

NAVIGATION VEST SUITE FOR BLIND INDIVIDUALS

By

Jiwoong Jung

Haoming Mei

Pump Vanichjakvong

Final Report for ECE 445, Senior Design, Spring 2025

Professor: Cunjiang Yu

TA: Rishik Sathua

05 December 2025

Project No. 5

Abstract

The Navigation Vest Suite is a wearable system that supports visually impaired users with real-time obstacle and navigation guidance in both indoor and outdoor environments. This device integrates microcontroller-based sensing modules with ranging sensors, which provides precise obstacle-distance awareness through haptic motors. Sensory output streams GPS and button data to Raspberry Pi 5 to provide object detection, navigation, and semantic context of the environment in tandem with the haptic feedback. In evaluation, our system demonstrated more than 80% accuracy in interpreting spoken destinations, reliable obstacle detection within a 2-meter 180° field of view, and 95% successful detection across ten object classes. The average end-to-end haptic latency was 2 seconds. The power subsystem demonstrated feasibility for real-world navigation assistance by supporting at least 2 hours of battery power on full system utilization, with rechargeable battery capabilities.

Table of Contents

1	Introduction	1
1.1	Problem	1
1.2	Solution.....	1
1.3	High Level Diagram and Block Diagrams.....	2
1.4	Mechanical Design.....	3
2	Design	4
2.1	Design Changes	4
2.1.1	Electrical	4
2.1.2	Software	4
2.2	Submodule Design.....	5
2.2.1	Power Subsystem Design.....	5
2.2.2	Sensing Subsystem Design	6
2.2.3	Sensory Output Subsystem Design.....	7
2.2.4	Navigation Subsystem Design	9
2.2.5	Computer Vision Subsystem Design.....	10
2.2.6	Audio Subsystem	12
3	Design Verification	13
3.1	Power Subsystem Verification.....	13
3.2	Sensing Subsystem Verification	14
3.3	Sensory Output Subsystem Verification	15
3.3.1	Time of Flight Sensor.....	15
3.3.2	Vibration Motors	16
3.4	Navigation Subsystem Verification.....	16
3.4.1	Computer Vision Subsystem Verification	16
3.4.2	Audio	16
4	Costs	17
4.1	Parts.....	17
4.2	Labor	17
4.3	Total Cost.....	17
4.4	Schedule	17

5	Conclusion.....	18
5.1	Accomplishments.....	18
5.2	Uncertainties.....	18
5.3	Ethical considerations	18
5.4	Future work.....	18
6	References.....	20
Appendix A	Requirements and Verification	22
Appendix B	Mechanical References	25
Appendix B.1	Full Vest Visual Diagram.....	25
Appendix C	Electrical References	26
Appendix C.1	PCB Schematics.....	26
Appendix C.2	PCB Layout and Annotations.....	30
Appendix C.3	Electrical Diagrams and Equations	31
Appendix C.4	Electrical Cost Analysis	32
Appendix D	Software References	39
Appendix D.1	Navigation functions	39
Appendix D.2	ONNX Quantization.....	39
Appendix D.3	RPI Training Loop	40
Appendix D.4	Button Debouncing Logic	40
Appendix D.5	Gemini/GPT Prompt Instructions.....	41
	[LLM Prompt Instructions].....	41
	[LLM Response]	42
Appendix D.6	Audio Subsystems Text to Speech and Speech to Text.....	42
	[Text to Speech]	42
	[Speech to Text]	42
	[Audio Verification Test Program].....	43
Appendix D.7	Raspberry Pi Master Program Flowchart.....	45
Appendix D.8	ToF Program Flowchart	46
Appendix E	Schedule	47

1 Introduction

1.1 Problem

People with Eye Disability often face significant challenges navigating around in their daily lives. Currently, most available solutions range from white canes and guide dogs to AI-powered smart glasses, many of which are difficult to use and can cost as much as \$3,000. A participant emphasized the financial barrier, stating, "These solutions are too expensive, I don't know anyone who can afford these without support from PFRON" [1].

Multiple resources show the urgent need for a navigation system suite for people with blindness. According to a study published in the National Library of Medicine, at least 2.2 billion individuals worldwide suffer from vision impairment, and there is an urgent need for efficient assistance due to the rapid rise in their population [2]. Additionally, research highlights that visual impairment (VI) significantly reduces quality of life ... due to poor product design and limited adaptation [3].

Thus, based on the above sources, we realized that there is an urgent need for us Electrical and Computer Engineers to build an innovative and intuitive vest navigation suite for people with blind disabilities.

1.2 Solution

Our solution is to create a Navigation Vest Suite that aids with people with visual impairments. It provides step-by-step routing through GPS and magnetometer and delivers guidance through haptic and audio cues. Users can interact with the system through voice commands and find out where the vest is by querying the destination through speech. While in motion, onboard sensors and object detection models will identify obstacles and people, announcing their presence and the distance through a speaker. Users will also receive direction of obstacles along with the turning angle required to avoid/acquire them through vibration motors and the speaker.

Haptic Action	Motion
Zero Haptic Feedback	No Obstacles Around
Front Haptic	Obstacle Present Infront
Right Haptic	Obstacle Present Right Side
Left Haptic	Obstacle Present Left Side

Fig 1: Haptic Logic Table

Based on the above table, motors can work together in conjunction to represent multi direction feedback as well. For instance, if both the front haptic and right haptic are activated, that means that the obstacles are present in both the front and the right side of the user. Additionally based on the proximity of the haptic vibration, the user will know how close the objects are. Higher the frequency, the obstacle is closer in proximity, lower the frequency, the obstacle is further in proximity. For now, we are planning on to create equation in the form of $\text{Motor Haptic Frequency} = 1/\text{Distance to the Object}$

equation will be tweaked later to compensate for each variable domain where: Distance to the object $\in [2\text{cm}, 400\text{cm}]$, Motor Haptic Frequency $\in [141\text{Hz}, 200\text{Hz}]$. The final effect is that whenever the user is closer to the object, the motor vibration will exponentially increase, whereas when the object is far away from the user, the vibration frequency will slowly decrease.

The main reason we chose to house our components within a vest over glasses comes down to comfort. When utilizing vibration motors, wearers might feel some discomfort due to long usage, given that they are using the device for over longer periods of time. By utilizing a vest, not only are we able to house more components onto our device, but we are also able to enable users to wear the device more comfortably.

1.3 High Level Diagram and Block Diagrams

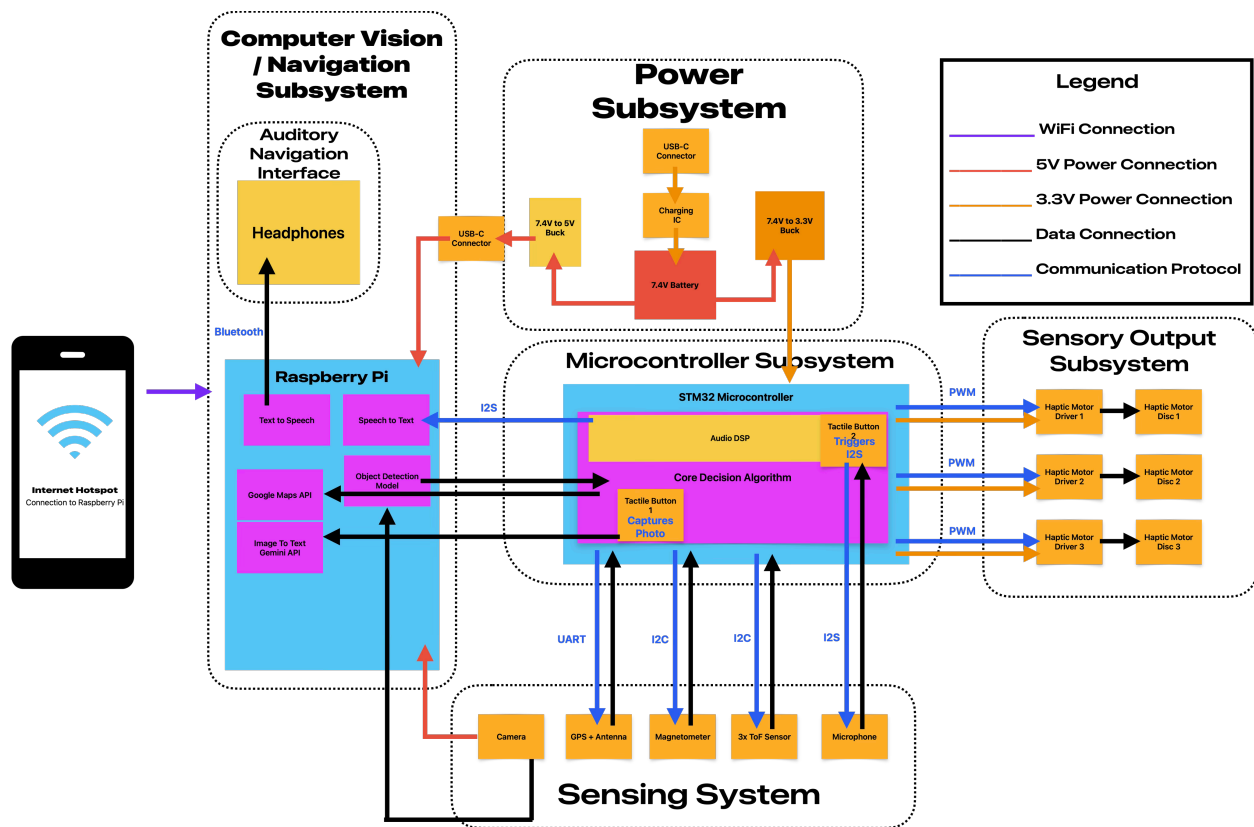


Fig 2 Block Diagram

The power subsystem receives power from a battery pack and regulates 5V and 3.3V through two buck converters. The 5V rail powers the Raspberry Pi for the navigation subsystem, while the 3.3V rail powers the microcontroller, sensors, and haptic motors necessary for obstacle avoidance. Subsystem also includes a charging circuit as well as a buck-disabling circuit that automatically disables the 3.3V and 5V buck when the device is connected to USB-C charging.

The microcontroller handles all communication with sensors, including Time-of-Flight, magnetometer, and GPS module. It processes the incoming data and generates the corresponding haptic feedback signals for obstacle avoidance and transmits real-time heading and GPS coordinates to the Raspberry PI for navigation guidance.

The sensory output subsystem consists of three Time-of-Flight (ToF) sensors, motor drivers, and haptic motors. The ToF sensors measure the distance to obstacle and nearby objects, and the microcontroller computes the PWM intensity for the motor drivers, which drives the haptic motors that warn the user of approaching obstacles.

The sensing subsystem includes the GPS and magnetometer. The microcontroller will send position and heading angle collected to the Raspberry PI 5. The Pi uses this information, together with the Google Navigation APIs to compute real time path for the user.

The navigation subsystem consists of a Raspberry Pi 5 running the software block responsible for high level perception and navigation. The subsystem uses a PI camera for object detection and supports an environmental scan triggered by user-input button. The also connects to a Bluetooth headphone, relaying environment descriptions and critical objects detected via text-to-speech. The user also can trigger the microphone using a separate button on the other hand, which activates the voice input, enabling speech-to-text to capture destination entries. The subsystem performs sensor fusion using the camera for detection, Google API for navigation, and real-time GPS and magnetometer data to generate audible navigation instructions as the user moves towards the destination.

1.4 Mechanical Design

Our design consists of a main enclosure, a Raspberry PI enclosure, and 3 ToF + Motor small enclosures (see Appendix B.1). The main PCB and the Raspberry PI enclosures sits at the chest level and is velcroid to the vest. The main PCB has routing to the ToF + motor enclosures, which are located at each shoulder blade and below the enclosures for the main PCB and PI.

2 Design

2.1 Design Changes

2.1.1 Electrical

During the developmental phase, the microcontroller was reelected to address limitation in PCB routing as well as insufficient amount of GPIO and UART pins for debugging, programming, as well as communicating with the Raspberry Pi. The newly chosen microcontroller, STM32H743VIT6 (Fig 3(a)), fulfill the requirement the previous microcontroller (Fig 3(b)) lacked and provided a much more flexibility space for PCB routing since unused pins gave more space for PCB routing.

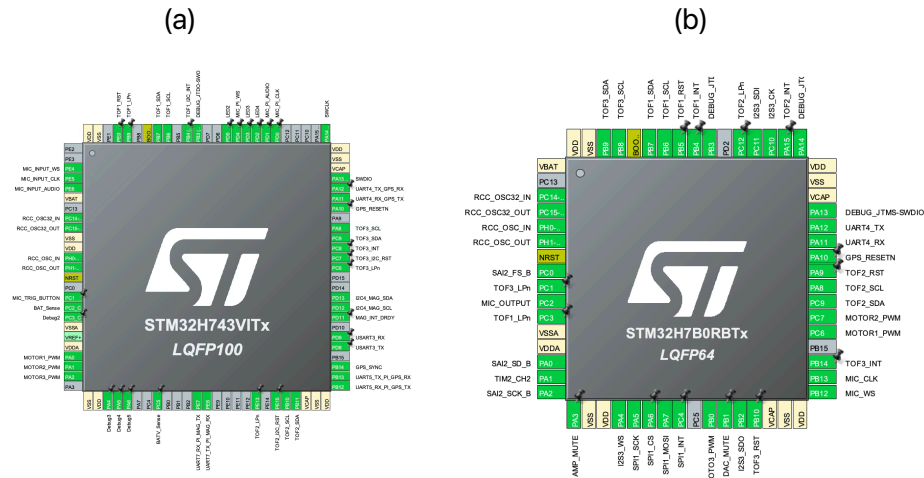


Fig 3: Microcontroller Reselection

Another significant challenge we encountered was debugging and ultimately replacing the original 5V buck converter, the MAX20410AF0C/VY+ [20]. This selected buck converter was initially chosen for its capability to supply nearly twice the required power for 5V system, which creates a lot of headroom at a reasonable price. However, the implemented circuit (Fig C.1.8) consistently failed despite extensive debugging efforts including cross-referencing with a verified breakout design, performing voltage, continuity, and resistance test. Because 5V is essential to run the software stack on the Raspberry Pi, a redesign was required for the third PCB revision. The new design we chosen utilizes the TPS563207DRLR [8], a non-QFN chip, and it successfully powered the system with minimal issues. It was also rated for approximately 5 W above our 5V rail requirements, providing sufficient headroom margin while simplifying the soldering and debugging process.

2.1.2 Software

There were two large tweaks we had to implement in the software. First is supporting multi-TOF. Multiple underlying API and TOF functions had to be amended to take variable objects inputs to as three independent TOF sensors must be supported. Refer to Appendix D.8 for more details. Additional change that we have done in the Raspberry Pi's master program is to include the logic for querying the Gemini Api. Gemini Api was our last add on to our subsystem. Please refer to Appendix D.7 for more detail.

