USC Navegar

Michael Castro, Riley Milligan, Way Zheng

USC Viterbi, School of Engineering

EE 459Lx: Embedded Systems Design Laboratory

May 13, 2025

1. Introduction:

Navigating a busy campus environment presents significant challenges for visually impaired individuals. Tasks that may seem simple to others, like figuring out where they are, which direction they are facing, or whether they are on the right route, can become challenging. For example, traveling from Taper Hall (THH) to Olin Hall (OHE) at USC can present a series of obstacles, particularly for users unfamiliar with the layout of the campus. This difficulty can lead to an exhaustive amount of time spent trying to understand the correct route to a destination, feelings of confusion, and potential hazards from aimless wandering. To address these challenges, we developed USC Navegar, a pair of smart glasses designed to enhance user mobility, safety, and independence through guided navigation.

USC Navegar actively engages the user with step-by-step navigation commands, delivering simple, real-time guidance. By providing a straightforward and efficient path between campus locations, Navegar minimizes the complexity of navigating an unfamiliar or evolving campus environment. Using a combination of GPS, compass sensors, and real-time feedback, the device ensures that users can reach their destination with confidence, efficiency, and a much lower cognitive burden.

An essential feature of USC Navegar is user-defined destination input. At the start of their journey, users select their desired destination by pressing a button located on the glasses frame. Each button press cycles through a set list of major campus locations, enabling users to easily and independently select where they want to go. Once the destination is set, the system

dynamically calculates the best route and actively monitors the user’s location to provide ongoing guidance.

Haptic feedback is another critical element incorporated into the system to make it more inclusive and user-friendly. A small buzzer embedded within the glasses delivers tactile signals to the user. Different patterns correspond to specific navigation commands: for instance, a short buzz might indicate "move forward," a double buzz might signal "turn right," a triple buzz for "turn left," and a long continuous buzz for "turn around." This silent communication method enhances the user experience by supplementing auditory guidance and ensuring that navigation assistance is accessible even in noisy environments.

The navigation algorithm behind USC Navegar was crafted to prioritize user experience, safety, and reliability. It incorporates several important features to ensure smooth operation in a real-world, campus-sized environment. The first major component of the algorithm is the concept of Campus Zones. This zoning allows the system to speed up the search process by only checking for nearby buildings or intersections within the user’s current zone, instead of the entire campus. It also helps reduce GPS noise, as small inaccuracies in the GPS signal are less likely to mislead the system when checks are localized to a specific area.

Once the user's current zone is determined, the system uses an approximate distance estimation to quickly identify the nearest node (i.e., building or intersection) without resorting to computationally heavy geographic formulas. If more precision is needed, such as when

determining if the user has arrived at a destination, a more accurate distance calculation based on the haversine formula is used.

To generate full paths between points, USC Navegar employs an algorithm across the campus' mapped intersection network. This ensures that users are guided through the most efficient and straightforward route possible, minimizing unnecessary turns and detours. Once a path is computed, the device dynamically tracks the user's progress from one intersection to the next, updating commands based on real-time heading and distance information. The entire navigational flow was designed to mirror the needs of visually impaired users — ensuring frequent updates, simple instructions, and recalculation if a mistake is made.

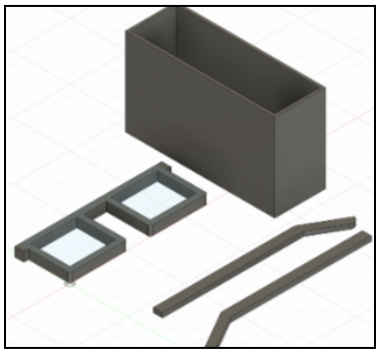


Figure 1: The Design of the 3-D Printed Glasses

Beyond the algorithmic components, the physical design of USC Navegar was also carefully considered. The glasses were made lightweight and ergonomic to ensure that they can be worn for extended periods without discomfort. Inspired by the recently released Minecraft movie, the physical appearance of the device features a playful yet stylish aesthetic, drawing from the pixelated, blocky visual themes of the film. This design approach helps make the device more appealing and approachable for a younger or broader audience, breaking away from sterile or clinical designs that might otherwise alienate users.

