



Puzzlebot NVIDIA Jetson® / Jetson Lidar Edition.

Introduction

Puzzlebot

{Learn, Create, Innovate};



Puzzlebot: NVIDIA Jetson Edition



Introduction

- The Puzzlebot NVIDIA JETSON® Edition is an extension of the Puzzlebot Hacker Edition encompassing an NVIDIA Jetson® CPU and a Raspberry Pi® Camera.
- Combining the power of the Hacker Board and the NVIDIA JETSON Nano® allows users to implement research-level, real-time algorithms such as AI & Computer Vision, SLAM and autonomous driving algorithms using ROS.
- The Puzzlebot NVIDIA JETSON® Edition works by communicating the Hacker Board (Plug and play) with the NVIDIA Jetson Nano®.

A small graphic of a blue robot head icon, similar to the one in the MCR logo, positioned above the letter 'b' in the word 'Puzzlebot'.

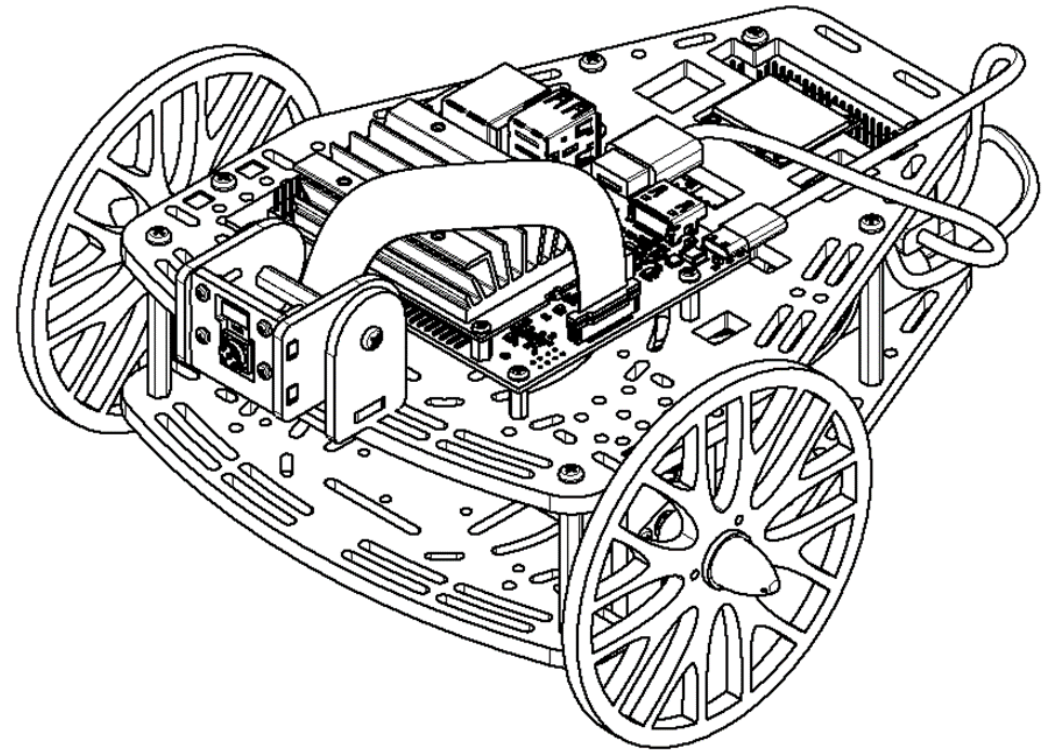
Puzzlebot



Puzzlebot: NVIDIA Jetson[®] Edition



- The following slides will guide the user through the basic usage of the Puzzlebot Jetson[®] Edition.
- The initial configuration will take place in two steps
 - The Hackeboard configuration: In this configuration, the user will learn how to
 - The NVIDIA Jetson[®] configuration.



NVIDIA Jetson Nano

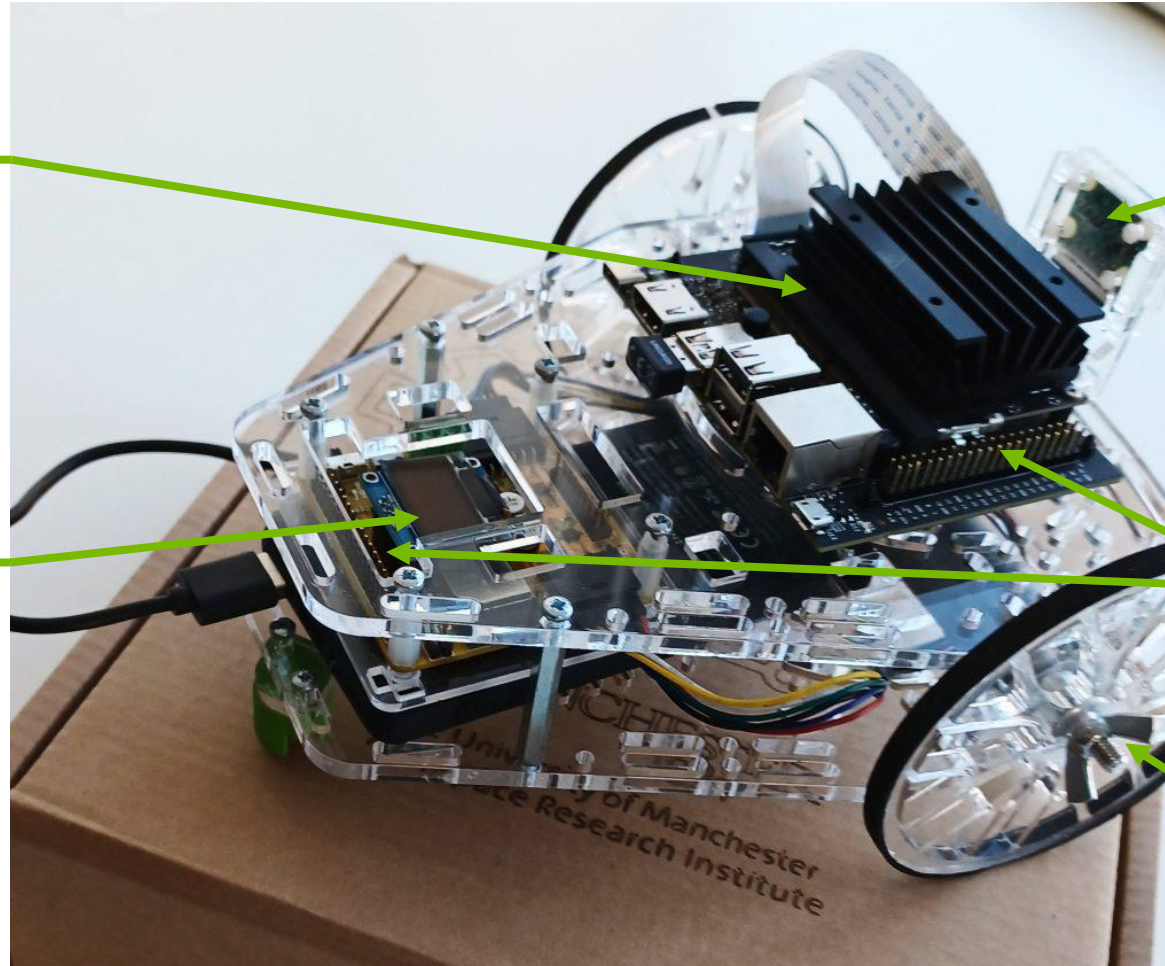
For AI and computer vision

- Higher processing power
- Time-sharing operating system
- Good for more complex, slower tasks
- Specifically designed by NVIDIA for AI applications

Hackerboard

For low-level control algorithms

- Low processing power
- Real-time operating system
- Good for simple, fast, time-sensitive tasks



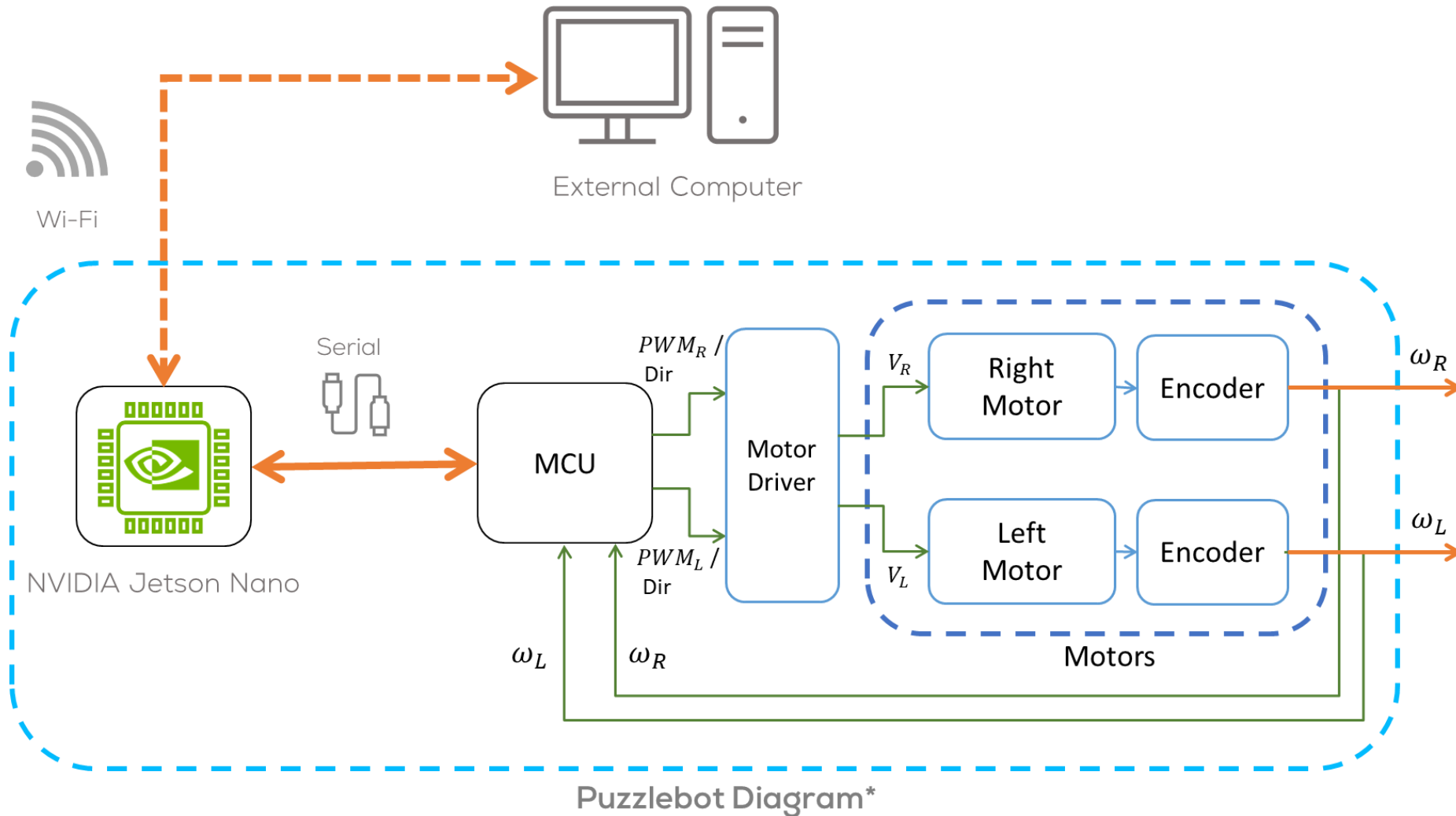
Raspberry Pi Camera

GPIO Arrays

Expansion is possible via the Jetson or the Hacker Board.

