Fully Autonomous Wheelchair for Indoor Mobility: A Hybrid Approach to Mapping and Navigation with ROS

Yaseen Abduljalil Sultan Al-Qadasi [1,*]
Hatem Al-Dois [2]
Mohammed Mohammed AlObaidi [3]
Ali Abdullah Qayid [3]
Fareed Ameen Al-Abbassi [3]
Emad Abdulwahed Saleh [3]

Received: 02/10/2024
Revised: 11/10/2024
Accepted: 7/12/2024

[1] Teaching Assistant of Robotics & Control Systems - Faculty of Engineering - Sana'a University. **ORCID: 0009-0002-9674-1075**
[2] Associate Professor of Robotics & Control, Electrical Engineering Department, Faculty of Engineering - Ibb University.
[3] Mechatronics Engineering Department, Faculty of Engineering - Sana'a University.
[*] Corresponding Author Designation, Email: qadasiyaseen1997@gmail.com

Fully Autonomous Wheelchair for Indoor Mobility: A Hybrid Approach to Mapping and Navigation with ROS

Yaseen Abduljalil Al-Qadasi
Mechatronics Engineering Department, Faculty of Engineering - Sana'a University
Sana'a, Yemen
qadasiyaseen1997@gmail.com

Hatem Al-Dois
Electrical Engineering Department, Faculty of Engineering - Ibb University
Ibb, Yemen
haldois@yahoo.com

Mohammed Mohammed *AlObaidi*
Mechatronics Engineering Department, Faculty of Engineering - Sana'a University
Sana'a, Yemen
alobaidy415@gmail.com

Ali Abdullah Qayid
Mechatronics Engineering Department, Faculty of Engineering - Sana'a University
Sana'a, Yemen
Alawiiabdullah1997@gmail.com

Fareed Ameen Al-Abbassi
Mechatronics Engineering Department, Faculty of Engineering - Sana'a University
Sana'a, Yemen
fareedalabassi@gmail.com

Emad Abdulwahed Saleh
Mechatronics Engineering Department, Faculty of Engineering - Sana'a University
Sana'a, Yemen
emad773544379@gmail.com

Abstract— People with severe physical disabilities often find existing mobility solutions inadequate, necessitating the need for reliable autonomous systems. This paper introduces a ROS-based autonomous wheelchair that utilizes the Kinect Xbox sensor for visual perception, RTAB-Map for mapping, and AMCL for localization. Additionally, it incorporates additional sensors that supply odometry and obstacle detection data to the ROS navigation stack, enabling precise path planning and real-time obstacle avoidance. Initial tests using the Gmapping SLAM algorithm revealed synchronization and accuracy issues. However, a hybrid RTAB-Map and AMCL approach resolved these issues, delivering superior detail in the maps and reliable navigation indoors. The system consistently reached its target destinations, proving robustness and accuracy. The Qt framework-built user-friendly interface enables destination selection, and AI-powered voice recognition integration is currently in progress to facilitate hands-free control.

Keywords— *ROS, Kinect Sensor, IMU, SLAM, RTAB-Map, Navigation, AMCL (Adaptive Monte Carlo Localization), Gazebo, Odometry.*

I. INTRODUCTION

The concept of a wheelchair originated well before the birth of Christ, approximately four or five centuries ago, and over time, it has evolved to incorporate the latest technologies to assist those in need. Initially, wheelchairs were designed to be manually pushed, allowing disabled individuals to achieve mobility. The development of electrical motors led to the introduction of the concept of motorized wheelchairs, which allow individuals to move independently without external assistance. However, a significant problem surfaced when manufacturers considered the methods of controlling the motors. The issue arose from the various types of disabilities that patients may have, leading to a shift in the developmental focus towards the field of control [**1**].

The interfacing and controlling methods developed from pushbuttons to joysticks to the smart body connected devices. The modern intelligent interfacing technologies include Brain Computer Interface (BCI) [**2**], muscles controlling neurons, and eye gaze interface. Since the last intelligent control technique was discussed and implemented in reality by our ex-classmates that's why we turned to work on more autonomous, reliable, and powerful mode as it will be explained after.

The new growing field of controlling vehicles in general is self-driving. As a developmental step, the field started by designing semiautomatic robots, for example, automatic obstacle avoidance systems or automatic parking systems. However, the development of the artificial intelligence field has led to the construction and use of fully automatic driven vehicles, also known as self-driving vehicles [**3**]. For people with extreme disabilities, we have applied the concepts of self-driven vehicles to design and implement a system to automatically move the wheelchair into the desired pose (and orientation).

This paper comprises seven main sections. The autonomous wheelchair addresses mobility challenges for individuals with severe disabilities in Section [II]. Section [III] reviews relevant research in ROS-based wheelchair systems and identifies gaps our project addresses. Section [IV] covers the system design and simulation, including theoretical background, CAD modeling, and path planning. Section [V] details the system implementation, including hardware setup and software integration. Section [VI] presents the results and discussion, highlighting the autonomous navigation

process, localization, and challenges addressed. Finally, Section [VII] concludes with key findings and suggests future work to enhance system capabilities.

II. PROBLEM STATEMENT

The number of people living with disabilities, primarily due to war injuries and natural diseases, has been increasing, thus contributing to a high demand for assistive technologies. These victims referred to host severe physical impairments, such as paralysis, Parkinson's disease, at times tetraplegia, and other diseases making it impossible for a patient to use a typical joystick-based wheelchair. Much research has also been done on alternative techniques, including voice command, EEG-based, and eye-tracking systems. However, existing systems have proven erratic, dangerous, unstable, and imprecise. With that, there is a serious need for a reliable, secure, and efficient wheelchair control system that is customized for people with severe disabilities.

The primary motivation of this pursuit is to address the unsolved challenges surrounding mobility and independence for this underserved population, which affects a significant proportion of the national and international community. The key objectives are:

- To design and implement an efficient system that enables individuals with severe physical disabilities to control their wheelchairs effectively.
- To ensure the safety of wheelchair users by applying robust algorithms and control mechanisms.
- To apply the knowledge and concepts of mechatronics engineering to develop a comprehensive assistive technology solution.

