Boston University
Electrical & Computer Engineering
EC464 Capstone Senior Design Project

User's Manual

Submitted to

Thomas Little
8 St. Mary's Street
Room 426
(617) 353-9877
tdcl@bu.edu

by

Team 30
SmoothOperator

Team Members

Celine Chen cchen@bu.edu
Eric Chen chene@bu.edu
Jacob Chin jacobc@bu.edu
Christian So cbso@bu.edu
Nicholas Nguyen nqnguyen@bu.edu

:

Submitted: 4/18/2025

SmoothOperator User Manual

Table of Contents

Executive Summary

Post-COVID staffing shortages in the hospitality industry have increased mobility challenges for travelers with physical impairments, particularly regarding luggage transportation. With over 61 million Americans (26% of adults) living with disabilities, there is a significant need for assistive solutions. The hospitality sector has experienced a 38% reduction in workforce since 2019, with airport ground staff decreasing by 25%, further limiting available assistance.

SmoothOperator addresses this need through an interactive robotic assistant designed to autonomously and safely transport luggage in consumer environments. We developed a complete, full-scale, end-to-end robotic solution that seamlessly integrates multiple

subsystems. This involved manufacturing custom components and assembling a sophisticated network of sensors, actuators, algorithms, and communication protocols. These systems allow SmoothOperator to navigate using either terminal destinations or direct user input via an onboard touchscreen interface. To ensure safe and reliable operation, we implemented multiple redundancy strategies and layered safety mechanisms, including dynamic obstacle avoidance that prioritizes human interaction. SmoothOperator is designed to enhance accessibility and convenience in the hospitality sector, improving the quality of life and providing greater confidence for individuals with mobility impairments when traveling.

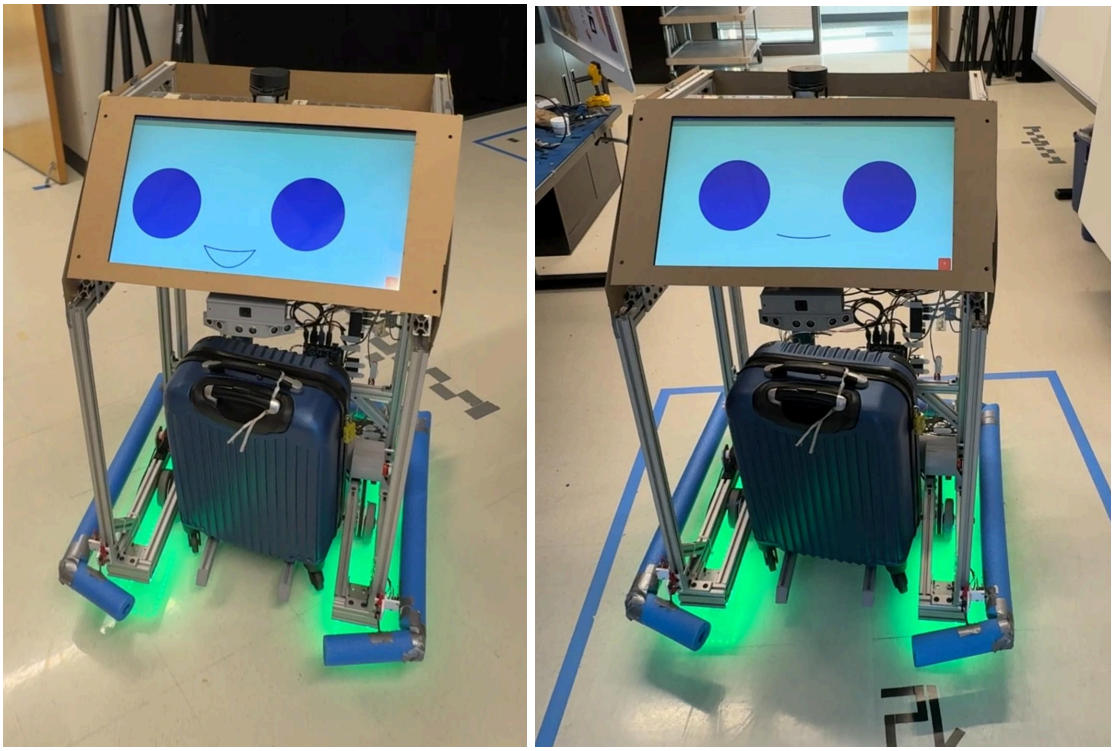

Figure 1: *SmoothOperator Robot happily and autonomously transporting Luggage.*

1 Introduction

This system is a fully autonomous and teleoperated robotic assistant, designed from the ground up to carry luggage while providing seamless, interactive service to users.

Built around the ROS (Robot Operating System) architecture, SmoothOperator integrates a custom-built navigation stack leveraging SLAM (Simultaneous Localization and Mapping), obstacle avoidance algorithms, and a user interface that is both onboard and remotely accessible. The entire system is powered by a mobile GPU-equipped computing platform (NVIDIA Jetson Nano), and features robust sensor integration, including a 2D LiDAR, ultrasonic rangefinders, wheel encoders, and an IMU, all synchronized via ROS nodes and serial communication protocols.

SmoothOperator offers an intuitive and highly responsive user experience. The robot's 21.5" touchscreen face features interactive visual and audio feedback, allowing users to load luggage, navigate menus, and activate autonomous delivery with ease. Once a boarding pass QR code is scanned, the robot automatically navigates to the designated gate, with a maximum speed of 1.2 m/s—matching the average walking pace.

From a hardware standpoint, SmoothOperator’s chassis can support up to 80 lbs, exceeding its 50 lb design requirement by 60%. Its dual-battery power system (12V, 12Ah each) ensures untethered operation, while a compound gear drivetrain and 10:1 reduction ratio provide necessary torque for stability and movement over long distances.

For manual control, both the onboard interface and a secure mobile app (React Native-based) enable real-time robot manipulation, with visual LED indicators (green for motion, red for halt) enhancing safety and clarity. In-app access requires a four-digit pairing code for security, ensuring only authorized users control the robot at any given time.

The robot is engineered with multiple safety redundancies. In addition to software-based path planning, a hardware-level override system—comprising ultrasonic sensors and limit switches—ensures that any physical proximity breach will immediately halt movement, regardless of higher-level navigation commands. A radio-frequency emergency stop (e-stop) system can shut down all actuators from up to 10 feet away.

To mitigate risks associated with limited vertical perception, especially blind spots under the 2D LiDAR sensor, the team implemented foam bumper-based collision detection. Though a 3D LiDAR would have offered more precise environmental sensing, the cost-prohibitive nature of such hardware was acknowledged and addressed with these more accessible fallback mechanisms.

Power distribution is safeguarded via a fused breaker system, preventing electrical overloads from cascading through the system. Each subsystem is electrically isolated in case of fault, and diagnostic telemetry is integrated to support quick troubleshooting and maintenance.

2 System Overview and Installation

2.1 Overview block diagram

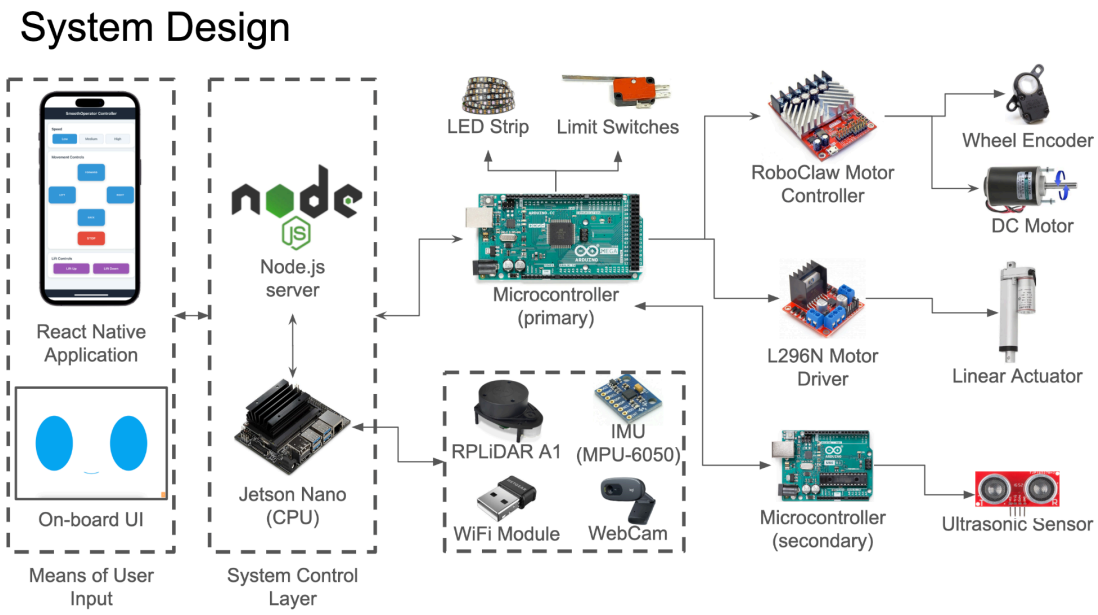

Figure 2: *Overall system design of dataflow for SmoothOperator.*

2.2 User Interface

The user interface encompasses the onboard touch screen and mobile application. Human factors, such as control pad, coordinated colors, etc., were considered to make the user interface intuitive and easy to use.

2.2.1 Onboard

The onboard touch screen displays the robot's face with blue eyes and a small smile, enhancing the hospitality of the robot. The face would react to the movements that the robot will be making in giving real-time feedback on the direction in which it's moving in. Additionally, any audio that is spoken by SmoothOperator, such as "Watchout! I'm coming through" or "Hi! I'm SmoothOperator," syncs with the mouth on the face.

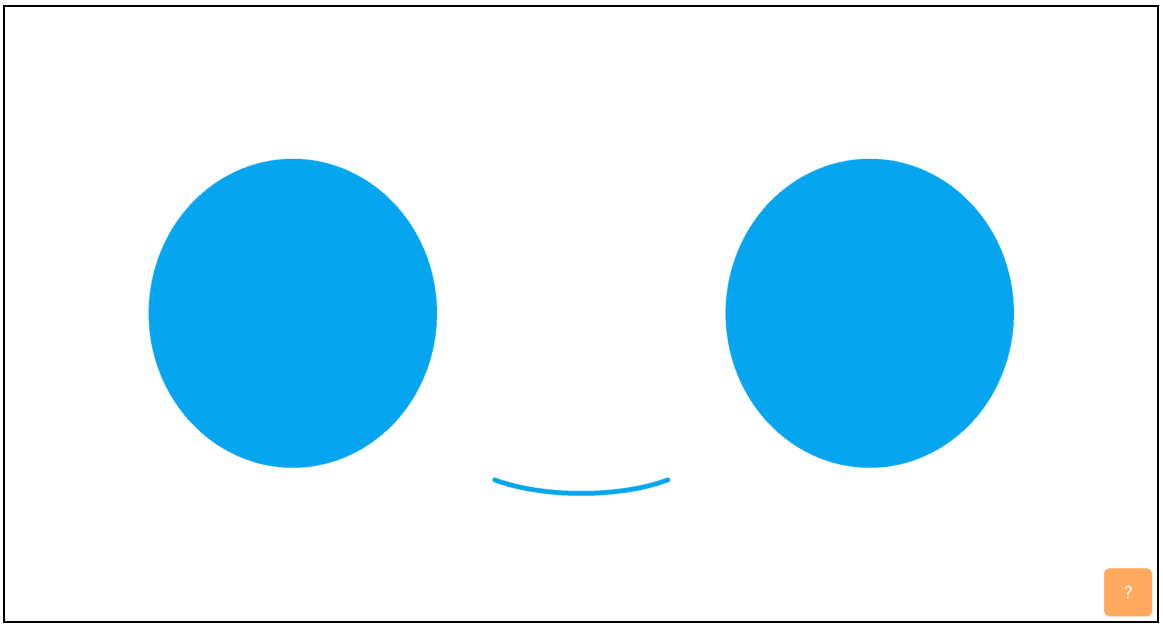

Figure 3: *The SmoothOperator face conveys a friendly and approachable demeanor through minimalistic visual elements.*

