

Arduino UNO

2016.1.31 history
creat

PREFACE

This manual is based on the Arduino UNO suite system and a tutorial to operate each module, the aim is to make user can quickly the grasp Arduino and familiar with the principle of each module and use how to use it. This document does not give a specific application sample, we hope the guest to learn and be familiar with each module in the tutorial, and flexibly apply to your creative products. Of course, Keyway also use arduino modules to design particularly creative products, these will target different products and provide corresponding suite, detailed tutorials and matching videos.

The Introduction of Arduino UNO

What is arduino ?

Arudino is originally created by Italian teacher Massimo Banzi for the convenience of the electronic major students can present their ideas through hardware. About in the winter of 2005, he united Spain chip engineer David Cuartielles, discussed the idea, so they two decided to design their own circuit boards, and recruited Banzi student David Mellis to write programming language for circuit board. It was named after the name of the Italian king Arduin. He named the Arduino and start to creat Arduino.

Subsequently, Banzi, Cuartielles and Mellis put the design online. Due to everyone only knew the open-source software, there was no heard of source hardware, and then they thought of the well-known Linux open-source software, so they hope Arduino to open source like Linux.

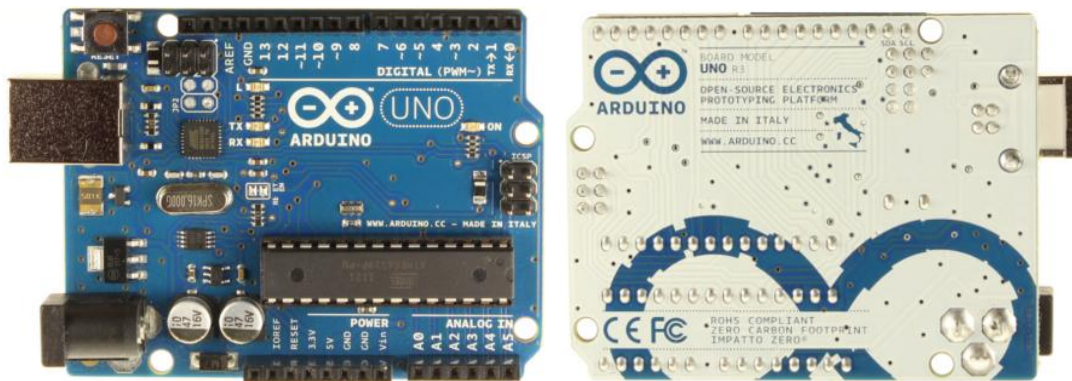
The emergence of the Arduino required original microcontroller programmer to possess relevant electronic or software background, become normal amateurs and soon learned how to use the Arduino. His design concept attracted many professional or non-professional people right after being introduced worldwide.

What is arduino uno Kit ?

Arduino uno is arduino USB series reference standard template. UNO processor core is Atmel company produces Atmega328, at the same time, with 14 digital input/output port (of which six can be used as a pwm output), 6 analog input, a 16 MHz crystal oscillator, a USB port, and a power socket, a ICSP header and a reset button. UNO has been released to the third edition.

- ◆ The processor ATmega328
- ◆ Working voltage 5v
- ◆ Input voltage (recommended) 7-12v
- ◆ Input voltage (range) 6-20v
- ◆ Digital IO pin 14 (of which six as a PWM output)
- ◆ Analog input pin 6
- ◆ IO pin DC 40 mA
- ◆ 3.3V pin DC 50 mA
- ◆ Flash Memory 32 KB (ATmega328, of which 0.5 KB for bootloader)
- ◆ SRAM 2 KB (ATmega328)
- ◆ EEPROM 1 KB (ATmega328)
- ◆ Clock 16 MHz

Picture of real products are as follows:




The Development Environment Building

Development Software

Download link: <https://www.arduino.cc/en/Main/Software>

Windows, Linux, Mac are all available for downloading.

Download the Arduino Software



ARDUINO 1.6.6

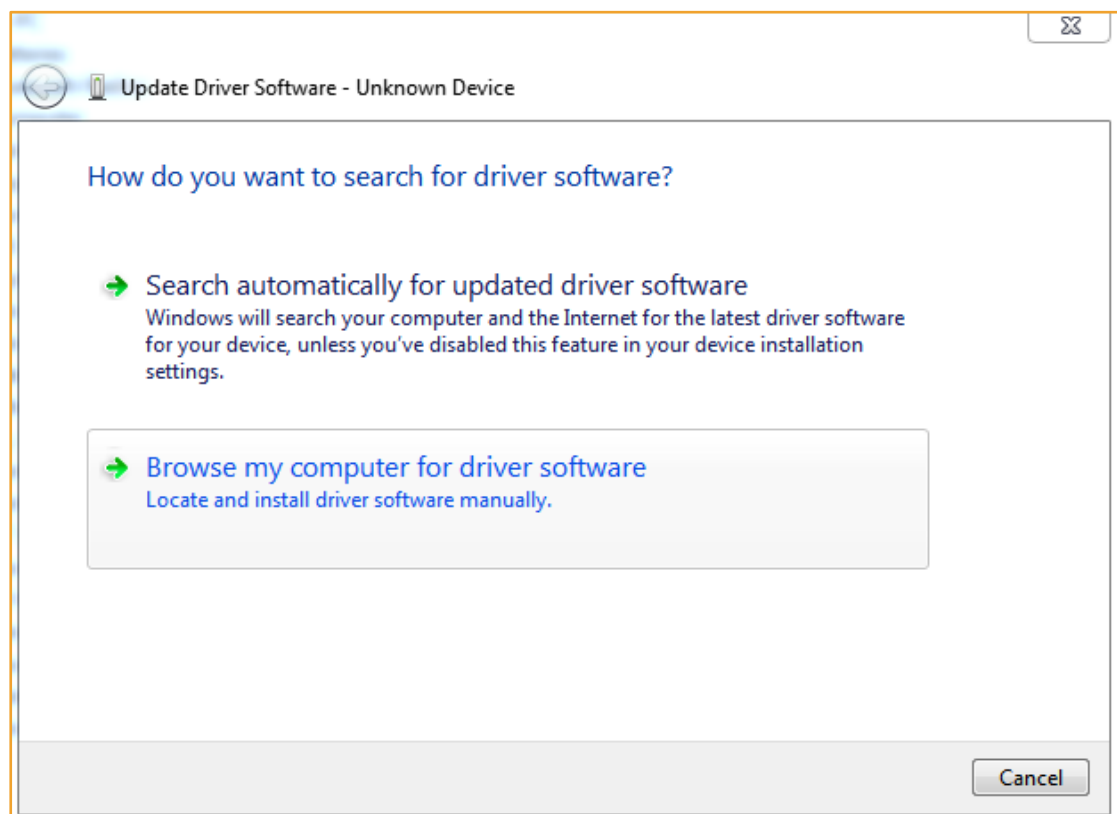
The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions.

Windows Installer
Windows ZIP file for non admin install
Mac OS X 10.7 Lion or newer
Linux 32 bits
Linux 64 bits
[Release Notes](#)
[Source Code](#)
[Checksums](#)

The interface of Arduino IDE is simple and the operation is rather convenient. If you want get more please click <https://www.arduino.cc/en/Guide/Environment>

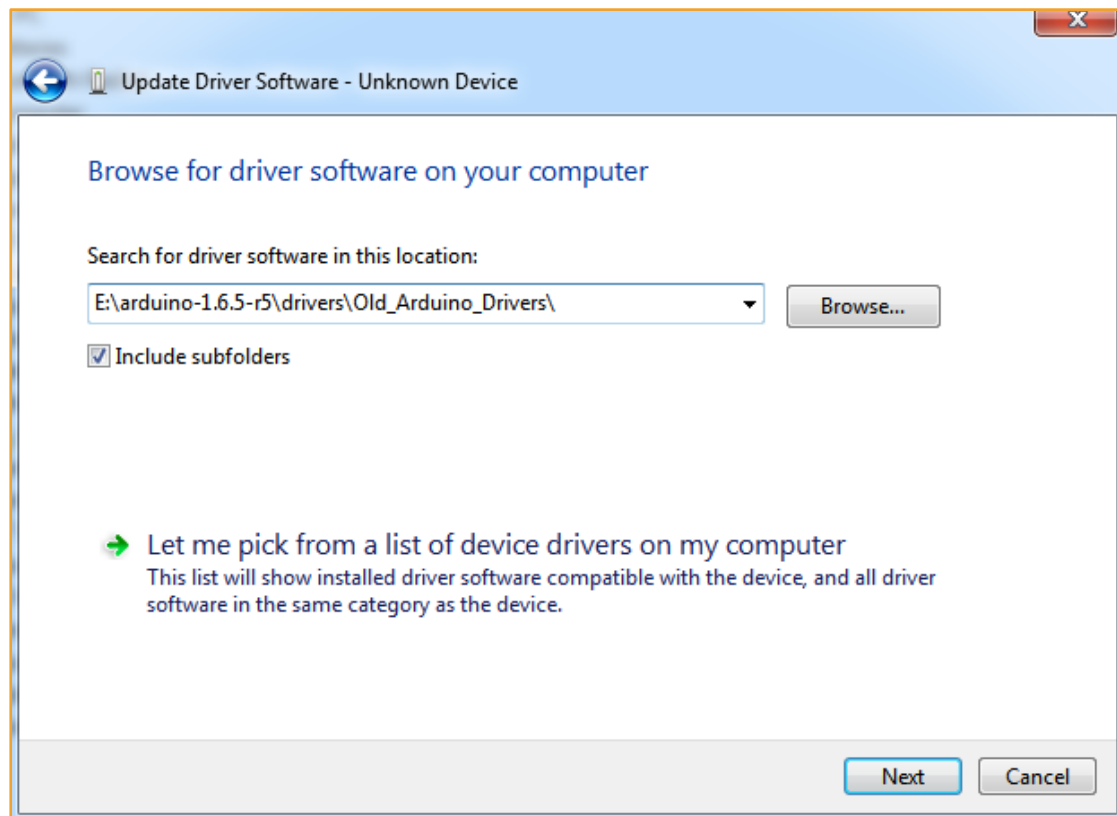
Install usb to serial port driver

Inserting USB cable will prompt as follows, choosing the specified location to install.

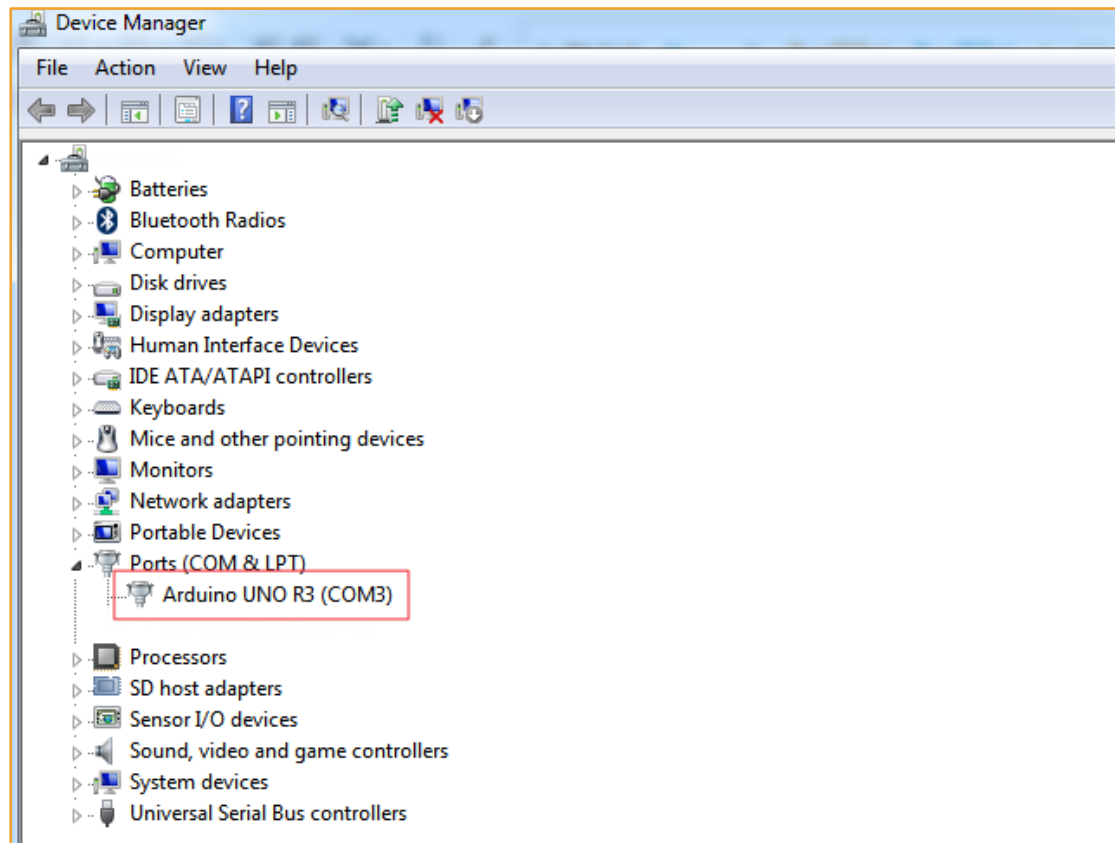


Select download Arduino ide file “E:\arduino-1.6.5-r5\drivers\Old_Arduino_Drivers\” Checking the type of USB serial chip on the board, if it is Atmel, then cho

ose the following path; if it is FTDI, you should choose the arduino\drivers\FTDI USB Drivers path.



Clicking on the next step, you will be prompted with a successful installation message. Now you can change to equipment management to see Arduino UNO R3



Linux Installation Environment Building

Download the Arduino Software select Linux 64bit and save.

```
Shell:# tar -vxf arduino-1.6.7-linux64.tar.xz
```

```
Shell:# cd arduino-1.67
```

```
Shell:# ./arduino
```

Mac Arduino Development Environment Building

Not available

Hello Arduino !

A programmer loves calligraphy years after retirement, one day, he is suddenly in aesthetic mood after a meal, so prepares "the Four Treasures of Study", writing brush, ink stick, ink slab and paper, grinds ink, spreads paper and lights a good sandal, quite a Wang Xizhi demeanor and Yan Zhenqing manner. Composed for a moment, then he splashed ink, earnestly writes down the words: "hello world!". Why programmers are so keen on this a few words? The inception of "Hello world" dates back to 1972, Bell Laboratory's famous researcher Brian Kernighan firstly used it(program) when he was writing "B Language tutorials and guidance (Tutorial the Introduction to the Language B)", this is the earliest known record at present when "hello" and "word" are used together in a computer work. Then, in 1978, he used this sentence pattern "hello, world" again in C Language bible "The C Programming Language" co-authored with Dennis Ritchie, as the first program in the opening. In this program, the output of the "hello, world" are all lowercase, without an exclamation point, a comma followed by a space. Although the initial form almost failed to survive after that, from then on, "hello, world" became a tradition of the program world to greet the outside. "Hello Arduino!", without exception, also became the first program in the tutorial.

Hardware Connection

There is a Atmega16u2 USB serial port on Arduino uno board, so the first program doesn't need to connect other equipment, only requires Arduino UNO to connect PC with a USB line directly.

Software Part

The codes are as follows:

```
void setup() {
    // put your setup code here, to run once:
    Serial.begin(115200);
}
void loop() {
    // put your main code here, to run repeatedly:
    Serial.println("Hello Arduino !");
    delay(1000);
}
```

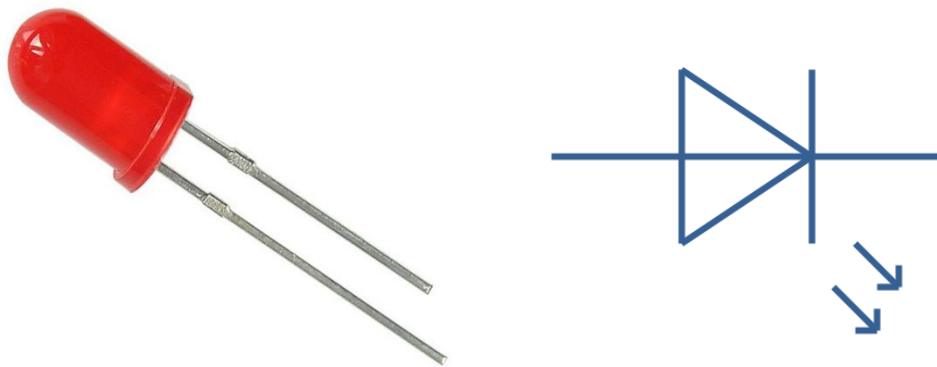

"Hello Arduino" is printed every 1second in the program.

Class 2

LED Running Lights

The introduction of light-emitting diode

LED, short for light-emitting diode, is made by mixed compound of gallium (Ga), arsenic (AS), phosphorus (P). Phosphorus gallium arsenide diode glow red, gallium phosphide diode glow green, silicon carbide diode glow yellow.



Working Principle

The reverse breakdown voltage of light-emitting diode is 5v. Its positive volt-ampere characteristic curve is too steep, it must be in series with current limiting resistor to control the current flowing through the diode when using it. Current limiting resistor R can be available through the following formula:

$$R = \frac{E - V_F}{I}$$

In the formula, E stands for power voltage, V_F is forward voltage drop of LED, I shows the general working current of LED. The working voltage of light-emitting diodes are generally from 1.5 V to 2.0 V, the working current is usually 10 ~ 20 mA. So in the digital logic circuit of 5v, We can use 220Ω resistor as a current limiting resistor.

