Intelligent Wearable Device for Visually Impaired Individuals

ESE 498/499 Capstone Design Project Formal Proposal

Submitted to Professor Wang and the Department of Electrical and Systems Engineering

April 12, 2025

Client:
Electrical & Systems Engineering Department
James Feher[1] - jdfeher@wustl.edu

Advisor:
Dorothy Wang[2] - dorothyw@wustl.edu

Engineers:
Max Saltrelli[3] - maxsaltrelli@wustl.edu
Jenny Lin[4] - jennyl@wustl.edu

Washington University in St. Louis
McKelvey School of Engineering

[1]Teaching Professor, Department of Electrical and Systems Engineering

[2]Lecturer, Department of Electrical and Systems Engineering

[3]Candidate for B.S. in Computer Engineering with minors in Robotics and Architecture

[4]Candidate for B.S. in Systems Science and Engineering with a second major in Physics and minors in Electrical Engineering and Music

Abstract

This project is an assistive device designed to improve visually impaired persons' ability to safely and efficiently navigate through their daily environments. This device operates utilizing a 360-degree array of VL53L1X LIDAR TOF depth sensors, whose output drives haptic motors to provide directional tactile feedback via vibration. Sensor readings are interpreted using advanced signal processing techniques such as Kalman filtering and moving average filtering to improve accuracy, along with machine learning models like XGBoost for object recognition. A Python-based simulation provides a visual representation of the system's performance for demo purposes. By combining engineering principles from control systems, signal processing, and machine learning, this project delivers a technically complex wearable assistive technology device while still ensuring accessibility, usability, and affordability.

1 Introduction

Individuals who are visually impaired face significant challenges in navigating their daily environments safely and independently. Existing assistive technologies often fall short due to their intrusive design, limited functionality, or lack of discretion, making them less appealing for users [1]. For our capstone project, we developed an innovative solution that combines functionality with user comfort and discreet design. The aim of this project is to create a lightweight, non-intrusive wearable device that leverages 360-degree depth sensing and haptic feedback to provide real-time environmental awareness. By enabling users to perceive their surroundings without relying on vision, this device enhances mobility, confidence, and independence for visually impaired individuals.

This project was developed in collaboration with the St. Louis Society for the Blind and Visually Impaired and was based on feedback from elderly individuals, whose input guided key design decisions around usability, comfort, and real-world functionality.

The primary objective of the Computer Engineering (CE) aspect of this project was developing the hardware and software required to manage the sensor input and haptic motor output of our vest-style wearable device. On this device, an Arduino Portenta H7 board takes in consistent data from eight VL53L1X LIDAR TOF sensors, communicates with a Python visualization as necessary, and outputs haptic feedback over eight haptic sensors based on the input of the depth sensors. This device runs all necessary code on-board, meaning that it does not need to connect to the internet to operate and can work offline. This code not only involves the sensor input and motor output but also the data management and machine learning as described below in the Systems Engineering aspect of this project. A major objective of the hardware aspect of this project was to keep it lightweight and unobtrusive for the user. This means that we were forced to refrain from using heavy or cumbersome components and to hide our hardware components from outside viewers as much as possible.

The primary objective of the Systems Engineering (SE) component of this project was to develop a framework that integrated theory and application for data collection, processing, and interpretation. We specifically addressed three critical aspects: signal processing, machine learning, and mathematical modeling. Signal processing focused on refining raw sensor data by minimizing noise, filtering out irrelevant fluctuations, and extracting meaningful information for obstacle detection. Machine learning techniques were applied to recognize objects based on sensor readings and to determine whether the object's shape was rectangular or circular. Mathematical modeling then extended the belt application into a theoretical framework for sensor networks with uncertain propagation channels and employed algorithms designed for simulation and error modeling.

2 Technical Methods

To ensure accurate and reliable depth sensing, we apply both SE and CSE techniques. Below, we outline the key methods used in our project.

2.1 Overall System

The overall system of the device runs on an Arduino Portenta H7 dual-core microcontroller. This board takes in the distance data (converted to mm) from the VL53L1X depth sensors and controls the haptic output from the DC3V vibration motors. The Arduino Portenta H7 also runs the signal processing and machine learning based on the sensor data (further explored in Sections 2.5 and 2.6).

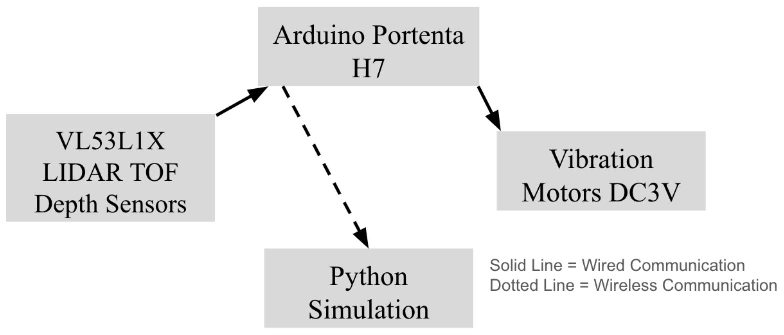

Arduino Portenta H7
VL53L1X LIDAR TOF Depth Sensors
Vibration Motors DC3V
Python Simulation
Solid Line = Wired Communication
Dotted Line = Wireless Communication

Figure 1: This block diagram shows the different major hardware components in this wearable device and how they communicate between each other.

The final task the microcontroller accomplishes is communicating wirelessly via the Portenta H7's Wi-Fi Access Point with a Python visualization of the depth data. An image of the visualization output can be seen in Figure 2. This visualization provides a top-down visualization of the environment the device is in based on the depth sensor input.

The colors serve to not only help to better visualize how close the wearer is to different obstacles, but they also represent the current haptic output. Green represents a safe distance, corresponding to no haptic output. Orange in the visualization indicates that the associated haptic motor is active, vibrating to alert the user of a nearby obstacle. As the detected object gets closer, the color shifts from brighter to darker orange, signaling increasing vibration intensity. Bright red represents the highest level of proximity, at which point the haptic motor is powered with the full 5 volts and delivers maximum vibration.

