MIROBOT

Autobot User Manual

Prepared by: Wayne Liu
9 April 2025
Version #: 20250409

TABLE OF CONTENTS

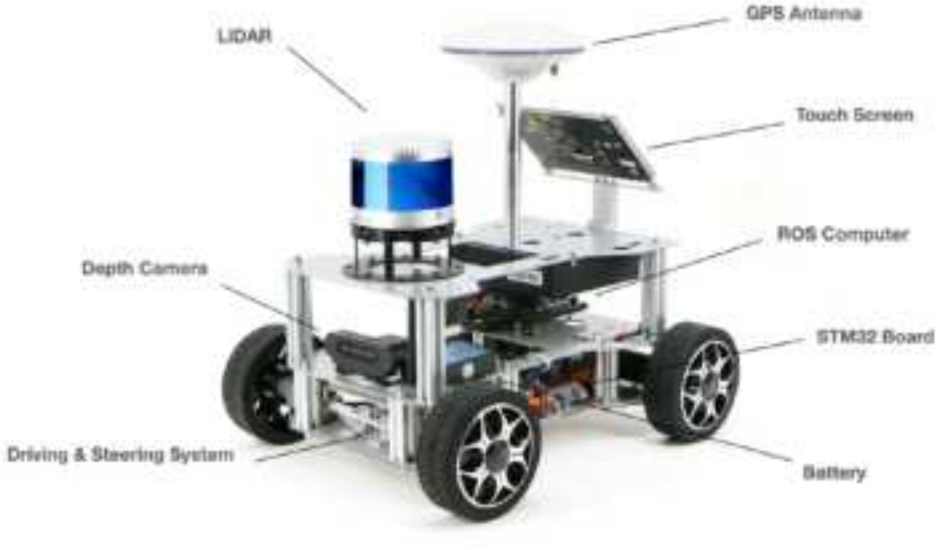
Summary

Powered by Autoware - the world's leading open-source autonomous driving software built on ROS (Robot Operating System) - Autobot is an all-in-one platform for autonomous driving projects. Tailored for ROS engineers, educators, and students, Autobot integrates Autoware's full capabilities with support for ROS 2 Humble. It also includes MiROS, a visual programming tool for ROS that enables intuitive, browser-based development across multiple operating systems.

Autobot comes with 4 models:

Autobot - Suitable for ROS beginners and low budget projects.
Autobot AC - Autobot bundled with Auto Charging station and software.
Autobot Plus - This is the 4WD version of Autobot with Independent Suspension Systems. This category is serious enough to be considered for industrial and commercial development.
Autobot Plus AC - Autobot Plus bundled with Auto Charging station and software.

1. Models and Key Component



⍰⍰⍰⍰⍰⍰	⍰⍰⍰⍰⍰
⍰⍰⍰⍰⍰⍰⍰	
⍰⍰⍰⍰⍰⍰⍰⍰ ⍰⍰⍰⍰	

2. Product Specifications

Product Matrix				
Product Name	Autobot	Autobot AC	Autobot Plus	Autobot Plus AC
Computer	Jetson Orin NX 16GB			
Motor Reduction Ratio	1:27		1:47	
Max Speed	1.3m/s		0.89m/s	
Weight	11.5 kg		42 kg	
Max Payload	10 kg		40 kg	
Size	445*358*443mm		810*670*600mm	
Minimal Turning Radius	0.77m		1.78m	
Battery Life	About 6 hours		About 9 hours	
Power Supply	24v 6000 mAh $LiFePO_4$ Battery + 25.55V 3A charger		24v 20000 mAh $LiFePO_4$ Battery + 25.55V 3A charger	
Steering Gear	S20F 20kg torque digital servo		DS5160 60kg torque digital servo	
Wheels	125mm diameters solid rubber wheels		254mm diameters inflatable rubber wheels	
Encoder	500 line AB phase high precision encoder			
Suspension System	Coaxial Pendulum Suspension System		4W Independent Suspension System	
Touch Screen	7" Touch Screen		14" Touch Screen	
Control Interface	iOS & Android App via Bluetooth or Wifi, PS2, CAN, Serial Port, USB			

3. GPS Communication System

RTK stands for Real-Time Kinematic, and it's a technique used in GPS/GNSS communication to provide high-precision positioning, typically down to the centimeter level — much more accurate than standard GPS.

What does RTK do?

RTK improves positioning accuracy by using corrections from a nearby base station to reduce GPS signal errors like:

- Satellite clock drift
- Atmospheric interference
- Multipath effects

How RTK works:

1. Base station: A GPS receiver placed at a known, fixed location.
2. Rover: A mobile GPS receiver (like one on a robot, drone, or vehicle).
3. Communication link: The base sends real-time correction data (via radio, 4G, etc.) to the rover.
4. Correction: The rover applies the correction to its own GPS signal to calculate a much more precise location.

Accuracy:

- Standard GPS: ~3–10 meters
- Autobot RTK GPS: ~1.5 meters

RTK is commonly used in:

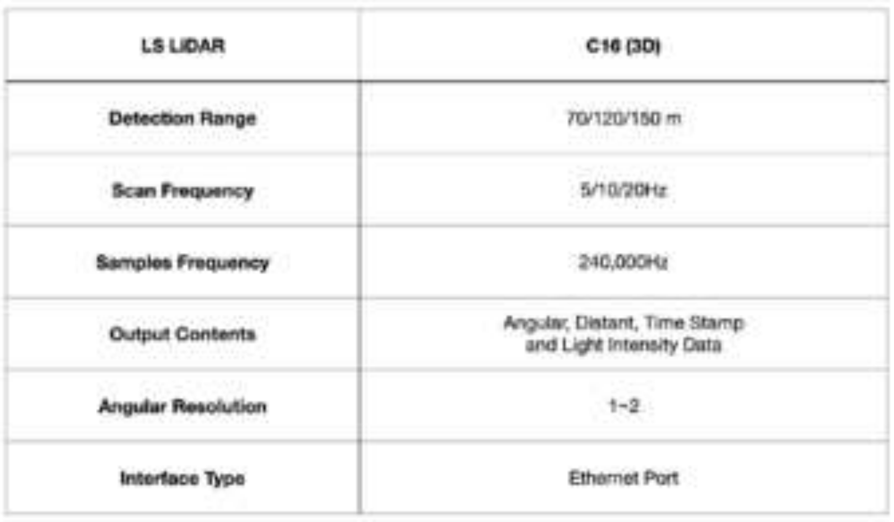
- Autonomous vehicles
- Drones and UAV mapping
- Precision agriculture
- Surveying and construction
- Robotics (especially outdoor)

Differential Positioning RTK Module:

Equipped with a GNSS module, Autobot can connect to a base station via a 4G network to access RTK differential signals and obtain high-precision positioning data.

