# AutoNav

## ECE 499 - Design Project II

August 6th 2025

University of Victoria

**Group Information**

| S. No. | Name | V Number |
|---|---|---|
| 1 | Mattias Kroeze | V00934043 |
| 2 | Eiden Sol Strozberg | V00936098 |
| 3 | Jairus Abad | V00943541 |

## Acknowledgment

We would like to express our gratitude to Dr. Papadopoulos for his valuable and helpful recommendations during the planning stage and analysis of our research. His patience and real-world engineering experience helped direct us towards creating an innovative project for helping those in need.

We gratefully acknowledge Professor Sana, the Teaching Assistants and the Department of Electrical and Computer Engineering for their instruction and guidance throughout the program. We also appreciate their work in organizing both the ECE 499 class and the Capstone Demo Competition, which provided an invaluable platform for showcasing our project and our technical skills. Additionally, we thank Dr. McGuire for generously providing the project's financial support.

We also acknowledge the support of our families and friends, whose encouragement enabled the completion of this project and its pursuit of an assistive solution.

## Table of Contents

1. Introduction........ 1
2. Objectives........ 2
3. Design Specifications........ 2
    3.1. Robotic Platform........ 2
    3.2. Raspberry Pi 5........ 2
    3.3. Custom Battery Power Supply........ 3
    3.4. LD19 LiDAR Sensor........ 3
    3.5. Pi Camera Module........ 3
    3.6. BNO085 Inertial Measurement Unit (IMU)........ 3
    3.7. Ergonomic Cane & Handle........ 3
    3.8. PCB Design........ 3
    3.9. Compliance and Ethical Considerations........ 4
4. Literature Survey........ 4
5. Team Duties & Project Planning........ 5
    5.1. Mattias Duties........ 5
        5.1.1. AutoNav Platform Construction........ 5
        5.1.2. Power Supply Design........ 5
        5.1.3. Autonomous Navigation Model Training........ 6
        5.1.4. System Integration........ 6
    5.2. Eiden Duties........ 7
        5.2.1. LiDAR Sensor........ 7
        5.2.2. PCB Design........ 7
    5.3. Jairus Duties........ 8
        5.3.1 Cane CAD Design........ 8
        5.3.2 Scene Description Subsystem........ 8
        5.3.3 Web App Integration........ 8
6. Design Methodology & Analysis........ 8
    6.1. Robotic Platform Design........ 8
        6.1.1. Sensor Selection........ 9
        6.1.2. Compute Hardware........ 9
        6.1.3. Pi Breakout PCB........ 10
        6.1.4. Batteries and Power Supply........ 10
        6.1.5. RC Platform Selection........ 11
        6.1.6. Electronics Chassis Design........ 12
        6.1.7. Cane Design........ 13
    6.2. Reinforcement Model Training........ 14
        6.2.1. Environment Model Creation........ 15
        6.2.2. Reinforcement Learning Simulation Environment........ 15
        6.2.3. Training Environment Creation........ 16
        6.2.4. Narrowing Sim-to-Real Gap........ 18
    6.3. Platform Software and Integration........ 19
        6.3.1. Inference Loop........ 19
        6.3.2. Visual Place Recognition........ 19
        6.3.3. OpenAI Scene Descriptor........ 20
        6.3.4. Shared State Subsystem........ 20
7. Design & Prototype........ 20

8. Testing & Validation........ 21
9. Cost Analysis........ 24
    9.1. Direct Costs........ 24
    9.2. Indirect Costs........ 25
    9.3. Funding........ 25
    9.4. Net Cost........ 25
    9.5. Tentative Pricing & Return on Investment (ROI)........ 25
    9.6. Cost Reduction........ 26
10. Conclusion & Recommendations........ 26
References........ 27
A.1 Appendix A – EGBC Code of Ethics........ A-1
B.1 Appendix B – Website Link........ B-1
C.1 Appendix C – Youtube Video of AutoNav........ C-1

## Executive Summary

AutoNav addresses the growing need for accessible indoor navigation for individuals with visual impairments through an autonomous robotic aid built on affordable, off-the-shelf hardware. The system combines a Raspberry Pi-based platform with a 360° LiDAR sensor and Pi Camera, leveraging OpenAI's Vision API for scene understanding. Navigation is handled by a Proximal Policy Optimization (PPO) reinforcement learning model trained in NVIDIA's Isaac Sim. AutoNav demonstrated its effectiveness in navigating complex environments and delivering practical assistance, ultimately enhancing user independence and safety. Future iterations of the platform would benefit from upgraded compute hardware and a more task-specific robotic platform, broadening the system's real-world reliability and applicability.

## 1. Introduction

Visually impaired individuals face substantial barriers when independently navigating indoor spaces. Traditional aids, such as white canes or guide dogs, offer primarily passive feedback and rely extensively on the user's prior familiarity with the environment or external assistance. This limitation can significantly restrict the independence and mobility of visually impaired persons, affecting their quality of life and autonomy. As indoor environments like workplaces, homes, hospitals, and public buildings become increasingly complex and dynamic, there is a pressing need for more sophisticated and active assisting methods that enable greater navigational autonomy.

The AutoNav project is designed explicitly to address these limitations by developing an autonomous smart-cane capable of actively guiding users to known destinations. By utilizing modern robotics and artificial intelligence, AutoNav offers a substantial improvement over conventional methods. The smart-cane uses an RC-based robotic platform controlled by a Raspberry Pi Linux computer [1], integrated with a high-resolution 360° planar LiDAR sensor [2], IMU [3], and a Pi Camera [4]. The LiDAR provides precise distance measurements, essential for active environments and obstacle avoidance. The IMU measures and provides the RC's acceleration and angular rate. Simultaneously, the Pi Camera captures visual data processed through OpenAI's Vision API, to provide environmental descriptions via audio feedback and localization of the smart-cane mid route [5]. Together, these three significantly enhance spatial awareness and safety for users.

This project aligns closely with the Engineers and Geoscientists BC (EGBC) Code of Ethics [6], particularly the principles emphasizing public safety, health, and welfare, as well as honest and transparent communication of technical capabilities and limitations. By focusing on user safety and clearly disclosing the system constraints, the project maintains the high ethical standard set out by EGBC. Furthermore, AutoNav directly supports the EGBC ethical principle of social responsibility, as it seeks to provide equitable access to mobility solutions, bridging existing gaps caused by current limitations in accessibility technologies.

The potential user base for AutoNav includes visually impaired individuals across a variety of indoor settings. Users can benefit significantly by gaining the ability to independently navigate environments that are typically challenging or inaccessible without assistance. AutoNav not only supports daily living activities but also enhances social inclusion by allowing users to participate more fully and confidently in professional, educational, and social activities.

## 2. Objectives

The primary objective of the project was to develop a cost-effective, intelligent navigation aid tailored for individuals with visual impairments, This included designing a mobile platform capable of indoor navigation and providing real-time auditory scene descriptions to the user. The team focused on achieving this goal through careful hardware integration, reinforcement learning, and user-friendly interface.

Initial development involved constructing the prototype using a reliable RC platform as the mobility base, followed by the design and fabrication of a custom 3D printed chassis to house the components securely. A compact, custom-designed PCB was created to improve system integration and maintain electrical reliability. Efforts were made to design an ergonomic smart cane equipped with intuitive controls to serve as the primary interface.

Key complex tasks and milestones include:

- Training a Proximal Policy Optimization (PPO) reinforcement learning model for autonomous navigation from an accurate simulation environment and robot model in Nvidia's Isaac Lab.
- Integrating a 360° LiDAR sensor and Pi Camera for precise environmental scanning and real-time scene analysis.
- Develop control logic for the user interface, ensuring seamless interaction between user commands and platform response.
- Design and build an autonomous vehicle capable of path navigation and obstacle avoidance.

## 3. Design Specifications

The AutoNav system incorporates several interconnected subsystems, each meticulously selected to optimize performance, cost-efficiency, reliability, and safety.

### 3.1. Robotic Platform

The RC base was chosen due to its reliability, flexibility, and modularity. The platform ensures smooth navigation across various surfaces. It includes a powerful DC motor, articulating suspension, 4-wheel drive, and servo steering capable of precise maneuvering, essential for accurate route following.

### 3.2. Raspberry Pi 5

A Raspberry Pi 5 (8 GB model) was selected as the primary compute board, offering a balance of computing power and cost for the project's requirements. It delivers sufficient performance to handle sensor data capture, motor control and on-board AI inference while remaining within budget.

## 3.3. Custom Battery Power Supply

A custom-made battery pack with regulated voltage output ensures stable, reliable power delivery. The battery management system safeguards against overcharge, discharge, and short circuits, aligning with the EGBC guidelines on public safety and product reliability standards.

## 3.4. LD19 LiDAR Sensor

The LD19 LiDAR provides a 360° planar field of view, measuring distances up to 12 meters with an accuracy of ±5 centimetres. Its high-frequency scanning capability (4500 Hz sampling rate) offers superior environmental perception, vital for obstacle avoidance and navigation. The LD19 was selected over ultrasonic or infrared sensors due to its higher resolution and radial scanning pattern.

## 3.5. Pi Camera Module

The Pi Camera captures images used for both the Visual Place Recognition (VPR) system and the OpenAI scene description pipeline. It offers 5 MP resolution with a 54°×43° viewing angle and adjustable focus, enabling consistent image quality under varied indoor lighting conditions. For VPR, these images are converted into visual embeddings to localize the robot within one of 50 predefined regions. This functionality is critical for supporting high-level navigation decisions. Additionally, the same camera is used to generate scene descriptions through OpenAI's Vision API, which are converted to audio feedback to assist visually impaired users in understanding their surroundings.

## 3.6. BNO085 Inertial Measurement Unit (IMU)

The integrated IMU provides crucial orientation and motion tracking data through a combination of accelerometers, gyroscopes, and magnetometers. It delivers accurate measurements of pitch, roll, yaw, and acceleration, significantly enhancing the robot's capability for precise localization and smooth navigation. The IMU chosen offers a high refresh rate (up to 100 Hz), essential for maintaining accurate real-time feedback in dynamic conditions. Its compact size, reliability, and cost-effectiveness make it ideal for seamless integration into the AutoNav system.

