Introduction

WAVE ROVER is a robust all-metal body 4WD mobile robot drive chassis with powerful off-road capabilities and earthquake resistance. Its source code is entirely open-source and available for secondary development. It supports various host devices (Raspberry Pi, Jetson Nano, Jetson Orin Nano, etc.) that communicate with the ESP32 slave computer through serial communication.

Equipped with an integrated 3S 18650 lithium battery UPS power module (3x 18650 lithium batteries in series (not included)), it provides a continuous power source for the robot while supporting charging and discharging simultaneously. The built-in multi-functional robot driver board can expand to support serial bus servos, PWM output, SD card, and other functions. The driver board is based on ESP32 and features onboard WiFi and Bluetooth.

It utilizes N20 reduction motors with high-quality gearboxes, which allows the mobile robot to drive at high speed with great power. The robot is equipped with soft rubber wheels, greatly reducing the impact from complex terrains, and allowing it to easily meet requirements for high-speed travel, shock absorption, and off-road capabilities. It also comes with an expansion platform that can install host devices (Raspberry Pi 4B, Jetson Nano, etc.), LD19/STL-27L LiDAR, and a pan-tilt camera, providing more possibilities for secondary development.

Features

- Equipped with N20 reduction motors that use high-quality gearboxes, enabling it to travel at a high speed of 1.25m/s.
- The flexible rubber tires significantly reduce the impact of complex terrains on the product.
- It features a 0.91-inch OLED screen for interactive purposes.
- The product is designed with a charging interface and automatic download circuit, allowing it to be used while charging.
- The UPS power module on board contains an INA219 acquisition chip, facilitating real-time monitoring of battery voltage and charging current.
- The UPS power module utilizes three 18650 batteries (not included) in series with a large capacity of 7800mAh, providing a higher output current and stronger motor power.
- It also offers 5V and 3.3V outputs for expanding other devices.
- The module includes a lithium battery protection circuit, offering functions such as overcharging, over-discharging, overcurrent, and short-circuit protection.
- The sample code is developed using Arduino IDE, eliminating the need for manual configuration of the compilation environment. Upon startup, the ESP32 automatically establishes a WIFI hotspot. Users can connect and log in to the control interface using a smartphone (Android/iOS) or a computer (Linux/Windows/Mac) with any Chromium-based browser, without requiring an app download.
- The slave ESP32 can be used to drive DC motors and serial bus servos and supports interfaces for OLED screens, TF card slots, a nine-axis IMU module, WiFi, and Bluetooth. Even without installing an upper-level device, it can be used independently.

- The system can be extended with various host devices, using serial communication to transmit control data in JSON format.
- The entire code is open-source and comes with online rich development documentation and tutorials.
- The open-source resource includes planar drawings and chassis structural drawings, including 3D models, facilitating secondary development.

WAVE ROVER User Manual

Note: Customers need to purchase and install three 18650 lithium batteries on their own before using the product. When connecting the batteries for the first time, please pay attention to whether the LED light is on. If the LED light is on, it indicates that the positive and negative poles of the batteries are reversed. Please check and ensure that the batteries are not reversed. Charging is prohibited if the batteries are connected in reverse, as it may cause a risk of explosion.