On the hardware side, USC Navegar integrates a GPS module and a digital compass module to deliver reliable heading and location information to the navigation algorithm. The GPS provides latitude and longitude data with sufficient precision for campus-level navigation, while the compass provides real-time heading information. These two sensor inputs are tightly integrated into the algorithm to ensure that user guidance remains consistent and accurate even if the user changes their orientation while walking.

Throughout the entire project, we placed a strong emphasis on ethical considerations to ensure that USC Navegar benefits its users without unintended risks. In line with the IEEE Code of Ethics, we made decisions that prioritize user safety, reliability, and comfort. For example, when selecting the buzzer module for haptic feedback, we ensured that the strength was noticeable but not so strong as to cause discomfort or distraction. Similarly, we avoided using modules that could generate high heat or pose other risks when worn on the face. Our adherence to ethical design principles also guided our approach to user privacy — the system does not collect, store, or transmit any personal location data beyond what is necessary for immediate navigation.

2. System Overview

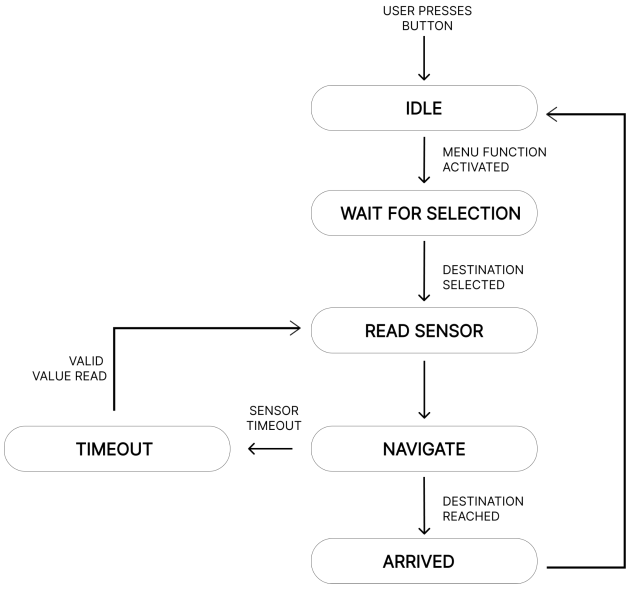


Figure 2: State-Machine Diagram of USC Navegar

USC Navegar is designed to guide users across the USC campus, demonstrated in the six-state finite state machine. The system begins in the IDLE state, where it awaits interaction. Once the user presses a physical button, it transitions to WAIT_FOR_SELECTION, enabling a destination selection process. The user selects from preloaded waypoints—specific locations on campus identified by their GPS coordinates—through successive button presses. Once a location is confirmed, the system enters READ_SENSOR, activating the GPS and compass modules to determine current position and orientation.

The GPS module receives standard NMEA sentences such as $GPGGA, which are then parsed to extract latitude and longitude values. These values are provided in Degree and Decimal Minutes (DMM) format and must be converted to pure degrees using the following formula:

Latitude = degrees + (decimal_minutes / 60)

For example, a coordinate string like 4807.038 would be parsed as:

48 + (7.038 / 60) ≈ 48.1173°

This absolute coordinate system allows the algorithm to pinpoint the user's location on a campus-wide grid.

To reduce computational complexity, the campus is divided into discrete zones, each defined by a range of latitude and longitude values. This zoning method narrows the area of search during navigation. The current zone is calculated as follows:

```
zone_row = floor((max_lat - current_lat) / zone_height)
zone_col = floor((current_long - min_long) / zone_width)
current_zone = (zone_row * 3) + zone_col
```

Once the system has established the current location, it enters the NAVIGATE state, executing a Breadth-First Search (BFS) algorithm to find the shortest path across campus. Each campus intersection is treated as a node, with stored coordinates and links to adjacent nodes. BFS ensures that the shortest route by intersection count is found before the system transitions to directional feedback.

Direction is determined using a compass module that reads X and Y axis magnetometer values. The system calculates the user's heading and compares it to the direction of the next node. The heading is computed using:

```
heading_degrees = atan2(y, x) * 180.0 / PI;
if (heading_degrees < 0) heading_degrees += 360.0;
```

This allows the system to determine whether the user should turn left, right, go forward, or turn around. If sensor data becomes unreliable—due to GPS signal degradation or fluctuating compass readings—the system enters a TIMEOUT state, attempting to reacquire sensor data. Once the user comes within 5 meters of the destination, the system enters ARRIVED, where it triggers a “destination reached” buzz pattern and returns to IDLE, ready for the next route.

