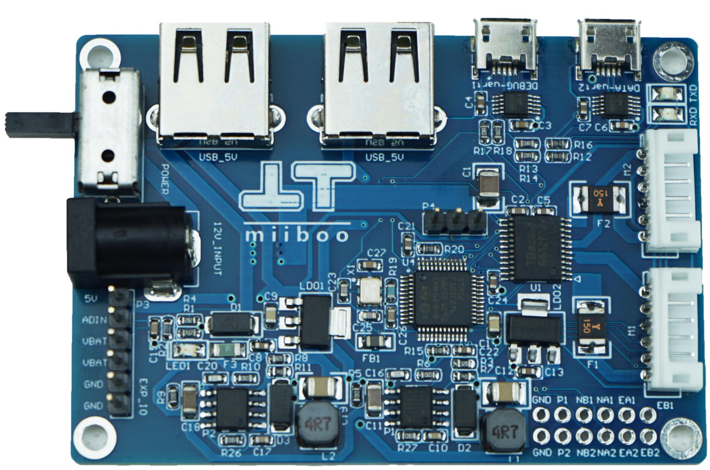
miiboo

miiboo robot

Build Your Dream

Motor driver board User guide

V 3.0



【About us】

website	***www.miiboo.cn***
github	***www.github.com/miiboo/***
amazon	***www.amazon.com/dp/B07X2HQ23D***
email	***robot4miiboo@163.com***

