Conceptual Design Report

SSG
MOBILE ROBOTICS CO.

Prepared by Group 10

Sebastien Sauter, Jamie Strain, and Marlow Gaddes

All work in the report is by the listed authors with the exception of properly cited material.

March 2, 2025

1.0 Executive Summary

According to The International Civil Aviation Organization, as stated in 2024, the demand for air transport will increase by an average of 4.3% per year [1]. Hence, as airport congestion increases, the efficiency of airport operations must increase to reduce wait times and promote safer travel. Mobile robot delivery systems can increase the efficiency of operations, as seen by the implementation of Demantic's robots in a new Asahi factory [2]. Therefore, this conceptual design introduces a semi-autonomous mobile robot that can build a static environment map, determine its position, and navigate with dynamic obstacles. This robot aims to improve the efficiency of delivery operations in an airport environment by reducing traffic.

The conceptual design presents a robust framework for a semi-autonomous mobile robot, outlining the hardware and software systems and their integration. The hardware architecture demonstrates a robot operating system (ROS2 Humble Hawksbill) that communicates with an Arduino to read sensor data and transmit velocity data so that the Arduino can drive the motors. This dual-processor configuration ensures high-level navigation tasks, including global path planning using the A* algorithm, local path planning from the dynamic window algorithmic approach, and localization using SLAM, are handled by the more powerful Raspberry Pi, while the Arduino provides real-time control over wheel encoder, IMU data, and motor operations. Offloading tasks between processors allows the robot to respond quickly and accurately to a dynamic environment.

The robot software system integrates several key processes: mapping and localization, where sensor data is fused to build and update an internal map; state estimation uses techniques such as extended Kalman filtering to refine the robot's position and orientation; goal computation calculates an optimal route using global path planning algorithms; and the output execution involves local path planning and PID to operate motors at a desired output velocity. These elements work in tandem within the ROS2 ecosystem to ensure the robot can adapt to its route in real-time, avoid obstacles, and reliably deliver supplies.

The planned hardware layout and software architectures enable this semi-autonomous mobile robot to meet delivery demands in dynamic environments. The complementary roles of the ROS and the Arduino allow for efficient data handling and rapid motor control. The software packages offer a comprehensive approach to complex mapping, navigation, and obstacle avoidance models. This design provides a flexible and scalable solution that applies to delivery applications.

Table of Contents

List of Figures

List of Tables

2.0 Conceptual Solution

SSG Robotics proposes a semi-autonomous mobile robot equipped to handle deliveries of goods in an airport environment upon user request. The proposed solution provides software and hardware solutions for robotics specialized in an airport environment. An airport provides a complex dynamic environment with high traffic, however, also provides simplicity in its floor plan layouts and accessibility. The airport dynamic allows for SSG Robotics to specialize in developing high quality indoor navigation and obstacle avoidance systems.

The robot's hardware architecture will feature a range of interoceptive and exteroceptive sensors combined with reliable mechanical components to facilitate an intricate software design. Interoceptive sensors include wheel encoders and an internal measurement unit, whilst the exteroceptive sensors include infrared range sensors and LiDAR. The robot will feature industry standard chassis and wheel design. The payload is set to be a latched durable polycarbonate design. Finally, the robot will feature lights and speakers to help navigate through the densest parts of the airport.

The core design elements of the mobile robot lie in the software design rather than the mechanical or hardware design. SSG Robotics is proposing a full software stack that can be used for a variety of mobile robots within the airport setting. Although the mechanical design limits the use of the robot to simple transportation of goods, the software design allows for the implementation of many different mobile robots each with their own delivery purposes. Future considerations for this technology can include autonomous luggage transportation, autonomous wheelchair applications and potentially assisting visitors with directions to their gate or to baggage claim.

2.1 Interoceptive Sensors

Interoceptive sensors are crucial to determine the robot's localization and odometry data. The odometry data consists of position, velocity, and orientation metrics. The wheel encoders and IMU provide the robot with data on how its moving through its environment and not how its environment is moving through it.

2.1.1 Wheel Encoders

The mobile robot features two-wheel encoders on both left and right-side wheels. Wheel encoders use an overlay of two PWM signals to determine the magnitude of the spin (encoder ticks per second) and direction (forwards or backwards). Equation **1** is used to calculate the estimate for the rotational speed of the wheels, in (rad/s), where TPR is ticks per revolution and T is the period.

$$\widehat{\omega}_L = 2\pi * (\#of\ encoder\ ticks/\ TPR) * 1000/T \qquad (1)$$

From the rotational speed of the wheels, we can determine tangential speed by multiplying by the wheel radius. With the tangential speeds of the right and left wheel separately calculating the average results in the overall tangential speed. Calculating the difference between the left and right wheels and dividing by the distance between the wheels yields the angular velocity of the robot. However, the above computations only work under a no slip condition. Therefore, without the wheels being able to steer, the wheel encoders are only accurate in representing translational speed for straight line motion. Therefore, the robot requires the use on an Internal Measurement Unit (IMU) for better interoceptive sensing while turning.