The combination of zoning, directional feedback, and sensor integration ensures that the user is provided with near-real-time guidance despite the sensitivity of the system to environmental conditions. USC Navegar leverages both software-based edge case handling and real-time sensor validation to deliver reliable pedestrian navigation on campus.

3. Hardware Design

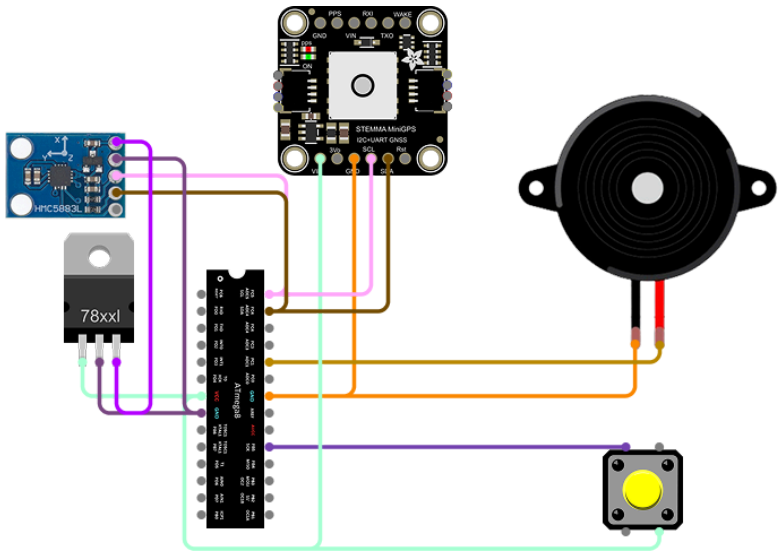


Figure 3: USC Navegar Pin-out Diagram

The USC Navegar hardware is composed of carefully selected components optimized for portability, power efficiency, and modular sensor integration. At the heart of the system is the ATmega328P microcontroller, operating at 16 MHz with 32KB of Flash memory and 2KB of SRAM. This chip was chosen for its low power consumption (15mA active at 5V) and its proven performance in embedded applications, providing consistent 50ms sensor polling and decision-making latency. The microcontroller's I/O includes UART, I2C, and GPIO lines that facilitate direct communication with external sensors and peripherals.

Positioning is handled by the Stemma MiniGPS module, which outputs location data via NMEA 0183 sentences. This module operates on 3.3V and draws approximately 25mA during active use. Its interface flexibility (UART and I2C support) makes it ideal for embedded applications. Accuracy is rated at ±2.5m CEP(circular error probable), and field testing revealed consistent performance within ±3m in open areas. However, performance deteriorates in indoor or urban "canyon" environments, with cold start times reaching up to 360 seconds near buildings compared to 27 seconds in open sky. The module provides data such as:

```
$GPGGA,123519,4807.038,N,01131.000,E,...
```

This string is parsed and converted into degrees to determine user position.

Directional sensing is handled by the PMOD CMPS2 magnetometer, which provides 3-axis magnetic field readings with 12-bit resolution. The system only uses the X and Y axes to compute user heading, which is calculated as follows:

```
heading = atan2(Y, X) * 180.0 / PI;
```

This allows the microcontroller to determine cardinal direction. The compass operates at 3.3V ±0.1V and communicates over I2C. It is highly sensitive to voltage variations and environmental interference. To address this, a voltage regulator (LD1117-3.3) was added, converting the system's 5V power supply to a stable 3.3V ±0.05V required by the compass. This addition drastically improved reading stability.

User interaction is implemented through a momentary tactile button using a pull-up resistor configuration. Debounce logic is handled in software using a ~50ms delay window. This button allows users to start the navigation sequence and cycle through destinations. For feedback, a piezoelectric buzzer is used, which emits different patterns based on navigation commands and system states (e.g., arrival confirmation). It draws roughly approximately 30mA when active.

