AMERICAN UNIVERSITY OF BEIRUT

Maroun Semaan Faculty of Engineering and Architecture
Department of Mechanical Engineering

MECH 501 Final Year Project
Group Progress Report
Spring 2024-2025

Project Title:
Developing Robot SHARE-C's Full Autonomy

by

Andrea Tarabay	*Electrical and Computer Engineering*
Jade El Masri	*Electrical and Computer Engineering*
Qasem Dib	*Mechanical Engineering*
Aya Abed	*Mechanical Engineering*
Rayan Abdul Samad El Skaff	*Electrical and Computer Engineering*

Advisor: Dr. Naseem Daher,
Electrical and Computer Engineering, Mechanical Engineering

17/4/2024

A Final Year Project
submitted in partial fulfillment of the requirements
for the degree of Bachelor of Engineering
to the Department of Mechanical Engineering
at the American University of Beirut

Beirut, Lebanon

Executive Summary

In response to the significant challenges healthcare workers face, particularly during infectious outbreaks like COVID-19, this project aims to enhance hospital safety and efficiency by advancing the autonomy of SHARE-C, a Social Healthcare Assistive Robot with a Lebanese ethno-cultural identity. Originally designed to assist medical staff and provide companionship to patients, SHARE-C utilizes a human-centered design approach to address issues such as infection control and workload distribution. This report details the ongoing work on SHARE-C, emphasizing its autonomy development, critical system components, and validation strategies.

The progress report outlines significant developments across several project components. Perception and sensing capabilities are being enhanced using the RTAB-Map SLAM algorithm in combination with an Intel RealSense camera D435i, with simulations conducted in Gazebo to refine navigation performance before implementation on SHARE-C. Path planning combines A* and Dynamic Window Approach (DWA) algorithms for global and local path planning, respectively, while move_base manages motion planning. These approaches are complemented by low-level controllers for improved performance in dynamic environments.

For locomotion, a two-wheel differential drive system is being designed and executed to ensure reliable mobility. Additionally, a proof of concept for an automated door-opening mechanism is being developed in simulations and experiments using the LoCoBot. Experimental validation will be performed with a compatible manufactured door and a handle. This involves designing and validating path planning and control algorithms for effective interaction with push/pull doors. Finally, a graphical user interface (GUI) is being developed to facilitate communication and control between users and the robot, with preliminary designs implemented on platforms such as Figma, Visual Studio Code, and Android Studio.

Acknowledgements

Acknowledgment is given to Dr. Naseem Daher for his continuous support during these challenging times. His incredible motivation helps drive the project forward and inspires perseverance. Dr. Daher provided valuable insights and constructive comments regarding the project which helped us immensely and gave us a drive to keep going forward.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In a previous study, SHARE-C, a Social Healthcare Assistive Robot with a Lebanese Ethno-Cultural Identity, was designed as an assistive social healthcare robot. Its primary role is to address the high risks present in hospital environments, particularly by utilizing a self-disinfecting system to assist with immunocompromised or highly contagious patients. SHARE-C serves as both an assistive delivery robot for medical staff and a social companion for patients, as it was designed, manufactured, and developed following a Human-Centered Design methodology. With its mechanical components—such as the body, arms, head, face, and locomotion—already implemented, this project focuses on developing SHARE-C's full autonomy to meet its intended performance specifications.

1.1 Motivation

Healthcare systems struggle with many hurdles when dealing with critical cases such as immunocompromised or highly contagious patients. Nurses would have to ensure strict infection control while balancing limited time and resources for patient care. Nurses should not only care for the highly contagious patients but at the same time protect themselves and other patients from getting infected.

In 2020, COVID-19 highlighted these vulnerabilities risking not only the patients but also healthcare workers. Statistics reveal that frontline workers like nurses are at a very high risk of exposure to infectious diseases in such cases. Around 1.6 million healthcare workers in 34 countries, as reported by the International Council of Nurses (ICN), were infected by COVID-19 by 31st December 2020 [1]. Not only that but according to WHO, 115,000 healthcare workers lost their lives due to COVID-19 between January 2020 and May 2021 [2]. In outbreaks like COVID-19, hospitals also face shortages in Personal Protective Equipment (PPE) which further increases the threat of nurses' exposure to infectious diseases when interacting with highly contagious patients. During COVID-19, due to shortages in masks and N95 respirators, approximately 87% of nurses reported they wore their supposed to be “single” use masks more than once [3]. National Nurses United (NNE) reported that 27% of nurses have been exposed to patients without even wearing any form of PPE [4]. In addition to that, the risk on nurses’ mental health increases when nurse-to-patient ratios and working periods in isolated hospital floors also increase. This leads to increased nurses’ burnout discouraging them as they try their best to fulfill their mission to care for others.

Nurses face a lot of challenges with wearing PPE for long working hours. Due to outbreaks, many nurses would have to work for more than eight consecutive hours daily which would lead to several physical discomforts [5]. Even though it protects them from infectious pathogens, it increases the possibility of getting ‘heat exhaustion’ and ‘cognitive impairment’ [6], [7]. Also, the process of wearing the PPE, called donning, takes around 10-15 minutes since it requires precise and accurate sterilization to secure safety [8]. This not only makes it tiresome for the healthcare workers but it also leads to longer waiting times by the patients. Patients would have to wait due to the delayed assistance from nurses, especially when there is an imbalanced nurse-to-patient ratio during outbreaks.

When nurses catch infectious pathogens, this does not only affects them, but they also become carriers of these pathogens to other vulnerable patients. This is called Healthcare Associated Infections (HAIs) which happens when patients catch these infections during their stay at the hospital. According to the Centers for Disease Control and Prevention (CDC), about 1 in 31 hospital patients become victims of HAIs [9]. Immunocompromised patients are the most affected in such situations.

In light of these challenges, the deployment of autonomous social robots in hospital settings offers a promising solution. These robots present two main characteristic qualities: ‘Social’ robots serve humans as companions, communicating and interacting with humans and other social robots based on social behaviors and rules [10]. Due to their human-like appearance and application in an area close to human emotions, social robots promote positive emotions, whether they are operating in educational or companionship settings. However, negative emotional reactions were exhibited when the robot was tasked with surgery assistance within healthcare conditions [11]. For this reason, a social robot adhering to well-intentioned social norms and serving as an assistant or companion in healthcare settings, with no critical surgery or medical tasks, is acceptable among patients. The second characteristic is the ‘service’ quality. These robots conduct assistive tasks like delivery, and in a hospital setting, they are useful in minimizing infection risks for both patients and medical workers as well as labor and time for medical workers. Robots exhibiting both qualities, being social and performing service tasks, are an efficient solution to our problem as described in a previous work.

1.2 Desired Needs:

- **User Interface:** Develop a GUI for medical staff to schedule tasks, monitor real-time location/status, and enhance communication (facial expressions, audio).
- **Autonomous Navigation:** Implement vSLAM for mapping and localization, integrating robust path planning and obstacle-aware motion planning.
- **Locomotion System:** Replace LoCoBot's limited traction with a wheeled system featuring electric motors and differential drive.
- **Self-Disinfecting:** Upgrade the sterilization mechanism for effective disinfection before and after patient interactions.
- **Door Opening:** Design an end effector and control system to autonomously open doors at AUBMC using sensors and real-time adjustments.

Chapter 2

Requirements and Deliverables

2.1 Requirements and Specifications

The main aim of this project is to implement autonomous navigation in a hospital, and this requires several critical tasks designed to accomplish a successful movement through a complex floor plan between rooms and the nurse station. This chapter presents an outline of the requirements, specifications, and deliverables that are indispensable for achieving the goal of this project.

1. To begin with, the first member is required to achieve sensing and perception as it is a fundamental requirement for the successful autonomous navigation of the robot. By installing multiple suitable sensors such as an Inertial Measurement Unit (IMU), Ultrasonic sensors, and cameras, real-time data is gathered to obtain information about the robot's surroundings. Thus, Visual Simultaneous Localization and Mapping (vSLAM) must be developed to help update the robot's map based on visual feedback while also being able to determine its exact location.

2. The second requirement is planning and control which is equally important to the previous task of guiding the robot in a hospital. This includes implementing algorithms for path planning, motion planning, and trajectory generation. Path planning is the process of defining waypoints that the robot should follow, while trajectory generation ensures safe navigation between these points. Motion planning accounts for potential obstacles, whether static or dynamic, all while coordinating the translational and rotational velocities of the robot. In addition, to ensure that the robot's motion aligns with the planned motion, it is required to design low-level controllers to have accurate steering, acceleration, and braking.

3. The third critical feature for this robot nurse is developing an efficient self-disinfection system to prevent the spread of contaminants as the robot

navigates within the hospital. This task includes choosing the specifications of components such as pump flow rate, tank size, hoses, and gears. After procuring and assembling the components, rigorous testing should be done to validate operational efficiency before implementing the disinfection system in the cavity of the robot.

4. Also, a new wheeled locomotion system must be designed and manufactured, and this base must incorporate electric motors to have differential drive capabilities and achieve the desired traction and stability. Previously, a LoCoBot platform was used, and it wasn't efficient as it had a limited payload capacity and traction. Thus, given that the robot will be operating in a hospital environment, it is required to have improved movement.

5. Additionally, for the robot to be able to access the rooms in the hospital, an efficient door-opening mechanism must be implemented. This task requires investigating the different types of end effectors to be suitable for the doors at the AUBMC. Available options are grippers and manipulators, designed to work with a specific door type (push/pull, swing, sliding...). One configuration of the door and handle type is agreed upon. Also, to achieve door detection, alignment, and force feedback, multiple sensors should be used such as LIDAR, camera, radar, and depth sensors. Furthermore, path planning and motion control algorithms are integrated to ensure a smooth robot operation of opening the door, along with a closed-loop control system designed to adjust actions based on input from the sensors. This ensures a safe and effective process for handling the opening the door mechanism.

6. For the last task, it is required to have a Graphical User Interface (GUI) that enables nurses to interact with the robot and place their orders, all while monitoring the status of the robot. Thus, a user-friendly GUI must be designed and implemented with a database system for logging in nurses and handling multiple task scheduling. Furthermore, this UI must also include visual feedback of the robot's real-time live location. Lastly, to ameliorate the robot's acceptance by the medical staff and patients, the facial features and audio outputs of the robots should be improved to create a successful interaction system.

2.2 Deliverables

The deliverables for this project entail the following:

1. Developing a comprehensive navigation and mapping system based on a vSLAM algorithm to enable real-time localization and mapping.

2. Developing a motion control system that encompasses path planning and trajectory generation to ensure the safe navigation of the robot.
3. A self-disinfection system mechanism with a compact robot base based on a differential drive system will be assembled and fully tested.
4. For the robot to enter a room smoothly, a door-opening mechanism should be fitted for operation and verified.
5. Building a GUI to provide an interface for the medical staff, enabling them to place and monitor orders while ensuring user-friendly communication features.

Chapter 3

Technical/ Non-Technical Constraints and Applicable Standards

3.1 Technical Constraints

The technical constraints are defined as specifications and limitations that must be adhered to during the design and development of the locomotion system for the nurse robot. These include:

1. **Payload Capacity**: The locomotion system must support the combined weight of the robot and its payload, ensuring adequate strength and rigidity in all components.

2. **Environmental Adaptability:** The robot must be able to navigate obstacles up to 10 cm in height and fit through doors with a minimum width of 80 cm.

3. **Precision and Accuracy:** Task execution accuracy should be within ±3 mm.

4. **Autonomous Navigation:** vSLAM accuracy should be within 5 cm of the actual location; obstacle detection should be within 1 meter.

5. **Battery Duration:** Minimum of 8 hours of operational time on a single charge. As for power management, the target recharge time should be under 2 hours.

6. **Real-Time Processing:** Data processing latency should be less than 100 milliseconds.

7. **Error Handling:** System should recover from faults within 10 seconds or less.

8. **Patient Safety:** Ensure all surfaces are smooth and free of sharp edges; follow ISO 13485 for medical device quality management.

9. **Wheel-Spring Interaction**: Springs must absorb vibrations and impacts from uneven surfaces without affecting wheel alignment or traction.

10. **Compact Design**: The entire locomotion system must fit within the specified dimensions of the robot's body while maintaining functional accessibility for repair or maintenance.

11. **Door and Handle Types**: We assume lever door handles that allow for both push and pull mechanisms.

12. **Automatic Door Closing**: Hospital doors are expected to close automatically, removing the need for the robot to close the door after entry and aligning with infection control standards.

13. **Force Estimation on Arm**: Due to the high cost of force/torque sensors, the force can be estimated using back electromotive force (EMF) as an alternative.

14. **Arm Mechanism**: The door-opening is implemented as a proof of concept, using the LoCoBot and a small constructed door for testing instead of a full-scale arm integrated with SHARE-C. The Robot Operating System (ROS) framework serves as the software infrastructure for the project.

3.2 Non-Technical Constraints

1. **Ethical Considerations:** The robot must ensure ethical use to protect the privacy of patients and not record their personal data.

2. **Cost and Budget:** The team must stay within the budget constraints imposed by the departments for purchasing equipment.

3. **Acceptance and Usability:** The final design must ensure ease of use and acceptance by healthcare staff and patients.

4. **Cultural and Social Impact:** The team should design the robot with cultural sensitivity and consider social acceptance.

5. **Patient and Staff Safety:** The team should assess and address potential health risks posed by the robot and develop emergency response protocols.

3.3 Applicable Standards

1. **ISO 13485**: This standard governs the quality management systems for medical devices. Our design process for SHARE-C's autonomous systems adhered to this standard by establishing systematic quality checks and risk management procedures.

2. **ISO/IEC 62366** : Focused on usability engineering for medical devices, this standard guided the design of SHARE-C's user interfaces, ensuring they are intuitive and safe for hospital staff.

3. **ISO 13482:** This safety standard for personal care robots influenced the development of safety features and emergency protocols in SHARE-C, ensuring patient and operator safety during interactions.

4. **ISO 14971:** We applied this standard's risk management principles extensively, especially in assessing and mitigating risks associated with autonomous navigation and patient interaction functionalities.

5. **ASME Y14.5:** This standard provides essential guidelines for mechanical design specifications and tolerances. We adopted its principles to ensure all components of SHARE-C meet precise dimensional requirements and tolerances, crucial for the modular assembly and maintenance of the robot.

6. **ACI 117:** *Specifications for Tolerances for Concrete Construction and Materials* provides limits for the flatness and levelness of concrete surfaces.

a. Flatness tolerance: Deviations within **3-6 mm** over a 3-meter length are generally acceptable for most applications.

b. Levelness tolerance: Deviations of **6-12 mm** over the entire slab may be acceptable, depending on the function.

Chapter 4

Technical Background

4.1 Sensing and Perception

4.1.1 Research: Existing Solutions and Efforts

Overview of SLAM

SLAM, or Simultaneous Localization and Mapping, is an important component for the navigation of autonomous robots, as it answers two key questions: 'Where am I?' and 'What are my surroundings like?' This means that the robot estimates its position and orientation based on sensor data without any prior knowledge and can construct a map of the surrounding environment in real-time [12]. As the robot navigates in its environment, it gathers data from sensors. There are two types of sensors used: exteroceptive sensors which measure externally (e.g., cameras, LiDAR), and interoceptive sensors which measure internally (e.g., IMUs). Then, the algorithm identifies unique points or landmarks by extracting features from consecutive sensor data. This is called visual odometry. The detected features across consecutive frames are then matched to help in tracking the robots' motion relative to previously observed landmarks. Sensor data is often noisy and not accurate. Therefore, to make sense of this data, state estimation uses mathematical techniques to filter out noise and relate information to predict the robot's accurate position. After updating the map, the robot identifies previously visited locations and uses this information to correct accumulated drifts to improve localization accuracy. It then performs Pose Graph Optimization to refine trajectory and the mapping by adjusting the robot's poses and positions to minimize overall error in the graph [13]. Check SLAM Framework in Figure 4.1

Visual SLAM

Visual SLAM (vSLAM) is a type of SLAM that uses visual sensors like cameras to perform localization and mapping. A visual sensor like a camera can be either

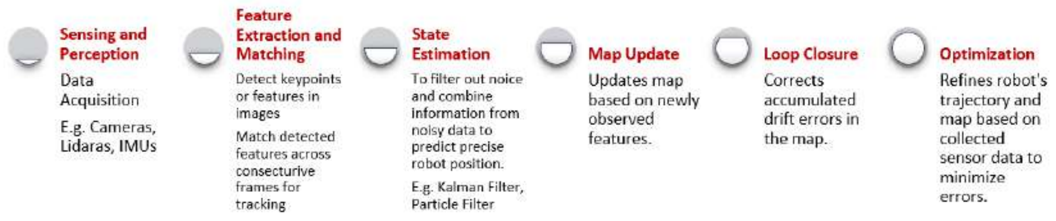

Sensing and Perception
Data Acquisition
E.g. Cameras, Lidaras, IMUs
Feature Extraction and Matching
Detect keypoints or features in images
Match detected features across consecturive frames for tracking
State Estimation
To filter out noice and combine information from noisy data to predict precise robot position.
E.g. Kalman Filter, Particle Filter
Map Update
Updates map based on newly observed features.
Loop Closure
Corrects accumulated drift errors in the map.
Optimization
Refines robot's trajectory and map based on collected sensor data to minimize errors.

Figure 4.1: SLAM Framework

monocular (single camera), Stereo (double camera), or RGB-D vSLAM (depth camera). Visual SLAM algorithms are categorized into sparse and dense methods based on the level of detail in the map. Sparse methods use only a small subset of the pixels in an image frame where maps are generated into point clouds. It is less computationally expensive for real-time applications as it focuses on specific key points to represent the scene. However, dense methods use most or all of the pixels in each received frame. Dense methods generate 3D maps that are more detailed but these methods are typically more computationally expensive. See Table 4.1.

Method	**Description**	**Advantages**	**Disadvantages**
Sparse Methods	Use key points from images to generate point cloud maps.	Computationally efficient; ideal for real-time applications.	Less detailed maps.
Dense Methods	Use most or all image pixels to create detailed 3D maps.	Creates detailed 3D maps.	Computationally intensive.

Table 4.1: Comparison of Sparse and Dense Mapping Methods

Visual SLAM approaches can also be divided into feature-based and direct methods [14]. Feature-based methods work by detecting and tracking distinct visual features, like corners or edges, across images and estimating the camera motion between these frames. Then, this method reconstructs the environment using feature correspondences. A descriptor, made up of clusters of points, is created for each feature. These descriptors are used to compare in future frames. These descriptors could be based on the intensity of the pixels or the appearance of the features. Feature-based is most effective in well-defined textures whereas it might face challenges in low texture or repetitive-pattern scenarios. It uses algorithms like ORB, SIFT, or SURF to detect these key points. On the other hand, direct methods operate on intensity values of specific regions or the entire image in the image and work on minimizing the photometric error to directly estimate motion. The minimization occurs by aligning image intensities between frames and operating on them directly without extracting specific features. In light of the previous discussion, feature-based methods are faster than direct methods in estimating position in the generated map. Thus, since SHARE-C will be operating in real-time and would require fast estimation of position and map building for path planning, a feature-based algorithm will be used. Since a Jetson TX2 will serve as the onboard computer, the algorithm must also be lightweight and efficient—making feature-based methods more suitable than computationally intensive direct methods for ensuring reliable real-time performance. See Table 4.1.[15] for a summary of the advantages and disadvantages of these methods.

Method	**Description**	**Advantages**	**Disadvantages**	**Examples**
Feature-Based Methods	Detects and tracks distinct features (e.g., corners, edges) across images to estimate camera motion.	Faster and effective in well-textured environments; uses algorithms like ORB, SIFT, or SURF.	Struggles in low-texture or repetitive-pattern scenarios.	• ORB-SLAM • PTAM • SVO
Direct Methods	Operates on intensity values of regions or entire images, minimizing photometric error to estimate motion.	Does not require feature extraction; works well in environments without distinct features.	More computationally intensive; slower for real-time applications.	• LSD-SLAM • DSO • DTAM

Table 4.2: Feature-Based vs Direct Methods

4.1.2 Analysis: Limitations of Current Solutions

Feature-Based Algorithms

Several vSLAM algorithms have been developed over the past decade, each with unique features and limitations. One of the first real-time monocular SLAM algorithms uses an Extended Kalman Filter (EKF) for state estimation. It first initializes the system and then updates the state vector estimating camera motion and environment using an extended Kalman filter (EKF). One of the downsides of MonoSLAM is that the initialization step needs a “known target”, which is not always available [16]. Not only that but in large environments, MonoSLAM is much more computationally expensive than other algorithms. To solve this issue, PTAM, Parallel Tracking and Mapping, implements bundle adjustment and separates the tracking and mapping processes into two threads which is beneficial for the CPU [17]. However, PTAM still lacks loop closure which could lead to accumulated errors over time. ORB-SLAM incorporates loop closure and pose-graph optimization to reduce drift which makes feature extraction more robust and ORB matching effective in dynamic environments [18]. In addition to these, RTAB-Map SLAM is a real-time appearance-based mapping system that supports RGB-D cameras and integrates loop closure detection with global optimization, making it effective in large-scale, dynamic environments.

ORB-SLAM Overview

ORB-SLAM stands for Oriented FAST and Rotated BRIEF SLAM and operates on ORB features, which are Oriented FAST (key points) and BRIEF (descriptors). FAST key points are pixels that are very different from their neighboring pixels which most likely turn out to be corner points. ORB enhances this feature by adding scale and rotation invariance. First, the algorithm tracks these features using FAST corner detection and then enhances it with scale and rotation invariance to make it more robust. Then, after extracting the ORB features, it calculates binary descriptors for each point to describe the area around the detected key point [18].

In order to increase its versatility and enable direct estimation of the state of the environment using depth information, ORB-SLAM2 expanded ORB-SLAM to support stereo and RGB-D cameras in addition to monocular camera setups [19]. Furthermore, by lowering computational overhead while preserving accuracy, ORB-SLAM2 is better suited for real-time operation. In addition to integrating data from Inertial Measurement Units (IMUs) to support Visual-Inertial SLAM, ORB-SLAM3 further improves the algorithm by introducing a new Maximum-a-Posteriori (MAP) initialization strategy that speeds up and improves the accuracy of the results [20]. Also, it introduces a multi-map Atlas structure that supports map merging allowing the system to recognize previously visited areas leading to better accurate localization. In general, ORB-SLAM3 improves robustness and accuracy even though it is more computationally expensive than ORB-SLAM2 due to the integration of IMU data.

RTAB-Map Overview

RTAB-Map (Real-Time Appearance-Based Mapping) is a graph-based SLAM algorithm that can be used in real-time and offline operation. First, RTAB-Map processes input from sensors like RGB-D cameras, Lidar, or even stereo cameras where it selects specific frames called keyframes to map and optimize efficiently. To reduce computational cost, keyframes are stored in a graph and then RTAB-Map uses appearance-based methods to detect loop closures, relying on visual features extracted from the environment. To create the visual dictionaries and to compare images for potential loop closures, it employs the Bag-of-Words (BoW) approach. It is designed for real-time use to especially reduce computational overhead by using adaptive techniques.

4.1.3 Problem Context and Relevance

The chosen solution must support real-time hospital navigation. Hospitals have dynamic obsticales (eg., moving people, equipment) and require accurate localization for smooth navigation. The ability of an autonomous robot to navigate in dynamic environments such as hospitals is crucial for efficient healthcare assistance. Hospitals have unpredictable obstacles such as moving people, medical equipment, and changes in layout, requiring precise localization and mapping.

Implementing an effective SLAM algorithm ensures accurate navigation, reducing risks associated with collisions, delays, and inefficiencies in hospital logistics. Choosing the correct SLAM approach significantly impacts the system's ability to adapt to real-world scenarios. This is why we choose RTAB-Map SLAM algorithm as a good balance needed to support real-time hospital navigation, but at the same time, localize and map accurately.

4.2 Path Planning and Control

4.2.1 Research: Existing Solutions and Efforts

Overview of Path Planning and Control

The main objective of Path Planning and Control is to utilize the map generated from vSLAM to devise an obstacle free path for the robot to reach its destination. In addition to that, path execution and trajectory generation while considering the robot's dynamic constraints are within the umbrella of path planning. Finally, Low Level Controllers are built to achieve precise navigation with low steady state error.

Path Planning

In [21], four types of path planning procedures are stated:

- Sampling based algorithms: Use an initial guess to traverse over feasible paths.
- Node based optimal algorithm: uses graphs and assigns weights to compute the shortest path.
- Mathematic model based algorithms: uses differential equations to describe the workspace and constraints, however, it has a huge time complexity.
- Bio-inspired algorithms: Comes with high computational cost as well, however it iteratively optimizes paths.
- Multi fusion based algorithms: Integrates the advantages of several procedures to achieve optimal action.

Path Planning is divided into two parts during autonomous navigation : Global path planning and Local Path Planning.

Global Path Planning

In [22], Dijkstra's algorithm is used as the global path planner. Dijkstra establishes a path using a breadth-first search methodology. One of its advantages is that it is easy to implement [21]. In [23] the A* star algorithm is used as the global path planner. A* adds a heuristic function to the Dijkstra thereby boosting performance. It is considered as an adjustment to Dijkstra. Dissimilar to A* and Dijkstra, which are node-based algorithms, the Rapidly Random Trees(RRT) algorithms, along with its improved version as stated in [24]and [25] is a sampling-based approach. One of its advantages is that it has low time complexity [21].

Local Path Planning

The Artificial Potential Field (APF), falling under the sampling-based path planning algorithms category, is a highly efficient local path planner that treats the environment as a pair of positive and negative forces. The positive forces correspond to free space while negative forces correspond to obstacle[26]. In addition to that, the Dynamic Window Approach(DWA) is used in [27] as the local path planner. DWA estimates a collision-free trajectory by sampling velocity groups according to the robot's state and chassis model. Afterward, it produces a speed command to control the robot. DWA is highly applicable for dynamic environments in light of the fact of its accomplishment of obstacle avoidance in real-time. In [23], the Time Elastic Band (TEB) is chosen as the local path planner. This algorithm works by detecting changes in the environment, reshaping the path generated by the global path planner and controls velocities as per [28] and [29].

Low Level Control

Fuzzy logic controller (FLC), as stated in [30], requires a trial-and-error tuning methodology. It turns linguistic control strategy (like english words or fuzzy sets like high/low) into an automatic control strategy [**lee1990fuzzy**]. Among the common control methods, Proportional-Integral-Derivative (PID) is a type of low-level controller that can adjust the speed or position of a DC motor with the help of manual tuning of Kp.Ki and Kd [**hirpo2017design**]. On the other hand, Model Predictive Controllers (MPCs) are a type of low-level controllers that calculate manipulated variable adjustments in order to optimize the future behavior of a plant.[**qin1997overview**]

4.2.2 Analysis: Limitations of Current Solutions

Global Path Planning

For Global Path planning, the Dijkstra's Algorithm is slow compared to the A* algorithm. While A* is faster (by integrating a heuristic function that prioritizes nodes that are closer to the goal), it sometimes provides non-smooth paths [21]. In addition to that, RRT does not always result in optimal solutions.

Local Path Planning

APF is susceptible to the local minima problem[31]. Local minima is where the robot gets stuck in a loop forever. In addition to that, [32] states that DWA does not account for the robot's kinematic constraints, which can result in over steering, for instance. Despite the benefits of TEB (outputs a smooth trajectory, is robust enough for dynamic environments, and considers kinematic constraints), there is a tendency for over-steering and wandering [32]. Given that the robot studied will be deployed in a dynamic environment, mechanical dynamics need to be considered.

Low-Level Control

Due to the fact that FLCs require trial-and-error methods, it proves the fact that FLCs are complex to design and time-consuming as well. MPCs demand significant computational effort, compromising their real-time efficiency as stated in [33]. In addition, despite the fact that PIDs are easy to implement, they do not result in optimal control as they lack cost function or predicting functionalities.

4.2.3 Problem Context And Relevance

The chosen algorithms for Path Planning and Control must adhere not only to hospital environments but also to robot's dynamics. The robot must generate smooth trajectories to avoid discontinuities and actuator saturation at sharp transitions. In addition to that, the robot must maintain its stability when encountering a dynamic obstacle. In the end, the robot has to autonomously map, localize and plan an optimal path to navigate to its goal.

4.3 Locomotion System

4.3.1 Research: Existing Solutions and Efforts

The development of locomotion systems for mobile robots has been a significant focus within the robotics and automation industries for many years, particularly in healthcare, logistics, and manufacturing sectors where robots are required to navigate uneven surfaces, adapt to dynamic loads, and maintain stability. Key applications include autonomous wheelchair mobility, where companies like Whill and Permobil have tackled the challenges of adapting to uneven terrains while maintaining user comfort and ensuring stability, safety, and energy efficiency [34]. Medical service robots like TUG by Aethon are deployed in hospitals to transport supplies, needing to efficiently navigate floors with slight irregularities [35]. Industrial Automated Guided Vehicles (AGVs) in warehouses also demonstrate the necessity for robust suspension systems capable of handling significant weight and traversing uneven terrain, emphasizing load distribution and durability [36]. Research in robotic locomotion systems is advancing through universities such as MIT, Stanford, and ETH Zurich, focusing on adaptive suspension mechanisms for real-world applications. Companies like Boston Dynamics and Omron Robotics contribute by innovating dynamic stabilization and adaptable suspensions for factory environments [37]. Startups like Bear Robotics and Keenon Robotics are also pioneering service robots for hospitality and healthcare [38].

4.3.2 Limitations of Current Solutions

Critically evaluating previous approaches reveals the limitations and advantages of various systems. Fixed suspensions, while simple, low-cost, and reliable, offer limited adaptability to uneven surfaces, making them inadequate for complex environments [39]. Active suspensions provide high adaptability and precision but are costly and energy-intensive, which may not be feasible for all applications [40]. Spring-based suspensions strike a balance as they are affordable and energy-efficient but typically handle only minor irregularities [41]. Hybrid systems combining elements of these approaches offer a promising compromise, providing better adaptability but at the cost of increased mechanical complexity.

4.3.3 Problem Context and Relevance

Our approach aims to differentiate itself by focusing on an optimized spring-based suspension designed to handle predictable irregularities up to ± 10 mm, which is crucial for dynamic load management in hospital settings. By improving the load distribution between the caster and the driving wheels and allowing easy adjustments and replacements of the springs and wheels, our design offers a customizable and modular solution. This approach addresses the shortcomings of previous designs and enhances the robot's adaptability and efficiency, making our system a particularly relevant and practical choice for hospital applications.

4.4 Self-Disinfection System

4.4.1 Research: Existing Solutions and Efforts

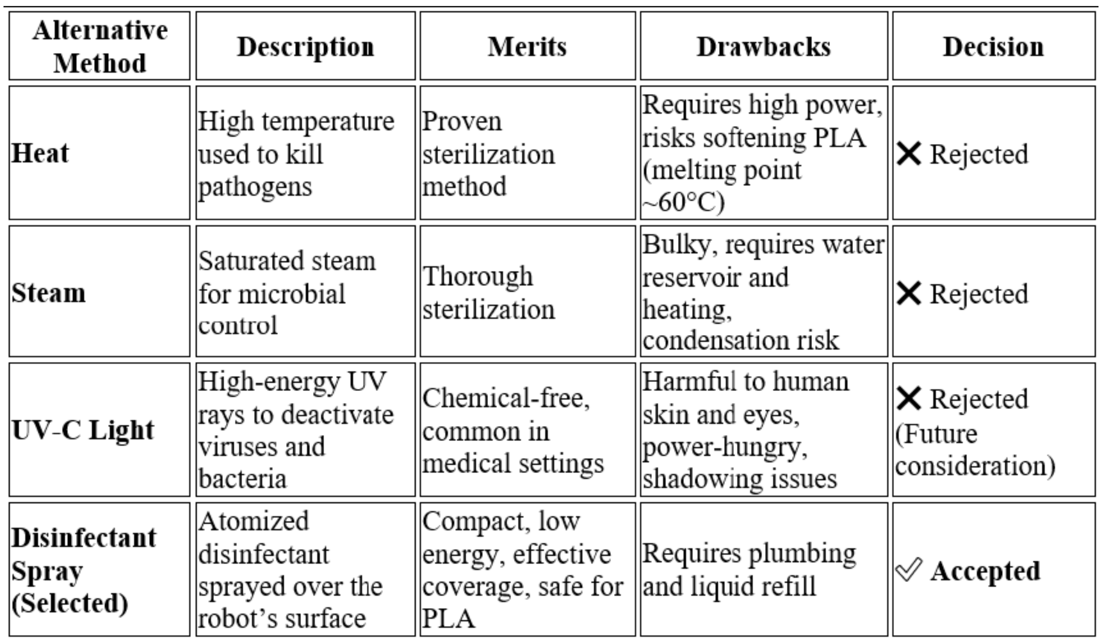
The development of self-disinfection systems in hospital robotics is critical to ensuring hygiene and minimizing the risk of pathogen transmission. Several disinfection technologies were considered for integration into the robotic platform, including ultraviolet (UV) radiation, electrostatic spraying, heat and steam-based disinfection, and chemical sprayers. The selection process was guided by international standards such as ISO 13485:2016, ISO 15883-1:2006, and EN 14476:2013 + A2:2019, which ensure that the disinfection system complies with medical safety, efficacy, and regulatory standards.

UV disinfection is widely used in hospitals, utilizing UV-C radiation to neutralize microorganisms. However, its effectiveness is limited due to the shadowing effect, preventing surfaces not directly exposed to UV light from being disinfected. Additionally, UV-C radiation poses health hazards and requires restricted operation in occupied spaces. UV systems also consume high amounts of energy, which is impractical for battery-powered robots. Moreover, UV light is less effective on porous surfaces, reducing its overall efficiency. As a result, UV-based disinfection was excluded from this application.

Electrostatic spraying, which charges disinfectant particles for better surface adhesion, was also evaluated. While this method offers uniform coverage, it requires specialized disinfectants that may not be compatible with hospital cleaning protocols. Additionally, disinfectant residues can accumulate and interfere with sensitive medical equipment. The complexity and cost of integrating an electrostatic spraying system made it less feasible for the robotic platform.

Heat- and steam-based disinfection methods use high-temperature steam to kill pathogens. Although effective, these methods require significant energy consumption, making them unsuitable for mobile robotic applications. Furthermore, high temperatures could damage plastic components and electronic circuits, leading to potential malfunctions. The humidity produced by steam disinfection could also negatively affect hospital environments, especially in areas with electronic medical devices. Consequently, heat- and steam-based disinfection was not considered viable for integration into the robot.

After evaluating these methods, a sprinkler-based chemical spraying system was selected as the most suitable solution for robotic self-disinfection. Unlike UV disinfection, which is limited by shadowing, and electrostatic spraying, which requires specialized disinfectants, a sprinkler system ensures direct surface contact with the disinfectant, providing consistent coverage. This system is compatible with standard hospital disinfectants and complies with established sanitation protocols.



Alternative Method	Description	Merits	Drawbacks	Decision
Heat	High temperature used to kill pathogens	Proven sterilization method	Requires high power, risks softening PLA (melting point ~60°C)	✕ Rejected
Steam	Saturated steam for microbial control	Thorough sterilization	Bulky, requires water reservoir and heating, condensation risk	✕ Rejected
UV-C Light	High-energy UV rays to deactivate viruses and bacteria	Chemical-free, common in medical settings	Harmful to human skin and eyes, power-hungry, shadowing issues	✕ Rejected (Future consideration)
Disinfectant Spray (Selected)	Atomized disinfectant sprayed over the robot's surface	Compact, low energy, effective coverage, safe for PLA	Requires plumbing and liquid refill	✓ **Accepted**

Figure 4.2: Comparison of Alternative Disinfection Methods for the Robot

The choice of a sprinkler-based system aligns with ISO 15883-1:2006, which sets performance requirements for disinfecting systems in medical environments. Furthermore, EN 14476:2013+A2:2019 ensures that the selected chemical disinfectants are effective against a broad spectrum of pathogens, including viruses. ISO 13485:2016 guarantees that the design, manufacture, and integration of the disinfection system meet international medical device safety and quality standards. The system operates with a low-power pump, making it energy-efficient for mobile robots. It is fully automated, activating upon the robot's entry and exit from a patient's room, ensuring disinfection at critical transition points. Moreover, its compact, non-corrosive design ensures that it does not interfere with the robot's structure or electronic components, while complying with medical safety and hygiene regulations.

4.4.2 Limitations of Current Solutions

The development of self-disinfection systems is an evolving field, and current solutions face several limitations. UV-based systems, while effective at killing microorganisms, face challenges due to their limited coverage area and the potential harm they pose to human health. Furthermore, they are energy-intensive, making them unsuitable for battery-powered robots. Electrostatic spraying, although promising for even coverage, requires specialized disinfectants and has limitations regarding compatibility with existing hospital cleaning protocols. Additionally, the accumulation of disinfectant residues poses risks to sensitive equipment and complicates maintenance. Heat- and steam-based disinfection, though effective in killing pathogens, consumes high energy and can damage delicate components of the robot. These limitations led to the exclusion of UV, electrostatic, and heat-based methods for the robotic self-disinfection system.

4.4.3 Problem Context and Relevance

The selected sprinkler-based chemical spraying system provides a reliable, efficient, and cost-effective solution for self-disinfection in hospital robots. By ensuring direct surface contact with the disinfectant, the system guarantees consistent coverage and is compatible with standard hospital disinfectants. It operates with low power, making it suitable for battery-powered robots, and its automated operation ensures disinfection at critical transition points. This system is not only effective but also compliant with international safety and hygiene regulations, ensuring its relevance in hospital settings where stringent hygiene standards must be met.