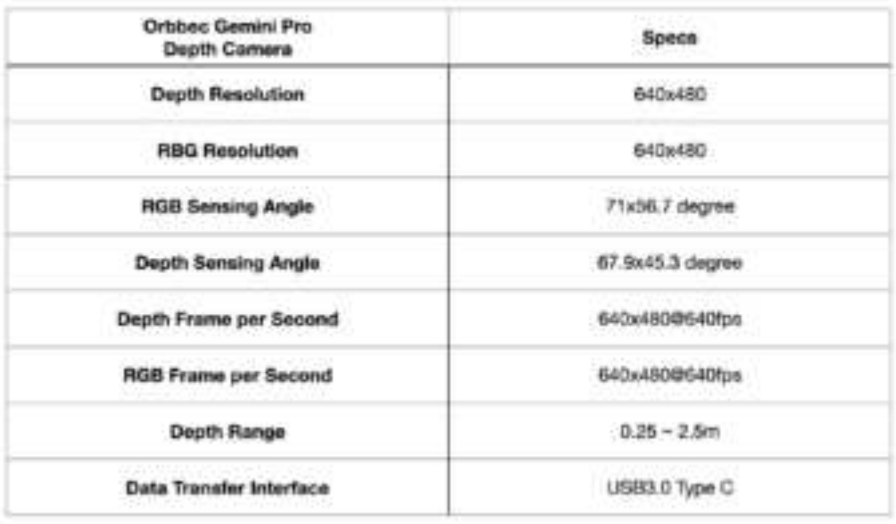
4. Sensing System: LiDAR & Depth Camera

A LS C16 LiDAR is installed on all Autobot models. These 3D LiDAR’s offer a 360 degree scanning range and surroundings perception and boast a compact and light design. They have a high Signal Noise Ratio and excellent detection performance on high/low reflectivity objects and perform well in strong light conditions. They have a max detection range of 150 metres and a max scan frequency of 20Hz. This LiDAR integrates seamlessly into the Autobots, ensuring all mapping and navigational uses can be easily achieved in your project. The below table summaries the technical specifications of the C16 LiDAR:



LS LiDAR	**C16 (3D)**
Detection Range	70/120/150 m
Scan Frequency	5/10/20Hz
Samples Frequency	240,000Hz
Output Contents	Angular, Distant, Time Stamp and Light Intensity Data
Angular Resolution	1~2
Interface Type	Ethernet Port

Additionally, all Autobots are equipped with an Orbbec Gemini Pro Depth Camera, which is an RGBD camera. This camera is optimized for a rage of uses including gesture control, skeleton tracking, 3D scanning and point cloud development. The following table summarizes the technical features of the depth camera.

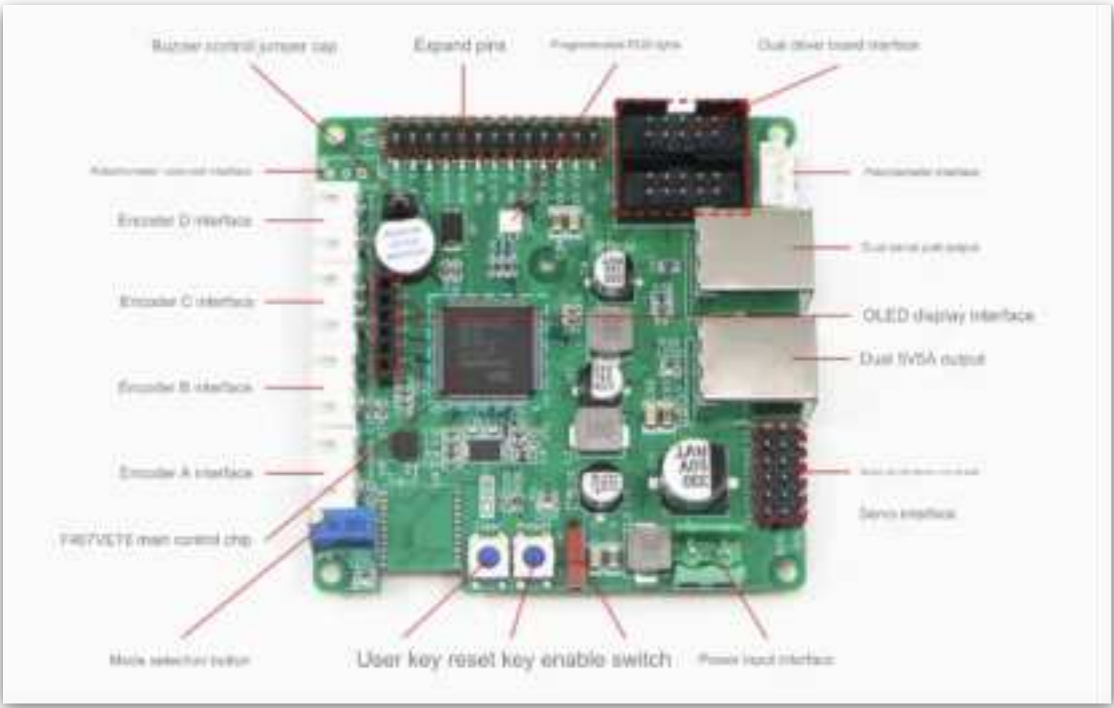


Orbbec Gemini Pro Depth Camera	**Specs**
Depth Resolution	640x480
RBG Resolution	640x480
RGB Sensing Angle	71x56.7 degree
Depth Sensing Angle	67.9x45.3 degree
Depth Frame per Second	640x480@640fps
RGB Frame per Second	640x480@640fps
Depth Range	0.25 ~ 2.5m
Data Transfer Interface	USB3.0 Type C

5. STM32 Board (Motor Control & IMU)

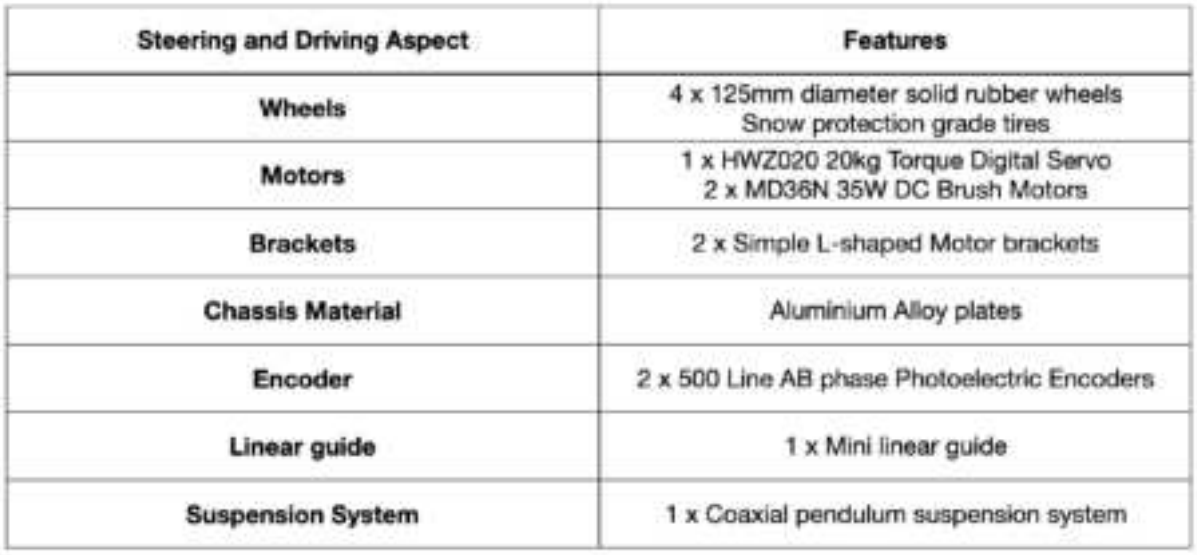
The STM32F103RC Board is the micro-controller used in all Autobots. It has a high performance ARM Cortex -M3 32-bit RISC core operating at a 72MHz frequency along with high-speed embedded memories. It operates in -40°C to +105°C temperature range, suiting all robotic applications in worldwide climates. There are power-saving modes which allow the design of low-power applications. Some of the applications of this microcontroller include: motor drives, application control, robotic application, medical and handheld equipment, PC and gaming peripherals, GPS platforms, industrial applications, alarm system video intercom and scanners.