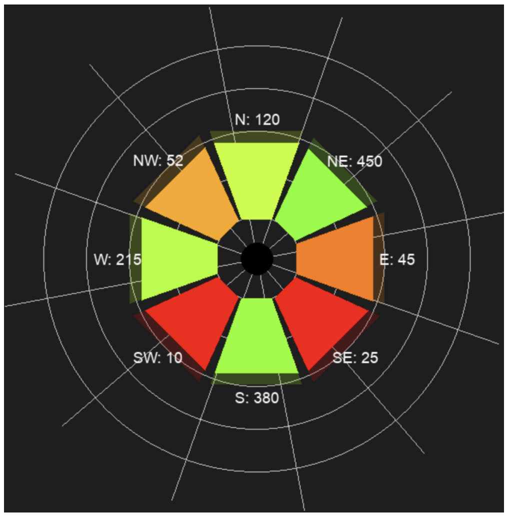


Figure 2: Visualization output

This visualization is solely used for development of the device and for device demos. We decided against the visualization affecting the motor output so that the device will still work when not in range of a laptop running the visualization. This will allow for the device to be used outdoors or in crowded areas with no effect on the user experience.

2.2 Equipment

- **Arduino Portenta H7 Lite Connected** - microprocessor used to control hardware components and run data management code [6].
- **VL53L1X TOF Laser-Ranging Sensor** - depth sensor used to receive distance data.
- **TCA9548A I2C Multiplexer** - manages the use of multiple sensors and haptic motors on the same I2C bus [7].
- **Vibration Motor DC3V 12000rpm** - haptic motor used to provide vibration feedback to the user.
- **ULN2803A Darlington Transistor Array** - controls the current flow between the LiPo battery and the 8 haptic motors [8].
- **3.7V 700mAh 603035 Lipo Battery Rechargable Lithium Polymer Ion Battery Pack** - battery that powers the 8 haptic motors.
- **XL6009 2A DC/DC Step-up Converter** - steps up the LiPo battery from 3.7V to 5V to increase the intensity of the haptic motors.
- **5V 5000mAh Rechargable Portable Power Bank** - powers the Portenta H7, which in turn provides power to the 8 depth sensors and control the motors.

2.3 Configuration

Figure 3 below shows a diagram of the general layout of the hardware components on the front of the vest. As you can see here, the haptic motors and depth sensors are aligned vertically with each other, with there being 8 of each total. Figure 4 below shows an image of the completed vest being worn.

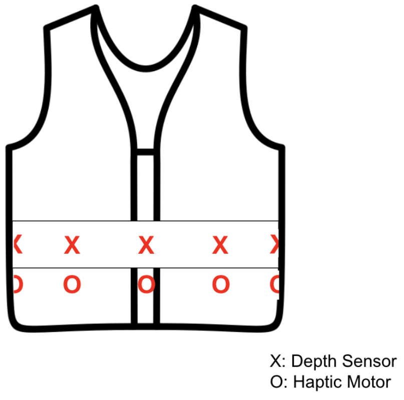


Figure 3: Vest layout diagram (front-view)

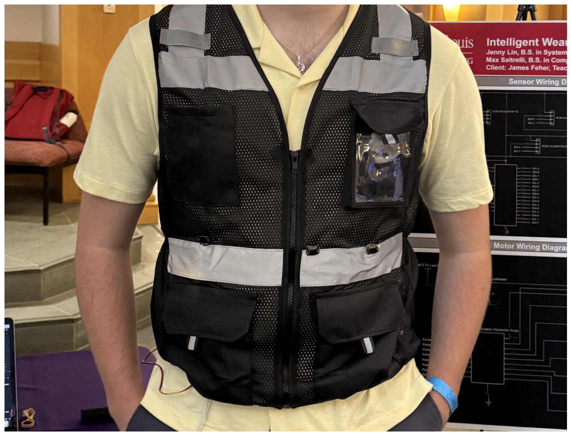


Figure 4: Vest being worn

The Arduino Portenta H7 board, as well as the other components that aren't motors or sensors, are not seen in the diagram or image above, as they are soldered onto a shared breadboard and placed in a pocket on the bottom right of the front of the vest. Figure 5 below shows how the inside of the vest is laid out to account for this.

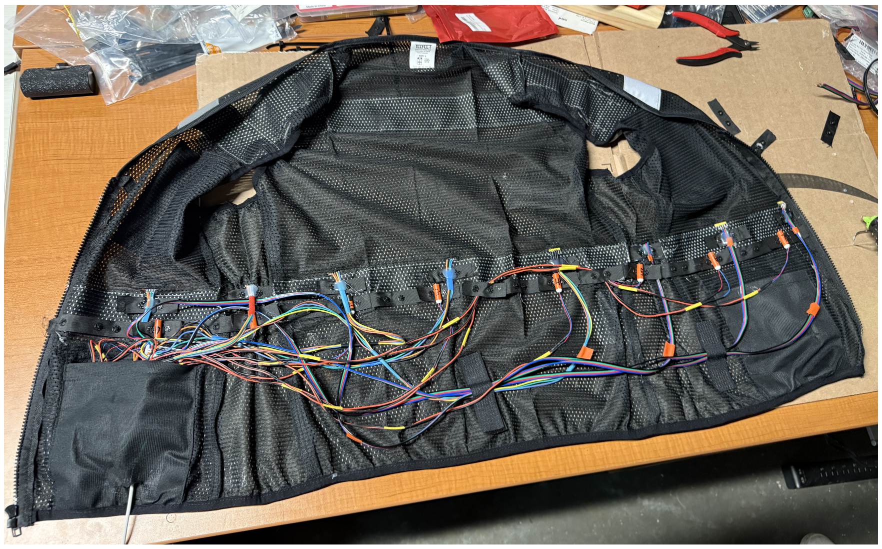


Figure 5: Inside of vest design

The following figures detail the electrical wiring setup for the system. Figure 6 shows how the Portenta H7 controls the haptic motors via PWM and the Darlington transistor array, powered by the 3.7V LiPo battery stepped up to 5V. Figure 7 shows the wiring of the 8 TOF sensors, managed via the TCA9548A I^{2}C multiplexer for independent addressing.

