SCOUT 2.0

AgileX Robotics Team

User Manual V.2.1.0 2021.03

This chapter contains important safety information, before the robot is powered on for the first time, any individual or organization must read and understand this information before using the device. If you have any questions about use, please contact us at support@agilex.ai. Please follow and implement all assembly instructions and guidelines in the chapters of this manual, which is very important. Particular attention should be paid to the text related to the warning signs.

⚠ Safety Information

The information in this manual does not include the design, installation and operation of a complete robot application, nor does it include all peripheral equipment that may affect the safety of the complete system. The design and use of the complete system need to comply with the safety requirements established in the standards and regulations of the country where the robot is installed.

SCOUT integrators and end customers have the responsibility to ensure compliance with the applicable laws and regulations of relevant countries, and to ensure that there are no major dangers in the complete robot application. This includes but is not limited to the following:

1.Effectiveness and responsibility

- Make a risk assessment of the complete robot system.
- Connect the additional safety equipment of other machinery defined by the risk assessment together.
- Confirm that the design and installation of the entire robot system's peripheral equipment, including software and hardware systems, are correct.
- This robot does not have a complete autonomous mobile robot, including but not limited to automatic anti-collision, anti-falling, biological approach warning and other related safety functions. Related functions require integrators and end customers to follow relevant regulations and feasible laws and regulations for safety assessment , To ensure that the developed robot does not have any major hazards and safety hazards in actual applications.
- Collect all the documents in the technical file: including risk assessment and this manual.
- Know the possible safety risks before operating and using the equipment.

2.Environmental Considerations

- For the first use,please read this manual carefully to understand the basic operating content and operating specification.
- For remote control operation, select a relatively open area to use SCOUT2.0, because SCOUT2.0 is not equipped with any automatic obstacle avoidance sensor.
- Use SCOUT2.0 always under -10°C~45°C ambient temperature.
- If SCOUT 2.0 is not configured with separate custom IP protection, its water and dust protection will be IP22 ONLY.

3.Pre-work Checklist

- Make sure each device has sufficient power.
- Make sure Bunker 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.

4.Operation

- In remote control operation, make sure the area around is relatively spacious.
- Carry out remote control within the range of visibility.
- The maximum load of SCOUT2.0 is 50KG. When in use, ensure that the payload does not exceed 50KG.
- When installing an external extension on SCOUT2.0, confirm the position of the center of mass of the extension and make sure it is at the center of rotation.
- Please charge in tine when the device is low battery alarm.
- When SCOUT2..0 has a defect, please immediately stop using it to avoid secondary damage.
- When SCOUT2.0 has had a defect, please contact the relevant technical to deal with it, do not handle the defect by yourself.
- Always use SCOUT2.0 in the environment with the protection level requires for the equipment.
- Do not push SCOUT2.0 directly.
- When charging, make sure the ambient temperature is above 0 °C.
- If the vehicle shakes during its rotation, adjust the suspension.

5.Maintenance

- Regularly check the pressure of the tire, and keep the tire pressure between 1.8bar~2.0bar.
- If the tire is severely worn or burst, please replace it in time.
- If the battery do not use for a long time, it need to charge the battery periodically in 2 to 3 months.

CONTENTS

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
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
USB to CAN communication module	X1

1.2 Tech specifications

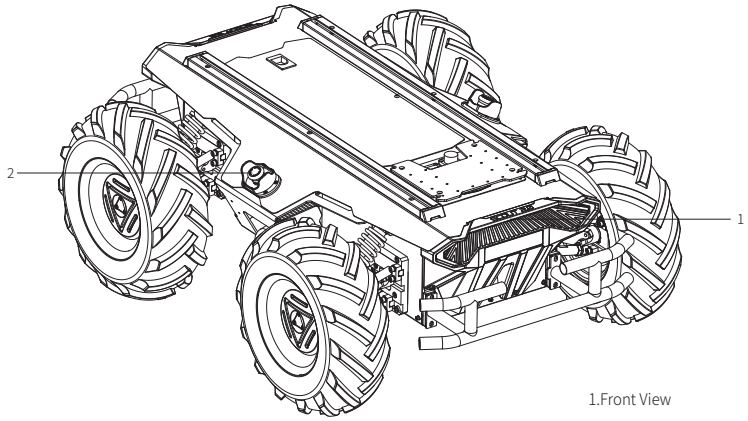
Parameter Types	Items	Values
Mechanical specifications	L × W × H (mm)	9930 X 699 X 349
	Wheelbase (mm)	498
	Front/rear wheel base (mm)	582 / 582
	Weight of vehicle body (kg)	68(±0.5)
	Battery type	Lithium battery 24V 30AH
	Motor	DC brushless 4 X 400W
	Reduction gearbox	1:30
	Drive type	Independent four-wheel 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

1.3Requirement for development

FS RC transmitter is provided (optional) in the factory setting pf 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.

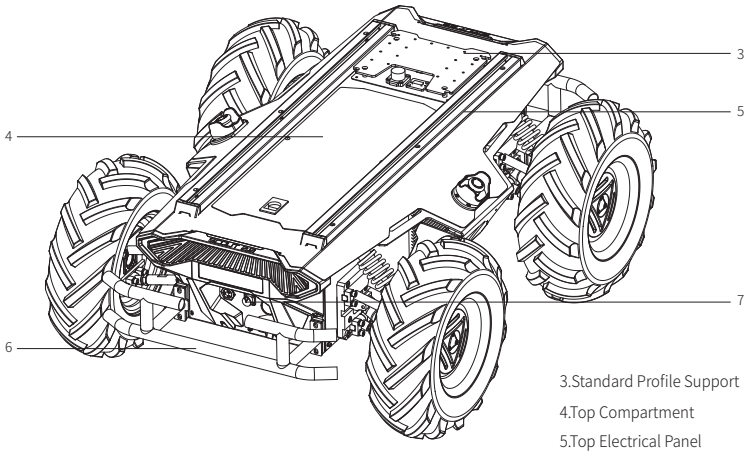
2 The Basics

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



1.Front View
2.Stop Switch

Figure 2.1 Front View



3.Standard Profile Support
4.Top Compartment
5.Top Electrical Panel
6.Retardant-collision Tube
7.Rear Panel

Figure 2.2 Rear View

SCOUT2.0 adopts a modular and intelligent design concept. The composite design of inflate rubber tyre and independent suspension on the power module, coupled with the powerful DC brushless servo motor, makes the SCOUT2.0 robot chassis development platform has strong pass ability and ground adapt ability, and can move flexibly on different ground.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 three 4-pin aviation connectors and one DB9 (RS232) connector.
The position of the top aviation connector is shown in Figure 2.3.

