BBC: Micro:bit Smart Car -- MicroPython

Product Introduction

Are you curious about the working principle of motor control, ultrasonic ranging and RGB LED lighting? BBC Micro: bit smart car will lead you to explore their mysteries.

The smart car's control board is a micro: bit development board launched by the BBC, which is designed for youth programming education. Its car body is small, easy to carry and simple in structure. Building block holes are reserved for expansion. It is also with a variety of functions, such as ranging, light metering, black and white line detection, infrared remote control and Bluetooth control.

The kit supports graphical programming (MakeCode) and Python. The former helps beginners to understand program logic and develop programming thinking, while the latter, also called MicroPython running on the BBC Micro: bit, is an Python implementation optimized for microcontrollers and embedded systems.

Product Features:

Building block programming, easy to learn (online programming)

Small volume (only half of the adult palm), easy to carry, round shape, comfortable feeling

Convenient assembly, only a few parts need to be assembled

12GFN20 motor, with stable operation and strong power

Strong extensibility, 8 pins, two IIC interfaces and one servo dedicated interface

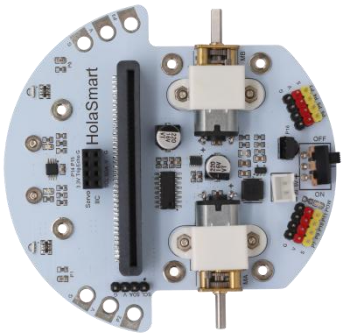
Reserved holes connecting building blocks

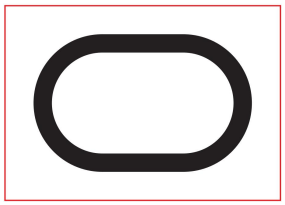
Product Parameters:

1. Connector port input: DC 4-5v
2. Sensor operating voltage: 3V
3. Motor speed: 200RPM
4. Operating temperature: 0-50℃
5. Assembled dimensions: 93x90x39mm
6. Environmental attributes: ROHS

Kit List

When receiving this product, please check on the following list to ensure that all accessories are intact. If any components are missing, please contact our sales staff immediately.

#	PIC	NAME	QTY
1		Car drive board	1
2		Holasmart ultrasonic sensor	1
3		Remote control	1
4		3-slot AAA battery holder	1

#	PIC	NAME	QTY
5		35mm wheel	2
6		3M double-sided adhesive tape 50 * 20 * 1mm	1
7		Map	1

What is Micro:bit?

Micro:bit is an open source hardware platform based on the ARM architecture launched by British Broadcasting Corporation (BBC) together with ARM, Barclays, element14, Microsoft and other institutions. The core device is a 32-bit Arm Cortex-M4 with FPU micro-processing.

Though it is just the size of a credit card, the Micro:bit main board is equipped with loads of components,including a 5*5 LED dot matrix, 2 programmable buttons, an accelerometer, a compass, a thermometer, a touch-sensitive logo and a MEMS microphone, a Bluetooth module of low energy, and a buzzer and others. Thus, it also boasts multiple functions.

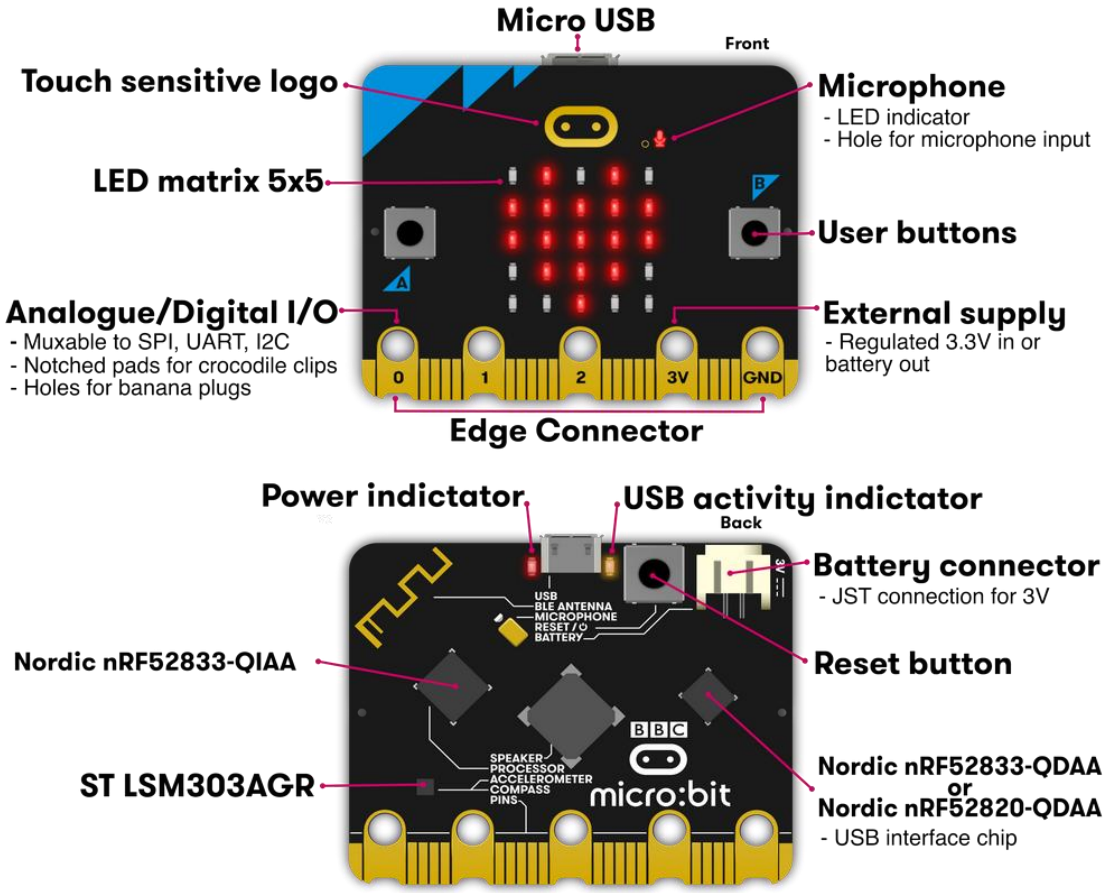
The buzzer built in the other side of the board plays all kinds of sounds without any external equipment. The golden fingers and gears provide a better fixing of crocodile clips. Moreover, this board has a sleeping mode to lower power consumption of batteries when users long-press the Reset & Power button on the back. It is capable of reading the data of sensors, controlling servos and RGB lights, and it is attaching with a shield so as to connect with various sensors. It also supports a variety of programming platforms and is compatible with almost all PCs and mobile devices. Also, driver is required. It is of high integration of electronic modules, and has a serial monitoring function for easy debugging.

The board has found wild applications. It can be applied in programming video games, making interactions between light and sound, controlling a robot, conducting scientific experiments, developing wearable devices.

Basically, it always be utilized to make some cool inventions like robots and musical instruments.

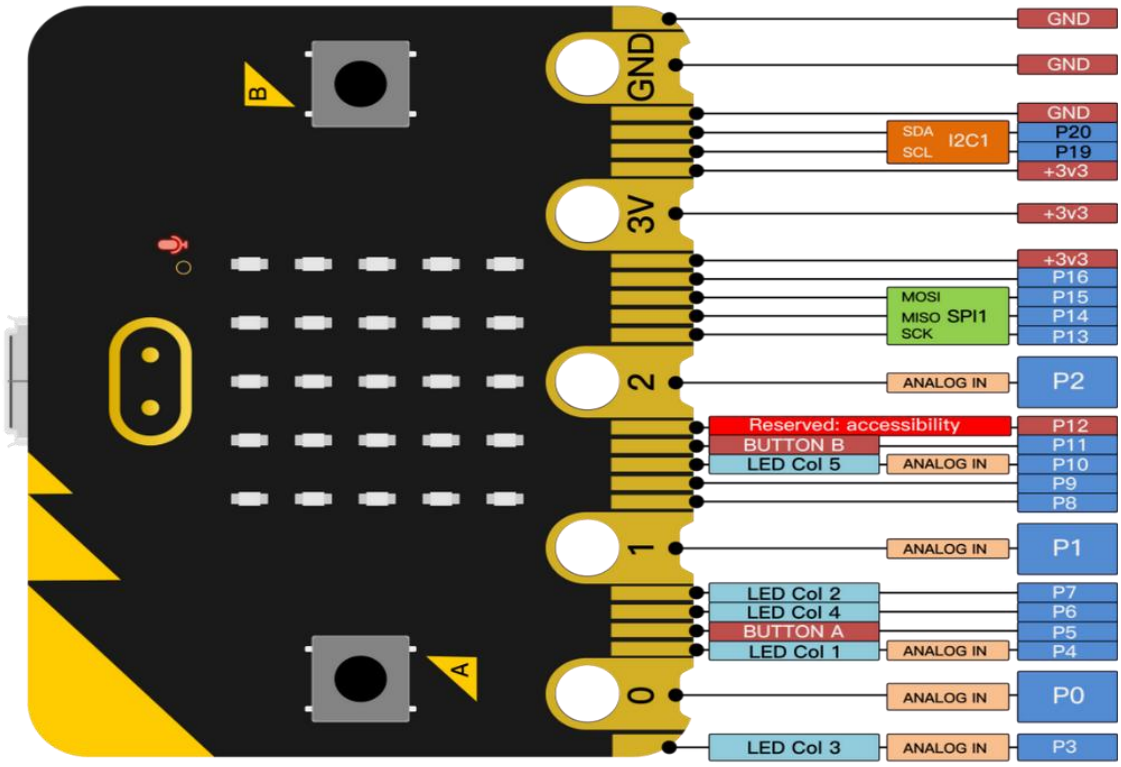
---

1. Micro:bit V2 Mainboard Layout



---

2. Micro:bit V2 Pin-out



Micro:bit pin functions:

Function	Pin
GPIO	P0, P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11, P12, P13, P14, P15, P16, P19, P20
ADC/DAC	P0, P1, P2, P3, P4, P10
IIC	P19(SCL), P20(SDA)
SPI	P13(SCK), P14(MISO), P15(MOSI)
PWM(commonly used)	P0, P1, P2, P3, P4, P10
PWM(uncommonly used)	P5, P6, P7, P8, P9, P11, P12, P13, P14, P15, P16, P19, P20
occupied	P3(LED Col3), P4(LED Col1), P5(Button A), P6(LED Col4), P7(LED Col2), P10(LED Col5), P11(Button B)

Visit the official website for more details:

Microbit hardware

https://microbit.org/guide/hardware/pins/

3. Notes for the Application of Micro:bit

- It is recommended to cover it with a silicone protector to prevent short circuit for it has a lot of sophisticated electronic components.
- Its IO port is very weak in driving since it can merely handle current less than 300mA. Therefore, do not connect it with devices operating in large current, such as servo MG995 and DC motor or it will get burnt. Furthermore, you must figure out the current requirements of the devices before you use them and it is generally recommended to use the board together with a Micro:bit shield.
- It is recommended to power the main board via the USB interface or via the battery of 3V. The IO port of this board is 3V, so it does not support sensors of 5V. If you need to connect sensors of 5 V, a Micro: Bit expansion board is required.
- When using pins(P3, P4, P6, P7 and P10) shared with the LED dot matrix, blocking them from the matrix or the LEDs may display randomly and the data about sensors connected maybe wrong.
- Pin 19 and 20 can not be used as IO ports though the Makecode shows they can. They can only be used as I2C communication.
- The battery port of 3V cannot be connected with battery more than 3.3V or the main board will be damaged.

Forbid to operate it on metal products to avoid short circuit.

To put it simple, Micro:bit V2 main board is like a microcomputer which has made programming at our fingertips and enhanced digital innovation. And

as for programming environment, BBC provides a website: https://microbit.org/code/, which has a graphical MakeCode program easy for use.

Drive Board

1.Introduction

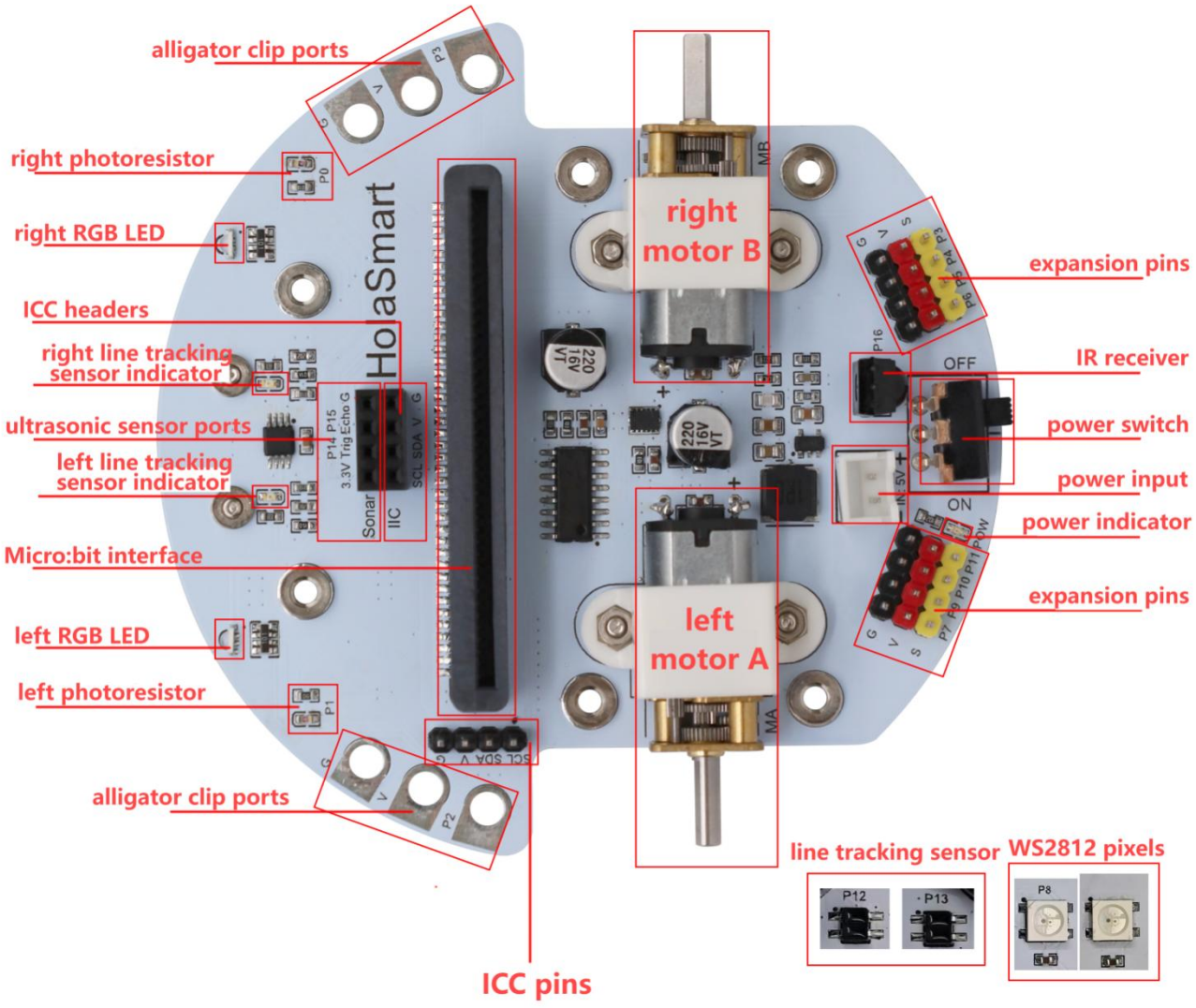
The car drive board integrates the infrared receiver, photoresistors, line tracking sensors, RGB LED, ws2812 pixels, ultrasonic sensor, IIC interface, motors and so on. So there is no worry about wiring. There are also 8 IO ports on the board for expansions.

Due to the shortage of micro: bit IO ports, the board sends commands to STC8G1K08 through IIC, and then ST8G1K08 controls HR8833 motor to drive IC via PWM, so as to control the motor rotation and LED RGB.

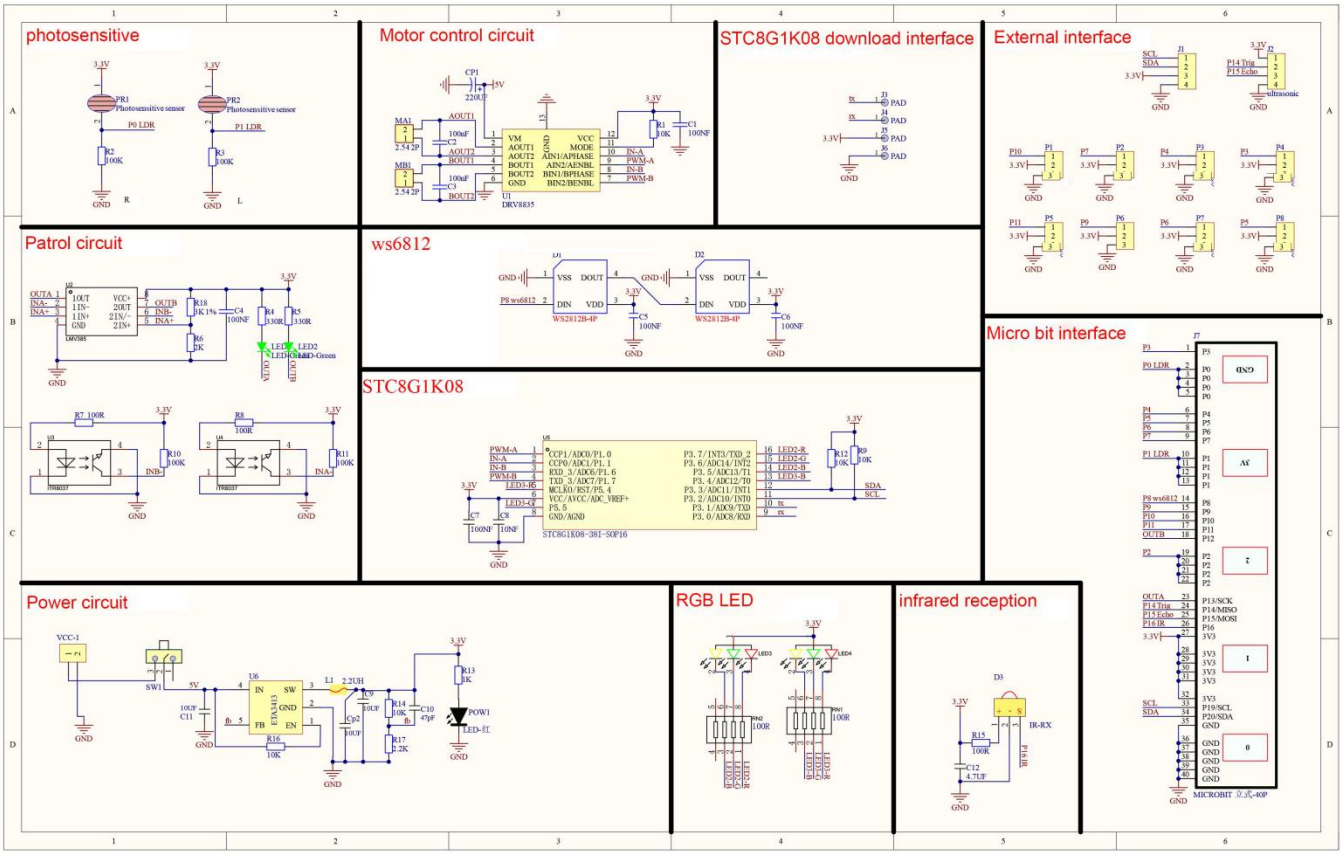
2.Parameters

- Connector port input: DC 5v
- Motor speed: 200RPM
- Operating temperature: 0-50℃
- Dimensions: 90*88*15mm
- Environmental attributes: ROHS

3.Function Diagram

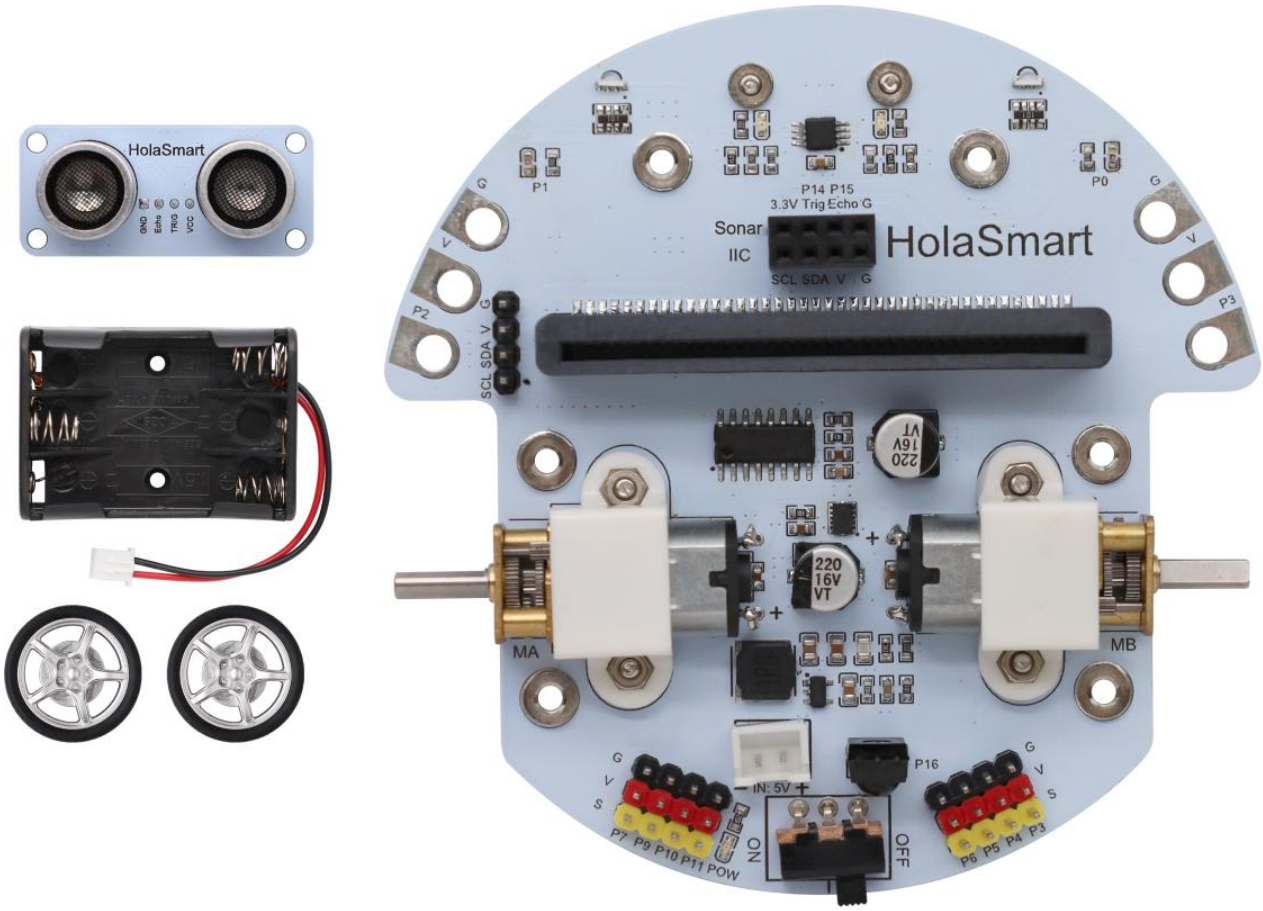


4.Schematic Diagram

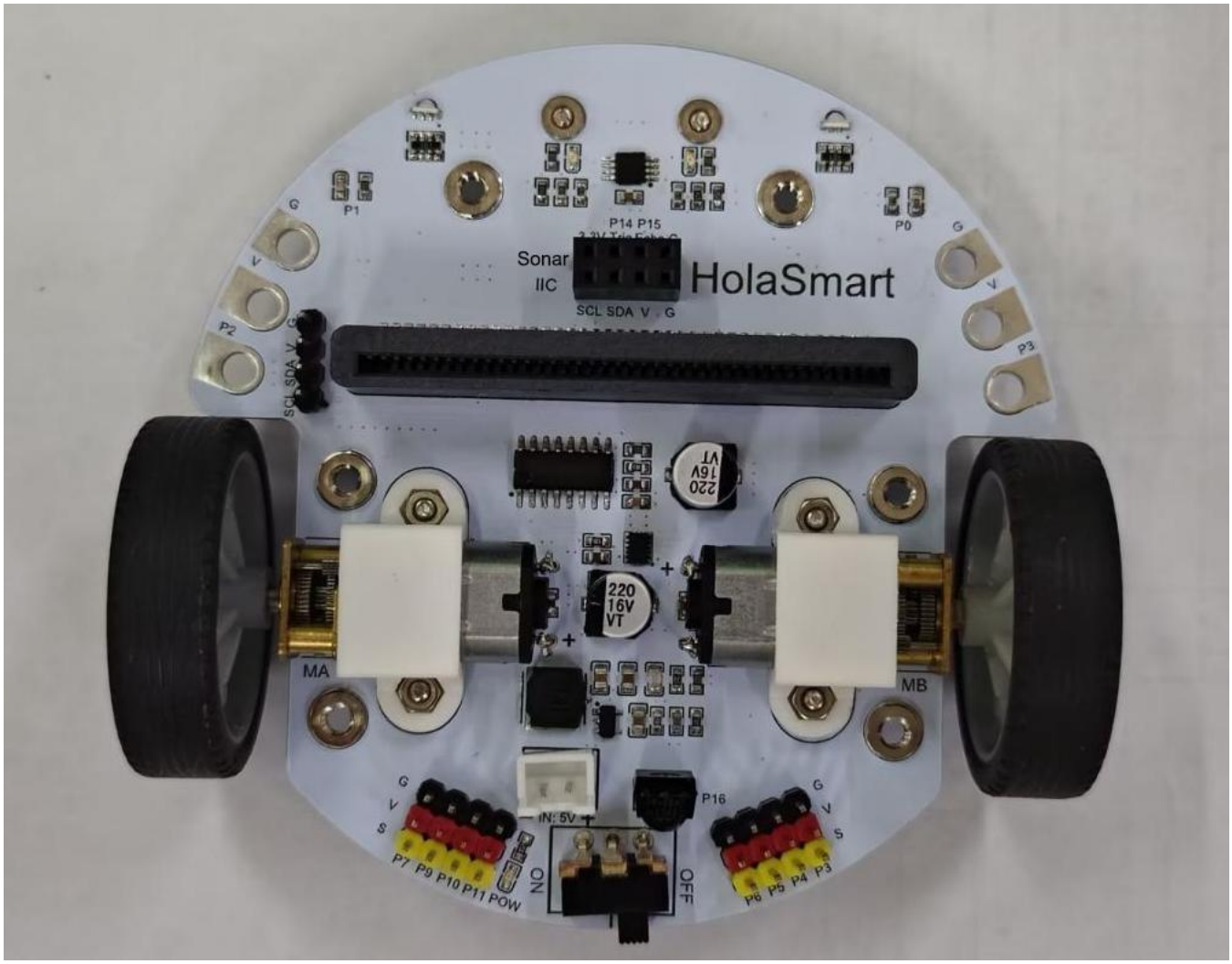


Assembly

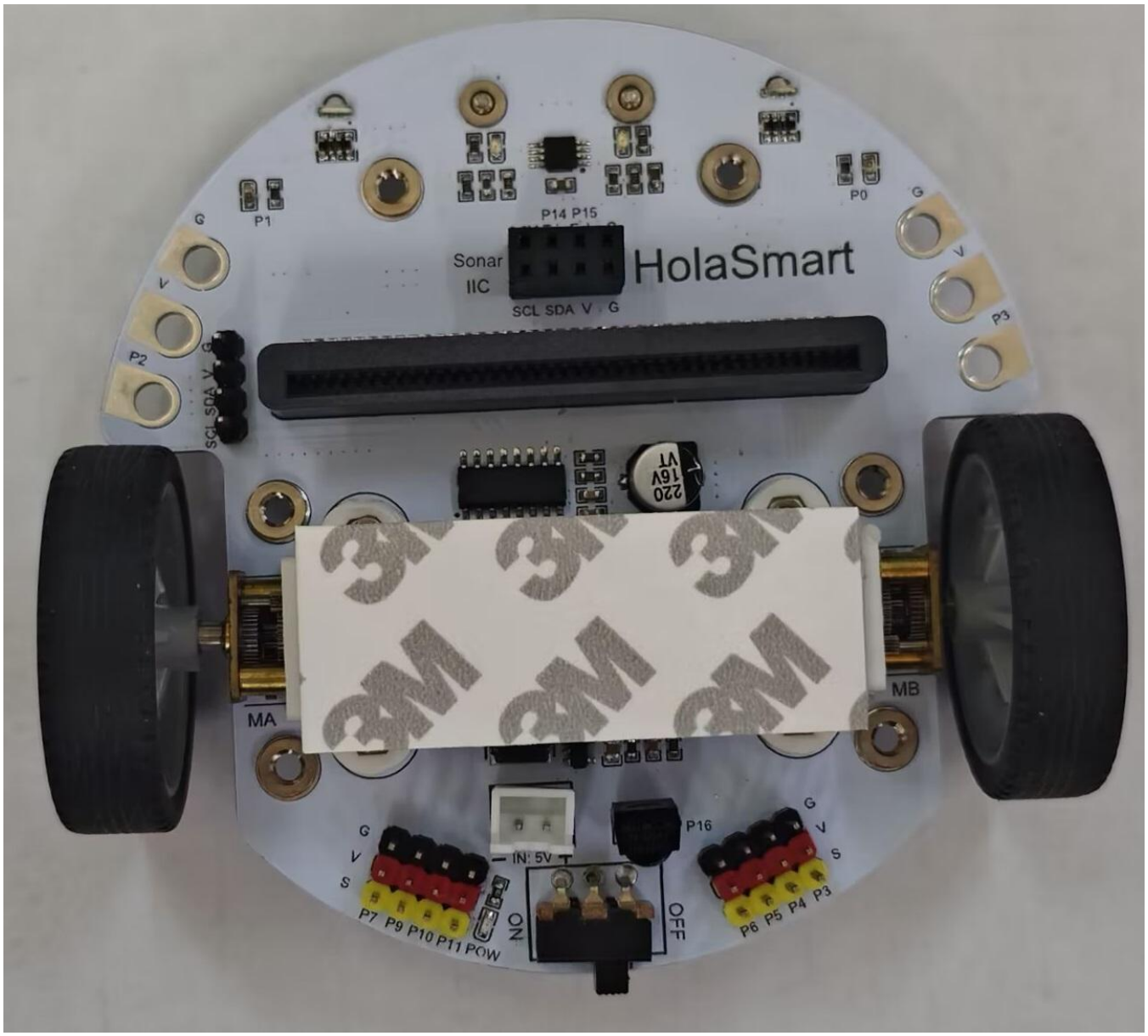
1. Remove the ultrasonic sensor, battery holder, wheels, double-sided tape, and car drive board from the package.



