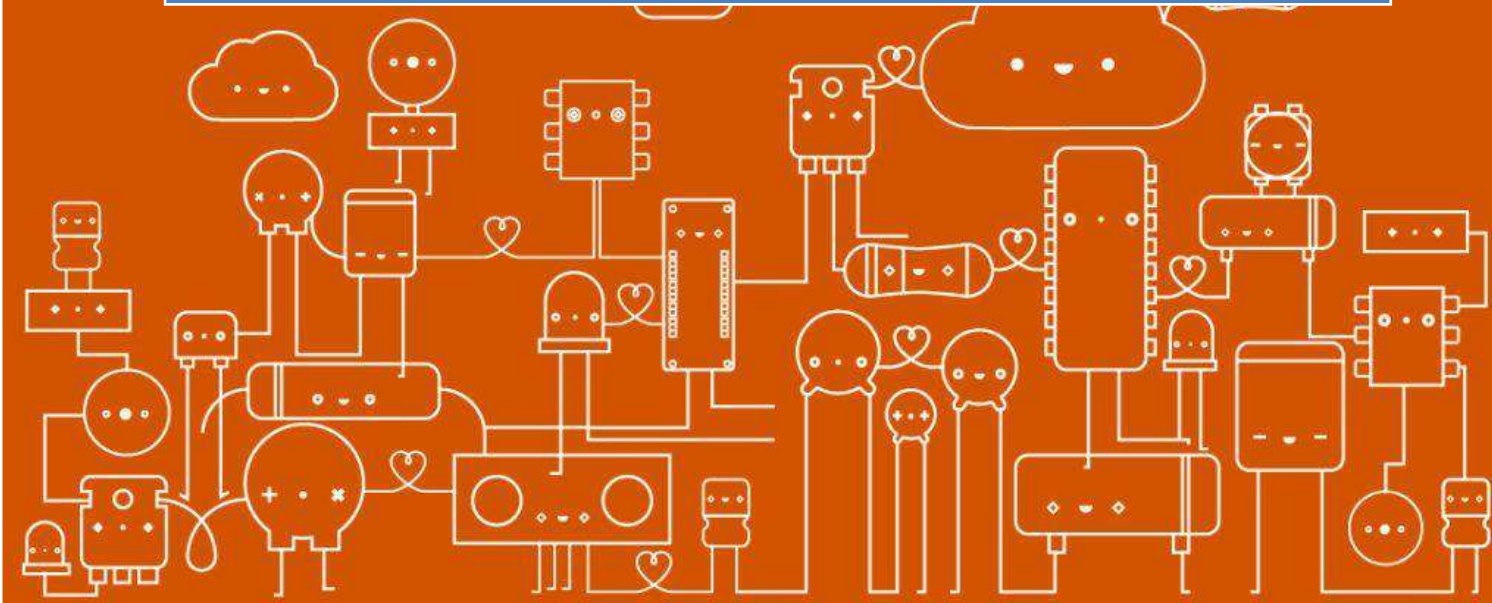


# 2019

## Arduino Starter Kit

### Arduino IDE Programming Tutorials



# C programming

---

1. Hello world	2
2. LED Twinkle	7
3. Analog value	10
4. Advertising lights	14
5. Traffic lights	18
6. Key control	22
7. Answering machine	26
8. Buzzer	30
8.1 Active buzzer	30
8.2 Passive buzzer	34
9. PWM dimming	40
10. Light controlled sound	44
11. Sensible heat light	48
12. 8x8 lattice dot matrix	52
13. Tilt switch	57
14. Flame alarm	61
15. Nixie tube	65
16. Four bit nixie tube	72
17. 74HC595	78
18. Servo control	83
19. IR control	88
20. 1602 display	97

# 1-Hello World!

## The purpose of the experiment:

This course is an experiment that allows Arduino and PC to communicate. The experimental results is to let Arduino say "Hello World!"

## List of components required for the experiment:

Arduino UNO board \*1

USB cable \*1

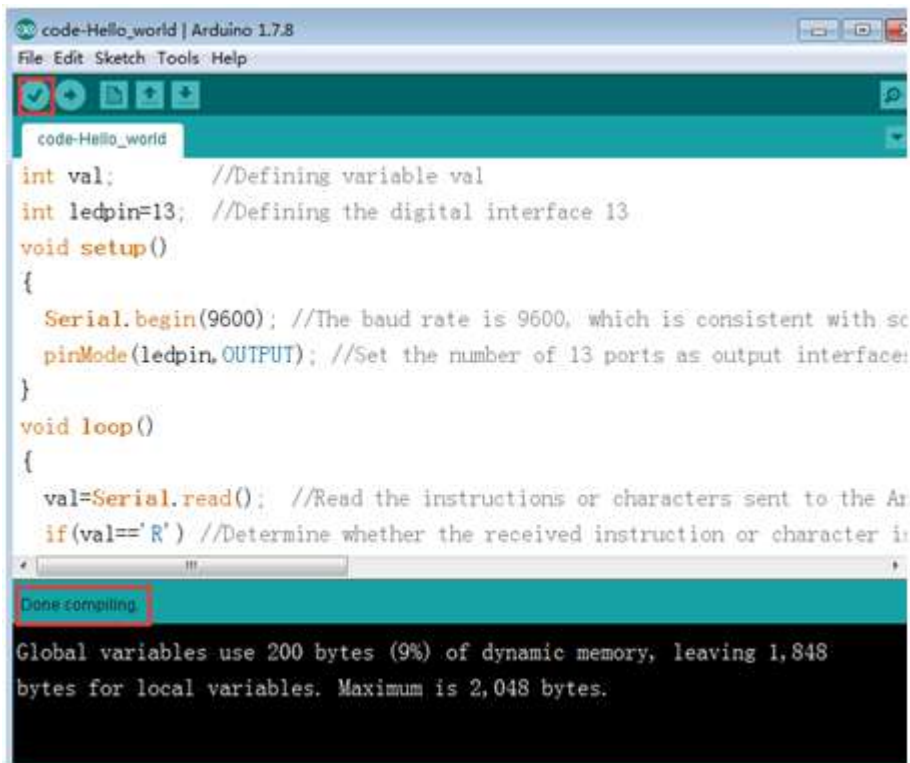


## Experimental code analysis:

```
int val;      //Defining variable val
int ledpin=13; //Defining the digital port 13
void setup()
{
  Serial.begin(9600); //The baud rate is 9600, which is consistent with software settings.
  When accessing specific devices (such as Bluetooth), we also have to agree with the
  baud rate of other devices.
  pinMode(ledpin,OUTPUT); //Set the number of 13 ports as output interfaces, and the
  I/O ports we use on Arduino have similar definitions.
}
void loop()
{
  val=Serial.read(); //Read the instructions or characters sent to the Arduino by the PC
  machine, and assign the instruction or character to val
  if(val=='R') //Determine whether the received instruction or character is "R"
  { //If the "R" character is received
    digitalWrite(ledpin,HIGH); //Light the number of 13 ports of LED
    delay(500);
    digitalWrite(ledpin,LOW); //Extinguish the number of 13 ports of LED
    delay(500);
    Serial.println("Hello World!"); //Display "Hello World! "
  }
}
```

## Experimental steps:

1. We need to open the code of this experiment: **code-Hello\_world.ino**, click "✓" under the menu bar to compile the code, and wait for the word "**Done compiling**" in the lower right corner, as shown in the figure below.



```

code-Hello_world | Arduino 1.7.8
File Edit Sketch Tools Help

code-Hello_world

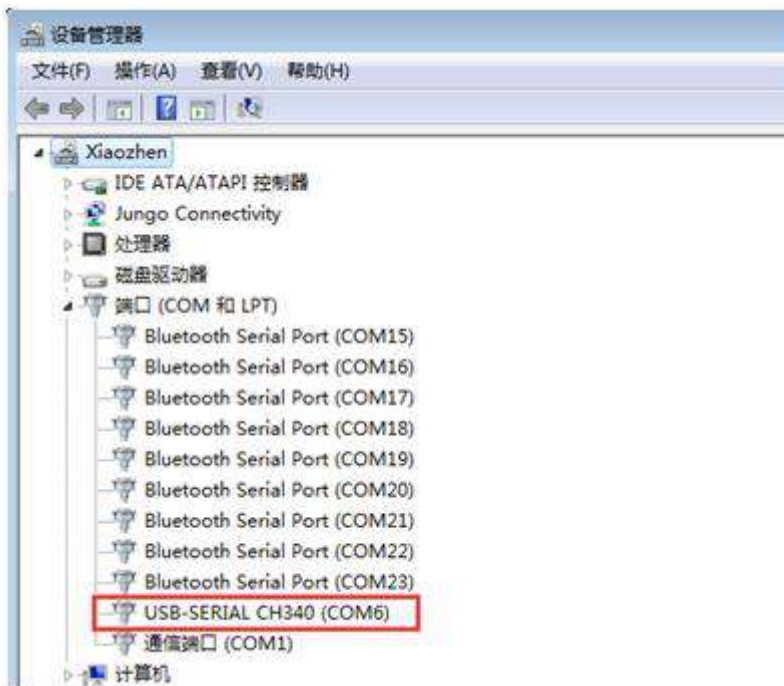
int val; //Defining variable val
int ledpin=13; //Defining the digital interface 13
void setup()
{
  Serial.begin(9600); //The baud rate is 9600, which is consistent with se
  pinMode(ledpin, OUTPUT); //Set the number of 13 ports as output interface
}
void loop()
{
  val=Serial.read(); //Read the instructions or characters sent to the Ar
  if(val=='R') //Determine whether the received instruction or character is

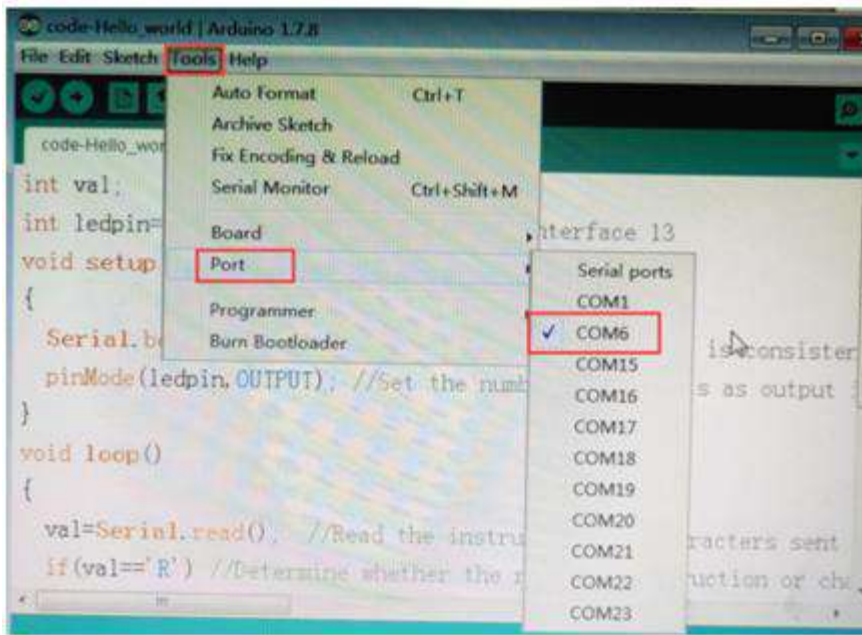
Done compiling.

Global variables use 200 bytes (9%) of dynamic memory, leaving 1,848
bytes for local variables. Maximum is 2,048 bytes.

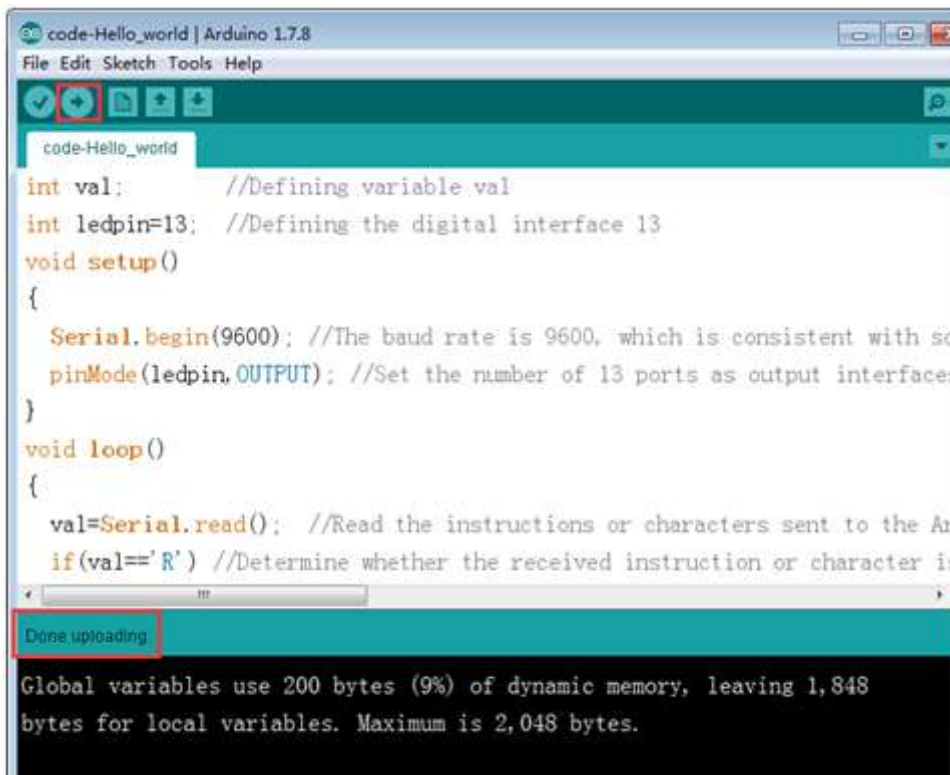
```

2. In the menu bar of Arduino IDE, we need to select **【Tools】** --- **【Port】** --- selecting the port that the serial number displayed by the device manager just now, as shown in the figure below.

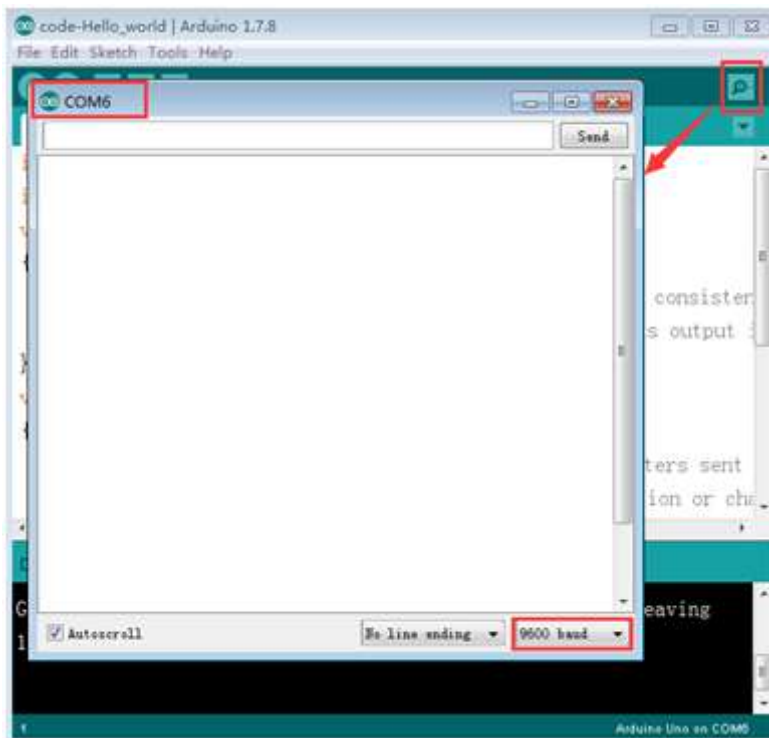




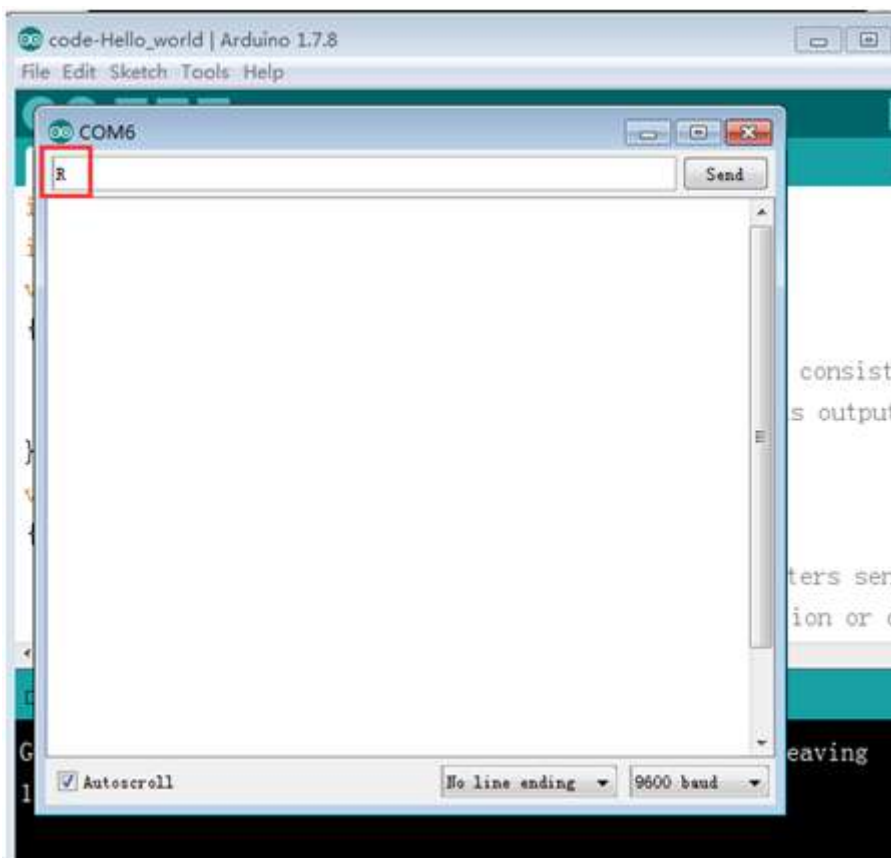
3. After the selection is completed, you need to click “→” under the menu bar to upload the code to the Arduino UNO board. When the word “**Done uploading**” appears in the lower left corner, the code has been successfully uploaded to the Arduino UNO board, as shown in the figure below.



4. We need to turn on the serial port monitor in the upper right corner of the Arduino IDE, and a serial port printing box of the Arduino port will appear. The baud rate is set to 9600 in the lower right corner, as shown in the figure below.

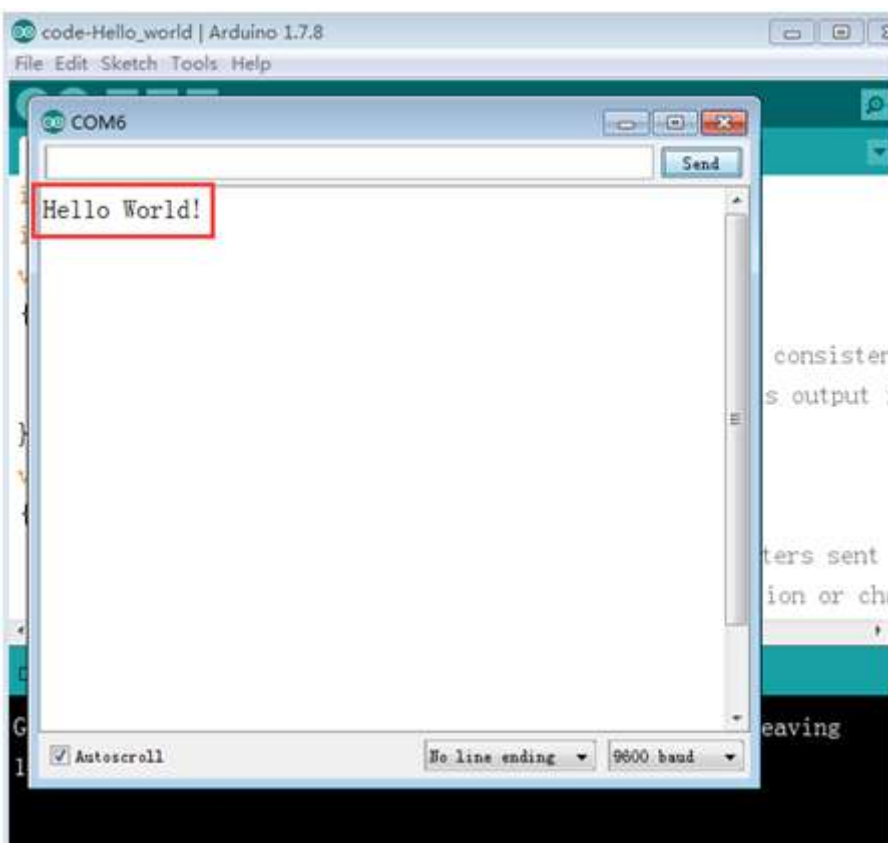
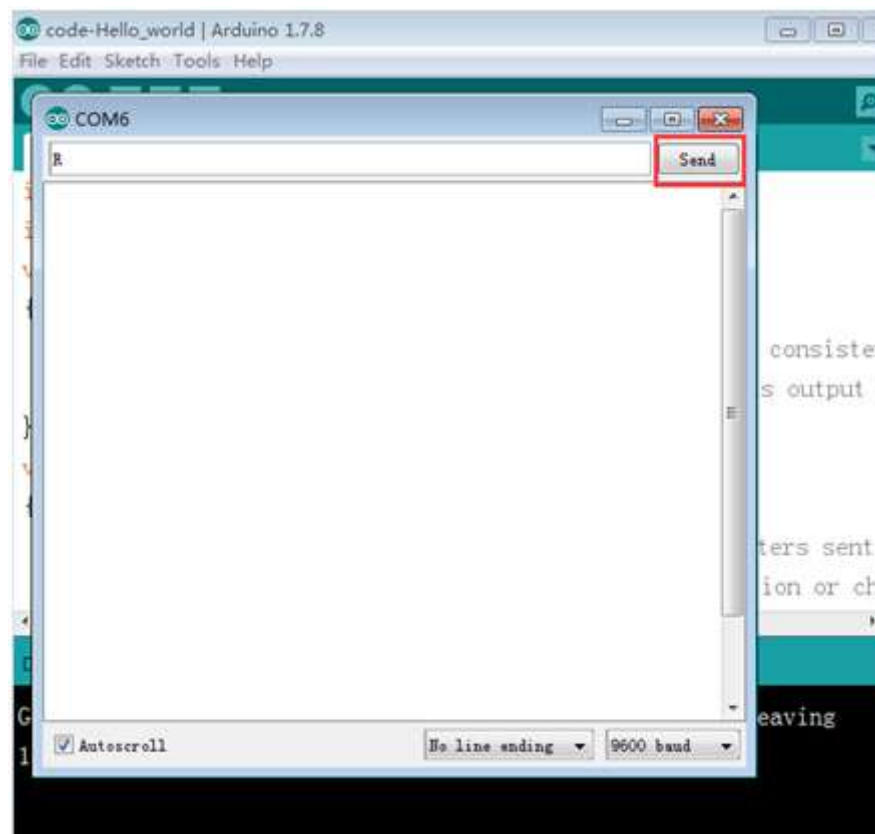


5. According to the code, we should enter "R" in the sending box, as shown in the figure below.



6. After clicking “send” on the upper right corner, we can receive the Hello World! in the receiving box below, as shown in the figure below.





## 2-Led Twinkle

### The purpose of the experiment:

This course is to use the I/O port on the Arduino UNO board and an external LED light to complete the experiment. The experiment is to make the LED light to twinkle, lights up for 1 second and turns off for 1 second.

### List of components required for the experiment:

Arduino UNO board \*1

USB cable \*1

LED\*1 (Color random)

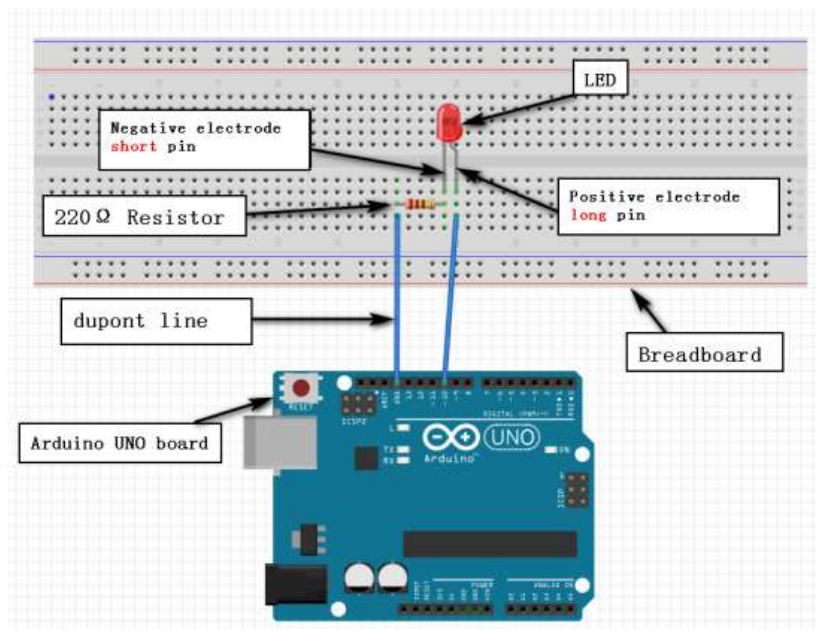
220Ω Resistor \*1

Breadboard \*1

Dupont line \*1 bunch

Actual object connection diagram:

We need to connect the circuit as shown in the figure below.



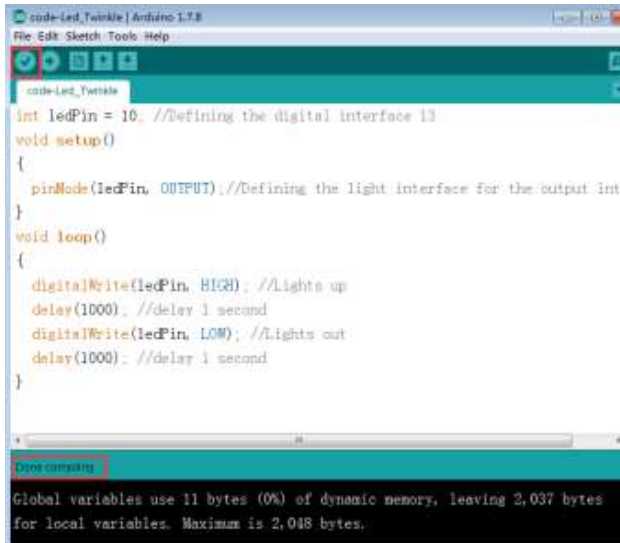
### Experimental code analysis:

```
int ledPin = 10; //Defining the digital port 10
void setup()
{
  pinMode(ledPin, OUTPUT); //Defining the light port for the output port
}
void loop()
{
  digitalWrite(ledPin, HIGH); //Lights up
  delay(1000); //delay 1 second
  digitalWrite(ledPin, LOW); //Lights out
  delay(1000); //delay 1 second
}
```

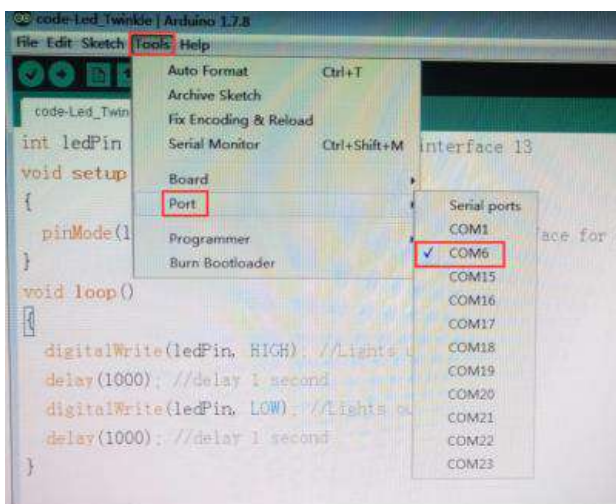
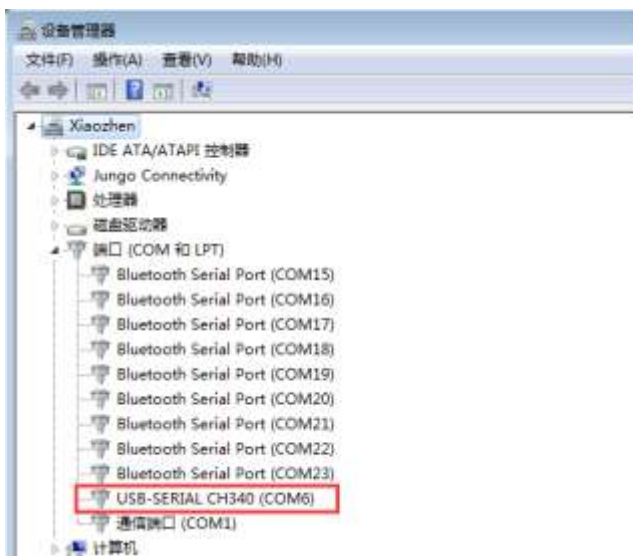
Experimental steps:



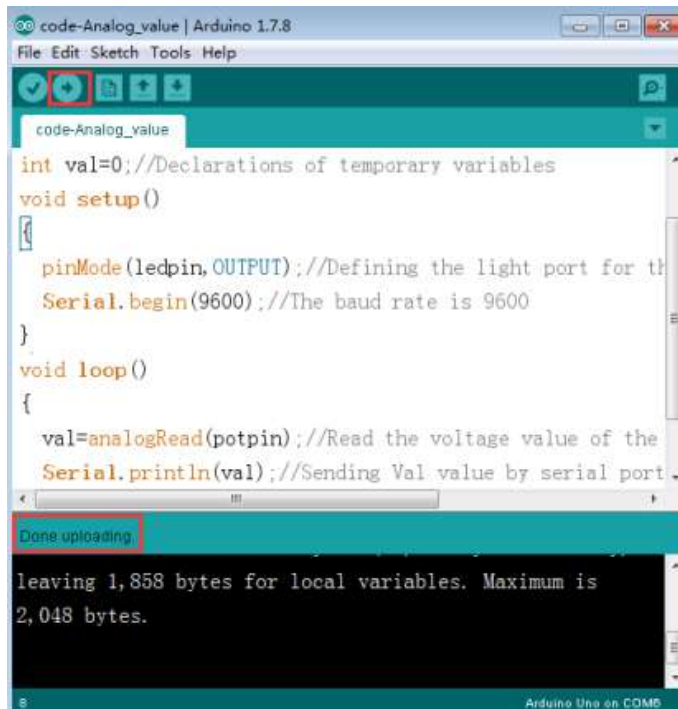
1. We need to open the code of this experiment: code-Led\_Twinkle.ino, click "✓" under the menu bar to compile the code, and wait for the word "Done compiling" in the lower right corner, as shown in the figure below.



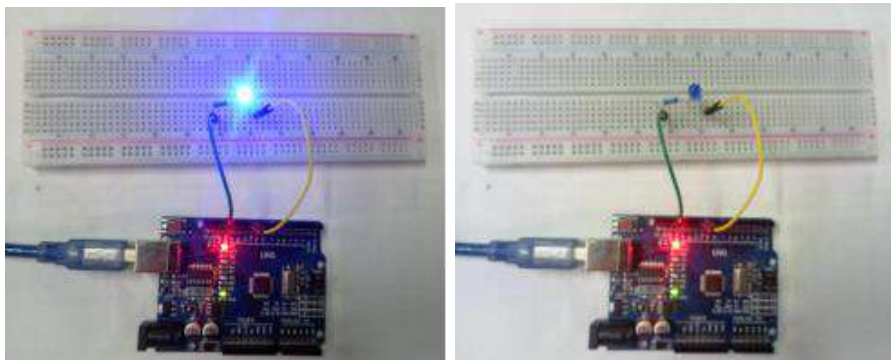
2. In the menu bar of Arduino IDE, we need to select **Tools** --- **Port** --- selecting the port that the serial number displayed by the device manager just now, as shown in the figure below.



3. After the selection is completed, you need to click “→” under the menu bar to upload the code to the Arduino UNO board. When the word “Done uploading” appears in the lower left corner, the code has been successfully uploaded to the Arduino UNO board, as shown in the figure below.



4. After the code is uploaded, we can see LED light twinkle every second, as shown in the picture below.



### 3-Analog value

#### The purpose of the experiment:

In this course, we learn how to use of simulate I/O interface by combining adjustable resistor. Arduino is equipped with 6 analog interfaces, port 0~part 5. They can be reused. In addition to the analog interface function, these 6 interfaces can be used as digital interfaces, No 14-19.

#### List of components required for the experiment:

Arduino UNO board \*1

USB cable \*1

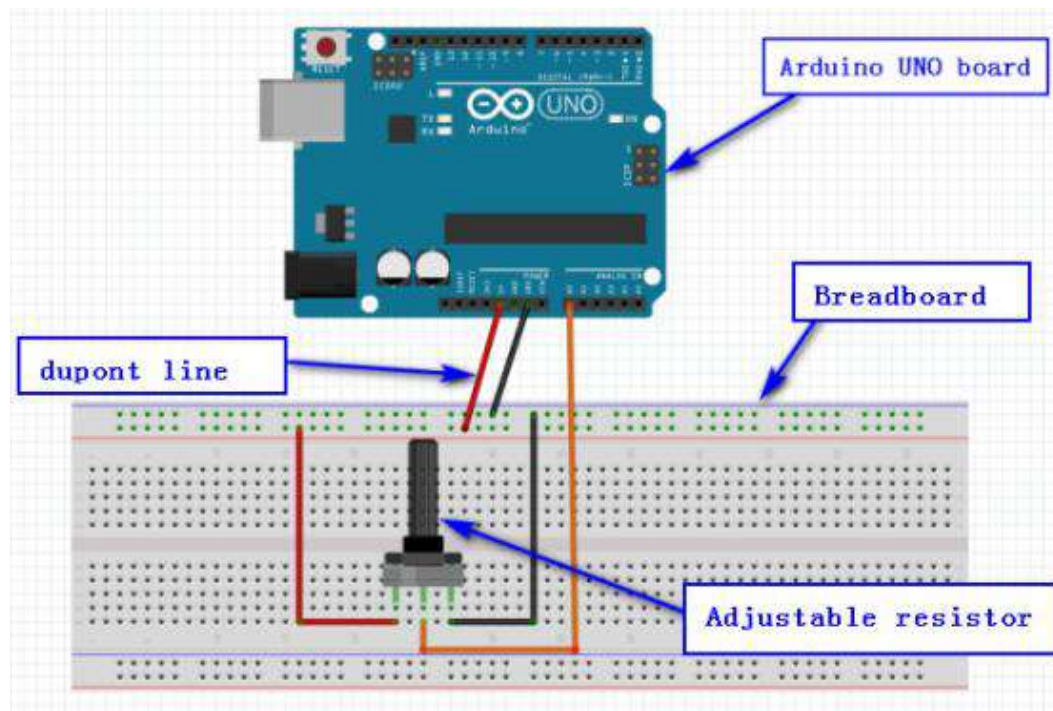
Adjustable resistor \*1

Breadboard \*1

Dupont line \*1bunch

Actual object connection diagram:

We need to connect the circuit as shown in the figure below.



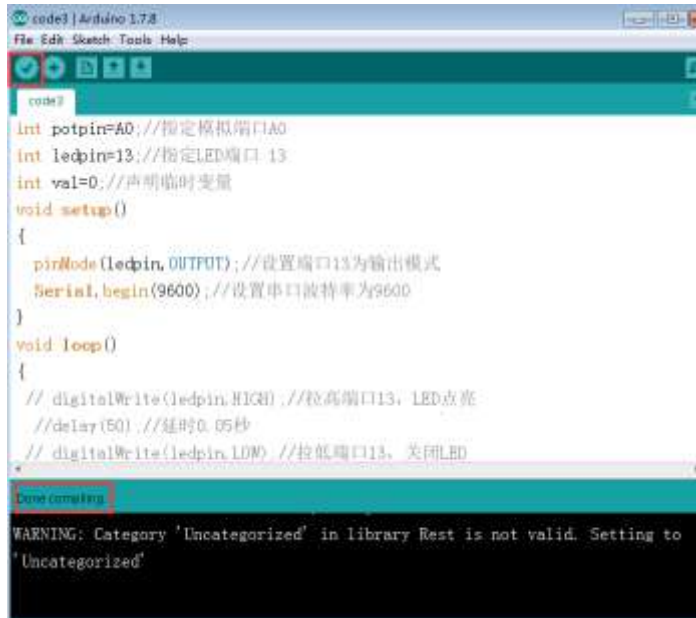
#### Experimental code analysis:

```
int potpin=A0; //Defining the analog port A0
int ledpin=13; //Defining the led port 13
int val=0; //Declarations of temporary variables
void setup()
{
  pinMode(ledpin,OUTPUT); //Defining the light port for the output port
  Serial.begin(9600); //The baud rate is 9600
}
void loop()
{
  val=analogRead(potpin); //Reading the voltage value of the A0 port and assign it to val
```

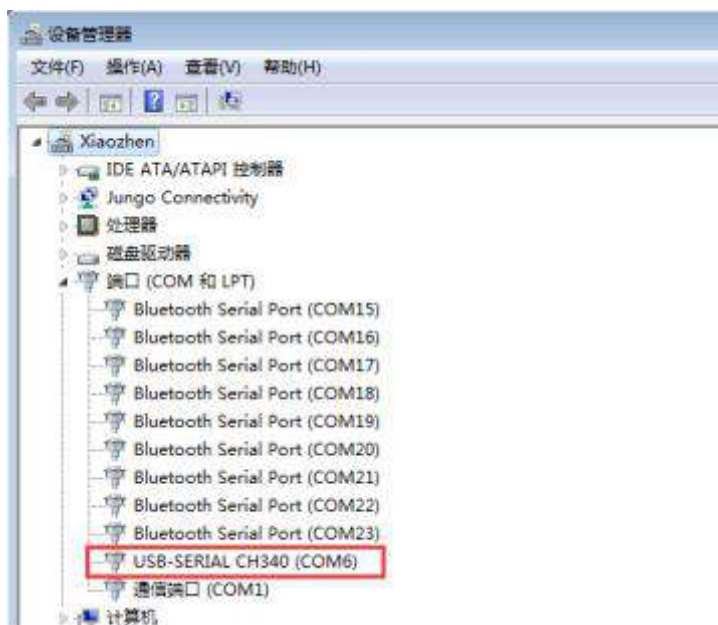
```
Serial.println(val); //Sending Val value by serial port
}
```

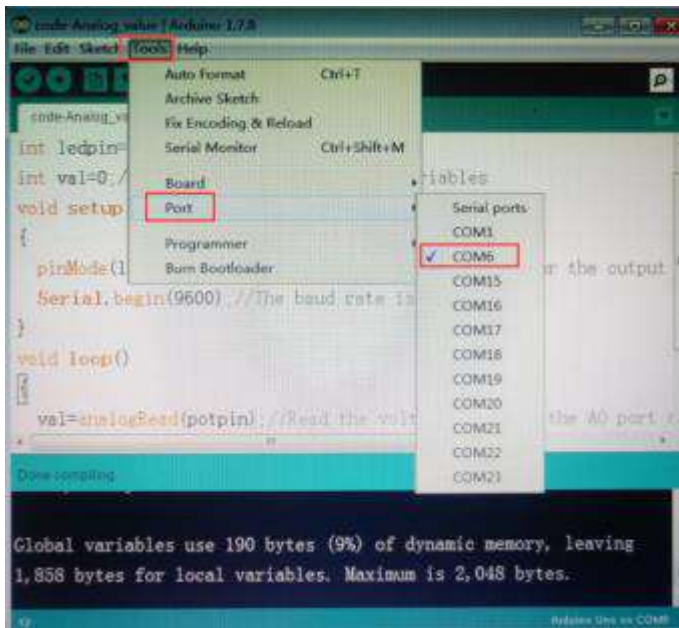
Experimental steps:

1. We need to open the code of this experiment: code-Analog\_value.ino, click“√” under the menu bar to compile the code, and wait for the word "Done compiling " in the lower right corner, as shown in the figure below.