Puzzlebot Hacker Edition

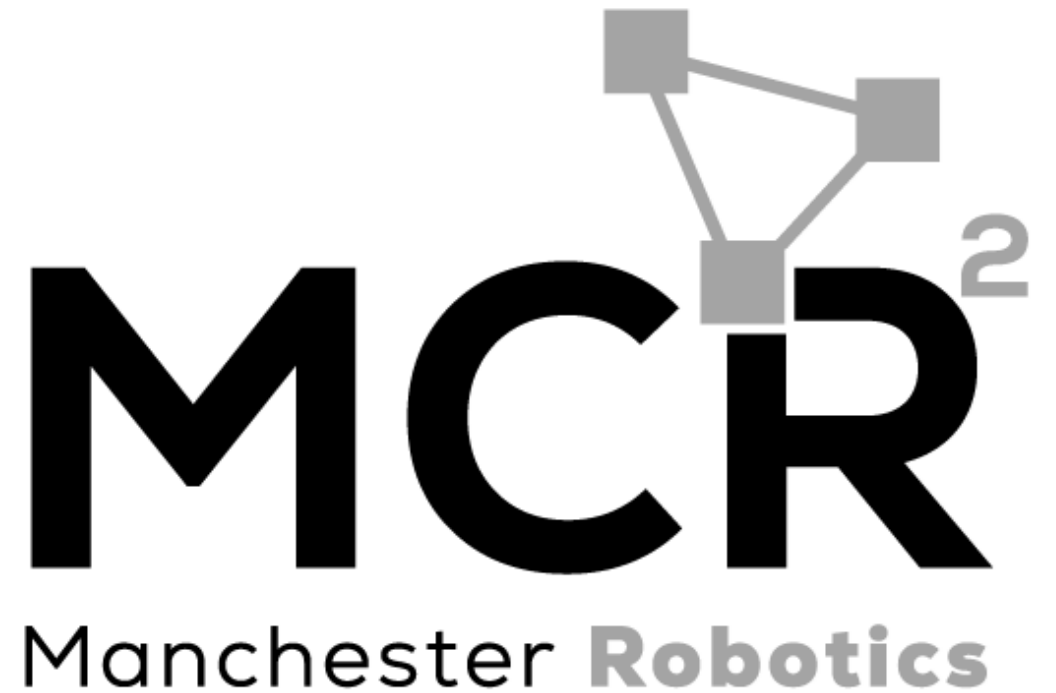
Puzzlebot Diagram



Hackerboard

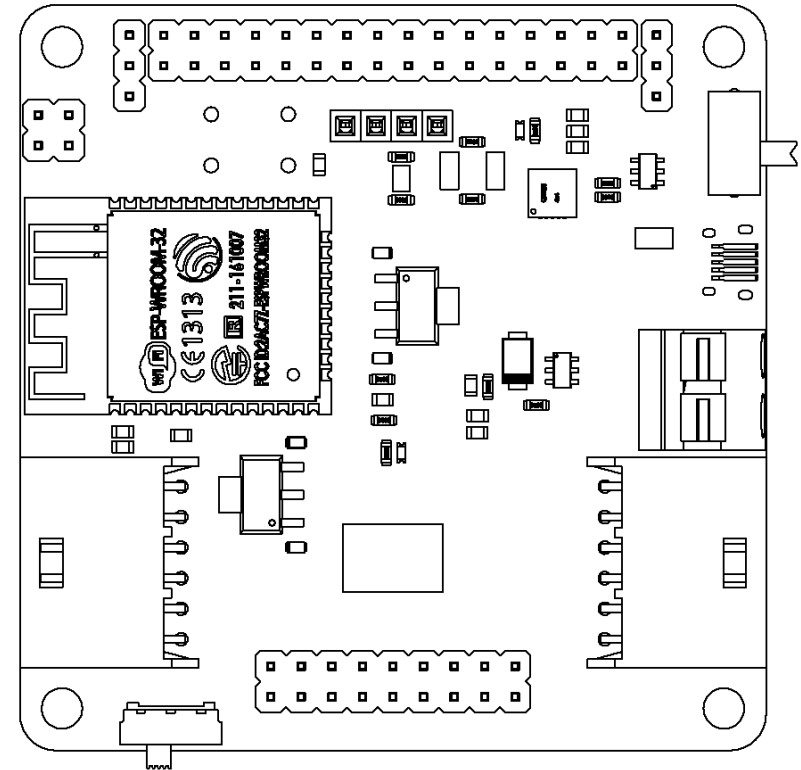
Introduction

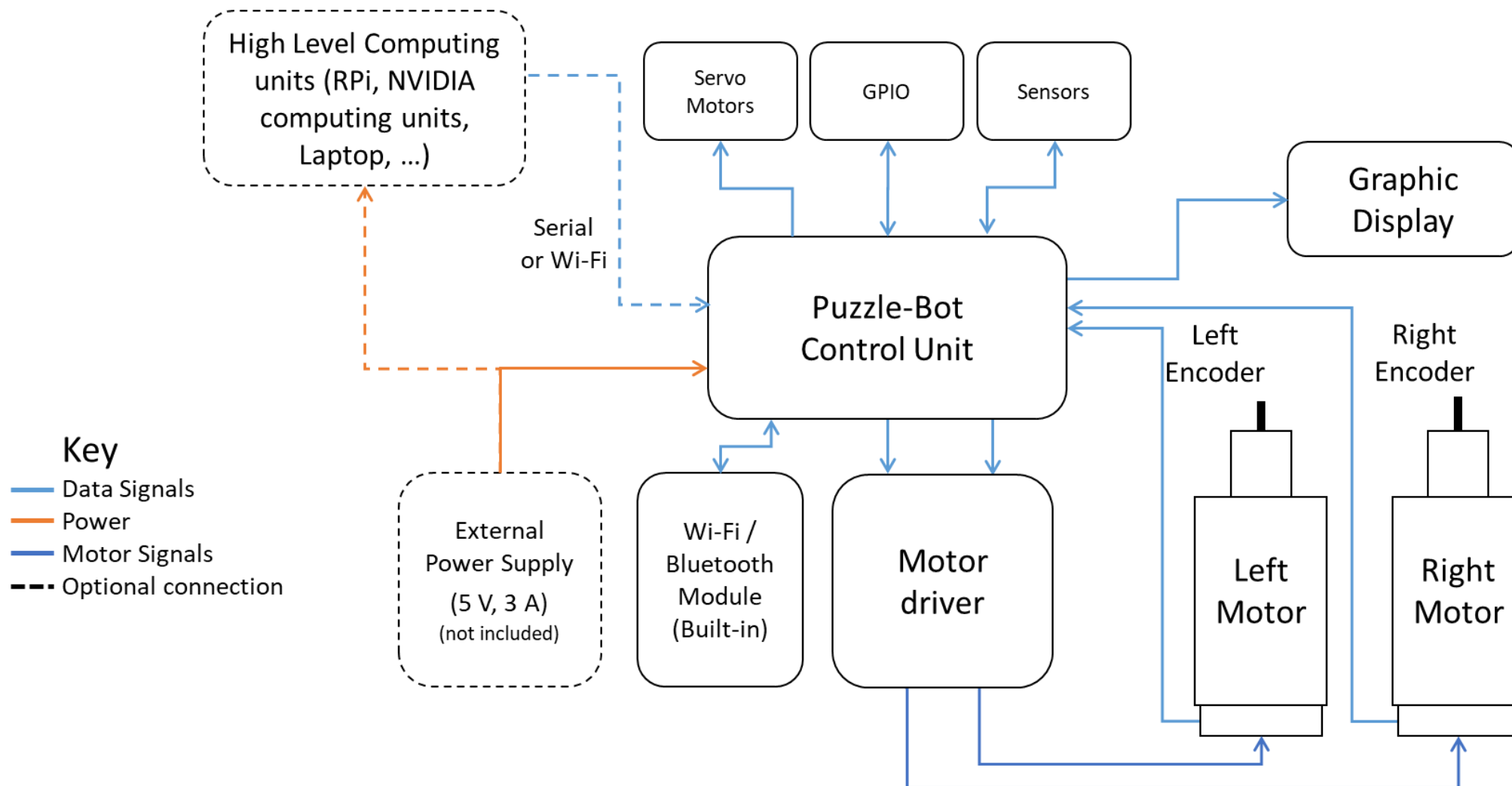
{Learn, Create, Innovate};



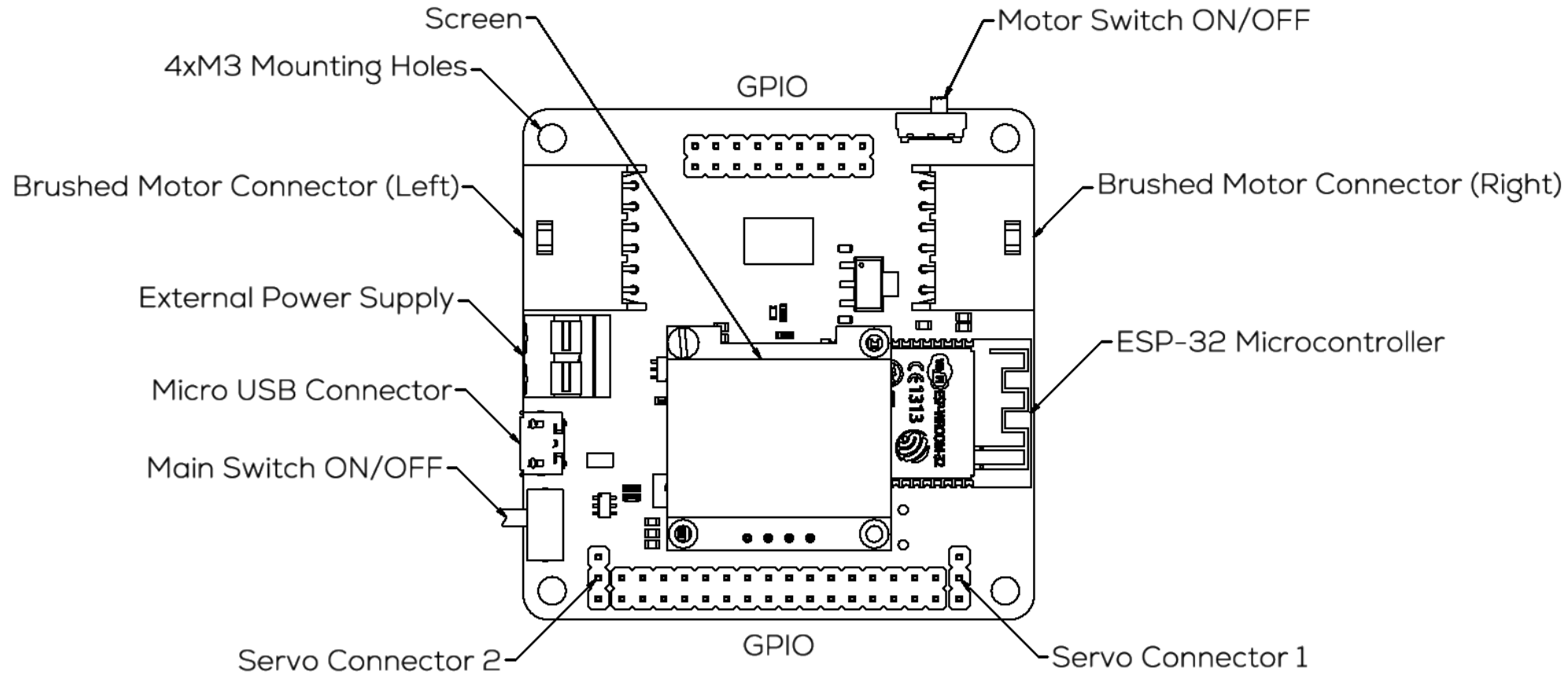
General characteristics

- ESP32-based Microcontroller
 - Xtensa dual-core 32-bit LX6 microprocessor
 - 520 KB of SRAM
 - Wi-Fi & Bluetooth
- DC-DC Converter
- Motor Driver
- 0.96" I2C LCD Display





The Hacker Board





Control Modes

The Hackerboard has two different control modes depending on the user's requirements.

- The two programming configurations:
 - Standalone Configuration
 - External-Control Configuration

Control Mode: Standalone Configuration (Information purpose only)

- The user directly programs the Hacker Board, using the Arduino IDE.
- Libraries for control and communication with computing units, sensors, and actuators are provided by MCR2.
- 3rd Party peripherals can be attached.
- This configuration will not be used for this Puzzlebot version.
- For additional examples and a more in-depth understanding, please consult the "Puzzlebot Hacker Edition" manual.



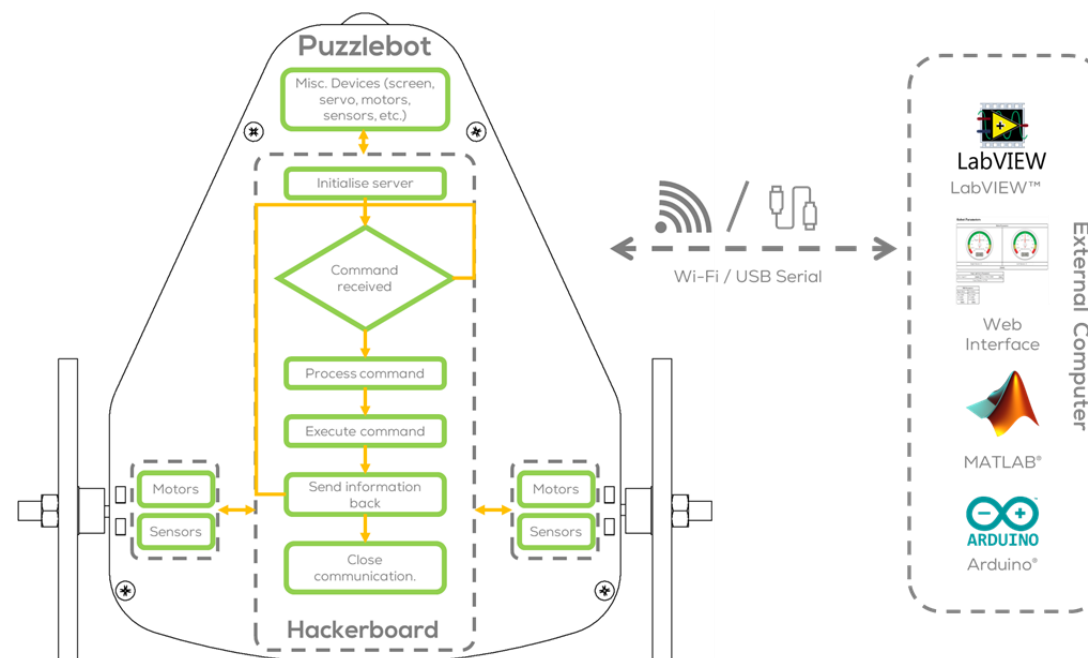
Hackerboard



Control Mode: External Control Configuration

The robot is controlled from an external computer via Wi-Fi or Serial Communication.

