Capstone Project II Final Report

Autonomous Wheelchair Navigation System Using SLAM and Vision-Based Control

Bekmurat Amangeldiyev
Aisultan Kaipiyev

A thesis submitted in part fulfilment of the degree of

BS in Robotics Engineering

Supervisor: Professor Almas Shintemirov

Instructor: Professor Anara Sandygulova

Department of Robotics and Mechatronics
School of Engineering and Digital Sciences
Nazarbayev University
May 7, 2025

Table of Contents

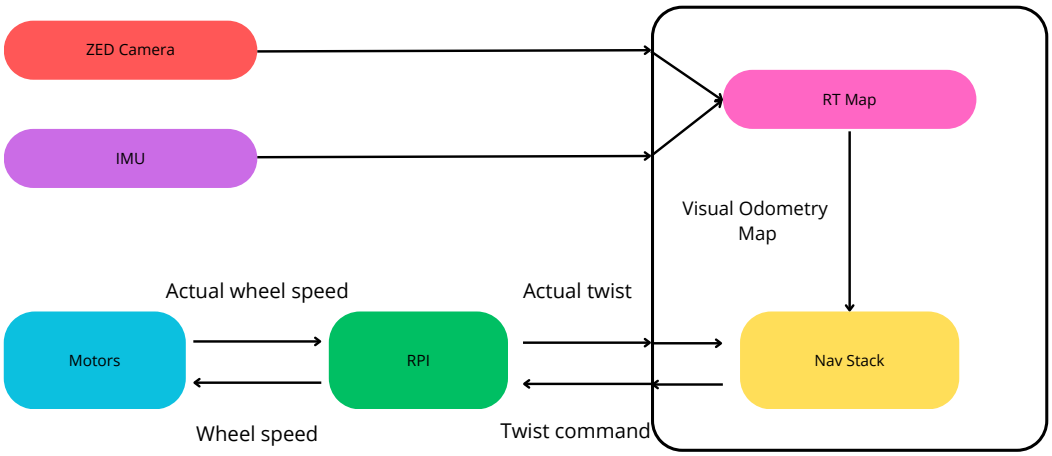
Abstract

This project focuses on enhancing the functionality of a power-assisted wheelchair by integrating autonomous navigation capabilities using a Raspberry Pi microcomputer, ZED 2 depth camera, and UM7-LT IMU sensor. The system leverages Simultaneous Localization and Mapping (SLAM) through the RTAB-Map framework, utilizing the ROS2 Navigation Stack for real-time obstacle avoidance and efficient path planning. A CAN-bus interface has been established between the control computer and the wheelchair, allowing precise control via terminal commands and enabling autonomous navigation in complex environments.

Progress includes completing a communication setup using a Raspberry Pi with CAN2RNET for direct wheelchair control, including commands for movement and feedback using an Xbox joystick and ROS Twist messages sent from the laptop. Additionally, SolidWorks designs were finalized to securely mount the ZED 2 camera and IMU sensor on the wheelchair. This ensures stable data acquisition, critical for accurate SLAM and navigation.

Future work will focus on integrating sensor data into ROS2 for enhanced navigation. By refining navigation algorithms and creating a user-friendly interface, this project aims to improve mobility for individuals with disabilities, enhancing independence and safety in their daily lives.

Index Terms - ROS2, SLAM, RTAB-Map, Autonomous Navigation, IMU, CAN-bus, Odometry, CAD Design.



Acknowledgments

Supervisor: Almas Shintemirov, PhD, Associate Professor Nazarbayev University, School of Engineering and Digital Sciences
https://orcid.org/0000-0002-6969-8529
Email: ashintemirov@nu.edu.kz

Dr. Almas Shintemirov received his PhD in Electrical Engineering and Electronics from the University of Liverpool, UK, in 2009. In 2011 he became one of three founding faculty members of the Department of Robotics and Mechatronics at Nazarbayev University (NU), where he has contributed to setting up undergraduate and graduate robotics and mechatronics academic programs as well as developing state-of-the-art teaching and research infrastructure and research capacity of the department. He is currently an Associate Professor of Robotics and Mechatronics and from September 2019 also serves as an Acting Chair of the Department of Electrical and Computer Engineering at NU. Dr. Shintemirov's main research interest is intelligent human-machine interface designs for serial and parallel industrial, assistive, and mobile robot control. He has been a PI or co-investigator of numerous projects funded by the Ministry of Education of Kazakhstan and Nazarbayev University. For his contribution to technology development, he received the "The Certificate of Merit of the Republic of Kazakhstan" award in 2017, and the Kazakhstan Scopus Award 2018 – Top Researcher in Engineering and Technologies from Elsevier. More information and video demonstrations of selected research, student courses, and graduate projects can be found at Dr. Shintemirov's research group website https://www.alaris.kz/.

We sincerely thank research assistant Artemiy Oleinikov for his dedicated support and valuable contributions throughout this project. His expertise gave us valuable insight into the realities of being a roboticist and the challenges engineers face daily. We believe this year has been pivotal in shaping our professional growth as robotics students.

Chapter 1: **Introduction**

The global population is experiencing a significant rise in the number of individuals with special needs and disabilities. According to the World Health Organization's World Report on Disability, approximately 15% of the world's population, or around one billion people, live with disabilities, with 300 million facing severe limitations that affect their ability to perform daily activities independently. Cerebrovascular incidents, such as strokes and brain injuries, are among the leading causes of motor disabilities, with 85% of cases resulting in long-term motor impairments. In Kazakhstan alone, nearly 473 out of every 100,000 individuals suffer from cerebrovascular accidents each year, a figure more than twice as high as that of the United States. Of the 49,000 people who suffer strokes annually in Kazakhstan, 80% are left fully or partially disabled, necessitating long-term institutional care or assistance.

These alarming statistics highlight the growing demand for assistive technologies to enhance the quality of life for people with mobility impairments. Autonomous wheelchairs, especially those integrated with intelligent systems like computer vision, represent a transformative solution for this demographic. These systems allow users to regain independence, navigating their environments safely and efficiently, even in cases where motor function is severely impaired. Autonomous wheelchairs not only reduce the burden on caregivers but also provide a much-needed sense of autonomy to individuals who may otherwise rely heavily on external support for daily mobility. The integration of advanced technologies into these wheelchairs is thus crucial in addressing the mobility challenges posed by the increasing number of individuals with disabilities worldwide.

The goal of this project is to develop a smart, power-assisted wheelchair integrated with Simultaneous Localization and Mapping (SLAM) and camera technologies to improve the mobility experience for individuals with disabilities. Navigating wheelchairs can be challenging in complex environments, and manual control often requires significant effort, especially for those with physical limitations. By integrating SLAM and camera systems, the wheelchair can autonomously map its surroundings, avoid obstacles, and assist users in navigating safely, reducing the cognitive and physical strain of controlling it manually. The project leverages advancements in robotics and sensor technology, making use of depth cameras and IMU sensors to achieve accurate motion planning.

The inclusion of a user-friendly interface allows for seamless control from a laptop or other devices, ensuring flexibility for different users' needs. This project aims to enhance autonomy, giving users greater freedom and confidence in moving through various environments, from homes to public spaces. The significance lies in its potential to improve the quality of life for people with disabilities, allowing for safer, more efficient, and less physically demanding mobility solutions. By simplifying wheelchair navigation, this project directly contributes to increasing independence and accessibility in daily life.

Objectives:

- Develop a power-assisted wheelchair with integrated SLAM and camera technologies for autonomous navigation;
- Implement a user-friendly control system that simplifies wheelchair operations for individuals with disabilities;
- Integrate sensors to enhance obstacle detection and safe navigation in diverse environ-

ments.

Significance:

- Enhances Independence. Provides users with greater autonomy by reducing the need for manual control and caregiver assistance.
- Improves Safety. Obstacle detection and avoidance technology increase safety during navigation.
- Promotes Accessibility. Simplifies wheelchair operation, making mobility more accessible for individuals with varying levels of physical ability.

Chapter 2: **Ethics Portfolio**

---

1. Identification of Ethical and Legal Issues

During the development of the autonomous wheelchair, several ethical and legal concerns emerged, particularly around data protection and user safety. The wheelchair relies on a depth camera and IMU sensors to capture environmental data for SLAM and obstacle avoidance. This raises privacy concerns(e.g. Location of the user) about how the data is collected, stored, and processed, especially in public environments where third-party individuals may inadvertently be recorded.

To address this, we ensured compliance with GDPR (General Data Protection Regulation) by anonymizing any data collected during testing and using it only to improve the system's navigation accuracy. Additionally, we think that any personal data from users (e.g., settings from their tablets, which are used for manual control of the wheelchair) should be encrypted to protect their privacy. As known, GDPR consists of seven principles. Here are a list of possible applroaches how we can meet some of them: Lawfulness, fairness and transparency: can be addressed by ensuring users were informed about how data was used. Purpose limitation: use acquired data solely to improve the system's performance. Data minimisation: only collecting necessary sensor data for navigation. Storage limitation: ensuring data was not kept longer than needed. Integrity and confidentiality (security): implement measures, like encryption, to protect sensitive information and set policies for storage limitation. Another key legal issue is intellectual property (IP). Since we are using open-source software like ROS2 and libraries for sensor integration, it was important to understand the licensing agreements to avoid violations. We adhered to the licensing terms (e.g., GNU GPL) and made sure to attribute any third-party code used in our system.

2. Informed Judgments and Trade-offs

One of the key trade-offs in this project was between safety and cost. The inclusion of high-quality sensors like the ZED2 depth camera, which provides advanced 3D mapping, was essential for ensuring the safe navigation of the wheelchair. However, this came at a significantly higher cost. After careful consideration, we prioritized safety over cost, as the target user group—individuals with severe mobility impairments—would benefit most from a reliable, safe system.

Additionally, we faced a trade-off between privacy and convenience. To ensure optimal navigation, it would be convenient to collect continuous data in real-world environments (this allows the constant update of the created 3D map). However, respecting the privacy of bystanders was crucial. We implemented a local data storage system where data could be purged after each session to balance these concerns.

3. Sustainability and Social Responsibility

From a sustainability standpoint, we reflected on the materials and power sources used in the system. The wheelchair operates on batteries, and our selection of hardware considered energy efficiency to minimize environmental impact. Using open-source tools like ROS2 also contributes to sustainability by reducing redundancy in software development.

Social responsibility plays a critical role in our decision-making process. Our project is aimed at enhancing the autonomy of individuals with disabilities, potentially reducing their

dependence on caregivers and improving their quality of life. However, regarding the societal implications of robotics and automation, the increasing use of autonomous systems could lead to job displacement in the caregiving sector. To mitigate this, we focused on designing a system that assists rather than fully replaces human involvement, ensuring that caregivers still play a critical role in complex tasks.

4. Examples and Scenarios

A hypothetical scenario we considered was the liability in case of accidents. Autonomous systems always carry the risk of malfunctioning. If the wheelchair were to cause harm, the question of accountability becomes critical. As part of the design process, we will try to introduce multiple safety layers, including manual overrides and emergency stop functions, to ensure that the user remains in control. Depending on the initial linear velocity of the wheels, protocols for slightly different conditions can be implemented. For example, if the speed is maximum(10 km/h) and the wheelchair continues to move in the direction of the obstacle when the distance is ¡ 5 meters, the wheelchair should decrease the speed and stop immediately. Second scenario, if the speed is not that high(3 km/h), the speed can be nullified if the distance is ¡ 1.5 meters.

Another example can be the potential job displacement in the caregiving sector, which as described above in section 3.

By considering ethical, legal, and societal implications, we aim to create an autonomous wheelchair system that is not only technically sound but also aligned with responsible innovation principles. This establishes our decision-making throughout the project, ensuring that our solution is sustainable, safe, and respectful of the rights and well-being of all potential stakeholders involved.

Chapter 3: **Literature Review(Background Research)**

The field of assistive technology has seen significant advancements over the past few decades, with autonomous wheelchairs emerging as a promising solution for individuals with severe mobility impairments. As the global population of individuals with disabilities continues to grow, particularly those with conditions stemming from cerebrovascular incidents, neurodegenerative diseases, and traumatic injuries, the demand for enhanced mobility aids has become more critical than ever. Autonomous wheelchairs, equipped with intelligent systems such as computer vision, artificial intelligence (AI), and haptic feedback mechanisms, are designed to provide greater independence to users by enabling them to navigate complex environments safely and efficiently without relying on full manual control.

