The Bat Belt

Authors: Alex Lin, Kelton Zhang, Ning Cao
Affiliation: Electrical and Computer Engineering, Carnegie Mellon University

Abstract—**The Bat Belt is a smart mobility tool for the visually impaired. The design goal of this project is to achieve a better affordability and functionality balance between blind assist canes and guide dog with a lightweight wearable belt that provides obstacle detection with intuitively actionable haptic feedback. A highlight feature of the system is that both ground-level and waist-level detection will be available through a depth camera and ultrasonic sensor array respectively. Our product can alert users of both ground-level and waist-level obstacles with high accuracy while being much more affordable and easy-to-maintain than more advanced solutions.**

Index Terms—**Computer Vision, Embedded Systems, Haptic Feedback, Ultrasonic Sensor, Wearable Device, assistive technology**

1 INTRODUCTION

This project aims to address a real pain point in the society at large through a technical solution that utilizes both hardware and software sides of our team's skill set. Following this thinking, we envisioned The Bat Belt, a smart sensing belt that gives real-time haptic feedback to the visually blind to avoid collision with obstacles. The two most adopted solutions, the guide dogs and the white cane, are either hard to acquire and maintain or limited in functionality. Guide dogs take extensive amount of training beforehand and care throughout its lifetime; the white cane only offers ground level tactile feedback for contact points within a small range. Emerging technical products also each have their shortcomings, e.g. cumbersome and conspicuous neckwear[9], minesweeper-like cane bounded by the direction it is facing[14]. Given the status quo, our project is designed to fill in the gap with a lightweight wearable that provides both ground-level and eye-level detection with intuitively actionable feedback. The implementation is based on a off-the-shelf belt bundled with depth camera, ultrasonic sensor array, a set of vibration coin motors and Raspberry Pi 4 and Arduino Uno boards (Figure 1) to drive computation. To measure the engineering outcome of The Bat Belt, we tested the prototype indoor against a concrete set of metrics that includes detection accuracy, response rate, feedback user testing as specified in the requirements section.

2 USE CASE REQUIREMENTS

From the rudimentary product definition of a wearable belt that alerts the visually impaired user with haptic feedback to avoid colliding with obstacles, we further develop the following qualitative use case requirements to guide the design and testing process:

1. Lightweight: should support a full day of use without fatigue. weight $\leq 1kg$
2. Long Battery life: should support a full day's movement after one battery charge. consecutive operating time $\geq$ 2 hours
3. Reasonable detection range: 4 meters range of detection
4. Reliable detection: should give confident product warnings when there is danger. response rate: false positives (alert when there is no danger) $\leq 10\%$, false negatives (no alert in danger) $\leq 5\%$.
5. Relatively low-cost: should be relatively affordable, our goal is $\leq$ 500$
6. Real-time feedback : should give the user enough and as much as possible time to react with the right move, system response time $\leq 0.1s$

Each of these requirements' quantitative constraint is placed by our users' needs, and our reasoning of these quantification is discussed in order:

In terms of weight, we want our user to be able to walk around without too much burden. Our reference point is based on two criteria: the first being that it is estimated that a normal person can carry around 20 % of their weight in backpack without feeling too tired [4]. For a person weighing 50kg, much lower than the average female adult (77kg), this brings us to around 10kg limit. The second is that utility belt typically weighs around 3.2 pounds (1.5 kg) without loading any tools[3]. Based on these criteria, we believe that having our belt weigh less or around 1kg, less than an utility belt and much less than the tiring limit, would satisfy our definition of lightweight.

For the battery life, we want our belt to support at least a full day of walking activity. Since design review, we have re-evaluated our quantification and made appropriate adjustments. It is reported that an average person walks for around 35 minutes everyday, and the suggested daily walking time for losing weight is 1 hour long[2]. As a result, we believe that doubling the suggested walking period would be enough to support the walking time spend by an user everyday. According to a study by research gate[1], around 12 % of waking time is spent walking everyday, which also adds to the reasoning that 2 hours of consecutive walking time would support a full day's activities ($2 \div 0.11 = 16.66$)

For detection distance, we believe that we want to provide a reasonable look ahead. According to BAWA cane[15], a blind cane production company, the proper length of a blind cane is about 2 inches shorter than the user's height. consider this with the average height of an adult, the average detection range of a blind cane is around 1.2 meters. We want to provide a length that is at least 3 times more than that, but also something that is not unnecessarily large. Therefore we estimated that our reasonable detection range would be at 4 meters.

For accurate sensor detection, given the detection range of 4 meters, we want the sensor to be accurate enough so we do not cause any harm to our user with false sensor data. In this case, we believe that an acceptable margin of error is around 2 %. In a 4 meter detection range, this means at most 8 cm error of detection. Within 2 meters, that error is limited to less than 4 centimeters, which is a distance that should not affect the means of the belt, and a quantification that matches the need of the user.

Regarding reliable response, we want to system to always be able to react on the correct scenarios. If the detection occurs, we want to see a valid reaction on the feedback system(the vibration motors) in the correct direction. This has to be accurate because if not, the user can be hurt. However, we do allow for more false positives, since giving the user more warnings is always better than giving the user no warnings at all. Therefore, we believe that we should contain our false positive responses to 10%, and our false negative responses to within 5%.

For the cost, our metric is based on a comparison to related market solutions. For our reference point, a smart assist walking cane can cost up to 599$ [14]. And guide dogs cost up to $50,000$[16]. Therefore a cost cap much lower than the existing solutions is something we want to achieve, and we estimate the maximum to be 500$

For real-time feedback, we quantified the real-time part by requiring the latency from sensing to feedback to be less than 0.1 second. This metric is based on the standards of human response time, where the visual response time is around 0.25 seconds, and the sensual response time is 0.15 second[6]. We want the belt to help blind users visualize their surroundings in time, so our metric is based on the difference in sensual response time and visual response time, which roughly estimates to around 0.1 seconds. If the belt can pass its information to the user through vibration in less than 0.1 seconds, we would achieve a user experience similar to how normal people react on visual information.

3 ARCHITECTURE AND/OR PRINCIPLE OF OPERATION

Our system can be divided to four subsystems: sensing, computation, feedback, and power supply. For the sensing subsystem, we use a depth camera for ground-level sensing and an array of ultrasonic sensors for above-ground sensing; data collected from the sensors are either sent to an Arduino Uno, where they are processed into more comprehensive data, before being sent to the Raspberry Pi, or directly sent to the Raspberry Pi . The Raspberry Pi models the surrounding, classifying objects and rating their threat level, and send feedback commands to the Arduino Uno, which drives the vibration motors in the feedback subsystem. The two ports of the power supply is connected to the Raspberry Pi(which also powers the Arduino) and the depth camera.

A block diagram of our system is shown in Fig. 1; physical images of our belt are shown in Fig. 2, 3 and 4.

4 DESIGN REQUIREMENTS

Based on the use case of alerting the visually impaired with real-time actionable feedback on a wearable form factor, we need to establish a set of requirements that inform product design and guide our decision making at every stage of development. Conceptually, this wearable device is powered by electricity to constantly run a sense and feedback loop so that users can rely on it for one outing where alerts also have to responsive. Hence, use case requirements of battery life and latency are to be in place for users to trust the device for everyday use. Based on these considerations, our quantitative design requirements are

1. Lightweight: $\sum w_{\text{component}} \leq 1\text{kg}$
2. Battery life: Battery power > 4,300 mAh
3. Resonable Detection range: Sensor detection range and depth camera detection range both $\geq$ 4m
4. Reliable response: False positive < 10%, false negative < 5% within 4m
5. Accurate sensor detection: sensor error rate < 2%
6. Real-time feedback: depth camera/ultrasonic sensor array detect-vibrate latency $\leq$ 0.1s

In terms of weight, our hardware components besides the belt include Oak-D depth camera (115g), Arduino Uno (25g), Raspberry Pi (46g), 6 HC-SR04 ultrasonic sensors (6x8.5g), 6 vibration coin motors (6x1g) and 1-2 Charmest portable battery (2x187g) which add up to 617g. We have picked these components to meet the required weight limit in mind.