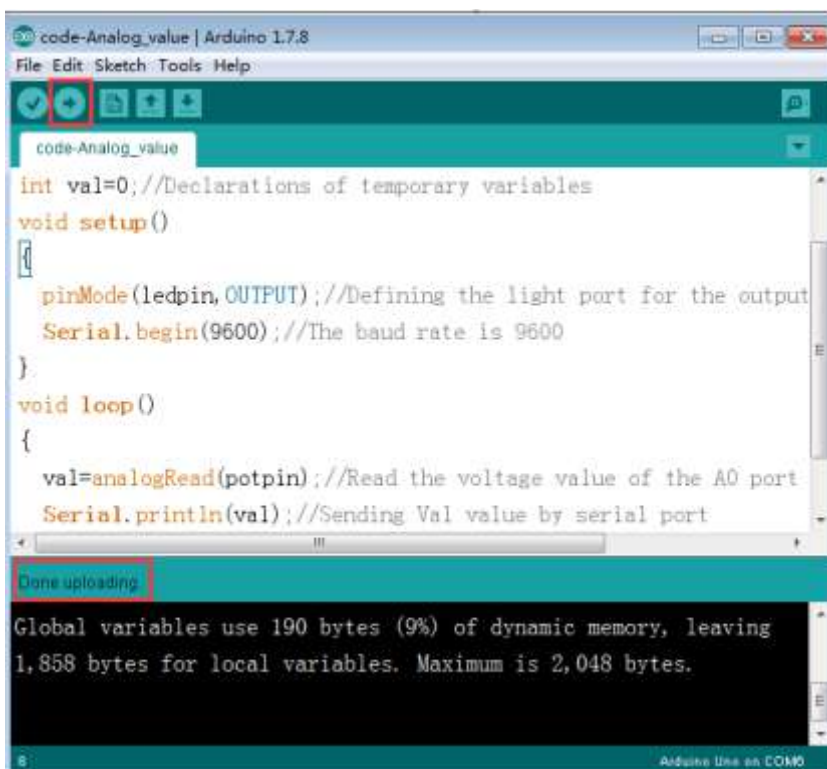


2. In the menu bar of Arduino IDE, we need to select **【Tools】** --- **【Port】** --- selecting the port that the serial number displayed by the device manager just now, as shown in the figure below.



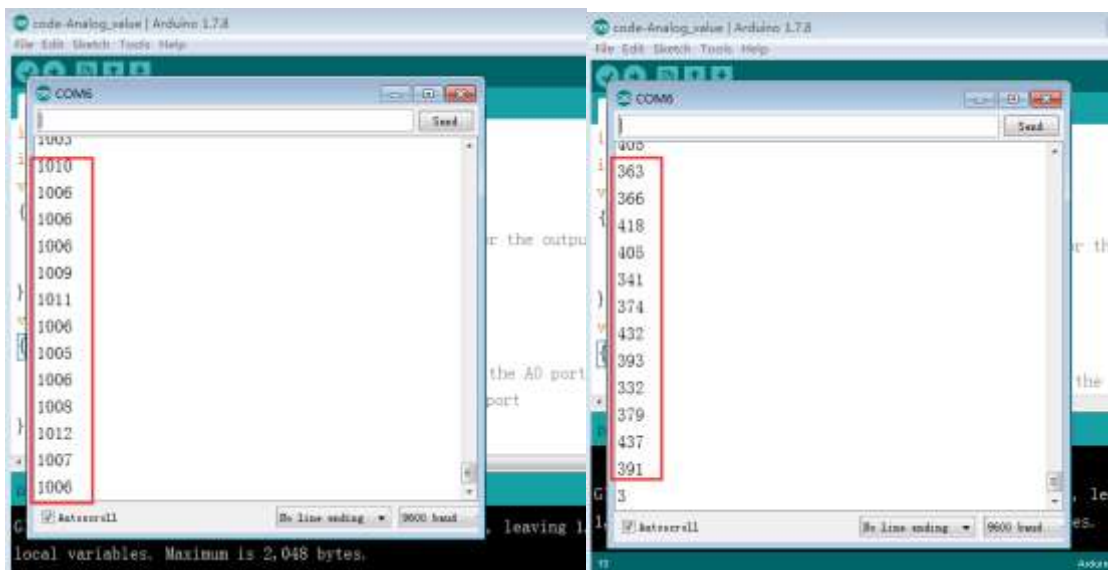
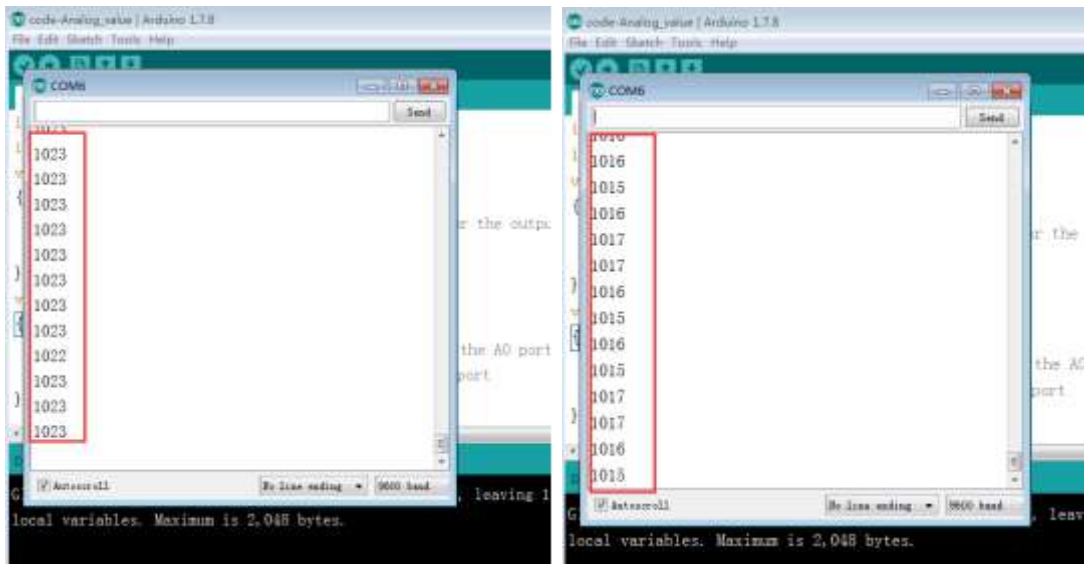


3. After the selection is completed, you need to click “→” under the menu bar to upload the code to the Arduino UNO board. When the word “Done uploading” appears in the lower left corner, the code has been successfully uploaded to the Arduino UNO board, as shown in the figure below.



4. After the code is uploaded. You need to open the serial port monitor on the top right corner of Arduino IDE, A serial port of Arduino port will appear, as shown in the following figure. When we rotate the adjustable resistor, we can see that the data printed in the serial port monitor will change accordingly, as shown in the following figure.







## 4- Advertisement lights

### The purpose of the experiment:

This course is to use the led lights programming to achieve the effect of the simulated advertising lights.

### List of components required for the experiment:

Arduino UNO board \*1

USB cable \*1

LED\*6 (Color random)

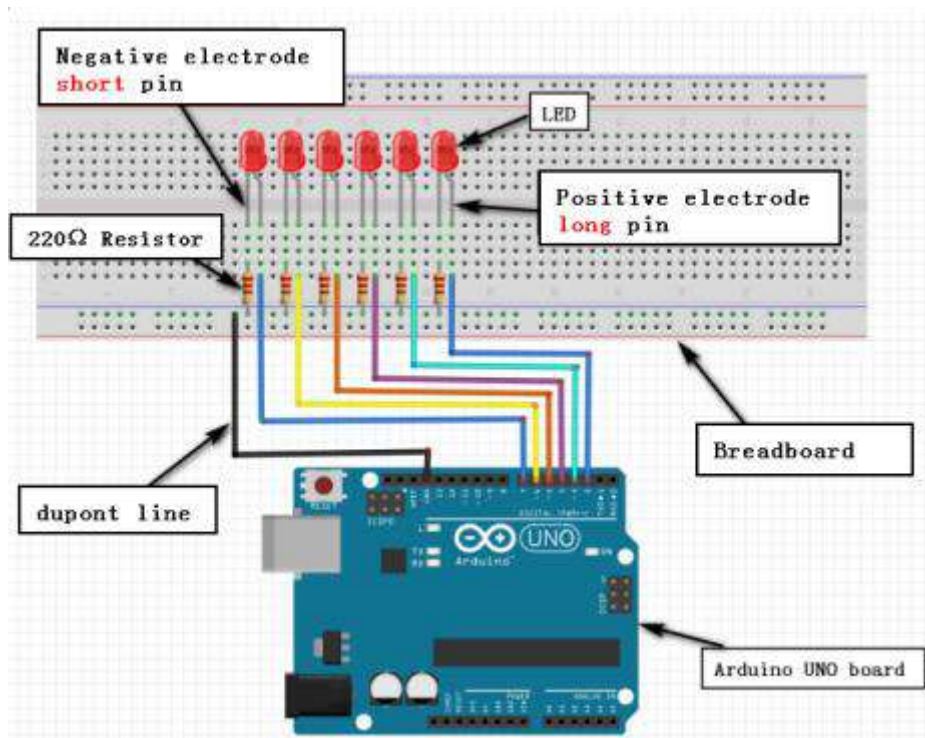
220Ω Resistor \*1

Breadboard \*1

Dupont line \*1bunch

Actual object connection diagram:

We need to connect the circuit as shown in the figure below.



### Experimental code analysis:

```
int BASE = 2 ; //The first LED I/O port
int NUM = 6;  //The total number of LED
int i=0;
void setup()
{
  for (int i = BASE; i < BASE + NUM; i ++)
  {
    pinMode(i, OUTPUT); //Defining the digital I/O port for output port
  }
}
```

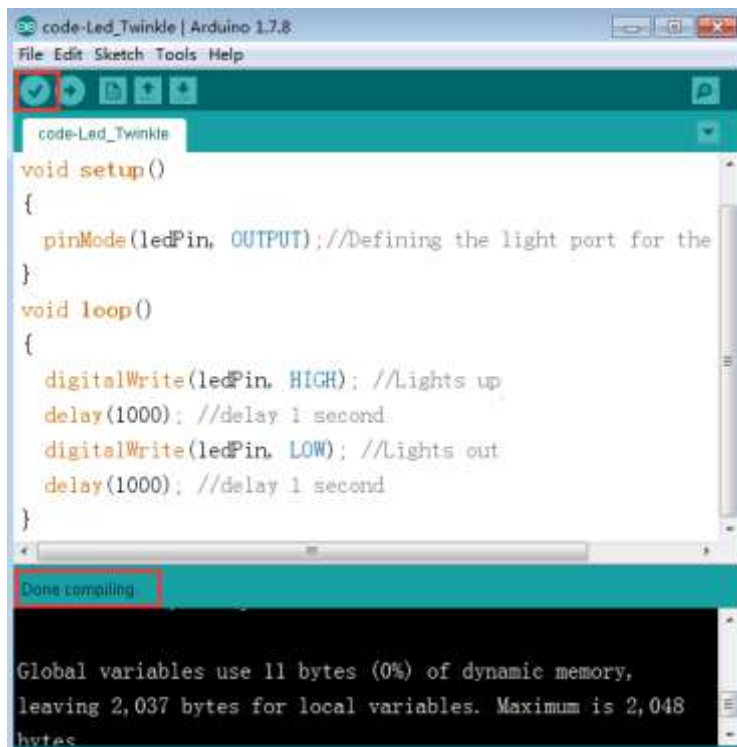
```

void loop()
{
  for (i = BASE; i < BASE + NUM; i++)
  {
    digitalWrite(i, LOW); //Set the number I/O port to be "LOW", that is, the order
    is extinguished.
    delay(200);
  }
  for (i = BASE; i < BASE + NUM; i++)
  {
    digitalWrite(i, HIGH); //Set the number I/O foot to be "HIGH", that is, the order is lit.
    delay(200);
  }
}

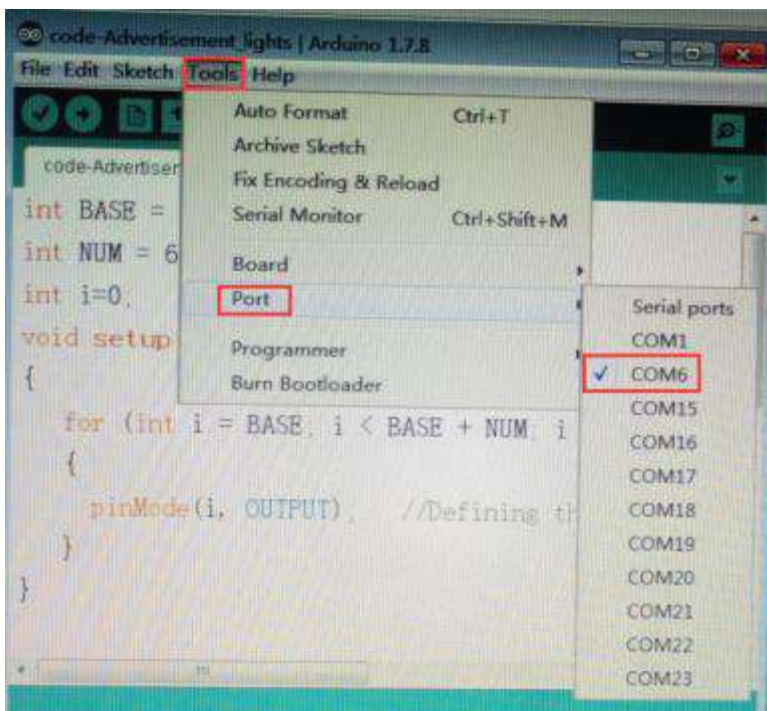
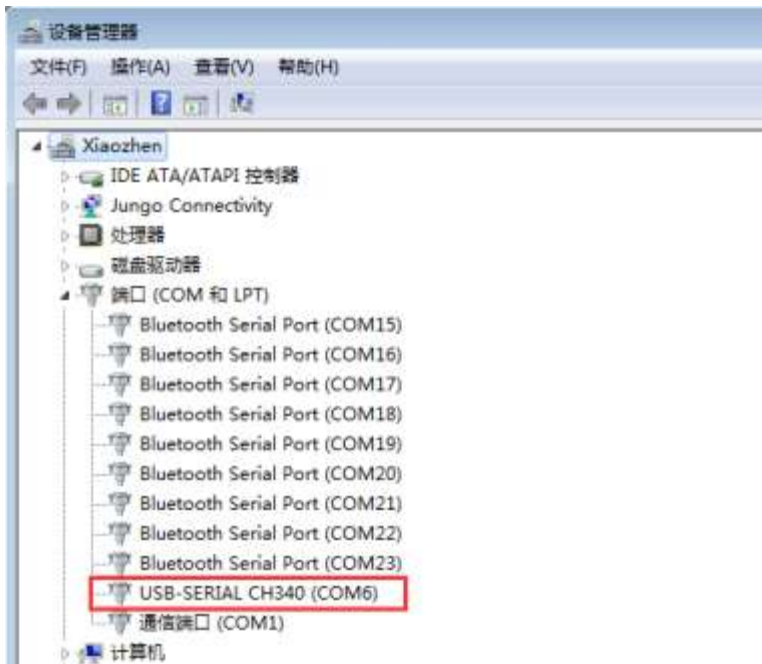
```


Experimental steps:

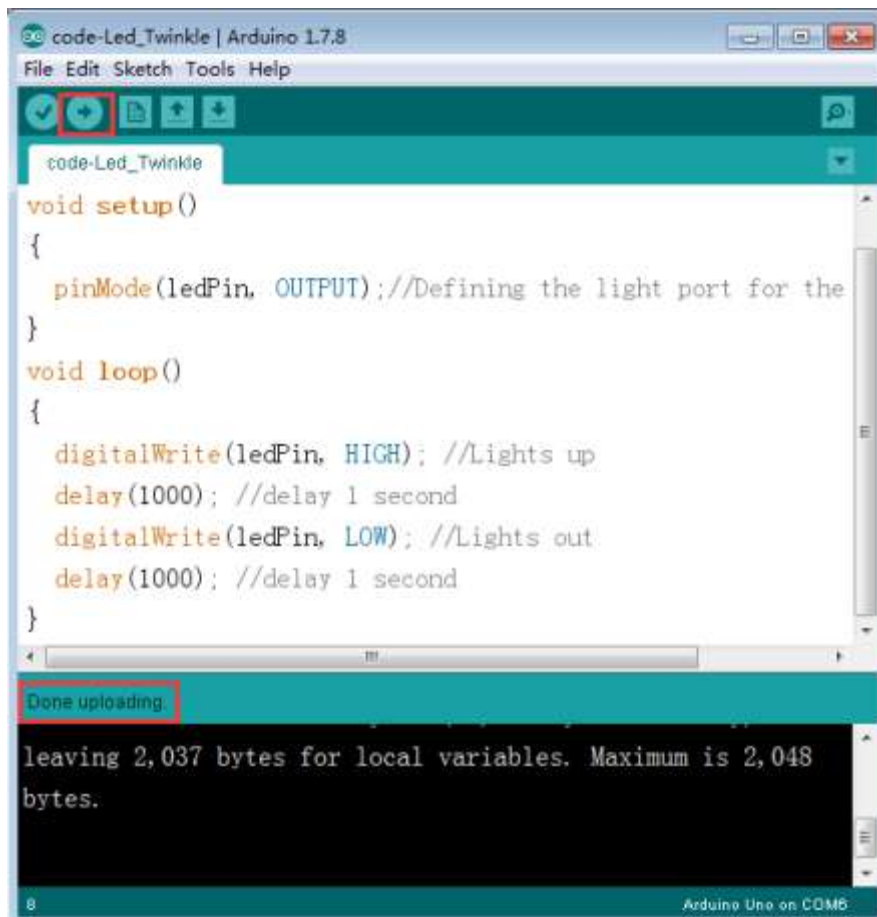
1. We need to open the code of this experiment: code-Advertisement\_lights.ino, click“√” under the menu bar to compile the code, and wait for the word "Done compiling " in the lower right corner, as shown in the figure below.



2. In the menu bar of Arduino IDE, we need to select **【Tools】** --- **【Port】** --- selecting the port that the serial number displayed by the device manager just now, as shown in the figure below.



3. After the selection is completed, you need to click “” under the menu bar to upload the code to the Arduino UNO board. When the word “Done uploading” appears in the lower left corner, the code has been successfully uploaded to the Arduino UNO board, as shown in the figure below.



```
code-Led_Twinkle | Arduino 1.7.8
File Edit Sketch Tools Help

code-Led_Twinkle

void setup()
{
  pinMode(ledPin, OUTPUT); //Defining the light port for the
}

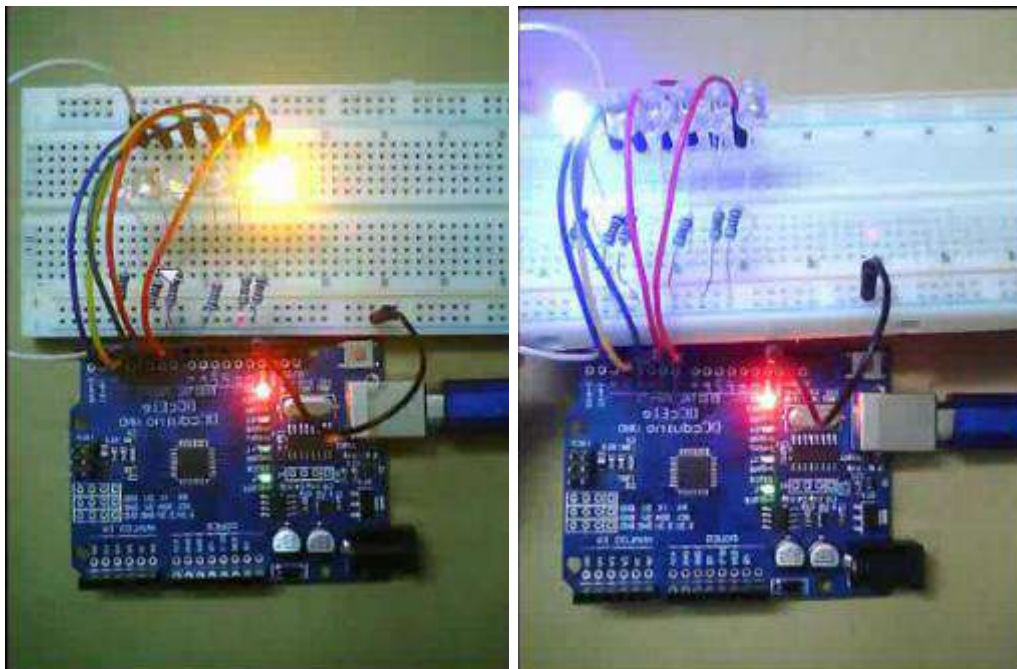
void loop()
{
  digitalWrite(ledPin, HIGH); //Lights up
  delay(1000); //delay 1 second
  digitalWrite(ledPin, LOW); //Lights out
  delay(1000); //delay 1 second
}

Done uploading.

leaving 2,037 bytes for local variables. Maximum is 2,048
bytes.

8 Arduino Uno on COM6
```

4. After the code is uploaded, we can see that 6 LED lights are turned on successively and then turned off successively, as shown in the figure below.





## 5-Traffic lights

### The purpose of the experiment:

This course uses led lights programming to realize the effect of simulated traffic lights.

### List of components required for the experiment:

Arduino UNO board \*1

USB cable \*1

LED\*3 (Color random)

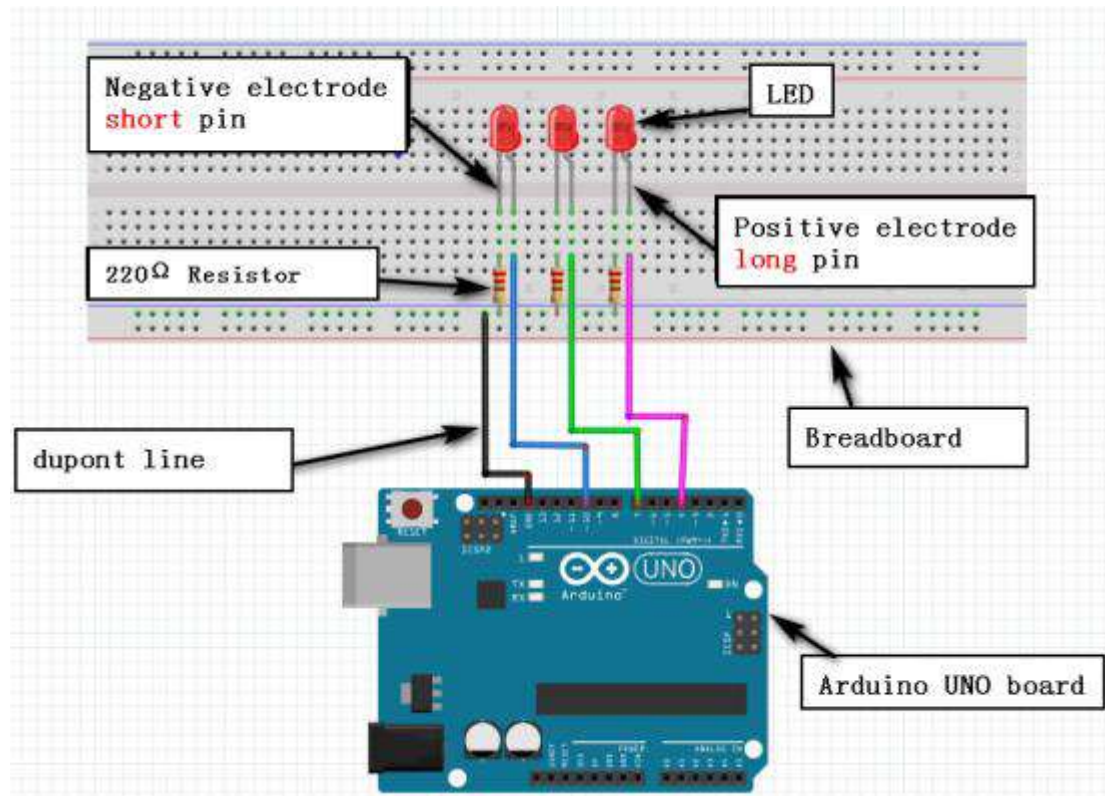
220Ω Resistor \*3

Breadboard \*1

dupont line \*1bunch

Actual object connection diagram:

We need to connect the circuit as shown in the figure below.



### Experimental code analysis:

```
int redled =10; //Defining the digital port 10
int yellowled =7; //Defining the digital port 7
int greenled =4; //Defining the digital port 4
void setup()
{
  pinMode(redled, OUTPUT); //Defining the red light port for the output port
  pinMode(yellowled, OUTPUT); //Defining the yellow light port for the output port
  pinMode(greenled, OUTPUT); //Defining the green light port for the output port
}
void loop()
```

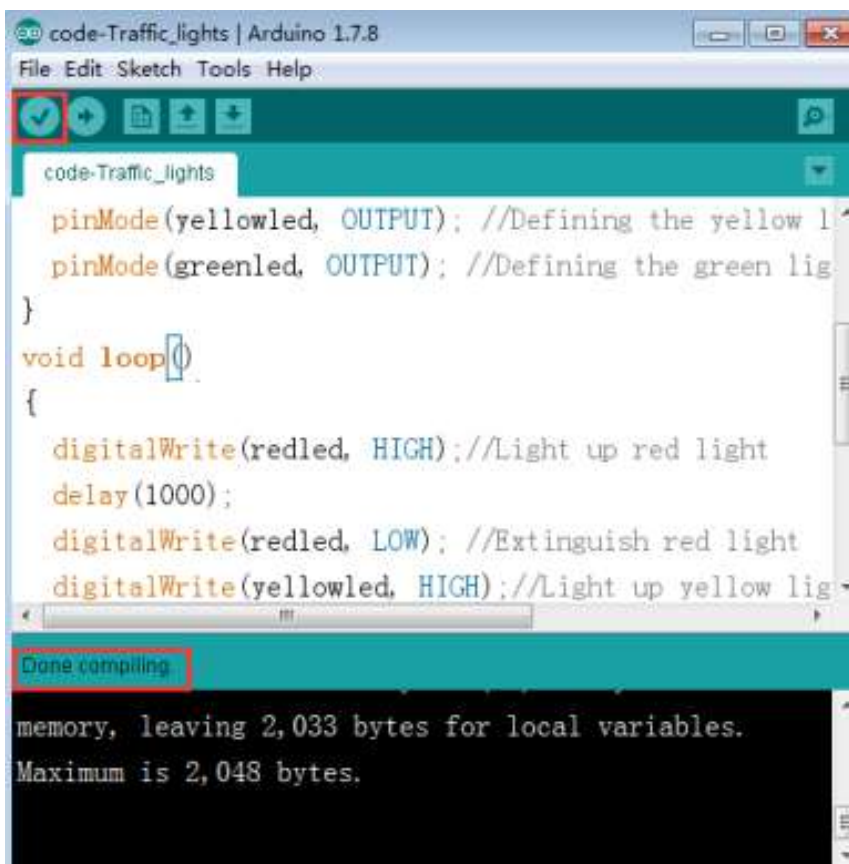
```

{
  digitalWrite(redled, HIGH); //Light up red light
  delay(1000);
  digitalWrite(redled, LOW); //Extinguish red light
  digitalWrite(yellowled, HIGH); //Light up yellow light
  delay(200);
  digitalWrite(yellowled, LOW); //Extinguish yellow light
  digitalWrite(greenled, HIGH); //Light up green light
  delay(1000);
  digitalWrite(greenled, LOW); //Extinguish green light
}

```

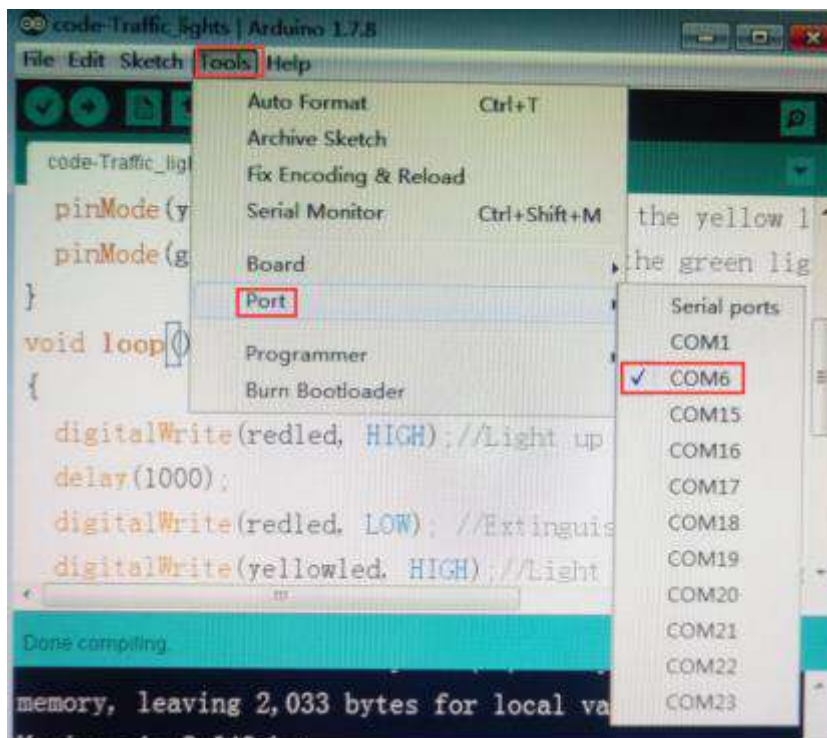
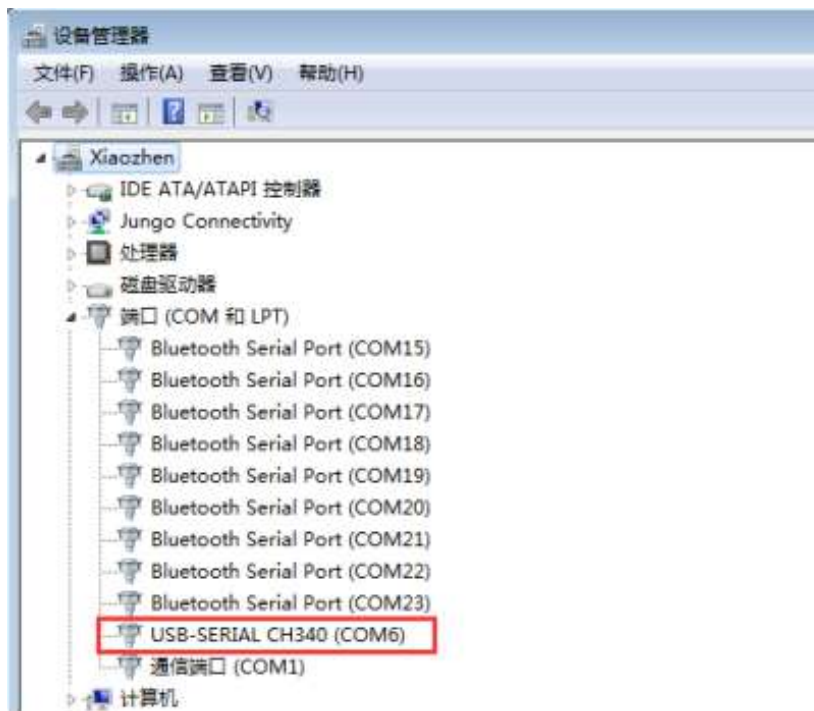
Experimental steps:

1. We need to open the code of this experiment: code-Traffic\_lights.ino, click "✓" under the menu bar to compile the code, and wait for the word "Done compiling" in the lower right corner, as shown in the figure below.

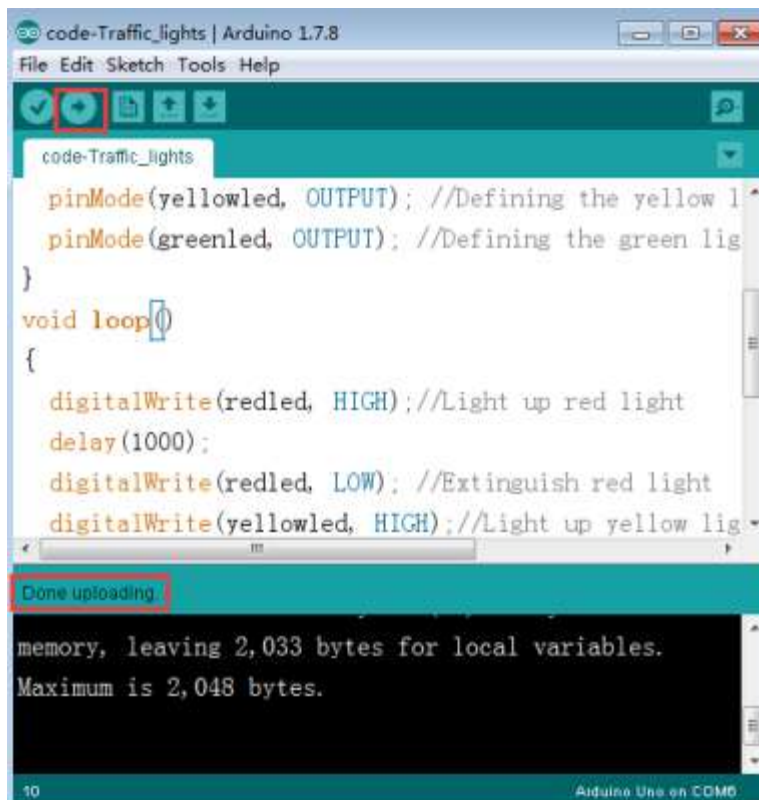


2. In the menu bar of Arduino IDE, we need to select **Tools** --- **Port** --- selecting the port that the serial number displayed by the device manager just now, as shown in the figure below.





3. After the selection is completed, you need to click “→” under the menu bar to upload the code to the Arduino UNO board. When the word “Done uploading” appears in the lower left corner, the code has been successfully uploaded to the Arduino UNO board, as shown in the figure below.



The screenshot shows the Arduino IDE interface with the sketch 'code-Traffic\_lights' open. The code defines three LEDs (red, yellow, green) and a loop that turns them on sequentially. The 'Upload' button is highlighted with a red box. Below the code editor, a status bar indicates 'Done uploading.' and a message box shows the memory usage: 'memory, leaving 2,033 bytes for local variables. Maximum is 2,048 bytes.'

```
code-Traffic_lights | Arduino 1.7.8
File Edit Sketch Tools Help

code-Traffic_lights

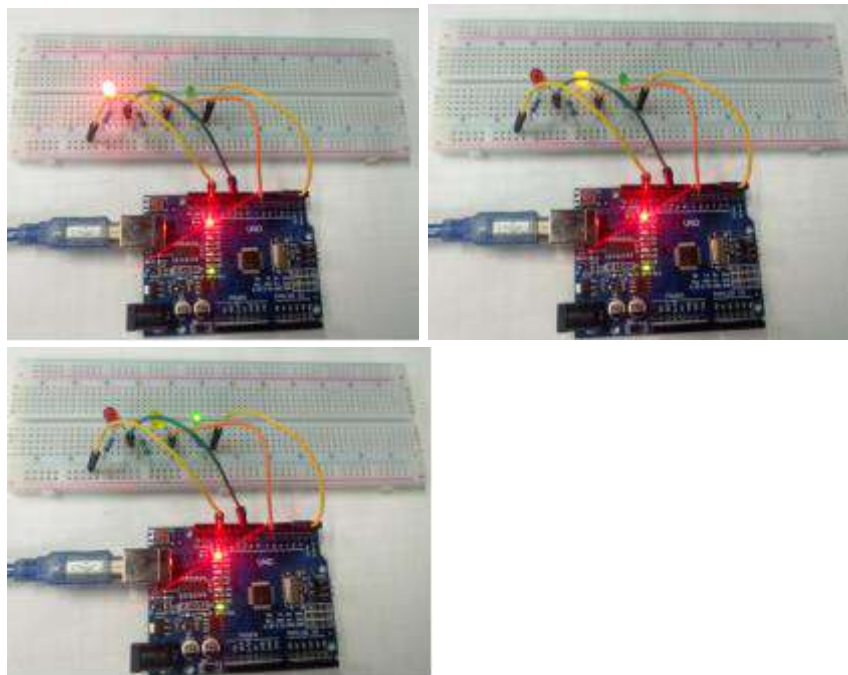
pinMode(yellowled, OUTPUT); //Defining the yellow l
pinMode(greenled, OUTPUT); //Defining the green lig
}
void loop()
{
  digitalWrite(redled, HIGH); //Light up red light
  delay(1000);
  digitalWrite(redled, LOW); //Extinguish red light
  digitalWrite(yellowled, HIGH); //Light up yellow lig
}
```

Done uploading.

memory, leaving 2,033 bytes for local variables.  
Maximum is 2,048 bytes.

10 Arduino Uno on COM6

4. After the code is uploaded, we can see that the red light is on for 1 second, the yellow light is on for 0.2 seconds, and the green light is on for 1 second, as shown in the figure below.



## 6- Key control

### The purpose of the experiment:

In this course, we will learn how to use the input function of I/O port of Arduino, that is, to read the output value of the external device. We use a key and a LED light to complete a combination of the input and output experiments.

### List of components required for the experiment:

Arduino UNO board \*1

USB cable \*1

LED\*1 (Color random)

220 $\Omega$  Resistor \*1

10k $\Omega$  Resistor \*1

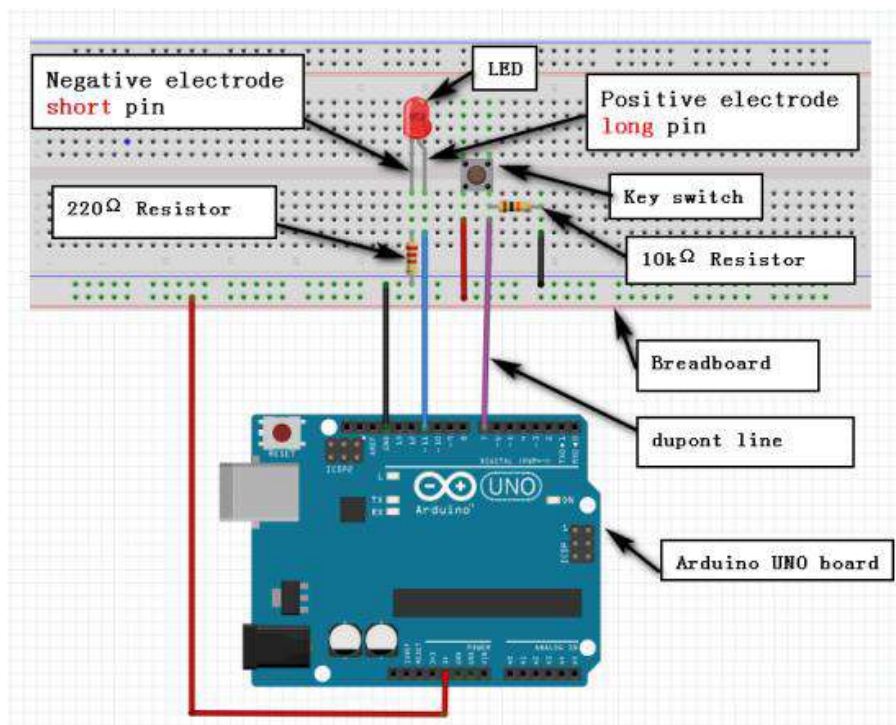
Key switch \*1

Breadboard \*1

Dupont line \*1 bunch

Actual object connection diagram:

We need to connect the circuit as shown in the figure below.