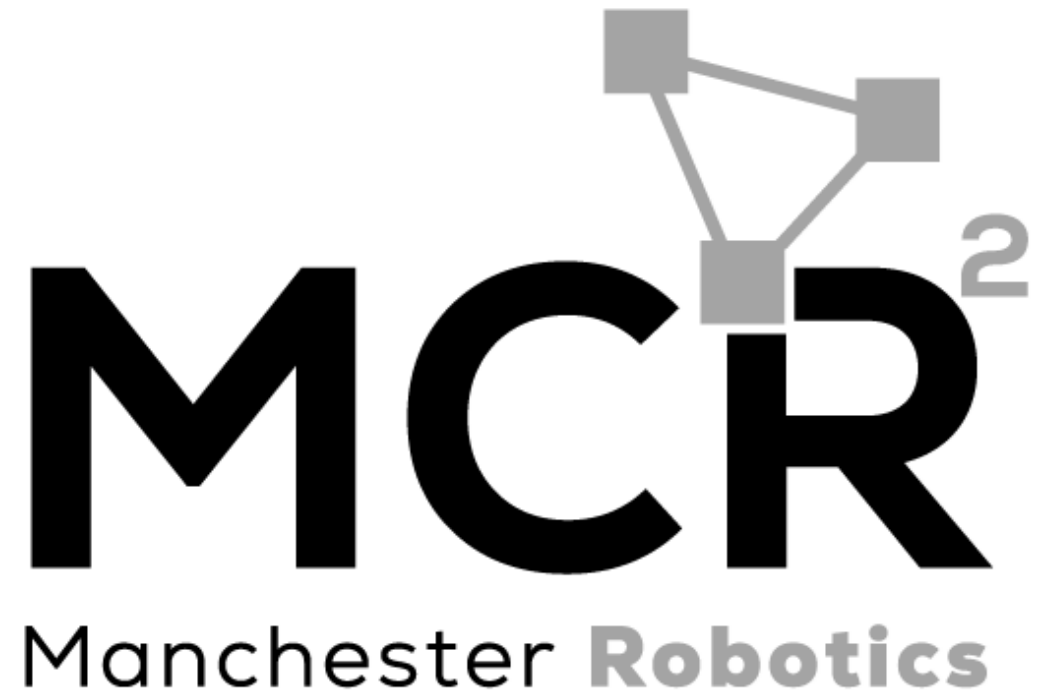
- The internal firmware and libraries for communicating with the robot's sensors and actuators are provided by MCR2.
- Basic web interface for configuring and testing provided.
- MCR2 provides MATLAB, ROS and LabVIEW libraries for communicating with the robot.
- This configuration will be used for this Puzzlebot version.
- For additional examples and a more in-depth understanding, on how to use MATLAB or LabVIEW communications and simulators, contact us.



Hackerboard

*Flashing the
Hackerboard*

{Learn, Create, Innovate};





Hackerboard: Flashing the Binaries



Flashing Hackerboard

- MCR2 provides the firmware binaries for the External Computing Unit Control Mode.
- This section will guide the user on how to flash such binaries.
- All the robots come preprogrammed with such binaries unless the user modifies the program (On Board Configuration).
- The original binaries can be flashed anytime by following the steps in this section.





Hackerboard: Flashing the Binaries.



Steps

1. Attach the micro-USB cable to the Puzzle-Bot Control Module. *Note the image is just for explanation purposes, detachment of the Puzzle-Bot Control module from the robot is not necessary.*
2. Make sure the Hackerboard power switch is turned on.
3. Connect the USB to any free USB port in the computer
4. Download the firmware from the [MCR2 GitHub](#) and extract it.
5. Select the OS of the computer that will perform the flash.
6. Run the file *"FirmwareFlash"*
 - For Linux and MacOS users, it may be necessary to set execution permissions in properties.

On/Off Switch





Hackerboard: Flashing the Binaries.



Steps

7. Select the SSID identification for the Hackerboard and press Enter (recommended to use a different one if several Puzzlebots are being used simultaneously).
8. Select the password for the Hackerboard's Wi-Fi Network and press Enter (recommended to use a different one if several Puzzlebots are being used simultaneously).
9. Wait until the program finish flashing.
10. The screen must turn on.

```
C:\WINDOWS\system32\cmd. X + v
PuzzleBot found on COM4
Enter details for COM4
Input SSID, leave blank for default SSID (Puzzlebot):
```

```
PuzzleBot found on COM4
Enter details for COM4
Input SSID, leave blank for default SSID (Puzzlebot):
Enter password, leave blank for default password (Puzzlebot72):
```



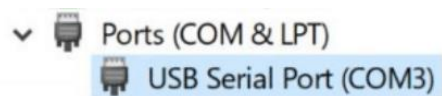


Troubleshoot



Troubleshoot (Drivers)

- Drivers are usually installed automatically by Windows and Ubuntu even for the Virtual Machines.
- How do I know if the drivers are properly installed (Windows)?
 - Plug the Puzzle-Bot into the USB port.
 - Go to Start > Device Manager
 - The Serial port should appear as shown in the following figure (The COM port may vary).



- If the computer cannot find the drivers, download the drivers from the following link
<https://ftdichip.com/drivers/vcp-drivers/>
- Verify that the USB cable is a data cable and not only a power cable!

- Scroll down and download the executable setup as shown in the following figure

Operating System	Release Date	X86 (32-Bit)	X64 (64-Bit)	PPC	ARM	MIPSII	MIPSIV	SH4	Comments
Windows*	2021-07-15	2.12.36.4	2.12.36.4	-	-	-	-	-	WHQL Certified. Includes VCP and D2XX. Available as a setup executable  Please read the Release Notes and Installation Guides .
Linux	-	-	1.5.0	-	-	-	-	-	All FTDI devices now supported in Ubuntu 11.10, kernel 3.0.0-19. Refer to TN-101 if you need a custom VCP VID/PID in Linux. VCP drivers are integrated into the kernel .
Mac OS X 10.3 to 10.8	2012-08-10	2.2.18	2.2.18	2.2.18	-	-	-	-	Refer to TN-105 if you need a custom VCP VID/PID in MAC OS
Mac OS X 10.9 to 10.14	2019-12-24	-	2.4.4	-	-	-	-	-	This driver is signed by Apple

Before Installing the drivers!!

- Unplug the Puzzlebot from the computer.
- Unzip the drivers and run the setup (some computers must be restarted after the installation).
- Plug the Puzzlebot back into the computer.





Troubleshoot



Troubleshoot (Drivers)

- Some Hackerboard have a different USB-UART chip the CP210x.
- Drivers are usually installed automatically by Windows and Ubuntu even for the Virtual Machines.
- Verify if they are installed by following the steps in the previous slide.
- Verify that the USB cable is a data cable and not only a power cable!.
- If the computer cannot find the drivers, download the drivers from the following link

<https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads>

Before Installing the drivers!!

- Unplug the Puzzle-Bot from the computer.
- Unzip the drivers and run the setup (some computers are required to be restarted after the installation).
- Plug the Puzzle-Bot back into the computer.

A troubleshoot guide can be found [here](#).





Troubleshoot



Troubleshoot (Drivers)

- My computer still not recognize the drivers even after the installation
- Plug the Puzzle-Bot into the USB port.
- Go to Start > Device Manager.
- Look for the USB Serial Converter as shown in the following picture.

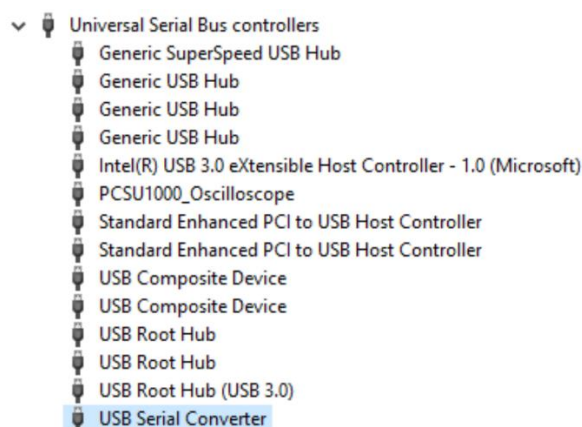


FIGURE: USB SERIAL CONVERTER

- Right Click to Properties > Advanced Tab.
- Make sure the Load VCP box is checked.
- Reconnect the Puzzle-Bot to the computer.

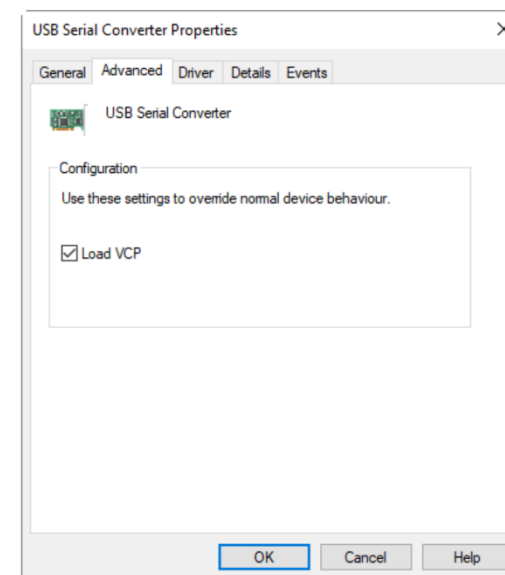
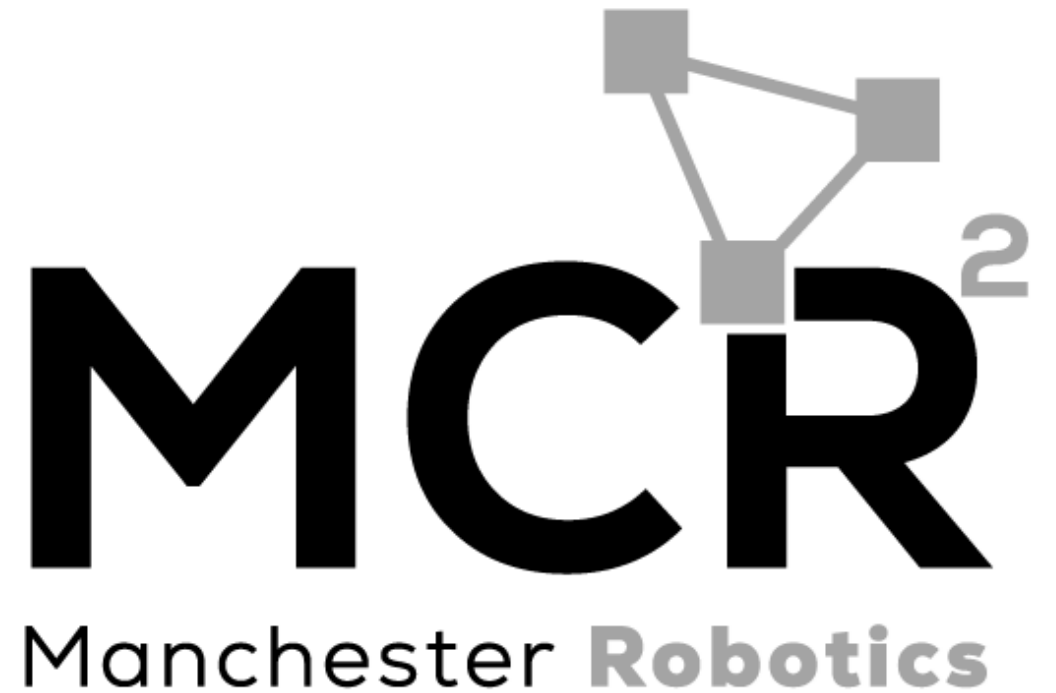


FIGURE: VCP PORT

Hackerboard

*Connections and configurations
for the Jetson Nano*

{Learn, Create, Innovate};





Connecting to the Hackerboard



Puzzlebot Web Interface

The Puzzlebot has a web interface allowing it to configure and test the different sensors and actuators equipped in the robot.

The website offers a visual interface that enables the user to configure and test various internal settings of the robot, including motor controllers, sensor activation, actuator activation (if applicable), and communication with external computing units.



