ROS 2 CONFIGURATION FOR DELTA ROBOT ARM KINEMATIC MOTION AND STEREO CAMERA VISUALIZATION

CHONG SOOK HUI

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

ROS 2 CONFIGURATION FOR DELTA ROBOT ARM KINEMATIC MOTION AND STEREO CAMERA VISUALIZATION

CHONG SOOK HUI

This report is submitted in partial fulfilment of the requirements for the degree of Bachelor of Electronic Engineering with Honours

Faculty of Electronic and Computer Engineering
Universiti Teknikal Malaysia Melaka

2022

DECLARATION

I declare that "ROS 2 Configuration for Delta Robot Arm Kinematic Motion and Stereo Camera Visualization" is the result of my own work except for quotes as cited in the references.

Signature	:	....
Author	:	CHONG SOOK HUI
Date	:	3 JANUARY 2022

APPROVAL

I hereby declare that I have read this thesis and in my opinion this thesis is sufficient in terms of scope and quality for the award of Bachelor of Electronic Engineering with Honours.

Signature	:	....
Supervisor Name	:	DR. WIRA HIDAYAT BIN MOHD SAAD
Date	:	3 JANUARY 2022

DEDICATION

This thesis is dedicated to my father, mother, and younger brothers.

ABSTRACT

The goal of this project is to develop a parallel delta robot simulation environment with stereo camera visualization using Robot Operating System 2 (ROS 2) environment. Unified Robot Description Format (URDF) which is the most commonly used eXtensible Markup Language (XML) specification file to describe the robot physical structural has the limitation of only allowing a robotic system to be modeled in a tree structure. Hence, Simulation Description Format (SDF) is used in this project to describe the delta robot arm that consists of closed-loop kinematic chain linkage to be simulated in Gazebo. Several Gazebo plugin libraries are implemented to perform the wheels motion control, stereo camera visualization, and the delta robot arm kinematic motion. Inverse kinematic motion is performed in this project and the approach of the implementation of the Gazebo plugin has given the result of average percentage error of 3.52 % for the angular rotational degree of all three joints connected in between the motors and biceps. The results of the angular rotational degree of all three joints measured are obtained through varying the position of the end-effector along the Z-axis.

ABSTRAK

*Matlamat projek ini adalah untuk membangunkan persekitaran simulasi robot delta selari dengan visualisasi kamera stereo menggunakan Sistem Operasi Robot 2 (ROS 2). Unified Robot Description Format (*URDF*) yang merupakan fail spesifikasi eXtensible Markup Language (XML) yang paling biasa digunakan untuk menerangkan robot struktur mempunyai had yang hanya membenarkan sistem robotik dimodelkan dalam pokok struktur. Oleh itu, Simulation Description Format (*SDF*) digunakan dalam projek ini untuk menggambarkan lengan robot delta yang terdiri daripada rangkaian rantai kinematik tertutup untuk disimulasikan dalam Gazebo. Beberapa Gazebo plugin diimplementasikan untuk melaksanakan kawalan gerakan roda, visualisasi kamera stereo, dan kinematik gerakan lengan robot delta. Pergerakan kinematik songsang dilakukan dalam projek ini dan pendekatan dengan menggunakan Gazebo plugin telah memberikan hasil purata peratusan ralat sebanyak 3.52 % untuk darjah putaran sudut untuk ketiga-tiga sambungan yang disambungkan di antara motor dan bisep. Keputusan darjah putaran sudut untuk ketiga-tiga sambungan diperoleh melalui pengubahan kedudukan platform bergerak di sepanjang paksi-Z.*

ACKNOWLEDGEMENTS

First of all, I wish to express my sincere gratitude to my parents and my younger brothers for their support throughout the journey. They have given me the motivation to complete the project and overcome all the challenges with a positive attitude.

I am also very thankful to my project supervisor, Dr. Wira Hidayat Bin Mohd Saad who has provided the guidance and motivation for me to complete the project.

Last but not least, I would like to thank my panels, PM. Dr. Lim Kim Chuan and Dr. Ahamed Fayeez Bin Tuani Ibrahim for their advice for the project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

3D	Three-dimensional.
CAD	Computer-aided design.
DOF	Degree Of Freedom.
EOL	End Of Life.
GDP	Gross Domestic Product.
GUI	Graphical User Interface.
LIDAR	Light Detection and Ranging.
LLE	Lower Limb Exoskeleton.
PID	Proportional, Integral, Derivative.
ROS 2	Robot Operating System 2.
RVIZ	ROS Visualization.
RVIZ 2	ROS Visualization 2.
SDF	Simulation Description Format.
TF	Transform.
URDF	Unified Robot Description Format.
V-REP	Virtual Robot Experimentation Platform.
XML	eXtensible Markup Language.
YAML	Yet Another Markup Language.

CHAPTER 1

INTRODUCTION

The word ‘robot’ defines an electromechanical device with multiple Degree Of Freedom (DOF) that is programmable by humans to accomplish multiple varieties of tasks [1]. In the year of 1979, the Robot Institute of America has defined the word ‘robot’ as a "re-programmable, multifunctional manipulator that is designed to move material, parts, tools, or specialized devices through variable programmed motions for the performance of a variety of tasks" [2]. A robot arm in a robotic system represents the end-effector that simulates the human arm to carry out tasks such as picking and placing.

Robotics has been implemented in the industry for years. By studying the similarities between both agriculture and industry field, a robotic system can also perform its role in the field of agriculture to complete some suitable tasks [3]. The study of robotics is to combine and simulate certain aspects of human body function by the implementation of mechanisms, sensors, actuators, and computers [4].

Based on research [5], agriculture is one of the crucial sectors that contribute to the world’s economic growth. For instance, the agriculture sector in Malaysia had contributed 7.1 percent to the Gross Domestic Product (GDP) in 2019 [6]. The increment of the population working in this field has promoted the development of autonomous robotic in agriculture to increase productivity and efficiency [4] [7]. Throughout the revolution in the field of agriculture, different types of robotics development have been implemented. The types of robotic technology included fruit pickers, weed pullers, drones, and Light Detection and Ranging (LIDAR) [8].

One of the robot types that can be implemented in the agriculture field is the delta robot. Delta robot which is also commonly known as a parallel robot is widely used in industrial applications nowadays due to its advantages in speed and product assembly [9]. Delta robot is known as a parallel robot as the top fixed base is always parallel to the bottom moving base in terms of their reference axis. Delta robot can perform the movement in X, Y, and Z direction with a fixed base on top that connects three arms, and all the arms are connected as a triangular form to the bottom base [10]. To enable the image visualization of the object to be picked up, a camera can be attached to the end-effector. In this project, a stereo camera will be attached to the end effector of the delta robot to provide the stereo visualization feature. The stereo camera consists of two or more image sensors which allow it to simulate human binocular vision by setting a suitable distance between two lenses thus allowing the reorganization of depth information [11].

1.1 PROBLEM STATEMENT

Nowadays, the involvement of robotic in the agricultural field shows the high degree of technology and robotics are capable to perform autonomous tasks [3]. In Malaysia, the process of harvesting cili-padi will be carried out by farmers starting from the second month until the sixth month of the cili-padi growth cycle. The co-founders of Suria Agro Trading, Ben Rayappan @ Papoo stated that the harvesting process of chillies is labour intensive as the chillies are small in size and delicate [12]. The idea of the development of an agricultural robot named Cili-Padi Picking Robot as shown in Figure 1.1 has been proposed in the Projek Jangka Pendek (PJP) 2020 to ease the picking process of the cili-padi by replacing the role of a human in performing the repetitive picking process.

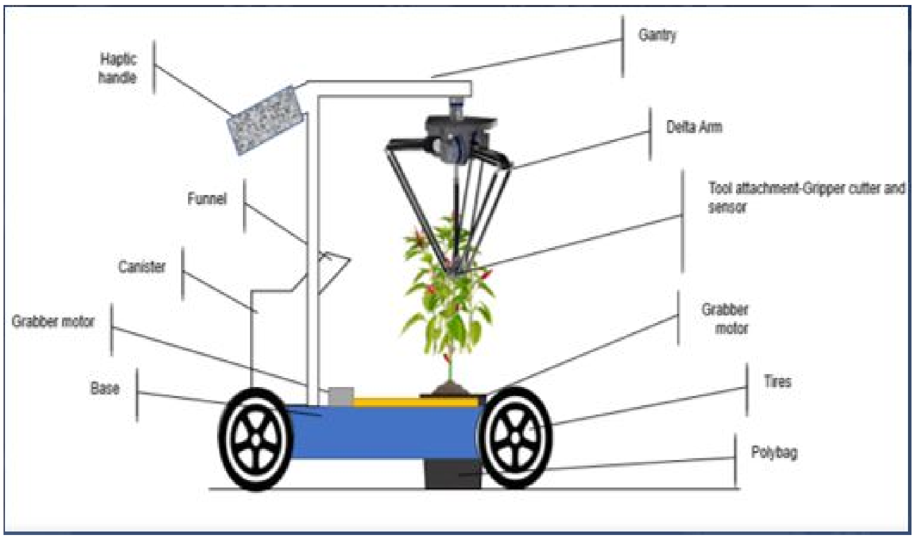


Figure 1.1: Cili-Padi Picking Robot.

Every development process of a physical robot needs to undergo a simulation stage in a suitable software platform to ensure the performance of the robot before further proceeding to the hardware installment and development. Hence, the simulation of the customized-design delta robot with stereo camera visualization in the Robot Operating System 2 (ROS 2) environment will be carried out in this project.

1.2 AIM AND OBJECTIVE

This project aims to develop the delta robot arm in Robot Operating System 2 (ROS 2) simulation environment with stereo camera visualization.

Objectives to achieve the aim:

1. To develop a parallel delta robot simulation environment with a stereo camera using ROS 2 environment.

2. To analyse the kinematic motion of the robot based on end-effector position and servo motor position.

1.3 SCOPE OF PROJECT

The scope of this project is focused on the development of the simulation of the delta robot arm in the ROS 2 Foxy Fitzroy environment. The stereo camera visualization feature of the delta robot will be developed. Besides, the kinematic motion simulation of the delta robot arm will be performed and the angular rotational degrees of the servo motors will be analyzed based on the position of the end-effector.

1.4 THESIS OUTLINE

This thesis paper is divided into five chapters:

Chapter 1 introduces the background of this project and the problem statement to be solved through the project development. The aim and objective, and scope of the project are also included.

Chapter 2 presents the literature review made by referring to some proposed projects by other organizations which carry out robot simulation by using the Robot Operating System middleware environment. The research gap is identified.

Chapter 3 discusses the research methodology, the overall project system flow, and the project development flow. The project development processes are explained.

Chapter 4 records the results obtained throughout the development process of the project. The obtained result is analyzed and discussed. The sustainability of the project is also discussed.

Chapter 5 concludes the overall achievement of the project. The potential future improvement of the project is also included.

CHAPTER 2

LITERATURE REVIEW

This chapter provides general information on the crucial elements in this project. The reviews of some related projects done by other organizations which are also carried out the robot development and simulation process in the Robot Operating System (ROS) environment are also included.

2.1 ROBOT OPERATING SYSTEM 2 (ROS 2)

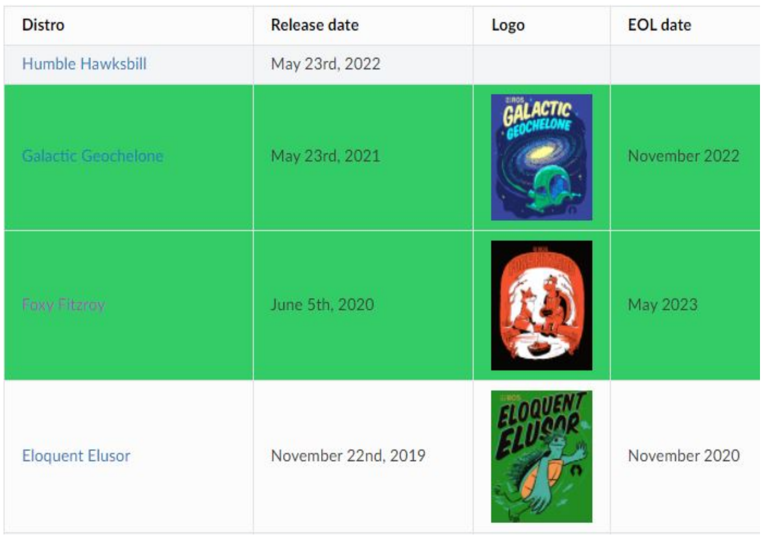
Robot Operating System which is known as ROS is a middleware framework that offers an abstraction layer between the hardware and the application layers. ROS allows the developers to perform manipulation on the robot hardware. ROS is an open-source platform that relies on the contribution of the developers and the ROS user community in the introduction of new packages or libraries. It is equipped with a package management system that allows users to perform code reuse easily and so allows developers to contribute their design and application back as a package to ROS [13].In the service robot sector, ROS goes on top of Linux Ubuntu and is becoming the standard in robotic programming [14].

Even though the name of ROS consists of the term "operating system", but it is not an actual operating system. ROS functions as a Communication System through the Publish-Subscribe communicating method and Remote Method Invocation. Besides, ROS is also an Ecosystem for language bindings, drivers, libraries, and simulation [15].

ROS 2 is a complete re-design of the framework of ROS to tackle the shortcomings of ROS which allow ROS 2 to effectively meet the industry needs and standards [16]. The biggest difference between ROS and ROS 2 is that ROS 2 selects the Data Distribution Service (DDS) middleware for the communication layer [15]. The DDS middleware is a type of software layer that can abstract the application from the details of the operating system, network transport, and low-level data formats. The Dynamic Discovery of publishers and subscribers in ROS 2 is provided by the DDS. With Dynamic Discovery, the endpoints for the communications do not require any configuration as they will be automatically discovered by DDS [17].

The distro of ROS 2 selected for this project is ROS 2 Foxy Fitzroy as shown in Figure 2.1 [18] with the codename of 'foxy'. Foxy Fitzroy is the sixth release of ROS 2 on June 5th, 2020 that will meet its End Of Life (EOL) date on May of 2023 as shown in Figure 2.2 [18].

Figure 2.1: Logo of Foxy Fitzroy.



Distro	Release date	Logo	EOL date
Humble Hawksbill	May 23rd, 2022		
Galactic Geochelone	May 23rd, 2021		November 2022
Foxy Fitzroy	June 5th, 2020		May 2023
Eloquent Elusor	November 22nd, 2019		November 2020

Figure 2.2: Release date and EOL date of ROS 2 Foxy Fitzroy.

2.1.1 Topic and Node

In ROS 2, a topic plays an important role in acting as a bus for the messages exchange between nodes as shown in Figure 2.3 [19]. A node may simultaneously publish data to several topics and subscribe to several topics. When a Publisher node wants to send a message to a Subscriber node, the Publisher node will need to publish the message to the topic so that message will be subscribed by the Subscriber node via the topic.

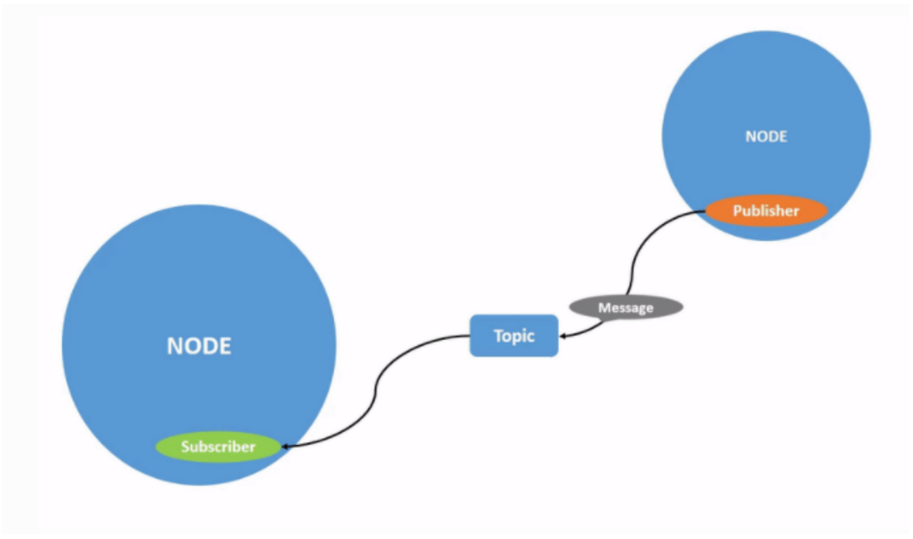
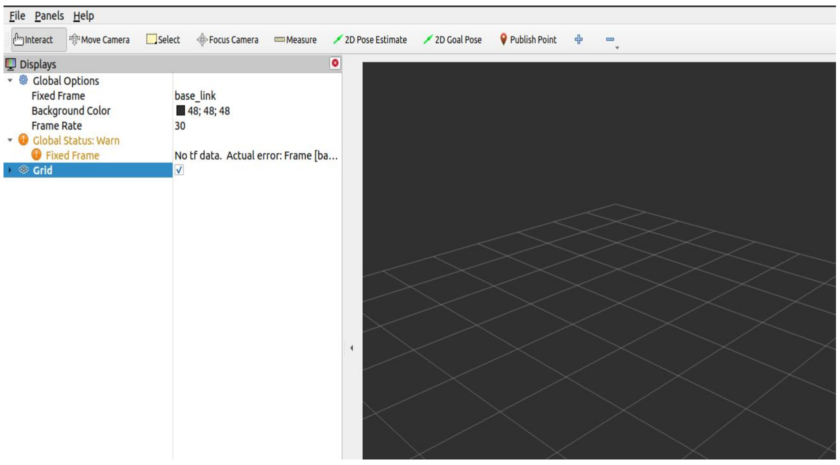


Figure 2.3: Message exchange between two nodes.

2.1.2 Rviz 2

Rviz stands for ROS Visualisation, is a ROS 3D visualization tool that allows users to visualize the robot model through the Transform (TF) published by a ROS package such as *robot_state_publisher*. Other than that, Rviz is capable to display the data captured by a camera or a laser scan in terms of image or point clouds. In ROS 2, the name of Rviz has been modified to Rviz 2. The Graphical User Interface (GUI) of Rviz 2 is shown in Figure 2.4.



