ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

OpenAI Glasses for Navigation

Team #3

JIASHEN REN (jiashen3)
HAOYU ZHU (haoyu13)
JINNAN ZHANG (jinnanz2)
DARUI XU (daruixu2)

Advisor: Bo Zhao

TA: Hengjia Ran

May 15, 2026

Abstract

OpenAI Glasses for Navigation is a wearable assistive-navigation prototype for blind or visually impaired users. The final system combines a Seeed XIAO ESP32S3 Sense sensing node, authenticated wireless video and audio transport, a Python application server, real-time vision models, a navigation state machine, speech recognition, text-to-speech output, and a browser debug console. At the demonstration stage, the implemented non-mechanical system supports blind-path following, obstacle warnings, crosswalk approach alerts, traffic-light checking, crossing-mode guidance, device audio playback, runtime tuning, and synchronized recording for debugging. The final software architecture intentionally moved heavy inference off the wearable device and onto a host computer so that the glasses remain lightweight while the host runs Torch/Ultralytics segmentation and detection models. Verification focused on software and integration behavior: 170 automated tests passed for configuration validation, packet integrity, RTP/JPEG video reassembly, vision postprocessing, state-machine transitions, web endpoints, ASR plumbing, and speech output. The prototype is therefore a functional engineering platform for navigation-assistance experiments, but it is not a certified mobility aid; real deployment would require broader field testing, human-subject evaluation, privacy review, and mechanical validation.

Contents

1 Introduction

1.1 Problem and Motivation

Blind or visually impaired pedestrians often need several kinds of information at the same time: where the tactile paving is, whether a nearby obstacle blocks the path, whether a crosswalk is approaching, and whether a traffic signal allows crossing. A phone-based assistant can provide some of this information, but it usually requires a hand, a screen, or a stable camera framing. A head-mounted device is more natural for walking because the camera points roughly where the user faces and the feedback can be delivered through speech.

The project goal was to build a hands-free prototype that combines a wearable mechanical carrier, front-end sensing electronics, real-time perception, and spoken guidance. The important engineering problem was not only model accuracy. The system also had to mount and position the camera, PCB, battery, and related electronic components in a stable and comfortable wearable form, while deciding when a detection should become a spoken instruction, when a crosswalk cue should interrupt blind-path navigation, and when unstable detections should be ignored. For that reason, the final design emphasizes mechanical stability, wearable comfort, state management, debouncing, transport reliability, and clear operating limits.

1.2 Final System Overview

Figure 1 shows the final system architecture. The mechanical wearable carrier provides the physical platform for the sensing and electronic components, while the wearable sensing node represents the front-end sensing and communication subsystem, including the camera, microphone, IMU, audio output, and embedded controller. The figure separates the physical carrier from the signal-processing path so that the mechanical support role and the data-flow architecture are both visible.

The wearable node captures JPEG video, PCM16 audio, and inertial measurements. Video is sent to the host by RTP/JPEG over UDP with a truncated HMAC-SHA256 authentication tag, while control, audio uplink, and optional audio downlink use WebSocket channels. The host server receives and validates those streams, runs vision inference, updates a navigation controller, broadcasts a browser monitoring view, and emits speech either to the web console or back to the device speaker.

The final system is organized into five main blocks. The mechanical carrier provides the wearable platform for mounting and aligning the front-end components. The wearable sensing block is responsible for collection and packetization only. The transport and server block validates packets, reconstructs video, and keeps the device, browser, and navigation loop synchronized. The perception block uses three model paths: blind-path/crosswalk segmentation, fixed-class YOLOE obstacle detection, and traffic-light detection. The navigation and speech block converts model outputs into rate-limited, mode-aware guidance.