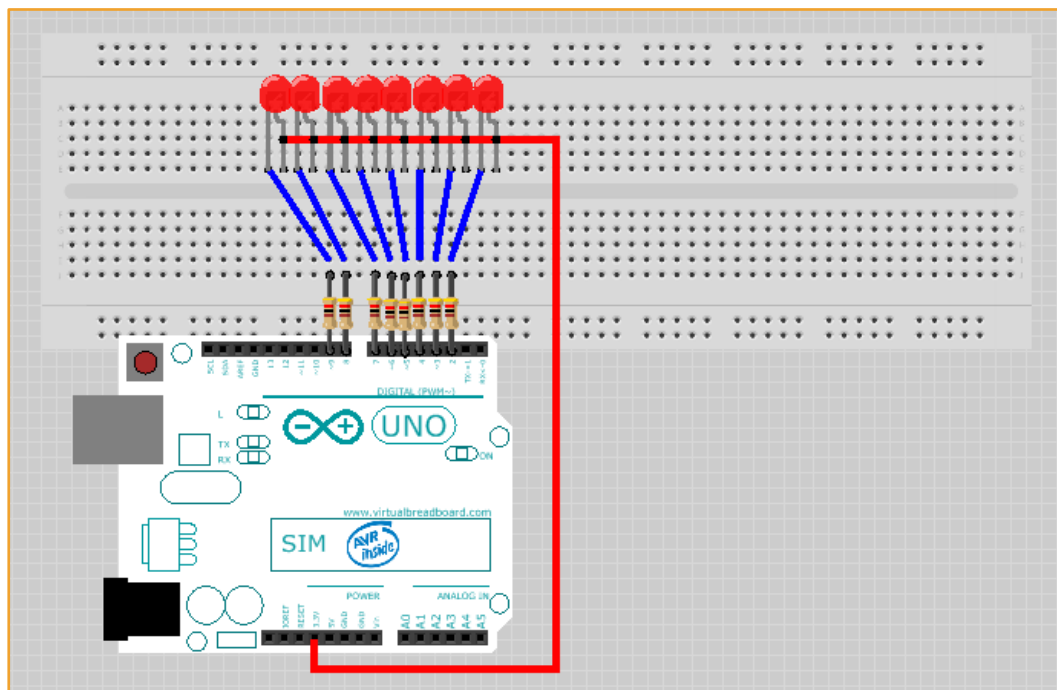
Experimental Purpose

What is a running lights ? We heard of running mountain stream, running river , etc. Just as its name implies, making lights light up like running water. Our purpose is to achieve the effect of a single light glows from left to right, then all the lights glow from left to right and this cycle continues.

Component List :

- ◆ Eight LEDs
- ◆ Eight 1k resistors
- ◆ A number of cables

The wiring is as follows:



Program Principle

Firstly, we set number 2-9 pin to high level, namely the initial state of all LED is putting out, and then switch the number 9 pin to low level, so the far left LED lights up. After delaying 500ms, we put number 8-2 pins for the low level and the other pins keep high level, so that the first round each LED is on for 500ms. In the second round, we set number 9 pin as low level, the first LED lights up, then put all LEDs into a low level from left to right, now all the lights glow, and this cycle continues. Its effect looks like "running water". If you want to make the led flashing fast, you can reduce the delay time, but if the delay time is too s

hort, it looks like all LEDs have been lit all the time in our eye ; If you want the LEDs slowly flashing, you can increase the delay time, but if time delays too long, you may fail to see the flickering effect.

The program is as follows:

```
int led_array[8] = { 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 } ;
int flash_speed = 500 ;
void led_flash(void) /* flash led form left to right one by one */
{
    int i ;
    for( i = 0 ; i < 8 ; i++ )
    {
        digitalWrite(led_array[i],LOW);
        delay(flash_speed);
        digitalWrite(led_array[i],HIGH);
    }
}
/* turn on all led form left to right */
void led_turn_on(void)
{
    int i ;
    for( i = 0 ; i < 8 ; i++ )
    {
        digitalWrite(led_array[i],LOW);
        delay(flash_speed);
    }
}
```

```

/* turn off all led */
void led_turn_off(void)
{
    int i ;
    for( i = 0 ; i < 8 ; i++ )
    {
        digitalWrite(led_array[i],HIGH);
    }
}

void setup() {
    // put your setup code here, to run once:
    int i ;
    Serial.begin(115200);
    for( i = 0 ; i < 8 ; i++ )
    {
        pinMode(led_array[i],OUTPUT);
        digitalWrite(led_array[i],HIGH); // set led control pin default HIGH
    }
    // turn off all LED
}

void loop() {
    // put your main code here, to run repeatedly:
    Serial.println("start flash led !");
    led_flash();
    led_turn_off();
    led_turn_on();
}

```

Do you see the effect or not? Surprise? What are you waiting for? Please hurry to make a heart-shaped running light to girlfriend.

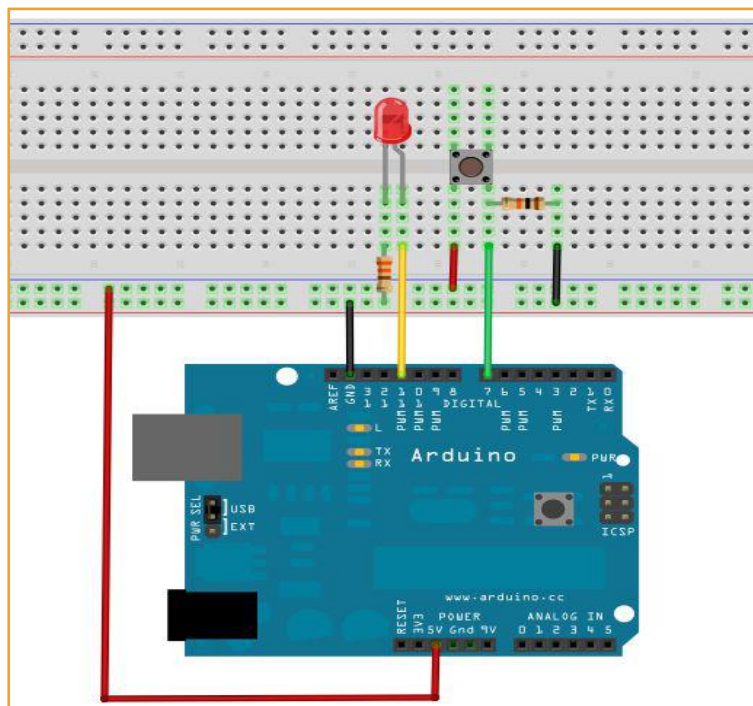
Key Experiment

The meaning of digital I/O port is the INPUT and OUTPUT interface, during the former LED running lights experiment, we only use OUTPUT function of the GPIO. Let's try to use the INPUT function of I/O in the Arduino, namely , the function is to read output value from an external device in this experiment. We use a key and an LED lamp to complete an experiment of using INPUT and OUTPUT as a combination.

Circuit Connection

We connect the button to number 7 interface, and the red light to number 11 interface (All 0-13 digital I/O interfaces in Arduino controller can be used to connect keys and lights, but try not to choose number 0 and 1 interfaces, because 0 and 1 are for interface function reuse, they are also used as a serial communication interface in addition to the function of I/O port. When downloading programs, the device is communicating with PC. so should keep number 0 and 1 interfaces dangling. In order to avoid the trouble of plugging lines, we don't choose number 0 and 1 interfaces),

Wiring the circuit according to the schematic diagram below.



Experiment Principle

Analyzing the circuit we can know, when pressing the button, number 7 interface is on high level, it leads to the output of number 11 is on high level, this can make the light on. When number 7 interface reads as low level, number 11 output stays at low level, now the light is off. The principle is the same as above.

```
int led_out = 11 ;           //GPIO 11  LED pin
int keypad_pin = 7 ;         // GPIO 7 key pin
int value;
void setup()
{
    pinMode(led_out,OUTPUT);   // init led pin output
    pinMode(keypad_pin,INPUT); // init key pin input
}
void loop()
{
    val = digitalRead(keypad_pin); // read key pad pin vaule
    if( value == LOW )
    {
        digitalWrite(led_out,LOW); // if key value is down  turn on LED
    }
    else
    {
        digitalWrite(led_out,HIGH); // if key value is down  turn off LED
    }
}
```

When downloaded the program, the light with keys experiment is done. The principle of the experiment is very simple, widely used in all kinds of circuits and electric appliances. In our real life, it is also not difficult to find this in a variety of devices, such as everyone's cell phone, when pressing a random button, the backlight will be on, clicking on a elevator's button, indicator lights on the elevator will light up, and so on.

AD analog-to-digital conversion

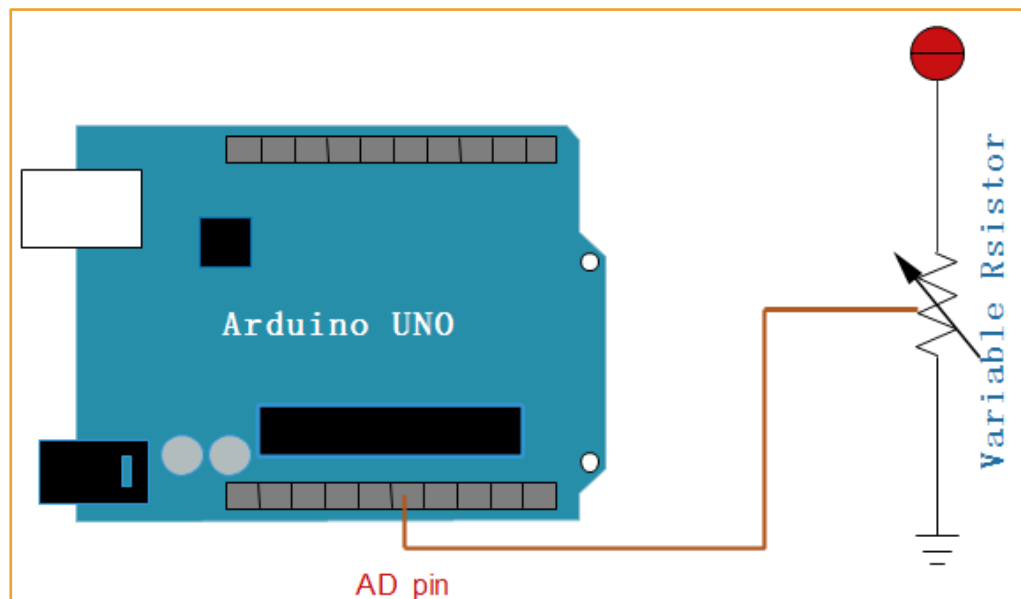
We often hear A/D or D/A conversion in the professional vocabulary, so what is the A/D and D/A? A/D (Analog to Digital) is the device to convert Analog signal into Digital signal, then DA converts Digital signals into Analog signals.

Arduino has six analog interfaces numbered from 0 to 5, the six interface can also be interface function reuse. In addition to the analog interface function, the six interface can be used as digital interfaces and numbered from digital 14-19. After the simple understanding, let us begin our experiments below. Potentiometer is a kind of well-known typical analog value output element, it will be used to complete this experiment.

Component List:

- ◆ 10 k range potentiometer
- ◆ Breadboard * 1
- ◆ Breadboard jumper 3

In this experiment, we'll convert resistance value of potentiometer into analog value and read it out, then the value will be displayed on the screen, this is also a very application example for us to grasp in order to complete our experiments required in the future.



Experiment Principle

Through the function `analogRead()`, statements can read out the value of analog interface. Arduino 328 takes A/D sampling by 10 bit, so the analog value range is

$0 \sim 2^{10} - 1$ (0-1023), the number is just the value of the AD, it needs to be converted into the actual voltage value, so we will use the following formula to calculate:

$$V_R = \frac{Value}{2^{10} - 1} \times V_{DD}$$

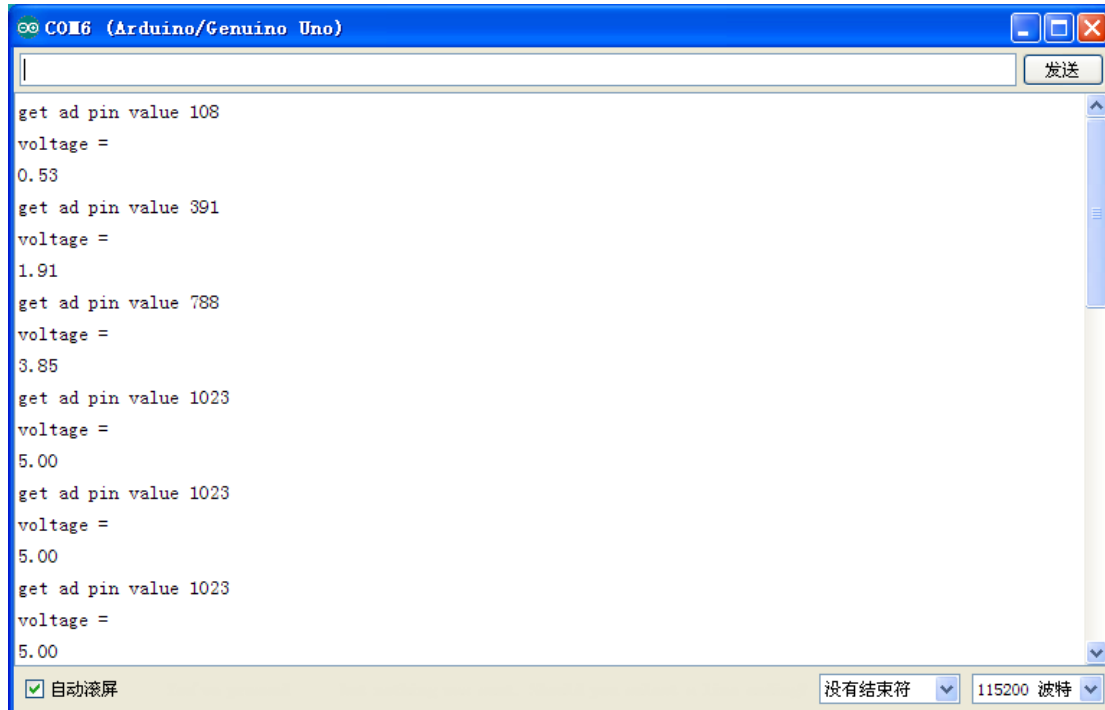
V_R : real voltage

Value : Sampled AD value

V_{DD} : AD reference voltage value

The reference source program is as follows:

```
int ADPIN = 0 ;
int LEDPIN = 13 ;
int value = 0 ;
float voltage = 0.0 ;
void setup()
{
    pinMode(ADPIN, INPUT); // define ADPIN input LEDPIN output
    pinMode(LEDPIN, OUTPUT);
    Serial.begin(115200); //Serial Baud rate is 115200
}
void loop()
{
    digitalWrite(LEDPIN, HIGH); // light on led
    value = analogRead(ADPIN); // read analog pin raw data
    voltage = ( (float) value ) / 1023 ;
    voltage = voltage * 5 ; // convert analog raw data to real voltage = (analog/1023)*5
    Serial.print("get ad pin value "); //printf Analog pin value
    Serial.print(value);
    Serial.println("\nvoltage = ");
    Serial.println(voltage);
    delay(1000);
    digitalWrite(LEDPIN, LOW); //turn off led
}
```

```
get ad pin value 108
voltage =
0.53
get ad pin value 391
voltage =
1.91
get ad pin value 788
voltage =
3.85
get ad pin value 1023
voltage =
5.00
get ad pin value 1023
voltage =
5.00
get ad pin value 1023
voltage =
5.00
```

This experiment is done here. Now when you rotate the potentiometer knob , you will see numerical changes on the screen. This method of reading the analog value will always accompany us, it is also our common function, because most of sensors output analog value, we read the analog value and do corresponding algorithm processing, then it can be applied to the function we need to implement.

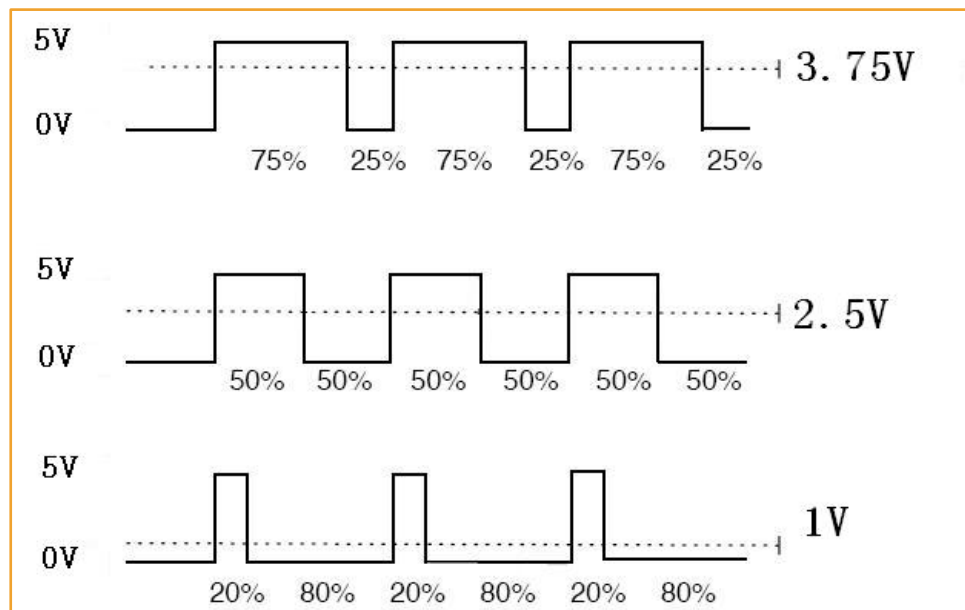
Class 5

PWM Experiment

Pulse Width Modulation is the full name of PWM. It is an analog signal level for digital encoding. Due to computers cannot output analog voltage but only 0 or 5v digital voltage value, we can apply the method of modulating duty cycle of square wave to encode a specific analog signal level through high resolution counter.

PWM signal is still digital, because in any given moment, amplitude of DC power supply is either full 5 v (ON) or 0 v (OFF). Voltage or current source is added to the analogue load through a (ON) or (OFF) repeat pulse sequence. That is when the DC power supply was added to the load, which is broken when power supply was disconnected. As long as there is enough bandwidth, any analog

value can code via PWM. The output voltage value is calculated through on-off time. Output voltage = (turn-on time/pulse time) * maximum voltage value



- 1、 Amplitude of pulse width variation (minimum/maximum)
- 2、 Pulse period (The reciprocal of the number of pulse frequency in 1 second)
- 3、 The voltage level (for example: 0 v to 5 v)

Arduino controller has six PWM interfaces, namely, digital interface 3, 5, 6, 9, 10, 11 (marked with silk-screen on the board ~), the PWM of Arduino don't have a lot of complicated operation, the output frequency is fixed at 50 Hz, application interface is analogWrite (pin, value).The first parameter is the pin, the second parameter value(0 ~ 256) corresponds to the average output voltage (0 ~ 5 V). Say the second parameter is 128, then the duty cycle of PWM output is 50%, the average voltage is 2.5 V.