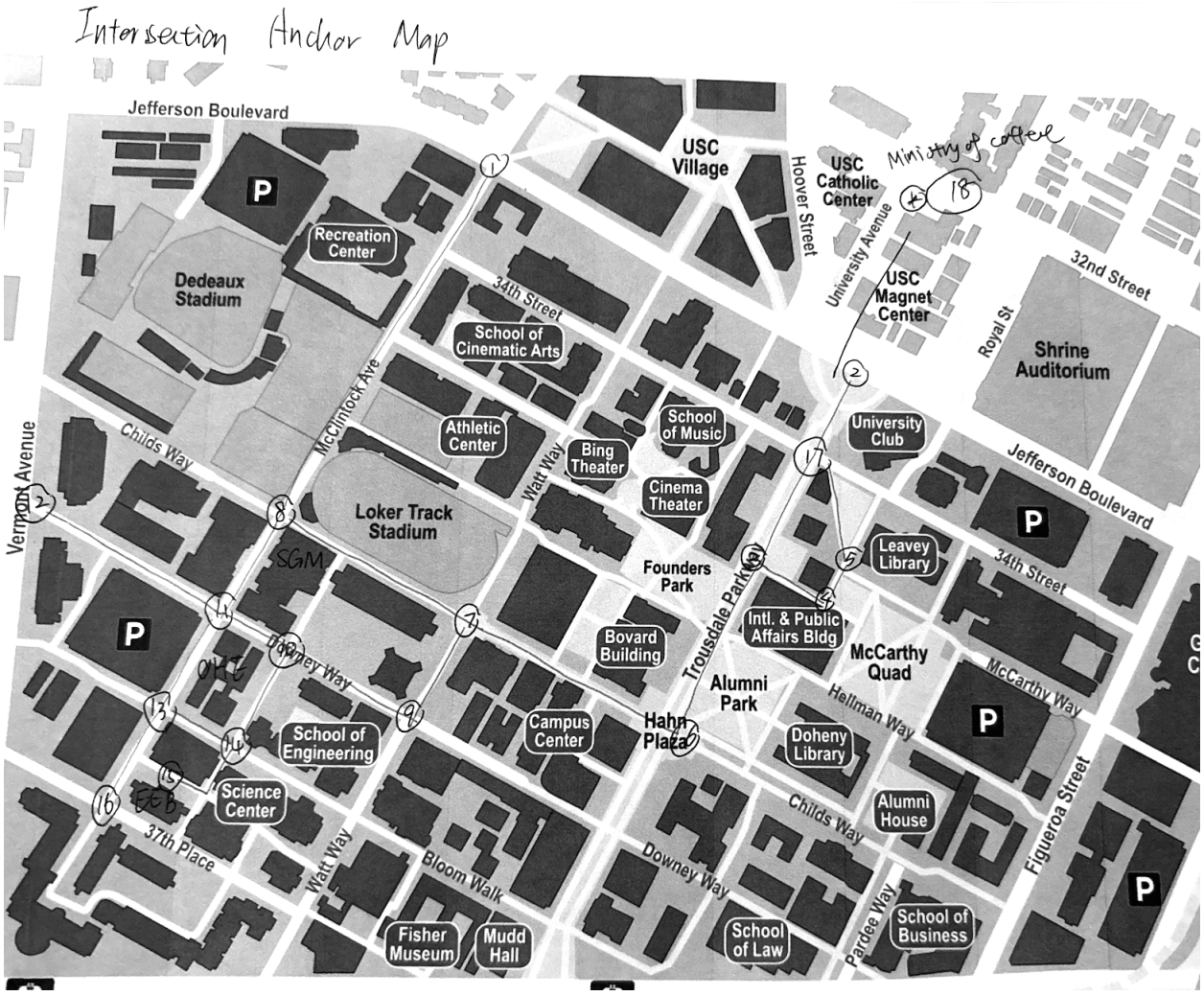
Originally, the team planned to use the NEO-M8 GPS and HMC5883L compass, but compatibility issues and inconsistent readings led to switching to the current modules. Key challenges included faulty solder joints, communication mismatches from 5V vs. 3.3V logic levels, and unreliable compass readings from a counterfeit sensor module. These issues were resolved through hardware replacements and stricter power regulation.

