SightMate

Josh Joung, Shakthi Angou, Meera Pandya

Department of Electrical and Computer Engineering,
Carnegie Mellon University

***Abstract*—**A system capable of providing an automated wearable navigational experience that will alert the user of obstacles in their vicinity along with the optional functionality of identifying the object. The detection will primarily focus on common indoor objects to provide visually impaired people the ability to explore unknown indoor spaces without the need of guide dogs. The system will broadly consist of an object recognition model, text-to-speech engine, and depth sensors tied together by an on-board computer to create a seamless navigation experience that will hopefully be a more accessible alternative to sighted guides.

***Index Terms*—**Distance Estimation (DE) feature, espeak, HC-SR04 Ultrasonic Sensor, NVIDIA Jetson Nano, Object Recognition (OR) model, PCB, pyttsx3, YoloV5

I. Introduction

Using guide dogs is one of the most common methods for visually impaired people to navigate through the obstacles and reach the destination. However, the problem occurs when the users are incapable of raising guide dogs due to the operation cost and unavailability. This project is attempting to address the inefficiency and inaccessibility in using these trained dogs for visually impaired people. Guide dogs are generally very costly to train and maintain, and sometimes unavailable in an indoor setting. In addition, they can be difficult for visually-impaired people to care for. The project aims to offer an accessible alternative for guide dogs, creating a wearable device to aid in maneuvering around obstacles for visually-impaired people. Specifically, the product is an automated navigation system that indicates to the user when they are approaching objects in front of them. It lets the users notice whatever object is in their pathways, similar to what guide dogs would do but with a much cheaper and easily maintainable solution. This product is intended to be used along with a cane, which is the most commonly used assistive device for the visually-impaired. There is a restriction to the scope of this project to well-lit indoor spaces with minimal to medium-level object crowding.

II. Use-Case Requirements

There were multiple requirements for the use case for the visually impaired people. Since our device is an affordable alternative to guide dogs, some of our requirements use guide dog qualifications as a baseline metric.

1. *Battery duration*

The first requirement was sufficient battery duration. It should operate for a minimum of 4 hours because a guide dog usually takes a break every 4 hours. People also typically spend less than 4 hours exploring around indoor settings.

2. *Accuracy of the object recognition model*

A relatively high accuracy of the object recognition model was necessary for visually impaired people to utilize the product. The minimum qualification to become a guide dog is 70%, so the project aimed for the minimum accuracy to be 70%. Yet, the objective accuracy is 80% to ensure the user's safety and usability.

3. *Detection distance*

The detection distance is 2 meters to give users enough time and distance to avoid an obstacle once the alert is triggered.

4. *Weight of the product*

The weight of the product was planned to be no more than 450 grams to minimize strain on the user's neck. As a backup plan, we considered offloading the battery pack to the waist, but this proved unnecssary after measuring the weight of all the components.

5. *Recognition delay*

The average walk speed of blind pedestrians is 0.8 m/s, so the upper limit of the recognition delay was constrained to be less than 2.5 seconds to permit 2-meter space.

6. *Noise detection*

Regardless of the audio device in the product, the users should be able to hear surrounding noises regardless to ensure safety and reduce danger concerns.

III. Architecture and/or Principle of Operation

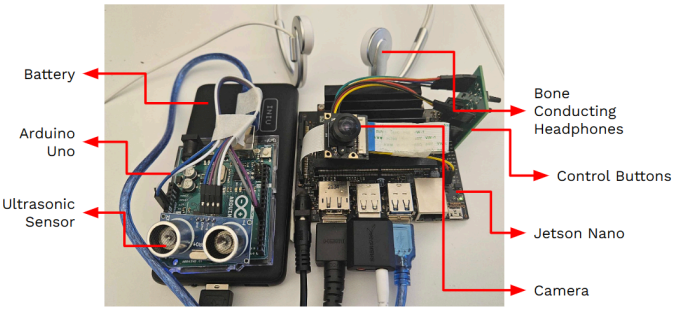


Fig. 1. Annotated image of the integrated device without casing.

The system primarily consists of an object recognition model with a distance estimation feature, a text-to-speech engine, and depth sensors tied together by an on-board computer, creating a seamless navigation experience

that will hopefully be a more accessible alternative to sighted guides. As seen in Fig. 1, we are using the NVIDIA Jetson Nano as our onboard computer due to its compatibility with the YoloV5 model given its GPU and its ability to work with ML libraries like OpenCV. An Arduino Uno is used to connect the ultrasonic sensor. There are two buttons to control the proximity and speech modes, a camera to collect visual data and bone conducting headphones that output speech without preventing the user from hearing their surroundings. The device is powered by a portable battery and has an estimated usage time of 5 hours.

objects from the OR model. By comparing this data with pre-collected reference measurements, it estimates the distance of the objects and identifies the closest one.

Lastly, the speech system uses a TTS engine called espeak that takes the OR model's output and converts it into speech. When button B is pressed, the user hears a speech output of the closest object in their path through bone-conducting headphones, as well as the relative distance of the obstacle (close or far) and where it is located relative to the user (left, right, or center). If the model is unable to detect the object successfully, the user hears "not detected".