At the current stage only papers [6,7,8] (summaries are provided respectively to references) of the whole reference list were analyzed, and the provided summary was comprised of the material of these papers:

The article "Power Wheelchair Navigation Assistance Using Wearable Vibrotactile Haptics" presents a novel approach to improving power wheelchair navigation through the use of vibrotactile haptic feedback. The research proposes equipping power wheelchair users with one or two vibrotactile armbands, each containing four actuators, to deliver navigation feedback. This feedback assists the user in navigating the environment while leaving full control over the wheelchair's motion. Unlike other systems that employ visual or auditory cues, this approach leverages haptic feedback to reduce cognitive load and avoid overstimulation, a frequent issue with more conventional sensory channels.

The proposed system provides rich and detailed environmental information about the direction and proximity of obstacles, significantly reducing the number of collisions during wheelchair operation, as shown in user studies. These studies reported a 49% reduction in collisions when using the vibrotactile armbands, which were also found to be comfortable and intuitive to use. The armbands offer a non-kinesthetic solution that maintains user autonomy, addressing challenges faced by individuals with reduced mobility or upper limb weakness who might find traditional kinesthetic joysticks difficult to operate. The vibrotactile system allows users to receive immediate environmental feedback, which is crucial for navigating narrow or complex spaces.

The article highlights the flexibility of the vibrotactile system, which can be adapted to different limbs and easily integrated with various wheelchair control systems, including standard joystick controllers. It emphasizes the importance of enhancing users' spatial awareness while preserving their autonomy, which contrasts with semi-autonomous systems that can take over some level of control. Moreover, the vibrotactile feedback system can be customized to provide different types of haptic cues, such as indicating the direction to move or warning about nearby obstacles.

In summary, this research introduces a highly promising, affordable, and flexible solution for assisting power wheelchair users. It combines safety improvements through enhanced environmental awareness with the preservation of user control and autonomy.

The article "Assistive Robotic Technologies for Next-Generation Smart Wheelchairs: Codesign and Modularity to Improve Users' Quality of Life" details the advancements made in the development of assistive technologies for electric wheelchairs as part of the European Union's Interreg ADAPT project. The focus of the article is on the integration of advanced sensing technologies and shared control algorithms to enhance user autonomy and safety. By addressing clinical needs identified by medical professionals, the research resulted in two innovative smart wheelchairs with complementary functionalities, as well as a virtual reality (VR)-based wheelchair simulator. These technologies were extensively tested in experimental trials conducted in both France and the United Kingdom.

The key contribution of the work lies in its human-centric and modular approach to design. The integration of various sensors, such as ultrasonic and time-of-flight sensors, along with omnidirectional vision systems, allows the wheelchair to navigate complex environments while providing real-time feedback to users. This feedback is critical for safe maneuvering and avoiding obstacles, which is a primary reason many individuals abandon electric wheelchairs due to the risk of collisions. The system's shared control algorithms enable the wheelchair to adjust to the user's needs, balancing autonomy with safety.

Additionally, the VR-based driving simulator offers an innovative method for training users in a safe and controlled environment. It allows repeated practice on identical circuits, facilitating skill improvement without the risks associated with real-world training. The VR system, compatible with modern head-mounted displays and immersive environments, aims to replicate the driving experience while ensuring safety and reducing training time.

The article emphasizes the collaborative nature of the project, where researchers, medical professionals, and end users worked together to create technologies that are adaptable to various user impairments and environmental conditions. The research also highlights the potential for future developments, such as enhancing social acceptability and dynamic adaptation of assistance based on real-time user feedback. Overall, the project marks a significant step forward in providing equitable access to mobility for individuals with disabilities.

The research paper [8] introduces a novel framework aimed at enhancing communication between autonomous wheelchairs and both their users and surrounding pedestrians through a shared external human–machine interface (eHMI). The core of this research is the use of an on-ground projection system to visually communicate the wheelchair's motion intentions. This innovation integrates real-world elements with virtual feedback, allowing both users and pedestrians to better predict the wheelchair's movement and respond accordingly.

By focusing on transparent communication of the wheelchair's path, the shared eHMI system reduces cognitive load and fosters proactive user adjustments, significantly improving the safety and usability of autonomous wheelchairs. The study presents the implementation of this system in a simulation environment, exploring how this novel projection-based approach enhances the user's navigation experience and social acceptance. The framework was evaluated through pilot studies involving both wheelchair users and pedestrians, revealing that this system improved trust and safety in interactions with autonomous wheelchairs.

Moreover, the authors emphasize the potential of shared eHMI to bridge the communication gap between humans and machines in complex and unpredictable environments. This projection-based method also enhances accessibility by providing unobtrusive and clear visual cues, contributing to smoother human-machine interaction. Future work is suggested to focus on refining the system's visibility and exploring real-world applications to enhance its practicality and trustworthiness.

Chapter 4: **Methodology**

Our capstone project(development project) aims to develop an autonomous power-assisted wheelchair, integrating Simultaneous Localization and Mapping (SLAM), camera technologies, and additional sensors to enhance mobility for individuals with disabilities. The system's design follows a modular approach, focusing on three key subsystems: odometry, path planning, and path following. Below is a detailed explanation of the methodology used:

1. **System Components:**
    - Wheelchair with CAN-bus Interface: The wheelchair is modified to interact with a control computer via a Raspberry Pi connected through a CAN2RNET cable. The Raspberry Pi communicates directly with the wheelchair, enabling control via terminal commands.
    - Sensors: ZED 2 Depth Camera: Utilized for real-time 3D mapping and obstacle detection. This camera's stereo vision captures high-resolution images and creates a depth map of the environment, crucial for navigation and SLAM. UM7-LT IMU Sensor: Provides orientation and motion tracking through accelerometers and gyroscopes. Its data is essential for accurate pose estimation and stable movement control.
    - RTAB-Map: RTAB-Map is integrated for real-time appearance-based SLAM. This framework builds and updates a map of the environment as the wheelchair moves, enabling autonomous navigation and localization.
    - ROS2 Navigation Stack: The Robot Operating System (ROS2) is used for higher-level control and integration of SLAM, odometry, and path planning. It handles sensor data, obstacle avoidance, and path execution. The navigation stack ensures that the wheelchair can autonomously navigate unknown environments safely.
    - Odometry: Odometry is achieved through data from the wheelchair's wheel encoders and IMU sensor. This enables the system to estimate the position and orientation of the wheelchair, though with possible errors due to cumulative drift.
2. **Methodology Stages:**
    - Odometry Setup: Wheel encoders and the IMU are used to measure and compute the wheelchair's pose (position and orientation) over time. This data is fed into the ROS2 system for real-time tracking.
    - Sensor Integration: Both the ZED 2 depth camera and IMU sensor are mounted securely on the wheelchair to minimize noise and ensure accuracy. The camera is placed above the seat for a clear field of view, and the IMU is positioned near the wheelchair's center of mass for balanced motion sensing.
    - SLAM Implementation: RTAB-Map is configured to work with the ROS2 navigation stack. It uses the depth data from the ZED 2 camera and odometry from the IMU to generate and update a 3D map of the environment. Loop closure detection ensures that errors are minimized by adjusting the map when previously visited areas are recognized.
    - Autonomous Navigation: The ROS2 navigation stack receives map data from RTAB-Map and sensor inputs to plan paths. The system uses global and local planners to create efficient paths and avoid obstacles dynamically.

3. Suitability of Methods

   The use of ROS2, RTAB-Map, and the ZED 2 camera makes the system suitable for real-time, large-scale environments. The SLAM approach ensures the wheelchair can navigate autonomously while continuously updating its map. Using CAN-bus for communication with the wheelchair's hardware allows for real-time control and feedback, and the IMU sensor improves odometry accuracy. Together, these methods create a robust system for reliable autonomous navigation, even in complex environments.

4.1 Wheelchair set-up development

4.1.1 CAN-bus

The Controller Area Network (CAN) bus is a communication protocol widely used in automotive and industrial applications to allow multiple microcontrollers and devices to communicate with each other without a central computer. It was developed by Bosch in 1986, primarily for the automotive industry to reduce wiring complexity and improve communication efficiency among electronic control units (ECUs) such as engine management systems, airbags, and braking systems.

Key Features:

- Message-Based Protocol: CAN transmits data in packets called messages. Each message has an identifier that prioritizes its importance, allowing critical data to be transmitted quickly.
- Fault-Tolerance: CAN bus is designed to handle noisy environments. It uses a differential signal to transmit data, which makes it resilient to electromagnetic interference, a common issue in vehicles and industrial settings.
- Multi-Master, Multi-Slave Network: Multiple nodes (controllers) can send and receive messages without a master controller. Nodes can operate independently and prioritize messages based on their importance.
- Efficient Data Transmission: The CAN bus supports real-time data exchange with a high level of error checking. It can handle data rates up to 1 Mbps for high-speed communication, though higher speeds limit the maximum length of the bus.
- Broadcast Communication: Messages on the CAN bus are broadcast to all nodes, and each node can decide whether to process or ignore a message based on its identifier.
- Wide Application: Apart from vehicles, the CAN bus is used in robotics, industrial automation, medical devices, and even elevators, where reliable and real-time communication is essential.

How It Works:

- CAN bus operates using two wires (CAN_H and CAN_L) with differential voltage signals to transmit data.
- Each node on the network monitors the bus to determine when it can send data and listens for errors in transmission.

- When a node sends a message, all other nodes receive it and determine if they need the data or should ignore it.

  Overall, the CAN bus reduces complexity, improves fault tolerance, and ensures efficient real-time communication, making it ideal for embedded systems with high reliability requirements.

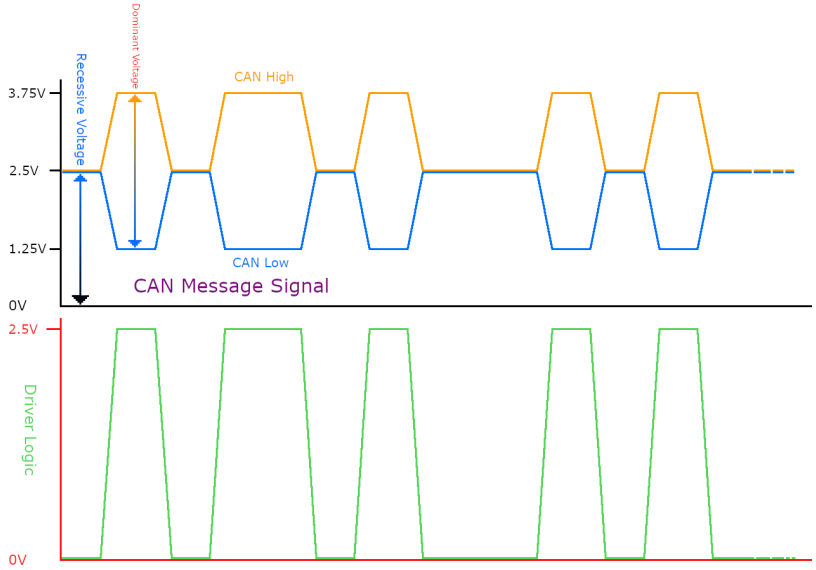


Figure 4.1: Overview of CAN signal.

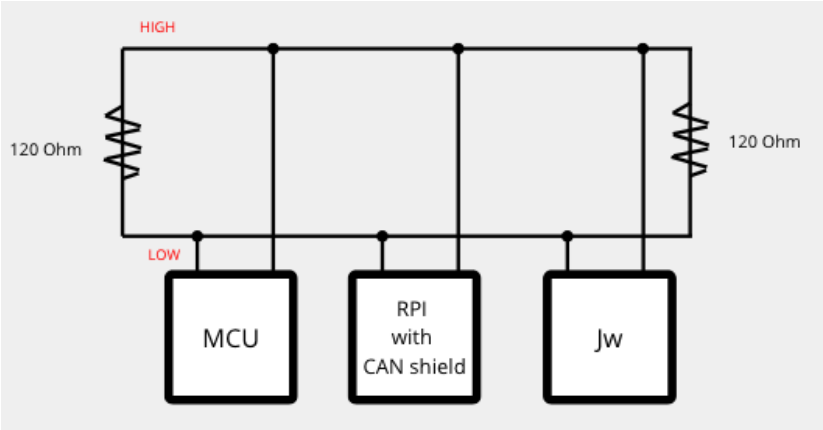


Figure 4.2: Integration scheme of CAN bus into our robot.