Despite challenges with sensor stability and power sourcing, the hardware architecture supports a robust navigation pipeline. Future improvements could account for IMUs(inertial measurement units) for tilt compensation or applying sensor fusion techniques to enhance GPS accuracy in occluded environments.

4. **Software Design**

To effectively guide the user, USC Navegar distinguishes between two types of landmarks: **building anchors** and **intersection nodes**. **Building anchors** represent specific destination points, like entrances to buildings, and are used primarily for identifying the user's current location at the start of navigation. These anchors are stored as GPS coordinates and are compared

to the user's real-time location using a proximity threshold (typically within 10 meters) to infer where the user is. In contrast, **intersection nodes** form the internal navigation graph and represent key decision points where users must turn or continue straight. The Breadth-First Search algorithm runs on this graph of intersections to calculate the shortest path, but the journey begins and ends at building anchors which are not part of the BFS traversal but are instead connected to their nearest intersections.



To ensure accurate turn instructions, USC Navegar calculates the angular difference between the user's current compass heading and the direction of the next navigation node. These angles are bucketed into simple categories to reduce user confusion:

- If the angle difference is **less than 30°**, the instruction is "go forward."
- If it is **between 30° and 135°**, the instruction is either "turn left" or "turn right," depending on the sign of the difference.
- If the angle is **greater than 135°**, the instruction is "turn around."

This thresholding helps avoid overcorrection from minor sensor fluctuations and ensures that guidance remains intuitive. Additionally, to prevent erratic feedback caused by rapidly shifting compass readings (common near metal objects or electronics), a smoothing technique is applied. The heading is averaged over the last few readings (typically 3–5 samples) to stabilize the direction before issuing new instructions.

Real-time updates are critical to maintaining accurate guidance. The system polls GPS and compass data approximately once per second. On each update, the user's current location and heading are re-evaluated. If the user deviates significantly from the expected path—for example, if they stray more than 10 meters from the next expected node or move in the wrong direction—Navegar recalculates the heading to that node and updates the direction. However, the actual BFS path is not recomputed unless the user manually restarts the process, which helps conserve processor resources on the embedded system.

To ensure accurate and reliable navigation, USC Navegar performs multiple validation steps on GPS data before it is used in pathfinding. First, the system checks that the latitude and longitude lie within the Earth's geographic limits. Then, it further restricts the acceptable region to the state

of California, based on known bounds for the USC campus area. Finally, it rejects any readings suspiciously close to (0,0), a common error output from GPS modules when signal acquisition fails. The system maintains three global variables that reflect the GPS state at any time: `current_lat`, `current_lon`, and `gps_valid`. These are updated only when a valid reading is received. Other parts of the navigation software, including zoning, direction computation, and proximity detection, rely on the accuracy of these global values. If `gps_valid` is false, the FSM will not transition into the NAVIGATE state and may instead enter TIMEOUT or wait until data is reacquired.

USC Navegar integrates three primary hardware components with the ATmega328P microcontroller: a GPS module (PM1010D), a compass module (Pmod CMPS2 with MMC34160PJ), and a user input button. All sensor communications are conducted over a shared I2C bus, while the button is connected to a GPIO pin configured for digital input.

The GPS module is initialized by sending a PMTK command that enables only the `$GNRMC` sentence, reducing processing overhead and bandwidth. GPS data is read in fixed-size I2C bursts and assembled into complete NMEA sentences using a line buffer. Only complete `$GNRMC` sentences are parsed. If the fix status is valid, the resulting coordinates are converted and stored in global state variables (`current_lat`, `current_lon`) for use by the navigation system.

The compass module is polled periodically for X and Y magnetometer readings. These raw values are used to calculate the user's heading relative to magnetic north. The compass data is not filtered by the module itself, so smoothing is applied in software to mitigate fluctuations from nearby interference.

User interaction is managed through a single push-button that triggers finite state machine transitions. Short presses are used to cycle through destination options, while a long press or timeout confirms the selection. To ensure reliable detection, a simple debounce check is performed in software.

