INTRODUCTION TO ROBOT PROGRAMMING

UNIVERSITY OF MARYLAND

Final Project

Project Final Report

Project Partners:

Neha Madhekar (*UID: 119374436*)

Harshal Shirsath (*UID: 119247419*)

Yashas Shetty (*UID: 119376348*)

Professor:

Zeid Kootbally

16th December 2022

Contents

List of Figures

1 Introduction

The goal of the project is to move the robot to a particular position and then detect an aruco marker in the vicinity of that position. Once the marker is detected, the robot moves to a goal position based on the message in the aruco marker. This kind of application is usually used for object detection and localization of objects or products. Whereas, in this project these fiducial markers would be used to retrieve information of the final destination where the robot has to reach.

One of the best examples of this type of technology is in Amazon warehouse robots called Proteus. These robots work in similar fashion as of this project,where the Aruco markers code are pasted on the ground. Robot detects the markers for the future commands, which informs the robot about the desired destination to be reached for receiving or delivering packages.

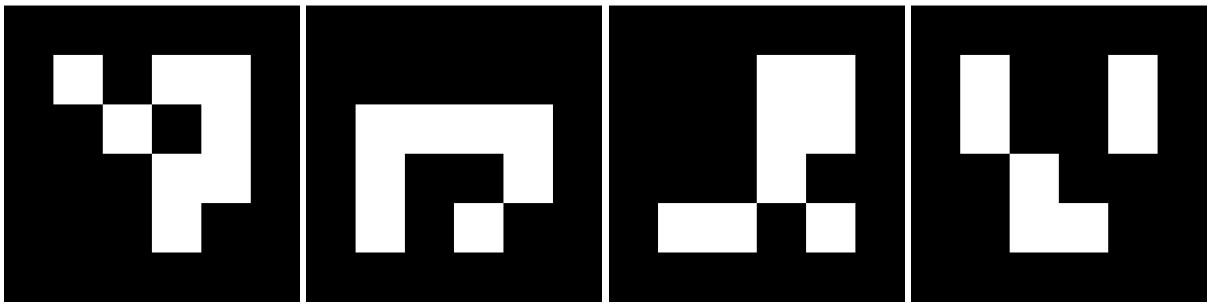

Figure 1: Aruco Markers

2 Approach

The project was split into multiple fragments to approach. They are listed below:

A. ODOM UPDATER PACKAGE

1. Broadcast frame /robot1/base_footprint as a child of robot1/odom. After this the trees will be connected: Initially, the tree of frames is disconnected. All the frames have to be connected to perform the transforms correctly.To connect the trees broadcaster has to be written which broadcasts frame /robot1/base_footprint as a child of robot1/odom. This is done by creating the package odom_updater. Node is created in the package to broadcast the /robot1/base_footprint as a child of /robot1/odom. This broadcaster cannot be a static one and the robot broadcaster has to subscribe to the topic /robot1/odom which will retrieve the current position of the robot in /odom which is done by subscriber callback. The tree below is the connected one in which the child is connected to the parent.

