Boston University
Electrical & Computer Engineering
EC464 Capstone Senior Design Project

Final Prototype Testing Report

SmoothOperator

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

Required Materials

Hardware*:*

- Clamping shaft collar - ½” Hex ID
- 6 mm D to Hex Adapter
- 8020, 1020 10 Series 1 Inch
- Radio E-Stop
- 85t x 9mm Wide Timing Belt
- T-slotted Framing End Cap for 2” High Double Rail
- Multipurpose 6061 Aluminum U-Channel 1 ft length
- RoboClaw Motor Controller
- High Torque DC Motor 12 V/24 V Permanent Magnet Motor Mini DC Motor DIY Generator Motor 30W CW/CCW 3500/7000RPM with Mount
- 0.500" Hex ID x 1.125" OD x 0.313" WD (Flanged Bearing v2)
- WCP-1420 15t x 9mm Wide Aluminum Pulley (HTD 5mm, 1/2" Hex Bore)
- 4 Caster Wheels
- Colson Performa (4" x 0.875", 1/2" Hex Bore)
- (4" x 0.875", 1/2" Hex Bore) VEX Colson Performa Wheels
- 8" x 8" Stainless Steel Sheet 0.12"
- Miady 12V 12 Ah Rechargeable Sealed Lead Acid Battery
- 2GB Nvidia Jetson Nano
- 22 AWG Jumper wires
- Breadboard
- Spade connectors
- PCB Arduino Uno Shield
- PCB Arduino Mega Shield
- Screw Terminals
- Maxbotix Ultrasonic Rangefinder - LV-EZ1
- RPLIDAR A1
- MPU6050 IMU
- Arduino Mega 2560 Rev3
- Programmable LED Strips

- 21.5 Inch Touchscreen Monitor
- Logitech 720p Webcam
- Limit Switch (V-153-1C25)
- Pool Noodles
- L296N Motor Driver
- 6-Inch Linear Actuator
- HC-SR04 Ultrasonic Sensors
- Through Bore Wheel Encoder

***Software**:*

- Node.js Server
- Nvidia Jetson Nano Client
- User Interface App Client
- On-board User Interface Client
- Serial Communication
- C Microcontroller ultrasonic and limit switches safety code
- Python Sensor Reading Code
- ROS NAV1 (odometry, sensor transform, sensor sources, map_server nodes) for mapping

Set-Up

The SmoothOperator setup is divided into three main parts: physical chassis, user interaction, and mapping.

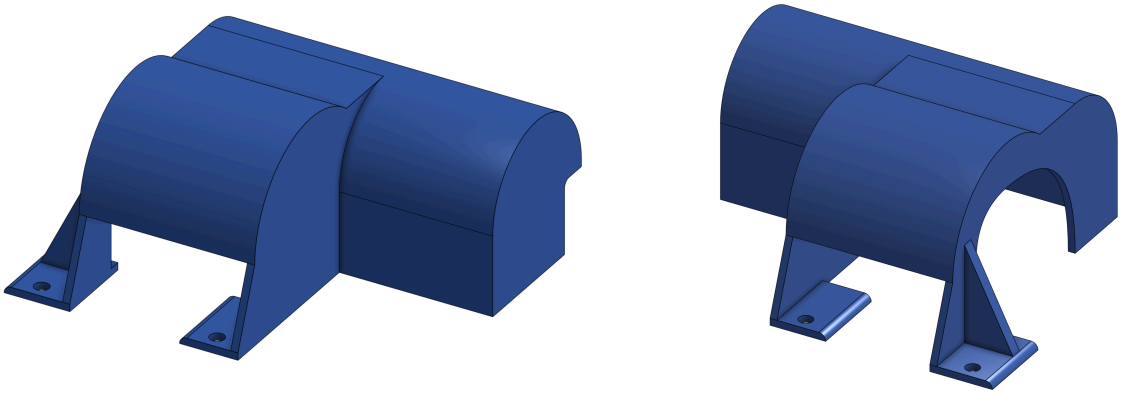
Mapping and Navigation:

Our mapping and navigation system consists of 11 ROS nodes. These nodes perform tasks such as environment mapping, localization, path planning, and autonomous navigation. The first step in setting up navigation is running the gmapping node, which uses the RPLidar A1M8 to generate laser scans of the environment. Following this, ten additional ROS nodes and two launch files are used to integrate and implement the full navigation stack.

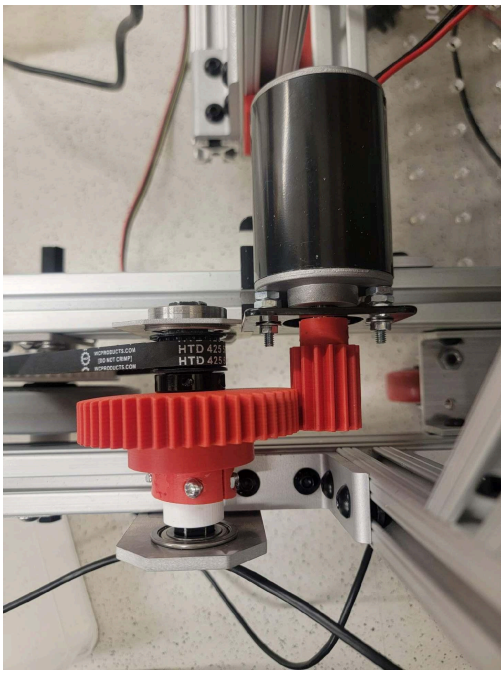
The sensor source node reads data from the RPLidar A1M8, formats it into ROS messages, and publishes it to a topic, a communication channel used for exchanging information. These laser scans are also used to create local cost maps within a specific radius around the robot, enabling dynamic obstacle avoidance. The costmaps use a sliding window approach, meaning obstacles are removed once they fall outside the robot's local sensing range. The odometry system consists of two nodes: one node acts as a data producer, extracting yaw data from the IMU and displacement/velocity data from the wheel encoders; the other node processes this information to compute the robot's updated pose and publishes it to the `/odom` topic. The map_server provides the static map created during the initial SLAM phase using gmapping, while the amcl node handles localization and produces the particle cloud used to estimate the robot's position within the map. The sensor transforms node publishes fixed coordinate transforms between rigidly connected parts of the robot. In our setup, we define static transforms from odom (IMU and wheel encoder frame) to base_link (chassis of the mobile base), and from base_laser (LiDAR) to base_link. The route_manager node processes QR code scans provided by the user and translates them into navigation goals. The move_base (comprising the costmaps, path planner, and recovery behaviors) computes the velocity commands (vx, vy, vtheta) and publishes them to the /cmd_vel topic. The robot_commands node then forwards these commands to the onboard server through an API endpoint. The launch file runs all the nodes we created, and the gmapping node uses sensor scans, odometry information, and static transforms to create a map, which can be visualized in Rviz.

Physical Chassis:

SmoothOperator's chassis is largely constructed from 80/20 T-slotted aluminum extrusions. This material choice grants the robot a durable frame while also allowing for high flexibility in mounting electronics and other hardware to the robot. The robot utilizes a tank drive drivetrain with two 4" wheels mounted at the center and 4 caster wheels mounted at the corners. The driven wheels are mounted ⅛" lower than the caster wheels to ensure that they consistently make contact with the floor. The drivetrain utilizes a 10:1 compound gear reduction to decrease the motors' speed and increase the motors' torque to our desired levels. SmoothOperator stands at 1.067 meters tall, intentionally designed to accommodate both the luggage and the display. A forklift mechanism is integrated within the U-shaped chassis, allowing the luggage to rest securely during transit. This design choice also supports one of our stretch goals: automating the loading and unloading of luggage. Custom designed shrouds have been fitted onto the drivetrain to protect the motors, gears, and pulleys from impacts and debris that could interfere with the robot's operation.