## 3.7. Ergonomic Cane & Handle

The smart-cane is ergonomically designed for user comfort, including intuitive button placement and weight distribution to minimize user fatigue. PLA filament [7] prioritizes durability, weight reduction, easy production, and compliance with ergonomic standards to promote extended usability and comfort.

## 3.8. PCB Design

A custom-designed PCB breakout board simplifies connections to all system components, minimizing wiring complexity and giving the prototype a refined, professional appearance.

## 3.9. Compliance and Ethical Considerations

All component selections and system designs strictly adhere to EGBC's Code of Ethics, specifically ensuring user safety, public welfare, transparency, and environmental responsibility. Regulations concerning electronic component safety, electromagnetic compatibility, and battery management were rigorously followed.

## 4. Literature Survey

Assistive technologies for visually impaired navigation have seen significant advancements, yet each solution presents distinct limitations in terms of cost, usability, and capability.

The Glidance device, marketed as the world's first fully autonomous mobility aid, combines robotics and AI to physically guide users without the need for traditional aids such as white canes or guide dogs. Although robust in capability, its subscription-based pricing model limits accessibility for many users due to high long-term costs [8]. Additionally, Glidance's comprehensive system can be bulky and require complex setup procedures, potentially hindering ease of use for everyday applications.

Stanford University's Intelligent Systems Lab developed a DIY augmented cane, priced at approximately $400 USD, incorporating LiDAR, GPS, and sensor fusion to actively guide users via gentle haptic feedback. This approach notably increased users' walking speed by approximately 20% compared to traditional white canes [9]. Despite its affordability, the augmented cane's reliance on GPS limits its accuracy indoors, especially in buildings with weak satellite reception.

Commercially available ultrasonic and infrared (IR)-based navigational devices typically provide simple proximity alerts rather than active guidance. For instance, devices like UltraCane [10] and BuzzClip [11] offer obstacle detection through haptic feedback but lack route planning or active navigation capabilities. These passive aids require significant cognitive effort from the user, thereby limiting their practical usability and effectiveness.

AutoNav improves upon these existing technologies by integrating active robotic navigation, precise environmental sensing through a 360° LiDAR sensor, and scene analysis through a Pi Camera linked to OpenAI's advanced Vision API. The PPO-based reinforcement learning [12] approach ensures robust navigation through dynamic environments, significantly surpassing the capabilities of simpler obstacle detection systems.

In terms of cost, AutoNav achieves a competitive advantage by combining affordable off-the-shelf hardware components, totalling approximately $801.14, not including prototype funding, with minimal ongoing operational costs, unlike subscription-based systems. Furthermore, the intuitive user interaction model of AutoNav, a simple button press connected with audio feedback, enhances ease of use and reduces the learning curve significantly compared to more complex devices like Glidance.

The project adheres strictly to the EGBC Code of Ethics, prioritizing safety, transparency, and social responsibility. Regulations for electronic safety, electromagnetic compatibility, and battery management standards have been rigorously followed, ensuring user safety and product reliability.

# 5. Team Duties & Project Planning

## 5.1. Mattias Duties

The responsibilities and project planning of group member one are highlighted in this section.

### 5.1.1. AutoNav Platform Construction

Serving as the basis for this project, we needed to research and select a reliable RC car platform that could support the weight, size, and power demands of an autonomous navigation system. The platform had to be compact enough for indoor use, lightweight for ease of transport, and robust enough to handle repeated testing and hardware modifications.

Once the platform was selected, I designed and 3D printed custom chassis components to mount all compute hardware, batteries, and sensors. Due to the size constraints of the project, tolerances between components were extremely tight, so significant care and planning went into the layout, wiring, and overall construction. Heat-set threaded inserts were embedded into the prints to allow for repeatable assembly and disassembly. These frames ensured mechanical stability while also enabling rapid iteration during integration and testing.

### 5.1.2. Power Supply Design

Design and troubleshoot a power supply capable of reliably powering the platform. RC hardware is typically tolerant to significant voltage sag due to the high-current demands placed on the batteries, but this behaviour is incompatible with compute hardware such as Raspberry Pis, which are highly sensitive to voltage drops and operate within a narrow tolerance range. Extensive troubleshooting and redesign were required to identify a battery and regulator combination that could maintain stable voltage under rapidly changing load conditions, eliminating the risk of voltage sag and resulting brownouts.

### 5.1.3. Autonomous Navigation Model Training

To enable autonomous navigation, I was responsible for training a reinforcement learning model using Proximal Policy Optimization (PPO). Rather than training the model on the physical robot, which would have been slow, hardware-intensive, and potentially destructive, I built a 1:1 simulation of both the robot and its environment using Isaac Sim. This allowed us to run hundreds of parallel training environments, accelerating development and enabling rapid iteration on reward structures and observation space design.

To create an accurate simulation, I used LiDAR scans from an iPhone 15 Pro to reconstruct the third floor of the ELW building and modified an existing Isaac Lab RC car 3D model to match our physical platform. I then developed a waypoint-based navigation system across 49 distinct regions, along with a fine-tuned observation and reward scheme to incentivize navigation and punish collisions.

Training posed continuous challenges, particularly when transferring the model to real hardware. To address the "sim-to-real" gap, I implemented domain randomization - added observation noise, and introduced observation and action delays into the simulation. I also compared IMU data between hardware and simulation to fine-tune vehicle dynamics. Despite these efforts, transferring our most intelligent models to hardware resulted in poor real-world performance. A simplified generalized policy was ultimately deployed as a fallback, which reliably demonstrated hallway following and obstacle avoidance in real-world tests.

### 5.1.4. System Integration

To facilitate the final operation of AutoNav, I developed a central controller script to manage a set of modular subsystems, each running on an independent thread to avoid blocking or interfering with others. All subsystems inherit from a shared base class that defines a consistent initialization pattern and loop structure, ensuring clean integration and predictable behaviour across the system.

A top-level app script controls the behaviour of the central controller and provides a user interface built with Plotly Dash. This frontend allows the user to interact with the system and view real-time telemetry visualizations, maps, and performance charts.

A set of custom drivers handles low-level communication with all onboard sensors and actuators. The controller collects data from these drivers and routes it to the appropriate subsystems. This architecture supports the main inference loop, the visual place recognition system, and the OpenAI scene description pipeline, while maintaining clear separation of responsibilities across components.

Significant care went into the design of this architecture due to the complexities of working with multithreaded programs. Without strict control over initialization order, shared state, and thread lifecycles, subtle bugs and race conditions can lead to system instability. The introduction of a subsystem base class was key in maintaining this control, as it enforced consistent structure, threading behaviour, and lifecycle management across all subsystems. Additional safeguards, such as centralized telemetry handling using a shared state class, were critical in ensuring stability and responsiveness under real-time conditions.

## 5.2. Eiden Duties

The responsibilities and project planning of this group member are highlighted in this section.

### 5.2.1. LiDAR Sensor

The integration and implementation of the LiDAR sensor subsystem were prioritized due to its critical role in navigation and real-time environment perception. Key responsibilities included selecting the appropriate LiDAR sensor (LD19), designing data parsing and validation algorithms, and ensuring seamless real-time communication with the Raspberry Pi.

An issue arose where the LD19 sensor was transmitting data faster than it could be read by the main program loop. As a result the serial buffer remained full of outdated data, causing the robot to react to stale environmental inputs. This led to delays in obstacle detection and erratic behaviour.

The solution involved running the LiDAR data acquisition on a separate high-priority thread. This thread continuously read from the serial port at the maximum possible rate to keep the buffer clear, ensuring up-to-date data was always available. Data was only passed to the main thread upon request, maintaining clean decoupling between data collection and usage. This adjustment significantly improved the responsiveness and reliability of the obstacle avoidance and demonstrated the importance of multithreaded architecture when handling high-throughput sensor data.

### 5.2.2. PCB Design

The PCB design task was essential for achieving compactness, reliability, and maintainability within the project. Responsibilities included designing a custom Pi Breakout PCB to streamline sensor connections and minimize wiring complexity. Challenges faced during this process involved finding proper footprints for the Pi and the IMU and ensuring proper component selection for longevity and reliability. A few revisions of the PCB were made, ultimately resulting in a highly reliable and effective final design.

## 5.3. Jairus Duties

The responsibilities and project planning of this group member are highlighted in this section.

### 5.3.1. Cane CAD Design

The CAD model of the Cane was considered to be a high-priority for the completion of the prototype. The cane was to be made ergonomic for extended use and required to support dual momentary triggers. Challenges encountered for this responsibility were miscalculations of tolerances when 3D printing, as well as the orientation of the model for durability. Several revisions were made to address these issues.

### 5.3.2. Scene Description Subsystem

The scene description subsystem was considered to be a medium-priority for the prototype since the sole function of the platform was for autonomous guidance. The development of the subsystem surfaced two constraints. First, the low-cost Raspberry Pi Camera Module exhibits limited dynamic range and low-light sensitivity which causes a rapid loss of visual features under indoor conditions at times. This limitation was left unresolved in the current prototype; future iterations will consider incorporating a higher quality camera. Secondly, initial captioning yields verbose outputs that include aesthetic context rather than mobility-relevant cues. To resolve this, the model was fine-tuned in order to only describe features that might block the user’s path.

### 5.3.3. Web App Integration

Integrating the scene description module with the web application was treated as a medium-priority task. The first text-to-speech engine was sluggish and contained noise. To resolve this issue, the engine was replaced with OpenAI’s text-to-speech endpoint which produced clearer, faster audio. A Python service calls the OpenAI API which converts each caption from the Scene Description Subsystem to an MP3 which the front end requests in order to play audio for the user and presents the caption text on the web app for testing and validation.

