"Implementation of Torque Based Position Control for trajectory tracing by a Five DOF Robotic Manipulator"

A Graduate Project Report submitted to Manipal Academy of Higher Education in partial fulfilment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY
In
Electronics and Communication Engineering

Submitted by

Snehal Singh Tomar
Reg. No.:160907094

Under the guidance of

Dr. Shubhendu Bhasin		**Mr. Prashanth M. Prabhu**
Associate Professor	**&**	Assistant Professor- SelectionGrade
Department of Electrical Engineering		Department of E&C Engineering
IIT Delhi		Manipal Institute of Technology

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
MANIPAL INSTITUTE OF TECHNOLOGY
(A Constituent Institution of Manipal Academy of Higher Education)
MANIPAL – 576104, KARNATAKA, INDIA

JUNE 2020

ABSTRACT

This project aims to develop and implement a torque based control algorithm that enables a 5 Degree of Freedom manipulator (robotic arm) to move its end-effector to a specific location in the joint-space using torque inputs for specific links. This manipulator can then be mounted on mobile robots or quadrotors for specific applications. The mobile arm shall have promising applications in sectors like manufacturing, retail, commerce and, aerospace.

We have used the *ROBOTIS* **Open Manipulator-X** for all implementation and analysis purposes. Open Manipulator-X is an open source 5 DOF robotic arm prototyping platform that can be manipulated using its SDK as well as standard ROS (Robot Operating System) tools. We have executed the project in 3 stages namely: familiarization, basic implementation and, robust implementation. During familiarization, the task cut-out was to understand working of the Open Manipulator and the Dynamixel- XM430 series of motors that are used in each of the manipulator's links. The subsequent stages were basic and robust implementation which were mainly about implementation in simulation and real-world respectively.

In this study, we have simulated the Open Manipulator-X in Gazebo using standard ROS packages. We have successfully demonstrated control of the Open Manipulator' simulation using an open source GUI controller as well as tele-operation using keyboard commands. Our key contribution in this work has been the development and implementation of a torque-based position controller for this simulation, which we have explained in later sections of this report.

Since, we have demonstrated torque-based position control for the Open Manipulator, this would pave the way for application of a variety modern control algorithms on it. Thus, enhancing its utility in automation applications

Hardware used: Open Manipulator-X, Dynamixel- XM 430 motors.
Software used: Dynamixel SDK, ROS-Kinetic, Gazebo, Ubuntu 16.04(Operating System).
Programming Languages: Python, C++

LIST OF TABLES

LIST OF FIGURES

Contents

CHAPTER 1
INTRODUCTION

This chapter aims to give an overview of the overall aspects of this work. We first describe the nature of problem at hand, approach for solution and the progress made.

Most modern day manipulators operate in position-control mode. Although position control provides stability and is well suited for movement of the end-effector to desired coordinates; Manipulation of external objects is best handled in torque-control mode as it provides techniques to accommodate external forces and other non-linear behaviour in the system's model.

As of now, torque-controlled manipulators are being employed only for research purposes most of which happens to be in academia. Industrial applications are still carried out in position-control mode itself. Industries require torque-controlled manipulators to perform nuanced operations. Thus, a well-designed product that bridges this gap is the need of the hour.

Since, our implementation will be on an open source platform; the software we develop will be applicable to most industry-grade manipulators as well thus acting as a template for better controlled future robotic manipulators.

1.1 Objective

1.1.1 Primary Objective: To implement a standard torque-control algorithm on the *Open Manipulator-X* for trajectory-tracing by its end-effector; while operating its *Dynamixel XM-430* motors in "current based position" mode and obtaining appropriate current feedback from them.

1.1.2 Secondary Objective: To couple the *Open Manipulator-X* with a [RaspberryPi + Pixhawk-PX4] controlled Quad Rotor followed by control of the overall system's dynamics.

1.4 Organization of Report

The report is organised in the same fashion as mentioned on the "contents" page. We provide a background about the hardware, its specifications and necessary details about the controller already in-place in Chapter-2 (Background Theory). We then focus on our research approach, mathematical aspects of our controller in chapter-3 (Methodology). We present the implementation of this controller in Chapter-4 (Result Analysis). We discuss the insights obtained from analysis of results and scope of future work in the final chapter (Chapter-5).

CHAPTER 2
BACKGROUND THEORY

In this chapter, we discuss the theoretical aspects of the hardware we are working on, its supporting software, background knowledge of the algorithm we plan on implementing and its simulation.

Introduction to Project Title: The title "Implementation of Torque Based Position Control for trajectory tracing by a Five DOF Robotic Manipulator" implies that:
i. We intend to implement a control algorithm which models the system as something that takes motor-current as input and returns position coordinates of the end-effector as output. The desired torque to commanded current conversion shall be a part of the algorithm itself.
ii. This controller will be deployed on the *Open Manipulator-X* which is a 5 Degree of Freedom Manipulator (1-Base, 3-Links and 1-Gripper). Motion of each of its links is governed by a Dynamixel-XM430 servo motor.

2.1 Literature Review

We present the theoretical understanding developed thus far by going through various sources related different aspects of the project in this section.

2.1.1 The Open Manipulator-X

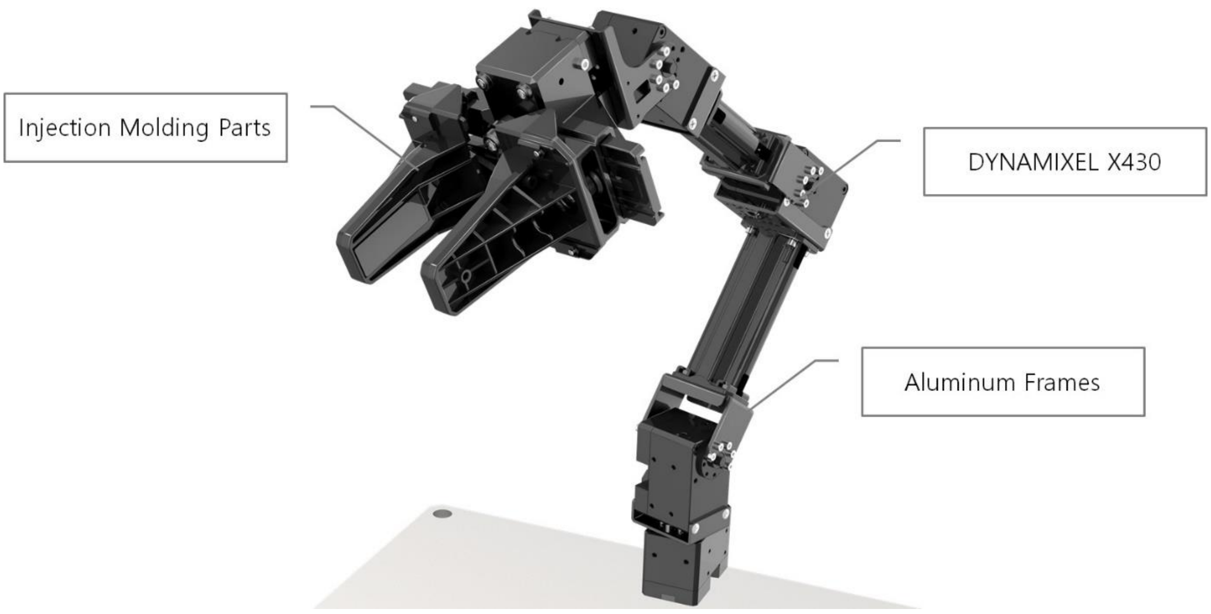


Figure 2.1: Physical Depiction of The Open Manipulator-X, Source: ROBOTIS e-manual for Open Manipulator-X [2]

The Open Manipulator is a ROS(Robot Operating System) enabled, 5-DOF robotic manipulator, which is manufactured by ROBOTIS INC. It is interfaced by via the OpenCR(embedded board), which is a U2D2 converter that helps publish ROS messages to the Open Manipulator. Technical specifications of the manipulator are as follows:

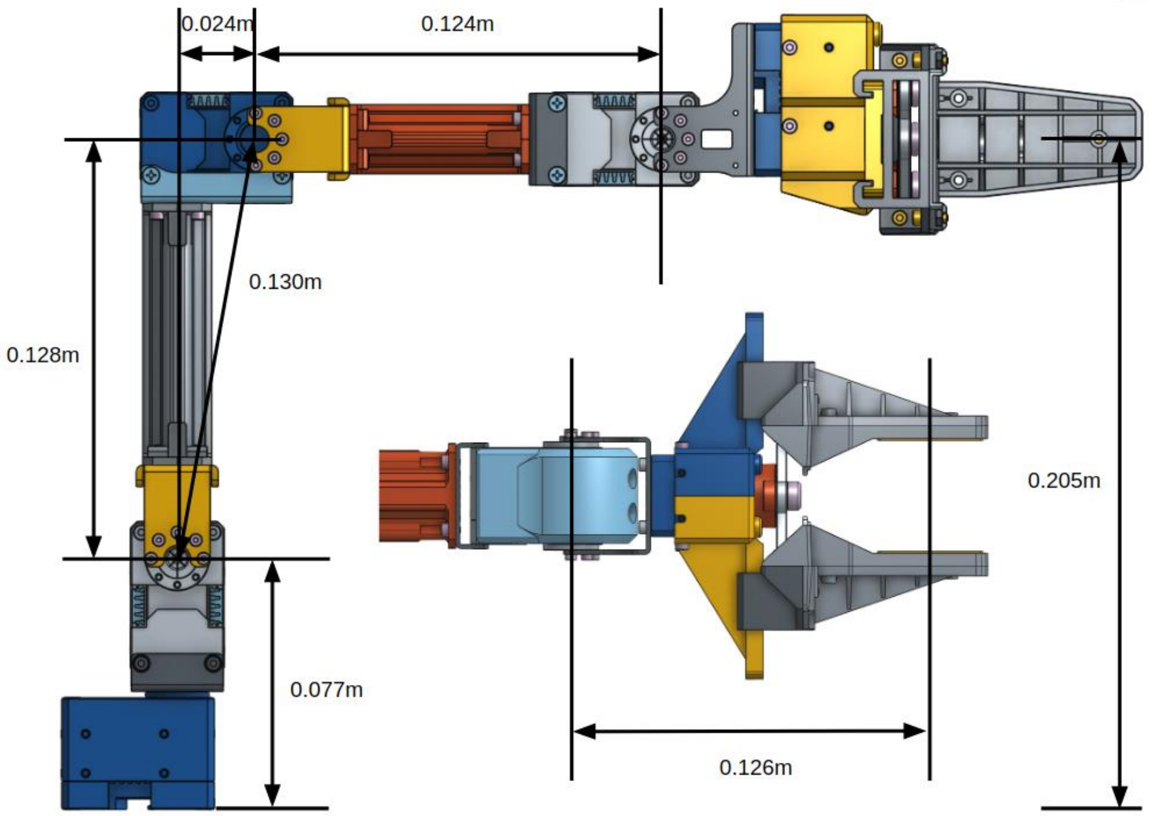
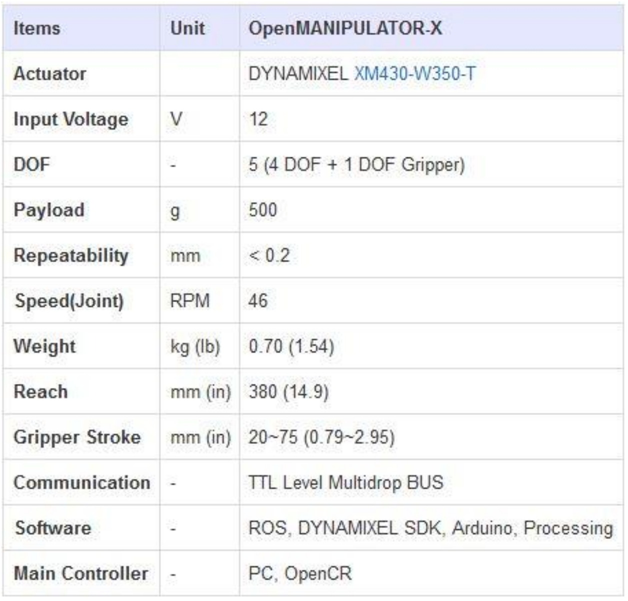


Figure 2.2: Technical Drawings of the manipulator describing its structure and dimesnsions, Source: ROBOTIS e-manual for Open- Manipulator-X [2]

Table 2.1: Technical Specifications of the Manipulator, Source: ROBOTIS e-manual for Open Manipulator-X [2]

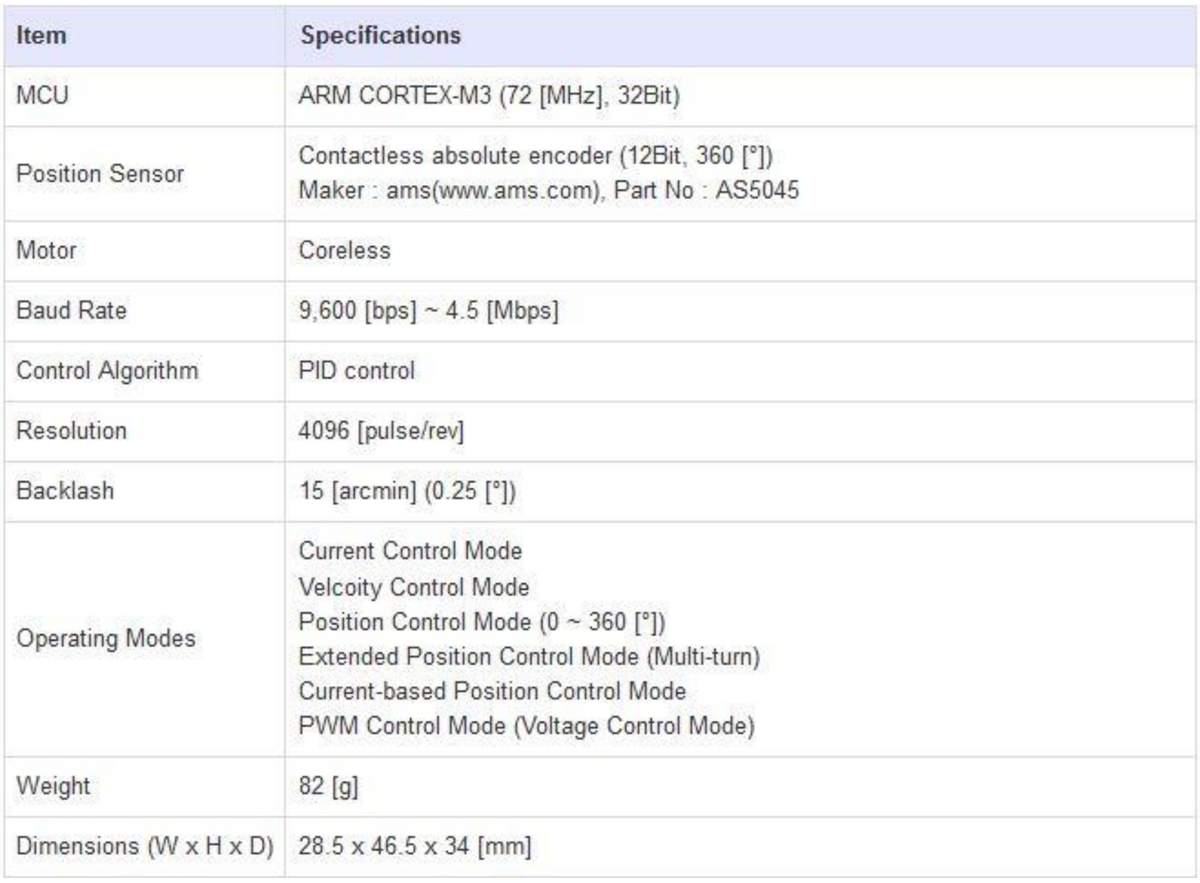


Items	Unit	OpenMANIPULATOR-X
Actuator		DYNAMIXEL XM430-W350-T
Input Voltage	V	12
DOF	-	5 (4 DOF + 1 DOF Gripper)
Payload	g	500
Repeatability	mm	< 0.2
Speed(Joint)	RPM	46
Weight	kg (lb)	0.70 (1.54)
Reach	mm (in)	380 (14.9)
Gripper Stroke	mm (in)	20~75 (0.79~2.95)
Communication	-	TTL Level Multidrop BUS
Software	-	ROS, DYNAMIXEL SDK, Arduino, Processing
Main Controller	-	PC, OpenCR

2.1.2 The Dynamixel XM-430 motor (actuator)

Each of the 5 links in the Open Manipulator is actuated by this servo motor. The motor can be controlled either using the Dynamixel SDK (a python package developed by ROBOTIS Inc. or through ROS messages. Technical specifications of this motor are as follows:

Table 2.2: Dynamixel-XM430 Specifications, Source: ROBOTIS e-manual for Dynamixel-XM430 [5]