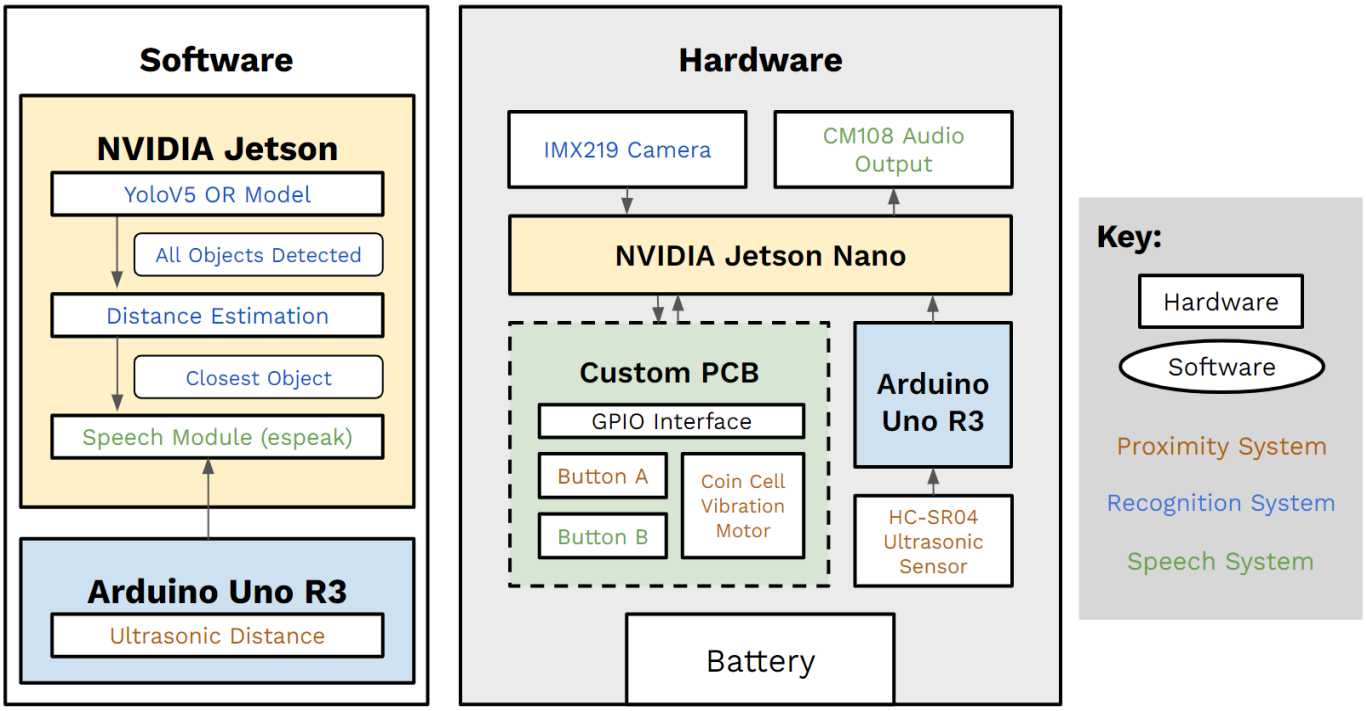


Fig. 2. The overall architecture of software and hardware components.

Our device can be divided into three subsystems: the proximity, the object recognition system, and the speech system.

The proximity system uses an Arduino Uno R3 board connected to an ultrasonic sensor that measures distances and channels data to the Arduino's serial output. The serial output stream is then pulled by the Jetson and stored in a global variable. A vibration motor is connected to the Jetson, and is triggered if the proximity setting is turned on with button A and an object is detected under 2m. This system runs alongside the OR model using the multithreading package in Python.

For the object recognition system, the camera attached to the Jetson will use the GStreamer program to send an image at 1FPS to the YoloV5 OR model, which then yields an array of all objects detected in the frame. Then, the DE module leverages pandas to extract the box width of detected

The peripheral devices used in these subsystems are connected to the Jetson by our custom PCB, which controls current flow using resistors and a transistor, and interfaces the buttons and vibration motor with the Jetson's GPIO pins. The camera, the audio converter, and Arduino are directly connected to the jetson through the MIPI connector port and USB ports, and the ultrasonic sensor connects to the Arduino's digital output pins. The entire system is powered by a 10,000 mAh rechargeable power bank, which can easily run the device for over 5 hours.

As seen in Fig. 2, we have utilized the bottom-up principle in which we have broken down the solution into several modules with a single functionality and later integrated them. Individual modules are tested first before the integration because it is challenging to identify the error in one module when there are several modules working together. Hence, we are using this principle to mitigate the risk as soon as possible at the earliest stage. It also allows parallelism because each team member can work on a distinct module at the same time, which reduces the implementation time. Once each module is successfully implemented, then modules are combined to

produce each subsystem. To illustrate, control buttons, an ultrasonic sensor, and a vibration motor are integrated for the proximity system. Likewise, one module can be utilized in multiple systems, and this allows an efficient and resource-saving development through such "recycling."

The principles of science have been used in the DE feature from the OR model. As it is impossible to determine the distance of an object with a single camera, we have utilized the principle of focal points to determine the distance. To do so, a known distance and width of the object are measured in advance. Then, The focal length is calculated by using

$$focal = \frac{(width\ in\ reference * known\ distance)}{known\ width}.$$

Width in reference is from the reference data collected beforehand by running the OR model on our selected items.

Then, the distance is calculated by using

$$distance = \frac{known\ width * focal}{width\ in\ frame}.$$

Width in frame is retrieved by $x_{max} - x_{min}$ that can be collected from the OR model.

The principles of mathematics have primarily been used in the testing. The concept of average and uncertainty has been utilized to determine the success of the testing stage and compare the results between each implementation.

IV. Design Requirements

A. Camera to OR model frame rate

The camera of the device captures images and forwards it to the object recognition module at a constant frequency of 1 frames per second (fps). This process of capturing images and directing it to the OR model happens continuously, regardless of the device modes described above. Given that humans perceive visuals comfortably up to 24 fps, achieving a higher frame rate would undoubtedly improve the experience for the user, in terms of usability. However, for the needs of this device, which is mainly to provide the user with situational awareness of a static environment, along with resource constraints for battery life and other hardware used, 1 fps is the metric we arrived on. This requirement ensures that the recognition delay is less than 2.5 seconds.

B. Proximity Module Feedback Frequency

The sensor polling frequency refers to how many times per second the ultrasonic sensor is going to measure for distance, and based on the hardware specification of the HC-SR04 Ultrasonic Sensors we are using, this value is set to 25 Hz. However, when rerouting this sensor data to the vibration motors, a much lower frequency is required so as to not overload the user with sensory output. A comfortable frequency of vibration is 2-4 Hz. To ensure that the proximity module has minimal latency, this module is separated from the OR module of the device, with minimal layers from end-to-end. This design can therefore contribute to meeting the recognition delay of 2.5 seconds by alerting the user sufficiently for them to notice the potential danger.

C. Proximity Detection Distance

A distance of 2 meters before the user encounters an obstacle is a safe range within which the user can move away from it safely. Given this, we set 2m to be the threshold of object detection for the proximity module, meaning that when the ultrasonic sensor detects an object under this threshold distance, this triggers the program that communicates to the user of the object they are approaching, as described in the above section on proximity feedback via vibration motors.