Global Options	
Fixed Frame	base_link
Background Color	■ 48; 48; 48
Frame Rate	30
Global Status: Warn	
Fixed Frame	No tf data. Actual error: Frame [ba...
Grid	✓

Figure 2.4: GUI of Rviz.

2.1.3 Gazebo

The GUI of Gazebo as shown in Figure 2.5 is a 3D robot simulator that allows the simulation of a robot to be carried out in a virtual world that is able to mimic the environment of a real world.

A Gazebo is able to provide a close substitute for how a robot would behave in a real-world physical environment as it may replicate forces such as gravity and torque in the simulation environment. On the left column in the GUI of Gazebo under the Physics tab as shown in the Figure 2.6, the parameters such as gravity or magnetic field can be activated when needed. With the features available in Gazebo, it allows developers to test and simulate the robot function in a simulated real-world environment before access to the hardware installation [20].

The main difference between Rviz and Gazebo can be summed up in the excerpt from Morgan Quiley which is one of the original developers of ROS in his book "Programming Robots with ROS": "Rviz visualize what the robot think is happening while Gazebo shows what is happening in a real-world environment." [21].

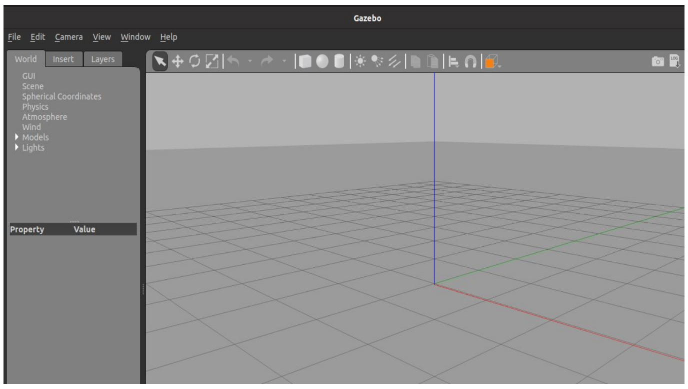
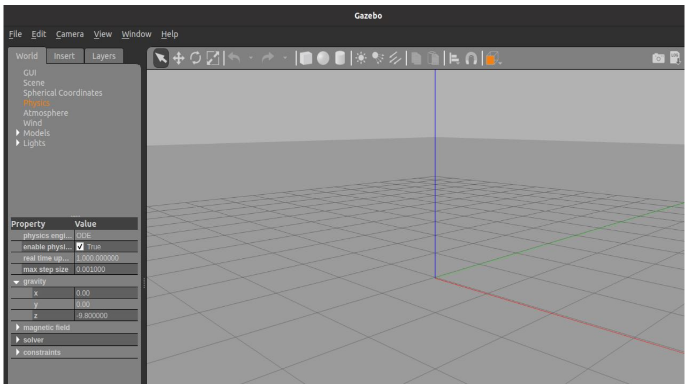


Figure 2.5: GUI of Gazebo.



Property	Value
physics engi...	ODE
enable physi...	✓ True
real time up...	1,000.000000
max step size	0.001000
gravity	
x	0.00
y	0.00
z	-9.800000
magnetic field	
solver	
constraints	

Figure 2.6: Physic parameters in Gazebo.

2.1.4 Universal Robot Description Format (URDF)

URDF stands for Universal Robot Description Format is a eXtensible Markup Language (XML) specification file that describes the robot's physical structural description such as links, joints, and actuators. URDF can be used in ROS to allow the program to understand how the robot actually looks like in real life [22]. In URDF, the robotic system is modeled in a tree structure which indicates that a child link can only have a single parent link thus it is not applicable for a robotic system such as a delta robot that consists of a close loop kinematic chain [23].

2.1.5 Simulation Description Format (SDF)

Simulation Description Format which can be abbreviated as SDFormat or commonly abbreviated as SDF is another XML format file that describes the robot's physical structural description and the environments for the robot simulators, visualization, and control. Other than kinematic and dynamic attributes, SDF also allows the properties such as sensor, surface properties, texture, and joint friction of a robot to be defined [24]. SDF is the standard format of the Gazebo simulation modeling where the sensors and actuators can be implemented in Gazebo by using plugin library [25].

2.2 DELTA ROBOT

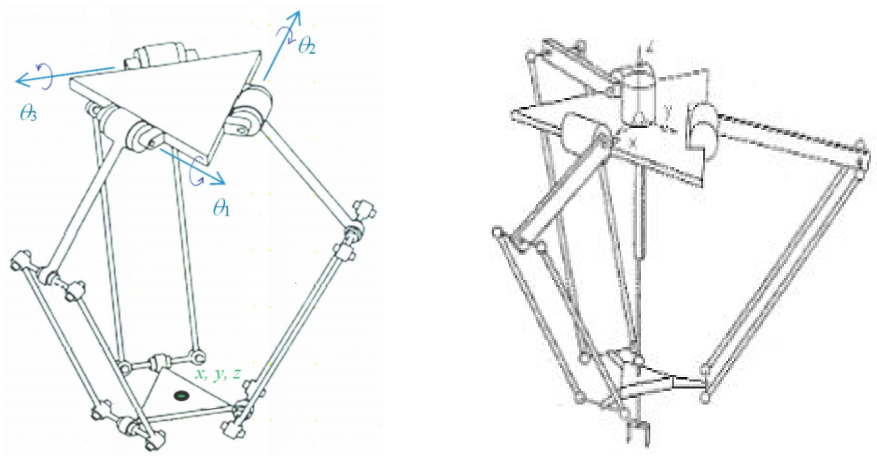
Delta Robot or also known as a parallel robot is widely used in industry due to its high-speed performance as it has a lightweight structure. It is a successful invention from Reymon Clavel in 1985. The Delta robot is often used in industrial involving picking and packaging process [26]. The Delta robot is composed of several kinematic chains that connect in between the end-effector and the top base to perform movement through the rotation of the servo motor and the joints [27].

Delta robot is available in different Degree Of Freedom (DOF) such as 3-dof or 4-dof as shown in Figure 2.8a [28] and Figure 2.8b [29]. Degree-of-freedom indicates the number of movable joints that are available in the Delta Robot. There are 3-dof for X-Y-Z translation for a 3-dof Delta Robot while 4-dof Delta Robot has an extra fourth inner leg which controls the rotation freedom of the platform of the end effector [28]. By analyzing the kinematic links of a Delta Robot, the total number of degrees of freedom can be computed through the Equation 2.1 [30]

$$F = b(n-g-1) + \sum_{i=1}^{g} fi - fid + s \tag{2.1}$$

where

- b = number of DOF in space
- n = number of elements
- g = number of joints
- fi = DOF of i-th element
- fid = number of identical DOF
- s = number of passive joints



(a) 3-dof Delta Robot. (b) 4-dof Delta Robot.

Figure 2.7: Types of Delta Robot.

In this project, a 3-dof Delta Robot is simulated in the ROS 2 environment. The 3-dof Delta Robot is composed of three identical kinematic chains or also known as arms that are connected in a parallelogram mechanism in between the top fixed base and the bottom move-able base (end-effector). The parallelogram in the arms is used to ensure that the orientation of the end-effector is constant [31].

2.2.1 Kinematics

Kinematics refers to the motion performed by the body of a robotic mechanism without consideration of the forces and torques that cause the motion. A robotic mechanism is formed by the connection between several rigid bodies through the joint where the orientation and position of the rigid body are known as "pose". Hence, the kinematics of a robotic mechanism describe the velocity, acceleration, pose, and all higher-order derivatives of the pose of the body that comprises a mechanism [32].

Based on the research paper done by M.Mahmoodi, K.Alipour, and M. G. Tabrizi [33], the geometric parameters of a 3-RRR delta robot are shown in Figure 2.8 where the location of 3 motors are located 120 degrees apart. The composition of RRR in the 3-RRR delta robot indicates the joint type of Revolute-Revolute-Revolute.

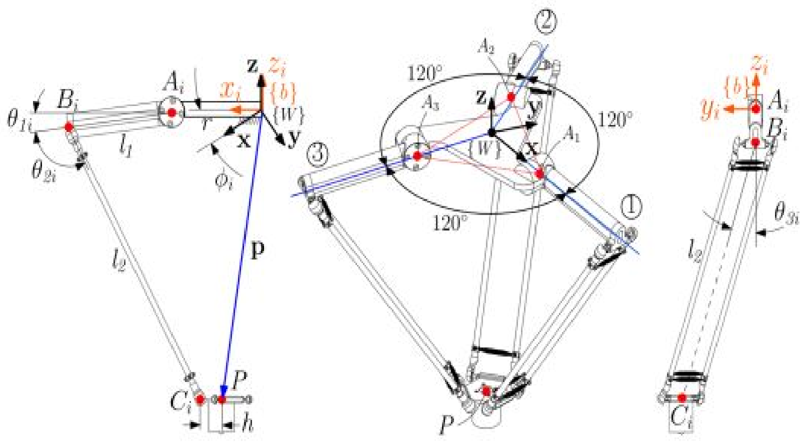


Figure 2.8: Geometric parameters of 3-RRR delta robot.

By referring to Figure 2.8, {w} represents the fixed world coordinate system that is located in the middle of the triangle formed by the points A_1, A_2, and A_3 while {b} indicated the body coordinate system. Note that three chains of the delta robot arms are identical to each other, hence Figure 2.8 is taking one of the chains for an explanation. The geometric parameters of the robot are labelled as l_1, l_2, r, h, and ϕ_i where i = 1,2,3. l_1 represents the length of the bicep, l_2 represents the length of the arm, r represents the radius of the top fix base, h represents the radius of the end-effector. Besides, $\theta_1 i$, $\theta_2 i$, $\theta_3 i$ represent the joint angles that define the configuration of each chain.

Based on the research paper [33], the general constraint equation of each of the three kinematic chains of a 3-RRR delta robot can be written Equation 2.2

$$({X_p}^2 - {X_i}^2) + ({Y_p}^2 - {Y_i}^2) + ({Z_p}^2 - {Z_i}^2) = {l_2}^2, \quad i = 1,2,3 \tag{2.2}$$

where

$$= \begin{cases} X_i = (r-h+l_1 cos\theta_1 i)cos\phi_i \\ Y_i = (r-h+l_1 cos\theta_1 i)sin\phi_i \quad ,i=1,2,3 \\ Z_i = -l_1 sin\theta_1 i \end{cases} \tag{2.3}$$

2.2.2 Forward Kinematic

Forward kinematic motion of a delta robot indicates the condition where the joint variable such as the angular rotation angle of the servo motor is known and is used to compute the position and the orientation of the end-effector of the delta robot arm [34].

2.2.3 Inverse Kinematic

Inverse Kinematic motion is performed when the position and orientation of the end-effector are known. The angle of angular rotation of the servo motor and the joints will be calculated and identified based on the orientation of the end-effector [34].

2.3 STEREO CAMERA

A stereo camera is a device that implements stereo vision technology that captures two images of the same object from two different viewpoints which involve two camera lenses to obtain the depths of points on an object in 3-dimension. The example of a stereo camera is shown in Figure 2.9 [35].

Figure 2.9: Example of a stereo camera.

2.3.1 Stereo Camera Visualization

In the visualization of a stereo camera, a pair of stereo images will be captured as shown in Figure 2.10 [36] by two camera lenses of a stereo camera that are located at a certain distance apart from each other. The distance between the two cameras is called the baseline distance of a stereo system as shown in Figure 2.11 [37] [38].

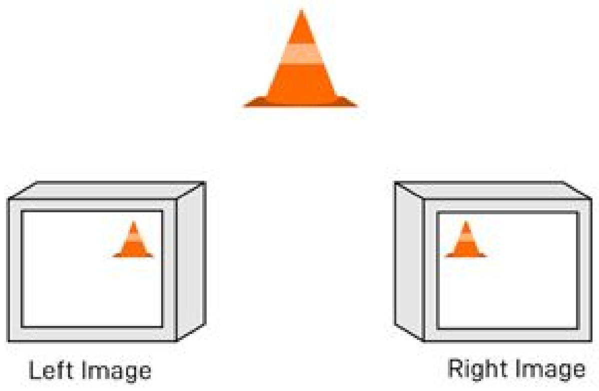


Figure 2.10: Left and right images captured by stereo camera.

There are two types of camera parameters that contribute to the generation of photos or images. Intrinsic parameters of the camera such as focal length and lens distortion. Extrinsic parameters are the external parameter such as translation and rotation of a camera that is used to describe the transformation between the camera and the external world [39].

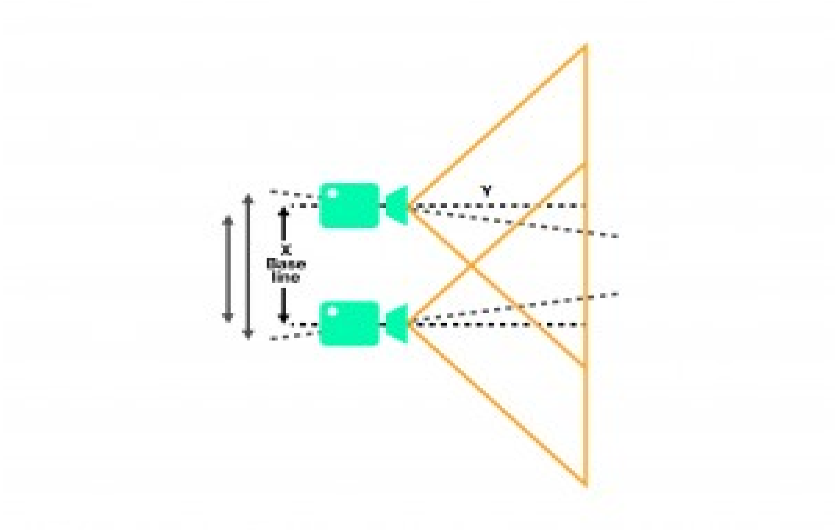


Figure 2.11: Baseline in between two camera lens.

2.3.2 RGB-D Image

The phrase RGB-D stands for Red, Green, Blue-Depth. Other than RGB information, an RGB-D image can also provide depth and infrared information which are very useful in object detection and localization. [40] A normal pixel-based image can provide three properties (R, G, B) color by locating the (x,y) coordinates while an RGB-D image can provide four properties including depth in each (x,y) coordinates. RGB-D image can be represented by a stack of single-valued images in which each pixel carries four properties as shown in Figure 2.12 [41].

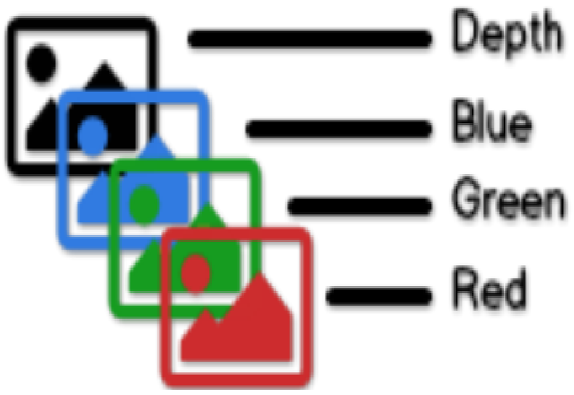


Figure 2.12: Representation of RGB-D image as a stack of single valued images.

2.4 RELATED PROJECTS

2.4.1 Delta robot controlled by robotic operating system

In this paper, D. Rivas-Lalaeo et al. has carried out the implementation of wireless control system on a parallel robot based on the RS-232 interface. The simulation environment chosen in this project is ROS for implementing the controlling of the robot joint actuators. Program scripts are built to apply the inverse kinematics of the delta robot. Besides, Python-based algorithms supported by OpenCV libraries are also developed for vectorizing and rasterizing images. The developed robot undergoes an experiment in drawing application and the inverse kinematics principle is used to determine the angle of each actuator by knowing the position of the end-effector. This project carried out the robot movement simulation process in Rviz and Markers is used to represent the simple shapes of the delta robot as shown in Figure 2.13 [42].

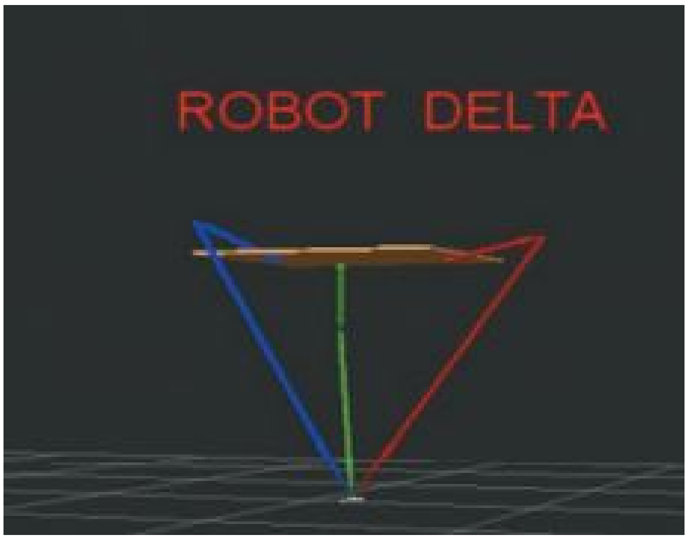


Figure 2.13: Delta robot represented with Marker in Rviz.

2.4.2 Robotic Harvesting of Fruiting Vegetables: A Simulation Approach in V-REP, ROS and MATLAB

In the paper done by Shamshiri et al., the sensors and manipulators of robotic harvesting of sweet pepper have experimented in the designed simulation and control platform in Virtual Robot Experimentation Platform (V-REP), ROS, and MATLAB. The simulation workspace was carried out in V-REP while the image moment method visual servoing with eye-in-hand configuration was implemented in MATLAB. MATLAB is used as the remote API to allow the visual servoing control code to be written, modified, and run. ROS was used as a platform for data exchanging between the simulated environment and the real workspace via publish and subscribe architecture [43].