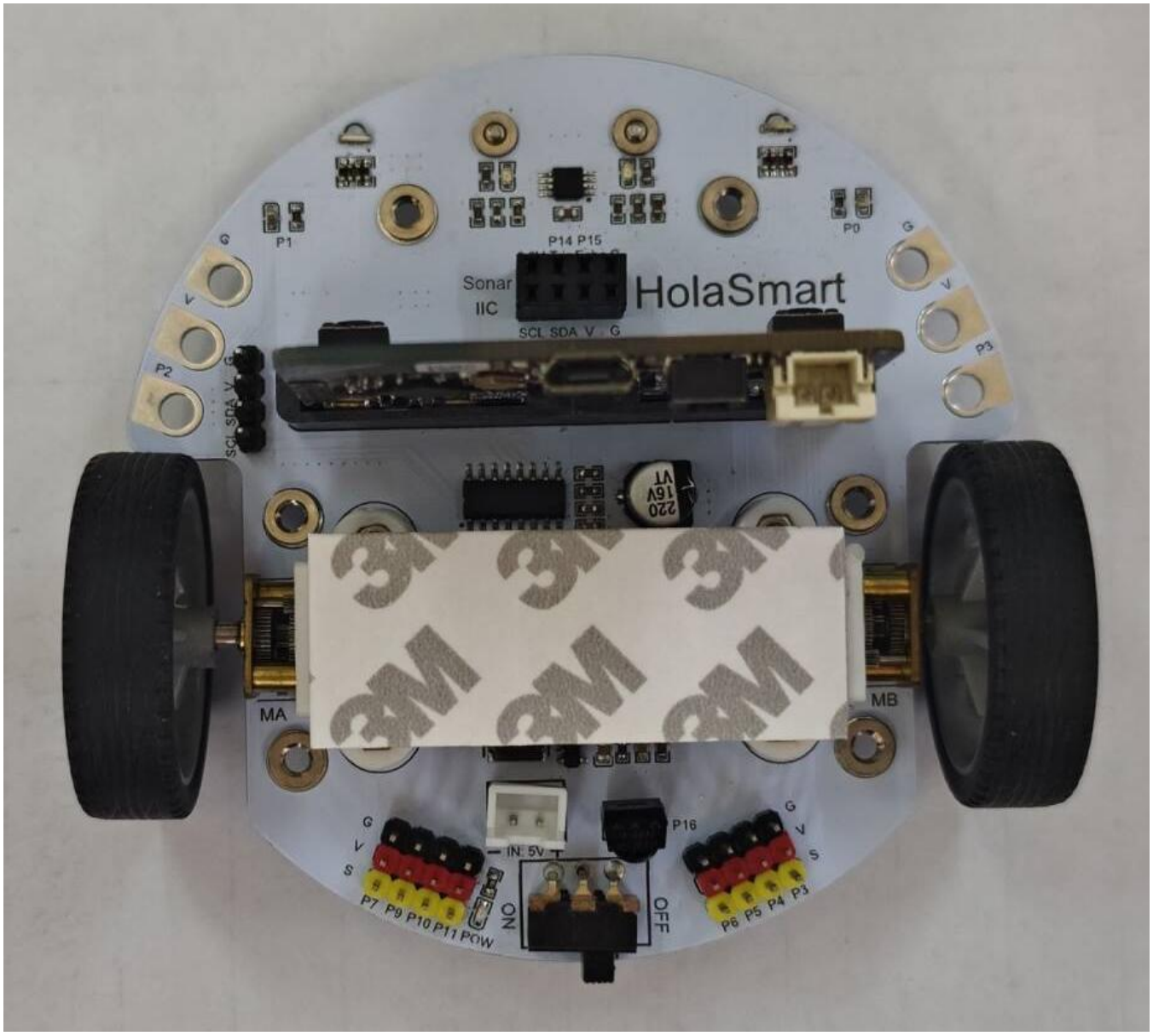
2. Install the two wheels onto the motor shaft.



3. Install tape, tear off the cover and attach it to the motor support.

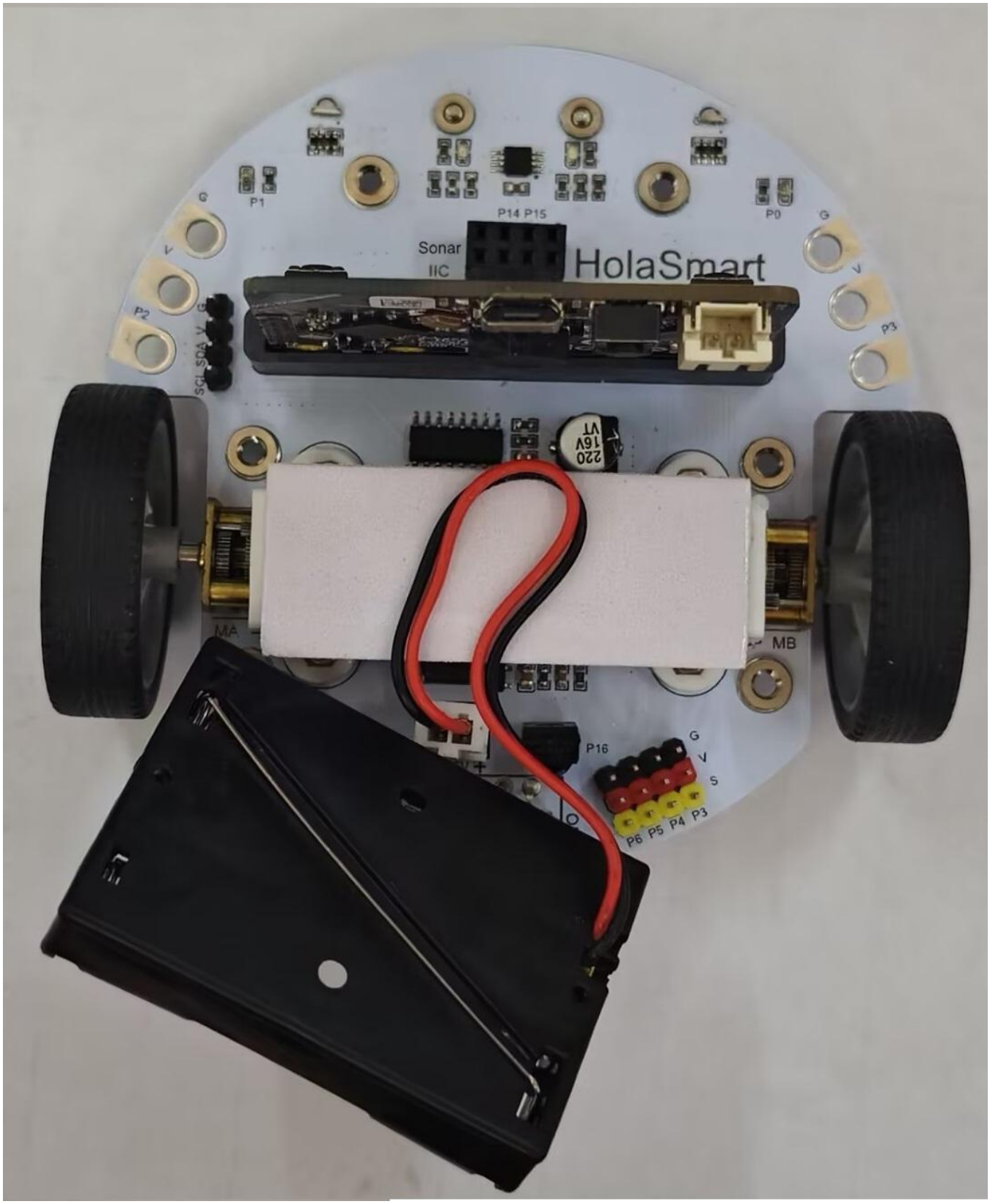


4. Install Micro bit board.



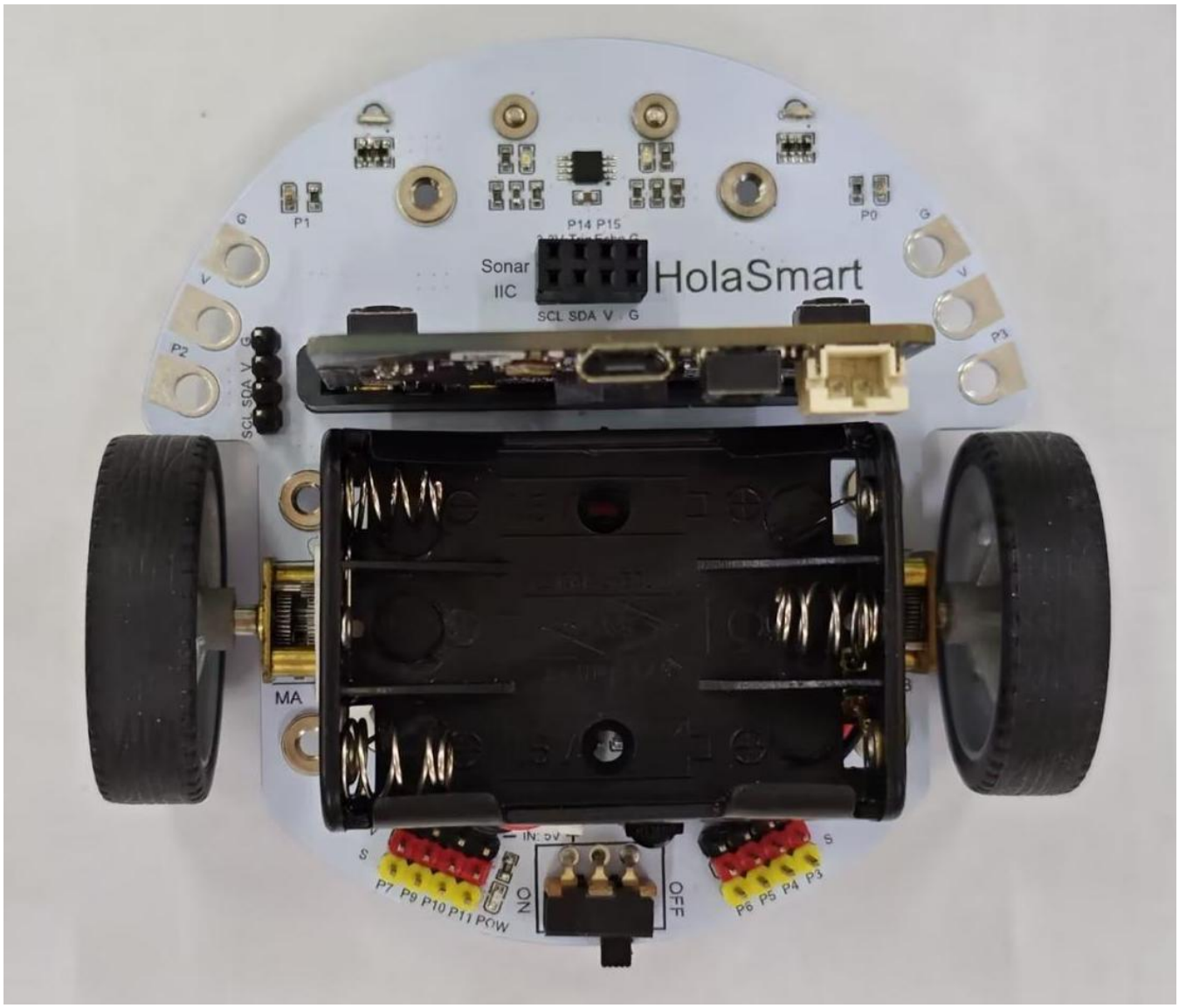
5. Mount battery holder.

Peel off the tape cover, and then plug the interface of the battery holder into the drive board, and paste the wire on the tape as follows:

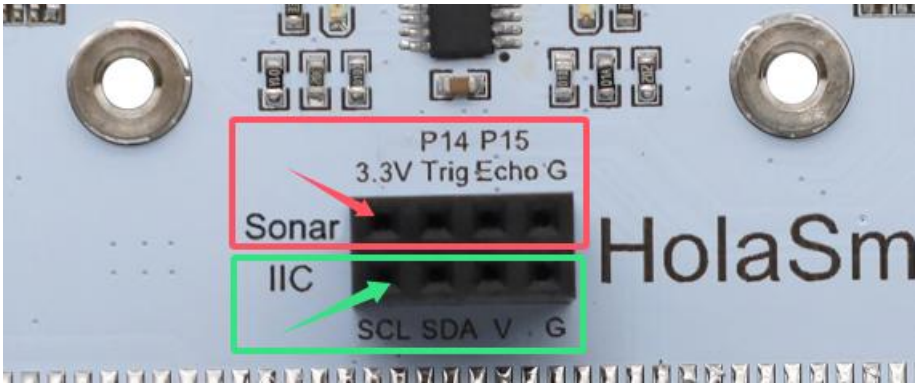


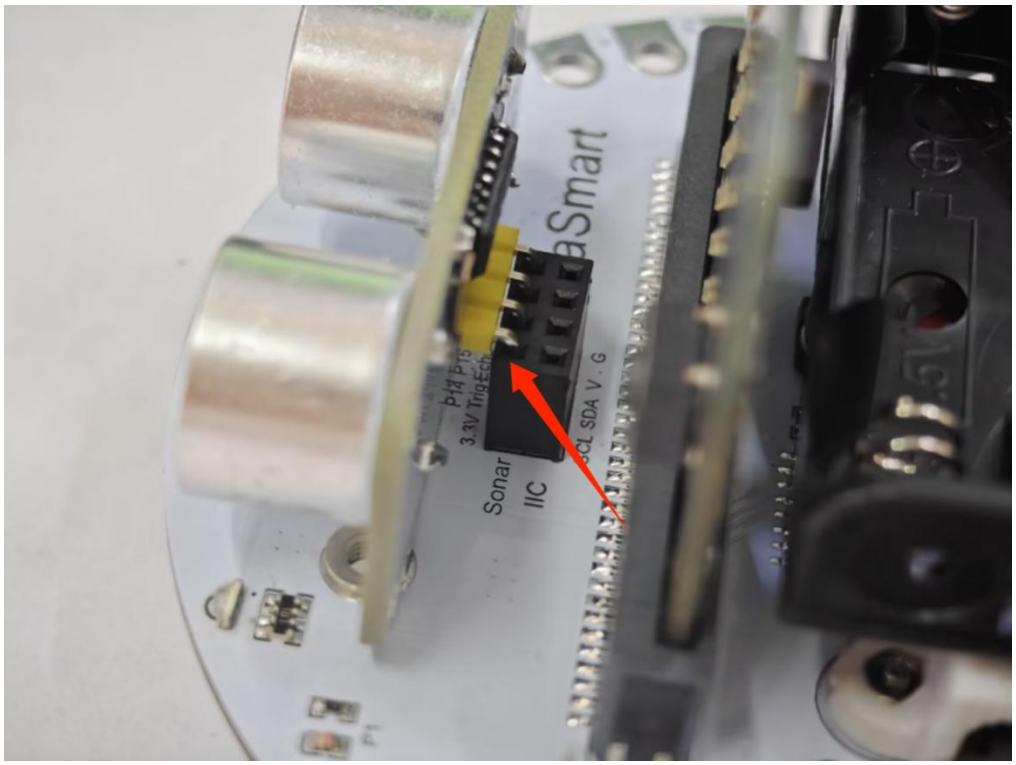
Put the battery box at the middle position and press it tightly.

(Note: please install the Micro: bit board before battery holder, because there should be a distance between the Micro: bit board and the battery holder as shown in the picture.)



6. Install the ultrasonic sensor. Please pay attention the position of it. Ports in the red box are for the ultrasonic sensor while that in the green are for ICC.





7. Now the small car is fully assembled. If you fell not convenience when mounting batteries, you may remove the Microbit for a while.

Development Software

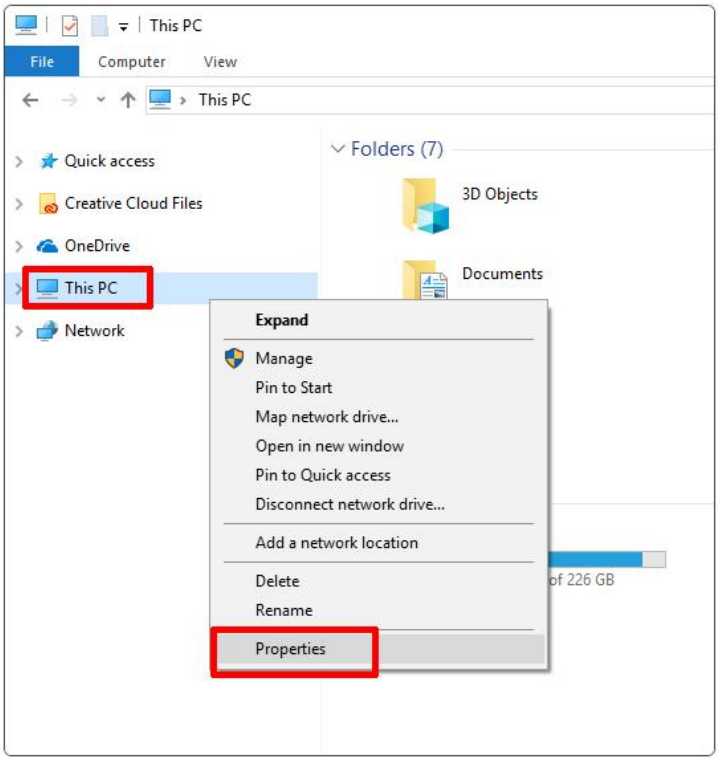
1.Install MU

Mu official: https://codewith.mu/

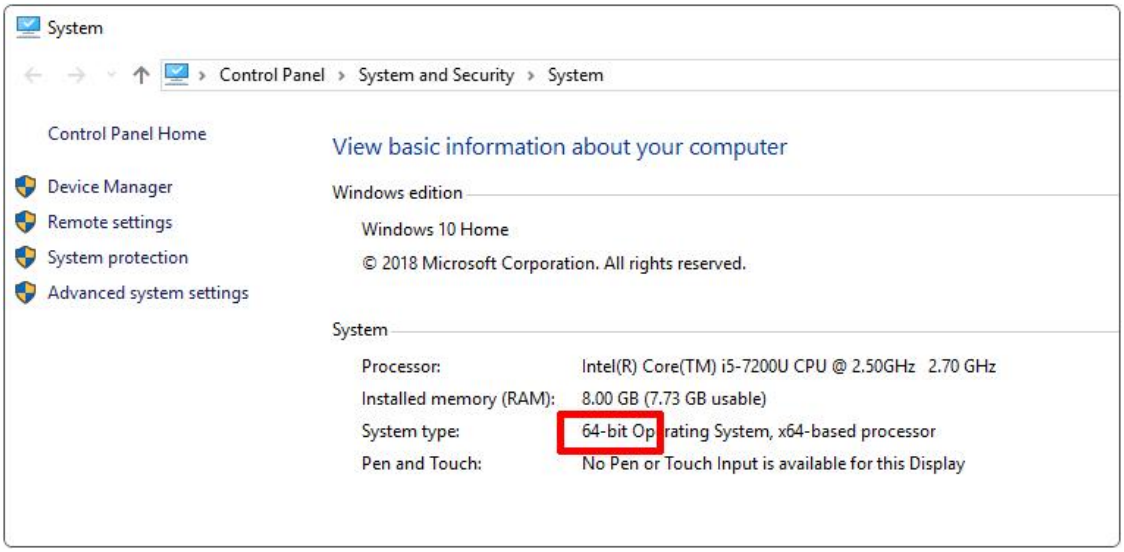
Mu is a Python code editor for novice programmers and faculty and students. We install Mu from official installer of Windows or Mac OSX (**Mu no longer supports 32-bit Windows**). The currently recommended version is Mu 1.2.0. So please update to this version on your computer.

Step 1-Determine the version and download the Mu installer

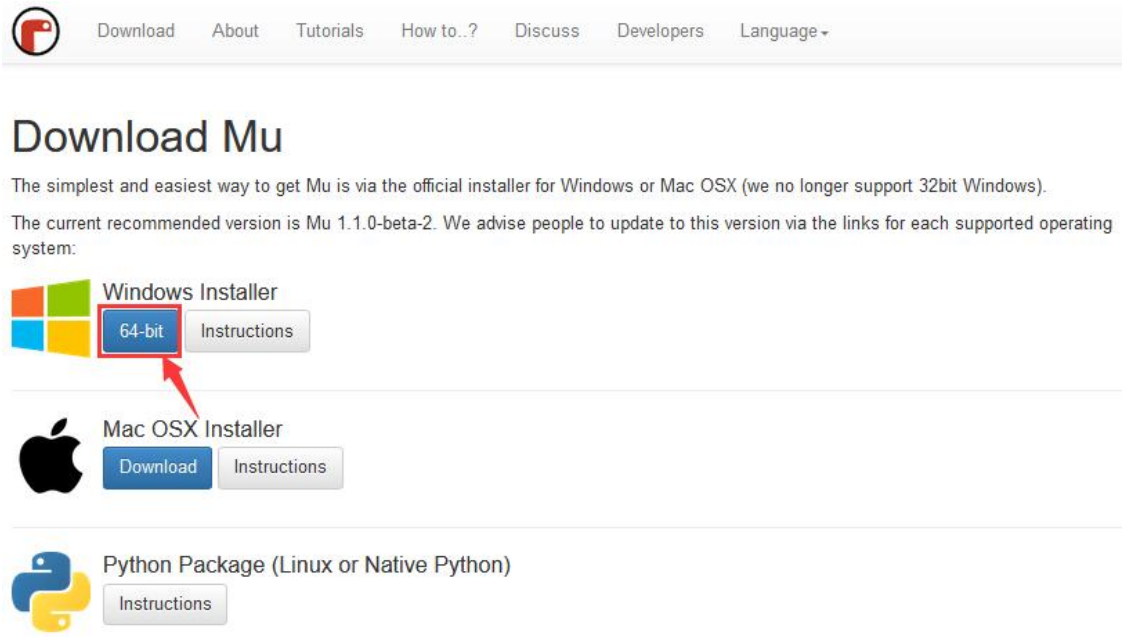
Check your operation system (Windows or Mac OSX); then open the explorer and click "this computer" to select attributes to know whether your system is 32 bit or 64 bit.



View system category, type is displayed in system: 64-bit or 32-bit:

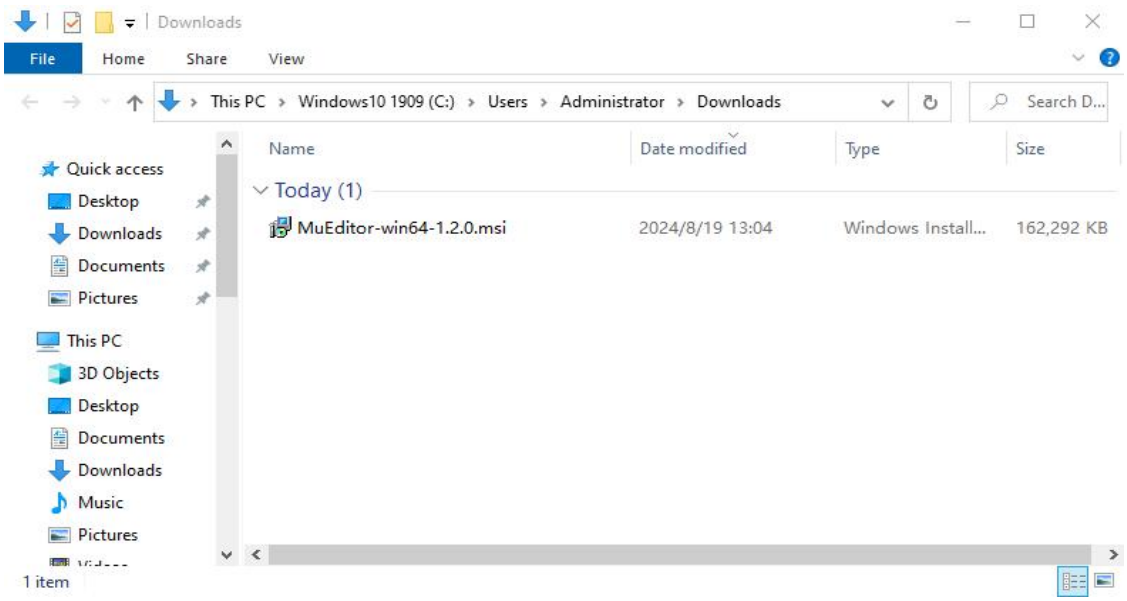


Click https://codewith.mu/en/download to download Mu.