2.1.2 Internal Measurement Unit (IMU)

An IMU provides the 6 degree of freedom measurements of accelerometer data in x, y, and z directions as well as gyroscope data for all three corresponding rotational axes. This accelerometer and gyroscope data can then be in principle be integrated to determine positional, velocity, and orientation data. However, in practice the raw form of IMU data is extremely noisy and must be passed through a series of transforms, filters, and integrations before it can be useful for any software system. The IMU data can also be fused with wheel encoder data to provide a more complete picture on the vehicle odometry. This result is achieved through the Extended Kalman Filter which is detailed in 2.5.3 Localization and State Estimation.

2.2 Exteroceptive Sensors

Obstacle avoidance is necessary for this robot, allowing it to safely operate in public spaces. To do this, exteroceptive sensors, including range sensors and LiDAR, are being integrated into the robot's design. These sensors will enable the robot to detect, map, and respond to its surroundings, giving it the ability to maneuver around obstacles safely and efficiently. The LiDAR will allow the robot to map the environment around it, while the range sensor will provide close-range collision prevention. Together the sensors will build a hardware base to program a strong obstacle avoidance system.

2.2.1 Range Sensors

Three sharp IR range sensors have been equipped to the robot to aid in close range obstacle detection. This experiment aims to determine the operating range for these sensors. The data is plotted in Figure **1** to determine an equation for calculating the distance. The equation of best fit provides a useful estimation for distances between 10 and 80 cm. For this reason, they will be implemented as an emergency stop function.

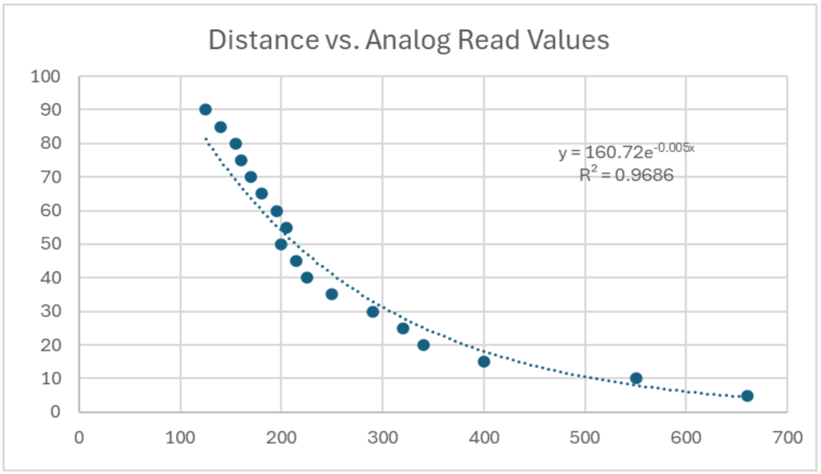

Figure 1: Plotted data from sharp IR range senor testing

2.2.2 LiDAR

The robot will also be equipped with a RPLiDAR A1M8 360-degree laser scanner to handle navigation. This will let the robot map its surroundings and navigate around obstacles in the area in real time. LiDAR documentation from SLAMTEC [3], as shown in Figure **2**, demonstrates the mapping capabilities of LiDAR in adjunction with ROS. It has a detection range of 0.15m to 6m, a sampling rate of 8,000 points per second, and a rotational speed of up to 10 Hz meaning it can rotate up to 10 times a second [3]. It provides high-resolution mapping within this range making it perfect for this robot's application. It has an error of ±1%, ensuring reliable object detection.

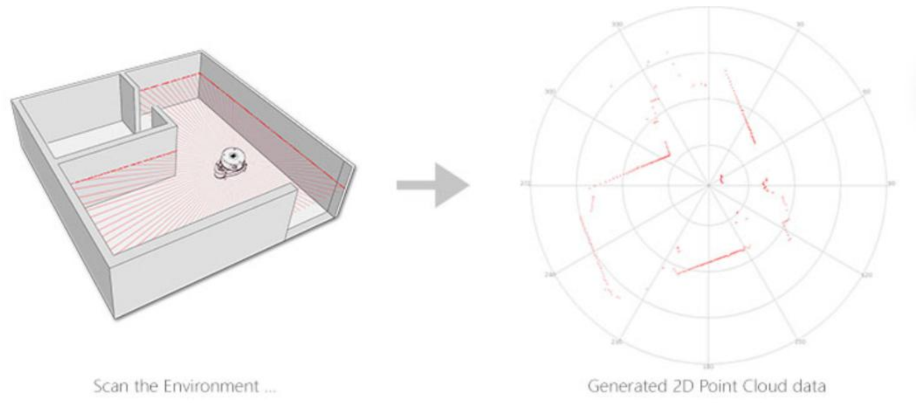

Figure 2: Example mapping done by LiDAR

2.3 Core Systems

2.3.1 Proportional Integral Derivative (PID) Control

The PID Control section discusses the use case of PID, demonstrates performance, and examines the integration with ROS2.