2.4.3 Cable-Driven Parallel Robot Simulation Using Gazebo and ROS

This project done by F. Okoli et.al studied the cable-driven parallel robot by developing a simulator by using Gazebo and ROS. The properties such as mass, inertia, pose, link visual, and collision of the robot are described in an SDF file. A dynamic controller is developed to illustrate the proposed simulator by detailing the tension distribution and various trajectories are performed. A trajectory controller is developed to compute the desired wrench to be applied to the platform and a Tension Distribution Algorithm is used to map the desired wrench [44].

2.4.4 Simulation and Control of a Six Degree of Freedom Lower Limb Exoskeleton

The paper done by M. S. Amiri et.al presented the development of the control method for a six DOF Lower Limb Exoskeleton (LLE) model in ROS. The structure of the robot model is described in the Unified Robot Description Format (URDF). By inserting the transmission tags in URDF, the Gazebo plugin is then added to parse the transmission tags and loads the hardware interface and controller manager.A Yet Another Markup Language (YAML) configuration file is that specify the controller types, joint names and Proportional, Integral, Derivative (PID) parameters are loaded to launch *joint_trajectory_controller* to publish trajectory commands to the joints [45].

2.4.5 Using ROS for agricultural robotics - design considerations and experiences

The paper done by R. Barth et.al reported the experience of using ROS middleware for the development of agricultural robots. In the project, ROS is used to construct software for several subsystems for sensing, perception, manipulator control, mission control, and system framework. The perception framework of the project is included physical sensors, virtual sensors, and sensor fusion. The dull implementation of the modular framework was conducted in ROS and MATLAB where the development and evaluation of algorithms are implemented in MATLAB and were transferred over to ROS once they are finalized [46].

2.4.6 Simulation and Framework for the Humanoid Robot TigerBot

In the paper done by F. Sze, TigerBot which is a full-scale humanoid robot with 7-dof legs is simulated in Gazebo. There are encoders set up on each of the joints of TigerBot for position control and its sensor and joints are connected to the Teensy microcontroller. ROS is used as the communication system in the project. MoveIt! that takes the URDF file is used for calculating the inverse kinematics to control the feet through the Trac-IK Kinematic Solver plugin and perform the path-planning function. *ros_control* package is used to set up the controllers such as position controller, velocity controller, and position controller to handle groups of joints. Position controller is used to control the legs in this project [47].

2.5 SUMMARY

To summarise, the general knowledge regarding the delta robot, kinematics, and the stereo camera is studied. Related projects proposed by other organizations or research groups which carried out the robotic system simulation process in ROS are also reviewed.

In this project, Gazebo will be used as the 3D simulator to simulate the operation of the robotic system and Rviz 2 will be used to visualize the stereo images captured by the virtual stereo camera attached to the delta robot arm. The research gap between this project and the project from [42], [43] and [45] is that this project aims to simulate the kinematic motion of a delta robot with 3-dof in Gazebo simulator by describing the robot physical structural in SDF file as SDF is able to describe the closed-loop kinematic chain linkage of the delta robot arm to be simulated in Gazbeo. Besides, the research gap between this project and the project proposed in [44] and [46] is that the simulation of this project will be carried out without the implementation of algorithm evaluation in MATLAB where the currently available plugin library resources in ROS 2 will be utilized without involving the development of a customize algorithm or controller.

CHAPTER 3

METHODOLOGY

This chapter explains the methodology implemented in this project to obtain the desired final result. The research methodology flowchart and the overall system development flowchart of the project are recorded and explained.

3.1 RESEARCH METHODOLOGY FLOWCHART

The research methodology flowchart as shown in Figure 3.1 records the processes that have been carried out in completing the entire project. First of all, the background study regarding the software, middleware, and other crucial elements required in this project has been done and the general development concept has been identified. The project is then developed to simulate a delta robot in Robot Operating System 2 (ROS 2) environment where the arm kinematic motion is performed. The delta robot is also equipped with a stereo camera at the end-effector of the arm to allow stereo camera visualization. The thesis of the project is completed based on the result obtained throughout the development and the completion of the project.

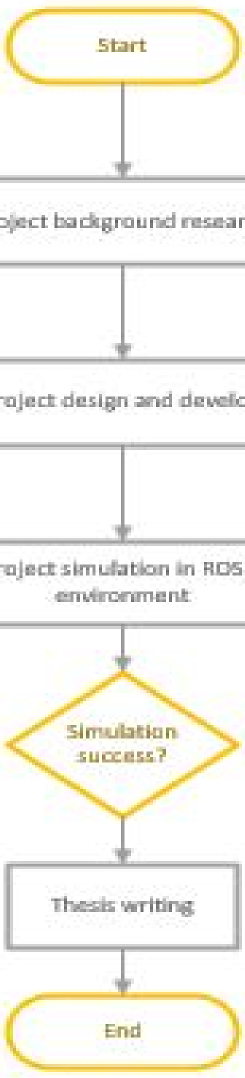

Start
Project background research
Project design and develop
Project simulation in ROS 2 environment
Simulation success?
Thesis writing
End

Figure 3.1: Research Methodology Flowchart.

3.2 SOFTWARE

This project is developed to carry out the simulation of the delta robot in the ROS 2 environment hence the development of the project is fully depends on software. The software involved in the development of the project are SOLIDWORKS and Oracle VirtualBox.

3.2.1 SOLIDWORKS

SOLIDWORKS software as shown in Figure 3.2 [48] is a Computer-aided design (CAD) software that allow user to create a Three-dimensional (3D) solid model component system. SOLIDWORKS software is equipped with powerful modeling functions and efficient modeling efficiency hence making it stand out in many other 3D modeling software [49].

Figure 3.2: SOLIDWORKS software.

3.2.2 Oracle VirtualBox

Oracle VirtualBox as shown in Figure 3.3 [50] is a x86 and AMD64/Intel64 virtualization product. VirtualBox can run on Windows, Linux, Macintosh, and Solaris hosts and is capable to support a large number of guest operating systems such as Windows, Linux, Solaris and OpenSolaris, and OpenBSD [51]. In this project, VirtualBox runs on Windows and supports the guest operating system Linux.

Figure 3.3: Oracle VirtualBox software.

3.3 PROJECT FLOWCHART

The project flowchart of this project is constructed to ensure that the project can be designed and developed systematically according to the milestone and the goal of each stage as shown in Figure 3.4.

The project flowchart starts with a background study on the software involved in this project. The software needed for the development of the project such as is Oracle VirtualBox and SOLIDWORKS are installed. In the VirtualBox software, ROS 2 is installed and the distro selected for this project is Foxy Fitzroy. After setting up the ROS 2 simulation environment, the Unified Robot Description Format (URDF) file of the Delta Robot is configured and exported from SOLIDWORKS. The URDF file is then converted to Simulation Description Format (SDF) and launched into Gazebo to ensure that all the joints and base of the robot had been correctly coordinated to form a delta robot close-loop kinematic chain linkage before proceeding to kinematic motion simulation.

The motion control of the wheels of the delta robot is then performed to allow the movement of the robot. In addition, a stereo camera is attached to the end-effector of the delta robot arm and the stereo images captured are viewed in ROS Visualization 2 (RVIZ 2). Lastly, the kinematic motion of the delta robot arm is simulated.

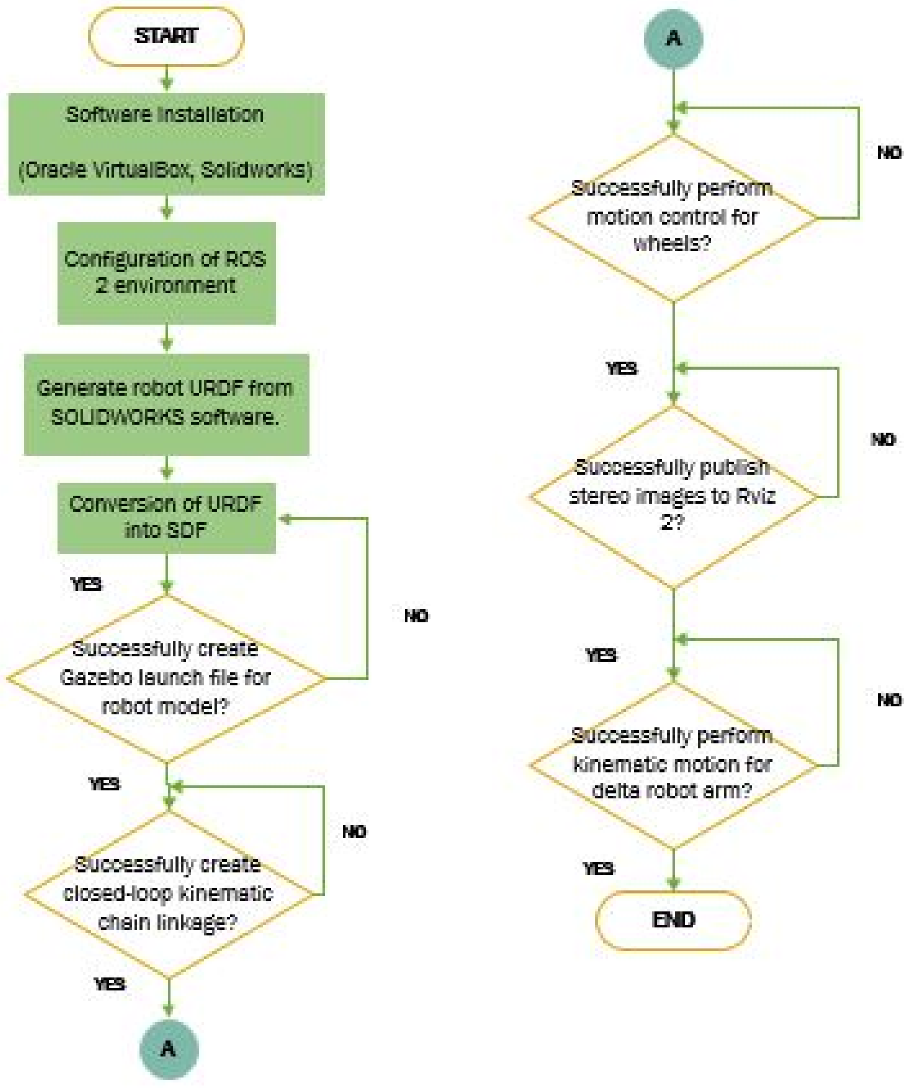

START
Software Installation
(Oracle VirtualBox, Solidworks)
Configuration of ROS 2 environment
Generate robot URDF from SOLIDWORKS software.
Conversion of URDF into SDF
YES
Successfully create Gazebo launch file for robot model?
NO
YES
Successfully create closed-loop kinematic chain linkage?
NO
YES
A
A
Successfully perform motion control for wheels?
NO
YES
Successfully publish stereo images to Rviz 2?
NO
YES
Successfully perform kinematic motion for delta robot arm?
NO
YES
END

Figure 3.4: Project Flowchart.

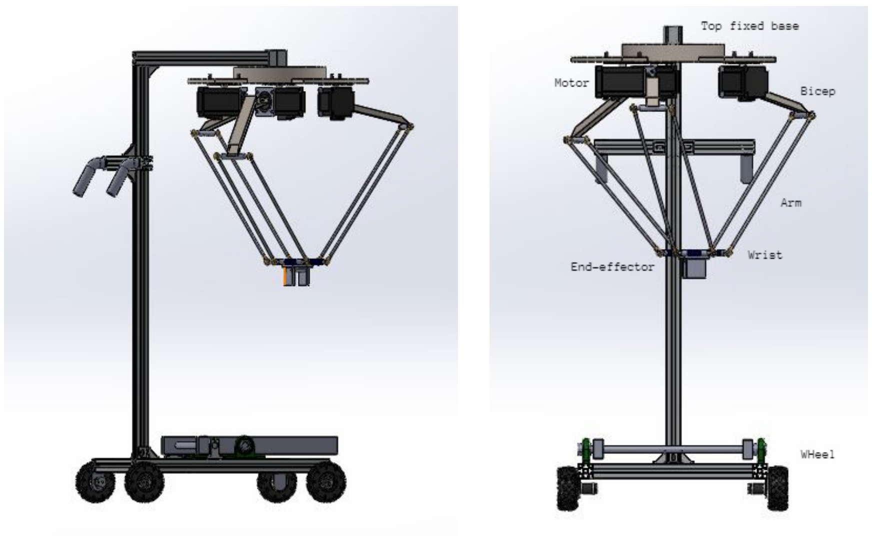
3.4 PROJECT DEVELOPMENT

3.4.1 Configuration of ROS 2 environment

Ubuntu Linux - Focal Fossa (20.04) 64-bit is loaded into Oracle VirtualBox software as the target platform for installing ROS 2 Foxy Fitzroy. The ROS 2 Foxy Fitzroy is installed through Debian packages. To set up the ROS 2 environment, tutorials from the official website of ROS need to be practiced and undergone to ensure that all the development tools, ROS tools, and dependencies needed for the environment have been installed.

3.4.2 Generate URDF file

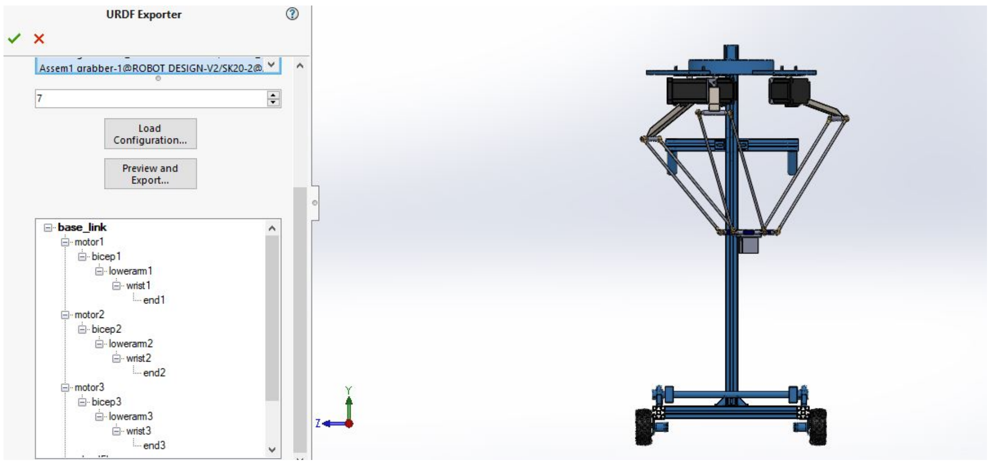
The Computer-aided design (CAD) of the delta robot used in this project is viewed in SOLIDWORKS software as shown in Figure 3.5 where Figure 3.6a shows the side view of the delta robot while Figure 3.6b shows the front view of the delta robot with the name of the parts labeled.



(a) Side view. (b) Front view.

Figure 3.5: Computer-aided design of delta robot.

By installing the SOLIDWORKS to the URDF exporter plugin, the URDF of the delta robot can be generated directly from the SOLIDWORKS. The interface for the configuration of the URDF file is shown in Figure 3.6. The parent link and child link for each of the parts of the delta robot needs to be configured to form a tree structure chain that describes the physical structure of the robot.



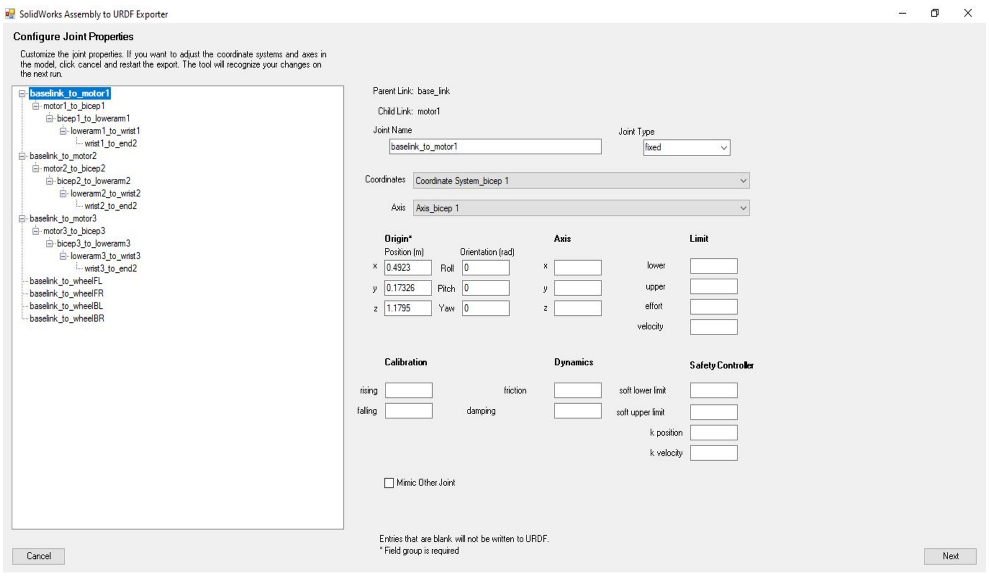
Assem1 grabber-1@ROBOT DESIGN-V2/SK20-2@.

7

```
base_link
  motor1
    bicep1
      lowerarm1
        wrist1
          end1
  motor2
    bicep2
      lowerarm2
        wrist2
          end2
  motor3
    bicep3
      lowerarm3
        wrist3
          end3
```

Figure 3.6: Interface for URDF converter configuration.

Once the tree structure of the links connection has been configured, the joint properties for each of the joints in the structure of the robot need to be configured as shown in Figure 3.7. The type of joint for each joint needs to be specified based on the function to be performed by the specific joint. The type of joints that can be selected include fixed, revolute, prismatic, continuous, floating, and planar. In this project, the joint from the motor to the bicep, the joint from bicep to arm, and the joint from arm to the wrist are selected to be revolute thus forming a 3-RRR delta robot. Other than that, the joint limit which indicates the highest and the lowest angular rotational angle limit that can be performed by a joint can be set while generating the URDF file of a robot.



