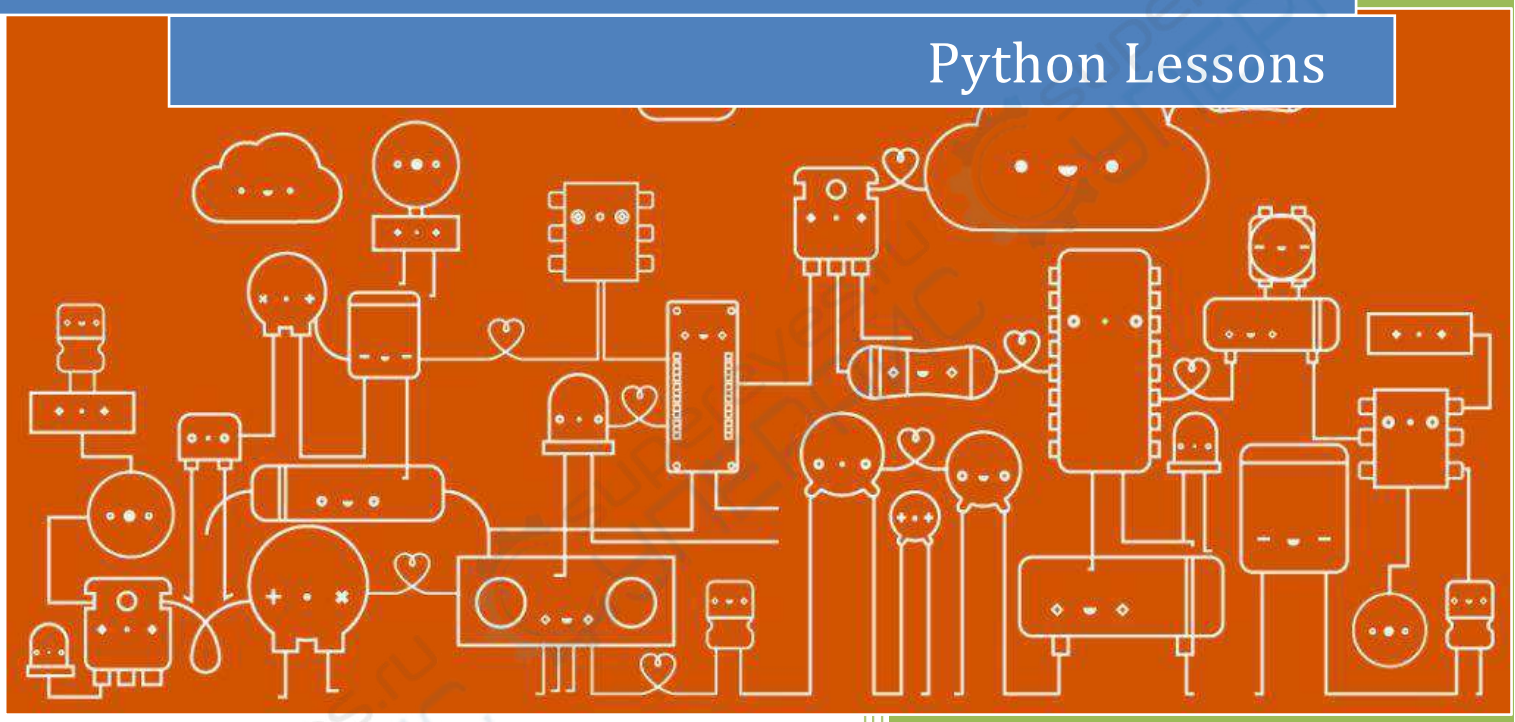
2019

Micro:bit Smart Robot Car

Python Lessons



Python programming

1-Hello,World!

Learning goals:

This lesson learns to scroll through the characters on a micro:bit dot matrix by Python programming.

Code :

```
from microbit import *
display.scroll(”Hello, World!”)
```

Programming and downloading:

1. You should open the Mu software, and enter the code in the edit window, as shown in Figure 1-1.

Note! All English and symbols should be entered in English, and the last line must be a space.

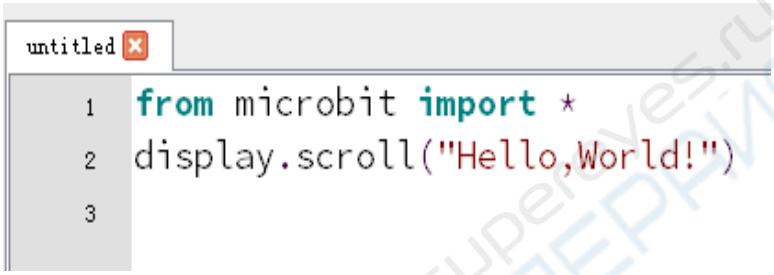


Figure 1-1

2. As shown in Figure 1-2, you need to click the Check button to check if our code has an error. If a line appears with a cursor or an underscore, the program indicating this line is wrong.

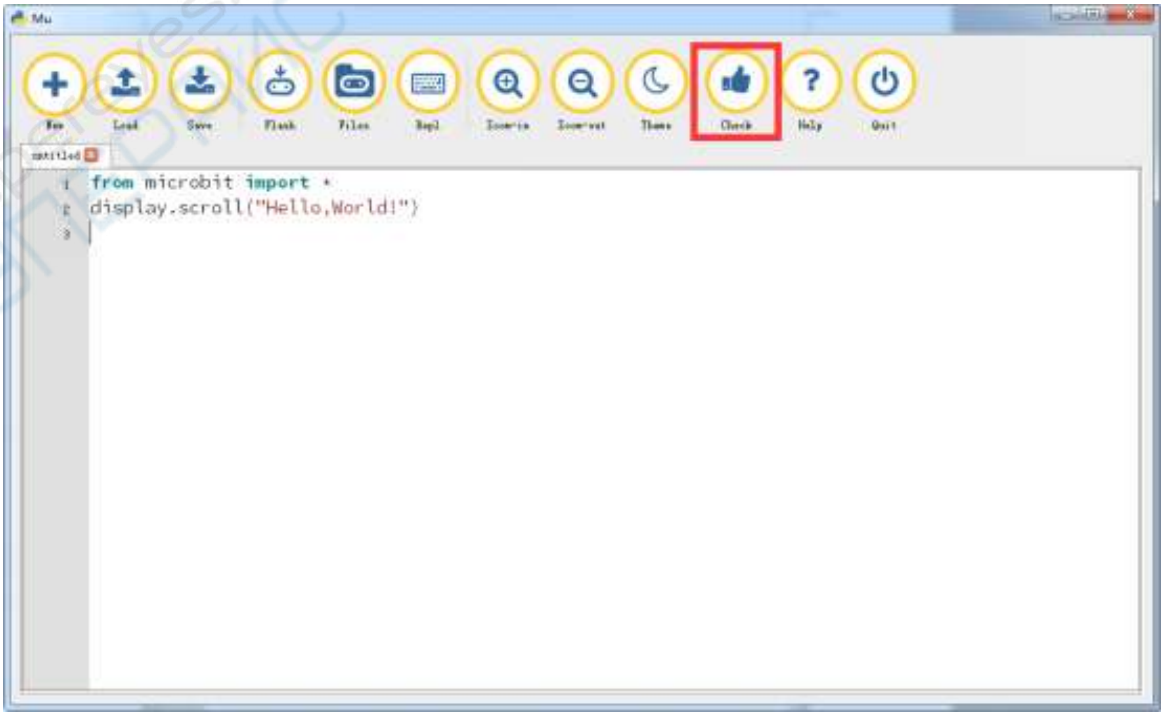


Figure 1-2

3. You need to connect the micro data cable to micro:bit and the computer, then click the Flash button to download the program to micro:bit as shown in Figure 1-3.

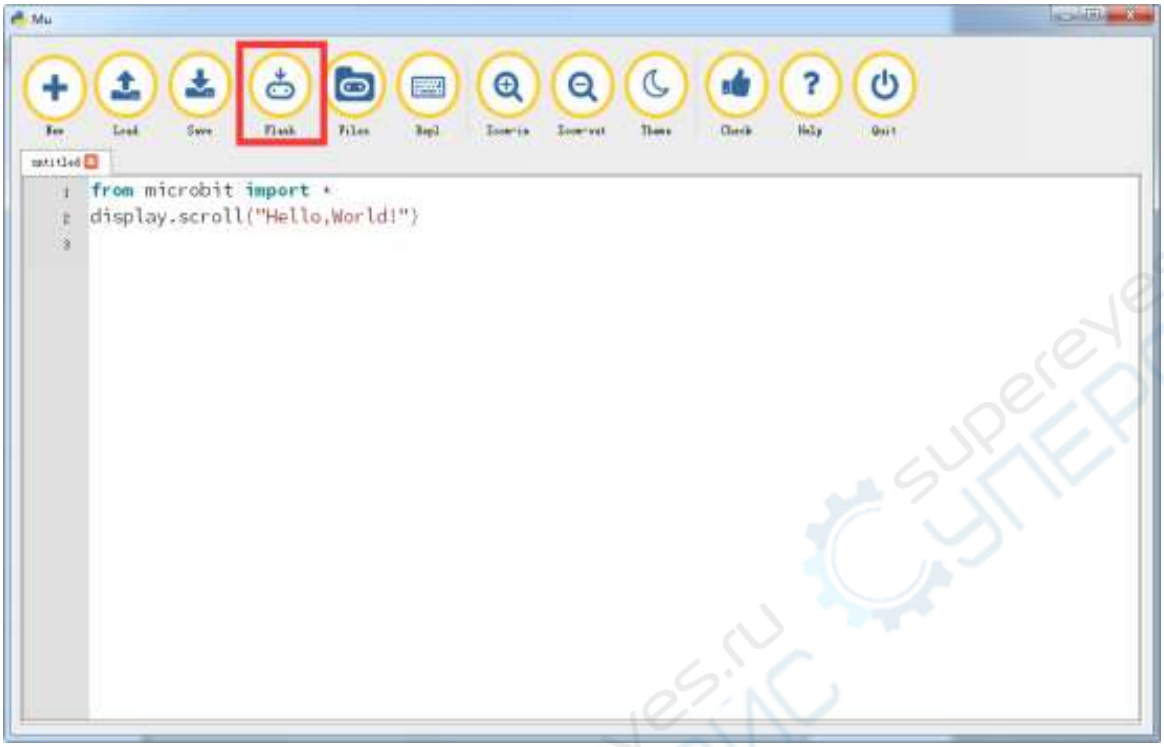


Figure 1-3

4.After the download is successful, you can see that the micro:bit dot matrix slowly moving to the left , “Hello, World!”, as shown in Figure 1-4 and Figure 1-5.

Figure 1-4

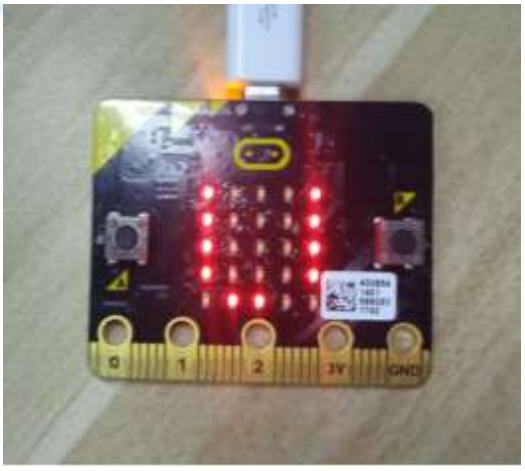


Figure 1-5

2-Display built-in image

Learning goals:

This lesson learns to display image on a micro:bit dot matrix by Python programming. For example: heart.

Code:

```
from microbit import *
display.show(Image.HEART)
```

Below is a list of built-in images:

• Image.HEART
• Image.HEART_SMALL
• Image.HAPPY
• Image.SMILE
• Image.SAD
• Image.CONFUSED
• Image.ANGRY
• Image.ASLEEP
• Image.SURPRISED
• Image.SILLY
• Image.FABULOUS
• Image.MEH
• Image.YES
• Image.NO
•Image.CLOCK12, Image.CLOCK11, Image.CLOCK10, Image.CLOCK9, Image.CLOCK8, Image.CLOCK7, Image.CLOCK6, Image.CLOCK5, Image.CLOCK4, Image.CLOCK3, Image.CLOCK2,Image.CLOCK1
•Image.ARROW_N, Image.ARROW_NE, Image.ARROW_E, Image.ARROW_SE, Image.ARROW_S, Image.ARROW_SW, Image.ARROW_W, Image.ARROW_NW
• Image.TRIANGLE
• Image.TRIANGLE_LEFT
• Image.CHESSBOARD
• Image.DIAMOND
• Image.DIAMOND_SMALL
• Image.SQUARE
• Image.SQUARE_SMALL
• Image.RABBIT
• Image.COW
• Image.MUSIC_CROTCHET
• Image.MUSIC_QUAVER
• Image.MUSIC_QUAVERS
• Image.PITCHFORK
• Image.XMAS
• Image.PACMAN
• Image.TARGET

- Image.TSHIRT
- Image.ROLLERSKATE
- Image.DUCK
- Image.HOUSE
- Image.TORTOISE
- Image.BUTTERFLY
- Image.STICKFIGURE
- Image.GHOST
- Image.SWORD
- Image.GIRAFFE
- Image.SKULL
- Image.UMBRELLA
- Image.SNAKE
- Image.ALL_CLOCKS
- Image.ALL_ARROWS

Programming and downloading:

1.You should open the Mu software, and enter the code in the edit window, , as shown in Figure 2-1.

Note! All English and symbols should be entered in English, and the last line must be a space.

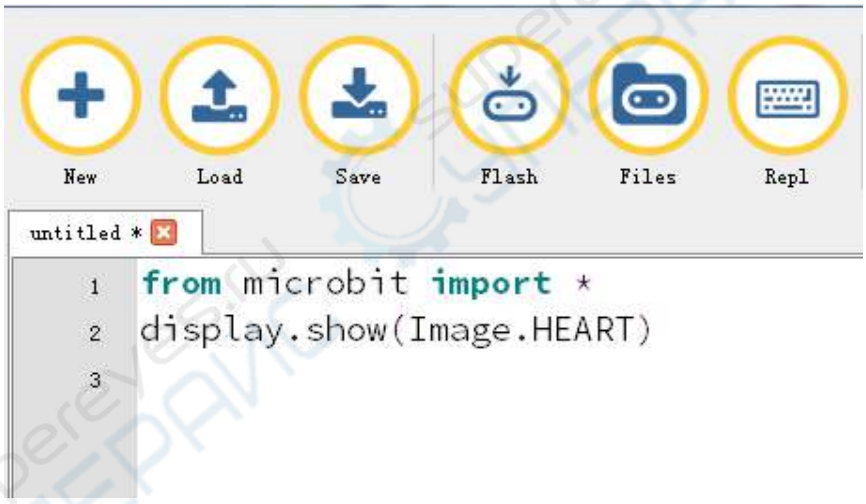


Figure 2-1

2.As shown in Figure 2-2, you need to click the Check button to check if our code has an error. If a line appears with a cursor or an underscore, the program indicating this line is wrong.

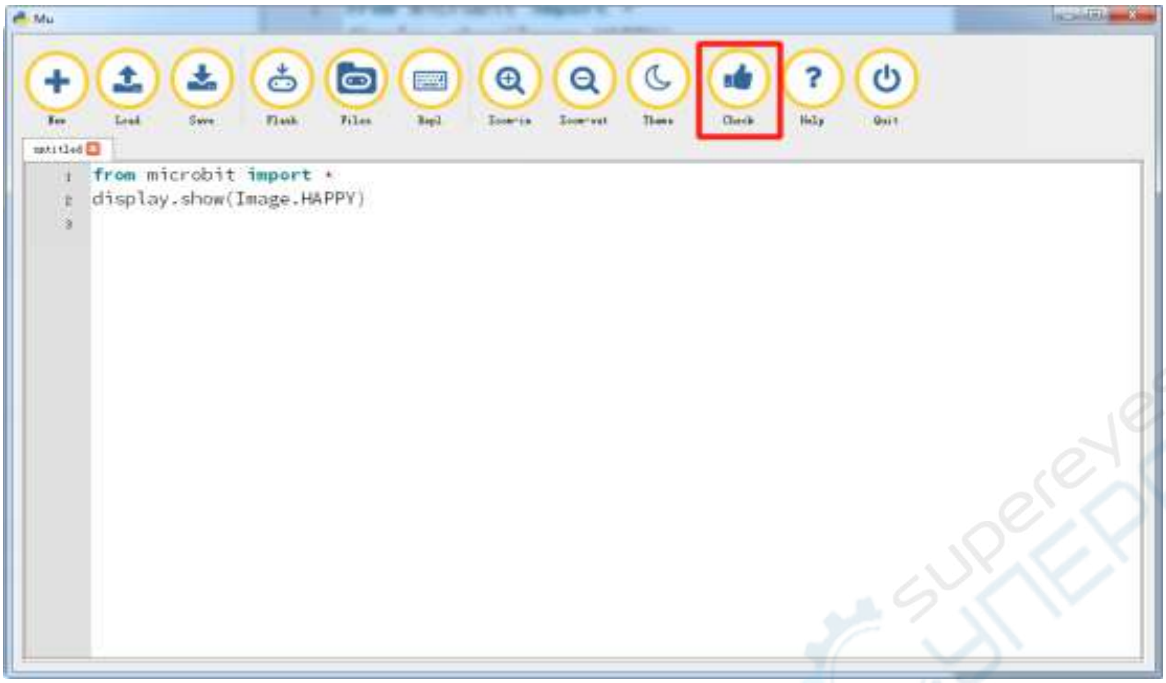


Figure 2-2

3. You need to connect the micro data cable to micro:bit and the computer, then click the Flash button to download the program to micro:bit as shown in Figure 2-3.

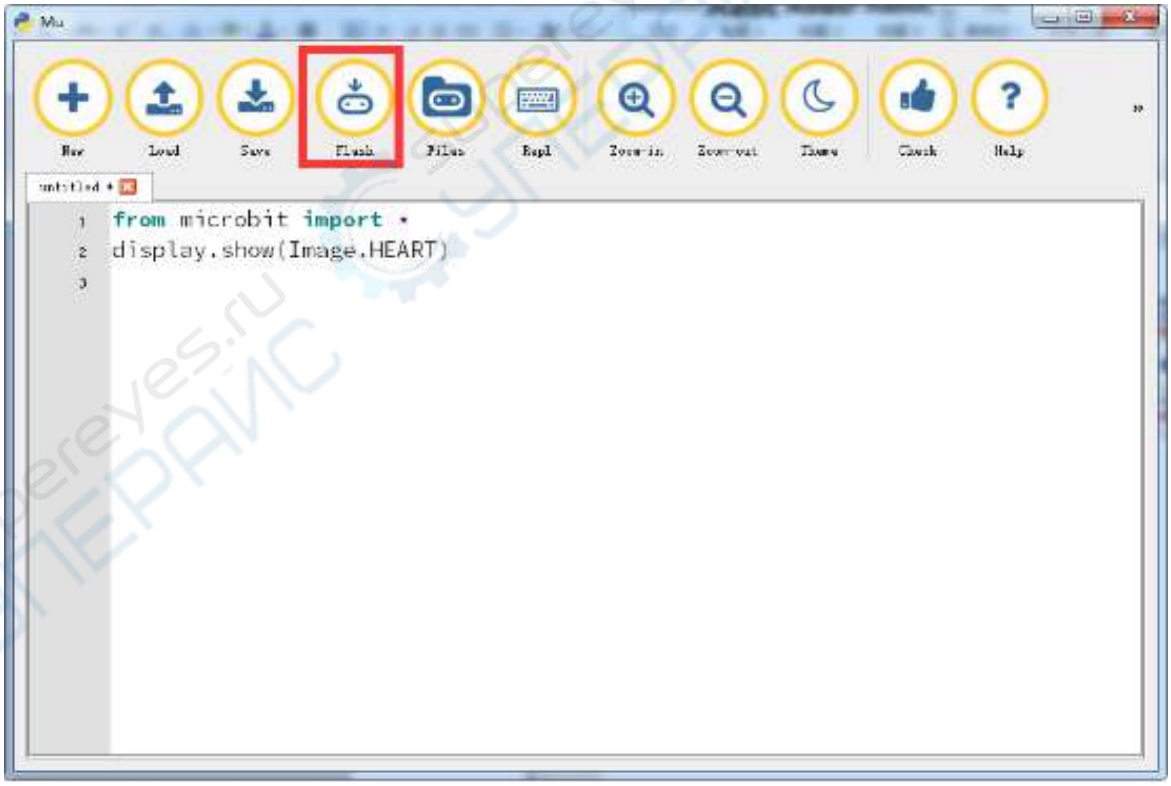


Figure 2-3

4.After the download is successful, you can see that a heart on the micro:bit dot matrix .as shown in Figure 2-4.

Figure 2-4.

3-Display custom

Learning goals:

This lesson learns to display custom image on a micro:bit dot matrix by Python programming. For example: boat.

Code1:

```
from microbit import *

boat = Image(”49494:”

        “49494:”

        “49494:”

        “99999:”

        “49994”)

display.show(boat)
```

Code 2:

```
from microbit import *

boat = Image(”49494:49494:49494:99999:49994”)

display.show(boat)
```

Note:
1 - The capital letter / lowercase letters must be distinguished!
2 - Correct spelling!
3 - Keywords such as # need a space between the content.
4 - The program ends with a blank program.
5 - The block body (such as the body of the while is marked by indentation), compared to the C language, Python completely eliminates the braces (along with the semicolon of the suffix), and uses the indentation structure to represent the relationship. You can only use the Tab key (tabulation key) for indentation.

Micro:bit possess a dot matrix of 5*5 LEDs, and each LED brightness on the dot matrix can be set to a value from 0 to 9. If the brightness of an LED is set to 0, then it goes out. If its brightness is set to 9, it is at the brightest level. Using this feature, we can display a custom image on the micro:bit dot matrix. The code implementation of our class shows a boat on the micro:bit dot matrix. The background brightness value is 5, the hull part. The brightness should be brighter, the brightness value is 9, you can set other brightness values to display different patterns.

Programming and downloading:

1.You should open the Mu software, and enter the code in the edit window, ,as shown in Figure 3-1.

Note! All English and symbols should be entered in English, and the last line must be a space.

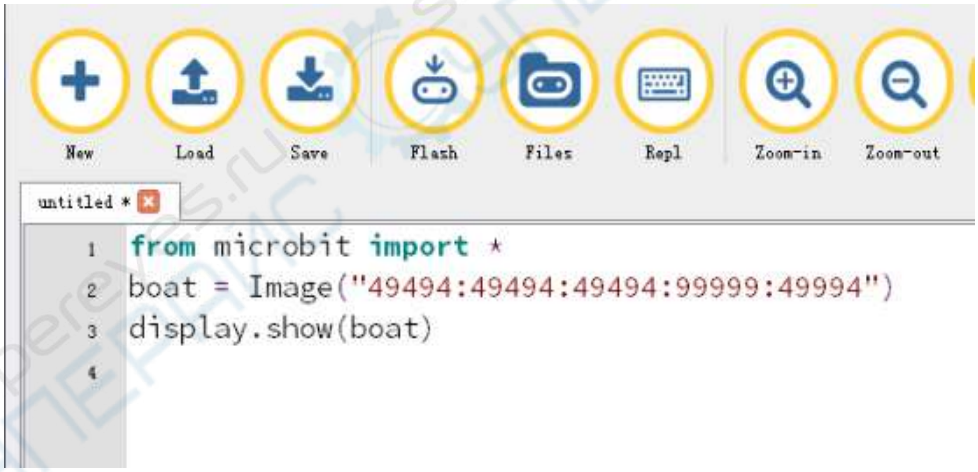


Figure 3-1

2.As shown in Figure 3-2, you need to click the Check button to check if our code has an error. If a line appears with a cursor or an underscore, the program indicating this line is wrong.

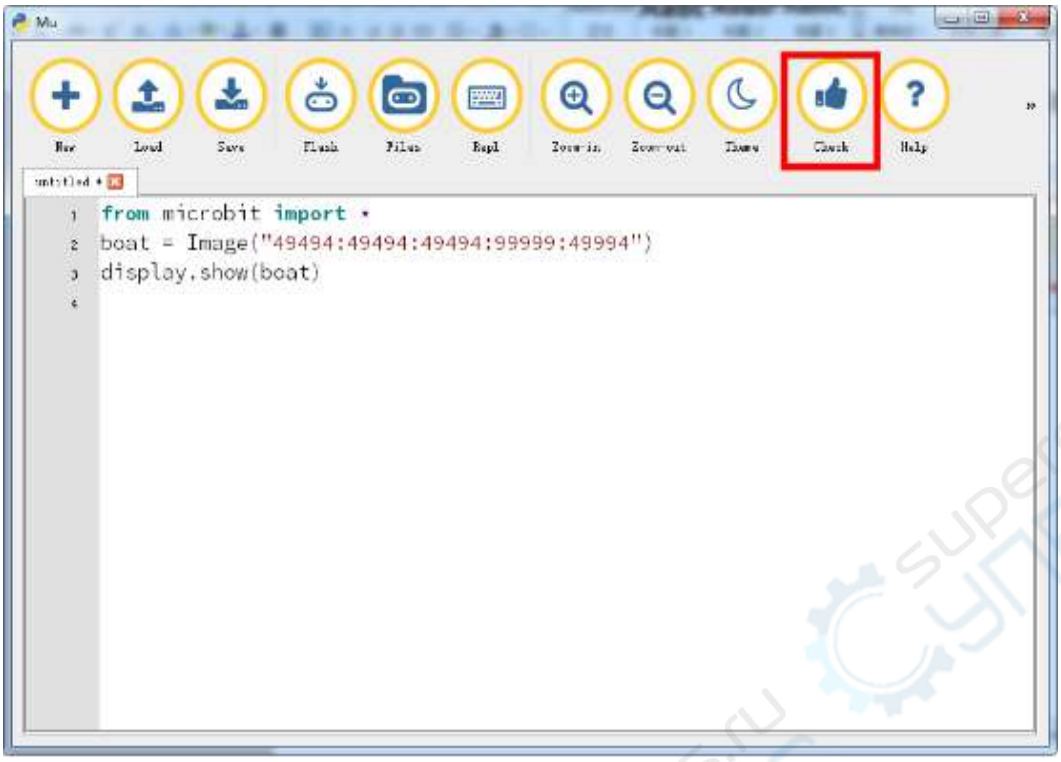


Figure 3-2

3.You need to connect the micro data cable to micro:bit and the computer, then click the Flash button to download the program to micro:bit as shown in Figure 3-3.

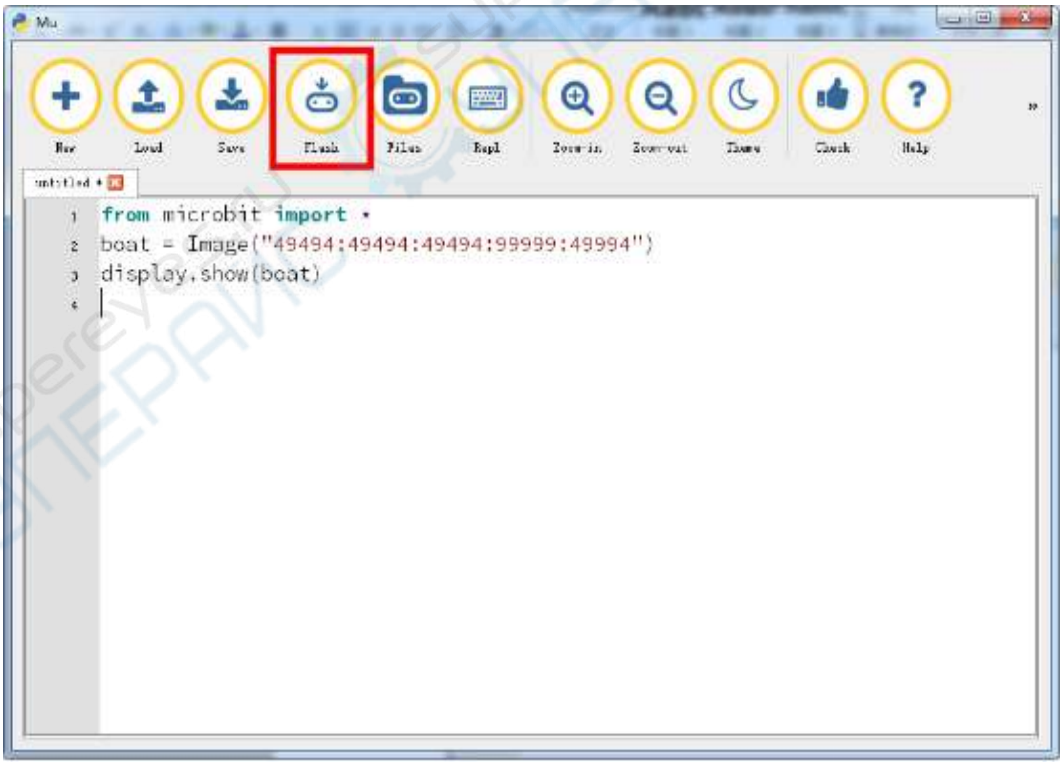


Figure3-3

4.After the download is successful, you can see that a boat on the micro:bit dot matrix . The brightness of the background is weaker than the brightness of the hull, as shown in Figure 3-4.

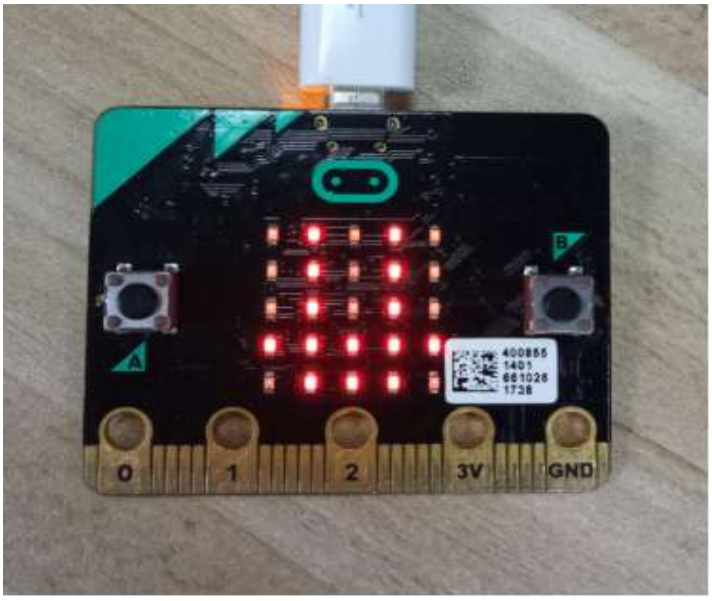


Figure 3-4

4-Display custom animation

This lesson learns to display custom animation on a micro:bit dot matrix by Python programming. For example: from smile to sadness and then to anger.

Code:

```
from microbit import *
while True:
    face1 = Image("00000:09090:00000:90009:09990")
    face2 = Image("00000:09090:00000:99999:00000")
    face3 =  Image("00000:09090:00000:09990:90009")
    face4 = Image("90009:99099:00000:09990:90009")
    face5 = Image("00000:00000:00000:00000:00000")
    all_faces = [face1, face2, face3, face4, face5, ]
    display.show(all_faces, delay=200)
```

Programming and downloading:

1.You should open the Mu software, and enter the code in the edit window, , as shown in Figure 4-1.

Note! All English and symbols should be entered in English, and the last line must be a space.

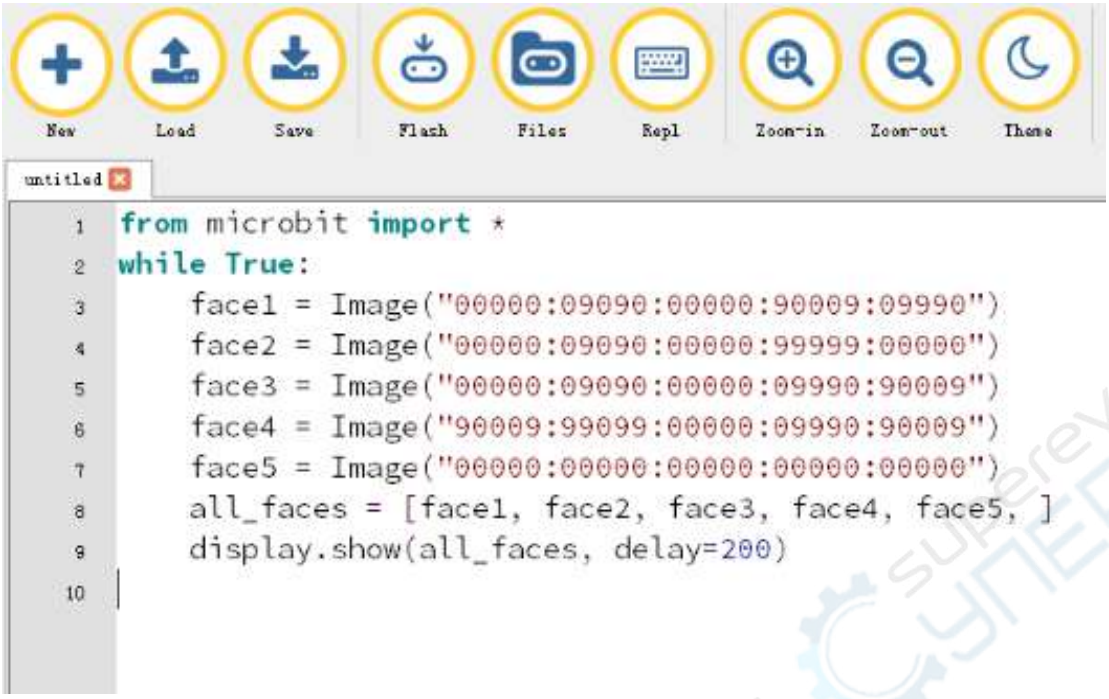


Figure 4-1

2.As shown in Figure 4-2, you need to click the Check button to check if our code has an error. If a line appears with a cursor or an underscore, the program indicating this line is wrong.