Item	Specifications
MCU	ARM CORTEX-M3 (72 [MHz], 32Bit)
Position Sensor	Contactless absolute encoder (12Bit, 360 [°]) Maker : ams(www.ams.com), Part No : AS5045
Motor	Coreless
Baud Rate	9,600 [bps] ~ 4.5 [Mbps]
Control Algorithm	PID control
Resolution	4096 [pulse/rev]
Backlash	15 [arcmin] (0.25 [°])
Operating Modes	Current Control Mode Velcoity Control Mode Position Control Mode (0 ~ 360 [°]) Extended Position Control Mode (Multi-turn) Current-based Position Control Mode PWM Control Mode (Voltage Control Mode)
Weight	82 [g]
Dimensions (W x H x D)	28.5 x 46.5 x 34 [mm]

2.1.3 ROS + open-CR interface

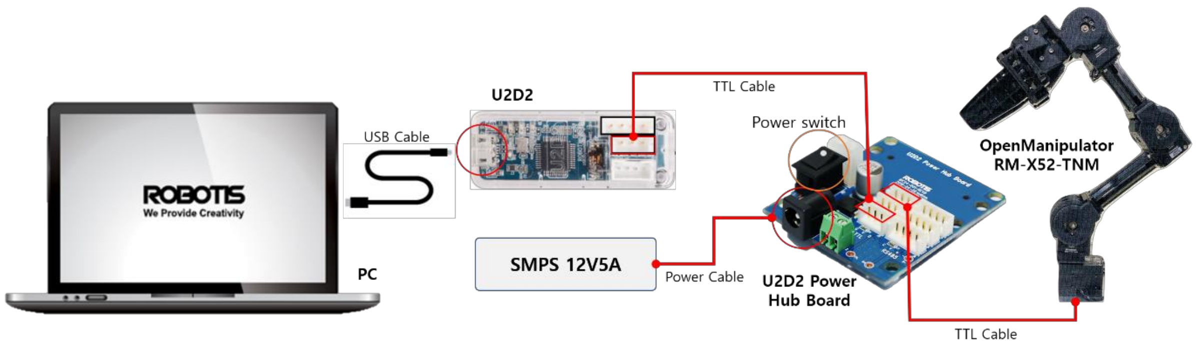


Figure 2.3: Interfacing the Open Manipulator using ROS via the U2D2, Source: ROBOTIS e-manual for Open Manipulator-X [2]

The Robot Operating System (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behaviour across a wide variety of robotic platforms. The U2D2 relays ROS messages (being published on specific topics) to the Open-CR thus enabling the manipulator to perform specific operations. Another method to control the Open Manipulator is via the *Dynamixel Python SDK.*

2.1.4 What is Torque-Control?

As stated in *Ricardo P. Aguilera* et al, Control of Power Electronic Components and Systems, 2018; Direct Torque control (DTC) for motor drive applications has been well established in both academia and industry. It offers a simple control structure, fast response, and robust operation. The torque and flux references are tracked using hysteresis controllers and a switching table implemented with LUT is used for selecting the optimum converter's output. Similarly, direct power control (DPC) for grid-connected applications was developed afterward to control the instantaneous active and reactive power directly by selecting the optimum switching state of the converter.

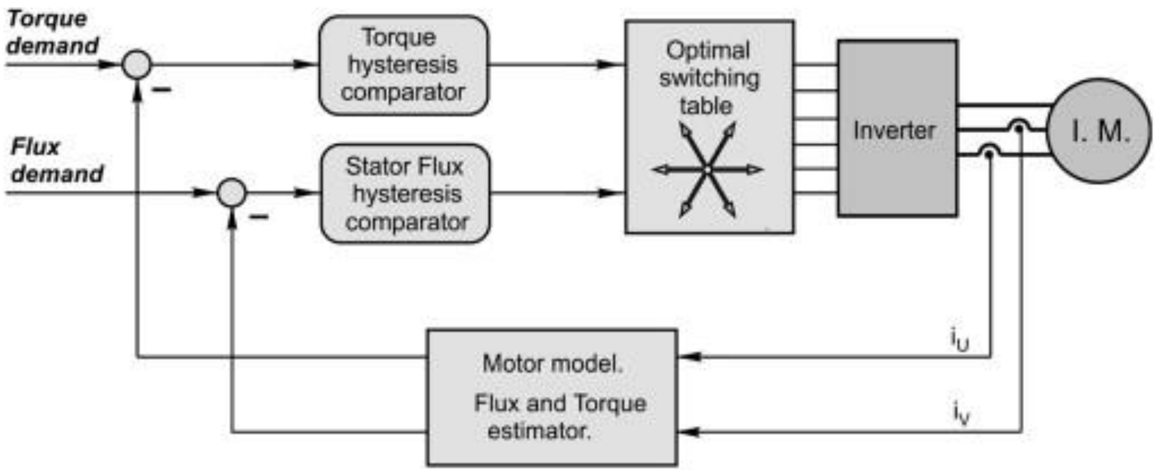

Torque demand
Flux demand
Torque hysteresis comparator
Stator Flux hysteresis comparator
Optimal switching table
Inverter
I. M.
Motor model. Flux and Torque estimator.
i_U
i_V

Figure 2.4: Block Diagram of a standard motor torque controller, Source: *Ricardo P. Aguilera* et al, Control of Power Electronic Components and Systems, 2018[9]

2.1.5 *About ROS*

ROS is an open source Meta operating system. It provides a structured communications layer above the host operating systems of a heterogeneous compute cluster. ROS is a collection of libraries and APIs that enable us to treat the various components of a robot system as nodes and establish communication between them in the form of standard ROS messages being transmitted over ROS topics. Thus ROS is extremely helpful in interfacing multiple sensors and actuators over a single controller. It is highly useful for testing, simulation, and operation of complex robotic systems. The following figure illustrates the key functionality provided by ROS:

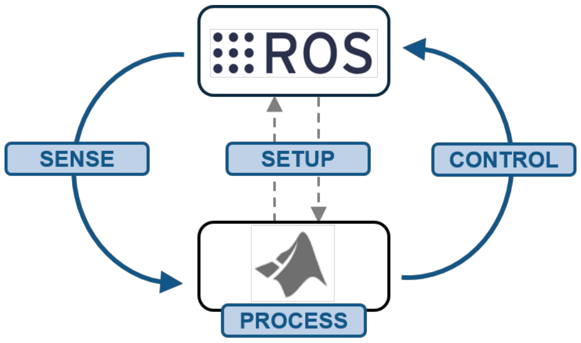


Figure 2.5: Overview of the Robot Operating System and its application scenario, Source: ROS Wiki [10]

2.1.6. About Gazebo

Gazebo offers the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments. It is a well-designed simulator that makes it possible to rapidly test algorithms, design robots, perform regression testing, and train AI system using realistic scenarios. It has a robust physics engine, high-quality graphics, and convenient programmatic and graphical interfaces. Most Importantly, It provides an interface for ROS packages to control simulated robots effectively. The following figure shows a simulation of the Open Manipulator-X in Gazebo:

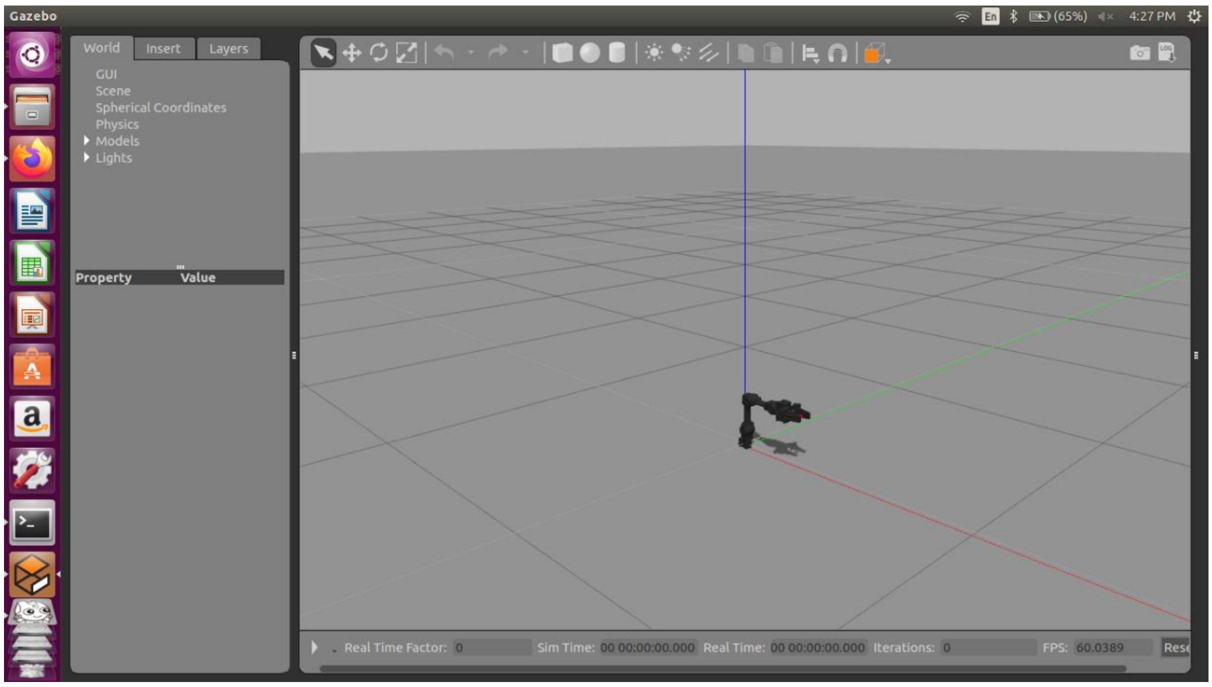


Figure 2.6: The Open Manipulator –X upon spawn in the Gazebo simulation environment

2.2 Summary of literature review

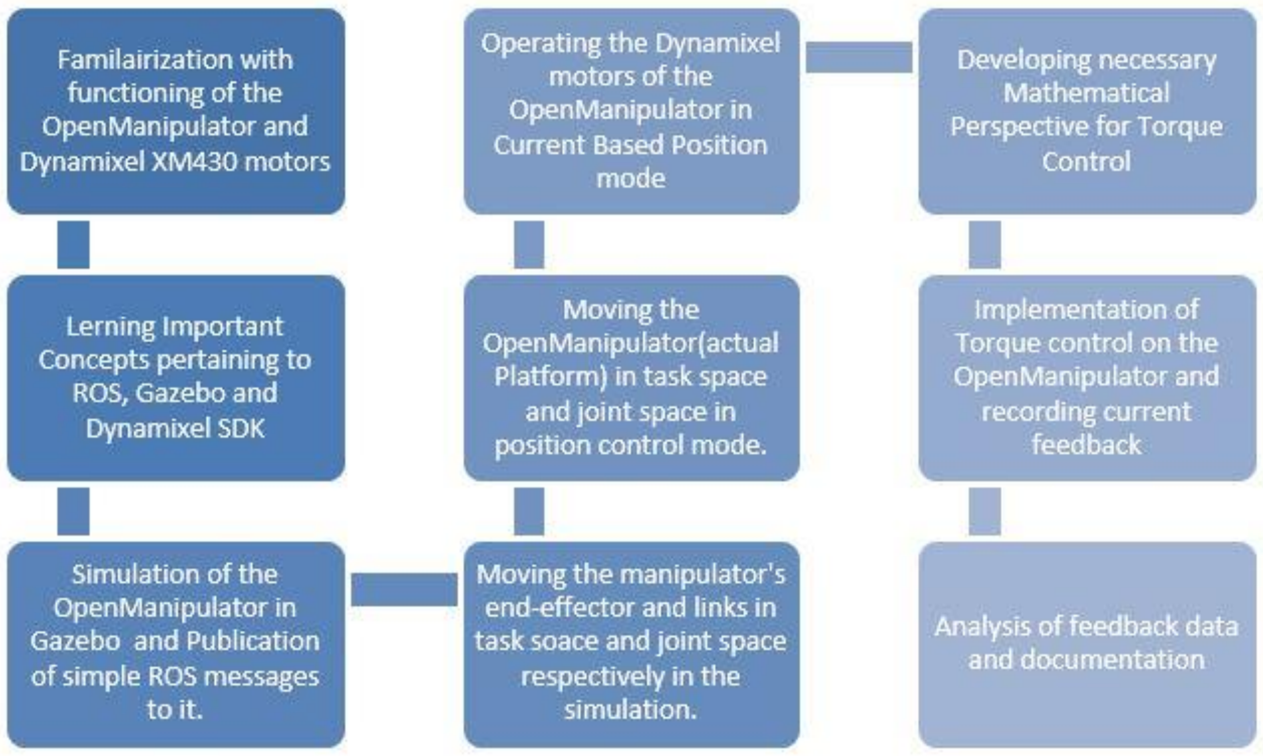
Based on an extensive literature survey, we were able to narrow down the tools to be used for simulation and the techniques required for implementation of the torque based position controller. Both Gazebo and ROS are highly versatile platforms with a variety of capabilities for simulation of robots. Their ease of inter-operability, is an added advantage which makes integration and simulation of complex robots simpler.

CHAPTER 3
METHODOLOGY

This chapter aims to provide an overall idea of the process adopted to work on this research problem. We also highlight all hardware and software technologies used in this project along with their specific application, constraints and problems caused during execution.

3.1 Detailed Methodology

The following diagram illustrates the execution process adopted for this project:



Familairization with functioning of the OpenManipulator and Dynamixel XM430 motors
Lerning Important Concepts pertaining to ROS, Gazebo and Dynamixel SDK
Simulation of the OpenManipulator in Gazebo and Publication of simple ROS messages to it.
Moving the manipulator's end-effector and links in task soace and joint space respectively in the simulation.
Moving the OpenManipulator(actual Platform) in task space and joint space in position control mode.
Operating the Dynamixel motors of the OpenManipulator in Current Based Position mode
Developing necessary Mathematical Perspective for Torque Control
Implementation of Torque control on the OpenManipulator and recording current feedback
Analysis of feedback data and documentation

Figure 3.1: A stage-wise graphical depiction of our proposed methodology.

We have accomplished the first three stages of our project execution along with GUI based manipulation, Tele-operation using keyboard, and formulation & implementation of a torque based position controller. We have understood the fundamentals of torque-control and have derived necessary mathematical foundation for control and dynamics (inverse kinematics) of the Open Manipulator. We discuss this mathematical analysis in the sections that follow:

3.1.1. Kinematics

In this section, we discuss standard forward and inverse Kinematics of multiple link manipulators as discussed in "*Introduction to Robotics" by John J. Craig* [7]. These ideas shall be of particular interest at a later stage, when we proceed with the controller design.

3.1.1.1 Forward Kinematics

Forward Kinematics is used to calculate the position of the end effector and each joint w.r.t to The base frame by calculating the transformation matrix of one joint w.r.t to the previous joint. Consider the figure is shown below in which two joints are shown and are connected by the link. In the figure we have three frames {P}, {Q}, and {R}. Frame {R} differs from frame {i-1} only by rotation α_{i-1} and Frame {Q} differs from {R} by a translation d_i.

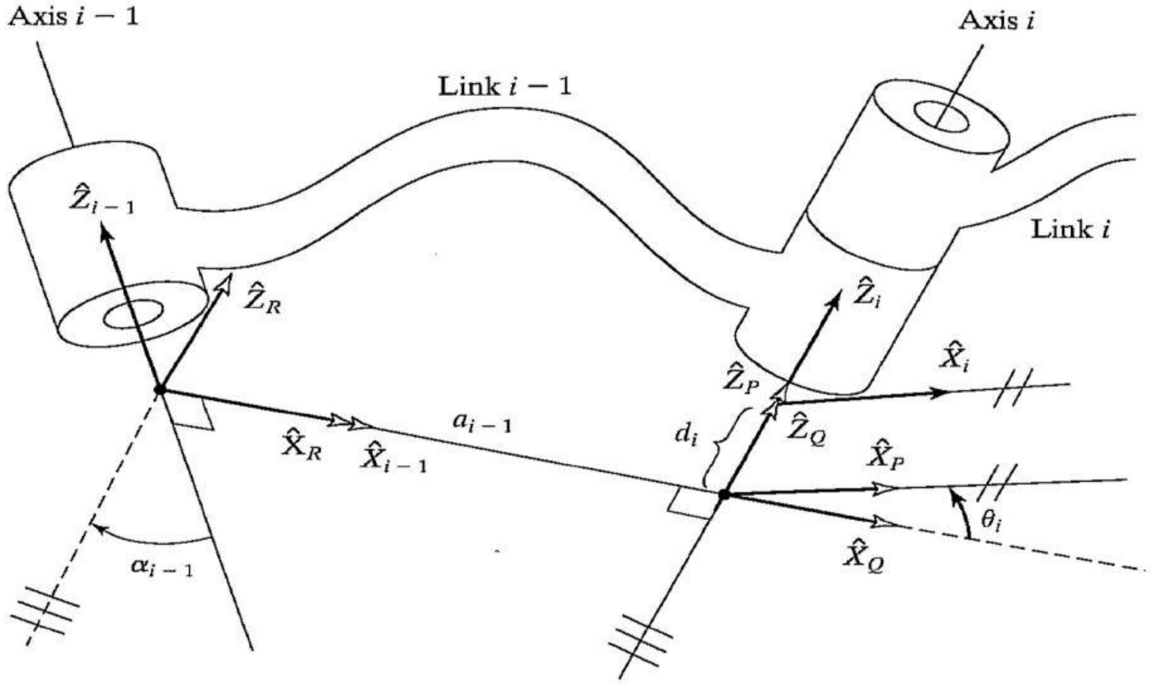


Figure 3.2: Two joints connected by a link with Frames {P}, {Q} and {R} as shown, source: "*Introduction to Robotics" by John J. Craig* [7]

The position of the ith joint can be written with request to the joint i-1 and the transformation matrix as follows. Here ${}^{i}P$ defines the position of the *ith* joint with respect to base or world frame and ${}^{A}T_{B}$ defines the transformation matrix of B with respect to A.

$$ {}^{i-1}P = {}^{i-1}_{R}T \, {}^{R}_{Q}T \, {}^{Q}_{P}T \, {}^{P}_{i}T \, {}^{i}P \quad (3.1.1) $$

$$ {}^{i-1}P = {}^{i-1}_{i}T \, {}^{i}P, \quad (3.1.2) $$

The formulae discussed in the above excerpt can be used to calculate the position of each joint, considering the manipulator's base as the origin.

