Master Thesis Defence

Autonomous Navigation of Quadruped Integrated with Manipulator

Venkata Prashanth Chinthalapati

Committee Members:

Pranav Bhounsule, Ph.D., Chair & Advisor

Michael Scott, Ph.D.

Young Soo Park, Ph.D.

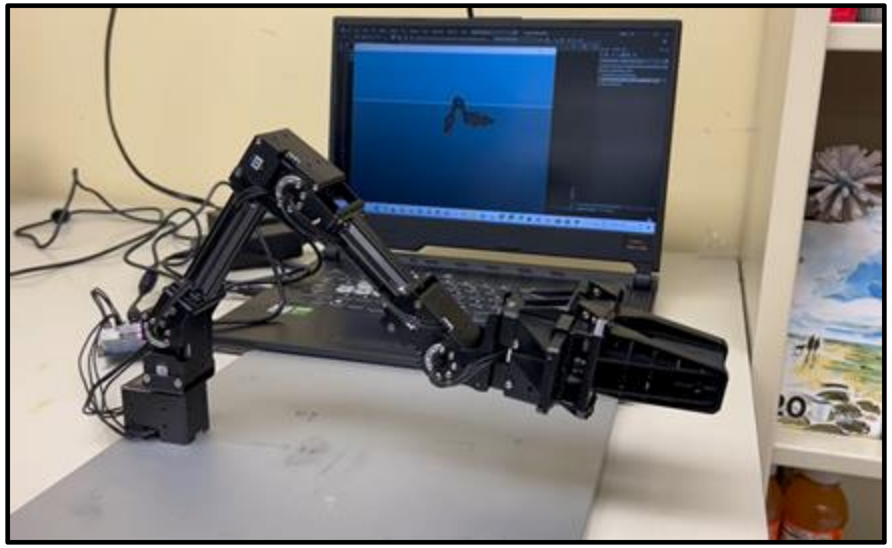
ROBOTICS AND MOTION LABORATORY

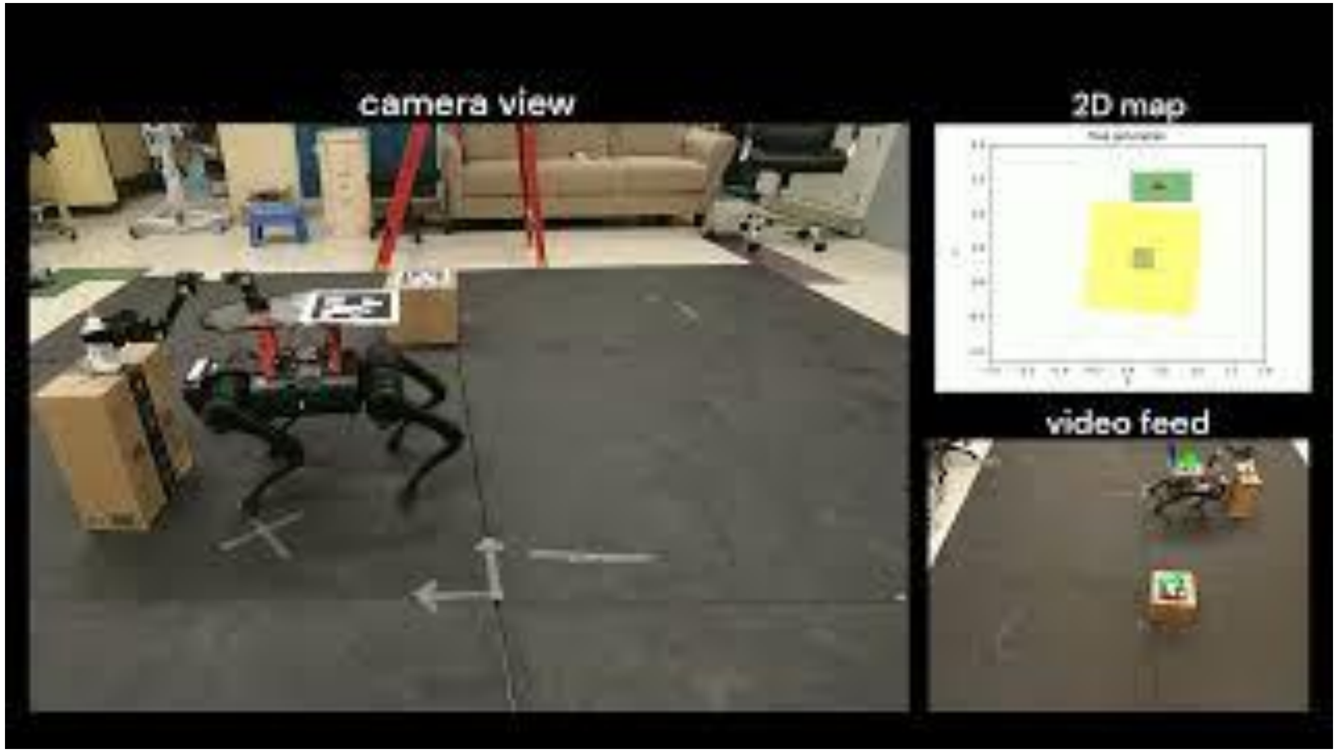
Initial Works

Open Manipulator-X
- FK/IK
- Trajectory tracking

Unitree A1 with Open Manipulator-X
- Vision-Based Navigation
- ArUco markers for Localization
- Teleoperated Manipulator

Goal
- Improve Quadruped – Manipulator Integration System
  - Obstacle detection
  - Autonomous Path Planning
  - Autonomous manipulator





Why is it useful

Logistics and Warehouse Management

- Package delivery
- Agriculture
- Industrial application

Unhabitable/Inaccessible Locations

- Fire Fighter
- First responder robots
- Machining/Welding inaccessible positions

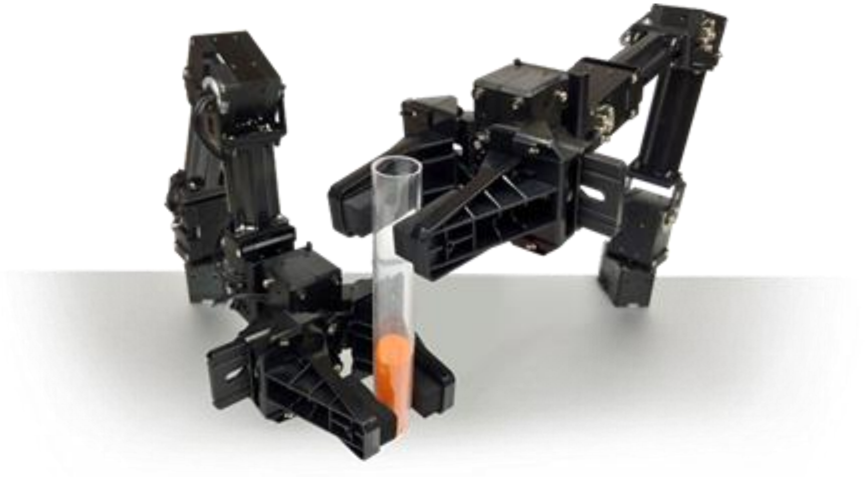
Military Applications

- Patrolling Mission points
- Carry equipment
- Bomb Disposal

Product Specifications

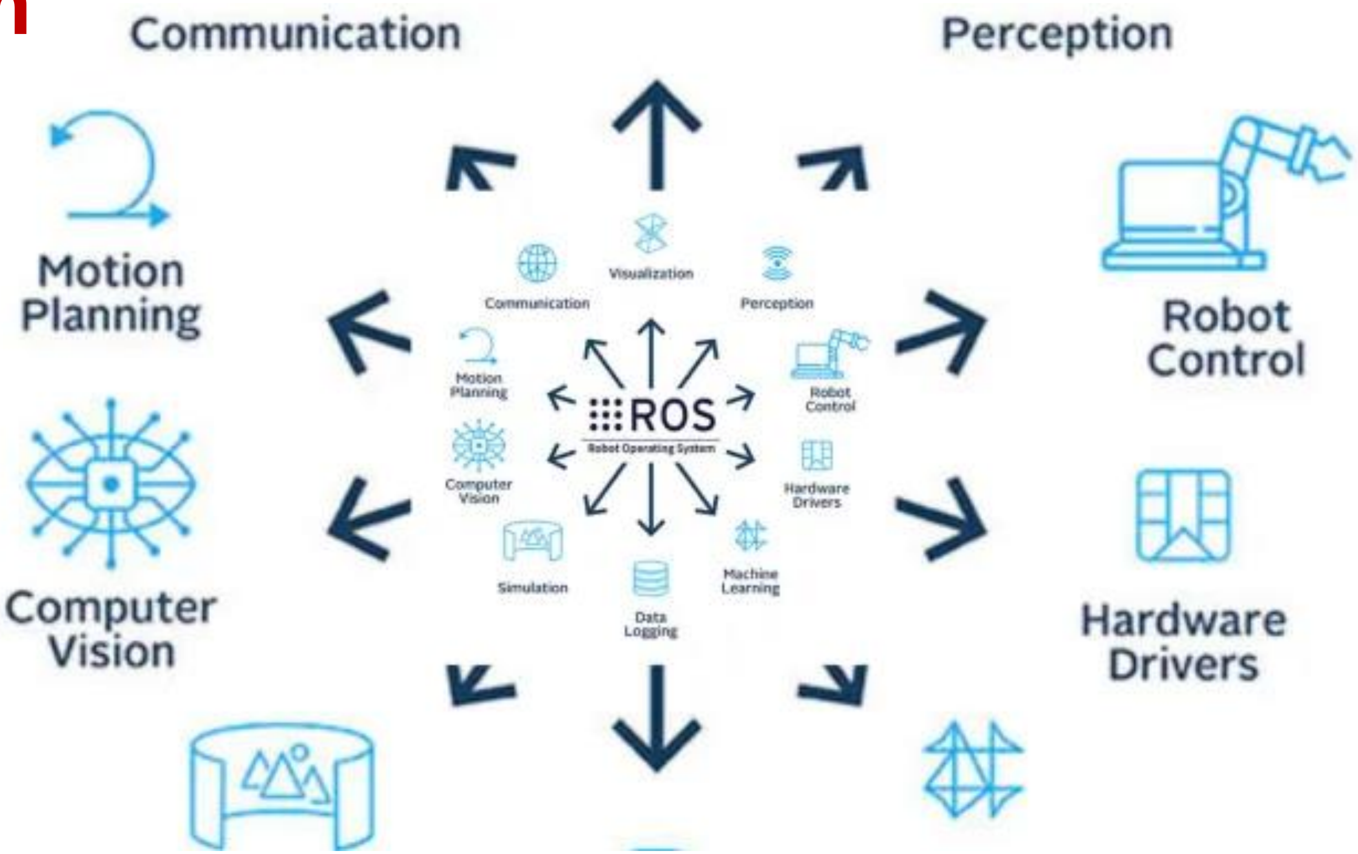
SLAMTEC Mapper M2 LiDAR		
Distance/Range	m	40
Mapping Resolution	m	0.05
Mapping area	m^2	300*300
Power	V	5
Sampling Rate	Hz	9200

Unitree Go1		
DOF		12
Depth Camera		5
Battery		Lithium Ion
Controllers		4 (3 Nano + 1 Raspberry Pi)
Power Output	V	24



Open Manipulator-X		
Actuator		Dynamixel XM430-W350-T
DOF		5 (4 DOF + 1DOF Gripper)
Reach	mm	380
Payload	g	500
Power	V	12

Robot Operating System



Why ROS?

