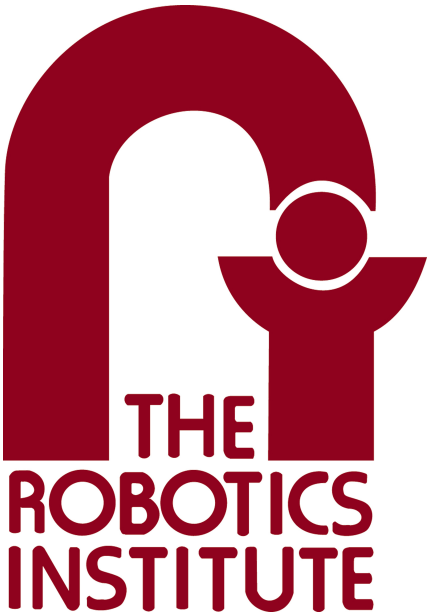

MRSD Project Course

Team I – AIce

Autonomous Zamboni Convoy

Individual Lab Report 2

Team

Rathin Shah

Nick Carcione

Yilin Cai

Jiayi Qiu

Kelvin Shen

Author

Rathin Shah

Feb 16, 2022

Contents

1. Individual Progress 2
1.1. Zamboni Localization: 2
1.2. Pure Pursuit Controller: Validation and Integration with Zamboni Simulator 4
1.3. PCB Schematic design: 5
2. Challenges 6
2.1. Zamboni Localization 6
2.2. Pure Pursuit Controller Integration 6
3. Teamwork 7
4. Future Work/Plans 8

1. Individual Progress

Since the last Progress Review, I progressed on the Localization of the robot and was able to validate its performance. Also, another subsystem that I finished was integrating the Pure Pursuit Controller developed in Simulink with the simulation setup in gazebo. I also worked on making the schematic for Power Distribution Board-PCB (MRSD Project Course 1 Task 6). The detail of each progress is listed below.

1.1. Zamboni Localization:

The localization of Zamboni is a very important functional subsystem for our system. As the Zamboni is localized in a closed environment of ice rink, there is no GPS to estimate its pose as well as validate the pose. Having accurate data is important because the path generation and path following controller performance depend upon the same. So the way we are performing Zamboni localization is by fusing the sensor data from wheel encoders as well as from IMU and getting fused pose information using Extended Kalman Filter.

I developed the sensor fusion package using the Robot_Localization package of the ROS. The robot_localization is a collection of state estimation nodes, each of which is an implementation of a nonlinear state estimator for robots moving in 3D/2D space. It contains two state estimation nodes, ekf_localization_node, and ukf_localization_node.

I decided to use the ekf_localization package because of the ease to tune and weigh each sensor data through the covariance of the Extended Kalman Filter (EKF). As this package can fuse any number of sensors that provide odometry data, I worked on fusing wheel odometry of both the rear wheels and IMU mounted on top of the vehicle.

Although the IMU provides acceleration as well as angular velocity data, I decided to use the angular velocities from IMU, and velocity and position information from the Wheel Odometry to fuse and use EKF to estimate accurate pose information (Based on advice from TA's and Prof. Dolan). The fused and filtered pose information of the Zamboni is published on /filtered/odometry topic to which other subsystems are subscribed.

To validate the performance of the sensor fusion in estimating the pose of the Zamboni, I added some noise to the data published by the wheel odometry and used the fusion algorithm to estimate the pose of the vehicle. It is observed that the EKF successfully estimates the pose of the Zamboni with very good accuracy (0.1% deviation) compared to the noisy estimates from only the wheel odometry.

