# Detailed Design Documentation

A Haptic Guidance System for Blind Individuals to Reach Points of Interest

MSE 4499

March 1st, 2021

Benjamin Boyd – 250918222

Aidan Bruneel – 250905188

Alexander Noens – 250878261

Julia Rybiak – 250883926

Marina Silic – 250906639

# Table of Contents

# Introduction

Life as an individual with a severe vision impairment presents many day-to-day challenges. These challenges often manifest as a struggle to locate places and features present in their surroundings, particularly when they are in an unfamiliar environment. People have tried to reduce this burden through various means such as white canes, guide dogs, tactile maps, and other assistive devices [1] [2].

During an interview with Dan Maggiacomo [3], Principal of W. Ross Macdonald School for the Blind in Brantford, Ontario, he posed us some difficult questions: How would a completely blind individual locate the urinal without touching it? What about their locker? This inspired us to solve these tough questions, and ultimately led to our goal statement.

The goal of our design is to help users find points of interest within their immediate vicinity without requiring them to touch it, and without the use of auditory feedback. We accomplished this with a design comprising of a vision system to locate the points of interest, and a wearable haptic feedback device to inform the user of the relative location of these points. The vision system uses a phone, equipped with a rear-facing camera, to scan their immediate surroundings. When a marker corresponding to the desired location is detected, directional information is sent via Bluetooth to the haptic system, which then vibrates in a pattern that informs the user of its position relative to them. As the user approaches the marker, the haptic system continues to update the user of its location until they are within an arm’s reach (i.e., 0.75 meters away) [4].

# Design Analysis

## Vision Subsystem

### Image Processing and Analysis

To extract images from a video feed for processing and analysis of the user's environment, the core library and contribution modules of OpenCV 4.5.1 (an open-source computer vision library for C++) [5] were built using CMake 3.19.2 [6] in Visual Studio 2019 [7]. One of OpenCV's additional contribution modules, ArUco [8], allows configurable dictionary encoding of up to 1024 ID integers via square fiducial markers represented by an $(n \times n)$-bit grid surrounded by a 1-bit black border (as seen in *Figure 1: IDs 0-188 of ArUco 4x4_1000 Marker Dictionary*).

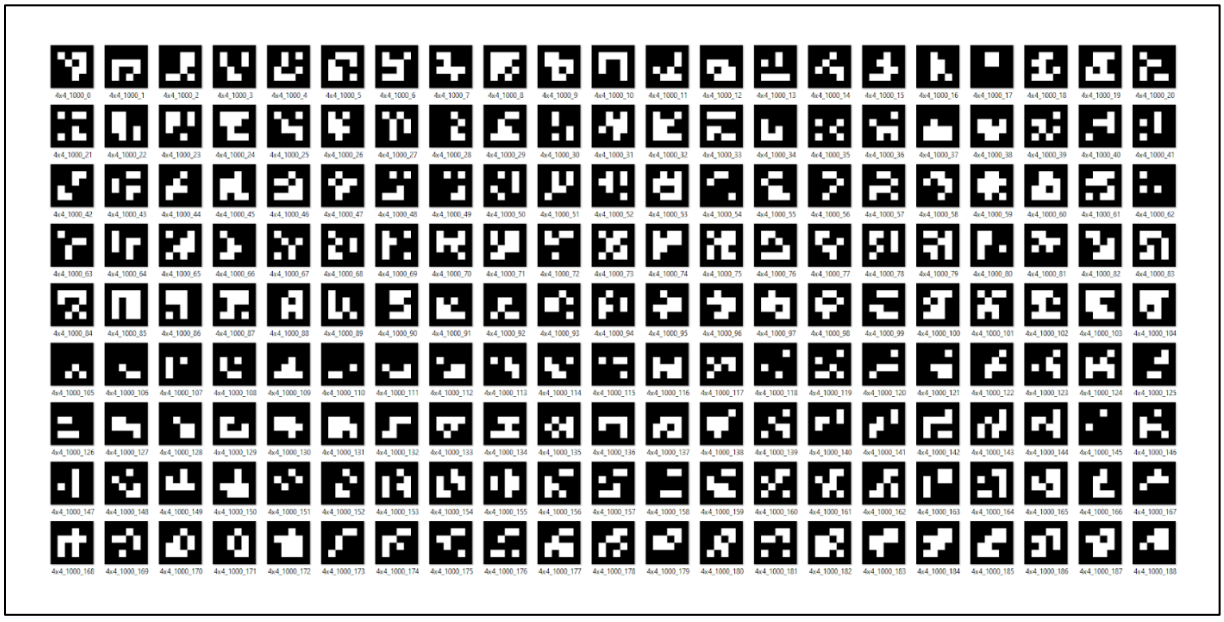


*Figure 1: IDs 0-188 of ArUco 4x4_1000 Marker Dictionary*

By using a camera's video feed to extract grayscale images, these ArUco markers are located in real time by segmenting contours of the black border via local adaptive thresholding and polygonal approximation, and decoded for their unique ID integer via homography and grid binarization [9]. The four prominent corner points of a detected marker are used to estimate the camera's pose [10], given that the camera's intrinsic parameters are known through a prior calibration process (as discussed in Design Documentation: Camera Calibration Instructions). The extrinsic parameters (i.e., the camera's pose) are found using the following linear system [11]:

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

2D Points | Intrinsic Parameters | Extrinsic Parameters | 3D Points

Table of Contents

## Pose Estimation - Distance Accuracy Study

With a calibrated camera, the accuracy of the estimated pose's Euclidean distance for different styles and sizes of ArUco markers is used as a benchmark to inform which printed marker format will best meet design constraints. The pose estimation from single ArUco markers and 3x3 ChArUco Diamond markers - both printed with varying side lengths of 10cm, 15cm, and 20cm - were recorded at 0.5m intervals between 0.5m and 8m. Printed markers were placed on a music stand. The camera was secured by a tripod and had its optical axis aligned with the centre of each marker at each distance (as measured by a measuring tape). This testing setup was adopted to ensure that the distance between the camera and the marker was the only independent.