III. LITERATURE REVIEW

It has been a decade since ROS (Robot Operating System) first found applications for autonomous wheelchair navigation, with significant improvements in flexibility, affordability, and the integration of sensors. Zhang and Xu showcased a ROS-powered voice-controlled smart wheelchair in 2015, highlighting its superior speed and flexibility compared to conventional wheelchair designs [**4**]. Xu et al. proposed a ROS-based mobile robot around that time, focusing on adaptable location and mapping features for wheelchair navigation systems [**5**]. In 2016, Nasri et al. implemented a new geo-localization algorithm together with omnidirectional sensors to navigate different paths accurately [**6**]. Marrón and his team suggested an open robotic platform using ROS and sensor fusion for the rapid integration of state-of-the-art robotics developments into intelligent wheelchairs (IWs) [**7**]. Subsequent research by Grewal et al. [**8**] and Gatesichapakorn et al. [**9**] implemented LIDAR and RGB-D cameras, improving obstacle detection and mapping accuracy. More recently, studies by Sun et al. [**10**] and Islam et al. [**11**] focused on SLAM-based navigation for elderly support and cloud-integrated control, respectively. Kappel and Ferreira developed a socially aware navigation system for motorized wheelchairs, prioritizing user comfort and adherence to social rules during indoor navigation [**12**].

Although there have been significant strides in the development of ROS-based autonomous wheelchair systems, our project is one of the few to emphasize cost-effectiveness and accessibility. In contrast to many of these, we used a Kinect Xbox sensor instead of the more expensive LiDAR, achieving reliable mapping with the RTAB-Map algorithm and precise localization from the AMCL. Our system combines the IMU, rotary encoders, and ultrasonic sensors with efficient navigation within the ROS framework. It has the Interface made user-friendly with Qt and Python, and there are voice control system integration perspectives. By balancing affordability and performance, our project offers a practical, inclusive mobility solution for individuals with disabilities.

IV. SYSTEM DESIGN AND SIMULATION

The System Design and Simulation section details the modeling of the wheelchair using URDF/CAD and testing in Gazebo. It integrates sensors (Kinect, IMU, encoders) and applies SLAM for mapping and path planning, ensuring accurate navigation in the simulated environment before physical implementation.

A. Theoretical Background

Differential drive robots are widely used in autonomous systems due to their simplicity and affordability. They consist of two main drive wheels, each controlled independently, allowing the robot to steer by adjusting the relative speed of these wheels. The system is nonholonomic, meaning it has constraints that prevent the robot from moving directly in all directions, limiting its motion to the x-y plane [**13**].

Kinematics of differential drive robots involves determining the robot's position and orientation (pose) based on wheel velocities. Forward kinematics calculates the robot's new pose given initial conditions and wheel speeds. The robot's movement is characterized by its instantaneous center of curvature (ICC), around which it rotates.

1. *Forward Kinematics*: If the initial pose is (x,y,θ) and wheel velocities V_{left} and V_{right}, then the new pose (x′,y′,θ′) after a time interval δt is calculated using [**14**]:

$$\omega = \frac{V_{\text{right}} - V_{\text{left}}}{L}$$

where L is the distance between the wheels. The updated orientation is:

$$\theta' = \theta + \omega\delta t$$

The new position (x′,y′) is:

$$x' = x + R\sin(\theta' - \theta)$$

$$y' = y - R\cos(\theta' - \theta)$$

where R (the radius of the ICC) is:

$$R = \frac{L}{2} \times \frac{V_{\text{left}} + V_{\text{right}}}{V_{\text{right}} - V_{\text{left}}}$$

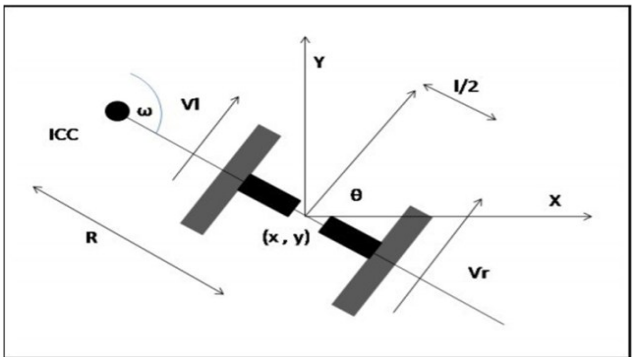

Fig. 1. Kinematic Configuration of Differential Drive Robot [**14**]

2. *Special Case in Motion:*
   - Straight Line Motion: $V_{\text{left}} = V_{\text{right}}$, so ω = 0, gives linear motion.
   - Rotation in Place: $V_{\text{left}} = -V_{\text{right}}$, leading to rotation around the robot's center.
   - Circular Path: Different V_{left} and V_{right} create a curved trajectory around an ICC.
3. *Inverse Kinematics:* For a given pose (x′,y′,θ′), inverse kinematics determines the wheel velocities V_{left} and V_{right}.
4. *Odometry and Sensor Integration:* Odometry, which works based on wheel encoders, plays a key role in estimating wheel velocities. These values continuously update the position of the robot in the forward kinematics equations. The rolling of the casters makes the robot stable, correcting for unevenness in terrain.

B. CAD Modeling and Sensor Integration

The modeling process for our wheelchair was initialized by gathering all required dimensions for the off-the-shelf powered wheelchair – TSS300. These measurements were critical to ensuring that 2D and 3D models in SolidWorks had updated detail in major components like the chassis, wheels, and sensor locations. Once the design was completed, the 3D models were converted into a Unified Robot Description Format (URDF) using a plugin, which is called sw2urdf that can convert any SLD parts into its equivalent URDF file which we really need to insert our wheelchair in ROS for simulation.

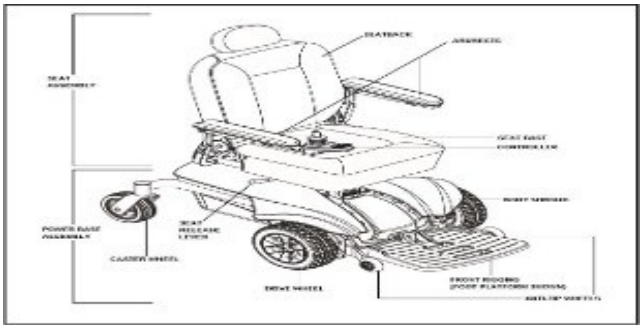

Fig. 2. TSS 300 Electrical Wheelchair Model [**15**]

Note that figure 3 illustrate two different wheelchair designs based on the camera's position. These design modifications were evaluated during simulation and adjusted according to mapping results from SLAM algorithms (Gmapping or Rtabmap). Further details will be covered in section VI.