### Experimental code analysis:

```
int ledpin=11;//Defining the digital port 11
int inpin=7;//Defining the digital port 7
int val;//Defining variable
void setup()
{
  pinMode(ledpin,OUTPUT);//Defining the light port for the output port
```

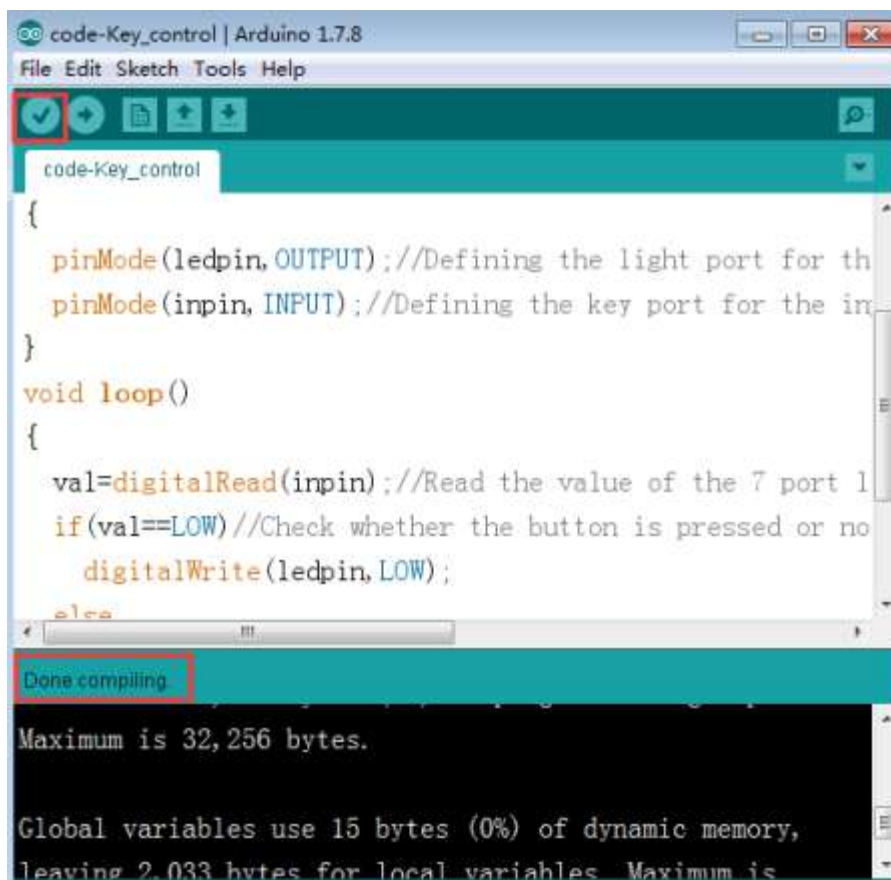
```

pinMode(inpin,INPUT);//Defining the key port for the input port
}
void loop()
{
  val=digitalRead(inpin);//Read the value of the 7 port level to the val
  if(val==LOW)//Check whether the button is pressed or not. When the button is
  pressed, the light will turn on.
    digitalWrite(ledpin,LOW);
  else
    digitalWrite(ledpin,HIGH);
}

```

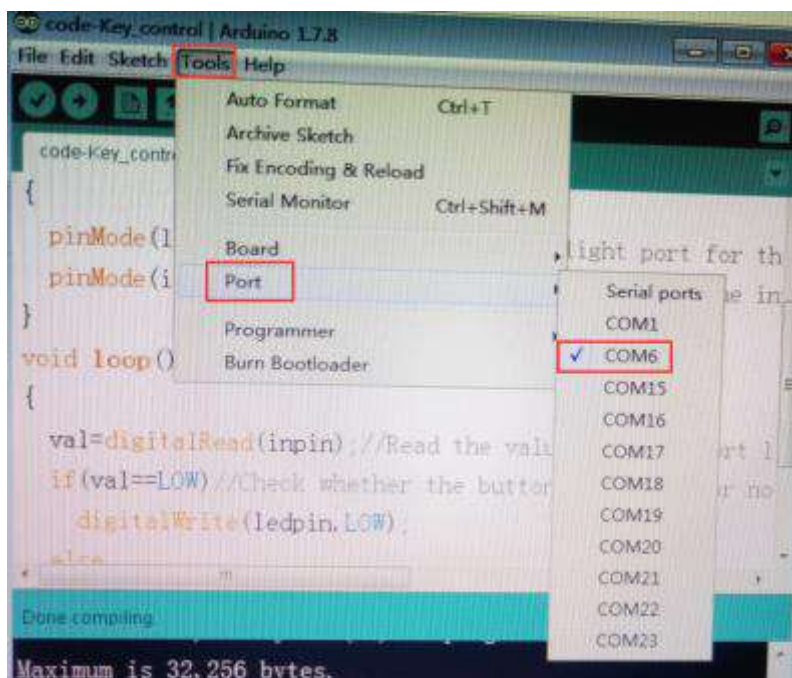
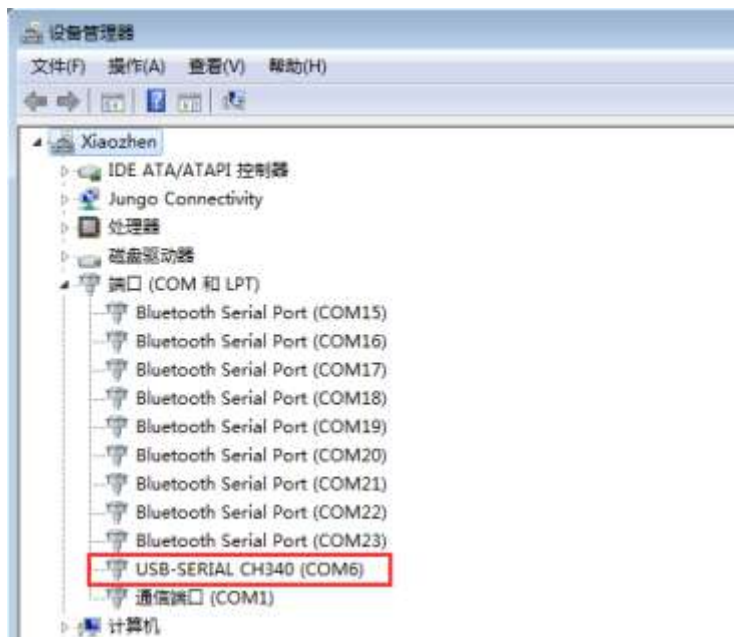
Experimental steps:

1. We need to open the code of this experiment: code-Key control.ino, click“√” under the menu bar to compile the code, and wait for the word "Done compiling " in the lower right corner, as shown in the figure below.

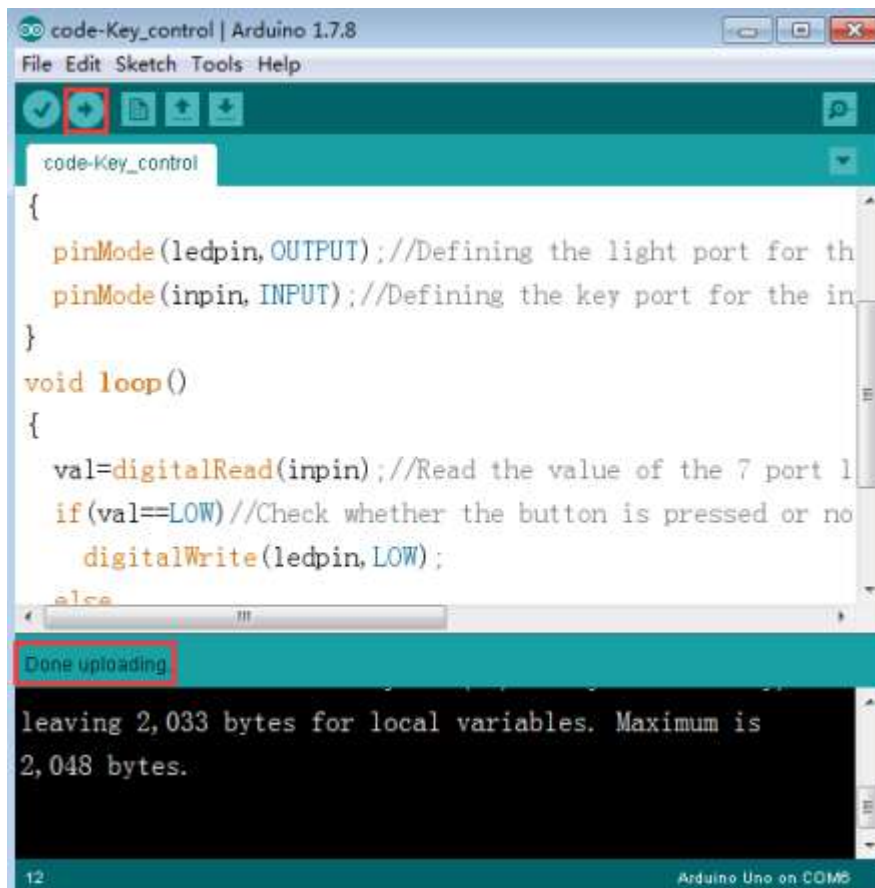


2. In the menu bar of Arduino IDE, we need to select **【Tools】** --- **【Port】** --- selecting the port that the serial number displayed by the device manager just now, as shown in the figure below. for example:COM6,as shown in the following figure.





3. After the selection is completed, you need to click “→” under the menu bar to upload the code to the Arduino UNO board. When the word “Done uploading” appears in the lower left corner, the code has been successfully uploaded to the Arduino UNO board, as shown in the figure below.



```
code-Key_control | Arduino 1.7.8
File Edit Sketch Tools Help

code-Key_control

{
  pinMode(ledpin, OUTPUT); //Defining the light port for th
  pinMode(inpin, INPUT); //Defining the key port for the in
}

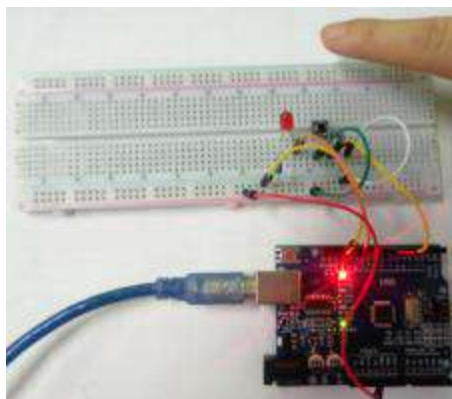
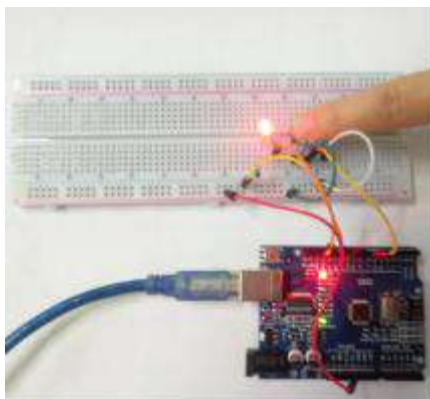
void loop()
{
  val=digitalRead(inpin); //Read the value of the 7 port 1
  if(val==LOW) //Check whether the button is pressed or no
    digitalWrite(ledpin, LOW);
  else
    digitalWrite(ledpin, HIGH);
}

Done uploading

leaving 2,033 bytes for local variables. Maximum is
2,048 bytes.

12 Arduino Uno on COM5
```

4. After the code is uploaded, when the button is pressed, the LED light will be on, and the LED light will be off when the button is released, as shown in the figure below.





## 7- Answering machine

### The purpose of the experiment:

This experiment is to extend the experiment of button control LED lights in the previous class into 3 buttons corresponding to 3 LED lights, which requires 6 digital I/O interfaces of Arduino.

### List of components required for the experiment:

Arduino UNO board \*1

USB cable \*1

LED\*3 (Color random)

220 $\Omega$  Resistor \*3

10k $\Omega$  Resistor \*3

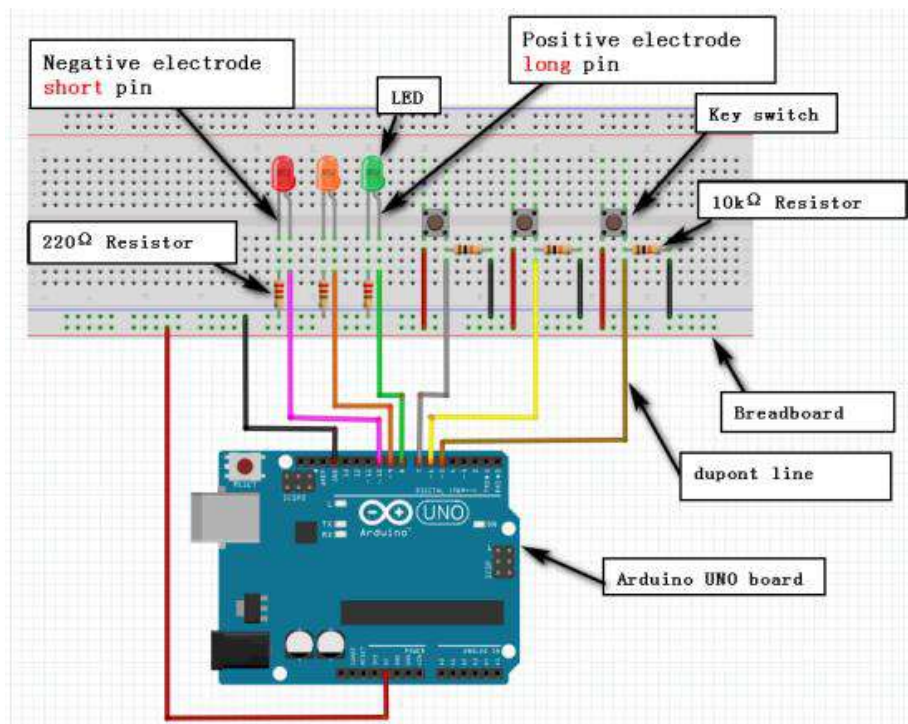
Key switch \*3

Breadboard \*1

Dupont line \*1bunch

Actual object connection diagram:

We need to connect the circuit as shown in the figure below.



### Experimental code analysis:

```
int redled=10; //set IO 10 to red LED
int yellowled=9; //set IO 10 to yellow LED
int greenled=8; //set IO 10 to green LED
int redpin=7; //red key pin IO 7
int yellowpin=6; //yellow key pin IO 6
int greenpin=5; //green key pin IO 5
```

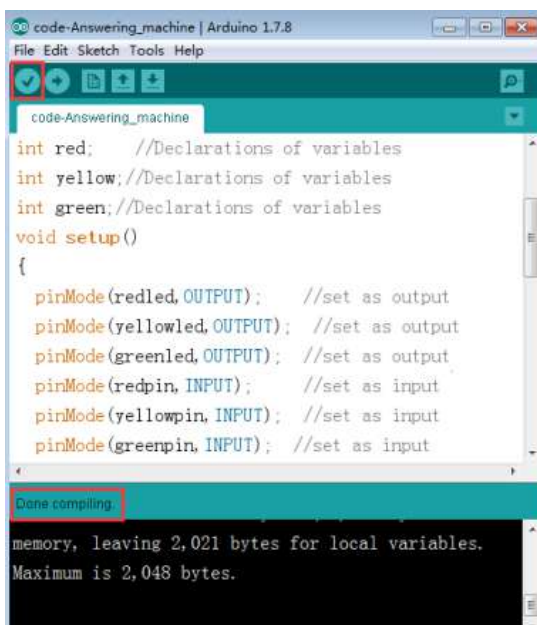
```

int red; //Declarations of variables
int yellow; //Declarations of variables
int green; //Declarations of variables
void setup()
{
  pinMode(redled,OUTPUT); //set as output
  pinMode(yellowled,OUTPUT); //set as output
  pinMode(greenled,OUTPUT); //set as output
  pinMode(redpin,INPUT); //set as input
  pinMode(yellowpin,INPUT); //set as input
  pinMode(greenpin,INPUT); //set as input
}
void loop()
{
  red=digitalRead(redpin);//Reading key state
  if(red==LOW) //Key state is LOW
  { digitalWrite(redled,LOW);}//LED turn off
  else //Key state is HIGH
  { digitalWrite(redled,HIGH);}//LED turn on
  yellow=digitalRead(yellowpin);
  if(yellow==LOW)
  { digitalWrite(yellowled,LOW);}
  else
  { digitalWrite(yellowled,HIGH);}
  green=digitalRead(greenpin);
  if(green==LOW)
  { digitalWrite(greenled,LOW);}
  else
  { digitalWrite(greenled,HIGH);}
}

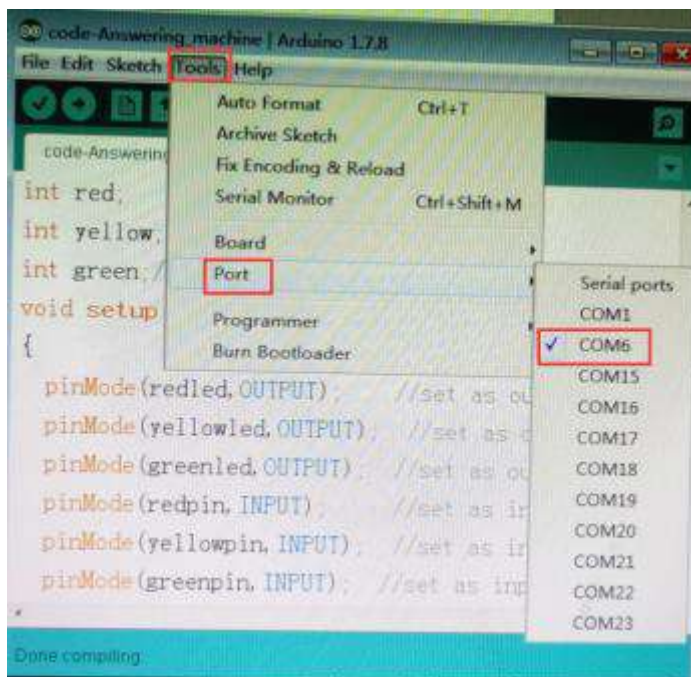
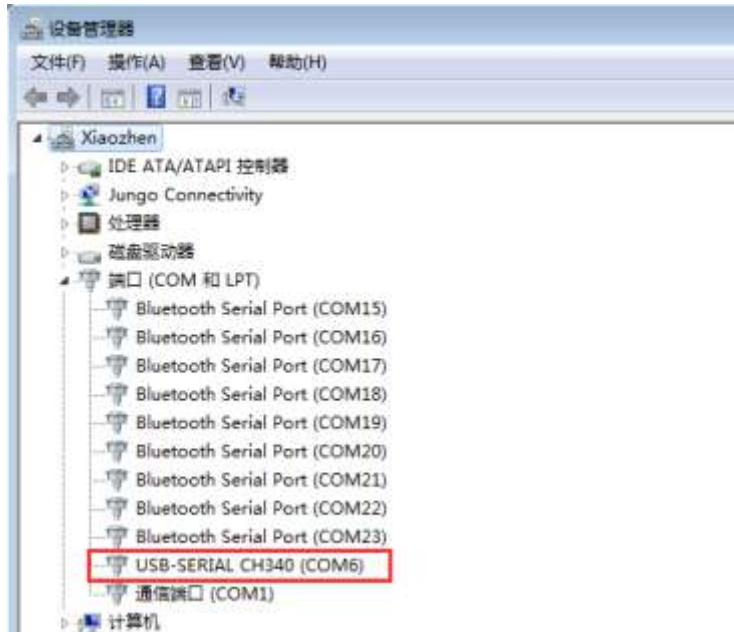
```

Experimental steps:

1. We need to open the code of this experiment: code-Answering machine.ino, click "✓" under the menu bar to compile the code, and wait for the word "Done compiling" in the lower right corner, as shown in the figure below.



2. In the menu bar of Arduino IDE, we need to select **【Tools】** --- **【Port】** --- selecting the port that the serial number displayed by the device manager just now, as shown in the figure below. for example: COM6, as shown in the following figure.



3. After the selection is completed, you need to click “→” under the menu bar to upload the code to the Arduino UNO board. When the word “Done uploading” appears in the lower left corner, the code has been successfully uploaded to the Arduino UNO board, as shown in the figure below.

```

code-Answering_machine | Arduino 1.7.8
File Edit Sketch Tools Help

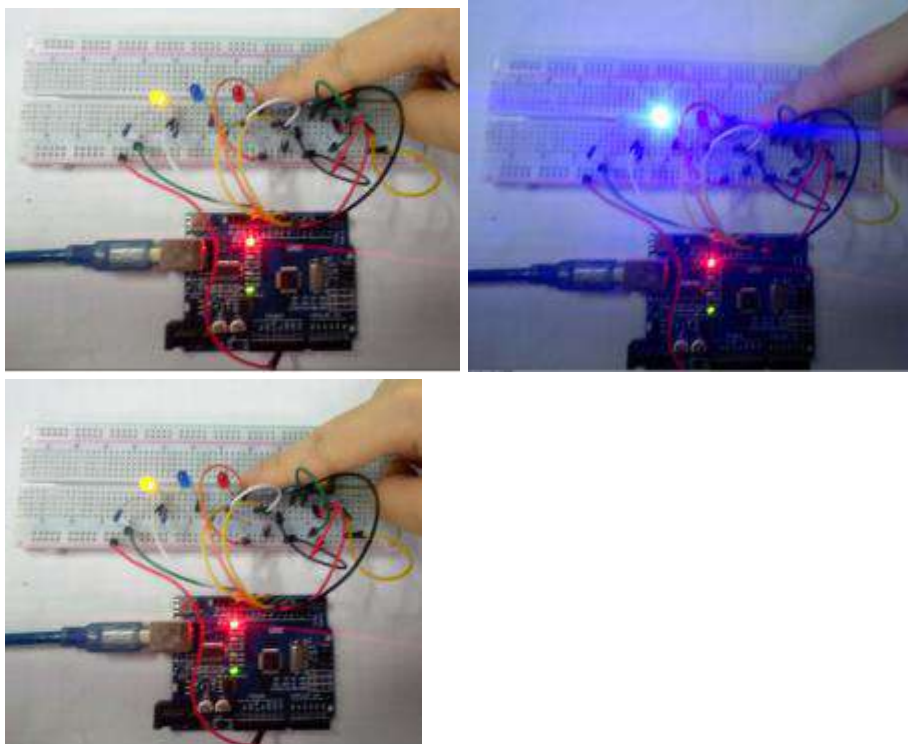
code-Answering_machine

int red;    //Declarations of variables
int yellow; //Declarations of variables
int green;  //Declarations of variables
void setup()
{
  pinMode(redled, OUTPUT);    //set as output
  pinMode(yellowled, OUTPUT); //set as output
  pinMode(greenled, OUTPUT);  //set as output
  pinMode(redpin, INPUT);     //set as input
  pinMode(yellowpin, INPUT);  //set as input
  pinMode(greenpin, INPUT);   //set as input
}

Done uploading.
memory, leaving 2,021 bytes for local variables.
Maximum is 2,048 bytes.
14 Arduino Uno on COM5

```

4. After the code is uploaded, when different buttons are pressed, the LED lights of different colors will be turned on and the LED lights will be extinguished when the button is released, as shown in the figure below.





## 8.1- Active buzzer

### The purpose of the experiment:

In this course, we need to use active buzzer to make an experiment to make the circuit sound.

### List of components required for the experiment:

Arduino UNO board \*1

USB cable \*1

220 $\Omega$  Resistor \*1

Active buzzer \*1

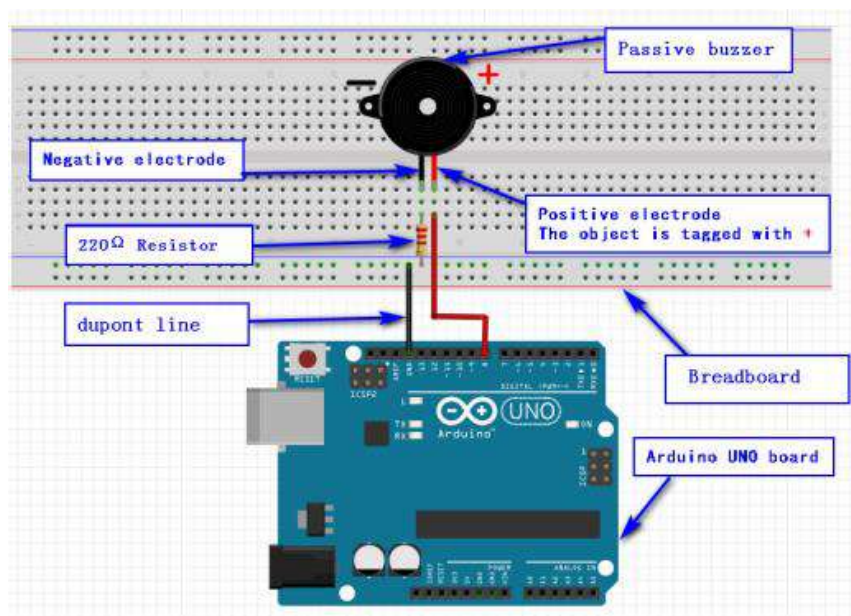
Breadboard \*1

Dupont line \*1bunch

Actual object connection diagram:

We need to connect the circuit as shown in the figure below.

Note: The active buzzer has positive and negative electrode. The actual object diagram below shows that the buzzer has positive and negative marks.



### Experimental code analysis:

```
int buzzer=8; //Defining the digital port 8 to control the buzzer
int i = 0;
void setup()
```

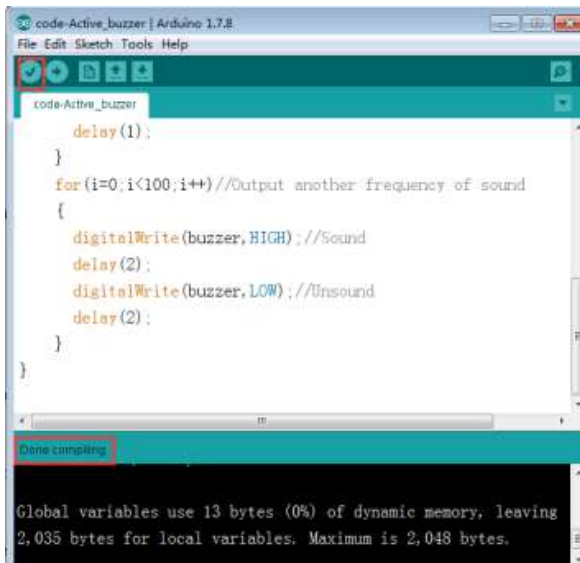
```

{
pinMode(buzzer,OUTPUT); //Defining the buzzer port for the output port
}
void loop()
{
  for(i=0;i<80;i++) //Output a frequency of sound
  {
    digitalWrite(buzzer,HIGH); //Sound
    delay(1);
    digitalWrite(buzzer,LOW); //Unsound
    delay(1);
  }
  for(i=0;i<100;i++) //Output another frequency of sound
  {
    digitalWrite(buzzer,HIGH); //Sound
    delay(2);
    digitalWrite(buzzer,LOW); //Unsound
    delay(2);
  }
}

```

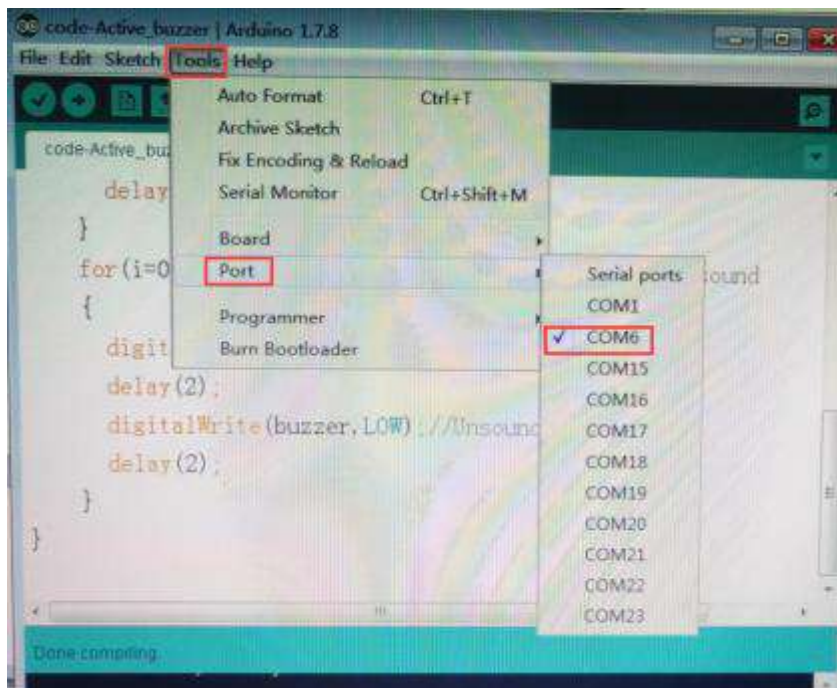
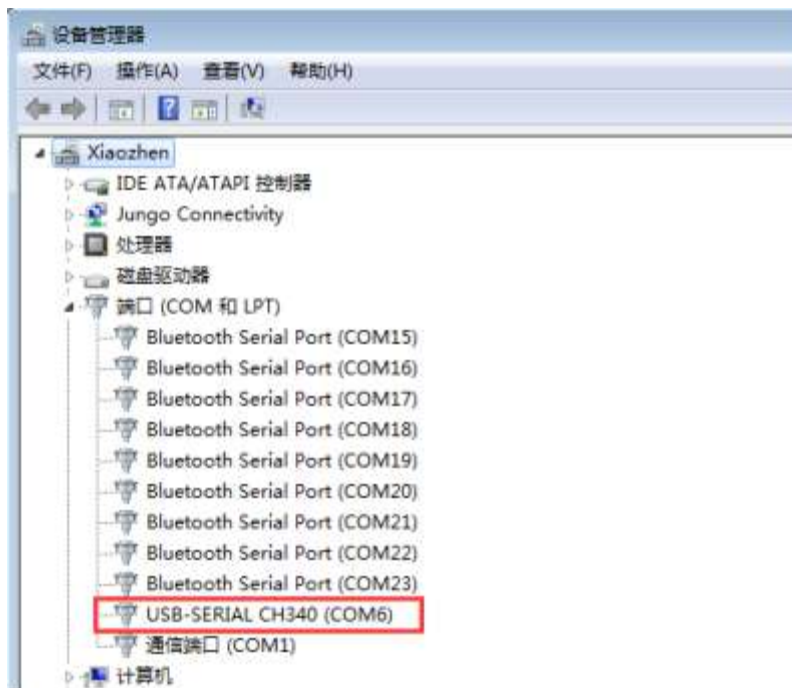
Experimental steps:

1. We need to open the code of this experiment: code-Active\_buzzer.ino, click“√” under the menu bar to compile the code, and wait for the word "Done compiling " in the lower right corner, as shown in the figure below.

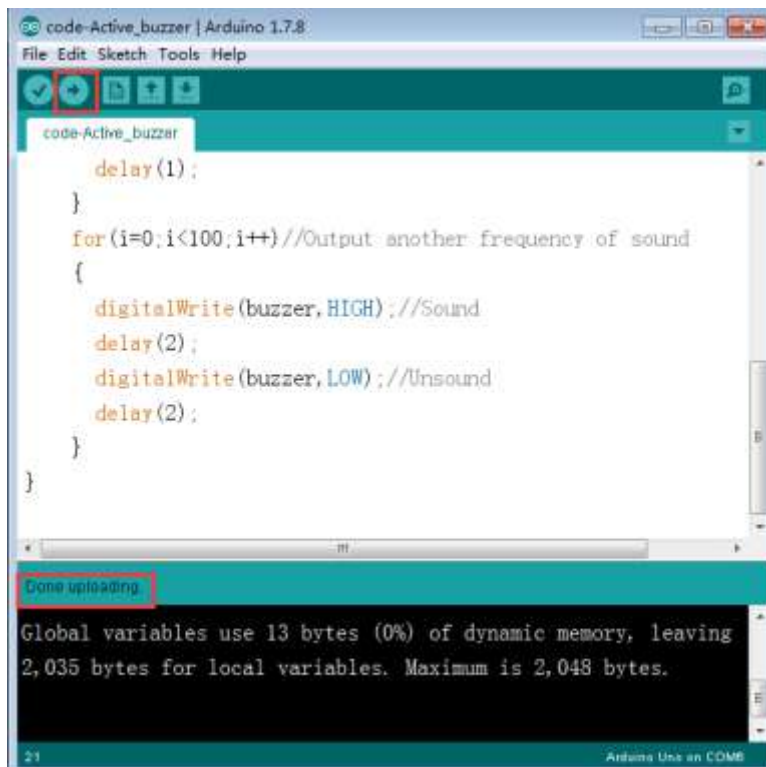


2. In the menu bar of Arduino IDE, we need to select **Tools** --- **Port** --- selecting the port that the serial number displayed by the device manager just now, as shown in the figure below. For example: COM6, as shown in the following figure.





3. After the selection is completed, you need to click “→” under the menu bar to upload the code to the Arduino UNO board. When the word “Done uploading” appears in the lower left corner, the code has been successfully uploaded to the Arduino UNO board, as shown in the figure below.

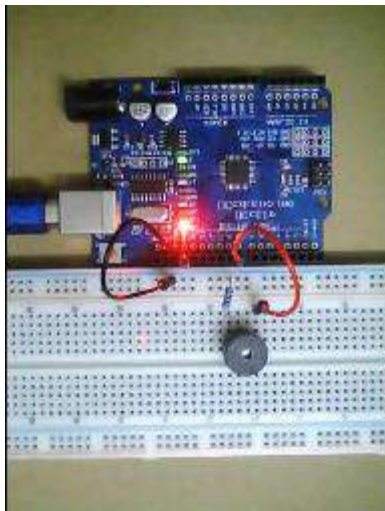


The screenshot shows the Arduino IDE interface. The top menu bar includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. The toolbar has icons for opening, saving, and uploading. The sketch editor displays the following code:

```
code-Active_buzzer
    delay(1);
}
for(i=0;i<100;i++)//Output another frequency of sound
{
    digitalWrite(buzzer,HIGH);//Sound
    delay(2);
    digitalWrite(buzzer,LOW);//Unsound
    delay(2);
}
}
```

At the bottom, a status bar indicates 'Done uploading.' and a message box shows: 'Global variables use 13 bytes (0%) of dynamic memory, leaving 2,035 bytes for local variables. Maximum is 2,048 bytes.'

4. After the code is uploaded, we can hear the buzzer making two different frequencies in succession, as shown in the following figure.



## 8.2-Passive buzzer

### The purpose of the experiment:

In this course, we need to use passive buzzer to make an experiment to make the circuit sound.

### List of components required for the experiment:

Arduino UNO board \*1

USB cable \*1

220Ω Resistor \*1

Passive buzzer \*1

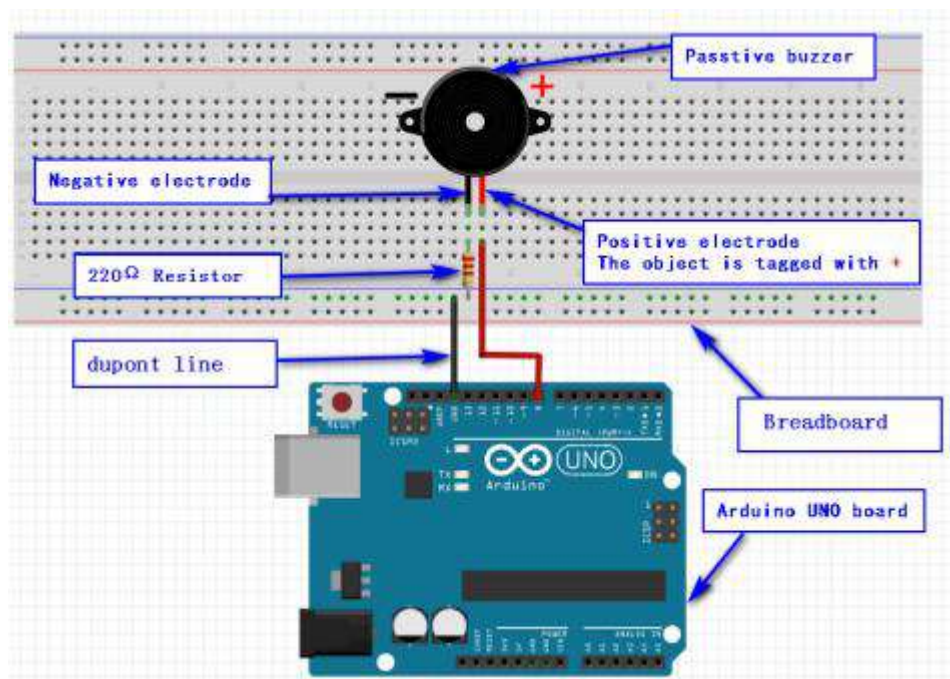
Breadboard \*1

Dupont line \*1bunch

Actual object connection diagram:

We need to connect the circuit as shown in the figure below.

Note: The passive buzzer has positive and negative electrode. The actual object diagram below shows that the buzzer has positive and negative marks.