WAVE ROVER Basic Use

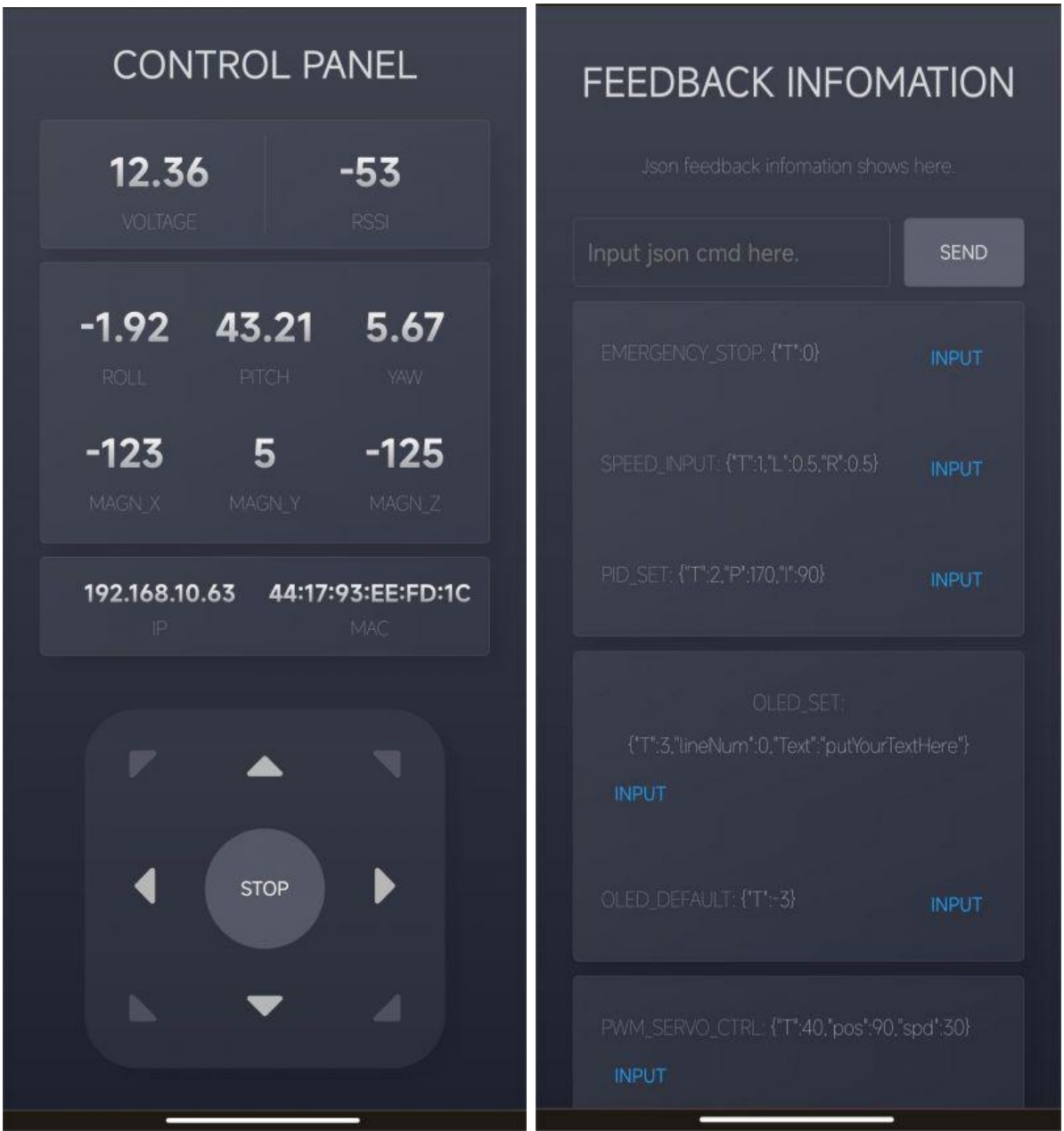
The driver board of WAVE ROVER is developed based on the ESP32 module. After starting up, ESP32 will give priority to connecting to known WIFI hotspots (the following tutorial will introduce how to set up known WIFI). If WIFI is not connected within 20 seconds after starting up, ESP32 will create a hotspot automatically.

Hotspot name: WAVE_ROVER

Hotspot password: 12345678

After connecting to this hotspot with your mobile phone, the mobile phone will prompt you that this hotspot has no network, and whether you are allowed to switch hotspots, please choose not to, keep the connection with the WAVE_ROVER hotspot.

Open the mobile browser (Google Chrome is recommended), visit the URL 192.168.4.1, and you can open the WAVE_ROVER control page. You can send commands in JSON format on this page to control and set the onboard serial bus servo device, get the serial bus servo feedback, obtain IMU information, control the angle of the PWM servo, set the content of the OLED display, etc., which is convenient for users to develop on the host compute.



The web interface will display the robot's voltage, signal strength (STA mode), heading angle, IP and MAC address, and other information in real time.

The movement of the robot can be controlled by the direction buttons, and the SLOW, MIDDLE, and FAST buttons below the direction buttons are used to select the moving speed of the robot.

The web page includes "heartbeat detection". After opening the web control interface, the web application will communicate with the robot continuously. If the connection is disconnected during the movement of the robot, the robot will automatically stop moving in a short time to avoid danger.