In Gazebo, we integrated the URDF model to test the robot's functionality in a virtual environment. Essential sensors, including the Kinect, IMU, and encoders, were added to replicate real-world behavior. This allowed us to evaluate the wheelchair's performance, apply SLAM algorithms, and test path planning strategies. The simulation provided a critical platform to refine our design and validate the system before proceeding to physical implementation.

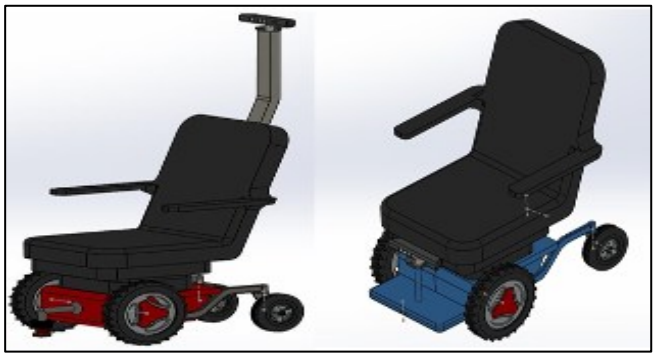

Fig. 3. Two Different Wheelchair Designs Based on the Camera's Position

C. Mapping, Localization and Path Planning

The depthimage_to_laserscan nodelet converts 3D environmental data from the robot's depth sensor (Kinect) into 2D laser scan data, enabling SLAM algorithms such as Gmapping and Rtabmap. Nodelets ensure fast, efficient processing by minimizing network overhead. The URDF model includes simulation-specific tags, such as "collision" and "inertial," while plugins like cliff sensors, contact sensors, and the differential drive plugin simulate behavior and compute odometry. The depth camera plugin simulates the Kinect's outputs. We use Rviz and Gazebo for visualisation and mapping, monitoring sensor data in real-time within a custom environment. SLAM algorithms generate maps while localizing the robot, and AMCL is

employed for precise localisation within pre-built maps. The Gmapping package creates 2D maps during teleoperation, with the final map saved for future use.

Additionally, we explore RTAB-Map, which generates both 2D and 3D maps using RGB-D data, making it suitable for large-scale environments. RTAB-Map is effective for both mapping and localization, enabling the wheelchair to autonomously navigate to target locations while avoiding obstacles. Path planning is performed using the ROS navigation stack, with target positions set in Rviz for autonomous movement.

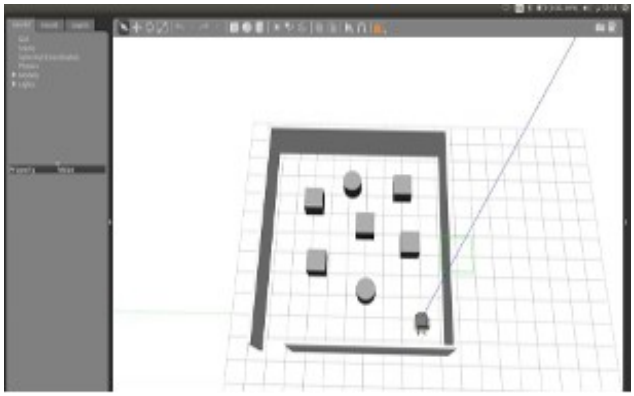

Fig. 4. Wheelchair in the room environment – Gazebo

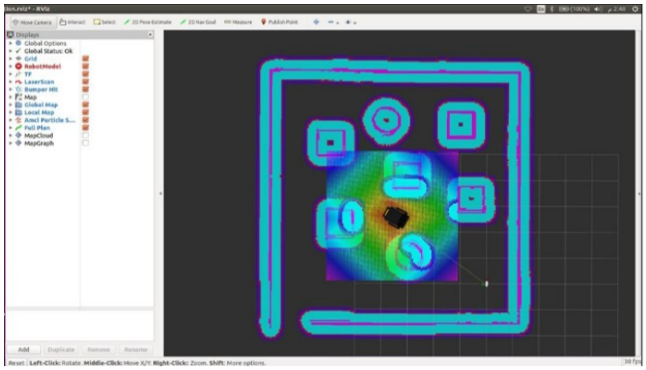

Fig. 5. Autonomous Navigation of Wheelchair - Rivz

V. System Implementation

In this section, we describe the hardware arrangement of the autonomous wheelchair, along with the software components. It also describes sensor-motor integration, low-level processing of sensor data, control, high-level processing for mapping and navigation, and the user interface that would be needed for its easy control.

A. Hardware Setup

The hardware setup consists of various elements centered on the microcontroller, Arduino Mega, together with main sensors including Kinect, IMU (MPU6050), ultrasonic sensor, and rotary encoders. Electrical layout includes piecing these together with motor drivers [BTS7960] and power supply, which will be 12V batteries, for efficient movement control. Each component is mounted for optimum functionality - Kinect on a steel base 22 cm from the footrest for effective mapping of the environment, while the encoders and ultrasonic sensors are placed in a way that would ensure correct navigation and the detection of dynamic obstacles. The cooling fan also acts to ensure that the control cabinet, hosting Arduino and motor drivers, does not heat up. This in return has an effect on performance, ensuring that there is controlled interaction of the components given to make sure proper and precise data processing is allowed with maximum autonomous operation.

Key components of the hardware setup necessary to implement an autonomous wheelchair:

- *Microcontroller:* Arduino Mega 2560 handles motor control, sensor inputs, and data processing.
- *Sensor Integration:* Kinect for mapping (mounted 22 cm above footrest), IMU (MPU6050) for accurate odometry, ultrasonic sensors for additional obstacle detection.
- *Drive System:* Two motors with built-in encoders for precise differential drive control.
- *Power Supply:* Two 12V batteries connected in series, with a DC-DC converter for regulated voltage.
- *Wiring and Connections:* Jumper cables and indoor electrical wires ensure stable communication.
- *Control Cabinet:* Houses electronics with cooling fan to prevent overheating during operation.