For battery life, we need a battery that can support the consecutive power draw of all components for more than 2 hours. in Section 5.3, we discussed the minimum

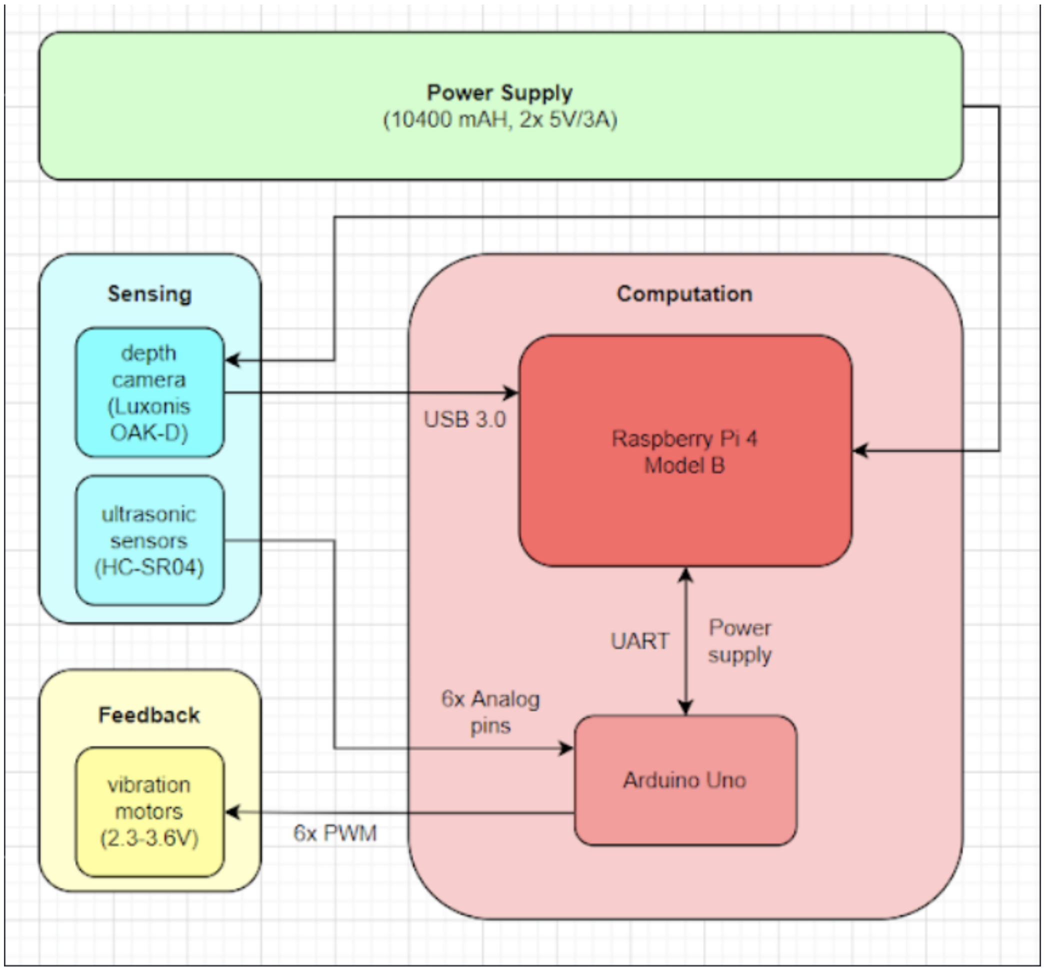


Figure 1: System Block Diagram.

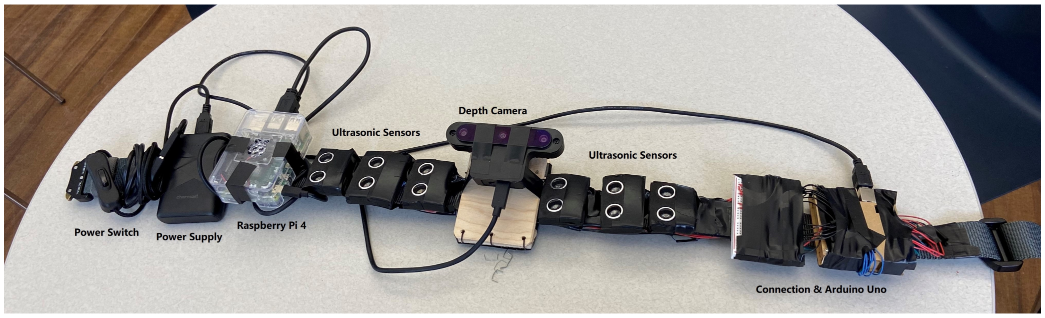


Figure 2: Front of the belt

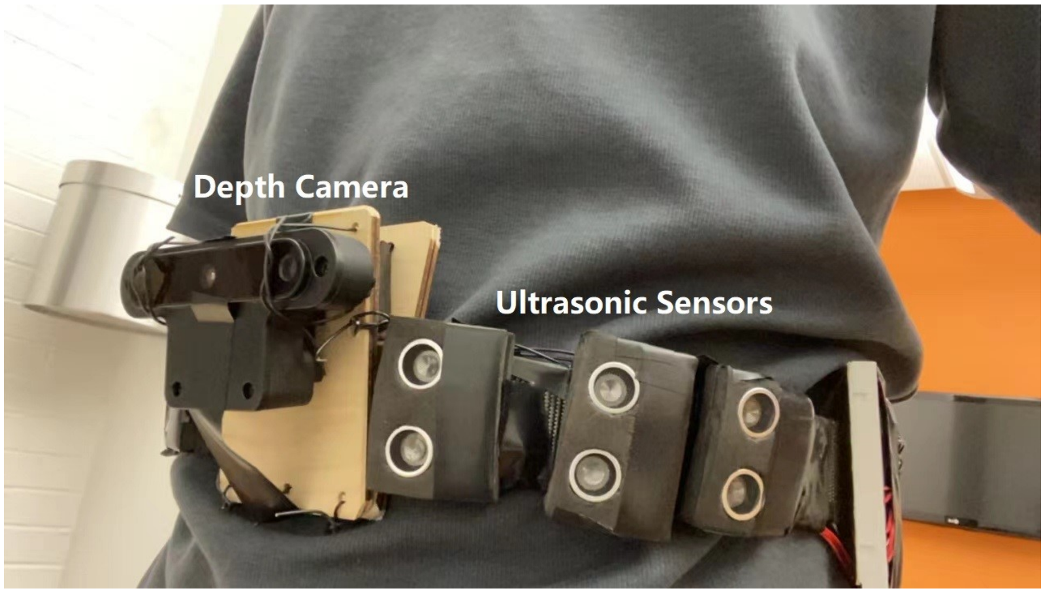


Figure 3: Belt when worn on the user's waist

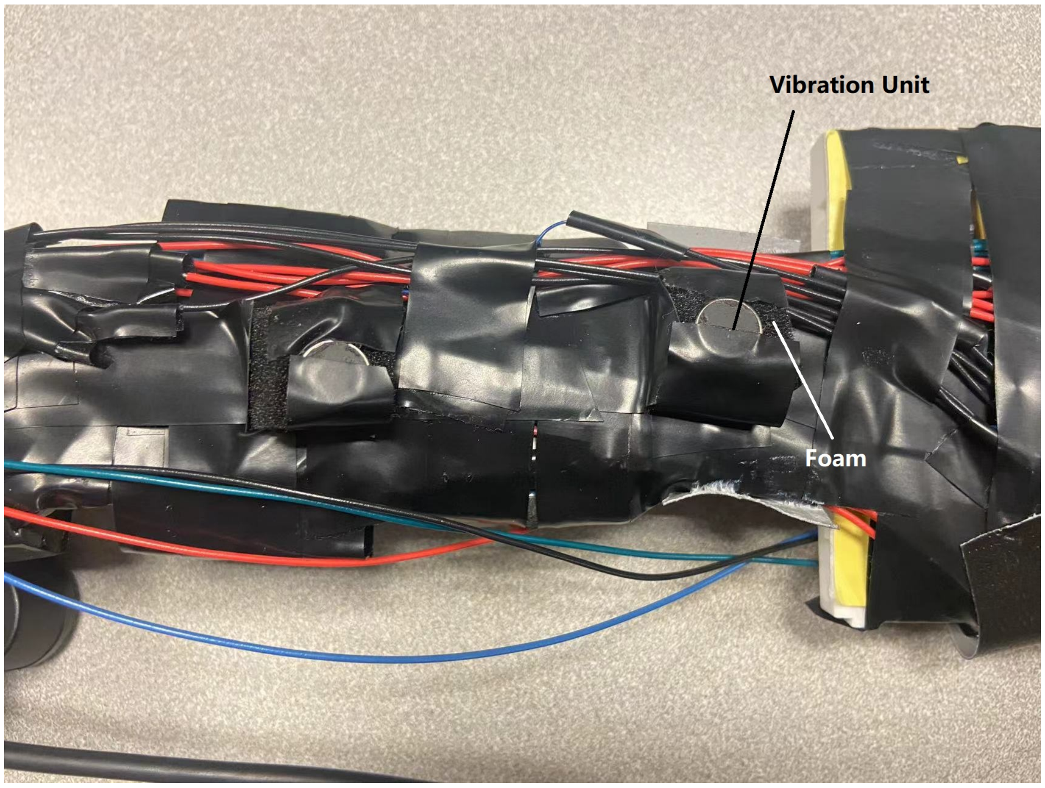


Figure 4: The vibration unit

battery requirement for our system to work consecutively for 2 hours, and the battery live requirement listed above is approximately 20% more than the minimum requirement.