Experiment Purpose

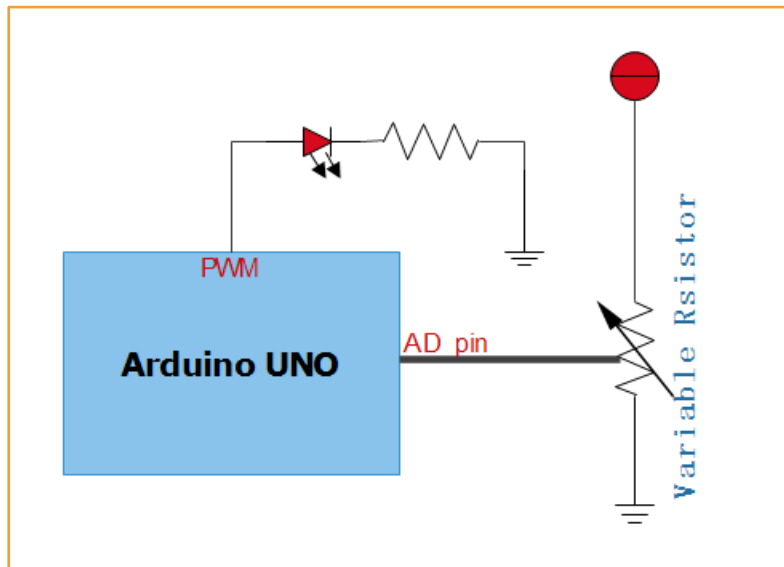
We have done Button-controlling lights experiment before, that is the experiment of digital signal controlling digital interface, and we also realized the potentiometer the experiment. Now we are to complete a experiment of light controlled by potentiometer.

Component List:

- ◆ Potentiometer module*1
- ◆ Red M5 in-line LED*1

- ◆ 220Ω in-line resistor
- ◆ Breadboard* 1
- ◆ Breadboard jumper* a sheaf

The Variable Rsistor connect to Arduino anlog input port ,led connect to Arduino pwm port .



In the process of writing programs, we constantly read the collected value of AD interface through `analogRead (pin)` and convert it into voltage values, convert it into duty cycle by 256 scale after that, and `analogWrite (pwm_pin, val)` function to export the average voltage value. The change of the voltage can be perceived through the brightness of the LED.

```
int ADPIN = 0 ;
int PWM_LEDPIN = 10 ;
int value = 0 ;
float voltage = 0.0 ;
void setup()
{
    pinMode(ADPIN, INPUT); // define ADPIN input PWM_LEDPIN output
    pinMode(PWM_LEDPIN, OUTPUT);
    Serial.begin(115200); //Serial Baud rate is 115200
}
void loop()
{
    value = analogRead(ADPIN); //read analog pin raw data
    voltage = ( ( float )value )/1023 ;
    value = (int)voltage * 256 ; //covert to voltage to PWM duty cycle
    analogWrite(PWM_LEDPIN,value);
    delay(1000);
}
```

Buzzer Experiment

Some appliances often buzz when in an electric state, this is actually from a buzzer, and the annoying bell at school is but a larger buzzer. There are two kinds of buzzers. One is active buzzer, the other is passive buzzer. “active” and “passive” don’t mean the common power source, but a buzzer with or without internal oscillators. Active buzzer will buzz as long as you electrify it, but the frequency is fixed. Passive buzzer, buzzer without internal oscillators, will not buzz when electrified internal oscillators, it requires 2~5 kHz square wave to actuate, then wave forms in different frequency can buzz with corresponding sound.



Active buzzer

Passive buzzer

Experiment Purpose

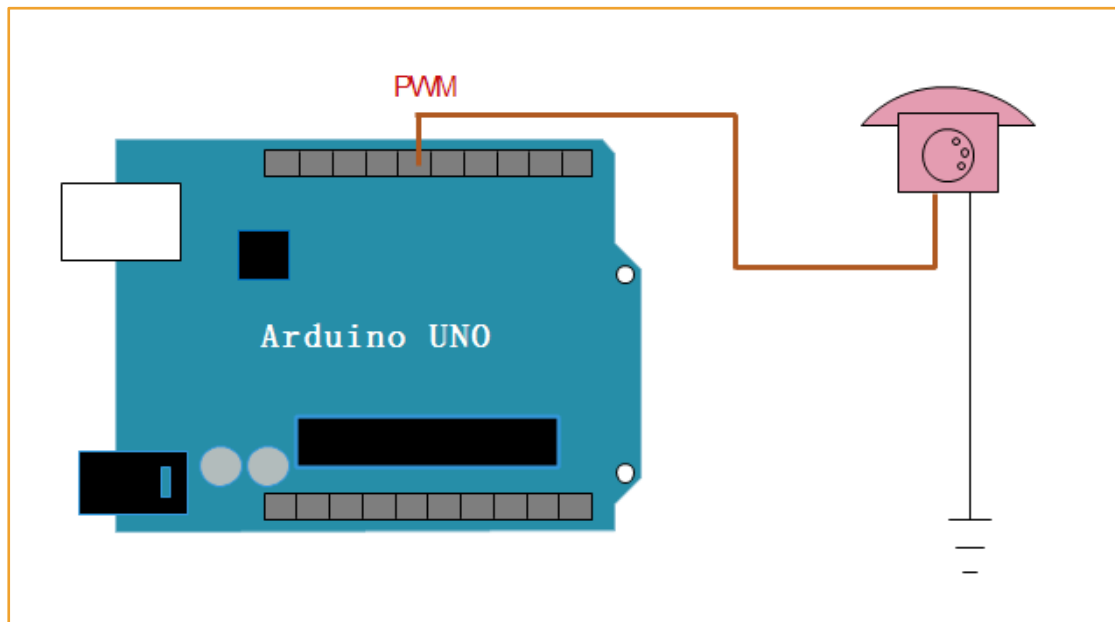
Arduino can be used to create a lot of interactive work, the most common and most commonly is the display of sound and light. We have been using LEDs in experiments before, now we use PWM to drive buzzer to play sound of two frequencies. As long as the frequency matches the music score, we can hear wonderful music.

Component List:

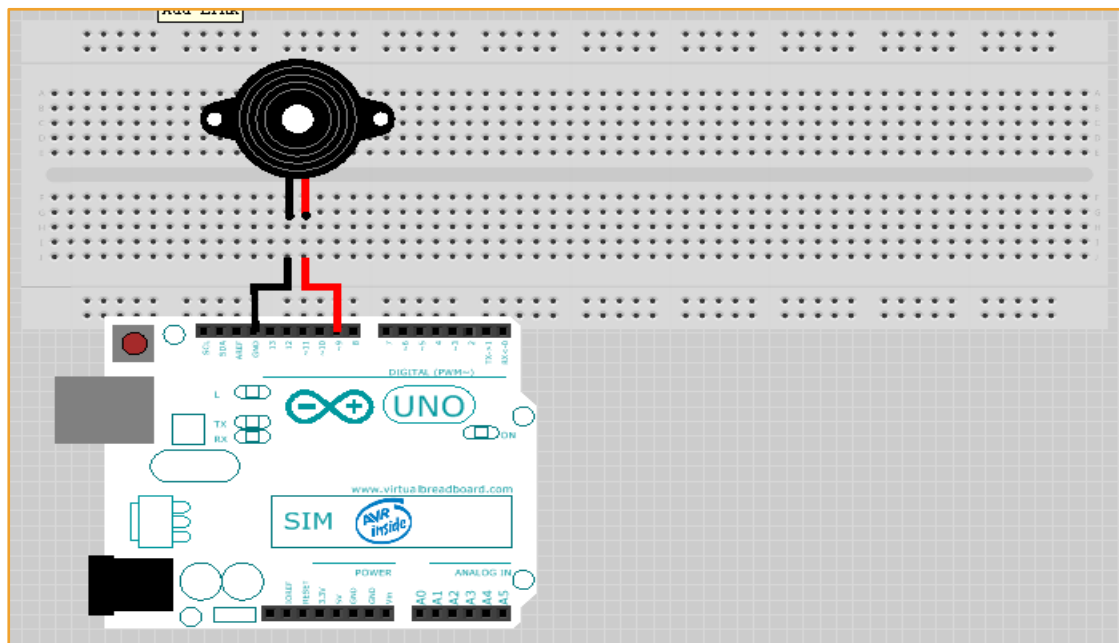
- ◆ Passive buzzer*1
- ◆ Key*1
- ◆ Breadboard* 1

- ◆ Breadboard jumper* a sheaf

The wiring of schematic diagram:



Wiring the circuit according to the following schematic diagram.



Notice that a buzzer has both a cathode and an anode. We can see the buzzer with two kinds of wiring, red and black, in the right physical diagram below.

The connection of circuit and programming are quite simple, the program is similar to the former. Due to the control interface in the buzzer is also digital interface, high and low level from output will control the sound of the buzzer.

```

int buzzer=8;    // set buzzer out pin
void frequency_1(void)    // 1k HZ
{
    int i ;
    for(i=0;i<80;i++)
    {
        digitalWrite(buzzer,HIGH) ;
        delay(1) ;
        digitalWrite(buzzer,LOW) ;
        delay(1) ;
    }
}
void frequency_2(void)    // 500 HZ
{
    int i ;
    for(i=0;i<100;i++)
    {
        digitalWrite(buzzer,HIGH) ;
        delay(2) ;
        digitalWrite(buzzer,LOW) ;
        delay(2) ;
    }
}
void setup()
{
    pinMode(buzzer,OUTPUT) ;
}
void loop()
{
    frequency_1() ;
    delay(100) ;
    frequency_2() ;
}

```

Once the program is download, we can hear the sound of two kinds of frequencies from the buzz.

The Sensor

Class 7

The Temperature Sensor Experiment

There are a lot of scenarios in practice require to measure temperature. In order to accurately measure temperature, it would require the temperature sensor, mercury thermometer is for body temperature measurement, PT100 / PT1000 are generally used to measure temperature of industrial instrumentation, LM35, 18B20 is commonly used in daily life to take temperature, this section will be based on LM35 to measure temperature.

LM35 series is a temperature sensor of precise integrated circuit temperature sensor, its output voltage is linearly proportional to the degree of Celsius temperature.

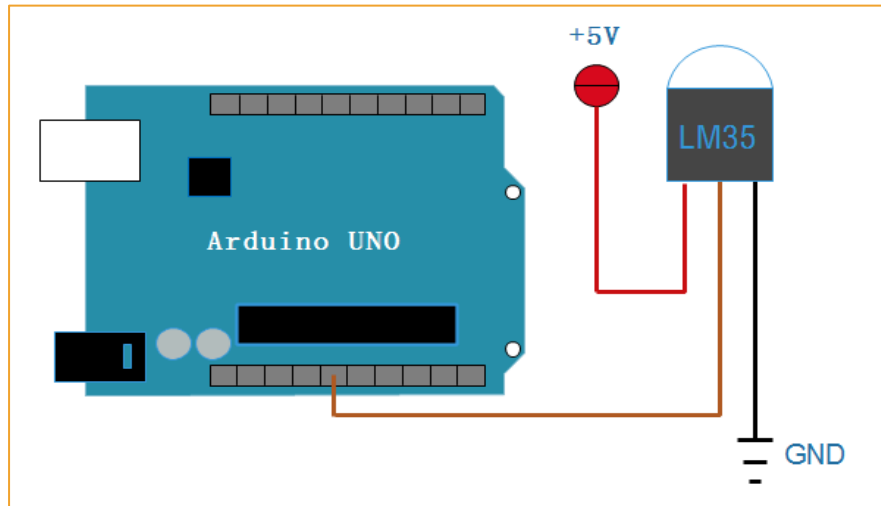
Therefore, LM35 is far more superior than the absolute scale linear temperature sensor. LM35 series sensor has been calibrated when produced, the output voltage corresponds to the degree of Celsius temperature, so it's very convenient for a application. The sensitivity of LM35 series sensor is $10.0 \text{ mV} / ^\circ\text{C}$, the precision is between $0.4 ^\circ\text{C}$ and $0.8 ^\circ\text{C}$ ($-55 ^\circ\text{C}$ to $+150 ^\circ\text{C}$ temperature range), and it is also with high reproducibility, low output impedance. Linear output and internal calibration accuracy make the readout or controlling circuit interfaces easy-to-use, it can work in single supply or the positive-negative power supply, and with the following features:

- *It can be directly calibrated under Celsius temperature*
- *+ $10.0 \text{ mV} / ^\circ\text{C}$ linear scale*
- *It can ensure the precision of $0.5 ^\circ\text{C}$ ($25 ^\circ\text{C}$)*
- *The rated temperature range is from $-55 ^\circ\text{C}$ to $+150 ^\circ\text{C}$*
- *It can be applied in long-distance*
- *Working voltage widely ranges from 4V to 30V*
- *Low power consumption, less than $60 \mu\text{A}$*
- *In the still air, its self-heating effect stays low, less than $0.08 ^\circ\text{C}$*
- *The nonlinear data is only plus or minus $1/4 ^\circ\text{C}$*
- *When passing 1mA current through it, the output impedance is only 0.1Ω*

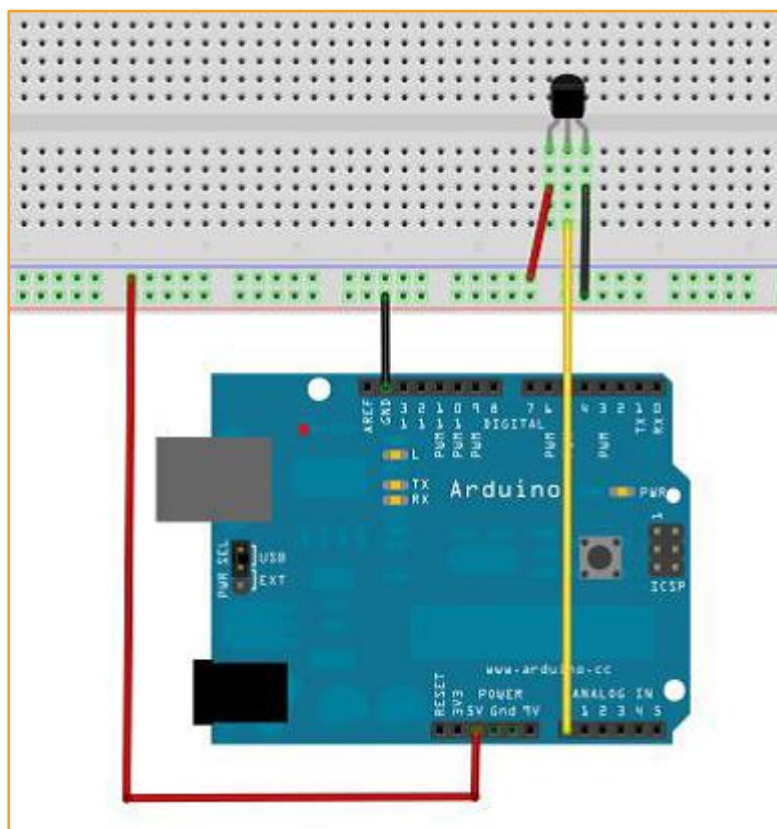
Component List:

- ◆ Direct plug-in LM35*1
- ◆ Breadboard* 1
- ◆ Breadboard jumper* a sheaf

Wiring the circuit according to the following schematic diagram.



The physical diagram is as follows:



Experiment Principle

Arduino collects the output value of LM35 through analogRead () function every 1 second. Firstly, we get the actual voltage by A/D analog-to-digital conversion formula.

$$V_R = \frac{Value}{2^{10}-1} \times V_{DD} \quad \text{即} \quad V = V_{ad} * 5 / 1023 \quad (5V)$$

Secondly, According to LM35 sensor precision: Temp = Vad (V) * 100 (°C / V), we can get the corresponding temperature value.

```
int Temp_Pin = 0;           // analog pin line LM35 numble 2 wire
int val;
int dat;
float voltage = 0.0 ;
void setup()
{
    Serial.begin(115200);    //init serial Baud rate 115200
}
void loop()
{
    val = analogRead(Temp_Pin);    // read analog raw data
    voltage = ( ( float )val )/1023 ;
    voltage = voltage * 5 ;        // convert analog value to real voltage
    dat = voltage * 100;          // convert voltage to temprature
    Serial.print("Current Temp : ");
    Serial.print(dat);
    Serial.println("C");
    delay(500);                 // Delay 0.5 s
}
```

Photosensitive Light Experiment

As we all know, the voice control lamp in a corridor has a sensor in addition to the voice control, that is photosensitive sensor.

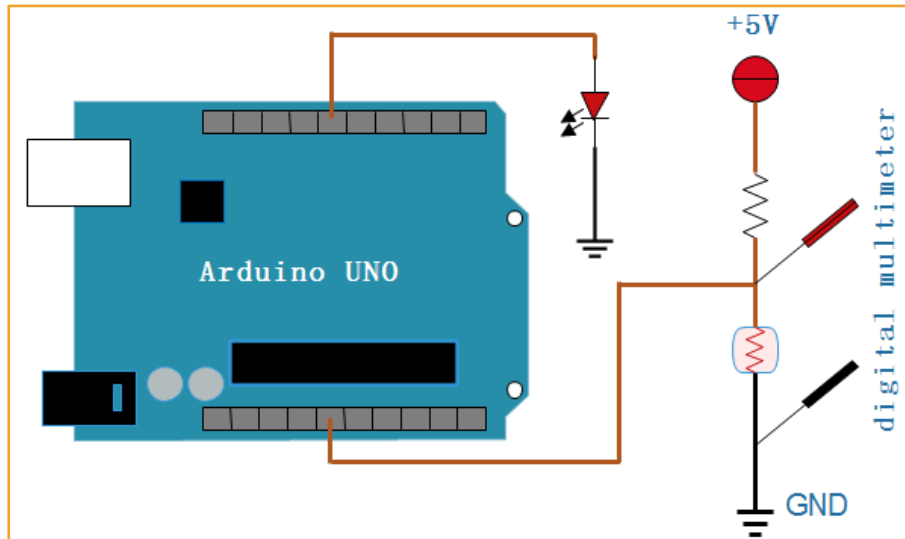
Photovaristor) is also known as photosensitive resistor. It(photoresistor or light - dependent resistor, which is abbreviated as LDR), is commonly made of cadmium sulphide. When the incident light rises, the resistance will reduce; the incident light weakens, the resistance will increase. Photovaristor is commonly used in light measurement, controlling and conversion(the change between light and electricity) would change (changes in the light into electricity), it also can be widely applied in all kinds of light-controlled electric circuit, say, the control and regulating of lamp as well as optical switch.

We first carry out a relatively simple experiment of using Photovaristor. Since photovaristor is an element which can be controlled by the intensity of light, naturally it requires to read analogue value via analog interface. According to the PWM interface experiment before, we can change the potentiometer to a photovaristor, then when changing the intensity of the light, the brightness of LEDs will shift corresponding.