The self-disinfection system is a simple, static installation consisting of a 4-liter disinfectant tank connected to a low-power pump and strategically placed sprinklers. Upon entry or exit from patient rooms, the system automatically activates, distributing disinfectant through fixed spray nozzles across the robot's external surfaces. The disinfectant naturally evaporates without residue, eliminating the need for manual intervention or drying mechanisms.

The system adheres to ISO 12100:2010, which provides a structured approach for risk assessment and reduction, ensuring potential hazards—such as excessive chemical exposure or slip risks—are identified and mitigated. Additionally, ISO 14937:2009 sets requirements for sterilization processes, ensuring that the disinfection cycle is repeatable and effective against hospital pathogens.

This approach guarantees automated sterilization, maintaining compliance with hospital safety regulations while ensuring the robot remains contamination-free in patient-care environments.

4.5 Door Opening Mechanism

4.5.1 Research: Existing Solutions and Efforts

Mobile robots with the ability to manipulate doors—that is, open, close, and move through doors—are increasingly important in robotics research due to their relevance in indoor environments. Their applications span healthcare, industrial, domestic, and security fields, where they help automate processes that are typically performed by humans.

Robots that can open doors are useful in hospitals by reducing human contact with door surfaces, thus reducing infection risk. They provide material handling and transportation in hazardous settings within industrial environments, saving human workers from potential injury. They are also helpful in domestic settings, where they enable individuals with compromised mobility to navigate their homes independently. Another interesting application within the security field involves autonomous patrols, whereby security robots walk around buildings independently, opening doors to scan various segments without human assistance. For example, Boston Dynamics' Spot robot has been paired with digital access systems to enable it to open secured spaces on its own [42].

There have been several research contributions that focus on enabling efficient door manipulation. The research work in [43] addresses door manipulation in office buildings, proposing a method where an autonomous robot navigates offices, opens doors, and recharges by plugging into power outlets. Their approach uses a tilting laser scanner for 3D point cloud generation for door detection and both laser scanning and stereo vision methods for handle localization. They perform door opening using compliant manipulation by Task Frame Formalism (TFF), allowing for flexible interaction with moving doors.

Similarly, [44] investigates door opening in mobile service robots with particular consideration of the limitations of differential drive robots. Their systematic method for door and handle detection relies mainly on the utilization of laser-range scanners with a focus on computational efficiency without the inclusion of visual processing methods.

Another method, presented in [45], solves door crossing challenges for autonomous scanning robots by using an a priori created 3D building model, segmented point clouds for door and handle detection, and calculated obstacle maps for safe door traversability.

Adaptive control techniques have also been explored. In [46], an adaptive strategy allows robots to open doors without prior knowledge of their properties by using real-time force/torque feedback. In [47], a complementary approach is proposed, combining adaptive force control with deep reinforcement learning and visual perception to enhance the robustness of the system to various door configurations.

4.5.2 Analysis: Limitations of Current Solutions

While these approaches represent important advancements, some constraints continue to limit their broader application. Some robots, such as those described in [43] and [44], are heavily dependent on laser scanners and can only adjust to pre-mapped structured environments, limiting the capacity of the robots to adjust when confronted with unknown or dynamic environments.

While [44] offers a computationally less burdensome solution by avoiding visual processing complexities, this efficiency is attained at the cost of degraded performance in settings where visual cues play a central role in effective localization and manipulation. Similarly, solutions based on environment priors, like those in [45], are ill-equipped to deal with divergence from everyday conditions against which they were originally developed.

Adaptive control strategies, as seen in [46] and [47], provide greater flexibility in dealing with unknown door properties but often demand extensive sensory input and substantial real-time computational power. This inevitably increases system complexity and raises hardware costs.

Reinforcement learning techniques, although promising for tackling a variety of door types and conditions, often require long training periods and careful tuning to achieve reliable generalization across settings. Finally, adding to these challenges, it is important to note that none of the reviewed approaches is designed with low-cost or easy deployment in mind. This gap presents a significant obstacle to adoption, particularly in healthcare facilities and other resource-constrained environments where affordable and practical solutions are most needed.

4.5.3 Problem Context and Relevance

In the specific context of healthcare settings, door-opening capability for assistive robots like SHARE-C is particularly vital, since it must operate reliably in partially known but still dynamic environments where emergency access and frequent door opening are common. In hospitals, robots must move between rooms efficiently to minimize physical contact of nurses with door surfaces to limit infection risk, particularly for immunocompromised patients.

Therefore, SHARE-C must minimize reliance on large pre-mapping of the environment and be able to properly deal with common hospital door types, such as those with lever handles which require push or pull actions. It must be functional with lightweight sensing, balancing performance against hardware complexity, and must integrate smoothly into larger navigation pipelines without impacting the overall system operation.

The limitations identified in existing solutions directly influence the design choices taken for SHARE-C, resulting in the motivation to develop a door-opening mechanism tailored specifically for practical real-world deployment in healthcare.

4.6 Graphical User Interface

4.6.1 Research: Existing Solutions and Efforts

The integration of graphical user interfaces in the medical field has gained significant attention, especially in environments like hospitals, where constant monitoring of patients is a challenge for the medical staff. Various applications exist that tackle similar challenges, focusing mainly on tasks such as sensor data visualization, human-centered design, and adaptability.

One example of a user interface is a mobile application developed to view the patient's health database collected by the robot "Pepper" in [48]. This app is developed using Flutter and it enables the medical staff and patients to monitor the collected health data across multiple platforms: mobile, desktop, and web clients. This application is tested on an 11" Android tablet and it retrieves the patients' data from MySQL database system and through a REST API for visualization purposes. The prominent elements for achieving a user-friendly app are their use of recognizable icons, high-contrast colors, and an adjustable font, completing an accessible user-centered design.

In another paper [49], a study at the A7L health facility uses six semi-autonomous robots equipped with HD cameras, audio systems that enable two-way communication, and touchscreens for the medical staff and patients. The main goal of these robots is similar to our SHARE-C's goal: reducing infection risks. The staff were able to control the robot using specific software on their PC or mobile phones, then the patients could respond by using the touchscreen on the robot, thus creating a communication loop. However, in this approach, the robots' main tasks are monitoring patients' vital signs through using a camera and allowing telecommunication between the medical staff and the patients.

Another approach was implemented in [50], where the M-Robot is used to provide multi-model types of control interfaces: voice, touch, and remote control through an Android-based interface. The robot is tested to have a speech recognition overall accuracy of 84%, proving its success in having robust speech recognition as a way to ease user interaction with the robot.

The Lio robot in [51] closely aligns with the scope of our project as it entails similar goals and functionality. Lio robot provides two separate user interfaces: the Nursing Interface and the Home Interface. Both interfaces provide clear and user-friendly updates about the current actions of the robot, real-time location, battery level, an emergency stop button, history of actions and logs, patient data, and manual steering of the robot. Furthermore, a voice recognition system is deployed to facilitate communication with the robot especially when elderly people need to use the robot and can't reach the on-screen display of Lio. However, both interfaces are websites and not applications, thus they can only be accessed through a browser.

Regarding facial expressions and audio phrases, [52] highlights the importance of balancing human-likeness in robots. A robot featuring human traits triggers anthropomorphism, which fosters social interaction, however, overly human-like robots cause feelings of restlessness and discomfort around them, a phenomenon known as the "Uncanny Valley". It is stated that overly human-like facial features such as realistic human eyes were rejected by 86% of participants in one study. As for the voice of the robot, research in [53] indicates that a robot with a high human-like voice enhances positive emotions and user likability, while also capturing greater visual attention.

Previous research in robotic applications in the healthcare field highlights the importance of ameliorating the communication system between the user and the robot. For instance, the M-robot in [50], although very effective in basic functions using voice commands, lacks a visual feedback system and hence doesn't allow the user to have constant monitoring of the robot. Moreover, the Lio robot proved its efficient multi-functionality [51], however, it requires nurse supervision as the robot heavily depends on predefined visualizations. This restricts the efficiency of the user interface as the nurse must not only check on patients but also supervise the robot using the Nurse Interface. In addition, the robot Pepper [48] provides a mobile application for the user to be able to view data collected from patients, however, this application lacks critical information concerning the current status of the robot. Similar to the approach in [49], where audio receivers and transmitters were used allowing two-way communication, SHARE-C will be able to play prerecorded audio at each interaction with the patient. To this end, this project addresses the gaps presented by developing a GUI that includes both smooth user adaptability and real-time visual feedback of the robot.

4.6.2 Analysis: Limitations of Current Solutions

To this end, it is important to develop an application and not a website for easier and smoother access. Also, the critical feature needed in the app, alongside assigning orders to the robot, is the feature of live monitoring the real-time status of the robot to ensure the optimal safety of the robot and patients.

4.6.3 Problem Context and Relevance

Our project addresses the gaps presented by developing a Graphical user interface that includes both smooth user adaptability and real-time visual feedback of the robot.

Chapter 5

Proposed Solution Methodology

5.1 Sensing and Perception

5.1.1 Overview of the Proposed Solution

To enable autonomous navigation of SHARE-C using the ROS navigation stack, both mapping and localization are essential. The proposed solution is to use RTAB-Map SLAM with an Intel RealSense D435i camera to generate a map of the hospital corridor and localize itself within it. RTAB-Map will be utilized for mapping and localization due to its robustness, real-time performance, and compatibility with both visual and inertial data. The system includes a depth camera, visual odometry, and RTAB-Map for vSLAM. When combined, these components allow the robot to continuously localize itself while navigating and avoiding obstacles using the ROS navigation stack.

The proposed solution aims to achieve these objectives:

- **Accurate Localization and Mapping:** Create a 2D occupancy map of the surroundings and keep localization accuracy within ±5 cm by utilizing RTAB-Map's visual SLAM capabilities.

- **Real-Time and Efficient Navigation:** Make use of RTAB-Map's real-time capabilities to keep data processing latency for responsive localization and mapping updates under 100 ms.

- **Visual Odometry Integration:** Estimate the motion of the robot using RTAB-Map's visual odometry by means of RGB-D data from the Intel RealSense D435i.

- **Environmental Awareness:** To account for while navigating, detect and avoid obstacles within a 1-meter range using RTAB-Map's depth sensing and loop closure de-tection features.

5.1.2 Methodology Steps

The following steps are to be taken to implement mapping and localization with RTAB-Map SLAM for SHARE-C:

1. **System Design:** The system is composed of an Intel RealSense D435i camera and an NVIDIA Jetson TX2. The camera provides synchronized RGB and depth data with IMU readings. The Jetson TX2 should be set up to run ROS Melodic to be compatible with the navigation stack and RTAB-Map. RTAB-Map and all required Intel RealSense camera dependencies are to be installed and run on the NVIDIA.

2. **Sensor Calibration and Frame Configuration:** Setting up the transforms TF for the robot and Intel RealSense camera D435i is crucial to ensure correct sensor alignment. The coordinate frames for the camera, IMU, and robot base (`base_link`) are to be configured using static transform publishers, enabling consistent frame transformations across all nodes in the ROS environment.

3. **Component Testing:** Each component of the full system is to be tested to ensure smooth operation. Camera frame rate should be tested along with the IMU data, ensuring the format published on the IMU topic is of the correct format. Not only that, but RTAB-Map parameters such as visual odometry are to be debugged and tested to ensure odometry is working by tuning different thresholds and loop closure parameters, optimizing performance.

4. **Data Collection and Analysis:** Mapping sessions are to be conducted in Oxy L4 corridor to collect test data. The resulting maps and trajectories are to be analyzed for drift, accuracy, and loop closure reliability. Based on this analysis, system parameters are to be fine-tuned to improve robustness and precision.

5. **Implementation Strategy:** After generating a reliable map, RTAB-Map is switched to localization mode. The saved database is loaded, and the system is launched to continuously estimate the robot's pose within the pre-mapped environment. This setup is then integrated with the ROS Navigation Stack (`move_base`) to enable autonomous path planning and obstacle avoidance.

5.1.3 Evaluation and Testing Plan

A systematic evaluation and testing procedure is used to evaluate RTAB-Map SLAM system's efficiency. Testing focuses on the system's integration with the

ROS navigation stack, as well as the quality of the generated map and localization accuracy.

- **Performance Metrics:**
  - **Localization Accuracy:** The robot's estimated pose is compared to ground truth measurements, aiming for a positional error within ±5 cm.
  - **Mapping Quality:** The 2D occupancy grid is evaluated for completeness, clarity, and alignment with the physical environment.
  - **Loop Closure Detection:** The system is assessed for its ability to detect and correct drift through consistent loop closures.
  - **System Latency:** Real-time responsiveness is validated by ensuring that mapping and localization updates occur within 100 ms.
- **Testing Conditions:**
  - Mapping and localization are tested in a hospital-like corridor (Oxy L4) with known dimensions and predefined obstacles.
  - The environment includes lighting variations to test system robustness.
- **Validation Methods:**
  - Ground truth comparison is used to validate localization accuracy in RViz.
  - The quality of the generated map is visually inspected and compared to reality.
  - The robot's ability to plan and follow paths using the generated map is tested through integration with the `move_base` package.
  - Repeated navigation trials are conducted to evaluate consistency and robustness of localization over time.

5.2 Path Planning and Control

5.2.1 Overview of the Proposed Solution

The aim of path planning and control is to navigate from the initial to the goal location securely with a recovery behavior implemented in case the robot gets lost, abiding by the error handling technical constraint. In addition to that, Low Level Controllers are built to achieve precise navigation abiding by the precision and accuracy restriction on the Arduino and Nvidia. The entire path planning task is engineered using the navigation stack, mainly move_base which is the heart of path planning in ROS Melodic. Move_base is a ROS navigation package that contains all the global and local path planning YAML (Yet Another Markup Language) files ,obstacle avoidance and recovery behavior functions [54] . This indicates the Path planning algorithms are chosen to respect the battery life limitation (slower algorithms lead to more battery drainage).

5.2.2 Methodology Steps

System or Product Design

SHARE-C's base is chosen to be a Two Wheel Differential Drive Robot (TWDDR). .It is equipped with 4 caster wheels that contribute in stabilizing its motion, an Intel Realsense D435i camera mounted on the head, that helps in mapping and localization. Besides that, low level controllers built on Nvidia which communicates with the Arduino via ROSSERIAL.

Move_base package is the heart of path planning. It developed using via its corresponding launch and YAML files. Path Planning algorithms are designed or chosen to be with low time complexity to abide by the battery duration constraint (slower algorithms lead to more battery drainage) and compatible with move_base. While low level controllers are engineered based on the precision and accuracy technical limitation as low level PID controllers help in accurately navigation to the goal location. Finally, if the robot perceives itself as stuck or cannot plan a path, it will trigger recovery behavior by move_base adhering to the error handling constraint and **ISO 13485** standard.

Initial Testing and Analysis

Initially, all the simulations were done on Gazebo on the turtlebot3, as it is also a differential drive robot that highly resembles our base. RTABMAP-SLAM and move_base, the tools planned to be used within this FYP were utilized in the simulation environment. Within the simulation, the turtlebot3 was able to plan a path and avoid obstacles, after mapping the environment using RTABMAP-SLAM.

Data Analysis and Selection

Simulations are required to highly replicate what is done in this project, meaning that 3D SLAM must successfully be performed using a stereo camera along with autonomous path planning between obstacles. Analysis will be based on the the success of autonomous navigation between obstacles and using 3D SLAM with stereo camera.

Development

Path Planning development starts with simulations on Gazebo as mentioned above on the turtlebot3. The plan is to map a feature-rich environment using a stereo sensor, utilizing RTABMAP-SLAM and subsequently be able to plan a path and validate autonomous navigation.

Navigation Stack in ROS is used to implement Path Planning, particularly utilizing and fine-tuning the YAML files of the Move_base package. The Navigation Stack is an essential ROS instrument that facilitates autonomous navigation, as described in [55].

Controllers are built using Arduino, then integrated into NVIDIA through the Rosserial Package, which enables the transmission of serialized messages over serial ports or network sockets [56].

Implementation

Deployment of all the files is done on Nvidia Jetson TX2, the brain of our robot. This high-end processor contains all the packages and YAML files for this project. It runs Ubuntu 18 as its operating system and communication can be achieved via RealVNC. It is also connected to an Arduino mega which receives ROS messages to the motors or other messages using rosserial.

5.2.3 Evaluation and Testing Plan

Subsequent to Gazebo simulations, path planning and low level control is evaluated based on the following criteria:

- Successful communication between the following nodes:
    1. Map_Server
    2. tf (transform between the links) or localization nodes
    3. Move_base
- The robot is able to autonomously navigate and plan a path in a known environment after it has mapped and successfully localized itself
- The robot achieves tracking between motors during forward movement and avoids drifting.

5.3 Locomotion System

5.3.1 Overview of the Proposed Solution

The proposed solution involves the design and development of a locomotion system for a robot, with key considerations around weight distribution, force analysis, and component selection. The system includes caster and driving wheels with specific spring constants and force distribution parameters. The design will comply with ISO 13482 and ISO 22878 standards for safety and durability. A CAD model will be created for the circular base, and various components will be selected and assembled, followed by experimental testing to ensure performance and safety.

5.3.2 Methodology Steps

- **Requirement Analysis and Problem Definition:**
  - Calculate the total weight of the robot (50 kg) and the distribution of force across caster and driving wheels.
  - Define operational tolerances, including ±10 mm spring compression and surface irregularities.
  - Identify relevant standards, such as ISO 13482 and ISO 22878, for guidance on design and performance.
- **Force Analysis and Component Selection:**
  - Perform calculations for force distribution, considering:
    - Caster wheels: Fcaster = 18.75 kg distributed over 4 wheels.
    - Driving wheels: Fdriving = 31.25 kg distributed over 2 wheels (2 springs per wheel).
  - Determine required spring constants:
    - Caster wheels: k = 6900 N/m.
    - Driving wheels: k = 7650 N/m.
  - Select springs and wheels with appropriate specifications and safety factors.
- **Design and Simulation:**
  - Develop a detailed CAD model of the circular base, including mounting points for wheels and springs.
- **Prototype Development:**
  - Assemble the selected components, ensuring proper alignment and mechanical integrity.
  - Test the system under real-world conditions to assess stability, load distribution, and performance.
- **Iterative Refinement:**

  - Refine the design based on test results by adjusting spring placement or stiffness and redesigning inefficient components.
  - Re-test the refined system to ensure consistent performance under various conditions.

- **Final Design and Optimization:**
  - Optimize materials and dimensions to balance cost, durability, and performance.
  - Update CAD drawings with precise tolerances according to ASME Y14.5 standards.
  - Validate the final design for safety, performance, and adherence to industry standards.

5.3.3 Evaluation and Testing Plan

The system will be evaluated through experimental testing under real-world conditions, focusing on:

- **Compression behavior:** Test on surfaces with ± 10 mm irregularities.
- **Stability:** Evaluate performance during maneuvers such as turning, acceleration, and deceleration.
- **Load distribution and suspension system:** Analyze the load distribution across caster and driving wheels.

Test results will be compared against ISO 13482 and ISO 22878 standards for personal care robot safety and caster wheel durability. Based on these evaluations, iterative refinements will be made to optimize the design and performance. The final design will undergo a thorough validation process to ensure safety, reliability, and compliance with relevant standards.

5.4 Self-Disinfection System

5.4.1 Overview of the Proposed Solution

The proposed solution involves designing a fully automated static self-disinfection system integrated within a healthcare robot. This system is specifically engineered to sterilize the external surfaces of the robot effectively. The solution includes a 4-liter disinfectant tank, connected via a network of pipes to a low-power electric pump and five strategically positioned spray nozzles. Each disinfection cycle operates for 5 seconds, allowing the system to complete 100 cycles per

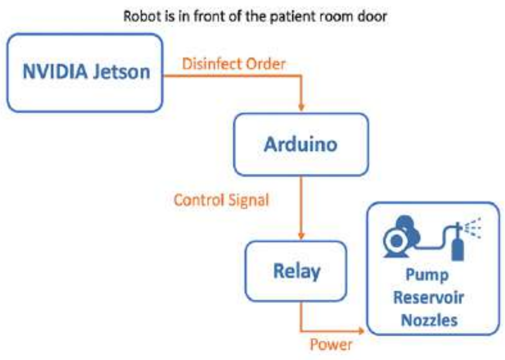


Figure 5.1: System Architecture for Autonomous Disinfection

reservoir refill. The core objectives include ensuring comprehensive surface disinfection, minimal chemical and energy usage, rapid evaporation without residues, and adherence to stringent international safety and environmental standards.

5.4.2 Methodology Steps

- **Given Data:**
  - **Tank Capacity:** 4 L
  - **Operating Time per Cycle:** 5 seconds
  - **Number of Cycles:** 100
  - **Number of Sprinklers:** 5
- **Total Flow per Cycle:**

$$\text{Flow per cycle} = \frac{\text{Tank Volume}}{\text{Number of Cycles}} = \frac{4\,L}{100} = 0.04\,L/\text{cycle} \tag{5.1}$$

- **Total Required Flow Rate (Q):**

$$Q = \frac{0.04\,L}{5\,s} = 0.008\,L/s \tag{5.2}$$

  Converting to L/min:

$$Q = 0.008\,L/s \times 60\,s/min = 0.48\,L/min \tag{5.3}$$

- **Flow Rate per Sprinkler:**

$$Q_{\text{sprinkler}} = \frac{Q_{\text{total}}}{\text{Number of Sprinklers}} = \frac{0.48\,L/min}{5} = 0.096\,L/min \tag{5.4}$$

- **Maximum Head (Pressure) Estimation:**

$$\text{Max Head} = 2.0\,\text{bar} \approx 20\,\text{meters} \tag{5.5}$$

- **Pump Power Calculation:**

$$P = \frac{\rho \cdot g \cdot Q \cdot H}{60} \tag{5.6}$$

  where:

  - $\rho = 1000\,kg/m^3$ (density of fluid)
  - $g = 9.81\,m/s^2$ (gravitational acceleration)
  - $Q = 0.48\,L/min = 0.00048\,m^3/min$
  - $H = 20\,m$

  Substituting values:

$$P = \frac{1000 \times 9.81 \times 0.00048 \times 20}{60} \tag{5.7}$$

  Computing:

$$P = \frac{94.176}{60} = 1.57\,W \tag{5.8}$$

  Considering an efficiency of 40%, the actual pump power required is:

$$P_{\text{motor}} = \frac{1.57\,W}{0.4} = 3.93\,W \tag{5.9}$$

  Thus, the recommended practical motor rating (with safety margin) is:

$$\text{Pump Motor Power} \approx 5 - 10\,W \tag{5.10}$$

5.4.3 Evaluation and Testing Plan

The testing and evaluation plan involves the following steps:

1. **Operational Testing:** Running multiple continuous cycles to evaluate reliability and durability.
2. **Coverage Testing:** Utilizing simulation and visual inspection to verify complete disinfectant coverage and absence of blind spots.
3. **Efficiency Testing:** Monitoring disinfectant consumption against calculated volumes, ensuring efficient resource utilization.
4. **Performance Metrics:** Tracking pump power consumption, nozzle performance, and disinfectant evaporation rates.
5. **Leakage Testing:** Inspecting all fluid connections and components under standard operating pressure to ensure no leakage occurs, following relevant safety and quality standards.

5.5 Door Opening Mechanism

5.5.1 Overview of the Proposed Solution

The proposed methodology for the door-opening mechanism includes the following steps: conducting simulations of the LoCoBot using ROS, RViz, Gazebo, and other relevant software; identifying the optimal solution process for the door opening approach; manufacturing a door and handle compatible with the LoCoBot; training computer vision models to detect the handle; and performing experiments using the LoCoBot and the manufactured door. Finally, the process is iteratively refined by improving path planning and manipulation algorithms to optimize experimental results and enhance accuracy.

5.5.2 Methodology Steps

The LoCoBot WidowX-200 5 DOF platform, available in the Vision and Robotics Lab (VRL), is simulated in Gazebo using its Unified Robotics Description Format (URDF) model. It is equipped with an Intel RealSense Depth Camera D435, a gripper as the arm's end effector, and an onboard computer for processing data and running algorithms in real-time. ROS 1 (Kinetic) serves as the software framework on a machine running Linux Ubuntu 16.04, as this configuration is compatible with the LoCoBot.

Simulation tasks include inserting a model of a door with a handle in Gazebo and defining its kinematic constraints to replicate realistic hospital doors. The LoCoBot is programmed to locate, approach, and manipulate the handle within the simulated environment through an iterative process to test and validate the path planning and control strategies.

Testing is also performed on the physical robot in the VRL. Testing begins with the robot's Depth Camera, Intel RealSense D435, to evaluate perception performance. A custom YOLO-based detection approach was developed to recognize door handles. The methodology involved training a deep learning model on a dataset of door handle images captured from the LoCoBot's onboard camera, detecting the door handle and localizing the graspable area using bounding box outputs, converting the detected handle's image coordinates to 3D space using depth information, and executing grasp planning to align the end-effector with the optimal grasp pose. The handle detection approach was optimized by collecting 400 images, labeling them, and training the model on Google Colab before deploying it on the LoCoBot.

As for the base, example navigation tools are executed using both velocity and position control to verify the robot's mobility and obstacle-avoidance capabilities in a physical environment. Additionally, example arm manipulation tools are run to assess joint and end-effector control. Testing continues with the developed manipulation plan that includes relative and absolute position control, trajectory tracking, and joint angle monitoring.

To ensure that the proposed approach aligns with the project constraints and builds upon existing work, a comparative analysis of door-opening methodologies was performed. Table 5.1 presents a summary of key considerations from relevant literature. The papers in [44] and [45] align most closely with the project's constraints, mainly because they propose solutions that leverage previously known environments, do not rely on Force/Torque sensors, and use an RGB-Depth camera.

Table 5.1: Comparison of Design Specifications Across Literature Sources

Specification	[43]	[44]	[45]	[46]	[47]
Differential Drive	✗	✓	✗	✗	✗
Known Environment	✗	✓	✓	✗	✗
Lever Handle Type	✓	✓	✓	✓	✓
Push/Pull Doors	✓	✓	✓	✓	✓
Laser Sensors	✓	✓	✓	✗	✓
RGB-D Camera	✓	✓	✗	✗	✓
No Force/Torque Sensors	✗	✓	✓	✗	✗
ROS Implementation	✓	✗	✗	✗	✓
Real-Time	✓	✓	✓	✓	✓

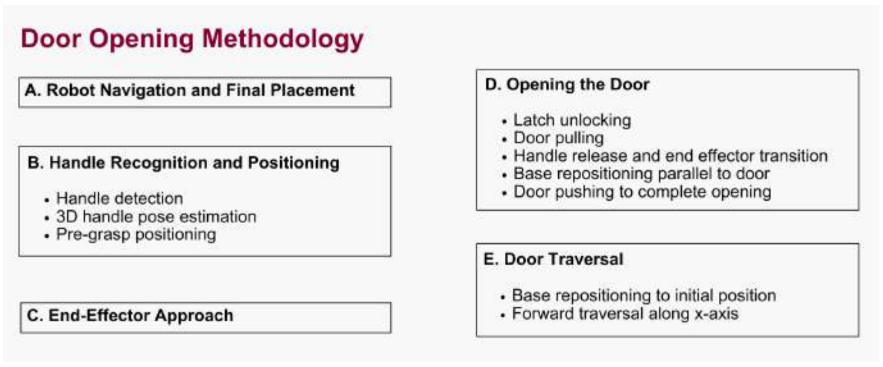

Door Opening Methodology
A. Robot Navigation and Final Placement
B. Handle Recognition and Positioning
• Handle detection
• 3D handle pose estimation
• Pre-grasp positioning
C. End-Effector Approach
D. Opening the Door
• Latch unlocking
• Door pulling
• Handle release and end effector transition
• Base repositioning parallel to door
• Door pushing to complete opening
E. Door Traversal
• Base repositioning to initial position
• Forward traversal along x-axis

Figure 5.2: Door Opening Methodology

5.5.3 Evaluation and Testing Plan

To conduct experiments, a small door and handle are designed using CAD software (SolidWorks) and manufactured in the SRB Workshops. The door design considers geometry and mass constraints to ensure compatibility with the LoCoBot's working payload of 200g at a maximum of 50% arm extension. Experiments are carried out in the Vision and Robotics Lab (VRL) on the manufactured door to validate the developed door-opening methodology. For physical experiments on the LoCoBot, we start by testing the handle detection using YOLO and evaluating the grasping accuracy. By analyzing failure cases, the motion planning strategy is refined based on experimental results compared to expected outcomes.

5.6 Graphical User Interface

The proposed methodology for developing a user interface mainly focuses on establishing an intuitive and easy-to-use interface for nurses to use as a way of interacting with the robot. Firstly, it was thought of having a user interface only on the station of the robot where it charges, however, this limits the reachability of the GUI.

5.6.1 Overview of the Proposed Solution:

The decision for a final type of user interface was made after deliberate research comparing two types of interfaces: A website and a mobile application. The following key factors were considered:

1. Portability and Accessibility: As nurses work in a dynamic and fast-paced environment, immediate and easy access to the user interface is critical. Thus, a mobile application is a solution for nurses where they can communicate with the robot while moving between patients' rooms and offices.

2. Optimized User-Experience: Mobile applications serve as an accessible platform for nurses to use easily. In comparison with a website, the app's interface leverages features like push notifications and haptic feedback, which provide a more engaging and efficient user experience.

3. Real-Time communication: When communicating with a robot in a healthcare environment, it is critically important to have an option to constantly monitor the robot in case of an emergency or a sudden crash. Therefore, persistent connectivity with the robot is essential to ensure real-time updates without requiring the user to log in or reload a website.

Telecommunication System: Video Calling App

To facilitate communication between patients and nurses, a new telecommunication video calling app must be added on SHARE-C's face, separate completely from the SHARE-C's nurse app stated above. This app's main function is to ease the communication between patients and doctors, thus allowing the patients a chance for an online meeting with their doctor as SHARE-C enters the room. The video calling app requires a video SDK, along with the Doctor's contact, where patients can enter a meeting with their doctor with the press of a button.

Facial Expressions and Audio Phrases

Speech and facial expressions play a critical role in user acceptance. If a robot displays very human-like characteristics, it may lead to discomfort. To this end, the screen of the robot must display simple happy or neutral expressions while avoiding intricate drawings that look like a human face. As for the audio, its voice will be human-like to enhance the user's likability and positive emotions.

5.6.2 Methodology Steps:

SHARE-C's mobile application

The SHARE-C's mobile application uses a client-server architecture: The app is the client and the backend is the control system of the robot that enables it to execute tasks based on data received from the client.

1. Frontend: The frontend system is the level of direct interaction of the user with the app. All the interactive visual elements are placed to establish an optimal (UI/UX). This project will be built using **Flutter** as it allows accessibility to multiple platforms: tablet, desktop, and also web.

2. Backend: The backend is responsible for managing the side of the server where the communication between the Frontend and the database system takes place. This project will use **Node.js** with the database system developed using **MongoDB**. At this level, user authentication, robot status, and task scheduling take place.

Video Calling App for telecommunication

To start with the telecommunication development process, a small-scale new Flutter-based application must be built to enable real-time communication for users. For the video calling feature, **Agora SDK** will be used to enable real-time communication (RTC) and integrate high-quality video and voice calling into the video calling application.

SHARE-C's Face Expressions

Based on the previous SHAREC's robot thesis, which was developed by Marwa Ismail, the process of adding SHAREC's face was based on designing face expressions animations and placing them on a tablet, which acted as the robot's face. To this end, the face animations were inspired on the previous designs, but due to the unavailability of the original drawings, the designs must be redesigned from scratch using **Adobe Illustrator** and **Adobe Animate**. The use of the research-based facial expressions ensured that SHARE-C conveys the needed emotions effectively. This helps in providing a comfortable approach for SHARE-C to ensure a natural and engaging interaction.

5.6.3 Evaluation and Testing Plan

To establish a preliminary design of the mobile application, **Figma** was used. Figma is a design tool that enables the app creator to create an intuitive first-level design of the GUI, before starting the actual coding process. Figma design must include the following critical components:

1. Login Page: This page allows the nurses to log in to be able to access the app. This process is critical to ensure that authorized medical staff members are allowed to access the control of the robot.

2. Robot status page: This page includes a visually pleasing illustration of the current battery of the robot, a map that shows the real-time location of the robot, a button that allows the robot to go to its charging base, and the current task the robot is completing and the time it needs for it to finish its task.

3. Tasks Page: Displays a list of tasks that the user could order from the robot, along with the room to which the robot is required to deliver the order. In addition, as all tasks are time-oriented, a clock should be used to allow users to choose the time to start their tasks.

4. Schedule Page: This page must also include a list of all the tasks done, the tasks in progress, and the future tasks to be completed.

Chapter 6

Preliminary and Final Design

6.1 Sensing and Perception

This chapter outlines the preliminary design, implementation, and testing of the vSLAM algorithms for SHARE-C's autonomous navigation within a hospital environment. The design decisions and justifications are discussed based on the technical constraints.

6.1.1 Design Alternatives

Mapping

To enable SHARE-C to move around autonomously within a hospital environment, two vSLAM algorithms, **ORB-SLAM3** and **RTAB-Map**, were identified as potential solutions. To assess the suitability of these algorithms and to understand how each algorithm operates, along with their respective strengths and limitations.

The algorithms were evaluated based on the following key criteria:

- **Tracking accuracy**
- **Computational efficiency**
- **Compatibility with corridor settings**
- **Hardware Feasibility on embedded systems like Jetson TX2**

Initially, **ORB-SLAM3** was selected for testing due to its advancements in feature tracking and loop closure detection. ORB-SLAM3 demonstrated strong localization performance during testing with the EuRoC dataset and the Intel RealSense D435i, validating its accuracy when compared to ground-truth data.

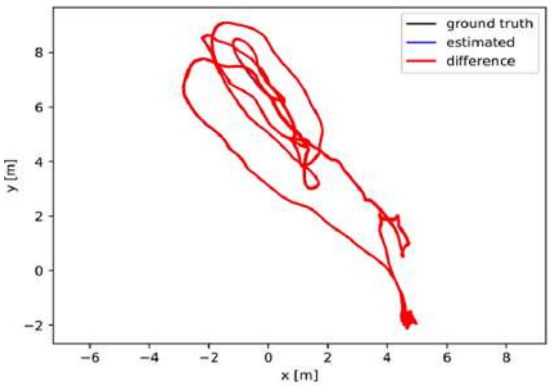


Figure 6.1: Validation of Estimate vs Ground Truth in ORB-SLAM3

However, limitations in map density, computational overhead, and adaptability to dynamic or low-light conditions emerged during real-world experimentation. These insights led to a reassessment of RTAB-Map's capabilities for dense RGB-D mapping.

Localization

To enable SHARE-C to localize itself within a pre-built map and navigate reliably, two primary localization methods were considered: **AMCL (Adaptive Monte Carlo Localization)** and **RTAB-Map Localization**. AMCL was initially chosen due to its tight integration with the `move_base` package in the ROS Navigation Stack and its wide adoption in mobile robot navigation. It provides probabilistic localization by comparing incoming 2D laser scans to a known occupancy grid map. The rationale for choosing AMCL included:

- Seamless compatibility with ROS Navigation Stack
- Lightweight computational demand
- Extensive documentation and community support

However, several critical issues emerged during implementation and testing:

- **Sensor Limitations:** AMCL only accepts 2D laser scans. Since the robot is equipped with a RealSense D435i RGB-D camera, a depth-to-laser projection node had to be used. This introduced additional noise and reduced scan fidelity.
- **Odometry Requirement:** AMCL requires reliable odometry. Testing showed that the onboard wheel encoders produced noisy and unstable odometry which was unsuitable for accurate localization.

- **Visual Odometry Attempt:** An attempt was made to generate visual odometry using RTAB-Map's visual odometry module. However, this approach introduced complex dependencies and synchronization issues, and failed to provide consistent localization results in practice.

These challenges, especially the mismatch between required inputs and available hardware, prompted a reassessment of the localization approach.

6.1.2 Design Decisions

Mapping

While ORB-SLAM3 excels in feature tracking and loop closure, **RTAB-Map SLAM** was ultimately chosen for SHARE-C's hospital navigation. This decision was based on the following justifications:

- **Superior Dense Mapping**: RTAB-Map SLAM generates dense maps, enabling reliable real-time obstacle detection.
- **Dynamic Adaptability**: It handles environmental changes dynamically using real-time map updates and loop closure mechanisms.
- **Lighting Robustness**: RTAB-Map operates effectively in low-light environments using RGB-D data, which is critical for poorly lit hospital corridors.
- **Real-Time Performance**: The system achieves low latency and consistent frame rates for real-time operation.

In addition, RTAB-Map is more compatible with the computational limitations of the NVIDIA Jetson TX2, which serves as the onboard processor for SHARE-C. Its real-time performance and support for RGB-D input make it a practical choice for embedded deployment.

Table 6.1 summarizes the advantages of using RTAB-Map SLAM for mapping across key design criteria.

Localization

Based on experimental evaluation and system constraints, **RTAB-Map Localization** was selected as the final localization strategy for SHARE-C.

This is because RTAB-Map has:

- **RGB-D and Visual Odometry Support:** RTAB-Map can compute visual odometry directly from the RealSense D435i camera without relying on external odometry or 2D scan data.