- Modular Architecture
- Compatible with many devices
- Packages for SLAM, Path planning, and Navigation
- Simulation and Testing Environment

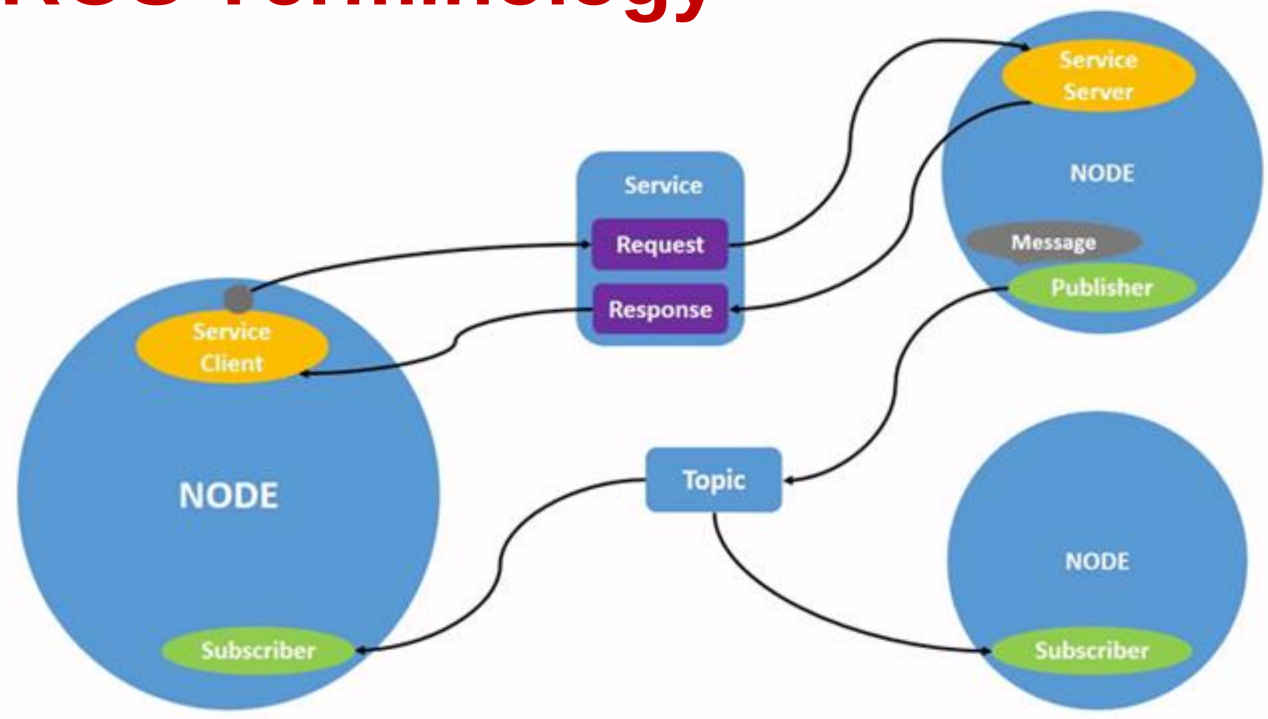
Software Installation

- Ubuntu 18.04 Virtual Machine
- ROS Melodic
- DYNAMIXEL Wizard 2.0
- RoboStudio

Ros Packages

- Unitree Go1 – https://github.com/unitreerobotics
- Open Manipulator-X – https://github.com/ROBOTIS-GIT/open_manipulator
- SLAMTEC Mapper M2 Lidar – https://www.slamtec.ai/downloads/

ROS Terminology



- **Node**: A node is an executable that uses ROS to communicate with other nodes.
- **Topics**: Nodes can publish messages to a topic as well as subscribe to a topic to receive messages.
- **Messages**: ROS data type used when subscribing or publishing to a topic.
- **Services**: Services allow nodes to send a request and receive a response

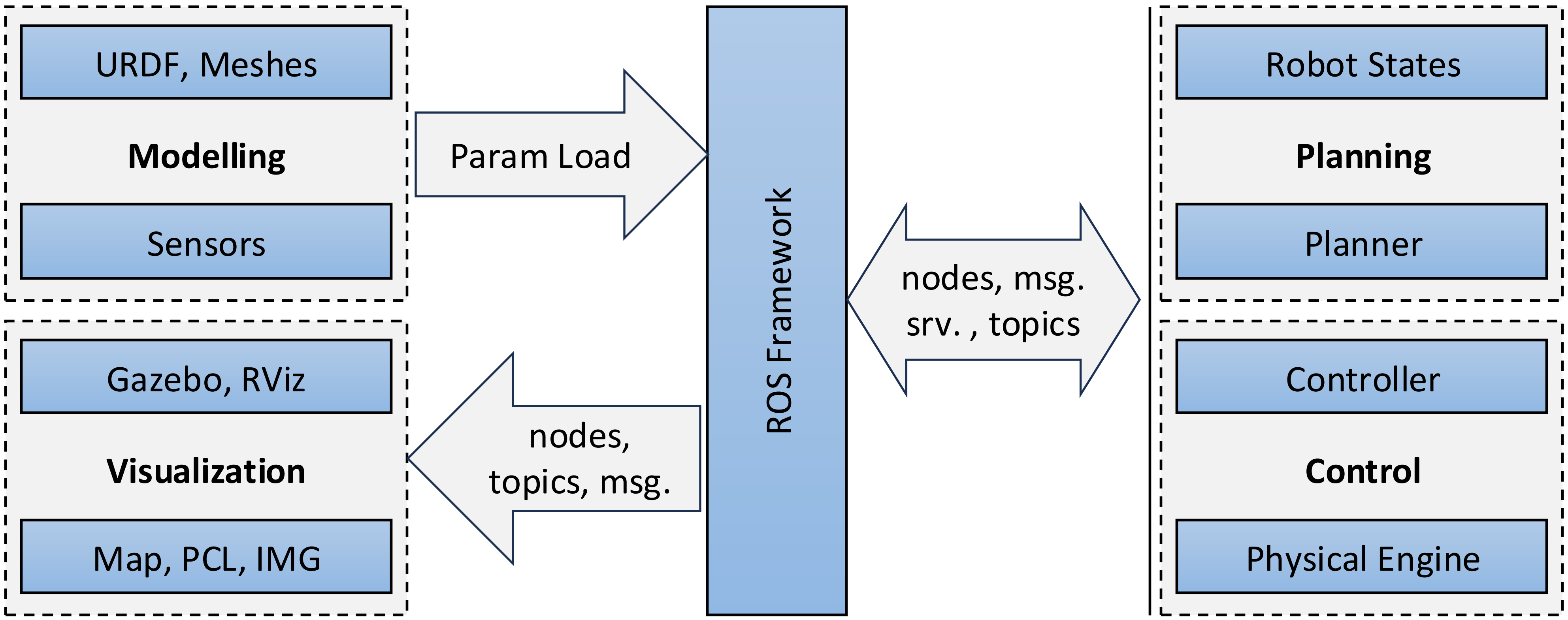
```
import rospy
from geometry_msgs.msg import Twist

def talker():
    pub = rospy.Publisher('cmd_vel', Twist, queue_size=10)
    rospy.init_node('talker', anonymous=True)
    rate = rospy.Rate(10) # 10hz

    while not rospy.is_shutdown():
        move_cmd = Twist()
        move_cmd.linear.x = 0.2
        pub.publish(move_cmd)
        rate.sleep()

if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        pass
```

Node: 'talker'

Topic: 'cmd_vel'