D. Battery Life Analysis

The goal in terms of battery life was to allow for 4 hours of continuous usage. This is in alignment with the typical length of navigation and aid provided by service dogs before they take breaks, and a reasonable target for duration spent outside for daily activities before access to recharging the device. Given this metric, below is a breakdown of the power consumption of the major components of the device:

Component	**Current (mA)**
NVIDIA Jetson Nano	2000
Camera	491
Ultrasonic Sensor	5
Vibration Motor	85
Audio Converter	70
Total	**2651**
4-Hour usage	**10,604 mAh**

To allow for the 4-hour usage requirement, a ~10,600 mAh battery was needed. The actual battery life per use may vary depending on the size of the OR module and other latencies in our overall system. After full subsystem integration, we ran the device on battery power until the battery ran out, and found that even after 4 hours, 60% of the battery capacity remained, indicating that the device is capable of running for up to 10 hours before recharge.

E. OR Module Accuracy

Mean Average Precision (mAP) is a widely used metric for evaluating the performance of object recognition models (Buhl), particularly in tasks such as image classification or object detection with multiple classes. It calculates the average precision across all classes, providing a comprehensive measure of a model's ability to correctly identify objects of interest in an image. Industry standard considers an mAP of 0.4 - 0.6 to be reasonable to good performance. In the context of guide dogs, which are trained to assist visually impaired individuals, it's noted that their

accuracy in guiding is expected to be around 0.7 mAP. To ensure that our OR model could effectively supplement or exceed the capabilities of guide dogs, the aim was to achieve an overall accuracy rate higher than 0.8 mAP. This higher number was in line with our goal to address the need for precise computer vision systems that help visually impaired people move safely and freely. Consequently, the user requirement of having 70% accuracy of OR model could be met.

F. *Wired Bone Conduction Earphone*s

The Jetson is connected to wired bone conduction earphones for the audio output. They allow the users to hear background noise and device output simultaneously through their structures and designs. Hence, this strategy met the use case requirement of the users being able to hear the surrounding sounds to notice a potential danger. The wired option is relatively cheap and easily accessible for the users, so it can also reduce the production cost. The device case also has a removable side, giving the user access to the audio port if they prefer to connect their own audio device.

G. *Modules with Minimal Weight for Integration*

The size and weight of individual modules are assessed and considered to alleviate the weight of the product as much as possible. The product consists of a small camera module that is capable of sending real-time image data, and one HC-SR04 Ultrasonic Sensor is used for a proximity measure. The minimum size of PCB that is able to handle the functionality of integration is used to contribute to minimizing the size and weight of the product. This design strategy met the weight requirement of 450 grams.

V. Design Trade Studies

A. Hardware

We chose to use an NVIDIA Jetson Nano to run the software and mount the hardware for our device, largely due to its highly performant GPUs which will be capable of continuously running the OR model.

Initially, we considered using a Raspberry Pi 4 as our single-board computer. It has a vast community along with abundant online resources relevant to OR models, camera modules, and ultrasonic sensors, which may potentially save us time in figuring out the integration of modules. However, the problem with RPi4 is overheating of the product. When multiple modules involving a ML program are run, it tends to overheat, causing high latency and potentially even a system shutdown during the run. This issue is relevant because it has a low power video processor, so using a camera module will frequently lead to overheating. To mitigate this overheating risk, we considered hosting the OR model on a server and using the RPi to communicate with the server to send images and receive recognition data. The issue regarding this mitigation plan is that RPi4 needs to be connected to Wi-Fi to send data to the server. Because it would be challenging for visually impaired people to manually connect Wifi in an unknown indoor environment, this plan has been aborted.

The hardware device that can handle the cons of RPi4 is the NVIDIA Jetson Nano. Although it has less online documentation and community support than RPi4, the biggest factor of using Jetson is that it contains higher performance and more powerful GPUs than RPi4. Therefore, Jetson is more suitable in using a ML model like this project. It has less frequent overheating and allows flexibility in development compared to the RPi.

B. OR model version

We chose the YoloV5 model that is developed by Ultralytics. It is built on the PyTorch framework, which makes it easier to use and fine-tune for developers. Considering that this project needed a model with a DE feature, using this version reduced the development time. The Yolo versions generally have relatively high performance in OR. They are also very commonly used for real time data processing, so sending real-time data with a camera module can be easily implemented in this project. Furthermore, there are vast online resources and tutorials, so the learning curve in using this model was reduced. A potential issue was the accuracy of the model, since there are more recent versions that have greater response time and higher accuracy. Another potential issue was the integration of the DE feature. Because the integration is not provided in the open source library, some time was designated to integrate the feature onto the YoloV5 model. However, there is open source documentation for the integration of the DE feature to the Yolov4 model, so it was used as the primary source for implementation.

Several alternatives to using the YoloV5 model were considered.

The Detectron2 model, supported by the Facebook research group, provides a modular programming design, so it can be flexible and customizable. It also includes several features including pre-trained models and mixed precision training. However, there is a steep learning curve to the model. Understanding its architecture, configuration, and API's is required to fine tune the model. Additionally, not only it relies on high-end GPUs and large amounts of memory to train the model but also many dependencies. This model could cause compatibility issues or dependency conflicts during the deployment.

An alternative plan was to use the Yolov4 OR model that has a DE feature attached to it. Because the model is available in an open source library, the project could have used the code directly, which can reduce a lot of development time. Using the results of the detected object and corresponding distance, only fine tuning the data is needed to output the desired functionality of the product. However, this model is a pretrained model with several objects that are irrelevant to indoor settings. Therefore, it cannot recognize a common indoor object like a table or a sofa. Furthermore, this model cannot be retrained with a personalized dataset anymore because the development team no longer supports a "darknet" module, which has been used to train the OR model. Despite the increase in development time, it is critical to re-train the OR model to an indoor object dataset to meet with the scope

of the project. Therefore, a different version needed to be considered.

Another consideration of the model is using Yolov8, which is the most recent version of the OR model. It has the highest detection speed and greatest accuracy among all versions. However, it is not built on the PyTorch framework, which makes this version harder to use and fine-tune for developers. Considering that a DE feature has to be attached to the model too, using Yolov8 can be problematic in spite of its high speed and accuracy.