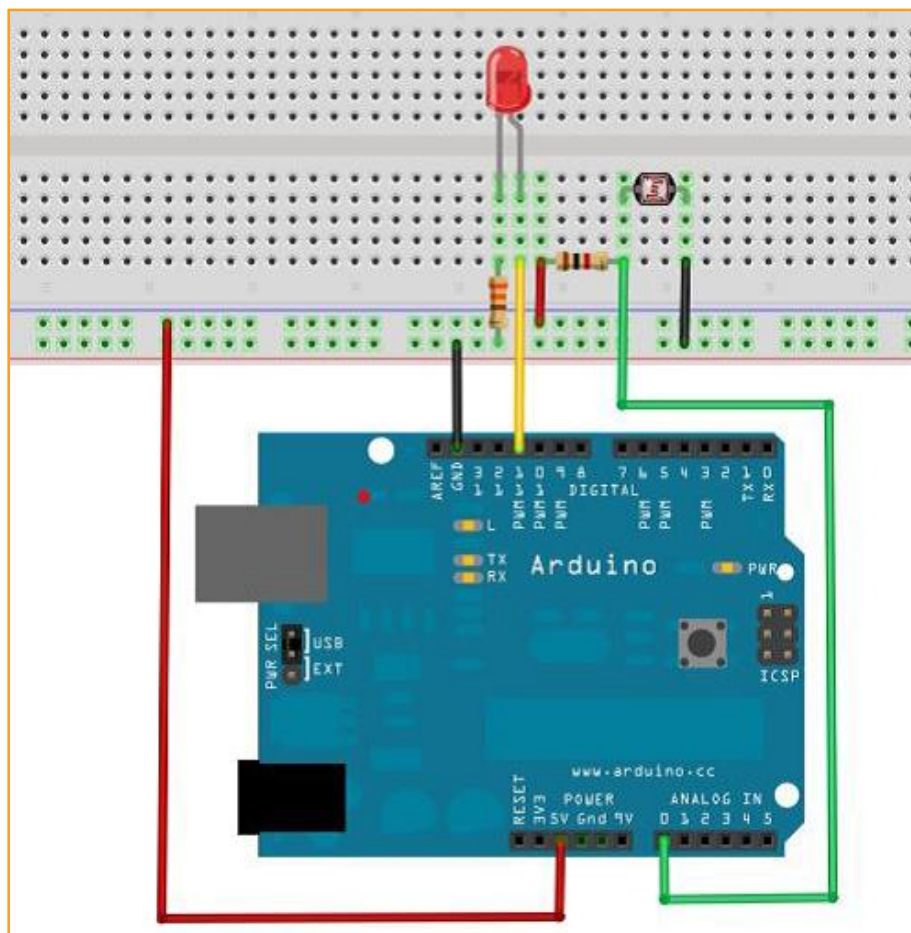
Component List:

- ◆ Photovaristor*1
- ◆ Red M5 in-line LED*1
- ◆ 10k Ω in-line resistor
- ◆ 220 Ω in-line resistor
- ◆ Breadboard* 1
- ◆ Breadboard jumper* a sheaf

Wiring the circuit according to the following schematic diagram.



Now we measure the photovaristor with a multimeter, then cover the photovaristor and we can clearly see the change of the resistance by lighting.



The reference source program is as follows:

```

int ADPIN = 0 ;
int LEDPIN = 13 ;
int value = 0 ;
float voltage = 0.0 ;
void setup()
{
    pinMode(LEDPIN,OUTPUT);
    Serial.begin(115200);    //Serial Baud rate is 115200
}
void loop()
{
    value = analogRead(ADPIN);
    voltage = ( ( float )value )/1023 ;
    value = (int)voltage * 256 ;           //convert voltage to value
    analogWrite(LEDPIN,value);
    delay(1000);
}

```

Class 9

Fire Alarm Experiment

In the public places, say, hotels, buildings and other places are all equipped with fire alarm, then how does it perceive a fire? As we know, when the fire break out, there will be particularly strong infrared, whether the device can detect fire by infrared.

Photo of Flame Sensor



Working Principle

In the spectrum, we refer to the light whose wavelength is from 0.76 to 400 microns as infrared, it is invisible. All materials which are above absolute zero (273.15

°C) can produce infrared. It is called thermal radiation in Modern physics. We know before that a photovaristor without light, there are weak reverse leakage current (dark current), the light-sensitive tube is not conducting at this moment. When struck by light, saturated reverse leakage current will rise immediately, then forms the photocurrent. It increases along with the change of incident light intensity within a certain range. Between the principles of infrared receiving tube and photovaristor, the only difference is that infrared receiving tube is not sensitive to visible light, only to infrared light. When there is no infrared light, the reverse leakage current is quite weak, and the infrared sensor is not conducting; when there is infrared, there will be photocurrent, and the on-resistance reduces along with the increase of infrared intensity. Given the flame-sensitive characteristics of flame sensors, flame can be detected by specially-made infrared receiving tube. As the flame grows, the on-resistance of infrared receiving tube will reduce.

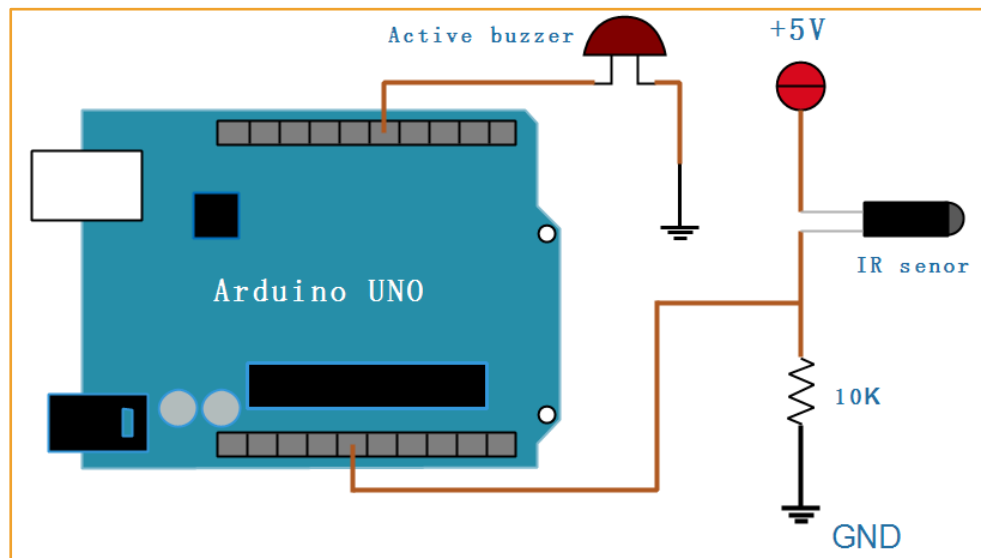
The Wiring of Flame Sensor

The short lead of infrared receiving triode is the cathode, the long is the anode. We link the 5v interface to the cathode according to the diagram below, then connect 10k resistor to the anode, the other end of the resistor is left for the GND interface. At last, we insert a jumper into the anode column of the flame sensor, and the other side of the jumper links to analog interface.

Component List:

- ◆ Flame sensor*1
- ◆ Buzzer*1
- ◆ 10k resistor*1
- ◆ Some jumper

The wiring of schematic diagram:



The wiring of physical map:

Connecting the buzzer to number 8 interface, The flame sensor to number 5 in terface to complete the whole wiring of the experiment.

Experiment Principle

The voltage value of analog interface is changing when there exists two kinds of circumstances--with or without the approaching of flame. Actually, if we measure the flame sensor without flame approached by a multimeter , the voltage value o f analog interface is about 0.3 V; When approaching the sensor with flame, the v oltage value is about 1.0 V. The closer the flame approaches, the higher the volta ge value is.

So in the beginning of the program, we can firstly store a voltage value when no flame, then continuously read the voltage value j of analog interface. Carrying o ut the formula $k = j - I$, then we compare k with 0.6v. For fear of error we are s ure flame actuates the buzzer if the difference value k is above 0.6v (Binary valu e is 123) five times in a row; if the difference is less than 0.6v, the buzzer stays silent.

Program Code

```

int fire_pin = 0 ;           // define analog 0 pin for fire-sensor pin
int buzzer = 9 ;             // buzzer driver pin
int val = 0;
int count = 0 ;
void setup()
{
    pinMode(buzzer,OUTPUT);    // buzzer pin is output
    pinMode(fire_pin,INPUT);    // fire-sensor pin is input
    Serial.begin(115200);       // init baud rate is 115200
    digitalWrite(buzzer,LOW);   // buzzer default value is 0
}
void loop()
{
    val = analogRead(fire_pin); // get fire-sensor analog value
    Serial.println(val);
    if( val > 600 )               // get value > 600 count add
    {
        count++;
    }else
    {
        count = 0 ;
    }
    if( count >= 5 )              // count > 5 ensure infrared radiation found and give an alarm
    {
        digitalWrite(buzzer , HIGH );
    } else
    {
        digitalWrite(buzzer , LOW ); // disable an alarm
    }
    delay(500);
}

```

This program can simulate the buzzer ringing when there exists fire, and everything is normal when no flame.

Infrared Remote Control

What is an infrared receiving header?

We have introduced an infrared receiving tube in a former passage, it is a kind of sensor which can identify the infrared. The integration of an infrared sensor receives and modulates the 38KHZ infrared. In order to make it in the process of wireless transmission from other infrared signal interference, the infrared remote control usually modulates the signal on a specific carrier frequency, and then launches it by the infrared emission diode. While the infrared receiving device needs to filter out other clutter, receives the specific frequency signal and restores it into a binary pulse code, namely, demodulation.

Working Principle

The built-in receiving tube converts the light signal emitted by an infrared transmitting tube to a weak electrical signal, the signal is amplified through the internal IC, then restored to the original code emitted by the infrared remote control through the automatic gain control, band-pass filtering, demodulation, waveform shaping, and input to the decoding circuit on an electric appliance through the output pin of the receiving header.



Infrared receiver header has three pins: connecting VOUT to analog interface, GND to the GND onboard and VCC to + 5 v.

Experiment Principle

The encoding of a remote control is required to be learned first if we want to decode it. The control code used in this product is: NEC protocol.

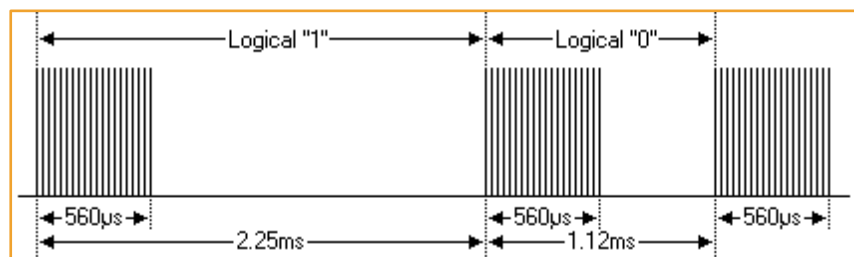
The following will introduce the NEC protocol:

Introduction of NEC

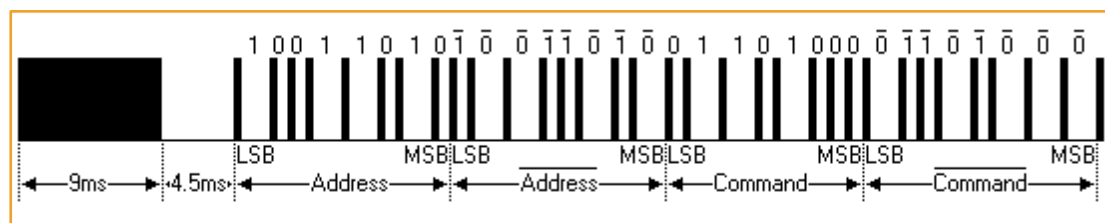
Characteristics:

- 8 address bit, 8 command bit
- Address bit and command bit are transmitted twice in order to ensure reliability
- Pulse-position modulation
- Carrier frequency 38kHz
- Every bit lasts 1.125ms or 2.25ms

The definitions of logic 0 and 1 are as below:



Transmitted pulse which the pressed button released immediately

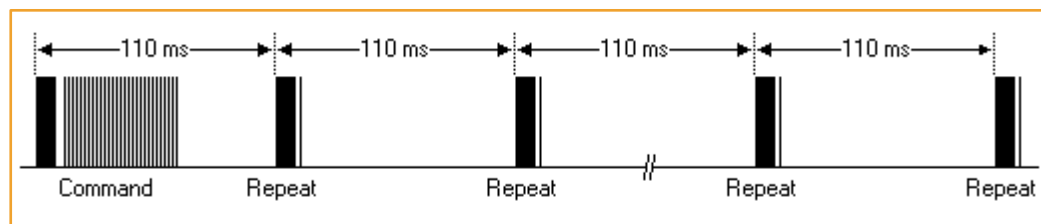


The picture above shows a typical pulse sequence of the NEC protocol. Notice:

The protocol of LSB (least significant) is firstly transmitted. In the above, pulse transmission address is 0x16 and the command is 0x59. A message starts fr

om a 9ms high level, then followed by a 4.5ms low level, and by the address code and command code. The address and command are transmitted twice. All bits flip in the second transmission, this can be used in the confirmation of the received message. The total transmission time is constant, because every bit repeats the flip length. If you are not interested, you can ignore this reliable inversion and expand the address and command every 16 bit as well !

Transmitted pulse which the pressed button last for a while



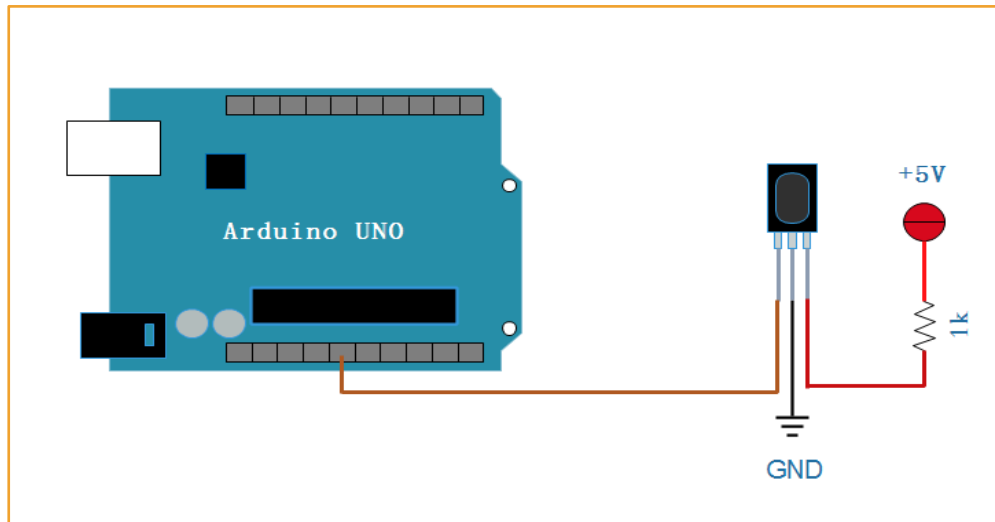
Even a button on the remote control is pressed again, the command is transmitted only once. When the button has been pressing, the first 110ms pulse is same as above, then the same code is transmitted every 110ms. The next repeated code consists of a 9ms high level pulse, a 2.25ms low level pulse and a 560μs high level pulse.

Notice: When the pulse received by the integrative header, the header needs to decode, amplify and shape the signal. So we should notice that the output is high level when the infrared signal is absent, otherwise, the output is low level, so the output signal level is reversed in transmitter. We can see the pulse of receiver through the oscilloscope and understand the program by waveform.

Component List:

- ◆ NEC infrared remote control: 1
- ◆ The integrative infrared receiving header: 1
- ◆ 1k resistor: 1
- ◆ A number of colorful wires

The Wiring of the Circuit:



Experiment Purpose

Decoding all the keys of the remote control by Arduino and displaying them in a serial port. Due to the decoding process is based on the given protocol, and the code is massive, so we use open source libraries to decode and copy IRremote to arduino 1.6.5 - r5 \ libraries.

The link: <https://github.com/shirriff/Arduino-IRremote> to arduino 1.6.5 - r5\libraries.

```
#include <IRremote.h>
int RECV_PIN = 11;
long int value ;
IRrecv irrecv(RECV_PIN);
decode_results results;
void setup()
{
    Serial.begin(9600);
    irrecv.enableIRIn(); // init ir rcv and enable Ir input
    pinMode(12,1);
}
void loop() {
    if (irrecv.decode(&results)) {
        value = results.value;
        Serial.print(value,HEX);
        Serial.print("\n");
        irrecv.resume(); // resume and receive next ir decoder
    }
}
```

Not available for now

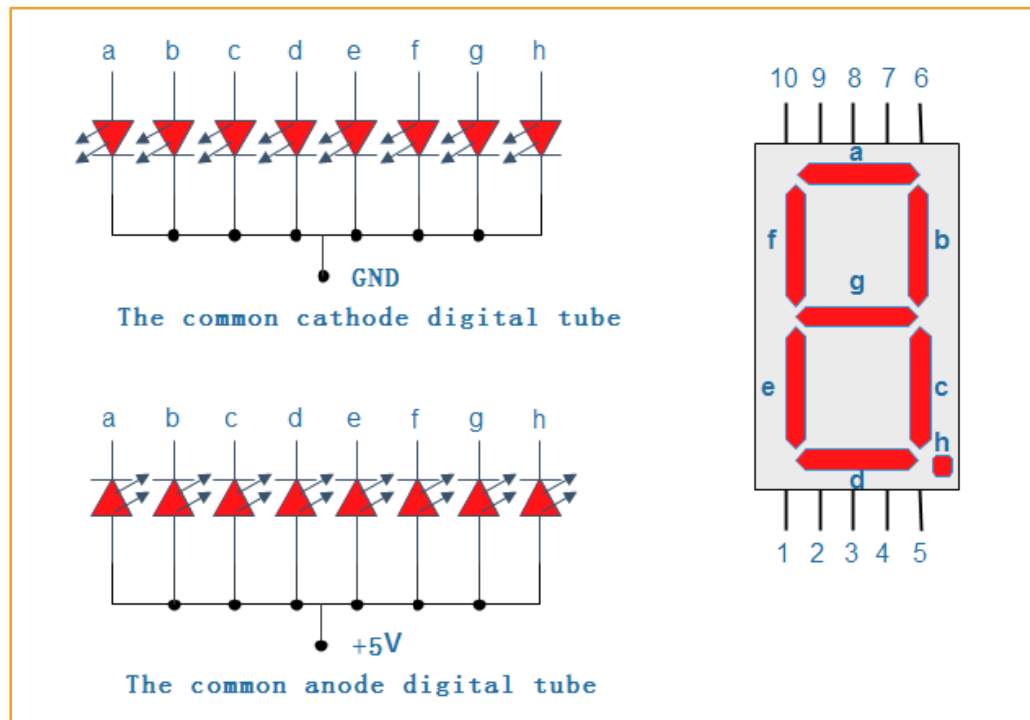
Display

Class 12

8-Bit Digital Display Experiment

Nixie tube is a semiconductor light emitting element, and its basic unit is a light-emitting diode. According to the number of segment, Nixie tube is divided into seven-segment tube and eight-segment tube. Eight-segment tube has one more light-emitting diode unit (one more a decimal point) segment than seven-segment tube.