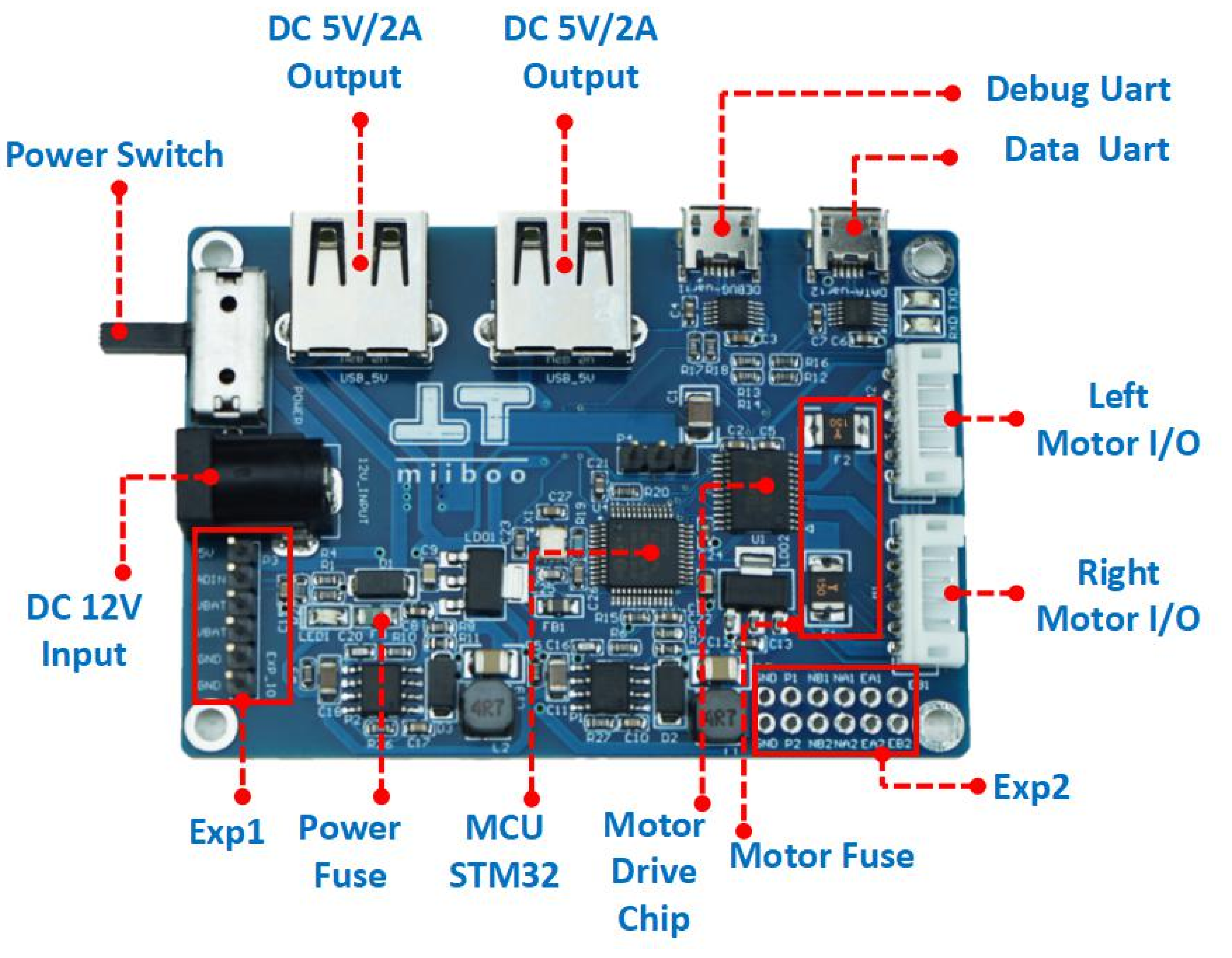
Contents

1.Introduction

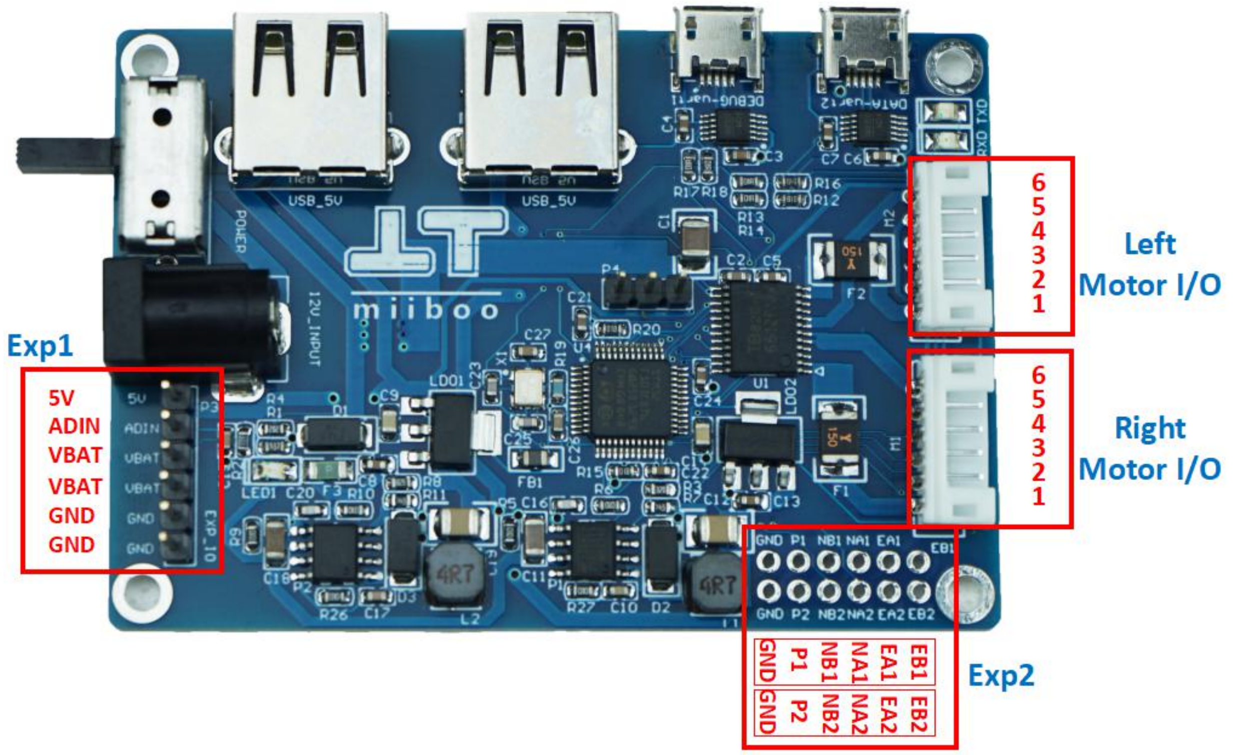
miiboo motor driver board is a low cost , highly integrated and well functional motor controll solution provided by *miiboo robot.* Our product can be used for ROS robot differential chassis motor control system, intellegent remote car controll system and encoded motor controll system. It not only support default GM37 coded decelletrate geared motor, but its onboarded IOextension pin also supports high-power coded motors. We also provide ROS driver program to help user control motro