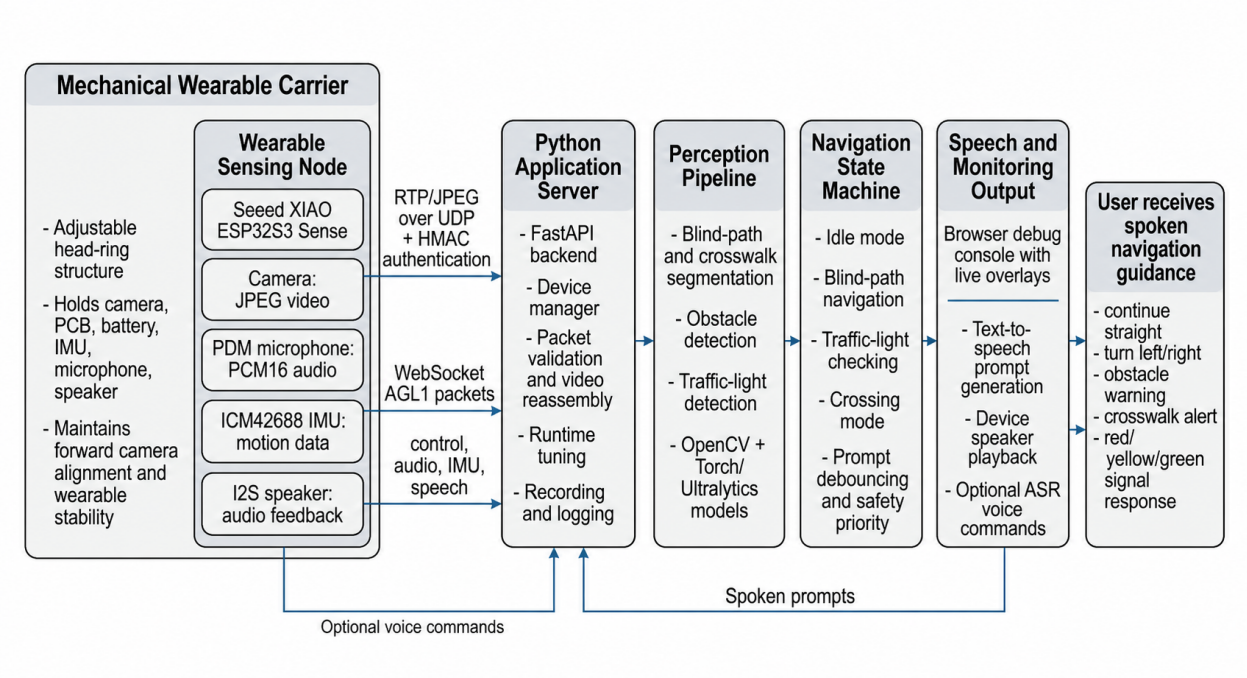

Figure 1: Final system architecture, including the mechanical wearable carrier, sensing node, authenticated transport, host-side perception and navigation pipeline, speech output, and user feedback loop.

1.3 High-Level Requirements

Table 1 lists the final high-level requirements used to judge the complete prototype, including the wearable mechanical carrier, sensing node, transport layer, perception software, navigation logic, and feedback interface.

1.4 Changes from Design Review

The design review document described a larger multimodal assistant that included open-vocabulary item search and hand-guided near-field object retrieval. The final codebase does not implement that item-search mode. To finish a reliable demonstration, the scope was reduced to the navigation-critical path: blind-path following, crosswalk awareness, traffic-light checking, obstacle filtering, speech commands, device audio output, web visualization, and recording. This was the right tradeoff because navigation prompts and road-crossing behavior have a higher safety impact than a partially finished item-search feature.

The transport design also changed substantially. The design review assumed simple streaming channels, while the final implementation uses authenticated RTP/JPEG over UDP for video and a project-specific AGL1 packet format over WebSockets for control, audio, IMU, and speech. This reduced latency and avoided large WebSocket video frames

Table 1: Final high-level requirements for the complete prototype

Requirement	Final specification
Wearable mechanical carrier	The prototype shall provide a wearable platform for mounting the camera, PCB, battery, and related sensing components, while keeping the front-facing camera aligned with the user's walking direction and maintaining physical stability during operation.
Hands-free sensing and feedback	The system shall accept live wearable video and audio, expose a no-hand browser/debug workflow for development, and produce spoken navigation prompts without requiring the user to view a screen.
Blind-path guidance	In blind-path mode, the system shall detect the tactile path, estimate lateral offset and heading, warn about obstacles in the walking corridor, and avoid repeating identical non-urgent prompts faster than the speech cooldown.
Crosswalk and signal support	The system shall warn when a crosswalk appears near the blind path, support an explicit crossing mode, prioritize red and yellow signals over green, and require repeated green evidence before starting crossing guidance.
Robust transport and configuration	The system shall reject malformed packets, bound payload sizes, authenticate UDP video, validate configuration values, and allow runtime tuning of important camera and vision thresholds.
Verification without unsafe field dependence	Core software behavior shall be testable without requiring blind users, live traffic, cloud availability, or attached hardware.

while still rejecting unauthenticated or malformed UDP packets.

Finally, the mechanical form factor changed from the original glasses-style carrier to a more stable wearable carrier. The initial glasses-style design placed the camera and electronic components near the front of the frame, which created a front-heavy load distribution and reduced wearing stability. The improved design uses an adjustable head-ring structure as the mechanical carrier for the sensing and electronic components. This structure provides more support points around the user's head, distributes the component weight over a larger area, and improves comfort and stability during use.

2 Design

2.1 Design procedure

2.1.1 Mechanical system

The mechanical structure was designed as the physical carrier of the wearable sensing node. Its role is to support the camera, PCB, battery, and other electronic components while maintaining wearing comfort and physical stability during operation. Since these components are concentrated near the front of the device to capture the user's walking direction, the mechanical design needed to reduce front-heavy imbalance and prevent the device from tilting forward. For this reason, the final prototype uses an adjustable head-ring structure rather than a simple glasses-style frame, providing more support points and distributing the component weight over a larger area.

The mechanical design was developed using CAD modeling, rapid prototyping, and simple load-distribution reasoning. CAD tools were used to define the head-ring geometry, component mounting locations, and relative positions of the camera, PCB, battery, and wiring. The prototype was then fabricated using 3D printing so that the team could quickly evaluate fit, comfort, component clearance, and physical stability. Qualitatively, the forward-tilting tendency can be related to the moment generated by the front-mounted components, $M = Wd$, where W is the component weight and d is the distance from the support point to the component center of mass. The final head-ring structure reduces this effect by providing more support points around the user's head and distributing the component load over a larger contact area.

