Dongguan WHEELTEC INTELLIGENT technology co., LTD

Aliexpress: minibalance.aliexpress.com/store/4455017

Taobao shop: minibalance.taobao.com

Web: www.wheeltec.net

STM32 Moving Chassis

Development Manual

Recommend to follow our official account for updated information

Version description:

version	date	Content description
V3.5	2021/03/9	First release

Preface

The full set of tutorials for ROS navigation robots includes three documents : " STM32 Moving Chassis Development Manual", " Ubuntu Configuration Tutorial", and " ROS Development Tutorial". For the Ubuntu configuration tutorials for virtual machines and Raspberry Pi (Jetson Nano , Jetson TX2 , industrial computer, etc.), please see " Ubuntu Configuration Tutorial"; for ROS development tutorials, please see " ROS Development Tutorial". The content of this document tutorial is mainly used to explain the kinematic analysis, communication protocol, control mode, etc. of the ROS robot moving chassis. The moving chassis is equipped with two controllers, namely the Raspberry Pi (Jetson Nano , Jetson TX2 , industrial computer, etc.) and the STM32 controller. Data is transmitted between the two through serial communication. For detailed instructions on the wiring, please see the chapter five. Among them, the Raspberry Pi (Jetson Nano , Jetson TX2 , industrial computer, etc.) is installed with Ubuntu to run ROS ; the STM32 controller is used to control the moving chassis and collect odometer information, battery information,IMU information.

The task names, function names, etc. appearing in this document are the contents of the built-in program of the moving chassis. If you do not plan to read the program of the robot STM32 controller , you can ignore the program functions and program tasks appearing below (for example: APP_Show() function, Send task data_task , data_transition() function), without affecting the reading and understanding of the entire development manual.

This document is applicable to differential car series, Ackerman car series, mecanum-wheel car series and omni-wheel car series. Each car series is divided into different models, as shown in the following table 0-0 .

Table 0-0 Car series and models

Model / series	0	1	2	3	4	5
Differential car series	Top version two-wheel differential car (conventional type)	Top version two-wheel differential car (heavy load type)	Four-wheel two-wheel drive pendulum suspension (conventional type)	Four-wheel two-wheel drive pendulum suspension (heavy load type)	Four-wheel two-wheel drive independent suspension (conventional type)	Four-wheel two-wheel drive independent suspension (heavy load type)
Ackerman car series	High version Ackerman car (conventional type)	High version Ackerman car (heavy load type)	Top version ackerman Pendulum suspension (conventional type)	Top version ackerman Pendulum suspension (heavy load type)	Top version ackerman Independent suspension (conventional type)	Top version ackerman Independent suspension (heavy load type)
Mecanum wheel car series	High version mecanum wheel without bearing seat	High version mecanum wheel Pendulum suspension	High version mecanum wheel Independent suspension	Top version mecanum wheel pendulum suspension (conventional type)	Top version mecanum wheel pendulum suspension (heavy load type)	Top version mecanum wheel independent suspension (conventional type)
Omni wheel car series	High version omni wheel (conventional type)	High version omni wheel (heavy load type)	Top version omni wheel (conventional type)	Top version omni wheel (heavy load type)		

Model / series	0-6	7
Mecanum wheel eight -drive series	Top version mecanum wheel independent suspension eight-drive conventional type	Top version mecanum wheel independent suspension eight-drive heavy-load type

Directory

Preface..2

1. Robot control mode.. 6

1. 1 Robot movement speed unit..6

1. 2 ROS (serial port 3) control.. 7

1. 3 APP control.. 17

1. 4 PS2 control.. 20

1. 5 HOT-RC remote control..21

1. 6 CAN control.. 23

1. 7 Serial port 1 control...26

2. OLED display content.. 29

2. 1 OLED specific content...29

2. 2 OLED general display content explanation.................................30

3. Elimination of gyroscope zero drift... 33

4. Robot kinematics analysis..34

4. 1 Two-wheel differential car... 34

4. 2 Ackerman Car..36

4. 3 Mecanum wheel car...38

4. 4 Omni wheel car... 40

4. 5 PI control program source code..41

5. Wiring instructions.. 43

5. 1 Power supply for Raspberry Pi...43

5. 2 Raspberry Pi and STM32 controller serial communication................ 43

5. 3 Connect Raspberry Pi to Slamtec A1 radar...................................44

5. 4 Connect Raspberry Pi to slamtec A2/A3 radar..............................44

6. Control flow chart...45

6. 1 Control flowchart of robot motor.. 45

6.2 Robot STM32 program structure diagram... 46
6.3 Robot controller connection diagram.. 46
7. Matters needing attention... 48
7.1 About the code..48
7.2 About the power interface on the adapter board................................. 48
7.3 About the motor...48
7.4 About the battery...49
8. How to download program to STM32 controller......................... 50
8.1 Serial download..50
8.2 SWD download... 51

1. Robot control mode

This chapter mainly gives a detailed description of the control and use of the robot. The robot supports 6 control modes: APP remote control, PS2 wired handle, ROS control, RC remote control, CAN control, and serial port control . The control mode is displayed in the lower left corner of the OLED display, and the ROS control mode is used by default after booting .

Note: The car will move a little distance after the initialization is completed to detect whether there is a problem with the movement of the car. If there is a problem, the car will not be controlled, and the upper right corner of the OLED display will display " TYPE : X ".

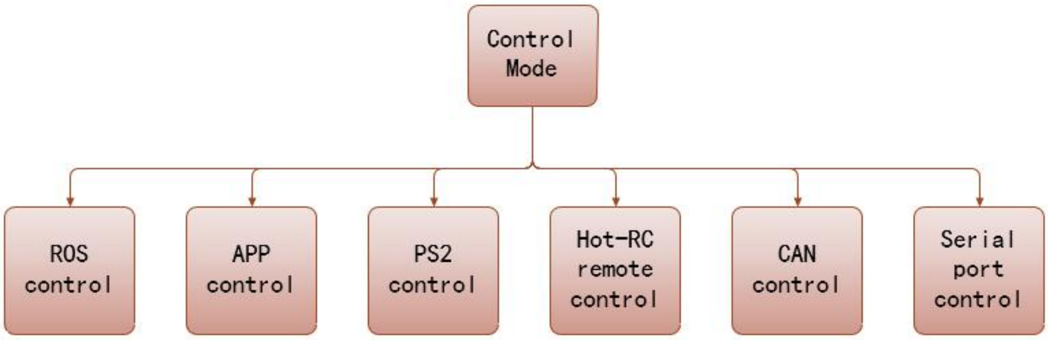

Figure 1-1　Robot control mode

1.1 Robot movement speed unit

Here is an explanation of the unit of robot movement speed, which is m/s (meters per second) . The calculation formula of the specific "encoder raw data" converted into " m/s (meters per second) " is shown in Figure 1-1-1 .

Encoder_A_pr is the raw data of the encoder, CONTROL_FREQUENCY is the control frequency (unit: HZ), Encoder_precision is the accuracy of the encoder (related to the mechanical structure of the motor and the encoder chip), wheel_perimeter is the circumference of the wheel (unit: meters), and the final MOTOR_A .Encoder is the actual movement speed of the robot (unit: m / s). Here only the encoder data conversion of A motor is shown , and the encoder calculation conversion of the other B , C and D motors is the same.

MOTOR_A.Encoder= Encoder_A_pr*CONTROL_FREQUENCY*Wheel_perimeter/Encoder_precision;

Figure 1-1-1 Robot movement speed conversion formula

The positive direction calibration of XYZ three-axis speed is shown in Figure 1-1-2 below.

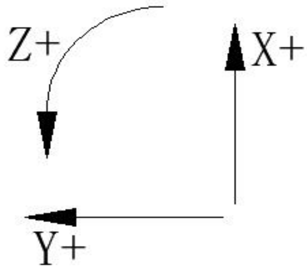

Figure 1 -1-2 The positive direction of robot XYZ three-axis speed

1. 2 ROS (serial port 3) control

After the robot is powered on, the ROS control mode is used by default. This section only describes how to make the robot move through the ROS environment , in particular the principles and ROS some relevant definition of the concept see " ROS Development Manual." For the virtual machine and ros development environment setup and how to remotely log in to the ubuntu system on the robot , please see " Ubuntu Configuration Tutorial".

Here is to use the keyboard on the virtual machine to control the robot's movement.

① virtual machine remotely logs in to the robot ubuntu system

It should be noted that using the virtual machine to control the robot movement requires two terminals to be opened separately, and each terminal must be separately remotely logged in to the robot ubuntu system. Enter the command shown in Figure 1-2-1 in the virtual machine to remotely log in to the ubuntu system of the robot .

```
passoni@passoni:~$ ssh wheeltec@192.168.0.100
```

Figure 1-2-1 ssh wheeltec@192.168.0.100

② Run the function package

In the first terminal, open the " turn_on_wheeltec_robot " under the " turn_on_wheeltec_robot " function package of the robot to enable the robot control node.

```
wheeltec@wheeltec:~$ roslaunch turn_on_wheeltec_robot  turn_on_wheeltec_robot.launch
```

Figure 1-2-2 roslaunch turn_on_wheeltec_robot turn_on_wheeltec_robot.launch

In the second terminal, open the keyboard control node of the robot movement. In the second terminal, open the " keyboard_teleop " keyboard control node under the " wheeltec_robot_rc " function package of the robot .

```
wheeltec@wheeltec:~$ roslaunch wheeltec_robot_rc keyboard_teleop.launch
```

Figure 1-2-3 roslaunch wheeltec_robot_rc keyboard_teleop.launch

After opening the keyboard control node, you can see some control prompts. At this time, you can use the keyboard to control the robot movement. The specific keyboard control corresponding effects are shown in Table 1-1 . Press " crtl+c " or close the terminal directly to exit the control node.

```
Control Your Turtlebot!
---------------------------
Moving around:
   u    i    o
   j    k    l
   m    ,    .

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%
space key, k : force stop
anything else : stop smoothly

CTRL-C to quit

currently:      speed 0.2       turn 1
```

Figure 1-2-4 Open the keyboard control node terminal information

Table 1-1 Description of keyboard control robot motion instructions

Keyboard keys	u	i	o	j	k	l	m	,
Robot achieve effect	Left front movement	Forward movement	Right front movement	Turn left	Emergency stop	Turn right	Left back movement	Back movement

Keyboard keys	.	Space	q	z	w	x	e	c
Robot achieve effect	Right back move ment	Emer gency stop	Move ment speed +10%	Move ment speed -10%	line speed +10%	line speed -10%	angle speed +10 %	angle speed -10%

③ STM32 sends data to ROS

The communication between ROS and STM32 controller (moving chassis) is realized through serial port. STM32 controller uses serial port 3 and the baud rate is 115200. Communication protocol includes: STM32 controller send data to ROS, ROS transmit data to STM32 controller. Open the source code of the STM32 controller. The code for serial communication is in the usartx.c file of the STM32 controller .

STM32 sends data to ROS and uses a date_task task to execute at a frequency of 20hz . The data sent includes: frame head and end, robot enable flag, robot XYZ three-axis speed, IMU three-axis acceleration, three-axis angular velocity, battery voltage, data check bit , detailed data sent see Table 1-2 below .

The way of sending data: Pack the data to be sent into an array, the length of the array is 24 bytes, and use the serial port to send out bit by bit. Because the serial port can only send one 8 -bit (1 byte) data at a time, the 2 -byte (short type) data will be split into the upper 8 bits and the lower 8 bits for transmission.

The function assigned to the data before sending it is the "data_transition()" function in the "usartx.c" file; the function that sends the data is the "USART3_SEND()" function in the "usartx.c" file.If you need to change the content of the data sent, just change the data_transition() function; if you need to change the length of the data to send, you need to change " SENT [] " array length of this at the same time also need to modify " USART3_SEND () " function for the number of cycles and the length of the data received by the ROS terminal must be modified accordingly.