Step 2-Run the installation program:

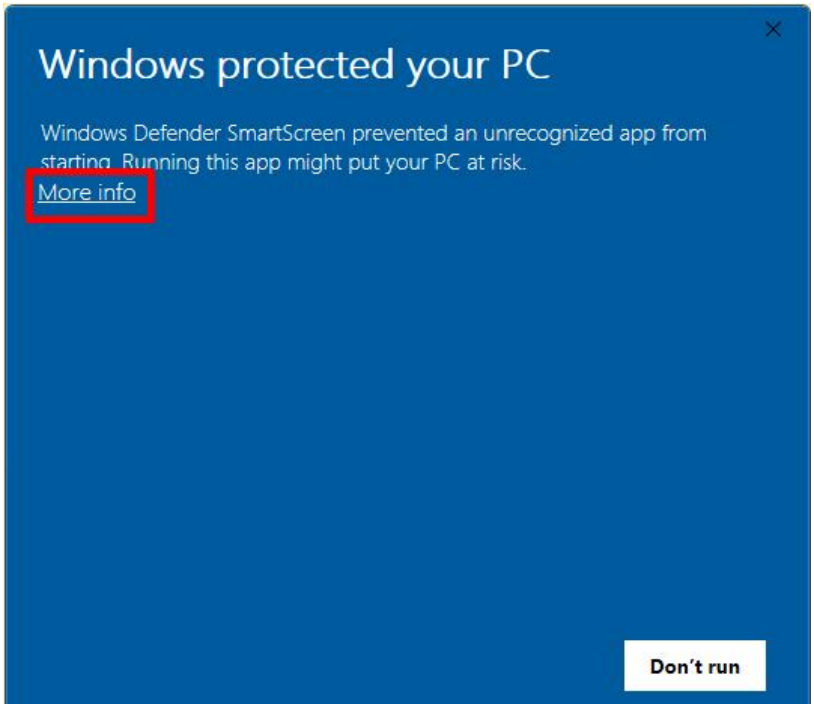
Find the installer you just downloaded (it may be in your download folder) and click to open it.



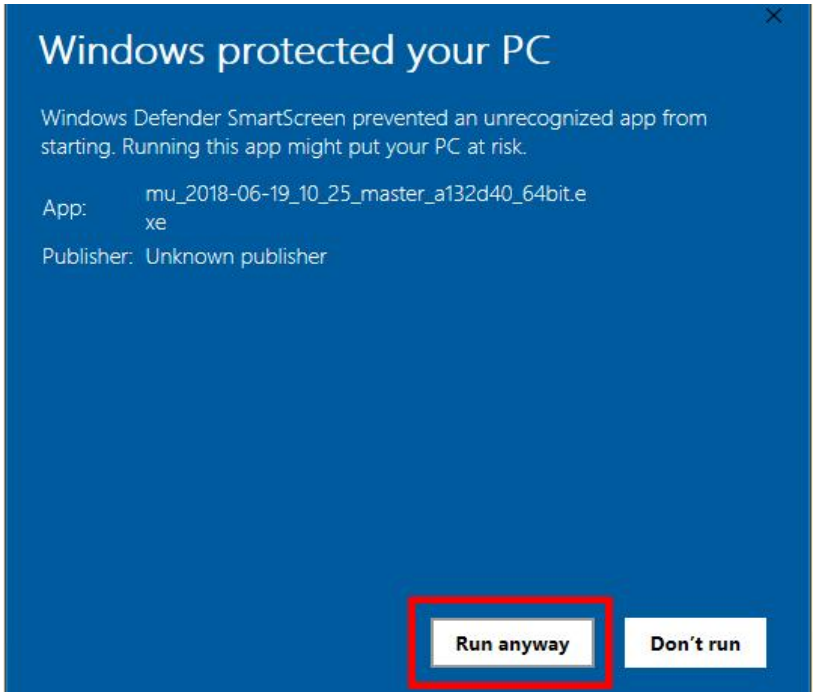
Here we outline the additional steps required to install Mu on Windows 10 (other versions of Windows are similar).

Install Mu on Mac OSX: https://codewith.mu/en/howto/1.1/install_macos

Warning occurred during installation on Windows 10: A warning message will be popped up on Windows Defender. Click "more info".



The message will change to provide more information about the installer, and display the Run anyway button. Click the button.

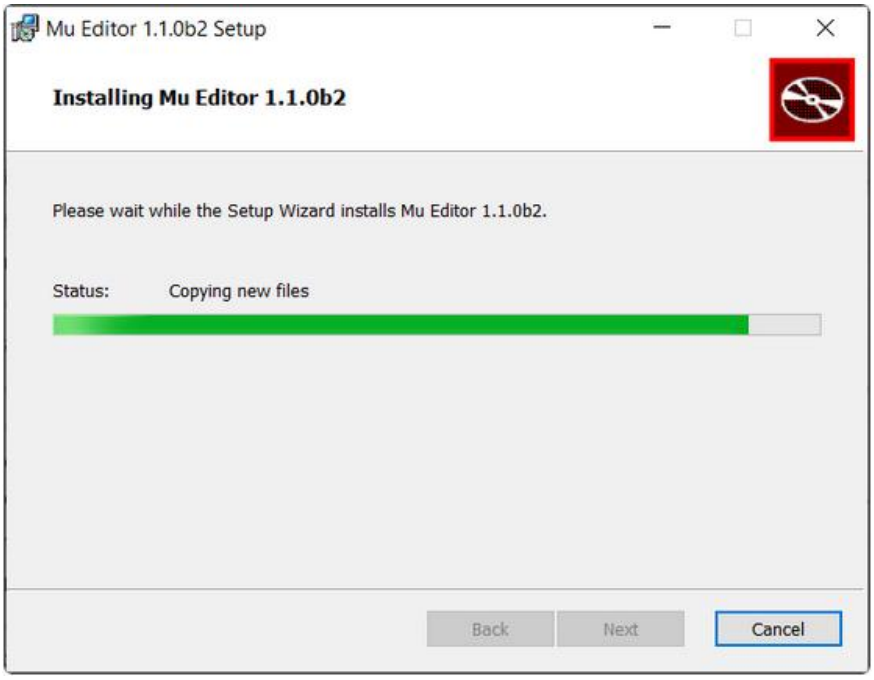


Step 3-License Agreement

Check the license, tick the accept box, and click Install.

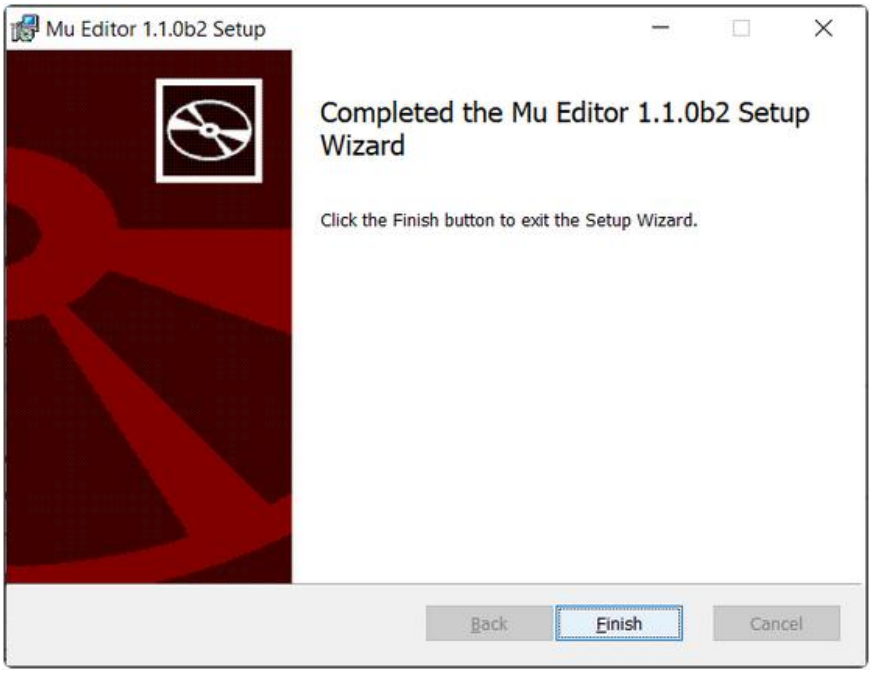
Step 4-Installation

When Mu is installed on your computer, it takes a few seconds.



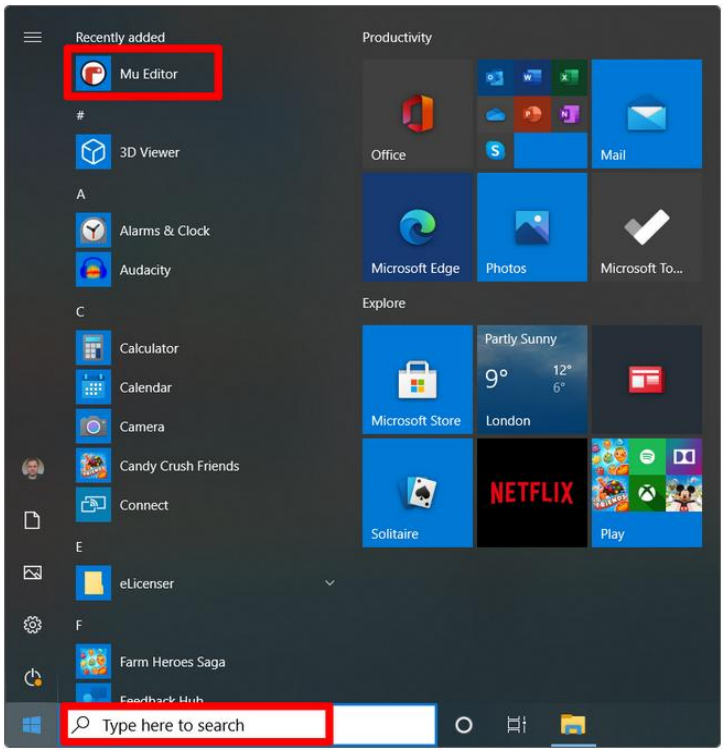
Step 5-Complete

The installation completed successfully, click Finish to close Setup.

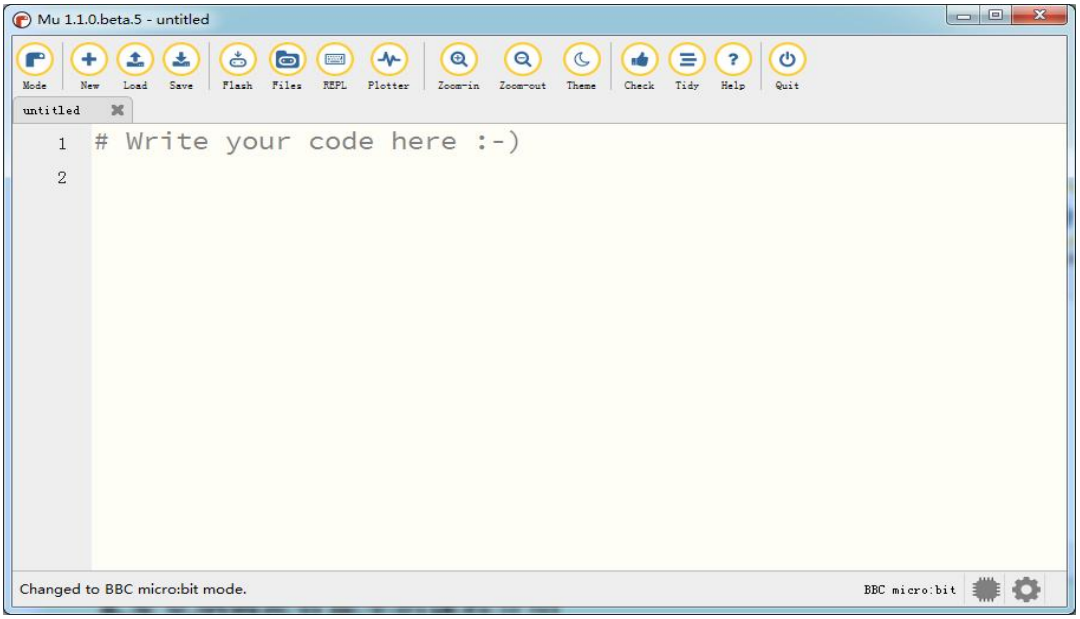


Step 6-Start the MU

Directly click Mu to enter it, or search it to open. At the first start, this may take a few seconds.



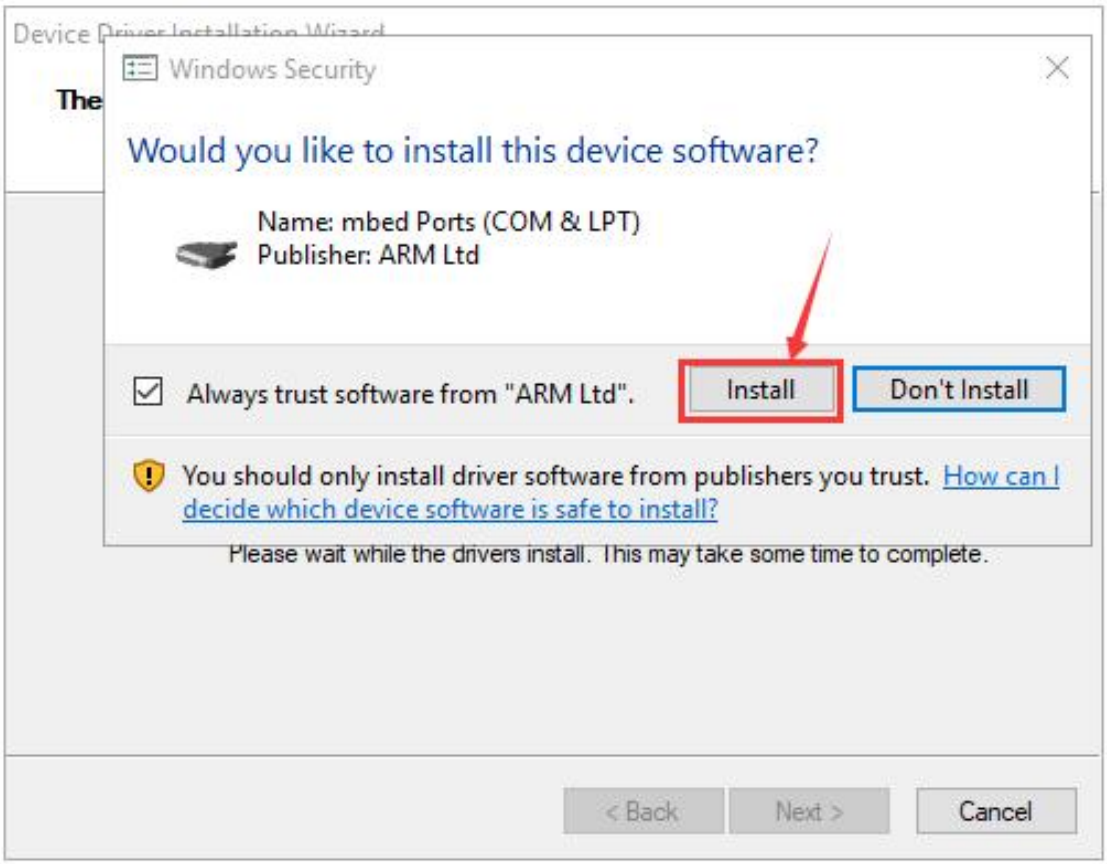
Mu Main interface:



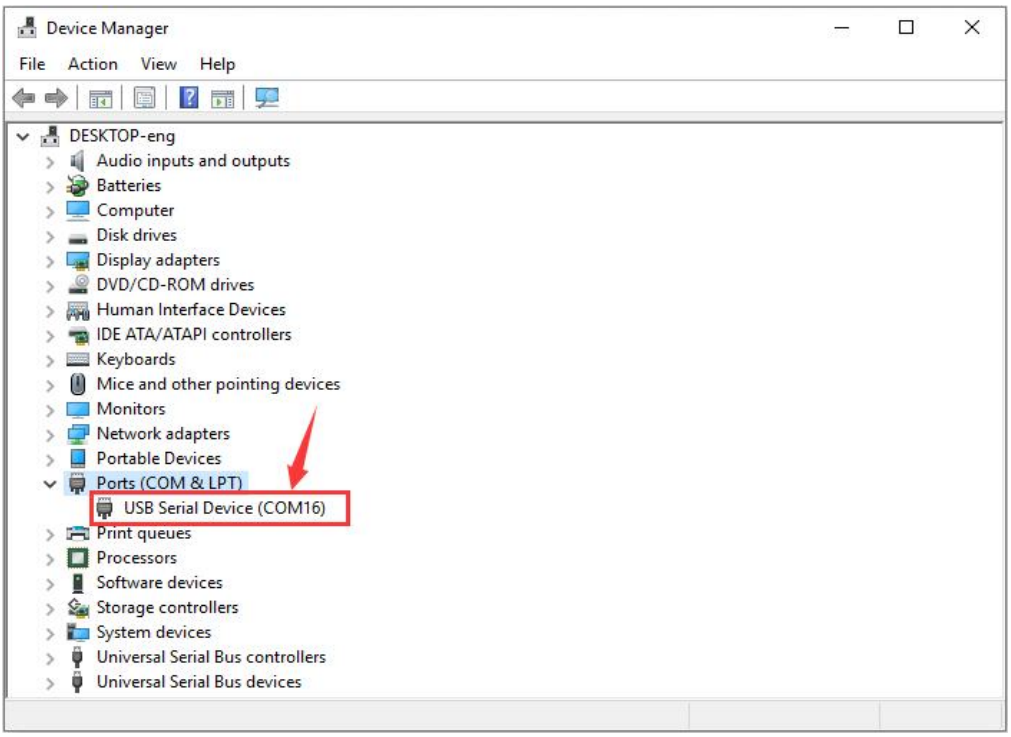
2.Development Board Driver

Micro: bit is USB driver-free. If your computer can not recognize the micro: bit board, you need to install the driver we provided: mbed_usb_2020_x64_1212.exe. Click here to download USB driver.

Run and click Install.

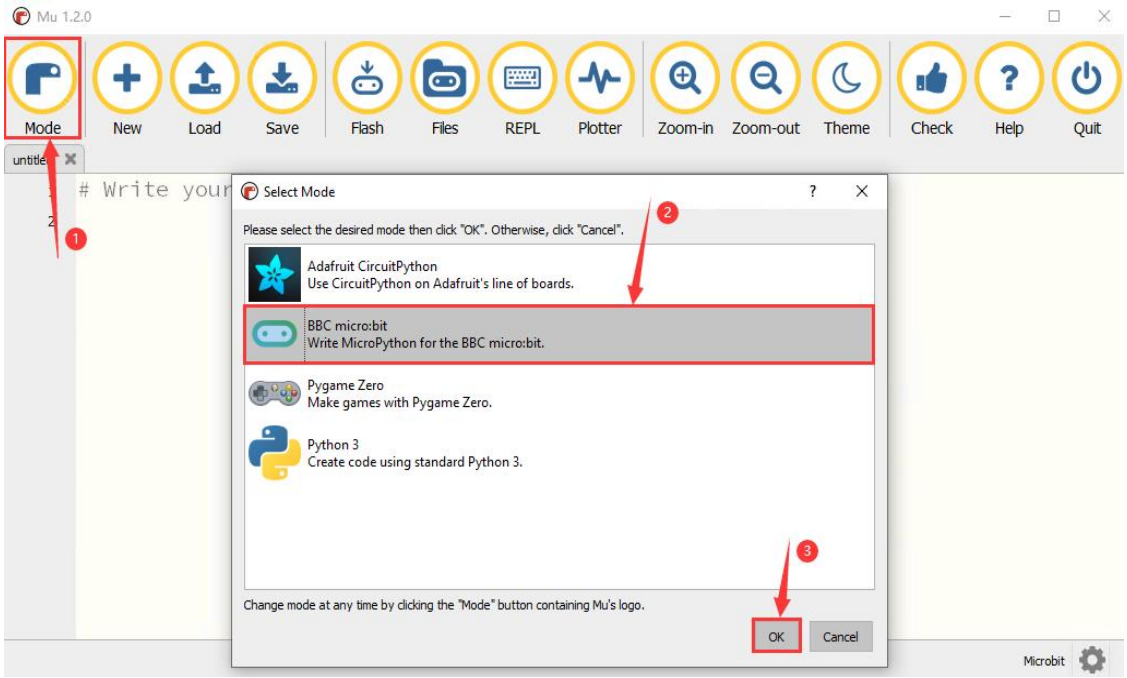


Click "Computer" —>"Properties"—> "Device manager".



3.Settings and Toolbar

Set "mode" to micro bit when first used. Open Mu, click the Mode to select "BBC micro: bit", and then click "OK".

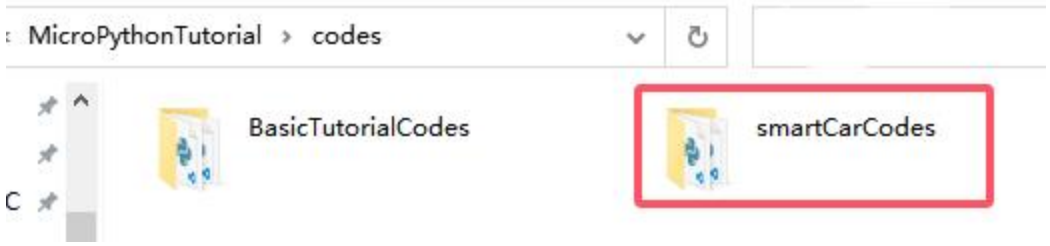


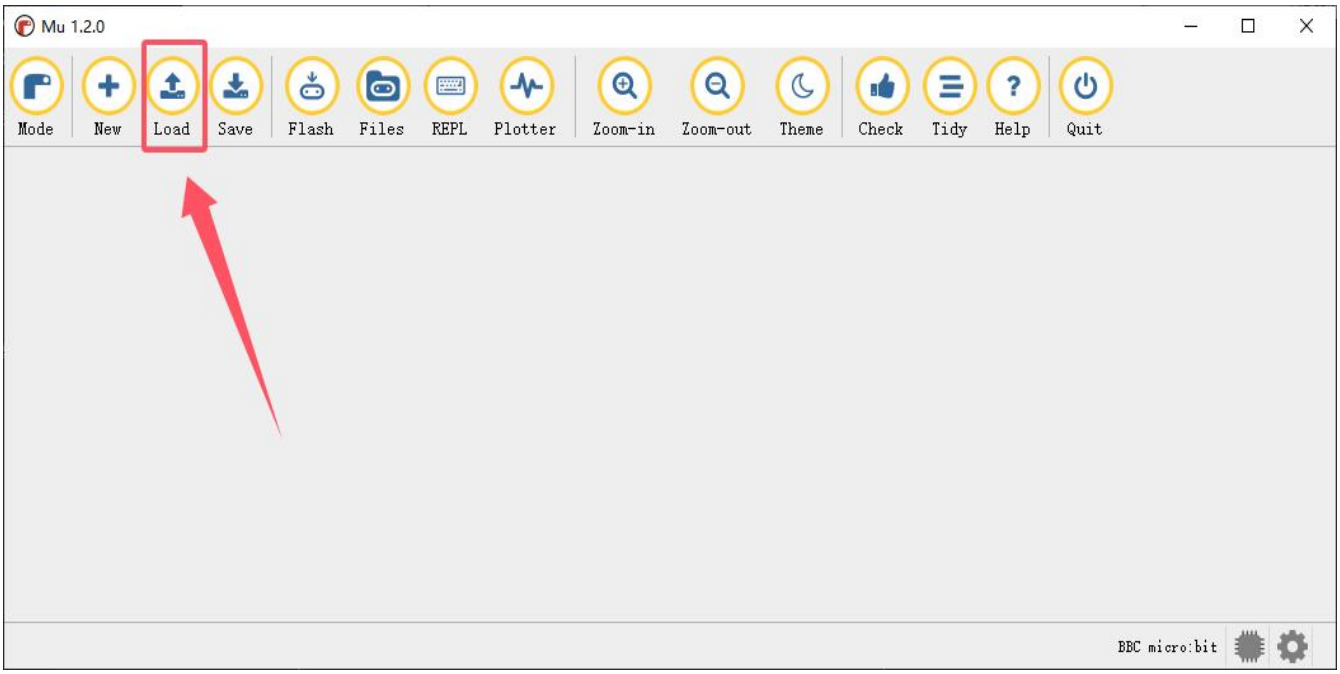
Please refer to the official introduction for the Mu operation page: https://codewith.mu/en/tutorials/1.1/start

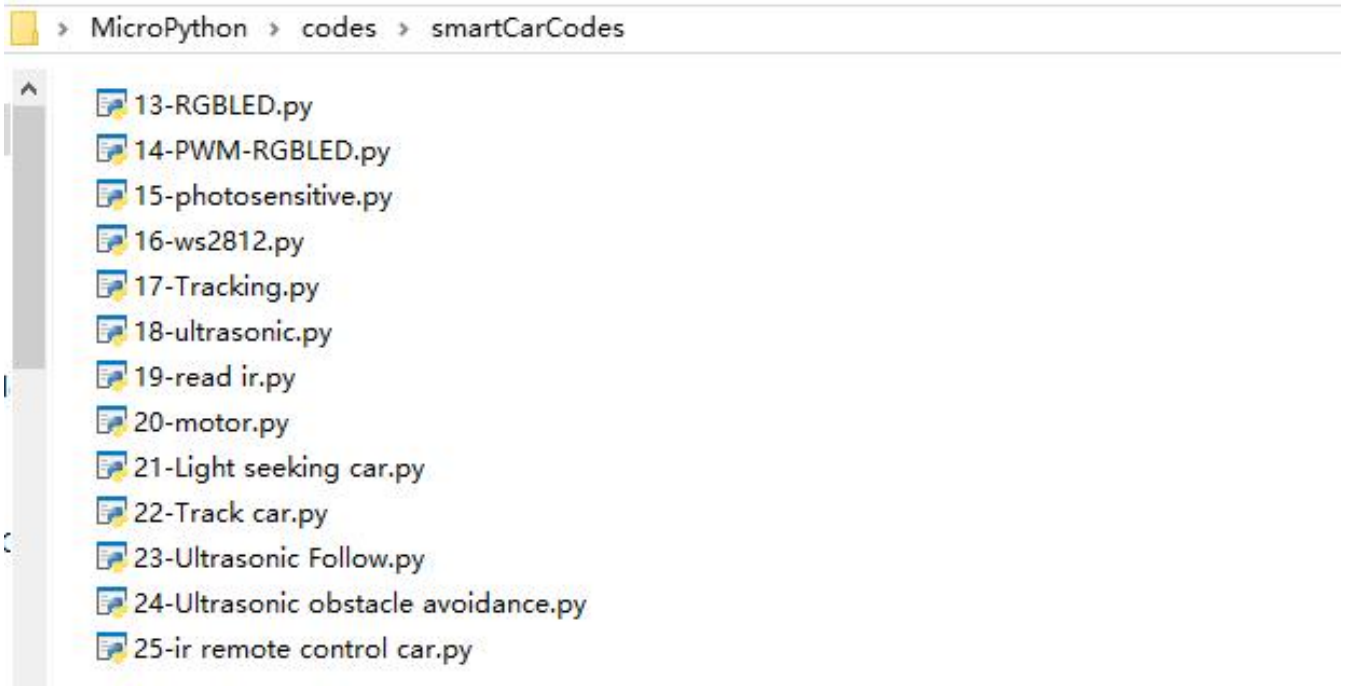
For more tutorials about using the Mu, please visit: https://codewith.mu/en/tutorials/

4.How to Load Code into MU

Here we take smart car projects "Project 1 RGB LED" as an example. Find the "codes" file and open the "1-RGBLED.py" file in "smartCarCodes" folder.