2.1.2 Non-mechanical system

The final design uses a split architecture: the wearable device performs sensing and lightweight packetization, while a host computer performs perception, state orchestration, recording, web monitoring, and speech synthesis. The main alternative was to run more intelligence on the embedded device. That approach would reduce network dependence, but the XIAO ESP32S3 Sense does not have enough compute or memory margin for the Torch/Ultralytics models used for segmentation and detection. Moving inference to the host allowed the team to use higher-quality models, benchmark them directly, and adjust thresholds during testing.

The transport design also followed this tradeoff. WebSockets are convenient for control and audio, but large JPEG frames over WebSockets create unnecessary overhead and make packet loss harder to isolate. The final design therefore uses RTP/JPEG over UDP for video, following the JPEG payload structure in RFC 2435 [1], and retains WebSockets for command/configuration messages and audio. Because UDP video can be injected by any host on the same network if left unprotected, each RTP/JPEG packet includes an authentication extension with an AGLA magic value and a 16 B HMAC-SHA256 tag. The backend verifies the tag before reassembling the frame.

For perception, the team chose separate models and heuristics instead of one monolithic

model. Blind-path/crosswalk segmentation, obstacle detection, and traffic-light detection have different failure modes. Keeping them separate made the system easier to debug in the browser console and allowed each block to have its own threshold, confidence filter, and safety priority. The implementation uses FastAPI for the web server [2], Torch/Ultralytics for YOLO-family inference [3], DashScope for streaming ASR or cloud TTS when enabled [4], and Piper as a local TTS option [5].

2.2 Design details

2.2.1 Mechanical carrier and wearable structure

The mechanical portion of the project focused on providing a stable and comfortable physical carrier for the sensing, processing, and feedback components. In the early design stage, the team considered a glasses-like structure because it naturally places the camera near the user's line of sight. However, prototype testing showed that mounting the camera, PCB, battery, and related components mainly near the front of the frame created a front-heavy load distribution. This caused the glasses frame to tilt forward during use, reducing comfort, stability, and practical wearability.

To address this issue, the final mechanical design moved from a simple glasses frame toward an adjustable head-ring structure. Compared with the glasses-style prototype, the head-ring design provides more contact points around the user's head and distributes the device weight over a larger support area. This reduces the tendency of the device to rotate forward and improves stability during walking. The adjustable structure also allows the device to fit different head sizes, which is important for a wearable assistive device intended for repeated user testing.

The main functional requirement of the mechanical structure was to keep the sensing components facing forward while maintaining a secure and comfortable fit. The camera and other core components were concentrated near the front of the device so that the system could capture the walking direction of the user. At the same time, the supporting ring helped balance the load and prevent the front-mounted components from causing excessive torque on the user's face or nose bridge.

The mechanical design also considered manufacturability. The structure was designed to be 3D printed so that the team could quickly revise dimensions, mounting holes, and component positions after each prototype test. This rapid prototyping approach allowed the team to evaluate fit, component placement, and stability before final integration with the sensing and feedback modules.

Overall, the mechanical design improved the practical usability of the wearable system. The transition from a glasses-style frame to an adjustable head-ring structure made the device more stable, more comfortable, and better suited for carrying the front-facing camera and electronic components required by the navigation-assistance system.

2.2.2 Wearable sensing and firmware

The wearable node is based on the Seeed XIAO ESP32S3 Sense platform [6]. The firmware configures the camera for JPEG capture, uses a PDM microphone for mono PCM16 audio, reads an ICM42688 IMU over SPI, and drives an I2S speaker path for optional speech playback. The default example configuration uses SVGA video at 4 frames/s, JPEG quality 10, 16 kHz audio, 100 ms audio chunks, and a 50 Hz IMU stream. These values are generated into a firmware header from the TOML configuration so that WiFi credentials, server URL, sample rate, video settings, and UDP authentication key remain synchronized between host and device.

The firmware runs separate FreeRTOS tasks for camera, audio, IMU, and status LED. Camera frames are captured from PSRAM-backed frame buffers. Audio samples are read continuously from the PDM microphone and sent as bounded PCM16 chunks. IMU readings are serialized as JSON and sent on the control channel. Downlink speech is accepted as binary AGL1 packets and written to the I2S speaker as stereo 32-bit samples expanded from mono PCM16.

2.2.3 IMU PCB carrier

The wearable node uses a compact ICM42688-P IMU carrier PCB to expose the inertial sensor to the ESP32 firmware. The board brings out VDD, VDDIO, SCL/SCLK, SDA/SDI, ADO/SDO, CS, INT1, and INT2/FSYNC/CLKIN on two side headers, which made the IMU easier to mount in the wearable assembly than a loose sensor package. In the final firmware, the IMU is read through the SPI-style signals connected to the ESP32S3: SCLK, MISO, MOSI, CS, VDD, and GND. Figure 2 shows the PCB layout and pinout, and Figure 3 shows the corresponding schematic.

2.2.4 Packet formats and backend transport

The project defines an AGL1 binary packet for WebSocket transport. Each packet contains a magic value, version, packet type, flags, sequence number, timestamp, payload length, and CRC32. The backend rejects packets with the wrong magic value, unsupported version, bad length, payloads above the channel limit, unknown type, or CRC mismatch. Control messages are capped at 8 KiB, audio uplink and speech downlink are capped at 64 KiB per packet, and generic payloads are capped at 1 MiB.