4.1.2 Depth camera ZED2

The ZED 2 camera is a high-performance stereo camera designed for depth perception and 3D mapping applications. Here's a detailed overview of its features and capabilities:

Key Features:

1. **Stereo Vision:** The ZED 2 utilizes two high-resolution cameras (left and right) to capture stereo images, allowing it to compute depth information by analyzing the disparity between the two images.

2. **Depth Perception:** It provides real-time depth mapping, enabling applications in robotics, augmented reality (AR), and virtual reality (VR) by creating 3D maps of the environment.

3. **High Resolution:** The camera can capture images at a resolution of up to 4416 x 1242 pixels, ensuring high-quality image capture for detailed analysis.

4. **Wide Field of View:** The ZED 2 features a field of view of approximately 110 degrees horizontally and 80 degrees vertically, allowing it to cover a broad area in its vicinity.

5. **High Frame Rate:** It supports frame rates of up to 100 frames per second, providing smooth video capture and depth data for dynamic environments.

6. **AI Capabilities:** The camera includes built-in algorithms for object detection, tracking, and classification, enhancing its usability in AI-driven applications.

7. **IMU Integration:** The ZED 2 comes with an Inertial Measurement Unit (IMU) that helps improve tracking accuracy and stability by providing orientation data.

8. **Real-Time Processing:** Depth maps can be processed in real-time, making it suitable for applications where immediate feedback is crucial, such as autonomous navigation and obstacle avoidance.

Connectivity:

- The ZED 2 camera connects to computers via USB 3.0, allowing for easy integration with various platforms and applications.
- It is compatible with popular robotics and computer vision frameworks, including ROS (Robot Operating System) and OpenCV, facilitating development and deployment in research and commercial applications.

The ZED 2 camera is a versatile and powerful tool for depth perception and 3D mapping, making it a valuable asset in robotics, AR/VR, and other applications requiring spatial awareness and accurate environmental mapping. Its high resolution, real-time processing capabilities, and integrated AI make it suitable for a wide range of innovative projects.

Figure 4.3: ZED2 Camera.

4.1.3 Differential Drive

A differential drive system is a common locomotion method for mobile robots, including wheelchairs. It consists of two independently driven wheels mounted on either side of the robot. The robot moves forward or backward when both wheels rotate at the same speed and turns when they rotate at different speeds.

This configuration is simple yet effective for planar navigation and is widely supported in ROS.

Inverse Kinematics for Differential Drive

Given a desired linear velocity v and angular velocity ω of the robot (typically from a Twist message), the goal is to compute the required wheel velocities v_L and v_R for the left and right wheels.

Parameters:
v: Linear velocity of the robot (m/s)
ω: Angular velocity of the robot (rad/s)
L: Distance between the two wheels (wheelbase width in meters)
v_L: Velocity of the left wheel (m/s)
v_R: Velocity of the right wheel (m/s)

Final Inverse Kinematics Equations for our wheelchair:

$$v_r = \frac{v}{0.17} + 1.618\dot{\theta}$$

$$v_l = \frac{v}{0.17} - 1.618\dot{\theta}$$

where L = 0.55 m - the distance between the two rear wheels, R = 0.17 m - radius of the wheel. (These values were used in the simplification of the equations)

These equations convert the desired robot motion into individual wheel velocities that can be sent via CAN or other interfaces to motor controllers(in the code "JoyLocal_udp.py"they are labeled as "joystick_x" and "joystick_y" variables).

4.1.4 IMU sensor

An **Inertial Measurement Unit** (IMU) is a sensor device used to detect and measure the motion and orientation of an object. It combines several sensors to provide comprehensive data about acceleration, angular velocity, and sometimes the magnetic field. Here's a detailed description of its components and functionalities:

Components:

1. Accelerometers:
    - Measure linear acceleration along one or more axes.
    - Used to detect changes in velocity and direction.
2. Gyroscopes:

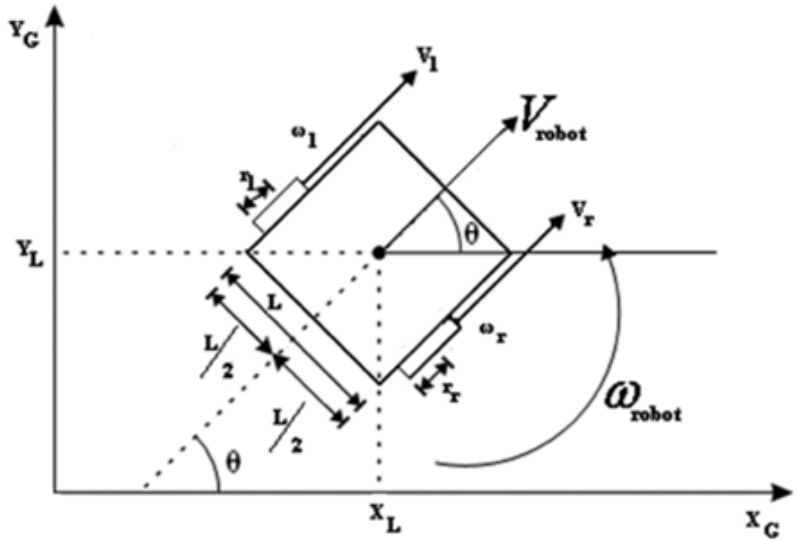


Figure 4.4: Kinematics of the Differential Drive.

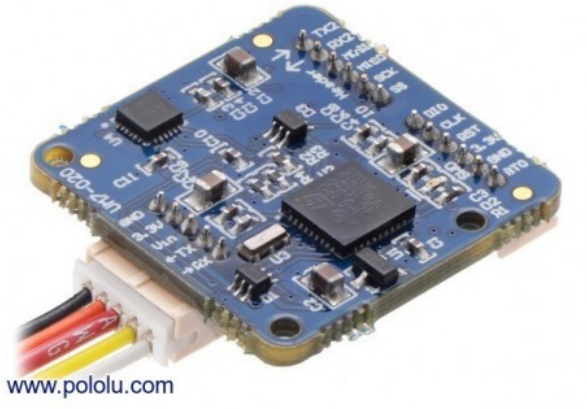


Figure 4.5: UM7-LT Orientation Sensor.

- Measure angular velocity around one or more axes.
- Help determine the object's orientation by tracking rotational movement.

3. Magnetometers(optional):
    - Measure the magnetic field strength and direction.
    - Can be used to provide heading information and improve orientation accuracy.

Functionality:

- Pose Estimation: By integrating data from accelerometers and gyroscopes, IMUs can provide real-time data on an object's orientation and movement.
- Motion Tracking: Useful in applications like robotics, drones, and smartphones to track movements and provide feedback.
- Stabilization: In applications like drones or cameras, IMUs help stabilize motion and compensate for unwanted movements.

Advantages:

- Compact and lightweight.
- Can operate in various environments, including high dynamics and vibrations.
- Provides real-time data essential for navigation and control applications.

For this project specifically, we will use **UM7-LT Orientation Sensor**.

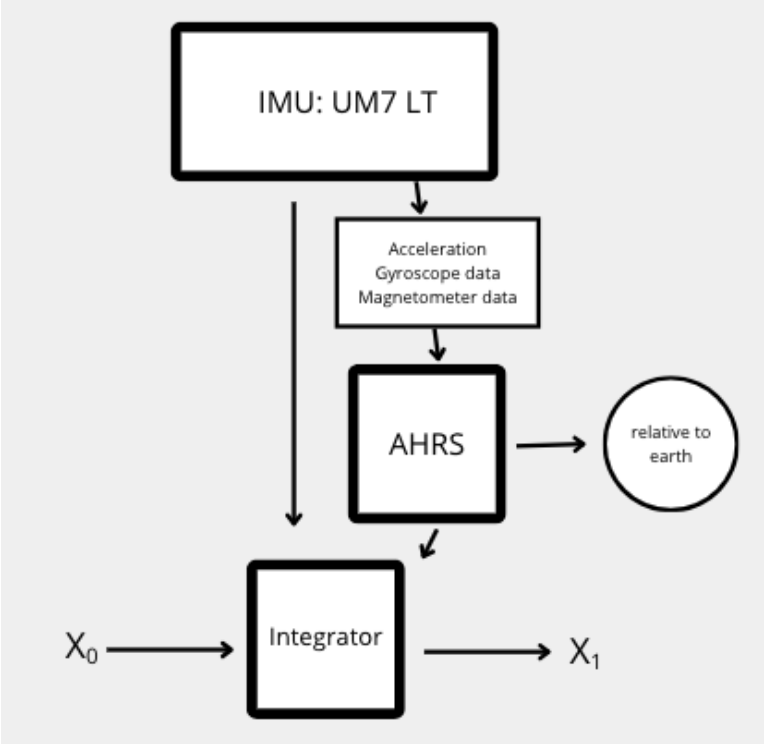


Figure 4.6: The parameters taken from the IMU sensor and the consequent data processing(in our case, RTABmap plays the role of the integrator).

4.1.5 RTABmap

RTAB-Map (Real-Time Appearance-Based Mapping) is an open-source Simultaneous Localization and Mapping (SLAM) framework that is widely used for 3D mapping and navigation in robotics applications. It is designed to work in real-time and provides efficient memory management, making it suitable for large-scale environments.

Features:

1. SLAM (Simultaneous Localization and Mapping):
    - RTAB-Map enables a robot or system to simultaneously build a map of an unknown environment and localize itself within that map. It is capable of operating in both 2D and 3D environments.
2. RGB-D and Stereo Cameras:
    - RTAB-Map is compatible with various sensor types, including RGB-D cameras (e.g., Microsoft Kinect, Intel RealSense) and stereo cameras (e.g., ZED cameras). It uses depth data and visual information to create accurate maps.
3. Loop Closure Detection:
    - One of RTAB-Map's strengths is its ability to detect "loop closures"
    - when the robot revisits a previously mapped area. This allows it to correct drift in the map and refine localization over time, improving the map's consistency and accuracy.
4. Memory Management:
    - RTAB-Map uses an appearance-based memory management approach, meaning it stores visual features of places as the robot moves through the environment. It can manage memory dynamically by discarding less important information to optimize resource usage and scale to larger environments.
5. Real-Time Performance:
    - The framework is designed to operate in real-time, processing data from sensors on the fly, making it suitable for use in autonomous robots, drones, and other applications where immediate feedback is crucial.
6. 3D Point Cloud Generation:
    - RTAB-Map can generate 3D point clouds, which represent the environment in a detailed three-dimensional format. These point clouds can be used for navigation, object recognition, or 3D reconstruction.
7. Multi-Session Mapping:
    - RTAB-Map supports multi-session mapping, which means it can continue building a map over multiple sessions and stitch these sessions together, creating a larger and more complete map of the environment.
8. Compatibility with ROS:
    - RTAB-Map integrates well with the Robot Operating System (ROS), making it a popular choice in robotics research and development. It provides ROS nodes for 2D and 3D SLAM, allowing easy integration into ROS-based systems.

9. Graph-Based Optimization:

    - It uses graph-based optimization techniques to refine the map and the robot's position after loop closures are detected. This ensures that the map is consistent and accurate over time.

Installation and Usage:

- Standalone GUI: RTAB-Map provides a graphical user interface (GUI) for visualizing the map-building process in real-time. The interface shows the 3D map, loop closures, and other key information for debugging and analysis.
- ROS Integration: In ROS, RTAB-Map provides nodes for sensor data acquisition, map generation, and localization, allowing seamless integration with existing ROS systems.

RTAB-Map is a powerful, versatile tool for real-time SLAM, making it ideal for use in robotics, drones, and AR/VR applications. Its capabilities in loop closure detection, 3D mapping, and memory management make it highly efficient for large-scale environments. With its strong integration with ROS, it is widely adopted for research and development in autonomous navigation and mapping systems.