Criteria	**Constraints**	**RTAB-Map SLAM Capability**
vSLAM Accuracy	Should be within 5 cm of the actual location.	Achieves this accuracy when paired with high-quality RGB-D sensors like RealSense.
Obstacle Detection	Should be within 1 meter.	Supports dense mapping, enabling reliable obstacle detection within 1 meter.
Dynamic Environment	Handle occlusions and moving objects while maintaining localization.	Handles dynamic changes effectively using real-time map updates and loop closure.
Lighting	Operate reliably in typical hospital lighting, including low-light areas.	Operates well in low-light environments using RGB-D data, independent of lighting.
Reflective Surfaces	Maintain accuracy despite reflective floors and low-texture surfaces.	Minimizes reliance on visual features by leveraging depth data for robust mapping.
Real-Time Processing	Ensure low latency and consistent frame rates for real-time performance. Data latency ¡ 100 ms.	Optimized for real-time performance, achieving low latency with efficient computation.
Adaptability	Quickly adjust to layout changes, such as moved furniture or equipment.	Dynamically updates maps to reflect environmental changes in near real-time.

Table 6.1: RTAB-Map SLAM Capabilities Based on Key Criteria

- **Unified Pipeline:** RTAB-Map integrates SLAM and localization within the same package, reducing complexity and improving robustness.
- **No External Odometry Needed:** Eliminates dependency on noisy encoder data or external packages.
- **Better Suitability for Sensor Setup:** Optimized for environments where RGB-D data is available but LiDAR is not.
- **Hardware Compatibility:** Performs efficiently on embedded platforms such as the NVIDIA Jetson TX2.

RTAB-Map Localization provided a more coherent and reliable solution tailored to the available sensor setup and computing resources. It reduced system integration issues, improved localization stability, and ensured consistency across the mapping and navigation stack.

Criteria	**AMCL**	**RTAB-Map Localization**
Sensor Input Compatibility	Requires 2D laser scan; not directly compatible with RGB-D cameras	Natively supports RGB-D cameras such as RealSense D435i
Odometry Requirement	Requires accurate odometry (e.g., from wheel encoders)	Can operate using internal visual odometry from the camera
Integration with move_base	Natively integrated and widely used in ROS Navigation Stack	Requires additional setup, but integrates well once configured
Noise Sensitivity	Sensitive to noise from depth-to-laser scan conversion	Directly uses depth input, avoiding scan conversion noise
System Complexity	Requires external odometry and scan conversion nodes	Single-package solution with built-in localization and visual odometry
Final Decision	Initially tested but rejected due to integration issues	Selected for final implementation

Table 6.2: Comparison of AMCL and RTAB-Map for Localization

6.1.3 Design Iterations

During the initial vSLAM runs using RTAB-Map, significant issues arose while mapping the corridor. The corridor environment lacks enough visual features, which reduced the performance of the feature-based RTAB-Map algorithm. These challenges appeared when:

- Incomplete or fragmented maps were constructed
- Localization drift and loop closure failures occurred

To isolate the root cause, multiple mapping sessions were conducted in different environments, ensuring the issues were not due to sensor misconfiguration or software errors. This iterative testing confirmed that the inaccurate mapping was due to the corridor environment itself.

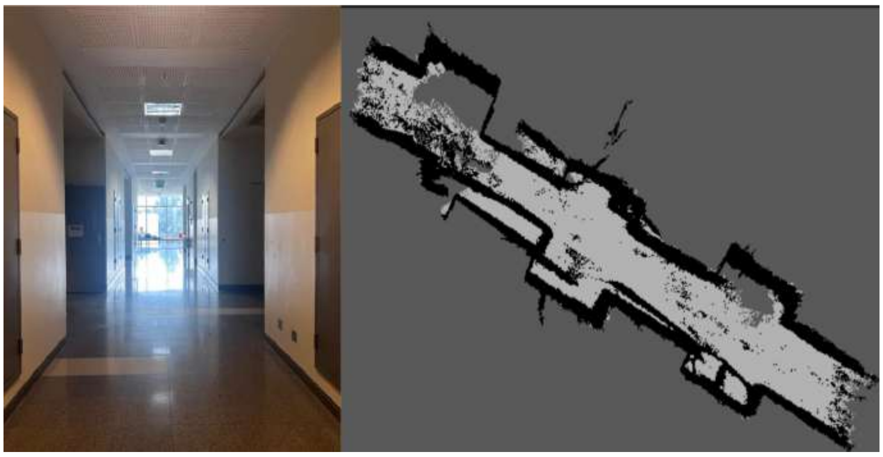


Figure 6.2: Corridor Problem

To generate a better map of the featureless corridor, the same corridor was mapped multiple times at different times of the day. Initial maps were captured during daylight hours, followed by additional sessions at night to observe performance in low-light conditions. In the day, the light was too bright from Oxy's large window, leading to a lot of reflections. To mitigate this, mapping was done in the afternoon and artificial features such as posters and objects were temporarily introduced along the corridor to improve feature density and enhance SLAM stability.

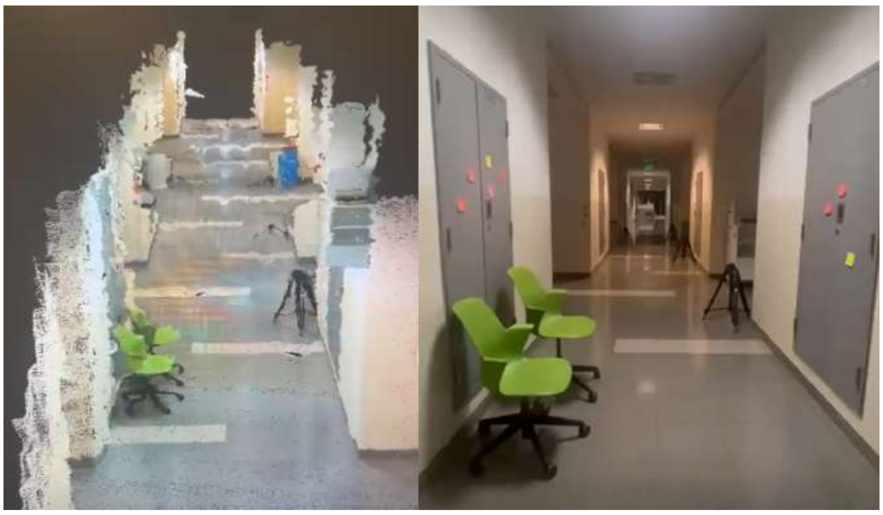


Figure 6.3: Adding Artificial Features in the Corridor (*Left*: 3D Map of Corridor, *Right*: Phone Image of the Corridor)

Final Mapping and Post-Processing

After refining parameters and remapping several times, a clean and consistent occupancy grid map was generated using RTAB-Map. To further enhance map usability for navigation, the map was processed using **GIMP** image editing software. This step involved:

- Removing noise
- Filling gaps and smoothing corridor walls for accurate obstacle inflation
- Adjusting map contrast and boundaries for move_base

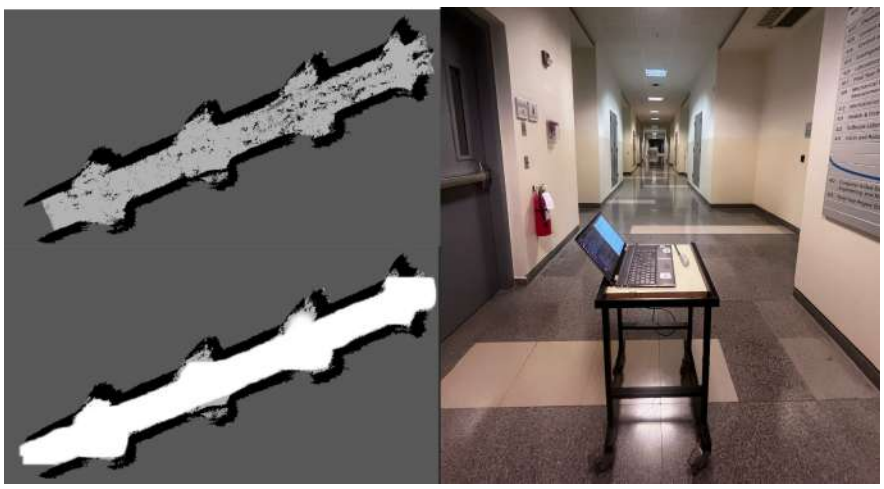


Figure 6.4: Map Reconstructed Before and After Cleaning

6.1.4 Applied Standards

In alignment with ISO 13485, which governs quality management systems for medical devices, the design and implementation of the sensing and perception system for SHARE-C followed a systematic approach to ensure quality and reliability. Quality checks were conducted throughout the development process, including rigorous validation of vSLAM algorithms (ORB-SLAM3) against benchmark datasets like EuRoC to verify tracking accuracy and mapping precision. Issues related to sensor compatibility, dependency installations, and software integration were debugged to minimize potential failures during real-time hospital navigation.

6.2 Path Planning and Control

6.2.1 Design Alternatives

As stated in chapter 5, design decision is based on following technical constraints and standards.

Design Alternative 1: Turtlebot3 with Google Cartographer

TurtleBot is a low-cost robot that features open-source software. The two Figures below portray a picture of the TurtleBot3 along with its corresponding URDF CAD model. Google cartographer is a ROS package that generates performs SLAM using the Lidar sensor.

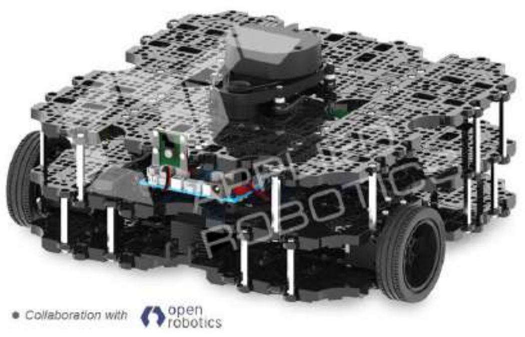


Figure 6.5: TurtleBot3 Waffle

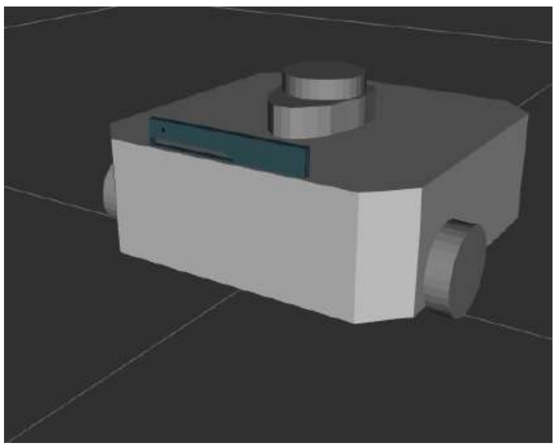


Figure 6.6: TurtleBot3 simulation model

Design Alternative 2: Custom URDF with SLAM_toolbox

A custom URDF model of a TWDDR with a spherical caster wheel and cylindrical LiDAR sensor is coded using XACRO. SLAM_toolbox package is then utilized to acquire a map and then autonomously navigate. Figure 6.3 shows a picture of the design.

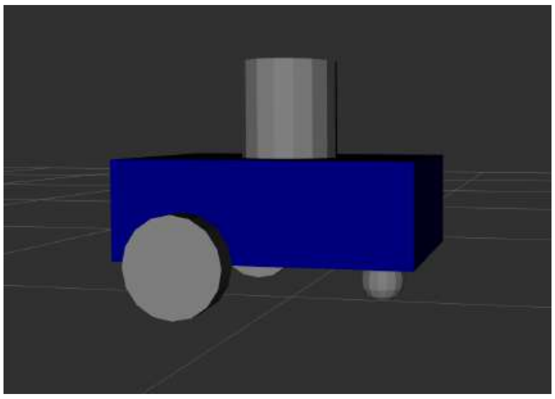


Figure 6.7: Custom made URDF model

Design Alternative 3: Turtlebot3 with RTAB-MAP

Similarly to Turtlebot3 with Google Cartographer, there is also other packages associated with Turtlebot3 called RTAB-MAP. These packages are responsible for mapping using RTAB-MAP. Thus, the same robot is being used but different sensors (Intel-Realsense in RTAB-MAP while Lidar in Google cartographer) and different SLAM algorithm is used.

6.2.2 Design Decisions

It is imperative to note that design and algorithms used in Gazebo must mimic what is really being done. Thus, 3D SLAM must be performed along with a robot model that has similar dynamics to the actual one being worked on in this project and after 3D perception and path planning. Also, path planning and low level control algorithms used in the back-end must respect the constraints mentioned in chapter 3, particularly, battery life, error handling and precision and accuracy.

6.2.3 Design Iterations

The previously stated design alternatives are all implemented in Gazebo. However, some had bad results or did not generate a 3D map. For instance, using design alternative 1, Turtlebot3 with Google Cartogrpaher generated a 2D map instead of a 3D one, which is different than that of what is required. Figure 6.6 shows the generated map from Google Cartographer package.

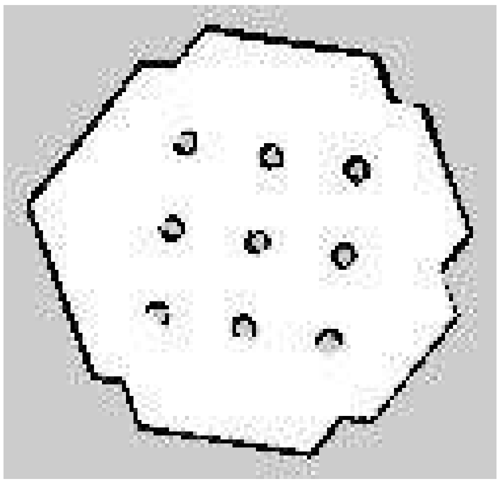
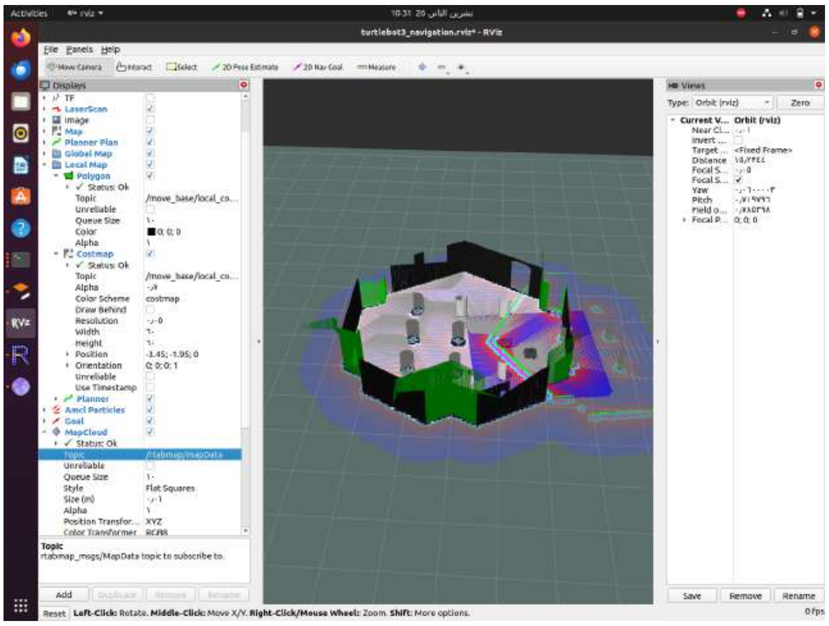


Figure 6.8: Map of Turtlebot3_world

Therefore, another strategy is needed. To enable greater flexibility and ease of modification, since Turtlebot3 is pre-programmed and hard to modify, design alternative 2 or a custom URDF model of a TWDDR with a spherical caster wheel and cylindrical LiDAR sensor above is coded. However, Lidar sensor is not utilized in this project and SLAM_Toolbox used also generated 2D map. Therefore, this alternative is avoided. Finally, Turtlebot3 with RTAB-MAP generated a 3D map of the environment with Intel-Realsense sensor, similarly to what is being done in this project. Therefore, the third alternative is a potential candidate. Figure 6.6 shows a 3D map of the environment.



```
TF
LaserScan            ✓
Image
Map                  ✓
Planner Plan         ✓
Global Map           ✓
Local Map            ✓
  Polygon            ✓
    ✓ Status: Ok
    Topic            /move_base/local_co...
    Unreliable
    Queue Size       ١٠
    Color            ■ 0; 0; 0
    Alpha            ١
  Costmap            ✓
    ✓ Status: Ok
    Topic            /move_base/local_co...
    Alpha            ٠٫٧
    Color Scheme     costmap
    Draw Behind
    Resolution       ٠٫٠٥
    Width            ٦٠
    Height           ٦٠
    Position         -3.45; -1.95; 0
    Orientation      0; 0; 0; 1
    Unreliable
    Use Timestamp
  Planner            ✓
Amcl Particles       ✓
Goal                 ✓
MapCloud             ✓
  ✓ Status: Ok
  Topic              /rtabmap/mapData
  Unreliable
  Queue Size         ١٠
  Style              Flat Squares
  Size (m)           ٠٫٠١
  Alpha              ١
  Position Transfor... XYZ
  Color Transformer  RGB8

Topic
rtabmap_msgs/MapData topic to subscribe to.
```

```
Type: Orbit (rviz)     Zero
Current V...   Orbit (rviz)
  Near Cl...   ٠٫٠١
  Invert ...
  Target ...   <Fixed Frame>
  Distance     ١٥٫٧٢٤٤
  Focal S...   ٠٫٠٥
  Focal S...   ✓
  Yaw          -٠٫٦٠٠٠٠٣
  Pitch        ٠٫٧١٩٧٩٦
  Field o...   ٠٫٧٨٥٣٩٨
  Focal P...   0; 0; 0
```

Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel: Zoom. Shift: More options.

Figure 6.9: 3D Map generated from RTAB-MAP

The above map serves as a basis to the planner to autonomously navigate in the environment.
In addition to design requirements, design constraints must be respected. For path planning and control, the constraints are on Battery Duration and Environmental Adaptability as mentioned in chapter 3. To respect the constraint of battery duration(8 hours per shift), fast algorithms or algorithms with low time complexity must be used(also respecting robot dynamics). The A* (global path planner) ensures minimal energy as these two algorithms are relatively faster. As a result, the low time complexity ensures less computational effort and thus less usage of energy or battery drain. In addition to that, Turtlebot3 was able to successfully or precisely arrive to its navigation goal avoiding obstacles at the same time, which complies with the precision and accuracy limitation and finally the robot changes path if a dynamic obstacle suddenly shows. This indicates the adherence to recovery behavior or error handling constraint.

6.2.4 Applied Standards

As mentioned in Chapter 3, the applicable standards were taken into account in Gazebo simulations. According to **ISO 13485**, a collision-free path was planned after mapping the environment to ensure the safety of the robot while serving or assisting the nurse. In addition to that, an extensive literature review was done on path planning and control algorithms to evaluate the most convenient for this project in terms of speed, accuracy, and implementation. As a result, risk is ensured to be mitigated. Similar standards are applied when working with the hardware.

6.3 Locomotion System

6.3.1 Design Alternatives

For the nurse robot's locomotion system, various design configurations were evaluated to ensure optimal mobility within hospital environments. The key alternatives considered were:

Design Alternative 1

A two-wheel differential drive system combined with four caster wheels, each equipped with a simple spring-based suspension. This configuration is designed for enhanced maneuverability and smooth handling of slight irregularities in floor surfaces.

Design Alternative 2

A four-wheeled configuration with independent spring suspensions, aimed at straightforward navigation on more uniform surfaces.

After thorough evaluation, **Design Alternative 1** was selected for its superior maneuverability and adaptability, both essential for efficiently navigating complex hospital layouts and transitioning between different types of flooring.

6.3.2 Design Decisions

Design Alternative 1:

Weight Distribution One of the critical design aspects involved calculating the weight distribution across all wheels to ensure stability and efficiency during operation. The robot's total weight W_{total} is distributed between the drive wheels and caster wheels, with the following calculations:

1. **Calculate Base and Wheel Radii:**

- **Outer Radius of the Ring** (R_{outer}): 25 cm (half of the outer diameter).
- **Radius to Driving Wheels** (R_{driving}): 15 cm (distance from the center of the base to the driving wheels).

2. **Determine Wheel Placement:**

- **Caster Wheels Placement**: At the outer edge of the ring, 25 cm from the center.
- **Driving Wheels Placement**: 15 cm from the center of the base.

3. **Calculate Lever Arms for Weight Distribution:**

- The lever arm principle states that the force exerted on a wheel is inversely proportional to its distance from the center of gravity (assuming it is at the geometric center of the base).
- **Lever Arm to Caster Wheels** (R_{caster}): 25 cm.
- **Lever Arm to Driving Wheels** (R_{driving}): 15 cm.

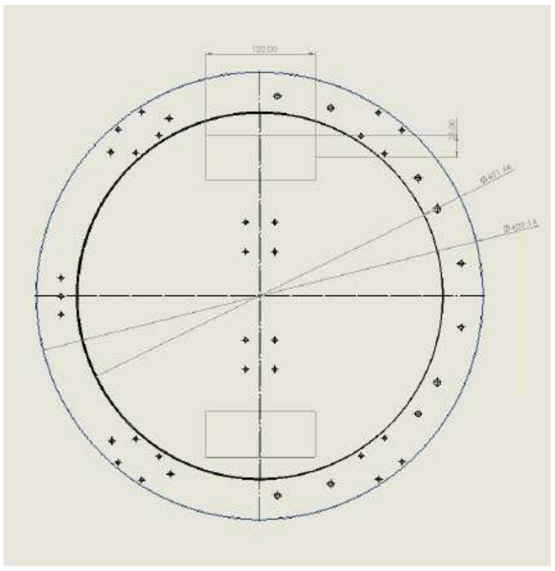


Figure 6.10: Disk Dimensions

4. Estimate Weight Distribution Using Proportional Distribution: Calculate the proportional distribution based on lever arms:

$$F_{\text{caster}} = \left(\frac{R_{\text{driving}}}{R_{\text{caster}} + R_{\text{driving}}}\right) \times \text{Total Weight}$$

$$F_{\text{driving}} = \left(\frac{R_{\text{caster}}}{R_{\text{caster}} + R_{\text{driving}}}\right) \times \text{Total Weight}$$

Substituting the distances and total weight ($W = 50\,\text{kg}$):

$$F_{\text{caster}} = \left(\frac{7.5}{40}\right) \times 50 \approx \mathbf{18.75}\,\text{kg}\,(\text{for caster wheel, 4 wheels total})$$

$$F_{\text{driving}} = \left(\frac{25}{40}\right) \times 50 \approx \mathbf{31.25}\,\text{kg total for driving wheels}$$

32.75% of the force is on the caster wheels and 62.5% of the force is on the driving wheels.

Spring Selection

Determine the Load on Each Wheel

Since the robot has four caster wheels and the load on them combined is 32.75% of the total weight, the load on each caster wheel is:

$$\text{Load per Caster Wheel} = \frac{18.75}{4} = 4.6875\,\text{kg}$$

For the driving wheels, with 62.5% of the weight distributed equally between two wheels:

$$\text{Load per Driving Wheel} = \frac{31.25}{2} = 15.625\,\text{kg}$$

Assume Desired Compression and Spring Constant Calculation

The ±10 mm compression specification helps account for variability in ground conditions, such as concrete impurities, ensuring that the robot can maintain stability and effective maneuverability over different surfaces.

Calculate Spring Constant for Each Wheel

$$k_{\text{caster}} = \Delta x \times F_{\text{caster}} = 0.01 \times 4.6875 \times 9.81 \approx 4596.4\,\text{N/m}$$

$$k_{\text{driving per spring}} = \frac{\Delta x \times F_{\text{driving}}}{2} = \frac{0.01 \times (15.625 \times 9.81)}{2} \approx 7660.4\,\text{N/m}$$

- **Spring Constant for Caster Wheels**: Approximately 4596 N/m per spring.
- **Spring Constant for Driving Wheels**: Approximately 7660 N/m per spring, with each driving wheel supported by two springs sharing the load.

Motor Selection

Parameters and Constants

- **Mass (m):** 35 kg
- **Gravitational Acceleration (g):** 9.81 m/s^2
- **Coefficient of Friction (µ):** 0.7
- **Wheel Radius (r):** 0.05 m (5 cm)
- **Maximum Acceleration (a):** 0.5 m/s^2
- **Desired Speed Limit (v):** 1 m/s

Calculations Normal Force (N) and Maximum Weight Force (Ftm)

$$F_{\text{tm}} = m \times g = 35 \times 9.81 = 343.35\,\text{N}$$

Frictional Force (Ff)

$$F_f = N \times \mu = 343.35 \times 0.7 = 240.345\,\text{N}$$

Force of Inertia (Fi)

$$F_i = m \times a = 35 \times 0.5 = 17.5\,\text{N}$$

Traction Force (Ft)

$$F_t \geq F_f + F_i = 240.345 + 17.5 = 257.845\,\text{N}$$

Motor Moment (Torque, Mm)

$$M_m = F_t \times r = 257.845 \times 0.05 = 12.89225\,\text{Nm}$$

Angular Velocity (ω) of the Motor

$$\omega = \frac{v}{r} = \frac{1}{0.05} = 20\,\text{rad/s}$$

Motor Speed (n)

$$n = \omega \times \left(\frac{30}{\pi}\right) = 20 \times \left(\frac{30}{\pi}\right) = 200\,\text{RPM}$$

Power Required by the Electric Motor (P)

$$P = M_m \times \omega = 12.89 \times 20 = 257.8\,\text{W}$$

For a single motor:

$$P = 128.9\,\text{W}$$

6.3.3 Applied Standards

In the design and development of the locomotion system for SHARE-C, critical engineering and construction standards were adhered to ensure the robot operates reliably in hospital environments. Here are the specific standards applied:

- **ASME Y14.5 - Dimensioning and Tolerance:** This standard was crucial in defining precise mechanical design specifications and tolerances for the locomotion system. By following ASME Y14.5, we ensured that all mechanical components were designed and manufactured to meet stringent accuracy requirements, vital for the assembly and functional operation of SHARE-C.
- **ACI 117 - Specifications for Tolerances for Concrete Construction and Materials:** Considering that SHARE-C will be navigating hospital corridors and rooms with concrete flooring, adherence to ACI 117 was essential. This standard guided us in accommodating floor flatness and levelness tolerances, optimizing the robot's design for navigating slight irregularities without compromising stability or performance by adding the designed springs in each wheel.

These standards were not only referenced but actively incorporated into the design process, influencing decisions related to component selection, system layout, and overall robotic mobility strategies. Their application has significantly contributed to the project's success by enhancing the robot's adaptability and performance constraints in its intended operational environment.

6.4 Self-Disinfection System

This chapter presents preliminary design alternatives for the placement of spray nozzles within the automated self-disinfection system integrated into the healthcare robot. Three distinct CAD designs were developed, each exploring different nozzle arrangements to maximize disinfectant coverage across the robot's surfaces. Computational Fluid Dynamics (CFD) simulations using ANSYS were conducted on each design to evaluate performance, ensure optimal disinfectant distribution, and ultimately select the most effective solution.

6.4.1 Design Alternatives

Design Alternative 1: Hand-End and Upper Back Nozzle Placement
This design includes three nozzles in total. Two nozzles are placed at the end of each robotic arm, directed inward toward the robot's torso. The third nozzle is mounted at the upper back, directed downward, ensuring coverage of the back and sides. This configuration aims to leverage arm movement for enhanced disinfectant distribution.

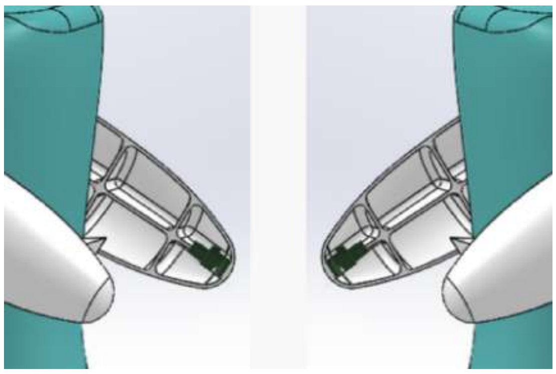


Figure 6.11: Design Alternative 1: Nozzles Positioned at the End of the Robot's Arms.

Design Alternative 2: Circular Bottom-Up Nozzle Arrangement
In this alternative, all five nozzles are arranged in a circular pattern around the robot's base, with spray directions oriented upward. This arrangement capitalizes

on upward spraying patterns to achieve comprehensive coverage from the lower sections upward across the robot's entire body surface.

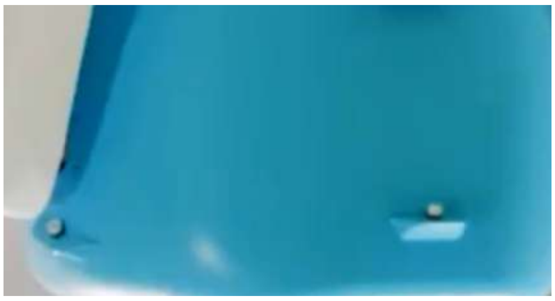


Figure 6.12: Design Alternative 2: Nozzles Arranged in a Circular Pattern at the Lower Part of the Robot.

Design Alternative 3: Shoulder-Level and Angled Back Nozzle Placement.
This configuration involves four nozzles placed in a circular pattern at shoulder height, directed downward to comprehensively disinfect the robot's body. A fifth nozzle is positioned at the robot's upper back, angled upward toward the head to ensure the head and upper surfaces are thoroughly disinfected. This design explicitly addresses the requirement for full-body coverage, including the head.

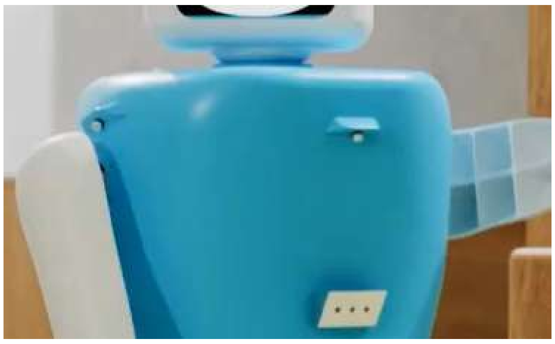


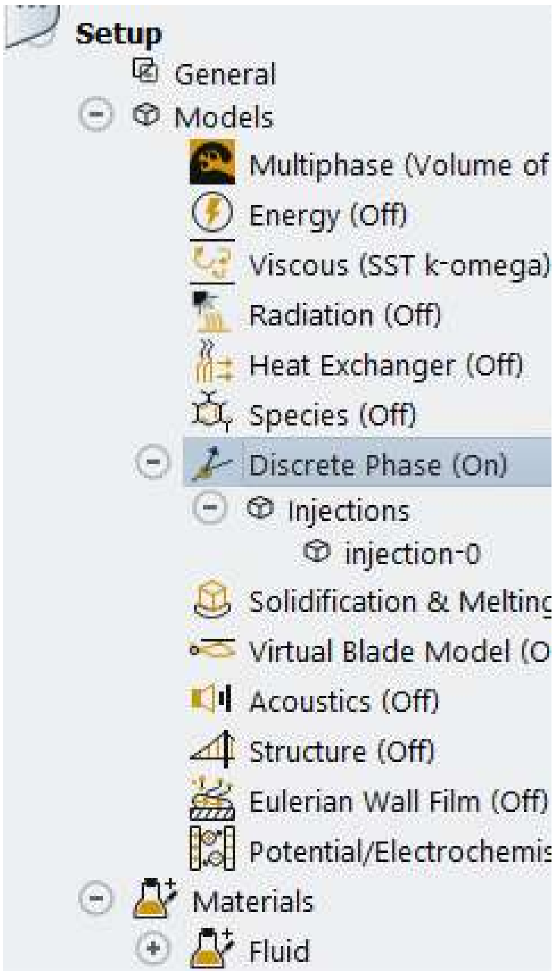
Figure 6.13: Design Alternative 3: Nozzles Positioned at Shoulder Level and Back.

6.4.2 Design Decisions

ANSYS Simulation Setup and Justifications

1. **Model Setup:**
   - **Volume of Fluid (VOF):** Captures the fluid-air interface dynamics between disinfectant droplets and air, essential for accurately modeling the multiphase spray phenomenon.

- **Discrete Phase Model (DPM):** Tracks individual droplets' behavior and interactions, evaluating droplet trajectories, evaporation, and deposition on surfaces.



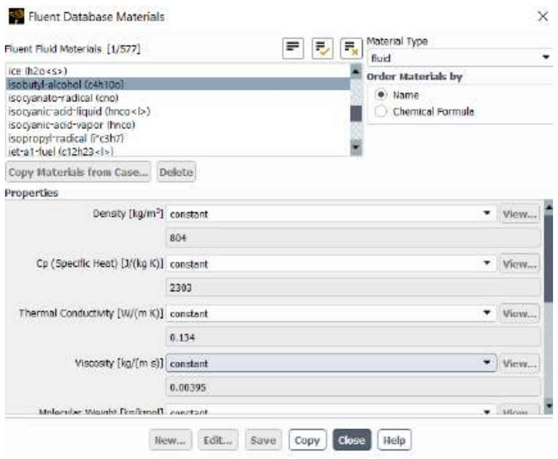
Setup
General
Models
Multiphase (Volume of
Energy (Off)
Viscous (SST k-omega)
Radiation (Off)
Heat Exchanger (Off)
Species (Off)
Discrete Phase (On)
Injections
injection-0
Solidification & Meltinc
Virtual Blade Model (O
Acoustics (Off)
Structure (Off)
Eulerian Wall Film (Off)
Potential/Electrochemis
Materials
Fluid

Figure 6.14: ANSYS Fluent Setup: Volume of Fluid (VOF) and Discrete Phase Model (DPM) Enabled.

2. **Boundary Conditions:**

   - **Velocity Inlets (Sprinklers):** Defined with an inlet velocity matching the calculated nozzle flow rate of 0.096 L/min per nozzle, ensuring realistic spray patterns.
   - **Pressure Outlet (Open Environment):** Allows fluid exit without artificial restrictions, realistically replicating an open atmospheric environment.
   - **Walls (Robot's Surface):** Modeled explicitly as walls to capture droplet deposition, accumulation, and surface interaction.

3. **Flow Rate and Coverage:**
    - **Total calculated flow rate:** 0.48 L/min (0.008 L/s total, equally distributed among five nozzles)
    - **Individual nozzle flow rate:** 0.096 L/min each, based on initial system calculations.
4. **Solver Type:**
   Pressure-based Solver: Chosen due to incompressible multiphase fluid conditions (disinfectant solution and air) at atmospheric pressure.
5. **Solution Methods:**
    - **Spatial Discretization (Second-order upwind):** Selected to accurately capture the droplet trajectory and surface coverage patterns, minimizing numerical diffusion and ensuring accurate simulation results.
    - **Temporal Discretization (First-order implicit):** Provides numerical stability in transient simulations involving droplet-air interactions.
6. **Gravity Effects:**
   Gravity (negative Y-direction, 9.81 m/s^2): Realistically simulates droplet settling due to gravity, essential for accurately predicting droplet trajectories and ensuring reliable simulation results.
7. **Material:**
   Isobutyl Alcohol (Fluent Database): Selected based on its suitable viscosity (0.00395 kg/m·s), density (804 kg/m^3), and evaporation characteristics, ensuring realistic simulation of disinfectant evaporation and surface interaction behaviors.



Fluent Database Materials

Fluent Fluid Materials [1/577]

ice (h2o<s>)
isobutyl-alcohol (c4h10o)
isocyanato-radical (cno)
isocyanic-acid-liquid (hnco<l>)
isocyanic-acid-vapor (hnco)
isopropyl-radical (i*c3h7)
jet-a1-fuel (c12h23<l>)

Material Type: fluid

Order Materials by: [● Name] ○ Chemical Formula

Properties

Density [kg/m3]	constant
	804
Cp (Specific Heat) [J/(kg K)]	constant
	2303
Thermal Conductivity [W/(m K)]	constant
	0.134
Viscosity [kg/(m s)]	constant
	0.00395
Molecular Weight [kg/kmol]	constant

Figure 6.15: Material Selection in ANSYS Fluent: Isobutyl Alcohol.

8. **Mesh Setup:**
   Mesh Refinement: A refined mesh was applied around nozzle areas and robot surfaces, enhancing accuracy in simulating droplet behavior, ensuring reliable analysis of spray patterns and disinfectant coverage.

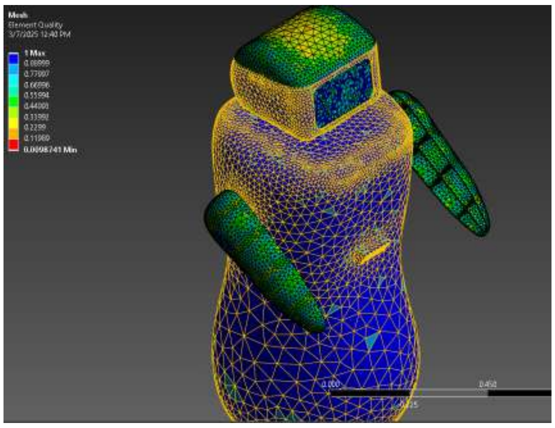


Figure 6.16: Mesh Generation for Full-Robot Self-Disinfection Simulation in ANSYS.

