TASK-LEVEL MOTION PLANNING OF A MULTI-MANIPULATOR SYSTEM

B.Tech. Project

By

Tarannum Perween (183103)

DEPARTMENT OF MECHANICAL ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY

HAMIRPUR (HP)-177005 (INDIA)

May, 2022

TASK-LEVEL MOTION PLANNING OF A MULTI- MANIPULATOR SYSTEM

A PROJECT

Submitted in partial fulfilment of the requirements for the award of the degreeof

BACHELOR OF TECHNOLOGY

By

Tarannum Perween

Under the guidance of

Dr. Rajesh Kumar Sharma

DEPARTMENT OF MECHANICAL ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY

HAMIRPUR – 177 005 (INDIA)

May, 2022

NATIONAL INSTITUTE OF TECHNOLOGY HAMIRPUR (HP)

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the project titled **"TASK LEVEL MOTION PLANNING OF A MULTI-MANIPULATOR SYSTEM"** in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology and submitted in the Department of Mechanical Engineering, National Institute of Technology Hamirpur, is an authentic record of my own work carried out during a period from July,2021 to May, 2022 under the supervision of **Dr. Rajesh Kumar Sharma**, Associate Professor, Department of Mechanical Engineering, National Institute of Technology Hamirpur.

The matter presented in this project report has not been submitted by me for the award of any other degree of this or any other Institute/University.

Sd/-
Tarannum Perween

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:25 MAY 2022

Sd/-
(Dr. Rajesh Kumar Sharma)
Associate Professor

The project Viva-Voce Examination of Tarannum Perween has been held on 25 MAY 2022.

Signature of Supervisor(s)

Signature of HOD

Acknowledgment

We appreciate the almighty God for his blessings in helping us write the report and putting our skills to good use. Our parents deserve our sincerest gratitude for working so hard to get us to where we are now.

Dr. Sant Ram Chauhan, Head of the Department of Mechanical Engineering, National Institute of Technology, Hamirpur, deserves our thanks for allowing us to participate in this undergraduate project. We are grateful to **Dr. Rajesh Sharma** for his assistance. We would not have completed this project without his consistent encouragement and help at every step. We appreciate his patience and willingness to correct our errors.

Abstract

Nowadays, Robotic arms are essential in manufacturing industries to optimize their production and other works. So, In one of the most critical projects, we decided to work on a problem that is a significant concern in the manipulator industry.

The main problem which we evidence in the industry after a lot of research is there is no multi-tasking customized working multi-manipulator system is available in the market to do multiple works at the same time. So, our main goal is to make a prototype that can do a set of multiple works at the same time.

We used MTC to complete multi-arm tasks like building structure and complex pouring tasks simulation Then we brought our work to the hardware stage. Here we have designed a robotic arm CAD model in Solidworks. Then we converted it into STL files and did the slicing using CURA software. We have printed the arm using the "Tevo Tarantula" printer which is based on Fusion Deposition Modelling (FDM) technique. Then we assembled the arm with 3 MG996R servos and 2 SG90S servos. URDF file is extracted from SolidWorks assembly. Then a package is created using Moveit Setup Assistant. By using the ROSSerial communication, angles are sent to Arduino to run the motor according to the task.

In this paper, we conclude our number of experiments on software like completing different and complex tasks with a multi-manipulator system and then tried the whole concept on a Real Hardware for now using Moveit, Robot Operating System frameworks, and rosserial communication as well.

CONTENTS

List of Figures

Chapter 1

Introduction

This project aims to do motion planning for complex manipulation tasks like pick and place, building structures, and pouring in a coordinated multi-manipulator Systems by using Moveit and ROS framework for Robotics. We have taken this topic as it is an important research area in robotics and we also want to explore it.

We did multiple simulations, experiments, and tests to understand what is possible and what's not. We completed the single arm pick and place, multi-coordinated arm pick and place, and Building structure with coordinated multi-arm in the Moveit framework.

Then we tried to implement these motion planning concepts on Real Hardware. In the beginning stage of the project, we tried to implement the RRT(Rapidly exploring random tree) motion planning algorithm with the help of ROS Visualization(Rviz), Rosserial Communication, and Robot Operating System concepts on the Real 3D printed single robotic arm by which we are able to move our 3D printed arm according to the task given with the help of MG996R and SG90 servo motors, Arduino, Motor driver and Power Supply. It involves solving kinematics and inverse kinematics for multi-manipulator systems.

Chapter 2

Literature Review

2.1 Introduction

We have used robotic manipulators and motion planning frameworks like Moveit to perform simple tasks. Thus, this work becomes the basis of more complex manipulation tasks performed in this project. We had mainly worked on the forward and inverse kinematics of the manipulator.

2.2 Moveit and Motion Planning

Motion planning decomposes the required movement task into different motions that satisfy movement limitations while potentially optimizing specific movement components. Consider a mobile robot traversing a structure to reach a distant destination.

Consider a mobile robot navigating to a remote location within a building. This task must be completed without clashing with walls or tumbling staircases. A motion planning algorithm defining these activities as input would generate the speed and turning directives delivered to the robot's wheels. Motion planning algorithms can be used to address robots with a more significant number of joints (e.g., industrial manipulators), more complex tasks (e.g., item manipulation), various limits (e.g., a car that can only travel forward), and uncertainty (e.g., imperfect models of the environment or robot).

Consider flying one or more uncrewed planes or drones through a crowded interior or outdoor environment. The same motion planning methods can be applied in a 3D setting with environmental disturbances such as wind and severe weather.

Robotics, such as autonomy, automation, and robot design in CAD software and other sectors such as animating digital characters, video games, artificial intelligence,

architectural design, robotic surgery, and molecular biological research, use motion planning.

2.3 Robot Operating System (ROS)

One of the most significant advancements in robotics has been the Robot Operating System (ROS). Robotic innovation has traditionally had a high entry hurdle. You had to design a comprehensive system if you wanted to develop a robot: the physical device and the control systems, interface, and inspection tools needed to get the robot up and running as a test platform.

ROS, the ultimate robotics middleware, has arrived. You now have a comprehensive ecosystem with powerful tools at your disposal to get your robot up and running.

2.4 RRT Algorithm (Rapidly Exploring Random Tree)

A rapidly exploring random tree (RRT) is a space-filling tree constructed at random to quickly investigate nonconvex, high-dimensional environments. The tree is built incrementally from samples drawn at random from the search space, and it is predisposed to spread to enormous areas of the problem that have yet to be studied. Because they effectively manage obstacles and differential limitations concerns, they are extensively used in autonomous robotic motion planning.

RRT's principle is essential. Points are produced at random and linked to the nearest available node. Every time a vertex is formed, it must be checked to see if it is outside an obstruction. In addition, impediments must be avoided when connecting the vertex to its neighbor. The process stops when a node is formed within the goal region, or a limit is reached.

It is a matter of design how a random place is determined. There are simple techniques available, such as using built-in random number generators. Such approaches are sufficient for basic applications. Developers should be aware. However, random number generators are not entirely unexpected and contain some bias. There is a whole field of study called Sampling Theory for the interested.

The manner of chaining the randomly produced vertex can also be changed.

Calculating the vector forming the shortest distance for the first three axes, the waist, the shoulder, and the elbow. A new node is added to the border at the intersection and connected to the randomly generated vertex. Alternatively, the vertex can be joined by chaining a connection of discretized nodes to the closest node. This solution uses more storage points but takes less calculation and is easier to implement.

RRT generates graphs that are pretty cubic. As nodes are connected to their nearest neighbors, this is expected. The structural structure of these graphs makes finding an optimal path difficult. A triangle's two legs are traveled across instead of taking the hypotenuse between two places. This distance is longer.

The algorithm's speed and simplicity make it appealing. RRT is relatively fast when compared to other path planning methods. The most expensive aspect of the procedure is locating its closest neighbor, which takes a long time depending on the number of vertices generated.

RRT Pseudo Code

```
Qgoal //region that identifies success
Counter = 0 //keeps track of iterations
lim = n //number of iterations algorithm should run for
G(V,E) //Graph containing edges and vertices, initialized as empty
While counter < lim:
    Xnew  = RandomPosition()
    if IsInObstacle(Xnew) == True:
        continue
    Xnearest = Nearest(G(V,E),Xnew) //find nearest vertex
    Link = Chain(Xnew,Xnearest)
    G.append(Link)
    if Xnew in Qgoal:
        Return G
Return G
```

2.5 URDF

The URDF (Universal Robot Description Format) model is a collection of files that inform ROS of a robot's physical attributes. The ROS (Robot Operating System) application uses these files to tell the computer how the robot appears in real life. URDF data are essential for ROS to understand and simulate situations before a researcher or engineer receives the robot.

The mass, center of gravity, and dimensions of the CAD file are described in URDF files, which are basic XML files. They make it simple to simulate in ROS before making a purchase. URDF files also allow you to modify the platform in ROS, allowing you to test different sensors, robotic arms, vision components, and payload capacities. In current robotics, ROS integration is required, and URDF models are required to combine a platform with ROS.

It contains all the information of a model which is required to configure or specify any model into any robotics software. For e.g. Mass, Inertia, orientation, color, etc.

Chapter 3

Design,Setup and Methodology

3.1 Articulated Robotic Arm

A rotary-jointed robot is known as an articulated robot. A comprehensive range of motion is possible with rotary joints. An articulated robot may feature one or more rotary joints, as well as other types of joints, depending on the robot's function and design.

A robot may perform extremely precise motions using rotary joints. Articulated robots are typically found in industries, where it can be bend in many direction due to its flexible nature. We can use numerous arms for increasing control or to perform multiple jobs simultaneously, as rotary joints allow robots to turn back and forth .

Articulated robots also has arms and legs which allow them to move and operate a wide range of objects. Some are intended as console units with arms, in which the unit is set in place and the arms are utilized to carry out tasks.. An articulated robot, for example, could be used to transfer and carry samples throughout a medical lab.
.

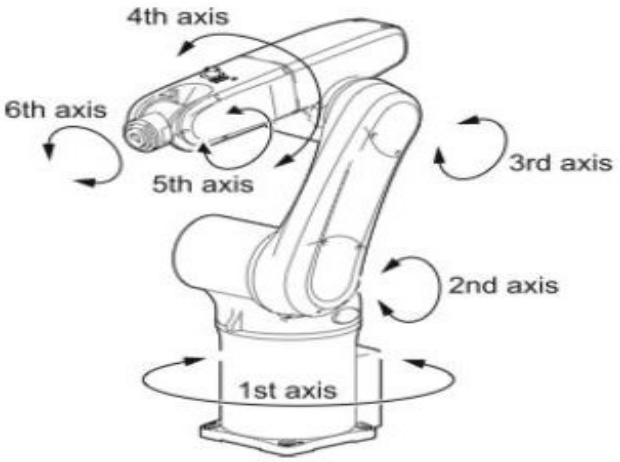


Fig 3.1 - An articulated robotic arm [9]

3.2 Our Design

Our robotic arm is also a type of vertical articulated robot. It consists of Six links and End effector. It has 5 revolute joints.
The arm has five degrees of freedom. It has the gripper attached with it.