4.1.6 ROS Navigation Stack

The **ROS Navigation Stack** is a collection of ROS packages and tools that allow robots to autonomously navigate in both known and unknown environments. It is widely used in robotics to enable mobile robots to move safely and efficiently while avoiding obstacles and reaching target locations. Here's a detailed description:

Key Features of the ROS Navigation Stack:

1. Localization:

    - The stack includes algorithms for localization, which help the robot determine its position within a given map. The most common method used is Adaptive Monte Carlo Localization (AMCL), a probabilistic approach that continuously estimates the robot's pose (position and orientation) based on sensor data.

2. Mapping:

    - For robots operating in unknown environments, the SLAM (Simultaneous Localization and Mapping) package can be integrated with the navigation stack. SLAM algorithms like Gmapping or RTAB-Map allow the robot to create a map of the environment while localizing itself within it.

3. Path Planning:

    - The navigation stack uses both global and local planners to determine a path from the robot's current position to a target destination.

4. Costmaps:

    - The navigation stack maintains costmaps to represent the environment around the robot. Costmaps assign costs to areas based on their distance from obstacles, helping the robot avoid collisions. There are two types of costmaps:

- Global Costmap: Used by the global planner, representing the static layout of the environment.
- Local Costmap: Used by the local planner, which includes dynamic obstacles like moving people or objects.

5. Obstacle Avoidance:

   - The stack uses sensor data (from LiDAR, cameras, or ultrasonic sensors) to detect and avoid obstacles in real time. The local planner adjusts the robot's movement to steer around obstacles while staying close to the global path.

6. Sensor Integration:

   - The stack integrates data from various sensors such as LiDAR, depth cameras, sonar, and wheel encoders. These sensors are used to build maps, detect obstacles, and estimate the robot's pose.

7. Action Server for Navigation Goals:

   - The Move Base package serves as the core of the navigation stack. It acts as an action server that takes navigation goals (specified in terms of position and orientation) and sends appropriate velocity commands to the robot's motor controllers to achieve the goal while avoiding obstacles.

8. Dynamic Reconfiguration:

   - The stack allows for dynamic reconfiguration of parameters while the system is running. This flexibility enables users to tune the system in real time, optimizing behavior for different environments or tasks.

9. Multi-Robot Support:

   - The ROS Navigation Stack can be extended to work with multiple robots, allowing them to share maps and collaborate on tasks, such as exploring an environment or completing coordinated missions.

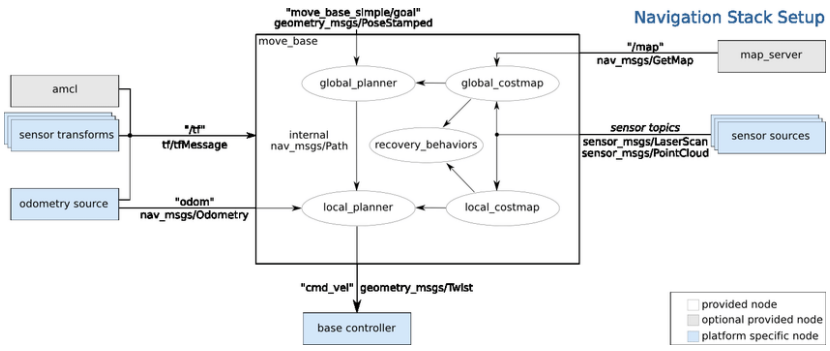


Figure 4.7: Standard ROS Navigation Stack.

For this project we will use ROS2 Navigation Stack, since ROS2 is currently the newest and popular version fo ROS, while the first version is used primarily for education. Navigation Stack diagram can be seen below:

The ROS Navigation Stack is a powerful, flexible tool that provides the core functionality required for autonomous navigation in robots. Its ability to integrate with various sensors, dynamic environment handling, and support for real-time planning makes it a popular choice for robotics research and commercial applications alike.

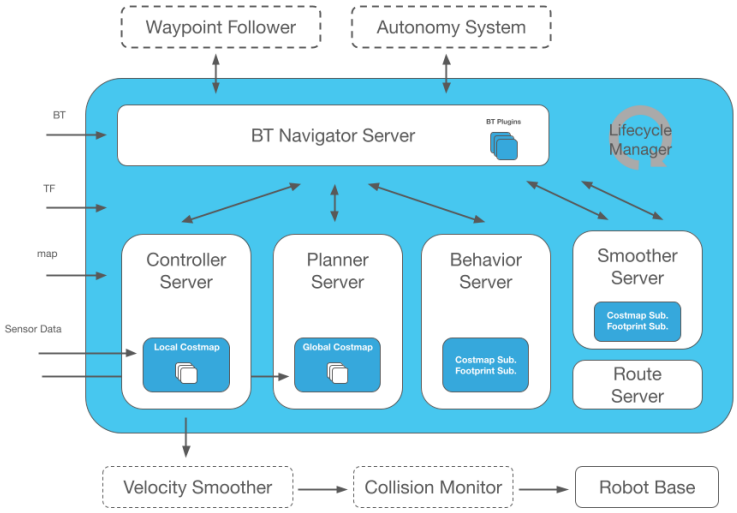


Figure 4.8: ROS2 Navigation Stack.

4.1.7 Odometry

Odometry is the process of estimating a robot's position and orientation (pose) over time by analyzing data from its sensors, typically wheel encoders or IMU. It calculates the robot's displacement based on the motion of its wheels or other movement mechanisms.

Key Concepts:

- Wheel Encoders: Measure the rotation of the wheels, allowing for the estimation of linear and angular displacements.
- Pose Estimation: Odometry provides the robot's position (x, y) and orientation θ in a 2D plane, or sometimes a 3D pose with more advanced setups.
- Cumulative Errors: Odometry is prone to drift and errors over time due to slippage, uneven surfaces, or sensor inaccuracies, which can accumulate, reducing long-term accuracy.

Use in Robotics: Odometry is commonly used for short-term navigation and dead reckoning, where other methods like GPS or SLAM can be combined to correct the drift for improved accuracy.

4.1.8 URDF file

A URDF (Unified Robot Description Format) file is an XML format used in ROS (Robot Operating System) to describe a robot's physical configuration. It defines the structure, geometry, kinematics, and dynamics of the robot, enabling simulation, visualization, and motion planning.

Key Components:

1. Links: Represent the robot's physical parts (e.g., body, wheels, sensors).
2. Joints: Define the connections between links, specifying how they move relative to each other (e.g., rotational, fixed).

3. Inertial Properties: Define the mass, inertia, and center of mass for each link.
4. Sensors and Actuators: Can include descriptions of sensors (e.g., cameras, LiDAR) and actuators (e.g., motors).

Use in ROS: URDF files are used to build accurate robot models in simulation environments like Gazebo, as well as for RViz visualization and motion control tasks.

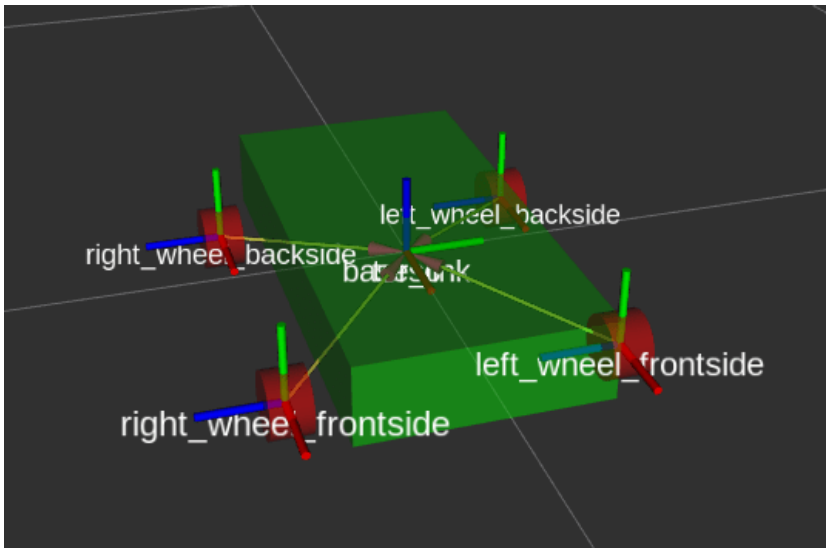


Figure 4.9: Example of the URDF file.

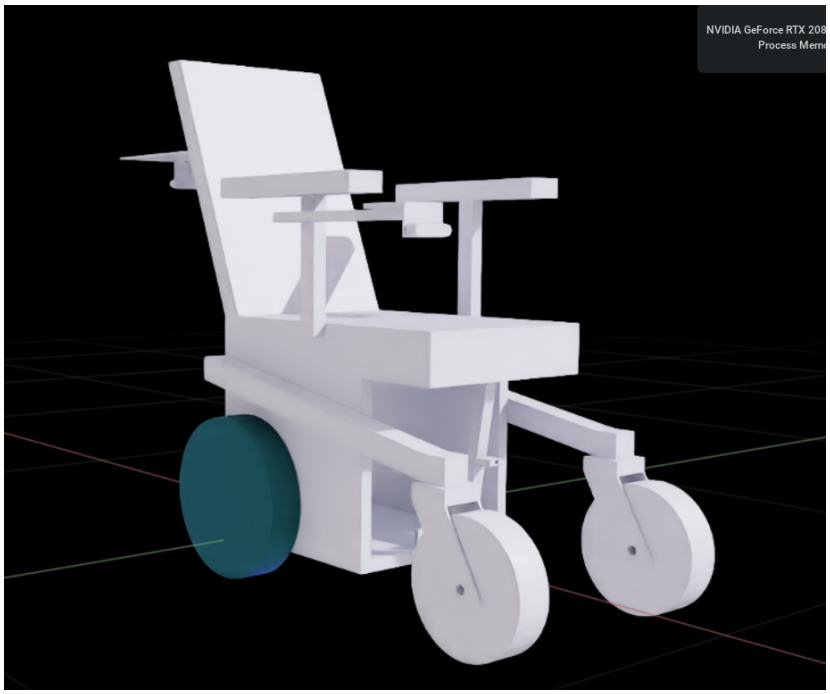


Figure 4.10: The wheelchair's URDF in ROS2.

Chapter 5: **Results**

5.1 Discussion

According to the milestone list, the main task for September and October is establishing direct control of the electric wheelchair. This requires the establishment of a connection between the laptop and the wheelchair via a Raspberry Pi and a CAN2RNET cable.

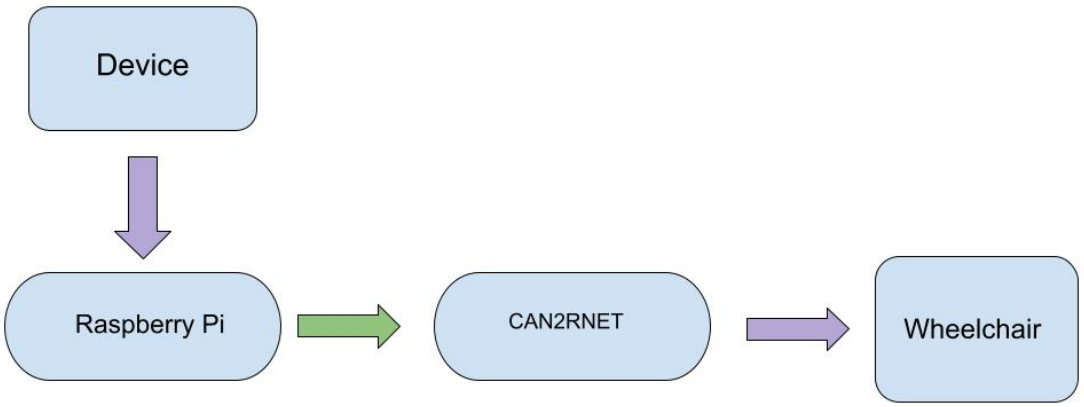


Figure 5.1: Communication between the device(laptop) and the wheelchair.

Therefore, for this task, we found a library on GitHub that provides this communication[1]. By running commands in the terminal, we can utilize the functionality of the wheelchair:

- Making noises, signals via the wheelchair's joystick;
- Control the wheels using the Xbox joystick.

The video demonstrations can be seen in [9] and [10].

Thus, the tasks that had to be done in September and October have already been completed.

