ENPM 662

INTRODUCTION TO ROBOT MODELLING

FINAL PROJECT

FORGEMATE

UNDER THE GUIDANCE OF**: PROF. REZA MONFERADI**

AUTHORS**: PRATEEK VERMA, LAVANYA SURESH KANNAN**

TABLE OF CONTENTS:

1. Introduction and organization:

Robotic Technology continues to advance and now-a-days it is very rare to see human workers on site. Robotic manipulator arms are very versatile and can complete many varieties of tasks. In this project, we will demonstrate a simulated manipulator arm with six degrees of freedom (DOF) capable of performing pick and place operations in the industry. This robot utilizes the inverse kinematics toc complete the grabbing and placing operations. The report starts with mathematical models and how it will operate. The kinematics requires the matrix theory and the use of Denavit-Hartenberg parameters. After the math is explained, the robot is focused on the model. By creating the model in SolidWorks or altering the already existing model, the 6-DOF manipulator arm will be exported as URDF to use in ROS. Then the gazebo simulation using ROS is explained. In the conclusion we will discuss our results in the project.

Keywords: Kinematics, Gazebo, ROS, URDF, SolidWorks.

2. Motivation and application:

Forging is a manufacturing process involving the shaping of metal through hammering, pressing, or rolling. The forging done with the help of presses is known as press forging. Press operations have been recognized as among the most hazardous performed in the factory. Over the years, due to human ingenuity, there have been many attempts to defeat the safety measures and accidents still occur which leads to some serious injuries to the workers such as losing a limb or even death. Our intent is to build a robot that will address

the present concern of humans working in immensely dangerous conditions on the factory floor. The proposed model of our 6 DOF industrial robotic arm on a mobile robot can operate in a forge and foundry workshop floor and move fast in any surrounding to pick and place the required metal/alloy part under the press. The human-like dexterity of this robot makes it efficacious in diverse applications in a variety of industries such as manufacturing, material handling, welding, painting, etc.

Figure (1) and Figure (2) show the process of forging on a factory floor.

Figure1

Figure2

3. Robot description:

In this project we used the existing robot model which is the PUMA-560 robotic arm. It is a 6-DOF robotic arm with 6 links and 6 revolute joints. Since our main aim is to do pick and place operation, we attached a gripper to the end effector. The gripper has two links that are connected to the end effector. So totally the arm has 8 links. The base needs to be sturdy and heavy to allow for full extension of the arm in case of maximum torque on the base. Figure 3 and Figure 4 shows the PUMA 560 robot model. Also figure 5 shows the order of manipulator arms.

Figure 3

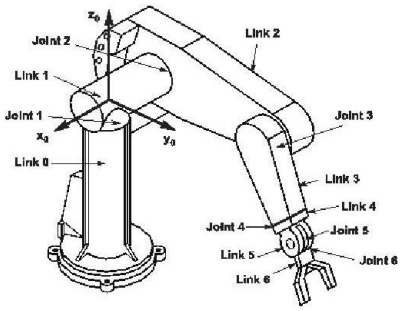


Figure 4

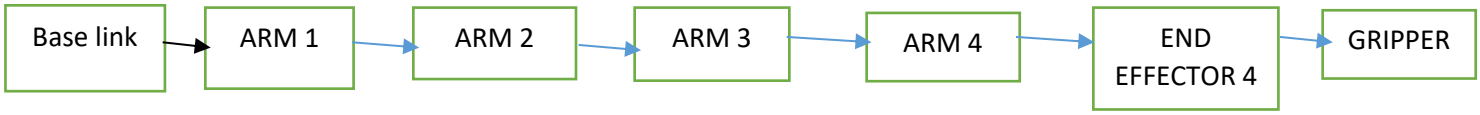


FIGURE 5: ORDER OF MANIPULATOR ARMS

4. Robot Appropriateness for the task:

The robot we intend to make is a manipulator that can move in a desired direction with proper balance. The robot will be equipped with a 6 DOF which can perform multiple tasks such as pick and place, cutting, drilling, sawing, etc. A proper gripper according to the required task is to be attached to the end effector for best performance.