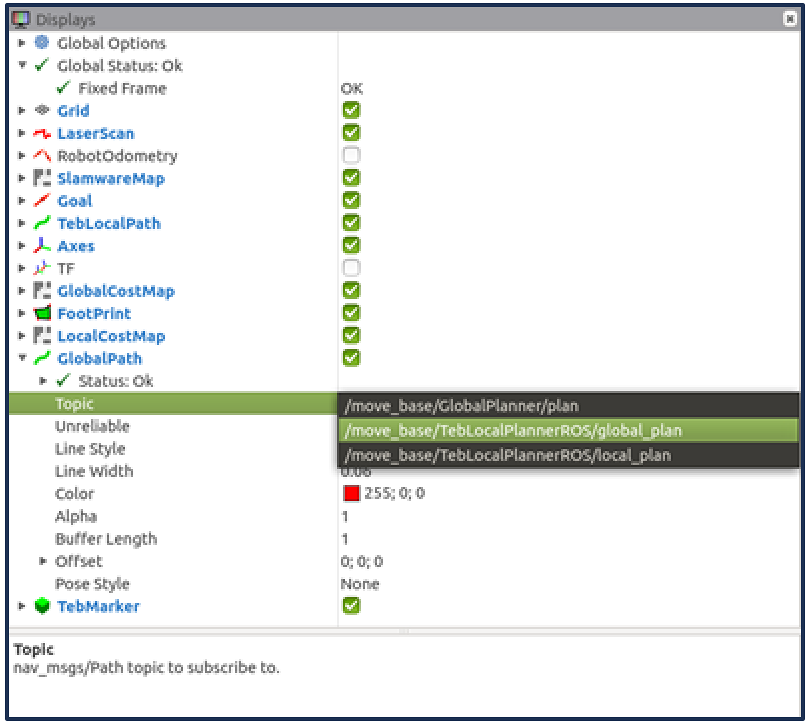
Message: linear.x =0.2

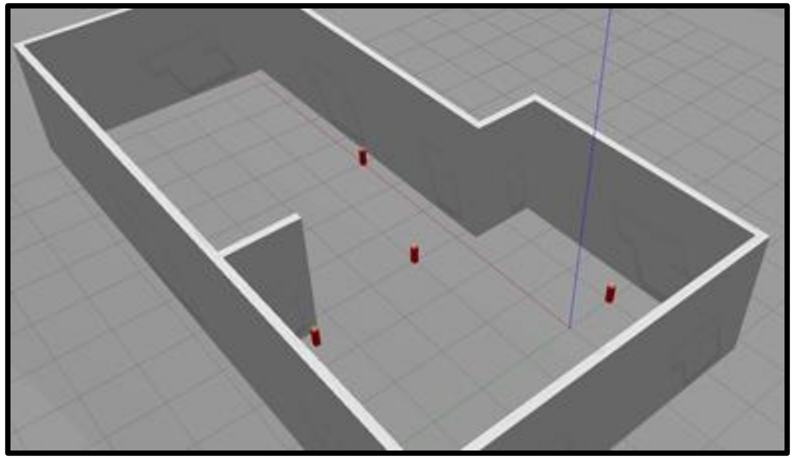
Overview - ROS Framework



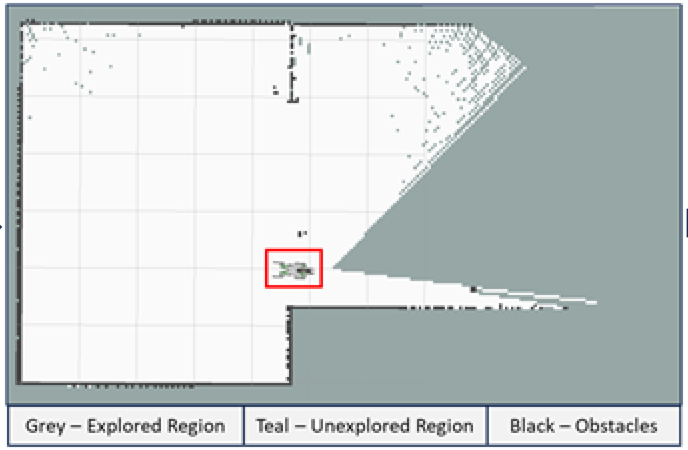
Simulation - Setup

- Gazebo - Build a Simulation World
  - Environment 1 – For Object Tracking
  - Environment 2 – For Obstacle Avoidance
- RViz
  - Map
  - Laser Scan
  - Robot States

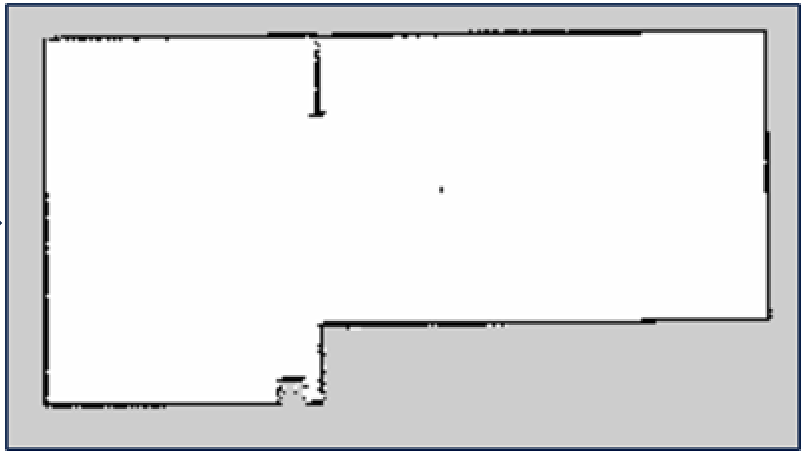




Gazebo Environment 1



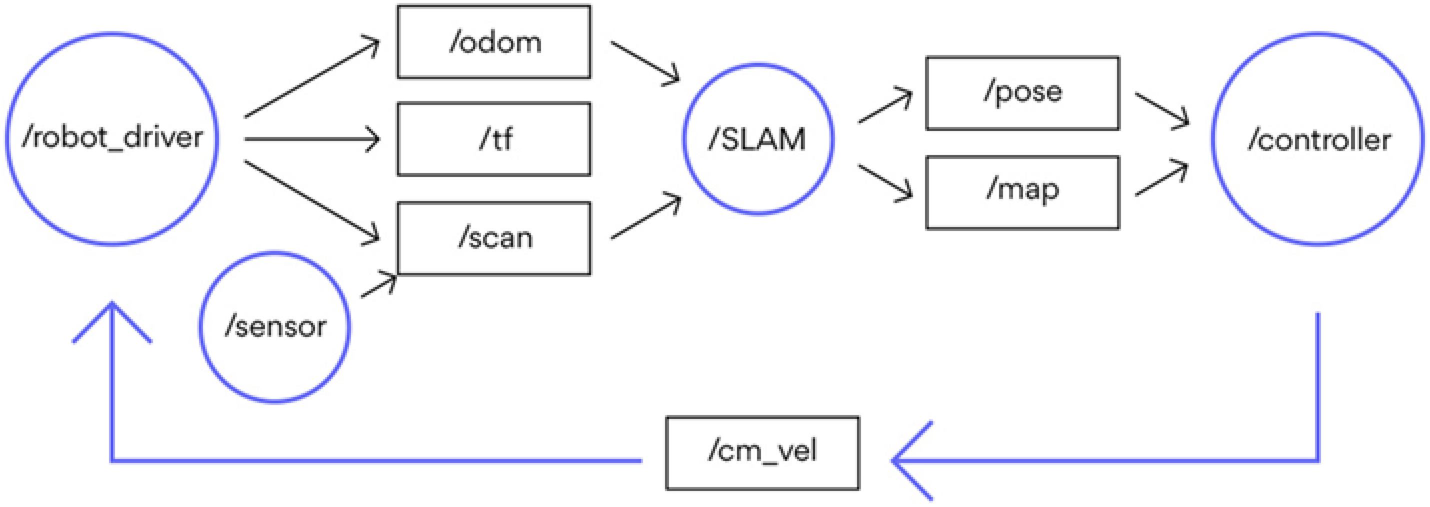
Gmapping View in RViz



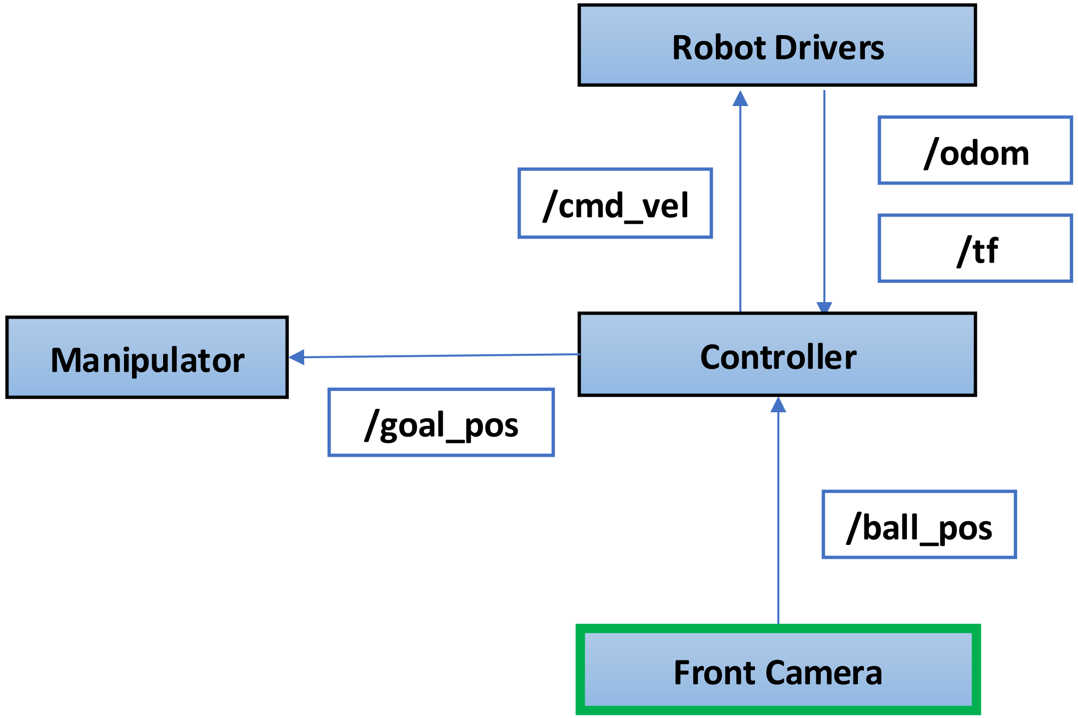
Map of Environment 1

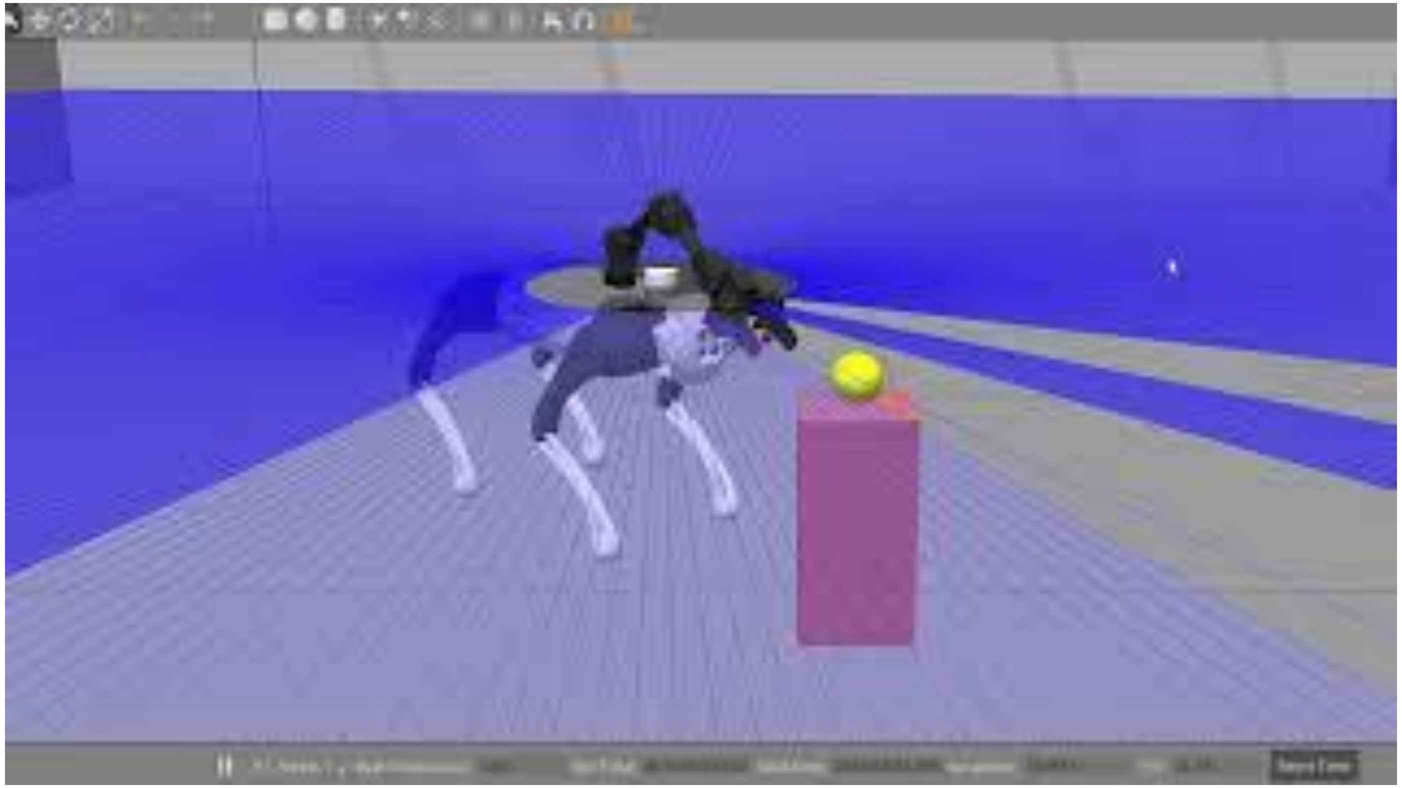
Overview - Navigation Stack

- **robot_driver** publishes odometry and transform data, and receives velocity commands.
- **sensor** sends laser scan data.
- **SLAM** uses odometry, transforms, and scan data to create a map and estimate the robot's pose.
- **controller** plans the path using the map and pose, then sends velocity commands to the robot_driver.