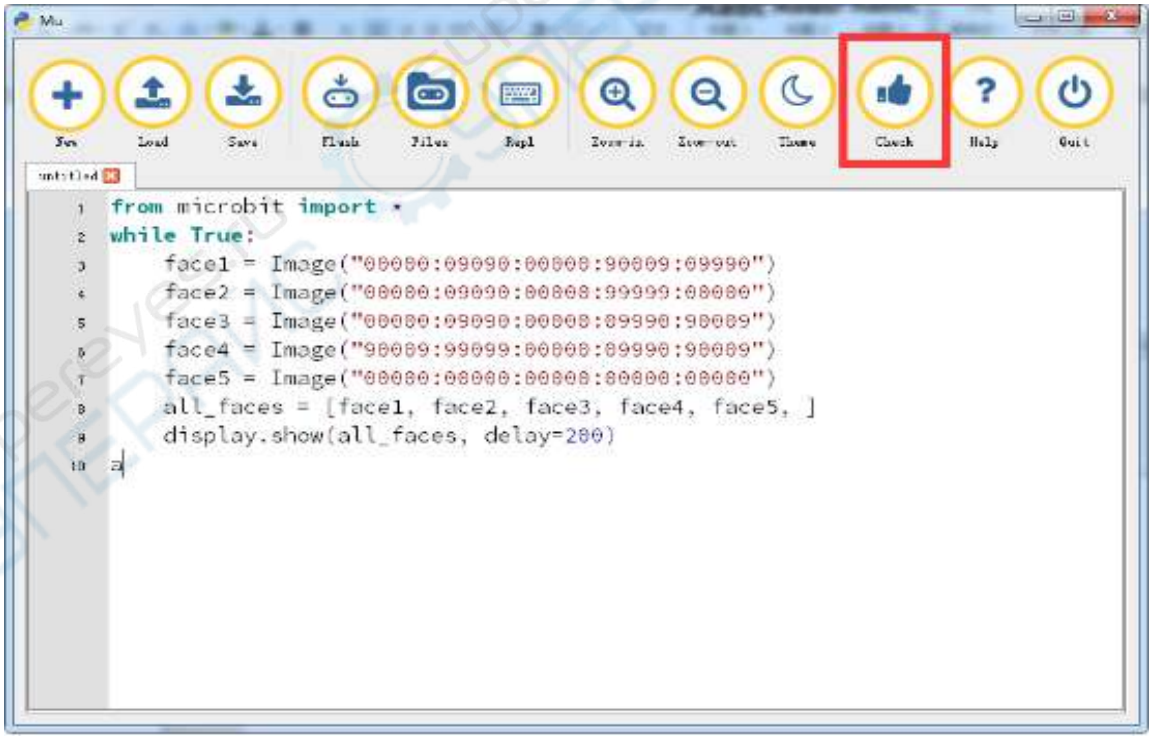


Figure 4-2

3.You need to connect the micro data cable to micro:bit and the computer, then click the Flash button to download the program to micro:bit as shown in Figure 4-3.

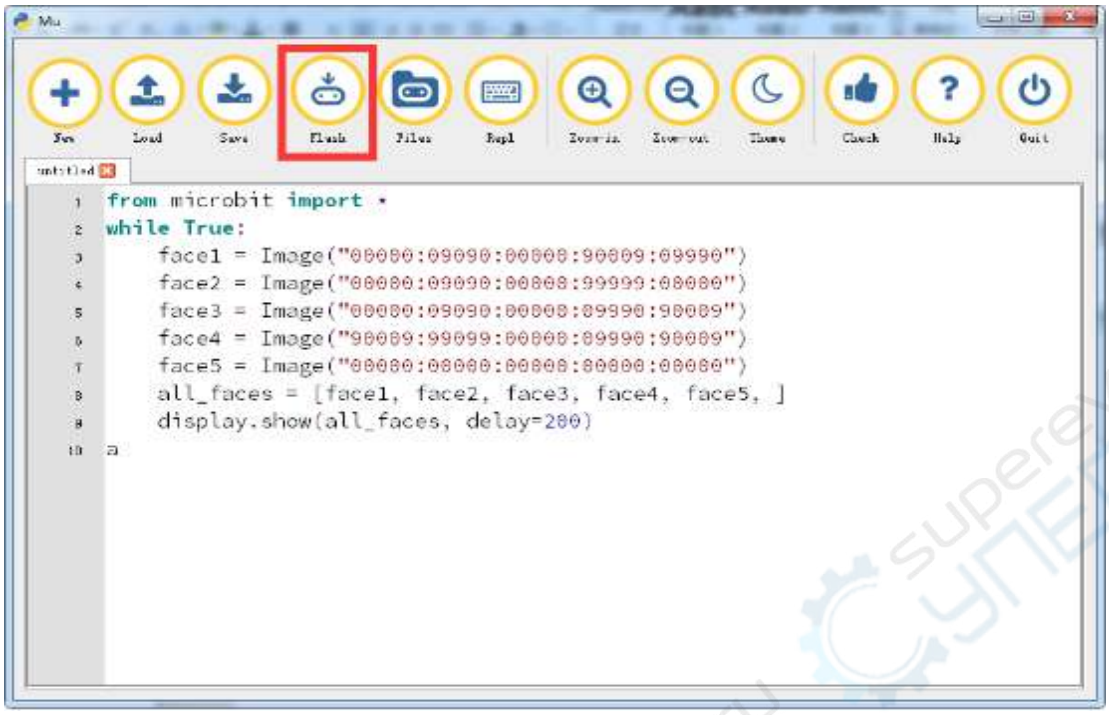


Figure 4-3

4. After the download is successful, you can observe the animation of the expression change on the micro:bit dot matrix, as shown in Figure 4-4 to Figure 4-7.

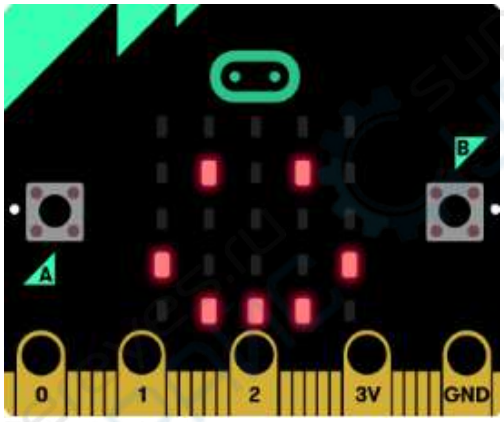


Figure 4-4

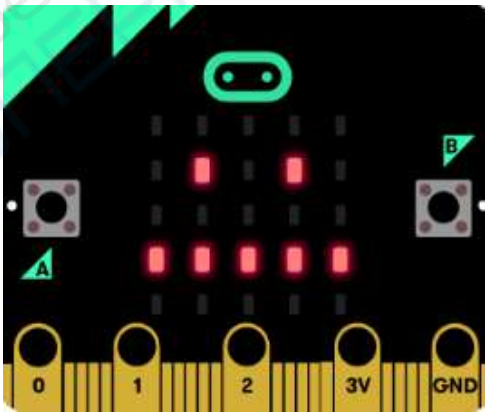


Figure 4-5

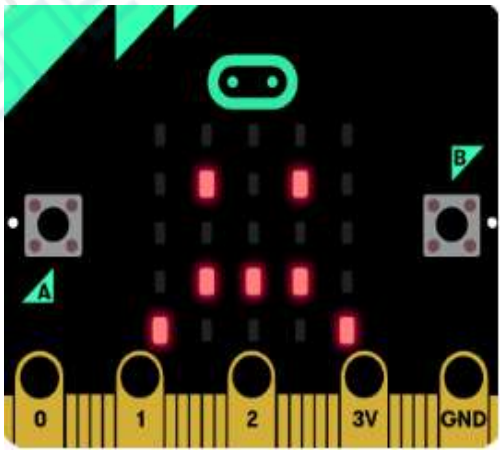


Figure 4-6

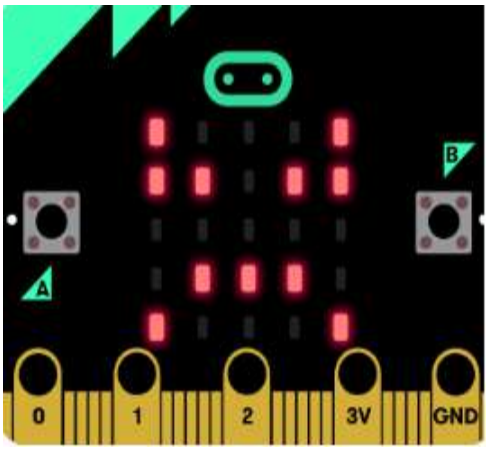


Figure 4-7

5-See who is pressing fast

Learning goals:

In this lesson, we will make a very simple micro:bit game. When we press the A button, the micro:bit dot matrix will display an arrow pointing to the A button; when we press the B button, the micro:bit will display an arrow pointing to the B button; if no button is pressed, the micro:bit It shows a heart.

Code:

```
from microbit import *

while True:

    if button_a.is_pressed():

        display.show(Image.ARROW_W)

    elif button_b.is_pressed():

        display.show(Image.ARROW_E)

    else:

        display.show(Image. HEART)

        display.clear()
```

Programming and downloading:

1.You should open the Mu software, and enter the code in the edit window, , as shown in Figure 5-1.

Note! All English and symbols should be entered in English, and the last line must be a space.

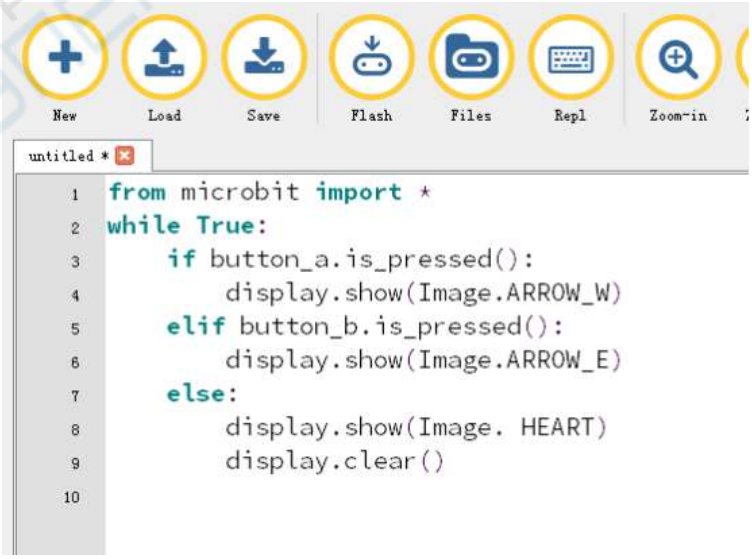


Figure 5-1

2.As shown in Figure 5-2, you need to click the Check button to check if our code has an error. If a line appears with a cursor or an underscore, the program indicating this line is wrong.

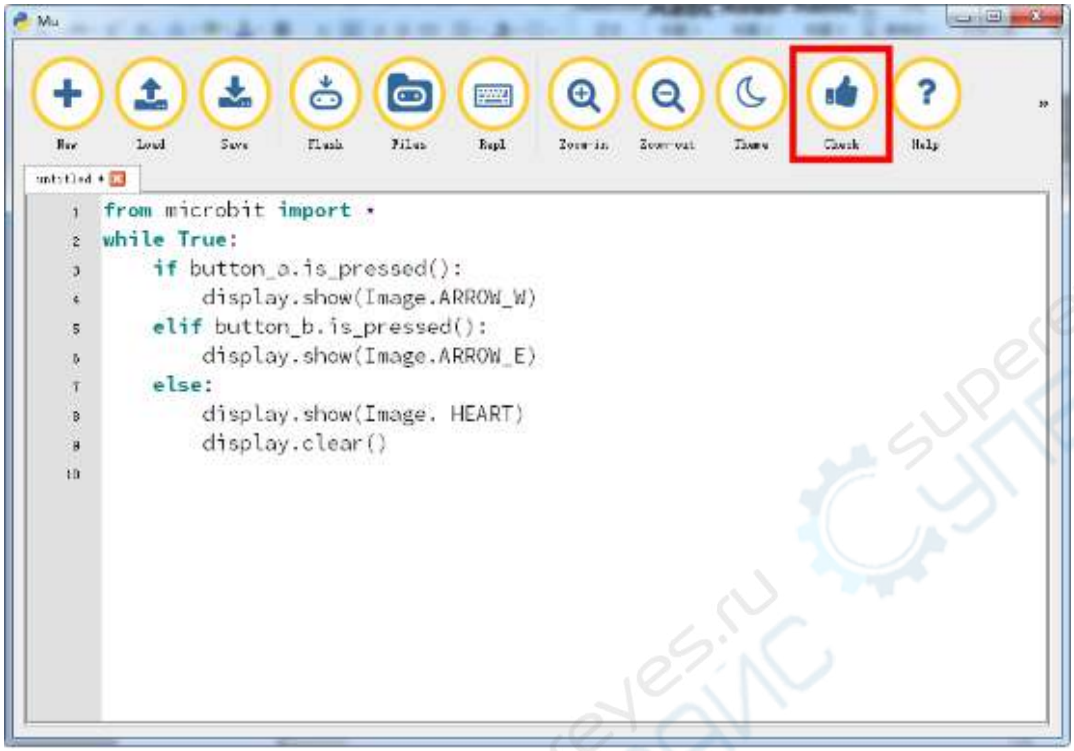


Figure 5-2

3.You need to connect the micro data cable to micro:bit and the computer, then click the Flash button to download the program to micro:bit as shown in Figure 5-3.

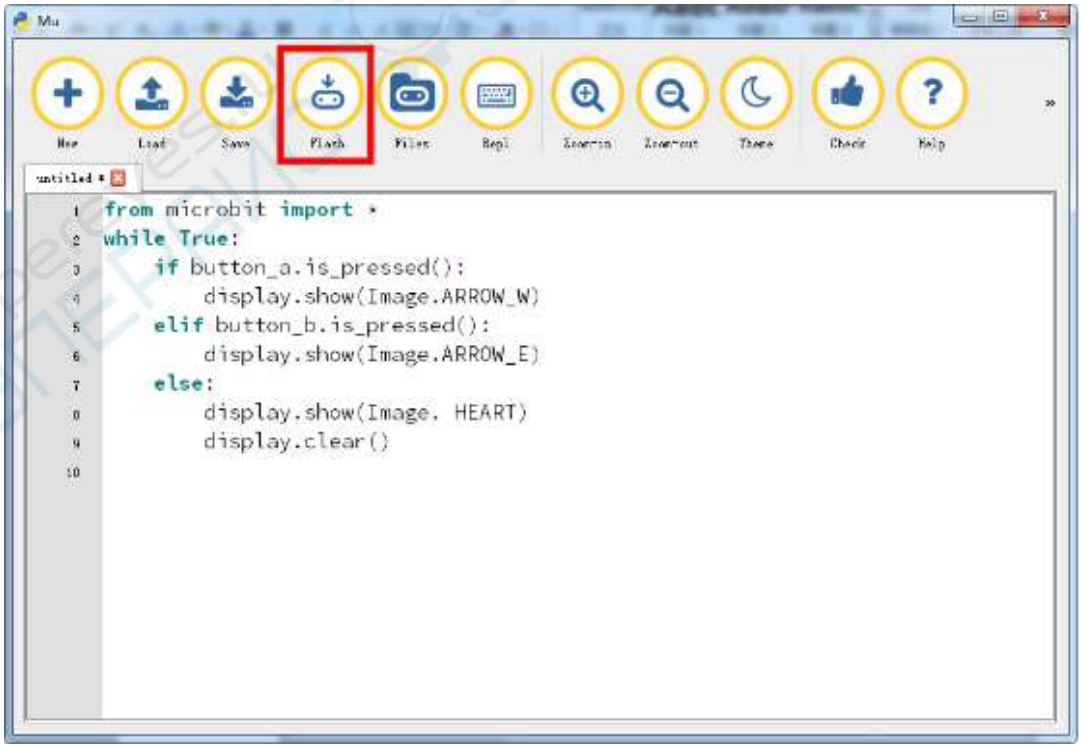


Figure 5-3

4.After the download is successful, we can see that the micro:bit shows a heart,as shown in Figure 5-4.When we press the A button, the micro:bit dot matrix will display an arrow pointing to the A button, as shown in Figure 5-5; when we press the B button, the micro:bit will display an arrow pointing to the B button,as shown in Figure 5-6.

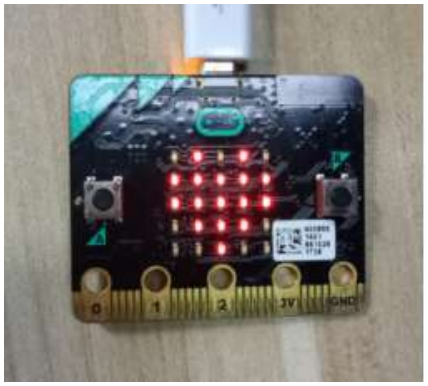


Figure 5-4

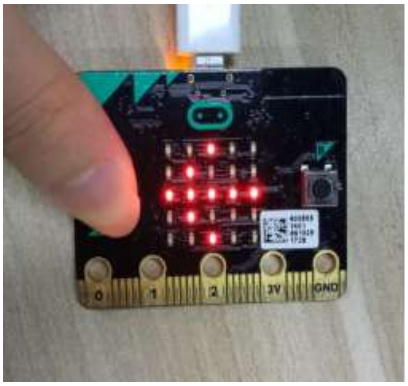


Figure 5-5

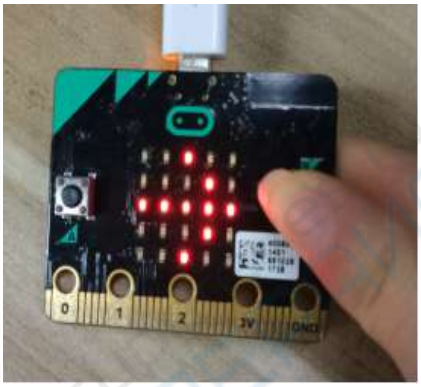


Figure 5-6

6-Sing a song

Learning goals:

In this lesson, you can use the micro:bit robot to play music, the robot sings a happy birthday song, and the dot matrix displays a buzzer pattern.

Code:

```
from microbit import *

import music

boat = Image("00090:90990:99990:90990:00090")

display.show(boat)

music.play(music.BIRTHDAY)
```

Below is a complete list of melody:

- music.DADADADUM
- music.ENTERTAINER
- music.PRELUDE
- music.ODE
- music.NYAN
- music.RINGTONE
- music.FUNK
- music.BLUES
- music.BIRTHDAY
- music.WEDDING

- music.FUNERAL
- music.PUNCHLINE
- music.PYTHON
- music.BADDY
- music.CHASE
- music.BA_DING
- music.WAWAWAWAA
- music.JUMP_UP
- music.JUMP_DOWN
- music.POWER_UP
- music.POWER_DOWN

Programming and downloading:

1.You should open the Mu software, and enter the code in the edit window, , as shown in Figure 6-1.

Note! All English and symbols should be entered in English, and the last line must be a space.

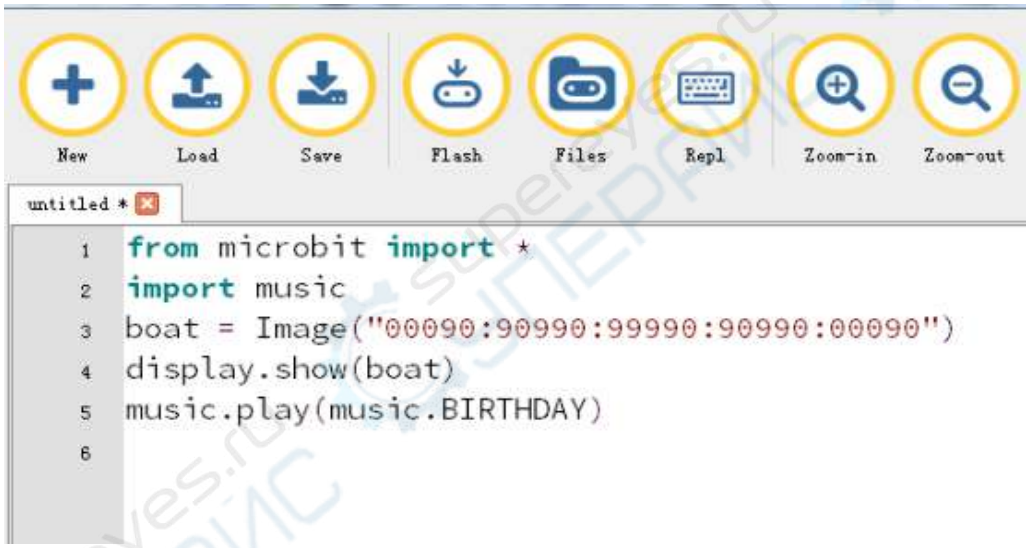


Figure 6-1

2.As shown in Figure 6-2, you need to click the Check button to check if our code has an error. If a line appears with a cursor or an underscore, the program indicating this line is wrong.

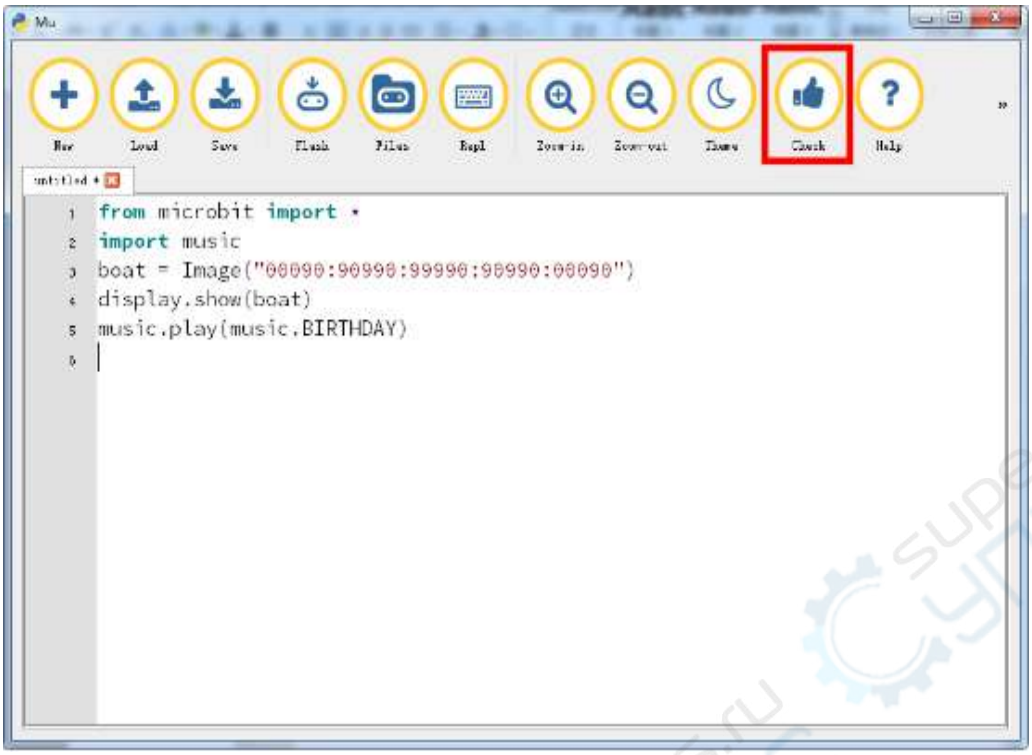


Figure 6-2

3.You need to connect the micro data cable to micro:bit and the computer, then click the Flash button to download the program to micro:bit as shown in Figure 6-3.

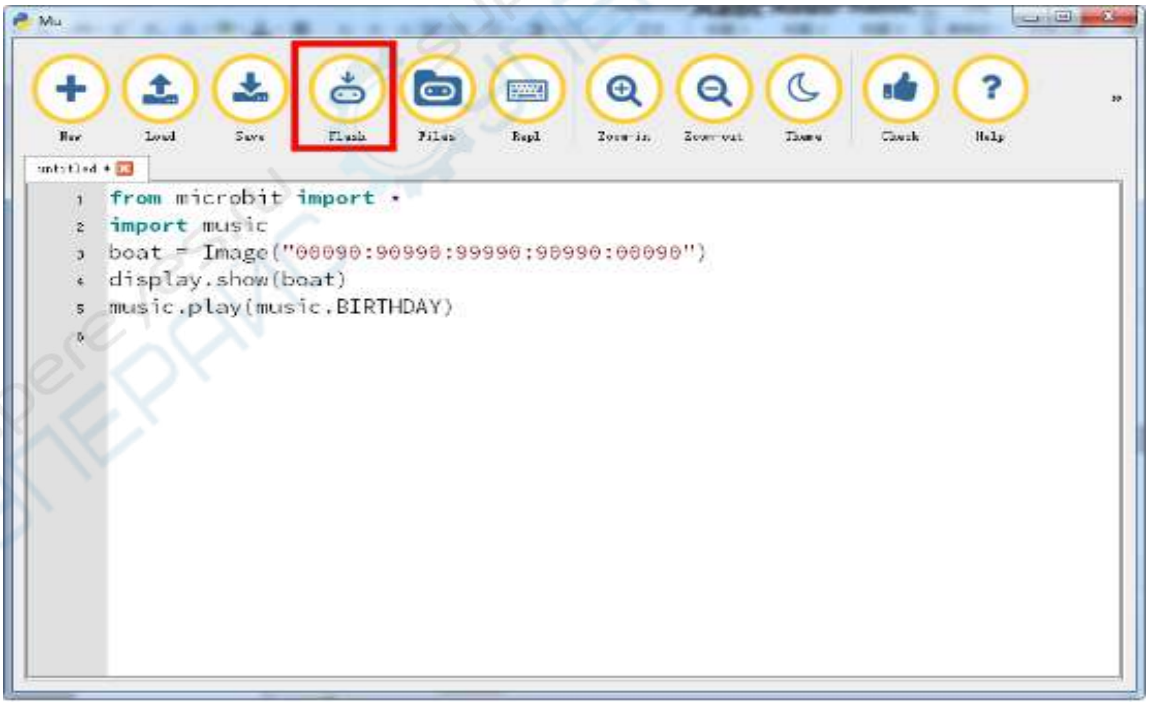


Figure 6-3

4.After downloading the program, you can hear Micro:bit playing Happy Birthday and a buzzer on the micro:bit dot matrix. as shown in Figure 6-4.

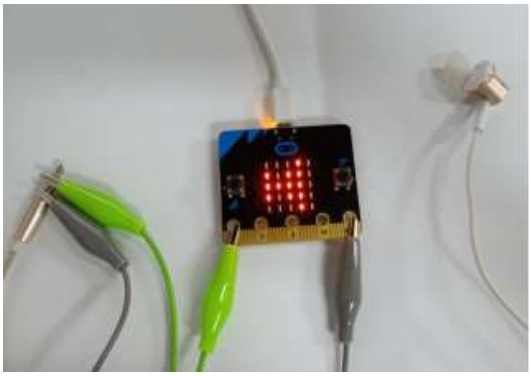


Figure 6-4

7-Play the custom music Painters

Learning goals:

In this lesson, you will learn how to play the music “The Painter.”

Code:

```
from microbit import *

import music

display.show(Image.MUSIC_QUAVER)

tune = [”G4:2”, “E4:2”, “G4:2”, “E4:2”, “G4:2”, “E4:2”, “C4:4”, “D4:2”, “F4:2”,

    “E4:2”, “D4:2”, “G4:4”, “E1:4”, “G4:2”, “E4:2”, “G4:2”, “E4:2”, “G4:2”,

    “E4:2”, “C4:4”, “D4:2”, “F4:2”, “E4:2”, “D4:2”, “C4:4”, “E1:4”, “D4:2”,

    “D4:2”, “F4:2”, “F4:2”, “E4:2”, “C4:2”, “G4:4”, “D4:2”, “F4:2”, “E4:2”,

    “D4:2”, “G4:4”, “E1:4”, “G4:2”, “E4:2”, “G4:2”, “E4:2”, “G4:2”, “E4:2”,

    “C4:4”, “D4:2”, “F4:2”, “E4:2”, “D4:2”, “C4:4”]

music.play(tune)
```

Programming and downloading:

1.You should open the Mu software, and enter the code in the edit window, , as shown in Figure 7-1.

Note! All English and symbols should be entered in English, and the last line must be a space.

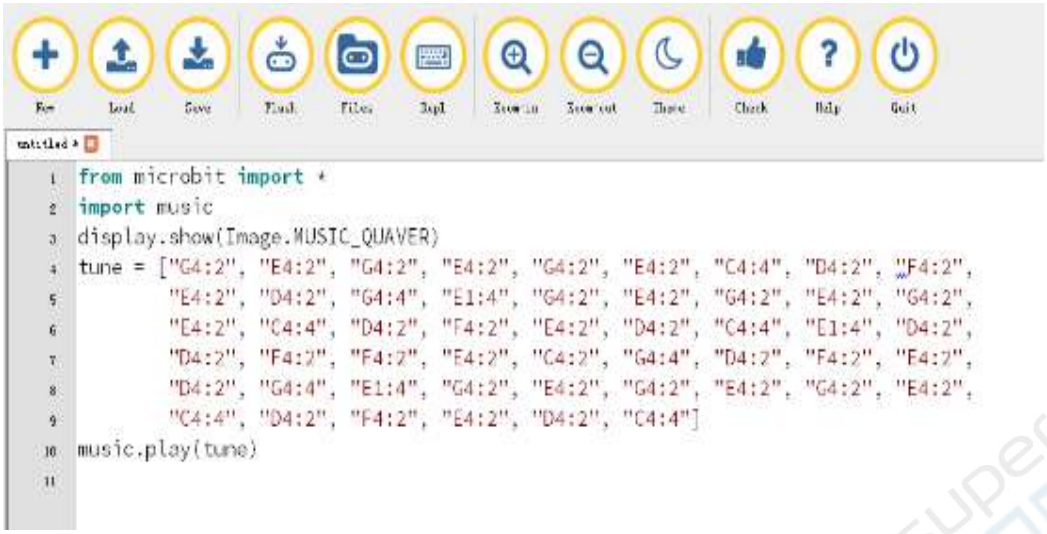


Figure 7-1

2. As shown in Figure 7-2, you need to click the Check button to check if our code has an error. If a line appears with a cursor or an underscore, the program indicating this line is wrong.

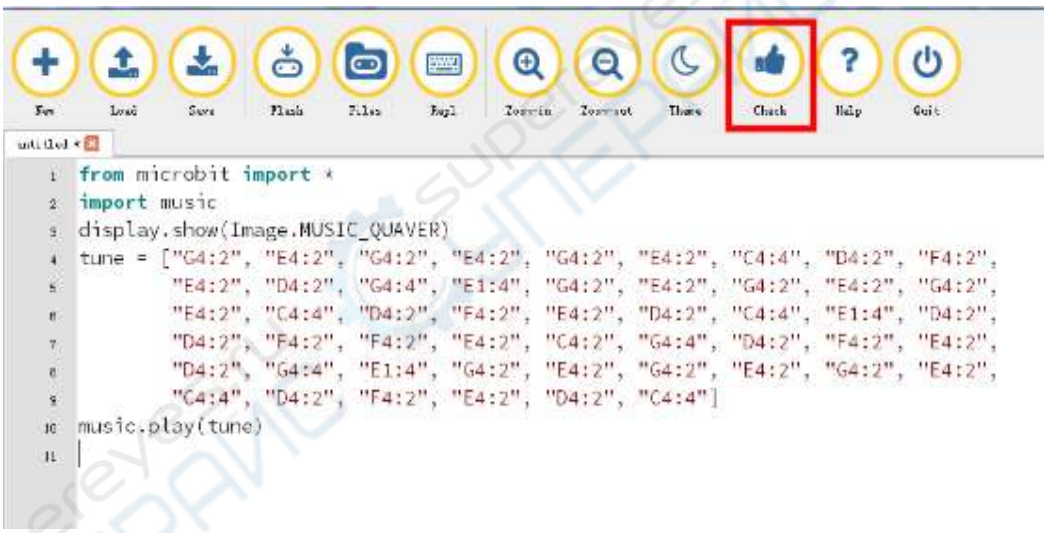


Figure 7-2

3.You need to connect the micro data cable to micro:bit and the computer, then click the Flash button to download the program to micro:bit as shown in Figure 7-3.

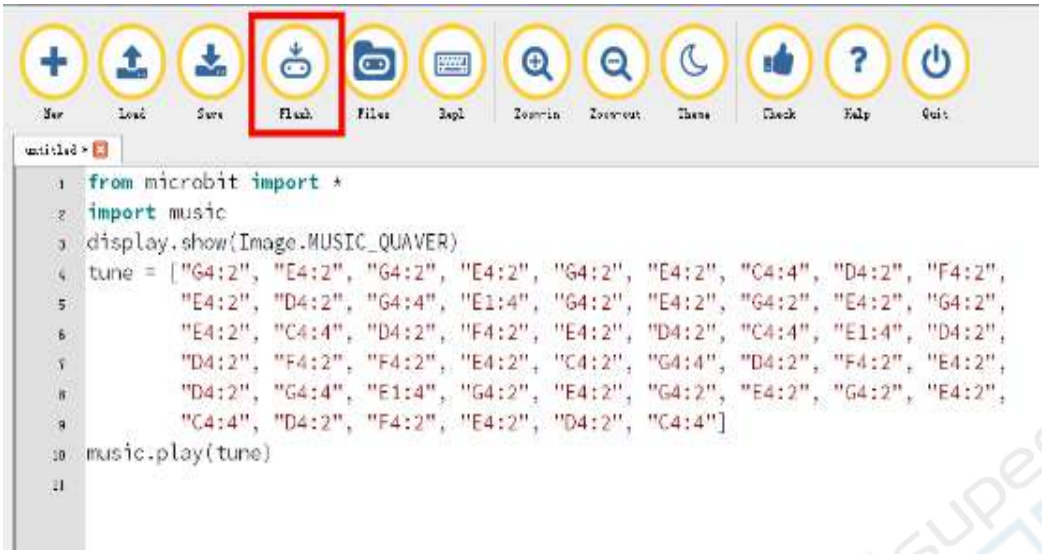


Figure 7-3

4.After downloading the program to micro:bit, you can hear t the music “Painter”, and there is a note on the dot matrix as shown in Figure 7-4.