5. Scope Description

The study will be focused on implementing the kinematics and applying them to the SOLIDWORKS2021 model robot we make. Dynamics of the robot will also be considered so that the actuators can perform without failure in the real world.

Ambitious goals:

1. Designing a differential drive and placing the manipulator on the top to perform various tasks.
2. Designing smaller robots (multi-agent system), working in coordination with each other replacing one big robot with six to eight small powerful robots reducing the investment cost.
3. Placing another arm manipulator on the top of the robot, so that it can support and perform lots of other tasks like material handling etc.

6. Scope Appropriateness:

As the problem statement says, the robot is supposed to do the work of picking and placing hot metal/alloy parts and positioning them under the press. Clearly, the desired domain in which the robot is supposed to work is very much dangerous and deadly. Employing a robot to perform such a task is a much better and favorable option as compared to making humans (laborers) do such tasks. Its advantages are no danger to the human life of workers or for their upcoming generations, reduced risk, a faster rate of work, eliminating factors like fatigue, etc. This study or project will make us understand the basic modeling of any robot in the real world. We will have a brief understanding of the Kinematic and Dynamic principles. We took this specific robot model to understand rigid body transformation, Inverse Kinematics and Forward Kinematics, contact modeling and grasping. Modeling this robot will make us learn the links and joints mechanism in actual modeling scenarios. Out of other things, this project will make me learn the grasping mechanism, trajectory, and path planning. Also, this project will make sure we learn to integrate controllers in ROS-Gazebo environments. This study covers all the essentials required to model a robot to work in the real world. This will help

all the industries working toward a robot that can work in extreme conditions and challenging environments.

7. Model Assumptions:

The proposed design for our robot will be a 6 DOF robotic arm. Going against the conventional human being working on the factory floor. This robotic arm will perform the task of picking and placing objects and several other tasks. Robot Model design assumptions:

1. All the joints and objects are considered to be rigid.
2. The friction and the other external disturbances are not taken into account.
3. Robot self-collision is not considered, only collision with the external obstacles is taken into account for this scope of the project.
4. The path of the arm or the robot is just one solution among all the other solutions it can have, this may or may not be the optimal solution.
5. As there is no analytical solution for a general 6-DOF manipulator arm with non-zero offset.

8. Approach to Performing the Work:

The first step is designing a CAD model in SOLIDWORKS 2021. We imported a PUMA 560 CAD model from GRABCAD. Then the design will be exported as URDF. Figure 6 shows the model of the robot in SolidWorks

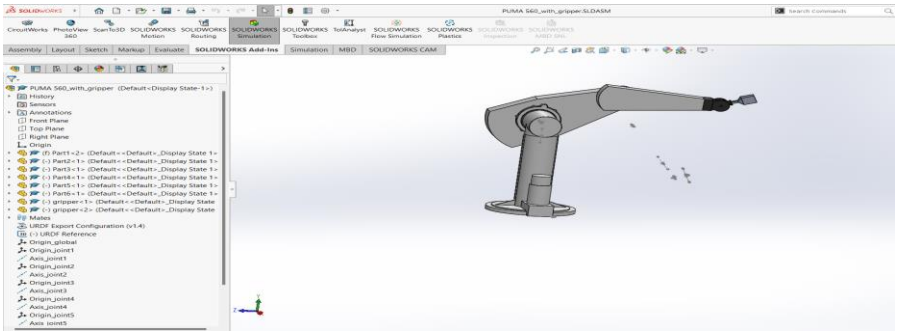
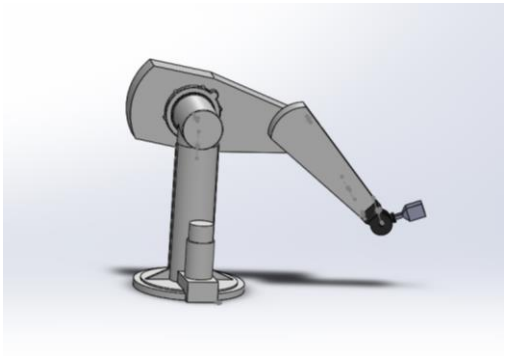
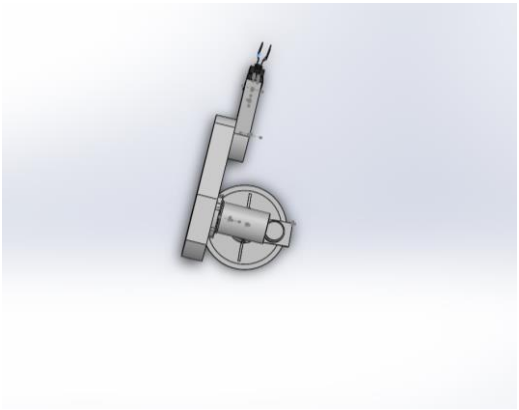