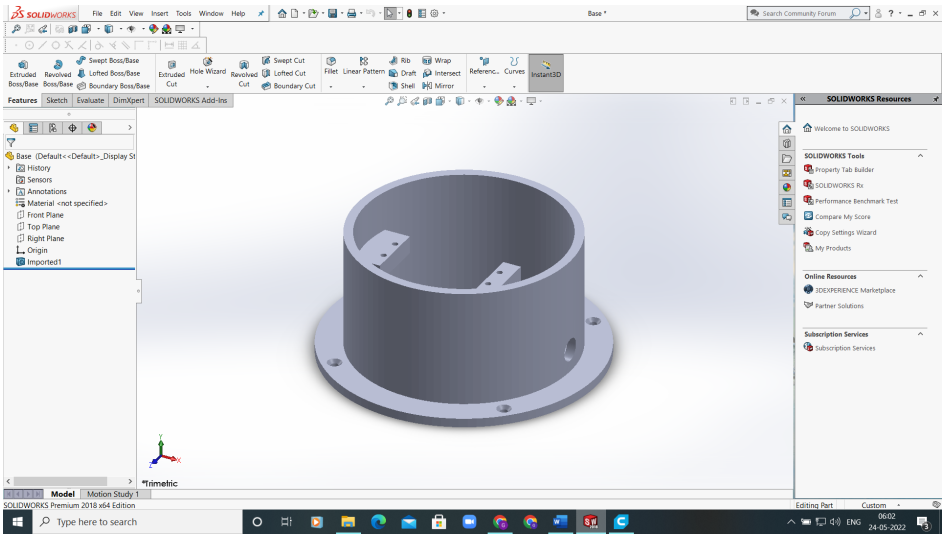


Fig 3.2- Base of the arm

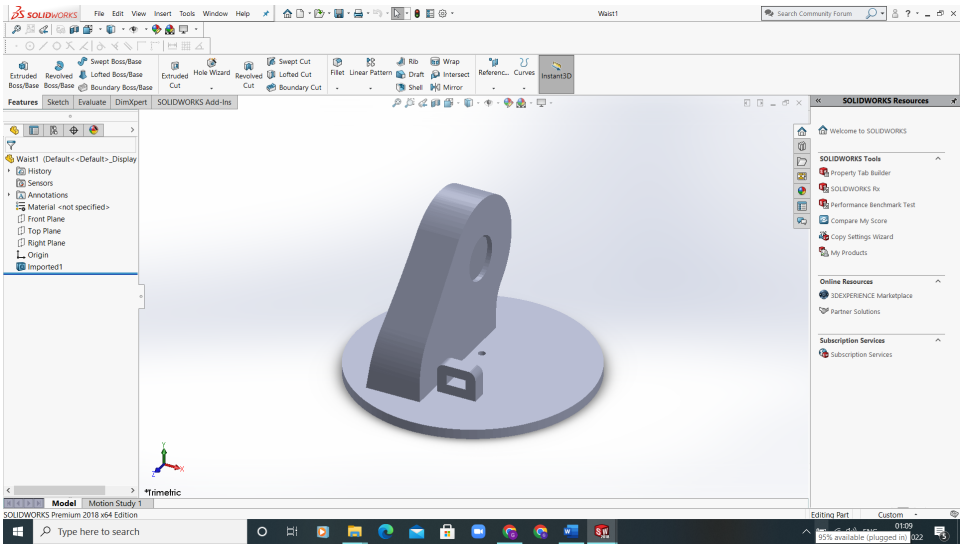


Fig 3.3 - Waist of the arm

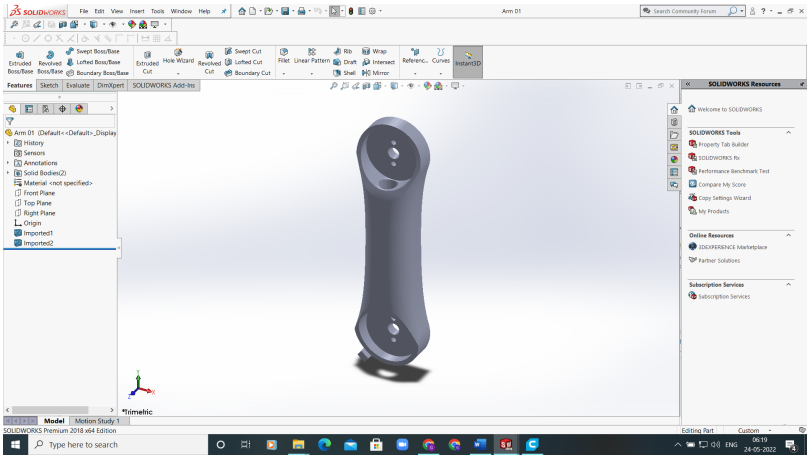


Fig 3.4 - Link 1

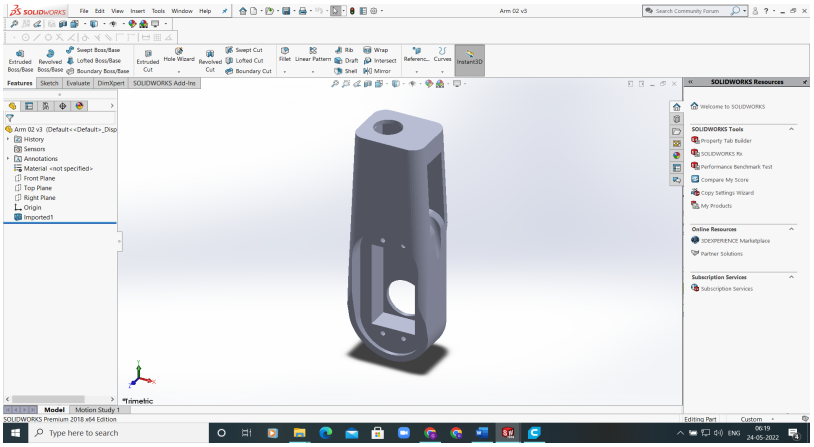


Fig 3.5 - Link 2

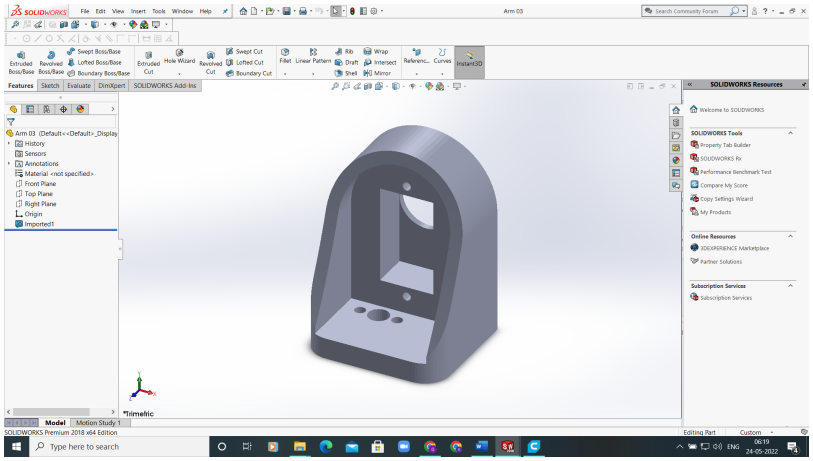


Fig 3.6 - Link 3

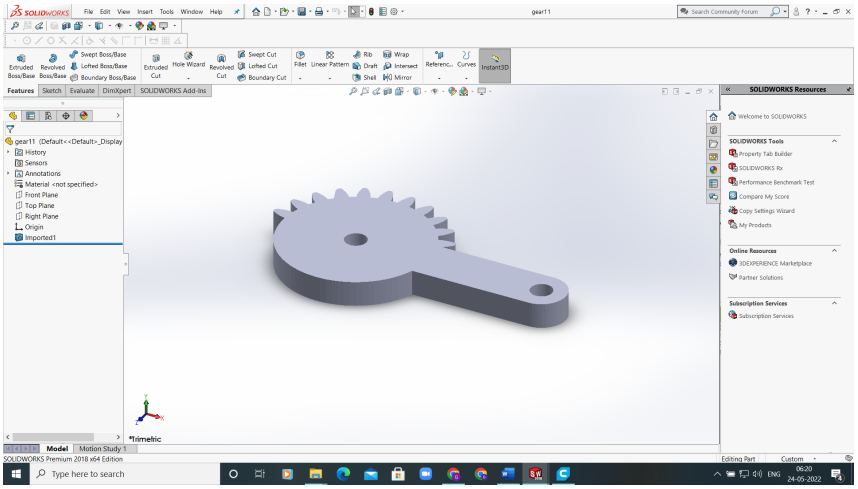


Fig 3.7 - Gear 1

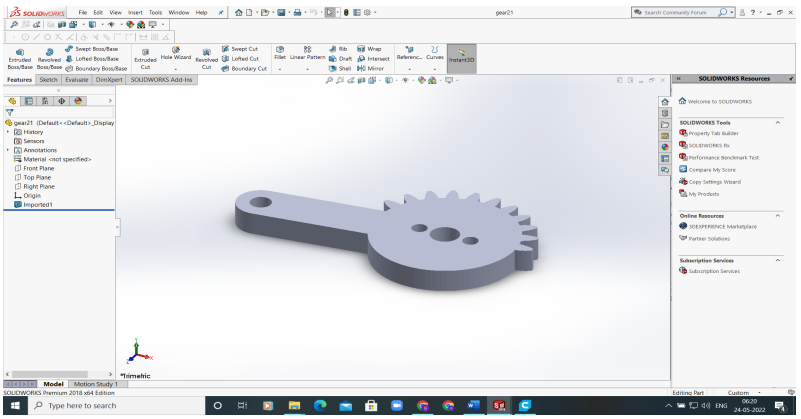


Fig 3.8 - Gear 2

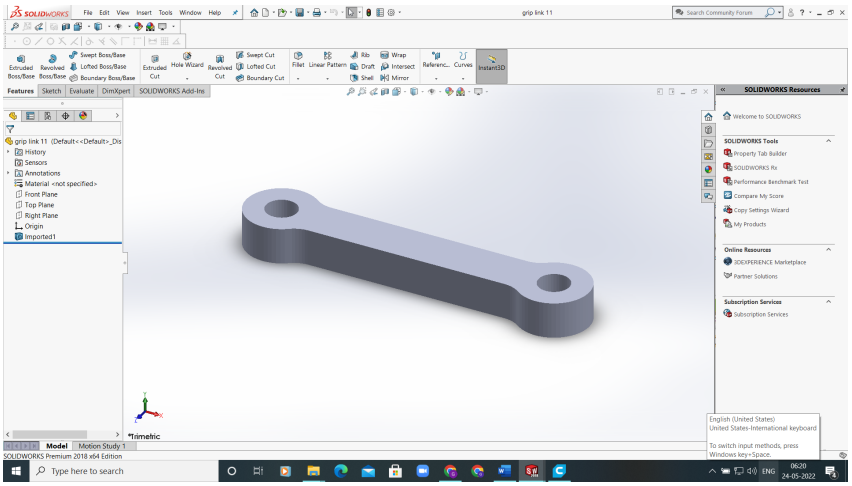


Fig 3.9 - Gripper Link

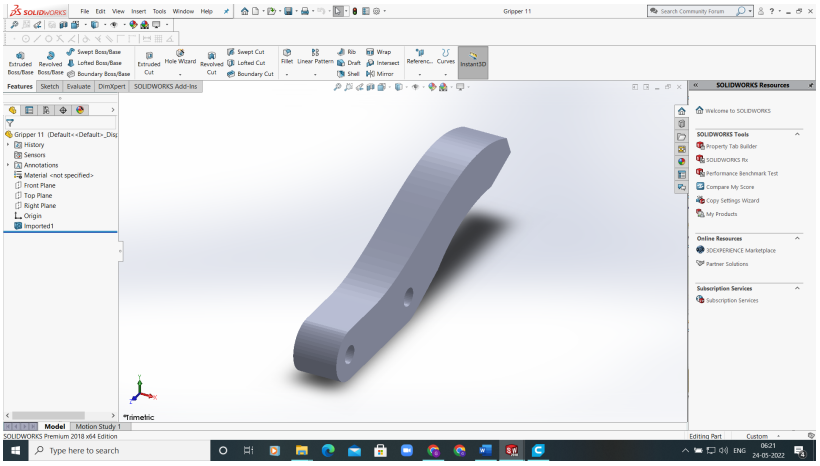


Fig 3.10 - Gripper hand

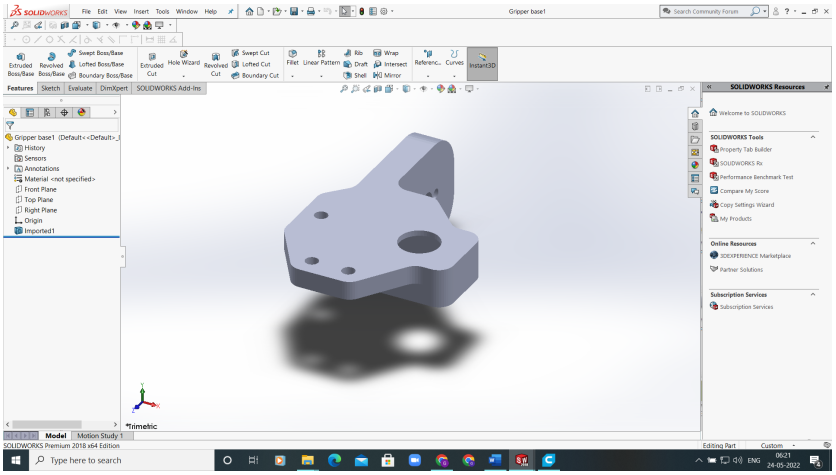


Fig 3.11 - Gripper Base

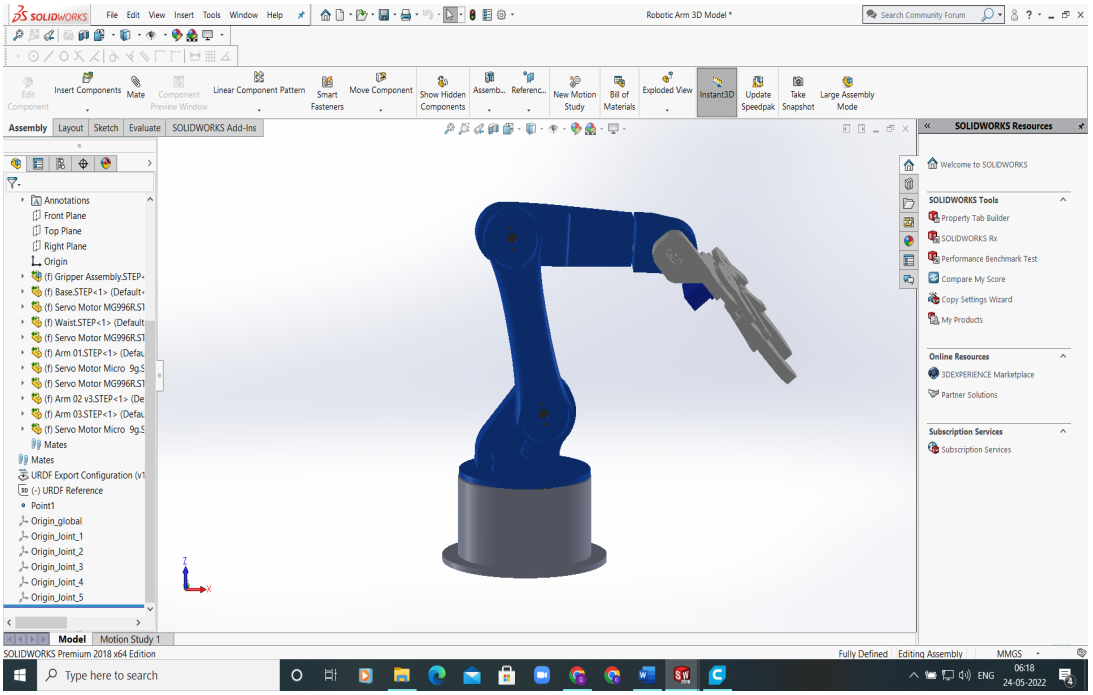


Fig 3.12- Final assembly of the robotic arm

Above shown are the robotic arm's part and final assembly designed in Solidworks. After making this we have converted the SLDPRT file to an STL file which is needed to 3D print the CAD model.

3.2.1 Cura Software

Cura is a 3D printer slicing tool that is free and open source. Cura allows us to convert an STL file that represents a part's shape into g-codes that can be read by a 3D printer. The g-code is downloaded into an SD card, which is then linked to the 3D printer to transfer the g-code. Cura also allows us to customize several settings, such as the layer height, heating temperature, infill, material used, and so on.

- STL -
  STL (short for "stereolithography") is a native file format for stereolithography CAD software. It is a universal format which is used by maximum available 3D printers in market. It helps to do the fast prototyping and 3D printing. STL files only define the object's surface geometry. It does not represent the texture, colour or other CAD features."

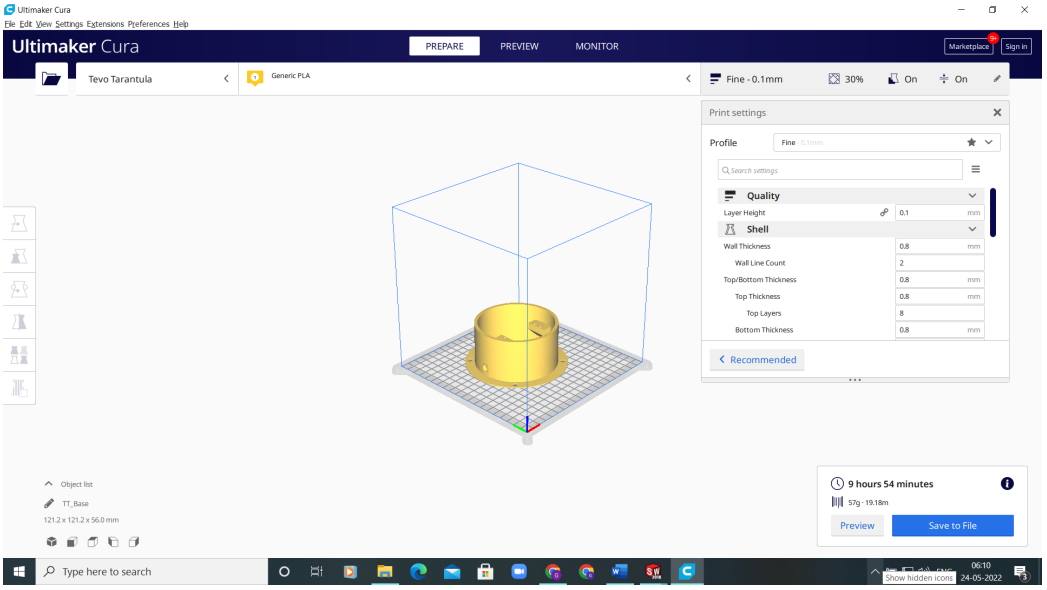


Fig 3.13 - Base of the robotic arm in Cura software for slicing

To begin, tell Cura, the printer you want to use, in this case, the Tevo Tarantula.
Cura then lists the primary parameters that can be changed on the right:
-Layer Height: This controls the layer height.
-Infill: This specifies the infill proportion.
-Generate Support: This will create a flat region surrounding or beneath your object that can be easily trimmed away later. It is required in order to retain integrity of structure and enable printing.
-Build Plate Adhesion: This creates a structure to support overhanging areas of the object. Such pieces would collapse during printing without these structures.
Other factors, such as flow through the nozzle, movement speed and wall thickness, are primarily concerned with the machine's flow.
All of these variables have an impact on the printing time. The printing time decreases as the layer height is increased. Reduced layer height, on the other hand, will increase printing time.
The same holds true for infill; more infill means more time spent printing.
Cura estimates the printing time, the length of the printing filament and the weight of the material used, and using all of these factors. Cura can print multiple parts at once as long as they all fit on the printing bed.
The advantage of printing the larger pieces first and the smaller ones later was significant. After a large area has been printed for around 4-5 hours, it can be assessed to see if the quality is satisfactory. Otherwise, it will have to be reprinted till the quality is acceptable [10].

3.2.2 Fusion Deposition Modeling (FDM)