CAD of shrouds to enclose the drivetrain.



Gear box of drivetrain with 3D printed gears and custom water jetted brackets.

User Interaction:

The user can control the SmoothOperator robot with the onboard UI screen. This front end can communicate bidirectionally to a node server hosted on the Jetson Nano, and from the Jetson Nano, information is bidirectionally communicated to the microcontroller (Arduino Mega) over serial communication. To initiate customer interaction with the SmoothOperator system, the user can tap anywhere on the face screen to pull up the menu screen. The user will slide the luggage under the SmoothOperator screen, where there is a forklift to carry the luggage. The user can raise the luggage through the UI by activating the lifting mechanism with an up and down button in the load luggage screen. After loading the luggage, the user has two means of controlling the robot: manual control and autonomous navigation. For manual control, the user can simply click the manual control option to see a control pad (up, down, left, right, and stop) allowing button press inputs. Speed is variable at low, medium, and high speeds, allowing the user to move the SmoothOperator robot at appropriate pace. To activate autonomous navigation, the user must scan their boarding pass since this will give information to the SmoothOperator system on the destination (terminal and gate). In our case, we have a QR code that will be scanned using the onboard camera and sent in JSON format to the system. The navigation (ROS) will then identify the destination and prompt the user if they'd like to activate the automatic transportation of their

items. If the user mistakenly chooses this option, they click no, which will bring them back to the menu screen. Upon activating autonomous navigation, the SmoothOperator will slowly move to the destination by running the navigation algorithm. Once SmoothOperator reaches its destination, the onboard screen will prompt the user to see if they're done using SmoothOperator. If they are not finished, the robot will return to the menu screen. If they are finished, they can send the robot away, and with this option, the robot will autonomously navigate back to its docking area. For the airport operations team, we have created a React Native application to help with the management of the SmoothOperator robot. By leveraging React Native's versatility, the application can be operated through a smartphone, tablet, or laptop. The onboard UI has an option to connect to the app, where it will display a four-digit code. In order to gain access to remote manual control, the four-digit code must be typed into the app. To ensure safety, the buttons operate with a momentary switch (non-latching), so when the user stops pressing the button, the SmoothOperator robot will stop. Additionally, user awareness is enhanced with visual indicators from the under glowing LED strips: green for movement and red for stopping. The manual control mode offers precise control while maintaining an accessible experience.

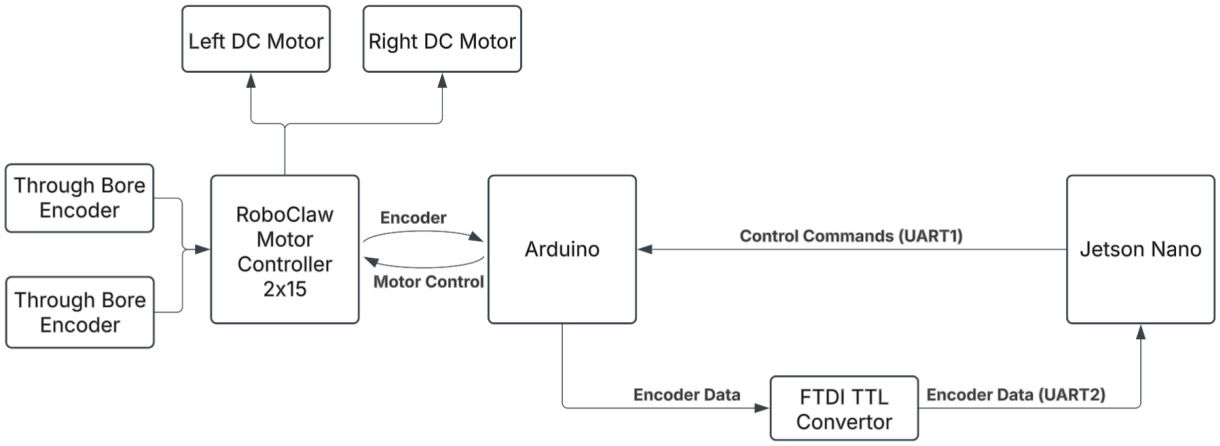


Diagram of all the Bi-Directional Communication between 3 Processors.

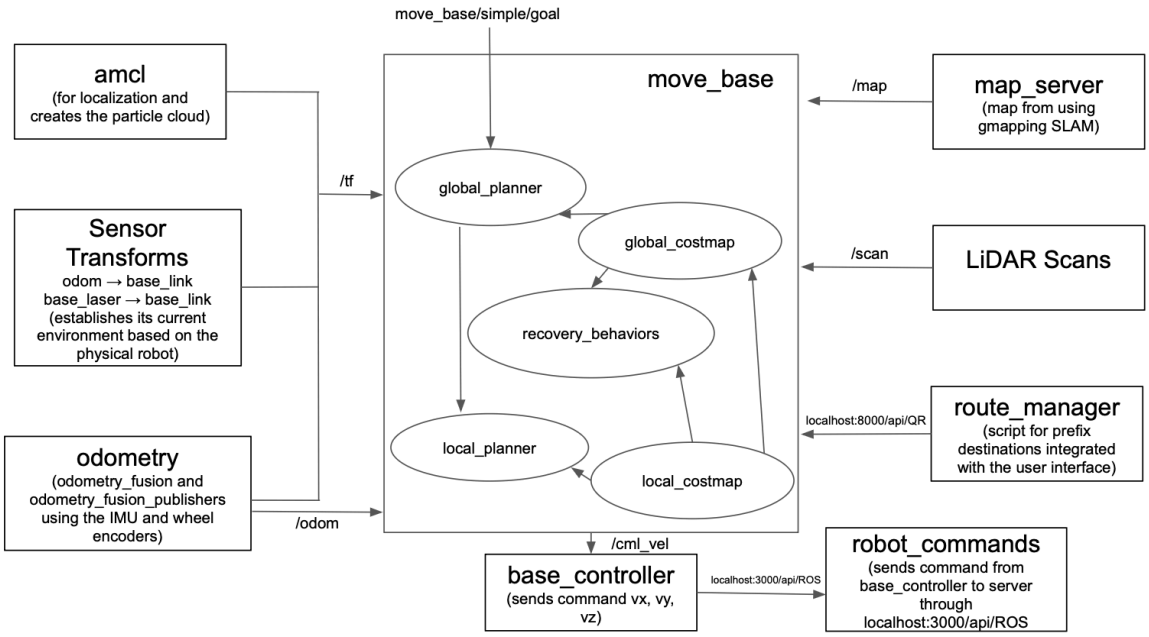


Diagram of all the ROS nodes working together.