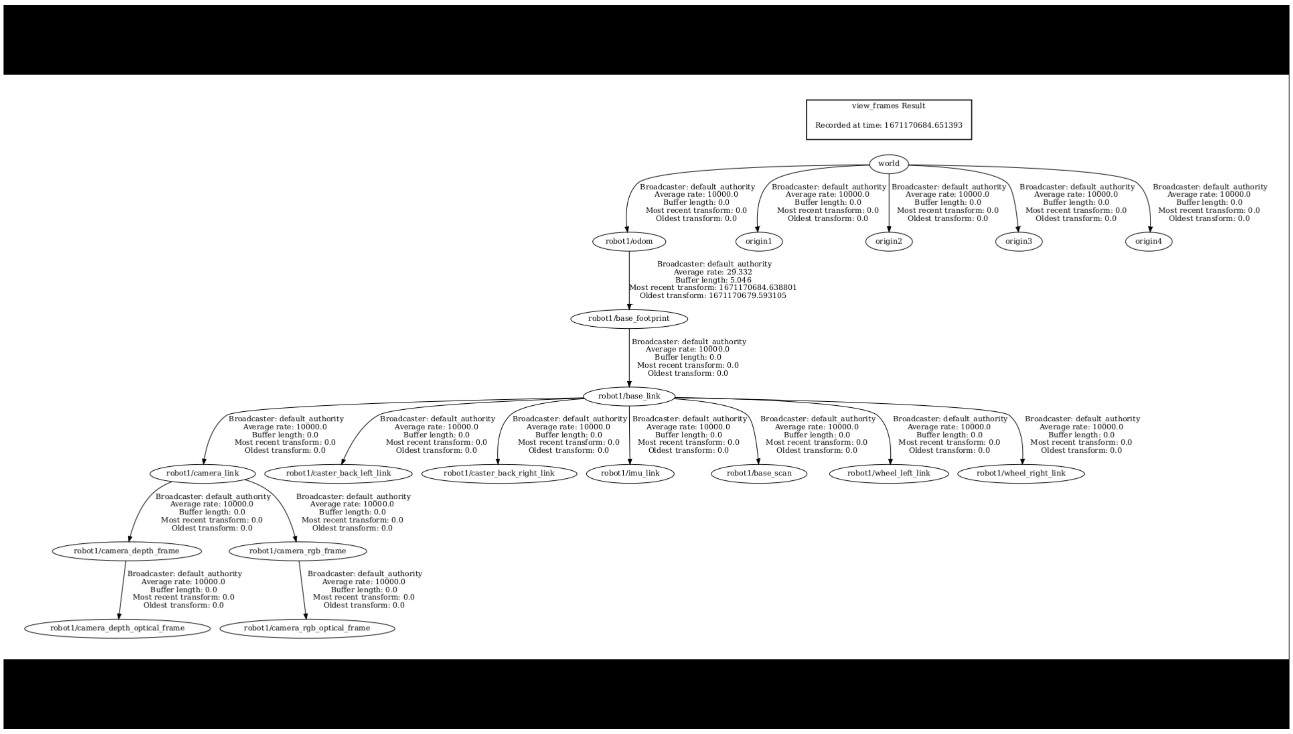

Figure 2: Connected Tree(Parent− >Child)

B. TARGET REACHER PACKAGE

1. Declare parameters defined in a file final_ params .yaml.

2. Read aruco_target coordinates and command the robot to go to that position.

The goal1 coordinates for the robot are read from the yaml file and the robot is commanded to go to goal1 using the method in bot_controller package.

3.Subscribe to topics /goal_reached and /aruco_ markers

It is given that once the goal1 is reached, a data ”true” will be published in the topic goal_reached and once the aruco marker is detected, the data will be published on /aruco_markers topic. Hence subscriber to both the topics is written.

4. Once, the data true is published on goal reached, publish angular velocity on robot1.

Further on ,the task makes the robot to rotate around itself unless and until it finds the fiducial marker and commands the robot to reach the particular ID position.

5. Once the marker is detected (aruco_markers publishes data) stop rotation of the robot.

6. Retrieve the marker ID from the published message and determine the target coordinate with respect to a given frame ID, mentioned in the . yaml file.

7. Create a frame final destination and broadcast as a child of frame defined by frame ID. After this, tree of frame looks like-