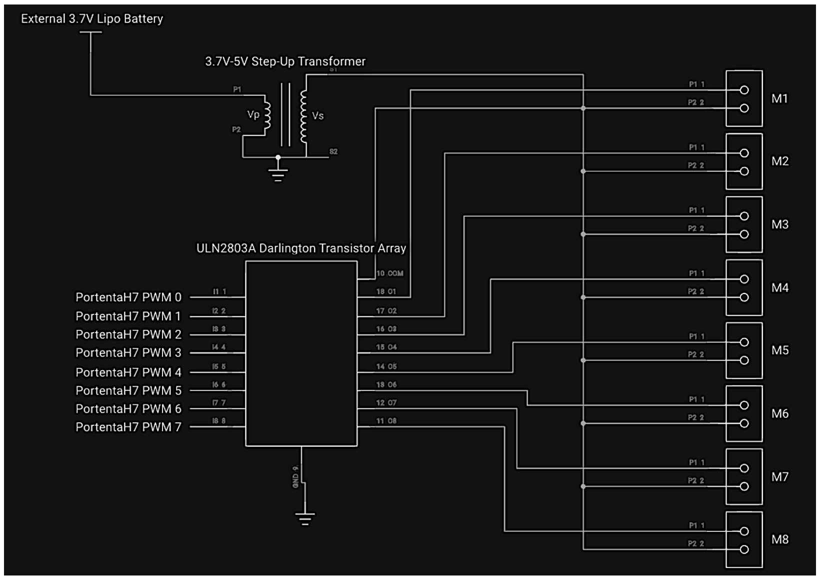


Figure 6: Haptic motor control wiring with ULN2803A and step-up transformer

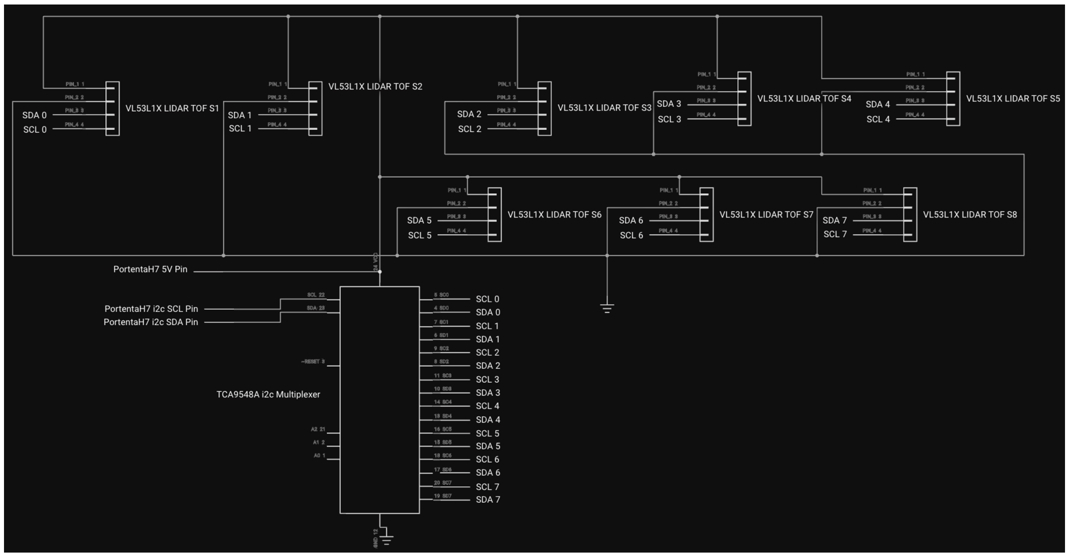


Figure 7: VL53L1X sensor I^{2}C wiring with TCA9548A multiplexer

2.4 Data Requirements and Acquisition

The primary data required for this project is sensor distance measurements, which are taken from the 8 depth sensors simultaneously on the Portenta H7 Connected Lite at a baud rate of 115200. Each sensor operates with a timing budget of approximately 20–100 ms, enabling a per-sensor update rate of up to 50 Hz under optimal conditions (we estimate this was closer to 30 Hz per-sensor due to the machine learning and communication with the visualization running simultaneously). These sensors are evenly spaced at waist-level of the vest to provide 360° of continuous depth sensing data as shown in Figure 8.

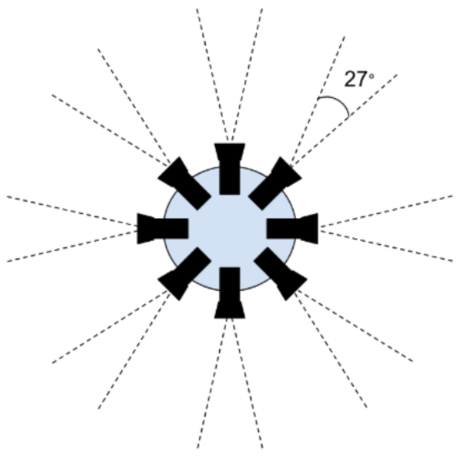


Figure 8: Illustration of the sensor placement

2.5 Signal Processing

To ensure accurate and stable depth sensor readings, we apply multiple signal processing techniques to reduce noise and enhance measurement reliability. Kalman filtering, moving average filtering, and moving median filtering are applied in sequence.

Kalman Filtering is used for real-time sensor data processing. It predicts and corrects each measurement based on previous data, improving stability by reducing sudden jumps or inconsistencies caused by environmental noise and sensor limitations. The steps are [2]:

1. **Prediction Step:** The system predicts the next state based on the previous state and system dynamics.

$$\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} \tag{1}$$

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k \tag{2}$$

where:

- $\hat{x}_k$ is the state vector (dimension: $n \times 1$).
- F_k is the state transition matrix (dimension: $n \times n$).
- P_k is the error covariance matrix (dimension: $n \times n$).
- Q_k is the process noise covariance matrix (dimension: $n \times n$).