The UDP video path is more specialized. The ESP32 parses the camera JPEG, extracts the scan data and quantization tables, and sends RFC 2435-compatible RTP/JPEG fragments. The backend checks RTP version, payload type, authentication extension, JPEG type, dynamic 8-bit quantization table length, fragment overlap, duplicate inconsistency, and frame timeout before rebuilding a standard JPEG. The resulting frame enters the same vision pipeline as a WebSocket video frame, so the rest of the system is independent of the selected video transport.

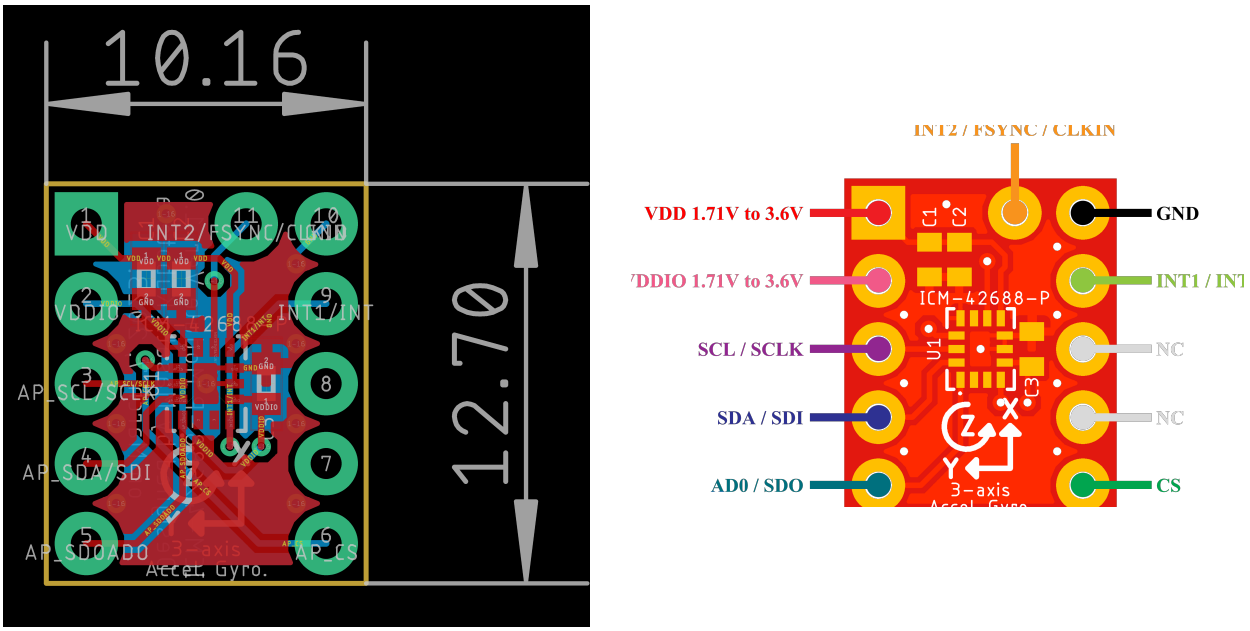

Figure 2: ICM42688-P IMU carrier PCB layout and pinout used by the wearable sensing node. The board is approximately 10.16 mm by 17.27 mm and exposes power, SPI/I2C-compatible signal labels, chip select, and interrupt pins on through-hole headers.

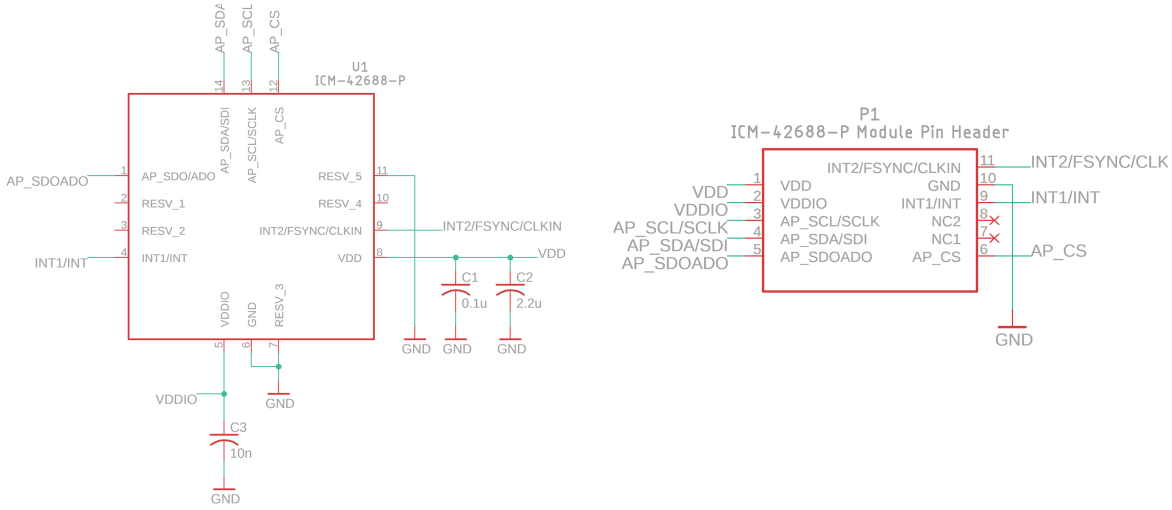

Figure 3: Schematic details for the ICM42688-P IMU carrier. The enlarged view shows the sensor connections, separate VDD and VDDIO rails, decoupling capacitors, and the two-row pin-header interface for the wearable electronics assembly.