2.Hardware interface definition

2.1. Interface



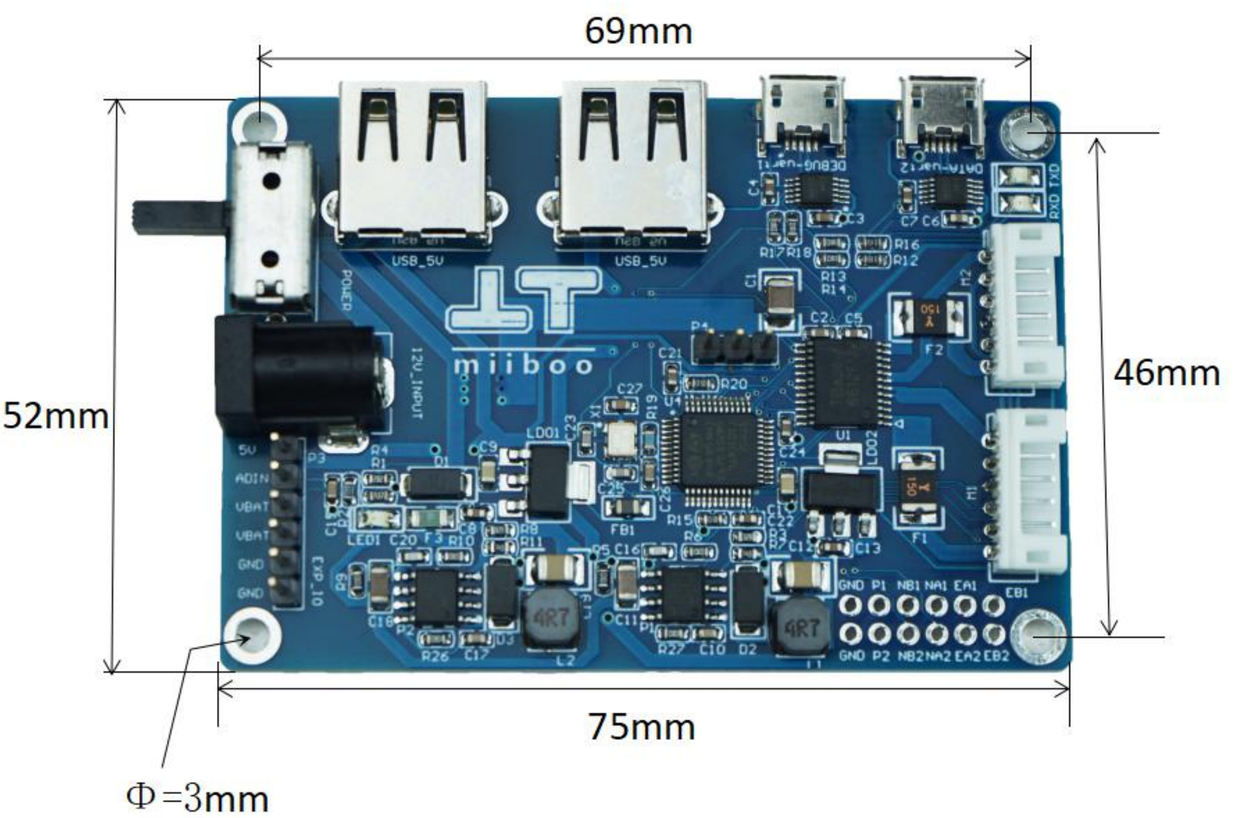
2.2. Electrical characteristics



Name	Interface type	Typical value	Note
DC 12V Input	DC5.5*2.1 round	typical value=12V,minimun value=9V, maximun value=12.6V	Power Input
DC 5V/2A Output	USB type-A	typical value=5V/2A	5V power output
DC 5V/2A Output	USB type-A	typical value=5V/2A	5V power output
Debug Uart	microUSB	CH340 port	COM for debugging
Data Uart	microUSB	CH340 port	COM for Data transmission
Left Motor I/O	PH2.0-6P	1: motor supply+ （12V motor） 2: Coder GND 3: Plause-code A （3.3V/5V） 4: Plause-codeB （3.3V/5V） 5: CoderVCC （5V） 6: motor supply- （12V motor）	left GM37 encode motor interface
Right Motor I/O	PH2.0-6P	1: motor supply+ （12V motor） 2: Coder GND 3: Plause-code A （3.3V/5V） 4: Plause-codeB （3.3V/5V） 5: CoderVCC （5V） 6: motor supply- （12Vmotor）	right GM37 encode motor interface