2. **Compute the Kalman Gain:** This step determines how much the predicted state should be corrected using the new measurement.

$$K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1} \tag{3}$$

where:

- K_k is the Kalman gain (dimension: $n \times m$), which balances trust between prediction and measurement.
- H_k is the measurement matrix (dimension: $m \times n$), mapping the state space to the measurement space.
- R_k is the measurement noise covariance matrix (dimension: $m \times m$).

3. **Update Step:** The state estimate is updated using the new measurement.

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(z_k - H_k \hat{x}_{k|k-1}) \tag{4}$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \tag{5}$$

where:

- z_k is the measurement vector (dimension: $m \times 1$), containing observed distance readings.
- I is the identity matrix (dimension: $n \times n$), ensuring proper matrix operations when updating uncertainty.

In this project, the Kalman filter is applied to smooth sensor readings from the VL53L1X depth sensors, reducing noise and improving measurement stability. The Kalman gain determines how much the system trusts the sensor data versus the predicted state, ensuring real-time adaptability to environmental changes.

Moving Average Filtering smooths fluctuations by averaging a sliding window of past sensor values, reducing small variations and noise in distance measurements.

Moving Median Filtering is more robust against outliers, effectively handling sudden spikes caused by sensor errors or reflections.

2.6 Machine Learning

Machine learning is utilized to recognize shapes of objects.

Data Acquisition: We simulate a dynamic environment by moving a flat board and a circular object in front of the three forward–looking sensors. This captures real-time readings of different shapes. We focus on the front sensors because users move forward—making that data most relevant—and because one motor per sensor keeps haptic feedback clear and not overwhelming.

Model Selection: We select four classifiers to recognize obstacle types from the sensor data: Support Vector Machine (SVM), Gaussian Naive Bayes (Gaussian NB), Random Forest, and XGBoost. SVM finds a boundary that maximizes the margin between classes and can use kernels for nonlinearity, making it robust for problems with clear separation.

Gaussian NB assumes features are independent and normally distributed, enabling very fast training and good performance even on small datasets. Random Forest builds an ensemble of decision trees on random subsets of data and features, which reduces overfitting and typically yields high accuracy out of the box. XGBoost uses gradient boosting to sequentially add trees that correct previous errors, with built-in regularization to prevent overfitting and deliver state-of-the-art results on structured data.

The algorithm starts with setting labels of rectangular shapes as 1 and circular shapes as 0, and merges them into one dataset. We extract the three front-sensor columns as features and split the data into 80% training and 20% testing sets. We used the following metrics to evaluate each classifier:*Precision* is the proportion of correctly predicted positive instances among all predicted positives. *Recall* is the proportion of correctly predicted positive instances among all actual positives. *F1-score* is the harmonic mean of precision and recall, providing a balanced measure of performance. *AUROC* (Area Under the Receiver Operating Characteristic Curve) measures the model's overall ability to discriminate between classes. We also presented *confusion matrices* (showing true positives, false positives, true negatives, and false negatives), *ROC curves*, and bar charts of these metrics to compare model performance.

2.7 Mathematical Modeling

This work extends the belt application to a theoretical framework for sensor networks operating over propagation channels with uncertainty.

1. Sensor Geometric Model

We assume there are N sensors placed in a two-dimensional plane. Their positions are given by:

$$\mathbf{p}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}, \quad i = 1, 2, \ldots, N.$$

For example, if the sensors are uniformly arranged in a circle with radius R, then each sensor's position is:

$$\mathbf{p}_i = \begin{bmatrix} R\cos\left(2\pi\frac{i-1}{N}\right) \\ R\sin\left(2\pi\frac{i-1}{N}\right) \end{bmatrix}, \quad i = 1, 2, \ldots, N.$$

This arrangement helps cover all directions (360°) and minimizes blind spots.

2. Signal Propagation Model

The signal travels from the sensor to the target and reflects back. Therefore, the total propagation delay for sensor i is:

$$\tau_i = \frac{2\|\mathbf{p}_i - \mathbf{p}_0\|}{c},$$

where:

- $\|\mathbf{p}_i - \mathbf{p}_0\|$ is the one-way Euclidean distance,

- c is the propagation speed of the signal,
- The factor 2 accounts for the round-trip travel time (from sensor to target and back).

Let the transmitted signal be denoted as $s(t)$. The sensor emits this signal, which travels to the target and is then reflected back. The total propagation delay for sensor i is given by:

$$\tau_i = \frac{2\|\mathbf{p}_i - \mathbf{p}_0\|}{c},$$

where $\mathbf{p}_0$ is the target position and c is the signal propagation speed.

The received signal at sensor i can be modeled as:

$$x_i(t) = \alpha_i \, s(t - \tau_i) + n_i(t),$$

where:

- α_i is the amplitude attenuation due to propagation loss and reflection,
- $n_i(t)$ is additive noise, typically modeled as zero-mean Gaussian white noise.

By stacking the received signals from all N sensors, we form the array observation vector:

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_N(t) \end{bmatrix}.$$

Under the far-field approximation and narrowband assumption, the time delay differences among sensors due to the target angle θ can be expressed through a phase term. In such a case, we model the array output as:

$$\mathbf{x}(t) = \mathbf{a}(\theta) \, s(t - \tau) + \mathbf{n}(t),$$

where:

- τ is the nominal delay (e.g., at array center),
- $\mathbf{a}(\theta)$ is the array *steering vector*,
- $\mathbf{n}(t)$ is the stacked noise vector.

For a Uniform Linear Array (ULA) with element spacing d and signal wavelength λ, the steering vector is:

$$\mathbf{a}(\theta) = \begin{bmatrix} 1 \\ e^{j\frac{2\pi}{\lambda} d \sin\theta} \\ e^{j\frac{2\pi}{\lambda} \cdot 2d \sin\theta} \\ \vdots \\ e^{j\frac{2\pi}{\lambda} (N-1) d \sin\theta} \end{bmatrix}.$$

This vector captures the phase shift between elements due to the direction of arrival θ.

