6.Raspberry Pi platform ------- tracking

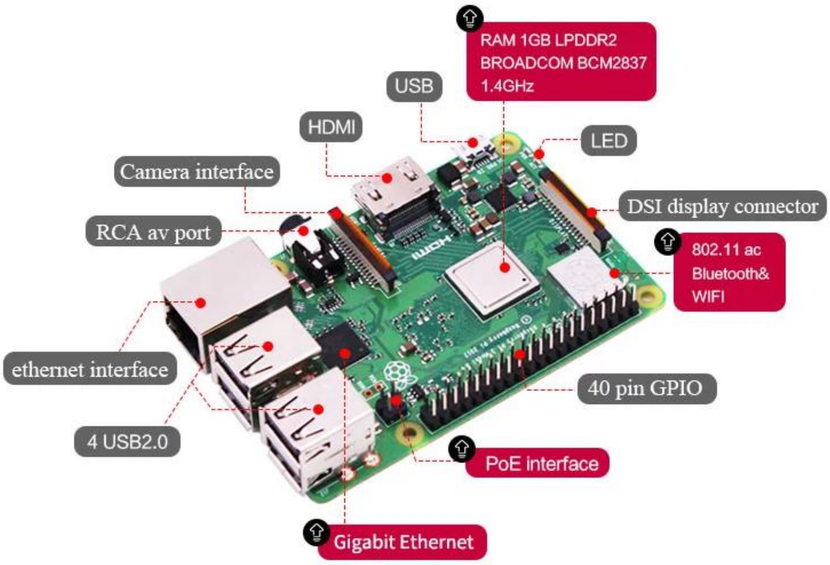
Note:

A.Before this experiment, we can adjust the sensitivity of the tracking module by rotating the potentiometer of the infrared tracking module to achieve better experimental results. For more detailed, please read the manual [Function debugging].

B. This experiment needs to be done indoors to reduce the interference of sunlight on the infrared receiver.

C. This experiment needs to start the car by pressing the button KEY.

1)Preparation



1-1　Raspberry Pi board

1-2　Infrared patrol module

2)Purpose of Experimental

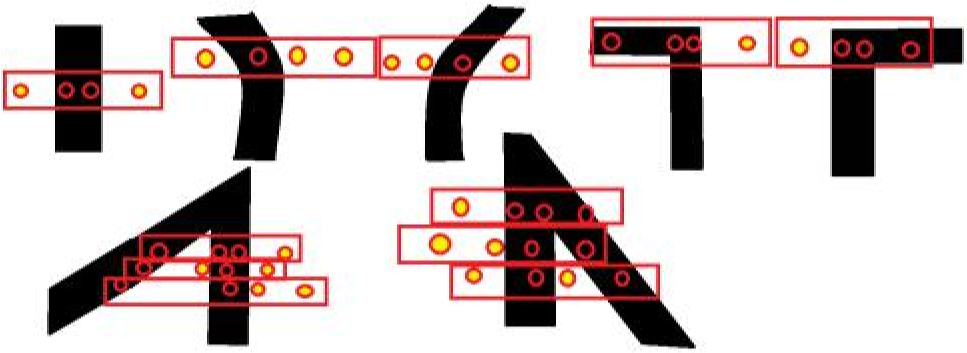
After running the tracking executable in the Raspberry Pi system, you need to press the K2 to start the car, and the tracking function is started. The robot car will automatically walk along the black line.

3)Principle of experimental

The basic principle of the infrared tracking sensor is to take advantage of the reflective nature of the object. In this experiment, we need the effect that the robot car walk along the black line. When the infrared light is emitted onto the black line, it will be absorbed by the black line, but when the infrared light is emitted onto the other colors line, it will be reflected onto the infrared receiver pin. According to this, we write the corresponding code to make the car complete tracking function.

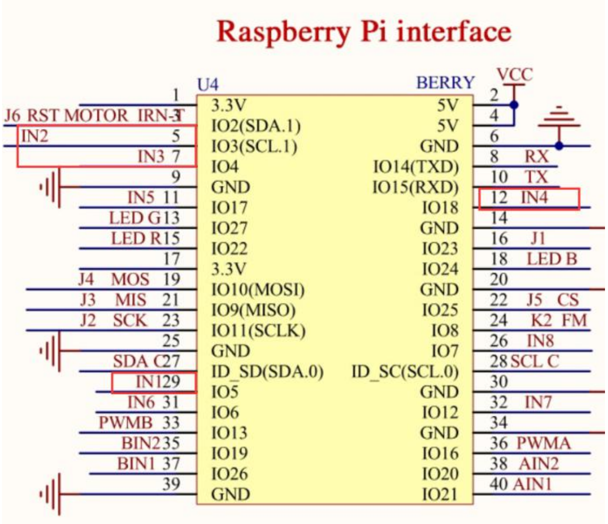
When the car detects the black line, the indicator status of the infrared tracking module is as shown in the figure below.

