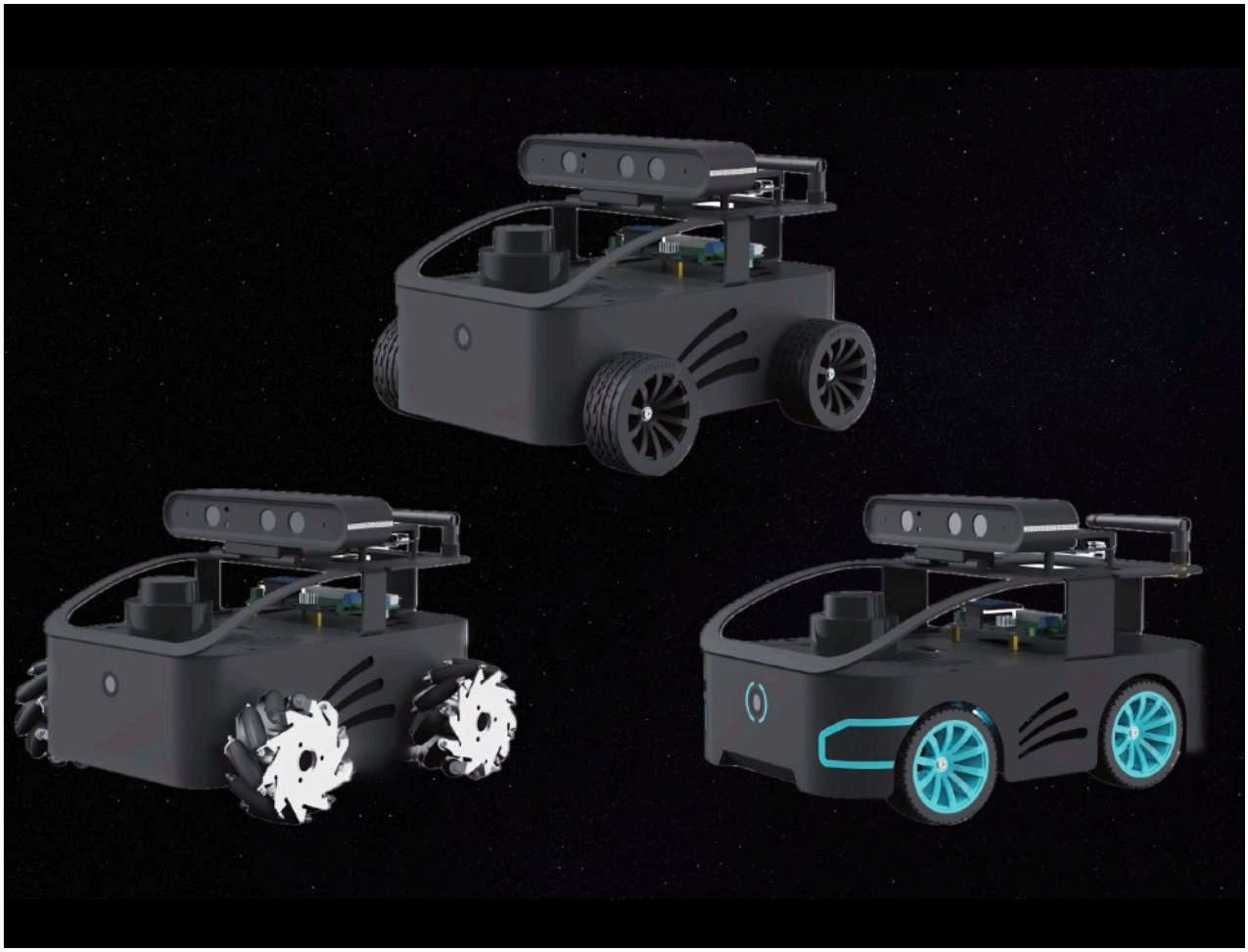


Xbot User Manual

Author: Wayne Liu

26 February 2025

TABLE OF CONTENTS

Summary

Xbot is an educational and research robot based on ROS (Robot Operating System) for robotic researchers, educators, students and developers.

Xbot is ideal for ROS beginners with affordable price, compact design and ready-to-go package. Xbot is also a solid Autonomous Mobile Robot (AMR) platform for robotic education and research projects.

Xbot is equipped with builtin ROS Controller, LiDAR, Depth Camera, STM32 Motor/Power/IMU Controller and metal chassis with three different driving systems.

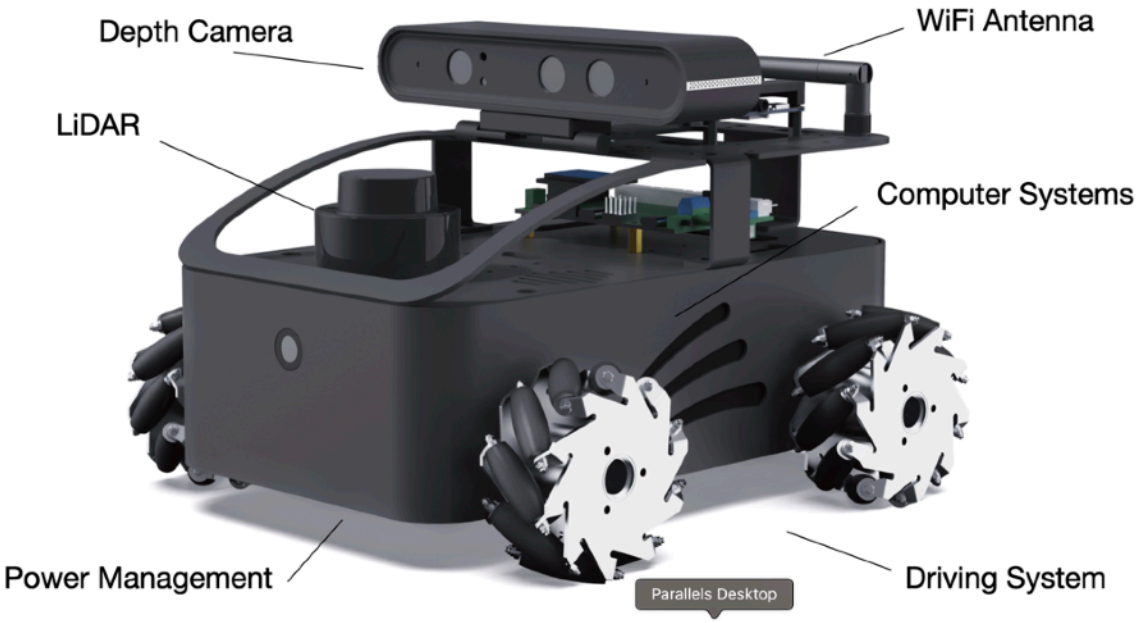
Based on the different driving systems, Xbot comes with three models:

Xbot Model A - Ackerman Drive
Xbot Model M - Mecanum Drive
Xbot 4WD - 4 Wheel Drive

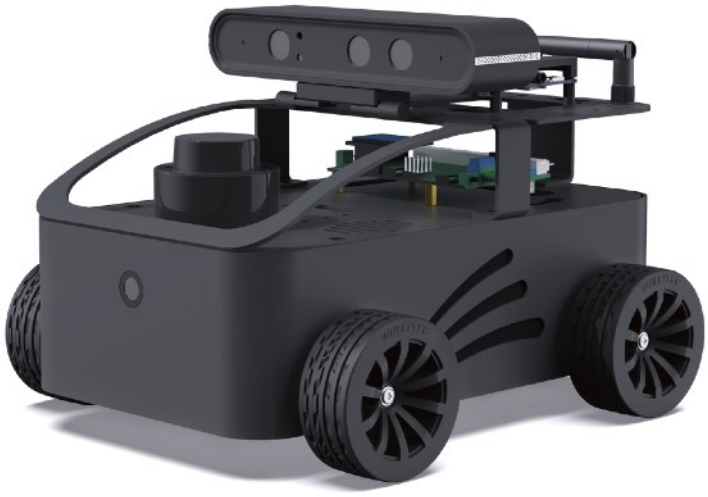
Xbot comes with popular ROS controllers such as:

- Jetson - Orin Nano
- Jetson - Orin NX
- Raspberry Pi 5

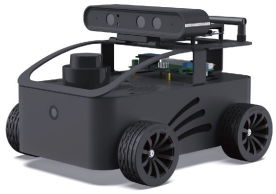
1. Key Components



Models	Image
Xbot - Model A	

Xbot - Model M	
Xbot 4WD	

2. Product Specifications

	Model A	Model M	4WD
Xbot			
ROS Computer	Raspberry Pi5 8G, Jetson Orin Nano 8G or Orin NX 8G		
Low Level Controller	STM32 Board		
LiDAR	LS M10P - 30 meters range		
Depth Camera	Astra Depth Camera		
Software System	ROS 2 Humble on Ubuntu, MiROS Visual ROS Programming Tool, Mobile Apps		
Dimension	289x195x185 mm	255x235x156 mm	255x220x186 mm
Weight	3.3 kg	4.2 kg	4 kg
Payload	4 kg		
Wheel Size (Diameter)	65 mm	75 mm	65 mm
Max Speed	1m/s		
Power Supply	12V, 5100 mAh battery, 2A charger		
Battery Life	5 hours without load, 3.5 hours with 1kg payload		
Motor and Reduction Ratio	MG513 Metal Gear Reduction Motor		
Encoder	500-line AB phase high-precision GMR encoder		
I/O Interface	CAN, Serial Ports, USB, HDMI		
Remote Control	iOS/Android Apps, Wireless PS2, MiROS & ROS		

3. Introduction of ROS Controllers

There are 3 types of ROS Controllers available for use with the Xbot based on Nvidia Jetson platform and Raspberry Pi 5. Jetson Orin Nano is ideal for education and research. Jetson Orin NX is used more often in prototyping and commercial applications. Raspberry Pi 5 is the most affordable Xbot platform.

The following table illustrates the main technical differences between the various controllers available from MiRobot. Both boards allow high level computation and are suited towards advanced robotic applications such as computer vision, deep learning and motion planning.

ROS Controller	Raspberry Pi 5	Jetson Orin Nano	Jetson Orin NX
CPU	ARM Cortex-A76 64bit@2.4GHz Quad-core	ARM Cortex-A57 64bit@1.43GHz Quad-core	ARM Cortex-A78AE v.82 64bit 1.5MB L2+4MB L3
GPU	VideoCore VII @800MHz	128-core MaxWell @921 MHz	32 Tensor Core, 1024 Core NVIDIA Ampere GPU @765 MHz
Computing Power	0.8TOPS (FP16)	0.5TOPS	70 TOPS
Memory	8GB	8GB	8GB
USB Ports	2 USB 3.0 + 2 USB 2.0	4 USB 3.0	3 USB 3.0 + 1 USB 2.0 + 1 Type C
HDMI Ports	Available	Available	Available

4. Sensing System: LiDAR & Depth Camera

An M10 Leishen LSLiDAR is installed on all Mecabot models being. These 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 detection range of 30 metres and a scan frequency of 12Hz. This LiDAR integrates seamlessly into the Mecabots, ensuring all mapping and navigational uses can be easily achieved in your project.