STM32F103RC	Features
Core	ARM32-bit Cortex –M3 CPU Max speed of 72 MHz
Memories	512 KB of Flash memory 64kB of SRAM
Clock, Reset and Supply Management	2.0 to 3.6 V application supply and I/Os
Power	Sleep, Stop and Standby modes V_{BAT} supply for RTC and backup registers
DMA	12-channel DMA controller
Debug Mode	SWD and JTAG interfaces Cortex-M3 Embedded Trace Macrocell
I/O ports	51 I/O ports (mappable on 16 external interrupt vectors and 5V tolerant)
Timers	4x16-bit timers 2 x 16-bit motor control PWM timers (with emergency stop) 2 x watchdog timers (independent and Window) SysTick timer (24-bit downcounter) 2 x 16-bit basic timers to drive the DAC
Communication Interface	USB 2.0 full speed interface SDIO interface CAN interface (2.0B Active)



6. Steering & Driving System

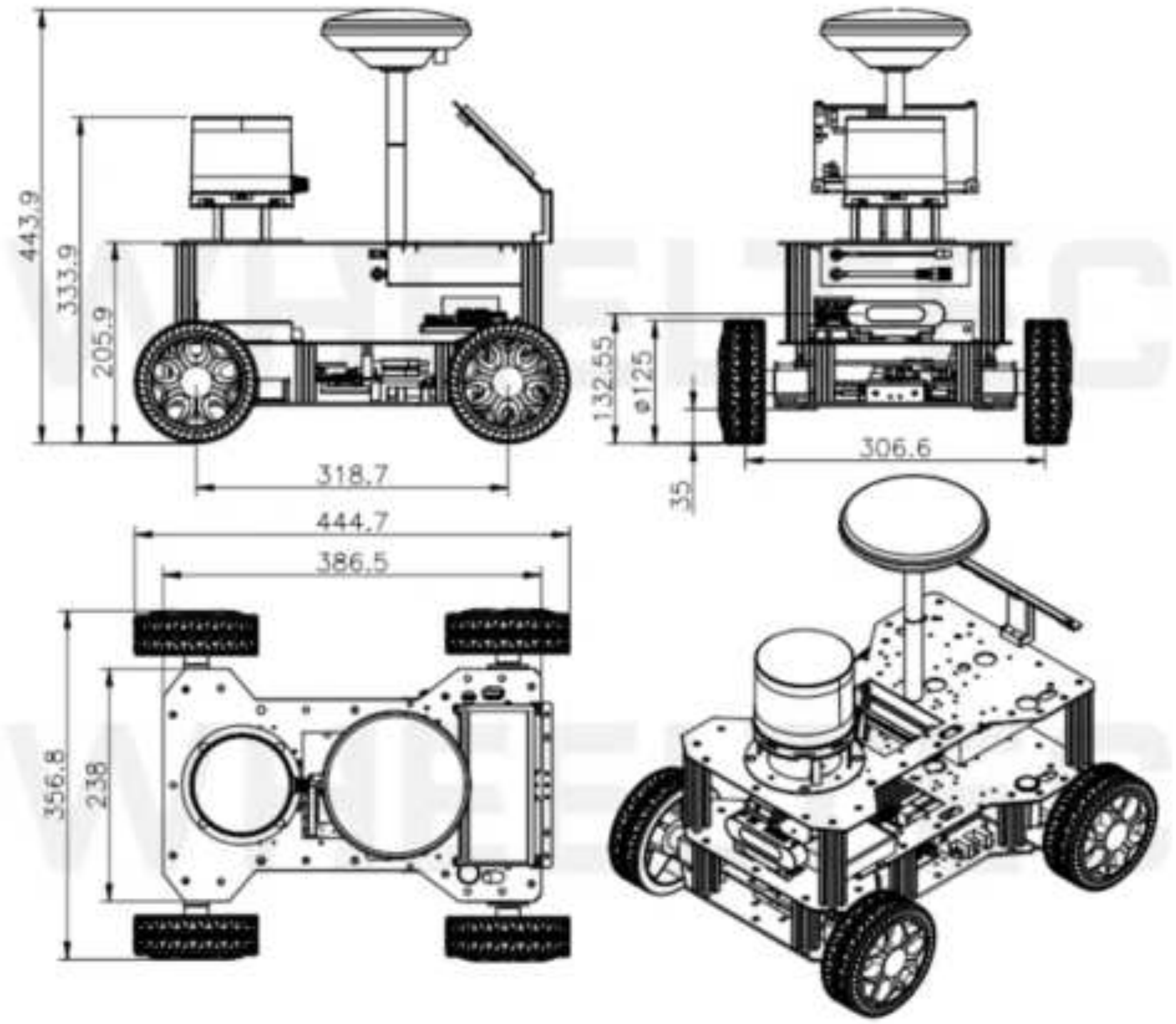
The Steering and Driving system is integrated with the design and build of the Autobot. Depending on the model purchased it will be either a 2 wheel or 4 wheel drive, with both options being suitable to a variety of research and development purposes. The wheels on all Autobots are solid rubber with snow protection grade tires. There is a coaxial pendulum suspension system, and the top range Autobots are equipped with shock absorbers with independent suspension systems, ensuring it is able to successfully navigate difficult terrain.
Steering and Driving Technical Specifications:



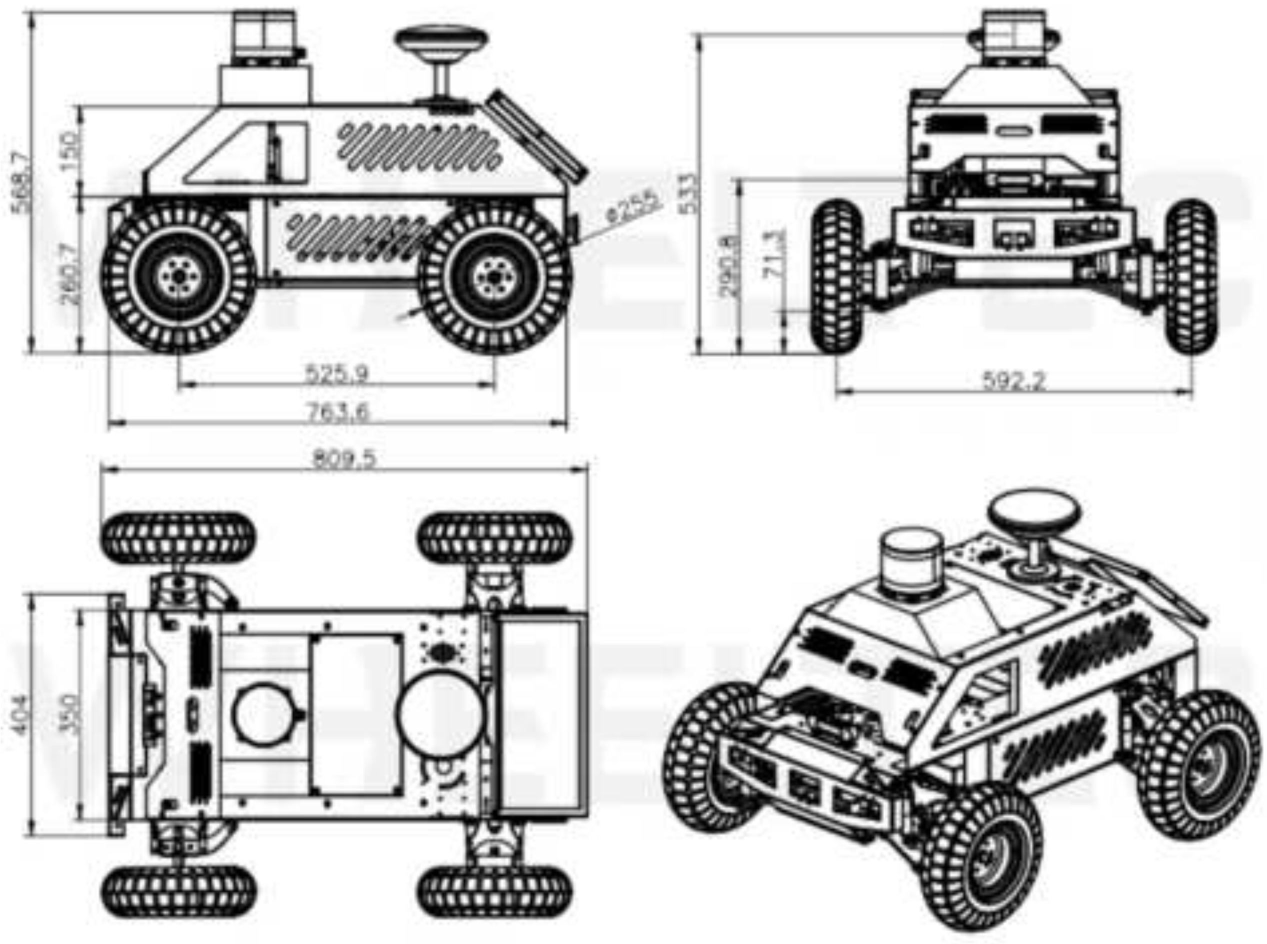
Steering and Driving Aspect	Features
Wheels	4 x 125mm diameter solid rubber wheels Snow protection grade tires
Motors	1 x HWZ020 20kg Torque Digital Servo 2 x MD36N 35W DC Brush Motors
Brackets	2 x Simple L-shaped Motor brackets
Chassis Material	Aluminium Alloy plates
Encoder	2 x 500 Line AB phase Photoelectric Encoders
Linear guide	1 x Mini linear guide
Suspension System	1 x Coaxial pendulum suspension system

Autobot Chassis Design Diagram:

Autobot



Autobot Plus



7. Power Management

LiFePO$_4$ Battery:

All Autobots come with a 6000 mAh or a 2000 mAh LiFePO$_4$ Battery and a Power Charger.

Technical Specifications:

Items	6000 mAh	20000 mAh
Autobot Model	Autobot & Autobot AC	Autobot Plus & Autobot Plus AC
Battery Pack	22.4V 6000mAh	22.4V 20000mAh
Core Material	Lithium Iron Phosphate	Lithium Iron Phosphate
Cutoff Voltage	16.5 V	16.5 V
Full Voltage	25.55 V	25.55 V
Charging Current	3A	3A
Shell Material	Metal	Metal
Discharge Performance	15A Continuous Discharge	20A Continuous Discharge
Plug	DC4017MM female connector (charging) XT60U-F female connector (discharging)	DC4017MM female connector (charging) XT60U-F female connector (discharging)

Battery Protection:

Short circuit, overcurrent, overcharge, over-discharge protection, support charging while using, built-in safety valve, flame retardant board.

Auto Charge:

Auto Charge is an Auto Charging Station bundled with Autobot AC and Autobot Plus AC models.

9. MiROS Visual Programming

MiROS is a cloud-based ROS (Robot Operating System) visual programming tool. ROS is based on Linux and requires programming skills in C/C++ or Python. MiROS enables Mac/Windows users to develop ROS programs by drag-and-drop coding without the need to install a Linux VM (Virtual Machine).

9.1 Install Docker Desktop

Dockerization is one of the fundamental design principles for MiROS. Visit the below website to download and install your respective Docker Desktop app:

https://www.docker.com/products/docker-desktop/

9.2 Install MiROS App

After installing Docker Desktop, visit the below website to download and install your respective MiROS app. Please make sure to select to correct installer according to your computer CPU architecture. The download website is here:

https://www.mirobot.ai/downloadmiros

Once you have successfully downloaded MiROS on your computer, you can locate the MiROS installer in your download folder of your computer with an icon like this:

To install MiROS, simply double click the MiROS installer. Once the installation has finished, you will find the MiROS app appears either on your Desktop or in your Application Folder.

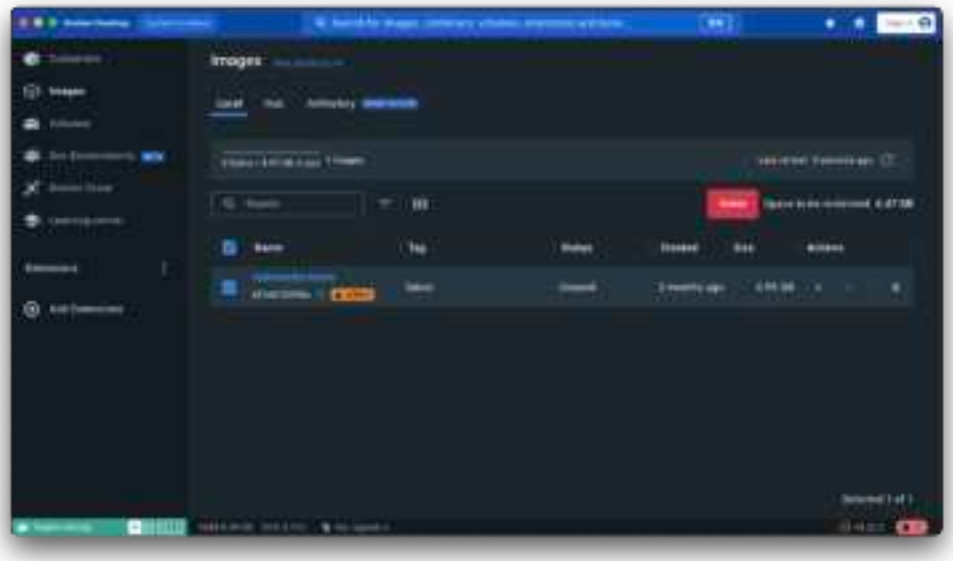
To launch MiROS, follow the below steps:

1. Launch Docker Desktop App.
2. Launch MiROS App.
3. You will see a Terminal window appears showing MiROS is pulling the ROS and its associated Ubuntu image from the Cloud to your Docker. Your computer screen could look like the picture shown below:

The above process will take about 3 ~ 5 minutes. Once this process has finished, your computer’s default web browser will launch the MiROS website.