3.1.1.2 Inverse Kinematics

In this, we are given initial position and desired position, we have to calculate the change in the angular position of each joint required to reach the final desired position. This can be computed using the position change of joints, orientation change, and jacobian. There are many ways to calculate the inverse kinematics but the one described below is used in the open manipulator package. The change in the angular position matrix can be written in terms of Jacobian and change in the position and orientation matrix.

$$\Delta = J^{-1}\Delta S \qquad (3.1.3)$$

$$\text{Where } \Delta S = [\Delta x \;\; \Delta y \;\; \Delta z \;\; \Delta\alpha \;\; \Delta\beta \;\; \Delta\gamma\,]^{T}$$

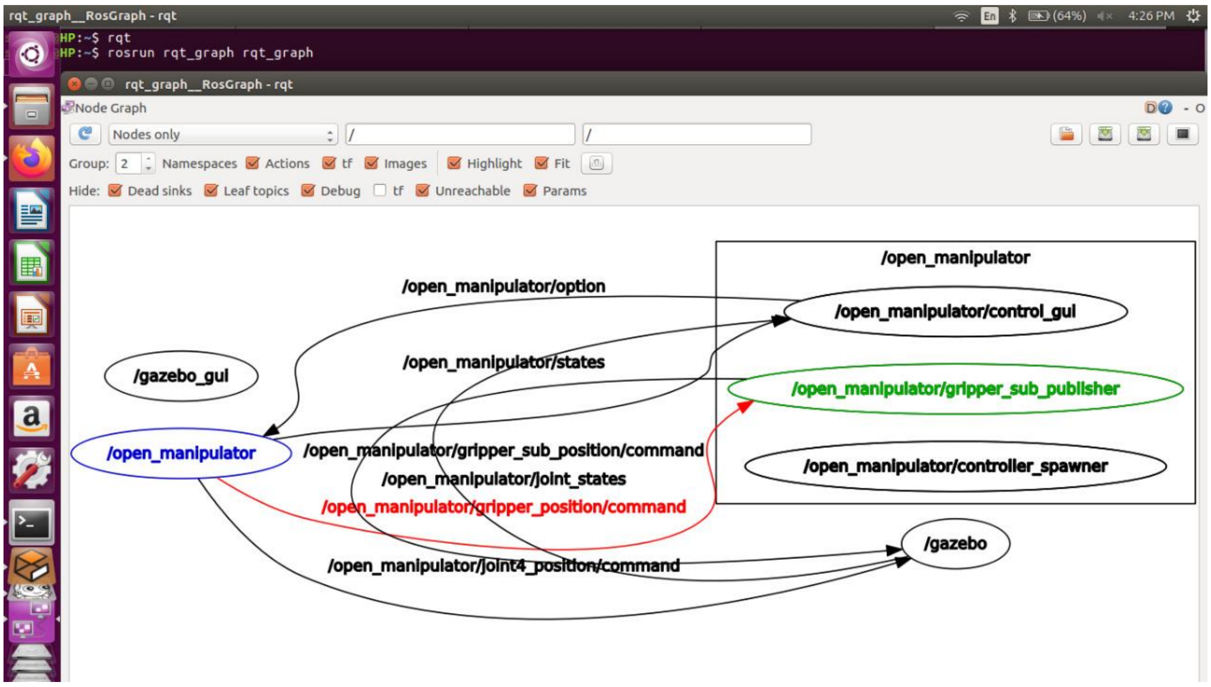
$$\Delta = [\Delta_1 \;\; \Delta_2 \dots \;\; \Delta_n\,]^{T}$$

$$J = \Big[\big[\partial x/\partial\theta_1 \; \partial y/\partial\theta_1 \; \partial z/\partial\theta_1 \; \partial\alpha/\partial\theta_1 \; \partial\beta/\partial\theta_1 \; \partial\gamma/\partial\theta_1\big] \;\; \vdots \;\; \big[\partial x/\partial\theta_n \; \partial y/\partial\theta_n \; \partial z/\partial\theta_n \; \partial\alpha/\partial\theta_n \; \partial\beta/\partial\theta_n \; \partial\gamma/\partial\theta_n\big]\Big]^{T}$$

The jacobian can be calculated using the rotation matrix and the difference in the original position and desired position. This formula can be used to calculate the angular position change of the joints required to move the manipulator to the desired position.

3.1.2 *Manipulation using GUI controller*

Robotis provides an open source ROS package called *open_manipulator_control_gui* as part of the *open_manipulator_controller* API that enables a user to command specific joint-position values to the Gazebo simulation for realization of a position in Joint space/ Task space. This tool enables control of the gripper (attached to end-effector) as well. The following is a plot of the ROS nodes and ROS topics, along with the way they Publish/Subscribe to topics with each other. These nodes and their Publishers as well as Subscribers were realized in python using the *rospy* library.


rqt_graph__RosGraph - rqt
Node Graph
/open_manipulator
/open_manipulator/option
/open_manipulator/control_gui
/gazebo_gui
/open_manipulator/states
/open_manipulator/gripper_sub_publisher
/open_manipulator
/open_manipulator/gripper_sub_position/command
/open_manipulator/controller_spawner
/open_manipulator/joint_states
/open_manipulator/gripper_position/command
/gazebo
/open_manipulator/joint4_position/command

```
HP:~$ rqt
HP:~$ rosrun rqt_graph rqt_graph
```

Nodes only

Group: 2 [✓ Namespaces] [✓ Actions] [✓ tf] [✓ Images] [✓ Highlight] [✓ Fit]

Hide: [✓ Dead sinks] [✓ Leaf topics] [✓ Debug] tf [✓ Unreachable] [✓ Params]

Figure 3.3: rqt_graph depicting the various ROS nodes, their topics and messages being relayed in the GUI control for simulation

Here "/open_manipulator" depicts the input node & "/open_manipulator/states" depicts the Rostopic that allows publication /subscription of a joint's angular position vector. "/gazebo" depicts the simulation node which subscribes to the "/open_manipulator/states" and reflects the command outputs in simulation.

3.1.3 Teleoperation using keyboard

The following figure shows the ROSnodes and Publisher/Subscriber structure teleoperation of the simulation using keyboard. Teleoperation essentially refers to moving particular joints of the openManipulaor's simulation in specific directions using keyboard commands.

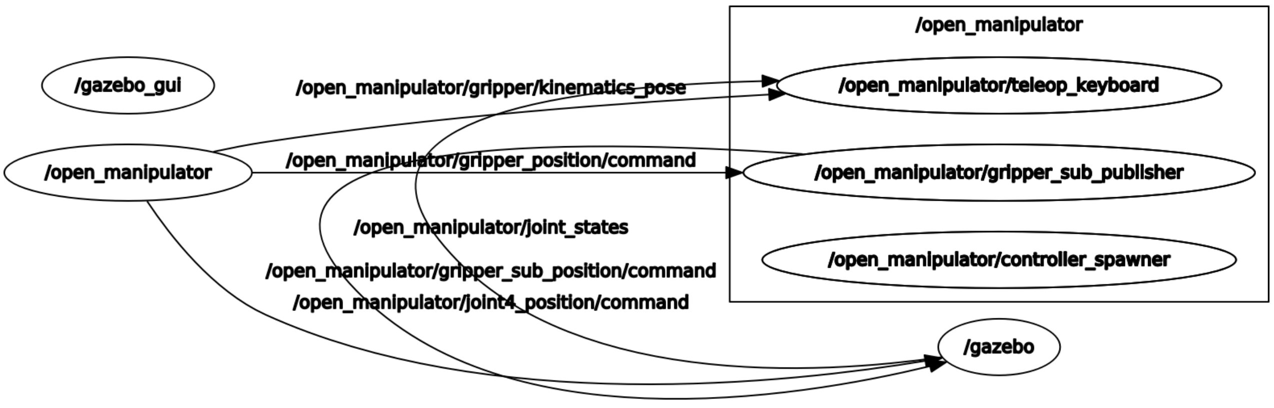

/open_manipulator
/gazebo_gui
/open_manipulator/gripper/kinematics_pose
/open_manipulator/teleop_keyboard
/open_manipulator
/open_manipulator/gripper_position/command
/open_manipulator/gripper_sub_publisher
/open_manipulator/joint_states
/open_manipulator/gripper_sub_position/command
/open_manipulator/controller_spawner
/open_manipulator/joint4_position/command
/gazebo

Figure 3.4: rqt_graph depicting the various ROS nodes, their topics and messages being relayed in the Teleoperation for simulation

3.2. Design of Torque Based Position Controller

In this section we present a detailed explanation of the based position controller that we have deployed on Link-4 Open Manipulator's simulation in Gazebo. The following figure provides an overall description of the controller:

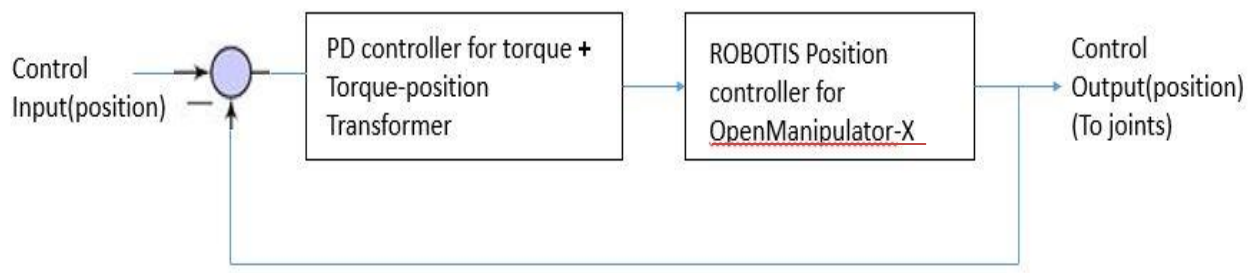

Control Input(position)
PD controller for torque + Torque-position Transformer
ROBOTIS Position controller for OpenManipulator-X
Control Output(position) (To joints)

Figure 3.5: Complete Block Diagram of the designed torque based position controller

The Robotis Position Controller is an open source package that is already employed by the Open Manipulator for position control. We have cascaded our torque based position controller with it to be able to provide position commands (calculated on the basis of torque input) to the pre-existing controller. The pre-existing ROBOTIS Position controller is responsible to taking a particular link to the desired position in simulation. We now take an in-depth look into the [PD controller for torque + Torque-position Transformer] block.

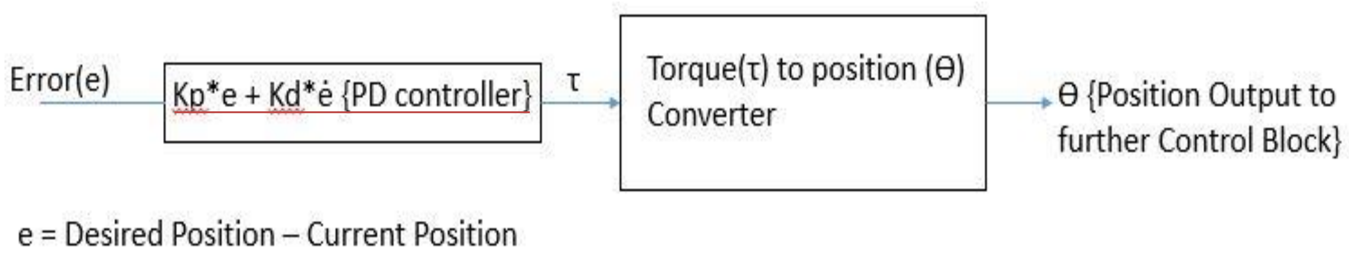

Error(e)
Kp*e + Kd*ė {PD controller}
τ
Torque(τ) to position (Θ) Converter
Θ {Position Output to further Control Block}
e = Desired Position – Current Position

Figure 3.6: A detailed block diagram of the PD controller as implemented in cascade with the pre-existing position controller

We now examine the mathematical aspects and working of this controller in detail;

3.2.1 Mathematical Aspects of the Controller

Working of the torque based position controller for Link-4 is explained in the following steps:

By "position", we imply angular position; Θ in radians.

1. Let the desired (commanded) Position be $\mathrm{P}_f = \begin{bmatrix} P_{fx} \\ P_{fy} \\ P_{fz} \end{bmatrix}$

2. Let the initial Position be $\mathrm{P}_i = \begin{bmatrix} P_{ix} \\ P_{iy} \\ P_{iz} \end{bmatrix}$

Where Components along X,Y and Z represent angular displacement from initial position along joints 1,2 and 3 respectively

3. Hence, the error in position (as shown in the block diagrams of the overall closed loop system as well as the PD torque controller) is given by:

$$\mathrm{E} = \mathrm{P}_f - P_i = \begin{bmatrix} E_x \\ E_y \\ E_z \end{bmatrix} \qquad (3.2.1)$$

Where; $\begin{bmatrix} E_x = P_{fx} - P_{ix} \\ E_y = P_{fy} - P_{iy} \\ E_z = P_{fz} - P_{iz} \end{bmatrix}$

4. Now, as depicted in Fig. 3.6, Torque to be applied to motors along different joints is given by:

$$\tau = K_p.[E] + K_d.[\dot{E}] \qquad (3.2.2)$$

$$where, [\dot{E}] = \begin{bmatrix} \dot{E}_x = \frac{\partial E_x}{\partial t} \\ \dot{E}_x = \frac{\partial E_x}{\partial t} \\ \dot{E}_x = \frac{\partial E_x}{\partial t} \end{bmatrix}$$

5. Since our controller provides torque along the various joints as output and the ROBOTIS position controller takes position (θ) as input; it is important that we convert torque to position along different joints. To do so we use the Inertia Tensor[I] provided in the OpenManipulator's URDF (Unified Robot Description Format) file.

6. The Inertia Tensor provides values for Moment of Inertia of a rigid body about its centre of mass about its various possible axes of rotation. For our application, we are concerned only with the Inertia Tensor of OpenManipulator's Link-4, which is given by(All values are in Kg-m^2) :

$$[I] = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} = \begin{bmatrix} 3.0654\text{x}10^{-5} & -1.2764\text{x}10^{-6} & -2.6874\text{x}10^{-7} \\ 0 & 2.4230\text{x}10^{-4} & 1.1559\text{x}10^{-8} \\ 0 & 0 & 2.5155\text{x}10^{-4} \end{bmatrix}$$

7. Since, $[\tau] = [\mathrm{I}].[\alpha]$ Therefore, $[\alpha] = [I]^{-1}.[\tau]$. (3.2.3)
where, $[\tau]$ is the torque vector, $[I]$ is the Inertia Tensor, and $[\alpha]$ is the angular acceleration vector.

8. Taking $[\tau]$ to be constant over an infinitesimally short period of time, T(.01 seconds) and taking $[\theta]$ to be the position output of the PD controller block we have;

$$\begin{bmatrix} \theta_x \\ \theta_y \\ \theta_z \end{bmatrix} = \begin{bmatrix} P_{ix} \\ P_{iy} \\ P_{iz} \end{bmatrix} + \begin{bmatrix} \frac{\alpha_x . T^2}{2} \\ \frac{\alpha_y . T^2}{2} \\ \frac{\alpha_z . T^2}{2} \end{bmatrix} \quad (3.2.4)$$

9. $[\theta]$ is then commanded to the ROBOTIS Position Controller which takes the link-4 to a new position vector, $[P_{new}]$.

10. $[P_i]$ is then updated as $[P_{new}]$ and the next iteration of the control loop begins.

11. These iterations then continue until, $[E] > [Tolerance]$ where;

$$[Tolerance] = \begin{bmatrix} 0.01rad \\ 0.01rad \\ 0.01rad \end{bmatrix}$$

The Python program for this implementation has been appended in Appendix-A.