The printer available with us is “Tevo Tarantula,” which uses the Fusion Deposition Modeling (FDM) technique for 3D printing.
Fused deposition modeling (FDM) is a widely used additive manufacturing technique for modeling and prototyping.

Fused Deposition Modelling (FDM) creates 3D objects directly from 3D CAD data. The filament (like PLA) made of thermoplastic material are extruded out of nozzle by a head which is temperature-controlled.
In this process first we import the model's STL file into a slicing software like Cura. Then the model is aligned and sliced in layers of thicknesses ranging from 0.127 mm to 0.254 mm. Then the support is generated if needed as it depend on the object geometry. Then this data is transferred to SD card and is put in the printer. The system then prints the model one layer at a time in the X, Y, and Z axes.
After printing, the supports are removed and the surface is finished.

Process:

After processing of STL files, slicing and support structures generation two materials are dispensed by the machine: one for the support and one for the model.

The tool path generated is followed by the extrusion head which then liquefies and deposits the thermoplastics. The materials are deposited in specified layer thickness and the part is produced one layer at a time in upward direction. The plastic filament or metal wire is unwound from a coil which delivers material to an extrusion nozzle. The nozzle is heated to melt the material. Nozzles can move in both horizontal and vertical direction. After extrusion from the nozzle the material solidifies immediately which produces the model. Stepper motors or servo motors are commonly used for operating the extrusion head.

Many types of materials are available which can be used as filament like polyphenyl sulfone,, as well as acrylonitrile butadiene styrene (ABS) polymer.. [11].

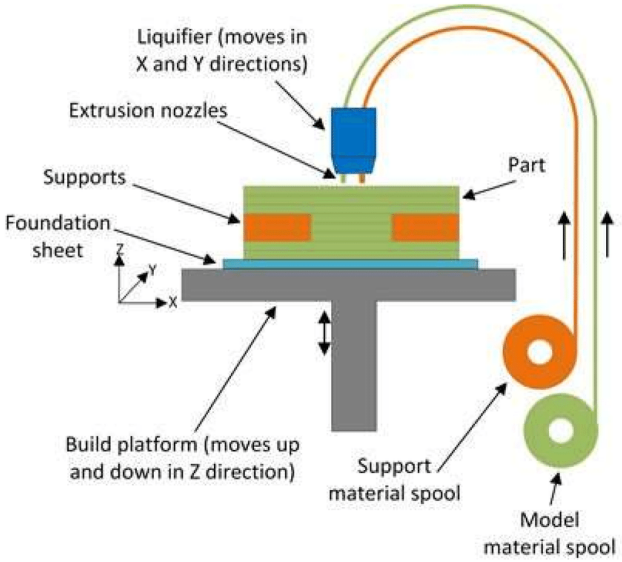


Fig 3.14- Fusion Deposition Modeling technique

This is our final printed arm made of PLA (Polylactic acid) filament. Because it can be produced at a low temperature and does not require a heated bed, it is the default filament for most extrusion-based 3D printers. Its weight is around 300 gms. All the 3D printed parts were assembled using a nut and bolt. we used the MG996R servos for the first three axes, the waist, shoulder, and elbow, and the smaller SG90 micro servos for the remaining two axes, wrist roll, and wrist pitch, as well as the gripper.

3.2.3 Converting to URDF file

The URDF (Universal Robot Description Format) model is a set of files that tell ROS about a robot's physical characteristics. These files are used by the ROS (Robot Operating System) program to tell the computer how the robot appears in real life. Before a researcher or engineer actually gets the robot, URDF data are required for ROS to understand and simulate situations with the robot.
The mass, center of gravity, and dimensions of the CAD file are described in URDF files, which are basic XML files. They make it simple to simulate in ROS before making a purchase. URDF files also allow you to modify the platform in ROS, allowing you to test different sensors, robotic arms, vision components, and payload capacities. In current robotics, ROS integration is required, and URDF models are required to combine a platform with ROS.

So here we have converted our robotic arm assembly to a URDF file.