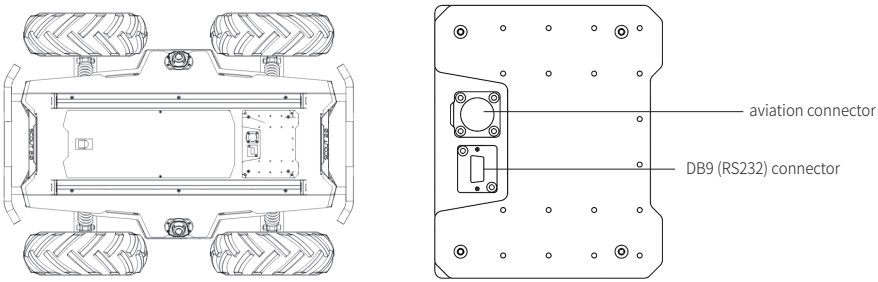
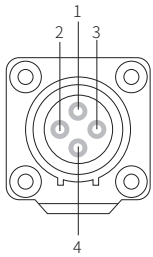


Figure 2.3 Schematic Diagram of SCOUT 2.0 Electrical Interface on Top

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.

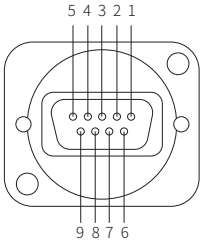
It should be noted that, the extended power supply here is internally controlled, which means the power supply will be actively cut off once the battery voltage drops below the pre-specified threshold voltage. Therefore, users need to notice that SCOUT 2.0 platform will send a low voltage alarm before 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	Power	GND	Power negative
3	CAN	CAN_H	CAN bus high
4	CAN	CAN_L	CAN bus low

Figure 2.4 Definitions for Pins of Top Aviation Extension Interface

Specific definitions for pins of Q4 are shown in Figure 2.5.



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

Figure 2.5 Illustration Diagram of Q4 Pins

2.2.2 Rear electrical interface

The extension interface at rear end is shown in Figure 2.6, 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.

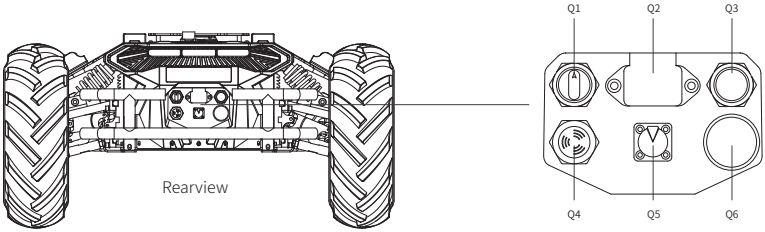
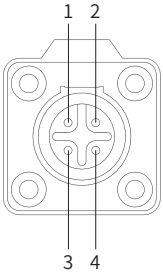


Figure 2.6 Rear View



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

Figure 2.7 Description of Front and Rear Aviation Interface Pins

2.3 Instructions on remote control
FS_i6_S remote control instructions

FS RC transmitter is an optional accessory of SCOUT2.0 for manually controlling the robot. The transmitter comes with a left-hand-throttle configuration. The definition and function shown in Figure 2.8. The function of the button is defined as: SWA and SWD are temporarily disabled, and SWB is the control mode select button, dial to the top is command control mode, dial to the middle is remote control mode; SWC is light control button; S1 is throttle button, control SCOUT2.0 forward and backward; S2 control is control the rotation, and POWER is the power button, press and hold at the same time to turn on.

Figure 2.8 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 2.9 in accordance with ISO 8855.

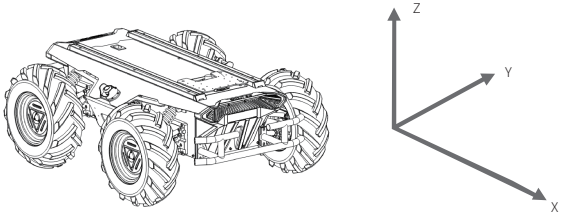


Figure 2.9 Schematic Diagram of Reference Coordinate System for Vehicle Body

As shown in Figure 2.9, the vehicle body of SCOUT 2.0 is in parallel with X axis of the established reference coordinate system. In the remote control mode, push the remote control stick S1 forward to move in the positive X direction, push S1 backward to move in the negative X direction. When S1 is pushed to the maximum value, the movement speed in the positive X direction is the maximum, When pushed S1 to the minimum, the movement speed in the negative direction of the X direction is the maximum; the remote control stick S2 controls the steering of the front wheels of the car body, push S2 to the left, and the vehicle turns to the left, pushing it to the maximum, and the steering angle is the largest, S2 Push to the right, the car will turn to the right, and push it to the maximum, at this time the right steering angle is the largest. In the control command mode, the positive value of the linear velocity means movement in the positive direction of the X axis, and the negative value of the linear velocity means movement in the negative direction of the X axis; The positive value of the angular velocity means the car body moves from the positive direction of the X axis to the positive direction of the Y axis, and the negative value of the angular velocity means the car body moves from the positive direction of the X axis to the negative direction of the Y axis.

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 through the SWC. Mode control description: the SWC lever is at the bottom of the normally closed mode, the middle is for the normally open mode, the top is breathing light mode.

- 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 SCOUT 2.0. Check whether there are significant anomalies; if so, please contact the after-sale service personal for support;
- Check the state of emergency-stop switches. Make sure both emergency stop buttons are released;

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.

3.2.1 Charging operation

- Make sure the electricity of SCOUT 2.0 chassis is powered off. Before charging, please make sure the power switch in the rear control condole is turned off;
- Insert the charger plug into Q6 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.

3.2.2 Battery replacement

SCOUT2.0 adopts a detachable battery solution for the convenience of users. In some special cases, the battery can be replaced directly. The operation steps and diagrams are as follows (before operation, ensure that SCOUT2.0 is power-off):

- Open the upper panel of SCOUT2.0, and unplug the two XT60 power connectors on the main control board (the two connectors are equivalent) and the battery CAN connector;
- Hang SCOUT2.0 in midair, unscrew eight screws from the bottom with a national hex wrench, and then drag the battery out;
- Replace the battery and fixed the bottom screws.
- Plug the XT60 interface and the power CAN interface into the main control board, confirm that all the connecting lines are correct, and then power on to test.

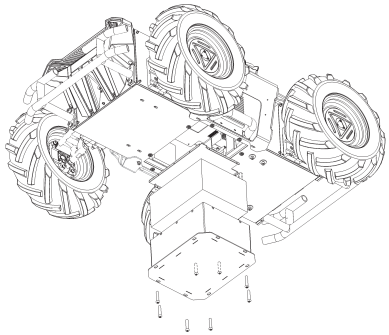


Figure 3.1 Schematic Diagram of replace battery

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 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 cable connection

SCOUT2.0 deliver with two aviation male plugs as shown in Figure 3.2. For wire definitions, please refer to Table 2.2.