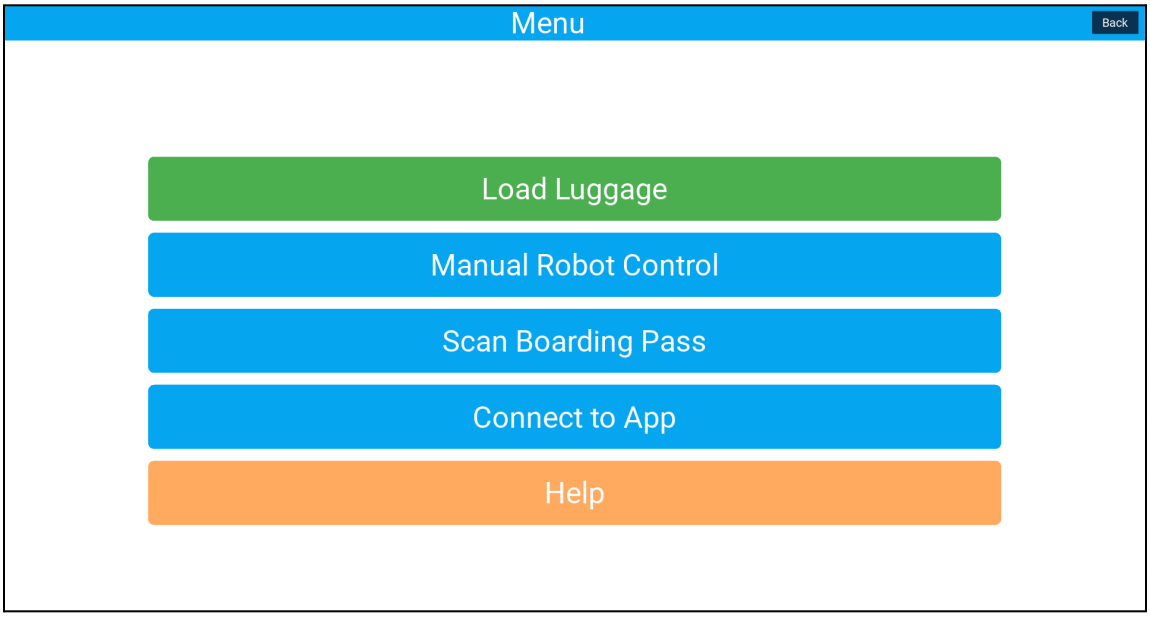

Figure 4: *The menu screen allows the user to navigate the onboard screen.*

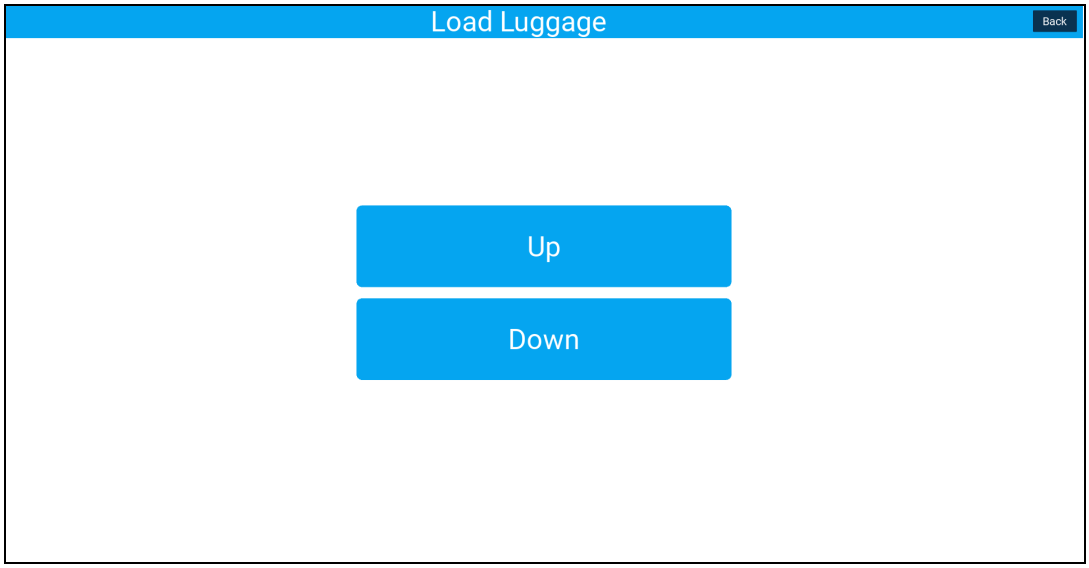

Figure 5: *The load luggage screen allows the user to raise and lower the lifting mechanism.*

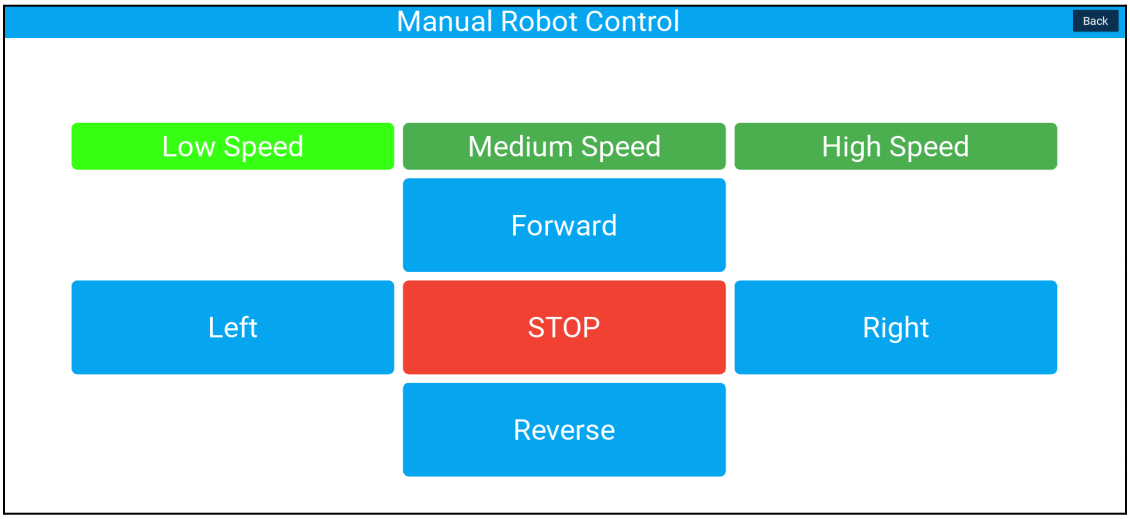

Figure 6: *The manual control screen allows the user to move SmoothOperator.*

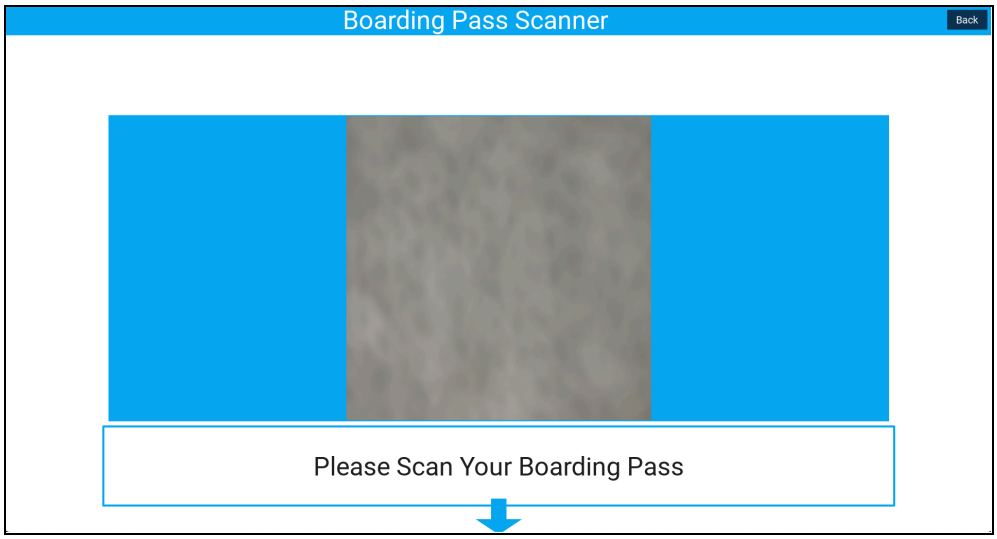

Figure 7: *The boarding pass scan screen allows the user to scan their boarding pass.*

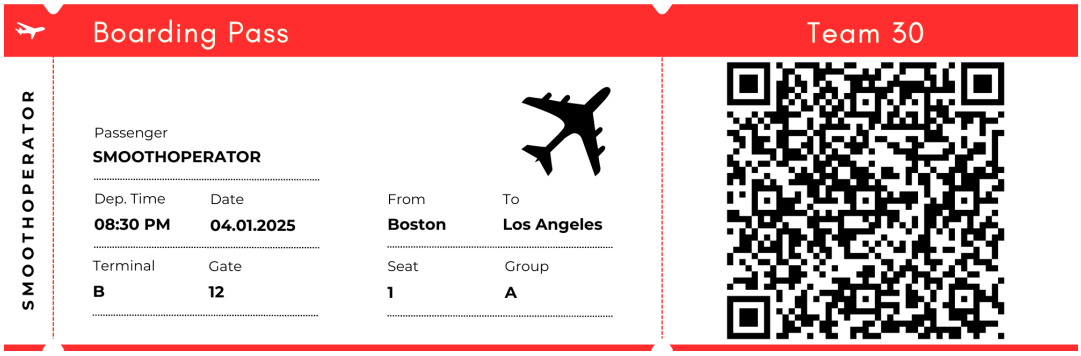

Figure 8: *Example of QR code boarding pass with QR code we created.*

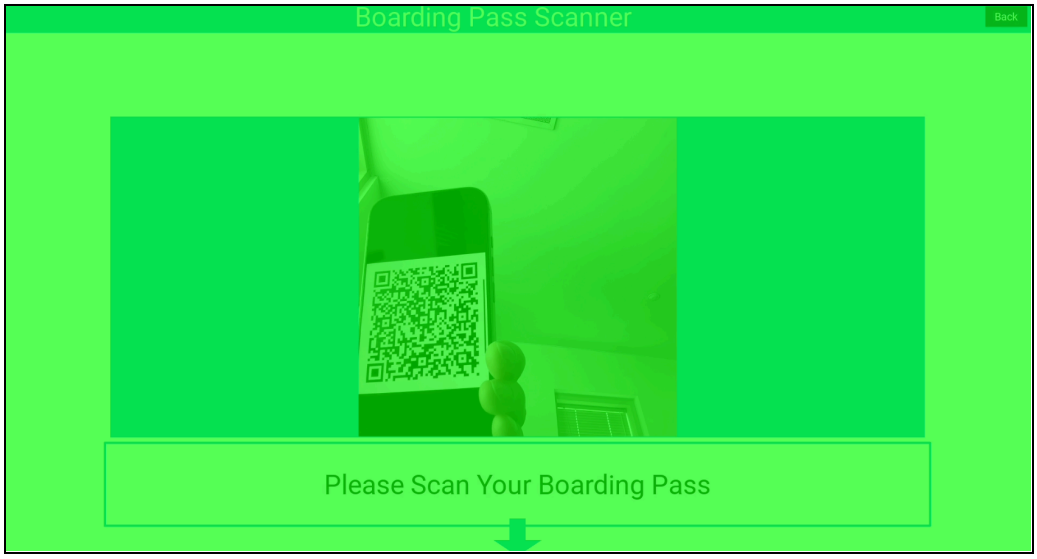

Figure 9: *The green flashing gives the user an obvious indication of a successful scan.*