### Experimental code analysis:

```
#define BL1 248
#define BL2 278
```

```
#define BL3 294
#define BL4 330
#define BL5 371
#define BL6 416
#define BL7 467
```

```
#define B1 495
#define B2 556
#define B3 624
#define B4 661
#define B5 742
#define B6 833
#define B7 935
```

```
#define BH1 990
#define BH2 1112
#define BH3 1178
#define BH4 1322
#define BH5 1484
#define BH6 1665
#define BH7 1869
```

```
#define NTC1 262
#define NTC2 294
#define NTC3 330
#define NTC4 350
#define NTC5 393
#define NTC6 441
#define NTC7 495
```

```
#define NTCL1 131
#define NTCL2 147
#define NTCL3 165
#define NTCL4 175
#define NTCL5 196
#define NTCL6 221
#define NTCL7 248
```

```
#define NTCH1 525
#define NTCH2 589
#define NTCH3 661
#define NTCH4 700
#define NTCH5 786
#define NTCH6 882
#define NTCH7 990
```

```
#define NTD0 -1
#define NTD1 294
#define NTD2 330
#define NTD3 350
#define NTD4 393
#define NTD5 441
```

```
#define NTD6 495
#define NTD7 556
```

```
#define NTDL1 147
#define NTDL2 165
#define NTDL3 175
#define NTDL4 196
#define NTDL5 221
#define NTDL6 248
#define NTDL7 278
```

```
#define NTDH1 589
#define NTDH2 661
#define NTDH3 700
#define NTDH4 786
#define NTDH5 882
#define NTDH6 990
#define NTDH7 1112
```

```
#define NTE1 330
#define NTE2 350
#define NTE3 393
#define NTE4 441
#define NTE5 495
#define NTE6 556
#define NTE7 624
```

```
#define NTEL1 165
#define NTEL2 175
#define NTEL3 196
#define NTEL4 221
#define NTEL5 248
#define NTEL6 278
#define NTEL7 312
```

```
#define NTEH1 661
#define NTEH2 700
#define NTEH3 786
#define NTEH4 882
#define NTEH5 990
#define NTEH6 1112
#define NTEH7 1248
```

```
int speakerPin= 8;
int buzzer=8;//Defining the digital port 8 to control the buzzer
int i = 0;
/*YeZi C*/
int tune[]=          //List the frequencies according to the simple spectrum
{
  NTC3, NTC5, NTC5, NTC3, NTC6, NTC6, NTC7, NTC6, NTC6, NTC6, NTC5, NTCH1,
  NTCH1, NTCH1, NTCH1, NTC6,NTCH1, NTC6, NTC5,NTC3, NTC5, NTC5, NTC5,
  NTC3, NTC6, NTC6, NTC7, NTC6, NTC6, NTC6, NTC5, NTCH1, NTCH1,
```



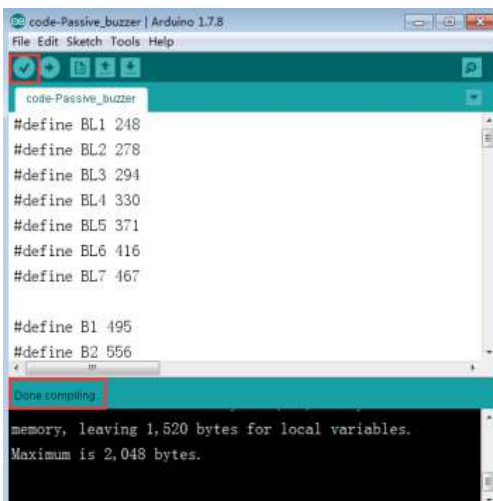
```

NTCH1,NTCH1, NTC6, NTC6, NTCH1,NTCH2,NTCH5, NTCH5, NTCH5,
NTCH5,NTCH5, NTCH3, NTCH2, NTCH1, NTCH1, NTC6, NTCH1, NTC6,
NTCH1,NTCH2,NTCH2, NTCH2, NTCH2,NTCH2, NTCH1, NTCH3, NTCH2,
NTCH2,NTCH3, NTCH3, NTCH3, NTCH3, NTCH2, NTCH2, NTCH1, NTCH1, NTCH1,
NTCH2, NTCH1, NTC6, NTC5, NTC5,NTC5, NTC5, NTC6, NTC5,NTCH2,
NTCH3,NTCH1,
};
float durt[]=          //List the tempo according to the simple spectrum
{
0.5, 0.5, 1.5, 0.5, 0.5, 0.5, 0.5, 1, 0.5, 0.5, 0.5, 1, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 2, 0.5, 0.5,
1, 0.5, 0.5, 0.5, 0.5, 0.5, 1, 0.5, 0.5, 0.5, 1, 0.5, 0.5,0.5, 0.5, 0.5, 0.5, 2,0.5, 1, 0.5, 0.5,
0.5, 0.5, 0.5, 0.5, 1, 1, 0.5, 0.5, 0.5, 1, 0.25, 0.5, 0.5, 0.5, 0.5, 1, 0.25, 2,0.5, 1, 0.5, 0.5,
0.5, 1, 0.5, 1, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 1,0.5, 0.5, 0.5, 0.5, 0.5, 2,
};
void PlayTest()
{
int length = sizeof(tune)/sizeof(tune[0]); //Calculate length
for(int x=0; x < length;x++)
{
tone(speakerPin,tune[x]);
delay(500*durt[x]); //This is used to adjust the time delay according to the beat,500
this index can be adjusted by yourself. In this music, I find that 500 is more suitable.
noTone(speakerPin);
}
}
void setup()
{
pinMode(buzzer,OUTPUT); //Defining the buzzer port for the output port
}
void loop()
{
PlayTest();
}

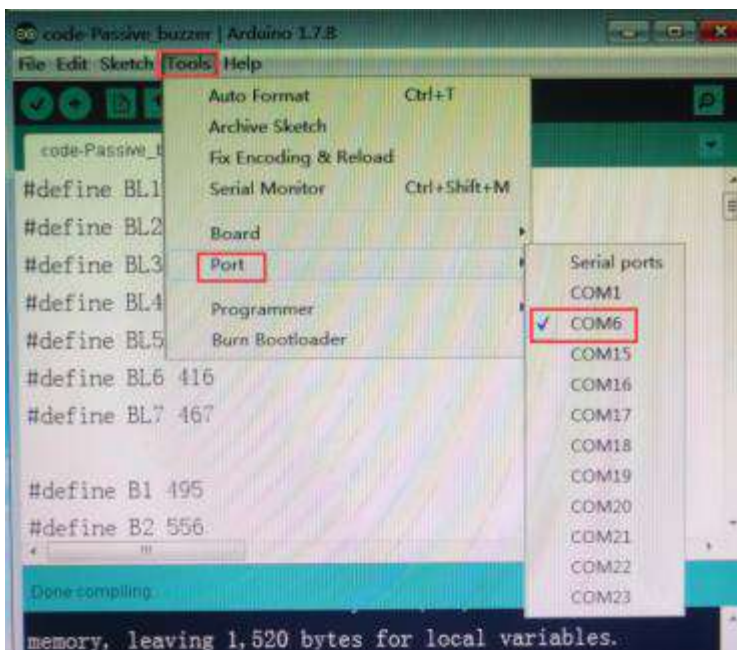
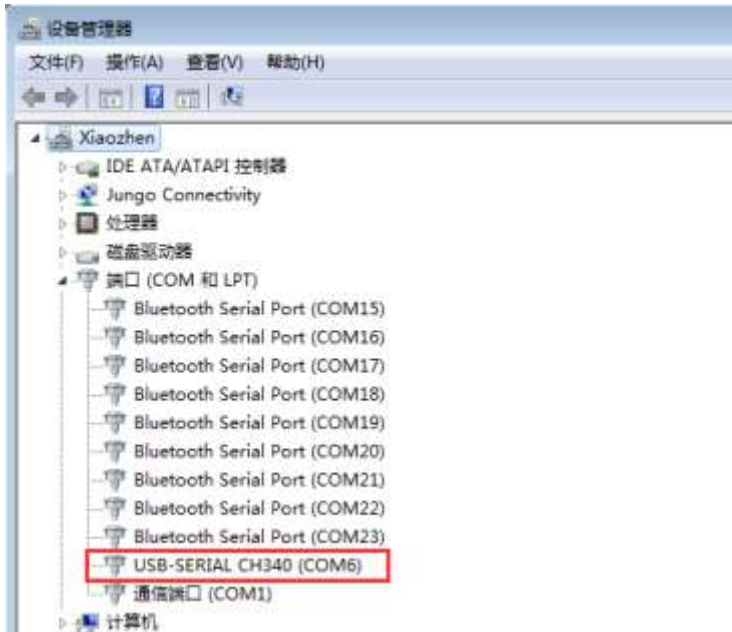
```

Experimental steps:

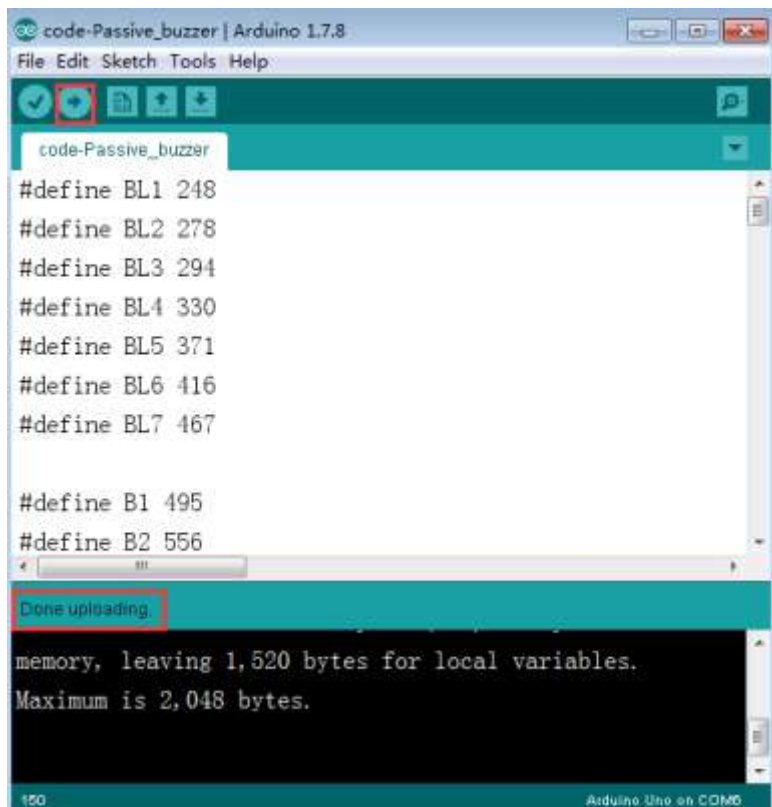
1.We need to open the code of this experiment: code-Passive\_buzzer.ino, click“√” under the menu bar to compile the code, and wait for the word "Done compiling " in the lower right corner, as shown in the figure below.



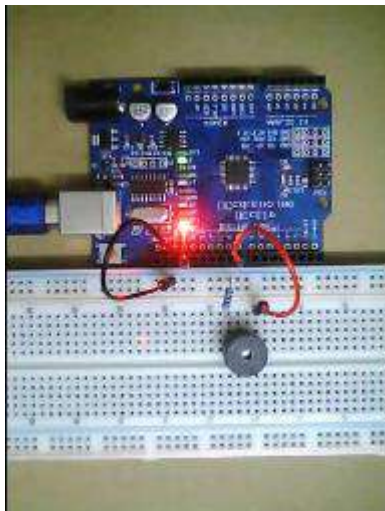
2. In the menu bar of Arduino IDE, we need to select **【Tools】** --- **【Port】** --- selecting the port that the serial number displayed by the device manager just now, as shown in the figure below. For example: COM6, as shown in the following figure.



3. After the selection is completed, you need to click "→" under the menu bar to upload the code to the Arduino UNO board. When the word "Done uploading" appears in the lower left corner, the code has been successfully uploaded to the Arduino UNO board, as shown in the figure below.



4. After the code is uploaded, We can hear the buzzer will sing according to the score written in the program, as shown in the following figure.



## 9- PWM dimming

### The purpose of the experiment:

Arduino controller has six PWM (Pulse Width Modulation) interface, which are digital interface 3, 5, 6, 9, 10, 11. In this course, we input the different analog voltage by adjusting the adjustable resistor, and the microcontroller recognizes the corresponding proportional PWM wave to control the brightness of the LED lights.

### Introduction of PWM:

PWM is used in many places, dimmable lighting, motor speed regulation, sound production and so on.

Its three basic parameters:

1. Pulse width
2. The pulse period (the reciprocal of the number of pulses in one second).
3. Voltage height (for example: 0V-5V)

### List of components required for the experiment:

Arduino UNO board \*1

USB cable \*1

220Ω Resistor \*1

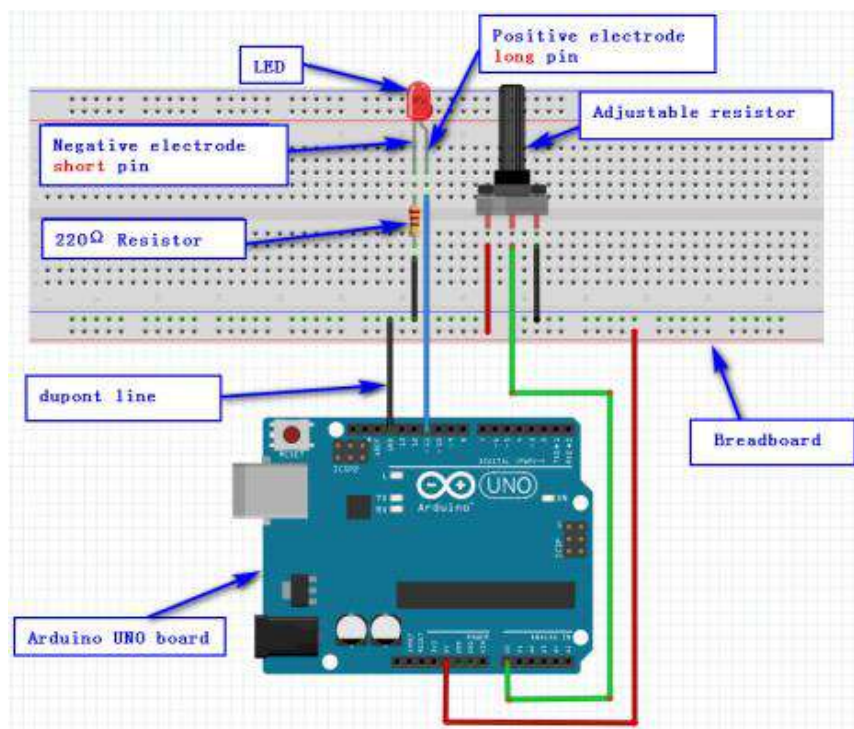
Adjustable resistor \*1

Breadboard \*1

Dupont line \*1bunch

### Actual object connection diagram:

We need to connect the circuit as shown in the figure below.



**Experimental code analysis:**

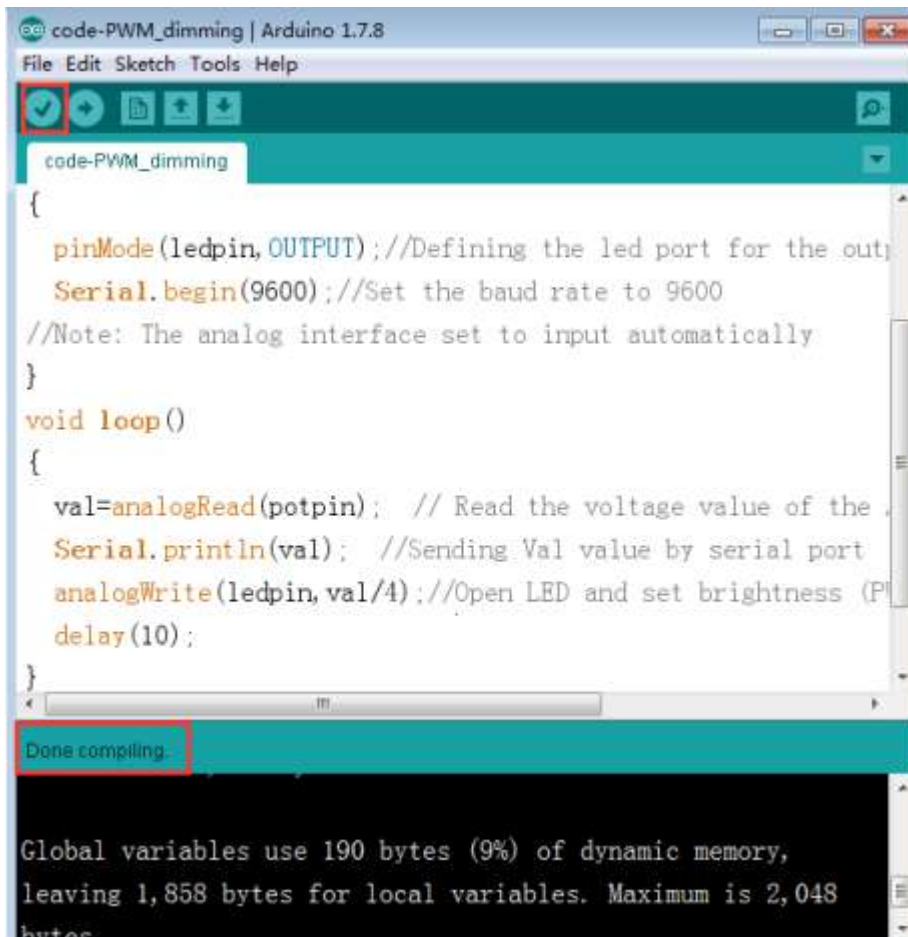
```

int potpin=A0; //Defining the analog port A0
int ledpin=13; //Defining the led port 13
int val=0; //Declarations of temporary variables
void setup()
{
  pinMode(ledpin,OUTPUT); //Defining the light port for the output port
  Serial.begin(9600); //The baud rate is 9600
}
void loop()
{
  val=analogRead(potpin); //Read the voltage value of the A0 port and assign it to val
  Serial.println(val); //Sending Val value by serial port
}

```

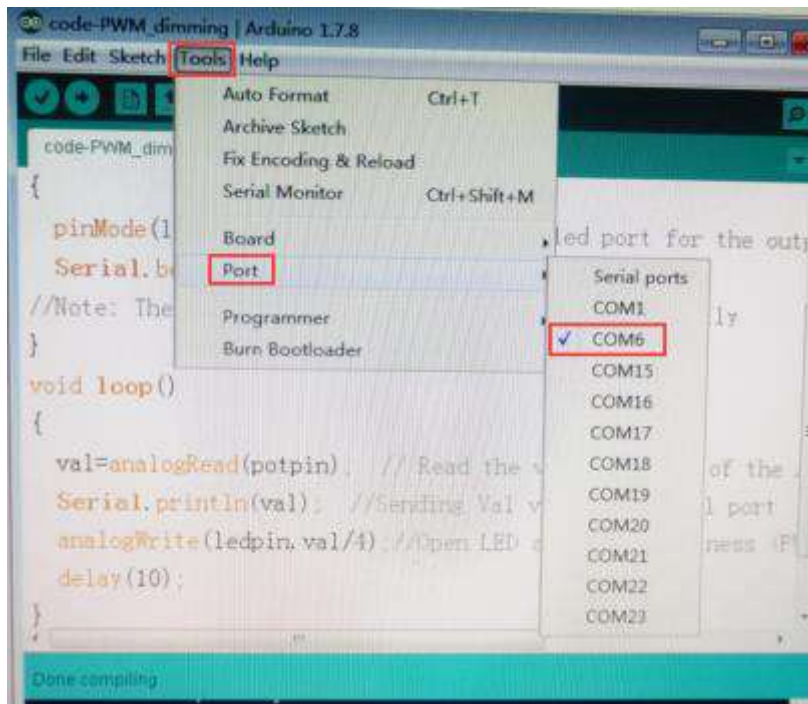
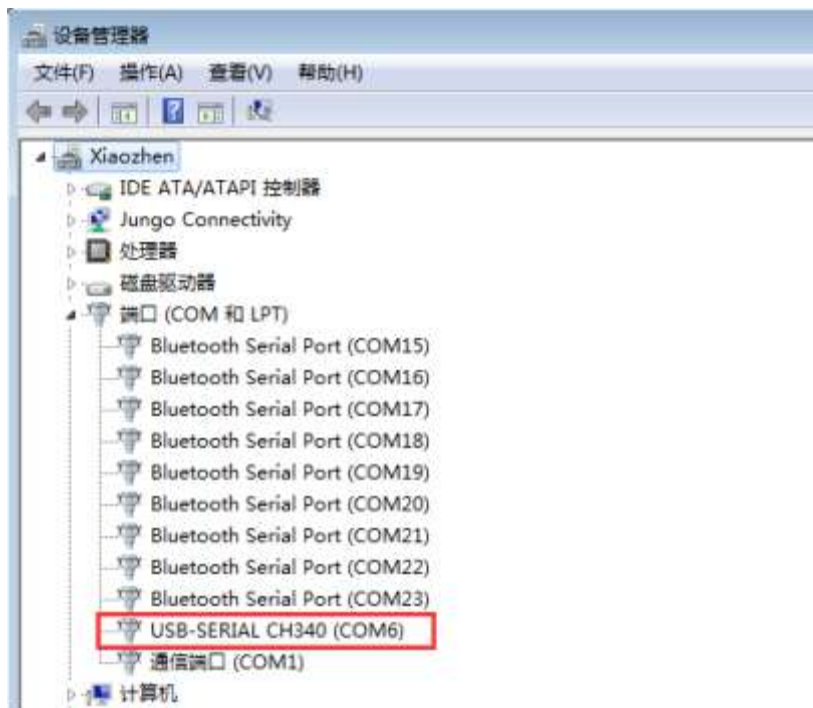
**Experimental steps:**

1.We need to open the program of this experiment: code-PWM-dimming.ino, click“√” under the menu bar to compile the program, and wait for the word "Done compiling " in the lower right corner, as shown in the figure below.



2.In the menu bar of Arduino IDE, we need to select **【Tools】** --- **【Port】** --- selecting the port that the serial number displayed by the device manager just now, as shown in the figure below. For example:COM6,as shown in the following figure.





3. After the selection is completed, you need to click “→” under the menu bar to upload the program to the Arduino UNO board. When the word “Done uploading” appears in the lower left corner, the program has been successfully uploaded to the Arduino UNO board, as shown in the figure below.

The left screenshot shows the Serial Monitor output after the first loop iteration. The value 1023 is highlighted by a red box. The right screenshot shows the Serial Monitor output after the second loop iteration. The values 1016, 1015, 1016, 1017, 1017, 1016, 1015, 1016, 1015, 1017, 1017, 1016, and 1013 are highlighted by a red box.

The image shows two screenshots of the Arduino IDE Serial Monitor window. The window title is "code-Analog\_value | Arduino 1.7.8". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". The Serial Monitor window has a "Send" button and a "Data" button. The output is displayed in a list format. A red box highlights the first 10 values in each screenshot.

**Left Screenshot Output:**

- 1005
- 1010
- 1006
- 1006
- 1006
- 1009
- 1011
- 1006
- 1005
- 1006

**Right Screenshot Output:**

- 363
- 366
- 418
- 405
- 341
- 374
- 432
- 393
- 332
- 379

# 10-Light controlled sound

## The purpose of the experiment:

The purpose of this experiment is to make you learn how to use a special resistor -- photosensitive resistor. This experimental result is: when the photosensitive resistance is connected in the circuit, the buzzer will make a small sound in the absence of light. In the case of illumination, the resistance of the photosensitive resistor will decrease, which causing the voltage across the buzzer to increase and the buzzer sound to become louder.

## List of components required for the experiment:

Arduino UNO board \*1

USB cable \*1

Photosensitive resistor \*1

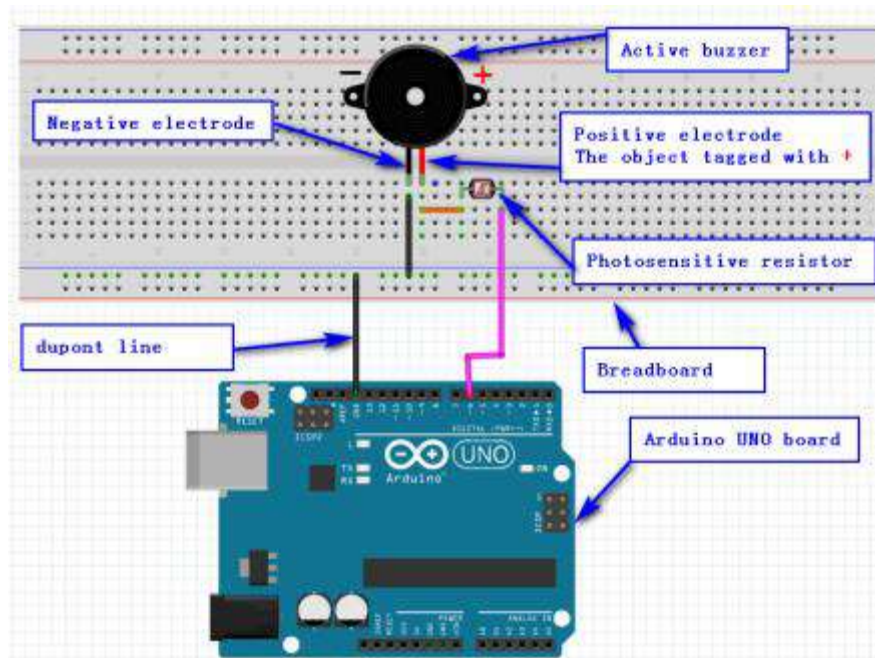
Active buzzer \*1

Breadboard \*1

Dupont line \*1bunch

## Actual object connection diagram:

We need to connect the circuit as shown in the figure below.



## Experimental code analysis:

```
int buzzer=6;//Defining the digital IO port6 to control the buzzer
int i = 0;
void setup()
{
  pinMode(buzzer,OUTPUT);//Defining the led port for the output port
}
void loop()
{
```

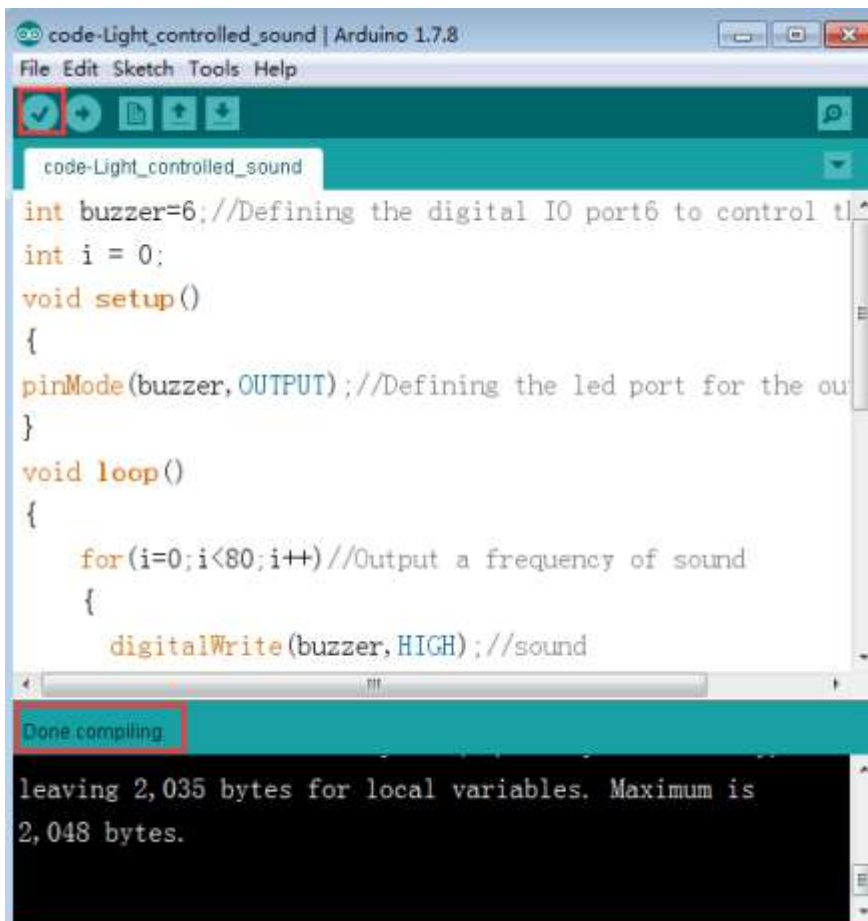
```

for(i=0;i<80;i++)//Output a frequency of sound
{
    digitalWrite(buzzer,HIGH);//sound
    delay(1);
    digitalWrite(buzzer,LOW);//unsound
    delay(1);
}
for(i=0;i<100;i++)//Output another frequency of sound
{
    digitalWrite(buzzer,HIGH);//sound
    delay(2);
    digitalWrite(buzzer,LOW);//unsound
    delay(2);
}
}

```

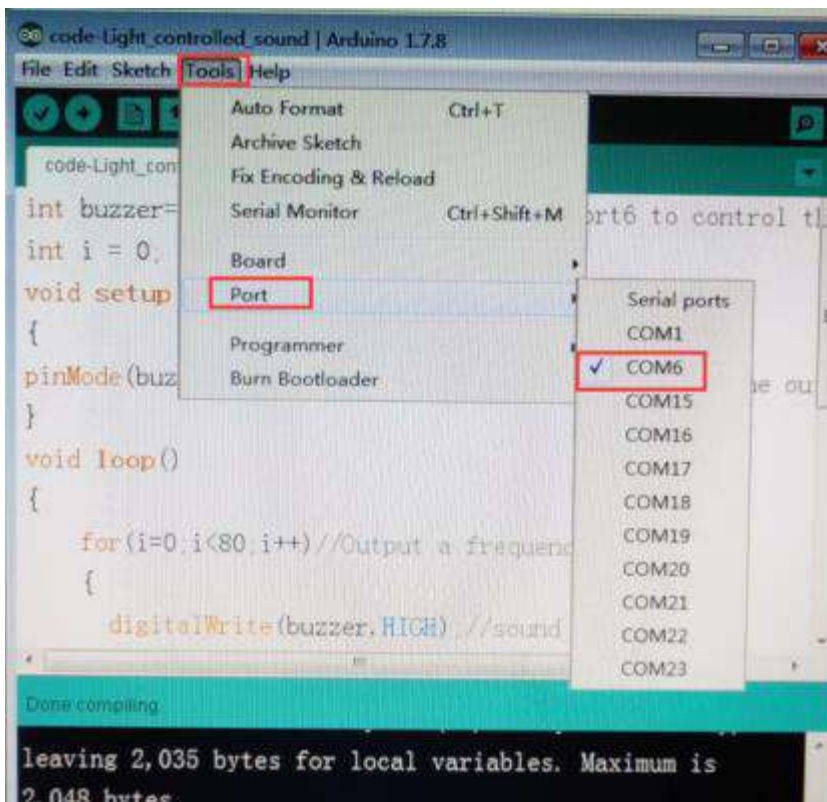
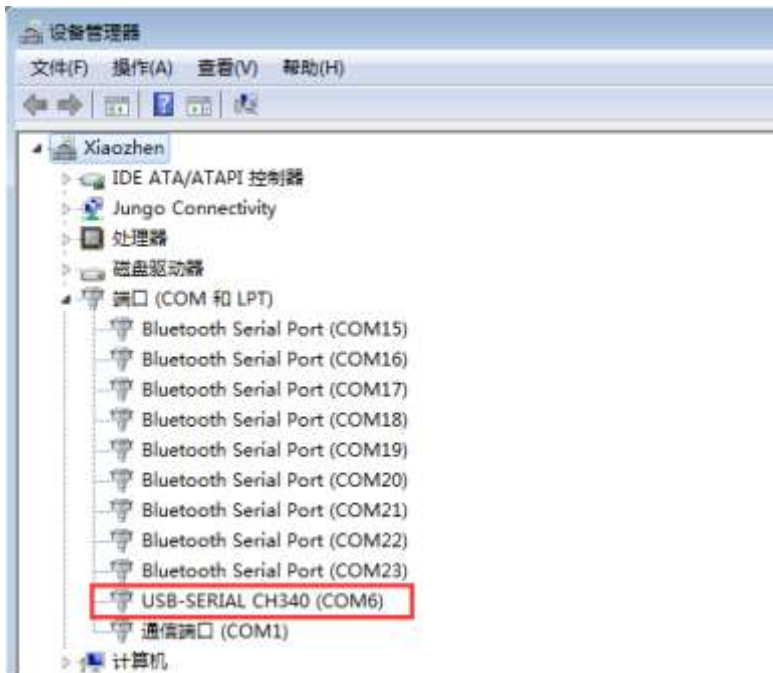
### Experimental steps:

1.We need to open the code of this experiment:code-Light\_controlled\_sound.ino, click“√” under the menu bar to compile the code, and wait for the word "Done compiling " in the lower right corner, as shown in the figure below.



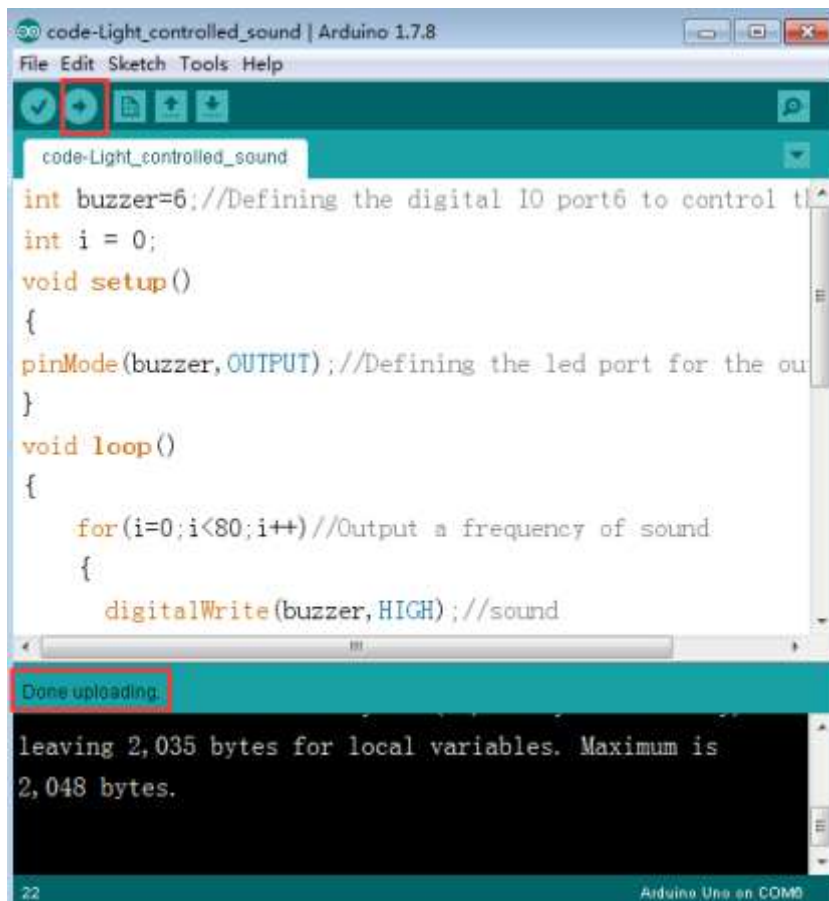
2.In the menu bar of Arduino IDE, we need to select **【Tools】** --- **【Port】** --- selecting the port that the serial number displayed by the device manager just now, as shown in the figure below. For example:COM6,as shown in the following figure.





3. After the selection is completed, you need to click “→” under the menu bar to upload the code to the Arduino UNO board. When the word “Done uploading” appears in the lower left corner, the code has been successfully uploaded to the Arduino UNO board, as shown in the figure below.





The screenshot shows the Arduino IDE interface with the sketch 'code-Light\_controlled\_sound' open. The code defines a buzzer on pin 6 and outputs a square wave in the loop. The 'Upload' button is highlighted with a red box. Below the code editor, a status bar shows 'Done uploading.' and a message box indicates 'leaving 2,035 bytes for local variables. Maximum is 2,048 bytes.' The bottom status bar shows '22' and 'Arduino Uno on COM6'.

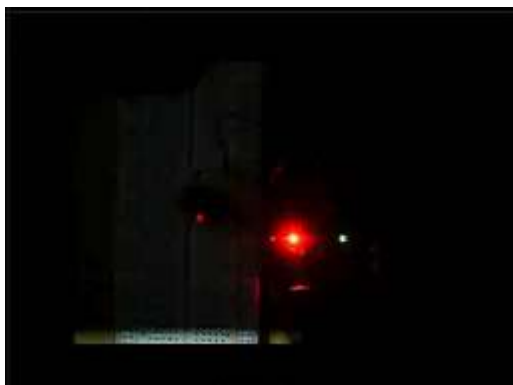
```
code-Light_controlled_sound | Arduino 1.7.8
File Edit Sketch Tools Help
code-Light_controlled_sound
int buzzer=6;//Defining the digital IO port6 to control the buzzer
int i = 0;
void setup()
{
  pinMode(buzzer,OUTPUT);//Defining the led port for the output
}
void loop()
{
  for(i=0;i<80;i++)//Output a frequency of sound
  {
    digitalWrite(buzzer,HIGH);//sound
  }
}
```

Done uploading.

leaving 2,035 bytes for local variables. Maximum is 2,048 bytes.

22 Arduino Uno on COM6

4. We need to give the photosensitive resistor different intensity of illumination, and as the light intensity changes, the size of the sound of the buzzer will also change. As shown in the figure below.



# 11-Sensible heat light

## The purpose of the experiment:

In this course, we use sensor -- thermistor to control PWM by controlling the change of thermistor resistance value, so as to control the brightness of LED light.

Introduction of thermistor:

Thermistor is a kind of sensitive element, which can be divided into positive temperature coefficient thermistor (PTC) and negative temperature coefficient thermistor (NTC) according to different temperature coefficient. The typical characteristic of thermistors is that they are sensitive to temperature and show different resistance values at different temperatures. Resistance value of positive temperature coefficient (PTC) thermistor will become higher when the higher the temperature, Resistance value of negative temperature coefficient (NTC) thermistor will become lower when the higher the temperature. They are both semiconductor devices.

## List of components required for the experiment:

Arduino UNO board \*1

USB cable \*1

Negative temperature coefficient Thermistor\*1

220 $\Omega$  resistor \*1

10k $\Omega$  resistor \*1

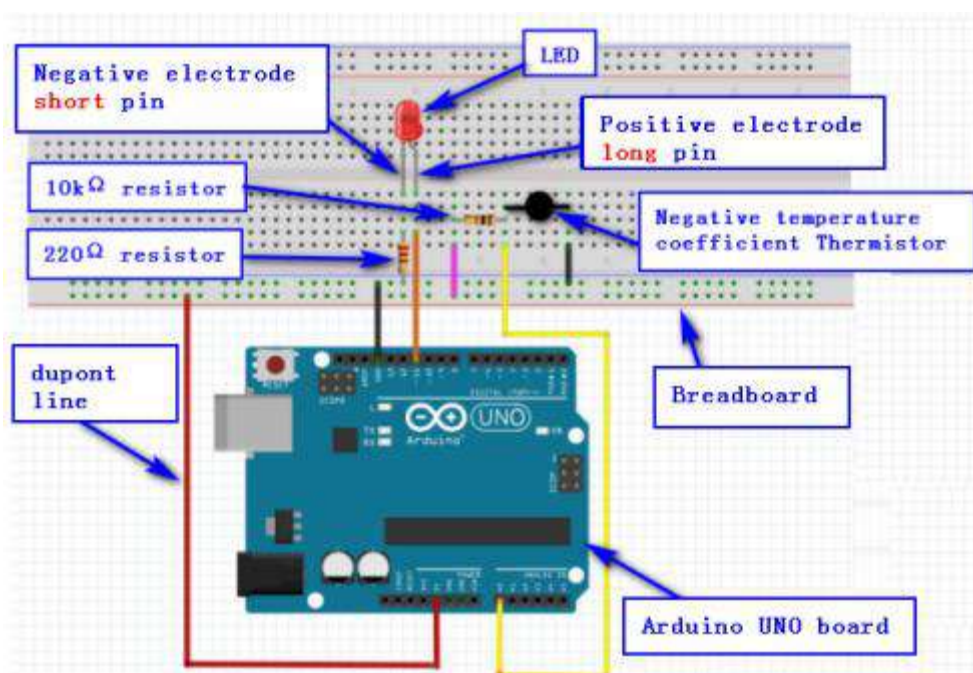
LED \*1

Breadboard \*1

Dupont line \*1 bunch

## Actual object connection diagram :

We need to connect the circuit as shown in the figure below.



