes, start using libraries, etc. The setup function will only run once, after each powerup or reset of the Arduino board.

- loop()

After creating a setup () function, which initializes and sets the initial values, the loop () function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.

- pinMode()

It configures the specified pin to behave either as an input or an output.

As of Arduino 1.0.1, it is possible to enable the internal pullup resistors with the mode INPUT_PULLUP. Additionally, the INPUT mode explicitly disables the internal pullups.

- digitalWrite()

It writes a HIGH or a LOW value to a digital pin.

If the pin has been configured as an OUTPUT with pinMode (), its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW.

If the pin is configured as an INPUT, digitalWrite () will enable (HIGH) or disable (LOW) the internal pullup on the input pin. It is recommended to set the pinMode () to INPUT_PULLUP to enable the internal pull-up resistor.

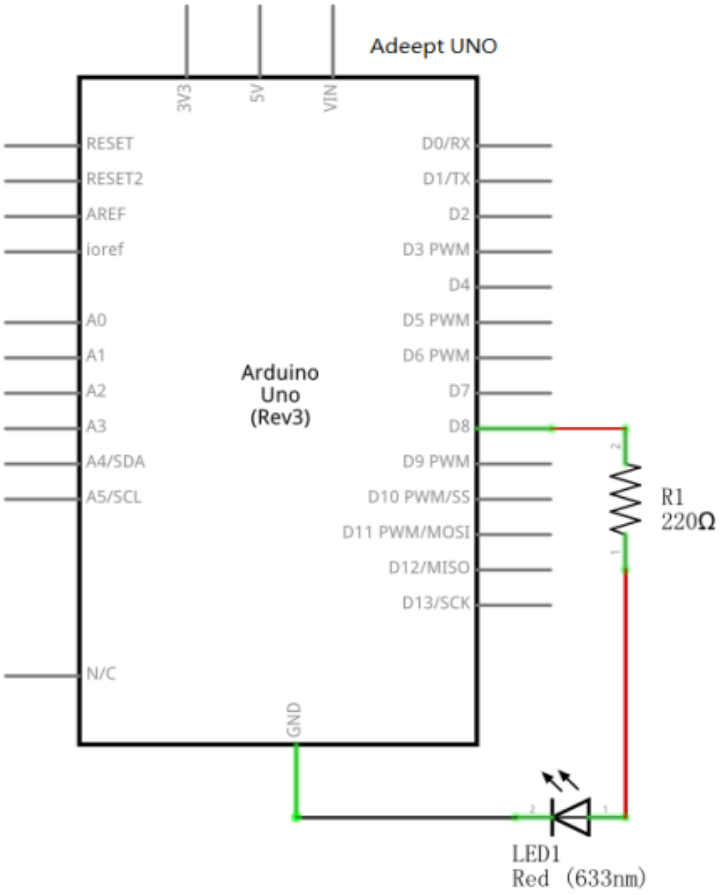
- delay()

Pauses the program for the amount of time (in miliseconds) specified as parameter. (There are 1000 milliseconds in a second.)

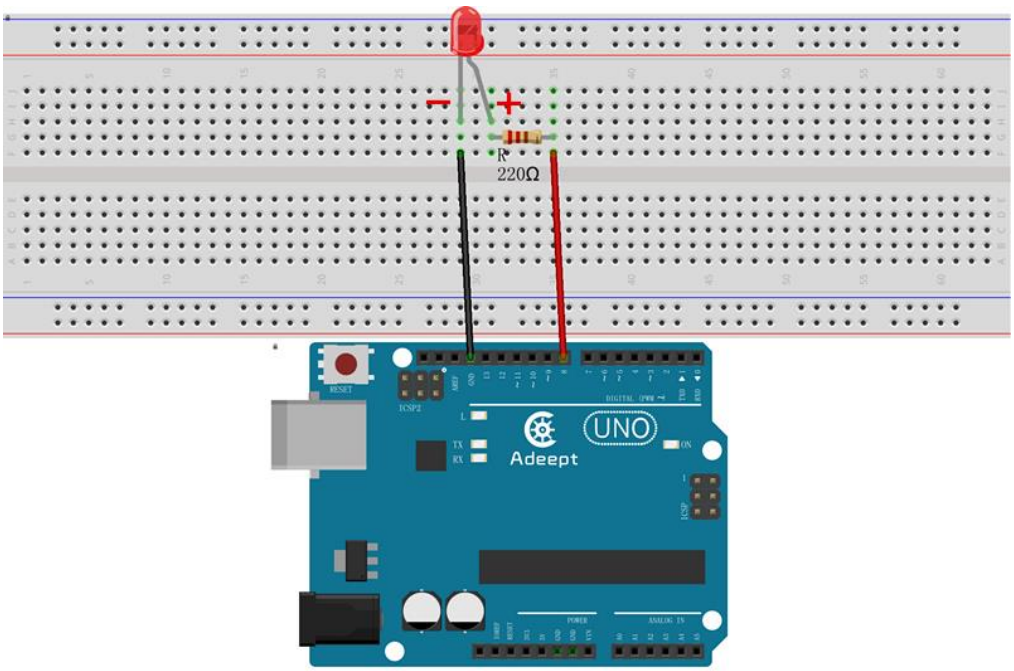
Operating steps:

1. When the power is off, we integrate all the required materials according to the schematic diagram or connection diagram.

Schematic diagram

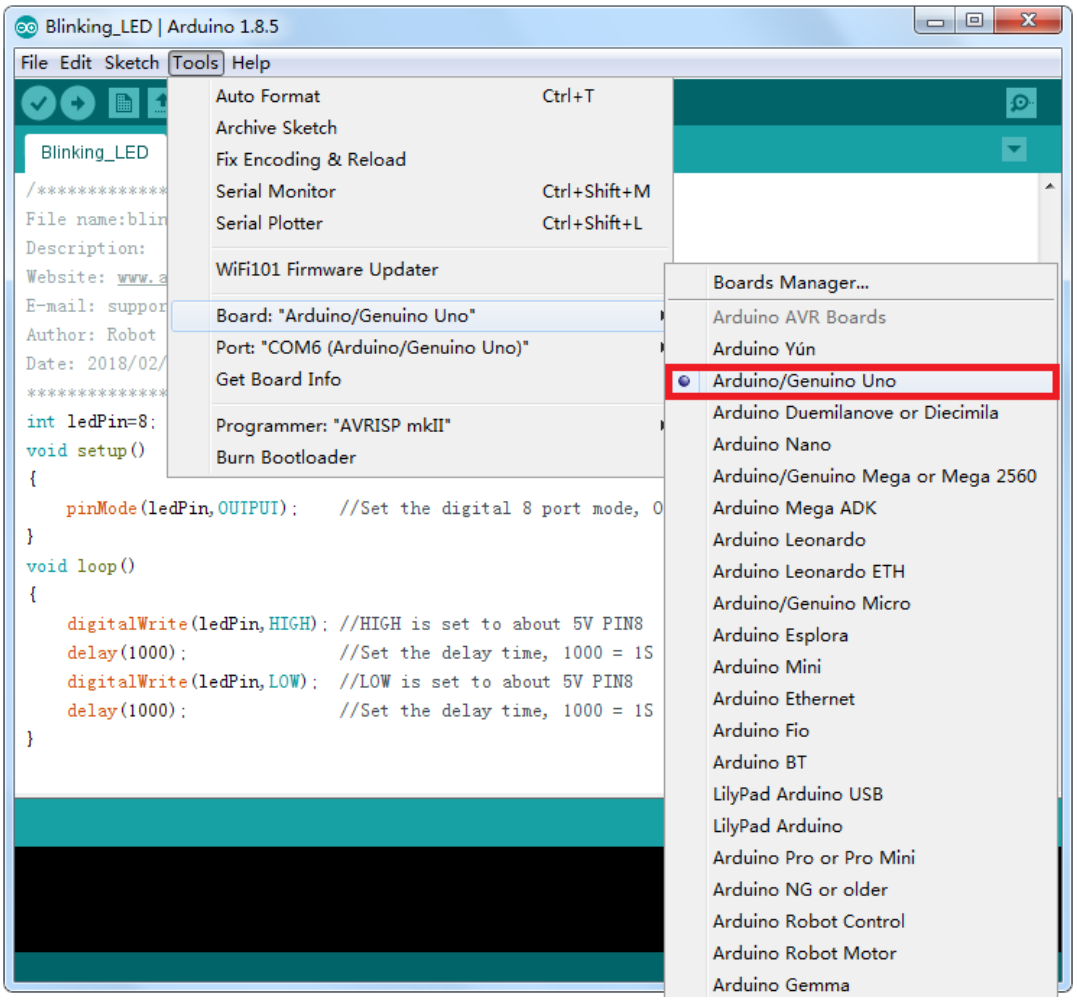


connection diagram

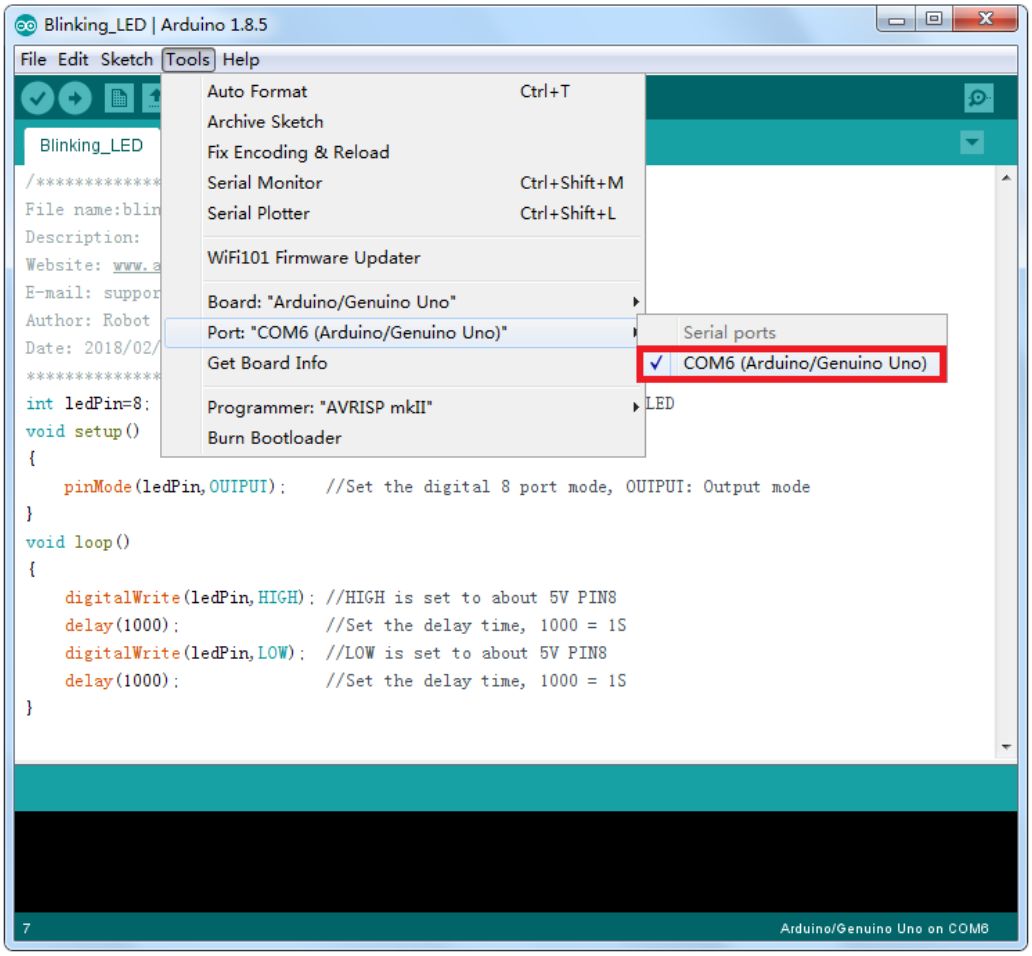


2. Examine the circuit connection carefully in step 1, and connect the arduino UNO to the computer.

3. Start arduino IDE and choose the development board type.



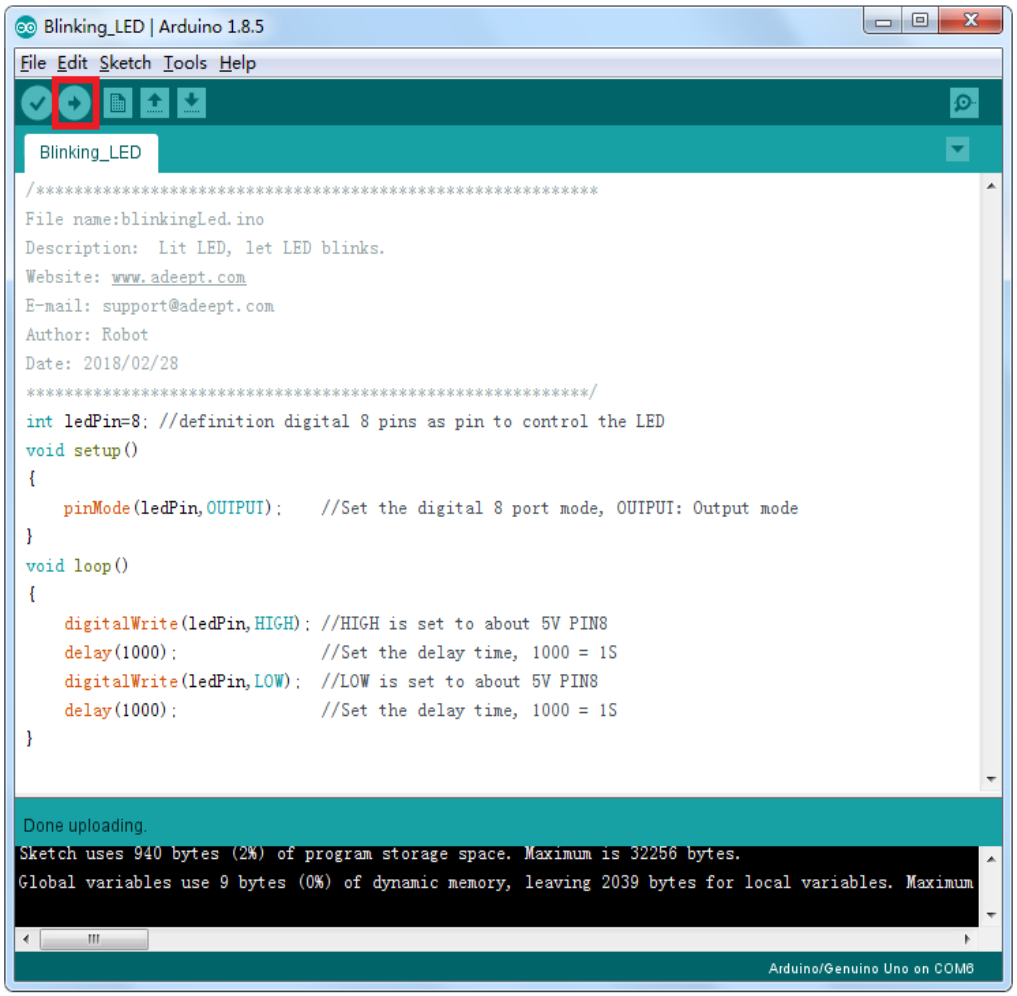
5. Choose the port number



Notice: In the tutorial, the port number of UNO which recognized by PC is COM6, here we can replace COM6 with COM1, COM2, COM3……

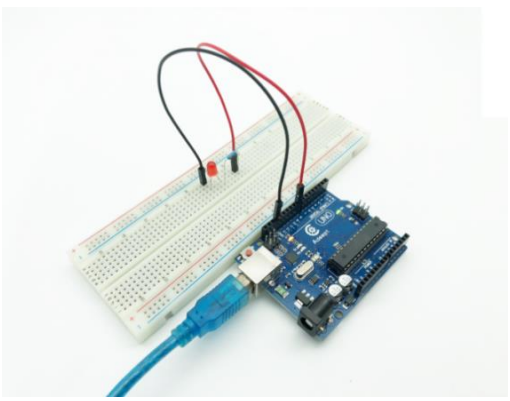
6. Compile the experimental code in IDE and download it to arduino UNO R3

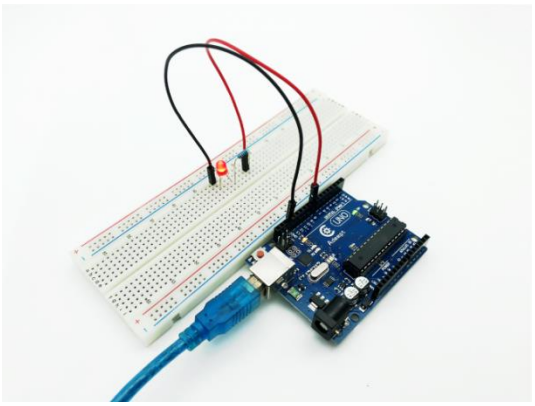
Experimental code:



Experimental result:

Now, we can see the LED lamp is flashing consecutively with one second on and one second off.





Experimental conclusion：

After this course, we know how to make the LED lamp flashing; next we will do more interesting experiments about the lamp.

Video Link: https://youtu.be/QpCAIuH7D_E

Project 1.2 LED Flowing Light

Experimental objective:

In this experiment, we will learn how to make an LED flowing light. This experiment is almost the same as the first one, and we need to control 8 LED lamps, and make it look like flowing water.

Required materials:

- 1* Arduino UNO
- 1* USB Cable
- 8* 220Ω Resistor
- 8* LED
- 1* Breadboard
- Several Jumper wires

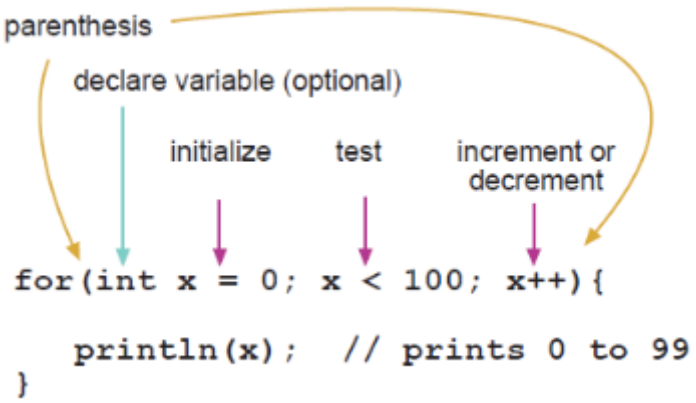
Required functions:

● for statements

The statement is used to repeat a block of statements enclosed in curly braces. An increment counter is usually used to increase and terminate the loop. The statement is useful for any repetitive operation, and is often used in combination with arrays to operate on collections of data/pins.

There are three parts to the for loop header:

```
For (initialization; condition; increment) {
//statement(s);
}
```



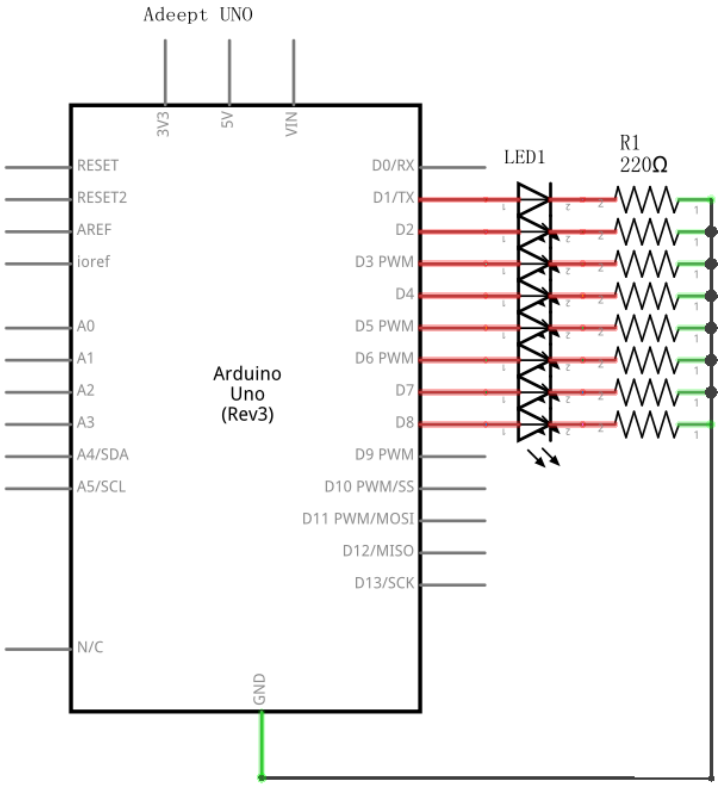
The initialization happens first and exactly once. Each time through the loop,

the condition is tested; if it's true, the statement block, and the increment is executed, then the condition is tested again. When the condition becomes false, the loop ends.

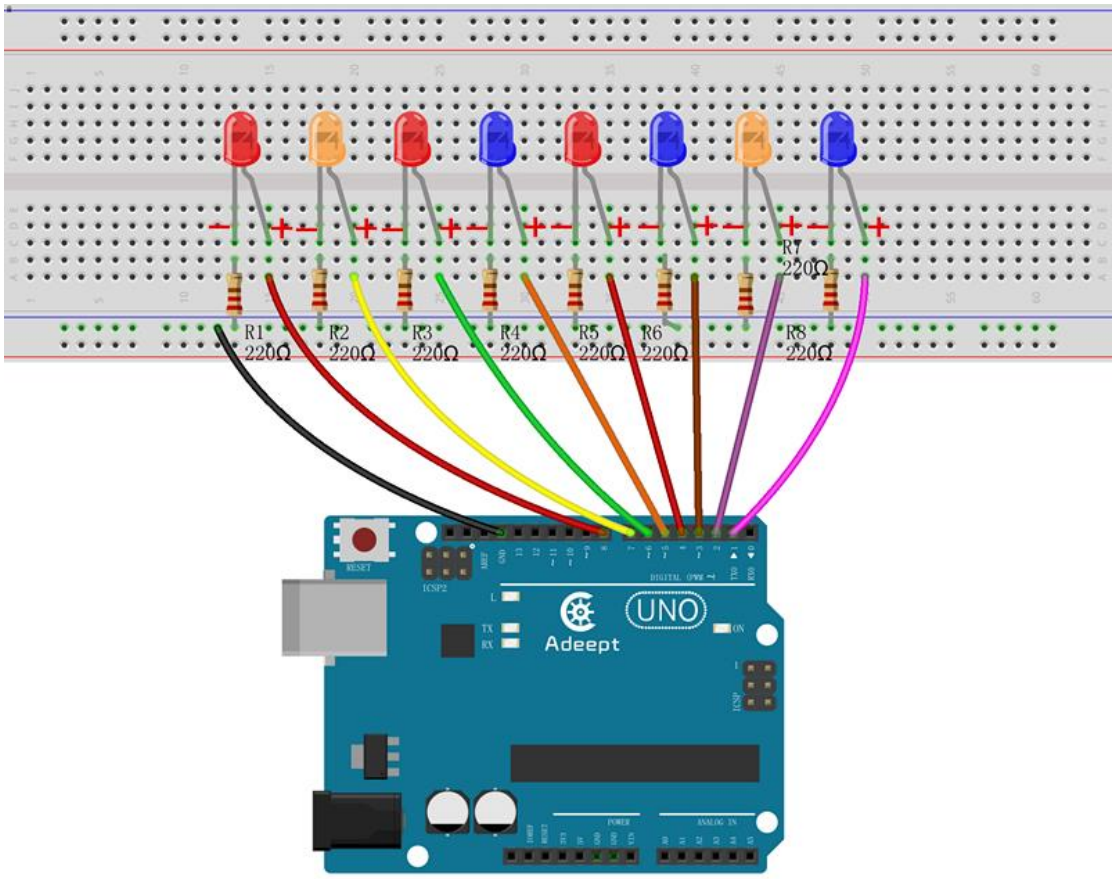
Operating steps :

1. When the power is off, we integrate all the required materials according to the schematic diagram or connection diagram.

Schematic diagram



connection diagram



2. Compile the experimental code and download it to arduino UNO R3

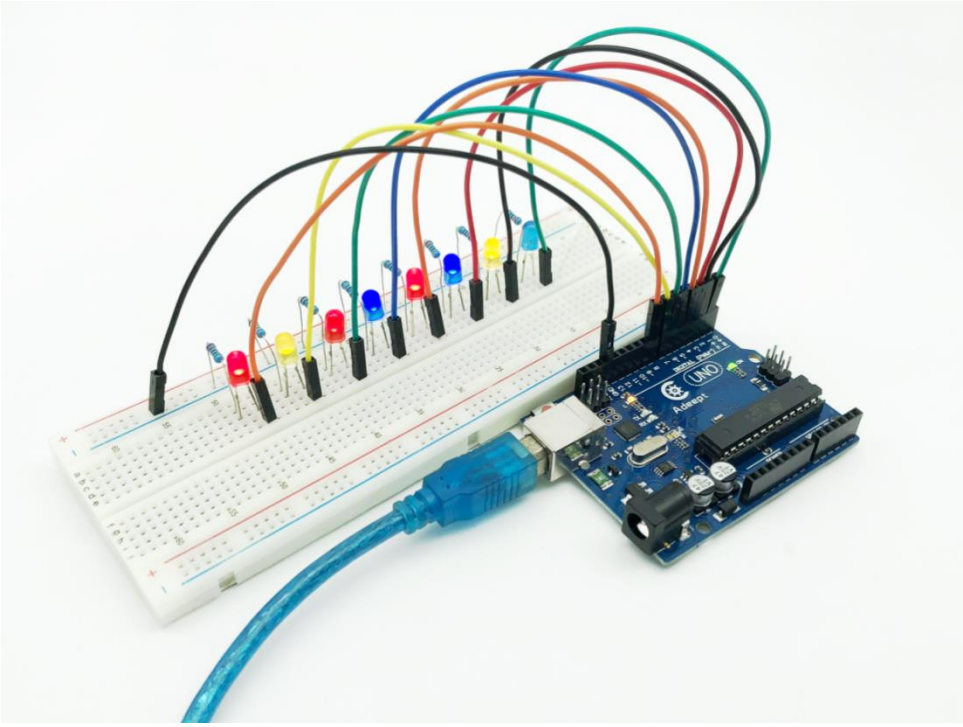
Experimental code

```
/*************************************************************
File name: flowingLed.ino
Description: LED turn lighting control
Website: www.adeept.com
E-mail: support@adeept.com
Author: Robot
Date: 2018/02/28
*************************************************************/
void setup()
{
  unsigned char ledPin;          //ledPin will be set to 1,2,3,4,5,6, 7
and 8.
  for(ledPin=1;ledPin<=8;ledPin++)//In turn set 1 ~ 8 digital pins to
output mode
  pinMode(ledPin,OUTPUT);        //Set the  ledPin pin to output mode
}
void loop()
```

```
{
  unsigned char ledPin;          //ledPin will be set to 1,2,3,4,5,6, 7
and 8.
  for(ledPin=1;ledPin<=8;ledPin++)//Every 200ms on in order LED1 ~ 8
  {
    digitalWrite(ledPin,HIGH);    //led on
    delay(200);                 //Delay 200 ms
  }
  for(ledPin=1;ledPin<=8;ledPin++)//Every 200ms off in order LED1 ~ 8
  {
    digitalWrite(ledPin,LOW);     //led off
    delay(200);                 //Delay 200 ms
  }
}
```

Experimental result:

Now we can see that the LED lamp is bright and extinguished in sequence just like flowing water



Experimental conclusion:

After this course, we know how to make an LED flowing light.

Chapter 2 Button

Project 2.1 Controlling an LED with a button

Experimental objective:

In this experiment, we will learn how to control an LED with a button.

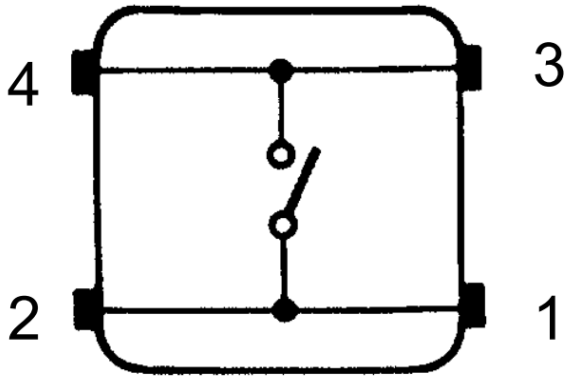
Required materials:

- 1* Arduino UNO
- 1* USB Cable
- 1* LED
- 1* Button
- 1* 10KΩ Resistor
- 1* 220Ω Resistor
- 1* Breadboard
- Several Jumper wires

Operating principle:

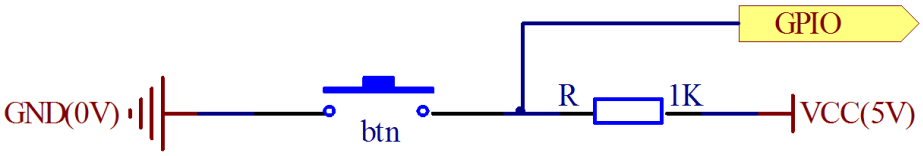
1. Button

Buttons are a common component used to control electronic devices. They are usually used as switches to connect or disconnect circuits. Although buttons come in a variety of sizes and shapes, the one used in this experiment will be a 12mm button as shown in the following pictures. Pins pointed out by the arrows of same color are meant to be connected.

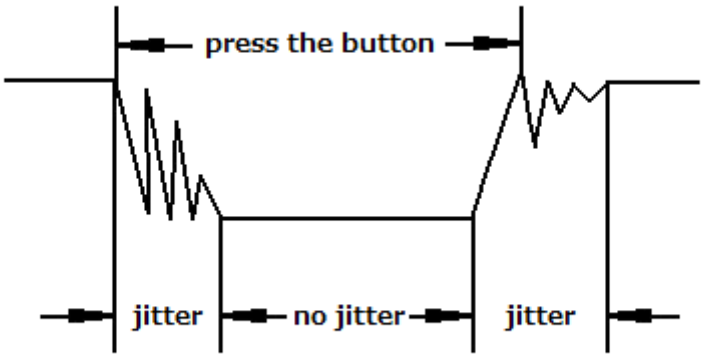


The button we used is a normally open type button. The two contacts of a button are in the off state under the normal conditions, only when the button is pressed they are closed.

The schematic diagram we used is as follows:

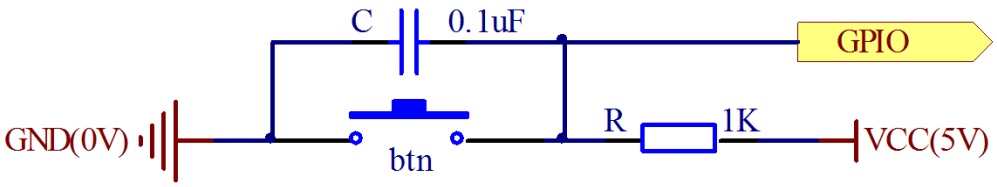


The button jitter must be happened in the process of using. The jitter waveform is as the flowing picture:



Each time you press the button, the Arduino will think you have pressed the button many times due to the jitter of the button. We must deal with the jitter of buttons before we use the button. We can use the software programming to remove the jitter of buttons, and you can use a capacitance to remove the jitter of buttons. We will introduce the software method. First, we detect

whether the level of button interface is low level or high level. When the level we detected is low level, 5~10 MS delay is needed, and then detect whether the level of button interface is low or high. If the signal is low, we can confirm that the button is pressed once. You can also use a 0.1 UF capacitance to clean up the jitter of buttons. The schematic diagram is shown below:



2. Interrupt

Hardware interrupts were introduced as a way to reduce wasting the processor's valuable time in polling loops, waiting for external events. They may be implemented in hardware as a distinct system with control lines, or they may be integrated into the memory subsystem.

Required functions:

● attachInterrupt(interrupt, ISR, mode)

It specifies a named Interrupt Service Routine (ISR) to call when an interrupt occurs. Replace any previous function that was attached to the interrupt. Most Arduino boards have two external interrupts: numbers 0 (on digital pin 2) and 1 (on digital pin 3).

Generally, an ISR should be as short and fast as possible. If your sketch uses multiple ISRs, only one can run at a time, other interrupts will be ignored (turned off) until the current one is finished. As delay () and millisecond () both rely on interrupts, they will not work while an ISR is running. delayMicroseconds (), which does not rely on interrupts, will work as expected.

Syntax

attachInterrupt (pin, ISR, mode)

Parameters

Pin: the pin number

ISR: the ISR will be called when the interrupt occurs; this function must take no parameters and return nothing. This function is sometimes referred to as an interrupt service routine.

Mode: defines when the interrupt should be triggered. Four constants are predefined as valid values:

-LOW to trigger the interrupt whenever the pin is low,

-CHANGE to trigger the interrupt whenever the pin changes value

-RISING to trigger when the pin goes from low to high,

-FALLING for when the pin goes from high to low.

● digitalRead()

Read the value from a specified digital pin, either HIGH or LOW.

Syntax

digitalRead (pin)

Parameters

Pin: the number of the digital pin you want to read (int)

Returns

HIGH or LOW

● delayMicroseconds(us)

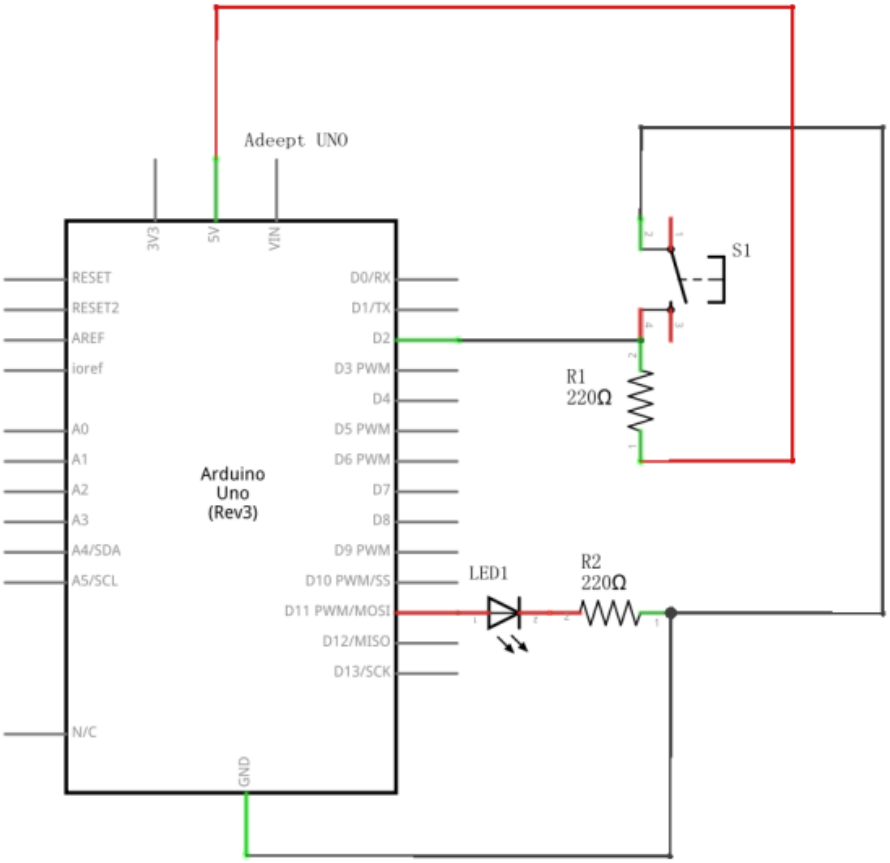
Pauses the program for the amount of time (in microseconds) specified as parameter. There are a thousand microseconds in a millisecond, and a million microseconds in a second.

Currently, the largest value that will produce an accurate delay is 16383. This could change in future Arduino releases. For delays longer than a few thousand microseconds, you should use delay () instead.

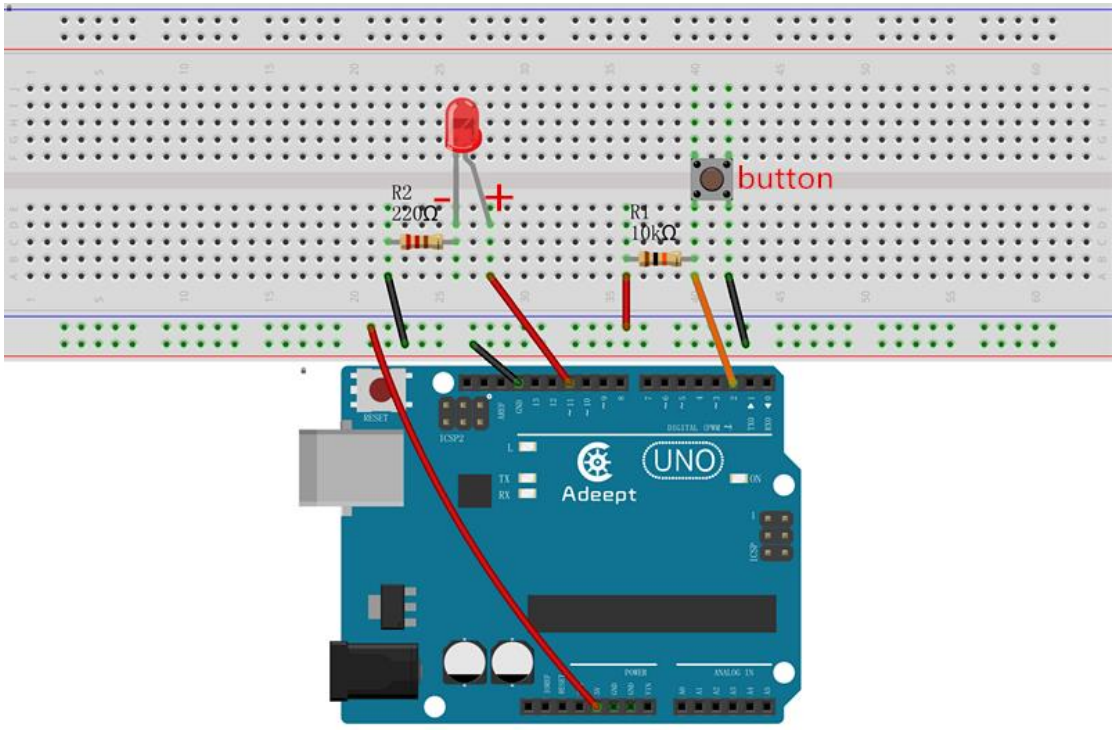
Operating steps：

1. When the power is off, we integrate all the required materials according to the schematic diagram or connection diagram.