FIGURE 6



SIDE VIEW



TOP VIEW

While exporting the mass of inertia along each axis was set up slightly higher than the actual value as later on in the ROS package controllers were supposed to be added with high PID values. Once the assembly was done the package was exported to ROS. The next step was to add all the necessary dependencies while creating a package in the workspace. The URDF file was integrated in a new XACRO file with gazebo plugin for effort joint position/velocity controllers. The controller plugin used for this project was gazebo_ros_control from the file libgazebo_ros_control.so. To execute the controllers for the specific joints, a configuration YAML file was made with the same namespace of the plugin.

Here all the joints were defined which were to be controlled. The PID values were tuned using the inbuilt GUI of RQT. The launch file was modified to initialize the controller. After the basic setup the model was spawned in a gazebo environment and was visualized in RVIZ. Finally, a python script was written as a ROS publisher which takes in the input and calculates the inverse kinematics and outputs the joint/actuator angles. Figure 7 shows the config file where we updated the controllers. Figure 8 shows the changes we made in RQT publisher.

```
forgemate:
  joint_state_controller:
    type: joint_state_controller/JointStateController
    publish_rate: 20

  joint_1_position_controller:
    type: effort_controllers/JointPositionController
    joint: joint1
    pid: {p: 2000.0, i: 50, d: 100.0}
        #pid: {p: 100.0, i: 30.0, d: 40.0}

  joint_2_position_controller:
    type: effort_controllers/JointPositionController
    joint: joint2
    pid: {p: 2000.0, i: 50, d: 100.0}
        #pid: {p: 100.0, i: 30.0, d: 40.0}

  joint_3_position_controller:
    type: effort_controllers/JointPositionController
    joint: joint3
    pid: {p: 2000.0, i: 50, d: 100.0}
        #pid: {p: 2000.0, i: 50, d: 100.0}

  joint_4_position_controller:
    type: effort_controllers/JointPositionController
    joint: joint4
    pid: {p: 2000.0, i: 50, d: 100.0}
        #pid: {p: 100.0, i: 30.0, d: 40.0}

  joint_5_position_controller:
    type: effort_controllers/JointPositionController
    joint: joint5
    pid: {p: 2000.0, i: 50, d: 100.0}
        #pid: {p: 100.0, i: 30.0, d: 40.0}

  joint_6_position_controller:
    type: effort_controllers/JointPositionController
    joint: grip_joint1
    pid: {p: 2000.0, i: 50, d: 100.0}
        #pid: {p: 100.0, i: 30.0, d: 40.0}

  joint_7_position_controller:
    type: effort_controllers/JointPositionController
    joint: grip_joint2
    pid: {p: 2000.0, i: 50, d: 100.0}
        #pid: {p: 100.0, i: 30.0, d: 40.0}
```