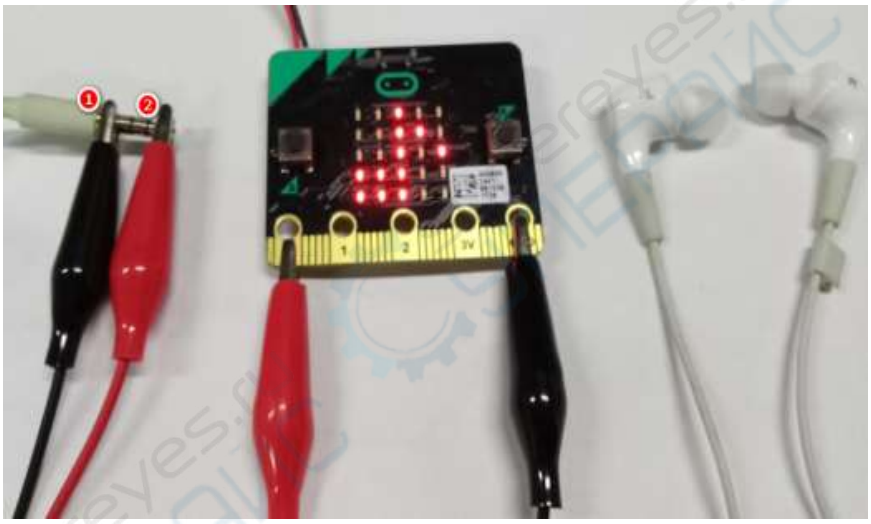


Figure 7-4

8-Dice game

Learning goals:

In this lesson, we will achieve shake a roll of micro:bit. There are number 1-6 randomly appearing on the dot matrix, which is exactly the same as playing the dice.

Code:

```
from microbit import *

import random

while True:

   gesture = accelerometer.current_gesture()

   if gesture == “shake”:

      display.show(str(random.randint(1, 6)))
```

Programming and downloading:

1.You should open the Mu software, and enter the code in the edit window, , as shown in Figure 8-1.

Note! All English and symbols should be entered in English, and the last line must be a space.

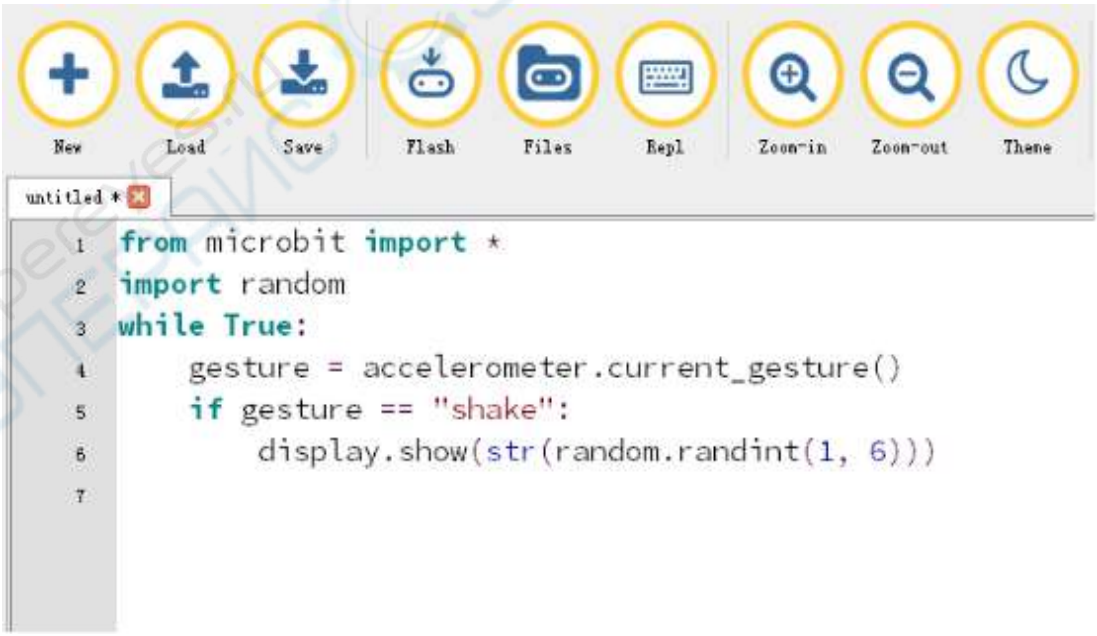


Figure 8-1

2.As shown in Figure 8-2, you need to click the Check button to check if our code has an error. If a line appears with a cursor or an underscore, the program indicating this line is wrong.

Figure 8-2

3.You need to connect the micro data cable to micro:bit and the computer, then click the Flash button to download the program to micro:bit as shown in Figure 8-3.

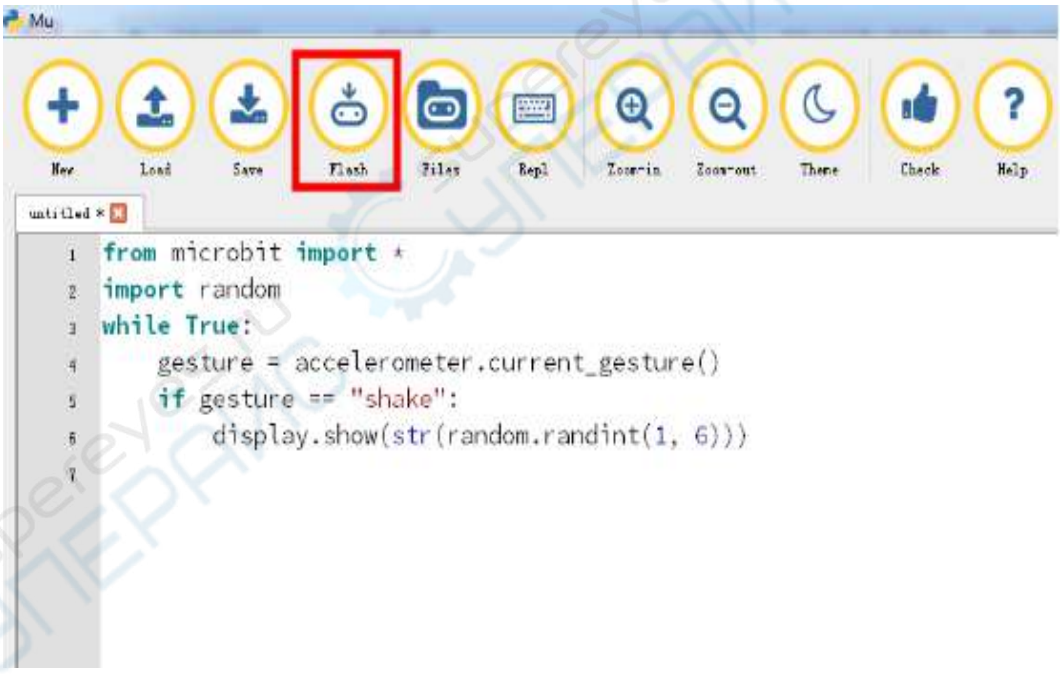


Figure 8-3

4.After downloading the program, shake a roll of micro:bit. There are number 1-6 randomly appearing on the dot matrix, which is exactly the same as playing the dice, as shown in Figure 8-4, Figure 8-5.

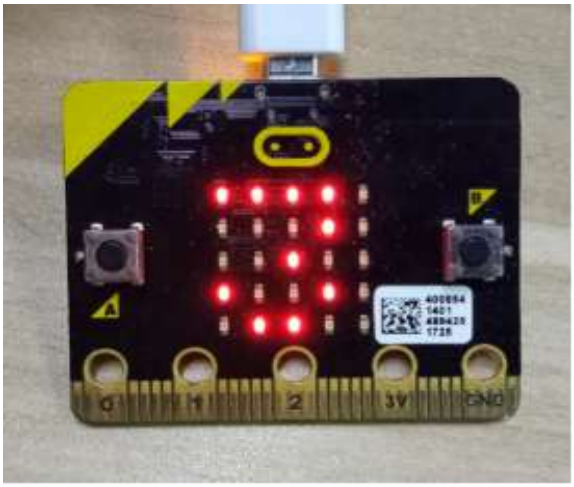


Figure 8-4

Figure 8-5

9-Direction follower

Learning goals:

This lesson learns the use of compasses to achieve the orientation of the micro:bit, and the arrows above the micro:bit dot matrix point to the north.

Code:

```
from microbit import *

compass.calibrate()

while True:

    needle = ((15 - compass.heading()) // 30) % 12

display.show(Image.ALL_CLOCKS[needle])
```

In the program, the compass.calibrate() function is first called to perform compass calibration. The calibration process is as shown in Figure 9-1. The small red dot in the center is drawn on the micro:bit dot matrix. After the description, a smiley, will appear on the dot matrix, which indicating calibration is completed.

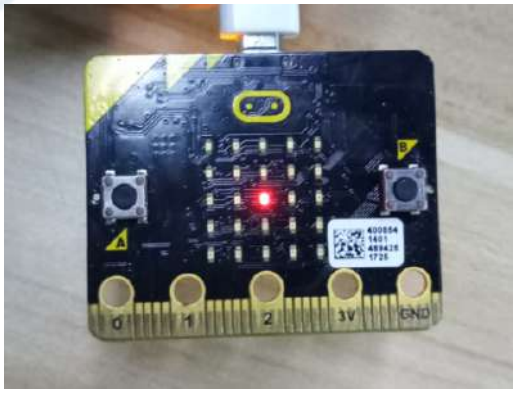


Figure 9-1

Programming and downloading:

1.You should open the Mu software, and enter the code in the edit window, , as shown in Figure 9-2.

Note! All English and symbols should be entered in English, and the last line must be a space.

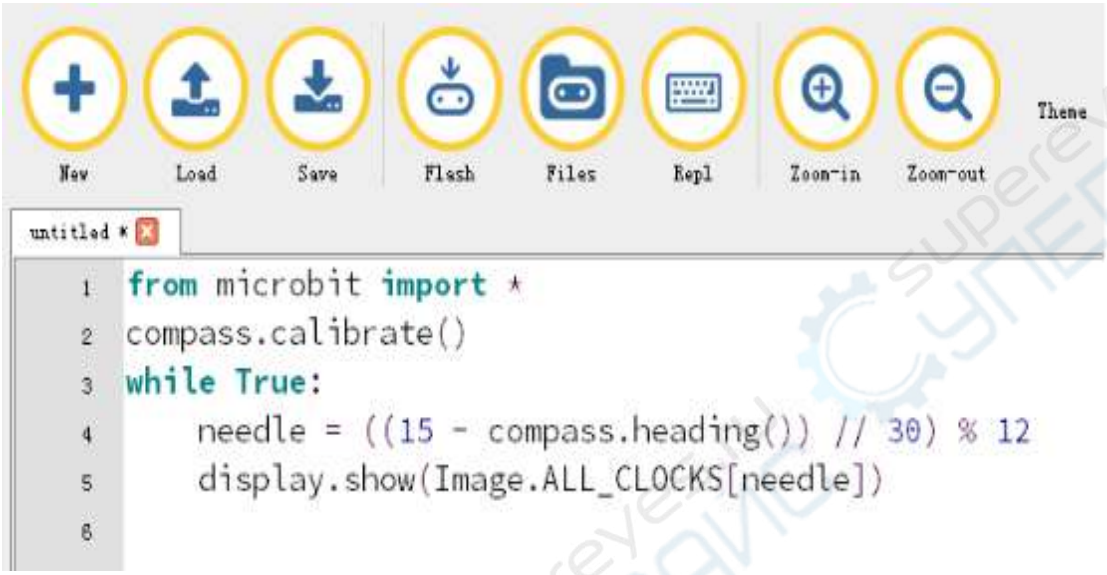


Figure 9-2

2.As shown in Figure 9-3, you need to click the Check button to check if our code has an error. If a line appears with a cursor or an underscore, the program indicating this line is wrong.

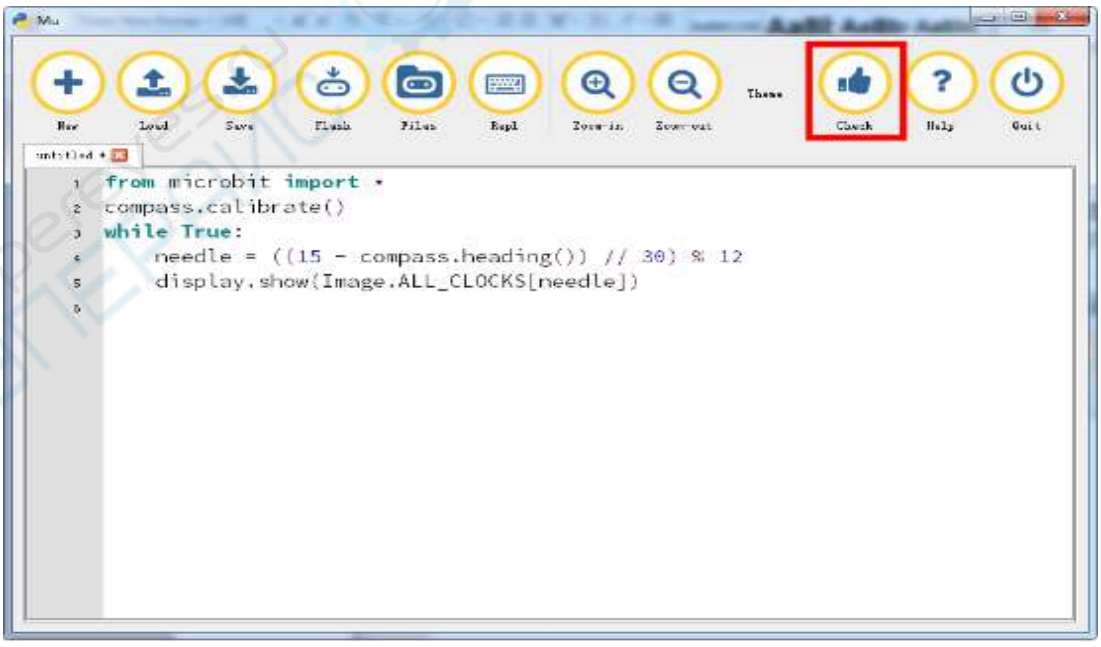


Figure 9-3

3.You need to connect the micro data cable to micro:bit and the computer, then click the Flash button to download the program to micro:bit as shown in Figure 9-4.

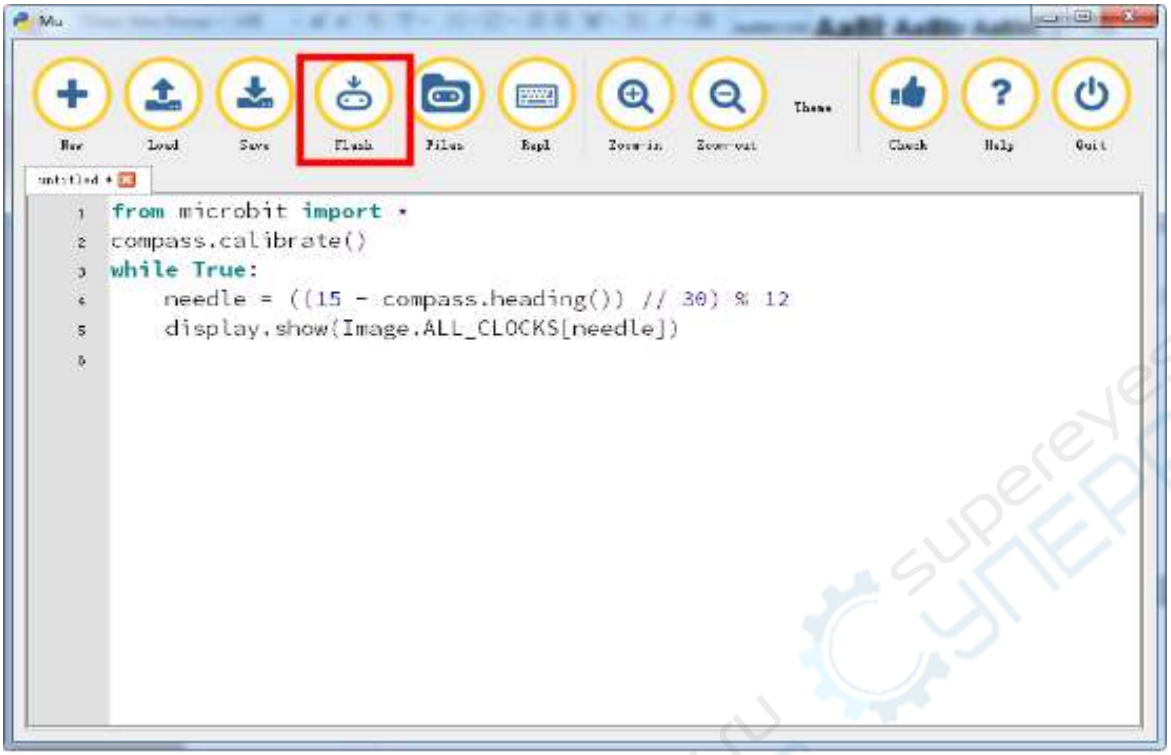


Figure 9-4

4. Experimental phenomena as shown in Figure 9-5 to Figure 9-10, no matter how you turn micro:bit, the pointers on the dot matrix point to the north.

Figure 9-5

Figure 9-6

Figure 9-7

Figure 9-8

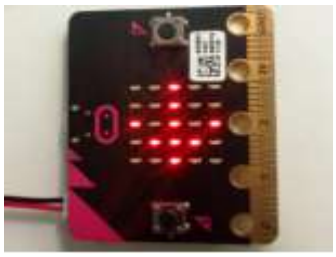


Figure 9-9

Figure 9-10

10-Microbit voice talk

Learning goals:

This lesson learns to use Python programming to let the micro:bit emit a voice.

Code:

```
from microbit import *

import speech

display.show(Image.HAPPY)

speech.say(”Hi, I’m an excellent robot from Yahboom.”)
```

In the program,import speech is means to import the speech library function, use the speech.say function in the library to play “Hi, I’m an excellent robot from Yahboom”.

“Hi, I’m an excellent robot from Yahboom” Change to whatever you want to play.

Programming and downloading:

1.You should open the Mu software, and enter the code in the edit window, , as shown in Figure 10-1.

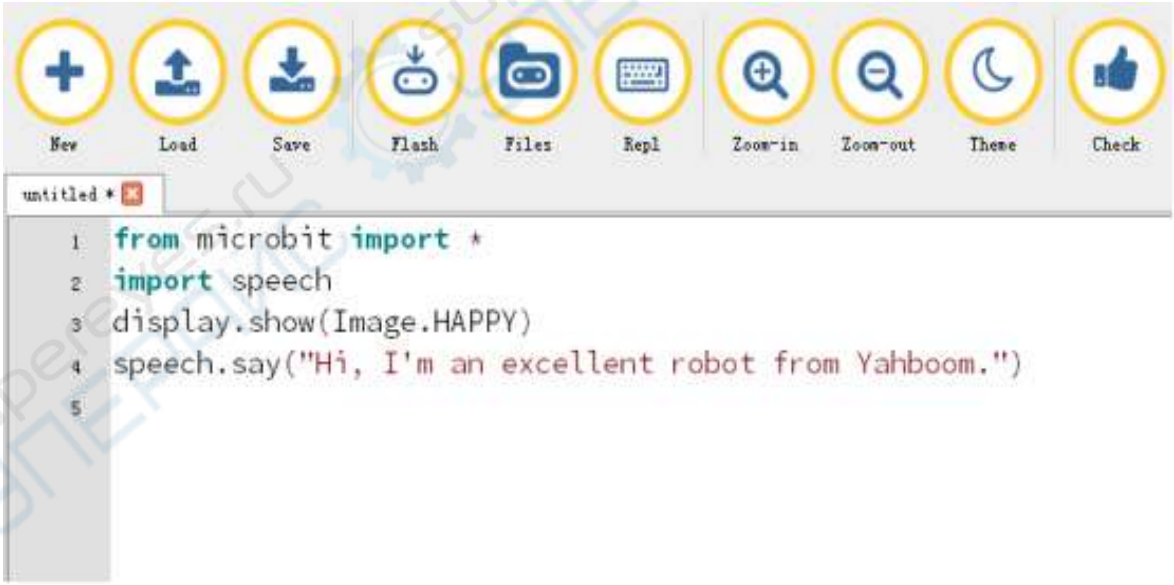


Figure 10-1

2.As shown in Figure 10-2, you need to click the Check button to check if our code has an error. If a line appears with a cursor or an underscore, the program indicating this line is wrong.

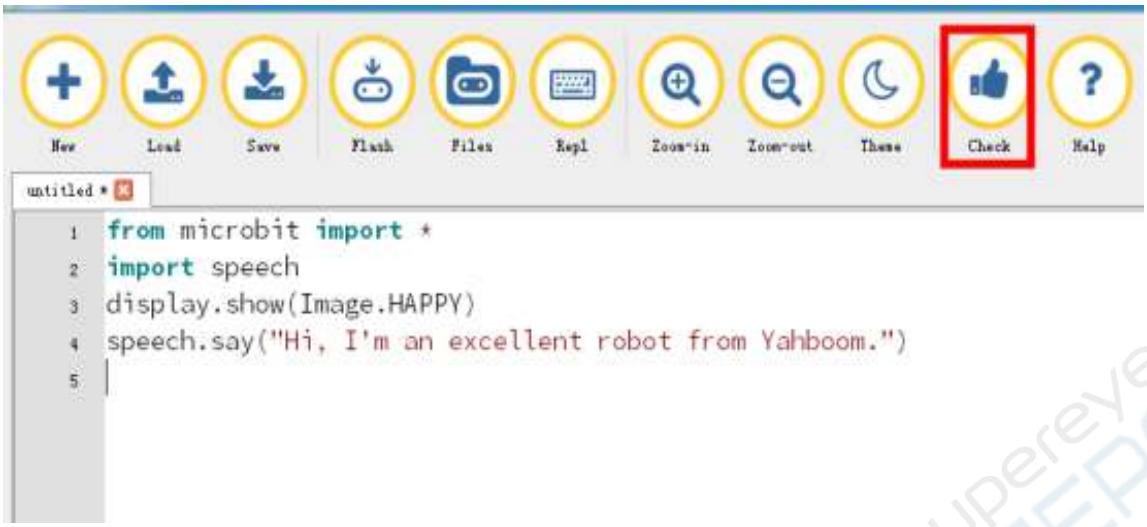


Figure 10-2

3.You need to connect the micro data cable to micro:bit and the computer, then click the Flash button to download the program to micro:bit as shown in Figure 10-3.

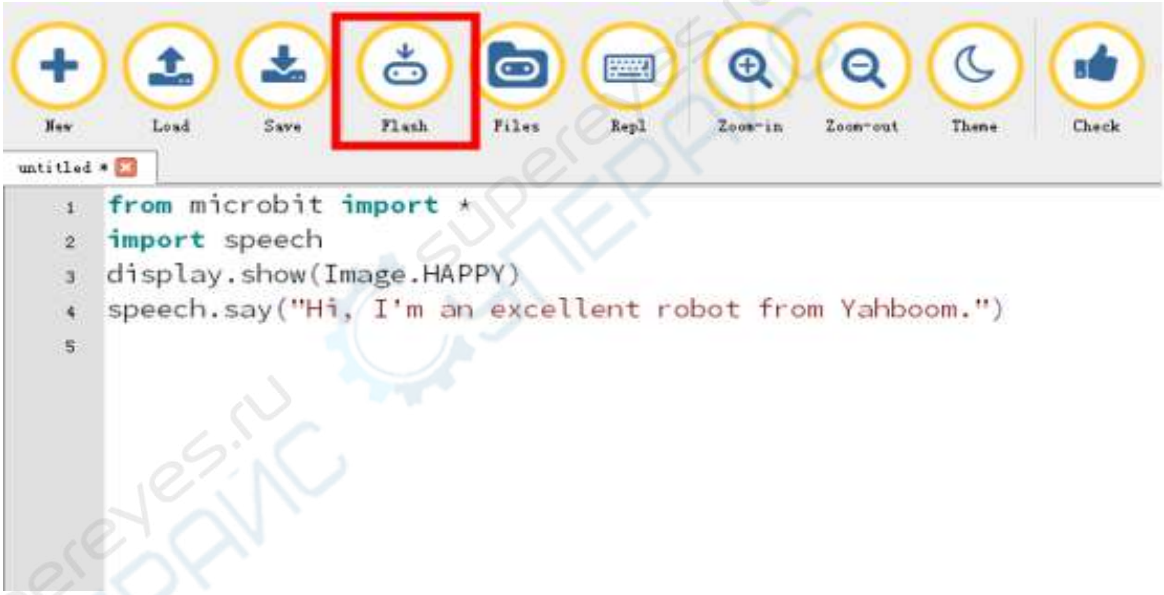


Figure 10-3

4.After downloading the program into micro:bit, you can see a smiley face on the robot's dot matrix and make a sound: Hi, I'm an excellent robot from Yahboom.

We need to use headphones or speakers to hear the sound. The two wiring methods are as shown below.

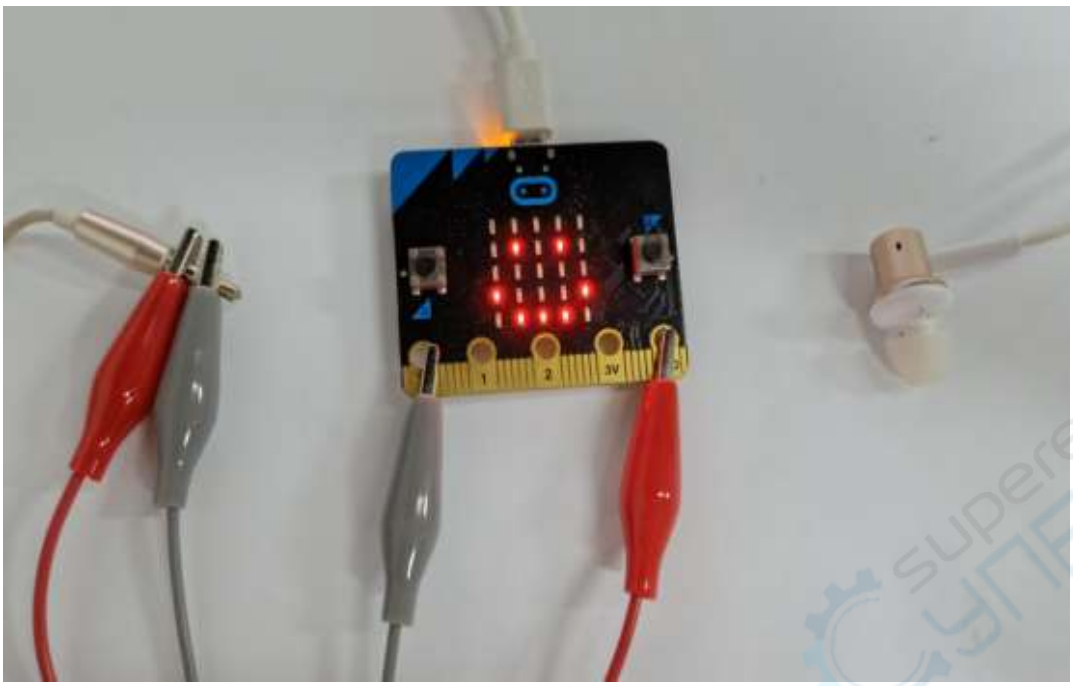


Figure 10-4

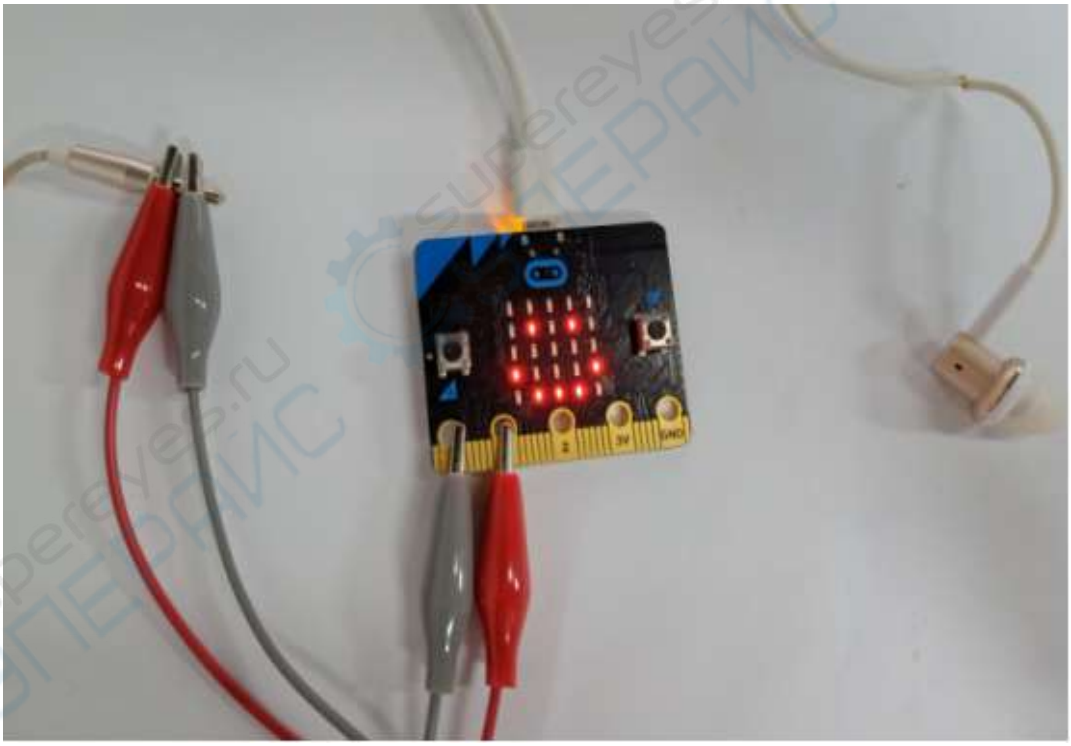
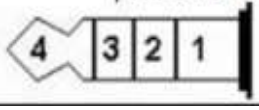


Figure 10-5

Headphone interface:

PIN	Description
1	Ground
2	Microphone
3	Right Side Earpiece
4	Left Side Earpiece



11-Colorful water lights

Learning goals:

This lesson learns to use Python programming to light up the water lights of micro:bit smart car.

Code:

```
from microbit import *

import neopixel

display.show(Image.HAPPY)

# The water lamp is connected to pin pin16, the number is 3

np = neopixel.NeoPixel(pin16, 3)

# iterate each LED in the water lights

for pixel_id in range(0, len(np)):

    # Light up the first water light to red

    np[0] = (255, 0, 0)

    # display color

    np.show()
```

In the program, import neopixel is means to import neopixel library, we can make micro:bit robot display a smile on the lattice. Then define the pin of the water light as pin16, the number is 3, iterate each LED in the water lights., np[0] = (255, 0, 0) means that the first water light is red. Modify the parameters in the brackets to change the color of the light.

Programming and downloading:

1.You should open the Mu software, and enter the code in the edit window, , as shown in Figure 11-1.

Figure11-1

2.As shown in Figure 11-2, you need to click the Check button to check if our code has an error. If a line appears with a cursor or an underscore, the program indicating this line is wrong.

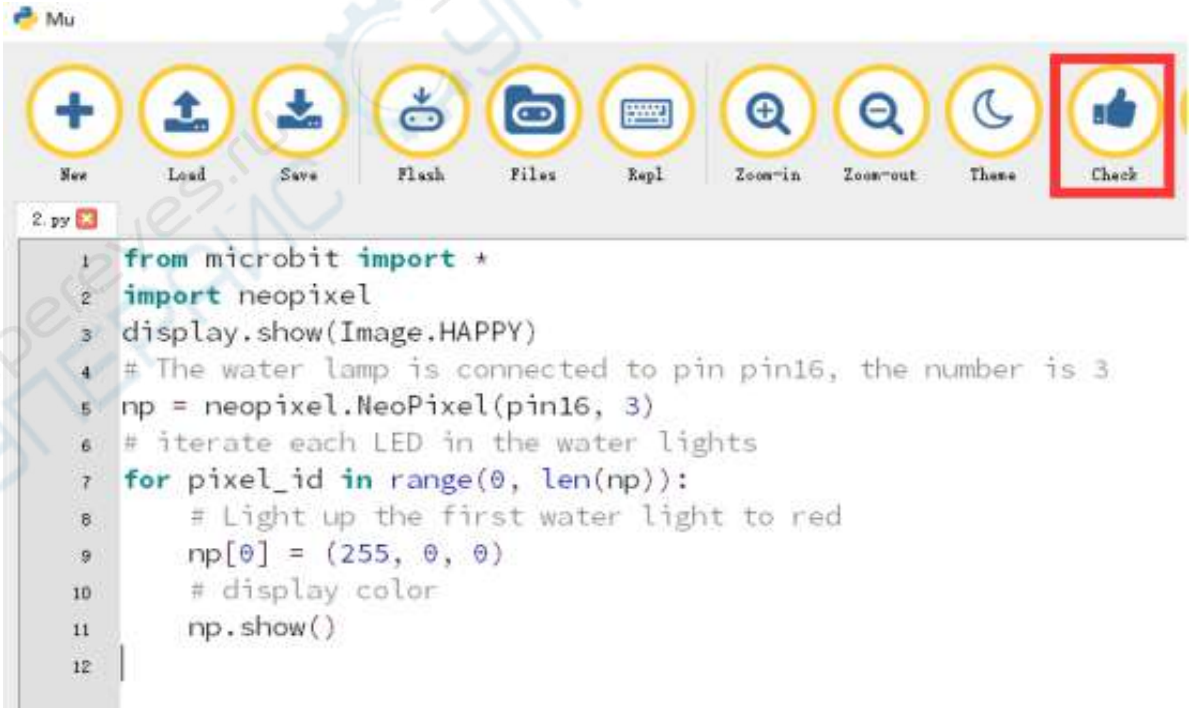


Figure 11-2

3.You need to connect the micro data cable to micro:bit and the computer, then click the Flash button to download the program to micro:bit as shown in Figure 11-3.