This hardware configuration provides a robust and well-integrated system for autonomous navigation and sensor data processing, forming the foundation for efficient wheelchair operation in real-world environments.

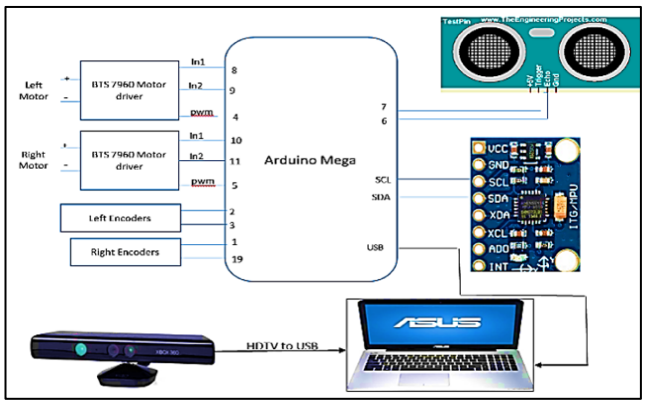

Fig. 6. Hardware High-Level Design

B. Software Implementation

The software architecture of the autonomous wheelchair integrates multiple layers to manage sensor data acquisition, high-level navigation planning, and real-time control. The overall structure is illustrated in the figure, which highlights the relationship between the **Low-Level Processing**, **High-**

Level Processing, and **User Interface** components (see Fig. 7).

1) Low-Level Processing: This layer handles sensor inputs and pre-processes them for the higher-level algorithms. Key components include:

a) Sensors: The wheelchair relies on various sensors such as the Kinect v2 for depth sensing, an ultrasonic sensor for distance measurements, an MPU for orientation (Euler angles), and wheel encoders for tracking position (ticks).

b) Camera Driver (openni): The Kinect v2 provides RGB and depth images that are pre-processed for mapping and navigation. The camera driver handles image registration and calibration.

c) Depth Image to Laser Scan: The depth data from the Kinect is converted to a laser scan format compatible with SLAM algorithms, ensuring efficient real-time mapping and obstacle detection.

d) Wheelchair Driver: Sensors, actuators, and control systems communicate via this basic module. The PID controller and wheelchair controller receive differential speed instructions using odometry data (ticks) and sensor input for accurate motion control.

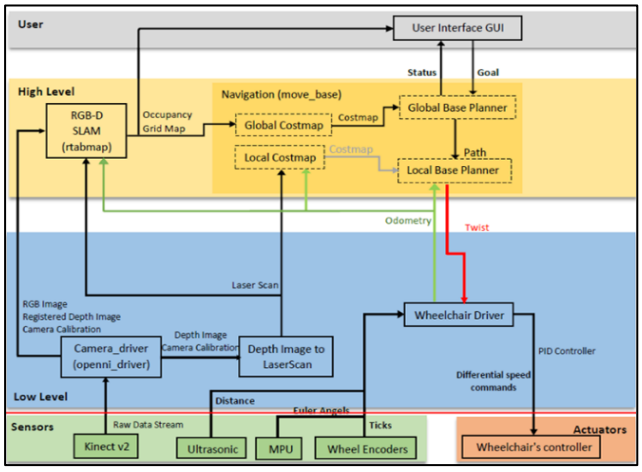

Fig. 7. Designed system architecture with sensing, processing, and user interaction layers.

2) High-Level Processing: The high-level processing layer involves the coordination of mapping, localization, and navigation strategies:

a) RGB-D SLAM (rtabmap): This SLAM algorithm creates detailed occupancy grid maps using data from the low-level sensors. It generates a consistent map of the environment while localizing the wheelchair within it.

b) Navigation Stack (move_base): The navigation stack includes:

i) *Global and Local Costmaps:* These costmaps are generated using sensor data to plan collision-free paths for the wheelchair. The global costmap relies on the pre-built map, while the local costmap dynamically adjusts to obstacles detected in real-time.

ii) *Global and Local Base Planners***:** These planners generate paths and adjust the wheelchair's movement, ensuring it follows the planned route while adapting to dynamic obstacles.

3) User Interface: The GUI for the autonomous wheelchair was designed to provide a simple and intuitive interface that abstracts the complexity of the underlying ROS system, ensuring accessibility for users with minimal technical expertise. The interface features large, clearly labeled buttons for essential functions such as starting, stopping, and navigating to predefined positions (Pose 1, Pose 2, Pose 3, and Home). It also provides a visual representation of the mapped environment, enabling users to monitor the wheelchair's current position and navigate intuitively within the mapped space.

a) Design and Functionality: The GUI communicates directly with the high-level processing layer using ROS's Actionlib framework, enabling robust goal management and real-time feedback during execution. Key components include:

i. *Start/Stop Buttons:* Allow users to initiate or halt wheelchair operations.

ii. *Pose Navigation:* Buttons for predefined positions simplify navigation tasks.

iii. *Real-time Feedback:* Visual updates display the robot's position and progress, improving situational awareness.

iv. Actionlib Integration:

- *Goals:* Sends navigation goals from the GUI to the action server.
- *Status and Feedback:* Provides real-time updates on task progress and intermediate states.
- *Cancel and Result:* Allows task cancellation and communicates final outcomes to the GUI.

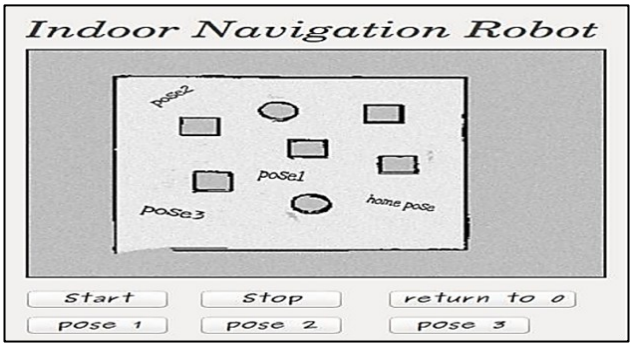

Fig. 8. Qt-Based User Interface for Navigation.