We will use eight-segment tube in this experiment. Light-emitting diode unit can be divided into common anode digital tube and common cathode digital tube according to the connection mode. The common anode digital tube refers to all anodes of a light-emitting diode linking to +5 V. When the cathode of any one segment of light-emitting diode is low level, the corresponding segment will light up; when the cathode is high level, the segment stays unlighted. The common cathode digital tube refers to all cathodes of a light-emitting diode linking to GND. When the anode of any one segment of light-emitting diode is high level, the corresponding segment will light up, when the anode is low level, the segment stays unlighted as well.



Each segment of a Nixie tube is composed of light emitting diode, so to display different numbers, the principle is that the corresponding LEDs are lit up. Say we want to display Number 0, that means “abcdef” is lightened and the other are turned out, so that we only need to see the corresponding the truth table of displayed number.

Light emitting diode of Nixie tube requires to connect current-limiting resistor, or light emitting diode will be destroyed due to excess of electricity. We will use common cathode digital tube in the experiment.

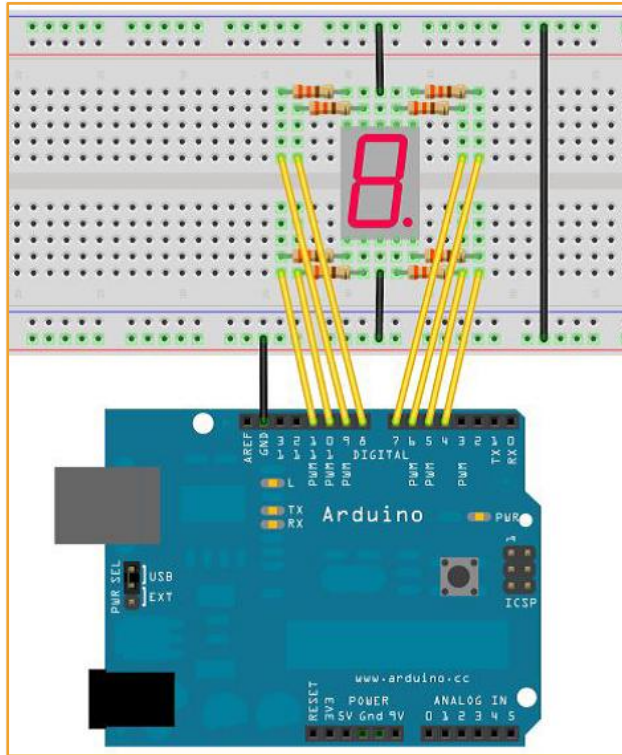
Experiment Purpose

Display Numbers 0-9 on eight-segment tube.

Component List :

- ◆ Eight-segment tube*1
- ◆ 220Ω in-line resistor
- ◆ Breadboard* 1
- ◆ Breadboard jumper* a sheaf

The wiring of physical map according to schematic diagram



Nixie tube has seven segments for displaying numbers and one for decimal point.

When we want to display numbers on the tube, as long as we lighten the corresponding segments. For example, if we want to display Number 1, then “b ” and “c” are lightened. We write a subroutine for each number, and make Nixie tube loop 1 ~ 8 every 2 seconds in the main program.

```

#define LED_A 7 // define Arduino GPIO7 for led a
#define LED_B 6 // define Arduino GPIO6 for led b
#define LED_C 5 // define Arduino GPIO5 for led c
#define LED_D 11 // define Arduino GPIO11 for led d
#define LED_E 10 // define Arduino GPIO10 for led e
#define LED_F 8 // define Arduino GPIO8 for led f
#define LED_G 9 // define Arduino GPIO9 for led g
#define LED_H 4 // define Arduino GPIO4 for led h
char value , dispaly_char ;
char LED_PIN[8] = { LED_A , LED_B , LED_C , LED_D , LED_E , LED_F , LED_
G , LED_H } ;
char LED_Display_Value[][2] =
{
    { '0', 0x3F } ,
    { '1', 0x06 } ,
    { '2', 0x5B } ,
    { '3', 0x4F } ,
    { '4', 0x66 } ,
    { '5', 0x6D } ,
    { '6', 0x7D } ,
    { '7', 0x07 } ,
    { '8', 0x7F } ,
    { '9', 0x6F } ,
    { 0 , 0x00 }
};
void DisplayOff(void)
{
    int i ;
    for( i = 0 ; i < 8 ; i++)
        digitalWrite(LED_PIN[i],LOW);
}
char GetDisplayValue(char Array[][2] , char DisplayChar )
{
    int i = 0 ;
    for( i = 0 ; i < 10 ; i++)
    {
        if( Array[i][0] == DisplayChar )
            return Array[i][1] ;
    }
    return 0 ;
}

```



```

void LED_Display ( char ch)
{
    int i ;
    for( i = 0 ; i < 8 ; i++)
    {
        if( ch & ( 1 << i ) )
        {
            digitalWrite(LED_PIN[i] , HIGH);
        }else{
            digitalWrite(LED_PIN[i],LOW);
        }
    }
}

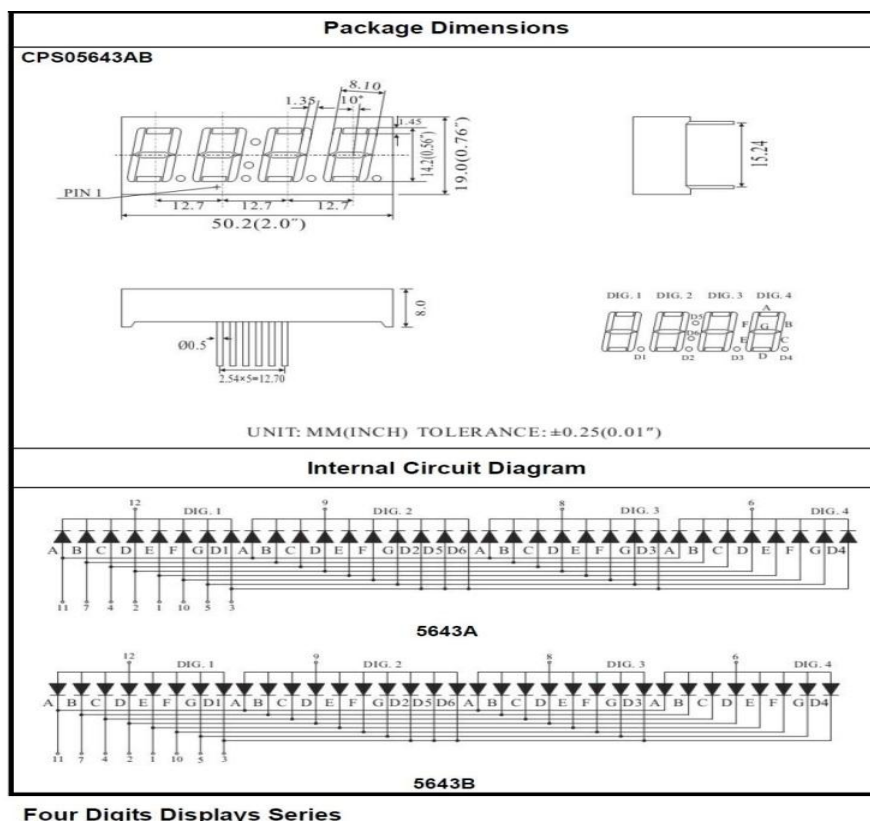
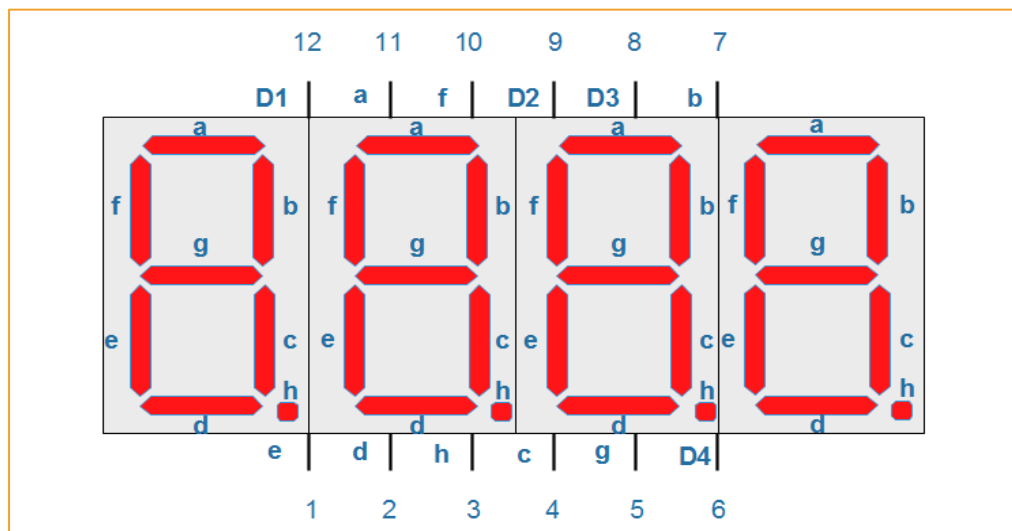
void setup()
{
    int i;
    Serial.begin(115200);
    for( i = 0 ; i < 8 ; i++ )
        pinMode( LED_PIN[i] ,OUTPUT ) ;           // set all led diplay array pin
output mode
    DisplayOff();
}

void loop()
{
    Serial.println("please input display char \n");
    value = Serial.read() ;
    dispaly_char = GetDisplayValue( LED_Display_Value , value ) ;
    Serial.print(value);
    Serial.print(dispaly_char);
    if ( dispaly_char != 0 )
    {
        DisplayOff();
        LED_Display( dispaly_char );
    }
}

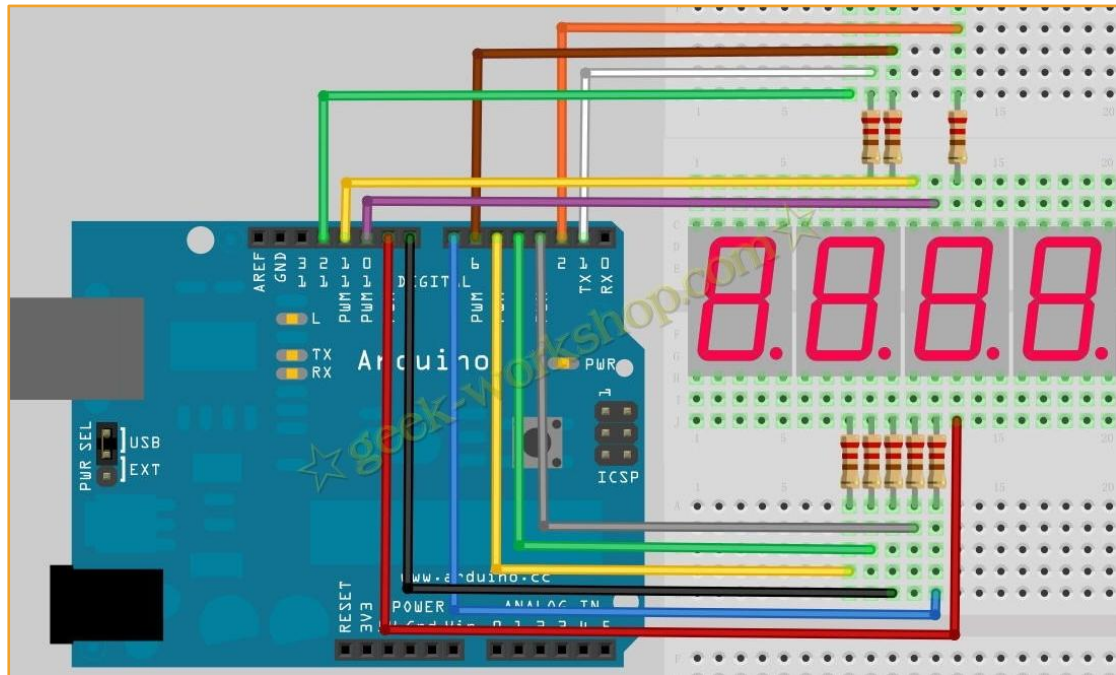
```

4 x 8 Nixie tube timer countdown

We used a eight-segment tube before. When we want to display more than one number, then multidigit tube is required. Here we introduce four digital tube, actually each individual 8-segment tube is almost the same as the tube used above. In this experiment, we will use the Arduino to drive a common anode of our digital tube.



Four digital tube has 12 pins. The upper left is the biggest number 12 pin. Besides the 8-segment we used to display “adbcdfehg”, there are another 4 pins D1, D2, D3, D4 to be used as the “bit” pins. When the “bit” pins of common anode four digital tube is high level, the corresponding tubes light up. The display principle of four digital tube is that constantly scanning D1, D2, D3, D4, and then the corresponding eight-segment tubes will light up in turn. Due to the residual effect of human eye, so it looks like the four digital tube display at the same time.



With the principle introduced above, we now make a simulated countdown time bomb like the movies do. The bomb will exploded in one minute.

Experiment Principle

The most important purpose of this program is how to scan the four digital tube dynamically. In fact, with the single digital tube display experiment before, the display of four digital tube is quite easy. Due to it is attributed to a common anode tube, first of all, we are going to set D1, D2, D3, D4 to low level, all LEDs turn out, then we output the truth table of “adbcdfehg” to the corresponding gpio port, select the corresponding bit pins and scan constantly. How to implement the 1 minute countdown? In the program, we will continuously get the current time through millis () function and determine whether it is greater than 1000ms. If so compared to the time before, the countdown time minus 1, then it is translated into character string that the Nixie tube displays.

```

#define LED_A 1 // define Arduino GPIO1 for led a
#define LED_B 2 // define Arduino GPIO2 for led b
#define LED_C 3 // define Arduino GPIO3 for led c
#define LED_D 4 // define Arduino GPIO4 for led d
#define LED_E 5 // define Arduino GPIO5 for led e
#define LED_F 6 // define Arduino GPIO6 for led f
#define LED_G 7 // define Arduino GPIO7 for led g
#define LED_H 8 // define Arduino GPIO8 for led h
#define LED_D1 9
#define LED_D2 10
#define LED_D3 11
#define LED_D4 12

#define COM_PORT 2 // 1:common negative port 2: common positive port
char LED_PIN[8] = { LED_A , LED_B , LED_C , LED_D , LED_E , LED_F , LED_G , LED_H } ;
char LED_SELECT[4] = {LED_D1 ,LED_D2 , LED_D3 , LED_D4 } ;

char disp[4] = { 0 , 0 , 0 , 0 } ; // display array value
int display_dat = 59, count = 0 ;

char LED_Display_Value[][3] =
{ // vaal negative positive
  { '0', 0x3F , 0xC0 } ,
  { '1', 0x06 , 0xF9 } ,
  { '2', 0x5B , 0xA4 } ,
  { '3', 0x4F , 0xB0 } ,
  { '4', 0x66 , 0x99 } ,
  { '5', 0x6D , 0x92 } ,
  { '6', 0x7D , 0x82 } ,
  { '7', 0x07 , 0xF8 } ,
  { '8', 0x7F , 0x80 } ,
  { '9', 0x6F , 0x90 } ,
  { 0 , 0x00 , 0xFF }
};

void DisplayOff(void)
{
  int i ;
  for( i = 0 ; i < 8 ; i++)
    digitalWrite(LED_PIN[i],LOW);
  for( i = 0 ; i < 4 ; i++)
    digitalWrite(LED_SELECT[i],LOW);
}

```

```

char GetDisplayValue(char Array[][3] , char DisplayChar )
{
    int i = 0 ;
    for( i = 0 ; i < 10 ; i++)
    {
        if( Array[i][0] == DisplayChar )
            return Array[i][COM_PORT] ;    //return the common positive port value
    }
    return 0 ;
}

void LED_Display ( char ch)
{
    int i ;
    for( i = 0 ; i < 8 ; i++)
    {
        if( ch & ( 1 << i ) )
        {
            digitalWrite(LED_PIN[i] , HIGH);
        }else{
            digitalWrite(LED_PIN[i],LOW);
        }
    }
}

void numble2dis( int numble )
{
    int numble_bit = 0 ;
    int bit_base = 1000 ;
    for( numble_bit = 0 ; numble_bit < 4 ; numble_bit++ )
    {
        if( numble/bit_base != 0)
        {
            disp[numble_bit] = numble/bit_base + '0' ; // int data convert to ASCII
            numble = numble%bit_base ;
        }else
        {
            disp[numble_bit] = '0'; // led display none
        }
        bit_base = bit_base / 10 ;
    }
}

```

```

void setup()
{
    int i;
    for( i = 0 ; i < 8 ; i++ )
        pinMode( LED_PIN[i] ,OUTPUT ) ;           // set all led diplay array pin
    output mode
    for( i = 0 ; i < 4 ; i++ )
        pinMode( LED_SELECT[i] ,OUTPUT ) ;         // set led select output mode
    DisplayOff();
    numble2dis(59);
}
void loop()
{
    int i ;
    if( count++ > 50 )
    {
        display_dat-- ;
        numble2dis(display_dat); // integer convert to ASCII value
        count = 0 ;
    }
    for( i = 0 ; i < 4 ; i++ )
    {
        LED_Display( GetDisplayValue(LED_Display_Value,disp[i]) ) ;
        digitalWrite(LED_SELECT[i] ,HIGH ) ;
        delay(5);
        digitalWrite(LED_SELECT[i] ,LOW ) ;
    }
    if(display_dat == 0 )
        while(1);
}