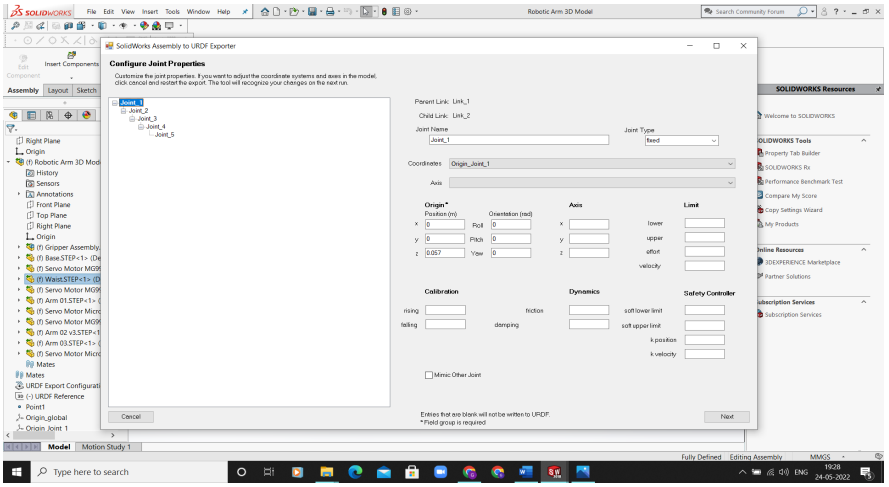


Fig 3.15 - Joints parameters in URDF file

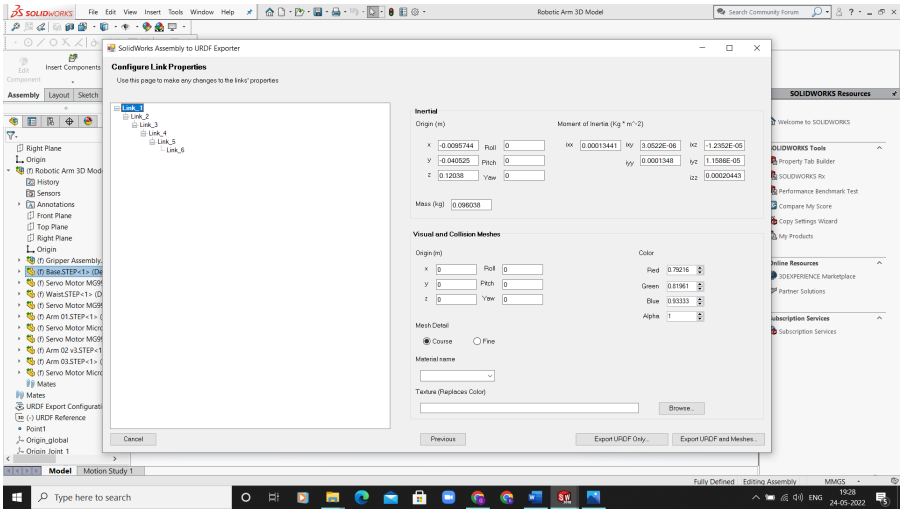


Fig 3.16 - Link parameters in URDF file

3.3 Robotic Arm Electronics Setup

3.3.1 Actuators

The actuators in robots are responsible for giving power to the robot joints. Any of the following sources can be used to power it: Hydraulic, Electrical, and Pneumatic.

3.3.1.1 Servo-Motor

A servo motor is a rotary actuator that enables accurate angular position control. It is made of a motor connected to a position sensor via a reduction gearbox. A servomotor is made up of four components: a standard DC motor, a gear reduction unit, a position-sensing device (often a potentiometer—a volume control knob), and a control circuit.

The servomotor's job is to take a control signal that specifies the desired servo shaft output position and apply power to its DC motor until the shaft turns to that position. It determines the rotational position of the shaft using the position-sensing device, so it knows which way the motor must spin to move the shaft to the commanded position.
Specification of servo motors used in our robotic arm:

(1) MG996R Servo -

Weight: 55g
Dimension: 40.7×19.7×42.9mm
Stall torque: 9.4kg/cm (4.8v); 11kg/cm (6.0v)
Operating speed: 0.19sec/60degree (4.8v); 0.15sec/60degree (6.0v)
Operating voltage: 4.8 ~ 6.6V
Gear Type: Metal gear

(2) SG90 Servo -

Weight: 9 gm
Operating voltage: 3.0V~ 7.2V
Servo Plug: JR
Stall torque @4.8V : 1.2kg-cm
Stall torque @6.6V: 1.6kg-cm

3.3.2 MicroController - Arduino UNO

The Arduino UNO is an ATmega328P-based microcontroller board. It features 14 digital input/output pins (six of which can be used as PWM outputs), six analog inputs, a 16 MHz ceramic resonator, a USB port, a power jack, an ICSP header, and a reset button. It comes with everything you need to get started with the microcontroller; simply connect it to a computer by USB or power it with an AC-to-DC adapter or battery.

Board	**Name**	Arduino UNO R3
	SKU	A000066
Microcontroller	ATmega328P	
USB connector	USB-B	
Pins	**Built-in LED Pin**	13
	Digital I/O Pins	14

	Analog input pins	6
	PWM pins	6
Communication	**UART**	Yes
	I2C	Yes
	SPI	Yes
Power	**I/O Voltage**	5V
	Input voltage (nominal)	7-12V
	DC Current per I/O Pin	20 mA
	Power Supply Connector	Barrel Plug
Clock speed	**Main Processor**	ATmega328P 16 MHz
	USB-Serial Processor	ATmega16U2 16 MHz
Memory	**ATmega328P**	2KB SRAM, 32KB FLASH, 1KB EEPROM
Dimensions	**Weight**	25 g
	Width	53.4 mm
	Length	68.6 mm

3.3.3 Motor Driver - MD13S

Motor drivers serve as a connection between motors and control circuits. The controller circuit works with low current signals. However, the motor requires a lot of currents. The purpose of motor drivers is to convert a low-current control signal

into a higher-current signal capable of driving a motor.

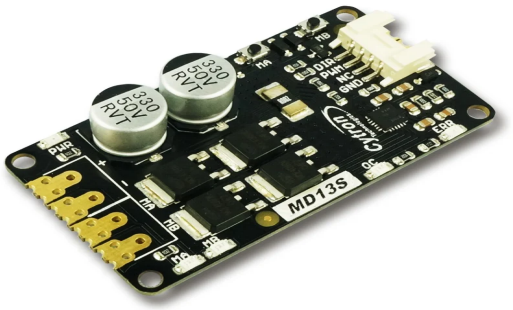


Fig 3.17 - MD13S Motor Driver [13]

(1)Support motor voltage ranges from 6V to 30V
(2) Maximum current up to 13A continuous and 30A peak (10 seconds).
(3) GROVE compatible
(4) 3.3V and 5V logic level input.
(4) SMD compatible
(5) Peak Current (A): 30 (10 seconds)
(6) Continuous Current (A): 1
(7) No. of Channels: 1

3.3.4 Power supply

We have used a 12V power supply.

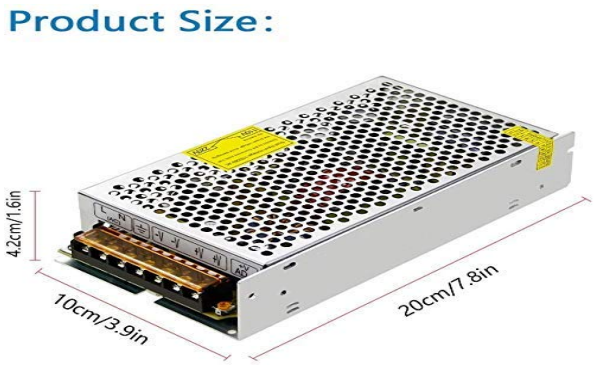


Fig 3.18 - 12 V Power Supply

(1) Input: AC110V/220V
(2) Output: DC 12V 10A
(3) Output Power: 120W
(4) Frequency: 50/60Hz
(5) Screw Terminal Number: 7

(6) Fix Screw Hole Diameter: 2.5mm(0.1 inch)
(7) Material: Metal, Electronic Parts
(8) Size: 19.5cm x 10cm x 4cm(L*W*H)
(9) Main Color: Silver Tone
(10) Net Weight: 400G

3.4 Motion Planning on a Software

After Converting the model into URDF, We created a package with the help of Moveit Setup Assistant which is provided by the Moveit framework.

Launch MoveIt Setup Assistant with the following command.

```
roslaunch moveit_setup_assistant setup_assistant.launch
```

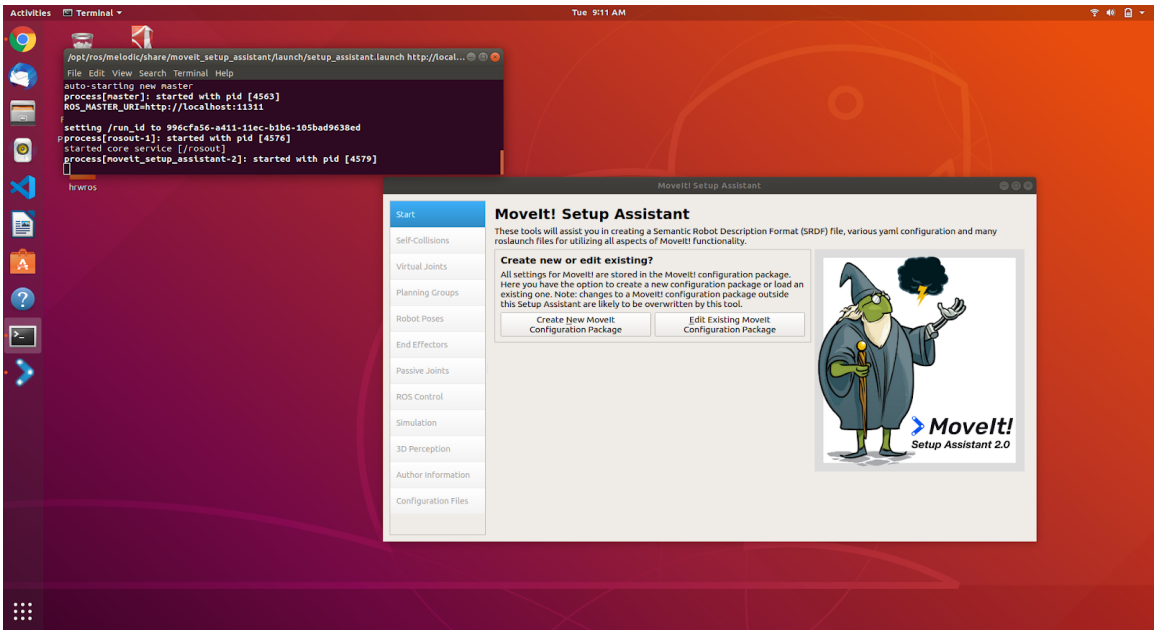


Fig 3.19 - Moveit setup Assistant

There are lots of parameters that are important in generating our customized package:-
That is, Self-collision, Virtual Joints, Planning Groups, Robot Poses, End Effectors, Passive Joints, 3D Perception, Simulation, and ROS COntrol.