# 6. Design Methodology & Analysis

To develop the AutoNav prototype, the design task was divided into three primary phases: robotic platform design, reinforcement model training, and system integration. Each phase involved iterative decision-making guided by technical constraints, real-world feasibility, and simulation-based analysis.

## 6.1. Robotic Platform Design

The core challenge in designing the robotic platform was to select sensors, compute hardware, and an RC car base that could support the demands of autonomous indoor navigation, while staying within a realistic prototype budget.

### 6.1.1. Sensor Selection

Our first priority was determining the minimum viable combination of sensors that would allow a reinforcement learning (RL) model to navigate reliably in known environments. Through a combination of research and empirical testing, we settled on a 360° planar LiDAR (LD19) and an IMU (BNO085) as the primary sensor inputs.

The LiDAR provides rich spatial information, enabling the agent to perceive obstacles and corridor geometry. The IMU contributes gyro yaw rate, linear acceleration data and compass heading, which the agent uses to estimate its motion, allowing it to smoothly maneuver through turns. Additionally, the IMU data allowed us to characterize the motion properties of the car, which we needed to match the behaviour of the simulation as closely as possible.

While this configuration allowed successful autonomous navigation through straight corridors and turns, we encountered an issue known as perceptual aliasing: certain locations appeared indistinguishable from others when relying solely on LiDAR. This was problematic in scenarios like T-junctions or intersections, where different actions (turn left vs. turn right) were context-dependent.

To address this, we initially tried incorporating the compass heading from the IMU, but this failed to differentiate regions with identical layout and orientation. We then introduced a Visual Place Recognition (VPR) system using an onboard camera to provide high-level contextual information. The VPR system allows the robot to be able to localize itself to any of 50 distinct regions we established in our 3rd-floor ELW testing grounds.

### 6.1.2. Compute Hardware

Choosing the onboard computer hardware required balancing inference speed, thermal/power constraints, and cost.

We initially considered high-performance edge computing platforms such as the NVIDIA Jetson Orin Nano [14], which offers strong inference performance through GPU acceleration. However, due to severe supply shortages and price inflation pushing prices above $700 CAD despite an MSRP of $350 CAD, this option became infeasible within our budget constraints. On the opposite end of the spectrum, low-cost microcontrollers, such as ESP32 [15] or Arduino-class devices [16], were ruled out due to insufficient processing capability, especially once vision-based features became necessary.

The optimal balance was found with the Raspberry Pi 5, a general-purpose Linux single-board computer with a quad-core 2.4GHz processor and 8GB of RAM. Compared to earlier Pi models, the Pi 5 delivers a significant performance boost, enabling higher-rate inference of our PPO-trained navigation policy, while leaving headroom for auxiliary subsystems such as our VPR and OpenAI loops, as well as serving the mobile webapp for end users to interact with.

### 6.1.3. Pi Breakout PCB

The complexity and compactness of the AutoNav necessitated the development of a custom breakout PCB specifically for the Raspberry Pi. Initially, direct connections using jumper wires made it difficult to work with due to their fragility, are susceptible to electromagnetic interference (EMI), and challenges in troubleshooting and maintenance. The Pi Breakout PCB was thus designed to significantly streamline the connections, provide reliability, and simplify debugging procedures. The PCB layout considers features like labelled terminals, secure screw-terminal connectors, and dedicated voltage rails. The PCB model can be seen below in Figure 1.

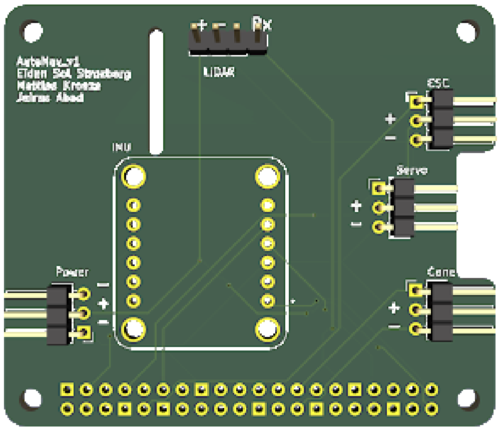


*Figure 1: 3D Viewer of PCB (KiCAD).*

This custom-designed PCB not only improved reliability and maintainability but also significantly reduced the overall project wiring complexity, enabling rapid system prototyping and iterative additions. The finalized breakout PCB design proved essential in addressing earlier integration issues and ensuring consistent performance throughout all phases of testing and demonstration.

### 6.1.4. Batteries and Power Supply

Most hobby-grade Electronic Speed Controllers (ESCs) include a basic voltage regulator as part of a Battery Eliminator Circuit (BEC), allowing the servos and RC receiver to be powered through the same three-wire interface. While this design greatly simplifies wiring for typical RC applications, it proved unsuitable for our system due to the strict power requirements of the Raspberry Pi 5.

The BECs found in typical ESCs can usually supply only around 1 A of current, and while they are labelled as 5 V, they often operate at slightly higher voltages (5.5 - 6 V) to accommodate the tolerance of standard RC components. These devices are designed to function across a broad voltage range and are generally unaffected by voltage sag during load. However, the Raspberry Pi is significantly less tolerant. It is rated for a strict 5 V ±5% range; if the voltage drops below 4.7 V, the Pi begins to throttle performance, and at 4.6 V or lower, it will shut down entirely.

This sensitivity is problematic under computationally heavy loads such as real-time inference or visual embedding extraction for our Visual Place Recognition (VPR) system. Even with a seemingly sufficient 5 V supply, rapid current draw would often cause voltage sag severe enough to trigger Pi throttling or full brownouts.

To address this, we replaced the ESC-integrated BEC with a higher-current external UBEC (Universal Battery Elimination Circuit), which significantly improved voltage stability under load. However, despite the improved regulator, we still observed occasional brownouts, especially when drawing high peak current during processing-intensive tasks. Further investigation with an oscilloscope revealed that the issue was not with the regulator itself, but with the battery: the standard 1400 mAh RC LiPo pack, while capable of high burst current, had unacceptable voltage sag during sustained draws.

To resolve this, we upgraded the battery to use a swappable set of four 18650 lithium-ion cells, a common battery cell used in laptops and electric vehicles. We configured four cells in a 2S2P arrangement - two cells in series, wired in parallel with another identical pair. This new configuration delivers cleaner power, increased energy storage, and improved voltage stability under load.

Since making this change, all power-related faults have been eliminated, and the Raspberry Pi has operated reliably under nearly all usage conditions, including peak inference workloads.

#### 6.1.5. RC Platform Selection

Due to the large-scale task of training an autonomous navigation system, we chose to focus our efforts on that core challenge and reduce the development overhead of building the robotic platform from scratch. To achieve this, we selected a common and well-regarded off-the-shelf RC platform. This allowed us to work with a reliable, robust, and easily serviceable base, enabling faster iteration and integration. While a future version of this project would benefit from a custom-built platform tailored specifically to this application, our approach was the most practical given our limited timeframe.

When selecting a platform, the most important considerations were scale and price. RC vehicles are typically categorized by scale relative to real vehicles. While 1/10th scale platforms are widely available, they are too large and expensive for our needs. Their size would make the system cumbersome for end users to transport, and their speed and weight could pose safety risks during failures. They are also poorly suited for navigating indoor environments such as hallways due to their size and speed.

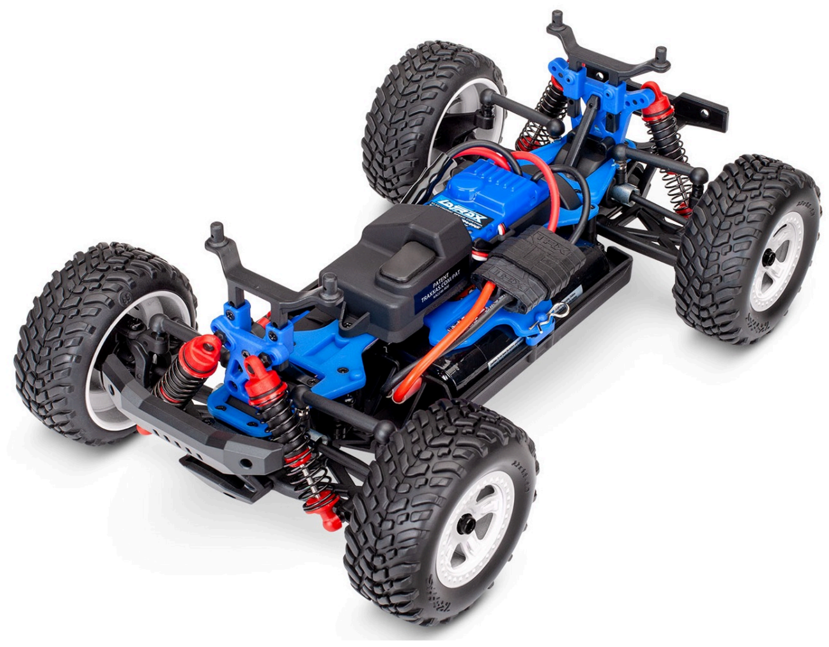


*Figure 2: The selected RC Car Platform.*

We ultimately chose the LaTrax Prerunner [17], a 1/18th scale vehicle that struck the ideal balance. It offered just enough internal space to house our onboard hardware with careful planning, while remaining compact and maneuverable. We selected this model in part because its parent company, Traxxas, is one of the most reputable names in the RC industry. The vehicle is easy to disassemble and repair, with widespread parts availability in local hobby stores.