Configure Joint Properties

Customize the joint properties. If you want to adjust the coordinate systems and axes in the model, click cancel and restart the export. The tool will recognize your changes on the next run.

```
baselink_to_motor1
  motor1_to_bicep1
    bicep1_to_lowerarm1
      lowerarm1_to_wrist1
        wrist1_to_end2
baselink_to_motor2
  motor2_to_bicep2
    bicep2_to_lowerarm2
      lowerarm2_to_wrist2
        wrist2_to_end2
baselink_to_motor3
  motor3_to_bicep3
    bicep3_to_lowerarm3
      lowerarm3_to_wrist3
        wrist3_to_end2
baselink_to_wheelFL
baselink_to_wheelFR
baselink_to_wheelBL
baselink_to_wheelBR
```

Parent Link: base_link

Child Link: motor1

Joint Name: baselink_to_motor1

Joint Type: fixed

Coordinates: Coordinate System_bicep 1

Axis: Axis_bicep 1

Origin*

	Position (m)		Orientation (rad)
x	0.4923	Roll	0
y	0.17326	Pitch	0
z	1.1795	Yaw	0

Axis: x, y, z

Limit: lower, upper, effort, velocity

Calibration: rising, falling

Dynamics: friction, damping

Safety Controller: soft lower limit, soft upper limit, k position, k velocity

☐ Mimic Other Joint

Entries that are blank will not be written to URDF.
* Field group is required

Figure 3.7: Interface for joint properties configuration.

3.4.3 Conversion of URDF Into SDF

As URDF can only support tree structure kinematic chain while delta robot has a close loop kinematic chain linkage, the URDF file needs to be converted into an SDF file that supports close-loop linkage. By assuming the filename to be "robot", the conversion of URDF file into SDF file is done by running the following command in the terminal:

```
gz sdf -p robot.urdf > robot.sdf
```

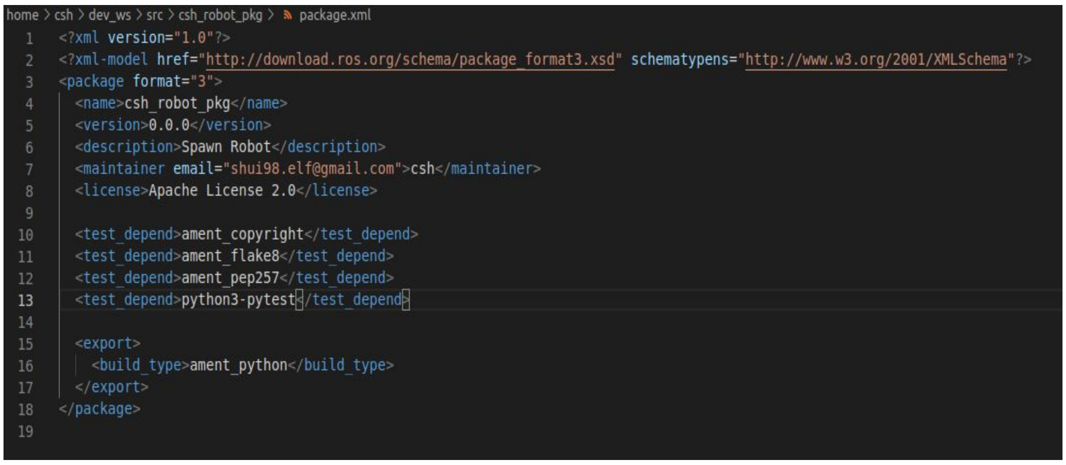
Note that the current directory needs to be the workspace that containing the URDF file. Once the SDF file has been generated in the current directory, the line *<?xml version="1.0"?>* needs to be added before the <sdf> tag in the file to add the eXtensible Markup Language (XML) property to the file.

3.4.4 Create Close Loop Linkage in SDF File

The SDF file converted from the URDF file is initially having the same structure description as the URDF which is in a tree structure. Therefore, some modifications need to be done in the code of the SDF file in order to form a close loop linkage between the three arms and the end-effector. The two extra repeated end-effectors links defined in the URDF have been removed and only an end-effector link has remained. Three of the arms are connected to the single end-effector through the joint located between wrist and end-effector. The child link defined in the joint connection of all three of the wrists is the same end-effector.

3.4.5 Create Gazebo Launch File

A launch file is created to ease the spawning process of the robot in Gazebo by only running a single line of command. The tutorial published by Automatic Addison [52] is referred to create a launch file for the delta robot in this project. A package with a customized name needs to be created where it will act as a container for the ROS 2 code. The package creation in ROS is using ament as the build system and colcon as the build tool. The example of package file created is shown in Figure 3.8 where the package created in this project is named as "*csh_robot_pkg*".



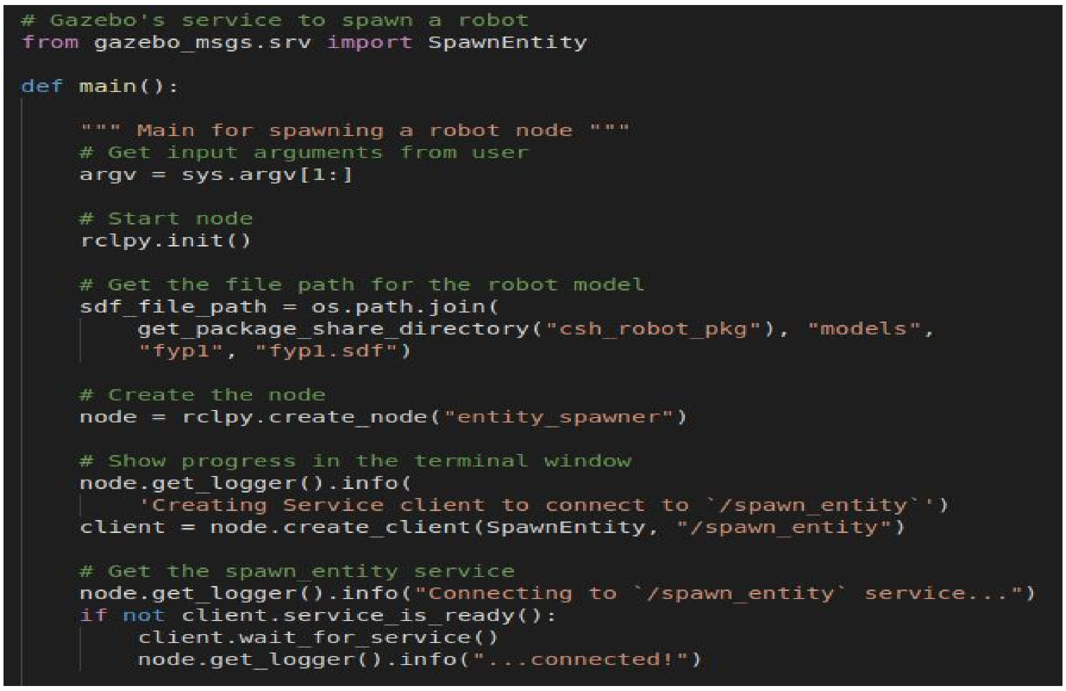
home > csh > dev_ws > src > csh_robot_pkg > package.xml

```
<?xml version="1.0"?>
<?xml-model href="http://download.ros.org/schema/package_format3.xsd" schematypens="http://www.w3.org/2001/XMLSchema"?>
<package format="3">
  <name>csh_robot_pkg</name>
  <version>0.0.0</version>
  <description>Spawn Robot</description>
  <maintainer email="shui98.elf@gmail.com">csh</maintainer>
  <license>Apache License 2.0</license>

  <test_depend>ament_copyright</test_depend>
  <test_depend>ament_flake8</test_depend>
  <test_depend>ament_pep257</test_depend>
  <test_depend>python3-pytest</test_depend>

  <export>
    <build_type>ament_python</build_type>
  </export>
</package>

```

Figure 3.8: Example of ROS 2 package file.

After the package is created, other needed files can then be created within the folder that is containing the package file.

Another Python file named "*spawn_demo.py*" will be created to act as a ROS 2 node that will be used to spawn the robot in Gazebo. In *spawn_demo.py*, the directory of the robot SDF file to be read is defined as shown in Figure 3.9 and the *spawn_entity* service will be called.



```python
# Gazebo's service to spawn a robot
from gazebo_msgs.srv import SpawnEntity

def main():

    """ Main for spawning a robot node """
    # Get input arguments from user
    argv = sys.argv[1:]

    # Start node
    rclpy.init()

    # Get the file path for the robot model
    sdf_file_path = os.path.join(
        get_package_share_directory("csh_robot_pkg"), "models",
        "fyp1", "fyp1.sdf")

    # Create the node
    node = rclpy.create_node("entity_spawner")

    # Show progress in the terminal window
    node.get_logger().info(
        'Creating Service client to connect to `/spawn_entity`')
    client = node.create_client(SpawnEntity, "/spawn_entity")

    # Get the spawn_entity service
    node.get_logger().info("Connecting to `/spawn_entity` service...")
    if not client.service_is_ready():
        client.wait_for_service()
        node.get_logger().info("...connected!")
```

Figure 3.9: Example of ROS 2 node file to spawn entity.

When all the necessary files have been created, the Gazebo launch file will be the one that calls the necessary nodes and files in order to successfully spawn the delta robot in the Gazebo. The world environment in Gazebo can be chosen based on the requirement of the project developed where there are some example world files such as mud.world, cafe.world, road.world or empty.world that can be launched. In this project, the world extension file to be used is the empty.world which only consists of a ground plane. As shown in Figure 3.10, the launch file will search and launch the world extension file that will be used in this project which is the empty.world together with the SDF file that describes the structure of the delta robot. The launch file created in this project is named "*gazebo_world.launch.py*.

3.4.6 Motion Control for Wheels

In this project, the delta robot arm is attached to a body that is equipped with 4 wheels as shown in Figure 3.5. To allow the movement of the robot around the world, a suitable Gazebo plugin will be implemented in the SDF file. In this project, the *skid_steer_drive* plugin is implemented. The *skid_steer_drive* is selected as dif-

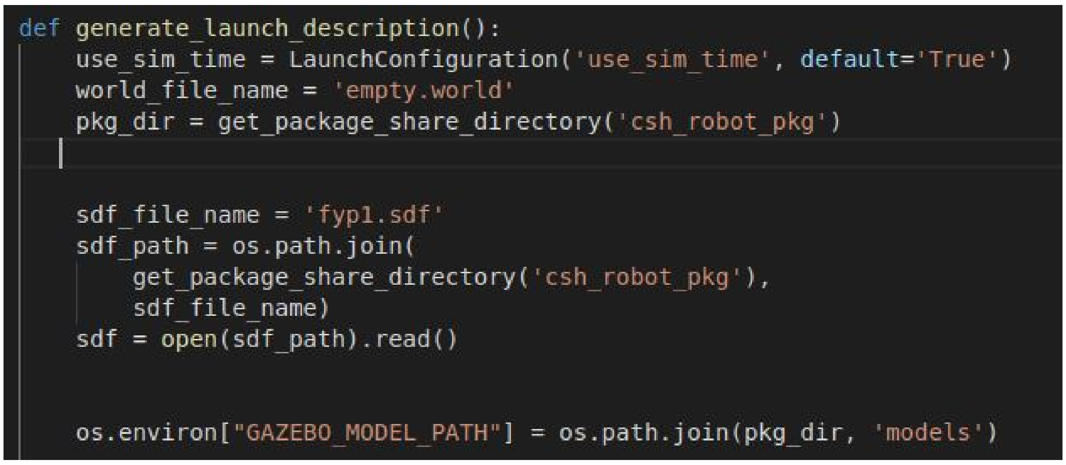

```
def generate_launch_description():
    use_sim_time = LaunchConfiguration('use_sim_time', default='True')
    world_file_name = 'empty.world'
    pkg_dir = get_package_share_directory('csh_robot_pkg')


    sdf_file_name = 'fyp1.sdf'
    sdf_path = os.path.join(
        get_package_share_directory('csh_robot_pkg'),
        sdf_file_name)
    sdf = open(sdf_path).read()


    os.environ["GAZEBO_MODEL_PATH"] = os.path.join(pkg_dir, 'models')
```

Figure 3.10: Parsing world extension file and SDF file.

ferent from the common *differential_drive* plugin that applies to 2 wheels robot, *skid_steer_drive* plugin is a Gazebo plugin that is suitable to be used with 4 wheels robot. The distance between the wheels and the links that represent the wheels are set in the plugin section in the SDF file. A topic named "*/demo/cmd_demo*" will be published by the plugin to allow the control of the motion of the wheels.

3.4.7 Stereo Camera Visualization

In order to perform the stereo camera visualization, a camera needs to be attached to the end-effector of the delta robot arm. The camera physical structure mesh used in this project is Intel Realsense camera. The camera is attached to the end-effector through a joint and the coordinate of the camera is adjusted so that the view of the camera is facing down. The *stereo_camera* sensor code is added to the link of the stereo camera and a *stereo_camera* plugin is implemented in the code in the SDF file to create topics that will publish the images captured by the stereo camera. The distortion and baseline value of the stereo camera is set in the code to ensure a pair of clear stereo images can be captured.

3.4.8 Delta Robot Arm Kinematic Motion

The kinematic motion performed by the delta robot arm in this project is implemented through the approach of implementing a Gazebo plugin named "*tricycle_drive*" that will control three motors that are connected to all three kinematic chains. By adding the code of the plugin in the SDF file and setting up the motors to be controlled, a topic named "*/demo/cmd_arm*" will be published to allow the motion control of the delta robot arm in terms of velocity.

3.5 SUMMARY

In short, the research methodology of the project, the software included during the project development process, and the major stages in the development processes of this project based on the project flowchart are explained in this chapter.

CHAPTER 4

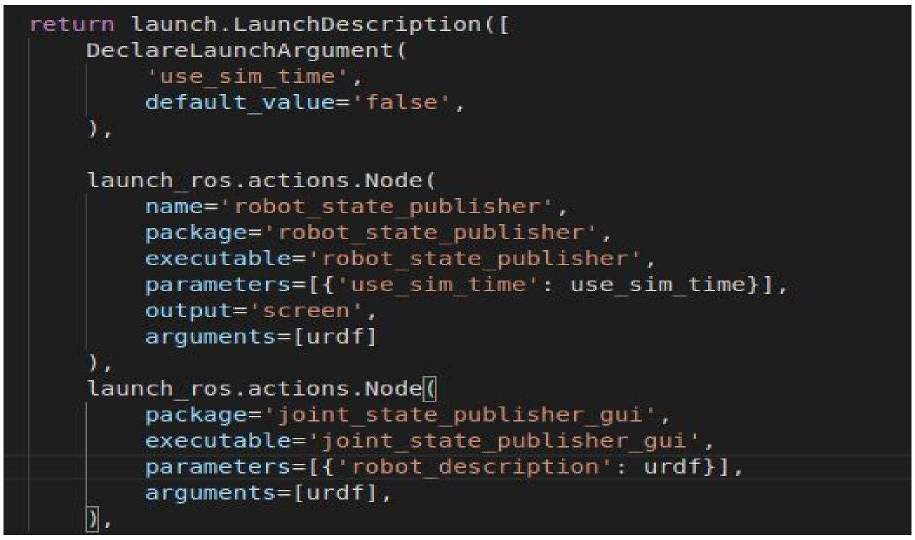
RESULTS AND DISCUSSION

This chapter records the results obtained throughout the development process of the project. The obtained result is then analyzed and discussed in this chapter. The sustainability of the project is also discussed.

4.1 SPAWNING DELTA ROBOT

4.1.1 Rviz 2

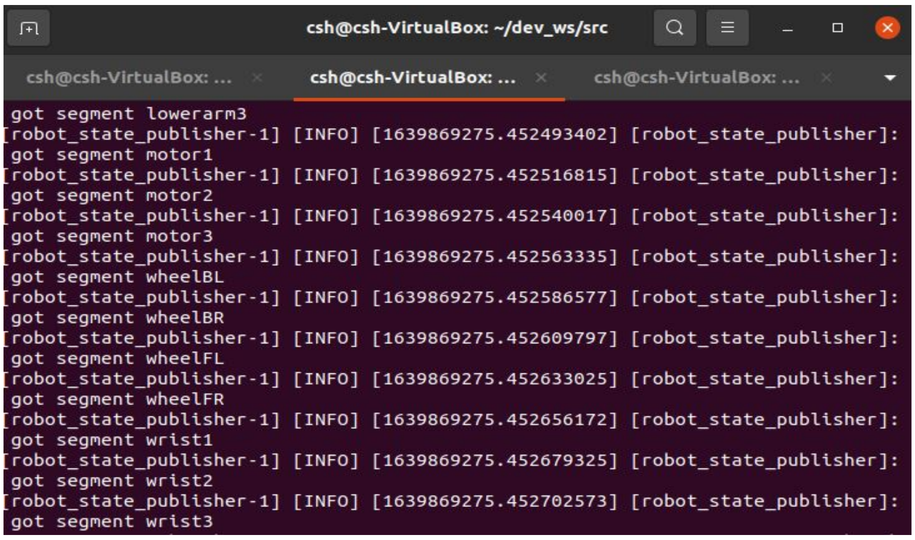
ROS Visualization (RVIZ) is a visualization platform that allows users to view the robot state and the sensor output such as image or point cloud. In order to view the robot state in Rviz 2, *robot_state_publisher* node needs to be run to allow Rviz 2 to parse the structure of the robot. *robot_state_publisher* uses the Unified Robot Description Format (URDF) file specified by the *robot_description* parameter as shown in Figure 4.1 . The *robot_state_publisher* and *joint_state_publisher_gui* are declared in a launch file to allow the launching of two nodes at the same time.



```
return launch.LaunchDescription([
    DeclareLaunchArgument(
        'use_sim_time',
        default_value='false',
    ),

    launch_ros.actions.Node(
        name='robot_state_publisher',
        package='robot_state_publisher',
        executable='robot_state_publisher',
        parameters=[{'use_sim_time': use_sim_time}],
        output='screen',
        arguments=[urdf]
    ),
    launch_ros.actions.Node(
        package='joint_state_publisher_gui',
        executable='joint_state_publisher_gui',
        parameters=[{'robot_description': urdf}],
        arguments=[urdf],
    ),
```

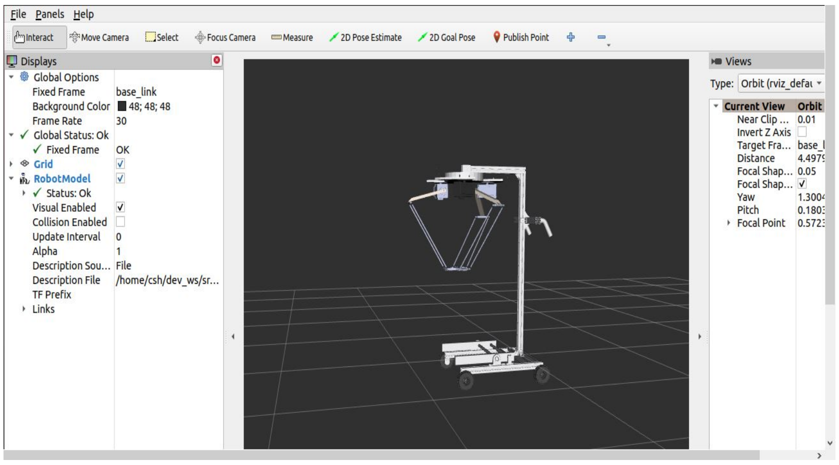
Figure 4.1: ROS nodes declared in a launch file.