To achieve reasonable range, we must have that our two information collecting components, the ultrasonic sensors and the depth camera, must have accurate detection at least up to 4 meters as stated in our use-case requirement.

Both reliable response and accurate sensor detection correspond to the reliable detection use-case requirement. Reliable response ensures that our software implementation is correct when an information is received. We need to make sure that when a threat does occur, our system is able to correctly reflect that threat and not miss out any information or have any problems when it comes to delivering the feedback. Accurate sensor detection is crucial to make sure that we are picking up the correct information. If our sensors are not accurate, then even if we do have the correct software system, we would not be able to deliver the correct feedback since we receive the wrong information to begin with. The quantification of these rates are related directly to the use-case quantification, our software system response rate should match the use-case response rate, and our distance error should be small enough to have little to no impact, for which we determined 2% to be a reasonable estimate of that parameter.

Since our system adopts two independent sensor system for ground- and waist-level detection (ultrasonic sensor and depth camera), we need to ensure that the detect-vibrate latency for both systems should be within 0.1s. We are aware that due to synchronization mechanisms, the detect-vibrate latency of the integrated system can still exceed 0.1s, but this requirement for both sensor systems are clearly necessary.

5 DESIGN TRADEOFF STUDIES

To control the cost of our product while satisfying the design requirements in Section 4, it is crucial that we carefully examine each subsystem. In this section we will discuss the trade-offs when we design the different subsystems, as shown in Fig. 1.

5.1 Sensing

Due to the different nature of above-ground sensing and ground-level sensing, we will discuss the trade-offs in two separate subsections.

5.1.1 Above-ground sensing

Unlike navigation of autonomous vehicles, which typically have their sensors rotating on the top of the vehicle bodies, Our design adopts multiple sensors facing slightly different directions for better coverage. Therefore, the key specifications we care about are measuring angle, maximum detection distance, and precision. Table 1 shows the 3 different distance sensors we have taken into consideration and their specifications. Comparing these data against the design requirements in Section 4, we chose the HC-SR04 ultrasonic sensors as our distance sensors.

5.1.2 Ground-level sensing

Compared to above-ground sensing, ground-level sensing have the following properties:

- Increased complexity. A change in the terrain, like potholes in the road, ramps, or stairs upward/downward, requires higher resolution and better classification methods.
- Lower Relative speed. Typical human walk speed is 5km/h or 1.4m/s, and ground-level threats are assumed to be mostly stationary.
- Smaller field of view required. Users are expected to walk mostly forward, so a smaller area in front of the user is sufficient.

Based on these properties, we have chosen the Luxonis OAK-D depth camera, with a 72° horizontal and 50° vertical field of view, 720P resolution, and individual Intel MyriadX chip for calculation of classification algorithms.

It should be noted that the inclusion of a depth camera with independent processing units places a great stress on power supply. We will discuss this in detail in Section 5.3.

5.2 Signal Processing & Computation

Limited by the form factor of wearable devices, we need a portable computational device with sufficient performance. We have adopted a Raspberry Pi 4 Model B with 2GB RAM for the computation, partly because it is readily available in the inventory. We have also considered using a Raspberry Pi Pico, which is smaller and less power hungry, but we eventually gave up because it does not support the communication interface and power supplies we need for various devices, especially the depth camera (see Section 5.1.2). However, the Raspberry Pi 4 is much more power hungry than a Raspberry Pi Pico. We will discuss this in detail in Section 5.3.

In order to better interpret signals from the ultrasonic sensors and drive vibration motors, We have adopted an Arduino Uno as a motor driver and an interface for signal processing. There are sufficient GPIO pins available on the board, so purchasing a more powerful board like Arduino Mega would be unnecessary.

5.3 Power Supply

As briefly discussed in Section 5.1.2 and 5.2, the depth camera and Raspberry Pi are the main contributor of power consumption. The typical working current of different parts

Table 1: Distance sensors

Sensor type	Model #	Measuring angle	Max distance	Precision	Frequency	Power	Cost
ultrasonic	HC-SR04	30°	4.5m	3mm	50Hz	10mW	$3
infrared	HiLetGo	35°	30cm	-	-	-	$0.9
LiDAR	TF-Luna	2°	10m	1cm	250Hz	350mW	$26

of our system is shown in Table 2. Given the battery life requirements in Section 4, we would need a battery of

$$(2 \times 6 + 900 + 50 + 800)mA \times 2h = 3,524mAh.$$

Combined with cost factor and the need of $5V/3A$ power supply for the Raspberry Pi, we decided to choose the Charmast Smallest USB-C Portable Charger with its 10,400 mAh battery and 2 5V/3A output.

Table 2: Power consumption of components

Parts	Working Current (typ. value, mA)
ultrasonic sensors	$2 \times 6 = 12$
depth camera	900
Arduino	50
Raspberry Pi	800

5.4 Haptic Feedback

Weight and form factor are the most important aspects of vibration motors. After researching some widely used vibration motors, we decide to use the linear resonant actuators (LRAs) because of their small form factor, light weight, and short response time. More details about vibration motors will be discussed in Section 6.5.

6 SYSTEM IMPLEMENTATION

Due to communication latency issues with pySerial (more in Section 6.2), We slightly modified our system design. In our current design, the Arduino is responsible for deciding the threat level on different directions.

6.1 Sensing

6.1.1 Above-ground Sensing

We have discussed in detail why we choose to use ultrasonic sensors in Section 5.1.1. In this section we will discuss how we integrate these sensors into our system.

We align an array of ultrasonic sensors in different directions to cover a total of 180°of area in front of the user. Directions of adjacent ultrasonic sensors will differ by approximately 30° apart to fully utilize the measuring angle of the sensors while minimizing the blind area. To avoid sensors interfering with each other, we choose to have sensors firing ultrasonic waves in turn. However, it takes time for the ultrasonic waves to travel before it is received by the sensor, so our current system can only achieve an update frequency of around 5Hz, which is lower than our goal of over 10Hz.

6.1.2 Ground-level Sensing

As briefly discussed in Section 5.1.2, we plan to use a Luxonis OAK-D depth camera for more sophisticated ground-level sensing. We position the depth camera slightly downward (about 35°downward from the horizontal line), so it can capture more details closer to the user.

The depth camera can generate a mapping of distance information in front of the user and update at a frequency of 30 Hz. From the mapping we can easily detect obstacles by comparing the depth mapping to an "ideal" model in which we assume the ground in front of the user is flat.

The depth camera has its own API, and requires the user to construct a pipeline to activace its internal modules. A diagram of the pipeline is show in Fig. 5.

The horizontal field of view (FOV) of the depth camera is around 72°, which is slightly larger than the FOV of two ultrasonic sensors combined. Therefore, we set the depth camera to be only responsible for updating the middle 2 vibration units, competing with 2 ultrasonic sensors in the middle of the sensor array.

6.2 Signal Processing & Computation

Our code flow is shown in Fig. 6. Some changes have been made since design review for this section. We initially planned to use a Raspberry Pi 4 Model B with 2GB RAM as our source of computation, and the Arduino serves as an ADC for ultrasonic sensors and driver of the vibration motors; however, during testing, we found that the pySerial we are using has extremely high latency (around 1 to 1.5s per update) under certain circumstances. Therefore, we moved the source of computation from the Raspberry Pi to the Arduino to improve the latency to an acceptable level.

The algorithm running on the Arduino Uno is as follows: For each cycle,

1. Collect data from ultrasonic sensors.
2. Calculate speed with current readings and historical readings.
3. Rate "threat level" of each identified obstacle by comparing them to pre-set distance and speed thresholds.
4. If there is a threat level update sent from the Raspberry pi. compare the corresponding threat levels and choose the maximum of the 2 sets of data.
5. Set vibration level to the vibration motors.