Pre-testing Setup Procedure:

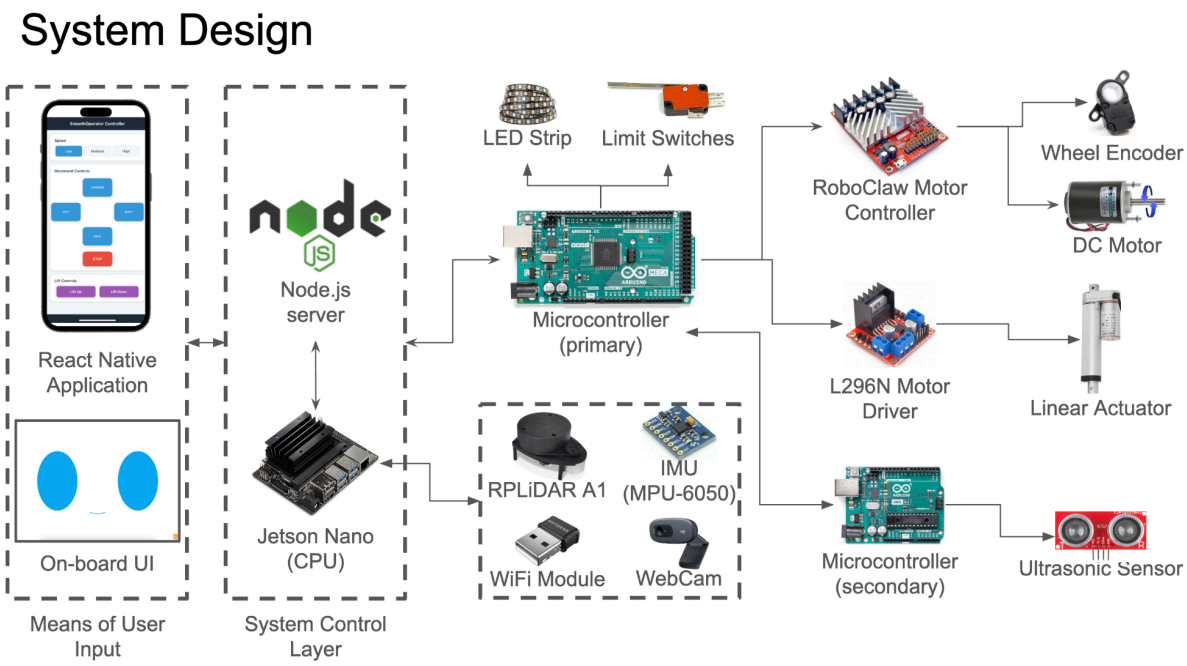


Diagram of Complete High-Level System Design.

Testing Procedure

Demonstrate End-to-End Test:

Setup

1. Open a new terminal and navigate to ~/Desktop/SmoothOperator/network/node-server
2. Run "sudo chmod 666 /dev/ttyTHS1" to adjust permissions for the UART port to enable serial communication
3. Run "node FinalDemoServer.js" to start the Node.js server
4. Verify that the server is listening on the correct port (port: 3000)
5. Open a new terminal and navigate to ~/Desktop/SmoothOperator/PythonUI
6. Run "Python3 FaceUI.py"
7. User approaches the SmoothOperator robot and clicks anywhere on SmoothOperator's face to access the Menu screen

Lifting Mechanism

8. To load the luggage, the user will slide their items over the lifting mechanism inside of SmoothOperator. Click on 'Load Luggage' and raise or lower the lifting mechanism
9. Once the luggage is properly loaded, the user clicks 'Back' to return to the Menu

On-board Manual Control

10. The user can click into the 'Manual Robot Control' page. Here the user can move the robot forward, backwards, left, and right along with 3 variable speeds

Autonomous Navigation

11. The user can click the 'Scan Boarding Pass' option on the main page. Here, the user must scan their boarding pass since this contains information about the terminal and gate number
12. Upon a successful scan, the user will be prompted to allow SmoothOperator to carry the luggage to the desired destination
    a. If yes, SmoothOperator will leave the user and move the luggage to the gate.
    b. If not, the user is returned to the Menu screen
13. After bringing the luggage to the desired destination, SmoothOperator will prompt the user to see if they're done using SmoothOperator
    a. If yes, SmoothOperator will leave the user and return to its home
    b. If not, SmoothOperator will remain with the user and bring the user to the Menu

screen

14. The user will select yes, and the SmoothOperator robot will return to its home

Obstacle Avoidance in Autonomous Navigation

15. Autonomous navigation is activated, and while SmoothOperator is traveling towards its goal. Members of the team will walk in front of the robot to show that SmoothOperator is intelligent and will respond to dynamic obstacles by determining a new path to the destination

Navigation Stack Demo:

1. Begin running the entire navigation stack
    a. "cd ~"
    b. "./runTerminals.sh" (opens multiple terminals, running all the necessary commands)
2. Open up user UI
3. Run "rosrun rviz rviz"
4. Show map and local obstacle layer updating

User Interaction Test:

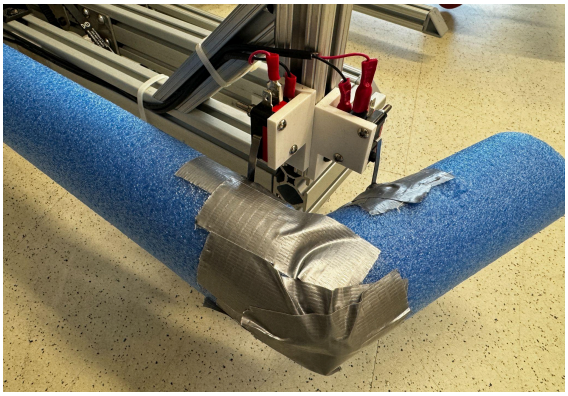
Remote Control App

1. The user will press the Connect to App button to reveal a four-digit passcode
2. On a separate device connected to BU Guest, the user will be prompted to type in a four-digit passcode. They must type the displayed four-digit passcode to ensure that the correct SmoothOperator robot is paired with the phone app. Only one person can connect to the robot at a time.
3. Upon a successful connection, the SmoothOperator on board UI will display a success screen, and the phone app will show a screen containing the control pad, variable speed, and lifting mechanism activation buttons
4. Open a new terminal and navigate to ~/UserUI
5. Run "npx expo start" to start the React Native app using Expo
6. Scan the QR code with a smartphone (Expo Go App) to launch the app
7. Show cross-platform support for the React native application on mobile device and web

8. Before switching the e-stop breaks to allow the actuators to run, ensure that the application is connected to the correct server and communicates with the server properly
9. Demonstrate UI functionality by pressing movement buttons to verify the correct LED lights responses (Green = moving; Red = stopping)
10. Demonstrate ultrasonic sensor functionality by pressing any movement button, then blocking the ultrasonic; the LEDs should change from green to red
11. Switch e-stop ON for the actuators to run
12. Show motors moving and matching corresponding color with the LEDs with manual control inputs

Hardware Test:

1. Place empty carry-on luggage (10”x22”x14”) onto the chassis platform and demonstrate that the chassis does not buckle or deflect when under load
2. Connect the two on-board batteries in parallel to the fuse box and check if all electronic indicator lights turn on
3. Make the robot traverse the room to the desired destination. Check that the robot does not use more than 18 amps at 12 volts while operating
4. Check that the stored luggage does not fall off the chassis while operating
5. Let the motors run and activate E-switch remote from 3 feet away and monitor if the motors’ power is cut off
6. Check if the chassis can stand by itself with wheels and bearings
7. Pass a piece of paper between the driving wheel and the ground and check if it can pass
8. Manually spin the driving wheels with hand and check if it can do more than 360^{o} turns supported by the bearings and custom brackets
9. Verify that the drivetrain’s gears remained wholly meshed after operation by removing the motor shrouds.




(Left) Limit switch bumper attached to pool noodle for a robust collision detection system. (Right) Camera and ultrasonic 3D printed housing for QR code scanning and obstacle detection.