2.2.5 Application server and monitoring console

The Python server creates a device manager, vision pipeline, navigation state machine, speech hub, ASR service, and FastAPI web app. The browser console provides live video, overlays, IMU status, speech/command logs, model status, video statistics, backend benchmark status, recording controls, camera settings, and tuning controls. This console is not only a convenience feature. It is an engineering instrument: it lets the team see which detection triggered each spoken prompt and adjust thresholds without recompiling firmware.

The device manager also records synchronized data. When recording is enabled, it stores JPEG frames and a JSONL metadata stream containing frame index, timestamp, latest vision observation, navigation state, tuning values, and IMU values. This made it possible to debug failures after a walking session instead of relying on memory or live observation.

2.2.6 Vision pipeline

The vision pipeline decodes incoming JPEG frames with OpenCV and runs up to three Torch/Ultralytics model paths. The blind-path model produces segmentation masks for the tactile path and crosswalk classes. The obstacle model is a fixed-prompt YOLOE segmentation model exported from an open-vocabulary model into 29 obstacle classes, including vehicles, bicycles, poles, bollards, cones, stones, boxes, benches, chairs, and similar hazards. The traffic-light model produces detection candidates for go, stop, countdown, and non-signal classes.

Each model output is converted into a compact observation consumed by the state machine. Segmentation masks are summarized by area ratio, normalized center offset, vertical position, fitted angle, confidence, and an approximate contour. Detections are summarized by label, confidence, bounding box, and area ratio. The traffic-light selector applies confidence, area, optional region-of-interest, stop/yellow priority, non-signal suppression, and conflict checks before exposing a final signal state.

Table 2: Key final perception and state thresholds

Parameter	Final value	Purpose
Blind-path model confidence	0.35	Minimum model confidence for blind-path segmentation.
Obstacle confidence	0.35	Minimum obstacle confidence before corridor filtering.
Traffic-light confidence	0.20	Base model threshold before signal selection filters.

Continued on next page

Table 2: Key final perception and state thresholds (continued)

Parameter	Final value	Purpose
Crosswalk confidence	0.65	Minimum crosswalk segmentation confidence after model inference.
Road alert area ratio	0.002	Allows early crosswalk warning during blind-path navigation.
Road stop area ratio	0.015	Marks a closer road/crosswalk cue that should stop the user.
Road stop bottom ratio	0.55	Requires the crosswalk contour to extend near the bottom of the image before stop guidance.
Crossing green frames	2	Requires repeated green/crosswalk evidence before entering active crossing.
Crossing timeout	45 s	Warns when crossing mode remains active unusually long.
Speech cooldown	2 s	Suppresses repeated non-urgent guidance.

2.2.7 Navigation state machine

The navigation state machine has four modes: idle, blind-path navigation, crossing, and traffic-light checking. Voice or debug commands start and stop these modes. In blind-path mode, the controller first checks for obstacles in the walking corridor, then checks for near-road/crosswalk cues, then converts tactile-path offset and angle into spoken guidance. Offset thresholds of ± 0.18 trigger left/right alignment prompts, while angle thresholds of $\pm 12^{\circ}$ trigger rotation prompts. If the path is centered and aligned, the system says to continue straight.

Crossing mode is deliberately stricter. Red and yellow signals have immediate priority. Green alone is not enough; the state machine also requires crosswalk progress evidence before starting active crossing. Progress evidence requires the crosswalk contour bottom to be at least 0.70 of image height and the mask area ratio to be at least 0.12. Once active crossing starts, the controller looks for completion evidence: after at least four active frames, the crosswalk must move toward the upper image region, have bottom ratio at most 0.35, and area ratio at most 0.08 for three candidate frames. This logic does not certify that crossing is safe, but it reduces the chance that one unstable frame changes the mode.

The state machine also debounces speech. Detection-loss messages must repeat for three frames before being spoken. Repeated identical prompts are suppressed, and non-urgent

prompts are rate-limited by a 2 s cooldown. Urgent messages such as obstacles, red lights, road-stop warnings, and crossing timeout warnings bypass the normal cooldown.

2.2.8 Speech, ASR, and audio output

The final design supports both UI-only speech and device audio playback. In UI mode, the server broadcasts the intended spoken text to the browser. In device mode, the server synthesizes PCM16 audio using either DashScope TTS or local Piper voices, then sends speech packets to the ESP32 over the control WebSocket. Navigation events drop older pending navigation speech so that stale guidance does not play after a newer instruction is generated.

The ASR service uses DashScope Paraformer realtime recognition when enabled. It accepts PCM16 audio from the device, sends it to the ASR service, and only forwards final sentence results into the command handler. This choice avoids triggering state transitions from partial speech. The command interface also supports browser/debug text injection, which was important for repeatable testing without relying on cloud ASR during every development run.

3 Verification

3.1 Verification approach

Verification was organized around the final risk areas of the complete prototype: mechanical stability, component mounting, packet integrity, authenticated video transport, configuration safety, perception postprocessing, navigation state transitions, speech output, and web/server behavior. Because testing a wearable navigation aid in live traffic can be unsafe before the system is proven, the first verification layer was automated software testing with synthetic packets, synthetic observations, and mocked services. In parallel, the mechanical carrier was verified through prototype assembly checks, component fit checks, camera-alignment checks, and basic wearing observations. Hardware-in-the-loop walking tests and blind-user studies remain future validation steps, not completed certification.