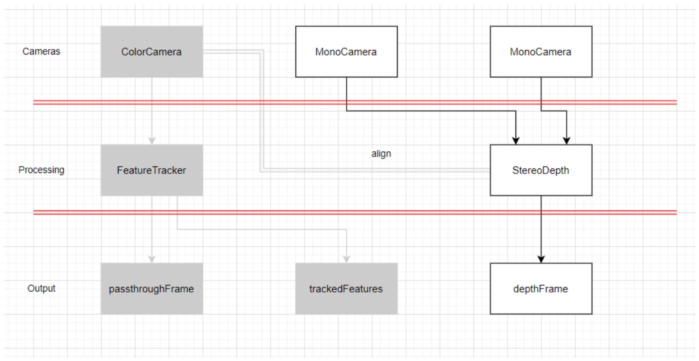


Figure 5: Pipeline design of the Luxonis OAK-D stereo depth camera. We initially implemented feature tracker (the grey blocks) for motion tracking, but it is eventually discarded due to difficulties in integration and a tight schedule.

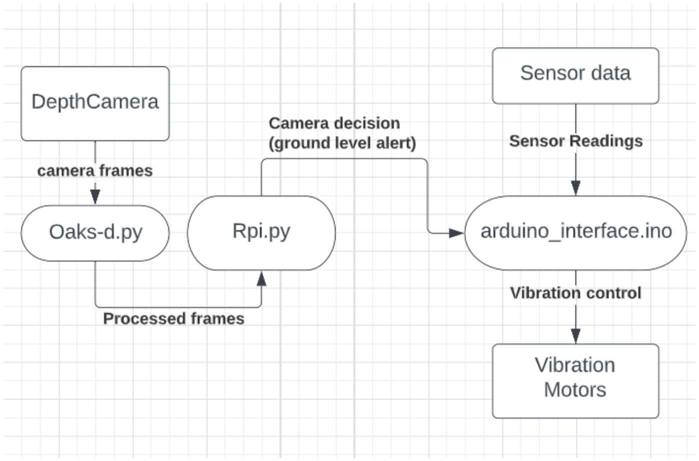


Figure 6: Flowchart of our code.

6. Update historical readings for future speed calculation.

The Raspberry Pi communicates with the Arduino through USB ports, using UART protocol with a Baud rate of 9,600 [13].

The algorithm running on the Raspberry Pi is as follows:

```
1. Request the most recent depth map from the depth camera. We call this frame the calibration frame.
2. Take the mid column of the calibration frame. This column represents the depth information of a thin band in front of the user, which we assume to be flat ground.
3. Use Ridge regression to get a model of the flat ground, and populate it across the frame to generate the base map. This is the depth map we expect to get if the ground in front of the user is flat and free of any obstacle.
4. begin loop:
5.     Request the most recent depth map from the depth camera.
6.     Use a hard threshold to determine the "threat level" of each valid pixel.
7.     Compare the depth map with the base map; if a pixel on the depth map has distance information very close to the pixel on the same location on the base map, we set the threat level of that pixel to 0.
8.     Count the number of pixels with threat level 0/1/2; if it passes a certain threshold, we raise the threat level to at least the corresponding level.
```

The visualization of the above algorithm is shown in the order of Fig. 7, 8 and 9.

The threat levels are set from Level 0 (safe) to 2 (most threatening) instead of 3 in the design report; the reasons are elaborated in Section 6.5.

6.3 Arduino circuit connection

Our Arduino subsystem controls both the sensors and the vibrators. We used the 6 analog pins(A0 - A5) as trigger pins for the sensors, the 6 PWM digital pins to control the vibrator motors, and the 6 remaining digital pins to read sensor echo. A circuit diagram is provided on page 10, see Fig. 10.

6.4 Power Supply

We have adopted a power bank with 10,400 mAh capacity and 2x 5V/3A output. Our current implementation plan is to power all other subsystems through the Raspberry Pi; however, we do notice that the OAK-D depth camera is also quite power hungry, so if the current design proved to be beyond the power supply capacity of the Raspberry Pi, we will directly power the depth camera with the power bank.

The power supply is probably our biggest failure throughout the project. Even though we have specifically purchased a power bank capable of 5V/3A output (most power banks in the market only support 5V/1A or 5V/2A output), and we choose a 10,400 mAh power bank which is theoretically sufficient (Section 5.3) to power the system for the expected battery life, we see a 50% decrease in battery life after a mere 2 hours. Worse still, we see a significant performance decrease in the Raspberry Pi when the battery life plunges to the ground, making the entire system dysfunctional. We did not see this coming because we did most of our implementation and testing with stable power supply attached. A more detailed discussion can be seen in Section 7.2.

6.5 Haptic Feedback

As briefly discussed in Section 6.2, we use 6 LRA coin motors as the feedback system. By controlling the output voltage through PWM pins, we can control the intensity of vibration from weakest (2.0V) to strongest (3.6V). We have planned to implement 3 levels of intensity to signal the user of the threat of certain obstacles; however, during testing, we find the vibration levels hard to differentiate when the PWM voltage is low. Therefore, we eventually implemented only 2 levels of vibration intensity to ensure clarity.

7 TEST & VALIDATION

For testing, we mainly conducted two classes of tests. The first class is verifying the design requirements. This includes:

- Weight test
- Battery life test
- Sensor accuracy
- System response rate
- Latency

For this class of tests, the main idea is to provide the system with controlled input and check if the output is as desired.

The second class is verifying our product's performance in the desired use-case, making sure that it will indeed help blind user's avoid obstacles while walking. For this test, we had our team members and invited 2 friends to try out the product in a controlled environment, and compared the performance to real life scenarios.

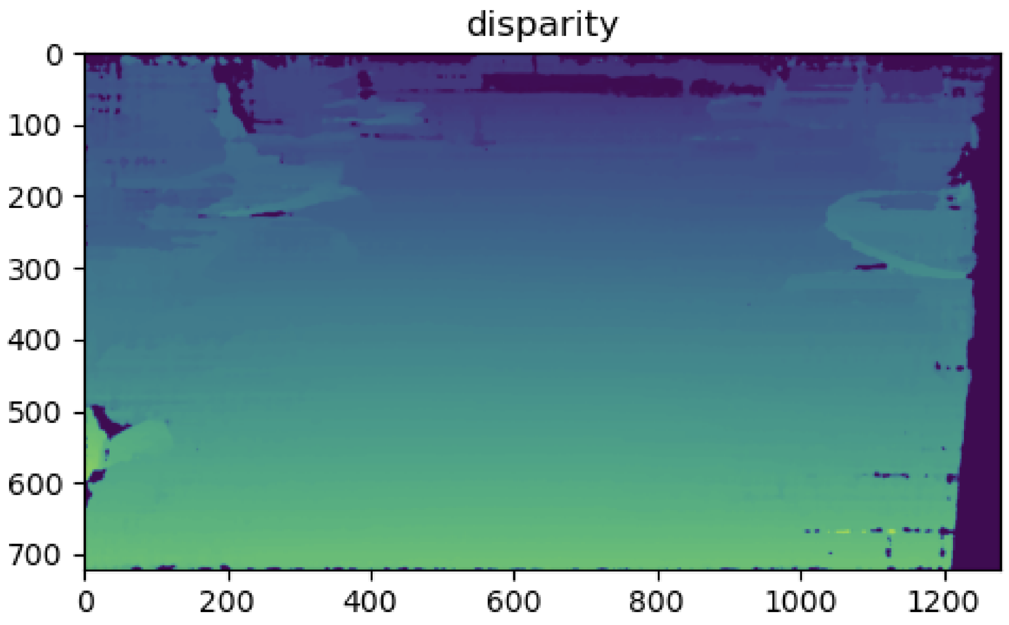


Figure 7: Depth map; the brighter the closer

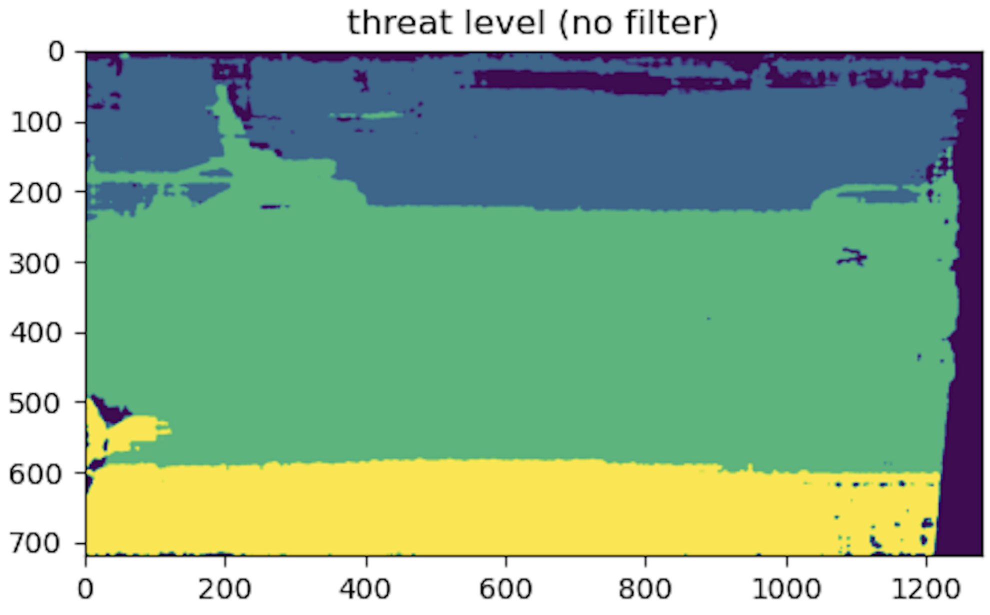


Figure 8: Threat level map without filtering ground; brighter the color, higher the threat

