Jetson Ethernet Setup and Turtlesim Nodes

Table of Contents

Introduction 2

Objective 2

High-Level Procedure 2

Required Equipment Checklist 2

Networking 3

Orin Access Options 3

Ethernet Setup Guide 4

TurtleBot Simulator 7

Launching the Simulator 7

Controlling the TurtleBot via Keyboard Teleoperation 7

Launch Files 9

Installation and Running at the Package Level 9

Quick Testing (Without Installation) 9

Assignment 11

Submission 12

Troubleshooting 12

Introduction

Objective

This lab has two primary goals:

1. Setup Remote Access: Configure your Jetson Orin Nano to allow reliable remote desktop access via an Ethernet connection using a VNC application
2. Control a TurtleBot Simulation: Learn ROS2 basics by writing and running code to control a simulated robot using the *turtlesim* package.

Upon completion, you will be able to access your Nano's desktop from your laptop and program it to perform tasks in a small simulation environment.

High-Level Procedure

The lab is divided into the following phases:

Phase 1: Hardware and Network Setup

1. Establish a Direct Connection: Connect your laptop directly to the Orin Nano using an Ethernet cable.
2. Enable Remote Desktop: Install and configure a VNC server on the Nano to view and control its desktop GUI from your laptop.

Phase 2: Software Environment Configuration

3. Connect to Campus IoT Network: Connect the Nano to the *CP-IoT-Secure* Wi-Fi network for internet access.
4. Install Required Software: Install the necessary ROS2 *turtlesim* packages on the Nano if they are not already present.

Phase 3: Application Development

5. Develop and Demo: Write, compile, and demonstrate a ROS2 node that commands the simulated TurtleBot to draw a square.

Required Equipment Checklist

- Ensure you have the following items before starting:
- Ethernet Cable (to connect your laptop directly to the Nano)
- Jetson Orin Nano Developer Kit
- Jetson Orin Nano Power Adapter
- Personal Laptop

Networking

Orin Access Options

This quarter, we will use one of the following methods to access and configure the Jetson Orin Nano (JON) devices:

1. Wired Connection (Recommended Initial Steup)
    - Connect an Ethernet cable directly from the JON to your computer to establish an *SSH* or *VNC* connection.

2. Wireless Setup (Required for Physical Robot Testing and Debugging)
    - Establish an *SSH* connection to the JON while it is on *CP-IoT-Secure* network and your laptop is connected to the *eduroam* network.

3. Direct Peripheral Connection (Use Sparingly)
    - Connect a monitor, keyboard, etc. directly to the JON.
    - **Important:** do *not* unplug or re-purpose any existing lab equipment

Lab Focus: Establishing an Ethernet Connection

For this lab assignment, we will focus on Method 1: Wired Connection. You will learn to establish a stable connection to the JON via your laptop's Ethernet port as the foundational step for subsequent work.

Ethernet Setup Guide

Before you proceed with this section, make sure that your host computer has an ethernet port.

Notes (how to access network configurations based on your host OS):

- macOS (Sequoia 15): System Settings → Network → Ethernet → Details → TCP/IP → Manually
- Windows: Control Panel → Network and Sharing Center → Change adapter settings

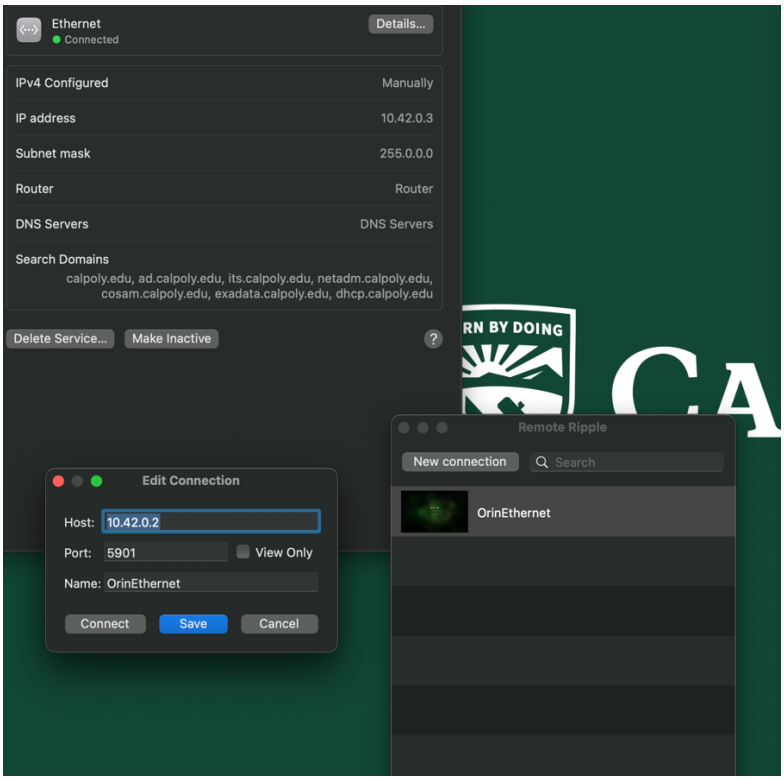
1. Configure your Ethernet port with the static IP address. You will need access the network settings on your computer.
    - IP: 10.42.0.3
    - Subnet Mask: 255.0.0.0

