INSIGHT: In-device Navigation and Scene Interpretation Glasses for Human-centered Travel of Low-vision Users

EECS 473 Fall 2025

Guanyu Xu, Haobo Fang, Jinlin Li, Yizhe Shen, Zhuoyang Chen, Ruopu Dong

I. Introduction and Overview of the Project

Individuals with visual impairments face ongoing challenges in moving independently, avoiding obstacles, and understanding new environments. A survey of 49 blind or visually impaired people found that over 60% had been injured at least once while navigating outdoors [1]. Traditional assistive devices, such as white canes or guide dogs, provide limited awareness of obstacles and do not help in understanding the surrounding scene [2]. Recent research in wearable robotics and multimodal assistive systems shows that we can combine computer vision, haptics, and speech interfaces to enable safer, more intuitive navigation for low-vision users [3-5]. For instance, multimodal learning methods have improved obstacle-avoidance feedback for wearable devices [4], and multimodal sensing systems have helped with scene understanding in assistive robotics [5]. Inspired by these developments and the need for accessible, low-cost assistive technologies, our team created **INSIGHT**. This integrated smart-glass system enables real-time navigation, obstacle detection, and scene interpretation directly on the device.

INSIGHT integrates an **ESP32-S3-based smart glass module** with a **Jetson Orin Nano base station** to provide audio and haptic navigation assistance. The glasses stream microphone and camera data wirelessly to the Jetson, which performs speech recognition, intent classification, YOLO-based obstacle detection, Valhalla navigation, vision-language scene interpretation, and text-to-speech synthesis. All computation runs locally to maintain user privacy and ensure predictable latency [6]. At the COE Design Expo, we demonstrated the complete offline pipeline: users could issue spoken navigation or scene-description requests, receive real-time obstacle avoidance feedback on the vibrational motor, and audio input generated entirely on the device. Because GPS reception is unavailable indoors, the Valhalla routing engine operated using a simulated GPS stream, but all other components behaved as in real outdoor use. This live demonstration showed the system functioning cohesively as an integrated prototype. It highlighted the feasibility of an affordable multimodal assistive smart-glass platform for visually impaired individuals (as shown in Figure 1).

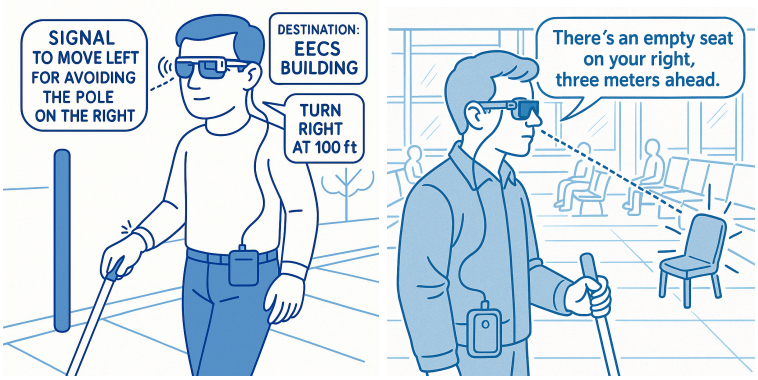


Figure 1 (AI-generated): Scene illustration of Obstacle-aware navigation (left) and Scene description (right)

We successfully achieved nearly all goals and milestones outlined in our original proposal. On the wearable side, every ESP32-based function—wake-word detection, audio capture and streaming, JPEG-compressed video transmission, haptic feedback, and audio playback—was implemented and shown to work both individually and as an integrated subsystem. On the Jetson Orin Nano base station, all planned software components were completed as well, including offline speech-to-text, intent classification, YOLO-based obstacle detection, Valhalla navigation, vision-language scene interpretation, text-to-speech synthesis, and IMU-based heading estimation. As a result, Milestone 1, Milestone 2, and the final Design Expo demonstration were completed on schedule with the intended functionality.

The remaining gaps relate primarily to robustness and user-experience challenges rather than missing features. The ESP32-S3's limited PSRAM and internal bandwidth leave very little headroom when running all tasks concurrently, reducing reliability under heavy load. The Valhalla routing engine, while functional, is not optimized for visually impaired users and therefore provides instructions that are not always meaningful in real navigation contexts. On the Jetson, the 8 GB memory limits the efficiency of audio and vision models running together, causing roughly a 2~5 second delay in speech feedback. Despite these constraints, the system demonstrated a complete offline assistive pipeline and validated the feasibility of our proposed smart-glass concept.