IMPORTANT:

Every time you launch MiROS on your Mac or Windows, you should launch Docker Desktop first. If you have successfully installed MiROS, your Docker Desktop should show the below docker image in your Images section shown as below:

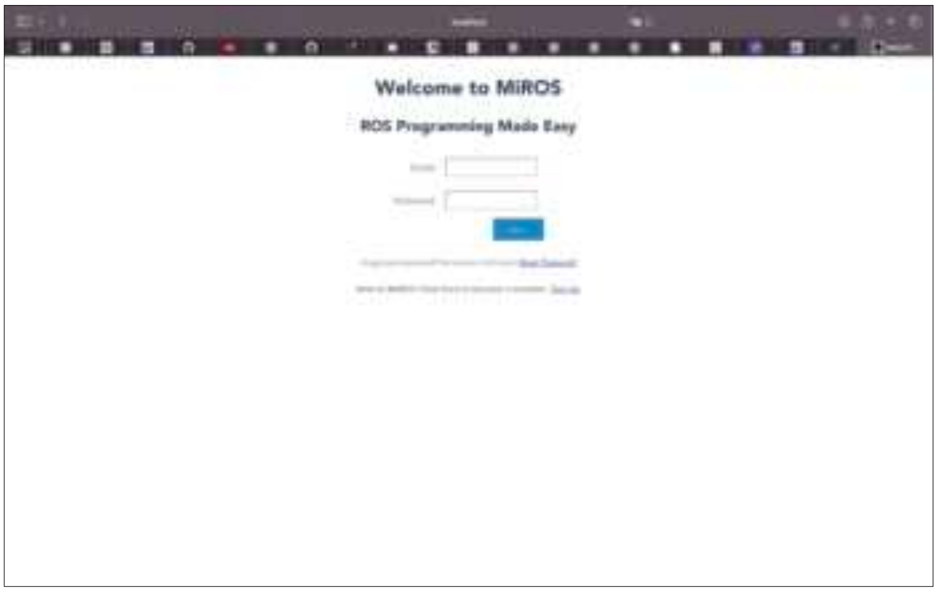


If your web browser

has launched, however, the MiROS website is not loading and the web browser is blank, you may enter the below URL to load the MiROS website:

localhost:8000

Once you see the below MiROS login page, you have successfully installed and launched MiROS.



If this is you are a first time user of MiROS, please register a user account first. Registering with MiROS will enable the following Cloud Services:

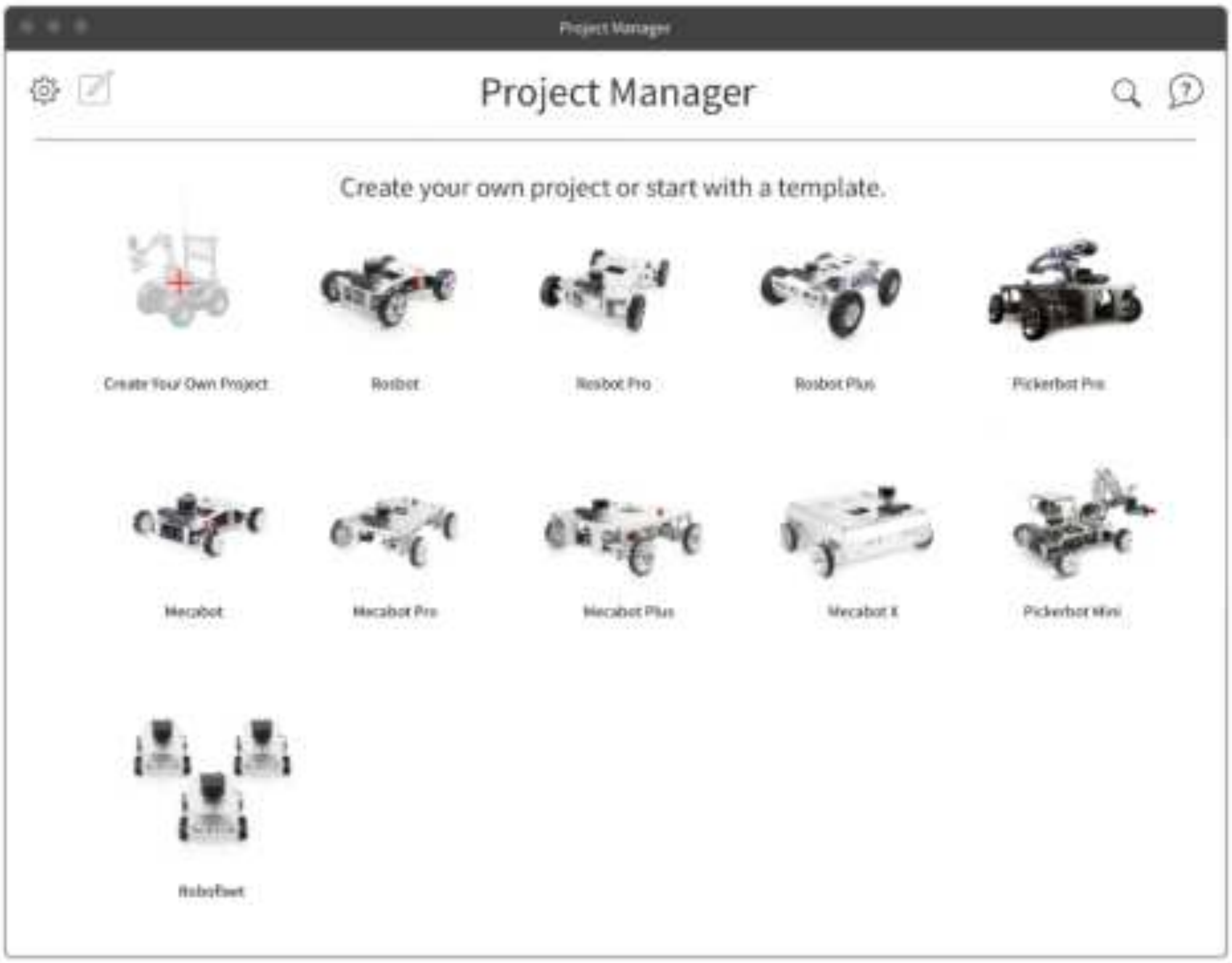
- Save and syn your projects on the MiROS Cloud.
- Access to your MiROS projects via any web browsers on any computers or robots.
- Export your ROS code to any computers or robots.
- Push your latest code on your GitHub repositories from any computers or robots.