The below table is a benchmark among 3 different kinds of LiDARs:

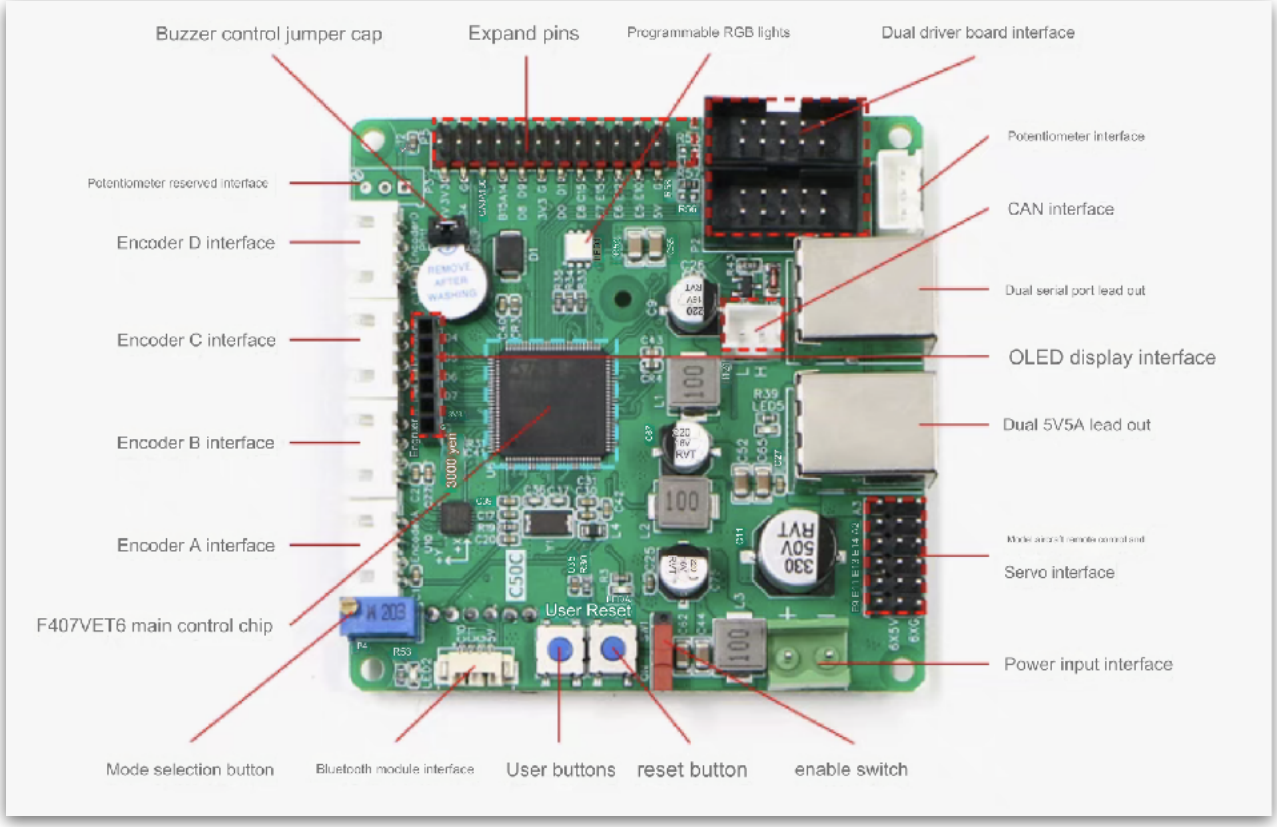
LS LiDAR	N10	M10	C16 (3D)
Detection Range	25m	30m	70/120/150 m
Scan Frequency	10Hz	12Hz	5/10/20Hz
Samples Frequency	4,500Hz	20,000Hz	240,000Hz
Output Contents	Angular, Distant and Light Intensity Data	Angular and Distant Data	Angular, Distant, Time Stamp and Light Intensity Data
Angular Resolution	0,8	0,22	1~2
Interface Type	Serial Port	Ethernet Port	Ethernet Port

Additionally, all Xbots are equipped with an Orbbec Astra 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 Astra Depth Camera	Specs
Depth Resolution	640x480
RBG Resolution	640x480
RGB Sensing Angle	63.1x49.4 degree
Depth Sensing Angle	58.4x45.5 degree
Monocular/Binocular Structural Light	Monocular Structural Light + Monocular RGB
Depth Frame per Second	640x480@30fps
RGB Frame per Second	640x480@30fps
Depth Range	0.6~4m
Data Transfer Interface	USB2.0 or above

5. STM32 Board (Motor Control, Power Management & IMU)

The STM32F103RC Board is the micro-controller used in all Xbots. 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 micro-controller 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.



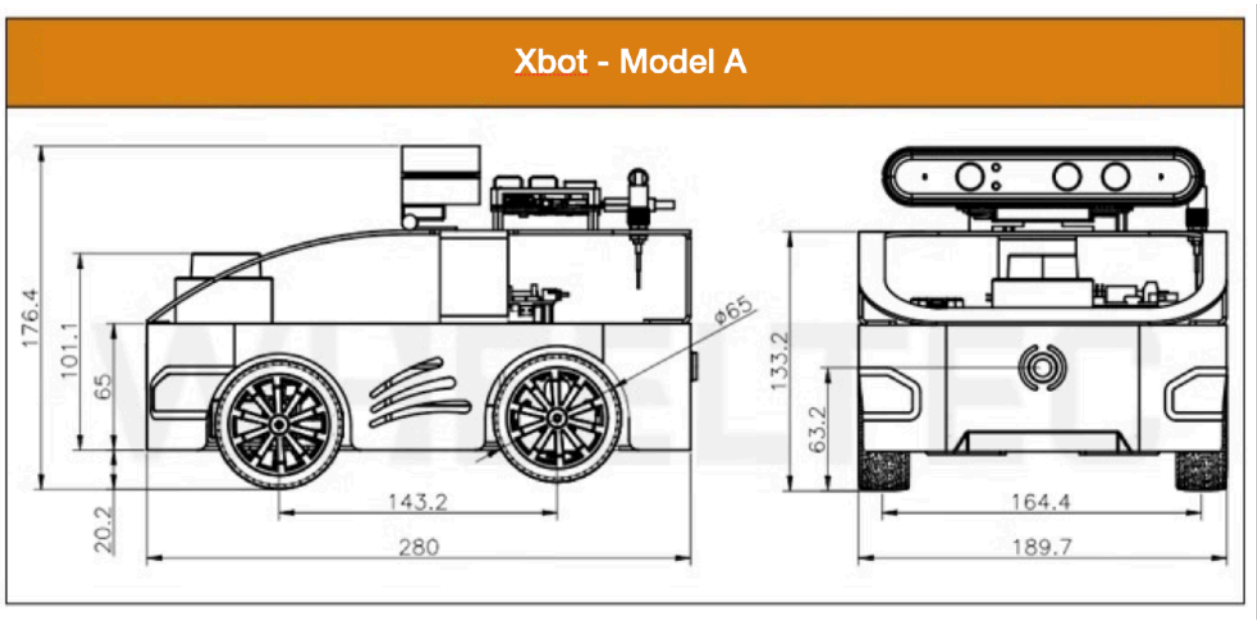
STM32F103RC	Features
Core	ARM32-bit Cortex –M3 CPU Max speed of 72 MHz
Memories	512 KB of Flash memory 64kB of SRAM
Clock, Reset and Supply Management	2.0 to 3.6 V application supply and I/Os
Power	Sleep, Stop and Standby modes V_{BAT} supply for RTC and backup registers
DMA	12-channel DMA controller
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