II. Project Description

a. Project Goal, System Concept, and Feasibility

The goal of the INSIGHT system is to develop a wearable assistive device that enables visually impaired users to navigate safely and interpret their surroundings in real time. It combines onboard sensing, wireless communication, and local AI-based scene interpretation and navigation into a single, low-cost system [4,5,8]. The smart-glass unit integrates an ESP32-S3 MCU, a microphone array, a camera, a vibrotactile motor, and a speaker on a custom PCB. It streams audio and video to a Jetson Orin Nano base station, which performs speech-to-text transcription, intent classification, obstacle detection using YOLO-v11, scene interpretation using a vision-language model (Moondream-2b), navigation routing using Valhalla, and text-to-speech synthesis. The system concept centers on real-time perception–action loops: the user speaks a command, the base station interprets it, analyzes the live video feed, and sends audio and haptic feedback back to the smart glasses.

The feasibility of this project was demonstrated through our hardware and software design choices and step-by-step testing. The ESP32-S3 platform provided sufficient peripheral support (I2S, DVP, Wi-Fi) to stream audio and video while remaining light and low-power for a wearable [9]. The Jetson Orin Nano offered sufficient GPU performance to run our detection and scene description models with acceptable latency when we optimized them and ran them sequentially [10-12]. In our proposal we had already flagged several limits: memory for the VLM, bandwidth for camera streaming, and power draw during continuous processing, and the final system showed that these could be handled in practice by using TCP

Wi-Fi streaming, sending compressed JPEG frames, avoiding running too many heavy tasks at once, and pairing the system with a compact battery pack with predictable discharge [13,14]. The live demo at the Design Expo, with obstacle detection and interactive scene description, shows that the system is technically feasible, robust, and reproducible using off-the-shelf parts within the cost and time constraints of EECS 473.

b. Design Constraints

Our project was shaped by several design constraints related to privacy, responsiveness, hardware resources, and time and budget constraints.

Privacy:

Because our system targets visually impaired users in public spaces, transmitting audio or video to cloud services would raise ethical and legal concerns. To address this, we required all perception, navigation, and speech-processing functions to run entirely offline. This constraint drove our architectural decision to deploy the YOLO-v11 obstacle-detection model, the Valhalla routing engine, and the Moondream-2B vision-language model directly on the Jetson Orin Nano.

Real-time multimodal feedback:

Safe mobility assistance requires that haptic cues, obstacle warnings, and spoken instructions be delivered with minimal latency, typically within a few hundred milliseconds for obstacle alerts and within 1~2 seconds for scene descriptions. Any cloud-based pipeline would introduce unpredictable network delays, making the system unsafe or unusable for real-world navigation. By ensuring that all computation remained on-device and that no raw sensor data ever left the ESP32–Jetson softAP network, we satisfied both the privacy requirement and the strict latency requirements needed for timely audio and haptic feedback.

Hardware Limitations:

Hardware resources imposed additional constraints. The ESP32-S3 module has limited PSRAM bandwidth and processing capability, which constrained us to JPEG-compressed video frames and to sequential audio/video handling. The Jetson Orin Nano, while capable of real-time inference, has only 8 GB of memory, which requires careful GPU memory profiling. Namely, a quantized vision-language model is needed to fit into Jetson's memory.

Time and Budget Constraints:

Given that EECS 473 projects follow a ten-week timeline, time was one of our biggest constraints. The system needed embedded development, PCB design and fabrication, model deployment on the Jetson, and outdoor testing. However, most of the iterative testing could only start after most of the implementation was done. This meant that integration, debugging, and user-focused adjustments were squeezed into the last part of the semester, leaving us with little time to improve reliability once all subsystems were connected. Our budget limit of about $1500

added another constraint; we couldn't afford many hardware iterations, so we had to justify each PCB revision and major component selection. The failure of the first PCB fabrication and the need for extra boards and mechanical parts took up a large part of the budget, leaving us with less room for further changes. These time and budget limits pushed us toward designs that were not only technically interesting but also practical to implement, test, and support within one semester.

c. System Architecture

The system architecture is defined by a distributed, human-centric design that splits functionality between a lightweight wearable interface and a powerful local compute unit. The system operates entirely offline to ensure user privacy and minimize latency.

Wearable Smart Glass Module:

We designed a customized PCB for the wearable module that includes the following components: an ESP32-S3 WROOM Module, an OV2640 camera, a MEMS microphone array, a speaker, and vibrational motors. The PCB design is partially adapted from the ESP32-S3-Korvo-2 audio development board [10]. On the software side, we use FreeRTOS in C to schedule five concurrent tasks: Task 1 performs wake-word detection to provide a user-friendly interface for low-vision users; Task 2 records audio after wakeword is detected and streams it to the Jetson; Task 3 acquires QVGA video frames at roughly 30 fps and transmits them over the network; Task 4 receives synthesized audio from the Jetson and plays it through the speaker; and Task 5 receives haptic commands and drives the vibromotor accordingly. To accommodate hardware limitations, it compresses video into JPEG frames and streams data over a TCP Wi-Fi connection. The function block diagram is shown in the Figure. 2.