Figure 11-3

4. The schematic diagram of the flow lamp of the robot is shown in Figure 11-4. As you can see, the flow light of the robot is connected to the pin16 of the micro:bit. Therefore, we set the pin of the flow lamp to pin16 in the program. After downloading the program to micro:bit, you can see a smile on the dot matrix of the robot as shown in Figure 11-5, and light the first water light to red.

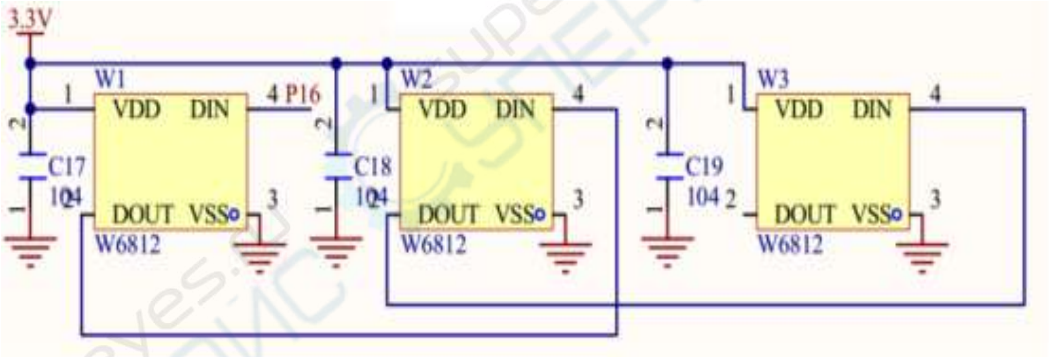


Figure 11-4

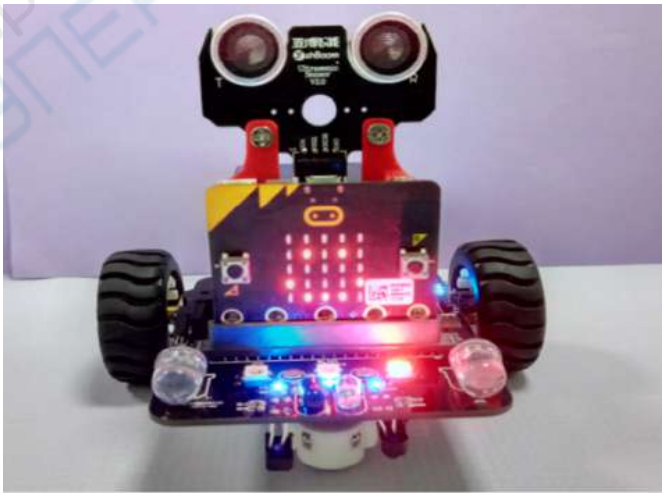


Figure 11-5

12-Colorful marquee

Learning goals:

This lesson learns to use Python programming to turn the micro:bit robot's water light from left to right.

Code:

```
from microbit import *

import neopixel

display.show(Image.HAPPY)

# The water lamp is connected to pin pin16, the number is 3

np = neopixel.NeoPixel(pin16, 3)

while True:

    for pixel_id in range(0, len(np)):

        np[0] = (255, 0, 0)

        np.show()

        sleep(200)

        np.clear()

        np[1] = (0, 255, 255)

        np.show()

        sleep(200)

        np.clear()

        np[2] = (0, 0, 255)

        np.show()

        sleep(200)

         np.clear()

        np[0] = (255, 255, 0)
```

```
np.show()
        sleep(200)
        np.clear()
        np[1] = (0, 255, 0)
        np.show()
        sleep(200)
        np.clear()
        np[2] = (255, 0, 255)
        np.show()
        sleep(200)
        np.clear()
```

Import neopixel is means to import the neopixel library function, first let the robot display a smile, then define the pin of the flow lamp as pin16, the number is 3, iterate each LED in the water lights. np[0] = (255, 0, 0) means that the first water light is red, and the delay is 200 milliseconds after lighting, clearing the display, lighting the second light, and so on.

Programming and downloading:

1.You should open the Mu software, and enter the code in the edit window, , as shown in Figure 12-1.

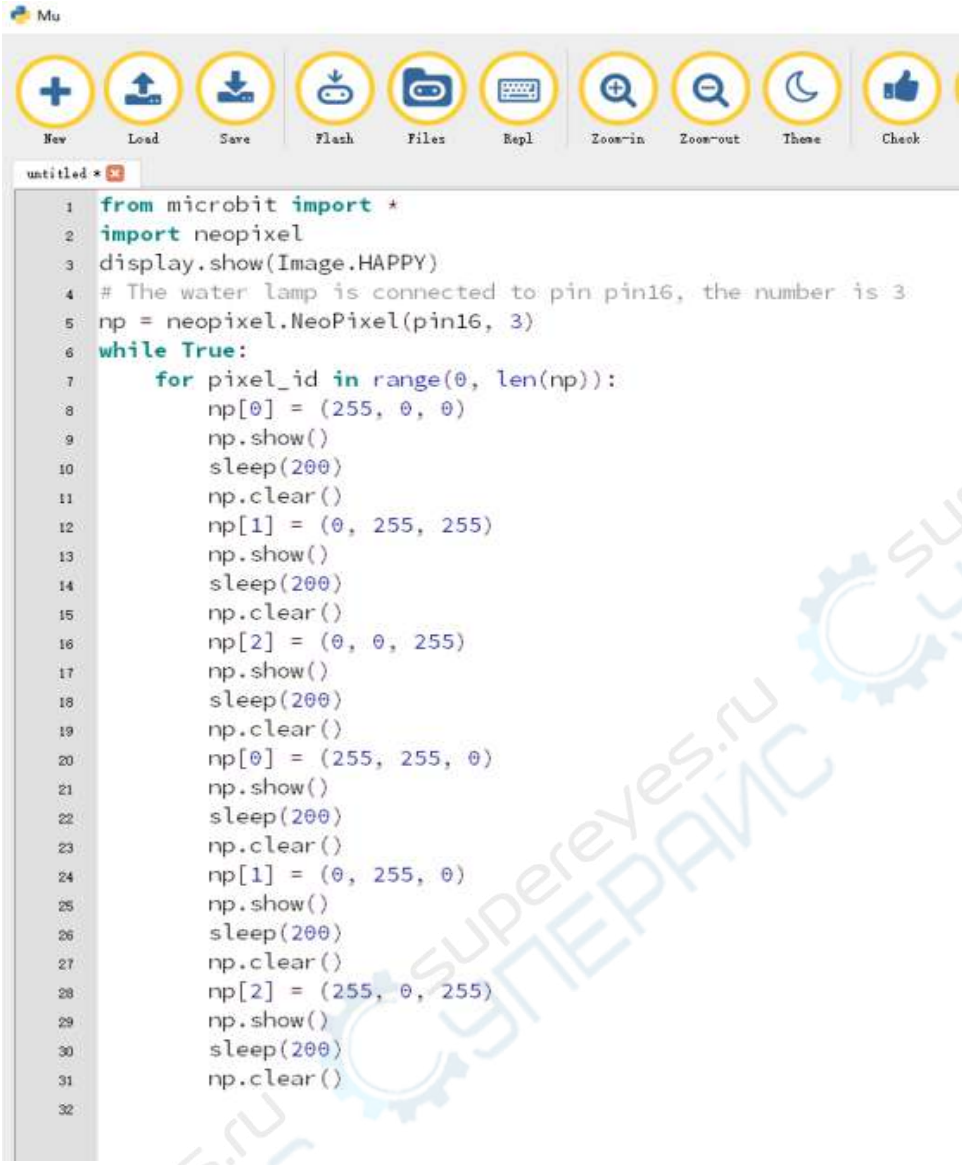


Figure 12-1

2.As shown in Figure 12-2, you need to click the Check button to check if our code has an error. If a line appears with a cursor or an underscore, the program indicating this line is wrong.

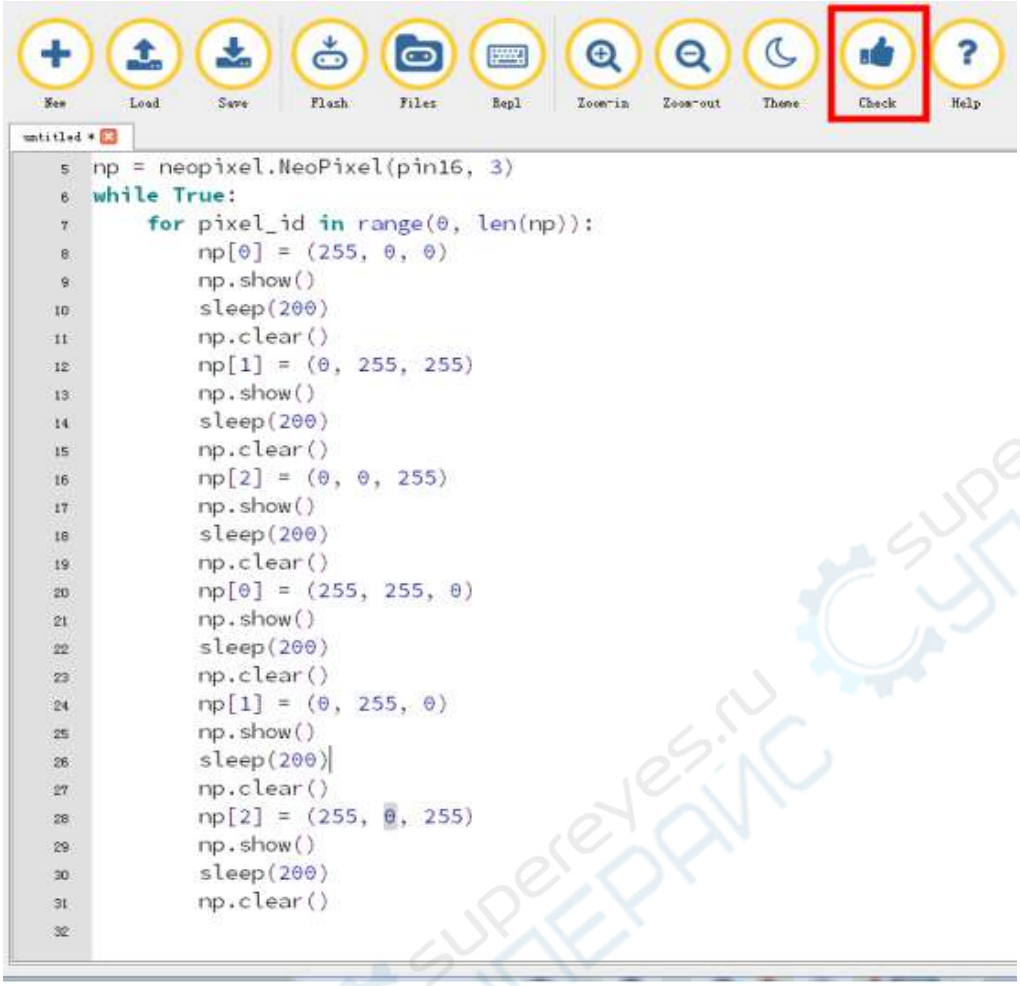


Figure 12-2

3.You need to connect the micro data cable to micro:bit and the computer, then click the Flash button to download the program to micro:bit as shown in Figure 12-3.

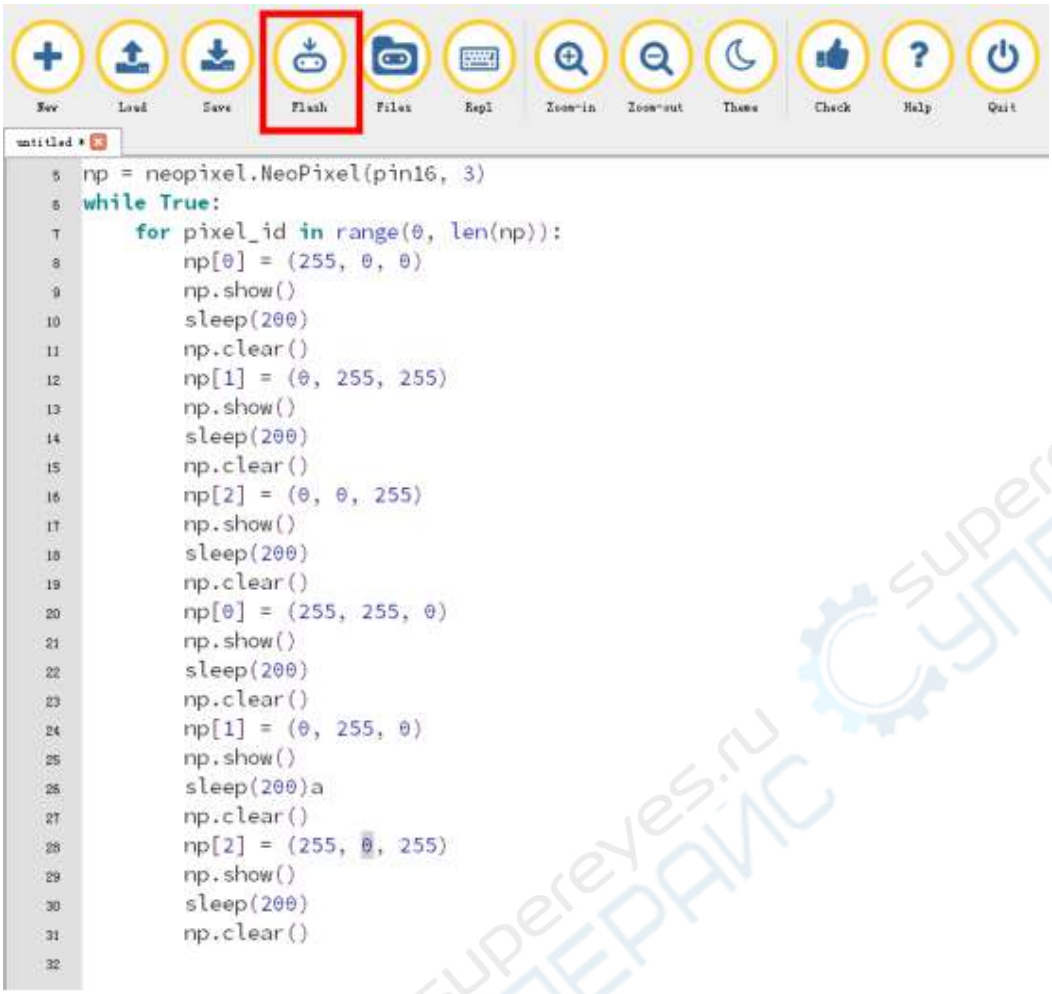
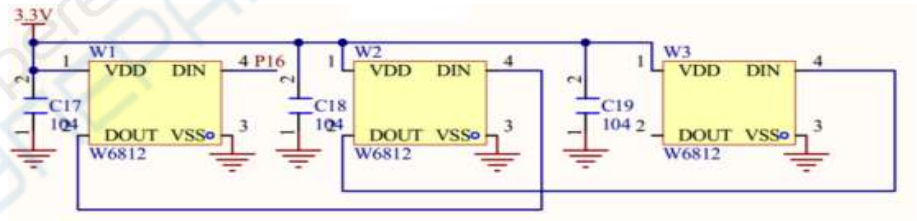


Figure 12-3

4. The schematic diagram of the robot's water lamp is shown in Figure 12-4. As you can see, the robot's flow lamp is connected to the micro:bit pin16. Therefore, we set the pin of the flow lamp to pin16 in the program. After downloading the program to micro:bit, you can see a smiley face on the robot's dot matrix and start running the marquee, as shown in Figures 12-5 to 12-7.



Figures 12-4

Figures 12-5

Figures 12-6

Figures 12-7

13-Colorful breathing light

Learning goals:

This lesson learns to use Python programming to make the water light slowly turn off and to achieve the effect of breathing light.

Code:

```
from microbit import *

import neopixel

display.show(Image.HAPPY)

np = neopixel.NeoPixel(pin16, 3)

while True:

for num in range(0, 255):

    for pixel_id in range(0, len(np)):

        np[pixel_id] = (num, 0, num)

        np.show()

        sleep(10)
```

Import neopixel is means to import the neopixel library function, first let the robot display a smiley face, then define the pin of the flow light as pin16, the number is 3, iterate between 0 and 255, and display in the water lights. np[pixel_id] = (num, 0, num) means that the purple color is displayed, and their brightness values are superimposed from 0 every 10 milliseconds to stop at 255.

Programming and downloading:

1.You should open the Mu software, and enter the code in the edit window, , as shown in Figure 13-1.

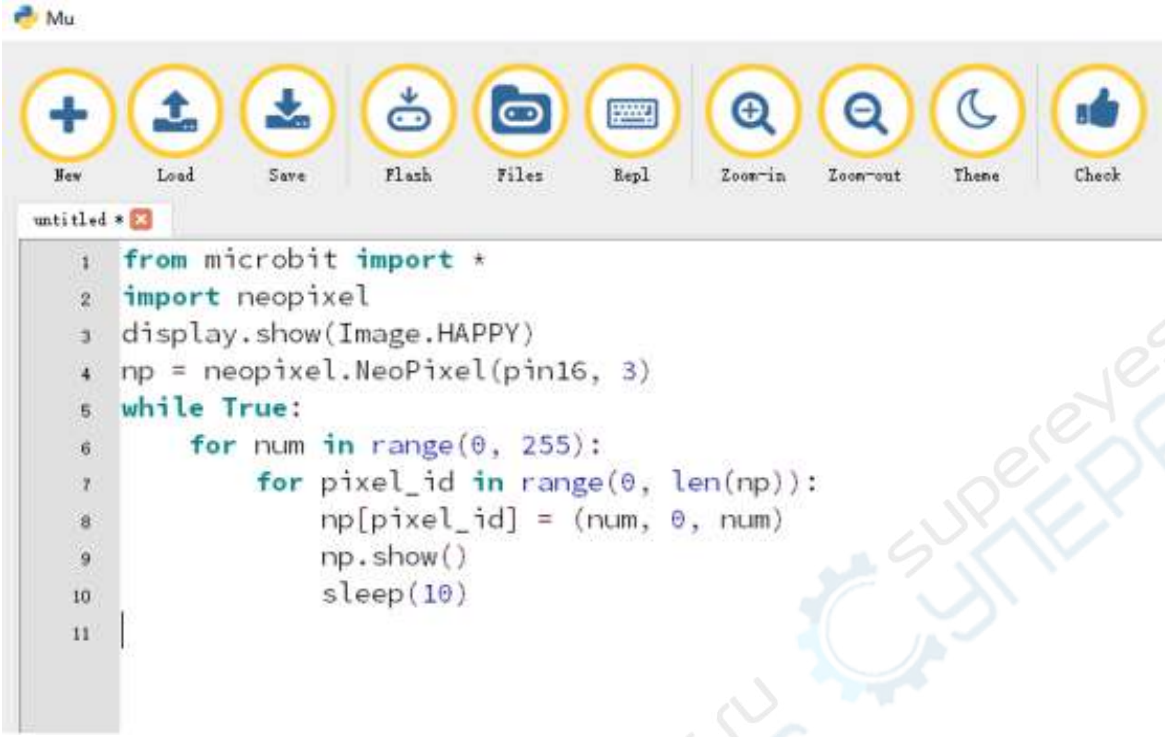


Figure 13-1

2.As shown in Figure 13-2, you need to click the Check button to check if our code has an error. If a line appears with a cursor or an underscore, the program indicating this line is wrong.

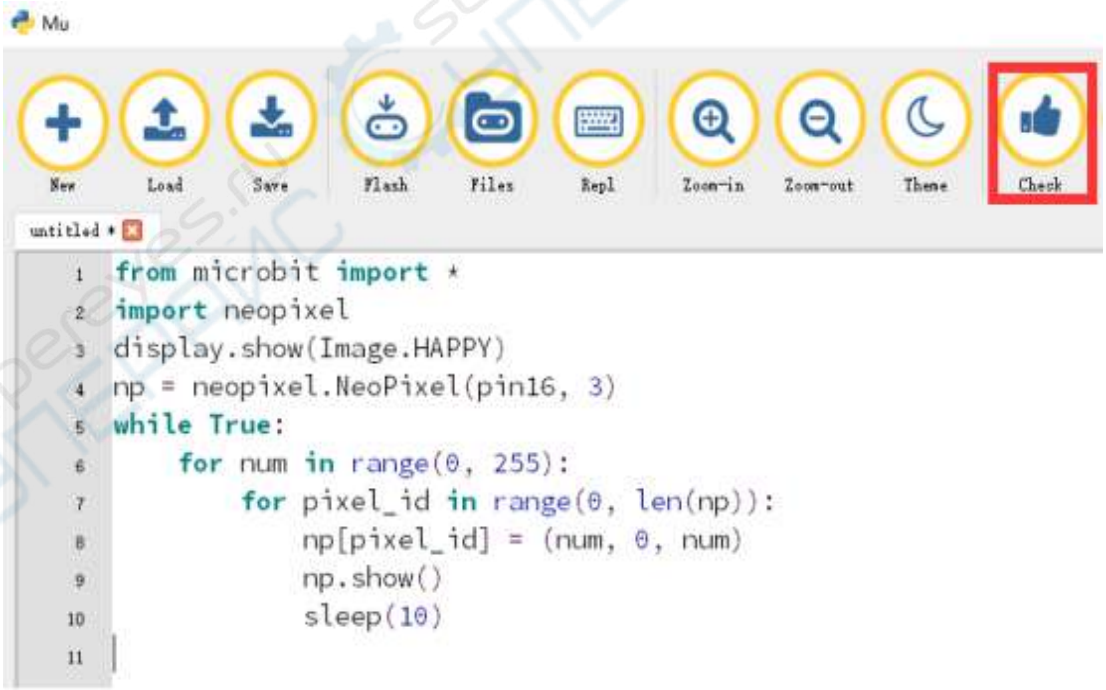


Figure 13-2

3.You need to connect the micro data cable to micro:bit and the computer, then click the Flash button to download the program to micro:bit as shown in Figure 13-3.

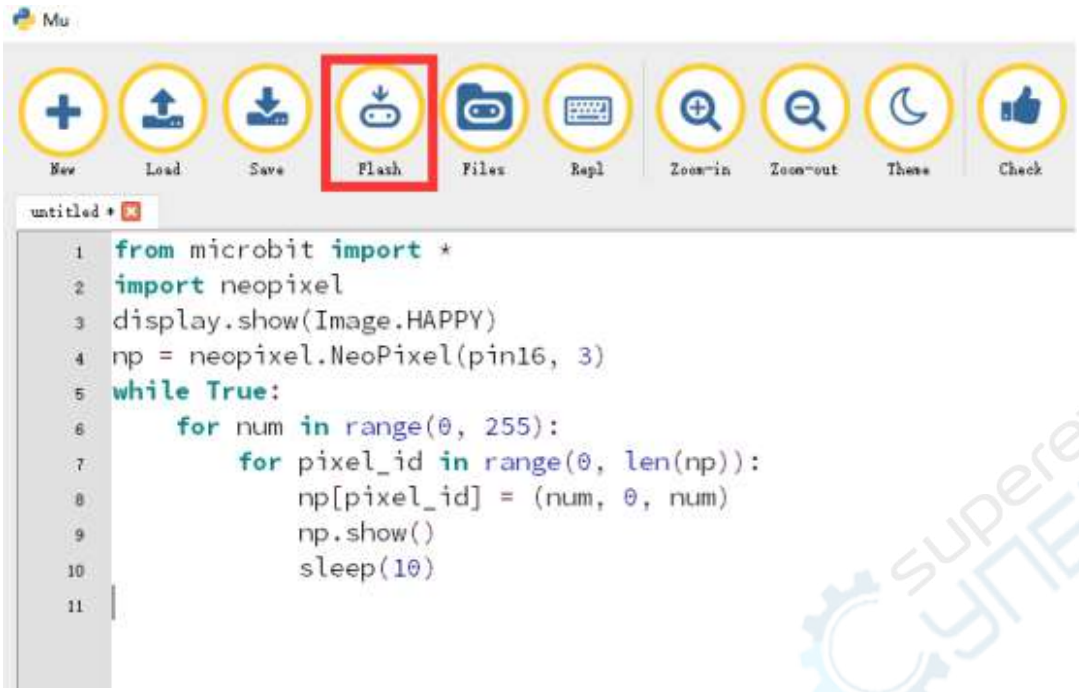


Figure 13-3

4.The schematic diagram of the robot's water lamp is shown in Figure 13-4. As you can see, the robot's flow lamp is connected to the micro:bit pin16. Therefore, we set the pin of the flow lamp to pin16 in the program. After downloading the program to micro:bit, you can see a smiley face on the robot's dot matrix and the running light slowly lights up, as shown in Figures 13-5 to 13-8.

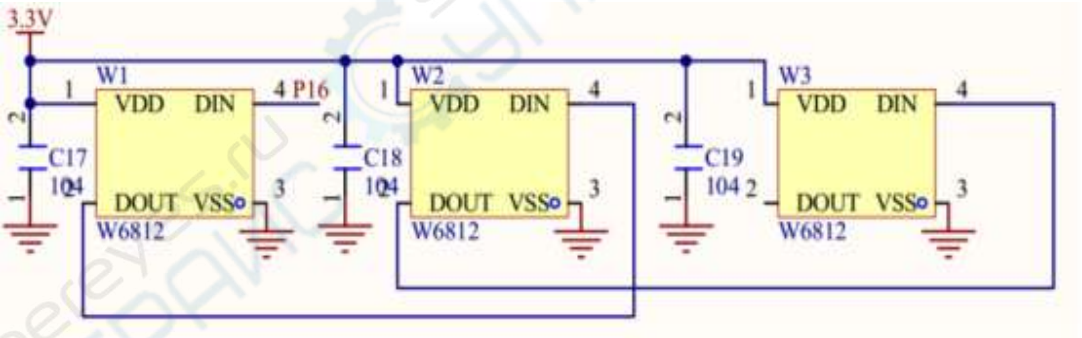
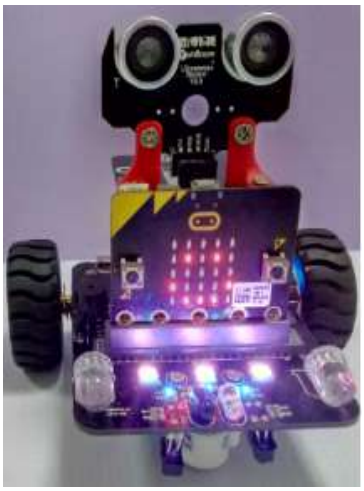


Figure 13-4

Figures 13-5

Figures 13-6



Figures 13-7

Figures 13-8

14-Robot advance

Learning goals:

This lesson learns to use Python programming to make the robot advance.

Code:

```
from microbit import *

import ustruct

import math

# Registers/etc:

PCA9685_ADDRESS = 0x41

MODE1 = 0x00

MODE2 = 0x01

SUBADR1 = 0x02

SUBADR2 = 0x03

SUBADR3 = 0x04

PRESCALE = 0xFE

LED0_ON_L = 0x06

LED0_ON_H = 0x07
```

```
LED0_OFF_L = 0x08
LED0_OFF_H = 0x09
ALL _
LED_ON_L = 0xFA
ALL_LED_ON_H = 0xFB
ALL _
LED_OFF_L = 0xFC

ALL_LED_OFF_H = 0xFD

# Bits:

RESTART = 0x80

SLEEP = 0x10

ALLCALL = 0x01

INVRT = 0x10

OUTDRV = 0x04

RESET = 0x00

class PCA9685():
    """PCA9685 PWM LED/servo controller."""
    def __init__(self, address=PCA9685_ADDRESS):
        """Initialize the PCA9685."""
        self.address = address
        i2c.write(self.address, bytearray([MODE1, RESET]))
        self.set_all_pwm(0, 0)
        i2c.write(self.address, bytearray([MODE2, OUTDRV]))
        i2c.write(self.address, bytearray([MODE1, ALLCALL]))
        sleep(5)  # wait for oscillator
        i2c.write(self.address, bytearray([MODE1]))
        mode1 = i2c.read(self.address, 1)
```

```
        mode1 = ustruct.unpack('<H', mode1)[0]
        mode1 = mode1 & ~SLEEP  # wake up (reset sleep)
        i2c.write(self.address, bytearray([MODE1, mode1]))
        sleep(5)  # wait for oscillator
    def set_pwm_freq(self, freq_hz):
        """Set the PWM frequency to the provided value in hertz."""
        prescaleval = 25000000.0    # 25MHz
        prescaleval /= 4096.0       # 12-bit
        prescaleval /= float(freq_hz)
        prescaleval -= 1.0
        # print('Setting PWM frequency to {0} Hz'.format(freq_hz))
        # print('Estimated pre-scale: {0}'.format(prescaleval))
        prescale = int(math.floor(prescaleval + 0.5))
        # print('Final pre-scale: {0}'.format(prescale))
        i2c.write(self.address, bytearray([MODE1]))
        oldmode = i2c.read(self.address, 1)
        oldmode = ustruct.unpack('<H', oldmode)[0]
        newmode = (oldmode & 0x7F) | 0x10    # sleep
        i2c.write(self.address, bytearray([MODE1, newmode]))  # go to sleep
        i2c.write(self.address, bytearray([PRESCALE, prescale]))
        i2c.write(self.address, bytearray([MODE1, oldmode]))
        sleep(5)
        i2c.write(self.address, bytearray([MODE1, oldmode | 0x80]))
```

```
    def set_pwm(self, channel, on, off):

        """Sets a single PWM channel."""

        if on is None or off is None:

            i2c.write(self.address, bytearray([LED0_ON_L+4*channel]))

            data = i2c.read(self.address, 4)
            return ustruct.unpack('<HH', data)
        i2c.write(self.address, bytearray([LED0 _
ON_L+4*channel, on & 0xFF]))
        i2c.write(self.address, bytearray([LED0_ON_H+4*channel, on >> 8]))
        i2c.write(self.address, bytearray([LED0 _ OFF_L+4*channel, off & 0xFF]))
        i2c.write(self.address, bytearray([LED0 _ OFF_H+4*channel, off >> 8]))
    def set _
all_pwm(self, on, off):

        """Sets all PWM channels."""

        i2c.write(self.address, bytearray([ALL_LED_ON_L, on & 0xFF]))
        i2c.write(self.address, bytearray([ALL_LED_ON_H, on >> 8]))
        i2c.write(self.address, bytearray([ALL _
LED_OFF_L, off & 0xFF]))

        i2c.write(self.address, bytearray([ALL_LED_OFF_H, off >> 8]))

    def duty(self, index, value=None, invert=False):
        if value is None:
            pwm = self.set _
pwm(index)

            if pwm == (0, 4096):

                value = 0
```

```
if invert:
            value = 4095 - value
        return value

    if not 0 <= value <= 4095:
        raise ValueError("Out of range")

    if invert:
        value = 4095 - value

    if value == 0:
        self.set_pwm(index, 0, 4096)

    elif value == 4095:
        self.set_pwm(index, 4096, 0)

    else:
        self.set_pwm(index, 0, value)

# Initialise the PCA9685 using the default address (0x41).

pwm = PCA9685()

# Configure min and max servo pulse lengths

servo_min = 150  # Min pulse length out of 4096  0?

servo_max = 600  # Max pulse length out of 4096: 180?

# Set frequency to 60hz, good for servos.

pwm.set_pwm_freq(60)

display.show(Image.HAPPY)

pwm.set_pwm(12, 0, 4095)

pwm.set_pwm(13, 0, 0)

pwm.set_pwm(15, 0, 4095)

pwm.set_pwm(14, 0, 0)

pwm.set_pwm(3, 0, servo_min)

sleep(1000)

pwm.set_pwm(3, 0, servo_max)

sleep(1000)
```

This section of the experiment uses I2C communication, through the PCA9685PW chip can output 16 PWM, so we can control the output of 4-way PWM control car forward.

Programming and downloading:

1.You should open the Mu software, and enter the code in the edit window, , as shown in Figure 18-1.