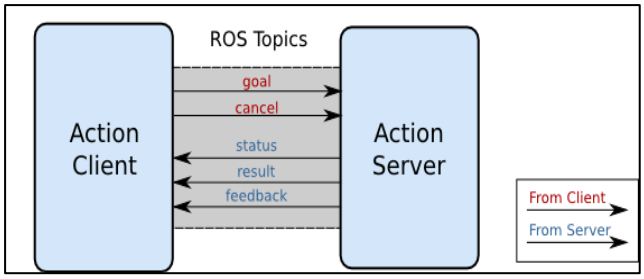

Fig. 9. Action Interface – Communicate via ROS topics [**17**]

b) Testing: The GUI was successfully tested in Gazebo simulations, demonstrating its capability to send navigation goals, receive feedback, and control the wheelchair effectively within a virtual environment. These tests confirmed the system's functionality and responsiveness (see figure 10). Real-world testing on the physical wheelchair was not feasible due to time constraints. Future plans include usability evaluations with actual users to collect feedback on accessibility, functionality, and overall user experience, guiding further refinements for practical use.

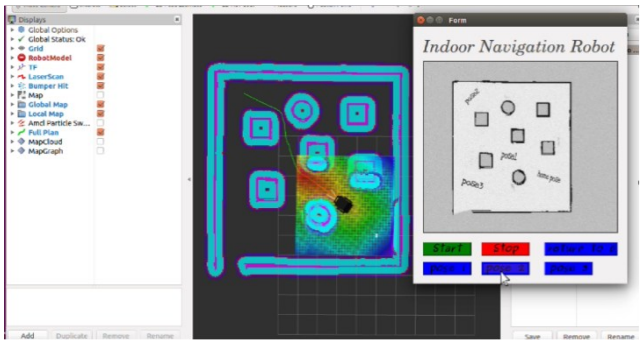

Fig. 10. Testing GUI in Gazebo with Pose 2

VI. Results and Discussion

This section provides an overview of the outcomes achieved in the self-guided navigation mode of our autonomous wheelchair, emphasizing the methodologies used and the obstacles successfully addressed. We used Simultaneous Localization and Mapping (SLAM) approaches, namely Gmapping and RGB-D SLAM (rtabmap), to construct an environmental map by integrating a Microsoft Kinect 360 sensor with the Robot Operating System (ROS). Our methodology allowed the wheelchair to maneuver to any specified location within the mapped surroundings while avoiding dynamic impediments.

Figure 11 depicts the altered wheelchair used in our study, which incorporates all hardware components such as motors, sensors, and the control system. An important inclusion is the custom-made steel camera base that firmly supports the Kinect sensor at an ideal height for efficient mapping of the surroundings. The Kinect + Base Assembly is essential for the capture of visual data, facilitating SLAM algorithms for the purposes of mapping and localization. The design preserves the original wheelchair frame while incorporating these customized improvements, guaranteeing reliable movement in indoor settings.

Fig. 11. Final Autonomous Wheelchair Design

A. Wheelchair Driver Implementation

The wheelchair driver module is in charge of low-level control and sensor integration, necessary for navigation in an independent way. This module will encompass the following components: the Arduino embedded code, processing of sensor data, and ROS nodes for handling the communication between the hardware and the navigation stack.

The user-created Python driver node integrates the Arduino with ROS. The Arduino node reads serial data from the Arduino and converts it into ROS topics for use by the navigation stack. It encapsulates IMU orientation data, wheel encoder values, and motor speed commands. It subscribes to topics like **/cmd_vel** and publishes motor speed targets using topics like **lwheel_vtarget** and **rwheel_vtarget**. A PID controller can provide stable and accurate speed control by gaining proportionately for integral, integration, and differentiation.

Additional ROS nodes like **depth_image_to_laser_scan** (for converting Kinect depth data to laser scans) and diff_tf.py (for broadcasting odometry transformations) support the integration of sensor data into the ROS environment. This ensures that the wheelchair has accurate positional feedback and can navigate smoothly. The **keyboard_teleop.py** node enables manual control for tasks like building a static map, which is essential before fully autonomous navigation.

B. Mapping and Localization Approaches

In this project, we explored two main approaches for mapping and localization: Gmapping SLAM and RGB-D SLAM (rtabmap). The primary goal was to build a reliable environment map and accurately localize the autonomous wheelchair within that map.

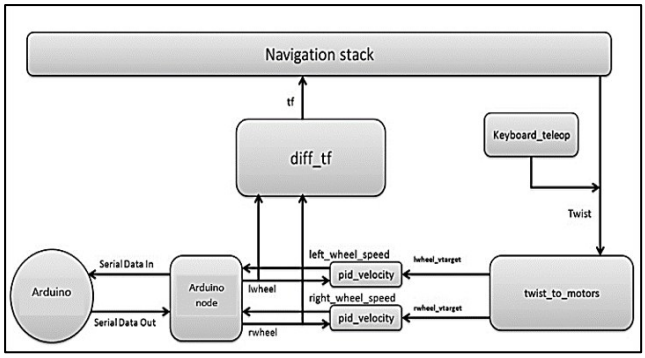

Fig. 12. Block diagram of "Wheelchair driver" module showing the ROS nodes [**16**]

1) Gmapping SLAM: The Gmapping algorithm, known for creating 2D occupancy grid maps using laser scan data and odometry, was initially tested with our Kinect 360 camera. Figure 13 illustrates the environment map created using the Gmapping algorithm after approximately 30 minutes of manual teleoperation. The map was generated by simulating laser scans with Kinect data and demonstrates the system's capability to produce a 2D occupancy grid for path planning. However, despite extensive parameter tuning (e.g., linear and angular updates, map intervals), the resulting maps lacked the required precision. Obstacle boundaries were often inaccurate, and overall map quality was suboptimal, leading us to consider alternative methods.