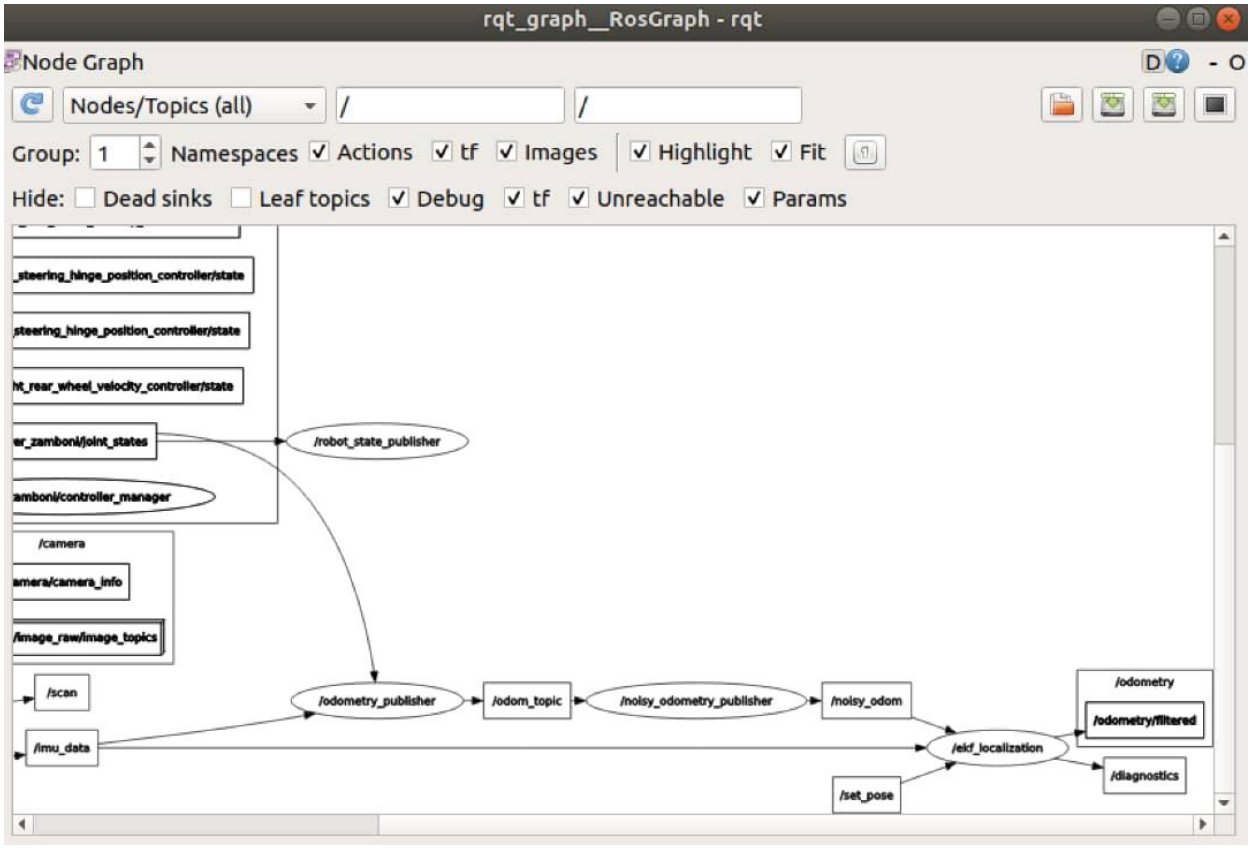

Figure 1. RQT Tree for Sensor Fusion

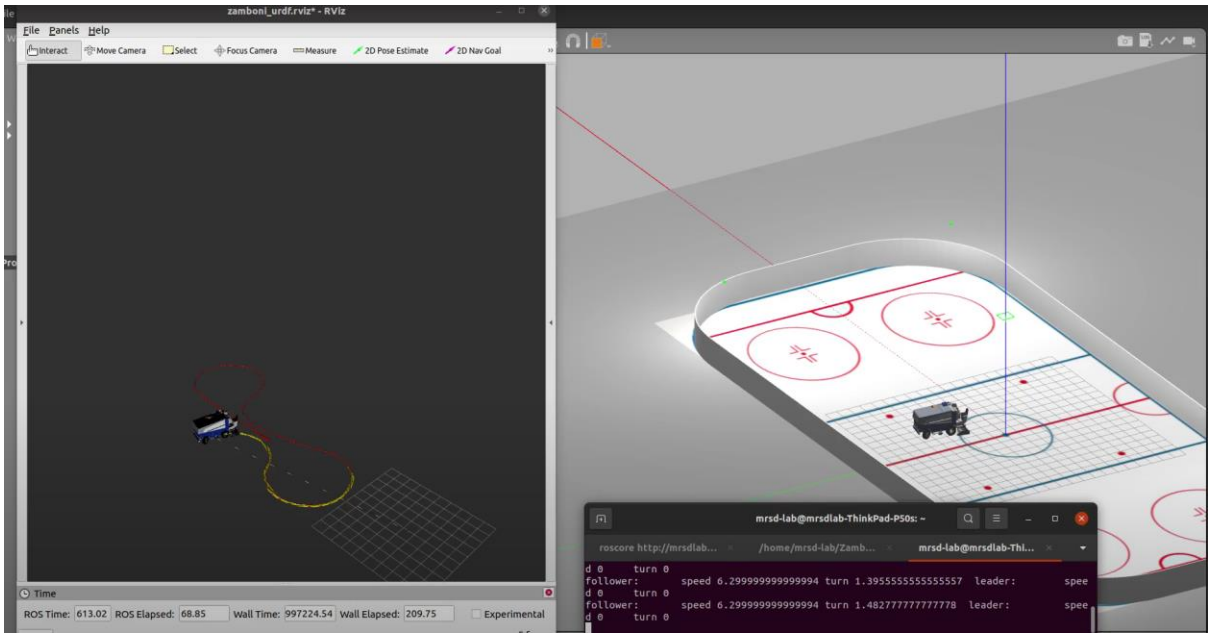

Figure 2. Pose Estimation using Sensor Fusion

As can be seen in the Figure 2., the yellow path is the pose estimated by only wheel odometry which can be seen to be noisy, however the filtered and fused estimate using the fusion algorithm (Red) seems is accurate and a better estimate.

1.2. Pure Pursuit Controller: Validation and Integration with Zamboni Simulator

In the last week I had developed a base version of Pure Pursuit Controller in Simulink that calculated the steering angles based on the waypoints generated that the vehicle needs to follow. The performance of the standalone controller alone was validated for an Ackermann-based geometry developed in SimScape.

