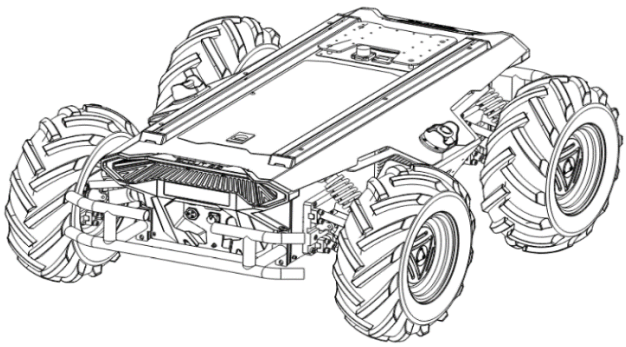
202003

SCOUT 2.0 User Manual

AgileX Robotics Team

Version 2.0 Release



To avoid the wrong operation, please read the user manual completely before operating SCOUT 2.0.

AgileX Robotics (Dongguan) Co., Ltd.

Building 10, Small and Medium Technology Enterprise Pioneer Park,
Songshan Lake, Dongguan, China
Tel：+86-0769-22892150
Email：sales@agilex.ai
www.agilex.ai

Revision History

Version	Content of Changes	Person in Charge	Date
V2.0.0	1. Added SCOUT2.0 product explanation pictures.	XieZhiqiang	2020/3/2

CONTENTS

0. Safety at a Glance

1. Environmental Considerations

- For remote control operation, select a relatively open area to use SCOUT 2.0, **because SCOUT 2.0 is not equipped with any automatic obstacle avoidance sensor.**
- Use SCOUT 2.0 always under **-20℃~45℃** ambient temperature.
- If SCOUT 2.0 is not configured with separate custom IP protection, its **water and dust protection will be IP22 ONLY.**

2. Pre-work Checklist

- Make sure each device has **sufficient power.**
- Make sure **SCOUT 2.0 does not have any obvious defects.**
- Check **if the remote controller battery has sufficient power.**
- When using, make sure **the emergency stop switch has been released.**

3. Operation

- In remote control operation, make sure the area around is **relatively spacious**.
- Carry out **remote control within the range of visibility**.
- As the maximum **load** of SCOUT 2.0 **is 50 kg**, when using, make sure **the effective load does not exceed 50 kg.**
- When SCOUT 2.0 is installed with external expansion, verify the barycenter location of the expansion and make sure is located in the **rotation center**.
- When SCOUT 2.0 **has low power alarm, charge it in time.**
- When SCOUT 2.0 **has a defect, please immediately stop using it to avoid secondary damage.**
- When SCOUT 2.0 has had a defect, **please contact the relevant technician to deal with it, do not handle the defect by yourself.**
- Always use SCOUT 2.0 **in the environment with the protection level required for the equipment**.
- **Do not push SCOUT 2.0 directly.**
- When charging, make sure the ambient **temperature is above 0℃.**
- If SCOUT 2.0 shakes in in-place rotation, adjust the suspension.

4. Maintenance

- Regularly check the tire pressure, and make sure it is within **1.8 bar~2.0 bar.**

- **If any tire is seriously worn out or has blown out,** please replace it in time.

1. Introduction

SCOUT 2.0 is designed as a multi-purpose UGV with different application scenarios considered: modular design; flexible connectivity; powerful motor system capable of high payload. Additional components such as stereo camera, laser radar, GPS, IMU and robotic manipulator can be optionally installed on SCOUT 2.0 for advanced navigation and computer vision applications. SCOUT 2.0 is frequently used for autonomous driving education and research, indoor and outdoor security patrolling, environment sensing, general logistics and transportation, to name a few only.

1.1 Component list

Name	Quantity
SCOUT 2.0 Robot body	X 1
Key lock	X 1
Battery charger (AC 220V)	X 1
Aviation plug (male, 4-pin)	X 2
USB to RS232 cable	X 1
Remote control transmitter (optional)	X 1

1.2 Tech specifications

Parameter Types	Items	Values
Mechanical specifications	L × W × H (mm)	930 X 699 X 348
	Wheelbase (mm)	498
	Front/rear wheel base (mm)	582 / 582
	Weight of vehicle body (kg)	62
	Battery type	Lithium battery 24V 30aH
	Motor	DC brushless 4 X 200W
	Reduction gearbox	1:30
	Drive type	Independent four-wheel drive
	Suspension	Independent suspension with single rocker arm
	Steering	Four-wheel differential steering
	Safety equipment	Servo brake/anti-collision tube
Motion	No-load highest speed (m/s)	≤1.5
	Minimum turning radius	Be able to turn on a pivot
	Maximum climbing capacity	≥30°
	Minimum ground clearance (mm)	135
Control	Control mode	Remote control Control command mode
	RC transmitter	2.4G/extreme distance 1km
	Communication interface	CAN / RS232

DJI RC transmitter is provided (optional) in the factory setting of SCOUT 2.0, which allows users to control the chassis of robot to move and turn; CAN and RS232 interfaces on SCOUT 2.0 can be used for user's customization.

2. The Basics

This section provides a brief introduction to the SCOUT 2.0 mobile robot platform, as shown in Figure 2.1 and Figure 2.2.

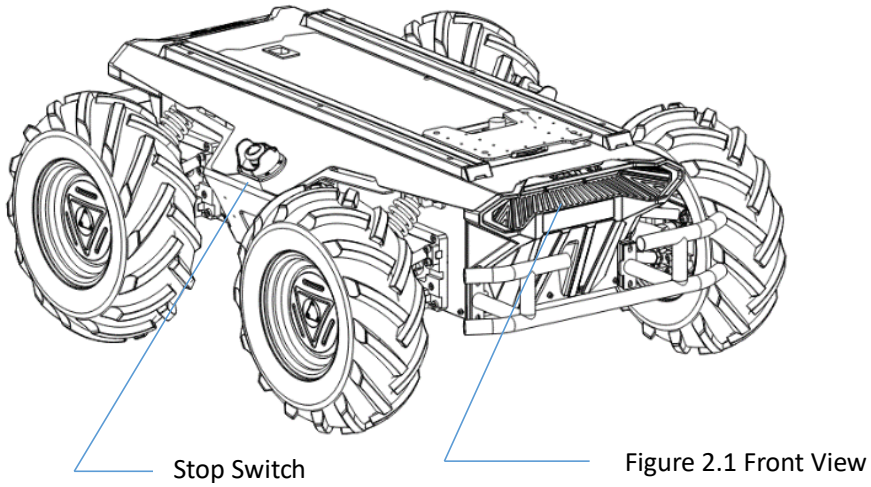


Figure 2.1 Front View

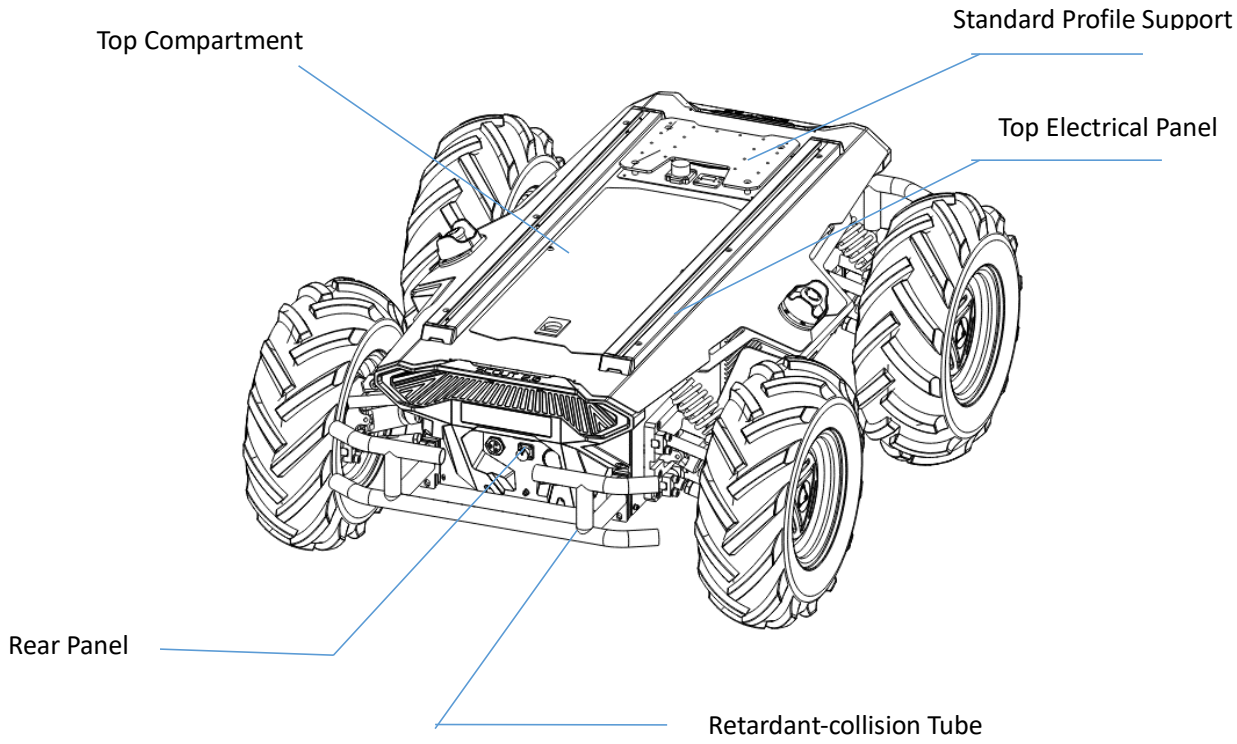


Figure 2.2 Rear View

Anti-collision beams are mounted around the vehicle to reduce possible damages to the vehicle body during a collision..