2.2 Submodule Design

2.2.1 Power Subsystem Design

The power subsystem is responsible for supplying and recharging all electrical energy required by the microcontroller, sensing modules, navigation unit, and haptic feedback motors. The system is powered by two 21700 3.7V lithium-ion cells connected in series, providing a nominal 7.4V supply with 5000 mAh capacity [4]. Although vendor did not specify a discharge rating, commercial 21700 lithium-ion cells of similar capacity typically support 8-15 A of continuous discharge, such as the NCR21700A and Samsung INR21700-50E [5], [6]. This range provides sufficient headroom for peak computation loads.

Power Rail	Power Requirement	Converter Capability	Efficiency	Converter Power Output Adjusted
5V	25W	30W	~90% [7]	27W
3.3V	8W	10W	~89% [8]	8.9W

Table 1: Power Requirement vs Converter Capability Table

To power the required power rails, the power subsystem included two high-efficiency buck converters [7], and [8]. The 5V converter circuit (Fig C.1.1) supplies up to 27W (Table 1) for the Raspberry Pi, while the 3.3V converter circuit (Fig C.1.2) provides up to 8.9W (Table 1). These specifications both fit the requirements demanded by both power rails as shown in Table 1. PCB design also included large copper zone for the 3.3V (Fig C.2.1) on the front and a full ground copper zone (Fig C.2.2) on the back, which is used to reduce overall temperature during operation. Additionally, custom-shaped copper zone is created for high localized temperature areas for both 5V (Fig C.2.3) and the battery power traces (Fig C.2.2), and traces width are also increased to above 40 – 50 mil (Fig C.2.2 and Fig C.2.3) to accommodate for high current demand and improve dissipation.

$$I_{ILIM} = \frac{640 \text{ (k}\Omega\text{)}}{3 \times R_{ILIM}} \text{ (A)}$$

Equation 1: USB-C Current Input Limit

The power subsystem also supports USB-C charging (Fig C.1.5) rated for 5V and at up to 3A. The MP2639C charger IC [10] is configured with an input current limit of roughly 2.98A using a 71.5 k Ω ILIM resistor (Equation 1), which prevents overstressing the USB-C adapter to beyond the current rating of 3 A [11]. Cell balancing is achieved through charger's MID sensing input (Fig 4), which monitors the midpoint of the two-cell pack and equalizes series cells via its internal balancing MOSFETS.

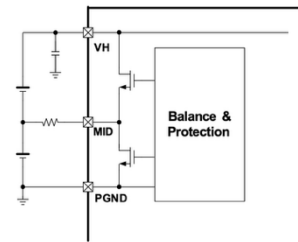


Fig 4: Cell Junction to MID Circuit
Reproduced from [10].

$$t_{\text{TRICKLE_TMR}} = \frac{0.1 C_{\text{bat}}}{I_{\text{TC}}}$$

Equation 2: Trickle Time. Derived from Fig C.3.2

MP2639C Charger circuitry is also implemented with a controlled-timing mechanism using an external trickle capacitor (Fig C.3.1). When the battery is below the pre-charge threshold, 5.9V, the charger enters the trickle-charge mode by injecting a reduced charge current that is roughly ~15% of the fast-charge current at 300mA [10]. The external capacitor defines the maximize allowable charge time to of this pre-charge phase [12] using Equation 2.

$$\tau_{TRICKLE_TMR} = 33.7 \text{ min} \times \frac{C_{TMR}(\mu F)}{0.1 \mu F}$$

Equation 3: Trickle-Capacitor Equation. Reproduced from [10].

Assuming that the pre-charge duration is limited to 10% of the Li-ion battery's total capacity. We use $C_{bat} = 5000 \text{ mAh}$ [4] and $I_{TC} = 300 \text{ mA}$ to calculate a maximize trickle time of roughly 100 minutes. Using Equation 3, we choose $C_T = 0.30 \mu F$, which gives us a maximal trickle-charge time of 101 minutes. Next, we want to ensure temperature safety during charge. We implemented a temperature-based control circuit using an NTC network as shown in Fig C.1.3, which works with the MP2639C (Fig C.3.3) to reduce the charge amount based on the temperature of the battery. This is done by inserting a thyristor inside of the battery, which monitors and ensures charge only when the temperature is within a safe operating range between 0 °C and 60 °C [10].

Lastly, we want to ensure deterministic charging behavior, a buck-disabling circuit (Fig C.1.4) is designed to drive the enable pins of the 5V and 3.3V buck converters low whenever a USB-C input is detected. This ensures that the no subsystem draws current during charge, allowing the battery to fully charge before usage.

2.2.2 Sensing Subsystem Design

The sensing subsystem is responsible for providing geolocation and heading angle to the Raspberry Pi, which is used for navigation guidance. The subsystem consists of L76L-M33 GPS module [14] and a I12MDC 3-axis magnetometer [15], both of which interfaces with the STMH7 microcontroller for preprocessing before data is streamed to the Raspberry Pi 5 for navigation guidance.

L76L GPS Chip Active Antenna Supported Band	Band Frequency Range [14]	GPS-14986 Compatability [16]
GPS L1 C/A	1575.42MHz	✓
GLONASS L1	~1598 – 1609 MHz	✓
Galileo E1	1575.42MHz	✓
BDS B1I	~1561 MHz	Partial
QZSS L1 C/A	1575.42MHz	✓
SBAS L1	1575.42 MHz	✓

Table 2: GPS Requirements vs Antenna Specification

To acquire geolocation data, the subsystem requires an GNSS antenna along with a 50Ω impedance input and antenna matching circuit that allows received satellite signals to maintain signal integrity. We chose to use the GPS-14986 [16] active antenna over passive antenna because its integrated low-noise amplifier (LNA) improves sensitivity, compensated for cable loss, and provided wider frequency range and has better precision. This antenna offers an operating range from 1575 – 1610 and 28dB

total gain. Although the total gain is above the recommendation amount of 18dB, the gain is partially reduced across the 3 m cable and through the decoupling components, and the residual gain is supported by the antenna circuitry.

As shown in Table 2, the L76-L recommends RF input band of 1559-1609MHz to cover all global L7 GNSS signals. The GPS-14986 antenna fully covers these L1 frequencies except BeiDou B1I at 1561MHz. However, it is not required for our application because the combination of existing band support (GPS L1, GLONASS L1, and Galileo) already provided sufficient satellite visibility and positioning accuracy for navigation.

For heading angle, the subsystem mounted the IIS2MDC 3-axis magnetometer's horizontal (XZ) plane parallel to user at an upright position. The magnetometer also had to be calibrated to adjust to soft-ion, hard-ion, and declination interference. Soft-ion and hard-ion describes magnetic field emitting object and wrap-around object that interferences with the magnetometer's magnetic field detection, and declination is the heading offset due to angular difference between magnetic and geographic north. These caused a distorted ellipsoid and an uncentered sphere, and a calibrated magnetometer would measure magnetic field and produce a sphere centered at the origin.

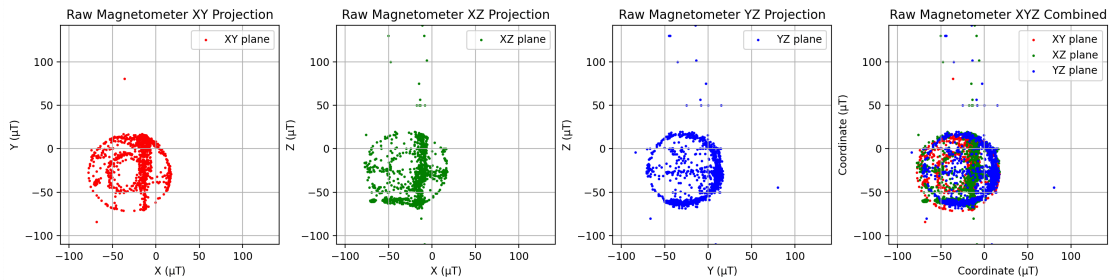


Fig 5: Raw Magnetometer Data

To correct these distortions, calibration process is performed by collecting raw magnetometer readings across all orientation (Fig 5) and effectively compensate for the soft and hard-iron distortions, transforming the ellipsoid into a sphere and centers it at the origin.

2.2.3 Sensory Output Subsystem Design

The sensory output subsystem provides real-time obstacle awareness to the user through a combination of Time-of-Flight (ToF) distance sensing and haptic feedback from toothbrush motors. The subsystem integrates three VL53L7CX multi-zone ToF sensory array (Fig C.1.7), and DRV8210 motor drivers (Fig C.1.7), and vibration motors mounted along with the ToF sensors. The goal of the sensory output subsystem is to convert distance measurements into intuitive haptic cues that assist users in detecting and avoiding nearby obstacles.

The motor-driver circuitry on the main PCB controls external vibration motors connected through dedicated harnesses (Fig C.1.7). Each DRV8210 driver is commanded by PWM signals from the microcontroller, allowing vibration intensity to scale with obstacle proximity.

When designing our Time-of-Flight circuit, we followed the design recommendations given by the documentation (Fig C.1.9). According to the datasheet [9], the I2C protocol required pull-up resistors for the SCL and SDA lines. This is important as the I2C protocol registers signal based on falling and rising edges, and its default level is high.

What we also discovered after designing our PCB through our breadboard prototyping phase was the behavior of the LPn pin. When we attempted implement the LPn as intended through adding a pull-up resistor, it caused the ToF sensor to not operate at all. Through suspicion, we also did test different connection methods such as adding a pull down instead, or entirely disconnecting the pin. What we discovered was the module became functional once we left the LPn pin on floating.

Due to our significant reliance on Time of Flight sensors for special awareness, it is important that our haptic output delay is minimized as much as possible. To optimize our feedback speed, we utilized two strategies which helped cut our delay time down to 50 percent of what we had initially. Our first optimization was increasing our module's clock speed through the STM32 ioc file. This reduced our delay from 4 seconds to 3 seconds. To give the wearer seamless haptic feedback, we designed an equation which closely models the discharge of a capacitor, where you would experience maximum output when you are incredibly close to an object, and gradually less as you move away from the object, having no feedback at distances further than 2 meters away. Through this equation, we were able to model the output graph shown below at Fig 6.

$$PWM[x] = \begin{cases} 60000 & , \text{ if } x < 5 \\ 60000 \cdot \left[e^{\frac{-x}{5000}} \right] & , \text{ if } 5 \leq x \leq 20 \\ 0 & , \text{ otherwise} \end{cases}$$

Fig 6: Distance-PWM Discrete Mapping

However, computing exponential functions does come at a cost of computational time. To optimize our output delay, we went with an approach which utilized Look-up tables instead of real-time computations. This reduced our feedback delay down to approximately 2 seconds. Also see Appendix D.8 for a flowchart view of our ToF program.

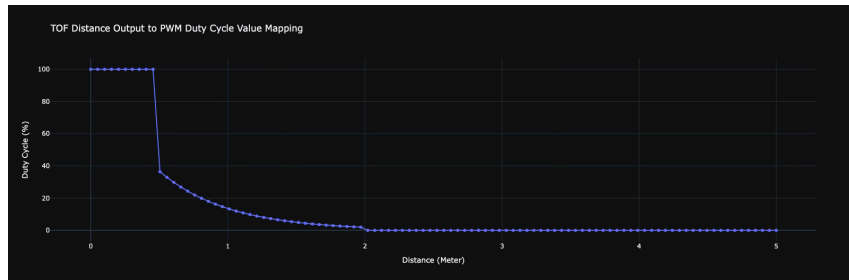


Fig 7 Distance-PWM Graph

2.2.4 Navigation Subsystem Design

The navigation subsystem enables the user to comfortably walk to a destination with ease through real-time navigational instructions. To achieve our high-level requirements, we implemented a navigation subsystem which utilizes two of our main boards: the main PCB, and the Raspberry Pi.

The main PCB is used to provide GPS coordinates and heading direction through a GPS module, and a magnetometer. The following outputs are routed by the Microcontroller through UART to the Raspberry Pi. The GPS module that we bought communicates its output by UART, and through this port, we can determine the location of the user. The GPS sensor that we used (L76-L) transmits the standard NMEA 0183 V4.10 sentences to the MCU [13]. The output of the GPS in one update cycle is shown at Fig 8:

```
$GPRMC,153503.00,A,4007.2345,N,08812.3456,W,0.0,A,0.0,M,0.0,M,0.0,0000.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
$GPGGA,153503.00,4007.2345,N,08812.3456,W,1,08,0.9,190.0,M,-33.9,M,0.0,0000.0,M,0.0,0000.0,0.0
$GPGSA,A,3,02,04,05,12,13,15,24,25,1.8,1.0,1.5*3C
$GPGSV,3,1,09,02,45,090,42,04,33,180,44,05,27,140,41,12,10,045,39*7E
$GPVTG,45.1,T,M,0.13,N,0.2,K,A*3F
```

Fig 8 Sample GPS Data

Our focus for this project is navigation, which requires the best precision. To achieve this, we are only concerned about GPGGA signals. Parsing one sentence will enable us to reduce UART data overhead while retaining information required for our positioning and guidance algorithms.

For magnetometer, we send the calibrated heading relative to the north to determine the direction that the user is looking at, and the data is sent with 2 floating point precision as shown in Fig 9:

```
7.18
1.18
7.92
9.23
13.81
7.52
1.73
9.31
3.61
7.36
6.62
7.46
5.64
7.98
6.23
```

Fig 9: Sample Magnetometer Data

Given the raw data, there are two software components which enables full navigation support: the speech-to-text module to obtain the user's desired destination, and the Google Maps API to obtain navigation instructions. The navigation logic is as follows (Fig 10):

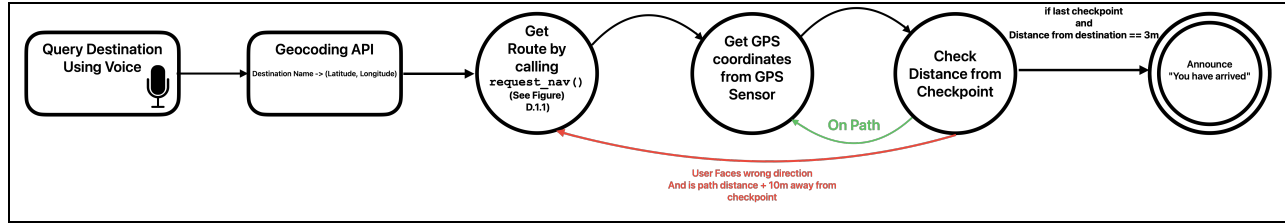


Fig 10: Navigation State Diagram

2.2.5 Computer Vision Subsystem Design

The computer vision subsystem allows the user to be aware of the surrounding important objects. To achieve our high-level requirements, the computer vision subsystem consists of two underlying systems: the YOLO Model and the Gemini API.