What needs to be explained in the data sent is that the frame header is a fixed value of 0X7B , the frame end is a fixed value of 0X7D , and flag_stop is the stop flag bit of the motor (0 is enabled, 1 is disabled). The calculation method of the data check bit is BCC check (exclusive OR of all data bits (including the frame header)), and the final result is the data check bit. The calculation process of the data check bit is shown in Figure 1-2-6 .

```
system.c    balance.c    usartx.c    pstwo.c
****************************************************
void USART3_SEND(void)
{
  unsigned char i = 0;
  for(i=0; i<24; i++)
  {
    usart3_send(Send_Data.buffer[i]);
  }
}
```

Figure 1-2-5 USART3_SEND() function

```
usartx.c
/**************************************************************
函数功能：计算发送的数据校验位
入口参数：
返回  值：检验位
**************************************************************/
u8 Check_Sum(unsigned char Count_Number,unsigned char Mode)
{
  unsigned char check_sum=0,k;
  //发送数据的校验
  if(Mode==1)
  for(k=0;k<Count_Number;k++)
  {
  check_sum=check_sum^Send_Data.buffer[k];
  }
  //接收数据的校验
  if(Mode==0)
  for(k=0;k<Count_Number;k++)
  {
  check_sum=check_sum^Receive_Data.buffer[k];
  }
  return check_sum;

}
```

Figure 1-2-6 Data check digit calculation function

Table 1-2 Data sent by STM32 to ROS

<table>
<tr><td>data content</td><td colspan="2">Frame header (Fixed value 0X7B)</td><td colspan="2">flag_stop</td><td colspan="2">robot X axis speed</td><td colspan="2">robot Y axis speed</td><td colspan="2">robot Z axis speed</td><td colspan="2">Acceleromet er X axis Acceleration</td><td colspan="2">Acceleromet er y axis Acceleration</td></tr>
<tr><td>data type</td><td colspan="2">Uint8</td><td colspan="2">Uint8</td><td colspan="2">short</td><td colspan="2">short</td><td colspan="2">short</td><td colspan="2">short</td><td colspan="2">short</td></tr>
<tr><td>Occupy byte</td><td colspan="2">1</td><td colspan="2">1</td><td colspan="2">2</td><td colspan="2">2</td><td colspan="2">2</td><td colspan="2">2</td><td colspan="2">2</td></tr>
<tr><td>Array number</td><td colspan="2">0</td><td colspan="2">1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td></tr>
<tr><td colspan="15"></td></tr>
<tr><td>data content</td><td colspan="2">Accelerom eter z axis Acceleratio n</td><td colspan="2">Angular velocity meter X axis Angular velocity</td><td colspan="2">Angular velocity meter Y axis Angular velocity</td><td colspan="2">Angular velocity meter Z axis Angular velocity</td><td colspan="2">battery voltage</td><td colspan="2">data check digit</td><td colspan="2">End of frame (Fixed value 0X7D)</td></tr>
<tr><td>data Types of</td><td colspan="2">short</td><td colspan="2">short</td><td colspan="2">short</td><td colspan="2">short</td><td colspan="2">short</td><td colspan="2">Uint8</td><td colspan="2">Uint8</td></tr>
<tr><td>Occupy byte</td><td colspan="2">2</td><td colspan="2">2</td><td colspan="2">2</td><td colspan="2">2</td><td colspan="2">2</td><td colspan="2">1</td><td colspan="2">1</td></tr>
<tr><td>Array number</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td colspan="2">22</td><td colspan="2">23</td></tr>
</table>

There is another point to note here . The raw data of the robot XYZ three-axis speed, accelerometer, angular velocity meter and battery voltage are floating-point data (float), because the floating-point data is inconvenient to use the serial port to transmit, so these four before sending the data, amplify the floating-point number by a thousand times (reserve three decimal places), then force the amplified floating-point number into short data, and finally split the short data into two 8-bit data before sending data. Correspondingly, after the host computer receives the data, it needs to merge the two 8 -bit data of the received data and convert it into a short type, and then convert the unit after shrinking it by a thousand times.

Let's explain how to combine two 8-bit data and convert them to short type to get our actual speed data: our control quantity unit is mm/s (0.001m/s) , and the control quantity direction is from the high 8 -bit data The highest bit of the decision.

Example 1 : 21 B6=0010 0001 1011 011 , the highest bit is 0 , positive number, the speed is

21 B6=(2*16+1)*256+(B*16+6)=(2*16+1) *256+(11*16+6)=8630(mm/s) ;

Example 2 : A1 2F=1010 0001 0010 1111 , the highest bit is 1 , negative number, the speed is

2^16(FF FF+1)-A1 2F=5E D0+1=(5*16+E)*256+(D*16+0)+1=24272(mm/s) .

The following picture shows the actual data received through the serial port assistant after we connect to the serial port 3 . (Note that our serial port 3 is not integrated with CH340 , and we need to use the data cable for communication between ROS and STM32 , so the serial port assistant can communicate with STM32 serial port 3) .

Figure 1 -2-7 Data sent by car serial port 3

Let's convert the received 24 bytes of data:

The first byte: 0x7B , header;

The second byte: 0X00 , the motor is in a non-stop state;

The 3rd and 4th bytes: X axis speed, high 8 bits 0X01 (hexadecimal) = 0000 0001 (binary) , low 8 bits 0X01 (hexadecimal) = 0000 0001 (binary) , the highest bit is 0 , a positive number (forward) , and the speed is 1*256+1=257(mm/s) . This speed is the actual speed calculated by the trolley on the encoder data.

The 5th and 6th bytes: Y- axis speed, high 8 bits 0X00 (hexadecimal) = 0000 0000 (binary) , low 8 bits 0X01 (hexadecimal) = 0000 0001 (binary) , the highest bit is 0 , a positive number (left shift) , and the speed is 0*256+1=1(mm/s) . This speed is the actual speed calculated by the trolley on the encoder data.

The 7th and 8th bytes: Z axis speed, high 8 bits 0X00 (hexadecimal) = 0000 0000 (binary) , low 8 bits 0X00 (hexadecimal) = 0000 0000 (binary) , the highest bit is 0, a positive number (counterclockwise rotation) , and the speed is

0*256+0=0(0.001rad/s) . This speed is the actual speed calculated by the car on the encoder data.

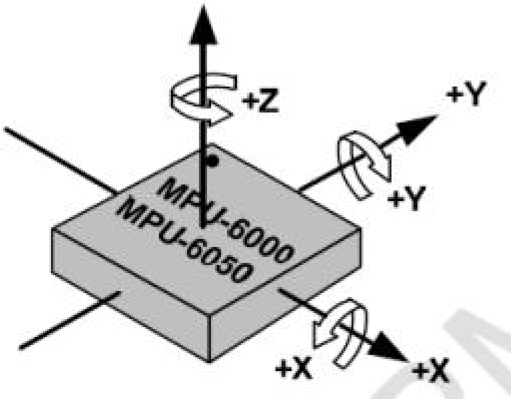

Figure 1 -2-8 Three-axis schematic diagram of MPU6050 calibration accelerometer and angular velocity meter

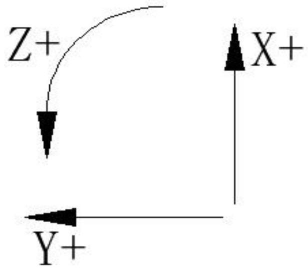

Figure 1 -2-9 Robot XYZ three-axis

The next 12 bytes are the data of the three-axis accelerometer and angular speedometer data. Note that the three-axis direction of XYZ has changed. The positive direction of X- axis is rightward, the positive direction of Y- axis is forward, and the positive direction of Z- axis is ascending. The speed of the accelerometer and angular speedometer is the speed of rotation around the XYZ axis, as shown in Figure 1-2-8 above. Because gyro XY robot axis direction with our calibration XY-axis direction there is a difference, so the XY angular velocity and the angular velocity data transmission shaft is made a correction before conversion.

```
Send_Data.Sensor_Str.Accelerometer.X_data=accel[1]; //加速度计X轴加速度
Send_Data.Sensor_Str.Accelerometer.Y_data=-accel[0]; //加速度计Y轴加速度
Send_Data.Sensor_Str.Accelerometer.Z_data=accel[2]; //加速度计Z轴加速度

/*将要发送的数据拆分到数组中*/
Send_Data.Sensor_Str.Gyroscope.X_data=gyro[1]; //角速度计X轴角速度
Send_Data.Sensor_Str.Gyroscope.Y_data=-gyro[0]; //角速度计Y轴角速度
```

Figure 1 -2-10 Correction of the XY axis direction of the robot

The 9th and 10th bytes: X-axis acceleration, high 8 bits 0XFE (hexadecimal)=1111 1110 (binary) , low 8 bits 0X96 (hexadecimal)=1001 0110 (binary) , the highest bit It is 1 , a negative number, and the size is 2^16(FF FF+1)-FE 96 =01 69=1*256+105=361 , which is converted into acceleration 361/1672=0.2159( ) m/s^2 .

The 11th and 12th bytes: Y-axis acceleration, high 8 bits 0XFD

(hexadecimal)=1111 1101 (binary) , low 8 bits 0XCE (hexadecimal)=1100 1110 (binary) , the highest bit It is 1, a negative number, and the size is 2^16(FF FF+1)-FD CE=0231=2*256+49=561, which is converted to acceleration 125/1672=0.3355( ) m/s^2 .

The 13th and 14th bytes: Z- axis acceleration, high 8 bits 0X40 (hexadecimal) = 0100 0000 (binary) , low 8 bits 0X80 (hexadecimal) = 1000 0000 (binary) , the highest bit It is 0 , a positive number, and the size is 64*256+128=16512 , which is converted to acceleration 15684/1672=9.8756( ) m/s^2 .

The 15th and 16th bytes: X axis angular velocity, high 8 bits 0XFF (hexadecimal)=1111 1111 (binary) , low 8 bits 0XFB (hexadecimal)=1111 1011 (binary) , the highest bit It is 1 , a negative number, and the size is 2^16(FF FF+1)-FF FB =00 04=0*256+4=4 , converted to angular velocity 9/3753=0.0011(rad/s) .

The 17th and 18th bytes: Y- axis angular velocity, high 8 bits 0X00 (hexadecimal) = 0000 0000 (binary) , low 8 bits 0X07 (hexadecimal) = 0000 0111 (binary) , the highest bit It is 0 , a positive number, and the size is 0*256+7=7 , which is converted to angular velocity 7/3753=0.0019(rad/s) .

The 19th and 20th bytes: Z axis angular velocity, high 8 bits 0X00 (hexadecimal) = 0000 0000 (binary) , low 8 bits 0X01 (hexadecimal) = 0000 0001 (binary) , the highest bit It is 0 , a positive number, and the size is 0*256+1=1 , which is converted to angular velocity 1/3753=0.0003 (rad/s) .

The 21st and 22nd bytes: battery voltage, high 8 bits 0X58 (hexadecimal)=0101 1000 (binary) , low 8 bits 0X38 (hexadecimal)=0011 1000 (binary) , the highest bit is 0 , a positive number, the size is 88*256+56=22584 , and the voltage size is 22584mv (millivolt) .

The 23rd byte: BBC check digit (exclusive OR of the first 22 bytes) , 0X83=0X7B^0X00^0X01^0X01^0X00^0X01^0X00^0X00^0XFE^0X96^0XFD^0XCE^0X40^0X80^0XFF^ 0XFB^0X00^0X07^0X00^0X01^0X58^0X38 .

(If you want to verify the result of this check digit, you can use some web online platforms to calculate the check digit)

The 24th byte: 0X7D , end of frame;

④ STM32 receives the data sent by ROS

The receiving data is received in the way of interrupted receiving. The received data includes the robot product signal, the enable control flag bit, the robot three-axis target speed, and the data check bit.