3. Error Model

Sensors measure distance by emitting a light signal and timing its return. The measurement error consists of two parts:

1. **Random Noise:** Caused by electronic noise, photon counting errors, or interference. It is modeled as a zero-mean Gaussian random variable:
$$v \sim N(0, \sigma^2).$$

2. **System Bias:** A constant error resulting from sensor miscalibration or environmental factors, denoted by b.

The measurement equation is:
$$z = d_{\text{true}} + b + v,$$
where d_{true} is the true distance, v is the random noise, and b is the bias.

For example, if the sensor accuracy is ± 5 cm, one may use:
$$v \sim N\Big(0, (0.05\,\text{m})^2\Big),$$
and if there is a fixed bias of 2 cm, then $b = 0.02\,\text{m}$.

Assume that the target moves with constant velocity. The state vector is defined as:
$$x_k = \begin{bmatrix} d_k \\ \dot{d}_k \end{bmatrix},$$
where d_k is the true distance at time k and $\dot{d}_k$ is the velocity.

The state update model over a time step Δt is:
$$\begin{aligned} d_{k+1} &= d_k + \dot{d}_k \Delta t + w_k^{(d)}, \\ \dot{d}_{k+1} &= \dot{d}_k + w_k^{(\dot{d})}, \end{aligned}$$
or in matrix form:
$$x_{k+1} = F\, x_k + w_k, \quad \text{with} \quad F = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix},$$
and the process noise:
$$w_k = \begin{bmatrix} w_k^{(d)} \\ w_k^{(\dot{d})} \end{bmatrix} \sim N(0, Q).$$

Since the sensor directly measures the distance, the measurement model is:
$$z_k = H\, x_k + v_k, \quad \text{with} \quad H = \begin{pmatrix} 1 & 0 \end{pmatrix},$$
and $v_k \sim N(0, R)$ (e.g., $R = (0.05\,\text{m})^2$).

3 Results

3.1 Vest Design and Haptic Feedback

Figures 9 & 10 below show the finished design of the wearable vest. Overall, the vest design successfully met our core objectives of comfort, wearability, and sensor integration. The haptic feedback was consistently easy to feel and interpret, with users able to quickly understand the direction of nearby obstacles based on motor activation. This confirmed the effectiveness of vibration-based feedback as a spatial cue for real-time navigation.

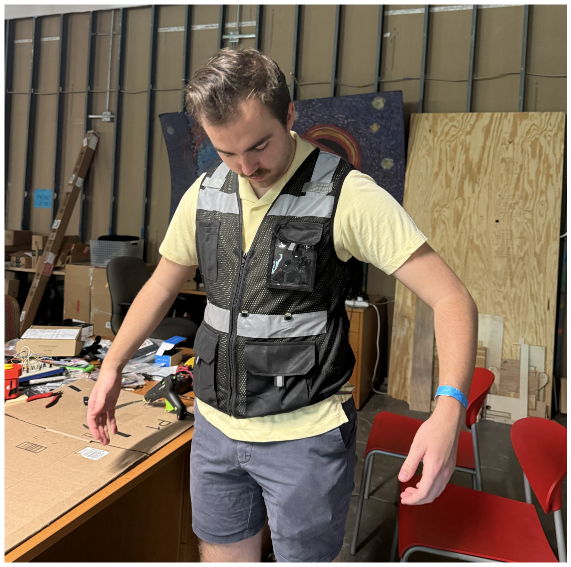


Figure 9: Vest front-view

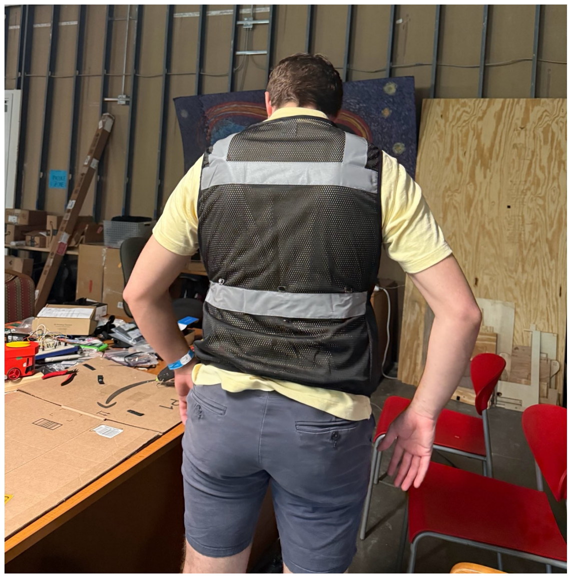


Figure 10: Vest back-view

3.2 Signal Processing

In this experiment, we compared three simple noise-filtering methods on the same extremely noisy sine-wave data: a moving average, which smooths the curve by replacing each point with the average of its neighbors but lags behind sudden changes; a moving median, which effectively removes isolated spikes by using the median in each window but can produce a rough outline around edges; and a one-dimensional Kalman filter, which uses a recursive prediction-update process to reduce noise while closely tracking the true signal with minimal lag. When plotted together, the Kalman filter stays closest to the underlying sine wave, the moving average gives the smoothest but slowest response, and the moving median removes most outliers at the cost of sharper transitions.

Notice that we also integrate signal-processing algorithms into our machine learning workflow. Before training the model, we apply these filters to remove outliers and smooth the data. The figure below shows an example of this preprocessing step used in ML.

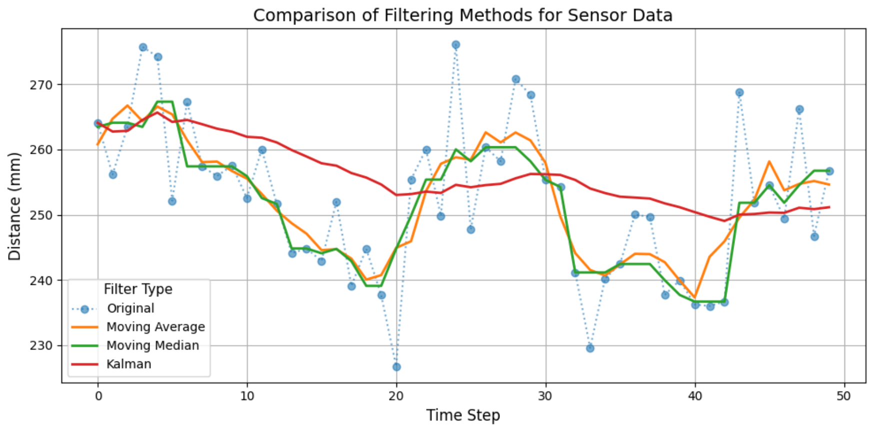


Figure 11: Comparison of Moving Average, Moving Median, and Kalman Filter on Noisy Sensor Data.

