FINAL REPORT: COFFEE DELIVERY ROBOT

Alden Daniels, Ernie Pellegrino, Pablo Ruiz

PROBLEM STATEMENT:

- Premade map of the environment.
- Move to a predefined navigation goal in map frame.
- Use environmental features (could be AR tags) to localize and position.
- Identify the clusters.
- Position arm, close the gripper, pick up the object.
- Move to a second predefined navigation goal.
- Position the gripper at the correct location for placement
- Open the gripper and avoid collision on departure.

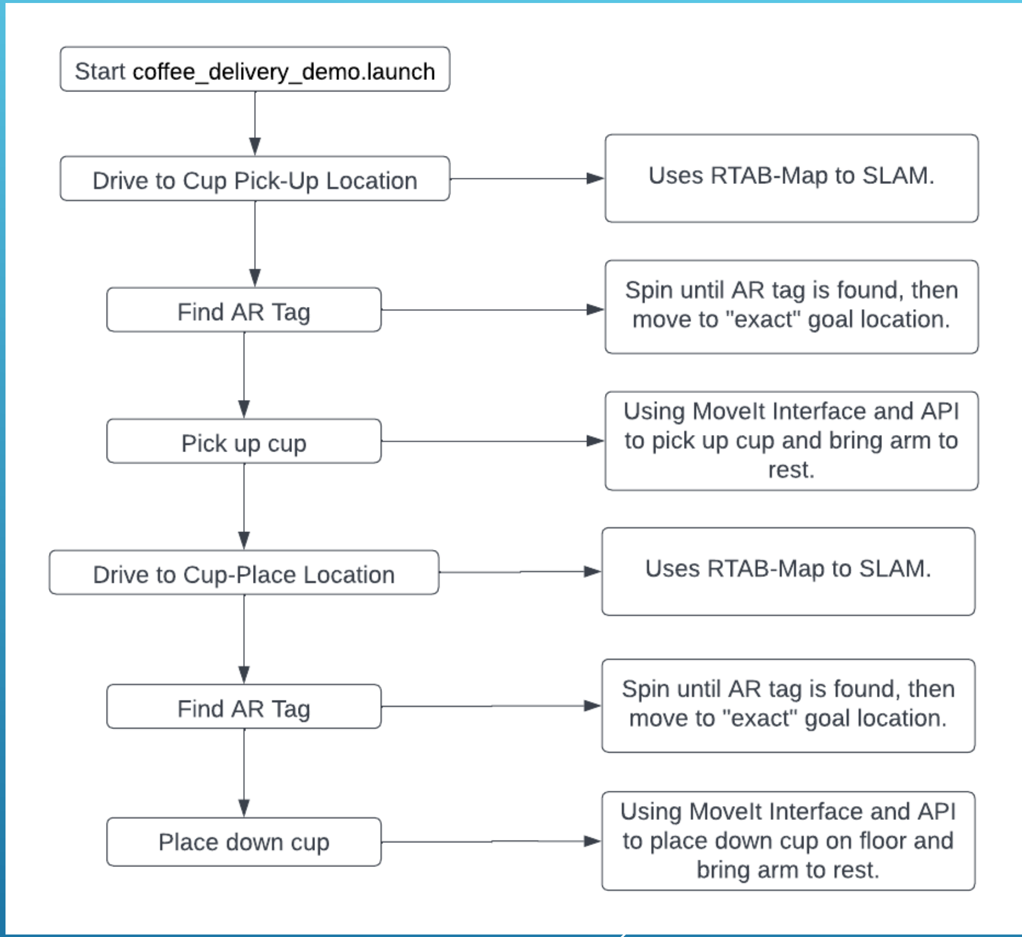

Figure 1. List of sub-tasks.

BACKGROUND:

HARDWARE

ROS SUPPORT

LAUNCH FILES

URDF

Transform Tree (TF)

MOVEMENT & MANIPULATION HARDWARE:

- Create3 iRobot base.
- WidowX 250 Robot Arm
    - 6-DOF.
    - 650mm reach.
    - 250g payload.
- 2. Custom 3D-Printed Fingers

Figure 2. Manipulation hardware.