Results

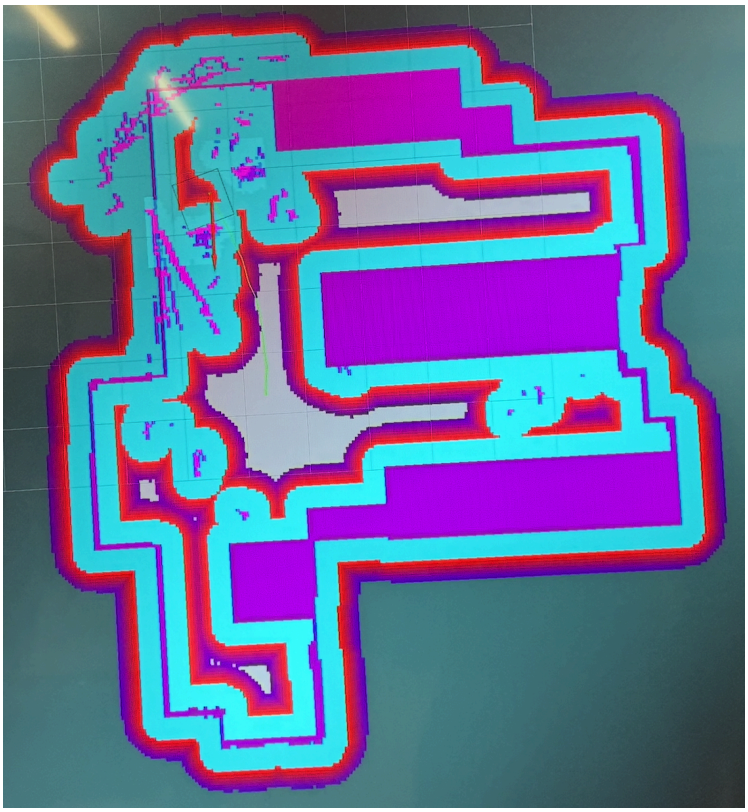
End-to-End Product Results:

Our end-to-end testing was a success, demonstrating seamless integration of all subsystems. We began the demo by walking through the user story, showing how a user would load their luggage, scan their boarding pass, and watch SmoothOperator autonomously navigate to the designated terminal while avoiding obstacles. We exceeded our MVP by accomplishing one of our reach goals: implementing a lifting mechanism to provide greater assistance and reduce strain for the user. To ensure safe navigation, SmoothOperator uses a custom-built navigation stack developed with ROS and NAV1. We created all the necessary files and nodes from scratch to support autonomous movement. While the 2D LiDAR provides primary obstacle detection, we also engineered custom physical bumpers as a secondary fail-safe, which act as a fail-safe by immediately halting all movement upon contact. We also showcased our intuitive and accessible on-screen user interface, along with our mobile app, which is compatible with any device or tablet. This provides convenient manual navigation for users or facilitators. In the end, everything worked as intended.

Navigation Results:

The navigation stack was able to seamlessly integrate all of the 11 ROS nodes together to achieve a good balance of accuracy and performance. The intelligence of our global path finding algorithm was able to recalculate the path when new environment conditions came up in a timely manner. Our local pathfinding algorithm was able to incorporate local costmap data, formed with data coming from our odometry and LiDAR, to evaluate a few timesteps ahead for each possible

velocity to determine the best fit. We defined the “best” velocity by using a weighted approach that considered factors such as adherence to the global path, obstacle avoidance based on the local costmap, and the robot’s ability to physically fit through the planned path. The combination of each of these aspects allowed SmoothOperator to dynamically avoid obstacles that it encountered during any part of the navigation process. Furthermore, the integration of the navigation stack with the user interface also proved to be successful. SmoothOperator is able to navigate to a specific location on the map when the user scans a QR code. Once the robot reaches its destination, SmoothOperator can return home upon user request by using odometry and the Monte Carlo Localization algorithm to accurately determine its current position before generating a new path back. Our navigation stack could easily allow future additions of new waypoints within our system and the generation of new environment maps.

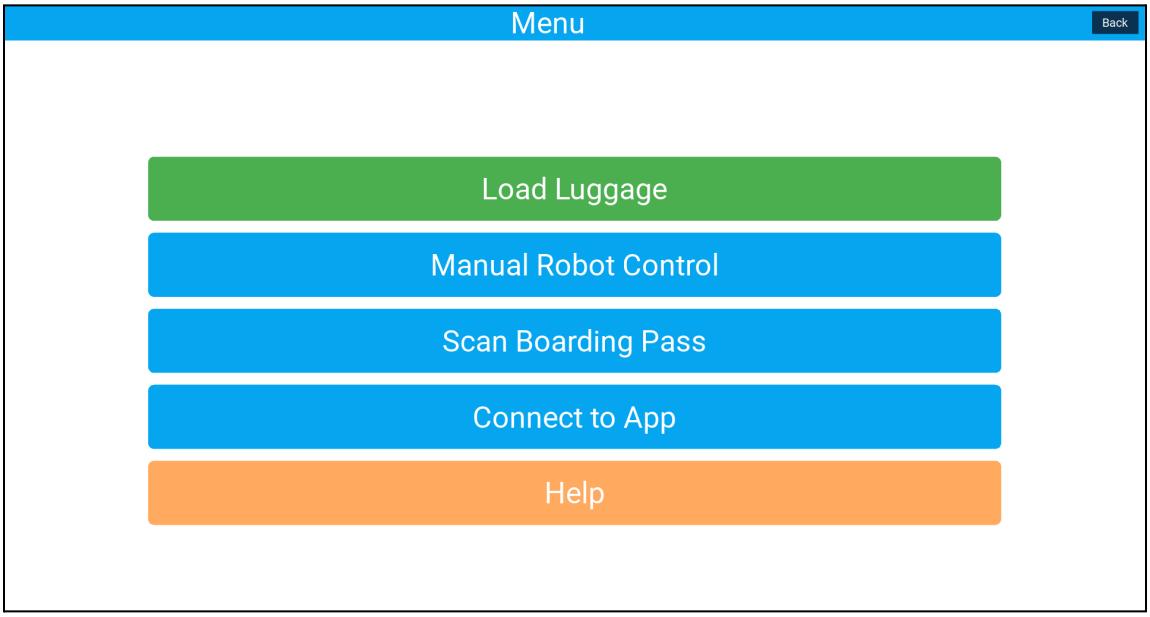


Map generated by SLAM and the overlaid costmap of the safe zones for the robot to navigate.

User Interface Test Results:

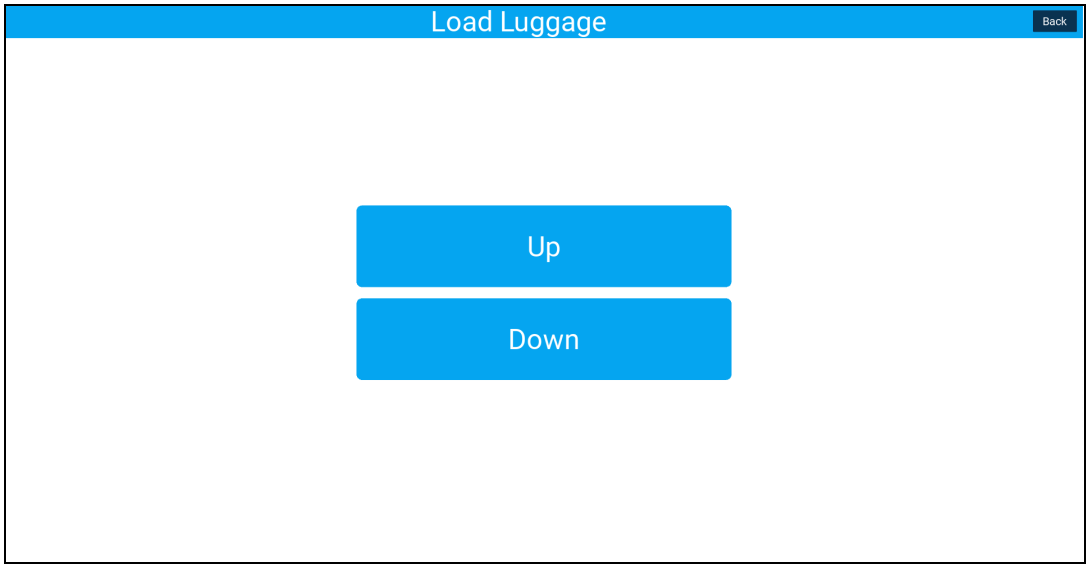
The user interface encompasses the on-board touch screen and mobile application. The on-board touch screen displays the robot's face with blue eyes and a small smile, enhancing the hospitality