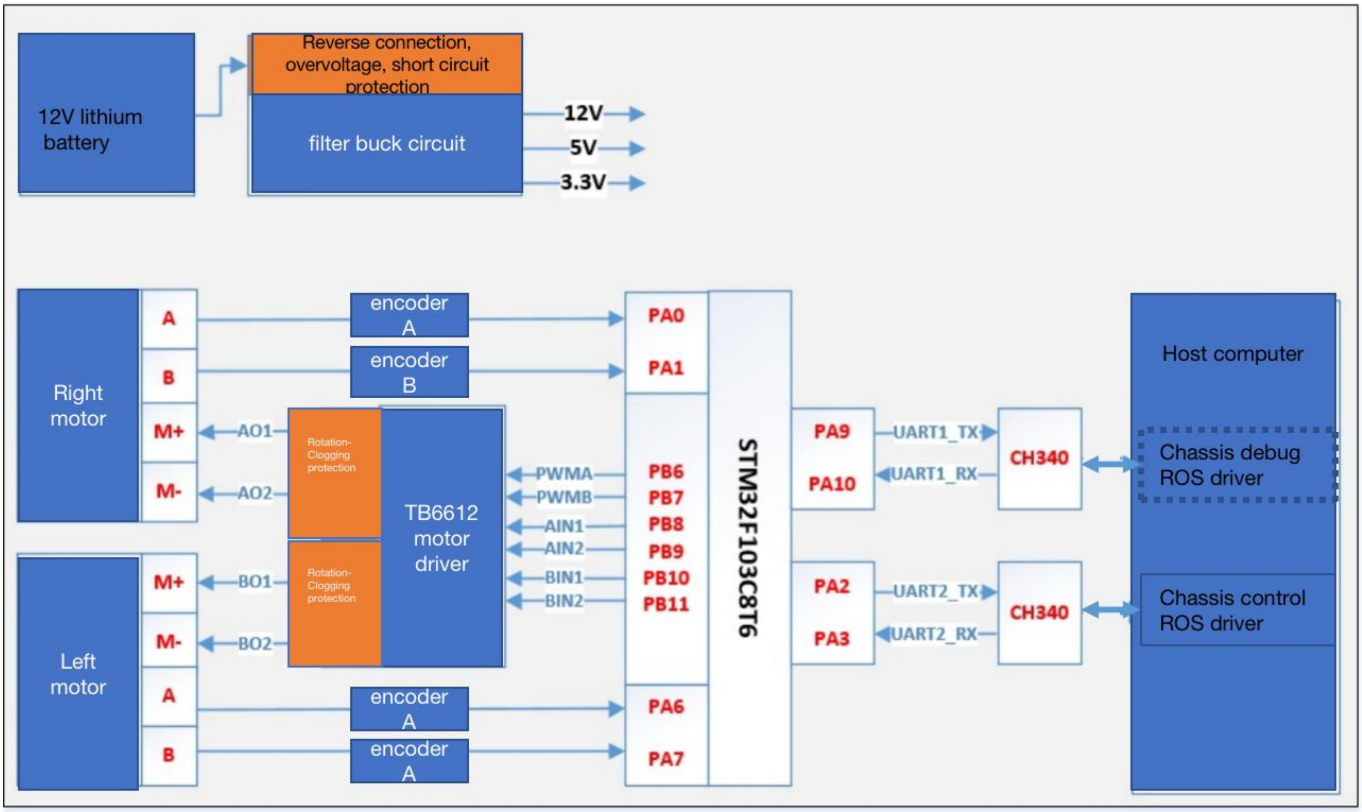
Name	interface type	Typical value	Note
Exp1	6Pin I/O 2.56mm	5V: 5V power output ADIN: perserved VBAT: total power valtege VBAT: total power valtege GND:power ground GND: power ground	power extension interface
Exp2	6*2Pin I/O 2.56mm	GND: power ground P1: PWM speed control signal （3.3V） NB1: Direction controll interface （3.3V） NA1: Direction controll signal （3.3V） EA1: Plause-code A （3.3V） EB1: Plause-code B （3.3V） GND: power ground P1: PWM speed control signal （3.3V） NB1: Direction controll interface （3.3V） NA1: Direction controll signal （3.3V） EA1: Plause-code A （3.3V） EB1: Plause-code B （3.3V）	large valtege extension interface for motor