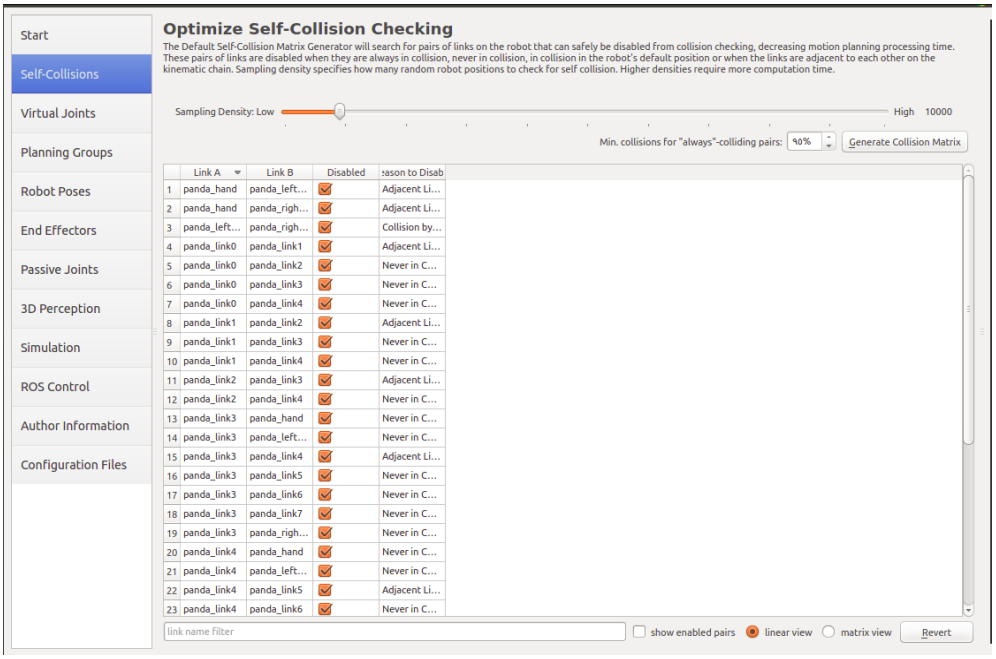


Fig 3.20 - Collision matrix generation

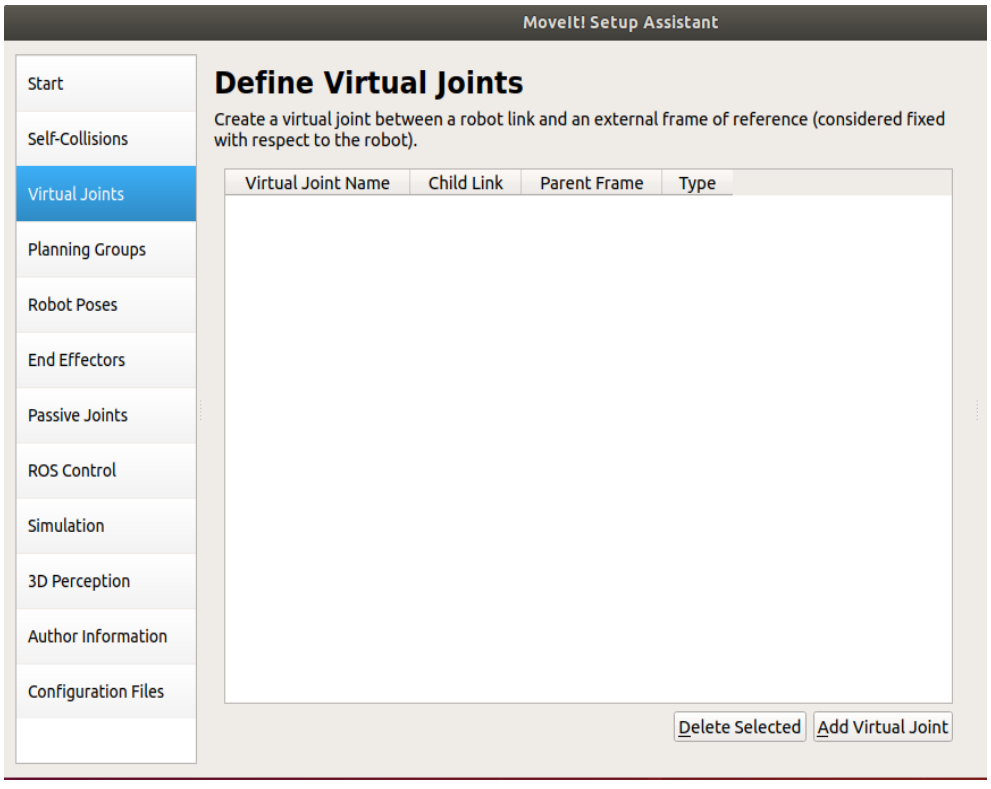


Fig 3.21 - virtual joints

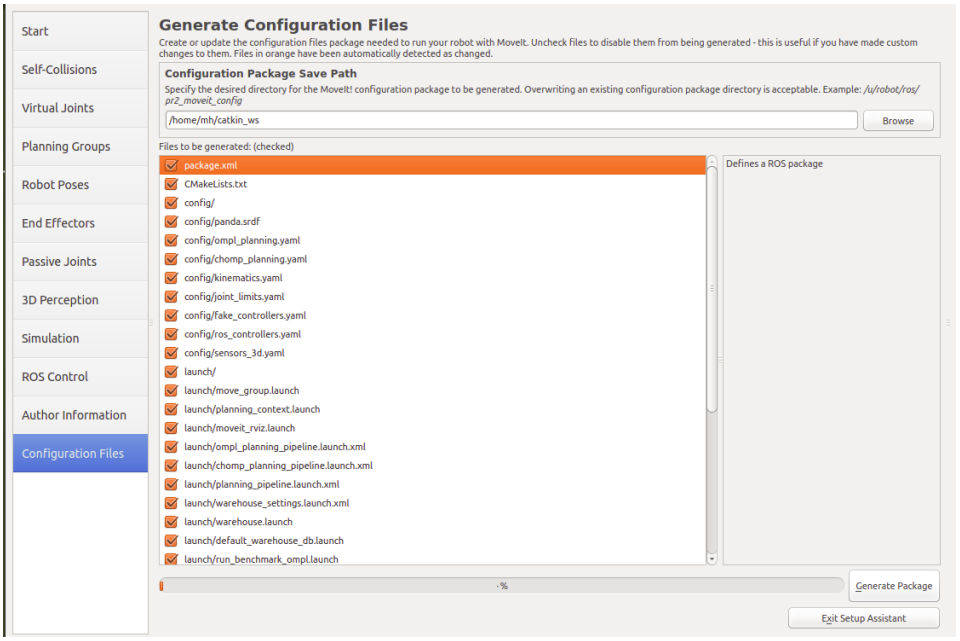


Fig 3.22 - Generating configuration files

3.5 MoveIt RobotCommander Interface

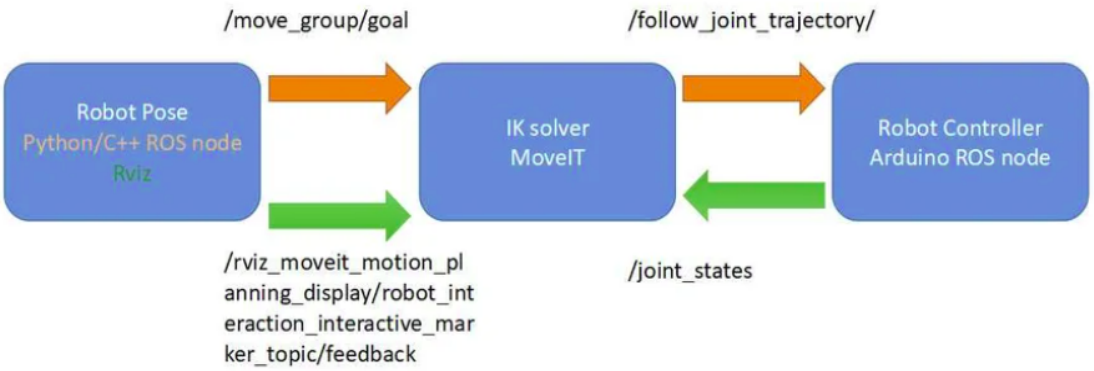


Fig 3.23 - joint state graph

We'll utilize the / joint states topic to provide the robot controller with servo angles because we don't get feedback. The robot controller is the only component that is missing.

The robot controller in our scenario will be an Arduino Uno running a ROS node with serial. This step includes the Arduino sketch code.

ROS node on the Arduino Uno merely subscribes to the /JointState topic published on the MoveIt, which converts the joint angles from the array to degrees and then feeds them to servos using the standard Servo.h library.

We published the movement trajectory on the /FollowJointState topic and then got a response on the /JointState topic. However, because the hobby servos in our arm are

unable to offer input, we will just subscribe to the /JointState topic, which is provided by the FakeRobotController node. We'll assume that the servo angles we gave them are executed flawlessly.

After uploading the sketch to the Arduino Uno, we connected it to the PC running our ROS installation via the serial cable.

ROSSERIAL -
Rosserial is a protocol for multiplexing many subjects and services through a character device, such as a serial port or network socket, by encapsulating standard ROS serialized messages. It's a point-to-point version of ROS serial communications that are meant to work with low-cost microcontrollers like the Arduino. A universal peer-to-peer protocol, Arduino libraries, and PC/Tablet nodes are all included in the ROS serial.

To use Rosserial, we need to install some packages and dependencies so that it can use those dependencies to word in a proper manner.
The commands to install packages and dependencies are-

1. Install the rosserial package binaries, using apt-get:

```
$ sudo apt-get install ros-melodic-rosserial-arduino
ros-melodic-rosserial-embeddedlinux ros-melodic-rosserial-windows
ros-melodic-rosserial-server ros-melodic-rosserial-python
```

2. For installing the rosserial_client library called ros_lib in Arduino, we must download the latest Arduino IDE for Linux 32/64 bit. The following is the link for downloading the Arduino.

Chapter 4

Result and Discussion

We describe a few typical manipulation tasks and showcase the involved planning stages in the following. For the demo, we are using the panda Franka robotics arm. We will also demonstrate various tasks that involve simple pick place, single pouring, creating some structure, and a multi-arm pouring demo.