3.3 Summary

We have explained various implementations done throughout the course of this project and their mathematical background. We shall discuss our results in the forthcoming section

CHAPTER 4
RESULT ANALYSIS

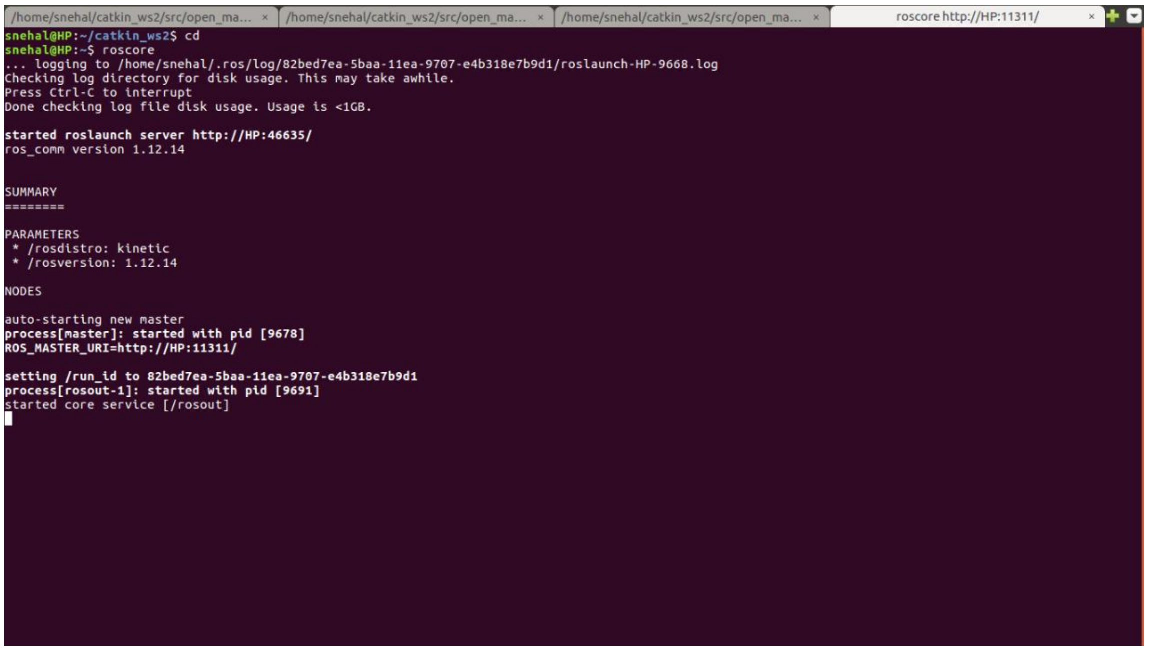
In this chapter, we first present our simulation along with a detailed discussion on the various ROS nodes involved, their topics and messages being relayed. These messages are responsible for conveying user instructions to the URDF file which then changes the robot position/state as commanded.

4.1 The Simulation

In this section, we showcase the process adopted and the outcome of simulating the Open Manipulator's CAD model in GAZEBO so that it behaves exactly as it would in a real world scenario. We also discuss the various nodes we will be launching and the topics, they will be communicating upon. All these simulations were executed on a ROS Kinetic+Ubuntu 16.04 environment with all Open Manipulator dependencies setup as per requirements.

4.1.1 Launch File

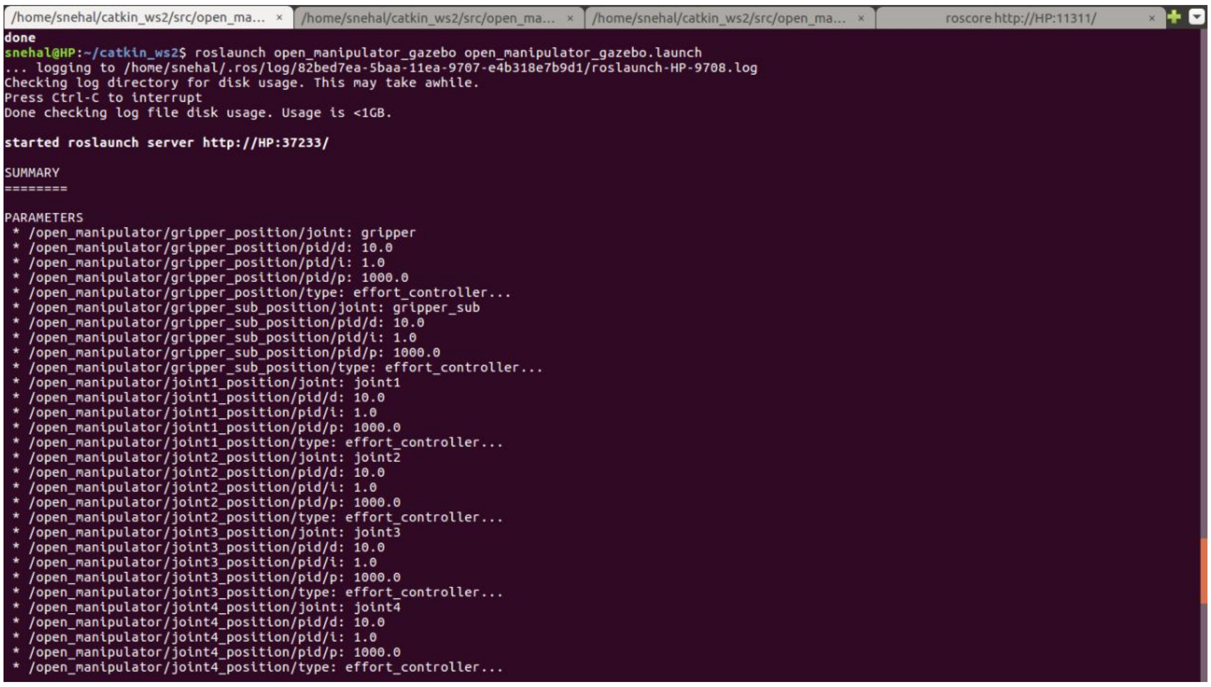
4.1.1.1 Starting the ROS core: The ROS core serves as the central hub with multiple ports that allows other nodes to communicate via its services using a request and acknowledge mechanism. In figure 4.1. We have shown Terminal-launch of the ROS core.


/home/snehal/catkin_ws2/src/open_ma...
/home/snehal/catkin_ws2/src/open_ma...
/home/snehal/catkin_ws2/src/open_ma...
roscore http://HP:11311/

```
snehal@HP:~/catkin_ws2$ cd
snehal@HP:~$ roscore
... logging to /home/snehal/.ros/log/82bed7ea-5baa-11ea-9707-e4b318e7b9d1/roslaunch-HP-9668.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://HP:46635/
ros_comm version 1.12.14


SUMMARY
========

PARAMETERS
 * /rosdistro: kinetic
 * /rosversion: 1.12.14

NODES

auto-starting new master
process[master]: started with pid [9678]
ROS_MASTER_URI=http://HP:11311/

setting /run_id to 82bed7ea-5baa-11ea-9707-e4b318e7b9d1
process[rosout-1]: started with pid [9691]
started core service [/rosout]
```

Figure 4.1: The ROS (kinetic) core started in Terminal.

4.2.1.2 Launching the Gazebo Simulation



```
done
snehal@HP:~/catkin_ws2$ roslaunch open_manipulator_gazebo open_manipulator_gazebo.launch
... logging to /home/snehal/.ros/log/82bed7ea-5baa-11ea-9707-e4b318e7b9d1/roslaunch-HP-9708.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://HP:37233/

SUMMARY
========

PARAMETERS
 * /open_manipulator/gripper_position/joint: gripper
 * /open_manipulator/gripper_position/pid/d: 10.0
 * /open_manipulator/gripper_position/pid/i: 1.0
 * /open_manipulator/gripper_position/pid/p: 1000.0
 * /open_manipulator/gripper_position/type: effort_controller...
 * /open_manipulator/gripper_sub_position/joint: gripper_sub
 * /open_manipulator/gripper_sub_position/pid/d: 10.0
 * /open_manipulator/gripper_sub_position/pid/i: 1.0
 * /open_manipulator/gripper_sub_position/pid/p: 1000.0
 * /open_manipulator/gripper_sub_position/type: effort_controller...
 * /open_manipulator/joint1_position/joint: joint1
 * /open_manipulator/joint1_position/pid/d: 10.0
 * /open_manipulator/joint1_position/pid/i: 1.0
 * /open_manipulator/joint1_position/pid/p: 1000.0
 * /open_manipulator/joint1_position/type: effort_controller...
 * /open_manipulator/joint2_position/joint: joint2
 * /open_manipulator/joint2_position/pid/d: 10.0
 * /open_manipulator/joint2_position/pid/i: 1.0
 * /open_manipulator/joint2_position/pid/p: 1000.0
 * /open_manipulator/joint2_position/type: effort_controller...
 * /open_manipulator/joint3_position/joint: joint3
 * /open_manipulator/joint3_position/pid/d: 10.0
 * /open_manipulator/joint3_position/pid/i: 1.0
 * /open_manipulator/joint3_position/pid/p: 1000.0
 * /open_manipulator/joint3_position/type: effort_controller...
 * /open_manipulator/joint4_position/joint: joint4
 * /open_manipulator/joint4_position/pid/d: 10.0
 * /open_manipulator/joint4_position/pid/i: 1.0
 * /open_manipulator/joint4_position/pid/p: 1000.0
 * /open_manipulator/joint4_position/type: effort_controller...
```

Figure 4.2: This figure illustrates the commands executed for launching the Gazebo Simulation of the Open Manipulator -X while spawning it using its URDF file.

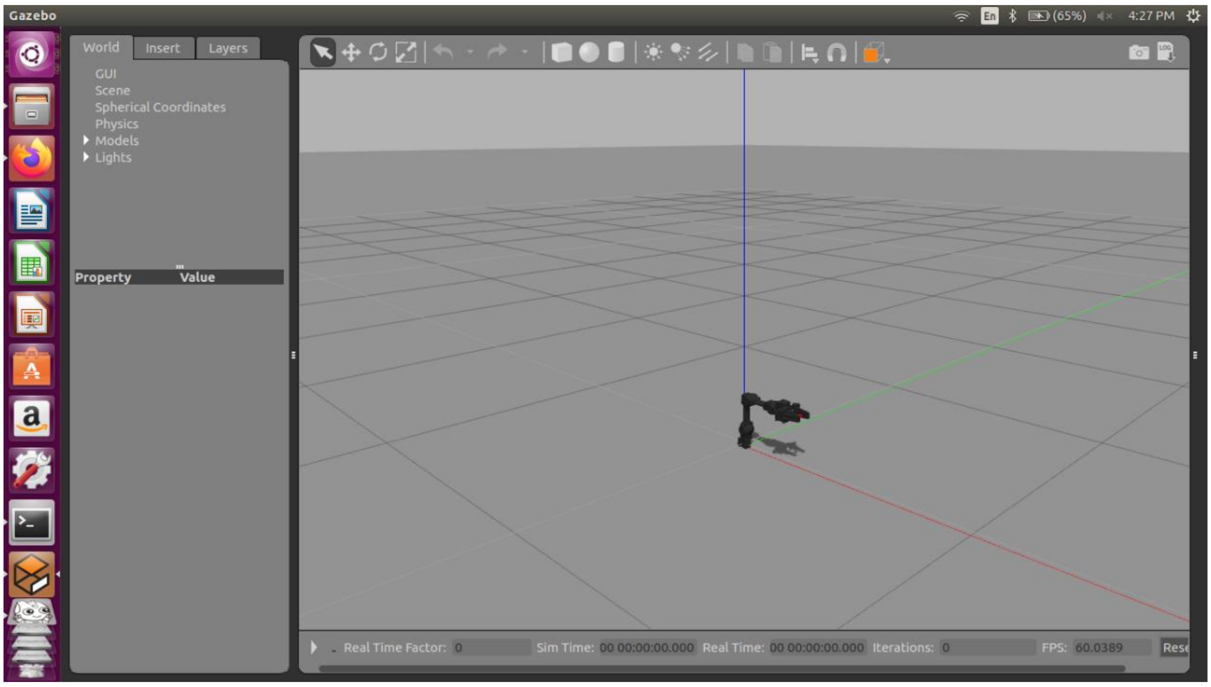
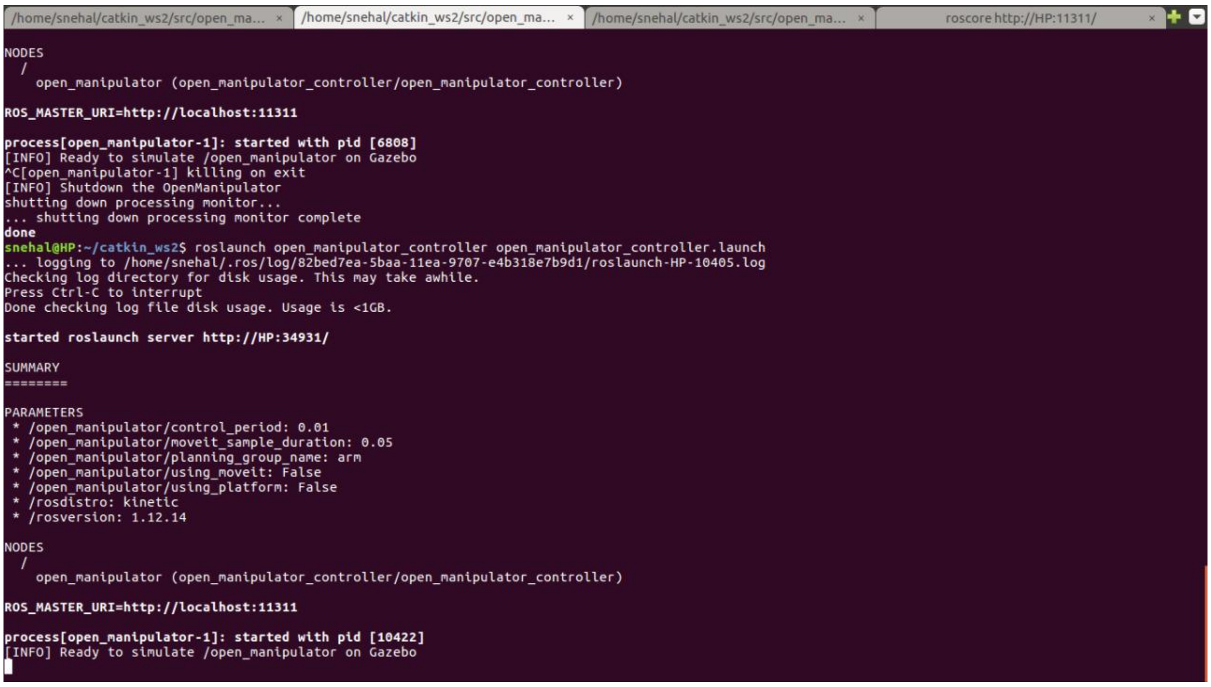


Figure 4.3: This figure illustrates the Gazebo environment and the Open Manipulator's position in it as per world coordinates.

4.1.1.3 Launching the Open_Manipulator_Controller

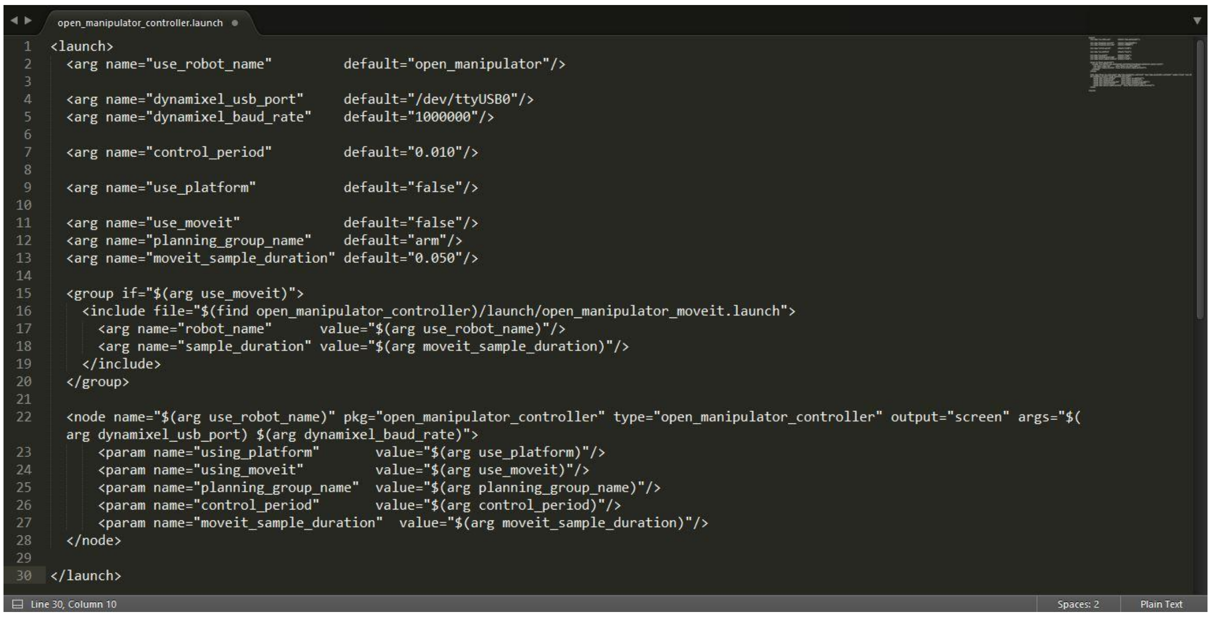
The Open_Manipulator_Controller (Github link enclosed in ‘references’ section) is a ROS (python) script that acts as an interface between the Gazebo Simulation and the Command Line application(Terminal in our case) that sends ROS messages (commands) to it.