Figure 3.2 Schematic diagram of aviation plug male connector

3.3.2 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.

3.3.3 CAN message protocol

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.

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	0×00 Standby mode 0×01 CAN command control mode 0×02 Serial port control mode 0×03 Remote control mode	
byte [2] byte [3]	Battery voltage higher 8 bits Battery voltage lower 8 bits	unsigned int16	Actual voltage × 10 (with an accuracy of 0.1V)	
byte [4]	Reserved	-	0×00	
byte [5]	Failure information	unsigned int8	Refer to Table 3.2 [Description of Failure Information]	
byte [6]	Reserved	-	0×00	
byte [7]	Count paritybit (count)	unsigned int8	0-255 counting loops, which will be added once every command sent	

Table 3.2 Description of Failure Information

Description of Failure Information		
Byte	bit	Meaning
byte [4]	bit [0]	Battery undervoltage fault (0: No failure 1: Failure) Protection voltage is 22V (The battery version with BMS, the protection power is 10%)
	bit [1]	Battery undervoltage fault[2] (0: No failure 1: Failure) Alarm voltage is 24V (The battery version with BMS, the warning power is 15%)
	bit [2]	RC transmitter disconnection protection (0: Normal 1: RC transmitter disconnected)
	bit [3]	No.1 motor communication failure (0: No failure 1: Failure)
	bit [4]	No.2 motor communication failure (0: No failure 1: Failure)
	bit [5]	No.3 motor communication failure (0: No failure 1: Failure)
	bit [6]	No.4 motor communication failure (0: No failure 1: Failure)
	bit [7]	Reserved, default 0

Note[1]: Robot chassis firmware version V1.2.8 is supported by subsequent versions, and the previous version requires firmware upgrade to support

Note[2]: The buzzer will sound when the battery under-voltage, but the chassis control will not be affected, and the power output will be cut off after the under-voltage fault

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	0x221	20ms	None
Date length	0×08			
Position	Function	Data type	Description	
byte [0]	Moving speed higher 8 bits	signed int16	Actual speed × 1000 (with an accuracy of 0.001rad)	
byte [1]	Moving speed lower 8 bits			
byte [2]	Rotation speed higher 8 bits	signed int16	Actual speed × 1000 (with an accuracy of 0.001rad)	
byte [3]	Rotation speed lower 8 bits			
byte [4]	Reserved	-	0x00	
byte [5]	Reserved	-	0x00	
byte [6]	Reserved	-	0x00	
byte [7]	Reserved	-	0x00	

The control frame includes control openness of linear speed and control openness of angular speed. 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	0x111	20ms	500ms
Date length	0×08			
Position	Function	Data type	Description	
byte [0]	Linear speed higher 8 bits	signed int16	Vehicle moving speed, unit mm/s (effective value+ -1500)	
byte [1]	Linear speed lower 8 bits			
byte [2]	Angular speed higher 8 bits	signed int16	Vehicle rotation angular speed, unit mm/s (effective value+ -1500)	
byte [3]	Angular speed lower 8 bits			
byte [4]	Reserved	—	0x00	
byte [5]	Reserved	—	0x00	
byte [6]	Reserved	—	0x00	
byte [7]	Reserved	—	0x00	

The mode setting frame is used to set the control interface of the terminal. For its detailed content of protocol, please refer to Table 3.5.

Table 3.5 Control mode setting frame

Command Name	Control Mode Setting Command			
Sending node	Receiving node	ID	Cycle (ms)	Receive-timeout (ms)
Decision-making control unit	Chassis node	0×421	None	None
Date length	0×01			
Position	Function	Date type	Description	
byte [0]	Control mode	unsigned int8	0×00 Standby mode 0×01 CAN command mode enable	

Description of control mode: In case the SCOUT 2.0 is powered on and the RC transmitter is not connected, the control mode is defaulted to standby mode. At this time, the chassis only receives control mode command, and does not respond other commands. To use CAN for control need to switch CAN command mode at first. If the RC transmitter is turned on, the RC transmitter has the highest authority, can shield the control of command and switch the control mode.

Status setting frame is use to clear the system errors. For its detailed content of protocol, please refer to Table 3.6.

Table 3.6 Status Setting Frame

Command Name	Status Setting Command			
Sending node	Receiving node	ID	Cycle (ms)	Receive-timeout (ms)
Decision-making control unit	Chassis node	0×441	None	None
Date length	0×01			
Position	Function	Date type	Description	
byte [0]	Errors clearing command	unsigned int8	0×00 Clear all failure 0×01 Clear Motor 1 failure 0×02 Clear Motor 2 failure 0×03 Clear Motor 3 failure 0×04 Clear Motor 4 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]
0x00	0x96	0x00	0x00	0x00	0x00	0x00	0x00

2.The vehicle steering 0.2rad/s

byte [0]	byte [1]	byte [2]	byte [3]	byte [4]	byte [5]	byte [6]	byte [7]
0x00	0x00	0x00	0xc8	0x00	0x00	0x00	0x00

The chassis status information will be feedback, and 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 motor numbers of the 4 motors in the chassis are shown in the figure below:

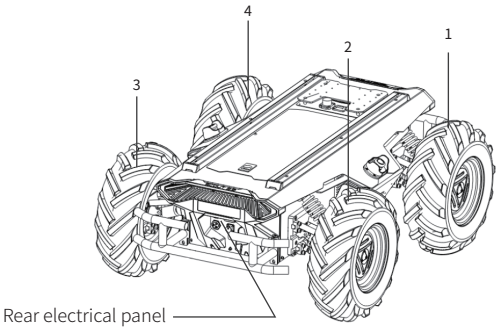


Figure 3.0 Schematic diagram Motor feedback ID

Table 3.7 Motor Speed Current Position Information Feedback

Command Name	Motor Drive High Speed Information Feedback Frame			
Sending node	Receiving node	ID	Cycle (ms)	Receive-timeout (ms)
Steer-by-wire chassis	Decision-making control unit	0x251~0x254	20ms	None
Date length	0×08			
Position	Function	Data type	Description	
byte [0]	Motor speed higher 8 bits	signed int16	Vehicle moving speed, unit mm/s (effective value+ -1500)	
byte [1]	Motor speed lower 8 bits			
byte [2]	Motor current higher 8 bits	signed int16	Motor current Unit 0.1A	
byte [3]	Motor current lower 8 bits			
byte [4]	Position highest bits	signed int32	Current position of the motor Unit: pulse	
byte [5]	Position second-highest bits			
byte [6]	Position second-lowest bits			
byte [7]	Position lowest bits			