Among them, the robot product model is used to identify the current running product for the STM32 controller. The specific content is described in detail in the section " 2. OLED display content"; flag_stop is the robot's enable control bit, which send enabled by default; the robot's three-axis target speed is used to control the movement of the robot, see Table 1-3 for the specific receiving content . It should be noted that the array number in the table is the array number of the data sent by the host computer.

Table 1-3 STM32 receives the data sent by ROS

data content	Frame header (Fixed value 0X7B)	Reserved	Reserved	robot X axis Target speed		robot Y axis Target speed		robot Z axis Target speed		data check digit	End of frame (Fixed value 0X7D)
data Types of	Uint8	Uint8	Uint8	short		short		short		Uint8	Uint8
Occupy byte	1	1	1	2		2		1		1	1
Array number	1	2	3	4	5	6	7	8	9	10	11

Note: Differential cars and Ackerman cars do not support Y-axis movement control. Only mecanum-wheel cars and omni-wheel cars support Y-axis movement control.

Among the 7 ways to control the robot, the control priority of ROS is the highest.

Whenever the serial port 3 of the STM32 controller receives data, it is forced to enter the ROS mode. The reason for not receiving data in the first 10 seconds is to eliminate the useless data sent during the robot power-on process. To start receiving data after a waiting period of 10 seconds, first detect the data frame header, and start receiving data after detecting the data frame header; after the data is received, verify whether the data check bit at the end of the frame is wrong, and data is not used until the data check bit is correct. Please refer to the USART3_IRQHandler() serial port interrupt function in the usartx.c file in the STM32 code for the serial port interrupt receiving data .

1.3 APP control

The robot supports APP Bluetooth control and online parameter adjustment. In APP mode, directly use the joystick to control the movement of the robot in space. The speed unit of APP controlling the robot movement is mm/s (millimeters / second). The acceleration and deceleration buttons on the diagonally above the joystick will increase / decrease the speed of the robot by 100 (mm/s) every time you press it .

While the APP is controlling the robot , the robot will send data to the mobile phone via Bluetooth (the APP supports both WIFI and Bluetooth communication) . You can see the sent data in the Debug column of the APP . You can see the APP_Show() function in the code's show.c file for details on what to send. APP control mode the principle is the Bluetooth (or WIFI) serial communications control, the robot 's APP control the serial port 2 , the baud rate is set to 9600 , its control instruction is in the serial port 2 receive interrupt service function.

① On-line parameter tuning

Install the latest version of MiniBalance APP on your Android phone, and then follow the corresponding video tutorials to remotely control the robot or perform online parameter adjustments. In the "Debug" interface, you can click the "parameter x " to customize the name of each channel . The specific effect is shown in Figure 1-3-1 and Figure 1-3-2 . In addition, before adjusting the PID parameters, we need to click [Get Device Parameters] (call up by the menu button in the upper right corner) , update the robot's PID parameters to the APP , and then drag the slider. When we let go, the APP can Send the parameters to the robot.

Parameter 0 is the speed parameter of the car, and adjusting parameter 0 can adjust the speed of the car.

Parameters 1 and 2 are PI parameters for the motion control of the trolley.

It should be noted that, assuming that the current robot control speed is 40 , when we want to adjust the speed parameter to 150 (out of range), we need to adjust it to 80 for the first time , and then click [Get device parameters] again , then the adjust range is changed to 0-160 . The robot was set for a speed range of 20-200 . For a detailed explanation of the robot control speed unit, please refer to "Chapter 3 Robot Movement Speed Unit" below.

Figure 1-3-1 Default tuning interface

Figure 1-3-2 Interface after obtaining device parameters

(Need to modify the parameter name by yourself)

② APP control robot

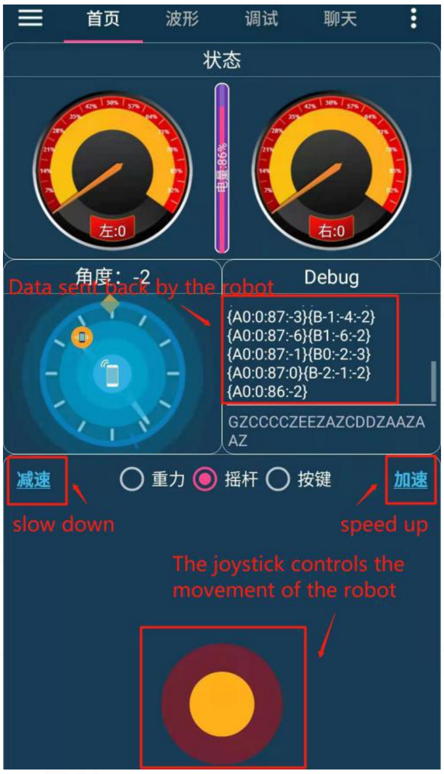

Figure 1-3-3 APP control interface

Each operation corresponding to the APP operation interface actually sends different instruction information to the robot (switching control methods and interfaces also send instructions), the robot responds after receiving the instruction information, and each operation of the detailed APP operation interface is sent correspondingly The information is shown in Table 1-4 :

Table 1-4 Description of APP interface operation instructions

APP joystick	↑	↗	→	↘	↓	↙	←	↖
Data received by the robot	0x41	0x42	0x43	0x44	0x45	0x46	0x47	0x48
Robot achieve effect	Forward movement	Right front movement	Turn right in place	Right back movement	Back movement	Left back movement	Turn left in place	Front left movement
button	gravity		joystick		button		slow down	accelerate

Data received by the robot	0x49	0x4a	0x4b	0x59	0x58

Note: After the mobile phone is successfully connected to the Bluetooth of the car, you need to push the joystick forward for 0.5 seconds to formally control the car.

1.4 PS2 control

The PS2 mode uses a PS2 wired controller. The PS2 wireless controller is designed based on gaming occasions. It is not suitable for remote control of this kind of robot with relatively large power . It will easily cause unstable communication and cause the robot to lose control. The sequence of PS2 mode control is: first plug in the handle before powering on, and then turn on the power. At this time, the red light on the handle is always on to indicate normal operation. If the indicator is not on, press the button above the indicator and press the START key presses into the handle mode, then you can see the lower left corner of the display shows " PS2 ."

In PS2 mode, use the left remote control to control the robot to move forward and backward, the right joystick to control the left and right steering of the robot (the omnidirectional motion car is the left remote control to control the movement of the car in 8 directions in the space, and the right joystick to control the car in place self-rotating motion) . The reason for this design is that if the same joystick is used for the front and rear and left rear controls, it is easy to accidentally touch the steering. The two buttons in the upper left corner are acceleration and deceleration buttons.

Note: In PS2 mode, the PS2 handle must be plugged in before powering on, otherwise it is easy to burn the handle and cause the motor to rotate randomly. PS2 handle in the boot mode , need to press PS2 handle button START after entering key PS2 handle mode.

Figure 1 -4-1 PS2 handle physical picture

1.5 HOT-RC remote control

The correct use steps of the HOT-RC remote control are: first connect the HOT-RC remote control receiver to the car, turn on the power of the HOT-RC remote control, power on the car, and the indicator light of the HOT-RC remote control receiver is always on, indicating that it is connected, and you can see the display in the lower left corner of the display " R-C ". The control method of the HOT-RC remote control is: the left joystick of the HOT-RC remote control controls the car to move forward and back, the right joystick controls the car to turn left and right (the omnidirectional car is the left remote control to control the movement of the car in 8 directions in the space, and the right joystick controls the car spins on the spot) . The SWC switch on the upper right controls the car to select normal mode and low speed mode. Before turning off the HOT-RC remote controller , you need to turn off the car and then turn off the remote control.

Figure 1-5-1 The physical picture of the hot-RC remote control

It should be noted that the remote control channel of the HOT-RC needs to be configured as shown in Figure 1-5-2 , where the leftmost button is in the middle, and the remaining four buttons are in the bottom. These channels are for adjusting the direction of control.

Figure 1-5-2 The physical image of the Hot-RC

Next, I will explain how to connect the RC receiver to the adapter board.

Figure 1 -5-3 Physical image of the Hot-RC Receiver

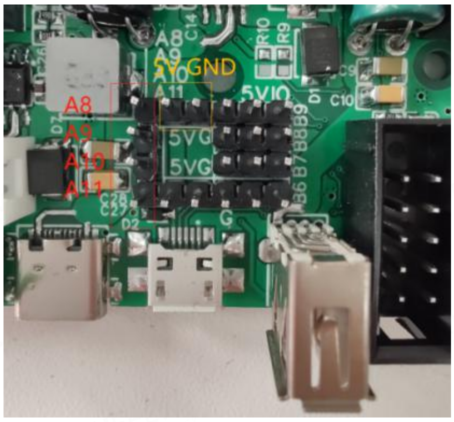

Figure 1 -5-4 Hot-RC controller interface on the adapter board

Table 1-5 Corresponding channels of model airplane remote control pins

Hot-RC receiver	GND	5V	CH1	CH2	CH3	CH4
Adapter board	G	5V	PA8	PA9	PA10	PA11

As shown in Figure 1-5-4 and Table 1-5 , the HOT-RC receiver has three columns, namely GND , 5V and signal line. When we use it, connect a pair of GND and 5V , then CH1, CH2, CH3 and CH4 of the signal line are respectively connected to PA8, PA9, PA10 and PA11 of the pin row on the connecting board.

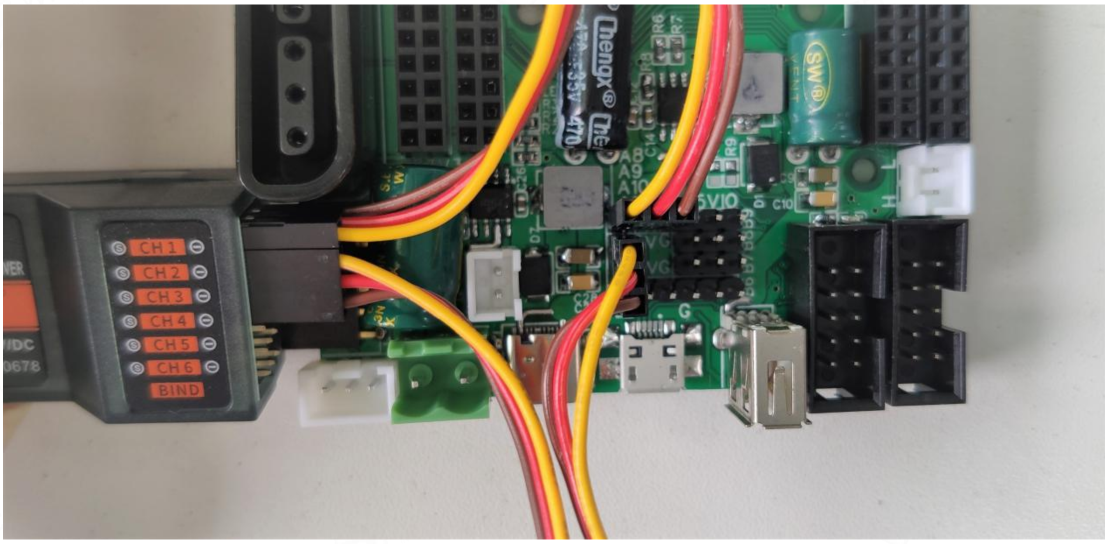

Figure 1 -5-5 Wiring example of Hot-RC

1. 6 CAN control

The car supports CAN communication, and the connection mode of CAN communication is CANL to CANL and CANH to CANH . If you want the car to receive CAN control commands, you need to send an enable command to the CAN interface first , and the baud rate is 1M .