Since, the controller is intended to relay messages to the simulation and not to the actual platform we have already set the “use_platform” qualifier to “false” in Launch File.


/home/snehal/catkin_ws2/src/open_ma...
/home/snehal/catkin_ws2/src/open_ma...
/home/snehal/catkin_ws2/src/open_ma...
roscore http://HP:11311/

```
NODES
  /
    open_manipulator (open_manipulator_controller/open_manipulator_controller)

ROS_MASTER_URI=http://localhost:11311

process[open_manipulator-1]: started with pid [6808]
[INFO] Ready to simulate /open_manipulator on Gazebo
^C[open_manipulator-1] killing on exit
[INFO] Shutdown the OpenManipulator
shutting down processing monitor...
... shutting down processing monitor complete
done
snehal@HP:~/catkin_ws2$ roslaunch open_manipulator_controller open_manipulator_controller.launch
... logging to /home/snehal/.ros/log/82bed7ea-5baa-11ea-9707-e4b318e7b9d1/roslaunch-HP-10405.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://HP:34931/

SUMMARY
========

PARAMETERS
 * /open_manipulator/control_period: 0.01
 * /open_manipulator/moveit_sample_duration: 0.05
 * /open_manipulator/planning_group_name: arm
 * /open_manipulator/using_moveit: False
 * /open_manipulator/using_platform: False
 * /rosdistro: kinetic
 * /rosversion: 1.12.14

NODES
  /
    open_manipulator (open_manipulator_controller/open_manipulator_controller)

ROS_MASTER_URI=http://localhost:11311

process[open_manipulator-1]: started with pid [10422]
[INFO] Ready to simulate /open_manipulator on Gazebo
```

Figure 4.4: Run Time View of The launched Open_Manipulator_Controller package in terminal

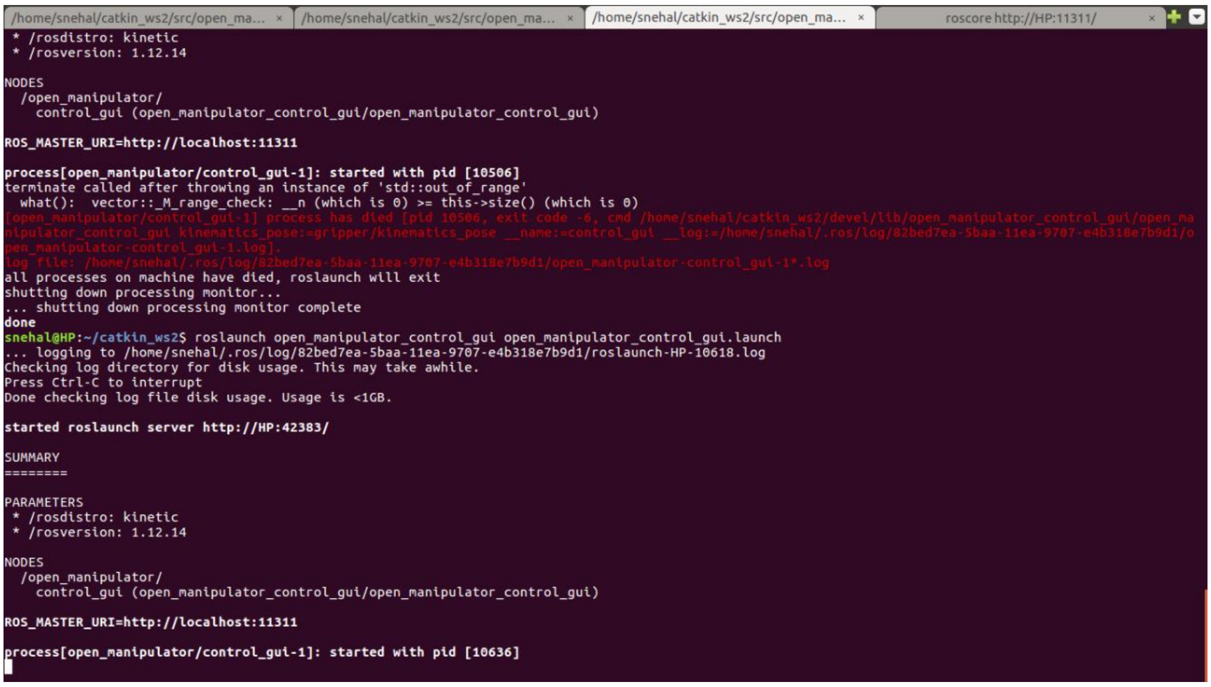


```
<launch>
  <arg name="use_robot_name"         default="open_manipulator"/>

  <arg name="dynamixel_usb_port"     default="/dev/ttyUSB0"/>
  <arg name="dynamixel_baud_rate"    default="1000000"/>

  <arg name="control_period"         default="0.010"/>

  <arg name="use_platform"           default="false"/>

  <arg name="use_moveit"             default="false"/>
  <arg name="planning_group_name"    default="arm"/>
  <arg name="moveit_sample_duration" default="0.050"/>

  <group if="$(arg use_moveit)">
    <include file="$(find open_manipulator_controller)/launch/open_manipulator_moveit.launch">
      <arg name="robot_name"      value="$(arg use_robot_name)"/>
      <arg name="sample_duration" value="$(arg moveit_sample_duration)"/>
    </include>
  </group>

  <node name="$(arg use_robot_name)" pkg="open_manipulator_controller" type="open_manipulator_controller" output="screen" args="$(
  arg dynamixel_usb_port) $(arg dynamixel_baud_rate)">
      <param name="using_platform"       value="$(arg use_platform)"/>
      <param name="using_moveit"         value="$(arg use_moveit)"/>
      <param name="planning_group_name"  value="$(arg planning_group_name)"/>
      <param name="control_period"       value="$(arg control_period)"/>
      <param name="moveit_sample_duration"  value="$(arg moveit_sample_duration)"/>
  </node>

</launch>
```

Figure 4.5: This figure show the modification, we have made to the open_manipulator_controller’s launch file. This modification enables the controller to relay messages to the simulation

4.1.1.4 The open_manipulator_control_gui

This is a GUI tool that allows us to command task-space and joint space coordinates as well as angles to the manipulator and its simulation. This gets reflected in the simulation in real time. The GUI also provides ROS services to guage feedback in real-time.



```
 * /rosdistro: kinetic
 * /rosversion: 1.12.14

NODES
  /open_manipulator/
    control_gui (open_manipulator_control_gui/open_manipulator_control_gui)

ROS_MASTER_URI=http://localhost:11311

process[open_manipulator/control_gui-1]: started with pid [10506]
terminate called after throwing an instance of 'std::out_of_range'
  what():  vector::_M_range_check: __n (which is 0) >= this->size() (which is 0)
[open_manipulator/control_gui-1] process has died [pid 10506, exit code -6, cmd /home/snehal/catkin_ws2/devel/lib/open_manipulator_control_gui/open_ma
nipulator_control_gui kinematics_pose:=gripper/kinematics_pose __name:=control_gui __log:=/home/snehal/.ros/log/82bed7ea-5baa-11ea-9707-e4b318e7b9d1/o
pen_manipulator-control_gui-1.log].
log file: /home/snehal/.ros/log/82bed7ea-5baa-11ea-9707-e4b318e7b9d1/open_manipulator-control_gui-1*.log
all processes on machine have died, roslaunch will exit
shutting down processing monitor...
... shutting down processing monitor complete
done
snehal@HP:~/catkin_ws2$ roslaunch open_manipulator_control_gui open_manipulator_control_gui.launch
... logging to /home/snehal/.ros/log/82bed7ea-5baa-11ea-9707-e4b318e7b9d1/roslaunch-HP-10618.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://HP:42383/

SUMMARY
========

PARAMETERS
 * /rosdistro: kinetic
 * /rosversion: 1.12.14

NODES
  /open_manipulator/
    control_gui (open_manipulator_control_gui/open_manipulator_control_gui)

ROS_MASTER_URI=http://localhost:11311

process[open_manipulator/control_gui-1]: started with pid [10636]
```

Figure. 4.6: Terminal Window depicting the launched GUI Controller node.

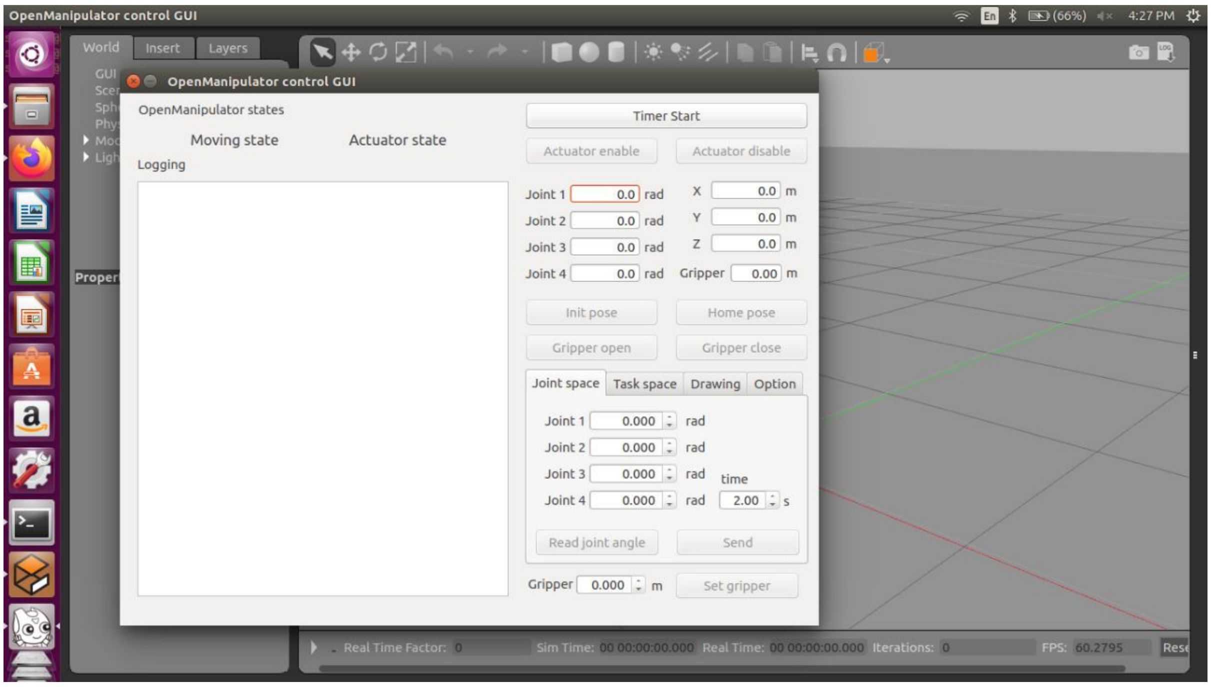
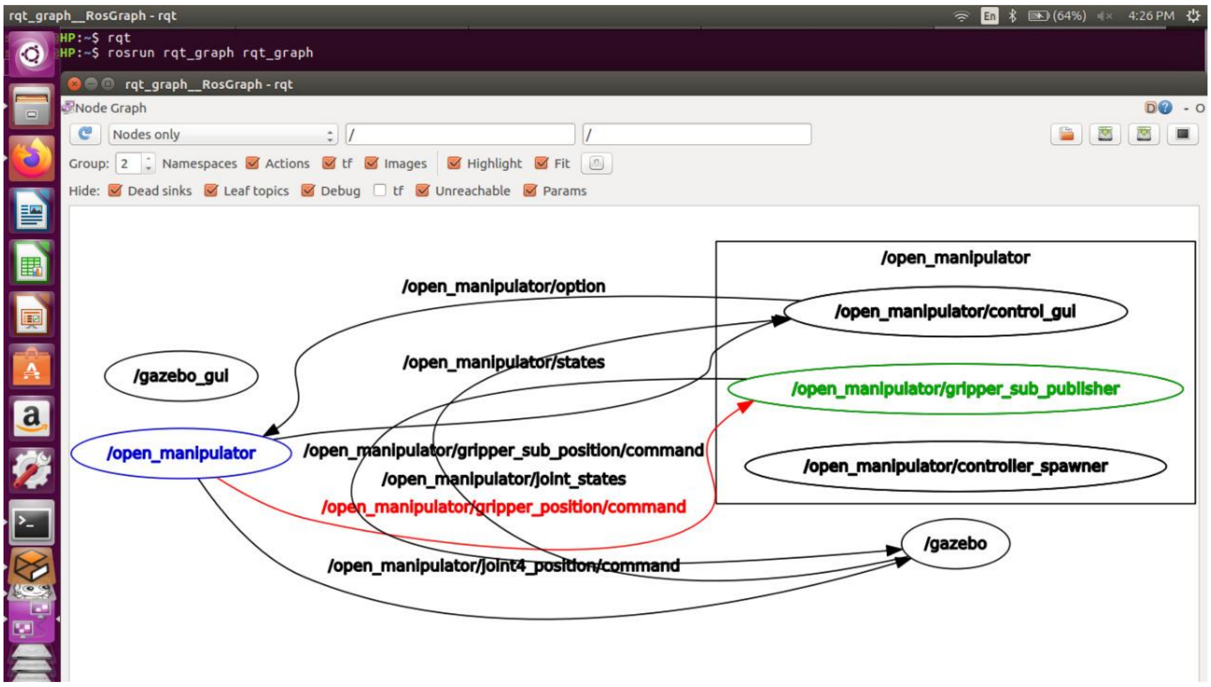


Figure. 4.7: We illustrate the GUI controller, its interface, and the type of ROS messages it relays in this figure.

4.2.2 Analysing the Simulation



```
HP:~$ rqt
HP:~$ rosrun rqt_graph rqt_graph
```

Group: 2 ☑ Namespaces ☑ Actions ☑ tf ☑ Images ☑ Highlight ☑ Fit
Hide: ☑ Dead sinks ☑ Leaf topics ☑ Debug ☐ tf ☑ Unreachable ☑ Params

rqt_graph__RosGraph - rqt
Node Graph
Nodes only
/open_manipulator
/open_manipulator/option
/open_manipulator/control_gui
/gazebo_gui
/open_manipulator/states
/open_manipulator/gripper_sub_publisher
/open_manipulator
/open_manipulator/gripper_sub_position/command
/open_manipulator/controller_spawner
/open_manipulator/joint_states
/open_manipulator/gripper_position/command
/gazebo
/open_manipulator/joint4_position/command

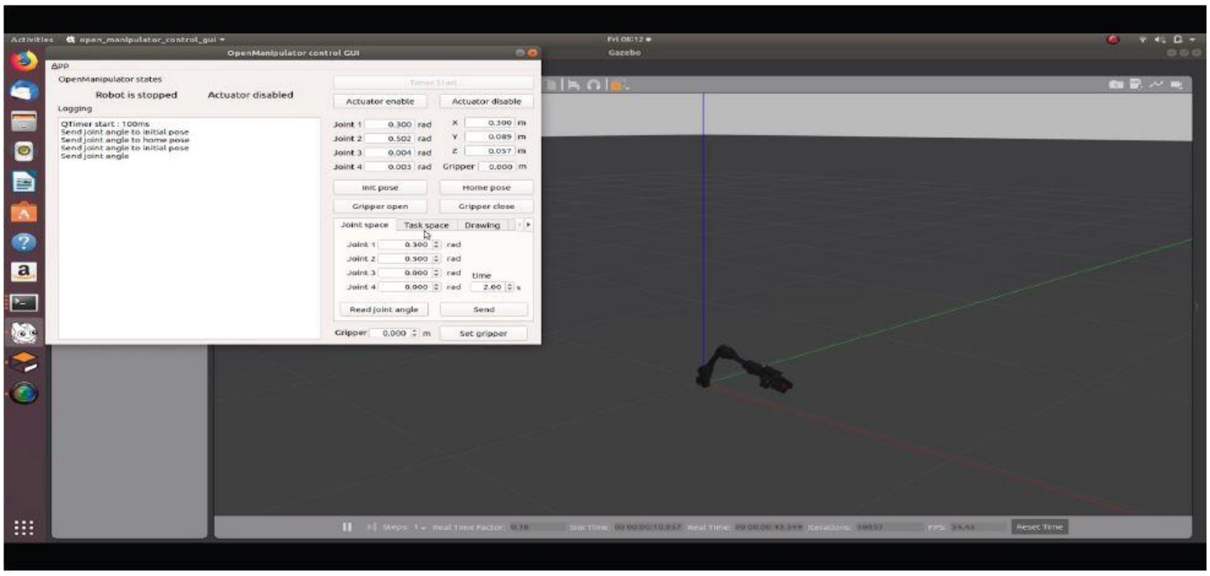
Figure 4.8: rqt_graph depicting the various ROS nodes, their topics and messages being relayed in the simulation

The nodes “/gazebo_manipulator”, “/gazebo_gui” and, “/open_manipulator” depict the real-world OpenCR{U2D2 Controller of the manipulator}, The GUI controller and, the simulation object in the Gazebo environment. The significance of each topic is explained as follows:

i. /Open_manipulator/joint_states: Commands on this topic are used to control the motion of each individual link in the Open Manipulator.
ii. /Open_manipulator/Gripper_position/command: This topic relays commands that open or close the gripper.
iii. /Open_manipulator/states/: Controls the coordinates of the base of the manipulator in the simulation environment
iv. Other topics are essential for behind-the-scenes operations that are handled by ROS.