### 6.1.6. Electronics Chassis Design

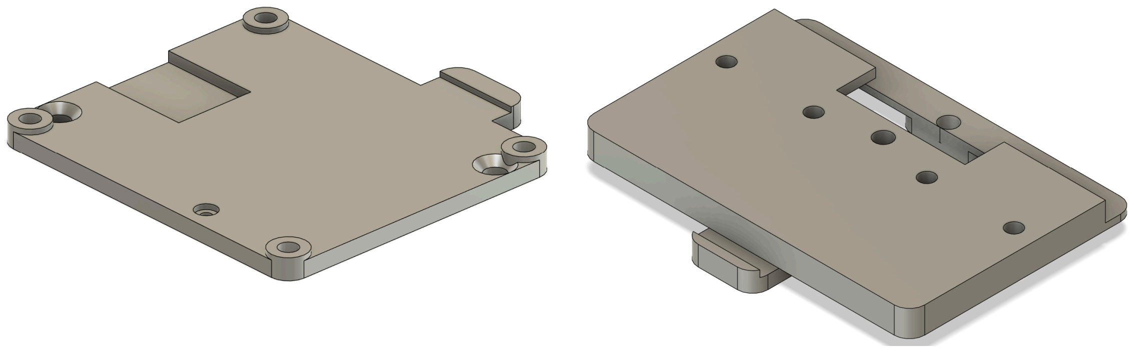


*Figure 3: Raspberry Pi (Left) and Battery (Right) Chassis.*

To mount all compute hardware, sensors, and batteries, we designed and 3D printed custom chassis components that affix directly to the LaTrax Prerunner platform. Where possible, we utilized the vehicle's existing mounting holes and hardware. Additional rigidity was achieved using zip ties in key locations. To allow for repeatable assembly and disassembly, we embedded heat-set brass threaded inserts into the printed components, avoiding the need to drive machine screws directly into plastic - an unreliable mechanical connection prone to wear and loosening.

The LiDAR and camera, and the cane's U-joint mount are fixed to a dedicated bracket that uses the RC chassis's original body-pin post.

#### 6.1.7. Cane Design

Several revisions of the handle were modelled in CAD, 3D printed and tested for ergonomics and control. The two most viable prototypes are presented in Figures 4 and 5 below. Figure 4 offers a visually refined design, however, its squared edges create a rather uncomfortable feeling for prolonged use. The revised spherical geometry in Figure 5 resolves these issues and was therefore selected for the final cane assembly. It accommodates two momentary trigger switches. One trigger was positioned for thumb actuations on the upper surface and the second trigger was positioned beneath the grip for index-finger input. The handle contains an internal conduit that neatly routes the wires to the vehicle. These interfaces tie directly to the Autonomous Guidance and Scene-Observation subsystem outlined in Section 6.3.

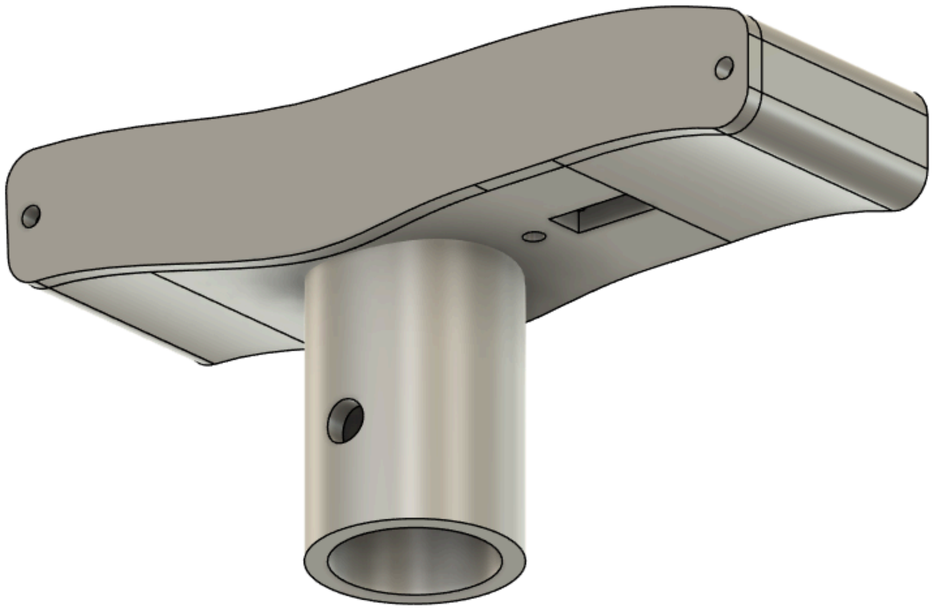


*Figure 4: First Revision of the Cane Handle.*

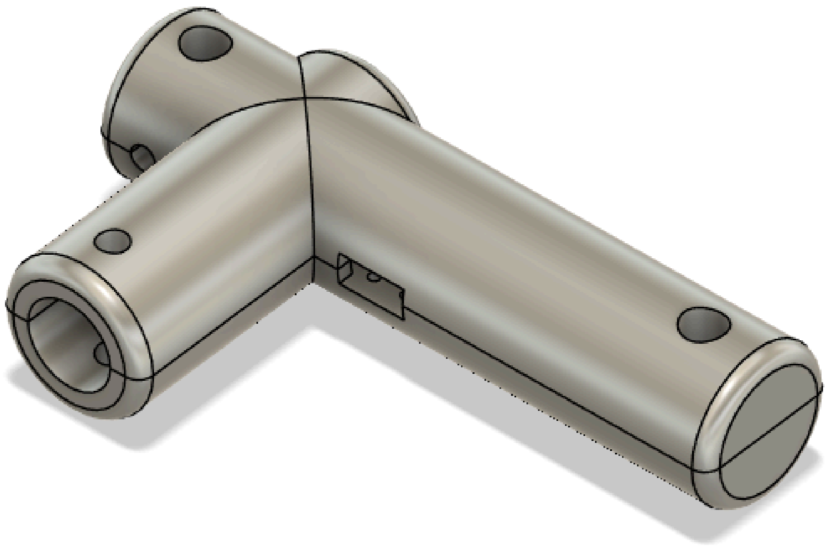


*Figure 5: Second Revision of the Cane Handle.*

In operation, the upper switch toggles autonomous-guidance engagement, while the lower switch initiates on-demand scene narration. The sealed wiring channel preserves a clean exterior profile, protects the electrical components and simplifies maintenance by allowing module replacement.

## 6.2. Reinforcement Model Training

For our robot to successfully autonomously navigate, we need to provide an AI agent with observations of the world and train a model to learn which actions are good and which are not. While we could have assembled the car and trained it in real-time, this would be an extremely slow and laborious process, and likely damaging to the platform due to frequent collisions during the early stages of learning. On top of that, the car would need to be fully assembled and operational before training could even begin, which would make it difficult to parallelize tasks across the project timeline.

A smarter approach is to build a 1:1 recreation of both the robot and our chosen indoor environment in simulation. This allows us to run hundreds of parallel environments for hours each day, accelerating training by orders of magnitude. Additionally, with so many parallel instances, we can iterate far more quickly on the reward structure and observation space design, whereas real-world training would require hours of runtime just to see results, leading to extremely slow feedback cycles.

### 6.2.1. Environment Model Creation

To create realistic models of our real-world testing environment, we chose to utilize the LiDAR scanning capabilities of an iPhone 15 Pro. Apps such as PolyCam and Kiri Engine [13], which leverage Apple's RoomPlan SDK, allowed us to walk through the hallways of the environment and meticulously scan every surface, generating a scale-accurate representation. We had the option of using a triangular mesh scan or the RoomPlan mode, which constructs 3D objects to represent the geometry of walls, doors, and furniture. While the mesh scan mode produced more detailed and accurate representations, the RoomPlan mode yielded a much cleaner model. This significantly reduced aliasing artifacts and provided a more navigable environment for the robot, minimizing collisions with jagged or irregular geometry.

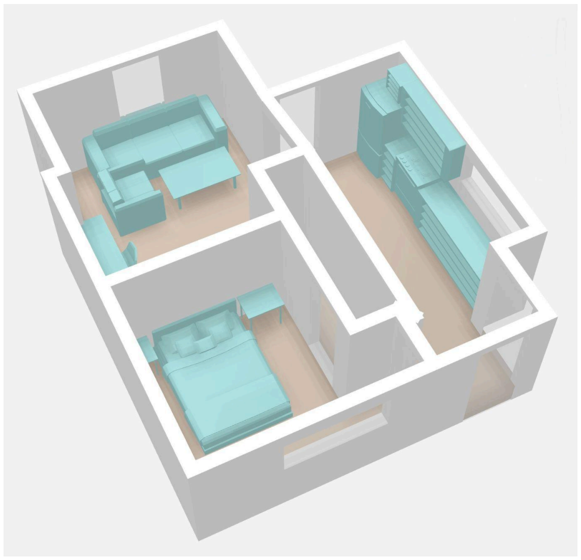


*Figure 6: KIRI Engine 3D model of Mattias' 1 Bedroom Suite.*

To model the car itself, we adapted a basic RC car model provided by an Isaac Lab community project - Leatherback [18]. The geometry was scaled down and modified to better reflect the physical dimensions and properties of our specific RC platform, ensuring the simulation closely matched real-world behaviour.

### 6.2.2. Reinforcement Learning Simulation Environment

When selecting a simulation platform, we considered three main contenders: PyBullet, Gazebo, and Isaac Lab [19, 20, 21]. PyBullet is extremely lightweight and great for rapid prototyping; however, its lower physics realism makes it less ideal for tasks requiring accurate sensor feedback or contact dynamics. Gazebo, by contrast, is widely adopted in robotics and provides a stable, highly configurable environment for robotics development. It's better suited to long-running, realistic simulations, though it lacks native reinforcement learning support and doesn't easily scale to parallel workloads.

Ultimately, we selected NVIDIA's Isaac Sim as our simulation platform. Designed specifically for robotics applications, Isaac Sim provides high-fidelity physics, GPU-accelerated rendering, and the ability to run multiple simulation environments in parallel, enabling efficient large-scale training. To effectively interface with Isaac Sim, we used Isaac Lab, a Python interface layer. This allows us to seamlessly utilize common RL frameworks such as Stable Baselines 3 and SKRL. A significant drawback of Isaac Sim is its heavy computational demand. NVIDIA specifies an RTX 3070 as the minimum supported GPU and recommends an RTX 4080 (~$1,500 CAD) or better for "good" performance.