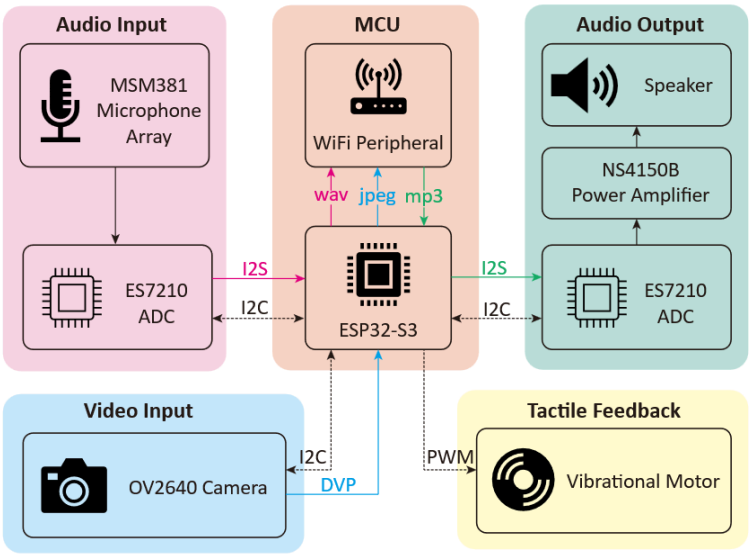


Figure 2: Function block diagram of the ESP32-based smart glass system.

Base Station (Compute Unit):

Heavy computational tasks are offloaded to a Jetson Orin Nano. This station establishes communication with the smart glass system using four TCP ports: Port 1 receives audio commands sent from ESP32 in WAV format; Port 2 receives video frames in JPEG format for later obstacle detection; Port 3 transmits audio feedback from navigation and scene description in MP3 format; Port 4 transmits vibrational motor feedback in the form of a specially formatted string. The software mainly runs a pipeline of locally deployed AI models, including YOLO-v11 for obstacle detection, Moondream-2b for scene interpretation, and Valhalla for navigation routing. It also handles speech-to-text (STT) and text-to-speech (TTS) conversion to interpret user commands and generate spoken instructions. The detailed block diagram of the base station is shown in Figure. 3.

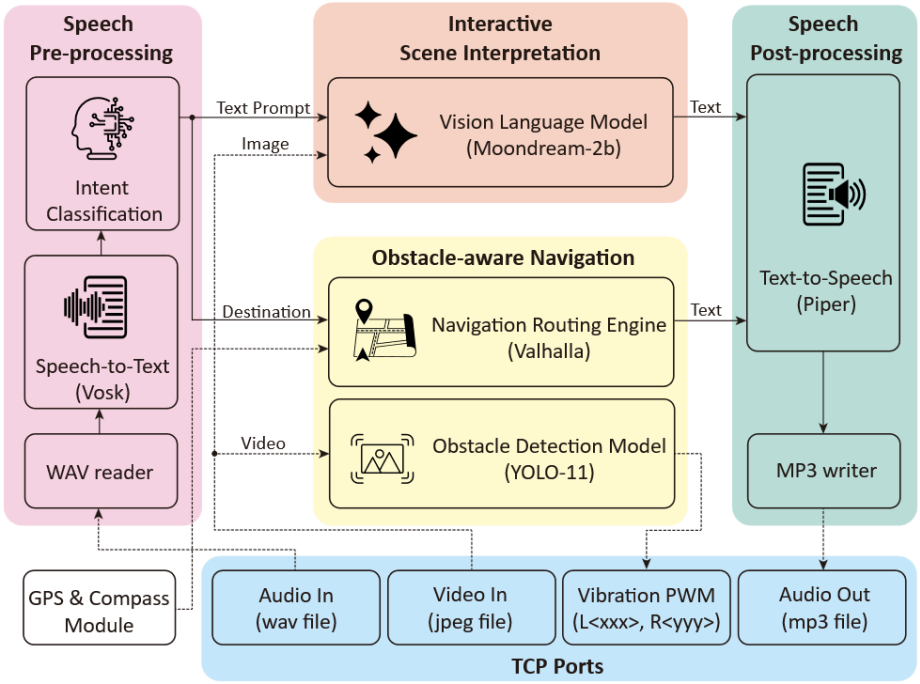


Figure 3: Function block diagram of the Jetson Orin Nano base station.

The system establishes a real-time perception–action loop in which the wearable streams sensor data to the base station, which analyzes the scene and returns audio and haptic feedback to the user. This local processing ensures that no raw sensor data leaves the secure softAP network. In our current deployment, the combined video and audio streams require about **300 KB/s** of sustained throughput on the softAP link.