Object Tracking





- Only tracks visible objects
- Object not reachable
- Path planning required

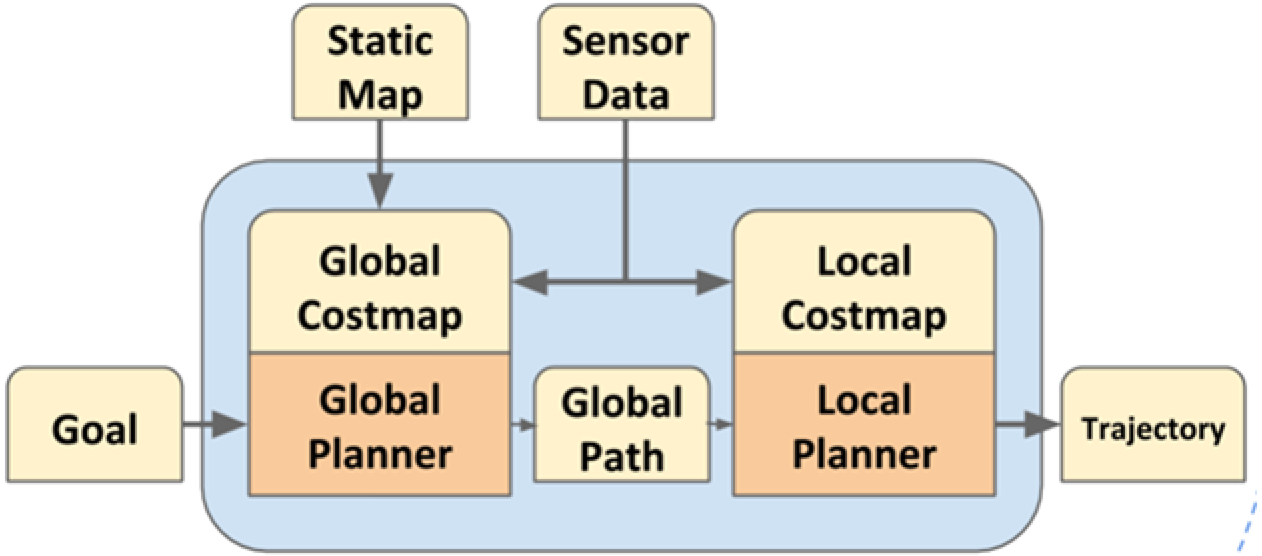
Global / Local Planners

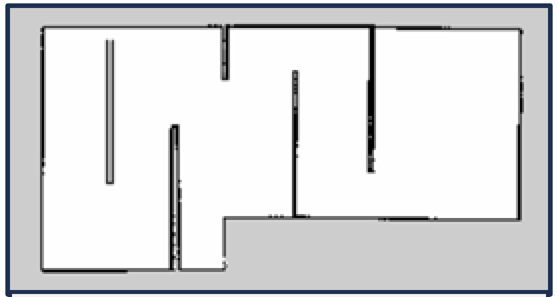
•Global Planner

- Creates a long-term path from the robot's current position to the goal.
- Uses algorithms like Dijkstra's or A* to find the shortest, collision-free path.

•Local Planner

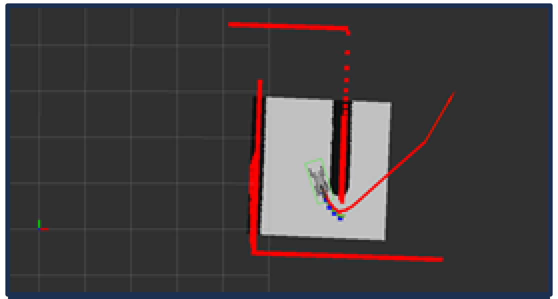
- Generates a local trajectory that the robot should follow in the short term.
- Uses algorithms like DWA or TEB continuously adjusts the path based on real-time sensor data to avoid obstacles.





Environment 2 - Static Map

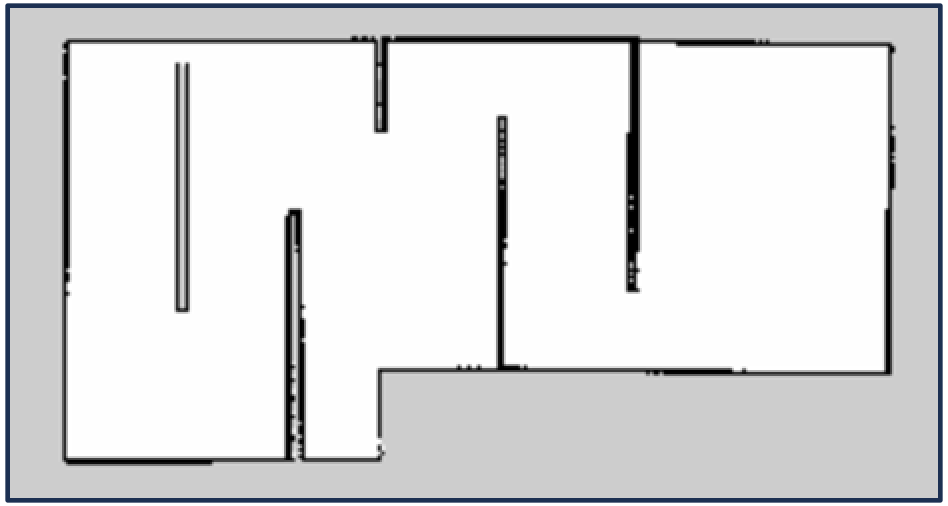
Global Costmap

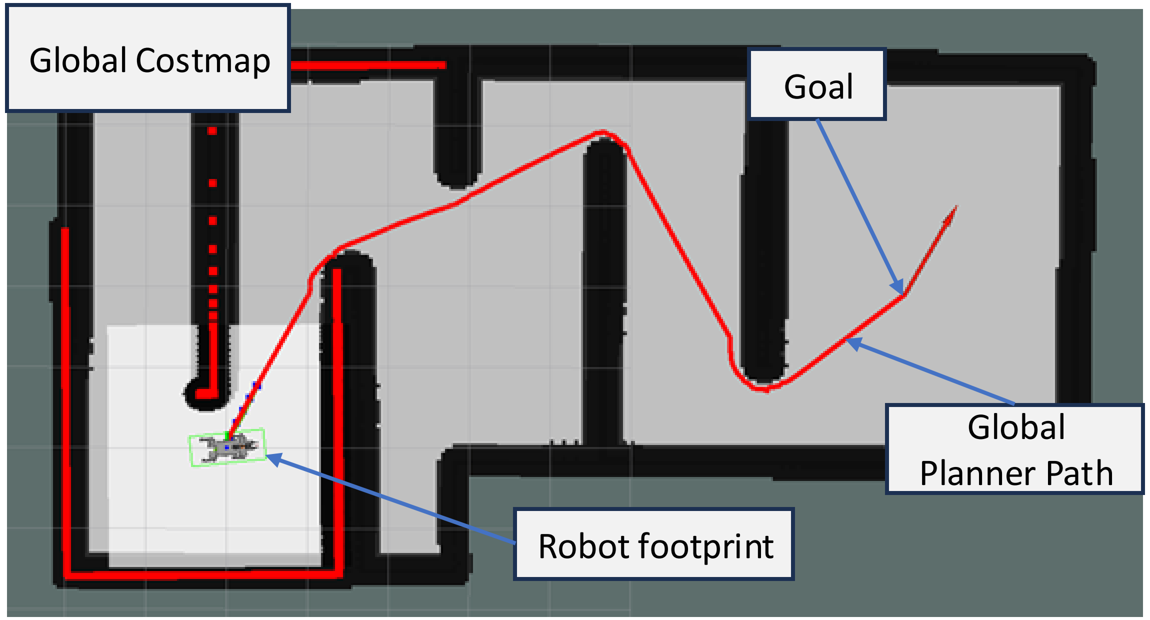


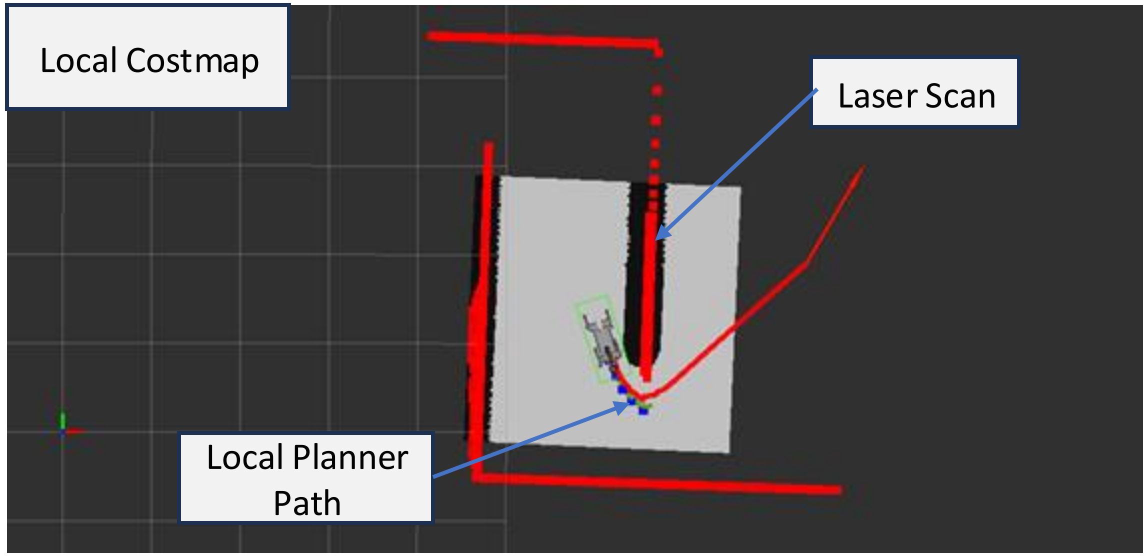
Local Costmap

Local / Global Costmaps

- **Static Map**
  - publishes Static Obstacles
- **Global Costmap**
  - Considers inflation around obstacles
- **Local Costmap**
  - Considers robot heading and base footprint