of the robot. The face would react to the movements that the robot will be moving in giving real-time feedback on the direction in which it's moving in. Additionally, any audio that is said by SmoothOperator such as "Watchout! I'm coming through" or "Hi! I'm SmoothOperator" sync with the mouth on the face. To use SmoothOperator, the user can tap anywhere on the face to bring up the menu. Upon each screen change, SmoothOperator will also speak about the screen giving the user audio ques and instructions. If audible instructions are insufficient, there is also a help screen for the user to read about SmoothOperator.



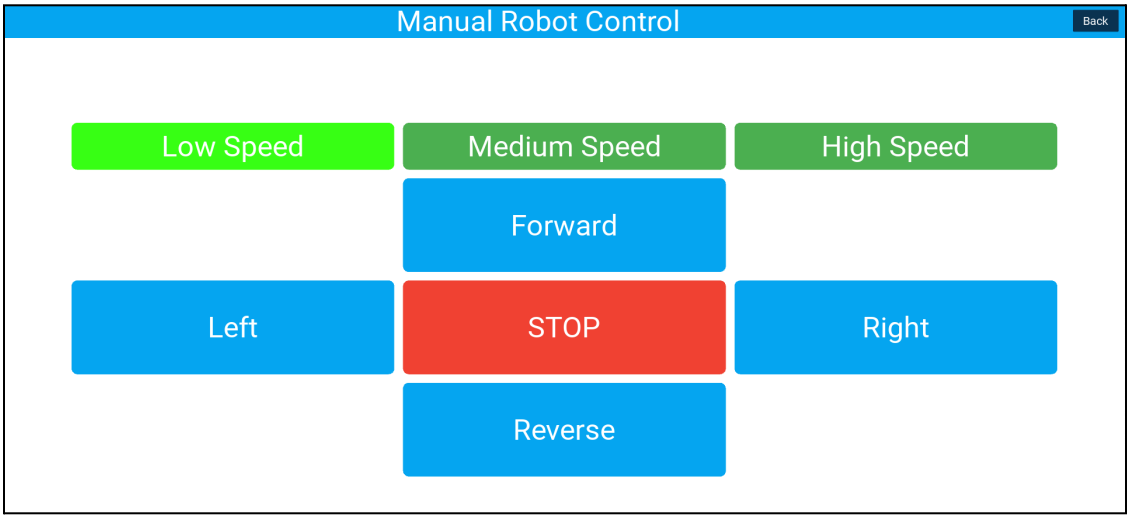
The menu screen allows the user to navigate the onboard screen.

Before operating SmoothOperator, the user must first load their luggage with our lifting mechanism. Pressing up would raise the lifting mechanism and down would lower it; the user will only need to slide their luggage in place before loading.



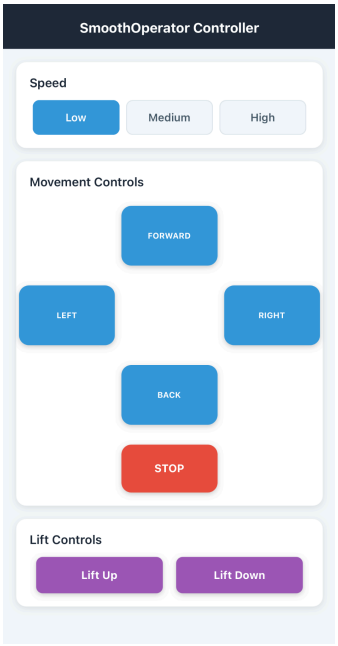
The load luggage screen allows the user to raise and lower the lifting mechanism.

As a means of input, the user can manually control the robot with the intuitive control-pad interface, with instruction provided. There are also three variable speeds allowing the user to use SmoothOperator at their preferred speed.



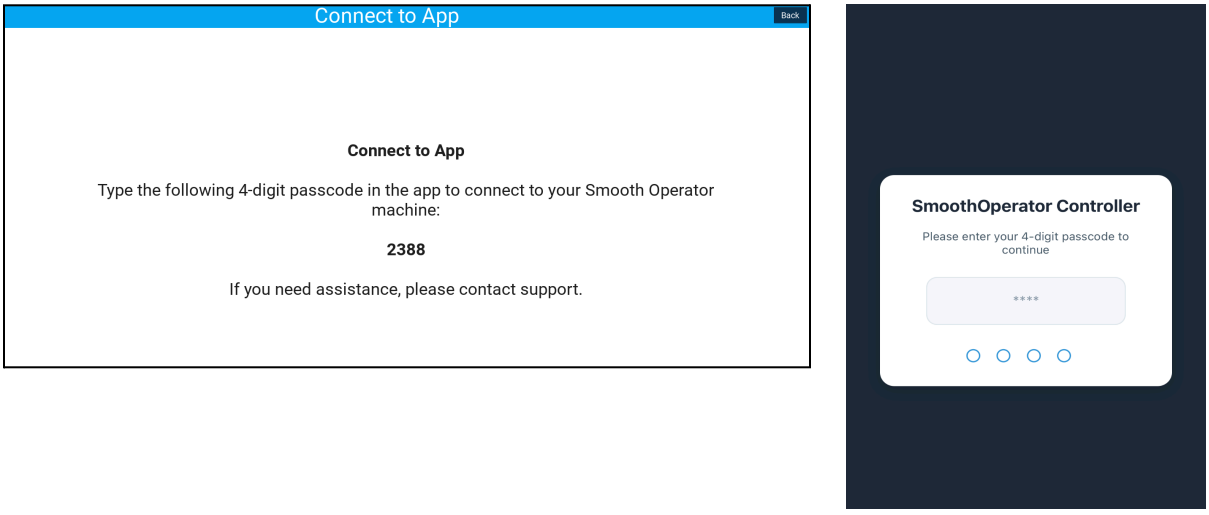
The manual control screen allows the user to move SmoothOperator.

For maintenance purposes, the SmoothOperator system comes with a phone app. This allows the airport maintenance user to control the robot from a distance.



The phone app control screen allows the user to operate the robot remotely.