2. Physical Connection
    - Plug in an Ethernet cable between your computer and the Orin nano. A cable should be provided with your kits.

3. VNC Viewer Setup
    - Install a VNC client (e.g. Remote Ripple, TigerVNC)

4. Establish Connection
    - In your VNC Client, connect to – 10.42.0.2:5901 (Orin's established IP address and the Port to use)
    - This should provide a dedicated Desktop View (not shared) of the Orin's interface

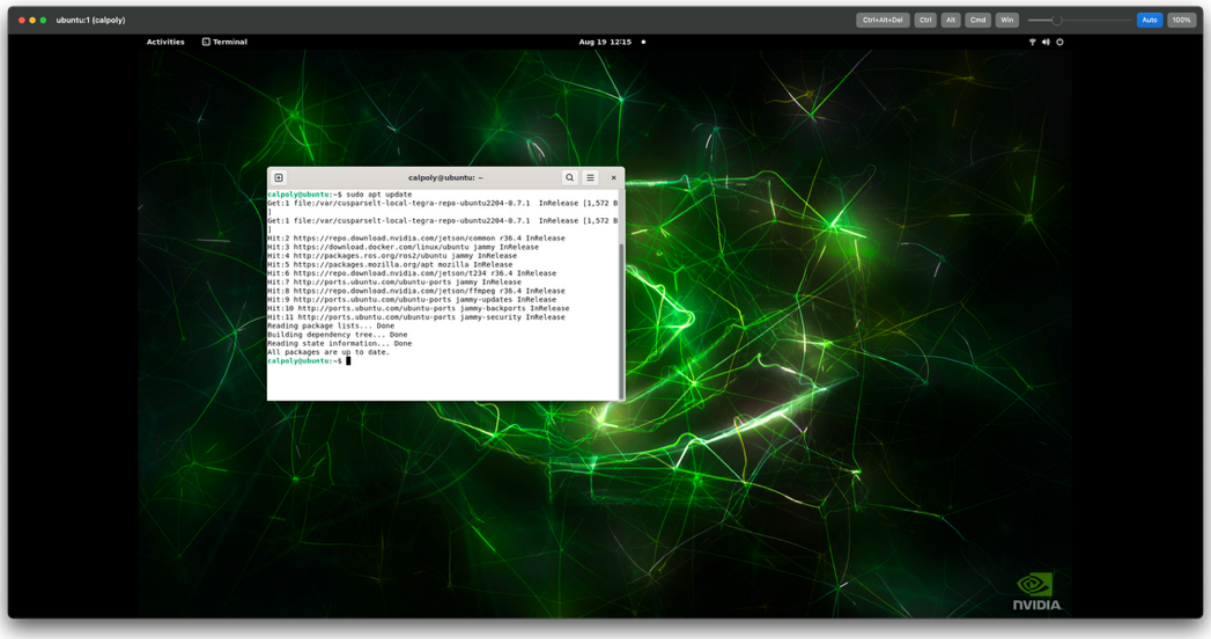
Below is a screenshot of the process using MacOS and Remote Ripple. Note some of the Orins may already have an established connection to *CP-IoT-Secure* and should already have the VNC *server* side setup on the Orins.