6.4.3 Simulation Results and Design Selection

Design Alternative 1: Hand-End and Upper Back Nozzle Placement
Simulation results revealed limited coverage, primarily disinfecting only the robot's torso region. Significant areas, particularly the head and lower regions, remained inadequately covered, leading to its elimination.

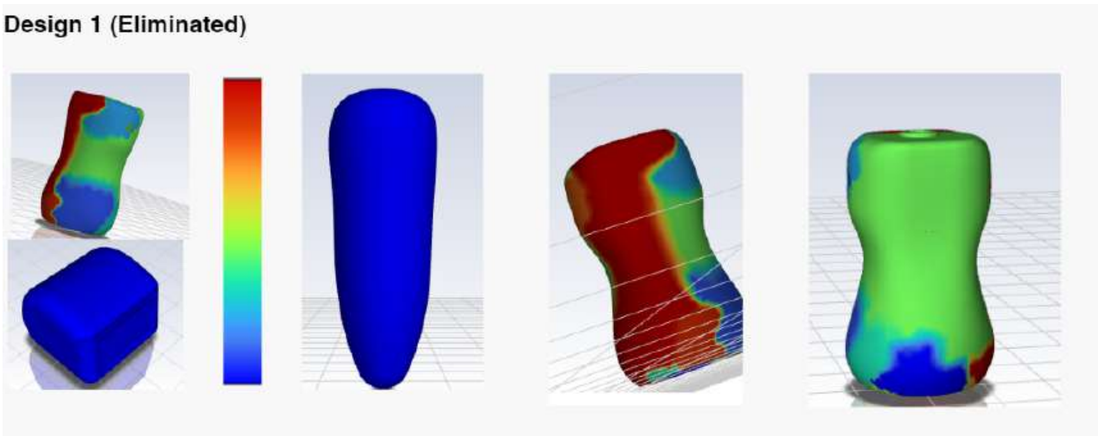


Figure 6.17: ANSYS Simulation Results for Design Alternative 1: Nozzles Positioned at the End of the Robot's Arms.

Design Alternative 2: Circular Bottom-Up Nozzle Arrangement
This design exhibited excellent coverage of lower and middle robot surfaces but showed insufficient disinfectant coverage of the robot's upper regions, including the head and shoulders, leading to its elimination.

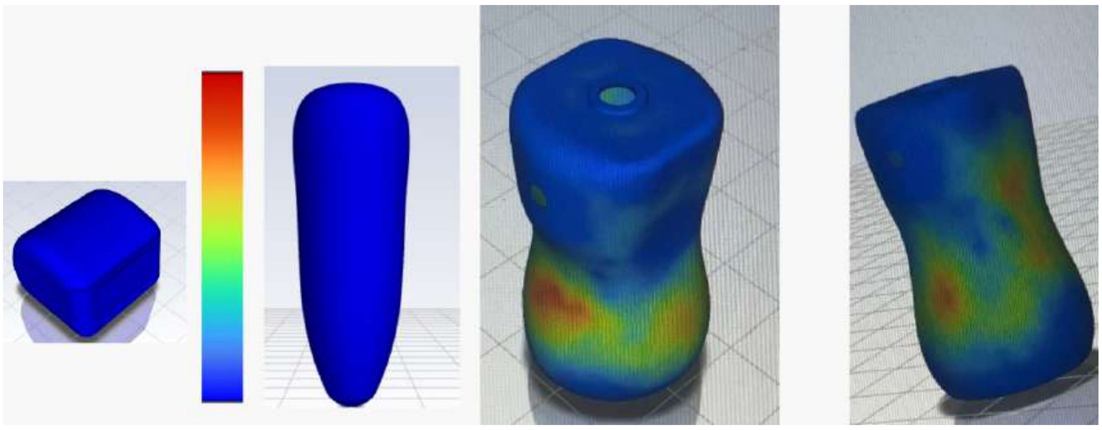


Figure 6.18: ANSYS Simulation Results for Design Alternative 2: Nozzles Arranged in a Circular Pattern at the Lower Part of the Robot.

Design Alternative 3: Shoulder-Level and Angled Back Nozzle Placement.
Simulation results demonstrated comprehensive and uniform disinfectant coverage of the robot's entire body, including the head, shoulders, torso, and lower sections. This design proved optimal for meeting the stringent requirements set by operational constraints and regulatory standards.

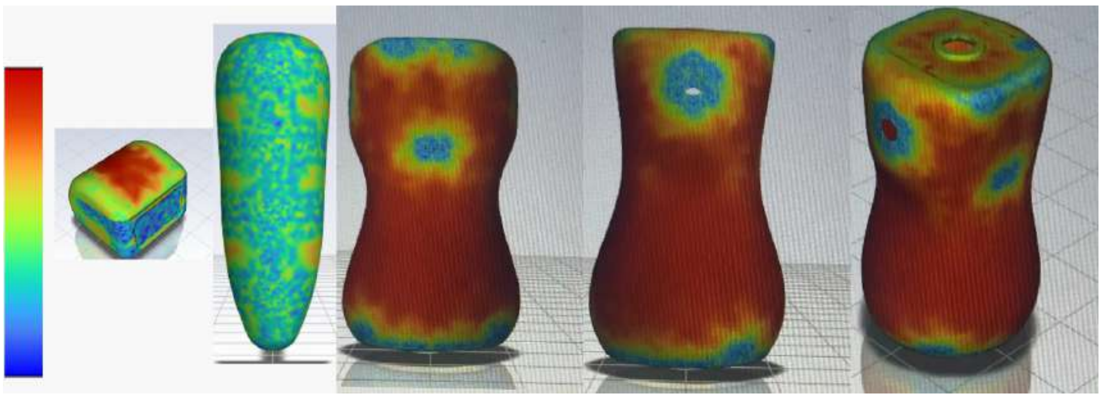


Figure 6.19: ANSYS Simulation Results for Design Alternative 3: Nozzles Positioned at Shoulder Level and Back.

Selected Design

Based on the simulation outcomes, Design Alternative 3 (Shoulder-Level and Angled Back Nozzle Placement) was selected due to its superior performance, effectively meeting all necessary operational, safety, and environmental constraints.

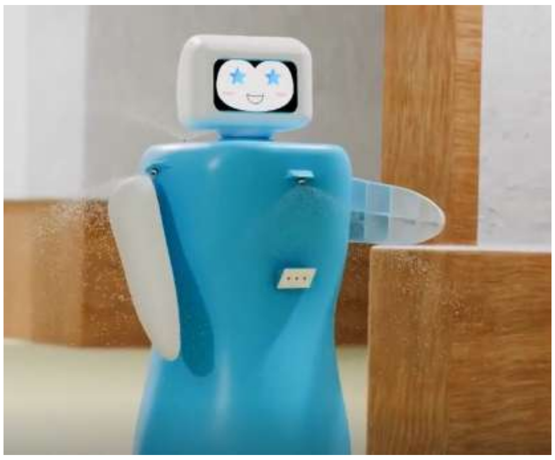


Figure 6.20: Selected Design Alternative 3: Optimized Disinfection Robot in Action

6.5 Door Opening Mechanism

6.5.1 Design Alternatives

The handle was designed to be of lever type (Technical Constraint: Door and Handle Types) and modeled using *Fusion 360*. The design was inspired by and imitates that of an actual door handle, aiming to replicate real-world hospital environments (Technical Constraint: Environment Adaptability). Several alternatives, including push-bar handles and knobs, were initially considered; however, the lever-type was chosen due to its low actuation force and large contact area, making it more compatible with the LoCoBot's limited dexterity and payload.

For manufacturing, PLA material was selected for its lightweight properties, compatibility with *3D printing*, and minimal cost (Non-Technical Constraint: Cost and Budget). This ensured that the final handle remained within the LoCoBot arm's working payload limit of 200g (Technical Constraint: Payload Capacity). Assembly of the parts relied on *adhesive bonding* (Alteco adhesive) and mechanical fasteners (screws) to maintain structural integrity while enabling ease of manufacturability and future modifications (Technical Constraint: Compact Design).

The door frame was designed to include a cavity for integrating a spring mechanism (Technical Constraint: Automatic Door Closing). This was intended to mimic the automatic door-closing behavior typical in hospital settings, ensuring that the interaction with the handle remained realistic without requiring external intervention. Precision operations such as drilling and milling were performed to ensure proper alignment and fitting of mechanical components (Technical Constraint: Precision and Accuracy).

To validate the feasibility of the handle design early on, *ROS* and *Gazebo* simulations were conducted. These simulations allowed testing of the LoCoBot's ability to approach and manipulate basic door models, confirming that the handle dimensions and orientation were within the robot's reach and actuation capabilities.

For handle detection, the initial plan was to utilize the built-in grasping and detection model provided by PyRobot. However, during preliminary trials, it was found that this model was ineffective for detecting door handles mounted at height, as it was designed for detecting planar objects placed on the ground. Consequently, a deep learning approach was adopted, initially using YOLOv8 trained on a custom dataset. Although detection performance was satisfactory, deployment issues with YOLOv8 on Ubuntu 16.04 and ROS Kinetic led to retraining the model using YOLOv3, which provided a compatible and efficient solution for real-time handle detection onboard the LoCoBot.

In terms of pose estimation, relying solely on 2D bounding box outputs was determined to be inadequate for precise grasping, especially for small objects like door handles. Depth information from the LoCoBot's onboard Intel RealSense D435 camera was therefore integrated, allowing conversion of image-based coordinates into accurate 3D grasp poses relative to the robot frame. Functions from the PyRobot package were adapted to facilitate these coordinate transformations, even though the complete grasping model itself was not used.

Regarding door manipulation, initial experiments that relied solely on arm motion to pull the door were unsuccessful. The door's inertia and hinge resistance required more substantial coordinated movement. As a result, synchronized motion between the arm and the mobile base was adopted. Feedback-based control strategies were considered but ultimately ruled out due to the LoCoBot's hardware limitations and absence of a Force/Torque sensor. Open-loop motion along a pre-calculated trajectory was selected instead, using the door's known pivot geometry to compute movement paths that the arm and base would follow during the door-opening task.

In the physical experimental setup, a lightweight foam-based door was manufactured (Figure **??**). Although a rigid or spring-loaded door design was initially considered to better simulate real hospital doors, concerns regarding the LoCoBot's torque, payload capacity, and safety during repeated experiments led to the decision to prioritize a lightweight structure. The door was constructed with a foam leaf and a simple wooden frame, measuring 60 cm in width and 80 cm in height, to fit the LoCoBot's operational footprint (base diameter 40 cm, height 65 cm). The handle was positioned at 40 cm height to align with the robot's reachable workspace, ensuring that the entire system provided a manageable, safe, and repeatable testing platform for validating the door-opening mechanism.

6.5.2 Design Decisions

The handle was designed to be of lever type (Technical Constraint: Door and Handle Types) and modeled using *Fusion 360.* Inspired by real-world door handles, the design aimed to replicate a realistic environment for robotic interaction (Technical Constraint: Environment Adaptability). Several handle types were considered, including push bars and knobs, but the lever design was selected due to its minimal actuation force requirement and larger contact area, which matched the LoCoBot's limited dexterity.

For manufacturing, PLA material was chosen for its lightweight properties, compatibility with *3D printing* processes, and minimal cost (Non-Technical Constraint: Cost and Budget), ensuring that the final handle remained within the LoCoBot arm's working payload limit of 200g (Technical Constraint: Payload Capacity). To ensure precision during assembly, *drilling* operations were performed using a stationary drill press, and the spring cavity was refined through *vertical milling* to guarantee a proper fit and function (Technical Constraint: Precision and Accuracy). Assembly relied on *adhesive bonding* (Alteco adhesive) and mechanical fasteners (screws), balancing manufacturability with structural durability (Technical Constraint: Compact Design).

The door frame was designed to integrate a spring mechanism to mimic automatic door-closing behavior (Technical Constraint: Automatic Door Closing), replicating the expected real-world dynamics of hospital doors without requiring external intervention. Throughout this design process, ROS and Gazebo simulations were used to validate the LoCoBot's ability to approach and manipulate the handle, ensuring alignment between the physical prototype and the robot's reach and actuation capabilities.

For handle detection, the initial plan was to leverage the built-in grasping and detection model provided by PyRobot. However, during preliminary trials, it became evident that this model was inadequate for detecting handles mounted above ground level, as it was designed primarily for planar, ground-placed objects. Consequently, a deep learning-based approach was adopted, and a YOLOv8 model was initially trained on a custom dataset of door handle images. Although detection accuracy was satisfactory, compatibility issues with the LoCoBot's Ubuntu 16.04 and ROS Kinetic setup prevented deployment. As a result, the detection system was retrained using YOLOv3, which offered real-time inference with full system compatibility.

In terms of pose estimation, early approaches relying solely on 2D bounding box centers were found to be insufficient for robust grasping. To enhance accuracy, depth information from the LoCoBot's onboard Intel RealSense D435 camera was integrated, enabling the conversion of image-based detections into 3D grasp poses relative to the robot's base frame. Functions from the PyRobot package were adapted to support these coordinate transformations, even though the full built-in grasping stack was not used.

Regarding door manipulation, initial experiments using only the robotic arm to pull the door proved ineffective. The door's inertia required a more coordinated motion strategy. Although feedback-based control was considered, the LoCoBot's limited sensing capabilities and lack of Force/Torque sensors made real-time adaptive strategies impractical. Therefore, an open-loop motion plan was selected, based on a pre-calculated trajectory that considered the door's pivot geometry. Furthermore, synchronized movement of the arm and mobile base was implemented, allowing the robot to apply consistent force and successfully traverse through the opened door.

In the physical experimental setup, a lightweight foam-based door was manufactured. Although more rigid or spring-loaded door designs were initially explored to better mimic hospital doors, safety considerations and the LoCoBot's limited payload and torque capabilities necessitated a lightweight structure. The door featured a foam leaf supported by a wooden frame, measuring 56 cm in width and 80 cm in height to match the robot's dimensions (base diameter of 40 cm and height of 65 cm). The handle was positioned at a height of 40 cm to ensure accessibility by the LoCoBot's arm, providing a practical and safe platform for validating the developed door-opening mechanism.

6.5.3 Design Iterations

The first 3D-printed prototype of the door handle exhibited rough surface finishes, necessitating additional abrasive processes. Sanding with sandpaper and grinding using a Dremel tool were applied to smooth the surfaces and ensure better functionality. During the spring integration phase, it became evident that the initial wooden door frame was too thin to accommodate the planned spring mechanism for automatic closing. To address this, a thicker slab of wood was cut using a band saw, the components were joined with adhesive bonding, and additional milling was performed to create a proper cavity for the spring. Each iteration refined the physical assembly, bringing the system closer to operational readiness.

In parallel, improvements were made to the handle detection pipeline. YOLOv8 was initially trained on a custom dataset of door handle images, but deployment difficulties arose due to incompatibility with the LoCoBot's Ubuntu 16.04 and ROS Kinetic configuration. Consequently, the detection system was retrained using YOLOv3, which offered robust real-time performance and was successfully deployed on the robot. Approximately 400 labeled images were used to specialize the YOLOv3 model for the manufactured door handle, improving localization reliability in the experimental environment.

Pose estimation also underwent refinement. Early methods based solely on 2D bounding box centers proved unreliable for consistent handle grasping. To address this, depth information from the RealSense D435 camera was integrated, allowing accurate computation of 3D grasp poses relative to the robot's frame.

The door manipulation strategy evolved through several stages. Initial experiments that relied solely on the robotic arm to pull the door were unsuccessful in achieving full door opening. Consequently, coordinated motion between the arm and the mobile base was introduced. The first implementation involved manually hardcoded sequences for both arm and base movement; however, these motions lacked precision and consistency. Further iterations explored alternative solutions, ultimately leading to the adoption of a trajectory generation method based on the mathematical equations presented in [45]. This method provided a more systematic and reliable approach for synchronized arm and base movement during door opening.

6.5.4 Applied Standards

The design and development of the various components of the door-opening mechanism for SHARE-C adhered to internationally recognized standards as mentioned in Chapter 3 to ensure safety, quality and functionality.

For instance, the standard **ISO 13485** was reflected in the design and manufacturing processes of the door handle responding to regulations on medical devices. Iterative refinements such as sanding and grinding the 3D-printed handle yielded an improved surface finish to ensure quality. Also, adjusting the wooden frame to properly accommodate the spring mechanism ensures the functionality of the handle design. In general, each step in the process, including Gazebo simulations and physical implementation, was systematically planned and documented to maintain reliability and consistency, which are critical in designs involving medical contexts.

The principles of **ISO 13482** influenced the safety-focused design of the door-opening mechanism. For example, the PLA material used for the 3D-printed handle kept the weight within the LoCoBot arm's payload limit of 200g, minimizing the risk of mechanical failure. The spring mechanism integrated into the design ensured smooth and controlled manipulation, safeguarding both the robot and its environment.

Finally, **ASME Y14.5** guidelines were applied to ensure precise mechanical design and assembly. This was particularly evident in the design of the handle in Fusion 360, where tolerances were defined to ensure efficient assembly between the parts. Manufacturing processes like drilling and milling were also executed with precision to achieve the required fit and functionality.

6.6 Graphical User Interface

6.6.1 Design Alternatives

Several alternatives were considered for building the user interface. Given the vast options of software systems used in this field, the choice should be studied warily to make sure that it aligns with the needs of this project. The first option is using Flutter to design the front end, which was taken into consideration due to its rapid development, compatibility with several platforms, and highly customizable UI components, however, it requires familiarity with the Dart programming language. Another option is using native development software like Kotlin/Swift, which provides optimized and high-level performance for both Android and iOS, but it requires a significant development time due to its complexity, since it needs separate code bases. To this end, **Flutter** is chosen as the project's front-end to ensure a simple integration of the frontend system and avoid any unwanted complexities. As for the database system, **MongoDB** is chosen over the other SQL alternatives such as PostgreSQL and MySQL due to the provided flexibility in handling unstructured data, seamless integration with the backend system, and scalability[57]. The provided document-based structure in MongoDB is the ideal choice for this application, needed for scheduling tasks, nurses' logs, and order updates, without needing the rigid schemas in traditional SQL databases.

The constraints mentioned in Chapter 6 are present to enhance safety and usability in healthcare facilities. The most crucial constraint is ethical considerations, where it is indispensable to make sure that the database system doesn't store any patient information or data, thus meeting the privacy regulations. Moreover, building an app that could be downloaded for free from the app store allows anyone with a phone to download the app and use the robot, minimizing any additional hardware investments. Also, taking into account the cultural and social impact, the app must align with the Lebanese healthcare settings.

6.6.2 Design Decisions

To ensure the user application is both functional and intuitive, a minimalistic layout is implemented to prioritize easy navigation and assuage the process of ordering a task from the robot. The design decision includes displaying the prominent features such as the robot's status, real-time location, and schedule display. In addition, the app showcases the robot's location and battery charge level, all while ensuring consistency and maintaining the same design throughout the app to avoid confusion. Furthermore, the login and sign-up pages must be designed with an email validation system to restrict access to only authorized users, ie: healthcare staff. For this application, only users with email: @mail.aub.edu are allowed. These features ensure prioritizing staff and patient safety, as they allow constant monitoring of the robot's status to take the necessary measures in case of any problem arising. As for the scheduling system, the user must be able to visualize the tasks in progress, pending tasks, and finished tasks for each day. This ensures that the nurses can easily track the robot's mission throughout the day. Lastly, a new face for SHARE-C must be designed along with a small speech to greet patients, all in a Lebanese accent to enhance user engagement and provide emotional reassurance.

6.6.3 Design Iterations

As with any application, several iterations and updates must be made to ameliorate functionality and usability. In the preliminary Figma design, the number of pages was redundant; thus, in the first iteration, the number of pages was streamlined to only include the critical tasks and functions to ease user navigation. In the second iteration, the designs of the login and sign-up pages were refined to ensure a more visually pleasing and user-friendly interface.

6.6.4 Applied Standards

According to the ISO/IEC 62366: "Usability Engineering for Medical Devices", the application must have a user-centered design while focusing on safety-critical tasks, such as designing a fail-safe workflow by ensuring that tasks can't conflict with each other so as not to disrupt hospital operations. Furthermore, to avoid an overwhelming interface, the colors used are minimal, and icons are carefully chosen to represent different features, thus enhancing the friendliness of the user interface and guaranteeing an easy navigation process throughout the app, especially for new users. By incorporating risk management principles, the app guarantees its alignment with ISO 14971, an international standard ensuring reliability and safety in a medical environment. The app includes a real-time status update with a map showing the robot's location, along with emergency notifications in case any risk is present. Thus, by allowing continuous monitoring of the robot through the user interface, the nurse is always in control, and any risk is immediately taken care of. Lastly, the app also implements hashing of nurses' passwords and stores the login data securely according to storage protocols, thus ensuring the utmost compliance with ISO 14971.

Chapter 7

Implementation and Testing

7.1 Sensing and Perception

7.1.1 Implementation

The implementation phase focused on deploying RTAB-Map SLAM for autonomous navigation of SHARE-C, in accordance with the project requirements of operating solely with an RGB-D camera (Intel RealSense D435i). RTAB-Map was selected for its compatibility with RGB-D sensors, built-in visual odometry, and support for both mapping and localization within a single ROS-based framework, making it especially suitable for embedded hardware like the Jetson TX2.

7.1.2 Experiment Setup and Testing

The process began with mapping and localizing the robot in a corridor of the Oxy building, chosen as an initial test environment due to its similarity to hospital hallways. However, the corridor quickly revealed major challenges for vSLAM: it was extremely feature-sparse, with long blank walls, minimal furniture or texture, and variable lighting caused by large windows. These conditions are known to degrade the performance of feature-based SLAM algorithms like RTAB-Map.

To assess whether these failures stemmed from software misconfiguration or environmental limitations, several steps were taken:

- The environment was mapped multiple times under different lighting conditions (day and night).
- The resulting occupancy grid maps were cleaned and refined using GIMP to remove noise and fill gaps.
- The cleaned maps were re-used for localization testing using RTAB-Map's localization mode.

Despite these efforts, the robot consistently failed to localize reliably in the corridor. Upon further debugging, it was discovered that transform inconsistencies between `base_link`, `camera_link`, and other relevant TF frames were causing additional failures. These were resolved by defining all static transforms explicitly using ROS static transform publishers.

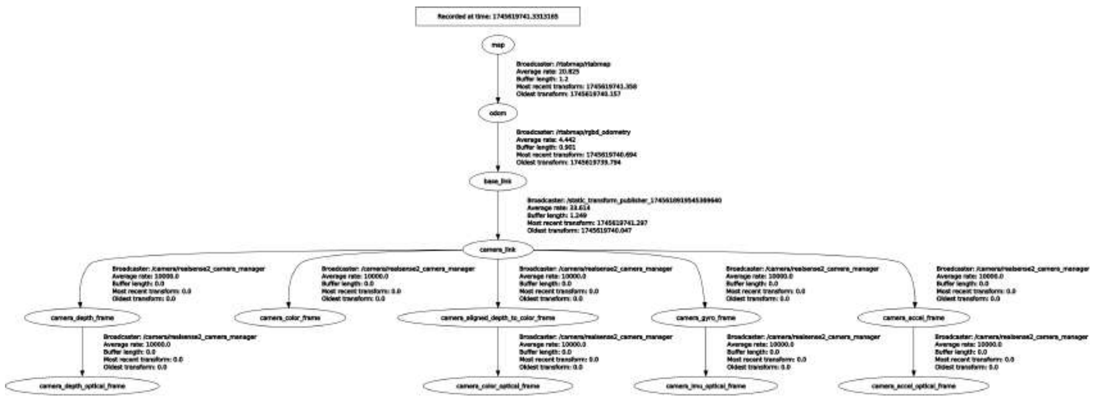

Recorded at time: 1745619741.3313165
map
Broadcaster: /rtabmap/rtabmap
Average rate: 20.825
Buffer length: 1.2
Most recent transform: 1745619741.358
Oldest transform: 1745619740.157
odom
Broadcaster: /rtabmap/rgbd_odometry
Average rate: 4.442
Buffer length: 0.901
Most recent transform: 1745619740.694
Oldest transform: 1745619739.794
base_link
Broadcaster: /static_transform_publisher_1745610919545389640
Average rate: 23.614
Buffer length: 1.249
Most recent transform: 1745619741.297
Oldest transform: 1745619740.047
camera_link
Broadcaster: /camera/realsense2_camera_manager
Average rate: 10000.0
Buffer length: 0.0
Most recent transform: 0.0
Oldest transform: 0.0
camera_depth_frame
camera_color_frame
camera_aligned_depth_to_color_frame
camera_gyro_frame
camera_accel_frame
camera_depth_optical_frame
camera_color_optical_frame
camera_imu_optical_frame
camera_accel_optical_frame

Figure 7.1: Transforms

To conclusively determine whether the remaining issues were environmental or system-related, a final implementation test was performed in the **VRL lab**, a visually richer indoor environment with more diverse textures, lighting conditions, and geometry. After configuring the corrected TF tree and launching the vSLAM pipeline in the VRL lab, the robot was able to:

- Successfully map the environment using RTAB-Map SLAM
- Localize within the generated map using visual odometry

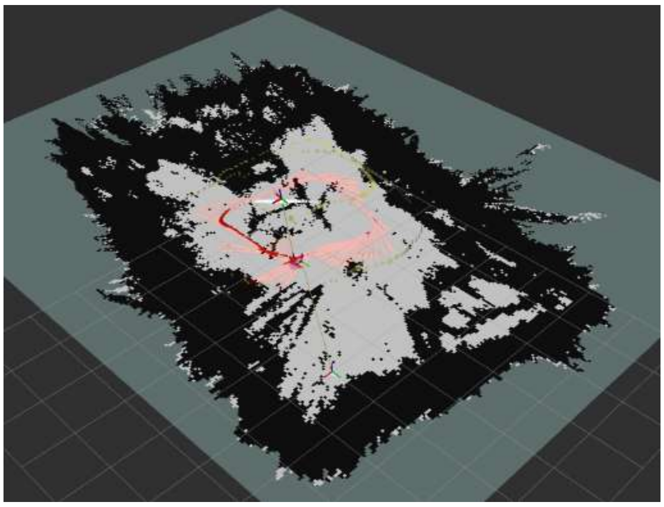


Figure 7.2: Robot localized correctly in mapped VRL room.

As shown in Figure 7.2, the robot is successfully localized within the map. The red arrows represent the visual odometry estimates, closely following the robot's path. Their consistency and alignment indicate that the robot maintains an accurate estimate of its position throughout its movement, confirming that localization is functioning correctly.

These results validated that the implementation was technically sound and that the system was capable of performing real-time SLAM and localization on the Jetson TX2, provided the environment contained sufficient visual features.

7.1.3 Discussion – Final Design Assessment

The final implementation successfully met many of the initial project constraints: using a camera-only setup and real-time localization. RTAB-Map proved capable of generating accurate maps and supporting localization under proper environmental conditions.

However, the corridor environment exposed limitations in the feature-based visual SLAM approach, particularly in spaces with low visual diversity and non-uniform lighting. Although this does not reflect a failure of the system design, it reveals important practical constraints when deploying in minimalistic indoor settings.

Future iterations may involve enhancing localization robustness by:

- Introducing more artificial features in such environments
- Switching to direct methods with computational trade-offs

These reflections underline the importance of aligning system design not only with hardware capabilities and algorithmic strengths, but also with the environment the robot will be deployed in.

7.2 Path Planning and Control

7.2.1 Implementation

ROS/Gazebo Simulations

Path Planning started as simulations on Gazebo during the Fall semester then switched to deploying on hardware in Spring Semester.

Locomotion System Analysis and Design Implementation

Kinematic and dynamic modelling of the TWDDR are done to apply model based control when working with the hardware.

Plant Representation: The approach adopted in this task was model-based control. Thus, a kinematic and dynamic model was derived.

- **Kinematic Model:** In [58], Two coordinate frames were defined to model the differential drive robot: The inertial coordinate system denoted as:

  $$\{X_I, Y_I\}$$

  and the robot coordinate system represented by:

  $$\{X_r, Y_r\}$$

  . Then, the robot's position and orientation in the inertial frame is defined by the following matrix:

  $$q^I = \begin{bmatrix} x_a \\ y_a \\ \theta \end{bmatrix} \tag{7.1}$$

  The two coordinates frames are related by the following orthogonal rotation matrix:

  $$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7.2}$$

  In Figure 7.1, the inertial and robot coordinate systems are illustrated. Point A, the midpoint between the two wheels, is referred to as the origin of the robot frame. While the center of mass, point C, is assumed to be on the axis of symmetry and at a distance d from A

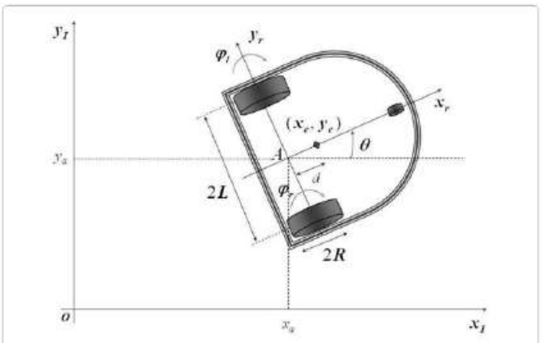


Figure 7.3: TWDDR Coordinate Frame

- **Kinematic Constraints of the Differential Drive Robot:** The differential drive robot has 2 actuators, one for each wheel, therefore, it possesses 2 degrees of freedom. Two non-holonomic constraints restrict the motion of the robot:

    1. No lateral slip motion, which means that the robot is only able to move forward or backward, and not sideward:

    $$\dot{y}_a^r = 0$$

    2. Pure rolling constraint, thus there is no slipping of the wheels in the longitudinal axis

    $$x_r = 0$$

    and no skidding in its orthogonal axis

    $$y_r = 0$$

    The two non-holonomic constraints are represented by the following:

    $$\Lambda(q) = \begin{bmatrix} -\sin\theta & \cos\theta & 0 & 0 & 0 \\ \cos\theta & \sin\theta & L & -R & 0 \\ \cos\theta & \sin\theta & -L & 0 & -R \end{bmatrix}$$

    where:

    $$\dot{q} = \begin{bmatrix} \dot{x}_a & \dot{y}_a & \dot{\theta} & \dot{\varphi}_R & \dot{\varphi}_L \end{bmatrix}^T$$

    with the wheel velocities represented by V_R and V_L:

    $$\begin{cases} v_R = R\dot{\varphi}_R \\ v_L = R\dot{\varphi}_L \end{cases}$$

    The linear velocity of the differential drive robot can then be obtained by the following equation:

    $$v = \frac{v_R + v_L}{2} = R\frac{\dot{\varphi}_R + \dot{\varphi}_L}{2} \tag{7.3}$$

    with the angular velocity represented by:

    $$\omega = \frac{v_R - v_L}{2L} = R\frac{\dot{\varphi}_R - \dot{\varphi}_L}{2} \tag{7.4}$$

    Finally, the forward kinematic model of the differential drive robot can be written as follows:

    $$\dot{q}^I = \begin{bmatrix} \dot{x}_a^r \\ \dot{y}_a^r \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

- Dynamic Model: To model and visualize the motion of the mechanical system, the Lagrangian dynamic approach is implemented below:

  $$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) + \frac{\partial L}{\partial q_i} = F - \Lambda^T(q)\lambda$$

  Where the Lagrangian function is:

  $$L = T - V$$

  with:

  - T: kinetic energy of the system
  - V: potential energy of the system
  - q_i: generalized coordinates
  - F: generalized force vector
  - Λ: constraints matrix
  - λ: vector of Lagrange multipliers associated with the constraints

  The potential energy of the system is considered to be zero, on account of assuming that it is moving only on XY plane. Thus L=T. The total kinetic energy of the system is described by the following equation:

  $$T = \frac{1}{2}\left(m(\dot{x}_c^2 + \dot{y}_c^2) - md\dot{\theta}(\dot{y}_c \cos\theta - \dot{x}_c \sin\theta) + I_c\dot{\theta}^2 + \frac{1}{2}I_w(\dot{\phi}_R^2 + \dot{\phi}_L^2)\right)$$

  with the total inertia:

  $$I = I_c + m_c d^2 + 2m_w L^2 + 2I_m$$

  and the total mass of the robot:

  $$m = m_c + 2m_w$$

  Thus, the dynamic model of the differential drive robot is represented by the following two equations:

  $$\begin{cases} \left(m + \frac{2I_w}{R^2}\right)\dot{v} - m_c d\dot{\omega}^2 = \frac{1}{R}(\tau_R + \tau_L) \\ \left(I + \frac{2L^2}{R^2}I_w\right)\dot{\omega} + m_c d\dot{v} = \frac{L}{R}(\tau_R - \tau_L) \end{cases}$$

Using the relationship between the linear velocity v of each wheel and the applied torque τ given by:

$$\mathrm{v} = \frac{R}{I_w}\tau$$

and substitute it into equations (2) and (3),

A dynamic model of the differential drive as a function of desired input linear velocity v_I and desired input angular velocity is obtained ω_I:

$$\begin{cases} \dot{v} = \frac{R^2 m_c d}{R^2+2I_w}\omega^2 + \frac{2I_w}{R^2+2I_w}v_I, \\ \dot{\omega} = -\frac{R^2 m_c d}{T_w+2L^2 I_w}\omega v + \frac{2I_w}{T_w+2L^2 I_w}\omega_I. \end{cases}$$

Matlab is utilized to linearize and have a useful model outlining the dynamics of the system. The differential drive system's properties are illustrated in Table 7.1.

Icons	Description	Value	Unit
R	Radius of Wheels	0.05	m
L	Distance between the two wheels	0.2531	m
d	Distance between the center of the two wheels and the center of mass	0.0805	m
m_b	Mass of the Chassis	7.7	Kg
m_w	Mass of each Wheel	0.1	Kg
T_w	Moment of Inertia of the wheels	0.0025	$\mathrm{Kg.m}^2$

Table 7.1: System Parameters

The linearized state-space model of the system about the equilibrium point(v = 1 m/s,ω = 7.959 rad/s,$\mathrm{v}_I = -2.02$m/s,ω_I = 1 rad/s):

$$\begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} 0 & 3.358 \\ -4.486 & -0.5637 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} + \begin{bmatrix} 6.6 & 0 \\ 0 & 0.1136 \end{bmatrix} \begin{bmatrix} v_I \\ \omega_I \end{bmatrix}$$

$$\begin{bmatrix} v_R \\ v_L \end{bmatrix} = \begin{bmatrix} 20 & 2.531 \\ 20 & -2.531 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

7.2.2 Experiment Set-Up

- Turtlebot3_world is used to perform simulation/experiment validation. Figure 13.7 below shows the environment used for demonstration.

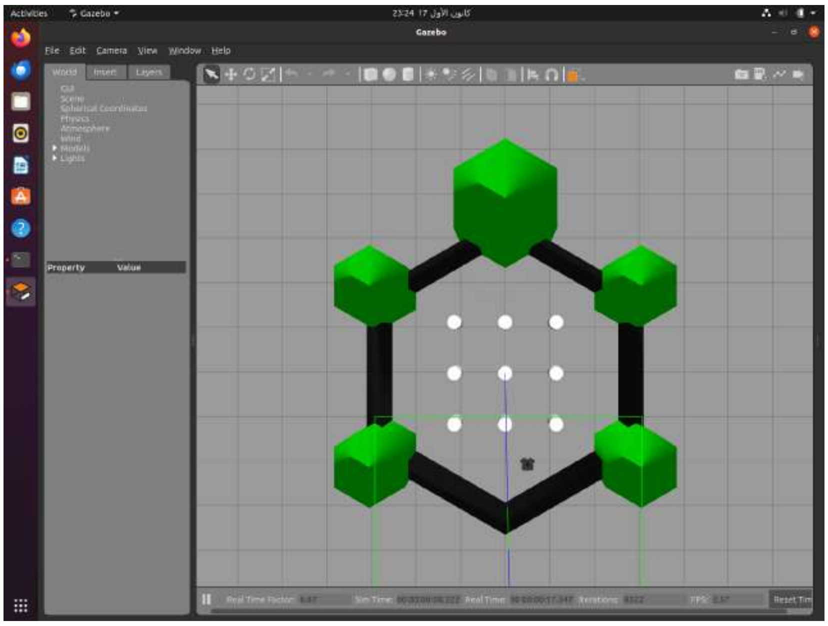


Figure 7.4: TurtleBot3_world

The world consists of static obstacles that might be useful as a first step to navigate between them.

- Intel realsense R-200 sensor is utilized for mapping. Figure 13.8 shows the an image of the sensor. This is the closest to the sensor used in this project.

Figure 7.5: Intel-Realsense r200

Using the above stated packages, SLAM gets triggered and thus a 3D map is generated using the Intel-Realsense R-200 sensor. This map is then loaded for autonomous navigation.