**Experimental code analysis:**

```

int potpin=0; //Defining the analog port A0
int ledpin=11; //Defining the digital port 11(for output PWM)
int val=0; //Defining Variable val
void setup()
{
  pinMode(ledpin,OUTPUT); //Defining the port11 for the output port
  Serial.begin(9600); //The baud rate is 9600
}
void loop()
{
  val=analogRead(potpin); //Reading the voltage value of the A0 port and assign it to val
  val = 245- val;
  if(val < 0)
    val = 0;
  Serial.println(val);
  analogWrite(ledpin,val); //PWM output is used to drive LED
  delay(100);
}

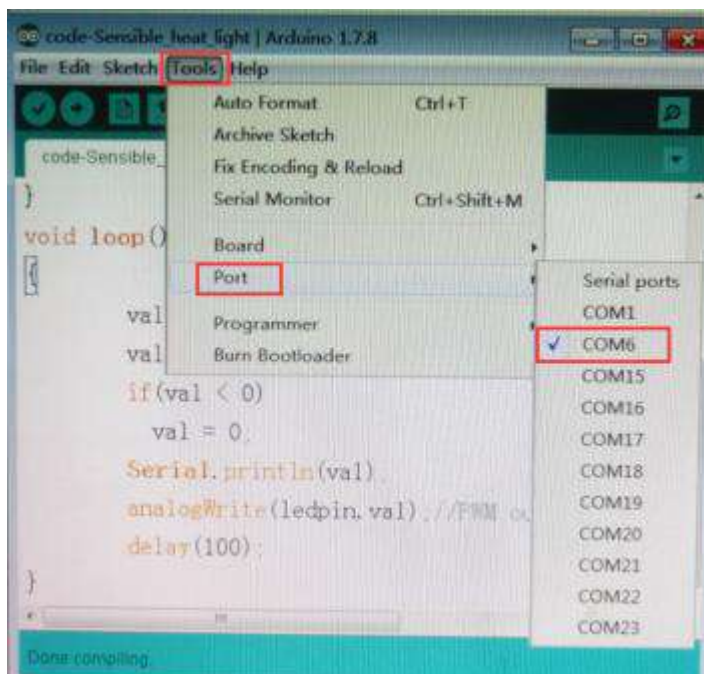
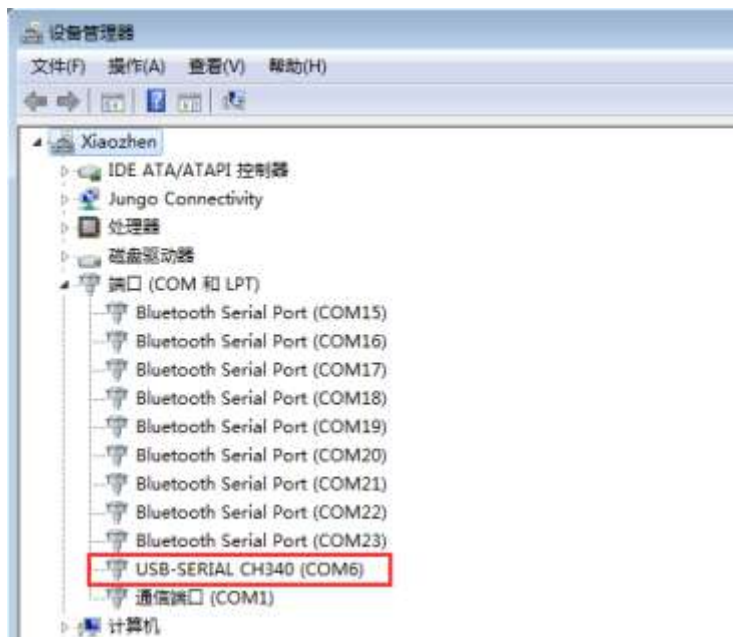
```

**Experimental steps:**

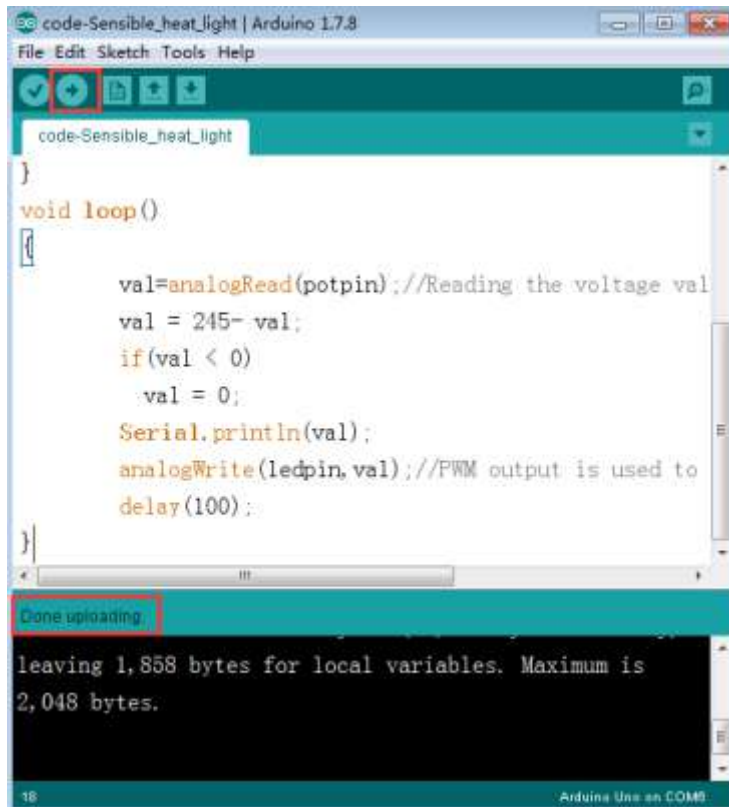
1.We need to open the code of this experiment: code-Sensible\_heat\_light.ino, click“√” under the menu bar to compile the code, and wait for the word "Done compiling " in the lower right corner, as shown in the figure below.



2. In the menu bar of Arduino IDE, we need to select **【Tools】** --- **【Port】** --- selecting the port that the serial number displayed by the device manager just now, as shown in the figure below. For example:COM6,as shown in the following figure.



3. After the selection is completed, you need to click “→” under the menu bar to upload the code to the Arduino UNO board. When the word “Done uploading” appears in the lower left corner, the code has been successfully uploaded to the Arduino UNO board, as shown in the figure below.



```

code-Sensible_heat_light | Arduino 1.7.8
File Edit Sketch Tools Help

code-Sensible_heat_light

}
void loop()

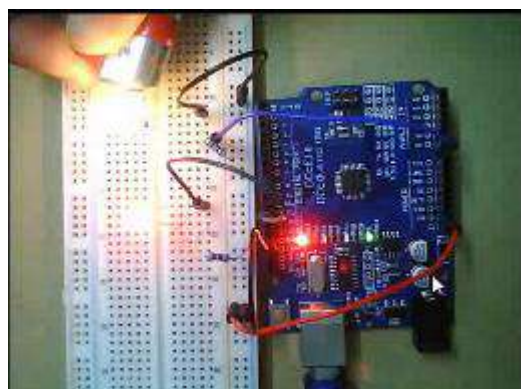
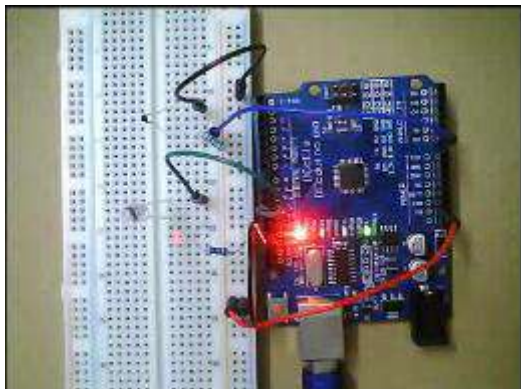
    val=analogRead(potpin); //Reading the voltage val
    val = 245- val;
    if(val < 0)
        val = 0;
    Serial.println(val);
    analogWrite(ledpin, val); //PWM output is used to
    delay(100);
}

Done uploading
leaving 1,858 bytes for local variables. Maximum is
2,048 bytes.

18 Arduino Uno as COM5

```

4. After the code is uploaded. When we do not heat the thermistor, the LED extinguish. When we heat the thermistor, the LED will bright, and the brightness of the LED will change with the change of the heat of the thermistor. At the same time, we can open the serial port monitor, and we can also see the change of the voltage value at both ends of the LED, as shown in the following figure.





## 12-8x8 dot matrix

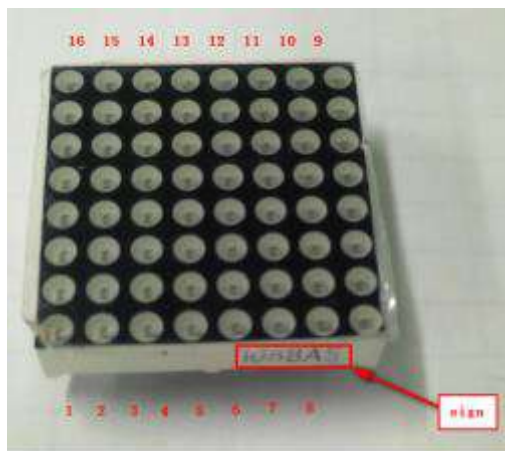
### The purpose of the experiment:

In this course, we will learn how to use 8x8 dot matrix. The experimental effect is to light the LED on the 8x8 dot matrix.

### Introduction of 8x8 dot matrix:

The 8x8 lattice is composed of 64 LED, and each LED is placed at the intersection of line and line. When one line is high level(1) and a column is low level(0), the corresponding diode will be bright. If you want to light up the first line, the ninth pin need to high level, and (13, 3, 4, 10, 6, 11, 15, 16) these pins are low level. If you want to light up the first column, the thirteenth pin need low level, and (9, 14, 8, 12, 1, 7, 2, 5) these pins are low level.

Pin identification as shown in the two figures below.



### List of components required for the experiment:

Arduino UNO board \*1

USB cable \*1

220Ω resistor \*8

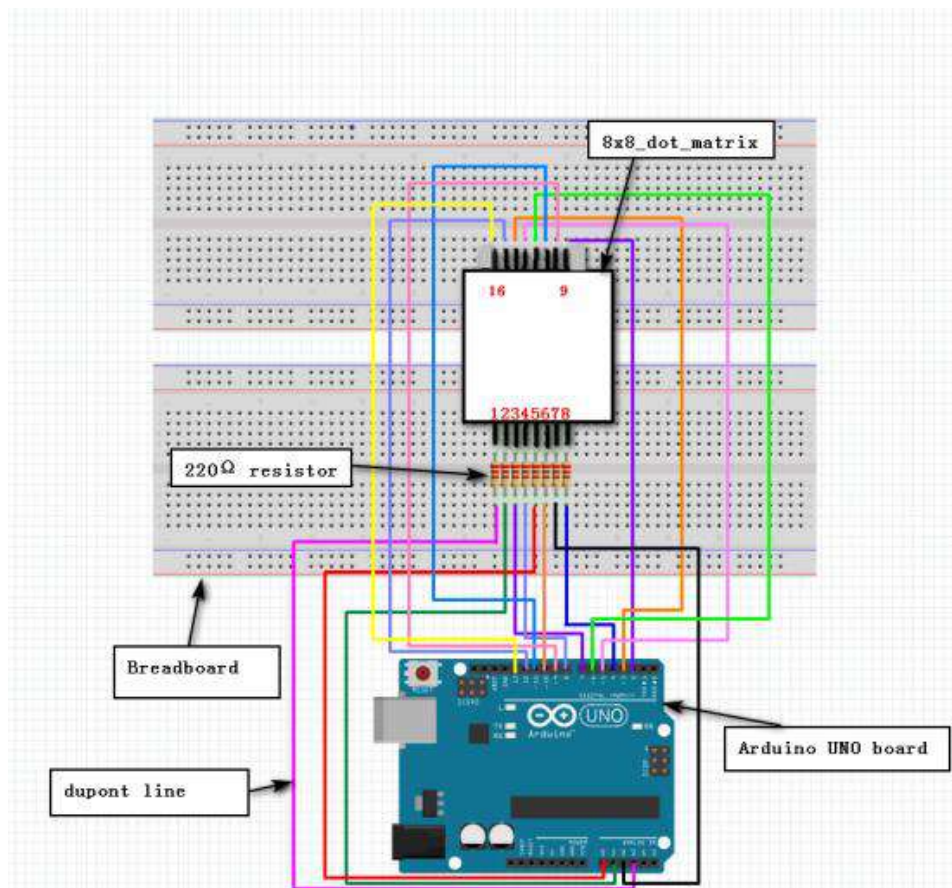
8x8 dot matrixLED\*1

Breadboard \*1

dupont line \*1bunch

### Actual object connection diagram :

We need to connect the circuit as shown in the figure below.



### Experimental code analysis:

```

const int row1 = 2; // Arduino Pin2 connect pin9 of the dot matrix
const int row2 = 3; // Arduino Pin3 connect pin14 of the dot matrix
const int row3 = 4; // Arduino Pin4 connect pin8 of the dot matrix
const int row4 = 5; // Arduino Pin5 connect pin12 of the dot matrix
const int row5 = 17; // Arduino Pin17 (A3)connect pin1 of the dot matrix
const int row6 = 16; // Arduino Pin16 (A2)connect pin7 of the dot matrix
const int row7 = 15; // Arduino Pin15 (A1)connect pin2 of the dot matrix
const int row8 = 14; // Arduino Pin14 (A0)connect pin5 of the dot matrix
//the pin to control COI
const int col1 = 6; //Arduino Pin6 connect pin13 of the dot matrix
const int col2 = 7; // Arduino Pin7 connect pin3 of the dot matrix
const int col3 = 8; //Arduino Pin8 connect pin4 of the dot matrix
const int col4 = 9; // Arduino Pin9 connect pin10 of the dot matrix
const int col5 = 10; //Arduino Pin10 connect pin6 of the dot matrix
const int col6 = 11; //Arduino Pin11 connect pin11 of the dot matrix
const int col7 = 12; // Arduino Pin12 connect pin12 of the dot matrix
const int col8 = 13; // Arduino Pin13 connect pin13 of the dot matrix
void setup()
{
  int i = 0 ;
  for(i=2;i<18;i++)
  {
    pinMode(i, OUTPUT);
  }
  for(i=2;i<18;i++) {
    digitalWrite(i, LOW);
  }
}

```

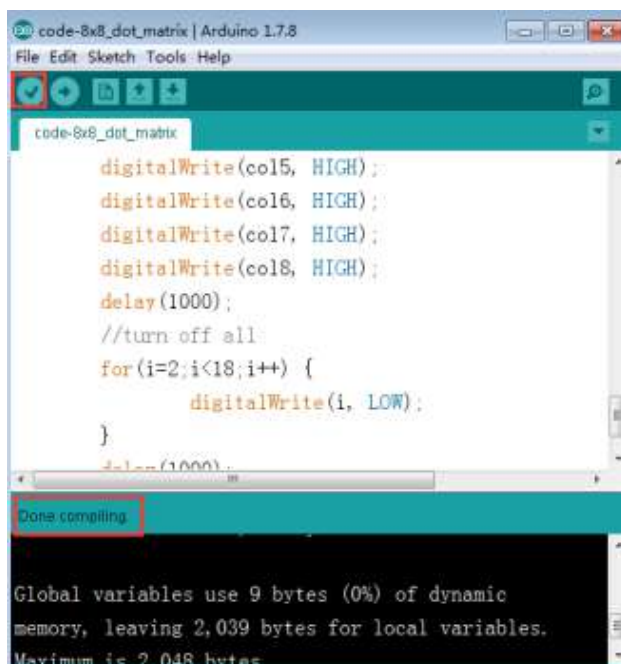
```

}
}
void loop()
{
  int i;
  //the row # 1 and col # 1 of the LEDs turn on
  digitalWrite(row1, HIGH);
  digitalWrite(row2, LOW);
  digitalWrite(row3, LOW);
  digitalWrite(row4, LOW);
  digitalWrite(row5, LOW);
  digitalWrite(row6, LOW);
  digitalWrite(row7, LOW);
  digitalWrite(row8, LOW);
  digitalWrite(col1, LOW);
  digitalWrite(col2, HIGH);
  digitalWrite(col3, HIGH);
  digitalWrite(col4, HIGH);
  digitalWrite(col5, HIGH);
  digitalWrite(col6, HIGH);
  digitalWrite(col7, HIGH);
  digitalWrite(col8, HIGH);
  delay(1000);
  //turn off all
  for(i=2;i<18;i++) {
    digitalWrite(i, LOW);
  }
  delay(1000);
}

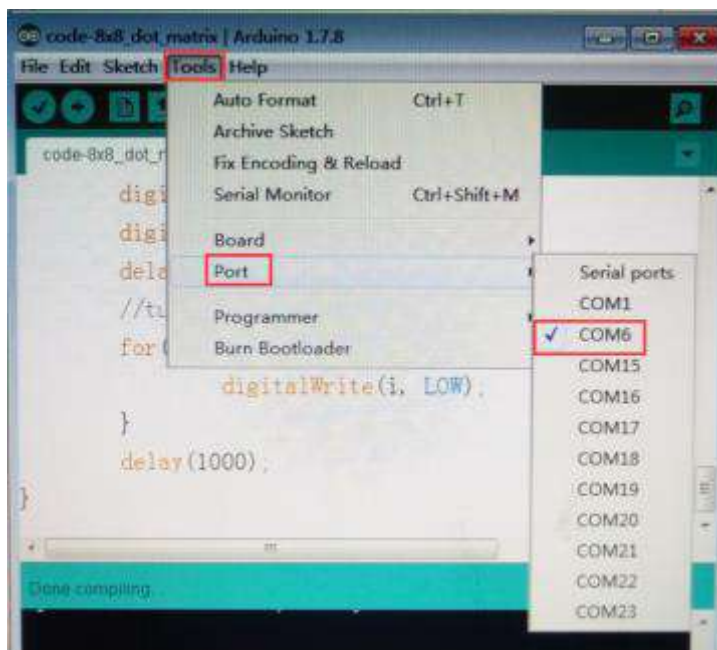
```

### Experimental steps:

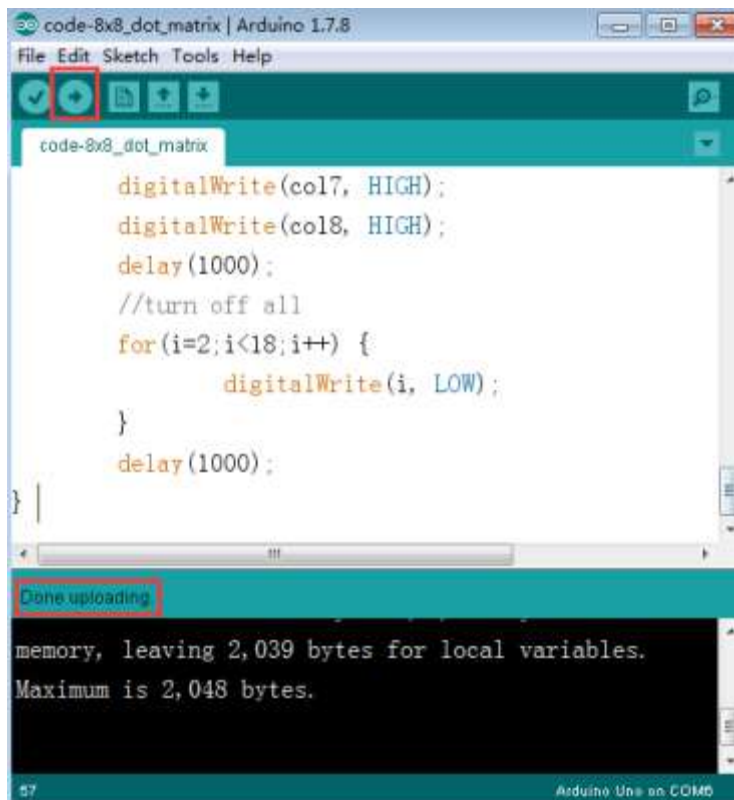
1. We need to open the code of this experiment: code-8x8\_dot\_matrix.ino, click "✓" under the menu bar to compile the code, and wait for the word "Done compiling" in the lower right corner, as shown in the figure below.



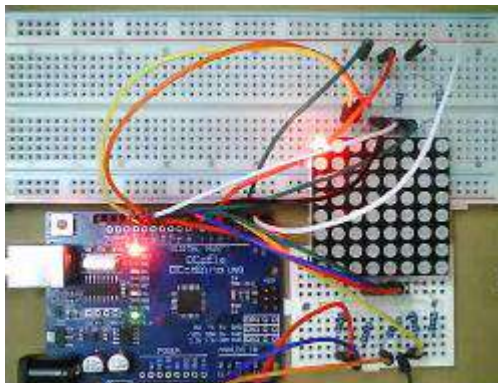
2. In the menu bar of Arduino IDE, we need to select **【Tools】** --- **【Port】** --- selecting the port that the serial number displayed by the device manager just now, as shown in the figure below. For example:COM6,as shown in the following figure.



3. After the selection is completed, you need to click “→” under the menu bar to upload the code to the Arduino UNO board. When the word “Done uploading” appears in the lower left corner, the code has been successfully uploaded to the Arduino UNO board, as shown in the figure below.



4. After the code is uploaded. We can see that the lights in the first row and first column of the dot matrix are twinkling, as shown in the following figure.





## 13-Tilt switch

### The purpose of the experiment:

This lesson is ball switch experiment, it also belongs to the tilt switch just name is different. It control the turning on or off of the circuit by the rolling contact pin of the beads in the switch, so the LED can be switched on and off.

### List of components required for the experiment:

Arduino UNO board \*1

USB cable \*1

220 $\Omega$  resistor \*1

10k $\Omega$  resistor \*1

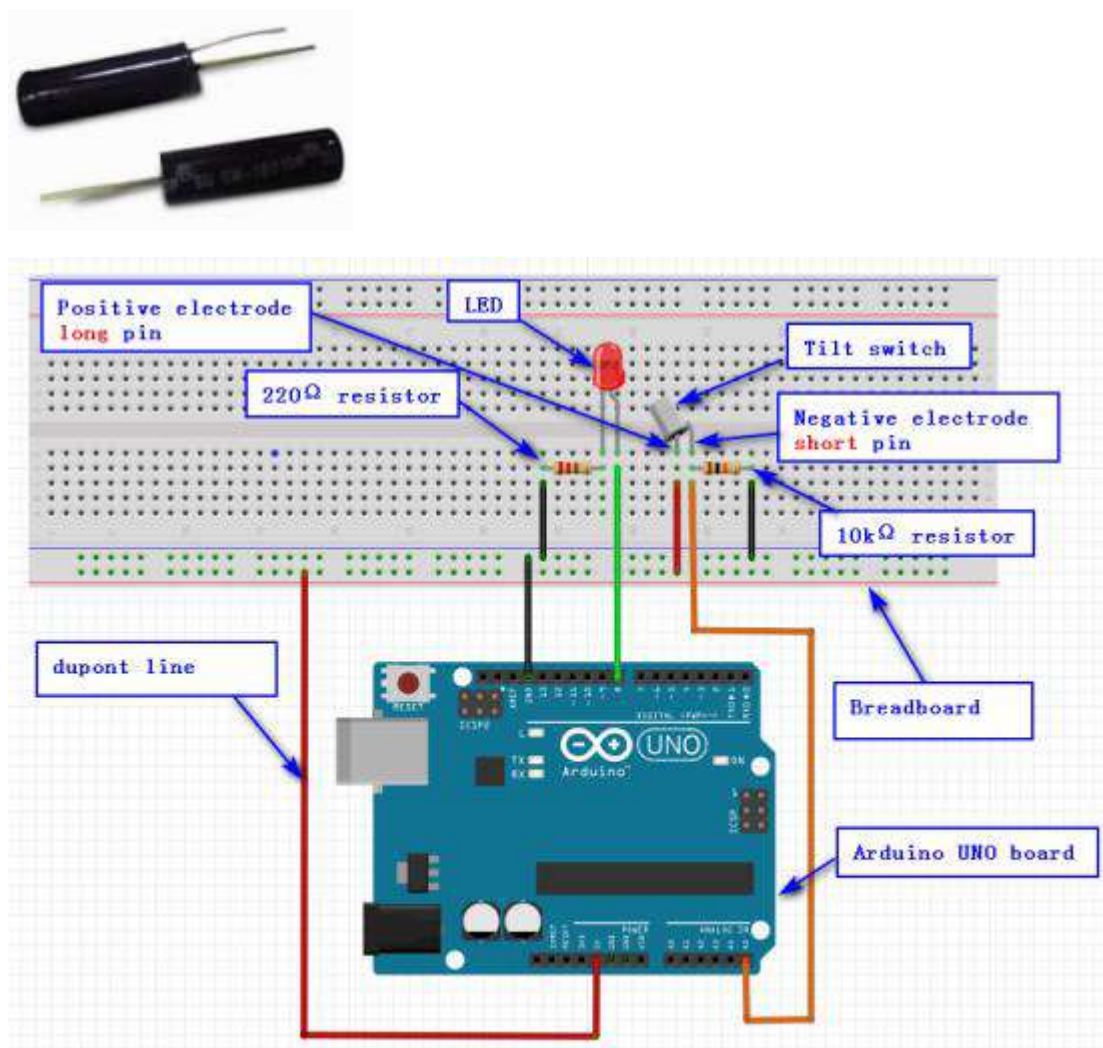
Tilt switch \*1

Breadboard \*1

Dupont line \*1 bunch

### Actual object connection diagram :

We need to connect the circuit as shown in the figure below.



### Experimental code analysis:

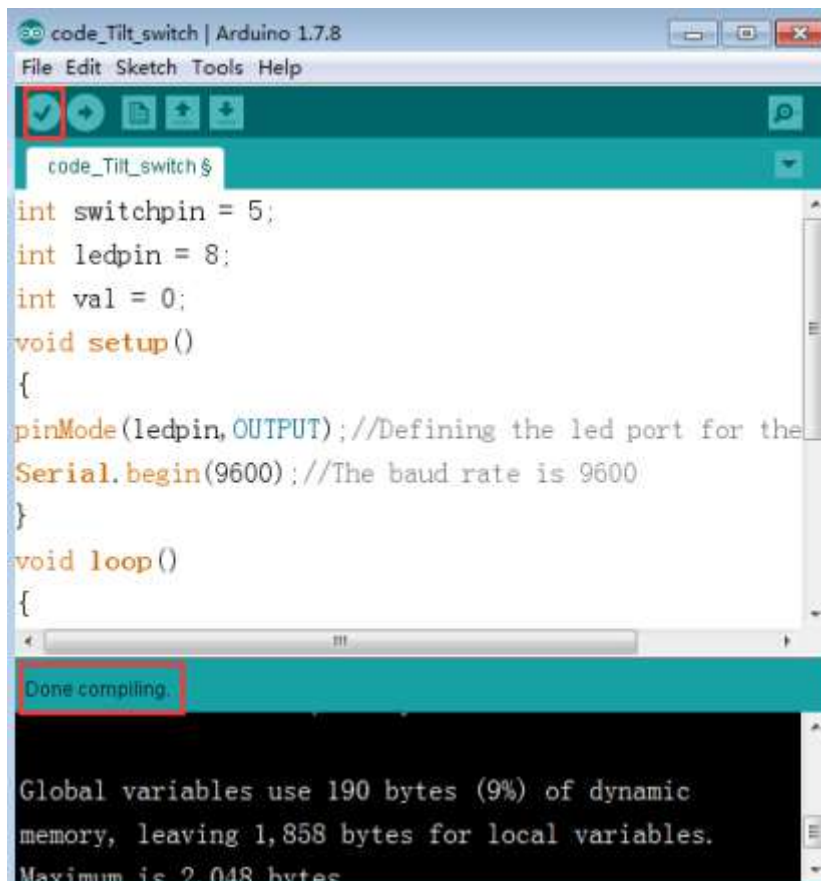
```

int switchpin = 5;
int ledpin = 8;
int val = 0;
void setup()
{
  pinMode(ledpin,OUTPUT);//Defining the led port for the output port
  Serial.begin(9600);//The baud rate is 9600
}
void loop()
{
  val = analogRead(switchpin);
  if(val>512)//The analog voltage value of 512 is exactly 2.5V
  digitalWrite(ledpin,HIGH);//If val Greater than 2.5 V
  else//If val less than or equal to 2.5 V
  digitalWrite(ledpin,LOW);
  Serial.println(val);
}

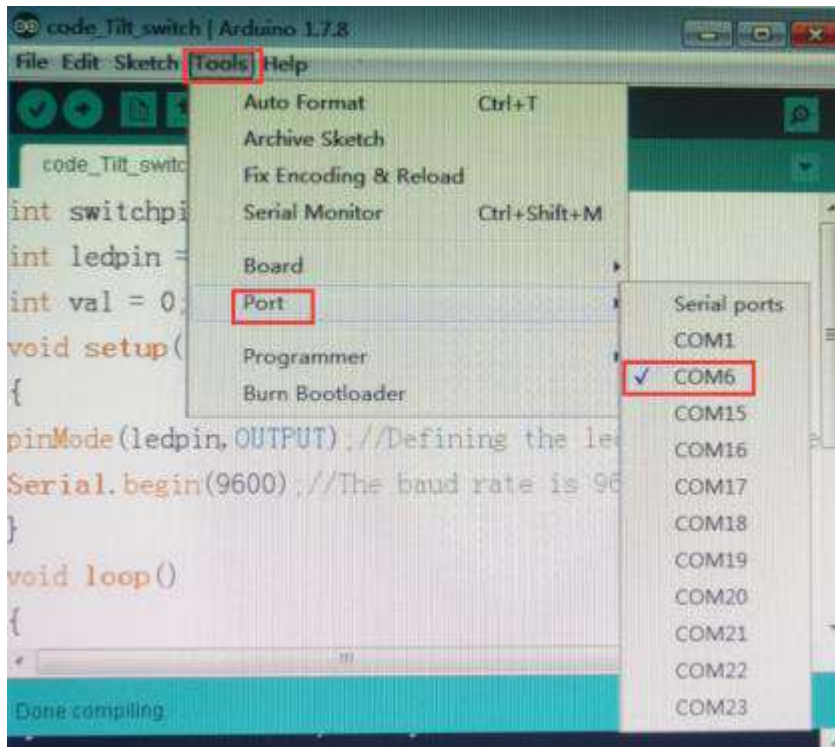
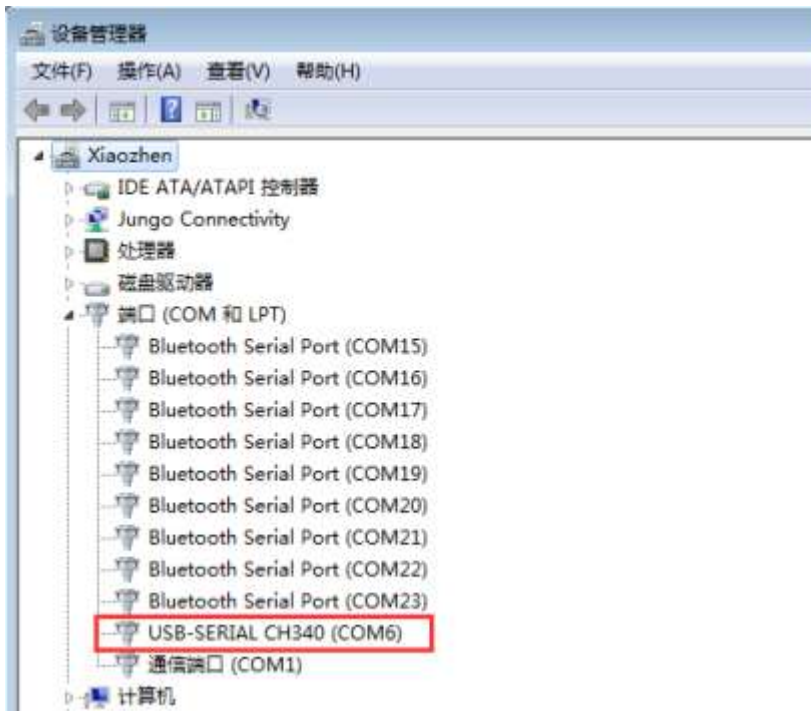
```

### Experimental steps:

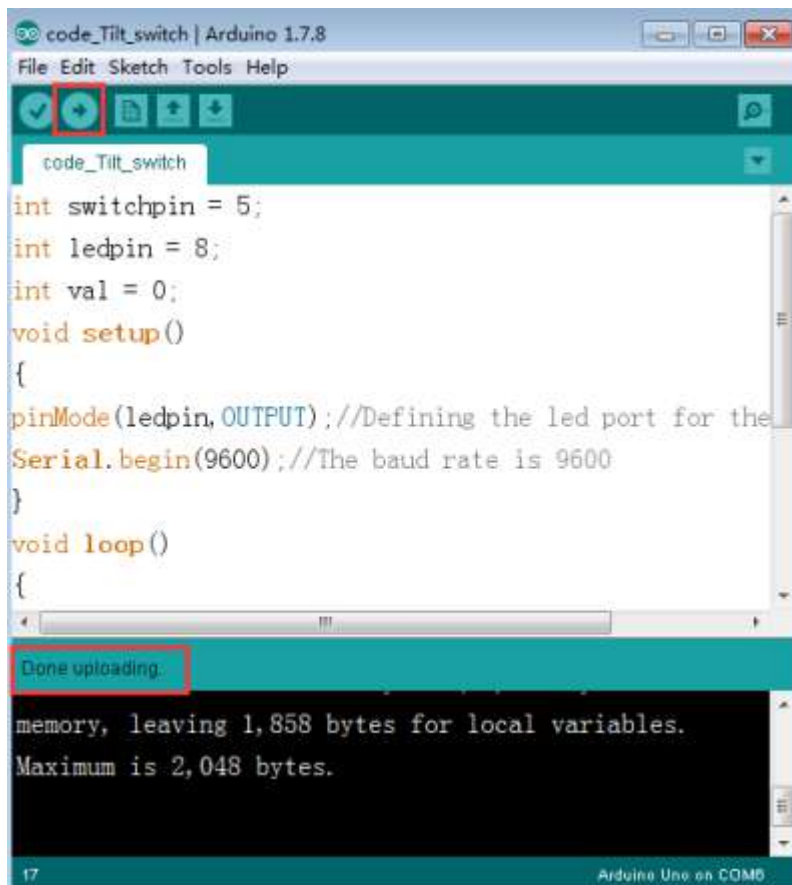
1. We need to open the code of this experiment: code-8x8\_dot\_matrix.ino, click "✓" under the menu bar to compile the code, and wait for the word "Done compiling" in the lower right corner, as shown in the figure below.



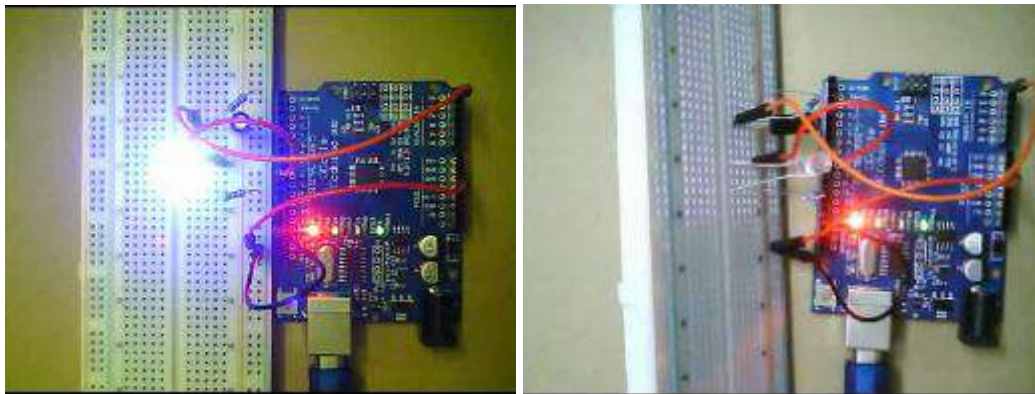
2. In the menu bar of Arduino IDE, we need to select **【Tools】** --- **【Port】** --- selecting the port that the serial number displayed by the device manager just now, as shown in the figure below. For example: COM6, as shown in the following figure.



3. After the selection is completed, you need to click “→” under the menu bar to upload the code to the Arduino UNO board. When the word “Done uploading” appears in the lower left corner, the code has been successfully uploaded to the Arduino UNO board, as shown in the figure below.



4. After the code is uploaded. The LED lights up when the ball switch is in the horizontal position, and the LED turns off when we put the ball switch in the tilt position., At the same time, we can open the serial port monitor, we can also see the change of the voltage value at both ends of the ball switch, as shown in the figure below.





## 14-Flame alarm

### The purpose of the experiment:

In this lesson, we need to complete the experiment of fire alarm. The experimental effect is: when there is no fire source approaching, the circuit is normal. When there is a fire source approaching, the buzzer will make a sound.

### Introduction of flame sensor:

The actual object is shown below. Flame sensor (Infrared receiving triode), Because infrared is very sensitive to flame, we use a special infrared receiver tube to detect the flame, and then convert the brightness of the flame into a level signal of high and low change, and we need to input these signals into the MCU. Finally the MCU makes corresponding program processing according to the change of the signals.



### List of components required for the experiment:

Arduino UNO board \*1

USB cable \*1

220Ω resistor \*1

10kΩ resistor \*1

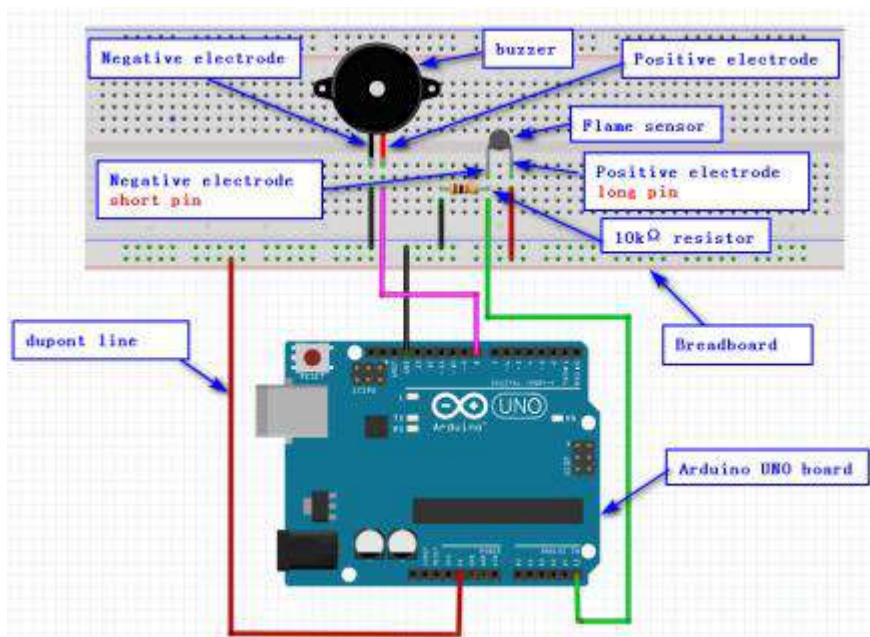
Tilt switch \*1

Breadboard \*1

Dupont line \*1 bunch

### Actual object connection diagram :

We need to connect the circuit as shown in the figure below.