PERCEPTION HARDWARE:

- RealSense D435 Depth Camera
  - 10m range.
  - Up to 1280 × 720 depth resolution.
- RPLIDAR A2M8 2D Laser Range Finder
  - 360° sweep.
  - 0.45° resolution.
  - 10Hz rotation.
  - 0.2m-12m range.

Figure 3. Perception hardware.

PACKAGES:

- Computer Vision:
  - OpenCV
- Manipulation:
  - MoveIt
- SLAM:
  - RTAB-Map
- Navigation:
  - Move_base
- Hardware Interface:
  - Interbotix/LocoBot Packages.
  - iRobot packages.

Figure 4. OpenCV logo.

Figure 5. MoveIt logo.

BACKGROUND:
ROS SUPPORT

- ROS2
  - Create3 base
- ROS
  - Nav Stack
  - MoveIt
  - Interbotix packages
- Python script acts as a bridge.
- Create3 base topics seen in ROS are not representative.

Figure 6. ROS logo.

BACKGROUND: LAUNCH FILES

- Best way to understand the Interbotix packages.
- Best way to modify the operation of the existing packages or replace them with better alternatives.
- Some of the arguments are loaded from YAML files.

```xml
<node if="$(arg use_rtabmapviz)"
  pkg="rtabmap_ros"
  type="rtabmapviz"
  name="rtabmapviz"
  args="$(arg rtabmapviz_args)"
  output="screen">
  <param name="subscribe_rgbd"              value="true"/>
  <param name="subscribe_scan"              value="$(arg use_lidar)"/>
  <param name="frame_id"                    value="$(arg robot_name)/base_footprint"/>
  <param name="odom_frame_id"               value="$(arg robot_name)/odom"/>
  <param name="wait_for_transform"          value="true"/>
  <remap from="scan"                        to="/$(arg robot_name)/scan"/>
</node>
```

Figure 7. Launch file example, RTAB-Map RVIZ plugin node.

BACKGROUND: URDF

- XML style document.
- Defined 3D description of robot for simulation visualization and MoveIt.
- Each component has:
  - visual mesh
  - collision mesh
  - joints
  - inertia

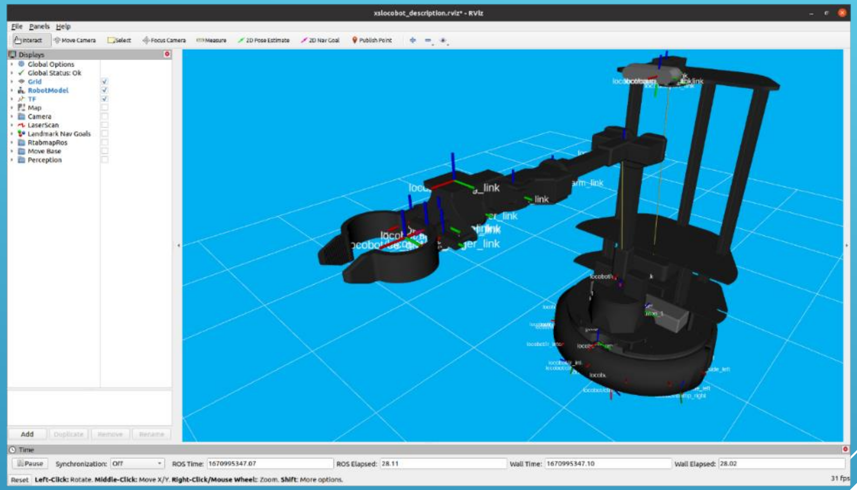

Figure 8. URDF visualized in RVIZ w/ new grippers.

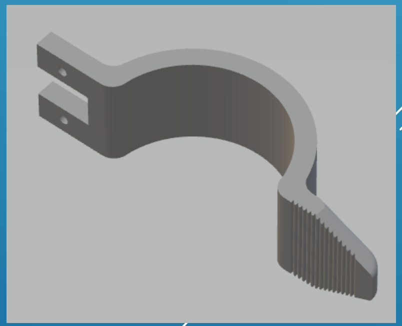

Figure 9. New gripper .stl mesh.