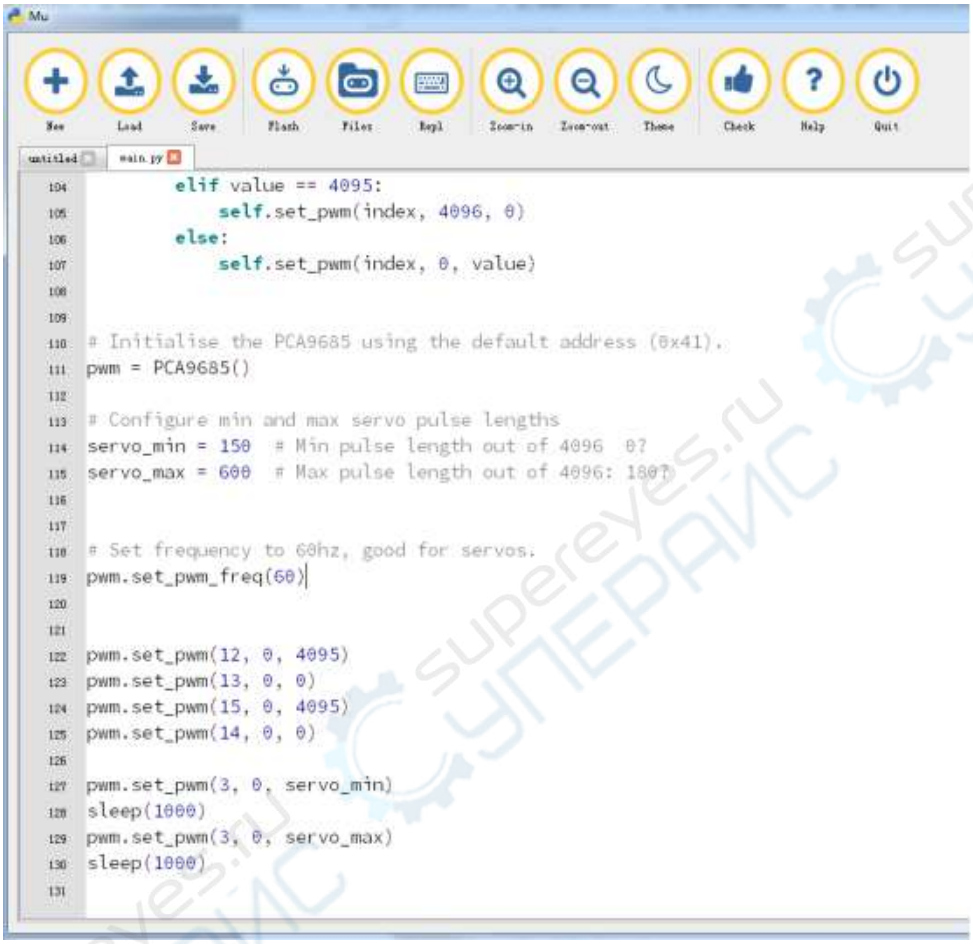


Figure 18-1

2.As shown in Figure 18-2, you need to click the Check button to check if our code has an error. If a line appears with a cursor or an underscore, the program indicating this line is wrong.

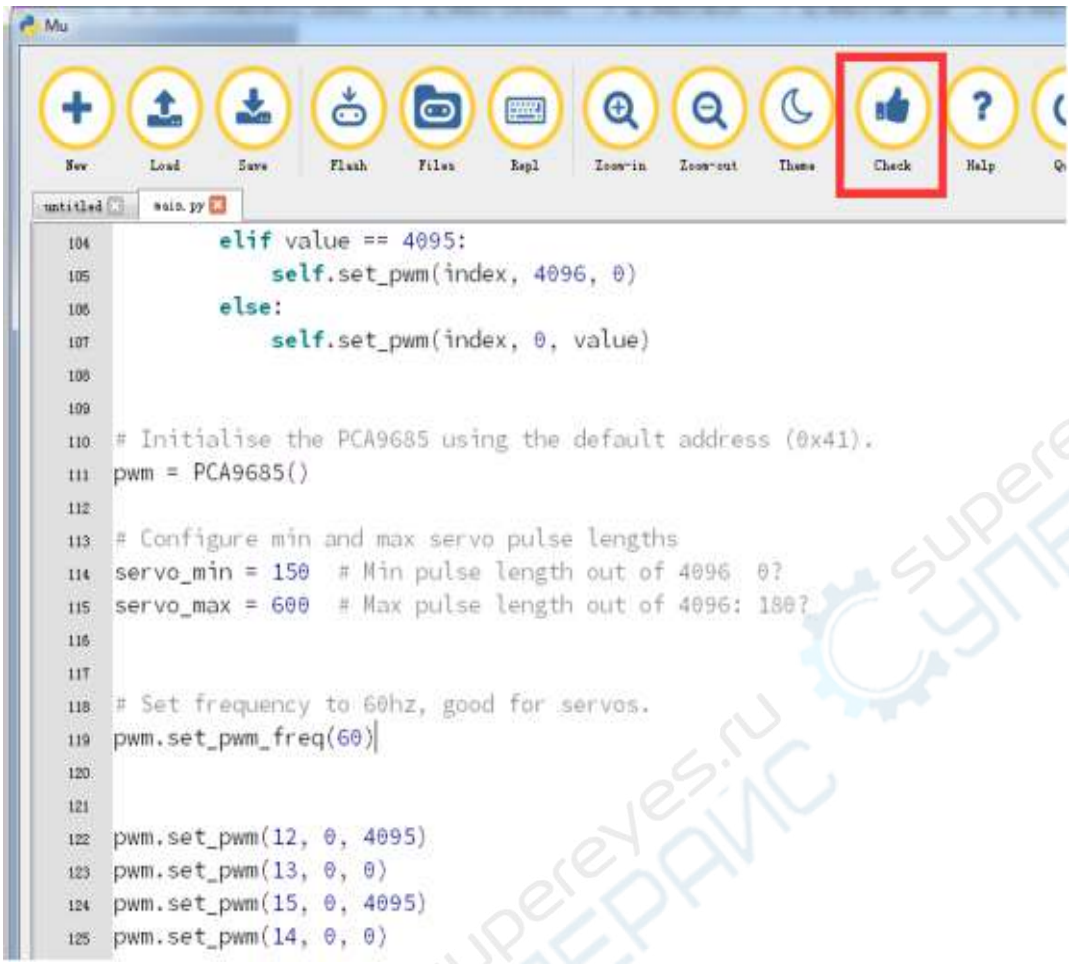


Figure 18-2

3.You need to connect the micro data cable to micro:bit and the computer, then click the Flash button to download the program to micro:bit as shown in Figure 18-3.

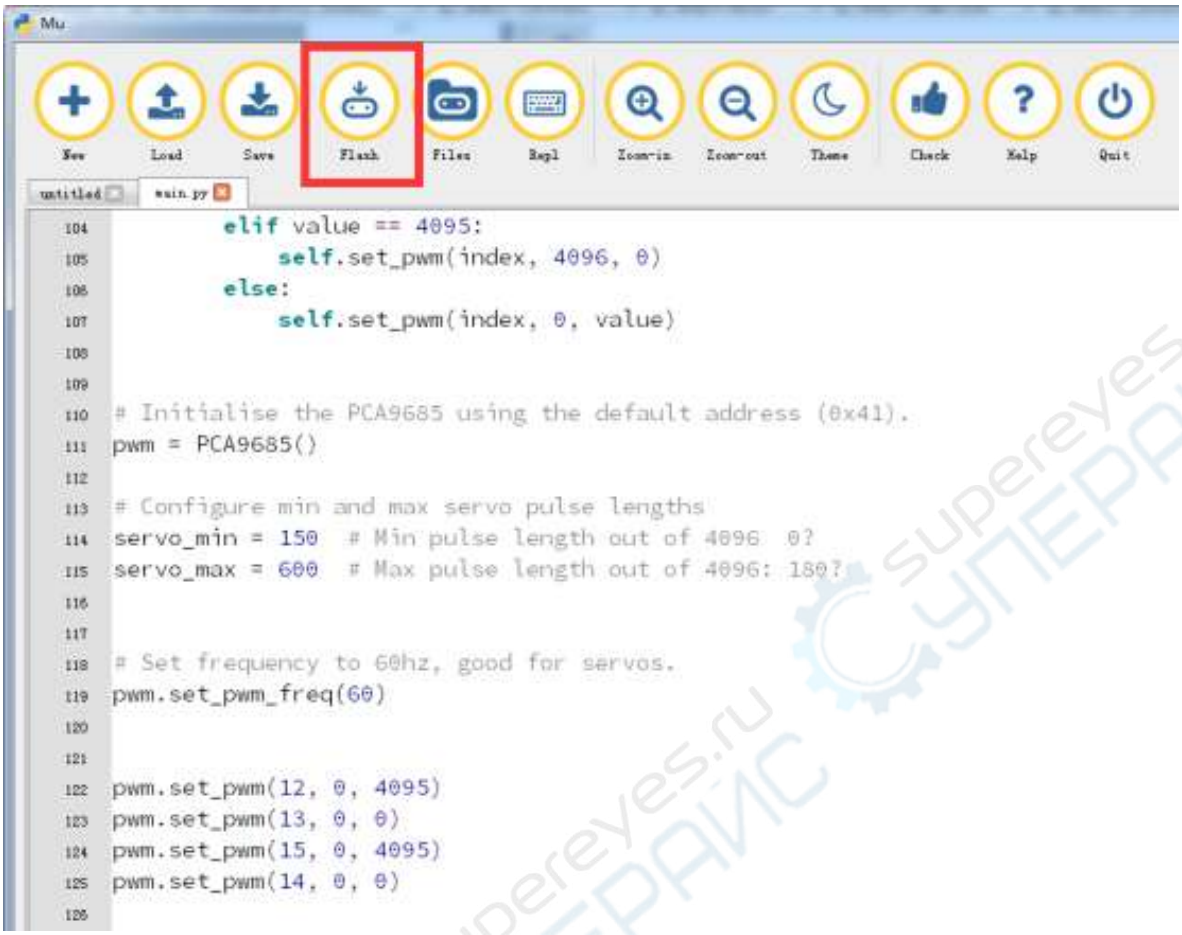


Figure 18-3

4. The schematic diagram of the robot's PCA9685PW chip and motor is shown in Figure 18-4 and Figure 18-5. As shown in the figure, the robot's motor is connected to the LINA, LINB, RINA, and RINB pins of the PCA9685PW chip, while the PCA9685PW The SCL and SDA are connected to the P19 and P20 pins of the micro:bit chip.

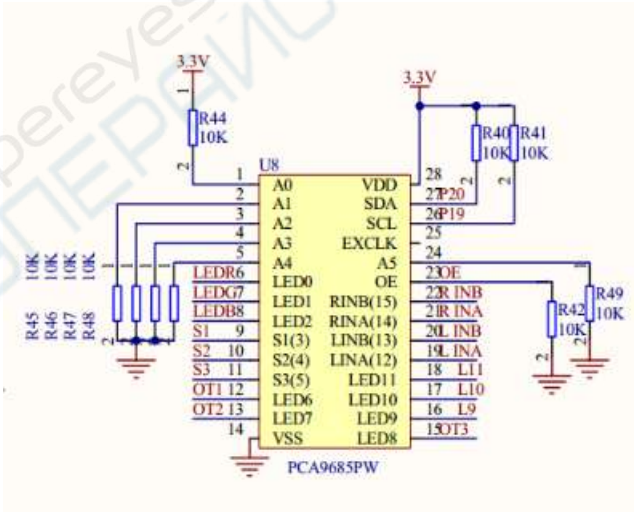


Figure 14-4

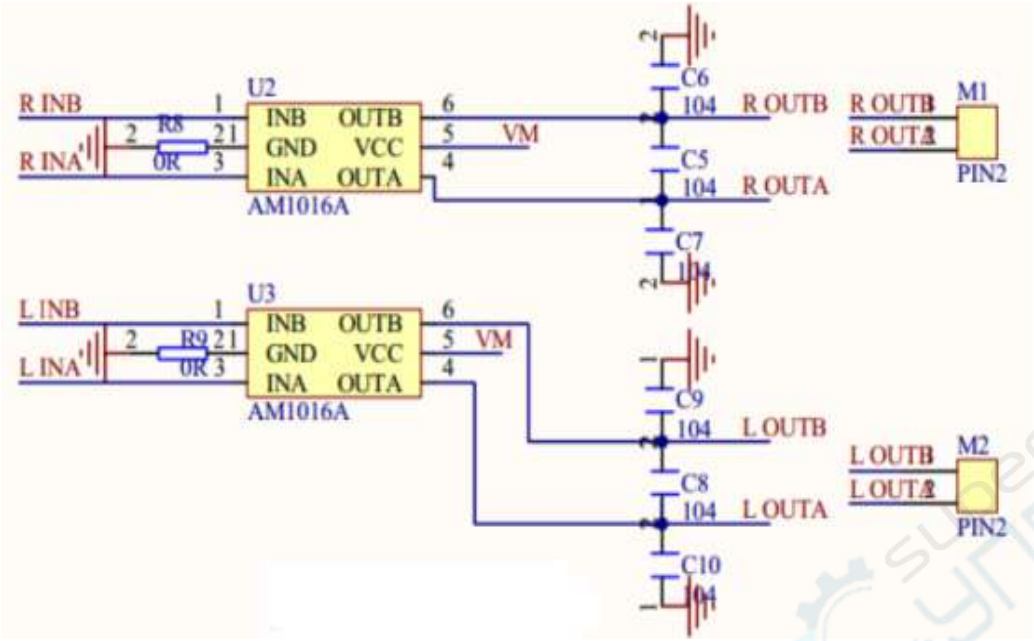
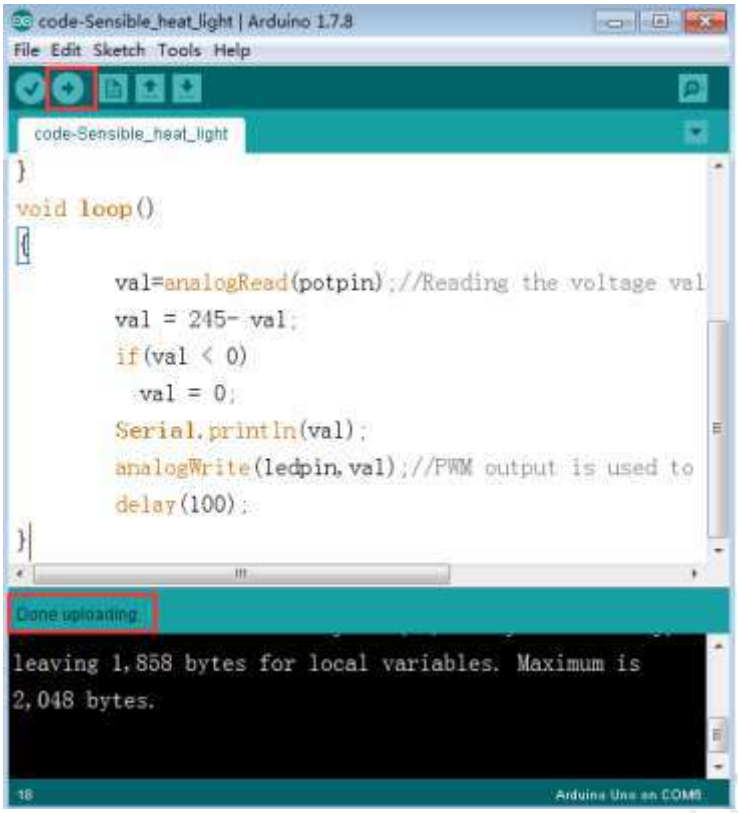


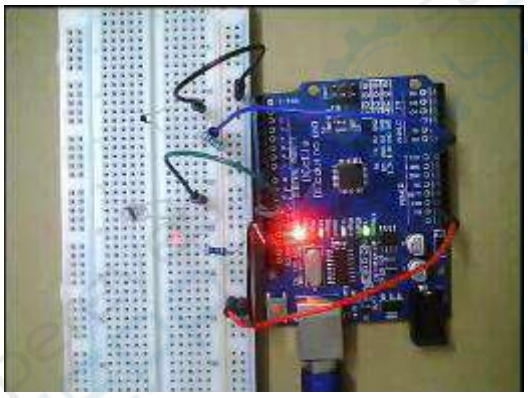
Figure 18-5

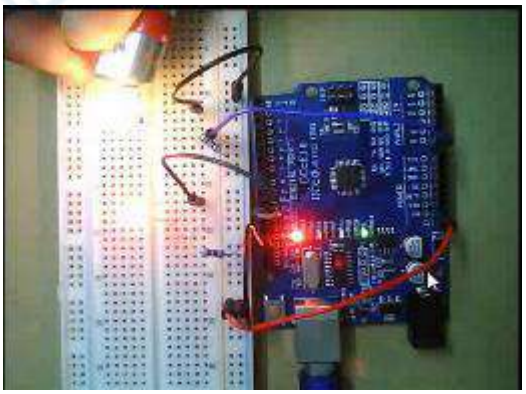
5. After downloading the program to micro:bit, you can see the robot will advance. As shown in Figures 18-6.

Figure 18-6



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

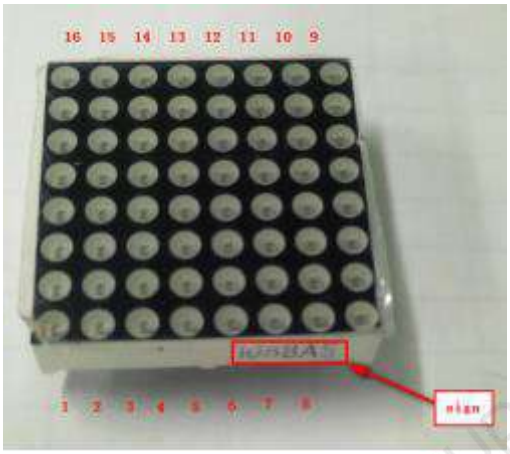
The purpose of the experiment:

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

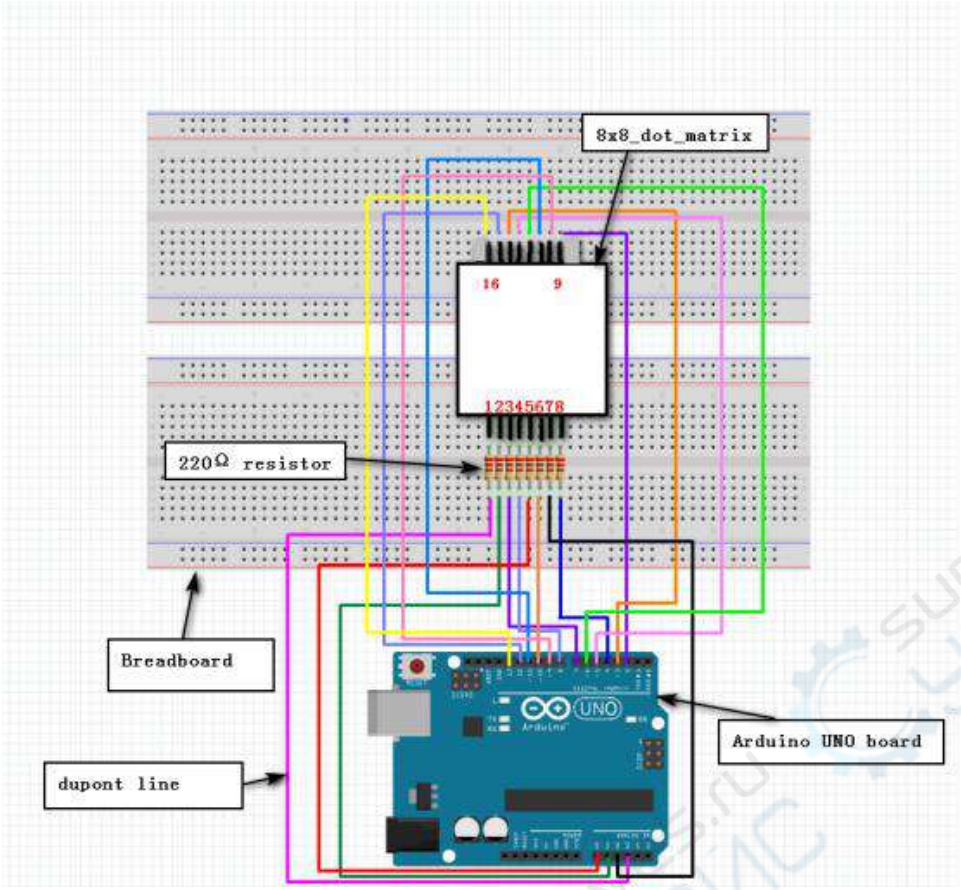
8x8 dot matrixLED*1

Breadboard *1

dupont line *1bunch

Actual object connection diagram :

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



Experimental code analysis:

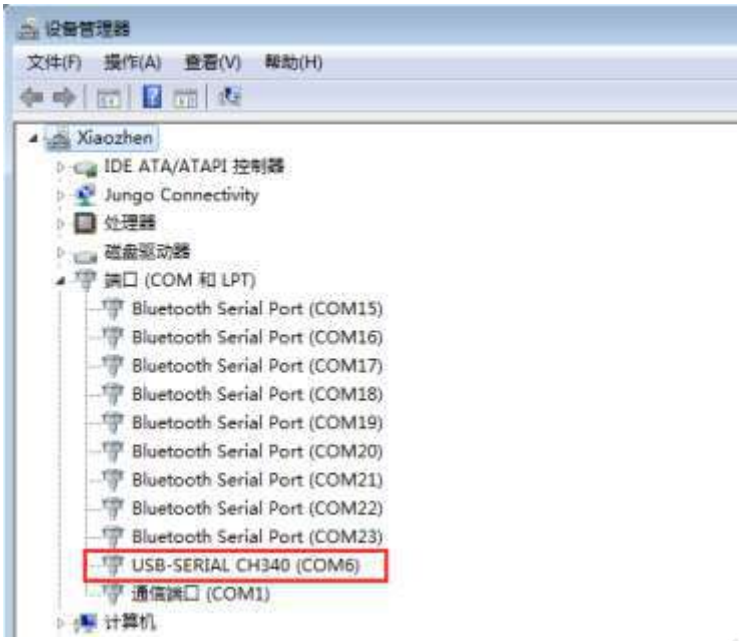
```
const int row1 = 2;   // Arduino Pin2 connect pin9 of the dot matrix
const int row2 = 3;   // Arduino Pin3 connect pin14 of the dot matrix
const int row3 = 4;   // Arduino Pin4 connect pin8 of the dot matrix
const int row4 = 5;   // Arduino Pin5 connect pin12 of the dot matrix
const int row5 = 17;  // 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 COl
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);
```