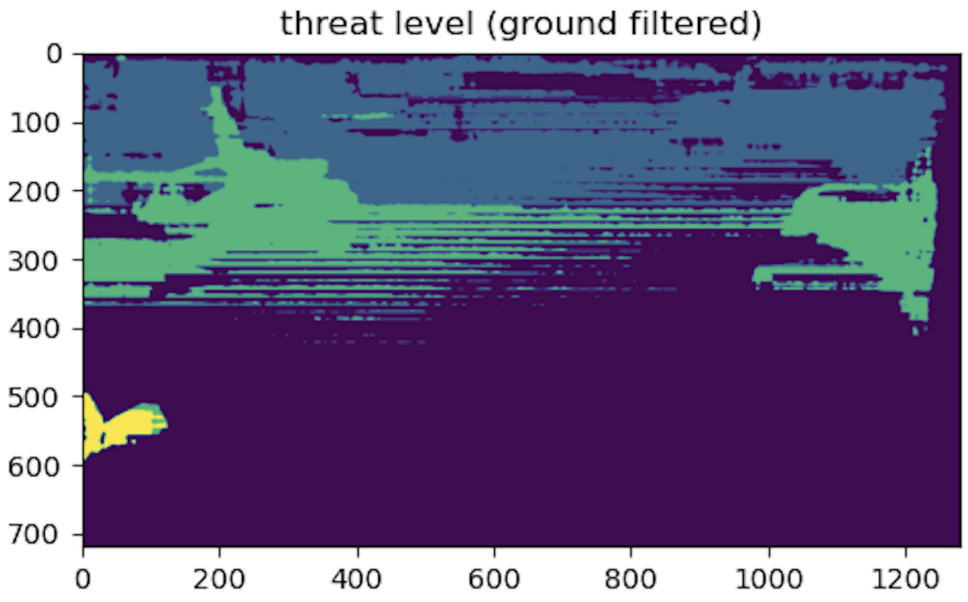


Figure 9: Final threat level map; ground is filtered and dark purple

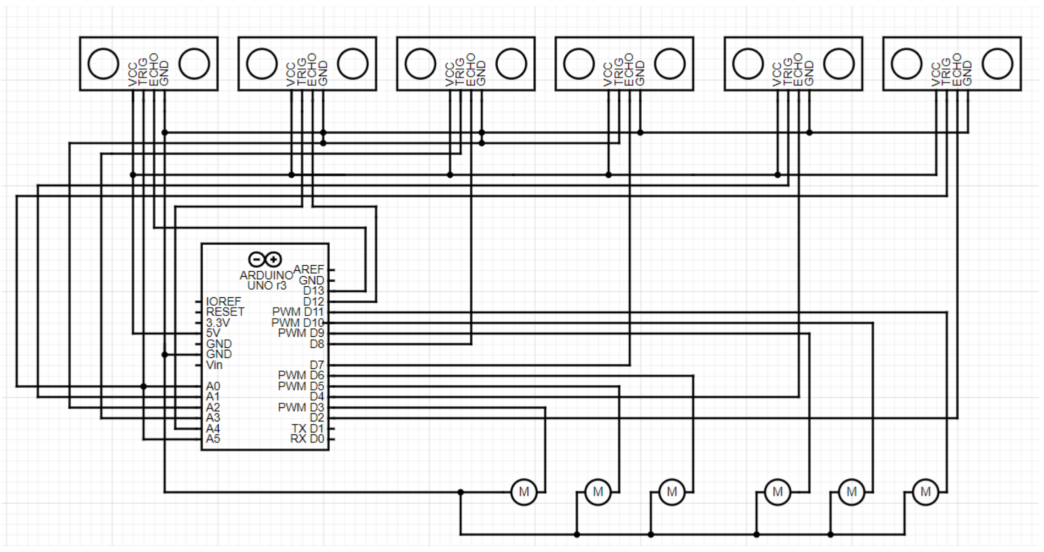


Figure 10: Circuit diagram.

7.1 Weight Test

We simply measured the weight of the belt on a precise scale, there is a small error range because when we hold the scale(to allow the belt to slope over it) our hands might not be perfectly steady:

Table 3: Weight

Requirement	$< 1kg$
Test result	1.025 kg (20g error range)

We ended up being around 25g heavier than what we had in our requirements. But we are still well below the weight of an utility belt and a 2% increase in weight should be insignificant to performance.

7.2 Tests for Battery Life

While we have calculated the expected battery life, we want to be able to test how well our product performs with the entire system intact. For this test, we tested the battery life in full operation mode. We internally set the system to turn on depth camera, all six sensors, and operate all six vibration unit at all times. This is the maximum power consumption for the system, and would give us an idea of how well the system runs . To conduct this test, one of our team members charged the power bank to full, and left the belt to operate while doing other work. He monitored the state of the raspberry pi through ssh to see when would something go wrong. Approximately 2 hours and 12 minutes later, the system reported that there is a lower voltage warning from the depth camera, at which point the main program terminated.

In our earlier design reports, we had planned to test on a daily life scenario for battery life. However, we soon realize that is not very meaningful for our design. The only difference in a real life scenario would be how often the vibration disk motors operate, which consumes very little power compared to the raspberry pi and the depth camera. And in a daily life operation, the power consumption rate depends on the user turning on and turning off the system by hand. Therefore we think having a minimum operating time is sufficient to estimate how well our system would run in daily life. We have since then re-evaluated and quantified our battery-life standard to 2 hours of operating time, as discussed in the Design Requirements section. Thus our system achieves the desired battery life.

Table 4: Operate time

Requirement	$\geq$ 2 hours operating time
Test result	2 hours and 12 minute operating time

7.3 Tests for Sensor Detection Accuracy and Range

For sensor detection distance, we want to make sure that the sensor components we use have desired accuracy and range when it comes to detecting obstacles. To conduct this test, we placed the belt vertically at a desired distance to a wall. We will measure that distance with a measuring tape so we have an accurate reference point for our tests. Then, at different distances, we will have the belt output 10 sets of sensor information to observe the differences between the belt readings and the measured distance. We completed this accuracy test for distances of 50 cm, 100 cm, 200 cm and 400 cm. This would then also give us the desired range because we are testing at 400cm, our specified maximum range of detection. In all of our tests, the maximum measured difference occurred at 400 cm, with an error of 5 cm. We divide the raw distance difference by the reference distance to compute our error rate, and use the maximum error rate over all tests at each distance as our maximum error rate. We were able to conclude that the overall error rate is capped at 2%, with specific error rates shown in Table 5:

Table 5: Sensor error rate

Requirement	$\leq$ 2% error
Test result (50cm)	$\leq$ 2%
Test result (100cm)	$\leq$ 1%
Test result (200cm)	$\leq$ 1%
Test result (400cm)	$\leq$ 1.25%
Overall result	$\leq$ 2%

7.4 Tests for System Response Rate

To conduct this test, we have decided to test out the depth camera subsystem and the sensor subsystem separately. This will help us understand which part of the system might need more improvements. The testing methods and results are discussed below:

- **Testing sensor response rate (stationary)** For this test, we want to make sure that our system always responds when the sensors conclude that we have an obstacle. And we also want to make sure that our system does not vibrate when the obstacle leaves the parameter of the warning threshold. Therefore to test for response rate, we had one of our team members put on the belt, and he would be holding a wooden stick. For each test, he would drop the stick down to the level of the belt at a certain direction, and see if there is a vibration in that direction. Then after a short period of time, he would raise the belt again and see if the vibration stops. We conducted 20 tests for each of the directions, giving us 20 test results for false positives (inserting) and 20 test results for false negatives (leaving). Results are shown in Table 8.