Schematic diagram



connection diagram



2. Compile the experimental code and download it to arduino UNO R3

Experimental code

Examine the high and low electrical level and judge the state of the button.

```
/*************************************************************
File name: Controlling an LED with a button.ino
Description: When you press the button, you can see the state
             of the LED will be toggled. (ON->OFF, OFF->ON).
Website: www.adeept.com
E-mail: support@adeept.com
Author: Robot
Date: 2018/02/28
*************************************************************/

int ledpin=11;           //definition digital 11 pins as pin to control
the LED
int btnpin=2;            //Set the digital 2 to button interface

Volatile int state = LOW; // Defined output status LED Interface

Void setup()
```

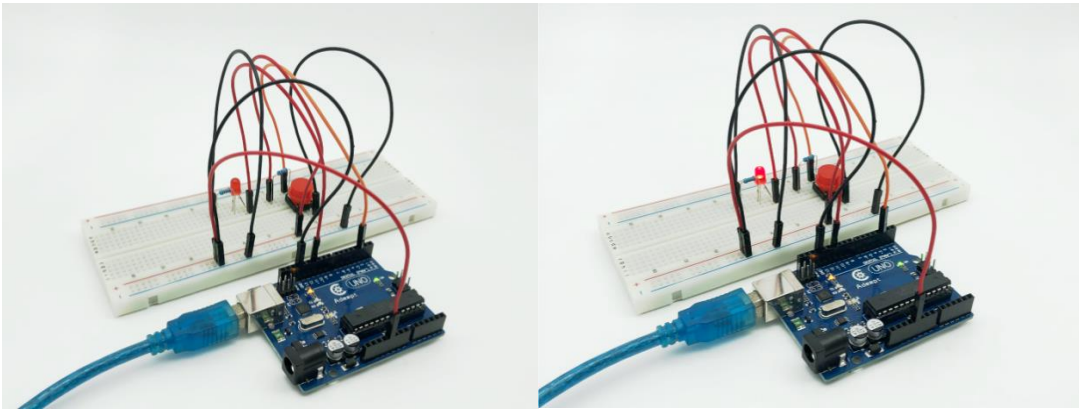
```
{
  pinMode(ledpin,OUTPUT);//Set digital 11 port mode, the OUTPUT for the
output
  pinMode(btnpin,INPUT); //Set digital 2 port mode, the INPUT for the input
}

void loop()
{
  if(digitalRead(btnpin)==LOW)         //Detection button interface to
low
  {
      delay(10);                      //Delay 10ms for the elimination of
key leading-edge jitter
      if(digitalRead(btnpin)==LOW)     //Confirm button is pressed
      {
        while(digitalRead(btnpin)==LOW);//Wait for key interfaces to high
        delay(10);                    //delay 10ms for the elimination of
key trailing-edge jitter
        while(digitalRead(btnpin)==LOW);//Confirm button release
        state = !state;               //Negate operation, each time you run
the program here, state the HGIH becomes LOW, or the state becomes the
LOW HGIH.
        digitalWrite(ledpin,state);    //Output control status LED, ON or
OFF
      }
   }
}
```

Interrupt the button and judge the state of it

```
/************************************************************
File name: btnAndLed02.ino
Description: Using interrupt mode, every time you press the
           button, LED status is switched(ON->OFF, OFF->ON).
Website: www.adeept.com
E-mail: support@adeept.com
Author: Robot
Date: 2018/02/28
************************************************************/
int ledpin=11;           //definition digital 11 pins as pin to control
the LED
int btnpin=2;            //Set the digital 2 to button interface
volatile int state = LOW; //Defined output status LED Interface

void setup()
{
  pinMode(ledpin, OUTPUT);                //Set digital 11 port mode, the
OUTPUT for the output
  attachInterrupt(0, stateChange, FALLING); //Monitoring Interrupt 0
(Digital PIN 2) changes in the input pins FALLING
}
void loop()
{
  digitalWrite(ledpin, state);            //Output control status LED,
ON or OFF
}
void stateChange()                        //Interrupt function
{
  if(digitalRead(btnpin)==LOW)            //Detection button interface
to low
  {
      delayMicroseconds(10000);           //Delay 10ms for the
elimination of key leading-edge jitter
      if(digitalRead(btnpin)==LOW)        //Confirm button is pressed
      {
          state = !state;                 //Negate operation, each time you
run the program here, state the HGIH becomes LOW, or the state becomes
the LOW HGIH.
      }
   }
}
```

Experimental result:

Now we can see that when the UNO is energized, if we press the button, the LED lamp will be bright, and if we press the button again, the lamp will extinguish.



Experimental conclusion:

After this course, we know how to use Arduino UNO to detect the state of an external button and then switch to LED.

Chapter 3 Tilt Switch

Project 3.1 Tilt Switch

Experimental objective：

In this experiment, we will learn how to use tilt switch and change the state of LED by changing the state of angle of the tilt switch.

Required materials:

- 1* Arduino UNO
- 1* USB Cable
- 1* Tilt Switch
- 1* LED
- 1* 220Ω Resistor
- 1* Breadboard
- Several Jumper Wires

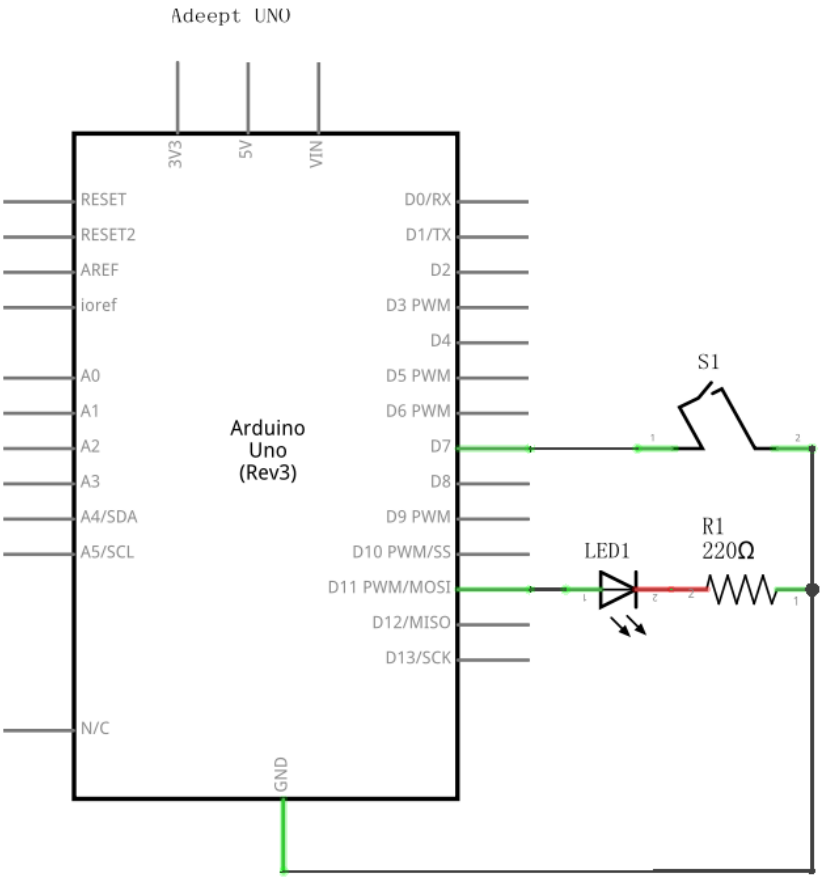
Operating principle：

The tilt switch is also called the ball switch. When the switch is tilted in the appropriate direction, the contacts will be connected, tilting the switch the opposite direction causes the metallic ball to move away from that set of contacts, thus breaking that circuit.

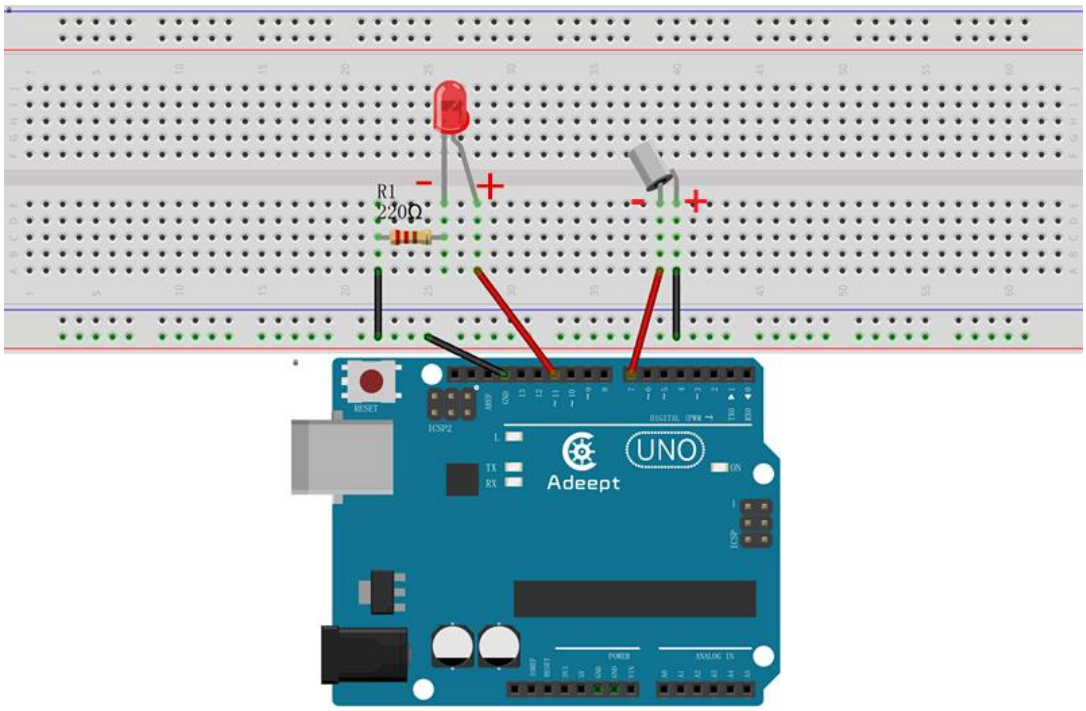
Operating steps：

1. When the power is off, we integrate all the required materials according to the schematic diagram or connection diagram.

Schematic diagram



connection diagram



2. Compile the experimental code and download it to arduino UNO R3

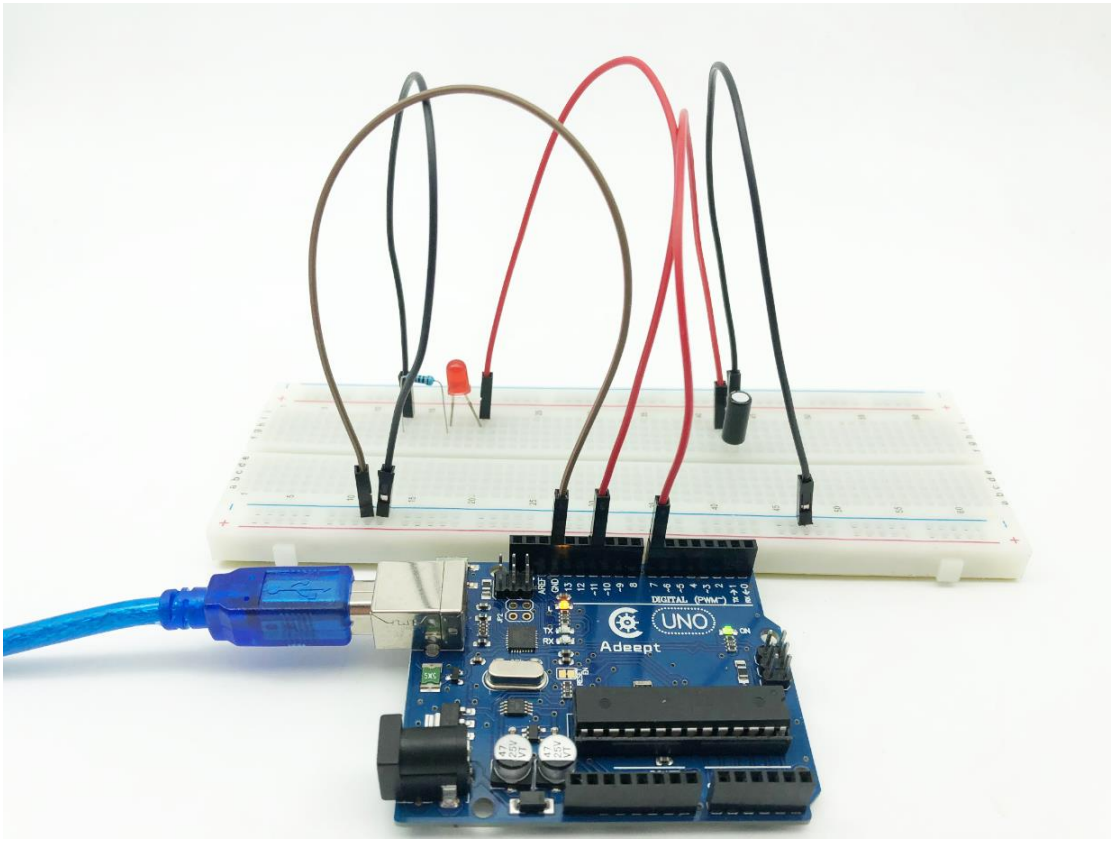
Experimental code

```
/*************************************************************
File name: tiltSwitch.ino
Description: Tilt switches to control the LED light on or off
Website: www.adeept.com
E-mail: support@adeept.com
Author: Robot
Date: 2018/02/28
*************************************************************/

int ledpin=11;      //definition digital 11 pins as pin to control the
LED
int tiltSwitchpin=7; //Set the digital 7 to tilt switch interface
int val;            //Define variable val

void setup()
{
  pinMode(ledpin,OUTPUT);     //Define small lights interface for the
output interface
  pinMode(tiltSwitchpin,INPUT_PULLUP);//define the tilt switch
interface for input interface
}
```

```
void loop()
{
  val=digitalRead(tiltSwitchpin);//Read the number seven level value is
assigned to val
  if(val==LOW)                  //Detect tilt switch is disconnected, the
tilt switch when small lights go out
  { digitalWrite(ledpin,LOW);}   //Output low, LED OFF
 else                          //Detection of tilt switch is conduction,
tilt the little lights up when the switch conduction
  { digitalWrite(ledpin,HIGH);}  //Output high, LED ON
}
```

Experimental result:

Now we can see that when the tilt switch is upright, the LED lamp extinguish, and when the tilt switch is tilted, the LED lamp will light.



Experimental conclusion:

After this class, we know another switch, and the use of switch is extensive, we can use it to do more interesting experiments after class.

Chapter 4 Buzzer

Project 4.1 Active Buzzer

Experimental objective：

In this experiment, we will learn how to use the active buzzer.

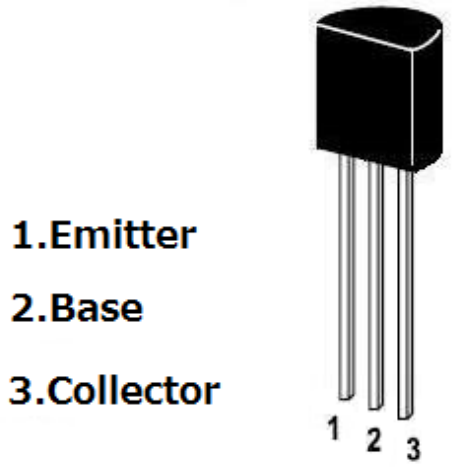
Required materials:

- 1* Arduino UNO
- 1* USB cable
- 1* Active buzzer
- 1* 220Ω Resistor
- 1* NPN Transistor (S8050)
- 1* Breadboard
- Several Jumper Wires

Operating principle：

A buzzer or beeper is an audio signaling device. As a type of electronic buzzer with integrated structure, which use DC power supply, buzzers are widely used in computers, printers, photocopiers, alarms, electronic toys, automotive electronic equipment, telephones, timers and other electronic products for voice devices. Buzzers can be categorized as active and passive buzzers (See the following pictures).

When you place the pins of buzzers upward, you can see that two buzzers are different, and buzzer that green circuit board exposed is the passive buzzer.
In this study, the buzzer we used is active buzzer. Active buzzer will sound as long as the power supply. We can program to make the Arduino output alternating high and low level, so that the buzzer sounds.
A slightly larger current is needed to make a buzzer sound. However, the output current of Arduino's GPIO is weak, so we need a transistor to drive the buzzer.
The main function of transistor is blowing up the voltage or current. The transistor can also be used to control the circuit conduction or deadline. And the transistor is divided into two kinds, one kind is NPN, for instance, the S8050 we provided; another kind is PNP transistor such as the S8550 we provided. The transistor we used is as shown in below:



There are two driving circuit for the buzzer:

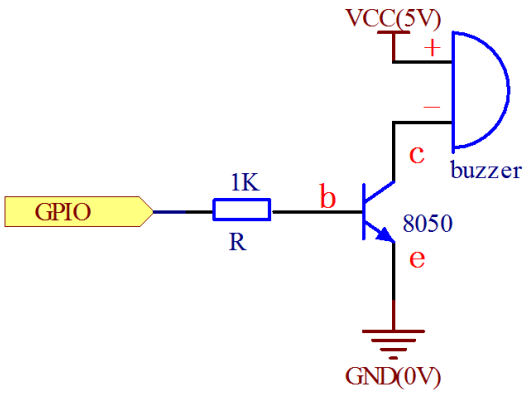


Figure1

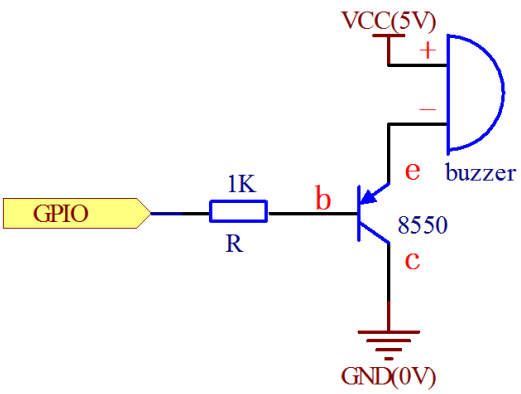


Figure2

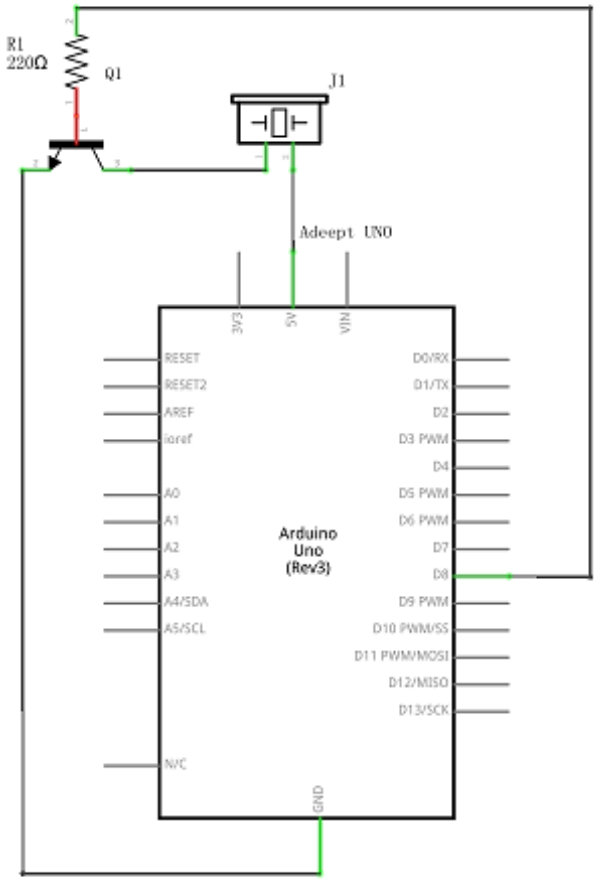
Figure 1: Set the Arduino GPIO as a high level, the transistor S8050 will

conduct, and then the buzzer will sound; set the Arduino GPIO as low level, the transistor S8050 will cut off, then the buzzer will stop.
Figure 2: Set the Arduino GPIO as low level, the transistor S8550 will conduct, and the buzzer will sound; set the Arduino GPIO as a high level, the transistor S8550 will cut off, then the buzzer will stop.

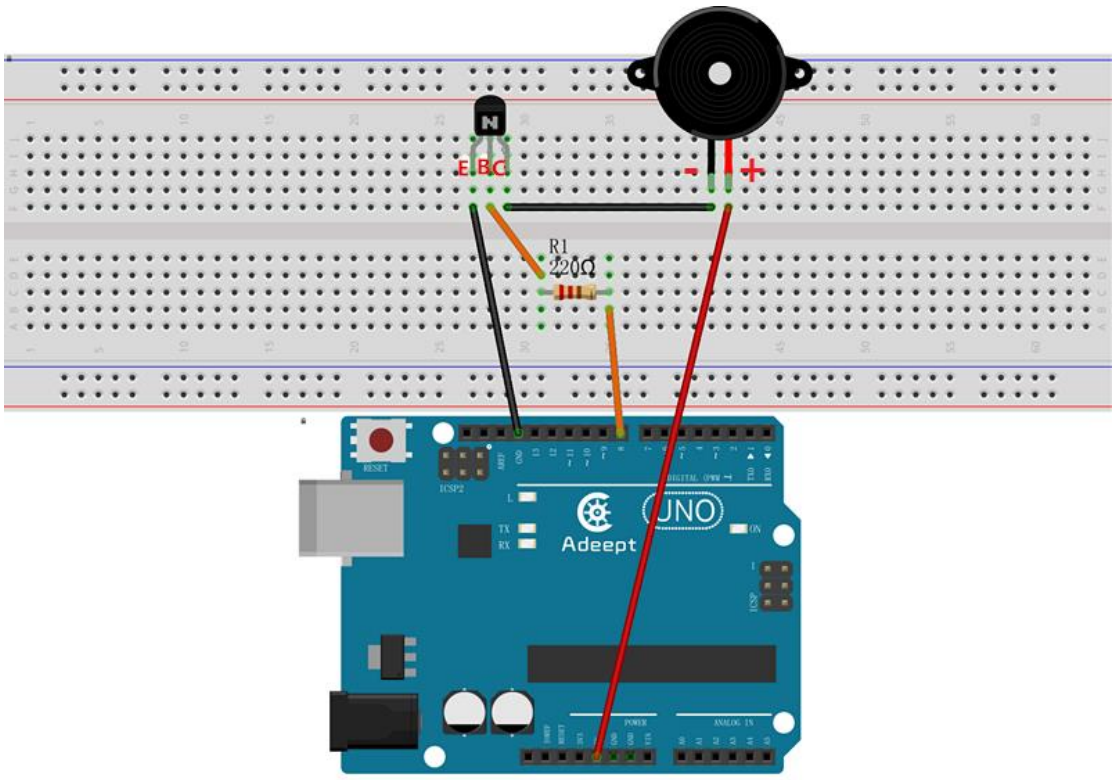
Operating steps：

1. When the power is off, we integrate all the required materials according to the schematic diagram or connection diagram.

Schematic diagram



connection diagram



2. Compile the experimental code and download it to arduino UNO R3

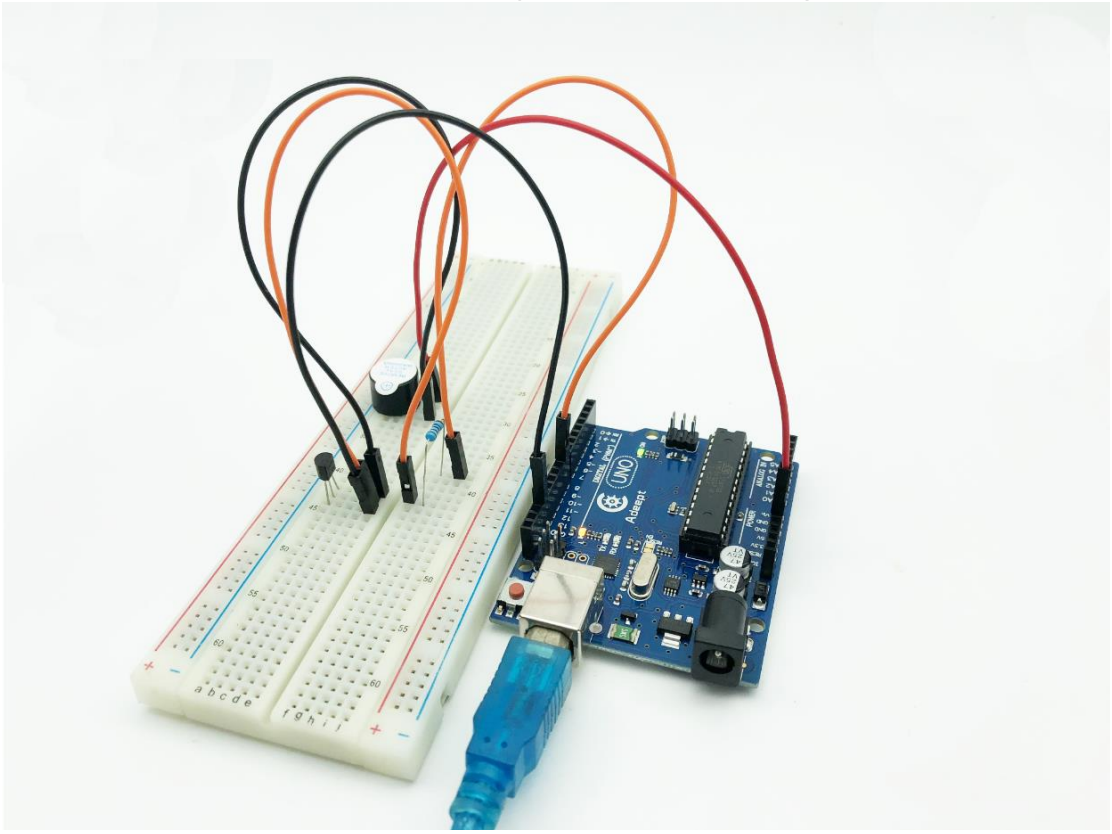
Experimental code

```
/************************************************************
File name:ActiveBuzzer.ino
Description: Arduino uno Continuous beeps control buzzer.
Website: www.adeept.com
E-mail: support@adeept.com
Author: Robot
Date: 2018/02/28
************************************************************/
int buzzerPin=8; //definition digital 8 pins as pin to control the buzzer
void setup()
{
    pinMode(buzzerPin,OUTPUT); //Set digital 8 port mode, the OUTPUT for
the output
}
void loop()
{
    digitalWrite(buzzerPin,HIGH); //Set PIN 8 feet as HIGH = 5 v
    delay(2000);                  //Set the delay time, 2000ms
    digitalWrite(buzzerPin,LOW);  //Set PIN 8 feet for LOW = 0 v
```

```
    delay(2000);                    //Set the delay time, 2000ms
}
```

Experimental result:

Now we can hear the active buzzer produce sound every two seconds.



Experimental conclusion:

In this course, we have learned the use of buzzer and the principle of triode. I hope you can use what you have learned in this course to do more interesting experiments.

Chapter 5 PWM

Project 5.1 Breathing LED

Experimental objective:

In this experiment, we will learn how to program Arduino to generate PWM signal and use PWM square wave to control LED's gradual lighten and darken.

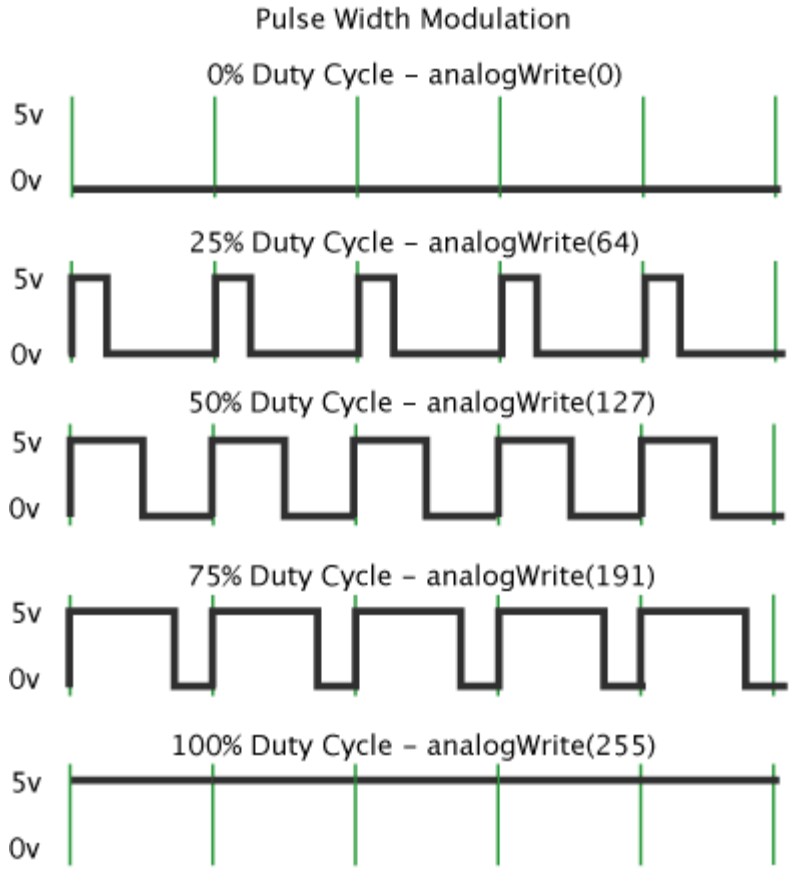
Required materials:

- 1* Arduino UNO
- 1* USB Cable
- 1* 220Ω Resistor
- 1* LED
- 1* Breadboard
- 1* Several Jumper wires

Operating principle:

Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages between full on (5 Volts) and off (0 Volts) by changing the portion of the time, the signal spends on versus, the time that the signal spends off. The duration of "on time" is called the pulse width. To get varying analog values, you change, or modulate that pulse width. If you repeat this on-off pattern fast enough with an LED for example, the result is as if the signal is a steady voltage between 0 and 5v controlling the brightness of the LED.

In the graphic below, the green lines represent a regular time period. This duration or period is the inverse of the PWM frequency. In other words, with Arduino's PWM frequency at about 500Hz, the green lines would measure 2 milliseconds each. A call to analogWrite () is on a scale of 0 - 255, such that analogWrite(255) requests a 100% duty cycle (always on), and analogWrite(127) is a 50% duty cycle (on half the time) for example.



Required functions:

●analogWrite()

Writing an analog value (PWM wave) to a pin can be used to light an LED at varying brightness or drive a motor at various speeds. After a call to analogWrite (), the pin will generate a steady square wave of the specified duty cycle until the next call to analogWrite () (or a call to digitalRead () or digitalWrite() on the same pin). You do not need to call pinMode () to set the pin as an output before calling analogWrite ().

Syntax

analogWrite (pin, value)

Parameters

Pin: the pin to write to.

Value: the duty cycle: between 0 (always off) and 255 (always on).

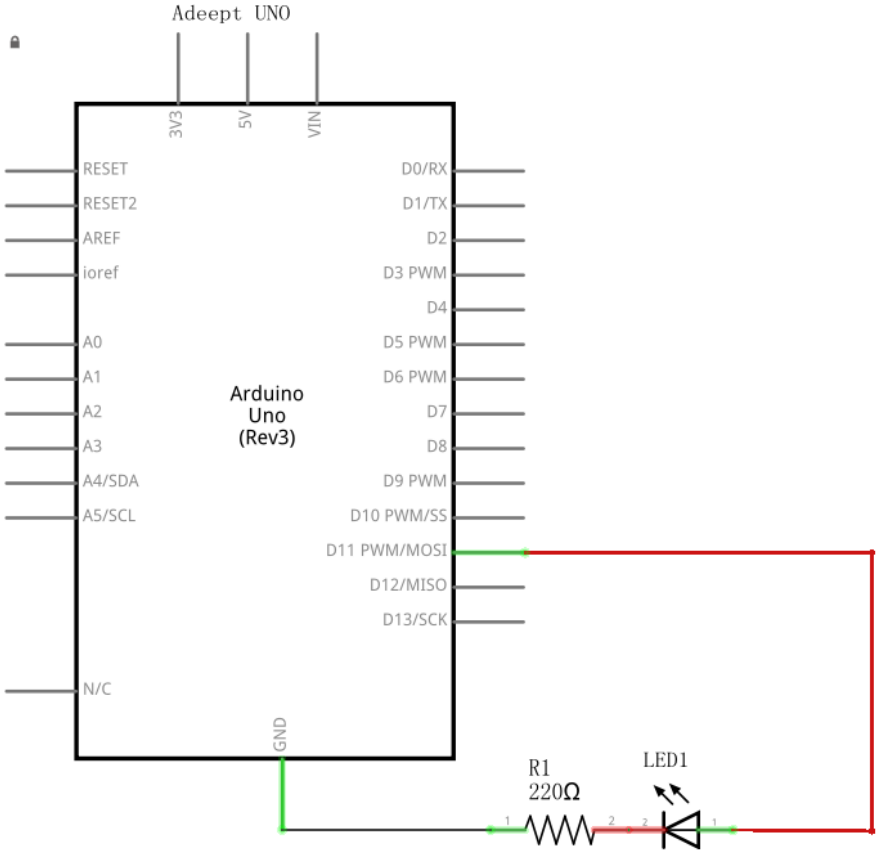
Returns

Nothing

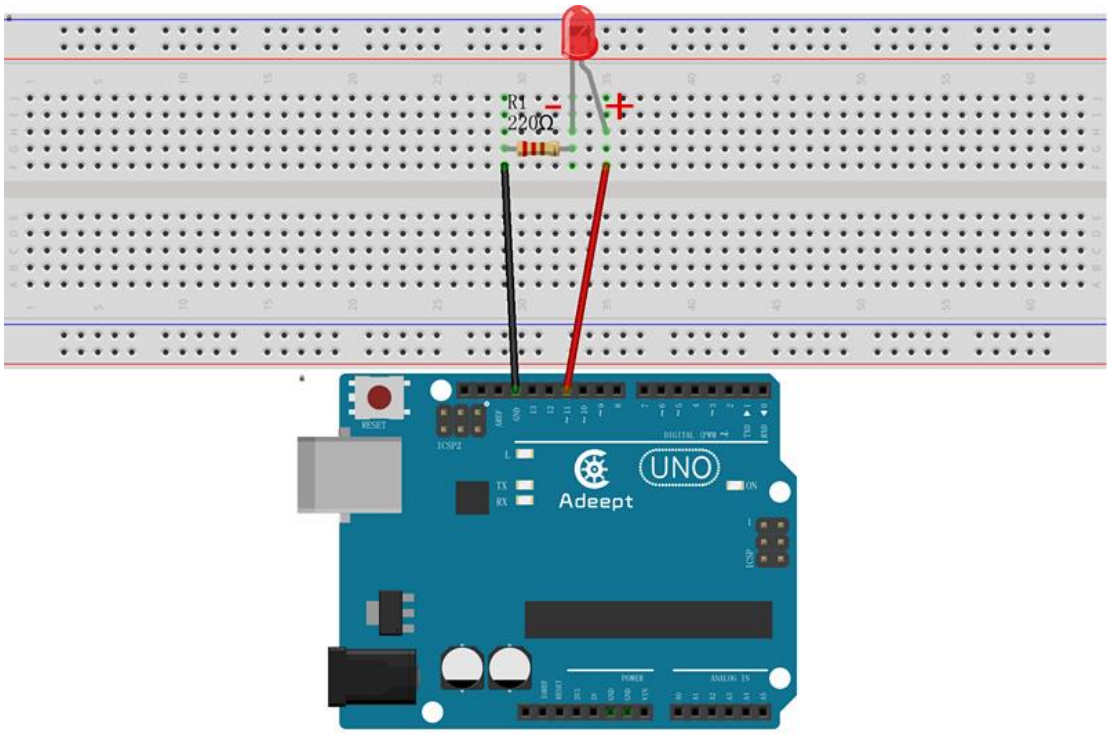
Operating steps：

1. When the power is off, we integrate all the required materials according to the schematic diagram or connection diagram.

Schematic diagram



connection diagram



2. Compile the experimental code and download it to arduino UNO R3

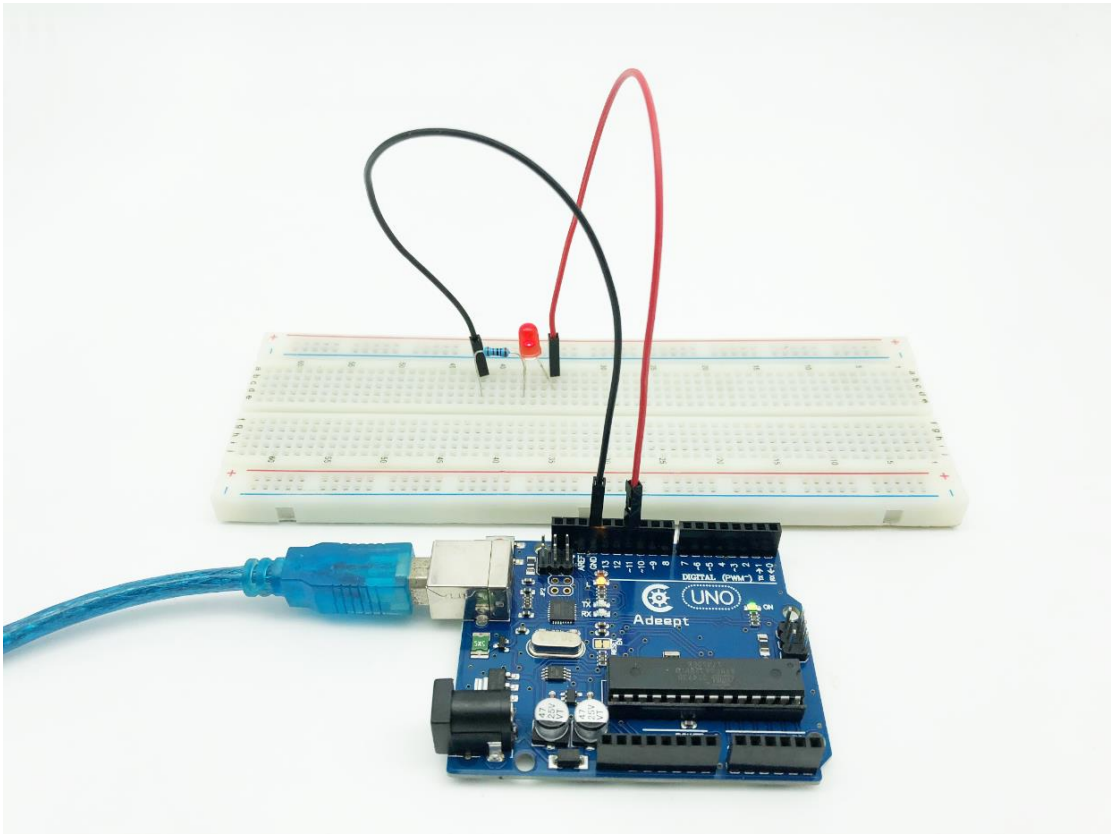
Experimental code:

```
/*************************************************************
File name: breathingLed.ino
Description: PWM control the LED gradually from dark to
           brighter, then from brighter to dark
Website: www.adeept.com
E-mail: support@adeept.com
Author:Robot
Date: 2018/02/28
*************************************************************/
int ledpin=11; //definition digital 11 pins as pin to control the LED

void setup ()
{
  pinMode(ledpin,OUTPUT); //Set digital 11 port mode, the OUTPUT for the
output
}

void loop()
{
```

```
   for (int a=0; a<=255;a++)  //Loop, PWM control of LED brightness
increase
   {
     analogWrite(ledpin,a);   //PWM output value a (0~255)
     delay(15);              //The duration of the current brightness
level. 15ms
   }
   for (int a=255; a>=0;a--)  //Loop, PWM control of LED brightness Reduced
   {
     analogWrite(ledpin,a);   //PWM output value a (255~0)
     delay(15);              //The duration of the current brightness
level. 15ms
   }
   delay(100);              //100ms delay
}
```

Experimental result:

Now we can see the brightness of LED is brighten gradually and darken gradually.



Experimental conclusion:

After this course, we know how to make a breathing lamp, up to now we have learned many experiments about lamp, and you can try more interesting experiments after class.

Project 5.2 Passive Buzzer

Experimental objective:

In this experiment, we will learn how to use passive buzzer to play a song.

Required materials:

- 1* Arduino UNO
- 1* USB Cable
- 1* 220Ω Resistor
- 1* NPN Transistor (S8050)
- 1* Passive Buzzer
- 1* Breadboard

Operating principle:

Before we have learned active buzzer, it is different from passive buzzer we are going to learn in this course.

Active buzzer:

There is vibration inside and drive circuit. Generate sound once power supplied. Generate the sound of the same frequency.