Lights are both mounted at front and at back of the vehicle, of which the white light is designed for illumination in front whereas the red light is designed at rear end for warning and indication.

Emergency stop buttons are installed on both sides of the robot to ensure easy access and pressing either one can shut down power of the robot immediately when the robot behaves abnormally.

Water-proof connectors for DC power and communication interfaces are provided both on top and at the rear of the robot, which not only allow flexible connection between the robot and external components but also ensures necessary protection to the internal of the robot even under severe operating conditions.

A bayonet open compartment is reserved on the top for users.

2.1 Status indication

Users can identify the status of vehicle body through the voltmeter, the beeper and lights mounted on SCOUT 2.0. For details, please refer to Table 2.1.

Status	Description
Voltage	The current battery voltage can be read from the voltmeter on the rear electrical interface and with an accuracy of 1V.
Replace battery	When the battery voltage is lower than 22.5V, the vehicle body will give a beep-beep-beep sound as a warning. When the battery voltage is detected as lower than 22V, SCOUT 2.0 will actively cut off the power supply to external extensions and drive to prevent the battery from being damaged. In this case, the chassis will not enable movement control and accept external command control.
Robot powered on	Front and rear lights are switched on.

Table 2.1 Descriptions of Vehicle Status

2.2 Instructions on electrical interfaces

2.2.1 Top electrical interface

SCOUT 2.0 provides two 4-pin aviation connectors and one DB9 (RS232) connector. (The current version can be used for upgrade of firmware but do not support for command.)

The position of the top aviation connector is shown in Figure 2.3.

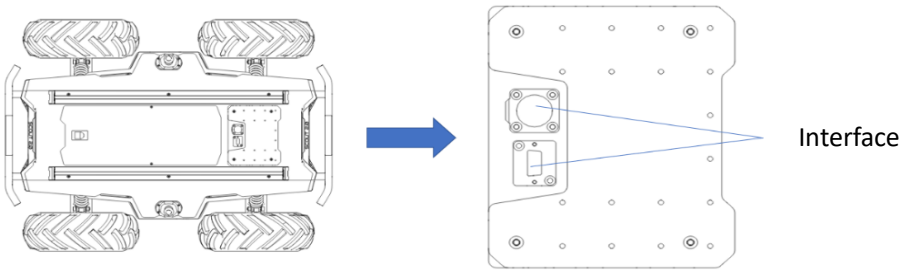
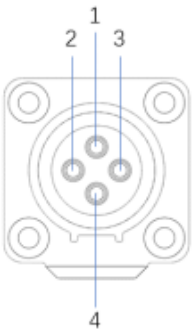


Figure 2.3 Schematic Diagram of SCOUT 2.0 Electrical Interface on Top

SCOUT 2.0 has an aviation extension interface both on top and at rear end, each of which is configured with a set of power supply and a set of CAN communication interface. These interfaces can be used to supply power to extended devices and establish communication. The specific definitions of pins are shown in Figure 2.4.

It should be noted that, the extended power supply here is internally controlled, which means the power supply will be actively cut off once the battery voltage drops below the pre-specified threshold voltage. Therefore, users need to notice that SCOUT 2.0 platform will send a low voltage alarm before the threshold voltage is reached and also pay attention to battery recharging during use.



Pin No.	Pin Type	Function and Definition	Remarks
1	Power	VCC	Power positive, voltage range 23 - 29.2V, MAX.current 10A
2		GND	Power negative
3	CAN	CAN_H	CAN bus high
4		CAN_L	CAN bus low

Figure 2.4 Definitions for Pins of Top Aviation Extension Interface

2.2.2 Rear electrical interface

The extension interface at rear end is shown in Figure 2.5, where Q1 is the key switch as the main electrical switch; Q2 is the recharging interface; Q3 is the power supply switch of drive system; Q4 is DB9 serial port; Q5 is the extension interface for CAN and 24V power supply; Q6 is the display of battery voltage.

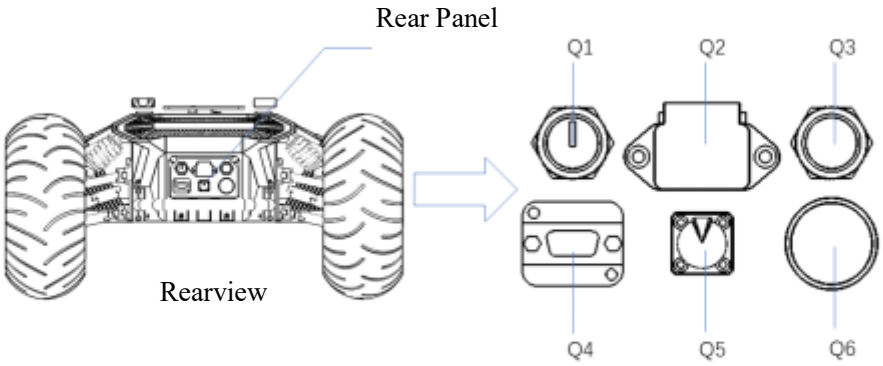
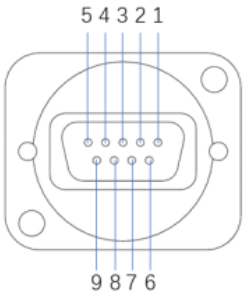


Figure 2.5 Rear View

Specific definitions for pins of Q4 are shown in Figure 2.6.



Pin No.	Definition
2	RS232-RX
3	RS232-TX
5	GND

Figure 2.6 Illustration Diagram of Q4 Pins

The rear panel provides the same CAN communication interface and 24V power interface with the top one (two of them are internally inter-connected). The pin definitions are given in Figure 2.7.

Pin No.	Pin Type	Function and Definition	Remarks
1	Power	VCC	Power positive, voltage range 23 - 29.2V, maximum current 5A
2		GND	Power negative
3	CAN	CAN_H	CAN bus high
4		CAN_L	CAN bus low

Figure 2.7 Description of Rear Aviation Interface Pins

2.3 Instructions on remote control

FS_i6_S remote control instructions

FS RC transmitter is an optional accessory of SCOUT 2.0 for manually controlling the robot. The transmitter comes with a left-hand-throttle configuration. In addition to the two sticks S1 and S2 used for sending linear and angular velocity commands, two switches are enabled by default: SWB for control mode selection (top position for command control mode and the middle position for remote control mode), SWC for lighting control. The two POWER buttons need to be pressed and held together to turn on or turn off the transmitter.

Figure 2.9 Schematic Diagram of Buttons on FS RC transmitter

2.4 Instructions on control demands and movements

A reference coordinate system can be defined and fixed on the vehicle body as shown in Figure 3.0 in accordance with ISO 8855.

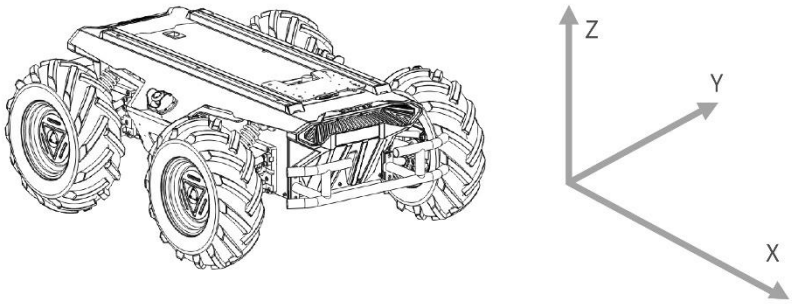


Figure 3.0 Schematic Diagram of Reference Coordinate System for Vehicle Body

As shown in Figure 3.0, the vehicle body of SCOUT 2.0 is in parallel with X axis of the established reference coordinate system. Following this convention, a positive linear velocity corresponds to the forward movement of the vehicle along positive x-axis direction and a positive angular velocity corresponds to positive right-hand rotation about the z-axis. In the manual control mode with a RC transmitter, pushing the C1 stick (DJI model) or the S1 stick (FS model) forward will generate a positive linear velocity command and pushing C2 (DJI model) and S2 (FS model) to the left will generate a positive angular velocity command.