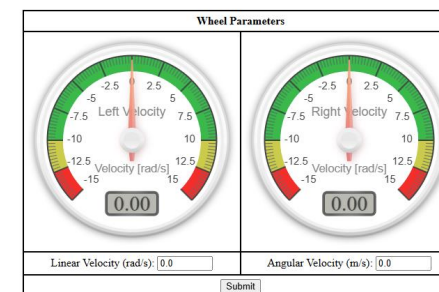
Restart Robot

Active Modules	
Servo Motor	<input type="checkbox"/>
Time-of-flight: Sonar	<input type="checkbox"/>
Time-of-flight: Laser	<input type="checkbox"/>
Reflectance Line Sensor	<input type="checkbox"/>
LIDAR	<input type="checkbox"/>
Screen	<input checked="" type="checkbox"/>
<input type="button" value="Save"/>	

Network Settings	
SSID:	<input type="text" value="Puzzlebot"/>
Password:	<input type="text" value="Puzzlebot72"/>
<input type="button" value="Save"/>	

Robot Parameters

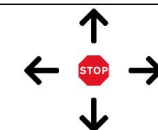
[Change Configuration](#)



Reset to Default Config

Motor-Encoder Settings		
Control Mode	<input type="button" value="Robot Velocities (v and u)"/>	
Invert Directions	Left	Right
Motors	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Encoders	<input type="checkbox"/>	<input type="checkbox"/>
<input type="button" value="Save"/>		

Robot Controls	
On-screen Controls	<input type="checkbox"/>
Keyboard Controls	<input checked="" type="checkbox"/>

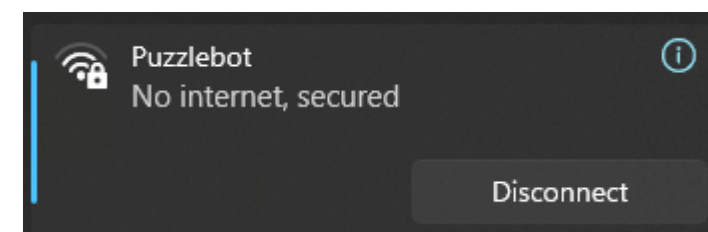
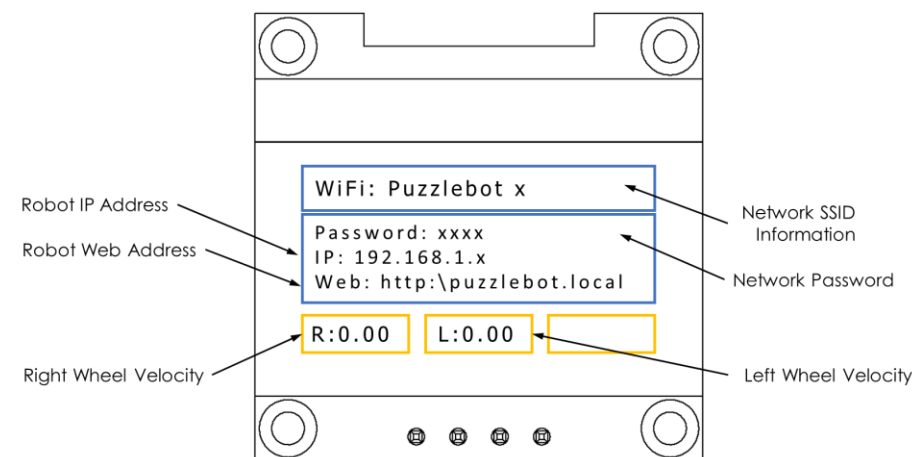
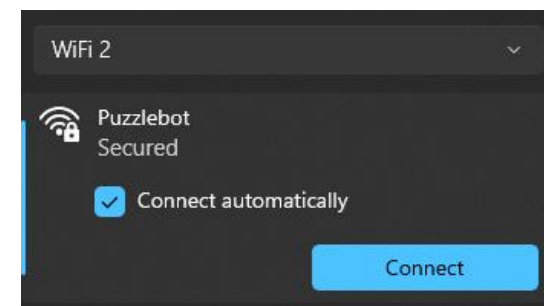




Connecting to the Hackerboard



1. Go to the Wi-Fi Network connections.
2. Connect to the Wi-Fi network created by the Puzzlebot by choosing the Puzzlebot Wi-Fi network of the robot you want (Puzzlebot x), then select Connect.
3. Type the network password that you selected when flashing (shown on the LCD Display), and then select Next.
4. Once Connected the network should say “No internet, secured” (Windows only).





Connecting to the Hackerboard



Connecting to the Puzzlebot Web Interface

1. Open a web browser



2. Type the following IP address on the search bar

	<input type="text" value="http:\\192.168.1.x"/>
--	---

3. The Puzzlebot Robot Parameters interface should load.



Restart Robot

Active Modules	
Servo Motor	<input type="checkbox"/>
Time-of-flight: Sonar	<input type="checkbox"/>
Time-of-flight: Laser	<input type="checkbox"/>
Reflectance Line Sensor	<input type="checkbox"/>
LIDAR	<input type="checkbox"/>
Screen	<input checked="" type="checkbox"/>
<input type="button" value="Save"/>	

Network Settings	
SSID:	<input type="text" value="Puzzlebot"/>
Password:	<input type="text" value="Puzzlebot72"/>
<input type="button" value="Save"/>	

Robot Parameters

[Change Configuration](#)

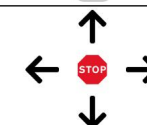
Wheel Parameters	
Linear Velocity (rad/s): <input type="text" value="0.0"/>	Angular Velocity (m/s): <input type="text" value="0.0"/>
<input type="button" value="Submit"/>	



Reset to Default Config

Motor-Encoder Settings		
Control Mode	<input type="button" value="Robot Velocities (v and w)"/>	
Invert Directions	<input checked="" type="checkbox"/> Left	<input checked="" type="checkbox"/> Right
Motors	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Encoders	<input type="checkbox"/>	<input type="checkbox"/>
<input type="button" value="Save"/>		

Robot Controls	
On-screen Controls	<input checked="" type="checkbox"/>
Keyboard Controls	<input type="checkbox"/>





Configuring the Hackerboard



- **Manual robot reboot:** Manually reboots the Hackerboard.
- **Factory Reset:** Reset all the configurations to factory configurations.
- **Network settings:** Configuration of the SSID and password of the Puzzlebot. Useful when multiple Puzzlebots are being used.
- **Puzzlebot controls:** Simple Puzzlebot controls for the user to move the robot forward, backwards or turning.
- **Sensor/actuators and comms. modules:** Activation of the sensors/actuators if included connected to the Hackerboard. The sensors and actuators in this section can connect to ROS. More information on the Puzzlebot Manual.

****This slide is only for informative purposes and is not required for basic Jetson connection.**

Manual robot reboot → **Restart Robot**

Robot Parameters → [Change Configuration](#) → Advanced config.

Factory reset → **Reset to Default Config**

Sensors/actuators and communication modules

Active Modules	
Servo Motor	<input type="checkbox"/>
Time-of-flight: Sonar	<input type="checkbox"/>
Time-of-flight: Laser	<input type="checkbox"/>
Reflectance Line Sensor	<input type="checkbox"/>
LIDAR	<input type="checkbox"/>
Screen	<input checked="" type="checkbox"/>
ROS	<input type="checkbox"/>
Save (?)	

Network settings

Network Settings	
SSID:	Puzzlebot
Password:	Puzzlebot72
Save (?)	

Wheel Parameters

Motor testing

Motor-Encoder Settings

Motor-Encoder Settings		
Control Mode (?)	Motor PWMs	
Invert Directions	Left	Right
Motors (?)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Encoders (?)	<input type="checkbox"/>	<input type="checkbox"/>
Save (?)		

Motor - Encoder settings

Motor PWM Controls

Motor PWM Controls	
On-screen Controls	Keyboard Controls

Puzzlebot controls

Controls: L (Left), R (Right), STOP (Red octagon), + (Forward), - (Backward)

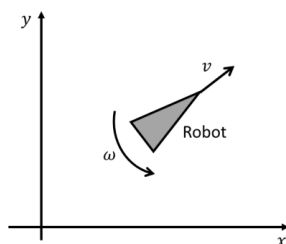
*Moving the cursor over the question marks (?) displays more information about each configuration parameter.




Configuring the Hackerboard



- **Advanced configuration:** Manually configure all aspects of the Puzzlebot. Warning! adjusting the parameters of the Puzzlebot could result in significant malfunctions.
- **Motor-Encoder Settings:** Motor and Encoder configurations.
 - Motor PWM: Values in the range of $[-1, 1]$ (No control)
 - Wheel velocities (ω_L and ω_R): Wheel velocities (Inner PID control used)
 - Robot Velocities (v and ω): Linear and angular speed of the robot.




Manual robot reboot  **Restart Robot**

Sensors/actuators and communication modules

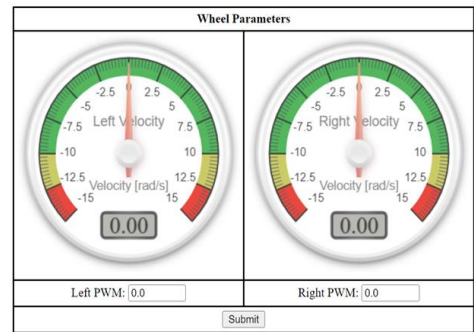
Active Modules	
Servo Motor	<input type="checkbox"/>
Time-of-flight: Sonar	<input type="checkbox"/>
Time-of-flight: Laser	<input type="checkbox"/>
Reflectance Line Sensor	<input type="checkbox"/>
LIDAR	<input type="checkbox"/>
Screen	<input checked="" type="checkbox"/>
ROS	<input type="checkbox"/>
Save	

Network settings


Network Settings	
SSID:	Puzzlebot
Password:	Puzzlebot72
Save	

Robot Parameters [Change Configuration](#) 

Wheel Parameters



Motor testing

 **Factory reset**

Reset to Default Config

Motor-Encoder Settings

Control Mode	Motor PWMs	
Invert Directions	Left	Right
Motors	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Encoders	<input type="checkbox"/>	<input type="checkbox"/>
Save		

Motor PWM Controls

On-screen Controls	<input type="checkbox"/>	Keyboard Controls	<input checked="" type="checkbox"/>
--------------------	--------------------------	-------------------	-------------------------------------

Motor - Encoder settings

Puzzlebot controls

L STOP R

↑ + ↑

↓ - ↓

- **Motor testing:** Manually send values to the motors for testing. The values sent depend on the Control Mode.



Connecting to the Hackerboard



Configuring ROS Serial communication

1. Click on the "Change Configuration" link on the top. This will take you to another website where all the internal configurations of the robot can be seen.
2. In the section "Ros" Change the value of "CommType" to "2".
3. Upload to the robot by pressing the button "Upload to the robot" at the top of the webpage.

Configuration parameters for the robot ("config_live.json" file)

[Home](#)

Upload to robot

Click Here

4. The Hackerboard will restart



Restart Robot



Active Modules	
Servo Motor	<input type="checkbox"/>
Time-of-flight: Sonar	<input type="checkbox"/>
Time-of-flight: Laser	<input type="checkbox"/>
Reflectance Line Sensor	<input type="checkbox"/>
LIDAR	<input type="checkbox"/>
Screen	<input checked="" type="checkbox"/>
Save	

Network Settings	
SSID:	<input type="text" value="Puzzlebot"/>
Password:	<input type="text" value="Puzzlebot72"/>
Save	

Click Here

Robot Parameters

[Change Configuration](#)

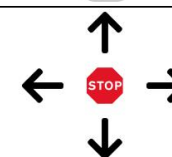
Wheel Parameters	
 Left Velocity Velocity [rad/s] 0.00	 Right Velocity Velocity [rad/s] 0.00
Linear Velocity (rad/s): <input type="text" value="0.0"/>	Angular Velocity (m/s): <input type="text" value="0.0"/>
Submit	



Reset to Default Config

Motor-Encoder Settings	
Control Mode	<input type="text" value="Robot Velocities (v and w)"/>
Invert Directions	<input checked="" type="checkbox"/> Left <input checked="" type="checkbox"/> Right
Motors	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
Encoders	<input type="checkbox"/> <input type="checkbox"/>
Save	

Robot Controls	
On-screen Controls	<input type="checkbox"/> Keyboard Controls



```
"Ros": {  
  "MasterIP": [  
    192,  
    168,  
    1,  
    2  
  ],  
  "CommType": 2,  
  "MasterPort": 9999,  
  "Dt": 0.005  
},
```



Connecting to the Hackerboard



Configuring the Hackerboard

- The ROS topics to be published by the Puzzlebot depend on the “Control Mode” Selected.
 - See table in next slide.
- Having selected the Control Mode and activated the ROS Module, press Save and disconnect from the robot.

Motor-Encoder Settings		
Control Mode ?	Robot Velocities (v and ω) ▼	
Invert Directions	Left	Right
Motors ?	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Encoders ?	<input type="checkbox"/>	<input type="checkbox"/>
Save ?		



