Project 1

By Apoorv Thapliyal

Overview

This assignment entails confirming the functionality of a virtual quadrotor setup, creating an "offboard" node, and plotting simulation results. Students complete simulations in ROS1 or ROS2, modify an offboard control program, and plot commanded versus actual vehicle positions.

1. Simulation Setup:
- For ROS1, setup includes starting offboard node, mavros, and QGroundControl.
- For ROS2, setup includes starting QGroundControl, MicroXRCE Agent, GZ bridge, and the 'px4 ros com' offboard node.
- Data recording is learned using 'rosbag' and 'ros2 bag record' commands.
2. Modify Offboard Control:
- Write a program to control vehicle position in 'offboard' mode.
- Program demonstrates a repeating pattern of movements and pauses at each waypoint.
3. Plotting Results:
- Recorded data is used to plot commanded versus actual vehicle positions as a function of time.
- Time is in seconds, and plots have labeled axes and legible font sizes for printing.

For this assignment, the X axis was considered as North and Y axis was considered as East. This places the Z axis into the plane.

Code Outline

The provided logic is for controlling a drone's movement in a simulated environment. Here's a concise report on the code:

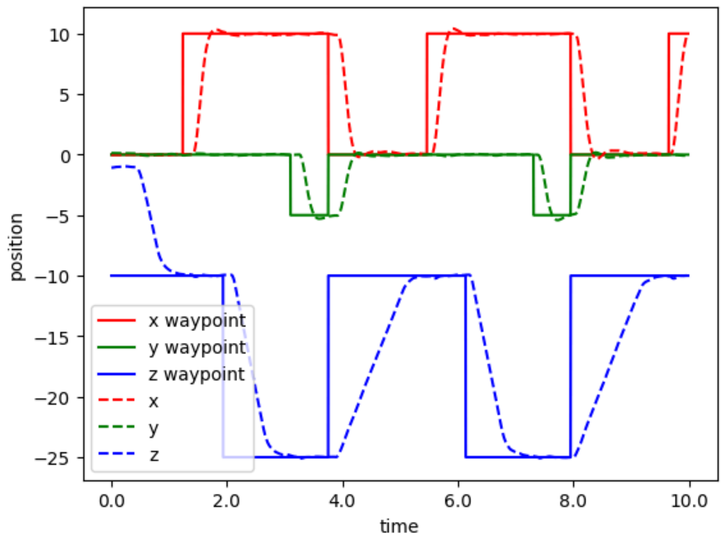
- The code begins by calling two methods, `publish_offboard_control_heartbeat_signal()` and `engage_offboard_mode()`, which handle communication with the drone's flight controller to initiate offboard mode and ensure proper communication.
- It is to be noted that the program waypoint navigation code does not arm the drone, arming is done manually.
- It checks if the drone is in offboard mode using the `vehicle_status.nav_state` attribute.
- Depending on the value of the `wp` variable, it publishes different position setpoints for the drone to follow.
- For each waypoint (`wp`), it publishes a specific position setpoint and checks if the drone is near that waypoint using the `dist2wp()` method.
- If the drone is 0.5m away from the waypoint, it increments a `counter` variable.
- Once the `counter` variable exceeds a certain threshold (375 in this case), it resets the counter and switches to the next waypoint (`wp`), cycling through a total of 4 waypoints.
- 375 was used to keep the drone at the waypoint for nearly 3s. 375 * publisher frequency (0.008) = 3s

- The code implements a basic waypoint navigation behavior, where the drone moves between predefined waypoints in the environment.
- A simple logic is used to determine when to switch to the next waypoint based on the distance to the current waypoint and the counter variable which tracks how long the drone has been at a position.

With this logic, the desired waypoints were tracked accurately, with the drone waiting at each waypoint for roughly 3s.

Output

Generating a bag file with the given code, the following graph was generated:



References

[1] https://github.com/fishros/ros2bag_convert
[2] https://code.umd.edu/conroyj/enae788m s24.git
[3] https://github.com/PX4/px4 ros com/tree/main