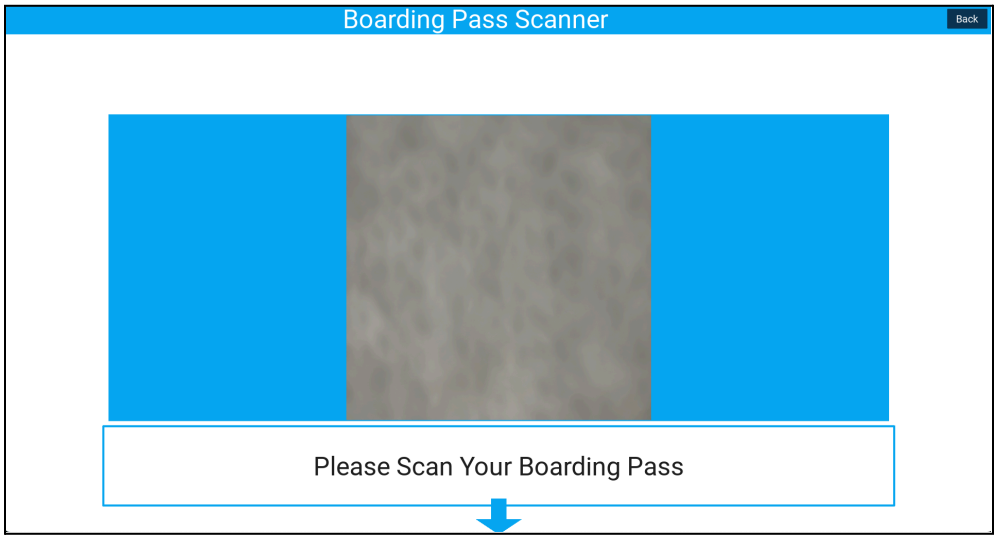
For security purposes, only one person can connect to the robot, so a four digit passcode must be entered on the app to connect.



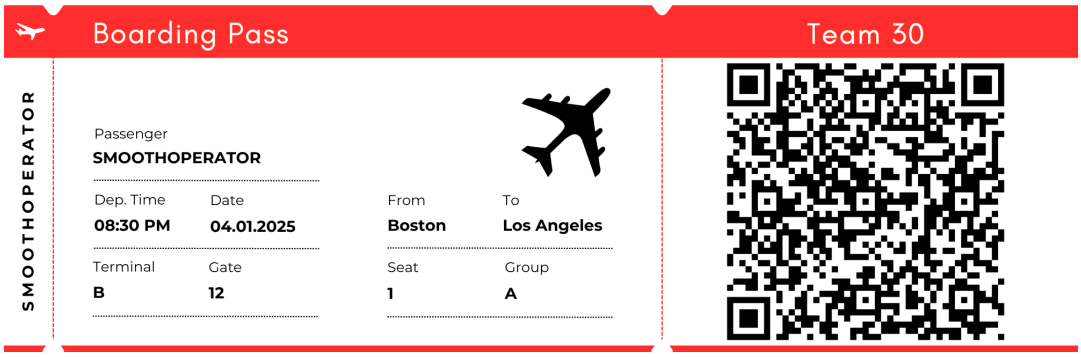
The onboard screen (left) displays a randomly generated unique passcode, and the phone app (right) displays the locked screen awaiting a correct passcode input.

After successfully typing in the correct passcode, the user will be brought to a screen allowing the same controls as the onboard screen: lifting and lowering the luggage loading mechanism,

manual movement controls, and variable speeds. If the user does not want to manually control SmoothOperator, they can opt to use the autonomous navigation feature. The autonomous navigation is a hands-off feature that enables SmoothOperator to smartly navigate the airport.

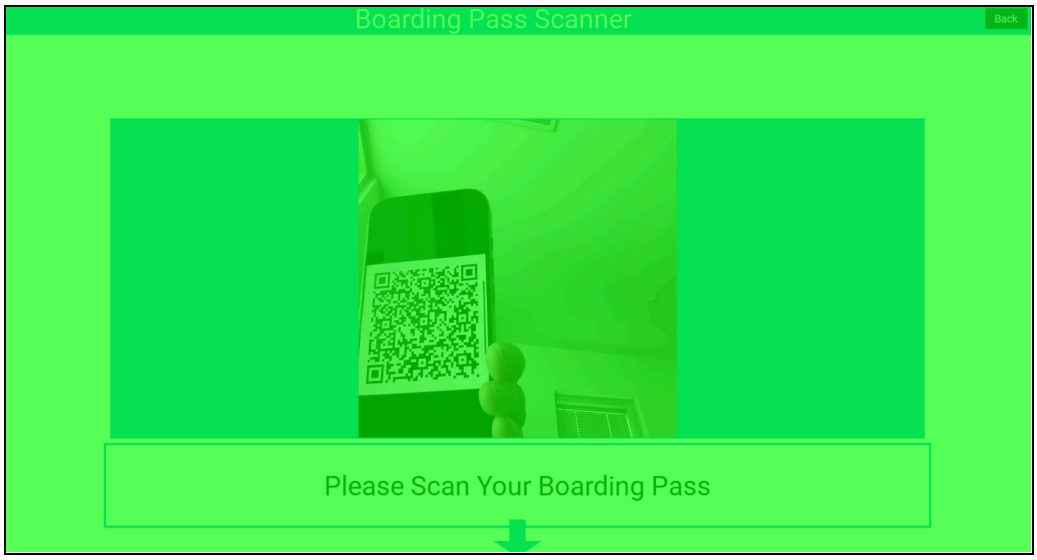


The boarding pass scan screen allows the user to scan their boarding pass.



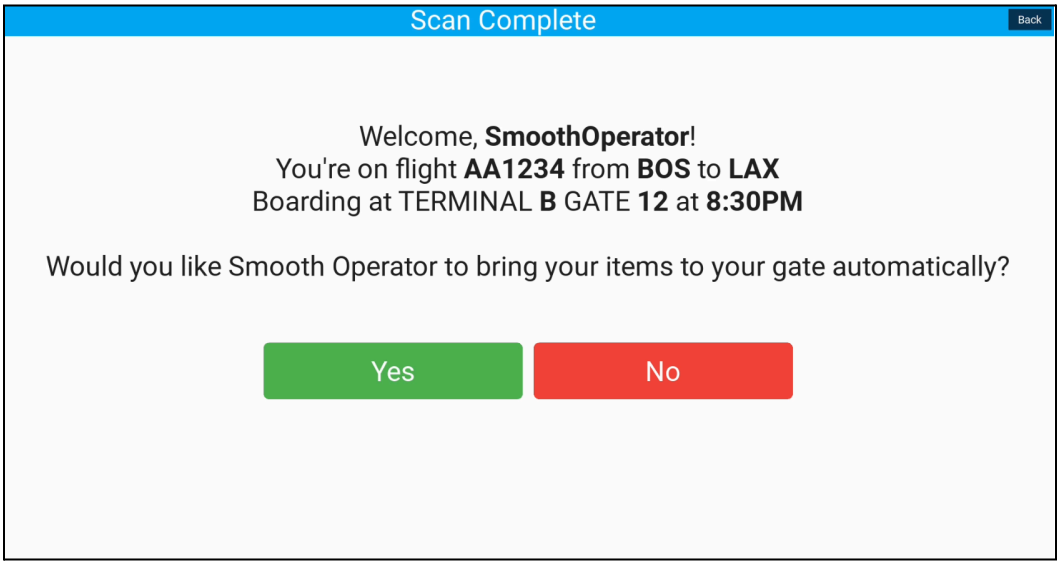
Example of QR code boarding pass with QR code we created.

To activate the autonomous navigation from the current location to the desired destination, the user must scan their boarding pass. The boarding pass will give SmoothOperator the necessary information such as terminal and gate.



The green flashing gives the user an obvious indication of a successful scan.

Upon a successful scan, the screen will flash a bright green giving the user the necessary feedback that the scan was successful, and then the user will be prompted to confirm that they want SmoothOperator to bring their items to their airport gate and terminal. SmoothOperator will also speak and welcome the user giving the user a nice hospitality experience.