The YOLO model is used to provide constant real time updates on general obstacles that are useful for the user to acknowledge. The model consists of 2624080 parameters, and it is processed on-device in Raspberry Pi 5. The baseline model was adopted from the repository: [21]

With the baseline model given, there were two major steps that had to be taken in order for the model to run effectively on Raspberry Pi's CPU. The first issue is that the model forward pass takes longer than 9ms. The duty of the on-device YOLO model is to provide the user with quick updates on their surroundings with cautious objects, in which we believe that 9ms could be a large delay. The second issue is that the provided baseline model does not fulfill the high-level requirement of detecting at least 10 distinct road signs, curbs, and pedestrians with at least 95% successful detection. The method of measurement is to provide samples of 100 un-curated data for each object and check the number of detections. The object with a 95% chance of detection gets a checkmark, and else gets a cross mark. Here were the results post optimization (Table 3):

Person	✓
Car	✓
Laptop	✗
Traffic Light	✓
Bicycle	✗
Motorcycle	✗
Stop Sign	✓
Bridge	✗
Scooter	✓
Cell Phone	✓

Table 3: Object Detection Initial Results

To resolve the first issue, we have used the ONNX quantization for processing. The benefit of ONNX quantization is that it reduces the model size, allowing faster processing speeds. Fundamentally for the model to run, it moves the model to random access memory fully to begin processing. Quantization allows us to reduce the size of each parameter, lower the total amount of memory required, and reduce RAM pressure to improve computational throughput.

The ONNX quantization reduces from FP32 to INT8 in size. That means that per parameter, the size is reduced from 4 bytes down to 1 byte. This is a 4-time reduction in model size, allowing faster weight loading and better utilization of cache and memory bandwidth during inference. This allowed us to reduce the processing time from 9ms to 3.5ms per query. ONNX quantization allows you to preserve accuracy via smart calibration. According to [22], ONNX quantization allows smart calibration on real data for optimal scales, using per-channel quantization, layers selectivity, and avoiding quantizing layers that are highly sensitive to precision loss. ONNX implementation can be done with a simple call to the ONNX library. Please check Appendix D.2 for ONNX quantization python implementation.

After ONNX quantization, we need to train the model to ensure that all the required objects are detectable with at least 95% success rate. To do so, we investigated RPI framework on how to train the YOLO model. For the most effective training loop, we have adopted the method from repository [23], which is the official RPI training package from Ultralytics. Additional benefit of the RPI training package is that it already has the provided YAML training data, which are un-curated to make sure that the model is evaluated under realistic, diverse, and noisy real-world conditions. We will be using both `HomeObjects-3k.yaml` and `coco128.yaml` provided datasets to ensure that our model gets trained on both ample amounts of indoor and outdoor objects. The `HomeObjects-3k.yaml` has around 2,285 training images and 404 validation images. For `coco128.yaml`, there are 128 images for training and 128 for validation in the YAML configuration. Example training loop code can be found under Appendix D.3 for more information.



Fig 11: PI-Yolo Logic

Details about the enhancement results will be discussed in more detail under section 3.5, Computer Vision Subsystem Verification.

Next section of the computer vision subsystem is the Gemini API. One of the most important features in the computer vision model is to provide rich semantic details about their surroundings when they need to. The intuitive design choice we had is to allow the user to press the button, which triggers Raspberry Pi's camera to take a snapshot of the surroundings and query the Gemini LLM API to fetch the detailed responses. After the responses are received, we use the TTS module to broadcast the information to the user.

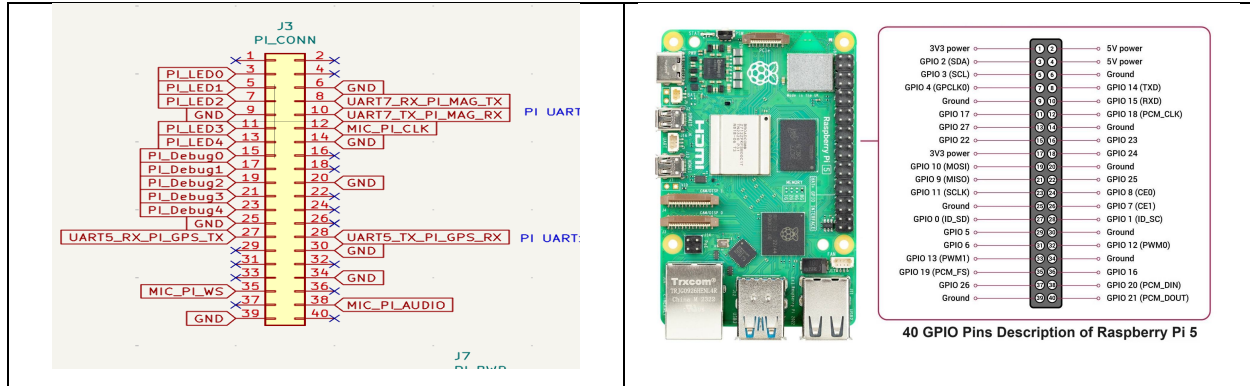


Fig 12: Raspberry Pi Pinout and Schematics

As shown on the above diagram (Fig 12), the PI_Debug pins are open for interacting with the GPIOs. We are dedicating GPIO 22 for the Gemini/GPT button and GPIO 10 for the GPS button. It was important for us to note that the button must be debounced. Debouncing logic can be found in Appendix D.4.

One more design consideration is prompt engineering. Along with the user picture input, we are responsible for providing instructions to the LLM with prompts to provide the user with correct semantic instructions. To check our prompt instructions and the sample output, check Appendix D.5.

2.2.6 Audio Subsystem

For the audio subsystem, we decided to take the python library approach to allow text to speech and speech to text interactions. Text to speech is used in both computer vision and navigation in order to provide detailed semantic information to the user. We will be using pyttsx3 library for text to speech. Implementation can be checked under Appendix D.6, speech to text section.

For the speech to text, we've used Python's sounddevice and speech_recognition libraires. First, sounddevice is used to extract the raw audio file form the hardware by setting up the sample rate, channel, and duration of the recording. Sounddevice library then outputs a raw audio file, which is then passed to the speech recognition library to be processed to text. Speech to text will play an integral part in detecting which destination the user wishes to set to. Specific implementation can be checked under Appendix D.6, speech to text section.

3 Design Verification

3.1 Power Subsystem Verification

As mentioned in Section 2.2.1, the power subsystem requires sufficient power rating to power the system for navigation and obstacle avoidance. This requires the buck converter output for both 3.3V and 5V rail to be within voltage tolerance of 5% for at least the whole range of the battery voltage (6V – 15V). As shown by Fig 13(a) and Fig 13(b), where the red region indicates tolerable voltage output for voltage range from 6V to 15V, and the blue data verifies that our buck converter is able to sufficiently a stable voltage within the tolerance range beyond the battery voltage range. For these graphs, we used the power supply to supply the PCB with voltages in increment of 0.5V and measured the buck converter output for each of the rail using a multimeter.

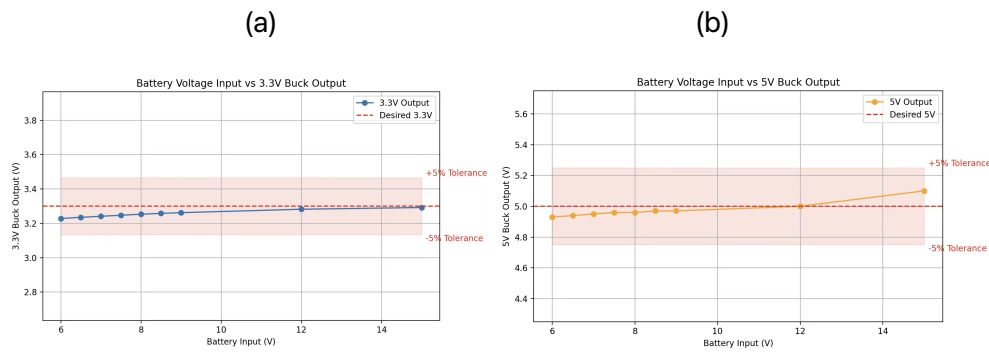


Fig 13: Buck Converters Output

Another requirement for the subsystem is a battery life of at minimum 2 hours assuming the system runs at the typical load at 2.5A, which is equivalent to 18.5W total. To validate this, we measured the battery voltage across the battery terminal every 15 minutes when we were testing the product outdoors. The results (Fig 14) show that the battery does indeed last ~2.2hours. We did not measure the indoor battery life as the validation process will be time consuming since the battery is rated for 37.5 watts-hours and indoor subsystem only consumes 2.5W, and this translate to roughly 16 hours of operation indoors.

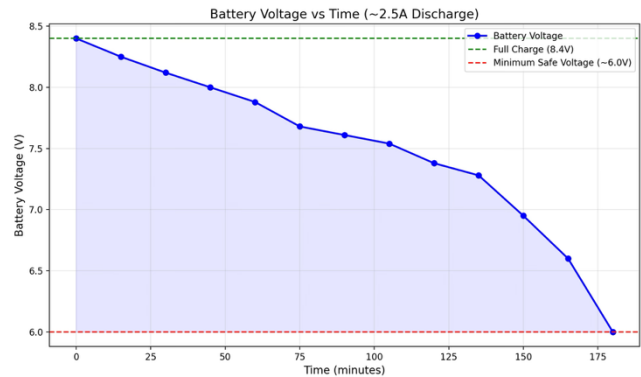


Fig 14: Battery vs Time (Outdoor Usage)

Additionally, we want to ensure that the battery is rechargeable since the user will require the device for both indoor and outdoor environments. We verified this by plugging in a USB-C charger into the PCB and measured the battery voltage on the board every 25 minutes. Experimentally, we found that the battery is rechargeable within roughly 3 hours as shown in Fig 15.

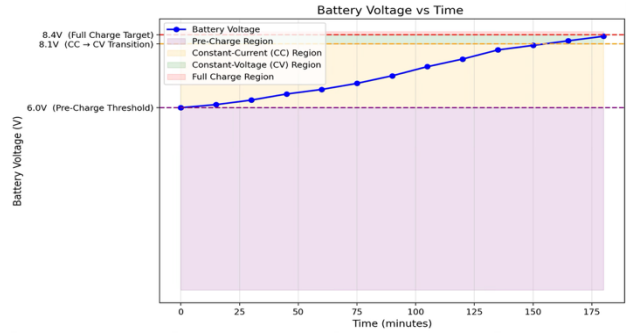


Fig 15: Recharging Graph

Lastly, we want to ensure that the power subsystem turns off during charge to minimize the charge time. As described in Section 2.2.1, we designed a battery-disabling circuit where that drives the enable pins of the buck converters low, effectively turning off the navigation and obstacle avoidance subsystems. We verified the circuit by measuring the voltage at the EN pins and used the power supply as the battery power source to verify that the current draw is 0A by repeatedly plugging and unplugging the USB-C charger for each iteration. We verified the logic specify at Table 4 works to ensure deterministic state behavior between charge and discharge.

USB-C Plugged ?	Current Draw (A)	EN Pin Voltage Level
Yes	Approximately 0A	~0V
No	Returns the Current Draw Prior USB-C Charge	~5V

Table 4: Buck-Disabling Circuit Logic Requirement Table

3.2 Sensing Subsystem Verification

The sensing subsystem, which includes GPS and magnetometer, is used provide geolocation as well as directional data to run the navigation guidance software. This requires precision and accuracy in our on-device data collection to run the navigation guidance software with high reliability. The three most important metrics we look for are position accuracy, number of satellites, as well as the horizontal dilution of precision (HDOP). This is important since high number of satellites increases the number of sampling source, reducing the overall error, and a low HDOP indicates a good geometric spread, which provides more reliable location data [18].

Location	Satellites	HDOP	iPhone Lat	iPhone Lon	Device Lat	Device Lon	Error (m)
House	6.000000	2.380000	40.112627	-88.231818	40.112598	-88.231855	3.211948
ECEB	13.000000	1.350000	40.114472	-88.228775	40.114495	-88.228805	2.632030
CIF	12.000000	1.430000	40.112693	-88.227645	40.112667	-88.227611	2.929056
Talbot	12.000000	1.260000	40.111975	-88.228715	40.111997	-88.228743	2.381016
Beckman	9.000000	1.880000	40.115482	-88.226735	40.115455	-88.226700	2.993058
AVG	10.400000	1.660000					2.829422

Table 5: GPS Verification Table

To ensure the reliability of GPS, we sampled the position data for at least 30 seconds at 5 different locations with clear sky, no user movement, and calculated the error relative to iPhone's position data. According to Table 5, we achieved the desired requirements for the number of satellites, HDOP, as

well a position accuracy within the tolerance range. For magnetometer, we spin in place at an increment of 20 degrees and compared the heading angle readings to the iPhone compass heading angle data, and verified that the error is 0.69%, which is within the requirement at 2% (Table 6).

Reference (°)	Mag Reading (°)	Error (°)	Error (%)
0	0.43	0.43	0.12
20	21.38	1.38	0.38
40	38.81	1.19	0.33
60	63.77	3.77	1.05
80	82.95	2.95	0.82
100	97.84	2.16	0.60
120	117.37	2.63	0.73
140	143.99	3.99	1.11
160	161.56	1.56	0.43
180	180.20	0.20	0.06
200	207.51	7.51	2.09
220	224.29	4.29	1.19
240	238.07	1.93	0.54
260	260.28	0.28	0.08
280	280.67	0.67	0.19
300	299.39	0.61	0.17
320	315.14	4.86	1.35
340	344.49	4.49	1.25
AVG		2.49	0.69

Table 6: Magnetometer Verification Table

3.3 Sensory Output Subsystem Verification

3.3.1 Time of Flight Sensor

To verify the accuracy of our measurements, we took an object of reference (wooden board) to take readings from the Time-of-Flight sensor and used a tape measurement to determine the ground truth (Fig 16). After each measurement, we used a real-time heat map to determine the readings from each grid on the Time-of-Flight array.

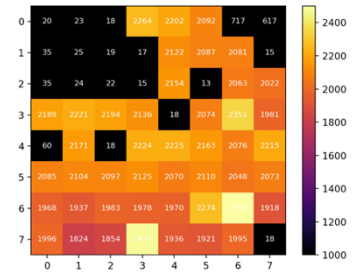


Fig 16: ToF Distance Heatmap.

What we realized was that you could not take the direct average of every single grid to give a single value due to possible errors based on the status codes.

3.3.2 Vibration Motors

To test whether our PWM output is consistent with the distance of the object from the Time-of-Flight, we connected the vibration motor's out pins to the oscilloscope and observed the duty cycle changes as shown below. As we moved closer towards the Time-of-Flight sensor, we would observe a larger width (Fig 17), which corresponds a higher duty cycle. At distances closer than 0.5 meters, our motors are working at 100 percent duty cycle.