BACKGROUND: TRANSFORM (TF) TREES

- Provides logical of links between reference frame IDs.
- Must be fully connected starting with the map and odom frames.
- Broken tree will disable the operation of any package acting/controlling on the affected nodes.

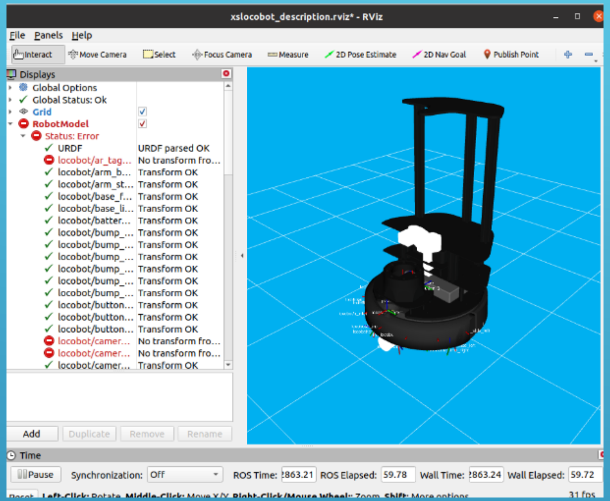

Figure 10. Broken tree in RVIZ.

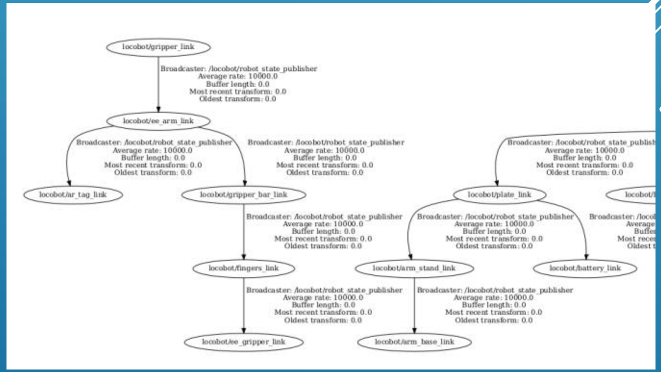

Figure 11. Broken tree using tf2_tools view_frames.py.

APPROACH:

MAPPING & LOCALIZATION

Navigation

Manipulation

APPROACH: MAPPING & LOCALIZATION

- Used RTAB-Map (Real-Time Appearance-Based Mapping)
- RTAB-Map is a RGB-D, Stereo and Lidar Graph-Based SLAM approach based on an incremental appearance-based loop closure detector.
- The loop closure detector uses a bag-of-words approach to determinate how likely a new image comes from a previous location or a new location.

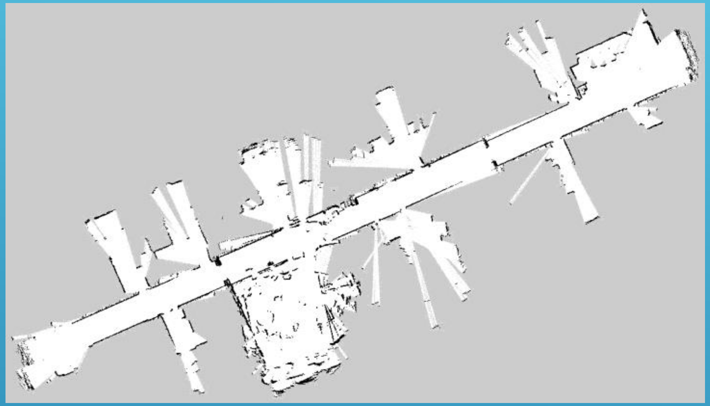

Figure 12. 2D Map of DAN407 and surroundings.