C. *Using headphone jack adapter instead of Bluetooth to connect audio device*

Since the Jetson does not have an audio connector onboard, we considered several methods of incorporating audio output into the device. In particular, we had to choose between using a wireless Bluetooth connection or a wired connection. The Bluetooth connection would be preferable for allowing the user greater mobility than the wired connection, but incorporating a Bluetooth module would require an interface for users to pair their device, and our design plan does not include an interface already. Additionally, we were concerned that Bluetooth would draw more power than a wired audio connection, especially if an interface needed to be implemented. We would then require a larger battery capacity to meet our 4-hour usage requirement. Since a larger battery introduces more weight on the user, we chose to implement the wired connection for the sake of the user's comfort.

VI. System Implementation

A. *OR module*

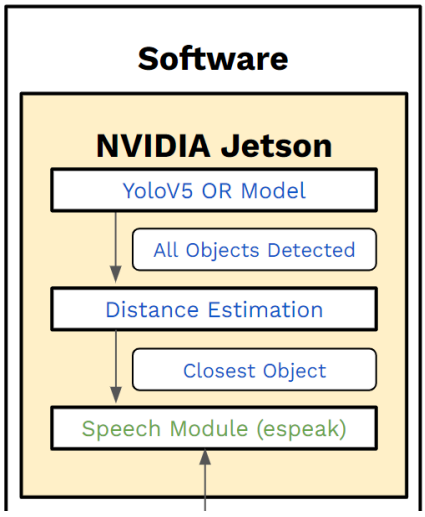


Fig. 3. The OR model architecture connecting a camera module, Jetson, and audio module

The initial plan was to use the OR model synchronously with the video capture module called Gstreamer from opencv. This means that for each frame captured from the camera, we run the OR model and DE feature to determine the closest object. However, we realized that due to the latency in the OR model, there was a frame delay, which created a huge data latency around 8 seconds for every 20 seconds of running, which heavily violated our use case requirement for the data latency.

Hence, we used multithreading to run the OR model asynchronously to the video capture program. This allows the OR model to use the frame, updated every second by the camera module. In the event that the race condition is met, the data fetched from the global variable would be from the previous instance, which is at most 1 frame behind real time, which works with the use case requirement. This would not be noticeable to the user and hence does not affect their navigation experience. For this reason, we decided not to use mutex or other lock methods for the global variable, which can potentially create a bottleneck and increase the latency of data transfer.

Once the OR model outputs all detected objects and relevant data such as boxwidth, x-min, x-max, object name, then it sends them to the distance estimation module. By comparing the reference data, it converts the boxwidth of each object to a distance. Then, it calculates the minimum distance and sends the closest object and its distance to the speech module. At the same time, it also sends x-min and x-max values, which are the x-coordinates of the box drawn on the detected object. By using these coordinates, the speech module determines whether the object is at the left or right of the user and whether it is at close distance or at far distance. Then, the user is able to hear the speech with the information of the object detected, its relative position and relative distance to the user, which supports our objective of assisting in navigation.

B. *Proximity module*

a. Software

We set a 2m range for the proximity module, meaning that objects detected within a 2m range are communicated to the user via the vibration motors if the setting is turned on. Based on the design requirements section highlighting the 25 Hz polling frequency of the ultrasonic sensor, we distilled the data transfer down to 2-4 Hz before producing vibration feedback, so as to avoid the risk of overwhelming the user. This process of filtering data before routing to the vibration motors is handled by this program.

b. Hardware

The initial plan was to connect the ultrasonic sensor directly to the NVIDIA Jetson Nano but upon testing of the subsystem, there were timing issues that remained unresolved even after implementing multithreading in Python. This prompted us to conduct modular testing of the hardware to try to isolate the cause. However, upon research in the NVIDIA community, we found that users had faced similar challenges with connecting ultrasonic sensors to the NVIDIA Jetson and the consensus was to avoid doing so due to the Jetson's inability to handle real time processing well. Moreover, the

particular sensor we had was most compatible with an Arduino Uno, and hence this triggered the switch to offloading the proximity module to an Arduino board. The Arduino communicates with the Jetson by directing the measured distance data to its serial output that the Jetson will then pull and convert to a vibration output if the mode is turned on by the user.

The vibration mode is toggled on and off with a push button which is directly mounted on the PCB, while the ultrasonic sensor and vibration motor are connected to the Arduino and PCB using jumper wires to allow for flexibility in placing the components. The vibration motor is located on the carrying strap on the back of the user's neck instead of on the main device, so that the vibration can be felt regardless of the user's clothing.

C. *Speech module*

a. Software

This module employs a text-to-speech engine called espeak, along with the wrapper library pyttsx3. The voice used as the speech output is customized with a rate, pitch, accent, and specific phrases that will help the user navigate. Initially we had implemented an output that simply gave the name of the object ("person") or "not detected" when the user pressed the button to identify an object. However, during our testing stage, we realized that the speech output had potential to provide even more helpful information to the user, especially since the OR model already had a distance estimation feature. Hence, we incorporated the relative direction of the object (left, right, center), and distance (close, far) into the speech outputs. An example would be "person detected at left in close distance". The relative distance is outputted based on the OR model's boxwidth and where the box edges are relative to the center of the user.

During integration of the speech module with the OR model on the Jetson, particularly during the transition to making the device "headless" (running entirely without a monitor), we were faced with many bugs relating to the permissions of the different programs running together. Our first step in the process was to make the main file run upon reboot of the Jetson, without needing us to manually run a script. For a main file to run during startup, it required root permissions that when given began interfering with the speech module's audio output permissions. We were faced with many hours of debugging to identify and resolve this and became familiar with the Advanced Linux Sound Architecture. The changes we had made to the permissions had soon become a problem and completely broke the OR model. As we had about a week to resolve the problem before the final demo, we decided to reflash the Jetson's SD card and redo the painstaking process of installing all the packages we needed and its dependencies, which took over 6 hours of installation. Despite this adding time to our initial work plan, restarting the process helped us start from a clean slate and avoid the mistakes we made in the previous round.

b. Hardware

The speech module output was implemented using a CM108-based USB-A to headphone jack adapter. Since the Jetson has USB-A ports available on the board but no onboard audio port, this was the simplest way for us to implement an audio connection, and allows the user to connect the headphones they are most comfortable with. The vibration setting can be toggled on and off by a push button, which is mounted on the custom PCB.