Passive buzzer:

The HZ of the sound can be controlled, and it can make the effect of Do Rai Mi Fa So La Xi Do

The main difference of active buzzer and passive buzzer: the input signal for products is different.

Required functions:

●tone()

Generate a square wave of the specified frequency (and 50% duty cycle) on a pin. A duration can be specified, otherwise the wave continues until a call to noTone (). The pin can be connected to a piezo buzzer or other speaker to play tones.

Only one tone can be generated at a time. If a tone is already playing on a different pin, the call to tone () will have no effect. If the tone is playing on the same pin, the call will set its frequency.

Use of the tone () function will interfere with PWM output on pins 3 and 11 (on boards other than the Mega).

NOTE: if you want to play different pitches on multiple pins, you need to call noTone () on one pin before calling tone () on the next pin.

Syntax

Tone (pin, frequency)

Tone (pin, frequency, duration)

Parameters

Pin: the pin on which to generate the tone

Frequency: the frequency of the tone in hertz - unsigned int

Duration: the duration of the tone in milliseconds (optional) - unsigned long

Returns

Nothing

● noTone()

It stops the generation of a square wave triggered by tone (). Has no effect if no tone is being generated.

NOTE: if you want to play different pitches on multiple pins, you need to call noTone () on one pin before calling tone () on the next pin.

Syntax

noTone (pin)

Parameters

Pin: the pin on which to stop generating the tone

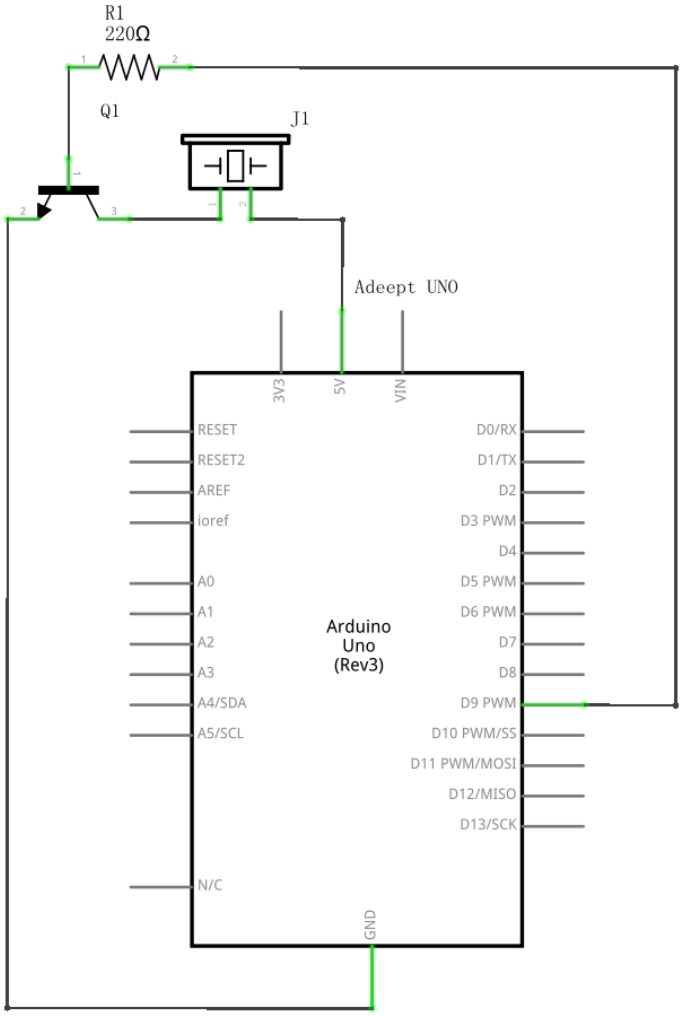
Returns

Nothing

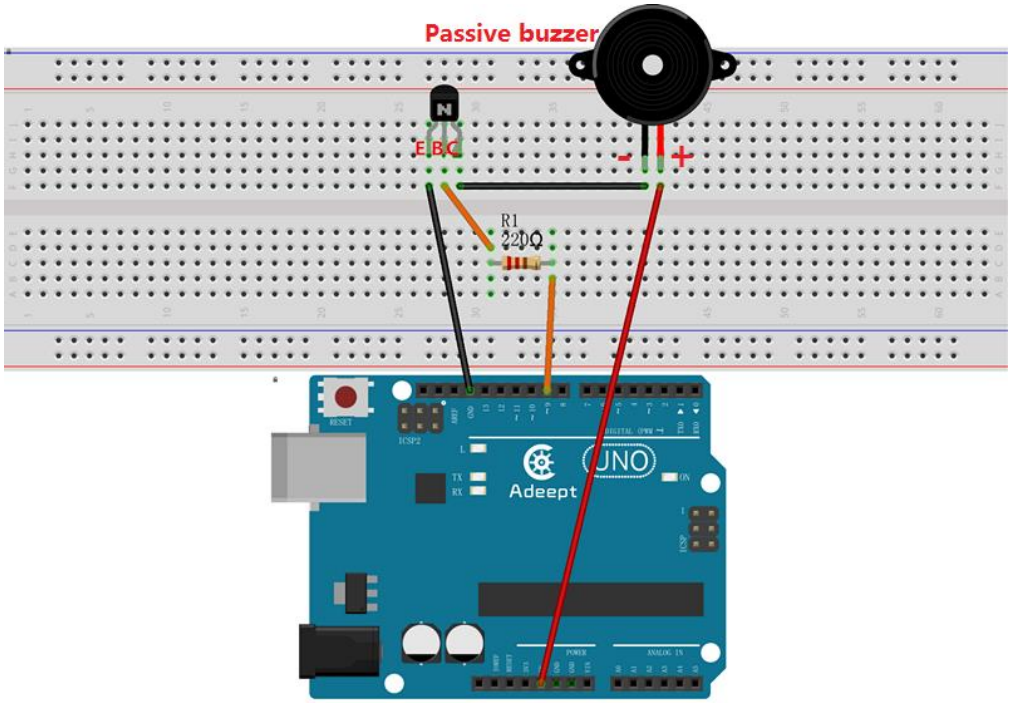
Operating steps :

1. When the power is off, we integrate all the required materials according to the schematic diagram or connection diagram.

Schematic diagram



connection diagram



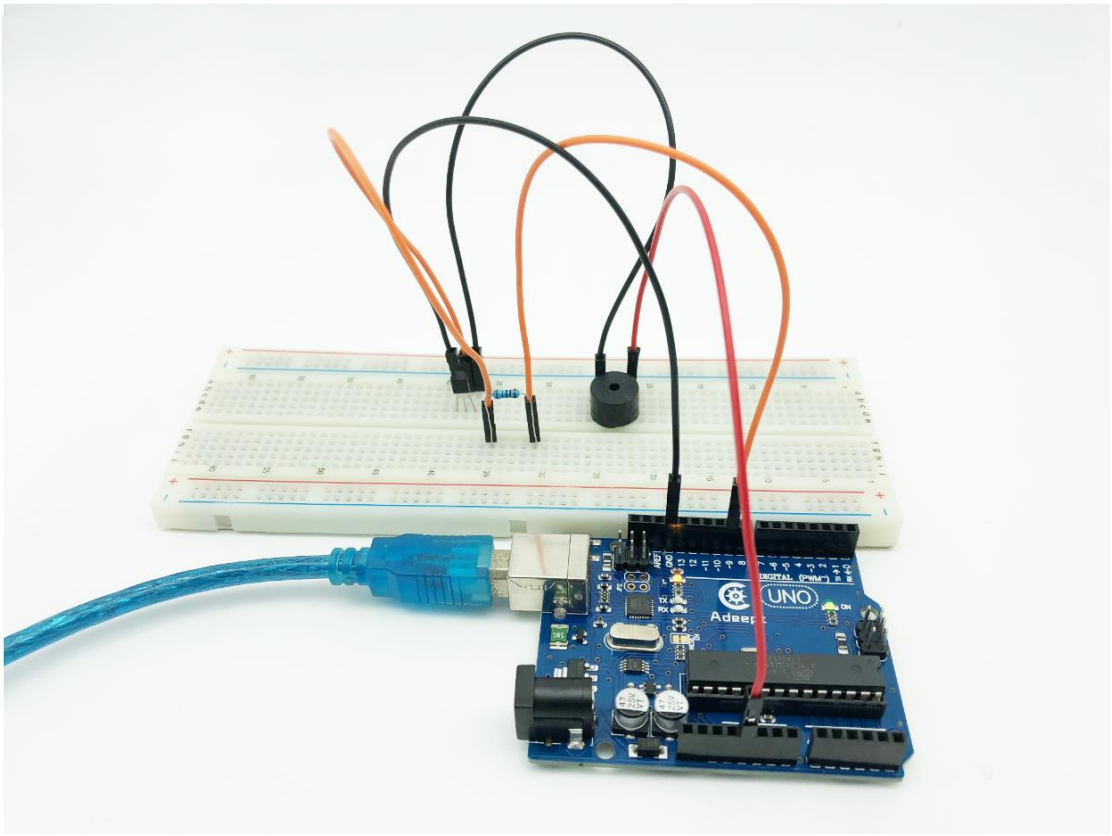
2. Compile the experimental code and download it to arduino UNO R3

Experimental code

Refer to Project 5.2 folder

Experimental result：

Now, we can clearly hear that the buzzer is playing music.



Experimental conclusion:

After this course, we know how to use passive buzzer and make a pleasant song, more than that, we can make buzzer play a song that we like.

Project 5.3 Controlling a RGB LED with PWM

Experimental objective:

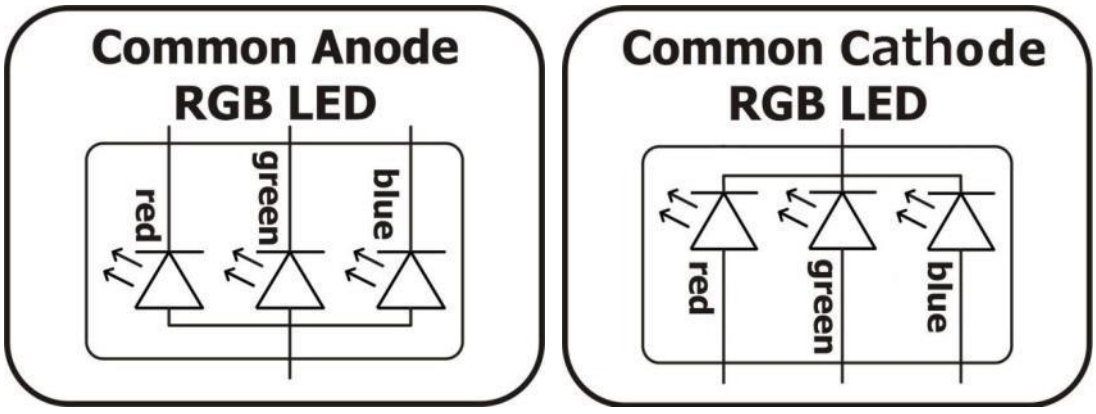
In this experiment, we will learn how to use PWM to control RGB lamp, and make it blink with different colors.

Required materials:

- 1* Arduino UNO
- 1* USB Cable
- 3* 220Ω Resistor
- 1* RGB LED
- 1* Breadboard
- Several Jumper wires

Operating principle:

RGB LEDs consist of three LEDs. Each LED actually has one red, one green and one blue light. These three colored LEDs are capable of producing any color. Tri-color LEDs with red, green, and blue emitters, in general using a four-wire connection with one common lead (anode or cathode). These LEDs can have either common anode or common cathode leads.



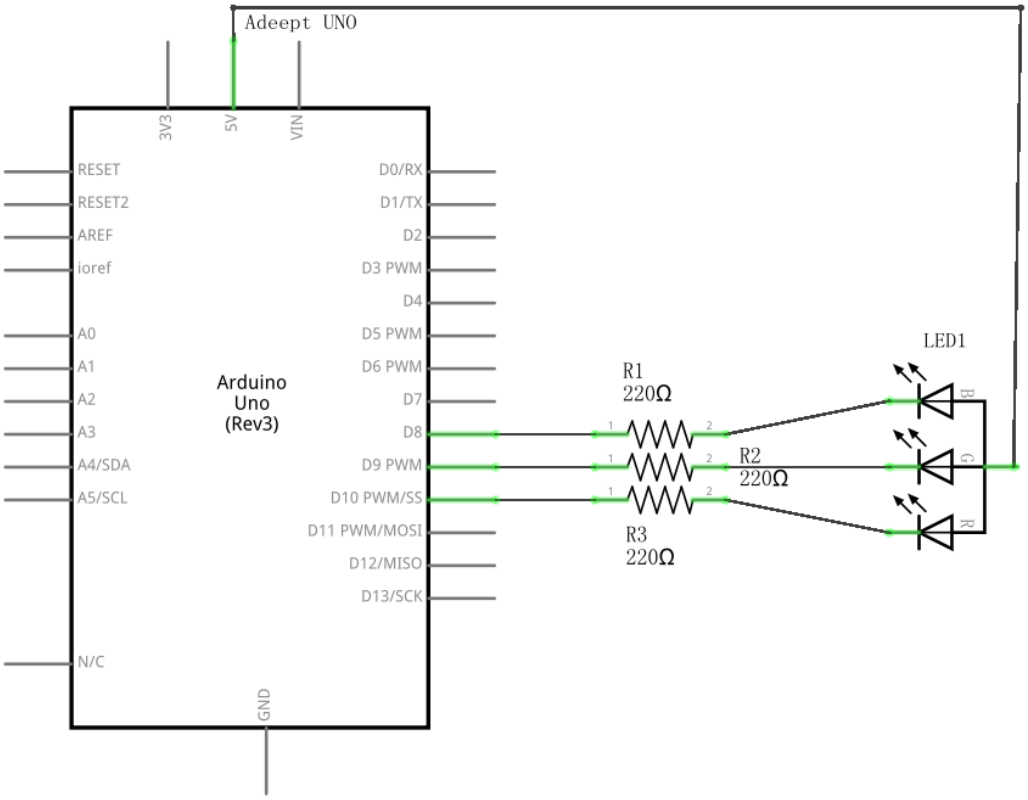
What we used in this experiment is the common anode RGB LED. The longest pin is the common anode of three LEDs. The pin is connected to the +5V pin of the Arduino, and the three remaining pins are connected to the Arduino's D9, D10, D11 pins through a current limiting resistor.

In this way, we can control the color of RGB LED by 3-channel PWM signal.

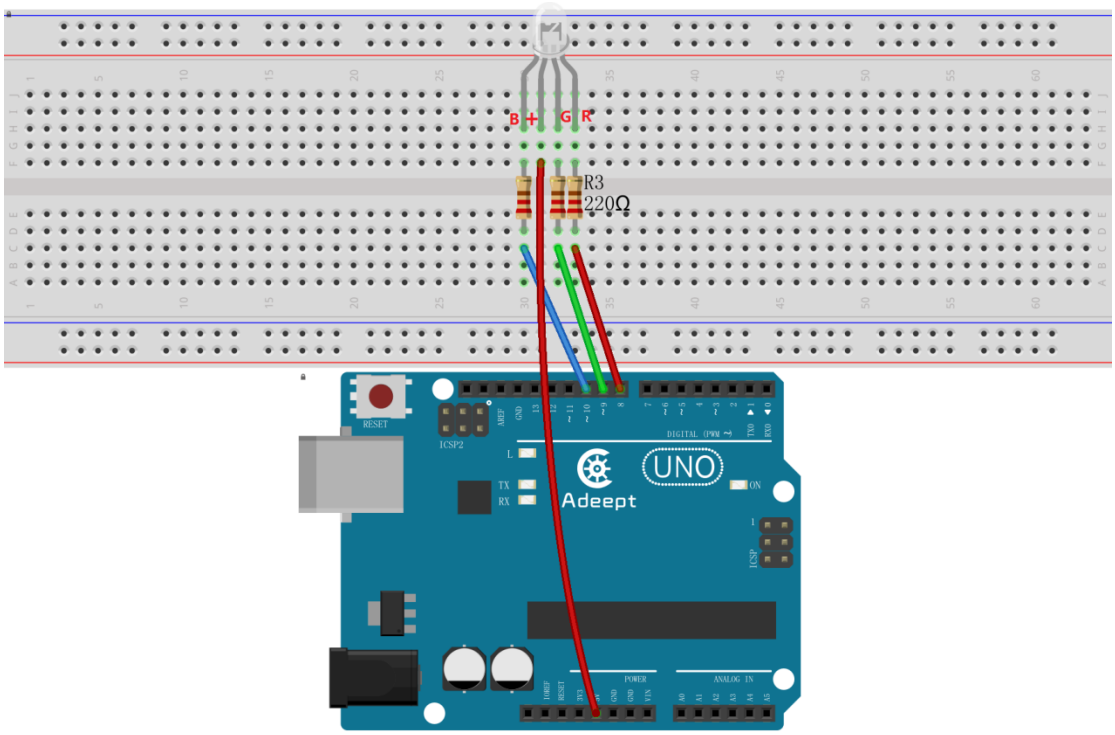
Operating steps :

1. When the power is off, we integrate all the required materials according to the schematic diagram or connection diagram.

Schematic diagram



connection diagram



2. Compile the experimental code and download it to arduino UNO R3

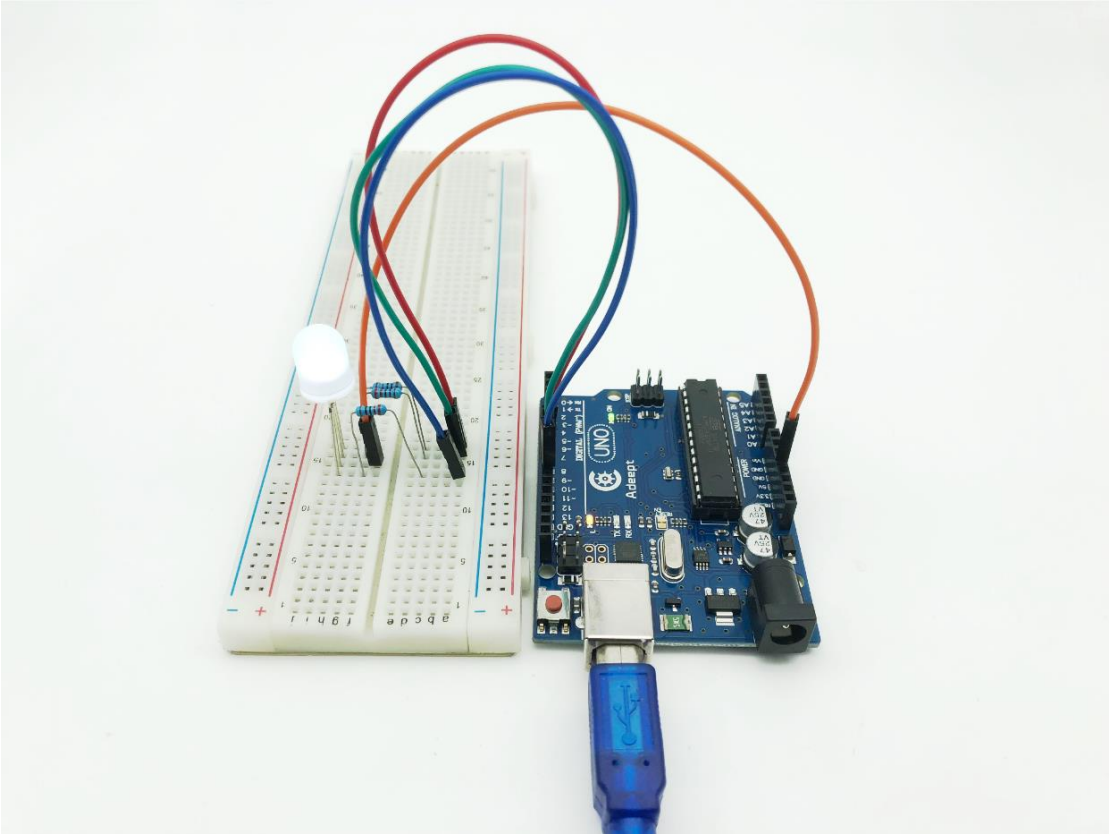
Experimental code

```
/*************************************************************
File name: rgbLed.ino
Description:Control the RGB LED emitting red, green, blue, yellow,
            white and purple light, then the RGB LED will be off,
            each state continues 1s, after repeating the above
            procedure.
Website: www.adeept.com
E-mail: support@adeept.com
Author: Robot
Date: 2018/02/28
*************************************************************/
int redPin = 10;   // R petal on RGB LED module connected to digital pin
11
int greenPin = 9; // G petal on RGB LED module connected to digital pin
9
int bluePin = 8;   // B petal on RGB LED module connected to digital pin
10
void setup()
{
```

```
    pinMode(redPin, OUTPUT);   // sets the redPin to be an output
    pinMode(greenPin, OUTPUT); // sets the greenPin to be an output
    pinMode(bluePin, OUTPUT);  // sets the bluePin to be an output
}
void loop()  // run over and over again
{
    // Basic colors:
    color(255, 0, 0); // turn the RGB LED red
    delay(1000);      // delay for 1 second
    color(0,255, 0);  // turn the RGB LED green
    delay(1000);      // delay for 1 second
    color(0, 0, 255); // turn the RGB LED blue
    delay(1000);      // delay for 1 second

    // Example blended colors:
    color(255,255,0);   // turn the RGB LED yellow
    delay(1000);        // delay for 1 second
    color(255,255,255); // turn the RGB LED white
    delay(1000);        // delay for 1 second
    color(128,0,255);   // turn the RGB LED purple
    delay(1000);        // delay for 1 second
    color(0,0,0);       // turn the RGB LED off
    delay(1000);        // delay for 1 second
}

void color (unsigned char red, unsigned char green, unsigned char blue)//
the color generating function
{
     analogWrite(redPin, 255-red);     // PWM signal output
     analogWrite(greenPin, 255-green); // PWM signal output
     analogWrite(bluePin, 255-blue);   // PWM signal output
}
```

Experimental result：

Now, we can see the RGB lamp is changing color constantly.



Experimental conclusion:

After this course, we basically understand how to use PWM to control the change of color of RGB lamp.

Chapter 6 7-segment display

Project 6.1 7-segment display

Experimental objective：

In this experiment, we will learn how to control the display of 7-segment.

Required materials:

- 1* Arduino UNO
- 1* USB Cable
- 1* 220Ω Resistor
- 1* 7-Segment display
- 1* Breadboard
- Several Jumper wires

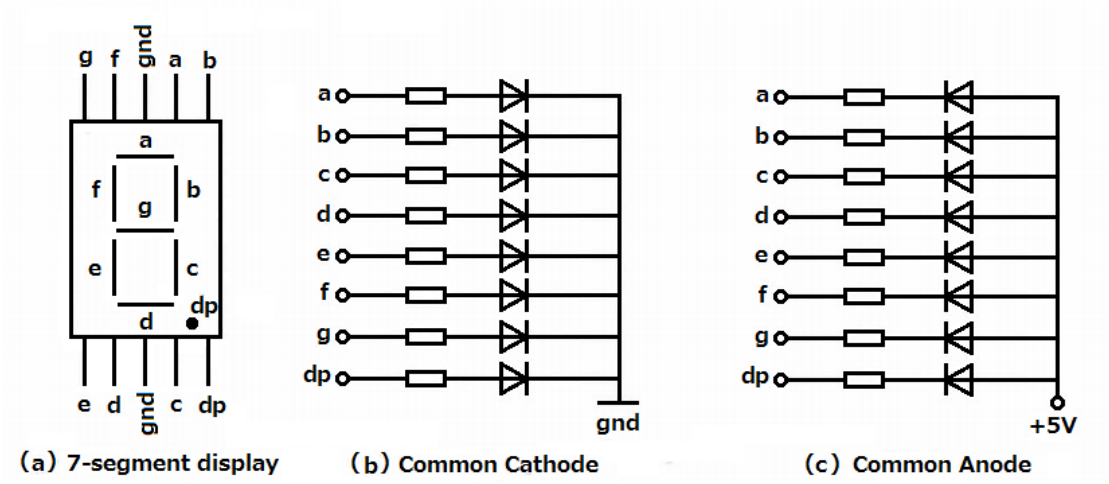
Operating principle：

The seven-segment display is a form of electronic display device for displaying decimal numerals that is an alternative to the more complex dot matrix displays.

Seven-segment displays are widely used in digital clocks, electronic meters, basic calculators, and other electronic devices that display numerical information.

The seven-segment display is an 8-shaped LED display device composed of eight LEDs (including a decimal point), these segments respectively named a, b, c, d, e, f, g, dp.

The segment display can be divided into common anode and common cathode segment display by internal connections.

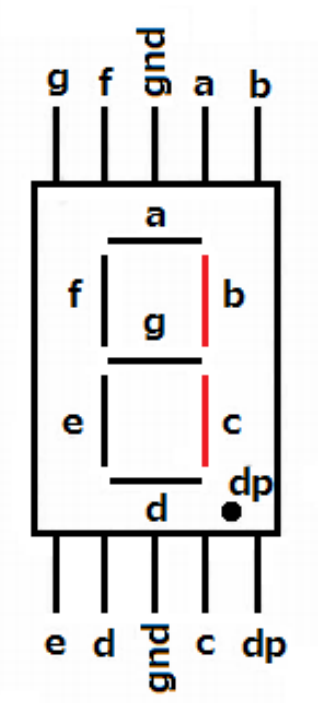


(a) 7-segment display (b) Common Cathode (c) Common Anode

When using a common anode LED, the common anode should be connected to the power supply (VCC); when using a common cathode LED, the common cathode should be connected to the ground (GND).

Each segment of a segment display is composed of LED, so a resistor is needed for protecting the LED.

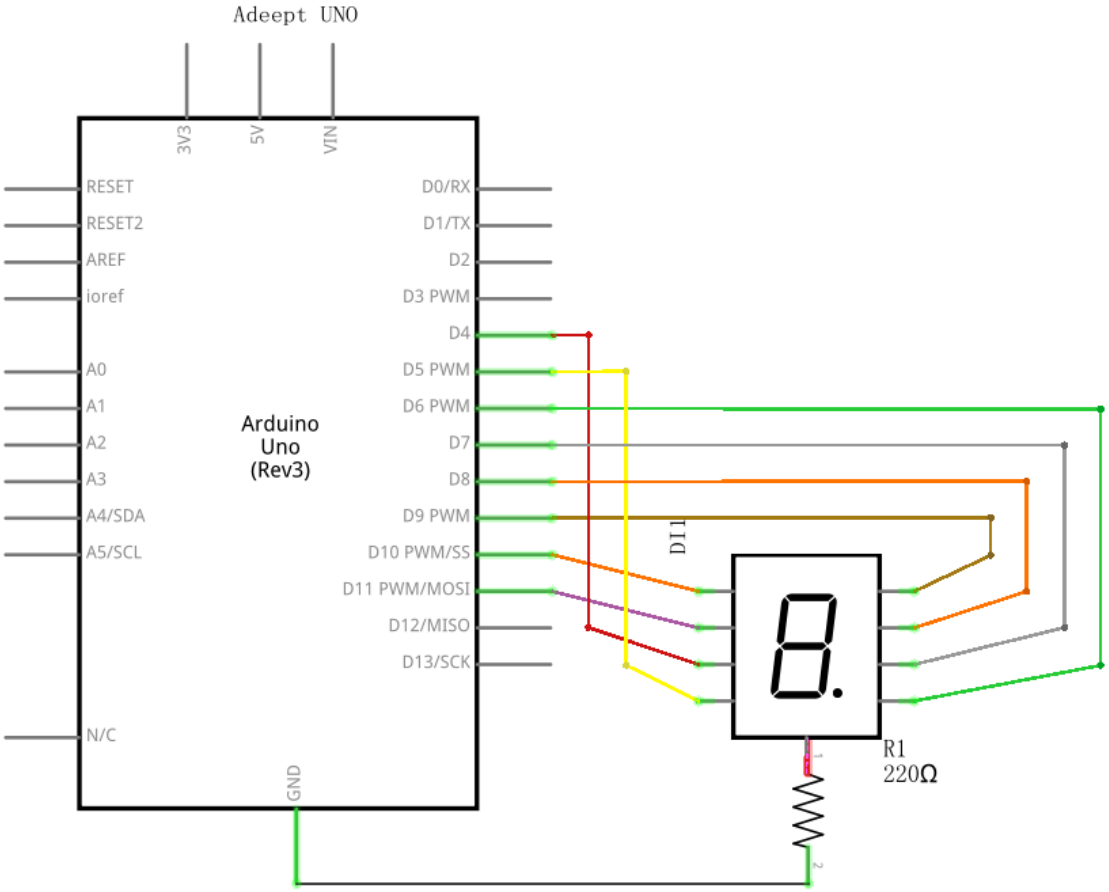
A 7-segment display has seven segments for displaying a figure and a segment for displaying a decimal point. If you want to display a number '1', you should only light the segment b and c.



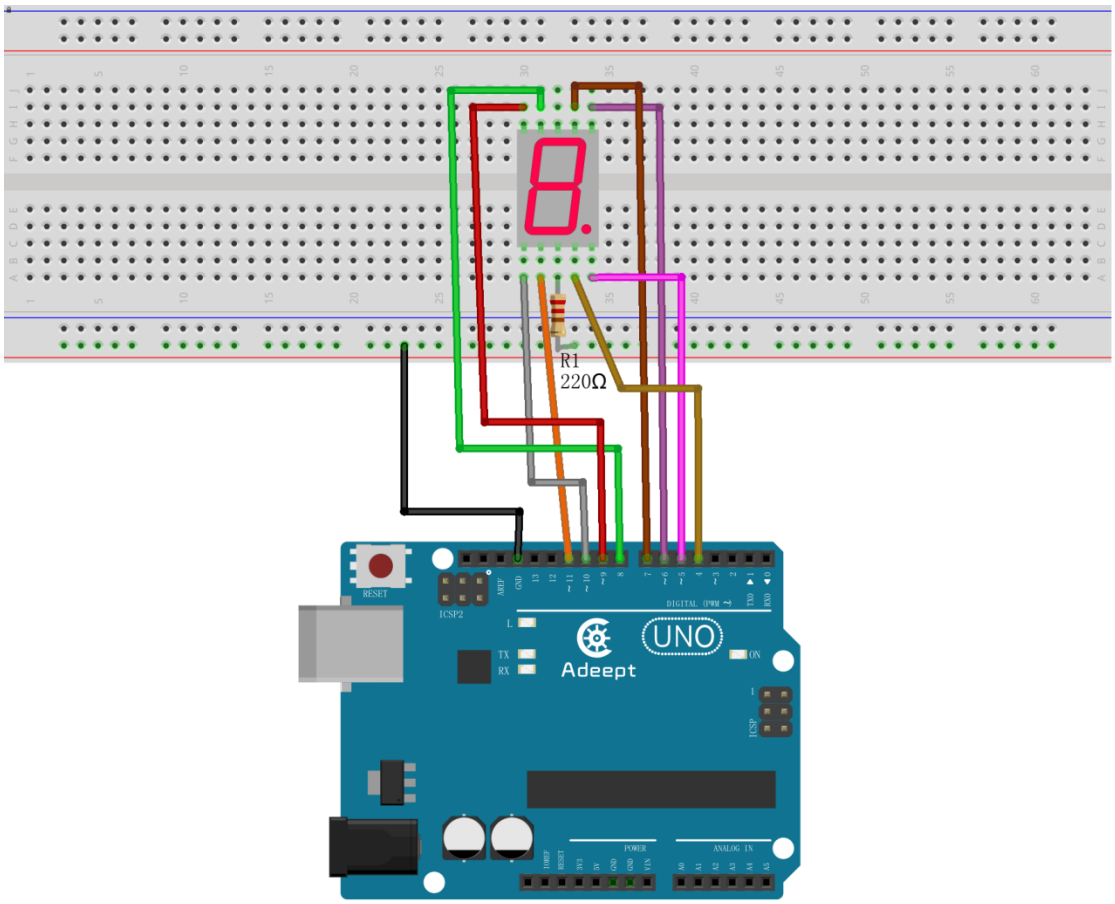
Operating steps:

1. When the power is off, we integrate all the required materials according to the schematic diagram or connection diagram.

Schematic diagram



connection diagram



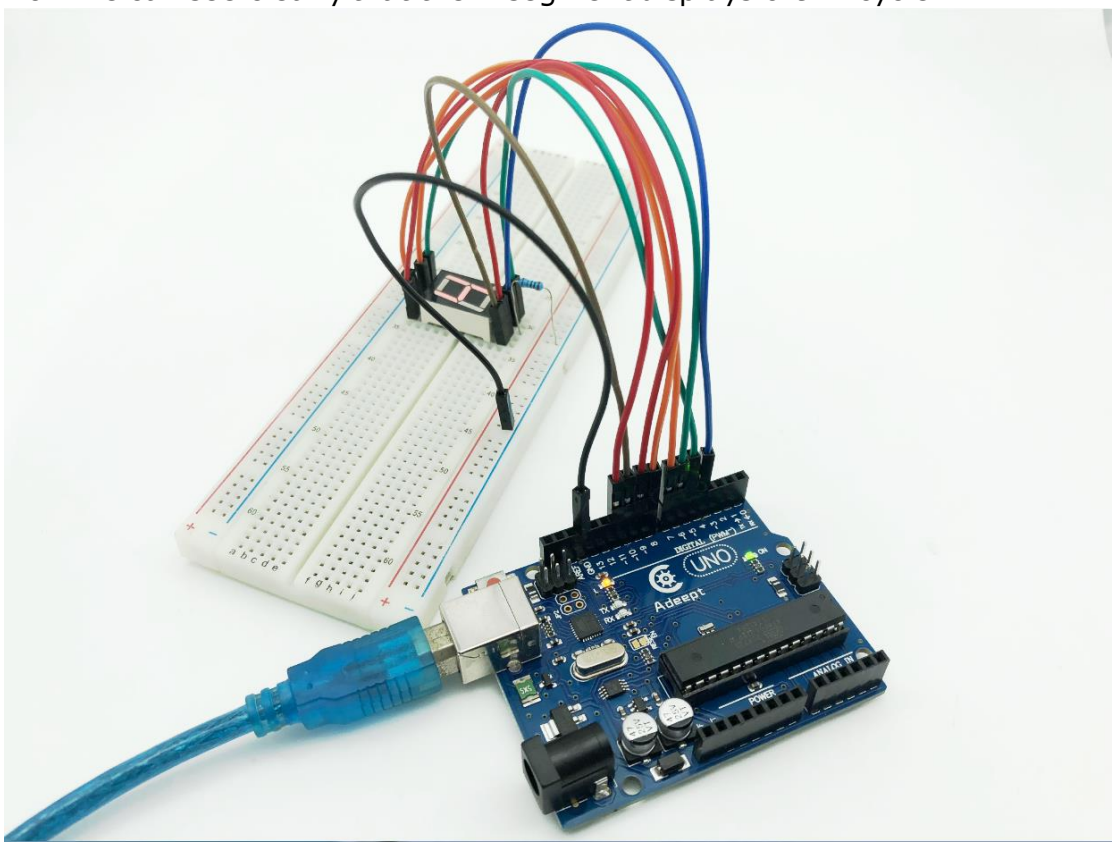
2. Compile the experimental code and download it to arduino UNO R3

Experimental code

Refer to Project 6.1 code folder.

Experimental result:

Now we can see clearly that the 7-segment displays 0-9 in cycle.



Experimental conclusion:

After this course, we know how to use 7-segment, and you can expand more with what we have learned in this course.

Chapter 7 Analog

Project 7.1 Photoresistor

Experimental objective：

In this experiment, we will learn how to use photoresistor and display its data on serial port.

Operating principle：

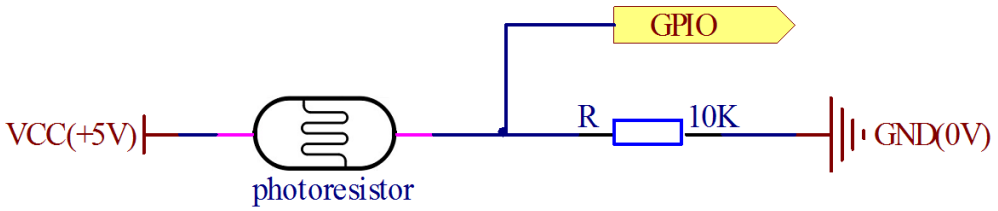
- 1* Arduino UNO
- 1* USB cable
- 1* Photoresistor
- 1* 10KΩ Resistor
- 1* Breadboard
- Several Jumper wires

Operating principle：

A photoresistor is a light-controlled variable resistor. The resistance of a photoresistor decreases with the increasing incident light intensity; in other words, it exhibits photoconductivity. A photoresistor can be applied in light-sensitive detector circuits.

A photoresistor is made of a high resistance semiconductor. In the dark, a photoresistor can have a resistance as high as a few megohms (MΩ), while in the light, a photoresistor can have a resistance as low as a few hundred ohms. If incident light on a photoresistor exceeds a certain frequency, photons absorbed by the semiconductor give bound electrons enough energy to jump into the conduction band. The resulting free electrons (and their hole partners) conduct electricity, thereby lowering resistance. The resistance range and sensitivity of a photoresistor can substantially differ among dissimilar devices. Moreover, unique photoresistors may react substantially differently to photons within certain wavelength bands.

The schematic diagram of this experiment is shown below:

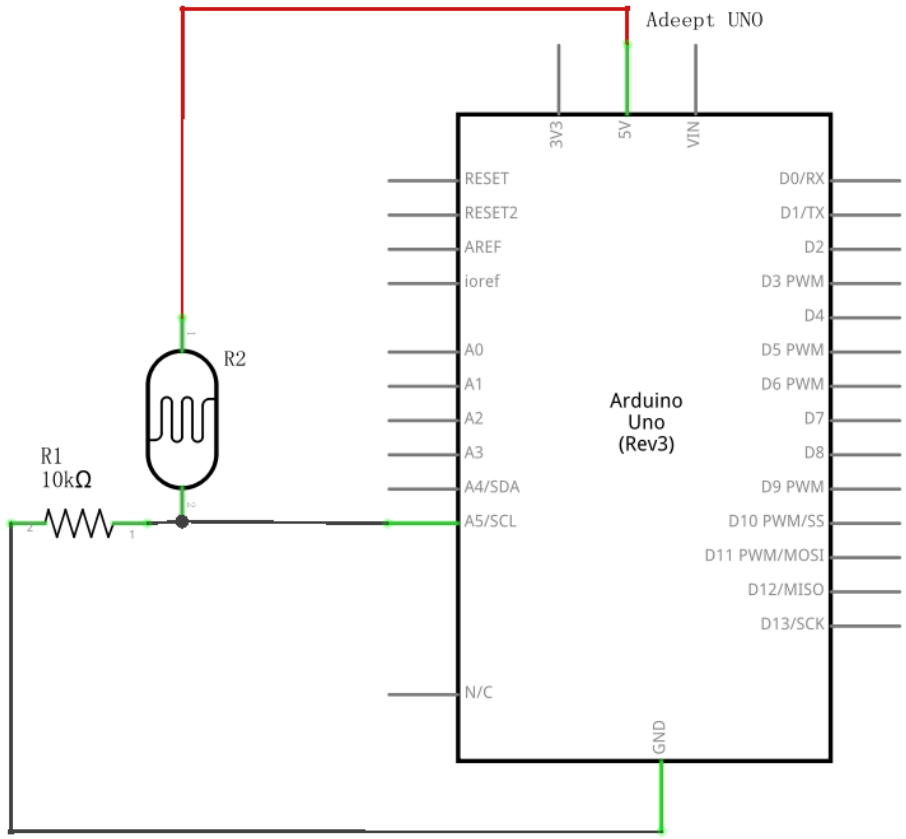


With the increase of the light intensity, the resistance of photoresistor will be decreased. The voltage of GPIO port in the above figure will become high.