Table 3.8 Motor temperature, voltage and status information feedback

Command Name	Motor Drive Low Speed Information Feedback Frame			
Sending node	Receiving node	ID	Cycle (ms)	Receive-timeout (ms)
Steer-by-wire chassis	Decision-making control unit	0x261~0x264	20ms	None
Date length	0×08			
Position	Function	Data type	Description	
byte [0]	Drive voltage higher 8 bits	unsigned int16	Current voltage of drive unit 0.1V	
byte [1]	Drive voltage lower 8 bits			
byte [2]	Drive temperature higher 8 bits	signed int16	Unit 1°C	
byte [3]	Drive temperature lower 8 bits			
byte [4]	Motor temperature	signed int8	Unit 1°C	
byte [5]	Drive status	unsigned int8	See the details in [Drive control status]	
byte [6]	Reserved	-	0x00	
byte [7]	Reserved	-	0x00	

Table 3.9 Drive Status

Byte	Bit	Description
byte[5]	bit[0]	Whether the power supply voltage is too low (0:Normal 1:Too low)
	bit[1]	Whether the motor is overheated (0:Normal 1:Overheated)
	bit[2]	Whether the drive is over current (0:Normal 1:Over current)
	bit[3]	Whether the drive is overheated (0:Normal 1:Overheated)
	bit[4]	Sensor status (0:Normal 1:Abnormal)
	bit[5]	Drive error status (0:Normal 1:Error)
	bit[6]	Drive enable status (0:Normal 1:Disability)
	bit[7]	Reserved

The front and external lights also support command control. The following table shows the control commands:

Table 3.10 Light Control Frame

Command Name	Light Control Frame			
Sending node	Receiving node	ID	Cycle (ms)	Receive-timeout (ms)
Decision-making control unit	Chassis node	0x121	20ms	None
Date length	0×08			
Position	Function	Date type	Description	
byte [0]	Light 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 100refers to maximum brightness[5]	
byte [3]	Rear light mode	unsigned int8	0x00 NC 0x01 NO 0x02BL mode 0x03 User-defined brightness	
byte [4]	Customize brightness for rear light	unsigned int8	[0, 100], where 0 refers to no brightness 100refers to maximum brightness[5]	
byte [5]	Reserved	–	0x00	
byte [6]	Reserved	–	0x00	
byte [7]	Parity bit (checksum)	unsigned int8	0 - 255 counting loops, which will be added once every command sent	

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

Table 3.11 Light Control Feedback Frame

Command Name	Steering Zero Setting Command			
Sending node	Receiving node	ID	Cycle (ms)	Receive-timeout (ms)
Steer-by-wire chassis	Decision-making control unit	0x231	20ms	None
Date length	0×08			
Position	Function	Date 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 0x02BL mode 0x03 User-defined brightness	
byte [2]	Current custom brightness of front light	unsigned int8	[0, 100], where 0 refers to no brightness 100refers to maximum brightness	
byte [3]	Current rear light mode	unsigned int8	0x00 NC 0x01 NO 0x02BL mode 0x03 User-defined brightness	
byte [4]	Current custom brightness of rear light	unsigned int8	[0, 100], where 0 refers to no brightness 100refers to maximum brightness	
byte [5]	Reserved	–	0x00	
byte [6]	Reserved	–	0x00	
byte [7]	Parity bit (checksum)	unsigned int8	0 - 255 counting loops, which will be added once every command sent	

Table 3.12 System Version Information Enquiry Frame

Command Name	System Version Information Enquiry Command			
Sending node	Receiving node	ID	Cycle (ms)	Receive-timeout (ms)
Decision-making control unit	Chassis node	0x411	None	None
Date length	0×01			
Position	Function	Date type	Description	
byte [0]	Enquire system version	unsigned int8	Constant 0×01	

Table 3.13 System Version Information Enquiry Frame

Command Name	System Version Information Feedback Frame			
Sending node	Receiving node	ID	Cycle (ms)	Receive-timeout (ms)
Steer-by-wire chassis	Decision-making control unit	0x41A	500ms	None
Date length	0×08			
Position	Function	Data type	Description	
byte [0]	The number of main control hardware version higher 8 bits	unsigned int16	Higher 8 bits is the main version number,	
byte [1]	The number of main control hardware version lower 8 bits		lower 8 bits is the second version number	
byte [2]	The number of drive hardware version higher 8 bits	unsigned int16	Higher 8 bits is the main version number,	
byte [3]	The number of drive hardware version lower 8 bits		lower 8 bits is the second version number	
byte [4]	The number of main control software version higher 8 bits	unsigned int16	Higher 8 bits is the main version number,	
byte [5]	The number of main control software version lower 8 bits		lower 8 bits is the second version number	
byte [6]	The number of drive software version higher 8	unsigned int16	Higher 8 bits is the main version number,	
byte [7]	The number of drive software version lower 8		lower 8 bits is the second version number	

Table 3.14 Mileometer Information Feedback

Command Name	Mileometer Information Feedback			
Sending node	Receiving node	ID	Cycle (ms)	Receive-timeout (ms)
Steer-by-wire chassis	Decision-making control unit	0x311	20ms	None
Date length	0×08			
Position	Function	Data type	Description	
byte [0]	Left wheel mileometer highest bit	signed int32	Chassis left wheel mileometer feedback Unit:mm	
byte [1]	Left wheel mileometer second-highest bit			
byte [2]	Left wheel mileometer second-highest bit			
byte [3]	Left wheel mileometer lowest bit			
byte [4]	Right wheel mileometer highest bit	signed int32	Chassis right wheel mileometer feedback Unit:mm	
byte [5]	Right wheel mileometer second-highest bit			
byte [6]	Right wheel mileometer second-highest bit			
byte [7]	Right wheel mileometer lowest bit			

Table 3.15 Remote Control Information Feedback

Command Name	Remote Control Information Feedback Frame			
Sending node	Receiving node	ID	Cycle (ms)	Receive-timeout(ms)
Steer-by-wire chassis	Decision-making control unit	0x241	20ms	None
Date length	0×08			
Byte	Function	Data type	Description	
byte[0]	SW feedback	unsigned int8	bit[0-1]: SWA:2- Up 3-Down bit[2-3]: SWB : 2-Up 1-Middle 3-Down bit[4-5]: SWC : 2-Up 1-Middle 3-Down bit[6-7]: SWD：2-Up 3-Down	
byte[1]	Right joystick left and right	signed int8	Range[-100,100]	
byte[2]	Right joystick up and down	signed int8	Range[-100,100]	
byte[3]	Left joystick up and down	signed int8	Range[-100,100]	
byte[4]	Left joystick left and right	signed int8	Range[-100,100]	
byte[5]	Left knob VRA	signed int8	Range[-100,100]	
byte[6]	Reserved	--	0x00	
byte[7]	Count Parity bit	unsigned int8	0~255 Loops counting	