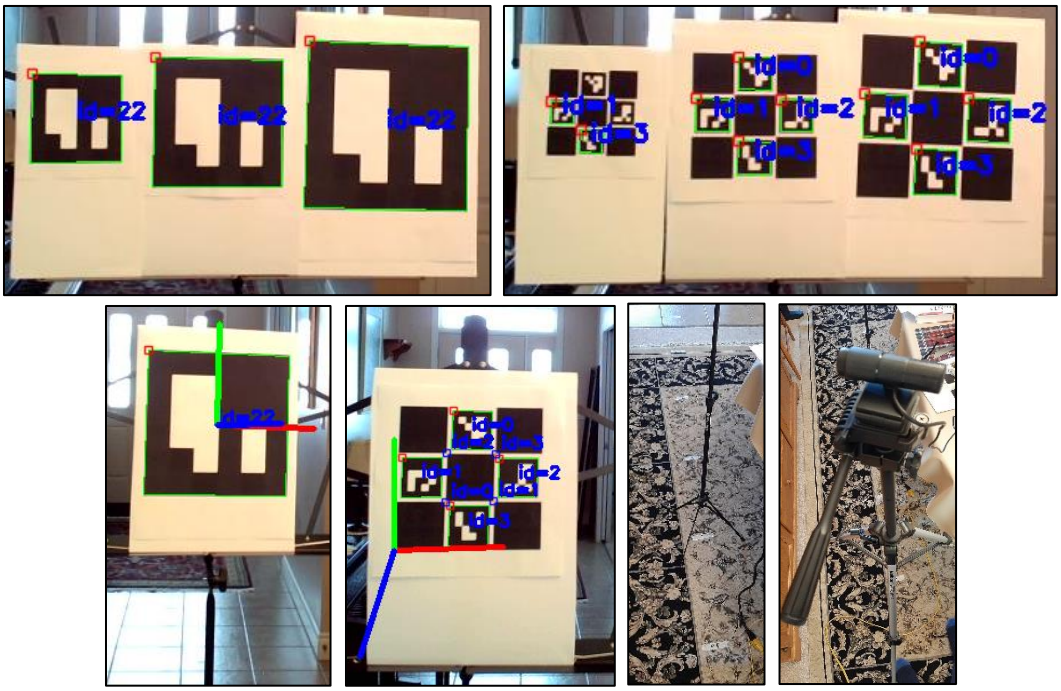


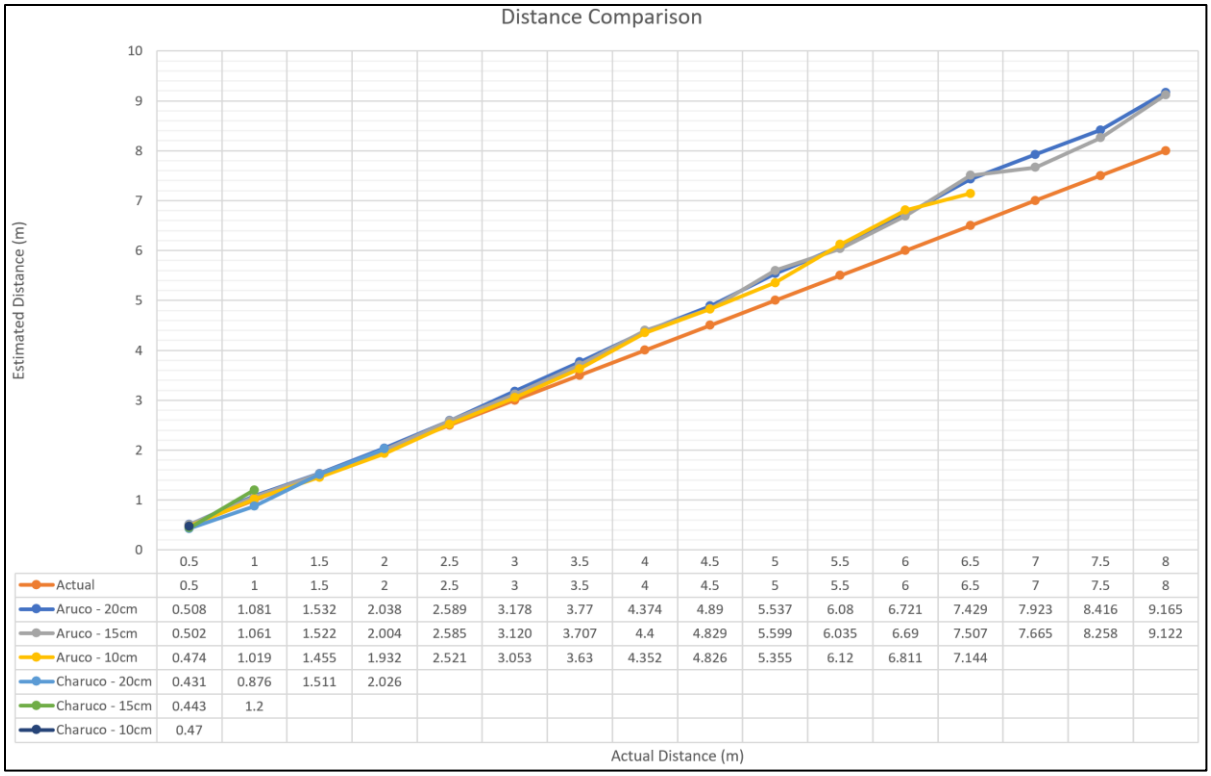

*Figure 2: Distance Accuracy Testing Setup*

The results of this study are in *Figure 3: Distance Accuracy Data*.

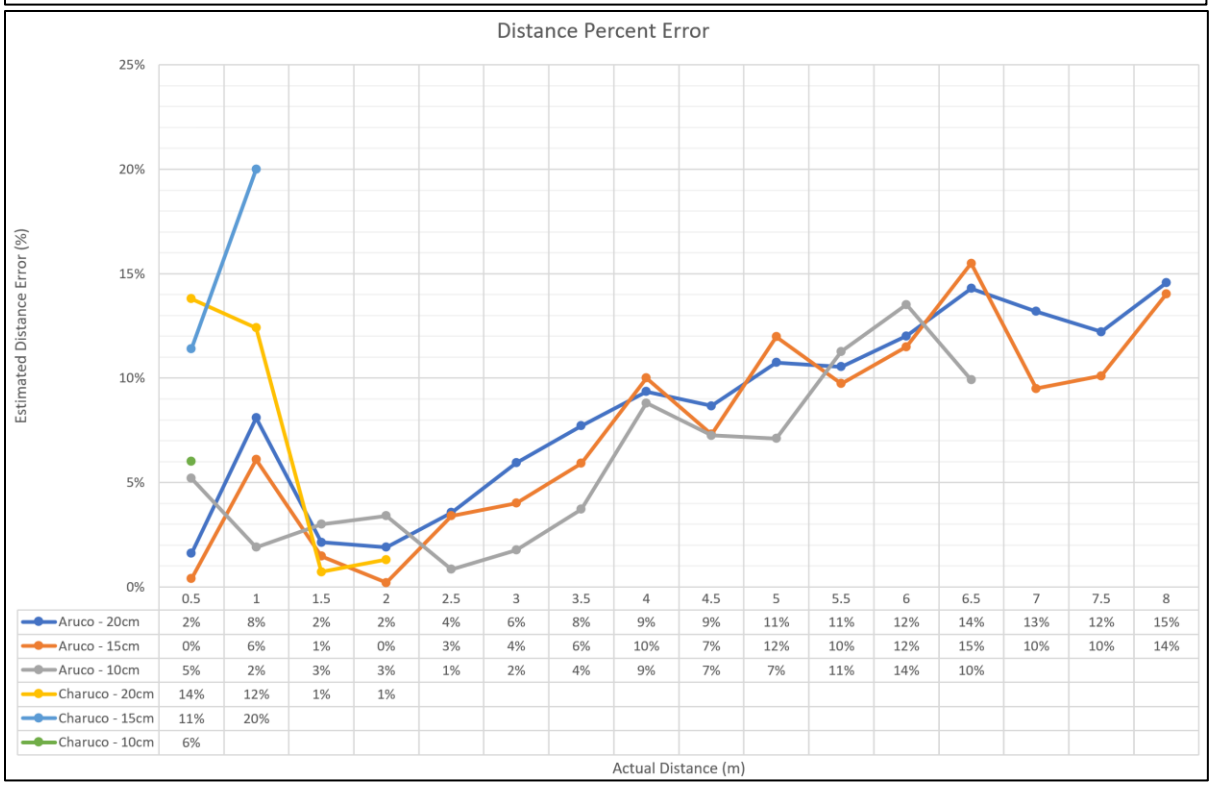
The data shows that ChArUco markers are not detectable past 2m, which is not a sufficient range to meet the design constraint of 5m scanning range. Therefore, single ArUco markers are used instead. For single ArUco markers of any size, there is up to 15% error when the marker is 8m away from the camera. As the marker gets closer, the error approaches zero. In practice the user should get within 0.5m of the desired object, which meets the design constraint of 0.75m, or the average arm length.

The optimal marker selected for our design was the 15cm single ArUco marker. The 15cm and 20cm markers could be recognized from the furthest tested actual distance (8m) with similar accuracy profiles. To print the 20cm marker, the margins of the page had to be changed, but this is not the case for the 15cm markers; thus, the 15cm single ArUco marker was selected to prevent potential printing error and to provide redundant white space around the ArUco marker to safeguard detectability in adverse lighting conditions.

Table of Contents




| | 0.5 | 1 | 1.5 | 2 | 2.5 | 3 | 3.5 | 4 | 4.5 | 5 | 5.5 | 6 | 6.5 | 7 | 7.5 | 8 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Actual | 0.5 | 1 | 1.5 | 2 | 2.5 | 3 | 3.5 | 4 | 4.5 | 5 | 5.5 | 6 | 6.5 | 7 | 7.5 | 8 |
| Aruco - 20cm | 0.508 | 1.081 | 1.532 | 2.038 | 2.589 | 3.178 | 3.77 | 4.374 | 4.89 | 5.537 | 6.08 | 6.721 | 7.429 | 7.923 | 8.416 | 9.165 |
| Aruco - 15cm | 0.502 | 1.061 | 1.522 | 2.004 | 2.585 | 3.120 | 3.707 | 4.4 | 4.829 | 5.599 | 6.035 | 6.69 | 7.507 | 7.665 | 8.258 | 9.122 |
| Aruco - 10cm | 0.474 | 1.019 | 1.455 | 1.932 | 2.521 | 3.053 | 3.63 | 4.352 | 4.826 | 5.355 | 6.12 | 6.811 | 7.144 | | | |
| Charuco - 20cm | 0.431 | 0.876 | 1.511 | 2.026 | | | | | | | | | | | | |
| Charuco - 15cm | 0.443 | 1.2 | | | | | | | | | | | | | | |
| Charuco - 10cm | 0.47 | | | | | | | | | | | | | | | |




| | 0.5 | 1 | 1.5 | 2 | 2.5 | 3 | 3.5 | 4 | 4.5 | 5 | 5.5 | 6 | 6.5 | 7 | 7.5 | 8 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Aruco - 20cm | 2% | 8% | 2% | 2% | 4% | 6% | 8% | 9% | 9% | 11% | 11% | 12% | 14% | 13% | 12% | 15% |
| Aruco - 15cm | 0% | 6% | 1% | 0% | 3% | 4% | 6% | 10% | 7% | 12% | 10% | 12% | 15% | 10% | 10% | 14% |
| Aruco - 10cm | 5% | 2% | 3% | 3% | 1% | 2% | 4% | 9% | 7% | 7% | 11% | 14% | 10% | | | |
| Charuco - 20cm | 14% | 12% | 1% | 1% | | | | | | | | | | | | |
| Charuco - 15cm | 11% | 20% | | | | | | | | | | | | | | |
| Charuco - 10cm | 6% | | | | | | | | | | | | | | | |

*Figure 3: Distance Accuracy Data*

Table of Contents

## Pose Estimation - Angular Accuracy Study

To test the accuracy of the estimated rotation angle compared to the actual rotation angle for Pitch, Yaw, and Roll (as seen in *Figure 4: Camera Pitch, Yaw, and Roll Relative to Marker*), the setup in *Figure 5: Angle Accuracy Testing Setup* was utilized. The marker and camera were both tilted to simulate a rotation about the relative x, y, and z axes. A smartphone's gyroscope was used to measure the angle in 15° increments.