A PID controller is a closed loop controller used to drive robots given a desired input speed. The controller determines the output based on three parameters the proportional, integral, and derivative errors. The error is determined simply by the difference between the measured and desired velocities. Each error term is multiplied by a constant which is found by tuning the PID system. The system in place on the Lynxmotion Rover uses only the proportional and integral (PI) control as this system provides high accuracy and reduced computational complexity.

Figure 3 demonstrates the step response of the PI controller to determine the accuracy by comparing the measured velocity (orange) against the desired velocity (navy). This plot illustrates effective implementation of integral control as the measured velocity recovers after overshooting the desired velocity. Additionally, Equation **2** is used to determine the error of the PI controller. The average measured velocity is determined using experimental values.

$$\%\ Error = \frac{V_{desired} - V_{avg}}{V_{avg}} * 100 = \frac{0.7 - 0.651}{0.651} * 100 = 7.53\% \qquad (2)$$

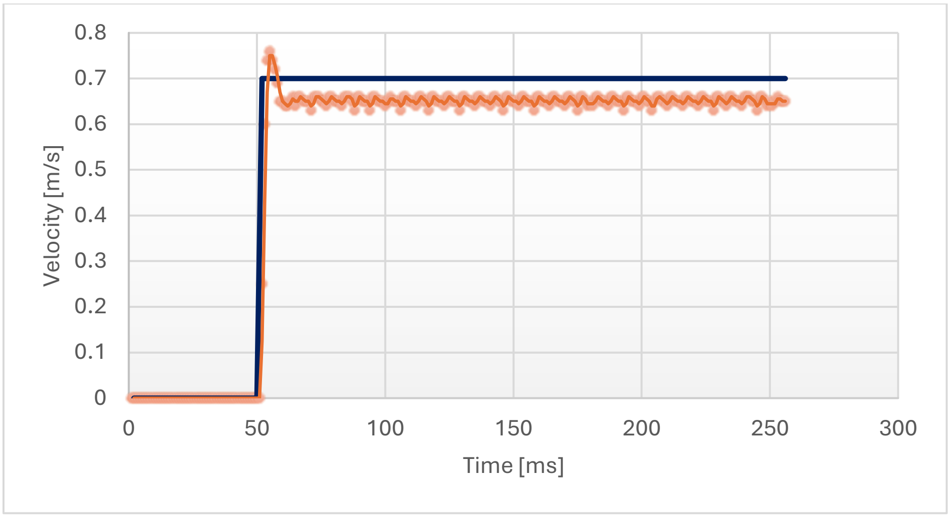

Figure 3: Unit Step Response of PI Controller

Therefore, the measured velocity is fairly accurate without sacrificing additional memory and computational complexity. Additionally, further tuning using a smaller sample rate will allow for even greater accuracy. Hence, SSG Robotics has decided to implement the simpler PI controller.

Figure 4, demonstrates the corrective response of the PI controller when subject to a pulsing input velocity. This figure highlights the delay for the robot to reduce its velocity to zero. The decrease in speed is gradual as the initial kinetic energy of the wheels decrease. It is important to note during this experiment the wheels were not in contact with any surface, hence when the wheels are in contact with a surface due to greater friction the wheels will stop sooner. This plot illustrates an important conclusion for edge cases during operation where the robot must stop immediately. Hence, additional safety considerations must be put in place to prevent these hazards. Further discussion on implementation of safety features is mentioned in section 2.4 Mechanical Components.