2.5 Instructions on lighting control

Lights are mounted in front and at back of SCOUT 2.0, and the lighting control interface of SCOUT 2.0 is open to the users for convenience. Meanwhile, another lighting control interface is reserved on the RC transmitter for energy saving.

Currently the lighting control is only supported with the FS transmitter, and support for other transmitters is still under development. There are 3 kinds of lighting modes controlled with RC transmitter, **which can be switched among each other by SWC toggling:**

- NC mode: In NC mode, if the chassis is still, the front light will be turned off, and the rear light will enter BL mode to indicate its current operating status; if the chassis is in the traveling state at certain normal speed, the rear light will be turned off but the front light will be turned on;
- NO mode: In NO mode, if the chassis is still, the front light will be normally on, and the rear light will enter the BL mode to indicate the still status; if in movement mode, the rear light is turned off but the front light is turned on;
- BL mode: Front and rear lights are both in breathing mode under all circumstances.

Note on mode control:

Toggling SWC lever respectively refers to NC mode, NO mode and BL mode in bottom, middle and top positions.

3. Getting Started

This section introduces the basic operation and development of the SCOUT 2.0 platform using the CAN bus interface.

3.1 Use and operation

The basic operating procedure of startup is shown as follows:

- Check
  - Check the condition of vehicle body. Check whether there are significant anomalies; if so, please contact the after-sale service personnel for support;
  - Check the state of emergency stop switches. Make sure both emergency stop buttons are released;
  - Take off the cover of rear panel and you will see it;
  - For first-time use, check whether Q3 (drive power supply switch) on the rear panel has been pressed down; if so, please release it, and then the drive will be powered off;
- Startup
  - Rotate the key switch (Q1 on the electrical panel), and normally, the voltmeter will display correct battery voltage and front and rear lights will be both switched on;
  - Check the battery voltage. If there is no continuous "beep-beep-beep..." sound from beeper, it means the battery voltage is correct; if the battery power level is low, please charge the battery;
  - Press Q3 (drive power switch button);
- Shutdown
  - Rotate the key switch to cut off the power supply;
- Emergency stop
  - Press down emergency push button both on the left and the right of SCOUT 2.0 vehicle body;

Basic operating procedure of remote control:

After the chassis of SCOUT 2.0 mobile robot is started correctly, turn on the RC transmitter and select the remote-control mode. Then, SCOUT 2.0 platform movement can be controlled by the RC transmitter.

3.2 Charging

SCOUT 2.0 is equipped with a 10A charger by default to meet customers' recharging demand.

The detailed operating procedure of charging is shown as follows:

- Make sure the electricity of SCOUT 2.0 chassis is powered off. Before charging, please make sure Q1 (key switch) in the rear control console is turned off;

- Insert the charger plug into Q2 charging interface on the rear control panel;
- Connect the charger to power supply and turn on the switch in the charger. Then, the robot enters the charging state.

Note: For now, the battery needs about 3 to 5 hours to be fully recharged from 22V, and the voltage of a fully recharged battery is about 29.2V; the recharging duration is calculated as 30aH ÷ 10A = 3h.

3.3 Communication using CAN

SCOUT 2.0 provides CAN and RS232 interfaces for user customization. Users can select one of these interfaces to conduct command control over the vehicle body.

3.3.1 CAN message protocol

SCOUT 2.0 adopts CAN2.0B communication standard which has a communication baud rate of 500K and Motorola message format. Via external CAN bus interface, the moving linear speed and the rotational angular speed of chassis can be controlled; SCOUT 2.0 will feedback on the current movement status information and its chassis status information in real time.

The protocol includes system status feedback frame, movement control feedback frame and control frame, the contents of which are shown as follows:

The system status feedback command includes the feedback information about current status of vehicle body, control mode status, battery voltage and system failure. The description is given in Table 3.1.

Table 3.1 Feedback Frame of SCOUT 2.0 Chassis System Status

Command Name	**System Status Feedback Command**			
Sending node	Receiving node	**ID**	Cycle (ms)	Receive-timeout (ms)
Steer-by-wire chassis	Decision-making control unit	**0x151**	20ms	**None**
Data length	0x08			
Position	Function	Data type	Description	
byte [0]	Current status of vehicle body	unsigned int8	0x00 System in normal condition 0x01 Emergency stop mode (not enabled) 0x02 System exception	
byte [1]	Mode control	unsigned int8	0x00 Remote control mode 0x01 CAN command control mode[1] 0x02 Serial port control mode	
byte [2]	Battery voltage higher 8 bits	unsigned int16	Actual voltage X 10 (with an accuracy of 0.1V)	
byte [3]	Battery voltage lower 8 bits			
byte [4]	Failure information higher 8 bits	unsigned int16	See notes for details □**□	

byte [5]	Failure information lower 8 bits		
byte [6]	Count paritybit (count)	unsigned int8	0 - 255 counting loops, which will be added once every command sent
byte [7]	Parity bit (checksum)	unsigned int8	**Parity bit**

Table 3.2 Description of Failure Information

Description of Failure Information		
Byte	**Bit**	**Meaning**
byte [4]	bit [0]	Check error of CAN communication control command (0: No failure 1: Failure)
	bit [1]	Motor drive over-temperature alarm[1] (0: No alarm 1: Alarm) Temperature limited to 55℃
	bit [2]	Motor over-current alarm[1] (0: No alarm 1: Alarm) Current effective value 15A
	bit [3]	Battery under-voltage alarm (0: No alarm 1: Alarm) Alarm voltage 22.5V
	bit [4]	RC transmitter disconnection protection (0: Normal 1: RC transmitter disconnected)
	bit [5]	Reserved, default 0
	bit [6]	Reserved, default 0
	bit [7]	Reserved, default 0
byte [5]	bit [0]	Battery under-voltage failure (0: No failure 1: Failure) Protective voltage 22V
	bit [1]	Battery over-voltage failure (0: No failure 1: Failure)
	bit [2]	No.1 motor communication failure (0: No failure 1: Failure)
	bit [3]	No.2 motor communication failure (0: No failure 1: Failure)
	bit [4]	No.3 motor communication failure (0: No failure 1: Failure)
	bit [5]	No.4 motor communication failure (0: No failure 1: Failure)
	bit [6]	Motor drive over-temperature protection[2] (0: No protection 1: Protection) Temperature limited to 65℃
	bit [7]	Motor over-current protection[2] (0: No protection 1: Protection) Current effective value 20A

[1]: The subsequent versions of robot chassis firmware version after V1.2.8 are supported, but previous versions need to be updated before supported.

[2] The over-temperature alarm of motor drive and the motor over-current alarm will not be internally processed but just set in order to provide for the upper computer to complete certain pre-processing. If drive over-current occurs, it is suggested to reduce the vehicle speed; if over-temperature occurs, it is suggested to reduce the speed first and wait the temperature to decrease. This flag bit will be restored to normal condition as the temperature decreases, and the over-current alarm will be actively cleared once the current value is restored to normal condition;

[3]: The over-temperature protection of motor drive and the motor over-current protection will be internally processed. When the temperature of motor drive is higher than the protective temperature, the drive output will be limited, the vehicle will slowly stop, and the control value of movement control command will become invalid. This flag bit will not be actively cleared, which needs the upper computer to send the command of clearing failure protection. Once the command is cleared, the movement control command can only be executed normally.

The command of movement control feedback frame includes the feedback of current linear speed and angular speed of moving vehicle body. For the detailed content of protocol, please refer to Table 3.3.

Table 3.3 Movement Control Feedback Frame

Command Name	**Movement Control Feedback Command**			
Sending node	Receiving node	**ID**	Cycle (ms)	Receive-timeout (ms)
Steer-by-wire chassis	Decision-making control unit	**0x131**	20ms	**None**
Data length	0x08			
Position	Function	Data type	Description	
byte [0]	Moving speed higher 8 bits	signed int16	Actual speed X 1000 (with an accuracy of 0.001m/s)	
byte [1]	Moving speed lower 8 bits			
byte [2]	Rotational speed higher 8 bits	signed int16	Actual speed X 1000 (with an accuracy of 0.001rad/s)	
byte [3]	Rotational speed lower 8 bits			
byte [4]	Reserved	-	0x00	
byte [5]	Reserved	-	0x00	
byte [6]	Count paritybit (count)	unsigned int8	0 - 255 counting loops, which will be added once every command sent	
byte [7]	Parity bit (checksum)	unsigned int8	**Parity bit**	

The control frame includes mode control, failure clearing command, control openness of linear speed, control openness of angular speed and checksum. For its detailed content of protocol, please refer to Table 3.4.