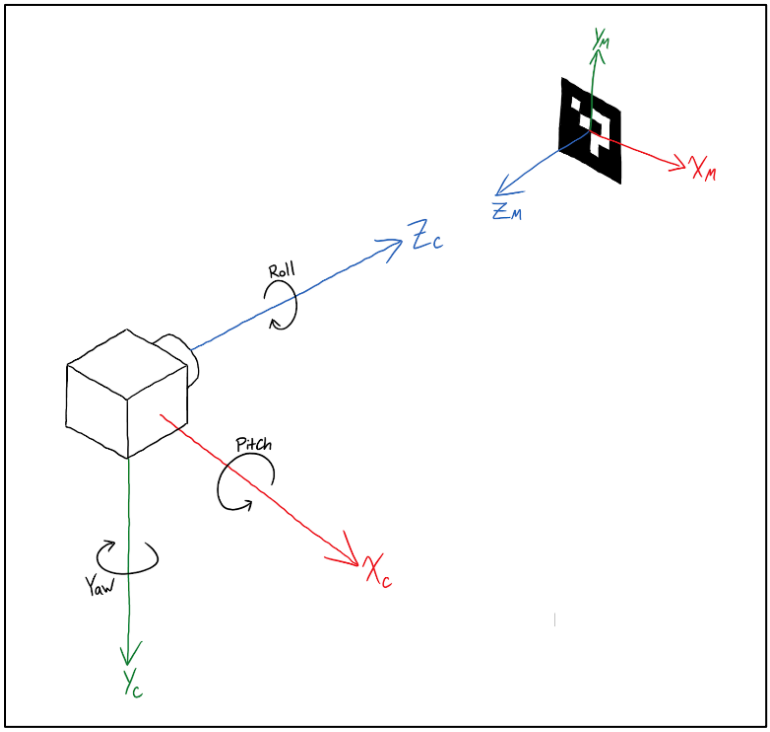



*Figure 4: Camera Pitch, Yaw, and Roll Relative to Marker*

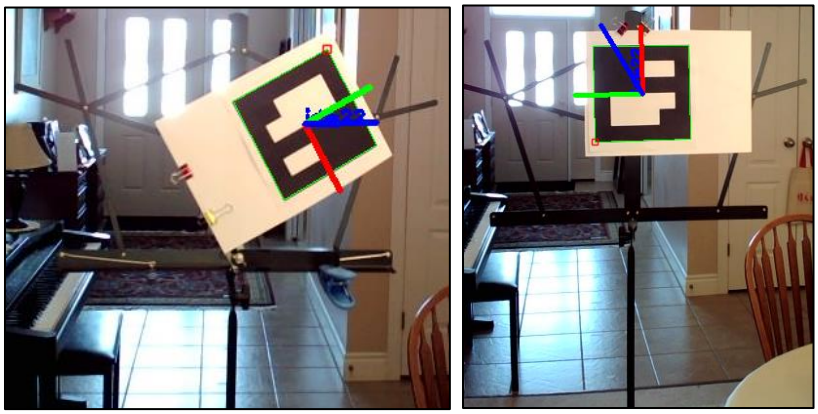


*Figure 5: Angle Accuracy Testing Setup*

The results of this study are in *Figure 6: Angle Accuracy Data*.

Table of Contents

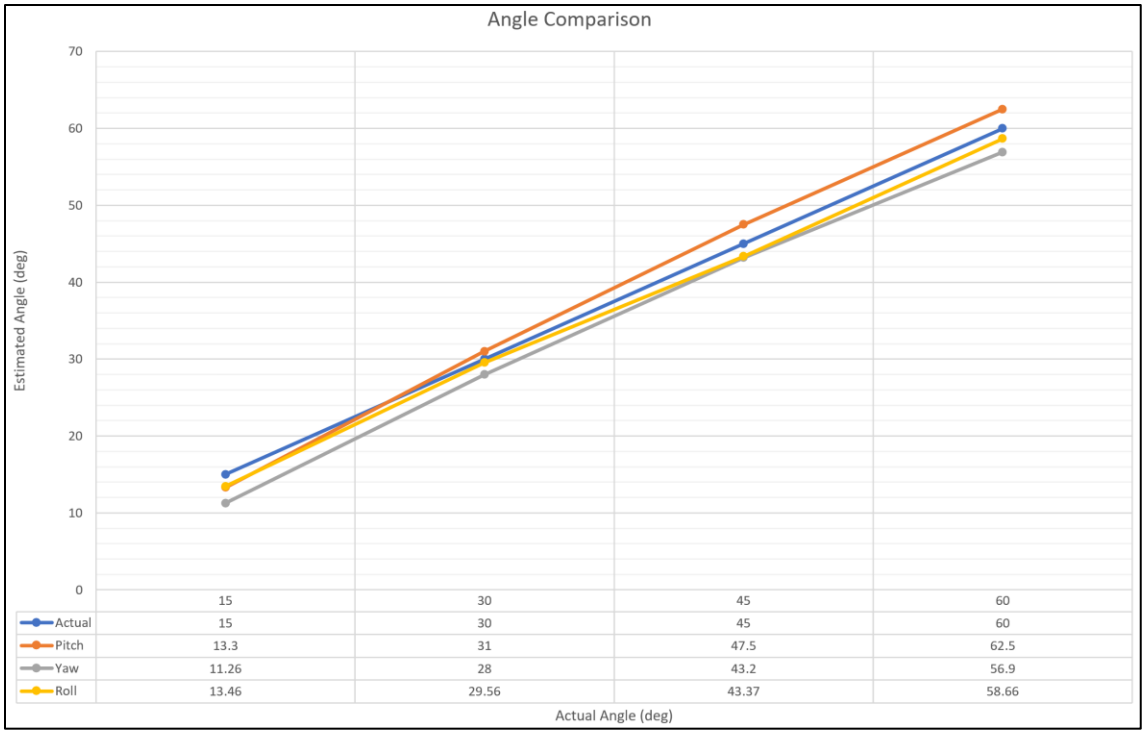



*Figure 6: Angle Accuracy Data*

From the test data, rotations between 15° and 60°, the estimated angles do not significantly deviate from the actual angles. Excluding one outlier, the average difference between the estimated and actual angles was 1.78° and the standard deviation was 0.71°. This means that the maximum difference between the estimated and actual angle is 2.49°. Considering that the haptic system delivers this information in 22.5° increments, the maximum angular error is 15.54°, as shown in *Figure 7: Maximum Error with Haptic System*. Assuming the system can place the user 0.75m away from the marker, this places them a maximum of 20cm on either side of the marker, and they would still be in the vicinity of the object they would like to reach.

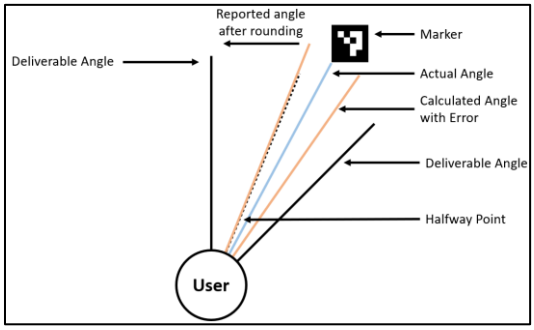



*Figure 7: Maximum Error with Haptic System*

Table of Contents

## CPU and Memory Diagnostics

Once the vision system class member functions had been programmed successfully, Visual Studio 2019's Diagnostics Tool [7] was used to study the CPU load and memory usage profiles through the start-up of the program to a period of high load due to simultaneously scanning and estimating the camera pose relative to six ArUco markers. The program was run on a computer with an Intel Core i5-4590 3.3GHz CPU (Quad-core) [12] and 16GB of DDR3 RAM. The profiles are shown below:

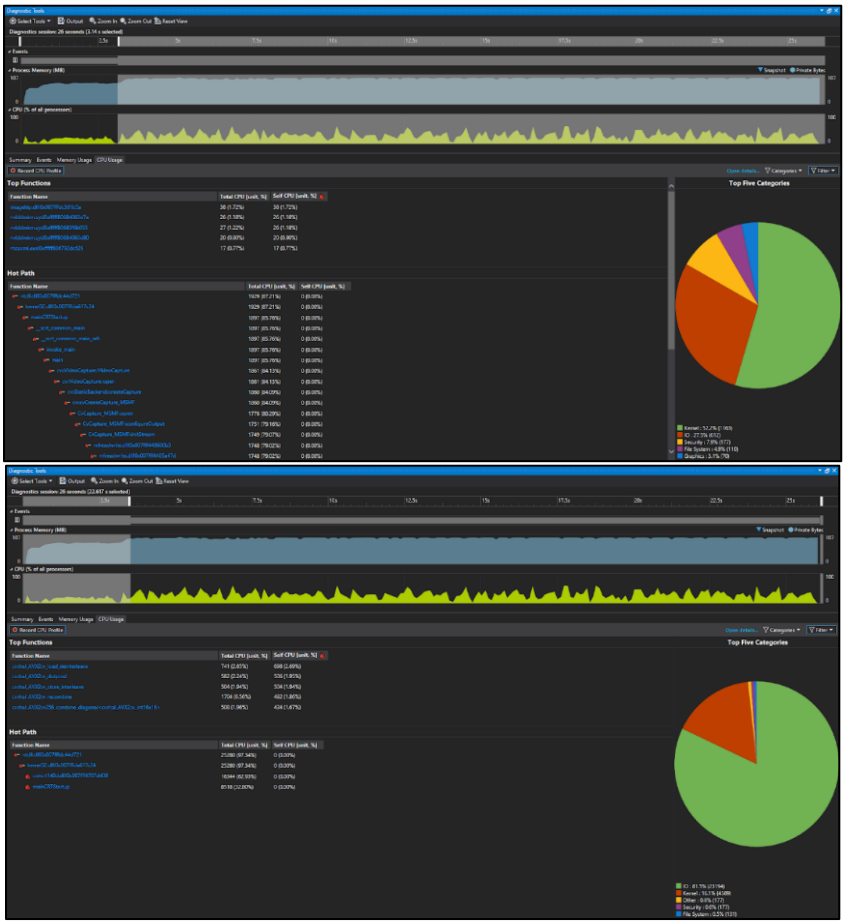


*Figure 8: Start-up and Runtime Diagnostics*

The average memory usages during start-up and runtime were 71MB and 110MB, respectively. Our benchmark Android device (Samsung Galaxy A5) has 3GB of RAM [13], so the program would only utilize 3.67% of the smartphone's available memory at a time.