FIGURE 7

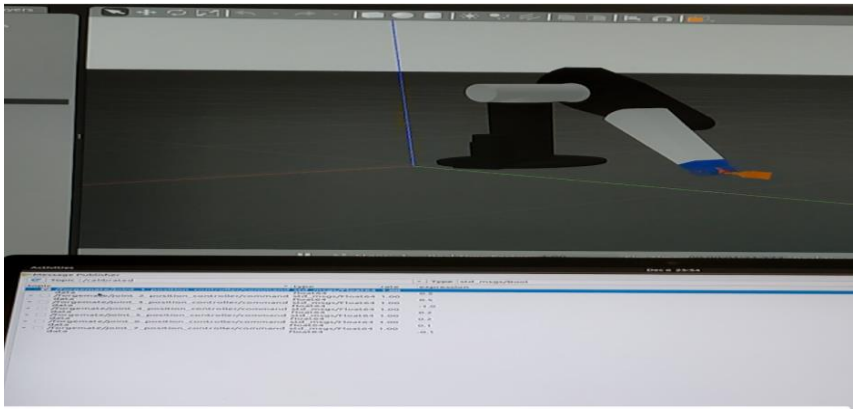


FIGURE 8

Table 1 shows the tools we used in the entire project

Reference design from GRABCAD
SolidWorks 2019 with URDF exporter package
Python code for kinematics calculations and verifications
Rviz for visualization (ROS)
Gazebo for spawning the model and performing the task. (ROS)