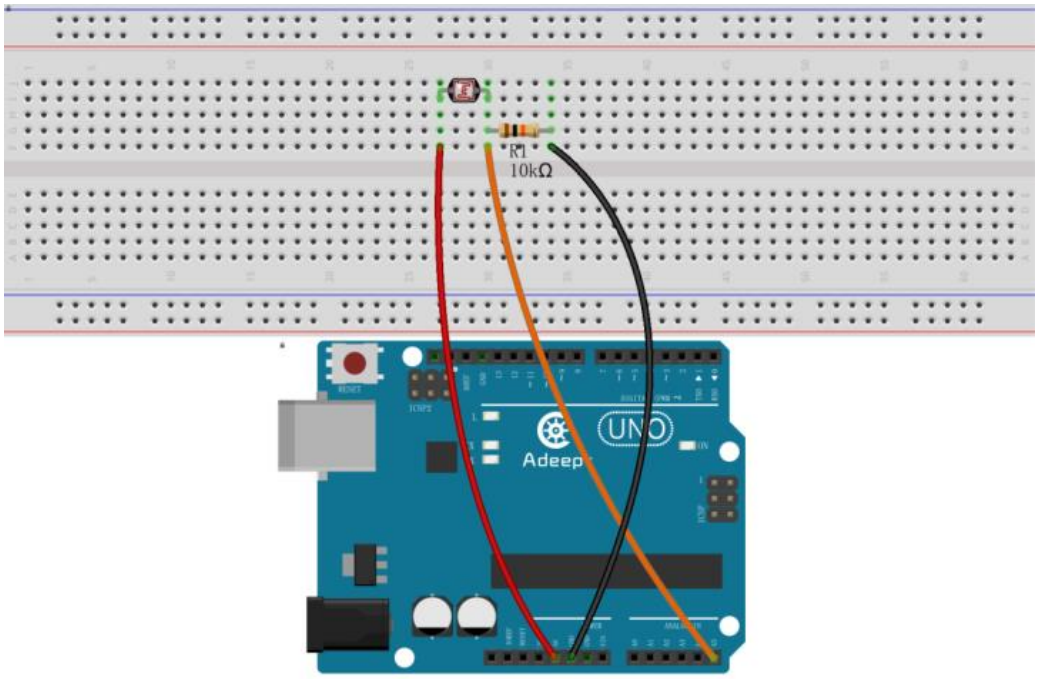
Operating steps：

1. When the power is off, we integrate all the required materials according to the schematic diagram or connection diagram.

Schematic diagram



connection diagram



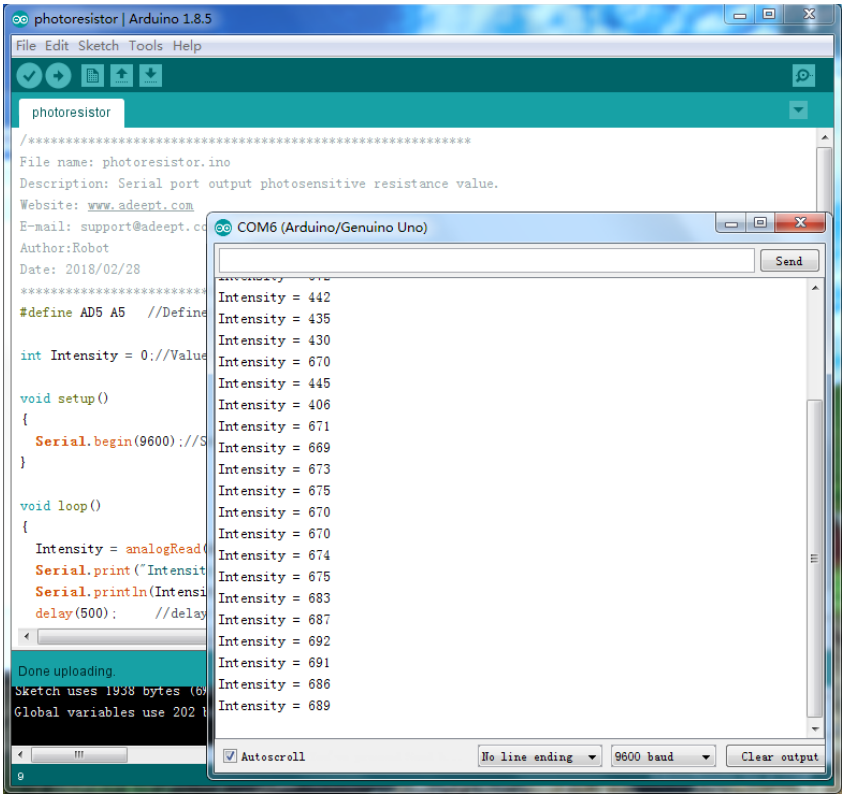
2. Compile the experimental code and download it to arduino UNO R3

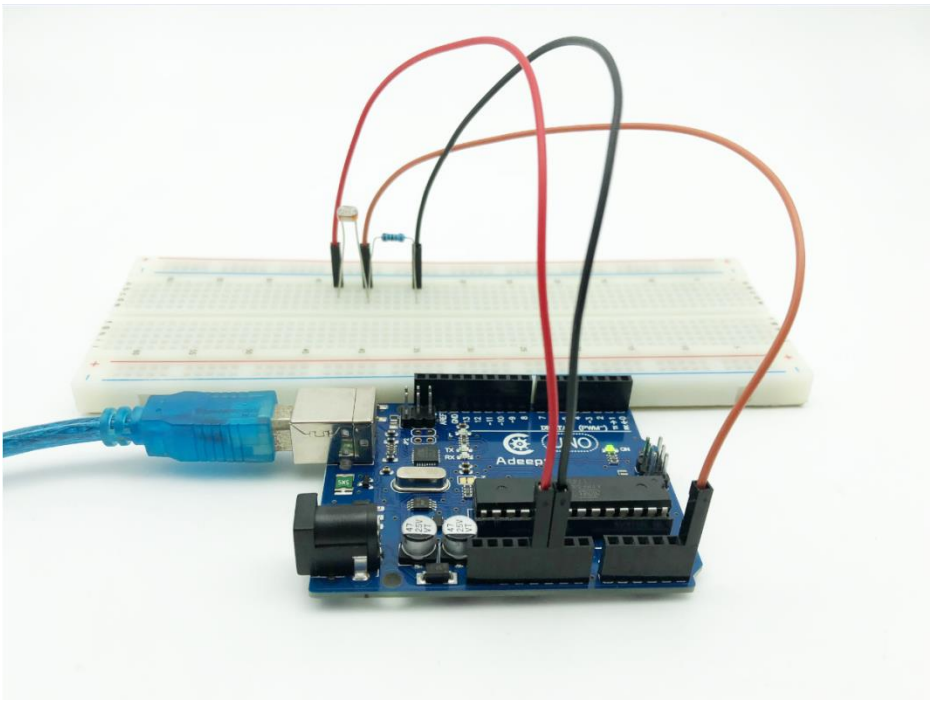
Experimental code

Refer to Project 7.1 code folder

Experimental result:

Now, we can see the data of photoresistor （0~1023）on the serial port of the computer.





Experimental conclusion:

After this course, we know the basic use of photoresistor, and next we will learn more usage about photoresistor.

Project 7.2 Photoresistor controls LED

Experimental objective：：

In this experiment, we will learn how to use photoresistor to control the brightness of LED. The brightness of LED will change along with the value of the photoresistor, the greater the intensity of the light, the brighter the LED lamp is, and the weaker the intensity of the light, the darker the LED lamp is.

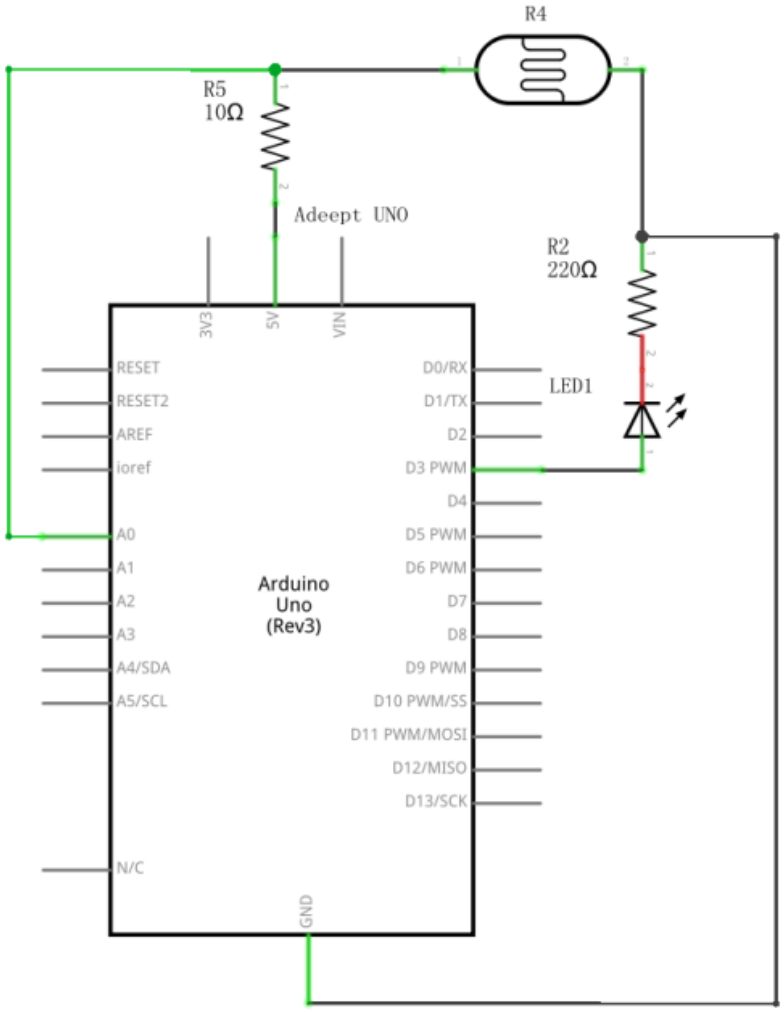
Required materials:

- 1* Arduino UNO
- 1* USB Cable
- 1* 10kΩ photoresistor
- 1* 220Ω Resistor
- 1* 10kΩ Resistor
- 1* LED
- 1* Breadboard

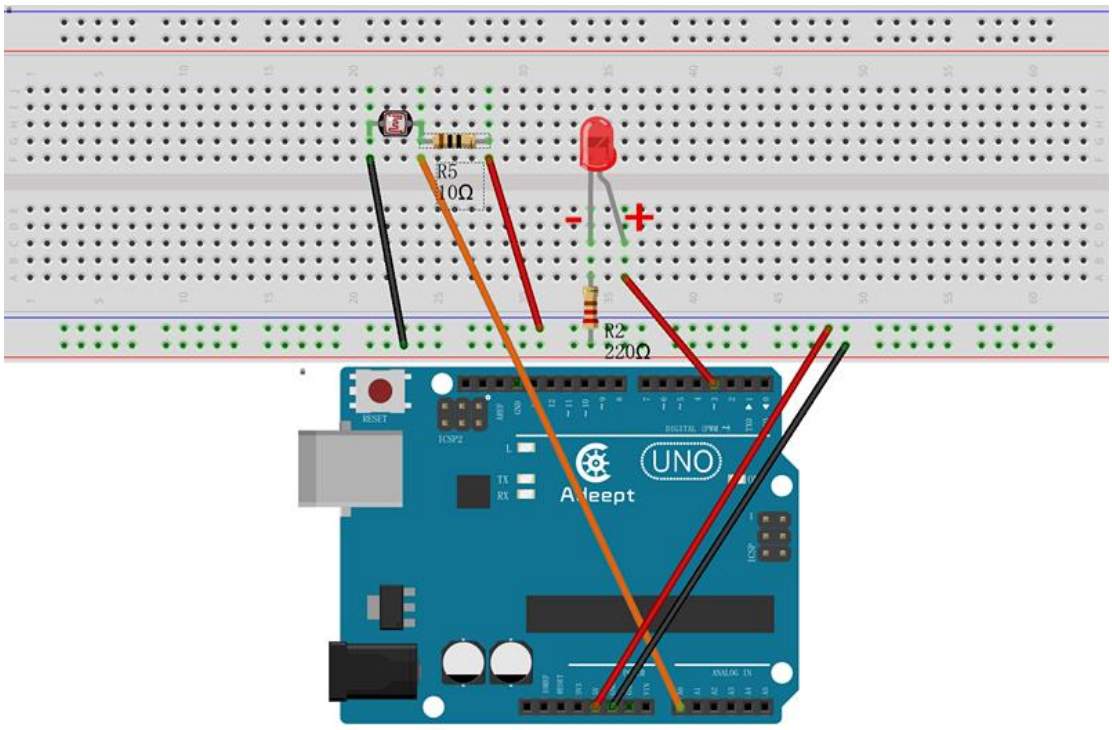
Operating steps:

1. When the power is off, we integrate all the required materials according to the schematic diagram or connection diagram.

Schematic diagram



connection diagram



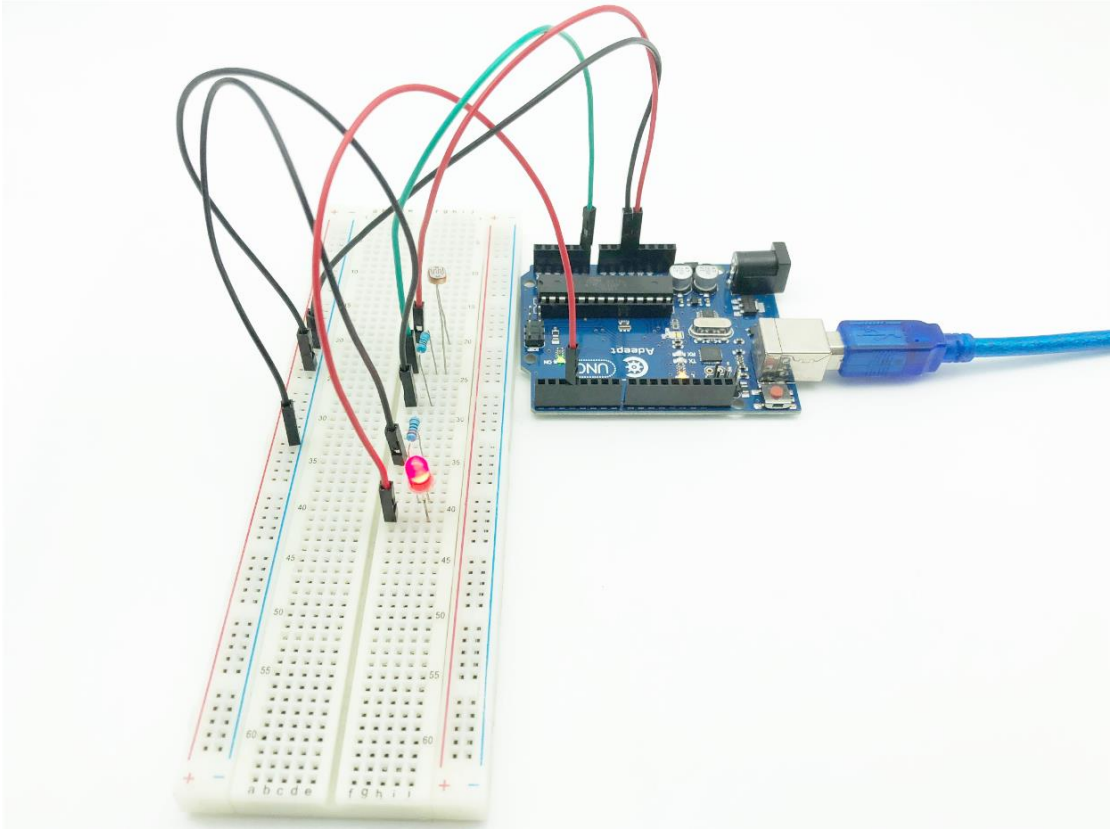
2. Compile the experimental code and download it to arduino UNO R3

Experimental code

Refer to Project 7.2 code folder.

Experimental result:

Now we can see that when we cover up the photoresistor with hand, the color of LED lamp will change, too.



Experimental conclusion:

After this course, we know how to use photoresistor to control the brightness of the LED lamp.

Project 7.3 Thermistor

Experimental objective:

In this experiment, we start to learn how to use thermistor.

Required materials:

- 1* Arduino UNO
- 1* USB Cable
- 1* 10KΩ Resistor
- 1* Thermistor sensor
- 1* Breadboard
- Several Jumper Wires

Operating principle:

A thermistor is a type of resistor whose resistance varies significantly with temperature, more than in standard resistors. We are using MF52 NTC thermistor type. BTC thermistor is usually used as a temperature sensor.
MF52 thermistor key parameters:

B-parameter : 3470.

25℃ resistor : 10KΩ.

The relationship between the resistance of thermistor and temperature is as follows:

$$R_{thermistor} = R * e^{\left(B*\left(\frac{1}{T_1}-\frac{1}{T_2}\right)\right)}$$

$R_{thermistor}$: The resistance of thermistor at temperature T1

R : The nominal resistance of thermistor at room temperature T2;

e : 2.718281828459 ;

B : It is one of the important parameters of thermistor;

T_1 : The Kelvin temperature that you want to measure.

T_2 : At the condition of room temperature 25 ℃ (298.15K), the standard resistance of MF52 thermistor is 10K;

Kelvin temperature = 273.15 (absolute temperature) + degrees Celsius;

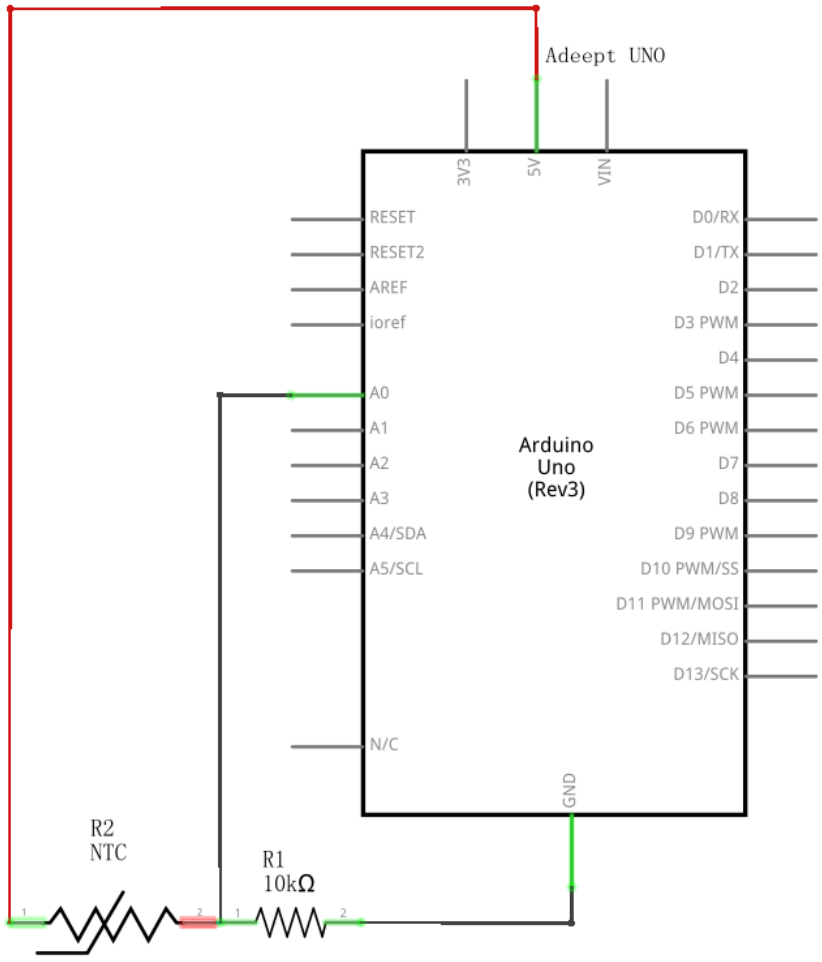
After transforming the above equation, we can get to the following formula:

$$T_1 = \frac{B}{\left(ln\left(\frac{R_{thermistor}}{R}\right) + \frac{B}{T_2}\right)}$$

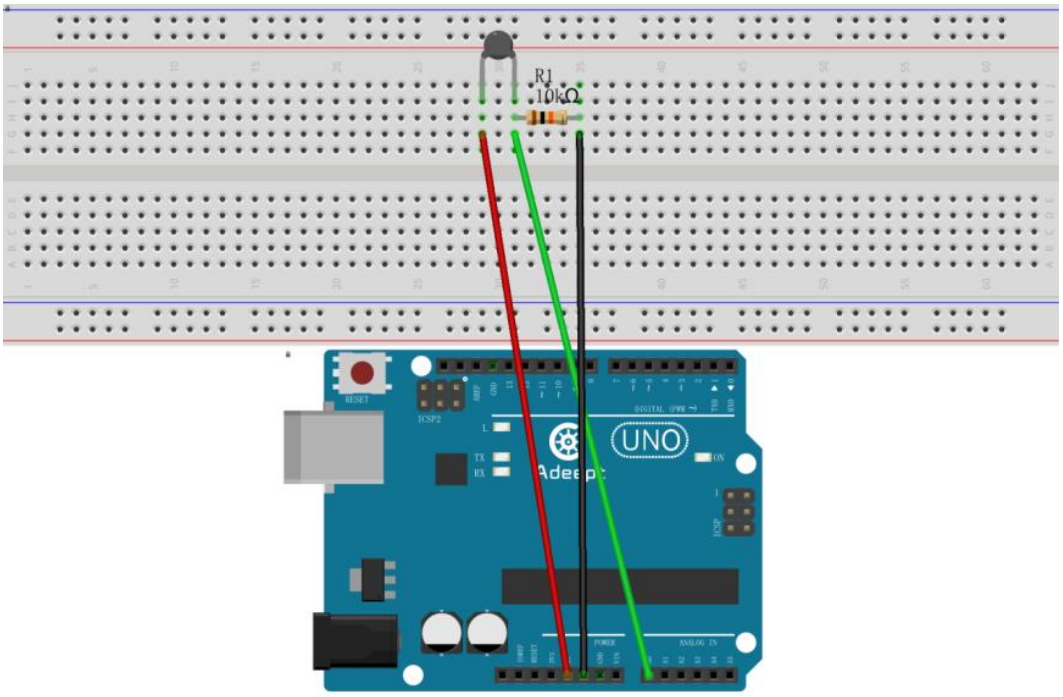
Operating steps：

1. When the power is off, we integrate all the required materials according to the schematic diagram or connection diagram.

Schematic diagram



connection diagram



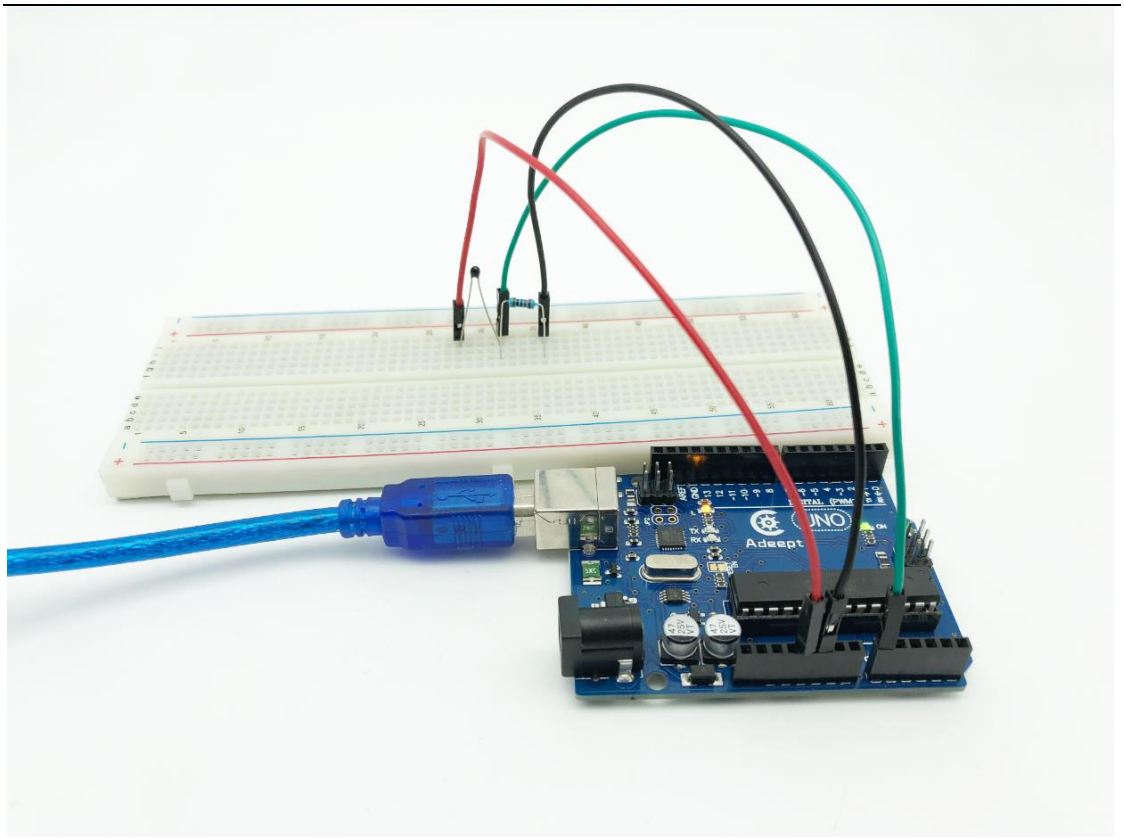
2. Compile the experimental code and download it to arduino UNO R3

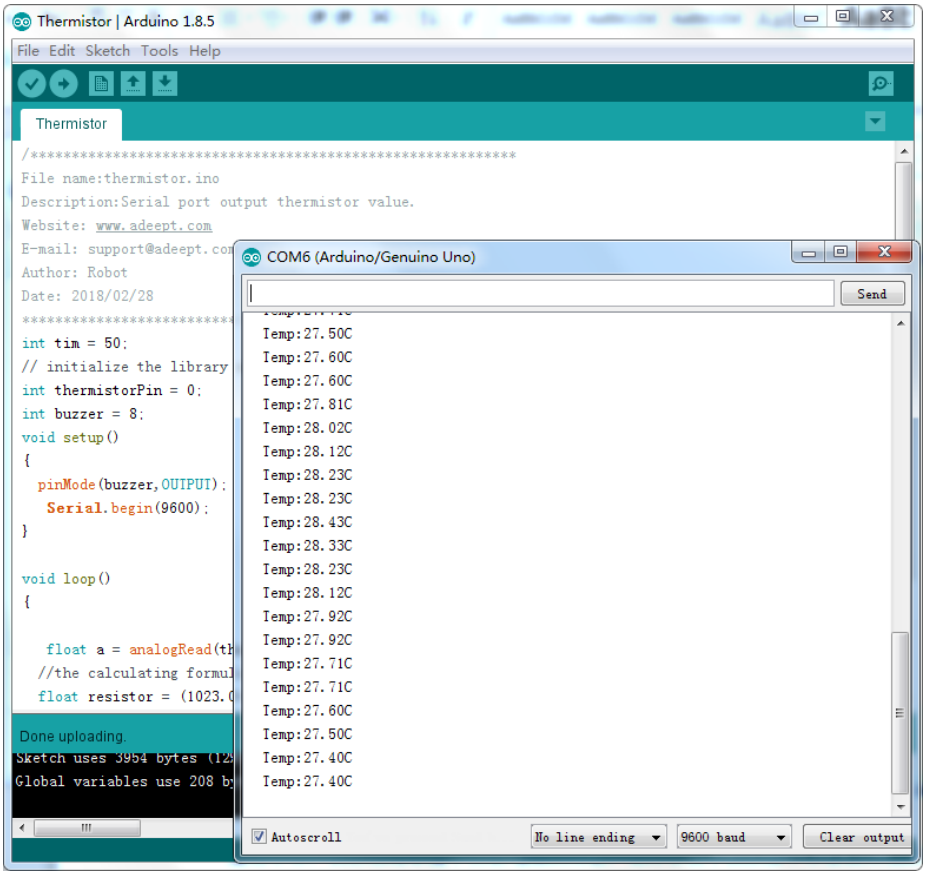
Experimental code

Refer to Project 7.3 code folder

Experimental result:

Now we can see that the value of thermistor has been displayed on the serial port.




Thermistor | Arduino 1.8.5
File Edit Sketch Tools Help
Thermistor
/***
File name:thermistor.ino
Description:Serial port output thermistor value.
Website: www.adeept.com
E-mail: support@adeept.com
Author: Robot
Date: 2018/02/28
****
int tim = 50;
// initialize the library
int thermistorPin = 0;
int buzzer = 8;
void setup()
{
pinMode(buzzer,OUTPUT);
Serial.begin(9600);
}
void loop()
{
float a = analogRead(th
//the calculating formul
float resistor = (1023.0
Done uploading.
Sketch uses 3954 bytes (12
Global variables use 208 by
COM6 (Arduino/Genuino Uno)
Send
Temp:27.50C
Temp:27.60C
Temp:27.60C
Temp:27.81C
Temp:28.02C
Temp:28.12C
Temp:28.23C
Temp:28.23C
Temp:28.43C
Temp:28.33C
Temp:28.23C
Temp:28.12C
Temp:27.92C
Temp:27.92C
Temp:27.71C
Temp:27.71C
Temp:27.60C
Temp:27.50C
Temp:27.40C
Temp:27.40C
Autoscroll
No line ending
9600 baud
Clear output

Experimental conclusion:

After this course, we know how to use the thermistor.

Project 7.4 Temperature alarm

Experimental objective:

In this experiment, we start to do the series experiment of the temperature sensor and buzzer

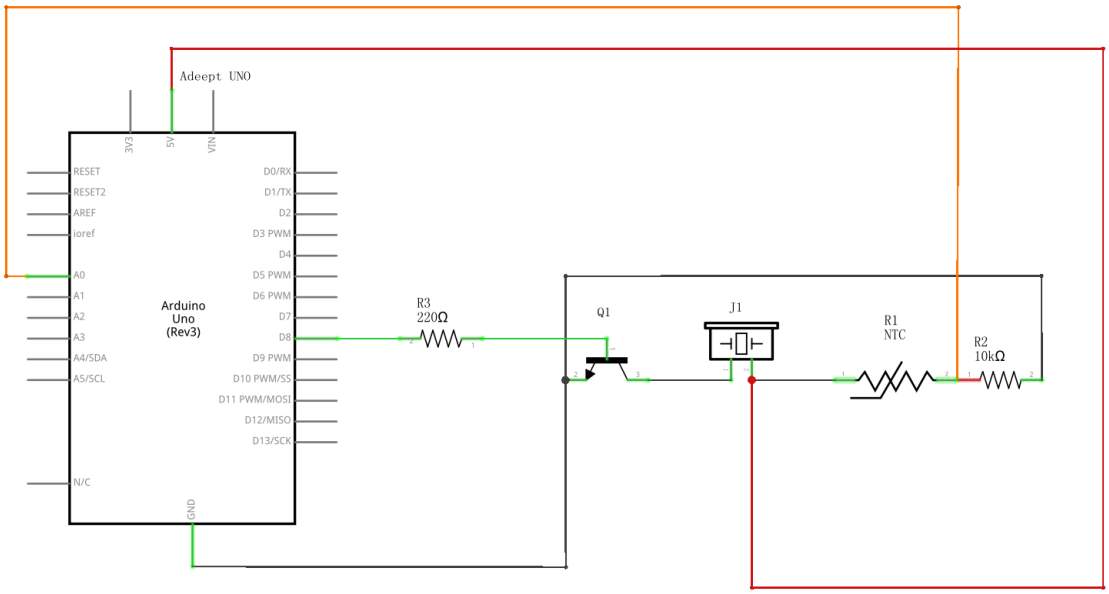
Required materials:

- 1* Arduino UNO
- 1* USB Cable
- 1* temperature sensor
- 1* buzzer
- 1* Breadboard
- 1* 10kΩResistor
- 1* S8050 (NPN)
- Several Jumper Wires

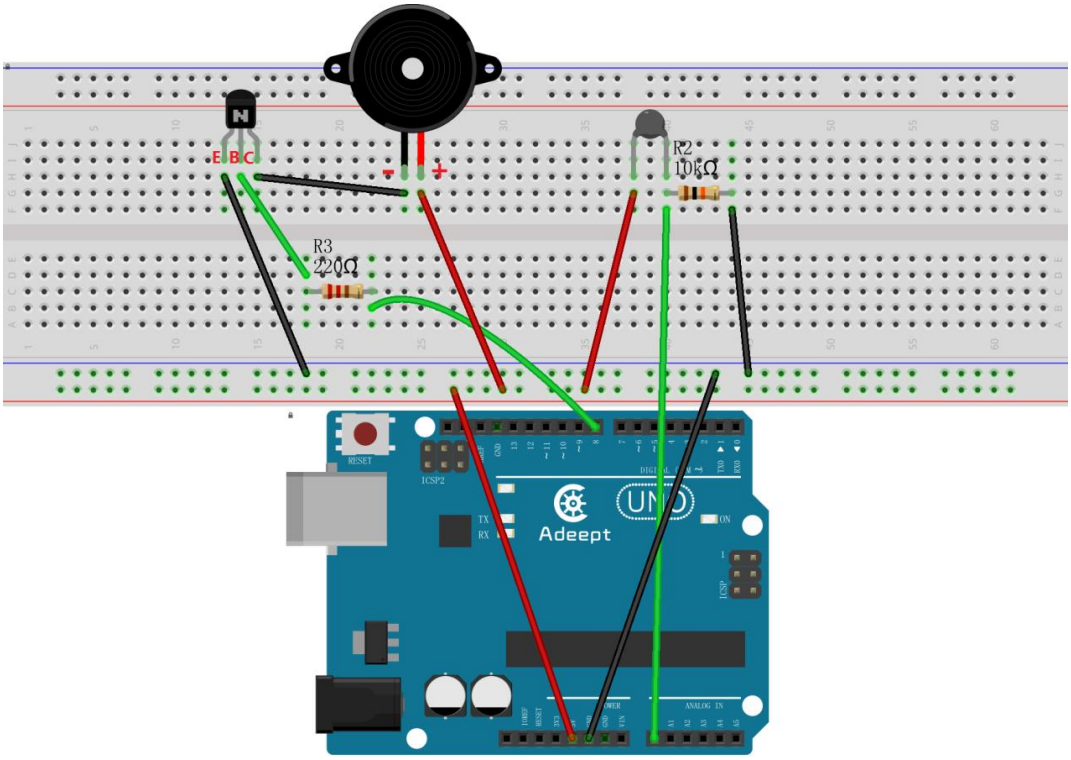
Operating steps:

1. When the power is off, we integrate all the required materials according to the schematic diagram or connection diagram.

Schematic diagram



connection diagram



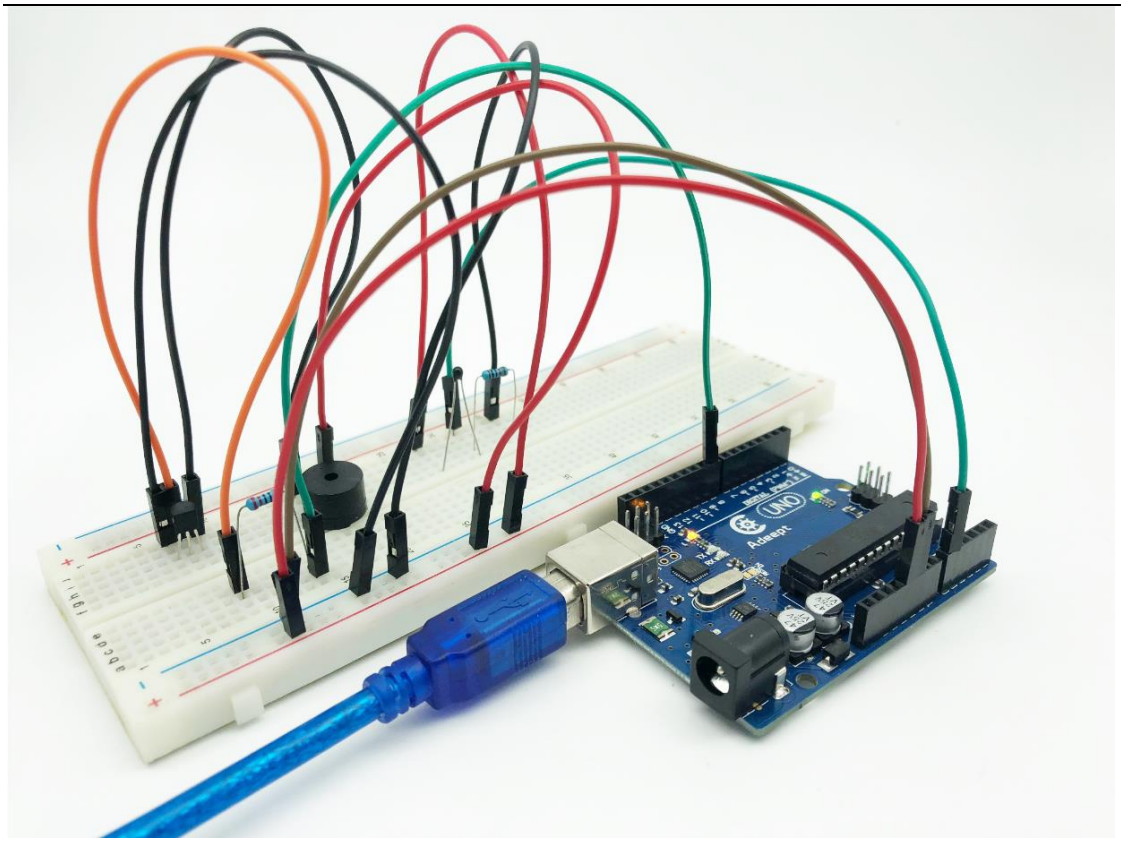
2. Compile the experimental code and download it to arduino UNO R3

Experimental code

Refer to Project 7.4 code folder

Experimental result:

Now, we put our hands on the thermistor, when it reads the temperature that is more than 30℃,the buzzer will alarm, if not, it doesn't alarm.



Experimental conclusion:

After this course, we know how to make a temperature alarm; you can think imaginatively to do the relevant and more interesting experiments.

Project 7.5 Electronic organ

Experimental objective：

In this experiment, we will learn how to make a simple electronic organ. When we press the button, the buzzer can generate sound with different HZ, thus we can make a simple numbered musical notation.