All sensor reads and button inputs are handled in the main control loop. Timing and sequencing are designed to avoid blocking behavior and to support continuous updates during navigation. Data is only considered valid if it passes internal consistency checks (e.g., coordinate range, fix status, signal reliability). This allows the system to recover gracefully from sensor noise, I2C glitches, or temporary loss of GPS fix.

5. User Interface and Operation

USC Navegar is operated using a single tactile button and provides feedback through vibrations and periodic audio cues. After system startup, the user is prompted to press the button to begin. This triggers a **5-second window**, during which each additional press is counted to select a destination (e.g., two presses for "Tommy Trojan", four for "GFS"). The destination is confirmed automatically after the window closes, and a short vibration indicates that navigation is starting.

During navigation, the system continuously monitors GPS position and compass heading, updating every **3 seconds**. Based on this data, it calculates the direction to the next node and delivers an **audio cue** such as:

- “Turn left”
- “Turn right”
- “Go forward”

- “Turn around”

These cues are updated once per GPS/compass read cycle (every 3 seconds), or immediately if significant heading deviation is detected. If a GPS signal is lost or invalid, the system pauses navigation and plays a warning tone or repeated buzz pattern until signal is restored.

Once the user comes within **5 meters** of the destination, the system enters the ARRIVED state and signals arrival with a **long triple vibration pattern**. The device then resets to the IDLE state, ready for the next trip.

6. Engineering Standards

Throughout the process of making USC Navigar, our team ensured to follow relevant safety and compliance standards. With our product, we promise to ensure a reliable product and a product that does not harm the user.

7a. Implemented Standards

IEC 61508 (Functional Safety):

Because our system offers guidance to users in a real-world environment, our product must ensure all prompts are accurate. Any pitfalls in this accuracy could cause our product to give incorrect information, or could extend as far as harming the user. As highlighted in the problems that arose in our implementation, our Compass and GPS module could not access readings at times. To stay true to IEC standards, we only provided audible advice when both sensors were able to capture a reading.

IEC 61140 (Protection Against Electric Shock):

Our technology requires voltage to operate. Despite this voltage being relatively low, any wearable device with close contact with the body could cause an electrical shock. To combat this,

our team developed the encasing for the appliance. Only exposing necessary components to the user, and components which would not electrocute the user.

Due to lack of time, we were not able to investigate some other relevant IEC codes. For future work, we would like to focus on the following codes:

IEC 60529 (IP Ratings or Ingress Protection Ratings):

Our hope for USC Navigar would be to be reliable in any sort of conditions when on campus, whether it is rainy or windy, or even factors like construction. This code focuses on testing our device in these different conditions. By gathering correct IP Rating, we would be able to advise our users of the ability to withstand protections against solids and liquids. Or better yet, implement features to provide the highest IP Rating possible. A step surely in the right direction would be to condense our circuitry, provide a more protective shell, and demo our product in said environments.

7b. Future Standards to Implement

IEC 60664 (Insulation Coordination):

Almost similar to our previous safety code regarding the environment, we should also consider factors such as sweat from the user or moisture in the air. This code also takes into account the circuitry and the chances of things like shorting and arcing from outside conductors. As Texas Instruments shares in their breakdown of this code in a motor drive application, factors like Creepage (Shortest Distances between conductive materials), Insulation Type (Whether the product has Basic, Double, or Reinforced Insulation), and Pollution Degree (Depends on Micro-Environment– in this case our product would be exposed to PD4) are all important when understanding out the limitations of said electronic and again, labeling where and when our device can or cannot be relied on.

IEC 60601 (Medical Electrical Equipment):

Although our design is not sufficient enough to allow safe and reliable transportation for those who are blind, our hopes would be to improve our product. To market our product to be a reliable method of transportation, we must mitigate all risks that would result in any errors, thus leading us to comply with pieces of IEC 60601 (Usability and Risk Management). When implementing this product, this IEC code also requires our device to not interfere with any devices that our users may have such as pacemakers, hearing aids, etc. We would need to dive into what factors or parts of our system could cause such scenarios. This code also requires us to investigate external sources that could potentially disrupt our product.

7. Testing and Results

When it came to the production of our product, each core module was tested individually to verify compliance. Afterwards, modules were all integrated together. In our tests, we discovered the following for our modules.