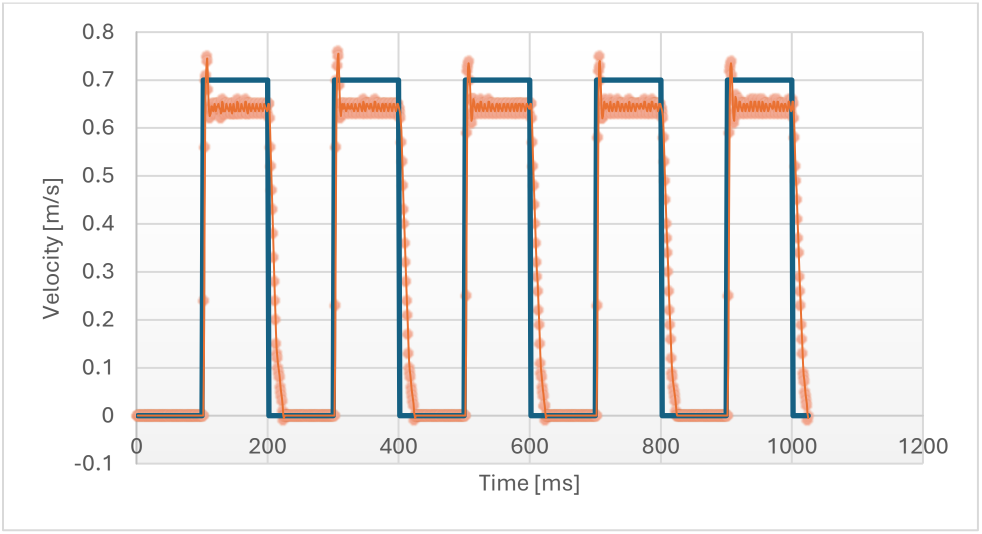

Figure 4: Pulse Response of PI Controller

The PI controller must be able to communicate with the ROS2 to receive a desired input speed and turning rate, hence, there must be a way to communicate between systems. Section 2.5.1 Hardware Integration discusses in further detail the communication between hardware systems; hence, the focus is on software integration. As discussed in section 2.5.5 Output Execution, the Arduino receives a desired velocity from ROS and presents this data as input to the PI controller. Subsequently, the PI controller will present the appropriate pulse-width-modulation (PWM) signal to the motor driver board to drive the wheels. Hence, this process continues as the navigation stack collects new data and computes a new desired velocity.

2.3.2 Navigation Stack (Nav2)

The navigation stack describes the software structure of the specifically navigation strategy that the robot seeks to implement. Nav2 [4] is the industry standard mobile robot navigation system and specializes in surface robotics. Nav2 computes global and local cost maps that are passed on to the global and local planners to determine the next movements of the robot. The expected inputs are TF transformations (localization), a static map, a behaviour tree (logic loop), and sensor inputs. When properly configured Nav2 outputs motor commands for the proposed non-holonomic robot.

Nav2 provides the tools for global path finding and for local dynamic obstacle avoidance. The team has isolated the A* algorithm for global path planning and a Dynamic Window Approach (DWA) for the local planning. The A* algorithm is a widely used low computation heuristic algorithm used to find the shortest path between two points on a map. The DWA calculates an optimal collision free velocity from a search space constructed from a set of velocities which produce a safe trajectory. The optimal velocity is selected based on clearance, magnitude of the velocity, and heading towards the global goal [5].

A study conducted by Handan University on robot avoidance optimization demonstrated positive results in complex changing environments for a hybrid A*-DWA algorithm [6]. The study observed reduced path length and reduced path planning time compared to traditional algorithms specifically for environments where dynamic obstacle avoidance is critical.

2.4 Mechanical Components

For added safety, the robot will be equipped with a signal light and speaker system to notify surrounding people of its presence and movement. The light will flash continuously, and the speaker will periodically emit sound to alert those nearby. This will ensure improved safety in a busy environment, reducing the risk of any accidents. Additionally, a polycarbonate payload compartment with an electric lock will be integrated to store and transport user content. The electric lock provides controlled access, ensuring that only authorized users can retrieve the stored items, protecting against any tampering to the users' contents. This combination of audio and visual alerts, along with secure storage, improves the robot's overall safety for its users.

2.5 Hardware and Software Integration Plan

The Integration Plan provides an overview of the hardware and software systems for a mobile robot application as depicted in section 2.0 Conceptual Solution. First is a hardware discussion on processing, communication, sensor integration, and data transfer. Followed by a software discussion on mapping, localization, and navigation.

2.5.1 Hardware Integration

The processors onboard the Lynxmotion Rover include the Arduino and Raspberry Pi. The desired use of the Raspberry Pi is to run navigation packages, filter odometry data, and collect LiDAR data. These navigation packages such as SLAM, neo-localization, and path planning algorithms are high level processes. In comparison the Arduino computes sensor data and drives the rover using a PI controller. The Arduino is incorporated in this design to compute low level processes to allow the Raspberry Pi to process high level processes quickly. The Raspberry Pi is also included to expand the rover's memory capabilities.

The Arduino and Raspberry Pi receive and transmit data over serial communication. In hardware, the Pi and Arduino are connected directly connected via their USB ports. The Arduino supports communication with the Raspberry Pi via Arduino libraries.

The sensor integration strategy is illustrated in Figure 5 below. This flow chart describes the type of data transfer and method for communication. The wheel encoder, IMU, motor drivers, and range sensors are connected to the Arduino via general input output (GPIO). This data is collected on the Arduino and transmitted via serial to the Raspberry Pi. The lights and Speakers are used as an additional form of communication between the rover and inertial observers, these will also be connected to the Arduino via GPIO. The LiDAR connects to the Raspberry Pi via transistor-transistor logic (TTL). Lastly, to determine the goal, as discussed in 2.5.4 Goal Computation, and show the status of delivery to the buyer a server must hold both the GPS data for the position of the rover and buyer. Hence, the Raspberry Pi must be connected to this server to transmit and receive the location of the rover and buyer respectively. Hence, the sensors and processors must all work as one to achieve the desired outcome.