4.1.3 Running the simulation

The following figure shows a running simulation wherein messages are being published through the GUI controller and is being realized by the simulated manipulator. This simulation was executed on Lab PC.



OpenManipulator states

Robot is stopped   Actuator disabled

Logging

```
QTimer start : 100ms
Send joint angle to initial pose
Send joint angle to home pose
Send joint angle to initial pose
Send joint angle
```

Joint 1	0.300 rad	X	0.300 m		
Joint 2	0.502 rad	Y	0.089 m		
Joint 3	0.004 rad	Z	0.057 m		
Joint 4	0.003 rad	Gripper	0.000 m		

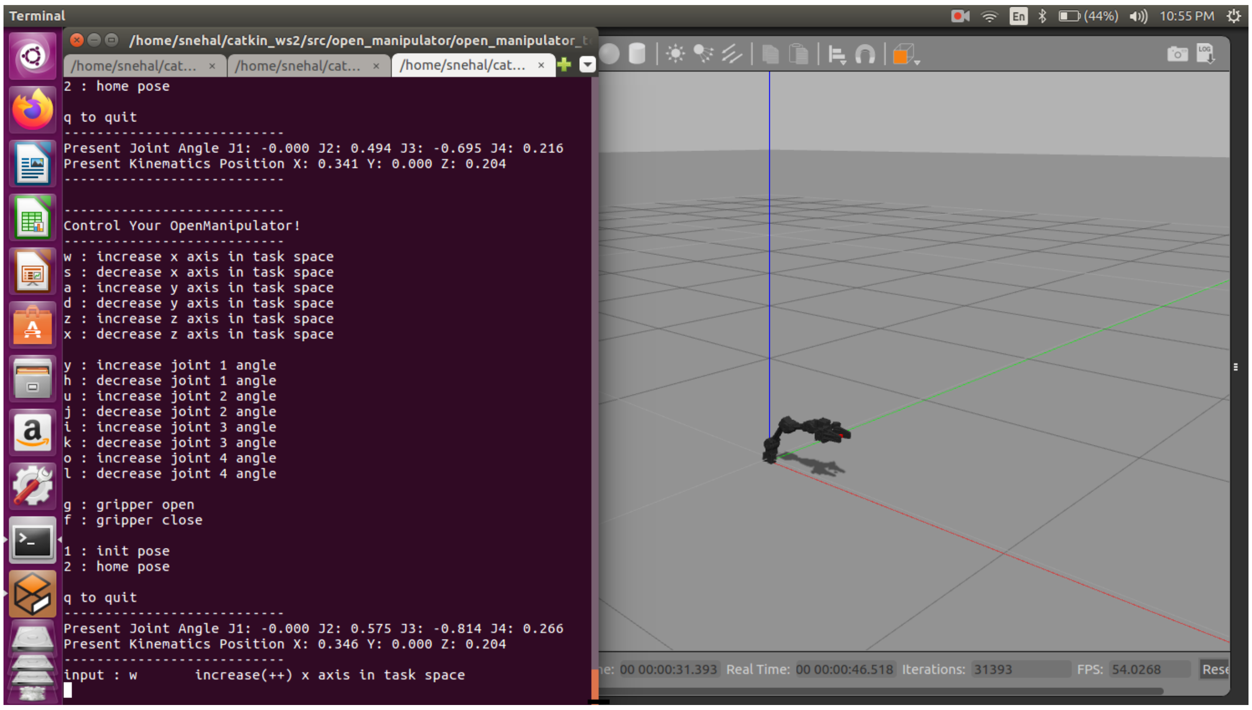
Joint 1	0.300 rad	
Joint 2	0.500 rad	
Joint 3	0.000 rad	time
Joint 4	0.000 rad	2.00 s

Gripper 0.000 m

Figure 4.9: The GUI control running in Simulation

4.2. Tele-Operation

In this section, present a simulation scenario in which we control motion of the Open Manipulator's end-effector along specific directions using standard keyboard commands. The ROS node/topic structure for this simulation has already been discussed in section 3.2.3. The following figure shows the running simulation:



```
2 : home pose

q to quit
---------------------------
Present Joint Angle J1: -0.000 J2: 0.494 J3: -0.695 J4: 0.216
Present Kinematics Position X: 0.341 Y: 0.000 Z: 0.204
---------------------------

---------------------------
Control Your OpenManipulator!
---------------------------
w : increase x axis in task space
s : decrease x axis in task space
a : increase y axis in task space
d : decrease y axis in task space
z : increase z axis in task space
x : decrease z axis in task space

y : increase joint 1 angle
h : decrease joint 1 angle
u : increase joint 2 angle
j : decrease joint 2 angle
i : increase joint 3 angle
k : decrease joint 3 angle
o : increase joint 4 angle
l : decrease joint 4 angle

g : gripper open
f : gripper close

1 : init pose
2 : home pose

q to quit
---------------------------
Present Joint Angle J1: -0.000 J2: 0.575 J3: -0.814 J4: 0.266
Present Kinematics Position X: 0.346 Y: 0.000 Z: 0.204
---------------------------
input : w       increase(++) x axis in task space
```

Terminal
/home/snehal/catkin_ws2/src/open_manipulator/open_manipulator_
/home/snehal/cat...
/home/snehal/cat...
/home/snehal/cat...
10:55 PM
e: 00 00:00:31.393 Real Time: 00 00:00:46.518 Iterations: 31393 FPS: 54.0268

Fig. 4.10.: Run-time view of the Tele-Operation Implementation in Gazebo

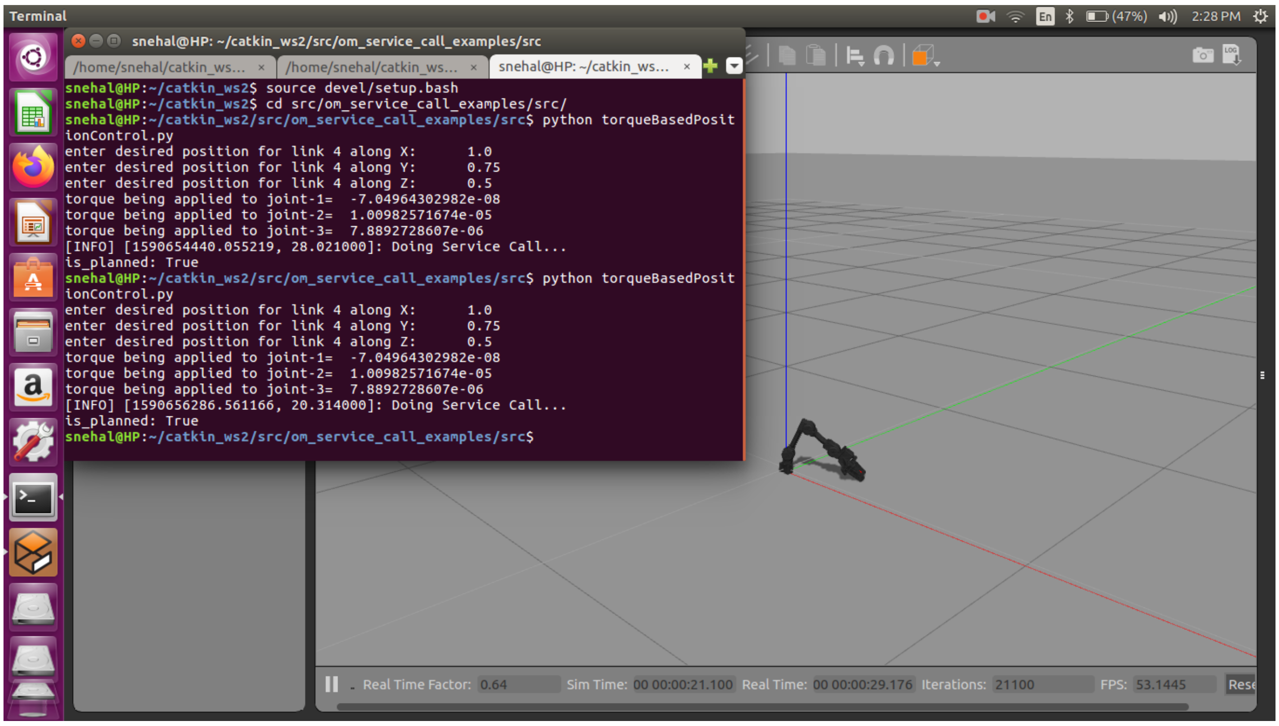
It is evident in the above snapshot that the simulation is responding to specific commands given via keyboard. Commands for performing different operations can also be seen I the figure.

4.3. Performance Analysis of the Torque based position controller

Having provided a detailed analysis of the controller's conceptualization and mathematical formulation in section 3.3, we present an analysis of its performance in simulation in this section.

4.3.1 The Output

The following image shows a run-time view of *Torque based Position Controller* in Terminal and Gazebo:


Terminal
snehal@HP: ~/catkin_ws2/src/om_service_call_examples/src
/home/snehal/catkin_ws...
/home/snehal/catkin_ws...
snehal@HP: ~/catkin_ws...
2:28 PM
Real Time Factor: 0.64
Sim Time: 00 00:00:21.100
Real Time: 00 00:00:29.176
Iterations: 21100
FPS: 53.1445

```
snehal@HP:~/catkin_ws2$ source devel/setup.bash
snehal@HP:~/catkin_ws2$ cd src/om_service_call_examples/src/
snehal@HP:~/catkin_ws2/src/om_service_call_examples/src$ python torqueBasedPosit
ionControl.py
enter desired position for link 4 along X:      1.0
enter desired position for link 4 along Y:      0.75
enter desired position for link 4 along Z:      0.5
torque being applied to joint-1=  -7.04964302982e-08
torque being applied to joint-2=  1.00982571674e-05
torque being applied to joint-3=  7.8892728607e-06
[INFO] [1590654440.055219, 28.021000]: Doing Service Call...
is_planned: True
snehal@HP:~/catkin_ws2/src/om_service_call_examples/src$ python torqueBasedPosit
ionControl.py
enter desired position for link 4 along X:      1.0
enter desired position for link 4 along Y:      0.75
enter desired position for link 4 along Z:      0.5
torque being applied to joint-1=  -7.04964302982e-08
torque being applied to joint-2=  1.00982571674e-05
torque being applied to joint-3=  7.8892728607e-06
[INFO] [1590656286.561166, 20.314000]: Doing Service Call...
is_planned: True
snehal@HP:~/catkin_ws2/src/om_service_call_examples/src$
```

Fig. 4.11: Run-time view of the Torque-Based Position Controller (Realised as per).

The above output was recorded for {Kp = Kd = 1e-(05)}. These values were optimized empirically based on the output plots and their stability analysis. In this simulation, the controller accepts position commands from the user, converts it torque commands and realises a position based on it (refer section 3.3.1). The following figure illustrates the architecture of ROS nodes/topics for this simulation:

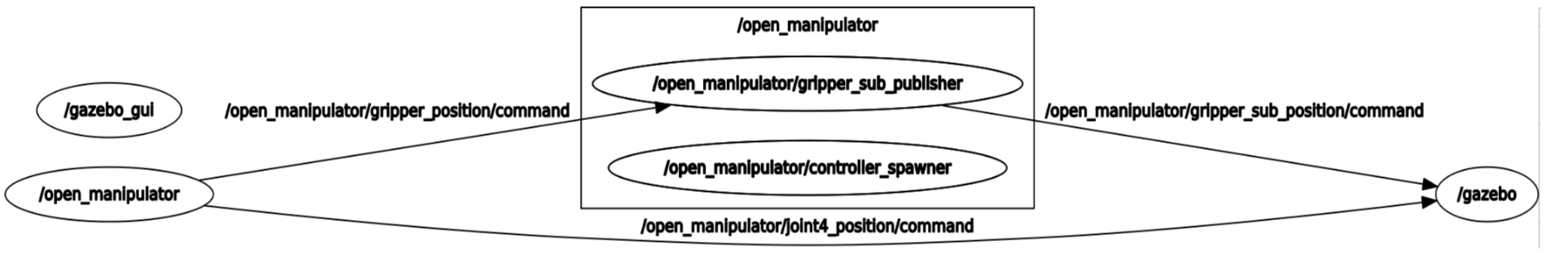

/open_manipulator
/open_manipulator/gripper_sub_publisher
/open_manipulator/controller_spawner
/gazebo_gui
/open_manipulator
/open_manipulator/gripper_position/command
/open_manipulator/gripper_sub_position/command
/open_manipulator/joint4_position/command
/gazebo

Fig. 4.12: ROS nodes/topics for the torque based position controller.

Here, /open_manipulator acts as the input node and communicates with simulation (/gazebo) via the services "/open_manipulator/joint4_position/command" and "/open_manipulator/gripper_sub_publisher".

The program for this implementation has been appended in Appendix-A.

4.3.2 Performance Analysis of the Controller

The following plots depict the variation in position along different joints with respect to time. These plots represent data recorded for the case shown in Fig. 4.11 that is the commanded positions are as follows:

1. Position along X (Joint-1): 1.0 rad
2. Position along Y (Joint-2): 0.75 rad
3. Position along Z (Joint-3): 0.50 rad

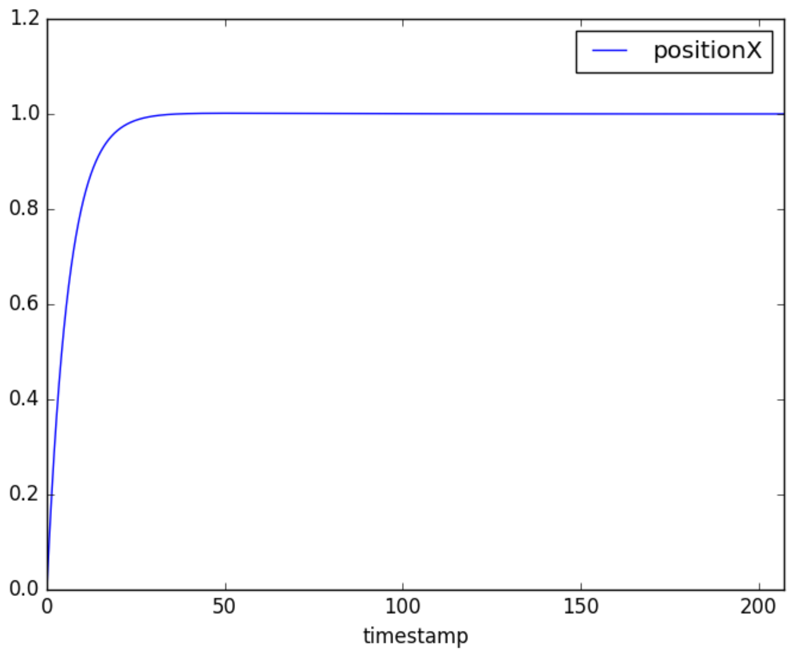


Fig. 4.13: Position along Joint-1 V/S time in seconds

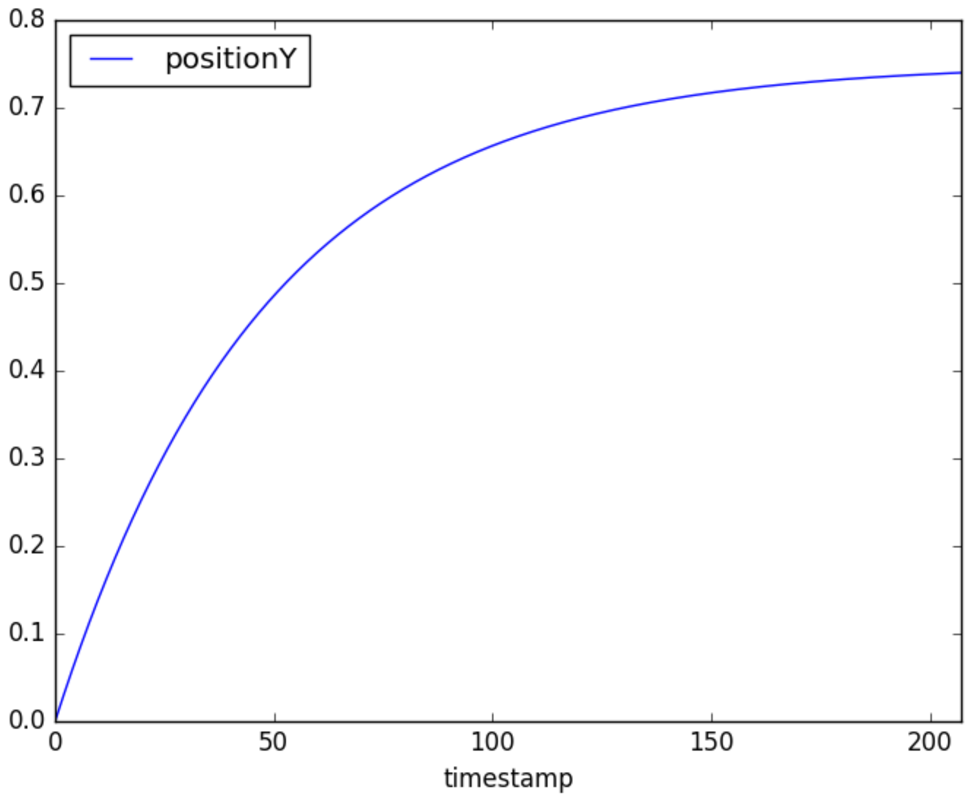


Fig. 4.14. Position along Joint-2 V/S time in seconds

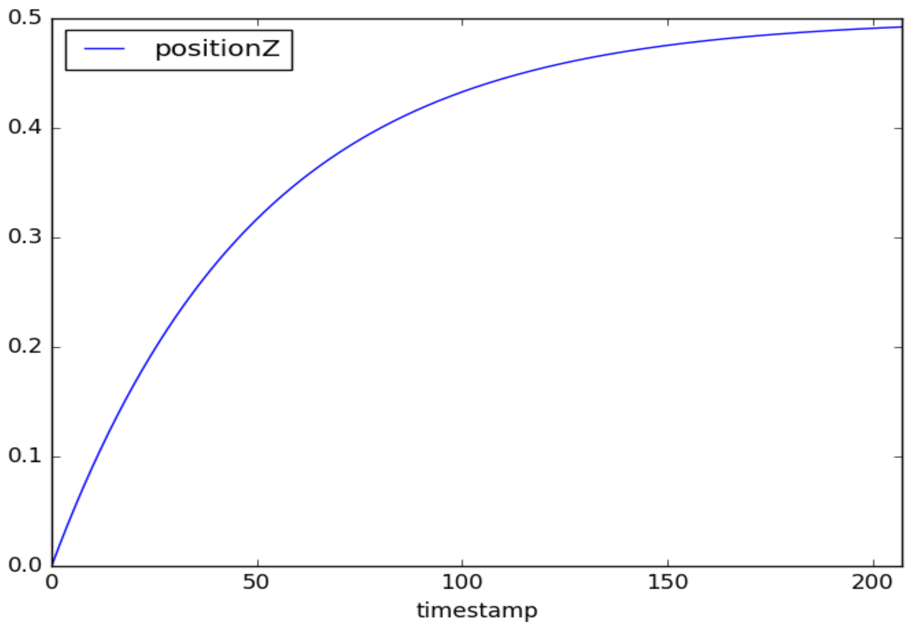


Fig. 4.15. Position along Joint-3 V/S time in seconds

Controller output data for multiple iterations in this test case has been appended in Appendix-B.

In figures 4.13 through 4.15, it can be observed that the realized position controller saturates position along all three joints well within the tolerance bounds of the commanded position. Aside from that, the time-domain response resembles that of an **under-damped** ($0 < \varepsilon < 1$) **system**, thus ensuring that the system's impulse response is absolutely summable and hence making the system **BIBO stable**.

4.4. Summary

To summarise, we have presented all our implementations in this section along with a critical analysis of results.

CHAPTER 5
CONCLUSION AND FUTURE SCOPE OF WORK

5.1 Summary of work done

The following **key outcomes** have been achieved:

i. We have developed a basic understanding of the Open Manipulator, its supporting software, its forward and inverse kinematics and its dynamics.
ii. We have Implemented standard position control for the Open Manipulator's simulation using the ROBOTIS GUI package
iii. We have Implemented Tele operation of the simulation using Keyboard Commands
iv. We have implemented a Torque Based Position Controller for the simulation and have analysed its performance using its response to control input. The controller performed satisfactorily and was found to be a BIBO stable system.

5.2 Future Work

Since, we have achieved all objectives pertaining to software implementation, we aim to concentrate on the hardware implementation of this project in near future. The tasks cut-out shall be:

i. Control of the Open Manipulator(Hardware) using ROS packages
ii. Operating the Dynamixel XM-430 motors in current-control mode
iii. Obtaining Current Feedback from the Dynamixel XM-430 motors
iv. Deploying current feedback based torque controller for each individual joint.

5.3 Conclusion

We have successfully demonstrated torque based position control for the Open Manipulator-X in its Gazebo Simulation. The realized system is **Stable**. Since the Open Manipulator – X is an otherwise position controlled manipulator, hardware implementation of this work would pave way for the implementation of a variety of modern control algorithms on it, thus increasing its utility in industrial automation applications.

REFERENCES

[1] *Automatic Control Systems by B.C. Kuo*
[2] *http://emanual.robotis.com/docs/en/platform/openmanipulator_x/overview/#overview*
[3] *http://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_sdk/overview/*
[4] *http://gazebosim.org/tutorials*
[5] *http://www.robotis.us/dynamixel-xm430-w350-r/*
[6] *http://wiki.ros.org/ROS/Tutorials*
[7] *Introduction to Robotics by John J. Craig*
[8] *O. Khatib, P. Thaulad, Taizo Yoshikawa and J. Park, "Torque-position transformer for task control of position controlled robots," 2008 IEEE International Conference on Robotics and Automation, Pasadena, CA, 2008, pp. 1729-1734.*
[9] *Ricardo P. Aguilera et al, Control of Power Electronic Components and Systems, 2018*
[10] *http://wiki.ros.org/*

APPENDIX – A

The Python Program for implementation of *Torque Based Position Control on the Open Manipulator's Gazebo simulation is given as below:*

```
import numpy as np
from math import exp, expm1
import matplotlib as plt
import csv
import rospy
import sys
from open_manipulator_msgs.srv import SetJointPosition, SetJointPositionRequest
rospy.init_node('service_set_joint_position_client')
rospy.wait_for_service('/open_manipulator/goal_joint_space_path')
goal_joint_space_path_service_client =
rospy.ServiceProxy('/open_manipulator/goal_joint_space_path', SetJointPosition)
goal_joint_space_path_request_object = SetJointPositionRequest()

pos_prev = np.array([[0.0],[0.0],[0.0]])
pos_new  = np.array([[0.0],[0.0],[0.0]])
error = np.array([[100.0],[100.0],[100.0]])
inertia = np.array([[3.0654178e-05, -1.2764155e-06, -2.6874417e-07],[0, 2.4230292e-04,
1.1559550e-08],[0, 0 , 2.5155057e-04]])
inertia_inv = np.linalg.inv(inertia)
cycleTime = 0.01
kp = 1e-05
kd = 1e-05
time = 0.0
x = raw_input("enter desired position for link 4 along X:      ")
y = raw_input("enter desired position for link 4 along Y:      ")
z = raw_input("enter desired position for link 4 along Z:      ")
pos_x = float(x)
pos_y = float(y)
pos_z = float(z)
pos_desired = np.array([[pos_x],[pos_y],[pos_z]])

positionX = 0.0
positionY = 0.0
positionZ = 0.0
```

```
with open('error.csv', 'w') as f:
        fieldnames = ['timestamp','ex','ey','ez','positionX', 'positionY', 'positionZ']
        theWriter = csv.DictWriter(f, fieldnames = fieldnames)
        theWriter.writeheader()
        while(abs(error[0]) >= 0.01 or abs(error[1]) >= 0.01 or abs(error[2]) >= 0.01):
                ex = float(error[0])
                ey = float(error[1])
                ez = float(error[2])
                positionX = float(pos_prev[0])
                positionY = float(pos_prev[1])
                positionZ = float(pos_prev[2])

                theWriter.writerow({'timestamp': time,'ex': ex, 'ey': ey, 'ez' : ez, 'positionX':
                positionX, 'positionY': positionY, 'positionZ': positionZ})
                error = np.subtract(pos_desired, pos_prev)
                torque = kp*error + kd*(error/cycleTime)
                alpha = np.dot(inertia_inv, torque)
                pos_new = pos_prev + (alpha*cycleTime*cycleTime)/2.0;
                pos_prev = pos_new
                time = time + cycleTime
print "torque being applied to joint-1= ", float(torque[0])
print "torque being applied to joint-2= ", float(torque[1])
print "torque being applied to joint-3= ", float(torque[2])
goal_joint_space_path_request_object.planning_group = 'arm'
goal_joint_space_path_request_object.joint_position.joint_name = ['joint1', 'joint2', 'joint3',
'joint4']
goal_joint_space_path_request_object.joint_position.position = [positionX, positionY,
positionZ, 0.0]
goal_joint_space_path_request_object.joint_position.max_accelerations_scaling_factor = 1.0
goal_joint_space_path_request_object.joint_position.max_velocity_scaling_factor = 1.0
goal_joint_space_path_request_object.path_time = 2.0

rospy.loginfo("Doing Service Call...")
result = goal_joint_space_path_service_client(goal_joint_space_path_request_object)
print result
```

APPENDIX - B

Control Output Data for initial few iterations of the torque based position controller for the test case as discussed in section 4.4.1 is as follows:

Here Position and error are in radians ad time has been recorded in seconds:

timestamp	error(x)	eroor(y)	error(z)	positionX	positionY	positionZ
0	100	100	100	0	0	0
0.01	1	0.2	0.4	0.00165	4.17E-05	8.03E-05
0.02	0.99835	0.199958	0.39992	0.003297	8.34E-05	0.000161
0.03	0.996703	0.199917	0.399839	0.004941	0.000125	0.000241
0.04	0.995059	0.199875	0.399759	0.006583	0.000167	0.000321
0.05	0.993417	0.199833	0.399679	0.008222	0.000208	0.000401
0.06	0.991778	0.199792	0.399599	0.009858	0.00025	0.000482
0.07	0.990142	0.19975	0.399518	0.011492	0.000292	0.000562
0.08	0.988508	0.199708	0.399438	0.013123	0.000333	0.000642
0.09	0.986877	0.199667	0.399358	0.014751	0.000375	0.000722
0.1	0.985249	0.199625	0.399278	0.016377	0.000416	0.000802
0.11	0.983623	0.199584	0.399198	0.018	0.000458	0.000882
0.12	0.982	0.199542	0.399118	0.01962	0.0005	0.000963
0.13	0.98038	0.1995	0.399037	0.021237	0.000541	0.001043
0.14	0.978763	0.199459	0.398957	0.022852	0.000583	0.001123
0.15	0.977148	0.199417	0.398877	0.024464	0.000624	0.001203
0.16	0.975536	0.199376	0.398797	0.026074	0.000666	0.001283
0.17	0.973926	0.199334	0.398717	0.027681	0.000707	0.001363
0.18	0.972319	0.199293	0.398637	0.029285	0.000749	0.001443
0.19	0.970715	0.199251	0.398557	0.030887	0.00079	0.001523
0.2	0.969113	0.19921	0.398477	0.032486	0.000832	0.001603
0.21	0.967514	0.199168	0.398397	0.034082	0.000873	0.001683
0.22	0.965918	0.199127	0.398317	0.035676	0.000915	0.001763
0.23	0.964324	0.199085	0.398237	0.037267	0.000956	0.001843
0.24	0.962733	0.199044	0.398157	0.038855	0.000998	0.001923
0.25	0.961145	0.199002	0.398077	0.040441	0.001039	0.002003
0.26	0.959559	0.198961	0.397997	0.042024	0.001081	0.002083
0.27	0.957976	0.198919	0.397917	0.043605	0.001122	0.002163
0.28	0.956395	0.198878	0.397837	0.045183	0.001164	0.002242
0.29	0.954817	0.198836	0.397758	0.046758	0.001205	0.002322
0.3	0.953242	0.198795	0.397678	0.048331	0.001247	0.002402
0.31	0.951669	0.198753	0.397598	0.049901	0.001288	0.002482
0.32	0.950099	0.198712	0.397518	0.051469	0.001329	0.002562
0.33	0.948531	0.198671	0.397438	0.053034	0.001371	0.002641
0.34	0.946966	0.198629	0.397359	0.054596	0.001412	0.002721
0.35	0.945404	0.198588	0.397279	0.056156	0.001454	0.002801
0.36	0.943844	0.198546	0.397199	0.057713	0.001495	0.002881

0.37	0.942287	0.198505	0.397119	0.059268	0.001536	0.00296
0.38	0.940732	0.198464	0.39704	0.06082	0.001578	0.00304
0.39	0.93918	0.198422	0.39696	0.06237	0.001619	0.00312
0.4	0.93763	0.198381	0.39688	0.063917	0.00166	0.0032
0.41	0.936083	0.19834	0.3968	0.065462	0.001702	0.003279
0.42	0.934538	0.198298	0.396721	0.067004	0.001743	0.003359
0.43	0.932996	0.198257	0.396641	0.068543	0.001784	0.003438
0.44	0.931457	0.198216	0.396562	0.07008	0.001826	0.003518
0.45	0.92992	0.198174	0.396482	0.071614	0.001867	0.003598
0.46	0.928386	0.198133	0.396402	0.073146	0.001908	0.003677
0.47	0.926854	0.198092	0.396323	0.074676	0.00195	0.003757
0.48	0.925324	0.19805	0.396243	0.076202	0.001991	0.003836
0.49	0.923798	0.198009	0.396164	0.077727	0.002032	0.003916
0.5	0.922273	0.197968	0.396084	0.079248	0.002073	0.003995
0.51	0.920752	0.197927	0.396005	0.080768	0.002115	0.004075
0.52	0.919232	0.197885	0.395925	0.082284	0.002156	0.004154
0.53	0.917716	0.197844	0.395846	0.083799	0.002197	0.004234
0.54	0.916201	0.197803	0.395766	0.085311	0.002238	0.004313
0.55	0.914689	0.197762	0.395687	0.08682	0.00228	0.004393
0.56	0.91318	0.19772	0.395607	0.088327	0.002321	0.004472
0.57	0.911673	0.197679	0.395528	0.089831	0.002362	0.004552
0.58	0.910169	0.197638	0.395448	0.091333	0.002403	0.004631
0.59	0.908667	0.197597	0.395369	0.092832	0.002444	0.00471
0.6	0.907168	0.197556	0.39529	0.094329	0.002485	0.00479
0.61	0.905671	0.197515	0.39521	0.095823	0.002527	0.004869
0.62	0.904177	0.197473	0.395131	0.097315	0.002568	0.004948
0.63	0.902685	0.197432	0.395052	0.098805	0.002609	0.005028
0.64	0.901195	0.197391	0.394972	0.100292	0.00265	0.005107
0.65	0.899708	0.19735	0.394893	0.101777	0.002691	0.005186
0.66	0.898223	0.197309	0.394814	0.103259	0.002732	0.005265
0.67	0.896741	0.197268	0.394735	0.104738	0.002773	0.005345
0.68	0.895262	0.197227	0.394655	0.106216	0.002815	0.005424
0.69	0.893784	0.197185	0.394576	0.10769	0.002856	0.005503
0.7	0.89231	0.197144	0.394497	0.109163	0.002897	0.005582
0.71	0.890837	0.197103	0.394418	0.110633	0.002938	0.005662
0.72	0.889367	0.197062	0.394338	0.1121	0.002979	0.005741
0.73	0.8879	0.197021	0.394259	0.113566	0.00302	0.00582
0.74	0.886434	0.19698	0.39418	0.115028	0.003061	0.005899
0.75	0.884972	0.196939	0.394101	0.116489	0.003102	0.005978
0.76	0.883511	0.196898	0.394022	0.117946	0.003143	0.006057
0.77	0.882054	0.196857	0.393943	0.119402	0.003184	0.006136
0.78	0.880598	0.196816	0.393864	0.120855	0.003225	0.006215
0.79	0.879145	0.196775	0.393785	0.122306	0.003266	0.006294
0.8	0.877694	0.196734	0.393706	0.123754	0.003307	0.006373
0.81	0.876246	0.196693	0.393627	0.1252	0.003348	0.006453
0.82	0.8748	0.196652	0.393547	0.126644	0.003389	0.006532