**Experimental code analysis:**

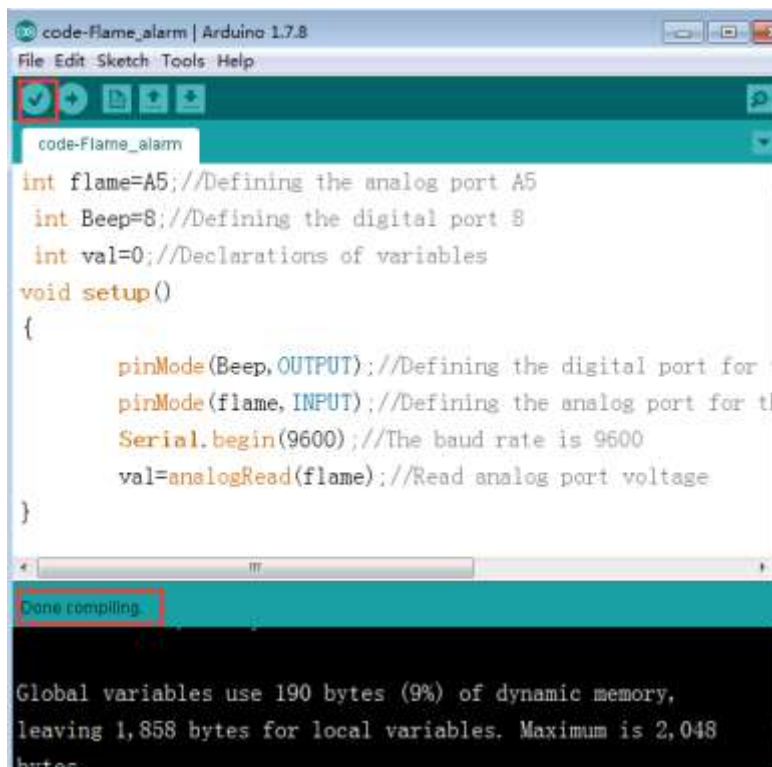
```

int flame=A5; //Defining the analog port A5
int Beep=8; //Defining the digital port 8
int val=0; //Declarations of variables
void setup()
{
  pinMode(Beep,OUTPUT); //Defining the digital port for the output port
  pinMode(flame,INPUT); //Defining the analog port for the input port
  Serial.begin(9600); //The baud rate is 9600
  val=analogRead(flame); //Read analog port voltage
}
void loop()
{
  Serial.println(analogRead(flame)); //The serial port sends the simulated voltage value
  if((analogRead(flame)-val)>=600) //Determine whether the simulated voltage value is
    greater than 600
    digitalWrite(Beep,HIGH);
  else
    digitalWrite(Beep,LOW);
}

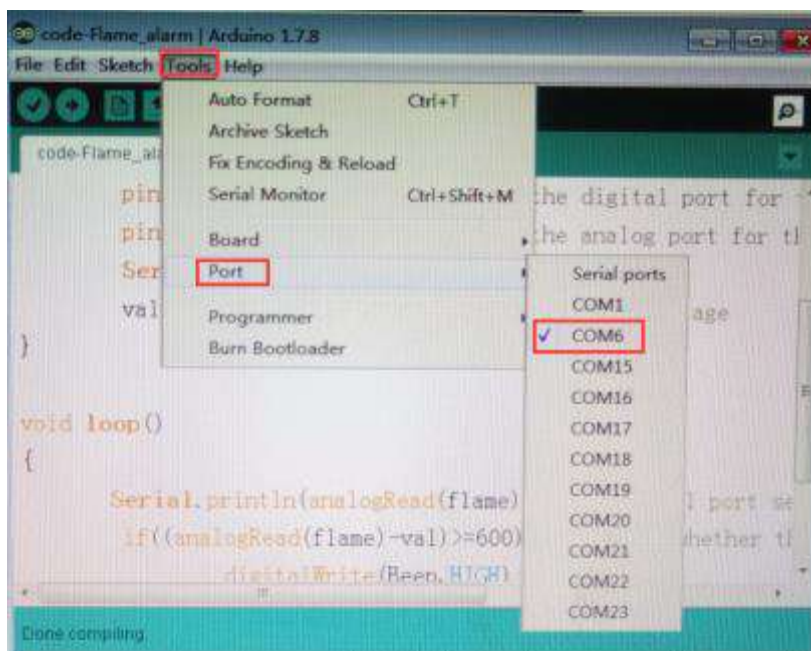
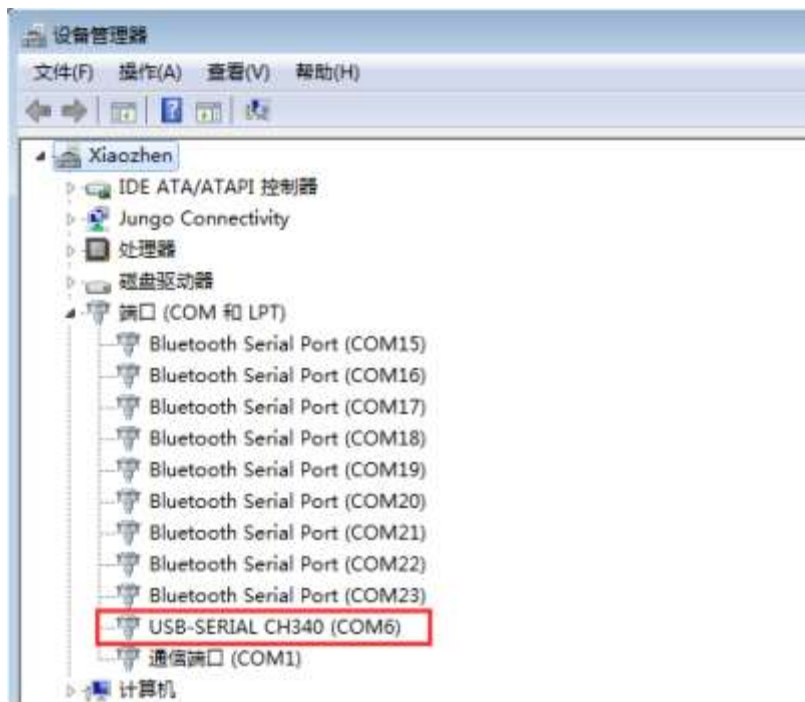
```

**Experimental steps:**

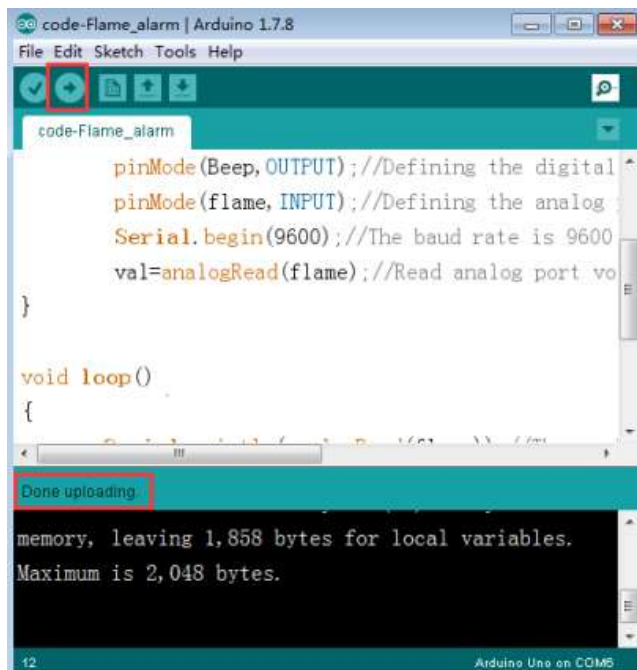
1. We need to open the code of this experiment: code-Tilt\_switch.ino, click "✓" under the menu bar to compile the code, and wait for the word "Done compiling" in the lower right corner, as shown in the figure below.



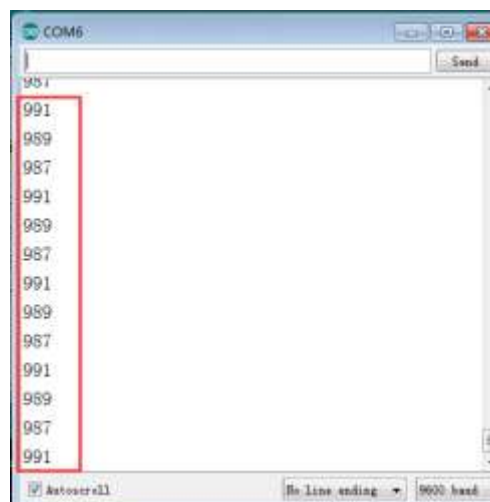
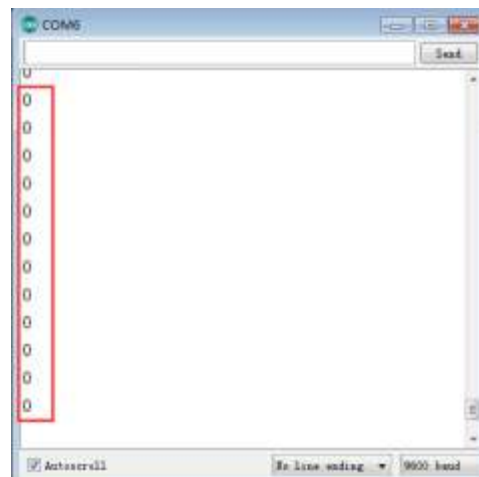
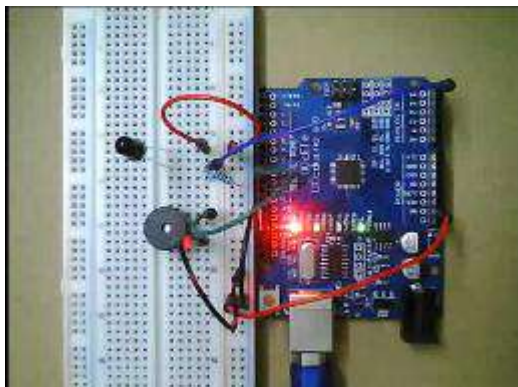
2. In the menu bar of Arduino IDE, we need to select **【Tools】** --- **【Port】** --- selecting the port that the serial number displayed by the device manager just now, as shown in the figure below. For example: COM6, as shown in the following figure.



3. After the selection is completed, you need to click “→” under the menu bar to upload the code to the Arduino UNO board. When the word “Done uploading” appears in the lower left corner, the code has been successfully uploaded to the Arduino UNO board, as shown in the figure below.



4. After the code is uploaded. When there is no fire source approaching, the circuit is normal. When there is a fire source approaching, the buzzer will make a sound to indicate the alarm. We can also open the serial monitor to observe the change in the value of the flame sensor, as shown in the figure below.



## 15-Nixie tube

### The purpose of the experiment:

In this experiment, we need to finish to display 1-9 on a single 8-segment Nixie tube.

### Introduction to digital tube:

Nixie tube is a semiconductor luminescent device, its basic unit is a light-emitting diode. It is divided into 7-segment Nixie tube and 8-segment Nixie tube. 8-segment Nixie tube more than 7-segment Nixie tube a light-emitting diode unit (more than a decimal point), this experiment we use the 8-segment Nixie tube. The actual object is shown below.



According to the light-emitting diode unit connection mode, it is divided into anode Nixie tubes and cathode Nixie tubes.

Anode Nixie tubes that connects the anodes of all light-emitting diodes together to form a common anode (COM). The common pole (COM) shall be connected to +5V when the common anode digital tube is applied. When the cathode of a certain field of light-emitting diode is low, the corresponding field will be light up. When the cathode of a field is high, the field does not light up.

Cathode Nixie tubes that connects the cathodes of all light-emitting diodes together to form a common cathode (COM). The common pole COM shall be connected to GND when the common cathode digital tube is applied. When the anode of a certain field of light-emitting diode is high, the corresponding field will be light up. When the anode of a field is low, the field does not light up.

### List of components required for the experiment:

Arduino UNO board \*1

USB cable \*1

220Ω resistor \*8

8-segment digital tube \*1

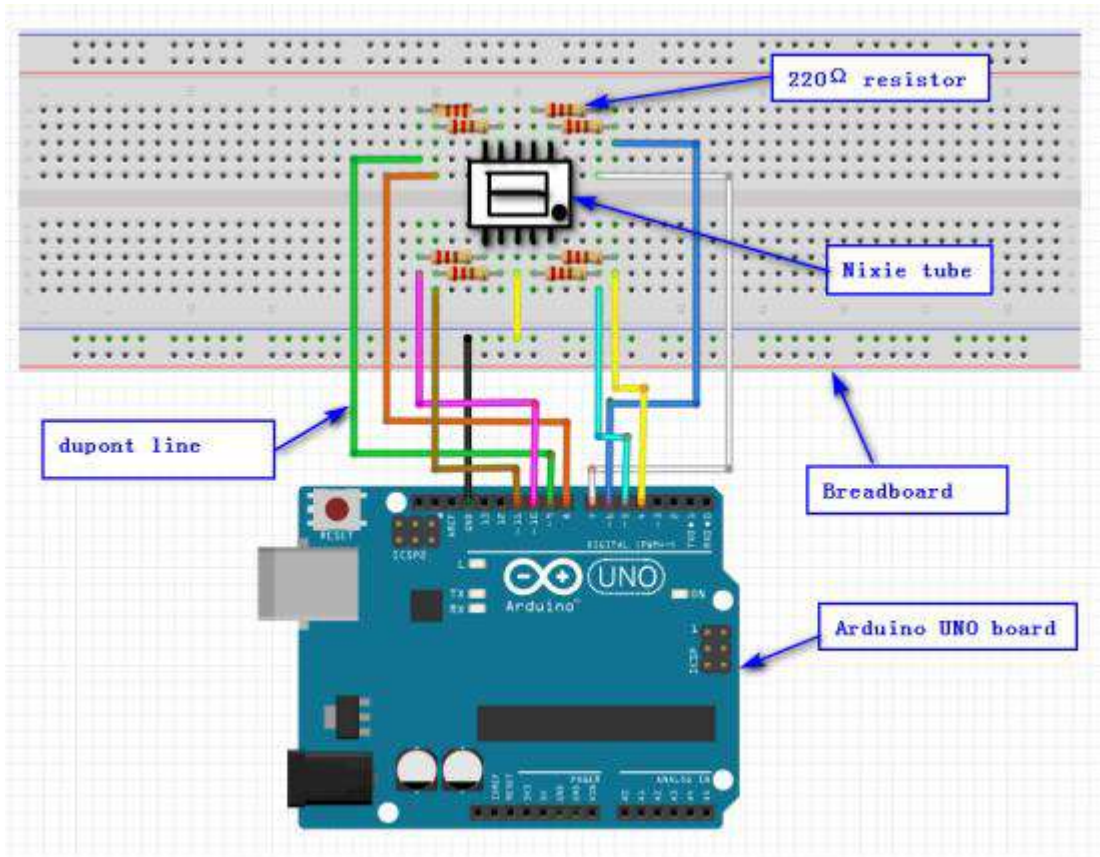
Breadboard \*1

Dupont line \*1bunch

### Actual object connection diagram :

We need to connect the circuit as shown in the figure below.





### Experimental code analysis:

```

int a=7; // Digital port 7 is connected to digital tube section a
int b=6; // Digital port 6 is connected to digital tube section b
int c=5; // Digital port 5 is connected to digital tube section c
int d=11; // Digital port 11 is connected to digital tube section d
int e=10; //Digital port 10 is connected to digital tube section e
int f=8; //Digital port 8 is connected to digital tube section f
int g=9; //Digital port 9 is connected to digital tube section g
int dp=4; //Digital port 4 is connected to digital tube decimal point section
void digital_1(void) //Displaying 1
{
  unsigned char j;
  digitalWrite(c,HIGH); //Light digital tube section c
  digitalWrite(b,HIGH); //Light digital tube section b
  for(j=7;j<=11;j++) //The level is pulled low of tube section 7~11(a,f,g,e,d)
    digitalWrite(j,LOW);
  digitalWrite(dp,LOW); //Tube decimal point section is off
}
void digital_2(void) //Displaying 1
{
  unsigned char j;
  digitalWrite(b,HIGH);
  digitalWrite(a,HIGH);
  for(j=9;j<=11;j++)
    digitalWrite(j,HIGH);
  digitalWrite(dp,LOW);
  digitalWrite(c,LOW);
  digitalWrite(f,LOW);
}

```



```

}
void digital_3(void) //Displaying 3
{
  unsigned char j;
  digitalWrite(g,HIGH);
  digitalWrite(d,HIGH);
  for(j=5;j<=7;j++)
    digitalWrite(j,HIGH);
  digitalWrite(dp,LOW);
  digitalWrite(f,LOW);
  digitalWrite(e,LOW);
}
void digital_4(void) //Displaying 4
{
  digitalWrite(c,HIGH);
  digitalWrite(b,HIGH);
  digitalWrite(f,HIGH);
  digitalWrite(g,HIGH);
  digitalWrite(dp,LOW);
  digitalWrite(a,LOW);
  digitalWrite(e,LOW);
  digitalWrite(d,LOW);
}
void digital_5(void) //Displaying 5
{
  unsigned char j;
  for(j=7;j<=9;j++)
    digitalWrite(j,HIGH);
  digitalWrite(c,HIGH);
  digitalWrite(d,HIGH);
  digitalWrite(dp,LOW);
  digitalWrite(b,LOW);
  digitalWrite(e,LOW);
}
void digital_6(void) //Displaying 6
{
  unsigned char j;
  for(j=7;j<=11;j++)
    digitalWrite(j,HIGH);
  digitalWrite(c,HIGH);
  digitalWrite(dp,LOW);
  digitalWrite(b,LOW);
}
void digital_7(void) //Displaying 7
{
  unsigned char j;
  for(j=5;j<=7;j++)
    digitalWrite(j,HIGH);
  digitalWrite(dp,LOW);
  for(j=8;j<=11;j++)
    digitalWrite(j,LOW);
}

```

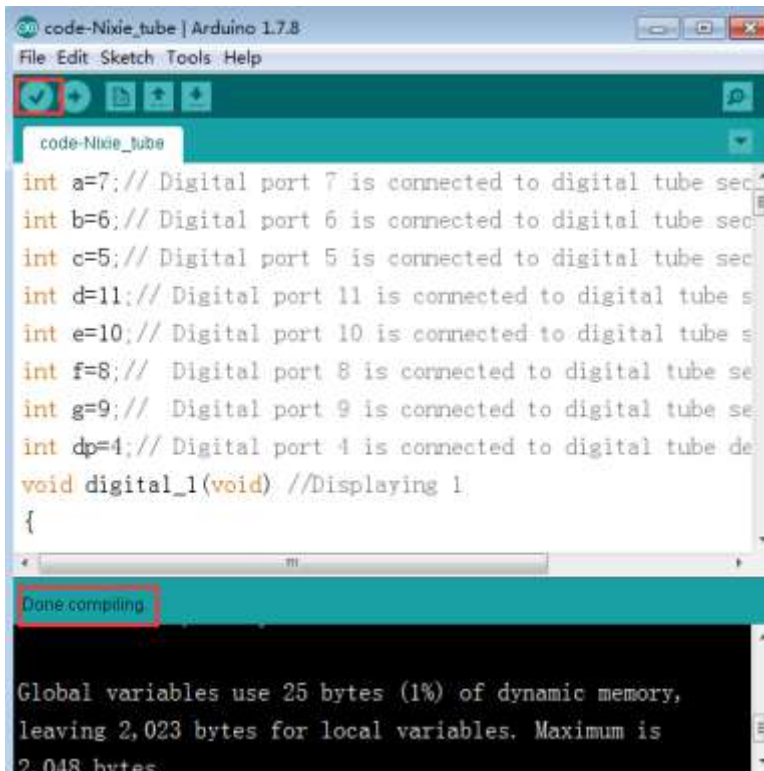
```

void digital_8(void) //Displaying 8
{
  unsigned char j;
  for(j=5;j<=11;j++)
    digitalWrite(j,HIGH);
    digitalWrite(dp,LOW);
}
void digital_9(void) //Displaying 9
{
  digitalWrite(a,HIGH);
  digitalWrite(b,HIGH);
  digitalWrite(c,HIGH);
  digitalWrite(d,HIGH);
  digitalWrite(e,LOW);
  digitalWrite(f,HIGH);
  digitalWrite(g,HIGH);
  digitalWrite(dp,HIGH);
}
void setup()
{
  int i; //Declarations of variables
  for(i=4;i<=11;i++)
    pinMode(i,OUTPUT); //Defining the port4-11 for the input port
}
void loop()
{
  while(1)
  {
    digital_1(); //Displaying 1
    delay(1000);
    digital_2(); //Displaying 2
    delay(1000);
    digital_3(); //Displaying 3
    delay(1000);
    digital_4(); //Displaying 4
    delay(1000);
    digital_5(); //Displaying 5
    delay(1000);
    digital_6(); //Displaying 6
    delay(1000);
    digital_7(); //Displaying 7
    delay(1000);
    digital_8(); //Displaying 8
    delay(1000);
    digital_9(); //Displaying 9
    delay(1000);
  }
}

```

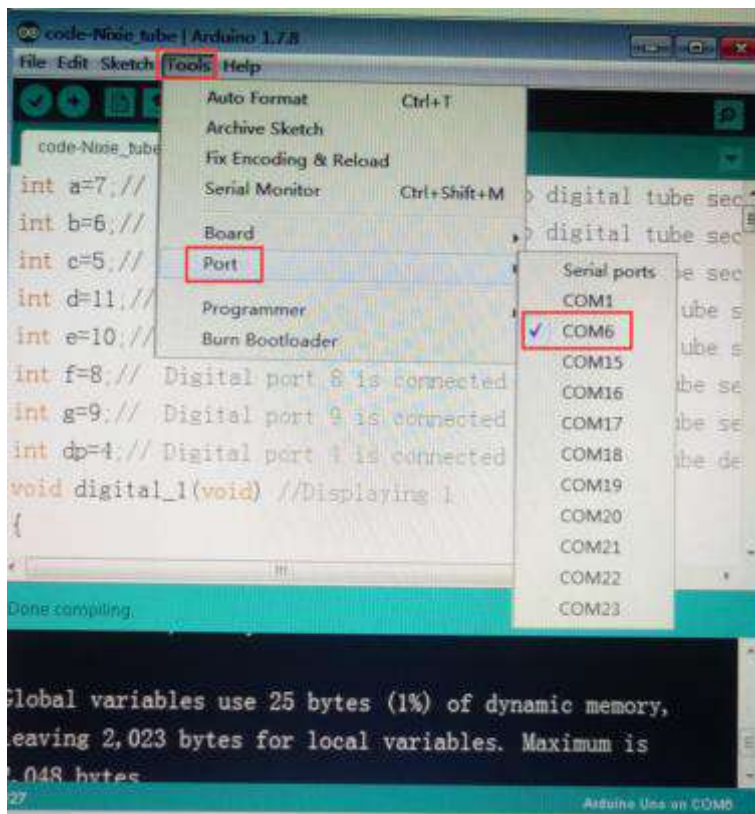
### Experimental steps:

1.We need to open the code for this experiment: code-Tilt\_switch.ino, click “√” under the menu bar, compile the code, and wait for the words of Done compiling in the lower left corner, as shown in the following figure.

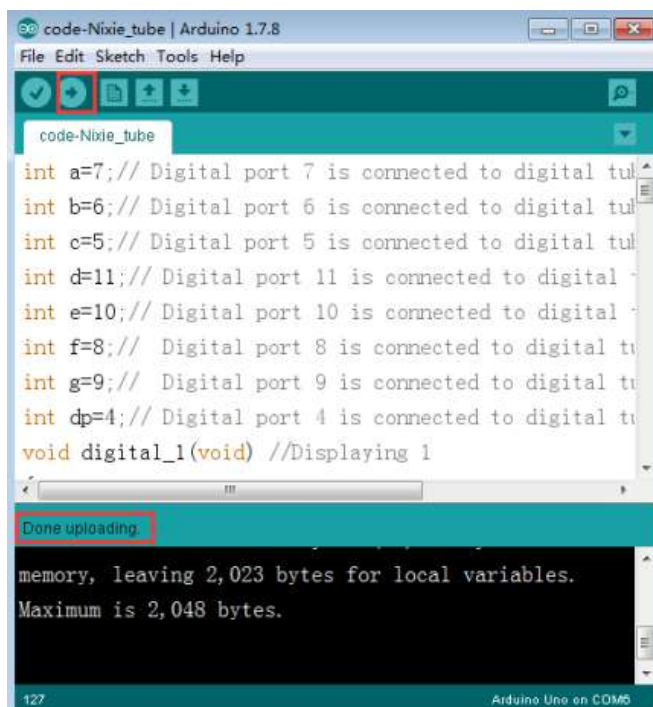


2. In the menu bar of Arduino IDE, we need to select **【Tools】** --- **【Port】** --- selecting the port that the serial number displayed by the device manager just now, as shown in the figure below. For example: COM6, as shown in the following figure.



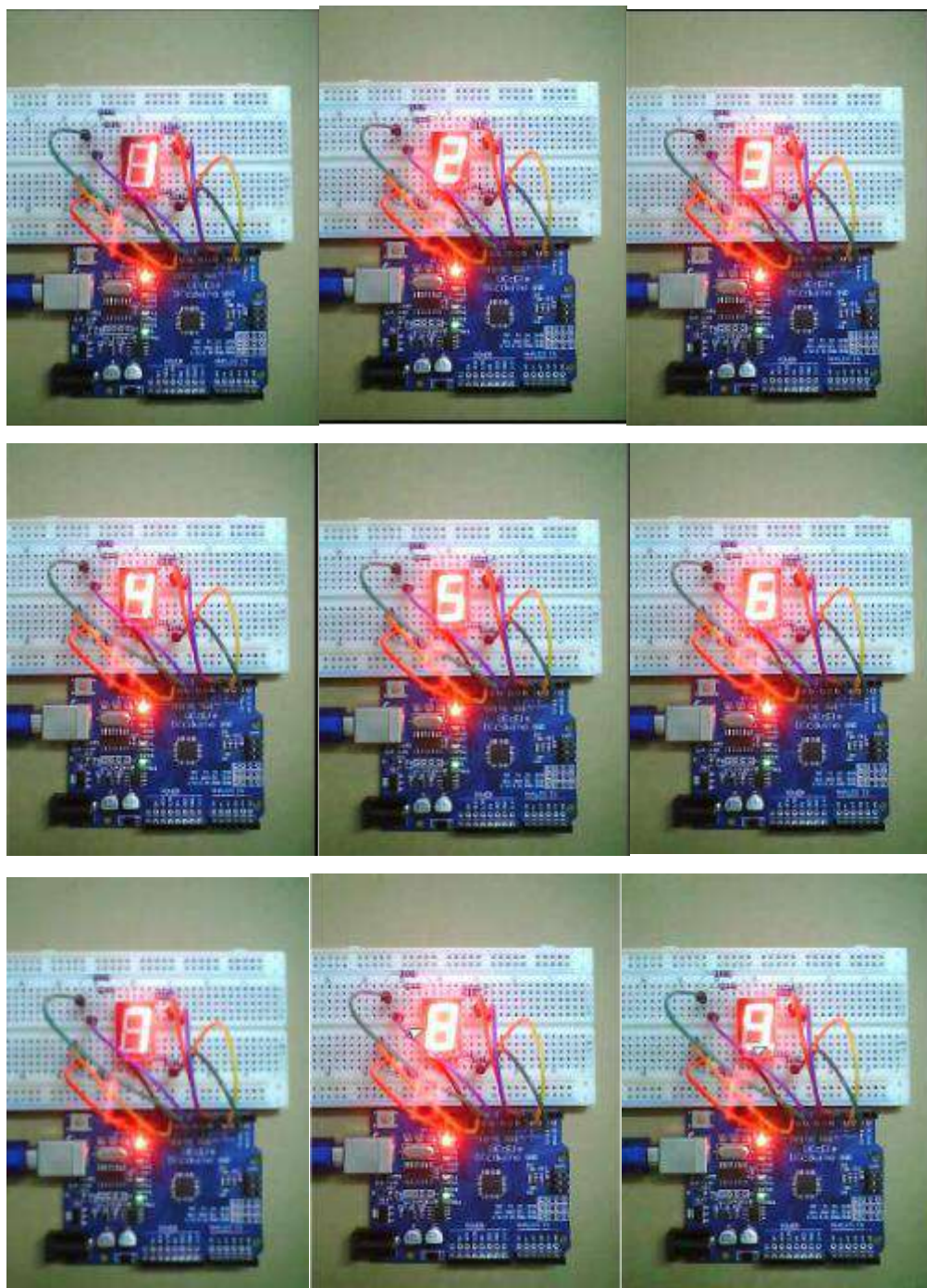


3. After the selection is completed, you need to click “→” under the menu bar to upload the code to the Arduino UNO board. When the word “Done uploading” appears in the lower left corner, the code has been successfully uploaded to the Arduino UNO board, as shown in the figure below.



4. After the code is uploaded, we can see that display 1-9 on a single 8-segment digital tube, as shown in the figure below.







## 16-4-Nixie tube

### The purpose of the experiment:

In this experiment, arduino was used to drive a four-digit tube with a common Yin. Is the purpose of the experiment: the first Nixie tube display 1, the second Nixie tube display 2, the third Nixie tube display 3 and fourth Nixie tube display 4 with such intervals of 0.5 seconds to display.

### Introduction to digital tube:

Nixie tube is a semiconductor luminescent device, its basic unit is a light-emitting diode. According to the number of digital tube is divided into 7-segment Nixie tube and 8-segment Nixie tube. 8-segment Nixie tube more than 7-segment Nixie tube a light-emitting diode unit (more than a decimal point), this experiment use the 8-segment Nixie tube. The actual object is shown below.



According to the light-emitting diode unit connection mode, it is divided into anode Nixie tubes and cathode Nixie tubes.

Anode Nixie tubes that connects the anodes of all light-emitting diodes together to form a common anode (COM). The common pole COM shall be connected to +5V when the common anode digital tube is applied. When the cathode of a certain field of light-emitting diode is low, the corresponding field will be light up. When the cathode of a field is high, the field does not light up.

Cathode Nixie tubes that connects the cathodes of all light-emitting diodes together to form a common cathode (COM). The common pole COM shall be connected to GND when the common cathode digital tube is applied. When the anode of a certain field of light-emitting diode is high, the corresponding field will be light up. When the anode of a field is low, the field does not light up.

### List of components required for the experiment:

Arduino UNO board \*1

USB cable \*1

220Ω resistor \*8

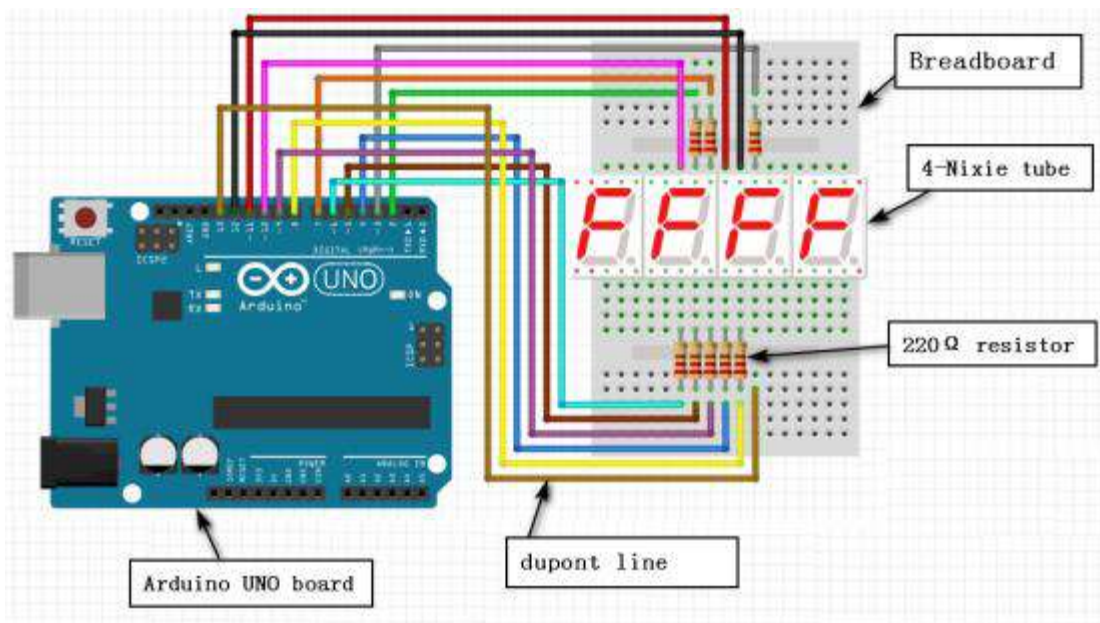
4bit 8-segment digital tube \*1

Breadboard \*1

dupont line \*1 bunch

Actual object connection diagram :

We need to connect the circuit as shown in the figure below.



### Experimental code analysis:

```
#define SEG_A 2 //Arduino Pin2--->SegLed Pin11
#define SEG_B 3 //Arduino Pin3--->SegLed Pin7
#define SEG_C 4 //Arduino Pin4--->SegLed Pin4
#define SEG_D 5 //Arduino Pin5--->SegLed Pin2
#define SEG_E 6 //Arduino Pin6--->SegLed Pin1
#define SEG_F 7 //Arduino Pin7--->SegLed Pin10
#define SEG_G 8 //Arduino Pin8--->SegLed Pin5
#define SEG_H 9 //Arduino Pin9--->SegLed Pin3

#define COM1 10 //Arduino Pin10--->SegLed Pin12
#define COM2 11 //Arduino Pin11--->SegLed Pin9
#define COM3 12 //Arduino Pin12--->SegLed Pin8
#define COM4 13 //Arduino Pin13--->SegLed Pin6
unsigned char table[10][8] =
{
{0, 0, 1, 1, 1, 1, 1, 1}, //0
{0, 0, 0, 0, 0, 1, 1, 0}, //1
{0, 1, 0, 1, 1, 0, 1, 1}, //2
{0, 1, 0, 0, 1, 1, 1, 1}, //3
{0, 1, 1, 0, 0, 1, 1, 0}, //4
{0, 1, 1, 0, 1, 1, 0, 1}, //5
{0, 1, 1, 1, 1, 1, 0, 1}, //6
{0, 0, 0, 0, 0, 1, 1, 1}, //7
{0, 1, 1, 1, 1, 1, 1, 1}, //8
{0, 1, 1, 0, 1, 1, 1, 1} //9
};
void setup()
{
pinMode(SEG_A,OUTPUT); //Defining the port for the output port
pinMode(SEG_B,OUTPUT);
pinMode(SEG_C,OUTPUT);
pinMode(SEG_D,OUTPUT);
pinMode(SEG_E,OUTPUT);
```

```

pinMode(SEG_F,OUTPUT);
pinMode(SEG_G,OUTPUT);
pinMode(SEG_H,OUTPUT);

pinMode(COM1,OUTPUT);
pinMode(COM2,OUTPUT);
pinMode(COM3,OUTPUT);
pinMode(COM4,OUTPUT);
}
void loop()
{
  Display(1,1); //Displaying 1 on the first bit of the Nixie tube
  delay(500);
  Display(2,2); //Displaying 2 on the second bit of the Nixie tube
  delay(500);
  Display(3,3); //Displaying 3 on the third bit of the Nixie tube
  delay(500);
  Display(4,4); //Displaying 4 on the fourth bit of the Nixie tube
  delay(500);
}
void Display(unsigned char com,unsigned char num)
{
  digitalWrite(SEG_A,LOW); //This is to get rid of the shadow
  digitalWrite(SEG_B,LOW);
  digitalWrite(SEG_C,LOW);
  digitalWrite(SEG_D,LOW);
  digitalWrite(SEG_E,LOW);
  digitalWrite(SEG_F,LOW);
  digitalWrite(SEG_G,LOW);
  digitalWrite(SEG_H,LOW);
  switch(com) //This is to select the display location
  {
    case 1:
      digitalWrite(COM1,LOW); //First bit of the Nixie tube
      digitalWrite(COM2,HIGH);
      digitalWrite(COM3,HIGH);
      digitalWrite(COM4,HIGH);
      break;
    case 2:
      digitalWrite(COM1,HIGH);
      digitalWrite(COM2,LOW); //Second bit of the Nixie tube
      digitalWrite(COM3,HIGH);
      digitalWrite(COM4,HIGH);
      break;
    case 3:
      digitalWrite(COM1,HIGH);
      digitalWrite(COM2,HIGH);
      digitalWrite(COM3,LOW); //Third bit of the Nixie tube
      digitalWrite(COM4,HIGH);
      break;
    case 4:
      digitalWrite(COM1,HIGH);

```

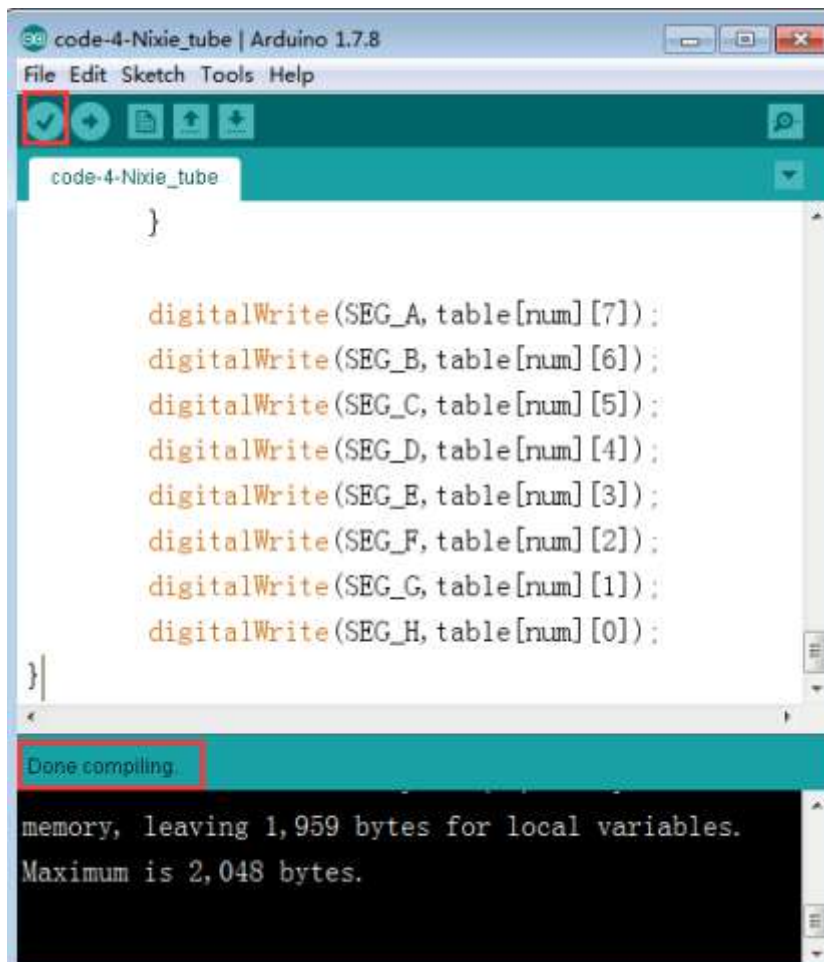
```

digitalWrite(COM2,HIGH);
digitalWrite(COM3,HIGH);
digitalWrite(COM4,LOW); //Fourth bit of the Nixie tube
break;
default:break;
}
digitalWrite(SEG_A,table[num][7]);
digitalWrite(SEG_B,table[num][6]);
digitalWrite(SEG_C,table[num][5]);
digitalWrite(SEG_D,table[num][4]);
digitalWrite(SEG_E,table[num][3]);
digitalWrite(SEG_F,table[num][2]);
digitalWrite(SEG_G,table[num][1]);
digitalWrite(SEG_H,table[num][0]);
}

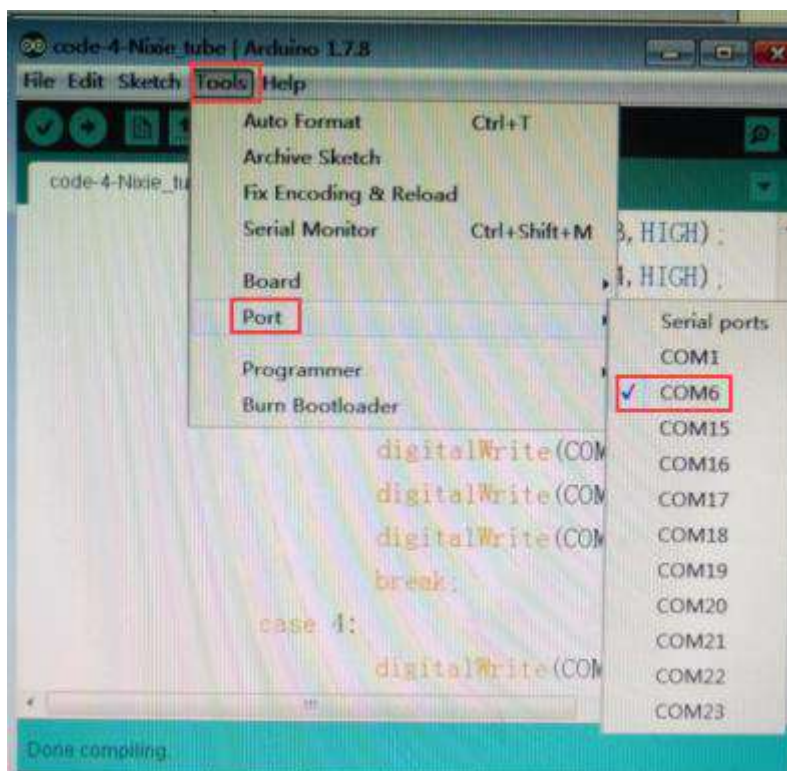
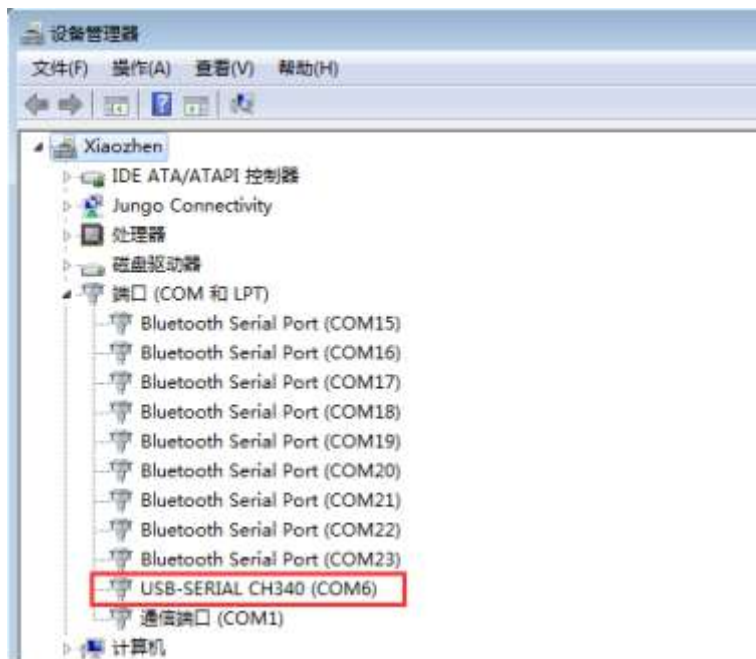
```

Experimental steps:

1.We need to open the code for this experiment: code-4-Nixie\_tube.ino, click “√” under the menu bar, compile the code, and wait for the words of Done compiling in the lower left corner, as shown in the following figure.