The completed automated software test suite passed:

170 tests run, 170 passed.

These tests were run with:

```
uv run python -m unittest discover -s tests
```

The warnings printed during the run were expected negative-test logs from malformed UDP and WebSocket packet cases. The mechanical carrier also passed the project-level verification criteria for component mounting, wearable fit, front-facing alignment, and physical stability during prototype operation.

3.2 Block-level verification

Table 3 summarizes the main block-level verification evidence. The detailed requirement and verification table is included in Appendix A.

Table 3: Verification summary by subsystem

Subsystem	Evidence	Result
Mechanical carrier	Prototype inspection verified that the camera, PCB, battery, and related electronic modules could be mounted on the wearable carrier. Basic wearing checks confirmed that the adjustable head-ring structure provided better support than the initial glasses-style frame, kept the front-facing camera aligned with the user's walking direction, and reduced the forward-tilting tendency caused by front-mounted components.	Passed

Continued on next page

Table 3: Verification summary by subsystem (continued)

Subsystem	Evidence	Result
Configuration	Tests validate device ID length, unknown fields, audio chunk limits, ASR sample-rate agreement, video authentication key format, and speech/device-mode compatibility.	Passed
AGL1 packet protocol	Tests verify packet round trip, CRC rejection, payload-limit enforcement, and channel-specific packet handling.	Passed
UDP RTP/JPEG video	Tests cover authenticated packet acceptance, missing authentication rejection, invalid RTP version/type rejection, fragment overlap rejection, duplicate inconsistency rejection, JPEG reconstruction, and stale-session handling.	Passed
Vision postprocessing	Tests cover YOLO output parsing, nonmaximum suppression, box/mask mapping, mask summaries, and class filtering for fixed obstacle labels.	Passed
Navigation state machine	Tests cover blind-path start/stop, correction prompts, obstacle priority, translated obstacle labels, road/crosswalk alerts, red/yellow/green priority, green stability, crossing completion, timeout, and optional crossing obstacle waits.	Passed
Speech and ASR plumbing	Tests cover TTS configuration validation, local/DashScope speech sink behavior at the interface level, and ASR command injection without requiring live cloud recognition.	Passed
Web application	Tests cover app creation, health/frame endpoints, device configuration endpoints, disconnect endpoint, WebSocket packet currentness checks, and recording/status paths.	Passed

3.3 Requirement verification

The wearable mechanical carrier requirement was verified through prototype assembly and basic wearing checks. The final mechanical carrier provides mounting locations for the camera, PCB, battery, and related electronic components. The camera can be positioned toward the user's walking direction, which is necessary for front-facing video acquisition. Compared with the initial glasses-style carrier, the adjustable head-ring structure provides more support points around the user's head and distributes the component load over a larger contact area. These checks confirmed that the sensing electronics can

be physically mounted, aligned, and worn in a stable form during prototype operation. Therefore, the mechanical carrier passed the project-level verification requirement for the final prototype.

The hands-free sensing requirement is supported by the firmware architecture and backend interfaces. The ESP32 firmware runs independent tasks for camera, audio, IMU, and LED status, while the backend exposes separate channels for video, control, audio uplink, and speech downlink. The automated tests do not prove RF stability in every physical environment, but they do prove that the software rejects bad packets and can process the accepted packet forms.

The blind-path requirement is verified at the state-machine and perception-interface level. Synthetic observations exercise path offset, path angle, obstacle position, obstacle area, and crosswalk presence. The tests show that an obstacle centered on the blind path produces a stop warning, while a side obstacle outside the path corridor does not interrupt straight guidance. The tests also show that a missing blind path requires stable evidence before a "path not visible" prompt, which reduces false loss messages from a single bad frame.

The crosswalk and signal requirement is verified by targeted state-transition tests. A red signal immediately produces a stop prompt. Yellow/countdown classes produce a caution prompt. Green does not start crossing unless crosswalk progress evidence is also present for the required number of frames. Once active crossing begins, the completion logic requires repeated evidence that the crosswalk has moved out of the near field. These tests match the final design goal: the system should be conservative and should not announce crossing from one isolated green detection.

The robust transport requirement is the strongest verified area. The UDP reassembler is tested against malformed RTP headers, missing extensions, bad authentication tags, unsupported JPEG parameters, duplicate fragments, overlapping fragments, missing quantization data, and session replacement. The AGL1 packet tests verify CRC and length checks. This matters because a wearable assistant can fail dangerously if stale or injected frames are accepted as valid.

3.4 Unverified and partially verified items

The final implementation does not include the item-search and MediaPipe hand-guidance mode described in the design document. It should therefore not be claimed as a verified feature. The mechanical carrier passed the project-level verification requirement for the final prototype, including component mounting, camera alignment, wearable fit, and physical stability during basic operation. However, extended mechanical validation such as long-duration comfort testing, repeated walking trials with multiple users, drop or vibration testing, rain/dust robustness testing, and blind-user usability studies remains future work. Long-duration battery life and real-world blind-user usability also require separate verification by the full team. In addition, the current test suite uses synthetic images or mocked model interfaces for many cases; final deployment would require quan-

titative field data across different lighting, sidewalks, intersections, camera angles, and network conditions.