The next milestone is "Installation of additional depth camera(s), IMU, and wheelchair encoder sensors to the test wheelchair and their interfacing with the control computer", and it

is set to December. We have already fully enrolled in doing these tasks. However, it is very important to note that to use the depth camera and IMU sensor, we have to fix them firmly on the wheelchair. To achieve this, we had to design a 3D model that would fixate on it. The point of fixation is a pipe located behind the seat(Figure ?). Both boxes(one for the IMU sensor and another for the depth camera) will be attached to the pipe. Only after the design stages will we be able to connect and effectively use both sensors. Otherwise, if sensors are not attached properly, it would cause unwanted noise, which in turn would disturb signal processing.

The design of the Solidworks boxes can be seen in Figure 5.2 below:

The encoder would have helped us to calculate the speed of the wheelchair. After a thorough investigation of the CAN2RNET library, we could not find the code snippet or command that would allow us to increase the speed. However, we can take the value of the current at each instant. The changing values of the current allow us to find the angular velocity. By using the acquired angular velocity V and the measured radius of the wheels R, we can find the linear velocity v.

Therefore, to conclude this section, we have printed the boxes for sensors, and currently, we are trying to connect them(sensors) to the control computer.

According to the Gantt Chart in Figure 7.1, we were supposed to finish CAD Designs and establish communication between the control computer and the wheelchair during the week(October 7th - 13th). However, we faced some difficulties, which hindered our progress.

CAD Model

The CAD model design for both the camera holder and the IMU sensor was extended till this week, and it has not been finished yet. We had multiple iterations of each design. The main problem was correctly measuring all parameters of the places where we needed to fix our sensor. Since the wheelchair initially was not designed to accommodate any additional parts, we checked again a few possible locations to fix our sensors. After making the final decision, we continued to build our designs. After finalizing the design of the IMU holder, we directed our efforts to the camera holder.

During that stage, we had many iterations of the camera holder design, both testing (specific parts of our design to check if the measurements were correct) and whole parts. The most recent draft that we made is:

However, since it has excessive space between the plates, it would bend due to the weight of the metal rod holding the camera, because the holder is made out of plastic. Thus, we decided to change the two-plate model to a single-layer plate to eliminate plate bending, simplify the design, and use less material for printing.

This design will be demonstrated during the final presentation.

ROS2 and URDF

Since a greater amount of time was spent on the CAD Design, the learning of ROS2 and the Navigation stack was postponed as well. We started by making a simple simulation of the turtlebot to understand the navigation and odometry. This is why we have decided to finish the semester by completing only the simulation in Gazebo and RViz part(that would require us to create a URDF file) and not going further to the testing on the real robot.

Navigation Simulation in ROS2 Gazebo

With the finalized URDF file, **the next crucial step** was to set up and conduct navigation simulations for the autonomous wheelchair within the ROS2 Gazebo. This involved configuring the simulation environment using ROS2 (Humble distribution) and Gazebo (version 11). The finalized URDF of the wheelchair robot was integrated into Gazebo by creating a dedicated ROS2 package that contained the URDF file along with any necessary launch files and configuration parameters. Also, a simple virtual environment, resembling an indoor space with walls and obstacles, was created within Gazebo to provide a context for the navigation simulations.

To control the wheelchair robot within this simulated environment, several methods were explored. Initially, keyboard teleoperation was implemented using existing ROS2 packages, allowing for manual control of the robot's linear and angular velocities via keyboard inputs. This manual control was essential for basic validation of the robot's model and its movement within the simulation. Subsequently, custom ROS2 nodes were developed to publish specific velocity commands to the robot, enabling more controlled and repeatable testing scenarios. The process of working out the robot's controls also involved tuning parameters such as the maximum linear and angular velocities to achieve smooth and predictable movements.

The use of ROS2 and Gazebo provided numerous benefits for the development of the autonomous wheelchair. ROS2 offers significant improvements in performance, robustness, and support for real-time systems. Its distributed architecture, based on the Data Distribution Service (DDS) middleware, enhances reliability by eliminating the single point of failure present in ROS1. This enhanced performance is crucial for an autonomous wheelchair, which needs to respond quickly and reliably to sensor data and environmental changes.

Draft RTAB MAP Simulations on Turtlebot3

To begin exploring the mapping and localization capabilities for the autonomous wheelchair, draft simulations were conducted using the Turtlebot3 Waffle Pi platform. The Turtlebot3 was chosen for these initial experiments due to its widespread use in robotics research and education, its affordability and accessibility, and its robust integration within ROS. Furthermore, the availability of accurate simulated models of the Turtlebot3 in Gazebo makes it an ideal platform for preliminary testing of algorithms like RTAB MAP.

The setup and execution of the draft RTAB MAP simulations involved installing the rtabmap_ros package within the ROS2 Humble environment. Subsequently, the Gazebo simulation environment was launched with the Turtlebot3 Waffle Pi model. The RTAB MAP nodes were then launched, configured to utilize the simulated sensor data from the Turtlebot3, primarily the lidar and camera feeds. The simulations were run by manually teleoperating the virtual Turtlebot3 within a simulated indoor environment, allowing RTAB MAP to process the sensor data in real-time.

The preliminary observations from these simulations were encouraging. RTAB MAP demonstrated the ability to generate a 2D occupancy grid map and a 3D point cloud map of the simulated environment as the Turtlebot3 was moving. The generated maps showed a reasonable level of accuracy in representing the walls and obstacles within the virtual world.

RTAB MAP Simulation with our wheelchair

The following describes our current progress in developing an RTABMap simulation using ROS2 within a Gazebo Classic environment. Our primary goal was to simulate wheelchair

navigation utilizing a depth camera plugin provided by Gazebo's official plugins. For smooth integration, we developed a customized RTABMap launch file, specifically adapted to our simulation setup. Using this tailored configuration, we achieved detailed and accurate 3D mapping results. The depth camera data enabled RTABMap to generate precise 3D point clouds representing the simulated environment, which were subsequently processed into a highly usable 2D occupancy grid map. This derived 2D map effectively supported autonomous wheelchair navigation using the Nav2 stack, demonstrating robust path planning and obstacle avoidance capabilities within the simulated scenario.

The provided RQT graph illustrates our simulation's node architecture, highlighting the established interactions ensuring all processes run seamlessly.

Joystick Hack for Twist Command

Given the objective of developing an autonomous robotic wheelchair, our initial approach was to establish real-time control from a remote laptop. While this ideally necessitates the installation of a full ROS 2 environment on both the control computer and the onboard Raspberry Pi(with assigning Raspberry PI as "master" and the laptop as "slave"), practical constraints—including limited processing capabilities of the Raspberry Pi—led us to adopt a more lightweight alternative. Instead of running a full ROS 2 stack on the Pi, we implemented a simplified Python listener script that directly interfaces with the wheelchair's motor controller via the CAN bus, using a dedicated Python library for command translation.

To enable directional movement and dynamic control of both linear and angular velocities, we integrated a teleoperation interface on the control laptop. A corresponding talker node was developed to capture teleoperation inputs and transmit them wirelessly to the Raspberry Pi using the UDP protocol. This communication method was chosen over TCP due to its lower latency, which is advantageous for real-time control despite UDP's lack of delivery guarantees. Importantly, both the sender (talker) and receiver (listener) scripts were configured with the specific IP address of the Raspberry Pi to ensure proper network connectivity and reliable data exchange.

The video demonstration can be seen in Google Drive.

5.2 Challenges and Solutions

During the implementation of the idea, we have encountered a few problems, both small and those that require more serious consideration. These problems were different in terms of their essence, ranging from simple but complex redesigning the CAD Model for our project multiple times to hard debugging problems, such as installing the required drivers for the utilization of parallel structures of the GPU to run simulations.

No problems or challenges that require serious consideration have occurred in the beginning(first term), since the initial parts of the project are mainly researching: finding libraries, and downloading necessary communication protocols for hardware. However, since we started to get more hands-on experience in simulations, more technical issues began to arise.

1. The autonomous wheelchair by default can not be communicated directly with the control computer. Thus, it was possible for us to connect it only via the CAN-bus and Raspberry Pi(with PiCAN bus). Only then we were able to apply terminal commands

from the library to control the wheelchair.

2. Due to the connection via CAN-bus, there are a multiple wires between three devices: the wheelchair and RaspPi, RaspPi and control computer, RaspPi and another external source, which was another laptop in our case(RasPi needs 5V as supply, but the wheelchair by default provides 24V which can burn the circuit, so we connected RasPi to another laptop via USB, which gives required voltage). That makes the wheelchair not freely movable: we had to lift the main(rear) wheels to ensure it remains in the same place while we are conducting tests. Thus, one of the challenges that have to be solved will be making the system which will allow us to move in the free space(so, every necessary component should be placed or hang on the wheelchair).

3. Another challenge was choosing the place for both camera and IMU sensor holders. Ideally, the camera has to be fixed somewhere above the seat, and the IMU sensor should also be fixed somewhere nearby the center of mass of the wheelchair. Thus, the rear pipe behind the seat was chosen as a point, as shown on figure 4.?, since it allows for both the camera to be above the seat and IMU sensor to be approximately at the center of the wheelchair.

4. **Navigation Process:** Throughout the nav. simulation process, various debugging issues arose. One common issue was the robot not moving as expected in response to control commands. This was often traced back to incorrect topic names or message types being used for communication between the control nodes and the robot model in Gazebo. Inspecting the active ROS2 topics using the **"ros2 topic list"** command and examining the content of the messages using **"ros2 topic echo"** helped to identify and rectify these communication problems.

5. Another challenge involved the robot colliding unexpectedly with virtual objects in the environment. This often necessitated adjustments to the collision geometry defined in the URDF file to better represent the robot's physical extent. In some instances, the robot exhibited unstable or erratic behavior, which required fine-tuning of the control parameters and further investigation into the robot's dynamic properties as defined in the URDF.

6. **RTAB MAP:** Some initial challenges were encountered, particularly with achieving consistent loop closure detection in certain areas of the environment, which is crucial for correcting accumulated errors in the map. Additionally, the computational load of running RTAB MAP and Gazebo simultaneously was noted, indicating a need for potential optimization in future implementations. The preliminary assessment also suggested that the Turtlebot3 was able to localize itself reasonably well within the map generated by RTAB MAP.

7. **Drivers for GPU and new Linux:** RTAB MAP is a pretty heavy environment and requires a lot of computational power. The first issues already arose during the installation process. The PC provided for this project has an NVIDIA Graphical Processing Unit, and a similar project was already held on the same computer, except that it was based on ROS1. It was assumed that it should be the same software for the second distribution. However, the same error was appearing during the installation multiple times, which pointed out how much the current amount of RAM is not enough to complete the procedure. In that case, we came up with a few solutions. First, to not overflow the RAM, it was suggested to increase the "Swap" memory. It should have provided additional space so that the computations would not overflow the RAM during the process and freeze the whole PC. As a result, using this method, it was indeed possible to install up to app. 54% of the files. However, it still was not enough, since the same mistake repeated itself. Secondly, we decided to divide the installation process into separate commands to decrease the load on the system. However, after partitioning the load and adding space to the "Swap" memory, we still were not able to run RTAB

MAP. Later, we decided to manually check if the computer utilizes the NVIDIA driver to support the installation. It turned out that the GPU was not utilized for this process at all. This question was examined, and it was found that it was required to install the drivers for GPUs manually on Linux(Ubuntu), unlike how it is done in Windows(everything is already working from the package). Therefore, we started trying every driver for our GPU, but due to internal package clash inside Ubuntu (as we assumed), the drivers were not compatible. At that stage, it was decided that reinstalling Ubuntu would be a faster way to solve the problem, rather than trying to find an internal clash of the packages. After the reinstallation of the fresh Ubuntu, all required ROS packages, it was possible without any major technical problems to launch the RTAB MAP.

8. **Gazebo End-of-Life:** The initial setup of RTABMap presented significant challenges. First, we had to reinstall Linux on our laboratory computer to correctly configure GPU drivers. Following successful driver installation, we focused on understanding and correctly configuring the depth camera plugin within Gazebo. Using our customized launch file, we successfully generated initial results, including detailed 3D mapping data and a derived 2D map suitable for navigation purposes.

   However, during our development and simulation phases, an unexpected obstacle emerged. Gazebo Classic, which had reached its end-of-life status in January of this year, abruptly ceased functioning towards the end of our semester. Consequently, we were unable to continue running simulations in our established environment. Forced by this development, we transitioned to Gazebo Harmonic.