The enable command format is as follows:

Identifier ID: 0X121

Frame type: standard frame

Frame format: DATA

DLC: 8 bytes

Table 1-6 CAN enable command

Data field	tx[0]	tx[1]	tx[2]	tx[3]	tx[4]	tx[5]	tx[6]	tx[7]
content	10	12	15	19	24	30	37	Flag

Note: The data in this table is 10 hexadecimal form

When Flag=1 , CAN control is enabled, and the controller will no longer receive commands from other control modes. After CAN control is enabled, you can see " CAN " in the lower corner of the display , and then we can send CAN commands to control the car. The description of the control instruction is as follows:

Identifier ID: 0X121

Frame type: standard frame

Frame format: DATA

DLC: 8 bytes

Table 1-7 The car receives CAN control commands

Data field	rx[0]	rx[1]	rx[2]	rx[3]	rx[4]	rx[5]	rx[6]	rx[7]
content	Car X direction control amount high 8 bits	Car X direction control amount low 8 bits	Car Y direction control amount high 8 bits	Car Y direction control amount low 8 bits	Car Z direction control amount high 8 bits	Car Z direction control amount low 8 bits	Reserved	Reserved

rx[6] and rx[7] are reserved data bits for us to add the data we need to transmit.

CAN communication comes with BBC check, so there is no need for data check bit here.

Note: Differential cars and Ackerman cars do not support Y-axis movement control. Only mecanum-wheel cars and omni-wheel cars support Y-axis movement control.

The car can also send its own data while receiving CAN command control. The function to execute CAN sending data is CAN_SEND() located in the file [usartx.c] . By default, the task of sending data has been opened and the data to be sent is set. If you need to customize sending other content, just replace the contents of Send_Data.buffer [I] in the CAN_SEND function.

Due to the large number of data to send a total of 24 bytes, the data is sent eight bytes at a time, in three tranches,the MCU that receives the data sent by the car can confirm which group of data is currently received through the identifier ID. The identifier for sending the first group of data is 0X101 , the identifier for sending the second group of data is 0X102 , and the identifier for sending the third group of data is 0X103. The content of CAN transmission data is the same as that of serial port 1 and serial port 3 (the interface for communicating with ROS) . For this part, please refer to section 1.2 ROS (serial port 3) control.

The matching CAN sending routine is equipped with the interrupt service function received by CAN to receive the data sent out by the car. Sample code which device is adapted STM32F103RC or same series microcontroller, at the same time, VP230 chip is needed to convert the data of SCM before CAN communication CAN be formally carried out with the car, the wiring as shown below, pins of PB8 and PB9 of SCM are connected to D and R pins of VP230, and CANH and CANL pins of VP230 are connected to CANH and CANL pins of trolley (that is, H to H, L to L).

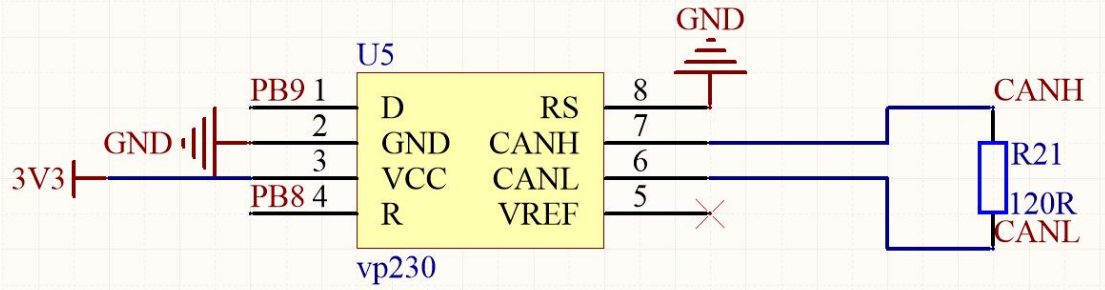

Figure.1 -6-1 The MCU and the car through VP230 conversion

If you want to check whether the data received by CAN communication is correct, you can use the host computer to connect the serial port 1 of the microcontroller . After the single-chip microcomputer receives the information sent by the CAN communication of the car, it will send it out through the serial port 1 of the single-chip microcomputer , and the baud rate is 9600 .

Note: The CAN and serial port routines provided in the information are not the codes running on the car. It does not mean that you need to download the serial port routines if you need serial communication. The car already has a program (firmware) by default, and you can operate it without downloading any program. The CAN and serial port routines provided by the data are programs that can be run on other microcontrollers to communicate with the car.

1.7 Serial port 1 control

The receiving control command of serial port 1 is exactly the same as that of ROS (serial port 3) . When serial port transmission, it is necessary to pay attention to the same baud rate settings of both communication parties. The baud rate of serial port 1 in the program is 115200 . After receiving the data from the serial port 1 , the car enters the serial port control mode, and the lower left corner of the display shows " USART ".

Note: Because the Hot-RC and serial port 1 share the pin PA9 (serial port TX pin) , the car can only be controlled through serial port 1 by default , but the car cannot send its own data (ie 24 bytes of data) . If you need the car to send car data through serial port 1, you need to manually comment out the Hot-RC initialization function TIM1_Cap_Init (0XFFFF, 72-1); (Near line 191 of system.c) , and uncomment the serial port 1 sending data function USART1_SEND(); (Near line 30 of usartx.c) .

Table 1-8 The car receives serial control commands

Data field	Frame header	tx[0]	tx[1]	tx[2]	tx[3]	tx[4]	tx[5]	tx[6]	tx[7]	tx[8]	End of frame
content	0X7B	Reserved	Reserved	The upper 8 bits of the X-direction control amount of the car	The lower 8 bits of the X-direction control amount of the car	The upper 8 bits of the Y-direction control amount of the car	The lower 8 bits of the Y-direction control amount of the car	The upper 8 bits of the Z-direction control amount of the car	The lower 8 bits of the Z-direction control amount of the car	BBC Check Digit	OX7D

Note: Differential cars and ackerman cars do not support Y-axis movement control. Only mecanum-wheel cars and omni-wheel cars support Y-axis movement control. Download the serial port sending instruction routine provided by us to another single-chip microcomputer, connect the serial communication line between the single-chip microcomputer and the car controller, and the car enters the serial port control mode.

```
usart2_send(0X7B); //帧头
usart2_send(0X00); //预留位
usart2_send(0X00); //预留位
usart2_send(0X01); //X轴速度高8位
usart2_send(0X10); //X轴速度低8位
usart2_send(0X00); //Y轴速度高8位
usart2_send(0X00); //Y轴速度低8位
usart2_send(0X00); //Z轴速度高8位
usart2_send(0X00); //Z轴速度低8位
usart2_send(0X6A); //BBC校验位
usart2_send(0X7D); //帧尾
```

Figure 1-7-1 The control command sent by the serial port command routine to the car

Similarly, you can directly send data to the serial port 1 of the car through the host computer serial port assistant , as shown in Figure 1-7-2 .

Figure 1-7-2 The control command sent by the serial port assistant to the car

Serial port 1 and serial port 3 (ROS) control the same commands for receiving and controlling. For the calculation and conversion of the speed control amount of the car and the content of the data sent by the car, please refer to our section 1.2 ROS (serial port 3) control.

2. OLED display content

2. 1 OLED specific content

The robot is equipped with an OLED display, and the contents displayed on the display of different types of robots are similar, as follows:

① **Two wheel differential car**

```
TYPE: 1  BIAS  -3   Line 1:Robot model selection and gyroscope zero deviation value
GYRO Z  -      40   Line 2:the value of the Z-axis angular velocity
L:+      0 +    0   Line 3:Target and measured values of left motor
R:+      0 +    0   Line 4:Target and measured values of right motor
MA +   0 MB +   0   Line 5:Left and right motor PWM value
ROS  ON   23.03V    Line 6:Control mode   Enable switch   Battery voltage
```

Figure 2-1-1 Contents of the OLED display of the two-wheel differential car

② **Ackerman car**

```
TYPE: 1  BIAS  +17   Line 1:Robot model selection and Z-axis zero deviation value
GYRO Z  +       60   Line 2:the value of the Z-axis angular velocity
L:+      0 +     0   Line 3:Target and measured values of left motor
R:+      0 +     0   Line 4:Target and measured values of right motor
SERVO:  +        0   Line 5:PWM control value of steering gear
ROS  ON   23.03V     Line 6:Control mode   Enable switch   Battery voltage
```

Figure 2-1-2 Ackerman car OLED display content

③ **Mecanum wheel car**

```
TYPE: 1  GZ   +17   Line 1:Robot model selection and Z-axis angular velocity value
+        0 +    0   Line 2:Target value and measured value of A motor
+        0 +    0   Line 3:Target value and measured value of B motor
+        0 +    0   Line 4:Target value and measured value of C motor
+        0 +    0   Line 5:Target value and measured value of D motor
ROS   ON  23.03V    Line 6:Control mode   Enable switch   Battery voltage
```

Figure 2-1-3 Contents of the OLED display of the mecanum wheel trolle

④ **Omni wheel car**

TYPE: 1	GZ	+17	Line 1:Robot model selection and Z-axis angular velocity value
+	0 +	0	Line 2:Target value and measured value of A motor
+	0 +	0	Line 3:Target value and measured value of B motor
+	0 +	0	Line 4:Target value and measured value of C motor
MoveZ :	-	10	Line 5:Z-axis direction movement amount of the robot
ROS	ON	23.03V	Line 6:Control mode Enable switch Battery voltage

Figure 2-1-4 OLED display content of omni wheel trolley

2. 2 OLED general display content explanation

① TYPE

The “ TYPE ” displayed on the upper left corner of the display is the robot model. The robot model can be changed by twisting the potentiometer on the left corner of the STM32 controller. For the specific robot model for each product, please see Table 1-3. Different series of robots use different STM32 codes, and cars of the same series use the same STM32 code, and products in the same series are distinguished by robot model.

For example, the product I currently use is "two-wheel differential car (conventional)", then the potentiometer in the upper left corner of the STM32 controller should be turned to the value " 0 " after the first power-on, and then click the reset button on the controller to restart the STM32 controller. In this way, the car is successfully switched to the two-wheel differential car (conventional type) to run; similarly, if the product I use is "top mecanum-wheel pendulum suspension (conventional type) ", the TYPE value in the upper left corner should be " 3 ".

We will test the robot before shipment, so the robot model is selected by default when you get the product.

Note: The car will move a little distance after the initialization is completed to detect whether there is a problem with the movement of the car. If there is a problem, the car will not be controlled, and the upper right corner of the OLED display will display " TYPE : X ".

Table 2-1 Robot product model description

Model / series	0	1	2	3	4	5
Differential car series	Top version two-wheel differential car (conventional type)	Top version two-wheel differential car (heavy load type)	Four-wheel two-wheel drive Pendulum suspension (conventional type)	Four-wheel two-wheel drive Pendulum suspension (heavy load type)	Four-wheel two-wheel drive Independent suspension (conventional type)	Four-wheel two-wheel drive Independent suspension (heavy load type)
Ackerman car series	High version Ackerman car (conventional type)	High version Ackerman car (heavy load type)	Top version Ackerman car (conventional type)	Top version Ackerman car (heavy load type)		
Mecanum wheel car series	High version mecanum wheel without bearing seat	High version mecanum wheel Pendulum suspension	High version mecanum wheel Independent suspension	Top version mecanum wheel Pendulum suspension (conventional type)	Top version mecanum wheel Pendulum suspension (heavy load type)	Top version mecanum wheel Independent suspension (conventional type)
Omni wheel car series	High version Omni wheel (conventional type)	High version Omni wheel (heavy load type)	Top version Omni wheel (conventional type)	Top version Omni wheel (heavy load type)		

② Control mode

Different control modes correspond to different display contents in the lower left corner, see Table 2-2 for details .