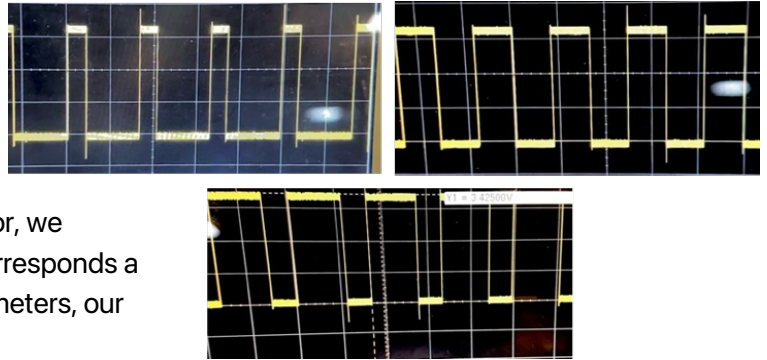


Fig 17: PWM Waveforms

3.4 Navigation Subsystem Verification

3.4.1 Computer Vision Subsystem Verification

First verification for the YOLO Model is to ensure that the detection rate is equal or above 95% of the objects mentioned under section A1.8. Similar to initial testing, we verified the model by forward passing each 100 different mixed un-curated and curated images that contains the required objects. After adjusting the epoch to 200 and learning rate to $1e-5$, the YOLO Model was able to successfully detect the required objects with above 95% success rate. This means that for every 100 forward passes, our trained model was able to successfully detect required objects 95% of the time. Here is the result:

Person	✓
Car	✓
Laptop	✓
Traffic Light	✓
Bicycle	✓
Motorcycle	✓
Stop Sign	✓
Bridge	✓
Scooter	✓
Cell Phone	✓

Table 7: Yolo-Detection

3.4.2 Audio

For audio verification, we made a custom test function that provides accuracy results based on string comparison between the expected list of strings and the audio converted strings. The user has to say list of 100 words based on the instructions, and the test program string compares the microphone output strings with the expected string output. Based on the result, we have achieved 95% success rate in terms of the audio performance. For more information on how the test program is written, refer to Appendix D.6 under Audio Verification Test Programs.

4 Costs

4.1 Parts

Most of our parts are purchased from Digikey and other manufacturers. We purchased as many components from Digikey as possible to save on fees due to shipping. However, some of our components are not available on Digikey and we had to purchase it from other manufacturers. Most of the budget were used for electrical components (see Fig C.4.1 and Fig C.4.2) and costed \$185.75 before taxes and shipping. Depending on the shipment destination the taxes and shipment may vary. Using FedEx/USPS for shipping will cost at least \$6.99 per manufacturer, and we purchased from 3 manufacturer (Digikey, MPS, and Amazon). Adding on the tax (~9%) and tariff gives us a rough total cost of \$250.

4.2 Labor

We will be using the following equation to calculate the labor cost for each team members:

Illinois Min Wage X Actual Hours Spent X 3. Illinois minimum wage is \$15.00 per hour based on [17]. Average time spent per day is 5 hours per person with the consideration that we have 16 weeks for ECE445, which totals to $\$15 \times 5 \times 5 \times 16 \times 3 = \6000 labor cost in total for the team.

Additionally, we utilized the machine shop's service for 3 days. Our machine shop specialist had a salary of \$70000, which is around \$33 per hour. With high bound estimate, the machine shops final labor results to $\$33 \times 4 \times 3 = \396 , regarding their part-time working on our project. In total, we have approximately \$6396 worth of labor cost for our project.

4.3 Total Cost

Total cost is $\$396 + \$6000 + \$250 = \6646

4.4 Schedule

Our schedule throughout the semester is shown in Fig E.1.

5 Conclusion

In conclusion, our project was successful in multiple ways, as we were able to run our system with full functionality with enough battery life. The Time-of-Flight sensors were able to give reliable feedback to the haptic motors to enable users to gauge their distances from obstacles, the YOLO model was able to produce consistently accurate feedback, and the navigation system was able to provide accurate GPS coordinates and provide accurate instructions.

5.1 Accomplishments

The Navigation vest successfully meets our previous requirements. The system successfully interprets voice commands with an accuracy of over 80% while maintaining more than 90% reliable navigation guidance. The sensing subsystem achieves up to 2.0 meters of obstacle detection with a 180° Field of view. The computer vision system can identify key environmental objects such as road signs, pedestrians, and vehicles with 95% detection accuracy. Finally, the integrated power subsystem supports at least two hours of sustained battery life outdoors on a single charge, ensuring the system's feasibility of being used in daily scenarios.

5.2 Uncertainties

Despite significant successes within our project, we did face some unavoidable issues. Due to hardware limitations, our Time-of-Flight and Haptic motors were not able to give the proposed a 200 ms delay in response, despite having utilized multiple optimizations. Since the Time-of-Flight module uses the I2C protocol, communication can be slower. A solution to this problem would be switching our current Time-of-Flight sensors with faster sensors which uses SPI. By utilizing SPI, we would be able to successfully output at a rate which is 30 times faster than our current design.

5.3 Ethical considerations

ACM code of ethics part 1.6 highlights that the responsibility of respecting privacy applies to computing professionals, and computing professionals should take special care for privacy when merging data collections [19]. To satisfy and comply with ACM code standards, for any sensory input data, we will be asking consent to the user. Additionally, we won't be saving user's private locations by default, but such functionality can be enabled under the user's will to activate smart destination mode selection. Lastly, we will never be asking user's personal data such as name, phone number, and home address by default as it is. Not required for the core navigation functionality of the vest and ensures privacy.

5.4 Future work

For future enhancements, we wish to add a 9-axis direction tracking. This allows smoother heading estimation and effectively handles tilts and movement with sensory fusion with accelerometer and gyro. Additional changes we wish to achieve is faster TOF and SPI support. SPI support can provide us with improved latency from ~2000ms down to less than 200ms. Lastly, we wish to work closer with

the machine shop to design smaller encloser. This allows to reduce our product's weight and remove unnecessary space that can make the user uncomfortable.

6 References

- [1] A. Karpov, R. Yusupov, I. Mporas, et al., "Human–Computer Interaction for Users with Disabilities: A Systematic Review of Assistive Robotics," *Journal of Signal Processing Systems*, 2024. doi: <https://doi.org/10.1007/s11265-023-01898-0>
- [2] N. M. Alotaibi, A. S. Alghamdi, M. S. Alzahrani, M. S. Alharbi, M. A. Almutairi, et al., "A smart wearable device for visually impaired persons based on deep and machine learning models," *Journal of Imaging*, vol. 10, no. 4, p. 82, 2024. doi: <https://doi.org/10.3390/jimaging10040082>
- [3] M. A. Chavarria, L. M. Ortiz-Escobar, B. Bacca-Cortes, V. Romero-Cano, I. Villota, et al., "Challenges and opportunities of the human-centered design approach: Case study development of an assistive device for the navigation of persons with visual impairment," *JMIR Rehabilitation and Assistive Technologies*, vol. 12, p. e70694, 2025. doi: <https://doi.org/10.2196/70694>
- [4] RadioMaster RC, "21700 5000mAh Battery for TX16S MKII," *RadioMaster RC*. <https://radiomasterrc.com/products/21700-5000mah-battery-for-tx16s-mkii> [Accessed Sep. 18, 2025]
- [5] Panasonic Industrial Devices "NCR21700A Lithium-Ion Rechargeable Battery Specification Sheet." 2017. Available: <https://www.batteryspace.com/prod-specs/NCR21700A.pdf>
- [6] Samsung SDI "INR21700-50E Lithium-Ion Cell Datasheet." 2018. Available: <https://www.dnkpowers.com/wp-content/uploads/2021/02/Samsung-INR21700-50E-3.6V-5000mAh.pdf>
- [7] Texas Instruments, TPS563207 4.3-V to 17-V Input, 3-A Synchronous Buck Converter in SOT563 datasheet, SLUSE25, Sept. 2020. [Online]. Available: <https://www.ti.com/lit/ds/symlink/tps563207.pdf> [Accessed: Dec. 07, 2025].
- [8] Texas Instruments, TPS56628 4.5-V to 18-V Input, 6-A Synchronous Step-Down Converter with Eco-mode™ Datasheet, SLVSC94A, Oct. 2013 (revised Dec. 2013). [Online]. Available: <https://www.ti.com/lit/ds/symlink/tps56628.pdf> [Accessed: Dec. 07, 2025].
- [9] STMicroelectronics, VL53L7CX — Long-Distance 8×8 Multizone Time-of-Flight Ranging Sensor Datasheet, Rev. 3, Apr. 2023. [Online]. Available: <https://www.st.com/resource/en/datasheet/vl53l7cx.pdf>
- [10] Monolithic Power Systems, MP2639C 2-Cell Li-Ion or Li-Polymer Switching Charger (Datasheet), Monolithic Power Systems, Nov. 22, 2024. [Online]. Available: https://www.monolithicpower.com/en/documentview/productdocument/index/version/2/document_type/Datasheet/lang/en/sku/MP2639C/?srsltid=AfmBOorKCS0Pb2eUfBHbZk6hF_wao8K8ZOALSPoMN87CzPsRW7cXmEBX [Accessed Dec. 8, 2025].
- [11] GCT, "USB4125 Product Specification," USB4125-spec.pdf, Dec. 20, 2024. [Online]. Available: <https://gct.co/files/specs/usb4125-spec.pdf> [Accessed: Dec. 8, 2025].

- [12] "What Are the 3 Stages of Lithium Battery Charging," Grepow. [Online]. Available: https://www.grepow.com/blog/what-are-the-3-stages-of-lithium-battery-charging.html?utm_source=chatgpt.com [Accessed: Dec. 8, 2025].
- [13] Quectel Wireless Solutions Co., Ltd., LC760Z & L76-L & L76-LB Software Difference Introduction, Version 1.0, Mar 2024. [Online]. Available: https://www.quectel.com/content/uploads/2024/05/Quectel_LC760ZL76-LL76-LB_Software_Difference_Introduction_V1.0.pdf
- [14] Quectel L76-L / L76-L-H Hardware Design, Quectel, PDF document. [Online]. Available: https://centerclick.com/ntp/docs/Quectel_L76L76-L_Hardware_Design_V3.3.pdf [Accessed: Dec. 8, 2025].
- [15] ST, IIS2MDC, MEMS digital output magnetic sensor (magnetometer) datasheet, IIS2MDC Datasheet, STMicroelectronics, [Online]. Available: <https://www.st.com/resource/en/datasheet/iis2mdc.pdf> [Accessed: Dec. 8, 2025].
- [16] GPS / GNSS Magnetic-Mount Antenna 3 m SMA, SparkFun. [Online]. Available: <https://www.sparkfun.com/gps-gnss-magnetic-mount-antenna-3m-sma.html#content-features> [Accessed: Dec. 8, 2025].
- [17] Illinois Department of Labor, "Minimum Wage Law," *Illinois.gov*, <https://labor.illinois.gov/laws-rules/fls/minimum-wage-law.html> (accessed Dec. 9, 2025).
- [18] HDOP: Horizontal Dilution of Precision in GPS and DGPS Systems," Lerus, Dec. 6, 2024. [Online]. Available: <https://www.lerus.com/articles/hdop-horizontal-dilution-of-precision-in-gps-and-dgps-systems.html>
- [19] Association for Computing Machinery, "ACM Code of Ethics and Professional Conduct," Section 1.6, "Respect privacy." ACM, adopted June 22, 2018. [Online]. Available: <https://www.acm.org/code-of-ethics>. [Accessed: Sept. 18, 2025].
- [20] Analog Devices, "MAX20408/MAX20410 Fully Integrated Buck Converters Datasheet," rev. 12, Nov. 19, 2024. [Online]. Available: <https://www.digikey.com/en/products/detail/analog-devices-inc-maxim-integrated/MAX20410AFOC-VY/17885186>
- [21] S.X. Dai, "rpi-object-detection," GITHUB repository, online. Available: <https://github.com/automaticdai/rpi-object-detection>. Accessed: Dec 7, 2025
- [22] ONNX Runtime, "Quantize ONNX models," ONNX Runtime Documentation: Model Optimizations — Quantization, [Online]. Available: <https://onnxruntime.ai/docs/performance/model-optimizations/quantization.html>. [Accessed: Dec. 08, 2025].
- [23] Ultralytics, "Train — Usage Examples," *Ultralytics YOLO Docs*, <https://docs.ultralytics.com/modes/train/> (accessed Dec. 8, 2025).

Appendix A Requirements and Verification

All our requirements were verified in Section 3.

Table A.1: Power Subsystem Verifications

Requirement	Verification	Verification status (Y or N)
5V Output within 5% tolerance	<ul style="list-style-type: none"> • Use Lab Power Supply for Battery Power Input (6V-15V) • Measure Voltage Across 5V and GND • Repeat for 6, 6.5, 7, 7.5, 8, 8.5, 9, 9.5, 10, 12, and 15V 	Y
3.3 Output within 5% tolerance	<ul style="list-style-type: none"> • Use Lab Power Supply for Battery Power Input (6V-15V) • Measure Voltage Across 3.3V and GND • Repeat for 6, 6.5, 7, 7.5, 8, 8.5, 9, 9.5, 10, 12, and 15V 	Y
Battery Life at minimum 2 hr	<ul style="list-style-type: none"> • Fully Charge the Battery • Run Full system Outdoor • Measure Voltage Every 15 minutes • Repeat until battery hits 6V 	Y
Recharges within 3 hr	<ul style="list-style-type: none"> • Fully Depleted the Battery by performing the previous verification • Battery Voltage might be 6V • Charge the Battery • Measure the Voltage Between BAT_PWR and GND 	Y
Disables Buck During Charge	<ul style="list-style-type: none"> • Power the System with Lab Power Supply • Prepare any 5V USB-C Charger • Plug USB-C Charger • Ensure EN Pins Pulled Low • Current Draw is nearly 0A • Unplug USB-C Charger • Ensure EN Pins are 5V (High) • Current Draw is nearly 0A 	Y

Table A.2: Senory Output Subsystem Verifications

Requirement	Verification	Verification status (Y or N)

ToF Detection-to-Haptic Latency < 200 ms	<ul style="list-style-type: none"> • Test ran circuit on the corridor • Filmed and recorded blind test run in random location which user does not know of (ECEB Basement) 	N
ToF Supports 170° Field of View	<ul style="list-style-type: none"> • Performed a full radius test • Walked around user at different distances from them • Determine if vibration is present by wearer 	Y
ToF Detection Range up to 2m	<ul style="list-style-type: none"> • Distance test using a large moveable board. • Ran heatmap tests on Python Program • Verified distances of each grid via. Ground truth from tape measurement 	Y
More than 4 Vibration Intensity Level	<ul style="list-style-type: none"> • Connected PWM circuit to oscilloscope • Ran the distance test • Observed varying changes in Duty cycle on the display • Validated output using UART debuggginga 	Y