- **Testing sensor response rate (motion detection)** We also conducted a test of motion detection for the sensors. Since beyond a certain distance, we would only warn the user if the obstacle is approaching, we want to test out its response as well. For this test, a team member wore the belt and stood at around 2 meters from a wall. He then walks towards the wall for a small distance, and stops before he reaches the threshold where the belt will warn you regardless of motion. Once again, this provided us with two sets of information, one for observing vibration during the walk (false negatives) one for upon stopping (false positives). We conducted 20 sets of test for each of the 6 directions, Results are shown in Table 8.
- **Testing depth camera response rate**. For this test, we once again conducted insertion, (depth camera is only used to observe terrain difference, so motion test is not needed). Our teammate dropped a cardboard box to the ground before him, and waited to see if there is vibration. Then he turns 180 degrees to face flat ground, and see if the vibration stops. This gave us similar false positive and false negative tests as the sensor. We conducted this for 20 test runs. Results are shown in Table 8.

Table 6: Sensor False rate

	False positives	False negatives
Requirement	$\leq 10\%$	$\leq 5\%$
Sensor (stationary)	0%	0%
Sensor (motion)	5%	5%
Depth Camera	0%	0%

7.5 Tests for Sensing-Vibration Latency

For this test, we once again measured the subsystem latency separately, one for the depth camera, and one for the sensors. For the depth camera measurement, we simply had the system calculate the system time difference between receiving an image and producing the corresponding vibration. For the ultrasonic sensors, we had the system calculate the time difference between the start of every cycle of triggering sensor response, and producing the corresponding vibration. For the sensor latency, we would see a bigger fluctuation because sensor sound waves take time to travel and the time to retrieve the signal could vary. The results are shown in Table 7.

Table 7: Latency

Requirement	$\leq 0.1s$
Depth Camera	$\leq 0.147s$
Sensor	$\leq 0.183s$ (average 0.15s)

This is a higher latency then what we would have hoped for. The depth camera latency can be potentially improved with more processing power or a optimizing the algorithm a little more. And for the sensor data, the bottle neck comes primarily from our design choice of grouping all 6 sensor data in a cycle. Each sensor is limited by physical limits of the speed of sound, and to receive data within our defined range, we have a timeout of 0.03 seconds. As a result grouping 6 together in cycles means that each cycle takes up to $0.03 \times 6 = 0.18s$, which is close to what we see in the testings. There is still room for improvement on latency but for a prototype an average increase of $0.05s$ is a reasonable error.

7.6 Tests for MVP Use Case

For this test, we want to get an estimate of how well our product actually performs. We utilized an empty hallway at Wean, and measured out 15 meters as our track. We constructed 4 obstacles out of chairs and cardboard boxes, elevating the boxes to waist level. Every test run, one of team members, or one of our two invited users would be blindfolded and led to the start of the track. The other team members would place the obstacles along the track and ask the user to proceed to walk through the track. As the user walks on the track another team member would follow behind him to ensure his safety. We then measure the time taken to pass and obstacles hit. We observe the following results for our test runs:

Table 8: Sensor False rate

Average hits per run per obstacle	0.1
Average completion time	16.63 seconds
Average speed	0.903 m/s

Overall, we see very little actual hits because the test subjects are typically very careful when they walk, and only observed 8 small collisions out of all of our 20 tests. However, this also reduces the speed for our runs, average at around 0.9 meters per second. For reference, the average normal walking speed is around 1.4 meters per second, and we see a 35.8% decrease in speed. Compared to the collision statistics we see, we feel that this is a reasonable performance for our MVP.

8 PROJECT MANAGEMENT

8.1 Schedule

The detailed schedule is shown in Fig. 12. towards the end of the report. Generally, we were on track according to the schedule with the exceptions depth image modelling and physical assembly where we spent four weeks as opposed to the expected two weeks. For depth image modelling, we had multiple iterations. We first tried segmenting the image and calculate the mean squared error compared to a calibrated base map, but it proved impossible to set the thresholds right; we then tried to apply hard thresholds on the depth map and use the base map as a ground filter and succeded. We tried using a feature tracker for motion tracking, but eventually gave up due to difficulties in integration and a tight schedule. Meanwhile, the physical assembly was delayed due to realizing the need of more parts such as vibration motors, protection foams, and different connection cables, as practical problems arise(e.g. vibrations interfere with each other when attached to belt directly) or parts(vibration motors with extremely thin wires) break midway.

8.2 Team Member Responsibilities

Xiaoran Lin:

- Arduino interface code
- software and hardware integration for sensor and vibration unit.
- Python code for raspberry pi and arduino serial communication
- Battery and circuit connection

Ning Cao:

- Depth Camera Integration with Raspberry Pi
- CV code for depth camera obstacle detection
- Potential obstacle classification for depth camera

Zhuoran Zhang:

- Raspberry Pi integration
- Raspberry Pi python code for threat level processing based on arduino and depth camera input
- Managing schedule and materials
- Belt physical form management

8.3 Bill of Materials and Budget

Our table is shown below in table 3 9.

8.4 AWS Usage

While we applied for AWS credit, we ended up using none of it. We originally applied for this credit in case we want to upload our history data to the cloud for more analysis. But we soon abandoned that feature because it was out of the scope of what we had planned.

8.5 Risk Management

Our main risks include the following

- **Sensor interference:** Since we are operating multiple ultrasonic sensors, one of our main risk is interference between the sensors. To handle this risk, we simply ran our sensors in a cycle, so that no two sensors would be operating at the same time. Our initial idea is to increase operating rate was to operate two relatively far sensors at the same time in a cycle, but we tested out the idea and there might still be interference for large items or close up objects, so we ended up operating our sensors one at a time.
- **Increasing sensor data rate**: Our sensors relied on the sound wave traveling to the target and later coming back to detect distance. And waiting for the full response is often a waste of time that seriously impacts our data rate. We resolved this by setting a hard limit (30 ms) on the sensor, so once the detection range is beyond what we wish to achieve, we timeout directly to operate at a faster rate.
- **Arduino Pin management:** Since we are utilizing so many sensors and vibration unites, we might run into the case where we do not have enough operating pins on the Arduino Uno model. From the design report, a plan was formulated to use multiplexers for sensor control. However, we soon realized that aside from the digital pins, we can also use the analog pins on the Arduino for digital write. So we now have 6 PWM pins, 6 digital pins, and 6 analog pins we can use, which happens to be just right for our system, as indicated by our current circuit connection (Fig. 10).
- **Serial communication latency:** As discussed in Section 6.2, during our development process we realized that bidirectional serial communication can be extremely slow, which becomes a bottleneck for our system. As a result, we made major adjustments from the design review. Instead of using the Raspberry Pi as our center control unit, the Raspberry Pi now only operates to process depth camera information into corresponding ground level threat detection. Afterwards, the information is sent one-way through serial into the Arduino for vibration motor control. The code that processes sensor information is now moved into the Arduino. This way, we can now avoid bidirectional communication and make our system much faster.
- **Getting real-time system visualizations:** Following up on the last risk, we have removed bidirectional serial communication from our implementation. This gives rise to a new problem, where we can not grab information from the Arduino to monitor our changes or demonstrate how our system works. In order to do so, we have added a feature to the Arduino code, so we can control whether we want the Arduino to report sensor and vibration motor data. We have also developed a python code written in tkinter to graph these data in real time. This was done

Table 9: Bill of materials

Description	Model #	Manufacturer	Quantity	Cost @	Total
Depth Camera	Oak-D	Luxonis	1	$199	$199
Raspberry Pi	Pi 4 2GB	Canakit	1	$139.99	$139.99
Battery	Smallest 10000mAh USB-C Portable Charger	Charmast	1	$23.39	$23.39
Arduino Uno	REV3	Arduino	1	$22.77	$22.77
Ultrasonic Sensor	HC-SR04	Smraza	6	$3.198	$19.188
Belt	Tactical Belt	FAIRWIN	1	$14.99	$14.99
Raspberry Pi SD Card	Pi 32GB Preloaded (NOOBS)	Raspberry Pi	1	$9.9	$9.9
Vibration coin motor	MIniVibration Motor For Mobile Phone Bluetooth	Tegg	6	$1.17	$6.99
					$458.988

deliberately in tkinter rather than existing modules such as matplotlib because existing graphing tools can not achieve the data rate we want, whereas in tkinter we can optimize our code to work specifically for display purposes.

