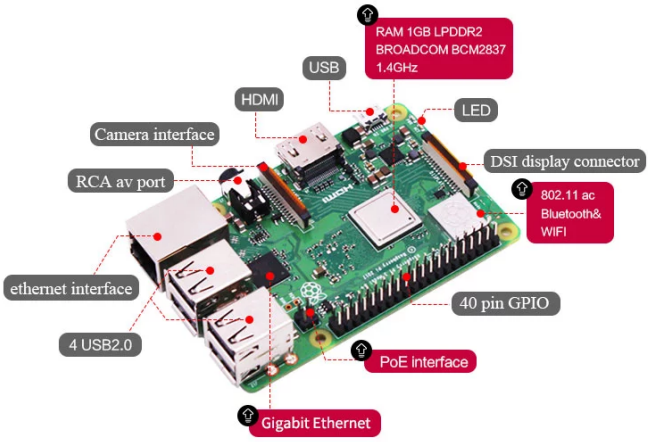
14.Raspberry Pi platform ------- Servo_control

1.Preparation

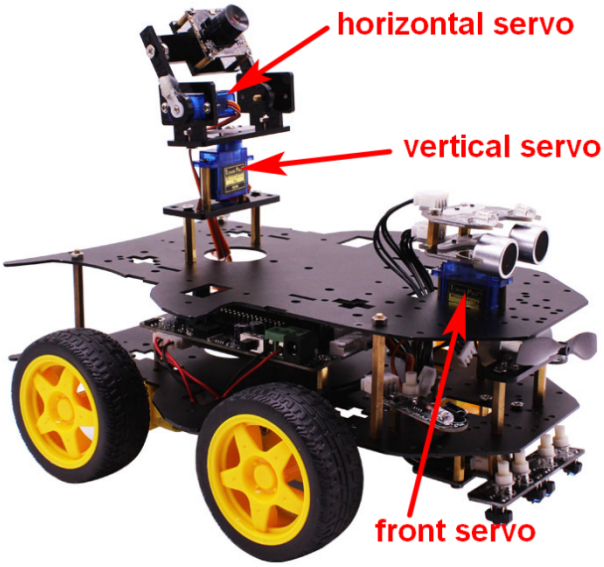


1-1 Raspberry Pi board

1-2 SG90 servo

2.Purpose of Experimental

After running the Servo_control executable in the Raspberry Pi system. The servo will start to turn to different angles.



3.Principle of experimental

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

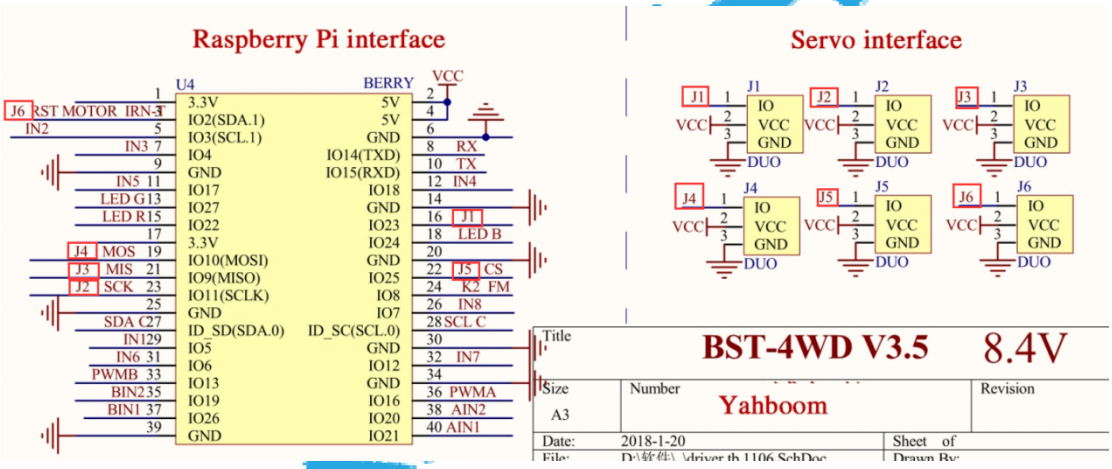
20ms and a width of 1.5ms. It will compares the DC bias voltage with the voltage of the potentiometer to obtain a voltage difference and output. The positive and negative of the voltage difference is outputted to the motor drive chip to determine the forward and reverse of the motor.