```

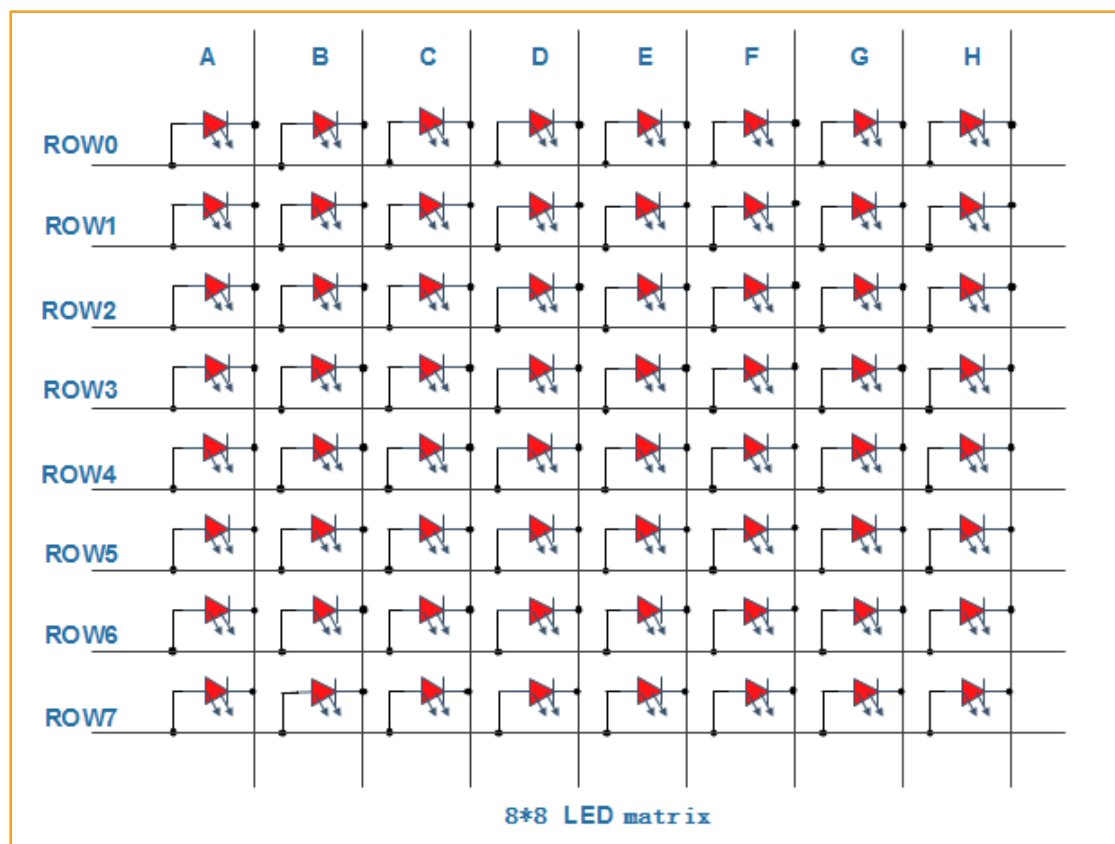
Notice that 4 numeric digits are converted into the value of ascii by numble2dis, say we are going to convert “1234”, this should be as follows:

Loop	numble	bit_base	disp
1	1234	1000	1
2	234	100	2
3	34	10	3
4	4	1	4

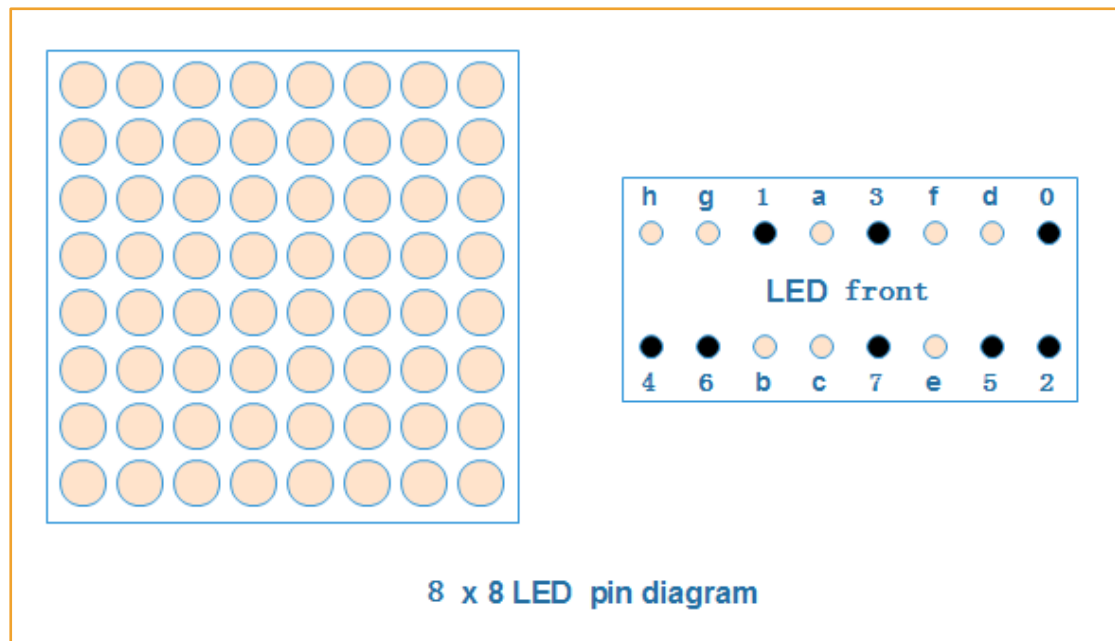
8x8 Dot-matrix Experiment

Now it is not difficult for us to display figures after learning the display of Nixie tube. However, if we want to show a variety of patterns in practice, Nixie tube clearly suffers from inadequate capacity, it requires LED dot-matrix. When you walk in the streets, all kinds of LED neon billboards you see are but $N \times N$ dot-matrix. Now let's take a look of the internal principle of 8×8 dot-matrix.

The schematic diagram of 8×8 dot-matrix



The pins of 8*8 dot-matrix



The graph displays the appearance of 8 X 8 LED dot-matrix and its pins, and the equivalent circuit is shown in figure (1). As long as its X, Y axes are forward biased, the corresponding LEDs will be lightened. For example, if you want to keep the top left LED lighting, just set ROW0 = 1, A = 0. Due to the through-current of LED is low in practice, if the driving voltage of Arduino is 5v, then we need to connect 1k resistor to the ROW pin.

The scanning of 8*8 dot-matrix

LED commonly displays through scanning, divided into three ways in practice:

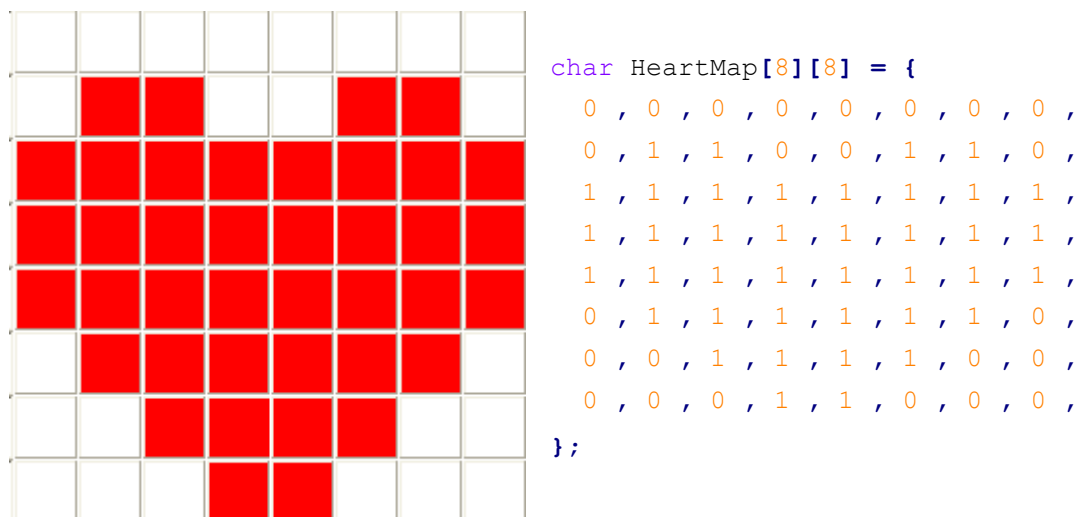
- *Spot scanning*
- *Row scanning*
- *Column scanning*

LED dot-matrix does not require to light up LED one by one, column scanning is more appropriate due to it is attributed to common cathode LED. First, we set the corresponding level of the first column according to the display values. The first column of A will light up when given to low level, the second and the third column will follow if we do the same, then repeating the loop. Now the LED image can be seen due to the visual residual effect of human eyes

The application of 8*8 led dot-matrix

The internal structure and appearance of dot-matrix are as follows. 8x8 dot-matrix consists of 64 light-emitting diodes, and each light emitting diode lies in the intersection of row and column. When the corresponding row is high level and the column is low level, the corresponding diode will be bright. If we want to light up the first diode, we need to set the 9th pin to high level and the 11th to low level; If we want to the first row to light, then the 9th pin needs to be high level and the (13, 3, 4, 10, 6, 11, 15, 16) pins are low level, then the first line will light up; as to the first column, the 13th pin requires to be low level and (9, 14, 8, 12, 1, 7, 2, 5) pins are high level, then the column will light up.

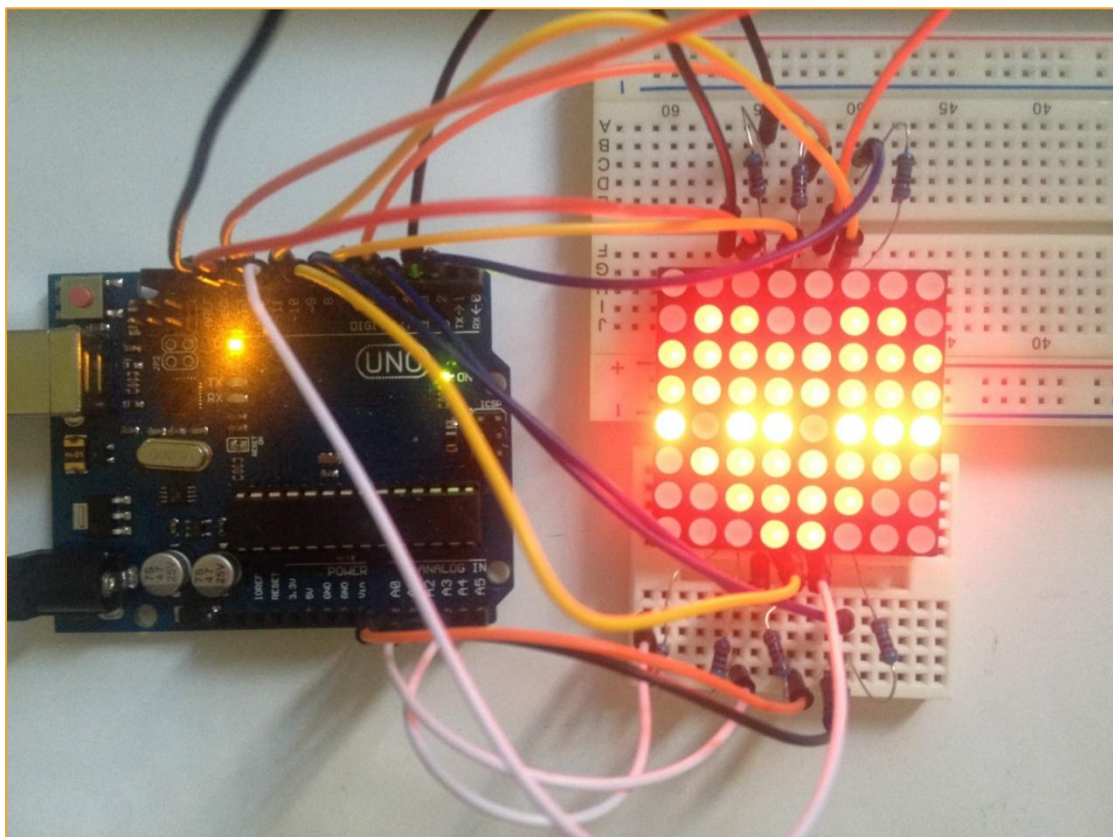
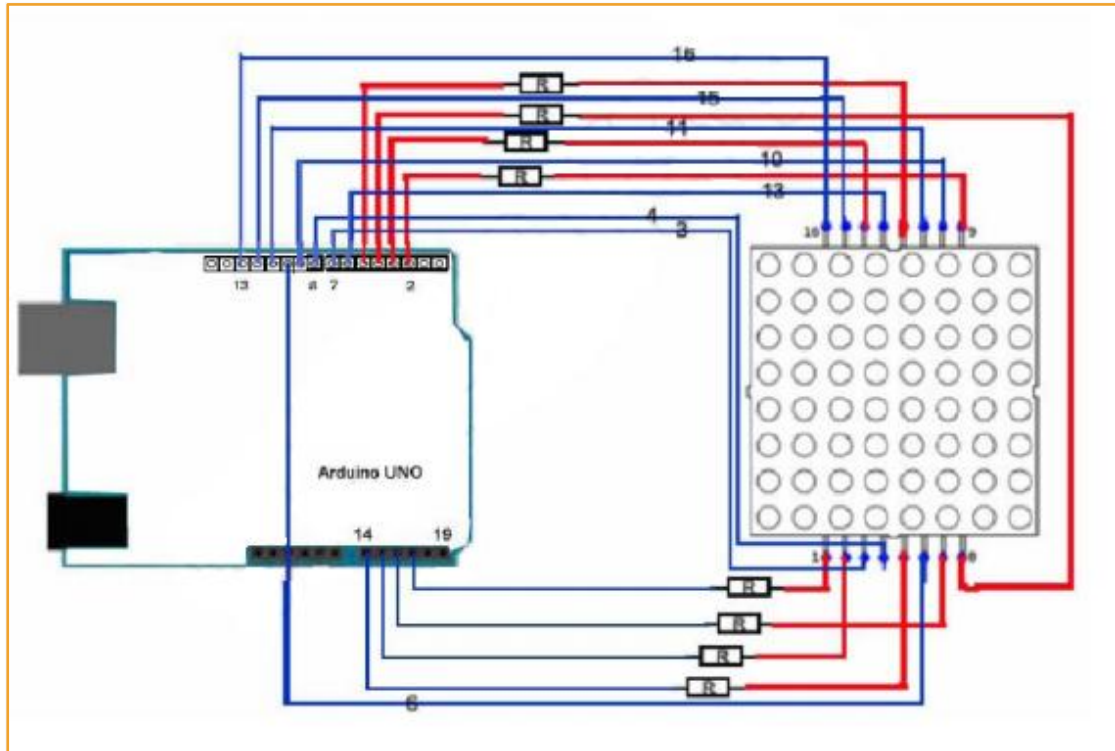
We try to display a heart-shaped figure in the experiment, so we set the red part to high level and the other to low level, The LED will show a heart-shaped figure after scanning dynamically.



Because it is dynamic scanning . We need to pay attention to two points is ghosting and flashing .We need to give each column in the process of scanning the corresponding level, and the common cathode port pull low level will, To scanning the next column digital tube ,we need to pull previous column all low level before. so it can prevent each column interleaved scan of ghosting .We are due to the column scanning, residual effect to human eyes are 25 hz.

but usually sweep frequency 50hz effect is better, so each column of the delay time can't more than $1000/50/8 = 2.5\text{ms}$ we delay set to 2ms effect is better.

The physical map of 8*8 LED dot-matrix is as follows:



```

//the pin to control ROW_
#define ROW_0 2 // the number of the ROW_ pin 9
#define ROW_1 3 // the number of the ROW_ pin 14
#define ROW_2 4 // the number of the ROW_ pin 8
#define ROW_3 5 // the number of the ROW_ pin 12
#define ROW_4 17 // the number of the ROW_ pin 1
#define ROW_5 16 // the number of the ROW_ pin 7
#define ROW_6 15 // the number of the ROW_ pin 2
#define ROW_7 14 // the number of the ROW_ pin 5
//the pin to control COL_
#define LED_A 6 // the number of the COL_ pin 13
#define LED_B 7 // the number of the COL_ pin 3
#define LED_C 8 // the number of the COL_ pin 4
#define LED_D 9 // the number of the COL_ pin 10
#define LED_E 10 // the number of the COL_ pin 6
#define LED_F 11 // the number of the COL_ pin 11
#define LED_G 12 // the number of the COL_ pin 15
#define LED_H 13 // the number of the COL_ pin 16
const char ROW_PIN[8] = {ROW_0 , ROW_1 , ROW_2 , ROW_3 ,ROW_4 ,ROW_5 ,
ROW_6 ,ROW_7 } ;
const char COL_PIN[8] = { LED_A , LED_B , LED_C , LED_D , LED_E ,LED_F ,
LED_G , LED_H } ;
char HeartMap[8][8] = {
    0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,
    0 , 1 , 1 , 0 , 0 , 1 , 1 , 0 ,
    1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 ,
    1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 ,
    1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 ,
    0 , 1 , 1 , 1 , 1 , 1 , 1 , 0 ,
    0 , 0 , 1 , 1 , 1 , 1 , 0 , 0 ,
    0 , 0 , 0 , 1 , 1 , 0 , 0 , 0 ,
};
void setup() {
    int i = 0 ;
    Serial.begin(115200);
    for( i = 0 ; i < 8 ; i++)
    {
        pinMode(ROW_PIN[i] , OUTPUT );
        pinMode(COL_PIN[i] , OUTPUT );
        delay(10);
        digitalWrite(ROW_PIN[i] ,LOW);
        digitalWrite(COL_PIN[i] ,HIGH);
    }
    digitalWrite(ROW_4,LOW);
}

```

```

void setup() {
    // put your setup code here, to run once:
    int i = 0 ;
    Serial.begin(115200);
    for( i = 0 ; i < 8 ; i++)
    {
        pinMode(ROW_PIN[i] , OUTPUT );
        pinMode(COL_PIN[i] , OUTPUT );
        delay(10);
        digitalWrite(ROW_PIN[i] ,LOW);
        digitalWrite(COL_PIN[i] ,HIGH);
    }
    digitalWrite(ROW_4,LOW);
    Serial.println("Start display 12\n");
}

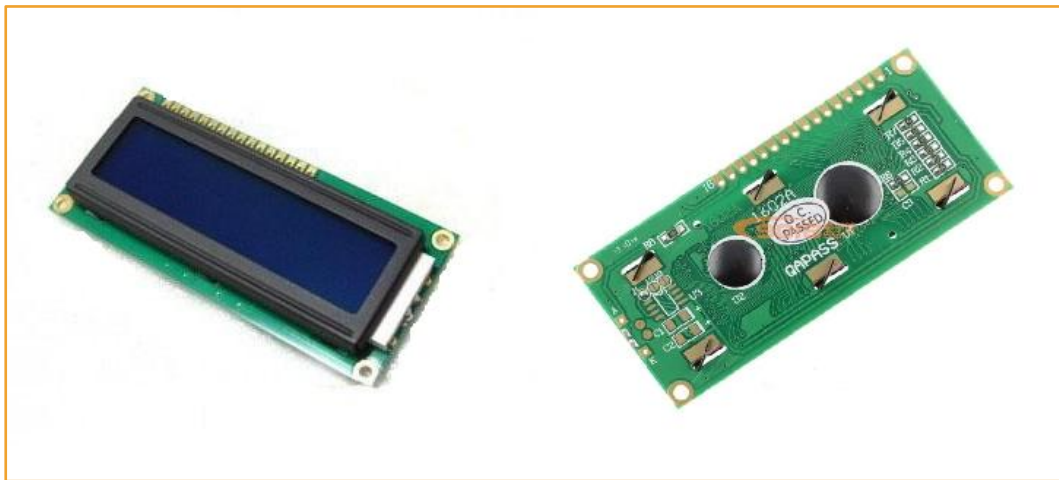
void loop() {
    // put your main code here, to run repeatedly:
    int i j;
    for ( j = 0 ; j < 8 ; j++ )
    {
        for( i = 0 ; i < 8 ; i++ )
        {
            if( HeartMap[i][j] )
                digitalWrite(ROW_PIN[i],HIGH);
            else
                digitalWrite(ROW_PIN[i],LOW);
        }
        digitalWrite( COL_PIN[j],LOW);
        delay(2);
        digitalWrite( COL_PIN[j],HIGH);
    }
}

```

The Display of LED1602

LCD1602 is a kind of character LCD module specializes in displaying letters, numbers and symbols. It is widely used in industry, say, electronic clock, temperature display. Character lcd on the market are mostly based on HD44780 character LCD chip, the control principle is identical. “1602” represents 2 rows and 16 characters of each row.