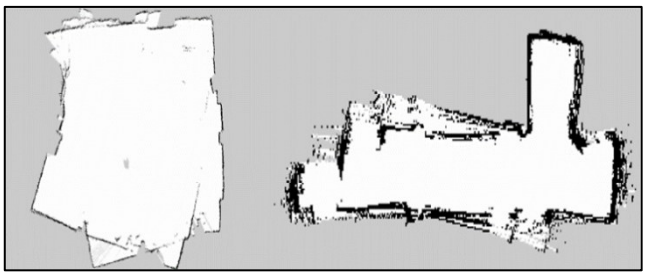

Fig. 13. Generated Grid-Maps by Gmapping SLAM Algorithm

2) RGB-D SLAM (rtabmap): Shifting to the RTAB-Map algorithm significantly enhanced mapping accuracy by utilizing synchronized RGB-D data to generate detailed 2D and 3D maps, proving more suitable for our Kinect-based system. The algorithm performed particularly well in environments with ample visual features, resulting in clear and consistent maps. Figure 15 shows the real-time 3D mapping generated by the RTAB-Map algorithm. The RGB-D data from the Kinect sensor provides a detailed 3D reconstruction, enhancing localization accuracy. The left panel displays the live camera feed, and the right panel visualizes the mapped environment in RViz.

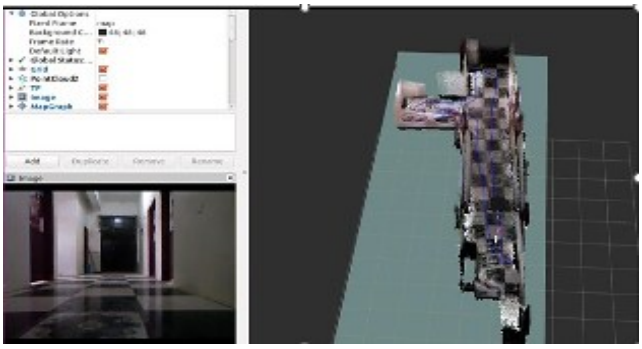

Fig. 14. Real-time 3D Mapping by RTAB-Map Algorithm

Additionally, Figure 15 shows 2D grid maps generated by RTAB-Map, which display better-defined boundaries and fewer inaccuracies compared to previous SLAM algorithms. These improvements were critical for more reliable autonomous navigation, especially in environments where traditional SLAM algorithms had difficulties with feature scarcity or sensor data synchronization issues. However, in environments with minimal visual features, such as plain walls, rtabmap struggled with localization due to insufficient data for object matching.

Fig. 15. Generated 2D Grid-Maps by RTAB-Map Algorithm

3) Mapping Accuracy: To evaluate the mapping accuracy, we compared the Gmapping and RTAB-Map algorithms. Table 1 summarizes key metrics such as resolution and accuracy. RTAB-Map demonstrated a significant improvement in map resolution, achieving 50 pixels/m² compared to 20 pixels/m² with Gmapping. Similarly, RTAB-Map achieved a 90% overlap accuracy, outperforming Gmapping's 75%. These results are visualized in Figure 13 and 15, which illustrates the maps generated by both algorithms, highlighting the superior precision and clarity of RTAB-Map maps.

4) AMCL for Improved Localization: The AMCL (Adaptive Monte Carlo Localization) algorithm uses a particle filter to estimate the robot's pose within a pre-built static map. This reliable localization performance was achieved with AMCL in environments with sparse features (see figure 15). Unlike RTAB-Map, which struggled with limited visual cues, AMCL proved more consistent and accurate by relying on probabilistic methods, enabling the wheelchair to maintain precise localization even in visually

simple or repetitive settings. This improvement was crucial for stable autonomous navigation within mapped indoor environments.

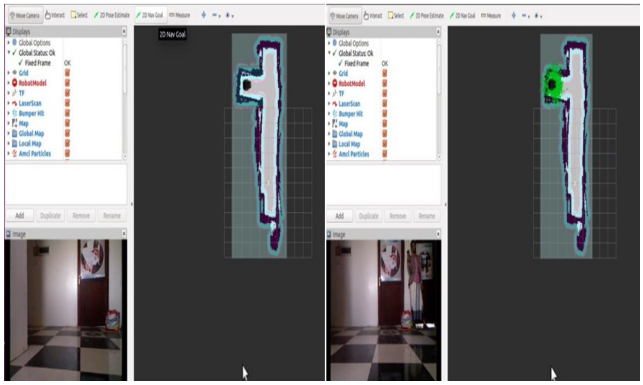

Fig. 16. Wheelchair localization within pre-built static map using AMCL

C. *Autonomous Navigation in a Pre-Built Map*

This subsection covers the implementation of autonomous navigation using ROS's *move_base* node. The system integrates odometry data, sensor inputs, and pre-built maps to allow the wheelchair to move autonomously from one point to another. The global planner uses pre-existing maps for path planning, while the local planner dynamically adjusts to avoid obstacles in real-time.

Global and local costmaps ensure the system accurately identifies free space and obstacles, while recovery behaviors (like rotating or clearing costmaps) help when the wheelchair encounters unexpected issues. The system localizes with AMCL, plans the route, and continuously adjusts until the goal is reached.

Key parameters were carefully tuned to optimize navigation, including [**18**]:

- Costmap Resolutions and Update Frequencies: Ensured smooth navigation by balancing map resolution with real-time updates.
- Goal Tolerances: Fine-tuning yaw and xy tolerances enabled accurate goal-reaching.
- Planner Frequencies and Recovery Behaviors: Adjustments to planner frequencies and recovery settings (like oscillation timeouts and rotation recovery) enhanced path reliability.

The results demonstrate that the system successfully navigated autonomously, consistently reaching target goals within the environment. The move_base node executed planned paths efficiently while adapting to real-time conditions. AMCL with 500 particles and a 0.1 resampling threshold provided precise localization, and recovery behaviors, including clearing costmaps every 15 seconds and rotation recovery after 10 seconds of oscillation, improved navigation reliability, particularly in dynamic obstacle scenarios.