Table 1: Tools used for the model and simulation

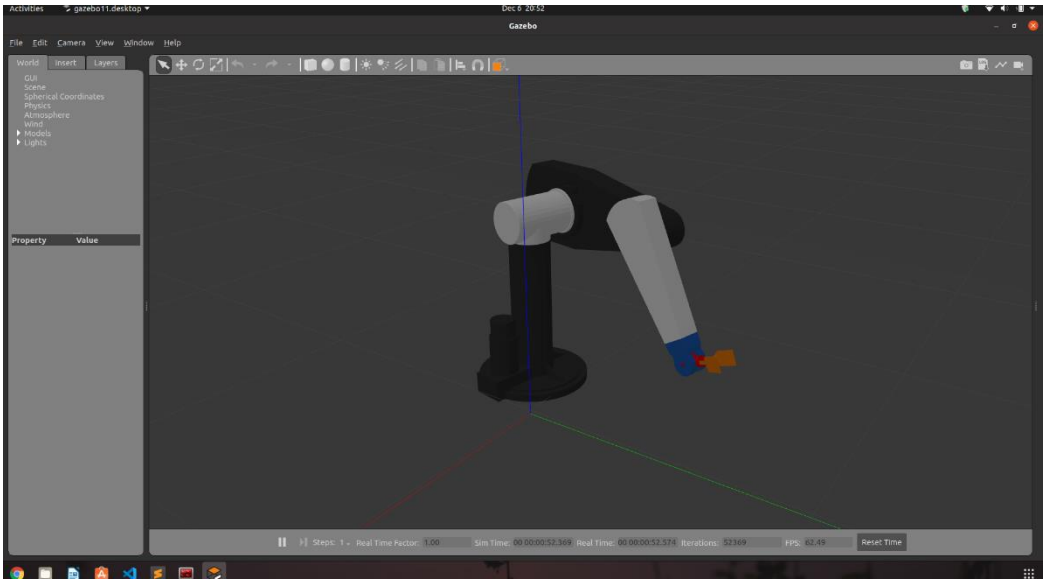


FIGURE 9: MODEL IN GAZEBO WORLD

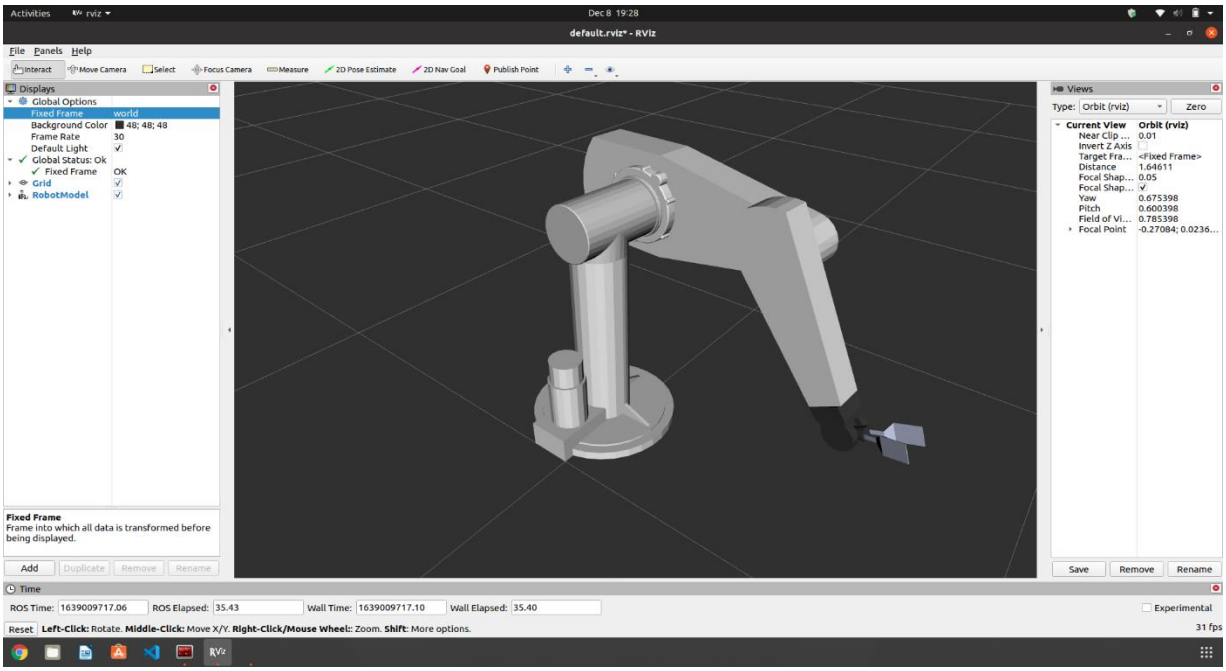


FIGURE 10: ROBOT IN Rviz

9. Kinematics:

Creating the forward and inverse kinematics of the arm is an essential step in creating the simulated model. Based on this model the Denavit– Hartenberg (DH) table can be computed. This table gives the reference frames for the model to the links of spatial kinematic chains. By assigning the specific joint angles desired, this table gives us the transformation matrices for the configuration between the frames. Using the DH table, the transformation matrices between the links can be found shown in Figure (8). These equations were computed in Python. These scripts are used in ROS later. In IK, the end-effector position is given instead of the joint angles, so the angles must be calculated based on the desired location. Figure 9 shows the transformation matrixes we calculated between each frame. We have also included the DH parameter in table 2.