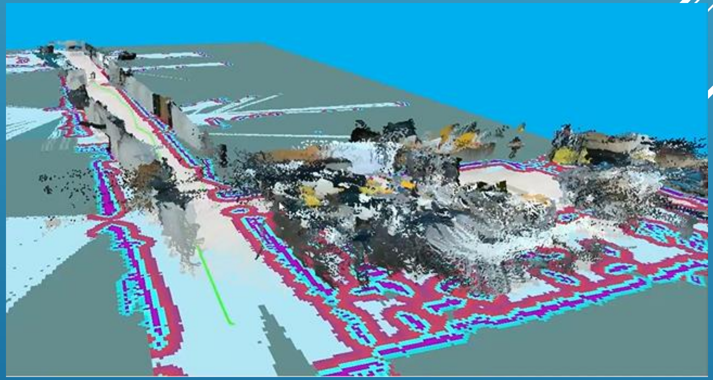

Figure 13. Example of RTAB-Map.

APPROACH: NAVIGATION

- The Interbotix packages for the Locobot utilize RTAB-Map together with Move_Base to perform navigation.
- Navigation goals can be set via RViz or by publishing to a topic and Move_Base will calculate a path and publish movement commands.
- Move_Base calculates 'cost maps' which represent obstructions in the environment and uses these for path planning. i.e.
  - walls have a cost -99
  - Free space has a cost of 0

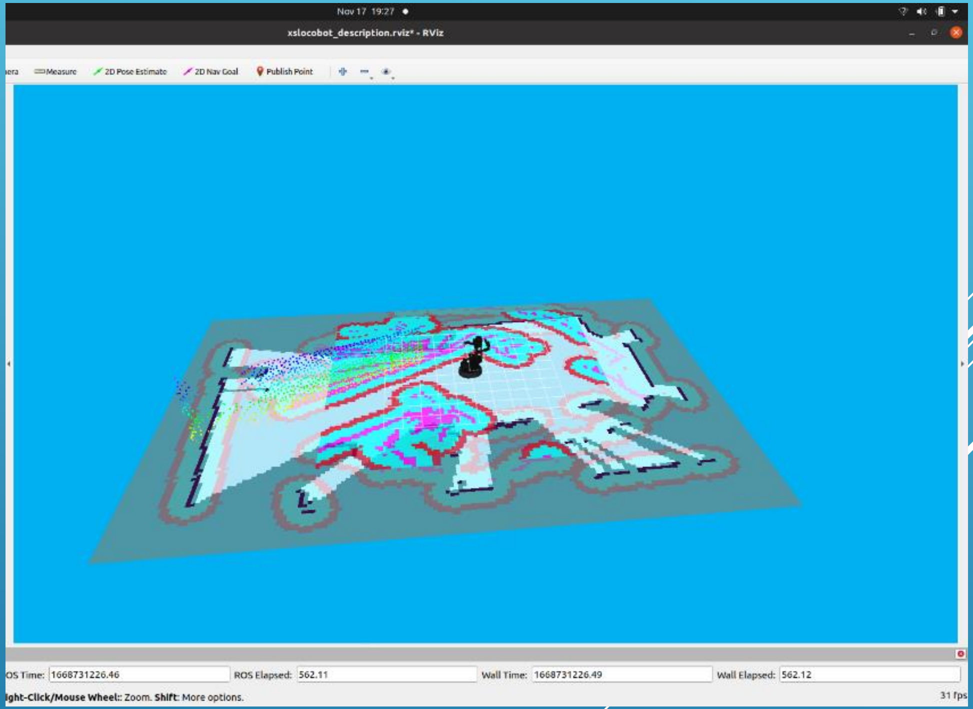

Figure 14. Cost map RVIZ view inside DAN407.

APPROACH: NAVIGATION

```
# setup move_base action server client
client = actionlib.SimpleActionClient('move_base', MoveBaseAction)
client.wait_for_server()

# setup current move_base goal
goal = MoveBaseGoal()
goal.target_pose.header.frame_id = "map"
goal.target_pose.header.stamp = rospy.Time.now()
goal.target_pose.pose.position.x = x
goal.target_pose.pose.position.y = y
goal.target_pose.pose.orientation.w = w

# send goal to move_base action server
client.send_goal(goal)

# waits until the action is complete
wait=client.wait_for_result()

# checks if a result was received successfuly
if not wait:
    rospy.logerr("Action server not available!")
    rospy.signal_shutdown("Action server not available!")
else:
    return client.get_result()
```

Figure 15. Example of a simple action client and goal message in use.