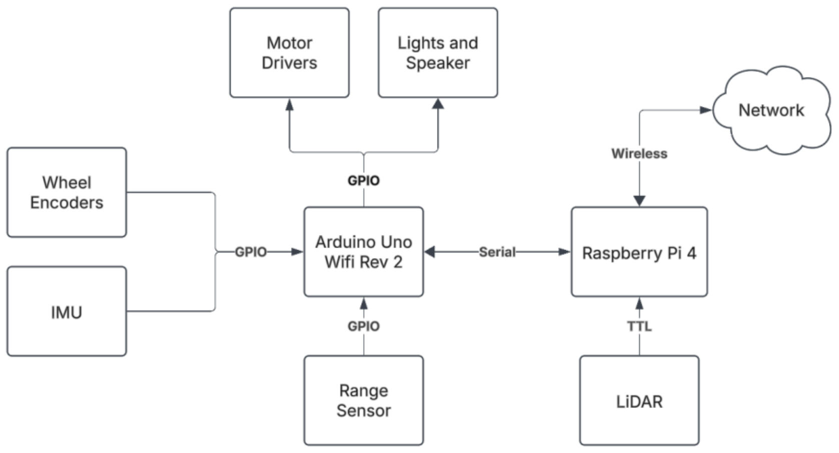

Figure 5: Hardware Systems Diagram

Figure 6 represents the proposed software stack for the mobile robot. The robot will use ROS 2 as the middleware due to it's open-source and robust well tested frameworks and packages. ROS 2 follows a subscriber-publisher communication model between its nodes. In Figure 6, the flow of data and communication is indicated by the direction of the arrows.

The core component of the software stack is the navigation stack (Nav2), previously described in 2.3.2 Navigation Stack (Nav2). The objective the software stack is to process an overall map of its environment, a variety of interoceptive and exteroceptive sensor inputs, and the destination/goal to compute a desired output in the form of a translational speed and angular velocity.

The high-level software stack should be considered separate from the robot's physical design, as its main value lies in providing a general software solution for any mobile robot design operating within an airport environment.

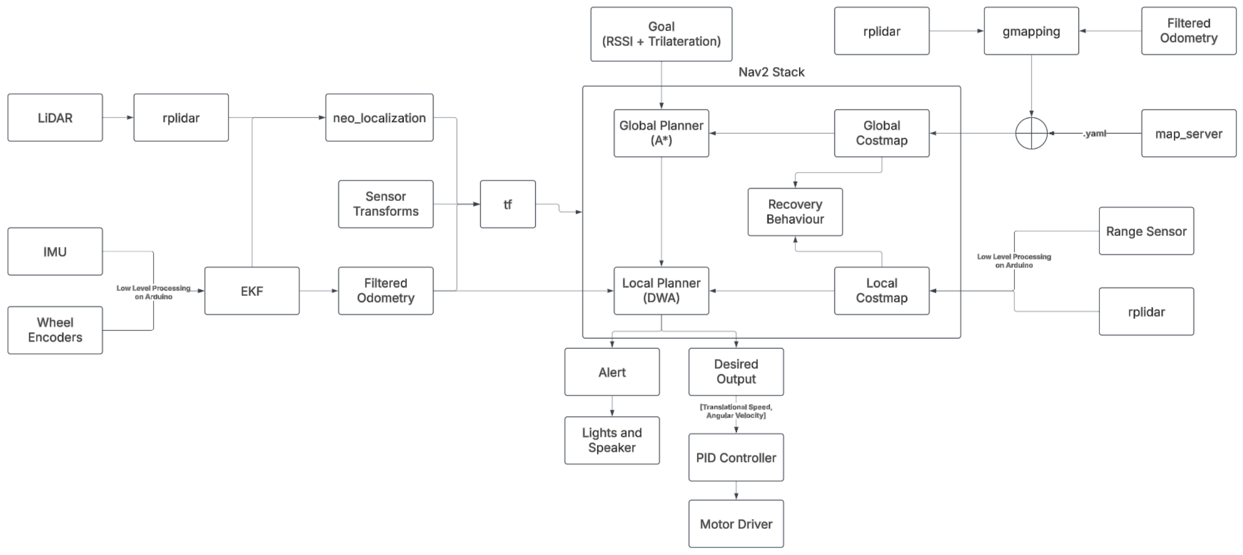

Figure 6: Software Stack - Systems Diagram [7]

2.5.2 Mapping

The Nav2 stack requires a map of the environment to compute it's global costmap. This can be obtained in two different ways,

1. Using the gmapping package [8] to run Simultaneous Localization and Mapping (SLAM).
2. Using the map_server package [9] to load a fixed pre-determined map to global costmap.

The robot must build its own map using SLAM in the initial mapping phase. The generated map created from scanning and localizing itself within the building can then be used as the map_server on YAML format. The map_server is the viable long-term solution because the robot continuously operates in a known environment.