As result, the links of the delta robot read by the *robot_state_publisher* will be printed on the terminal as shown in Figure 4.2 and the delta robot will be viewed in Rviz 2 as shown in Figure 4.3.



```
csh@csh-VirtualBox: ~/dev_ws/src
got segment lowerarm3
[robot_state_publisher-1] [INFO] [1639869275.452493402] [robot_state_publisher]:
got segment motor1
[robot_state_publisher-1] [INFO] [1639869275.452516815] [robot_state_publisher]:
got segment motor2
[robot_state_publisher-1] [INFO] [1639869275.452540017] [robot_state_publisher]:
got segment motor3
[robot_state_publisher-1] [INFO] [1639869275.452563335] [robot_state_publisher]:
got segment wheelBL
[robot_state_publisher-1] [INFO] [1639869275.452586577] [robot_state_publisher]:
got segment wheelBR
[robot_state_publisher-1] [INFO] [1639869275.452609797] [robot_state_publisher]:
got segment wheelFL
[robot_state_publisher-1] [INFO] [1639869275.452633025] [robot_state_publisher]:
got segment wheelFR
[robot_state_publisher-1] [INFO] [1639869275.452656172] [robot_state_publisher]:
got segment wrist1
[robot_state_publisher-1] [INFO] [1639869275.452679325] [robot_state_publisher]:
got segment wrist2
[robot_state_publisher-1] [INFO] [1639869275.452702573] [robot_state_publisher]:
got segment wrist3
```

Figure 4.2: Output of robot_state_publisher printed on terminal.

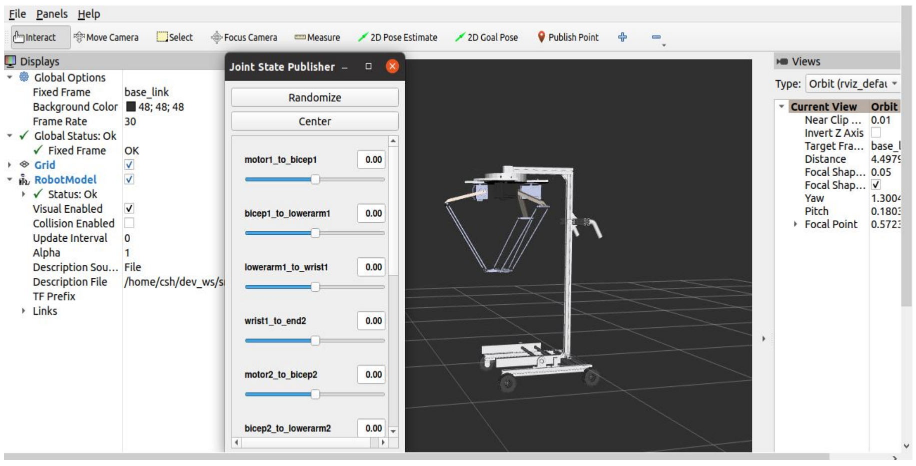


```
Global Options
  Fixed Frame          base_link
  Background Color     48; 48; 48
  Frame Rate           30
Global Status: Ok
  Fixed Frame          OK
Grid                   ✓
RobotModel             ✓
  Status: Ok
  Visual Enabled       ✓
  Collision Enabled
  Update Interval      0
  Alpha                1
  Description Sou...   File
  Description File     /home/csh/dev_ws/sr...
  TF Prefix
  Links
```

```
Type: Orbit (rviz_defau
Current View    Orbit
  Near Clip ...   0.01
  Invert Z Axis
  Target Fra...   base_l
  Distance        4.497
  Focal Shap...   0.05
  Focal Shap...   ✓
  Yaw             1.3004
  Pitch           0.1803
  Focal Point     0.5723
```

Figure 4.3: Robot state visualization in Rviz 2.

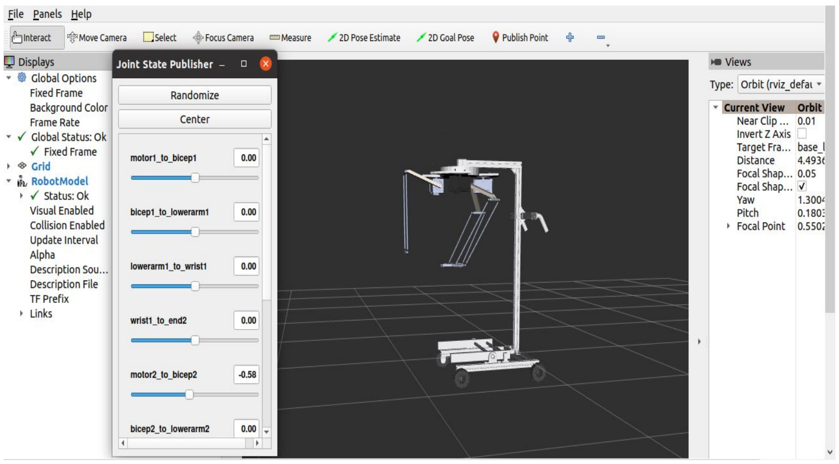
joint_state_publisher is a ROS package that reads the *robot_description* parameter to find all the non-fixed joints and publishes a JointState message. The *joint_state_publisher_* node will then be run to show the Graphical User Interface (GUI) of Joint State Publisher as shown in Figure 4.4 that can be used to adjust the angular rotational degree of the non-fixed joints.



```
motor1_to_bicep1       0.00
bicep1_to_lowerarm1    0.00
lowerarm1_to_wrist1    0.00
wrist1_to_end2         0.00
motor2_to_bicep2       0.00
bicep2_to_lowerarm2    0.00
```

Figure 4.4: Joint State Publisher GUI.

However, as URDF only supports the robot kinematic chain structure as a tree structure, hence the closed-loop kinematic chain linkage of a delta robot is unable to be achieved by using URDF. When the sliders in *joint_state_publisher_gui* are varied to change the angular rotational degree of the joints, the connection of the arm can be seen as broken as shown in Figure 4.5.



```
motor1_to_bicep1      0.00
bicep1_to_lowerarm1   0.00
lowerarm1_to_wrist1   0.00
wrist1_to_end2        0.00
motor2_to_bicep2      -0.58
bicep2_to_lowerarm2   0.00
```

```
Type: Orbit (rviz_defau
Current View   Orbit
Near Clip ...  0.01
Invert Z Axis  ☐
Target Fra...  base_l
Distance       4.4936
Focal Shap...  0.05
Focal Shap...  ✓
Yaw            1.3004
Pitch          0.1803
Focal Point    0.5502
```

Figure 4.5: Broken connection of delta robot arm.

Based on the result obtained, shown that URDF is not suitable to be used to describe the robot physical structure of delta robot that consists of closed-loop kinematic chain linkage as it can only parse the robot structure as a tree structure.

4.1.2 Gazebo

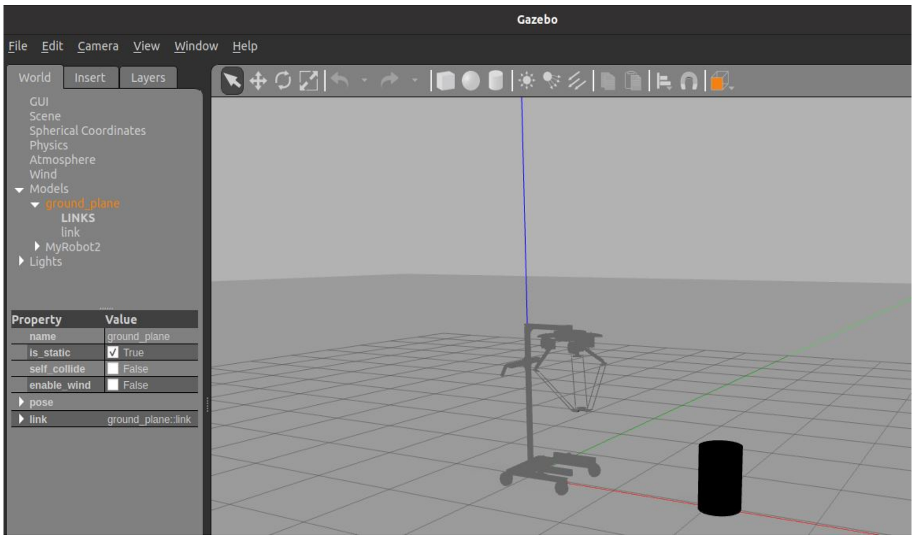
From the result obtained by using URDF to describe the delta robot and visualize it in Rviz 2, it is proved that URDF is not suitable to be used in this project to describe the robot structure of the delta robot. Hence, the Simulation Description Format (SDF) file will be used to describe the robot physical structure of the delta robot in this project.

Chapter 3 has explained the creation of a launch file for spawning the delta robot in Gazebo by parsing the SDF file of the robot. The delta robot in this project can be spawned in Gazebo by running the following command in the terminal:

ros2 launch csh_robot_pkg gazebo_world.launch.py

The *ros2launch* command will start the entire system including the Gazebo world and all the nodes called in the launch file. By specifying the name of the package right after the command *ros2launch*, will allow the system to search within the package to get the launch file named "gazebo_world.launch.py".

The output of the command is shown in Figure 4.6 where the empty. world world extension file that consists only of the ground plane is launched in the Gazebo. The delta robot is also successfully launched at the origin coordinate in the Gazebo.



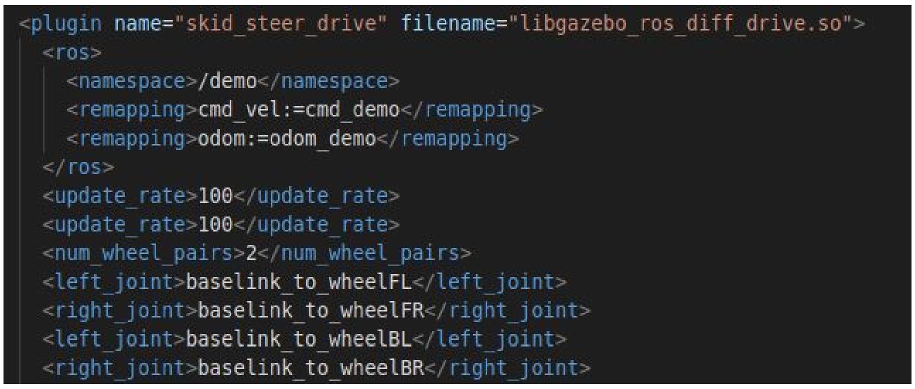
Property	Value
name	ground_plane
is_static	✓ True
self_collide	False
enable_wind	False
pose	
link	ground_plane::link

Figure 4.6: Robot spawned in Gazebo.

As the packages such as *robot_state_publisher* and *joint_state_publisher*$_g$*ui* depend on the *robot_description* to be run while *robot_description* get the parameters by parsing URDF file, hence *robot_state_publisher* and *joint_state_publisher*$_g$*ui* are unable to be used together with SDF file. The robot state of the delta robot described in the SDF file is then will not be viewed in Rviz 2. Therefore, the simulation of the delta robot in this project will be carried out in Gazebo.

4.2 WHEEL MOTION CONTROL

The wheel motion control of the delta robot is performed through the implementation of the Gazebo plugin in the SDF file of the delta robot. The gazebo plugin used for the wheel motion control in this project is the *skid_steer_drive* plugin which can be used to control four wheels. In the SDF file, the plugin is added with the tag <plugin>, and the .so file which is the object file of the plugin library is specified to allow the system to run the function in the library code. The wheels joints to be controlled are specified in the plugin code as shown in Figure 4.7.



```
<plugin name="skid_steer_drive" filename="libgazebo_ros_diff_drive.so">
  <ros>
    <namespace>/demo</namespace>
    <remapping>cmd_vel:=cmd_demo</remapping>
    <remapping>odom:=odom_demo</remapping>
  </ros>
  <update_rate>100</update_rate>
  <update_rate>100</update_rate>
  <num_wheel_pairs>2</num_wheel_pairs>
  <left_joint>baselink_to_wheelFL</left_joint>
  <right_joint>baselink_to_wheelFR</right_joint>
  <left_joint>baselink_to_wheelBL</left_joint>
  <right_joint>baselink_to_wheelBR</right_joint>
```

Figure 4.7: skid_steer_drive plugin in SDF file.

The *skid_steer_drive* plugin will publish a topic named /demo/cmd_demo. The /demo is indicated in the <namespace> tag and the cmd_demo is the remapping of the cmd_vel which is a topic that will be used for controlling the velocity of a joint.

There are several approaches that can be implemented to send the velocity command to the topic published by the plugin. For example, the velocity command can be sent through the geometry_msgs/Twist message by running the command line:

ros2 topic pub /demo/cmd_demo geometry_msgs/Twist '{linear: {x: 1.0}}' -1

This command line will allow the robot to move in the direction towards the x-axis with a velocity of 1 meter per second.

Besides, the wheel motion of the robot can be controlled through the robot steering function equipped in the rqt. The Graphical User Interface (GUI) for the robot steering as shown in Figure 4.8 can be found in rqt>Plugins>Robot Tools>Robot Steer. By moving the slider, the velocity of the robot can be varied. This approach is also sending the geometry_msgs/Twist message to the topic published by the plugin.

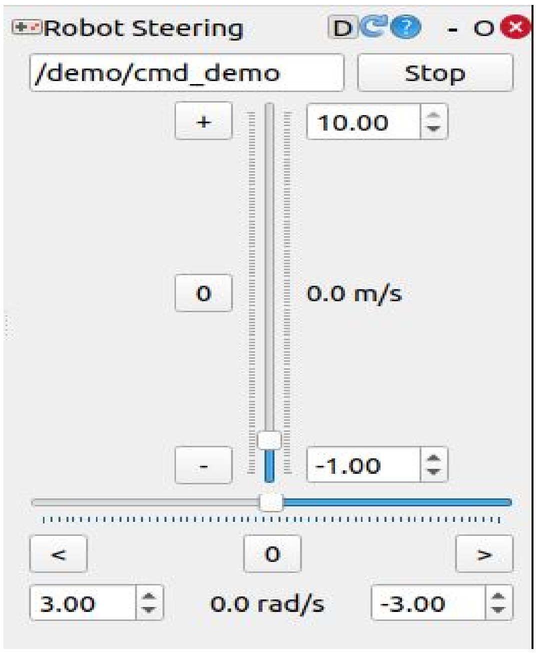


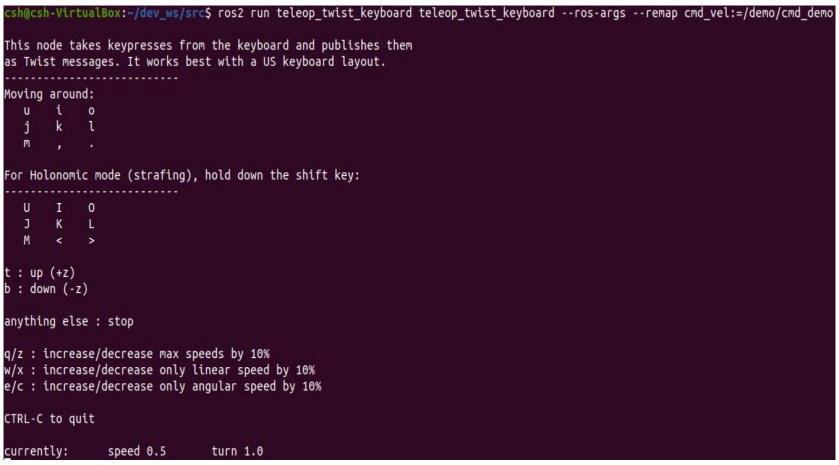
Figure 4.8: Robot steering GUI.

In this project, the approach chosen to control the wheel motion is through the *teleop_twist_keyboard* package. The node of *teleop_twist_keyboard* can be run by running the command line:

ros2 run teleop_twist_keyoboard teleop_twist_keyboard - - ros-args

- -remap cmd_vel:=/demo/cmd_demo

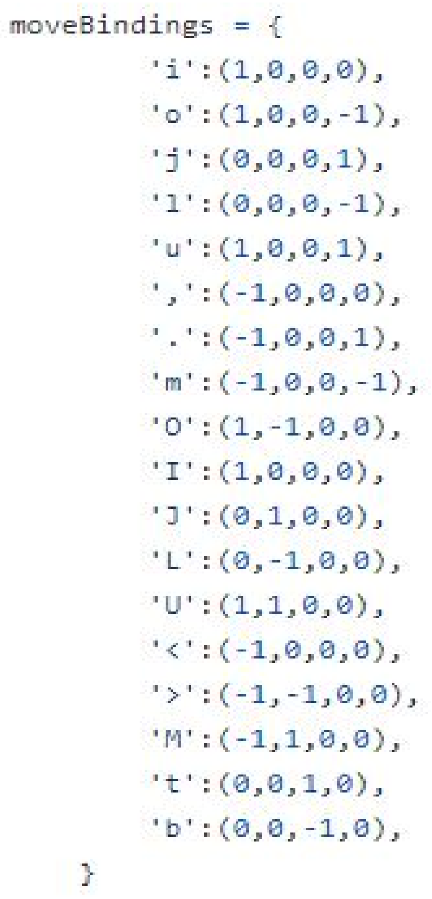
The instruction of using the *teleop_twist_keyboard* will be printed on the terminal as shown in Figure 4.9.



```
csh@csh-VirtualBox:~/dev_ws/src$ ros2 run teleop_twist_keyboard teleop_twist_keyboard --ros-args --remap cmd_vel:=/demo/cmd_demo

This node takes keypresses from the keyboard and publishes them
as Twist messages. It works best with a US keyboard layout.
---------------------------
Moving around:
   u    i    o
   j    k    l
   m    ,    .

For Holonomic mode (strafing), hold down the shift key:
---------------------------
   U    I    O
   J    K    L
   M    <    >

t : up (+z)
b : down (-z)

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit

currently:      speed 0.5       turn 1.0
```

Figure 4.9: Output of teleop_twist_keyboard node on terminal.