D. *Custom PCB*

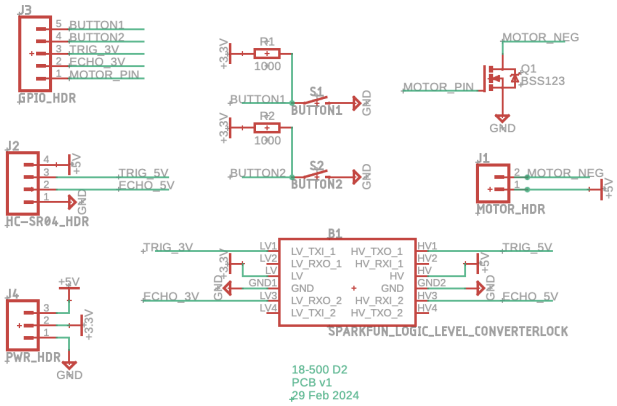


Fig. 4. A design of a custom PCB.

Designing a custom PCB allowed us to safely connect the buttons and vibration motor to the Jetson Nano's GPIO pins. The J4 PWR_HDR pins connect to the Jetson's 5V, 3.3V, and ground pins to supply power to the PCB, and the J3 GPIO_HDR pins connect to the Jetson GPIO pins which we configure in software for each hardware component.

The two control buttons each have a pull-up resistor to regulate current flow to the Jetson to prevent damaging the pins (walterfms2i). The BUTTON1 and BUTTON2 pins are configured as GPIO inputs for the Jetson to read the button value.

The vibration motor we used is a 3V DC motor. Since the direction and speed of the motor is not necessary for us to control, we used an N-channel MOSFET to supply current to the motor instead of using an H-bridge or an off-shelf motor driver. The gate of the MOSFET is driven by a GPIO output pin, and the motor connects to the PCB via the J1 MOTOR_HDR pins.

E. *Wearable Case*

The wearable device case consists of a box housing the hardware components and a carrying strap integrated with the vibration motor. The box, designed using Fusion 360, is made of laser-cut cardboard pieces for each side, with tabbed edges that interlock together to improve the case's stability. Excluding the front and right side panels, all sides are secured together with hot glue. The front and right panels have longer tabs with Command Velcro Strips to attach them to the rest of the case. This allowed us to access the Jetson's USB ports, which we needed for running the device non-headlessly for debugging purposes, as well as for changing the audio output device. Internally, the case has cardboard supports to contain the battery, Arduino, and Jetson, preventing them from shifting position while the device is in use.

Our camera strap is a modified camera strap, since camera straps come with additional padding to ensure the user's comfort. We modified the strap by running the vibration motor along the strap to the back of the user's neck, and sewed a fabric strip onto the strap to conceal the motor and its wires.

VII. Test, Verification and Validation

A. *Results for Accuracy of OR module*

The accuracy of the recognition system was dependent on two features. The first is testing the accuracy of the object recognition. The verification is done if the OR model is able to recognize the items in an indoor object dataset that has multiple items for each image. Both a pre-trained YoloV5 model that is trained by MS COCO, which consists of millions of images with many different objects, and a trained YoloV5 model, which is trained with the customized indoor object dataset consisted of around three thousand images, are tested and compared to determine which model to use. After training the YoloV5 model with the indoor dataset, it was run to detect a test dataset of indoor objects. Surprisingly, among 58 real objects, the trained model only detected 21 objects correctly, which yields a 36.2% accuracy. On the other hand, the pre-trained model successfully detected 49 out of 58 objects, which yields a 84.4% accuracy. Although the tradeoff of using a pre-trained model is that the classification can only be done for predetermined indoor objects, it had far greater accuracy than the trained model.

The second test is for the DE feature of the OR model. This functionality was verified by checking whether it can identify the closest item in different types of images that consists of one of the five chosen indoor objects. When running a test file that goes through 40 test images for 5 objects, the DE feature correctly identified the closest object for 38 images. This result gives a 95% accuracy, which greatly exceeds the use case requirement of 70% OR. These two tests are relevant and applicable to this use case requirement because the type of model to use heavily determines the accuracy. Once the first feature testing passed, the DE testing was done subsequently.

B. *Results for proximity module*

Two features of the proximity module were tested. The first feature is the accuracy of distance detection and the second is its capability to vibrate when an object within 2m has been detected.

The first feature testing is relevant to the use case requirement to accurately detect the existence of an object. Only if this module accurately captures the distance of an object then can the module alert the user of an immediate obstacle. 8 different distances were tested for the accuracy of the ultrasonic sensor. When it ran on 5cm, 10cm, 50cm, 100cm, 150cm, 200cm, 250cm, and 300cm, it gave only -1cm error, which gives an uncertainty of approximately 0.06%. This result is significant when compared to how the testing for the distance measured by the DE module resulted in 21.5% uncertainty. Therefore, we tested the accuracy of the proximity module in distance detection, which is the use case requirement to detect distance at least 2m for ± 30cm error.

The second feature testing is relevant to this modular testing because the proximity module should be able to detect an obstacle within 2m. This element is essential for users to avoid obstacles and is relevant to their safety, so this testing was conducted rigorously. By placing a person at 10cm, 50cm, 100cm, 150cm, 200cm, 250cm, and 300cm, the test checked whether the vibration motor turns on when the detected distance is less than or equal to 2m. The test result shows that the vibration motor is turned on for all distances under 2m, which yields a 100% accuracy on the proximity sensor. This accuracy meets the use case requirement of having a 95% accuracy on the proximity module.

C. Results for speech module

The main form of testing included a user-testing to determine whether the speed of speech output by the device is a comfortable one. Along with this, we ensured that the device's outputs do not prevent the user from still being able to hear their surroundings as we know that the visually impaired do rely on their hearing. The TTS engine used is one of industry standard and is proven to have high performance accuracy and was customizable to meet our requirements with rate of speech output, volume, accent etc. The quantitative aspect in the testing process includes fine-tuning the length of speech output (how descriptive does the object identification need to be), along with the time given between the outputs in the continuous object identification mode. Hence, testing this module was done mainly user-centric, with some part of it involving quantitative testing to meet overall latency requirements. The first part of the testing was done by consulting three students on determining the rate and volume of the speech module. A speech rate from 100 to 200 was tested with the increment of 10, and all participants responded with 170 as the most comprehensible speed. With the same respondents, we tested the volume of the speech from 1.0 to 5.0. There was a mixed result with a mean of 3.0, so the volume was decided to be 3.0. This user testing allowed the device to meet the user's comfort as best as possible.