Command Name	BMS Data Feedback			
Sending node	Receiving node	ID	Cycle (ms)	Receive-timeout(ms)
Steer-by-wire chassis	Decision-making control unit	0x361	500ms	None
Date length	0×08			
Position	Function	Data type	Description	
byte[0]	Battery SOC	unsigned int8	Range 0~100	
byte[1]	Battery SOH	unsigned int8	Range 0~100	
byte[2]	Battery voltage higher 8 bits	unsigned int16	Unit: 0.01V	
byte[3]	Battery voltage lower 8 bits	signed int8	Range[-100,100]	
byte[4]	Battery current higher 8 bits	signed int16	Unit: 0.1A	
byte[5]	Battery current lower 8 bits	signed int8	Range[-100,100]	
byte[6]	Battery temperature higher 8 bits	signed int16	Unit: 0.1℃	
byte[7]	Battery temperature lower 8 bits			

3.4 Serial Communication Protocol

3.4.1 Instruction of serial protocol

It is a standard for serial communication jointly formulated by the Electronic Industries Association (EIA) of the United States in 1970 conjunction with Bell Systems, modem manufacturers and computer terminal manufacturers. Its name is "Technical Standard for Serial Binary Data Exchange Interface Between Data Terminal Equipment (DTE) and Data Communication Equipment (DCE)". The standard stipulates that a 25-pin DB-25 connector is used for each connector. The signal content of each pin is specified, and the levels of various signals are also specified. Later, IBM's PC simplified RS232 into a DB-9 connector, which became the practical standard. The RS-232 port of industrial control generally only uses three lines of RXD, TXD, and GND.

3.4.2 Serial Connection

Use the USB to RS232 serial cable in our communication tool to connect to the serial port at the rear of the car, use the serial tool to set the corresponding baud rate, and use the sample data provided above to test. If the remote control is turned on, it is necessary to switch the remote control to command control mode. If the remote control is not turned on, just send the control command directly. It should be noted that the command must be sent periodically. If the chassis exceeds 500MS and the serial port command is not received, it will enter the loss of connection protection. status.

3.4.3 Serial Protocol Content

Basic Communication Parameter

Item	Parameter
Baud Rate	115200
Parity	No test
Data bit length	8 bits
Stop bit	1 bit

Instruction of protocol

Start bit		Frame length	Command type	Command ID	Data field			Frame ID	Checksum composition
SOF		frame_L	CMD_TYPE	CMD_ID	data	...	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 the start bit, frame length, frame command type, command ID, data range, frame ID, and checksum. The frame length refers to the length excluding the start bit and the checksum. The checksum is the sum of all data from the start bit to the frame ID; the frame ID bit is from 0 to 255 counting loops, which will be added once every command sent.

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

Figure 3.3 Serial Parity algorithm code example

Protocol Content

Command Name	System Status Feedback Frame		
Sending node	Receiving node	Cycle (ms)	Receive-timeout (ms)
Steer-by-wire chassis	Decision-making control unit	100ms	None
Frame length	0×0C		
Command type	Feedback command(0×AA)		
Command ID	0×01		
Data length	8		
Position	Function	Data type	Description
byte [0]	Current status of vehicle body	unsigned int8	0×00 System in normal condition 0×01 Emergency stop mode (not enabled) 0×02 System exception
byte [1]	Mode control	unsigned int8	0×00 Standby mode 0×01 CAN command control mode 0×02 Serial control mode[1] 0×03 Remote control mode
byte [2] byte [3]	Battery voltage higher 8 bits Battery voltage lower 8 bits	unsigned int16	Actual voltage × 10 (with an accuracy of 0.1V)
byte [4]	Reserved	—	0×00
byte [5]	Failure information	unsigned int8	Refer to [Description of Failure Information]
byte [6]	Reserved	—	0×00
byte [7]	Reserved	—	0×00

Description of Failure Information		
Byte	Bit	Meaning
byte[5]	bit[0]	Battery under-voltage failure (0: No failure 1: Failure) Protection voltage is 22V (Battery with BMS version, protection power is 10%
	bit[1]	Battery under-voltage alarm (0: No alarm 1: Alarm) Alarm voltage is 24V (Battery with BMS version, alarm power is 15%
	bit[2]	Remote control loss contact protection (0: Normal 1: Remote control loss contact)
	bit[3]	Motor 1 communication failure (0: No failure 1: Failure)
	bit[4]	Motor 2 communication failure (0: No failure 1: Failure)
	bit[5]	Motor 3 communication failure (0: No failure 1: Failure)
	bit[6]	Motor 4 communication failure (0: No failure 1: Failure)
	bit[7]	Reserved, default 0

Note[1]: Robot chassis firmware version V1.2.8 is supported by subsequent versions, and the previous version requires firmware upgrade to support.

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	0×0C		
Command type	Feedback command (0×AA)		
Command ID	0×02		
Data length	8		
Position	Function	Data type	Description
byte [0]	Moving speed higher 8 bits	signed int16	Actual speed × 1000 (with an accuracy of 0.001rad)
byte [1]	Moving speed lower 8 bits		
byte [2]	Rotation speed higher 8 bits	signed int16	Actual speed × 1000 (with an accuracy of 0.001rad)
byte [3]	Rotation speed lower 8 bits		
byte [4]	Reserved	-	0×00
byte [5]	Reserved	-	0×00
byte [6]	Reserved	-	0×00
byte [7]	Reserved	-	0×00

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	0×0A		
Command type	Control command (0×55)		
Command ID	0×01		
Data length	6		
Position	Function	Data type	Description
byte [0]	Movement speed higher 8 bits	signed int16	Vehicle moving speed, unit: mm/s
byte [1]	Movement speed lower 8 bits		
byte [2]	Rotation speed higher 8 bits	signed int16	Vehicle rotation angular speed, unit: 0.001rad/s
byte [3]	Rotation speed lower 8 bits		
byte [4]	Reserved	-	0x00
byte [5]	Reserved	-	0x00

Control Mode Setting Command

Command Name	Control Mode Setting Command		
Sending node	Receiving node	Cycle (ms)	Receive-timeout (ms)
Decision-making control unit	Chassis node	None	None
Frame length	0×05		
Command type	Control command (0×55)		
Command ID	0×02		
Data length	1		
Position	Function	Date type	Description
byte [0]	Serial control mode enable	unsigned int8	0×00 Standby mode 0×02 Serial mode enable Power-on enters standby mode default