Table 3.4 Control Frame of Movement Control Command

Command Name	Control Command			
Sending node	Receiving node	**ID**	Cycle (ms)	Receive-timeout (ms)
Decision-making control unit	Chassis node	**0x130**	20ms	**500ms**
Data length	0x08			

Position	Function	Data type	Description
byte [0]	Control mode	unsigned int8	0x00 Remote control mode 0x01 CAN command control mode[1] 0x02 Serial port control mode
byte [1]	Failure clearing command	unsigned int8	See Note 2 for details*
byte [2]	Linear speed percentage	signed int8	Maximum speed 1.5m/s, value range (-100, 100)
byte [3]	Angular speed percentage	signed int8	Maximum speed 0.5235rad/s, value range (-100, 100)
byte [4]	Reserved	-	0x00
byte [5]	Reserved	-	0x00
byte [6]	Count paritybit (count)	unsigned int8	0 - 255 counting loops, which will be added once every command sent
byte [7]	Parity bit (checksum)	unsigned int8	Parity bit

Note 1 - Control mode instructions

In case the RC transmitter is powered off, the control mode of SCOUT 2.0 is defaulted to command control mode, which means the chassis can be directly controlled via command. However, even though the chassis is in command control mode, the control mode in the command needs to be set to 0x01 for successfully executing the speed command. Once the RC transmitter is switched on again, it has the highest authority level to shield the command control and switch over the control mode.

Note 2 - Information about failure clearing command:

- **0x00 No failure clearing command**
- **0x01 Clear battery under-voltage failure**
- **0x02 Clear battery over-voltage failure**
- **0x03 Clear No.1 motor communication failure**
- **0x04 Clear No.2 motor communication failure**
- **0x05 Clear No.3 motor communication failure**
- **0x06 Clear No.4 motor communication failure**
- **0x07 Clear motor drive over-temperature failure**
- **0x08 Clear motor over-current failure**

Note 3 - Example data: The following data is only used for testing

1. The vehicle moves forward at 0.15m/s.

byte [0]	byte [1]	byte [2]	byte [3]	byte [4]	byte [5]	byte [6]	byte [7]

0x01	0x00	0x0a	0x00	0x00	0x00	0x00	0x44

2. The vehicle rotates at 0.07853rad/s.

byte [0]	byte [1]	byte [2]	byte [3]	byte [4]	byte [5]	byte [6]	byte [7]
0x01	0x00	0x00	0x0a	0x00	0x00	0x00	0x44

3. When the vehicle stays still, switch the control mode to command mode (test without RC transmitter switched on)

byte [0]	byte [1]	byte [2]	byte [3]	byte [4]	byte [5]	byte [6]	byte [7]
0x01	0x00	0x00	0x00	0x00	0x00	0x00	0x3a

The chassis status information will be fed back; what's more, the information about motor current, encoder and temperature are also included. The following feedback frame contains the information about motor current, encoder and motor temperature:

The serial numbers of 4 motors in the chassis are shown in the figure below:

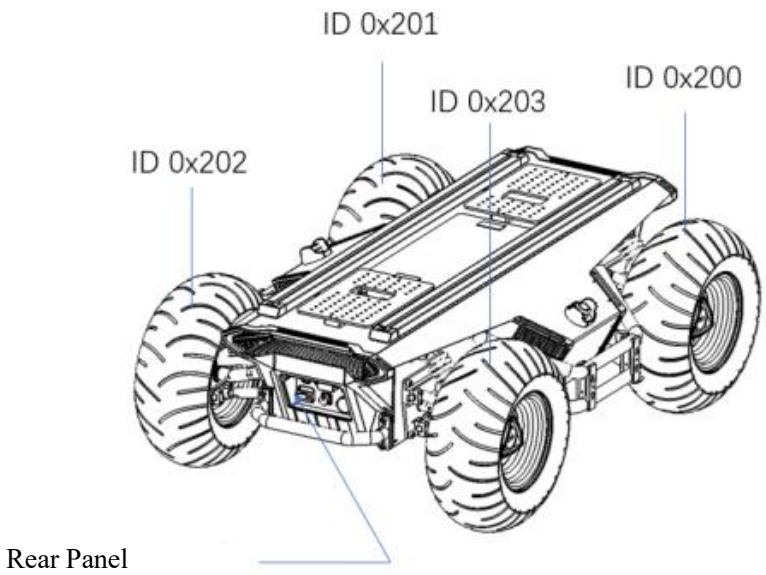


Figure 3.0 Schematic Diagram of Motor Feedback IDs

Table 3.5 No.1 Motor Information Feedback

Command Name	**No.1 Motor Drive Information Feedback Frame**			
Sending node	Receiving node	**ID**	Cycle (ms)	Receive-timeout (ms)
Steer-by-wire chassis	Decision-making control unit	**0x200**	20ms	**None**
Data length	0x08			
Position	Function	Data type	Description	

byte [0]	No.1 drive current higher 8 bits	unsigned int16	Actual current X 10 (with an accuracy of 0.1A)
byte [1]	No.1 drive current lower 8 bits		
byte [2]	No.1 drive rotational speed higher 8 bits	signed int16	Actual motor shaft velocity (RPM)
byte [3]	No.1 drive rotational speed lower 8 bits		
byte [4]	No.1 hard disk drive (HDD) temperature	signed int8	Actual temperature (with an accuracy of 1℃)
byte [5]	No.1 motor temperature	signed int8	Actual temperature (with an accuracy of 1℃)
byte [6]	Count parity (count)	unsigned int8	0 - 255 counting loops, which will be added once every command sent
byte [7]	Parity bit (checksum)	unsigned int8	**Parity bit**

Table 3.6 No.2 Motor Information Feedback

Command Name	**No. 2 Motor Drive Information Feedback Frame**			
Sending node	Receiving node	**ID**	Cycle (ms)	Receive-timeout (ms)
Steer-by-wire chassis	Decision-making control unit	**0x201**	20ms	**None**
Data length	0x08			
Position	Function	Data type	Description	
byte [0]	No.2 drive current higher 8 bits	unsigned int16	Actual current X 10 (with an accuracy of 0.1A)	
byte [1]	No.2 drive current lower 8 bits			
byte [2]	No.2 drive rotational speed higher 8 bits	signed int16	Actual motor shaft velocity (RPM)	
byte [3]	No.2 drive rotational speed lower 8 bits			
byte [4]	No.2 hard disk drive (HDD) temperature	signed int8	Actual temperature (with an accuracy of 1℃)	
byte [5]	No. 2 motor temperature	signed int8	Actual temperature (with an accuracy of 1℃)	
byte [6]	Count parity bit (count)	unsigned int8	0 - 255 counting loops, which will be added once every command sent	
byte [7]	Parity bit (checksum)	unsigned int8	**Parity bit**	

Table 3.7 No. 3 Motor Information Feedback

Command Name	**No. 3 Motor Drive Information Feedback Frame**			
Sending node	Receiving node	**ID**	Cycle (ms)	Receive-timeout (ms)
Steer-by-wire	Decision-making	**0x202**	20ms	**None**

chassis	control unit			
Data length	0x08			
Position	Function	Data type	Description	
byte [0]	No.3 drive current higher 8 bits	unsigned int16	Actual current X 10 (with an accuracy of 0.1A)	
byte [1]	No.3 drive current lower 8 bits			
byte [2]	No.3 drive rotational speed higher 8 bits	signed int16	Actual motor shaft velocity (RPM)	
byte [3]	No.3 drive rotational speed lower 8 bits			
byte [4]	No.3 hard disk drive (HDD) temperature	signed int8	Actual temperature (with an accuracy of 1℃)	
byte [5]	No. 3 motor temperature	signed int8	Actual temperature (with an accuracy of 1℃)	
byte [6]	Count parity bit (count)	unsigned int8	0 - 255 counting loops, which will be added once every command sent	
byte [7]	Parity bit (checksum)	unsigned int8	**Parity bit**	

Table 3.8 No. 4 Motor Information Feedback

Command Name	**No. 4 Motor Drive Information Feedback Frame**			
Sending node	Receiving node	**ID**	Cycle (ms)	Receive-timeout (ms)
Steer-by-wire chassis	Decision-making control unit	**0x203**	20ms	**None**
Data length	0x08			
Position	Function	Data type	Description	
byte [0]	No. 4 drive current higher 8 bits	unsigned int16	Actual current X 10 (with an accuracy of 0.1A)	
byte [1]	No. 4 drive current lower 8 bits			
byte [2]	No. 4 drive rotational speed higher 8 bits	signed int16	Actual motor shaft velocity (RPM)	
byte [3]	No. 4 drive rotational speed lower 8 bits			
byte [4]	No. 4 hard disk drive (HDD) temperature	signed int8	Actual temperature (with an accuracy of 1℃)	
byte [5]	No. 4 motor	signed int8	Actual temperature (with an accuracy of	

	temperature		1℃)
byte [6]	Count parity bit (count)	unsigned int8	0 - 255 counting loops, which will be added once every command sent
byte [7]	Parity bit (checksum)	unsigned int8	Parity bit

Table 3.9 Lighting Control Frame

Command Name	**Lighting Control Frame**			
Sending node	Receiving node	**ID**	Cycle (ms)	Receive-timeout (ms)
Decision-making control unit	Steer-by-wire chassis	**0x140**	25ms	**None**
Data length	0x08			
Position	Function	Data type	Description	
byte [0]	Lighting control enable flag	unsigned int8	0x00 Control command invalid 0x01 Lighting control enable	
byte [1]	Front light mode	unsigned int8	0x00 NC 0x01 NO 0x02 BL mode 0x03 User-defined brightness	
byte [2]	Custom brightness of front light	unsigned int8	[0, 100], where 0 refers to no brightness, 100 refers to maximum brightness[5]	
byte [3]	Rear light mode	unsigned int8	0x00 NC 0x01 NO 0x02 BL mode 0x03 User-defined brightness	
byte [4]	Customize brightness for rear light	unsigned int8	[0, 100], where 0 refers to no brightness, 100 refers to maximum brightness	
byte [5]	Reserved	--	0x00	
byte [6]	Count parity bit (count)	unsigned int8	0 - 255 counting loops, which will be added once every command sent	
byte [7]	Parity bit (checksum)	unsigned int8	Parity bit	

Note [5]: The values are valid for custom mode.