APPROACH: MANIPULATION

- Manipulation can be performed by utilizing the Interbotix API in conjunction with the perception package.
- The perception package uses the depth camera to identify objects as point clusters located in a box in front of the robot.
- A provided "Pick and Place" python script can then be used to direct the arm to 'pick' the identified objects and 'place' them nearby. (This was modified for our purposes)

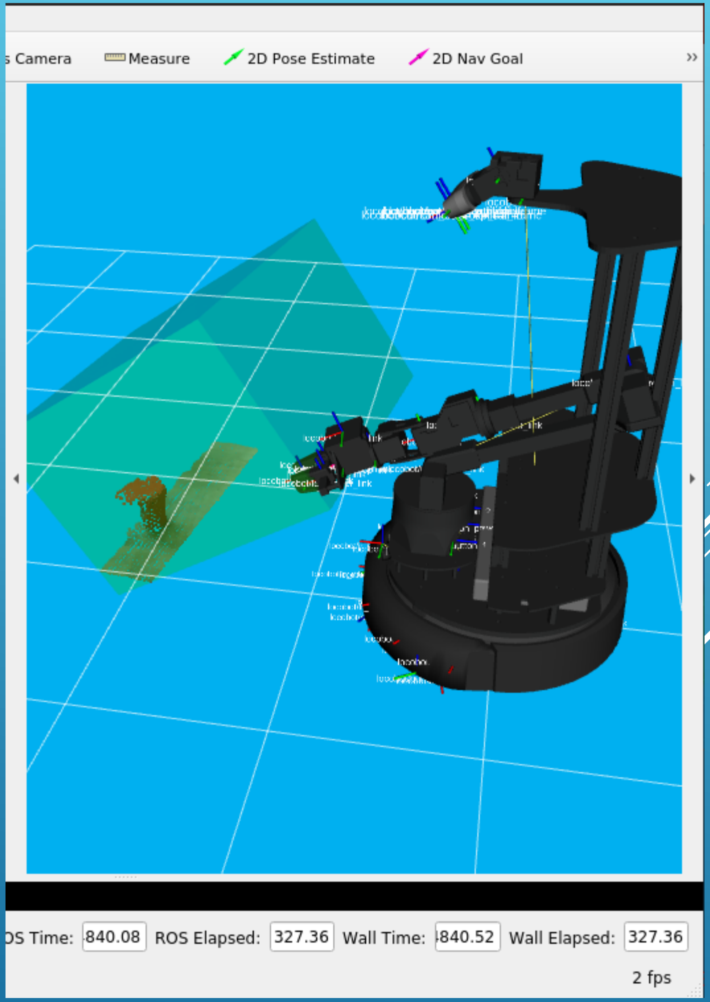

Figure 16. Perception box tuning.

APPROACH: MANIPULATION

```python
# sort the clusters such that they appear from left-to-right w.r.t. the 'locobot/arm_base_link'
success, clusters = bot.pcl.get_cluster_positions(ref_frame="locobot/arm_base_link", sort_axis="y", reverse=True)
# move the robot back so it's centered and open the gripper
bot.arm.set_ee_pose_components(x=0.3, z=0.2, moving_time=1.5)
bot.gripper.open()

# pick up each object from left-to-right and drop it in a virtual basket on the left side of the robot
for cluster in clusters:
    x, y, z = cluster["position"]
    bot.arm.set_ee_pose_components(x=x, y=y, z=z+0.05, pitch=0.5)
    bot.arm.set_ee_pose_components(x=x, y=y, z=z, pitch=0.5)
    bot.gripper.close()
```

Figure 17. Template code for grasping using Interbotix python API..

INSTRUCTIONS FOR INSTALLATION & RUNNING:

- Installer shell (.sh) script
- Remote Connection (ssh)
- Create3 setup through GUI.
- Run default or custom launch file.
- Visualize remotely through RVIZ.