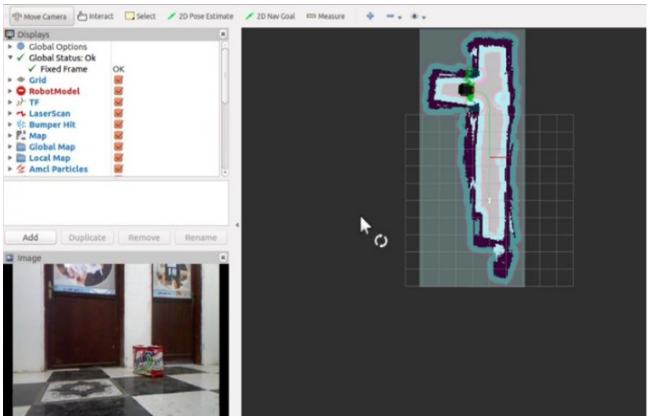

Fig. 17. Wheelchair generating a path toward the target goal

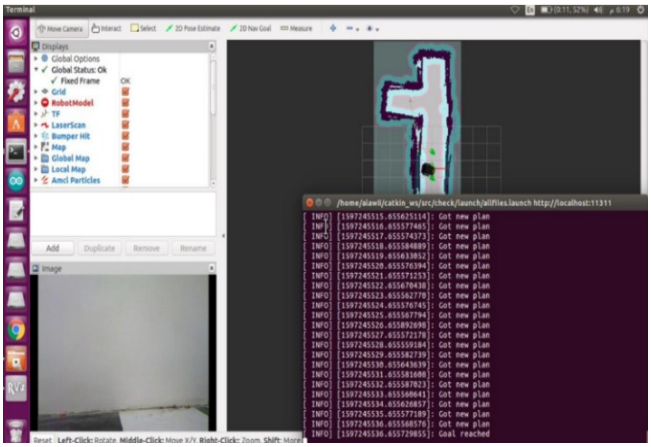

Fig. 18. Wheelchair successfully reaching the goal position and orientation.

Costmaps with a resolution of 0.05 m/cell and a 5 Hz update rate effectively mapped obstacles and free spaces. The global planner, using the A* algorithm, calculated efficient routes, while the local planner, based on the Dynamic Window Approach (DWA), dynamically adjusted paths. Planner frequencies were set at 10 Hz, and goal tolerances of 0.1 m and 5° ensured accurate goal-reaching.

Figures 17 and 18 show the wheelchair's localization and path planning process, as well as the successful achievement of the goal position and orientation. The first image illustrates the system's localization within the pre-built map using the AMCL algorithm. The global and local planners are activated to plan the path to the goal. The second one demonstrates that the wheelchair successfully reached the goal, confirming accurate position and orientation.

The system's navigation performance was evaluated in both static and dynamic environments, achieving a 95% success rate in static settings and a robust 85% in dynamic scenarios with moving obstacles. RTAB-Map enhanced efficiency by reducing the average time to reach goals from 15 seconds (Gmapping) to 10 seconds, while also optimizing resource utilization, lowering CPU usage from 80% to 60%.

TABLE 2 - KEY CHALLENGES AND CORRECTIVE ACTIONS

Challenge	DESCRIPTION	Corrective Action
Mapping Accuracy	Gmapping algorithm produced inaccurate maps due to sensor synchronization issues.	Switched to RGB-D SLAM (rtabmap) for better synchronization and improved map detail.
Localization in Sparse Environments	Localization failed in environments with minimal detectable features like plain walls.	Integrated AMCL for robust particle filter-based localization in such environments.
Path Planning & Obstacle Avoidance	The robot exhibited unstable navigation with default move_base parameters.	Fine-tuned parameters like costmap settings, tolerances, and recovery behaviors.
Hardware Integration & Data Synchronization	Inconsistent data transmission between Arduino and ROS led to communication lags.	Optimized Arduino code and used rosserial to ensure efficient data flow.
Camera Base Position for Accurate Mapping	A poorly placed camera missed low obstacles during mapping, leading to inaccurate maps.	Designed a stable steel base to position the camera at an optimal height of 22 cm above ground.
MPU 6050 Orientation Data Issues	Non-normalized quaternion values from the MPU caused issues in ROS odometry calculations.	Modified the MPU library to send Euler angles instead, allowing ROS to compute and normalize the quaternion.
Qt Library Limitation for Voice Recognition	Qt GUI lacked support for voice recognition, impacting accessibility.	Transitioned to Tkinter, which supports integration with the SpeechRecognition library.

D. Challenges and Corrective Actions

During the development and testing of the autonomous wheelchair, several challenges arose, ranging from hardware limitations to software integration issues. This subsection summarizes the major challenges encountered and the corresponding corrective actions taken to address them.
The next table summarizes key challenges and solutions in our autonomous wheelchair project. Issues like inaccurate mapping, poor localization, unstable navigation, and hardware communication were resolved by switching algorithms, fine-tuning parameters, optimizing code, and improving hardware setup. Adjustments were also made for better voice recognition and overheating control.

VII. CONCLUSION

This work solves the critical need for low-cost and autonomous mobility solutions for people with severe disabilities by providing an affordable wheelchair system based on ROS. In fact, using Gmapping SLAM have shown major challenges with sensor synchronization and mapping accuracy; these are, therefore, counterproductive to good navigation. As a result, we implemented a hybrid approach that employs RTAB-Map for mapping and AMCL for localization has significantly improved the precision of the map, localization accuracy, and reliable indoor navigation. Integration with the mapping system, adaptive path planning, and user-friendly controls in one system is promising for enhancing accessibility in diverse real-world scenarios.

The hybrid RTAB-Map and AMCL approach has practical implications for users, enabling reliable and seamless navigation in dynamic indoor environments. Its affordability, achieved through the use of cost-effective sensors like Kinect, makes it accessible to underserved populations, especially in low-resource settings. Additionally, the system's scalability and adaptability allow it to meet various environmental and user-specific requirements, addressing mobility challenges outlined in the Problem Statement.

Finally, this prototype can be enhanced by replacing the laptop with a Raspberry Pi and a slim screen. This will make the overall weight of the system less, improving its portability and making it more convenient to use. Usability of the system may be improved with richer GUI options and offline voice recognition. The further development of the outdoor navigation capabilities of the system and the refinement of the control algorithms in various environments will further expand the applicability of the wheelchair, enabling it to be more versatile and adaptive in real situations.