   Migrating to Gazebo Harmonic introduced new complexities. Significant refactoring was required for our launch and URDF files to ensure compatibility with the Harmonic and Fortress versions of Gazebo. Further complications arose from GPU compatibility issues. Gazebo Harmonic demanded precise GPU settings, leading us to spend considerable time troubleshooting this aspect. Unfortunately, initial attempts resulted in unsuccessful simulations.

   To resolve these technical challenges, we performed a complete reinstallation of the Nvidia GPU drivers and adjusted the MESA settings meticulously. Despite these extensive efforts, achieving stable and functional simulations in Gazebo Harmonic remains an ongoing task that we continue to address actively.

9. **Limited Computational Resources on Raspberry Pi:** The Raspberry Pi struggled to handle a full ROS 2 environment due to its limited CPU and memory. We were forced to forgo a traditional ROS 2 node-based architecture on the Pi and instead implement lightweight, standalone Python scripts.

10. **Communication Between the Control Laptop and Raspberry Pi:** During the initial testing phase of the communication, one of the primary challenges was the lack of observable output on either the control laptop or the Raspberry Pi, making it difficult to identify the source of the issue. It was initially unclear whether the laptop was successfully transmitting data or if the Raspberry Pi was receiving and processing it. This ambiguity necessitated extensive step-by-step debugging of both the talker and listener scripts. The root cause was eventually traced to the listener script running on the Raspberry Pi. Specifically, the issue stemmed from the improper parsing of incoming UDP data, which prevented correct translation and subsequent transmission of commands to the wheelchair's motor controller via the CAN bus.

11. **Inverse Kinematics and Velocity Scaling:** To ensure accurate transmission of motion commands to the wheelchair's motors, it was essential to correctly interpret and scale the velocity data received from the teleoperation interface. Since the raw velocity values generated by the teleop node do not directly correspond to the physical motion parameters of the wheelchair, a scaling adjustment was required. Addressing

this challenge necessitated an understanding of the system's inverse kinematics. Fortunately, the wheelchair operates on a differential drive mechanism, which simplifies the kinematic model. By reducing the relevant parameters and deriving the appropriate relationships between linear and angular velocities and individual wheel speeds, we were able to implement a transformation that maps Twist commands to motor-specific instructions. This enabled the accurate encoding and transmission of velocity commands via the CAN bus to the motor controllers.

12. **Challenges in Teleoperation-Based Control:** Despite successfully identifying the appropriate scaling factors for the Twist messages transmitted from the control laptop, difficulties persisted in achieving accurate directional control of the wheelchair through the teleoperation interface. Specifically, the mapping between keyboard inputs and the resulting linear and angular velocity commands was unclear, which led to inconsistent or unintended robot behavior. This challenge highlighted the need for a deeper understanding of differential drive kinematics and the typical motion behavior of mobile robots. To address this, we conducted a series of simulations using packages such as turtlesim and TurtleBot, which provided valuable insights into velocity command interpretation and movement execution in simpler systems. These experiments ultimately informed improvements to our teleop configuration.

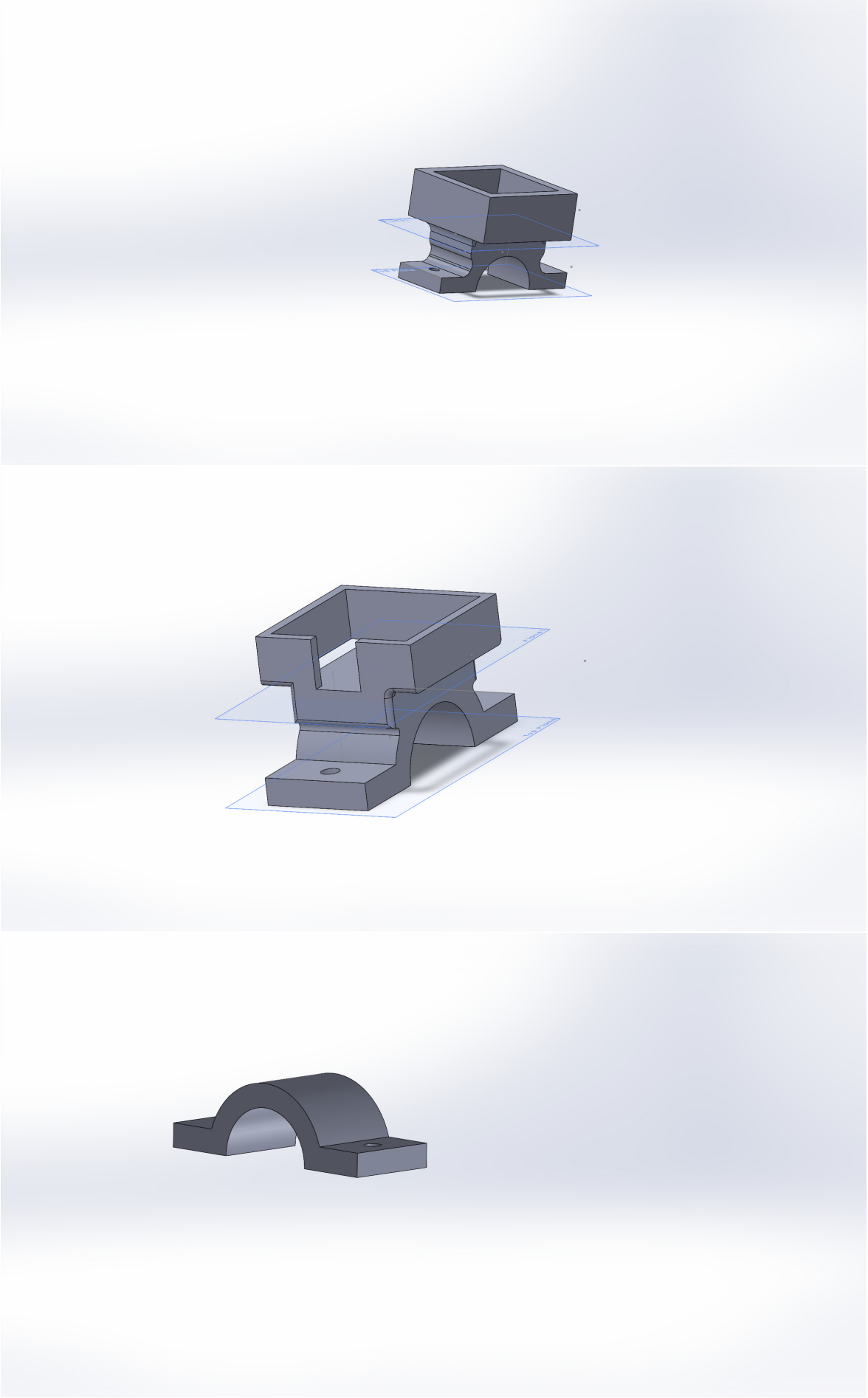


Figure 5.2: First CAD designs of the (a) camera holder, (b) IMU sensor holder, (c) bottom part.

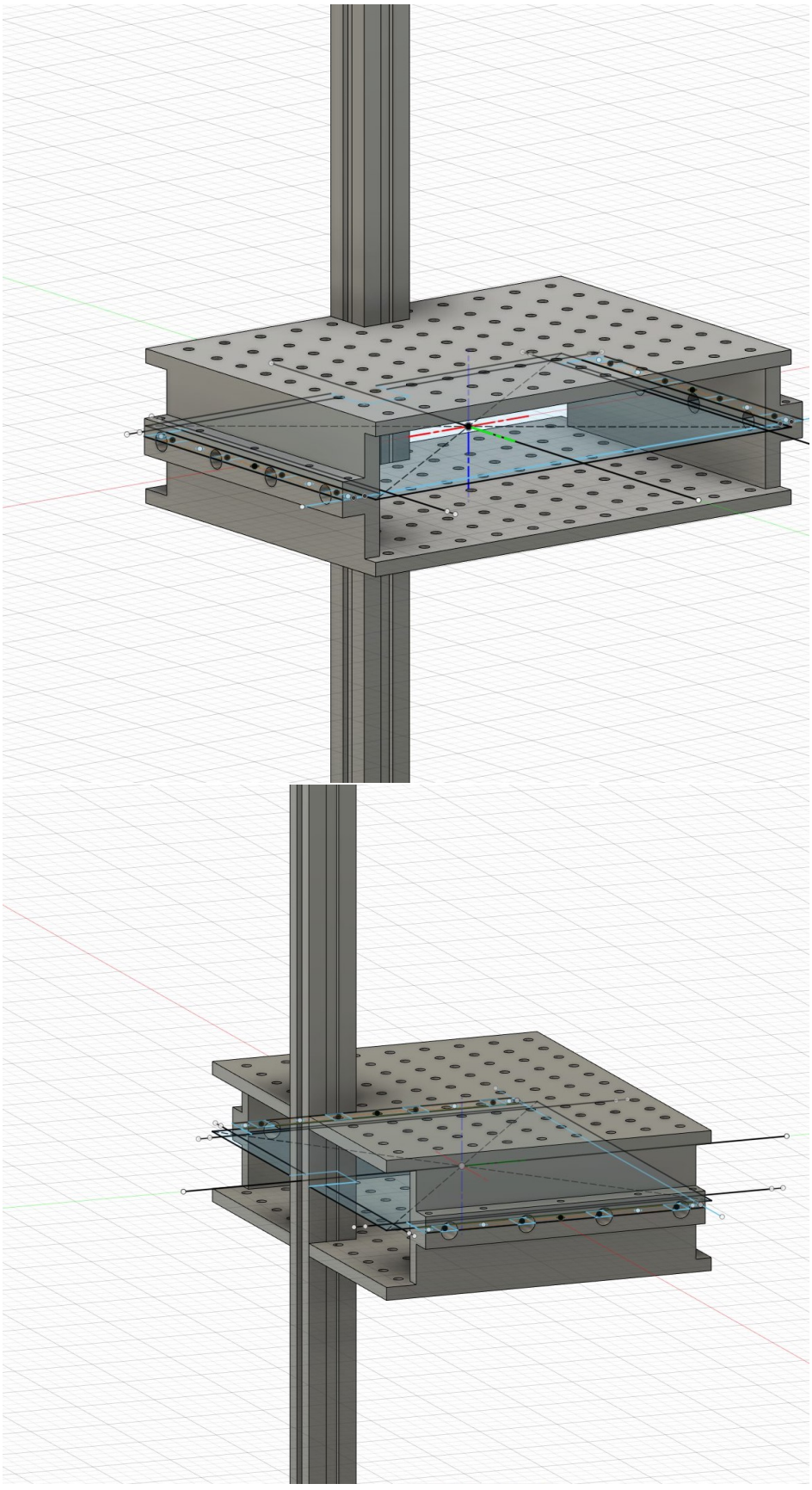


Figure 5.3: The most recent CAD design of the camera holder(1 and 2 point of views).