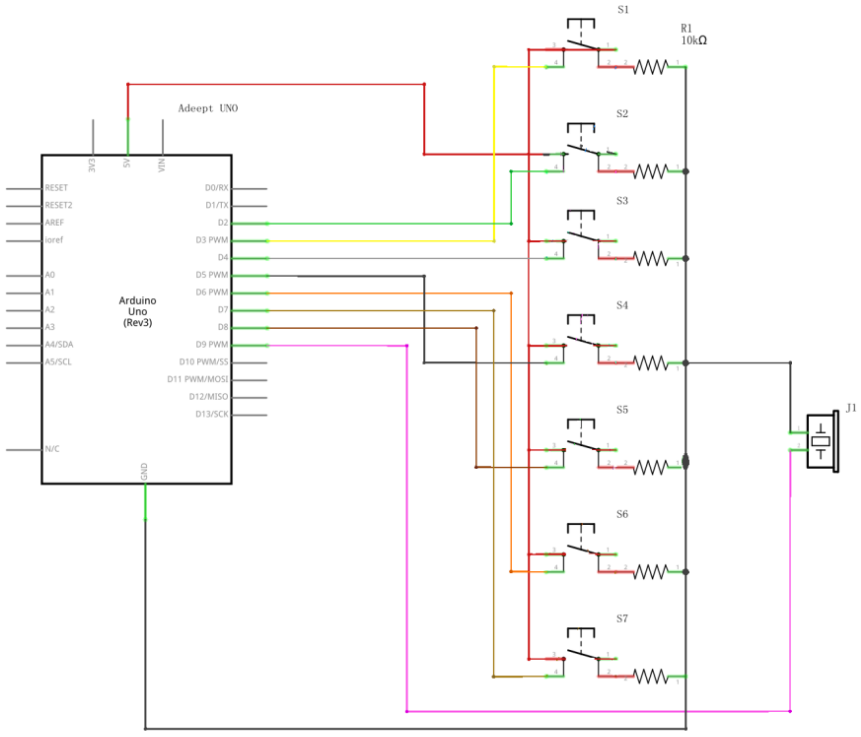
Required materials:

- 1* Arduino UNO
- 1* USB Cable
- 7* 10KΩ Resistor
- 7* button
- 1* passive buzzer
- 1* Breadboard
- Several Jumper Wires

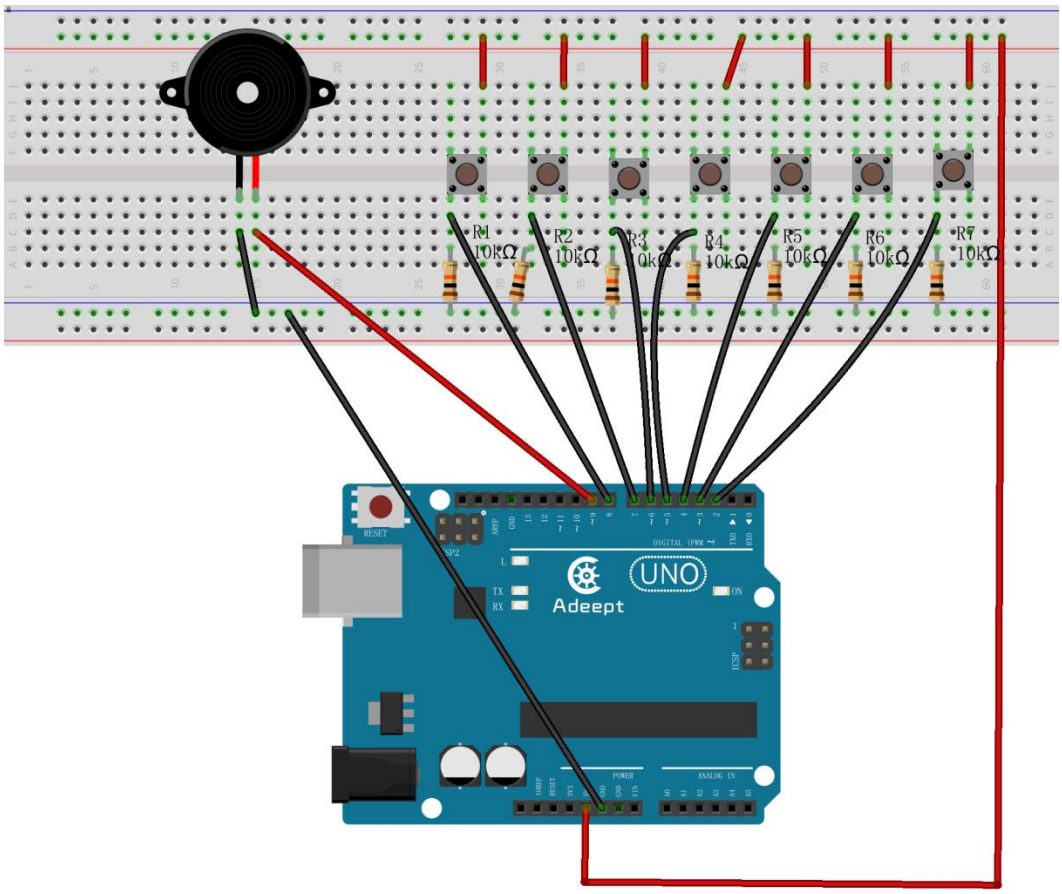
Operating steps：

1. When the power is off, we integrate all the required materials according to the schematic diagram or connection diagram.

Schematic diagram



connection diagram



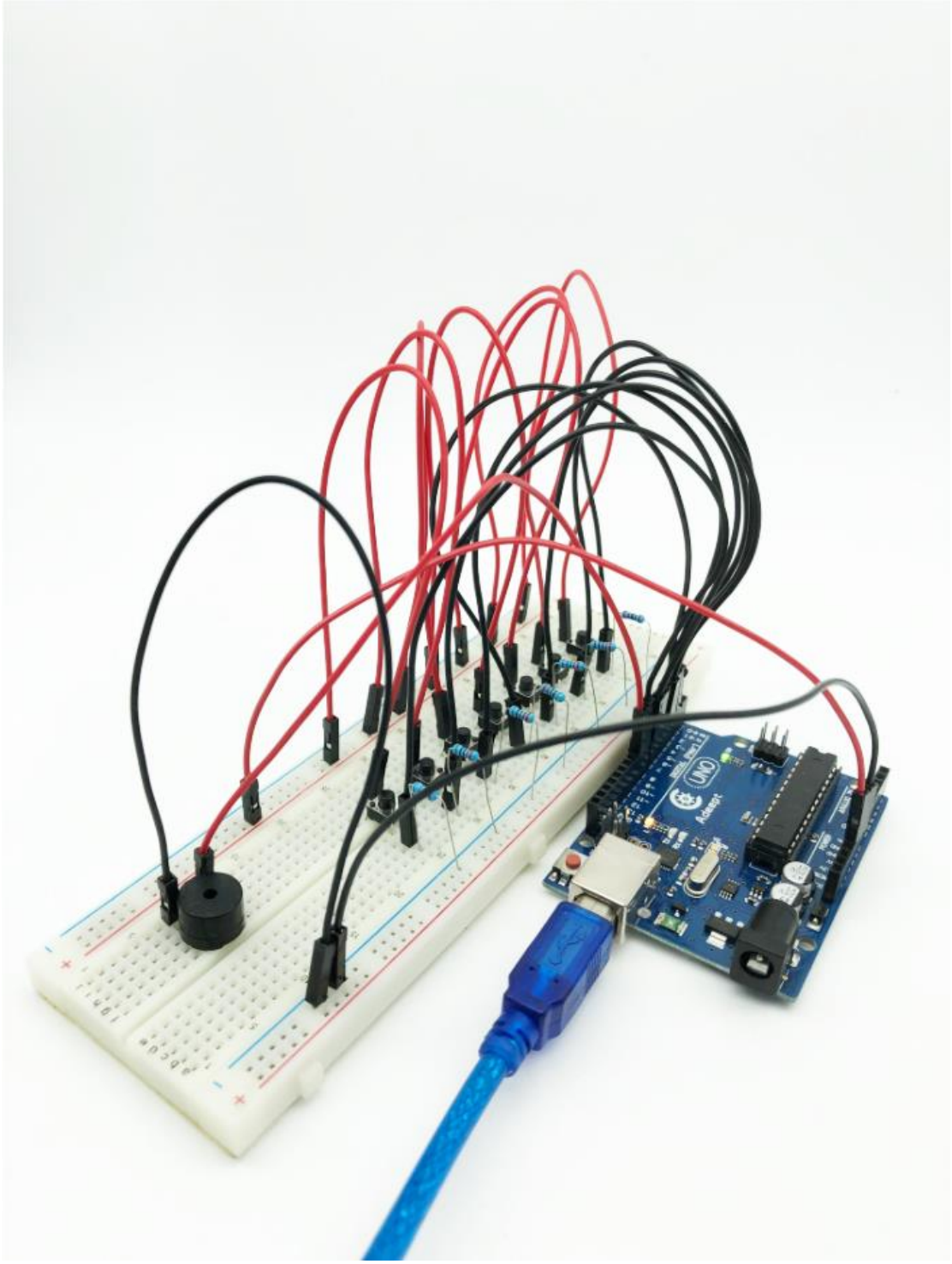
2. Compile the experimental code and download it to arduino UNO R3

Experimental code

Refer to Project 7.5 code folder

Experimental result:

Now, when we press the button in sequence, we can hear that the buzzer produces seven kinds of tones.



Experimental conclusion:

After this course, we have learned how to make the button match the buzzer and make a simple electronic organ, and we can use it to play simple songs.

Chapter 8 LCD1602

Project 8.1 LCD1602

Experimental objective：

In this experiment, we will start to learn how to use LCD 1602.

Required materials:

- 1* Arduino UNO
- 1* USB cable
- 1* LCD1602
- 1* 10KΩ Potentiometer
- 1* Breadboard
- Several Jumper wires

Operating principle：

LCD1602 is a kind of character LCD display. The LCD has a parallel interface, meaning that the microcontroller has to manipulate several interface pins at once to control the display. The interface consists of the following pins:

● A register select (RS) pin that controls where in the LCD's memory you're writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, which is where the LCD's controller looks for instructions on what to do next.

● A Read/Write (R/W) pin that selects reading mode or writing mode

● An Enable pin that enables writing to the registers

● 8 data pins (D0-D7). The state of these pins (high or low) is the bits that you're writing to a register when you write or the values when you read.

● There's also a display contrast pin (Vo), power supply pins (+5V and Gnd) and LED Backlight (Bklt+ and BKlt-) pins that you can use to power the LCD, control the display contrast, and turn on or off the LED backlight respectively.

The process of controlling the display involves putting the data that form the image of what you want to display into the data registers, then putting instructions in the instruction register. The LiquidCrystal Library simplifies this

for you so you don't need to know the low-level instructions.

The Hitachi-compatible LCDs can be controlled in two modes: 4-bit or 8-bit. The 4-bit mode requires seven I/O pins from the Arduino, while the 8-bit mode requires 11 pins. For displaying text on the screen, you can do most everything in 4-bit mode, for example shows how to control a 2x16 LCD in 4-bit mode.

A potentiometer, informally a pot, is a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider. If only two terminals are used, one end and the wiper, it acts as a variable resistor or rheostat. Rotating the potentiometer can adjust the contrast of LCD.

Required functions:

●begin()

It specifies the dimensions (width and height) of the display.

Syntax

lcd.begin (cols, rows)

Parameters

lCD: a variable of type Liquid Crystal

Cols: the number of columns that the display has

Rows: the number of rows that the display has

●setCursor()

Positions the LCD cursor; that is, set the location at which subsequent text written to the LCD will be displayed.

Syntax

lcd.setCursor (col, row)

Parameters

lcd: a variable of type LiquidCrystal

Col: the column at which to position the cursor (with 0 being the first column)

Row: the row at which to position the cursor (with 0 being the first row)

●scrollDisplayLeft()

Scroll the contents of the display (text and cursor) one space to the left.

Syntax

lcd.scrollDisplayLeft ()

Parameters

LCD: a variable of type Liquid Crystal

Example

scrollDisplayLeft () and scrollDisplayRight ()

See also

scrollDisplayRight ()

● print()

Print text to the LCD.

Syntax

lcd.print (data)

lcd.print(data, BASE)

Parameters

lcd: a variable of type LiquidCrystal

Data: the data to print (char, byte, int, long, or string)

BASE (optional): the base in which to print numbers: BIN for binary (base 2), DEC for decimal (base 10), OCT for octal (base 8), HEX for hexadecimal (base 16).

Returns

Byte

Print () will return the number of bytes written, though reading that number is optional

● clear()

Clears the LCD screen and positions the cursor in the upper-left corner.

Syntax

lcd.clear ()

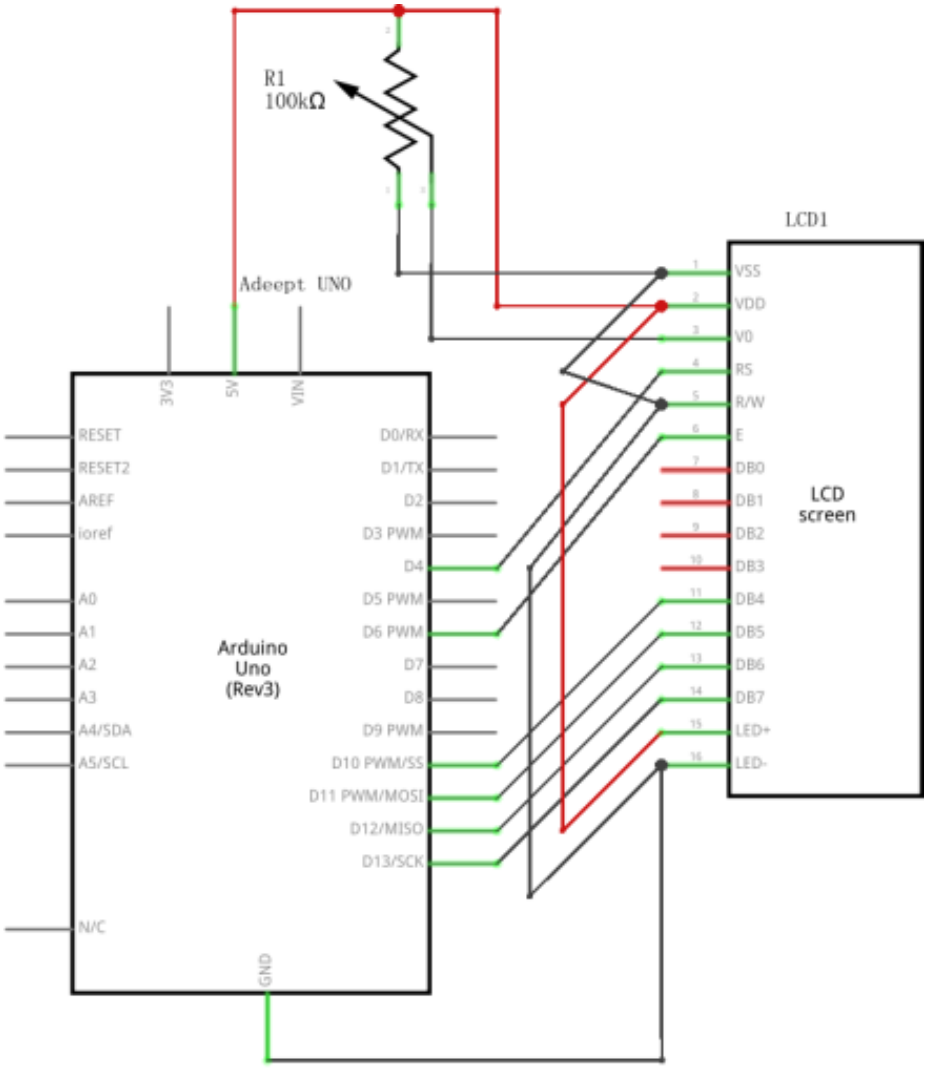
Parameters

lcd: a variable of type LiquidCrystal

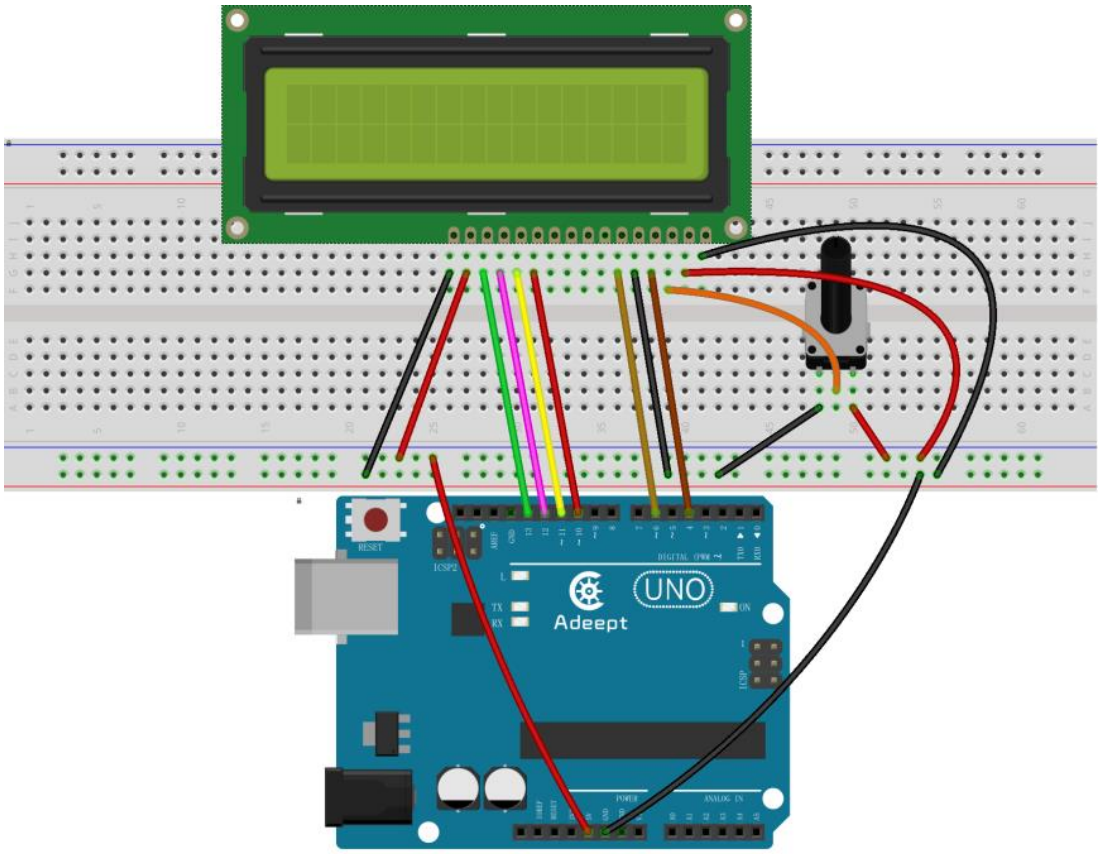
Operating steps:

1. When the power is off, we integrate all the required materials according to the schematic diagram or connection diagram.

Schematic diagram



connection diagram



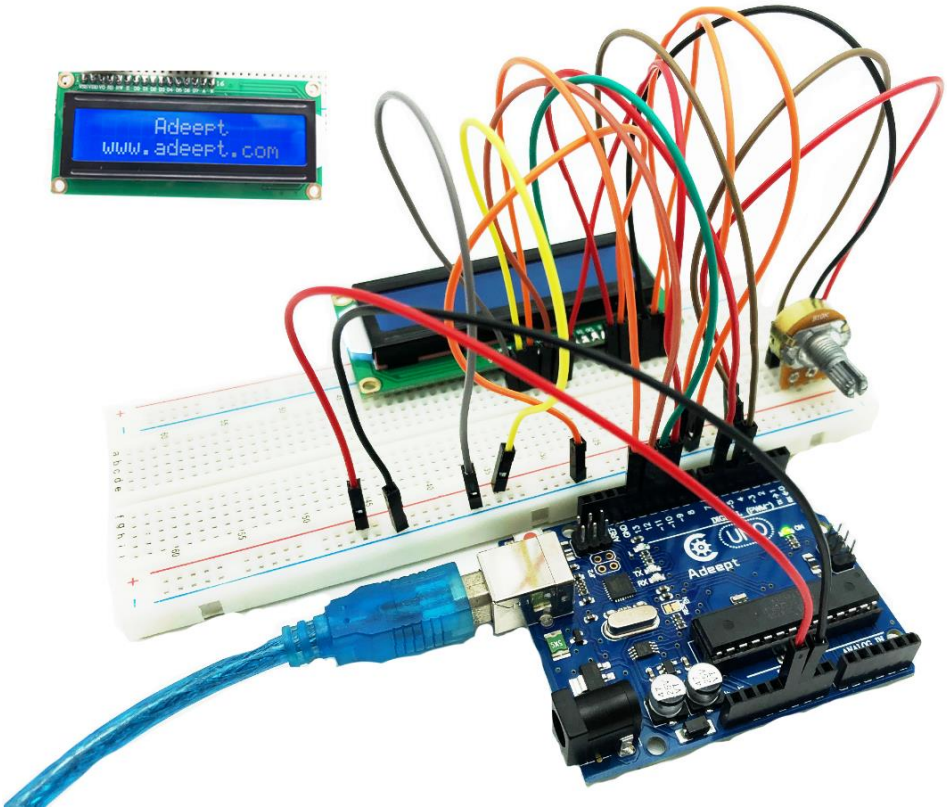
2. Compile the experimental code and download it to arduino UNO R3

Experimental code

Refer to Project 8.1 code folder

Experimental result：

Now we can see what we have input on LCD screen: Adeept hello geeks! www.adeept.com。



Experimental conclusion:

After this course, we know how to use LCD and adjust the backlight, later, we can transfer different information to LCD and display it.

Project 8.2 IIC Interface module

Experimental objective：

In this experiment, we will start to do series experiment of LCD1602 and IIC interface module.

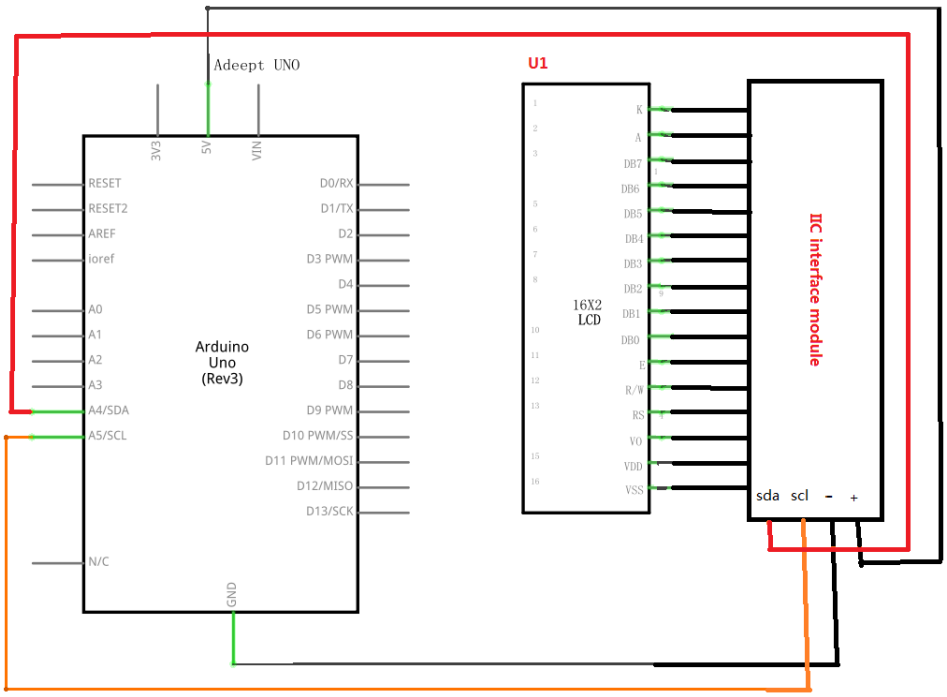
Required materials:

- 1* Arduino UNO
- 1* USB cable
- 1* LCD1602
- 1* IIC interface module
- 1* Breadboard
- 1* 4-Pin wires

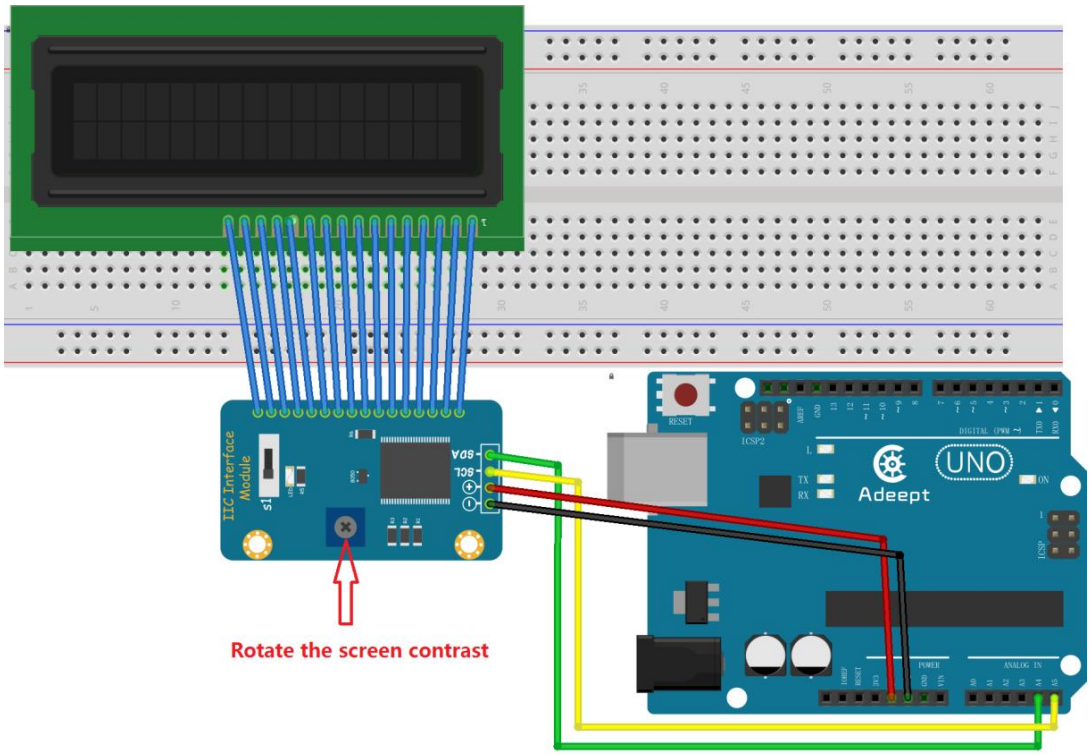
Operating steps :

1. When the power is off, we integrate all the required materials according to the schematic diagram or connection diagram. Connecting IIC interface module and LCD display screen and directly display the content of LCD.

Schematic diagram



connection diagram



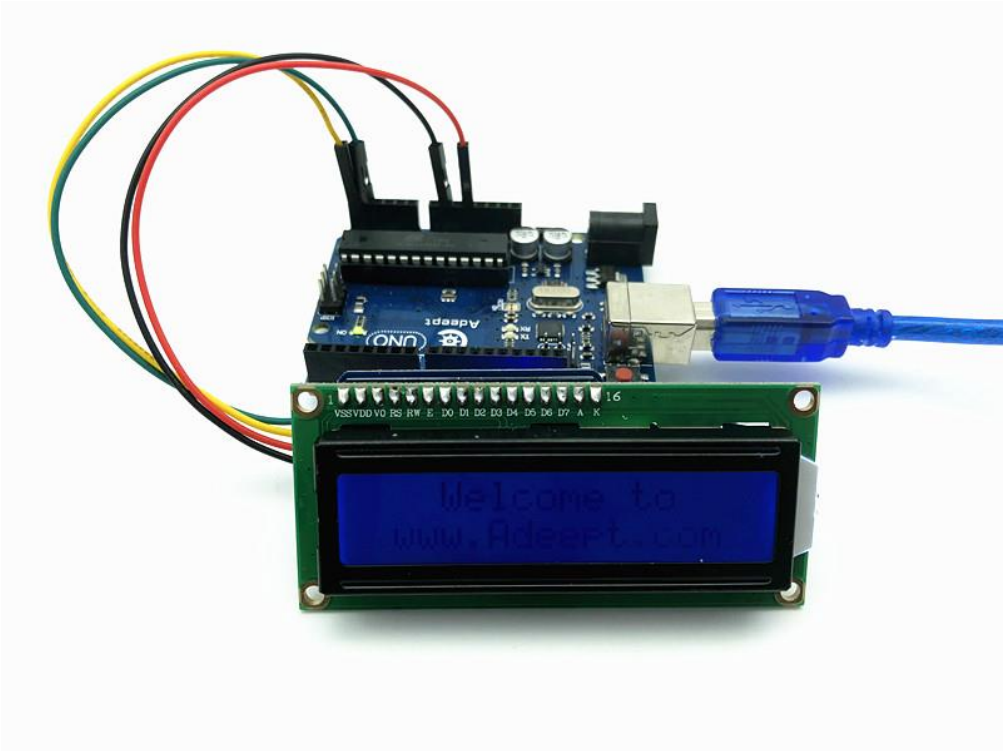
2. Compile the experimental code and download it to arduino UNO R3

Experimental code

Refer to Project 8.2 code folder.

Experimental result:

Now we can see the LCD displays : Welcome to www.Adeept.com".



Experimental conclusion:

After this course, we know how to connect IIC interface module with LCD screen and displays: welcome to www.adeept.com.

Project 8.3 LCD1602 Display Brightness

Experimental objective:

In this experiment, we start to learn how to display the data that Photoresistor has read on the LCD display screen with IIC interface module.

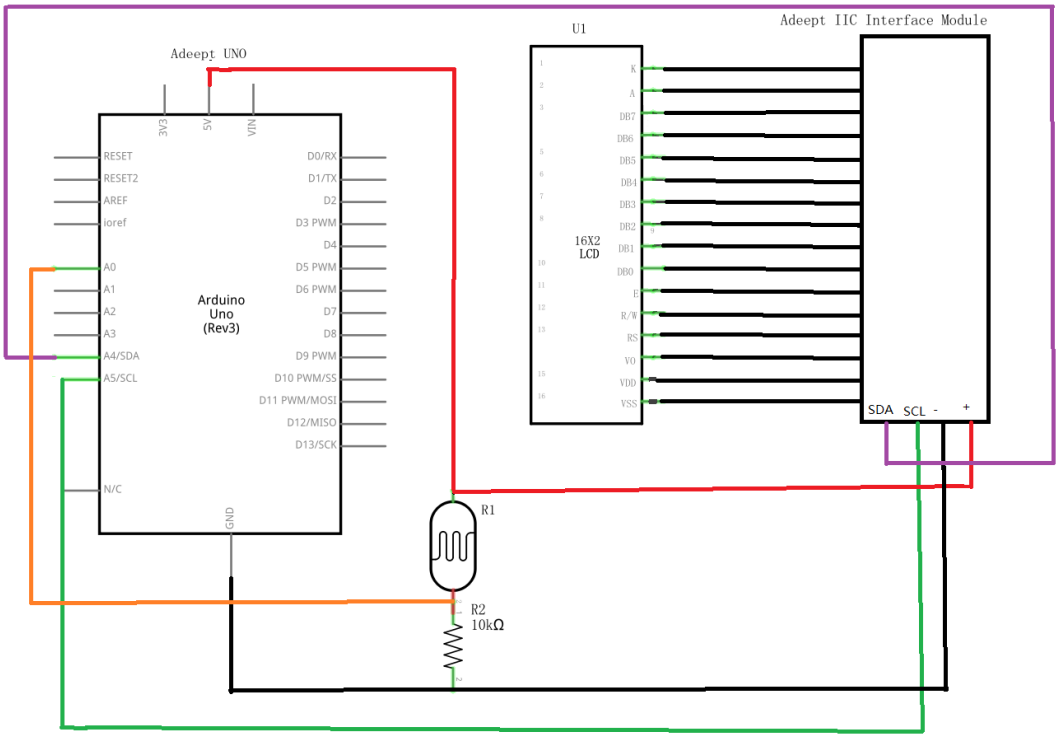
Required materials:

- 1* Arduino UNO
- 1* USB Cable
- 1* Photoresistor
- 1* 10kΩ resistor
- 1* LCD1602 module
- 1* IIC interface module
- 1* Breadboard
- 1* 4-Pin wires
- Several Jumper wires

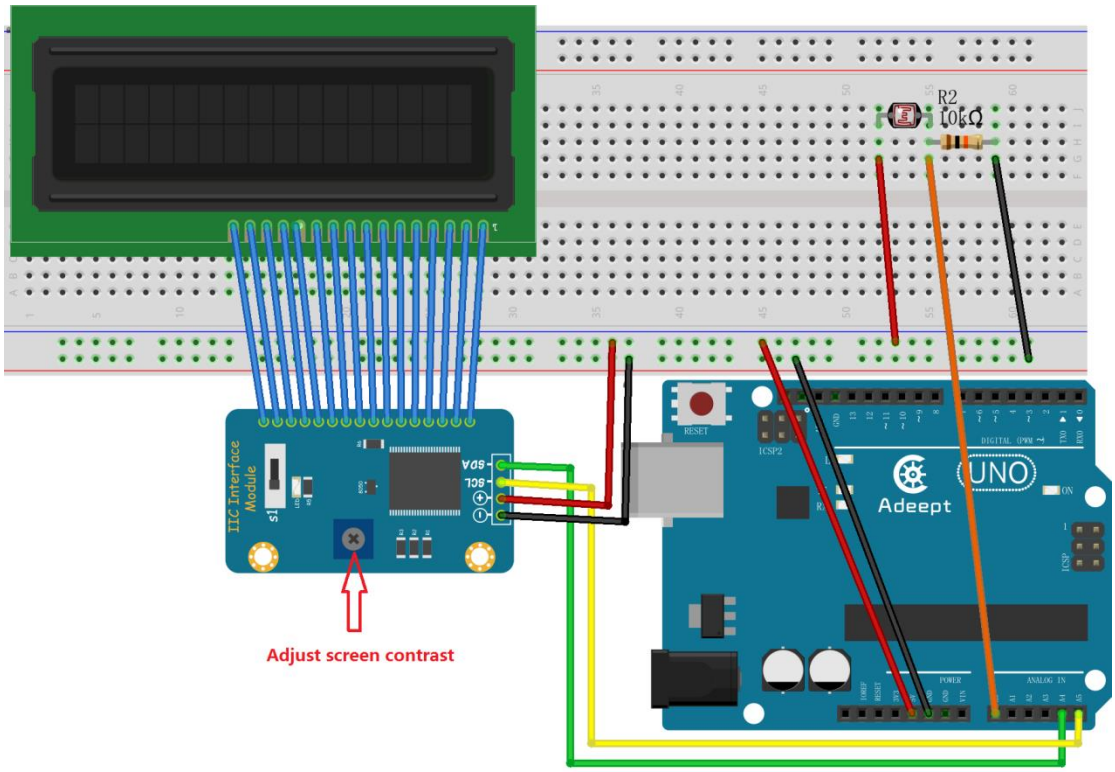
Operating steps：

1. When the power is off, we integrate all the required materials according to the schematic diagram or connection diagram.

Schematic diagram



connection diagram



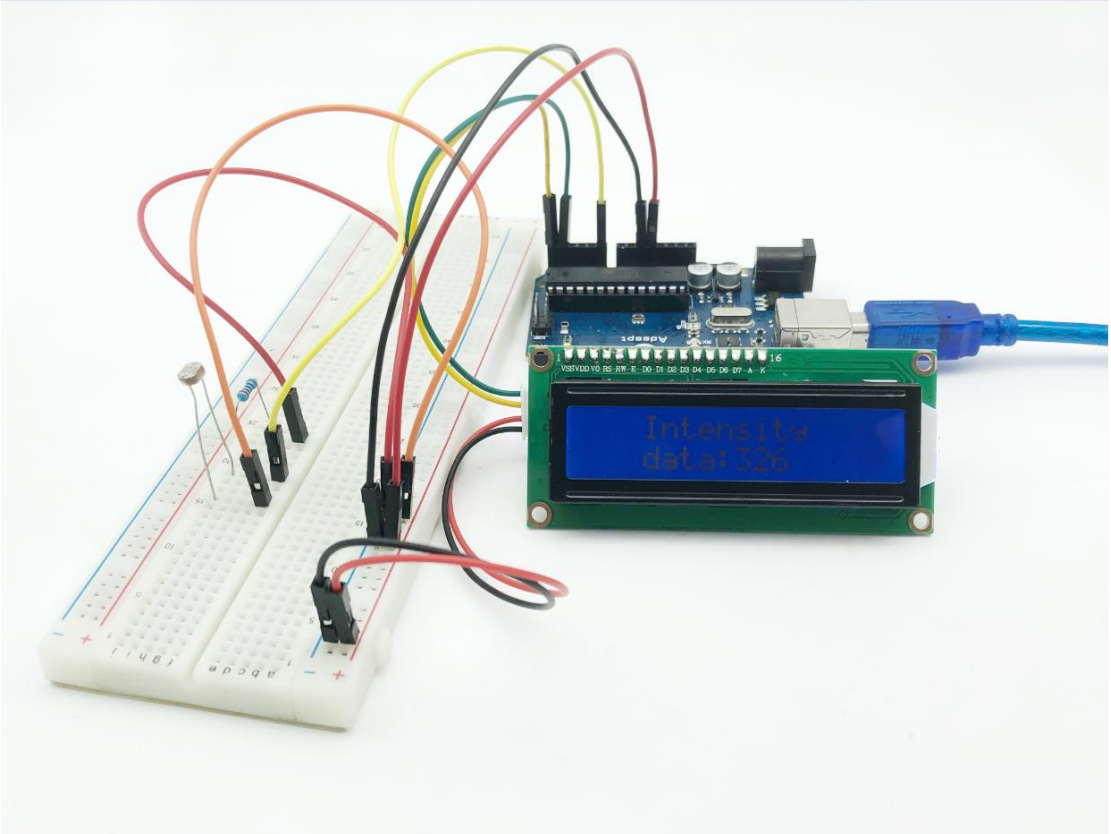
2. Compile the experimental code and download it to arduino UNO R3

Experimental code

Refer to Project 8.3 code folder.

Experimental result：

Now, we can see the data of the Photoresistor displayed on LCD screen.



Experimental conclusion:

After this course, we know how to display the data of photoresistor on the LCD screen with IIC interface module.

Chapter 9 Frequency meter and DC motor

Project 9.1 Frequency meter

Experimental objective:

In this experiment, we will use Arduino UNO to make a simple frequency meter

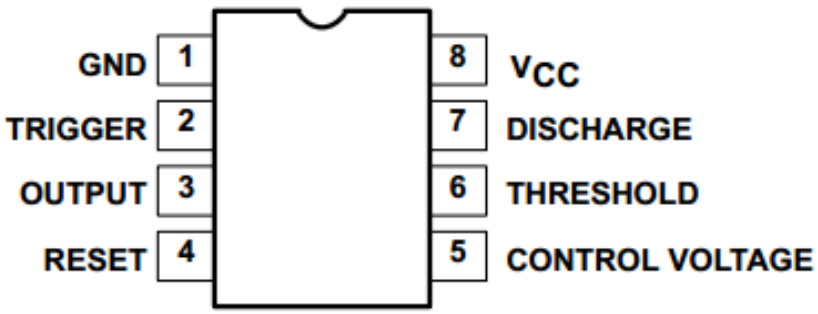
Required materials:

- 1* Arduino UNO
- 1* USB Cable
- 1* NE555 Timer
- 1* 10KΩ Resistor
- 1* 10KΩ Potentiometer
- 2* 104 Capacitor
- 1* Breadboard
- Several Jumper Wires

Operating principle:

NE555

The 555 integrated circuit is originally used as a timer, and that is why it is called 555 timer or 555 time-base circuit. It is widely used in various electronic products because of its reliability, convenience and low price. There are dozens of components in the 555 integrated circuit, such as divider, comparator, basic R-S trigger, discharge tube, buffer and so on. It is a complex circuit and a hybrid composed of analog and digital circuit.