9.3 Project Manager

Once you log in to MiROS, you will land in Project Manager.

Start with a template

If your robot model is listed in one of the templates, you can select the correct template and proceed to create a new Workspace for your project. By selecting the right template, your project will start with all the factory default ROS packages preinstalled on your robot.



IMPORTANT：

If you create a new Workspace by selecting a robot template, the ROS packages you are going to create and the factory default ROS packages are all stored and run on the MiROS Cloud and the docker container in your localhost computer, **not on your robot**.

You can connect to your robot during your project development by topic subscriptions or publications or trigger launch files on your robot remotely from MiROS on your localhost computer. The ROS software on your robot is untouched throughout your project development on MiROS until you export your own code to your robot and compile it.

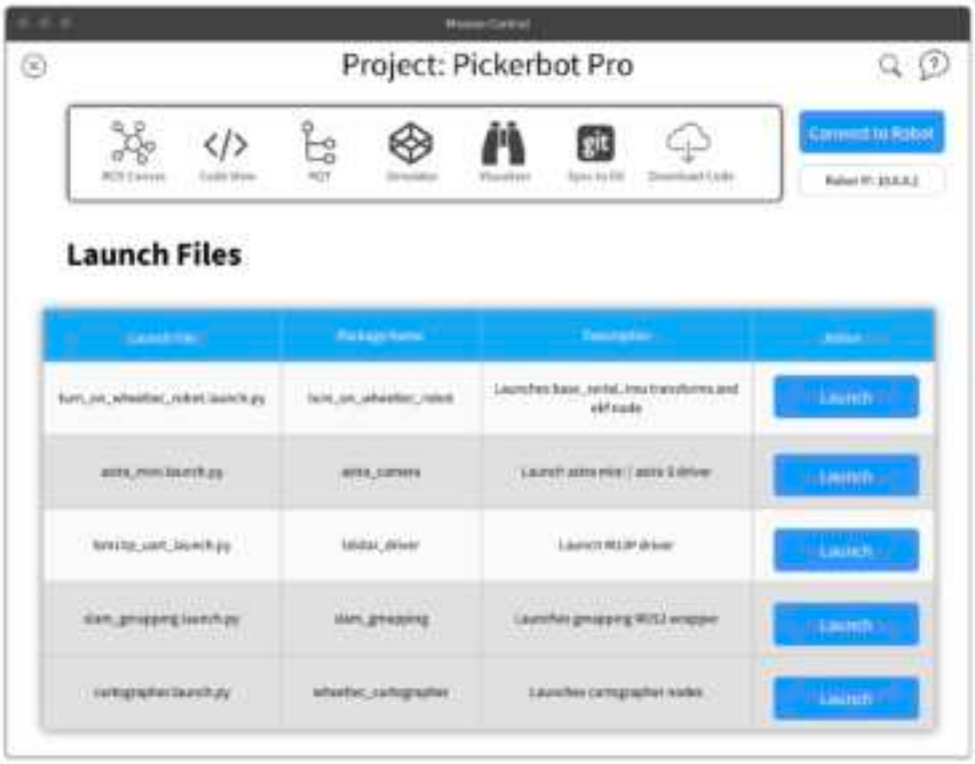
Start from scratch

If your robot is not listed as one of the templates, you will need to create your own project from scratch by clicking the red cross button.

When you are creating your project from scratch, you can still load the ROS packages from your robot to MiROS webpage. You will learn about the details in the next chapter.

8.4 Mission Control

Mission Control is your control center to monitor, communicate and command your robot either in a physical environment or in a simulated environment. The below screenshot is the Mission Control user interface:



Launch File	Package Name	Description	Action
turn_on_wheeltec_robot.launch.py	turn_on_wheeltec_robot	Launches base_serial, imu transforms and ekf node	Launch
astra_mini.launch.py	astra_camera	Launch astra mini / astra S driver	Launch
lsm10p_uart_launch.py	lslidar_driver	Launch M10P driver	Launch
slam_gmapping.launch.py	slam_gmapping	Launches gmapping ROS2 wrapper	Launch
cartographer.launch.py	wheeltec_cartographer	Launches cartographer nodes	Launch

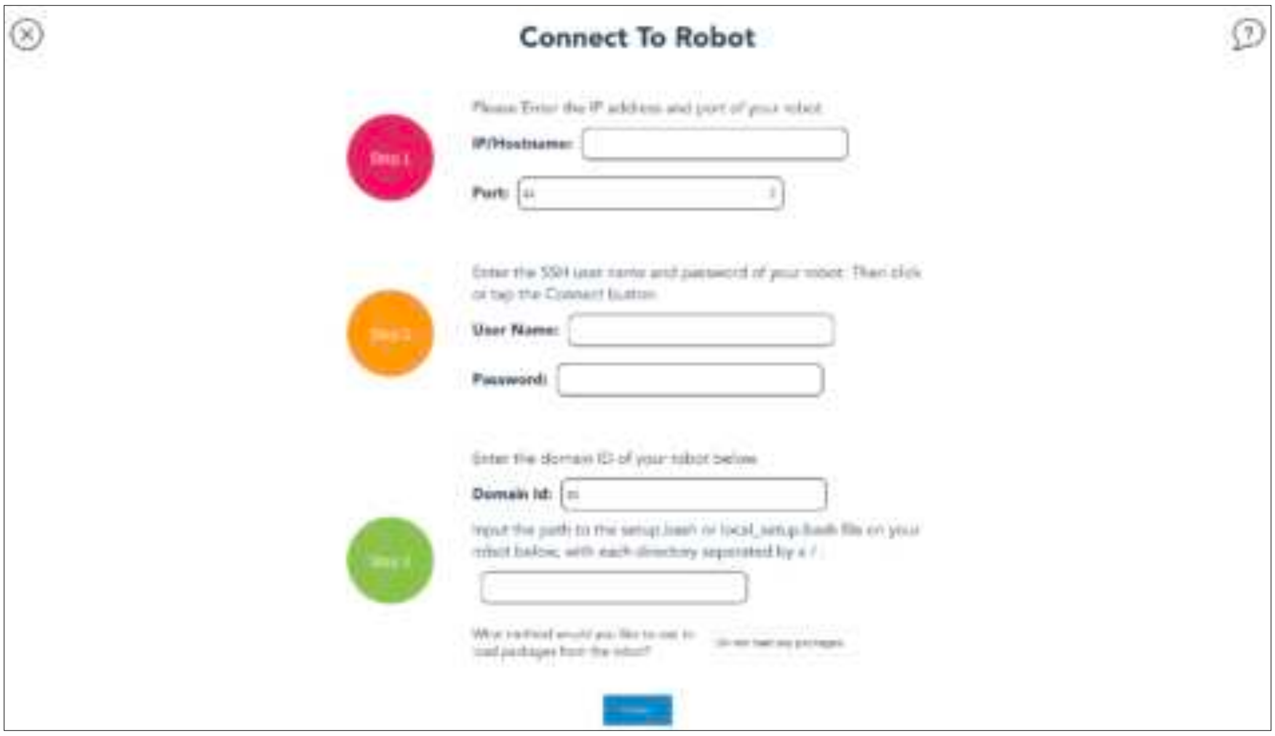
There are 3 main sections of Mission Control:

- Tool Bar - The Tool Bar contains the following function buttons:
  - ROS Canvas - access to GUI-based programming environment.
  - Code View - access the code-base programming environment.
  - RQT - access ROS RQT tool.
  - Simulator - access ROS simulators such as Gazebo and Webots.
  - Visualiser - access ROS visualisation tools such as Rviz and Foxglove.
  - Sync to Git - connect to your GitHub account and sync with your GitHub repositories.
  - Download Code - download your MiROS generated ROS code to your localhost computer.
- Connect to Robot - a button to trigger connection between MiROS web interface and your robot via local Wifi network.
- Launch Files - send launch file commands to your robot via constant ssh connection.

9.5 Connect to Robot

MiROS connects to your robot via constant ssh connections. There are three requirements in order to maintain the constant ssh connection between the MiROS website and your robot:

- Autobot IP: **192.168.0.100**
- SSH User Credentials:
    - User Name: wheeltec
    - Password: dongguan
- Enter the path of the setup.bash file:
  /home/wheeltec/wheeltec_ros2/install/setup.bash



Connect To Robot

Please Enter the IP address and port of your robot

IP/Hostname:

Port: 22

Enter the SSH user name and password of your robot. Then click or tap the Connect button

User Name:

Password:

Enter the domain ID of your robot below

Domain Id: [illegible]

Input the path to the setup.bash or local_setup.bash file on your robot below, with each directory separated by a /

What method would you like to use to load packages from the robot? [illegible]

After connection is established between MiROS running on your localhost computer and your robot, you can carry out the following actions:

- You can send launch commands from your Launch File table in MiROS to your robot.
- You can retrieve all of the ROS packages and active messages from your robot to MiROS.
- You can test your code and how your robot functions in real-time.

To connect to your robot, follow the following steps:

1. Click on “Connect to Robot” button on the top right corner of the Mission Control interface.

2. You will see the following screenshot to enter your robot's IP, domain ID and the ssh login information.

IMPORTANT：

1. You should enter the setup.bash or local_setup.bash file on your robot.

2. If your project is based on an existing robot template, you don't need to load all the ROS packages from your robot to MiROS anymore. You should keep the "Do not load any packages" option just above the blue "Connect" button. If you start your project from scratch, you may change the option to "Load all packages from robot".

After you have successfully connected to your robot, you will see the following items added to your MiROS project:

- Your robot's IP is displayed on the top right corner of your Mission Control.
- Your Launch File table should be filled with the launch files copied from your robot.
- Enter into ROS Canvas, you will see all of your robot's ROS packages are displayed and labelled in red.

9.6 Launch Files

A Launch File in ROS is an XML file used to automate the process of starting multiple nodes and setting up their configurations. These files make it easier to manage complex robotic systems by launching multiple nodes, setting parameters, and defining how nodes interact with each other, all in a single command.

Here are the key functions of a ROS launch file:

1. Launch Multiple Nodes: Instead of manually starting each node, a launch file can start several nodes simultaneously.

2. Set Parameters: You can define and set global or node-specific parameters for the ROS system.

3. Remap Topics: Launch files allow remapping of topic names so nodes can communicate even if they are expecting different topic names.

4. Namespace Assignment: It can define namespaces to organize the nodes and topics in a structured way.

5. Include Other Launch Files: Complex systems can be modularized by including other launch files.

A basic example of a launch file (`example.launch`) looks like this:

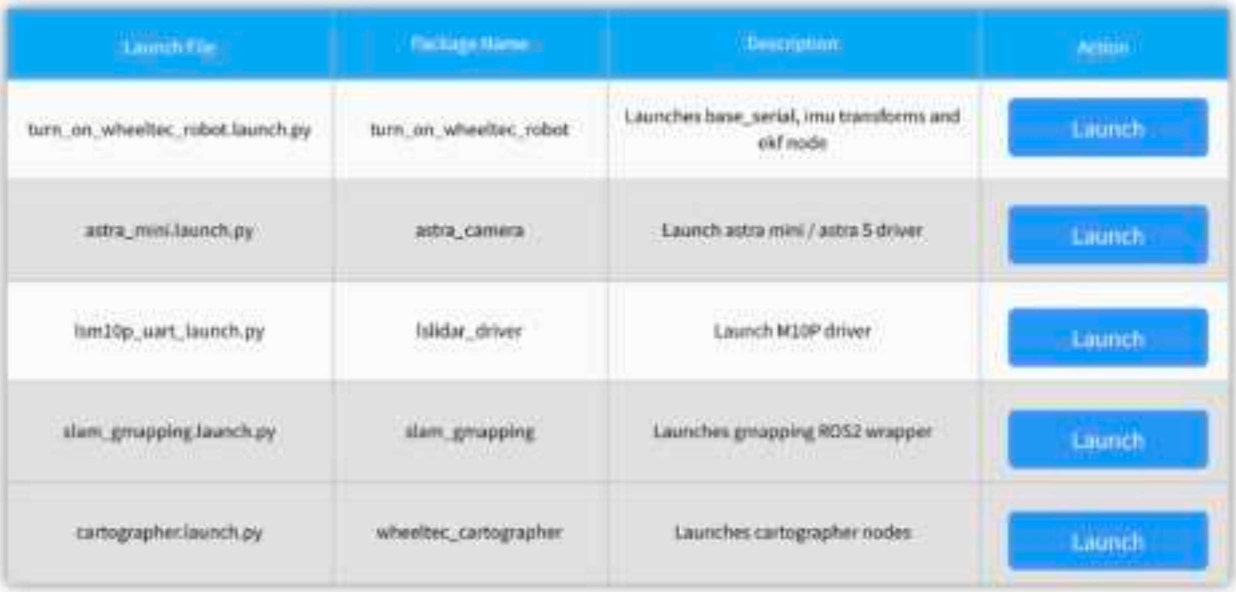
```xml
<launch>
    <!-- Launch node1 -->
    <node name="node1" pkg="package_name" type="node_executable" output="screen">
        <param name="param_name" value="param_value"/>
    </node>

    <!-- Launch node2 with remapped topic -->
    <node name="node2" pkg="package_name" type="node_executable">
        <remap from="/old_topic" to="/new_topic"/>
    </node>
</launch>
```

This launch file starts two nodes (`node1` and `node2`), sets parameters, and remaps a topic for `node2`. You can run it using the following command in ROS 2:

```
roslaunch package_name example.launch
```

Using launch files simplifies the management of large and complex robot systems in ROS.