Deploying everything in Hardware

All of the above subsections describe the simulation part of autonomous navigation. Following simulation, hardware implementation on Nvidia Jetson TX2 and Arduino is done. As mentioned in chapter 8, Navigation stack is deployed within the Nvidia Jetson TX2 to achieve autonomous navigation (perception and path planning). It consists of the following architecture:

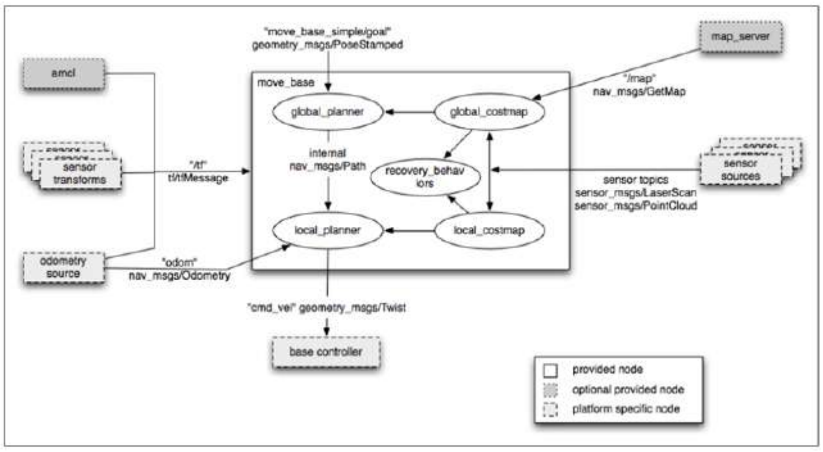


Figure 7.6: Navigation Stack Architecture

- Map_Server Node: Responsible for deploying the map. It publishes
- Move_Base Node: Heart of Path Planning. It subscribes to the /map topic where the map_server node publishes the map data. In addition, move_base requires odometry data or localization node to be active and publishing for it to work. After perception requirements are fulfilled, it is ready to start sending actuator commands and plan a path.
  Several YAML files are included within the parameter folder within the autonomous_navigation ROS package that contain the parameters for path planning. YAML (Yet Another Markup Language) is a human-readable file that configures files and data publishing between ROS nodes.
  - Global_costmap_params.yaml: used for global costmap configuration and long-term path planning.
  - Global_planner_params.yaml: sets up which path planning algorithm to yuse (A* or Dijkistra)
  - Local_costmap_params.yaml: used for local costmap configuration and short-term planning (reaction or obstacle avoidance in dynamic environments)
  - Base_local_planner_params.yaml: responsible for computing actuator commands

  - Dwa_local_planner_params.yaml: configures behavior when DWA is applied

  - Move_base_params.yaml: adjusts the integration between planners and costmaps.

- amcl node : Adaptive Monte Carlo Localization Algorithm is an algorithm that uses . This node is responsible for localizing the robot or publishing odometry data. However, in this project, the RTABMAP-Localization algorithm is used rather than the amcl procedure for geolocation.

Below is a similar implementation of navigation stack on SHAERE-C'S base:

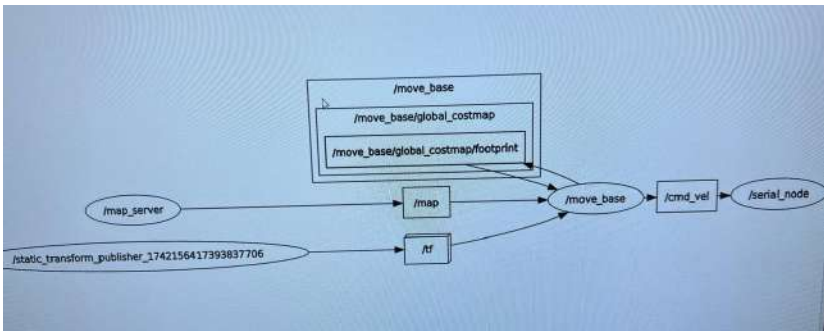

/move_base
/move_base/global_costmap
/move_base/global_costmap/footprint
/map_server
/map
/move_base
/cmd_vel
/serial_node
/static_transform_publisher_1742156417393837706
/tf

Figure 7.7: Navigation Stack Implementation on Nvidia Jetson TX2

The above figure shows succesful communication between the navigation nodes.

The global planner used within navigation stack is A* being compatible with the framework and lower time complexity than Dijkstra (which is the default one). This explains the acceptance of the battery life constraint. Also, the local path planner chosen is DWA being the default as well.

Low Level Control

Abiding by the Precision and Accuracy constraint, low-level PID controllers are built to control the speed of the two motors built on the base of the robot. The role of these controllers is to control linear speed and avoid drifting. The below block diagram summarizes the strategy applied.

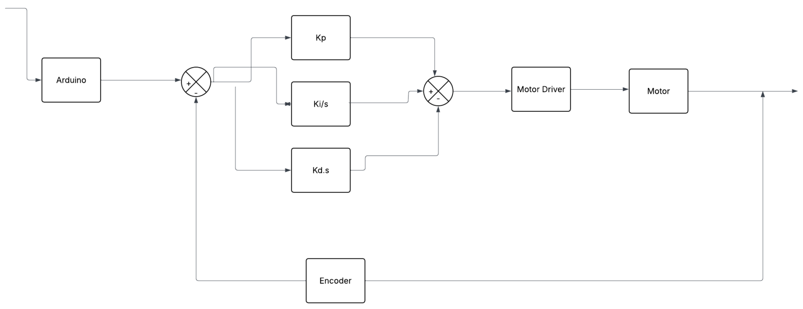


Figure 7.8: PID Block Diagram

The graphs below show the results of low level control of the base with and without load(SHARE-C's body).

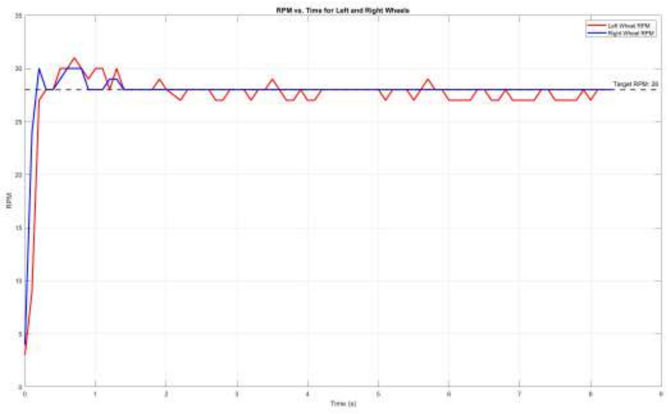


Figure 7.9: PID control results

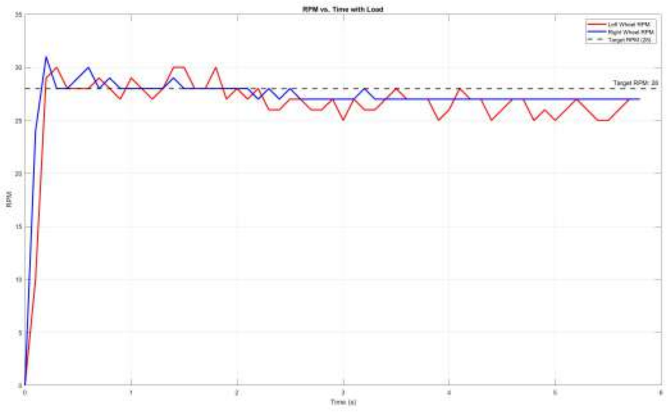


Figure 7.10: PID control results with load

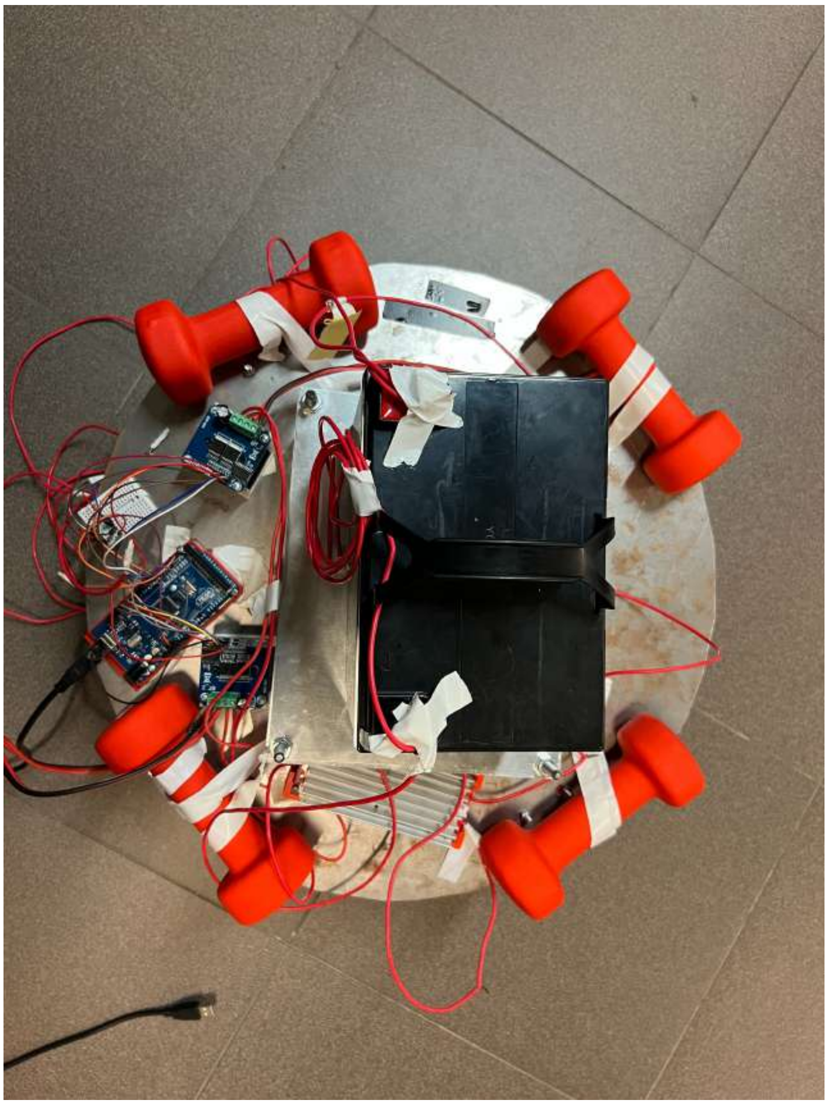


Figure 7.11: SHARE-C's base with Dumbbells

Dumbbels were put on the castor wheel of the base (4 dumbbells , 2Kg each) to mimic SHARE-C's body. Figures 7.7 and 7.8 show decent tracking between the two motors, with a slight steady state error. Thus, SHARE-C's base avoids drifting and navigates to its goal with precision complying with the precision and accuracy requirement.

Path Planning

Move_base, the motion planner of the whole system, translates high level navigation way points into actuator commands. Nvidia publishes the high level coordinates to Arduino via ROSSERIAL. Subsequently, Arduino sends these actuator commands, generated by move_base to the motors.

After mapping the Vision and Robotics lab, the following local costmap resulted from move_base:

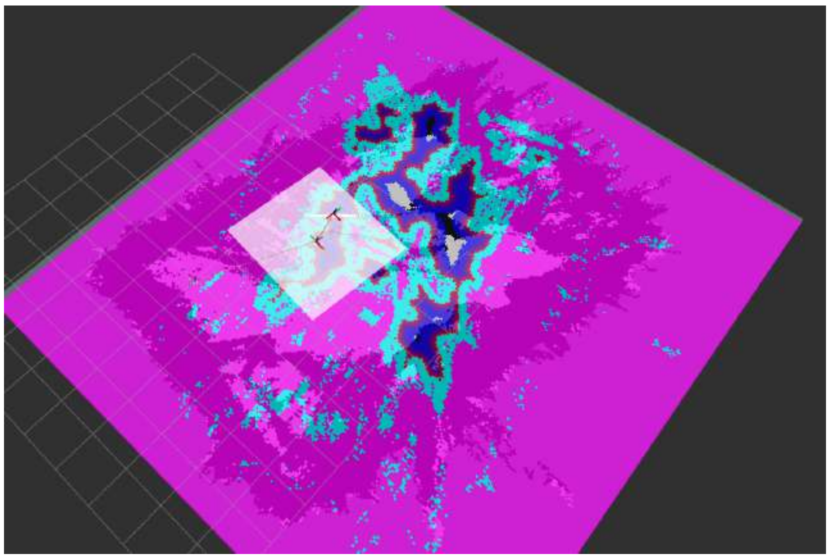


Figure 7.12: Local Costmap of Vision and Robotics Lab

The magenta line indicates unknown space while the cyan indicates uncertain regions. Thus, move_base publishes 0 translation and rotation to the cmd_vel topic since the robot cannot find a path after perceiving itself as stuck. Subsequently, the robot enters recovery mode abiding by the technical constraint even though its sensing phantom obstacles (like sensor noise).It is imperative to note that the region that SHARE-C was deployed in is obstacle free. However, since most of the environment is unknown (magenta) and the known part is uncertain (cyan), move_base cannot proceed with the actuator commands and plan a path which ultimately generates the "Robot is Stuck" command.

In an attempt to debug this discrepancy , the old map (noisy one) was cleaned using Gimp. However, map_server published an inverted map .

.

7.2.3 Final Design Assessments

3D mapping using Intel-Realsense R-200 sensor followed by Autonomous Navigation validates the success of Gazebo simulations. Furthermore, simulations highly resemble real life. In addition to that, respecting the ISO 13485 standard and the precision and accuracy constraints manifest a decent assessment of the engineering design.

Therefore, the following criteria from chapter 5 were met:

- The robot achieves tracking between motors during forward movement and avoids drifting (met)

- Successful communication between navigation nodes (met)
- The robot is able to autonomously navigate and plan a path in a known environment after it has mapped and successfully localized itself(pending)

While the system successfully complied with the design constraints and standards, it could still be improved by remapping the rtabmap-slam's topics and frames to map_server.

7.3 Locomotion System

Following the detailed specifications and component selection procedures outlined in section 9, the implementation phase involved the systematic acquisition and assembly of various parts of the locomotion system.

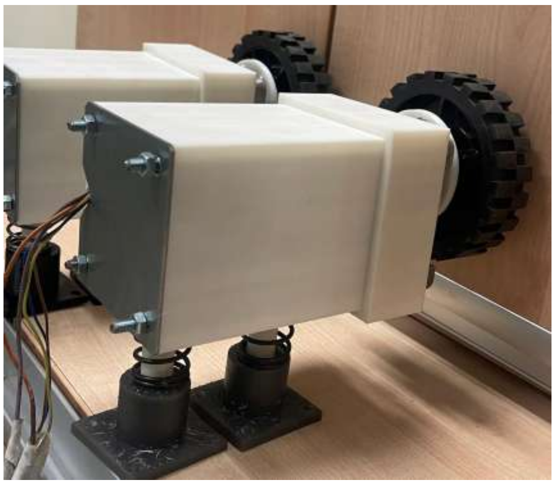


Figure 7.13: Driving motors assembly

7.3.1 Implementation

Custom Components Manufacturing

Motor Housing: Custom housings for the driving motors were created using 3-D printing technology. ABS plastic was selected as the material due to its excellent mechanical properties, which include high impact resistance and toughness. This is crucial as the housing needs to support the weight and vibrations of the motors during operation. Additionally, ABS is suitable for environments where a moderate temperature resistance is required, which is typical in motor operation scenarios. The design of the housing was optimized to allow easy attachment to the suspension system while providing robust support to the motors. This approach was chosen as an alternative to welding, which was unsuitable due to the potential for heat damage to the sensitive electronics within the motors.

Sliders, Slider Rods and Nuts: The sliders of the suspension system were 3-D printed using draft resin. This material was chosen specifically for its rapid prototyping abilities, allowing for quick iterations in the design process without compromising on strength. Draft resin offers a good balance of flexibility and rigidity, which is essential for parts that require both structural integrity and the ability to absorb impacts during movement. The rods and their nuts were meticulously machined using a lathe, and precision threading was applied at 14 mm to ensure seamless assembly between them. The threading process was carefully monitored to maintain high tolerance levels, critical for preventing misalignment and ensuring smooth operation of the sliding mechanism.

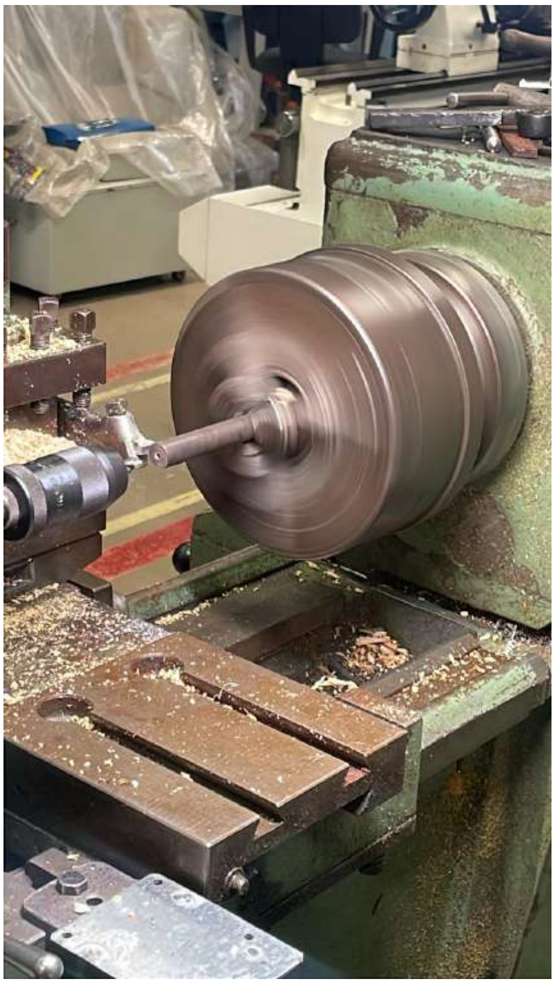


Figure 7.14: Rods and nuts threading on the lathe machine

Springs: The springs, custom-made at a specialized shop in Beirut, required high stiffness with minimal compression to maintain the structural integrity and operational stability of the robot under various loads.

7.3.2 Experiment Setup and Testing

Assembly Process

Disk Modification: The central disk underwent multiple modifications, including precision punching for bolt installation. These modifications were critical to ensure that all components could be securely attached, providing a stable platform for the robot's locomotion system.

Suspension System Assembly: A hydraulic press was utilized to compress the custom springs slightly during pre-assembly. This step was necessary to facilitate easier installation of the springs into the system, ensuring that they would function optimally once the hydraulic pressure was released, allowing the system to maintain proper tension and flexibility.

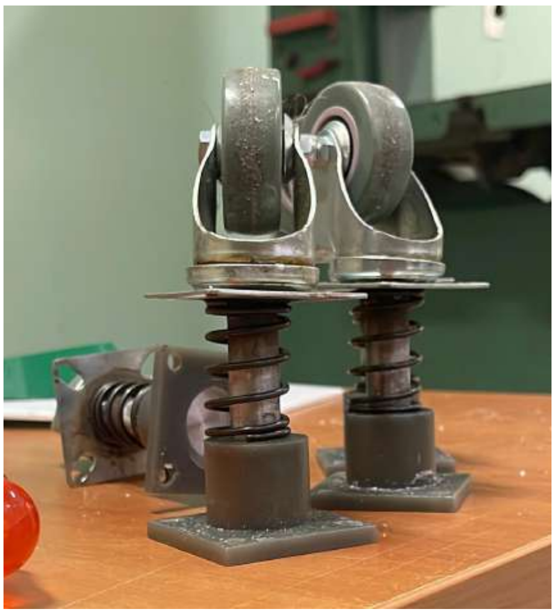


Figure 7.15: Suspension system assembly

Testing and Validation

Initial Testing: The initial testing phase involved a thorough inspection to ensure all components were correctly assembled and functioning as intended. Adjustments were made to optimize the alignment and integration of the locomotion system.

Static Weight Distribution: A distributed static weight of 50 kg was placed on the disk to test the compression of the suspension system. Initially, the compression was measured at 7.9 mm. After applying lubrication between the rods and the slider to reduce friction and wear, the compression increased to 9.4 mm, which aligns perfectly with the operational requirements of our application.

Inclination Stability Testing: Stability tests were performed at different inclination angles (5°, 10°, 15°) to simulate various scenarios the robot might encounter in a hospital setting. The results showed that the locomotion system maintained rigid stability across all tested angles, confirming its robustness and reliability for varied terrain within hospital premises.

Safety Considerations: Recognizing the importance of safety in a hospital environment, special attention was given to smoothing all sharp edges, particularly those of the disks. This precaution helps prevent accidental injuries to hospital staff and patients, enhancing the overall safety of the robot's operational environment.

7.3.3 Discussion - Final Design Assessment

Final Validation: The final validation phase confirmed that the locomotion system met all predefined specifications and was capable of operating under extended use scenarios. Stress tests and continuous operation simulations ensured durability and performance consistency.

7.4 Self-Disinfection System

Following the completion of ANSYS simulations and the selection of the optimal nozzle arrangement (Design Alternative 3: Shoulder-Level and Angled Back Nozzle Placement), the self-disinfection system has been fully developed and assembled. This system replicates the final intended design and is ready for integration into the robot. The system includes:

1. A 4-liter chemical-resistant HDPE tank.
2. A low-power (10-Watt) DC electric pump capable of delivering a flow rate of 0.48 L/min at a head of 20 meters (2 bar).
3. Chemical-resistant PVC piping and secure fittings to ensure leak prevention.
4. Five mist nozzles precisely arranged at shoulder level and one nozzle at the upper back angled towards the robot's head for full-body coverage.

The primary objective of developing this system was to practically evaluate and confirm the feasibility, functionality, and effectiveness of the proposed disinfection solution before it was finally integrated into the robot.

7.4.1 Implementation

The development of the self-disinfection system transitioned from concept to reality through a structured and carefully executed building phase. This section outlines the full process of integrating both the mechanical and electrical subsystems onto the robot, ensuring alignment with the previously validated design.

Component Placement and System Layout

The system layout was designed with careful consideration of space optimization, ease of maintenance, and weight distribution. As shown in the figure below, the 4-liter tank was installed securely at the back inside the robot's frame. This location allowed for efficient routing of pipes toward the pump and ultimately the misting nozzles, while also keeping the system out of the way of other robot subsystems such as mobility or sensing modules.

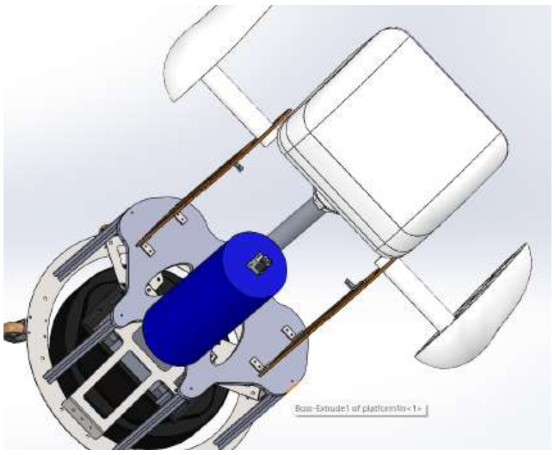


Figure 7.16: Placement of the Reservoir

The back compartment was chosen for its relatively open space, enabling straightforward access for refilling, inspections, and potential maintenance tasks. A bracket system was used to stabilize the tank and absorb vibrations during robot movement.

Hydraulic Subsystem Setup

With the tank in place, a chemical-resistant PVC pipe was connected from the tank's outlet to a low-power 10-Watt DC pump. The pump was positioned nearby inside the same compartment to reduce pipe length and minimize pressure drop.

From the pump, the output pipe was split into five lines using T-joints. These lines were routed along the robot's inner frame to reach the nozzle positions determined during the design phase:

1. Four mist nozzles were mounted at shoulder level, two on each side of the robot. These nozzles were carefully selected to produce a fine mist, ensuring even coverage of disinfectant across the robot's surface. The fine material spray enhances the efficiency of the disinfection process, allowing for better distribution and quicker drying times.
2. One additional nozzle was placed at the upper back, angled toward the head to ensure top-down coverage.

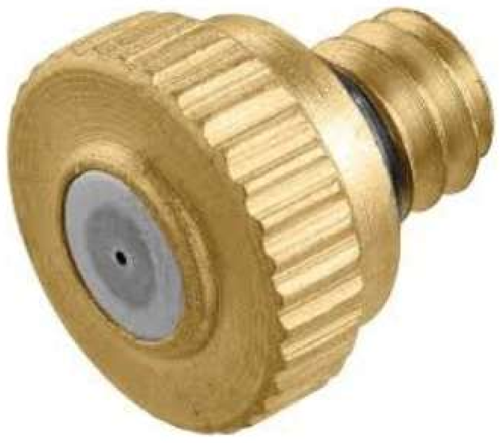


Figure 7.17: Nozzles Selected

All joints were tightly sealed with secure fittings to prevent leaks. The final configuration replicated the simulated design precisely, maximizing disinfectant coverage and minimizing blind spots.

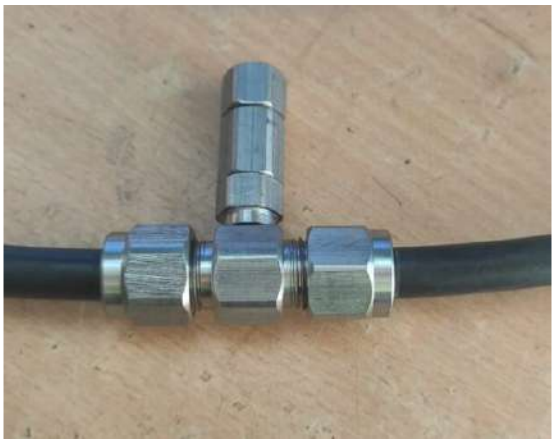


Figure 7.18: Piping and T-Joint Fitting

Electrical Subsystem and Control

The electrical subsystem was assembled to manage the automatic operation of the disinfection process. Key components included:

1. **Water Level Sensor:** Installed inside the tank, the float-based water level sensor continuously monitors the fluid level. It is connected to an Arduino, which alerts the system when the fluid is low and requires a refill.

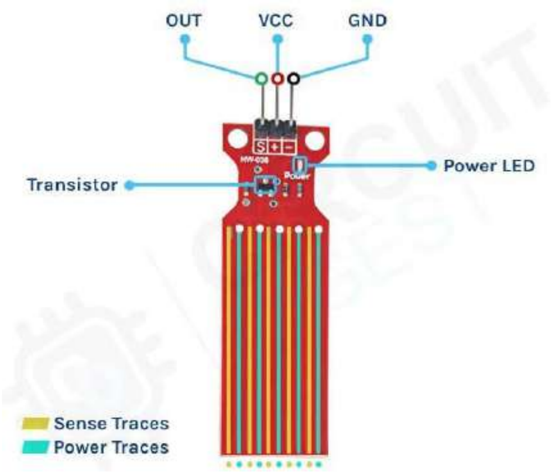


Figure 7.19: Water Level Sensor

2. **Relay Module:** The pump was connected to a relay controlled by the Arduino. This allows the pump to be automatically turned on or off depending on commands from the robot's main controller.

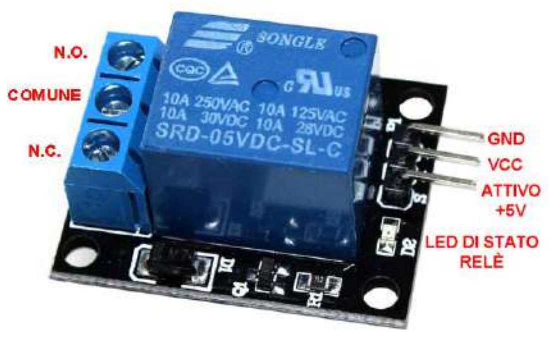


Figure 7.20: Relay Module for Pump Control

3. **Power Supply and Circuit Protection:** The pump and controller components were powered through a stable DC supply, the main 12-V battery.

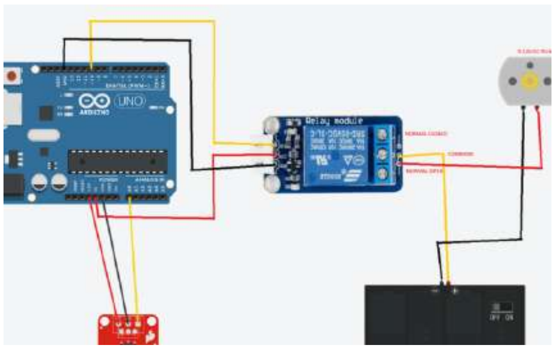
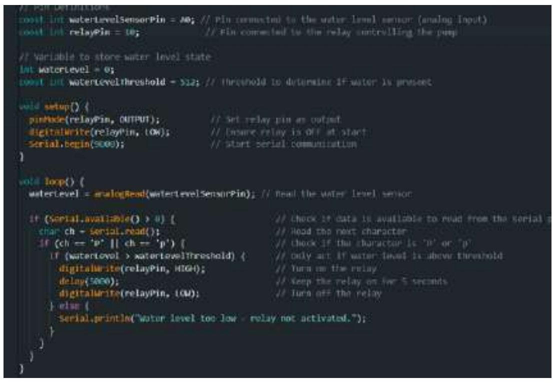


Figure 7.21: Full Wiring Diagram for Self-Disinfection System



```
// Pin Definitions
const int waterLevelSensorPin = A0; // Pin connected to the water level sensor (analog input)
const int relayPin = 10;            // Pin connected to the relay controlling the pump

// Variable to store water level state
int waterLevel = 0;
const int waterLevelThreshold = 512; // Threshold to determine if water is present

void setup() {
  pinMode(relayPin, OUTPUT);          // Set relay pin as output
  digitalWrite(relayPin, LOW);        // Ensure relay is OFF at start
  Serial.begin(9600);                 // Start serial communication
}

void loop() {
  waterLevel = analogRead(waterLevelSensorPin); // Read the water level sensor

  if (Serial.available() > 0) {                 // Check if data is available to read from the serial
    char ch = Serial.read();                    // Read the next character
    if (ch == 'P' || ch == 'p') {               // Check if the character is 'P' or 'p'
      if (waterLevel > waterLevelThreshold) {   // Only act if water level is above threshold
        digitalWrite(relayPin, HIGH);           // Turn on the relay
        delay(5000);                            // Keep the relay on for 5 seconds
        digitalWrite(relayPin, LOW);            // Turn off the relay
      } else {
        Serial.println("Water level too low - relay not activated.");
      }
    }
  }
}
```

Figure 7.22: Arduino Code that was Inverted to ROS Package

7.4.2 Experiment Setup and Testing

The self-disinfection system underwent rigorous testing at multiple stages before and after its integration into the *Sharec* robot. The testing process ensured that the system functioned as expected at every phase of development, with incremental improvements based on the results.

Pre-Implementation Testing (Before Integration into Sharec)

Before implementing the system into the robot, the self-disinfection setup was tested as a standalone unit. During this phase, the mechanical components, such as the pump, tank, and nozzles, were assembled and tested for their physical performance. The focus was on:

- **Fluid flow and pressure:** Ensuring that the pump delivered the required flow rate and pressure to the mist nozzles.

- **Coverage:** Verifying that the mist nozzles distributed disinfectant evenly across a test surface, simulating the robot's body.
- **Leakage:** Inspecting the piping and fittings to ensure there were no leaks under pressure.

This phase was crucial to identify any mechanical issues before moving to the more complex integration stage.

Testing After Initial Integration (Without Electrical Components)

Once the mechanical system passed the initial testing, the next step involved integrating it into *Sharec* without the electrical components. This allowed for:

- **Structural fit:** Verifying that the tank, pump, and nozzles fit correctly within the robot's chassis, especially in the designated back compartment, as shown in the design.
- **Mobility:** Ensuring that the robot could still move freely without any interference from the newly added components.
- **Water distribution:** Testing the nozzles in their fixed positions on the robot to confirm that the mist was still evenly distributed across the robot's surface.

At this stage, the system was manually operated, and mechanical performance was monitored to ensure smooth operation.

Final Testing After Electrical Integration

The final phase involved fully integrating the electrical system, including the water level sensor, Arduino microcontroller, relay, and wiring. Testing focused on:

- **Automation and Control:** Verifying that the Arduino successfully controlled the relay and pump based on input from the water level sensor and disinfection commands.
- **Sensor Accuracy:** Ensuring that the water level sensor triggered a refill alert when the tank reached a low level.
- **Relay Functionality:** Testing the relay's ability to switch the pump on and off reliably without issues.
- **Overall Coordination:** Confirming that all systems worked together, from the sensors and relay to the nozzles, enabling automated disinfection cycles.

This final stage was crucial to confirm that the system could perform autonomously within the robot, ensuring reliable and efficient disinfection in real-world conditions.

7.4.3 Discussion - Final Design Assessment

The final design of the self-disinfection system was assessed following successful implementation and extensive testing. This section evaluates the system's functionality, efficiency, safety, and integration with the *Sharec* robot.

Functionality and Performance

The system performed reliably, maintaining a consistent flow rate of 0.48 L/min (2 bar), ensuring uniform disinfectant coverage. The water level sensor accurately triggered refill alerts, and the relay module seamlessly controlled the pump, enabling autonomous operation.

Efficiency and Coverage

The nozzle arrangement (Design Alternative 3) ensured uniform disinfectant coverage, including difficult-to-reach areas like the head and lower regions. The system showed quick drying times, minimal residue, and efficient disinfectant use, aided by the fine mist. The chemical-resistant pipes and fittings prevented leakage, optimizing system efficiency.

Safety and Reliability

The system passed all safety tests, including pressure and leakage tests, without failures. All internal components were securely housed to prevent damage and chemical exposure, and sharp edges were smoothed for safety.

Integration with the Robot

The system was smoothly integrated into the robot with minimal impact on its mobility and functionality. The compact design ensured that the system did not interfere with other subsystems, and the routing of piping and wiring was optimized for unobstructed movement.

Final Assessment and Conclusion

The self-disinfection system successfully met all design objectives, including efficient disinfectant distribution, automated operation, and seamless integration with the robot. The system demonstrated high reliability, safety, and performance in testing.

7.5 Door Opening Mechanism

7.5.1 Implementation

Hardware Implementation

The design of the door handle was first created using *Fusion 360*, as shown in Figures 7.23 and 7.24. The handle was designed as a lever-type mechanism to replicate realistic door interactions while remaining compatible with the LoCoBot's payload and workspace constraints.

Then, the manufacturing process for the door handle began. The steps were as follows:

1. Additive Manufacturing: The handle was *3D-printed* using PLA material.
2. Post-Processing: The printed handle underwent *sanding* and *grinding* to smooth the rough surfaces.
3. Drilling: Using a *stationary drill press*, holes were created to fix the 3D-printed handle onto the wooden frame.
4. Cutting and Milling: A thicker wooden slab was cut using a *band saw* and refined through *vertical milling* to integrate the spring cavity.
5. Assembly: The handle and frame were joined using screws and *adhesive bonding* with Alteco adhesive.

A proof-of-concept door was then manufactured, measuring 80 cm in height and 56 cm in width to fit within the LoCoBot's operational range. The door leaf, constructed from 5 cm thick foam for lightweight manipulation, was mounted onto the wooden frame using hinges. The door handle was positioned at a height of 40 cm to align precisely with the LoCoBot's home arm position (39.8 cm). The final door and handle setup is shown in Figure 7.26.

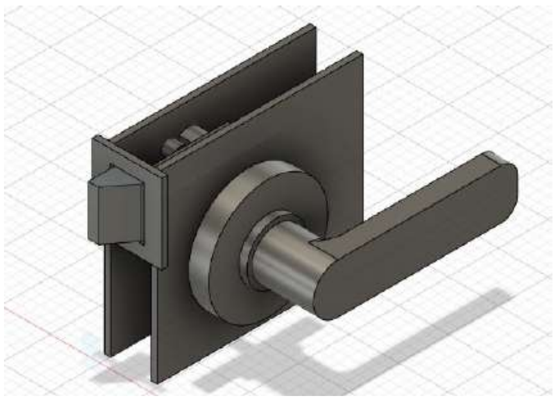


Figure 7.23: Computer aided design (CAD) of the door handle

The physical manufacturing process successfully produced a functional and durable testing setup. The final assembled door handle is shown in Figure 7.25.

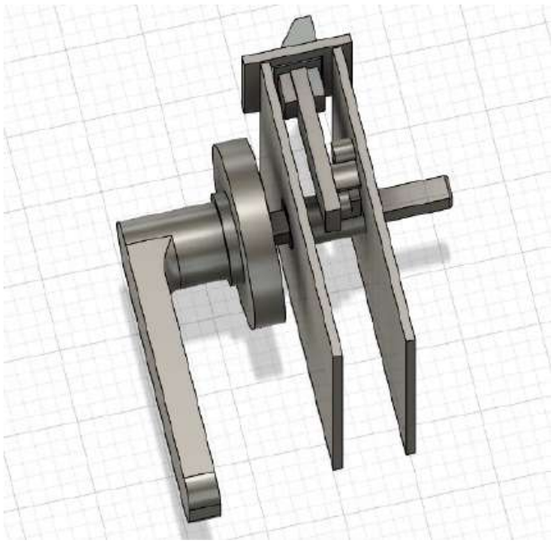


Figure 7.24: Computer aided design (CAD) of the door handle

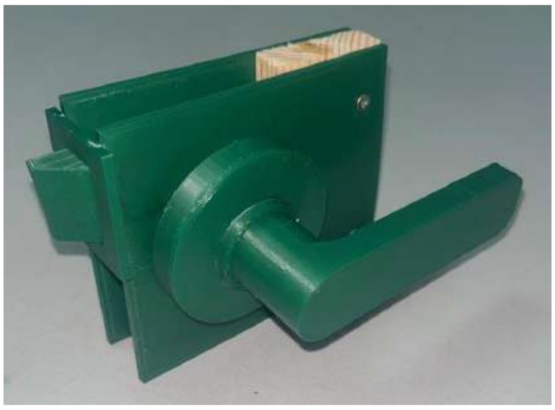


Figure 7.25: Manufactured door handle.