When using a tablet computer or PC to open the web control interface, you can use the WASD keys on the keyboard to control the robot. In order to avoid misuse, the first few shortcut keys after login are unresponsive, and you can start to control the robot by pressing the WASD keys once or several times in a cycle.

As the web application is completely open-source, you can change its interface and functions by changing WebPage.h. You can follow the customizable development tutorial we provide to learn.

WAVE ROVER's JSON Instruction Interaction Tutorial

What Is JSON?

JSON (JavaScript Object Notation) is an open standard file format and data interchange format that is easy for humans to read and write and can exchange data between multiple languages. It is also easy for machines to parse and generate.

Why Interact With JSON Commands and WAVE ROVER?

As WAVE ROVER boasts more onboard resources, rich demo functions, and is easy to connect external host computers, it is easily controlled by the host computer in more aspects. Hence, we can use JSON commands to interact with WAVE ROVER. Also, we can customize these functions based on the current framework to make them more suitable for your needs.

JSON Commands Interaction Method

We provide two interaction methods for WAVEROVER and the JSON commands:
1. Through the web application (or send a request through the program, which can be used for wireless JSON interaction with the host computer).
2. Through UART communication. The 40PIN expansion header on the driver board of WAVE ROVER has a corresponding RX and TX header, using a 3.3V logic level (applicable to host computers such as Raspberry Pi, Jetson nano, etc., not suitable for Arduino Uno with a serial port logic level of 5V.) Also, you can interact with the host computer through the Type-C connector

and USB interface. There are two Type-C connectors on the WAVE ROVER driver board, and the Type-C connector in the middle is used to communicate with the ESP32 and upload the demo, no matter which way you use to connect to the host computer, the communication baud rate is 1000000.

How to Use JSON Commands

Open the Web terminal control interface, you can see the "FEEDBACK INFOMATION" under the interface. There is an input field and a "SEND" button on the top of the board. You can fill in the JSON data here and click the SEND button. If the command has feedback, the feedback information will be displayed above the input field. Below the input field, there are examples of JSON commands. There is an INPUT button behind each example. Click it, and the instruction will be filled in the input field. You can edit the instruction in the input field, and then press the SEND button to interact with the robot.

Of course, you can also generate this JSON command through the host computer and send it to the robot through the serial port, the baud rate is 1000000, the logic level of the RX/TX pin of the serial port is 3.3V, and the USB cable of the Type-C connector can also be used for communication.

Each JSON command will be explained later. First of all, let's take the JSON command to control the movement as an example, the JSON command to control the movement is SPEED_INPUT followed by {"T":1, "L":0.5, "R":0.5}, click on INPUT after the command, and the command will be filled in the input field above, and the parameter of the most important parameter of each JSON command is "T", which represents the type of the command. The parameter in front of each JSON command is "T", which represents the type of this command, for example, "T":1 means this command is a command to control the movement of the robot, the movement command, in addition to the T parameter, there are also two other parameters, L and R, which control the power of the left side motors and the power of the right side motors, respectively, and you can take the value from -1 to +1 (the higher the value is, the higher the power of the motors is, the faster the speed is, a positive value means forward, duplicate means backward), but positive value means forward and duplicate means backward. The higher the value, the higher the motor power, and the faster the speed, a positive value means backward), but due to the low-speed characteristics of DC geared motors, the absolute value of the power value needs to be greater than or equal to 0.2. When the absolute value of the speed value is lower than 0.2 but not zero, the motor will run at 20% power, and the wheels will run forward when the speed is positive; when the speed is 0, the wheels will stop rotating; and the wheels will rotate backward when the speed is negative.

For example, if you want the robot to go backward at full speed, you can edit this command as {"T":1, "L":-1, "R":-1}, after clicking the SEND button, the robot will move backward at full speed for a period of time and then stop. Due to heartbeat detection, when the robot doesn't receive any new command for a certain period of time, it will stop running automatically to avoid danger.

The following is used to describe the meaning of each JSON directive.

EMERGENCY_STOP: {"T":0}

An emergency stop instruction is also triggered if instruction type T does not have a function corresponding to it.