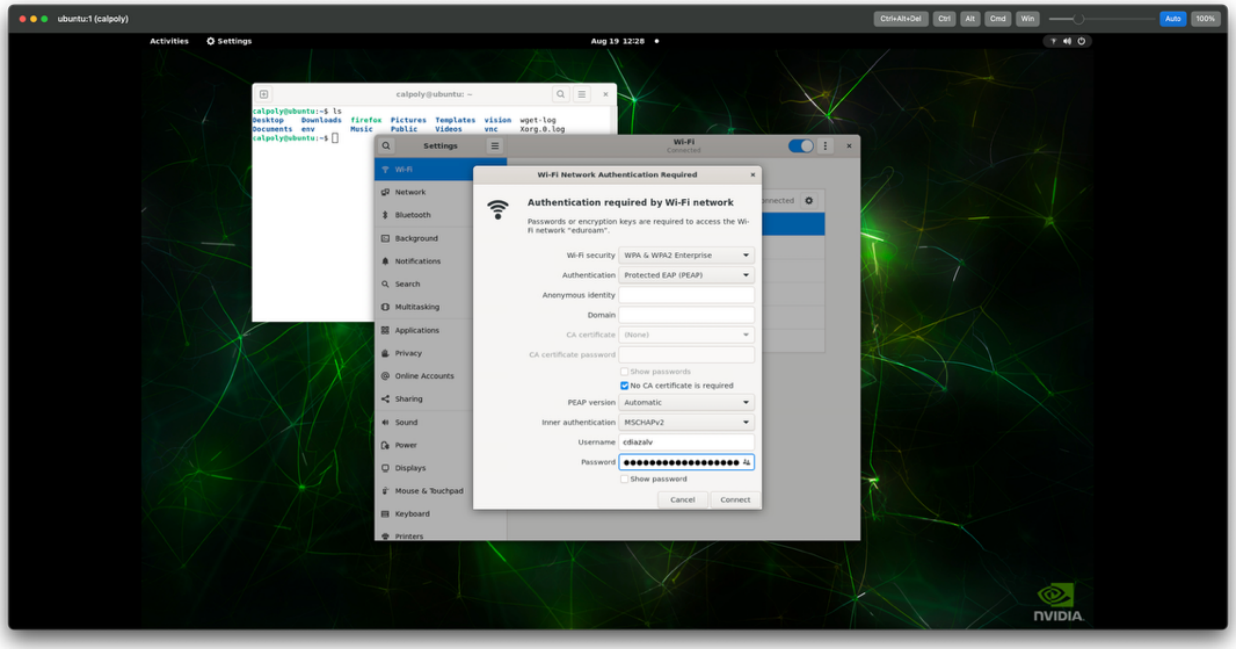
Issues with the "Authenticate the User" repeatedly popping up. If you experience this issue, you should create a pkla file (under the */etc/polkit-1/localauthority/50-local.d/* directory) and populate it with the following info:

```
[Allow Wifi Scan]
Identity=unix-user:*
Action=org.freedesktop.NetworkManager.wifi.scan;org.freedesktop.NetworkManage
r.enable-disable-
wifi;org.freedesktop.NetworkManager.settings.modify.own;org.freedesktop.Netwo
rkManager.settings.modify.system;org.freedesktop.NetworkManager.network-
control
ResultAny=yes
ResultInactive=yes
ResultActive=yes
```

You can name the file *45-allow-wifi.pkla.* Note that you will need to be superuser (root) permission to perform this task.



Above is a screenshot of the Desktop view from Remote Ripple. Furthermore, below we see how to connect the Orin Nano to the *CP-IoT-Secure* network so that we can access the Orin wirelessly later as well.