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°～180° corresponds, as shown below.

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

4.Experimental principle

4-1 About the schematic



4-1 Servo 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

4-2 Raspberry Pi 40 pins comparison table

The control interface of this 4WD car possess J1, J2, J3,J4,J5,J6 six interfaces (some of the six interfaces and other functions are multiplexed interfaces, which cannot be used at the same time).

Note:The front servo is be connected to servo interface J1

The horizontal servo is be connected to servo interface J2

The vertical servo is be connected to servo interface J3

4-2 According to the circuit schematic:

J1---16(Physical pin)-----4(wiringPi)

J2---23(Physical pin)-----13(wiringPi)

J3---21(Physical pin)-----14(wiringPi)

(Note: We use the wiringPi library to write code.)

4-3 About the code

See the Servo_control.c file for the code.

```
/**
* @par Copyright (C): 2010-2019, Shenzhen Yahboom Tech
* @file          Servo_control.c
* @author        xiaozhen
* @version       V1.0
* @date          2019.02.14
* @brief         Servo_control
* @details
* @par History
*
*/
#include <wiringPi.h>
#include <softPwm.h>

//Define the servo p
int ServoPin =  4;      //front servo J1---4(wiringPi), hor
/*If you need to control two other servos, please modify
```

If you need to control two other servos, please modify the definition of this pin.

5.Steps

First, you need to remotely transfer the Servo_control.c we provide to the Raspberry Pi image system via SSH.

The transfer steps are as follows:

1).Click on the location shown below to get the Winscp transfer tool.

2).Click on the location shown below to log in to the Raspberry Pi system.

(The IP address is 192.168.0.1 and the password is yahboom.)

3). Click on the location shown below to transfer file.

4). We need to drag the Servo_control.c file to the right Raspberry Pi system.

5). After the drag and drop is completed, as shown in the figure below, we can see Servo_control.c in the Raspberry Pi system on the right.

After the above steps, we have successfully transferred the Servo_control.c file to the Raspberry Pi image.

Then, you need to remotely log in to the Raspberry Pi system by putty, as shown below:

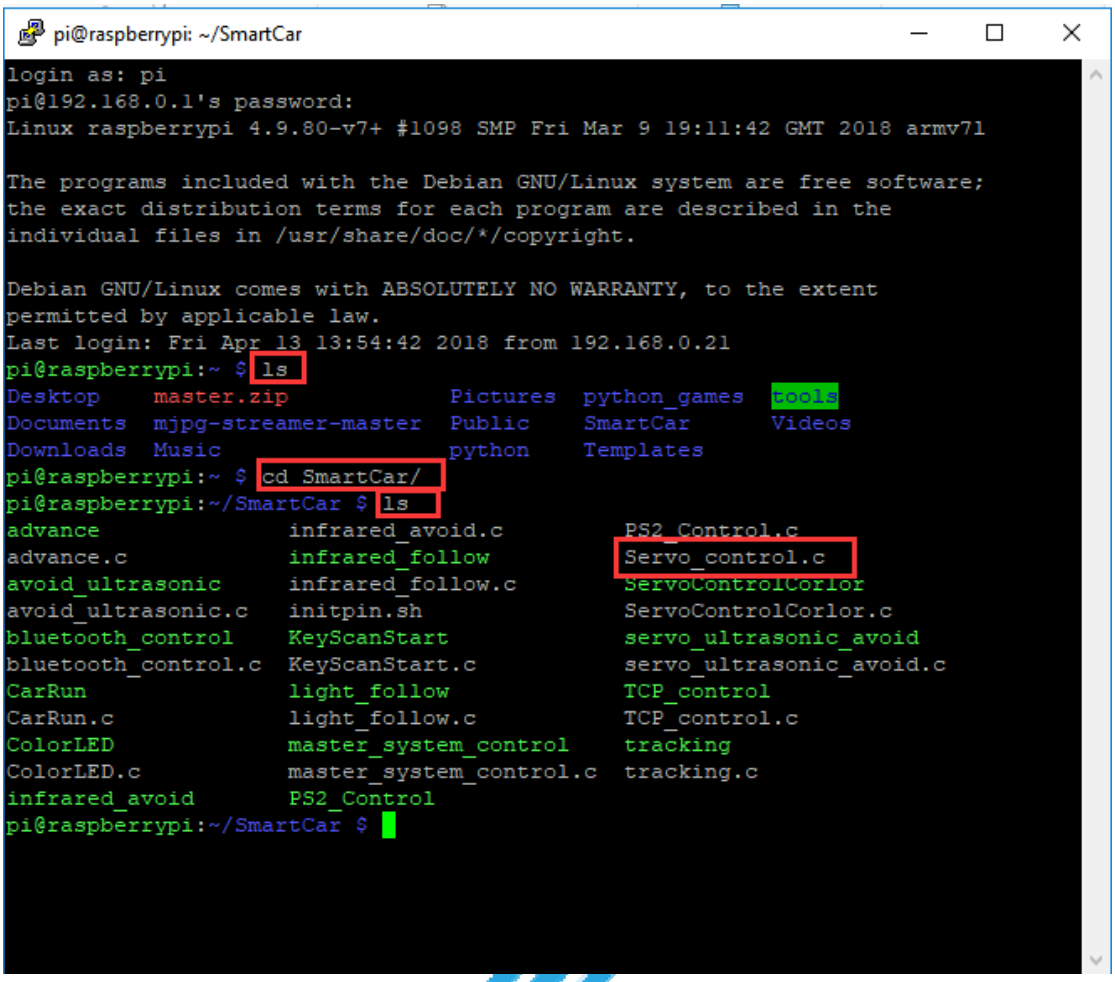
1).We need to go to the SmartCar directory:

Enter command: cd SmartCar

ls

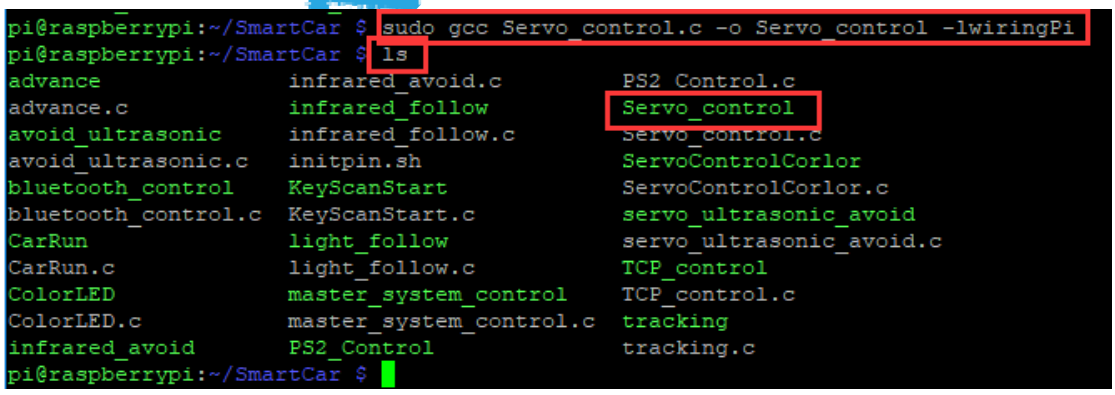
We can see the Servo_control.c file inside.

As shown below:



2).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: sudo gcc Servo_control.c -o Servo_control -lwiringPi



3).We need to run the compiled executable file in the Raspberry Pi system.We need to input: ./Servo_control

As shown in the figure below.

```
pi@raspberrypi:~/SmartCar $ sudo gcc Servo_control.c -o Servo_control -lwiringPi
pi@raspberrypi:~/SmartCar $ ls
advance               infrared_avoid.c         PS2_Control.c
advance.c             infrared_follow          Servo_control
avoid_ultrasonic      infrared_follow.c        Servo_control.c
avoid_ultrasonic.c    initpin.sh               ServoControlCorlor
bluetooth_control     KeyScanStart             ServoControlCorlor.c
bluetooth_control.c   KeyScanStart.c           servo_ultrasonic_avoid
CarRun                light_follow             servo_ultrasonic_avoid.c
CarRun.c              light_follow.c           TCP_control
ColorLED              master_system_control    TCP_control.c
ColorLED.c            master_system_control.c  tracking
infrared_avoid        PS2_Control              tracking.c
pi@raspberrypi:~/SmartCar $ ./Servo_control
```

4)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 SmartCar directory.)
You need to input: chmod 777 initpin.sh
./initpin.sh

After completing the above steps, the experiment is over.