8a. Module Testing

GPS Module: Proved to have extremely accurate results. Datasheet mentioned that results could have ~10 meters of error, but as long as the GPS was out in the open, it proved pinpoint accuracy.

Compass Sensor: Provided real-time directional readings but with a catch, readings gave four distinct general directions, ie. 45°, 135°, 225°, 315°. Proved to not be an issue with the simple four-direction (or less) intersections at USC, but could be an issue if the desired function was to be more precise or more than four pathways at an intersection.

Button Input: Successfully detected the users input with consideration of debouncing, worked reliably

Buzzer Output: Audio cues triggered as expected and were audible during peak campus activity.

8b. Navigation Accuracy and Route Testing

To ensure our navigation system performed as expected, we tested on three distinct routes at several different starting points. In all test cases, once the Compass and GPS were able to get a reading, the printed out path to reach the selected destination was correct. The selected duration between each reading was defined to be three seconds, this of course can be modified to update users more frequently. During all of our tests, no nodes were observed to be skipped and no incorrect paths were given.

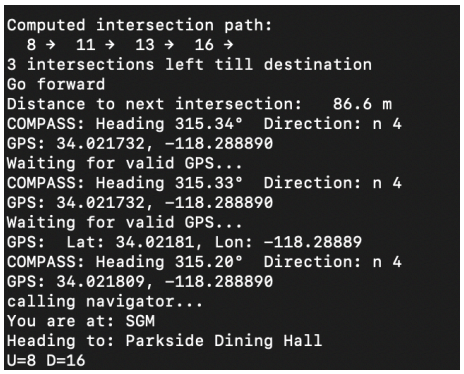


Figure 4: Debug Console

Note: This information is not to be displayed to the user, but to verify the navigation accuracy.

8c. Environmental Testing

While we conducted little environmental testing, when it came to capturing data from the GPS, it did experience issues of gathering readings when the sky was filled with clouds. The GPS module would also struggle to capture a reading when it was underneath any dense roofing, or in an area that contained relatively tall buildings. An example of this being in an area between Ronald Tutor Hall and Oden Hall of Engineering.

8d. User Feedback and Use Testing

Conducted a series of blind tests, where our user would simply follow the audible cues, and not knowing where they were selected to navigate to. Although tests only consisted of four unique participants, the audio cues allowed for our users to navigate in the desired path, as well as correct the direction the user should face if they somehow got off track. The button functionality was proved to be intuitive after specifying how to use it and how many times a user would need to press it in order to select desired location.

8e. Known Limitations

Despite our feats, we have several known issues:

- ➢ Calibration of GPS takes several seconds to begin reading useful results (roughly thirty seconds upon boot up).
- ➢ GPS not being able to read in certain locations on campus.
- ➢ Limited Compass degree precision.
- ➢ Compass itself must be positioned almost perfectly parallel to the ground to be able to receive reading from the module.
- ➢ No current method to tell the user that either the Compass or GPS module (or both) are unable to retrieve a reading.
- ➢ No implemented portable power supply, currently requires direct hookup to a laptop.
- ➢ No full implementation of shell casing created for the project
- ➢ Limited memory

8. Reflections and Lessons Learned

Through our trial and error, our team learned and gained experience in the following.

9a. What Worked Well

Breaking down our modules allowed for easier debugging and navigational cues were intuitive to follow once explained to the user. In the end, we were able to construct a fully functional navigation system that got users from point A to B.

9b. Issues Faced

A majority of our issues stemmed from the debugging of modules themselves. One of the major roadblocks dealt with an off the shelf navigation module that was a knock-off sensor. Highlighting the importance of verifying sourced modules and caution when selecting sensors from scalpers. Another grand issue was not being able to get certain modules to work properly. The acceleration module caused our team to experience issues with communication, despite using correct I2C Oscilloscope debugging processes. Also dealing with a MP3 module that proved to be unreliable as time progressed, causing issues like the module itself to overheat despite proper setup.

9c. Key Lessons Learned

Ultimately our team learned the importance of testing early, often, and in-depth analysis of presented readings. Early tests of modules could have allowed us to pivot from our original GPS, Compass, and MP3 modules to select more reliable modules.