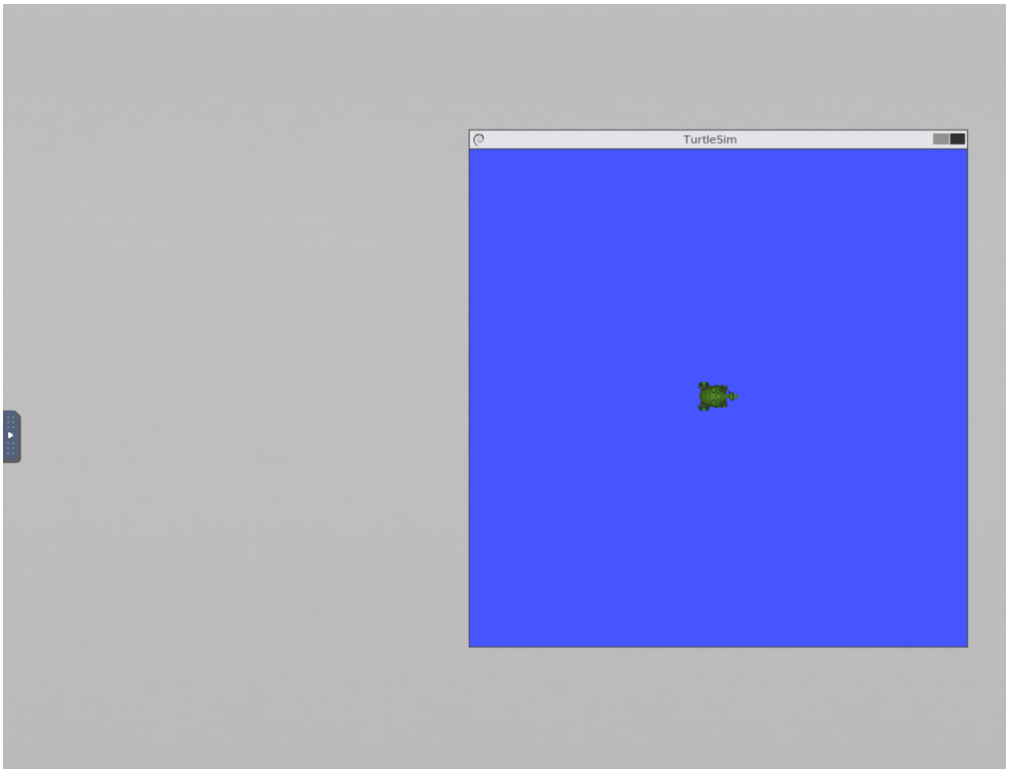
TurtleBot Simulator

This guide walks you through launching the TurtleBot simulator and controlling the robot using keyboard teleoperation.

Launching the Simulator

The turtlesim package provides a simple 2D simulation environment. To start the simulator, run the following command in your terminal:

```
ros2 run turtlesim turtlesim_node
```



Expected Result:
A new graphical window will appear, similar to the screenshot below. This window contains the TurtleBot (represented by a turtle icon) in its world. (For reference look at the above screenshot)

Controlling the TurtleBot via Keyboard Teleoperation

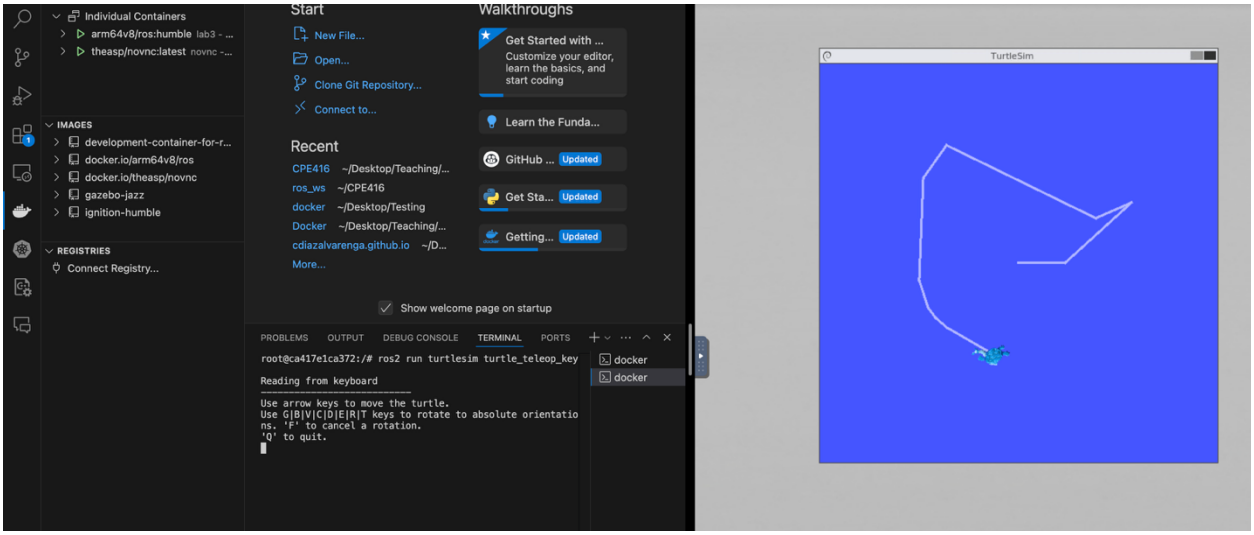
To control the TurtleBot, you need to start the teleoperation node in a separate terminal. It is crucial that this new terminal has its ROS 2 environment sourced (e.g., by running *source /opt/ros/humble/setup.bash*).