Path Planning Algorithm Cost function

Parameters	Value	Units
Global Planner	Dijkstra	
Local Planner	TEB	
max_vel x,y	0.3	m/sec
max_vel_theta	0.2	rad/sec
acc_lim x,y	0.2	m2/sec
acc_lim_theta	0.2	rad/sec
xy_goal tolerance	0.05	m
yaw_goal tolerance	0.1	m
min_obstacle dist	0.25	m
Inflation radius	0.25	m
resolution	0.025	m
footprint	0.45*0.175	m2

•TEB Local Planner

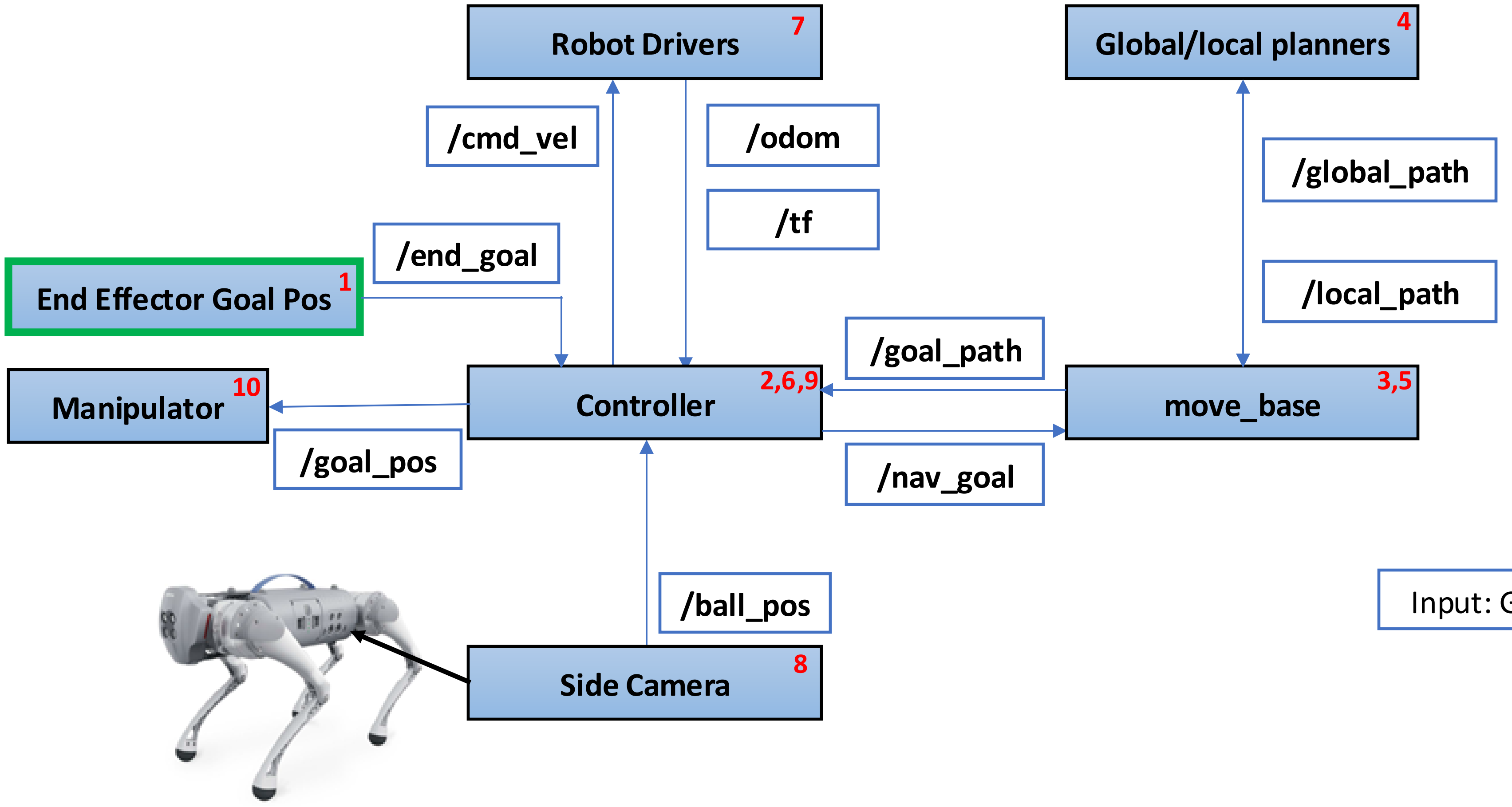
- Optimize:
  - Distance between the robot and the obstacle
  - length of the path
  - execution time of the trajectory
- Optimizing Variables
  - Robot Pose $Q = \{X_i\} \ i = 0,1,2 \cdots n, n \epsilon \mathbb{N}$
  - Time interval $\tau = \{\Delta T_i\} \ i = 0,1,2 \cdots n-1, n \epsilon \mathbb{N}$

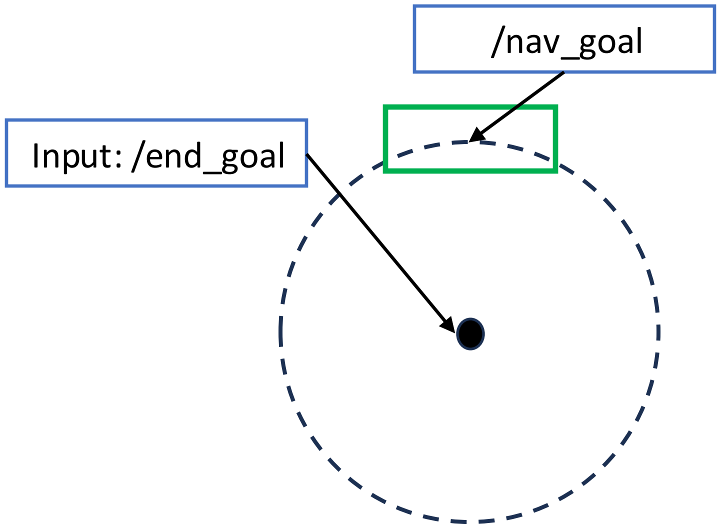
Define the state set $B(Q, \tau)$:

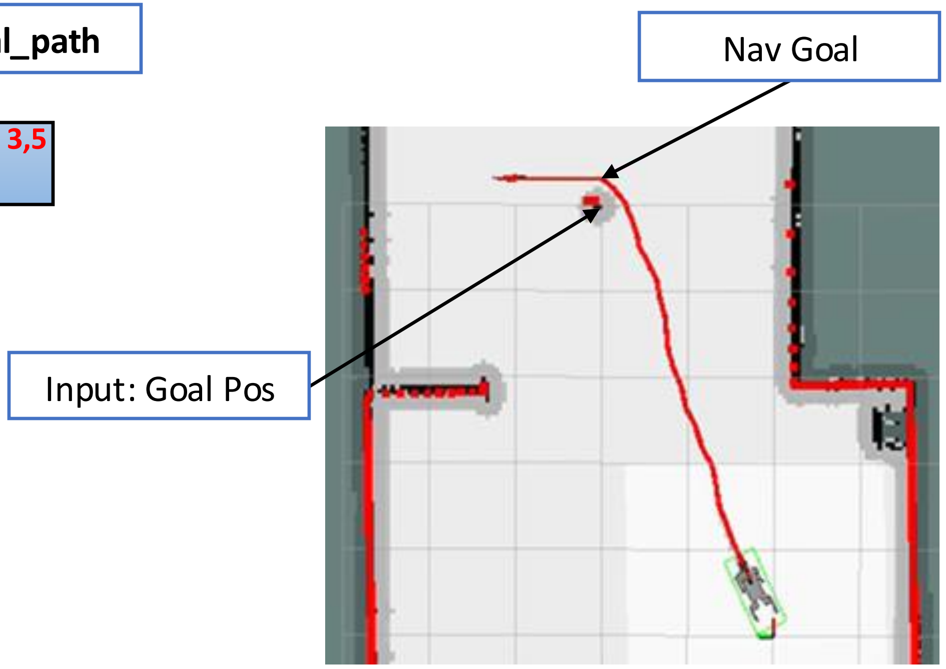
$$f(B) = \sum_k \gamma_k f_k(B)$$

$$B^* = \underset{B}{argmin} f(B)$$

Path Planning Algorithm







Path Planning Environment 1

- Input: End Effector Position
  - X = 5
  - Y = 2

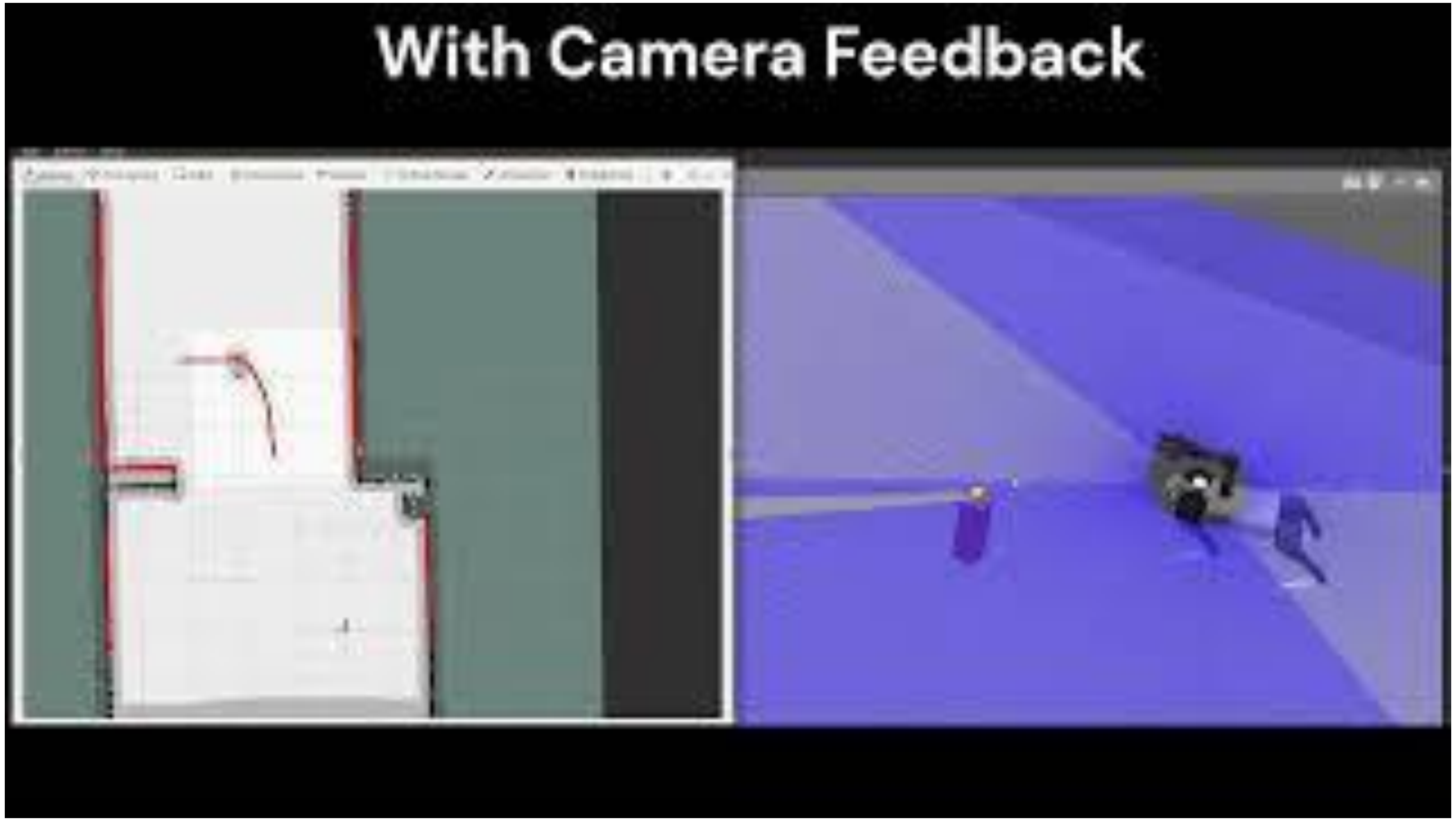
•Without Camera

- Less accuracy
- Less Processing
- Stable system

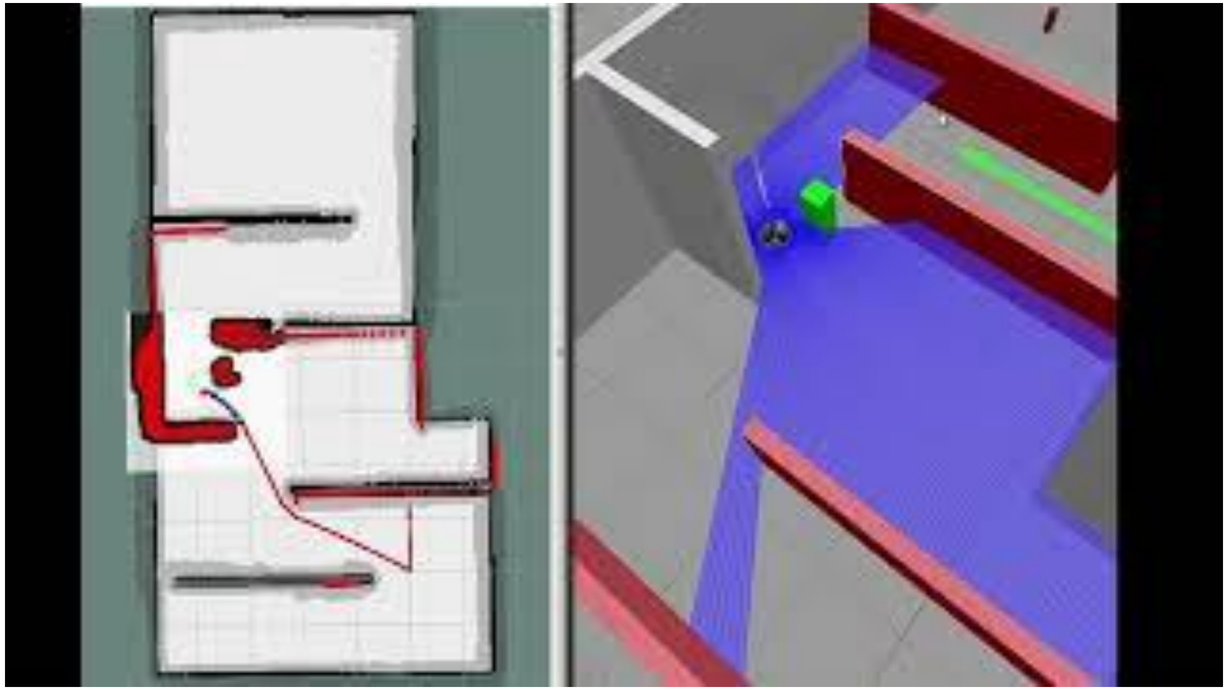
•With Camera

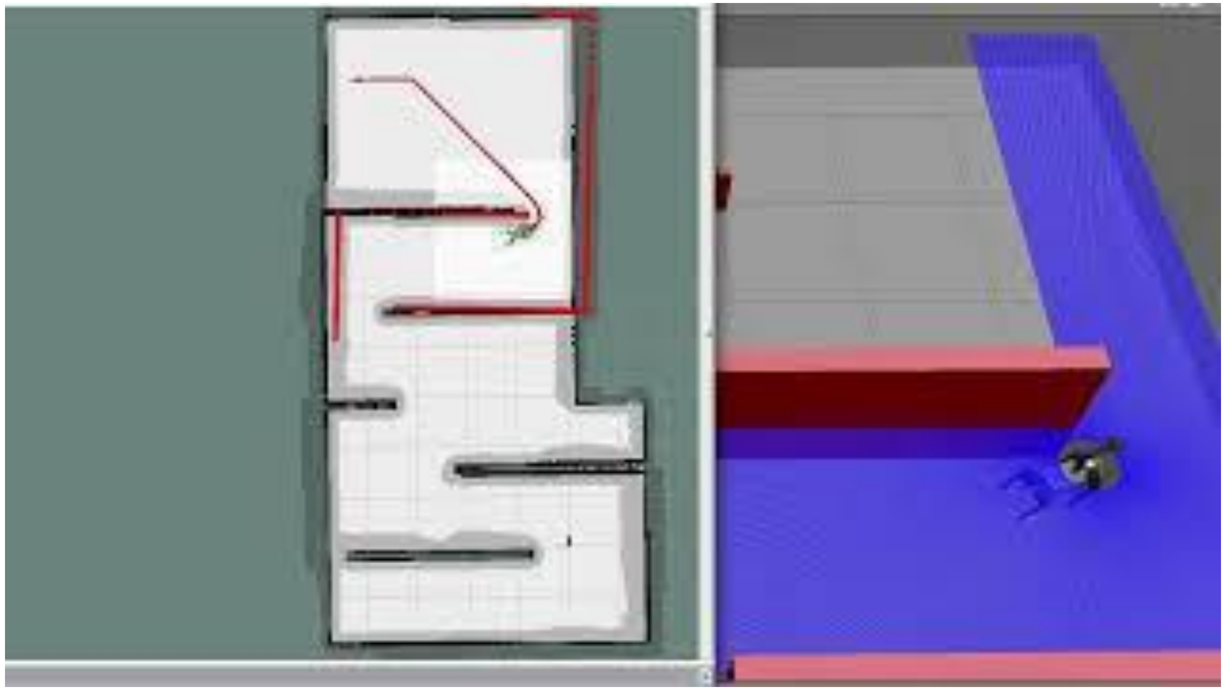
- More accuracy
- More processing
- Less stable system

Distance/ Range	Without Feedback		With Camera Feedback	
	X	Y	X	Y
Mean	3.666	5.866	4.884	2.534
Stand. Deviation	44.46	25.80	19.869	14.697