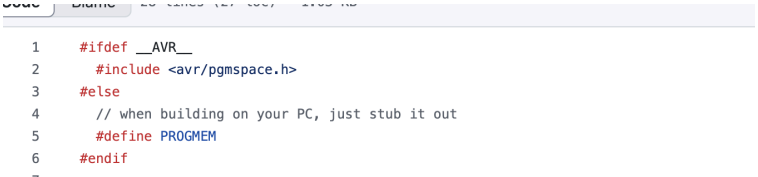
When ensuring each module by itself can work, how a module is setup can cause them to conflict with another module. The original button function which was originally implemented as a blocking function, when a non-blocking function was needed to prevent data interruption.

Importance of viewing beyond technical function and considering usability. Such as the idea that a well-built system that works does not mean it is user-friendly.

10. Conclusion

Ultimately, our team managed to gain experience in technical problem-solving aspects, ranging from process management to user experience considerations. Although the system met its basic function goals, we understand the lack of dependability, clear user feedback, and risk management aspects that could be improved. We would like to emphasize that any of the issues we encountered could have all been avoided through proactive testing. As we look ahead, our goal would be to refine USC Navegar to be more user-friendly, robust, and become an accessible navigation system. For anyone interested in taking the next step for a much needed navigational system, we emphasize the following improvements:

1. Add Fail-Safe Feedback for Sensor Errors
2. Portable Power Supply Integration
3. Speech Recognition module
4. Integrated speaker module that provides audible feedback
5. Improve GPS Precision
6. Upgrade Compass Module
7. Full Encasing and Weatherproofing
8. Gather Feedback from Visually Impaired Users



```
#include "anchors.h"

const Anchor anchors[] PROGMEM = {
    //-5 get index, ie. Parkside is 7 - 5 = 2
    {34.01871626, -118.2884699, "Iovine and Young Hall (IYH)", 0},
    {34.02024813, -118.2897299, "Main RTH Entrance", 0},
    {34.01903279, -118.290926, "Parkside Dining Hall", 0},
    {34.01984521, -118.2900357, "Hughes Aircraft EEB", 0},
    {34.01955069, -118.2894503, "SAL", 0},
    {34.01964845, -118.2887903, "Science & Engineering Library", 0},
    {34.02014991, -118.2898977, "Ronald Tutor Hall", 0},
    {34.01874962, -118.2852301, "Hoffman Hall (HOH)", 1},
    {34.01923924, -118.2855424, "Leventhal School of Accounting (ACC)", 1},
    {34.01887906, -118.2857416, "Bridge Hall (BRI)", 1},
    {34.01931549, -118.2863466, "Zumberge Hall (ZHS)", 1},
    {34.02031526, -118.2852543, "Tommy Trojan Intersection", 1},
    {34.01924457, -118.2878716, "USC School of Architecture", 1},
    {34.01926827, -118.2878938, "USC Roski School of Fine Arts", 1},
    {34.02013848, -118.2874129, "Rapp Engineering (RRB)", 1},
    {34.02020558, -118.2837041, "Doheny Memorial Library", 2},
    {34.01961854, -118.2828424, "Widney Alumni House", 2},
    {34.01870409, -118.2842854, "USC Gould School of Law", 2},
    {34.01882314, -118.2823525, "Fertitta Hall (JFF)", 2},
    {34.01875175, -118.2829885, "Popovich Hall (JKP)", 2},
    {34.02073234, -118.2895212, "Viterbi School Plaza", 3},
    {34.02087880, -118.28969904, "OHE",3},
    {34.02253997, -118.2909059, "Kaprielian Hall (KAP)", 3},
```

Parts List:

Component	Estimated Cost
ATmega328P Microcontroller	$ 3.00
PM1010D GPS Module	$ 30.00
PMOD CMPS2 Compass Module	$ 8.00
Button	$ 2.00
Buzzer	$ 3.00
3D Printed Eyeglass Frame + Extra Material	$ 55.00
Wires, Connectors, PCB Board	$ 5.00

Total Cost: $106

Signature Sheet/Work Allocation:

Michael Castro:

Signature: _____

Riley Milligan:

Signature: _____

Way Zheng:

Signature: _____

Team Member(s)	Hardware Design	Software Design
Michael Castro	35%	30%
Riley Milligan	35%	30%
Way Zheng	30%	40%