In Mission Control, the Launch Files are presented in a table view shown as the below screenshot:



Launch File	Package Name	Description	Action
turn_on_wheeltec_robot.launch.py	turn_on_wheeltec_robot	Launches base_serial, imu transforms and ekf node	Launch
astra_mini.launch.py	astra_camera	Launch astra mini / astra 5 driver	Launch
lsm10p_uart_launch.py	lslidar_driver	Launch M10P driver	Launch
slam_gmapping.launch.py	slam_gmapping	Launches gmapping ROS2 wrapper	Launch
cartographer.launch.py	wheeltec_cartographer	Launches cartographer nodes	Launch

The Launch File table contains the Launch File Name, Package Name where the file belongs to, a brief description and a “Launch” button to quickly send launch command to your robot.

IMPORTANT：

In order to send launch command from your MiROS project to your robot and maintain a constant ssh connection, the below requirements should be met:

- Your localhost computer running MiROS and your robot should be connected to the same local Wifi network.
- You should know the ssh login information of your robot including its IP.
- Your robot has installed MiROS Linux version. Without MiROS installed on your robot, you still can connect to your robot from MiROS. However, the ssh connection is not constant.

10. ROS 2 Quick Start

For Linux users who prefer command lines instead of visual programming, you can follow the below instruction to start up Autobot in ROS 2.

When the robot is first powered on, it is controlled by ROS by default. Meaning, the STM32 chassis controller board accepts commands from the ROS 2 Controller such as Jetson Orin.

Initial setup is quick and easy, from your host PC (Ubuntu Linux recommended) connect to the robot's Wi-Fi hotspot. Password by default is "**dongguan**".

Next, connect to robot using SSH via the Linux terminal, IP address is 192.168.0.100, default password is **dongguan**.

~$ ssh wheeltec@192.168.0.100

With terminal access to the robot, you can navigate to the ROS 2 workspace folder, under "wheeltec_ROS 2"

Prior to running test programs, navigate to wheeltec_ROS 2/turn_on_wheeltec_robot/ and locate wheeltec_udev.sh - This script must be run, typically only once to ensure proper configuration of peripherals.

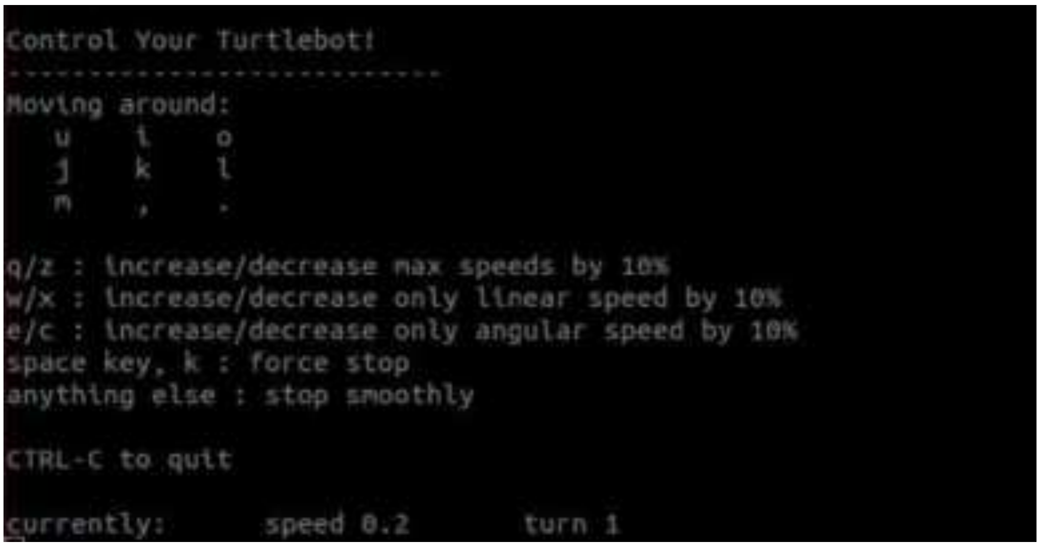
You are now able to test the robot's functionality, to launch the ROS 2 controller functionality, run:

"roslaunch turn_on_wheeltec_robot turn_on_wheeltec_robot.launch"

~$ ros2 launch turn_on_wheeltec_robot turn_on_wheeltec_robot.launch

In a second terminal, you can use the keyboard_teleop node to validate chassis control, this is a modified version of the popular ROS 2 Turtlebot example. Type (more tele-op control is available in section 8):

"ros2 run wheeltec_robot_keyboard wheeltec_keyboard"



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

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

CTRL-C to quit

currently:      speed 0.2       turn 1
```

11. Pre-installed ROS 2 Humble Packages

Below are the following user-oriented packages, whilst other packages may be present, these are dependencies only.

turn_on_wheeltec_robot

This package is crucial for enabling robot functionality and communication with the chassis controller. The primary script “turn_on_wheeltec_robot.launch” must be used upon each boot to configure ROS 2 and controller.

wheeltec_rviz2

Contains launch files to launch rviz with custom configuration for Pickerbot Pro.

wheeltec_robot_slam

SLAM Mapping and localisation package with custom configuration for Pickerbot Pro.

wheeltec_robot_rrt2

Rapidly exploring random tree algorithm - This package enables Pickerbot Pro to plan a path to it’s desired location, by launching exploration nodes.

wheeltec_robot_keyboard

Convenient package for validating robot functionality and controlling using the keyboard, including from remote host PC.

wheeltec_robot_nav2

ROS 2 Navigation 2 node package.

wheeltec_lidar_ros2

ROS 2 Lidar package for configuring Leishen M10/N10.

wheeltec_joy

Joystick control package, contains launch files for Joystick nodes.

simple_follower_ros2

Basic object and line following algorithms using either laser scan or depth camera.

ros2_astra_camera

Astra depth camera package with drivers and launch files.

12. Autoware Software Stack

Autoware in ROS 2 is an open-source autonomous driving software stack built on top of the Robot Operating System 2 (ROS 2) framework. It provides everything needed to build, simulate, and deploy self-driving vehicle applications.

What is Autoware?

- A modular, full-stack software platform for autonomous vehicles.
- Developed and maintained by the Autoware Foundation.
- Originally built on ROS 1 (as Autoware.AI), but now fully transitioned to ROS 2 (as Autoware or formerly Autoware.Auto).

Core Components of Autoware:

- Perception: Lidar-based 3D object detection, classification, tracking
- Localization: Using sensor fusion (e.g., GPS + Lidar + IMU)
- Planning: Route planning, behavior planning, trajectory generation
- Control: Sending control commands to the vehicle (steering, throttle, brake)
- Simulation: Integration with simulators like LGSVL, CARLA, and Gazebo
- Mapping: HD map usage and generation

Use Cases:

- Robotaxis
- Autonomous delivery vehicles
- Research in self-driving technology
- Education and prototyping of mobility systems

Repositories & Tools:

- GitHub: https://github.com/autowarefoundation

- Autoware Universe: The ROS 2-based version, now the mainline development branch
- Simulation tools: Integration with LGSVL, CARLA, and Gazebo for testing
- Autoware Launcher: A GUI-based system to manage modules and configurations