Battery Life:

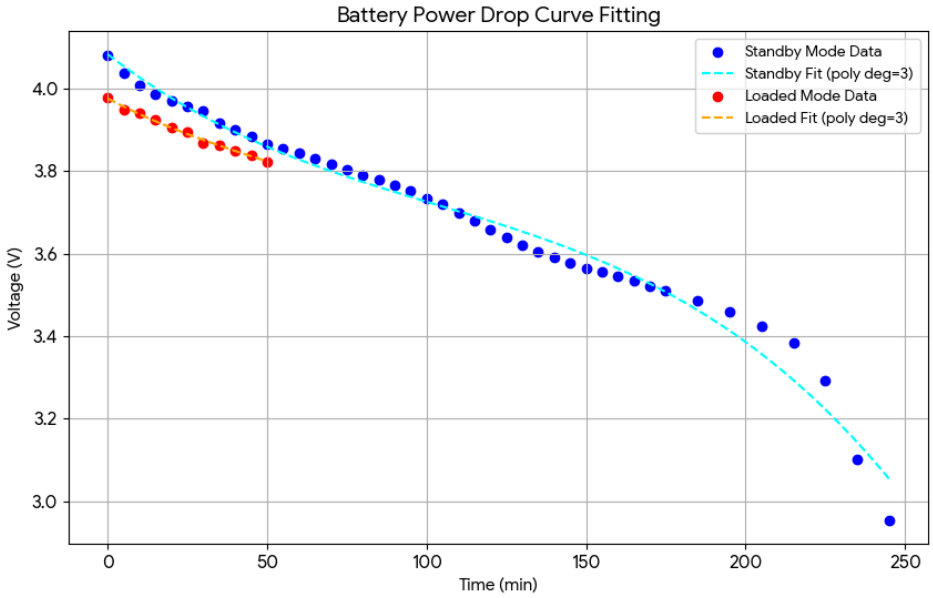


Figure 4. Power drop curve.

The Smart Glass's power characterization shows a robust energy profile that excels in both idle and active states. In standby mode, the system draws minimal quiescent current, ensuring exceptional longevity and charge preservation during periods of inactivity. Crucially, in loaded mode, the battery voltage remains highly stable despite the increased power demand of the peripherals; the discharge curve shows a controlled decline that stays comfortably above the critical 3.3V operating threshold for a substantial duration. This confirms that the power management architecture successfully balances the efficiency required for extended standby times with the current capability needed to sustain stable voltage levels during prolonged active usage.

III. Milestones, Schedule, and Budget

Our project followed a structured four-phase development schedule: (1) sensor and communication setup, (2) core software pipeline development and integration, (3) PCB Design, and (4) full-system testing and adjustment. Overall, we adhered closely to our original schedule, with most major milestones completed on time. PCB revisions required additional iteration, which delayed final integration by approximately two to three weeks, but did not affect the completion of core deliverables before the Design Expo.

For Milestone 1, we were required to demonstrate the core sensing and communication pipeline: wake-word-based audio capture on the ESP32-S3, reliable Wi-Fi streaming of audio and camera data to the Jetson, an initial speech-to-text plus intent-classification pipeline, and a prototype obstacle-detection

model running on recorded frames. All of these were completed on time, and we were able to show end-to-end data flow from the glasses to the Jetson and back. For Milestone 2, the requirement increased to near-complete functionality: real-time obstacle detection on live video, GPS-based navigation with Valhalla, preliminary haptic feedback on the vibromotor, and a first integrated scene-description path using the vision-language model. We met these goals with only a slight delay due to replacing the unreliable VK-172 GPS module and tuning the VLM on the Jetson Orin Nano. Still, the navigation, obstacle detection, and haptic loop were all working. By the time of the Design Expo, our requirement was a fully integrated prototype suitable for a live demo: the user could issue spoken commands, receive route guidance and obstacle alerts through audio and vibration, and request scene descriptions, all running offline on our ESP32–Jetson platform. We met this final goal and successfully demonstrated the complete system at Expo under realistic outdoor conditions.

Our budget performance evolved significantly over the course of the project. In our original proposal, we estimated a total cost of $641.72, assuming a single PCB fabrication run, one ESP32-S3 development board, a minimal sensor set, and no major hardware replacements. The primary reasons for this increase were several necessary hardware upgrades and reorders. First, we added a 512 GB NVMe SSD for Jetson Orin Nano, which was not included in the original estimate but was required to support fully on-device model inference and navigation. Second, the first PCB had design issues and failed, forcing us to place a second JLCPCB fabrication and assembly order, which substantially increased PCB-related costs. Additional growth came from purchasing extra ESP32-S3 boards for debugging, mechanical mounting hardware, vibration motors, connectors, and small accessories needed to make the system robust enough for real-world use.