SPEED_INPUT: {"T":1,"L":255,"R":255}

Movement control command, the parameter represents the left side motor power and right side motor power respectively, the speed value can be positive or negative, for ± 255 for a full load operation, for 0 to stop.

PID_SET: {"T":2,"P":170,"I":90}

PID control for closed-loop motor control, note that this function is not available for chassis without speed feedback like WAVE ROVER.

OLED_SET: {"T":3,"lineNum":0,"Text":"putYourTextHere"}

OLED screen display content settings, lineNum parameter for the line settings, can be: 0, 1, 2, 3, and a total of 4 lines of content can be displayed. Each time you set a line of content, the new content will not affect the other lines of the content displayed but will replace the original content before this line.

The text parameter is the content setting, you can input text here, and the text will be displayed on the corresponding line.

After using this command, the OLED screen will no longer display the robot's information but will display what the command tells it to display.

OLED_DEFAULT: {"T":-3}

A command type of -3 resets the OLED screen to its initial state, displaying information about the robot.

PWM_SERVO_CTRL: {"T":40,"pos":90,"spd":30}

PWM servo angle control instructions, PWM signals generated by the driver board's IO4 pin, most of the PWM signals servo rotation range of 0-180 °, the pos parameter for the angle to be rotated to, 90 ° for the servo's middle position, spd parameter for the speed of the reserved interface, the demo in order to avoid clogging the threads, so the spd parameter is not used.

PWM_SERVO_MID: {"T":-4}

The PWM servo is turned to the center position, which is the 90° position.

BUS_SERVO_CTRL: {"T":50, "id":1, "pos":2047, "spd":500, "acc":30}

Regarding the serial bus servo control commands, here you should note that the connected serial bus servo operating voltage and the driver board power supply voltage are consistent, you can directly use our ST3215 serial bus servo.

id: ID of the bus servo, the ID of the serial bus servo connected to the driver board should not be duplicated.

pos: The target position to be rotated to by the servo, for ST3215 servo, in angle control mode, this value can be 0-4095, which corresponds to the clockwise rotation range of 0-360°; in continuous rotation mode, the value can be ±32766.

spd: rotation speed of the bus servo, the number of steps per unit of time (per second), 50 steps/second = 0.732 RPM (revolutions per minute), the larger the value, the faster the speed (but the speed of the servo has a limit), when the parameter is 0, it will run at the maximum speed.

acc: acceleration of bus servo rotation, the smaller the value, the smoother the start-stop, the value can be 0-254, such as set to 10, then according to the 1000 steps per second square acceleration speed, when the parameter is 0, then according to the maximum acceleration operation.

BUS_SERVO_MID: {"T":-5, "id":1}

Rotates the serial bus servo of a certain ID to the servo middle position (provided that this servo

is operating in 0: position servo mode).
id: ID of the target servo.
BUS_SERVO_SCAN: {"T":52,"num":20}
Used to scan which servos are connected to the bus, no duplicate IDs are allowed, and the result will be returned and displayed at the top of the input field.
num: Maximum ID of the servo, the larger the value, the longer the scanning time.
BUS_SERVO_INFO: {"T":53,"id":1}
It is used to get the information feedback of a particular servo, which contains the position, speed, voltage, torque, and other information of the servo.
BUS_SERVO_ID_SET: {"T":54,"old":1,"new":2}
Used to change the ID of a servo, the ID of each new servo is 1 by default, if you don't know the ID of the servo at hand, you can use the above BUS_SERVO_SCAN command to get it, but the premise is that when checking the ID of this servo, the driver board is connected to only this one servo, otherwise you don't know which ID corresponds to which servo.
old: ID of the servo whose ID is to be changed.
new: New ID to be set
BUS_SERVO_TORQUE_LOCK: {"T":55,"id":1,"status":1}
Servo torque lock switch.
id: ID of the servo to control the torque lock.
status: parameter of torque lock switch, 1 is to enable torque lock, the servo will keep its position; 0 is to disable torque lock, the servo will rotate under external force.
BUS_SERVO_TORQUE_LIMIT: {"T":56,"id":1,"limit":500}
Servo torque limit command.
id: ID of the target servo.
limit: Torque limiting ratio, 500 is 50%* locked rotor torque, 1000 is 100%* locked rotor torque, after limiting the torque, when the servo is subjected to external force when the torque of the external force is greater than the limiting torque, the servo will rotate with the external force, but it will still provide this torque, this function can be used to design the clamp of the robotic arm.
BUS_SERVO_MODE: {"T":57,"id":1,"mode":0}
Servo operation mode.
id: ID of target servo.
mode: operation mode value, 0: position servo mode, used to control the absolute angle of the servo, you can use the BUS_SERVO_CTRL command to control the angle of the servo; 3: stepper servo mode, also use the BUS_SERVO_CTRL command to control the servo, the difference is that in the stepper servo mode, the pos parameter of BUS_SERVO_CTRL is negative in the range of ±32766, and it will rotate positively for one revolution if it is given to 4096, and reverse for one revolution if it is given to -4096.
WIFI_SCAN: {"T":60}
WIFI scanning command, which disconnects the existing WIFI connection to scan for surrounding WIFI hotspots.
WIFI_TRY_STA: {"T":61}
WIFI connection command, STA mode, for connecting to a known WIFI.
WIFI_AP_DEFAULT: {"T":62}
Enabling the WIFI hotspot command, AP mode, the robot will automatically create a WIFI