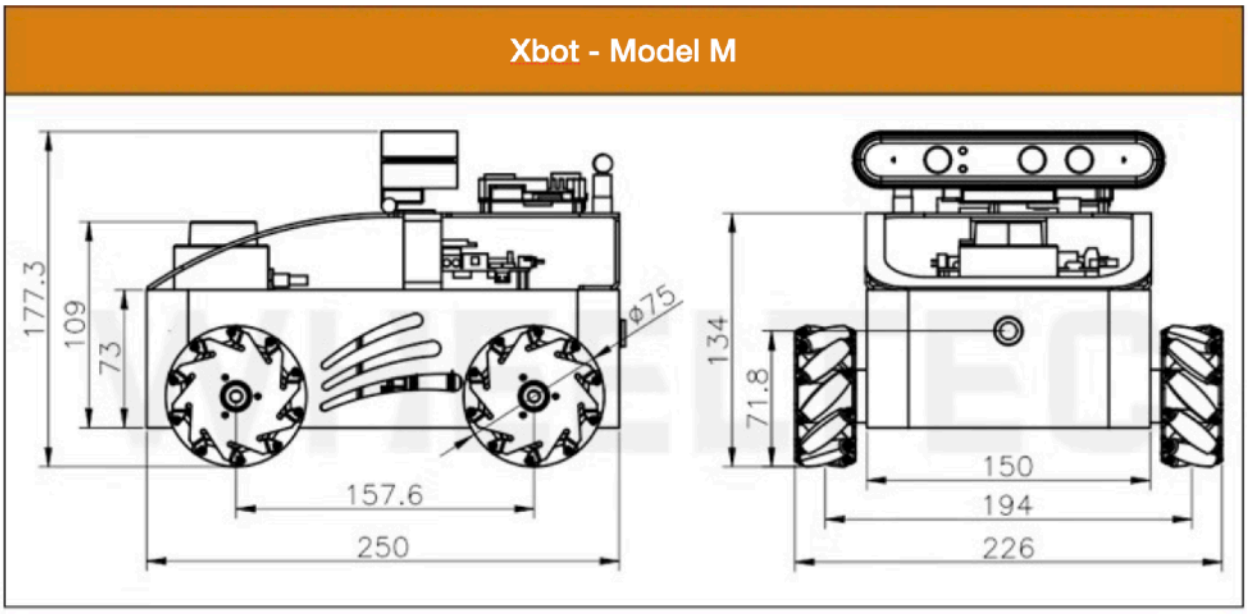
The Steering and Driving system is integrated with the design and build of the Xbot. Depending on the model purchased it will be either a 2 wheel or 4 wheel drive, with both options being suitable to a variety of research and development purposes. The wheels on all Xbots are omnidirectional mecanum wheels with all varieties besides the standard Xbot inclusive of an independent suspension system. The Xbot family of robots are ideal for a wide variety of research and commercial applications making it the perfect robot for your next project.

Design Diagram:

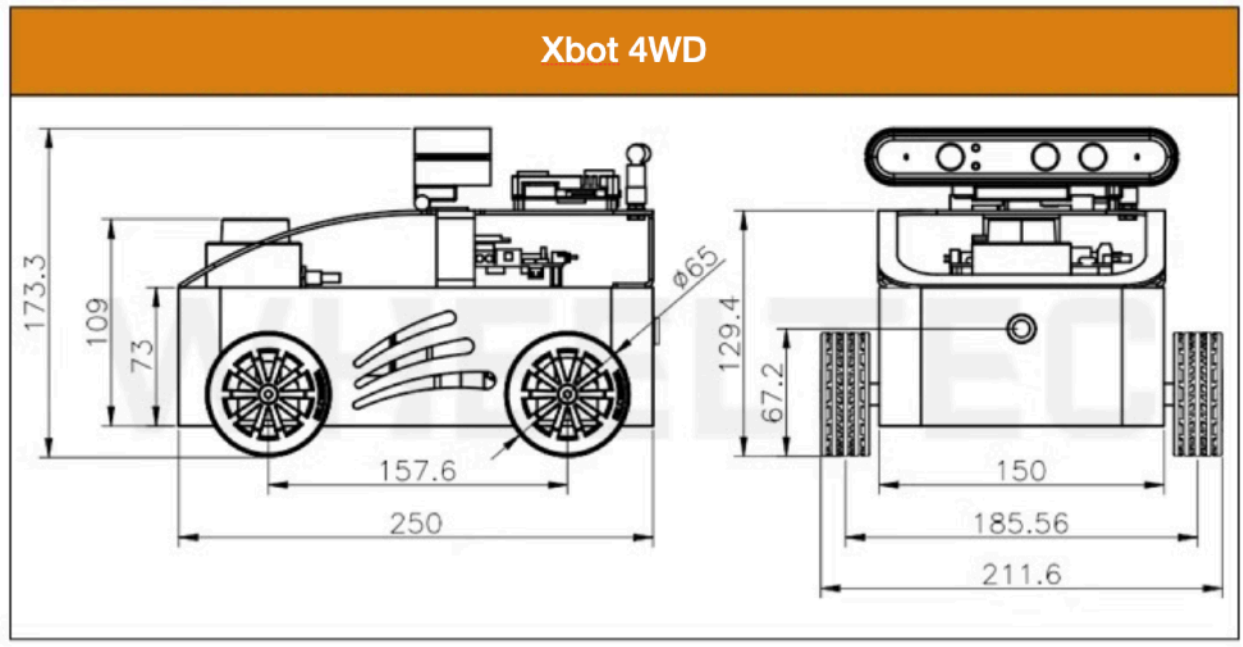
Xbot - Model A



Xbot - Model M:



Xbot 4WD



7. Tele-operation

There are 4 ways to tele-operate the robot:

7.1 Controlled by PS2 controller:

8.1.1. Connect the PS2 controller to the PCB board

8.1.2. Wait until the indicator turns red on the controller and then press the Start button.

8.1.3. On the pcb board screen, push the left joystick forward and change it from ros to ps2 control mode.

The following photo shows the two different control modes: ROS or PS2:

7.2 Controlled by ros2 node and keyboard

7.2.1. Change the control mode to ros

7.2.2. Make sure robot bring up is running (see section 9)

7.2.3. Run this command:

python3 ros2/src/wheeltec_robot_keyboard/wheeltec_robot_keyboard/wheeltec_keyboard.py

7.2.4. Alternatively, you can run this command:

ros2 run wheeltec_robot_keyboard wheeltec_keyboard

7.3 Controlled by ros2 node and a USB A controller

7.3.1. Connect a USB A controller

7.3.2. Change the control mode to ros

7.3.3. Make sure robot bring up is running (see section 9)

7.3.4. Run this command:

ros2 launch wheeltec_joy wheeltec_joy.launch.py

7.4 Controlled by Mobile App via Wifi or Bluetooth connection

Visit Roboworks' App Station website and navigate to the Remote Control Mobile Apps section to download the Mobile App for your mobile phone:

8. 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).

8.1 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:

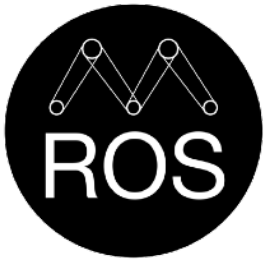
https://www.docker.com/products/docker-desktop/

8.2 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:

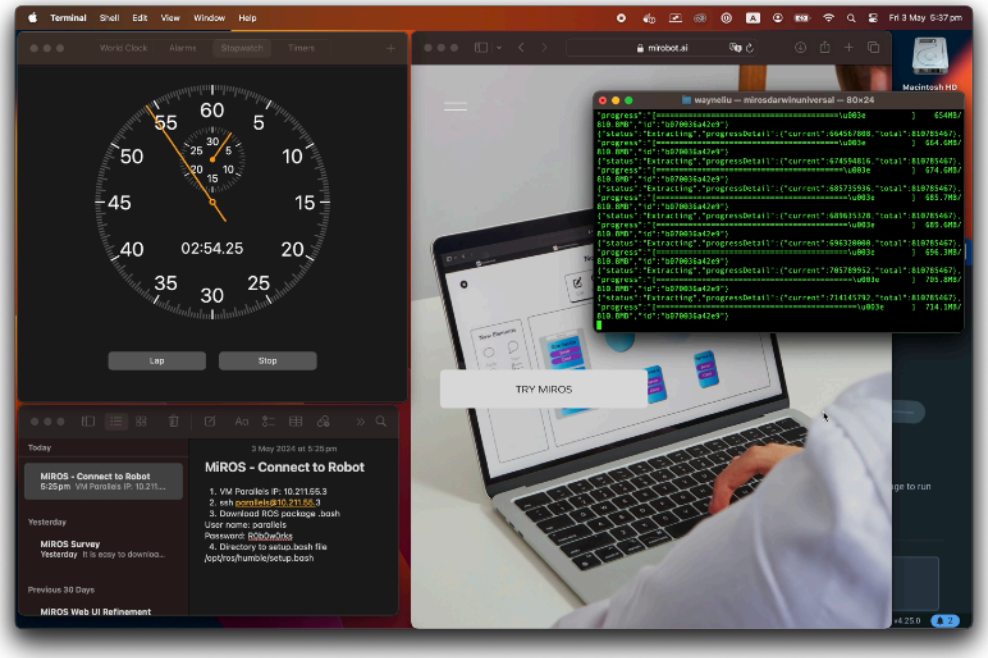


To install MiROS, simply double click the MiROS installer. Once the installation has finished, you will find the MiROS app appears either on your Desktop or in your Application Folder.

To launch MiROS, follow the below steps:

1. Launch Docker Desktop App.
2. Launch MiROS App.

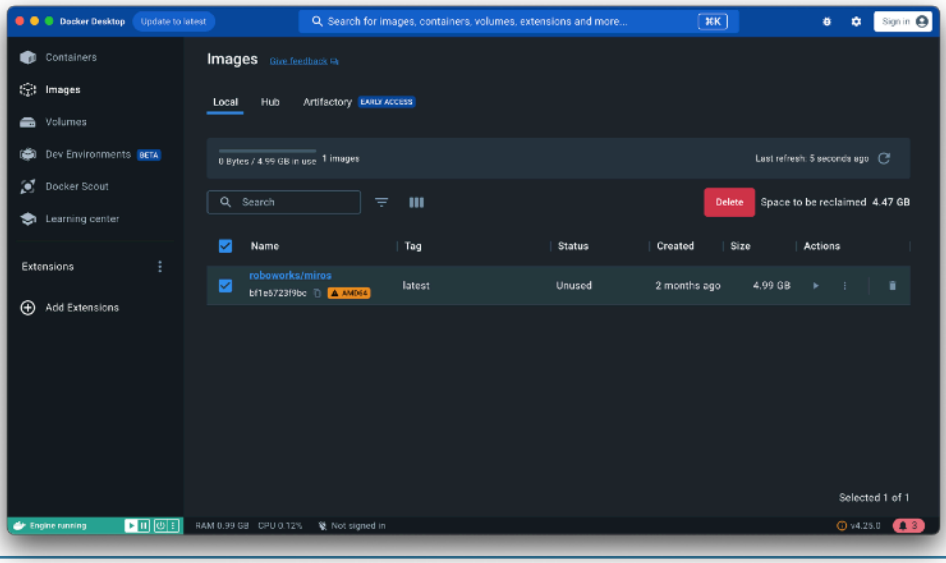
3. You will see a Terminal window appears showing MiROS is pulling the ROS and its associated Ubuntu image from the Cloud to your Docker. Your computer screen could look like the picture shown below:



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:

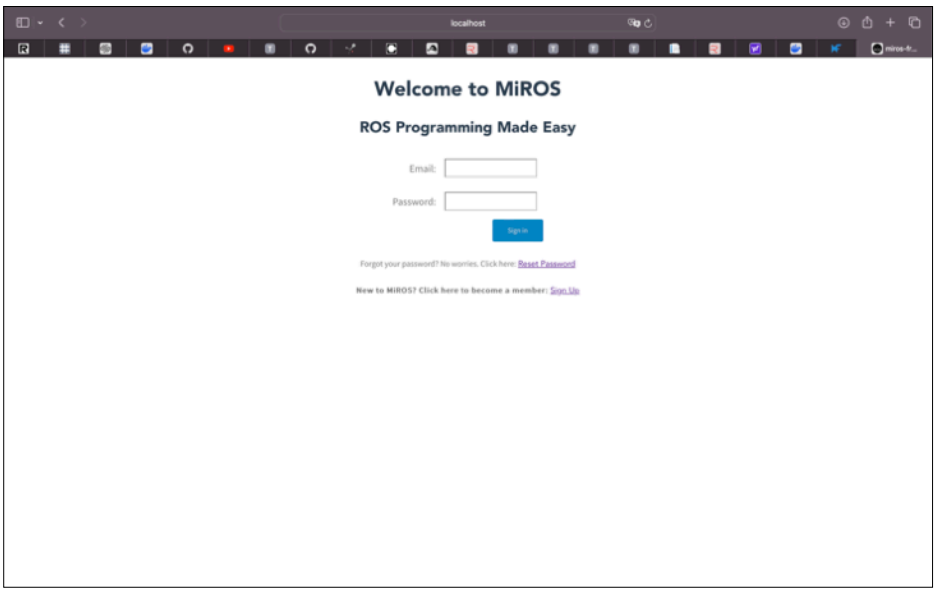
Every time you launch MiROS on your Mac or Windows, you should launch Docker Desktop first. If you have successfully installed MiROS, your Docker Desktop should show the below docker image in your Images section shown as below:



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:

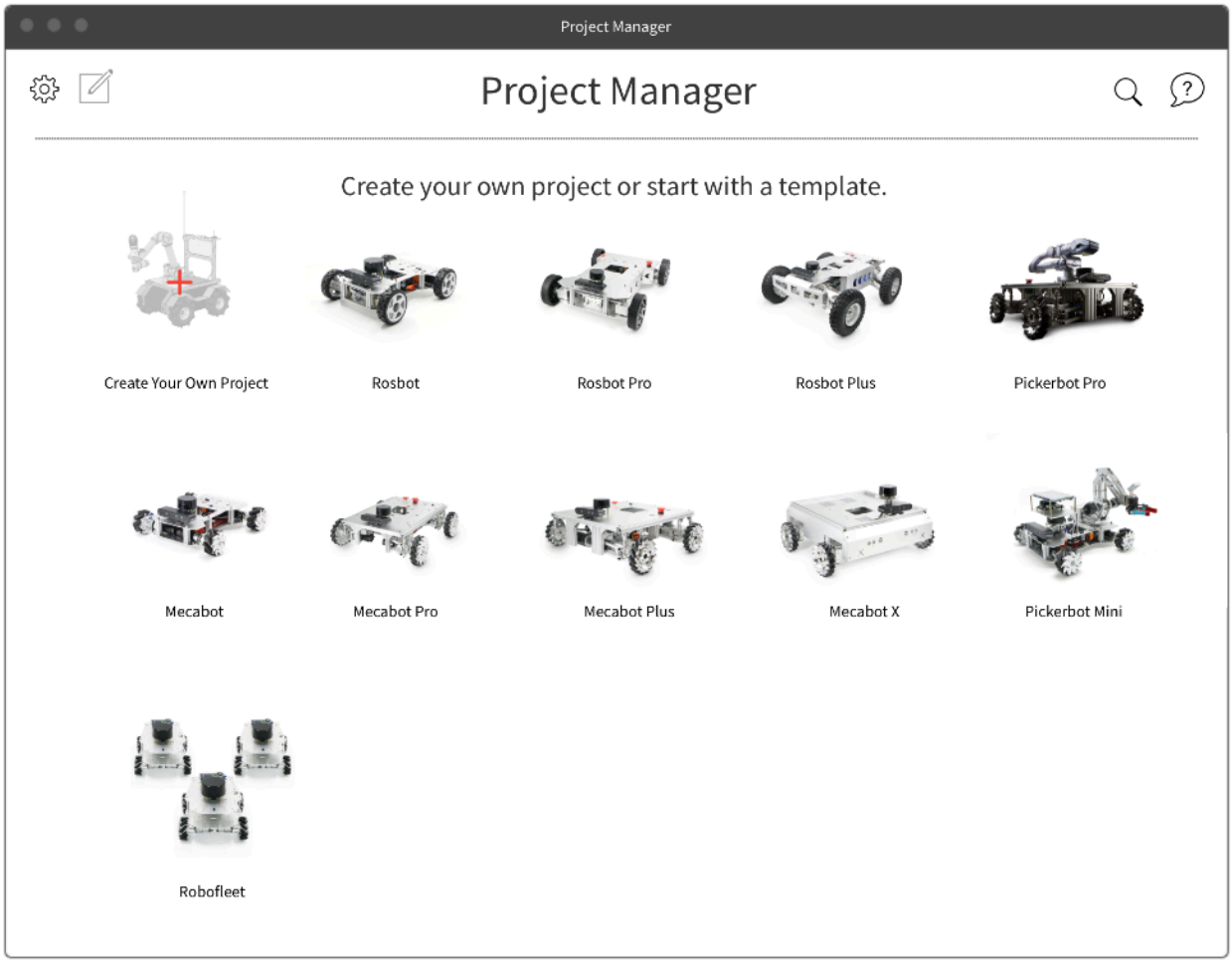
- Save and syn your projects on the MiROS Cloud.
- Access to your MiROS projects via any web browsers on any computers or robots.
- Export your ROS code to any computers or robots.
- Push your latest code on your GitHub repositories from any computers or robots.