- **Vibration motor disk fragility:** Since the low voltage vibration motor disk we bought were small models meant for small stationary projects, their connection lines were extremely thin, which means that when we attach them to our wearable belt, the connection lines were very easy to break and the motors could lose connections. To account for this, we soldered and taped their lines so that they would be more protects. While this does not guarantee that they would be strong enough, we have also wired the vibration motors so that they are on the surface of the belt circuits. So in case we need to replace them, we can do so without too much trouble.

9 ETHICAL ISSUES

Ethics is crucial for this project as it is designed to ensure physical safety for visually impaired users. It is unethical to release a product that does not reliably meet the expectations communicated to the user for everyday use. With this belief, we think that our current system is not ready for the market for two main reasons. First, the system as a whole is not fast enough in computing feedback based on sensing. Second, the system has not gone through sufficient user testing in miscellaneous scenarios. For example, an object going from one ultrasonic sensing sector to another among the 6 could confuse the system with a speed spike and result in an overly heightened vibration alert for the user. This falls under the error of false positive, and false negatives can have more severe consequences. These two categories of errors are to be discussed further below along with latency.

9.1 Latency

The latency exceeded the requirement of under 0.1 second from sense to feedback. With the current worst case latency of 0.18s and average latency of 0.15s, the user could get hurt when alerted by a fast moving object whose presence should be communicated sooner.

9.2 False negatives

False negatives leads to our system not warning the user of existing obstacles. We find this issue much more dangerous than any other issues because it is directly linked with the safety concerns of the user, who may be misled into serious safety threats. There might be safety threats to other people in proximity or potential economic loss, but we place a heavier emphasis on the safety of the user above anything else since they are at a disadvantage in the first place.

We have considered two ways of mitigating the false negative issues:

1. We can ask the user to use our system along with a white cane, so the ability of sensing ground-level threats are guaranteed to be preserved. However, there is a conflict of functionality between the white cane and our depth camera.
2. We can set a low false negative rate. This is the current method we employed; however, we can never guarantee a 0% false negative rate, so we can limit the possibility of the safety issues at best.

9.3 False positives

False positives leads to our system warning the user of non-existent obstacles. A false positive is much milder than false negatives, because even in the worst-case scenario it confuses the user and impedes their movements. We consider this to be a minor issue since it is better safe than sorry; this is also part of the reason our goal for false positives is much lower than false negatives.

10 RELATED WORK

During ideation, our team has extensively researched technical products currently available in the market with the same value proposition of helping the visually impaired navigate. A capstone at SUTD, prototype built by MIT CSAIL, and startup product are discussed and linked below for reference.

- MIT CSAIL Lab research project : Device provides information from a depth camera, via vibrating motors and a Braille interface. The depth camera with an on-board computer is hung around the neck at chest level while a separate vibration belt is wrapped the user's waist. The researchers opted for Braille interface instead of audio feedback because blind people rely heavily on hearing. The system can recognize objects and use braille pads to inform user, e.g., a "t" for "table" or "c" for "chair.". This system has not seen application for 5 years after initial press and we believe the Braille interface presents too much cognitive load in a real-time environment while the separate belt and neckwear add communication challenges. [7]

- Strap Tech startup product Ara : A round pod strapped at the chest, which presents a system we think is best among the three for the following features. First, it is able to detect obstacles at head, chest, and below waist levels. It also covers a wide range of obstacles from poles, stairs, to holes and tree branches. Furthermore, Ara helps the user to both detect obstacles and walk in a straight line with its own haptic feedback language that produces a sequence of vibrations specifc for each situation (obstacle detection and straight line navigation). The system includes 4 Light Time-of-Flight, 5 ultrasonic sensors, an accelerator, a gyroscope, 6 vibrators at the pod and 4 along the strap. Priced at 500 USD while its most expensive component of Light ToF sensor looks like a model around 100 USD, the product is questionable in terms of affordability. And the product is still in pre-order stage after 3 years of development. [8]

- SUTD Capstone Project N'Able : The system is a neckband with two ultrasonic sensors at the front to detect obstacles and provide audio as well as vibrational feedback to user. A highlight feature is toggleable modes for selecting audio and/or vibrational feedback. In terms of user experience, it is more comfortable than the MIT CSAIL project as its weight is evenly distributed around the user's neck instead of strapped. However, the system is limited in that it only handles above waist level obstacle detection with merely two ultrasonic sensors. Meanwhile, the wearable is conspicous and not aesthetic as a neckband. Additionally, the cost as well as quantitative results of the system are unknown. [9]

11 SUMMARY

11.1 Future work

In the future, we can improve our product in the following directions:

- Due to the limited computational power of Raspberry Pi, we limited our motion tracking to only recording one set of historical data. We can add more historical data records for more accurate motion tracking.

- During implementation, we have planned using feature trackers for motion tracking of the depth camera. However, this idea was eventually discarded due to difficulty in integration and limited time schedule. In the future when the schedule is more relaxed, we can resume the integration of feature trackers in the depth camera pipeline. This can also tap into the depth camera's built-in compute instead of transmitting whole image frames to raspberry pi for modelling, and thus reducing the latency.

- Audio feedback could provide rich information about the environment such as the types of obstacles and suggestions on how to step over or walk around obstacles. Currently, we have tested the feature's feasibility by outputting audio from Raspberry Pi to headphone through Bluetooth connection but have not fully built out an audio feedback pipeline. Capturing more information and feedback through audio is a functionality worth considering looking forward.

- Currently we are using only 1 set of hard-coded thresholds. These threshold values for speed and warning distances are fine-tuned to meet the requirements of an indoor, low-speed environment (many close obstacles with low speed). In an outdoor scenario (less obstacles, further away with high speed), the belt would not react well. On the other hand, if we had tuned the threshold to outdoor environment, the system would likely provide too much warnings. in the future, we can devise multiple sets of thresholds for different scenarios or even allow the user to adjust the thresholds themselves.

11.2 Lessons learned

- We distributed our work to different team members to accelerate the implementation of the system, but the integration took much longer than expected, and even leads to multiple rounds of design iterations in the process. We have learnt that integration of different subsystems are sometimes much more difficult than implementing them.

- We found difficulties quantifying and/or visualizing certain data due to sub-optimal design, and we wasted much time on adjusting the designs. We have learnt that when designing the system, we should keep ease of testing and demonstration in mind instead of focusing on the functionalities only.

- We have learnt that Python is NOT a fast language.

References

[1] Castro Teixeira, Denilson & Hernandes, Nidia & Probst, Vanessa & Ramos, Ercy & Brunetto, Antonio & Pitta, Fábio. (2012). Profile of physical activity in daily life in physically independent elderly men and women. Revista Brasileira de Educação Física e Esporte. 26. 645-655. 10.1590/S1807-55092012000400009.

[2] BASSETT, DAVID R. JR.1; WYATT, HOLLY R.2; THOMPSON, HELEN2; PETERS, JOHN C.3; HILL, JAMES O.2 Pedometer-Measured Physical Activity and Health Behaviors in U.S. Adults, Medicine & Science in Sports & Exercise: October 2010 - Volume 42 - Issue 10 - p 1819-1825 doi: 10.1249/MSS.0b013e3181dc2e54

[3] Croxton, K. (2018, August 26). Haul of duty. Burlington Times News. Retrieved May 7, 2022, from https://www.thetimesnews.com/story/news/politics/county/2018/08/26/haul-of-duty-how-much-does-burlington-polices-gear-weigh/10955541007/

[4] W, A. (2021, November 24). How much weight should you carry backpacking. Exploration Solo. Retrieved May 7, 2022, from https://explorationsolo.com/how-much-weight-should-you-carry-backpacking/

[5] Code base of the project:
https://github.com/Kelton8Z/the_Bat_Belt

[6] Human response time. Experiment: How fast your brain reacts to stimuli. (n.d.). Retrieved May 7, 2022, from https://backyardbrains.com/experiments/reactiontime

[7] MIT CSAIL research project
https://techcrunch.com/2017/06/02/mit-develops-a-vibrating-wearable-to-help-people-with-visual-impairments-navigate/

[8] Strap Tech startup
https://strap.tech

[9] SUTD Capstone Project N'Able
https://capstone2021.sutd.edu.sg/projects/n-able-a-navigational-wearable-for-the-blind-and-visually-impaired

[10] Five Ways To Run a Program On Your Raspberry Pi At Startup
https://www.dexterindustries.com/howto/run-a-program-on-your-raspberry-pi-at-startup/