Table 3.10 Lighting Control Feedback Frame

Command Name	**Lighting Control Feedback Frame**			
Sending node	Receiving node	**ID**	Cycle (ms)	Receive-timeout (ms)
Steer-by-wire chassis	Decision-making control unit	**0x141**	20ms	**None**
Data length	0x08			
Position	Function	Data type	Description	
byte [0]	Current lighting control enable flag	unsigned int8	0x00 Control command invalid 0x01 Lighting control enable	

byte [1]	Current front light mode	unsigned int8	0x00 NC 0x01 NO 0x02 BL mode 0x03 User-defined brightness
byte [2]	Current custom brightness of front light	unsigned int8	[0, 100], where 0 refers to no brightness, 100 refers to maximum brightness
byte [3]	Current rear light mode	unsigned int8	0x00 NC 0x01 NO 0x02 BL mode 0x03 User-defined brightness
byte [4]	Current custom brightness of rear light	unsigned int8	[0, 100], where 0 refers to no brightness, 100 refers to maximum brightness
byte [5]	Reserved	--	0x00
byte [6]	Count parity bit (count)	unsigned int8	0 - 255 counting loops, which will be added once every command sent
byte [7]	Parity bit (checksum)	unsigned int8	**Parity bit**

The data Parity bit is the last valid byte in the data segment of each frame of CAN message. Its checksum is calculated as follows: checksum =(ID_H + ID_L + data_length+ can_msg.data[0] + can_msg.data[1] + can_msg.data[2] + can_msg.data[3] +can_msg.data[4]+ …+ can_msg.data[n]) & 0xFF

- **ID_H and ID_L are respectively higher 8 bits and lower 8 bits of a frame ID.For example, if ID is 0x540, the corresponding ID_H is 0x05 and ID_L is 0x40;**
- **Data_length refers to the valid data length of a data segment in one frame of CAN message, which includes the checksum byte;**
- **can_msg.data[n] is the specific content of each byte in the valid data segment; the count parity bit needs to participate in the calculation of checksum, but the checksum itself does not participate in the calculation.**

```
/**
 * @briefCAN message checksum example code
 * @param[in]id : can id
 * @param[in]*data : can message data struct pointer
 * @param[in]len : can message data length
 * @returnthe checksum result
 */
staticuint8Agilex_CANMsgChecksum(uint16 id,uint8*data,uint8len)
{
uint8 checksum =0x00;
checksum =(uint8)(id &0x00ff)+(uint8)(id >>8)+len;
for(uint8 i =0; i <(len-1); i++)
{
checksum += data[i];
```

```
}
return checksum;
}
```

Figure 3.1 CAN Message Check Algorithm

3.3.2 CAN cable connection

2 aviation male plugs are supplied along with SCOUT 2.0 as shown in Figure 3.2. Users need to lead wires out by welding on their own. For wire definitions, please refer to Table 2.2.

Figure 3.2 Schematic Diagram of Aviation Male Plug

Note: In the current SCOUT 2.0 version, only the top interface is open to external extension. The maximum achievable output current is typically around 5 A.

3.3.3 Implementation of CAN command control

Correctly start the chassis of SCOUT 2.0 mobile robot, and turn on DJI RC transmitter. Then, switch to the command control mode, i.e. toggling S1 mode of DJI RC transmitter to the top. At this point, SCOUT 2.0 chassis will accept the command from CAN interface, and the host can also parse the current state of chassis with the real-time data fed back from CAN bus. For the detailed content of protocol, please refer to CAN communication protocol.

3.4 Communication using RS232

3.4.1 Introduction to serial protocol

This is a serial communication standard which was formulated collectively by Electronic Industries Association (EIA) together with Bell System, modem manufacturers and computer terminal manufacturers in 1970. Its full name is called "the technical standard for serial binary data exchange interface between data terminal equipment (DTE) and data communication equipment (DCE). This standard requires to use a 25-pin DB-25 connector of which each pin is specified with corresponding signal content and various signal levels. Afterwards, RS232 is simplified as DB-9 connector in IBM PCs, which has become a de facto standard since then. Generally, RS-232 ports for industrial control only use 3 kinds of cables - RXD, TXD and GND.

3.4.2 Serial message protocol

- **Basic parameters of communication**

Item	Parameter
Baud rate	115200
Check	No check
Data bit length	8 bits
Stop bit	1 bit

- **Protocol specification**

Start bit		Frame length	Command type	Command ID	Data field			Frame ID	Checksum composition
SOF		frame_L	CMD_TYPE	CMD_ID	data [0]	…	data[n]	frame_id	check_sum
byte 1	byte 2	byte 3	byte 4	byte 5	byte 6	…	byte 6+n	byte 7+n	byte 8+n
5A	A5								

The protocol includes start bit, frame length, frame command type, command ID, data field, frame ID, and checksum composition. Where, the frame length refers to the length excluding start bit and checksum composition; the checksum refers to the sum from start bit to all data of frame ID; the frame ID is a loop count between 0 to 255, which will be added once every command sent.

- **Protocol content**

  - System status feedback command

Command Name	System status feedback command

```
* @briefserial message checksum example code
 * @param[in]*data : serial message data struct pointer
 * @param[in]len :serial message data length
 * @returnthe checksum result
 */
staticuint8Agilex_SerialMsgChecksum(uint8*data,uint8len)
{
    uint8checksum =0x00;
    for(uint8 i =0; i <(len-1); i++)
    {
        checksum += data[i];
    }
    return checksum;
}
```

Figure 3.3 Example of Serial Check Algorithm Codes

Sending node	Receiving node	Cycle (ms)	Receive-timeout (ms)
Steer-by-wire chassis	Decision-making control unit	20ms	None
Frame length	0x0a		
Command type	Feedback command (0xAA)		
Command ID	0x01		
Data field length	6		
Position	Function	Data type	Description
byte [0]	Current status of vehicle body	unsigned int8	0x00 System in normal condition 0x01 Emergency stop mode (not enabled) 0x01 System exception
byte [1]	Mode control	unsigned int8	0x00 Remote control mode 0x01 CAN command control mode[1] 0x02 Serial port control mode
byte [2]	Battery voltage higher 8 bits	unsigned int16	Actual voltage X 10 (with an accuracy of 0.1V)
byte [3]	Battery voltage lower 8 bits		
byte [4]	Failure information higher 8 bits	unsigned int16	See notes for details □**□
byte [5]	Failure information lower 8 bits		

Description of Failure Information		
Byte	**Bit**	**Meaning**
byte [4]	bit [0]	Check error of CAN communication control command (0: No failure 1: Failure)
	bit [1]	Motor drive over-temperature alarm[1] (0: No alarm 1: Alarm) Temperature limited to 55°C
	bit [2]	Motor over-current alarm[1] (0: No alarm 1: Alarm) Current effective value 15A
	bit [3]	Battery under-voltage alarm (0: No alarm 1: Alarm) Alarm voltage 22.5V
	bit [4]	Reserved, default 0
	bit [5]	Reserved, default 0
	bit [6]	Reserved, default 0
	bit [7]	Reserved, default 0
byte [5]	bit [0]	Battery under-voltage failure (0: No failure 1: Failure) Protective voltage 22V
	bit [1]	Battery over-voltage failure (0: No failure 1: Failure)

	bit [2]	No.1 motor communication failure (0: No failure 1: Failure)
	bit [3]	No.2 motor communication failure (0: No failure 1: Failure)
	bit [4]	No.3 motor communication failure (0: No failure 1: Failure)
	bit [5]	No.4 motor communication failure (0: No failure 1: Failure)
	bit [6]	Motor drive over-temperature protection[2] (0: No protection 1: Protection) Temperature limited to 65℃
	bit [7]	Motor over-current protection[2] (0: No protection 1: Protection) Current effective value 20A

[1]: The subsequent versions of robot chassis firmware version after V1.2.8 are supported, but previous versions need to be updated before supported.