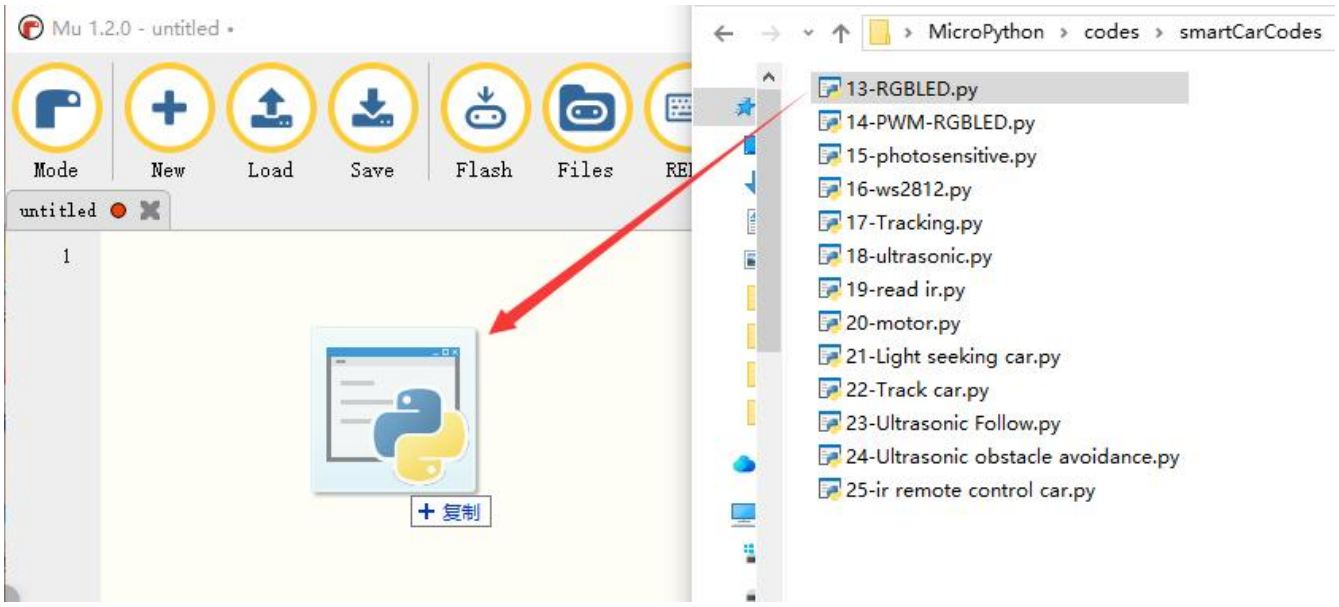




In addition to the above method of loading (import), there is a simpler way:

Click 

and choose "1-RGBLED.py", drag it into Mu:



Successful loading:

5.Import Library to Mciro:bit on MU

You need to upload a .py code to the microbit before importing library. Here we take the "13-RGBLED.py" in the car codes as an example.

Import **smartCar.py** from "lib" file.

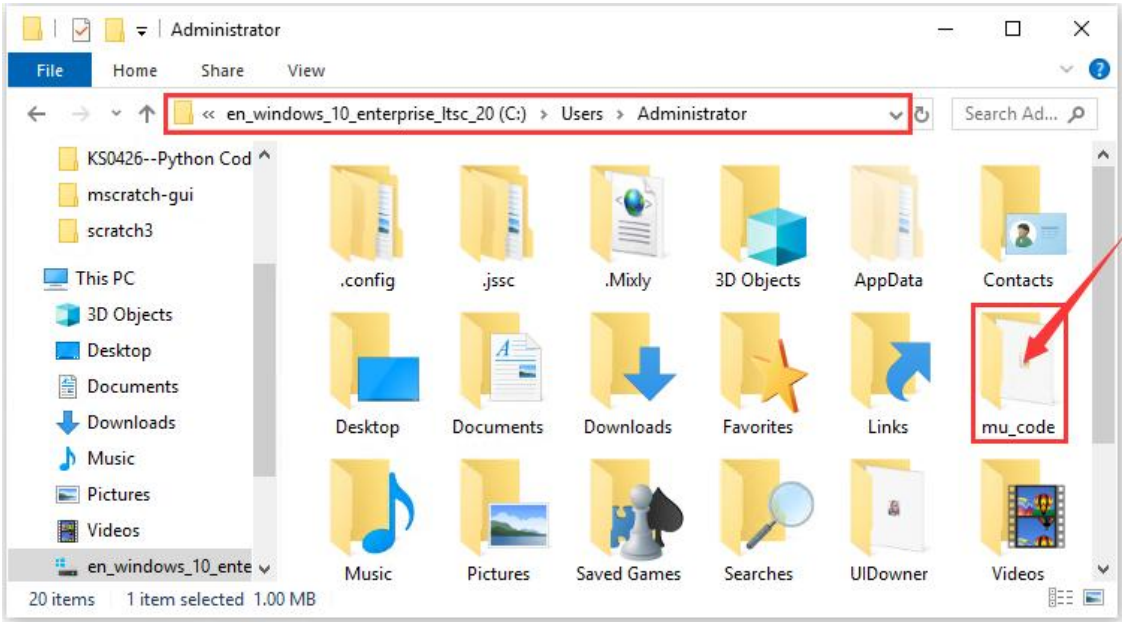
The default directory for the Mu file is "Mu_code" located in the root directory of the user directory.

References: https://codewith.mu/en/tutorials/1.0/files

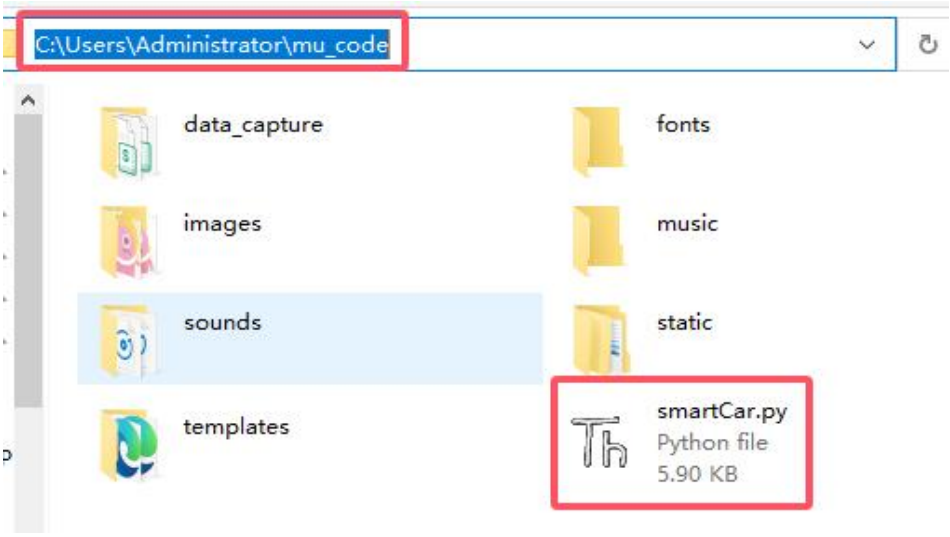
For example, in Windows, assuming that it is installed on the Disk C, the user name is "Administrator", and the path of the "mu_code" directory is

"C:\Users\Administrator\mu_ code". On the Linux system, the path "~/home/mu_code".

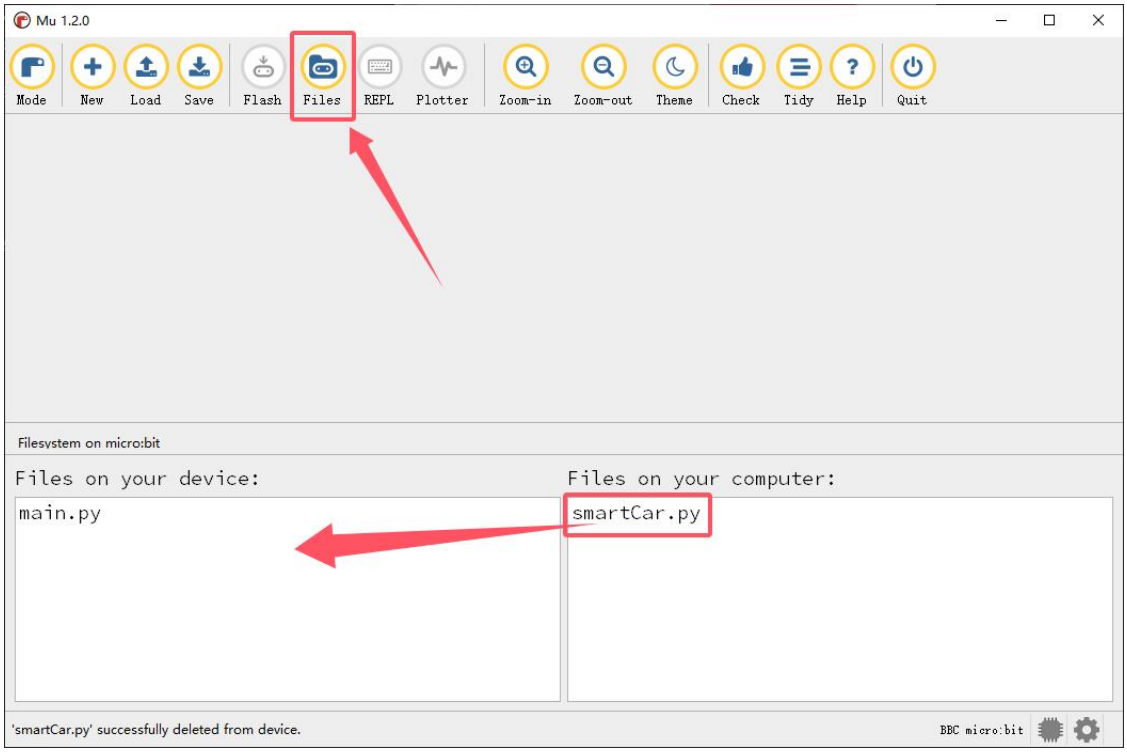
Enter "mu_code" folder.



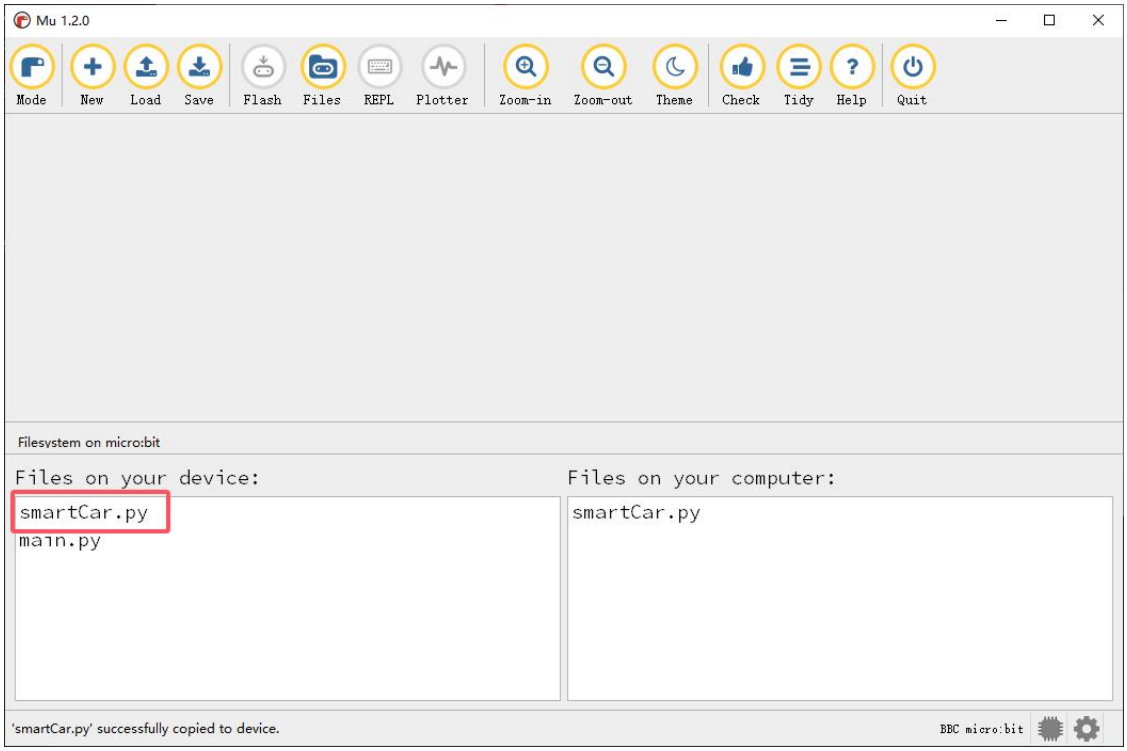
Copy the smartCar.py library to the mu_code folder. The code path is as follows. Successfully copied:



Open Mu and connect micro: bit to computer, then click "Files" to select smartCar.py file and drag it to micro: bit.

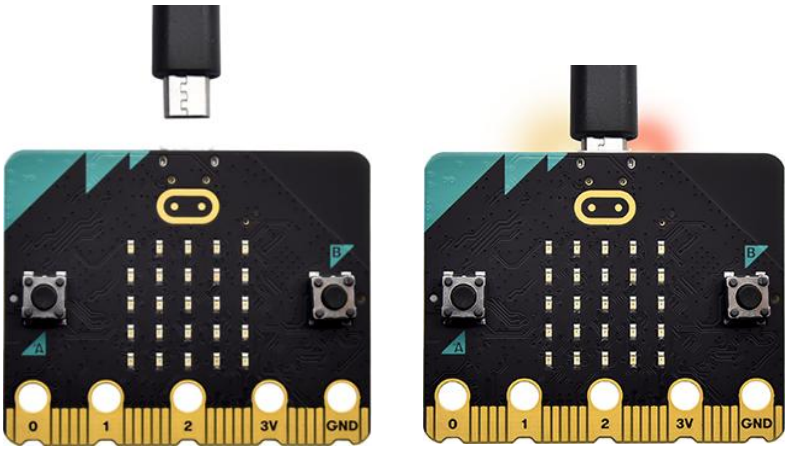


Successful uploading of the library:

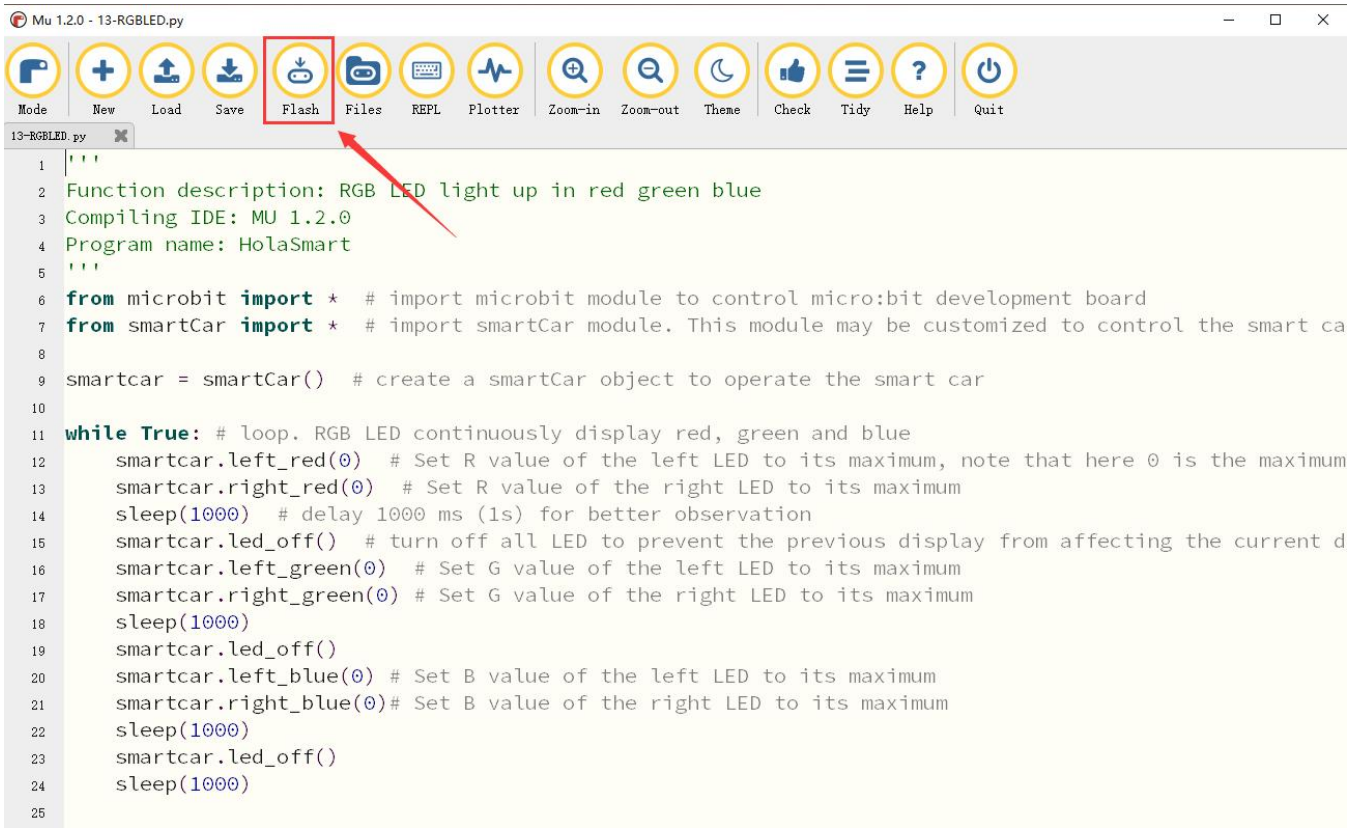


6.How to Download Code to Mciro:bit

Connect micro:bit board to computer via USB cable.



Click "Flash" to load the code to micro:bit board.

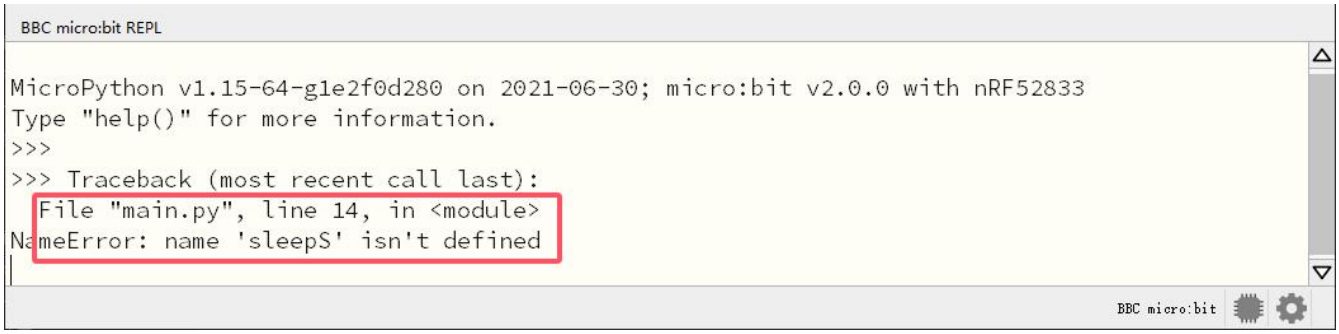



```
'''
Function description: RGB LED light up in red green blue
Compiling IDE: MU 1.2.0
Program name: HolaSmart
'''
from microbit import *  # import microbit module to control micro:bit development board
from smartCar import *  # import smartCar module. This module may be customized to control the smart ca

smartcar = smartCar()  # create a smartCar object to operate the smart car

while True: # loop. RGB LED continuously display red, green and blue
    smartcar.left_red(0)  # Set R value of the left LED to its maximum, note that here 0 is the maximum
    smartcar.right_red(0)  # Set R value of the right LED to its maximum
    sleep(1000)  # delay 1000 ms (1s) for better observation
    smartcar.led_off()  # turn off all LED to prevent the previous display from affecting the current d
    smartcar.left_green(0)  # Set G value of the left LED to its maximum
    smartcar.right_green(0) # Set G value of the right LED to its maximum
    sleep(1000)
    smartcar.led_off()
    smartcar.left_blue(0) # Set B value of the left LED to its maximum
    smartcar.right_blue(0)# Set B value of the right LED to its maximum
    sleep(1000)
    smartcar.led_off()
    sleep(1000)

```

If the code includes errors, it can also be downloaded to micro: bit, but it does not work properly. And a crying face  will be displayed on the dot matrix.

After downloading, the matrix prompts some errors and the wrong line number.

Click "REPL" and then press the reset button on the back (not buttons A or B), and the error message will be displayed in the REPL box:



Modified error "sleepS" to "sleep", and error exclusion.

For more tutorials of Mu, please visit: https://codewith.mu/en/tutorials/

Micro:bit Basic Projects

Project 1 Heartbeat

1.1 Introduction

Microbit Basic Projects utilizes the on-board sensors and modules.

This project is easy to conduct with a micro:bit main board, a Micro USB cable and a computer. The micro:bit LED dot matrix will display a beating heart. It serves as a start for your entry to the programming world!

1.2 Test Code

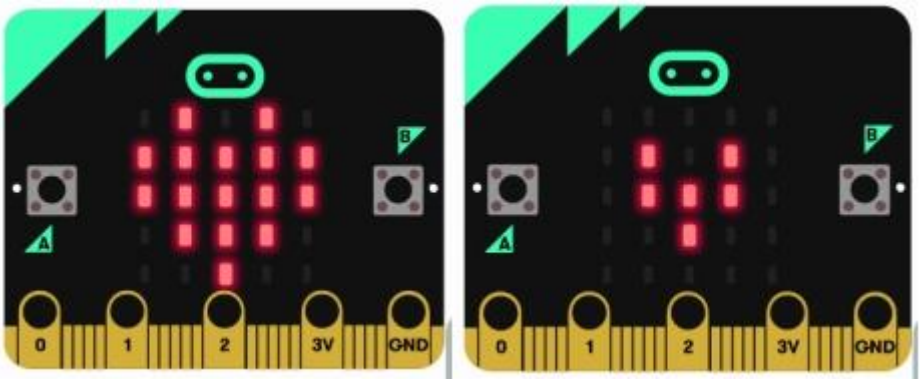
Open "BasicTutorialCodes" in "codes" folder, and open file "1-microbit-Heartbeat.py"

```
from microbit import *


while True:
    display.show(Image.HEART)
    sleep(500)
    display.show(Image.HEART_SMALL)
    sleep(500)
```

1.3 Test Result

Upload code to micro:bit, and power on via USB cable. The LED dot matrix on the micro:bit board circularly shows "  " and "  ".

If there is a download problem, disconnect the USB cable and the Micro:bit board, then reconnect them and reopen the compiler to try to download again.



1.4 Code Explanation

代码	Explanation
from microbit import	Import micro:bit library
while True:	Loop. Codes in it runs circularly.
display.show(Image.HEART)	LED dot matrix on micro:bit shows "  "
sleep(500)	Delay 500ms
display.show(Image.HEART_SMALL)	LED dot matrix on micro:bit shows "  "

1.5 Extended Knowledge

"Image.HEART" =  ; "Image.HEART_SMALL" = 

These can be directly called from microbit.

The following is a list of built-in images. If interested, you can select one to replace "Image.HEART" in code "show()"

- 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

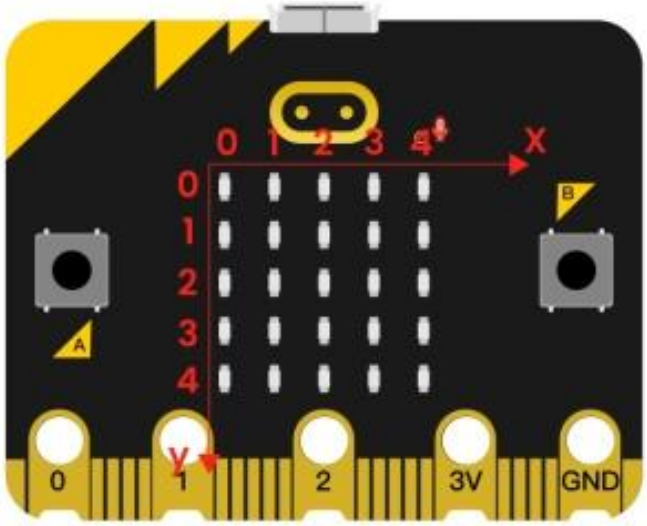
- Image.ARROW_N, Image.ARROW_NE, Image.ARROW_E, Image.ARROW_SE, Image.ARROW_S, Image.ARROW_SW, Image.ARROW_W, Image.ARROW_NW
- Image.TRIANGLE
- Image.TRIANGLE_LEFT
- Image.CHESSBOARD
- Image.DIAMOND
- Image.DIAMOND_SMALL
- Image.SQUARE
- Image.SQUARE_SMALL

- Image.RABBIT
- Image.COW
- Image.MUSIC_CROTCHET
- Image.MUSIC_QUAVER
- Image.MUSIC_QUAVERS
- Image.PITCHFORK
- Image.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

Project 2 Single LED Blinking

2.1 Introduction

The LED dot matrix of Micro: Bit main board contains 5x5=25 LED. We establish a coordinate system with axis x and y marking 0~4 from top to bottom and from left to right. Therefore, the LED sat in the second of the first line is (1,0) and the LED positioned in the fifth of the fourth column is (3,4), and others likewise.



2.2 Test Code

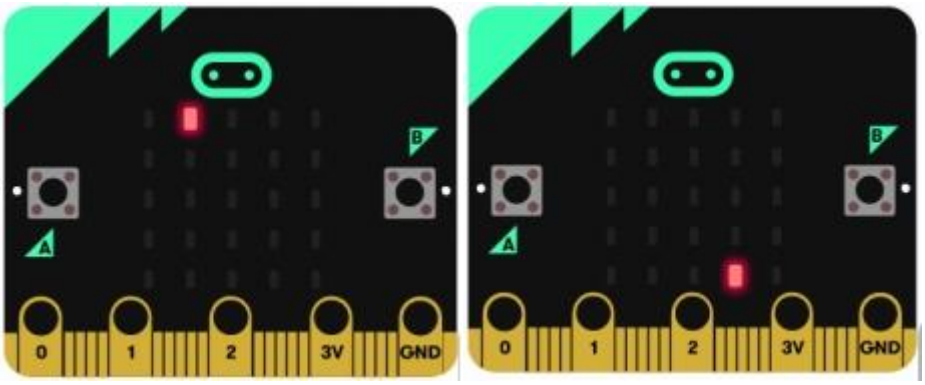
Open "BasicTutorialCodes" in "codes" folder, and open file "2-microbit-Light up an LED.py"

```
from microbit import *


val1 = Image("09000:""00000:""00000:""00000:""00000:")
val2 = Image("00000:""00000:""00000:""00000:""00090:")
val3 = Image("00000:""00000:""00000:""00000:""00000:")
```

```
while  True:
    display.show(val1)
    sleep(500)
    display.show(val3)
    sleep(500)
    display.show(val2)
    sleep(500)
    display.show(val3)
    sleep(500)
```

2.3 Test Result

Upload code to micro:bit, and power on via USB cable. You will see the LED in (1,0) and in (3,4) alternately light up for each 0.5s and this action will repeat.



2.4 Code Explanation

Code	**Explanation**
from microbit import	Import micro:bit library

Code	**Explanation**
val1 = Image("09000:""00000:""00000:" "00000:""00000:") val2 = Image("00000:""00000:""00000:" "00000:""00090:") val3 = Image("00000:""00000:""00000:" "00000:""00000:")	Assign the Image () to val1(every LED on micro:bit can be set to one of ten values. 0(zero) for off (brightness =0) and 9 for the brightest. Assign the Image () to val2. Assign the Image () to val3.
while True:	Loop. Codes in it runs circularly.
display.show(val1) sleep(500) display.show(val3) sleep(500)	LED at coordinate (1,0) ON/OFF for 0.5s
display.show(val2) sleep(500) display.show(val3) sleep(500)	LED at coordinate (3,4) ON/OFF for 0.5s

Project 3 5x5 LED Dot Matrix

3.1 Introduction

Dot matrices are very commonplace in daily life. They have found wide applications in LED advertisement screens, elevator floor display, bus stop announcement and so on.

The LED dot matrix of Micro: Bit main board contains 25 LEDs in a grid. Previously, we have succeeded in controlling a certain LED to light by integrating its position value into the test code. Theoretically, we can turn on many LEDs at the same time to show patterns, digits and characters.

What’ s more, we can also click ” show icon “ to choose the pattern we like to display. Last but not the least, we can design patterns by ourselves as well.