Once you log in to MiROS, you will land in Project Manager shown as below:

8.3 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.

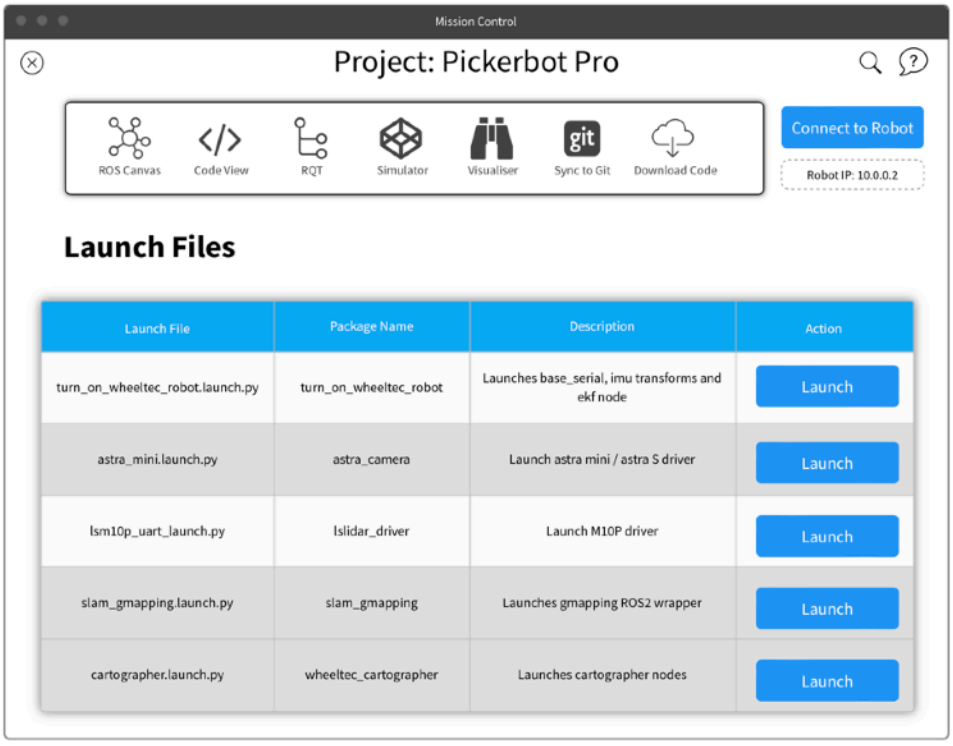
Start from scratch

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:



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.

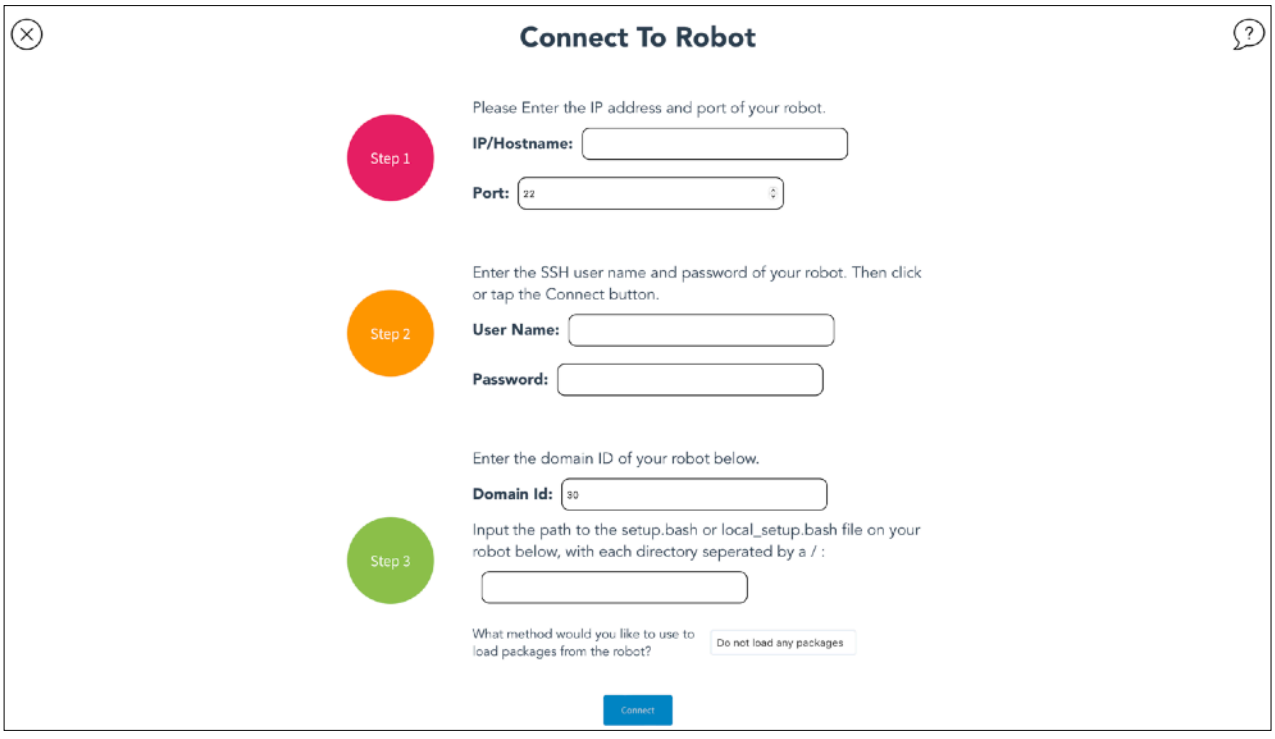
- **Connect to Robot** - a button to trigger connection between MiROS web interface and your robot via local Wifi network.
- **Launch Files** - send launch file commands to your robot via constant ssh connection.