Despite the advantages of simulation, a key challenge introduced by not working directly with hardware is the sim-to-real gap. Subtle differences between the simulated environment and real-world conditions can lead to poor transfer of learned behaviours. This challenge further motivated our decision to use Isaac Sim, as its high-fidelity physics and sensor models help reduce these discrepancies. Additionally, Isaac Sim supports domain randomization, allowing us to expose the agent to a wide range of variations during training. This improves the model's robustness and increases the likelihood of successful deployment in real-world scenarios.

### 6.2.3. Training Environment Creation

Once the environment and robot models were complete, we developed a new training environment based on the third floor of the Engineering Lab Wing. To structure this space for navigation, we divided it into 49 distinct regions, each representing a physically separated or semantically meaningful location. These regions formed the basis of a waypoint-based navigation system.

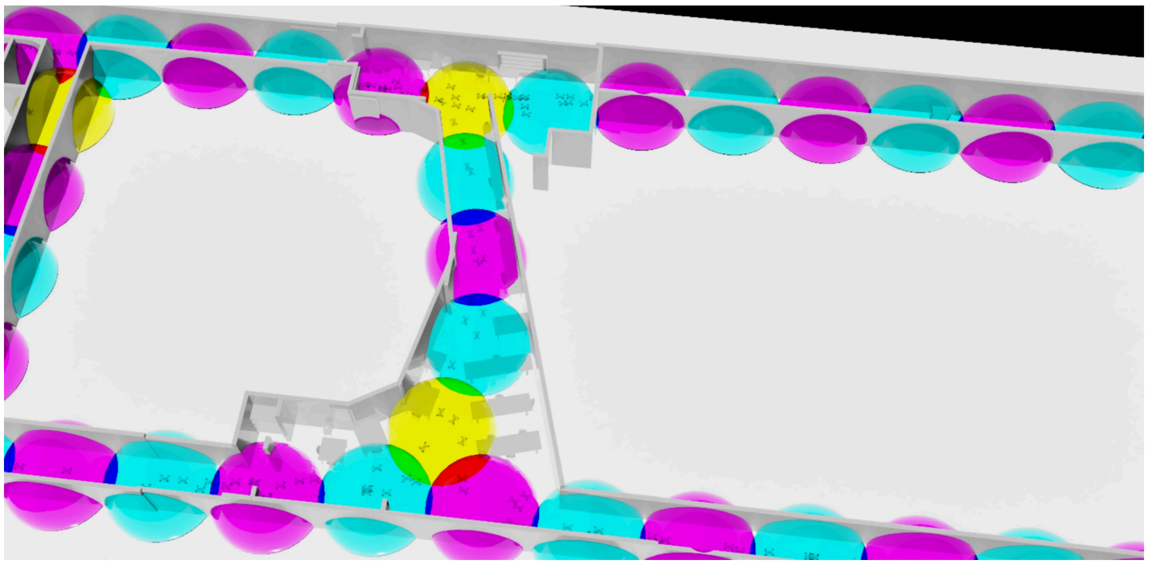


*Figure 7: Regions formed in the Training Simulation.*

Initially, our approach was to train separate models for each navigation goal. The idea was that each model would specialize in reaching a specific destination, and at the start of each episode, the robot would be spawned at a random region along a route to that destination, facing toward the next waypoint. This required the agent to learn to progress through the route step-by-step using local observations.

However, we recognized a critical limitation - routes that require turns of different directions at locations that are perceptually identical to LiDAR were fundamentally impossible without additional context. To resolve this, we introduced a Visual Place Recognition (VPR) system, discussed further in Section 6.3.2, which allowed the robot to more reliably identify its current region among the 49 possibilities.

With this capability in place, we revised our training approach to use a single unified model. Instead of relying on region-specific training, we added two additional features to the agent's observation space: an 8-directional encoding of the relative direction to the next waypoint (e.g., North, South-East). This enabled us to randomize both the starting location and orientation at the beginning of each episode, and simply instruct the agent which direction the next waypoint lies in. Routes between any two regions are generated using an A* pathfinding algorithm, and in cases where no valid route exists, the agent is trained to remain idle.

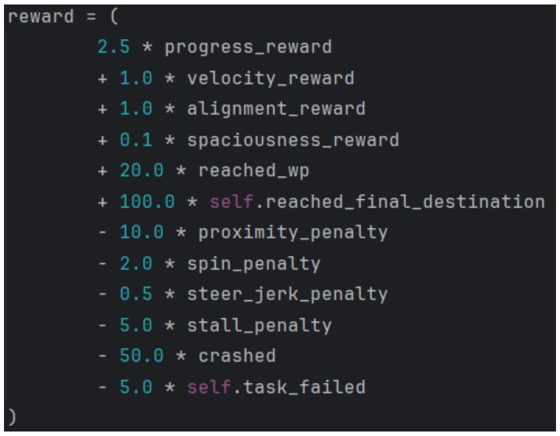

```
reward = (
        2.5 * progress_reward
        + 1.0 * velocity_reward
        + 1.0 * alignment_reward
        + 0.1 * spaciousness_reward
        + 20.0 * reached_wp
        + 100.0 * self.reached_final_destination
        - 10.0 * proximity_penalty
        - 2.0 * spin_penalty
        - 0.5 * steer_jerk_penalty
        - 5.0 * stall_penalty
        - 50.0 * crashed
        - 5.0 * self.task_failed
)
```

*Figure 8: Training Environment Composite Reward.*

The reward function used during training was composed of 12 weighted terms, each designed to reinforce or discourage specific behaviours, as shown in Figure 8. The most significant components included: rewards for progressing along the route and reaching waypoints, incentives for maintaining higher velocity, and penalties for proximity to walls or collisions.

With this finalized environment structure, the robot receives a 36-dimensional observation vector, which is passed into an SKRL Proximal Policy Optimization (PPO) model. The model outputs two continuous action values: throttle and steering. The observation vector includes the two most recent actions to provide temporal context, 28

LiDAR distance readings spaced every 10° from –135° to +135°, the gyroscope's yaw rate, the linear acceleration from the accelerometer, the sine and cosine of the robot's global heading, and two values encoding the cardinal direction of the next waypoint relative to the current position.

### 6.2.4. Narrowing Sim-to-Real Gap

Unfortunately, despite the significant advancements in simulation technology offered by Isaac Sim, it is still not sufficient to fully bridge the sim-to-real gap. Small discrepancies between the simulated environment and real-world conditions often result in trained behaviours failing to transfer effectively to the physical platform, despite strong performance in simulation. This challenge was a persistent issue throughout development.

To mitigate this, we implemented a range of domain randomization techniques intended to improve the robustness of the trained model. Observation and action delays were introduced into the simulation to match the measured latencies on the real system. Gaussian noise and dropout were applied to the simulated LiDAR readings to increase tolerance to sensor noise and modelling inaccuracies. Additionally, randomized biases were added to both throttle and steering commands to simulate discrepancies between the simulated car and the final platform.

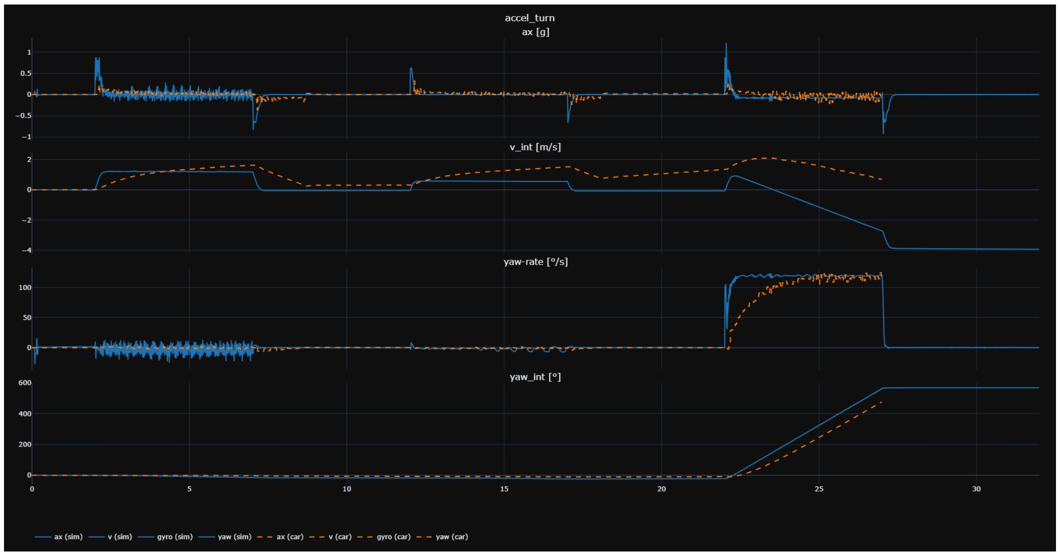


Figure 9: Graphs of IMU Performance.

To reduce the gap between the physical characteristics of the platform and the simulated car, we ran identical drive profiles on both systems and recorded the resulting IMU data. We then overlaid and compared these plots, as shown in Figure 9, to iteratively refine the simulated car's behaviour until it closely matched the dynamics of the real platform.

Despite these measures, we continued to encounter overfitting-related issues. While the model could complete all simulated point-to-point routes reliably within the simulation after extended training, its performance degraded in the real world. This indicates a sensitivity to imperceptible simulation-specific details that do not generalize. Further analysis is provided in Section 8 – Testing & Validation.

## 6.3. Platform Software and Integration

To run the final trained model on the platform and support additional tasks such as the Visual Place Recognition system and the OpenAI scene description script, we developed a web app frontend using Plotly Dash. This frontend interfaces with a centralized controller class, which initializes and manages a collection of subsystems and drivers. Each subsystem inherits from a common base class that defines a consistent initialization structure and interaction scheme, ensuring clean and modular integration across the following independent components. Each of the following subsystems is designed to run in asynchronously threaded loops to ensure non-blocking operation and responsive performance across the entire system.