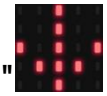
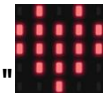
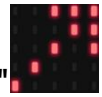
3.2 Test Code

Open "BasicTutorialCodes" in "codes" folder, and open file "3-microbit-5×5 LED Dot Matrix.py"

```
from microbit import *
val = Image("00900:""00900:""90909:""09990:""00900")
display.show('1')
sleep(500)
display.show('2')
sleep(500)
display.show('3')
sleep(500)
```

```
display.show('4')
sleep(500)
display.show('5')
sleep(500)
display.show(val)
sleep(500)
display.scroll("hello!")
sleep(200)
display.show(Image.HEART)
sleep(500)
display.show(Image.ARROW_NE)
sleep(500)
display.show(Image.ARROW_SE)
sleep(500)
display.show(Image.ARROW_SW)
sleep(500)
display.show(Image.ARROW_NW)
sleep(500)
display.clear()
```

3.3 Test Result

After uploading the code and powering on via the USB cable, you will see that the LED dot matrix start to show numbers "1", "2", "3", "4", "5", "  ", "hello!", "  ", "  ", "  ", "  ", "  ". Each interval is 500ms.

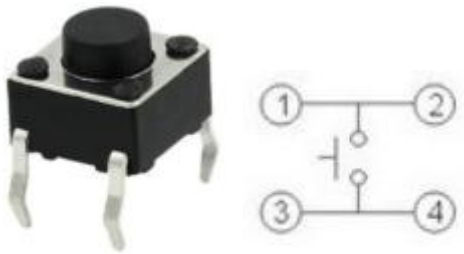
3.4 Code Explanation

Code	**Explanation**
from microbit import\	Import micro:bit library
val = Image("09000:""00000:""00000:""00000:""00000:")	Assign the Image () to val
display.show(val)	micro:bit LED dot matrix shows "  "
display.show('1')	micro:bit LED dot matrix shows "1"
sleep(500)	Delay 500ms
display.scroll("hello!")	micro:bit LED dot matrix shows "hello!"
display.show(Image.HEART)	micro:bit LED dot matrix shows "  "
display.show(Image.ARROW_NE) display.show(Image.ARROW_SE) display.show(Image.ARROW_SW) display.show(Image.ARROW_NW)	micro:bit LED dot matrix shows "  ", "  ", "  ", "  ".
display.clear()	Clear micro:bit LED dot matrix

Project 4 Programmable Buttons

4.1 Introduction

The button can control the circuit on and off. The circuit is disconnected when the button is not pressed. The circuit is connected as soon as you press the button, but it breaks when you release it. Why does it only work when you press it? It starts from the internal structure of the button, which is shown in the figure below.



Before the button is pressed, 1 and 2 are on, 3 and 4 are also on, but 1, 3 or 1, 4 or 2, 3 or 2, 4 are off (not working). Only when the button is pressed, 1, 3 or 1, 4 or 2, 3 or 2, 4 are on.

Micro: Bit main board boasts three buttons: two programmable buttons (marked with A and B), and a reset button at back. By pressing the two programmable buttons, three different signals can be input. We can press button A or B or both so that the LED dot matrix shows A, B and AB respectively. Let’ s get started!

4.2 Test Code

Open "BasicTutorialCodes" in "codes" folder, and open file "4-microbit-Programmable Buttons.py"

```
from microbit import *
```

```
while  True:
    if  button_a.is_pressed():
        display.show("A")
    elif  button_a.is_pressed()  and  button_b.is_pressed():
        display.scroll("AB")
    elif  button_b.is_pressed():
        display.show("B")
```

4.3 Test Result

After uploading the code, the 5*5 LED dot matrix shows A if button A is pressed and then released, shows B if button B pressed and released, and shows AB if button A and B pressed together and then released.

4.4 Code Explanation

Code	Explanation
from microbit import	Import micro:bit library
while True:	Loop. codes in it runs circularly.

if button_a.is_pressed():
display.show("A")
elif button_a.is_pressed() and button_b.is_pressed():
display.scroll("AB")
elif button_b.is_pressed():
display.show("B")
	If button A is pressed, dot matrix shows "A"
	If button A and B is pressed, dot matrix shows "AB"
	If button B is pressed, dot matrix shows "B"

Project 5 Temperature Detection

5.1 Introduction

The Micro:bit board is actually not equipped with a temperature sensor, but uses nNFR52833 chip for temperature detection. Therefore, the detected value is much closer to the temperature of the processor, so there maybe deviation from the ambient value.

In this project, we detect ambient temperature and display the results on the monitor. LED dot matrix will show different patterns in different temperatures.

nRF52833 chip

5.2 Test Code

Open "BasicTutorialCodes" in "codes" folder, and open file "5-microbit-Temperature Measurement.py"

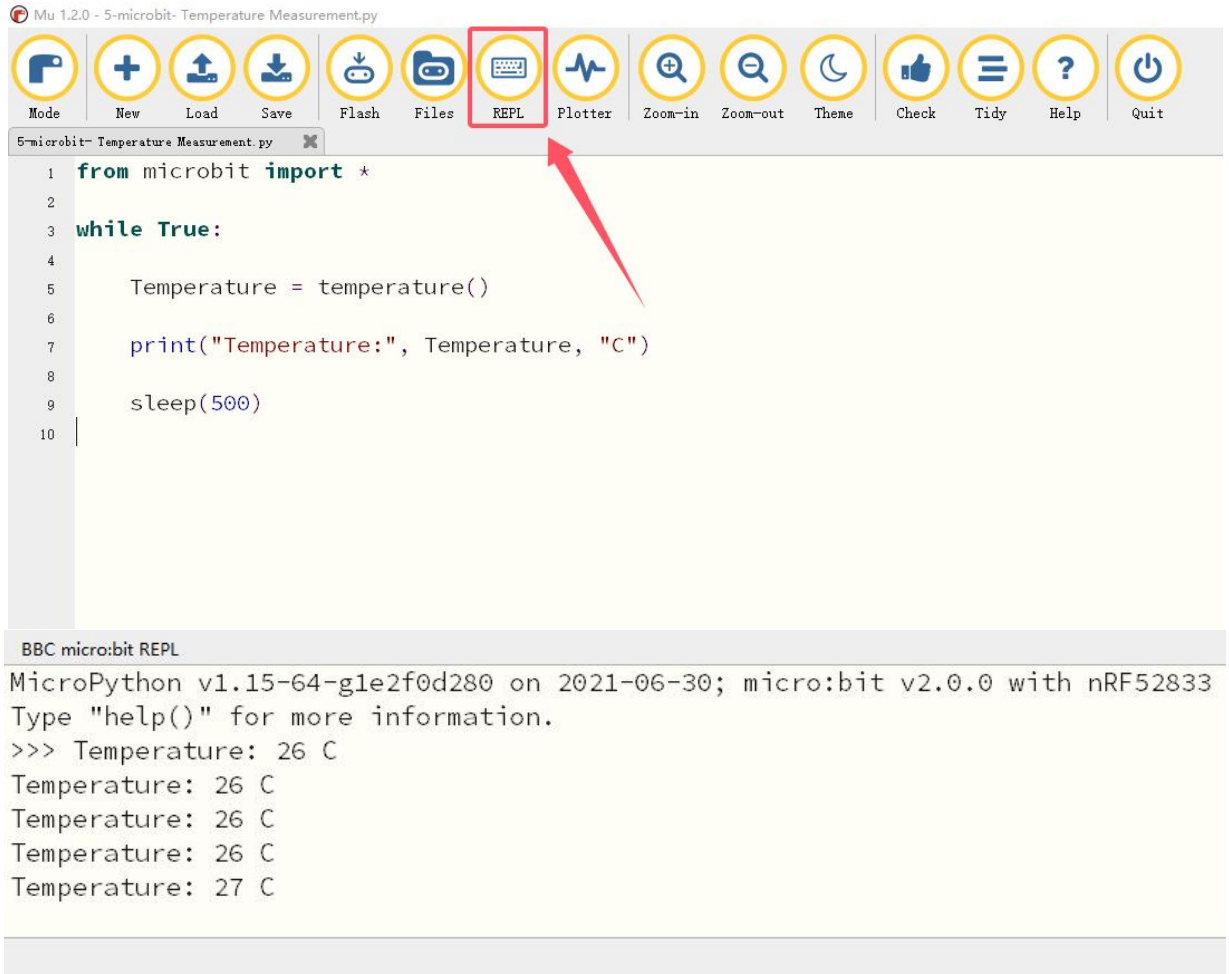
```
from microbit import *


while True:
```

```
Temperature = temperature()
print("Temperature:", Temperature, "C")
sleep(500)
```

5.3 Test Result

After uploading test code to micro:bit main board, powering the main board via the USB cable, and click "REPL" and the reset button on the back of micro:bit.

The REPL shows temperature values in ℃ (The letter C indicates the Celsius Degree, because ℃ may cause garbled words).



5.4 Code Explanation

Code	Explanation
from microbit import	Import micro:bit library
while True:	Loop. Codes in it runs circularly.
Temperature = temperature()	Assign temperature() to Temperature
print("Temperature:", Temperature, "C")	BBC microbit REPL prints temperature value in ℃.
sleep(500)	Delay 500ms

Project 6 Geomagnetic Sensor / Compass

6.1 Introduction

This project aims to explain the use of the Micro:bit geomagnetic sensor, which determines bearings by showing readings in the value from 0 to 360 relative to the north magnetic pole. We need to calibrate it for the first by rotating. Metal materials around may attenuate the accuracy of the reading and calibration. Here is the position of Geomagnetic Sensor:

6.2 Test Code

Open "BasicTutorialCodes" in "codes" folder, and open file "6-microbit-Magnetic sensor -1.py"

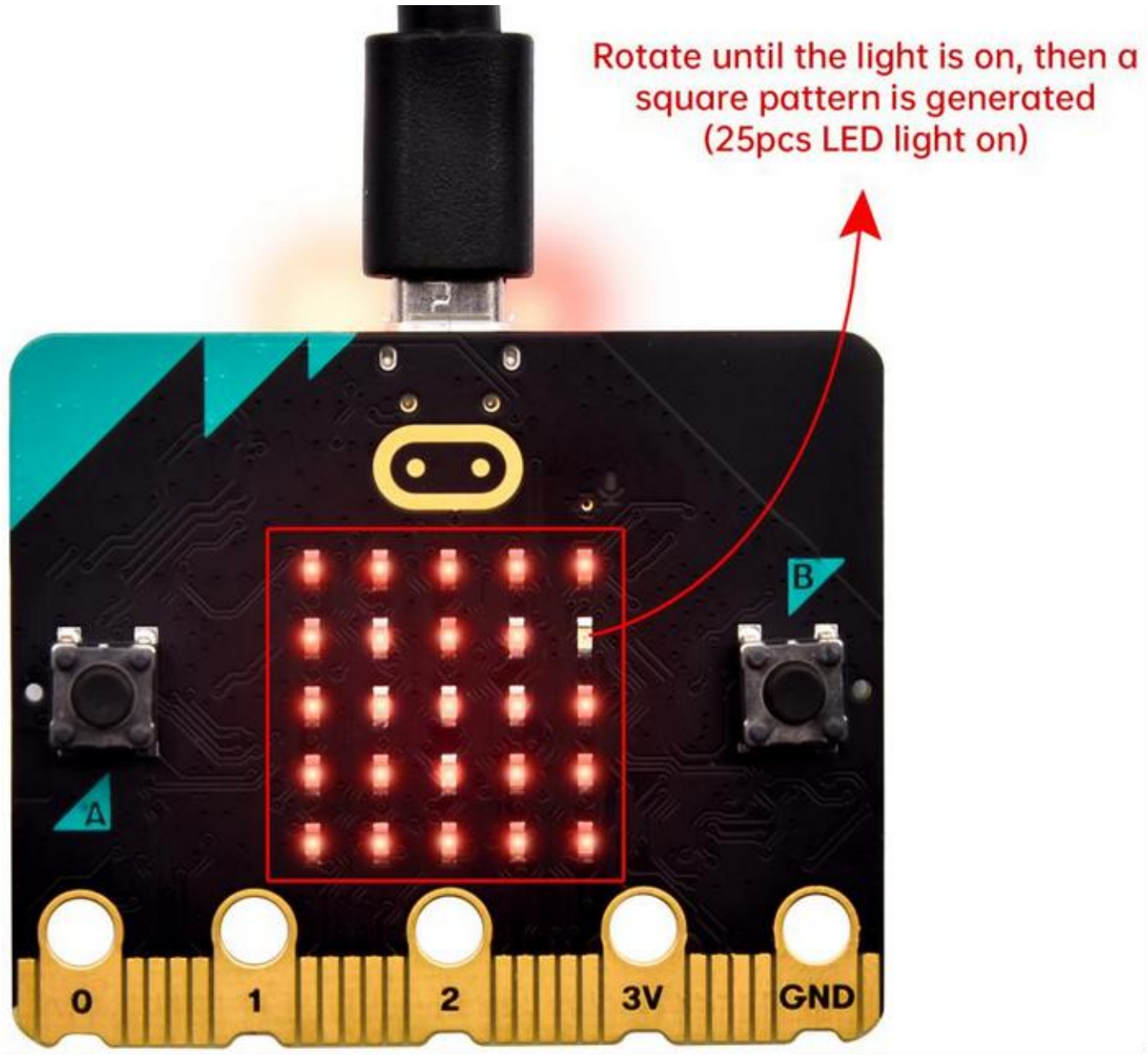
```
from microbit import *

compass.calibrate()

while True:

    if button_a.is_pressed():
```

6.3 Test Result

Note: it is imperative to calibrate the Micro:bit board because different geomagnetic fields exist in different places. And the board requires a calibration for the first using time.

After uploading code and powering on, press the button A and the board need to be calibrate when you see LED dot matrix shows "TILT TO FILL SCREEN".

Rotate the board until all 25 red LEDs are on as shown below.



After completing calibration, a smile  will appear.

Press button A, and the detected magnetometer value will show. And the direction north, east, south and west correspond to 0°, 90°, 180° and 270° respectively.

6.4 Extension Code

Open "BasicTutorialCodes" in "codes" folder, and open file "6-microbit-Magnetic sensor -2.py"

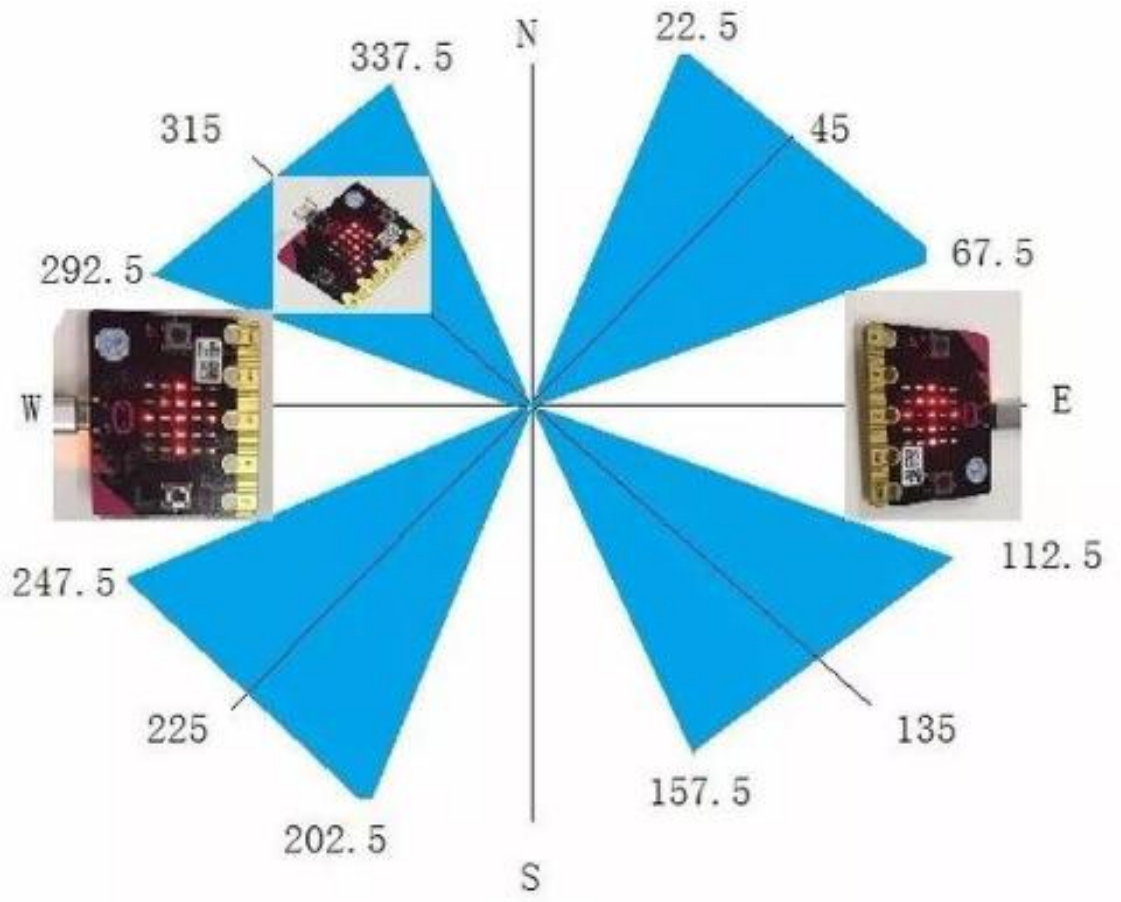
```
from microbit import *
compass.calibrate()
x = 0
while True:
    x = compass.heading()
    if x >= 293 and x < 338:
        display.show(Image("00999:""00099:""00909:""09000:""90000"))
    elif x >= 23 and x < 68:
        display.show(Image("99900:""99000:""90900:""00090:""00009"))
    elif x >= 68 and x < 113:
        display.show(Image("00900:""09000:""99999:""09000:""00900"))
    elif x >= 113 and x < 158:
        display.show(Image("00009:""00090:""90900:""99000:""99900"))
    elif x >= 158 and x < 203:
        display.show(Image("00900:""00900:""90909:""09990:""00900"))
    elif x >= 203 and x < 248:
        display.show(Image("90000:""09000:""00909:""00099:""00999"))
    elif x >= 248 and x < 293:
        display.show(Image("00900:""00090:""99999:""00090:""00900"))
    else:
        display.show(Image("00900:""09990:""90909:""00900:""00900"))
```

6.5 Test Result

The arrow on the dot matrix always points to the current magnetic North Pole.

As shown below, the arrow pointing to the upper right when the value ranges from 292.5 to 337.5.

0.5 can’ t be input in the code, so the values we get are 293 and 338. We add other statements to make a set of complete code.



After uploading code and calibrating, place the board horizontally. Tilt micro:bit board and the LED dot matrix displays the direction signs.

6.6 Code Explanation

Code	Explanation
from microbit import	Import micro:bit library
compass.calibrate()	Compass calibration
while True	Loop. Codes in it runs circularly.
if button_a.is_pressed(): display.scroll(compass.heading())	When button A is pressed, the dot matrix shows the sensor value
x = 0	Set variable x to 0
x = compass.heading()	Assign the sensor value to x
if...elif...else	Conditional statement
display.show(Image("00999:""00099:""00909:""09000:""90000"))	Put the board horizontally, and it shows arrows: Point to the top right corner
display.show(Image("99900:""99000:""90900:""00090:""00009"))	Point to the upper left corner
display.show(Image("00900:""09000:""99999:""09000:""00900"))	Point to the left
display.show(Image("00009:""00090:""90900:""99000:""99900"))	Point to the lower left corner
display.show(Image("00900:""00900:""90909:""09990:""00900"))	Point directly below
display.show(Image("90000:""09000:""00909:""00099:""00999"))	Point to the lower right corner
display.show(Image("00900:""00090:""99999:""00090:""00900"))	Point to the right
display.show(Image("00900:""09990:""90909:""00900:""00900"))	Point directly above

Project 7 Accelerometer

7.1 Introduction

The micro:bit board boasts a built-in LSM303AGR acceleration sensor (accelerometer) which is with resolution of 8/10/12 bits and range of ±2g, ±4g, or ±8g. It is often used to detect the posture of devices.

In this project, we will introduce how to measure the position of the board with the accelerometer. And then we may have a look at the original three-axis value output by the accelerometer.

7.2 Test Code

Open "BasicTutorialCodes" in "codes" folder, and open file "7-microbit-Three-axis acceleration sensor -1.py"

```
from microbit import *


while True:
    gesture = accelerometer.current_gesture()


    if gesture == "shake":
        display.show("1")
    if gesture == "up":
        display.show("2")
    if gesture == "down":
        display.show("3")
    if gesture == "face up":
```

```
        display.show("4")
    if gesture == "face down":
        display.show("5")
    if gesture == "left":
        display.show("6")
    if gesture == "right":
        display.show("7")
    if gesture == "freefall":
        display.show("8")
```

7.3 Test Result

After uploading code and powering on, if we shake the Micro:Bit board(any direction), the LED dot matrix displays the digit “1”.

When the logo is kept above, number 2 displays.

When it is kept upside down (logo below the LED dot matrix), it shows as below.

When it is placed on the desk, the number 4 appears.

When it is covered on the desk, the number 5 exhibits.

When the board is tilted to the left, the LED dot matrix shows the number 6 as shown below.

When the board is tilted to the right , the LED dot matrix displays the number 7 as shown below:

When the board falls down to the floor (a free fall), the LED dot matrix shows the number 8. (Please note that this test is not recommended for it may damage the main board.)

If you’ d like to try this function, you can also set the acceleration to 3g, 6g or 8g. But still ,we do not recommend.

7.4 Extension Code

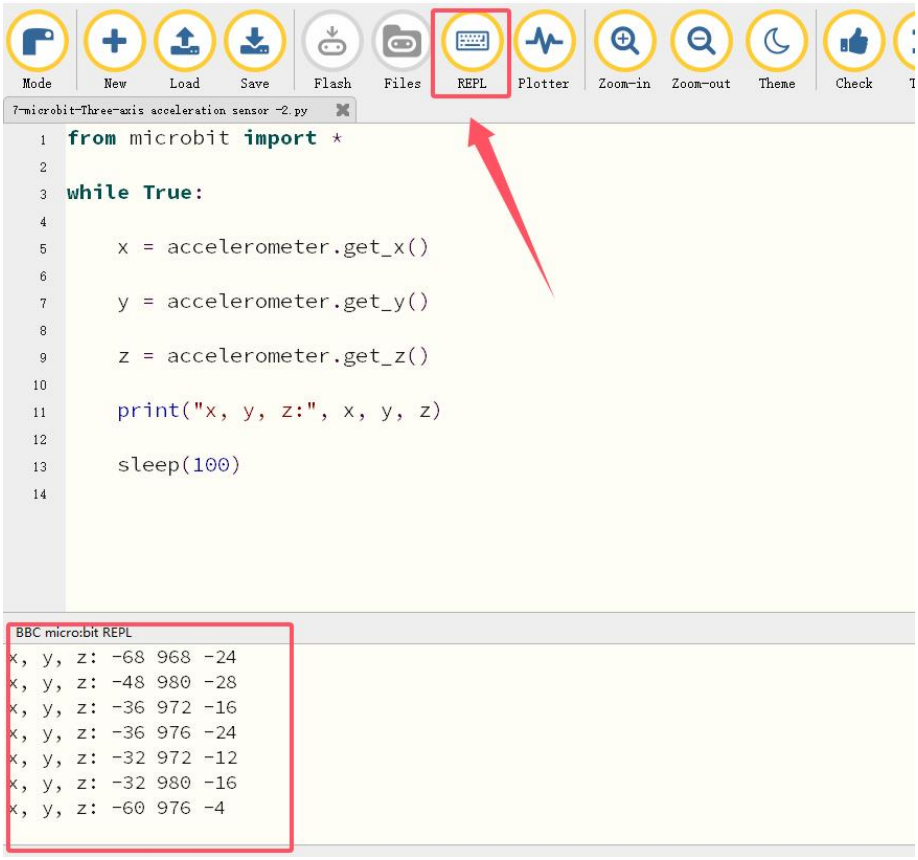
Open "BasicTutorialCodes" in "codes" folder, and open file "7-microbit-Three-axis acceleration sensor -2.py"

```
from microbit import *


while True:
    x = accelerometer.get_x()
    y = accelerometer.get_y()
    z = accelerometer.get_z()
    print("x, y, z:", x, y, z)
    sleep(100)
```

7.5 Test Result

After uploading test code to micro:bit main board, powering the main board via the USB cable, and click "REPL" and the reset button on the back of micro:bit.

BBC microbit REPL shows the decomposition value of acceleration in X axis, Y axis and Z axis respectively.



7.6 Code Explanation

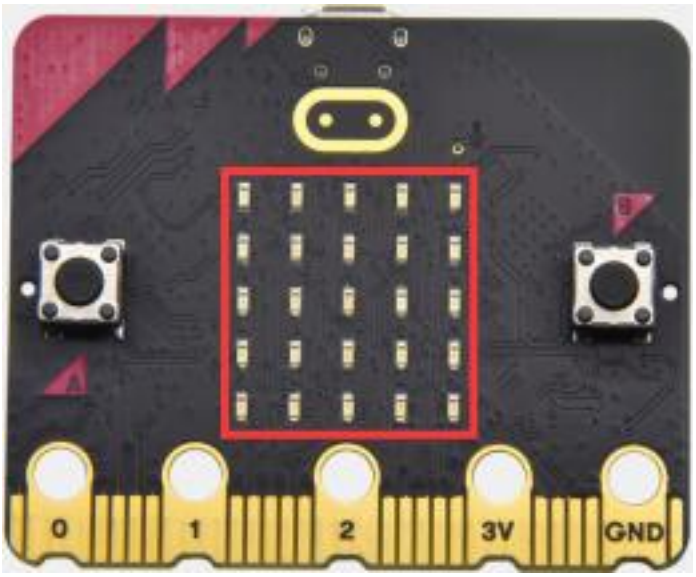
Code	Explanation
from microbit import	Import micro:bit library
gesture = accelerometer.current_gesture()	Assign accelerometer.current_gesture() to gesture
while True:	Loop. Codes in it runs circularly.
if gesture == "shake":display.show("1") if gesture == "up": display.show("2") if gesture ==	Shake micro:bit, and the dot matrix shows "1" Put the micro:bit Logo upwards, and the dot matrix shows "2" Put the micro:bit Logo downwards, and the

Code	Explanation
"down":display.show("3") if gesture == "face up":display.show("4") if gesture == "face down":display.show("5") if gesture == "left": display.show("6") if gesture == "right":display.show("7") if gesture == "freefall":display.show("8")	dot matrix shows "3" Put the micro:bit dot matrix upwards, and the dot matrix shows "4" Put the micro:bit dot matrix downwards, and the dot matrix shows "5" Tilt micro:bit to the left, and the dot matrix shows "6" Tilt micro:bit to the right, and the dot matrix shows "7" micro:bit falls down, and the dot matrix shows "8"
x = accelerometer.get_x() y = accelerometer.get_y() z = accelerometer.get_z()	Read the acceleration of axis x (mg) in integer, and assign it to x Read the acceleration of axis y (mg) in integer, and assign it to y Read the acceleration of axis z (mg) in integer, and assign it to z
print("x, y, z:", x, y, z)	BBC microbit REPL prints the acceleration values(mg) of the axis x, y, z
sleep(100)	Delay 100ms

Project 8 Light Brightness Detection

8.1 Introduction

In this experiment, we will use the micro:bit board to detect light intensity. Since the micro:bit board does not contains its own photoresistor, the LED dot matrix will shoulder this job. The light signal will convert into input, and the voltage decay time is sampled so that the detected light intensity is a relative value.