The physical map of LED1602



The function of each pin:

LCD 1602 has standard 14 pins (without backlight) or 16 pins (with backlight) interface, the specifications of each pin is shown in table 10-13.

Numble	Pin Name	Pin description	Numble	Pin Name	Pin description
1	VSS	GND	9	D2	Data 2
2	VDD	VDD	10	D3	Data 3
3	VL	Liquid Crystal bias	11	D4	Data 4
4	RS	data/command select	12	D5	Data 5
5	R/W	Read/write select	13	D6	Data 6
6	E	Enable signal	14	D7	Data 7
7	D0	Data 0	15	BLA	Backlight VDD
8	D1	Data 1	16	BLK	Backlight VSS

Pin 1: VSS is power supply.

Pin 2: VDD is for +5v.

Pin 3: VL is for LCD contrast adjustment. When VL is connected to the anode of power supply, the contrast is the weakest; when to GND, there will be the highest contrast.

Excess contrast will lead to "shadow", so we require a 10k potentiometer to adjust the contrast when using it.

Pin 4: RS stands for register select. When the pin is high level, we select data registers, otherwise, command registers are selected.

Pin 5: R/W stands for read/write. Read operations are for high level, write operations are for low level. When the RS and R/W are all low level, write operations and address display can be carried out; when the RS is low level and R/W is high, Busy signal is red, otherwise, data can be written in.

Pin 6: E is enable pin. When E jumps to low level from high level, the LCD module takes commands.

Pin 7 ~ 14: D0 ~ D7 is 8-bit bidirectional data line.

Pin 15: The anode of backlight.

Pin 16: The cathode of backlight.

Command 1: Clear Display

Function	Command Code									
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Clear Screen	0	0	0	0	0	0	0	0	0	1

Function: cmd code 01H .clear all screen content and reset display print cursor to address 0x00H (Show back the top left of monitor) set address address counter (AC) a value of 0

Command 2: Reset Cursor

指令功能	Command Code									
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Reset cursor	0	0	0	0	0	0	0	0	1	x

Reset cursor , set address to 0x00H

Command 3: Mode Setting

Function	Command Code									
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Mode Setting	0	0	0	0	0	0	0	1	I/D	S

Function: Set data to 1 bit at a time after the shift direction of the cursor, and set the write one character at a time if mobile.

I/D : 0= write one bit data and left shift of the cursor 1= write one bit data and right shift of the cursor

S : 0 = writing new data after the screen does not move 1 = writing new data display overall moves to the right after 1 characters

Command 4: Display ON/OFF Control

Function	Command Code									
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Display Control	0	0	0	0	0	0	1	D	C	B

Function: Display on/off control, The cursor flashing/closed, The cursor show or not

Control Bit	Setting	
D	0=disable display	1= enable display
C	0=cursor not show	1= show cursor
B	0=cursor flash	1= cursor not flash

Command 5: Set the display instruction or cursor

movement direction

Function	Command Code									
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Direction control	0	0	0	0	0	0	S/C	R/L	C	B

Function: Cursor or display screen shift control.

Control bit	setting	
S/C	R/L	
0	0	The cursor left shift 1, and AC value decrease 1
0	1	The cursor right shift 1, and AC value increase 1
1	0	All characters on a shift to the left one, but the cursor does not move
1	1	All characters on a shift to the right one, but the cursor does not move

Command 6: Function setting

Function	Command Code									
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Function setting	0	0	0	0	1	DL	N	F	X	X

Function : Data bus digits, show the number of rows and fonts

Control bit Setting

DL 0= Data bus for 4 1= Data bus for 8

N 0= show one line 1= show two line

F 0= 5 x 7 lattice/per character 1= 5 x 10 lattice/per character

Command 7: CGRAM Address Setting

Function	Command Code									
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
CGRAM Address	0	0	0	0	CGRAM Address (6bit)					

Function: Set the next to deposit the CGRAM address of data.

DB5DB4DB3 char code, the address of the characters。 (000~111)

DB2DB1DB0 line numble。 (000~111)

Command 8: Set DDRAM address

Function	Command Code									
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
DDRAM 地址	0	0	0	DDRAM address (7bit)						

Function: Set the next to deposit data of DDRAM addresses.

Command 9:Read the busy signal or AC address instruction

Function	Command Code									
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
AC/busing	0	0	FB	AC 内容 (7bit)						

Function : Read busy signal BF bit ,BF=1 mease led1206 device is busy can not [receive](#) data or command from MCU.

BF=0 led1206 device is not busy can [receive](#) data or command from MCU.

2. read AC register.

Command 10: write data to DDRAM or CGRAM

Function	Command Code									
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
DD/CGRAM	0	0	Write data D0 - D7							

Funtion :

- <1> write char data to DDRAM,
- <2> write graph code to CGRAM. DB7DB6DB5 is ignore, default value “000”. DB4DB3DB2DB1DB0 Font data that corresponds to each row 5

Command 11: read DDRAM or CGRAM Value

Function	Command Code									
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Read DD/CGRAM	0	0	Read value D0 - D7							

Function : read value from DDRAM or CGRAM.

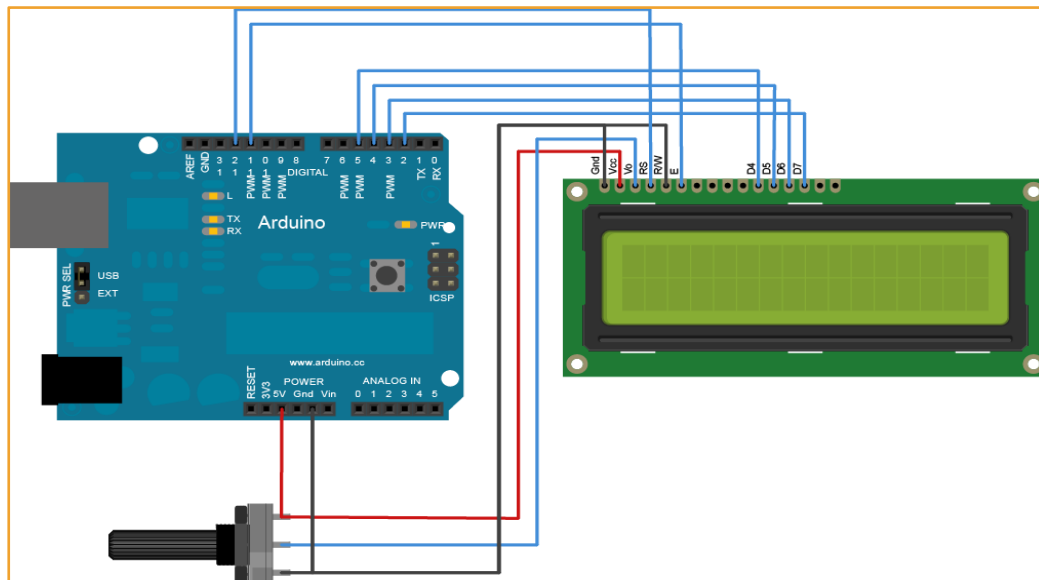
Timing:

- Read status input: RS=L, RW=H, E=H output: DB0~DB7
- Read cmd input: RS=L, RW=L, E= Falling edge pulse, DB0~DB7
- Read Data input: RS=H, RW=H, E=H output: DB0~DB7
- Write Data input: RS=H, RW=L, E= Falling edge pulse , DB0~DB7

Experiment Purpose

The first row of LED1602 displays “hello Arduino !”,the second displays “keyway !”

The Schematic Diagram



Experiment Principle

- First, LED1602 takes commands above

- Writing `lcd1602_write_cmd (0x38)` by command 6 initializes 8-bit interface, 2 rows and 5x7 character size.
- Writing `lcd1602_write_cmd x(0x06)` by command 3; Setting the input as auto-increment, without the display of shifting.
- Writing `lcd1602_write_cmd x(0x0E)` by command 4; displaying the screen turned on, the cursor display, flicker-free.
- Writing `lcd1602_write_cmd (0x01)` by command 1; empty the screen, the cursor position return to zero.

```
#define DB0 2 // the number of the ROW_ pin 9
#define DB1 3 // the number of the ROW_ pin 14
#define DB2 4 // the number of the ROW_ pin 8
#define DB3 5 // the number of the ROW_ pin 12
#define DB4 6 // the number of the COL_ pin 13
#define DB5 7 // the number of the COL_ pin 3
#define DB6 8 // the number of the COL_ pin 4
#define DB7 9 // the number of the COL_ pin 10
//the pin to control COL_
#define LCD1602_RS 10 // the number of the COL_ pin 6
#define LCD1602_RW 11 // the number of the COL_ pin 11
#define LCD1602_E 12 // the number of the COL_ pin 15
const char LCD1602_DB[8] = { DB0 , DB1 , DB2 , DB3 , DB4 , DB5 , DB6 , DB7 } ;
void lcd1602_write_cmd(unsigned char cmd)
{
    int i ;
    for ( i = 0 ; i < 8 ; i++ )
    {
        digitalWrite( LCD1602_DB[i] ,cmd & ( 1 << i )); //cmd hung on data
pin
    }
    digitalWrite(LCD1602_RS , LOW) ; // cmd mode
    digitalWrite(LCD1602_RW , LOW ) ; // write data
    digitalWrite(LCD1602_E, HIGH) ; // enable
    delay(10);
    digitalWrite(LCD1602_E,LOW);
    delay(10) ;
}

void lcd1602_write_data(unsigned char dat)
{
    int i ;
    for ( i = 0 ; i < 8 ; i++ )
    {
        digitalWrite( LCD1602_DB[i] ,dat & ( 1 << i )); //cmd hung
on data pin
    }
    digitalWrite(LCD1602_RS , HIGH) ; // data mode
    digitalWrite(LCD1602_RW , LOW ) ; // write data
    digitalWrite(LCD1602_E, HIGH) ; // enable
}
```

Control

Class 16

Direct Current Motor

Not available for now

The Stepping Motor

The principle of stepping motor

The stepping motor is a kind of mechanical and electrical device that converts electrical pulses into angular displacement. More generally, when the stepping driver receives a pulse signal, it will drive the stepping motor rotate a certain degree according to the setting direction (namely, the stepping angle). You can control the amount of angular displacement by controlling the amount of pulse, so as to achieve the purpose of accurate positioning, and at the same time you can also control motor rotation speed and acceleration by controlling the pulse frequency, so as to achieve the purpose of speed controlling.

The classification of stepping motor

Permanent Magnet (PM) : It is generally with two phase, torque and volume are low, the stepping angle is 7.5 degrees or 15 degrees.

Reaction (VR) : It is generally with three phase, which can realize large torque output, the stepping angle is 1.5 degrees commonly, but the noise and vibration are quite enormous.

Hybrid (HB) : It combines the advantages of permanent magnet and reaction and is divided into two and five phase: two phase stepping angle is 1.8 degrees and five phase stepping angle is 0.72 degrees in general. This kind of stepping motor is more widely used.

Technical parameters

The static indicators of stepping motor

The step angle:

It means that every time control system sends a step pulse signal, the fixed angle of a certain permanent magnet stepping motor is $3.75^\circ/7.5^\circ$ (the value of half step driving is 3.75° , the whole step driving is 7.5°). The step angle can be called "fixed angle of a stepping motor", it is not necessarily the actual angle of a working motor, the actual angle relates to the driver.

The phase number:

It refers to the number of coil group inside a motor. At present, the commonly used stepping motors are two-phase, three-phase, four-phase and five-phase motors.

The step angle of a motor varies along with phase number. Generally, the step angle of two-phase motor is $0.9^\circ/1.8^\circ$, three-phase is $0.75^\circ/1.5^\circ$ and five-phase is $0.36^\circ/0.72^\circ$. When there is no subdividing driver, users mainly meet their requirements of the step angle by selecting different phase number motors. If using subdividing driver, the "number" of phase number will become meaningless, users only need to change subdividing number of drives, then the step angle can be shifted.

The number of beats:

It refers to the pulse number or conducting state needed to complete the periodic changes in a magnetic field, and it can also be defined as the pulse number that a motor turns a certain step angle. Let's take four-phase motor for an example, four phase and four beats operation mode, namely, AB - BC - CD - DA - AB, four phase and eight beats operation mode is A - AB - B - BC - C - CD - D - DA - A.

The dynamic indicators of stepping motor**The precision of stepping angle**

It refers to the error between the actual value and the theoretical value when a motor turns a certain step angle. Expressing as a percentage: the angle error / step angle * 100%. This value varies with the number of beats, when the motor runs by 4 beats, it should be within 5%, 8 beats within 15%

Out of step

When the drive pulse number for the motor is not equal to the steps during operation, we will call this out of step.

The misalignment angle

It refers to the deflective angle between rotor axis and stator axis. There definitely exists misalignment angle when a motor is running. The error caused by misalignment angle can't be solved even we use subdividing drives.

The driving modes of stepping motor

The driving mode of stepping motor is also known as exciting mode, divided into full step excitation and half step excitation. The former one also can be divided, one-phase(one-beat drive) and two-phase excitation(full step drive); the

latter also refers to one-two phase excitation(one step drive). Suppose every rotation needs 200 pulse signal to excite, then each excitation signal can make the stepping motor rotate 1.8° .

The comparison of the three drive mode:

Drive mode	Step Angle	Power Consumption	Advantages and disadvantages
Single step	5.625	P	Simple control, low consumption, but the minimum output torque, vibration is larger, when step easy alienation
Full step	5.625	2P	Maximum power consumption, large output torque, small vibration, step is stable
Half step	2.8125	1.5P	Performance between step taken in single drive and the drive, only half the stepping Angle, smooth operation, the most widely used

The timing tables of the three drive mode:

Wire/step	Step1	Step2	Step3	Step4	Step5	Step6	Step7	Step8
Blue/A	1	1	0	0	0	0	0	1
Pink/B	0	1	1	1	0	1	0	0
Yellow/C	0	0	0	1	1	1	0	0
Orange/D	0	0	0	0	0	1	1	1

(c). Half step

Wire/step	Step1	Step2	Step3	Step4
Blue/A	1	0	0	0
Pink/B	0	1	0	0
yellow/C	0	0	1	0
orange/D	0	0	0	1

Single step

The reduction stepping Motor

Diameter: 28mm

Voltage: 5 v

The step angle: $5.625 \times 1/64$

The reduction ratio: 1/64

The no-load power consumption of the stepping motor is under 50mA, it is equipped with a 64 times speed reducer, namely, 64 drive pulse. If the external belt ro

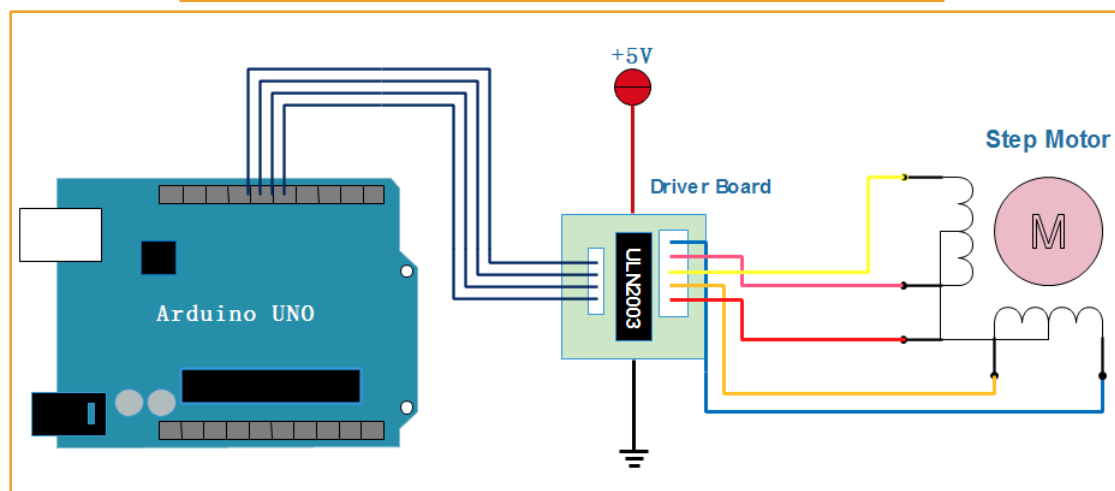
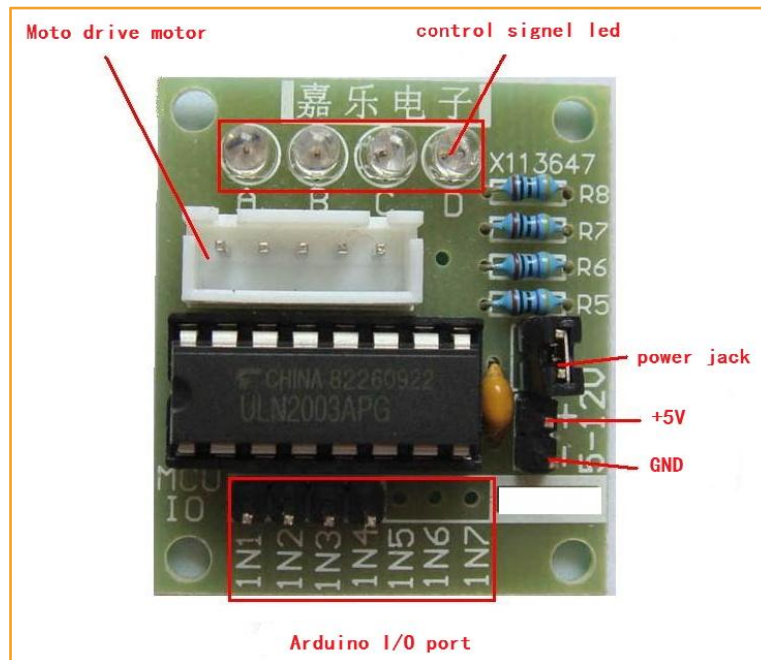
Wire/step	Step1	Step2	Step3	Step4
Blue/A	1	0	0	1
Pink/B	1	1	0	0
Yellow/C	0	1	1	0
orange/D	0	0	1	1