### 6.3.1. Inference Loop

This is the most critical subsystem, responsible for processing observations from the environment and computing the appropriate action using a GaussianPolicy neural network. The resulting action is returned to the controller, which monitors the Pi's GPIO to check if the trigger on the cane's handle is depressed. If so, it issues throttle and steering commands to the Electronic Speed Controller and servos via our PWMController driver.

### 6.3.2. Visual Place Recognition

The Visual Place Recognition (VPR) subsystem handles localization. The controller passes an input image from the Pi Cam driver to the VPR script, which converts it into an embedding using the *efficientnet_b0* visual transformer [13]. This embedding is then compared via cosine similarity against a database of 2,450 precomputed embeddings, representing 50 embeddings per identified region. These were generated from over 18,000 images captured throughout the ELW building.

The VPR subsystem returns a ranked list of the most probable regions along with confidence values for each. While the full point-to-point navigation system would use these outputs to determine cardinal directions toward the next waypoint, our current implementation allows the user to define a stopping destination. As the generalized navigation model drives, if the VPR identifies the specified region with confidence above a set threshold, the car applies full braking to signify that it has reached the target destination.

### 6.3.3. OpenAI Scene Descriptor

The controller monitors the state of a secondary switch on the cane. When this switch is pressed, the most recent image captured by the camera driver is sent to the observer subsystem. The image is resized, compressed, and base64-encoded before being sent to the OpenAI image recognition endpoint with a prompt tailored for users with visual impairments.

The resulting scene description is returned by the API, then sent to OpenAI's text-to-speech endpoint to generate an MP3 audio file. This file is returned to the controller, passed to the frontend, and finally received by a JavaScript function that autoplays the audio for the end user directly from the web app.

### 6.3.4. Shared State Subsystem

To facilitate reliable communication between subsystems in a multithreaded environment, we implemented a centralized telemetry subsystem. This shared state manager ensures thread-safe data exchange, avoiding race conditions and enabling clean, consistent interaction across all components. Each subsystem reads from and writes to this centralized structure to publish internal state and access data from other parts of the system.

## 7. Design & Prototype

The working prototype depicted in Figure 10 below combines an ergonomic cane and a lightweight smart RC-car platform to deliver autonomous indoor guidance and real-time scene narration for visually-impaired users. Every major subsystem specified in Section 3 was realized and integrated. A summary of how the prototype satisfies or partially satisfies the project objectives is shown below in Table 1.

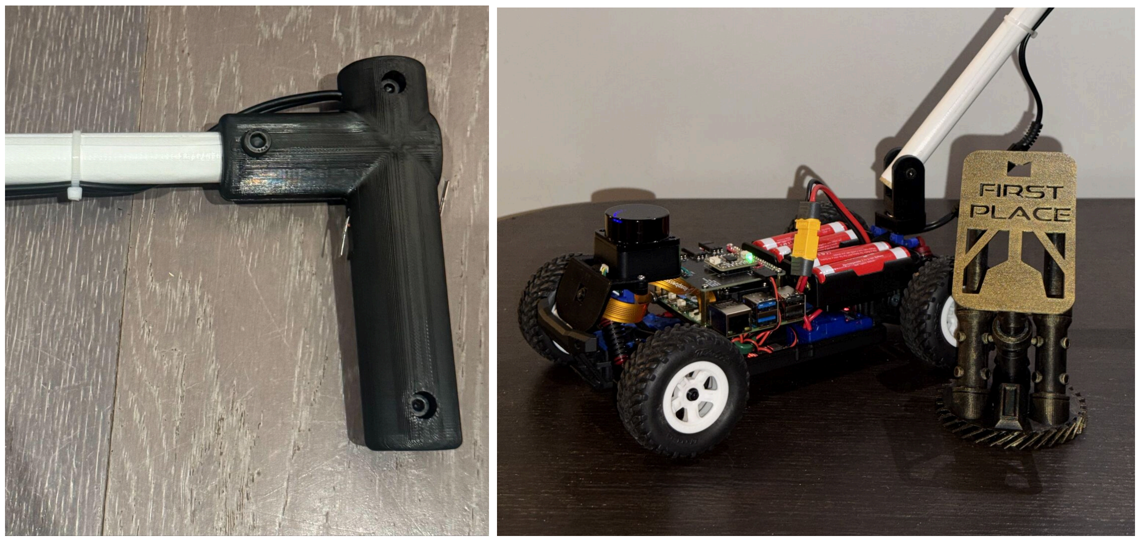


*Figure 10: A Photo of the Cane (left) and AutoNav (right).*

*Table 1: A Summary of the Objective Criteria.*

| Objectives | Status | Notes |
|---|---|---|
| Training a Proximal Policy Optimization (PPO) reinforcement learning model for autonomous navigation from an accurate simulation environment and robot model in Nvidia's Isaac Lab | Met | We produced our RC chassis and corridor environments in Isaac Lab accurately, and trained a PPO policy for ~ 6 million steps with domain randomization and transferred the quantised weights to the Pi 5, where the corridors at around 0.9m/s with less than 10% collisions. |
| Integrating a planar LiDAR sensor and Camera for environmental scanning and real-time scene analysis | Met | A LiDAR sensor supplies obstacle vectors while a Pi Camera streams frames to OpenAI's Vision and Text-to-Speech functionality for scene descriptions. Combined, they yield an 80ms description latency and a 1.8s end-to-end scene narration. |
| Develop control logic for the user interface | Met | A dual trigger handle feeds a finite-state controller on the Pi, allowing users to toggle between autonomous navigation or scene description with spoken feedback via Bluetooth. |
| Design and build an autonomous vehicle capable of path navigation and obstacle avoidance | Partially Met | The autonomous navigation platform, powered by a Pi 5 and Lithium-ion pack completes ~2 hour runs, executing PPO-based steering and LiDAR safety bubbles that clear 90 degree hallway turns and static obstacles. |

The fourth objective outlined in Table 1 above was only partially achieved, primarily due to time constraints. Although AutoNav reliably detects and avoids obstacles, it does not yet compute an optimal trajectory to the user-specified destination; instead, it reaches the intended path via a sub-optimal route. Testing also revealed a bias towards right-hand turns. This issue can be slated for correction through additional data collection, policy retraining and more time.

Because the reinforcement-learning policy was trained exclusively in indoor environments, system performance degrades in some outdoor areas. In open areas lacking corridor boundaries, sporadic LiDAR returns can induce looping in circular patterns. Within buildings, however, AutoNav operates robustly and fulfills its primary design objective.

## 8. Testing & Validation

Due to the challenges outlined in Section 6.2.4. – *Narrowing the Sim-to-Real Gap*, the model that successfully completed all point-to-point navigation tasks within the ELW simulation was ultimately never fully functional on hardware within the limited 2.5-month development timeline. Despite extensive domain randomization and simulation tuning, subtle discrepancies between the simulated and real-world environments consistently prevented reliable policy transfer.

To address this limitation, we shifted focus toward training a simplified, generalized navigation model. Rather than following predefined waypoint routes, this policy was trained to navigate down hallways, avoid obstacles, and maintain forward progress without relying on precise localization or route planning. Surprisingly, this lightweight model performed well in real-world tests, demonstrating reliable hallway-following behaviour and robust obstacle avoidance across various scenarios. While it lacks the goal-directed intelligence of the original approach, it served as a practical and effective fallback, enabling a functional demonstration of autonomous navigation under real-world conditions.

The final prototype, equipped with the generalized model, was evaluated in both known and unknown settings. Trials took place in the Engineering Lab Wing (ELW) which is a known environment to the model as well as in several other UVic campus buildings and outdoor locations that were unfamiliar to the model as illustrated in Figures 11, 12 and 13. Additional outdoor tests were also conducted to assess the prototype's robustness under varied lighting and terrain conditions.

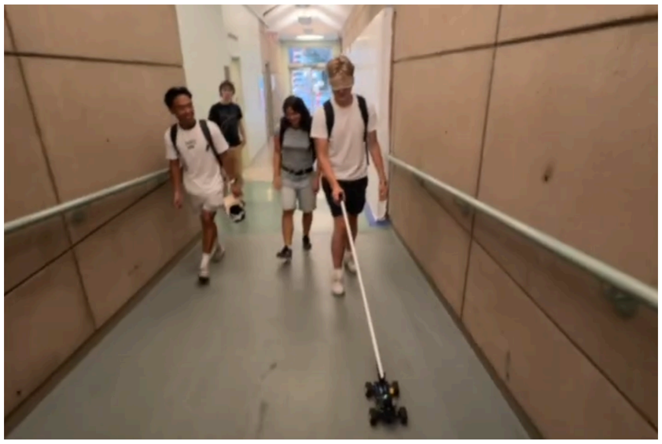


*Figure 11: Mattias testing the Generalized Model in a known indoor-environment.*

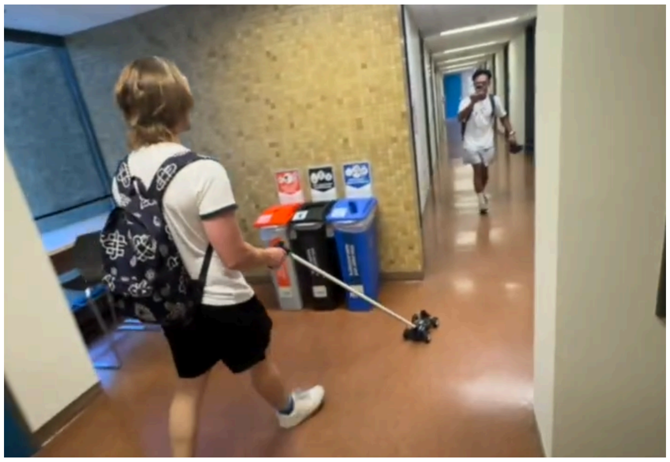


*Figure 12: Eiden testing the Generalized Model in an unknown indoor-environment.*

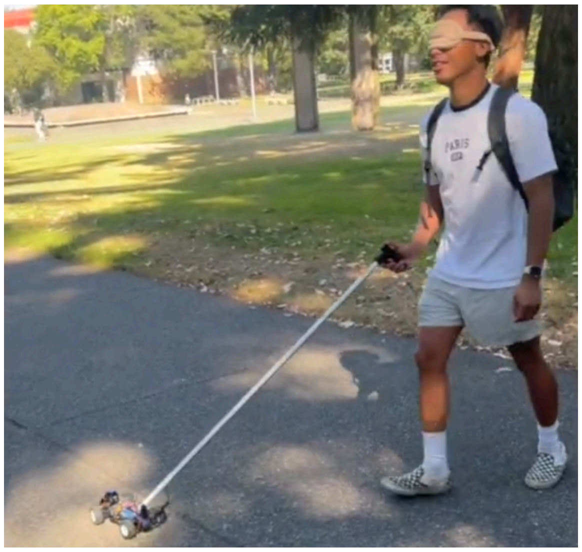


*Figure 13: Jairus testing the Generalized Model outside.*

AutoNav consistently detected and avoided obstacles; however, the system showed a tendency to favour right-hand turns and only rarely executed left turns. This bias is due to a training set that emphasized right-turn avoidance maneuvers. Interestingly, the prototype performed better in the unfamiliar Clearihue building than in ELW, and it maintained reliable obstacle detection and avoidance outdoors, provided that the LiDAR sensor can detect obstacles in its vicinity.