Table 2-2 Robot OLED display display control mode

mode	ROS	APP	PS2 handle	Hot-RC remote control	CAN	Serial port
Display content	ROS	APP	PS2	R-C	CAN	USART

③ Enable switch

The enable switch is on the adapter board of the robot STM32 controller, and the robot motor can be controlled only when the enable switch is in the " ON " state. Taking into account the stability of the robot in the program , the motor is in a forced disable state within 10s after the STM32 controller is powered on . At this time, even if your enable switch is in the " ON " state, it will display " OFF ", and it will be updated to " ON " after 10s . It should be noted that the serial port 3 will not receive data within this 10s in the program .

3. Elimination of gyroscope zero drift

The IMU sensor is needed in the ROS navigation system. In our ROS robot system, the IMU sensor is integrated into the STM32 controller. The STM32 controller collects the IMU data and sends it to ROS . The IMU used on the STM32 moving chassis is the MPU6050, the IMU integrated three-axis gyroscope and a three-axis accelerometer. Gyroscopes cannot avoid the problem of zero drift, so a zero drift elimination mechanism is set in the program.

In the first 10 seconds of power-on , the gyroscope reads the angular velocity value without removing the zero drift value. At the 10th second, the program reads the current angular velocity value as the drift value. The gyroscope data read after 10 seconds will subtract the zero point offset. At this time, the LED light changes from constant light to flashing. The angular velocity value read in the subsequent reading process will subtract the zero point drift value, and the result is the angular velocity that eliminates the zero drift.

If you feel that the gyroscope zero drift value obtained in 10 seconds is not accurate enough, you can press the "user button" on the STM32 controller at any time to retrieve the gyroscope zero drift value.

The process of collecting gyroscope data described above is in the MPU6050.c file of the STM32 controller code, as shown in Figure 3-1 .

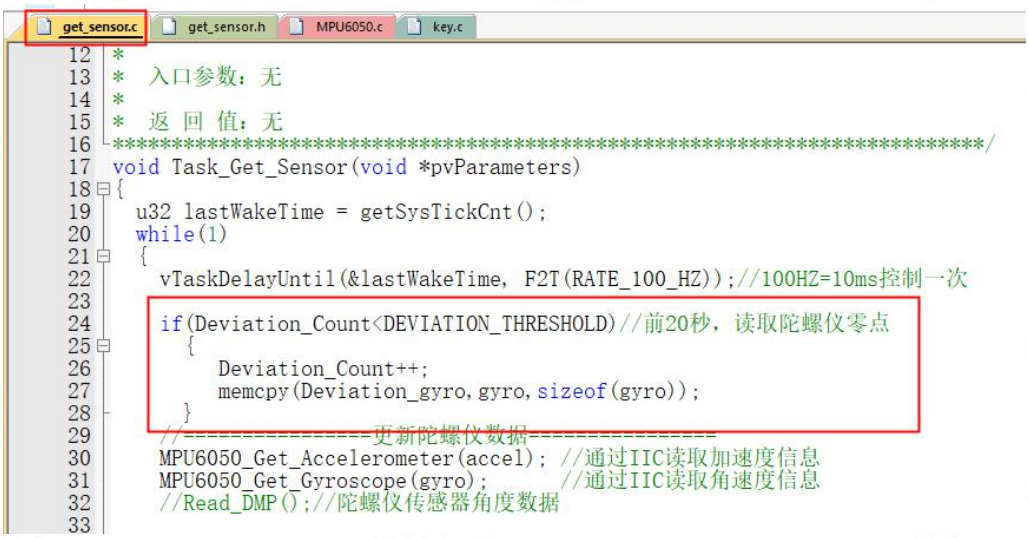

Figure 3-1 Collect gyroscope data

4. Robot kinematics analysis

In order to make the robot move, it is not enough to provide the target speed. The target speed of the robot needs to be converted to the actual target speed of each motor. Finally, the control of the motor is realized according to the target speed of the motor. The process of converting the target speed of the robot into the target speed of the motor is called “kinematic analysis”. Kinematics analysis is divided into forward and inverse solutions. Before kinematics analysis, let’s explain separately what are forward kinematics and inverse kinematics. :

① Positive kinematics solution: Calculate the speed of the robot in the X , Y and Z directions through the speed of each wheel of the robot .

②Kinematics inverse solution: Calculate the speed of each wheel of the robot by the speed of the robot's X- axis, Y- axis and Z- axis.

4. 1 Two-wheel differential car

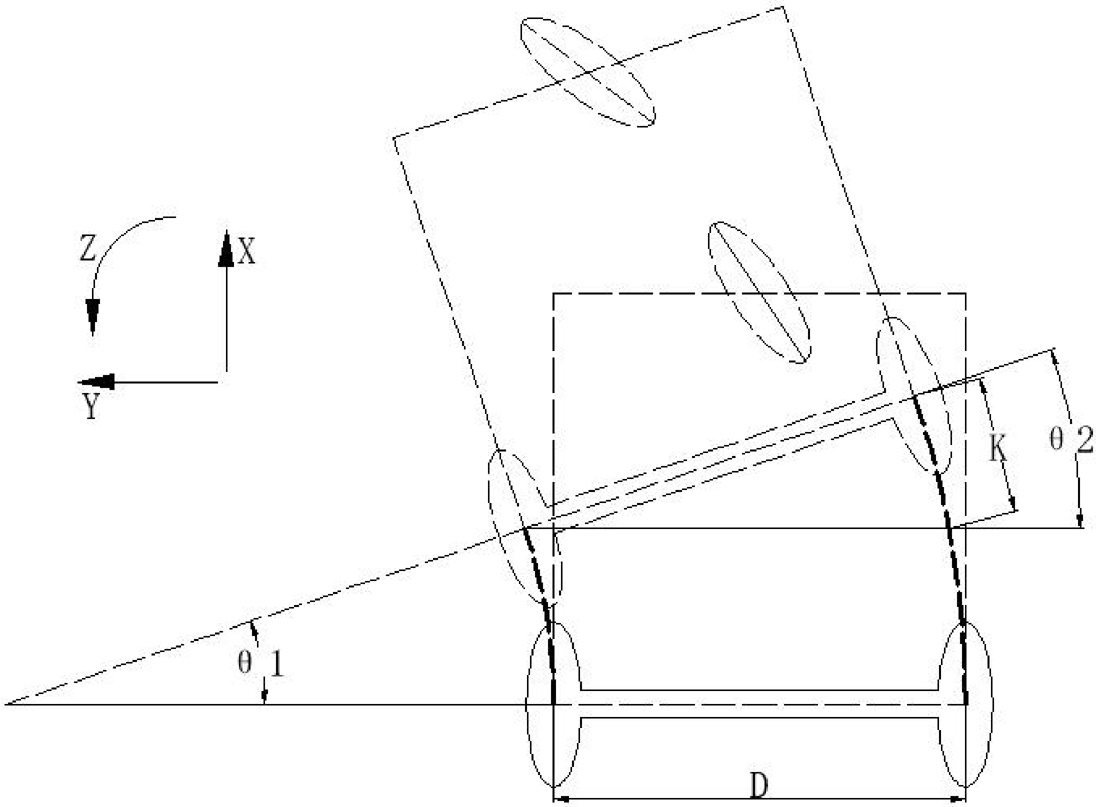

Figure 4-1 Kinematics model of two-wheel differential car

① kinematic analysis

The simplified motion model of the robot is shown in Figure 4-1, X-axis positive direction is forward, the Y-axis positive direction is the left translation, the Z-axis positive direction is counterclockwise (the following products are the same, will not be described) . The distance between the two wheels of the robot is D , the speeds of the X-axis and Z-axis of the robot are respectively: V_x and V_z , and the speeds of the left and right wheels of the robot are respectively: V_L and V_R .

Assuming that the robot travels in a left-front direction for a certain distance, the distance that the right wheel of the robot travels more than the left wheel is approximately K, and the point on the wheel of the robot is used as the reference point to extend the reference line, and then we get: $\theta_1 = \theta_2$.

Since this Δt is very small (10ms), the amount of angle change θ_1 is also very small, so there is an approximate formula:

$$\theta_2 \approx \sin(\theta_2) = \frac{K}{D}$$

From mathematical analysis, the following formula can be obtained:

$$K = (V_R - V_L) * \Delta t, \quad \omega = \frac{\theta_1}{\Delta t}$$

The result of the positive kinematics solution can be solved by the above formulas:

The speed in the X- axis direction of the robot is $V_X = \frac{V_L + V_R}{2}$, and the speed in the Z- axis direction of the robot is $V_Z = \frac{V_R - V_L}{D}$.

The result of the inverse kinematics solution is obtained directly from the positive solution:

The speed of the left wheel of the robot is $V_L = V_x - \frac{V_z * D}{2}$, the speed of the right wheel of the robot is $V_R = V_x + \frac{V_z * D}{2}$.

② C language implementation

There are two motors with encoders on the robot . We need to write the above motion relationships in C language, and then control the motors. code show as below:

```
void Drive_Motor(float vx,float vz)
{
    Target_Left  = vx-vz * WIDTH_OF_ROBOT / 2.0f; // Calculate the target speed of the left wheel
    Target_Right = vx+vz * WIDTH_OF_ROBOT / 2.0f; // Calculate the target speed of the right wheel
}
```

The above statement is to find the target speed of the two motors (inverse kinematics solution) through the speed of the X and Z axis of the robot , where WIDTH_OF_ROBOT is the macro definition of the linear distance between the two wheels.

4.2 Ackerman Car

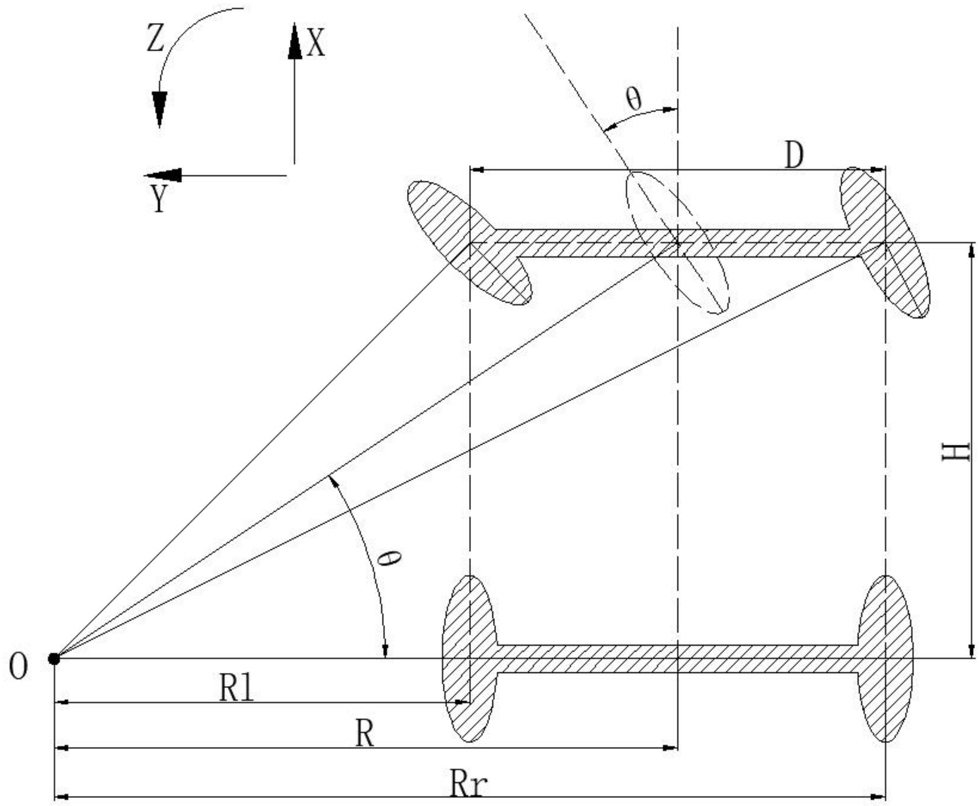

Figure 4 -2 Kinematics model of Ackerman car

① kinematic analysis

The difference between the ackerman car and the two-wheel differential car is that the front wheels of the two-wheel differential car are omnidirectional wheels or universal wheels, while the front wheels of the ackerman car are ordinary "one-way wheels". At this time, to realize pure rolling motion of ackerman car, it is necessary to ensure that the normal lines of the four wheels of the car intersect at one point, which is the steering center point, as shown in Figure. 4-2, point O.