In the new terminal, run the following command:

```
ros2 run turtlesim turtle_teleop_key
```

Expected Result:
The terminal will display control instructions and wait for your keyboard input. The TurtleBot will now draw a visible path as it moves.

This will allow you to control the TurtleBot using your keyboard! You should see it drawing a line underneath itself as it moves!



Important:
For the teleoperation to work, the terminal window where you ran the *turtle_teleop_key* command must be the active (focused) window. Your keyboard input will not be registered if the simulator window or another application is focused.

Control Keys:

- Arrow Keys: Control the TurtleBot's linear and angular velocity.
    - Up/Down: Move forward/backward
    - Left/Right: Rotate counterclockwise/clockwise
- Q: Quit the teleoperation node.

Visualizing the Result:
As you drive the TurtleBot, it will leave a colored trace behind it, visualizing its path. To see the robot move and the path it draws, you must keep both the simulator window and the teleoperation terminal visible on your screen. This setup provides a foundational understanding of ROS2 node interaction, where two separate nodes (the simulator and the teleop controller) communicate via topics to produce the system's behavior.

Launch Files

ROS2 Python launch files are powerful tools for managing complex applications. They allow you to start and stop multiple nodes, react to events, and configure your ROS environment—all from a single command. This eliminates the need for numerous terminal windows and simplifies deployment.

Key Benefits:

- Multi-Node Management: Start several nodes simultaneously.
- Event Handling: Trigger actions based on node lifecycle events.
- Environment Configuration: Set parameters and environment variables easily.

Installation and Running at the Package Level

For your launch file to be discoverable by ROS2 at the package level, you need to properly install it:

1. Create the Launch Directory: Ensure your package has a *launch* directory.

2. Update setup.py: In your package's setup.py, add the launch directory to the data files so it gets installed:

```
import os
from glob import glob
from setuptools import setup

setup(
    # ... your other package configuration ...
    data_files=[
        # ... other data files ...
        (os.path.join('share', package_name), glob('launch/*.py')),
    ],
)
```

3. Build Your Package: Rebuild and source your workspace (*colcon build && source install/setup.bash*).

4. Run from Anywhere: Once installed, you can run the launch file from any directory using:

```
ros2 launch your_package your_launchfile.py
```

Quick Testing (Without Installation)

For quick testing during development, you can run the launch file directly from its directory:

```
cd src/your_package/launch
ros2 launch your_launchfile.py
```

For a comprehensive guide, please refer to the official ROS2 launch documentation. Note we are writing the launch file in *Python*.

Assignment

1. Write a node that commands the TurtleBot to trace a 1x1 meter square path when started. Name the executable *draw_square*. (10 points)
2. Write a node that commands the TurtleBot to trace a spiral path. Name the executable *draw_spiral*. (10 points)
3. Write a node that drives the TurtleBot to the world coordinates (1, 1) using an open-loop control strategy. The robot must end with its heading pointing left (180 degrees or -π radians). You can name the executable *open_loop.* (10 points)
4. Write a Python launch file that starts both the *turtlesim_node* and your *draw_square* node simultaneously. (10 points)

You can find the starter code in the Github repo under the following path: *references/code_examples/turtlesim/*

To fully understand the basic structure of a ROS2 node that interacts with the *turtlesim* simulator and extend the starter code, please review the following core concepts discussed in class:

- Event-Based Execution (Subscription): How nodes react to incoming messages (e.g., responding to sensor data).
- Timing-Based Execution (Publisher): How nodes perform actions at regular intervals (e.g., sending commands periodically).
- Robot Reference Frames: Understand the coordinate system (*x, y, theta*) used by the turtle in its world.
- Velocity Commands: How linear and angular velocity messages (*geometry_msgs/Twist*) control the robot's movement.

Analyzing the example code with these concepts in mind will clarify how the different components work together to create robot behaviors.

Submission

Code Demos. You are expected to demo your nodes and code in the next lab section. Make sure that I get your name down after your successful demo.

After you've demo'ed your code and passed, I will need you to submit your source code.

1. Delete the: *build*, *log*, and *install* folders in your ros workspace. (I do not need to check your compilation outputs.)
2. Compress your ros workspace and upload the compressed file to Canvas. For example, a zip file would work. This compressed file should contain your source code.

Troubleshooting

- "Command not found" error: Ensure your ROS2 environment is properly *sourced* in the terminal.
- Keyboard input does nothing: Check that the terminal running *turtle_teleop_key* is the focused window.
- No turtle appears: Verify that the *turtlesim_node* started successfully without errors.