Connecting to the Hackerboard

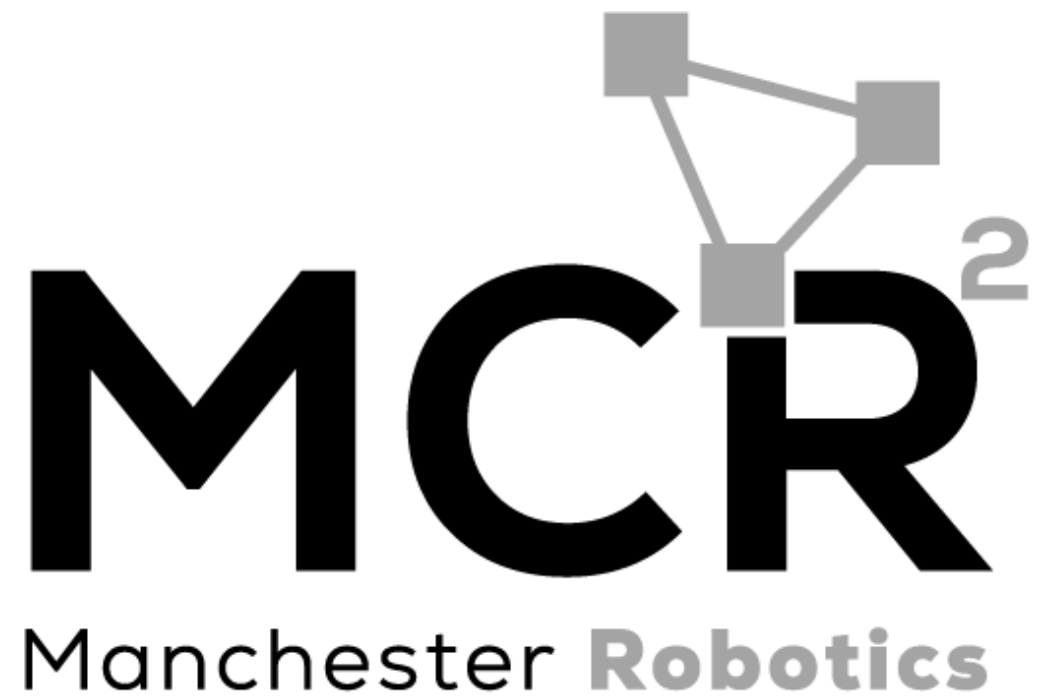


Control Mode	Description	Topic	Type	Information
Motor PWMs	Wheel PWM voltage signal	/ControlL, /ControlR	std_msgs/Float32	data: PWM duty cycle for each motor [-1,1]
Wheel velocities (ω_L and ω_R)	Wheel angular velocities setpoint (PID control)	/VelocitySetR, /VelocitySetL	std_msgs/Float32	data: Control Set Point for wheel velocities
Robot Velocities (v and ω)	Linear and Angular Velocities (PID control)	/cmd_vel	geometry_msgs/Twist	linear x: Linear speed of the robot y: Not used z: Not Used angular: x: Not Used y: Not Used z: Angular Speed of the robot

NVIDIA Jetson Nano[®]

*Connections and
configurations*

{Learn, Create, Innovate};



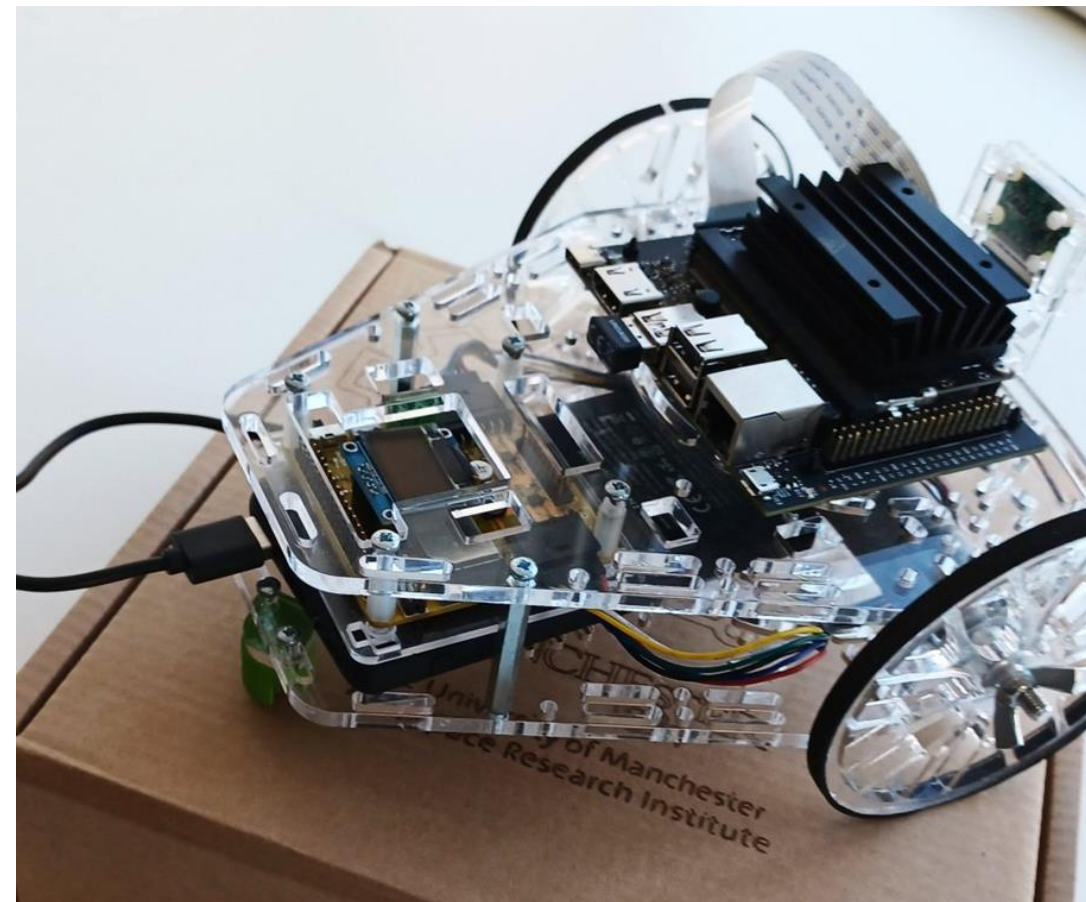


NVIDIA Jetson Nano[®], 2 GB



Overview

- MCR2 provides a modified version of the Jetson Image, with some preloaded packages, that connects with the Hackerboard (plug and play).
- This section will guide the user on how to flash such image.
- The original image can be flashed anytime by following the steps in this section.
- This guide will help you start using the Jetson board on the Puzzlebot.



General characteristics

- 128-core NVIDIA Maxwell GPU
- 1.43 GHz Quad-core ARM A57 CPU
- 2 GB of 64-bit LPDDR4 Memory
- SD card for storage
- Ethernet & Wi-Fi
- CSI-2 Connector for Camera
- Runs a modified version of Ubuntu 18.04, with the following preinstalled software:
 - ROS2
 - OpenCV
 - Nvidia Camera nodes



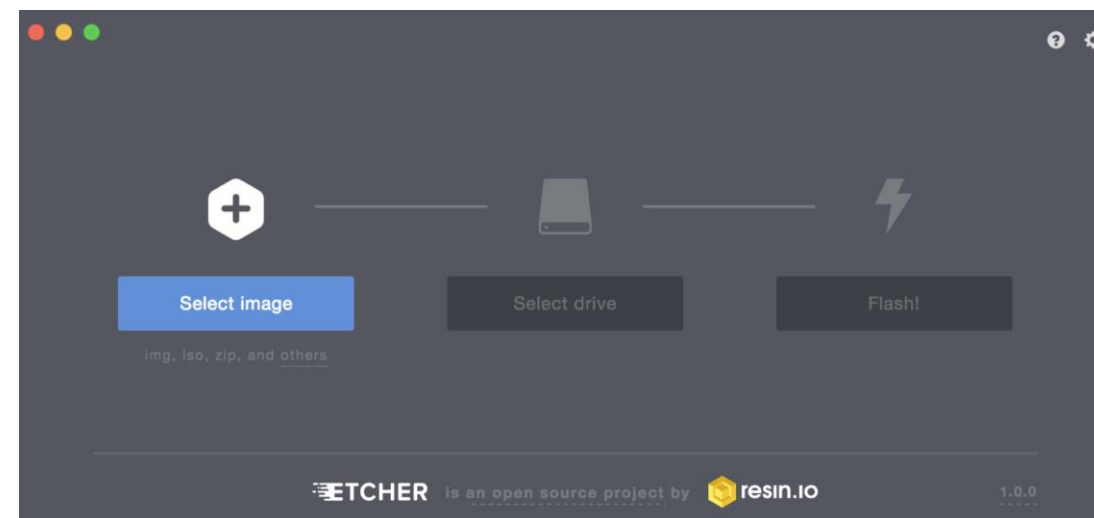


MCR2 Image Installation

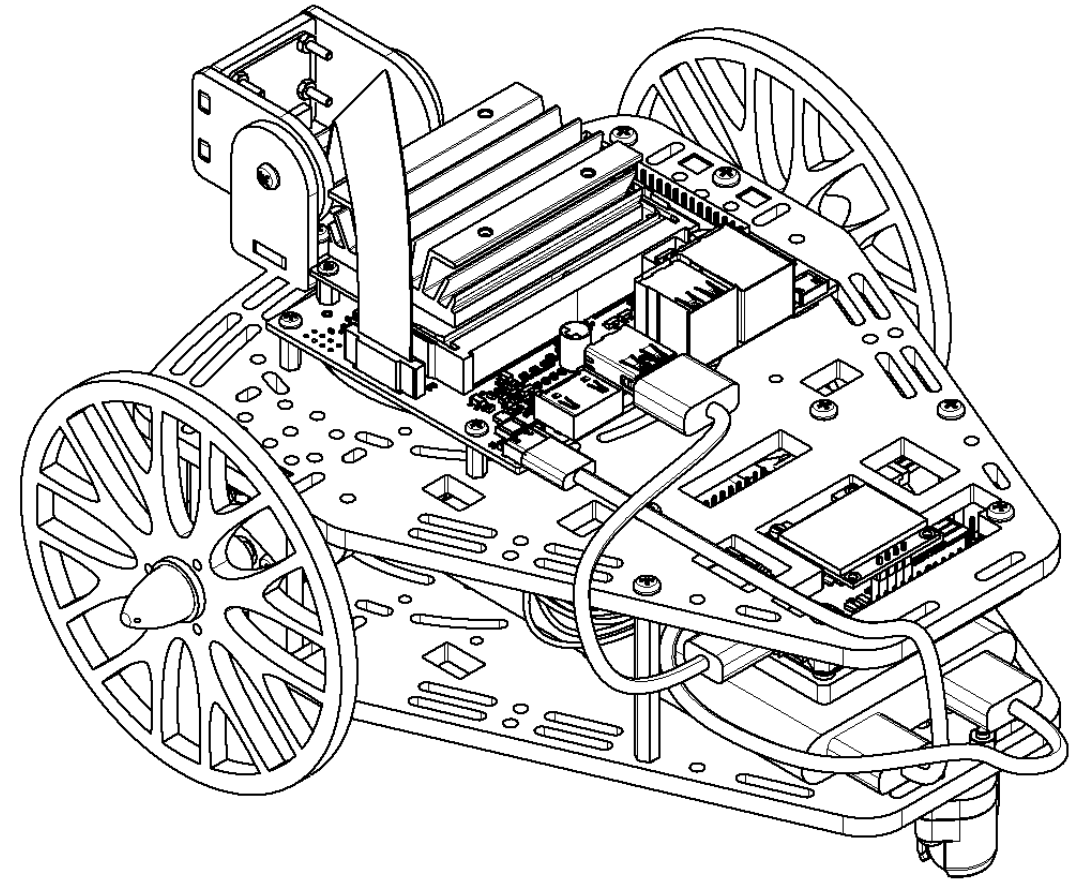
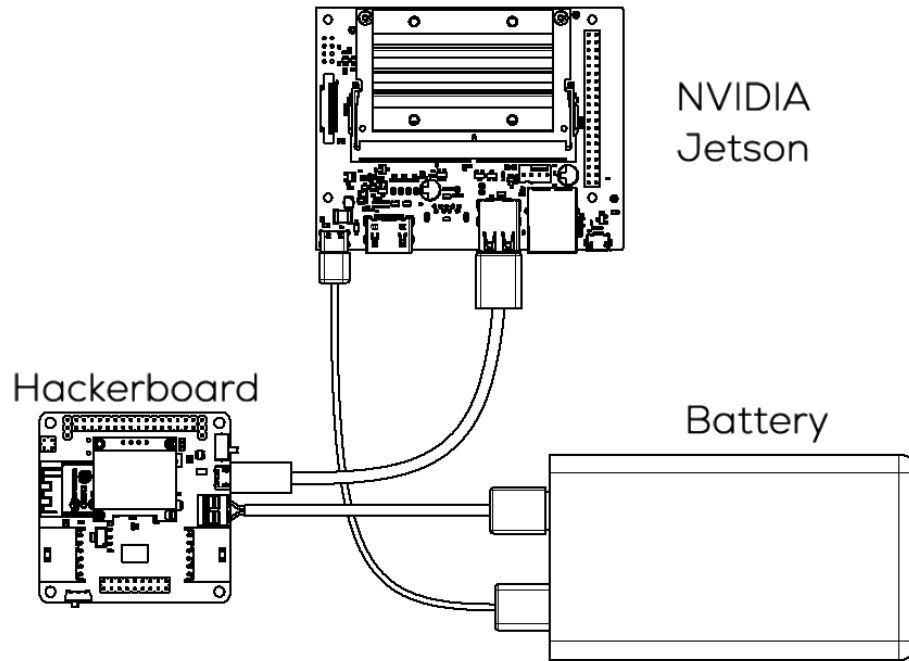


General characteristics

- The OS for the Jetson is stored on an SD card
- An image must be flashed to the SD card; download it from [here](#)
- This is flashed from an image using the Balena Etcher tool
- To flash the SD card:
 1. Insert the SD into your PC
 2. Launch the etcher software
 3. Select the image downloaded from the link above in the “Flash from file” section
 4. Select the SD card in the “Select target” section
 5. Click “Flash” and wait.
 6. Once Flashed insert the SD Card into the Puzzlebot



Electrical Connection Diagram



*The battery must be able to provide 5V, 3A per port independently.