ACKNOWLEDGMENTS

We are really grateful to all who contributed to the successful completion of this project. We would like to thank our supervisor, ***Dr. Hatem Al-Dois***, for actually helping and supporting us. We acknowledge also the graduation members of the team: ***Eyad Al-Junied, Awad Al-Faieq, Moath Jubarah***, ***and Yousef Saleem*** for the contributions throughout this project.

REFERENCES

[1] B. Woods and N. Watson, "history of the wheelchair," *Encyclopedia Britannica*, 11 Feb. 2015. [Online]. Available: https://www.britannica.com/technology/history-of-the-wheelchair. [Accessed: 01-Aug-2024].

[2] M. F. Ansari, D. R. Edla, S. Dodia, and V. Kuppili, "Brain-Computer Interface for wheelchair control operations: An approach based on Fast Fourier Transform and On-Line Sequential Extreme Learning Machine," *Clinical Epidemiology and Global Health*, vol. 7, no. 3, pp. 391–397, Jul. 2019, doi: 10.1016/j.cegh.2018.10.007.

[3] S. Abdallaoui, H. Ikaouassen, A. Kribèche, A. Chaibet, and E.-H. Aglzim, "Advancing autonomous vehicle control systems: An in-depth overview of decision-making and manoeuvre execution state of the art," *The Journal of Engineering*, vol. 2023, no. 23, pp. 1-12, Nov. 2023, doi: 10.1049/tje2.12333.

[4] Y. Zhang and S. Xu, "ROS Based Voice-Control Navigation of Intelligent Wheelchair," *Applied Mechanics and Materials*., vol. 733, pp. 740–744, Feb. 2015, doi: 10.4028/www.scientific.net/AMM.733.740.

[5] Q. Xu, J. Zhao, C. Zhang and F. He, "Design and implementation of an ROS based autonomous navigation system," *2015 IEEE International Conference on Mechatronics and Automation (ICMA)*, Beijing, China, 2015, pp. 2220-2225, doi: 10.1109/ICMA.2015.7237831.

[6] Y. Nasri, V. Vauchey, R. Khemmar, N. Ragot, K. Sirlantzis, and J.-Y. Ertaud, "ROS-based autonomous navigation wheelchair using omnidirectional sensor," *International Journal of Computer Applications*, vol. 133, no. 6, pp. 12–17, Jan. 2016. doi: 10.5120/ijca2016907533.

[7] M. Marrón, J. C. García, Ł. Księżak, P. Del Moral, D. Pinedo and J. León, "Open platform and open software for an Intelligent Wheelchair with autonomous navigation using sensor fusion," *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, Florence, Italy, 2016, pp. 5929-5934, doi: 10.1109/IECON.2016.7793775

[8] H. Grewal, A. Matthews, R. Tea and K. George, "LIDAR-based autonomous wheelchair," *2017 IEEE Sensors Applications Symposium (SAS)*, Glassboro, NJ, USA, 2017, pp. 1-6, doi: 10.1109/SAS.2017.7894082.

[9] S. Gatesichapakorn, J. Takamatsu and M. Ruchanurucks, "ROS based Autonomous Mobile Robot Navigation using 2D LiDAR and RGB-D Camera," *2019 First International Symposium on Instrumentation, Control, Artificial Intelligence, and Robotics (ICA-SYMP)*, Bangkok, Thailand, 2019, pp. 151-154, doi: 10.1109/ICA-SYMP.2019.8645984

[10] J. Sun, X. Yu, X. Cao, X. Kong, P. Gao and H. Luo, "SLAM Based Indoor Autonomous Navigation System For Electric Wheelchair," *2022 7th International Conference on Automation, Control and Robotics Engineering (CACRE)*, Xi'an, China, 2022, pp. 269-274, doi: 10.1109/CACRE54574.2022.9834195.

[11] M. T. Islam, I. R. Hameem, S. Saha, M. J. R. Chowdhury and M. E. Deowan, "A Simulation of a Robot Operating System Based Autonomous Wheelchair with Web Based HMI Using Rosbridge," *2023 3rd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*, Dhaka, Bangladesh, 2023, pp. 175-180, doi: 10.1109/ICREST57604.2023.10070046.

[12] K. Kappel and P. R. Ferreira, "Towards Comfortable and Socially Acceptable Navigation in Autonomous Motorized Wheelchairs," *2023 Latin American Robotics Symposium (LARS), 2023 Brazilian Symposium on Robotics (SBR), and 2023 Workshop on Robotics in Education (WRE)*, Salvador, Brazil, 2023, pp. 319-324, doi: 10.1109/LARS/SBR/WRE59448.2023.10332989.

[13] M. Quigley, B. Gerkey, and W. D. Smart, *Programming Robots with ROS: A Practical Introduction to the Robot Operating System*. Sebastopol, CA, USA: O'Reilly Media, Inc., 2015.

[14] L. Joseph and J. Cacace, *Mastering ROS for Robotics Programming*. 2nd Edition, Packt Publishing Ltd, February, 2018.

[15] *TSS300 Owner's Manual*, Marc's Mobility. Rev B. December 2009. [Online]. Available: https://www.manualslib.com/manual/694478/Pride-Mobility-Tss300.html [Accessed: 06-Aug-2024].

[16] F. Al-Fahaidy, H. Al-Dois, F. A. K. Al-Fu-Haidy, and E. A. Qasabah, "Design and Implementation of an Eye-Controlled Self-Driving Wheelchair," *ResearchGate*, Nov. 2021. [Online]. Available: https://www.researchgate.net/publication/356587774_Design_and_Implementation_of_an_Eye-Controlled_Self-Driving_Wheelchair. [Accessed: 14-Aug-2024].

[17] P. Chahal, "Robot operating systems (ros) overview & (1)," *SlideShare*, May 3, 2011. [Online]. Available: https://www.slideshare.net/robot-operating-systems-ros-overview/17822782 [Accessed: 08-Aug-2024].

[18] Yokozuka, M., Hashimoto, N., Tomita, K., & Matsumoto, O. Development of Autonomous Wheelchair for Indoor and Outdoor Traveling. *Internet of Things Summit*. 2014.