2.3. Mechanical parameters



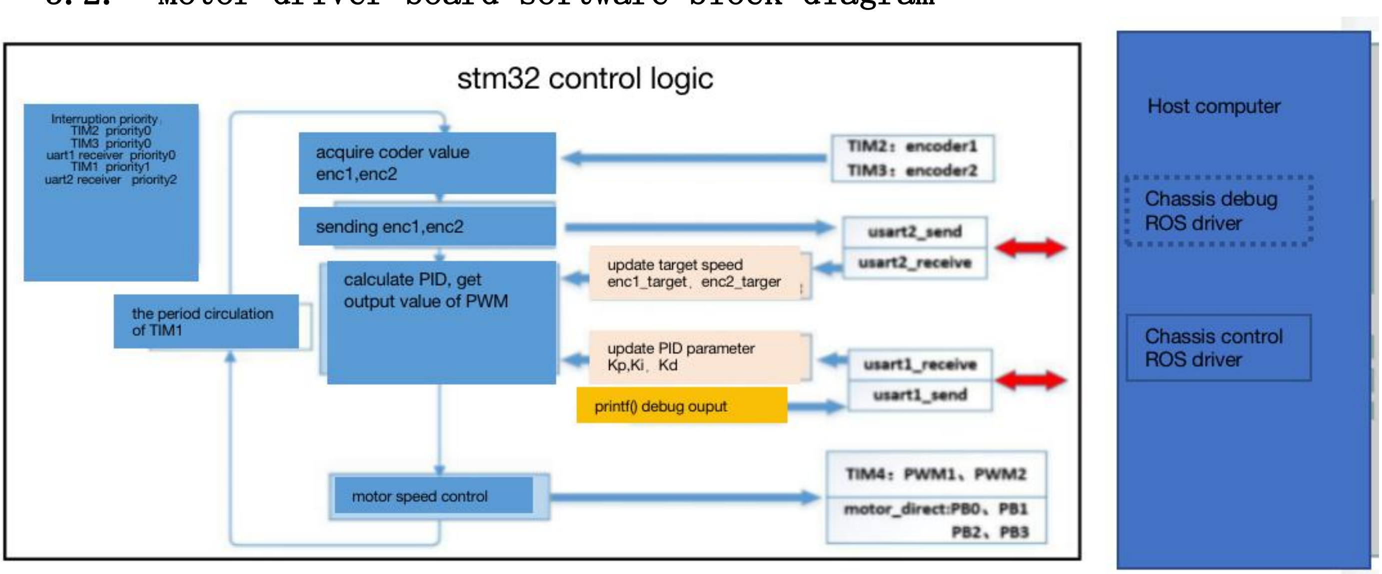
3.Function analysis

3.1. Hardware diagram of motor driver board



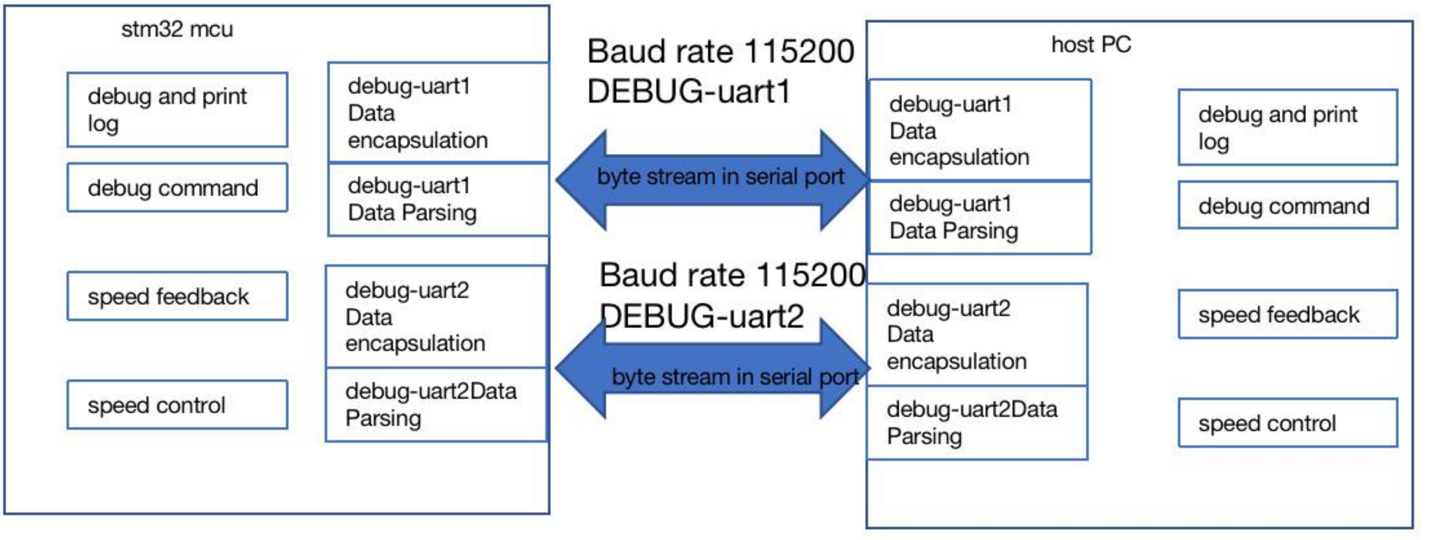
The miiboo motor drive board is equipped with STM32 MCU and TB6612 motor driver chip. It can also run PID algorithm to realize speed closed-loop control of DC coded decelerating motor. Receive motion control instructions from the host computer through serial communication, and feed back motor code mileage information. The onboard DC-DC module provides dual 5V/2A power supply interface. The total power input provides short-circuit, over-voltage protection, motor blocking protection, so that the drive board can work more stably.

3.2. Motor driver board software block diagram



The big loop executes PID motor control. The control process includes reading of coded feedback signals, calculating PID, and outputting PWM to control the motor speed. The motor drive board communicates with the host computer through the serial port, receives the control commands sent by the host computer, and feeds back mileage information。

3.3. Comminication protocol



The communication protocol consists of two parts: DEBUG-uart1 and DATA-uart2. DEBUG-uart1 is used to transmit debugging print information and debugging commands between stam32 and the upper computer; DATA-uart2 is used to transmit speed feedback and speed control between stam32 and the upper computer. And DEBUG-uart1 and DATA-uart2 both use the baud rate 115200 for data transmission.

Speed feedback(stm32 MCU ==> Host PC))

top		enc1_sig	enc1_val			enc2_sig	enc2_val			checksum
ff	ff	xx	xx	xx	xx	xx	xx	xx	xx	xx