NVIDIA Jetson Nano[®], 2 GB



Initial Setup

1. Connect the Hackerboard to the Jetson.
2. Connect the Jetson Nano to a screen
3. Connect the Jetson to the battery
4. Open a terminal *"LXTerminal"*
5. Type *"ros2 topic list"*, you should see the following:

```
puzzlebot@jetson:~$ ros2 topic list
/parameter_events
/rosout
puzzlebot@jetson:~$
```





NVIDIA Jetson Nano[®], 2 GB



Testing the robot

1. Open a terminal *"LXTerminal"* and type the following

```
ros2 launch puzzlebot_ros micro_ros_agent.launch.py
```

2. Open another terminal and the following topics should be shown (according to the Hackerboard configuration)

```
ros2 topic list
```

3. Publish the following command if the /cmd_vel topic is active. The wheels should rotate.

```
$ ros2 topic pub /cmd_vel geometry_msgs/msg/Twist  
'linear:  
  x: 0.0  
  y: 0.0  
  z: 0.0  
angular:  
  x: 0.0  
  y: 0.0  
  z: 0.7'
```

```
puzzlebot@jetson:~$ ros2 launch puzzlebot_ros micro_ros_agent.launch.py  
[INFO] [launch]: All log files can be found below /home/puzzlebot/.ros/log/2024-01-17-16-51-55-632029-jetson-21307  
[INFO] [launch]: Default logging verbosity is set to INFO  
[INFO] [micro_ros_agent-1]: process started with pid [21313]  
micro_ros_agent-1 [1705510316.120015] info | TermiosAgentLinux.cpp | init | running... | fd: 3  
micro_ros_agent-1 [1705510316.120845] info | Root.cpp | set_verbose_level | verbose level: 4  
micro_ros_agent-1 [1705510316.174538] info | Root.cpp | create_client | client key: 0x0F351064, session id: 0x81  
micro_ros_agent-1 [1705510316.174615] info | SessionManager.hpp | establish_session | client key: 0x0F351064, address: 0  
micro_ros_agent-1 [1705510316.210814] info | ProxyClient.cpp | create_participant | participant created | client key: 0x0F351064, participant_id: 0x000(1)  
micro_ros_agent-1 [1705510316.228220] info | ProxyClient.cpp | create_topic | topic created | client key: 0x0F351064, topic_id: 0x000(2), participant_id: 0x000(1)  
micro_ros_agent-1 [1705510316.240262] info | ProxyClient.cpp | create_subscriber | subscriber created | client key: 0x0F351064, subscriber_id: 0x000(4), participant_id: 0x000(1)  
micro_ros_agent-1 [1705510316.254516] info | ProxyClient.cpp | create_datareader | datareader created | client key: 0x0F351064, datareader_id: 0x000(6), subscriber_id: 0x000(4)  
micro_ros_agent-1 [1705510316.272286] info | ProxyClient.cpp | create_topic | topic created | client key: 0x0F351064, topic_id: 0x001(2), participant_id: 0x000(1)  
micro_ros_agent-1 [1705510316.283271] info | ProxyClient.cpp | create_publisher | publisher created | client key: 0x0F351064, publisher_id: 0x000(3), participant_id: 0x000(1)  
micro_ros_agent-1 [1705510316.296126] info | ProxyClient.cpp | create_datawriter | datawriter created | client key: 0x0F351064, datawriter_id: 0x000(5), publisher_id: 0x000(3)  
micro_ros_agent-1 [1705510316.312307] info | ProxyClient.cpp | create_topic | topic created | client key: 0x0F351064, topic_id: 0x002(2), participant_id: 0x000(1)  
micro_ros_agent-1 [1705510316.323256] info | ProxyClient.cpp | create_publisher | publisher created | client key: 0x0F351064, publisher_id: 0x001(3), participant_id: 0x000(1)  
micro_ros_agent-1 [1705510316.336056] info | ProxyClient.cpp | create_datawriter | datawriter created | client key: 0x0F351064, datawriter_id: 0x001(5), publisher_id: 0x001(3)  
micro_ros_agent-1 [1705510316.352647] info | ProxyClient.cpp | create_topic | topic created | client key: 0x0F351064, topic_id: 0x003(2), participant_id: 0x000(1)  
micro_ros_agent-1 [1705510316.363668] info | ProxyClient.cpp | create_publisher | publisher created | client key: 0x0F351064, publisher_id: 0x002(3), participant_id: 0x000(1)  
micro_ros_agent-1 [1705510316.376270] info | ProxyClient.cpp | create_datawriter | datawriter created | client key: 0x0F351064, datawriter_id: 0x002(5), publisher_id: 0x002(3)
```

```
puzzlebot@jetson:~$ ros2 topic list  
/VelocityEncl  
/VelocityEncR  
/cmd_vel  
/parameter_events  
/robot_vel  
/rosout
```

*Make sure the Hackerboard is turned On, and the Motor Switch is turned On.



Raspberry Pi Camera



- NVIDIA provides a package for interfacing with a CSI camera.
- To access the camera and test the connection type on a terminal

```
nvgstcapture-1.0
```

- To interface the camera with ROS, NVIDIA provides a ROS package that comes pre-installed on the Puzzlebot image
- For this package, several launch files are available. Only 2 are of interest to us:
 - `ros_deep_learning video_viewer.ros2.launch`
 - `ros_deep_learning video_source.ros2.launch`

- On your Jetson, run the command:

```
ros2 launch ros_deep_learning video_viewer.ros2.launch
```

- The camera view should be displayed on the screen.
- To only publish the image to a topic without any viewer, run the following command and then run “`ros2 topic list`” command on another terminal.

```
ros2 launch ros_deep_learning video_source.ros2.launch
```



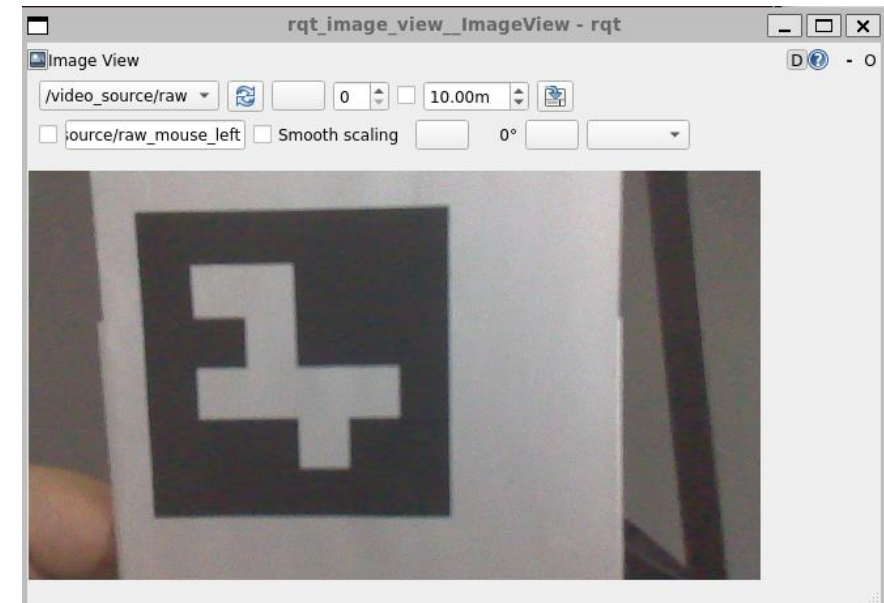
Raspberry Pi Camera



- MCR2 provides a launch file that contains the previous node and also publishes the camera information, required for different algorithms such as localisation.

```
$ ros2 launch puzzlebot_ros camera_jetson.launch.py
```

```
mario@MarioPC:~$ ros2 topic list  
/VelocityEncL  
/VelocityEncR  
/camera_info  
/cmd_vel  
/parameter_events  
/robot_vel  
/rosout  
/video_source/raw
```



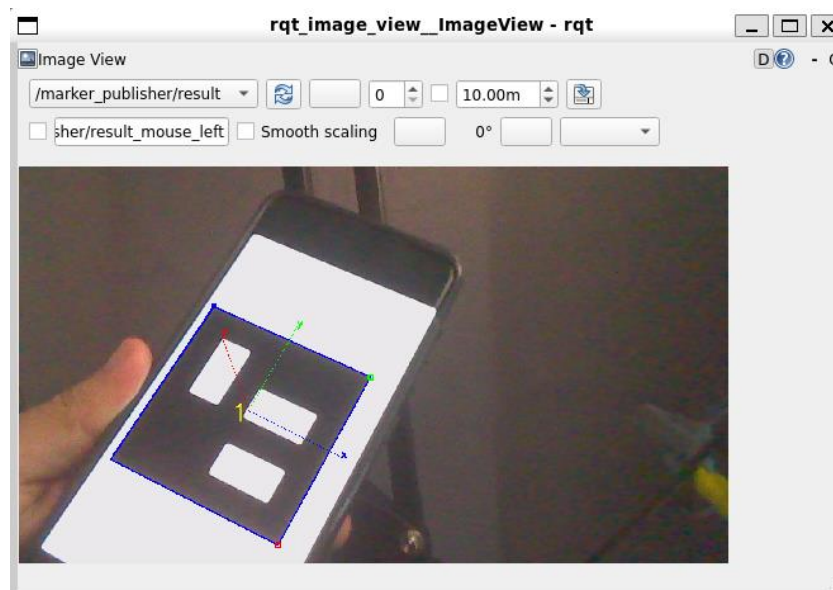
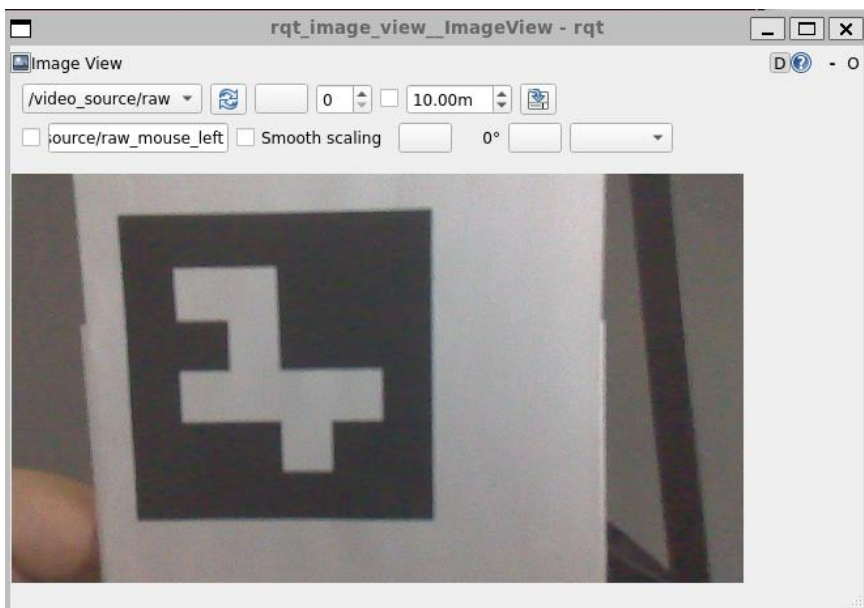


Raspberry Pi Camera



- MCR2 also provides April Tag marker detection

```
$ ros2 launch puzzlebot_ros aruco_jetson.launch.py
```



```
mario@MarioPC:~$ ros2 topic list
/camera_info
/marker_publisher/debug
/marker_publisher/markers
/marker_publisher/markers_list
/marker_publisher/result
/parameter_events
/rosout
/tf
/tf_static
/video_source/raw
```