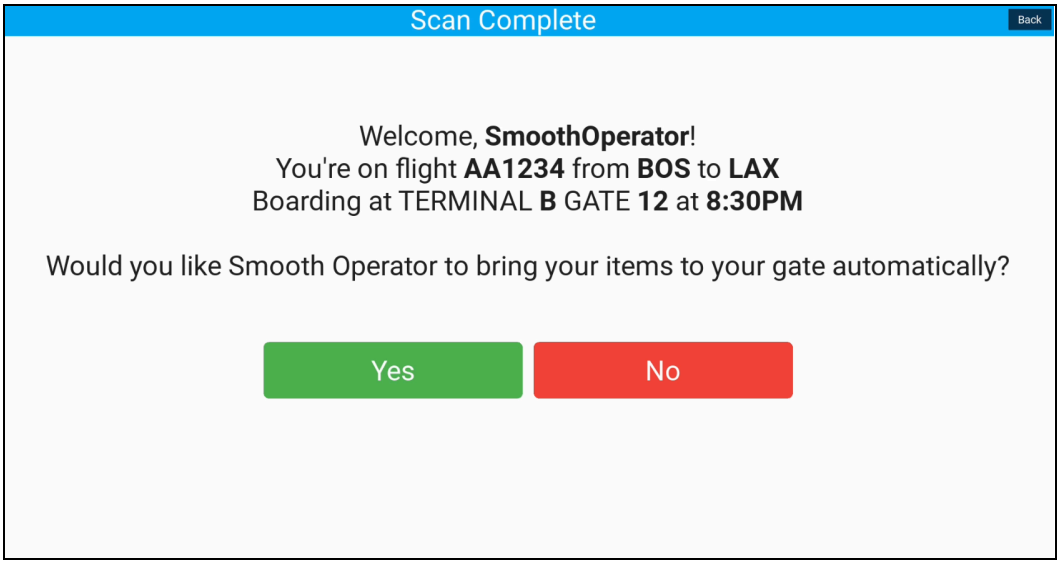

Figure 10: *The confirmation screen ensures that the user wants SmoothOperator to navigate autonomously.*

2.2.2 Phone Application

For security, the mobile application must connect to the correct SmoothOperator unit. Upon starting the mobile application, the user will be prompted to input the correct passcode. The passcode can be found on the onboard touch screen through the 'Connect to App' menu.

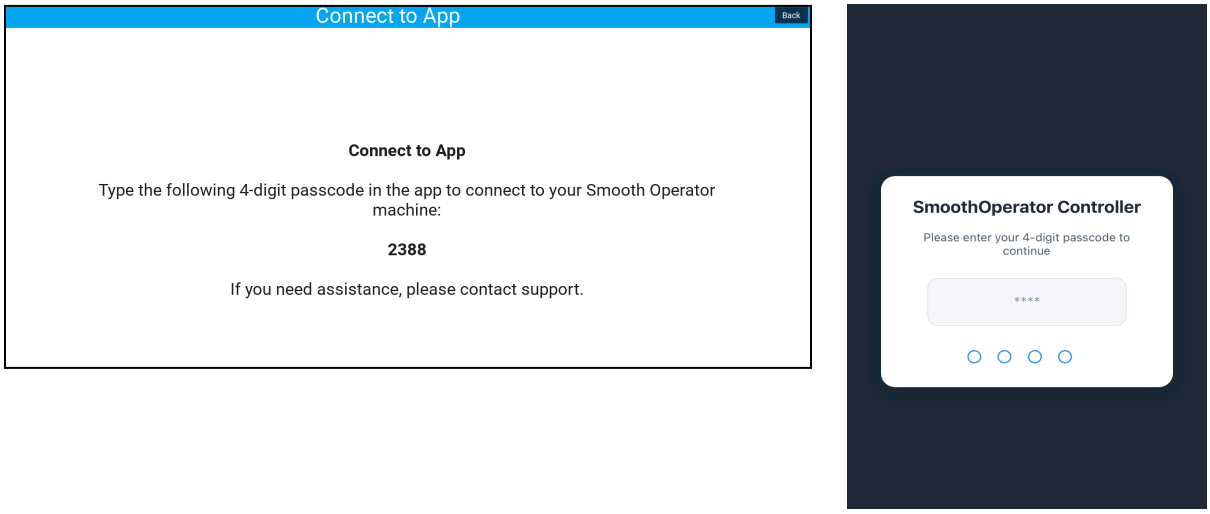

Figure 11: *The onboard screen (left) displays a randomly generated unique passcode, and the phone app (right) displays the locked screen awaiting a correct passcode input.*

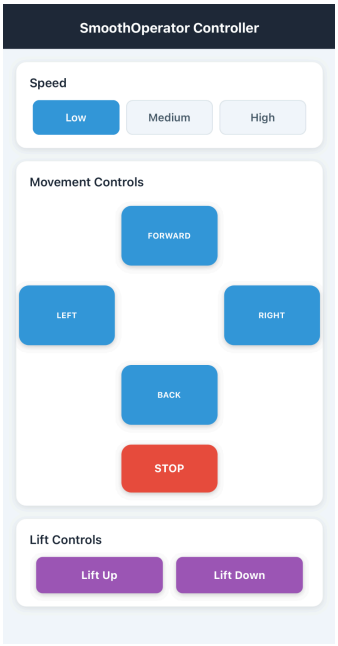

Figure 12: *The phone app control screen allows the user to operate the robot remotely.*

2.3 Physical description

The figures below illustrate the physical description of all the hardware components that make up SmoothOperator. The figures show from the overall construction down to the electronics suite and the inner gear boxes that are being used to make the robot move.

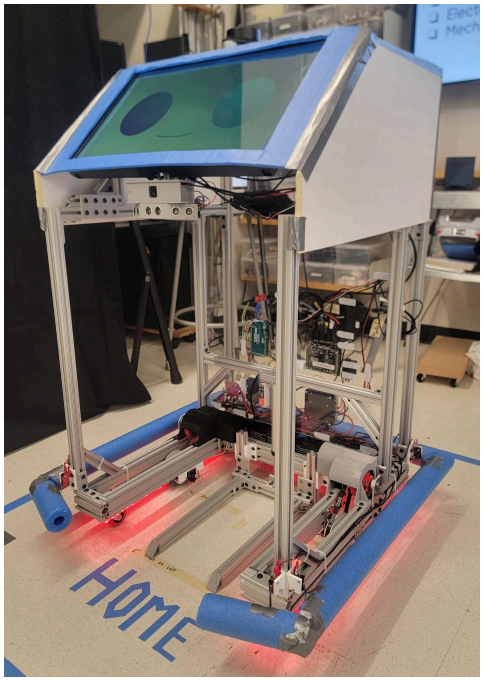

Figure 13: *The fully built robot. Luggage is loaded in the center of the robot.*

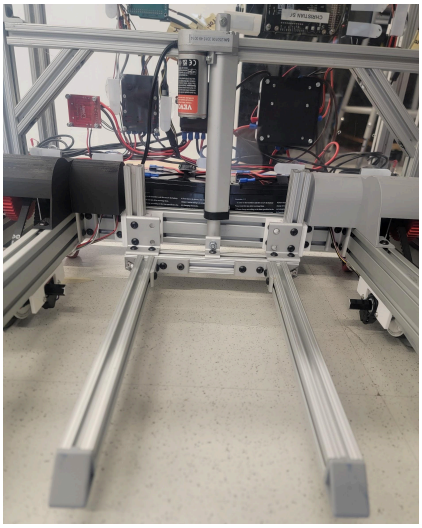

Figure 14: *The luggage lifting mechanism.*

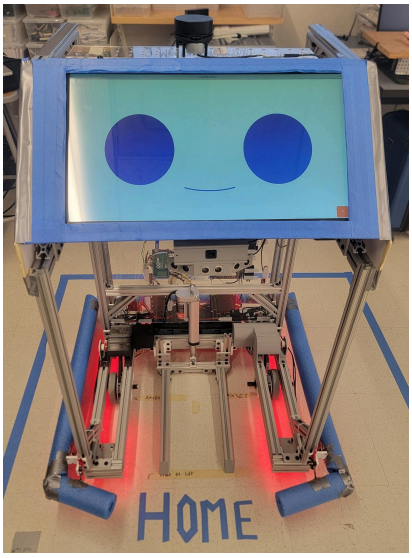

Figure 15: *The facial UI display on the robot's touch screen monitor.*

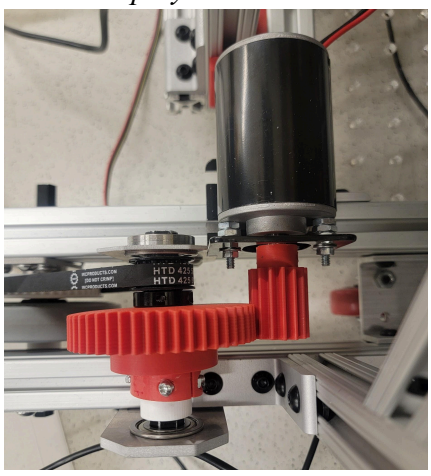

Figure 16: *The drivetrain design uses a 10:1 compound gear train consisting of gears, pulleys, and a HTD timing belt.*

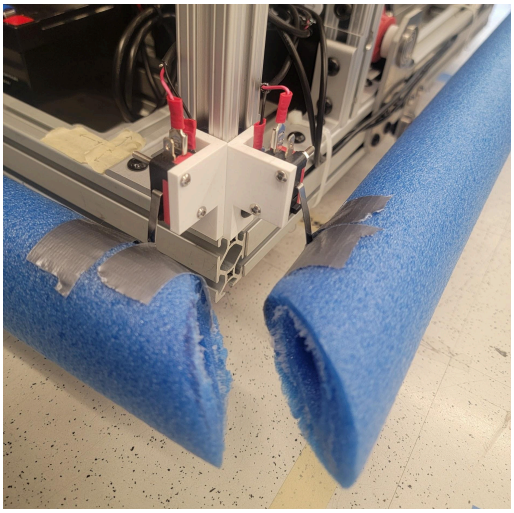

Figure 17: *The robot features a floating bumper for additional collision detection. Motion halts when the bumper activates attached limit switches.*