The average CPU loads during start-up and runtime were 18% and 35%, respectively. Our benchmark Android device has an Octa-core 1.9GHz Cortex-A53 [13]. These processors are not very comparable, unfortunately, since power factors into CPU performance. Since the CPU load is nontrivial, processing optimizations (e.g., disabling multiple poses estimated per frame, reducing image resolution, etc.) may need to be implemented as the vision system is integrated into the Android application.

Table of Contents

# Haptic Subsystem

## Battery Selection

*Table 1: Haptic System Component Current Draw*

| Component | Current Draw |
|---|---|
| Arduino Uno | 50mA [14] |
| Breakout Boards | 5mA [15] |
| Vibrating Motors (at 100% duty cycle) | 100mA [16] |

When continuously vibrating at full power:

$$I_{draw,total} = (50\text{mA}) + (3 \times 5\text{mA}) + (3 \times 100\text{mA}) \\ = 365\text{mA}$$

*Table 2: Battery Comparison*

| Battery Type | Rechargeable | Voltage per Cell | Charge Capacity | Cells Required[1] | Minimum Hours of Use (for 365mA draw) |
|---|---|---|---|---|---|
| AA Alkaline | No | 1.5V | 2000-3000mAh [17] | 5-6 | 5.47h |
| Lithium-Ion Polymer | Yes | 3.7-4.2V | 2500mAh [18] | 2-3 | 6.85h |
| PP3 | No | 9V | 400-1200mAh [19] | 1 | 1.1h |
| D-Cell | No | 1.5V | 2000-18000mAh [20] | 5-6 | 5.47h |
| Nickel-Metal Hydride AA | Yes | 1.25V | 1900mAh [21] | 6 | 5.2h |

The intended user of this product is a secondary school student attending a school for the blind. An average school day in a secondary school lasts for 6-7 hours [22]. Assuming 7 hours of use, none of the proposed battery solutions can last the entire day.

LiPo has the best charge capacity and almost lasts the entire day; however, it is unlikely that a secondary school student would use this device continuously for the entire school day, since a majority of the day is spent sitting in a classroom. To estimate actual usage time in a five-period school day, it will be assumed that students have two 5-minute breaks between first and second,

[1] Requirement stems from Arduino Uno external power voltage requirement of 7-12V. Assume cells are placed in series to meet this requirement [14].

Table of Contents

and fourth and fifth period; that it is in use for two 15-minute periods before classes begin and after classes end; that there is constant usage during a 1-hour lunch break; and 25% usage during the four 75-minute periods of class.

$$\begin{aligned} t_{use,total} &= 60\ \text{min} + (2 \times 5\ \text{min}) + (2 \times 15\ \text{min}) + (4 \times 75\ \text{min}) \times 0.25 \\ &= 75\ \text{min} \\ &= 2.9167\ \text{h} \end{aligned}$$

All batteries, excluding the PP3, can supply enough power to last for the estimated 3 hours of usage during the school day. To narrow down battery selection, further analysis is required.

Firstly, D-Cell batteries are large and heavy, and to meet the voltage requirements for powering the Arduino Uno, five to six of these batteries would be required [20]. This is unideal for a wearable system, so they will be eliminated.

Lithium polymer rechargeable batteries, while they seem promising, come with many safety concerns. LiPo batteries pose a safety hazard, because any damage done to the batteries that bend, puncture, or otherwise damage them can create small explosions or catch fire [18]. For a wearable device that needs to be durable, withstand falls and rough use, a delicate battery is not ideal. Furthermore, LiPo batteries cannot be easily purchased in grocery or "big box" stores, where users are likely to purchase batteries. LiPo batteries purchased online are another safety risk because many online manufacturers create faulty batteries that are prone to failure. For this reason, LiPo will be eliminated from consideration.

Remaining are AA alkaline and NiMH AA batteries. Although AA alkaline batteries last slightly longer than NiMH batteries, NiMH batteries are rechargeable and just as easily purchased in stores as AA alkaline [21]. There are no other differences between these options, so NiMH AA-size batteries will be recommended and used for this product.

Table of Contents

## Charging Board

To charge the six AA NiMH batteries used to power the microcontroller, a charging controller and its associated circuit were designed. To test this circuit, the board would be plugged into the wall using a 10V DC converter, with the battery pack, and the microcontroller within the case of the device. As the batteries are charged, temperature measurements should be taken periodically. If the temperature was found to be above its normal operating range, 70°C, components such as heat sinks and fans can be added to the design to dissipate the excess heat, and vents can be added to the case around where the board is located. There is also a temperature monitor that can be added to the circuit so it can regulate itself and stop charging as needed. The charge of the batteries should be measured periodically using a multimeter to check how quickly they are charging and to ensure they charge fully before the charging controller cuts them off. The accuracy of the output pins from the charging board that report the charging setting of the board (trickle, fast, or no charge) can be tested by checking the input pins on the microcontroller of the system and the current flow from the board to the batteries. The life of the rechargeable batteries should be tested to ensure they can power the system for 3 hours as specified by the design constraints.

Table of Contents

## Armband Material Selection

Material selection of the haptic armband was conducted based on qualitative data regarding the breathability, comfort, elasticity, and durability of different fabrics. Sources such as CES Edupack materials database do not include data for different fabrics as they do for different metals and alloys. As such, research was done to find qualitative data.

Cotton [23]

- Comfortable in all climates
- Breathable and moisture wicking
- Retains body heat in cold weather
- Weakens from extended exposure to sunlight
- Hypoallergenic
- Inexpensive
- Moderately durable

Lycra [23]

- Stretchy and flexible
- Can be stretched 4-7× times its initial length without plastic deformation
- Durable
- Always combined with other fibers (100% lycra products not available)
- Moisture wicking
- More expensive

Polycotton [23]

- Very strong and durable
- Breathable
- Shrinks less than pure cotton
- Cheaper than 100% cotton
- Can be washed more often without deformation

For the haptic wearable it is important that it is comfortable, durable, breathable, and washable. Since this product would be used in a school setting, sanitary concerns would mean the product will be washed frequently for other students to use. The product is designed as an upper armband, where sweat from the underarm may be absorbed by the armband; this makes breathability and moisture wicking desirable.

Lycra, while it is moisture wicking and durable, the need for elasticity is unnecessary because the armband is already designed to be modular in size for different users. Elasticity is not a large benefit to the product. It is also the most expensive of the three types of fabric, so it will be eliminated [23].

Cotton and polycotton (a polyester and cotton blend) are comparable; however, polycotton has increased durability and decreased cost, while maintaining the breathability of cotton. Polycotton survives more washes without deformation, which is beneficial because of how often this product will be washed in a school setting [23]. For these reasons, polycotton is the selected material for the haptic armband.

Table of Contents

# Haptic Wearable – Human Studies

## Nerve Resolution

To determine the minimum distance the vibrating motors should be placed on the arm such that they are detected as individual points, instead of one single point, a two-point discrimination test was conducted. To do this, two pencils were placed simultaneously on the upper arm of a test subject [24]. The distance between the two points is measured, and the subject guesses whether one or two pencils are touching the arm. The data obtained from this test was then compared to standard data available online [25].

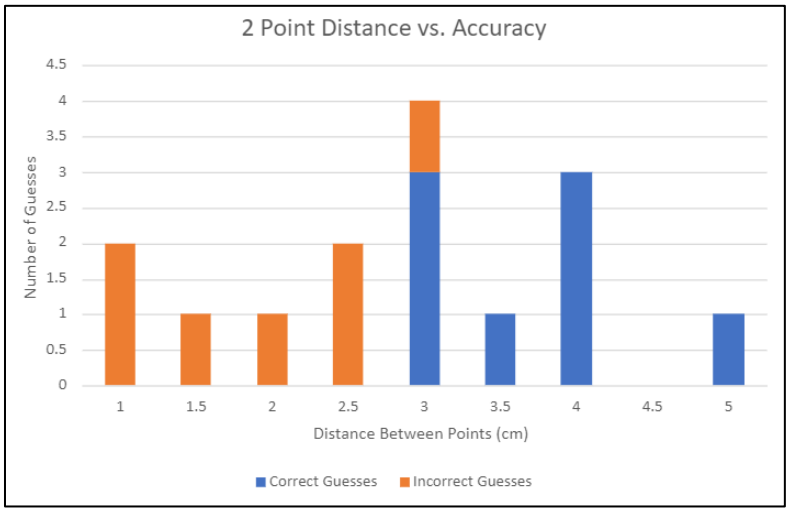



*Figure 9: 2-Point Discrimination Test Results*

At 3cm, the distance between points was guessed correctly and incorrectly. Any distance less than 3cm was guessed incorrectly as one point, instead of two. This experimental data matches available data for two-point discrimination tests for the given location on the body, which is found to be 30mm to 40mm [25]. Therefore, the vibrating motors on the haptic wearable will be kept at minimum 3cm apart from one another, but it is recommended to keep them 4cm apart. This condition will have to be checked by the user when donning the wearable, since the location of the vibrating motors is customizable to suit different arm sizes.

While it is true that your nerves do not detect pressure and vibration in the same way, [26] this test served to find a minimum distance which we would begin to test the motors at.

## Feedback Methods

The haptic wearable uses three vibrational motors to convey distance and orientation information. Two different methods of conveying directional information through the haptic wearable to the user were programmed, implemented, and tested. The two methods are called “vector” and “compass”.