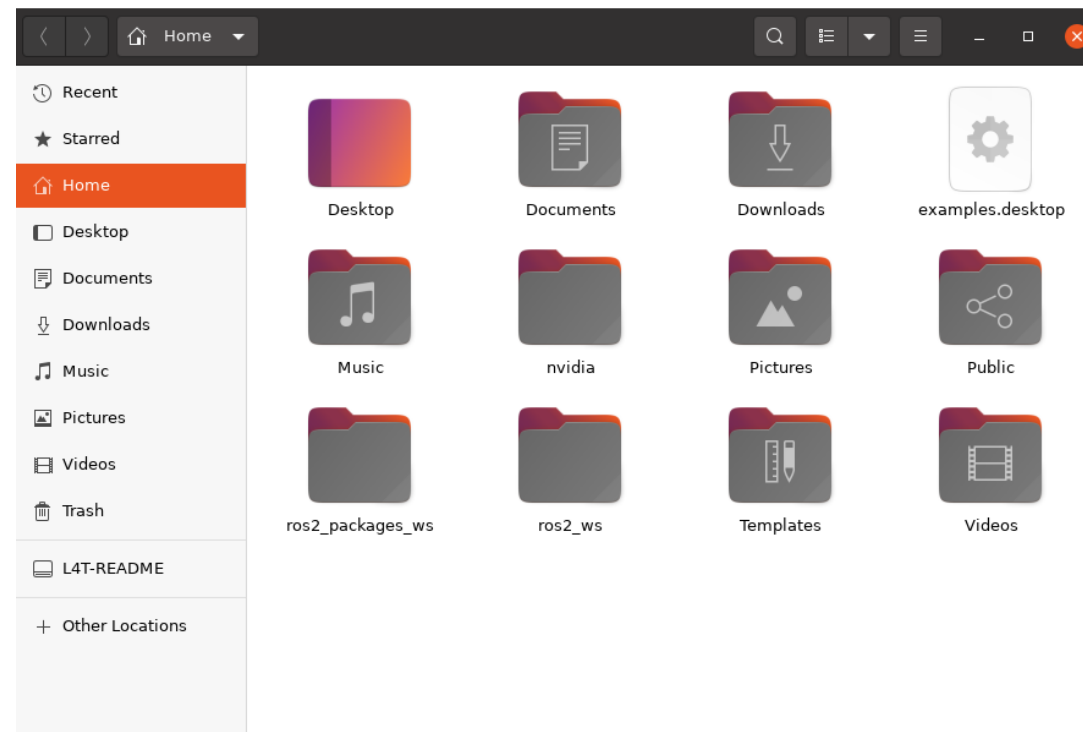


MCR2 Jetson Image



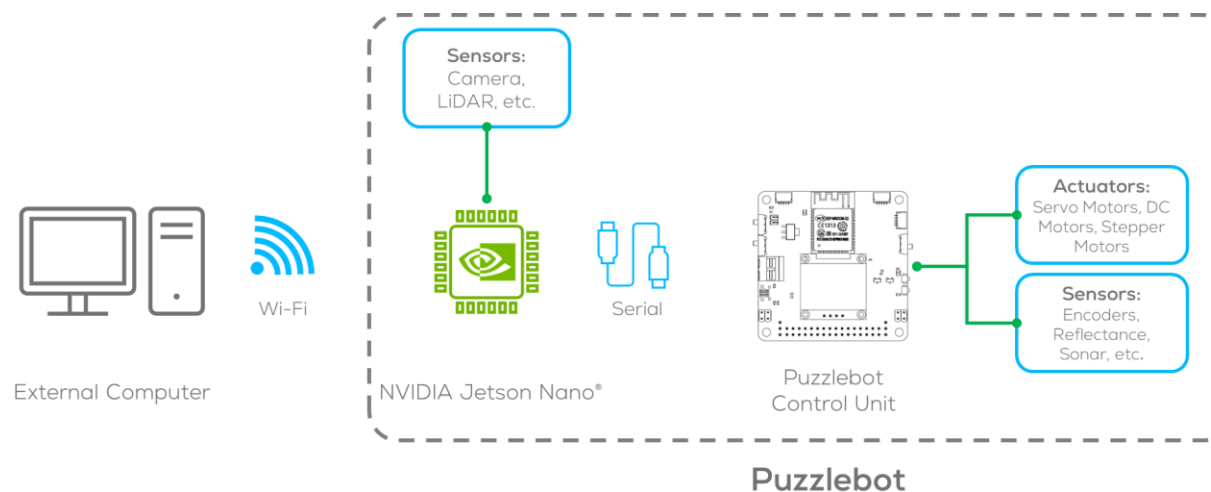
General Information

- ROS is always running on Jetson nano.
- No nodes are running at boot, the user must run its own nodes or the ones pre-loaded by MCR2.
- There is a pre-setup workspace with some nodes on the Jetson called `ros2_ws` where the users can deploy their nodes.
- The “*ros2_packages_ws*” are some MCR2 nodes and libraries necessary for the Puzzlebot and camera communication and should not be changed in any way.



Remote Access

1. It can be useful to control the Jetson from a remote PC, as the robot cannot be in motion and hooked up to a monitor
 2. To do this, two ways of communication are available SSH or ROS Network. Both uses Wi-Fi to give your computer access to the Jetson.
 3. For both connection in this tutorial, the Jetson needs to be configured as a Hotspot to generate its own access point.
- To prevent conflicts when many Jetsons are used, change the Network Name to something unique (if applicable)



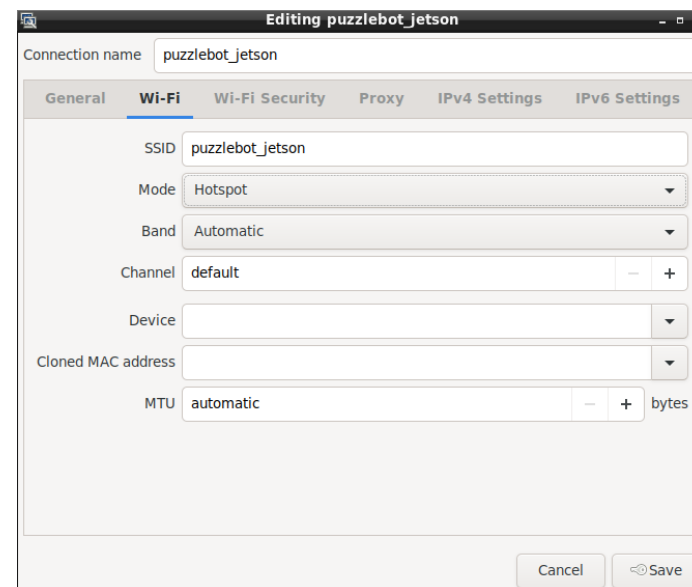
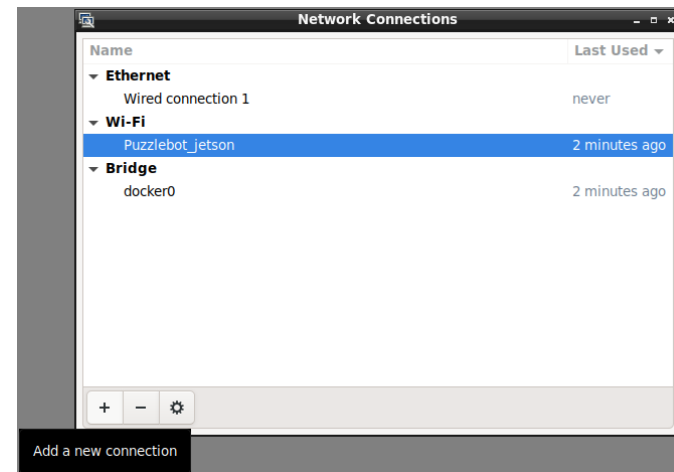
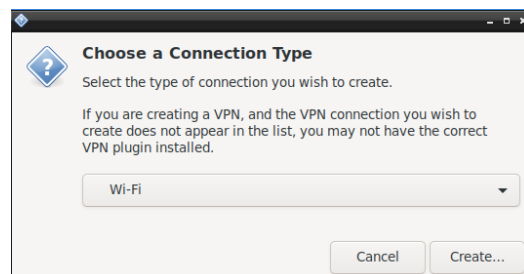
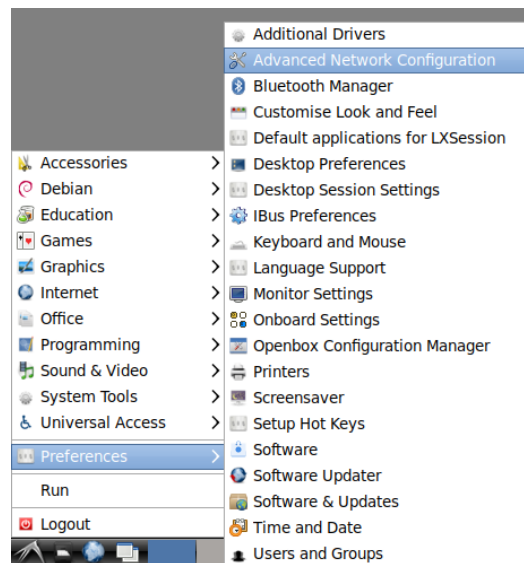


Making a Hotspot



Hotspot Setup

1. Set up a hotspot by going to the Start Menu>> Preferences >> Advanced Network Configuration
2. In the pop-up window click on the “+” signal to add a new connection.
3. Select on the pop-up window to Wi-Fi
4. Configure the name of the connection (you can use any name).
5. Configure the SSID (the name that you will see in other computers)
6. Change the mode to “Hotspot”



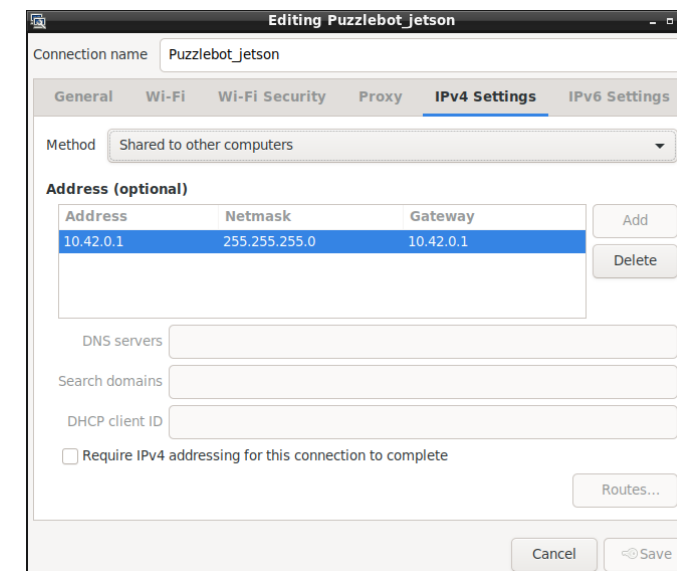
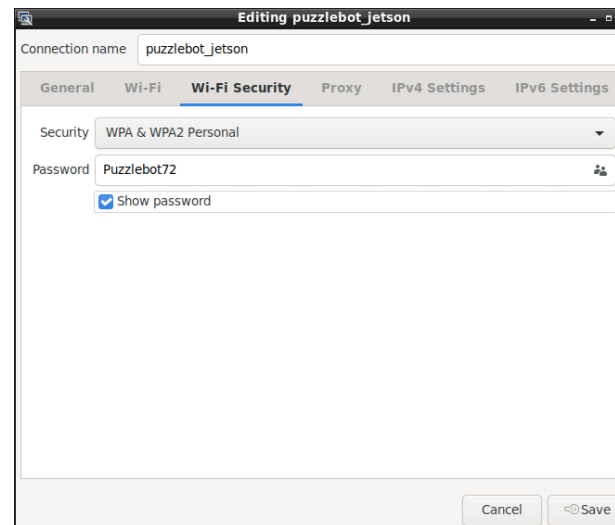


Making a Hotspot



Hotspot Setup

7. In the “Wi-Fi” security tab establish a password for your own robot.
8. On the tab IPv4 setting configure an static IP address to connect to the robot.
 - In the example the addresses selected are the following
 - Address: 10.42.0.1
 - Netmask: 255.255.255.0
 - Gateway: 10.42.0.1
9. Save the configuration by clicking the “Save” button.





Remote Access



1. On the external PC (Windows or Ubuntu), open a Terminal window and type:
 - `ssh puzzlebot@10.42.0.1` (Configured IP Address)
 - Password: `Puzzlebot72` (Configured Password)
2. If prompted, type **yes** and then press enter.
3. This command window is now equivalent to one running on the Jetson.
4. Once any control code is written on the Jetson, it can be tested and debugged remotely via SSH, enabling the PuzzleBot to move around

```
PS C:\Users\mario> ssh puzzlebot@10.17.0.1
puzzlebot@10.17.0.1's password:
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 4.9.337-tegra aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

57 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

Last login: Wed Jan 17 16:54:17 2024 from 10.17.0.59
puzzlebot@jetson:~$
```



Testing



- Test the ROS communication, run the following command (if not already running on the Jetson)

```
ros2 launch puzzlebot_ros micro_ros_agent.launch.py
```

- Echo the different topics using `ros2 topic echo`
 - Echo the topics `/VelocityEncR`, and `/VelocityEncR`, and rotate the wheels
 - The speed of the wheels should be displayed
- Publish to the command topics, the wheels should turn
 - If control mode 1 is used, publish to `/cmd_vel`
 - If control mode 2 is used, publish to `/VelocitySetR` and `/VelocitySetL`
 - If control mode 3 is used, publish to `/ControlL` and `/ControlR`
 - The control mode is changed on the Hackerboard webpage

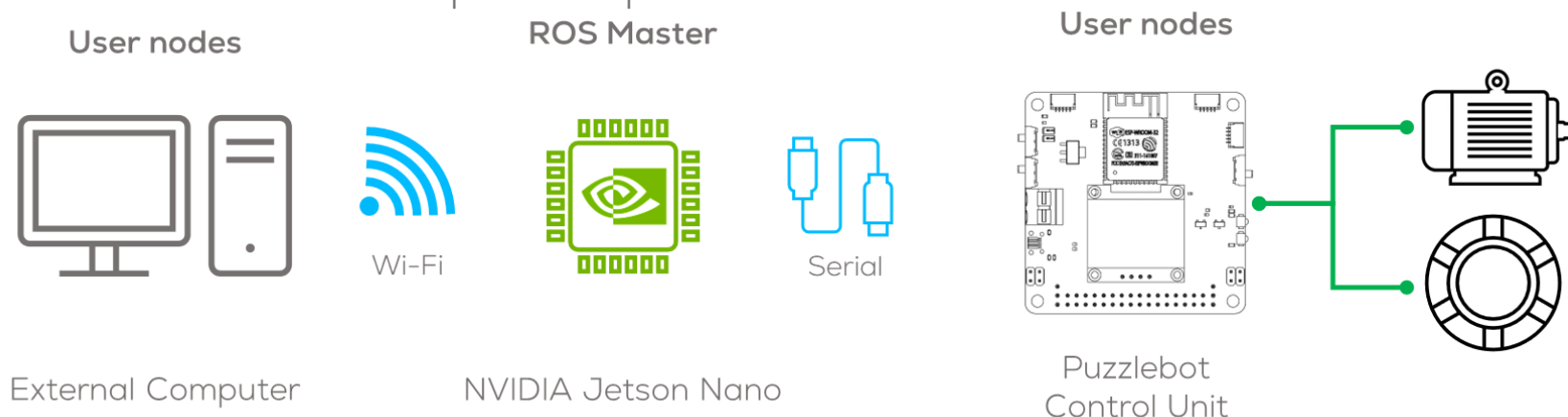


Multi-device Communication with ROS (ROS network)



Introduction

- ROS facilitates communication between various devices, with ROS2 installed, on a shared network (No SSH).
- The Jetson Nano acts as an Access Point (AP) to provide this network.
- In the context of the Puzzlebot, by connecting to the Jetson's AP, other devices can communicate via ROS topics and publishers.