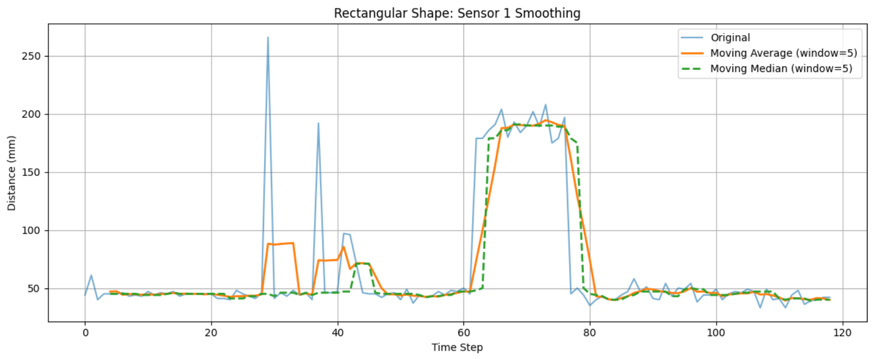


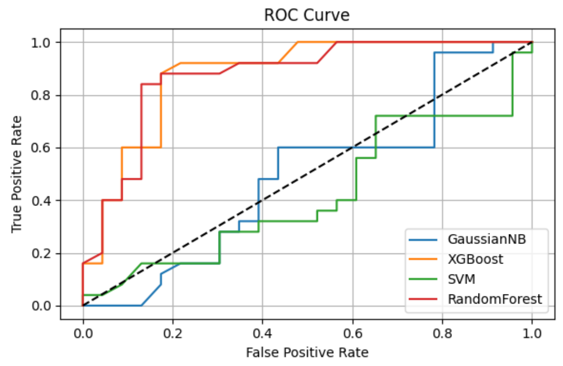
Figure 12: Example of Signal-Processing Preprocessing Applied Before Machine Learning.

3.3 Machine Learning

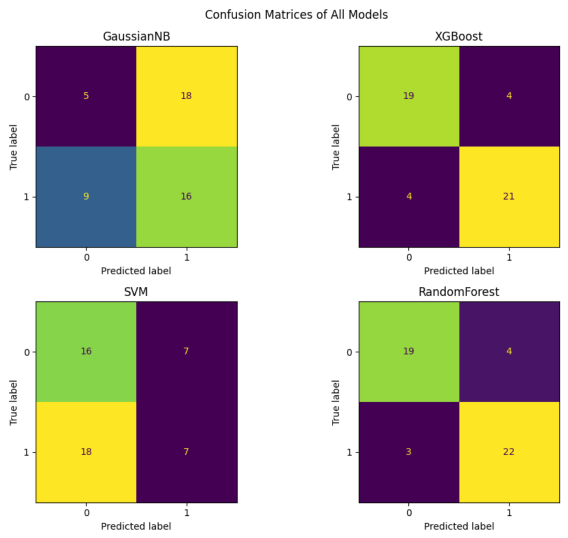
In our machine learning pipeline, we trained four classifiers—Gaussian Naive Bayes, XGBoost, Support Vector Machine, and Random Forest. The ROC curves below show that XGBoost achieves the highest AUROC, indicating the best balance between sensitivity and specificity. Figure 13a plots these ROC curves, revealing that Random Forest follows very closely, while GaussianNB and SVM lag behind. The confusion matrices in Figure 13b confirm this ranking by showing only a handful of misclassifications for XGBoost and Random Forest, compared to many false positives and false negatives for the other two models. Finally, the bar charts in Figure 14 once again show that XGBoost and Random Forest achieve best results.

3.4 Mathematical Modeling

In this simulation, we model a target moving toward the sensor at constant velocity, generate its true distance trajectory by adding small process noise, and then produce biased, noisy measurements by adding a fixed bias and Gaussian noise. We initialize a two-state Kalman filter with the state transition matrix F, measurement matrix H, process noise covariance



(a) ROC curves for GaussianNB, XGBoost, SVM, and Random Forest



(b) Confusion matrices of all models on the test set

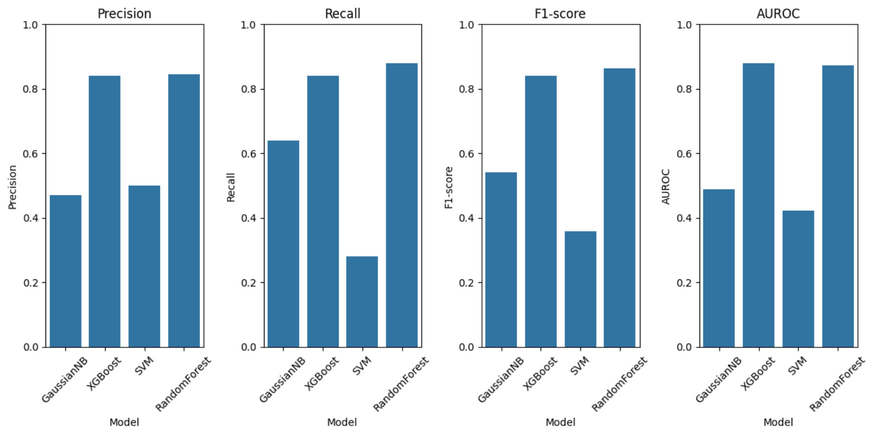


Figure 14: Precision, Recall, F1-score, and AUROC for Each Model.