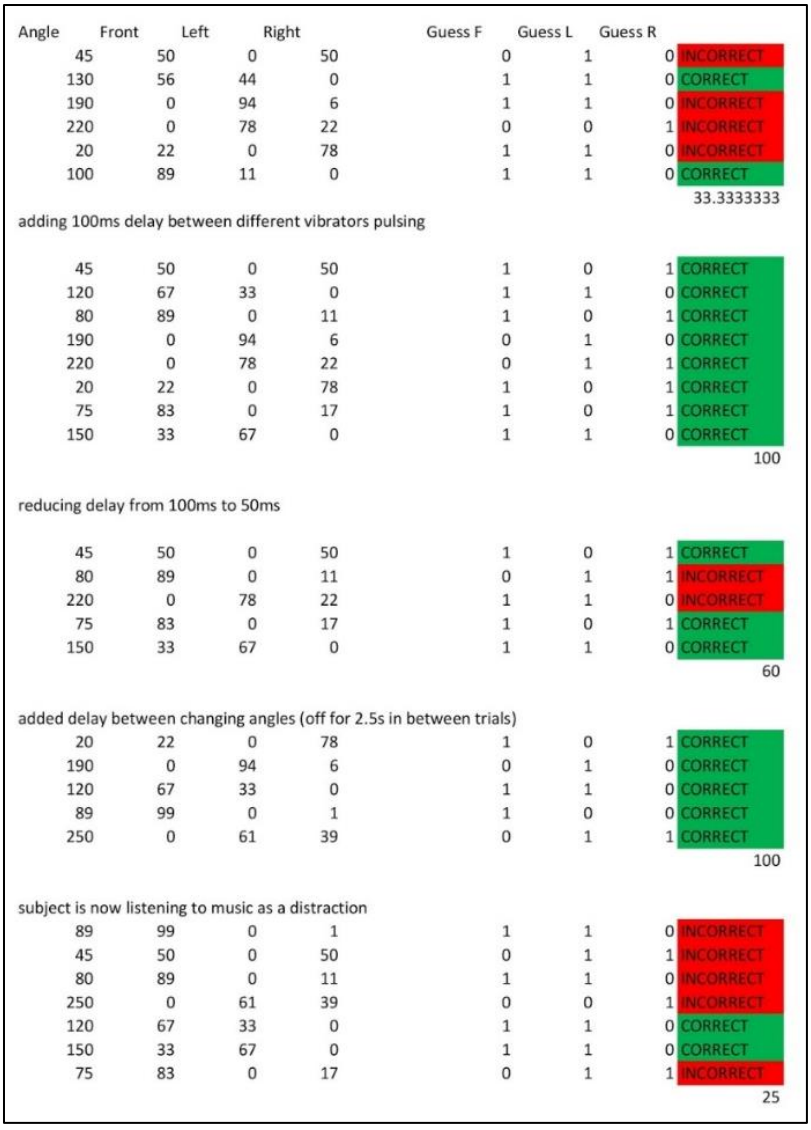
Note that the three vibrators will be referred to as the front, left, and right vibrators. 0° is considered directly to the right of the user, and the amplitude/strength of vibration ranges from 0% to 100%.

The vector method operates by turning on the motors in a combination that represents the angle. For example, if the front and right vibrators turn on at equal strength, that would indicate a 45° angle. Alternatively, if the front vibrator was on at 90% strength, and the right vibrator was on at 10% strength, that would indicate an angle of 80°. There is no indication of distance from the point of interest in this mode.

The compass method operates by giving three pulses, followed by a rest pulse, to indicate direction. Amplitude of the vibration is not relevant in this mode and all vibrators operate at the same amplitude. The pattern of the pulses indicates the direction and the frequency of the pattern repetition indicates the distance from the point of interest, with a higher frequency corresponding to a shorter distance. For example, if the vibrational pattern was right, right, front, rest, that would indicate an angle between 11° and 34°. Each pattern indicates a 23° range where the point of interest is located relative to the user. Closing the distance increases the frequency exponentially such that the closer the user moves, the less change in distance is required to alter the frequency. For example, 8m away from the target results in 40bpm frequency, 4m away is 120bpm, and 1m away is 360bpm. This is modelled by the curve $y$=463.35*$e$^(−0.312$x$), where $y$ is the resulting beats per minute, and x is the distance away in meters.

A test subject was asked to close their eyes while wearing the haptic wearable and guess what vibrators were on at minimum or guess what direction it was indicating.

Table of Contents



| Angle | Front | Left | Right | Guess F | Guess L | Guess R | |
|---|---|---|---|---|---|---|---|
| 45 | 50 | 0 | 50 | 0 | 1 | 0 | INCORRECT |
| 130 | 56 | 44 | 0 | 1 | 1 | 0 | CORRECT |
| 190 | 0 | 94 | 6 | 1 | 1 | 0 | INCORRECT |
| 220 | 0 | 78 | 22 | 0 | 0 | 1 | INCORRECT |
| 20 | 22 | 0 | 78 | 1 | 1 | 0 | INCORRECT |
| 100 | 89 | 11 | 0 | 1 | 1 | 0 | CORRECT |
| | | | | | | | 33.3333333 |
| adding 100ms delay between different vibrators pulsing | | | | | | | |
| 45 | 50 | 0 | 50 | 1 | 0 | 1 | CORRECT |
| 120 | 67 | 33 | 0 | 1 | 1 | 0 | CORRECT |
| 80 | 89 | 0 | 11 | 1 | 0 | 1 | CORRECT |
| 190 | 0 | 94 | 6 | 0 | 1 | 0 | CORRECT |
| 220 | 0 | 78 | 22 | 0 | 1 | 1 | CORRECT |
| 20 | 22 | 0 | 78 | 1 | 0 | 1 | CORRECT |
| 75 | 83 | 0 | 17 | 1 | 0 | 1 | CORRECT |
| 150 | 33 | 67 | 0 | 1 | 1 | 0 | CORRECT |
| | | | | | | | 100 |
| reducing delay from 100ms to 50ms | | | | | | | |
| 45 | 50 | 0 | 50 | 1 | 0 | 1 | CORRECT |
| 80 | 89 | 0 | 11 | 0 | 1 | 1 | INCORRECT |
| 220 | 0 | 78 | 22 | 1 | 1 | 0 | INCORRECT |
| 75 | 83 | 0 | 17 | 1 | 0 | 1 | CORRECT |
| 150 | 33 | 67 | 0 | 1 | 1 | 0 | CORRECT |
| | | | | | | | 60 |
| added delay between changing angles (off for 2.5s in between trials) | | | | | | | |
| 20 | 22 | 0 | 78 | 1 | 0 | 1 | CORRECT |
| 190 | 0 | 94 | 6 | 0 | 1 | 0 | CORRECT |
| 120 | 67 | 33 | 0 | 1 | 1 | 0 | CORRECT |
| 89 | 99 | 0 | 1 | 1 | 0 | 0 | CORRECT |
| 250 | 0 | 61 | 39 | 0 | 1 | 1 | CORRECT |
| | | | | | | | 100 |
| subject is now listening to music as a distraction | | | | | | | |
| 89 | 99 | 0 | 1 | 1 | 1 | 0 | INCORRECT |
| 45 | 50 | 0 | 50 | 0 | 1 | 1 | INCORRECT |
| 80 | 89 | 0 | 11 | 1 | 1 | 0 | INCORRECT |
| 250 | 0 | 61 | 39 | 0 | 0 | 1 | INCORRECT |
| 120 | 67 | 33 | 0 | 1 | 1 | 0 | CORRECT |
| 150 | 33 | 67 | 0 | 1 | 1 | 0 | CORRECT |
| 75 | 83 | 0 | 17 | 0 | 1 | 1 | INCORRECT |
| | | | | | | | 25 |

*Figure 10: Vector Mode Test Results*

Several changes were made to the program in between trials, and the changes are described in the image above. Overall, the results varied, with a 63.66% success rate. This success rate noticeably dropped once the test subject was distracted by music while trying to interpret the vibrations. For all trials, the test subject only guessed which vibrators were turned on and was unable to guess what direction they were indicating because variance in amplitude was difficult or impossible to detect.

The results for compass testing are shown below. Under ideal circumstances, more test data would be collected using many different test subjects; however, due to COVID-19 restrictions, only two test subjects were used.

Compass Mode Testing
Using BenHapticCodeV1

| Intensity | BPM | Angle | Pattern | Guess | Correct? |
|---|---|---|---|---|---|
| 100 | 500 | 102.7439 | NNW | NNW | Y |
| 100 | 500 | 198.2278 | WWW | WWW | Y |
| 100 | 500 | 357.5561 | EEE | EEE | Y |
| 100 | 500 | 321.7935 | EEE | EEE | Y |
| 100 | 500 | 63.94497 | NNE | NNE | Y |
| 100 | 500 | 228.2986 | WWW | WWW | Y |
| 100 | 500 | 170.2108 | WWW | WWW | Y |
| 100 | 500 | 146.7919 | NW | NW | Y |
| 100 | 500 | 331.7263 | EEE | EEE | Y |
| 100 | 500 | 192.2447 | WWW | WWW | Y |
| 100 | 500 | 17.77541 | EEN | EEN | Y |
| 100 | 500 | 172.6597 | WWW | WWW | Y |
| 100 | 500 | 179.0818 | WWW | WWW | Y |
| 100 | 500 | 64.43235 | NNE | NNE | Y |
| 100 | 500 | 63.23144 | NNE | NNE | Y |
| 100 | 500 | 178.5063 | WWW | WWW | Y |
| 100 | 500 | 125.2197 | NW | NW | Y |
| 100 | 500 | 28.81846 | EEN | EEN | Y |
| 100 | 500 | 69.9252 | NNE | NNE | Y |
| 100 | 500 | 4.697562 | EEE | EEE | Y |
| 50 | 500 | 256.7258 | WWW | WWW | Y |
| 50 | 500 | 83.67669 | NNN | NNN | Y |
| 50 | 500 | 262.2727 | WWW | WWW | Y |
| 50 | 500 | 133.5025 | NW | NW | Y |
| 50 | 500 | 201.0022 | WWW | WWW | Y |
| 50 | 500 | 49.28118 | NE | NE | Y |
| 50 | 500 | 172.4357 | WWW | WWW | Y |
| 50 | 500 | 290.4723 | EEE | EEE | Y |
| 50 | 500 | 156.4497 | WWN | WWN | Y |
| 50 | 500 | 77.14779 | NNE | NNE | Y |
| 50 | 500 | 85.43026 | NNN | NNN | Y |
| 50 | 500 | 163.8599 | WWN | WWN | Y |
| 50 | 500 | 171.8346 | WWW | WWW | Y |
| 50 | 500 | 43.25872 | NE | NE | Y |
| 50 | 500 | 177.1338 | WWW | WWW | Y |
| 50 | 500 | 85.57621 | NNN | NNN | Y |
| 50 | 500 | 10.05703 | EEE | EEE | Y |
| 50 | 500 | 70.63314 | NNE | NNE | Y |
| 50 | 500 | 159.4993 | WWN | WWN | Y |
| 50 | 500 | 148.1756 | WWN | WWN | Y |
| 50 | 500 | 138.7396 | NW | NW | Y |