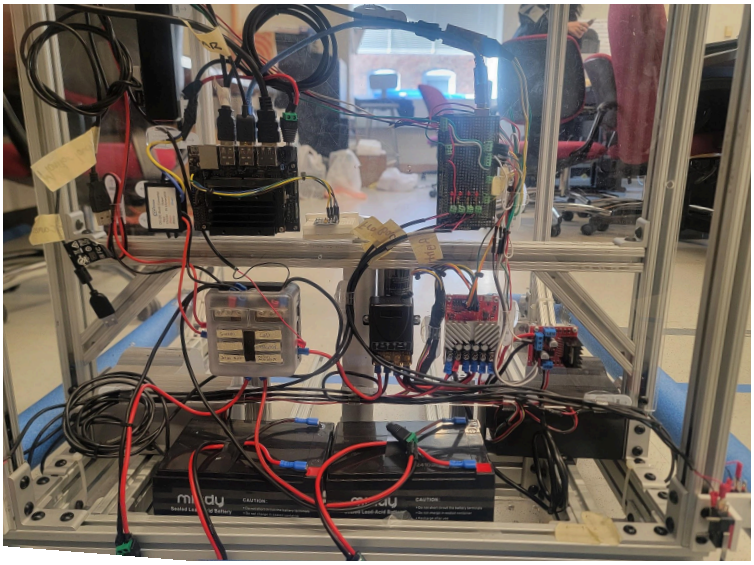

Figure 18: *The robot's electronics suite.*

2.4 Navigation

The ROS (Robot Operating System) navigation stack is composed of multiple nodes working together to enable autonomous navigation, including tasks such as global path planning, real-time obstacle avoidance, and safe motion control. Key components include nodes for reading sensor data, like the LiDAR for environmental perception and wheel encoders/IMU for motion estimation. A dedicated node performs differential drive kinematics to update the robot's odometry, determining its orientation and position over time. Prior to navigation, a map of the environment is created by a mapping node. During operation, a localization node continuously estimates the robot's position within this map. This helps with relating the coordinate frame of the odometry system to the reference frame of the environment map. The system also includes transform (TF) nodes to maintain coordinate relationships between the robot and its sensors, and costmap nodes that generate local and global costmaps to mark navigable and hazardous regions.

Navigation nodes compute both global and local plans, while recovery behavior nodes define fallback actions when the robot encounters obstacles or gets stuck. Another node sends velocity commands from the navigation stack to the arduino. Additionally, an interface node handles communication with the user interface, allowing destination inputs to be received and processed by the navigation stack.

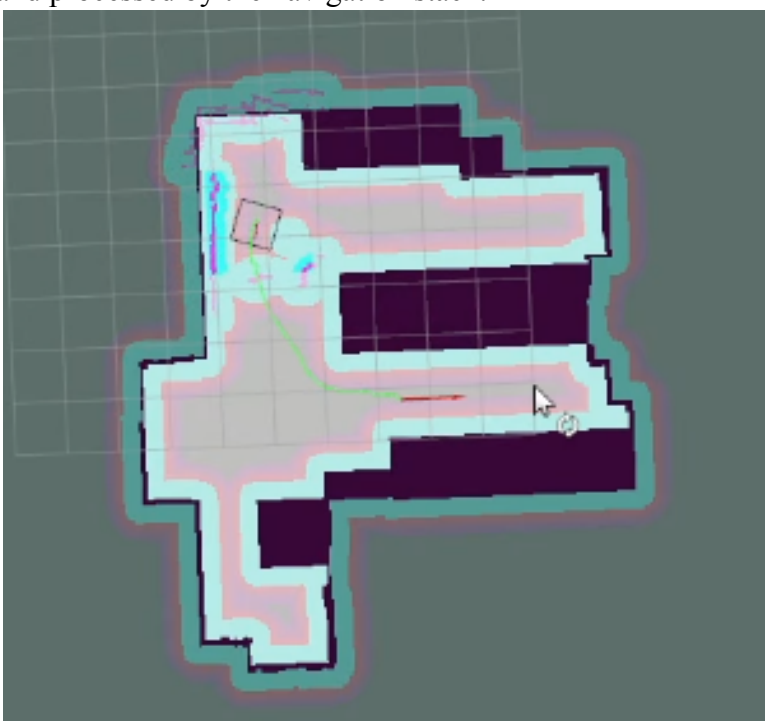

Figure 19: *The navigation nodes computed global and local plans as indicated by the green line.*

2.4.1 Navigation Stack Setup

The computer must have ROS Melodic installed (we are using a Nvidia Jetson Nano). If the user is using a Linux or Ubuntu-based machine, then they can follow the instructions on the ROS wiki: https://wiki.ros.org/melodic/Installation/Ubuntu.

From the "~" directory of the Jetson Nano, the user can run "./runTerminals.sh". This script automates the process of opening several terminals and running the necessary commands in each of them to get the entire navigation stack running. Due to the interaction of using a script to open terminals, there are two terminals that require rerunning commands. These terminals are named "faceUI" and "node-server". In the "faceUI" terminal, please run "python3 faceUI.py". In the "node-server" terminal, please run "node FinalDemoNode.js".

2.5 Installation, setup, and support

Hardware

Plug in the two male DC barrel jacks from the two batteries into the two female terminals connected to the robot. Once connected, the system should power on. The Jetson Nano, Arduino Mega, Monitor, and FTDI Connector should have indicator lights that turn on. Following that, the monitor screen displays an output. If the system does not start, verify

that the batteries are charged with a multimeter. Each battery should be ~12.6V, indicating sufficient power.

Software
For navigation, install ROS Melodic and use Python 2. Using other versions may lead to package incompatibilities within the ROS nodes.

For the mobile user interface, download the Expo Go app on your mobile device. Use Node.js version 16 to ensure compatibility with the current Expo SDK and React Native. For the onboard user interface, install Python 3.13.3 and use it for running all Kivy packages and the Pi Camera integration used in the application.

3 Operation of the Project

3.1 Power On/Off

To power on the robot, plug in the two male DC barrel jacks into the two female barrel jacks connected to the robot. The connections should look like the figure below. Indicator lights across the electronic system should turn on. To engage the actuators, the electronic e-stop must be turned on using the provided remote. It is normally closed. To indicate if actuators are engaged, the RoboClaw and L289N motor controller lights should turn on.

Figure 20: *DC barrel jack connection. The left jack is the female (charger and to the fuse box) and the right is the male jack (each battery has one connector).*

To turn the system off, shut off the electronics and stop with the remote. Then, “Shut Down” the Jetson Nano through the monitor. This can be done through the power symbol on the screen or keyboard shortcuts. Once the Jetson Nano lights are off, then it is safe to unplug the DC barrel jacks from the robot, disengaging the batteries.

To charge the batteries, use the intelligent charger shown in the figure below that plugs directly into a standard wall outlet. Once the charger is plugged into the wall, use the female barrel jack end and connect it to one of the batteries. The charger by default, will be on the car battery mode. If not, select the car battery mode. Once the battery is connected, the charger will automatically begin charging. The battery is fully charged when the charger displays “FUL.”

Figure 21: *Intelligent charger to charge the 12V12A Lead-Acid Batteries. Can be used to recover dead and sulfated batteries. Has safe charging and alerts when charging is complete. Has a female DC barrel jack connector.*

3.2 Operating Mode 1: Tele-Operation

To activate tele-operation, the user must type in the 4-digit passcode displayed on the on-board UI. This unique passcode is randomly generated, and ensures that the user is connecting to the correct SmoothOperator robot. Only one user can connect to the robot at a time to prevent robot misuse. Additionally, the 4-digit passcode is a security level that prevents unintended users from connecting to the robot. The user workflow is as follows:

1. Open the app.

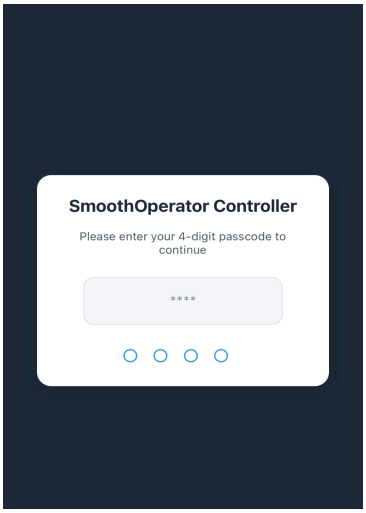

Figure 22: *The phone app displays the locked screen awaiting a correct 4- digit passcode input.*

2. Enter the 4-digit passcode

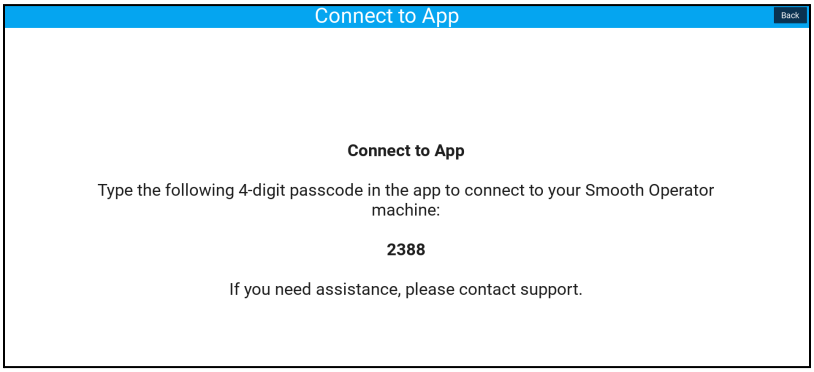

Figure 23: *The onboard screen displays a randomly generated unique passcode.*

3. Upon a successful attempt, the robot will change screens to an intuitive controller pad layout. It contains speed control, lifting/unloading luggage, and movement control all on one screen.

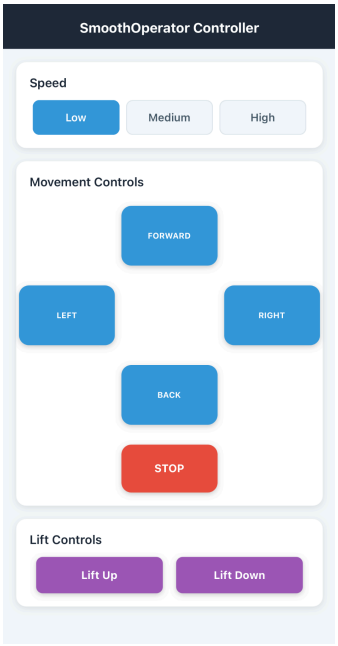

Figure 24: *The phone app control screen allows the user to operate the robot remotely.*

After gaining remote access, the user can operate the robot at a desired speed. The slow speed is very slow, and if the user is up for it, they can certainly increase the speed to their operation proficiency level.

3.3 Operating Mode 2: Autonomous Operations

1. Press anywhere on the screen to enter the main menu. From the user interface main menu, the user will press "scan boarding press".