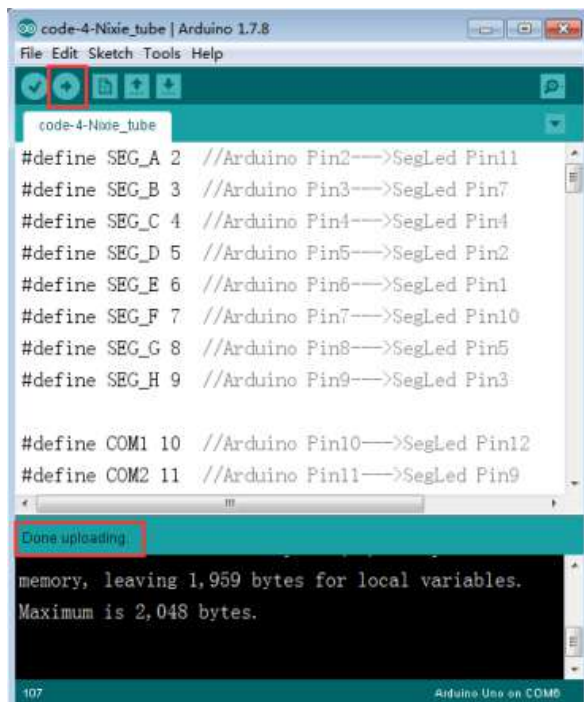


2. In the menu bar of Arduino IDE, we need to select the **【Tools】** --- **【Port】** --- select the port that the serial number displayed by the device manager just now. for example:COM6,as shown in the following figure.



3. After the selection is completed, click “→” under the menu bar, and upload the code to the Arduino UNO board, when appears to Done uploading on the lower left corner , that means that the code has been successfully uploaded to the Arduino UNO board, as shown in the following figure.





```

code-4-Nixie_tube | Arduino 1.7.8
File Edit Sketch Tools Help

code-4-Nixie_tube

#define SEG_A 2 //Arduino Pin2-->SegLed Pin11
#define SEG_B 3 //Arduino Pin3-->SegLed Pin7
#define SEG_C 4 //Arduino Pin4-->SegLed Pin4
#define SEG_D 5 //Arduino Pin5-->SegLed Pin2
#define SEG_E 6 //Arduino Pin6-->SegLed Pin1
#define SEG_F 7 //Arduino Pin7-->SegLed Pin10
#define SEG_G 8 //Arduino Pin8-->SegLed Pin5
#define SEG_H 9 //Arduino Pin9-->SegLed Pin3

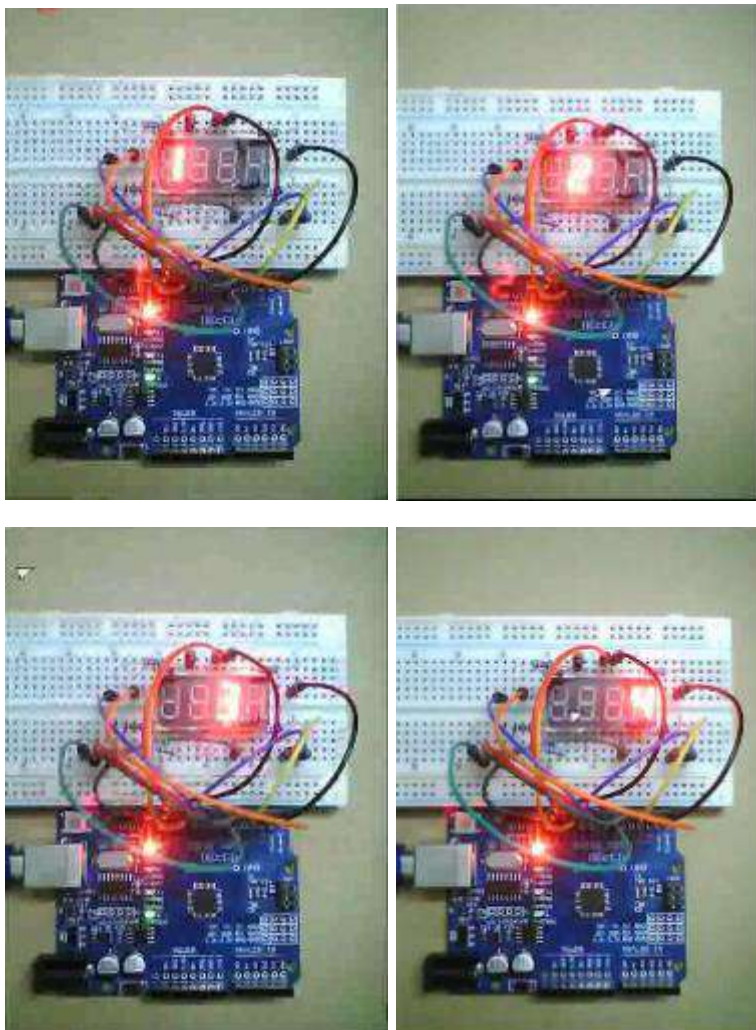
#define COM1 10 //Arduino Pin10-->SegLed Pin12
#define COM2 11 //Arduino Pin11-->SegLed Pin9

Done uploading.
memory, leaving 1,959 bytes for local variables.
Maximum is 2,048 bytes.

107 Arduino Uno on COM6

```

4. After the code is uploaded, the first Nixie tube display 1, the second Nixie tube display 2, the third Nixie tube display 3 and fourth Nixie tube display 4 with such intervals of 0.5 seconds to display.



## 17-74HC595

### The purpose of the experiment:

74HC595 is an 8-bit serial input and parallel output displacement buffer: the parallel output is three-state output. In this course, we use three digital I/O ports of Arduino to control 8 LED lights by 74HC595, so that they were lit in 8-bit binary (0-256) order.

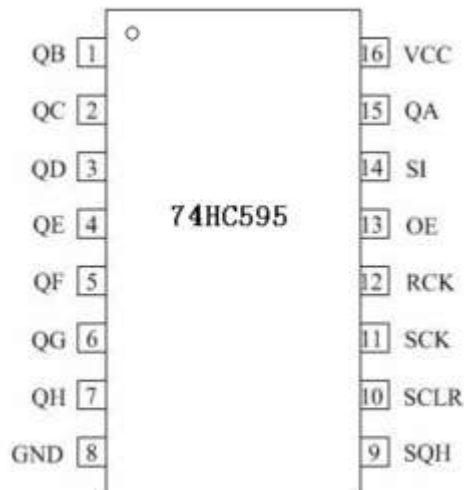
The actual object is shown below.



Binary order:

```
00000001 00000010 00000011 00000100 00000101 00000110
00000111 00001000 00001001 00001010 00001011 00001100
```

```
.....
10000000
```



number of pin	name of pin	Description
1,2,3,4,5,6,7,15	QB,QC,QD,QE,QG,QH,QA	Tri-state output pin
8	GND	GND
9	SQH	Serial port data output pin
10	SCLR	Shift register clear
11	SCK	Data input clock line
12	RCK	Output memory latch clock line
13	OE	Output enable
14	SI	Data line

16

VCC

VCC

**List of components required for the experiment:**

Arduino UNO board \*1

USB cable \*1

74HC595 \*1

220 $\Omega$  resistor \*8

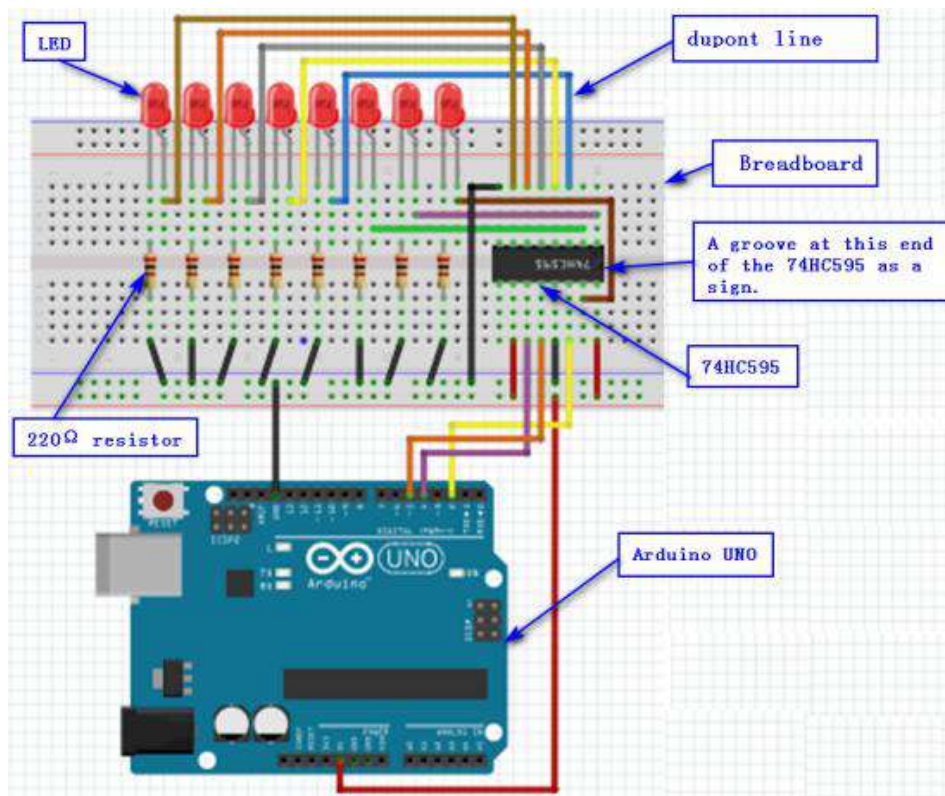
LED \*8

Breadboard \*1

Dupont line \*1bunch

**Material object connection diagram :**

We need to connect circuit as shown in the following figure.

**Experimental code analysis:**

```
//connect 74hc595 pin10:MR--->VCC; Pin13:OE--->GND
```

```
int latchPin = 5; //to 595 pin12
```

```
int clockPin = 4; //to 595 pin11
```

```
int dataPin = 2; //to 595 pin14
```

```
void setup ()
```

```
{
```

```
  pinMode(latchPin,OUTPUT); //Defining the port5 for the output port
```

```
  pinMode(clockPin,OUTPUT); //Defining the port4 for the output port
```

```
  pinMode(dataPin,OUTPUT); //Defining the port2 for the output port
```

```
}
```

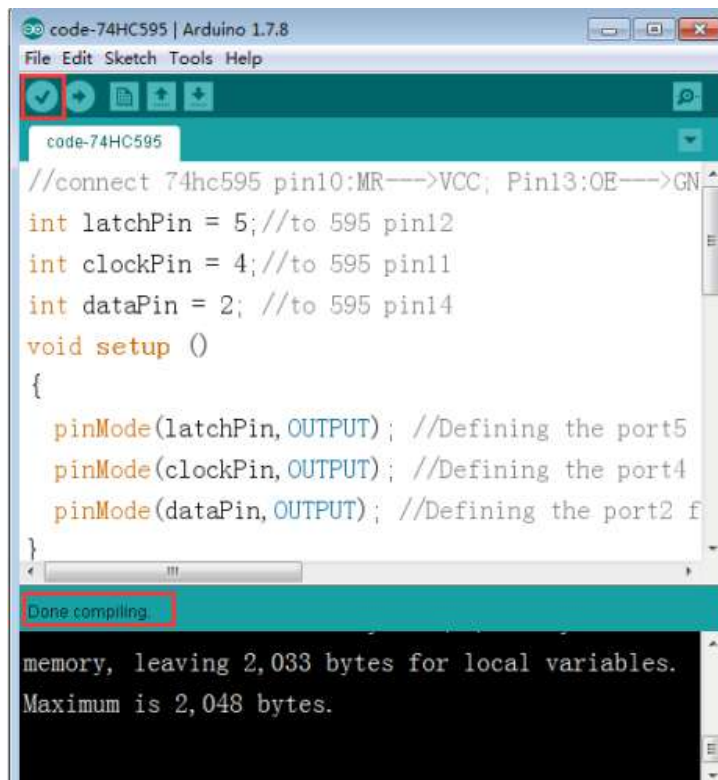
```

void loop()
{
  for(int a=0; a<256; a++) //The meaning of this loop is to let a variable increase by 1
  until it is equal to 256.
      //The following activities are performed every cycle.
  {
    digitalWrite(latchPin,LOW); //Giving a low level to the port ST_CP indicates that the
    chip is ready to receive data.
    shiftOut(dataPin,clockPin,MSBFIRST,a);
    /*
    dataPin : Data output pin, each bit of data will be output sequentially. Mode of pin
    needs to be set to output.
    clockPin : Clock output pin. Mode of pin needs to be set to output
    bitOrder : Data shift order selection bit.The type of this parameter is byte,
    High-level first-entry MSBFIRST or low-level first-entry LSBFIRST Can be selected
    by yourself.
    a:The data value to be output.
    */
    digitalWrite(latchPin,HIGH); //Giving a low level to the port ST_CP
    delay(1000); //Pause for 1 second to make you see the effect
  }
}

```

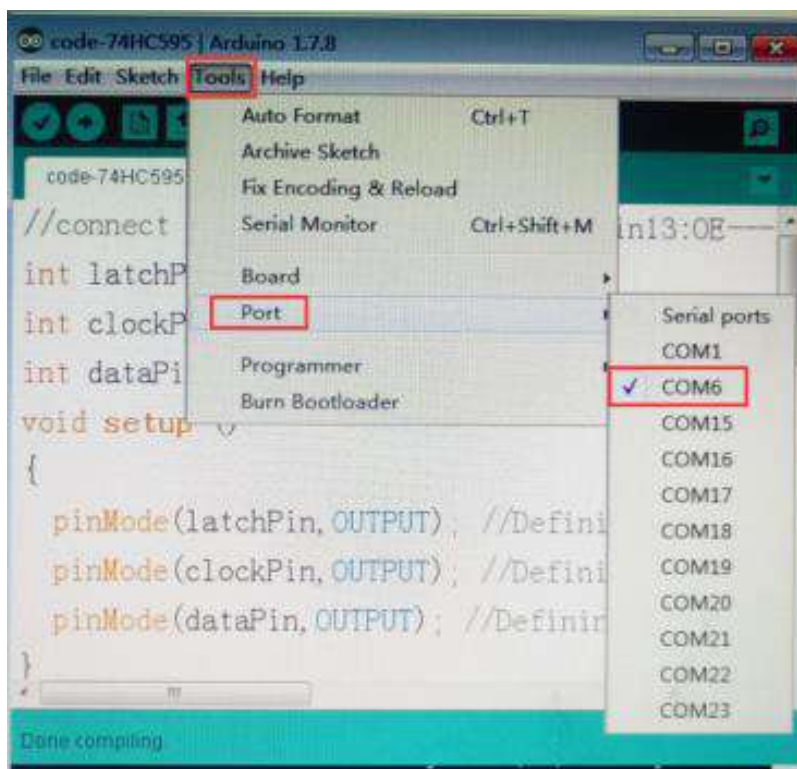
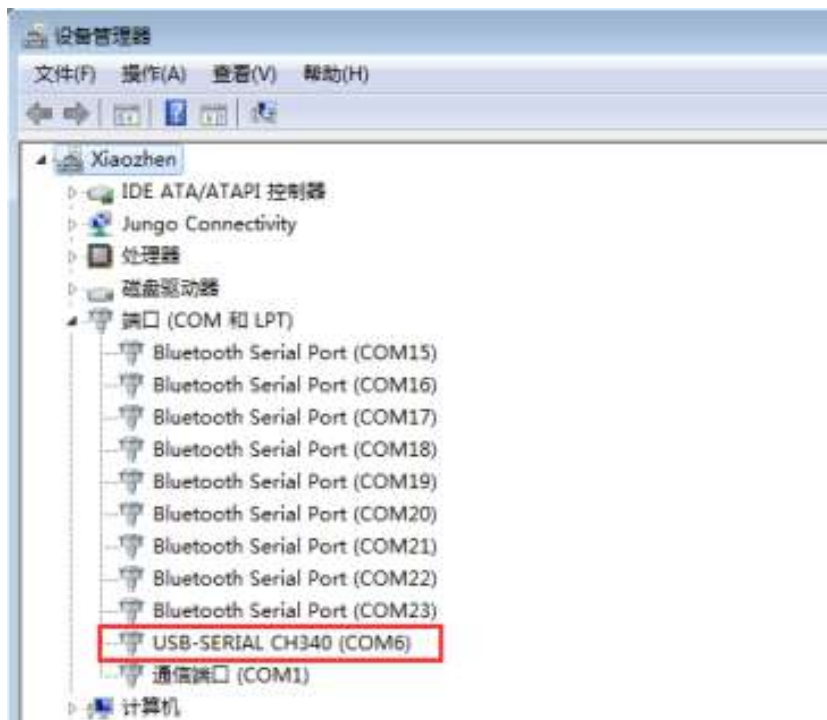
### Experimental steps:

1.We need to open the code for this experiment: code-74HC595.ino, click “√” under the menu bar, compile the code, and wait for the words of Done compiling in the lower left corner, as shown in the following figure.



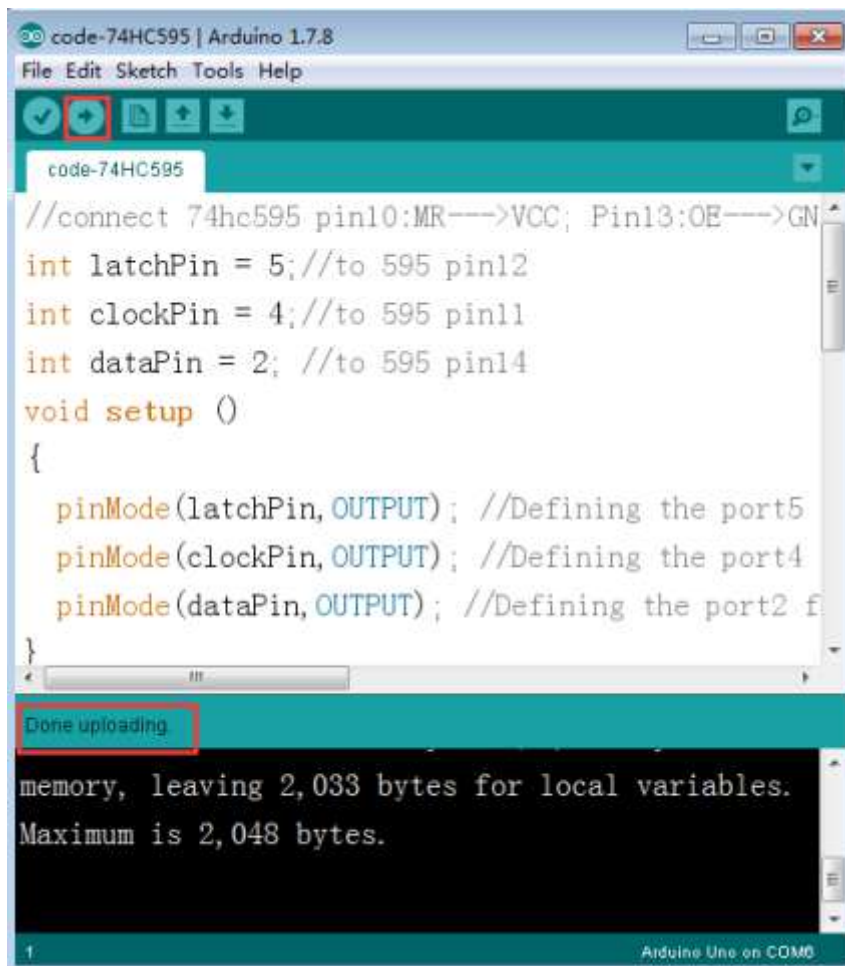
2. In the menu bar of Arduino IDE, we need to select the **Tools** --- **Port** --- select the port that the serial number displayed by the device manager just now. for example: COM6, as shown in the following figure.





3. After the selection is completed, you need to click “→” under the menu bar, and upload the code to the Arduino UNO board, when appears to Done uploading on the lower left corner , that means that the code has been successfully uploaded to the Arduino UNO board, as shown in the following figure.





```

code-74HC595 | Arduino 1.7.8
File Edit Sketch Tools Help

code-74HC595
//connect 74hc595 pin10:MR-->VCC; Pin13:OE-->GN
int latchPin = 5; //to 595 pin12
int clockPin = 4; //to 595 pin11
int dataPin = 2; //to 595 pin14
void setup ()
{
  pinMode(latchPin, OUTPUT); //Defining the port5
  pinMode(clockPin, OUTPUT); //Defining the port4
  pinMode(dataPin, OUTPUT); //Defining the port2 f
}

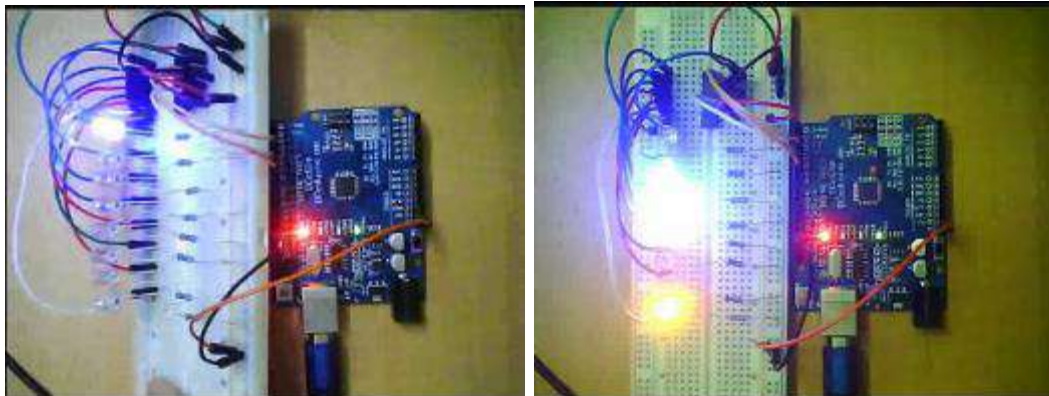
```

Done uploading.

memory, leaving 2,033 bytes for local variables.  
Maximum is 2,048 bytes.

1 Arduino Uno on COM6

4. After the code is uploaded, We can see that 8 LEDs will be lit from 00000001 to 10000000, as shown in the following figure.(Just an example)



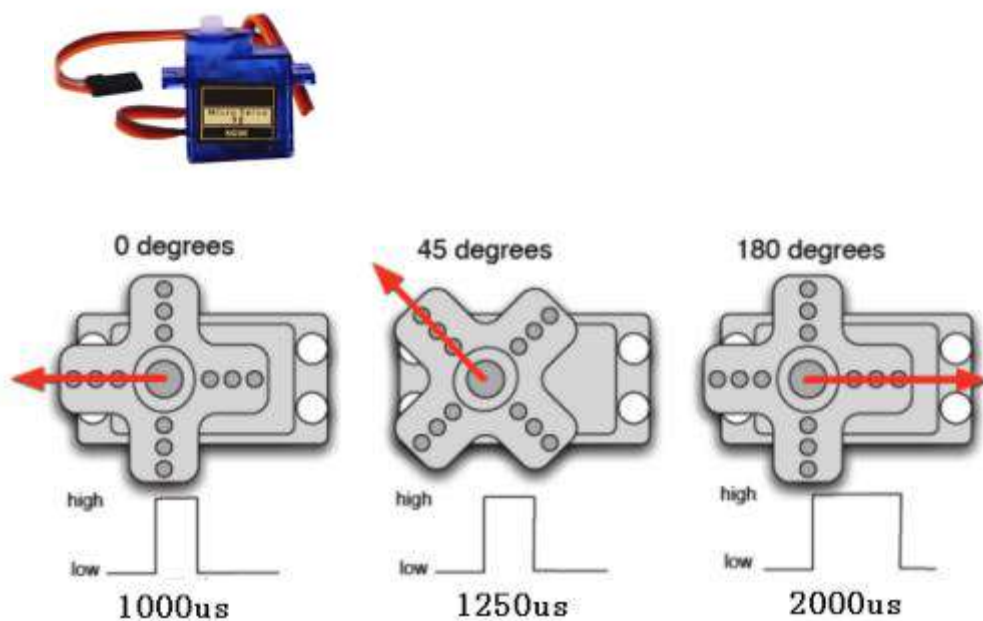
# 18-servo control

## The purpose of the experiment:

Based on Arduino UNO, a code is written to rotate the servo to the angle corresponding to the user's input number, and the angle print is displayed on the serial monitor of the Arduino IDE.

About the servo :

The actual object is shown below. Servo rotation angle is by adjusting the duty ratios of PWM (pulse width modulation) signal. The standard PWM (pulse width modulation) signal has a fixed period of 20ms (50Hz). Theoretically, pulse width distribution should be between 1 ms to 2 ms, but in fact between pulse width can be 0.5 ms and 2.5 ms. Pulse width and the servo rotation angle  $0^{\circ} \sim 180^{\circ}$  corresponds, as shown in the figure below.



Servo have many specifications, but all of the servo possess external three lines, with brown, red, orange, three kinds of color to distinguish. Due to brand is different, color is different, **brown for the grounding line, red for positive line, orange for signal lines.**

**Note:** Due to brand is different, for the same signal, different brands of servo rotation angle will be different.

List of components required for the experiment:

Arduino UNO board \*1

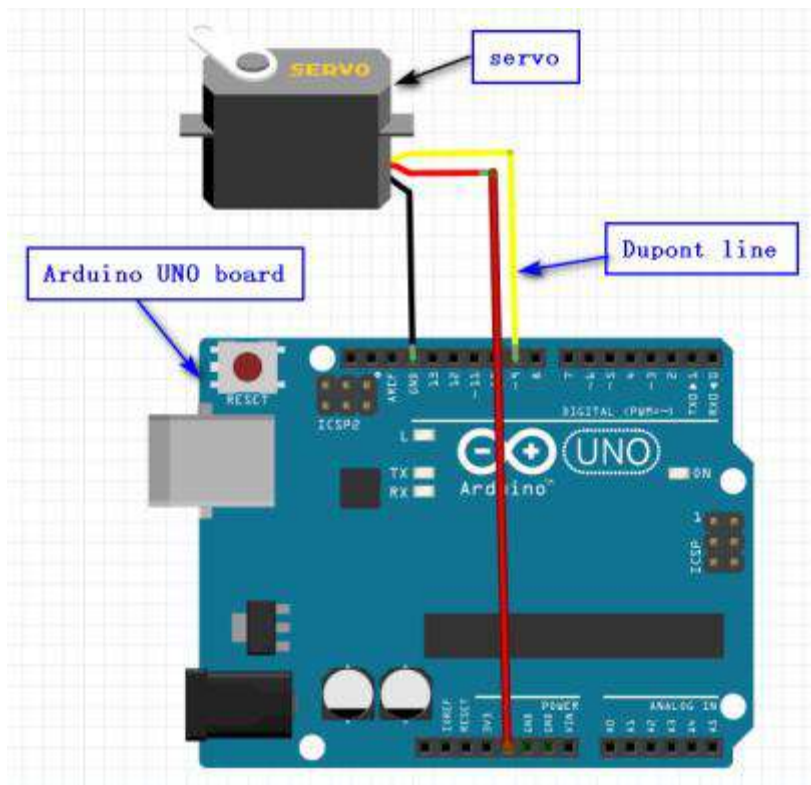
USB cable \*1

Servo \*1

Dupont line \*1 bunch

Actual object connection diagram :

We need to connect the circuit as shown in the figure below.



### Experimental code analysis:

```
//UART send 1~9==>20~180 degree
int servopin=9;//Defining the port 9 for the servopin
int myangle;//Define Angle variable
int pulsewidth;//Define the pulse width variable
int val;
void servopulse(int servopin,int myangle)
/*A pulse function is defined to generate PWM values by simulation */
{
    pulsewidth=(myangle*11)+500;//Convert the Angle to 500-2480 pulse width
    digitalWrite(servopin,HIGH);//Giving a high level to the servo interface
    delayMicroseconds(pulsewidth);//The number of microseconds of delay pulse width
    digitalWrite(servopin,LOW);//Giving a low level to the servo interface
    delay(20-pulsewidth/1000);//The remaining time in the delay period
}
void setup()
{
    pinMode(servopin,OUTPUT);//Defining the servopin port for the output port
    Serial.begin(9600);//The baud rate is 9600
    Serial.println("servo=o_serail_simple ready" );
}
void loop()
{
    val=Serial.read();//Reading the data received by the serial port
    if(val>'0'&&val<='9')//Determining whether the received data values conform to the
    range
    {
        val=val-'0';//Convert ASCII code to a value,for exmaple : '9'-'0'=0x39-0x30=9
        val=val*(180/9);//Convert Numbers into angles,for exmaple : 9* (180/9) =180
        Serial.print("moving servo to ");
```

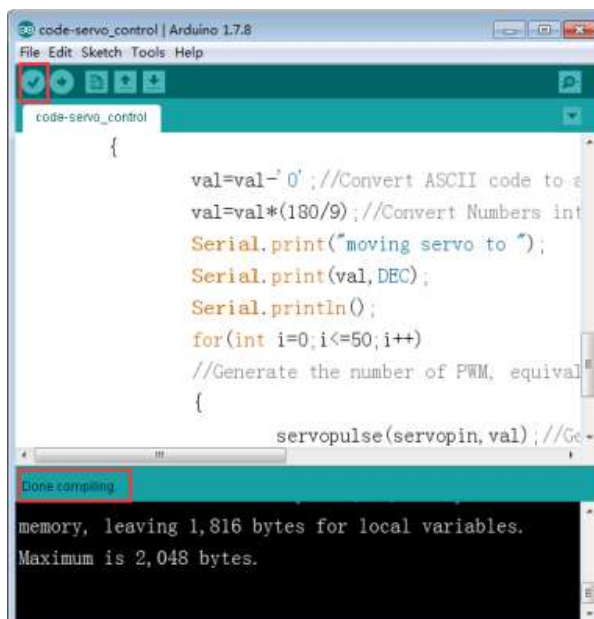
```

Serial.print(val,DEC);
Serial.println();
for(int i=0;i<=50;i++)
//Generate the number of PWM, equivalent delay to ensure that the response Angle
can be turned
{
servopulse(servopin,val);//Generate PWM values by simulation
}
}
}

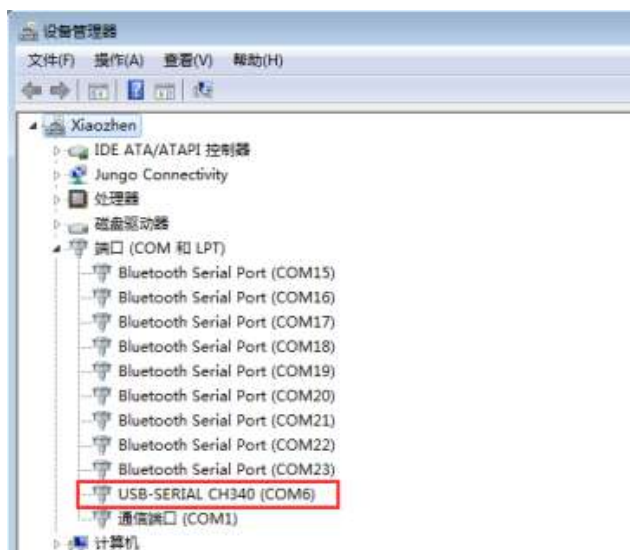
```

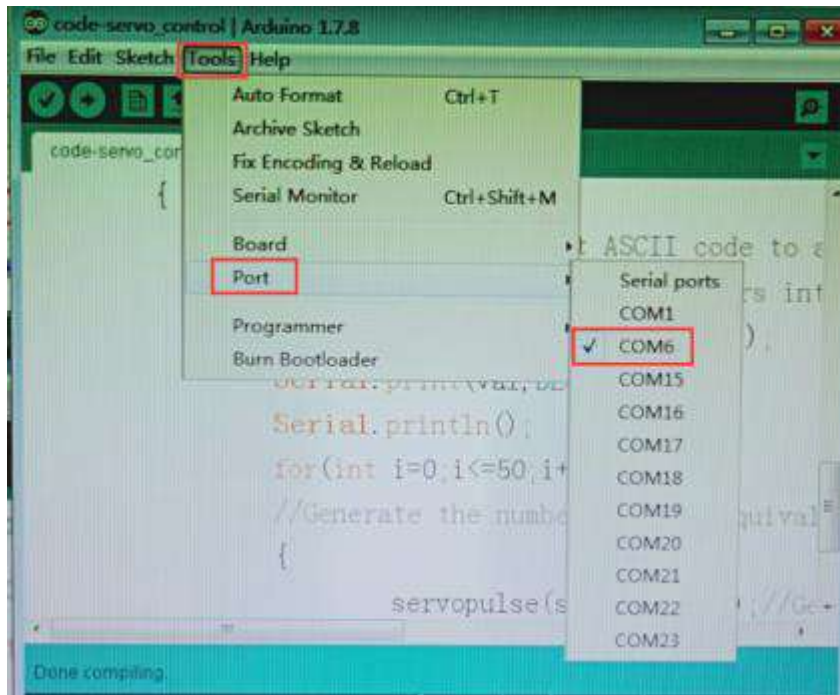
Experimental steps:

1. We need to open the code for this experiment: code-servo\_control.ino, click “√” under the menu bar, compile the code, and wait for the words of Done compiling in the lower left corner, as shown in the following figure.

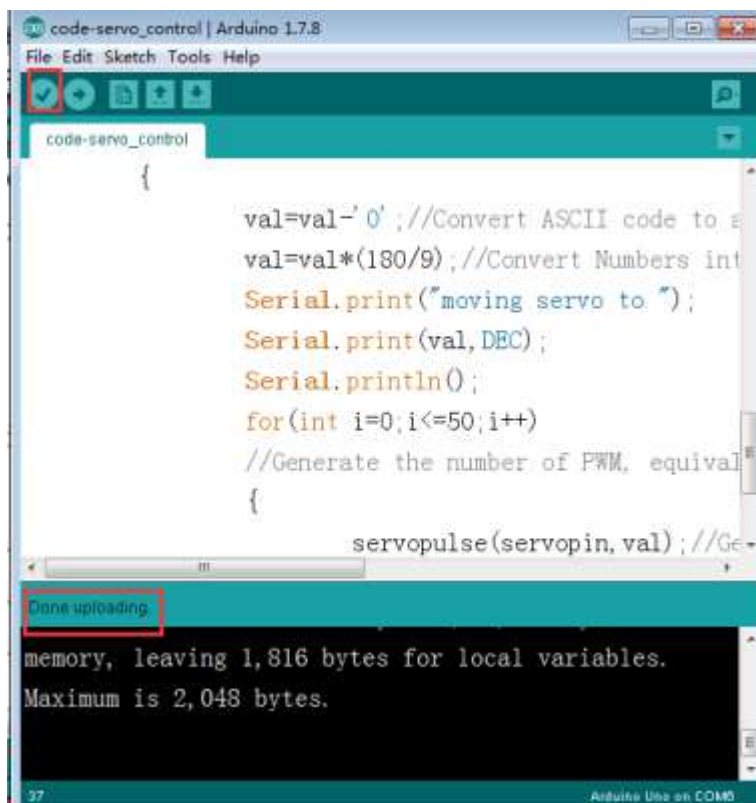


2. In the menu bar of Arduino IDE, you need to select the **【Tools】** --- **【Port】** --- select the port that the serial number displayed by the device manager just now. for example: COM6, as shown in the following figure.



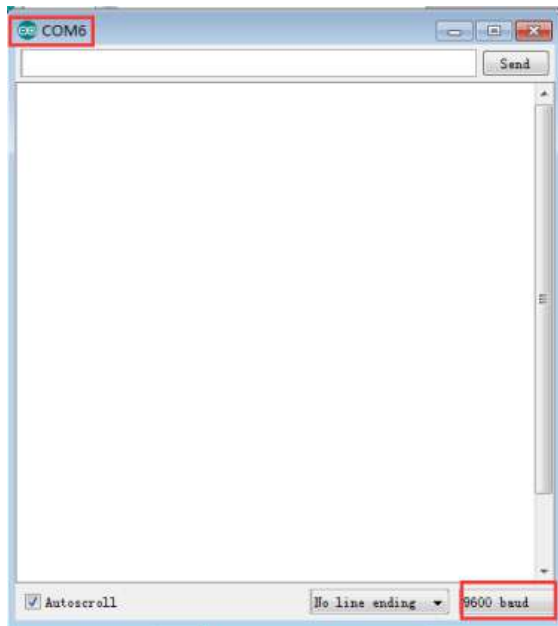


3. After the selection is completed, you need to click “→” under the menu bar, and upload the code to the Arduino UNO board, when appears to Done uploading on the lower left corner , that means that the code has been successfully uploaded to the Arduino UNO board, as shown in the following figure.