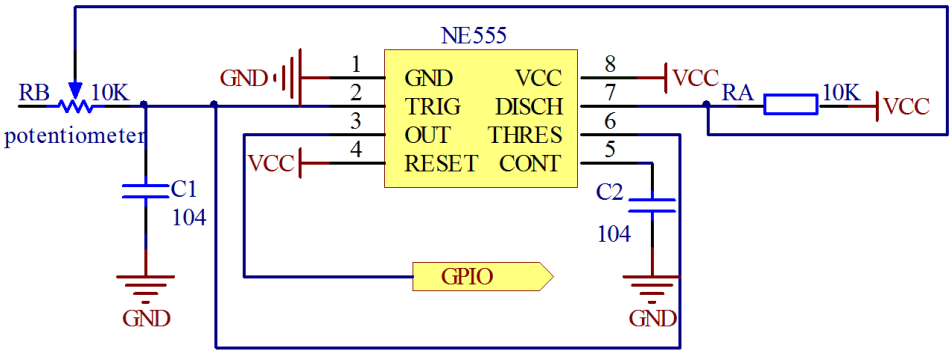
As shown in the above picture, the 555 integrated circuit is dual in-line with 8

pins package (DIP). Thereinto:

Pin 6 is the THRESHOLD for the input of upper comparator;
Pin 2 is TRIGGER for the input of lower comparator;
Pin 3 is the OUTPUT having two states of 0 and 1 decided by the input electrical level;
Pin 7, the DISCHARGE which has two states of suspension and ground connection also decided by input, is the output of the internal discharge tube;
Pin 4 is the RESET that outputs low level when supplied low voltage level;
Pin 5 is the CONTROL VOLTAGE that can change the upper and lower level triggers value;
Pin 8 (Vcc) is the power supply;

Pin 1(GND) is the ground.

The circuit schematic diagram used in the experiment is shown in below:



The circuit can generate a square wave signal that the frequency is adjustable. The frequency can be calculated by the formula:

$$Frequency \approx \frac{1.44}{(R_A + 2R_B) * C}$$

Required functions:

2. Key functions :

● pulseIn()

Read a pulse (either HIGH or LOW) on a pin. For example, if value is HIGH,

pulseIn() waits for the pin to go HIGH, starts timing, then waits for the pin to go LOW and stops timing. Return the length of the pulse in microseconds. Gives up and returns 0 if no pulse starts within a specified time out.

The timing of this function has been determined empirically and will probably show errors in longer pulses. Work on pulses from 10 microseconds to 3 minutes in length.

Syntax

pulseIn (pin, value)

pulseIn (pin, value, timeout)

Parameters

Pin: the number of the pin on which you want to read the pulse. (int)

Value: type of pulse to read: either HIGH or LOW. (int)

Timeout (optional): the number of microseconds to wait for the pulse to start; default is one second (unsigned long)

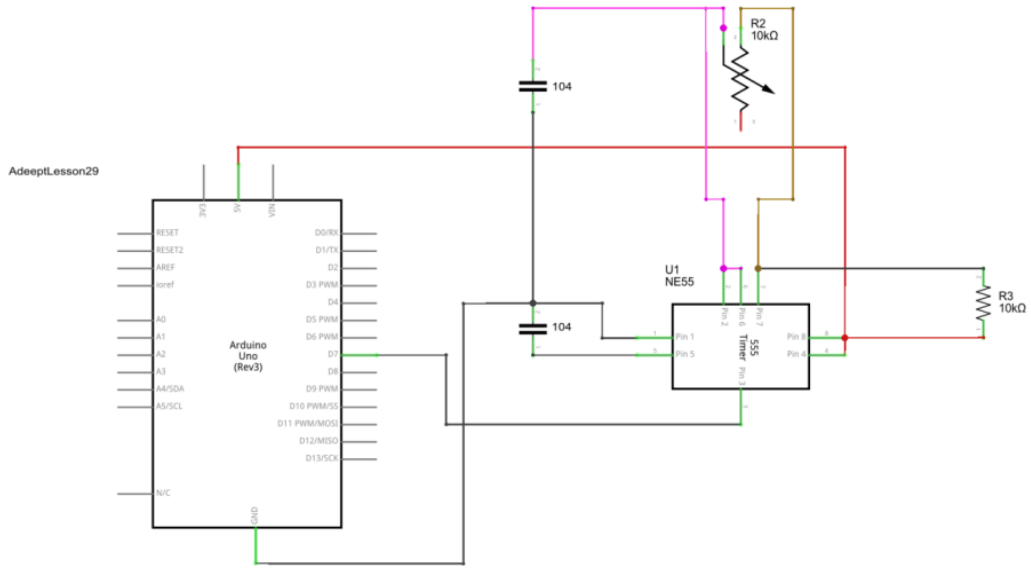
Returns

The length of the pulse (in microseconds) or 0 if no pulse started before the timeout (unsigned long)

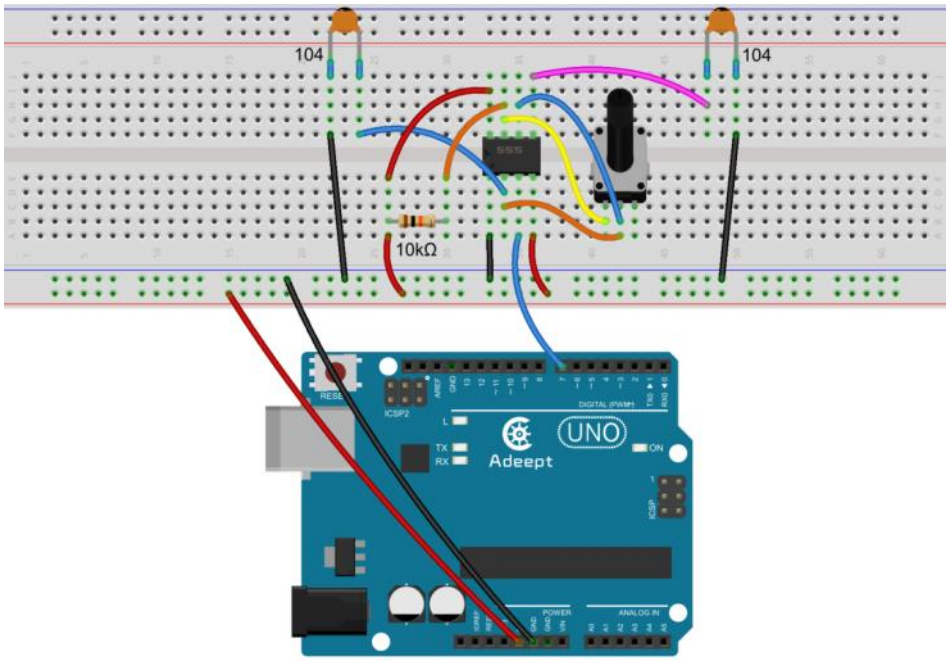
Operating steps:

1. When the power is off, we integrate all the required materials according to the schematic diagram or connection diagram.

Schematic diagram



connection diagram



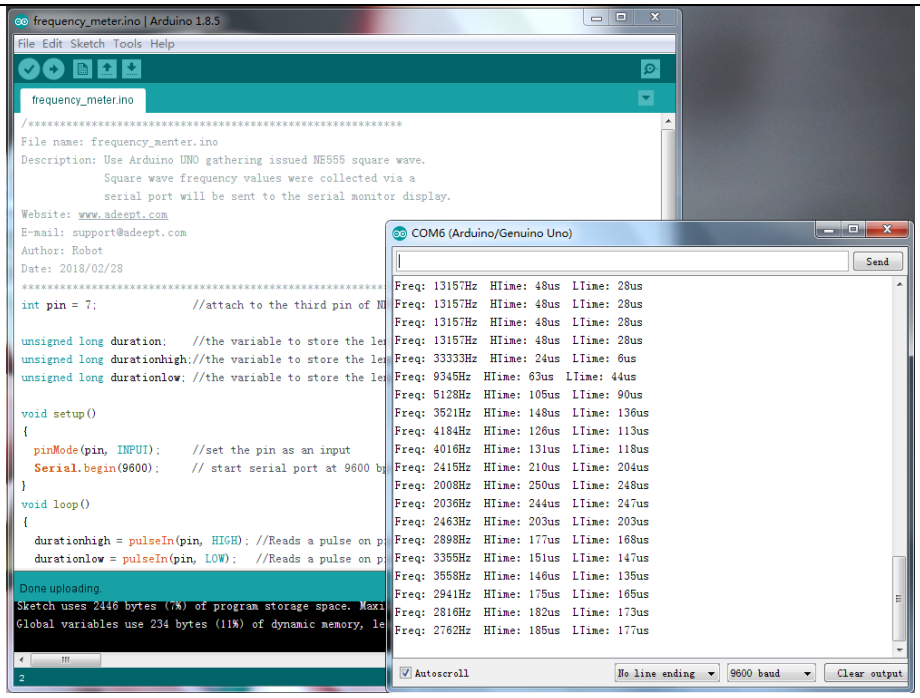
2. Compile the experimental code and download it to arduino UNO R3

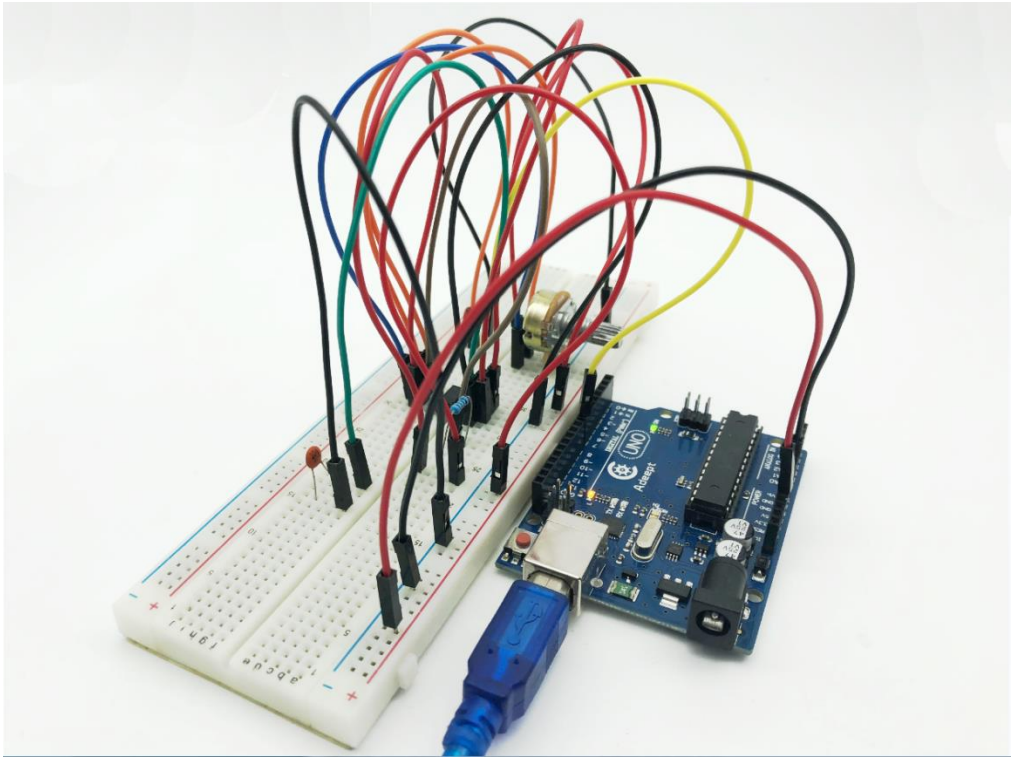
Experimental code

Refer to Project 9.1 code folder

Experimental result:

Now, we can see the frequency data on the serial port, and if we rotate the potentiometer, the frequency data will change, too.


frequency_meter.ino | Arduino 1.8.5
File Edit Sketch Tools Help
frequency_meter.ino
/***
File name: frequency_menter.ino
Description: Use Arduino UNO gathering issued NE555 square wave.
Square wave frequency values were collected via a
serial port will be sent to the serial monitor display.
Website: www.adeept.com
E-mail: support@adeept.com
Author: Robot
Date: 2018/02/28
****
int pin = 7; //attach to the third pin of N
unsigned long duration; //the variable to store the le
unsigned long durationhigh;//the variable to store the le
unsigned long durationlow; //the variable to store the le
void setup()
{
pinMode(pin, INPUT); //set the pin as an input
Serial.begin(9600); // start serial port at 9600 b
}
void loop()
{
durationhigh = pulseIn(pin, HIGH); //Reads a pulse on p
durationlow = pulseIn(pin, LOW); //Reads a pulse on p
Done uploading.
Sketch uses 2446 bytes (7%) of program storage space. Maxi
Global variables use 234 bytes (11%) of dynamic memory, le
2
COM6 (Arduino/Genuino Uno)
Send
Freq: 13157Hz HTime: 48us LTime: 28us
Freq: 13157Hz HTime: 48us LTime: 28us
Freq: 13157Hz HTime: 48us LTime: 28us
Freq: 13157Hz HTime: 48us LTime: 28us
Freq: 33333Hz HTime: 24us LTime: 6us
Freq: 9345Hz HTime: 63us LTime: 44us
Freq: 5128Hz HTime: 105us LTime: 90us
Freq: 3521Hz HTime: 148us LTime: 136us
Freq: 4184Hz HTime: 126us LTime: 113us
Freq: 4016Hz HTime: 131us LTime: 118us
Freq: 2415Hz HTime: 210us LTime: 204us
Freq: 2008Hz HTime: 250us LTime: 248us
Freq: 2036Hz HTime: 244us LTime: 247us
Freq: 2463Hz HTime: 203us LTime: 203us
Freq: 2898Hz HTime: 177us LTime: 168us
Freq: 3355Hz HTime: 151us LTime: 147us
Freq: 3558Hz HTime: 146us LTime: 135us
Freq: 2941Hz HTime: 175us LTime: 165us
Freq: 2816Hz HTime: 182us LTime: 173us
Freq: 2762Hz HTime: 185us LTime: 177us
Autoscroll
No line ending
9600 baud
Clear output



Experimental conclusion:

After this class, we know how to make a simple frequency meter.

Project 9.2 DC motor

Experimental objective:

In this experiment, we will learn how to use Arduino to control the state of DC motor and display the state on LED, the states of DC motor include forward, backward, acceleration, deceleration and stop.

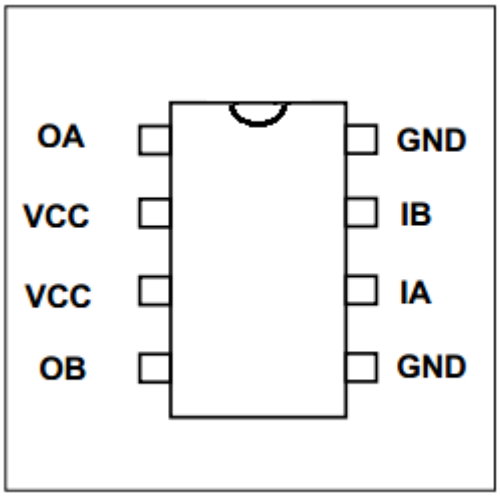
Required materials:

- 1* Arduino UNO
- 1* USB Cable
- 1* L9110 DC Motor Driver
- 1* DC Motor
- 4* LED
- 4* 220Ω Resistor
- 1* 9V Battery Holder
- 1* Breadboard
- Several Jumper Wires

Operating principle:

L9110

L9110 is a driver chip which is used to control and drive motor. The chip has two TTL/CMOS compatible input terminals, and possesses the property of anti-interference: it has high current driving capability, two output terminals that can directly drive DC motor, each output port can provide 750~800mA dynamic current, and its peak current can reach 1.5~2.0A; L9110 is widely applied to various motor drives, such as toy cars, stepper motor, power switches and other electric circuits.



OA, OB: These are used to connect the DC motor.

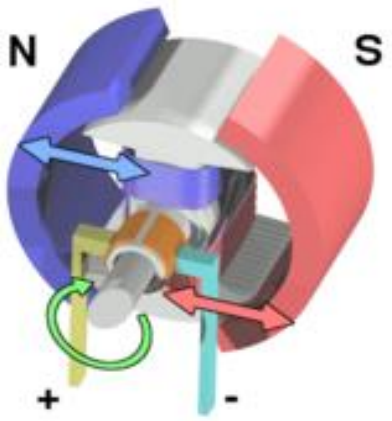
VCC: Power supply (+5V)

GND: The cathode of the power supply (Ground).

IA, IB: The input terminal of drive signal.

DC motor

A DC motor is any of a class of electrical machines that converts direct current electrical power into mechanical power. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic, to periodically change the direction of current flow in part of the motor. Most types produce rotary motion; a linear motor directly produces force and motion in a straight line.



DC motors were the first type widely used, since they could be powered from existing direct-current lighting power distribution systems. A DC motor's speed can be controlled over a wide range, using either a variable supply voltage or by changing the strength of current in its field windings. Small DC motors are used in tools, toys, and appliances. The universal motor can operate on direct current but is a lightweight motor used for portable power tools and appliances.

Required functions:

- switch / case statements

Like if statements, switch···case controls the flow of programs by allowing programmers to specify different code that should be executed in various conditions. In particular, a switch statement compares the value of a variable to the values specified in case statements. When a case statement is found whose value matches that of the variable, the code in that case statement is run.

The break keyword exits the switch statement, and is typically used at the end of each case. Without a break statement, the switch statement will continue executing the following expressions ("falling-through") until a break, or the end of the switch statement is reached.

Example

```
Switch (var) {
    Case 1:
      //do something when var equals 1
      Break;
    case 2:
```

```
      //do something when var equals 2
      break;
    default:
      // if nothing else matches, do the default
      // default is optional
  }
```

Syntax

```
switch (var) {
  case label:
    // statements
    break;
  case label:
    // statements
    break;
  default:
    // statements
}
```

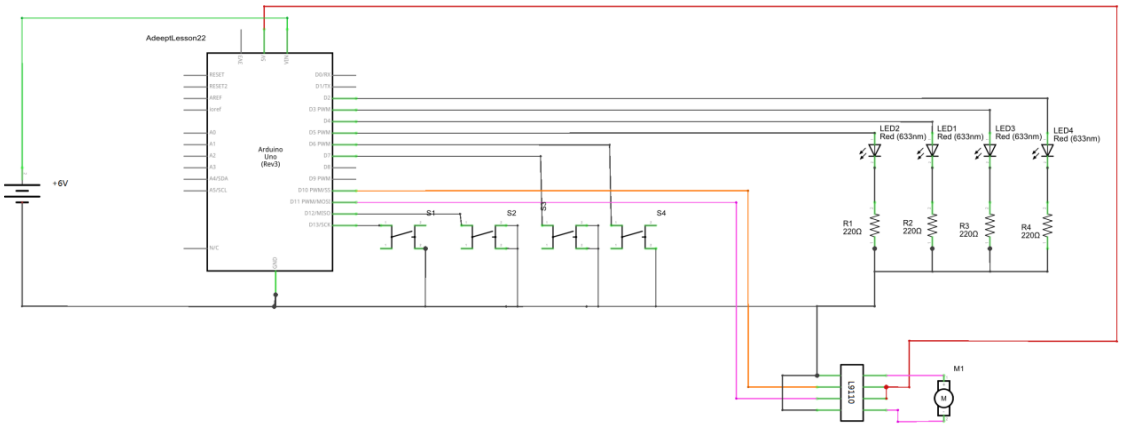
Parameters

var: the variable whose value to compare to the various cases label: a value to compare the variable to

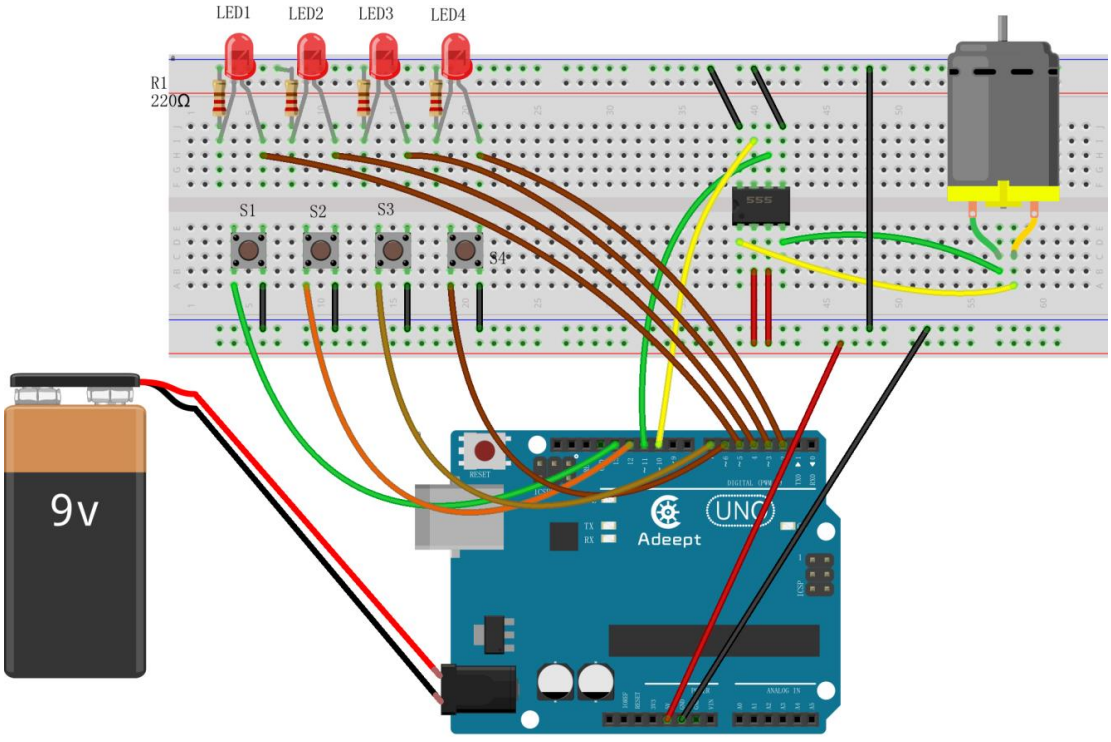
Operating steps：

1. When the power is off, we integrate all the required materials according to the schematic diagram or connection diagram.

Schematic diagram



connection diagram



2. Compile the experimental code and download it to arduino UNO R3

Experimental code

Refer to Project 9.2 code folder.

Experimental result：

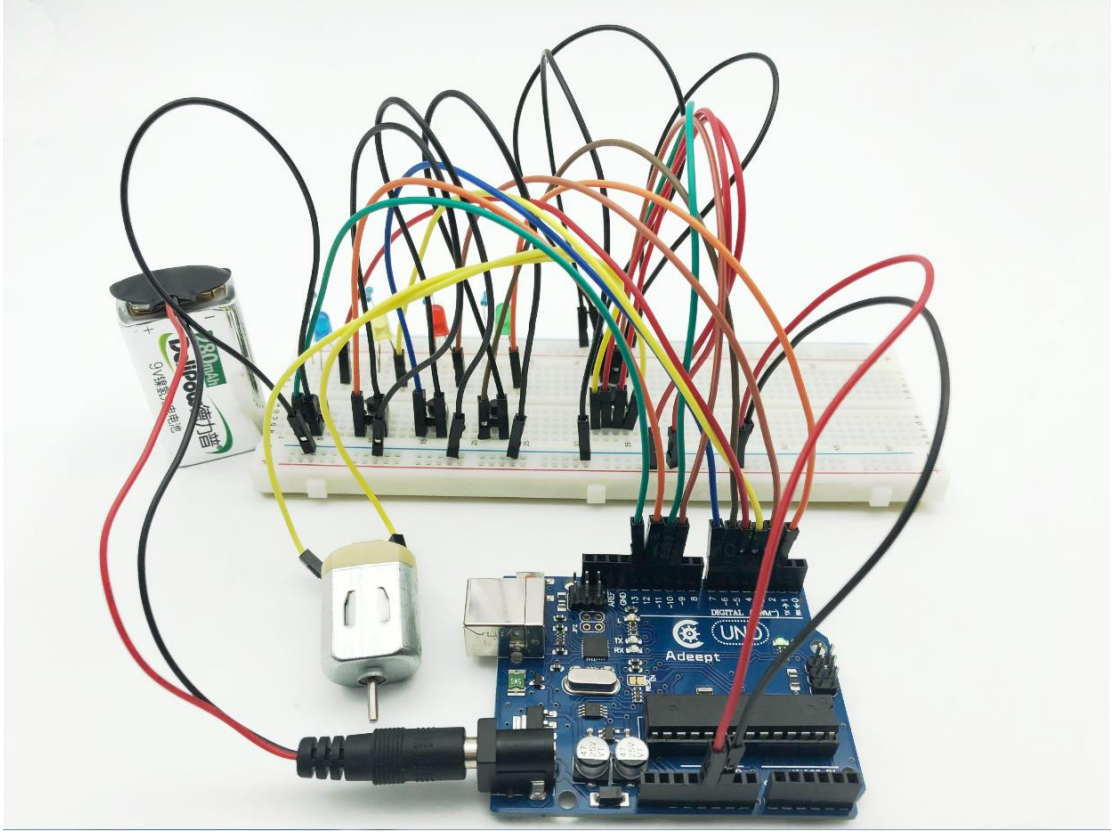
Press button 1 to stop or run the DC motor.

Press button 2 to forward or reverse the DC motor.

Press button 3 to accelerate the DC motor.

Press button 4 to decelerate the DC motor.

Press any one of the buttons, the LED lamp will blink.



Experimental conclusion:

In this course we have learned the basic theory and programming of the DC motor. We can make it forward and backward, more than that, we can adjust its speed.

Chapter 10 Rotary Encoder Module

Project 10.1 Rotary Encoder Module

Experimental objective：

In this experiment, we will learn how to use the rotary encoder.

Required materials:

- 1* Arduino UNO
- 1* USB cable
- 1* Rotary encoders
- 1* 5-pin wires

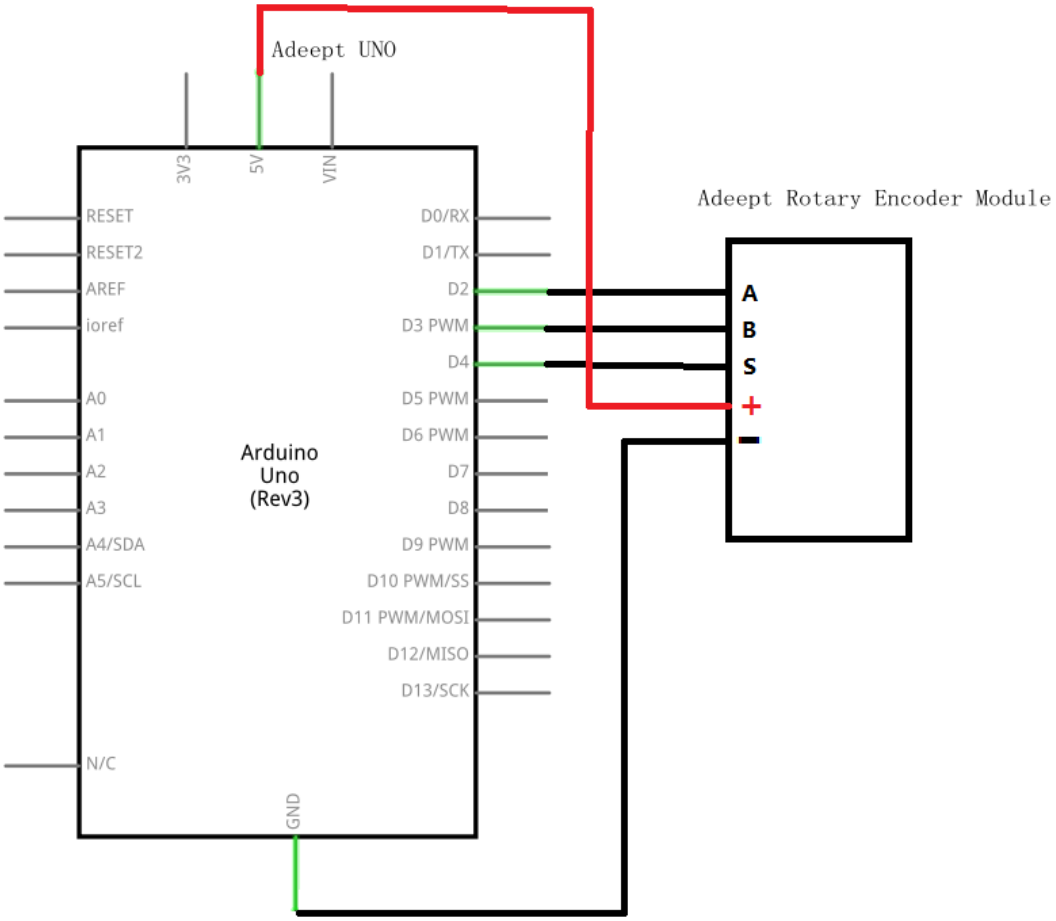
Operating principle：

Rotary incremental encoder outputs pulse when rotating and locates it by counting equipment, when the encoder stop or the power is off, it remembers the location by the inside memory of counting equipment. In this way, when the power is off, the encode can’t move in any way, when the power is connected, in the process of outputting, there can’t be any interruption which leads to losing pulse, otherwise, the memorial null point of counting equipment will change and the amount of changes can’t be measured, we can only know it after mistake happened.

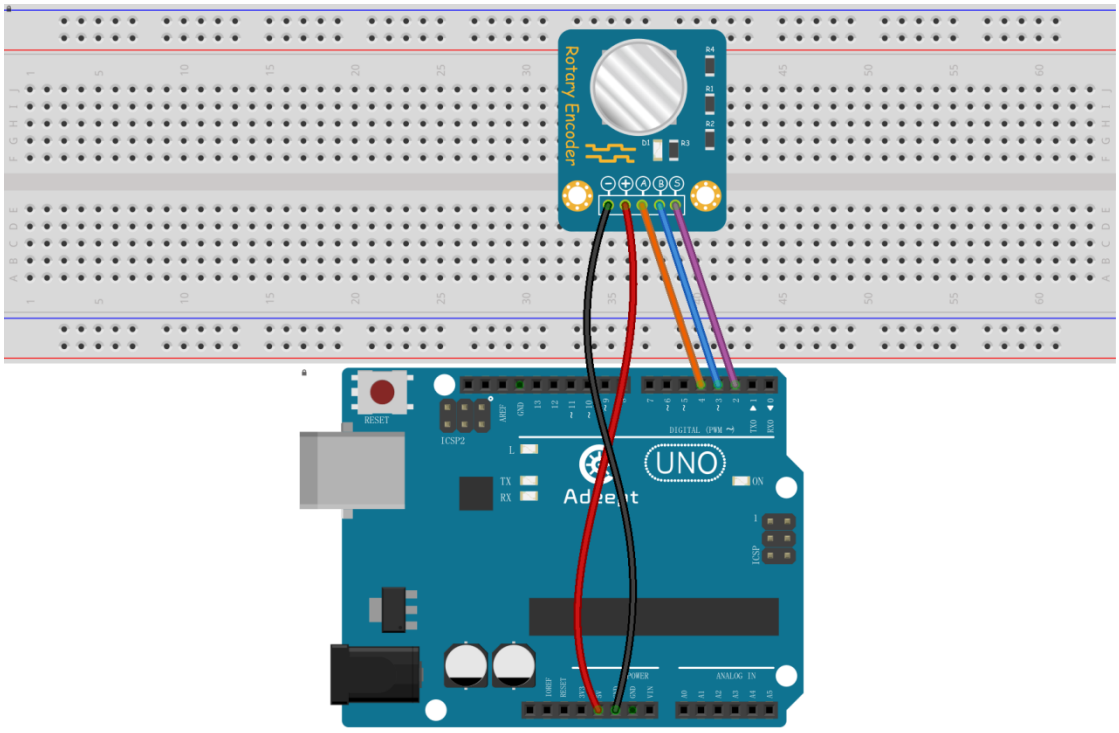
Operating steps：

1. When the power is off, we integrate all the required materials according to the schematic diagram or connection diagram.

Schematic diagram



connection diagram



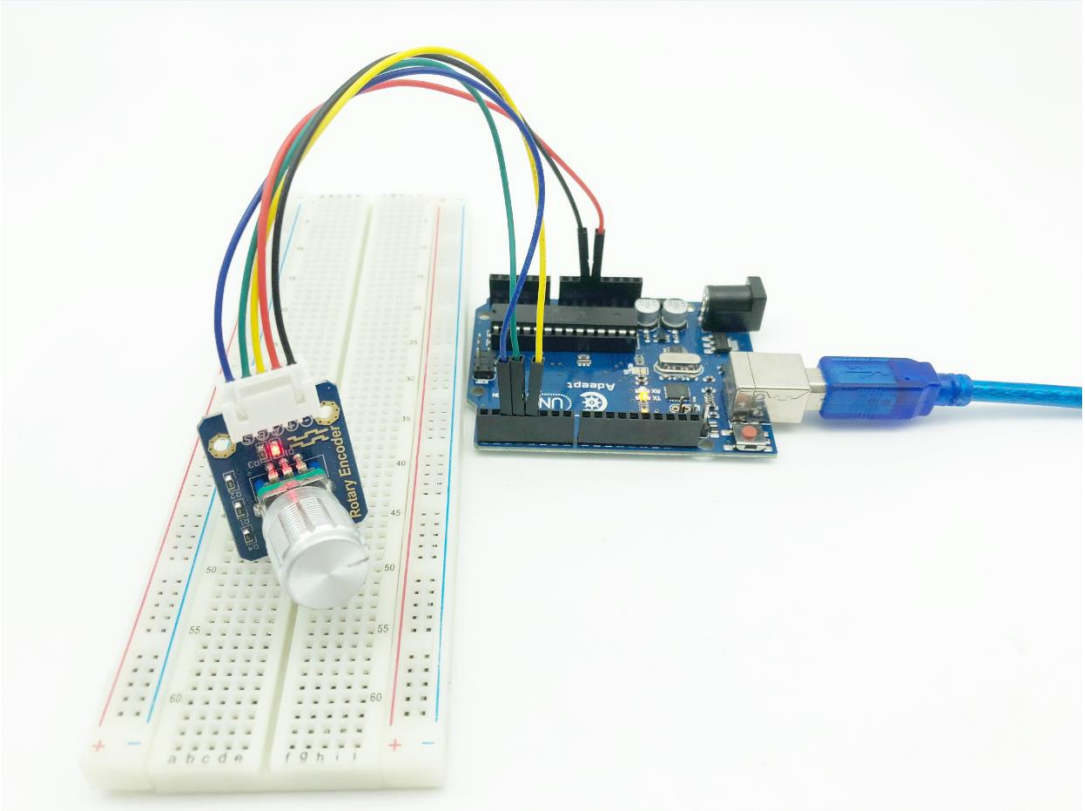
2. Compile the experimental code and download it to arduino UNO R3

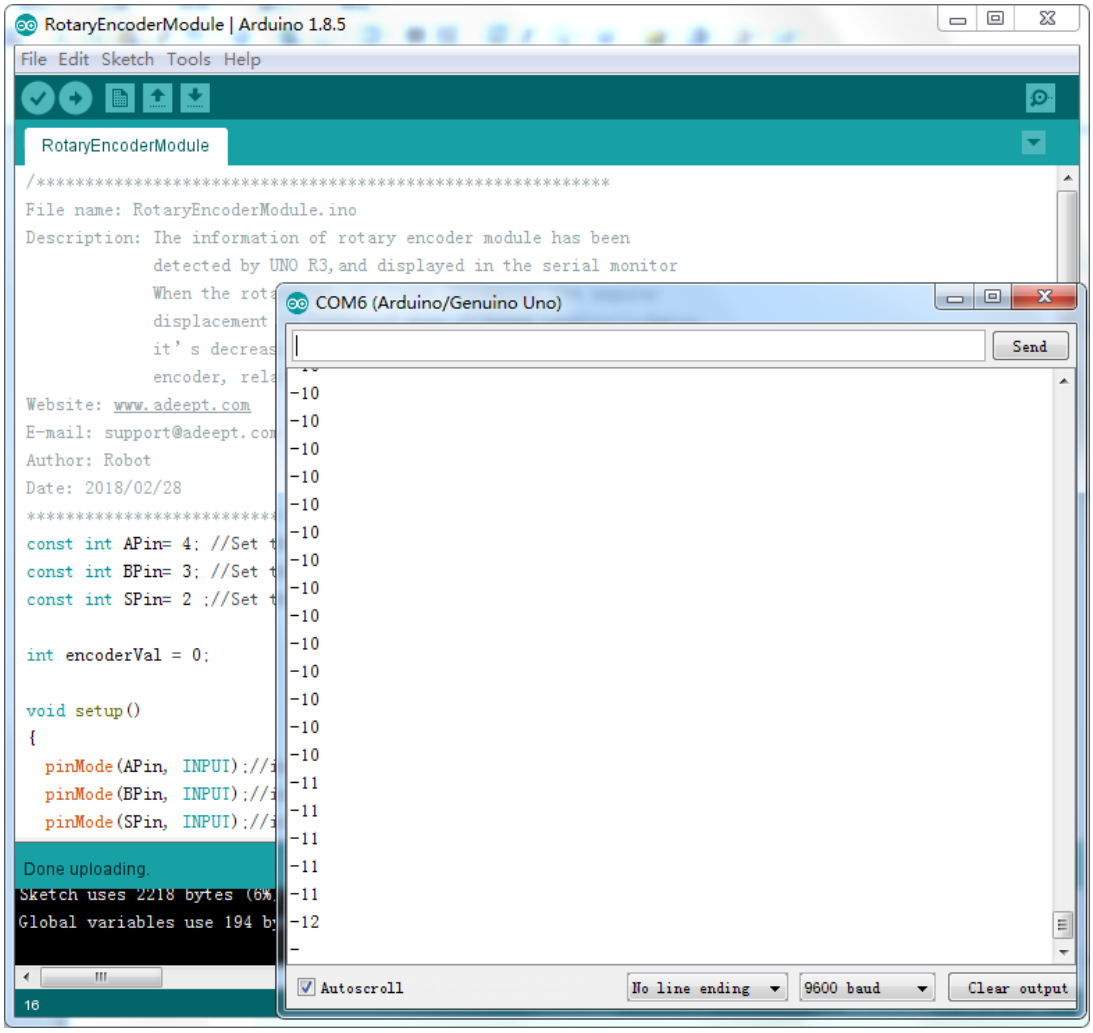
Experimental code

Refer to Project 10.1 code folder

Experimental result:

When we rotate the rotary encoder clockwise, the value of rotary encoder will increase progressively, likewise, when we rotate the rotary encoder anticlockwise, the value of rotary encoder will decrease progressively. When we press the rotary encoder, its value will return to zero.





Experimental conclusion:

After this course we have known how to read the value of rotary encoder, and in the following courses, we will gradually learn some series experiment about rotary encoder.

Project 10.2 Rotary Encoder controls RGB

Experimental objective:

In this experiment, we will learn how to rotate the rotary encoder to control RGB lamp.