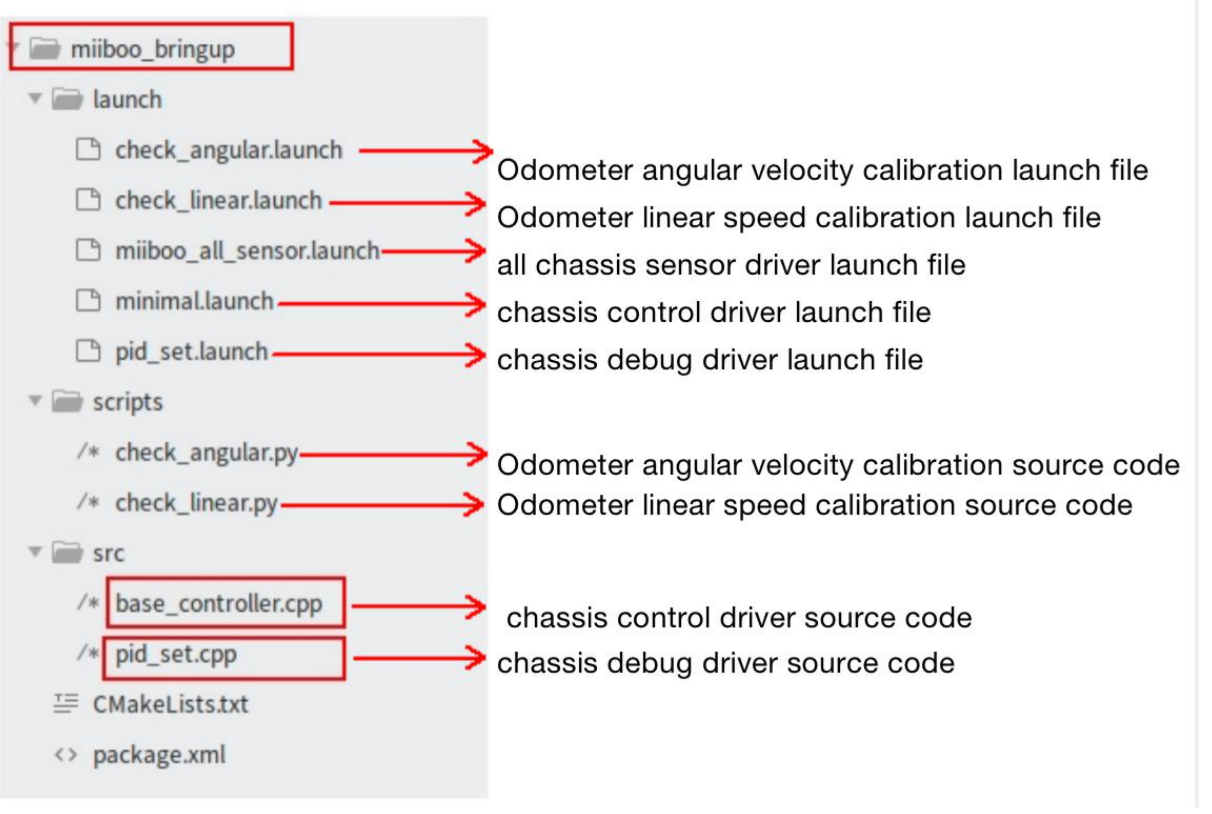
The data frame consists of 11 bytes. As shown in the table above, from left to right is defined as:
top[0],top[1]: frame header, fixed value ff ff
enc1_sig: left wheel speed symbol,When the velocity is negative, the value is 0, otherwise is none 0
enc_val: left wheel speed, it represent high, middle and low position successively. Put it together tis 24 bit positive number
enc2_sig: right wheel speed symbol,When the velocity is negative, the value is 0, otherwise is none 0
enc2_val: right wheel speed, it represent high, middle and low position successively. Put it together tis 24 bit positive number
checksum: Add all the previous bytes and take the lower 8 bits

Speed control (stm32 MCU <== Host PC))

top		enc1_sig	enc1_val			enc2_sig	enc2_val			checksum
ff	ff	xx	xx	xx	xx	xx	xx	xx	xx	xx

The data frame consists of 11 bytes. As shown in the table above, from left to right is defined as:
top[0],top[1]: frame header, fixed value ff ff
enc1_sig: left wheel speed symbol,When the velocity is negative, the value is 0, otherwise is none 0
enc_val: left wheel speed, it represent high, middle and low position successively. Put it together tis 24 bit positive number
enc2_sig: right wheel speed symbol,When the velocity is negative, the value is 0, otherwise is none 0
enc2_val: right wheel speed, it represent high, middle and low position successively. Put it together tis 24 bit positive number
checksum: Add all the previous bytes and take the lower 8 bits

3.4. Host computer ROS driver



The user can run the *miiboo* ROS driver program on the host computer to operate and set the motor. The host computer can be a PC computer with ROS system installed, an industrial computer, or ARM host such as a Raspberry Pi, etc. The miiboo motor driver board need to be connected with the host computer via a USB cable and communicate via uart(COM). Our program includes the functions of chassis motor control, odometer calculation, PID parameter setting, odometer linear velocity and angular velocity calibration.