4.1 Building a structure using multi-arm

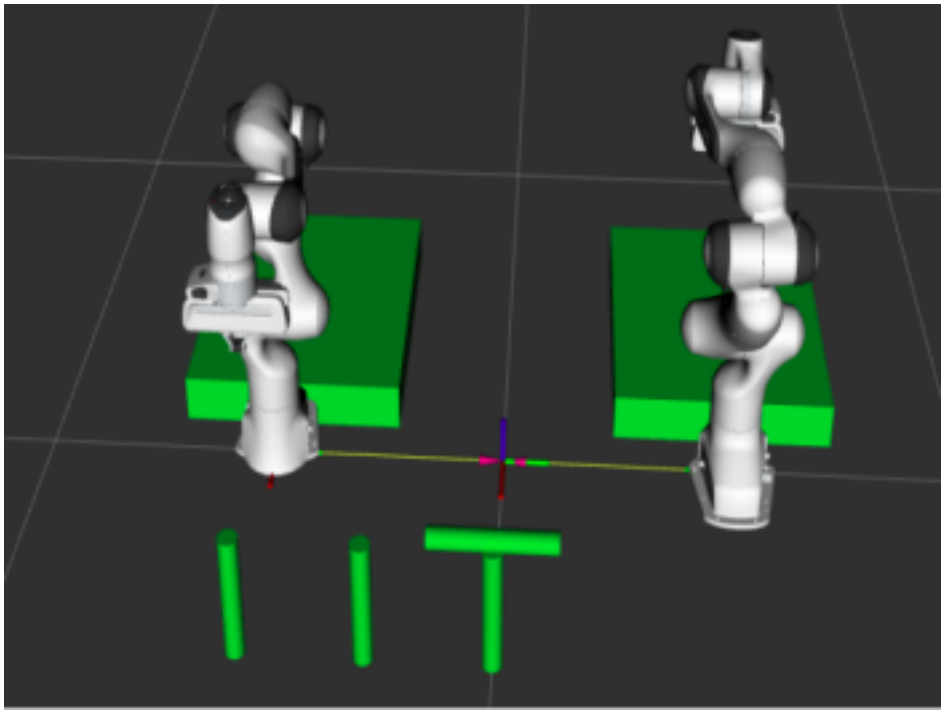


Fig 4.1 Pick Place Scene

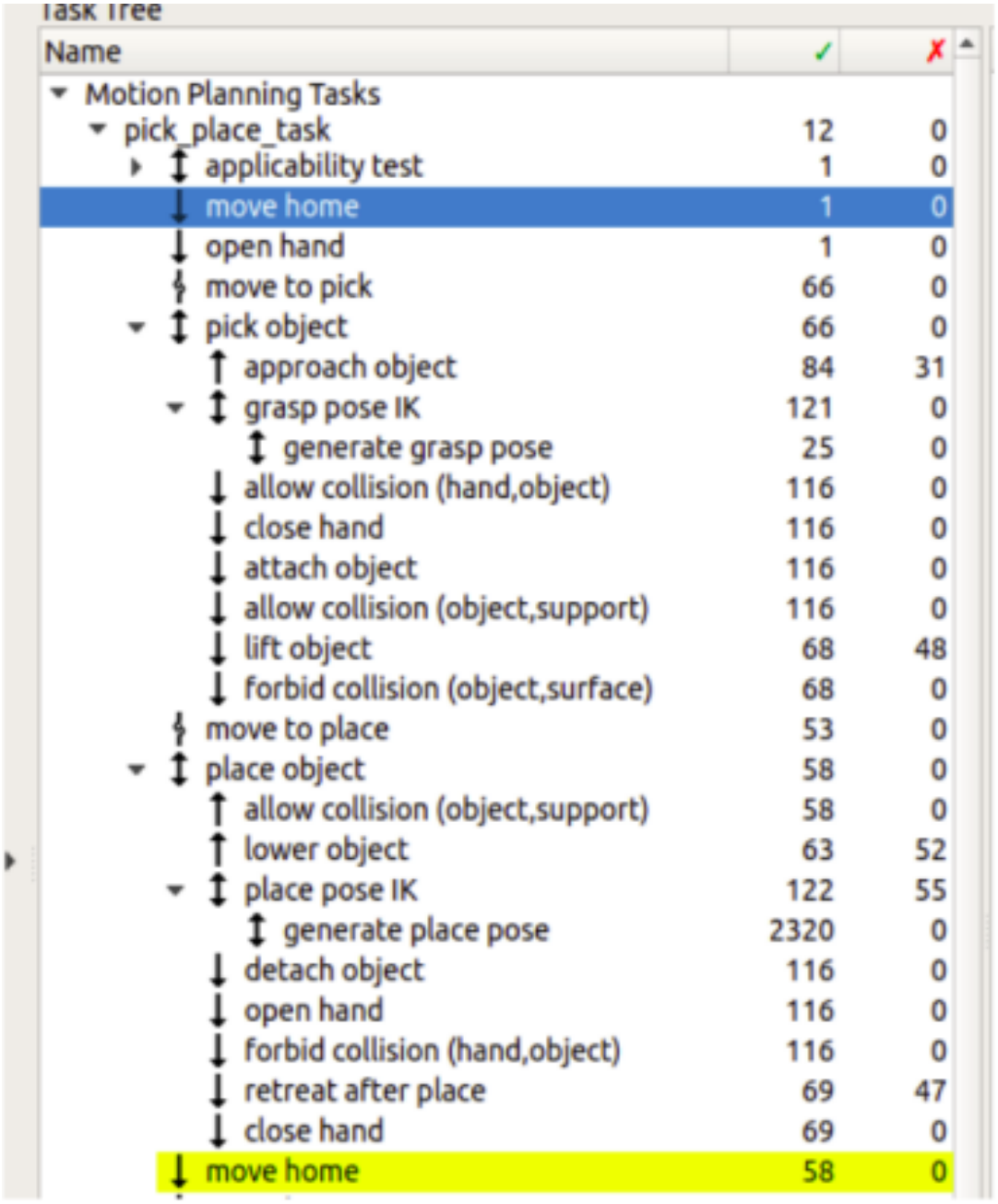


Fig 4.2 PickPlace Stage for Panda arm 1

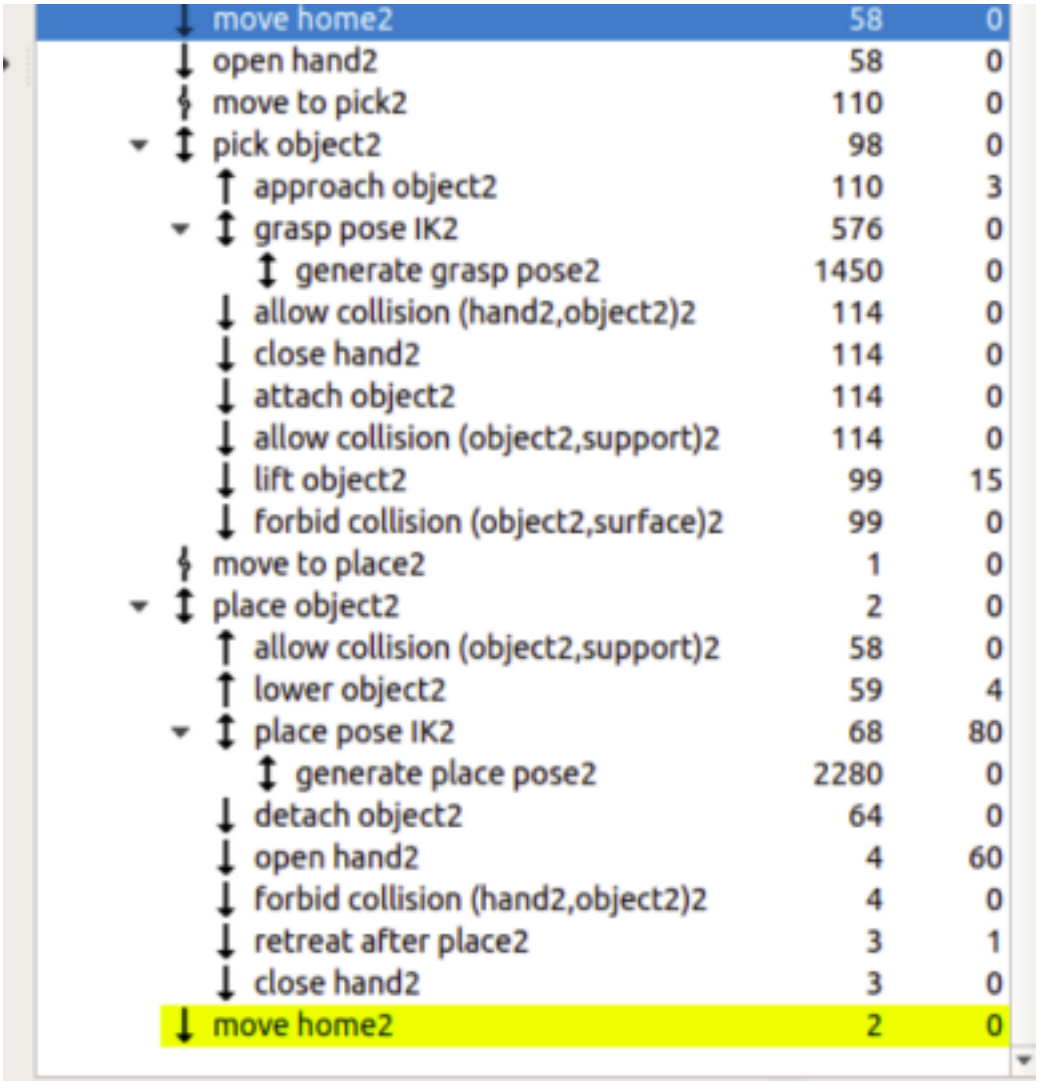


Fig 4.3 Pick place Stage for Panda arm 2

A separate stage is given because picking up an object is a common subtask for many manipulation jobs. Only a few basic properties must be defined to apply this stage to a specific robot, such as the end-effector to employ, the object to grip, and the intended approach and retract directions. Another generic stage, the grabbing stage, offered as a modifiable child stage to the chosen template[7], plans the actual grasping.