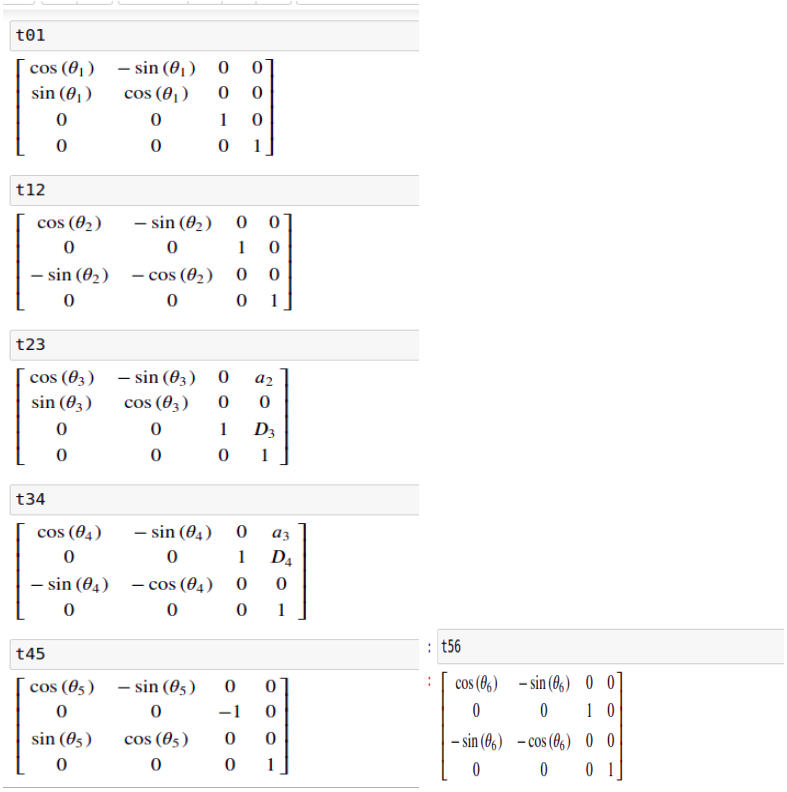


Figure 10: Values obtained from python code.

Link	Link offset	Joint angle	Link length	Twist angle
1	0	θ_1	0	Pi/2
2	0	θ_2	A2	0
3	D3 = 0.150	θ_3	A3	- Pi/2
4	D4 = 0.43	θ_4	0	Pi/2
5	0	θ_5	0	- Pi/2
6	0	θ_6	0	0

Table 2. DH parameters.

10. <u>VALIDATION PLAN</u>:

● Checking the appropriateness of the CAD model: ○ We can check whether the URDF file exported from SOLIDWORKS to Gazebo is properly parsed or not by using the following terminal command:

$ check_urdf .urdf

- Whenever we make any changes to our model (in the URDF file) we can check the appropriate parent-child relationship between the links using the above command.
- Joint movement and performance is validated using RQT publisher by providing multiple input values and observing the output motion.
- Publisher node was tested using the **$ rostopic list** command.

● We wrote python scripts to calculate the forward and inverse kinematic solutions using the sympy library. The correctness of the solutions to the homogeneous transformations and the kinematics can be found by comparing analytical solutions with the output of the automated scripts in Python to perform the same calculations.

● Validation for the movement of the robot has been done through simulation using ROS in a Gazebo environment.

11. LEARNING OUTCOMES:

1. Understood and calculated Forward and Inverse kinematics for any real-world robot.

2. Learned how to simulate in different environments and model the real world in ROS - Gazebo & Rviz.

3. Understood link/joint parent-child relationships in SOLIDWORKS and how to model them.

4. Understood ROS and its functionalities like Teleop etc.

5. Trajectory planning for the manipulator arm

6. To pick and place objects avoiding obstacles in free space.

7. Validated and verified calculated Forward and Inverse kinematics solutions using a developed simulated environment.

8. Understood the integration of controllers in a simulated environment.

9. Understood grasping analysis of a manipulator's arm in a real-world scenario.

12. FUTURE WORK:

- In future, this robotic arm can be placed over a mobile robot.
- This will enhance the mobility of the robot.
- The gripper in the end effector can be modified in the future for the specific tasks.
- This will allow us to perform pick and place operation under various industrial applications.

13. ISSUES FACED:

Initially we faced a lot of issues while making the CAD model. Because of which we decided to use the GRABCAB model. When launched in gazebo the robot arms where floating, which was then fixed. The gripper was jittering for a long time which was the fixed by changing the joints. Tunning the PID values took a lot of time than the other process.