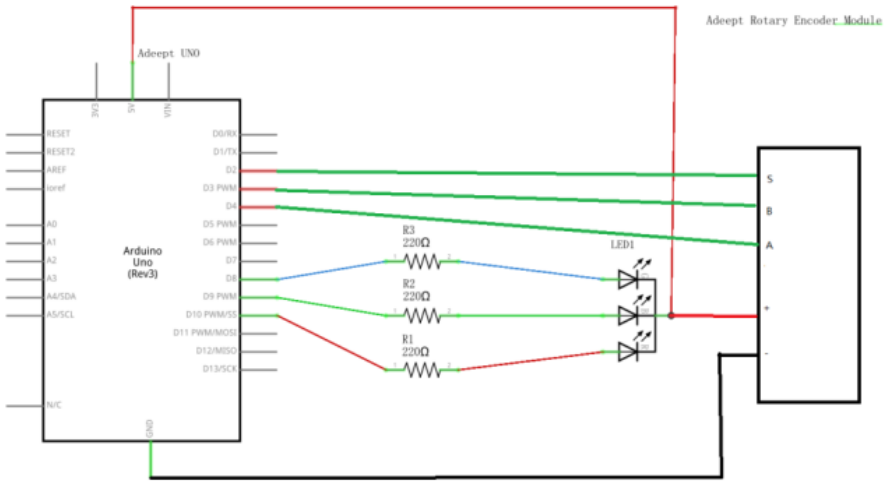
Required materials:

- 1* Arduino UNO
- 1* USB cable
- 1* Rotary encoders
- 1* RGB LED
- 3* 220Ω Resistor
- 1* 5-pin wires
- Several Jumper wires
- 1* Breadboard

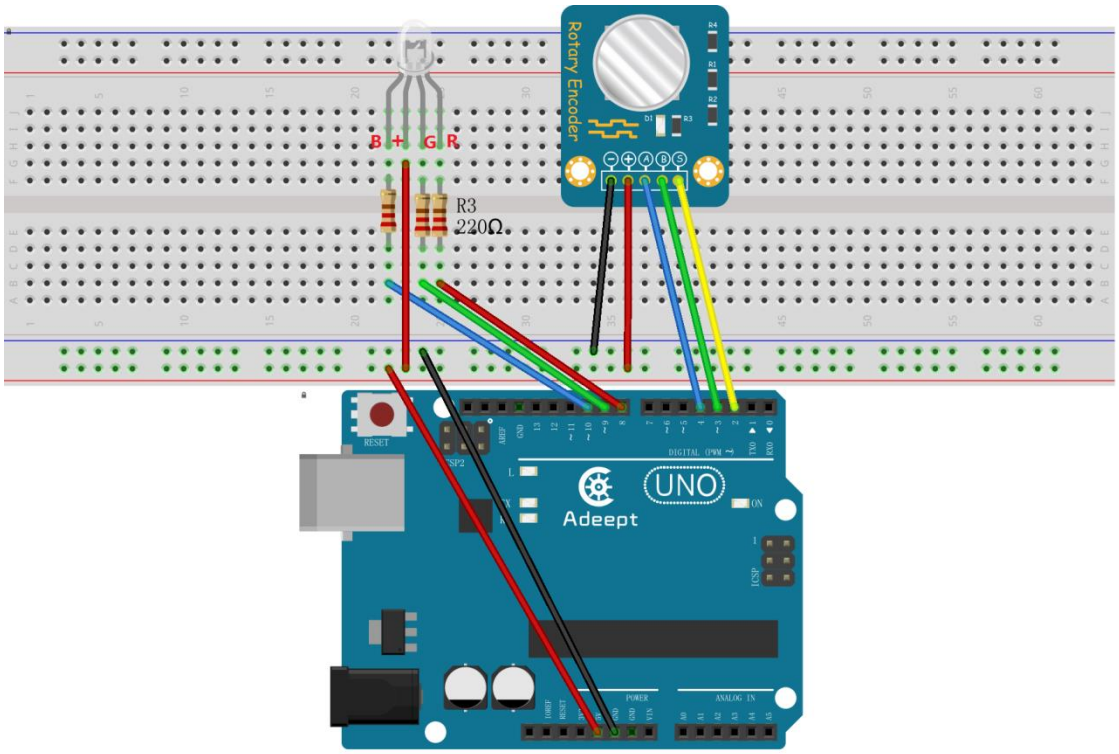
Operating steps:

1. When the power is off, we integrate all the required materials according to the schematic diagram or connection diagram.

Schematic diagram



connection diagram



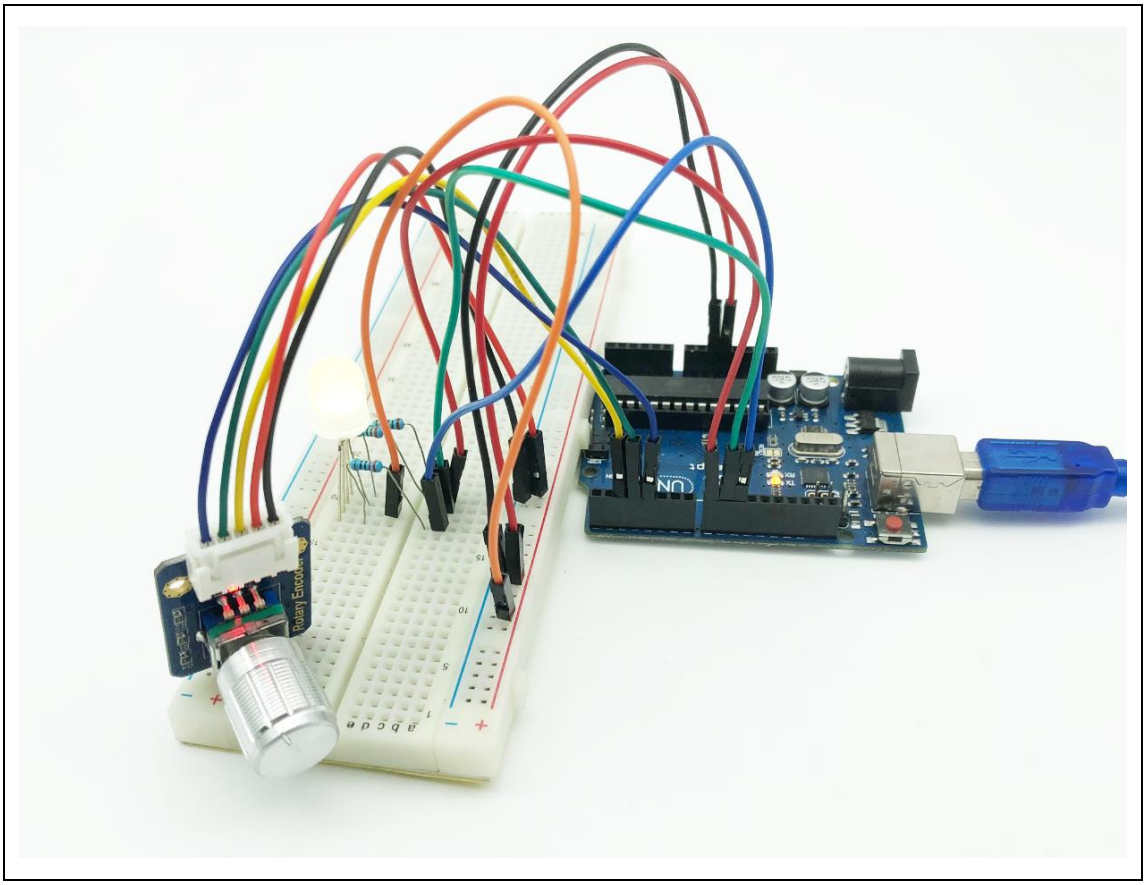
2. Compile the experimental code and download it to arduino UNO R3

Experimental code

Refer to Project 10.2 code folder.

Experimental result:

Now we can see that when we rotate the rotary encoder, the RGB lamp will change color, when the value of rotary encoder get to 5, the RGB lamp will change color automatically.



Experimental conclusion:

After this course, we know how to use the rotary encoder to control RGB lamp. Then we can draw inferences and use rotary encoder to control the other modules.

Project10.3 LCD1602 display the value of the rotary encoder

Experimental objective:

In this experiment, we will combine the last two courses and learn how to read the value of rotary encoder to LCD1602 which is connected to IIC interface module.

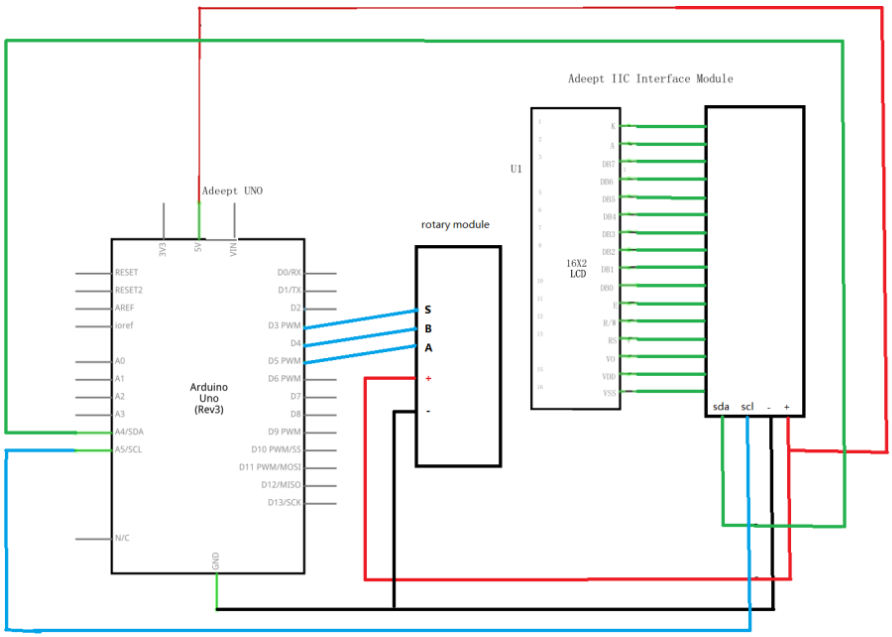
Required materials:

- 1* Arduino UNO
- 1* USB cable
- 1* Rotary Encoders
- 1* LCD 1602
- 1* IIC Interface module
- 1* Breadboard
- 1* 4-pin wires
- 1* 5-pin wires
- Several Jumper wires

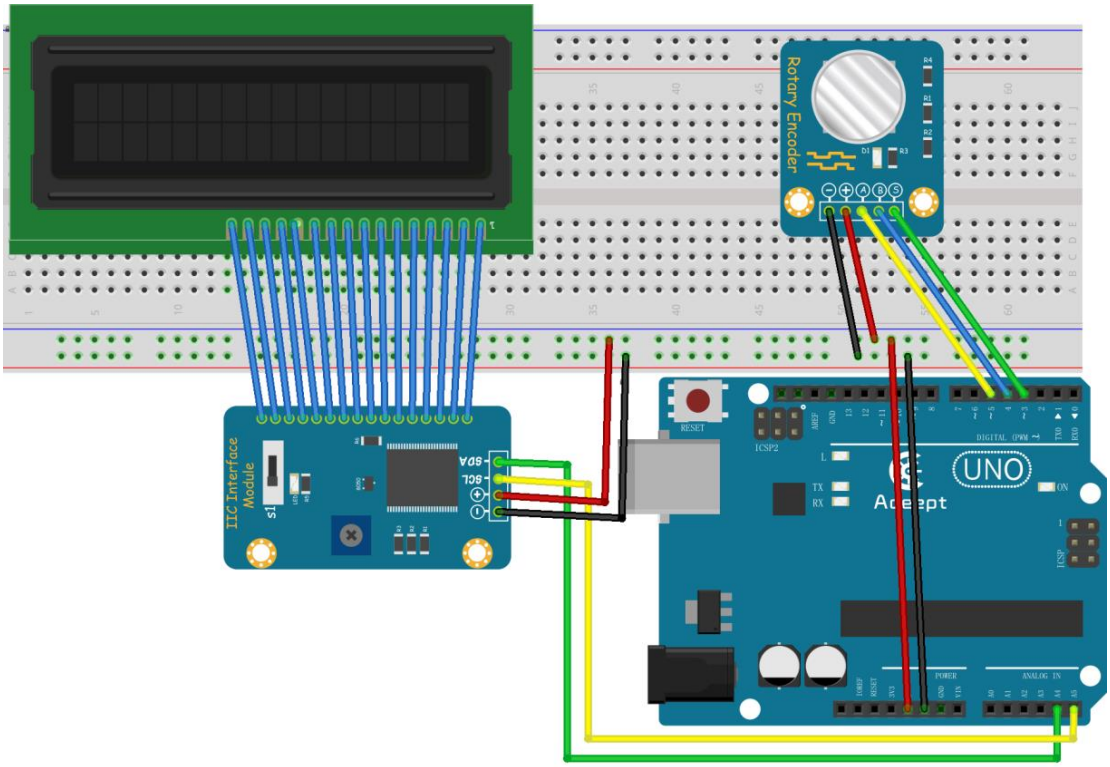
Operating steps：

1. When the power is off, we integrate all the required materials according to the schematic diagram or connection diagram.

Schematic diagram



connection diagram



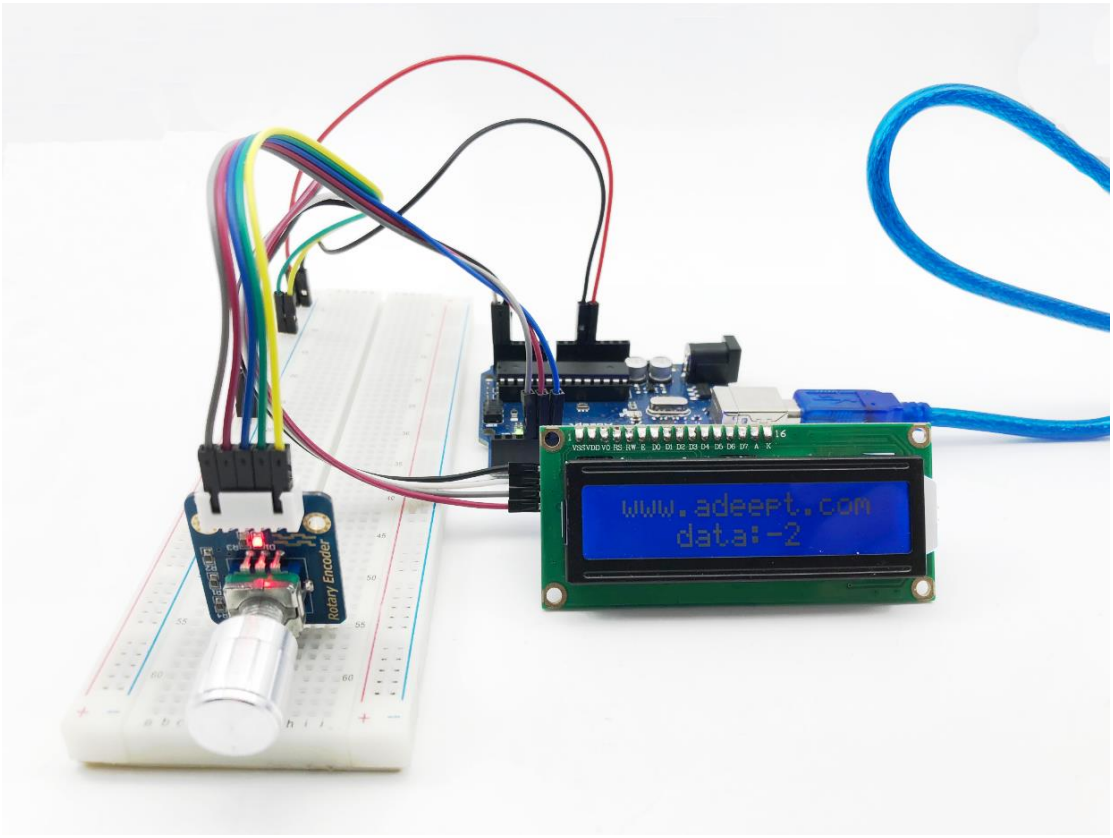
2. Compile the experimental code and download it to arduino UNO R3

Experimental code

Refer to Project 10.3 code folder

Experimental result:

When we rotate the rotary encoder, we can see the corresponding data on LCD.



Experimental conclusion:

After this course we have a better understanding about the rotary encoder and further consolidate the usage of LCD1602.

Chapter 11 Ultrasonic Distance Sensor

Project 11.1 Ultrasonic Distance Sensor

Experimental objective:

In this experiment, we will learn about ultrasonic module.

Required materials:

- 1* Arduino UNO
- 1* USB Cable
- 1* Ultrasonic Distance Sensor
- Several Jumper Wires

Operating principle:

This recipe uses the popular Parallax PING ultrasonic distance sensor to measure the distance of an object ranging from 2 cm to around 3 m.

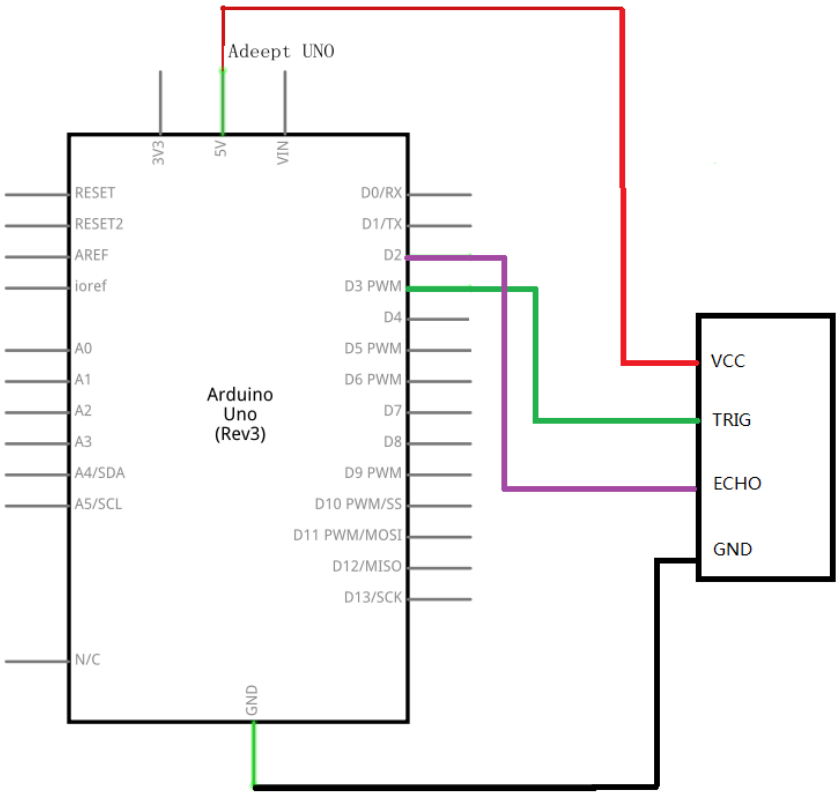
Ultrasonic sensors provide a measurement of the time it takes for sound to bounce off an object and return to the sensor. The “ping” sound pulse is generated when the pingPin level goes HIGH for two micro-seconds. The sensor will then generate a pulse that terminates when the sound returns. The width of the pulse is proportional to the distance the sound traveled and the sketch then uses the pulseIn function to measure that duration. The speed of sound is 340 meters per second, which is 29 microseconds per centimeter. The formula for the distance of the round trip is: RoundTrip = microseconds / 29.

So, the formula for the one-way distance in centimeters is: microseconds / 29 / 2

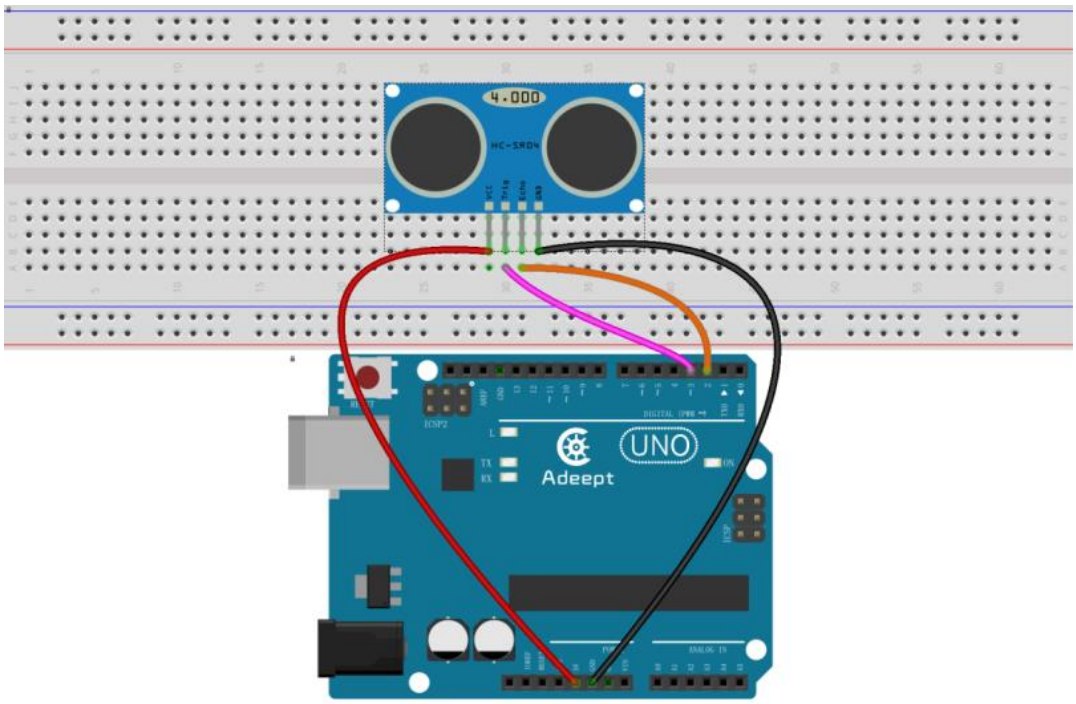
Operating steps:

1. When the power is off, we integrate all the required materials according to the schematic diagram or connection diagram.

Schematic diagram



connection diagram



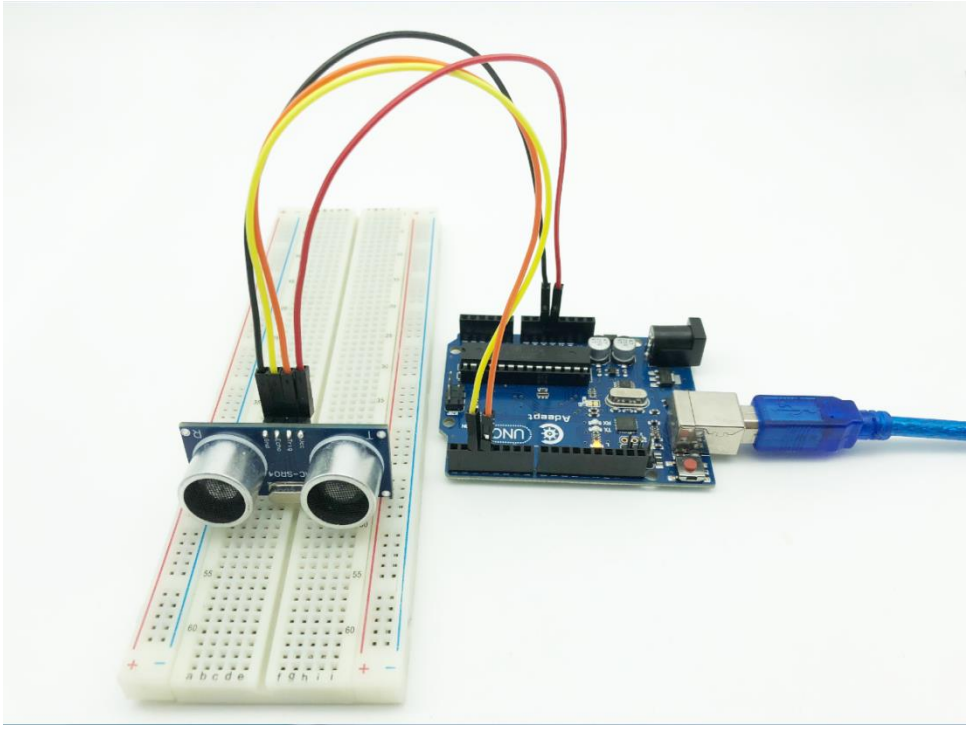
2. Compile the experimental code and download it to arduino UNO R3

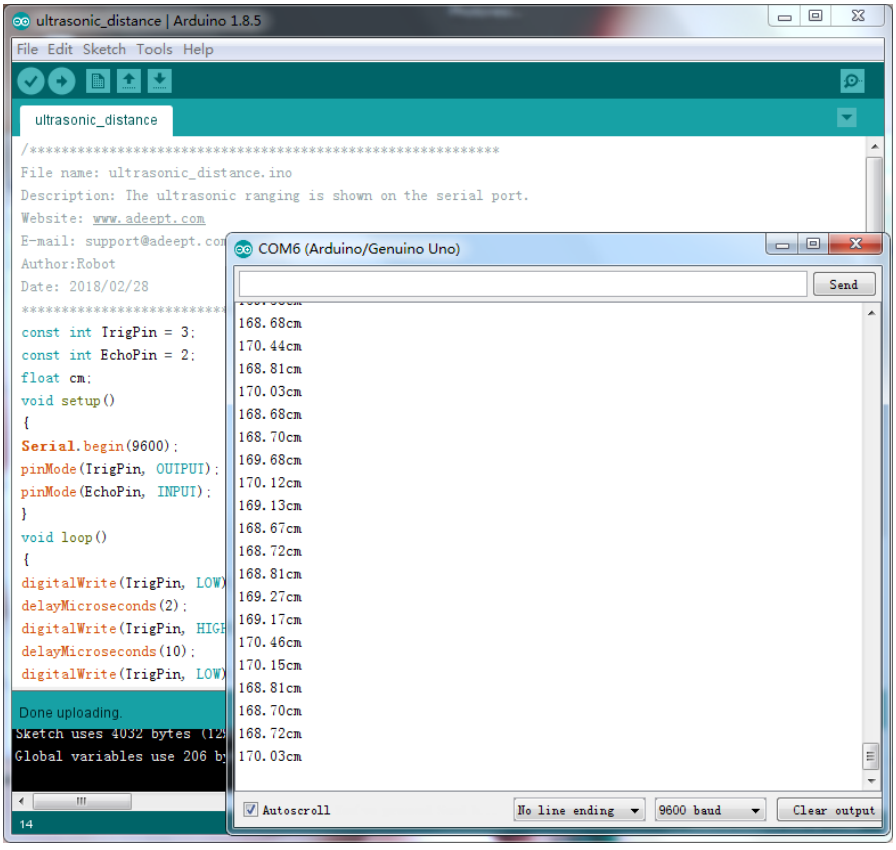
Experimental code

Refer to Project 11.1 code folder.

Experimental result:

Now we can see that ultrasonic distance senor displays the data that it read on the serial port.





Experimental conclusion:

After this course, we have a basic understanding about the ultrasonic module and we can display the value it read on the serial port.

Project 11.2 LCD1602 display Ultrasonic value

Experimental objective:

In this experiment, we will combine the ultrasonic module we have learned before with LCD and IIC interface module and make series experiment.

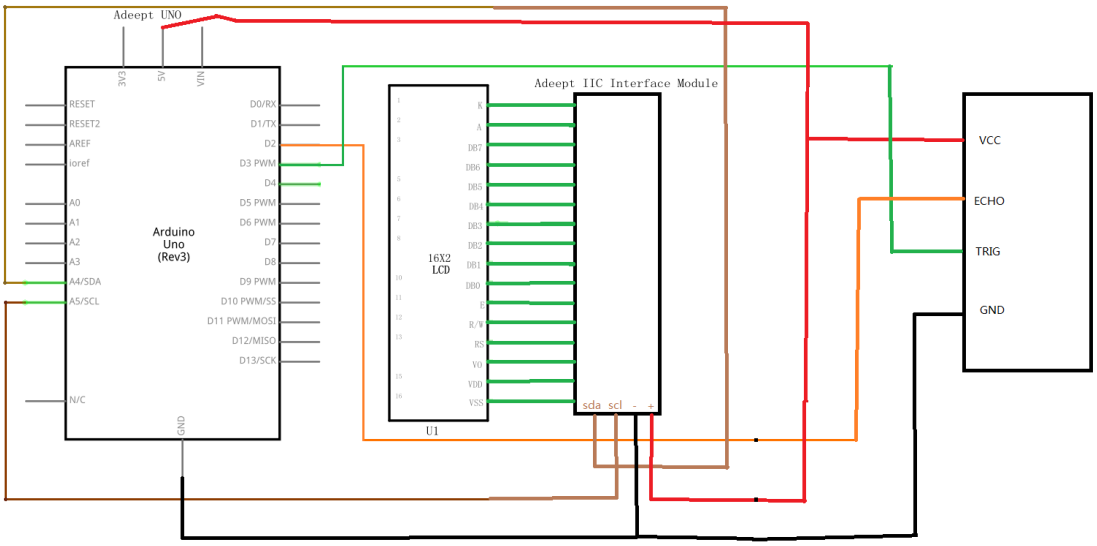
Required materials:

- 1* Arduino UNO
- 1* USB Cable
- 1* Ultrasonic Distance Sensor
- 1* IIC Interface modules
- 1* LCD1602
- 1* 4-pin wires
- Several Jumper Wires

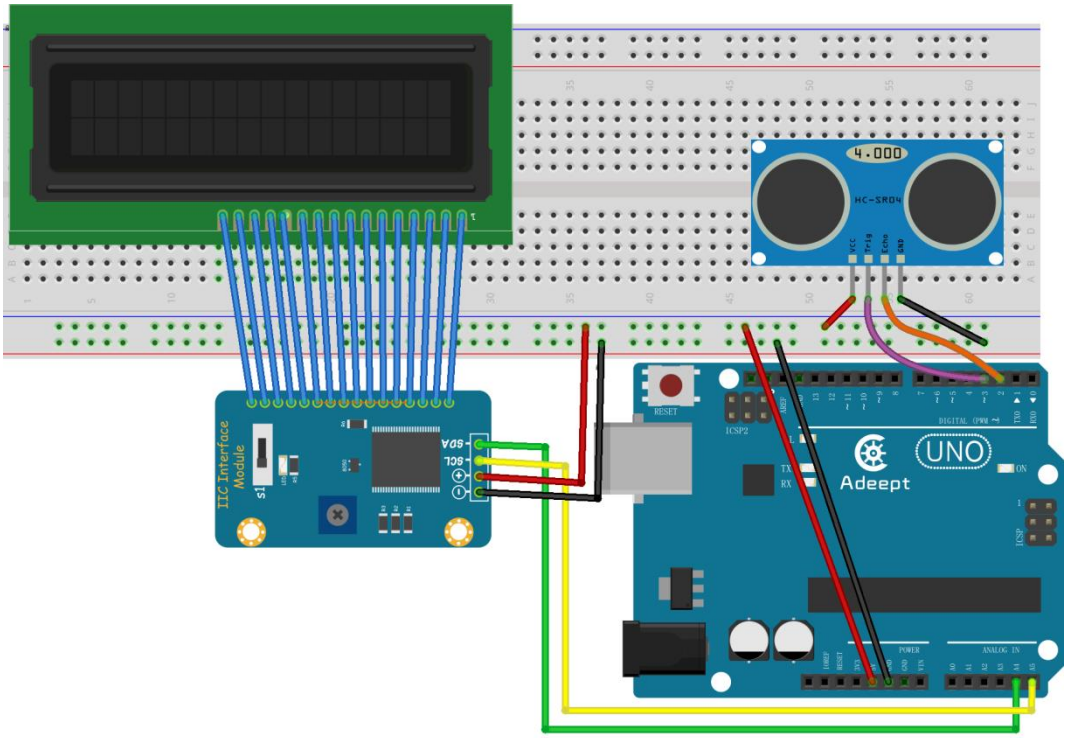
Operating steps:

1. When the power is off, we integrate all the required materials according to the schematic diagram or connection diagram.

Schematic diagram



connection diagram



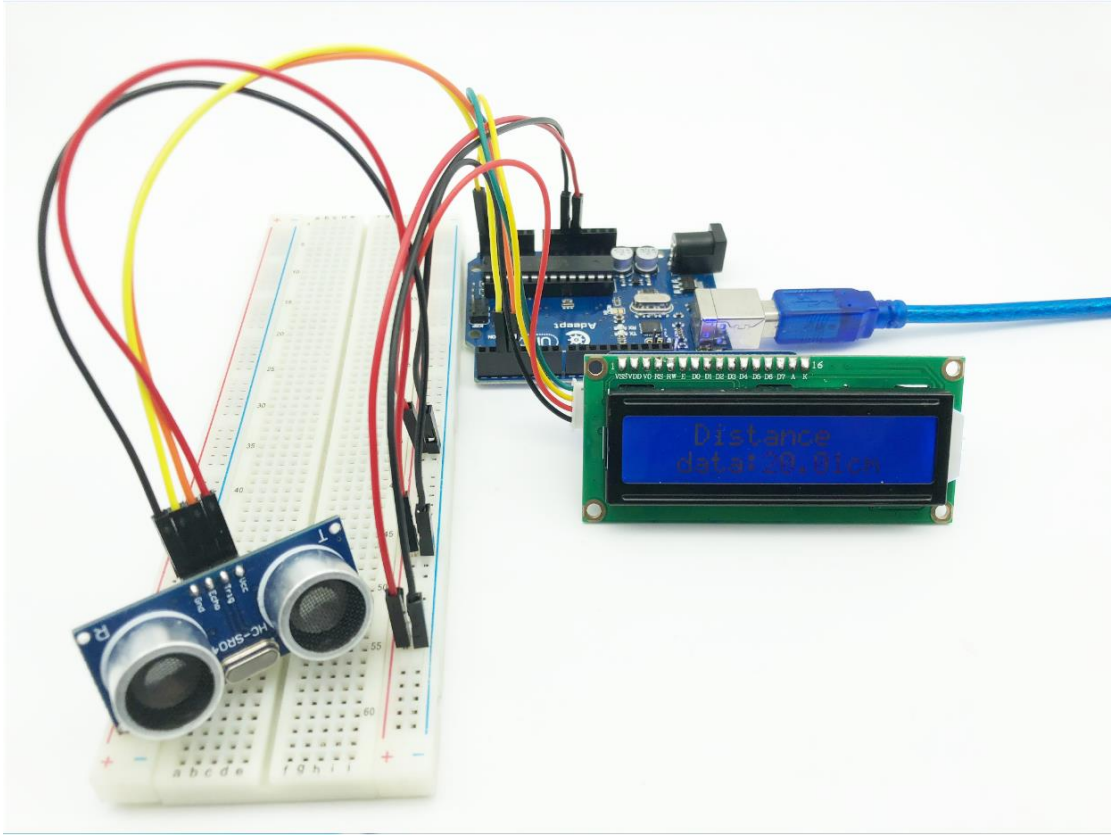
2. Compile the experimental code and download it to arduino UNO R3

Experimental code

Refer to Project 11.2 code folder.

Experimental result：

Now we can see the value read by ultrasonic on LCD1602.



Experimental conclusion:

After this course, we know how to read the value of ultrasonic and display it on LCD. We can use these components and parts to do other interesting experiments in the future.

Chapter 12 processing

Project 12.1 Photoresistor Control picture brightness

Experimental objective：

In this experiment, we start to learn the software Processing, and combine it with Arduino, then use photoresistor to control the brightness of the picture.

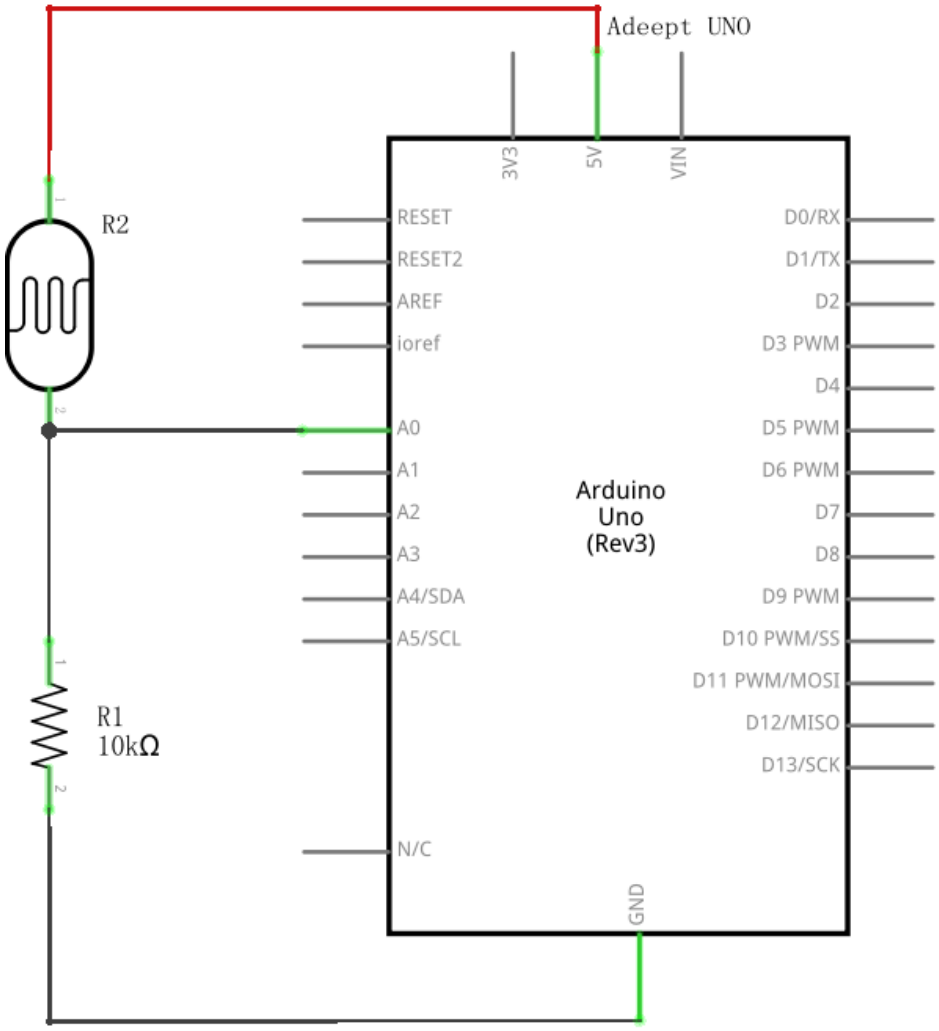
Required materials:

- 1* Arduino UNO
- 1* USB Cable
- 1*10KΩ Resistor
- 1* photosistor
- 1* Breadboard

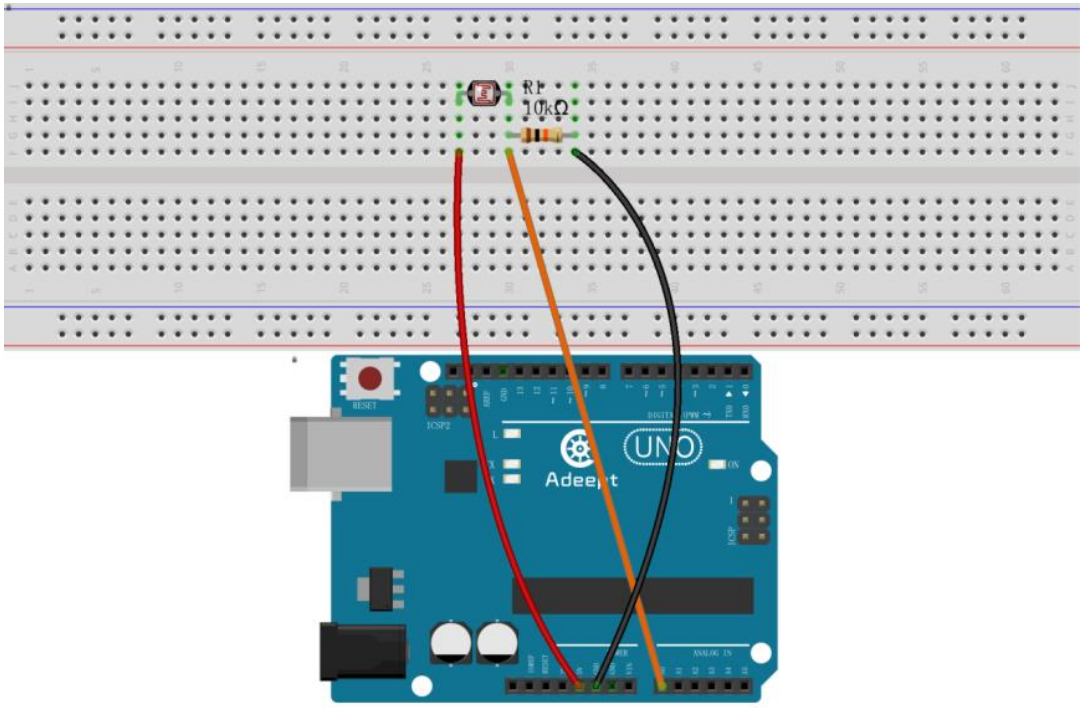
Operating steps：

1. When the power is off, we integrate all the required materials according to the schematic diagram or connection diagram.