Q, and measurement noise covariance R, and iterate through prediction and update steps at each time step. After processing all measurements, we compute the root-mean-square error (RMSE) of the distance estimate, which is approximately 0.05 m, indicating that the filter effectively removes noise and bias. As shown in Figure 15, the Kalman-filtered estimate (green line) closely follows the true distance (blue line) while the raw measurements (orange dots) scatter around the true trajectory.

4 Discussion

We used surveys and interviews to focus on reliability, comfort, and ease of use, and suggested some future improvements.

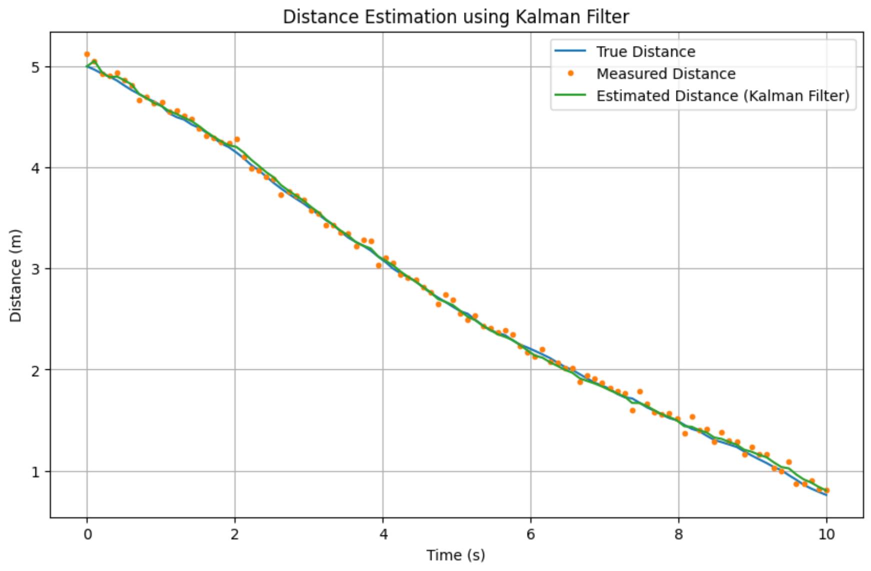


Figure 15: True distance, noisy measurements, and Kalman filter estimates over time.

4.1 Real-World Relevance

To ensure the design of our assistive technology aligned with real user needs, we conducted community outreach with visually impaired individuals throughout the semester. As preliminary research in the fall, we distributed a survey to participants in *When I'm 64*, a WashU course composed of older adults, including individuals with visual impairments. From their responses, we learned that for them, the three most important qualities in an assistive device are reliability, comfort, and ease of use. These insights helped to strongly guide our design-decisions early in the development of the device.

Midway through the semester, we also began consulting with a user from the St. Louis Society for the Blind and Visually Impaired, who provided much more detailed qualitative feedback in an interview format. One of the clearest takeaways was that if a device isn't intuitive, he simply won't use it. Ease of use emerged as the single most critical factor. Additionally, he emphasized that peripheral awareness is where he struggles most—he can usually navigate obstacles in front of him, but blind spots at the edges present significant challenges. Finally, he expressed that the device should be easy to take on and off, without requiring complex adjustments or tight fittings.

These real-world insights strongly validated the core principles guiding our design: simplicity, minimal setup, and clear, non-overwhelming feedback. The device's layout, with an easy-to-wear vest and evenly spaced haptic motors, was tailored to address these needs directly. Ultimately, this user-centered feedback ensured our system was not just functional, but also relevant, accessible, and grounded in the lived experiences of those who would benefit most from it.

4.2 Improvements

Although our prototype demonstrated that the core concept works, several areas for improvement emerged during testing and user interaction.

Sensor Reliability

The current VL53L1X depth sensors occasionally produced noisy or inconsistent readings, particularly in dynamic or low-reflectivity environments. This affected both haptic feedback accuracy and machine learning performance, especially when obstacles moved quickly. In future iterations, we may explore alternative sensor models with improved range and noise tolerance, or enhance filtering techniques to compensate for instability.

Device Durability and Integration

While the breadboard-based prototype allowed for flexible testing, it lacked the durability required for daily use. The wiring was fragile, and the components were not well-secured within the vest. To address this, future versions should incorporate a custom PCB to reduce bulk and soldered connections for greater reliability. Additionally, we plan to explore sewing the sensors directly into the garment to create a fully integrated, comfortable, and unobtrusive wearable device.

User Fit and Accessibility

Because users vary in body shape and size, the current one-size vest may not work equally well for all individuals. More user testing and calibration will be needed to ensure the device is adaptable and inclusive. In the future, we might explore an elastic and velcro connection at the front of the vest as opposed to the current zipper. No matter the changes, ease of donning and removing the vest should be a top priority, as emphasized in user interviews.

Adaptive Feedback Modes

Another promising direction for future development is incorporating multiple feedback modes that adapt the device's sensitivity and sensor emphasis based on the user's context. For example, a "low-sensitivity" mode could be activated in crowded environments to reduce unnecessary alerts and avoid overwhelming the user with constant vibrations. Similarly, a "peripheral-only" mode could be implemented to deactivate the front-facing sensors for users who are primarily concerned about obstacles approaching from the sides. These customizable modes would increase the device's versatility, allowing users to tailor the feedback system to their personal needs and specific environments.

Machine Learning Extensions

While the current machine learning models effectively distinguished between simple shapes, their usefulness in real-world navigation is limited. Expanding the model to recognize specific object types or predict movement patterns could greatly enhance the device's utility. This would likely require additional sensing modalities, which would increase both complexity and power requirements.

5 Conclusion

This project demonstrated that a wearable vest combining depth sensors and haptic feedback can enhance spatial awareness for visually impaired users. The system successfully delivered real-time obstacle detection and unobtrusive but strong directional feedback, and our machine learning models showed strong potential for object shape classification. Importantly, the device aligned well with user feedback emphasizing ease of use, comfort, and intuitiveness—validating our core design choices. However, inconsistencies in sensor readings and the physical fragility of the prototype highlight the need for more robust components. In future iterations, we may explore alternative sensors, a more durable and inclusive vest design, and extensions to the current machine learning capabilities.