Multi-device Communication with ROS (ROS Network)



- From the host computer in Ubuntu with ROS2 Installed, connect to the “Hotspot” of the Jetson.
 - Make sure the IP address is correct using the command “ifconfig” on the terminal.
 - ROS Should be able to detect the nodes, topics and subscribers.
-
- This connection is **not** a remote access into the Jetson like SSH, we cannot start nodes or run other commands
 - However, it can be more useful, as we cannot easily display visualisations via SSH.

```
mario@MarioPC:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.17.0.31 netmask 255.255.255.0 broadcast 10.17.0.255
    ether 5e:bb:f6:9e:ee:fa txqueuelen 1000 (Ethernet)
    RX packets 546 bytes 77617 (77.6 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 441 bytes 99820 (99.8 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 218 bytes 55509 (55.5 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 218 bytes 55509 (55.5 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

mario@MarioPC:~$ ros2 topic list
/VeLocityEnCL
/VeLocityEncR
/cmd_vel
/parameter_events
/robot_vel
/rosout
```



Troubleshooting



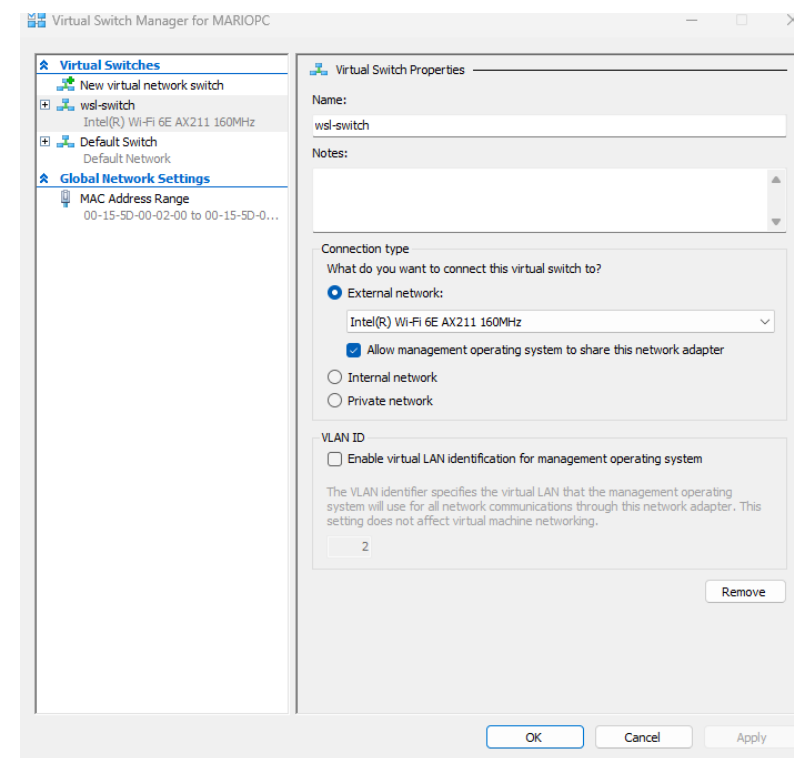
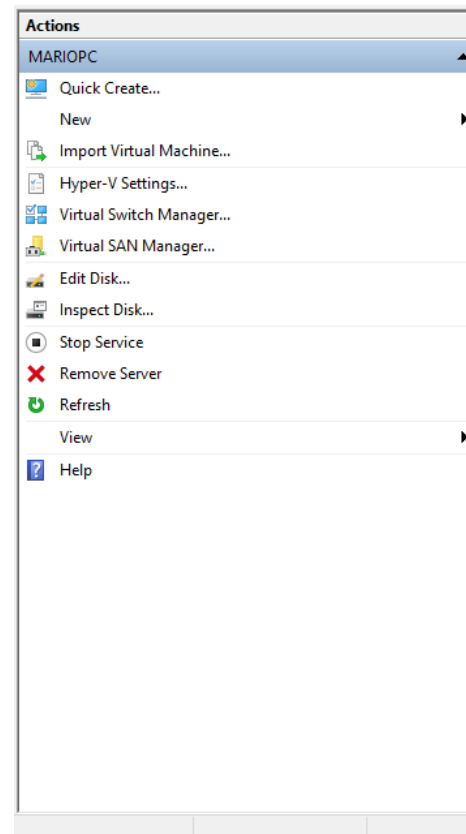
- The Hacker Board communication will only work correctly if the Jetson is connected to its own Wi-Fi network (AP)
 - Go to Network -> Connect to Hidden Wi-Fi network and select "Hotspot" from the dropdown.
 - The PuzzlebotJetson Network should now be visible in the networks; connect to it, to activate the hotspot.
- Sometimes, the Jetson Wi-Fi Network takes a couple of minutes to appear on other devices after a reboot. If it does not, make sure no other connections are saved. It is only guaranteed to connect to PuzzlebotJetson on boot if there are no other connections saved
- Always consider checking the Hackerboard if these don't work: is ROS turned on?



WSL2 IP Configuration



- WSL requires the user to configure the IP address to mirror the one in the host computer and connect to the Puzzlebot.
- Open Hyper-V manager and press “Virtual Switch Manager”
- Press the button “Create a Virtual Switch”
- Give a name to the switch “wsl-switch” and select the WiFi adapter to use (in case you have different) to create the Switch.
- Press “Apply” and “OK”





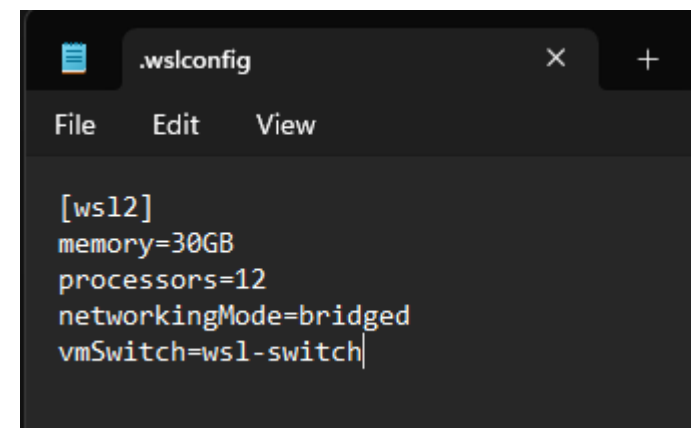
WSL2 IP Configuration



- Create a new file called “.wslconfig” on the “User” folder e.g., “C:\Users\mario”
- You can use a terminal to create the file

```
PS C:\Users\mario> notepad .wslconfig
```

- Add the following inside the file, save it and close it.
- Start the WSL2,
 - If you have previously started it shutdown the WSL as follows: `wsl -- shutdown`
 - Then start the WSL.
- The IP address should be one given by the switch.
- Now you can connect to ROS Network “`ros2 topic list`”



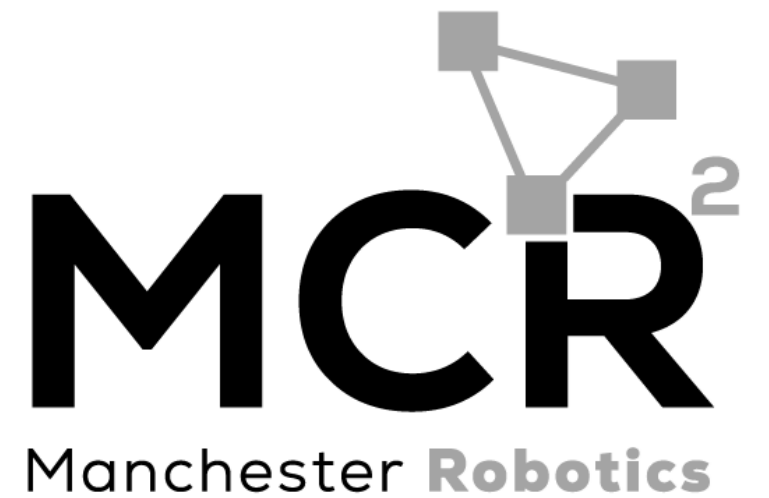
```
[wsl2]
memory=30GB
processors=12
networkingMode=bridged
vmSwitch=wsl-switch
```

```
mario@MarioPC:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.17.0.31 netmask 255.255.255.0 broadcast 10.17.0.255
    ether 5e:bb:f6:9e:ee:fa txqueuelen 1000 (Ethernet)
    RX packets 1364 bytes 89865 (89.8 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 493 bytes 37544 (37.5 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 7 bytes 795 (795.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 7 bytes 795 (795.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```


Thank you

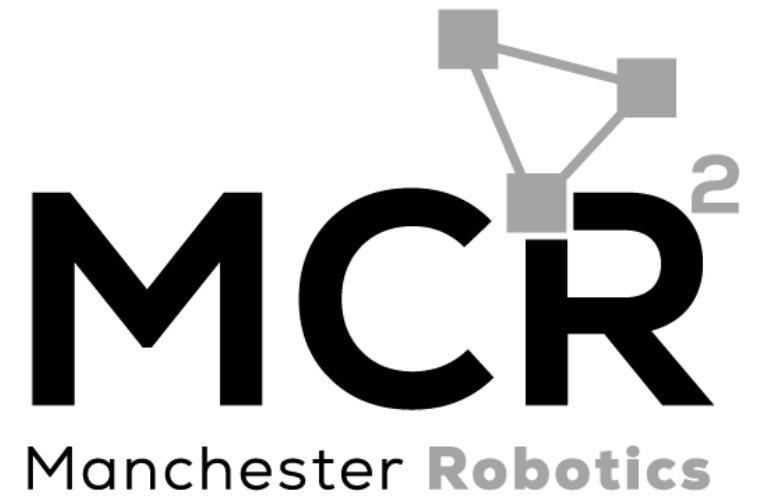
{Learn, Create, Innovate};



T&C

Terms and conditions

{Learn, Create, Innovate};





Terms and conditions



- *THE PIECES, IMAGES, VIDEOS, DOCUMENTATION, ETC. SHOWN HERE ARE FOR INFORMATIVE PURPOSES ONLY. THE DESIGN IS PROPRIETARY AND CONFIDENTIAL TO MANCHESTER ROBOTICS LTD. (MCR2). THE INFORMATION, CODE, SIMULATORS, DRAWINGS, VIDEOS PRESENTATIONS ETC. CONTAINED IN THIS PRESENTATION IS THE SOLE PROPERTY OF MANCHESTER ROBOTICS LTD. ANY REPRODUCTION, RESELL, REDISTRIBUTION OR USAGE IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF MANCHESTER ROBOTICS LTD. IS STRICTLY PROHIBITED.*
- *THIS PRESENTATION MAY CONTAIN LINKS TO OTHER WEBSITES OR CONTENT BELONGING TO OR ORIGINATING FROM THIRD PARTIES OR LINKS TO WEBSITES AND FEATURES IN BANNERS OR OTHER ADVERTISING. SUCH EXTERNAL LINKS ARE NOT INVESTIGATED, MONITORED, OR CHECKED FOR ACCURACY, ADEQUACY, VALIDITY, RELIABILITY, AVAILABILITY OR COMPLETENESS BY US.*
- *WE DO NOT WARRANT, ENDORSE, GUARANTEE, OR ASSUME RESPONSIBILITY FOR THE ACCURACY OR RELIABILITY OF ANY INFORMATION OFFERED BY THIRD-PARTY WEBSITES LINKED THROUGH THE SITE OR ANY WEBSITE OR FEATURE LINKED IN ANY BANNER OR OTHER ADVERTISING.*