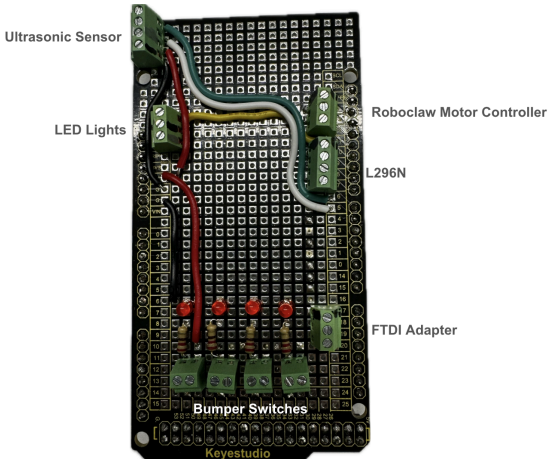


The confirmation screen ensures that the user wants SmoothOperator to navigate autonomously.

If the user clicks yes, SmoothOperator will be on its way to deliver luggage to the destination! Upon clicking no, SmoothOperator will return back to the menu screen.

Hardware Test Results:

The robotic system was designed with a redundant safety hardware layer to ensure robust fault tolerance. Critical proximity sensing, comprising limit switches and ultrasonic rangefinders, operates independently on an Arduino Mega microcontroller responsible for direct motor actuation. Navigation commands are generated by an onboard Jetson Nano and transmitted serially to the Arduino Mega, where they are decoded into motor control signals. Crucially, the Arduino firmware incorporates a hardware-level override mechanism that autonomously supersedes navigation commands when critical proximity thresholds are breached, forming a last-line safety defense. Structural load testing demonstrated the robot's chassis can sustain 80 pounds—representing a 60% surplus beyond the original 50-pound design specification—ensuring a substantial safety margin. During peak load conditions with all computational and mechanical processes active, the system's electrical draw remained within 12V and 18A, confirming compliance with power budgets. Overcurrent protection was integrated via a dedicated fused breaker system; any subsystem exceeding its current threshold results in an immediate, isolated shutdown, safeguarding the remainder of the architecture.



Arduino Mega PCB Board Shield with Screw Terminals for more Reliable Connections and Modularity in Debugging.

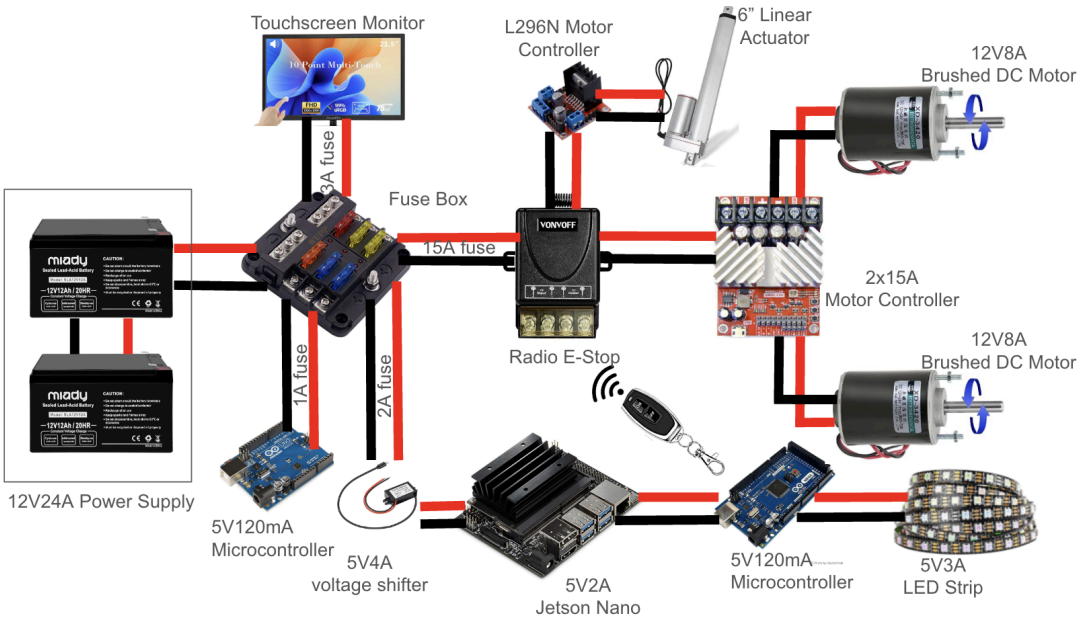


Diagram of Power Distribution from the 2 Lead-Acid Batteries in Parallel Batteries.

Mobility is fully autonomous and untethered, enabled by dual 12V 12A lead-acid batteries with total weight and center-of-gravity impacts accounted for in the drivetrain and motor torque calculations. Embedded diagnostic systems were integrated across the platform, including distributed power and signal indicators for real-time telemetry of power health, data transmission, actuator status, and logic anomalies, significantly enhancing debugging and maintenance workflows. The robot also incorporates a dedicated radio-frequency emergency stop (e-stop) system, leveraging an onboard relay-triggered kill circuit. The e-stop can reliably deactivate all drive and actuation systems within a 3-foot operational radius—extendable up to 10 feet in ideal line-of-sight conditions—thus providing a rapid-response hardware interlock to arrest runaway behaviors and prevent system or environmental damage. All emergency circuit designs ensure fail-safe de-energization without compromising the integrity of the logic subsystems.

Mechanical Test Results:

The drivetrain and lifting mechanism operated successfully during testing. The metal frame made of 80/20 aluminum extrusions was assembled securely and did not deform or otherwise fail during testing. The team meticulously manufactured each part using a wide range of machining techniques, such as water jet cutting, 3D printing, milling, drill pressing, laser cutting, band

sawing, and cold sawing. As a result, the robot frame can hold a luggage of up to 50 pounds without any buckling or deformation occurring. Even with a sturdy frame, SmoothOperator can operate at a top speed of 1.2 m/s which is the average human walking speed. To ensure that the wheels were mounted at an appropriate height, custom wheel mounts were water jetted from metal allowing the robot to stand at a perfect height. The pulleys for the compound gear train were 3D printed in PLA and the gears for the gear train were printed in ABS. These printed parts successfully held their shape and remained wholly meshed during testing operation, demonstrating their high durability and accuracy.

Conclusion

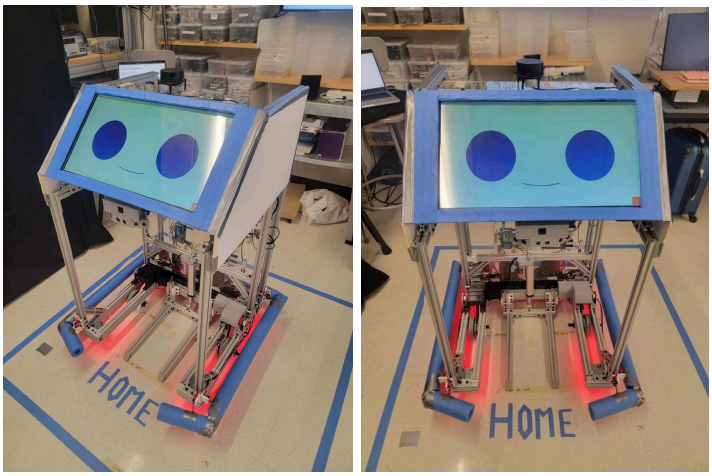
For our senior design project, we successfully completed the development of a full-scale (2'x2'x3') end-to-end teleoperated and fully autonomous robotic luggage assistant capable of dynamic obstacle avoidance and seamless terminal delivery. Every aspect of the system—from hardware to software—was built entirely from scratch and tightly integrated to ensure robust performance. We designed and fabricated the mechanical framework in-house, utilizing 3D printing for motor shrouds and custom gears, as well as precision water jetting to manufacture custom aluminum brackets to optimize drivetrain geometry, power efficiency, and structural integrity. A custom wireless communication network was developed to enable real-time interaction with the robot, and GPU acceleration was leveraged for onboard parallel processing of graphics and navigation tasks. We engineered a complete navigation stack capable of autonomously delivering heavy luggage to designated terminals, featuring dynamic rerouting in response to moving obstacles. To enhance user accessibility, we fully integrated a boarding pass scanning module into the system and onboard touch screen UI, enabling intuitive human-robot interaction.