Light intensity sensing area

8.2 Test Code

Open "BasicTutorialCodes" in "codes" folder, and open file "8-microbit-Detect Light Intensity by Microbit.py"

```
from microbit import *


while True:
    Lightintensity = display.read_light_level()
```

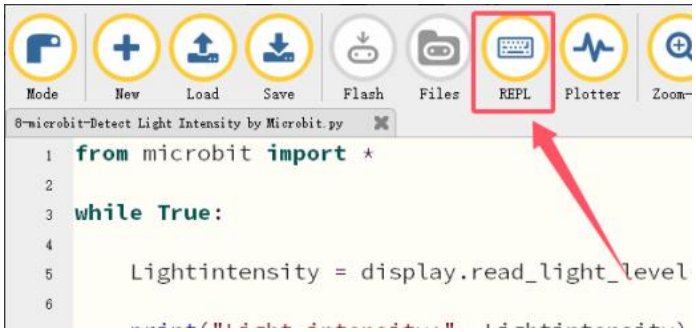
```
    print("Light  intensity:",  Lightintensity)
sleep(100)
```

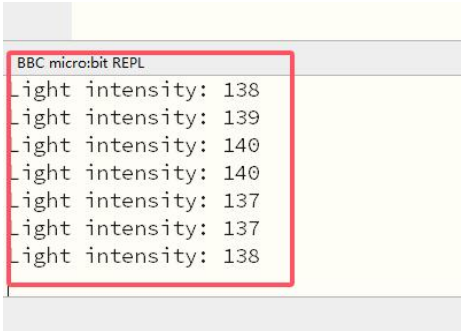
8.3 Test Result

After uploading test code to micro:bit main board, powering the main board via the USB cable, and click "REPL" and the reset button on the back of micro:bit.

BBC microbit REPL shows the analog value detected by the on-board photoresistor.

When the LED dot matrix is covered by hand, the light intensity is approximately 0; when the LED dot matrix is exposed to light, the light intensity value gets greater.





8.4 Code Explanation

Code	Explanation
from microbit import	Import micro:bit library
gesture = accelerometer.current_gesture()	Assign accelerometer.current_gesture() to gesture
while True:	Loop. Codes in it runs circularly.

Code	Explanation
Lightintensity = display.read_light_level()	Assign display.read_light_level() to Lightintensity
print("Light intensity:", Lightintensity)	BBC microbit REPL prints the brightness value detected by photoresistor.
sleep(100)	Delay 100ms

Project 9 Speaker

9.1 Introduction

Micro: Bit board boasts an built-in speaker, which makes sound to the programs easier. It is also able to make a variety of interesting sound as well as all kinds of tones. You can even compose a music!

Speaker

9.2 Test Code

Open "BasicTutorialCodes" in "codes" folder, and open file "9-microbit-Speaker.py"

```
from microbit import *

import audio

display.show(Image.MUSIC_QUAVER)

while True:
    audio.play(Sound.GIGGLE)
    sleep(1000)
    audio.play(Sound.HAPPY)
    sleep(1000)
    audio.play(Sound.HELLO)
    sleep(1000)
    audio.play(Sound.YAWN)
    sleep(1000)
```

9.3 Test Result

After uploading code and powering on, the speaker utters sound and the LED dot matrix shows the logo of music.

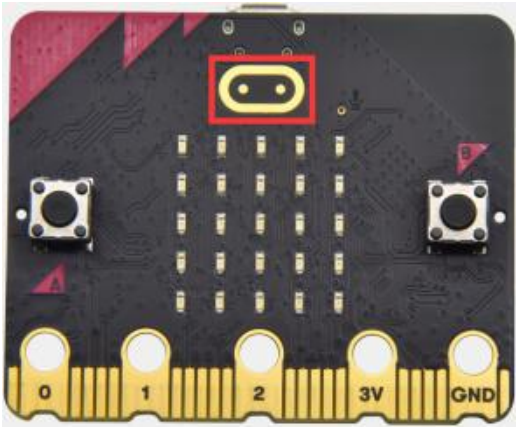
9.4 Code Explanation

Code	Explanation
from microbit import	Import micro:bit library
import audio	Audio library
while True:	Loop. Codes in it runs circularly.
audio.play(Sound.GIGGLE)	Sound giggle
sleep(1000)	Delay 1000ms

Project 10　Touch-sensitive Logo

10.1 Introduction

The Micro: Bit main board is equipped with a golden touch-sensitive logo, which can act as an extra button. This capacitive touch sensor senses small changes in the electric field when it is pressed or touched, just like screens of phone or tablet. When you touch it, you can control the micro: bit board to do something.



Touch-sensitive Logo

10.2 Test Code

Open "BasicTutorialCodes" in "codes" folder, and open file "10-microbit-Touch Sensitive Logo.py"

```
from microbit import *
time = 0
start = 0
running = False

while True:
    if button_a.was_pressed():
        running = True
        start = running_time()
    if button_b.was_pressed():
        if running:
            time += running_time() - start
        running = False
    if pin_logo.is_touched():
        if not running:
            display.scroll(int(time/1000))

    if running:
        display.show(Image.HEART)
        sleep(300)
        display.show(Image.HEART_SMALL)
        sleep(300)
    else:
```

```
        display.show(Image.ASLEEP)
```

10.3 Test Result

After uploading the code and powering on, press button A to start the stopwatch. After that, the LED dot matrix exhibits a beating heart. When you press button B, it stops. It constantly adds time value, like a real stopwatch. Touch the logo, and the measured time will show on the dot matrix in the unit of second.

Press the reset button on the back of the board to zero out the value.

10.4 Code Explanation

(1) Micro:bit records time when it was started in milliseconds. This is called runtime.

(2) When you press button A, a variable named start is set to the current runtime.

(3) When you press button B, the start time will be subtracted from the new runtime to calculate how much time has elapsed since you started the stopwatch. This difference was added to the total time, which is stored in a variable named time.

(4) If you press the LOGO, board displays the total time in millisecond. It gives a result of an integer.

(5) The program also uses a Boolean variable called running. A Boolean variable can have only two values: true or false. If "running" is "true", the stopwatch has started. If "running" is "false", the stopwatch is not started or has been stopped.

(6) If "running" is "true", a beating heart will be shown on the dot matrix.

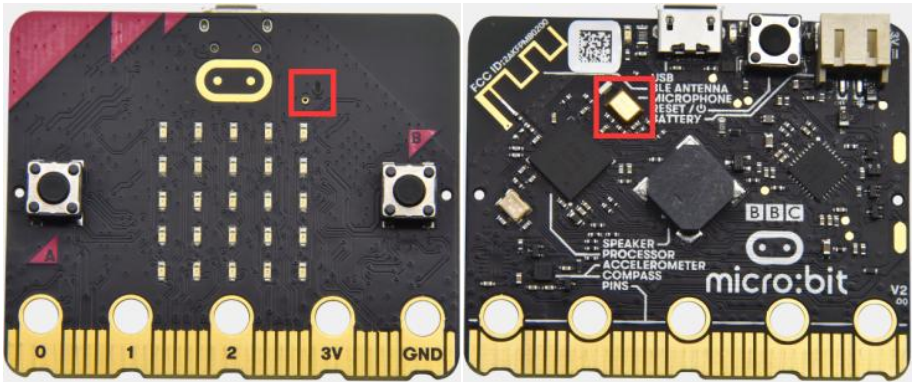
(7) If the stopwatch is stopped, "running" is "false", and touch the LOGO to show the total time.

(8) If the stopwatch is started, "running" is "true", and make sure the time variable changes only when button B is pressed and the code also prevents incorrect reading.

Project 11 Microphone

11.1 Introduction

The Micro:bit main board is built with a microphone which can test the volume of ambient environment. When you clap, the microphone LED indicator turns on. So, you can make a disco lighting changing with music. The microphone is placed on the opposite side, and an LED indicator is next to the hole that lets sound pass. When the board detects sound, the LED indicator lights up.



11.2 Test Code

Open "BasicTutorialCodes" in "codes" folder, and open file "11-microbit-Microphone-1.py"

```
from microbit import *


while True:
    if microphone.current_event() == SoundEvent.LOUD:
        display.show(Image.HEART)
        sleep(200)
    if microphone.current_event() == SoundEvent.QUIET:
        display.show(Image.HEART_SMALL)
```

11.3 Test Result

After uploading test code to micro:bit main board and powering the board via the USB cable, the LED dot matrix displays "  " when you clap, and "  " appears when it is quiet around.

11.4 Extension Code

Open "BasicTutorialCodes" in "codes" folder, and open file "11-microbit-Microphone-2.py"

```
from microbit import *
maxSound = 0
lights = Image("11111:"
```

```
                    "11111:"
                    "11111:"
                    "11111:"
                    "11111")
# ignore first sound level reading
soundLevel = microphone.sound_level()
sleep(200)

while True:
    if button_a.is_pressed():
        display.scroll(maxSound)
    else:
        soundLevel = microphone.sound_level()
        display.show(lights * soundLevel)
        if soundLevel > maxSound:
            maxSound = soundLevel
```

11.5 Test Result

Upload test code and power on.

When the button A is pressed, the LED dot matrix displays the value of the biggest volume. Please note that the biggest volume can be reset via the Reset button.

Clap your hands. The louder the measured sound is, the brighter these LED will be.

11.4 Code Explanation

Code	Explanation
from microbit import	Import micro:bit library
while True:	Loop. Codes in it runs circularly.
if microphone.current_event() == SoundEvent.LOUD:display.show(Image.HEART)sleep(200) if microphone.current_event() == SoundEvent.QUIET:display.show(Image.HEART_SMALL)	If the microphone detects sound, the dot matrix shows  for 200ms. If it detects no sound,  will be displayed.
maxSound = 0	Set maxSound initial value to 0
lights = Image("11111:""11111:""11111:""11111:""11111")	Assign Image() to lights
soundLevel = microphone.sound_level()	Assign microphone.sound_level() to soundLevel
if button_a.is_pressed():display.scroll(maxSound) else: soundLevel = microphone.sound_level()display.show(lights * soundLevel) if soundLevel > maxSound:maxSound = soundLevel	Press button A to show the sound value on the dot matrix. Or else, assign microphone.sound_level() to soundLevel. The louder the sound is, the brighter the LED will be.

Project 12　Bluetooth

Introduction

Although the micro: bit boasts a low-power Bluetooth module that can send data, it is with only 16k RAM. The BLE stack will occupy 12k, so there is not enough space to run the microPython. This means only one of microPython and Bluetooth can be operated.

In the future, 32k RAM may be equipped to support Bluetooth.

https://microbit-micropython.readthedocs.io/en/latest/ble.html

The above experiments use the micro:bit LED dot matrix, yet the following ones will combine micro:bit and keyestudio Mecanum 4WD smart car V2.0 board, as well as multiple sensors and modules integrated on the car, including IR receiver(microPython is not supported), ultrasonic sensor, servo, WS2812 RGB and RGB LED.

Attention: To prevent burning out the micro:bit board, please disconnect USB cable and turn OFF the power supply on the motor drive board. So does the car expansion board.

Smart Car Projects

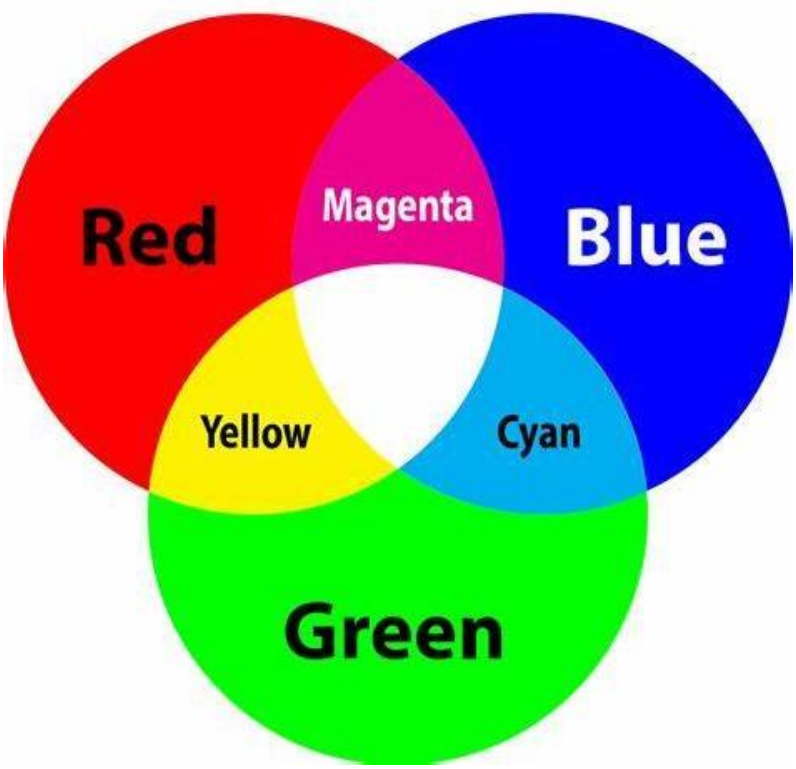
Project 13　RGB LED

13.1 Introduction

RGB color mode is a color standard in the industry. A variety of colors can be got through the change of red (R), green (G), blue (B) three color channels and their superposition. This standard is one of the most widely used color systems, which includes almost all the colors that human vision can perceive.

Most of the display adopts RGB color standard. They show colors through the electron gun emitting red, green and blue. Generally, computers can display 32-bit with more than 10 million colors. All the colors are made up of the three colors mixed in different proportions. And a group of reds, greens, and blues is a minimal display unit. Any color can be recorded and expressed by a set of RGB values, so they are also known as the three primary colors of light: R(red), G(green), B(blue).

RGB, simply, shows colors by mixing red, green, blue. When they are superficially combined, the color is mixed, and the brightness is equal to the sum of the three. Thus, the more mixed lights is, the higher the brightness will be. That’ s what we called addictive mixing. The brightest area in the center is white. Usually we called seven-color LED in daily life.

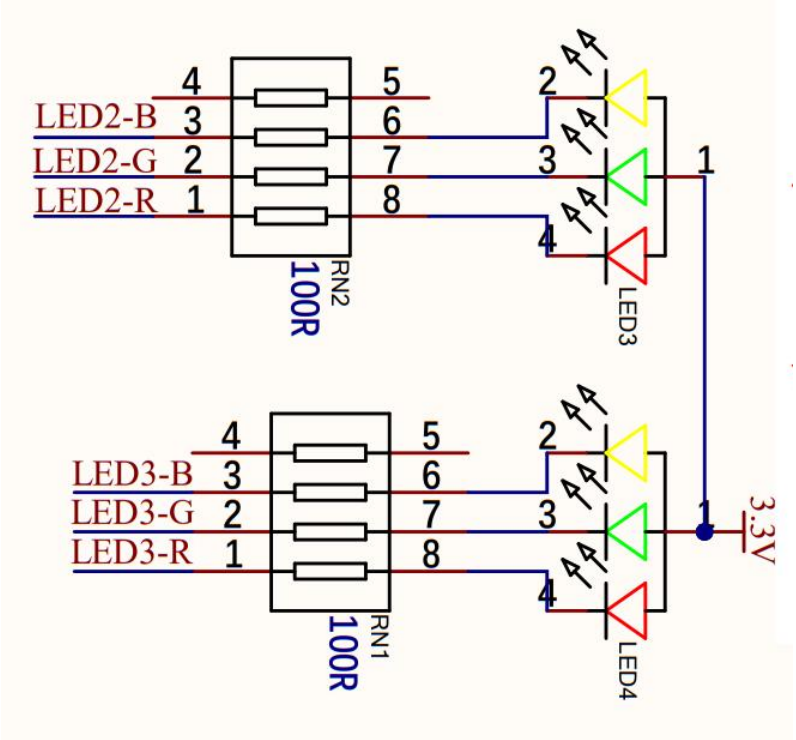


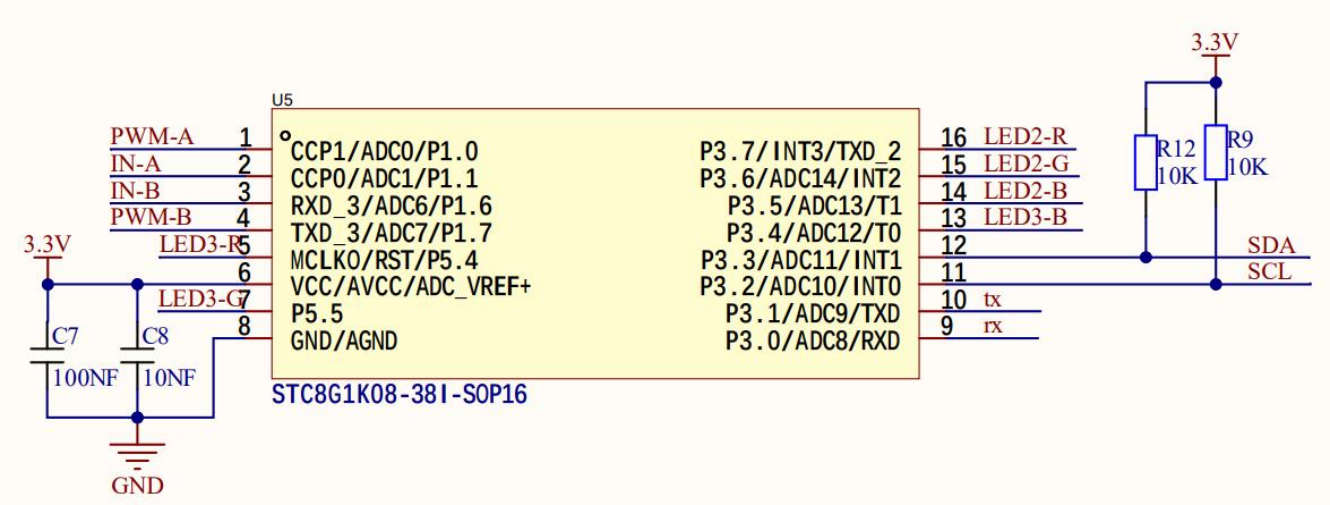
Color	RGB (R,G,B)	Color	RGB (R,G,B)
LED Off	255,255,255	red	0,255,255
green	255,0,255	blue	255,255,0
cyan	255,0,0	magenta(purple)	0,255,0
yellow	0,0,255	white	0,0,0
.....	.....	.....	.....

This LED is common anode, so 255 is the darkest value while 0 is the brightest.

In this project, we will finish two experiments. One is to light up the two LED in red, green, blue, cyan, purple, yellow and white circularly; The other is to make them gradient-display different colors of light.

13.2 Working Principle





Working Principle:

With Microbit as the master, it sends instructions to the slave STC8G1K08 through IIC to output PWM to control RGB LED. This greatly saves the IO ports because only IIC is needed to control two motors and RGB LED.

13.3 Code Explanation

RGB LED related functions in smartCar.py library:

Left.red(0-255): set left RGB LED to red (0 for the brightest, 255 for the darkest)

Left.green(0-255): set left RGB LED to green (0 for the brightest, 255 for the darkest)

Left.blue(0-255): set left RGB LED to blue (0 for the brightest, 255 for the darkest)

right.red(0-255): set right RGB LED to red (0 for the brightest, 255 for the darkest)

righr.green(0-255): set right RGB LED to green (0 for the brightest, 255 for the darkest)

righr.blue(0-255): set right RGB LED to blue (0 for the brightest, 255 for the darkest)

Led_off(): turn off all LED

13.4 Test Code

Open "smartCarCodes" folder in "codes" folder, and open file "13-RGBLED.py"

```
'''
Function description: RGB LED light up in red green blue
Compiling IDE: MU 1.2.0
Program name: HolaSmart
'''
from microbit import *  # import microbit module to control micro:bit development board
from smartCar import *  # import smartCar module. This module may be customized to control the smart car


smartcar = smartCar()  # create a smartCar object to operate the smart car


while True: # RGB LED continuously display red, green and blue
```

```
    smartcar.left_red(0)  # Set R value of the left LED to its maximum, note that here 0 is th
e maximum while 255 is the darkest

    smartcar.right_red(0)  # Set R value of the right LED to its maximum

    sleep(1000)  # delay 1000 ms (1s) for better observation

    smartcar.led_off()  # turn off all LED to prevent the previous display from affecting the cu
rrent display

    smartcar.left_green(0)  # Set G value of the left LED to its maximum

    smartcar.right_green(0) # Set G value of the right LED to its maximum

    sleep(1000)

    smartcar.led_off()

    smartcar.left_blue(0) # Set B value of the left LED to its maximum

    smartcar.right_blue(0)# Set B value of the right LED to its maximum

    sleep(1000)

    smartcar.led_off()

    sleep(1000)
```

13.5 Test Result

After uploading the code, the two RGB LED on the front of the car will be on in red, green, blue, cyan, purple, white, yellow in turn, with each color lighting for one second, and then they will go off for one second. These actions repeat.

Project 14　Breathing RGB LED

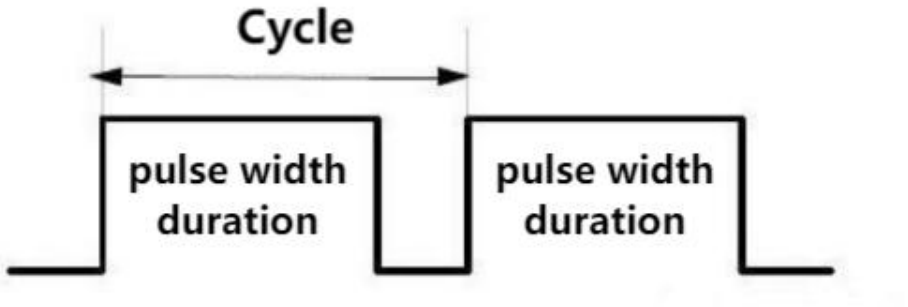
14.1 Introduction

Breathing LED uses the on-board programmable PWM to output analog waves whose duty cycle controls the brightness of LED. With PWM, the brightness is able to change as time, so the LED will gradually light up and dim, generating colorful or tranquil light effect.

14.2 Working Principle

PWM (Pulse width modulation) simulates the change of analog signal through digital signal.

Pulse width is the high level in a complete square wave cycle. So, pulse width modulation is to adjust the high level(of course, in other words, low level is also adjusted).



- **PWM frequency**:

  the number of times the signal going from high level to low level and back to high level in 1 second (one cycle), that is, how many cycles there are in a second.

  Unit: Hz

  Expression: 50Hz 100Hz

- **PWM cycle**

$$T = \frac{1}{f} \qquad \text{cycle} = \frac{1}{frequency}$$

If the frequency is 50Hz, the cycle will be 20ms, i.e., there are 50 PWM cycles in one second.

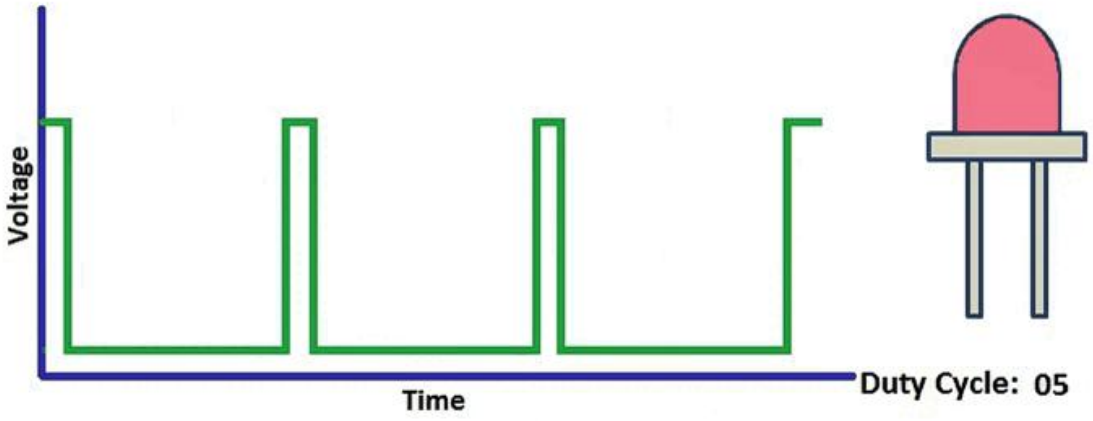
- **PWM duty cycle**

The ratio of high level time to the whole cycle time.

Unit: %(1% ~ 100%)

Cycle: The time of a pulse signal. The number of cycles in 1s equals the frequency.

Pulse width time: high level time.



The relationship between duty cycle and LED brightness

14.3 Code Explanation

for num in range(0,255):

This for loop traverses an integer from 0 to 255. range(0,255) generates a sequence containing 0 but not 255. That means a total of 255 cycles.

14.4 Test Code

Open "smartCarCodes" folder in "codes" folder, and open file "14-PWM-RGBLED.py"

```
'''
Function description: RGB LED dimming and lighting up
Compiling IDE: MU 1.2.0
Program name: HolaSmart
'''
from microbit import *  # import microbit module to control micro:bit development board
from smartCar import *  # import smartCar module. This module may be customized to control the smart car


smartcar = smartCar()  # create a smartCar object to operate the smart car


while True:  # loop. LED dims and lights up
    for num in range(0 , 255):  # for loop. range from 0 to 254 (range excluds endpoint)
        num += 1  # num adds 1: num = num + 1
        smartcar.left_red(255 - num)  # set the R value of left LED to decrease with the increase of num
        smartcar.right_red(255 - num)  # set the R value of right LED to decrease with the increase of num
        sleep(10)  # delay 10ms


    for num in range(0 , 255):  # one more for loop: LED from off to on
        num += 1  # num adds 1
```

```
            smartcar.left_red(num)   # set the R value of left LED to increase with the increase of
 num
            smartcar.right_red(num)   # set the R value of right LED to increase with the increase
of num
            sleep(10)   #  delay  10ms
```

14.5 Test Result

After uploading the code, both RGB LED will gradually light up in red and then dim. Then they gradually light up again, in a loop.

Project 15　Photoresistor

15.1Introduction

Have you ever noticed that the street lights outside turn on when it gets dark and turn off when it gets light! In this project, we will mainly adopt a photoresistor to simulate this light.

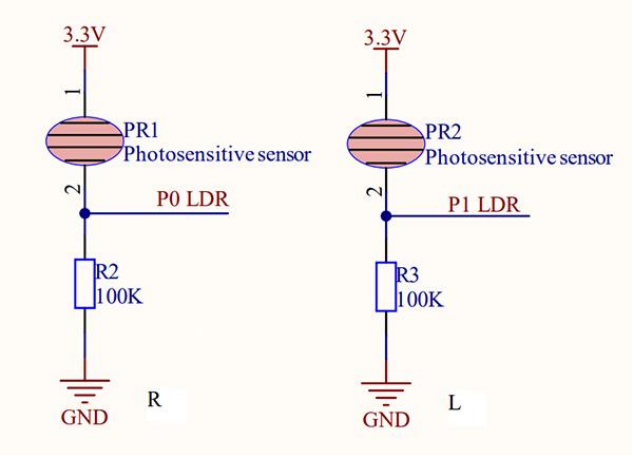
15.2 Working Principle

Photoresistor works on the basis of semiconductor photoelectric effect, so it is very sensitive to ambient light whose resistance value changes with light intensity. We design a circuit with photoresistor in this experiment.

There is an analog port on the photoresistor. When the light gets brighter, the resistance decreases and the port voltage increases, so the analog value of the MCU also rises. On the contrary, when the light weakens, the resistance increases and the voltage decreases, so the analog value of the

MCU becomes smaller. In this way, the photoresistor can be used to read the corresponding analog values and sense the ambient light.

Generally, photoresistors are applied to light measurement and control, photoelectric conversion (converting changes in light into changes in electricity) and light regulation in light control circuits and switches.



15.3 Code Explanation

ldr_L = pin1.read_analog(): read the analog value of pin P1, and assign it to variable ldr_l.