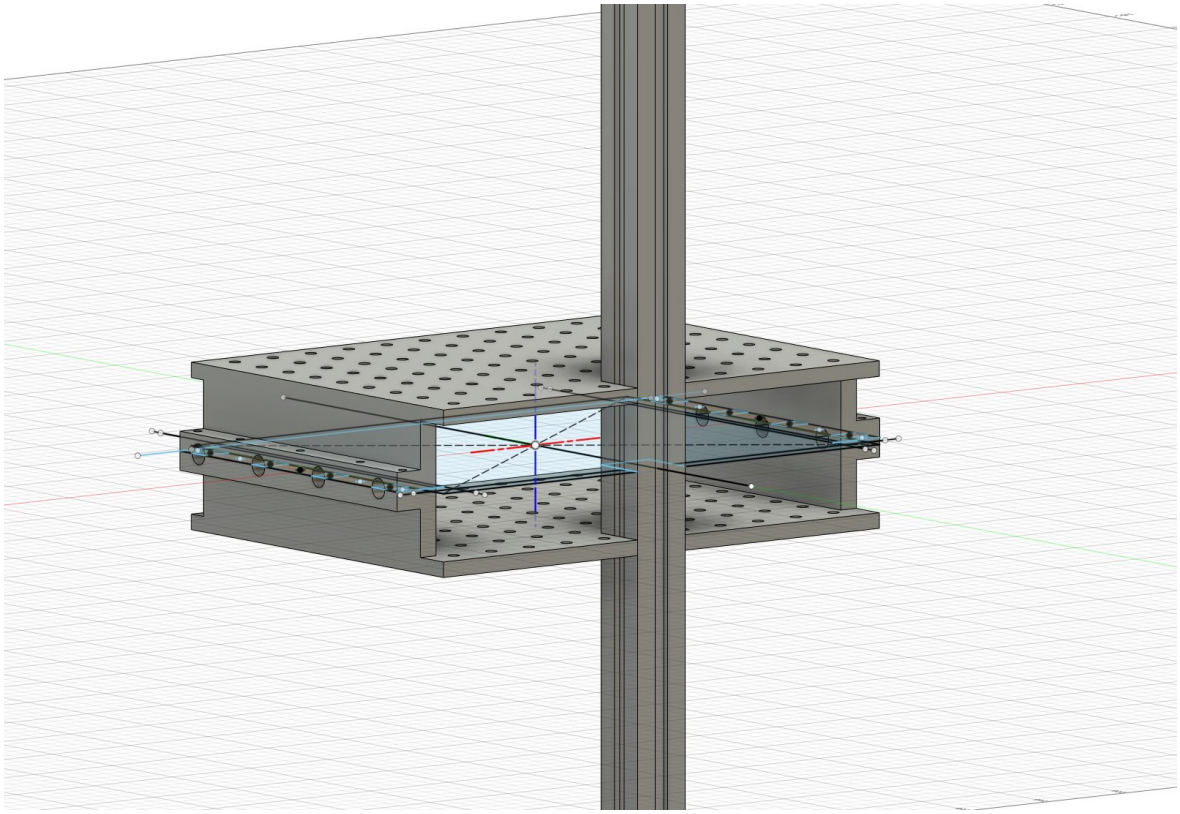


Figure 5.4: The most recent CAD design of the camera holder(3 point of view).

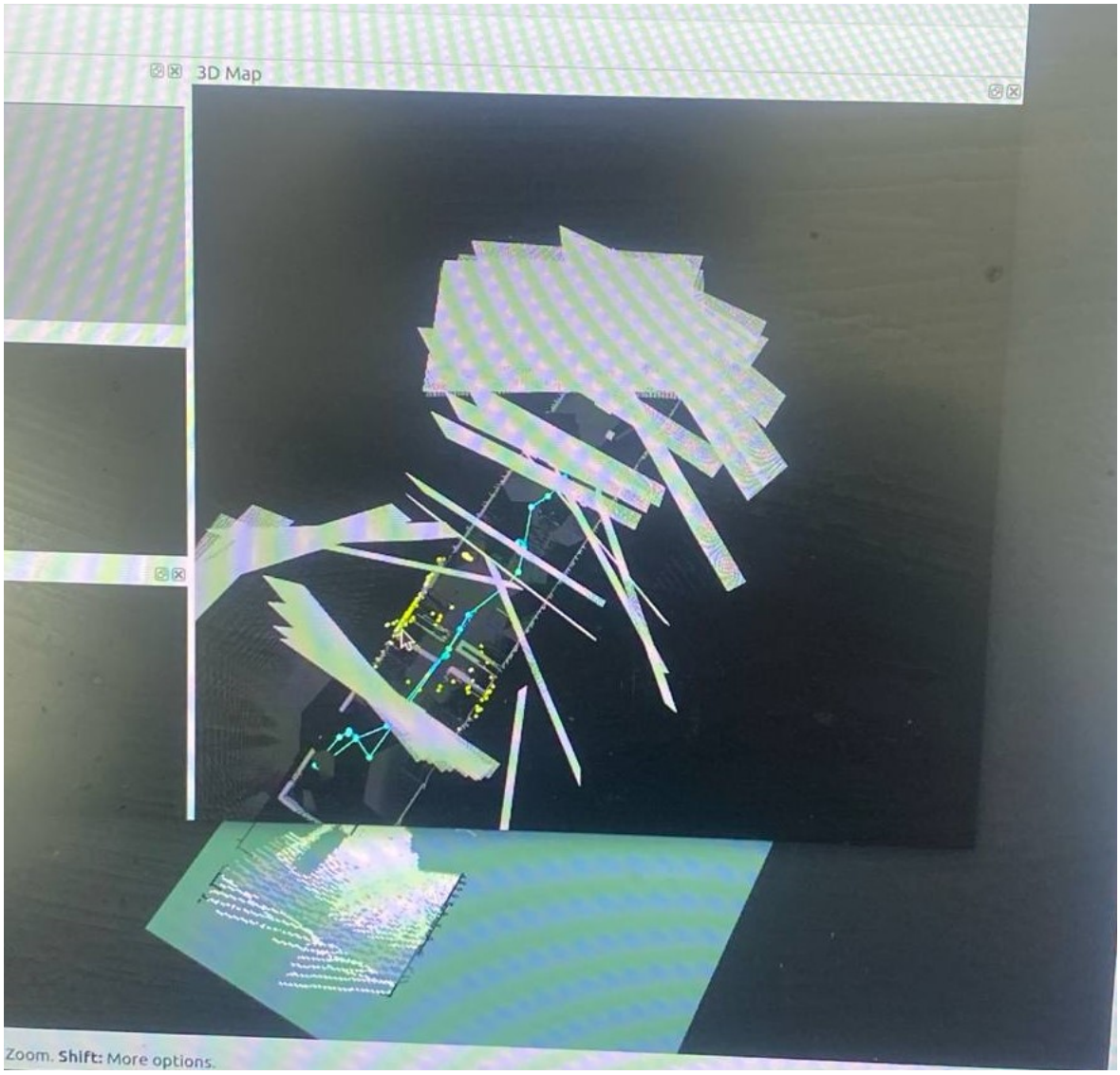


Figure 5.5: Simulation of RTABmap.

Figure 5.6: Costmap.

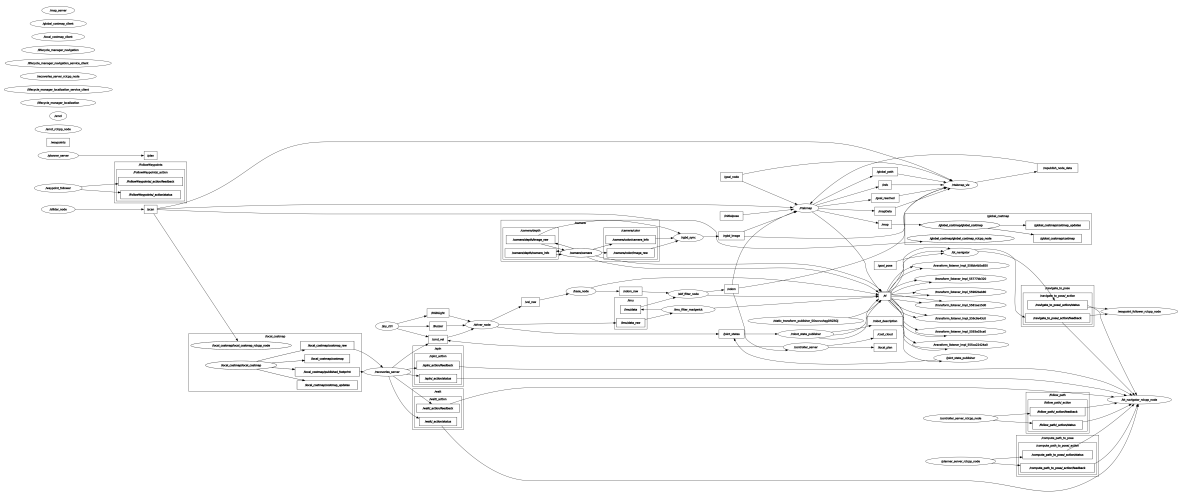


Figure 5.7: The interconnection of ROS topics of the project.

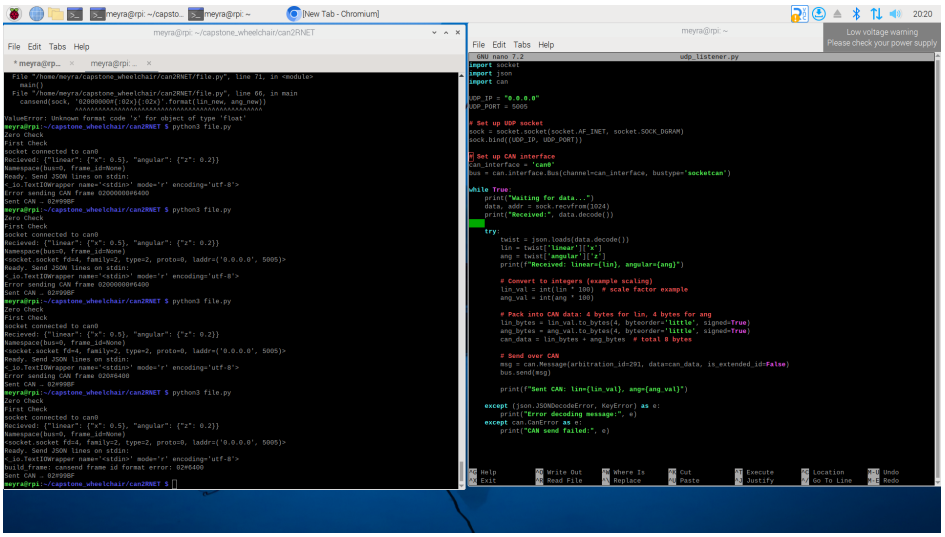


Figure 5.8: Listening to Twists' commands sent from the laptop to the Raspberry Pi

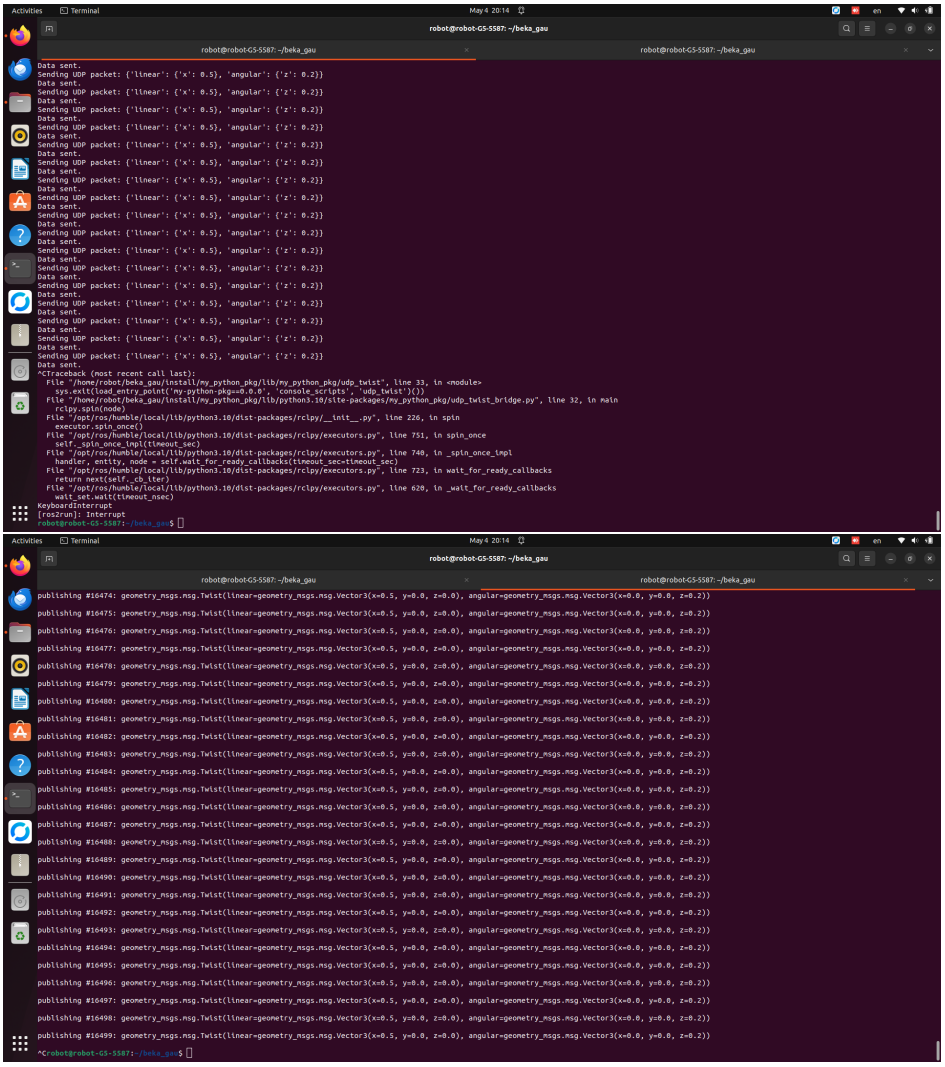


Figure 5.9: Sending the twist messages with specified values of linear and angular velocities via the UDP