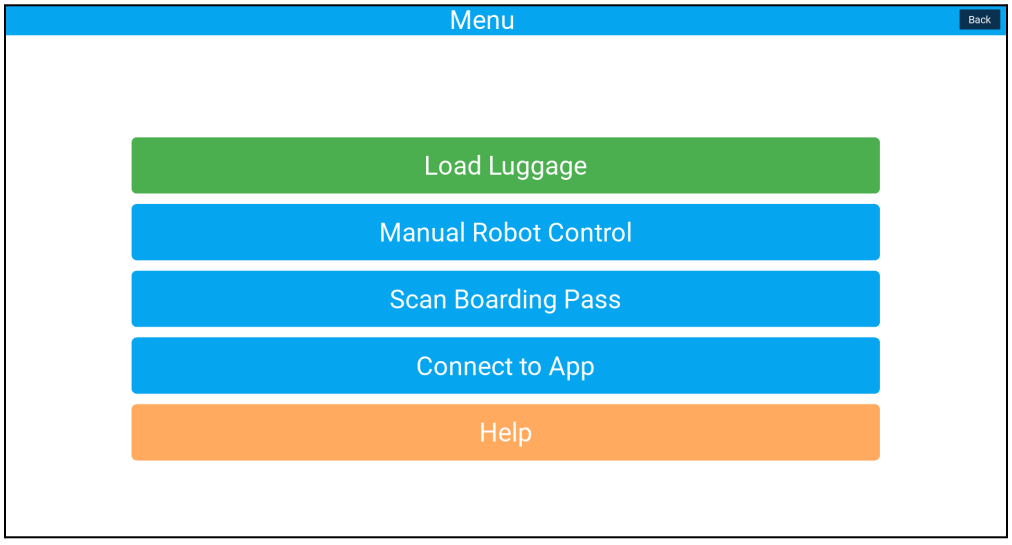

Figure 25: *User Interface Main Menu.*

2. This will prompt the user to scan their boarding pass by placing it in front of the camera located below the screen.

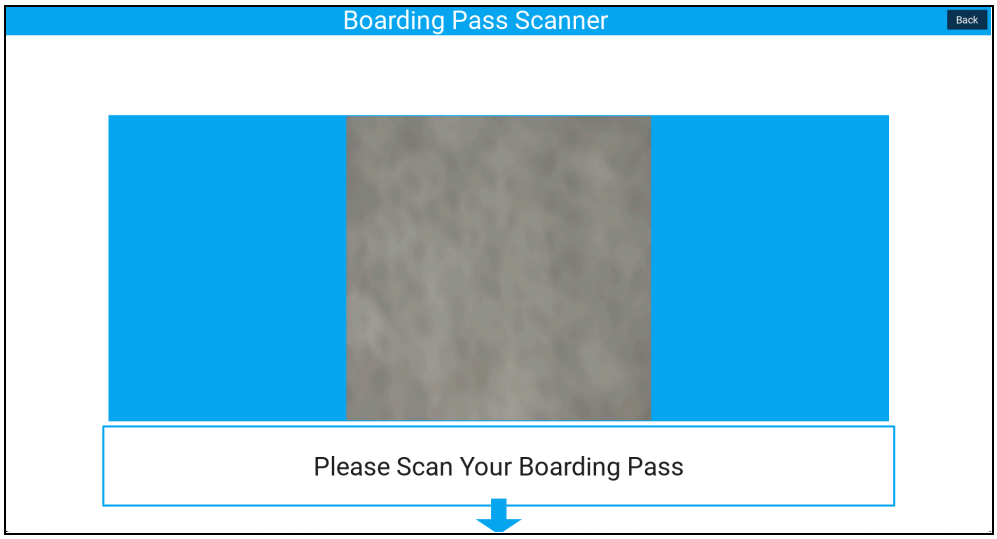

Figure 26: *Boarding pass scanning screen where the user will scan their boarding pass by lining it up with the camera below the screen.*

3. Once scanned, the interface will ask the user if they want their luggage to be brought to their gate. There will also be a green flash across the screen, giving the user feedback that their scan was successful.
    a. If the user presses yes, the robot will begin autonomously navigating to the terminal. Any obstacles encountered along the path will cause the robot to create a new route to avoid any collisions.
    b. If the user presses no, they will return to the main menu.

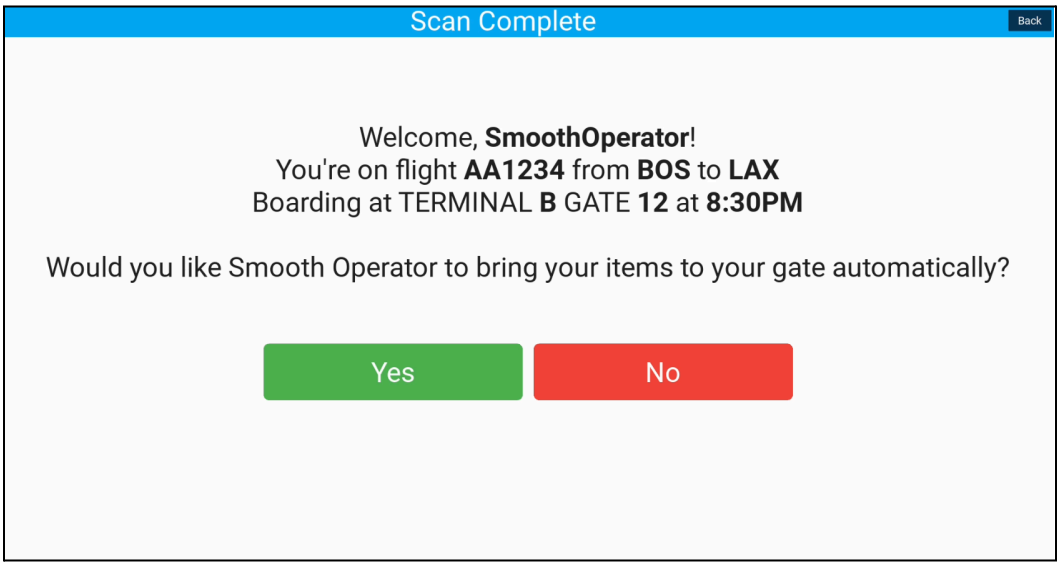

Figure 27: *A confirmation screen after a successful boarding pass scan is shown to ensure that the user would like to continue with autonomous navigation.*

4. When the robot has arrived at the terminal, it will ask the user whether or not to return SmoothOperator back home.

   a. If the user presses yes, the robot will reinitialize its position and begin autonomously navigating back home.
   b. If the user presses no, SmoothOperator will not autonomously navigate back home.
5. If SmoothOperator malfunctions or is in a dangerous situation, the user can press the remote e-stop switch to halt any motor movements.

3.4 Safety Issues (Recovery)

3.4.1 Screen Freeze

Impact:

- Could delay user operations or cause confusion.
- May lead to unintended autonomous behavior if commands are not acknowledged.

Mitigation & Recovery:

- A manual emergency stop (e-stop) is always available and wirelessly accessible, overriding software faults. Typically used by the fleet manager.
- LED indicators help bystanders identify whether the robot is active (green) or halted (red).
- System status logs are available for debugging upon reboot.
- The Arduino Mega operates on independent firmware and continues monitoring bumper and proximity sensors, maintaining safety even during high-level UI failures.

3.4.2 Battery Dies

Risks

- Unexpected battery depletion can cause the robot to stall mid-operation, potentially blocking pathways or leaving luggage stranded.

Impact:

- Sudden power loss could interrupt navigation or cause loss of UI communication.
- May lead to data loss or improper shutdown of sensitive components such as the Jetson Nano.

Mitigation & Recovery:

- All data-critical processes are automatically saved or gracefully shut down on power loss.
- Timer of one hour is monitored when the robot is use.
- Battery supply is capable of sustaining longer than one hour for electrical safety.

3.4.3 Safety Switch Triggered

Risk:

- During autonomous navigation, SmoothOperator may collide with a low-lying object or user leg if not detected by the 2D LiDAR.

Impact:

- Potential physical injury to bystanders.
- Could cause cargo to fall or be mishandled.

Mitigation & Recovery:

- Physical bumpers with limit switches act as a redundant kill system, immediately stopping movement.
- Ultrasonic rangefinders complement the LiDAR, detecting obstacles below the LiDAR plane.
- The robot enters a safe recovery state upon trigger: all motors stop, visual indicators turn red, and user input is required to resume operation.

3.4.4 Fuse Break

Risk:

- Electrical faults, such as shorts or overcurrent, can damage internal components or cause localized heating.

Impact:

- Risk of electrical fire or component failure.
- Could damage batteries, motor controllers, or power lines.

Mitigation & Recovery:

- The power architecture includes fused circuits and breaker-based isolation, ensuring only the affected subsystem shuts down.
- Electrical draws are capped at 12V/18A under full load—well within the safe limits for all components.
- Visual telemetry indicators help identify the failed circuit for quick servicing.
- All materials used (such as 80/20 aluminum framing and ABS/PLA prints) are non-flammable under operating temperatures.

4 Technical Background

Electronics
SmoothOperator's electrical system forms the backbone of its autonomous behavior, user interface functionality, and safe operation. This section outlines the key electronic components, communication protocols, and technical principles that enable the system to operate as a smart, self-contained robotic assistant.

System Overview
The electronics architecture is built on a five-processor model, distributed across:

- NVIDIA Jetson Nano: for high-level processing (navigation, UI, vision
- Arduino Mega 2560; for motor control, sensor interfacing, and safety logic
- Arduino Uno: for dedicated ultrasonic sensor processing

Each processor serves a specialized role and communicates with the others through serial protocols, allowing parallelized task handling while ensuring robustness and modularity. The connectors are all modular for easy replacement of parts and debugging.

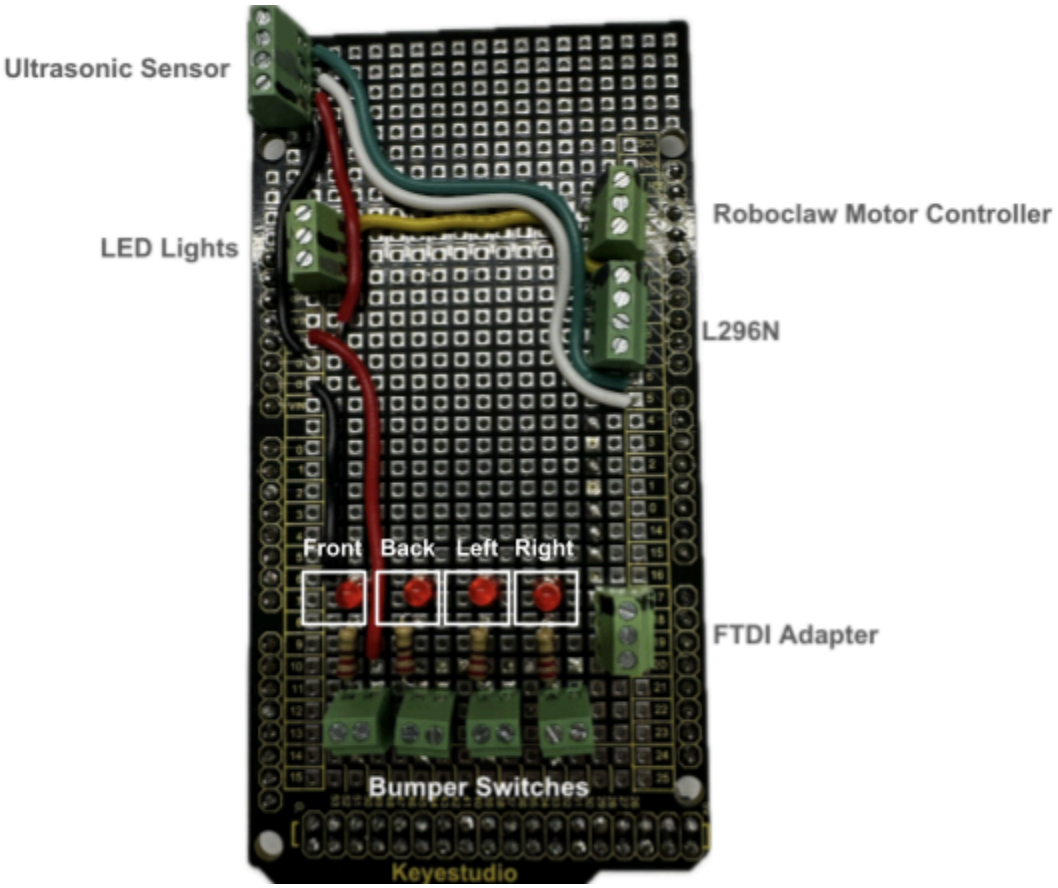

Figure 28: *Modular Arduino Mega custom shield with screw terminals allowing for easy replacements of sensors and electronics connected to the Arduino Mega. There are four LED lights that indicate which of the safety bumpers are being engaged or hit.*

Power Architecture
SmoothOperator is powered entirely by two 12V 12Ah sealed lead-acid batteries, wired in parallel to increase available current without increasing voltage. This power supply feeds all components through a fuse-protected distribution board, ensuring electrical safety and modular isolation in case of overloads. Each subsystem (motors, sensors, computing units) is protected by appropriately rated fuses, and any overcurrent triggers immediate disconnection to prevent damage.