Status Setting Command

Command Name	Status Setting Command		
Sending node	Receiving node	Cycle (ms)	Receive-timeout (ms)
Decision-making control unit	Chassis node	None	None
Frame length	0×05		
Command type	Control command (0×55)		
Command ID	0×03		
Data length	1		
Position	Function	Date type	Description
byte [0]	Errors clearing command	unsigned int8	0×00 Clear all not serious failure 0×01 Clear motor 1 failure 0×02 Clear motor 2 failure 0×03 Clear motor 3 failure 0×04 Clear motor 4 failure

Motor Drive High Speed Information Feedback Frame

Command Name	Motor Drive High Speed Information Feedback Frame		
Sending node	Receiving node	Cycle (ms)	Receive-timeout (ms)
Steer-by-wire chassis	Chassis node	20ms	None
Frame length	0×0C		
Command type	Feedback command (0×AA)		
Command ID	0×03~0×06		
Data length	8		
Position	Function	Data type	Description
byte[0]	Motor speed higher 8 bits	signed int16	Current speed of the motor Unit: RPM
byte[1]	Motor speed lower 8 bits		
byte[2]	Motor current higher 8 bits	unsigned int16	Motor current Unit: 0.1A
byte[3]	Motor current lower 8 bits		
byte[4]	Position highest bits	signed int32	Current position of the motor Unit: pulse
byte[5]	Position second-highest bits		
byte[6]	Position second-lowest bits		
byte[7]	Position lowest bits		

Motor Drive Low Speed Information Feedback Frame

Command Name	Motor Drive Low Speed Information Feedback Frame		
Sending node	Receiving node	Cycle (ms)	Receive-timeout (ms)
Steer-by-wire chassis	Decision-making control unit	100ms	None
Frame length	0×0C		
Command type	Feedback command (0×AA)		
Command ID	0×07~0×a		
Data length	6		
Position	Function	Data type	Description
byte [0]	Drive voltage higher 8 bits	unsigned int16	Current voltage of drive Unit: 0.1V
byte [1]	Drive voltage lower 8 bits		
byte [2]	Drive temperature higher 8 bits	signed int16	Unit: 1°C
byte [3]	Drive temperature lower 8 bits		
byte [4]	Motor temperature	signed int8	Unit: 1°C
byte [5]	Drive status	unsigned int8	See the details in the table below
byte [6]	Reserved	-	0×00
byte [7]	Reserved	-	0×00

Byte	Bit	Description
byte[5]	bit[0]	Whether the power supply voltage is too low (0:Normal 1:Too low)
	bit[1]	Whether the motor is overheated (0:Normal 1:Overheated)
	bit[2]	Whether the drive is over current (0:Normal 1:Over current)
	bit[3]	Whether the drive is overheated (0:Normal 1:Overheated)
	bit[4]	Sensor status (0:Normal 1:Abnormal)
	bit[5]	Drive error status (0:Normal 1:Error)
	bit[6]	Drive enable status (0:Normal 1:Disability)
	bit[7]	Reserved

Light Control Frame

Command Name	Light Control Frame		
Sending node	Receiving node	Cycle (ms)	Receive-timeout (ms)
Decision-making control unit	Chassis node	20ms	500ms
Frame length	0×0A		
Command type	Control command (0×55)		
Command ID	0×04		
Data length	6		
Position	Function	Date type	Description
byte [0]	Light 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 0x02BL 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

Light Control Feedback Frame

Command Name	Light Control Feedback Frame		
Sending node	Receiving node	Cycle (ms)	Receive-timeout (ms)
Decision-making control unit	Chassis node	500ms	500ms
Frame length	0×0A		
Command type	Control command (0×AA)		
Command ID	0×a1		
Data length	8		
Position	Function	Date type	Description
byte [0]	Current light 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]	Custom brightness of front light	unsigned int8	[0, 100], where 0 refers to no brightness, 100refers to maximum brightness[5]
byte [3]	Rear light mode	unsigned int8	0x00 NC 0x01 NO 0x02BL 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]	Reserved	—	0x00
byte [7]	Reserved	—	0x00

Mileage Feedback Frame

指令名称	Mileage Feedback Frame		
Sending node	Receiving node	Cycle (ms)	Receive-timeout(ms)
Steer-by-wire chassis	Decision-making control unit	20ms	None
Frame length	0×0C		
Command type	Feedback command (0×AA)		
Command ID	0×a2		
Data length	8		
Byte	Function	Data type	Description
byte [0]	Left wheel mileometer highest bit	signed int32	Chassis left wheel mileometer feedback, unit:mm
byte [1]	Left wheel mileometer second-highest bit		
byte [2]	Left wheel mileometer second-lowest bit		
byte [3]	Left wheel mileometer lowest bit		
byte [4]	Right wheel mileometer highest bit	signed int32	Chassis right wheel mileometer feedback, unit:mm
byte [5]	Right wheel mileometer second-highest bit		
byte [6]	Right wheel mileometer second-lowest bit		
byte [7]	Right wheel mileometer lowest bit		

Version Information Enquiry Frame

Command Name	System Version Information Enquiry Command		
Sending node	Receiving node	Cycle (ms)	Receive-timeout (ms)
Decision-making control unit	Chassis node	None	None
Date length	0×05		
Command type	Control command (0×55)		
Command ID	0×05		
Data length	1		
Position	Function	Date type	Description
byte [0]	Enquire system version	unsigned int8	Constant 0×01

Version Information Feedback Frame

Command Name	System Version Information Feedback Frame		
Sending node	Receiving node	Cycle (ms)	Receive-timeout (ms)
Steer-by-wire chassis	Decision-making control unit	20ms	None
Frame length	0×0C		
Command type	Control command (0×AA)		
Command ID	0×a3		
Data length	8		
Position	Function	Data type	Description
byte [0] byte [1]	The number of main control hardware version higher 8 bits The number of main control hardware version lower 8 bits	unsigned int16	Higher 8 bits is the main version number, lower 8 bits is the second version number
byte [2] byte [3]	The number of drive hardware version higher 8 bits The number of drive hardware version lower 8 bits	unsigned int16	Higher 8 bits is the main version number, lower 8 bits is the second version number
byte [4] byte [5]	The number of main control software version higher 8 bits The number of main control software version lower 8 bits	unsigned int16	Higher 8 bits is the main version number, lower 8 bits is the second version number
byte [6] byte [7]	The number of drive software version higher 8 The number of drive software version lower 8	unsigned int16	Higher 8 bits is the main version number, lower 8 bits is the second version number