IV. Lessons learned

Throughout the development of INSIGHT, several aspects of the project went well. Our team was able to design and verify each subsystem primarily on schedule. Early modular testing paid off: speech recognition, intent classification, obstacle detection, and scene-description pipelines worked reliably once integrated. The fully offline architecture also proved to be a strong design decision, providing predictable latency and avoiding reliance on unstable networks.

Some parts of the project, however, were more challenging than anticipated. Field testing was significantly constrained by winter weather, which limited outdoor navigation trials and forced us to simulate certain conditions, such as GPS signals, during the Expo demonstration. Hardware constraints were also more restrictive than expected: the ESP32-S3 operated near the limits of its PSRAM and DRAM, making the WIFI AP less robust under load, and the Jetson Orin Nano's memory ceiling created unavoidable latency in the audio feedback loop. Additionally, Valhalla's navigation instructions were not designed for visually impaired users, reducing the practical usefulness of that subsystem.

If we could send a memo back in time, we would recommend setting more achievable intermediate goals, allowing more buffer time for debugging, and starting robustness testing much earlier, especially

outdoors. We also would have planned a more straightforward fallback navigation strategy in case Valhalla proved unsuitable.
Individually, we gained substantial technical experience: embedded real-time programming with FreeRTOS, ESP32 memory and peripheral management, GPU-accelerated computer vision and VLM inference on the Jetson platform, IMU integration, TCP streaming optimization, and the full workflow of PCB design, fabrication, and hardware bring-up. As a team, we learned the importance of iterative integration, realistic scheduling, and designing with human-factor constraints in mind—especially for assistive technologies.

V. Contributions of each Member of the team

Jinlin Li (16.67% effort)
I focused mainly on the embedded and system-integration side. On the glasses, I worked on the ESP32-S3 hardware and firmware, implementing wake-word detection, I2S audio capture and playback, and an audio_manager RTOS semaphore module to coordinate the microphone and speaker on a shared I2S interface, and I debugged several ESP-ADF memory and task-creation issues to make the audio pipeline stable. I also contributed to the PCB design. On the base station, I set up the Jetson Orin Nano and built a Flask web interface to monitor live logs, video frames, haptic feedback, and real-time GPS mapping. I extended our navigation stack using OpenStreetMap and Valhalla by modifying the umich_nav module, logging waypoints to CSV, and designing the turn-by-turn guidance behavior based on GPS/IMU input. In addition, I contributed to the written documentation, including figure captions, conclusions, and proper citations and acknowledgments for external code, ideas, and hardware.

Haobo Fang (16.67% effort)
In our EECS 473 project, I served as the main embedded systems lead. On the ESP32 side, I implemented Tasks 1–4, including the core firmware for Wi-Fi/SoftAP setup, real-time video and audio streaming, reverse-audio playback, and vibration feedback, and organized them into a robust FreeRTOS multi-task architecture. I designed the task structure, priorities, and stack sizes, and debugged issues such as stack overflows, PSRAM usage, and socket reconnection to keep the system stable under continuous load. After the initial bring-up, I maintained most of the embedded codebase, refactoring communication protocols between the ESP and Jetson, fixing parsing bugs after IP changes, and improving error handling so the system could recover without full reboot. On the hardware side, I co-designed and laid out the PCB with Guanyu that integrate the ESP32, audio front-end, haptic drivers, and power management, and led multiple rounds of prototyping (hand-wired boards, soldering, and hardware debugging) to bring up and validate the full system. In addition, I participated in and contributed to all of the paperwork.

Guanyu Xu(16.67% effort)
Guanyu's contributions focused on the Jetson Nano software stack and the Smart Glass PCB design. On the Jetson side, he designed and implemented several key components of the main pipeline, including the

audio reception and transcription module that streams speech from the glasses to the Jetson, and intent-classification module that maps transcribed text to navigation or scene description commands, and the integration of the Moondream vision language model to provide scene descriptions on demand. He also helped organize how these components interact with the rest of the system so that perception, navigation, and user queries run smoothly together. On the hardware side, Guanyu contributed heavily to the Smart Glass PCB design, focusing on component placement, power distribution, and sensor and vibrational motor interfaces. Throughout the project, he participated in system integration and helped test and debug the overall system, as well as document these components in the final report.

Yizhe Shen(16.67% effort)

I focused on the interaction pipeline between the ESP32-S3 and the Jetson. On the embedded side, I implemented wake-word detection on the ESP32-S3 using the ESP-Skainet model and helped build the FreeRTOS-based pipeline that coordinates Skainet inference with ESP-ADF I2S audio capture and playback and streaming over a TCP connection. I also contributed to the PCB design. On the software side, I integrated a balanced, fast-and-accurate text-to-speech model - espeak, so that responses generated on the Jetson are sent back over TCP and played through the ESP's audio stack. I also helped debug the end-to-end pipeline after multiple on-field tests, fixing timing, buffering, and communication issues to make the system stable in real use. In addition, I participated in and contributed to all of the paperwork.