[11] Python and Bluetooth – Scanning For Devices And Services
https://geektechstuff.com/2020/06/01/python-and-bluetooth-part-1-scanning-for-devices-and-services-python/

[12] Oak-D camera's API reference of python example using stereo depth
https://github.com/luxonis/depthai-python/tree/main/examples/StereoDepth

[13] Raspberry pi arduino serial communication - everything you need to know - the robotics back. End. (2021, November 15). Retrieved May 7, 2022, from https://roboticsbackend.com/raspberry-pi-arduino-serial-communication/

[14] Wewalk cane. WeWALK Smart Cane. (2021, June 29). Retrieved May 7, 2022, from https://wewalk.io/en/about/

[15] T. B. C. Team, "How to pick the correct length of a white cane for a blind person?," BAWA Cane, Nov. 15, 2018. https://www.bawa.tech/blog/pick-the-correct-length-of-a-white-cane/

[16] [1]Colby, "How Much Does A Guide Dog Cost?," Puppy In Training, Apr. 03, 2019. https://puppyintraining.com/how-much-does-a-guide-dog-cost/

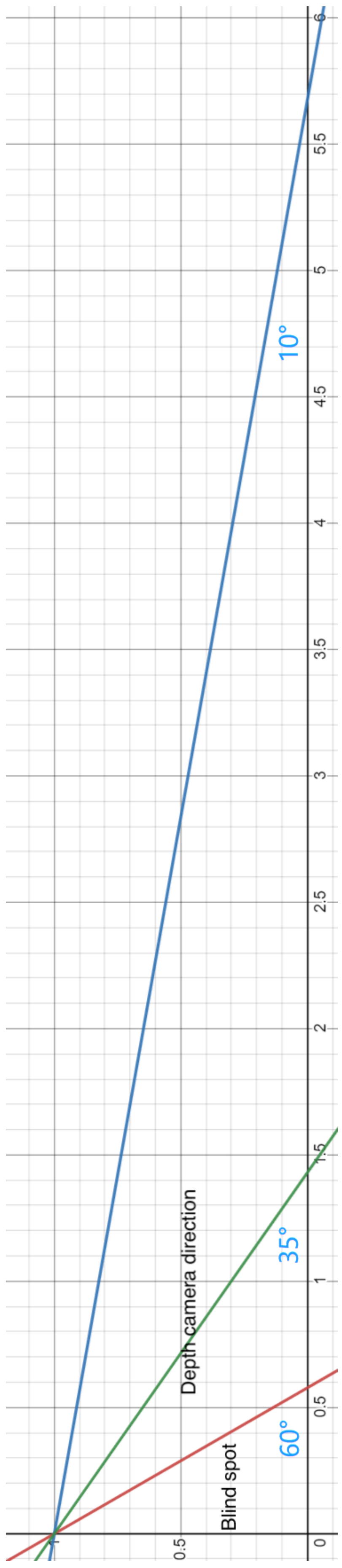
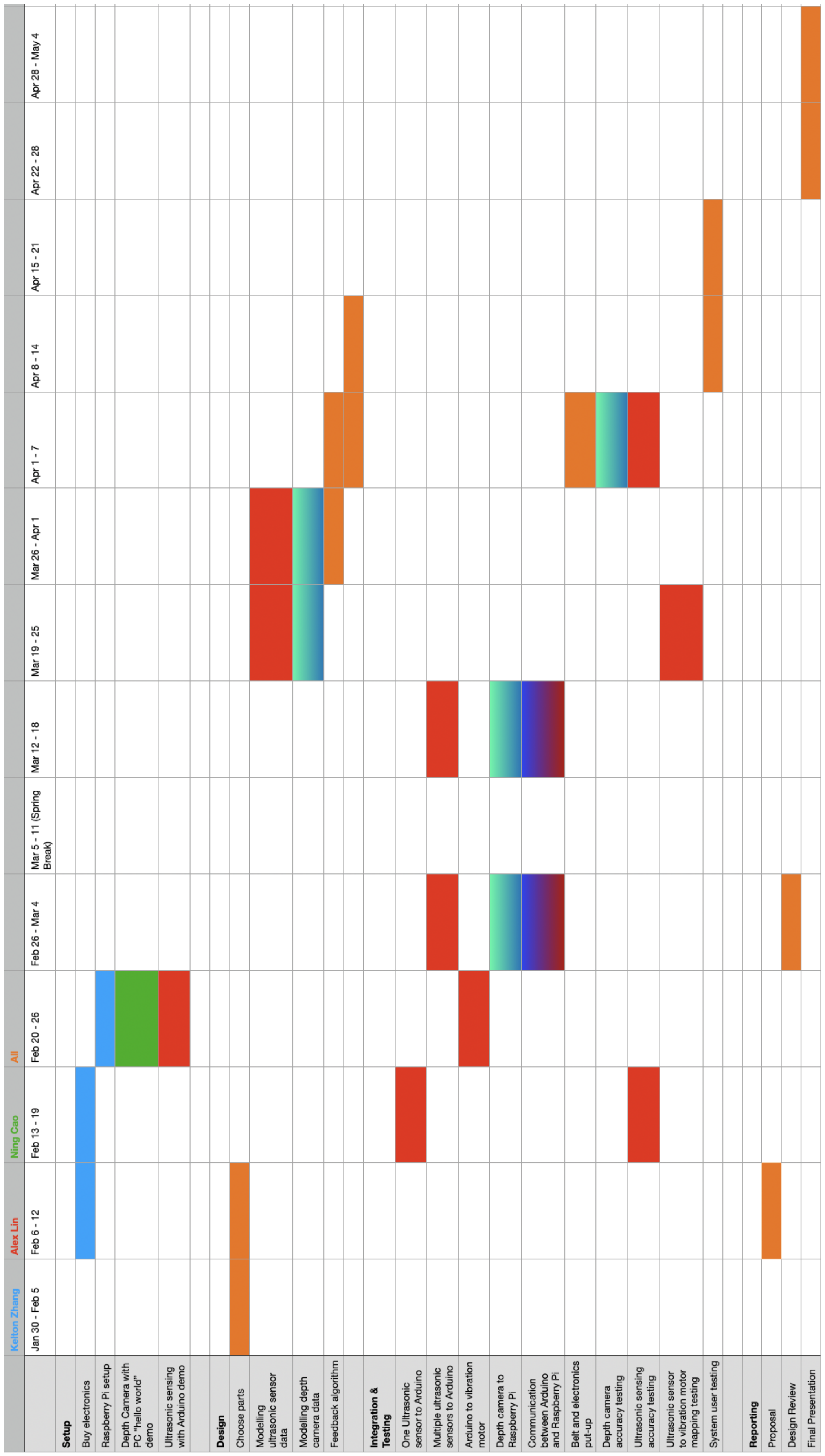


Figure 11: Depth camera cover area



🟦 Kelton Zhang 🟥 Alex Lin 🟩 Ning Cao 🟧 All

	Jan 30 - Feb 5	Feb 6 - 12	Feb 13 - 19	Feb 20 - 26	Feb 26 - Mar 4	Mar 5 - 11 (Spring Break)	Mar 12 - 18	Mar 19 - 25	Mar 26 - Apr 1	Apr 1 - 7	Apr 8 - 14	Apr 15 - 21	Apr 22 - 28	Apr 28 - May 4
Setup														
Buy electronics		🟦	🟦											
Raspberry Pi setup				🟦										
Depth Camera with PC "hello world" demo				🟩										
Ultrasonic sensing with Arduino demo				🟥										
Design														
Choose parts	🟧	🟧												
Modelling ultrasonic sensor data								🟥	🟥					
Modelling depth camera data								🟩🟦	🟩🟦					
Feedback algorithm									🟧	🟧				
										🟧	🟧			
Integration & Testing														
One Ultrasonic sensor to Arduino			🟥											
Multiple ultrasonic sensors to Arduino					🟥		🟥							
Arduino to vibration motor				🟥										
Depth camera to Raspberry Pi					🟩🟦		🟩🟦							
Communication between Arduino and Raspberry Pi					🟦🟥		🟦🟥							
Belt and electronics put-up										🟧				
Depth camera accuracy testing										🟩🟦				
Ultrasonic sensing accuracy testing			🟥							🟥				
Ultrasonic sensor to vibration motor mapping testing								🟥						
System user testing											🟧	🟧		
Reporting														
Proposal		🟧												
Design Review					🟧									
Final Presentation													🟧	🟧

Figure 12: Gantt Chart