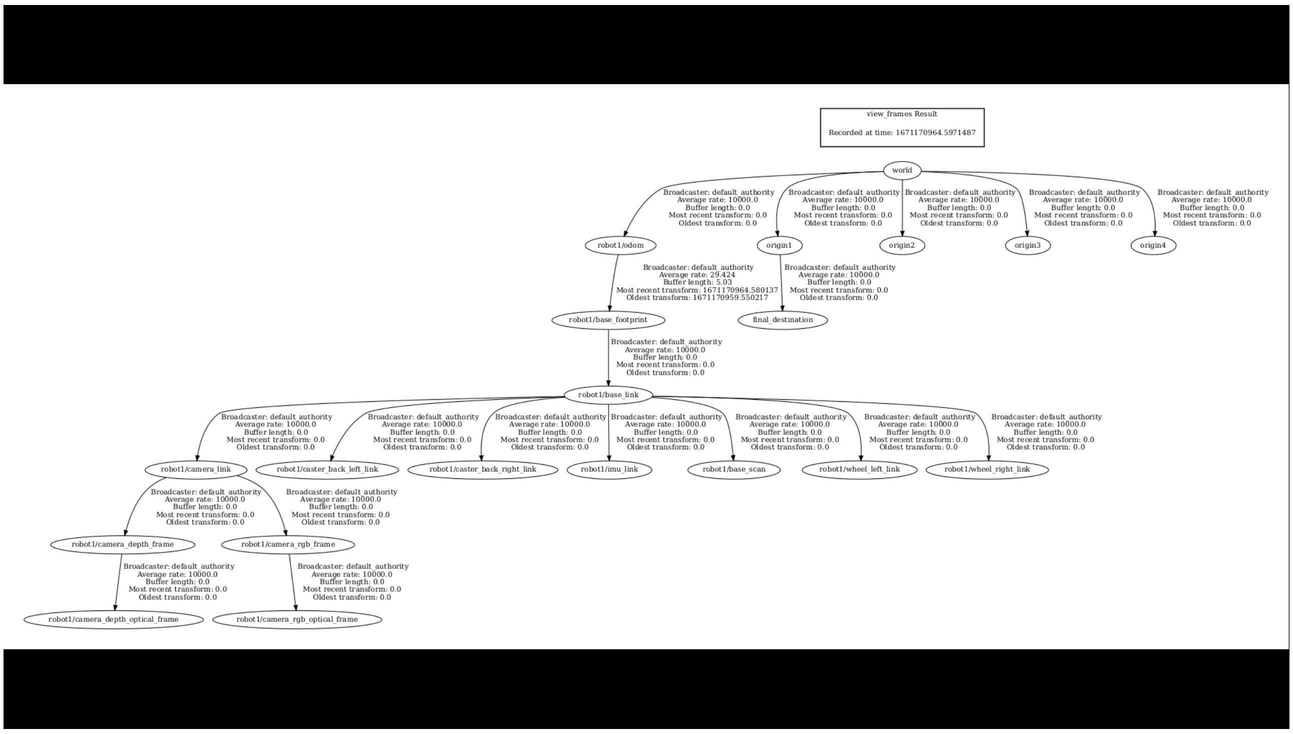
view_frames Result
Recorded at time: 1671170964.5971487
world
robot1/odom
origin1
origin2
origin3
origin4
Broadcaster: default_authority
Average rate: 29.424
Buffer length: 5.03
Most recent transform: 1671170964.580137
Oldest transform: 1671170959.550217
Broadcaster: default_authority
Average rate: 10000.0
Buffer length: 0.0
Most recent transform: 0.0
Oldest transform: 0.0
robot1/base_footprint
final_destination
robot1/base_link
robot1/camera_link
robot1/caster_back_left_link
robot1/caster_back_right_link
robot1/imu_link
robot1/base_scan
robot1/wheel_left_link
robot1/wheel_right_link
robot1/camera_depth_frame
robot1/camera_rgb_frame
robot1/camera_depth_optical_frame
robot1/camera_rgb_optical_frame

Figure 3: Final Destination

8.Compute the transform between final destination and robot1_odom by writing transform listener.