Once the functional validation was completed, the next step I completed was integrating the controller with the ROS Simulator developed by Yilin. MATLAB provides ROS Toolbox which has the capability to connect to ROS network and exchange data.

The basic flow of the same is described in the figure below

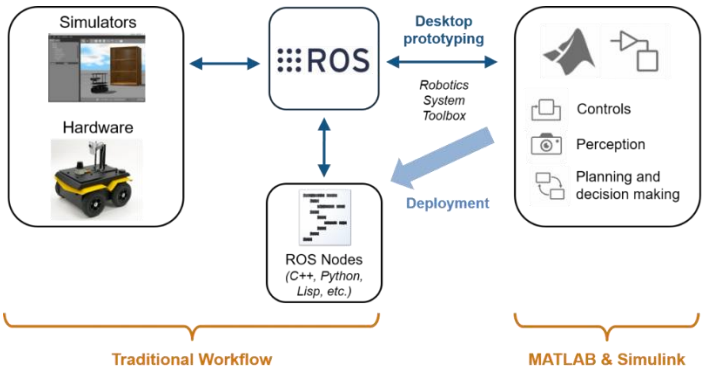

Figure 3. MATALB-ROS Workflow

I followed the same flow and developed a model that subscribes to the pose data published on filtered/odom topic, calculates steering angle, and publishes it on steering control topic. The flowchart below describes how I did it.