To simplify the model, suppose that the front wheel has only one wheel (the realization theory is the same) , which is located in the middle of the front axle, as shown in the front wheel depicted by the dotted line in Figure 4-2 .

As seen from the ackerman car kinematics model, when the current wheel Angle is θ , the car steering radius is R. Let the forward speed of the car be V (i.e. V_x ,) , the left wheel velocity is V_L and right wheel velocity is V_R , the angular velocity can be obtained by the consistency that: $\frac{V}{R}=\frac{V_L}{R_L}=\frac{V_R}{R_R}$,and that $\frac{H}{R}=\tan\theta$, $R_L = R - \frac{D}{2}$, $R_R = R + \frac{D}{2}$ .Then , $V_L = \frac{V}{R}R_L = V*(1-\frac{D*\tan\theta}{2H})$, $V_R = \frac{V}{R}R_R = V*(1+\frac{D*\tan\theta}{2H})$.

② C language implementation

```
void Drive_Motor(float vx,float vz)
{
    int K=1130,amplitude=2;// speed limit
    Angle=target_limit_float(Angle,-0.38f,0.38f);// Servo angle limit
    if(Angle<0)Ratio=0.83f;// Symmetry processing on the left and right sides of
the steering gear
    else if(Angle>0)Ratio=1.11f;
    else Ratio=0;

    MOTOR_A.Target=vx*(1-Wheel_axlespacing*tan(Angle)/2/Wheel_spacing);//A
( left ) motor target control value
    MOTOR_B.Target=vx*(1+Wheel_axlespacing*tan(Angle)/2/Wheel_spacing);//B
( right ) motor target control value
    MOTOR_A.Target=target_limit_float(MOTOR_A.Target,-amplitude,amplitude);//
Motor limit
    MOTOR_B.Target=target_limit_float(MOTOR_B.Target,-amplitude,amplitude);//
```

```
Motor limit

Servo=SERVO_INIT+Angle*K*Ratio;// Servo target
}
```

The function input is X-axis speed and front wheel angle, K and Ratio are the correction coefficients of the steering gear, Wheel_axlespacing is the wheelbase parameter of the car (front and rear) , and Wheel_spacing is the tread parameter of the car (left and right) .

4. 3 Mecanum wheel car

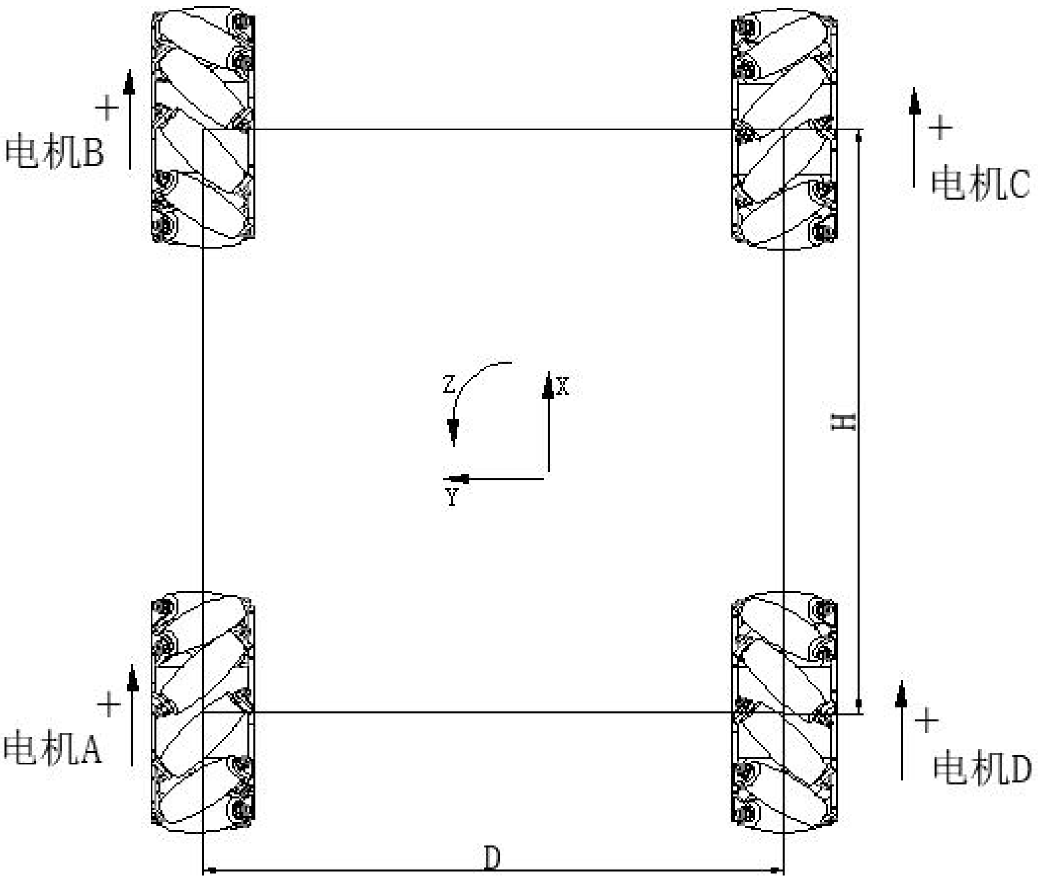

Figure 4 -3 Kinematics model of mecanum wheel car

① kinematic analysis

To simplify the mathematical model of kinematics, the following two idealized assumptions are made:

(1) The omni wheel does not slip with the ground, and the ground has sufficient friction;

(2) The 4 wheels are distributed on the 4 corners of the rectangle or square, and the wheels are parallel to each other.

Here we linearly decompose the rigid body motion of the car into three components, then only need to calculate the speed of the four wheels when the mecanum-wheel chassis is translated in the X+, Y+ direction and the Z+ direction is rotated, and then the formula can be combined to calculate the rotational speed of the four wheels required for the "translation + rotation" movement synthesized by these three simple movements .

Wherein , V_A, V_B, V_C, V_D respectively, A , B , C , D four wheel rotational speed, i.e. the rotational speed of the motor; V_x is the translation speed of the car along the X axis, V_y is the translation speed of the car along the Y axis, and ω is the rotation speed of the car along the Z axis; $a=\frac{D}{2}$ is half of the tread D of the car, and $b=\frac{H}{2}$ is half of the wheelbase H of the car.

When the car moves along the X axis:

$V_A=+V_x$ $\quad V_B=+V_x$ $\quad V_C=+V_x$ $\quad V_D=+V_x$

When the car moves along the Y axis:

$V_A=+V_y$ $\quad V_B=-V_y$ $\quad V_C=+V_y$ $\quad V_D=-V_y$

When the car rotates around the geometric center:

$V_A=-\omega(a+b)$ $\quad V_B=-\omega(a+b)$ $\quad V_C=+\omega(a+b)$ $\quad V_D=+\omega(a+b)$

Combine the above three sets of equations we can calculate the speed of the four wheels according to the motion state of the car:

$$V_A=+V_x+V_y-\omega(a+b)$$

$$V_B=+V_x-V_y-\omega(a+b)$$

$$V_C=+V_x+V_y+\omega(a+b)$$

$$V_D=+V_x-V_y+\omega(a+b)$$

② C language implementation

```
void Drive_Motor(float vx,float vy,float vz)
{
    MotorTarget.A    = (+vx+vy-vz*(Wheel_spacing+Wheel_axlespacing));
    MotorTarget.B    = (+vx-vy-vz*(Wheel_spacing+Wheel_axlespacing));
    MotorTarget.C    = (+vx+vy+vz*(Wheel_spacing+Wheel_axlespacing));
    MotorTarget.D    = (+vx-vy+vz*(Wheel_spacing+Wheel_axlespacing));
}
```

Wheel_axlespacing is the wheelbase parameter of the car (front and rear) , and Wheel_spacing is the tread parameter of the car (left and right) .

4. 4 Omni wheel car

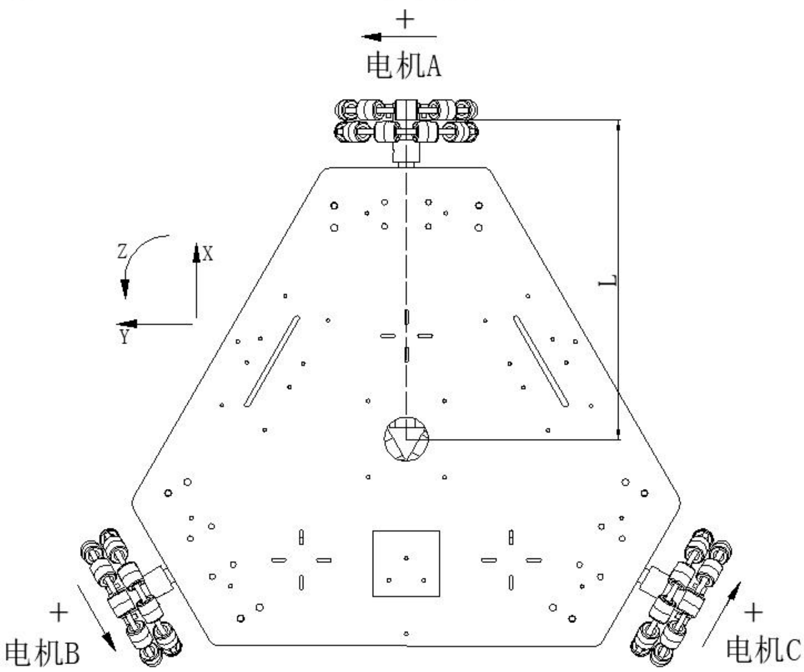

Figure 4 -4 Kinematics model of omni wheel car

① kinematic analysis

Before kinematic modeling, in order to simplify the mathematical model of kinematics, the following idealized assumptions are made:

(1) The omni wheel does not slip with the ground, and the ground has sufficient friction;

(2) The axis center of the motor is exactly the center of gravity of the chassis;

(3) Each wheel is installed at 120 ° absolutely mutually .

Through simple speed decomposition, the following formula can be obtained:

$$V_A = V_Y + L_\omega$$

$$V_B = -Cos30°V_x - Sin30°V_Y + L_\omega$$

$$V_C = +Cos30°V_x - Sin30°V_Y + L_\omega$$

That is , $V_A = V_Y + L_\omega$

$$V_B = -\frac{\sqrt{3}}{2}V_x - \frac{1}{2}V_Y + L_\omega,$$

$$V_C = +\frac{\sqrt{3}}{2}V_x - \frac{1}{2}V_Y + L_\omega$$

ω is the angular velocity of the robot,L is the distance between the center of omnidirectional wheel and the center of chassis, and V_A , V_B , V_C are the rotational speeds of the three wheels respectively, Vx and Vy are the motion speeds of the robot in X and Y directions.

② C language implementation

```
void Drive_Motor(float vx,float vy,float vz)
{
    MotorTarget.A    = +vy+vz*Parament.Z;
    MotorTarget.B    = -vx*Parament.X-vy*Parament.Y+vz*Parament.Z;
    MotorTarget.C    = +vx*Parament.X-vy*Parament.Y-vz*Parament.Z;
}
```

$Parament.X = \frac{\sqrt{3}}{2}$, $Parament.Y = \frac{1}{2}$, $Parament.Z = 1$, corresponding to our kinematic analysis parameters.