hotspot.
Hotspot name: WAVE_ROVER
Hotspot Password: 12345678
WIFI_INFO: {"T":65}
For obtaining WIFI information:
WIFI_OFF: {"T":66}
Disable the WiFi function.
INA219_INFO: {"T":70}
Get information about the INA219, including the voltage and current power of the power supply.
IMU_INFO: {"T":71}
Used to obtain IMU information, including heading angle, geomagnetic field, acceleration, attitude, temperature, etc.
ENCODER_INFO: {"T":73}
Used to get motor encoder information. (Not applicable to WAVE ROVER as there are no encoders).
DEVICE_INFO: {"T":74}
Used to get the device information, the device information is required to be customized by the user, used to introduce the use of this device or other information.
IO_IR_CUT: {"T":80,"status":1}
It is used to control the high and low levels of the IO5 pin on top of the driver board, which can be used to control the night vision function switch of the infrared camera or to control the relay.
SET_SPD_RATE: {"T":901,"L":1.0,"R":1.0}
Used to adjust the power of each side motor. When giving the robot the same power for the left and right motors, if the robot's movement is not straight, you can use this command to fine-tune the power of the motors on both sides, and the value will be used as a coefficient multiplied by the motor's output power to change the motor's power.
L: Power coefficient of left side motor.
R: The power coefficient of the right side motor.
GET_SPD_RATE: {"T":902}
Get the power coefficients of the left and right side motors.
SPD_RATE_SAVE: {"T":903}
Saving the power coefficient of the motor will be saved in the nvs area of the ESP32 and will not be lost after power down, and this speed coefficient will be read and loaded by the nvs area after power up.
GET_NVS_SPACE: {"T":904}
Get the remaining space in the nvs area of ESP32.
NVS_CLEAR: {"T":905}
Clear nvs zone, this command deletes the entire contents of the nvs zone, and the speed coefficient changes back to the default value of 1.0.

Driver Board General Driver for Robots Module Usage Tutorial

How To Install Arduino IDE
Tutorial I: Motor With Encoder Control Demo
Tutorial II: Motor Without Encoder Control Demo

Tutorial III: ST3215 Serial Bus Servo Control Demo
Tutorial IV: PWM Servo Control Demo
Tutorial V: IMU Data Reading Demo
Tutorial VI: SD Card Reading Demo
Tutorial VII: INA219 Voltage And Current Monitoring Demo
Tutorial VIII: OLED Screen Control Demo

Motor Specifications

Model: GF12-N20 Motor 12V200rpm Gearbox
Rated voltage: 12V
Rated current: 0.055A
Locked rotor current: 0.45A
Rated torque: 0.09kg.cm
Locked rotor torque: 0.7kg.cm
Rated output power: 1.5W
No-load speed: 66±10%RPM
Motor size: 34*12mm
Output shaft size: 4*10mm

Resource

Open-source Demo

WAVE ROVER Open-source Demo

Robot Dimensions & Mounting Plate Drawing

Chassis DXF Drawing
Chassis PDF Drawing
Mounting Plate DXF Drawing
Mounting Plate PDF Drawing

Robot Model

WAVE ROVER 3D Model