Power characteristics:

- Operating voltage: 12V DC
- Peak current draw (full system active): ≤18A
- Subsystem fuses: rated at current thresholds specific to each unit (e.g., 5A for logic, 10A for motors)

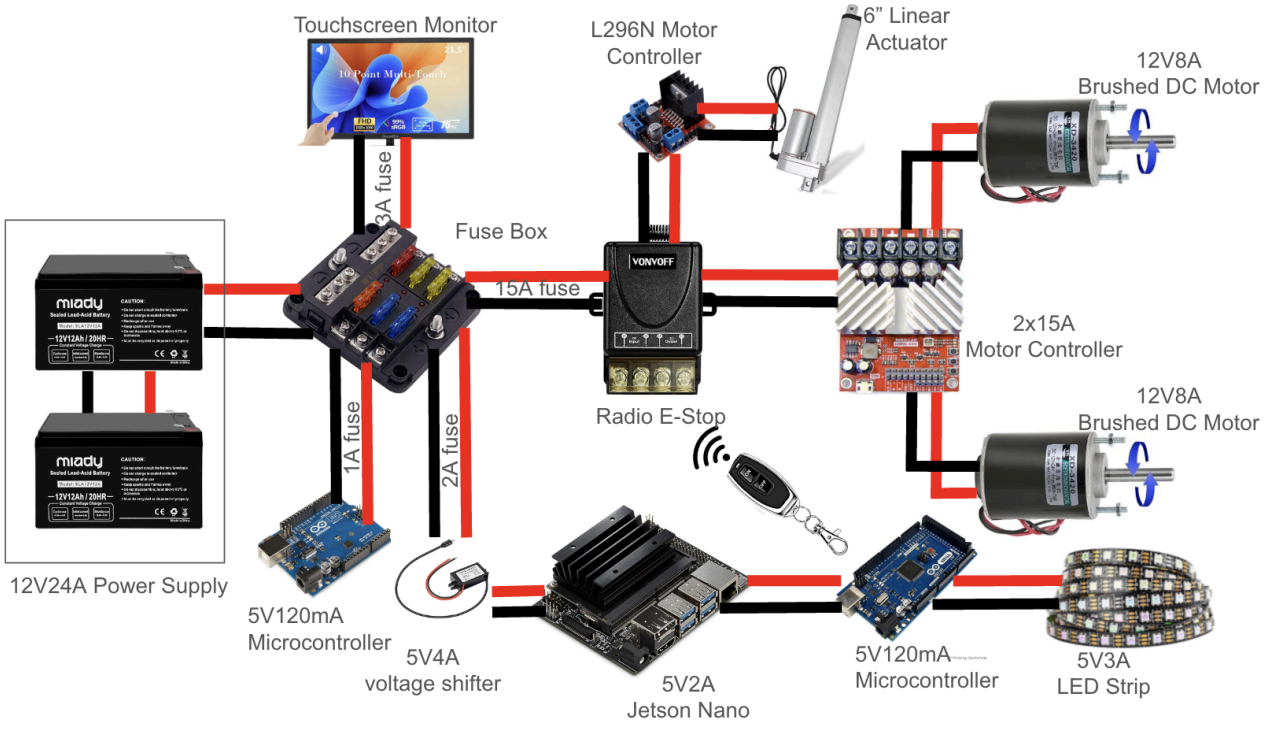

Figure 29: *Diagram of Power Distribution from the 2 Lead-Acid Batteries in Parallel Batteries.*

Motor Control and Locomotion
The drivetrain consists of two high-torque DC motors coupled to the wheels via a 10:1 compound gear reduction system. This setup increases torque and reduces speed to optimize stability, especially when carrying heavy luggage. The motors are controlled via a RoboClaw 15A motor controller, which receives PWM commands from the Arduino Mega.

Position and speed feedback are gathered using:

- Through Bore Wheel Encoders: for measuring displacement and velocity. Has quadrature resolution of 2048 Cycles per Revolution (8192 Counts per Revolution)
- MPU6050 IMU: for rotational motion (yaw) detection

These readings are used for odometry, enabling real-time estimation of the robot's position, which feeds into the navigation stack.

Sensor System
SmoothOperator relies on a blend of digital and analog sensors to achieve autonomous perception and safe interaction:

- RPLidar A1: Provides 360° planar laser scans of the environment. This forms the foundation for SLAM (Simultaneous Localization and Mapping) and dynamic obstacle avoidance.
- Ultrasonic Rangefinders (HC-SR04): Placed in a custom 3D printed casing for 60º field of view and handled by the Arduino Uno, these sensors provide short-range obstacle detection below the LiDAR's plane, particularly useful for detecting legs, small bags, or children.
- Limit Switches: Integrated into foam bumpers (pool noodles), these act as a large-scale mechanical contact sensor. Triggering any bumper immediately halts all motion via a hardware interrupt on the Arduino Mega.
- Camera (Logitech C270 HD 720p): Used to scan QR codes on boarding passes, which are processed on the Jetson Nano using computer vision.

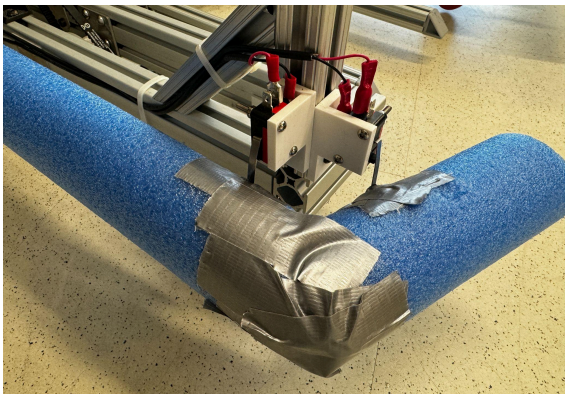

Figure 30: *Floating bumper made with foam cylinders and limit switches, minimizing contact damage.*

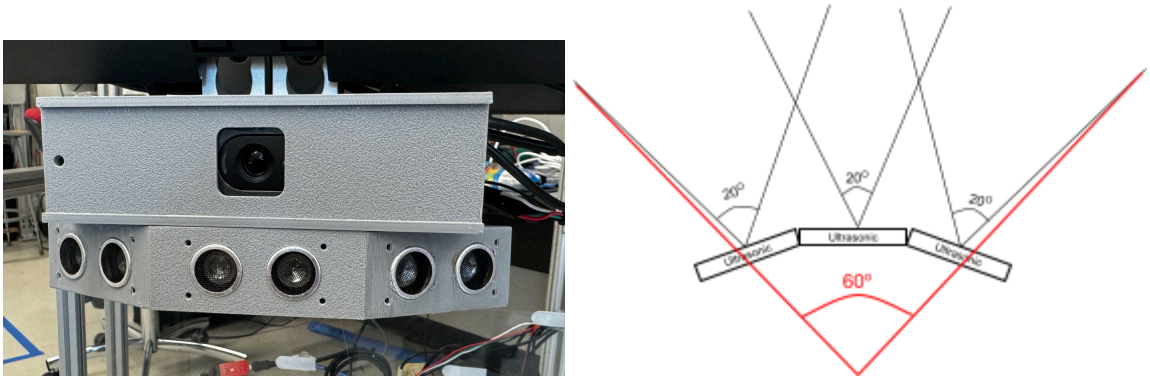

Figure 31: *Camera and ultrasonic sensor suite for boarding pass scanning and safety. The ultrasonics are angled to maximize the field of view, about 60°.*

Control and Communication

The communication pipeline is carefully designed for responsiveness and fail-safe operation:

- Serial USB/UART: between the Jetson Nano and the Arduino Mega handles movement commands and sensor and command data exchange. From the Arduino Mega to the Jetson, a FTDI USB adapter is used to increase data bandwidth and reliability.
- SoftwareSerial: on the Arduino handles motor control signaling and encoder feedback, using separate pins to reduce latency and congestion.
- WiFi module: Connected to the Jetson Nano, supports remote access and mobile app connectivity.

All communication includes built-in feedback confirmation, ensuring no command is executed without successful transmission and acknowledgment.

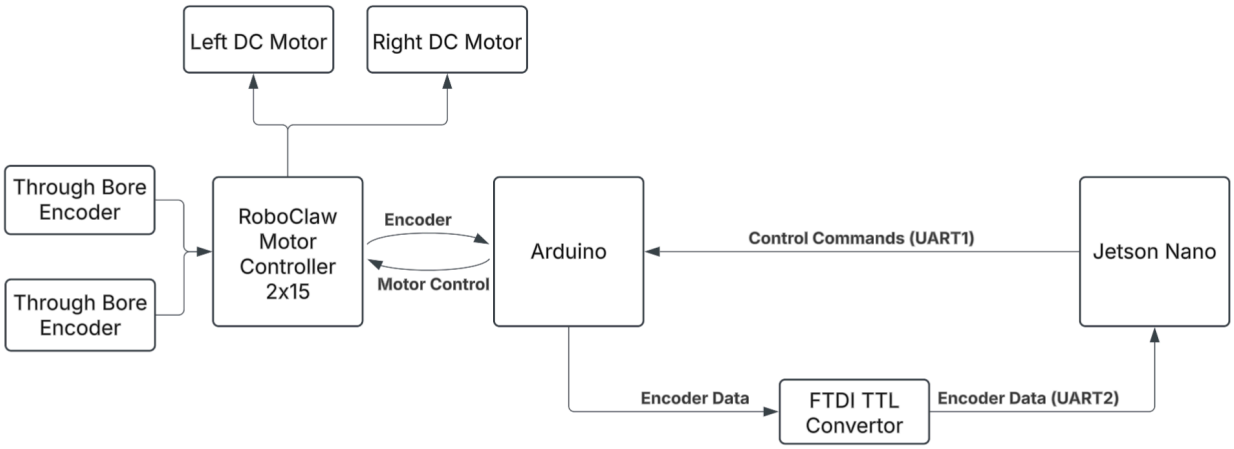

Figure 32: *Diagram of the communication feedback loops involved in the immediate navigation (excludes Jetson Nano long-term autonomous navigation).*

Visual Feedback and Safety Indicators
A ring of programmable RGB LED strips under the chassis visually communicates the robot's state:

- Green = In motion
- Red = Stopped (manual halt, obstacle, or bumper contact),

This visual feedback system is controlled by the Arduino Mega and updated in real-time to reflect system status, aiding both user experience and safety awareness.