4. 5 PI control program source code

What is obtained through kinematic analysis is the target speed of the motor. We need to send this target value to the PID controller for speed closed-loop control, so that the actual output speed of the motor approaches the target value. The PI controller source code in the program is as follows:

```
/************************************************** *************************
Function performance: Incremental PI controller
Entry parameters: encoder measurement value, target speed
```

```
Return value: Motor PWM
According to the incremental discrete PID formula
pwm+=Kp[e ( k ) -e(k-1)]+Ki*e(k)+Kd[e(k)-2e(k-1)+e(k-2)]
e(k) represents this deviation
e(k-1) represents the deviation of the last time   so on
pwm stands for incremental output. In our speed control closed-loop system, only
PI control is used
pwm+=Kp[e ( k ) -e(k-1)]+Ki*e(k)
************************************************** ************************/
int Incremental_PI_A (int Encoder,int Target)
{
static int Bias,Pwm,Last_bias;
Bias=Encoder-Target;                    // Calculate the deviation
Pwm+=Velocity_KP*(Bias-Last_bias)+Velocity_KI*Bias; // Incremental PI controller
if(Pwm>7200)Pwm=7200;
if(Pwm<-7200)Pwm=-7200;
Last_bias=Bias;                         //Save the last bias
return Pwm;                             //Incremental output
}
```

5. Wiring instructions

The STM32 controller integrates three 5V power supplies: one 5V power supply supplies power to the STM32 controller and peripherals (encoders, bluetooth, handles, etc.), one 5V power output supplies power to the Raspberry Pi (TYPE-C interface), and the other 5V power output supplies power to slamtec A2/A3 radar (TYPE-A interface).

This chapter mainly demonstrates several key wiring instructions. Please see the diagram directly for specific wiring.

5. 1 Power supply for Raspberry Pi

The 5V power circuit of the Raspberry Pi is integrated on the adapter board of the STM32 controller, and it uses a TYPE-C to TYPE-C cable that can pass a current of more than 3A .

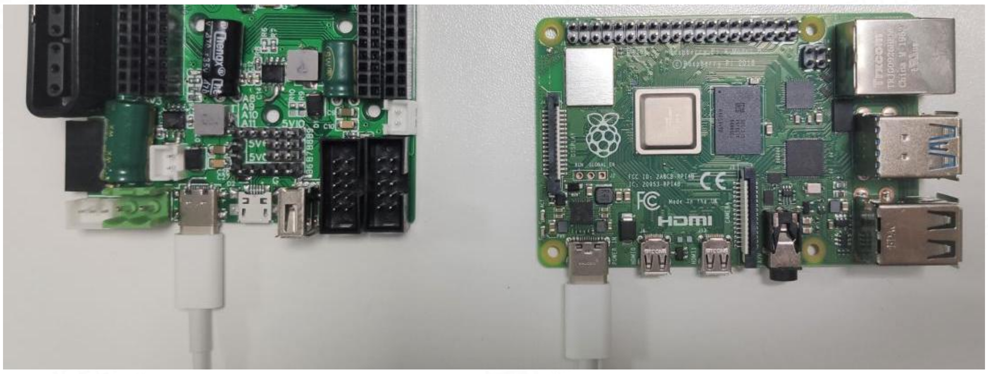

Figure 5-1-1 Raspberry Pi power supply wiring

5. 2 Raspberry Pi and STM32 controller serial communication

Because the Raspberry Pi is used as the host computer to communicate with the STM32 controller here, the communication between the two requires level conversion. The serial port 3 of the STM32 controller has integrated the CP2102 level conversion chip in it, so here only needs to be connected with ordinary Mirco USB cable.

Figure 5-1-2 Raspberry Pi and STM32 communication

5.3 Connect Raspberry Pi to Slamtec A1 radar

The connection between the Raspberry Pi and the radar here is a normal Mirco USB cable. The Raspberry Pi also communicates with the lidar while powering the lidar, so no additional power supply is required.

Figure 5-1-3 Radar wiring for Raspberry Pi and slamtec A1 radar

5.4 Connect Raspberry Pi to slamtec A2/A3 radar

Unlike the A1 radar, because the brushless motor requires a larger starting current, the A2/A3 radar adapter board will have an additional power supply interface.

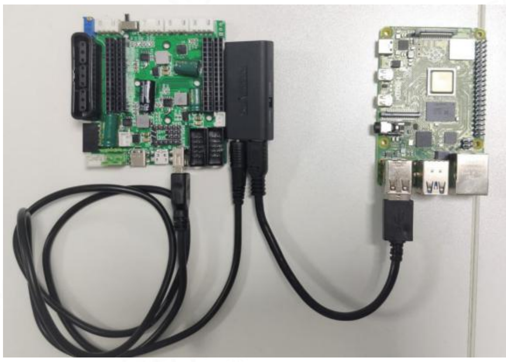

Figure 5-1-4 Raspberry Pi and slamtec A2/A3 radar wiring

6. Control flow chart

6.1 Control flowchart of robot motor

The robot supports 6 control modes, and the principles of these 6 control modes are realized by changing the target speed of the robot. The target speed obtains the actual output of each motor through the kinematic analysis function, and finally realizes the speed control of the motor through the PID controller (PID speed control function).

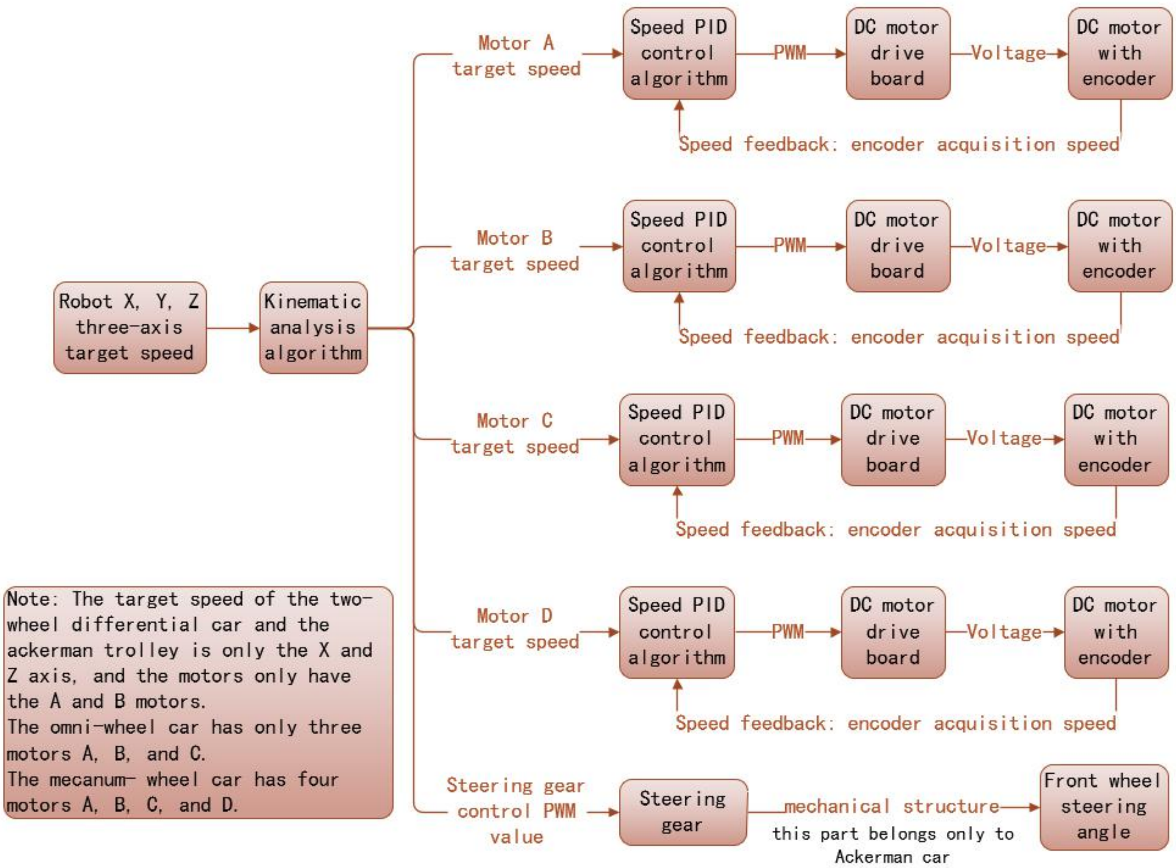

Figure 6-1 Robot motor control flow chart

6. 2 Robot STM32 program structure diagram

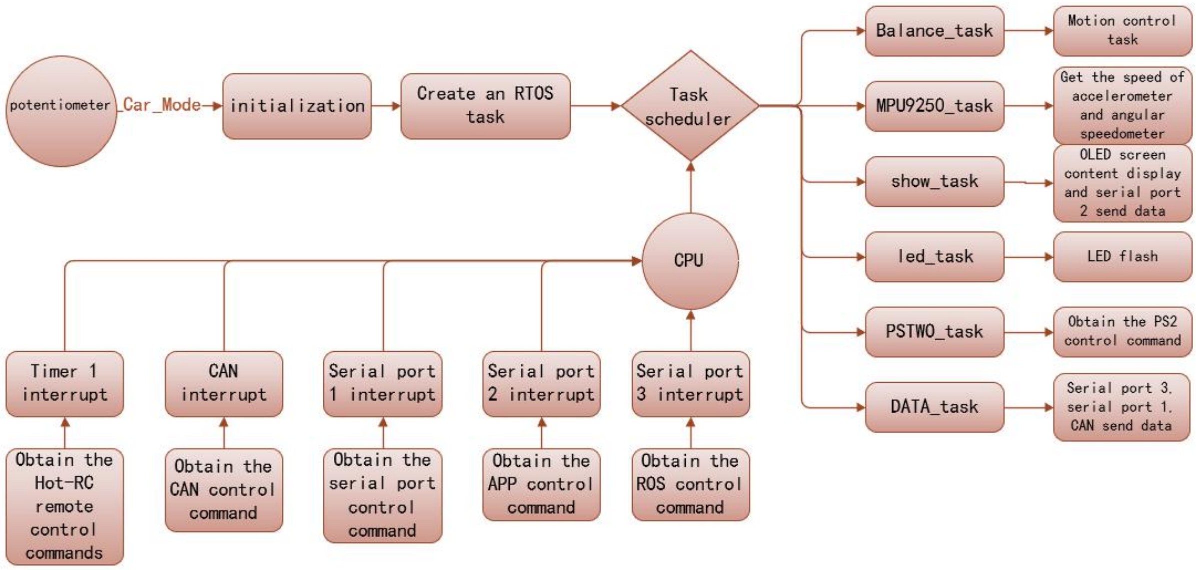

Figure 6-2 The program execution flowchart of the robot STM32 controller

The RTOS task scheduler determines the execution order of tasks according to the priority of the task (the task order in Figure 6-2 does not represent the task priority, the specific task priority needs to check the priority setting in the program), and the execution time of each task is very short, so it is almost equivalent to executing all tasks at the same time. If an interrupt occurs during this period, it will respond to the interrupt. The serial port 2 interrupt is used for APP bluetooth control, and the serial port 3 interrupt is used to receive information from ROS .

Car_Mode determines the polarity of the PWM output control motor, which has been set before leaving the factory, and the user does not need to change it.

6. 3 Robot controller connection diagram

Many controllers and peripherals are used in the robot, including: Raspberry Pi (Jetson Nano , Jetson TX2 , industrial computer, etc.), laser radar, STM32 controller, motor, encoder, dual drive, bluetooth, PS2 wired handle, hot-RC remote control, gyroscope, etc., while providing serial port 1 and CAN interface to facilitate users to expand control, the connection between these controllers and controllers, peripherals and controllers, as shown in Figure 6-3 .