*Figure 11: Compass Mode Test Results*

Table of Contents

NEW TEST SUBJECT

| | | | | | |
|---|---|---|---|---|---|
| 100 | 100 | 122.4955 | NNW | NNW | Y |
| 100 | 100 | 150.9427 | WWN | WWN | Y |
| 100 | 100 | 116.5288 | NNW | NNW | Y |
| 100 | 100 | 25.87333 | EEN | EEN | Y |
| 100 | 100 | 85.47622 | NNN | NNN | Y |
| 100 | 100 | 118.2837 | NNW | NNW | Y |
| 100 | 100 | 147.3926 | WWN | WWN | Y |
| 100 | 100 | 122.8351 | NNW | NNW | Y |
| 100 | 100 | 178.1839 | WWW | WWW | Y |
| 100 | 100 | 75.81921 | NNE | NNE | Y |
| 100 | 100 | 156.0555 | WWN | WWN | Y |
| 100 | 100 | 162.6112 | WWN | WWN | Y |
| 100 | 100 | 40.50468 | NE | NE | Y |
| 100 | 100 | 35.67699 | NE | NE | Y |
| 100 | 100 | 150.0881 | WWN | WWN | Y |
| 100 | 100 | 13.30237 | EEN | EEN | Y |
| 100 | 100 | 20.21618 | EEN | EEN | Y |
| 100 | 100 | 76.87615 | NNE | NNE | Y |
| 100 | 100 | 57.85762 | NNE | NNE | Y |
| 100 | 100 | 119.2117 | NNW | NNW | Y |

Adding in Distance Determination

| Intensity | BPM | Distance | Angle | Pattern | Faster/Slower? | Guess | Correct D? | Correct O? |
|---|---|---|---|---|---|---|---|---|
| 100 | 100 | 2 | 81.00521 | NNN | N/A | NNN | Y | N/A |
| 100 | 100 | 3 | 78.97264 | NNE | slower | NNE | Y | Y |
| 100 | 100 | 1 | 78.13101 | NNE | faster | NNE | Y | Y |
| 100 | 100 | 4 | 2.90704 | EEE | slower | EEE | Y | Y |
| 100 | 100 | 4 | 169.3755 | WWW | same | WWW | Y | Y |
| 100 | 100 | 3 | 102.9924 | NNW | faster | NNW | Y | Y |
| 100 | 100 | 2 | 5.76591 | EEE | faster | EEE | Y | Y |
| 100 | 100 | 1 | 21.35795 | EEN | faster | EEN | Y | Y |
| 100 | 100 | 2 | 110.4602 | NNW | faster | NNW | N | Y |
| 100 | 100 | 1 | 89.91502 | NNN | same | NNN | N | Y |

New Test Subject

| Intensity | BPM | Angle | Pattern | Guess | Correct? |
|---|---|---|---|---|---|
| 100 | 100 | 73.3408 | NNE | NNE | Y |
| 100 | 100 | 33.29624 | EEN | EEN | Y |
| 100 | 100 | 21.54163 | EEN | EEN | Y |
| 100 | 100 | 137.2694 | NW | NW | Y |
| 100 | 100 | 31.88136 | EEN | EEN | Y |
| 100 | 100 | 5 | EEE | EEE | Y |
| 100 | 100 | 94.86891 | NNN | NNN | Y |
| 100 | 100 | 148.4266 | WWN | WWN | Y |

*Figure 12: Compass Mode Test Results Continued*

Table of Contents

| | | | | | |
|---|---|---|---|---|---|
| 100 | 100 | 20.97045 | EEN | EEN | Y |
| 100 | 100 | 138.0658 | NW | NW | Y |
| 100 | 100 | 76.61814 | NNE | NNE | Y |
| 100 | 100 | 89.16998 | NNN | NNN | Y |
| 100 | 100 | 101.3351 | NNN | NNN | Y |
| 100 | 100 | 83.41898 | NNN | NNN | Y |
| 100 | 100 | 138.9938 | NW | NW | Y |

| Intensity | BPM | Distance | Angle | Pattern | Faster/Slower? | Guess | Correct D? | Correct O? |
|---|---|---|---|---|---|---|---|---|
| 100 | 100 | 2 | 110.323 | NNW | N/A | NNW | Y | N/A |
| 100 | 100 | 3 | 179.5759 | WWW | slower | WWW | Y | Y |
| 100 | 100 | 2 | 93.31999 | NNN | faster | NNN | Y | Y |
| 100 | 100 | 5 | 92.58766 | NNN | slower | NNN | Y | Y |
| 100 | 100 | 1 | 134.8491 | NW | faster | NW | Y | Y |
| 100 | 100 | 2 | 128.8954 | NW | slower | NW | Y | Y |
| 100 | 100 | 3 | 76.12817 | NNE | faster | NNE | N | Y |
| 100 | 100 | 4 | 120.9532 | NNW | slower | NNW | Y | Y |
| 100 | 100 | 4 | 131.6001 | NW | slower | NW | N | Y |
| 100 | 100 | 3 | 117.1074 | NNW | faster | NNW | Y | Y |
| 100 | 100 | 5 | 109.5829 | NNW | same | NNW | N | Y |
| 100 | 100 | 4 | 62.05845 | NNE | faster | NNE | Y | Y |
| 100 | 100 | 1 | 40.72234 | NE | faster | NE | Y | Y |
| 100 | 100 | 3 | 9.556283 | EEE | same | EEE | N | Y |
| 100 | 100 | 2 | 117.613 | NNW | slower | NNW | N | Y |

*Figure 13: Compass Mode Test Results Continued*

Note, that "Correct D?" means correct distance (i.e. correct identification of a faster or slower frequency), and "Correct O?" means correct orientation (i.e. correct identification of the compass pattern). In these results, the front vibrator is referred to as the north vibrator, right is east, and left is west.

From the data above, it is evident the compass mode testing was highly successful. Users were able to identify the pulse pattern and what direction easily and correctly it was indicating. Furthermore, distance was easily identifiable based on frequency increasing or decreasing, with the user stating whether the frequency was faster or slower, indicating a smaller or larger distance away from the target. The base beats per minute (shown in the second column) and amplitude of the vibration (shown in the first column) did not affect the user's ability to discern distance and orientation information. Therefore, to conserve power, the amplitude used will be 50% of maximum power, and the base beats per minute frequency will be 100bpm. Incorrect answers were typically a result of either identical information being conveyed, with the user assuming the new trial would be different from the previous trial and providing a different response instead of an identical one, or because the change in distance/frequency was minor and difficult to discern. Overall, this mode was found to be intuitive and easy to use, and test subjects needed only one or two trials to understand the information pattern.

Due to the reasons previously outlined, the compass method was selected for use in the final design.

Table of Contents

## Electrical Enclosure Design

The enclosure design was straightforward as it was designed to be as small as possible. Finite element analysis (FEA) was done on the enclosure clip, to ensure it could flex enough to slip into the belt of a potential user. The two halves of the enclosure fit together with M2 screws and barbed press fit brass threaded inserts.

The default FEA settings were used, with a “max von mises stress” failure condition. A load of 20N was applied to the end of the clip, and fixed constraints applied to the backplane of the enclosure. The two figures below show the undeformed model, and the deformed model.