Zhuoyang Chen(16.67% effort)

My primary contribution focused on the architectural development of the Nvidia Jetson Nano base station, where I engineered the central processing pipeline that powers Smart Glass intelligence. This involved integrating high-performance AI modules, specifically the Moondream-2b Vision-Language Model (VLM) for detailed scene description and YOLO-v11 for real-time obstacle detection, alongside a Text-to-Speech (TTS) interface. Furthermore, I spearheaded the implementation of the navigation module, leveraging OpenStreetMap data and the Valhalla routing engine to enable accurate wayfinding. Beyond core development, I actively participated in system integration, rigorous testing, and debugging, and contributed to technical documentation, including the initial proposal and the final project report.

Ruopu Dong(16.67% effort)

I focused on the GPS navigation of the system and its field testing. I built the interface between the GPS module connected to the base station by USB serial port and the Valhalla navigation engine. I also utilized an adaptive strategy to control the navigation threshold for rerouting. In testing the system's navigation in the open field, I replicated a blind person's situation. I travelled multiple times with the smart glass on my eyes and a blind stick to test the robustness of the system. Furthermore, I contributed to the PCB design, audio transmission and prototyping the debug version of the smart glass.

VI. Cost of Manufacture

To evaluate the manufacturing cost of our PCB design, we obtained a quote from JLCPCB. The service provided a total cost of $6863.77 for a production run of 1200 boards. This includes fabrication and partial assembly, with some components requiring separate sourcing and cost estimation.

Total cost (1200 pcs): $6863.77

Cost per board: $6863.77 ÷ 1200 ≈ $5.72

A batch size of 1200 units was selected as a reasonable production quantity for our application. At approximately $5.72 per board, the cost is highly reasonable for a design that includes both manufacturing and assembly. This price point is competitive compared to typical PCB production costs and demonstrates that the design is practical to produce at scale. The project can be implemented without an excessive financial burden while still meeting the technical requirements.

VII. Parts and Budget

Category	Item	Qty	Unit Price (USD)	Tax	Shipping	Total Cost
Compute Modules	ESP32-S3-Korvo-2 Development Board	2	$50.00	$6.00	$0.00	**$106.00**
Vision Module	ESP32-CAM with OV2640 Camera	1	$15.99	$0.96	$0.00	**$16.95**
PCB Fabrication v1	4-Layer Bare Rigid PCB + Rigid-Flex (JLCPCB)	1	$304.14	$186.61	$39.00	**$528.74**
GPS Modules	VK-172 USB GPS Receiver	1	$9.99	$0.00	$0.00	**$9.99**
GPS Modules	VFAN USB GPS Receiver	1	$19.99	$0.00	$0.00	**$19.99**
Power	Talentcell 64.75 Wh Li-ion Battery Pack	1	$53.99	$4.30	$0.00	**$58.43**
PCB Fabrication v2	4-Layer Bare Rigid PCB + Rigid-Flex (JLCPCB)	1	$325.34	$166.24	$38.14	**$529.42**

Compute (Extra)	ESP32-S3-Korvo-2 (additional debugging unit)	1	$50.00	$3.00	$0.00	**$53.00**
Mechanical / Demo	Folding Blind Cane Walking Stick	1	$9.97	$0.60	$0.00	**$10.57**
Mechanical / Demo	Mr. Pen - Elastic Band	1	$4.85	$0.29	$0.00	**$5.14**
Mechanical / Demo	M3 Motherboard Standoffs&Screws&Nuts Kit	1	$14.15	$0.85	$0.00	**$15.00**
Mechanical / Demo	802025 3.7V Lipo Battery	1	$30.16	$1.82	$0.00	**$31.98**
Compute Modules	3PCS ESP32-S3-DevKitC-1	1	$17.99	$1.08	$0.00	**$19.07**
Mechanical / Demo	ESP32-CAM Wireless WiFi+Bluetooth Development Board	1	$15.99	$0.96	$0.00	**$16.95**
Mechanical / Demo	20PCS 10mmx3mm Mini Vibration Motors	1	$22.68	$1.36	$0.00	**$24.04**
Mechanical / Demo	20 Sets SH1.0mm 2 Pin Connector Plug Male	1	$8.49	$0.51	$0.00	**$9.00**
Compute Modules	NVIDIA Jetson Orin Nano Super Developer Kit	1	$249	$14.94	$0.00	**$263.94**
Compute Modules	Silicon Power 512GB NVMe M.2 PCIe Gen3x4 2280 SSD	1	$37.97	$2.28	$0.00	**$40.25**
Total						**$1759.27**

VIII. References and Citations

Citations

[1] F. E. Z. El-Taher et al., "A Survey on Outdoor Navigation Applications for People with Visual Impairments," IEEE Access, vol. 11, pp. 14647–14666, 2023, doi: 10.1109/ACCESS.2023.3244073.