INSTRUCTIONS FOR INSTALLATION & RUNNING:
BASHRC & ROS_MASTER_URI

```
export ROS_MASTER_URI=http://locobot.local:11311
source /home/locobot/interbotix_ws/devel/setup.bash
export INTERBOTIX_XSLOCOBOT_BASE_TYPE=create3
```

Figure 18. Critical Bashrc settings.

INSTRUCTIONS FOR INSTALLATION & RUNNING: CREATE3 CONFIG

- Care must be taken to ensure the Create3 base is properly configured so that it can communicate with the Locobot computer.
- This can be performed by using an HTTP interface provided by a webserver on the Create3 base itself, accessible via a browser.
- Instructions are available (now) in the Interbotix documentation: [Create3 Configuration]

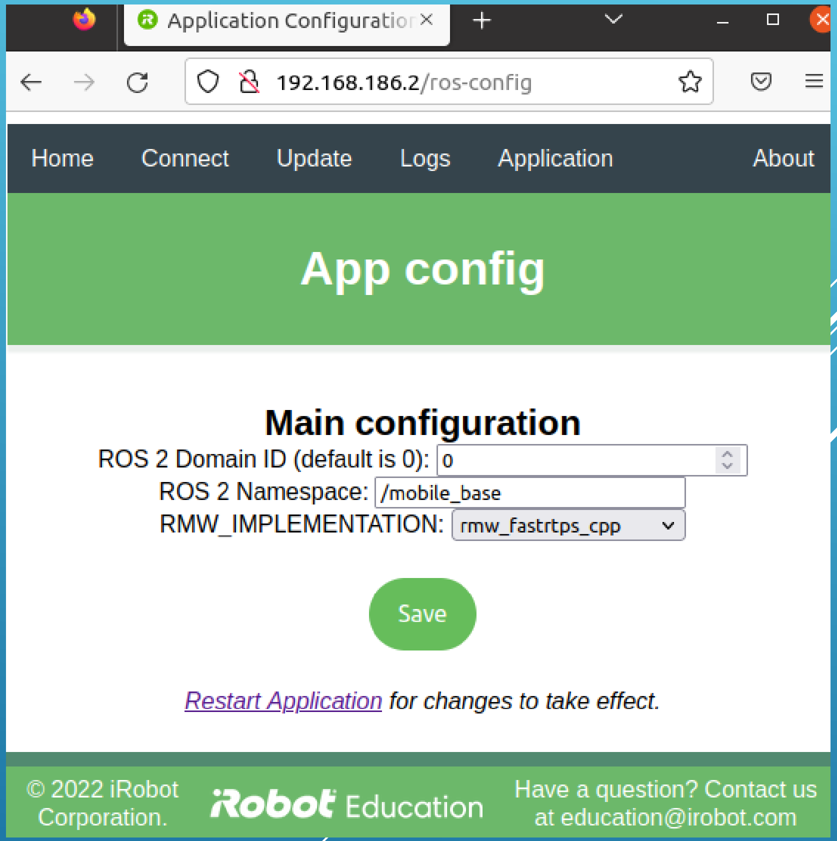

Figure 19. Create3 Config UI.

RESULTS:

- ROS launch file successfully implemented that launches Interbotix packages together with custom ROS node.
- Move_Base package used to perform navigation to sequence of preset coordinates on prior generated map.
- Pick and place routine run via python script to pick up and move a cup using the printed gripper .
- All the tasks were tested in isolation but not in sequence.

Figure 20. Locobot picking up a 4oz cup.

DISCUSSION:

- Broken Transform Tree Issue
- Problems with RTAB-Map:
  - Too much confidence in pre-existing map.
  - Heavily relies on the depth camera.
  - Sub-par localization.
- Investigating multiple methods for SLAM
- Reduce robot radius in Move_Base settings to reduce size of cost map.
- Bring all the components together, errors stack.
- Implement landmarks for error correction (AR tags).

