SPACE CONCORDIA - ROBOTICS DIVISION

Intro Task Guide

Your Starter Checklist:

To help you get up and running smoothly, here's a list of foundational topics to explore and become comfortable with:

- **Git & GitHub Basics**
  Learn essential Git commands:
  `git fetch`, `git pull`, `git commit`, `git push`, `git switch`, `git log`, etc.
- **Linux Command Lines**
  Get familiar with:
  - `cd`, `ls`, `mkdir`, `code .`
  - How to source your ROS workspace in `.bashrc`
  - Creating and managing ROS 2 workspaces and packages
- **ROS 2 Fundamentals**
  Understand the core structure and purpose of ROS 2. Learn how it enables distributed robot systems.
- **Nodes & Topics**
  These are the building blocks of ROS-based applications. Get to know how they work and interact.
- **The Rover**
  Finally, start learning about the rover itself—its architecture, components, and how you'll interface with it through code and testing. Hands-on time is highly encouraged!

Getting Started

There are many sources you can use in order to get the ball rolling on your learning path. To get you started, here are a couple of documents/videos you could look at to familiarize yourself with ROS2:

Videos:

This is a great starter course on everything there is to know about ROS2 basics [Concordia has a partnership with udemy therefore, you can watch all of this for free!])

*Note: If you decide to follow the udemy course above, make sure to download **ROS2 HUMBLE** on **UBUNTU 22.04** and not ROS2 Jazzy on Ubuntu 24.04! That is the only change you need to make.

- Another good udemy course on getting started with ROS2
- Video on how to download UTM [if on mac]

Documentation:

ROS2 Cookbook (A General Guide)

ROS2 Humble Concept Documention

ROS2 Humble Ubuntu Installation (ROS2 humble download documentation)

Ubuntu 22.04 download, select the **desktop image**

Oracle Virtualbox (Good for virtualizing linux on your machine)

Github Command Cheat Sheet (Good Refresher)

Your First Task: GitHub + ROS2 Packages

To help you get hands-on and reinforce the basics, we'd like you to complete this introductory task. It will give you practice with ROS2, GitHub, and both C++ and Python development. Feel free to use any of the videos/documentation found above!

Here's what you need to do:

1. **Create a new GitHub repository**
   This is where you'll upload your code for this task and future personal practice. Make it public or private — up to you!
2. **Set up a new ROS2 workspace**
   Use either your virtual machine or your local Linux setup (Ubuntu 22.04). Make sure the workspace builds and sources correctly.
3. **Inside that workspace, create two packages:**
   - A **Python** package (using `ament_python`)
   - A **C++** package (using `ament_cmake`)
4. **Implement a simple publisher and subscriber in each package**
   - You could do anything you'd like! For example: have the publisher send a string like `"Hello from ROS2!"` and have the subscriber print it to the terminal.
5. **Build your workspace and test your nodes**
   Make sure they work using `ros2 run` or by launching them together.
6. **Push everything to your GitHub repo**
   Be sure to commit your `src/`, `CMakeLists.txt`, `package.xml`, and any launch or config files you make.

If you're ever unsure about anything, don't hesitate to ask questions — everyone here started somewhere, and we're all learning together.

Next Steps & Communication

- **Share Your Work with Us**
    - Send the link to your GitHub repository to both Rayan Raad and Micheal Lythgoe
    - Don’t be afraid to communicate with either of us for help or guidance
- **Get Feedback**
    - We will review your repository, run your code, and give you feedback.
    - If any issues come up, we’ll guide you through fixing them
- **Choose Your First Intro Task**
    - After approval, we’ll assign or suggest an **introductory task** (from our current task list).
    - This will include either more integrated work with the rover, or something that will be actively used by the rover pilots on our next test!
- **Stay in Touch**
    - Always let us know in the team chat when:
        - You’ve completed a task.
        - You’re stuck and need help.