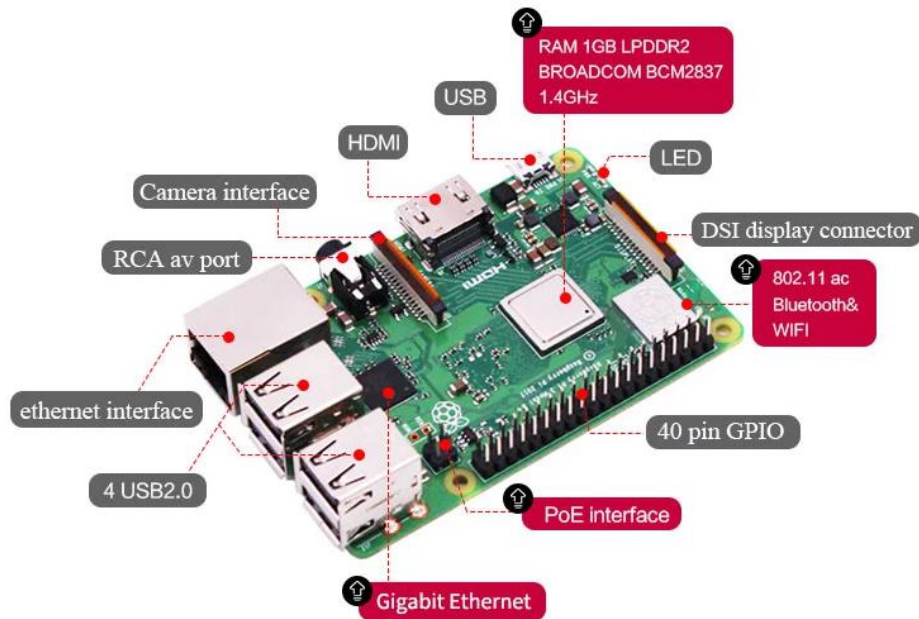
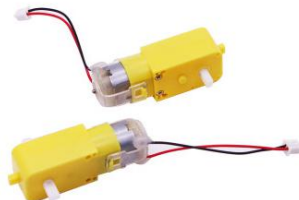


4. Raspberry Pi platform-----direction_control

1) Preparation



1-1 Raspberry Pi board



1-2 4 motors



1-3 9G metal servo

2) Purpose of Experimental

After running the direction_control executable in the Raspberry Pi system. After the car will delay 0.5s, advance 1s, back 1s, turn left 1s, turn right 1s, turn left in place, turn right in plac, stop 2s.

3) Principle of experimental

We use the TB6612FNG driver chip to drive the motor. Control the forward, reverse, and stop of the motor by controlling the level of AIN1, AIN2, BIN1, BIN2, PWMA, and PWMB of the driver chip.

In this experiment, we control robot car advance by controlling AIN1 to be high level, AIN2 is low level, BIN1 is high level, BIN2 is low level. And we control speed of the robot car by controlling PWMA, PWMB(0-255). One-channel PWM control the speed of the motor on one side of the robot car.

The working principle of the servo: the control signal enters the signal modulation chip from the channel of the receiver to obtain the bias voltage of the DC. It has a reference circuit inside, which generates a reference signal with a period of 20ms and a width of 1.5ms. It will compares the DC bias voltage with the voltage of the potentiometer to obtain a voltage difference and output. The positive and negative of the voltage difference is outputted to the motor drive chip to determine the forward and reverse of the motor.

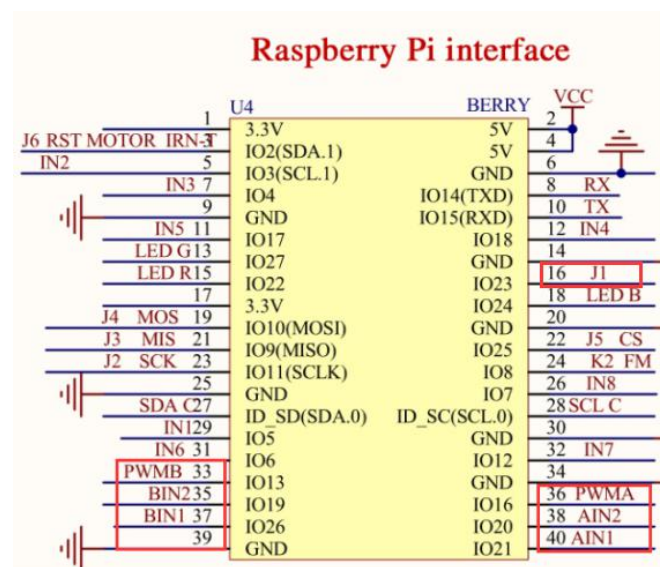
Servo rotation angle is by adjusting the duty ratios of PWM (pulse width modulation) signal. The standard PWM (pulse width modulation) signal has a fixed period of 20ms (50Hz). Theoretically, pulse width distribution should be between 1 ms to 2 ms, but in fact between pulse width can be 0.5 ms and 2.5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

| | |
|------------|------|
| 0.5ms----- | 0° |
| 1.0ms----- | 45° |
| 1.5ms----- | 90° |
| 2.0ms----- | 135° |
| 2.5ms----- | 180° |

(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

| 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:

AIN1-----40(Physical pin)----29(wiringPi)
 AIN2-----38(Physical pin)----28(wiringPi)
 PWMA-----36(Physical pin)----27(wiringPi)
 BIN1-----37(Physical pin)----25(wiringPi)
 BIN2-----35(Physical pin)----24(wiringPi)
 PWMB-----33(Physical pin)----23(wiringPi)
 J1---16(Physical pin)-----4(wiringPi)

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 direction_control.c -o direction_control -lwiringPi`

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

```
pi@raspberrypi:~/TrikeBotCar $ gcc direction_control.c -o direction_control -lwiringPi
pi@raspberrypi:~/TrikeBotCar $ ./direction_control
```

(3) We can input: **ctrl+c** to stop this process, which means 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@yahb ~ $ sudo chmod 777 initpin.sh
pi@yahb ~ $ ./initpin.sh
```

B. For python code

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

python CarRun.py

```
pi@raspberrypi:~/python_code $ python CarRun.py
```

2) We can input: **ctrl+c** to stop this process, which means 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@yahb ~ $ sudo chmod 777 initpin.sh
pi@yahb ~ $ ./initpin.sh
```

After completing the above steps, the experiment is over.

5) Experimental phenomenon

After running program, the car will delay 0.5s, advance 1s, back 1s, turn left 1s, turn right 1s, spin left, spin right, stop 2s.