Table A.3: Sensing Subsystem Verifications

Requirement	Verification	Verification status (Y or N)
GPS Coordinate is within 3m relative to iPhone Data	<ul style="list-style-type: none"> • Visit 5 Locations • Print GPS module output through serial • Compare it against iPhone geolocation data 	Y
GPS Antenna Capable of Communicating with >10 satellite	<ul style="list-style-type: none"> • Check the same data payload from the coordinate verification above 	Y
GPS HDOP < 2.5	<ul style="list-style-type: none"> • Check the same data payload from the coordinate verification above 	Y
Magnetometer Heading Error is within 2% Relative to iPhone Data	<ul style="list-style-type: none"> • Find an open area • Serial print magnetometer data • Compare it with iPhone Compass App data 	Y

Table A.4: Machine Learn Subsystem Verifications

Requirement	Verification	Verification status (Y or N)
Yolo Model can detect required objects with > 95% success rate	<ul style="list-style-type: none"> • Prepare 100 sets of images that includes all the required objects. • Forward pass it to the model, check out the detect status. 	Y

Table A.5: Audio Subsystem Verifications

Requirement	Verification	Verification status (Y or N)
STT can convert audio to text with > 95% accuracy	<ul style="list-style-type: none"> • Prepare list of 100 strings. • Allow the tester to say 100 strings, check if the STT logic output strings. • String compare the expected output and the processed output. 	Y

