Ella Blue Jackson

Intro to Robotics

Lab 1 reflection

A node is a program that runs in ROS2 and does a task, like publish data or listen to data. On the other hand, a topic is the channel that nodes use to talk to each other. In this lab, the talker and the listener “communicate” on the /chatter topic. We use lots of small nodes in ROS2 because it is easier to manage. It is easier to build and debug smaller pieces than it is for a larger program. In addition, it allows us to reuse nodes and run them on different machines.

The hardest part for me was actually getting my package to build right. At some point, I moved my setup.py somewhere else I think, which caused me a lot of trouble. I also had missing files in resource/. For debugging, I did use AI. Unfortunately, it was not much help to me so I ended up deleting my package and restarting. Once I did that, everything seemed to work smoothly. I did have to refresh my memory on Python a bit but since I have been doing C++ for the past 6 semesters, things seemed easier than I had remembered.

Some tools provided were very helpful. The ros2 node list and topic list were helpful in seeing if my nodes existed in the right place and were actually running. The ros2 topic echo was probably the most helpful because I was able to see the numbers/strings coming through.

Sometimes it felt like messages were skipped or delayed. Before I restarted the lab, the numbers were coming in at random order so I knew something was wrong. After doing it a second time, I still saw that sometimes there was a delay. I wasn’t sure if it was because I had many terminals open and my laptop is old, or if it was just the program. After searching it up, I realized that this can happen when the subscriber can’t keep up, and old messages get dropped because of the queue size. If the publisher goes faster, the subscriber might fall behind and miss things. On the other hand, if the publisher is too slow, then the data isn’t useful in real time.

For counter and even number subscriber, I used the talker/listener code as a skeleton. For the counter, I switched to Int32 and added a variable that goes up by one every second. For the even subscriber, I copied the listener and added a check to print only the even numbers. At first, I forgot to add them in setup.py so ros2 run couldn’t find them, but once that was fixed, everything worked. To change it so the subscriber only prints multiples of 5 I would change the callback to:

if msg.data % 5 == 0:

...

This talker/listener pattern is simple, but I can imagine scaling it to a real robot. Each sensor would be its own talker, and then motor controllers would be subscribers. I am curious about how Quality of Service works in ROS2 when making sure the messages aren’t dropped when you really need them. I also want to learn how to connect ROS2 with simulation tools so I can test robot behavior without needing the actual body.