Remote Control Information Feedback Frame

Command Name	Remote Control Information Feedback Frame		
Sending node	Receiving node	Cycle (ms)	Receive-timeout(ms)
Steer-by-wire chassis	Decision-making control unit	20ms	None
Frame length	0×0C		
Command type	Feedback command (0×AA)		
Command ID	0×a4		
Data length	8		
Byte	Function	Data type	Description
byte [0]	SW feedback	unsigned int8	bit[0-1]: SWA:2- Up 3-Down bit[2-3]: SWB : 2-Up 1-Middle 3-Down bit[4-5]: SWC : 2-Up 1-Middle 3-Down bit[6-7]: SWD：2-Up 3-Down
byte [1]	Right joystick left and right	signed int8	Range[-100,100]
byte [2]	Right joystick up and down	signed int8	Range[-100,100]
byte [3]	Left joystick up and down	signed int8	Range[-100,100]
byte [4]	Left joystick left and right	signed int8	Range[-100,100]
byte [5]	Left knob VRA	signed int8	Range[-100,100]
byte [6]	Reserved	--	0x00
byte [7]	Reserved	--	0x00

BMS Feedback Frame

Command Name	BMS Data Feedback		
Sending node	Receiving node	Cycle (ms)	Receive-timeout (ms)
Steer-by-wire chassis	Decision-making control unit	20ms	None
Frame length	0×0C		
Command type	Feedback command (0×AA)		
Command ID	0×a5		
Data length	8		
Position	Function	Data type	Description
byte[0]	Battery SOC	unsigned int8	Range 0~100
byte[1]	Battery SOH	unsigned int8	Range 0~100
byte[2] byte[3]	Battery voltage higher 8 bits Battery voltage lower 8 bits	unsigned int16	Unit: 0.01V
byte[4] byte[5]	Battery current higher 8 bits Battery current lower 8 bits	signed int16	Unit: 0.1A
byte[6] byte[7]	Battery temperature higher 8 bits Battery temperature lower 8 bits	signed int16	Unit: 0.1°C

Example Data

By controlling the chassis to move forward at a linear speed of 0.15m/s, the following is the specific data content.

Start bit		Frame Length	Command Type	Command ID	Data Range			Frame ID	Checksum
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 following is the content of the data range:

Position	Function	Value
byte [0]	Linear speed higher 8 bits	0x00
byte [1]	Linear speed lower 8 bits	0x96
byte [2]	Angular speed higher 8 bits	0x00
byte [3]	Angular speed lower 8 bits	0x00
byte [4]	Reserved	0x00
byte [5]	Reserved	0x00

The entire data content is: 5A A5 0A 55 01 00 96 00 00 00 00 00 F5

- Other note: This protocol requires firmware version more than V1.5.12

3.5 Firmware upgrades

In order to facilitate users to upgrade the firmware version used by SCOUT 2.0 and bring customers a more complete experience, SCOUT 2.0 provides a firmware upgrade hardware interface and corresponding client software. A screenshot of this application is shown in Figure 3.3.

Upgrade preparation

- SERIAL CABLE × 1
- USB-TO-SERIAL PORT × 1
- SCOUT 2.0 CHASSIS × 1
- COMPUTER (WINDOWS OPERATING SYSTEM) × 1

Firmware upgrade software

- https://github.com/agilexrobotics/agilex_firmware

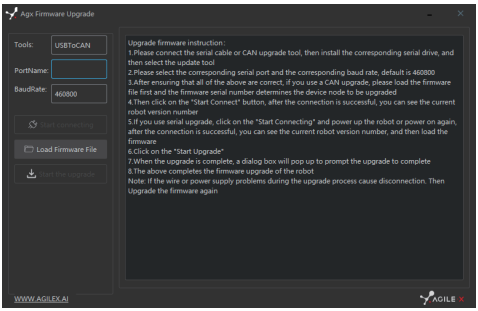


Figure 3.3 Client Interface of Firmware Upgrade

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 3s before power-on; if the waiting time is more than 3s, 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 turn the power off and on again.

3.6 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 SCOUT2.0 ROS Package

ROS provide some standard operating system services, such as hardware abstraction, low-level device control, implementation of common function, interprocess message and data packet management. ROS is based on a graph architecture, so that process of different nodes can receive, and aggregate various information (such as sensing, control, status, planning, etc.) Currently ROS mainly support UBUNTU.

Development Preparation

Hardware preparation

- CANlight can communication module ×1
- Thinkpad E470 notebook ×1
- AGILEX SCOUT 2.0 mobile robot chassis ×1
- AGILEX SCOUT 2.0 remote control FS-i6s ×1
- AGILEX SCOUT 2.0 top aviation power socket ×1

Use example environment description

- Ubuntu 16.04 LTS (This is a test version, tasted on Ubuntu 18.04 LTS)
- ROS Kinetic (Subsequent versions are also tested)
- Git

Hardware connection and preparation

- Lead out the CAN wire of the SCOUT 2.0 top aviation plug or the tail plug, and connect CAN_H and CAN_L in the CAN wire to the CAN_TO_USB adapter respectively;
- Turn on the knob switch on the SCOUT 2.0 mobile robot chassis, and check whether the emergency stop switches on both sides are released;
- Connect the CAN_TO_USB to the usb point of the notebook. The connection diagram is shown in Figure 3.4.

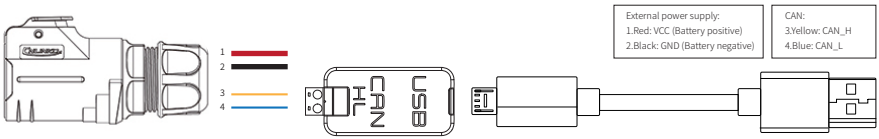


Figure 3.4 CAN connection diagram

ROS installation and environment setting

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

Test CANABLE hardware and CAN communication

Setting CAN-TO-USB adaptor

- Enable gs_usb kernel module
  $ sudo modprobe gs_usb
- Setting 500k Baud rate and enable can-to-usb adaptor
  $ sudo ip link set can0 up type can bitrate 500000
- If no error occurred in the previous steps, you should be able to use the command to view the can device immediately
  $ ifconfifig -a
- Install and use can-utils to test hardware
  $ sudo apt install can-utils
- If the can-to-usb has been connected to the SCOUT 2.0 robot this time, and the car has been turned on, use the following commands to monitor the data from the SCOUT 2.0 chassis
  $ candump can0
- Please refer to:
  [1]https://github.com/agilexrobotics/agx_sdk
  [2]https://wiki.rdu.im/_pages/Notes/Embedded-System/-Linux/can-bus-in-linux.html

AGILEX SCOUT 2.0 ROS PACKAGE download and compile

- Download ros package
  $ sudo apt install ros-$ROS_DISTRO-controller-manager
  $ sudo apt install ros-$ROS_DISTRO-teleop-twist-keyboard
  $ sudo apt install ros-$ROS_DISTRO-joint-state-publisher-gui
  $ sudo apt install libasio-dev
- Clone compile scout_ros code
  $ cd ~/catkin_ws/src
  $ git clone https://github.com/agilexrobotics/scout_ros.git
  $ git clone https://github.com/agilexrobotics/agx_sdk.git
  $ cd scout_ros && git checkout scout_v2
  $ cd ../agx_sdk && git checkout scout_v2
  $ cd ~/catkin_ws
  $ catkin_make
  Please refer to：https://github.com/agilexrobotics/scout_ros

Start the ROS node

- Start the based node
  $ roslaunch scout_bringup scout_minimal.launch
- Start the keyboard remote operation node
  $ roslaunch scout_bringup scout_teleop_keyboard.launch
- Start gazebo simulate node
  $ roslaunch scout_bringup scout_base_gazebosim.launch

4 Precautions

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

4.1Battery

- 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 -10°C to 45°C; 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 is -10°C to 45°C; please do not use it below -10°C and above 45°C ;
- 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°C;
- 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.