Another test involved recognizing the surrounding noises despite the audio output. We conducted concurrent testing on the participants to determine to which direction they heard a clap sound while the audio output was running. The indoor objects have been output to the bone-conducting headphones while the clap sounds are made in the background. The users successfully recognized where the noises came from and which object the speech module output for 20 trials on 5 different objects respectively. They also all agreed that the bone-conducting headphones allow no interference of background noises. With 100% accuracy on the speech module, we met the use case requirement of the users being able to hear background noises.

D. Results for device settings

We tested the device's speech and vibration modes in several stages. First, we tested the buttons on the PCB once it was fabricated by connecting the Jetson's GPIO pins and writing a simple program to ensure that a button press was detected. Failure in this stage would have indicated that the PCB may need to be debugged and redesigned. Once this stage succeeded, we used the button presses to set the device's settings to each possible combination of speech and vibration modes and ensured that the device could support each mode. We considered this test a success if every mode combination was supported.

The first testing was conducted by having a program output a print statement once the button is pressed. For each button A and button B, 20 trials were done and achieved success on every trial, yielding another 100% on the unit testing. This result indicates that the device does not have any problems on the PCB side and simply needed to revise the mode control commands.

The second testing was done on an integrated speech, proximity, and recognition system. While the OR model was running, the test was conducted by clicking on the buttons to change the vibration setting and the speech mode. 20 trials for each button were done to test the functionality of each button. Button A is used for changing the vibration mode while button B is used to output a single speech identification of an object. Again, the trials resulted in 100% success in changing the modes. By achieving 100% on the functionality of the buttons, we met the use case requirement for the device controls. This result was essential for the user's convenience and usability of the device.

E. Results for the integrated device

After integrating different modules into one device, weight testing, functionality testing, and battery life testing were conducted. Using a scale, we measured that the device weighed 426g, which included an Arduino board, ultrasonic sensor, portable battery, Jetson Nano, and a camera module. By achieving a weight less than 450g, which was the use case requirement for the device weight, our device ensured the user's comfort. A comfortable neck strap was also used to emphasize comfort even with long-term usage of the device.
Functionality testing was measured by setting up an environment with indoor objects and having the user navigate through the environment by identifying the closest obstacles in

front of them. This testing was conducted on one user with a blindfold on, and there were 5 different setups for chairs and a person. The user accurately detected the closest object with 100% accuracy. However, we faced limitations regarding the environment. The test was conducted in a closed setting with no moving objects and very few objects placed in the user's path. We realized that the device gave false positive results when there were many objects placed in a lab, for example, where there were many people and chairs along with many other objects. Since we limited our scope to an indoor environment with few objects, this issue did not yield significant harm in our project.

Battery life was tested by connecting the portable battery to run the program on the Jetson Nano. We measured the percentage of battery loss after a certain period of time to estimate the battery life of the device. During the test, we ran the program headless and let it run for 4.86 hours. Then, 60% of its battery was depleted. Therefore, we estimated that the battery life was at least 5 hours, greatly exceeding the use case requirement of having 4 hours of battery life. This was essential because it allowed the user to navigate in an indoor environment for a longer period of time without worrying about the device shutting down.

F. Results for the recognition delay

The recognition delay was tested by measuring the time it took to recognize an object when a camera moved to a different direction to face a different object. This testing was relevant to the real user setting because it could measure the time it took to detect a new closest object when the user moved around. A total of 30 trials were conducted, and the result averaged to 1.88s to recognize an object. Two different camera settings of 1FPS and 2FPS were tested to see if there was a significant change in latency, but the 2FPS recognition delay was 1.61s, which was very minute considering that the use case requirement was 2.50s. To reduce the overhead on the device, we decided to use the 1FPS setting because it still had a low data latency while sending less data input to the device program. Having 1.88s of latency allowed the user to identify the obstacle at ease at a walking speed.

VIII. Project Management

A. Schedule

We faced many schedule changes throughout the project due to various factors. We were pushed to continuously adapt our schedule due to design changes, unforeseen delays in implementation and debugging, and iterative testing. Planning for slack time along the way was very helpful and enabled us to stay on track despite all the changes we encountered. Attached to the end of the report, at appendix A, is the schedule we used for this project,

.

B. Team Member Responsibilities

We separated our overall system into 3 subcategories that each of the members of our team take the lead on. However, the overarching functionality of the device was ensured by all of us, covering overall integration, the testing and verification process, as well as the outer look (design) of the device. This splits the work as follows:

Team Member(s)	**Primary Responsibility**
Josh Joung	OR and DE Modules
Meera Pandya	Hardware Implementation and Device Casing
Shakthi Angou	Proximity and Speech Modules
All	Overall Integration, Device Design, Testing and Verification

C. Bill of Materials and Budget

We were able to source several components for our device from personal or class inventories. The remaining components that we purchased from external sources are well within our budget, with some main costs including the customized PCB, the strap to hold the device and make it wearable, bone-conducting headphones, and the power banks. A summary of the items purchased as well as estimated future costs are listed at the end of this report (appendix B).

D. Risk Mitigation

1. Weight of device

Our goal was to keep the overall weight of the device under 400-450g. However, we anticipated that this estimate might still be uncomfortable, or that the device might become even heavier, jeopardizing the intended use-case of the device. To address this, we had planned for the backup of offloading the battery pack to a waist-strap, so as to distribute the total weight and improve practicality. We made this plan during our testing phase, anticipating feedback from participants who volunteered to test our device.

During development, we successfully met the weight requirement, and during user testing, users were comfortable with the weight. We had a risk plan of offloading the battery, but fortunately, it wasn't needed.

2. Connection to peripherals (Custom PCB)

We planned to connect the peripherals (buttons, sensor, and vibration motor) to the GPIO pins of the Jetson, with a custom PCB in between to manage the voltage and current levels. A risk with this approach was that custom PCBs take time to order, and there may not have been enough time to redesign a PCB if there were bugs. We managed this risk by first breadboarding the PCB circuit to ensure it was adequate for safely connecting the peripherals before we placed the PCB order. Our contingency plan in case the PCB still had bugs was to replace the PCB with an Arduino, which would have required us to switch to serial communication between the

Jetson and Arduino and would have caused us to reevaluate our power and weight requirements.
However, we got the PCB right on the first try, avoiding the need for any contingency measures.