Safety was addressed holistically across hardware, software, and user interfaces: redundant sensor architectures safeguarded against hardware failures while dynamic obstacle avoidance algorithms mitigated collision risks during autonomous navigation. Additional safety features were incorporated visually and auditorily through a custom-designed touch screen facial UI, enabling both informative and emergency interactions.

The robot was entirely self-contained, powered by an onboard energy system, requiring no

tethering to external infrastructure for operation. For mapping and localization, we implemented a custom SLAM (Simultaneous Localization and Mapping) solution that allowed full-room mapping with human-annotated restricted zones, eliminating the need for airports or hotels to install external localization beacons—an ongoing challenge in robotics research that we were proud to address successfully.

While the system met all major functional and safety requirements, it exhibited a known limitation: a blind spot below the 2D LiDAR sensor. We pragmatically addressed this constraint with a lightweight bumper system fabricated from pool noodles and integrated limit switches and a suite of ultrasonic sensors. Although a 3D LiDAR would have resolved this issue more elegantly, cost constraints precluded its integration. Despite these challenges, we are proud of the fully autonomous, mobile robotic system we created, and we believe it lays a strong foundation for future work. Moving forward, we plan to design and fabricate a clear acrylic enclosure for the robot, providing both security for the luggage and visibility for users. Additionally, enhancements such as 3D LiDAR integration, improved semantic mapping, and tighter human-robot collaboration frameworks could further elevate the system toward real-world deployment. These areas remain active fields of research, as reliable 3D perception, contextual scene understanding, and intuitive human-robot interaction continue to face challenges related to cost, computational demands, and system robustness. As these technologies mature, autonomous service robots like ours will become increasingly capable of operating safely and intelligently in dynamic, human-centered environments.



Different angles of the complete system at its homing stage.

Measurable Criteria

- ☐ Luggage fits in the chassis
- ☐ Electronics does not use more than 12v 24A DC
- ☐ Touchscreen UI reacts to user inputs
    - ☐ Change between tabs
    - ☐ Teleoperation of Chassis
    - ☐ Changes between 3 preset speeds
    - ☐ Talks about instructions
- ☐ Phone UI interacts with robot and user input
    - ☐ Password Access Teleoperation Controller
    - ☐ Controls the robot forward, backward, and 0-degree left and right turns.
    - ☐ Changes between 3 preset speeds
- ☐ Indicator LED lights reflect stops (red) and moving (green)
- ☐ Floating bumper stops the robot from all movement
- ☐ Motor shrouds protect external debris from vertical
- ☐ Radio E-stop stops actuator system with remote
- ☐ Footprint of the robot moves 20 pixels within RViz when the robot moves in 1 meter
- ☐ Navigation stack generates new paths when it encounters an obstacle in the local window
- ☐ Navigates from point (X_1, Y_1, Θ_1) to ($X_1 \pm 0.2$m, $Y_1 \pm 0.2$m, $\Theta_1 \pm 0.05$rad)
- ☐ Robot moves at a max speed of 1.2 m/s
- ☐ Chassis can sustain $\leq$ 50 LB
- ☐ PID left and right speed control for straight driving

Score Sheet

Criteria	Point
Demo	8/**8**
User able to load luggage and robot lifts luggage off the ground	1/1
User is able to scan the boarding pass and robot decodes and sets the destination	1/1
Robot moves autonomously to the set location by boarding pass	1/1
User is able to unload luggage at final destination	1/1
Manually adjust the position of the robot with teleoperation control UI	1/1
When floating bumpers are hit, the robot does a full stop	1/1
Navigates home autonomously	1/1
Obstacle avoidance	1/1
Mechanical Design	4/**4**
Frame can support a 50 lb luggage	1/1
Frame can fit a standard carry-on luggage	1/1
Top speed of 1.2 m/s	1/1
Motor shrouds protects gears and allows for top speed movement	1/1
Hardware	**4/4**
Electronics does not exceed 12v and 22A	1/1
Motors move with respect to the manual controls sent	1/1
E-Stop reacts to remote switch from at least 3 ft. away, shutting down actuators	1/1
Does not use a wall outlet and is powered by onboard batteries	1/1
Total Score:	16/**16**

Hardware Pinout

Pin	Usage/Description
Microcontroller (Arduino Mega)	
5v	Ultrasonic
TX	UART RX of Jetson (Serial communication) for movement commands
RX	UART TX of Jetson (Serial communication) for movement commands
TX1	UART RX of Jetson (Serial communication) using FTDI USB adapter for encoder data
RX1	UART TX of Jetson (Serial communication) using FTDI USB adapter for encoder data
5	Ultrasonic sensor data from Arduino Uno pin 13
11	LED Strip Signal Pin
12	Serial communication via 'SoftwareSerial' for motor controller. Controls motor speed and reads encoder data.
13	Serial communication via 'SoftwareSerial' for motor controller. Controls motor speed and reads encoder data.
7	ENA of L296N Motor Controller
8	IN1 of L296N Motor Controller
9	IN2 of L296N Motor Controller
32	Right Bumper Switch
38	Left Bumper Switch
44	Back Bumper Switch
50	Front Bumper Switch
GND	Ground of Power Supply (Miady 12V 12Ah/20HR), Jetson Nano, L296N motor controller, Roboclaw Motor Controller 2x15, Arduino Uno, LED Lights, and Bumper Switches
Microcontroller (Arduino Uno)	
5v	Ultrasonic Sensors

Barrel Jack	12V 1A from Fuse Box
13	Output signal connected to Arduino Mega Pin 11
4, 5	(Trigger Pin, Echo Pin) Ultrasonic 1
6, 7	(Trigger Pin, Echo Pin) Ultrasonic 2
8, 9	(Trigger Pin, Echo Pin) Ultrasonic 3
GND	Arduino Mega, Ultrasonic Sensors
NVIDIA Jetson Nano	
Barrel Jack	5v 4A Power Supply
3	IMU SDA
5	IMU SCL
5V	IMU Vin
GND	IMU GND
Arduino Mega USB1	For sending movement commands to the Arduino Mega to control all actuators and lights
Arduino Mega USB2	Reading encoder data forwards the front motor controller to the Arduino Mega. Connected via FTDI USB Adapter.
LiDAR	Used for navigation and mapping
USB 3.0 Hub	Extends ports and is used by low priority data communication. Touchscreen, wifi module, keyboard/mouse (for debugging), and camera are connected.
GND	Ground of Power Supply, IMU, and Arduinos, and all corresponding USB ports.
RoboClaw 15x2 Motor Controller	
S1	Digital Pin 13 of Arduino
S2	Digital Pin 12 of Arduino
GND	Ground of Arduino Mega
L296N Motor Controller	

IN1	Digital Pin 8 of Arduino. Controls Clockwise rotation
IN2	Digital Pin 9 of Arduino. Controls Counter Clockwise rotation
ENA	Digital Pin 7 of Arduino. Controls PWM
Vin	12V 2A Power Supply
Vout	Power line of 6in Linear Actuator in Lifting Mechanism
GND	Ground of Arduino Mega and Power Supply
MPU6050 IMU	
GND	Ground of Jetson Nano
Vin	5V of Jetson Nano
SDA	Pin 3 of Jetson Nano
SCL	Pin 5 of Jetson Nano