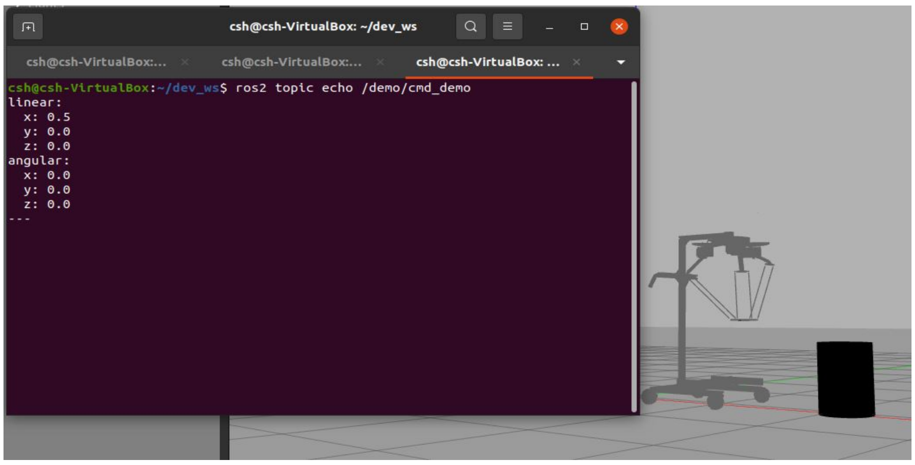
Figure 4.10 shows a section in the *teleop_twist_keyboard* source code that explains the function of each of the keys in the keyboard.



```
moveBindings = {
        'i':(1,0,0,0),
        'o':(1,0,0,-1),
        'j':(0,0,0,1),
        'l':(0,0,0,-1),
        'u':(1,0,0,1),
        ',':(-1,0,0,0),
        '.':(-1,0,0,1),
        'm':(-1,0,0,-1),
        'O':(1,-1,0,0),
        'I':(1,0,0,0),
        'J':(0,1,0,0),
        'L':(0,-1,0,0),
        'U':(1,1,0,0),
        '<':(-1,0,0,0),
        '>':(-1,-1,0,0),
        'M':(-1,1,0,0),
        't':(0,0,1,0),
        'b':(0,0,-1,0),
    }
```

Figure 4.10: Explanation of each key on the keyboard.

When the key "i" is pressed, the wheels of the robot will move towards the x-axis with a velocity of 0.5 meters per second. By echoing the topic published by the plugin, it can be seen that the topic is receiving the output from *teleop_twist_keyboard* as shown in Figure 4.11.



```
csh@csh-VirtualBox:~/dev_ws$ ros2 topic echo /demo/cmd_demo
linear:
  x: 0.5
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
---
```

Figure 4.11: Echo of the topic /demo/cmd_demo.

By using the Gazebo Plotting Utility feature, the Velocity vs Sim Time graph is plotted as shown in Figure 4.12. Some overshoots can be observed at the starting time of the graph as the robot starts to change its state from a static state to a moving state. However, when the velocity is varied after the starting time, the overshoot is seen to be reduced for every time of variation.

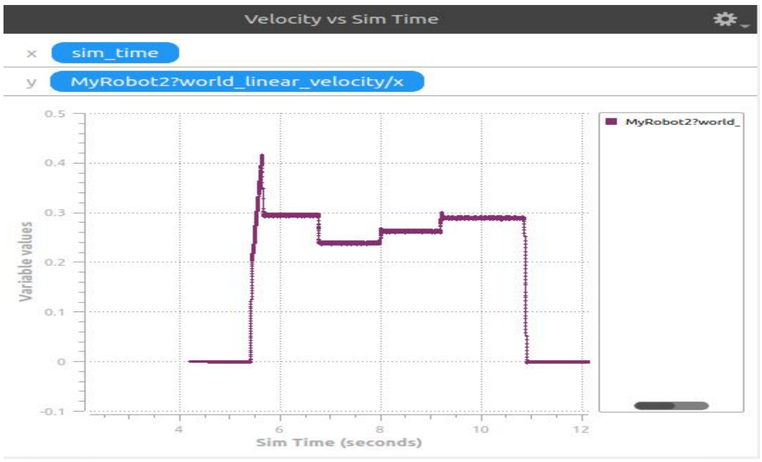
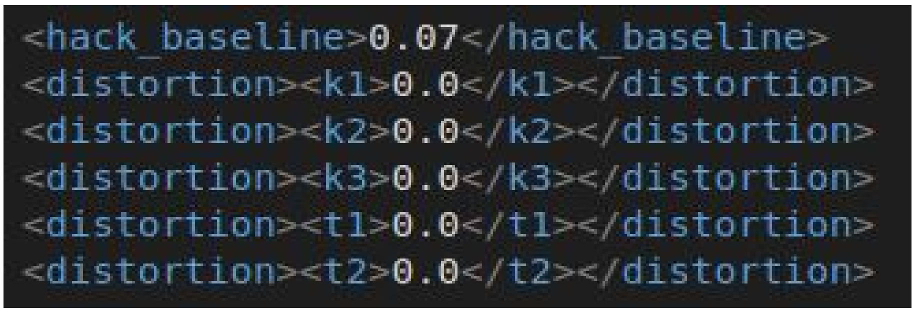


Figure 4.12: Velocity vs Sim Time graph.

4.3 STEREO CAMERA VISUALIZATION

The stereo visualization of the delta robot is performed through the implementation of the stereo camera sensor and stereo camera plugin in the SDF file. In the plugin code, the baseline is set to 0.07 while the distortion is set to 0 as shown in Figure 4.13 to generate a pair of clear images without distortion. The baseline of the stereo camera represents the distance between the center of the left and right imagers. By referring to the Intel Realsense D400 Series Product Family Datasheet [53], the baseline value for the stereo depth module is suggested to be in the range of 20mm to 70mm. Hence, the baseline for the stereo camera in this project is set to 70mm which is 0.07m.



```
<hack_baseline>0.07</hack_baseline>
<distortion><k1>0.0</k1></distortion>
<distortion><k2>0.0</k2></distortion>
<distortion><k3>0.0</k3></distortion>
<distortion><t1>0.0</t1></distortion>
<distortion><t2>0.0</t2></distortion>
```

Figure 4.13: Base line and distortion value set in plugin.

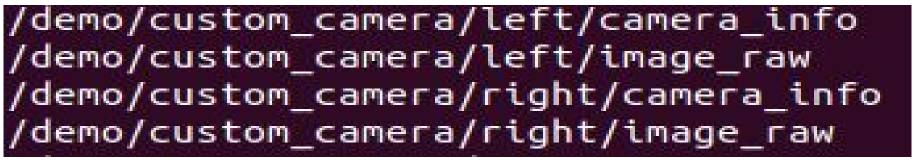
The stereo camera plugin will publish several topics such as

$/demo/custom\_camera/left/camera\_info$,

$/demo/custom\_camera/right/camera\_info$,

$/demo/custom\_camera/left/image\_raw$ and

$/demo/custom\_camera/right/image\_raw$ as shown in Figure 4.14



```
/demo/custom_camera/left/camera_info
/demo/custom_camera/left/image_raw
/demo/custom_camera/right/camera_info
/demo/custom_camera/right/image_raw
```

Figure 4.14: Topics published by stereo camera plugin.

To allow the visualization of the stereo images captured by the stereo camera in Rviz 2, the topics of left and right /image_raw need to be subscribed as shown in Figure 4.15.

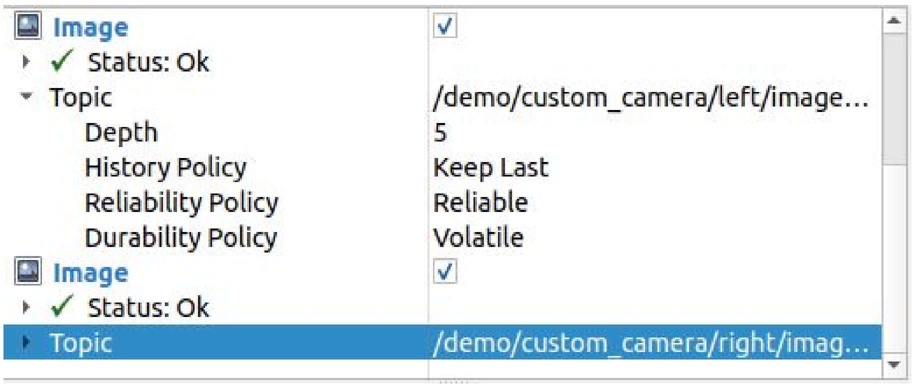


Image	✓
✓ Status: Ok	
Topic	/demo/custom_camera/left/image...
Depth	5
History Policy	Keep Last
Reliability Policy	Reliable
Durability Policy	Volatile
Image	✓
✓ Status: Ok	
Topic	/demo/custom_camera/right/imag...

Figure 4.15: Subscription of stereo camera topic in Rviz 2.

The stereo camera visualization of the robot is successfully generated where the output of the left and right images captured by the stereo camera attached to the delta robot arm is shown in Figure 4.16. The top image represents the left image and the bottom image represents the right image.

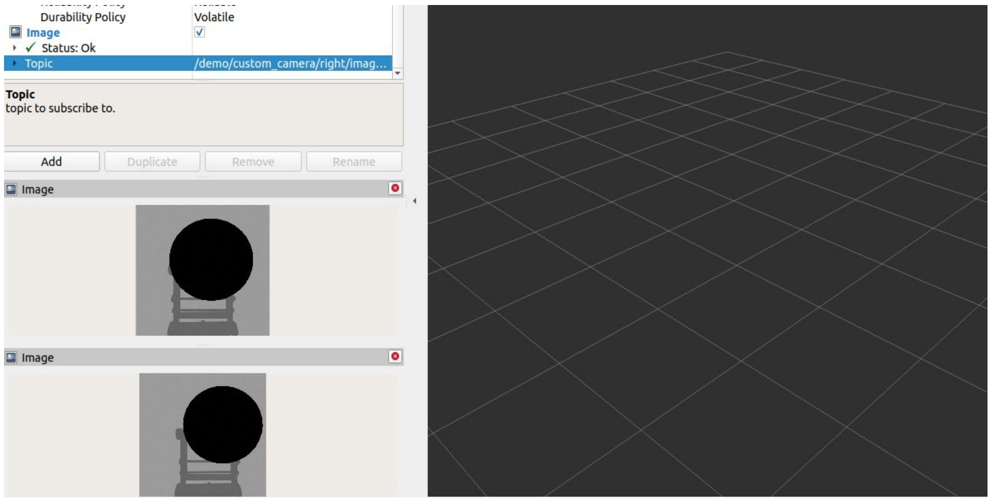
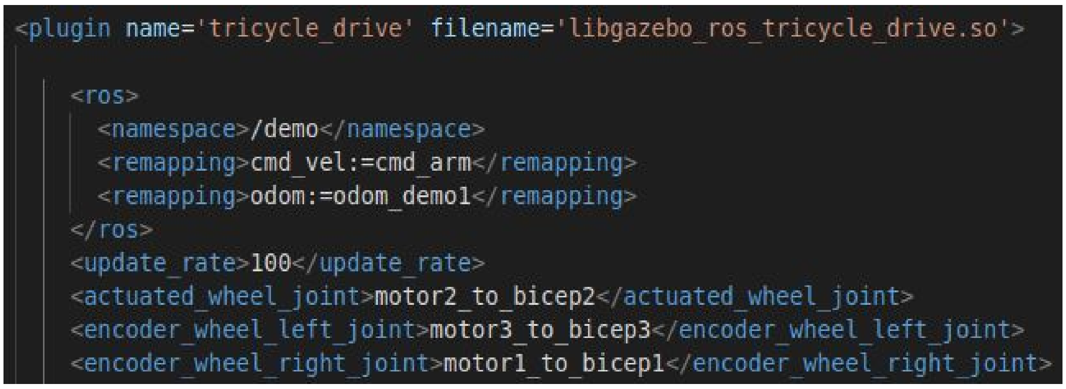


Figure 4.16: Left and right images generated in Rviz 2.

4.4 DELTA ROBOT ARM KINEMATIC MOTION

In this project, the kinematic motion performed by the delta robot arm is implemented through the approach of implementing a Gazebo plugin named *tricycle_drive* that will control the rotational velocity of three motors. In the plugin code that is added to the SDF file, the joints that are connecting the motors to biceps are specified as shown in Figure 4.17.



```
<plugin name='tricycle_drive' filename='libgazebo_ros_tricycle_drive.so'>

    <ros>
      <namespace>/demo</namespace>
      <remapping>cmd_vel:=cmd_arm</remapping>
      <remapping>odom:=odom_demo1</remapping>
    </ros>
    <update_rate>100</update_rate>
    <actuated_wheel_joint>motor2_to_bicep2</actuated_wheel_joint>
    <encoder_wheel_left_joint>motor3_to_bicep3</encoder_wheel_left_joint>
    <encoder_wheel_right_joint>motor1_to_bicep1</encoder_wheel_right_joint>
```

Figure 4.17: Joints specified in plugin code.

teleop_twist_keyboard will be used to send the velocity command to the motors that are connecting to three of the kinematic chains. The topic that is published by the *tricycle_drive* plugin is */demo/cmd_arm* which is the remapping of *cmd_vel*. The command line to run the *teleop_twist_keyboard* to control the motors is by specifying the topic published by the plugin:

ros2 run teleop_twist_keyoboard teleop_twist_keyboard - - *ros-args*

- -*remap cmd_vel:=/demo/cmd_arm*

Figure 4.18 shows the labeled delta robot biceps with the XYZ frame in the Gazebo. Based on the plugin code, the velocity command will be sent to all three joints where the joint between motor2 and bicep 2 is the main actuator while the joint between motor 1 and bicep 1 and the joint between motor 3 and bicep3 are the encoder.

As motor 2 is facing toward the X-axis, hence, to perform the kinematic motion of the delta robot arm along the Z-axis, the joint *motor2_to_bicep2* need to rotate clockwise toward the direction of the X-axis to allow the end-effector to move down and rotate counter-clockwise toward the direction of negative X-axis to move the end-effector up. Based on the key function stated in *teleop_twist_keyboard* source code as shown in Figure 4.10, the key "i" will be pressed to allow the end-effector to move down while the key "," will be pressed to allow the end-effector to move up.

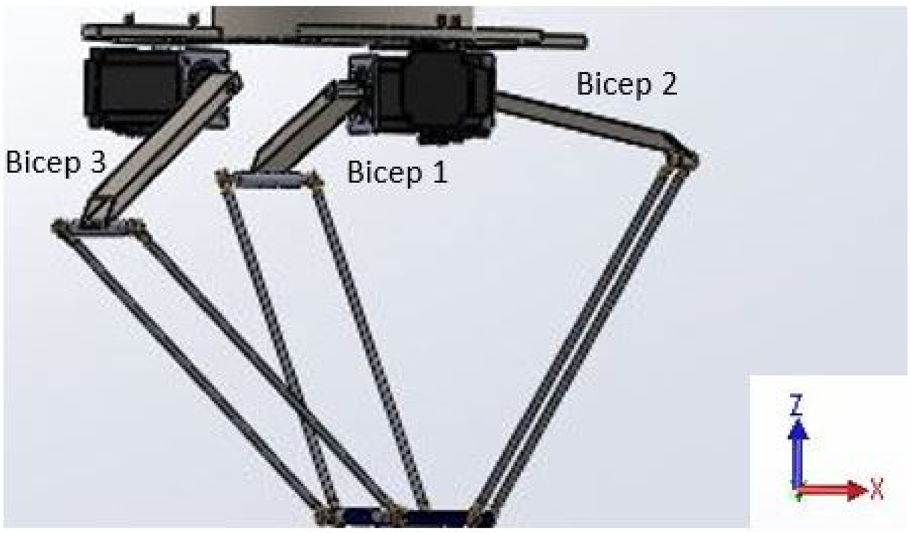


Figure 4.18: Labeled bicep of delta robot arm.

4.4.1 Kinematic Motion Analysis

In this project, inverse kinematic is performed where the position of the end-effector is known and the joint angle of the motors will be determined.

A cylindrical object is placed at the coordinate of (2 0 0) in the Gazebo world as the target object that the end-effector will reach. After the delta robot is driven to the location where the end-effector is located right above the object, the kinematic motion of the delta robot arm is performed.

By varying the height of the object to indicate the position of the end-effector along the Z-axis, the positions of the joints of three motors in terms of angular rotational degree (radian) are recorded.

4.4.1.1 Initial Angle Offset of Biceps

Ideally, the angular rotational degree of three of the motors will be 0 degrees when the biceps are parallel to the floor. From Figure 4.18, it can be seen that three of the biceps are not parallel to the floor. As the SDF file is converted from the URDF file exported from SOLIDWORKS software, hence the robot state in the SOLIDWORKS software will become the initial position of the delta robot where the current position of the biceps with offset will be indicated as 0 degrees during the simulation. The offset angle of each of the biceps is identified by using the measuring tool in SOLIDWORKS software. The offset angle is computed through the trigonometric formed at the motors as shown in Figure 4.19, Figure 4.20 and Figure 4.21.