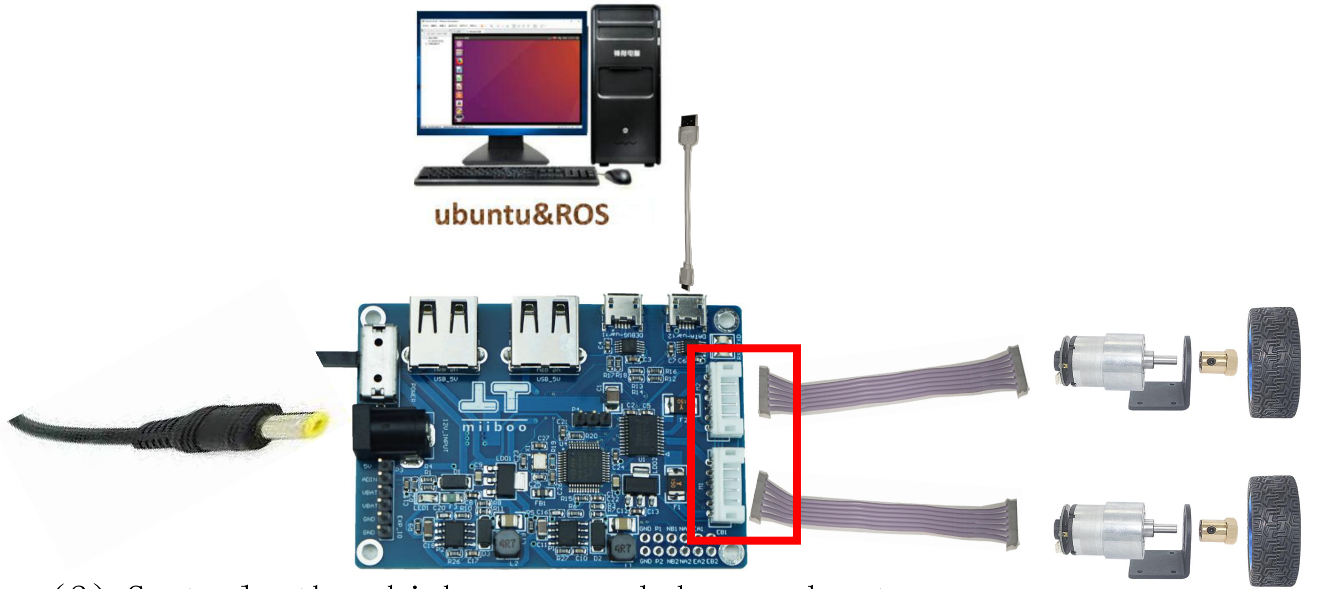
subscribe topic	/cmd_vel	(geometry_msgs::Twist)
publish topic	/wheel_left_speed /wheel_right_speed /odom /tf	(msgs::Float32) (msgs::Float32) (nav_msgs::Odometry) (odom->base_footprint)

The driver provides a subscription and publishing interface in ROS. The subscription interface */cmd_vel* is used to receive speed control data, and the publishing interface*/odom* is used to send odometer data.

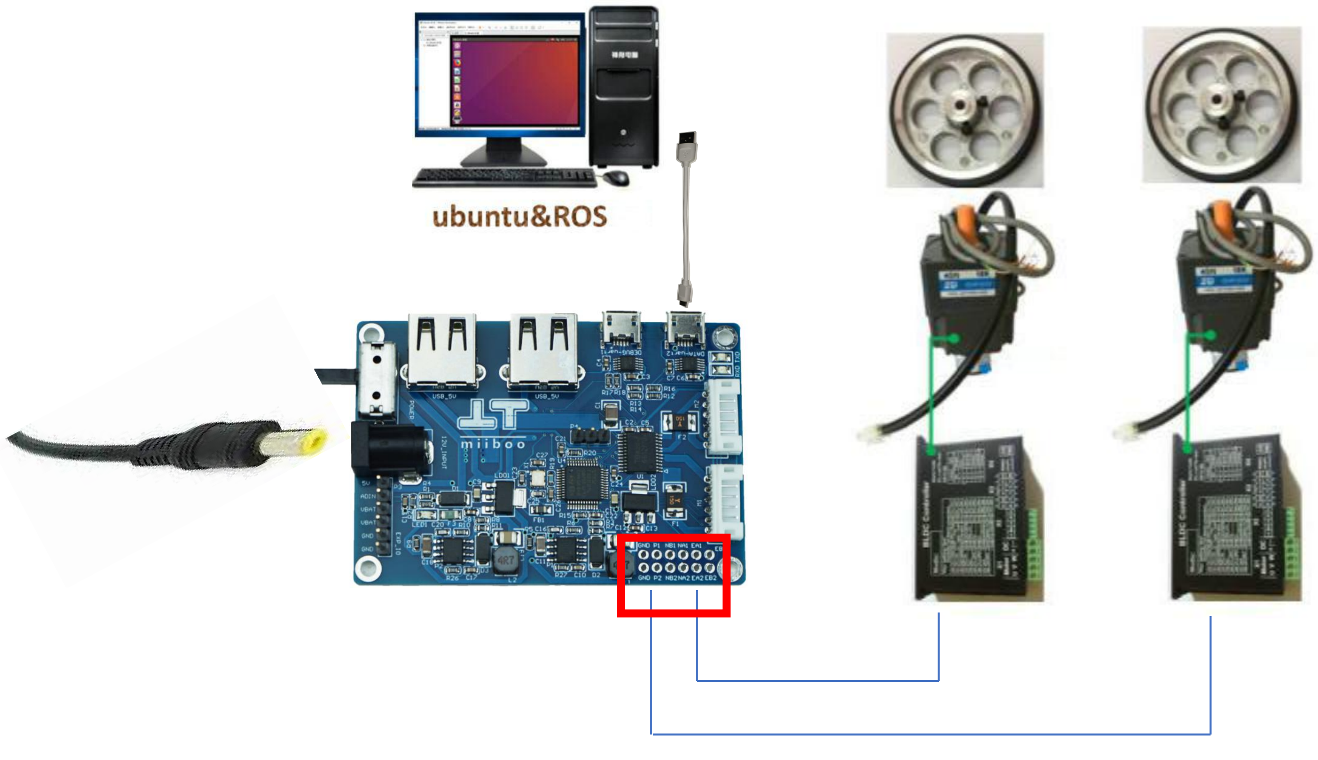
4.Tutorial

4.1. Cable connection

(1) Control the official standard GM37 coded geared motor



(2) Control other high-power coded geared motors



4.2. Tutorial for how to use ROS driver on host computer

(1) Download miiboo_bringup driver package

Download the miiboo_bringup.zip compressed package and unzip it in your ROS workspace such as catkin_ws/src/).