(Note:The TrikeBot car controls the direction of the front rubber wheel by controlling the servo. In this way, the direction of the car is controlled. In this experimental, we set the angle of the servo to 55 degrees according to the actual situation, indicating that the car is in the front. We set the angle of the servo to 110 degrees according to the actual situation, indicating that the car is turning left. We set the angle of the servo to 0 degrees according to the actual situation, indicating that the car is turning right.)

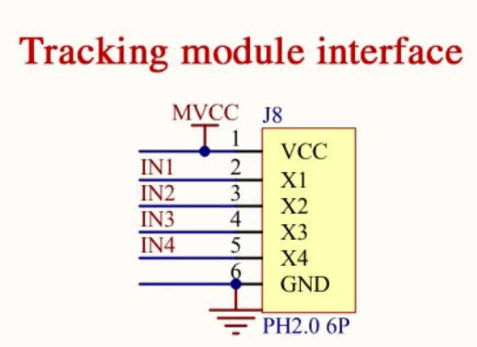


4)Experimental Steps

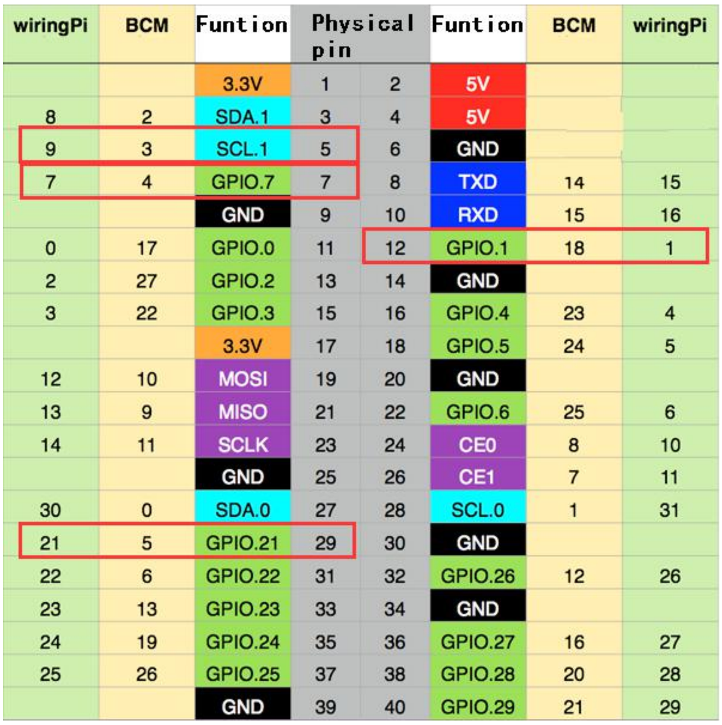
4-1 About the schematic



4-1 Raspberry Pi interface circuit diagram



4-2 Tracking module interface



wiringPi	BCM	Funtion	Physical pin		Funtion	BCM	wiringPi
		3.3V	1	2	5V		
8	2	SDA.1	3	4	5V		
9	3	SCL.1	5	6	GND		
7	4	GPIO.7	7	8	TXD	14	15
		GND	9	10	RXD	15	16
0	17	GPIO.0	11	12	GPIO.1	18	1
2	27	GPIO.2	13	14	GND		
3	22	GPIO.3	15	16	GPIO.4	23	4
		3.3V	17	18	GPIO.5	24	5
12	10	MOSI	19	20	GND		
13	9	MISO	21	22	GPIO.6	25	6
14	11	SCLK	23	24	CE0	8	10
		GND	25	26	CE1	7	11
30	0	SDA.0	27	28	SCL.0	1	31
21	5	GPIO.21	29	30	GND		
22	6	GPIO.22	31	32	GPIO.26	12	26
23	13	GPIO.23	33	34	GND		
24	19	GPIO.24	35	36	GPIO.27	16	27
25	26	GPIO.25	37	38	GPIO.28	20	28
		GND	39	40	GPIO.29	21	29

4-3 Raspberry Pi 40 pins comparison table

4-2 According to the circuit schematic:

Left1 infrared sensor-----5(Physical pin)----- 9(wiringPi)
Left2 infrared sensor-----29(Physical pin)----- 21(wiringPi)
Right1 infrared sensor-----7(Physical pin)----- 7(wiringPi)
Right2 infrared sensor-----12(Physical pin)----- 1(wiringPi)

(Note: In this experiment, we can adjust the sensitivity of the tracking module by rotating the potentiometer of the infrared tracking module to achieve better experimental results.)

4-3 About the code

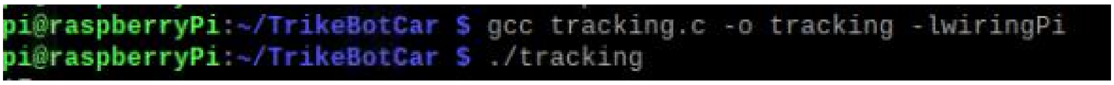
Please view .py and.c file

A. For .c code

(1) We need to compile this file in the Raspberry Pi system. (Note: we need to add -lwiringPi to the library file.)