3. Image classification base model and DE feature

Our current OR module was built off of an existing industry standard model called the YoloV5. This was a change from the initial plan to use the Yolov4, which upon further research we decided to change as that version of the model did not accommodate training with our own dataset, which was a key functionality we wanted our model to have. This steered us towards the YoloV5, which had improved functionality and support. If we faced issues with the accuracy of the model, we had a stretch-goal to upgrade to the Yolov8 as our base model, which was the most recent version. The same mitigation method as the OR module would have been executed to raise the accuracy of the DE feature. However, in the final stages, we were happy with the accuracy of the Yolov5 OR model and were able to make the adjustments we needed, such as adding reference objects we needed, calibrating the distance estimation feature to improve its accuracy, and reducing any data latency by removing any bottlenecks.

4. Ultrasonic detection range

Ultrasonic sensors have a range of approximately 2 centimeters to 4 meters, with a sensing cone of 30 degrees ("Ultrasonic Distance Sensor"). If we had experienced failures to detect an obstacle within range, we planned to include additional sensors as needed to cover the entire necessary sensing range. If the sensors falsely indicate the presence of obstacles in the 2-meter range, we planned to reduce the detection threshold on the software end to prevent erroneous sensing. However, upon testing an integration, we found the ultrasonic sensor tied to the Arduino Uno and ran on its own separate thread using multithreading to be a very accurate subsystem that did not need any further fixing. This was due to the major design change made to offload the proximity module to the Arduino as described above.

IX. Ethical Issues

With a device that directly concerns the safety of the user, along with privacy due to the presence of an OR model, ethical issues should certainly be considered. Although to us as the designers of this device the functionality may appear easy to understand, to someone who does not have knowledge of the technical aspects of this device may be more prone to misinterpretation or misuse of the device. For example, using the device in a dark environment or running while using it would both be instances that the device's scope does not cover and someone who doesn't understand this can definitely misuse the device. This may not only result in incorrect navigational outputs given to the user that may confuse them and lead them in unintended paths, it can also hurt them if they are in harm's way. Hence, to mitigate a situation like this, it is important to accompany such a device with educational information that will teach the user exactly when, where, and how to use the device so as to reduce chances for misuse. Another solution would be to add features that communicate to the user when they are using the device incorrectly. For example, if the lighting is too dim for the object recognition model to accurately output a value, we could implement a "pre-check" test to see if the lighting is good and if not instead of forcing an output, we can simply indicate to the user that there is poor lighting. Or if they are running while using the device, we could have a warning beep or a voice output to tell them to slow down before using the device.

Another edge case would be the range of detection and notification of obstacles to the user. The user should not entirely rely on the proximity module to know when an object is approaching as in some cases, where there is a fast moving object like a person on a skateboard passing by, the 2m detection range may not be adequate. So if the user is either unaware of the range, or becomes overly reliant on it, this may lead to issues. Once again, an educational package along with the device could help reduce the likelihood of this happening.

X. Related Work

During the initial stages of our project, we drew inspiration from a device named Theia that was built by Anthony Camu (Camu) , a final year Industrial Design student at Loughborough University in 2021. Their device has a similar purpose of aiding navigation for the visually impaired via a handheld device. It uses LIDAR technology in combination with a control moment gyroscope to construct a 3D image of the user's surroundings and determine the safest path to take. Our device modifies most parts of this project but we certainly were inspired by this student-lead project to provide a cheaper and more accessible alternative to guide dogs.

XI. Summary

Our system was capable of meeting all of our design specifications, and significantly exceeded our requirement for battery life. Unfortunately, our device does have several limitations in performance. Firstly, the angle of detection of the ultrasonic sensor is significantly limited in comparison with the camera. As a result, the sensor struggled to detect obstacles at ground level even when the obstacle was within the frame of the camera, and the camera's wide angle lens detected objects which were not a direct obstacle due to the user. Additionally, we were unable to increase the length of the strap due to the vibration motor's wires being a fixed length along the strap, which led to varying effectiveness depending on the user's height. Given more time, we would resolve these issues by incorporating additional ultrasonic sensors to increase the angle of detection, and we would either purchase a camera with a narrower angle of visibility or would incorporate image processing before running the OR model to ensure that the device detects only imminent obstacles. To resolve the height issue, we would solder longer wires onto a vibration motor and look into ways of disguising the extra wire length when the strap is extended.

A. Lessons Learned

To other groups interested in developing navigation devices

or accessibility devices, we would emphasize the importance of consulting the group of people they intend to help. For us, speaking to the LAMP visual impairment advisory board was incredibly valuable in our ideation and design stages. We received constructive feedback on our idea and heard suggestions for features to incorporate, such as adding cats to our list of detected objects. Speaking to the group that the device assists is useful in determining whether there exists a need for the project and what improvements could be made. We would also like to emphasize the importance of conducting user testing on a diverse group of users. For us, the device's performance varies based on the user's height, and conducting more rigorous testing with users of different body types would have helped us refine the wearable case to be more universally effective.

The neck-wearable device essentially guides visually impaired users in an indoor environment to avoid obstacles and identify an item right in front of them. It encourages visually impaired people to go to an unknown environment, such as visiting an acquaintance's house, with less safety concerns. This project consequently motivates them to connect with more people and explore the world around them.

Glossary of Acronyms

MQTT – Message Queuing Telemetry Transport
OBD – On-Board Diagnostics
RPi – Raspberry Pi
DE - Distance Estimation
OR - Object Recognition

References

[1] Buhl, Nikolaj. "Mean Average Precision in Object Detection : A Comprehensive Guide." *Encord*, 5 November 2023, https://encord.com/blog/mean-average-precision-object-detection/. Accessed 1 March 2024.

[2] Camu, Anthony. "Introducing Theia: The Handheld Robotic Guide Dog – Polytec Personnel Engineering and Scientific Staff Cambridge Cambridgeshire." *Polytec Personnel*, https://www.polytec.co.uk/whats-new/introducing-theia-the-handheld-robotic-guide-dog.html. Accessed 1 March 2024.