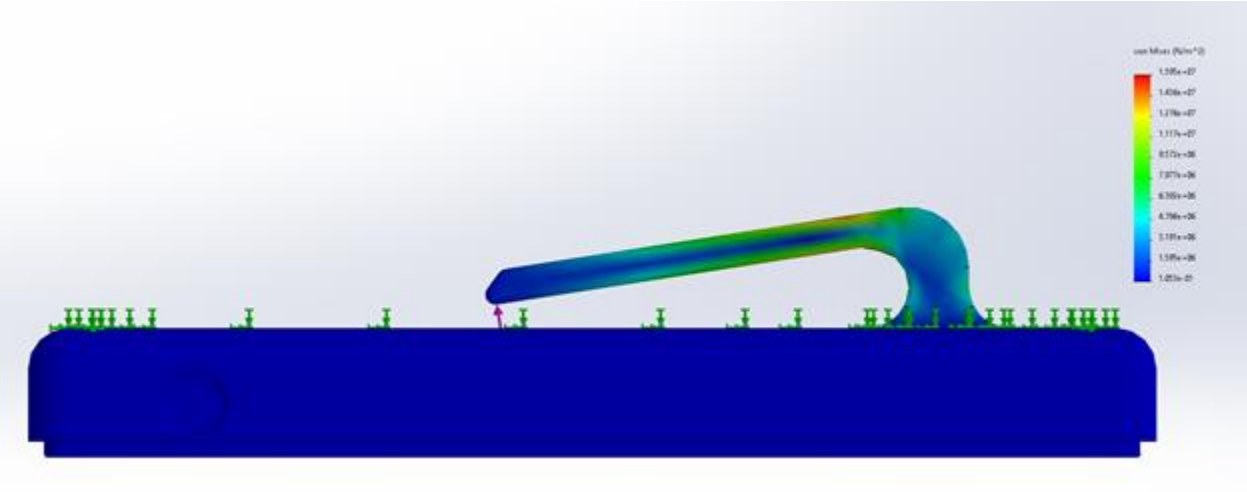


*Figure 14: Undeformed Model with Stress Gradient*

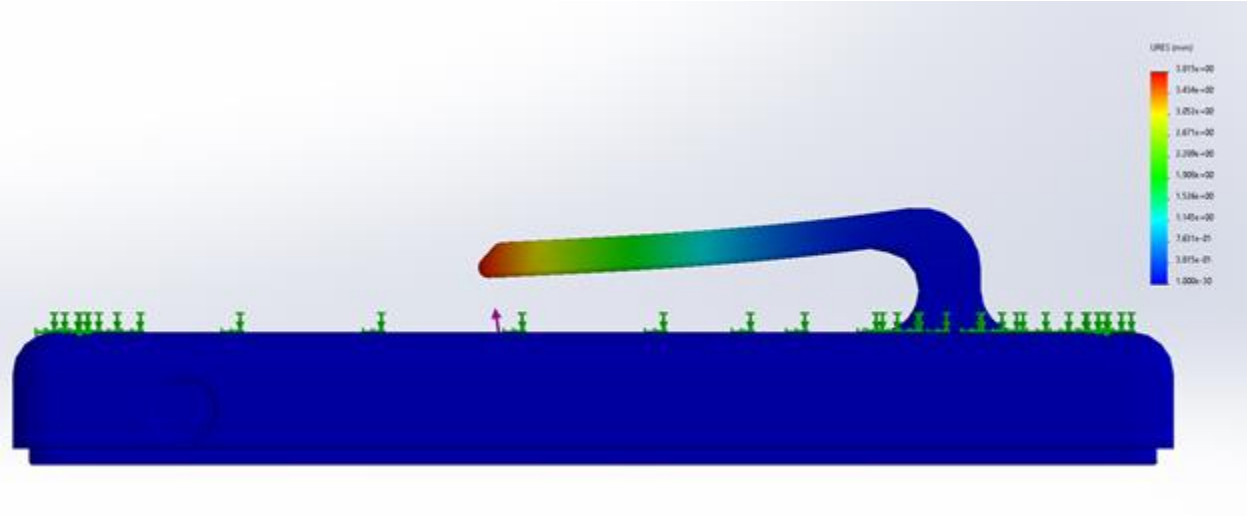


*Figure 15: Deformed Body with Displacement Gradient*

Table of Contents

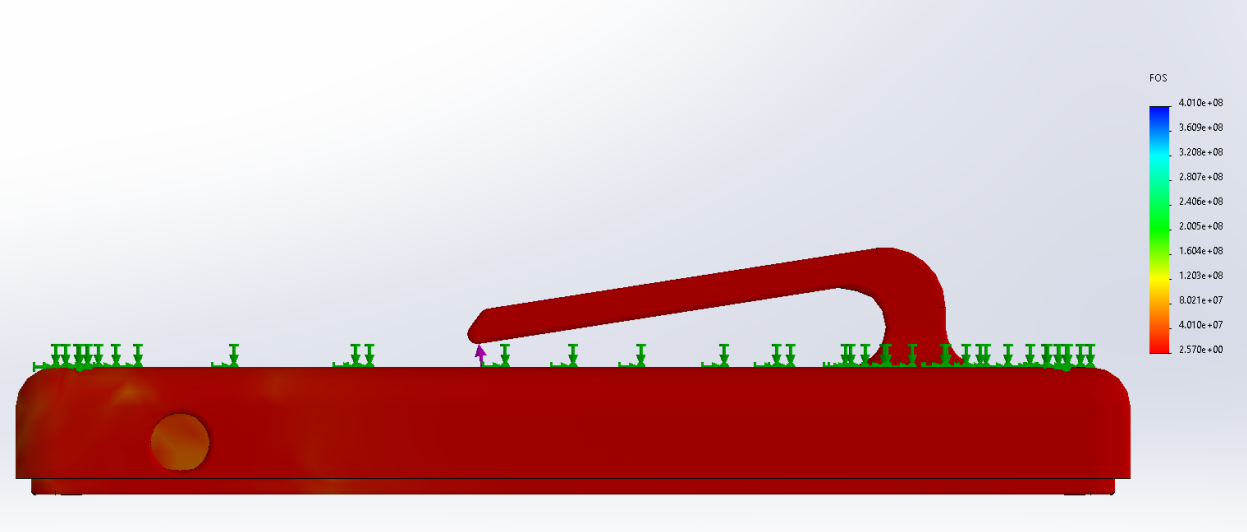



*Figure 16: Factor of safety Gradient*

The minimum factor of safety was 2.57, with 20N of direct force. A real-world setting would yield a less directed force, reducing stress concentration and yielding an even higher FOS. Cyclical analysis was not done, as the clip is not intended to be cycled at a high rate.

# Design Documentation

## Vision Subsystem

### Camera Calibration Software

The camera calibration software works by automatically running through a phased process to capture images and record corner points of a ChArUco board at varying positions and orientations within the camera's visual field. Following the instructions outlined after the code below, the software will guide the user through six different stages of calibration. During this time, the user will move the board in front of the phone's camera as instructed, and the system will collect 10 images over the course of each phase.

If the user is not satisfied with captured images or the system times out before sufficient images are captured, they may choose to restart the previous phase of calibration. At the end of the image collecting period, the 60 images will be used to calculate the distortion coefficients and camera matrix that will later be used to calculate the distance and angle between the user's camera and the marker they are approaching. This process will only need to be completed when the system is used for the first time, but if the user is receiving an inaccurate angle or distance during use, they may wish to run the calibration process again.

#### Charuco_Cali_R3.cpp

```
#include <opencv2/highgui.hpp>
#include <opencv2/calib3d.hpp>
#include <opencv2/aruco/charuco.hpp>
#include <opencv2/imgproc.hpp>
#include <vector>
#include <iostream>
#include <ctime>

#include <sstream>
#include <fstream>

#pragma warning(disable : 4996)

using namespace std;
using namespace cv;

// Create variables for collecting info from frames
vector<vector<vector<Point2f>>> allCorners, tempCorners;
vector<vector<int>> allIds, tempIds;
vector<Mat> allImgs, tempImages;
Size imgSize;

// Calibration Board settings
int squaresX = 5;
int squaresY = 7;
double squareLength = 0.036; //need this from the user
double markerLength = 0.021; //also this
int dictionaryId = 10;

// Calibration program settings
string outputFile = "CharCaliOut";
bool showChessboardCorners = true;
int calibrationFlags = 0, totalImageCollect=60;
float aspectRatio = 1;

// Create ArUco parameters
Ptr<aruco::DetectorParameters> detectorParams = aruco::DetectorParameters::create();
Ptr<aruco::Dictionary> dictionary = aruco::getPredefinedDictionary(aruco::PREDEFINED_DICTIONARY_NAME(dictionaryId));
Ptr<aruco::CharucoBoard> charucoboard = aruco::CharucoBoard::create(squaresX, squaresY, squareLength, markerLength, dictionary);
Ptr<aruco::Board> board = charucoboard.staticCast<aruco::Board>();

Mat cameraIntrinsics, distortionCoeffs;
vector<Mat> rvecs, tvecs;

vector<Mat> allCharucoCorners;
vector<Mat> allCharucoIds;
vector<Mat> filteredImages;
```