Path Planning Environment 2

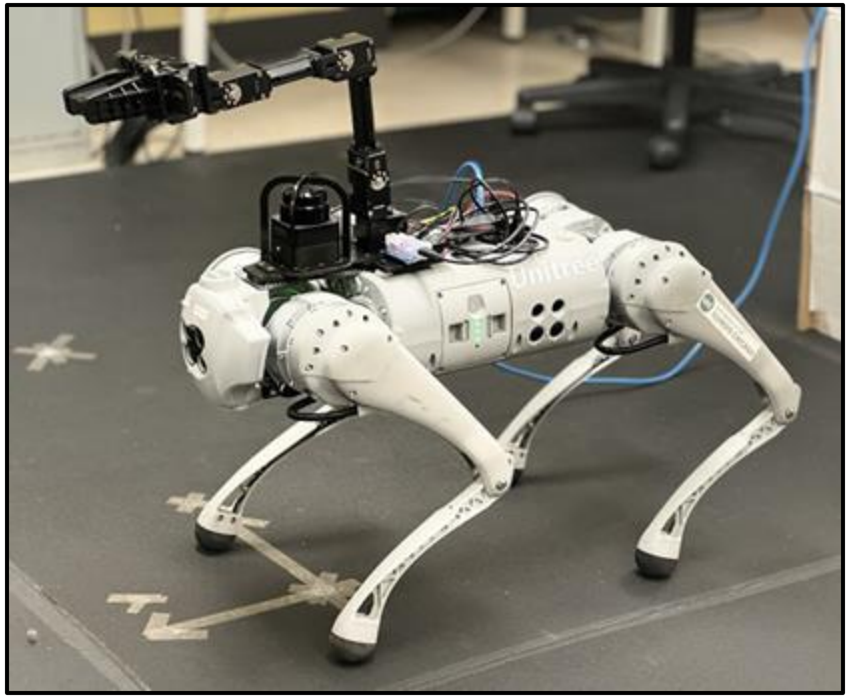


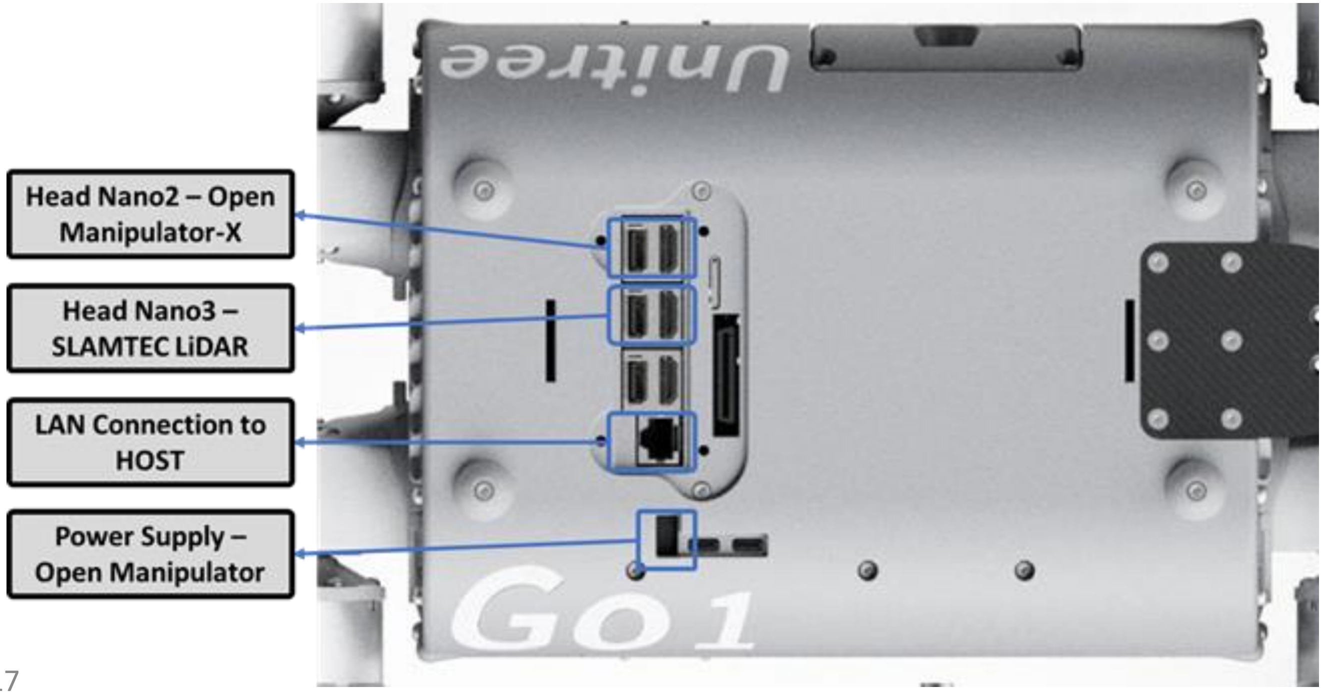


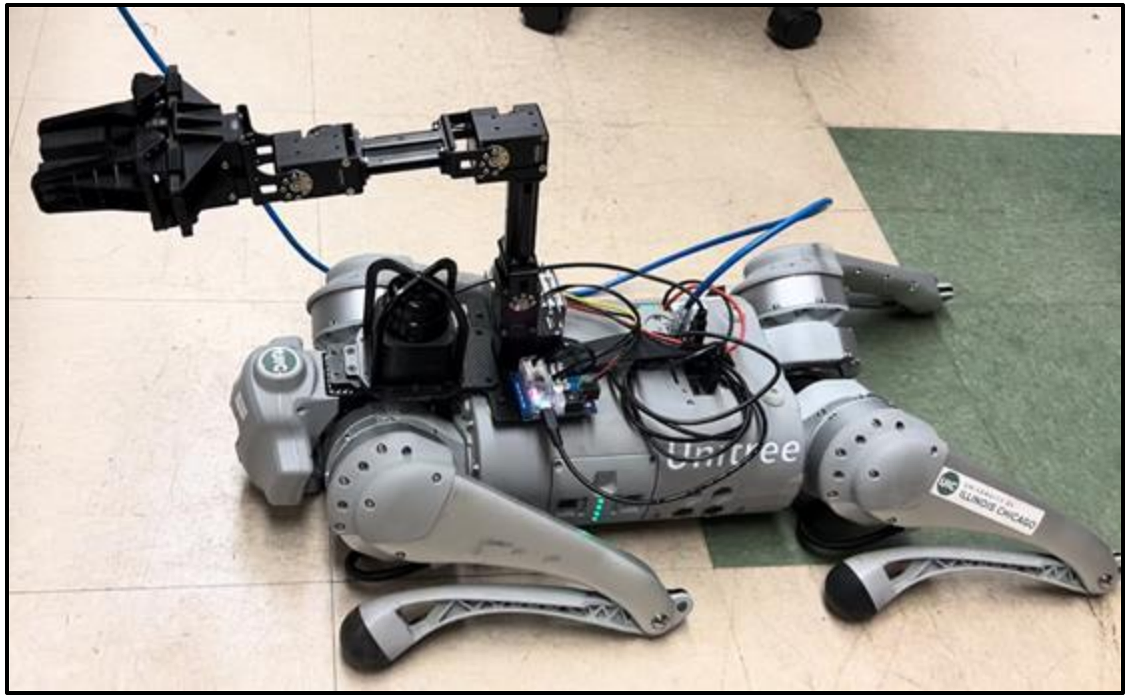
- Blue Rays – LiDAR
- Green Block – Obstacle
- Left – RViz
- Right – Gazebo world

Hardware Mechanical Setup

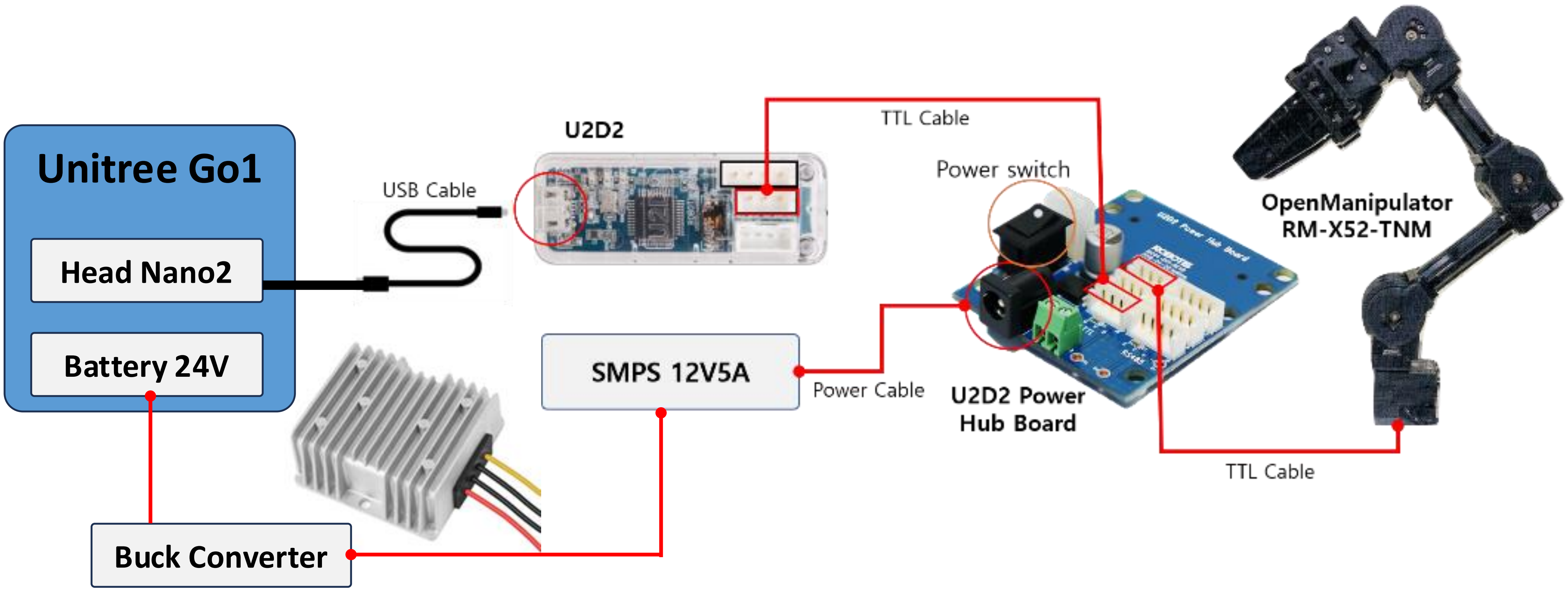
- **Open Manipulator-X, LiDAR, Manipulator controller, Buck Converter supported on a 3D printed base plate.**