[2] K. M. P. Hoogsteen et al., "Beyond the Cane: Describing Urban Scenes to Blind People for Mobility Tasks," ACM Transactions on Accessible Computing, 2022, doi: 10.1145/3522757.

[3] G. I. Okolo, T. Althobaiti, and N. Ramzan, "Assistive Systems for Visually Impaired Persons: Challenges and Opportunities for Navigation Assistance," Sensors, vol. 24, no. 11, article 3572, 2024, doi: 10.3390/s24113572.

[4] Y. Gao et al., "A Wearable Obstacle Avoidance Device for Visually Impaired Individuals with Cross-Modal Learning," Nature Communications, vol. 16, article 2857, 2025, doi: 10.1038/s41467-025-58085-x.

[5] P. Slade et al., "Multimodal Sensing and Intuitive Steering Assistance Improve Navigation and Mobility for People with Impaired Vision," Science Robotics, vol. 6, no. 59, eabg6594, 2021, doi: 10.1126/scirobotics.abg6594.

[6] F. Cheraghpour Samavati and A. Rahimi Ghasem Abadi, "Assistive Technologies for the Visually Impaired: A Review," Cureus Journal of Computer Science, vol. 2, article es44389-025-06891-1, Oct. 13, 2025, doi: 10.7759/s44389-025-06891-1.

[7] A. K. Sidhi, S. Pabboju, and P. Kiranmaie, "Captain Hope: An Offline AI-Powered Wearable Mobility Assistant for the Visually Impaired," Authorea, preprint, Sept. 25, 2025, doi: 10.22541/au.175882486.65199983/v1.

[8] Z. Chen et al., "A Wearable Navigation Device for the Visually Impaired by Integrating Semantic Visual SLAM and a Mobile Computing Platform," Sensors, vol. 21, no. 4, article 1536, 2021, doi: 10.3390/s21041536.

[9] Espressif Systems, ESP32-S3 Series Datasheet: 2.4 GHz Wi-Fi + Bluetooth LE SoC, ver. 1.6, 2023.

[10] Espressif Systems, ESP32-S3-Korvo-2 V3.1 Multimedia Development Board User Guide, 2024.

[11] NVIDIA Corporation, NVIDIA Jetson Orin Nano Developer Kit Datasheet, 2023.

[12] K. Wali, "Hardware-Software Co-Design for Power-Efficient Edge AI Systems," Science Data & Learning Machine Intelligence: Artificial Intelligence Journal, vol. 2, no. 4, pp. 2754–2760, 2024, doi: 10.51219/JAIMLD/karthik-wali/580.

[13] Ultralytics, "Quick Start Guide: NVIDIA Jetson with Ultralytics YOLO11," Ultralytics YOLO Docs, 2024. [Online]. Available: https://docs.ultralytics.com/guides/nvidia-jetson/

[14] M. Shafique et al., "Towards Energy-Efficient and Secure Edge AI: A Cross-Layer Framework," in Proc. 40th IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD), 2021, pp. 1–9, doi: 10.1109/ICCAD51958.2021.9643539.

[15] R. Mustafa et al., "A Cross-Layer Secure and Energy-Efficient Framework for the Internet of Things: A Comprehensive Survey," Sensors, vol. 24, no. 19, article 7209, 2024, doi: 10.3390/s24197209.

Interface Code

- **ESP32-S3 Interface Code:** The interface code for the ESP32-S3 platform can be found in this GitHub repository: https://github.com/realfanghb/Smart_glass_esp32. This section contains the relevant function declarations and macros for configuring and accessing hardware resources on the ESP32-S3.
- **Jetson Pipeline Code:** The Jetson platform code, including pipeline modules for sensor input, data processing, and hardware control, is available in this GitHub repository: https://github.com/GuanyuXu04/Smart_Glass. This section provides the APIs and typedefs necessary to integrate Jetson hardware into a processing pipeline. It covers how sensor data flows through the system, how intermediate results are handled, and how outputs are passed to actuators or higher-level modules.
- **Jetson Navigation Code:** In addition to the hardware interfaces, the Jetson platform also includes navigation-related code. This code handles path planning, sensor integration, and motion control logic, and can be found in this GitHub repository: https://github.com/janChen0310/OSM-Valhalla-Routing-Demo. These functions form the basis of the navigation stack and demonstrate how the Jetson interacts with sensors and actuators to achieve autonomous movement.

For more information on using these interfaces, please refer to the README.md file in each repository. The README provides additional context, usage examples, and integration notes.