Software Implementation

The software implementation process began with setting up a simulated environment using Ubuntu 16.04 and ROS Kinetic. The Gazebo simulation was configured following tutorials on nodes, topics, and packages to establish the necessary workspace for testing the robot. This step was crucial in validating the feasibility of different door-opening approaches before physical testing.

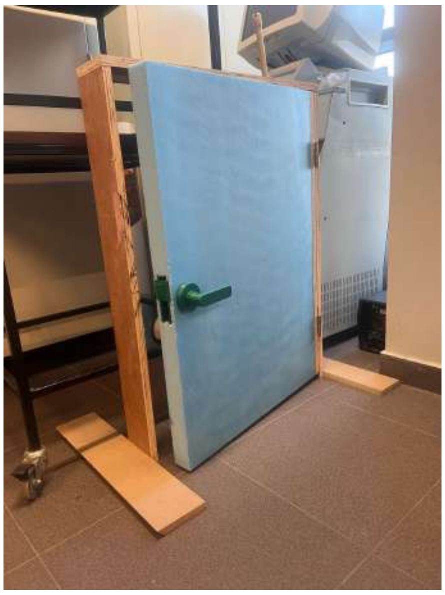


Figure 7.26: Final assembly of door and handle.

After setting up the simulation environment, the Kobuki base was launched in Gazebo to test navigation. The `kobuki_playground.launch` file provided a simulated environment with predefined obstacles to evaluate mobility and obstacle avoidance, as shown in fig. 7.27. A simulated door was modeled in Gazebo and also shown in Figure 7.28. The door handle design was also first developed in Fusion 360 before being incorporated into the simulation.

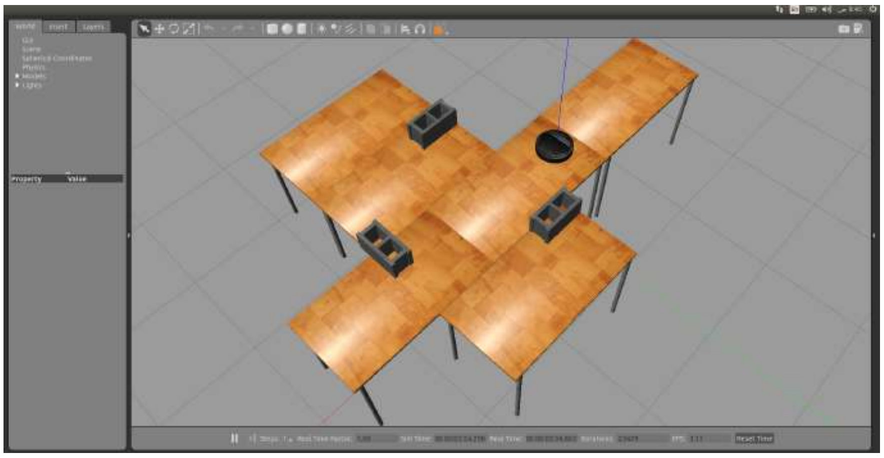


Figure 7.27: Kobuki base launched in a playground environment

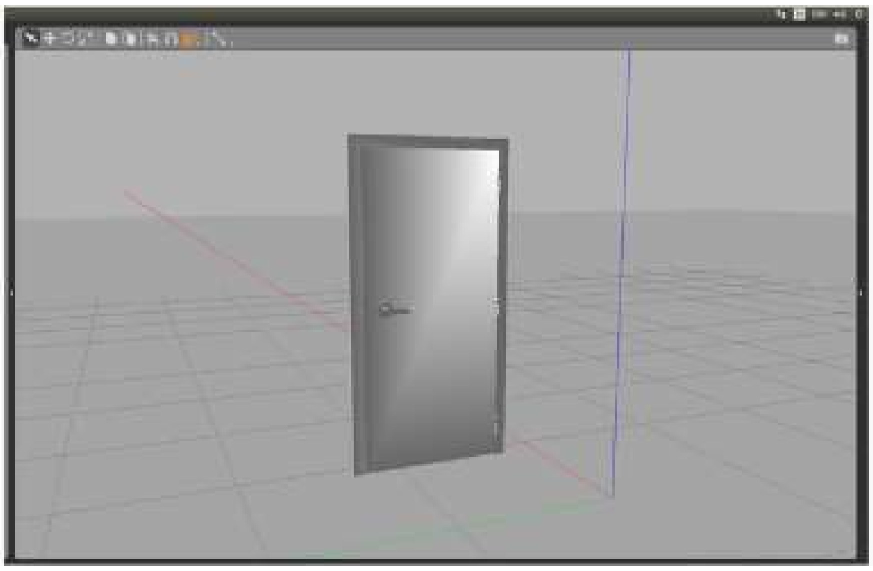


Figure 7.28: Simulated door in Gazebo

Initial attempts to use the built-in grasping model for detecting and manipulating the handle failed, as shown in Figure 7.30, necessitating a different approach. A custom YOLO-based detection system was developed. The process involved capturing images, detecting the handle, cropping the image, passing it to the grasping model for pose prediction, converting the bounding box to 3D coordinates, and executing arm positioning. A dataset of 400 images was collected from varying camera angles and robot positions. The images were labeled using LabelImg to draw bounding boxes around the handles. The YOLOv8 model was trained and validated on a structured dataset to ensure reliable performance. The dataset was divided as follows:

- **Training Set:** 252 images (80% of the dataset)
- **Validation Set:** 64 images (20% of the dataset)
- **Test Set:** 39 images (10% of the dataset)

Upon testing, the model successfully detected the handles in all 39 test images, indicating that the model generalizes well to new, unseen images and is reliable for real-world deployment in handle detection applications. A picture of the results is seen in Figure 7.29.

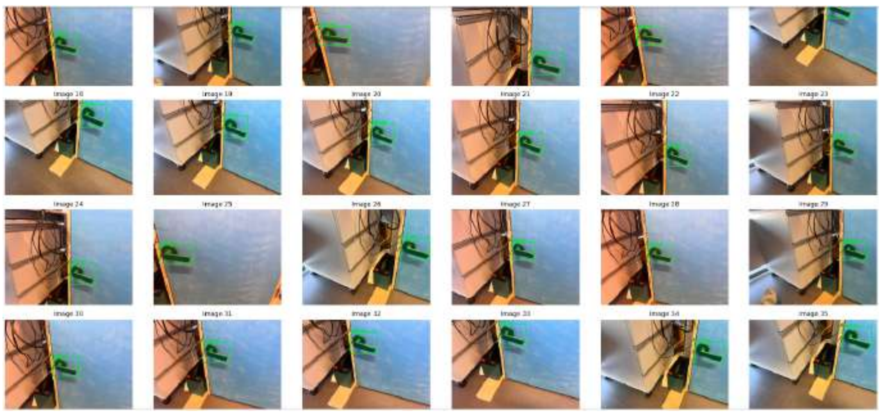


Figure 7.29: YOLOv8 handle detection results.

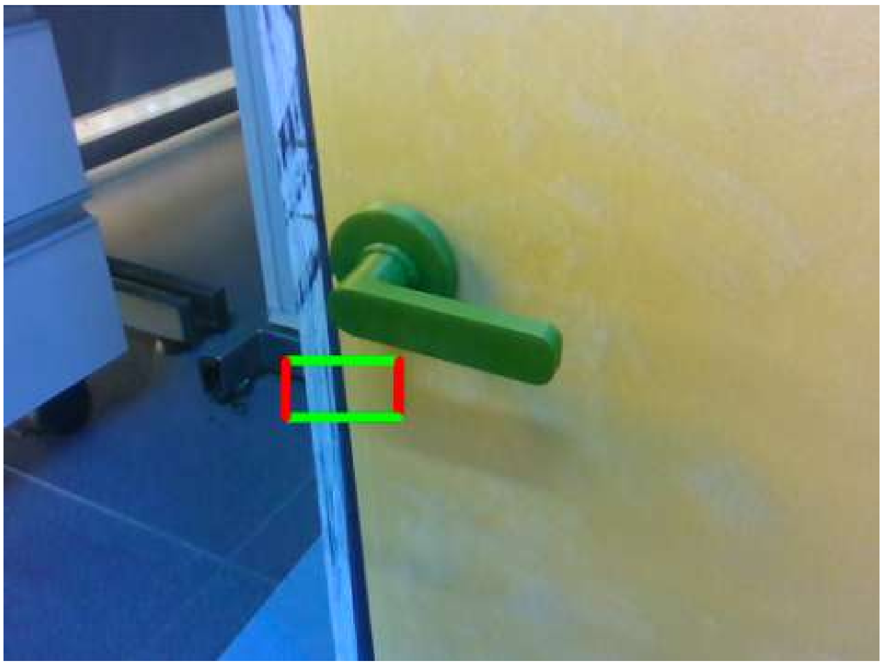


Figure 7.30: Initial grasping model detection failure.

However, due to ROS Kinetic's use of Python 2.7 while YOLOv8 required Python 3.8+, a virtual environment was created to handle data exchange between the two frameworks. During testing, challenges arose when importing YOLO due to OpenSSL version incompatibilities. This led to testing YOLOv3-tiny as an alternative to YOLOv8. Once the model was trained on Google Colab, it was successfully deployed on the LoCoBot to be used by its mounted Intel RealSense camera. The handle detection being implemented on the LoCoBot from the camera is seen in Figure 7.31, with the testing results in Figure 7.32, which show a precision, recall, and F1-Score of 1.00. While this is overly perfect, it is compatible with our application where the model should recognize only one class: this door handle.

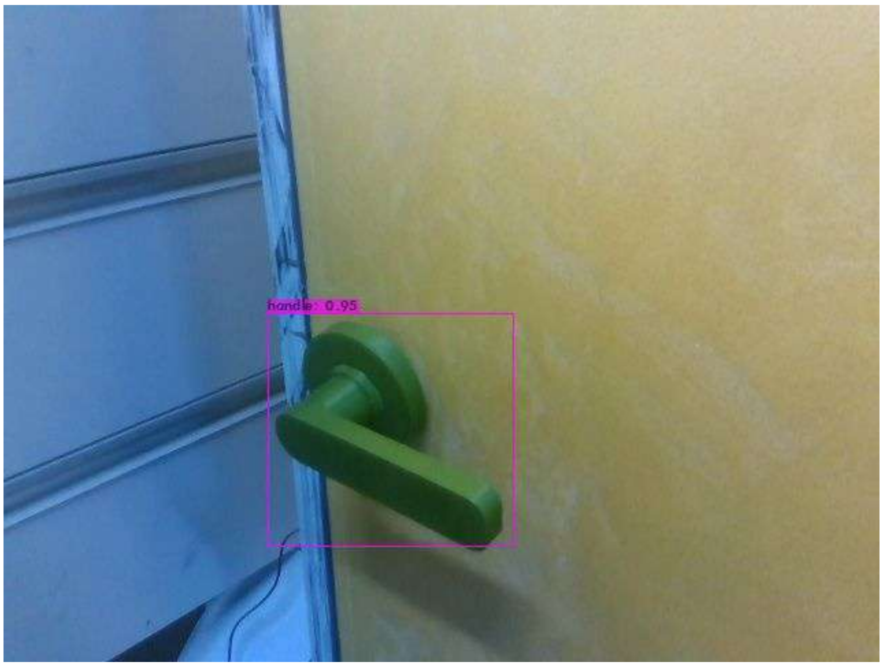
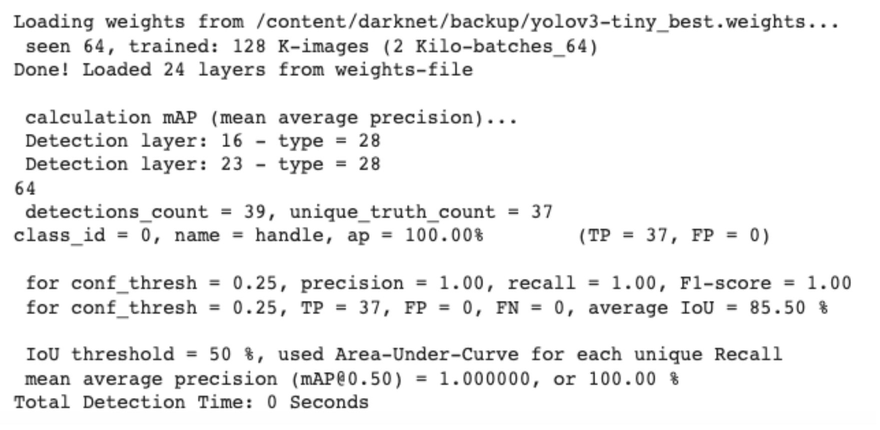


Figure 7.31: Handle detection deployed on the locobot.



```
Loading weights from /content/darknet/backup/yolov3-tiny_best.weights...
 seen 64, trained: 128 K-images (2 Kilo-batches_64)
Done! Loaded 24 layers from weights-file

 calculation mAP (mean average precision)...
 Detection layer: 16 - type = 28
 Detection layer: 23 - type = 28
64
 detections_count = 39, unique_truth_count = 37
class_id = 0, name = handle, ap = 100.00%        (TP = 37, FP = 0)

 for conf_thresh = 0.25, precision = 1.00, recall = 1.00, F1-score = 1.00
 for conf_thresh = 0.25, TP = 37, FP = 0, FN = 0, average IoU = 85.50 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 1.000000, or 100.00 %
Total Detection Time: 0 Seconds
```

Figure 7.32: YOLOv3-tiny testing results.

Once the handle detection task is successful, we started with implementing the functions for the door opening methodology.

The process begins with the `detect_handle()` function, which launches a YOLOv3 model to detect the door handle from the LoCoBot's camera feed. The bounding box coordinates returned from detection are then converted to 3D space using depth data. This is achieved through a series of functions:

`get_bbox_coordinates()`, `process_depth()`, and `get_3D()`, which together estimate the 3D position of the handle relative to the base frame by transforming points from the camera frame.

Once the 3D pose is obtained, the `execute_pose()` function is used to send the end effector to a pre-grasp pose near the handle, employing a custom use of PyRobot's `set_ee_pose_pitch_roll()` method. Fine adjustments are then made using `prep_for_grasp()`, which moves the gripper closer to the handle for grasping. The gripper is closed around the handle using `grasp_handle()`.

Handle manipulation is then initiated by `rotate_handle()`, which rotates the wrist joint to unlatch the door. After rotation, the gripper performs a slight backward displacement via `gripper_linear_disp()` to fully disengage the latch.

The coordinated pulling of the door is handled by a motion planning step. The `pull_door_calcs()` function uses the geometric model proposed in [45] to compute synchronized trajectories for both the robot base and the first arm joint, ensuring the robot pulls the door open while maintaining grasp on the handle.

These dual trajectories are then executed using `run_dual_trajectory()`, which runs base motion and joint motion in parallel using multithreading.

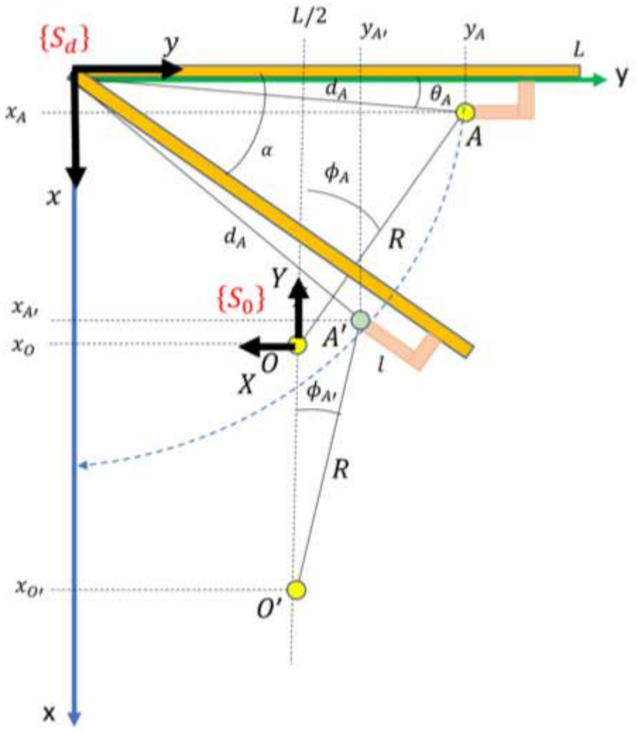


Figure 7.33: Diagram of the door being pulled by the robot for geometry purposes, retrieved from [45].

$$x'_O = \sqrt{R^2 - (\frac{L}{2} - d_A cos(wt + \theta_A))^2 + d_A sin(wt + \theta_A)},$$

Figure 7.34: The equation of the displacement in the x-axis undergone by the base for door pulling, retrieved from [45].

$$\Phi'_A = arctan(\frac{x'_A - x'_O}{y'_A - y'_O}),$$

Figure 7.35: The equation of the displacement of the shoulder joint undergone by the arm for door pulling, retrieved from [45].

Once the door is partly opened, the robot releases the handle using `handle_letgo()` and repositions the arm through `handle_reposition()` to avoid collision with the door. To fully open the door, the robot executes `push_door()`, where the arm is reset to a safe configuration with `set_push_arm()`, and the base navigates to align itself parallel with the half-opened door, otherwise the normal force acting on the door will be 0 which will not open it. Then, the base rotates to push the door open while moving sideways.

Finally, the robot performs door traversal with the `traverse_door()` function, where it retracts its arm and moves forward through the doorway to complete the task.

Throughout the script, multithreading is used to ensure that the arm and base motions occur concurrently, particularly during door pulling and pushing. This design achieves a fully autonomous and efficient door-opening behavior, integrating perception, pose estimation, manipulation, and navigation into a unified pipeline.

The LoCoBot and door states at each stage are shown in Figure 7.36.

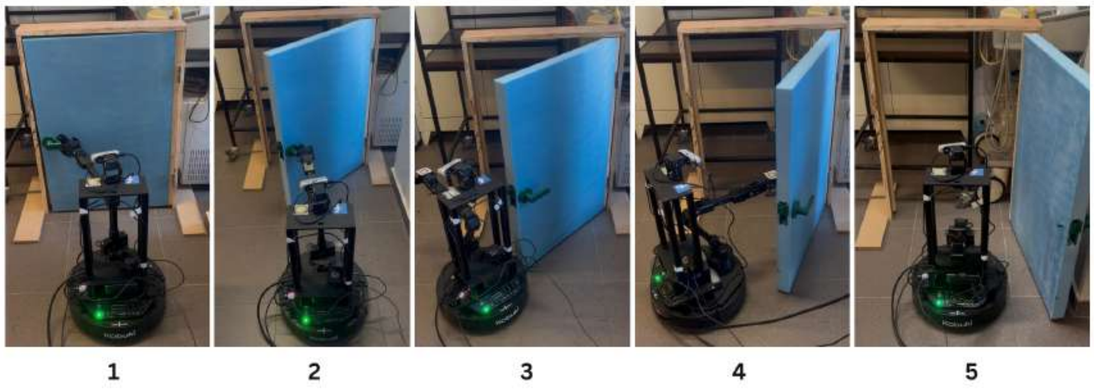


Figure 7.36: The LoCoBot and door states at each stage: (1) after executing the pose, (2) after door pulling, (3) after positioning the base parallel to the partially opened door, (4) after rotating the base to push the door open, and (5) after going to the initial position and while traversing the door.

7.5.2 Experiment Set-up and Testing

The physical testing environment consisted of a manufactured door mounted securely to a wooden frame, and the LoCoBot positioned approximately 0.4 meters away from the door. This distance was determined experimentally, as greater distances led to inverse kinematics (IK) solution failures. The robot was either centered in front of the door or facing the handle directly. Upon detecting the handle with the onboard camera, the robot generated the corresponding 3D pose to align the end effector for grasping.

The following parameters were used to evaluate system performance:

- Handle detection success.
- Grasping success.
- Door unlatching success.
- Successful execution of synchronized base and arm dual trajectories while maintaining grasp.
- Successful repositioning of the base to become parallel to the half-opened door.

- Successful rotation of the base to fully push the door open.
- Successful traversal of the robot through the opened door.

The initial conditions for each trial included the door being fully closed and latched, with the LoCoBot system initialized and the battery sufficiently charged.

The testing process followed a sequence of launching the ROS master node and running each function corresponding to each step sequentially. During testing, several typical failure modes were observed. Misaligned grasp attempts and loss of contact with the handle were the most frequent. To address these issues, rubber pads were added to the gripper fingers, significantly improving grasp stability and reducing slippage during door manipulation.

7.5.3 Discussion - Final Design Assessment

Performance Evaluation

The final system was assessed against the original design objectives. The YOLOv3 detection model performed reliably in identifying the door handle; however, slight offsets were observed, primarily due to the inherent limitations in the LoCoBot's low-cost sensing hardware and limited resolution. Similarly, while integrating depth data enabled the estimation of the 3D position of the handle, minor inaccuracies persisted for the same reasons, slightly affecting the alignment for grasping.

In terms of the door-opening process, the addition of multithreading for coordinated arm and base movement significantly improved the smoothness and overall reliability of the operation. The traversal stage was consistently successful, with the robot reliably passing through the doorway due to sufficient clearance and effective planning of navigation.

Strengths

The final system demonstrated several important strengths. The design remained compatible with healthcare-inspired constraints, emphasizing lightweight construction, safety, and a compact operational footprint suitable for hospital environments. Additionally, the system operated in real-time using low-cost, commercially available hardware components, achieving a fully autonomous door-opening sequence without requiring external intervention or corrections during operation.

Limitations

Despite the successful results, some limitations were identified during testing. The open-loop nature of the motion control strategy made the system vulnerable to slight variations in the door's position, as no feedback correction mechanisms were implemented. Furthermore, the lack of dynamic feedback, such as force or torque sensing, prevented real-time adaptation during manipulation, meaning that small deviations could not be corrected during execution. The success of the full sequence was also heavily dependent on the accuracy of the initial handle detection and pose estimation, making the system sensitive to perception errors.

Future Improvements

Future improvements to the system could address the identified limitations by incorporating adaptive force control strategies, requiring only minimal additional sensing hardware. Improving the generalization of the system to adapt to different door types, configurations, and layouts would also enhance its robustness. Finally, increasing resilience against environmental disturbances, such as unexpected pushing or pulling by patients or staff, would further support real-world deployment of the robot in dynamic healthcare environments.

7.6 Graphical User Interface

7.6.1 Implementation

SHARE-C's App

The first design was completed using Figma, where the color theme of the app was chosen to be Blue. In the field of designing medical equipment, color psychology plays a prominent role as it shapes the user experience and perception. Blue is commonly associated with serenity, security, and trust, key factors needed in a hospital environment to create a sense of comfort, thus improving the overall user experience. Furthermore, according to ISO 9241, clarity and simplicity are required for designing a medical app to let the user navigate smoothly, minimizing confusion and errors. The first preliminary design using Figma is shown below:

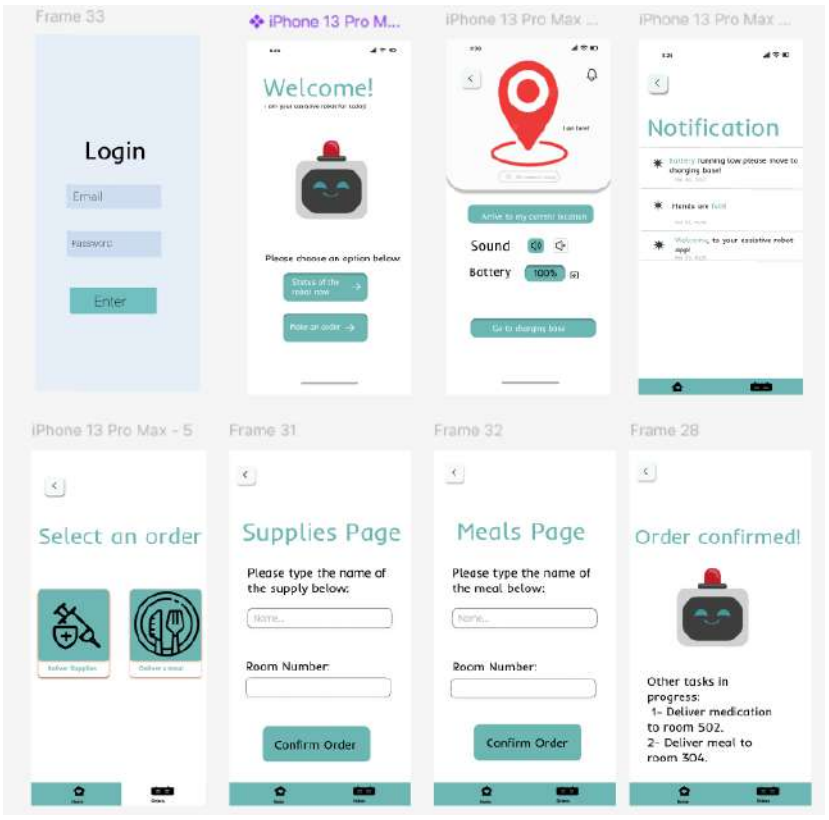


Figure 7.37: UI design using Figma

Then, it is required to implement the client-server architecture: The Flutter app running on the nurse's device collects the input data from the user as they sign in. Then, Flutter constructs a "POST" request that contains this user data and sends it to the NODE.js backend which consequently receives this data and stores it in the MongoDB database.

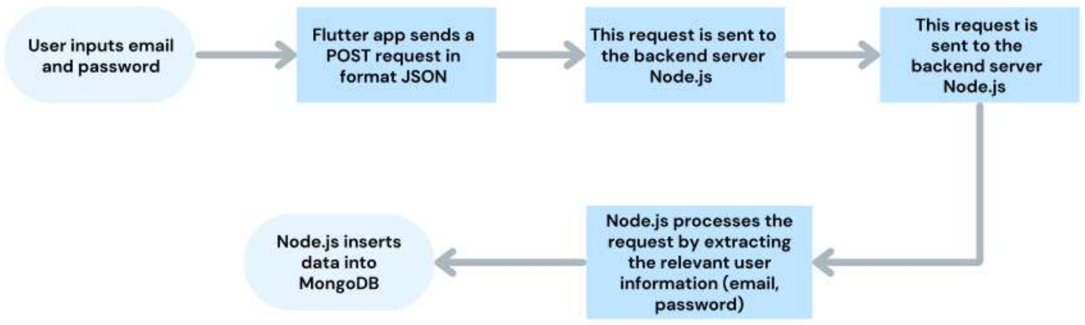
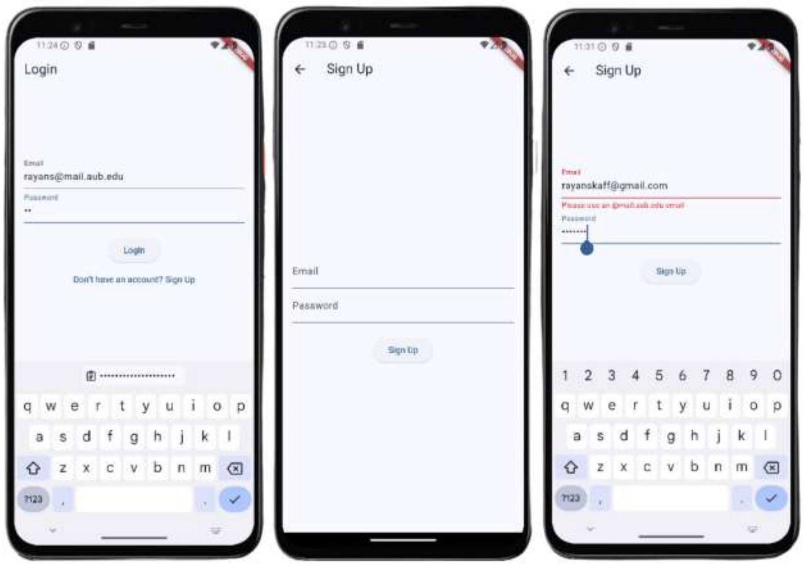

User inputs email and password
Flutter app sends a POST request in format JSON
This request is sent to the backend server Node.js
This request is sent to the backend server Node.js
Node.js processes the request by extracting the relevant user information (email, password)
Node.js inserts data into MongoDB

Figure 7.38: Communication between Frontend and Backend

This way, for every logged-in nurse, data will be recorded and stored in MongoDB in a JSON format. Below is what the MongoDB contains after a nurse logs in:

The first draft of the application was done using Visual Studio Code, then, the design was imported into Android Studio to be able to view the application on an Emulator. An emulator is a software program that allows the programmer to substitute a real device. Below is the programmed login page using an Android Emulator of Pixel 4 and API 35:



rayans@mail.aub.edu

rayanskaff@gmail.com

Please use an @mail.aub.edu email

Figure 7.39: Login and Sign Up Page

For new users not registered in the app, they would have to sign up using their "AUBMC" email only, and any other email will not be accepted. The home page in 7.39 features a blue color to create a calming sensation for the user and provide a comfortable experience. As for the design iteration, the user login and sign up page designs were enhanced to meet user-friendliness and visual appeal:

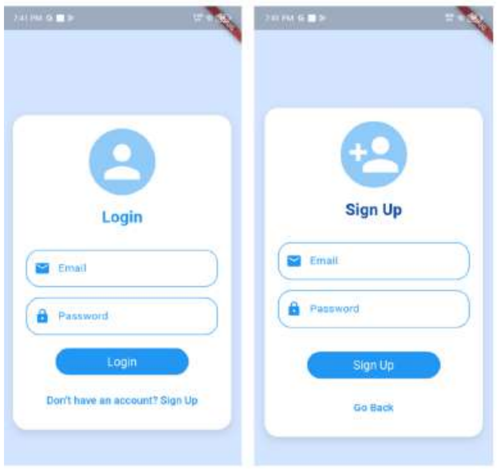


Figure 7.40: Enhanced Login and Sign up pages

Also, a preliminary illustration of the robot is strategically placed on the page to give a sense of friendliness, however, once the robot is ready, this illustration will be replaced with an actual image of the robot. Furthermore, the page features two buttons, each clearly labeled with its function to ease user navigation through the app and avoid any misunderstandings.

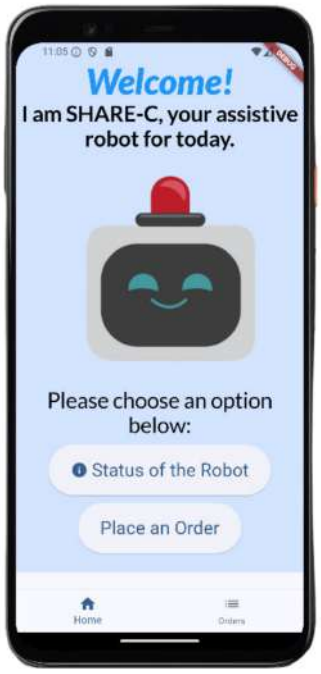

Welcome!
I am SHARE-C, your assistive robot for today.
Please choose an option below:
Status of the Robot
Place an Order
Home
Orders

Figure 7.41: Home Page

Several iterations were done to the "place an order" page seen in 7.42, which resulted in an outcome different than the one designed in Figma, and that is due to new updates to improve the app's user-friendliness and easy use. For example, to make an order, the nurse can use the drop-down menu and choose from a list of orders, without the need for the nurse to type in anything. Then, the user can choose the room number and the priority level of the task: Normal or High. If the user chooses a high priority at the time time of a normal priority task, the robot will automatically reschedule the normal priority task and ensure that the high priority task is done in time. This process is also monitored by the nurse as they should receive a notification alerting them of a change in the schedule and they would have the option to schedule another time. Also, a clock is displayed on the orders page for the nurse to choose the time of the task to be done. After the order is confirmed, a pop-up message will appear to confirm a successful order placement.

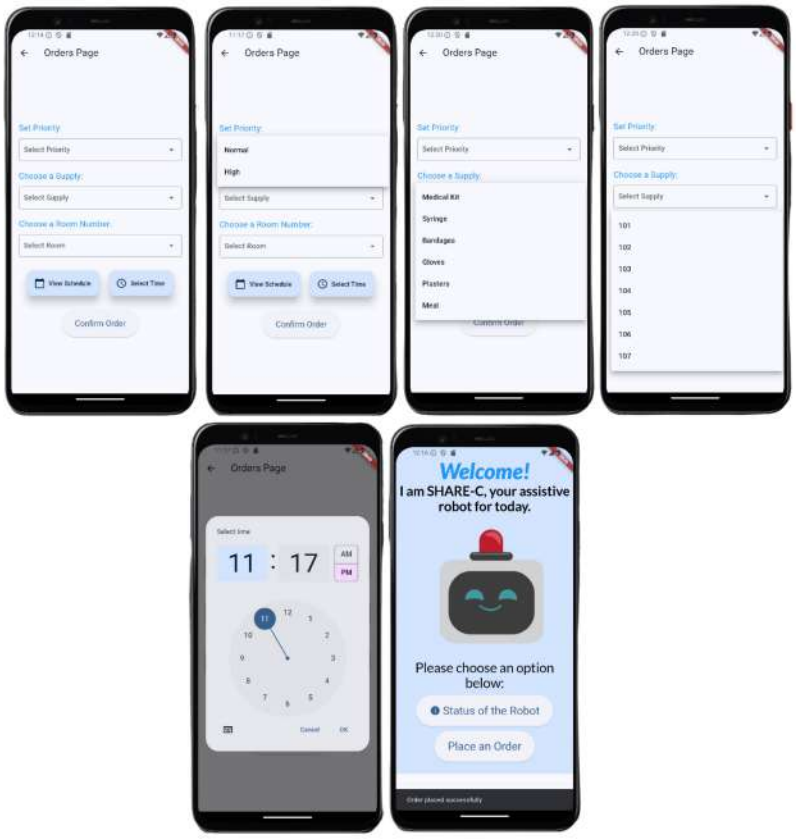


Figure 7.42: Orders Page

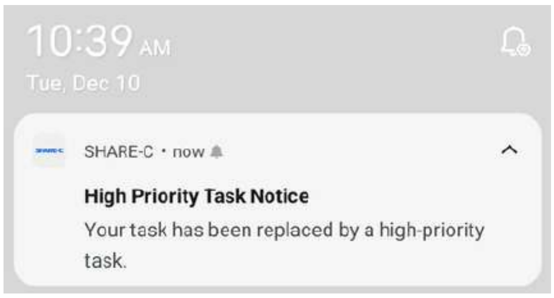
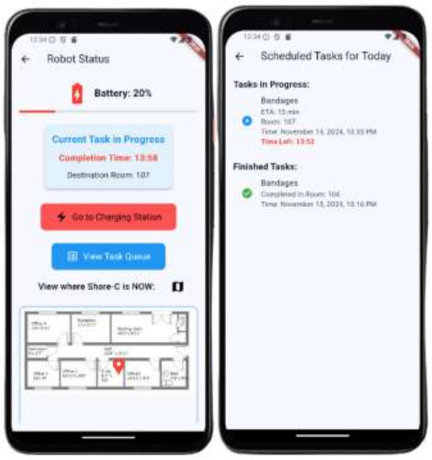

10:39 AM
Tue, Dec 10
SHARE-C • now
High Priority Task Notice
Your task has been replaced by a high-priority task.

Figure 7.43: Notification alerting the user that the assigned "Normal" priority task has been replaced by another high priority task

The robot's status page 7.44 includes visually pleasing and critical components for the robot to operate safely in a hospital. Firstly, a charger bar is displayed for the user to view the current charge of the robot, and it is currently red-colored since the robot's charge is low. Below the robot's battery, there is a button where the user can press it and the robot will automatically navigate to its charging station. In addition, a "View Task Queue" button is placed to allow the user to view the finished and current tasks in progress with the time needed for the task to be completed, as well as cancel the unwanted tasks. The current task in progress is also clearly displayed in a separate widget with the estimated completion time and the destination room. Lastly, a map of the hospital floor is displayed with a red pin to indicate the robot's real-time location. With all of these components present on the status page, the user will be confident that the robot can be constantly monitored at any time during the day. This is also an integral part of the user interface experience: providing comfort and relief.

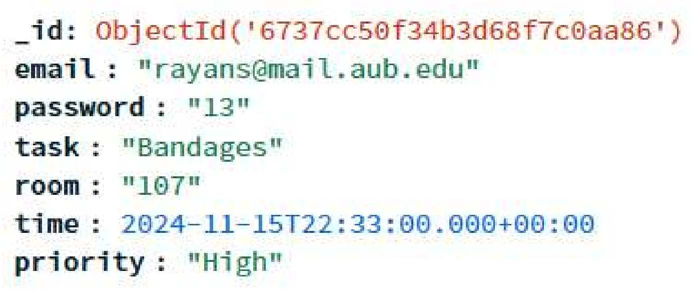

Robot Status
Battery: 20%
Current Task in Progress
Completion Time: 13:58
Destination Room: 107

View where Share-C is NOW:

Scheduled Tasks for Today
Tasks in Progress:
Bandages
ETA: 15 min
Room: 107
Time: November 15, 2024, 10:33 PM
Time Left: 13:52
Finished Tasks:
Bandages
Completed in Room: 104
Time: November 15, 2024, 10:16 PM

Figure 7.44: Status Page

As for MongoDB, the chosen tasks of each logged-in user will be stored in the database system:



```
_id: ObjectId('6737cc50f34b3d68f7c0aa86')
email : "rayans@mail.aub.edu"
password : "13"
task : "Bandages"
room : "107"
time : 2024-11-15T22:33:00.000+00:00
priority : "High"
```

Figure 7.45: Database for a logged-in nurse

Telecommunication System: Video Calling App

Although Agora SDK provides a reliable communication solution, integrating it with a Flutter application is a challenge due to the need to properly configure dependencies and solve version issues. An Agora Developer account is created to be able to get the needed credentials to use in the Flutter app and initialize video streaming and calling: Temporary Token, App ID, and channel name.