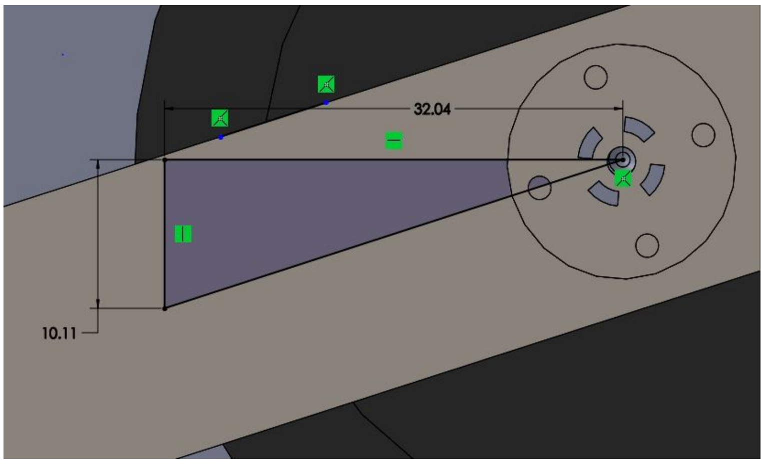


Figure 4.19: Right-angled triangle formed at Bicep 1.

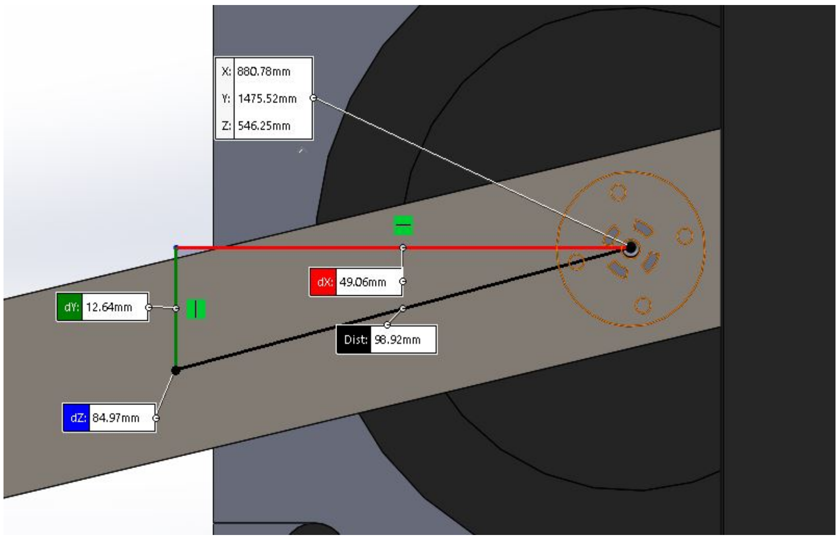


Figure 4.20: Right-angled triangle formed at Bicep 2.

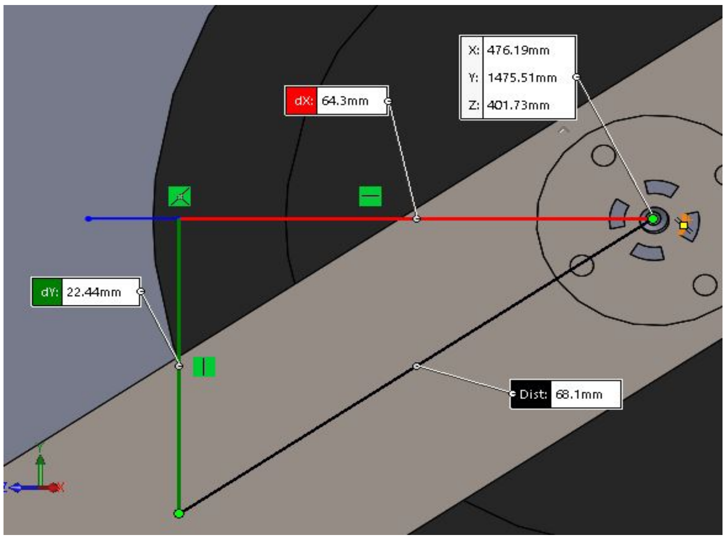


Figure 4.21: Right-angled triangle formed at Bicep 3.

By applying the trigonometric theorem, the offset angles of three biceps are computed and recorded in Table 4.1.

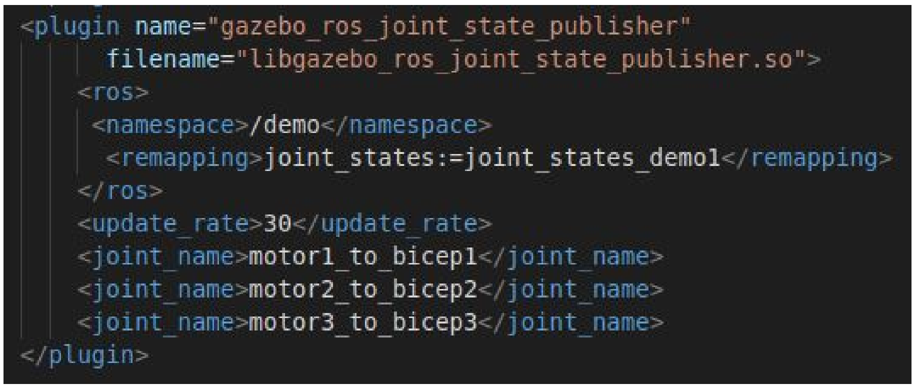
Table 4.1: Offset angle of biceps.

Bicep	Offset angle (degree)	Offset angle (radian)
Bicep 1	17.513	0.3057
Bicep 2	14.448	0.2522
Bicep 3	19.238	0.3358

4.4.1.2 Kinematic Motion Simulation Result

The height of the cylindrical object is varied for five different values to analyze the angular rotational degree of the motor when the end-effector is located at a different coordinate.

A gazebo plugin named "*gazebo_ros_joint_state_publisher*" is added to the SDF file of the delta robot to read the joint state of the joints specified in the code as shown in Figure 4.22.



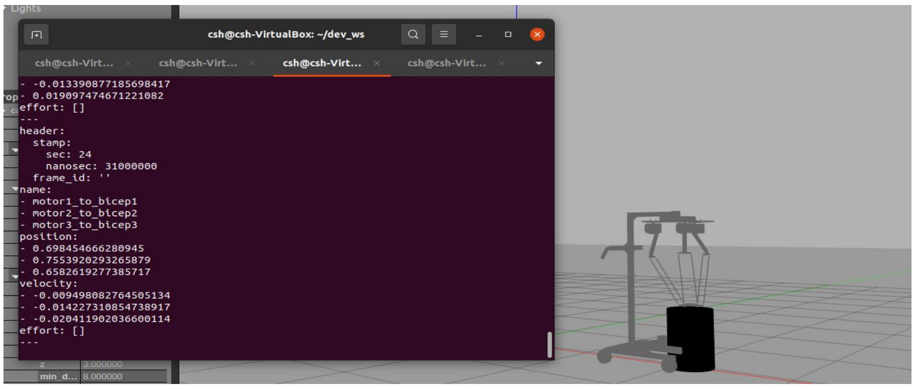
```
<plugin name="gazebo_ros_joint_state_publisher"
      filename="libgazebo_ros_joint_state_publisher.so">
    <ros>
     <namespace>/demo</namespace>
      <remapping>joint_states:=joint_states_demo1</remapping>
    </ros>
    <update_rate>30</update_rate>
    <joint_name>motor1_to_bicep1</joint_name>
    <joint_name>motor2_to_bicep2</joint_name>
    <joint_name>motor3_to_bicep3</joint_name>
</plugin>
```

Figure 4.22: *gazebo_ros_joint_state_publisher* plugin code.

The information such as the joint name, position, and velocity published by the *gazebo_ros_joint_state_publisher* plugin can be viewed by echoing the topic published by the plugin which is $/demo/joint\_states\_demo1$. The following command line is run:

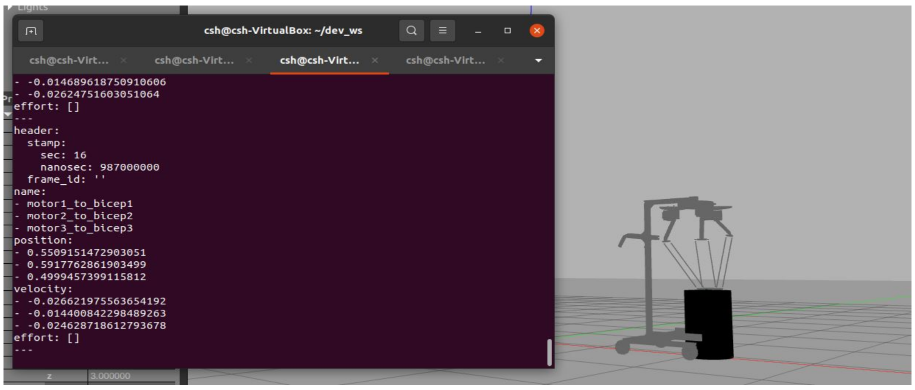
ros2 topic echo /demo/joint_states_demo1

Although this plugin has almost the same name as the *joint_states_publisher* package that parses URDF files, they are actually two different packages as this plugin is applicable for the SDF file and has a different source code from the package that parses URDF file. The joint positions of the three joints that connected in between motor and bicep are printed in terms of angular rotational degree in radian on the terminal as shown in Figure 4.23, Figure 4.24, Figure 4.25, Figure 4.26 and Figure 4.27.



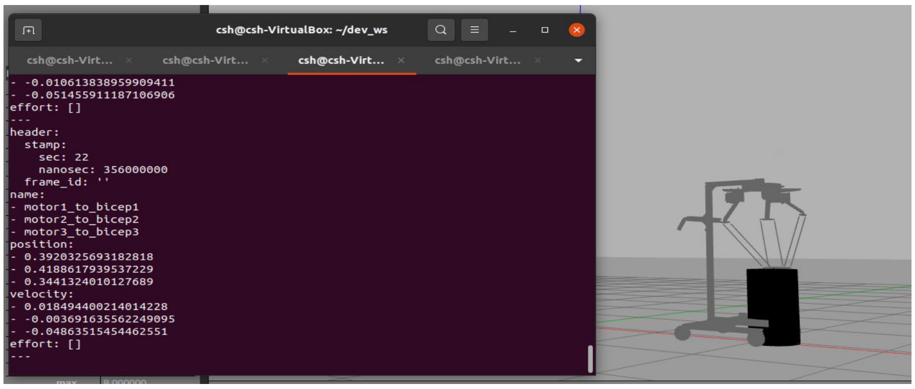
```
- -0.013390877185698417
- 0.019097474671221082
effort: []
---
header:
  stamp:
    sec: 24
    nanosec: 31000000
  frame_id: ''
name:
- motor1_to_bicep1
- motor2_to_bicep2
- motor3_to_bicep3
position:
- 0.698454666280945
- 0.7553920293265879
- 0.6582619277385717
velocity:
- -0.009498082764505134
- -0.014227310854738917
- -0.020411902036600114
effort: []
---
```

Figure 4.23: Joints position when end-effector at Z=550mm.



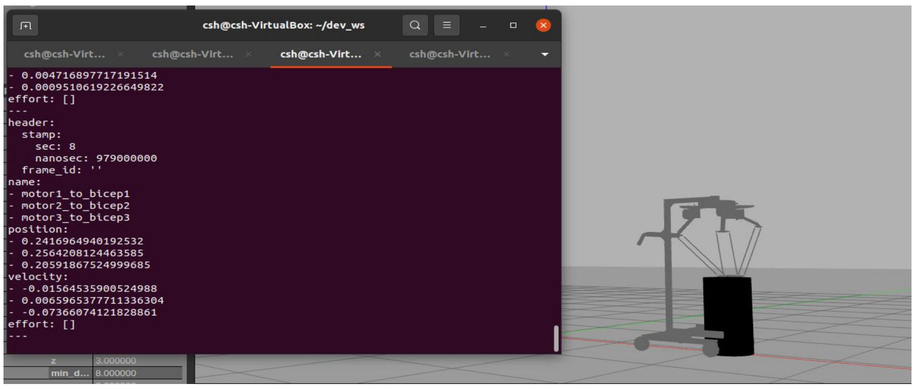
```
- -0.014689618750910606
- -0.02624751603051064
effort: []
---
header:
  stamp:
    sec: 16
    nanosec: 987000000
  frame_id: ''
name:
- motor1_to_bicep1
- motor2_to_bicep2
- motor3_to_bicep3
position:
- 0.5509151472903051
- 0.5917762861903499
- 0.4999457399115812
velocity:
- -0.026621975563654192
- -0.014400842298489263
- -0.024628718612793678
effort: []
---
```

Figure 4.24: Joints position when end-effector at Z=600mm.



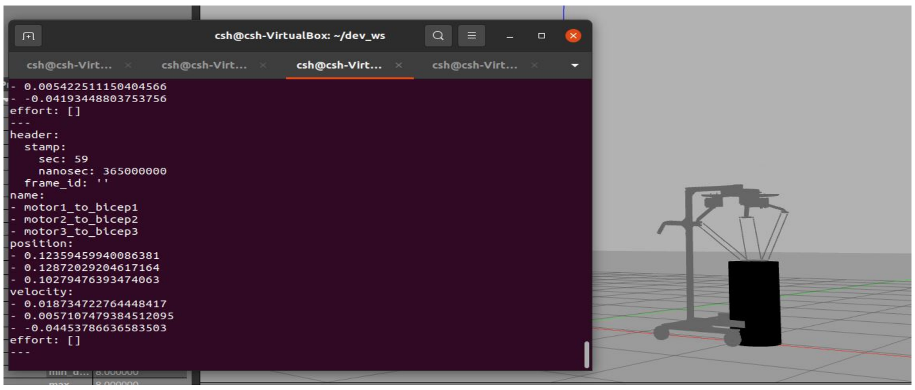
```
- -0.010613838959909411
- -0.051455911187106906
effort: []
---
header:
  stamp:
    sec: 22
    nanosec: 356000000
  frame_id: ''
name:
- motor1_to_bicep1
- motor2_to_bicep2
- motor3_to_bicep3
position:
- 0.3920325693182818
- 0.4188617939537229
- 0.3441324010127689
velocity:
- 0.018494400214014228
- -0.003691635562249095
- -0.04863515454462551
effort: []
---
```

Figure 4.25: Joints position when end-effector at Z=650mm.



```
- 0.004716897717191514
- 0.0009510619226649822
effort: []
---
header:
  stamp:
    sec: 8
    nanosec: 979000000
  frame_id: ''
name:
- motor1_to_bicep1
- motor2_to_bicep2
- motor3_to_bicep3
position:
- 0.2416964940192532
- 0.2564208124463585
- 0.20591867524999685
velocity:
- -0.01564535900524988
- 0.0065965377711336304
- -0.07366074121828861
effort: []
---
```

Figure 4.26: Joints position when end-effector at Z=700mm.



```
- 0.005422511150404566
- -0.04193448803753756
effort: []
---
header:
  stamp:
    sec: 59
    nanosec: 365000000
  frame_id: ''
name:
- motor1_to_bicep1
- motor2_to_bicep2
- motor3_to_bicep3
position:
- 0.12359459940086381
- 0.12872029204617164
- 0.10279476393474063
velocity:
- 0.018734722764448417
- 0.0057107479384512095
- -0.04453786636583503
effort: []
---
```

Figure 4.27: Joints position when end-effector at Z=750mm.

The angular rotational degree (radian) of each of the joints are recorded and the theoretical value of the angular rotational degree are determined by using the online Rotary Delta Robot Forward/Inverse Kinematics Calculations website [54] by referring to the paper done by Y. Kadam et.al [55]. The results are tabulated in Table 4.2 where the error of percentages of each result are also identified.

Table 4.2: Comparison between theoretical value and measured value.

End-effector position along Z-axis (mm)	Theoretical value (rad)	Measured value + offset angle (rad)	Percentage error (%)

550	Θ1= 0.9752
	Θ2= 0.9752
	Θ3= 0.9752
		Θ1= 0.6985 + 0.3057 = 1.0042
		Θ2= 0.7554 + 0.2522 = 1.0066
		Θ3= 0.6583 + 0.3358 = 0.9941
			Θ1= 2.97
			Θ2= 3.22
			Θ3= 1.94

600	Θ1= 0.8140
	Θ2= 0.8140
	Θ3= 0.8140
		Θ1= 0.5509 + 0.3057 = 0.8566
		Θ2= 0.5918 + 0.2522 = 0.8440
		Θ3= 0.5000 + 0.3358 = 0.8358
			Θ1= 5.23
			Θ2= 3.69
			Θ3= 2.68

650	Θ1= 0.6713
	Θ2= 0.6713
	Θ3= 0.6713
		Θ1= 0.3920 + 0.3057 = 0.6977
		Θ2= 0.4189 + 0.2522 = 0.6711
		Θ3= 0.3441 + 0.3358 = 0.6799
			Θ1= 3.93
			Θ2= 0.03
			Θ3= 1.28

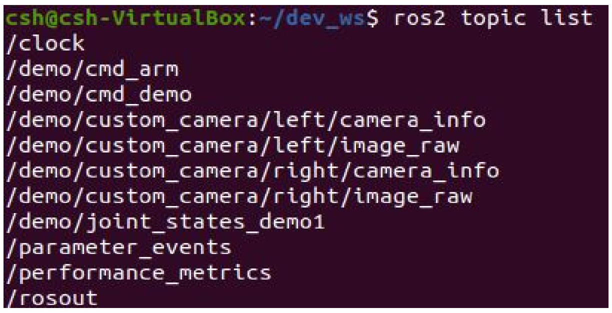
700	Θ1= 0.5373
	Θ2= 0.5373
	Θ3= 0.5373
		Θ1= 0.2417 + 0.3057 = 0.5474
		Θ2= 0.2564 + 0.2522 = 0.5086
		Θ3= 0.2059 + 0.3358 = 0.5417
			Θ1= 1.88
			Θ2= 5.34
			Θ3= 0.82

750	Θ1= 0.4065
	Θ2= 0.4065
	Θ3= 0.4065
		Θ1= 0.1236 + 0.3057 = 0.4293
		Θ2= 0.1287 + 0.2522 = 0.3809
		Θ3= 0.1028 + 0.3358 = 0.4386
			Θ1= 5.61
			Θ2= 6.30
			Θ3= 7.90

From the result obtained, the average percentage error of $\theta 1$ is 3.92 %, $\theta 2$ is 3.72 % and $\theta 3$ is 2.92 %.