6 Deliverables

All deliverables listed below were promised in the proposal and have been completed.

- **Technical Report and Documentation**
  - Final Report: A detailed document covering the technical approach and results.
  - Webpage: A comprehensive project site featuring an overview of the project.
- **Final Device**
  - Vest with 8 embedded VL53L1X TOF depth sensors and 8 haptic motors.
  - All components wired together in the device to prevent any reliance on wireless connection/communication.
  - Sleek and unobtrusive design to prevent worry about outside stigma affecting the user experience.
- **Core Algorithms and Code**
  - Signal Processing Code: Implementation of Kalman filtering, moving average filtering, and moving median filtering.
  - Machine Learning Models: Training and optimization of XGBoost and RF classifiers for object recognition.
- **Experimental Datasets**
  - Collected raw sensor data, filtered data, and labeled training datasets for model development and testing.
- **Python Visualization**

- Visualization taking real-time data from sensors to give a visual representation of what the sensors are detecting to show off the device.

7 Schedule

Our schedule shows the main tasks and timeline from January to May 2025 (see Figure 16). We adhered closely to this timeline by tracking process in weekly meetings and reallocating tasks as needed to avoid delays. We begin with signal processing and the first prototype, then collaborate on hardware development and data acquisition. Next, we update the machine-learning algorithms with new datasets and develop the Python simulation. On ESE Day (April 25), we presented our demo and poster. Finally, we focus on writing the final report and creating the project webpage.

Team Responsibilities:

- **Max Saltrelli**: Led hardware integration and vest assembly, developed python visualization, managed sensor wiring and microcontroller programming for haptic feedback.
- **Jenny Lin**: Developed signal-processing algorithms, took data for ML, trained and evaluated machine-learning models and developed mathematical modeling.

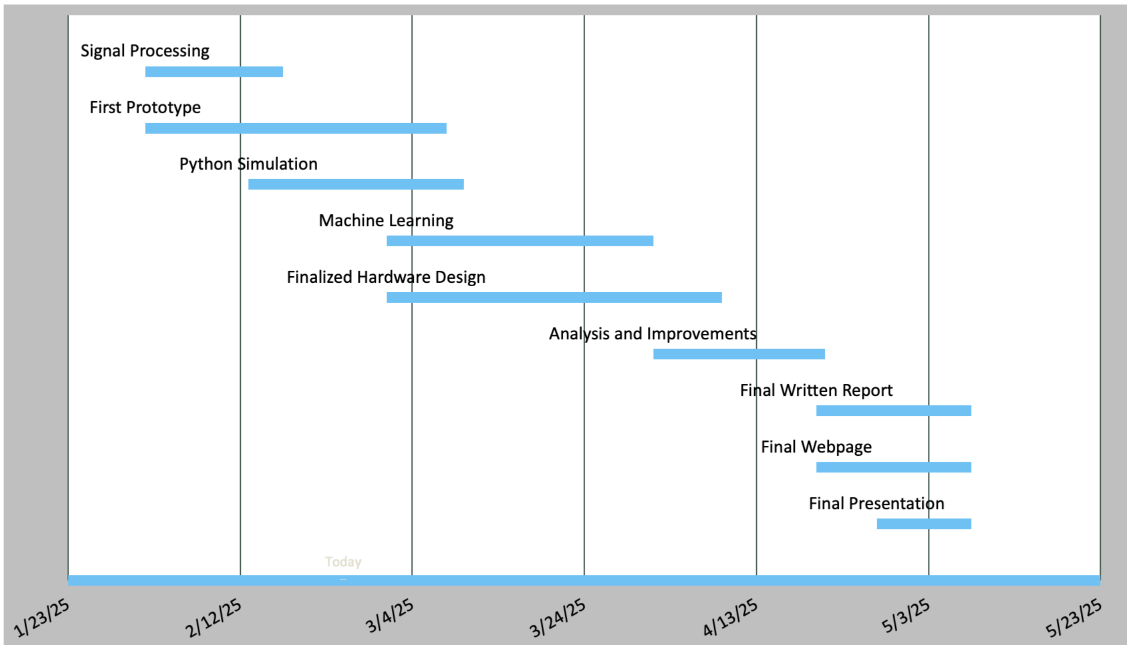


Figure 16: Timeline for completion of the project

References

[1] Pune Blind School, ”Common Problems Faced by Visually Impaired Individuals,” Pune Blind School Blog, Feb. 2025. Available: `https://www.puneblindschool.org/blogs/common-problems-faced-by-visually-impaired-individuals`.

[2] Wikipedia contributors, ”Kalman filter,” Wikipedia, The Free Encyclopedia. Available: `https://en.wikipedia.org/wiki/Kalman_filter`.

[3] Pimoroni, ”VL53L1X Python Library,” GitHub repository. Available: `https://github.com/pimoroni/vl53l1x-python`.

[4] H. Hassan, A. H. Sodhro, S. Pirbhulal, and V. H. C. de Albuquerque, ”Making intelligent in-home health monitoring ubiquitous using machine learning and IoT approaches,” *IEEE Access*, vol. 6, pp. 24667-24681, 2018. Available: `https://ieeexplore.ieee.org/abstract/document/8276628`.

[5] M. Bujnowski, P. Wozniak, and P. Strumillo, ”Obstacle Detection for the Blind Using a Mobile Device and a Tiny Depth Sensor,” in *Computers Helping People with Special Needs*, vol. 9758, Lecture Notes in Computer Science, pp. 199-206, Springer, 2016. Available: `https://link.springer.com/chapter/10.1007/978-3-319-42321-0_33`.

[6] Arduino, ”Portenta H7,” Arduino Documentation. Available: `https://docs.arduino.cc/hardware/portenta-h7/`.

[7] Adafruit, ”TCA9548A I2C Multiplexer,” Product #2717. Available: `https://www.adafruit.com/product/2717`.

[8] SparkFun, ”ULN2803A Darlington Transistor Array Datasheet.” Available: `https://cdn.sparkfun.com/assets/f/0/6/6/5/uln2803a.pdf`.