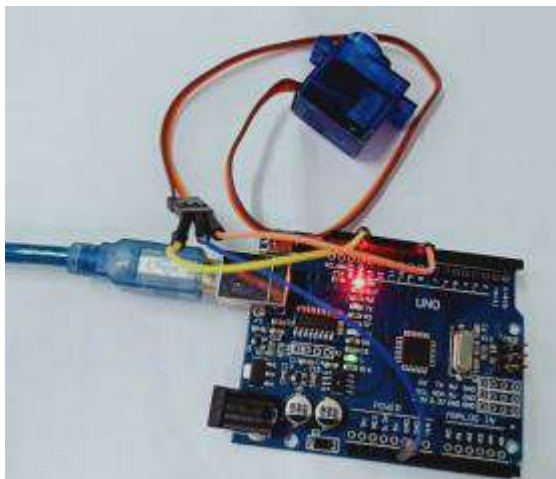
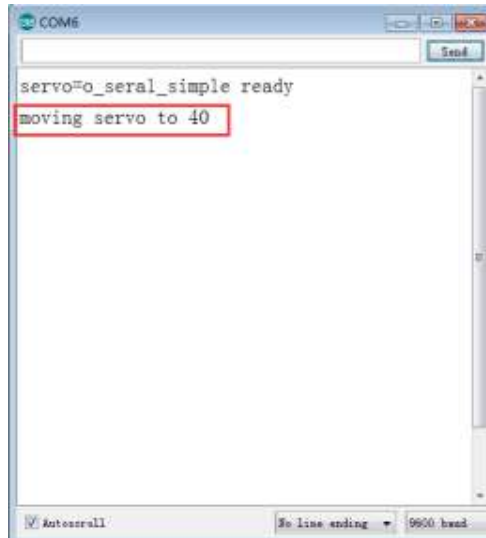
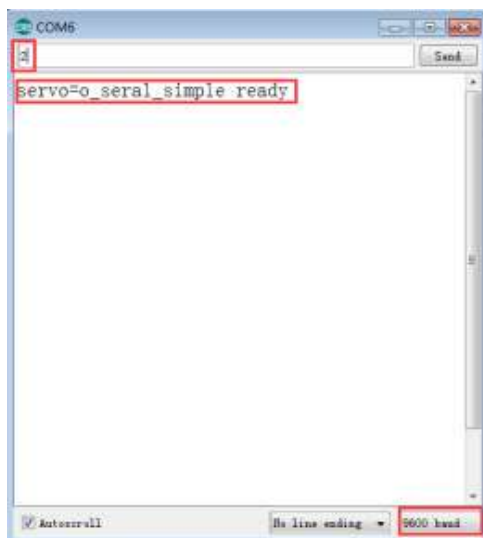


4. You can open the serial port monitor on the top right corner of Arduino IDE, A serial port of Arduino port will appear, and the baud rate is set to 9600 on the lower right corner, as shown in the following figure.





5. After the code is uploaded, we open the serial port monitor of Arduino IDE, you can see the words "servo=o\_serail\_simple ready" written in the program. And then input a number between 1 ~ 9 randomly in the send box, servo will turn the corresponding angle. Moreover, the serial port monitor will print out the corresponding angle, a comment in the program: "UART send 1~9= >20~180 degree" as shown in the figure below (for example only).



## 19-IR control

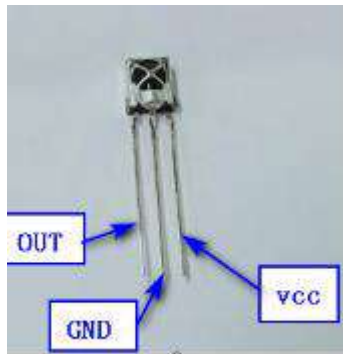
### The purpose of the experiment:

In this experiment, we will make the IR remote controller communicate with the IR receiver sensor.

### About the infrared remote control :

The signal from the IR remote controller is a series of binary pulse codes. In order to protect it from other infrared signals during wireless transmission. It is modulated on a specific carrier frequency, and then transmitted by infrared emission sensor. The infrared receiving device need to filter out other waveform and receive the signal of the specific frequency and restore it to binary pulse code, this process is called demodulation.

The IR receiver sensor converts the optical signal emitted by the infrared emission sensor to a weak electrical signal. These signals are restored to the original encode by various circuits, finally outputs the signal to the control circuit.



### List of components required for the experiment:

Arduino UNO board \*1

USB cable \*1

IR receiver sensor \*1

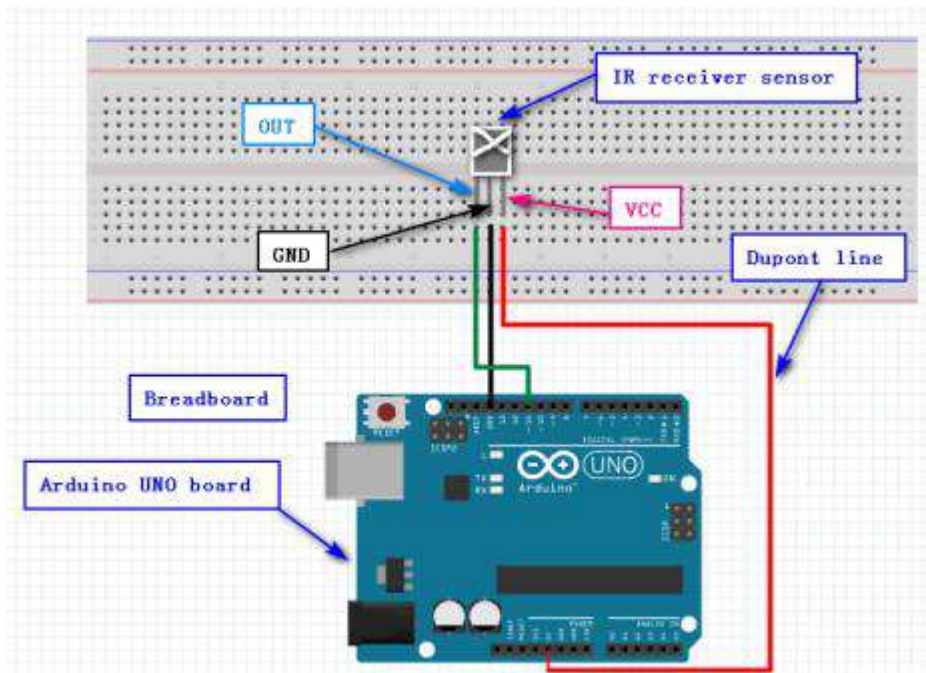
IR remote controller \*1

Breadboard \*1

Dupont line \*1 bunch

### Actual object connection diagram :

We need to connect the circuit as shown in the figure below.



### Experimental code analysis:

```
#include <IRremote.h> //Including infrared library
int RECV_PIN = 11; // Declarations of port
int LED1 = 2;
int LED2 = 3;
int LED3 = 4;
int LED4 = 5;
int LED5 = 6;
int LED6 = 7;
long on1 = 0x00FF6897; //Code the example to match the send
long off1 = 0x00ff30CF;
long on2 = 0x00FF9867;
long off2 = 0x00FF18E7;
long on3 = 0x00FFB04F;
long off3 = 0x00FF7A85;
long on4 = 0x00FF10EF;
long off4 = 0x00FF42BD;
long on5 = 0x00FF38C7;
long off5 = 0x00FF4AB5;
long on6 = 0x00FF5AA5;
long off6 = 0x00FF52AD;
IRrecv irrecv(RECV_PIN);
decode_results results; //Declarations of struct
// Dumps out the decode_results structure.
// Call this after IRrecv::decode()
// void * to work around compiler issue
//void dump(void *v) {
// decode_results *results = (decode_results *)v
void dump(decode_results *results)
{
int count = results->rawlen;
if (results->decode_type == UNKNOWN)
{
```

```

Serial.println("Could not decode message");
}
else
{
if (results->decode_type == NEC)
{
Serial.print("Decoded NEC: ");
}
else if (results->decode_type == SONY)
{
Serial.print("Decoded SONY: ");
}
else if (results->decode_type == RC5)
{
Serial.print("Decoded RC5: ");
}
else if (results->decode_type == RC6)
{
Serial.print("Decoded RC6: ");
}
Serial.print(results->value, HEX);
Serial.print(" ");
Serial.print(results->bits, DEC);
Serial.println(" bits");
}
Serial.print("Raw ");
Serial.print(count, DEC);
Serial.print("): ");
for (int i = 0; i < count; i++)
{
if ((i % 2) == 1)
{
Serial.print(results->rawbuf[i]*USECPERTICK, DEC);
}
else
{
Serial.print(-(int)results->rawbuf[i]*USECPERTICK, DEC);
}
Serial.print(" ");
}
Serial.println("");
}
void setup()
{
pinMode(RECV_PIN, INPUT); //Defining the RECV port for the input port
pinMode(LED1, OUTPUT); //Defining the LED1 port for the output port
pinMode(LED2, OUTPUT); //Defining the LED2 port for the output port
pinMode(LED3, OUTPUT); //Defining the LED3 port for the output port
pinMode(LED4, OUTPUT); //Defining the LED4 port for the output port
pinMode(LED5, OUTPUT); //Defining the LED5 port for the output port
pinMode(LED6, OUTPUT); //Defining the LED6 port for the output port
pinMode(13, OUTPUT); //Defining the port13 for the output port

```

```

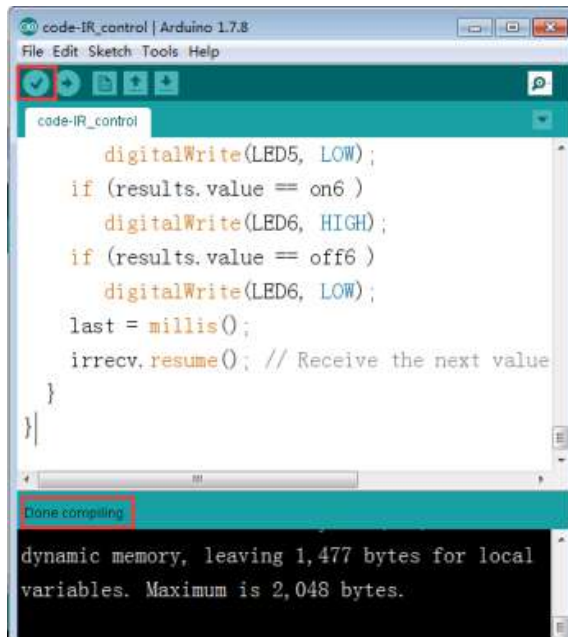
Serial.begin(9600); //The baud rate is 9600
irrecv.enableIRIn(); // Start the receiver
}
int on = 0;
unsigned long last = millis();
void loop()
{
  if (irrecv.decode(&results)) //Calling the library function: decode
  {
    // If it's been at least 1/4 second since the last
    // IR received, toggle the relay
    if (millis() - last > 250)
    {
      on = !on;
      digitalWrite(13, on ? HIGH : LOW);
      dump(&results);
    }
    if (results.value == on1 )
      digitalWrite(LED1, HIGH);
    if (results.value == off1 )
      digitalWrite(LED1, LOW);
    if (results.value == on2 )
      digitalWrite(LED2, HIGH);
    if (results.value == off2 )
      digitalWrite(LED2, LOW);
    if (results.value == on3 )
      digitalWrite(LED3, HIGH);
    if (results.value == off3 )
      digitalWrite(LED3, LOW);
    if (results.value == on4 )
      digitalWrite(LED4, HIGH);
    if (results.value == off4 )
      digitalWrite(LED4, LOW);
    if (results.value == on5 )
      digitalWrite(LED5, HIGH);
    if (results.value == off5 )
      digitalWrite(LED5, LOW);
    if (results.value == on6 )
      digitalWrite(LED6, HIGH);
    if (results.value == off6 )
      digitalWrite(LED6, LOW);
    last = millis();
    irrecv.resume(); // Receive the next value
  }
}

```

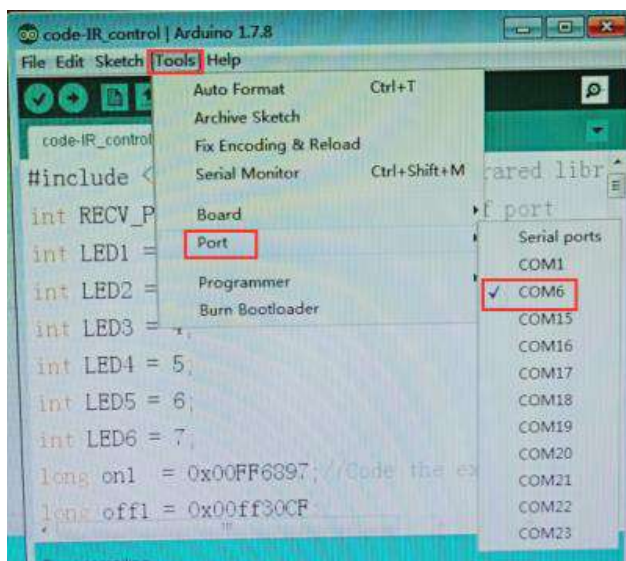
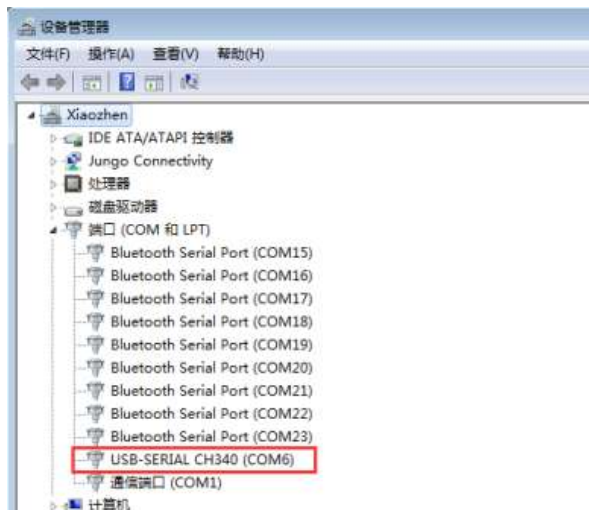
### Experimental steps:

1. You need to open the code for this experiment: code-IR\_control.ino, click “√” under the menu bar, compile the code, and wait for the words of Done compiling in the lower left corner, as shown in the following figure.

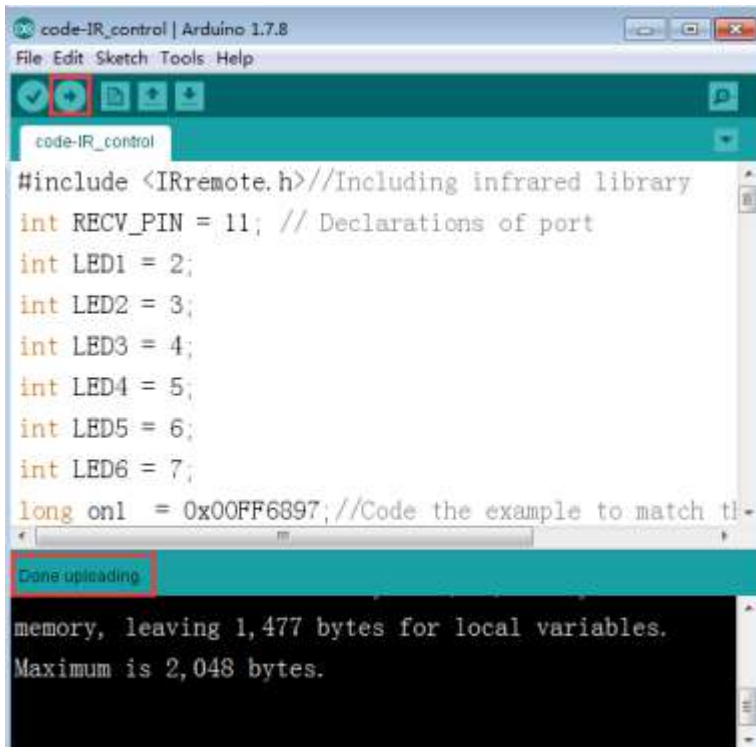




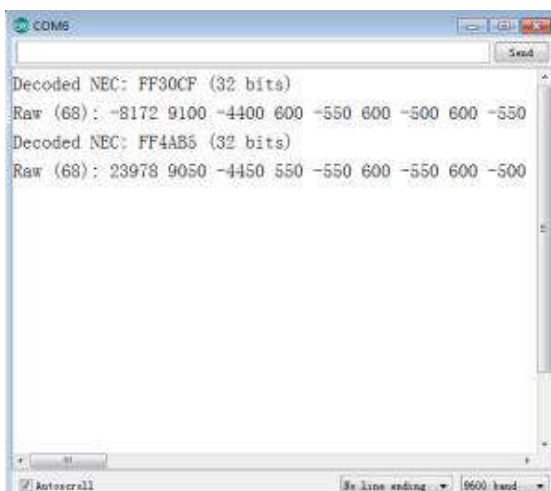
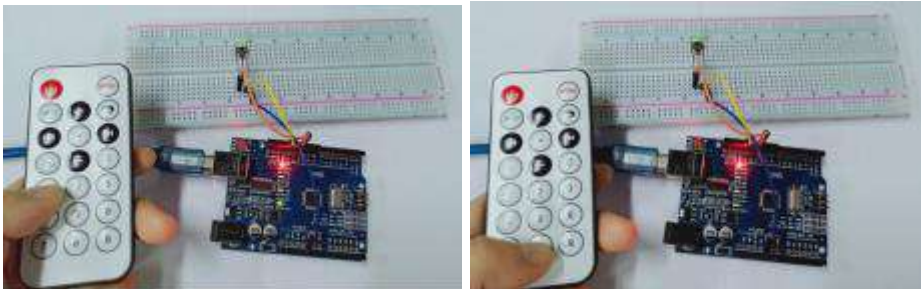
2. In the menu bar of Arduino IDE, select the **Tools** --- **Port** --- select the port that the serial number displayed by the device manager just now. for example: COM6, as shown in the following figure.



3. After the selection is completed, you need to click “→” under the menu bar, and upload the code to the Arduino UNO board, when appears to Done uploading on the lower left corner , that means that the code has been successfully uploaded to the Arduino UNO board, as shown in the following figure.



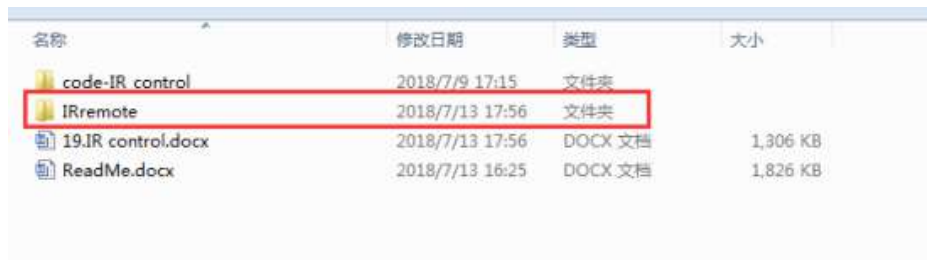
4. After the code is uploaded, we need to open the serial monitor of Arduino IDE, and set the baud rate to 9600. When we press the button on the infrared remote controller, we can see the code value of the corresponding button on the serial monitor, as shown below ( Just for example).



## Steps to add a library file

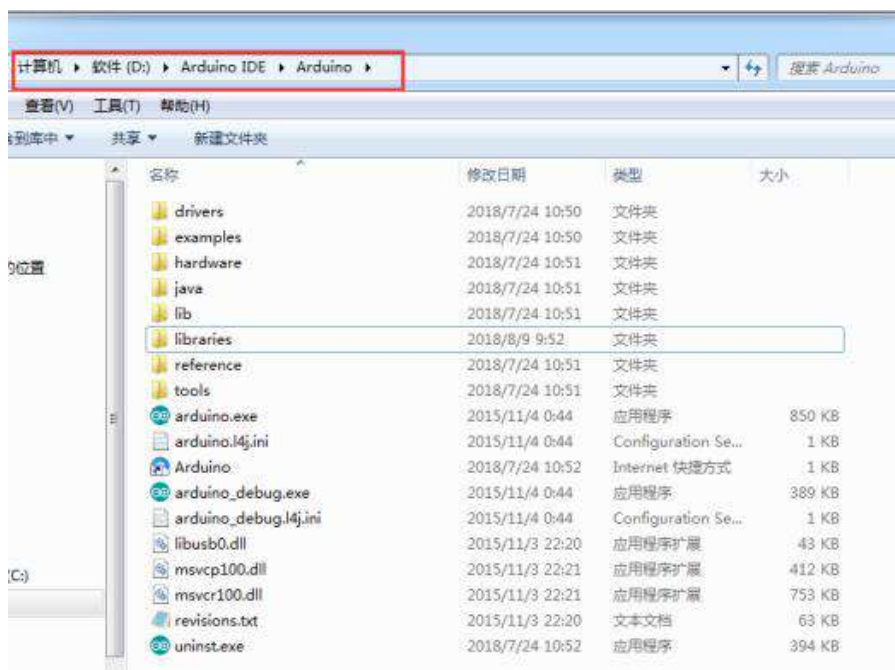
**Note:** Before you compile the code, you must look at this steps.

1. We need to add IRremote file, as shown in the figure below.



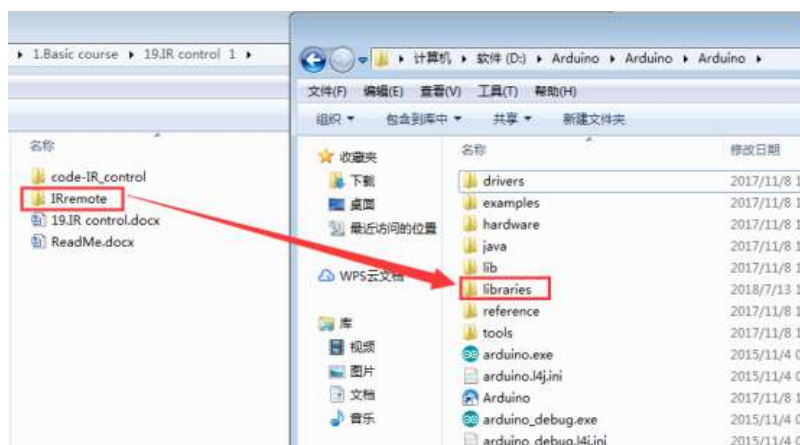
2. You need to find the installation path of Arduino. As shown in the figure below. ( just for example)

This is my Arduino installation path.

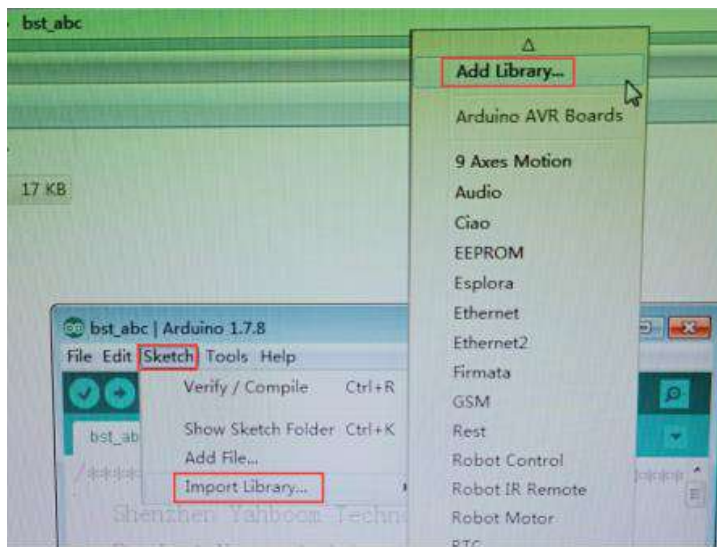


3. You need to copy this file into the libraries folder in the Arduino installation path.

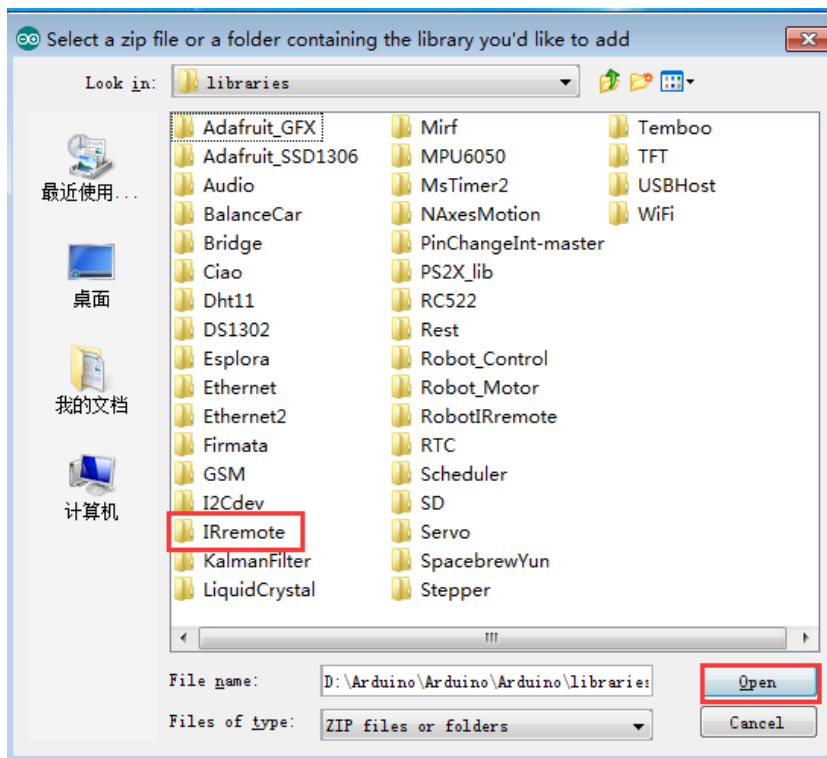
As shown in the figure below.



4.You need to open Arduino IDE and click 【Sketch】 --- 【Import library】 --- 【Add library】 . As shown in the figure below.



5.You need to add 【IRremote】 to here. As shown in the figure below.



6.After the addition is completed, the words “Library added to your libraries.” will appear in the lower right corner of the Arduino IDE. As shown in the figure below.



```

pinMode(DHT11PIN, OUTPUT);
Serial.begin(9600);
}

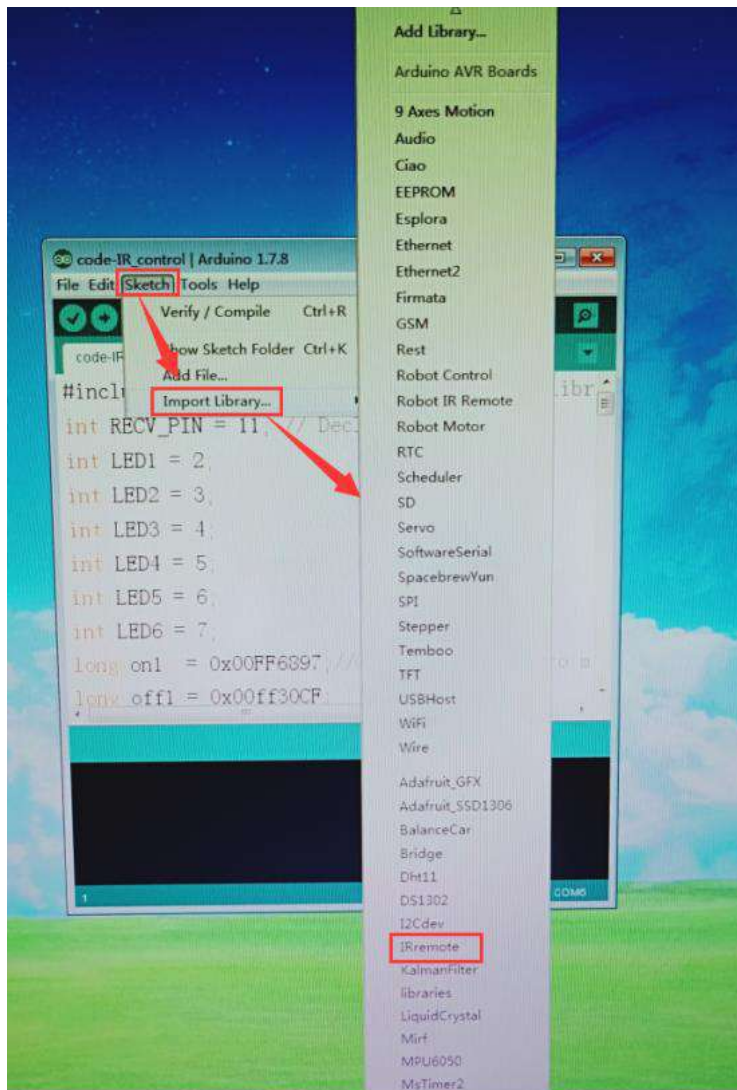
void loop() {
  int chk = DHT11.read(DHT11PIN);
  Serial.print("Tep: ");
  Serial.print((float)DHT11.temperature, 2);
  Serial.println("C");
  Serial.print("Hum: ");
  Serial.print((float)DHT11.humidity, 2);
  Serial.println("%");
}

```

Library added to your libraries. Check "Import library" menu

WARNING: Category 'Uncategorized' in library Rest is not valid. Setting to 'Uncategorized'

7. You can see these library files on the Arduino IDE. As shown in the figure below.



8. After completing the above steps, you can compile and upload this code successfully .



## 20-1602 display

### The purpose of the experiment:

In this experiment, we use Arduino UNO to directly drive 1602 display letters.

Introduction of 1602 :

The actual object is shown below.



### Main specification of 1602LCD:

Display capacity: 16 x 2 characters;

Working current: 2.0mA

Operating voltage: 5.0v

Size of character: 2.95 \* 4.35 (W \* H) mm.

1602 possess 16 pins:

Pin 1: VSS is ground power

Pin 2: VDD is connected to 5V positive power supply

Pin 3: V0 is the LCD contrast adjustment pin, which can be adjusted by a 10K adjustable resistor.

Pin 4: RS is the register selection pin, data register is selected at high voltage and instruction register is selected at low voltage.

Pin 5: R/W is the signal line for reading and writing. Reading operation is carried out at high level and writing operation is carried out at low level.

Pin 6: E pin is the enable pin. When this pin changes from high level to low level, the LCD module executes the command.

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

Pin 15: power positive pole of backlight.

Pin 16: power negative pole of backlight.

List of components required for the experiment:

Arduino UNO board \*1

USB cable \*1

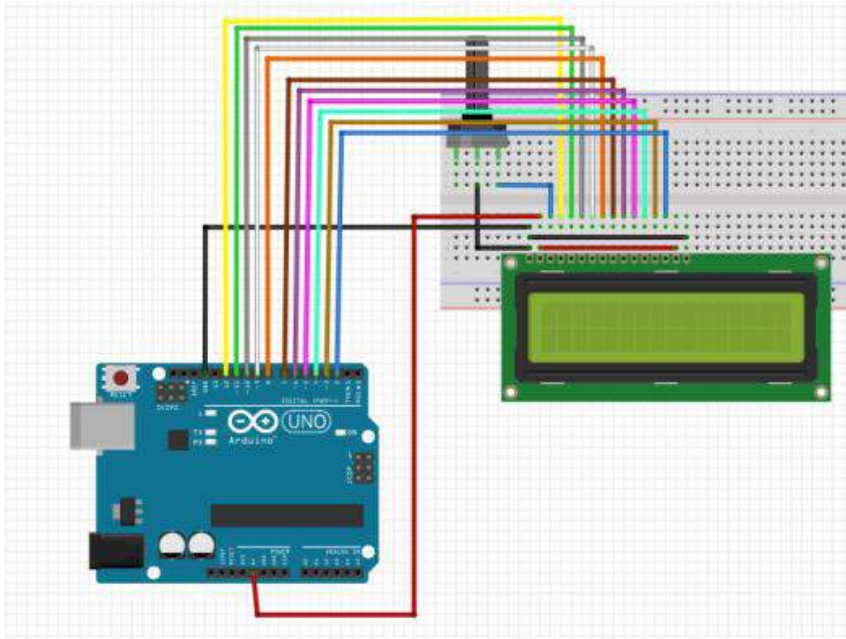
1602 \*1

Dupont line \*1 bunch

Breadboard \*1

### Actual object connection diagram :

We need to connect the circuit as shown in the figure below.



### Experimental code analysis:

```
#include <LiquidCrystal.h>
//Declaring the Arduino digital port connected to the 1602 LCD pin,
//8-wire or 4-wire data mode, either one
LiquidCrystal lcd(12,11,10,9,8,7,6,5,4,3,2);
//LiquidCrystal lcd(12,11,10,5,4,3,2);
int i;
void setup()
{
  lcd.begin(16,2);    //Initialization of 1602
                      //The 1602 LCD display range is defined as 2 lines and 16 columns
  characters
  while(1)
  {
    lcd.home();      //Moving the cursor back to the upper left corner,output from the
beginning
    lcd.print("Hello World");
    lcd.setCursor(0,1); //The cursor is positioned on line 1, column 0
    lcd.print("Welcome to Yahboom-Arduino");
    delay(500);
    for(i=0;i<3;i++)
    {
      lcd.noDisplay();
      delay(500);
      lcd.display();
      delay(500);
    }
    for(i=0;i<24;i++)
    {
      lcd.scrollDisplayLeft();
      delay(500);
    }
    lcd.clear();
```

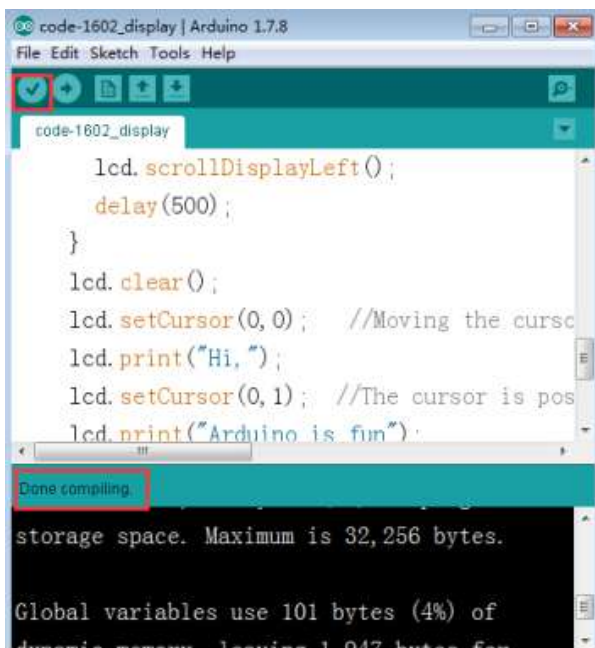
```

    lcd.setCursor(0,0); //Moving the cursor back to the upper left corner,output from the
beginning
    lcd.print("Hi,");
    lcd.setCursor(0,1); //The cursor is positioned on line 1, column 0
    lcd.print("Arduino is fun");
    delay(2000);
}
}
void loop()
{} //Initialization is complete and the main loop is not need to do anythings

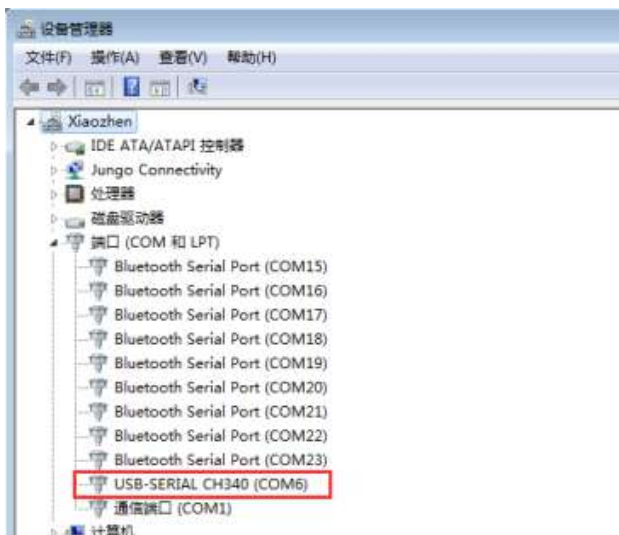
```

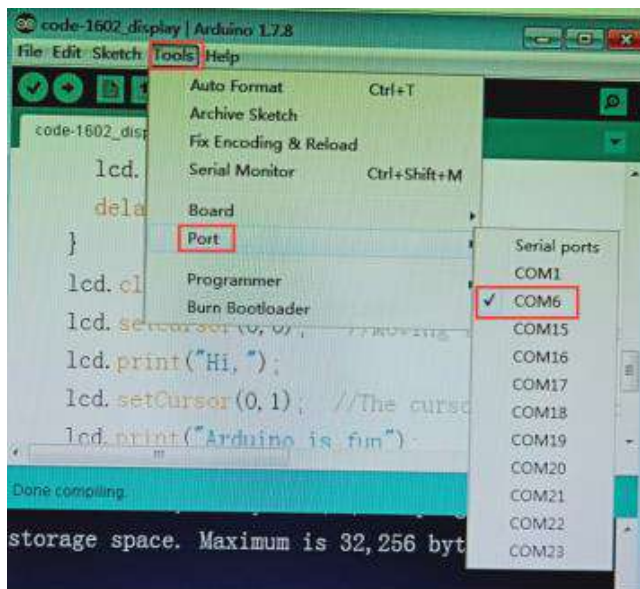
### Experimental steps:

1. We need to open the code for this experiment: code-1602\_display.ino, click “√” under the menu bar, compile the code, and wait for the words of Done compiling in the lower left corner, as shown in the following figure.

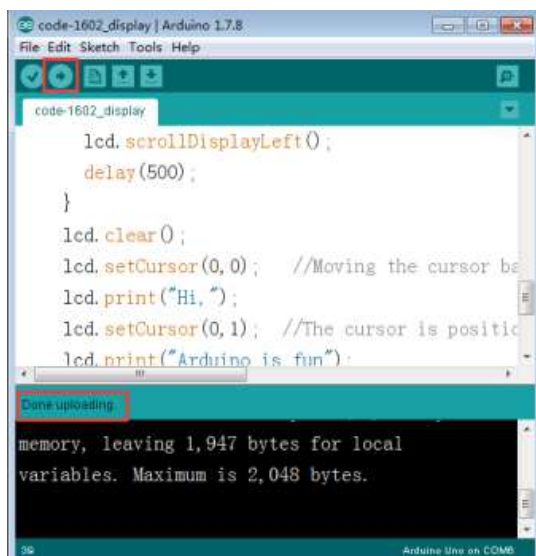


2. In the menu bar of Arduino IDE, you need to select the 【Tools】 --- 【Port】 --- select the port that the serial number displayed by the device manager just now. for example: COM6, as shown in the following figure.





3. After the selection is completed, you need to click “→” under the menu bar, and upload the program to the Arduino UNO board, when appears to Done uploading on the lower left corner , that means that the code has been successfully uploaded to the Arduino UNO board, as shown in the following figure.



4. After the code is uploaded. First, the 1602 screen will display “Hello World, Welcome to yahboom-arduino” and flash three times. Then, “Hello World, Welcome to yahboom-arduino,” is displayed from the right to the left. Next, “Hi, Arduino is fun.” is displayed on the 1602. Finally, it clears the screen, and continues the endless cycle.

As shown in the figure below.