SHAREC's Face Expression and Audio

Two expressions were made using Adobe: A smiling face and a talking face, and these animations were implemented as .gif files and added to the video calling application. Regarding the audio generation, due to the limitations of the AI voice generators that accurately reproduce a Lebanese accent in a natural tone, audio generation relied totally on generating Lebanese speech using ChatGPT's calling feature. This process was done by manually writing the prompt to ChatGPT and recording the audio, then enhancing it using Audacity to make it gender-neutral and robotic-like, particularly important in human-robot interaction in Lebanese hospitals.

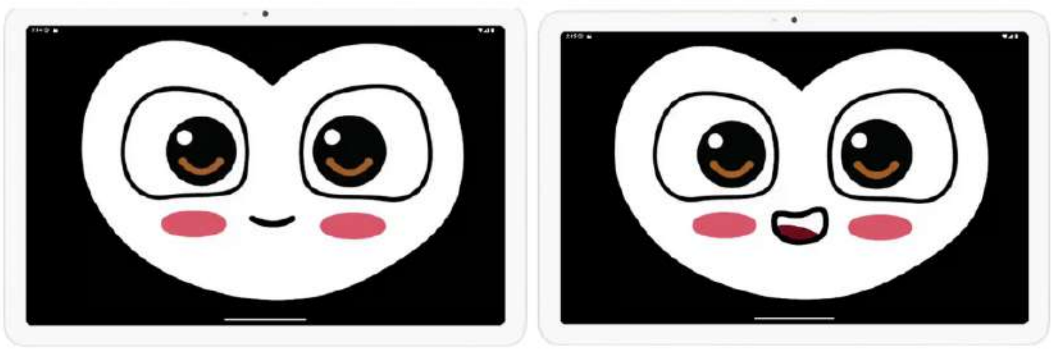


Figure 7.46: SHAREC's Expressions designed using Adobe

7.6.2 Experiment setup and Testing

SHARE-C's App

To ensure the app's functionality beyond the emulator used in Android Studio, the application was tested on a physical Android phone. The process of testing the app using an actual device enabled more accurate and precise testing of the performance, usability, and responsiveness. The Firebase Cloud Messaging notifications system is working successfully and it sends a notification to the user when there is an urgent task reassigning procedure. The application is now completely functional and can be deployed and used on any Android phone.

Figure 7.47: SHARE-C APP on Android Phone

The application was tested on a physical Android device to ensure its functionality beyond the development environment. Moving from the emulator to an actual device allowed for more accurate testing of performance, responsiveness, and real-world usability. Notifications were successfully integrated and verified to work seamlessly, ensuring that users receive updates and alerts as intended. With these features operational, the application is now fully functional and can be deployed on any Android device. However, for the app to operate consistently, the server must be properly configured to remain active at all times, as it handles real-time data and notifications. This requirement highlights the need for a reliable backend setup to ensure uninterrupted communication between the app and the server.

Video Calling App

The complication that surfaced is the need to constantly refresh tokens to ensure uninterrupted access to video calls, as the tokens generated are all temporary. To this end, a method that allows constant token generation is added to Flutter through Agora. Furthermore, the Android device was blocking the camera access due to device policies; however, after additional debugging, these issues were solved, and now the video call feature is working successfully. Once the backend was integrated with Agora, each unique channel name allowed users to join a video call with the specified doctor. Also, to avoid layout distortions, several UI adjustments were made to adapt the video call screen to the different screen sizes, which are the tablet screen on SHAREC's face and the doctor's phone.

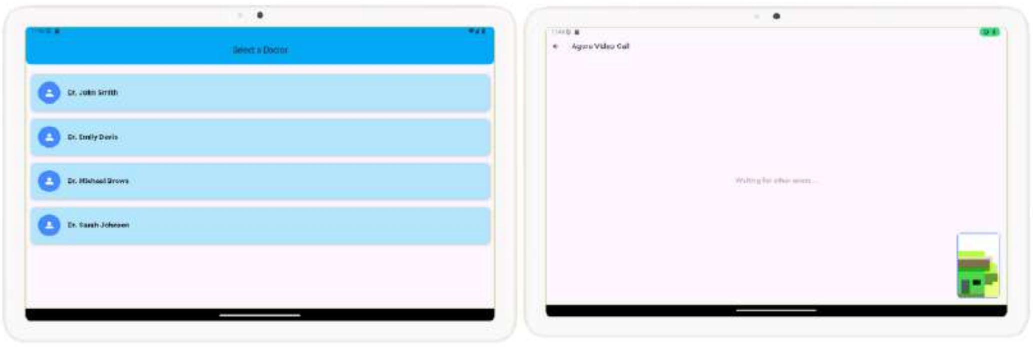
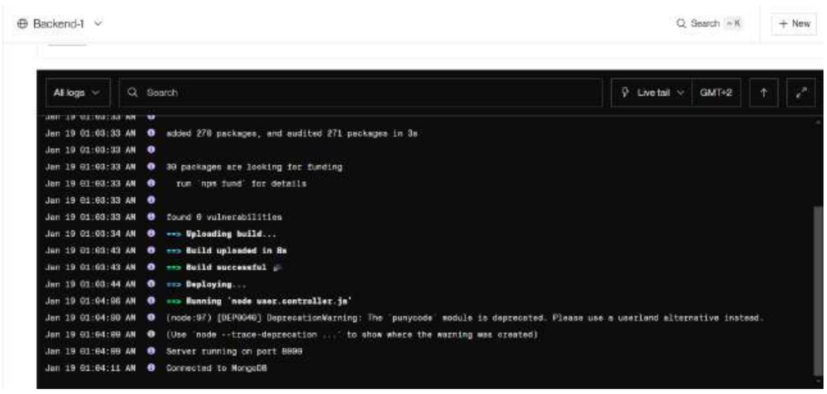


Figure 7.48: Video Calling App

SHARE-C's Backend Deployment on the cloud

After testing the SHARE-C's app, it was evident that it is crucial to enable an independent cloud operation without the constant reliance on running a local server. Consequently, the backend of the nurse application was deployed to the cloud using **Render**. This process facilitates the routes of users login in, signing up, and placing orders, along with a smooth interaction with the frontend Flutter. For hosting the backend, a GitHub repository containing the backend code is linked to a new service created on Render. This allows the automatic deployment of backend updates when committed and pushed to the repository. Then, Render detects the project environment and issues a start command to ensure the start of the backend service: "node user.controller.js". During deployment, multiple issues occurred, such as missing files and dependencies like Express and firebase-service-account.json; however, these issues were solved by implementing the necessary adjustments to the environment variables to align with Render's requirements.

After successfully deploying the backend to Render, a public URL provided by Render is used as an endpoint in the Flutter app. Before updating the front end, Postman was used, and API requests were tested to ensure a successful deployment of the backend. Then, the Flutter app was updated to send the requests to the backend hosted on Render using the URL: "https://backend-3-491z.onrender.com". This greatly facilitates the function of the application as there is no need to update the IP address when running the server each time, instead, everything is deployed to the cloud and remains functional without the dependency on a local machine, thus guaranteeing scalability and optimized accessibility.



```
Jan 19 01:03:33 AM  added 270 packages, and audited 271 packages in 3s
Jan 19 01:03:33 AM
Jan 19 01:03:33 AM  30 packages are looking for funding
Jan 19 01:03:33 AM    run `npm fund` for details
Jan 19 01:03:33 AM
Jan 19 01:03:33 AM  found 0 vulnerabilities
Jan 19 01:03:34 AM  ==> Uploading build...
Jan 19 01:03:43 AM  ==> Build uploaded in 8s
Jan 19 01:03:43 AM  ==> Build successful
Jan 19 01:03:44 AM  ==> Deploying...
Jan 19 01:04:08 AM  ==> Running 'node user.controller.js'
Jan 19 01:04:09 AM  (node:97) [DEP0040] DeprecationWarning: The `punycode` module is deprecated. Please use a userland alternative instead.
Jan 19 01:04:09 AM  (Use `node --trace-deprecation ...` to show where the warning was created)
Jan 19 01:04:09 AM  Server running on port 8080
Jan 19 01:04:11 AM  Connected to MongoDB
```

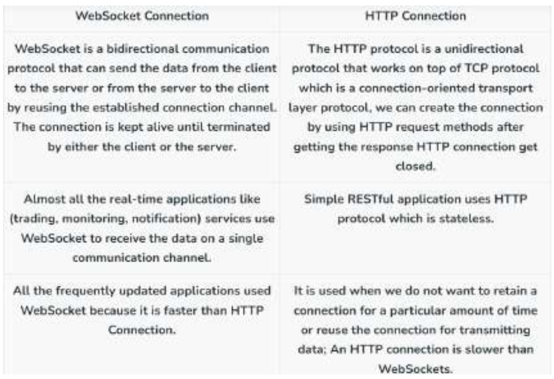
Figure 7.49: Successful Deployment on the cloud

Moving SHARE-C using the SHARE-C's App

To send goals to the ROS navigation system, the process starts with Flutter sending a goal to the backend server, Node.js. Then, the backend publishes this goal to the ROS1 topic responsible for moving the robot. This ensures that SHARE-C follows a navigation path according to the received coordinates and the provided map.

Currently, as a way of communication, Flutter uses HTTP protocol for REST API requests and non continuos data exchange, such as user authentication or placing orders. However, HTTP is unidirectional, and to be able to constantly view the robot's location in real time, a different communication method must be added: WebSocket. Websockets provide bidirectional communication, allowing for continuous data exchange, which is necessary for constantly monitoring and moving SHARE-C. To this end, the plan is to keep HTTP running for user authentication and order placement, along with WebSocket for moving the robot.

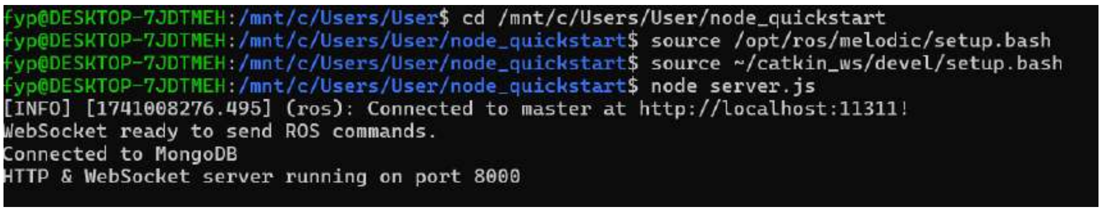
The first step implemented was installing Windows Subsystem for Linux (WSL) to be used as a development environment where ROS would be installed. Instead of using a virtual machine (VM), WSL was chosen due to its easier integration with Windows and the fact that it allows executing the backend code, already on Windows, without the need to transfer the code to a VM. Furthermore, WSL eliminates the need for a specific RAM/CPU allocation like a VM, thus helping in optimizing the system resources. Also, with the nurse app currently using MogoDB and Node.js, WSL allows easier networking since it doesn't require port forwarding when communicating with the Linux-based service needed to move the robot: ROS. Then, to provide users with the real-time location updates of the robot, a specific ROS topic is used to constantly stream SHARE-C's position. Node.js forwards this data to the frontend using WebSockets with a low latency transmission. Finally, the map showing the precise location of SHAREC is updated according to the provided location, thus allowing the continuous tracking of the robot's movement in the hospital.



WebSocket Connection	HTTP Connection
WebSocket is a bidirectional communication protocol that can send the data from the client to the server or from the server to the client by reusing the established connection channel. The connection is kept alive until terminated by either the client or the server.	The HTTP protocol is a unidirectional protocol that works on top of TCP protocol which is a connection-oriented transport layer protocol, we can create the connection by using HTTP request methods after getting the response HTTP connection get closed.
Almost all the real-time applications like (trading, monitoring, notification) services use WebSocket to receive the data on a single communication channel.	Simple RESTful application uses HTTP protocol which is stateless.
All the frequently updated applications used WebSocket because it is faster than HTTP Connection.	It is used when we do not want to retain a connection for a particular amount of time or reuse the connection for transmitting data; An HTTP connection is slower than WebSockets.

Figure 7.50: HTTP vs. WebSocket

Following the installation of WSL on Windows, Ubuntu 18.04 was installed with ROS Melodic to manage the robot movement. Then, rosdep was initialized to correct and resolve the package dependencies needed for ROS. Also, Node.js and MongoDB were installed in WSL for WebSocket communication and the storage of the databases. Lastly, the backend server was launched successfully in WSL along with ROS at localhost: 11311 and WebSocket server on port 8000. After installing ROS, ROS melodic's environment was sourced, and a catkin workspace was made to initialize ROS functionalities.



```
fyp@DESKTOP-7JDTMEH:/mnt/c/Users/User$ cd /mnt/c/Users/User/node_quickstart
fyp@DESKTOP-7JDTMEH:/mnt/c/Users/User/node_quickstart$ source /opt/ros/melodic/setup.bash
fyp@DESKTOP-7JDTMEH:/mnt/c/Users/User/node_quickstart$ source ~/catkin_ws/devel/setup.bash
fyp@DESKTOP-7JDTMEH:/mnt/c/Users/User/node_quickstart$ node server.js
[INFO] [1741008276.495] (ros): Connected to master at http://localhost:11311!
WebSocket ready to send ROS commands.
Connected to MongoDB
HTTP & WebSocket server running on port 8000
```

Figure 7.51: Succesful HTTP with Websocket implementation

After configuring the communication, velocity commands to move the robot were sent to ROS using a rosbridge_server, which is a bridge between Flutter and ROS. Thus, rosbridge_server was installed and ran on port 9090 using the

IOWebSocket channel, a Flutter channel package responsible for handling WebSocket communication. After running rosbridge_server, logs displayed that the server successfully started and is ready to receive velocity commands from the Flutter app using the topic: turtle1/cmd_vel.

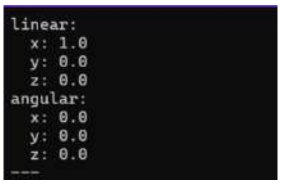
Testing Robot movement Using Turtlesim

To ensure the correct initialization of WebSocket along with ROS, the Turtlesim simulator was used before the deployment of the movement commands on SHARE-C. Using the lightweight ROS simulation allows accurate testing of the velocity commands along with all the networking aspects to verify the Flutter app's capability of sending commands to ROS before implementing this scenario on the real hardware.

1. WebSocket Communication Issues: The main task is to move the turtle by sending commands from the Android phone where SHARE-C's app is installed and running. The initial problem faced was that movement messages were being sent from the Flutter app, but there was no response received from WebSocket. To debug this issue, a ping command was used to test the server from both the PC and the Android device, however, there was no acknowledgment, although rosbridge_server was running successfully.

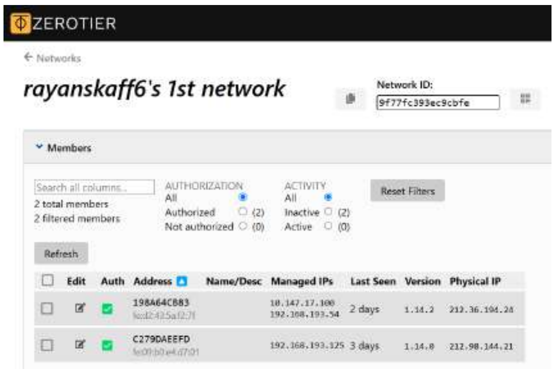
2. Isolating networking issues by running on Localhost: The rosbridge_server was set up solely on the developer's PC, and the Android emulator was used. This eliminated the dependence on an external IP address, and the known IP address for emulators was used: 127.0.0.1. By using the emulator, Turtlesim moved instantly, and WebSocket was able to receive the movement commands. Thus, this confirms the fact that the possible issue stopping the Android device from moving the turtle is related to a network problem.

3. Using a virtual networking tool: After tracing the issue to be related to a limitation in the network, a virtual networking tool, ZeroTier, was implemented to develop a peer-to-peer connection between the Flutter app and the PC. Below is a screenshot from ZeroTier's website showing the successful implementation of a new network, "rayanskaff6's 1st network", along with the two devices connected to the network: the Android phone and the PC.



```
linear:
  x: 1.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
---
```

Figure 7.54: Turtle moving forward: rostopic echo /turtle1/cmd_vel



ZEROTIER

← Networks

rayanskaff6's 1st network

Network ID: 9f77fc393ec9cbfe

Members

Search all columns...
2 total members
2 filtered members

AUTHORIZATION
All ●
Authorized ○ (2)
Not authorized ○ (0)

ACTIVITY
All ●
Inactive ○ (2)
Active ○ (0)

Reset Filters

Refresh

	Edit	Auth	Address	Name/Desc	Managed IPs	Last Seen	Version	Physical IP
☐		✓	198A64C883		10.147.17.100 192.168.193.54	2 days	1.14.2	212.36.194.24
☐		✓	C279DAEEFD		192.168.193.125	3 days	1.14.0	212.98.144.21

Figure 7.52: ZeroTier Networking Tool

After this implementation, the Android phone could smoothly communicate with the rosbridge_server. Consequently, the velocity commands were being sent from SHARE-C's app, and the turtle started moving accordingly.

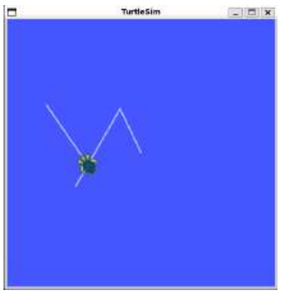


Figure 7.53: Turtle Following Commands Sent from Flutter

From Simulation to Real world: Moving SHARE-C

Adding the option of manual control of SHARE-C through the Flutter application is a critical feature, as it allows the nurses and doctors to control the robot in case of an emergency. To this end, several key steps were implemented to add this feature. Firstly, instead of sending the ROS topics for velocity commands to /turtle1/cmd_vel, which was used initially for simulation purposes. On the Flutter side, each button was mapped to be able to send velocity commands in the format of ROS messages to the topic /teleop, and these messages were then sent through the WebSocket connections to the ROSBridge server. A ROS node subscribed to the teleop topic received these movement messages and converted them into geometry_msgs_Twist velocity messages. As for the connection aspect between the app and the NVIDIA Jetson TX2, the Jetson was connected to the virtual network via ZeroTier, thus we were enable to move the robot successfully. Furthermore, the WebSocket connection allowed the Flutter app to also subscribe to different needed topics for real time monitoring of the robot. For example, the topic /battery_percentage allows the user to view the percentage of the battery charge. Also, the topic /fluid_level displays the real-time value of the amount of disinfectant left in the tank of the robot.

To monitor SHARE-C's real-time location when completing its assigned task, the app subscribes to the topic rtabmap_odom, then the app can continuously receive position and orientation updates, and then the x and y coordinates are extracted, as well as the quaternion angles. To have a complete representation of the robot's heading, a conversion function was used to extract the yaw angle from the quaternion angles, and then these coordinates were passed through a callback function onPositionUpdate and which updated the map display in the app accordingly. This setup successfully allows the continuous tracking of the robot's position and orientation within the specified environment.

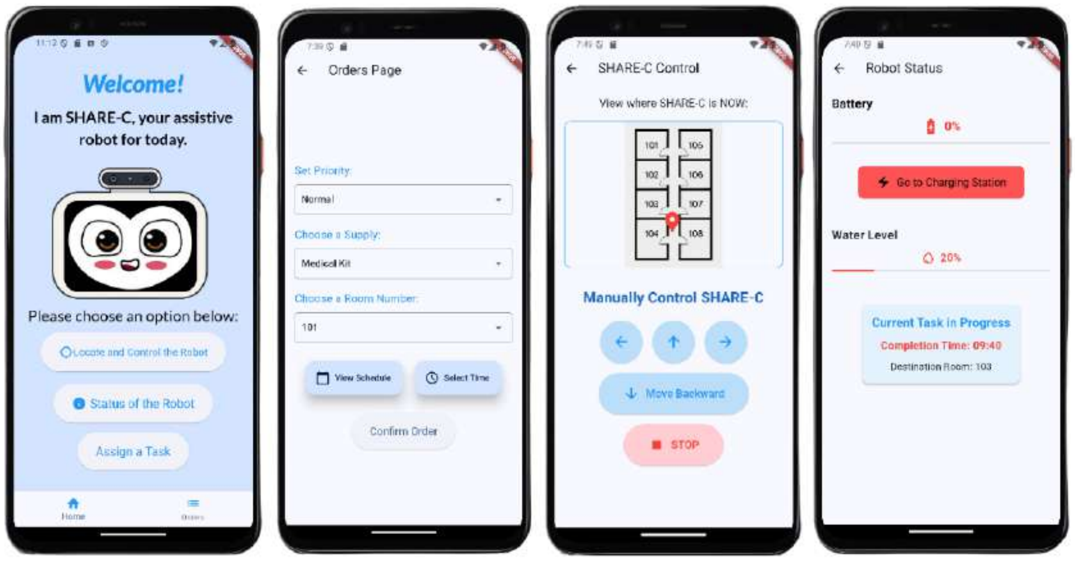


Figure 7.55: Final App Update with Full Functionality

7.6.3 Discussion - Final Design Assessments

The built user interface successfully allowed healthcare workers to continuously monitor the robot in real-time, all while ensuring minimal latency with robust and reliable feedback during operation. Furthermore, the integration of the friendly animated face expressions coupled with a short audio allows patients to feel at ease when interacting with the robot. Also, to decrease the feeling of isolation, a successful and smooth integration of a video calling feature was implemented, allowing patients to communicate with their doctors.

The finished product explicitly satisfies the initially mentioned constraints, both functional and non-functional. Namely, the clarity of the app, the friendliness of the designed faces, and the responsiveness of the integrated system, all contribute to the main goal of designing a user interface: to achieve an intuitive, seamless, and impactful human-robot interaction.

Chapter 8

Conclusion

8.0.1 Reflection and Potential Improvements

This project aimed to enhance the capabilities of SHARE-C, an assistive social healthcare robot, to achieve full autonomy within a hospital environment. Throughout this project, significant progress was made in developing critical components such as autonomous navigation, a Locomotion system, and an efficient door-opening mechanism. While we have not yet tested these components in real-world scenarios, the theoretical frameworks and simulations suggest that these developments could significantly improve the efficiency and safety of medical care environments.

Our focus on the integration of advanced sensor technologies and robust navigation algorithms has laid a strong foundation for the subsequent testing phase. The use of Visual Simultaneous Localization and Mapping (vSLAM) is expected to enhance the robot's ability to navigate complex hospital layouts autonomously.

Looking ahead, the next steps will involve rigorous improvement and testing of the integrated systems in controlled environments to fine-tune the robot's performance before deployment. We anticipate that the completion of this project will provide valuable insights into the practical implementation of robotics in healthcare and lead the way for further innovation in this vital field.

Appendix A

Detailed Project Schedule

The work plan and schedule detail the tasks to be completed, their respective durations, assigned responsibilities, key deadlines, milestones, and deliverables, in close alignment with the Proposed Solution Methodology described in Chapter 5. A Work Breakdown Structure (WBS) and a Gantt chart are developed in this section.

A.1 Work Breakdown Schedule (WBS)

1. **Sensing and Perception**
   - 1.1. System Setup and Configuration
     - 1.1.1. Set up Ubuntu 18.04 and ROS Melodic
     - 1.1.2. Install RTAB-Map SLAM
     - 1.1.3. Install Intel RealSense SDK and ROS Wrapper
     - 1.1.4. Setup Nvidia Jetson TX2
   - 1.2. Testing ORB-SLAM3 on Dataset
     - 1.2.1. Test EuRoC dataset in stereo mode
     - 1.2.2. Validate against ground truth
   - 1.3. RTAB-Map SLAM with RealSense Camera D435i
     - 1.3.1. Test Camera and IMU odometry.
   - 1.4. SLAM Implementation in Simulation
     - 1.4.1. Test the algorithm in Gazebo
     - 1.4.2. Identify inaccuracies in the simulated environment
   - 1.5. Implementation and Optimization
     - 1.5.1. Implement RTAB-Map SLAM on SHARE-C to map the corridor and localize.

1.5.2. Integrate and evaluate performance with path planning

2. **Path Planning and Control**

   2.1. Gazebo Simulations

   2.1.1. Demonstrate perception and autonomous navigation on TurtleBot3

   2.1.2. Use the gained knowledge to simulate on the custom URDF

   2.1.3. Realizing that what was being done in simulation does not mimic what is actually being done. Therefore, gravitating towards RTAB-MAP.

   2.2. Locomotion System Analysis and Design

   2.2.1. Establish a kinematic and dynamic model

   2.2.2. Choose states to be X,Y and ψ for outer loop

   2.2.3. Choose states as v and ω for inner loop

   2.3. Path and Motion Planning

   2.3.1. Implement A* and DWA as the global and local path planners respectively

   2.3.2. Utilize move_base for motion planning

   2.4. Low Level Control

   2.4.1. Implement a low level PID controller

   2.5. Hardware Interface

   2.5.1. Implement all the established algorithms on the hardware

3. **Self-Disinfection System**

   3.1. Analysis and Specification

   3.1.1. Define system constraints and operational specifications

   3.1.2. Identify relevant safety and environmental standards (ISO 14001, ISO 45001, OSHA 1910, REACH)

   3.1.3. Perform preliminary feasibility study and review

   3.2. Component Selection and Acquisition

   3.2.1. Select a suitable disinfectant tank (4L HDPE)

   3.2.2. Select a low-power pump (10 Watts, 0.48 L/min, 2 bar)

   3.2.3. Choose chemical-resistant pipes and fittings

   3.2.4. Select optimal mist nozzles (flow rate 0.096 L/min per nozzle)

   3.3. System Design and Calculations

3.3.1. Conduct detailed calculations for flow rate, pressure head, and pump power
3.3.2. Verify compliance with defined constraints
3.3.3. Document all design calculations clearly

3.4. CAD Design and Simulation

3.4.1. Create three CAD design alternatives for nozzle placement
3.4.2. Set up and run CFD simulations using ANSYS Fluent
3.4.3. Evaluate simulation results and select optimal design

3.5. Prototype Construction

3.5.1. Procure all required components
3.5.2. Assemble system components based on selected CAD design
3.5.3. Ensure secure assembly with leak-resistant fittings

3.6. Experimental Setup and Testing

3.6.1. Build a mock-up structure representing robot dimensions for testing
3.6.2. Establish testing protocols to measure coverage, flow rate, and leakage
3.6.3. Conduct initial tests and collect performance data

3.7. Data Analysis and System Refinement

3.7.1. Analyze testing results for coverage effectiveness and reliability
3.7.2. Refine the prototype design based on test data
3.7.3. Ensure system meets environmental and safety standards

3.8. Final Integration and Implementation

3.8.1. Integrate finalized system into the healthcare robot
3.8.2. Conduct comprehensive operational tests post-integration
3.8.3. Document final validation results and compliance with all standards

4. **Locomotion System**

4.1. Analysis and Problem Definition

4.1.1. Calculate total weight and force distribution
4.1.2. Define operational tolerances
4.1.3. Identify relevant standards

4.2. Force Analysis and Component Selection

4.2.1. Determine variables and parameters like force, torque, spring constants

    - 4.2.2. Research for and select available wheels, motors, and springs
    - 4.2.3. Include a safety factor
  - 4.3. Design and Simulation
    - 4.3.1. Create a detailed CAD model
  - 4.4. Prototype Development
    - 4.4.1. Assemble the selected components
    - 4.4.2. Verify the mechanical integrity of the system
  - 4.5. Experimental Testing and Validation
    - 4.5.1. Test the prototype on surfaces with ± 10mm irregularities
    - 4.5.2. Test stability during movement and under loads
    - 4.5.3. Compare test results against ISO 13482 and ISO 22878
  - 4.6. Iterative Refinement
    - 4.6.1. Adjust spring placement and stiffness
    - 4.6.2. Redesign components more efficiently
  - 4.7. Final Design
    - 4.7.1. Achieve balanced cost, durability, and performance
    - 4.7.2. Include precise tolerances in CAD drawings according to ASME Y14.5 standards

5. **Door Opening Mechanism**
  - 5.1. LoCoBot Simulations
    - 5.1.1. Install the PyRobot software
    - 5.1.2. Import a door model with physical characteristics
    - 5.1.3. Implement path planning and control algorithms
  - 5.2. LoCoBot Physical Testing
    - 5.2.1. Test the physical robot's camera and laser sensors
    - 5.2.2. Run example navigation tools with velocity and position control
    - 5.2.3. Run example manipulation tools with joint and end-effector control
    - 5.2.4. Run additional examples from PyRobot's tutorial
  - 5.3. Door Opening Approach
    - 5.3.1. Assess between path planning and control algorithms
    - 5.3.2. Develop an optimal step-by-step approach based on simulations performance
  - 5.4. Experimental Validation

5.4.1. Manufacture a small door with a handle
5.4.2. Perform experiments on the LoCoBot
5.4.3. Obtain acceptable performance identified through simulations

6. **Graphical User Interface**

6.1. Build the Frontend
6.1.1. Create preliminary designs on Figma
6.1.2. Use Flutter for easy accessibility

6.2. Construct the Backend Logic
6.2.1. Establish user authentication, robot status, and task scheduling features using Node.js as a framework
6.2.2. Create and manage a database system using MongoDB

6.3. Set up robot's facial expressions

6.4. Set up robot's audio expressions

A.2 Gantt Chart

The Gantt chart in Figure A.1 is developed according to the Work Breakdown Schedule above and generated using Python.

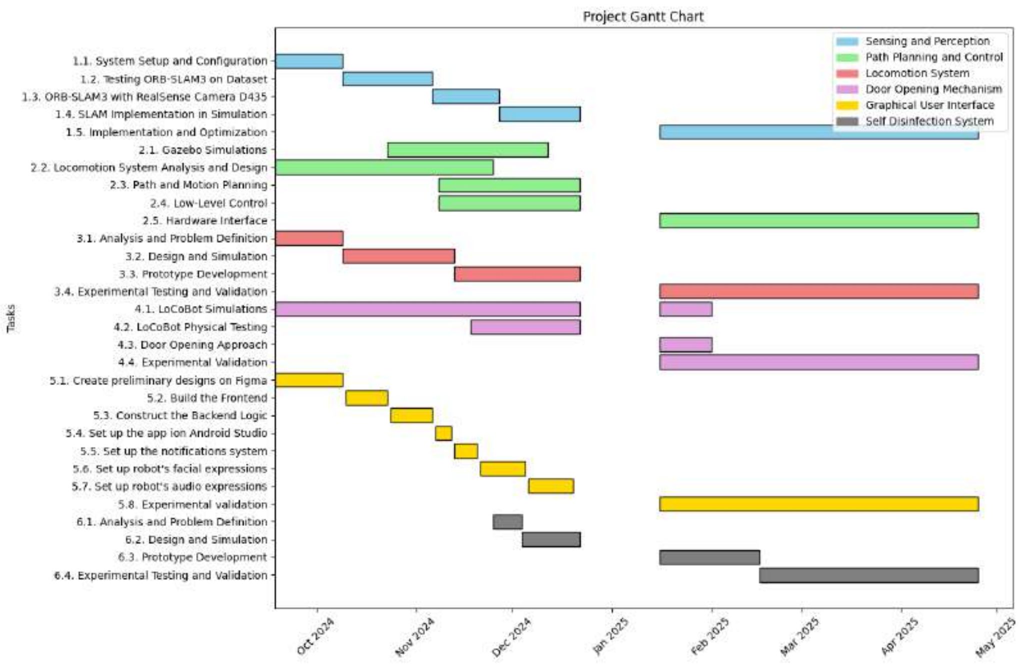


Figure A.1: Project Gantt Chart

Appendix B

Budget

Item Name	**Description**	**Vendor**	**Price (USD)**
Caster Wheels	Swivel caster wheels with 360° rotation	From the previous Prototype	Free
Driving Wheels	Rubber wheels with 10 cm diameter	Previous FYP: Glass Façade Cleaning Robot	Free
Springs (Caster)	Compression springs, $k = 6900\,\mathrm{N/m}$	O.G.K	10
Springs (Driving)	Compression springs, $k = 7650\,\mathrm{N/m}$	O.G.K	10
Electric Motors	18V DC motors, 257.85 W power output, 78 RPM	Previous FYP: Glass Façade Cleaning Robot	Free
Circular Base	Aluminum circular plate	SRB	Free
Miscellaneous Fasteners	Screws, nuts, and bolts for assembly	SRB	Free
Intel RealSense D435i	Camera	VRL lab	Free
LoCoBot with a Kobuki base and a WidowX-200 arm	Robot	VRL lab	Free
4L Tank	The tank is a 4-liter container, ensuring durability and secure storage of disinfectant.	repurposed from a car	Free
DC Pump	The pump is a low-power 10-Watt DC electric pump, delivering a flow rate of 0.48 L/min at a pressure of 2 bar.	Ubuy Lebanon	19

Item Name	Description	Vendor	Price (USD)
1200W DC to DC boost step up convertor 8-60V to 12-83V	A high-efficiency converter used to step up voltage from 8-60V to a stable 12-83V output for powering various components.	ELECTROSLAB	12
Voltage Sensor	A sensor that measures the voltage levels in the system to ensure proper operation and prevent over-voltage conditions.	ElectrosLab	1
BTS7960 Motor Driver	A powerful motor driver capable of controlling high-current motors, providing smooth and efficient operation for robotics applications.	Robot Pi Shop	7
Lead Acid Battery12V-36Ah	A reliable rechargeable battery providing 12V power with a 36Ah capacity, ideal for providing energy storage and consistent power for robotic systems.	Al Raed Contracting	69
Water Level Sensor	A sensor that monitors the water level in the tank, triggering alerts when the fluid reaches a low point to prevent dry running of the pump.	Robotics Club	Free
Tablet as Robot Face	The robot features a tablet as its face, which displays relevant data and interacts with users for various functions.	VRL	Free

Item Name	**Description**	**Vendor**	**Price (USD)**
Fine Mist Nozzles	These are fine mist nozzles, imported from China through Amazon, designed for efficient and even distribution of disinfectant across the robot's surface.	Amazon	41
Nvidea Jetson tx2	A powerful embedded computing platform designed for AI and machine learning applications, providing the computational power needed for the robot's autonomous operations and sensor integration.	VRL	Free
Arduino Mega	A microcontroller board	Robotics Club	Free
Sum	-	-	168

Table B.1: Budget Table

Appendix C

Technical and Engineering Drawings

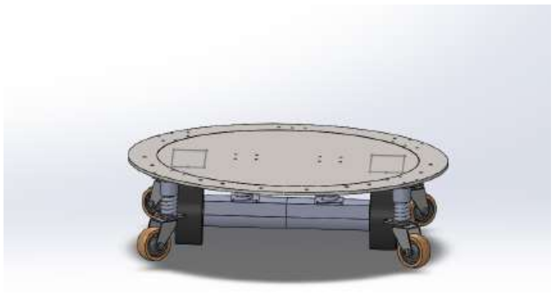


Figure C.1: Locomotion System Mechanical Design via Solid Works

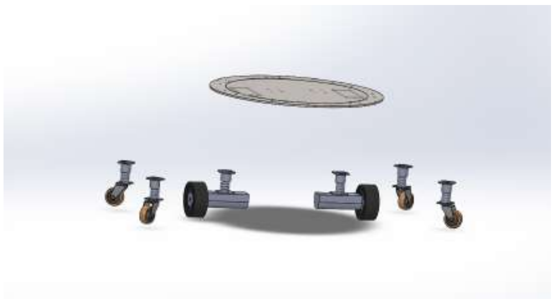


Figure C.2: Locomotion System Mechanical Design Exploded View

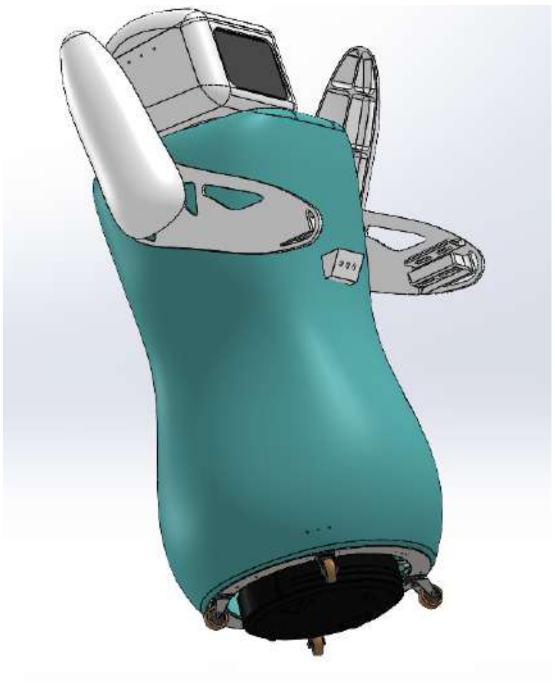


Figure C.3: Full Robot CAD Design

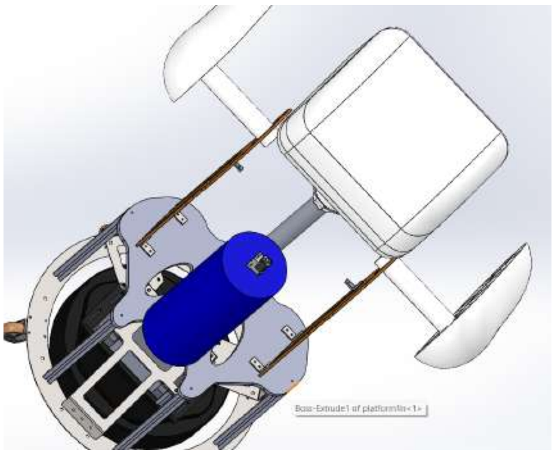


Figure C.4: Placement of the Tank in the Robot

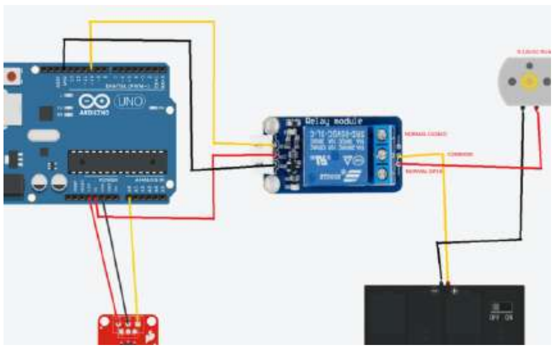


Figure C.5: Full Wiring Diagram for Self-Disinfection System

Appendix D

LIST OF RESOURCES AND ENGINEERING TOOLS NEEDED

- ROS1 Kinetic *(New)*
- ROS1 Noetic
- ROS2 Humble
- ROS1 Melodic
- Arduino IDE
- Gazebo *(New)*
- RViz, RViz2
- MATLAB
- Simulink
- Flutter *(New)*
- Visual Studio Code
- Android Studio *(New)*
- MongoDB *(New)*
- Figma *(New)*
- Solid Works 2024
- Ansys 2024

- Fusion 360 *(New)*

Appendix E

UPDATED PROJECT DESCRIPTION AND AGREEMENT FORM

Project Description and Agreement Form (PDAF)

Final Year Project (MECH 501/502), Department of Mechanical Engineering
MSFEA, American University of Beirut
2024-2025

Design Project Title Include a descriptive title, this is not necessarily the final one that will be adopted by the team.	Developing Robot SHARE-C's Full Autonomy
Name of Sponsor Include only if there is industry support or funding for the design project	
Faculty Advisor	Name: Naseem Daher Dept. EECE / MECH
Co-Advisor (if any)	Name: Dept.
Design Project Type Experimental, analytical, both, or other	The project entails both aspects: analytical and experimental.
Design Project Description Briefly specify the desired need(s) the final product is expected to meet	In a previous research project, a <u>S</u>ocial <u>H</u>ealthcare <u>A</u>ssistive <u>R</u>obot with a Lebanese <u>E</u>thno-<u>C</u>ultural Identity (SHARE-C) was designed by following the human-centered design (HCD) process. The main features of the robot were implemented (namely its mechanical components: body, arms, head, face, and locomotion), however, the robot lacked autonomy and the manufactured proof-of-concept (POC) prototype did not include its novel self-disinfection function. In this project, the students will work on developing SHARE-C's full autonomy in order to meet its intended performance specifications. In particular, the following needs are required to be met by the final product: • Autonomous navigation in a hospital environment with a typical floor plan that includes multiple rooms and a nurse station. The following tasks have to be completed to enable autonomous navigation: o <u>Sensing & Perception (Member #1)</u>: After the selection, procurement, and installation of suitable sensors (camera, ultrasonic, IMU, etc.), the team needs to design a Visual Simultaneous Localization and Mapping (vSLAM) algorithm to build a map of the environment while simultaneously localizing the robot within the generated map, based on visual feedback captured by the onboard camera. o <u>Planning & Control (Member #2)</u>: Implementation of path planning (waypoints), trajectory generation (with suitable smoothing techniques), and motion planning (desired translational and rotational velocities) algorithms, while considering dynamic constraints and obstacles (static and dynamic). Low-level controllers are to be designed to achieve the robot's actual motion based on the motion planning system's commands (steering, acceleration, braking).