Full step

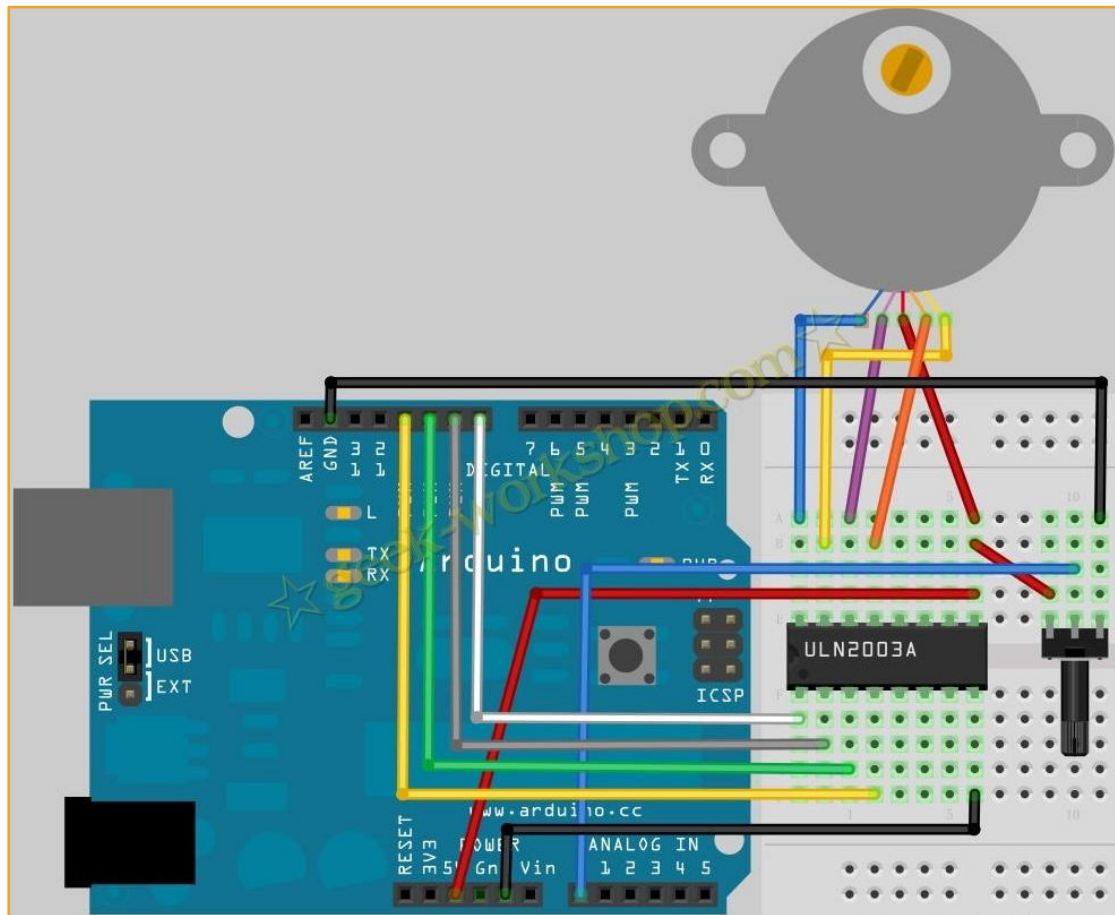
tates a circle, the stepping motor spindle needs to rotate 64 circles due to the 1:64 reduction gear in the motor.

The output torque is massive, so heavy load could be driven, it is suitably used on a development board. Notice: This stepping motor is equipped with a 64 times speed reducer, the rotational speed is slower compared with those without a speed reducer. For the convenience of observation, we can stick a piece of cardboard on the output shaft.

The stepping motor driver board



Hardware connection:



The principle of the program:

We choose half step drive, so the stepping motor is driven according to the above diagram (c). Half step means the step angle shortens to half of the original: $5.625/2 = 2.8125$. To make the stepping motor equipped with speed reducer rotate a whole circle, it takes $(360/5.625/2) * 64 = 8256$ pulses.

6

The program is as follows:

```
#define STEPPER_MOTOR_A 8
#define STEPPER_MOTOR_B 9
#define STEPPER_MOTOR_C 10
#define STEPPER_MOTOR_D 11

#define STEPPER_MODE_4 4
#define STEPPER_MODE_8 8
boolean StepperDir = 0;
int stepperSpeed = 3 ,CurrentStep = 0 ;
//set Stepper motor speed delays
int stepsum = 0 ;
void setup()
{
    pinMode(STEPPER_MOTOR_A, OUTPUT);
    pinMode(STEPPER_MOTOR_B, OUTPUT);
    pinMode(STEPPER_MOTOR_C, OUTPUT);
    pinMode(STEPPER_MOTOR_D, OUTPUT);
    Serial.begin(115200);
    Serial.println("Stepper Motor And Start :");
}
void loop()
{
    switch( CurrentStep )
    {
        case 0:
            digitalWrite(STEPPER_MOTOR_A, LOW);
            digitalWrite(STEPPER_MOTOR_B, HIGH);
            digitalWrite(STEPPER_MOTOR_C, HIGH);
            digitalWrite(STEPPER_MOTOR_D, HIGH);
            break;
        case 1:
            digitalWrite(STEPPER_MOTOR_A, LOW);
            digitalWrite(STEPPER_MOTOR_B, LOW);
            digitalWrite(STEPPER_MOTOR_C, HIGH);
            digitalWrite(STEPPER_MOTOR_D, HIGH);
            break;
        case 2:
            digitalWrite(STEPPER_MOTOR_A, HIGH);
            digitalWrite(STEPPER_MOTOR_B, LOW);
            digitalWrite(STEPPER_MOTOR_C, HIGH);
            digitalWrite(STEPPER_MOTOR_D, HIGH);
            break;
```

```

    case 3:
        digitalWrite(STEPPER_MOTOR_A, HIGH);
        digitalWrite(STEPPER_MOTOR_B, LOW);
        digitalWrite(STEPPER_MOTOR_C, LOW);
        digitalWrite(STEPPER_MOTOR_D, HIGH);
    break;
    case 4:
        digitalWrite(STEPPER_MOTOR_A, HIGH);
        digitalWrite(STEPPER_MOTOR_B, HIGH);
        digitalWrite(STEPPER_MOTOR_C, LOW);
        digitalWrite(STEPPER_MOTOR_D, HIGH);
    break;
    case 5:
        digitalWrite(STEPPER_MOTOR_A, HIGH);
        digitalWrite(STEPPER_MOTOR_B, HIGH);
        digitalWrite(STEPPER_MOTOR_C, LOW);
        digitalWrite(STEPPER_MOTOR_D, LOW);
    break;
    case 6:
        digitalWrite(STEPPER_MOTOR_A, HIGH);
        digitalWrite(STEPPER_MOTOR_B, HIGH);
        digitalWrite(STEPPER_MOTOR_C, HIGH);
        digitalWrite(STEPPER_MOTOR_D, LOW);
    break;
    case 7:
        digitalWrite(STEPPER_MOTOR_A, LOW);
        digitalWrite(STEPPER_MOTOR_B, HIGH);
        digitalWrite(STEPPER_MOTOR_C, HIGH);
        digitalWrite(STEPPER_MOTOR_D, LOW);
    default:
        digitalWrite(STEPPER_MOTOR_A, LOW);
        digitalWrite(STEPPER_MOTOR_B, LOW);
        digitalWrite(STEPPER_MOTOR_C, LOW);
        digitalWrite(STEPPER_MOTOR_D, LOW);
    break;
}
if( StepperDir )
{
    CurrentStep++;
}else{
    CurrentStep--;
}
}

```

```

    if( CurrentStep < 0 ){
        CurrentStep = STEPPER_MODE_8-1 ;
    }
    delay(stepperSpeed);
    stepsum++;
    if( stepsum == 4096 )
    {
        Serial.println(stepsum);
        stepsum = 0 ;
        delay(5000);
    }
}

```

Class 18

Servo Motor

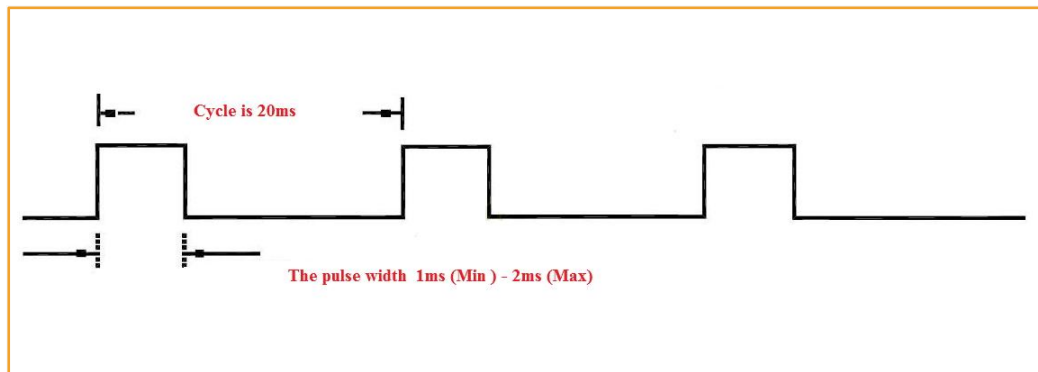
I guess you must have seen robots or high-tech products in American science-fiction films, at least heard of the noise of some moving automatic mechanical arms and audiences' scream. The noise comes from the rotation of the steering gear.

The steering gear is a kind of position (angle) servo driver, it can be rotated to any angle between 0 and 180 degrees, then precisely stop at your command, so it is suitable for those control systems which require angle changing and keeping .

At present, it has been widely used in high-grade remote control toys, such as model aircraft, including the model plane, submarine model and remote control robot. Steering gear is an unprofessional name, in fact it is a kind of servo motor, a set of automatic control device which consists of DC motor, reduction gear group, sensor and control circuit. What is the automatic control? The so-called automatic control — continuously adjusting the output deviation by using a closed-loop feedback control circuit — makes the system output constant.

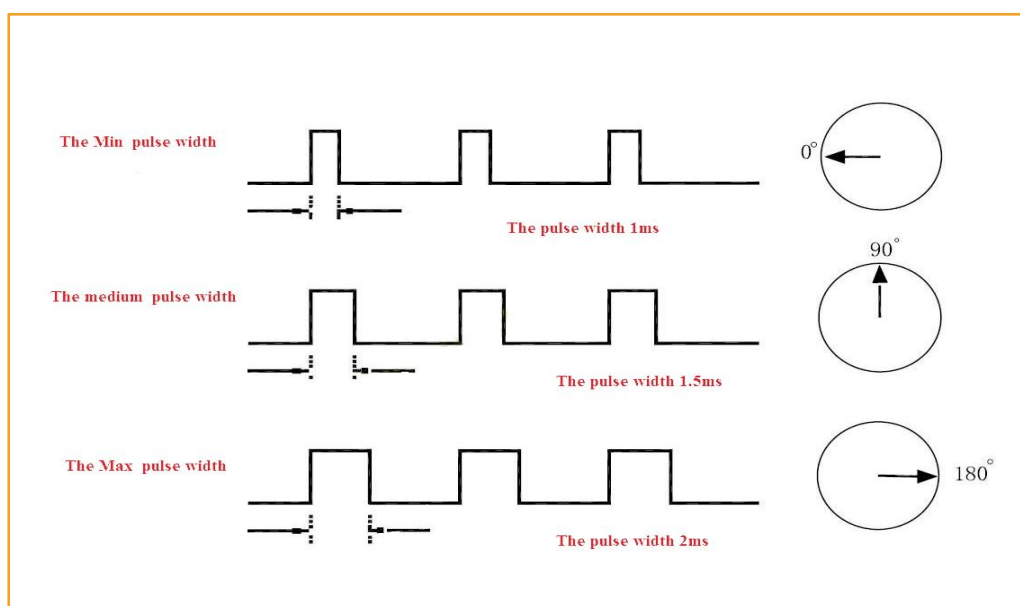
The steering gear servo system can be controlled by variable bandwidth pulse, the control line is used to transmit pulse. The parameters of the pulse consist minimum value, maximum value and frequency. In general, the cycle of the reference signal of the steering gear is 20ms, the bandwidth is 1.5ms. The reference signal is from the middle position. The steering gear has the maximum rotation angle, the middle position refers to the volumes from this position to the minimum angle and the maximum angle are exactly identical. The most important part, the maxi

imum rotation angle varies with different steering gears, but of which the bandwidth of the middle position is certain, that is 1.5 ms.

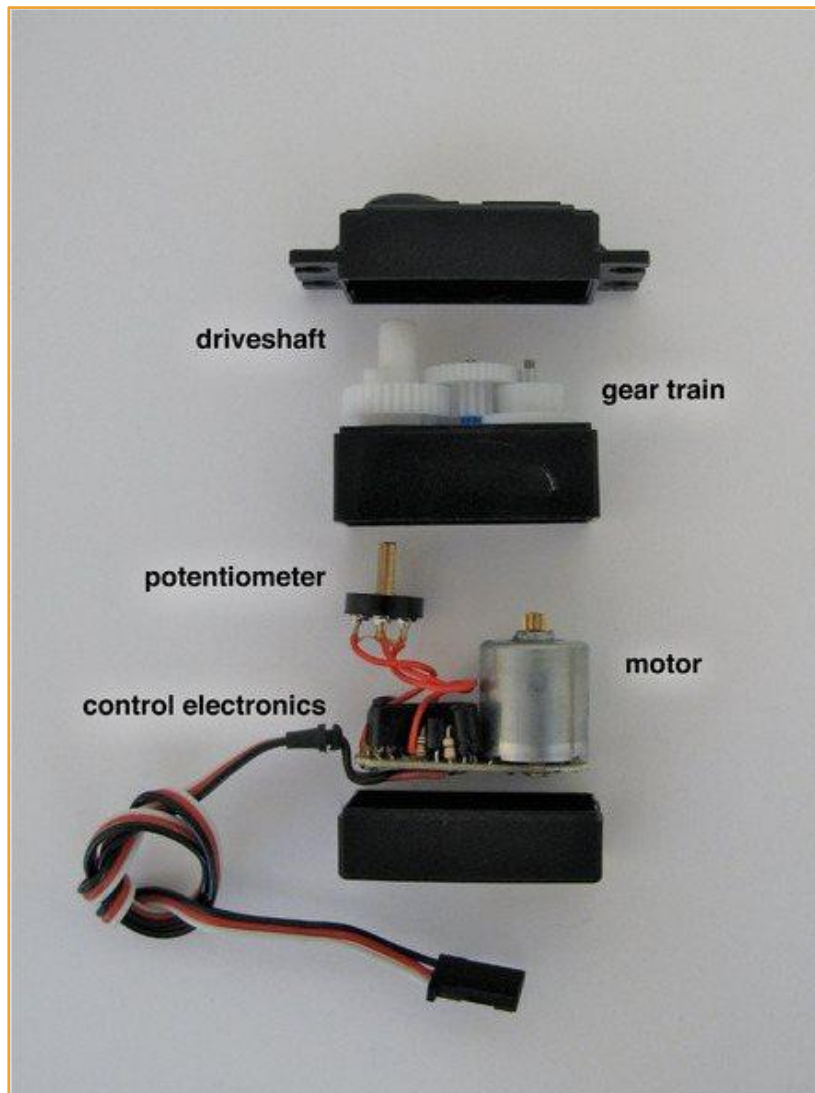


The rotation angle is produced by the continuous pulse from control line. This method is called pulse modulation. The length of pulse decides the rotation angle of steering gear. For example: the steering gear rotates to the middle position by 1.5 millisecond pulse (for 180° steering gear, the middle position is 90°). When the control system issues commands to move the steering gear to a particular position and make it keep a certain angle, then the influence of the external force won't change the angle, but the ceiling is its biggest torsion. Unless the control system continuously issues pulse to stable the steering angle, the angle will not always stay the same.

When the steering gear receives a pulse less than 1.5ms, the output shaft will take the middle position as standard and rotate a certain angle counterclockwise; when the received pulse is greater than 1.5ms, then the output shaft rotates clockwise. Different brands of steering gears, and even the same brand of different steering gears, the maximum and minimum value could be different.



The internal structure of steering gear



Experiment Purpose

Controlling the motor rotation through the potentiometer

Component List:

- ◆ 10k potentiometer*1
- ◆ Steering gear*1
- ◆ Breadboard*1
- ◆ A number of breadboard jumper

Connecting the left pin of the potentiometer to 3.3V, the right to GND and the middle to analog interface 0.

Connecting 5V and GND to the steering gear, the signal port to number 7 interface.

The Program Code:

```
int readPin = 0;
int servopin = 7;

void servopulse(int angle)
{
    int pulsewidth=(angle*11)+500;
    digitalWrite(servopin,HIGH);
    delayMicroseconds(pulsewidth);
    digitalWrite(servopin,LOW);
    delayMicroseconds(20000-pulsewidth);
}

void setup()
{
    pinMode(servopin,OUTPUT);
}

void loop()
{
    int readValue = analogRead(readPin);
    int angle = readValue / 4;

    for(int i=0;i<50;i++)
    {
        servopulse(angle);
    }
}
```