2.5.3 Localization and State Estimation

Now that the robot understands it's environment it needs to be able to place itself and understand its movement within the environment. This is achieved through two main packages:

1. The neo_localization package [10] is the successor to the Adaptive Monte Carlo Localisation (amcl) package [11] which uses a particle filter to track the position of a robot against a known map (map_server). The neo_localization package seeks to achieve a higher accuracy as well as more robustness in challenging environments compared to its predecessor.
2. The tf package [12] allows the robot to keep track of a tree of multiple coordinate frames over time, thus enabling it to understand how its maneuvering through its environment.

Together the neo_localization and tf packages will enable the sensors to give the Nav2 stack localization data of the robot with respect to the map. However, the sensors must have their raw data processed. The IMU and wheel encoders data will be fused and filtered using the Extended Kalman Filter (EKF) which is widely used to produce state estimation for nonlinear process models [13]. The state estimation includes low noise and a reduced drift estimate of vehicle position, velocity and angular orientation. The output of the EKF is filtered odometry data that can be sent to the local planner and localization node. The LiDAR sensor will use the rplidar package [14] to provide basic handling of the 2D laser scanner and output usable data for the localization node.

2.5.4 Goal Computation

In most cases the robot requires a dynamic goal location to navigate towards. The goal will be obtained via the airports Wi-Fi network. Most airport Wi-Fi networks are robust and consist of multiple routers of known locations. Using the Received Signal Strength Indicator (RSSI) from at least three routers of known locations, a separate server can make a trilateration calculation to determine the goal position that is sent and updated wirelessly to the robot. However, there also remains the option for the user to have a pre-set destination for the robot.

2.5.5 Output Execution

Once the Nav2 stack has received all its inputs, its able to compute a desired output that must be relayed to the motor drivers to drive the wheels. To receive the velocity data from ROS2 on the

Arduino there exists an Arduino library called ros2arduino [15], which converts ROS messages into usable data by the Arduino. Subsequently, to regulate the response to the desired speeds a PID controller is included in the Arduino. The designed PID controller achieves the conclusions made in 2.3.1 Proportional Integral Derivative (PID) Control.

3.0 Implementation Strategy

The conceptual design has identified a solution for mobile robotics in airports that meets challenging navigational, operational, and security constraints. Navigation has been addressed with the use of robust algorithmic approach within the industry standard Nav2 navigational stack. Operational constraints are addressed through payload security, fallback options with lights and speakers, as well as leveraging the sophisticated Wi-Fi network present in all major airports.

The implementation strategy seeks to detail the deployment and practical functionality of the mobile robot. However, before the mobile robots are functional for any given airport it is required create the map in which it will need to navigate. To obtain the map, SSG Robotics would need to perform high resolution 2D scans of the airport or allow the robot to explore on its own and build its map using SLAM. High resolution scans will require a higher upfront cost but will translate to long-term success for all mobile robotics within the airport. Allowing the robot to self explore has virtually no upfront cost but will result in a less accurate map. In both cases, the robots cannot be deployed instantly after purchase.

3.1 Operational Process

Once a map generation has been completed for the robot's operational range the typical system workflow is as follows:

1. User makes an order request via the website or application on the airports Wi-Fi network.
2. The robot queues the task and reports the task to the retailer to begin preparation.
3. The robot takes the path of least resistance and shortest time to the retailer.
4. The retailer staff deposits the goods into the secured compartment.
5. The robot navigates towards the final destination of the user.
6. The user unlocks the compartment with their own unique identifier.
7. The robot repeats steps 2-6 with the next task in the queue, taking while simultaneously taking new requests.

3.2 Implementation Objectives

SSG Robotics has identified four key obtainable milestones for the future work on the project.

1. Complete a full map generation using either manual scanning methods or SLAM.
2. Achieve less than 10 % error in localization drift on a single battery charge.
3. Navigate to any location within the operational range with an average speed faster than walking speed (1.2 m/s).
4. Full compliance with UL 3300 and CSA guidelines.

4.0 Regulations and Standards

The use of mobile delivery robots in airports will require compliance with established safety standards set by UL 3300 [16] and CSA guidelines [17], along with societal impact considerations, to ensure safety to the public. These standards will require the robot to have navigation and collision avoidance, cybersecurity and data protection [18], and must safely interact with humans and the public. To meet requirements in navigation and collision avoidance, both range sensors and LiDAR will be used to map the robot's surroundings, allowing them to avoid obstacles and prevent collisions. Cybersecurity measures will be taken to protect the user data and prevent tampering with robots' controls. Protecting the robot control systems and the user's data will be done using multi-factor authentication to access the systems, employing Splunk ES to monitor the systems and send alerts if the systems are breached and data will be encrypted using end-to-end encryption [19], TLS 1.3 [20] and AES-256 Encryption [21]. For human and public safety, a flashing light and speaker will be integrated into the robot to notify the surrounding public of its location. Since this robot will be delivering a payload, this must also be protected from tampering. This will be done by locking the contents in a sealed compartment attached to the robot and will require a keycode from the user to unlock. In addition, an emergency stop button will be added in the case that the robot malfunctions. The button can be pressed to shut it off immediately. As well, the robot will not cross through security and remain on one side to avoid any interference with security or contraband accidently crossing security.