4.4.2 ROS 2 Topic

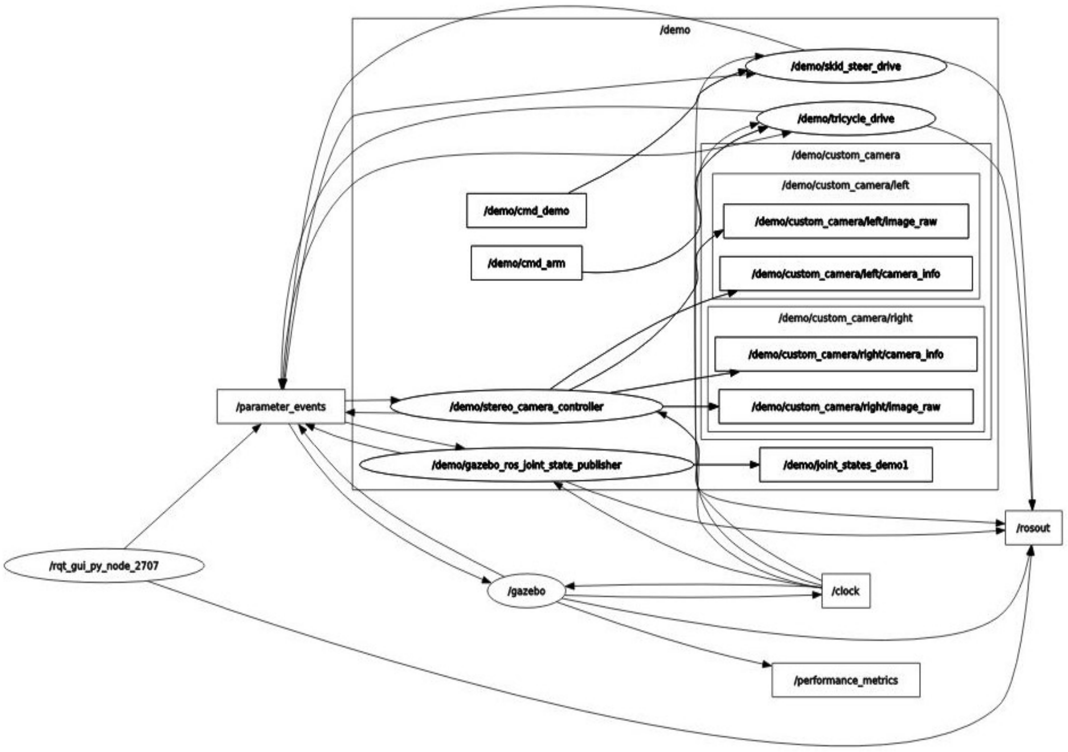
In this project, several topics have been published by different Gazebo plugins implemented. All the ROS 2 topics published are shown in the Figure 4.28 by running the command line: *ros*2 *topic list*



```
csh@csh-VirtualBox:~/dev_ws$ ros2 topic list
/clock
/demo/cmd_arm
/demo/cmd_demo
/demo/custom_camera/left/camera_info
/demo/custom_camera/left/image_raw
/demo/custom_camera/right/camera_info
/demo/custom_camera/right/image_raw
/demo/joint_states_demo1
/parameter_events
/performance_metrics
/rosout
```

Figure 4.28: ROS 2 topic list.

The relationship between each topic and node can be determined through the rqt_graph as shown in Figure 4.29.



/demo
/demo/skid_steer_drive
/demo/tricycle_drive
/demo/custom_camera
/demo/custom_camera/left
/demo/cmd_demo
/demo/custom_camera/left/image_raw
/demo/cmd_arm
/demo/custom_camera/left/camera_info
/demo/custom_camera/right
/demo/custom_camera/right/camera_info
/parameter_events
/demo/stereo_camera_controller
/demo/custom_camera/right/image_raw
/demo/gazebo_ros_joint_state_publisher
/demo/joint_states_demo1
/rosout
/rqt_gui_py_node_2707
/gazebo
/clock
/performance_metrics

Figure 4.29: rqt graph.

4.5 PROJECT SUSTAINABILITY

The aspect of sustainability of the project can be seen by the choice of the ROS 2 distro chosen which is the ROS 2 Foxy Fitzroy which has the longest End Of Life (EOL) date as shown in Figure2.2. Hence, this project can be further developed and modified by other users or organizations by using the same simulation environment. The controlling method for the delta robot arm kinematic motion can also be further improved by a customized Gazebo plugin written to increase the accuracy of the robot arm movement. Furthermore, the stereo camera visualization feature of this project can also be further modified by processing the images captured by the stereo camera to obtain information such as the distance between the camera and the object captured. The information obtained can be then implemented in the customized Gazebo plugin written to allow the automatic kinematic motion to be performed based on the coordinate of the object identified.

4.6 SUMMARY

In summary, the results obtained from the development of the project are recorded and explained in this chapter. The analysis of the data obtained is carried out and the rqt graph that shows the relationship between the topics generated is provided. Lastly, the project sustainability is discussed.

CHAPTER 5

CONCLUSION AND FUTURE WORKS

This chapter concludes the overall achievement of the project and the proposal of future works that can be carried out for the project improvement.

5.1 Conclusion

In conclusion, the delta robot arm is successfully simulated in ROS 2 Foxy Fitzroy environment set up in VirtualBox software. The primary robot structure description file, Unified Robot Description Format (URDF) is generated from the SOLIDWORKS software and has undergone conversion into Simulation Description Format (SDF) file to allow the forming of closed-loop kinematic chain linkage of delta robot. The stereo camera visualization feature of the delta robot is performed by adding a stereo camera link to the end-effector of the delta robot arm and implementing of stereo camera Gazebo plugin. Rviz 2 is used as the platform to view the pair of left and right images captured by the stereo camera.

The simulation of the kinematic motion of the delta robot arm is performed through the approach of implementing the *tricycle_drive* Gazebo plugin that controls three motors. The kinematic motion is controlled through the velocity command sent to the topic published by the plugin to control the rotation of joints connected in between motors and biceps. Through the implementation of the approach of using *tricyle_drive*

Gazebo plugin, $\theta 1$ has the average error of the percentage of 3.92 %, $\theta 2$ has 3.72 % and $\theta 3$ has 2.92 % during the performance of inverse kinematic motion.

5.2 Future Works

For future improvement of this project, a customized gazebo plugin can be written to allow full control for each of the three motors of the delta robot arm. The accuracy of the kinematic motion performed by the delta robot arm can be improved when the motors can be controlled individually by sending the desired angular rotational degree to each of the motors and allowing them to rotate accordingly. The customized gazebo plugin is written can be then contributed to the Robot Operating System 2 (ROS 2) community through GitHub to allow code reuse for other developers to perform a similar function in their project.

Furthermore, the left and right images captured by the stereo camera can be further processed by using *cv_bridge* to convert the ROS images to OpenCV images to extract the information of the images thus allowing the user to determine the distance between the stereo camera and the target object. The real-time distance identified can be inserted into the customized Gazebo plugin code to allow automatic kinematic motion of the delta robot arm based on the coordinate of the object.

REFERENCES

[1] B. William, “An Introduction to Robotics,” March 2021. [Online]. Available: https://www.ohio.edu/mechanical-faculty/williams/html/PDF/IntroRob.pdf

[2] A. Beasley, R, “Medical Robots: Current Systems and Research Directions,” *Journal of Robotics*, August 2012.

[3] P. Gonzalez-de Santos, R. Fernández, D. Sepúlveda, E. Navas, L. Emmi, and M. Armada, “Field Robots for Intelligent Farms- Inhering Features from Industry.” *Agronomy*, October 2020.

[4] K. D. Sowjanya, R. Sindhu, M. Parijatham, K. Srikanth, and P. Bhargav, “Multipurpose autonomous agricultural robot,” vol. 2, 2017, pp. 696–699.

[5] T. W. Bank, “Agriculture and Food,” April 2021. [Online]. Available: https://www.worldbank.org/en/topic/agriculture/overview

[6] “Selected Agricultural Indicators, Malaysia, 2020,” Department Of Statistics Malaysia Official Portal, May 2021. [Online]. Available: https://www.dosm.gov.my/v1/index.php?r=column/cthemeByCat&cat=72&bul_id=RXVKUVJ5TitHM0cwYWxlOHcxU3dKdz09&menu_id=Z0VTZGU1UHBUT1VJMFlpaXRRR0xpdz09

[7] R. Rahmadian and M. Widyartono, “Autonomous Robotic in Agriculture: A Review.” *2020 Third International Conference on Vocational Education and Electrical Engineering (ICVEE)*, pp. 1–6, 2020.

[8] R. Lallensack, “Five Roles Robots Will Play in the Future of Farming.” May 2021. [Online]. Available: https://www.smithsonianmag.com/innovation/five-roles-robots-will-play-future-farming-180973242/

[9] T. H. T. Tran, D. S. Nguyen, N. T. Vo, and H. N. Le, “Design of Delta Robot Arm based on Topology optimization and Generative Design Method,” 2020, pp. 157–161.

[10] C. Joochim, A. Kunapinum, S. Kaewkorn, and P. Keeratiwintakorn, “Development of Pick and Place Delta Robot.” pp. 475–486, 2020.

[11] “What is a stereo vision camera?” May 2021. [Online]. Available: https://www.e-consystems.com/blog/camera/what-is-a-stereo-vision-camera/

[12] “Feeding the craving for quality chillies in Malaysia,” September 2021. [Online]. Available: https://foundingbird.com/my/blog/feeding-the-craving-for-quality-chillies-in-malaysia

[13] P. Estefó, R. Robbes, J. Simmonds, and J. Fabry, “The Robot Operating System: Package Reuse and Community Dynamics,” *Journal of Systems and Software*, October 2018.

[14] R. Tellez, “What is Robot Operating System (ROS),” April 2021. [Online]. Available: https://www.theconstructsim.com/what-is-ros/

[15] H. Kutluca, “Robot Operating Systems 2 (ROS 2) Architecture,” 2021. [Online]. Available: https://medium.com/software-architecture-foundations/robot-operating-system-2-ros-2-architecture-731ef1867776

[16] “What is ROS,” September 2021. [Online]. Available: https://ubuntu.com/robotics/what-is-ros

[17] D. Foundation, “What is DDS?” September 2021. [Online]. Available: https://www.dds-foundation.org/what-is-dds-3/

[18] “ROS 2 Documentation: Foxy,” September 2021. [Online]. Available: https://docs.ros.org/en/foxy/Releases/Release-Foxy-Fitzroy.html

[19] “Understanding ROS 2 topics,” September 2021. [Online]. Available: https://docs.ros.org/en/foxy/Tutorials/Topics/Understanding-ROS2-Topics.html

[20] “ROS Gazebo: Everything You Need To Know,” April 2021. [Online]. Available: https://roboticsimulationservices.com/ros-gazebo-everything-you-need-to-know/

[21] A. Sears-Collins, “What is the difference between rviz and Gazebo?” September 2021. [Online]. Available: https://automaticaddison.com/what-is-the-difference-between-rviz-and-gazebo/

[22] “How URDF Models and 3D Models Can Help Your Next Robotics Project,” September 2021. [Online]. Available: https://stanleyinnovation.com/urdf-models-3d-models-robotics-projects/

[23] H. Deng, J. Xiong, and Z. Xia, “Mobile manipulation task simulation using ROS with MoveIt,” July 2017, pp. 612–616.

[24] "What is SDFormat," September 2021. [Online]. Available: http://sdformat.org/

[25] "Xacro, URDF, SDF, Gazebo," September 2021. [Online]. Available: https://nu-msr.github.io/me495_site/lecture10_sdf_gazebo.html

[26] École polytechnique fédérale de Lausanne (EPLF) School of Engineering, "Reymond Clavel, creator of the delta robot, reflects on his career," April 2021. [Online]. Available: https://sti.epfl.ch/reymond-clavel-creator-of-the-delta-robot-reflects-on-his-career/

[27] E. Coronado, M. Maya, A. Cardenas, O. Guarneros, and D. Piovesan, "Vision-based Control of a Delta Parallel Robot via Linear Camera-Space Manipulation." *Journal of Intelligent & Robotic Systems*, pp. 93–106, 2017.

[28] R. L. Williams II, "The Delta Parallel Robot: Kinematics Solutions," January 2016. [Online]. Available: https://www.ohio.edu/mechanical-faculty/williams/html/PDF/DeltaKin.pdf

[29] R. Clavel, M. Bouri, S. Grousset, and M. Thumeysen, "A NEW 4 D.O.F PARALLEL ROBOT THE MANTA," November 1999.

[30] M. Opl, M. Houlb, J. Pavlik, F. Bradáč, P. Blecha, J. Kozubik, and J. Coufal, "DELTA - Robot with Parallel Kinematics," 2011.

[31] Y. Li, D. Shang, and Y. Liu, "Kinematic modeling and error analysis of Delta robot considering parallelism error," *International Journal of Advances of Robotic Systems*, 2019.

[32] K. Waldron and J. Schmiedeler, "Handbook of Robotics Chapter 1: Kinematics," 2007.

[33] M.Mahmoodi, M.G.Tabrizi, and K.Alipour, "A new approach for Kinematics-based design of 3-RRR delta robots with a specified workspace," *2015 AI & Robotics (IRANOPEN)*, 2015.

[34] S. Kucuk and Z. Bingul, "Robot Kinematics: Forward and Inverse Kinematics," *Industrial Robotics: Theory, Modelling and Control*, 2006.

[35] "Intel RealSense Tracking Camera T265," May 2021. [Online]. Available: https://www.intelrealsense.com/tracking-camera-t265/

[36] "Stereo Vision for 3D Machine Vision Applications," September 2021. [Online]. Available: https://www.clearview-imaging.com/en/blog/stereo-vision-for-3d-machine-vision-applications

[37] A. Islam, M. Asikuzzaman, M. O. Khyam, M. Noor-A-Rahim, and M. R. Pickering, “Stereo Vision-Based 3D Positioning and Tracking.” *IEEE Access*, vol. 8, pp. 138 771–138 787, 2020.

[38] S. Simon, “Choosing Lenses for Stereo Camera Applications,” April 2021. [Online]. Available: https://www.novuslight.com/choosing-lenses-for-stereo-camera-applications_N7885.html

[39] Z. Zhengyou, “Camera Parameters (Intrinsic, Extrinsic),” *Computer Vision: A Reference Guide*, pp. 81–85, 2014.

[40] F. Longsheng, G. Fangfang, W. Jingzhu, L. Rui, K. Manoj, and Z. Qin, “Application of consumer RGB-D cameras for fruit detection and localization in field: A critical review,” *Computers and Electronics in Agriculture*, vol. 177, p. 105687, 2020.

[41] J. Shi, “An introduction towards 3D Computer Vision.” May 2021. [Online]. Available: https://medium.com/@jianshi_94445/an-introduction-towards-3d-computer-vision-71be8ce11956

[42] D. Rivas-Lalaleo, D. Tumbaco-Mendoza, E. Galarza-Zambrano, and W. Quimbita-Zapata, “Delta robot controlled by robotic operating system,” *ITECKNE Innovación e Investigación en Ingeniería*, 2014.

[43] R. Shamsiri, R, A. Hameed, I, M. Karkee, and C. Weltzien, “Robotic Harvesting of Fruiting Vegetables: A Simulation Approach in V-REP, ROS and MATLAB,” 2018.

[44] F. Okoli, Y. Lang, O. Kermorgant, and S. Caro, *Cable-Driven Parallel Robot Simulation Using Gazebo and ROS*, May 2019.

[45] M. S. Amiri, R. B. Ramli, M. A. A. Tarmizi, M. F. Ibrahim, and K. D. Narooei, “Simulation and Control of a Six Degree of Freedom Lower Limb Exoskeleton,” 2020.

[46] R. Barth, J. Baur, T. Buschmann, Y. Edan, T. Hellström, T.-T. Nguyen, O. Ringdahl, W. Saeys, C. Salinas, and E. Vitzrabin, “Using ROS for agricultural robotics - design considerations and experiences,” May 2014.

[47] F. Sze, “Simulation and Framework for the Humanoid robot TigerBot,” 2018.

[48] “SOLIDWORKS LOGO,” October 2021. [Online]. Available: https://1000logos.net/solidworks-logo/

[49] D. Wu, Z. Zhang, and Z. Wang, “Application research of solidworks in modeling of straw carbonization preparation plant,” *Journal of Physics: Conference Series*, vol. 1303, p. 012048, August 2019.

[50] “The new version of virtualbox 6.0.8 has already been released, solving some problems,” October 2021. [Online]. Available: https://www.linuxadictos.com/en/The-new-version-virtualbox-6-0-8-has-already-been-released%2C-solving-some-problems.html

[51] “Virtualbox,” October 2021. [Online]. Available: https://www.virtualbox.org/

[52] A. Sears-Collins, “How to Simulate a robot using Gazebo and ROS 2,” 2021. [Online]. Available: https://automaticaddison.com/how-to-simulate-a-robot-using-gazebo-and-ros-2/

[53] “Intel Realsense D400 Series Product Family Datasheet,” November 2021. [Online]. Available: https://www.intel.com/content/dam/support/us/en/documents/emerging-technologies/intel-realsense-technology/Intel-RealSense-D400-Series-Datasheet.pdf

[54] “Rotary Delta Robot Forward/Inverse Kinematics Calculations,” 2021. [Online]. Available: https://www.marginallyclever.com/other/samples/fk-ik-test.html

[55] Y. Kadam, R. Ajabe, T. More, M. Jangam, and R. Patil, “Development of Delta Robot for Pick and Place on Moving Conveyor,” *IJSRD - International Journal for Scientific Research & Development*, 2019.