CONCLUSION:

- Built-in SLAM functionality working.
- A launch file was successfully implemented which initiates the Locobot navigation stack
- Custom launch file and node that was implemented could be extended to both utilize the Locobot perception module and properly perform a sequence of tasks.
- Additional orientation correction could be made by introducing AR tag detection logic

DEMO VIDEOS:

Video 1. SLAM Running:
https://youtu.be/p1fKRDKZbWc

Video 2. Pick and Place Demo Adjusted for Cup:
https://youtube.com/shorts/5GdlfPOB93E?feature=share

Video 3. Robot Autonomous Moving:
https://youtu.be/vFcMZ8oWO9A

NEXT STEPS:

- Work with MoveIt Motion Planning regarding achieving our manipulation goals.
    - Pick and Place

- Connect functionality
    - Move Base
    - AR Tag Localization
    - Perception
    - Manipulation

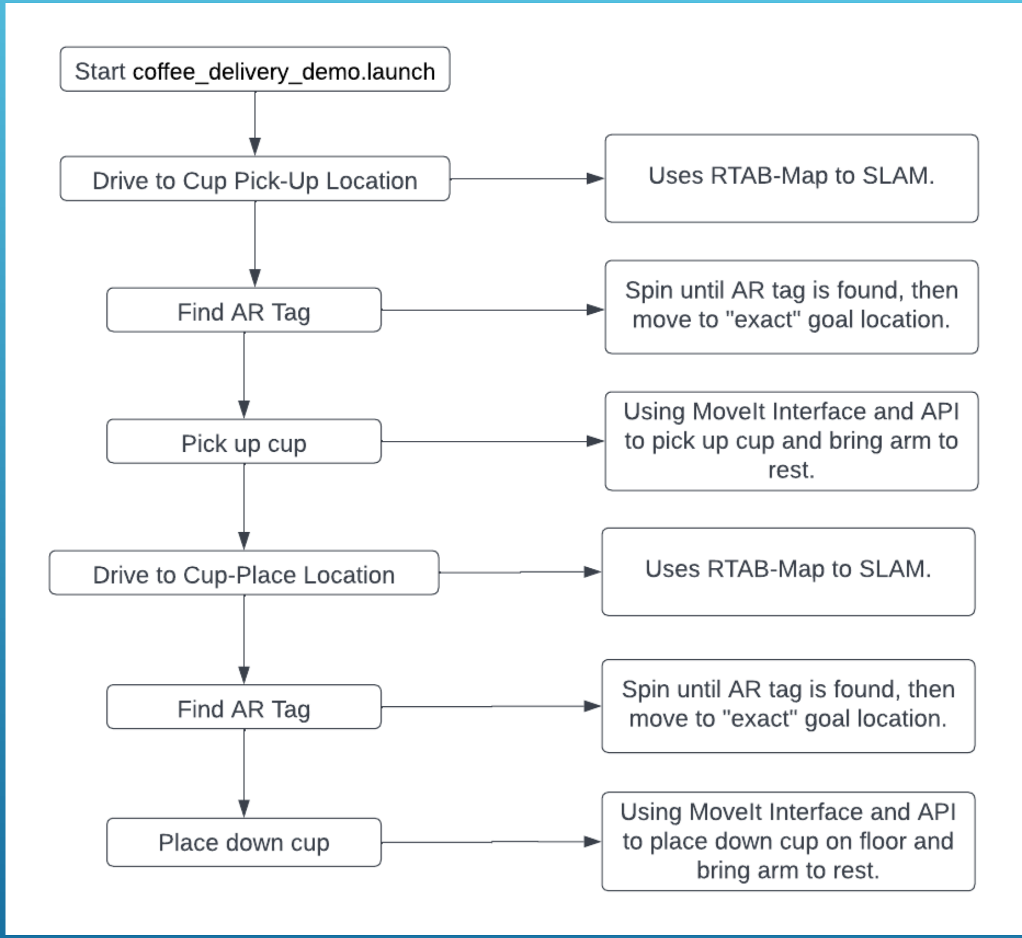

Figure 17. State Diagram.

Thank You!