RBE500: Foundations of Robotics

Group Assignment Part 3

Student 1: Dhruv Agrawal
Student 2: Shreyas Khobragade
Student 3: Pranjali Rangnekar
Student 4: Roman Henry

Questions

This assignment is about controlling the robot joints.

1. **Implement a position controller for your robot joints. Here we want you to write PD Controllers yourselves. DO NOT use the readily available ROS packages.**
2. **You need to apply your controller only to one of the actuators of the robot: Actuator 4. That is the actuator right before the gripper.**
3. **You will need to switch the mode of the actuator 4 to current (effort) control mode. The rest of the actuators can stay in position control mode so that they maintain their positions rigidly while you are experimenting with actuator 4.**
4. **Your package will read the joint position values from actuator 4, receive a position reference value for the actuator through a service, and publish joint efforts/currents (continuously with high sampling rates) to the actuator to make it move to the desired location.**
5. **You will need to tune the PD gains (you do not need to calculate them in this assignment; the dynamics equation of robot is not provided anyways). While tuning, start with very small K_p gain and zero K_d gain. Gradually increase the K_p gain until you see some overshoot. Then, increase K_p and K_d gains together, and try to achieve a fast convergence with minimal overshoot.**
6. **Test your control node for three different joint position references, record the reference positions and current positions of the joint in a text file for a period of time (e.g. for 10 seconds with sampling time of 0.01 secs) and plot them via Matlab (or any other visualization software).**
7. **Take a video of the execution of the controller. Please use a common video codec and compression so that it is easy to open using common video players but does not take much space.**
8. **Please write a report about your implementation. The report does not have to be long, but it should explain all the steps of the implementation and it should include your control plots. Copy-pasting the code and the results is not enough.**

Implementation

- The dynamixel_sdk_examples.zip folder was used and modified for our implementation, where the read_write_node.cpp and the current_read_write_node.cpp file were combined to form a single file combined_read_write.cpp. The same process was done for combining the hpp files.
- From these files it was made sure that the topic set_current and the service get_position are available and working.
- We made sure that corresponding message and service are present in the dynamixel_sdk_custom_interfaces folder and they were named as SetCurrent.msg and GetPosition.srv.
- The PD controller node contains a publisher to set the current values to be given to motor 4, a client for getting the position values from the encoder of motor 4 and a custom service "PosRef.srv" for setting the desired position.
- While trying out different gain values for the PD controller, we found that the values $K_p = 0.3$ and $K_d = 0.01$ worked optimally for our case. This can also be verified by the video provided alogn with the report.
- We tried out 3 different desired positions: 2200, 1200 and 2900, and have plotted the results below.

Results

The results for 3 different desired position values can be seen from figures 1, 2 and 3

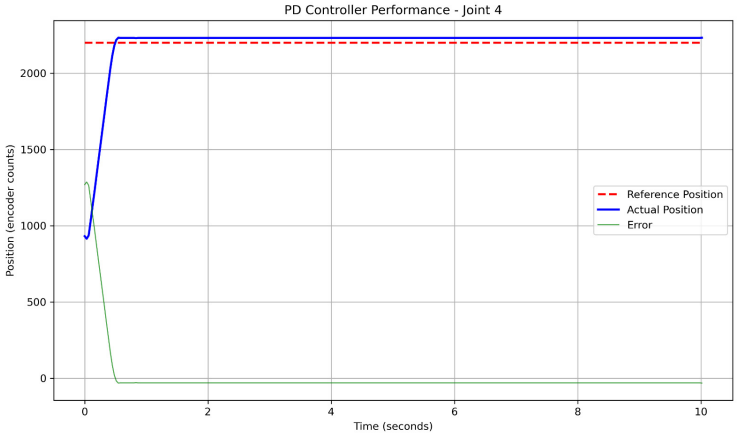

Figure 1: Desired Position 2200

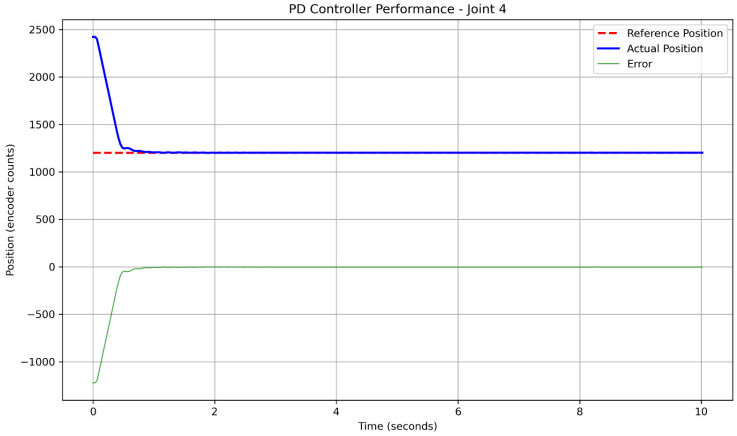

Figure 2: Desired Position 1200

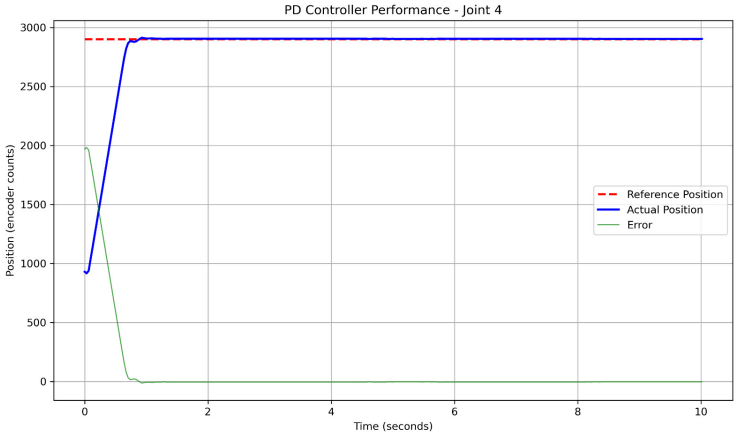

Figure 3: Desired Position 2900

From the figures it can be seen that the motor is tracking the desired position values in satisfactory time and with minimal overshoot.
The desired position, current position and error are plotted with respect to time, and from all the plots it is verified that the error converges to 0 within 1 second.