4 Costs

The cost estimate separates reimbursed prototype purchases, cloud/service costs, mechanical prototyping material cost, and labor. The reimbursed purchase list includes the main electronic components, sensing modules, audio components, battery-related items, documentation/test-support items, software-service cost, and 3D-printing material cost used during prototyping. The mechanical carrier was part of the final prototype, serving as the wearable platform for mounting and aligning the sensing and electronic components. Its direct material cost is represented by the 3D printing material entry in Table 4. The receipts were issued in RMB, so the table reports the paid receipt values directly instead of applying a currency conversion.

Table 4: Reimbursed prototype purchase cost

Item	Qty.	Unit cost (RMB)	Paid cost (RMB)
XIAO ESP32S3 Sense development board	1	95.00	95.00
5 V lithium battery / battery pack	1	44.98	44.98
Nusign A4 clipboard for documentation and test support	1	21.05	21.05
Planwith folder for documentation and test support	1	12.80	12.80
ICM42688 IMU module / sensor	1	51.00	51.00
XIAO ESP32S3 Sense development-board kit	1	104.90	104.90
8 ohm, 1 W round speaker	1	5.22	5.22
MAX98357 I2S audio-amplifier module	1	13.80	13.80
UCloud model-training / software-service fee	1	1000.00	1000.00
ZDCP drone battery	1	28.80	28.80
3D printing material for mechanical carrier	1	10.00	10.00
Reimbursed prototype purchase total			**1387.55**

The ZDCP battery line uses the corrected paid amount of 28.80 RMB from the receipt attachment, rather than the 37.10 RMB value that appeared in the reimbursement summary draft. After adding the 10.00 RMB 3D printing material cost for the mechanical carrier, the resulting reimbursed prototype purchase total is therefore 1387.55 RMB. The host laptop or GPU workstation is treated as existing laboratory infrastructure. If this prototype were

converted into a self-contained product, the compute cost would increase because the host would need to be replaced by an embedded AI module or custom mobile compute board. In addition, a manufacturable version would require a more refined mechanical carrier with production-ready material selection, enclosure design, wiring protection, and adjustable fit features. The current split architecture and prototype-level mechanical carrier are therefore economical for a senior design demonstration but not a final manufacturing architecture.

ECE 445 also asks teams to report labor with a standard multiplier. This is not a reimbursed expense; it is a hypothetical engineering-cost estimate using the course formula:

$$\text{labor cost} = \text{hourly rate} \times \text{hours} \times 2.5.$$

Table 5 uses an assumed engineering rate of \$50/h and a conservative equal workload estimate of 120 h per team member for the complete prototype design, implementation, integration, and verification work. This includes the electrical, firmware, software, perception, communication, speech-feedback, mechanical-carrier design, mechanical integration, and testing efforts. This keeps the labor estimate separate from the receipt-based hardware, service, and material purchases above.

Table 5: Estimated engineering labor cost

Contributor	Hours	Rate	Labor cost
Jiashen Ren	120	\$50/h	\$15,000
Haoyu Zhu	120	\$50/h	\$15,000
Jinnan Zhang	120	\$50/h	\$15,000
Darui Xu	120	\$50/h	\$15,000
Total	**480**		**\$60,000**

The direct reimbursed prototype purchase cost is 1387.55 RMB. The estimated non-reimbursed engineering labor cost is \$60,000 under the ECE 445 formula. The two values are reported separately because they answer different questions: one is actual project spending based on receipts, and the other is the course-required engineering labor estimate. The mechanical carrier is included in both the direct prototype purchase cost through the 3D printing material entry and the engineering labor estimate through the mechanical-carrier design, integration, and testing effort.

5 Conclusions

The final non-mechanical system demonstrates a working architecture for wearable navigation assistance. It captures first-person video, audio, and IMU data from an ESP32-based wearable node; authenticates and reconstructs UDP video; runs Torch/Ultralytics perception models; converts detections into stateful navigation guidance; and delivers feedback through a browser console or device audio playback. The strongest completed contributions are the reliable transport layer, the mode-aware navigation state machine, the runtime-tunable vision pipeline, and the automated verification suite.

The project also reached a clear boundary. It should be described as an assistive prototype, not as a finished mobility device. The original item-search and hand-guidance workflow was not implemented in the final codebase, and the crossing guidance has not been validated through the volume of real-world trials that would be required before any independent use. The next technical steps are to collect more walking and intersection recordings, measure model precision and recall by scene type, tune thresholds using those recordings, add field metrics to the report, and decide whether item search is still part of the product scope.

5.1 Ethical Considerations

This project interacts with a safety-critical activity: pedestrian navigation. The IEEE Code of Ethics requires engineers to hold public safety, health, and welfare paramount [7]. For this project, that means the system must not be marketed or demonstrated as a replacement for a cane, guide dog, trained orientation practice, or independent traffic judgment. The speech prompts should be treated as advisory. Any demo near roads should use a sighted assistant and controlled conditions.

Privacy is also a central issue. The system captures first-person video, nearby speech, user commands, IMU data, and inferred scene information. It can store recordings for debugging and may send audio to cloud services when ASR or DashScope TTS is enabled. A responsible deployment would require clear consent, visible capture indicators, short retention windows, secure storage, and a local-only mode when cloud services are not acceptable.