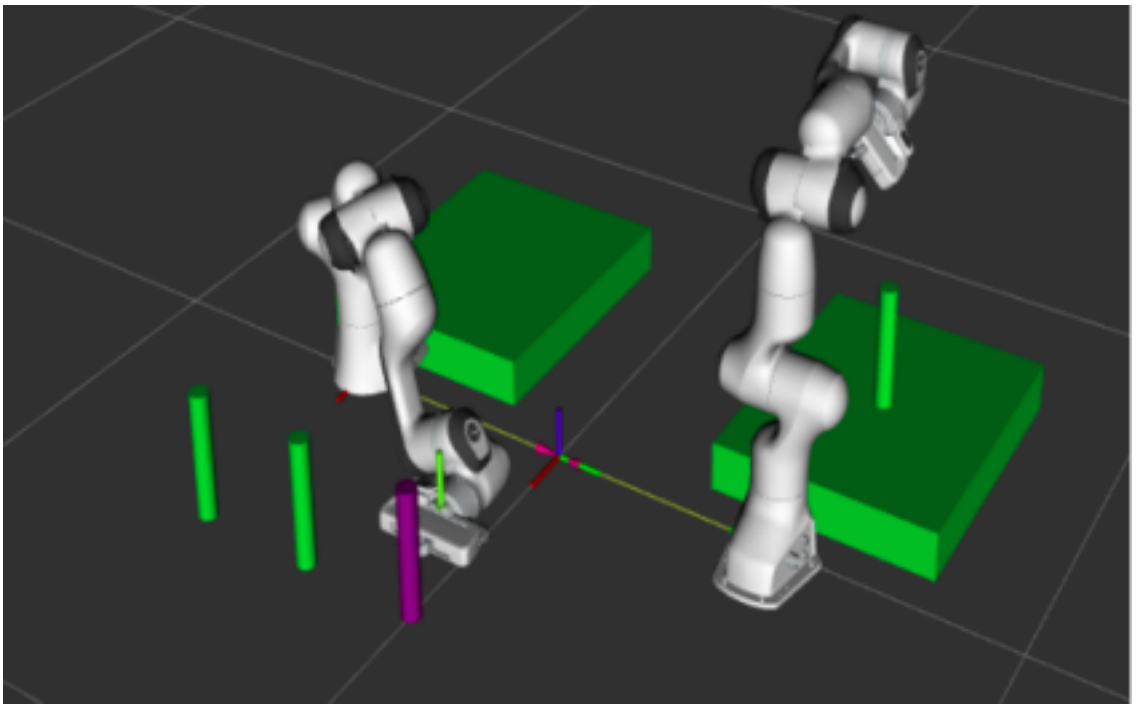


Fig 4.4 Build Structure 1

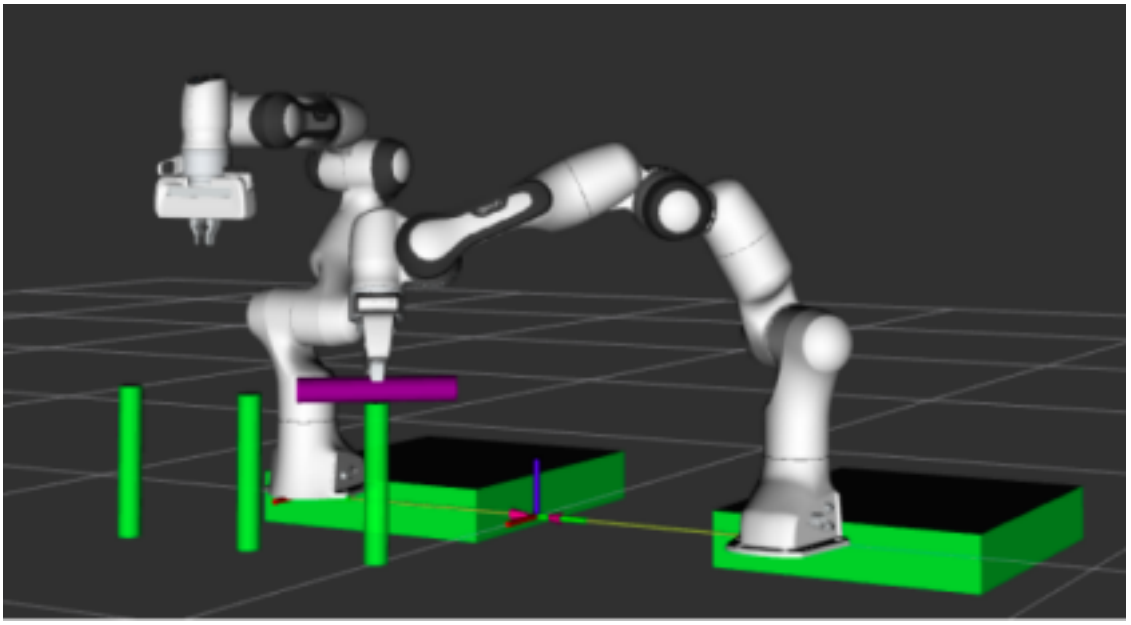


Fig 4.5 Build Structure 2

4.2 Multi-arm pouring using panda

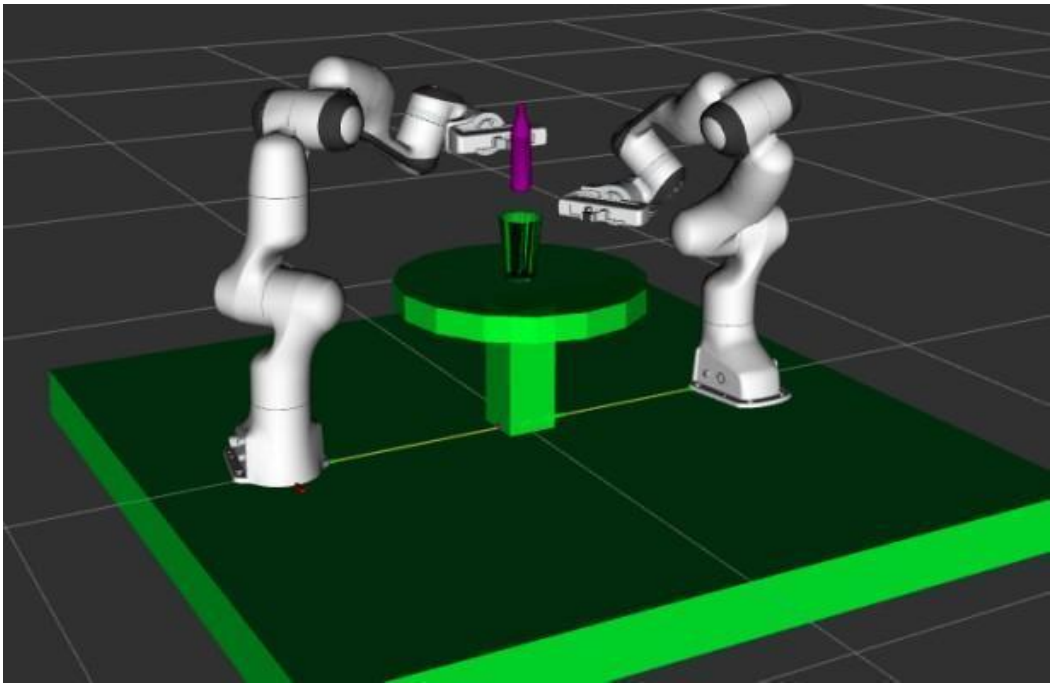


Fig 4.6 Pouring Scene

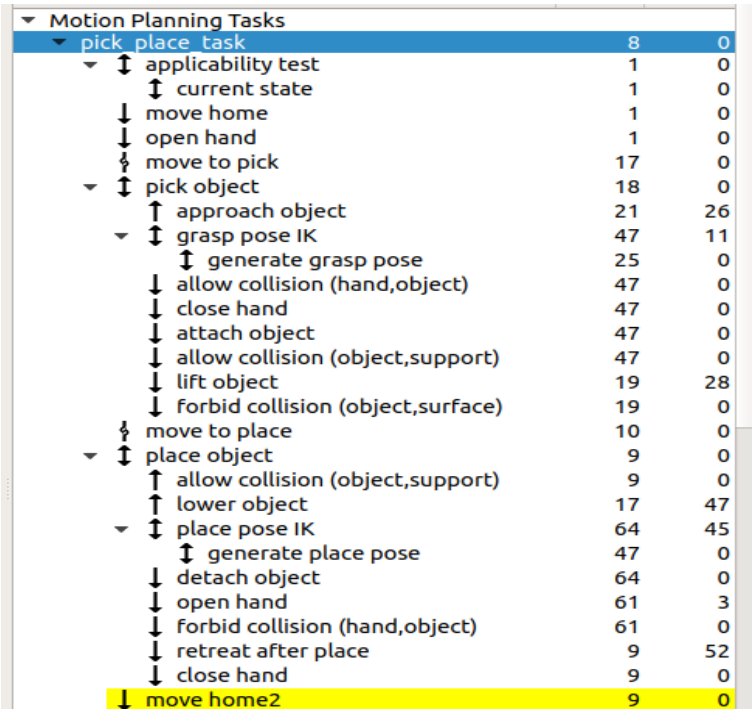


Fig 4.7 Pouring Stage for Panda arm 1

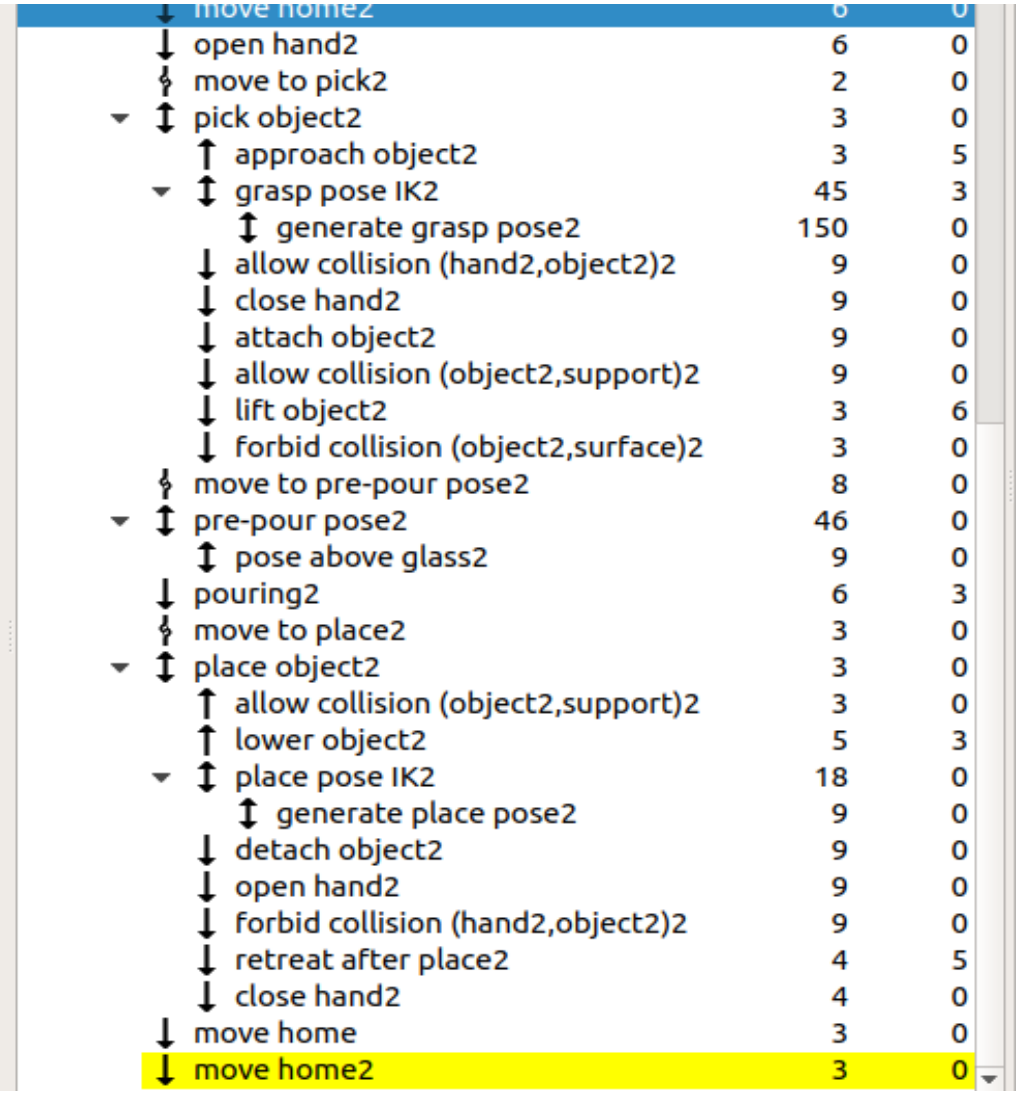
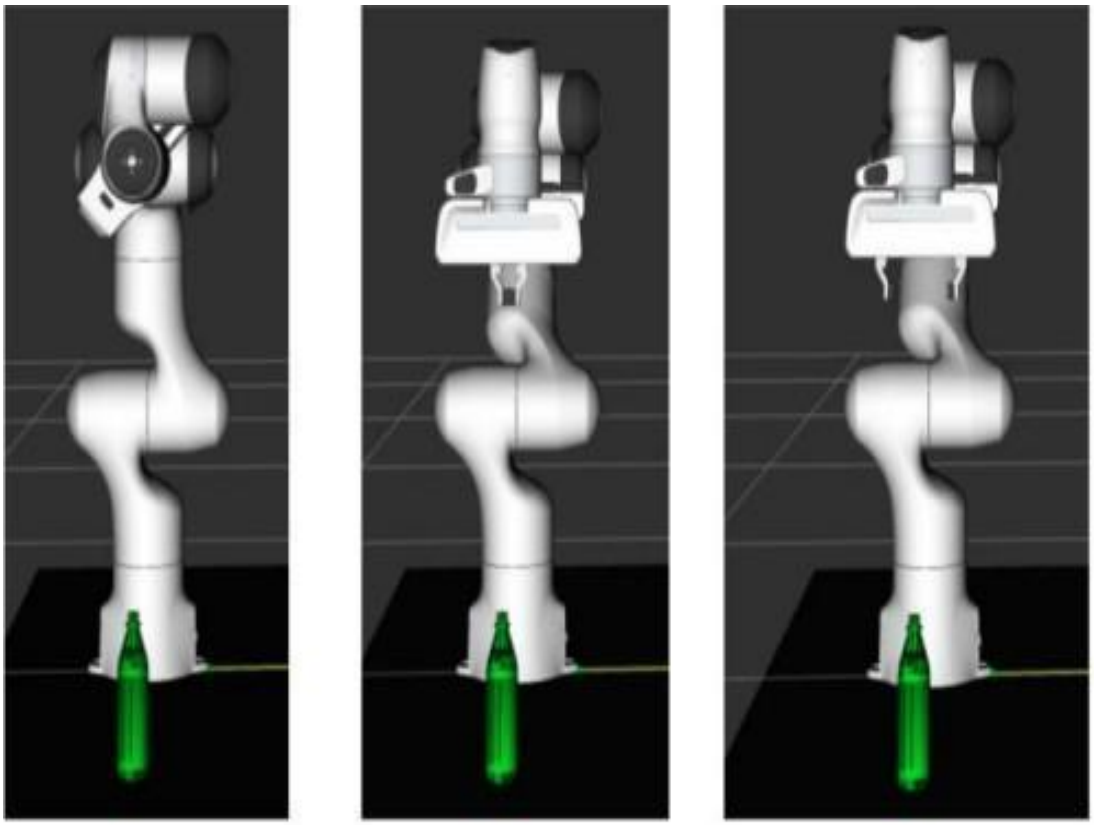


Fig 4.8 Pouring Stage for Panda arm 2

This application demonstrates how to use task pipelines with custom modules. Other stages can be utilized by modifying the parameter in the standard stages, but this task requires a custom pouring stage. The bottle pick stage is used to pick up bottles, while the bottle place stage is used to compute place motion sequences. The bottle is tilted in a pouring motion over the glass for some time to implement the pouring stage, and this part is solved using the cartesian plane and object-centric waypoints. These four generating stages are interconnected, and they all rely on the grip pose selected in the pick stage. As a result, they generally monitor the solutions provided by the grabbing stage and produce matching solutions based on them.

Another consideration during the return of the bottle to its original location is the necessity to avoid spilling the drink. The stage description specifies this requirement. The underlying trajectory planner receives this constraint. An OMPL-based solution with configuration space approximation is used to speed up planning with constraints. The challenge generates its first full answer after 91.38 seconds in our testing utilizing sequential planning[8].