The Scene-Description subsystem was validated in a controlled indoor environment. Sample imagery captured by the Pi Camera is presented in Figure 14, while the corresponding caption output is displayed in Figure 15. Results met expectations, demonstrating accurate scene summaries suitable for user feedback.

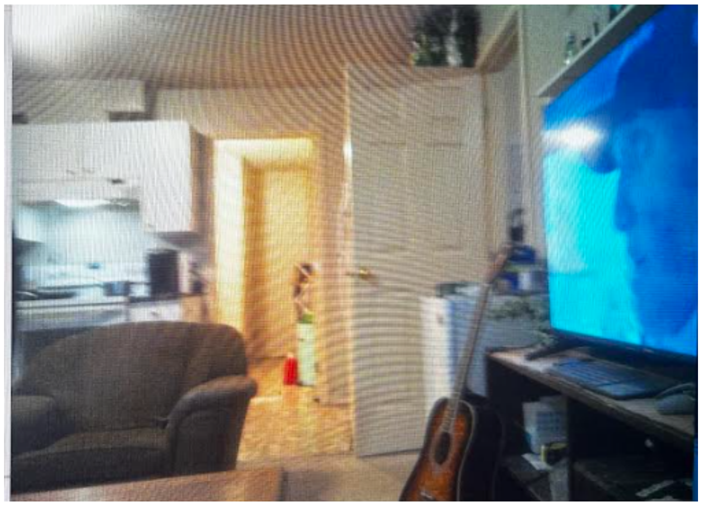
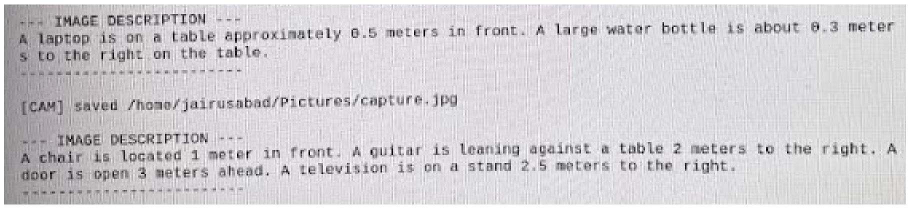


*Figure 14: Pi Camera's Output during the Testing Stage.*



```
--- IMAGE DESCRIPTION ---
A laptop is on a table approximately 0.5 meters in front. A large water bottle is about 0.3 meter
s to the right on the table.
-------------------------

[CAM] saved /home/jairusabad/Pictures/capture.jpg

--- IMAGE DESCRIPTION ---
A chair is located 1 meter in front. A guitar is leaning against a table 2 meters to the right. A
door is open 3 meters ahead. A television is on a stand 2.5 meters to the right.
-------------------------
```

*Figure 15: OpenAI API's Text Output during the Testing Stage.*

## 9. Cost Analysis

The cost analysis was broken down into three separate sections; the direct costs, the indirect costs, and the funding received.

### 9.1. Direct Costs

The direct expenses incurred in the completion of this project include purchases of various hardware components, materials, and supplies detailed below in Table 2.

*Table 2: A Table of Direct Costs for AutoNav.*

| Item | Cost (CAD) |
|---|---|
| LaTrax Prerunner | $223.99 |
| Raspberry Pi 5 8GB | $156.79 |
| LD19 LiDAR | $116.37 |
| Adafruit BNO085 IMU | $49.67 |
| 4x 18650 Cells | $31.07 |
| 6x 18650 + Charger | $38.07 |
| UBEC Regulator | $26.44 |
| JLC Custom PCB | $19.50 |
| Pi Terminal Breakout Hat | $21.26 |
| Battery Holders | $16.79 |
| XT60H Battery Connectors | $16.79 |
| Servo Leads | $8.28 |
| M2.5 Standoff Kit | $17.91 |
| 1KG Polymaker PLA | $38.07 |
| Stacking Header Kit | $20.14 |
| **Total Direct Costs** | **$801.14** |

## 9.2. Indirect Costs

Indirect costs include the value of labour hours invested, consultation fees, and overtime. The team's cumulative labour hours are as follows:

- Mattias' hours: ~30 hours/week for 2.5 months at $40/hour

$$25.6\ hours/week \times 10\ weeks\ =\ 256\ hours$$
$$\$40/hour \times 256\ hours\ =\ \$10,240$$

- Eiden's hours: ~10 hours/week for 2.5 months at $40/hour

$$11\ hours/week \times 10\ weeks\ =\ 110\ hours$$
$$\$40/hour \times 110\ hours\ =\ \$4,400$$

- Jairus' hours: ~12 hours/week for 2.5 months at $40/hour

$$15\ hours/week \times 10\ weeks\ =\ 150\ hours$$
$$\$40/hour \times 150\ hours\ =\ \$6,000$$

$$\sum Each\ partner's\ hourly\ wage$$
$$\$10,240\ +\ \$4,400\ +\ \$6,000\ =\ \$20,640$$

## 9.3. Funding

Funding received for this project includes:

- Electrical and Computer Engineering Department Grant: $120.00
- First Place Prize Money: $375.00

Total Funding: $495.00

## 9.4. Net Cost

$$Direct\ costs\ +\ Indirect\ costs\ -\ Funding\ =\ Net\ cost$$
$$\$801.14\ +\ \$20,640\ -\ \$495.00\ =\ \$20,946.14$$

## 9.5. Tentative Pricing & Return on Investment (ROI)

Considering the total net cost of approximately $21,000, a viable market price must factor in at least a 50 to 60% markup to cover manufacturing, additional research and development, marketing, distribution, and profit margins. The exact pricing strategy will be refined through comprehensive market research, competitive analysis, and potential customer feedback. It is important to consider that the process can now be greatly reduced in hours since the structure of the product does not have to be recreated for each item. Arguably cutting the hours required to recreate this in half or to a third. The expected return on investment will be calculated based on anticipated sales volumes, production scalability, and market adoption rates, ensuring profitability and sustainability.

### 9.6. Cost Reduction

Potential methods to reduce prototype costs include utilizing lower-cost alternative components with similar capabilities, and ordering components in bulk or from suppliers offering educational discounts. Additionally, incorporating open-source software and components can lower licensing and development expenses. Streamlining the overall design to eliminate redundancy and unnecessary complexity can further enhance cost efficiency, beneficial for further versions and possibly commercialization.

## 10. Conclusion & Recommendations

The AutoNav project successfully achieved its primary goal of designing and implementing a functional, low-cost autonomous navigation aid for individuals with visual impairments. All core objectives, including robotic platform development, environment mapping, user interface integration, and scene description, were met through an interdisciplinary approach that prioritized user safety, real-world applicability, and ethical engineering standards as guided by the EGBC Code of Ethics.

Through a combination of reinforcement learning, LiDAR/Camera-based localization, and camera-powered scene understanding, the final prototype demonstrated robust indoor navigation and real-time auditory feedback capabilities despite hardware limitations.

Several challenges were encountered throughout the design process. Power instability, hardware mounting limitations, limited computing power of the Raspberry Pi, and difficulty parsing real-time sensor data required interactive refinement of both electrical and mechanical subsystems – all of which were diagnosed and resolved, leading to a more reliable and consistent final prototype.

For future iterations, the following improvements are recommended:

1. Upgrade to a more powerful onboard computing platform, such as the Nvidia Jetson series, to enhance inference speed and generalization.
2. Replace the camera module with a high dynamic range (HDR) camera to improve scene description in varied lighting conditions.
3. Improve battery management and power delivery circuits for longer runtime.
4. Expand testing into larger and more complex environments, including stairways, elevators, and areas with heavy foot traffic.
5. Integrate user feedback loops from actual visually impaired users to guide further ergonomic and interaction design enhancements.
6. Modularize the software even further for plug-and-play sensor support and easier field upgrades.

AutoNav not only addressed a critical accessibility need but also demonstrated how student-led engineering projects can align with professional standards and societal impact goals. This prototype stands as a foundation for future iterations and broader research into ethical, AI-driven assistive technologies.

## References

[1] “Raspberry Pi 5,” Raspberry Pi, [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-5/ [Accessed Jul. 30, 2025].

[2] “DTOF LIDAR LD19,” DTOF LIDAR LD19 - Waveshare Wiki, [Online]. Available: https://www.waveshare.com/wiki/DTOF_LIDAR_LD19 [Accessed Jun. 6, 2025].

[3] “Adafruit 9-DOF orientation IMU Fusion Breakout - BNO085 (BNO080),” Adafruit, [Online]. Available: https://www.adafruit.com/product/4754 [Accessed Jul. 28, 2025].

[4] Raspberry Pi Foundation, “Getting Started with Picamera,” Raspberry Pi Projects, [Online]. Available: https://projects.raspberrypi.org/en/projects/getting-started-with-picamera/0 [Accessed: Jun. 20, 2025].