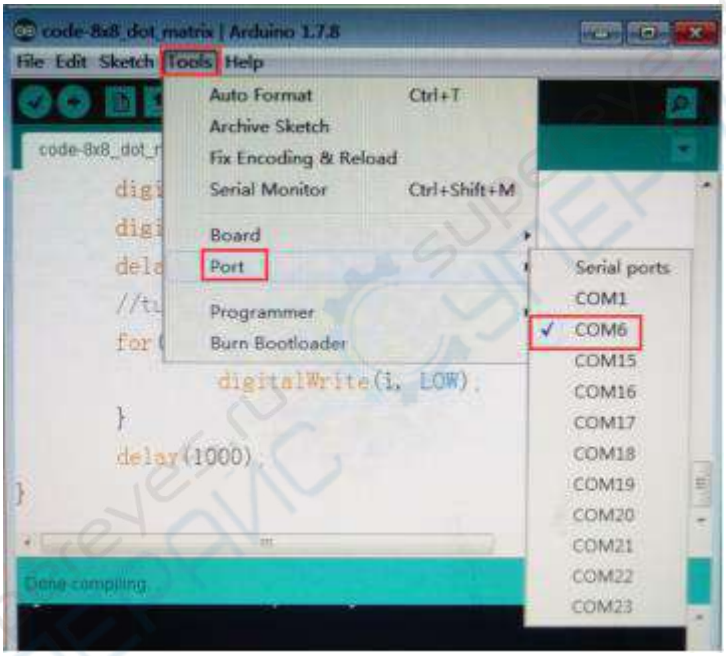
```
}
}
void loop()
{
int i;
//the row # 1 and col # 1 of the LEDs turn on
digitalWrite(row1, HIGH);
digitalWrite(row2, LOW);
digitalWrite(row3, LOW);
digitalWrite(row4, LOW);
digitalWrite(row5, LOW);
digitalWrite(row6, LOW);
digitalWrite(row7, LOW);
digitalWrite(row8, LOW);
digitalWrite(col1, LOW);
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:

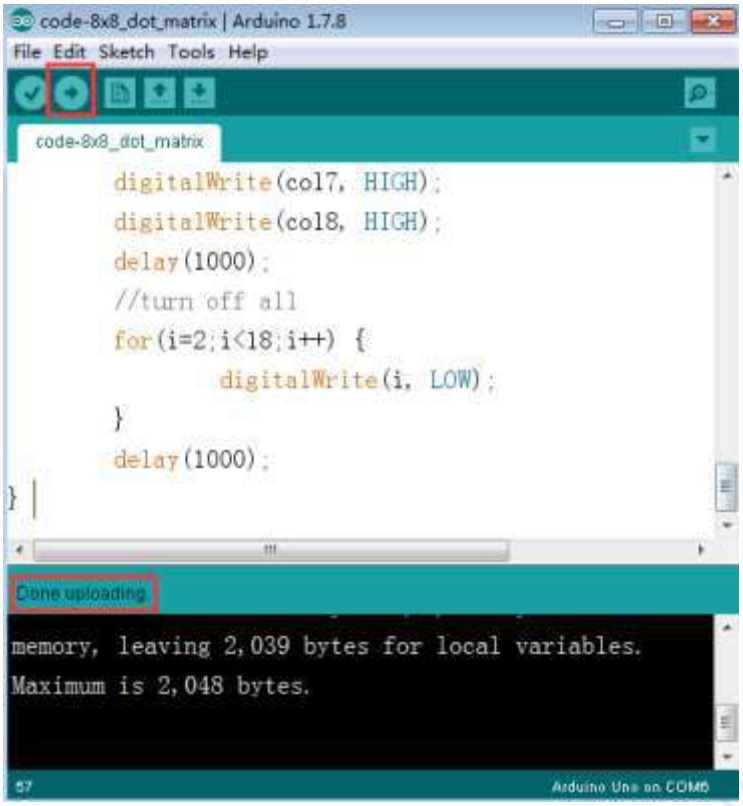
1.We need to open the code of this experiment: code-8x8_dot_matrix.ino, click“√” under the menu bar to compile the code, and wait for the word "Done compiling " in the lower right corner, as shown in the figure below.

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.

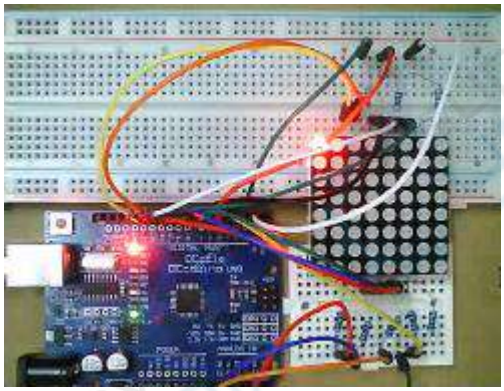




3. After the selection is completed, you need to click “→”under the menu bar to upload the code to the Arduino UNO board. When the word “Done uploading” appears in the lower left corner, thecode has been successfully uploaded to the Arduino UNO board, as shown in the figure below.



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

220Ω resistor *1

10kΩ resistor *1

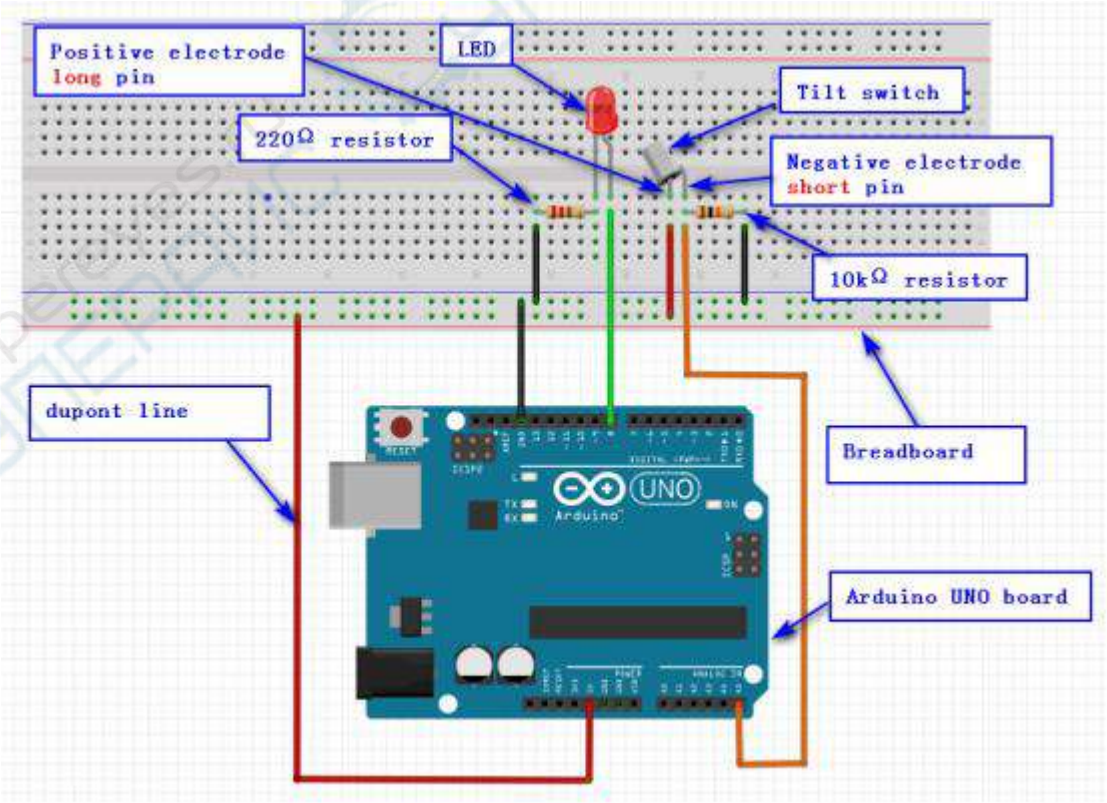
Tilt switch *1

Breadboard *1

Dupont line *1 bunch

Actual object connection diagram：

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

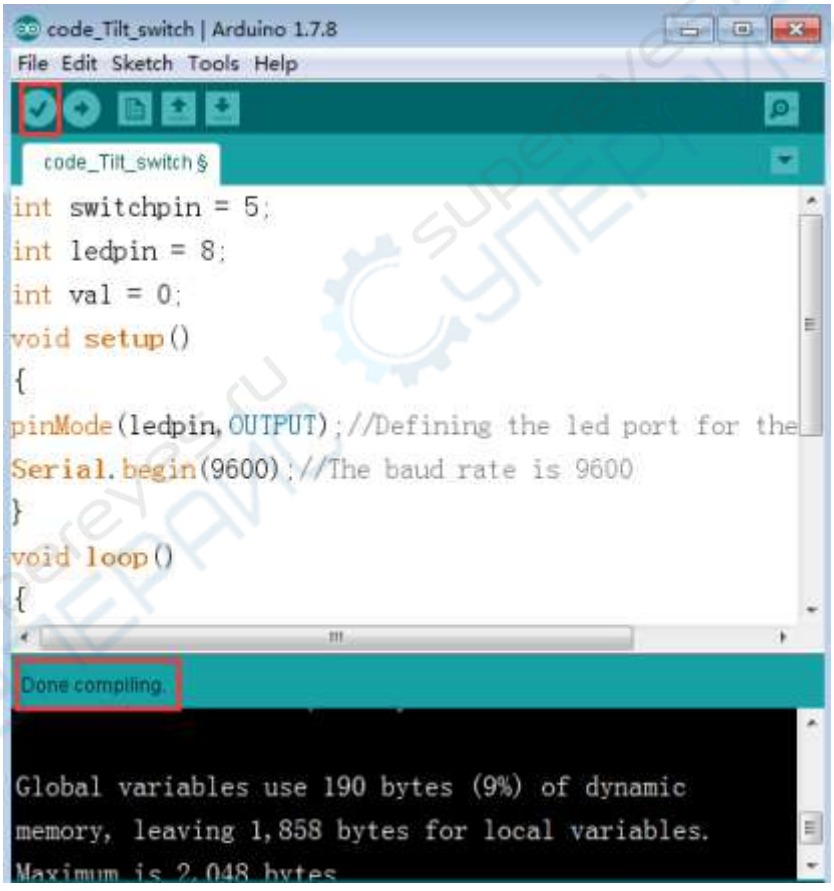


Experimental code analysis:

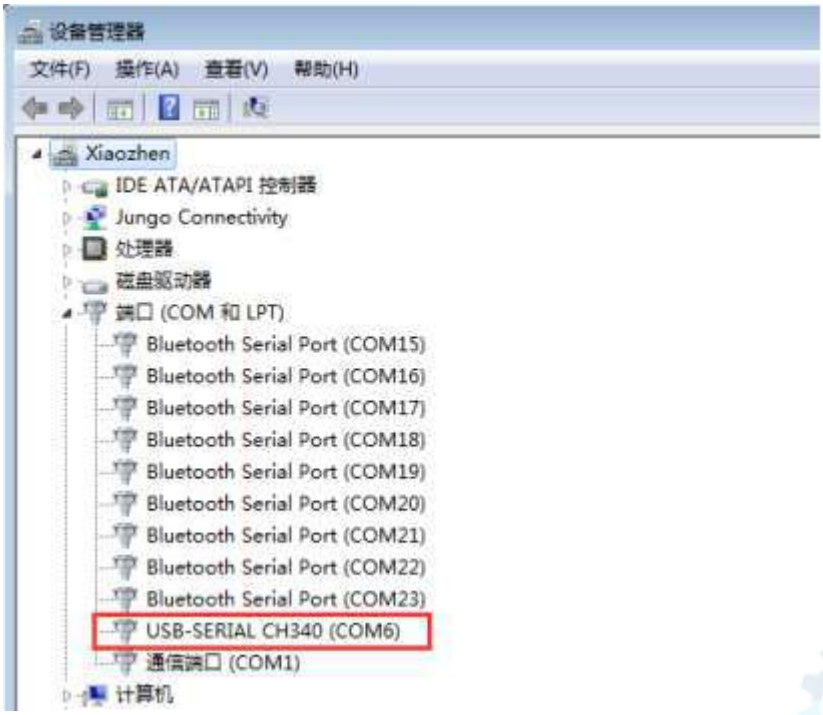
```
int switchpin = 5;
int ledpin = 8;
int val = 0;
void setup()
{
pinMode(ledpin,OUTPUT);//Defining the led port for the output port
Serial.begin(9600);//The baud rate is 9600
}
void loop()
{
      val = analogRead(switchpin);
      if(val>512)//The analog voltage value of 512 is exactly 2.5V
      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);
}
```

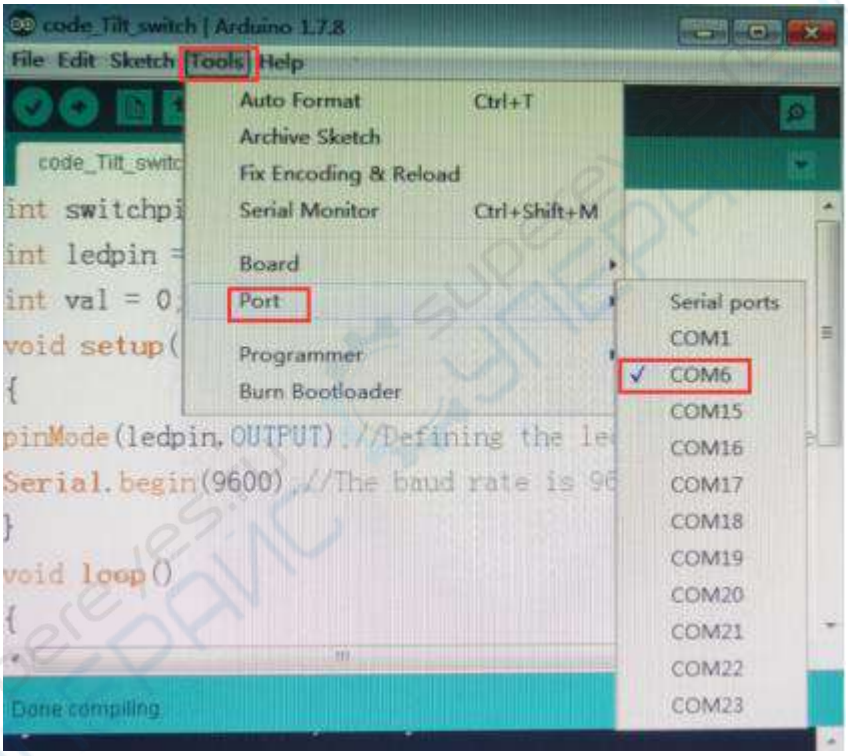
Experimental steps:

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.

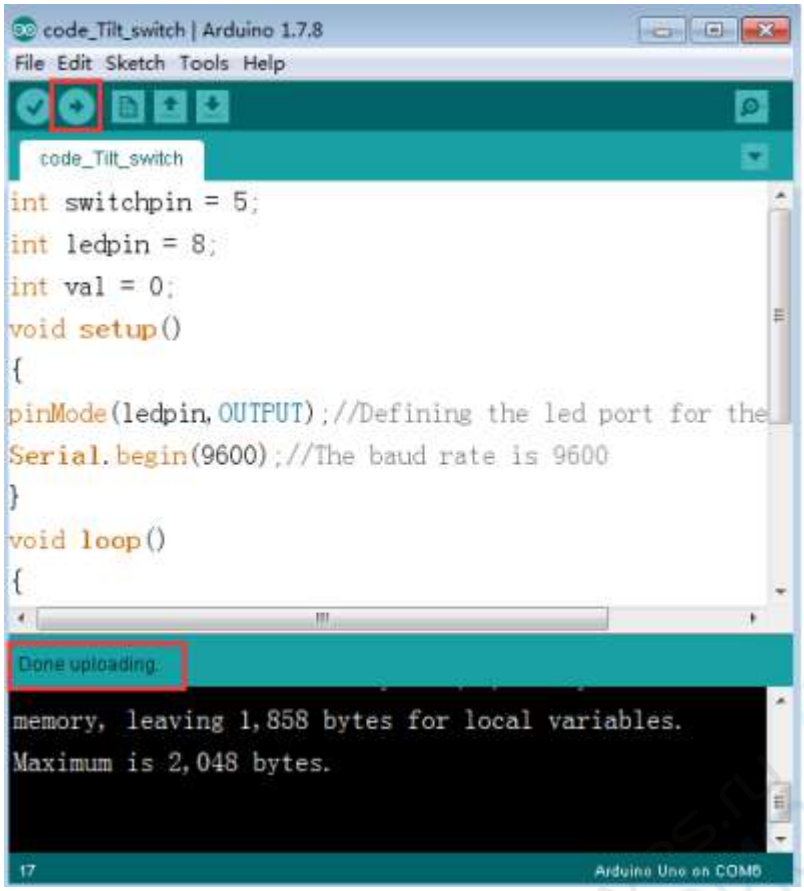


2. In the menu bar of Arduino IDE, we need to select 【Tools】---【Port】--- selecting the port that the serial number displayed by the device manager just now, as shown in the figure below. For example:COM6,as shown in the following figure.

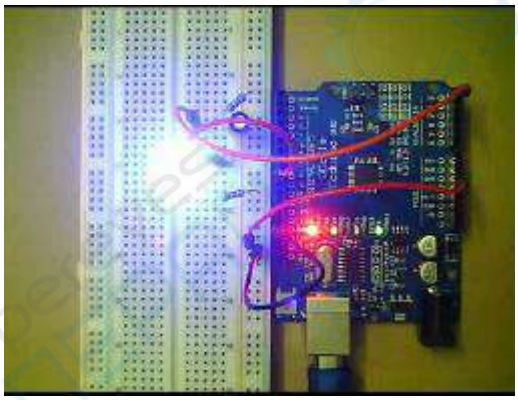


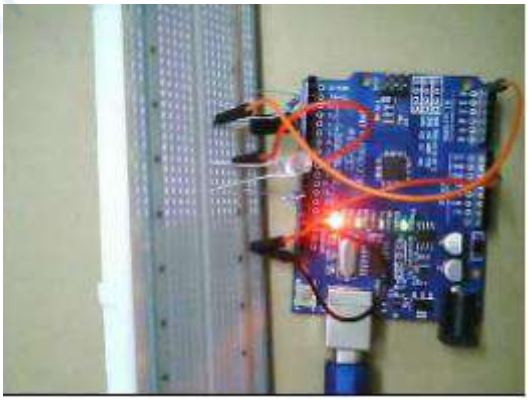


3. After the selection is completed, you need to click "→"under the menu bar to upload thecode to the Arduino UNO board. When the word "Done uploading" appears in the lower left corner, the code has been successfully uploaded to the Arduino UNO board, as shown in the figure below.



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

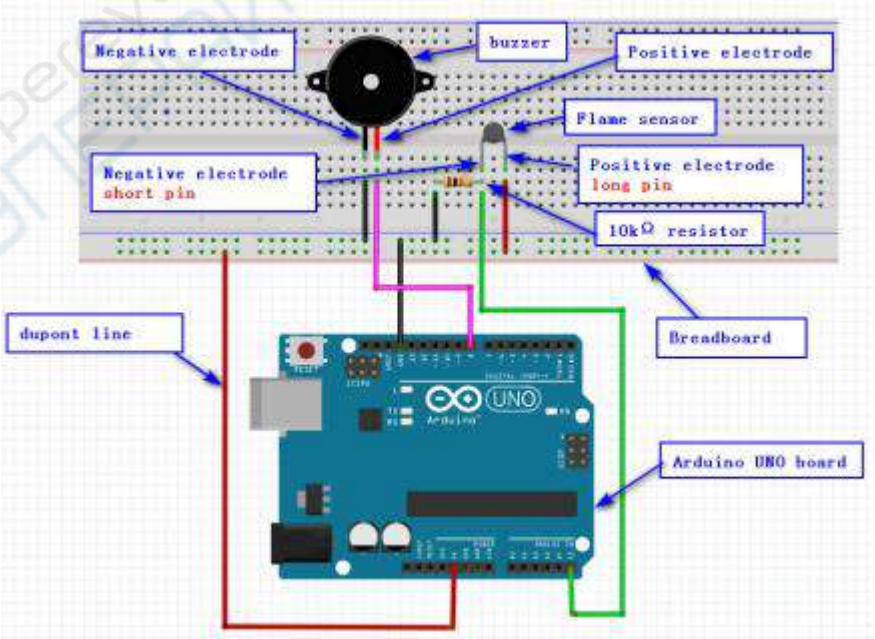
Tilt switch *1

Breadboard *1

Dupont line *1 bunch

Actual object connection diagram：

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

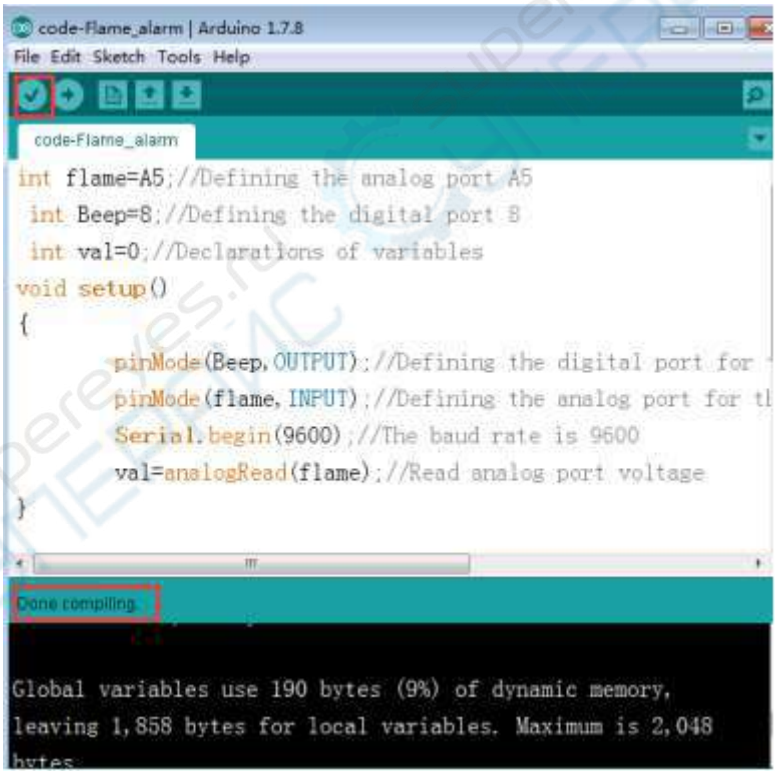


Experimental code analysis:

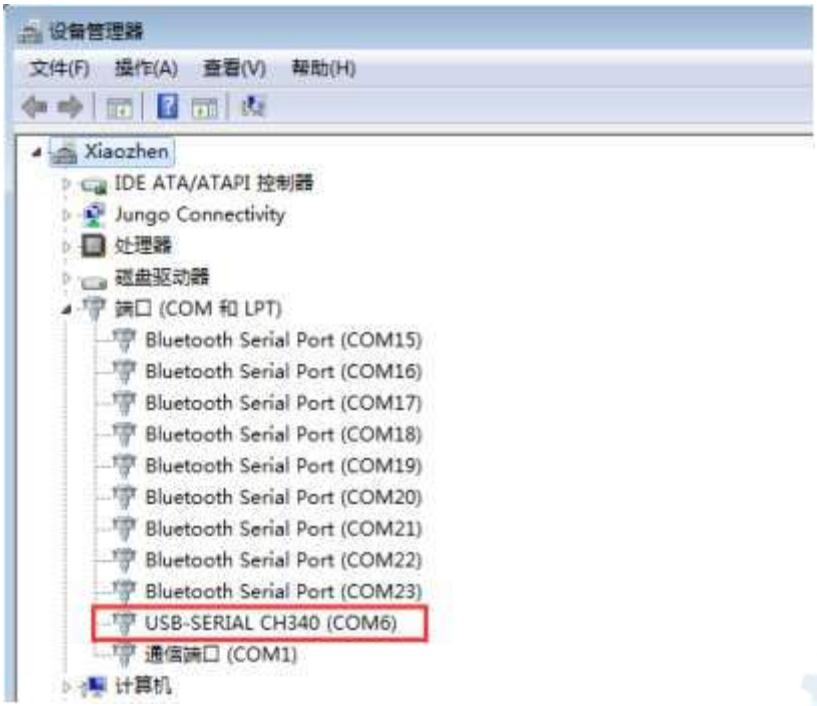
```
int flame=A5;  //Defining the analog port A5
int Beep=8;  //Defining the digital port 8
int val=0;   //Declarations of variables
void setup()
{
pinMode(Beep,OUTPUT);  //Defining the digital port for the output port
pinMode(flame,INPUT);  //Defining the analog port for the input port
Serial.begin(9600);//The 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);
 }
```

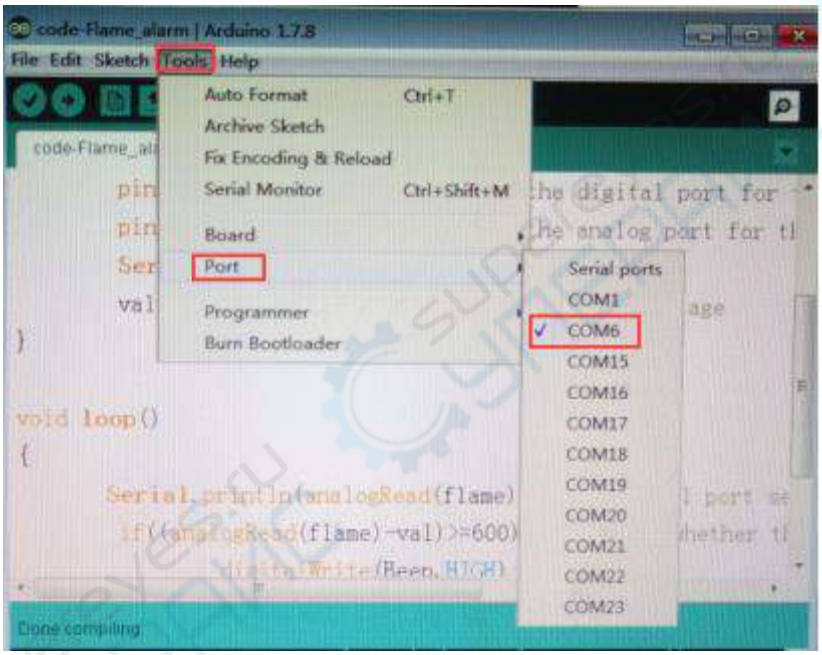
Experimental steps:

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.

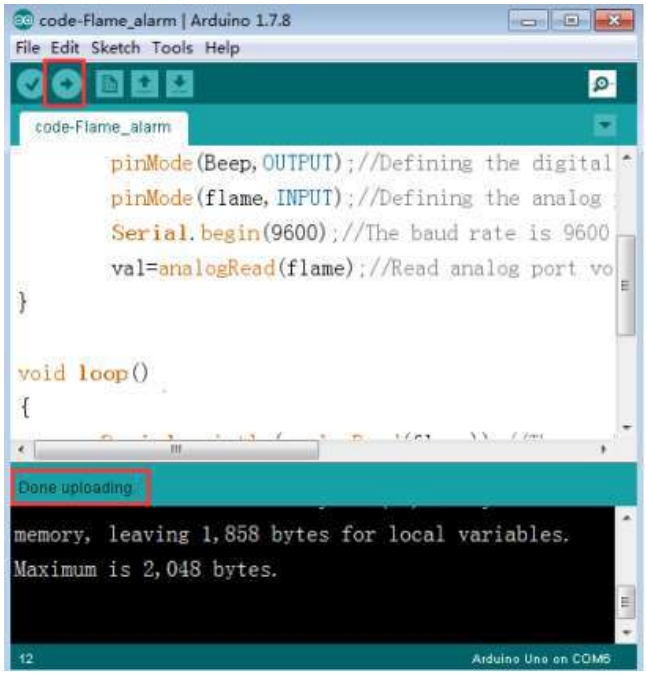


2. In the menu bar of Arduino IDE, we need to select 【Tools】---【Port】--- selecting the port that the serial number displayed by the device manager just now, as shown in the figure below. For example:COM6,as shown in the following figure.

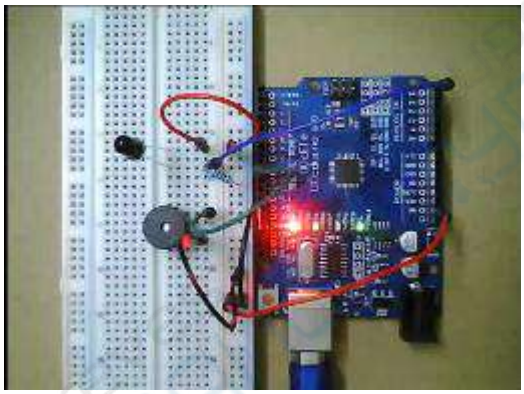


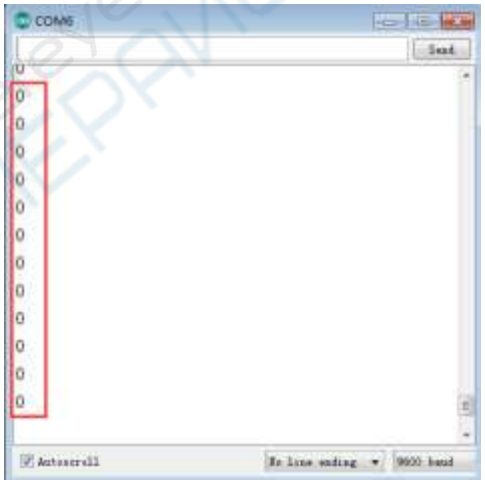


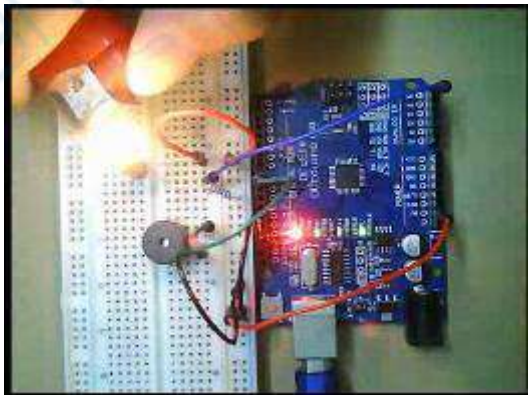
3.After the selection is completed, you need to click "→"under the menu bar to upload the code to the Arduino UNO board. When the word "Done uploading" appears in the lower left corner, thecode has been successfully uploaded to the Arduino UNO board, as shown in the figure below.

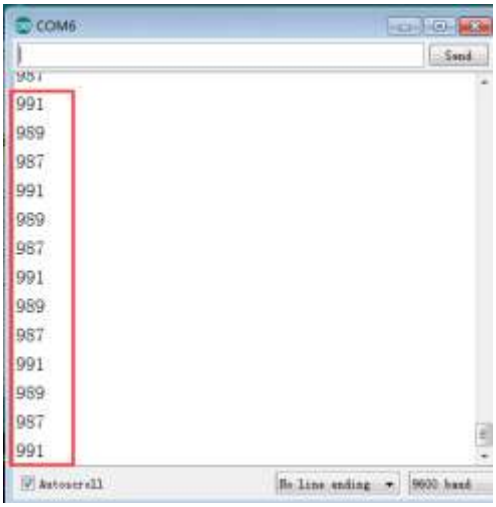


4.After the code is uploaded. When there is no fire source approaching, the circuit is normal. When there is a fire source approaching, the buzzer will make a sound to indicate the alarm. We can also open the serial monitor to observe the change in the value of the flame sensor, as shown in the figure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

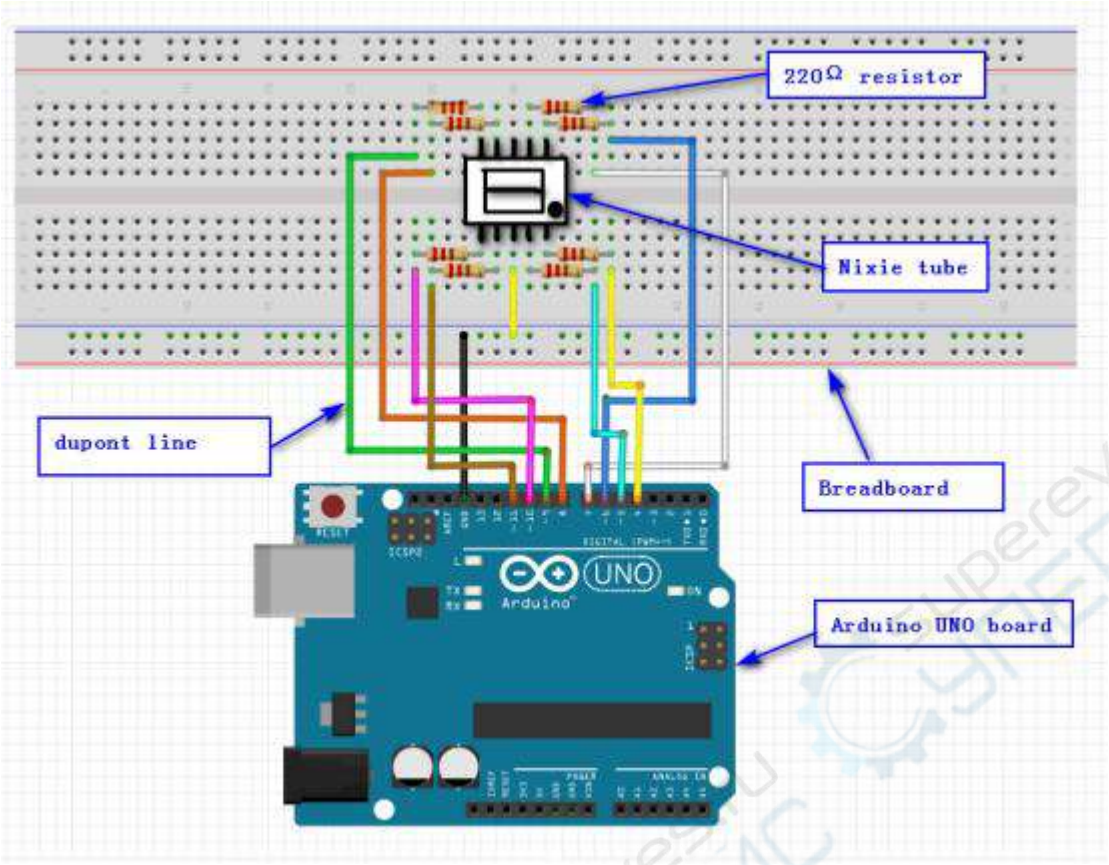
8-segment digital tube *1

Breadboard *1

Dupont line *1bunch

Actual object connection diagram：

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



Experimental code analysis:

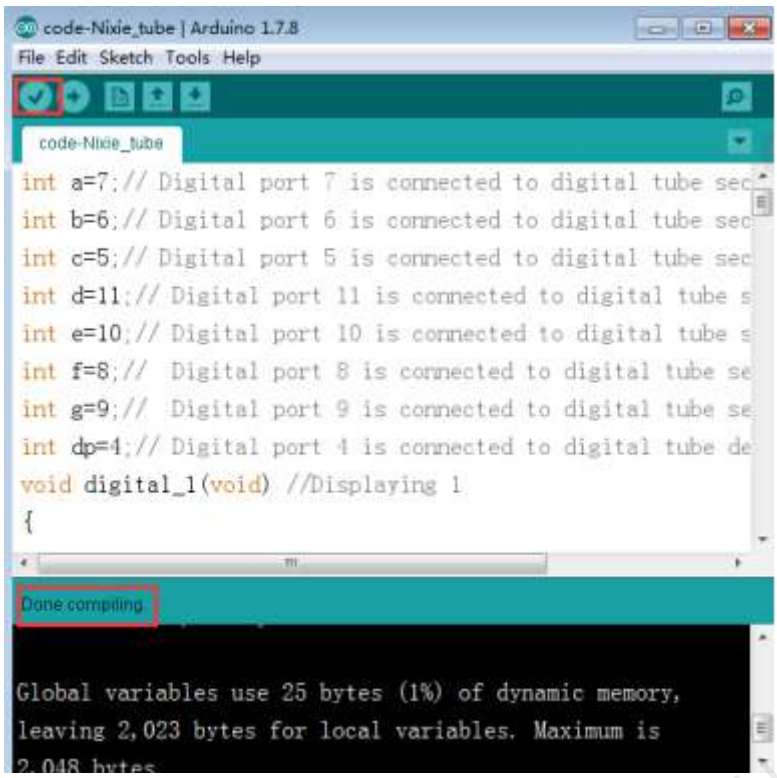
```
int a=7;  // Digital port 7 is connected to digital tube section a
int b=6;  // Digital port 6 is connected to digital tube section b
int c=5;  // Digital port 5 is connected to digital tube section c
int d=11;  // Digital port 11 is connected to digital tube section d
int e=10;  //Digital port 10 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);
}
```

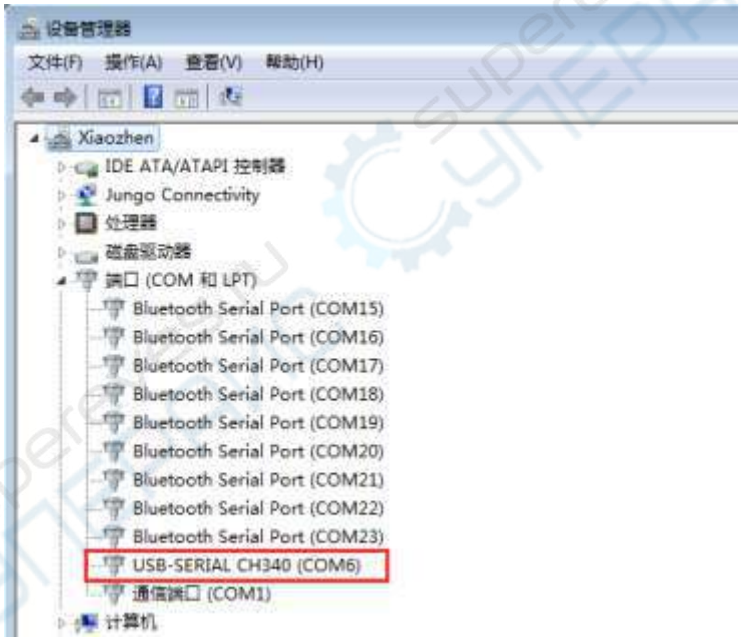
```
void digital_8(void) //Displaying 8
{
unsigned char j;
for(j=5;j<=11;j++)
digitalWrite(j,HIGH);
digitalWrite(dp,LOW);
}
void digital_9(void) //Displaying 9
{
digitalWrite(a,HIGH);
digitalWrite(b,HIGH);
digitalWrite(c,HIGH);
digitalWrite(d,HIGH);
digitalWrite(e,LOW);
digitalWrite(f,HIGH);
digitalWrite(g,HIGH);
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);
  }
}
```

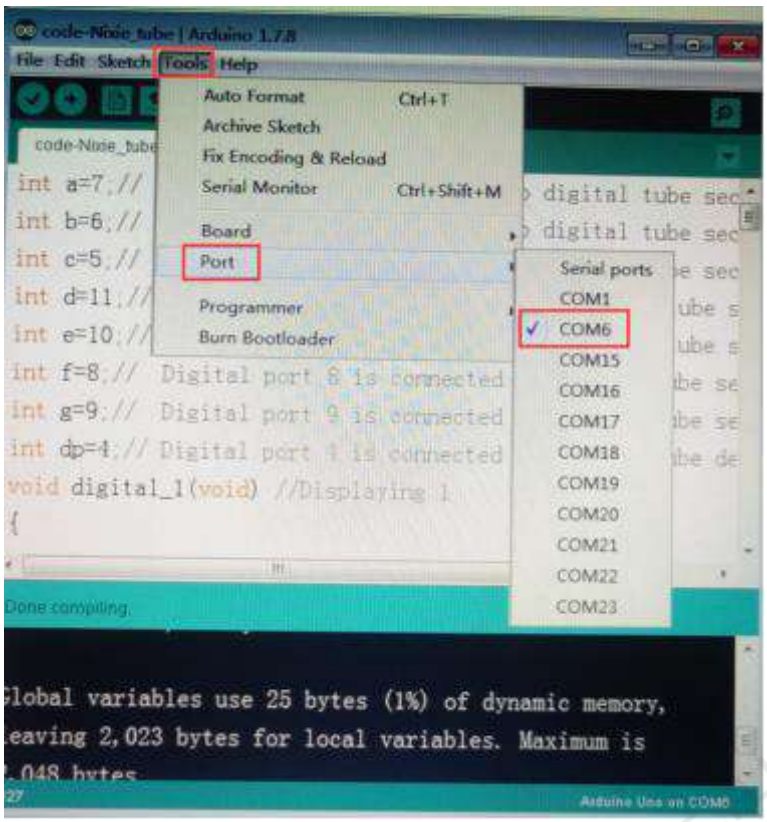
Experimental steps:

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.

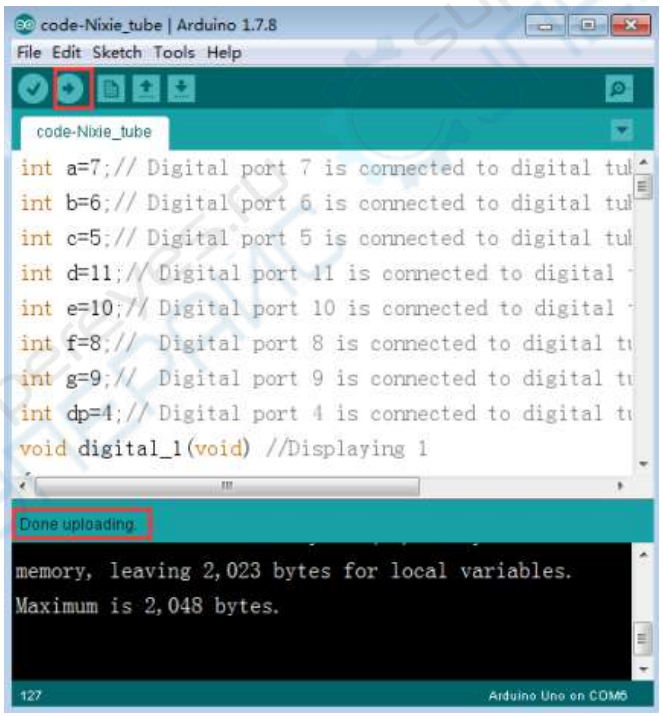


2. In the menu bar of Arduino IDE, we need to select 【Tools】---【Port】--- selecting the port that the serial number displayed by the device manager just now, as shown in the figure below. For example:COM6,as shown in the following figure.

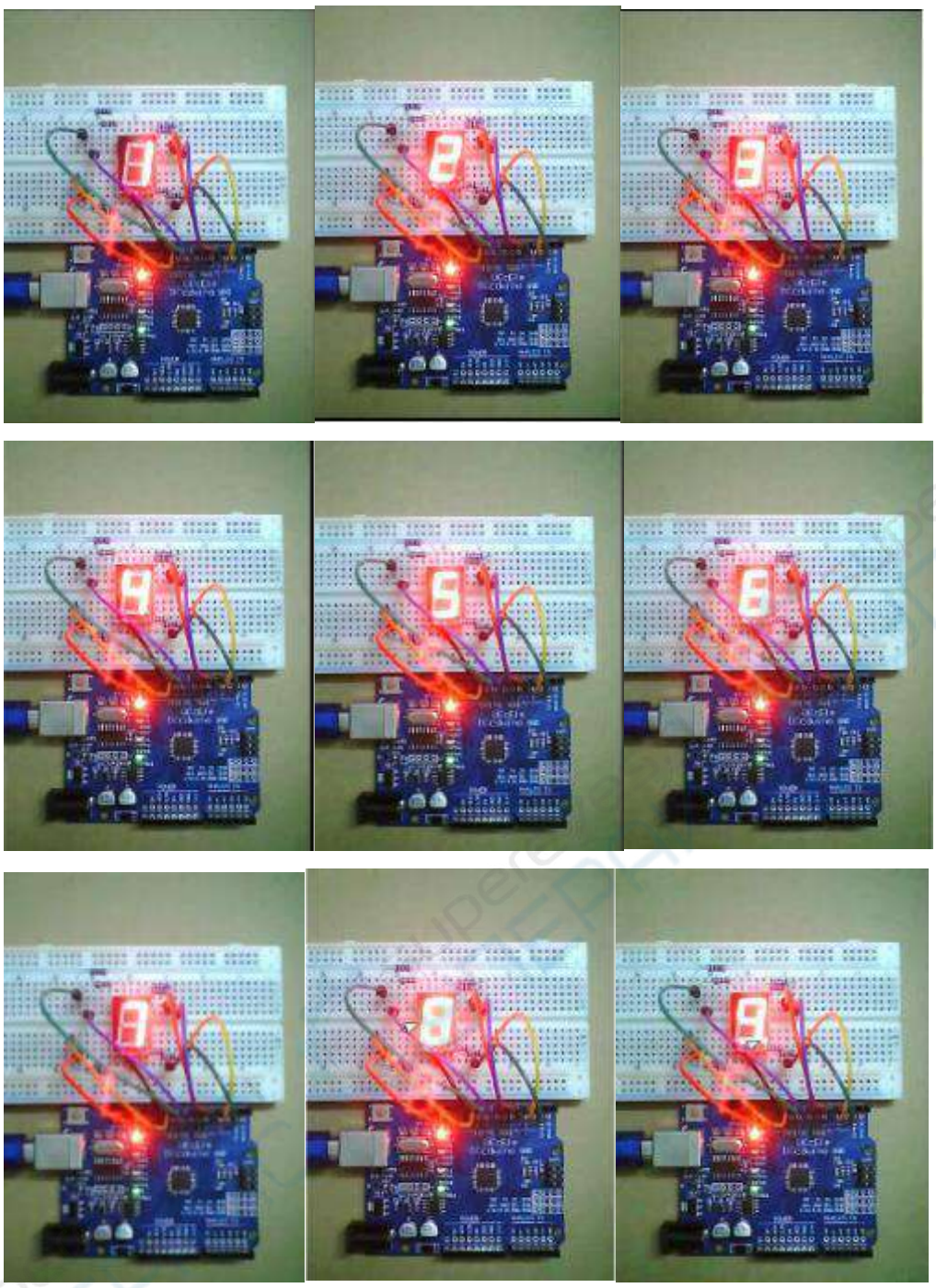




3. After the selection is completed, you need to click "→"under the menu bar to upload thecode to the Arduino UNO board. When the word "Done uploading" appears in the lower left corner, the code has been successfully uploaded to the Arduino UNO board, as shown in the figure below.



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 display4 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 the8-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 cathodeNixie 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

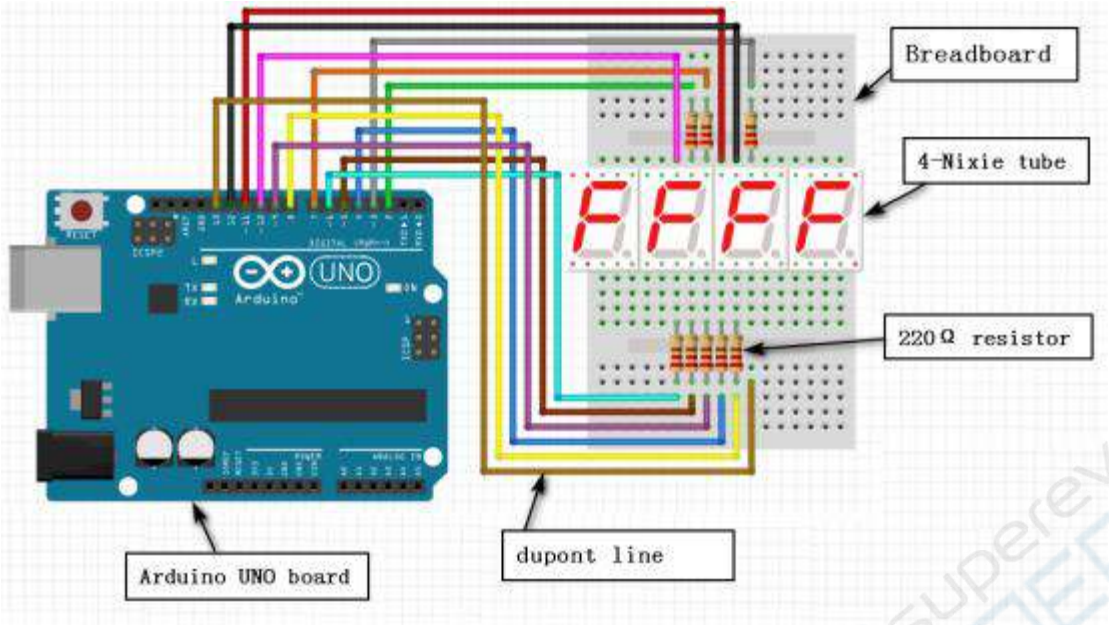
4bit 8-segment digital tube *1

Breadboard *1

dupont line *1bunch

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);
```