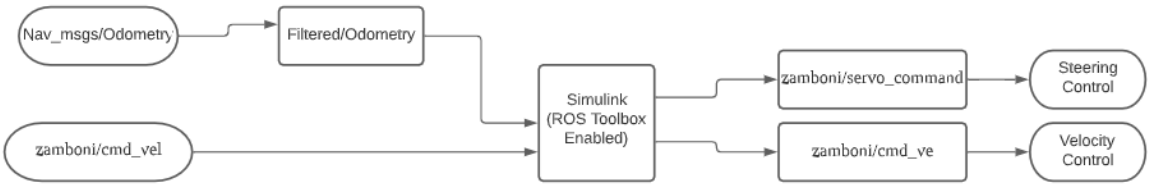

Figure 4. MATLAB-ROS Integration Flowchart

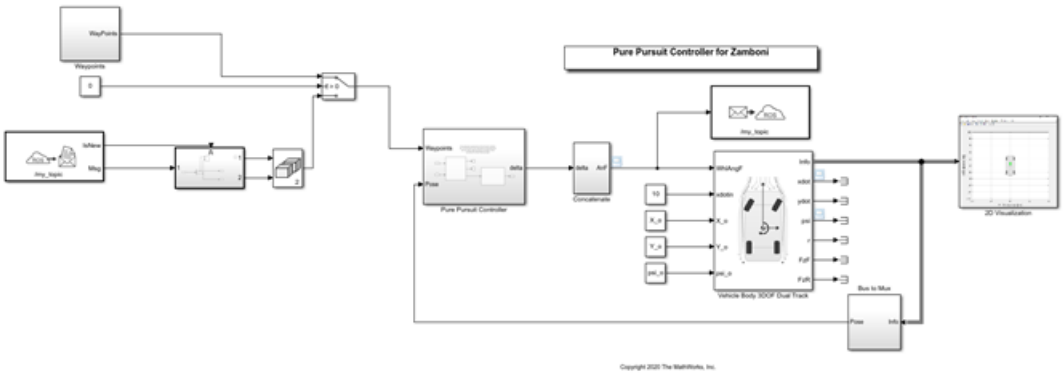

Figure 5. MATLAB-ROS Simulink Controller

I was successfully able to control the Zamboni using the Steering Controller developed in Simulink. The figure below the path followed by the vehicle based on the waypoints provided.

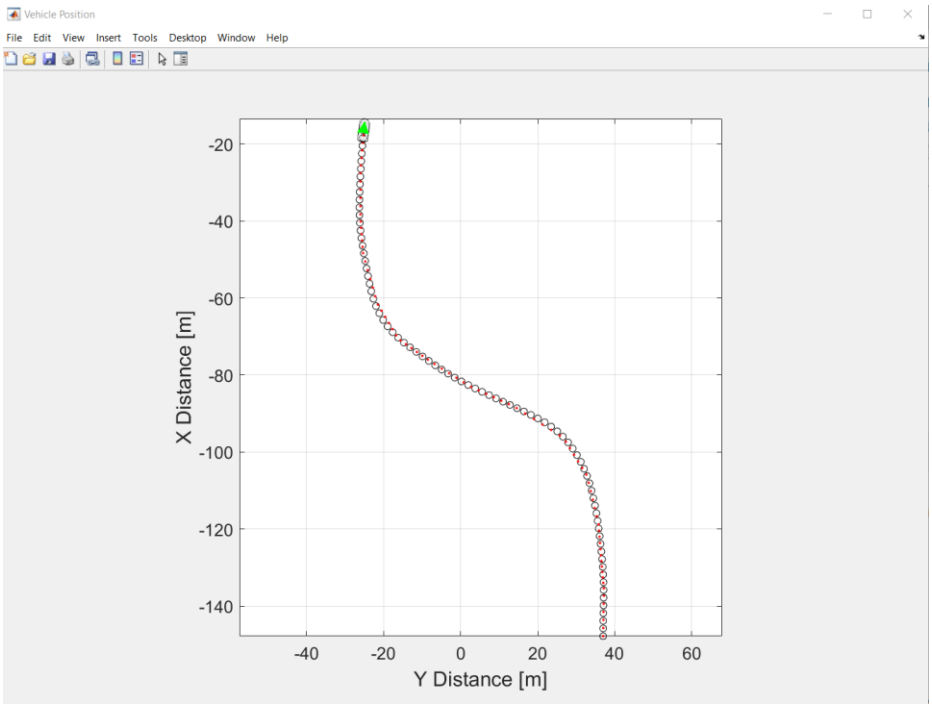

Figure 6. MATLAB-ROS Simulink Controller

1.3. PCB Schematic design:

Apart from working directly on the project, I handled the PCB assignment for our team. I designed the basic schematic from the part list given. Figure 1 shows the schematic developed. I decided to use the voltage linear regulators to convert and regulate voltage . I used fuses rated to break at 150% the expected current at the input connector and the

output of each regulator. For the input that meant limiting the current to 15A, then 3A for the 12V regulator output, and 1.5A for both the 5V and the 3.3V regulator outputs.