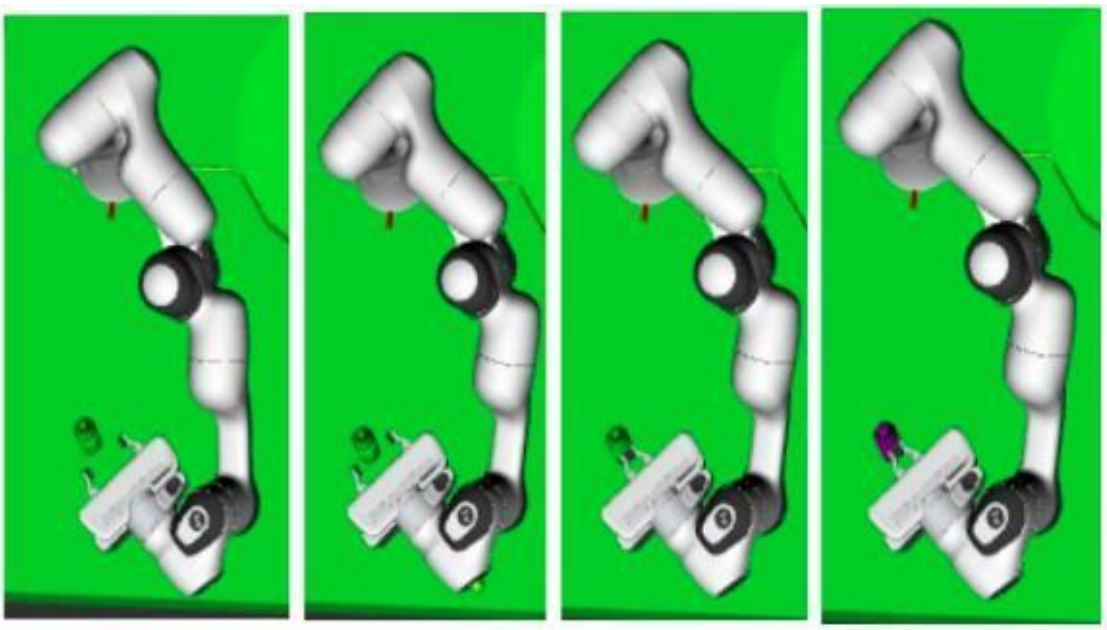
Various Stage Involved in pouring task



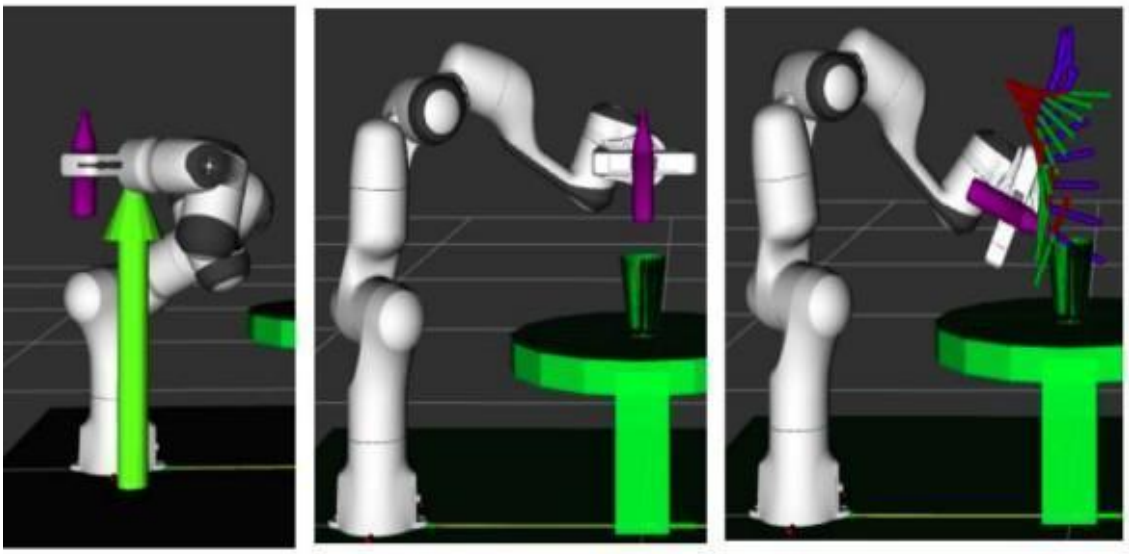
1. Current State

2. MoveTo Home

3. Open Hand



4. MoveTo Pick 5. Approach 6. Grasp 7. Attach



8. Lift 9. MoveTo Pre-Pour 10. Pouring

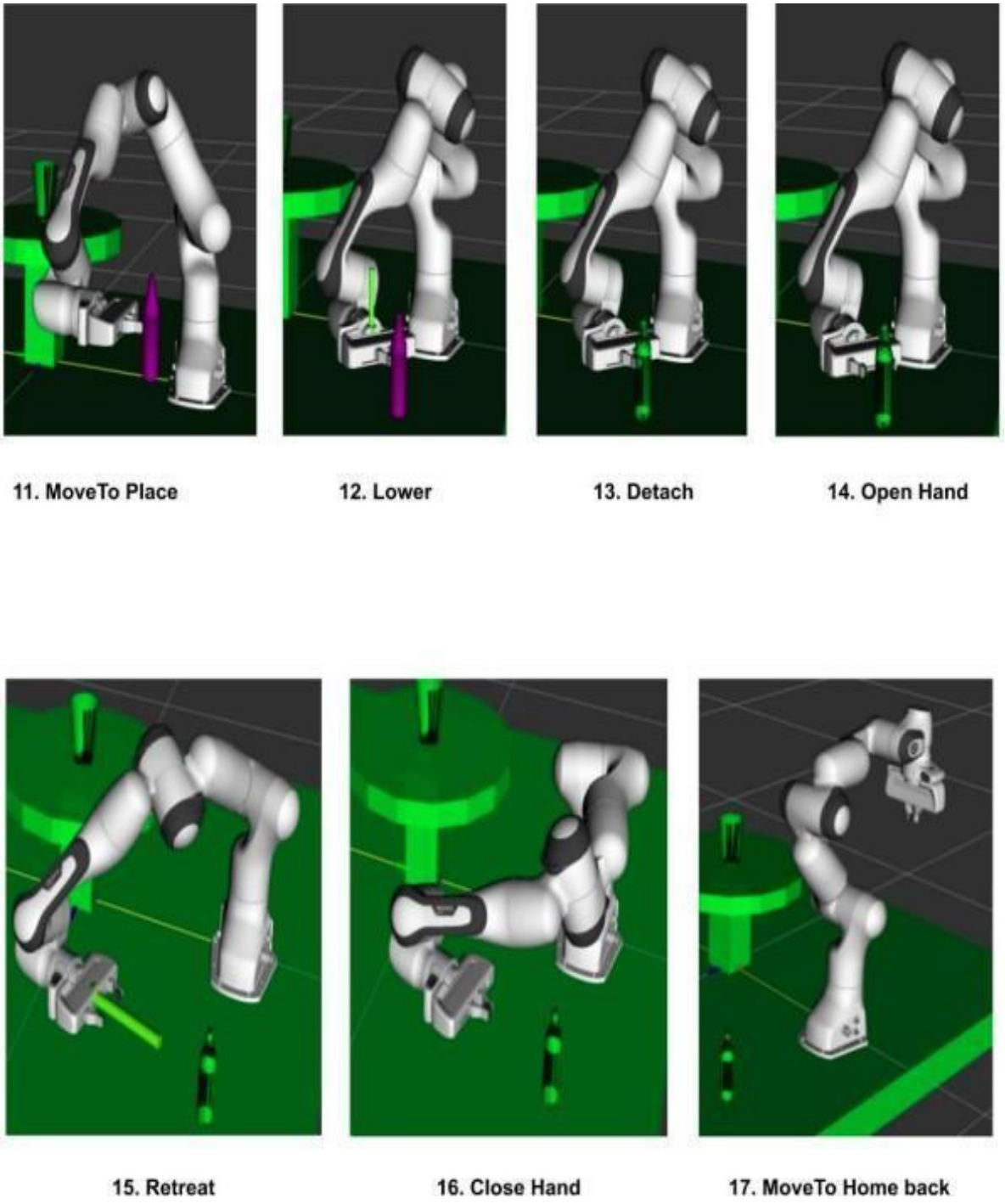


Fig 4.9 Various stages involved in pouring task

4.3 Motion Planning on a Customized Arm

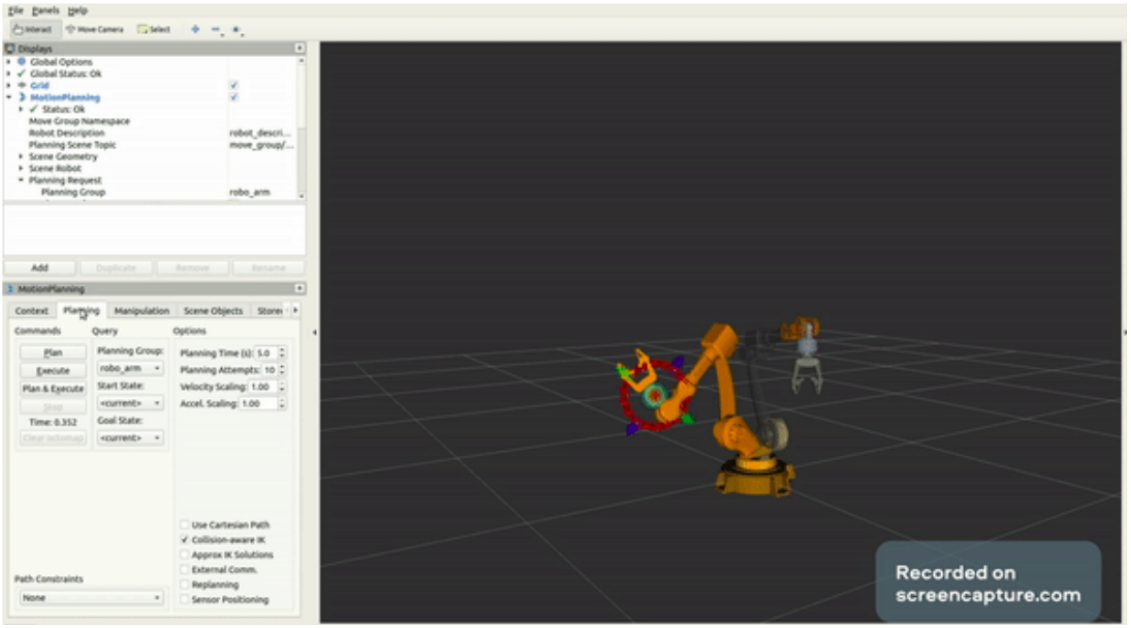


Fig 4.10 Motion Planning on a Customized Arm

4.4 Final Hardware Arm Model

Chapter 5

Conclusion and Future Work

We have discussed the Moveit, motion planning, Robotic Operating system(ROS), and URDF. Also, we have discussed the measured and adaptable planning framework to fill the gap between low-level, high-level motion planning and symbol task planning for manipulating robots. There is a concrete task plan consisting of individually characterized substages. For completing the whole task, the proposed system can yield combined trajectories. In this, failure stages can also be analyzed by visualization. The problematic stages can be isolated, and we can analyze adequately to know the reason for failure. We used MTC to complete multi-arm tasks like building structure and complex pouring tasks simulation.

Then the design and implementation of a robotic arm were presented. The manipulation of the robotic arm is done using Moveit, Robot Operating System frameworks, and rosserial communication as well.

In the future, we can print multi-arms and can develop a coordinated dual-arm mechanical prototype and can perform those complex simulations on real hardware.

Bibliography

[1] “Mtc roscon presentation.” [Online]. Available: https://roscon.ros.org/2018/presentations/ROSCon2018/MoveitTaskPlanning.pdf

[2] ROS Moveit Framework - https://ros-planning.github.io/moveit_tutorials/doc/setup_assistant/setup_assistant_tutorial.html dated on 22-05-2022

[3] I. A. ,Sucan, M. Moll, and L. E. Kavraki, “The Open Motion Planning Library,” IEEE Robotics & Automation Magazine, vol. 19, no. 4, pp. 72–82, December 2012, http://ompl.kavrakilab.org.

[4] Kallman, M., & Mataric, M. (2004). Motion planning using dynamic roadmaps. IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA ’04. 2004.[5] Instuctables https://www.instructables.com/ROS-MoveIt-Robotic-Arm-Part-2-Robot-Controller/

[6] How to Mechatronics https://howtomechatronics.com/tutorials/arduino/diy-arduino-robot-arm-with-smartphone-control/

[7] “MoveIT” [Online]. Available: https://moveit.ros.org/ dated on 22-0522

[8] Hernandez-Mendez, S., Maldonado-Mendez, C., Marin-Hernandez, A., Rios-Figueroa, H. V., Vazquez-Leal, H., & Palacios-Hernandez, E. R. (2017). Design and implementation of a robotic arm using ROS and MoveIt! 2017 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC).

[9] Research Gate https://www.researchgate.net/figure/Model-of-Articulated-robotic-arm_fig5_276062380 dated on 22-05-22

[10] GitHub https://upb.ro/wp-content/uploads/2017/11/Final-report-robotic-arm.pdf

[11] IIT Kharagpur
https://www.iitk.ac.in/dordold/index.php?option=com_content&view=category&layout=blog&id=200&Itemid=219

[12] Arduino Software- https://docs.arduino.cc/hardware/uno-rev3 dated on -22-05-2022

[13] Robu https://robu.in/product/md13s-13amp-dc-motor-driver/

[14] “ROS” [Online]. Available: https://www.ros.org/ dated on 22-05-22

[15] Yi, J., Ahn, M. S., Chae, H., Nam, H., Noh, D., Hong, D., & Moon, H. (2020). Task Planning with Mixed-Integer Programming for Multiple Cooking Task Using dual-arm Robot. 2020 17th International Conference on Ubiquitous Robots (UR).