ME 410 - Week 3 Summary

Logan Boswell

Summary

This week's objectives were to configure the pitch and thrust joysticks and implement proportional, derivative, integral, and full PID control for the drone's pitch angle.

We began by setting joystick amplitudes: 10 degrees for pitch and 100 (with a neutral value of 100) for thrust. These were used to convert joystick counts into meaningful desired values for the controller.

The first controller tested was a proportional controller, using a gain of 10. Front motor speeds were set to desired_thrust + P_PITCH*pitch_err, and back motor speeds to desired_thrust - P_PITCH*pitch_err, inducing corrective torque based on pitch error. This was tested in four scenarios: level ground with thrust input, level ground with pitch input, changing pitch with no input, and changing pitch with thrust input. Sections of the plot below follow the listed order.

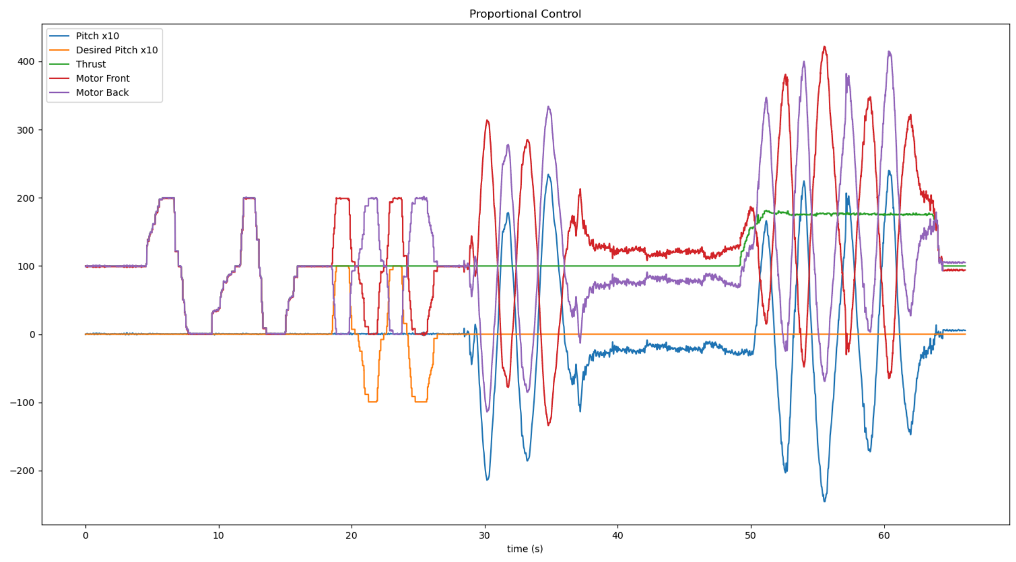

Figure 1: Proportional Controller Testing

Next, we implemented a derivative controller with a gain of 1. Motor speeds were adjusted based on angular velocity from the IMU (imu_data[5]) to dampen quick movements: front motors used desired_thrust - D_PITCH*imu_data[5] and back motors used desired_thrust +

D_PITCH*imu_data[5]. This was tested with pitch changes both with and without thrust input (Figure 2 below). Note: pitch data appeared noisy, likely due to hand shaking.

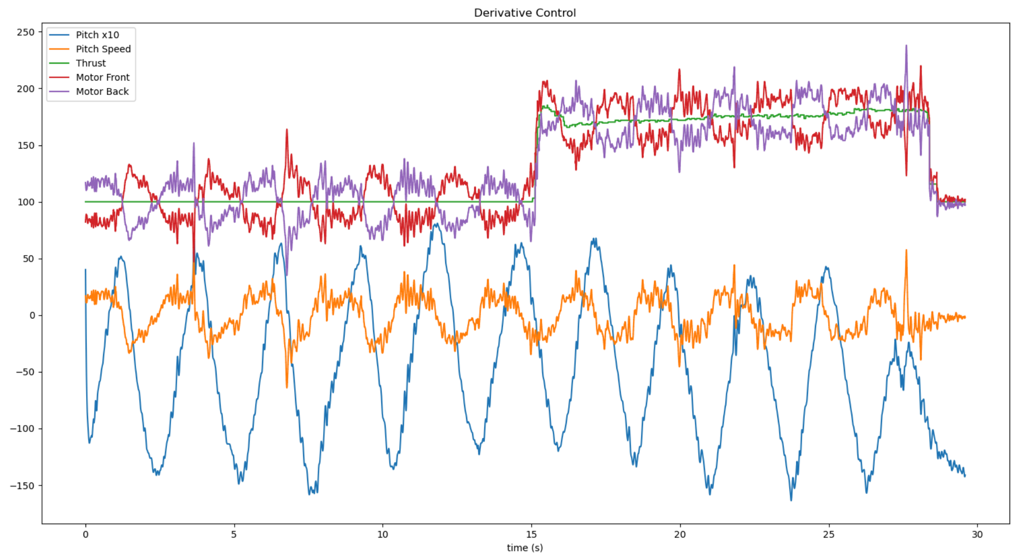

Figure 2: Derivative Controller Testing

An integral controller was added using a gain of 0.1. Each iteration, the contribution int_pitch += I_PITCH*pitch_err was calculated and bounded between ±100 (I_SAT) to prevent windup. This value was added and subtracted from front and back motor speeds, respectively, similar to the proportional approach. Tests included: no input, varying thrust input, and varying pitch input on level ground. The three sections of the plot are in the same order as the scenarios listed above.