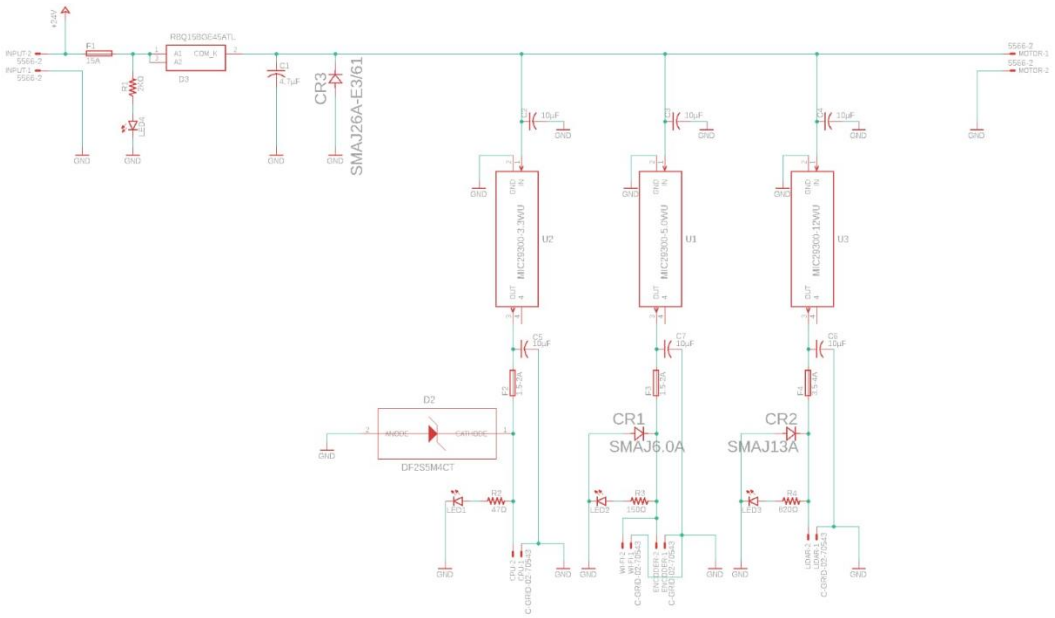

Figure 7. MATLAB-ROS Simulink Controller

2. Challenges

2.1. Zamboni Localization

The major challenges were faced during the testing phase.

The wheels continue rotating sometimes when the Zamboni hit the wall, and then the odometer became invalid, and fusion wasn't making sense. Had to implement a collision check and tune the EKF such that when such a situation happens, the IMU data is given more weightage

Also, the odometry values are too accurate in simulation when the Zamboni is driving and hasn't hit an obstacle. So, to validate the performance of the fusion algorithm, I infused some noise in odometry and validated the algorithm.

2.2. Pure Pursuit Controller Integration

There was a learning curve challenge as there were not lot of resources available to look into this integration. Integrating the MATLAB-Ros communication setup was one major challenge, especially synchronizing the master and client simulation clock. I had to spend some time debugging the communication lag.

Also tuning the pure pursuit controller for good performance was a difficult task as I had to tune the lookahead distance to get the steering angles such that the path/waypoint is smoothly followed.

3. Teamwork

a. Kelvin Shen

Kelvin integrated RealSense D435i into Zamboni model in Gazebo. Using the RealSense Camera he estimated the pose of the leader Zamboni using ArUco board and the RGB image. Another area that he worked was on extracting depth values in the Depth image after cropping out the ROI based on the marker detections

b. Nick Carcione

Nick familiarized with current Zamboni architecture and identified hardware components necessary for drive-by-wire conversion and began looking into their availability. He also acquired the backup RC car platform and has started working on it to integrate the hardware and sensors.

c. Yilin Cai

Yilin worked on multi-robots spawning in Gazebo and command the two Zamboni moving individually by keyboard teleoperation. He also worked on working wheel odometry message for follower Zamboni .He also helped me in integration of robot localization package fusing odometry and imu for follower's pose estimation

d. Jiayi Qui

Jiayi solved ROS version conflict when launching Zamboni URDF model . She majorly focused on working with ROS navigation including teb_local_planner plugins, move_base package which will be used in future for path planning and navigation.

4. Future Work/Plans

In the coming weeks, my focus will be majorly on getting the RC platform and making it ready to use. The aim is to deploy all the software on the RC car and also integrate the sensors on the vehicle. By doing so we will be able to validate most of our subsystems on hardware before deploying the software stack on the actual Zamboni. I also plan to work on developing the algorithm to generate a smooth path based on the waypoints and send that path to the Pure Pursuit controller.