0.83	0.873356	0.196611	0.393468	0.128085	0.00343	0.00661
0.84	0.871915	0.19657	0.39339	0.129524	0.003471	0.006689
0.85	0.870476	0.196529	0.393311	0.13096	0.003512	0.006768
0.86	0.86904	0.196488	0.393232	0.132394	0.003553	0.006847
0.87	0.867606	0.196447	0.393153	0.133826	0.003594	0.006926
0.88	0.866174	0.196406	0.393074	0.135255	0.003635	0.007005
0.89	0.864745	0.196365	0.392995	0.136682	0.003676	0.007084
0.9	0.863318	0.196324	0.392916	0.138107	0.003717	0.007163
0.91	0.861893	0.196283	0.392837	0.139529	0.003757	0.007242
0.92	0.860471	0.196243	0.392758	0.140949	0.003798	0.007321
0.93	0.859051	0.196202	0.392679	0.142367	0.003839	0.0074
0.94	0.857633	0.196161	0.3926	0.143782	0.00388	0.007478
0.95	0.856218	0.19612	0.392522	0.145195	0.003921	0.007557
0.96	0.854805	0.196079	0.392443	0.146605	0.003962	0.007636
0.97	0.853395	0.196038	0.392364	0.148014	0.004003	0.007715
0.98	0.851986	0.195997	0.392285	0.14942	0.004044	0.007793
0.99	0.85058	0.195956	0.392207	0.150823	0.004084	0.007872
1	0.849177	0.195916	0.392128	0.152225	0.004125	0.007951
1.01	0.847775	0.195875	0.392049	0.153624	0.004166	0.00803
1.02	0.846376	0.195834	0.39197	0.15502	0.004207	0.008108
1.03	0.84498	0.195793	0.391892	0.156415	0.004248	0.008187
1.04	0.843585	0.195752	0.391813	0.157807	0.004288	0.008266
1.05	0.842193	0.195712	0.391734	0.159197	0.004329	0.008344
1.06	0.840803	0.195671	0.391656	0.160584	0.00437	0.008423
1.07	0.839416	0.19563	0.391577	0.161969	0.004411	0.008502
1.08	0.838031	0.195589	0.391498	0.163352	0.004452	0.00858
1.09	0.836648	0.195548	0.39142	0.164733	0.004492	0.008659
1.1	0.835267	0.195508	0.391341	0.166112	0.004533	0.008737
1.11	0.833888	0.195467	0.391263	0.167488	0.004574	0.008816
1.12	0.832512	0.195426	0.391184	0.168862	0.004615	0.008894
1.13	0.831138	0.195385	0.391106	0.170233	0.004655	0.008973
1.14	0.829767	0.195345	0.391027	0.171603	0.004696	0.009051
1.15	0.828397	0.195304	0.390949	0.17297	0.004737	0.00913
1.16	0.82703	0.195263	0.39087	0.174334	0.004777	0.009208
1.17	0.825666	0.195223	0.390792	0.175697	0.004818	0.009287
1.18	0.824303	0.195182	0.390713	0.177057	0.004859	0.009365
1.19	0.822943	0.195141	0.390635	0.178416	0.004899	0.009444
1.2	0.821584	0.195101	0.390556	0.179771	0.00494	0.009522
1.21	0.820229	0.19506	0.390478	0.181125	0.004981	0.0096
1.22	0.818875	0.195019	0.3904	0.182476	0.005021	0.009679
1.23	0.817524	0.194979	0.390321	0.183826	0.005062	0.009757
1.24	0.816174	0.194938	0.390243	0.185173	0.005103	0.009835
1.25	0.814827	0.194897	0.390165	0.186517	0.005143	0.009914
1.26	0.813483	0.194857	0.390086	0.18786	0.005184	0.009992
1.27	0.81214	0.194816	0.390008	0.1892	0.005224	0.01007
1.28	0.8108	0.194776	0.38993	0.190538	0.005265	0.010149

1.29	0.809462	0.194735	0.389851	0.191874	0.005306	0.010227
1.3	0.808126	0.194694	0.389773	0.193208	0.005346	0.010305
1.31	0.806792	0.194654	0.389695	0.194539	0.005387	0.010383
1.32	0.805461	0.194613	0.389617	0.195869	0.005427	0.010462
1.33	0.804131	0.194573	0.389538	0.197196	0.005468	0.01054
1.34	0.802804	0.194532	0.38946	0.198521	0.005508	0.010618
1.35	0.801479	0.194492	0.389382	0.199843	0.005549	0.010696
1.36	0.800157	0.194451	0.389304	0.201164	0.005589	0.010774
1.37	0.798836	0.194411	0.389226	0.202482	0.00563	0.010853
1.38	0.797518	0.19437	0.389147	0.203798	0.00567	0.010931
1.39	0.796202	0.19433	0.389069	0.205113	0.005711	0.011009
1.4	0.794887	0.194289	0.388991	0.206424	0.005751	0.011087
1.41	0.793576	0.194249	0.388913	0.207734	0.005792	0.011165
1.42	0.792266	0.194208	0.388835	0.209042	0.005832	0.011243
1.43	0.790958	0.194168	0.388757	0.210347	0.005873	0.011321
1.44	0.789653	0.194127	0.388679	0.21165	0.005913	0.011399
1.45	0.78835	0.194087	0.388601	0.212951	0.005954	0.011477
1.46	0.787049	0.194046	0.388523	0.21425	0.005994	0.011555
1.47	0.78575	0.194006	0.388445	0.215547	0.006035	0.011633
1.48	0.784453	0.193965	0.388367	0.216842	0.006075	0.011711
1.49	0.783158	0.193925	0.388289	0.218134	0.006115	0.011789
1.5	0.781866	0.193885	0.388211	0.219425	0.006156	0.011867
1.51	0.780575	0.193844	0.388133	0.220713	0.006196	0.011945
1.52	0.779287	0.193804	0.388055	0.221999	0.006237	0.012023
1.53	0.778001	0.193763	0.387977	0.223283	0.006277	0.012101
1.54	0.776717	0.193723	0.387899	0.224565	0.006317	0.012178
1.55	0.775435	0.193683	0.387822	0.225845	0.006358	0.012256
1.56	0.774155	0.193642	0.387744	0.227123	0.006398	0.012334
1.57	0.772877	0.193602	0.387666	0.228398	0.006438	0.012412
1.58	0.771602	0.193562	0.387588	0.229672	0.006479	0.01249
1.59	0.770328	0.193521	0.38751	0.230943	0.006519	0.012568
1.6	0.769057	0.193481	0.387432	0.232213	0.006559	0.012645
1.61	0.767787	0.193441	0.387355	0.23348	0.0066	0.012723
1.62	0.76652	0.1934	0.387277	0.234745	0.00664	0.012801
1.63	0.765255	0.19336	0.387199	0.236008	0.00668	0.012879
1.64	0.763992	0.19332	0.387121	0.237269	0.006721	0.012956
1.65	0.762731	0.193279	0.387044	0.238528	0.006761	0.013034
1.66	0.761472	0.193239	0.386966	0.239785	0.006801	0.013112
1.67	0.760215	0.193199	0.386888	0.241039	0.006841	0.013189
1.68	0.758961	0.193159	0.386811	0.242292	0.006882	0.013267
1.69	0.757708	0.193118	0.386733	0.243543	0.006922	0.013345
1.7	0.756457	0.193078	0.386655	0.244791	0.006962	0.013422
1.71	0.755209	0.193038	0.386578	0.246038	0.007002	0.0135
1.72	0.753962	0.192998	0.3865	0.247282	0.007043	0.013578
1.73	0.752718	0.192957	0.386422	0.248525	0.007083	0.013655
1.74	0.751475	0.192917	0.386345	0.249765	0.007123	0.013733

1.75	0.750235	0.192877	0.386267	0.251003	0.007163	0.01381
1.76	0.748997	0.192837	0.38619	0.25224	0.007203	0.013888
1.77	0.74776	0.192797	0.386112	0.253474	0.007244	0.013965
1.78	0.746526	0.192756	0.386035	0.254706	0.007284	0.014043
1.79	0.745294	0.192716	0.385957	0.255936	0.007324	0.01412
1.8	0.744064	0.192676	0.38588	0.257164	0.007364	0.014198
1.81	0.742836	0.192636	0.385802	0.25839	0.007404	0.014275
1.82	0.74161	0.192596	0.385725	0.259614	0.007444	0.014353
1.83	0.740386	0.192556	0.385647	0.260837	0.007485	0.01443
1.84	0.739163	0.192515	0.38557	0.262057	0.007525	0.014507
1.85	0.737943	0.192475	0.385493	0.263275	0.007565	0.014585
1.86	0.736725	0.192435	0.385415	0.264491	0.007605	0.014662
1.87	0.735509	0.192395	0.385338	0.265705	0.007645	0.01474
1.88	0.734295	0.192355	0.38526	0.266917	0.007685	0.014817
1.89	0.733083	0.192315	0.385183	0.268127	0.007725	0.014894
1.9	0.731873	0.192275	0.385106	0.269335	0.007765	0.014972
1.91	0.730665	0.192235	0.385028	0.270541	0.007805	0.015049
1.92	0.729459	0.192195	0.384951	0.271745	0.007845	0.015126
1.93	0.728255	0.192155	0.384874	0.272947	0.007885	0.015203
1.94	0.727053	0.192115	0.384797	0.274147	0.007925	0.015281
1.95	0.725853	0.192075	0.384719	0.275345	0.007965	0.015358
1.96	0.724655	0.192035	0.384642	0.276541	0.008005	0.015435
1.97	0.723459	0.191995	0.384565	0.277736	0.008045	0.015512
1.98	0.722264	0.191955	0.384488	0.278928	0.008085	0.015589
1.99	0.721072	0.191915	0.384411	0.280118	0.008125	0.015667
2	0.719882	0.191875	0.384333	0.281306	0.008165	0.015744
2.01	0.718694	0.191835	0.384256	0.282493	0.008205	0.015821
2.02	0.717507	0.191795	0.384179	0.283677	0.008245	0.015898
2.03	0.716323	0.191755	0.384102	0.284859	0.008285	0.015975
2.04	0.715141	0.191715	0.384025	0.28604	0.008325	0.016052
2.05	0.71396	0.191675	0.383948	0.287218	0.008365	0.016129
2.06	0.712782	0.191635	0.383871	0.288395	0.008405	0.016206
2.07	0.711605	0.191595	0.383794	0.28957	0.008445	0.016283
2.08	0.71043	0.191555	0.383717	0.290742	0.008485	0.01636
2.09	0.709258	0.191515	0.38364	0.291913	0.008525	0.016438
2.1	0.708087	0.191475	0.383562	0.293082	0.008565	0.016515
2.11	0.706918	0.191435	0.383485	0.294249	0.008605	0.016591
2.12	0.705751	0.191395	0.383409	0.295414	0.008645	0.016668
2.13	0.704586	0.191355	0.383332	0.296577	0.008684	0.016745
2.14	0.703423	0.191316	0.383255	0.297738	0.008724	0.016822
2.15	0.702262	0.191276	0.383178	0.298897	0.008764	0.016899
2.16	0.701103	0.191236	0.383101	0.300055	0.008804	0.016976
2.17	0.699945	0.191196	0.383024	0.30121	0.008844	0.017053
2.18	0.69879	0.191156	0.382947	0.302364	0.008884	0.01713
2.19	0.697636	0.191116	0.38287	0.303515	0.008924	0.017207
2.2	0.696485	0.191076	0.382793	0.304665	0.008963	0.017284

2.21	0.695335	0.191037	0.382716	0.305813	0.009003	0.017361
2.22	0.694187	0.190997	0.382639	0.306959	0.009043	0.017437
2.23	0.693041	0.190957	0.382563	0.308103	0.009083	0.017514
2.24	0.691897	0.190917	0.382486	0.309245	0.009123	0.017591
2.25	0.690755	0.190877	0.382409	0.310385	0.009162	0.017668
2.26	0.689615	0.190838	0.382332	0.311524	0.009202	0.017744
2.27	0.688476	0.190798	0.382256	0.31266	0.009242	0.017821
2.28	0.68734	0.190758	0.382179	0.313795	0.009282	0.017898
2.29	0.686205	0.190718	0.382102	0.314928	0.009321	0.017975
2.3	0.685072	0.190679	0.382025	0.316059	0.009361	0.018051
2.31	0.683941	0.190639	0.381949	0.317188	0.009401	0.018128
2.32	0.682812	0.190599	0.381872	0.318315	0.009441	0.018205
2.33	0.681685	0.190559	0.381795	0.31944	0.00948	0.018281
2.34	0.68056	0.19052	0.381719	0.320564	0.00952	0.018358
2.35	0.679436	0.19048	0.381642	0.321685	0.00956	0.018435
2.36	0.678315	0.19044	0.381565	0.322805	0.009599	0.018511
2.37	0.677195	0.190401	0.381489	0.323923	0.009639	0.018588
2.38	0.676077	0.190361	0.381412	0.325039	0.009679	0.018664
2.39	0.674961	0.190321	0.381336	0.326153	0.009718	0.018741
2.4	0.673847	0.190282	0.381259	0.327266	0.009758	0.018817
2.41	0.672734	0.190242	0.381183	0.328376	0.009798	0.018894
2.42	0.671624	0.190202	0.381106	0.329485	0.009837	0.01897
2.43	0.670515	0.190163	0.38103	0.330592	0.009877	0.019047
2.44	0.669408	0.190123	0.380953	0.331697	0.009917	0.019123
2.45	0.668303	0.190083	0.380877	0.3328	0.009956	0.0192
2.46	0.6672	0.190044	0.3808	0.333902	0.009996	0.019276
2.47	0.666098	0.190004	0.380724	0.335002	0.010035	0.019353
2.48	0.664998	0.189965	0.380647	0.336099	0.010075	0.019429
2.49	0.663901	0.189925	0.380571	0.337195	0.010115	0.019506
2.5	0.662805	0.189885	0.380494	0.33829	0.010154	0.019582
2.51	0.66171	0.189846	0.380418	0.339382	0.010194	0.019658
2.52	0.660618	0.189806	0.380342	0.340473	0.010233	0.019735
2.53	0.659527	0.189767	0.380265	0.341562	0.010273	0.019811
2.54	0.658438	0.189727	0.380189	0.342649	0.010312	0.019887
2.55	0.657351	0.189688	0.380113	0.343734	0.010352	0.019964
2.56	0.656266	0.189648	0.380036	0.344817	0.010391	0.02004
2.57	0.655183	0.189609	0.37996	0.345899	0.010431	0.020116
2.58	0.654101	0.189569	0.379884	0.346979	0.01047	0.020192
2.59	0.653021	0.18953	0.379808	0.348057	0.01051	0.020269
2.6	0.651943	0.18949	0.379731	0.349133	0.010549	0.020345
2.61	0.650867	0.189451	0.379655	0.350208	0.010589	0.020421
2.62	0.649792	0.189411	0.379579	0.351281	0.010628	0.020497
2.63	0.648719	0.189372	0.379503	0.352352	0.010668	0.020574
2.64	0.647648	0.189332	0.379426	0.353421	0.010707	0.02065
2.65	0.646579	0.189293	0.37935	0.354488	0.010747	0.020726
2.66	0.645512	0.189253	0.379274	0.355554	0.010786	0.020802

2.67	0.644446	0.189214	0.379198	0.356618	0.010826	0.020878
2.68	0.643382	0.189174	0.379122	0.35768	0.010865	0.020954
2.69	0.64232	0.189135	0.379046	0.358741	0.010904	0.02103
2.7	0.641259	0.189096	0.37897	0.3598	0.010944	0.021106
2.71	0.6402	0.189056	0.378894	0.360857	0.010983	0.021183
2.72	0.639143	0.189017	0.378817	0.361912	0.011023	0.021259
2.73	0.638088	0.188977	0.378741	0.362965	0.011062	0.021335
2.74	0.637035	0.188938	0.378665	0.364017	0.011101	0.021411
2.75	0.635983	0.188899	0.378589	0.365067	0.011141	0.021487
2.76	0.634933	0.188859	0.378513	0.366115	0.01118	0.021563
2.77	0.633885	0.18882	0.378437	0.367162	0.011219	0.021639
2.78	0.632838	0.188781	0.378361	0.368207	0.011259	0.021715
2.79	0.631793	0.188741	0.378285	0.36925	0.011298	0.02179
2.8	0.63075	0.188702	0.37821	0.370291	0.011337	0.021866
2.81	0.629709	0.188663	0.378134	0.371331	0.011377	0.021942
2.82	0.628669	0.188623	0.378058	0.372369	0.011416	0.022018
2.83	0.627631	0.188584	0.377982	0.373405	0.011455	0.022094
2.84	0.626595	0.188545	0.377906	0.37444	0.011495	0.02217
2.85	0.62556	0.188505	0.37783	0.375473	0.011534	0.022246
2.86	0.624527	0.188466	0.377754	0.376504	0.011573	0.022322
2.87	0.623496	0.188427	0.377678	0.377533	0.011612	0.022397
2.88	0.622467	0.188388	0.377603	0.378561	0.011652	0.022473
2.89	0.621439	0.188348	0.377527	0.379587	0.011691	0.022549
2.9	0.620413	0.188309	0.377451	0.380611	0.01173	0.022625
2.91	0.619389	0.18827	0.377375	0.381634	0.011769	0.022701
2.92	0.618366	0.188231	0.377299	0.382655	0.011809	0.022776
2.93	0.617345	0.188191	0.377224	0.383674	0.011848	0.022852
2.94	0.616326	0.188152	0.377148	0.384692	0.011887	0.022928
2.95	0.615308	0.188113	0.377072	0.385708	0.011926	0.023003
2.96	0.614292	0.188074	0.376997	0.386722	0.011966	0.023079
2.97	0.613278	0.188034	0.376921	0.387735	0.012005	0.023155
2.98	0.612265	0.187995	0.376845	0.388746	0.012044	0.023231

Although, we have presented data only for 300 iterations, the control loop was executed for 18,375 iterations for the test-case discussed in section 4.4.1.