(2) Compile the miiboo_bringup driver package

Compile the miiboo_bringup driver package that you just decompressed in your own ROS workspace. Take the catkin_ws workspace as an example. The compilation command is as follows:

```
$ cd catkin_ws
$ catkin_make -DCATKIN_WHITELIST_PACKAGES="miiboo_bringup"
```

(3) Uart setting

Firstly, check the Data-Uart connection between the host computer and the miiboo motor driver board is connnected properly then check the serial device number that recognized by the Data-Uart serial port on the host computer . The command is as follows:

```
$ ll /dev/ | grep ttyUSB*
```

For example, the serial device number we inquired here is /dev/ttyUSB0 ,gran the serial device number read and write permissions, the command is as follows:

```
$ sudo chmod 777 /dev/ttyUSB0
```

Next, edit the file miiboo_bringup/launch/minimal.launch, set the value of the com_port parameter to /dev/ttyUSB0. Save and exit.

(4) Launch miiboo_bringup driver package

Before we launch the driver package, we need to activate our ROS workspace The command is as follows:

```
$ source catkin_ws/devel/setup.bash
```

Then use the following command to launch the driver package

```
$ roslaunch miiboo_bringup minimal.launch
```

Once the driver package is lanunched, you can send control information to the topic /cmd_vel to control the movement of the chassis motor, at the same time subscribe to the information from the topic /odom to obtain the odometer.

4.3. Odometer Calibration

(1) Grand executable permissions to the calibration script

```
$ cd miiboo_bringup/scripts/
$ sudo chmod +x ./*
```

(2) Start the ROS driver motor driver board

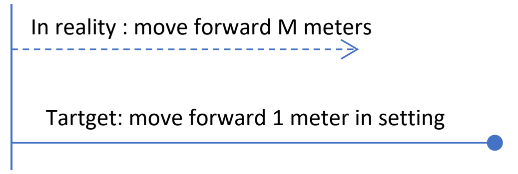
For the detailed operation of the startup method, see the steps in 4.2. It is assumed that the previous settings have been completed, so directly use the following command to start:

```
$ roslaunch miiboo_bringup minimal.launch
```

(3) Calibrate the straight line meter

Start the straight-line calibration procedure:

$ roslaunch miiboo_bringup check_linear.launch



calibrate the straight line and set the goal of moving forward 1 meter. Measure the actual straight-line distance M when the chassis stops. adjust the odometer straight-line parameter *speed_ratio* according to the following rules, which is in the miiboo_bringup/launch/minimal.launch file:

If M > 1meter, increase speed_ratio

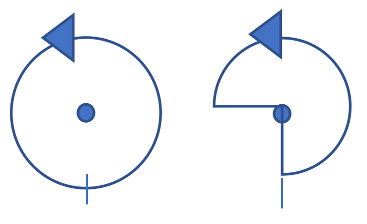
If M < 1meter, decrease speed_ratio

Every time you modify the *speed_ratio* parameter, you have to save it first then close the calibration program and the motor driver program. Repeat the steps (2)(3), until the calibration error of the straight line is within the acceptable range. Close the calibration program and skip to the next step.

(4) Calibrate the rotation angle

Start the angular calibration procedure:

$ roslaunch miiboo_bringup check_angular.launch



Target:360 degrees rotation   In reality: A degrees rotation

Set a 360 degrees rotation target and calibrate the rotation angle. Measure the actual rotation angle A when the chassis stops rotating, and adjust the odometer rotation angle parameter *wheel_distance* according to the following rules, which is in the miiboo_bringup/launch/minimal.launch file

if A > 360 degrees, decrease wheel_distance

if A < 360d egrees, increase wheel_distance

Every time you modify the *wheel_distance* parameter, you need to save it first then close the calibration program and the motor driver program. Repeat the steps (2)(4) until the calibration error of the rotation angle is within the acceptable range, then close the calibration program.

4.4. PID parameter tuning

If you are using the official standard GM37 coded geared motor, the PID parameters have already been tuned, and you don't have to tune it again. If you are using other types of motors, you may have to tune the PID. The host computer ROS driver of the miiboo motor drive board provides an interface for PID tuning. Please refer to the miiboo robot development document "Chapter 4 Differential Chassis" for the specific operation process . Design-the content of Section 5.。

5.Common Q&A

Q: The front, back, left, and right movement of my chassis motor is opposite to the actual situation?
A: Please try to change the connection sequence of the left motor and the right motor.

Q: The speed of my motor cannot be adjusted. No matter what speed it is, the motor will rotate at the maximum speed?
A: Please check whether the A and B signal wires of the encoder are connected correctly. Change the wiring sequence of the A and B signal wires of the encoder to see if it still exist.。

Q: Why does my motor vibrate severely during rotation?
A: Please try to tune the PID parameters