We need to input:gcc tracking.c -o tracking -lwiringPi

(2) We need to run the compiled executable file in the Raspberry Pi system.We need to input: ./tracking



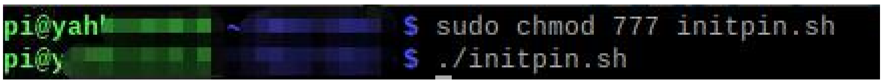
```
pi@raspberryPi:~/TrikeBotCar $ gcc tracking.c -o tracking -lwiringPi
pi@raspberryPi:~/TrikeBotCar $ ./tracking
```

(3)We can input: ctrl+c to stop this process, which mean is send a signal to the linux kernel to terminate the current process, but the state of the relevant pin is uncertain at this time, we also need to run a script to initialize all pins.

(Note:The initpin.sh script file is included in the TrikeBotCar/python_code directory.)

You need to input: chmod 777 initpin.sh
./initpin.sh

As shown in the figure below.

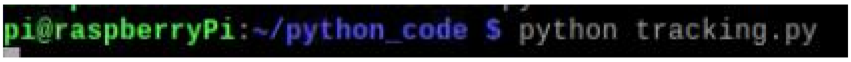


```
pi@yah[redacted] $ sudo chmod 777 initpin.sh
pi@y[redacted] $ ./initpin.sh
```

B. For python code

1) We need to input following command to run python code.

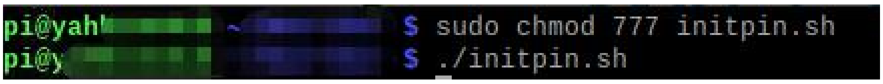
python tracking.py



```
pi@raspberryPi:~/python_code $ python tracking.py
```

2) We can input: ctrl+c to stop this process, which mean is send a signal to the linux kernel to terminate the current process, but the state of the relevant pin is uncertain at this time, we also need to run a script to initialize all pins.

3) You need to input: sudo chmod 777 initpin.sh
./initpin.sh

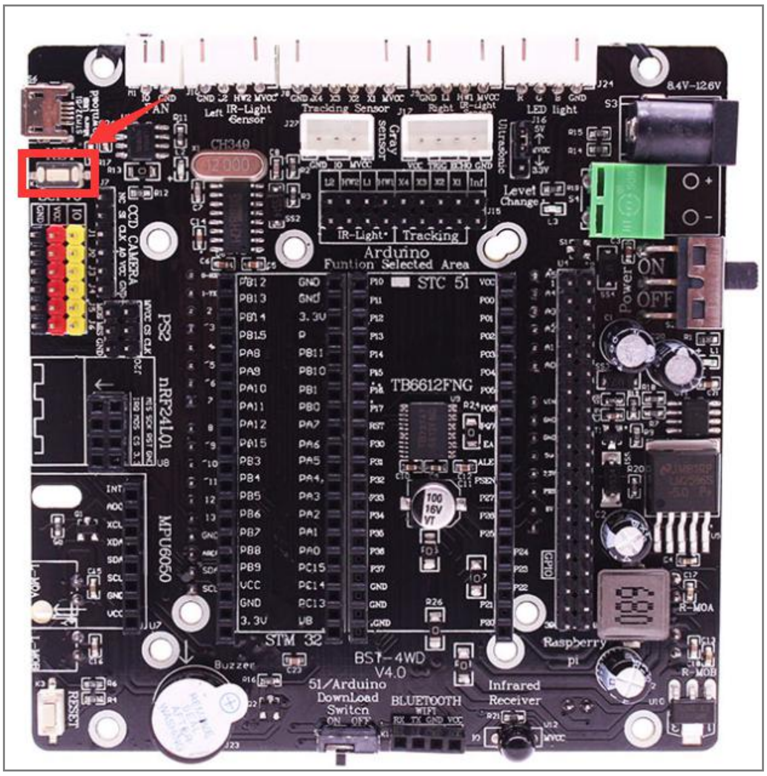


```
pi@yah[redacted] $ sudo chmod 777 initpin.sh
pi@y[redacted] $ ./initpin.sh
```

After completing the above steps, the experiment is over.

5) Experimental phenomenon

After running the programs. **You need to press the K2 to start the car,** and the tracking function is started. The robot car will automatically walk along the black line.


IR-Light Sensor
Tracking Sensor
Gray Sensor
Ultrasonic
LED light
8.4V-12.6V
Level Change
CH349
Arduino Funtion Selected Area
IR-Light
Tracking
STC 51
TB6612FNG
CCD CAMERA
PS2
nRF24L01
MPU6050
STM 32
Power
ON
OFF
Raspberry pi
GPIO
BST-4WD V4.0
Buzzer
51/Arduino DownLoad Switch
ON OFF
BLUETOOTH WIFI
RX TX GND VCC
Infrared Receiver
RESET
L-MOA
L-MOB
R-MOA
R-MOB