Appendix B Mechanical References

Appendix B.1 Full Vest Visual Diagram

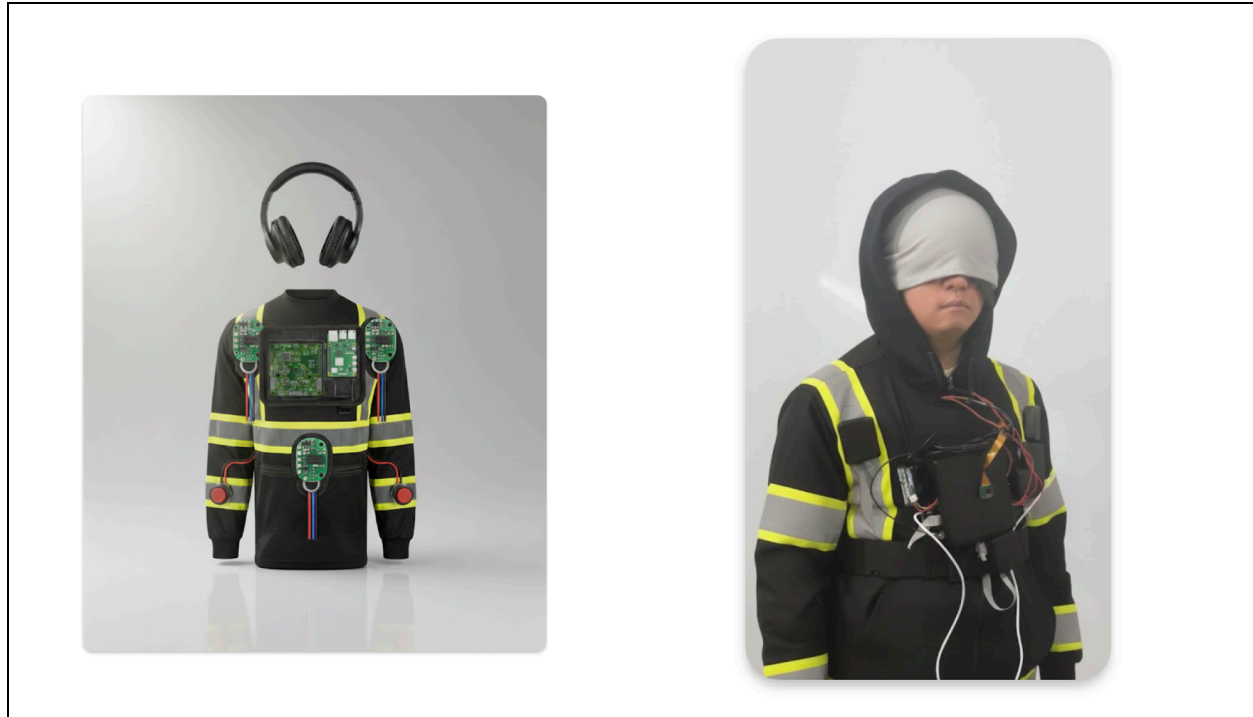


Fig B.1.1: Visual Diagram and Actual Product

Appendix C Electrical References

Appendix C.1 PCB Schematics

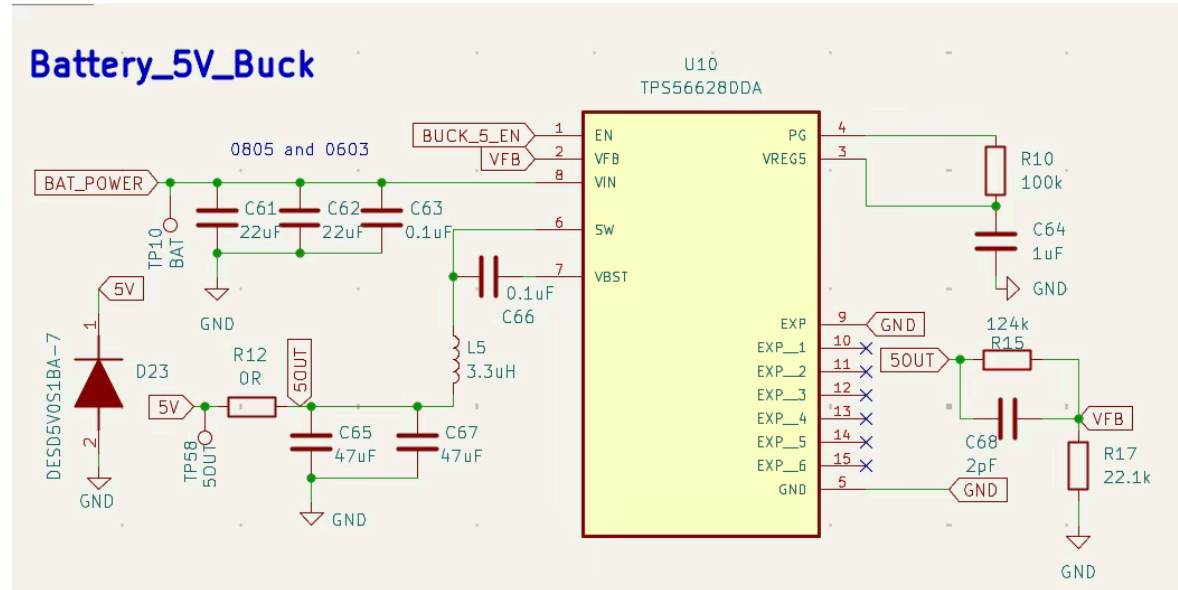


Fig C.1.1: 5V Buck Schematics

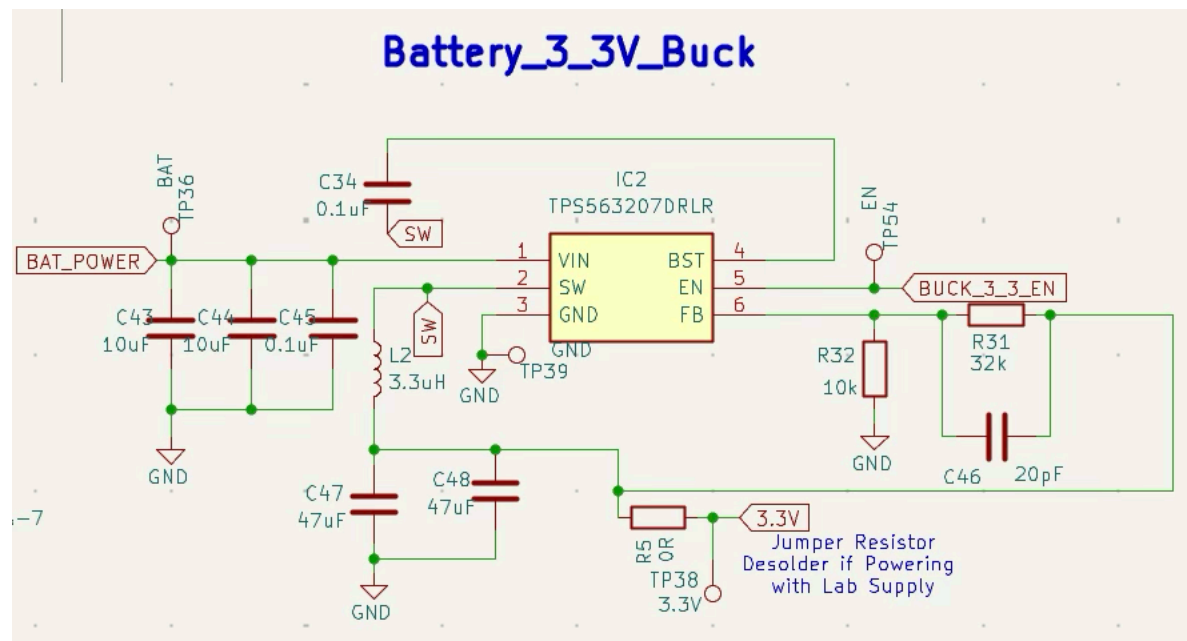


Fig C.1.2: 3.3V Buck Schematics

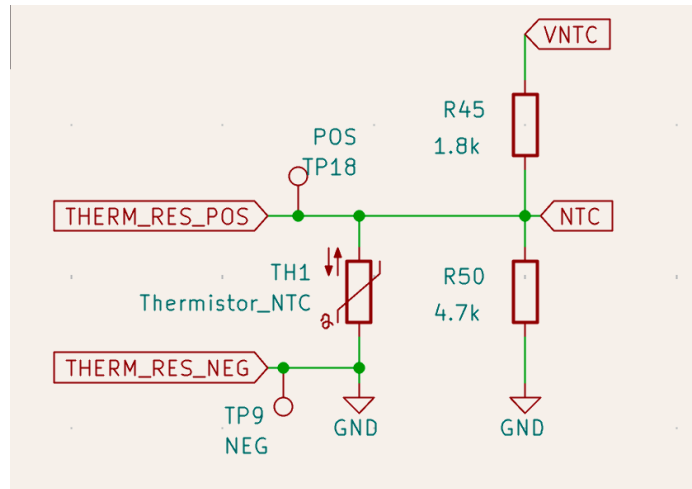


Fig C.1.3: MP2639C Charger Network Schematics

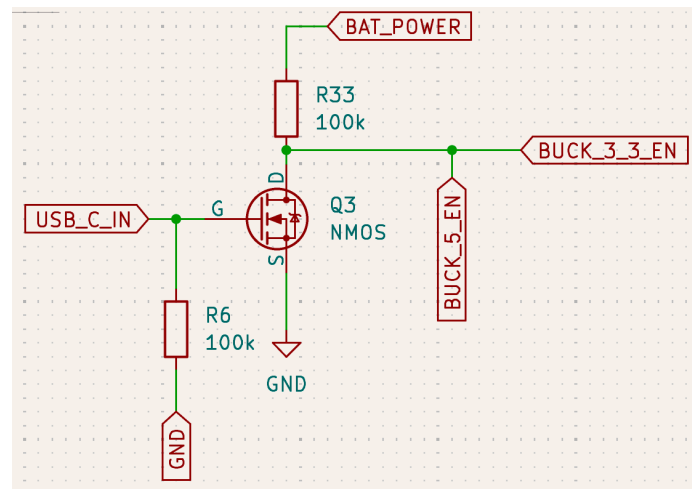


Fig C.1.4: Buck-Disabling Circuit Schematics

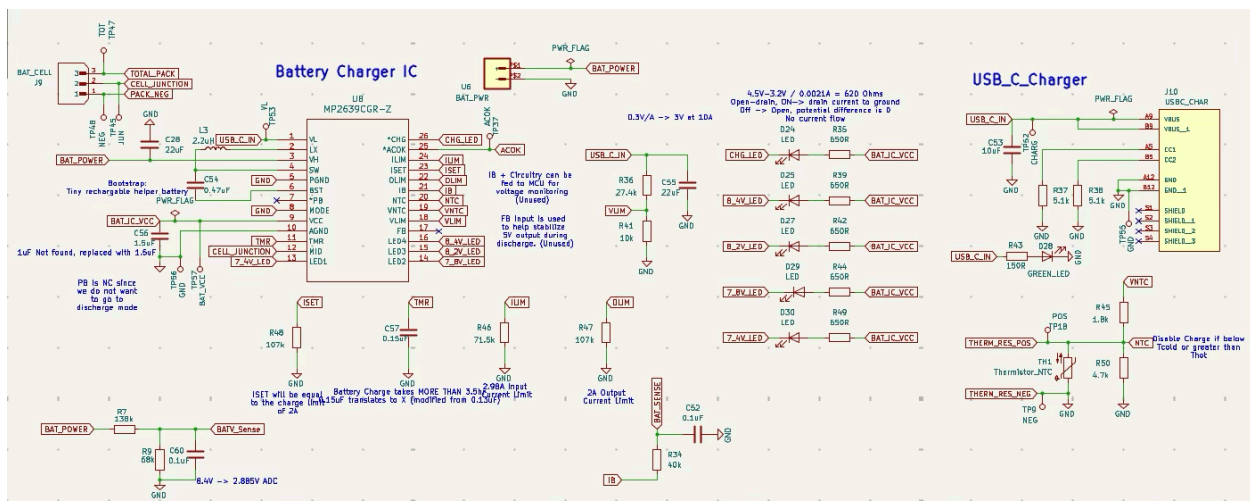


Fig C.1.5: Battery Charger Schematics

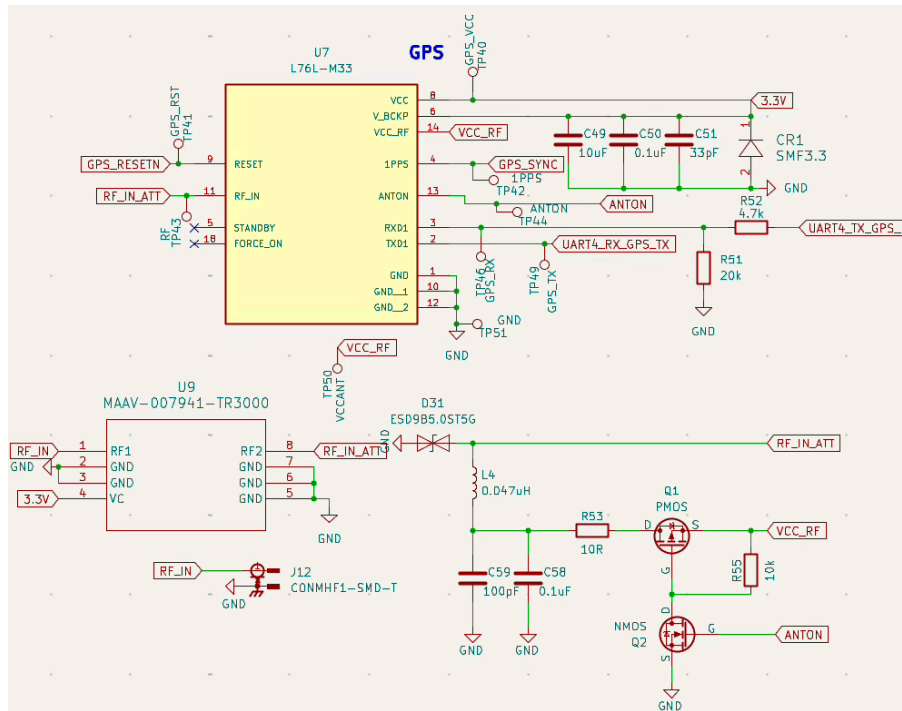


Fig C.1.6: GPS and GPS Antenna Circuit Schematics

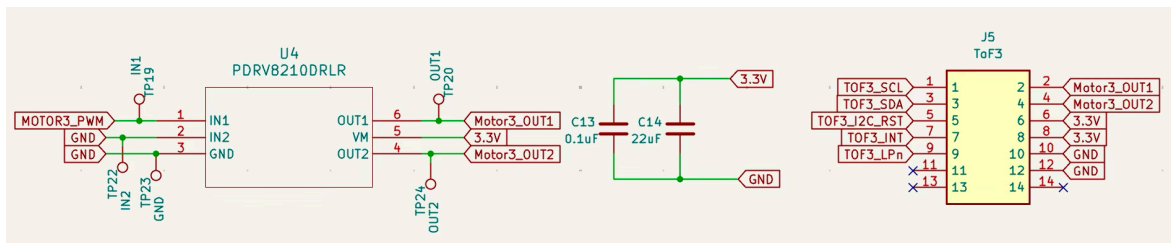


Fig C.1.7: Main PCB Motor Driver + ToF Connector Schematics

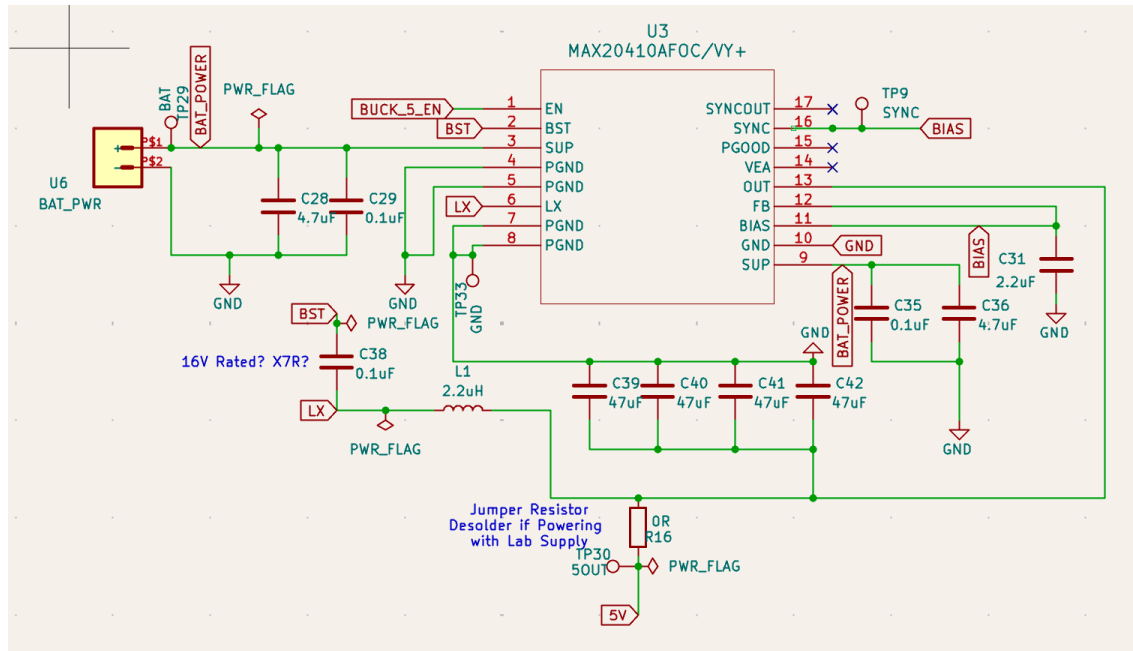


Fig C.1.8: Old 5V Buck Converter Schematics

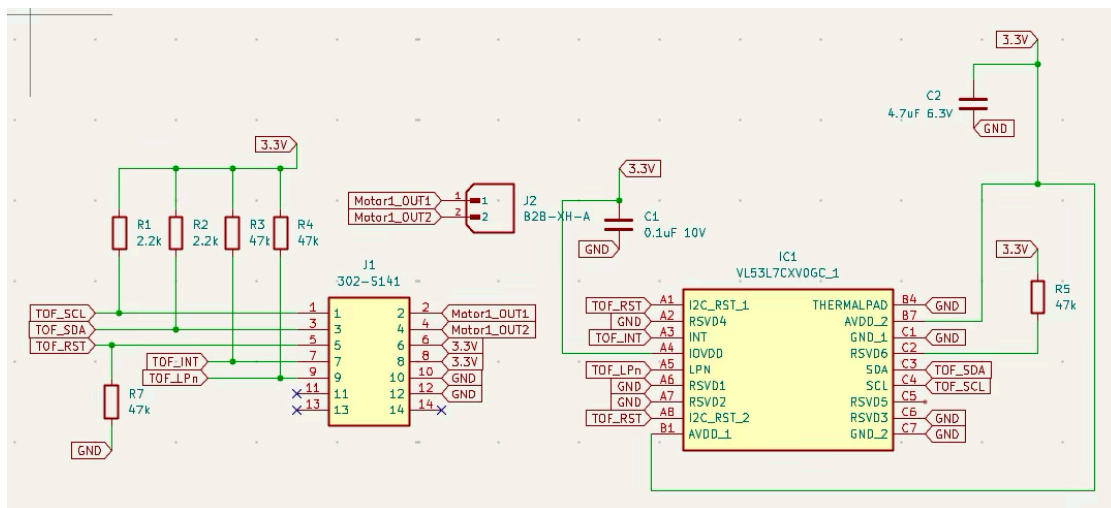


Fig C.1.9: ToF and Motor Schematics

Appendix C.2 PCB Layout and Annotations

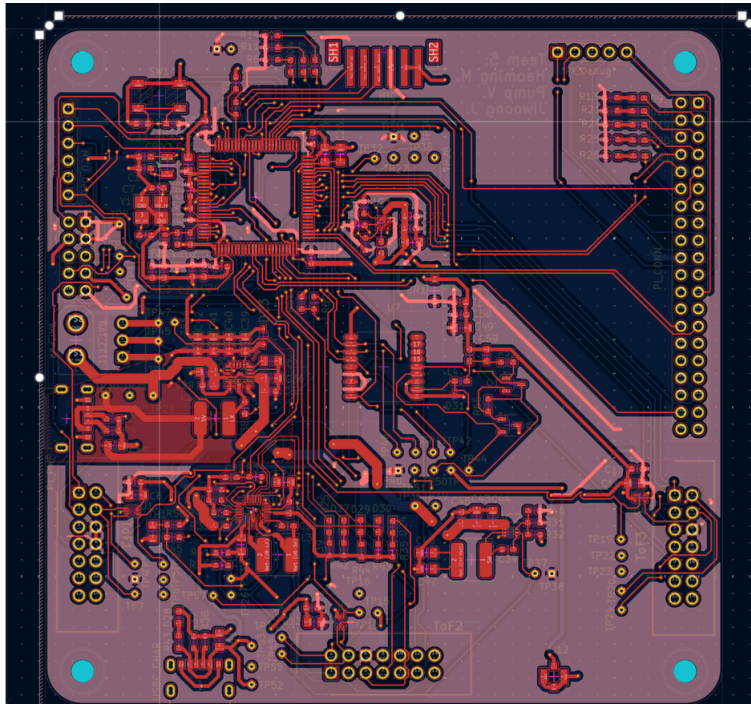


Fig C.2.1: 3.3V Layout Copper Zone

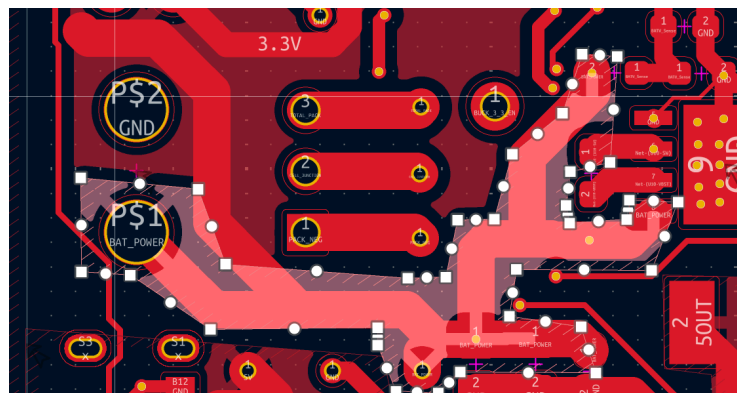


Fig C.2.2: Custom-Copper Zone and Increased Trace Width for Battery Rail

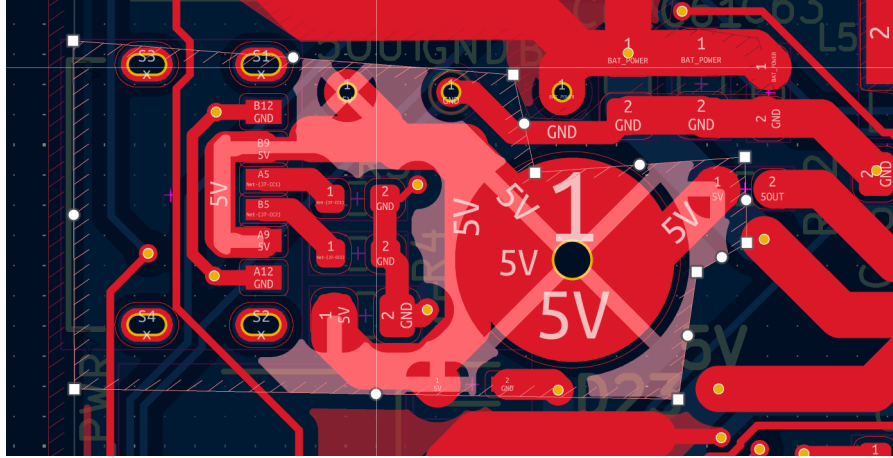


Fig C.2.3: Custom-Copper Zone and Increased Trace Width for Battery Rail

Appendix C.3 Electrical Diagrams and Equations

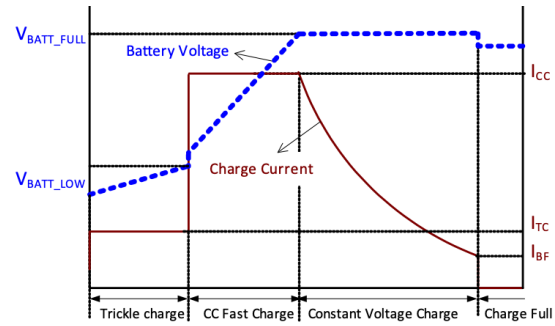


Fig C.3.1: MP2639C Charging Phases. Reproduced from [10].

$$I = \frac{dQ}{dt}$$

Fig C.3.2: Trickle Time Equation

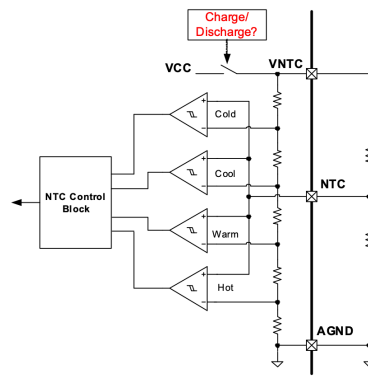


Fig C.3.3: MP2639 NTC Network Diagram. Reproduced from [10].

Appendix C.4 Electrical Cost Analysis

Fig C.4.1 Digikey Purchases

Part Name	Digikey Part #	Unit Cost (USD)	Order Quantity	Subtotal (USD)
Capacitor: YAGEO CAP CER 0.1UF 16V X7R 0603	CC0603KRX7R7BB104	\$0.006	27	\$0.16
Capacitor CAP CER 22UF 16V X5R 0603 KYOCERA AVX	0603YD226MAT2A	\$0.47	4	\$1.88
CAP CER 18PF 16V X8R 0603 KEMET	C0603C180K4HACTU	\$0.10	4	\$0.4
CAP CER 10UF 16V X5R 0603	GRM188R61C106MAALD	\$0.22	6	\$1.32
CAP CER 4.7UF 16V X5R 0603	CL10A475KO8NNNC	\$0.08	3	\$0.24
CAP CER 2.2UF 16V X5R 0603	CL10A225KO8NNNC	\$0.08	3	\$0.24
CAP CER 0.22UF 25V X7R 0603	GRM188R71E224KA88D	\$0.10	1	\$0.10
CAP CER 47UF 16V X5R 1206 TDK Corporation	C3216X5R1C476M160AB	\$0.81	6	\$4.86
CAP CER 33PF 25V COG/NP0 0603 KYOCERA AVX	KGM15ACG1E330JT	\$0.10	1	\$0.10
KA-TR CAP CER 0.47UF 25V X7R 0603 Taiyo Yuden	TMK107B7474	\$0.12	1	\$0.12
CAP CER 1.5UF 16V X7S 0603 TDK Corporation	CGA3E1X7S1C155K080AC	\$0.27	1	\$0.27

CAP CER 100PF 25V C0G/NP0 0603 KYOCERA AVX	KGM15ACG1E101JT	\$0.10	1	\$0.10
CAP CER 0.15UF 25V X7R 0805 KYOCERA AVX	KGM21ER71E154KU	\$0.13	1	\$0.13
LED YELLOW CLEAR 0603 SMD Würth Elektronik	150060YS75000	\$0.15	8	\$1.20
LED GREEN CLEAR 0603 SMD Würth Elektronik	150060GS75000	\$0.15	6	\$0.90
ESD9B5.0ST5G	TVS DIODE 5VWM SOD923 onsemi	\$0.10	1	\$0.10
SMF3.3 Littelfuse Inc. TVS DIODE 3.3VWM 6.8VC SOD123F	F7701CT-ND	\$0.49	1	\$0.49
Diodes Incorporated TVS DIODE 5VWM 14VC SOD323	DESD5V0S1BA-7	\$0.10	1	\$0.10
RES 10K OHM 1% 1/10W 0603 YAGEO	RC0603FR-0710KL	\$0.10	7	\$0.70
RES 5.1K OHM 1% 1/10W 0603 YAGEO	RC0603FR-075K1L	\$0.10	1	\$0.40
RES 300 OHM 1% 1/10W 0603 Vishay Dale	CRCW0603300RFKEAC	\$0.10	1	\$0.80
RES 4.7K OHM 1% 1/10W 0603 YAGEO	RC0603FR-074K7L	\$0.10	4	\$0.40
RES SMD 1K OHM 1% 1/8W 0603 Vishay Dale	CRCW06031K00FKEA	\$0.10	1	\$0.10
RES 32K OHM 1% 1/10W 0603	RN73R1JTDD3202F50	\$0.10	1	\$0.10