[3] Dal, Asadullah. "Yolov4-Detector-And-Distance-Estimator." GitHub, 16 Apr. 2022, github.com/Asadullah-Dal17/Yolov4-Detector-and-Distance-Estimator.

[4] Jocher, Glenn. "Ultralytics/Yolov5." GitHub, 21 Aug. 2020, github.com/ultralytics/yolov5.

[5] nemestomi2. "Using HC-SR04 Ultrasonic Sensor with Jetson Nano?" NVIDIA Developer Forums, 11 June 2020, forums.developer.nvidia.com/t/using-hc-sr04-ultrasonic-sensor-with-jetson-nano/78861/3. Accessed 01 Mar. 2024.

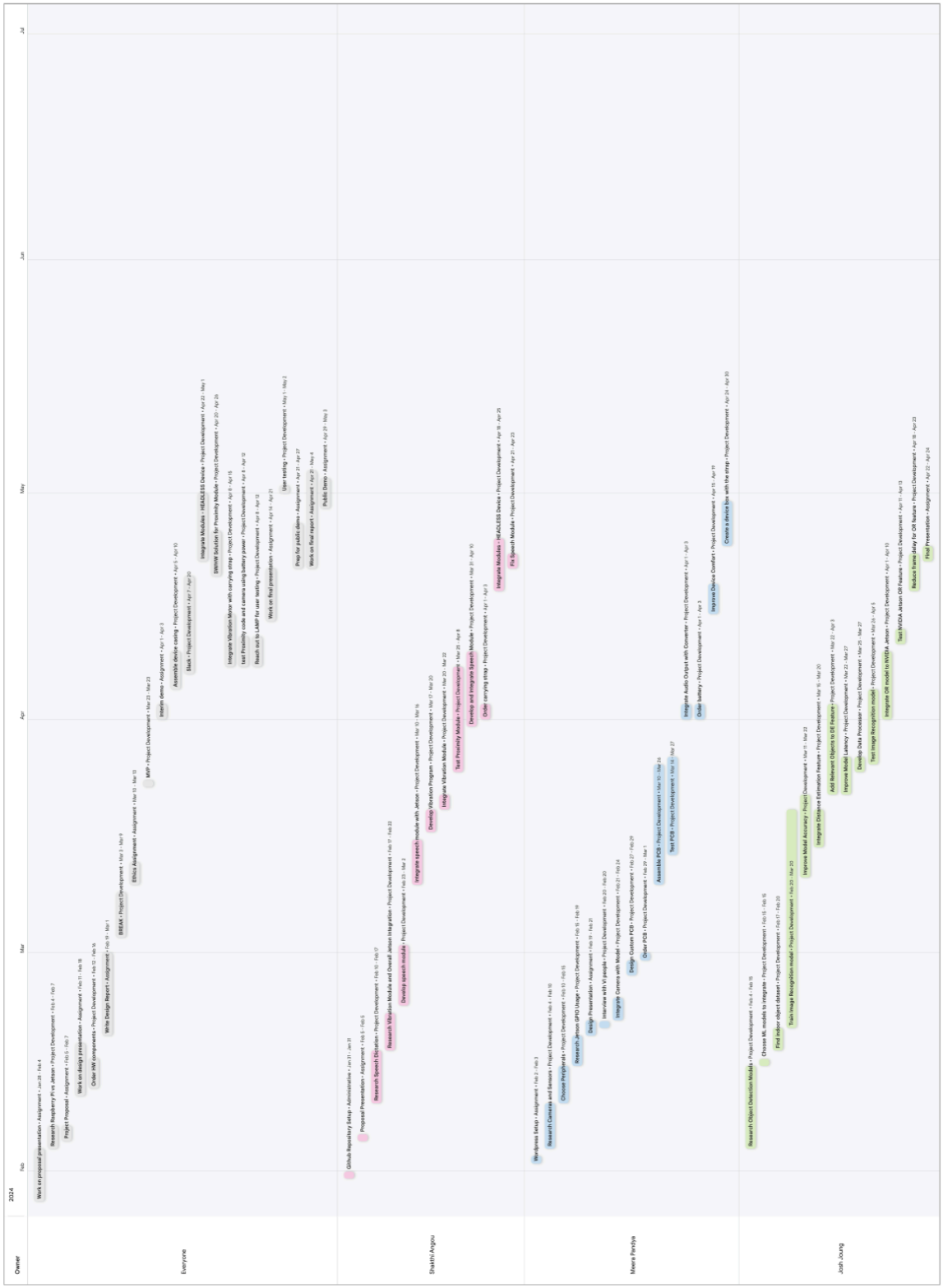
[6] "Ultrasonic Distance Sensor (HC-SR04)." PiSupply, uk.pi-supply.com/products/ultrasonic-distance-sensor-hc-sr04#:~:text=The%20HC%2DSR04%20sensor%20works,to%20the%20nearest%200.3cm.

[7] User, Deci. "Overview and Comparison of Neural Network Training Libraries." Deci, 4 Apr. 2022, deci.ai/blog/neural-network-training-libraries-tools/.

[8] walterfms2i. "Adding Buttons to Jetson Nano." NVIDIA Developer Forums, 19 Sept. 2019, forums.developer.nvidia.com/t/adding-buttons-to-jetson-nano/81942. Accessed 01 Mar. 2024.

[9] "YOLOv8 vs. YOLOv5: Choosing the Best Object Detection Model." Www.augmentedstartups.com, www.augmentedstartups.com/blog/yolov8-vs-yolov5-choosing-the-best-object-detection-model.

Appendix A: Schedule



Appendix B: Bill of Materials

Purchased Items	**Quantity**	**Cost**
IMX219-160 Camera	1	$28
Zyamy Micro Flat Vibration Motor	10 count	$7.59
64 GB MicroSD Card*	1	$13.99
5V 2A Power Supply*	1	$9.95
Ultrasonic Sensors	10 count	$9.99
CM108 Audio Card	1	$6.99
Custom PCB	5	$34.80
Battery Pack	2	$35.99
Wearable Device Straps and Casings	1	$14.99
Total		$162.29

Inventory Items	**Quantity**
Nvidia Jetson Nano	1
Breadboard, Resistors, Capacitors	Variable

* May not come from project budget , paid for by 18-500