5.2 Broader Impact

If developed responsibly, the project could make navigation-assistance research more accessible by showing that low-cost wearable sensing can be paired with host-side perception and speech feedback. Economically, the prototype uses inexpensive embedded hardware and existing compute resources, which is suitable for experimentation. Environmentally, the current split architecture avoids replacing the host computer with a power-hungry embedded AI board during prototyping, but a product version would need careful battery and component-lifetime analysis. Societally, the main positive impact is increased independence for users with visual impairment; the main risk is overconfi-

dence in a system that has not yet been validated across enough people, places, weather, lighting, and traffic conditions.

References

[1] L. Berc, W. Fenner, R. Frederick, and S. McCanne, *RTP Payload Format for JPEG-compressed Video*, RFC 2435, 1998. Accessed: May 15, 2026. [Online]. Available: https://www.rfc-editor.org/rfc/rfc2435.

[2] FastAPI. "FastAPI Documentation," Accessed: May 15, 2026. [Online]. Available: https://fastapi.tiangolo.com/.

[3] Ultralytics. "Ultralytics YOLO Documentation," Accessed: May 15, 2026. [Online]. Available: https://docs.ultralytics.com/.

[4] Alibaba Cloud. "DashScope Model Studio Documentation," Accessed: May 15, 2026. [Online]. Available: https://help.aliyun.com/zh/model-studio/.

[5] OHF-Voice. "Piper Text to Speech," Accessed: May 15, 2026. [Online]. Available: https://github.com/OHF-Voice/piper1-gpl.

[6] Seeed Studio. "Seeed Studio XIAO ESP32S3 Sense," Accessed: May 15, 2026. [Online]. Available: https://wiki.seeedstudio.com/xiao_esp32s3_getting_started/.

[7] IEEE. "IEEE Code of Ethics," Accessed: May 15, 2026. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html.

Appendix A Requirement and Verification Table

Table 6: Detailed requirement and verification matrix

Requirement	Verification method	Evidence	Status
Wearable mechanical carrier supports component mounting and stability	Inspect the assembled prototype, check component placement and camera alignment, and perform basic wearing observations with the final carrier.	The camera, PCB, battery, and related electronic modules were successfully mounted on the wearable carrier. The adjustable head-ring structure provided better support than the initial glasses-style frame, kept the front-facing camera aligned with the user's walking direction, and reduced the forward-tilting tendency caused by front-mounted components.	Passed
Configuration rejects unsafe settings	Run configuration tests for invalid audio chunk sizes, missing UDP key, malformed key, mismatched ASR sample rate, unknown fields, invalid transport modes, and invalid speech mode.	Covered by automated configuration unit tests.	Passed
AGL1 packets preserve payload integrity	Pack and unpack valid packets; corrupt payload bytes; alter length fields; lower per-channel payload limit.	CRC, magic/version, packet type, and length checks reject malformed packets.	Passed
UDP video rejects unauthenticated packets	Send RTP/JPEG packets with no extension or wrong HMAC tag.	Backend drops packet and emits device-error messages.	Passed
UDP video reconstructs valid JPEG frames	Split a fixture JPEG into RTP/JPEG fragments with quantization tables and marker bit, then reassemble.	Backend rebuilds a standard JPEG and passes it to video handling.	Passed

Continued on next page

Table 6: Detailed requirement and verification matrix (continued)

Requirement	Verification method	Evidence	Status
Blind-path mode can be commanded hands-free	Inject start/stop navigation command text through the command interface.	State machine enters and exits blind-path mode with speech confirmation.	Passed
Blind-path guidance follows path geometry	Provide synthetic blind-path observations with left/right offset and angle errors.	Controller emits left/right alignment, left/right turn, or straight guidance.	Passed
Obstacles on path override normal guidance	Provide obstacle detections centered on the blind-path corridor and off to the side.	Centered obstacles produce stop warnings; side obstacles do not stop the user.	Passed
Crosswalk warning interrupts blind-path guidance	Provide crosswalk mask evidence with sufficient area and bottom position during blind-path mode.	Controller emits road/crosswalk warning and latches the stop state.	Passed
Red/yellow signals have priority in crossing mode	Provide traffic-light observations with stop or countdown labels.	Controller emits red or yellow prompt even when crosswalk is missing.	Passed
Green crossing requires repeated evidence	Provide repeated green observations with near crosswalk progress evidence.	Controller waits for the configured green-frame count before active crossing.	Passed
Speech output can target UI or device	Validate speech configuration and send synthesized PCM16 chunks through the speech sink interface.	UI sink always broadcasts text; device mode requires audio-down configuration.	Passed
Web application endpoints and monitoring operate correctly	Run web-application tests for app creation, health/frame endpoints, device configuration endpoints, disconnect endpoint, WebSocket packet currentness checks, and recording/status paths.	Backend endpoints and browser-facing interfaces behave correctly under the tested cases.	Passed

Continued on next page

Table 6: Detailed requirement and verification matrix (continued)

Requirement	Verification method	Evidence	Status
Final system is safe for independent use	Conduct statistically meaningful field testing with blind or visually impaired users in varied real intersections.	Not completed within this project scope.	Not verified
Item-search/hand-guidance mode works	Implement and test target object localization plus hand guidance.	Feature was removed from final scope.	Not implemented