KOA Speer Electronics, Inc.				
RES SMD 40K OHM 0.1% 0.15W 0603 Vishay Dale Thin Film	PAT0603E4002BST1	\$0.60	1	\$0.60
RES 620 OHM 1% 1/10W 0603 Vishay Dale	CRCW0603620RFKEAC	\$0.10	1	\$0.50
RES 27.4K OHM 1% 1/10W 0603 Stackpole Electronics Inc	RMCF0603FT27K4	\$0.10	1	\$0.10
RES SMD 150 OHM 5% 1/10W 0603 Panasonic Electronic Components	ERJ-3GEYJ151V	\$0.10	1	\$0.10
RES 1.8K OHM 5% 1/10W 0402 KOA Speer Electronics, Inc.	RK73B1ETTP182J	\$0.10	1	\$0.10
RES 71.5K OHM 1% 1/10W 0603 KOA Speer Electronics, Inc.	RK73H1JTDD7152F	\$0.10	1	\$0.10
RES 107K OHM 1% 1/10W 0603 YAGEO	RC0603FR-07107KL	\$0.10	2	\$0.20
RES 20K OHM 1% 1/10W 0603 YAGEO	RC0603FR-0720KL	\$0.10	1	\$0.10
RES 10 OHM 1% 1/10W 0603 Stackpole Electronics Inc	RMCF0603FT10R0	\$0.10	1	\$0.10
RES 10K OHM 1% 1/10W 0603 YAGEO	RC0603FR-0710KL	\$0.10	7	\$0.70
RES 5.1K OHM 1% 1/10W 0603 YAGEO	RC0603FR-075K1L	\$0.10	1	\$0.40

RES 300 OHM 1% 1/10W 0603 Vishay Dale	CRCW0603300RFKEAC	\$0.10	1	\$0.80
RES 4.7K OHM 1% 1/10W 0603 YAGEO	RC0603FR-074K7L	\$0.10	4	\$0.40
RES SMD 1K OHM 1% 1/8W 0603 Vishay Dale	CRCW06031K00FKEA	\$0.10	1	\$0.10
RES 32K OHM 1% 1/10W 0603 KOA Speer Electronics, Inc.	RN73R1JTDD3202F50	\$0.10	1	\$0.10
RES SMD 40K OHM 0.1% 0.15W 0603 Vishay Dale Thin Film	PAT0603E4002BST1	\$0.60	1	\$0.60
RES 620 OHM 1% 1/10W 0603 Vishay Dale	CRCW0603620RFKEAC	\$0.10	1	\$0.50
RES 27.4K OHM 1% 1/10W 0603 Stackpole Electronics Inc	RMCF0603FT27K4	\$0.10	1	\$0.10
RES SMD 150 OHM 5% 1/10W 0603 Panasonic Electronic Components	ERJ-3GEYJ151V	\$0.10	1	\$0.10
RES 1.8K OHM 5% 1/10W 0402 KOA Speer Electronics, Inc.	RK73B1ETTP182J	\$0.10	1	\$0.10
RES 71.5K OHM 1% 1/10W 0603 KOA Speer Electronics, Inc.	RK73H1JTDD7152F	\$0.10	1	\$0.10
RES 107K OHM 1% 1/10W 0603 YAGEO	RC0603FR-07107KL	\$0.10	2	\$0.20
RES 20K OHM 1% 1/10W 0603	RC0603FR-0720KL	\$0.10	1	\$0.10

YAGEO				
RES 10 OHM 1% 1/10W 0603 Stackpole Electronics Inc	RMCF0603FT10R0	\$0.10	1	\$0.10
CONN HDR 40POS 0.1 GOLD PCB R/A Sullins Connector Solutions	PPPC202LJBN-RC	\$2.43	11	\$2.43
CONN HDR 10POS 0.1 TIN PCB Sullins Connector Solutions	PPTC052LFBN-RC	\$0.60	1	\$0.60
CONN HDR 6POS 0.1 TIN PCB Sullins Connector Solutions	PPTC061LFBN-RC	\$0.42	1	\$0.42
CONN UMC JACK STR SMD TE Connectivity Linx	CONMHF1-SMD-T	\$0.20	1	\$0.20
CONN HDR 40POS 0.1 GOLD PCB R/A Sullins Connector Solutions	PPPC202LJBN-RC	\$2.43	11	\$2.43
CONN HDR 10POS 0.1 TIN PCB Sullins Connector Solutions	PPTC052LFBN-RC	\$0.60	1	\$0.60
CONN HDR 6POS 0.1 TIN PCB Sullins Connector Solutions	PPTC061LFBN-RC	\$0.42	1	\$0.42
CONN UMC JACK STR SMD TE Connectivity Linx	CONMHF1-SMD-T	\$0.20	1	\$0.20
FIXED IND 47NH 400MA 0.6OHM SMD Murata Electronics	LQG18HN47NJ00D	\$0.16	1	\$0.16

MOSFET P-CH 30V 4.3A SOT23-3L Alpha & Omega Semiconductor Inc.	AO3407A	\$0.46	1	\$0.46
SWITCH TACTILE SPST-NO 0.05A 12V Alps Alpine	SKQGABE010	\$0.31	1	\$0.31
CRYSTAL 24.0000MHZ 10PF SMD Abracon LLC	ABM8G-24.000MHZ-B4Y-T	\$0.84	1	\$0.84
FC-135 32.7680KA-K0	SER4087TR-ND	\$0.54	1	\$0.54
SENSOR MR I2C/SPI 12LGA STMicroelectronics	IIS2MDCTR	\$2.56	1	\$2.56
Texas Instruments IC REG BUCK ADJ 3A SOT563	TPS563207DRLR	\$0.32	1	\$0.32
SENSOR OPTICAL I2C STMicroelectronics	VL53L7CXV0GC/1	\$8.59	3	\$25.77
RF TXRX MODULE NAVIGATION SMD Quectel	L76L-M33	\$10.57	1	\$10.57
DRV8210DRLR	296-DRV8210DRLRTR-ND	\$0.69	3	\$2.07
STM32H743VIT6	497-17720-ND	\$14.50	1	\$14.50
USB4125-GF-A	2073-USB4125-GF-ATR-ND	\$0.59	2	\$1.18
TPS563207DRLR	296-TPS563207DRLRTR-ND	\$0.32		\$0.32
TPS56628DDA	296-47733-ND	\$2.42	1	\$2.42
HIGH QUALITY GOLD PLATED XT30 MA DFRobot	1738-1436-ND	\$1.90	1	\$1.90
B3B-XH-A	455-2248-ND	\$0.11	1	\$0.11
B57861S0103F040	495-2142-ND	\$2.00	1	\$2.00
VLS6045EX-2R2N	445-173049-2-ND	\$0.34	1	\$0.34
VLS6045EX-3R3N	445-173052-2-ND	\$0.34	1	\$0.34
LQG18HN47NJ00D	490-6582-2-ND	\$0.16	1	\$0.16

SC1431	2648-SC1431-ND	\$70.00	1	\$70.00
--------	----------------	---------	---	---------

Fig C.4.2 Non-Digikey Purchases

Part Name	Part #	Unit Cost (USD)	Order Quantity	Subtotal (USD)	Link
MP2639C	MP2639CGR-P MP2639CGR-Z	\$3.84	1	\$3.84	Here
Battery	21700 5000mAh Battery for TX16S and TX12 MKII	\$16.99	1	\$16.99	Here

Appendix D Software References

Appendix D.1 Navigation functions

```
def request_nav(origin_latitude, origin_longitude, dst_latitude,
dst_longitude):
    #! Message to send in the desired json format required by
the API
    body = {
        "origin": {
            "location": {"latLng": {"latitude": origin_latitude,
"longitude": origin_longitude}}
        },
        "destination": {
            "location": {"latLng": {"latitude": dst_latitude,
"longitude": dst_longitude}}
        },
        "travelMode": "WALK",
        "polylineQuality": "HIGH_QUALITY",
        "polylineEncoding": "ENCODED_POLYLINE"
    }
    response = requests.post(url, headers=headers, json=body)
    #Read Navigation directions via. The newly created .json
file
    with open("response.json", "w") as f:
        json.dump(response.json(), f, indent=4)

    return response
```

Fig D.1.1: request_nav() function program

Appendix D.2 ONNX Quantization

```
from onnxruntime.quantization import(quantize_static,
CalibrationDataReader, QuantFormat, QuantType, )
import onnx
model_fp32 = "model.onnx"
model_int8 = "model_int8.onnx"
quantize_static(model_input = model_fp32, model_output = model_int8,
calibration_data_reader = dr, quant_format = QuantFormat.QDQ,
activation_type = QuantType.QInt8, weight_type = QuantType.QInt8,
per_channel = True, )
```

Appendix D.3 RPI Training Loop

```
import yaml
from ultralytics import YOLO

def merge_yamls(yaml1, yaml2, out_yaml="combined_dataset.yaml"):

    with open(yaml1, "r") as f:
        data1 = yaml.safe_load(f)
    with open(yaml2, "r") as f:
        data2 = yaml.safe_load(f)

    combined = {
        "path": ".",
        "train": data1["train"] + data2["train"],
        "val": data1["val"] + data2["val"],
        "names": {},
    }

    all_names = list(data1["names"].values()) +
list(data2["names"].values())
    combined["names"] = {i: name for i, name in
enumerate(all_names)}

    with open(out_yaml, "w") as f:
        yaml.dump(combined, f)

    return out_yaml

def train():
    merged_yaml = merge_yamls("Home0bjects-3K.yaml", "coco128.yaml")

    model = YOLO("yolo11n.pt")

    model.train(
        data=merged_yaml, epochs=50, imgsz=640, device="cpu",
workers=16
    )
    model.save("yolo_merged.pt")

if __name__ == "__main__":
    train()
```

Appendix D.4 Button Debouncing Logic

```
#!/ Sample Debouncing Logic From First Draft
def test_simulate_gemini():
    check_system()
```

```

test_sound() ## for testing the audio driver in the raspberry pi
button = Button(22, pull_up=False, bounce_time=0.1)
button_logic = 0
button_counter = 0;
while(1):
    if(button.is_pressed and button_logic == 0):
        capture_camera_picamera()
        query_llm() ## Check with the LLM
        button_counter += 1
        print(f'Button is pressed {button_counter} times!')
        button_logic = 1 ## for debouncing

    if(not button.is_pressed):
        button_logic = 0
return

```

- Same button logic was used for the GPS query.

Appendix D.5 Gemini/GPT Prompt Instructions

[LLM Prompt Instructions]

```

guidances_blind = """
I am blind and cannot see anything. All instructions must rely only
on confirmed, non-visual information. Do not guess.

Tell me where I am using only reliable facts such as landmarks,
building names, intersections, or sensor data. Describe my position
in a way a blind person can understand.

Tell me how my location relates to the destination using clear
relational descriptions. Examples: "The destination is the next
doorway on your right," or "It is across this walkway."

Describe what is directly in front of me. Clearly state whether the
path is open or if there is a wall, railing, door, or corner ahead.

Give clear, certain movement instructions. Use unambiguous language
like "turn left," "turn right," "continue forward," "stop," or
"slightly left/right."

All guidance must come from verified surroundings, detectable
landmarks, or sensor data. Avoid vague or visual-based descriptions.

Do not include any asterisks in your response.

Provide directions strictly in a step1, step2, step3 format.
"""

```

[LLM Response]

Current Location:

You are standing in front of a wall with a door to your left and another door directly in front of you. There is a sign with the number "2" on the wall to your right, along with a fire alarm.

Orientation and Immediate Surroundings:

- **In front of you:** Door.
- **To your left:** Door.
- **To your right:** Wall with a sign.

Step-by-step guidance to exit:

1. **Continue straight** towards the door directly in front of you.
2. **Open the door** (you may need to use a push bar to open it).
3. **Proceed through the doorway.**
4. **Stop** once you are completely through the doorway. I need more information about your surroundings once you are through the door to provide further instructions.