	• <u>Self-Disinfection System (Member #3)</u>: Review the existing mechanism sizing and design, propose needed upgrades or changes, properly size the selected components (pump flow rate, tank size, hoses, gears, etc.), procure and assemble the components, test and validate the mechanism design before integration in the robot's cavity. • <u>Locomotion System (Member #3)</u>: The previous locomotion system was based on the LoCoBot, which has a low payload and thus limited traction abilities. The group needs to design a new wheeled locomotion system (electric motors + wheels) that enables differential drive and has adequate traction force. • <u>Door Opening Mechanism (Member #4)</u>: The robot must be able to open the doors of typical hospital rooms at AUBMC in order to enter and exit the patients' rooms. Special attention should be paid to the following features: ○ **Mechanical Manipulation**: Investigate the type of end effector needed to open doors at AUBMC (gripper, manipulator, etc.), make a selection, design/manufacture or procure the mechanism, and integrate it to the robot. Of course, this mechanism depends on the type(s) of door(s) to be opened (push/pull, swing, sliding) – but the scope of this FYP is limited to one type only. ○ **Sensors**: The following sensors (camera, radar, LIDAR, depth, ultrasonic) are needed for: a) detecting doors and aligning the robot with them, b) providing force feedback on the resistance level to adjust the robot's grip and traction. ○ **Path Planning and Motion Control**: Implement algorithms to determine the optimal movements to operate the doors. Furthermore, a grasping algorithm is to be implemented to ensure secure and effective handling of the door mechanism. ○ **Closed-loop Control**: Design a controller to adjust the robot actions based on real-time sensor feedback. • <u>Graphical User Interface (Member #5)</u>: Design of a GUI for the medical staff to interact with and command the robot to execute various item delivery missions from various start/end points and return to the home station or charging station. The team should integrate UI/UX principles, database management for scheduling multiple tasks/missions, and visual feedback on the robot's live location, status, etc. Furthermore, the team member needs to investigate the existing communication features (facial expressions and audio phrases) to improve upon them and increase the robot's acceptance by patients.
Expected Deliverables Required deliverable(s) from the team at the conclusion of the design project	The team is required to deliver a fully-functional robot that can autonomously navigate a hospital floor level based on commands received by the medical staff. The robot should follow the following sequence: 1) Start from any location (including the base/home station or any room) and move to a target location (room), assuming that the item(s) is/are placed on the robot's tray by the medical staff.

	2) Initiate the self-disinfection system until sterilization is achieved (coverage and timing are TBD). 3) Open the door and enter the room. 4) Approach and greet the patient according to a preset protocol. 5) Stop near the patient's bed and wait until the item is picked up. 6) Say goodbye to the patient and wish them a good day. 7) Turn around and stop near the door. 8) Initiate the self-disinfection system until sterilization is achieved (coverage and timing are TBD). 9) Open the door and exit the room. 10) Go to the next location as commanded by the medical staff, or go to the charging station if the battery level is too low.
Technical Constraints A list of multiple realistic constraints; examples: manufacturability, sustainability, power, accuracy, real-time operation etc. The **technical** constraints included should be detailed and specific to the design project not generic. Example: at least 50%, by mass, of the product designed must be derived from recycled material.	• **Environmental Adaptability**: The robot must be able to navigate obstacles up to 10 cm in height and fit through doors with a minimum width of 80 cm. • **Precision and Accuracy**: Task execution accuracy should be within ±3 mm. • **Autonomous Navigation**: vSLAM accuracy should be within 5 cm of the actual location; obstacle detection should be within 1 meter. • **Battery Duration**: Minimum of 8 hours of operational time on a single charge. As for power management, the target recharge time should be under 2 hours. • **Real-Time Processing**: Data processing latency should be less than 100 milliseconds. • **Error Handling**: System should recover from faults within 10 seconds or less. • **Patient Safety**: Ensure all surfaces are smooth and free of sharp edges; follow ISO 13485 for medical device quality management.
Non-Technical Constraints List multiple realistic non-technical constraints; ex: cost, environmental friendliness, social acceptance, political, ethical, health and safety, etc. The **non-technical** constraints included should be detailed and specific to the project not generic. For example: the bill of materials required to build the product designed in this project must cost less than $300 when purchased over the counter in Beirut, at the time of project completion.	• **Ethical Considerations**: The robot must ensure ethical use and protect the privacy of patients and not record their personal data. • **Cost and Budget**: The team must stay within the budget constraints imposed by the departments for purchasing equipment. • **Acceptance and Usability**: The final design must ensure the ease of use and acceptance by healthcare staff and patients. • **Cultural and Social Impact**: The team should design the robot with cultural sensitivity and consider social acceptance. • **Patient and Staff Safety**: The team should assess and address potential health risks posed by the robot and develop emergency response protocols.
• **Contemporary Issues** • Cite recent article(s) pertaining to the project or project area: news articles, blog discussions, academic articles, conference topics, etc.	• https://vinbrain.net/medical-robots-and-the-latest-14-applications-in-hospitals • https://www.intel.com/content/www/us/en/healthcare-it/robotics-in-healthcare.html • https://www.automate.org/robotics/service-robots/service-robots-medical • https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10287569/
Possible Applicable Standards Number and identify all applicable standards	• ISO 13485: Medical devices – Quality management systems • ISO 9241: Ergonomics of human-computer interaction • ISO/IEC 62366: Usability engineering for medical devices

	• ISO 13482: Robots and robotic devices – Safety requirements for personal care robots • ISO 14971: Medical devices – Application of risk management • ISO 60601: Medical electrical equipment – Safety and performance • ISO/IEC 27001: Information security management systems • ISO 9283: Performance of industrial robots – Test methods	
Resources and Engineering Tools Number and identify resources and engineering tools needed and whether they are available or need to be acquired; ex: software licenses, instruments, facilities, components, etc.	• Computer-Aided Engineering (CAE) Tools such as Computer-Aided Design (CAD) software (*e.g.*, SolidWorks). • MATLAB/Simulink® and/or LabVIEW • Robot Operating System (ROS), RViz, Gazebo • Embedded systems (*e.g.*, Arduino, Raspberry Pi) • Sensors (distance, force, pressure, etc.) and actuators (electric motors, pump, etc.) • Lab space: IOEC 4th floor as a sample floorplan for initial testing and validation.	
List of Disciplines At least **three** specialties within or outside your department must be identified (by circling or by writing in the place provided).	Interdisciplinary (within ME) - [✓ Mechanisms/machine design] - HVAC - Applied energy - [✓ CAD] - [✓ CAE (simulations)] - Materials - Manufacturing - [✓ Robotics] - [✓ Controls] - Other (Identify) ________	[✓ Multidisciplinary (External to ME)] - [✓ Electrical and Computer Eng.] ________ - Civil and Environmental Eng. ________ - Engineering Management ________ - Architecture / Graphic Design ________ - Chemical Engineering ________ - Other (Identify) ________

This form describes the final year project as initially envisioned by the team. The signatures below indicate the willingness of each individual to work in good faith in order to bring this final year design project to its conclusions as specified above.

Student names

_ Aya Abed______________________ ID ______________ Signature ________________ Date ___________

_Qasem Dib_____________________ ID ______________ Signature ________________ Date ___________

_Jade El Masri__________________ ID ______________ Signature ________________ Date ___________

_ Rayan Abdul Samad El Skaff_____ ID ______________ Signature ________________ Date ___________

_ Andrea Tarabay________________ ID ______________ Signature ________________ Date ___________

The faculty advisor(s) should sign the form only after it is completed.

Faculty advisor: _Naseem Daher_____ Signature ______________ Dept: _EECE/MECH__ Date _21-Aug-2024_

Co-advisors *(if any)*: ____________________ Signature ________________ Dept: ___________ Date __________

Appendix F

WEEKLY MEETING MINUTES

F.1 August 28, 2024

- Read about SLAM and understand how the algorithm works.
- Research about implemented door-opening mechanisms and methodologies.
- Research about the principles for developing UI/UX systems.
- Research about a 2-wheel differential drive locomotion systems.
- Go over several path planing and motion planning algorithms.

F.2 September 4, 2024

- Researched about different SLAM algorithms but didn't decide on which to implement.
- Decided on lever handle, push/pull doors, proof-of-concept implementation (LoCoBot and manufactured door).
- Iterate on the user interface designed using Figma and set the main components to be present in the UI.
- Researched about path planning algorithms and indicating their compatibility to the project.

F.3 September 11, 2024

- Researched more vSLAM algorithms and made a comparison between feature-based vs direct and Sparse vs dense algorithms.

- Researched more door-opening methodologies. Established preliminary sequence of arm movements and a minimum number of joints (4-DOF) and type (revolute).
- Figma is done, now I need to research about the best platform to use to build the GUI.
- Defined data requirements to be stored in the database: nurses' login information and chosen tasks throughout the day.
- Selecting components mainly wheels and motors that depend on the weight distribution over the locomotion system
- Assessed the pros and cons of different global and local path planning. Also, went over low level controllers.

F.4 October 9 2024

- Learned how Visual Odometry works and understood how ORB-SLAM features are extracted and matched. Decided to start working on ORB-SLAM3 since it provides a good balance of accuracy, robustness, and speed.
- URDF models for the LoCoBot and structured data file (SDF) for the door will be used. Estimators will be designed for forces and torques instead of sensors.
- The connection between the robot and the user interface will be made using WIFI, and we will use state machines and not behavioral trees. For task management, we will depend on task prioritization with notifications for scheduling.
- Start with the CAD designs of the previous robot; eliminating LoCoBot and redesigning new locomotion system.
- Researched on low level control algorithms, constructed a block diagram to better understand and visualize the autonomous navigation procedure and agreed that the the outer loop states to be X,Y and ψ and inner loop states are v and ω

F.5 October 16, 2024

- Installed dependencies and ORB-SLAM3 but faced issues in testing on the EuRoC dataset due to segmentation faults and authentication errors.

- Established a more detailed door-opening sequence. However, it was advised to start with simulations as deciding on the door-opening methodology will be an iterative process (through simulations and tests).
- The color theme of the app will be blue, and we will use MongoDB as a database system.
- Continue with the CAD assemly of the base locomotion system.
- Assesed Motion planning algorithms and pointed out the pros and cons of each

F.6 October 23, 2024

- Debugged the segmentation error in the code and was able to run the simulation on the dataset. Found camera parameters specific to D435.
- Setup simulation environment with Ubuntu 16.04, ROS Kinetic, Gazebo, and simulated Kobuki base. LoCoBot arm should be used with the PyRobot package.
- Login page is done with the communication through MongoDB, but a signup page is needed to be done.
- Decide on the importance of implementing a simple spring suspension system for the 6 wheels(4 casters and 2 driving).
- Demonstrated perception and autonomous navigation using A* algorithm on TurtleBot3 on Gazebo.

F.7 November 1, 2024

- Faced authentication issues in the libeGL package. Setup ROS Noetic for the Camera but faced issues with the virtual image storage space so couldn't download RealSense SDK.
- More simulations of the Kobuki base in Gazebo was done. For the python-pip errors faced in PyRobot installation, it would be useful to ask students who have worked with it before.
- Set up the priority algorithm with ”Normal” and ”High” priority, but need to add a buffer time and work on the “status of the robot” page.
- Selection type of springs needed.

- Built the custom URDF and deployed it in a virtual world. Also, a great number of bugs encountered when converting locomotion system's CAD model to URDF. Also, derived a kinematic and dynamic model of the plant(TWDDR)

F.8 November 6, 2024

- When increasing the storage of the virtual image, the image disk got corrupted. Repeated the installation of the packages and required dependencies. Solved the authentication and segmentation in simulation and was able to generate validation of estimate vs ground truth.
- Hinged door modeled in Gazebo. Still facing PyRobot installation errors.
- Finished the status page, but need to update the map and use an actual hospital's floor map instead of Google Maps.
- Implementing the suspension system withing the CAD models
- Performed SLAM using the SLAM_toolbox, mapped the environment using Lidar sensor and loaded the map, however faced issues with autonomous navigation. Faced issues when utilzing a stereo camera instead of a Lidar for perception.

F.9 November 13, 2024

- Connecting the camera to the laptop resulted in issues in running the ROS node of the camera. Mainly, the error is: "Resource Temporary Unavailable". Still debugging the issue.
- For PyRobot installation errors, visited VRL to check dependencies. Continued debugging as many errors were faced. Everything (error+solution) is documented for inclusion in the VRL documentation database. It is better to work on the physical LoCoBot only to avoid errors due to virtual machines or other dependencies.
- Imported the application to Android Studio and debugged some errors. Need to activate the notifications system using FCM.
- Decide on start building a primary design for the locomotion system.
- Debugged the stereo camera problem and discovered the problem of autonomous navigation.

F.10 January 27, 2025

- Deployed the backend code on Render by linking the GitHub repository, and connecting it to the Flutter app, then successfully testing the API.
- Discussed various self-disinfection methods for the robot and accepted the disinfectant spray method for its safety, effectiveness, and compatibility with the robot's design.
- Introduced hardware block diagram and made sure that the motors work
- Began hands-on experimentation on the LoCoBot platform after completing the initial design and simulations
- Got RTAB-Map SLAM working. Need to setup NVIDIA Jetson TX2.

F.11 February 4, 2025

- Decided on building a new Flutter app so that users can interact with the robot and select doctors to call using Agora SDK, along with a voice interaction.
- Did battery calculations and introduced move_base and navigation stack
- Finalized system specifications including the 4-liter tank, 10-Watt pump, and confirmed flow rates and pressure requirements.
- Struggled to get the robotic arm to make basic motions due to bugs and dependencies.
- Setting up Jetson, facing authentication issues with flashing the Jetson. Fix these issues for next week.

F.12 February 10, 2025

- Built a Flutter app for video calling using Agora SDK to implement real-time video calls along with a user-friendly UI.
- Chose Design Alternative 3 for nozzle placement (shoulder-level and angled back) after evaluating three design alternatives and conducting CFD simulations.
- Implemented move_base and made sure that encoders work
- The arm is successfully moving along with the gripper and doing basic manipulation tasks, after setting up correct environment.

- Jetson set with ubuntu 18. RTAB-Map SLAM is installed on it.

F.13 February 17, 2025

- Unlocked the tablet, which is the robot's face, and enabled development mode to upload the robot's face and video calling feature. Tested the video calling up and debugged several dependencies issues.
- Finalizing battery sizing calculations and implemented and showed a successful a PWM code using the motor drivers
- Completed mechanical layout and integration, ensuring the 4-liter tank was securely placed and connected to the pump and mist nozzles.
- Captured handle images, labelled them, and attempted to use YOLOv8 for handle detection but could not as it was dependent and incompatible with Ubuntu 16.04.
- Set Intel RealSense Viewer. Fixed a lot od dependencies. RTABMAP works fine and produces occupancy grid.

F.14 February 24, 2025

- Added ROS Bridge for the communication between the robot and the Flutter app. Also, decided to use Websocket communication to enable real-time monitoring.
- Built the whole hardware part and implemented a teleop_keyboard code using rosserial.
- Integrated hydraulic subsystem, installed five nozzles, and performed leak detection using secure connections and effective sealing methods.
- Trained YOLOv8 on Google Colab. However still deployment is impossible with LoCoBot's environment.
- Controlling Motor drivers via the NVIDIA Jetson TX2 with Arduino Mega.

F.15 March 3, 2025

- The app crashed using the Lottie JSON animation on the tablet, thus, we need another compatible solution. Decided to use WSL instead of a VM integration. Installed WSL with Ubuntu 18.04, Node.js, ROS Melodic, and MongoDB.

- Finished electrical subsystem, connected the water level sensor and relay module, and performed dry-run tests to ensure functionality.
- Mapped Bechtel and Oxy L4. In addition to studying the open loop behavior of the two motors
- Attempted to train a YOLOv3-tiny model, still in progress because of bugs.
- Planning Autonomous Navigation and mapped the corridor oxy. Found out about the corridor problem. Mapped oxy and bechtel several times.

F.16 March 10, 2025

- started developing move_base and the whole navigation stack on the Nvidia Jetson Tx2
- Conducted pre-implementation testing of fluid flow, pressure, and nozzle coverage to ensure the system operated as expected.
- Started researching about the best way for communication between the robot and the phone. Tried several solutions, all failed due to network privacy and firewall.
- Deployed YOLOv3 successfully on the LoCoBot and sustained stable real-time handle detection performance.
- Mapped oxy with and without features several times. Fine-tuned RTAB-Map parameters.

F.17 March 17, 2025

- Created a ZeroTier network to allow communication between the phone and turtlesim using WIFI, and it worked successfully. Now the turtle moves using the Flutter app.
- implemented a successful PID controller for the base and developed move_base on the NVIDIA, and successfully tested communication.
- Integrated self-disinfection system into *Sharec* robot for structural fit testing and confirmed no interference with robot mobility.
- Implemented early 3D pose estimation but encountered large errors due to calibration and transformation mismatches.
- Testing out localization: IMU error and time synchronization issue. Plus, faced odometry issues.

F.18 March 24, 2025

- Decided to generate the voice of SHARE-C using ChatGPT, as other AI generators sounded too robotic, not gender neutral, and not with a Lebanese accent. Then adjusted the pitch and speech using Audacity.
- Fully integrated electrical components, ensuring disinfection cycles and verifying proper coordination of sensors and pump control.
- Fully assembled the whole electric and mechanincal system (SHARE-C's body and base) with motor drivers, microcontroller (Arduino Mega) and Nvidia Jetson Tx2.
- Optimized the pose estimation process via depth filtering and frame transformation correction.
- Tested teleop keyboard package.

F.19 April 7, 2025

- By publishing commands to /keyboard_cmd instead of /turtle1/cmd_vel, the robot now moves using the Flutter app. Also, a new map on Flutter was designed to display OXY building map. Lastly, a fluid level sensor display was added.
- Completed final testing, ensuring the system operated autonomously with proper fluid refilling alerts and reliable pump activation.
- Added a voltage sensor to monitor the battery's voltage, implemented wireless communication using RealVNC
- Faced Inverse Kinematics (IK) issues in grasping, especially reaching the handle at certain angles. Modified pre-grasp poses and offsets to avoid unfeasible IK solutions at the handle's height.

F.20 April 14, 2025

- Subscribed to percentage via a web socket service. Also, we subscribed to the rtabmap/odom to view the live location of the robot. Now, we can view the battery and location of share-c in real time.
- Conducted final assessment, confirming the self-disinfection system met all performance, efficiency, and safety standards, ready for deployment in the *Sharec* robot.

- Tested move_base in the corridor
- Designed coordinated base and arm motion planning scripts to improve door-pulling behavior.
- NVIDIA turning off. Facing issues in Autonomous Navigation.

F.21 April 22, 2025

- The video calling app was crashing, so we disabled the hardware accelerator in the tablet and let the app run only using the CPU and not the GPU. Then, the app stopped crashing and was working smoothly.
- Tried debugging move_base issues.
- Completed end-to-end door-opening process from detection, grasping, unlatching, pulling, pushing, and travel.
- AMCL is getting lost due to missing odometry.

Bibliography

[1] International Council of Nurses (ICN). *ICN COVID-19 Update Report.* Accessed: 2024-11-16. 2021. URL: https://www.icn.ch/sites/default/files/inline-files/ICN%20COVID19%20update%20report%20FINAL.pdf.

[2] World Health Organization. *Health and care worker deaths during COVID-19.* https://www.who.int/news/item/20-10-2021-health-and-care-worker-deaths-during-covid-19. Accessed: 2024-11-15. 2021.

[3] Jennifer Cohen and Yana van der Meulen Rodgers. "Contributing factors to personal protective equipment shortages during the COVID-19 pandemic". In: *Preventive Medicine* 141 (2020). PMCID: PMC7531934, p. 106263. DOI: 10.1016/j.ypmed.2020.106263. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC7531934/.

[4] National Nurses United. *New survey of nurses provides frontline proof of widespread employer, government disregard for nurse and patient safety.* https://www.nationalnursesunited.org/press/new-survey-results. Accessed: 2024-11-15. 2020.

[5] Petros Galanis et al. "Nurses' burnout and associated risk factors during the COVID-19 pandemic: A systematic review and meta-analysis". In: *Journal of Advanced Nursing* 77.8 (2021). PMCID: PMC8250618, pp. 3286–3302. DOI: 10.1111/jan.14839. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC8250618/.

[6] Sinu Jose, Maneesha C. Cyriac, and Manju Dhandapani. "Health Problems and Skin Damages Caused by Personal Protective Equipment: Experience of Frontline Nurses Caring for Critical COVID-19 Patients in Intensive Care Units". In: *Indian Journal of Critical Care Medicine* 25.2 (2021). PMCID: PMC7922454, pp. 134–139. DOI: 10.5005/jp-journals-10071-23713. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC7922454/.

[7] Raymond Haward, Ridhima G, and Meenakshi Kalyan. "The Impact of Personal Protective Equipment on Healthcare Workers on COVID-19 Duty in a Tertiary Care Hospital in South India". In: *Cureus* 15.7 (2023). PMCID: PMC10425167, e41910. DOI: 10.7759/cureus.41910. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC10425167/.

[8] Heather L. Evans et al. "Development of a Sterile Personal Protective Equipment Donning and Doffing Procedure to Protect Surgical Teams from SARS-CoV-2 Exposure during the COVID-19 Pandemic". In: *Surgical Infections* 21.8 (2020). PMID: 32628871, pp. 671–676. DOI: 10.1089/sur.2020.140. eprint: https://doi.org/10.1089/sur.2020.140. URL: https://doi.org/10.1089/sur.2020.140.

[9] Centers for Disease Control and Prevention (CDC). *Healthcare-Associated Infections (HAIs)*. https://www.cdc.gov/healthcare-associated-infections/about/index.html. Accessed: 2024-11-15. 2024.

[10] Md Samiul Haque Sunny et al. "Chapter 1 - Assistive robotic technologies: An overview of recent advances in medical applications". In: *Medical and Healthcare Robotics*. Ed. by Olfa Boubaker. Medical Robots and Devices: New Developments and Advances. Academic Press, 2023, pp. 1–23. ISBN: 978-0-443-18460-4. DOI: https://doi.org/10.1016/B978-0-443-18460-4.00004-4. URL: https://www.sciencedirect.com/science/article/pii/B9780443184604000044.

[11] Tyra Wallén. "Investigating the Emotional Impact of Social Robots: A Comparative Study on the Influence of Appearance and Application Area on Human Emotions". Supervisor: Sam Thellman, Examiner: Tom Ziemke. MA thesis. Linköping University, Department of Computer and Information Science, 2023. URL: https://liu.diva-portal.org/smash/get/diva2:1816944/FULLTEXT01.pdf.

[12] Hugh Durrant-Whyte and Tim Bailey. "Simultaneous Localization and Mapping: Part I". In: *IEEE Robotics & Automation Magazine* 13.2 (2006), pp. 99–110. DOI: 10.1109/MRA.2006.1638022. URL: https://ieeexplore.ieee.org/document/1638022.

[13] Cesar Cadena et al. "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age". In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1309–1332. DOI: 10.1109/TRO.2016.2624754.

[14] Weifeng Chen et al. "An Overview on Visual SLAM: From Tradition to Semantic". In: *Remote Sensing* 14.13 (2022), p. 3010. DOI: 10.3390/rs14133010. URL: https://www.mdpi.com/2072-4292/14/13/3010.

[15] Amin Basiri, Valerio Mariani, and Luigi Glielmo. "Improving Visual SLAM by Combining SVO and ORB-SLAM2 with a Complementary Filter to Enhance Indoor Mini-Drone Localization under Varying Conditions". In: *Drones* 7 (June 2023), p. 404. DOI: 10.3390/drones7060404.

[16] Andrew J. Davison et al. *MonoSLAM: Real-time single camera SLAM*. Accessed: 2024-11-16. 2007. URL: https://www.doc.ic.ac.uk/~ajd/Publications/davison_etal_pami2007.pdf.

[17] Georg Klein and David Murray. "Parallel Tracking and Mapping for Small AR Workspaces". In: *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. 2007, pp. 225–234. DOI: `10.1109/ISMAR.2007.4538852`.

[18] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. "ORB-SLAM: A Versatile and Accurate Monocular SLAM System". In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1147–1163. DOI: `10.1109/TRO.2015.2463671`.

[19] Raul Mur-Artal and Juan D. Tardos. "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras". In: *IEEE Transactions on Robotics* 33.5 (Oct. 2017), pp. 1255–1262. ISSN: 1941-0468. DOI: `10.1109/tro.2017.2705103`. URL: `http://dx.doi.org/10.1109/TRO.2017.2705103`.

[20] Carlos Campos et al. "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM". In: *IEEE Transactions on Robotics* 37.6 (Dec. 2021), pp. 1874–1890. ISSN: 1941-0468. DOI: `10.1109/tro.2021.3075644`. URL: `http://dx.doi.org/10.1109/TRO.2021.3075644`.

[21] Liang Yang et al. "Survey of robot 3D path planning algorithms". In: *Journal of Control Science and Engineering* 2016.1 (2016), p. 7426913.

[22] Li-sang Liu et al. "Path planning for smart car based on Dijkstra algorithm and dynamic window approach". In: *Wireless Communications and Mobile Computing* 2021.1 (2021), p. 8881684.

[23] Xiaohui Wang, Xi Ma, and Zhaowei Li. "Research on SLAM and path planning method of inspection robot in complex scenarios". In: *Electronics* 12.10 (2023), p. 2178.

[24] Yang Ying et al. "Path planning of mobile robot based on Improved RRT Algorithm". In: *2019 Chinese Automation Congress (CAC)*. 2019, pp. 4741–4746. DOI: `10.1109/CAC48633.2019.8996415`.

[25] Haiyan Tu et al. "Improved RRT global path planning algorithm based on Bridge Test". In: *Robotics and Autonomous Systems* 171 (2024), p. 104570.

[26] Oussama Khatib. "Real-time obstacle avoidance for manipulators and mobile robots". In: *The international journal of robotics research* 5.1 (1986), pp. 90–98.

[27] Jianxin Ren et al. "SLAM, path planning algorithm and application research of an indoor substation wheeled robot navigation system". In: *Electronics* 11.12 (2022), p. 1838.

[28] Sean Quinlan and Oussama Khatib. "Elastic bands: Connecting path planning and control". In: *[1993] Proceedings IEEE International Conference on Robotics and Automation*. IEEE. 1993, pp. 802–807.

[29] Christoph Rösmann et al. "Trajectory modification considering dynamic constraints of autonomous robots". In: *ROBOTIK 2012; 7th German Conference on Robotics*. VDE. 2012, pp. 1–6.

[30] Tesfaye Deme Tolossa et al. "Trajectory tracking control of a mobile robot using fuzzy logic controller with optimal parameters". In: *Robotica* (2024), pp. 1–24.

[31] Kuanqi Cai et al. "Mobile robot path planning in dynamic environments: A survey". In: *arXiv preprint arXiv:2006.14195* (2020).

[32] Hongwei Qin et al. "Review of autonomous path planning algorithms for mobile robots". In: *Drones* 7.3 (2023), p. 211.

[33] Ibrahim Harbi et al. "Model-Predictive Control of Multilevel Inverters: Challenges, Recent Advances, and Trends". In: *IEEE Transactions on Power Electronics* 38.9 (2023), pp. 10845–10868. DOI: 10.1109/TPEL.2023.3288499.

[34] Alícia Casals et al. "A Preliminary Approach to a Wheelchair with Embedded Exoskeleton Capabilities". In: *Iberian Robotics conference*. Springer. 2022, pp. 252–263.

[35] Stan Arledge et al. "RESNA Wheelchair Service Provision Guide." In: *Resna (Nj1)* (2011).

[36] Günter Ullrich et al. "Automated guided vehicle systems". In: *Springer-Verlag Berlin Heidelberg. doi* 10 (2015), pp. 978–3.

[37] Gabriel Alfredo Alvarado García and Fávell Eduardo Núñez Rodríguez. "Gesture-Based Control of an OMRON Viper 650 Robot". In: *Engineering Headway* 12 (2024), pp. 49–59.

[38] Veronica Ahumada-Newhart et al. "Evaluation of the toyota human support robot (HSR) for social interaction and learning". In: *International journal of technology, knowledge and society* 19.1 (2023), p. 21.

[39] American Society of Civil Engineers. "Minimum design loads and associated criteria for buildings and other structures". In: American Society of Civil Engineers. 2017.

[40] Isaac C Higgins. *Impact of changes in the design methodology between the American Concrete Institute's draft code provisions for GFRP-reinforced concrete and the ACI 440.1 R-15 guide for the design and construction of structural concrete reinforced with FRP bars*. Widener University, 2020.

[41] Bing Wang et al. "A modified Johnson–Cook constitutive model and its application to high speed machining of 7050-T7451 aluminum alloy". In: *Journal of Manufacturing Science and Engineering* 141.1 (2019), p. 011012.

[42] The Verge. “Boston Dynamics partners with Assa Abloy to let the dogs in”. In: *The Verge* (Sept. 2024). URL: https://www.theverge.com/2024/9/23/24252240/assa-abloy-boston-dynamics-spot-robot-patrol-dog-door-smart-entry.

[43] Wim Meeussen et al. “Autonomous door opening and plugging in with a personal robot”. In: *2010 IEEE International Conference on Robotics and Automation*. 2010, pp. 729–736. DOI: 10.1109/ROBOT.2010.5509556.

[44] Tristan Mueller, Steffen Mueller, and Horst-Michael Gross. “Door Manipulation as a Fundamental Skill Realized on Robots With Differential Drive”. In: *ISR Europe 2023; 56th International Symposium on Robotics*. 2023, pp. 338–345.

[45] Samuel A. Prieto et al. “Passing through Open/Closed Doors: A Solution for 3D Scanning Robots”. In: *Sensors (Basel)* 19.21 (2019), p. 4740. DOI: 10.3390/s19214740.

[46] Yiannis Karayiannidis et al. “An Adaptive Control Approach for Opening Doors and Drawers Under Uncertainties”. In: *IEEE Transactions on Robotics* 32.1 (2016), pp. 161–175. DOI: 10.1109/TRO.2015.2506154.

[47] Gyuree Kang et al. “A versatile door opening system with mobile manipulator through adaptive position-force control and reinforcement learning”. In: *Robotics and Autonomous Systems* 180 (2024), p. 104760. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2024.104760. URL: https://www.sciencedirect.com/science/article/pii/S0921889024001441.

[48] Caterina Neef, Katharina Linden, and Anja Richert. “Exploring the Influencing Factors on User Experience in Robot-Assisted Health Monitoring Systems Combining Subjective and Objective Health Data”. In: *Applied Sciences* 13.6 (2023). ISSN: 2076-3417. DOI: 10.3390/app13063537. URL: https://www.mdpi.com/2076-3417/13/6/3537.

[49] Marcin Bartosiak et al. “Advanced Robotics as a Support in Healthcare Organizational Response: A COVID-19 Pandemic Case Study”. In: *Healthcare Management Forum* 35.1 (2022). Epub 2021 Oct 13, pp. 11–16. DOI: 10.1177/08404704211042467.

[50] Yang-Yen Ou et al. “A Friendly Interface Medical Service Robot with Fault-tolerant Voice Recognition and Mobile Phone Remote Control”. In: *2022 10th International Conference on Orange Technology (ICOT)*. Shanghai, China, 2022, pp. 1–5. DOI: 10.1109/ICOT56925.2022.10008156.

[51] Justinas Mišeikis et al. “Lio-A Personal Robot Assistant for Human-Robot Interaction and Care Applications”. In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 5339–5346. DOI: 10.1109/LRA.2020.3007462.

[52] A. Sacino et al. “Human- or object-like? Cognitive anthropomorphism of humanoid robots”. In: *PLoS One* 17.7 (July 2022), e0270787. DOI: 10.1371/journal.pone.0270787.

[53] Mingming Li et al. “Effects of robot gaze and voice human-likeness on users’ subjective perception, visual attention, and cerebral activity in voice conversations”. In: *Computers in Human Behavior* 141 (2023), p. 107645. ISSN: 0747-5632. DOI: https://doi.org/10.1016/j.chb.2022.107645. URL: https://www.sciencedirect.com/science/article/pii/S0747563222004654.

[54] Shi-Peng Chen et al. “Comparison of 2D and 3D LiDARs Trajectories and AMCL Positioning in ROS-Based move$_baseNavigation$”. In: *2023 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*. 2023, pp. 1–6. DOI: 10.1109/COINS57856.2023.10189271.

[55] Rajesh Kannan Megalingam, Anandu Rajendraprasad, and Sakthiprasad Kuttankulangara Manoharan. “Comparison of Planned Path and Travelled Path Using ROS Navigation Stack”. In: *2020 International Conference for Emerging Technology (INCET)*. 2020, pp. 1–6. DOI: 10.1109/INCET49848.2020.9154132.

[56] Sergio Hernandez-Mendez et al. “Design and implementation of a robotic arm using ROS and MoveIt!” In: *2017 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*. IEEE. 2017, pp. 1–6.

[57] AltexSoft. *Comparing Database Management Systems: MySQL, PostgreSQL, MSSQL Server, MongoDB, Elasticsearch, and Others*. Accessed: 2024-06-17. 2024. URL: https://www.altexsoft.com/blog/comparing-database-management-systems-mysql-postgresql-mssql-server-mongodb-elasticsearch-and-others/.

[58] Rached Dhaouadi and A Abu Hatab. “Dynamic modelling of differential-drive mobile robots using lagrange and newton-euler methodologies: A unified framework”. In: *Advances in robotics & automation* 2.2 (2013), pp. 1–7.