9. Translation part of the above matrix represents the final goal coordinates in robot1_odom frame.

10. The robot is commanded to go to the final destination.

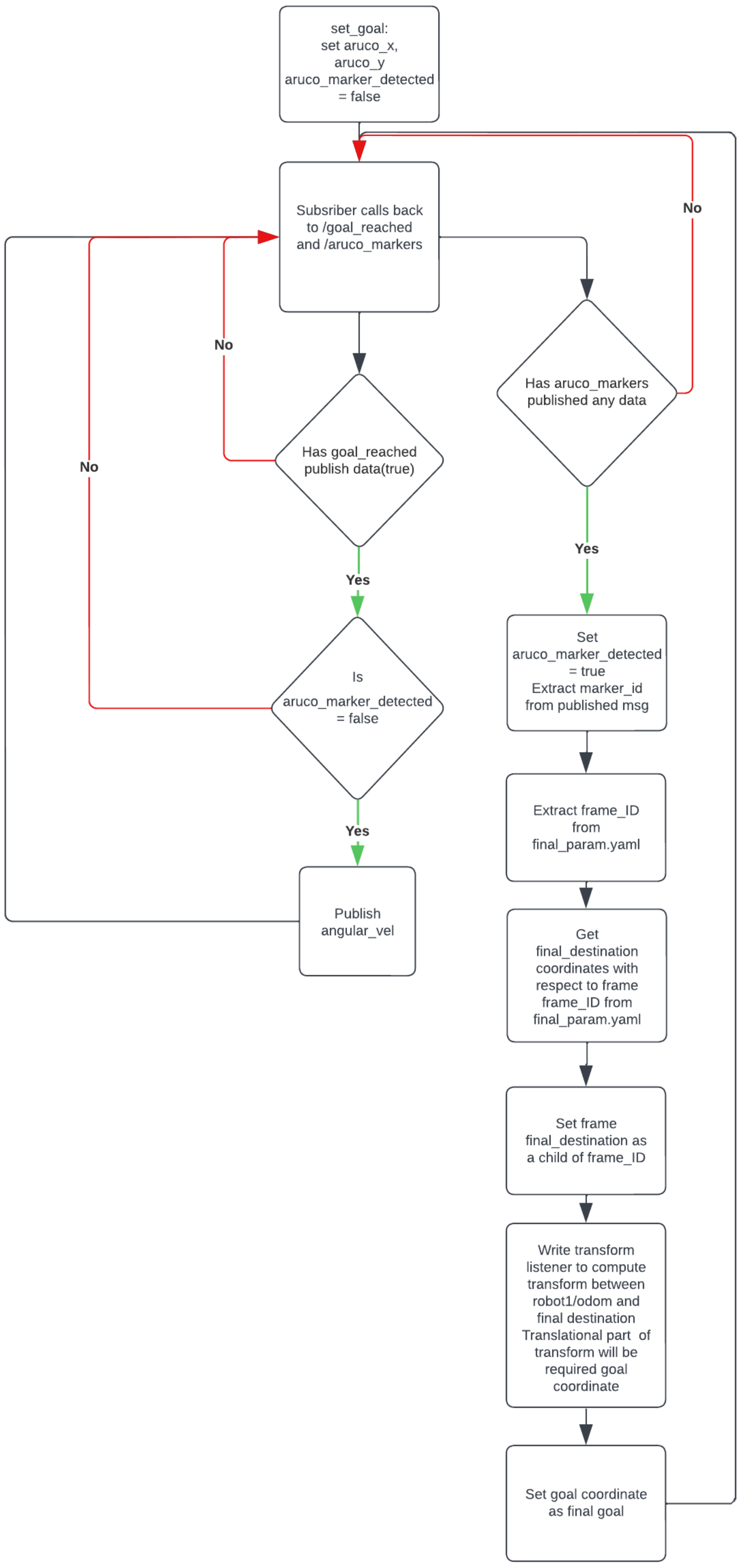

Figure 4: Flowchart for Class Target_Reacher

3 Challenges

1. Writing non static broadcaster to broadcast robot1/base_footprint as a child of /robot1/odom by subscribing to the topic /robot1/odom.
2. After reaching the final goal, the robot was still rotating about itself. This was because a constant angular velocity was given in goal reached subscriber callback after the /goal_reached topic published true (as it was needed to detect the aruco marker). To stop this we created an attribute m_aruco_marker and set its default value to be false. It is set true when aruco marker is detected. In goal reached callback, status of this attribute is checked, and in case the value is true, angular velocity is not published.

4 Contribution

Neha Madhekar: Worked on broadcaster, target reacher .
Harshal Shirsath: Worked on report, class diagram & Doxygen.
Yashas Shetty: Worked on Flowchart, report, target_reacher code.