5.0 Cost Analysis

5.1 Cost Breakdown

The cost analysis provides a detailed breakdown of the total cost of designing, developing and producing the robot. This includes the material costs per robot, the initial design and development cost, the ongoing costs of assembly and the cost to maintain the software license needed to operate the robot. Table 1 illustrates the raw material costs to build this robot with all the necessary hardware. Table 2 breaks down the costs for software licenses and

Table 3 illustrates labor costs for a custom robot designed by SSG Robotics. The costs of hardware and software are taken from retail pricing [22] [23] [24] and licensing cost [25] [26], while the labor costs are taken from industry standards[g] for assembly and our firm's rate for research and development.

Table 1: Material Cost Breakdown

Materials	Cost per Unit (CAD$)	Quantity (per robot)	Total
Arduino Uno WiFi Rev 2	$38.87	1	$38.87
Battery	$29.38	1	$29.38
Lynxmotion aluminum A4WD1 Rover kit*	$363.87	1	$363.87
IMU	$58.71	1	$58.71
LiDAR	$125.00	1	$125.00
Light/Speaker	$18.08	1	$18.08
Motor Driver	$29.40	1	$29.40
Sharp IR Range Sensor	$24.18	3	$72.54
Raspberry Pi 4	$98.95	1	$98.95
IO Expansion Shield for Arduino V7.1	$14.34	1	$14.34
Wheel Encoder	$11.69	2	$23.38
Electric Lock	$11.62	1	$11.62
Miscellaneous Parts**	N/A	N/A	$30.00
TOTAL			**$914.14**

* Lynxmotion aluminum A4WD1 Rover kit includes four wheels, motors and the chassis

** Miscellaneous Parts includes wiring, solder, screws etc.

Table 2: Software Cost Breakdown

Software	Licensing Cost/Month
Splunk ES	$1,500.00
Ubuntu Pro	$41.67
Ros 2	$0.00
TOTAL	**$1,541.67**

Table 3: Labour Cost Breakdown

Category	**Rate (CAD$)**	**Project Timeline**	**Total Cost**
Research and Development	$500/hr	6 Months	$520,000
Assembly	$200/robot	Ongoing	$200000 / 1000 robots

5.2 Cost-Benefit Analysis

Using equation **3**, an estimated price per robot can be found. Assuming a desired profit per robot of $2500, an operational period of 5 years and that the amount of robots built is equal to those sold, letting that be 1000 units, we get that each robot should be sold at $4226.64, as shown in equation **4**. Increasing the number of robots sold will cause a decrease in the price since it will bring down the labour cost per robot, improving the overall profitability.

$$\text{Profit} = \text{Profit per Robot} * \text{Robots Sold} = \text{Revenue} - \text{Fixed Costs} - \text{Variable Costs} \qquad (3)$$

$$\begin{aligned}\text{Profit} = (\text{Robot Price} * \text{Robots sold}) - (\text{Material Cost} * \text{Robots Built}) - \text{Labour Costs} \\ - (\text{Licensing} * \text{Operational Period})\end{aligned}$$

$$\begin{aligned}\text{Robot Price} = [(\text{Profit per Robot} * \text{Robots Sold}) + (\text{Material Cost} * \text{Robots Built}) \\ + \text{Labour Costs} + (\text{Licensing} * \text{Operational Period})]/\text{Robots Sold}\ (4)\end{aligned}$$

$$\text{Robot Price} = [(2500 * 1000) + (914.14 * 1000) + 720000 + (1541.67 * 60)] / 1000$$

$$\textbf{Robot Price} = \$\mathbf{4226.64}$$

References

[1] ICAO, "ICAO," 2024. [Online]. Available: https://www.icao.int/Meetings/FutureOfAviation/Pages/default.aspx. [Accessed 2 March 2025].

[2] Demantic, "Demantic," September 2019. [Online]. Available: https://www.dematic.com/en-us/insights/case-studies/radial-groningen-netherlands/. [Accessed 2 March 2025].

[3] SLAMTEC, "RPLIDAR," 4 July 2016. [Online]. Available: https://www.generationrobots.com/media/rplidar-a1m8-360-degree-laser-scanner-development-kit-datasheet-1.pdf?srsltid=AfmBOore_Rv_uWhZoPAEqjtggFVnpN7nZTQO2pNiNXgRf2euiOE-qVD8. [Accessed 1 March 2025].

[4] S. Macenski, F. Martín, R. White and J. Ginés Clavero, "The Marathon 2: A Navigation System," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* 2020.