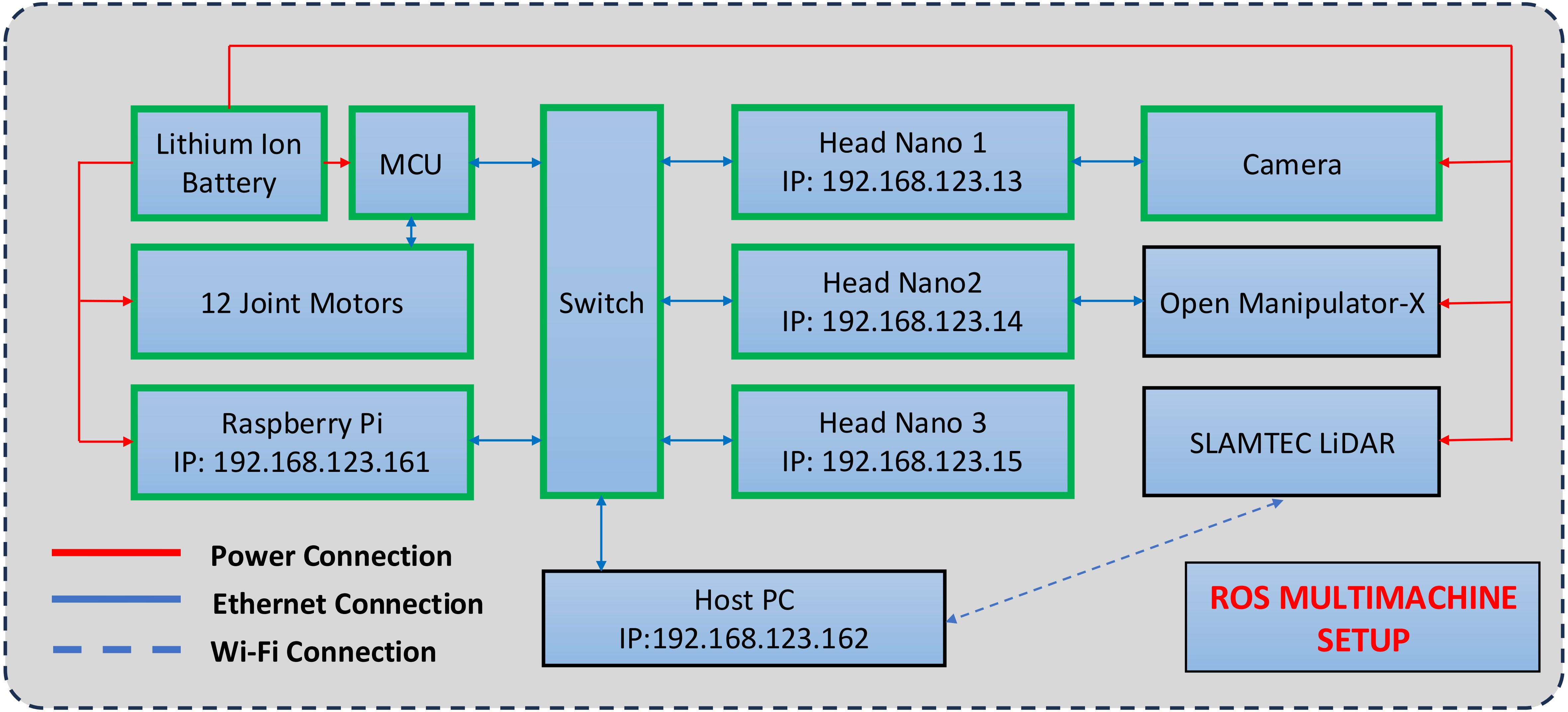




Open Manipulator - Communication



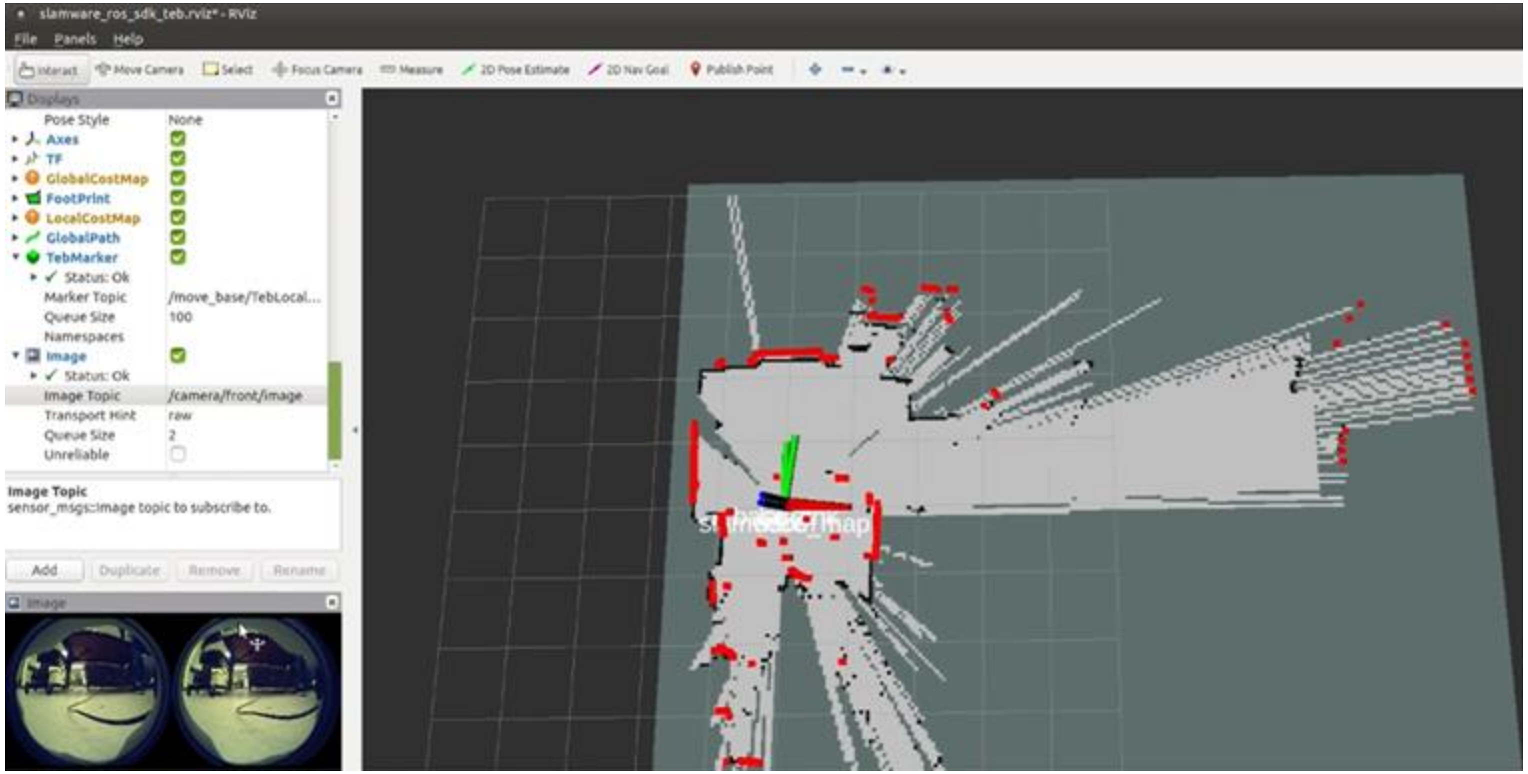
Hardware Network Setup



Hardware Camera Setup

•Head Nano 1

- Fish-eye view
- Latency issues

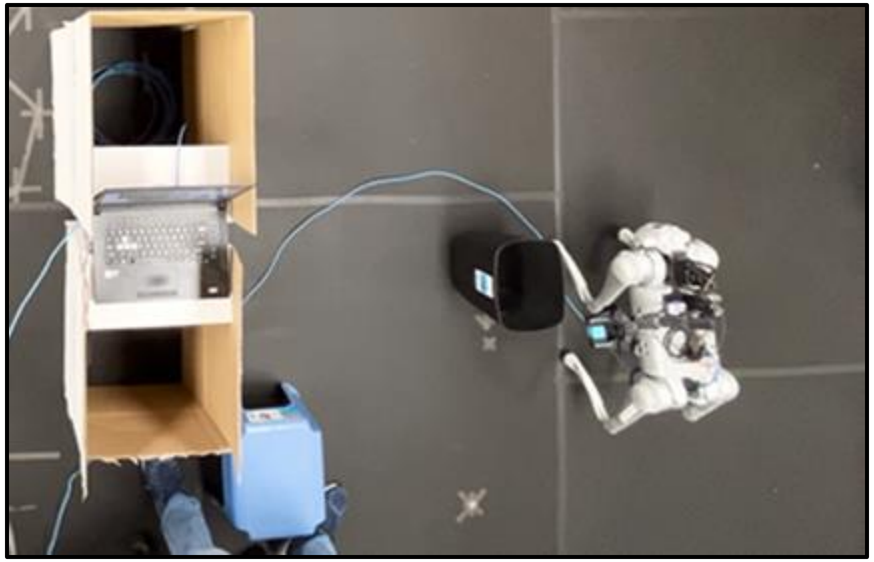


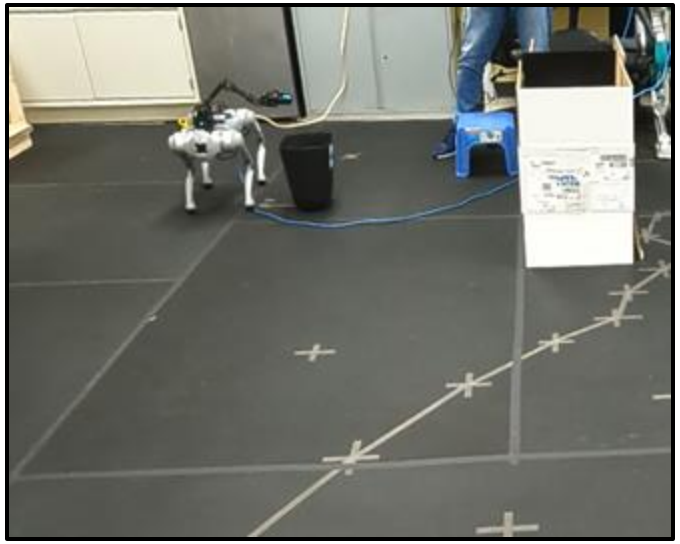
Hardware Testing

Hardware Testing

•Repeatability

- Tests Performed = 15
- Successful attempts = 11





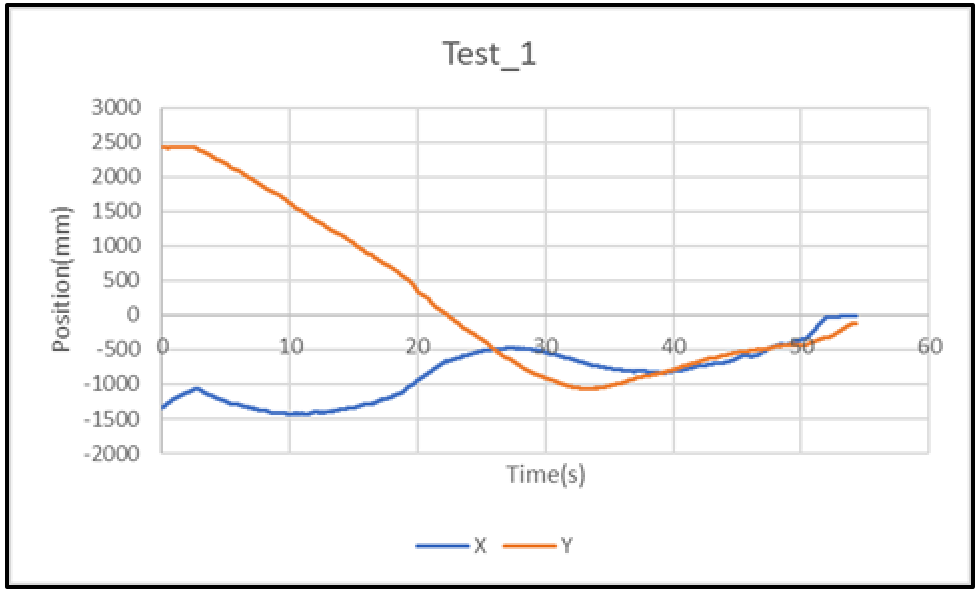
Length = 270

Width = 200

Hardware Results

- **Mocap-Map**
  - X_transform = 2.35
  - Y_transform = -116.87

- **Accuracy**
  - Mean_X = 4.56
  - Mean_Y = -35.97



	Mocap_X	Mocap_Y	Map_X	Map_Y
Test 1	-10.35	-127.56	-12.7	-10.69
Test 2	11.77	-128.25	9.42	-11.38
Test 3	19.31	-158.58	16.96	-41.71
Mean Error			4.56	-21.97
Std Dev.			12.58	14.46

*All Dimensions in mm

Conclusion

- Improve Quadruped – Manipulator Integration System
  - Obstacle detection
  - Autonomous Path Planning
  - Autonomous manipulator

- Depth Camera for object tracking

References

- https://github.com/ROBOTIS-GIT/open_manipulator
- https://www.slamtec.ai/downloads/
- https://github.com/unitreerobotics/unitree_ros.
- Chitta et al. ros_control: A generic and simple control framework for ros. The Journal of Open Source Software, 2017. doi:10.21105/joss.00456
- Thushara Sandakalum and Marcelo H. Ang. Motion planning for mobile manipulators—a systematic review. Machines, 10, 2 2022. doi:10.3390/MACHINES10020097.
- Deep Robotics. Jueying x20, 2022. Accessed on Dec. 28, 2022. URL: https://www.deeprobotics.cn/en/products.html.

Questions?

Thank you!

Appendix