4.5 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.

4.4 Additional safety advice

- 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?

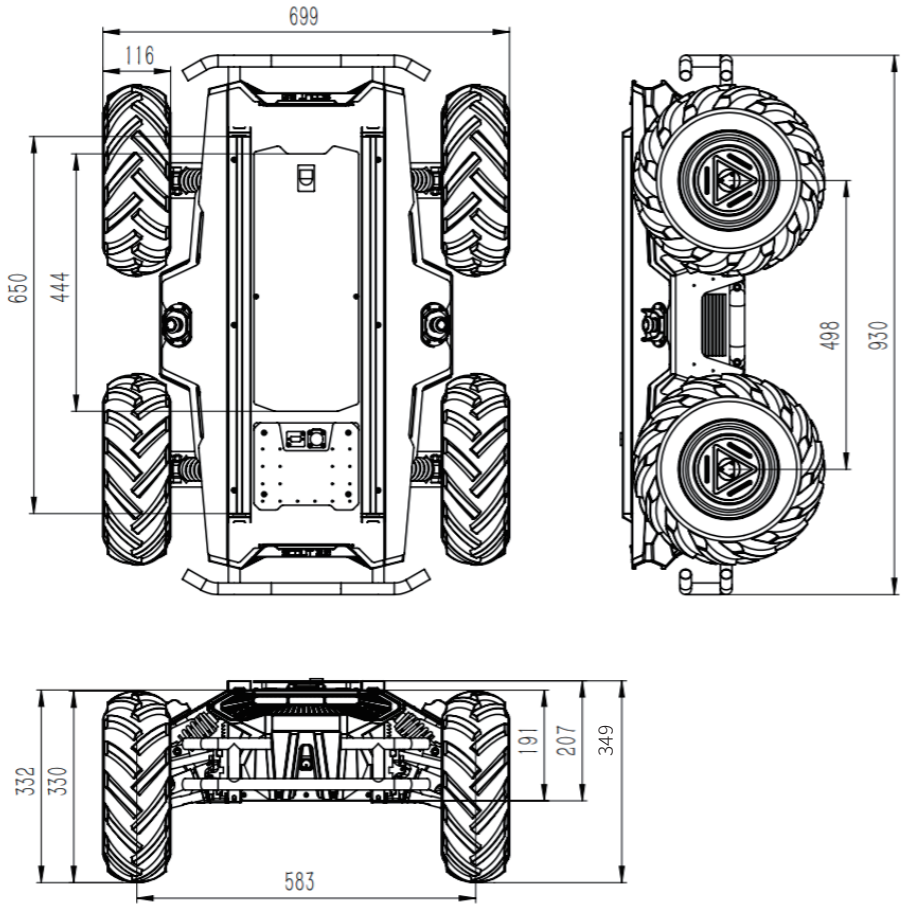
A: The tire wear of SCOUT 2.0 is normally seen when it is running. As SCOUT 2.0 is based on the four-wheel differential steering design, sliding friction and rolling friction both occur when the vehicle body rotates. If the floor is not smooth but rough, tire surfaces will be worn out. In order to reduce 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?

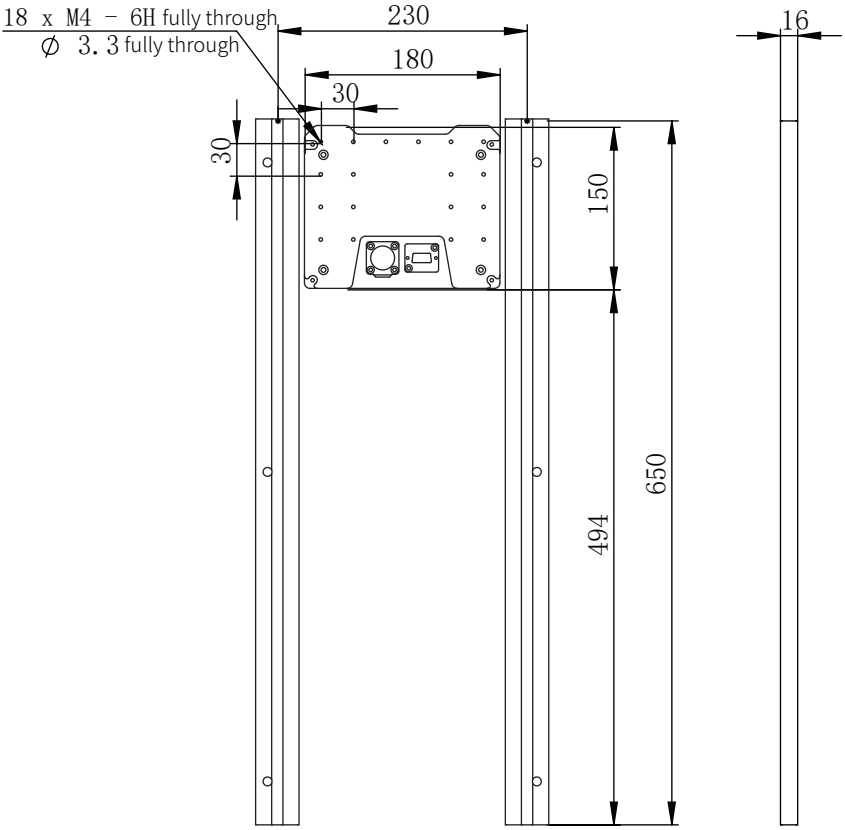
A: There is a communication protection mechanism inside SCOUT 2.0, which means the chassis is provided with timeout protection when processing external CAN control commands. Suppose the vehicle receives one frame of communication protocol, but it does no receive the next frame of control command after 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



AGILE·X
AgileX Robotics CO., Ltd
WWW.AGILEX.AI
Email：sales@agilex.ai
TEL:+86-769-22892150