Emergency and Safety Systems

Two layers of hardware fail-safes exist:

- Radio-frequency Emergency Stop: A handheld remote can wirelessly disable all motor activity within a 3 to 10 ft range via a relay-based circuit breaker.
- Hardware Override (Bumpers and Ultrasonics): Operate independently of high-level software. If triggered, the Arduino Mega cancels all active commands and halts all actuators.

Additionally, every component is physically protected with custom 3D-printed motor shrouds and water-jetted metal brackets, shielding against environmental debris and accidental contact.

Mechanical
The chassis is designed to be sturdy to hold up to repeated use of the robot throughout its lifespan. A majority of the chassis is constructed from 80/20 T-slotted aluminum extrusions due to its strength, availability, and modularity. Its size strikes a balance between material cost and the interior storage space and visibility of the robot. There is a bumper made from a pool noodle that is mounted to limit switches around the perimeter of the robot that halt the robot's motion when activated. The chassis itself sits within a 24"x24" profile, with the attached bumper extending the profile of the robot to 31"x31".

The compound train used for the drivetrain consists of a 15 tooth aluminum pulley, a 45 tooth PLA pulley, a 15 tooth ABS gear and a 50 tooth ABS gear combined to achieve a 10:1 gear reduction. This gear reduction reduces the speed and increases the torque of the motors to values that are far closer to our desired values. The ABS gears were printed with 40% gyroid infill and 4 wall loops, and the PLA pulley was printed with 25% cubic infill and 2 wall loops. The overall printing time for these parts is around 7 hours and 24 minutes.

The central driven wheels tilt the robot back slightly to ensure the driven wheels always make contact with the ground and to help retain luggage while the robot is in motion. The thermoplastic elastomer tread allows the wheels to reject small debris while the robot is

in motion. The lifting mechanism is similar to a forklift, consisting of a linear actuator mounted to a pair of forks made of 8020. This design allows for luggage to be rolled into position above the forks within the robot, which helps maintain the position of the robot's center of mass.

Network
We use WiFi to enable seamless communication between all components of SmoothOperator. The onboard interface running on the Jetson Nano and a mobile interface built with React Native both communicate with the backend server, which is also hosted on the Jetson. This server, developed using Node.js, handles both HTTP API requests and real-time WebSocket communication. Through this server, the frontend sends commands, such as navigation or lifting instructions, which are then processed and forwarded to the Arduino Mega over a USB serial (UART) connection. The Arduino, which controls the robot's motors via RoboClaw motor drivers, receives these commands in the form of velocity and lift values. It also sends back sensor data or status updates to the Jetson, which are then relayed to the frontend over WebSocket. In addition to direct control, the server interfaces with the ROS navigation stack running locally on the Jetson. For example, when a QR code is scanned, the resulting destination data (such as terminal and gate) is forwarded via HTTP to a ROS node that handles autonomous navigation. This network architecture, built on top of WiFi, allows for real-time robot control, autonomous routing, and a flexible interface between hardware and software components, all without the need for wired communication between the user and the robot.

User-Interface
The User Interface allows the user to interact with the SmoothOperator system. It is composed of an onboard robot UI touchscreen and mobile application. The onboard UI is intended for any user who wants to approach the robot and use the SmoothOperator services, while the mobile application is intended for maintenance users who want to control SmoothOperator remotely.

The onboard UI is developed using a Python Kivy framework, a framework optimized for touchscreen interaction. The screen is intuitive to use as it consists of various screens with corresponding functions: scanning a boarding pass, loading luggage, manual control, app connection, and a help page. Additionally, the SmoothOperator robot has a robot styled face that can speak and look in the direction of movement. The animated mouth and eyes give a friendly appearance that makes the robot seem engaging. All of this is possible with the Kivy ScreenManager package and custom widgets (rounded buttons, consistent styling, etc.). As for the integration of the QR code scanning functionality with the UI, the system utilizes OpenCV ad Pyzbar to capture and decode the boarding pass data (in our case, it is a QR code containing passenger details in a JSON format). Using the Kivy animations, the screen will flash green upon a successful scan. The JSON payload will be sent to the backend server through an HTTP POST request and trigger the start of the autonomous navigation algorithm.

The mobile application UI is developed using the React Native framework, a framework that can be used to parallelize cross-platform app development. Upon startup, the user is prompted to enter a 4-digit passcode that is securely transmitted to the robot's backend

over an HTTP POST request verification. This passcode system ensures that only authorized users can remotely control the robot. After a successful authentication, the backend server sends an AUTH_SUCCESS signal over a WebSocket connection that reflects on the on-board UI as well as the mobile application. The app then transitions to an interactive gamepad interface that includes manual control, speed control, and lifting and loading the luggage buttons.

4.1 Navigation

As detailed briefly in section 2.4, our navigation system is made up of interconnected nodes that work together to allow the robot to autonomously navigate in any environment. Below is a more technical breakdown of each major component:

1. Sensor Data Nodes
    a. These nodes are responsible for interfacing with hardware sensors to provide real-time environmental and motion data:
        i. LiDAR Node: Continuously publishes 2D laser scan data representing the surrounding environment. The LiDAR scans in a 360 degree radius with a range of 12 meters.
        ii. Wheel Encoder & IMU node: Reads and publishes raw odometry data, including displacement, velocity, and orientation.
2. Odometry Calculation Node (Differential Drive Kinematics)
    a. This node uses the data from the wheel encoders and IMU to calculate the robot's position and orientation over time via dead reckoning. The differential drive kinematics model considers the physical wheelbase and updates the robot's estimated trajectory.
3. Mapping Node (Performed Pre-Navigation)
    a. This node constructs a static 2D occupancy grid map using SLAM (Simultaneous Localization and Mapping) techniques. Our specific implementation uses the Rao-Blackwellized Particle Filter (RBPF), where each particle represents a potential trajectory of the robot and carries its own map of the environment. The algorithm selectively resamples particles only when necessary, mitigating particle depletion. The user/operator is able to annotate the map after it is produced to correct minor blemishes or mark new environment obstacles.
4. Localization Node
    a. This node estimates the robot's current position with respect to the pre-built map by using Monte Carlo Localization (AMCL). It compares incoming LiDAR scans with the static map, which helps to align the odometry-based coordinate frame (odom) with the map-based global frame (map), allowing for accurate path planning and correction of accumulated drift (known issue for dead reckoning).
5. Transform (TF) Nodes
    a. Multiple TF nodes maintain a dynamic tree of coordinate frames between the robot base, its sensors, and the world. These transforms enable seamless integration of sensor data and motion commands by keeping all information spatially aligned.
6. Costmap Nodes

   a. Two separate costmaps are maintained:
      i. Global Costmap: Built using the static map produced by the mapping node. It is used for long-range path planning and marks static and annotated obstacles.
      ii. Local Costmap: Continuously updated with real-time LiDAR data to account for dynamic obstacles (e.g., humans). It is used for short-range navigation and obstacle avoidance.
   b. Both costmaps are implemented with a grid-based representation of space where each cell in the grid holds a cost value indicating whether the area is free, occupied, or unknown, and whether it's near an obstacle. Both costmaps also inflate a region around obstacles to create a buffer zone, so the robot doesn't plan paths too close to walls or objects.
7. Path Planning Nodes
   a. Global Planner: Generates a high-level path from the robot's current pose to the goal, taking into account the global costmap to avoid static and restricted areas. It uses Dijkstra's algorithm to compute the shortest path with cost-based weighting.
   b. Local Planner: Implements the Dynamic Window Approach (DWA) which considers multiple possible velocity commands and simulates the robot's motion over a short time step for each sampled command. It evaluates the resulting trajectories using a cost function that balances multiple objectives (how well it follows the global path, if it is avoiding obstacles in the local costmap, and if the robot can fit through the path).
8. Recovery Behavior Node
   a. This node defines actions the robot should take when it becomes stuck or cannot make forward progress. Examples include reversing slightly, reattempting the global path from a new position, or clearing the local costmap.
9. Command Velocity Node
   a. This node subscribes to the velocity outputs produced by the local planner and sends them to the microcontroller (in our case, an Arduino) via a serial interface. The microcontroller then translates these commands into motor signals to drive the robot.
10. User Interface Communication Node
   a. This node facilitates communication between the robot and the user. It can receive destination commands from the user interface and relays them to the path planning nodes. This allows for dynamic goal-setting during operation without restarting the navigation stack.

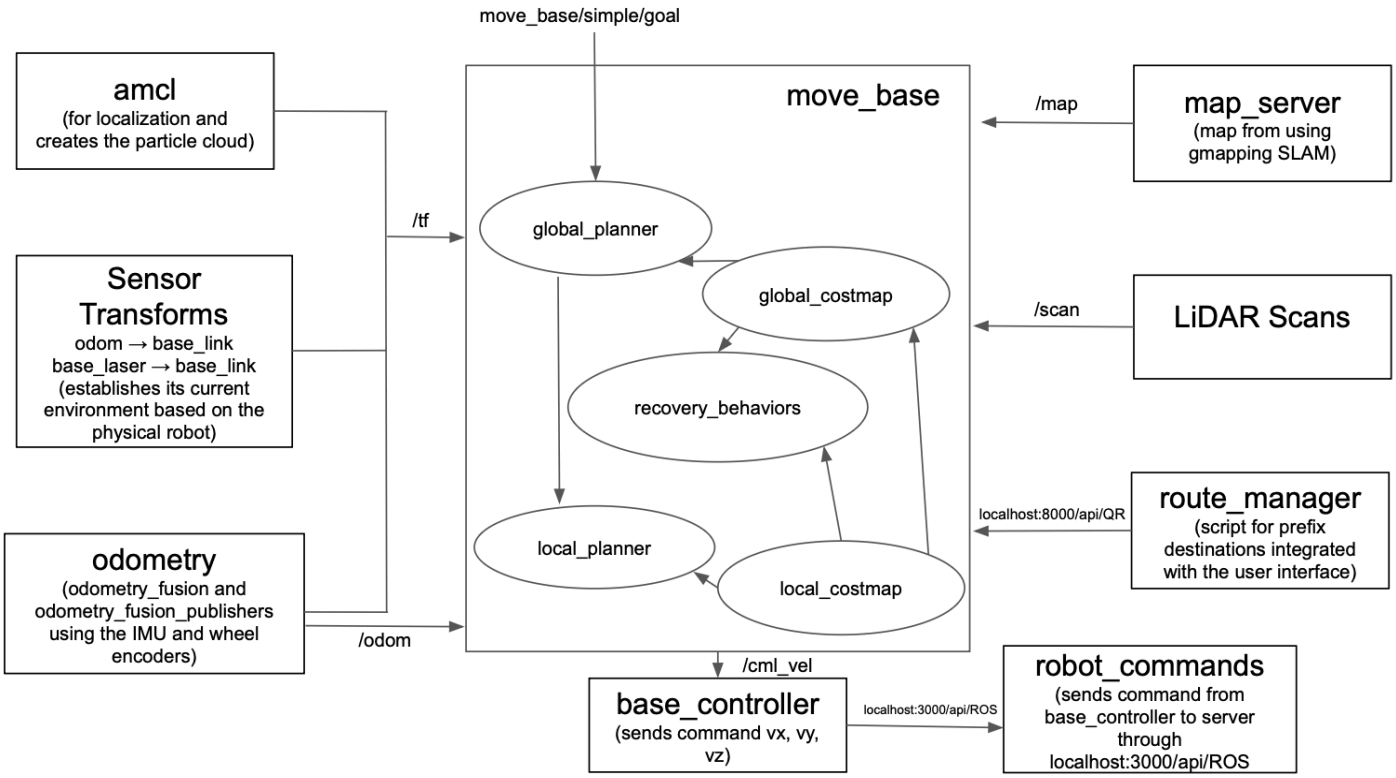

Figure 33: *Diagram of the feedback loops for the navigation stack.*

5 Relevant Engineering Standards

SmoothOperator’s design and implementations follow several key engineering standards to ensure safety and reliability. For instance, IEEE 802.11 and IEEE 802.15 are showcased through the use of our Wi-Fi and Bluetooth protocols for wireless communication between the Jetson Nano, mobile interface, and sensors.