15.4 Test Code

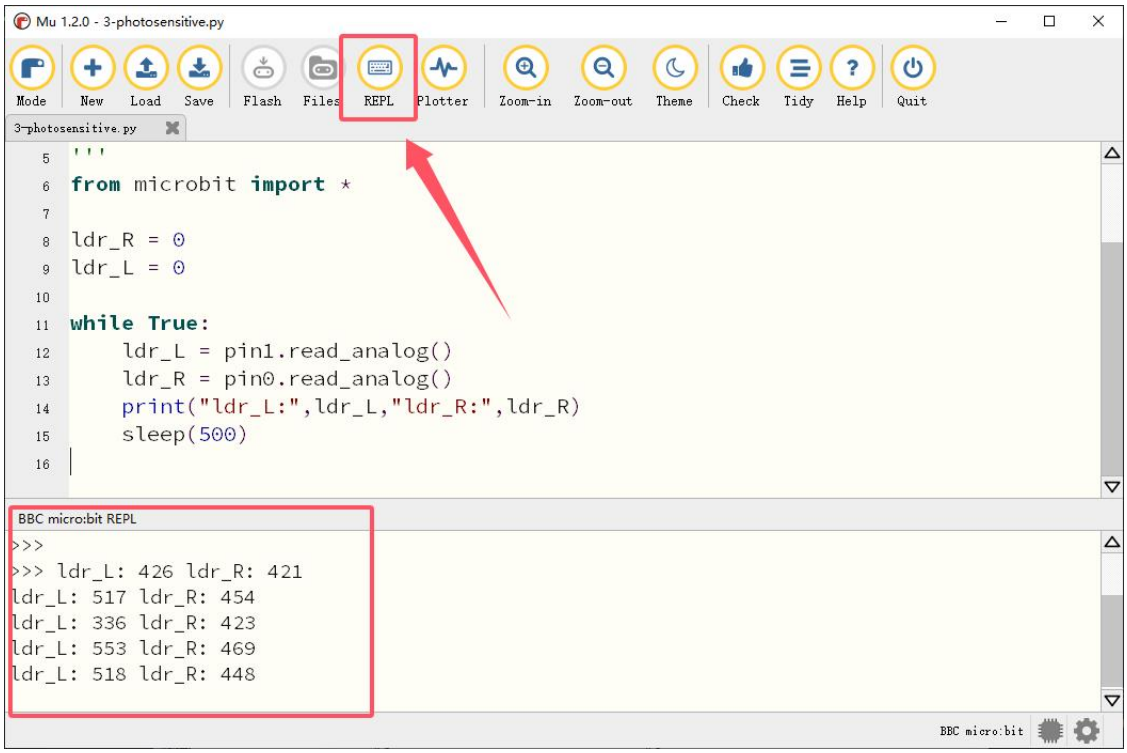
Open "smartCarCodes" folder in "codes" folder, and open file "15-photosensitive.py"

```
'''
Function description: serial monitor prints photoresistor value
Compiling IDE: MU 1.2.0
Program name: HolaSmart
'''
from microbit import *  # import microbit module to control micro:bit development board
```

```
ldr_R = 0   # initialize the right photoresistor value
ldr_L = 0   # initialize the left photoresistor value

while True:   # loop. print the values all the time
    ldr_L = pin1.read_analog()   # read the analog value of the photoresistor connected to pin1, and assign to ldr_L
    ldr_R = pin0.read_analog()   # read the analog value of the photoresistor connected to pin0, and assign to ldr_R
    print("ldr_L:", ldr_L, "ldr_R:", ldr_R)   # print the two photoresistor values
    sleep(500)   # delay 500ms
```

15.5 Test Result

Click "REPL" and press the reset button on the back of the micro:bit, and you will see the analog values of the photoresistors.



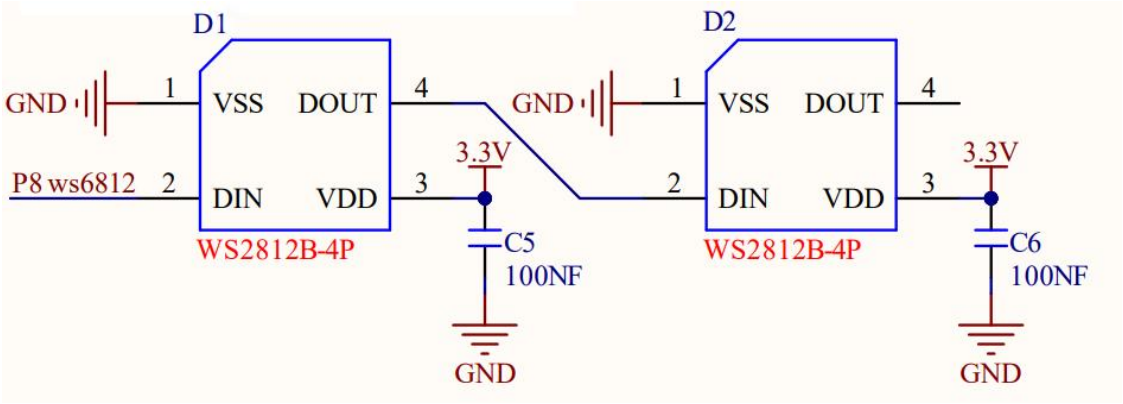
Project 16　WS2812 Pixel

16.1 Introduction

The WS2812 pixel integrates the control circuit and the RGB chip in the 5050 package to form a complete pixel control unit. Each RGB LED is with 256 brightness levels so they form 16,777,216 colors with a refresh rate of no less than 400Hz.

Besides, it supports cascade control to transmit signals through single wire series ports, and there is no additional circuits requirement for transmission within 5 meters. At a refresh rate of 30fps, it can control no less than 512 pixels at low-speed mode and more than 1024 pixels at high-speed mode.

16.2 Working Principle



WS2812 adopts single-bus communication protocol, and each pixel supports 24-bit color control (RGB888). Signals are input through DIN, and after passing one pixel, the first 24 bits of data will be latched, and the remaining data will be output from DOUT after shaping, which forms cascade control.

You can query the RGB three primary color table to adjust the RGB value to display the corresponding color.

16.3 Test Code

Open "smartCarCodes" folder in "codes" folder, and open file "16-ws282.py

```
'''
Function description: ws2812 shows: red, green, blue, cyan, pink, yellow and off
Compiling IDE: MU 1.2.0
Program name: HolaSmart
'''
from microbit import *  # import microbit module to access hardware
import neopixel  # import neopixel library to control WS2812 LED

# the number of NeoPixel pixels connected to pin P8
NUM_PIXELS = 2
# create NeoPixel object, connect to pin P8
np = neopixel.NeoPixel(pin8 , NUM_PIXELS)

while True:
    #show red
    for num in range(0, 2) :      # traverse all connected LED
        np[num] = (255, 0, 0)    # Set the current LED to red
        np.show()    # Update the display status of the LED
    sleep(1000) # delay 1s
    #Show green
    for num in range(0, 2) :      # traverse all connected LED
        np[num] = (0, 255, 0)   # Set the current LED to green
```

```
        np.show()  #Update the display status of the LED
    sleep(1000)  # delay 1s
    # Similarly, set blue, cyan, pink, yellow, and off
    #Show blue
    for num in range(0, 2) :
        np[num] = (0, 0, 255)
        np.show()
    sleep(1000)
    #Show cyan
    for num in range(0, 2) :
        np[num] = (0, 255, 255)
        np.show()
    sleep(1000)
    #Show pink
    for num in range(0, 2) :
        np[num] = (255, 0, 255)
        np.show()
    sleep(1000)
    #Show yellow
    for num in range(0, 2) :
        np[num] = (255, 255, 0)
        np.show()
    sleep(1000)
    #turn off
    for num in range(0, 2) :
        np[num] = (0, 0, 0)
        np.show()
```

```
    sleep(1000)
```

16.4 Test Result

After uploading the code, the ws2812 LED at the bottom of the car will light up and take turns in red, orange, yellow, green, blue, cyan, pink, yellow, and off. Each maintains for 1 second.

Project 17　Line Tracking Sensor

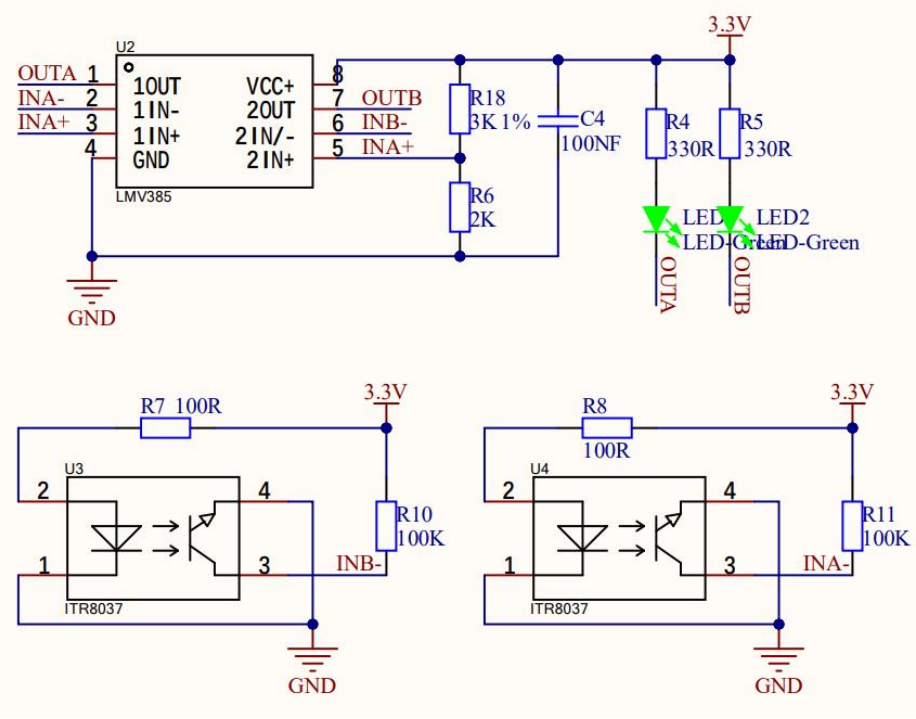
17.1 Introduction

Line tracking sensor detects and tracks a particular line by infrared/photoelectric technology to determine the presence/absence of a path by transmitting and receiving light. It is widely used in intelligent cars, autonomous vehicles and industrial robots.

Generally, the sensor is composed of a transmitter and a receiver. When the light encounters a black or dark surface, most of it will be absorbed and little is reflected (reflectivity is low). So the sensor will detect a change in signal, so as to determine whether the car deviates from the line. For accurate tracking, multiple sensors can be used simultaneously to form a detection array, providing more accurate motion control for mobile devices.

It features fast response speed, accurate positioning, simple structure and low cost, so it is widely applied to intelligent technology. With development, this sensor also improves towards higher intelligence and integration.

17.2 Working Principle



17.3 Code Explanation

val_L = pin12.read_digital() : read the digital signals of pin P13, and assign it to val_L

17.4 Test Code

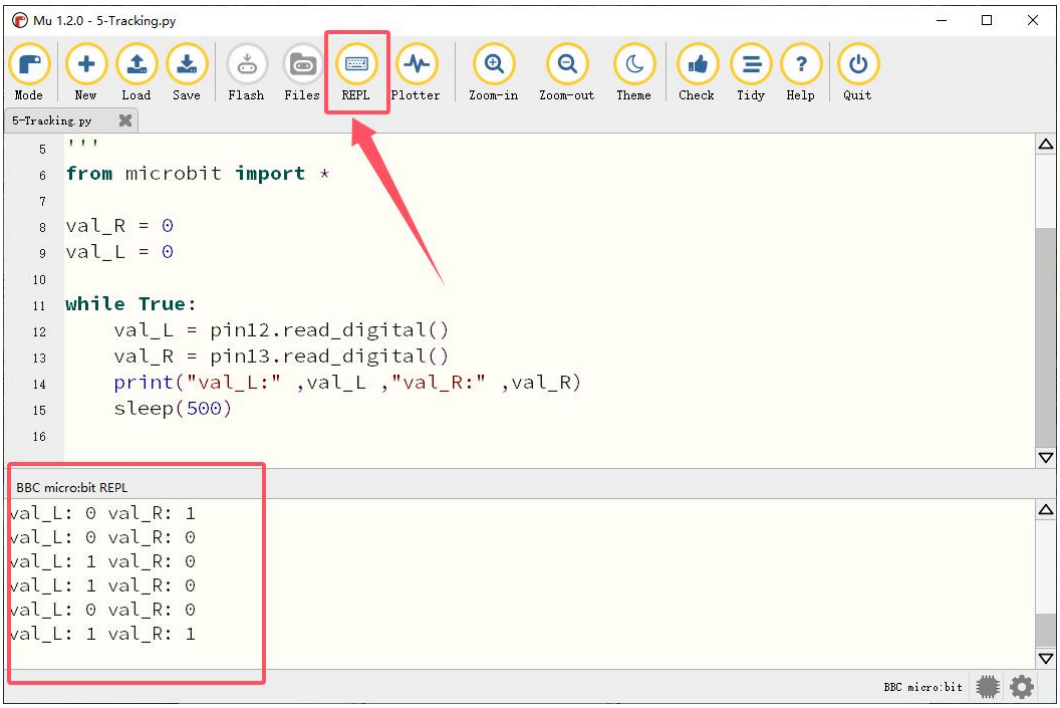
Open "smartCarCodes" folder in "codes" folder, and open file "17-Tracking.py"

```
'''
Function description: serial monitor prints the line tracking sensor values
Compiling IDE: MU 1.2.0
Program name: HolaSmart
'''
from microbit import *  # import microbit module to control micro:bit development board
```

```
val_R = 0     #initialize the right line tracking sensor value
val_L = 0     #initialize the left line tracking sensor value


while True:
    val_L = pin12.read_digital()      #read the digital value connected to pin pin12, and assign it to val_L
    val_R = pin13.read_digital()      #read the digital value connected to pin pin13, and assign it to val_R
    print("val_L:" ,val_L ,"val_R:" ,val_R) # print the two line tracking sensors' values on the serial monitor
    sleep(500)   # delay 500ms
```

17.5 Test Result

After uploading the code, click "REPL" and press the reset button on the back of the microbit, and the serial monitor will print the values of line tracking sensors. The results will be refreshed every 500ms.



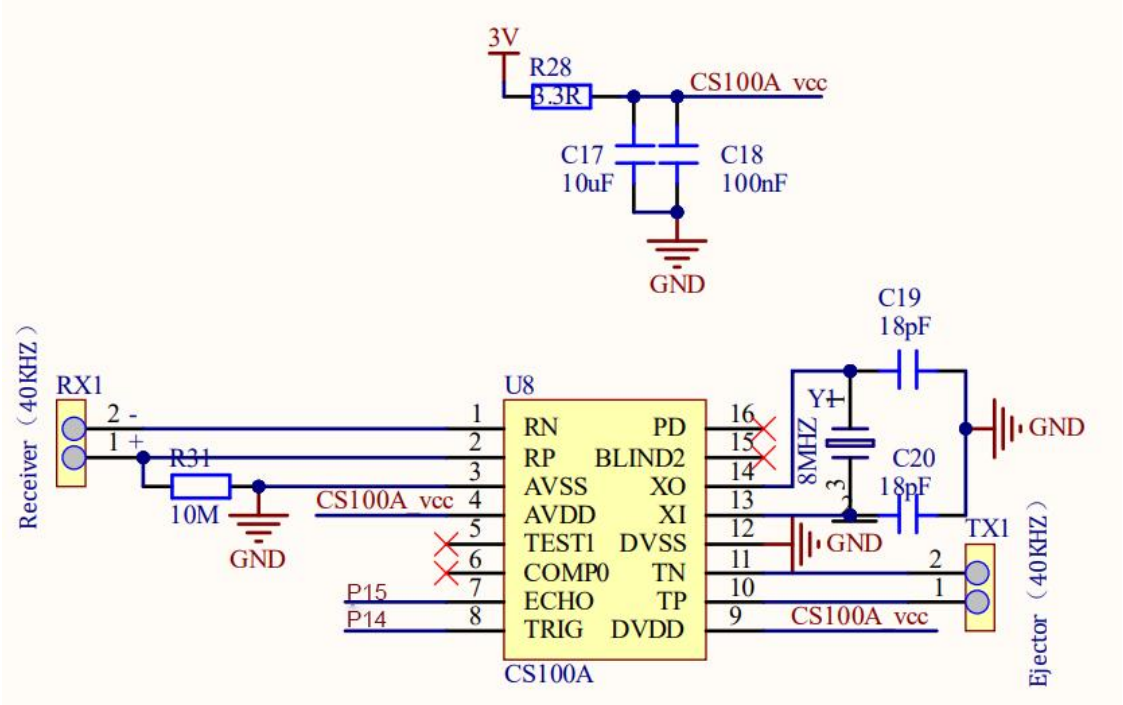
Project 18　Ultrasonic Sensor

18.1 Introduction

Ultrasonic sensor measures the distance of an object through sound waves.

Its transmitter emits sound waves, and its receiver receives the returned sound waves, so that the distance can be calculated according to the time difference between them. Beyond distance measurement, it can also be used to detect the shape of objects, build automatic doors, measure flow rates and pressures, detect the presence of objects, and so on.

18.2 Working Principle



Working Principle:

Like bats, the ultrasonic sensor sends an ultrasonic signal with a high frequency that human cannot hear. If these signals encounter obstacles,

they will immediately reflect back and be received by the sensor. After that, the distance between the sensor and the obstacle is calculated according to the time difference between signals transmitting and receiving.

Maximum detection distance: 3M

Minimum detection distance: 4cm

Detection angle: no greater than 15 degrees

18.3 Test Code

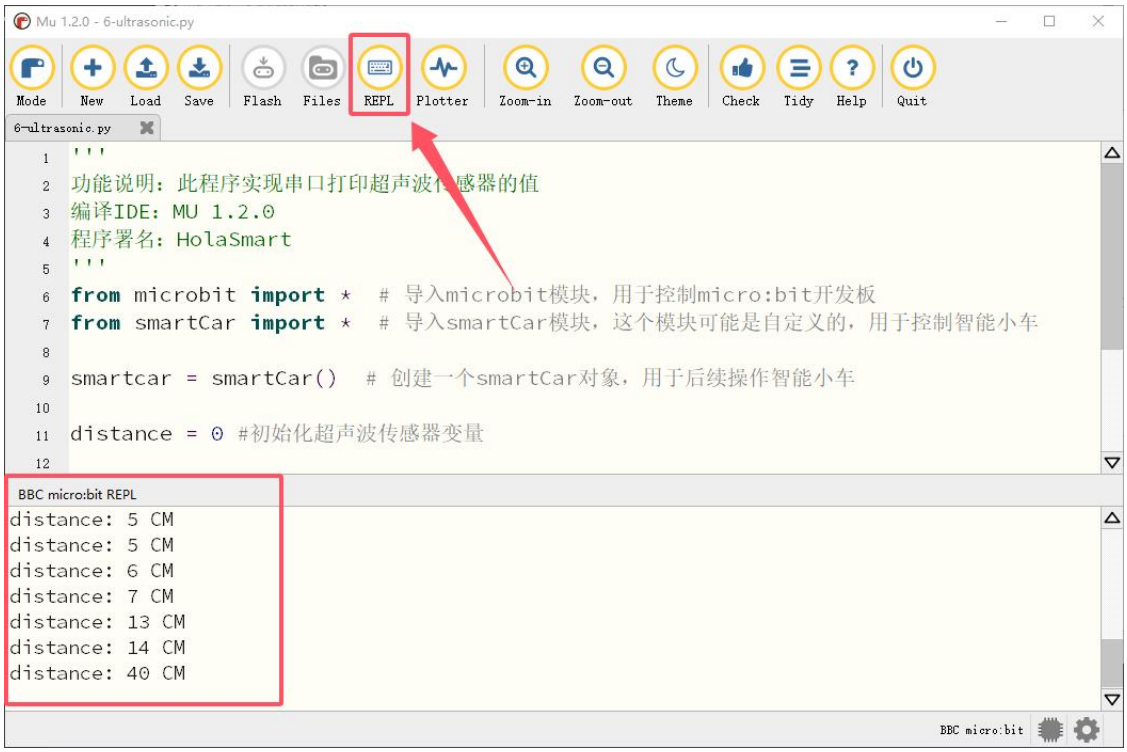
Open "smartCarCodes" folder in "codes" folder, and open file "18-ultrasonic.py"

```
'''
Function description: serial monitor prints ultrasonic sensor values
Compiling IDE: MU 1.2.0
Program name: HolaSmart
'''
from microbit import *  # import microbit module to control micro:bit development board
from smartCar import *  # import smartCar module. This module may be customized to control the smart car


smartcar = smartCar()  # create a smartCar object to operate the smart car


distance = 0 # initialize the variable distance value detected by the ultrasonic sensor


while True:
    distance = smartcar.get_distance()  # assign smartcar.get_distance() to distance
    print("distance:", distance ,"CM")          # print the distance value
```

```
sleep(1000)
```

18.4 Test Result

After upload code, click "REPL" and press the reset button on the back of the microbit, and the serial monitor prints the distance value detected by the ultrasonic sensor. The results will be refreshed every 500ms.



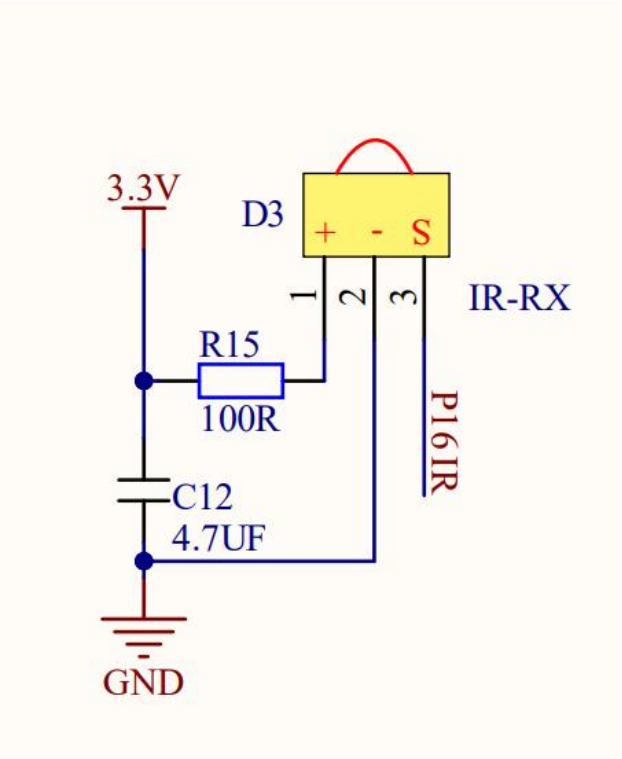
Project 19　IR Receiver

19.1 Introduction

Infrared remote control is so ubiquitous in daily life that it is hard to imagine what the world would be like without it. It is used to control various home appliances such as televisions, stereos, video recorders and satellite signal receivers.

Infrared remote control is composed of infrared emission and infrared receiving system. To put it simple, they are a remote control, an infrared receiver and a single chip computer that can decode.

19.2 Working Principle



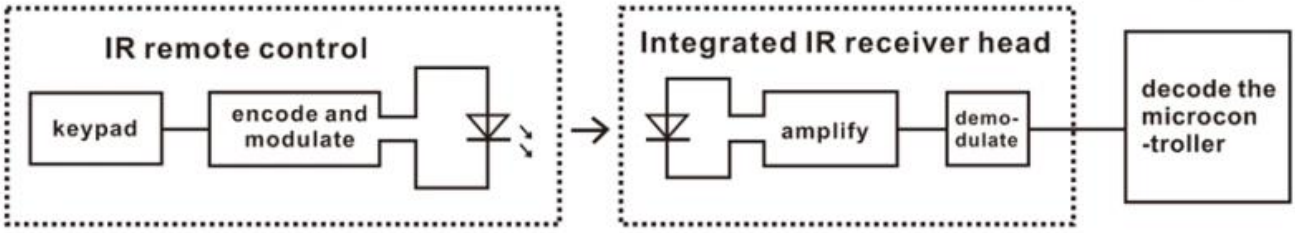
The coding chip in the remote control encodes the emitted 38K infrared carrier signal, which is composed of a guide code, user code, data code and data reverse code. These codes are generally all zeros and ones. It determines 0 or 1 through the time interval of the pulse (if the ratio of high and low levels is about 1:1, it is considered to be signal 0).

The user code of one remote control maintains unchanged, so it determines that which button is pressed by data codes. When a button is pressed, the remote control sends infrared carrier signals, which will be decoded after receiving by the infrared receiver.

The MCU decodes the received 0 and 1 signals to determine which button is pressed.

The ob-board infrared receiver is a device that combines receiving, amplifying and demodulation. Its internal IC has been demodulated to complete infrared receiving and outputting that are compatible with TTL level signal. It outputs digital signals.

It is suitable for infrared remote control and infrared data transmission.



The control pin of the IR receiver on the car is pin P16 of the micro:bit board.

19.3 Code Explanation

ir = smartcar.ir_data() : read the IR receiver value and assign it to variable ir

19.4 Test Code

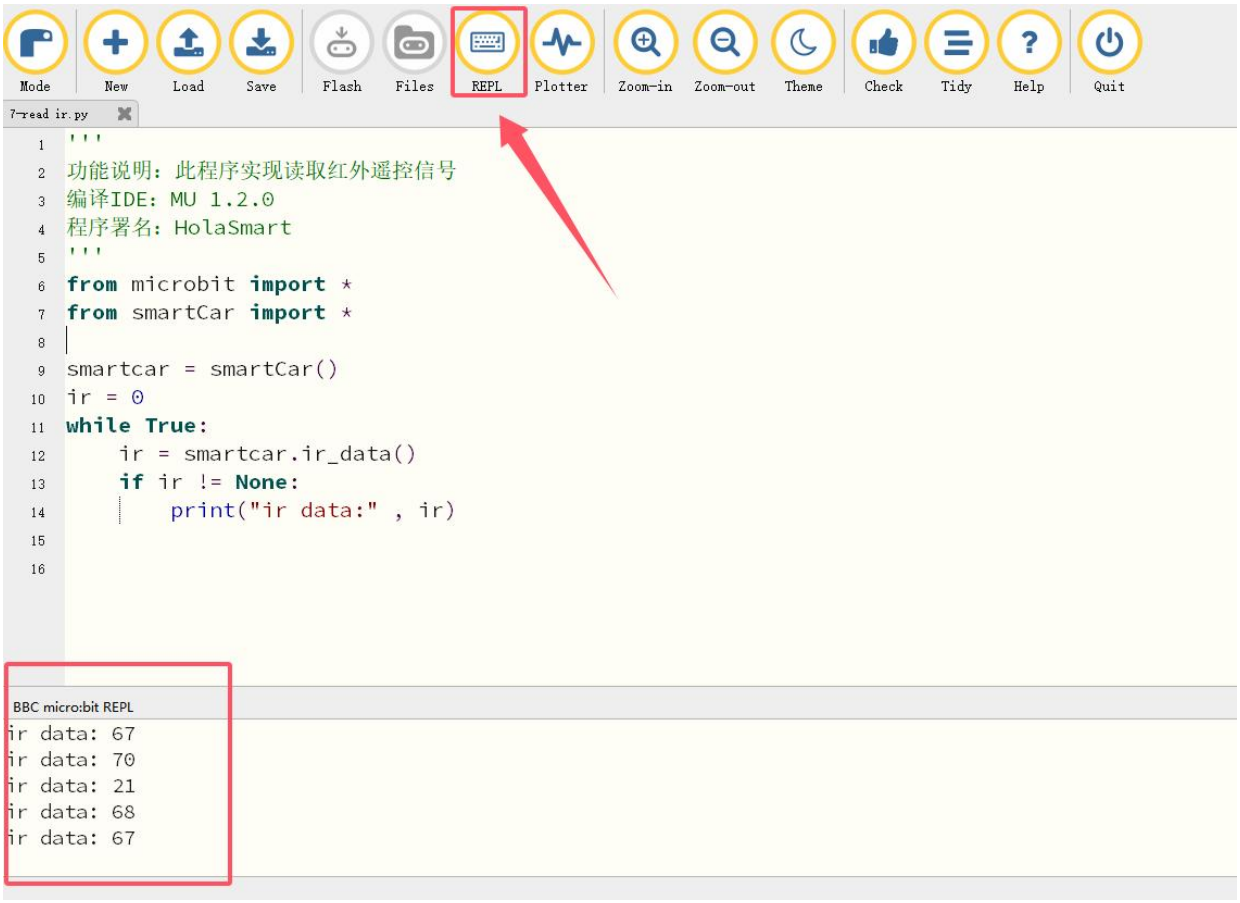
Open "smartCarCodes" folder in "codes" folder, and open file "19-read ir.py"

```
'''
Function description: read the IR receiver signals
Compiling IDE: MU 1.2.0
Program name: HolaSmart
'''
```

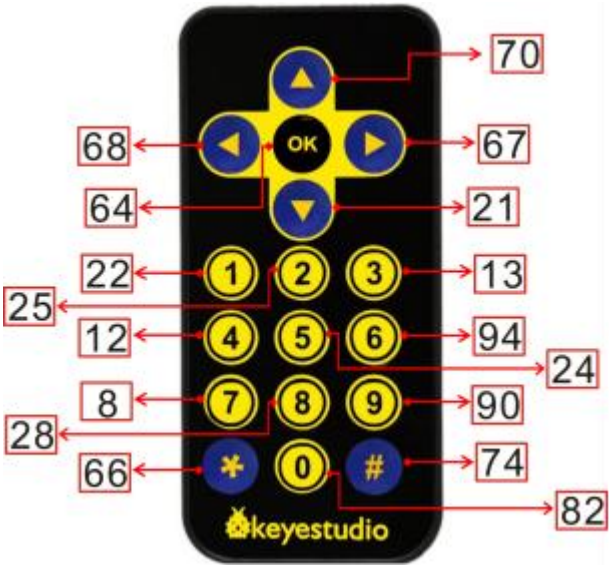
```
from microbit import *
from smartCar import *

smartcar = smartCar()

# initialize a variable ir to store the data read by the IR receiver on the car
ir = 0

# loop. Keep reading and processing IR receiver data
while True:
    # call smartcar example ir_data(): it returns the current data read by the IR receiver
    # assign the returned value to variable ir
    ir = smartcar.ir_data()

    # Check if the variable ir is not a None. This may be to ensure that valid data is read, b
ecause in some cases the sensor may be not able to read a single value.
    if ir != None:
        # If a valid infrared data is read, print it
        print("ir data:" , ir)
```

19.5 Test Result

After uploading the code, click "REPL" and press the reset button on the back of the microbit, and the serial monitor will print the button value received by the IR receiver. We add conditions in the code, so the monitor only outputs values when buttons on the remote control are pressed.