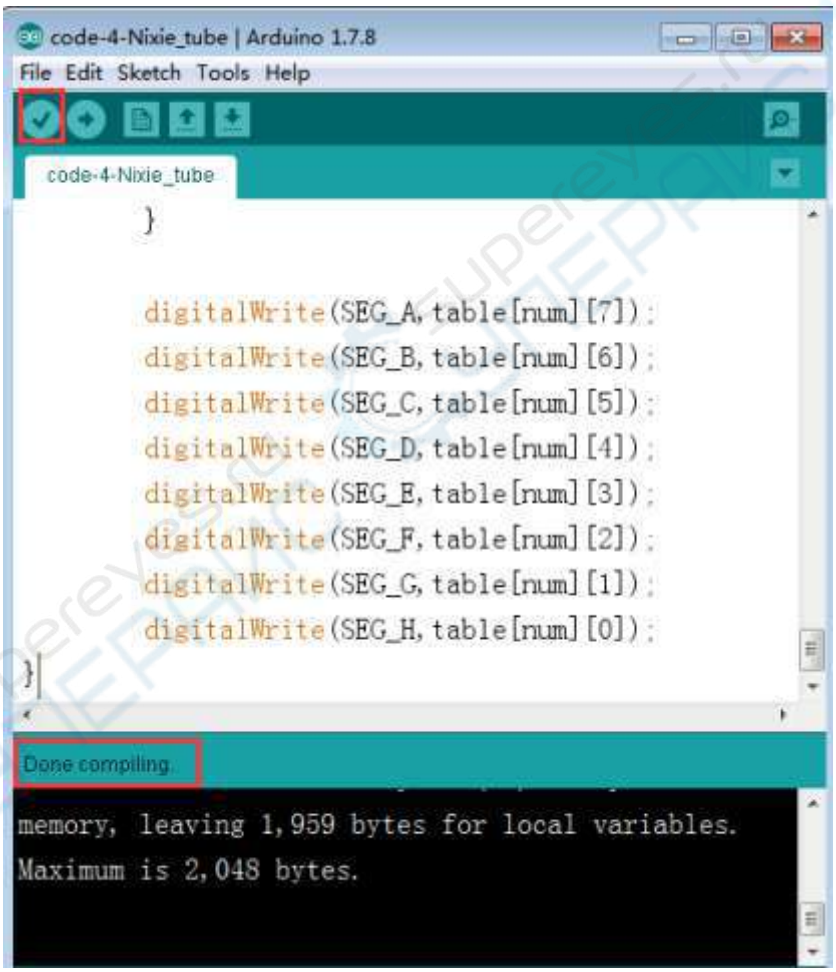
```
pinMode(SEG_F,OUTPUT);
pinMode(SEG_G,OUTPUT);
pinMode(SEG_H,OUTPUT);

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);
```

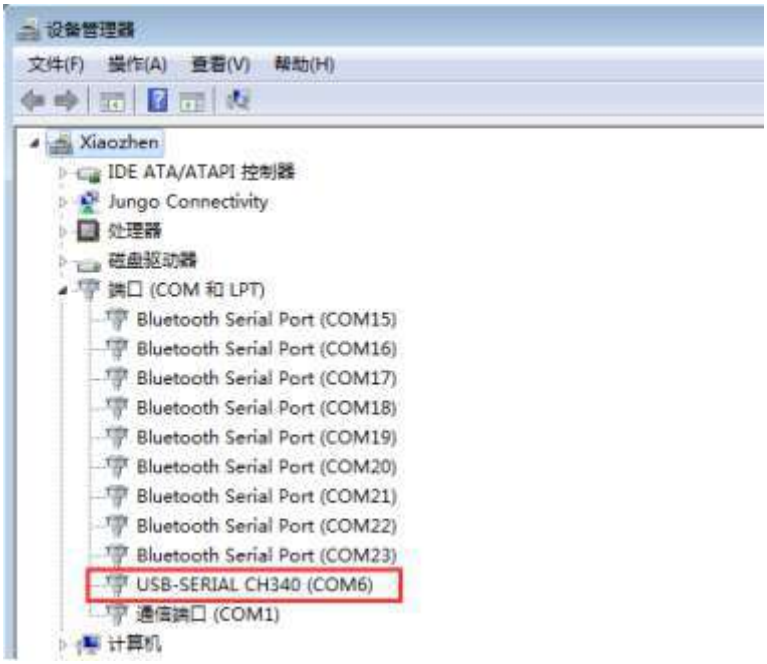
```
digitalWrite(COM2,HIGH);
digitalWrite(COM3,HIGH);
digitalWrite(COM4,LOW);  //Fourth bit of the Nixie tube
break;
default:break;
}
digitalWrite(SEG_A,table[num][7]);
digitalWrite(SEG_B,table[num][6]);
digitalWrite(SEG_C,table[num][5]);
digitalWrite(SEG_D,table[num][4]);
digitalWrite(SEG_E,table[num][3]);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

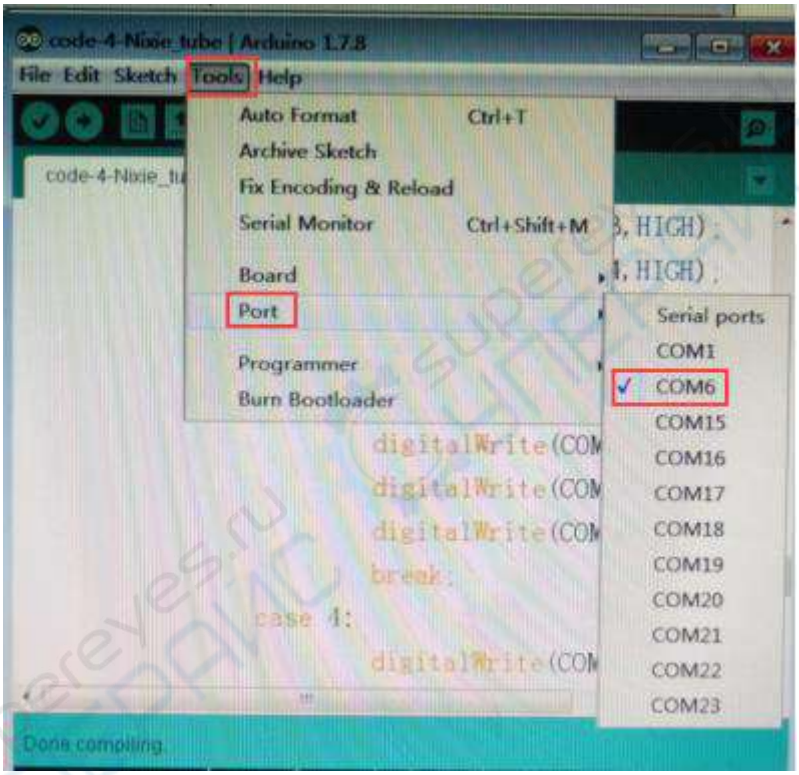
Experimental steps:

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.

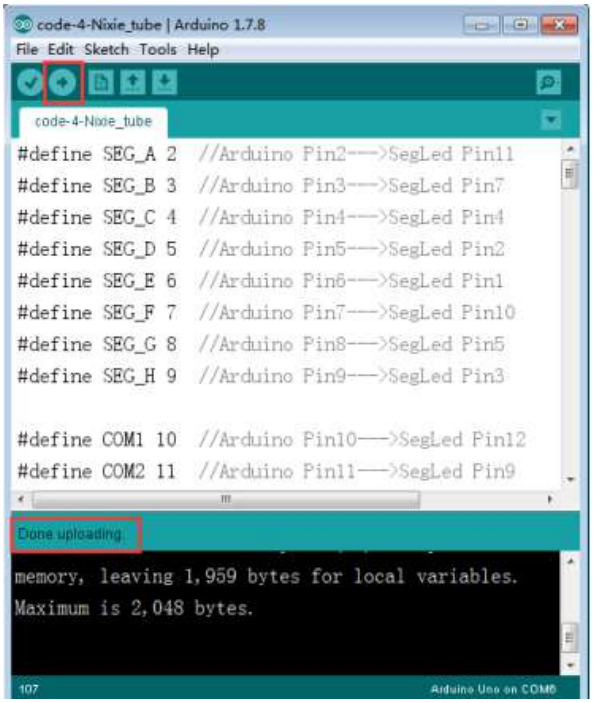


2. In the menu bar of Arduino IDE，we need to select the 【Tools】---【Port】--- select the port that the serial number displayed by the device manager just now.for example:COM6,as shown in the following figure.

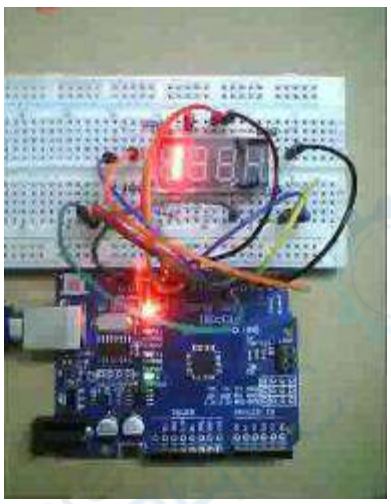


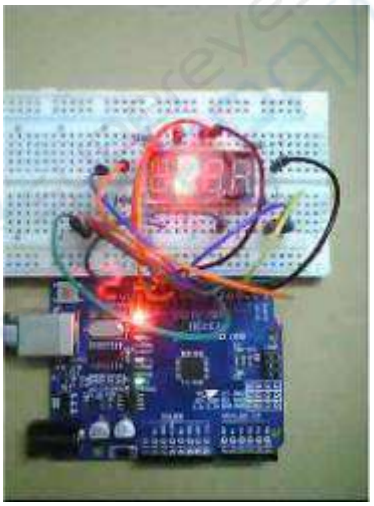


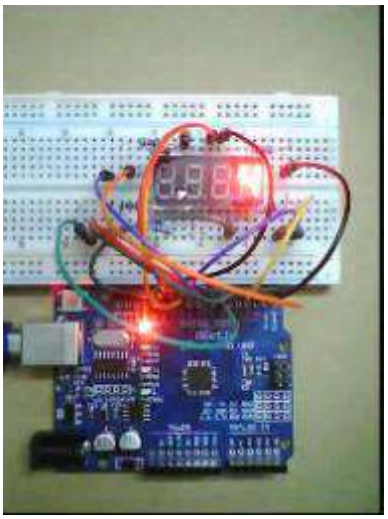
3. After the selection is completed, click "→"under the menu bar,and upload the code to the Arduino UNO board, when appears to Done uploading on the lower left corner , that means that the code has been successfully uploaded to the Arduino UNO board, as shown in the following figure.



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 display4 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

220Ω resistor *8

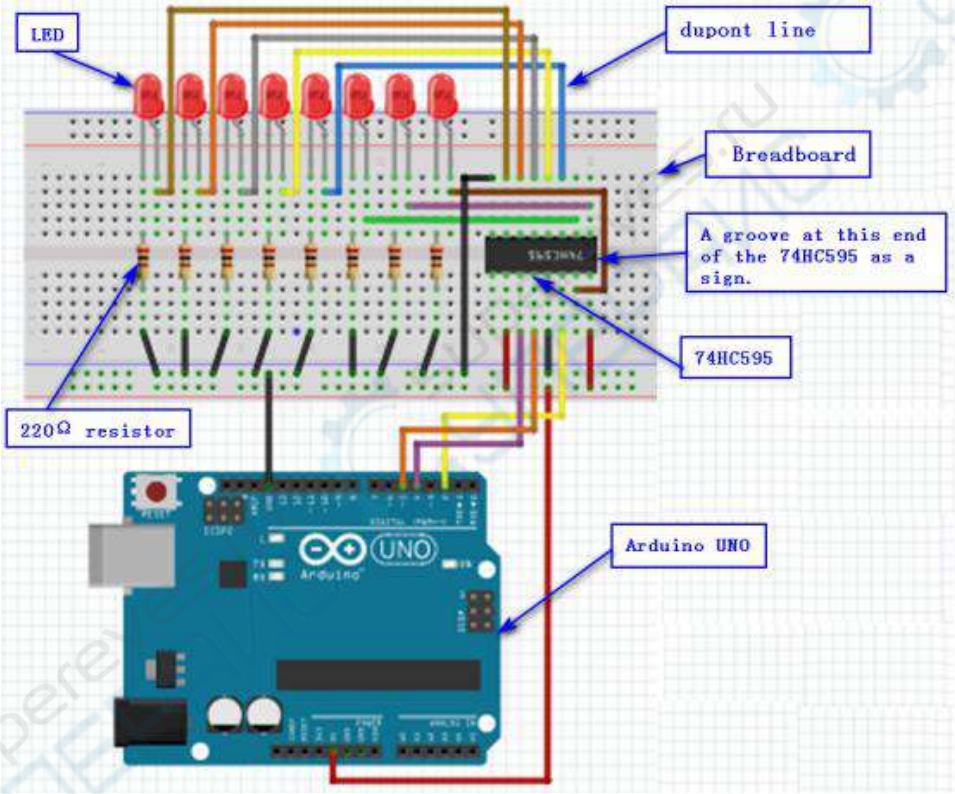
LED *8

Breadboard *1

Dupont line *1bunch

Material object connection diagram：

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



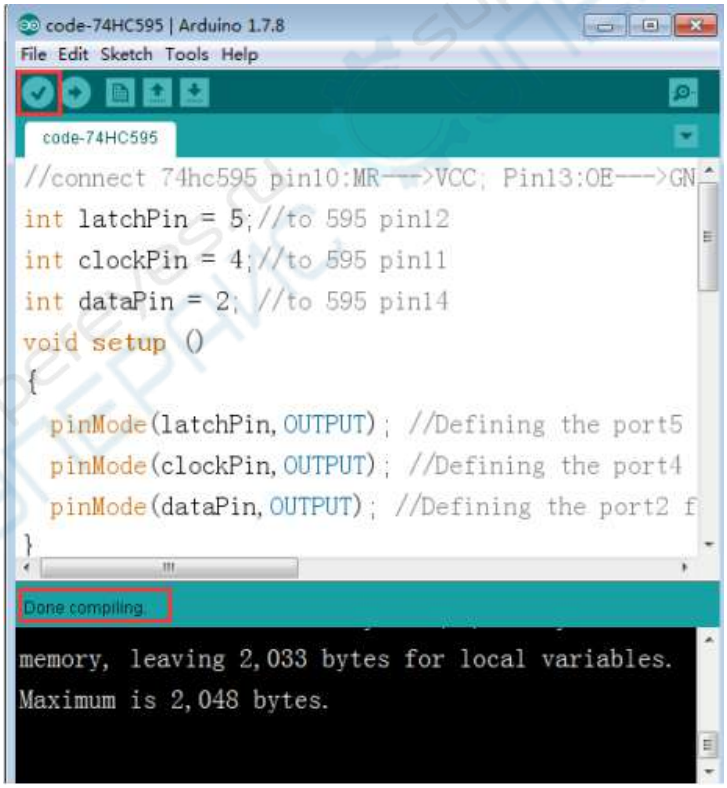
Experimental code analysis:

```
//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
}
```

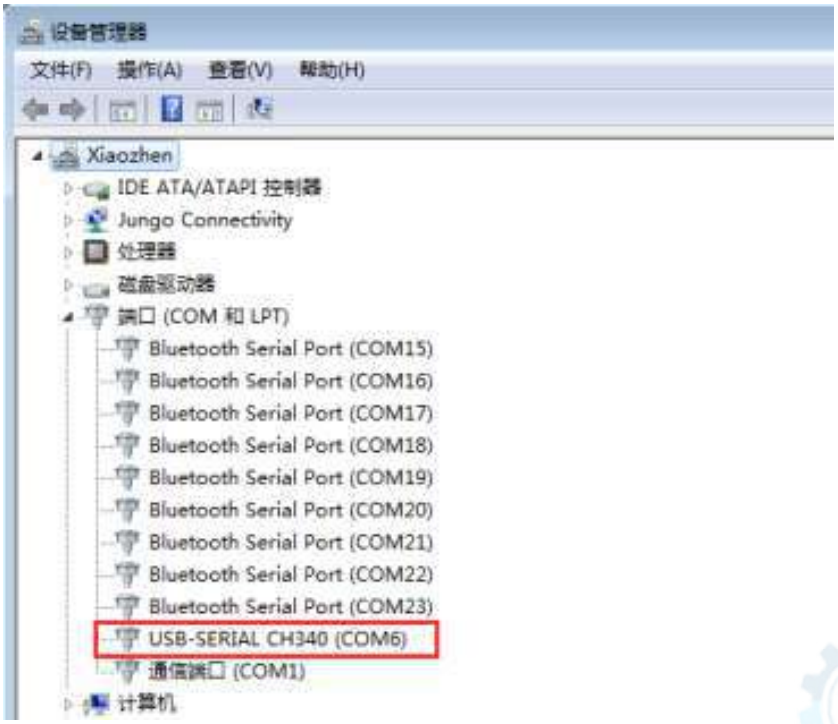
```
void loop()
{
  for(int a=0; a<256; a++)  //The meaning of this loop is to let a variable increase by 1 until it is equal to 256.
                                    //The following activities are performed every cycle.
  {
    digitalWrite(latchPin,LOW); //Giving a low level to the port ST_CP indicates 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 youself.
    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
  }
}
```

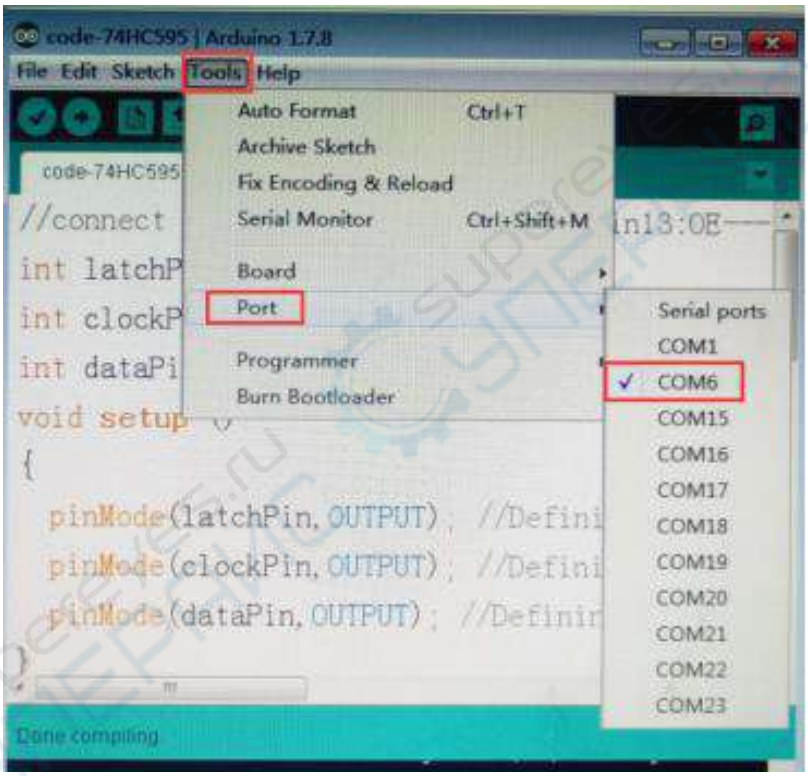
Experimental steps:

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.

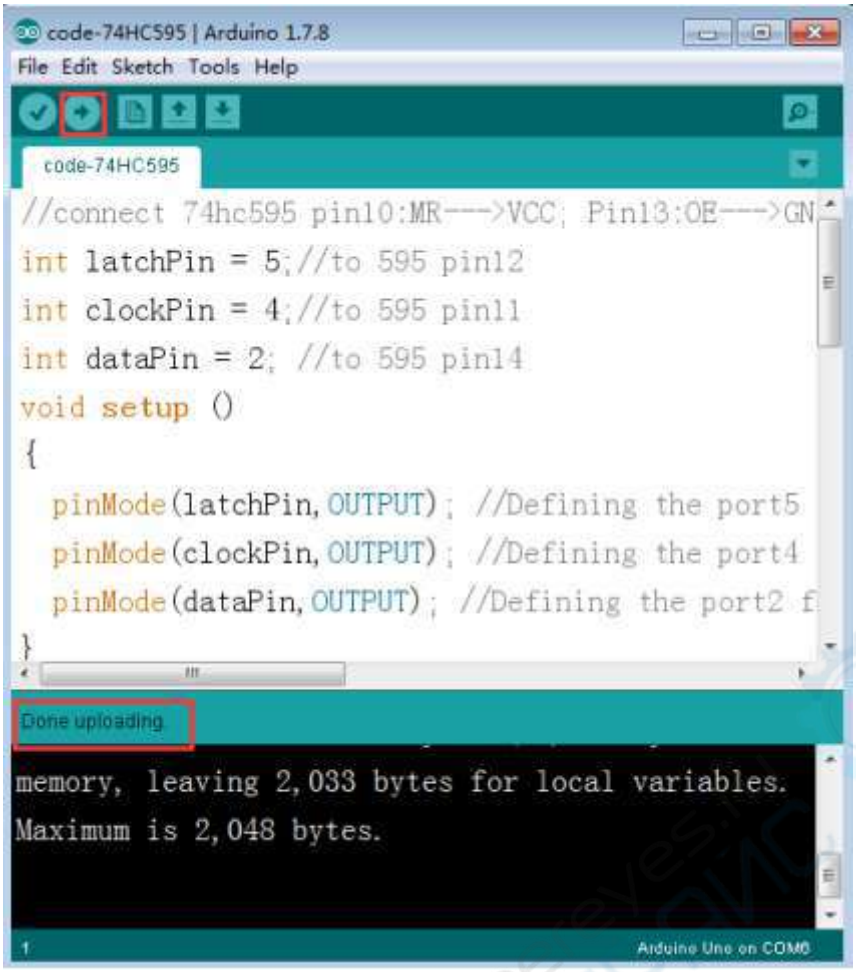


2. In the menu bar of Arduino IDE, we need to select the 【Tools】---【Port】--- select the port that the serial number displayed by the device manager just now.for example:COM6,as shown in the following figure.

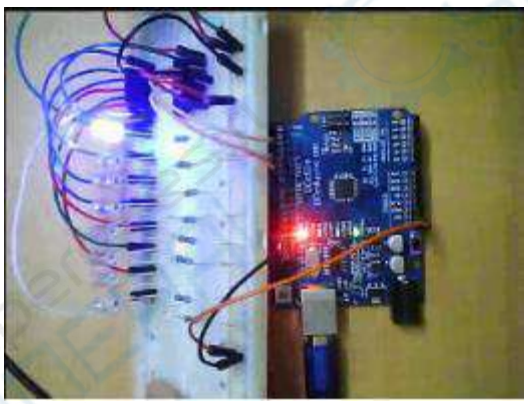


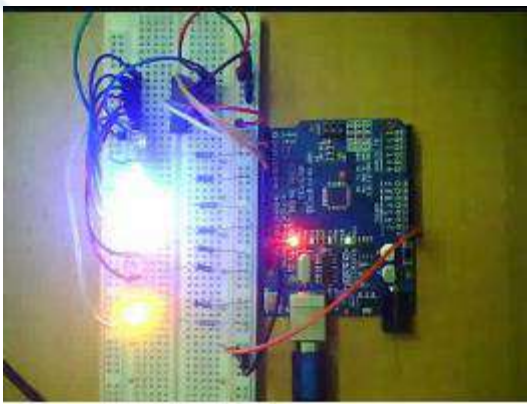


3. After the selection is completed, you need to click “→”under the menu bar,and upload thecode to the Arduino UNO board, when appears to Done uploading on the lower left corner , that means that the code has been successfully uploaded to the Arduino UNO board, as shown in the following figure.



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





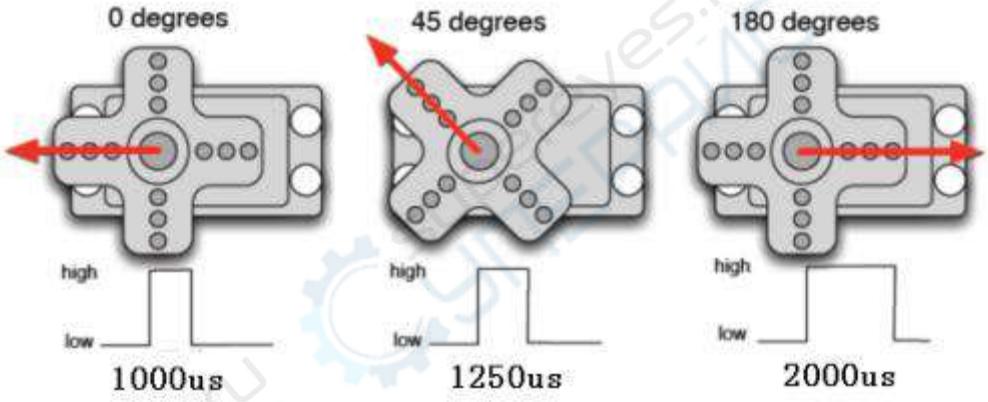
18- servo control

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°～180° 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

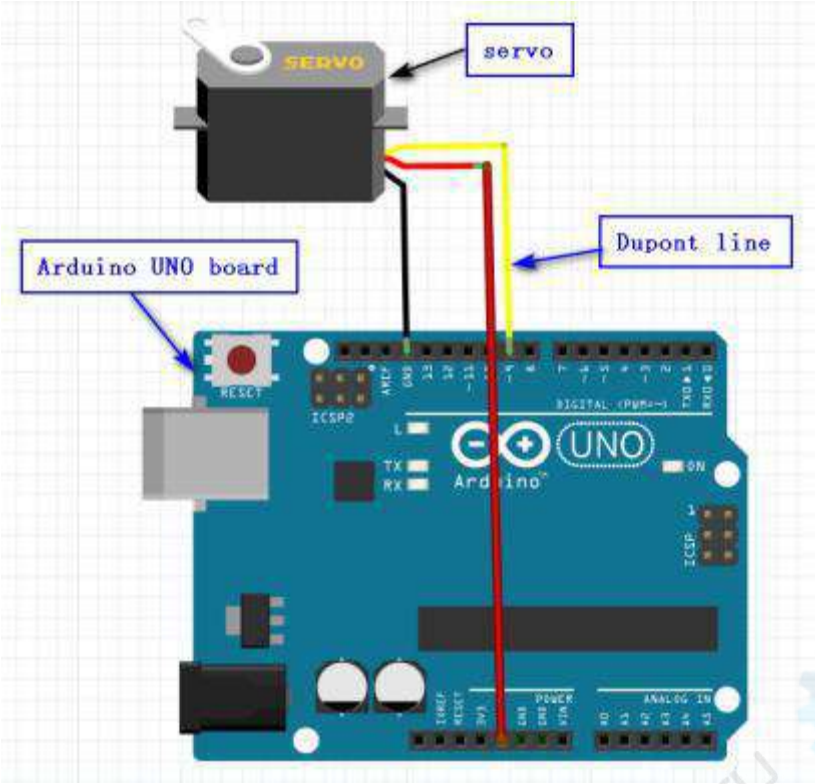
USB cable *1

Servo *1

Dupont line *1 bunch

Actual object connection diagram：

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

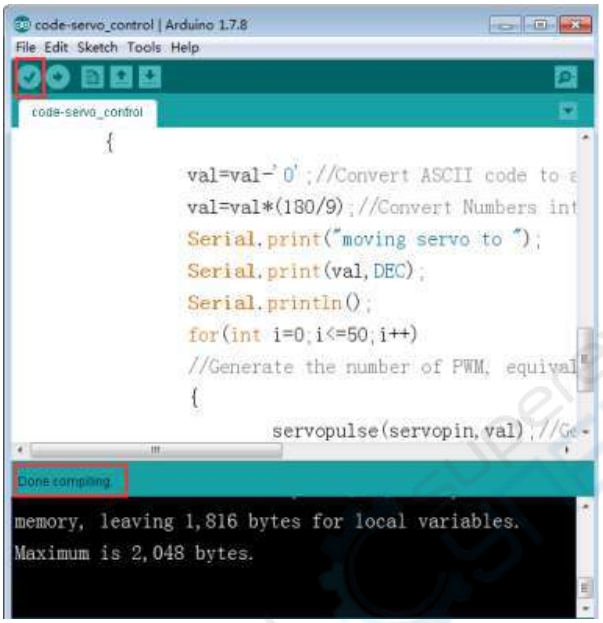


Experimental code analysis:

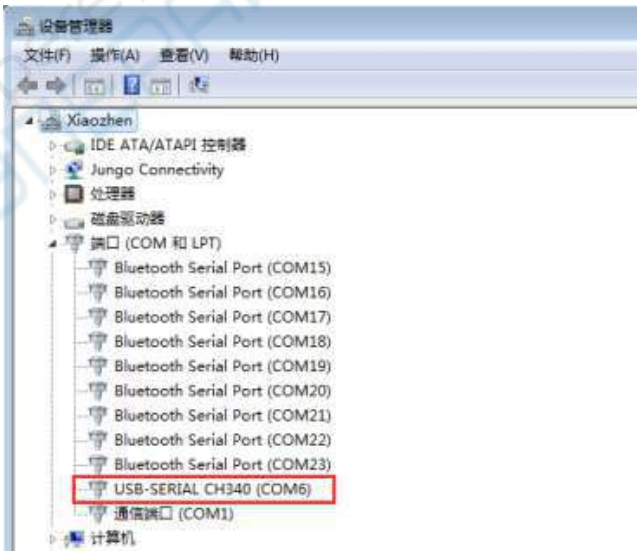
```
//UART send 1~9==>20~180 degree
int servopin=9;//Defining the port 9 for the servopin
int myangle;//Define Angle variable
int pulsewidth;//Define the pulse width variable
int val;
void servopulse(int servopin,int myangle)
/*A pulse function is defined to generate PWM values by simulation */
{
  pulsewidth=(myangle*11)+500;//Convert 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_seral_simple ready" ) ;
}
void loop()
{
val=Serial.read();//Reading the data received by the serial port
if(val>'0'&&val<='9')//Determineing 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 ");
```

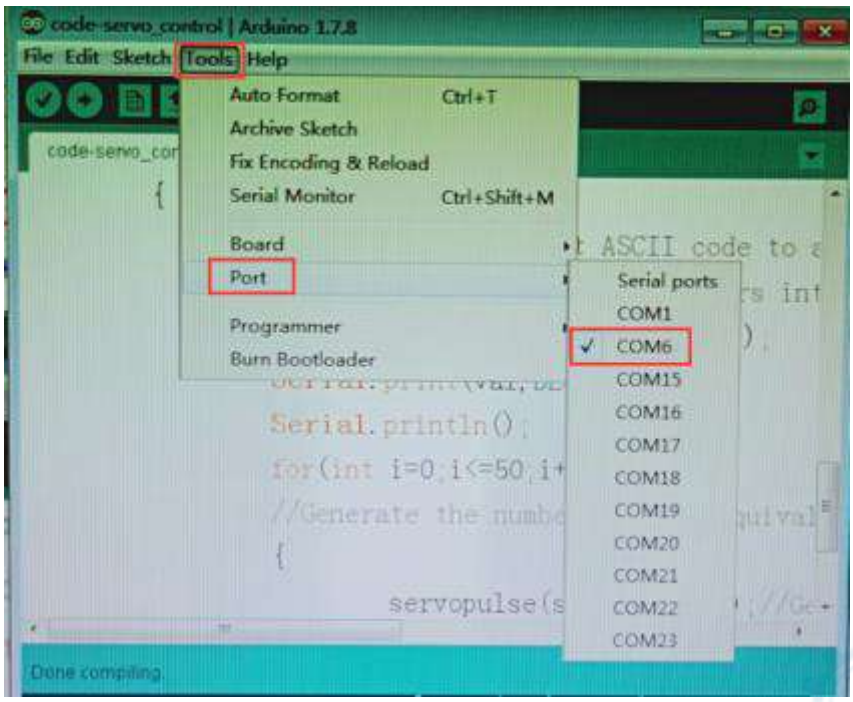
Serial.print(val,DEC);
Serial.println();
for(int i=0;i<=50;i++)
//Generate the number of PWM, equivalent delay to ensure that the response Angle can be turned
{
servopulse(servopin,val);//Generate PWM values by simulation
}
}
}
Experimental steps:

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.

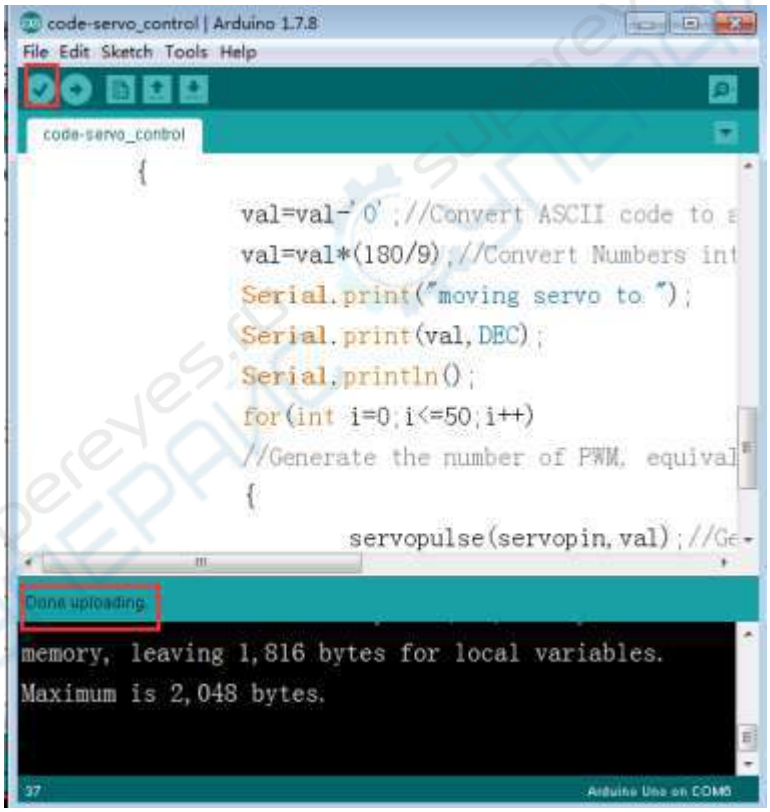


2. In the menu bar of Arduino IDE，you need to select the 【Tools】---【Port】---select the port that the serial number displayed by the device manager just now.for example:COM6,as shown in the following figure.

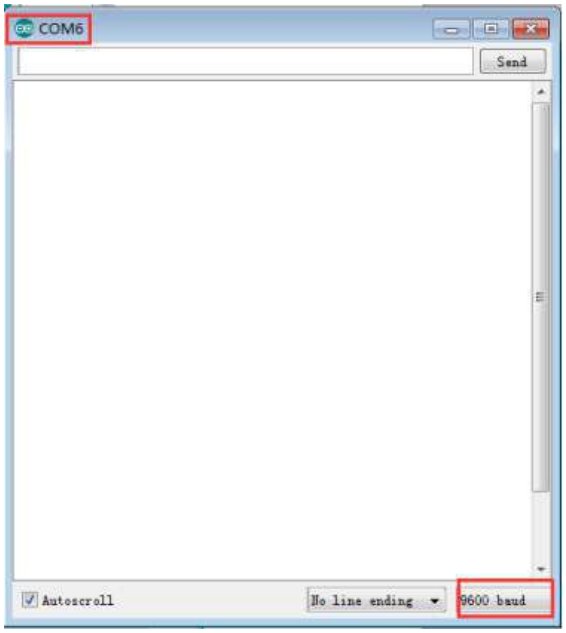




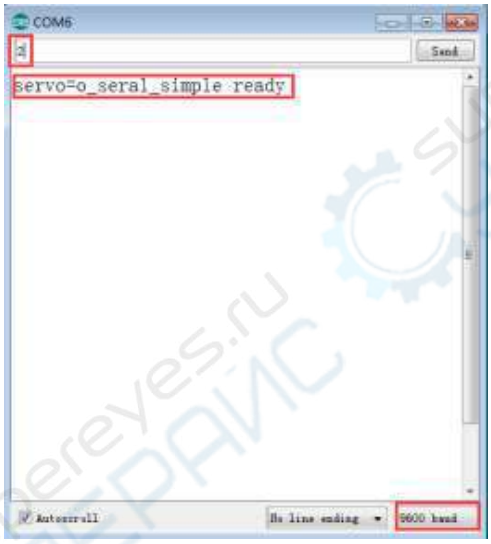
3. After the selection is completed, you need to click "→"under the menu bar,and upload the code to the Arduino UNO board, when appears to Done uploading on the lower left corner , that means that the code has been successfully uploaded to the Arduino UNO board, as shown in the following figure.

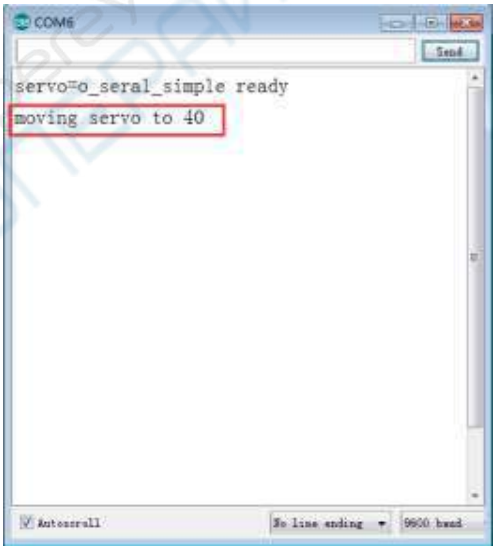


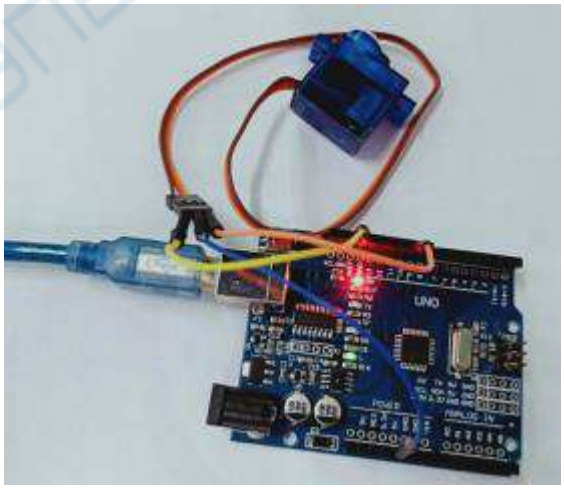
4. You can open the serial port monitor on the top right corner of Arduino IDE, A serial port of Arduino port will appear, and the baud rate is set to 9600 on the lower right corner, as shown in the following figure.



5.After the code is uploaded, we open the serial port monitor of Arduino IDE, you can see the words "servo=o_seral_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

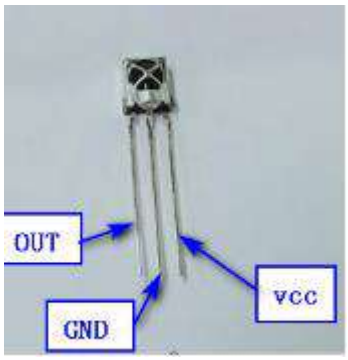
The purpose of the experiment:

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

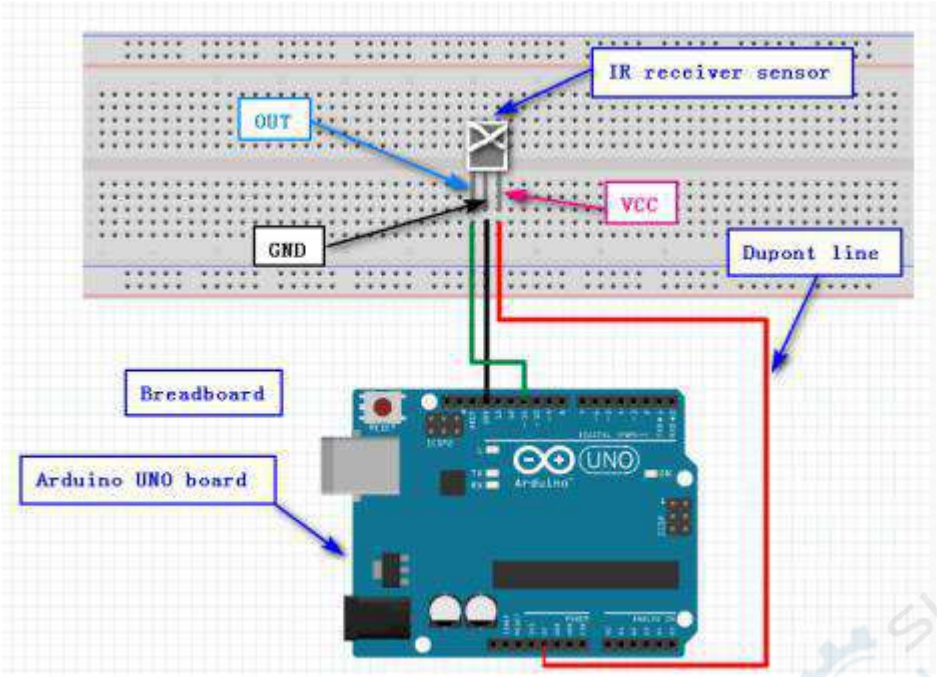
IR remote controller *1

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:

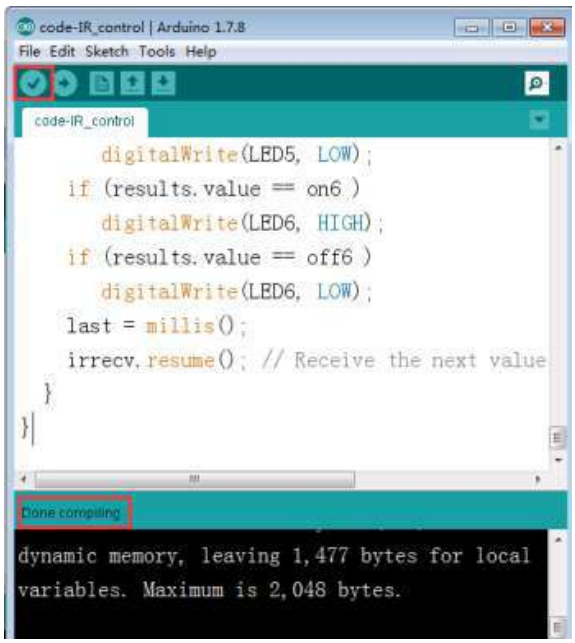
```
#include <IRremote.h>//Including infrared library
int RECV_PIN = 11; // Declarations of port
int LED1 = 2;
int LED2 = 3;
int LED3 = 4;
int LED4 = 5;
int LED5 = 6;
int LED6 = 7;
long on1  = 0x00FF6897;//Code the example to match the send
long off1 = 0x00ff30CF;
long on2 = 0x00FF9867;
long off2 = 0x00FF18E7;
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
```

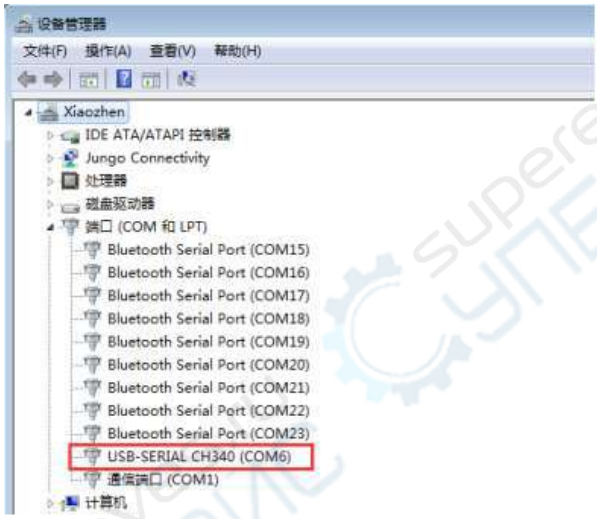
```
Serial.begin(9600); //The baud rate is 9600
irrecv.enableIRIn(); // Start the receiver
}
int on = 0;
unsigned long last = millis();
void loop()
{
  if (irrecv.decode(&results)) //Calling the library function: decode
   {
    // If it's been at least 1/4 second since the last
    // IR received, toggle 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
  }
}
```

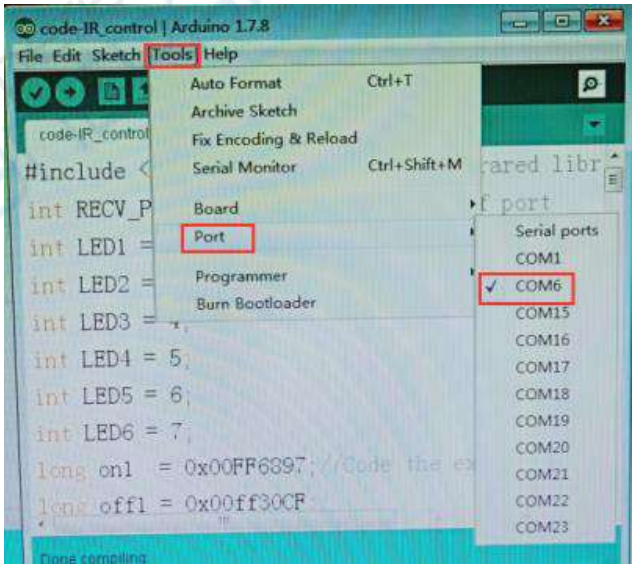
Experimental steps:

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.

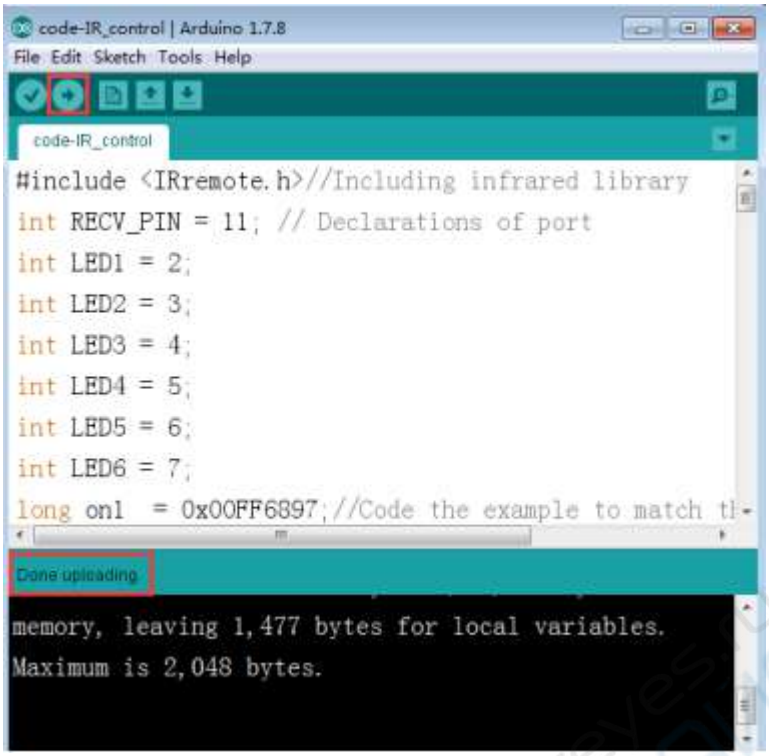


2. In the menu bar of Arduino IDE, select the 【Tools】---【Port】--- select the port that the serial number displayed by the device manager just now.for example:COM6,as shown in the following figure.

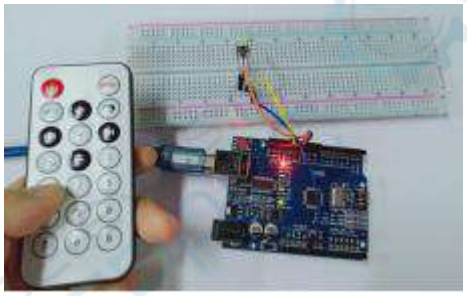


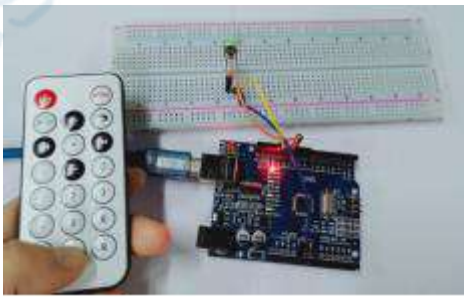


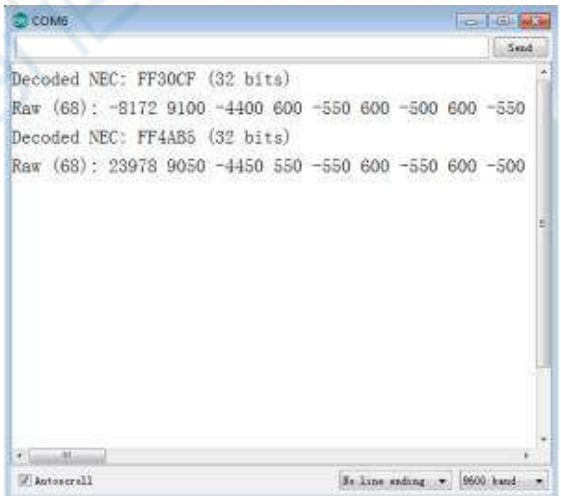
3. After the selection is completed, you need to click "→"under the menu bar,and upload the code to the Arduino UNO board, when appears to Done uploading on the lower left corner , that means that the code has been successfully uploaded to the Arduino UNO board, as shown in the following figure.



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).







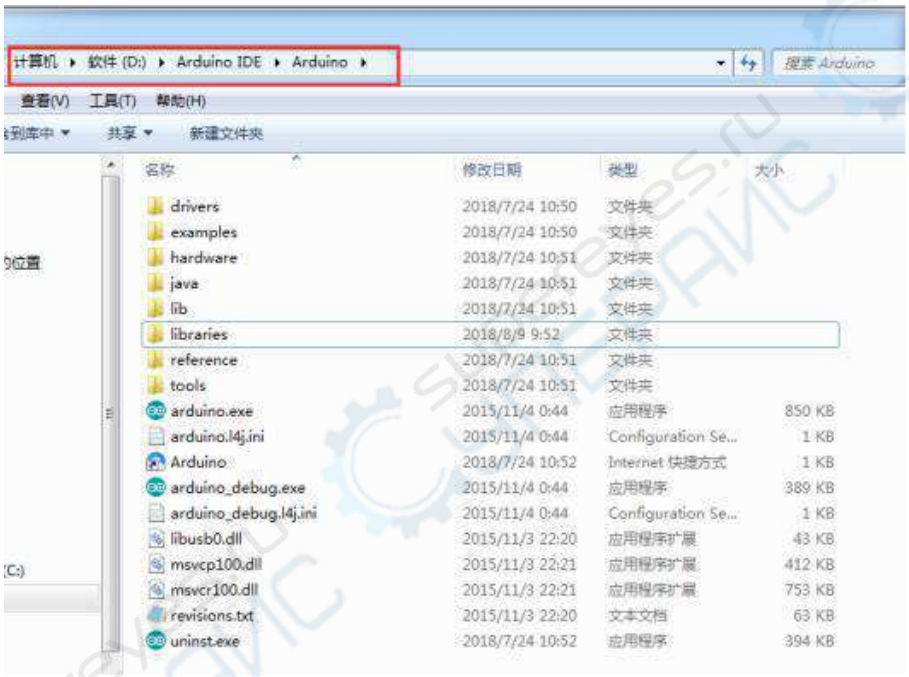
Steps to add a library file

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.

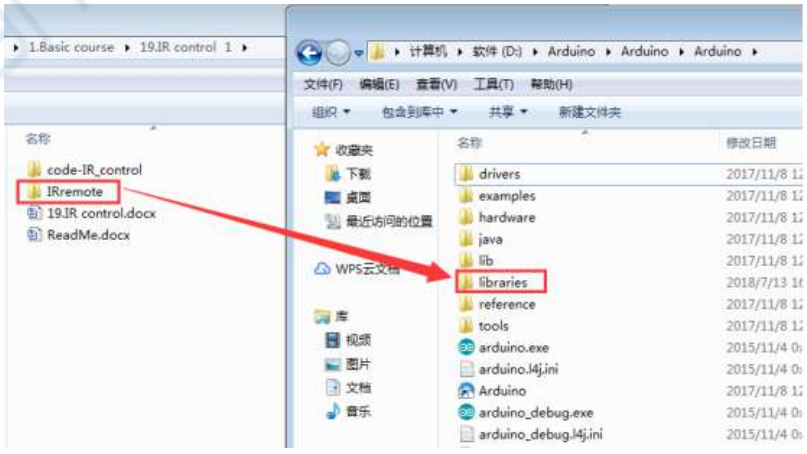
2.You need to find the installation path of Arduino. As shown in the figure below.(just for example)

This is my Arduino installation path.

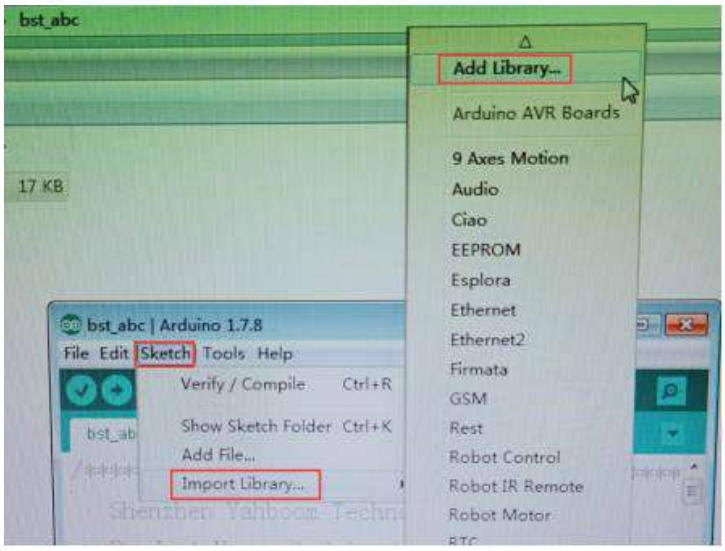


3. You need to copy this file into the libraries folder in the Ardunio installation path.

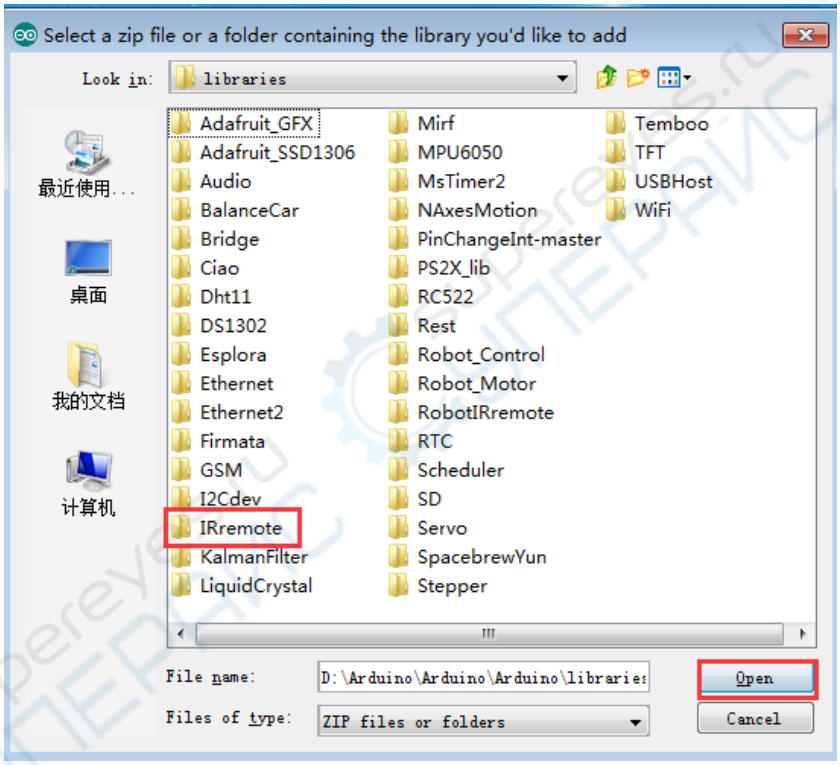
As shown in the figure below.



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.

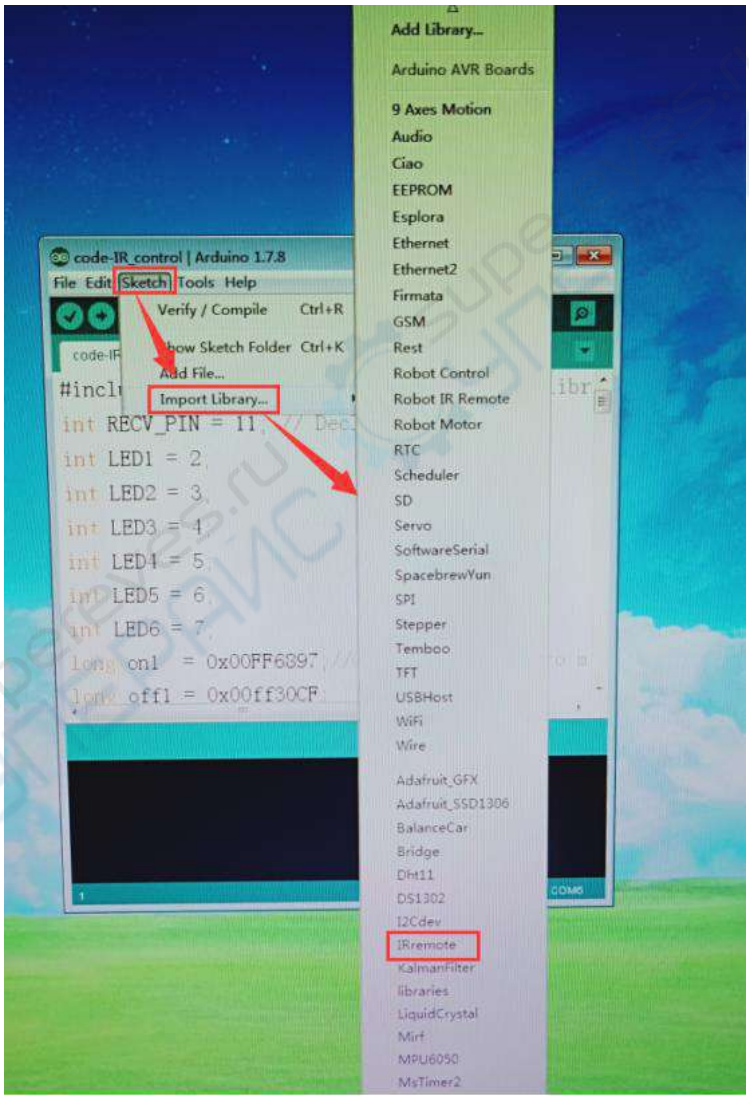
```
  pinMode(DHT11PIN,OUTPUT);
  Serial.begin(9600);
}

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 instru ction 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 LC D 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

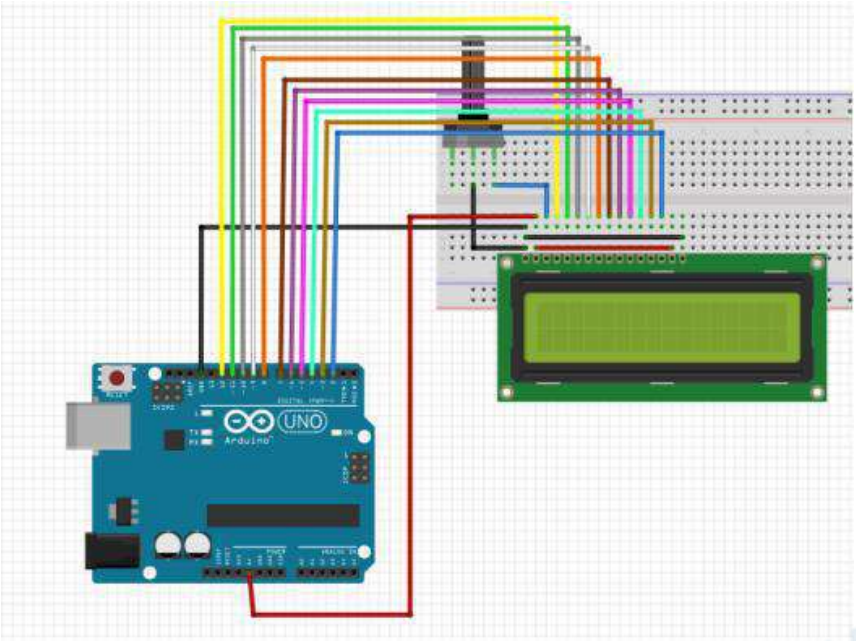
1602 *1

Dupont line *1 bunch

Breadboard *1

Actual object connection diagram：

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



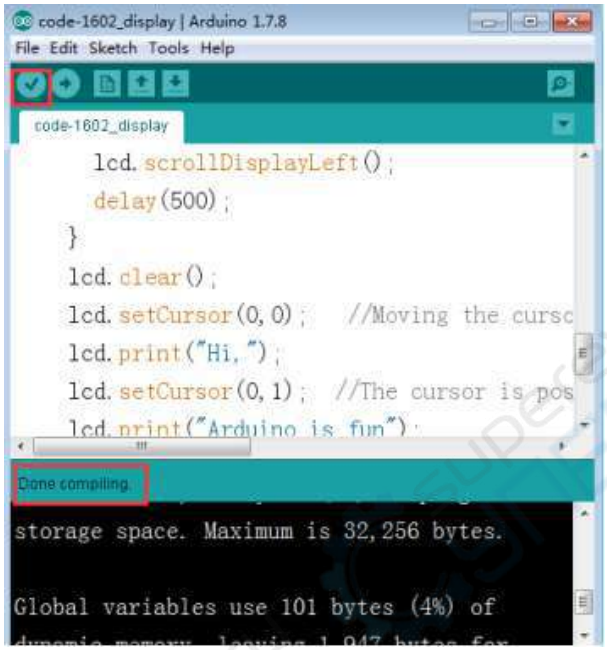
Experimental code analysis:

```
#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();
```

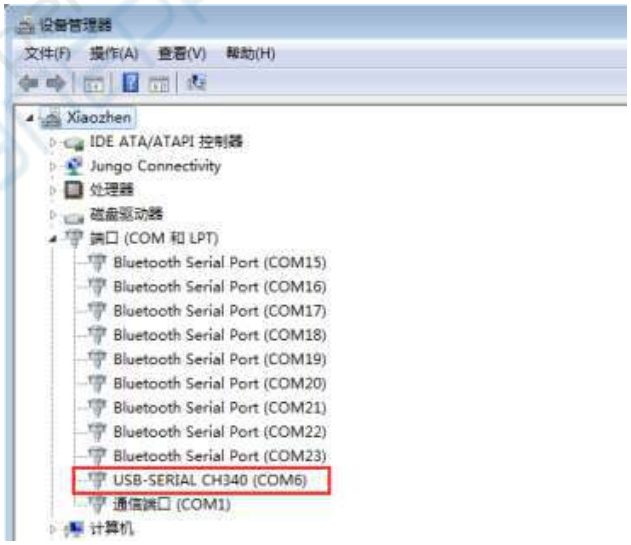
```
    lcd.setCursor(0,0);   //Moving the cursor back to the upper left corner,output from the beginning
    lcd.print("Hi,");
    lcd.setCursor(0,1);  //The cursor is positioned on line 1, column 0
    lcd.print("Arduino is fun");
    delay(2000);
  }
}
void loop()
{}//Initialization is complete and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

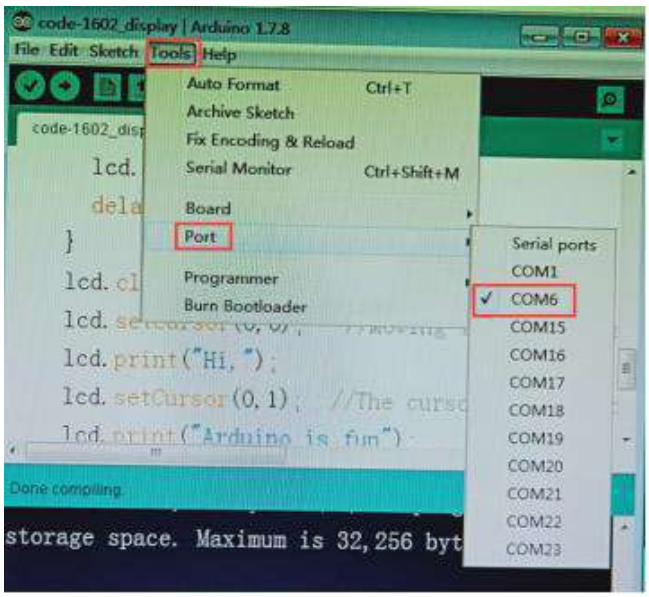
Experimental steps:

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, itclear the screen, and continue the endless cycle.

As shown in the figure below.