Schematic diagram



connection diagram



2. Compile the experimental code and download it to arduino UNO R3

Arduino Experimental code

Refer to Project 12.1 cod/ photo_data folder

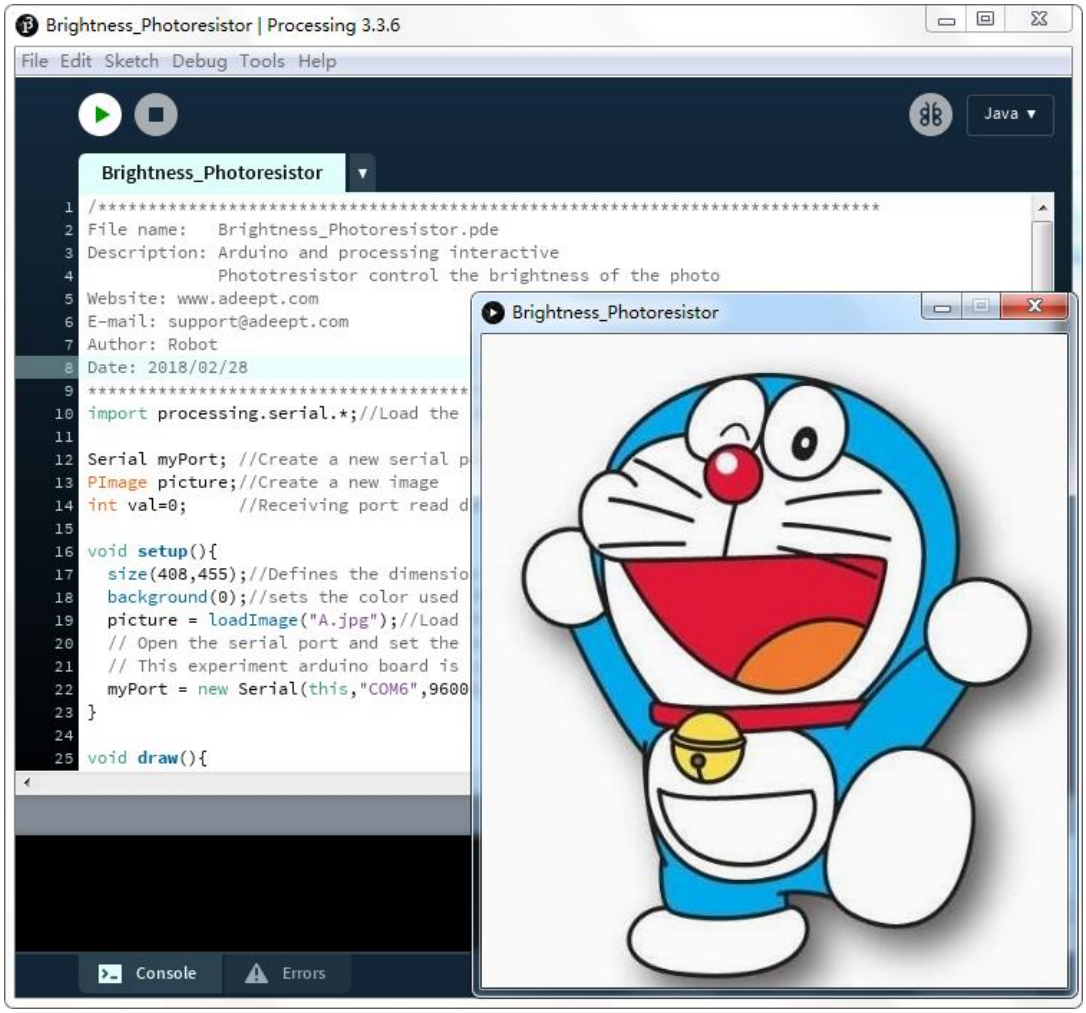
3. Start Processing, compile and execute Processing code.

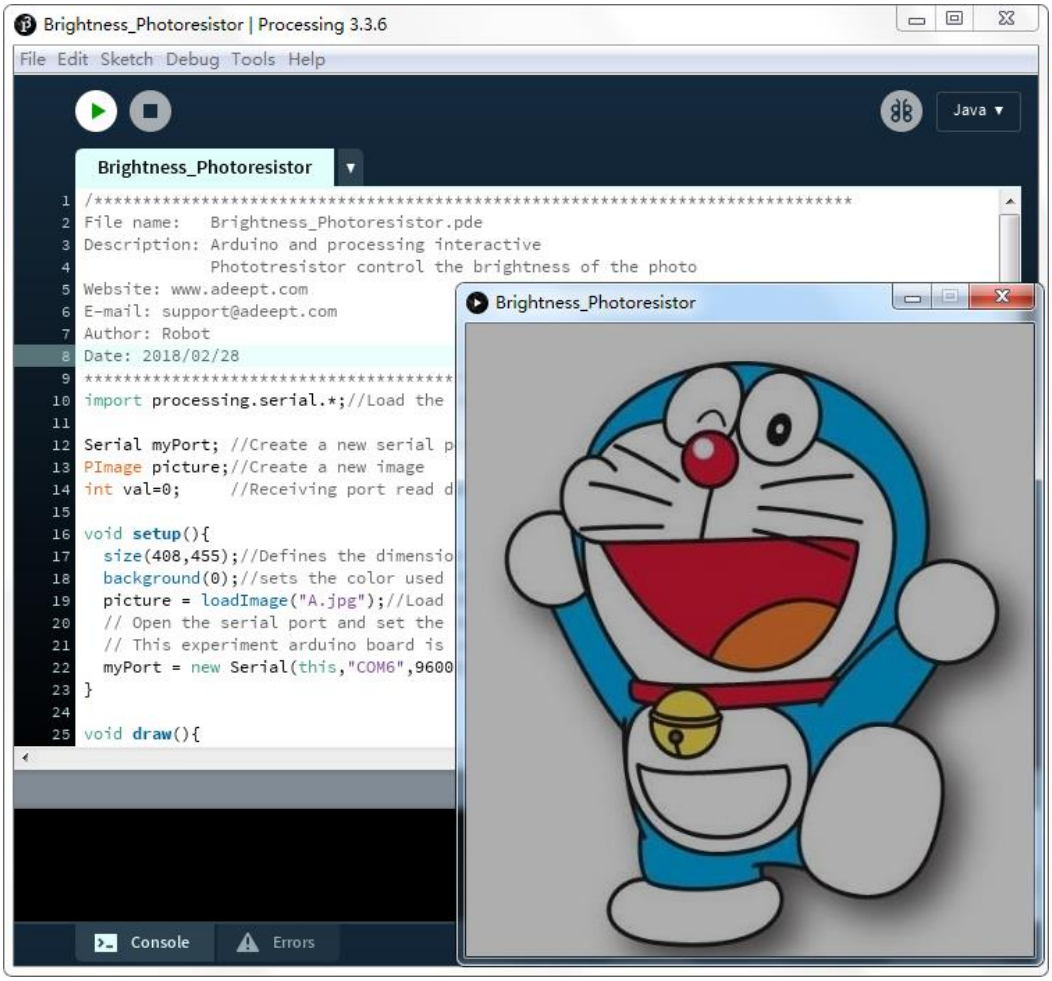
Processing Experimental code：

Refer to Project 12.1 cod/ Brightness_Photoresistor folder（Notice: the picture must be put in Processing folder, and then it can be operated correctly）

Experimental result:

Now, when we cover the photoresistor on arduino UNO with hand, we can see that the brightness of the picture will change correspondingly.





Experimental conclusion:

After this course, we know how to combine Processing with Arduino, and use photoresistor to control the brightness of the picture.

Project 12.2 processing Control RGB

Experimental objective:

In this experiment, we will learn how to control the color of RGB lamp through Processing.

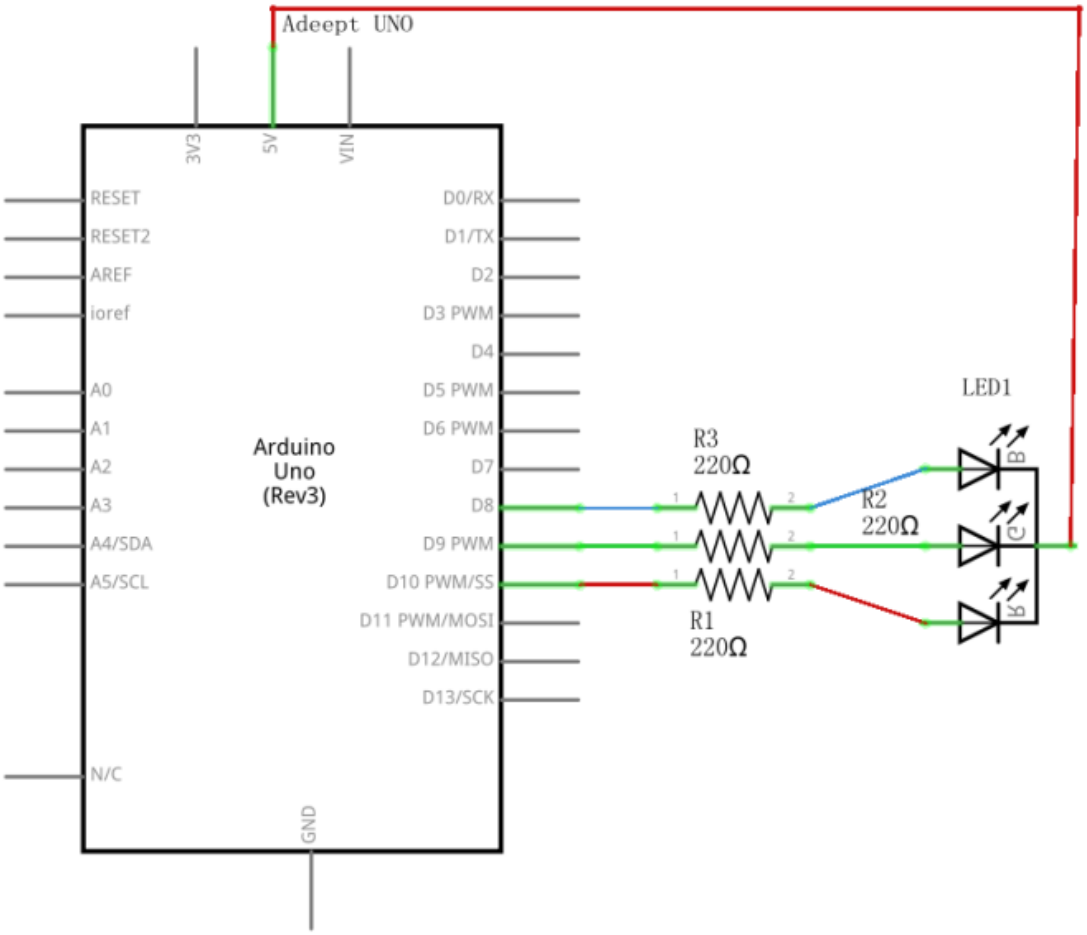
Required materials:

- 1* Arduino UNO
- 1* USB Cable
- 3* 220Ω Resistor
- 1* RGB LED
- 1* Breadboard
- Several Jumper Wires

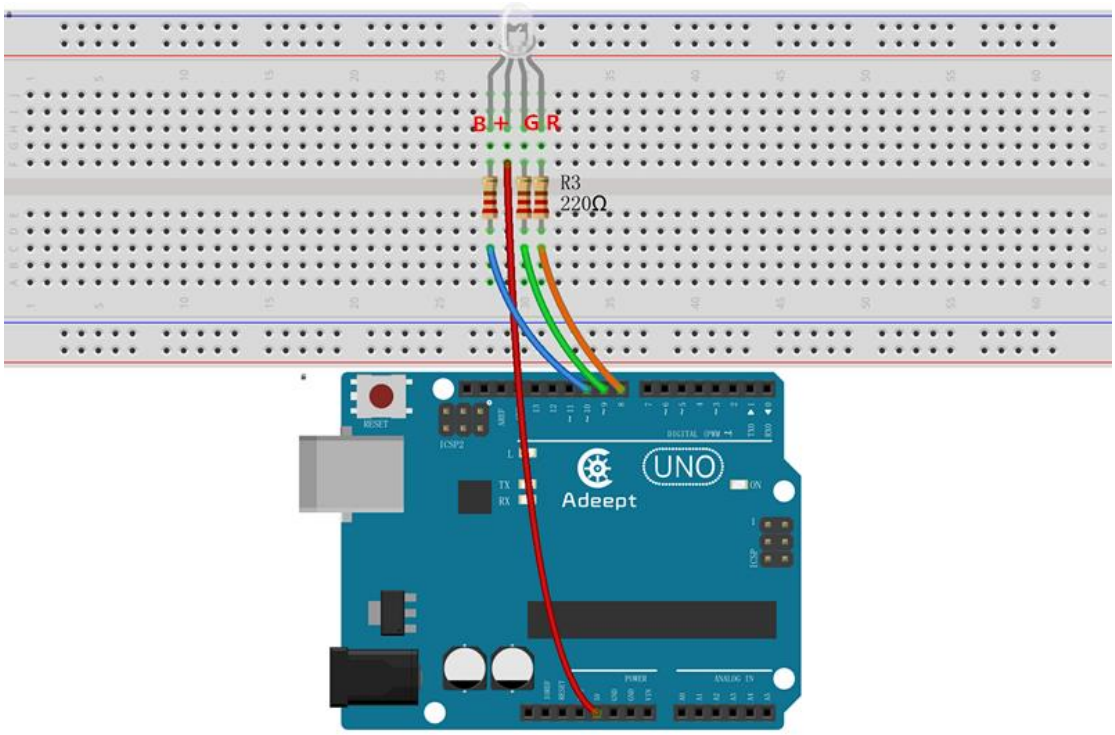
Operating steps:

1. When the power is off, we integrate all the required materials according to the schematic diagram or connection diagram.

Schematic diagram



connection diagram



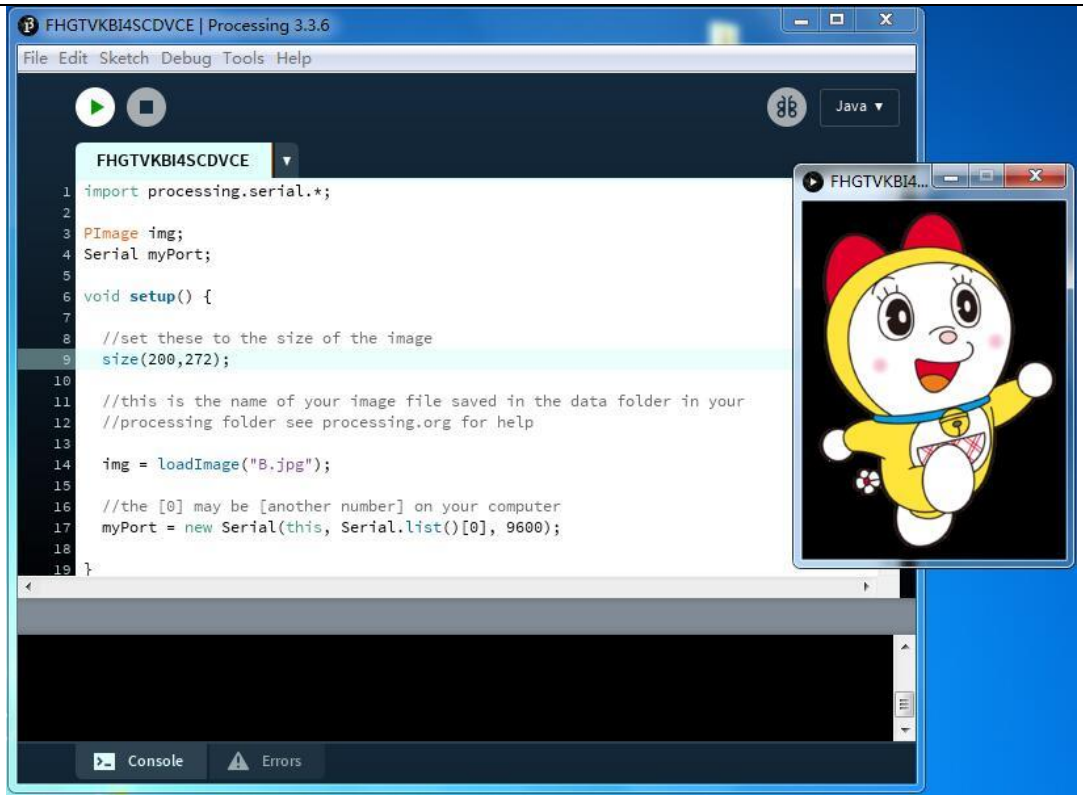
2. Compile the experimental code and download it to arduino UNO R3

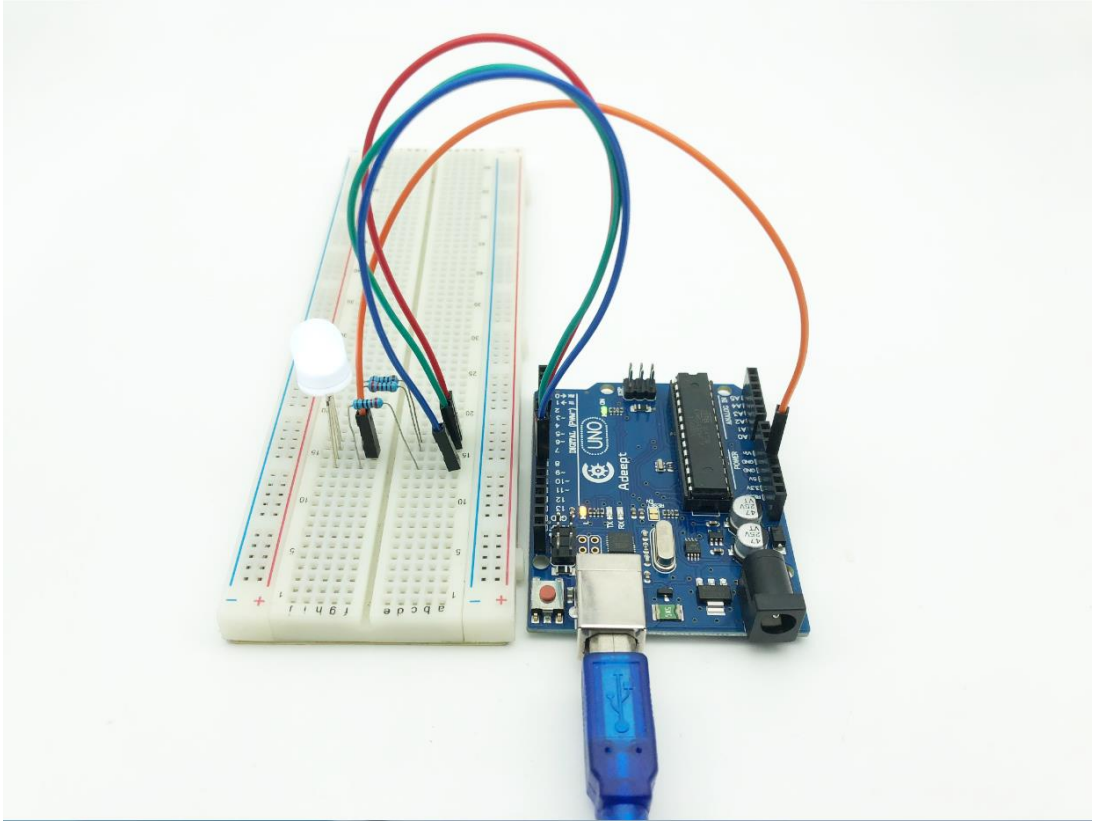
Experimental code

Refer to Project 12.2 code folder

Experimental result:

Now, we click the picture on Processing, and the RGB lamp will show different colors.





Experimental result:

After this course, we can click any part of the picture on Processing and make the RGB on arduino display different colors.

Project 12.3 processing Detect ultrasonic data

Experimental objective:

In this experiment, we start to learn how to display the data that read by ultrasonic on Processing.

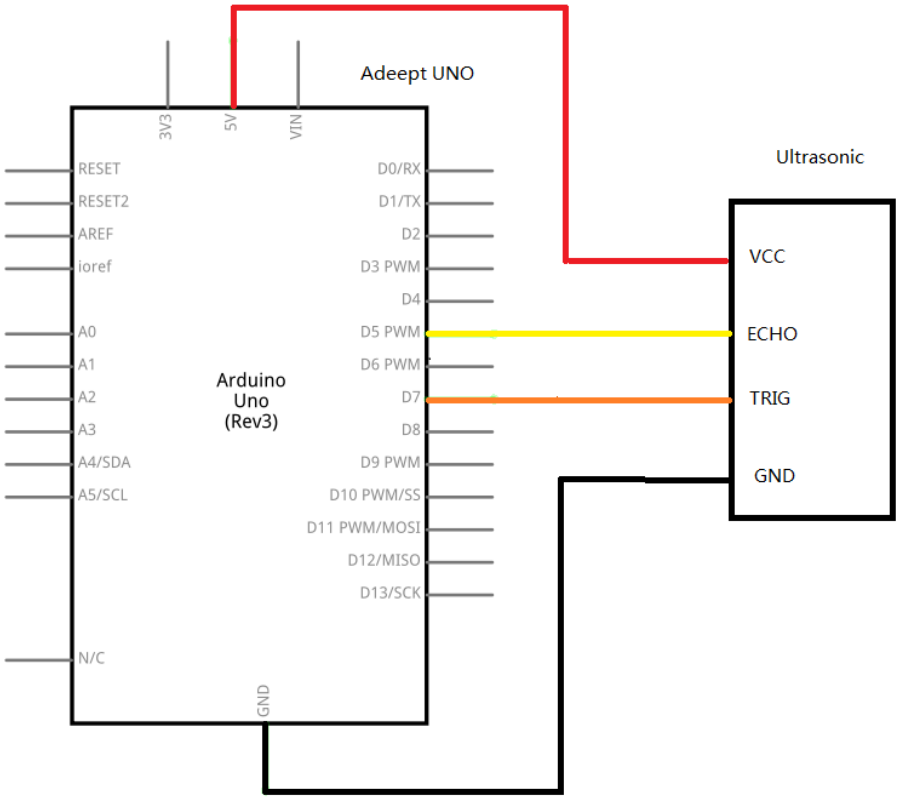
Required materials:

- 1* Arduino UNO
- 1* USB Cable
- 1* Ultrasonic module
- 1* Breadboard
- Several Jumper Wires

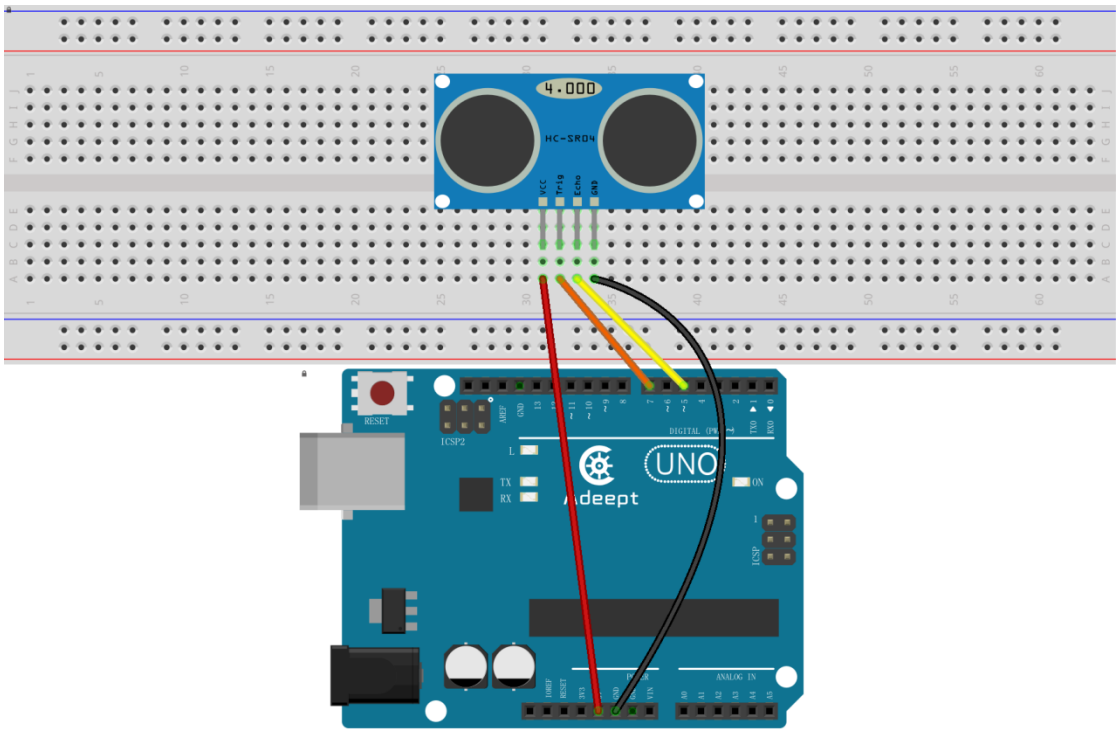
Operating steps:

1. When the power is off, we integrate all the required materials according to the schematic diagram or connection diagram.

Schematic diagram



connection diagram



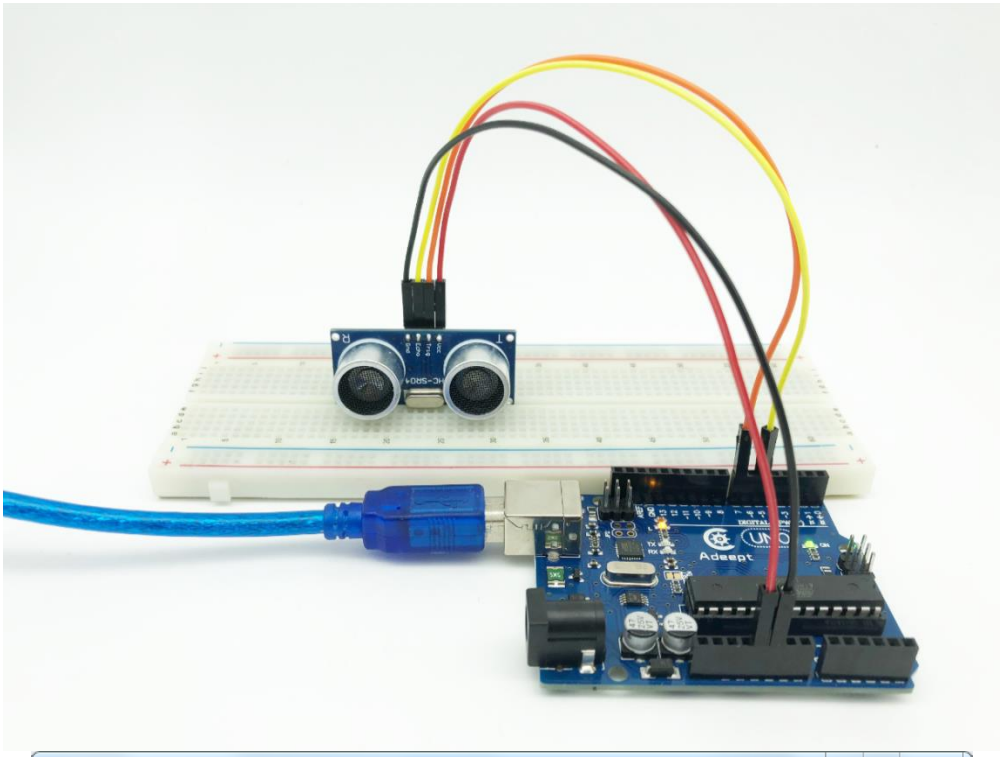
2. Compile the experimental code and download it to arduino UNO R3

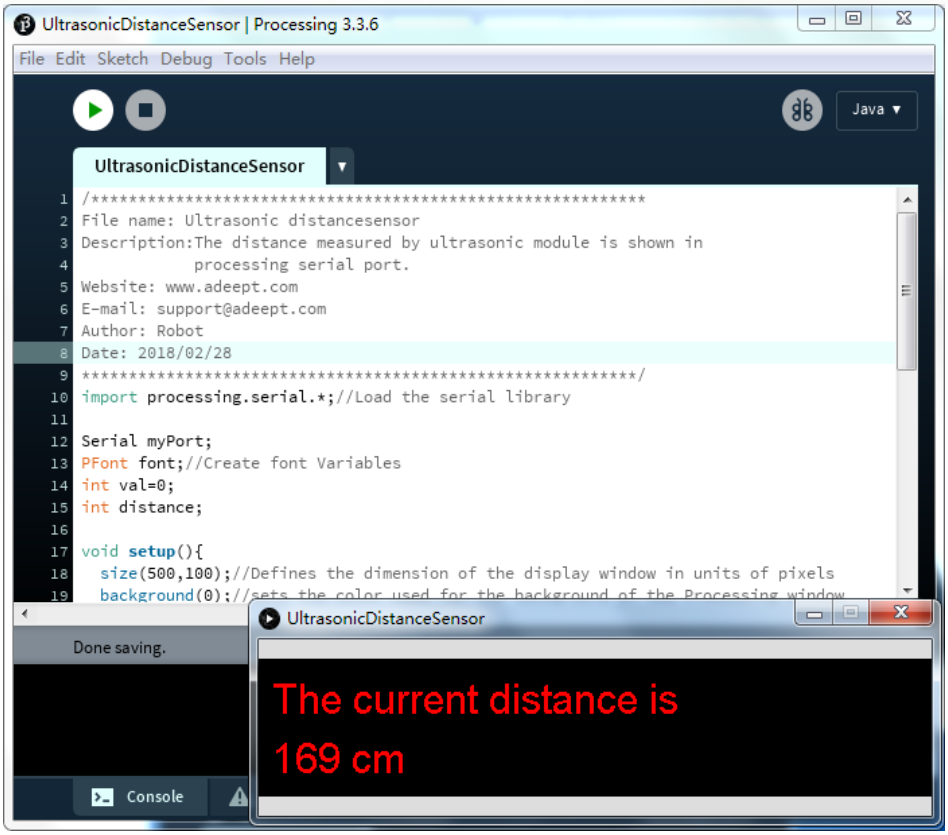
Experimental code

Refer to Project 12.3 code folder.

Experimental result:

Now we can see the value of distance detected by arduino on Processing.





Experimental conclusion:

After this course, we know how to display the value of distance detected by ultrasonic on Processing.

Project 12.4 Processing pong

Experimental objective:

In this course, we will learn to make a ping pong game.

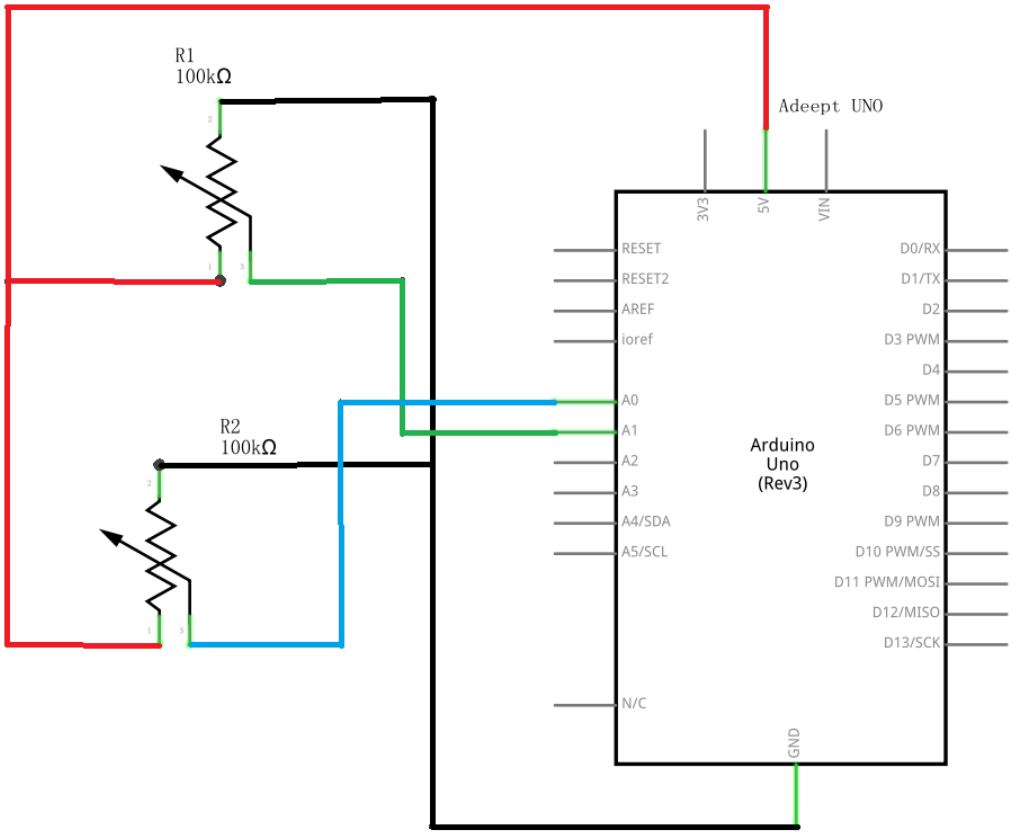
Required materials:

- 1* Arduino UNO
- 1* USB Cable
- 2* 10KΩ Potentiometer
- 1* Breadboard
- Several Jumper Wires

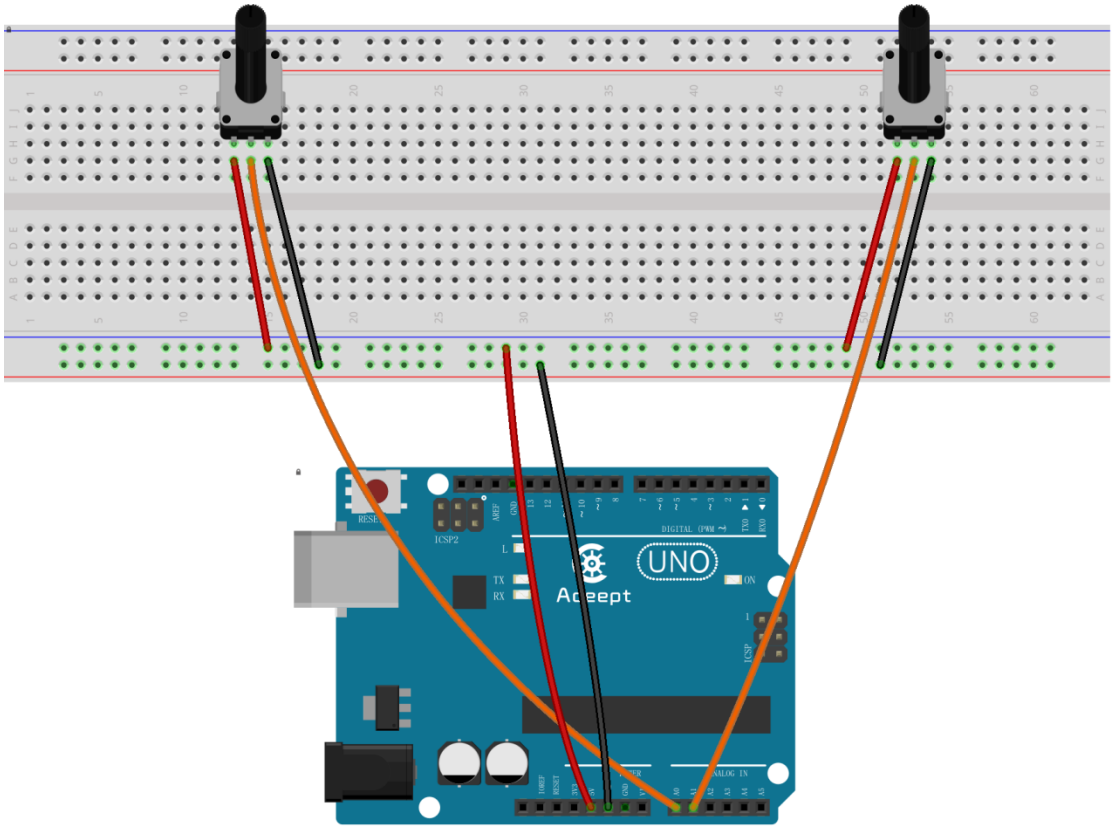
Operating steps:

1. When the power is off, we integrate all the required materials according to the schematic diagram or connection diagram.

Schematic diagram



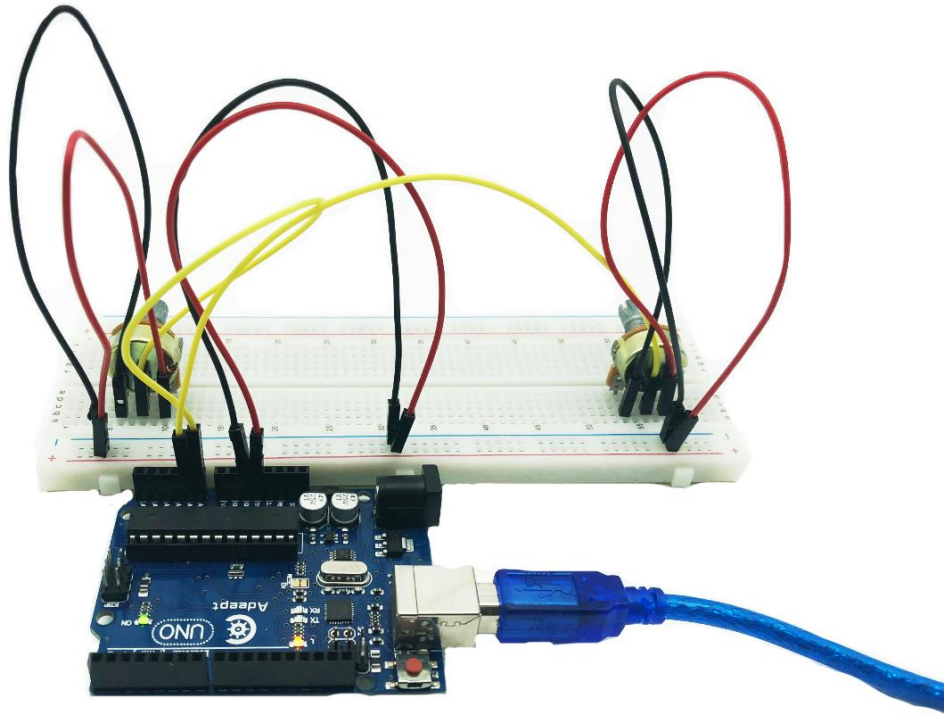
connection diagram



2. Compile the experimental code and download it to arduino UNO R3
Refer to Project 12.4 code folder
the code downloading process :
First we download the ping program in folder to the UNO development board, and then start Processing, next compile and execute the code in pong folder. (notice: we must download the program according to the procedure, only in this way we can get the correct result.)

Experimental result：

Now we can use two potentiometers to control the baffles on both sides of the table.



Experimental conclusion:

After this course, we know how to use the potentiometer to control the sliding of Processing ping pong bats and we can play it with our friends.

Adeept

Sharing Perfects Innovation

E-mail: support@adeept.com

website: www.adeept.com