[5] “OpenAI platform,” Images and Vision, [Online]. Available: https://platform.openai.com/docs/guides/images-vision?api-mode=responses [Accessed Jul. 28, 2025].

[6] Engineers and Geoscientists BC, “Code of Ethics,” Engineers and Geoscientists BC, [Online]. Available: https://www.egbc.ca/Complaints-Discipline/Code-of-Ethics/Code-of-Ethics. [Accessed: 20-Jun-2025].

[7] “Pla filament,” Filaments.ca, [Online]. Available: https://filaments.ca/collections/pla-filament?srsltid=AfmBOooeAZH5SKRDsVDkNVZ8q96mfR7AOzweX0aEeOxKlTBptv_eMXwl [Accessed Jul. 28, 2025].

[8] “Self-guided mobility device for blind and low vision,” Glidance, [Online]. Available: https://glidance.io/. [Accessed: 20-Jun-2025].

[9] A. Myers, “Stanford researchers build $400 self-navigating smart cane,” Stanford HAI, [Online]. Available: https://hai.stanford.edu/news/stanford-researchers-build-400-self-navigating-smart-cane. [Accessed: 20-Jun-2025].

[10] “About the UltraCane,” UltraCane, [Online]. Available: https://www.ultracane.com/about_the_ultracane [Accessed Jul. 28, 2025].

[11] J. Ingber, “Feel the buzz of the Buzzclip,” The American Foundation for the Blind, [Online]. Available: https://www.afb.org/aw/18/3/15228 [Accessed Jul. 29, 2025].

[12] “Proximal Policy Optimization (PPO),” Proximal Policy Optimization (PPO) - skrl (1.4.3), [Online]. Available: https://skrl.readthedocs.io/en/latest/api/agents/ppo.html [Accessed Jul. 28, 2025].

[13] M. Tan and Q. V. Le, “EfficientNet: Rethinking model scaling for Convolutional Neural Networks,” [Online]. Available: https://arxiv.org/abs/1905.11946 [Accessed Jul. 28, 2025].

[14] “Jetson Orin Nano Developer Kit Getting Started Guide,” NVIDIA Developer, [Online]. Available: https://developer.nvidia.com/embedded/learn/get-started-jetson-orin-nano-devkit [Accessed Jul. 29, 2025].

[15] “ESP32,” ESP32 Wi-Fi & Bluetooth SoC | Espressif Systems, [Online]. Available: https://www.espressif.com/en/products/socs/esp32 [Accessed Jul. 30, 2025].

[16] “Arduino hardware,” Arduino, [Online]. Available: https://www.arduino.cc/en/hardware/ [Accessed Jul. 30, 2025].

[17] “Latrax Desert Prerunner,” Traxxas, [Online]. Available: https://traxxas.com/76064-5-latrax-desert-prerunner [Accessed Jul. 30, 2025].

[18] MuammerBay, “Muammerbay/Leatherback,” GitHub, [Online]. Available: https://github.com/MuammerBay/Leatherback [Accessed Jul. 29, 2025].

[19] “Bullet and PyBullet,” Bullet RealTime Physics Simulation, [Online]. Available: https://pybullet.org/wordpress/ [Accessed Jul. 30, 2025].

[20] “Simulate before you build,” Gazebo, [Online]. Available: https://gazebosim.org/home [Accessed Aug. 1, 2025].

[21] NVIDIA, “Isaac Gym and Sim-to-Real Transfer for Reinforcement Learning,” NVIDIA Developer, [Online]. Available: https://developer.nvidia.com/isaac-gym. [Accessed: Jun. 23, 2025].

## A.1 Appendix A – EGBC Code of Ethics

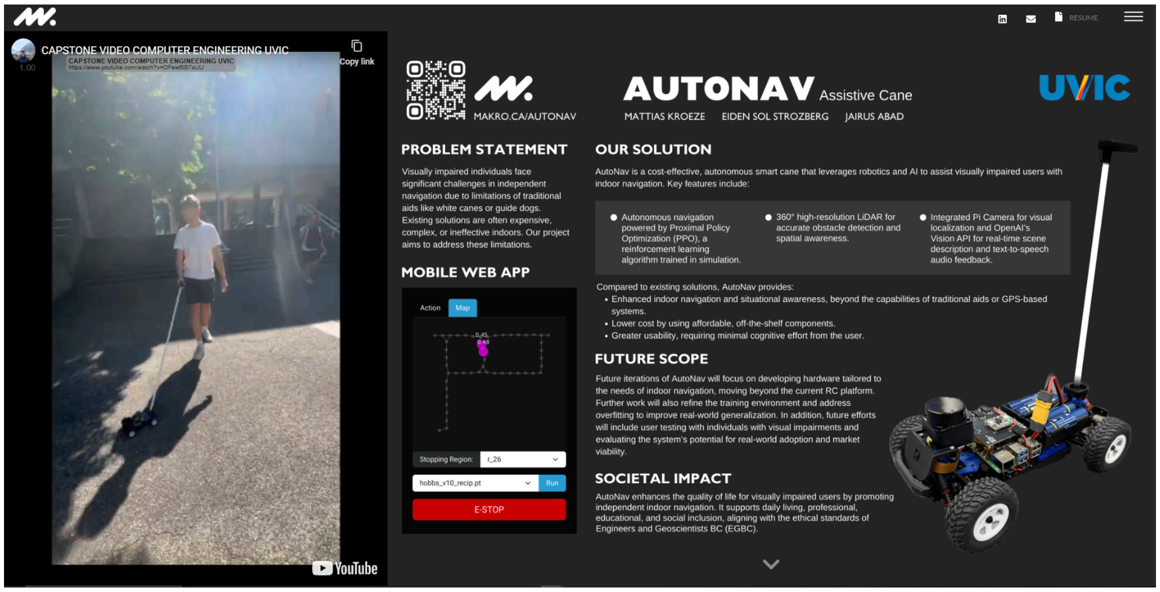
A registrant must adhere to the following Code of Ethics:

Registrants must act at all times with fairness, courtesy and good faith toward all persons with whom the registrant has professional dealings, and in accordance with the public interest. Registrants must uphold the values of truth, honesty, and trustworthiness and safeguard human life and welfare and the environment. In keeping with these basic tenets, registrants must:

1. Hold paramount the safety, health, and welfare of the public, including the protection of the environment and the promotion of health and safety in the workplace;
2. Practice only in those fields where training and ability make the registrant professionally competent;
3. Have regard for the common law and any applicable enactments, federal enactments, or enactments of another province;
4. Have regard for applicable standards, policies, plans, and practices established by the government or Engineers and Geoscientists BC;
5. Maintain competence in relevant specializations, including advances in the regulated practice and relevant science;
6. Provide accurate information in respect of qualifications and experience;
7. Provide professional opinions that distinguish between facts, assumptions, and opinions;
8. Avoid situations and circumstances in which there is a real or perceived conflict of interest and ensure conflicts of interest, including perceived conflicts of interest, are properly disclosed and necessary measures are taken so a conflict of interest does not bias decisions or recommendations;
9. Report to Engineers and Geoscientists BC and, if applicable, any other appropriate authority, if the registrant, on reasonable and probable grounds, believes that:
    a. The continued practice of a regulated practice by another registrant or other person, including firms and employers, might pose a risk of significant harm to the environment or to the health or safety of the public or a group of people; or
    b. A registrant or another individual has made decisions or engaged in practices which may be illegal or unethical;
10. Present clearly to employers and clients the possible consequences if professional decisions or judgments are overruled or disregarded;
11. Clearly identify each registrant who has contributed professional work, including recommendations, reports, statements, or opinions;
12. Undertake work and documentation with due diligence and in accordance with any guidance developed to standardize professional documentation for the applicable profession; and
13. Conduct themselves with fairness, courtesy, and good faith towards clients, colleagues, and others, give credit where it is due and accept, as well as give honest and fair professional comment.

## B.1 Appendix B – Website Link

The Website link of the project can be found in the link: https://makro.ca/autonav. The main page image of the image is illustrated below in Figure B.1.



AUTONAV Assistive Cane

MATTIAS KROEZE EIDEN SOL STROZBERG JAIRUS ABAD

PROBLEM STATEMENT

Visually impaired individuals face significant challenges in independent navigation due to limitations of traditional aids like white canes or guide dogs. Existing solutions are often expensive, complex, or ineffective indoors. Our project aims to address these limitations.

MOBILE WEB APP

OUR SOLUTION

AutoNav is a cost-effective, autonomous smart cane that leverages robotics and AI to assist visually impaired users with indoor navigation. Key features include:

- Autonomous navigation powered by Proximal Policy Optimization (PPO), a reinforcement learning algorithm trained in simulation.
- 360° high-resolution LiDAR for accurate obstacle detection and spatial awareness.
- Integrated Pi Camera for visual localization and OpenAI's Vision API for real-time scene description and text-to-speech audio feedback.

Compared to existing solutions, AutoNav provides:

- Enhanced indoor navigation and situational awareness, beyond the capabilities of traditional aids or GPS-based systems.
- Lower cost by using affordable, off-the-shelf components.
- Greater usability, requiring minimal cognitive effort from the user.

FUTURE SCOPE

Future iterations of AutoNav will focus on developing hardware tailored to the needs of indoor navigation, moving beyond the current RC platform. Further work will also refine the training environment and address overfitting to improve real-world generalization. In addition, future efforts will include user testing with individuals with visual impairments and evaluating the system's potential for real-world adoption and market viability.

SOCIETAL IMPACT

AutoNav enhances the quality of life for visually impaired users by promoting independent indoor navigation. It supports daily living, professional, educational, and social inclusion, aligning with the ethical standards of Engineers and Geoscientists BC (EGBC).

*Figure B.1: The Main Page of the Website.*

## C.1 Appendix C – Youtube Video of AutoNav

A video of testing the AutoNav in multiple environments can be seen in the Youtube link AutoNav Testing Video.