Appendix D.6 Audio Subsystems Text to Speech and Speech to Text

[Text to Speech]

```
def _speak(string):  
    print(string)  
    speaker = pyttsx3.init()  
    speaker.say(string)  
    speaker.runAndWait()  
    time.sleep(2)  
    return
```

[Speech to Text]

```
import sounddevice as sd  
import speech_recognition as sr  
  
def pcm_raspb_audio():  
    """  
    For this function, the pcm pins inputs, and it is memory mapped.  
    Need to ensure that  
    """  
    print("Recording Progress...")
```

```

sample_rate = 48000 ## from the main pcb
duration = 5
total_number_sample = sample_rate * duration

audio_out = sd.rec(int(total_number_sample),
                    samplerate = sample_rate,
                    channels = 1,
                    dtype = 'int16') ## the audio should be
extracted

## Channel = 1 -> Mono
## Channel = 2 -> Stereo
sd.wait()

recognize = sr.Recognizer()
try:
    destination = recognize.recognize_google(audio_out)
except Exception as error:
    destination = ""

print(f'Recording Done. Said Destination: {destination}')

return destination ## output string

```

[Audio Verification Test Program]

```

import speech_recognition as sr
import os
import pyttsx3
import time
import sounddevice as sd

def test_audio(string):

    print("Recording Progress...")

    speaker = pyttsx3.init()
    speaker.say(string)
    speaker.runAndWait()
    # time.sleep(1)

    sample_rate = 48000 ## from the main pcb
    duration = 5
    total_number_sample = sample_rate * duration

    audio_out = sd.rec(int(total_number_sample),

```

```

        samplerate = sample_rate,
        channels = 1,
        dtype = 'int16') ## the audio should be
extracted

## Channel = 1 -> Mono
## Channel = 2 -> Stereo
sd.wait()

recognize = sr.Recognizer()
try:
    destination = recognize.recognize_google(audio_out)
except Exception as error:
    destination = ""

print(f'Recording Done. Said Destination: {destination}')

return destination ## output string

if __name__ == "__main__":
    test_string = [
        "apple",
        "banana",
        "coffee",
        "music",
        "river",
        "window",
        "orange",
        "camera",
        "sunshine",
        "keyboard" .....] ## List truncated for space purposes, test
    string is length of 100

    get_string = []

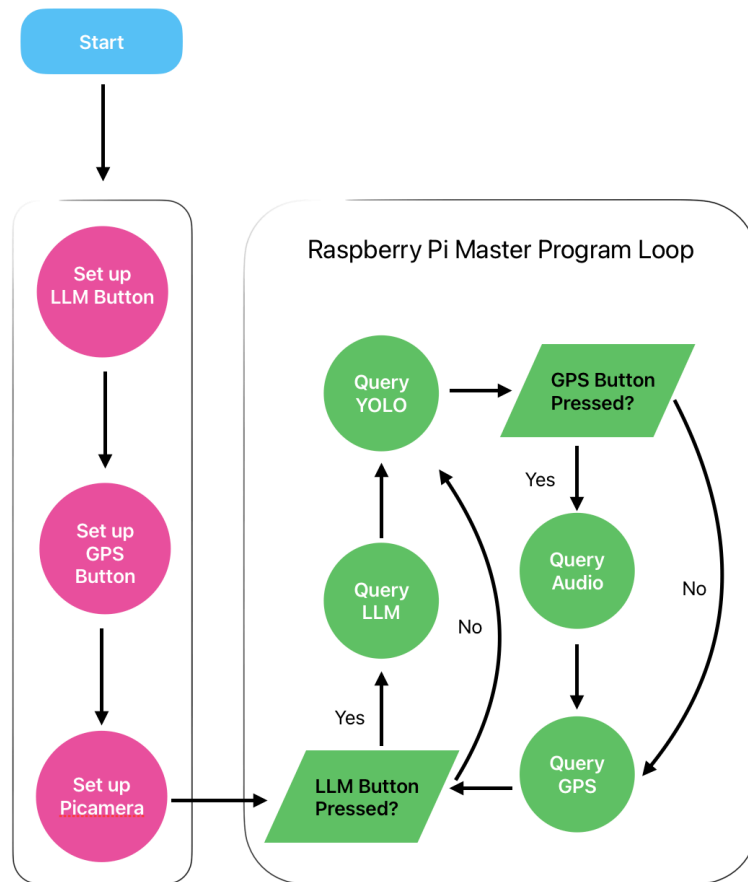
    for i in range(len(test_string)):
        get_string.append(test_audio(test_string[i]))

    total_equal = 0
    for i in range(len(test_string)):
        if (test_string[i] == get_string[i]):
            total_equal += 1

    print(f'Total Accuracy: {total_equal/len(test_string)}')

```

Appendix D.7 Raspberry Pi Master Program Flowchart



Appendix D.8 ToF Program Flowchart

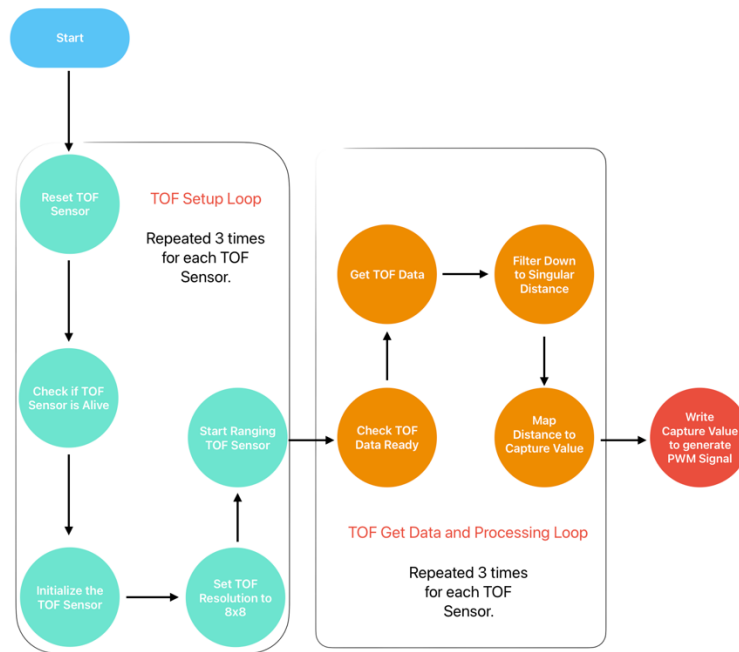


Fig E.1 Project Schedule and Weekly Member Contribution

Week 1	<p>Jiwoong:</p> <ul style="list-style-type: none"> • Initial Webpost • Brainstorming ideas for Pace <p>Haoming:</p> <ul style="list-style-type: none"> • Brainstorm ideas for pace <p>Pump:</p> <ul style="list-style-type: none"> • Initial Webpost • Brainstorming ideas for Pace
Week 2	<p>Jiwoong:</p> <ul style="list-style-type: none"> • Decision and working on Early Project Approval • Generate and finalize high level diagrams and requirements. <p>Haoming:</p> <ul style="list-style-type: none"> • Help revise High Level Requirement, Project Scope, and Subsystems required <p>Pump:</p> <ul style="list-style-type: none"> • Considered depth sensor options with Professor Kumar • Determined the components to be proposed on the Early Project Approval • Helped revise Project Scope, and Subsystem Components
Week 3	<p>Jiwoong:</p> <ul style="list-style-type: none"> • Searched and drafted specific parts for the project. • Proposal Writeup. • Design motion encoding table. • Drafted motor frequency and proximity graph for distance sensor to duty cycle mappings. <p>Haoming:</p> <ul style="list-style-type: none"> • Identified requirements based on the features of the project • Identified potential Subsystem needed to meet the project requirements • Researched Components Required such as Magnetometer <p>Pump:</p> <ul style="list-style-type: none"> • Developed initial FSM pseudocode for the ToF and PCB module • Researched limits to components and calculated feasibility of hotspots for outdoor usage • Designed the validation pipeline for the components • Picked out possible components for the haptic motors
Week 4	Jiwoong:

	<ul style="list-style-type: none"> • Finalizing Proposal Write up. • Raspberry Pi Specification Selection. • Microphone & Speaker Audio System Performance Metrics and Design. • Metric Measurement for Google Map API. • ML Model Performance Metric Design <p>Haoming:</p> <ul style="list-style-type: none"> • Researched components needed haptic feedback, distance sensing, and etc • Researched connectors required to connect the pcbs together • Researched hardware required for audio output • Selected battery according to power requirement • Selected buck converter needed for 5V and 3.3V Rail • Started the Schematics and helped debug the ToF for demo • Selected the GPS and microcontroller <p>Pump:</p> <ul style="list-style-type: none"> • Finalization of Proposal Write up. • Researched requirements for the Time-of-Flight sensors • Designed the Time-of-Flight PCB • Planned outline for first Breadboard demo • Refined calculations for sensor tolerance
Week 5	<p>Jiwoong:</p> <ul style="list-style-type: none"> • Researched top 8 on device model selection for computer vision. • Conducted structures for model score metrics. • Specified the model dependency and steps for setup. • Identified candidate model failures, which reduced the model selection down to RPI-YOLO Model. <p>Haoming:</p> <ul style="list-style-type: none"> • Worked on project proposal • Addressed concerns for thermal/heat • Aided group in creating visual aid • Selected battery charger IC • Purchased Breakout for Demo and Drew Schematics • Started and finished the Schematics for Microcontroller and sensing <p>Pump:</p> <ul style="list-style-type: none"> • Presented proposal of project to Professor and peers • Created High Level Diagrams, and Visual aid • Crafted a initial design of the product • Consulted the Machine shop for enclosure fabrication • Discussed synchronization challenges between MCU and Raspberry Pi
Week 6	Jiwoong:

	<ul style="list-style-type: none"> Conducted Model Scoring for YOLO Model on M4 Pro Macbook Pro and analyzed performance. Conducted Model Scoring for YOLO Model on Raspberry Pi and analyzed performance. Setup and recheck IOC for STM32 IDE on dev F1 board. Setting up the codebase for the TOF, which includes setting up the TOF object and API. <p>Haoming:</p> <ul style="list-style-type: none"> Finished Schematics for Sensory Output (magnetometer + gps) as well as the motor drivers Supervised the ToF + motor pcb design Finished DAC and Battery Charger IC Schematics Added Circuitry Protection <p>Pump:</p> <ul style="list-style-type: none"> Examined ST ToF driver Structure Studied platform-level driver functions for STM32 ToF program Helped set up the codebase for the ToF, planning out the code for the drivers Designed initial PWM mapping function using reciprocal -> later changed to exponential
Week 7	<p>Jiwoong:</p> <ul style="list-style-type: none"> Update the IOC file for the breadboard demo. Design algorithm and implement the motor test code to check if the PWM signals generates properly. Connected to the motor to check the output. Designed and drafted the TOF sensor code. Designed and implemented the ultrasonic sensor code as a backup. Used it in the first breadboard demo. Designed more specific mapping functions for the distance to motor duty cycle . <p>Haoming:</p> <ul style="list-style-type: none"> Added LEDs and test points for debugging Assigned footprint to prepare for the pcb layout Worked on design doc and the system level requirements <p>Pump:</p> <ul style="list-style-type: none"> Helped design the motor algorithm and debugged the PWM output. Drafted the initial ToF sensor code. Helped create the ultrasonic demo with the vibration motor for the first breadboard demo. Verified responsiveness and current draw levels of the demo circuit via oscilloscope Created first draft for PWM code for more specific mapping functions.

	<ul style="list-style-type: none"> Planned more robust ToF testing when it arrives
Week 8	<p>Jiwoong:</p> <ul style="list-style-type: none"> Updated IOC to reposition the SDA and SCL. Implemented the I2C test case program to ensure that TOF sensors are detectable in the system. Finalized TOF sensor software implementation and test. <p>Haoming:</p> <ul style="list-style-type: none"> Selected Heatsink Added Pads for 3.3V, 5V, and battery to allow debugging without using the battery Added Circuit for Voltage and Current Sensing <p>Pump:</p> <ul style="list-style-type: none"> Fixed ToF issues by updating the breadboard setup through adding more resistors Verified ToF output through using UART to debug Implemented test cases to ensure that the ToF sensors were working Built a Python heatmap visualization tool to detect common trends of zone outputs Ran distance tests under light/no light conditions from 200cm to 10 cm Studied the ToF status codes and common reflection issues
Week 9	<p>Jiwoong:</p> <ul style="list-style-type: none"> Worked on Grid -> Singular distance Algorithm through finding the status meaning. Designed the tof_grid_filter function to bring 8 x 8 grid down to singular distance. Updated the mapping function from distance to duty cycle to make sure that the equation is more smooth with exponential function. <p>Haoming:</p> <ul style="list-style-type: none"> Researched and designed magnetometer interface logic Wrote magnetometer core logic Helped draw the PCB solder plan <p>Pump:</p> <ul style="list-style-type: none"> Created setup for second breadboard demo. Figuring out solutions for the vibration motors which broke due to force. Assisted the design of the ToF grid filter function to bring the 8x8 values to a singular output. Helped fine tune the mapping function from distance to duty cycle. Soldered the protoboards to hold the motor for the demo as a solution to the force issue.
Week 10	<p>Jiwoong:</p> <ul style="list-style-type: none"> Updated the TOF motor function to take in count to Three TOF sensor. Decided the clock for the TOF sensor for faster scanning.

	<ul style="list-style-type: none"> • Help finalize the component placements. <p>Haoming:</p> <ul style="list-style-type: none"> • Reverified PCB schematics • Verified 3.3V Buck Converter • Debug 5V and decided to reselect and redesigned 5V Buck Converter • Debug the Battery Charger IC <p>Pump:</p> <ul style="list-style-type: none"> • Soldered main PCB and ToF boards • Corrected connector orientation to reduce noise • Did continuity tests to verify all the components soldered
Week 11	<p>Jiwoong:</p> <ul style="list-style-type: none"> • Soldering the TOF Sensor and Testing • Soldering the 5V Buck Converter <p>Haoming:</p> <ul style="list-style-type: none"> • Designed buck-disabling circuit upon usb-c charging • Soldering MCU and Debug Battery Charger IC <p>Pump:</p> <ul style="list-style-type: none"> • Researched the implementation of the GPS module • Studied NMEA sentence outputs from the GPS • Identified useful components for GPS navigation • Began writing first draft for MCU GPS parser
Week 12	<p>Jiwoong:</p> <ul style="list-style-type: none"> • Help verify 5V buck. • Started designing the Master program for the Raspberry Pi, that allows to connect the GPS subsystem, ML subsystem, and Audio subsystem. <p>Haoming:</p> <ul style="list-style-type: none"> • MainPCB version 4 arrived • Verified 5V buck converter • Verified the MCU and it is now programmable <p>Pump:</p> <ul style="list-style-type: none"> • Identified limitations of GPS + ToF + YOLO alone • Researched alternatives in the market and common solutions • Proposed the vision + LLM solution to enhance awareness and provide context
Week 13	<p>Jiwoong:</p> <ul style="list-style-type: none"> • Implemented the button configuration for the Gemini Model along with software implementation.

	<ul style="list-style-type: none"> Finished quantization and model training. Tested model performance. <p>Haoming:</p> <ul style="list-style-type: none"> Tested the entire system with microcontroller, motor driver, tof, and vibration motors Verified Voltage Stability Debug GPS <p>Pump:</p> <ul style="list-style-type: none"> Conducted additional GPS testing and soldering refinement Validated current NMEA sentences Debugging GPS chip output Assisted in the verification and soldering of the GPS antenna
Week 14	<p>Jiwoong :</p> <ul style="list-style-type: none"> Switched Raspberry Pi program from threading to sequential. Implemented the TTS and STT final software. Implemented Bash and Requirement software for development purpose. Help create wrapper function for navigation. <p>Haoming</p> <ul style="list-style-type: none"> Debug GPS Soldered and verified the magnetometer Fixed the magnetometer code Performed magnetometer calibration Found GPS issue + replaced incorrect components Verified GPS in outdoor and indoor Help verify navigation and debug communication between MainPCB and raspberry PI <p>Pump:</p> <ul style="list-style-type: none"> Implemented the Google Maps API using Python Requests Built helper functions to retrieve navigation data Designed full navigation logic and integrated it into the main Raspberry Pi function
Week 15	<p>Jiwoong:</p> <ul style="list-style-type: none"> Prepare the final presentation. Created extra credit video. Help testing via being a testing subject. Help create final presentation and report. <p>Haoming:</p> <ul style="list-style-type: none"> Prepared for final presentation Collected data for power subsystem and sensing subsystem Verified navigation in multiple location Identified points for improvement

	<p>Pump:</p> <ul style="list-style-type: none"> • Tested GPS output outdoors. • Recorded coordinate accuracy outdoors. • Verified outputs and ran software tests on computer via. SSH to Raspberry Pi • Helped with fall back algorithm for GPS through Hotspot • Prepared for final presentation. • Identified possible points for improvement on Navigation subsystem
--	---