[2]: The over-temperature alarm of motor drive and the motor over-current alarm will not be internally processed but just set in order to provide for the upper computer to complete certain pre-processing. If drive over-current occurs, it is suggested to reduce the vehicle speed; if over-temperature occurs, it is suggested to reduce the speed first and wait the temperature to decrease. This flag bit will be restored to normal condition as the temperature decreases, and the over-current alarm will be actively cleared once the current value is restored to normal condition;

[3]: The over-temperature protection of motor drive and the motor over-current protection will be internally processed. When the temperature of motor drive is higher than the protective temperature, the drive output will be limited, the vehicle will slowly stop, and the control value of movement control command will become invalid. This flag bit will not be actively cleared, which needs the upper computer to send the command of clearing failure protection. Once the command is cleared, the movement control command can only be executed normally.

■ Movement control feedback command

Command Name	**Movement Control Feedback Command**		
Sending node	Receiving node	Cycle (ms)	Receive-timeout (ms)
Steer-by-wire chassis	Decision-making control unit	20ms	**None**
Frame length	0x0A		
Command type	Feedback command (0xAA)		
Command ID	0x02		
Data field length	6		
Position	Function	Data type	Description
byte [0]	Moving speed higher 8 bits	signed int16	Actual speed X 1000 (with an accuracy of 0.001m/s)
byte [1]	Moving speed lower 8 bits		
byte [2]	Rotational speed higher 8 bits	signed int16	Actual speed X 1000 (with an accuracy of 0.001rad/s)
byte [3]	Rotational speed lower 8 bits		
byte [4]	Reserved	-	0x00

byte [5]	Reserved	-	0x00

■ Movement control command

Command Name	**Control Command**		
Sending node	Receiving node	Cycle (ms)	Receive-timeout (ms)
Decision-making control unit	Chassis node	20ms	**500ms**
Frame length	0x0A		
Command type	Control command (0x55)		
Command ID	0x01		
Data field length	6		
Position	Function	Data type	Description
byte [0]	Control mode	unsigned int8	0x00 Remote control mode 0x01 CAN command control mode[1] 0x02 Serial port control mode See Note 2 for details*
byte [1]	Failure clearing command	unsigned int8	
byte [2]	Linear speed percentage	signed int8	Maximum speed 1.5m/s, value range (-100, 100)
byte [3]	Angular speed percentage	signed int8	Maximum speed 0.7853rad/s, value range (-100, 100)
byte [4]	Reserved	-	0x00
byte [5]	Reserved	-	0x00

■ No.1 motor drive information feedback frame

Command Name	**No.1 Motor Drive Information Feedback Frame**		
Sending node	Receiving node	Cycle (ms)	Receive-timeout (ms)
Steer-by-wire chassis	Decision-making control unit	20ms	**None**
Frame length	0x0A		
Command type	Feedback command (0xAA)		
Command ID	0x03		
Data field length	6		
Position	Function	Data type	Description
byte [0]	No.1 drive current higher 8 bits	unsigned int16	Actual current X 10 (with an accuracy of 0.1A)
byte [1]	No.1 drive current lower 8 bits		
byte [2]	No.1 drive rotational speed higher 8 bits	signed int16	Actual motor shaft velocity (RPM)
byte [3]	No.1 drive rotational speed lower 8 bits		
byte [4]	No.1 hard disk drive (HDD) temperature	signed int8	Actual temperature (with an accuracy of 1℃)

byte [5]	Reserved	--	0x00

■ No. 2 motor drive information feedback frame

Command Name	**No. 2 Motor Drive Information Feedback Frame**		
Sending node	Receiving node	Cycle (ms)	Receive-timeout (ms)
Steer-by-wire chassis	Decision-making control unit	20ms	**None**
Frame length	0x0A		
Command type	Feedback command (0xAA)		
Command ID	0x04		
Data field length	6		
Position	Function	Data type	Description
byte [0]	No.2 drive current higher 8 bits	unsigned int16	Actual current X 10 (with an accuracy of 0.1A)
byte [1]	No.2 drive current lower 8 bits		
byte [2]	No.2 drive rotational speed higher 8 bits	signed int16	Actual motor shaft velocity (RPM)
byte [3]	No.2 drive rotational speed lower 8 bits		
byte [4]	No.2 hard disk drive (HDD) temperature	signed int8	Actual temperature (with an accuracy of 1℃)
byte [5]	Reserved	--	0x00

■ No. 3 motor drive information feedback frame

Command Name	**No. 3 Motor Drive Information Feedback Frame**		
Sending node	Receiving node	Cycle (ms)	Receive-timeout (ms)
Steer-by-wire chassis	Decision-making control unit	20ms	**None**
Frame length	0x0A		
Command type	Feedback command (0xAA)		
Command ID	0x05		
Data field length	6		
Position	Function	Data type	Description
byte [0]	No.3 drive current higher 8 bits	unsigned int16	Actual current X 10 (with an accuracy of 0.1A)
byte [1]	No.3 drive current lower 8 bits		
byte [2]	No.3 drive rotational speed higher 8 bits	signed int16	Actual motor shaft velocity (RPM)
byte [3]	No.3 drive rotational speed lower 8 bits		
byte [4]	No.3 hard disk drive	signed int8	Actual temperature (with an

	(HDD) temperature		accuracy of 1℃)
byte [5]	Reserved	--	0x00

■ No. 4 motor drive information feedback frame

Command Name	**No. 4 Motor Drive Information Feedback Frame**		
Sending node	Receiving node	Cycle (ms)	Receive-timeout (ms)
Steer-by-wire chassis	Decision-making control unit	20ms	**None**
Frame length	0x0A		
Command type	Feedback command (0xAA)		
Command ID	0x06		
Data field length	6		
Position	Function	Data type	Description
byte [0]	No. 4 drive current higher 8 bits	unsigned int16	Actual current X 10 (with an accuracy of 0.1A)
byte [1]	No. 4 drive current lower 8 bits		
byte [2]	No. 4 drive rotational speed higher 8 bits	signed int16	Actual motor shaft velocity (RPM)
byte [3]	No. 4 drive rotational speed lower 8 bits		
byte [4]	No. 4 hard disk drive (HDD) temperature	signed int8	Actual temperature (with an accuracy of 1℃)
byte [5]	Reserved	--	0x00

■ Lighting control frame

Command Name	**Lighting Control Frame**		
Sending node	Receiving node	Cycle (ms)	Receive-timeout (ms)
Decision-making control unit	Chassis node	20ms	**500ms**
Frame length	0x0A		
Command type	Control command (0x55)		
Command ID	0x02		
Data field length	6		
Position	Function	Data type	Description
byte [0]	Lighting control enable flag	unsigned int8	0x00 Control command invalid 0x01 Lighting control enable
byte [1]	Front light mode	unsigned int8	0x00 NC 0x01 NO 0x02 BL mode 0x03 User-defined brightness

byte [2]	Custom brightness of front light	unsigned int8	[0, 100], where 0 refers to no brightness, 100 refers to maximum brightness[5]
byte [3]	Rear light mode	unsigned int8	0x00 NC 0x01 NO 0x02 BL mode 0x03 User-defined brightness
byte [4]	Custom brightness of rear light	unsigned int8	[0, 100], where 0 refers to no brightness, 100 refers to maximum brightness
byte [5]	Reserved	--	0x00

■ Lighting control feedback frame

Command Name	**Lighting Control Feedback Frame**		
Sending node	Receiving node	Cycle (ms)	Receive-timeout (ms)
Steer-by-wire chassis	Decision-making control unit	20ms	**None**
Frame length	0x0A		
Command type	Feedback command (0xAA)		
Command ID	0x07		
Data field length	6		
Position	Function	Data type	Description
byte [0]	Current lighting control enable flag	unsigned int8	0x00 Control command invalid 0x01 Lighting control enable
byte [1]	Current front light mode	unsigned int8	0x00 NC 0x01 NO 0x02 BL mode 0x03 User-defined brightness
byte [2]	Current custom brightness of front light	unsigned int8	[0, 100], where 0 refers to no brightness, 100 refers to maximum brightness
byte [3]	Current rear light mode	unsigned int8	0x00 NC 0x01 NO 0x02 BL mode 0x03 User-defined brightness
byte [4]	Current custom brightness of rear light	unsigned int8	[0, 100], where 0 refers to no brightness, 100 refers to maximum brightness
byte [5]	Reserved	--	0x00

- Example data

The chassis is controlled to move forward at a linear speed of 0.15m/s, from which specific data is shown as follows:

Start bit		Frame length	Command type	Command ID	Data field			Frame ID	Checksum composition
byte 1	byte 2	byte 3	byte 4	byte 5	byte 6	…	byte 6+n	byte 7+n	byte 8+n
0x5A	0xA5	0x0A	0x55	0x01	…	…	…	0x00	0x6B

The data field content is shown as follows:

Position	Function	Value
byte [0]	Control mode	0x02
byte [1]	Failure clearing command	0x00
byte [2]	Linear speed percentage	0x0A
byte [3]	Angular speed percentage	0x00
byte [4]	Reserved	0x00
byte [5]	Reserved	0x00

The entire data string is:

5A A5 0A 55 01 02 00 0A 00 00 00 00 6B

- **Additional notes**

This protocol requires a firmware version above V1.3.3.