[5] A. Boeing, "Adrian Boeing: Blog," Blogger, 8 May 2012. [Online]. Available: https://adrianboeing.blogspot.com/2012/05/dynamic-window-algorithm-motion.html. [Accessed 1st March 2025].

[6] P. Li, L. Hao, Y. Zhao and J. Lu, "Robot obstacle avoidance optimization by A* and DWA fusion algorithm," *PLOS ONE,* 2024.

[7] G. Granosik, K. Andrzejczak, M. Kujawinski, R. Bonecki, Ł. Chlebowicz, B. Krysztofiak, K. Mirecki and M. Gawryszewski, "USING ROBOT OPERATING SYSTEM FOR AUTONOMOUS CONTROL OF ROBOTS IN EUROBOT, ERC AND ROBOTOUR COMPETITIONS," *Acta Polytechnica CTU Proceedings,* vol. VI, pp. 11-17, 2016.

[8] Open Robotics, "ROS.org," 4 February 2019. [Online]. Available: https://wiki.ros.org/gmapping. [Accessed 2 March 2025].

[9] Open Robotics, "ROS.org," 23 March 2020. [Online]. Available: https://wiki.ros.org/map_server. [Accessed 2 March 2025].

[10] Neobotix, "Neobotix Documentation," 2024. [Online]. Available: https://neobotix-docs.de/ros/packages/neo_localization.html. [Accessed 2 March 2025].

[11] Open Robotics, "ROS.org," 27 August 2020. [Online]. Available: https://wiki.ros.org/amcl. [Accessed 2 March 2025].

[12] Open Robotics, "ROS.org," 2 October 2017. [Online]. Available: http://wiki.ros.org/tf . [Accessed 2 March 2025].

[13] T. Konrad, J.-J. Gehrt, J. Lin, R. Zweigel and D. Abel, "Advanced state estimation for navigation of automated vehicles," *Annual Reviews in Control,* vol. XLVI, pp. 181-195, 2018.

[14] Open Robotics, "ROS.org," 2 August 2019. [Online]. Available: http://wiki.ros.org/rplidar. [Accessed 2 March 2025].

[15] Arduino, "Arduino Documentation," [Online]. Available: https://docs.arduino.cc/libraries/ros2arduino/. [Accessed 2 March 2025].

[16] "Standards & Engagement," 14 May 2024. [Online]. Available: https://ulse.org/ul-standards-engagement/standards-matter/how-standards-are-making-robots-safe-public-and-commercial. [Accessed 2 March 2025].

[17] "Workplace Safety & Prevention Services," 24 March 2022. [Online]. Available: https://www.wsps.ca/resource-hub/articles/safeguarding-and-robots-csa-standards-update. [Accessed 2 March 2025].

[18] W. Dudek and W. Szynkiewicz, "Cyber-security for Mobile Service Robots – Challenges for Cyber-physical System Safety," *Journal of Telecommunications and Information Technology,* vol. II, pp. 29-36, 2019.

[19] R. Battat, "Preveil," 30 August 2024. [Online]. Available: https://www.preveil.com/blog/end-to-end-encryption/. [Accessed 2 March 2025].

[20] J. Mehta, "Encrypted Fence," [Online]. Available: https://certera.com/blog/tls-1-3-everything-you-need-to-know/. [Accessed 2 March 2025].

[21] Luxsci, "Luxsci," [Online]. Available: https://luxsci.com/256-bit-aes-encryption-for-ssl-and-tls-maximal-security/. [Accessed 2 March 2025].

[22] RobotShop, "RobotShop," [Online]. Available: https://ca.robotshop.com/search?q=safety%20light. [Accessed 2 March 2025].

[23] "AliExpress," [Online]. Available: https://www.aliexpress.com/i/32880431474.html. [Accessed 2 March 2025].

[24] "Adafruit," [Online]. Available: https://www.adafruit.com/product/5134. [Accessed 2 March 2025].

[25] "Canonical Ubuntu," [Online]. Available: https://ubuntu.com/pro/subscribe. [Accessed 2 March 2025].

[26] "Splunk," [Online]. Available: https://www.splunk.com/en_us/products/pricing.html?utm_campaign=google_amer_en_search_brand&utm_source=google&utm_medium=cpc&utm_content=Splunk_Prod_Pricing_EN&utm_term=splunk%20cost&device=c&_bt=686731841999&_bm=p&_bn=g&gad_source=1&gclid=CjwKCAiA5pq-BhB. [Accessed 2 March 2025].

[27] Whill.inc, "Industry Applications," [Online]. Available: https://whill.inc/ca/mobility-service/industry.

[28] Mordor Intelligence, "Route Optimization Software Market Size & Share Analysis - Growth Trends & Forecasts (2024 - 2029)," 2024. [Online]. Available: https://www.mordorintelligence.com/industry-reports/route-optimization-software-market.