The National Electrical Safety Code (NESC) is applied through the use of a centralized fuse box, which ensures safe power distribution and allows us to precisely manage how 12V power is delivered to multiple subsystems and devices. We use proper grounding and isolation practices to protect against overcurrent and electrical faults.

The National Electrical Code (NEC) (NFPA 70) further governs our internal wiring and grounding practices, ensuring that all power distribution within the dolly complies with U.S. electrical safety standards.

ISO 12100 – Safety of Machinery is reflected in our approach to mechanical design, where we performed risk assessments and implemented appropriate protective measures (e.g., bumpers, enclosure shielding) to reduce hazards associated with movement and human interaction.

IEEE Robotics and Automation standards are reflected in the robot’s dynamic obstacle avoidance, bumper-triggered emergency stops, and layered safety systems for autonomous navigation in human environments.

As for government and accessibility standards, the Americans with Disabilities Act (ADA) also informs our physical design choices by ensuring that SmoothOperator does not obstruct public walkways and can coexist respectfully with other assistive technologies in shared environments.

ISO/IEC software standards are demonstrated through a structured, modular ROS-based architecture and version-controlled documentation, while standard communication protocols like HTTP, JSON, and WebSocket support reliable backend interfacing. Overall, SmoothOperator was developed with core engineering principles in mind to deliver a safe, efficient, and high-quality product.

6 Cost Breakdown

***Table 1**: Bill of Materials for Beta Version*

Project Costs for Production of Beta Version (Next Unit after Prototype)				
Item	Quantity	Description	Unit Cost	Extended Cost
Mechanical Hardware				
1	4	Clamping Shaft Collar - 1/2" Hex ID	$0.99	$3.96
2	3	8020, 1020, 10 Series 1 Inch x 2 Inch T-Slotted Aluminum Extrusion DIY Extruded Linear Slot Bar Rail 80/20 (Smooth, Clear Anodize, 72" Long)	$47.43	$142.29
3	5	8020, 1020, 10 Series 1 Inch x 1 Inch T-Slotted Aluminum Extrusion DIY Extruded Linear Slot Bar Rail 80/20 (Smooth, Clear Anodize, 72" Long)	$33.25	$166.25
4	2	85t x 9mm Wide Timing Belt (HTD 5mm) (WCP-0622)	$6.99	$13.98
5	6	T-Slotted Framing End Cap for 2" High Double Rail	$3.05	$18.30
6	1	Multipurpose 6061 Aluminum U-Channel 1 ft length 0.13" Thick, 1" High x 2" Wide Outside	$9.55	$9.55
7	2	High Torque DC Motor 12 V/24 V Permanent Magnet Motor Mini DC Motor DIY Generator Motor 30W CW/CCW 3500/7000RPM with Moun	$25.79	$51.58
8	6	0.500" Hex ID x 1.125" OD x 0.313" WD (Flanged Bearing v2)	$1.49	$8.94
9	4	WCP-1420 15t x 9mm Wide Aluminum Pulley (HTD 5mm, 1/2" Hex Bore)	$9.99	$39.96
10	2	18-8 Stainless Steel Button Head Hex Drive Screw, 1/4"-20 Thread Size, 3/8" Long, 50 count	$6.76	$13.52
11	3	T-Slotted Framing, End-Feed Nut for 1" and 25 mm High Rail, 1/4"-20 Thread, 25 count	$7.83	$23.49
12	1	⅛" Acrylic Sheets (Custom Laser Cut)	$288.26	$288.26

13	4	Caster Wheels	$7.65	$30.60
14	2	Colson Performa (4" x 0.875", 1/2" Hex Bore)	$7.49	$14.98
15	1	1/2" OD Hex Stock (18")	$4.49	$4.49
16	12	90º 8020 Brackets	$8.00	$96.00
17	1	PLA 3D Printed Parts (Variety)	$3.00	$3.00
18	1	ABS 3D Printed Parts (Variety)	$2.16	$2.16
19	1	12V 6" DC Linear Actuator	$30.99	$30.99
20	4	Pool Noodles	$1.25	$5.00
Processors, Electronics, and Sensors				
21	1	RoboClaw Motor Controller	$99.99	$99.99
22	2	REV Through Bore Encoder	$49.00	$105.53
23	1	SparkFun 9DoF IMU Breakout-ICM-20948	$18.50	$18.50
24	1	6 Circuit 12V Fuse Box	$21.99	$21.99
25	1	ELEGOO Arduino Mega 2560 R3	$22.99	$22.99
26	1	Jetson Nano 4GB RAM 16G	$229.99	$229.99
27	1	FT232RL Mini USB to TTL Serial Converter Adapter Module 3.3V/5.5V	$6.49	$6.49
28	1	USB 3.0 Hub	$6.36	$6.36
29	1	Touchscreen Monitor	$127.49	$127.49
30	1	WS2812B RGB LED Strips - 300 Pixels, 16.4' Waterproof	$22.99	$22.99
31	1	ELEGOO Arduino Uno R3	$14.99	$14.99
32	3	HC-SR04 Ultrasonic Sensors	$1.91	$5.73
33	1	Logitech C270 HD Webcam	$19.99	$19.99
34	1	L298N Motor Driver	$6.19	$6.19
35	1	RPLiDAR A1	$99.00	$99.00
36	8	NO Limit Switches	$1.29	$10.32
Electrical Components				
37	1	12V12A Batteries (2 Pack)	$39.99	$39.99

38	2	Male DC Barrel Jack Connectors	$2.00	$4.00
39	3	Female DC Barrel Jack Connectors	$2.00	$6.00
40	1	22AWG Solid Copper Core Wires (Variety of colors and lengths)	$10.00	$10.00
41	1	12V to 5V4A Power Leveler	$8.99	$8.99
42	1	Radio E-Stop	$15.99	$15.99
43	1	Arduino Mega PCB Board	$1.29	$1.29
44	8	Screw Terminals (Varying Sizes)	$0.50	$4.00
45	14	Female Spade Connectors	$0.69	$9.66
56	1	12V/24V 14A Intelligent Automatic Battery Charger	$32.99	$32.99
			Beta Version-Total Cost	**$1,918.75**

The development of *SmoothOperator* was guided by careful consideration of cost constraints and strategic material sourcing, made possible through a combination of institutional support, donations, and targeted funding. Our core objective was to maximize functionality while remaining within a limited budget, leveraging prototyping materials and selectively integrating more expensive components only where they were essential to core performance.

Several key components were generously provided by our client, significantly reducing upfront expenses. These included the 2D LiDAR unit, which served as the primary sensor for obstacle detection and navigation, as well as essential assembly hardware such as PLA and ABS filament, jumper wires, screws, fasteners, mounting brackets, and a Logitech C270 HD Webcam. The 2D LiDAR, while less capable than its 3D counterparts, offered sufficient environmental awareness for indoor, semi-structured environments and aligned with our computational and financial constraints. A 3D LiDAR system would have necessitated a more powerful (and expensive) processing unit, further increasing cost and complexity.

The Jetson Nano 4GB RAM 16GB storage was contributed by one of our team members and was not purchased using project funding.

To construct a protective and visually accessible enclosure around SmoothOperator, we obtained acrylic material through ESIF funding made available by BTEC. The modular frame was built using 8020 1x1 aluminum extrusions, which were provided by Professors Thomas Little and Ryan Lagoy. These contributions enabled a robust yet lightweight structure that could be easily reconfigured during iterative development.

However, the unit costs of the contribution pieces from the different sources are all accounted for in the table.

Material selection was informed by both prototyping needs and forward-looking considerations for scalability. PLA and ABS plastics, while not optimal for long-term durability, offered rapid iteration capabilities. In a production setting, these parts could be replaced with more durable materials such as Nylon or injection-molded components. Similarly, the 8020 aluminum and acrylic panels, though excellent for prototyping, could be substituted with less expensive alternatives when mass-produced.

Overall, this project was executed with a lean budget through thoughtful material sourcing, academic collaboration, and conscious design trade-offs. Every component was selected with both immediate feasibility and long-term viability in mind.

7 Appendices

7.1 Appendix A - Specifications

***Table 2**: The specifications and requirements of SmoothOperator.*

Requirement	Value, range, tolerance, units
Robot Maximum dimension	2.25’x 2.25’x4.75’
Maximum Luggage dimension	14”x16”x30”
Maximum Speed	1.2 m/s
Turning Degree	0^{o}
Load weight	≤ 80 lbs
Latency	≤ 100 ms
Robot Battery Life	1 hour
Power Rating	12V 24A

7.2 Appendix B – Team Information

Christian So: Computer Engineering student who brings extensive experience in robotics and AI, having competed in numerous robotics competitions and actively participating in machine learning and robotics research, with interests in data efficient learning and accessible safety controls. As the director and founder of BU Hacks HS, an organization that hosts an annual high school hackathon at Boston University, he also demonstrates strong leadership skills. As project manager, he developed the end-to-end system integration, project schedule, and the low-level controllers of the robot. He wore many different hats throughout the team to help fully realize SmoothOperator.

Eric Chen: Computer Engineering student who has extensive experience in embedded systems, acquired through both coursework and his internship at Textron last summer. He also showcases strong leadership abilities as a teaching assistant for several engineering courses at Boston University. Eric will applies these skills in designing and assembling the embedded hardware, as well as developing the firmware and software that interface with the system. His expertise in system optimization and reliability ensures seamless product performance, contributing to high-quality service.

Jacob Chin: Double major in Computer Engineering and Biomedical Engineering with many experiences as a software engineer in the biotech industry. He has leadership experience as he was the former President of the BU Biomedical Engineering Society. He will use his skills and contribute in both hardware and software components of the product. His background in biomedical engineering enhances his understanding of

human-centered design, which helps tailor the product to meet the comfort, safety, and usability standards expected in any industry.

Celine Chen: Computer Engineering student who has a strong background in embedded systems and software, gained through her research and software engineering experiences. She also possesses notable leadership skills as the former director of BostonHacks, Boston University’s largest student-run hackathon. Celine contributes to the software development of the product. Her attention to detail, user interface design skills, and ability to streamline processes enhance the product’s user-friendliness and appeal, creating an overall better experience.

Nicholas Nguyen: Mechanical Engineering student with significant experience in robotics design and manufacturing, competing in multiple robotics competitions. He has leadership experience as a team lead for several robotics competitions. Nicholas brings skills in CAD and manufacturing to make extensive contributions to the hardware components of the product, working to realize the product in both a virtual and physical space.