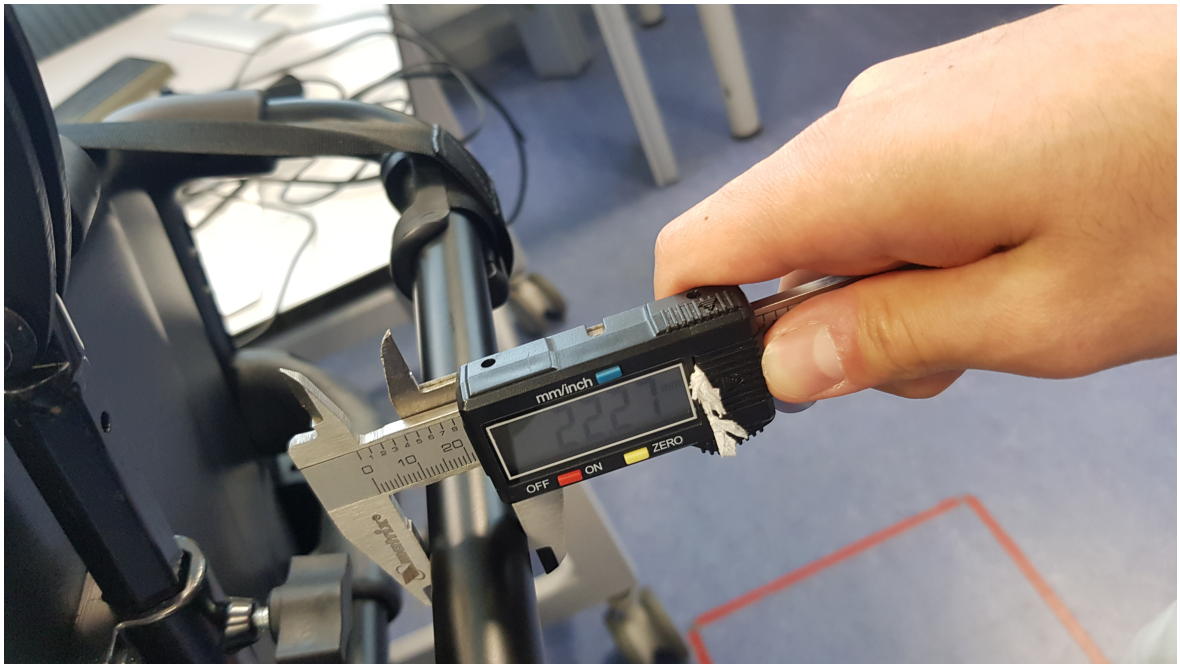


Figure 5.10: Initial fix point for ZED2 camera and IMU holder.

Chapter 6: **Conclusion & Future Development**

The primary objective of this project was to transform a conventional power wheelchair into a fully autonomous robotic system. The core milestones of the project were successfully achieved, including the establishment of wireless communication between the control computer and the wheelchair, the development of a URDF model, the implementation of SLAM in a simulated environment, and the execution of basic navigation tasks with a real wheelchair using the ROS 2 teleoperation interface.

To further advance the capabilities of the system, several enhancements can be pursued in future iterations of the project, including:

- Migrate the whole package to Gazebo Harmonic, so that every library, plugin, driver, and dependency would properly work;
- Test out the Nav2 stack on the maps that we acquired from the RTAB MAP simulations;
- Harmonize the navigation for the real-world wheelchair that we have in the laboratory, since we had very rough estimations during the simulation process;
- Optimize the controls of the wheelchair through the CAN BUS, so that the data would accurately fit the real-world environment(i.e., scale the transmitted Twist messages from the laptop);
- Using more sensors to acquire more precise data, i.e., establishing another depth camera, or incorporating more expensive, harder to implement, but more accurate LiDAR, which can be installed under the seat of the wheelchair.
- Using another SLAM algorithm. During this project, we stick to the simple, already provided solutions from ROS. However, other open-source algorithms specific for solving these tasks can be applied. It can also be achieved by writing a proprietary algorithm specifically for our project, which would utilize all the potential of the sensors and the wheelchair.
- Switching to **Nvidia Jetson:** Since Raspberry Pi has its computational power limitations, we would need to consider switching to Nvidia Jetson for faster processing.

Chapter 7: **Learning Portfolio**

Key Learning Experiences:

Throughout the development of the autonomous wheelchair, our team encountered a variety of technical and non-technical challenges that required continuous learning and adaptation. Since our capstone topic is a development project, it requires all the topics we have covered so far and additional knowledge on them. Until this point, we have covered such concepts as Signal Processing, System Dynamics, Linear Control Theory, Mechanical Design, working with ROS1, and so forth, which are essential for every roboticist and are foundational skills to enter any serious engineering project. However, after understanding the approximate goal of the project, we concluded that we will have to acquire new skills in more specific areas such as odometry, ROS Navigation package, ROS2, programming microcontrollers(Raspberry Pi), working with sensors(IMU and depth camera) and maybe even soldering.

Thus, it is clear that although we have already some theoretical knowledge of robotics, our project requires more specific expertise and practice. It should be noted that similar projects have already been done before, and there are already some existing commercialized ideas like this. Thus, the knowledge of how to build such projects already exists, there are a few sources. The concepts that were mentioned above, such as odometry, have been established ages ago, and it is open-source. Therefore, what our team had to do(and will continue) was just to learn these solutions.

One of the major learning experiences was integrating the Raspberry Pi with the wheelchair's control system using CAN-bus. None of us had prior experience with CAN-bus communication protocols, so we had to familiarize ourselves with how the Raspberry Pi communicates with the wheelchair's motor control system. Through independent learning and online tutorials, we successfully established this communication, using a CAN2RNET library found on GitHub.

In terms of hardware, setting up the ZED2 depth camera and UM7-LT IMU sensor required in-depth knowledge of sensor integration and data processing. The hands-on experience of physically mounting and calibrating the sensors taught us lessons in sensor noise management, particularly in ensuring accurate data acquisition for mapping and obstacle avoidance. We learned how to securely mount these components and realized how critical stable sensor placement is for reducing errors.

From a project management perspective, managing time across team members was another learning experience. With each of us having different strengths and time schedules, dividing tasks such as hardware setup, software integration, and documentation required constant communication and clear timelines.

Therefore, if we initially started just with the theoretical knowledge of the aforementioned concepts and a little practical experience, now we are on the way to mastering concepts of odometry, ROS2, working with microcontrollers, and mechanical design.

Skills Gained:

Technical Skills:

- **CAD Design:** we had to come up with a few designs that would allow the sensors to

be fixed firmly on some surface for correct data transmission. That made our "plan → build → try again" skills more refined.

- **CAN-bus Communication:** understanding and implementing real-time control of the wheelchair through Raspberry Pi using CAN-bus significantly enhanced our system integration skills.
- **Odometry:** We learned the importance of combining data from encoders and IMUs to improve odometry.

Tech skills to be learned until the end of the term:

- **Sensor Integration:** We have to learn how to interface sensors (ZED2 camera, IMU) with the control system and calibrate them for real-world usage, improving our understanding of how sensors interact in a robotics system.
- **ROS2:** We need to gain proficiency in working with ROS2 for handling the control stack and RTAB-Map for implementing SLAM. This required not just technical understanding, but also the ability to troubleshoot issues related to sensor fusion and map drift.

Soft skills:

- **Teamwork and Communication:** Coordinating tasks between team members and dividing responsibilities based on individual strengths was essential.
- **Time Management:** Meeting deadlines for each phase of the project required careful planning and prioritization. We developed Gantt charts to manage our progress and ensure that we stayed on track with our timeline.
- **Problem-solving:** With every technical hurdle (like integrating CAN2RNET protocol), we honed our ability to research, test, and iterate until we found solutions.

It should be noted that time management skills are the most important during the development of this project. Since it is an extensive project allocated for 2 semesters and it does not have a straight path of completion, it is essential to allocate time weekly to come work in the laboratory with the equipment.

Future Learning Goals:

While we've made some progress, there are many more areas where further learning is necessary:

- **Advanced Path Planning and Obstacle Avoidance:** while we've been learning the basic ROS2 Navigation Stack, we need to dive deeper into optimizing path planning algorithms, particularly in dynamic environments. Learning about advanced motion planning algorithms would be beneficial for improving obstacle avoidance in real-time scenarios.
- **Machine Learning(Computer Vision) for Object Detection:** Integrating machine learning techniques for object recognition (using the ZED2 camera) could improve how the wheelchair perceives its surroundings. Exploring frameworks like TensorFlow or PyTorch for object classification would be a valuable skill for future enhancements.

- **Knowledge of SLAM:** Since we are required to analyze the surroundings of the robot to implement path planning, it would be essential for us to implement a basic SLAM setup. After that more advanced approaches, such as Graph-based SLAM or incorporating loop closure detection could be implemented, which could improve the accuracy and robustness of the wheelchair's navigation system. Understanding more about these techniques would be beneficial for future projects that involve autonomous systems.

Through this capstone project, we've developed a range of technical skills, from ROS2 to sensor integration, alongside important soft skills like teamwork and time management. Moving forward, we aim to enhance our knowledge in advanced path planning, machine learning, and project management to further our capabilities in the field of robotics and autonomous systems.

Bibliography

[1] R. Dawson, "Knowledge capabilities as the focus of organisational development and strategy," *Journal of Knowledge Management*, vol. 4, no. 4, pp. 320–327, 2000.

[2] R. Dragon, "can2rnet," 2018.

[3] Unknown, "Using robot operating system for autonomous control of robots in eurobot, erc and robotour competitions." `https://www.researchgate.net/figure/Standard-ROS-Navigation-Stack-7_fig2_312018119`, 2024.

[4] Unknown, "Navigation2 documentation." `https://docs.nav2.org/`, 2024.

[5] S. Günther, "Urdf tutorial - robot operating system." `https://admantium.com/blog/ros04_urdf_tutorial_1/`, 2024.

[6] L. Devigne, M. Aggravi, M. Bivaud, N. Balix, C. S. Teodorescu, T. Carlson, T. Spreters, C. Pacchierotti, and M. Babel, "Power wheelchair navigation assistance using wearable vibrotactile haptics," *IEEE Transactions on Haptics*, vol. 13, no. 1, pp. 52–57, 2020.

[7] F. Morbidi, L. Devigne, C. S. Teodorescu, B. Fraudet, Leblong, T. Carlson, M. Babel, G. Caron, S. Delmas, F. Pasteau, *et al.*, "Assistive robotic technologies for next-generation smart wheelchairs: Codesign and modularity to improve users' quality of life," *IEEE Robotics & Automation Magazine*, vol. 28, no. 3, pp. 12–28, 2022.

[8] X. Zhang, Z. Song, Q. Huang, Z. Pan, W. Li, R. Gong, and B. Zhao, "Shared ehmi: Bridging human–machine understanding in autonomous wheelchair navigation," *Applied Sciences*, vol. 14, no. 463, 2024.

[9] B. Amangeldiyev, "wheelchair_control_xbox." https://drive.google.com/file/d/1LMwFCe08kYGOsAg49tT1KvfQ3PnvskRI/view?usp=sharing, 2024.

[10] B. Amangeldiyev, "wheelchair_sounds." https://drive.google.com/file/d/1LFyQUpvYu3jK7kkyeU58YJ1giZXzd_Vm/view?usp=sharing, 2024.

[11] Stereolabs, "Zed ros2 wrapper." `https://github.com/stereolabs/zed-ros2-wrapper`, 2024.

[12] ROS-Drivers, "Issue #21: Um7 imu gives incorrect roll/pitch reading when flipped." `https://github.com/ros-drivers/um7/issues/21`, 2024.

[13] "Rtab-map: Real-time appearance-based mapping," n.d.

[14] "Rtab-map ros," n.d.

[15] ROS, "Urdf: Unified robot description format," n.d.

[1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15]