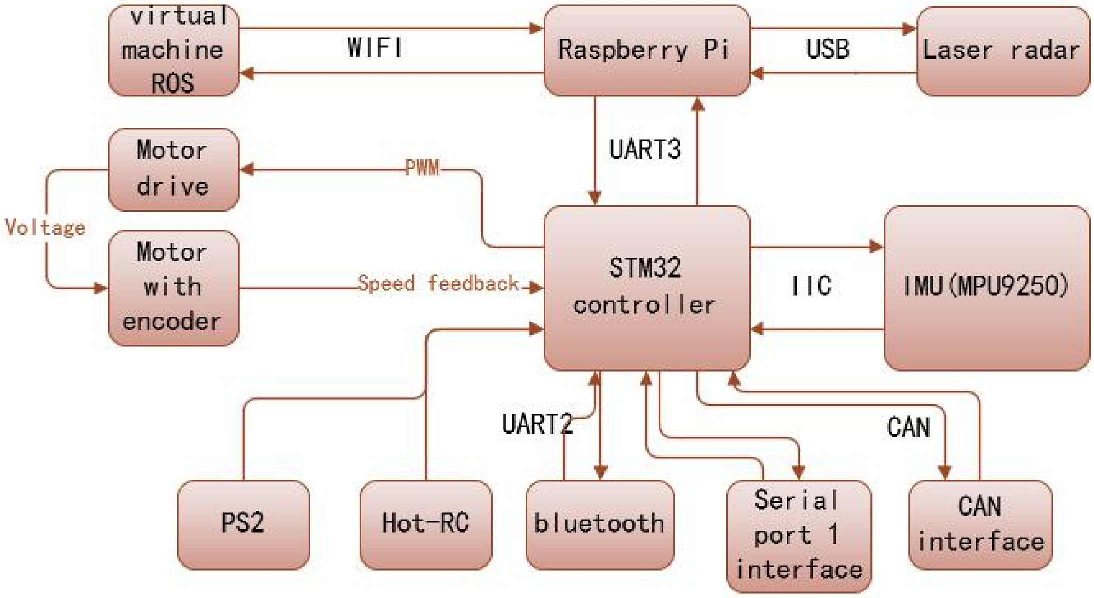

Figure 6-3　Robot controller connection diagram

\

7. Matters needing attention

7. 1 About the code

The programming method on the robot STM32 controller is based on the RTOS system, which is different from the interrupt control method. The RTOS is executed in the form of tasks in turn, and tasks with higher priority have higher execution priority (interrupt priority is higher than task priority). It should be noted that if a task has an execution logic error, the program will be stuck at the error and cannot continue. For example: If there is a sending task of serial port 3 in the program , but the serial port 3 is not initialized or the initialization code is wrong, the program will get stuck when executing the sending task of serial port 3 . Therefore, if the program is stuck during debugging and cannot be executed normally, you need to check whether there is a bug stuck on a task by task .

7. 2 About the power interface on the adapter board

The 5V and 3.3V pins on the adapter board can supply external power, but they cannot carry loads with too much current. Among them, 5V output is not recommended to carry a load exceeding 1.5A , and 3.3V output is not recommended to carry a load exceeding 200mA . As you can see from the schematic diagram of the adapter board provided by us, the adapter board is equipped with two 5V power circuits, one of which supplies power to the basic peripherals, and the other independently supplies power to the Raspberry Pi (Jetson Nano) (USB female socket). Wiring instructions are explained in detail in Chapter 5 " 5.Wiring Instructions".

7. 3 About the motor

Avoid blocking the motor during use, otherwise it is easy to burn the motherboard. Before the robot is completely tested and passed, please stand up the robot and let the motor hang in the air to avoid unnecessary damage caused by misoperation.

7. 4 About the battery

The voltage of the battery is displayed on the display. When the battery is low (less than 21V), please charge it in time. The battery comes with over-discharge and over-charge protection. Please do not use the battery when charging the battery. The battery charging connection is shown in Figure 7-4-1 .

Figure 7-4-1 Wiring diagram of robot battery and charger

After the battery is charged, the cover of the charging port should be closed to prevent the battery from being disconnected by accidental touch, as shown in Figure 7-4-2 .

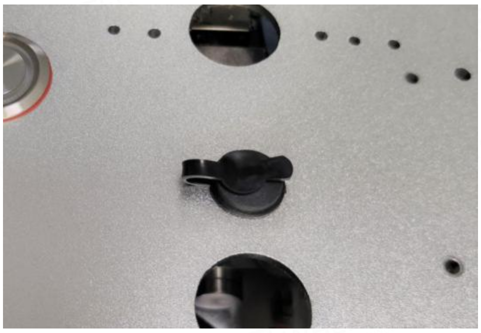

Figure 7-4-2 Robot battery charging cable interface

8. How to download program to STM32 controller

The STM32 controller can download the program through the serial port or the SWD interface. The serial port is downloaded via the USB data cable, and it is given by default. The SWD interface recommends using the STlink with the metal shell to download, and you need to bring your own.

8. 1 Serial download

The motherboard uses a one-click download circuit, which is very convenient for downloading programs. Just a MicroUSB data cable of mobile phone is ok.

① Hardware preparation

hardware:

1.STM32 controller

2. MicroUSB mobile phone data cable

② Software preparation

Software: FlyMcu recovering software (included in the attached information), the corresponding USB to TTL module CH340G driver. There are also drivers in the attached materials. If the driver installation is really difficult, download a driver wizard.

After the installation is successful, you can open the device manager to see, you can see that the driver has been installed successfully, otherwise there will be a red exclamation mark.

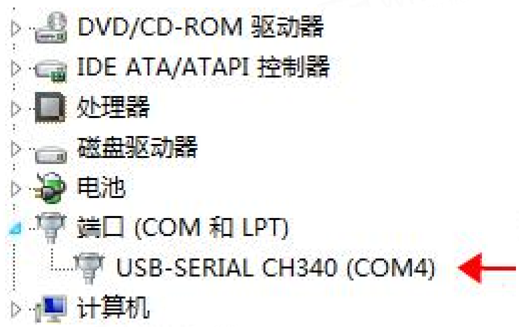

Figure 8-1-1 Device manager view CH340 driver

The wiring of the serial port download program is very simple, just connect the data cable to the computer and the board. Open the FlyMcu software in the attached document and set it according to the operation sequence in Figure 8-1-2 .

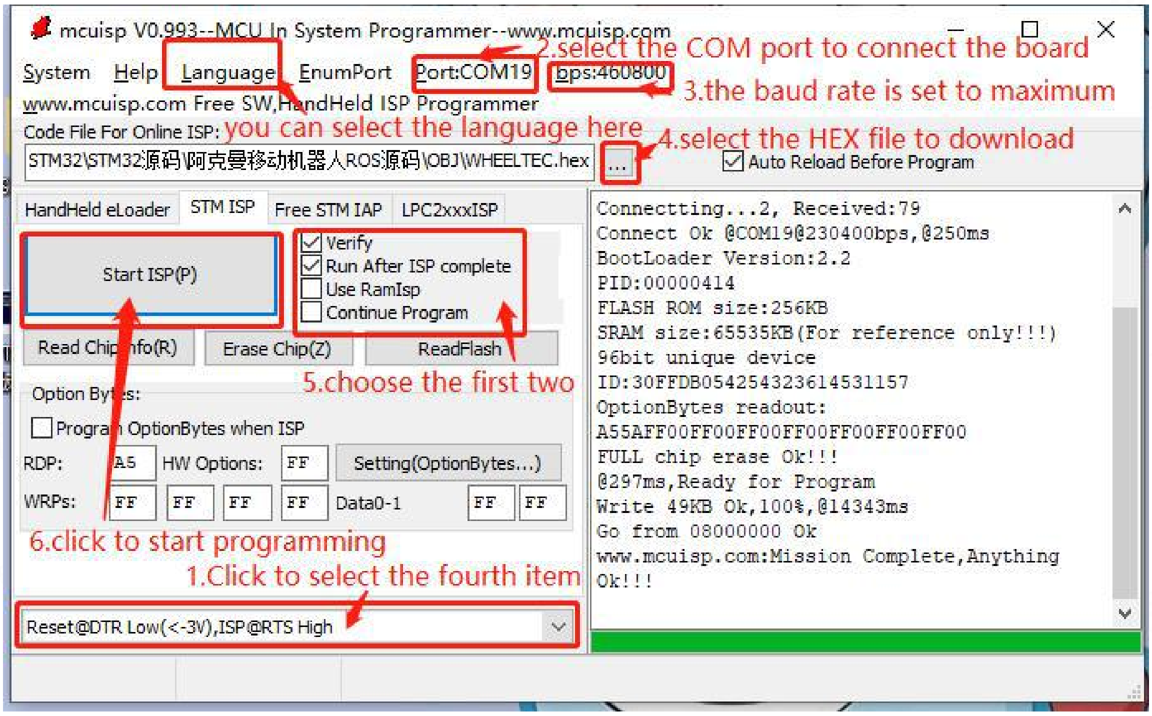

Figure 8-1-2 FlyMcu download program configuration description

OK, everything is ready, and then click to start programming, the program can be downloaded. Because the option to execute after programming is checked, the program will run automatically after downloading. (Note: It is forbidden to check the option of write option byte when programming to FLASH . If you are using an F4 board, the baud rate needs to be set to 76800)

8. 2 SWD download

The STM32 controller can download the program through the SWD interface, , there are logos on the motherboard, namely PA13 and PA14..

① **Hardware preparation**

1. STM32 controller

2.STlink

② **Software preparation**

Install the corresponding STlink or Jlink driver.

After the installation is successful, you can open the device manager to see if the STLink device is displayed again .

Figure 8-2-1　Device manager view STlink driver

You can see that the driver has been successfully installed!

③ Wiring

STlink ----------------- STM32 controller

SWDIO-----------------PA13

SWCLK-----------------PA14

GND--------------------GND

OK, everything is ready.

④ program

Click the button pointed by the arrow in Figure 8-2-2 , and the program can be downloaded! Because the option to execute after programming is checked, the program will run automatically after downloading. The default program configuration is for STlink , if you need to configure Jlink download, you need to modify the MDK settings.

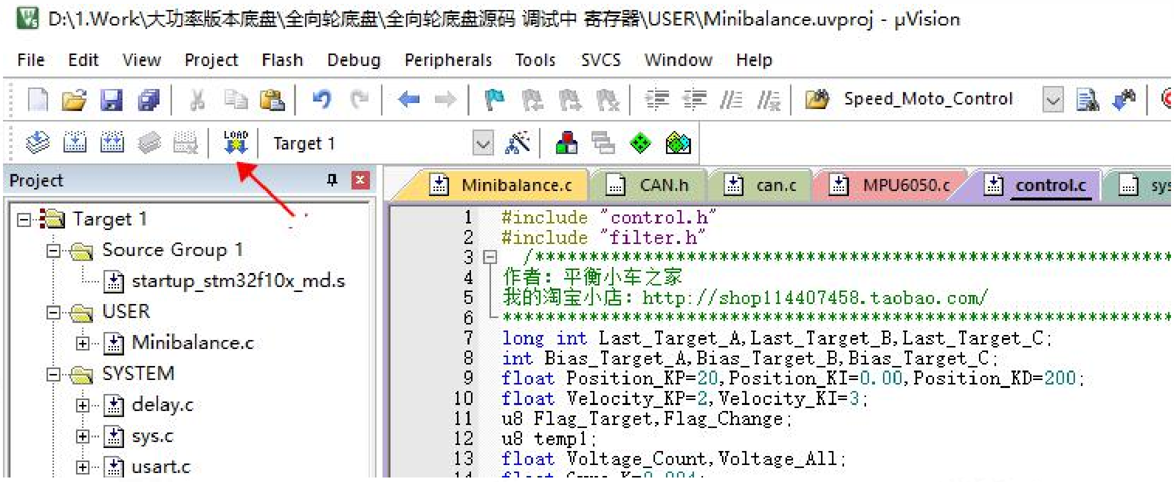

Figure 8-2-2 STlink download program interface