3.4.3 Serial connection

Take out the USB-to-RS232 serial cable from our communication tool kit to connect it onto the serial port at the rear end. Then, use the serial port tool to set corresponding baud rate, and conduct the test with the example date provided above. If the RC transmitter is on, it needs to be switched to command control mode; if the RC transmitter is off, directly send the control command. It should be noted that, the command must be sent periodically, because if the chassis has not received the serial port command after 500ms, it will enter the disconnected protection status.

3.5 Firmware upgrades

The RS232 port on SCOUT 2.0 can be used by users to upgrade the firmware for the main controller in order to get bugfixes and feature enhancements. A PC client application with graphical user interface is provided to help make the upgrading process fast and smooth. A screenshot of this application is shown in Figure 3.3.

Upgrade preparation:

- Serial cable X 1
- USB-to-serial port X 1
- SCOUT 2.0 chassis X 1

- Computer (Windows operating system) X 1

Upgrade procedure:

- Before connection, ensure the robot chassis is powered off;
- Connect the serial cable onto the serial port at rear end of SCOUT 2.0 chassis;
- Connect the serial cable to the computer;
- Open the client software;
- Select the port number;
- Power on SCOUT 2.0 chassis, and immediately click to start connection (SCOUT 2.0 chassis will wait for 6s before power-on; if the waiting time is more than 6s, it will enter the application); if the connection succeeds, "connected successfully" will be prompted in the text box;
- Load Bin file;
- Click the Upgrade button, and wait for the prompt of upgrade completion;
- Disconnect the serial cable, power off the chassis, and then turn the power off and on again.

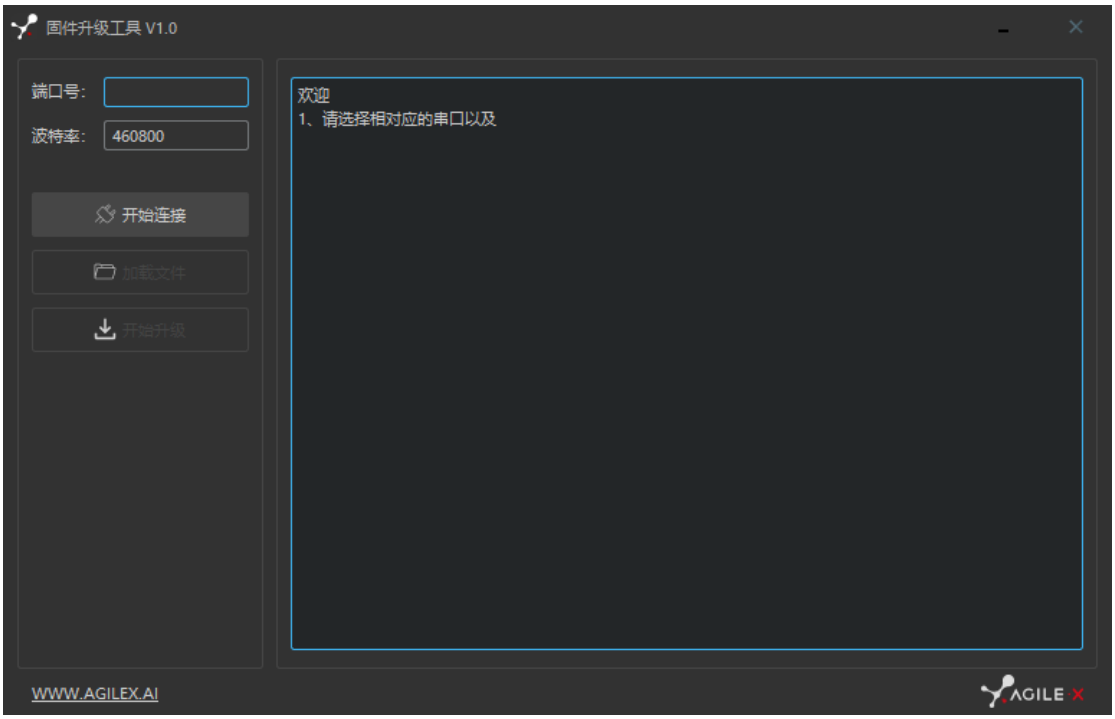


Figure 3.3 Client Interface of Firmware Upgrade

3.6 Use example of SCOUT 2.0 SDK

In order to help users implement robot-related development more conveniently, a cross-platform supported SDK is developed for SCOUT 2.0 mobile robot.

SDK software package provides a C++ based interface, which is used to communicate with the

chassis of SCOUT 2.0 mobile robot and can obtain the latest status of the robot and control basic actions of the robot. For now, CAN adaptation to communication is available, but RS232-based adaptation is still under way.

Based on this, related tests have been completed in Nvidia Jetson TX2.

3.7 Use example of SCOUT 2.0 ROS Package

ROS provides some standard operating system services, such as hardware abstraction, control of underlying devices, implementation of common functions, and management of inter-process messages and data packets. ROS is based on a kind of graphic structure, so that processes of different nodes can accept, post and aggregate various kinds of information (such as sensing, control, status, planning, etc.). At present, ROS is best supported in Ubuntu.

Preparation for development

- **Hardware preparation**
  - CANable can communication module X 1
  - Thinkpad E470 laptop computer X 1
  - AGILEX SCOUT 2.0 mobile robot chassis X 1
  - AGILEX SCOUT 2.0 supported RC transmitter FS-i6s X 1
  - AGILEX SCOUT 2.0 top aviation socket X 1
- **Instructions on use example environments**
  - Ubuntu 16.04 LTS (beta version, which has been tested in Ubuntu 18.04 LTS)
  - ROS Kinetic(subsequent versions also have been tested)
  - Git

ROS installation and environment setup

For installation details, please refer to http: //wiki.ros.org/kinetic/Installation/Ubuntu.

CANable hardware testing and CAN communication

- Set CAN-TO-USB adapter.
  - Enabe gs_usb kernel module.

```
$ sudo modprobe gs_usb
```

  - Set the baud rate to 500k and enable can-to-usb adapter.

```
$ sudo ip link set can0 up type can bitrate 500000
```

  - If no error occurs in previous steps, you should be able to use the command to immediately view can equipment.

```
$ ifconfig -a
```

  - Install and use can-utils to test hardware.

```
$ sudo apt install can-utils
```

- If can-to-usb has already been connected to SCOUT 2.0 this time and the vehicle has been turned on, following command can be used to monitor the data from SCOUT 2.0 chassis.

```
$ candump can0
```

Reference sources:

[1] https://github.com/westonrobot/SCOUT 2.0_sdk

[2] https ://wiki.rdu.im/_pages/Notes/Embedded-System/Linux/can-bus-in-linux.html

Download and compilation of AGILEX SCOUT 2.0 ROS Package

- Download ROS dependent packages.

```
$ sudo apt install ros-melodic-controller-manager
$ sudo apt install ros-melodic-teleop-twist-keyboard
```

- Clone and compile SCOUT 2.0_ros source codes.

```
$ cd ~/catkin_ws/src
$ git clone https://github.com/westonrobot/SCOUT 2.0_ros.git
$ cd ..
$ catkin_make
```

- Reference source: https: //github.com/westonrobot/SCOUT 2.0_ros

Hardware connection and preparation

- Lead out CAN cable from SCOUT 2.0 top or rear aviation socket, and connect the CAN_H and CAN_L wires to the corresponding CAN_TO_USB adapter terminals;
- Turn on the rotary switch of SCOUT 2.0 mobile robot chassis, and press down the control switch of drive power supply. Normally, you can hear the beeper to give a "beep" sound, which means the drive has been properly powered on; if there is no sound, check whether the E-stop switches on both sides have been released;
- Connect CAN_TO_USB to the usb port of laptop, and you can test again whether CAN is in normal condition.

Launching of ROS nodes

- Launch basic nodes.

```
$ roslaunchSCOUT 2.0_bringupSCOUT 2.0_minimal.launch
```

- Launch remote operation nodes for keyboard.

```
$ roslaunchSCOUT 2.0_bringupSCOUT 2.0_teleop_keyboard.launch
```

4. Precautions

This section includes some precautions that should be paid attention to for SCOUT 2.0 use and development.