Remote control button values:



Project 20　Motor Drive

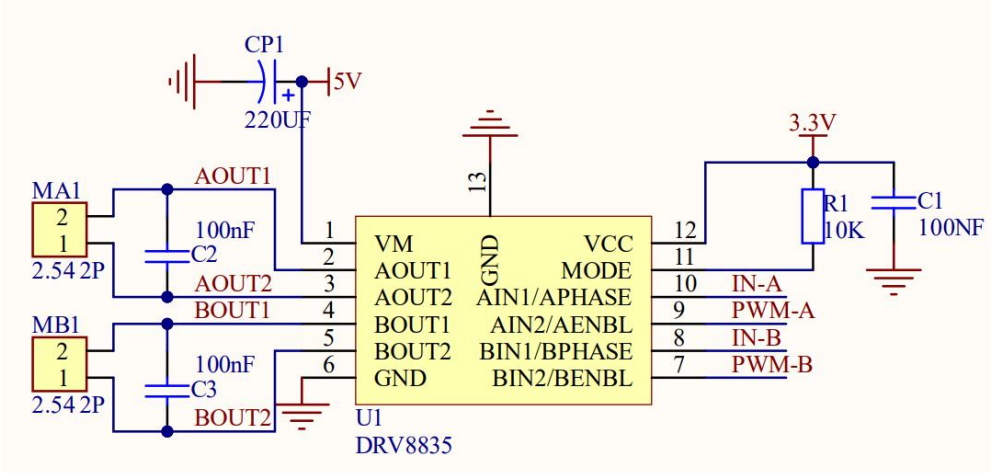
20.1 Introduction

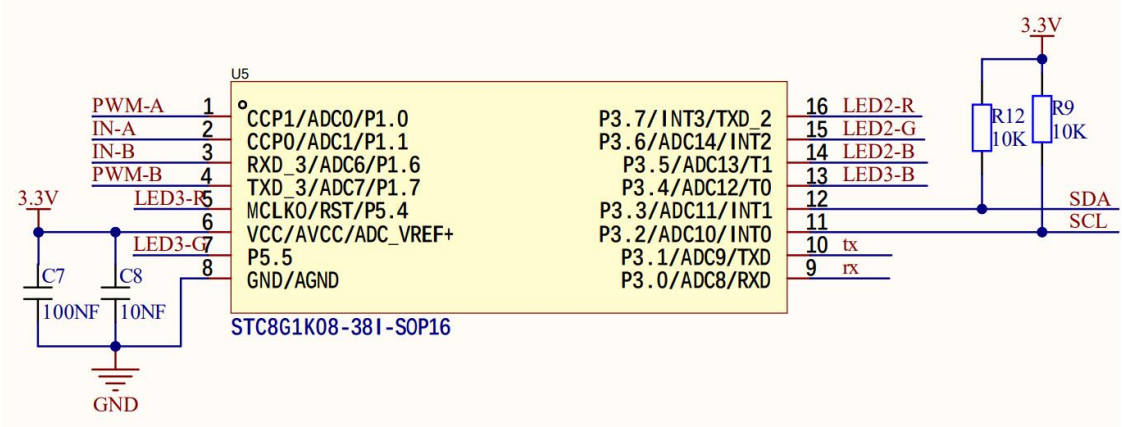
The expansion board of the car is equipped with two DC gear motors, which boast additional gear reducers compared to ordinary DC motors. These reducers provide lower speed and greater torque, and their different reduction ratios correspond to different speeds and torques. This has greatly increased the use of DC motors in the automation industry.

Gear Motor refers to the integration of a reducer and a motor, and it is with simple design and small volume. In application, it is widely used in steel and machinery industry.

20.2 Working Principle

Motors on the car expansion board are controlled by STC8G1K08 chip and DRV8835 motor control chip. To save IO ports, instructions are sent to the STC8G1K08 chip through the micro:bit IIC to enable the DRV8835 motor driver chip to control the rotation direction and speed of the two DC gear motors.





20.3 Code Explanation

Motor control logic table:

A - left motor	B - right motor	car state
smartcar.Motor_L(1, 200)	smartcar.Motor_R(1, 200)	Go forward at a speed value of 200
smartcar.Motor_L(0, 200)	smartcar.Motor_R(1, 200)	Go back at a speed value of 200
smartcar.Motor_L(0, 200)	smartcar.Motor_R(1, 200)	Turn left at a speed value of 200
smartcar.Motor_L(1, 200)	smartcar.Motor_R(1, 200)	Turn right at a speed value of 200
smartcar.Motor_L(1, 0)	smartcar.Motor_R(1, 0)	Stop

20.4 Test Code

Open "smartCarCodes" folder in "codes" folder, and open file "20-motor.py"

```
'''
Function description: motor drive
```

```
Compiling IDE: MU 1.2.0
Program name: HolaSmart
'''
from microbit import *
from smartCar import *

smartcar = smartCar()

while True:
    #Forward
    smartcar.Motor_L(1, 200)
    smartcar.Motor_R(1, 200)
    sleep(2000)
    #back
    smartcar.Motor_L(0, 200)
    smartcar.Motor_R(0, 200)
    sleep(2000)
    #turn left
    smartcar.Motor_L(0, 200)
    smartcar.Motor_R(1, 200)
    sleep(2000)
    #turn right
    smartcar.Motor_L(1, 200)
    smartcar.Motor_R(0, 200)
    sleep(2000)
```

20.5 Test Result

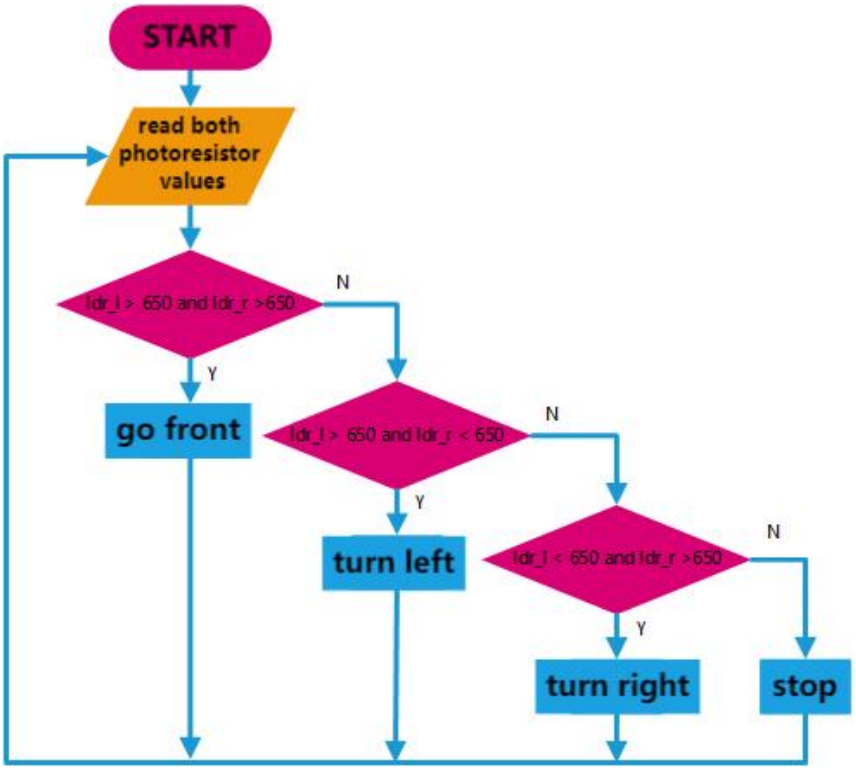
After uploading the code, the car goes forward, goes back, turns left, turns right and then stop, with each action lasts for 2s.

Project 21　Light Following Car

21.1 Introduction

How to make a light following car? We compare the analog values of the two photoresistors on both side of the car, and then program to control the car to go with the brighter side. If the two values are almost equal, the car will go forward. But note that please do this experiment in a relatively dark environment, and turn on your flashlight.

21.2 Code Flow



21.3 Code Explanation

if ldr_l > 650 and ldr_r > 650: determine whether both the left and right photoresistor value is greater than 650 , if the two conditions are satisfied, it outputs true.

21.4 Test Code

Open "smartCarCodes" folder in "codes" folder, and open file "21-Light seeking car.py"

```
'''
Function description: car follows light
Compiling IDE: MU 1.2.0
Program name: HolaSmart
'''
from microbit import *
from smartCar import *

smartcar = smartCar()
ldr_l = 0
ldr_r = 0

while True:
    # read the two photoresistors values
    ldr_l = pin1.read_analog()
    ldr_r = pin0.read_analog()
    #determine whether both ldr_l and ldr_r are both greater than 650. If yes, car goes forward
    if ldr_l > 650 and ldr_r > 650:
```

```
            smartcar.Motor_L(1, 150)
            smartcar.Motor_R(1, 150)
        #determine whether ldr_l  is greater than 650 and ldr_r is less than 650. if yes, car turns l
eft
        elif ldr_l > 650 and ldr_r < 650:
            smartcar.Motor_L(0, 200)
            smartcar.Motor_R(1, 200)
        #determine whether ldr_l is less than 650 and ldr_r is greater than 650. if yes, car turns ri
ght
        elif ldr_l < 650 and ldr_r > 650:
            smartcar.Motor_L(1, 200)
            smartcar.Motor_R(0, 200)
        #if all above conditions are not satisfied, car stops
        else:
            smartcar.Motor_L(0, 0)
            smartcar.Motor_R(1, 0)
```

21.5 Test Result

After uploading the code, put the car in a relatively dark environment. Turn on your flashlight and the car will follow the light to move.

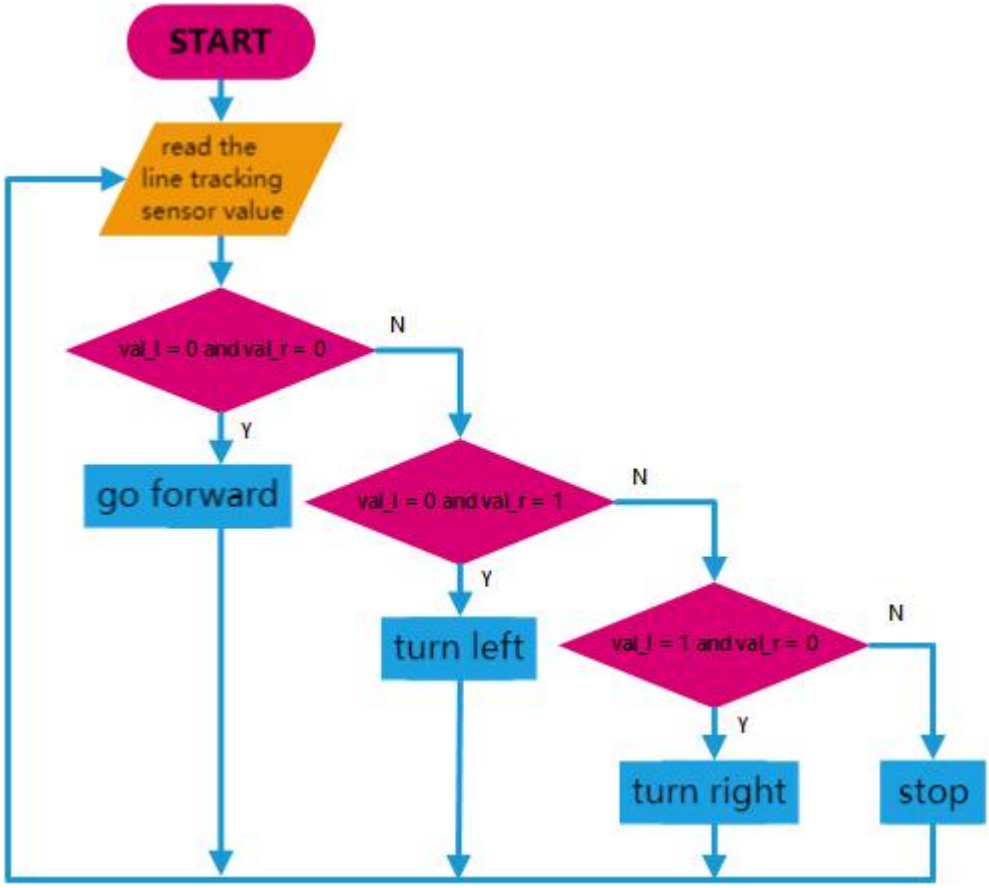
If you want the car to turn left, focus the light on the left side of its head. Similarly, focus the light on the right and it turns right. Focus the light on the front and it goes forward.

Project 22　Line Tracking Car

22.1 Introduction

In this project, the car adopts line tracking sensor to determine whether it is black or white under it. If it is white, the car turns to the black side. If it is always black, the car keeps going. In the kit, we provide you with a map with black line.

22.2 Code Flow



22.3 Test Code

Open "smartCarCodes" folder in "codes" folder, and open file "22-Track car.py

```
'''
Function description: car follows black line
Compiling IDE: MU 1.2.0
Program name: HolaSmart
'''
from microbit import *
from smartCar import *

smartcar = smartCar()
val_l = 0
val_r = 0

while True:
    # read the line tracking sensor values
    val_l = pin12.read_digital()
    val_r = pin13.read_digital()
    #determine whether both val_l and val_r equal 0. if yes, car goes forward
    if val_l == 0 and val_r == 0:
        smartcar.Motor_L(1, 150)
        smartcar.Motor_R(1, 150)
    #determine whether val_l = 0 and val_r = 1. if yes, car turns left
    elif val_l == 0 and val_r == 1:
        smartcar.Motor_L(0, 200)
```

```
        smartcar.Motor_R(1, 200)
    #determine whether val_l = 1 and val_r = 0. if yes, car turns right
    elif val_l == 1 and val_r == 0:
        smartcar.Motor_L(1, 200)
        smartcar.Motor_R(0, 200)
    # if none of the above conditions are satisfied, the car stops
    else:
        smartcar.Motor_L(0, 0)
        smartcar.Motor_R(1, 0)
```

22.4 Test Result

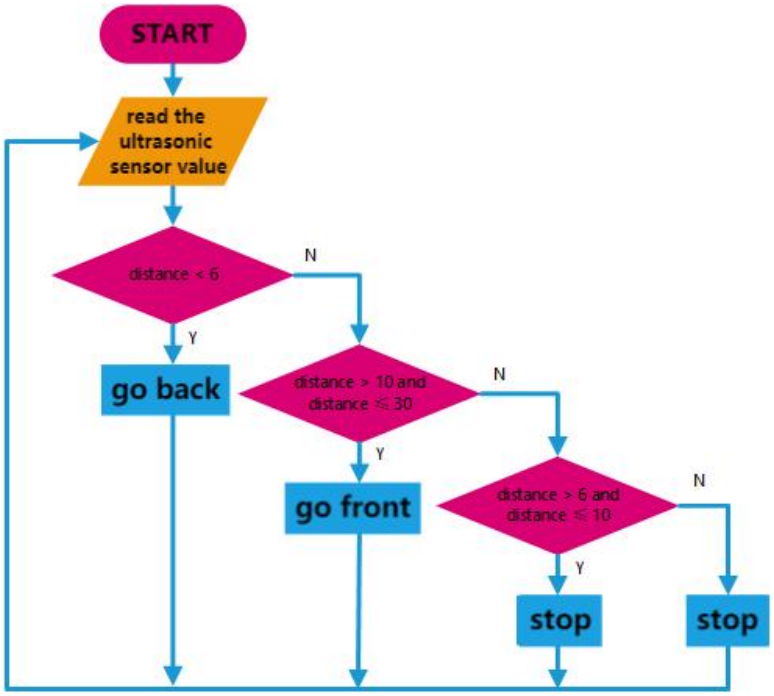
After uploading the code, unfold the map in the kit, and put the car on the map. You will see that the car follows the black line.

Project 23 Ultrasonic Object Following Car

23.1 Introduction

We make an object following car by an ultrasonic sensor in this project. If the distance between the car and the obstacle in front is within a certain range, the car moves forward and follows; if the they are too close to a certain range, the car moves back.

23.2 Code Flow



23.3 Test Code

Open "smartCarCodes" folder in "codes" folder, and open file "23-Ultrasonic Follow.py"

```
'''
Function description: car uses ultrasonic sensor to follow object
Compiling IDE: MU 1.2.0
Program name: HolaSmart
'''
from microbit import *
from smartCar import *

smartcar = smartCar()
distance = 0
```

```
while True:
    # read the ultrasonic sensor value
    distance = smartcar.get_distance()
    # determine whether distance is nearer than 6. if yes, car goes back
    if distance < 6:
        smartcar.Motor_L(0, 150)
        smartcar.Motor_R(0, 150)、
    #determine whether distance is within 10 to 30 (can equal to 30). if yes, car goes forward
    elif distance > 10 and distance <= 30:
        smartcar.Motor_L(1, 150)
        smartcar.Motor_R(1, 150)
    #determine whether distance is within 6 to 10 (can equal to 10). if yes, car stops
    elif distance > 6 and distance <= 10:
        smartcar.Motor_L(1, 0)
        smartcar.Motor_R(0, 0)
    #if none of above conditions are satisfied, car stops
    else:
        smartcar.Motor_L(0, 0)
        smartcar.Motor_R(1, 0)
```

23.4 Test Result

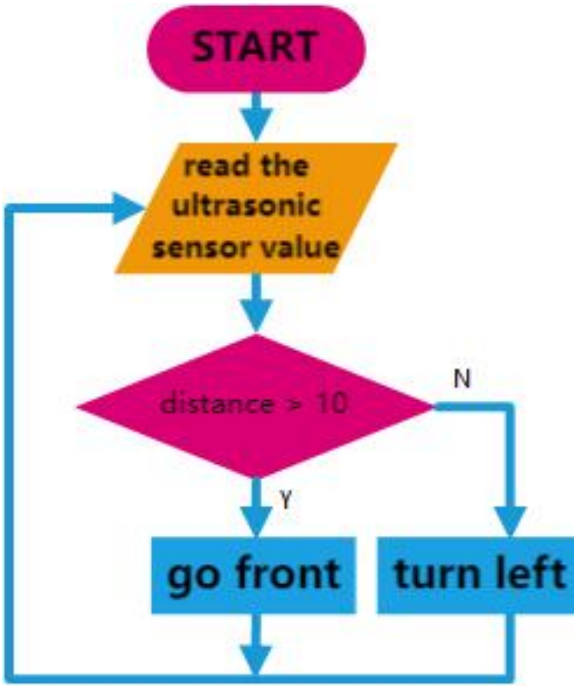
After uploading the code, put your hand in front of the car. When the distance between the car and your hand is less than 6CM, the car goes back. When the distance is within 10CM and 30CM, the car goes forward. If 6CM < distance ≤ 10CM, the car stops. If none of the conditions are satisfied, the car still stops.

Project 24　Ultrasonic Obstacle Avoidance Car

24.1 Introduction

In this project, an ultrasonic sensor is adopted to detect the distance value between the car and obstacles, so that the car can turn left to avoid them if the value is less than the set value.

24.2 Code Flow



24.3 Test Code

Open "smartCarCodes" folder in "codes" folder, and open file "24-Ultrasonic obstacle avoidance.py"

```
'''
Function description: the car uses ultrasonic sensor to avoid obstacles
Compiling IDE: MU 1.2.0
Program name: HolaSmart
```

```
'''
from microbit import *
from smartCar import *

smartcar = smartCar()
distance = 0

while True:
    #read the ultrasonic sensor value
    distance = smartcar.get_distance()
    #determine whether distance is farther than 10. if yes, car goes forward
    if distance > 10:
        smartcar.Motor_L(1, 150)
        smartcar.Motor_R(1, 150)
    #if the above condition is not satisfied, car turns left for 500ms
    else:
        smartcar.Motor_L(0, 150)
        smartcar.Motor_R(1, 150)
        sleep(500)
```

24.4 Test Result

After uploading code, the ultrasonic sensor detects distance between the car and obstacles. When the value is greater than 10CM, the car goes forward. When it is less than 10CM, the car turns left for 500ms.

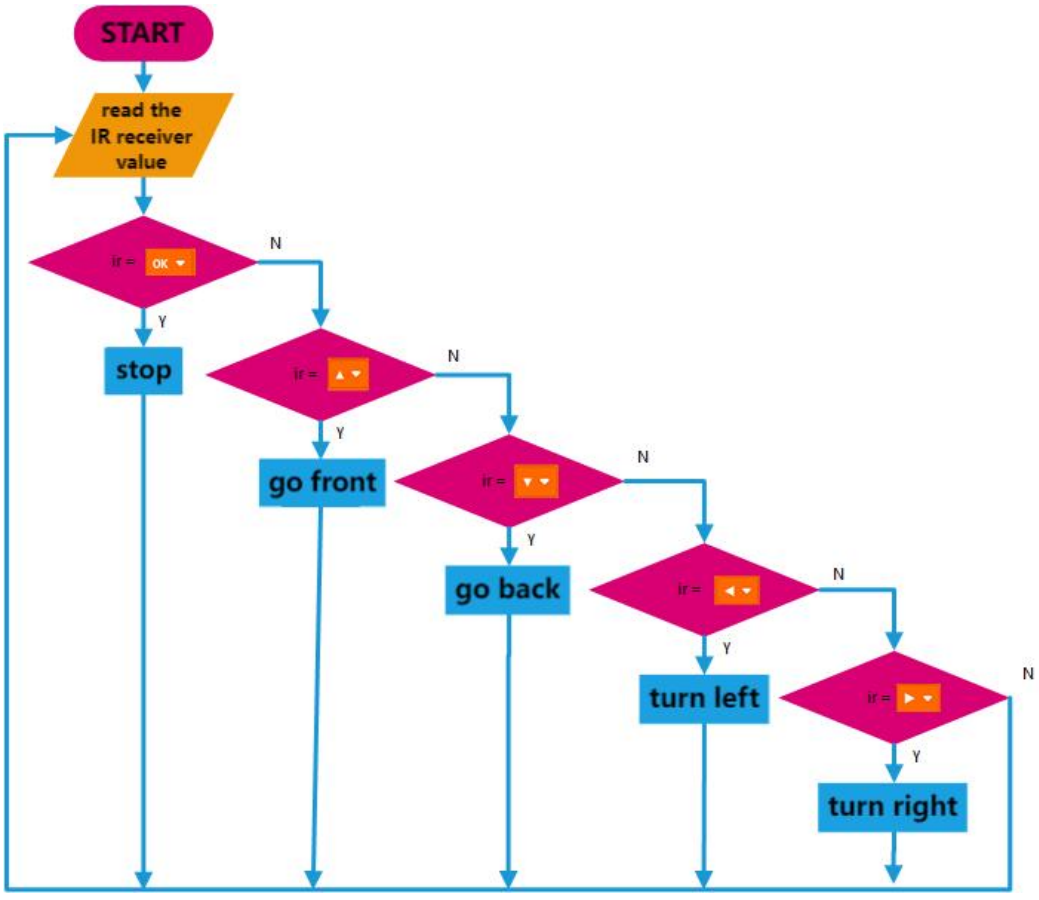
Project 25　IR Remote Control Car

25.1 Introduction

IR remote control car is controlled by infrared signals. A remote control are included in this kit to emit infrared signals, and n infrared receiver is designed on the car to receive signals to direct the movement of the car. During controlling, you only need to press the button on the remote control to make the car go forward, backward, turn or stop.

Besides, it is simple and easy to control, so it is suitable for users of all ages to experience interesting and interactive pleasure.

25.2 Code Flow



25.3 Test Code

Open "smartCarCodes" folder in "codes" folder, and open file "25-ir remote control car.py"

```
'''
Function description: read the IR remote control button values to control the car
Compiling IDE: MU 1.2.0
Program name: HolaSmart
'''
from microbit import *
from smartCar import *

smartcar = smartCar()
ir = 0
while True:
    #read the IR receiver values
    ir = smartcar.ir_data()
    #determine whether ir equals 70. if yes, car goes forward
    if ir == 70:
        smartcar.Motor_L(1, 200)
        smartcar.Motor_R(1, 200)
    #determine whether ir equals 21. if yes, car goes back
    elif ir == 21:
        smartcar.Motor_L(0, 200)
        smartcar.Motor_R(0, 200)
    #determine whether ir equals 68. if yes, car turns left
```

```
    elif ir == 68:
        smartcar.Motor_L(0, 200)
        smartcar.Motor_R(1, 200)
    #determine whether ir equals 67. if yes, car turns right
    elif ir == 67:
        smartcar.Motor_L(1, 200)
        smartcar.Motor_R(0, 200)
    #determine whether ir equals 64. if yes, car stops
    elif ir == 64:
        smartcar.Motor_L(1, 0)
        smartcar.Motor_R(1, 0)
```

25.4 Test Result

After uploading the code, you can control the car with the remote control in the kit.

Press  and the car goes forward; Press  and the car goes back; Press  and the car turns left, Press  and the car turns right; Press  and the car stops.

Troubleshooting

1. After uploading the code, micro:bit on-board dot matrix shows a crying face and then scrolls to display the error messages.

   If you upload the code of Mciro:bit basic projects, please check whether any characters or signs are accidentally added or deleted. Click 

   to check. Yet some warnings may show up after checking, which are not errors.

   If you upload the code of the smart car projects, please check whether the micro:bit library is imported. For how, please refer to "5.Import Library to Mciro:bit on MU". Or again, check whether any characters or signs are accidentally added or deleted (Re-download the code for the upload).

2. When the battery voltage is below 3.7V, the required current will be too large to lower the voltage during the change of direction of motor rotation, so as to cause the stc8G1K08 chip to be reset.

3. After uploading the code, there is no message on the serial monitor after you click 

   . Please remember to press the on-board reset button at the back of Micro:bit, as shown below:


FCC ID:2AKFPMB0200
USB
BLE ANTENNA
MICROPHONE
RESET / ⏻
BATTERY
3V
SPEAKER
PROCESSOR
ACCELEROMETER
COMPASS
PINS
BBC
micro:bit
V2
00