14. CONCLUSIONS:

The implementation of the robotic arm in the Gazebo and RViz environment was a success. Though it did not reach our ambitious goals of performing several varieties of complex tasks in simulation, it did meet all the set expectations.

15. APPENDIX:

Code for publisher

```
#!/usr/bin/env python
import rospy
import time

from threading import Thread
from std_msgs.msg import Float64

def get_publishers():
    topic_name = '/forgemate/joint__position_controller/command'
    publishers = []
    for i in range(7):
        topic_name     =     '/forgemate/joint_'     +     str(i+1)     +
'_position_controller/command'
        publisher = rospy.Publisher(topic_name, Float64, queue_size=10)
        publishers.append(publisher)

    return publishers


def main():

    trajectory = []
    # angles_t0 = [0.3] * 7
    # angles_t1 = [0.9] * 7
    # angles_t2 = [0.5] * 7

    angles_t0 = [0.1, 0.1, 0.0, 3.0, -1.0, 0.0, 0.0]
    angles_t1 = [0.1, 0.1, 0.0, 3.0, -1.0, 0.2, 0.2]
    angles_t2 = [0.1, 0.3, 0.5, 3.0, -1.0, 0.2, 0.2]
    angles_t3 = [0.9, 0.3, 0.5, 3.0, -1.0, 0.2, 0.2]
    angles_t4 = [0.9, 0.1, 0.0, 3.0, -1.0, 0.2, 0.2]
    angles_t5 = [0.9, 0.1, 0.0, 3.0, -1.0, 0.0, 0.0]
    angles_t6 = [0.9, 0.3, 0.5, 3.0, -1.0, 0.0, 0.0]
    angles_t7 = [0.1, 0.3, 0.5, 3.0, -1.0, 0.0, 0.0]

    trajectory.append(angles_t0)
    trajectory.append(angles_t1)
    trajectory.append(angles_t2)
    trajectory.append(angles_t3)
    trajectory.append(angles_t4)
    trajectory.append(angles_t5)
    trajectory.append(angles_t6)
    trajectory.append(angles_t7)


    publishers = get_publishers()

    # for t in range(len(trajectory)):
```

```
    #     angles = trajectory[t]
    #     for i in range(len(angles)):
    #          rospy.loginfo('Angle = {} for joint = {}'.format(angles[i],
i+1))
    #         publishers[i].publish(angles[i])

    #     rospy.sleep(2.0)

    for t in range(len(trajectory)):
        angles = trajectory[t]
        t1 = Thread(publishers[0].publish(angles[0]))
        t2 = Thread(publishers[1].publish(angles[1]))
        t3 = Thread(publishers[2].publish(angles[2]))
        t4 = Thread(publishers[3].publish(angles[3]))
        t5 = Thread(publishers[4].publish(angles[4]))
        t6 = Thread(publishers[5].publish(angles[5]))
        t7 = Thread(publishers[6].publish(angles[6]))

        t1.start()
        t2.start()
        t3.start()
        t4.start()
        t5.start()
        t6.start()
        t7.start()
        rospy.sleep(2.0)




if __name__=="__main__":
    rospy.init_node('forgemate_node', anonymous=True)
    rospy.loginfo('Initialized node...')
    while not rospy.is_shutdown():
        main()
```

16. REFERENCES:

-
- https://www.mdpi.com/2076-3417/11/9/3985
- https://www.astesj.com/publications/ASTESJ_050340.pdf
- https://charitha94.medium.com/how-can-you-simulate-your-cad-model-of-the-robot-in-ros-9c6eda86aeb6
- https://www.youtube.com/watch?v=p9c9KoKjEe0
- All the reference provided in elms.

- Robot Modeling and Control 1st Edition by Mark W. Spong (Author), Seth Hutchinson (Author), M. Vidyasagar (Author)

Links:

Github link: https://github.com/lavanyasureshkannan/FORGEMATE_FINAL

YouTube video link: https://www.youtube.com/watch?v=rh0qNyymAoU