4.1 Battery

- The battery supplied with SCOUT 2.0 is not fully charged in the factory setting, but its specific power capacity can be displayed on the voltmeter at rear end of SCOUT 2.0 chassis or read via CAN bus communication interface. The battery recharging can be stopped when the green LED on the charger turns green. Note that if you keep the charger connected after the green LED gets on, the charger will continue to charge the battery with about 0.1A current for about 30 minutes more to get the battery fully charged.
- Please do not charge the battery after its power has been depleted, and please charge the battery in time when low battery level alarm is on;
- Static storage conditions: The best temperature for battery storage is -20℃ to 60℃; in case of storage for no use, the battery must be recharged and discharged once about every 2 months, and then stored in full voltage state. Please do not put the battery in fire or heat up the battery, and please do not store the battery in high-temperature environment;
- Charging: The battery must be charged with a dedicated lithium battery charger; lithium-ion batteries cannot be charged below 0°C (32°F) and modifying or replacing the original batteries are strictly prohibited.

4.2 Operational environment

- The operating temperature of SCOUT 2.0 outdoors is -10℃ to 45℃; please do not use it below -10℃ and above 45℃ outdoors;
- The operating temperature of SCOUT 2.0 indoors is 0℃ to 42℃; please do not use it below 0℃ and above 42℃ indoors;
- The requirements for relative humidity in the use environment of SCOUT 2.0 are: maximum 80%, minimum 30%;
- Please do not use it in the environment with corrosive and flammable gases or closed to combustible substances;
- Do not place it near heaters or heating elements such as large coiled resistors, etc.;
- Except for specially customized version (IP protection class customized), SCOUT 2.0 is not water-proof, thus please do not use it in rainy, snowy or water-accumulated environment;
- The elevation of recommended use environment should not exceed 1,000m;
- The temperature difference between day and night of recommended use environment should not exceed 25℃;

- **Regularly check the tire pressure, and make sure it is within 1.8 bar to 2.0bar**
- **If any tire is seriously worn out or has blown out, please replace it in time.**

4.3 Electrical/extension cords

- For the extended power supply on top, the current should not exceed 6.25A and the total power should not exceed 150W;
- For the extended power supply at rear end, the current should not exceed 5A and the total power should not exceed 120W;
- When the system detects that the battery voltage is lower than the safe voltage class, external power supply extensions will be actively switched to. Therefore, users are suggested to notice if external extensions involve the storage of important data and have no power-off protection.

4.4 Mechanical load

4.5 Other notes

- SCOUT 2.0 has plastic parts in front and rear, please do not directly hit those parts with excessive force to avoid possible damages;
- When handling and setting up, please do not fall off or place the vehicle upside down;
- For non-professionals, please do not disassemble the vehicle without permission.

4.6 Additional safety advices

- In case of any doubts during use, please follow related instruction manual or consult related technical personnel;
- Before use, pay attention to field condition, and avoid mis-operation that will cause personnel safety problem;
- In case of emergencies, press down the emergency stop button and power off the equipment;
- Without technical support and permission, please do not personally modify the internal equipment structure.

5. Q&A

Q: SCOUT 2.0 is started up correctly, but why cannot the RC transmitter control the vehicle body to move?

A: First, check whether the drive power supply is in normal condition, whether the drive power switch is pressed down and whether E-stop switches are released; then, check whether the control mode selected with the top left mode selection switch on the RC transmitter is correct.

Q**: SCOUT 2.0 remote control is in normal condition, and the information about chassis status and movement can be received correctly, but when the control frame protocol is issued, why cannot the vehicle body control mode be switched and the chassis respond to the control frame protocol?**

A: Normally, if SCOUT 2.0 can be controlled by a RC transmitter, it means the chassis movement is under proper control; if the chassis feedback frame can be accepted, it means CAN extension link is in normal condition. Please check the CAN control frame sent to see whether the data check is correct and whether the control mode is in command control mode. You can check the status of error flag from the error bit in the chassis status feedback frame.

Q: **SCOUT 2.0 gives a "beep-beep-beep..." sound in operation, how to deal with this problem?**

A: If SCOUT 2.0 gives this "**beep-beep-beep**" sound continuously, it means the battery is in the alarm voltage state. Please charge the battery in time. Once other related sound occur, there may be internal errors. You can check related error codes via CAN bus or communicate with related technical personnel.

Q:**Is the tire wear of SCOUT 2.0 is normally seen in operation?**

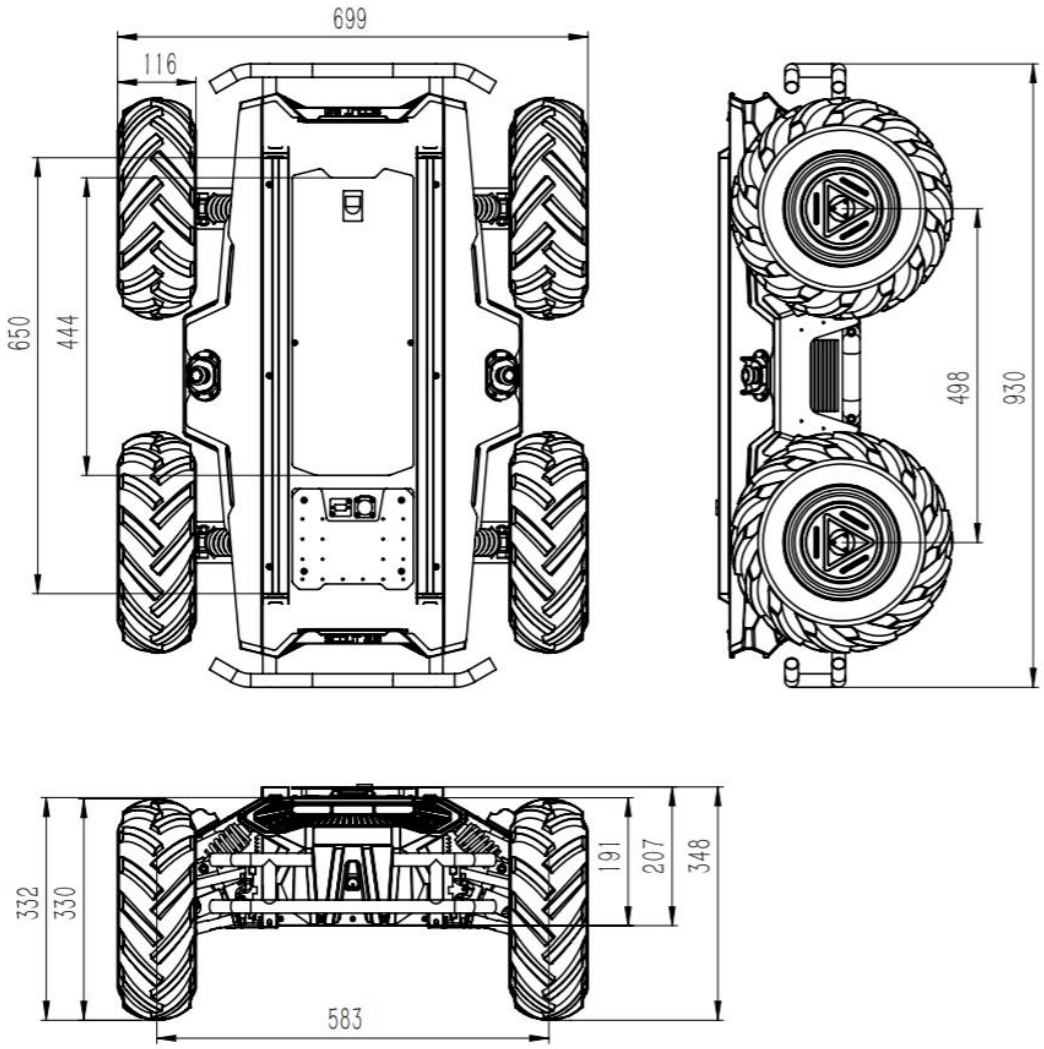
A: The tire wear of SCOUT 2.0 is normally seen when it is running. As SCOUT 2.0 is based on the four-wheel differential steering design, sliding friction and rolling friction both occur when the vehicle body rotates. If the floor is not smooth but rough, tire surfaces will be worn out. In order to reduce or slow down the wear, small-angle turning can be conducted for less turning on a pivot.

Q: **When communication is implemented via CAN bus, the chassis feedback command is issued correctly, but why does not the vehicle respond to the control command?**

A: There is a communication protection mechanism inside SCOUT 2.0, which means the chassis is provided with timeout protection when processing external CAN control commands. Suppose the vehicle receives one frame of communication protocol, but it does no receive the next frame of control command after 500ms. In this case, it will enter communication protection mode and set the speed to 0. Therefore, commands from upper computer must be issued periodically.

6. Product Dimensions

6.1 Illustration diagram of product external dimensions



6.2 Illustration diagram of top extended support dimensions