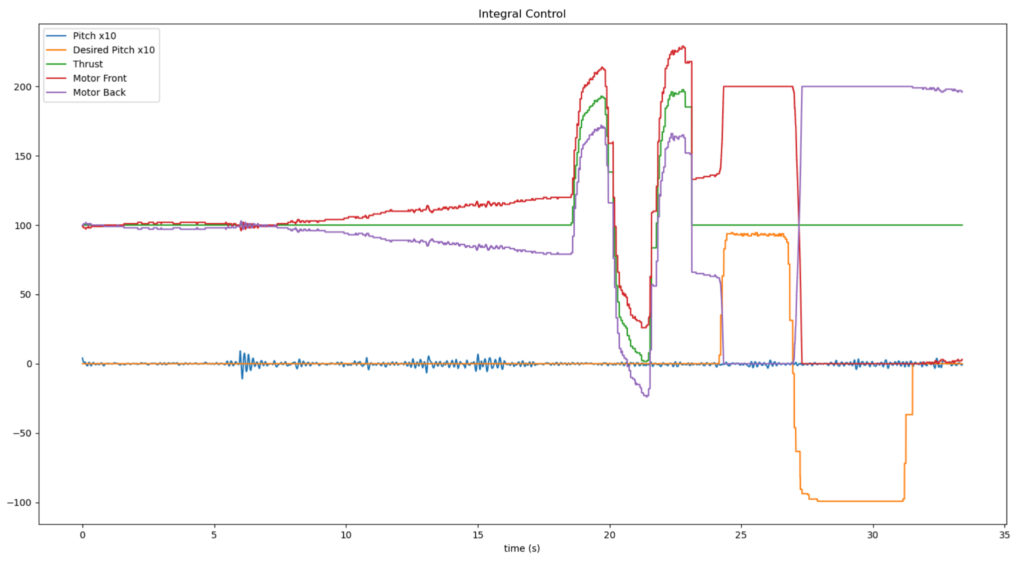

Figure 3: Integral Controller Testing

Finally, a full PID controller was implemented with combined gains. Front motor speeds used desired_thrust + P_PITCH*pitch_err - D_PITCH*imu_data[5] + int_pitch, and back motor speeds used desired_thrust - P_PITCH*pitch_err + D_PITCH*imu_data[5] - int_pitch. Testing included: fast step changes in pitch, slow changes in pitch input, slow changes in thrust input, and no input. Results are shown in the same order.

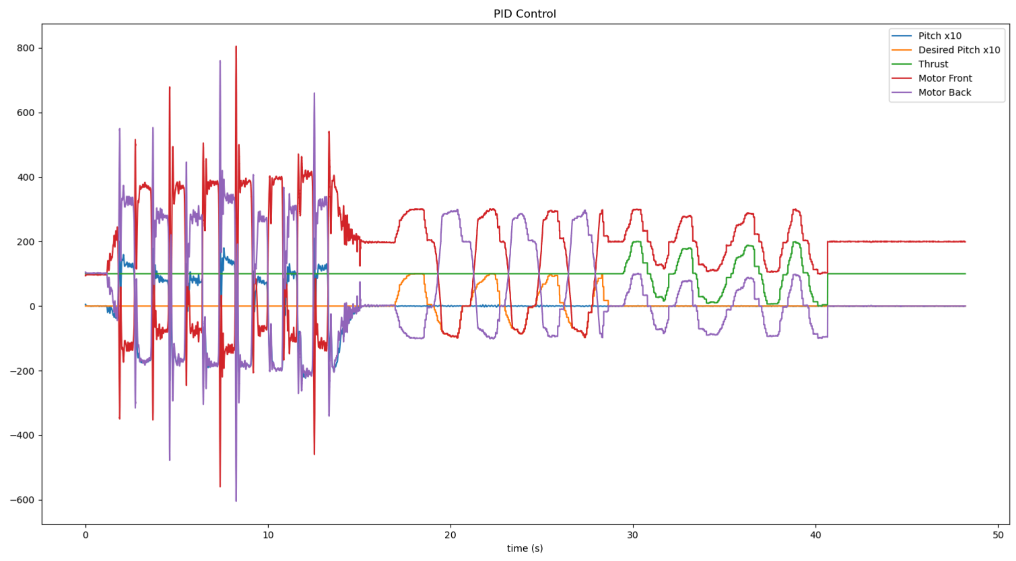

Figure 4: PID Controller Testing

Assessment - What Went Well

This week, I feel like mostly everything went well. We used the remote SSH extension with vscode to more easily edit code. Implementing the controllers and plotting data went quickly without many issues. From our plots we were able to clearly identify the sections associated with each controller test.

Assessment - What Did Not Go Well

I feel like we did not have many issues this week. There was a little delay between collecting data and writing the python script to plot the data for the first controller, but this process went quicker for the other controllers.

Assessment - Adjustments for Next Class

Overall, I think the main adjustment we need to make is just better debugging if we encounter errors, so we can use our in-class time more effectively and finding a quicker way to collect data and plot it.

Team Member Effort

Me - 49% (Worked on plotting code on my laptop after collect data as csv file; helped Ben with editing our .cpp file)

Ben - 51% (We worked on his laptop to edit our .cpp file; saved our data to csv file)