8.5 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:

- Xbot 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



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.

8.6 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:

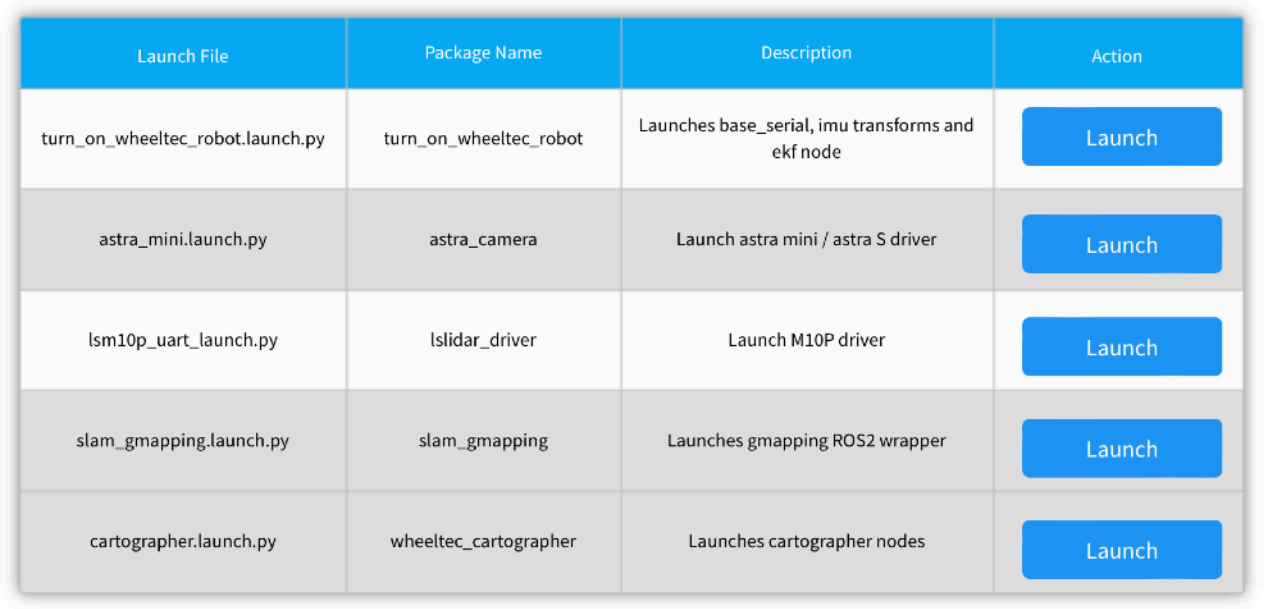
```xml
<launch>
    <!-- Launch node1 -->
    <node name="node1" pkg="package_name" type="node_executable" output="screen">
        <param name="param_name" value="param_value"/>
    </node>

    <!-- 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 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

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.

9. ROS 2 Quick Start

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

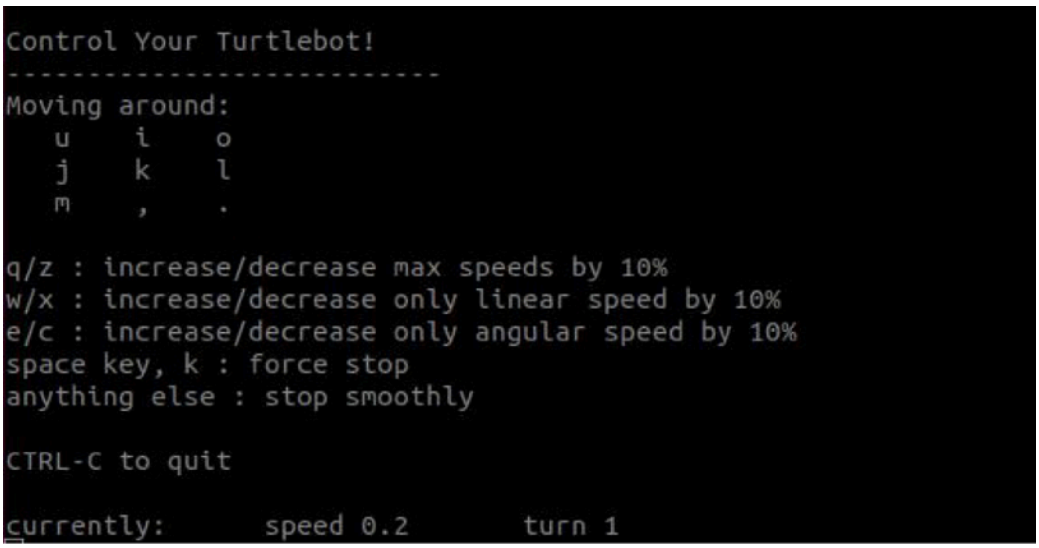
You are now able to test the robot's functionality, to launch the ROS 2 controller functionality, run:

"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"



10. 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.