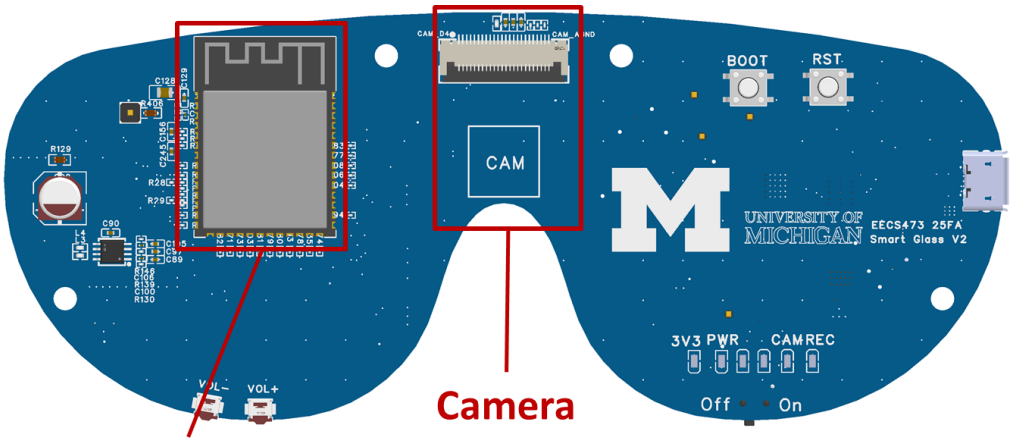
Use of External Code, Ideas, and Hardware

Our project builds on several prior efforts and open-source tools. We reused and adapted ideas and code from our earlier smart-glasses and offline navigation prototypes, including the overall ESP32–Jetson architecture, the use of an ESP32-S3-based audio/camera pipeline, and a Valhalla/OSM-based routing stack. We also rely on external frameworks and models that we did not write, such as Espressif's ESP-IDF/ESP-ADF for embedded development, Vosk for speech recognition, YOLO for object detection, Moondream and other vision-language and NLP tools, OpenCV for image processing, and Valhalla with OpenStreetMap data for navigation. Off-the-shelf hardware platforms such as the ESP32-S3 development board, the Jetson Orin Nano kit, standard IMUs, cameras, microphones, vibration motors, and batteries are used according to their vendor datasheets and reference designs.

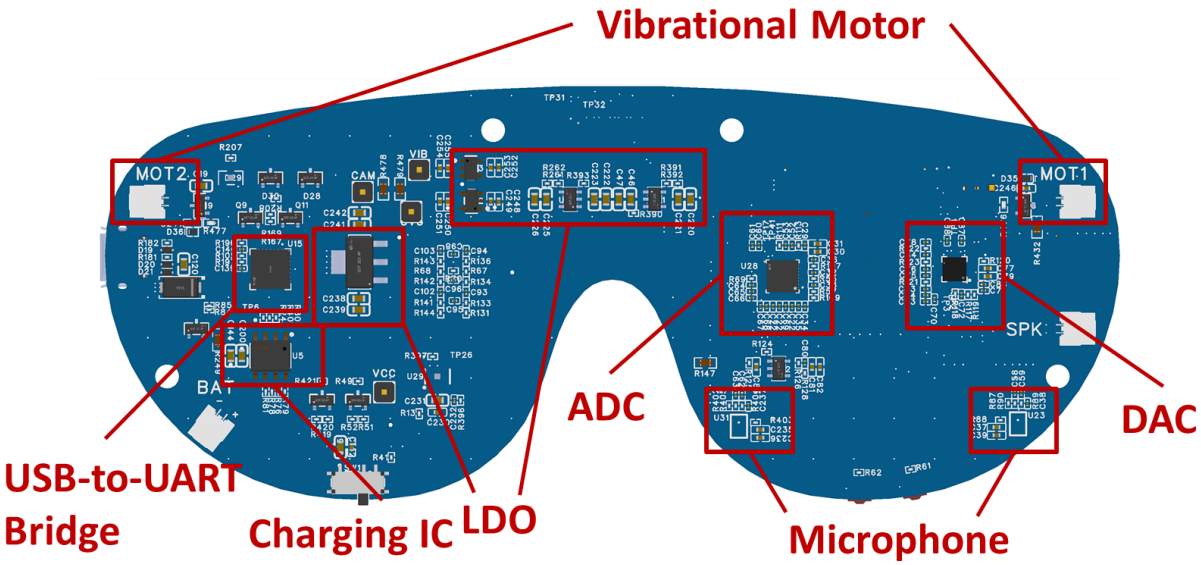
Acknowledgments

We are grateful to our instructors Prof. Mark Prehob, James Carl and Matt Smith and GSIs (Anna, Frederick, Alec) for their guidance on hardware selection, system integration, and debugging, and to friends and classmates who tested early versions of the glasses and gave feedback on usability and comfort. We also thank roommates and family members for their practical help and support during long build/debug sessions, even though they are not officially part of the project team.

Soldered PCB



(a). Top View



(b). Bottom View