Table of Contents

```
// saveCameraCalibration() function for saving calibrated camera intrinsics and distortion coefficients to a readable file
bool saveCameraCalibration(string name, Mat cameraIntrinsics, Mat distortionCoeffs)
{
    cout << "Writing Calibration File..." << endl;
    ofstream outStream(name);
    if (outStream)
    {
        uint16_t rows = cameraIntrinsics.rows;
        uint16_t columns = cameraIntrinsics.cols;

        outStream << rows << endl;
        outStream << columns << endl;

        for (int r = 0; r < rows; r++)
        {
            for (int c = 0; c < columns; c++)
            {
                double value = cameraIntrinsics.at<double>(r, c);
                outStream << value << endl;
            }
        }

        rows = distortionCoeffs.rows;
        columns = distortionCoeffs.cols;

        outStream << rows << endl;
        outStream << columns << endl;

        for (int r = 0; r < rows; r++)
        {
            for (int c = 0; c < columns; c++)
            {
                double value = distortionCoeffs.at<double>(r, c);
                outStream << value << endl;
            }
        }

        outStream.close();
        cout << "Calibration File Saved!" << endl;
        return true;

    }

    return false;
}

// collectImages() function for automatically capturing images when sufficient corners of calibration board are detected
bool collectImages(VideoCapture vid, int framesPerSecond, int imageCollect, int numMin)
{
    for (int i = 0; i < (framesPerSecond * numMin * 60); i++) {
        //int everyOther = floor((framesPerSecond * numMin * 60) / imageCollect);
        int everyOther = 8;
        Mat image, imageCopy, imageCopy_out;
        string cont;

        if (!vid.read(image))
            break;

        vector<int> ids;
        vector< vector<Point2f>> corners, rejected;

        // Detect markers
        aruco::detectMarkers(image, dictionary, corners, ids, detectorParams, rejected);

        // Interpolate charuco corners
        Mat currentCharucoCorners, currentCharucoIds;
        if (ids.size() > 0)
            aruco::interpolateCornersCharuco(corners, ids, image, charucoboard, currentCharucoCorners,
                currentCharucoIds);

        // Draw detected markers
        image.copyTo(imageCopy);
        if (ids.size() > 0) aruco::drawDetectedMarkers(imageCopy, corners);

        if (currentCharucoCorners.total() > 0)
            aruco::drawDetectedCornersCharuco(imageCopy, currentCharucoCorners, currentCharucoIds);


        flip(imageCopy, imageCopy_out, 1);
        imshow("Webcam", imageCopy_out);

        char character = waitKey(1000 / framesPerSecond); // need to find another way to do this

        switch (character)
        {
        case 27:
            //exit
            return 0;
            break;
        default:
            // Automatically collect images
            if (!(i%everyOther) && ids.size() > 0 && currentCharucoCorners.total()>4) {
```

```
                tempCorners.push_back(corners);
                tempIds.push_back(ids);
                tempImages.push_back(image);
                imgSize = image.size();
                cout << "Saved Image!" << "   Image #: " << tempImages.size() << endl;
                cout << "Num of Corners: " << currentCharucoCorners.total() << endl;
            }
            cout << "Time Left: " << ((numMin * 60) - (floor(i/framesPerSecond))) << endl;
        }

        if (tempImages.size() == imageCollect) {
            bool approve=false;

            while (!approve) {
                cout << "Images collected so far: " << allImgs.size() << endl;
                cout << "Ready to start next stage of calibration? (Y/N)" << endl;
                cin >> cont;

                if (cont == "Y") {
                    approve = true;
                    for (int j = 0; j < tempIds.size(); j++) {
                        allCorners.push_back(tempCorners.at(j));
                        allIds.push_back(tempIds.at(j));
                        allImgs.push_back(tempImages.at(j));
                    }
                    tempCorners.clear();
                    tempIds.clear();
                    tempImages.clear();
                    return true;
                }
                else if (cont == "N") {
                    i = 0;
                    tempCorners.clear();
                    tempIds.clear();
                    tempImages.clear();
                    approve = true;
                }
                else {
                    cout << "That is not a valid input. Please try again." << endl;
                }
            }
        }

        if (allImgs.size() == totalImageCollect) {
            break;
        }
    }

    return true;

}

// Function for converting collected images and calibration board points into
bool completeCalibration()
{
    // Initialize calibration variables
    bool CaliSuccess, saveOk;

    vector<vector<Point2f>> allCornersConcatenated;
    vector<int> allIdsConcatenated;
    vector<int> markerCounterPerFrame;
    markerCounterPerFrame.reserve(allCorners.size());

    for (unsigned int i = 0; i < allCorners.size(); i++) {
        markerCounterPerFrame.push_back((int)allCorners[i].size());
        for (unsigned int j = 0; j < allCorners[i].size(); j++) {
            allCornersConcatenated.push_back(allCorners[i][j]);
            allIdsConcatenated.push_back(allIds[i][j]);
        }
    }

    int nFrames = (int)allCorners.size();
    allCharucoCorners.reserve(nFrames);
    allCharucoIds.reserve(nFrames);

    for (int i = 0; i < nFrames; i++) {
        // Interpolate using camera parameters
        Mat currentCharucoCorners, currentCharucoIds;
        aruco::interpolateCornersCharuco(allCorners[i], allIds[i], allImgs[i], charucoboard, currentCharucoCorners, currentCharucoIds,
cameraIntrinsics, distortionCoeffs);

        allCharucoCorners.push_back(currentCharucoCorners);
        allCharucoIds.push_back(currentCharucoIds);
        filteredImages.push_back(allImgs[i]);
    }

    if (allCharucoCorners.size() < 4) {
        cerr << "Not enough corners for calibration" << endl;
        return 0;
    }
```

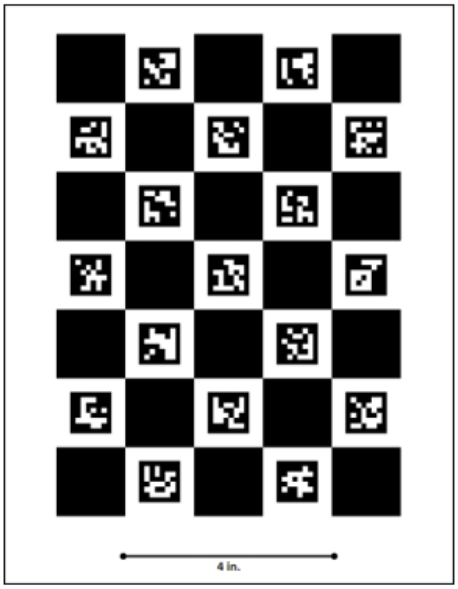
```
 // Calibrate camera using charuco
    CaliSuccess = aruco::calibrateCameraCharuco(allCharucoCorners, allCharucoIds, charucoboard, imgSize, cameraIntrinsics, distortionCoeffs,
rvecs, tvecs, calibrationFlags);

    // Check calibration success and call saveCameraCalibration() if successful to write calibration file
    if (CaliSuccess) {
        cout << "Calibration Successful!" << endl;
        saveOk = saveCameraCalibration(outputFile, cameraIntrinsics, distortionCoeffs);
    }
    else {
        cout << "Calibration Failed!" << endl;
        saveOk = false;
    }

    if (!saveOk) {
        cerr << "Cannot Write Calibration File!" << endl;
        return 0;
    }

    return CaliSuccess;
}

int main()
{
    // Open video capture device
    VideoCapture vid(0);
    if (!vid.isOpened())
    {
        return 0;
    }
    namedWindow("Webcam", WINDOW_AUTOSIZE);

    // collectImages() settings
    int framesPerSecond = 20, numMin = 1, imageCollect = 10;


    // Begin calibration phases
    for (int k = 0; k < 7; k++) {
        cout << "Begin phase " << k + 1 << " of calibration. " << endl;
        collectImages(vid, framesPerSecond, imageCollect, numMin);
    }

    cout << "Finished Collecting Images!!" << endl;

    // Call completeCalibration() function to calculate camera's intrinsic parameters
    completeCalibration();

           return 0;
}
```

## Camera Calibration Instructions

An instructional document outlining the camera calibration process should be used in conjunction with the provided camera calibration program (as seen in Camera Calibration Software):

### Getting Started: Camera Calibration

This is the ChArUco board used to calibrate the camera. It must be printed and the length at the bottom checked to ensure it is the proper size to correctly calibrate the camera for the system.




When the system is first installed, it will ask to calibrate the camera. When the prompt appears, please print the document titled "Calibration_Printout.pdf" and ensure that the bar at the bottom of the page is 4 inches. If not, please ensure the print settings are set to scale the file at 100% before printing again.

Once this file has been printed, attach it to a rigid, flat object. This will be used to move the pattern in front of the camera and calibrate it. At the end of each step of calibration, a prompt will ask if you believe you were able to complete the motions described below. If not, you may select "Retry" to record the step again. If you believe you were able to successfully capture the movement outlined in the given step, please select "Continue". At the end of the calibration process, it will ask if you wish to complete the calibration. Select "Retry" to redo the previous step or select "Continue to complete calibration. Please note that this may take several minutes.

Table of Contents

Phase 1: Parallel ChArUco Board

Hold the board parallel to the lens of the camera and slowly move in the pattern below:

**Top View:**

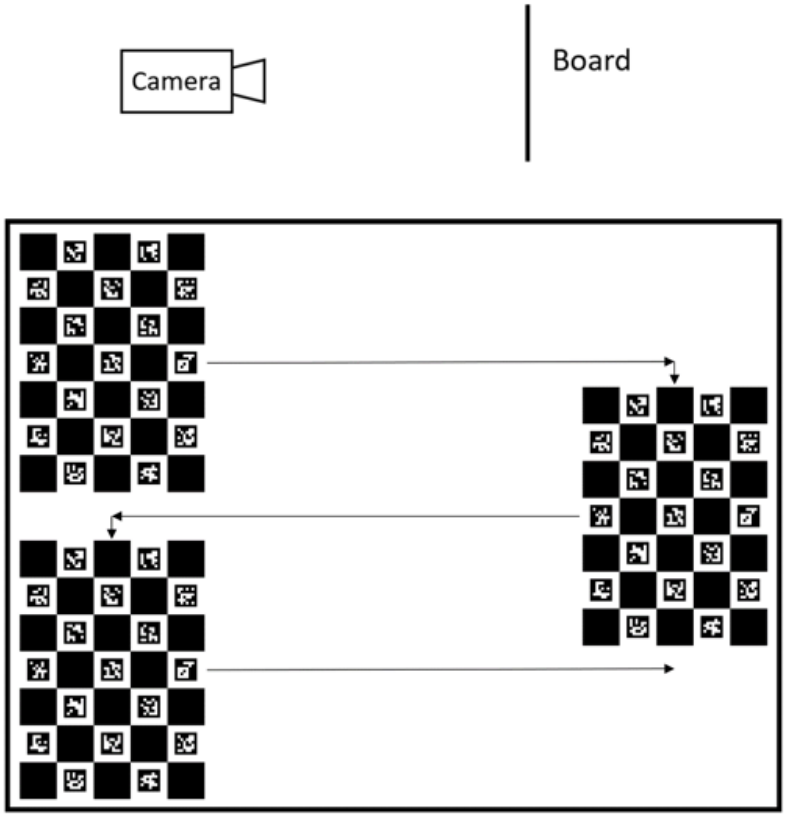



Table of Contents

### Phase 2: Positive Yaw

Next hold the board at a 45-degree angle to the camera and move the board around the camera's field of view as seen below:

**Top View:**

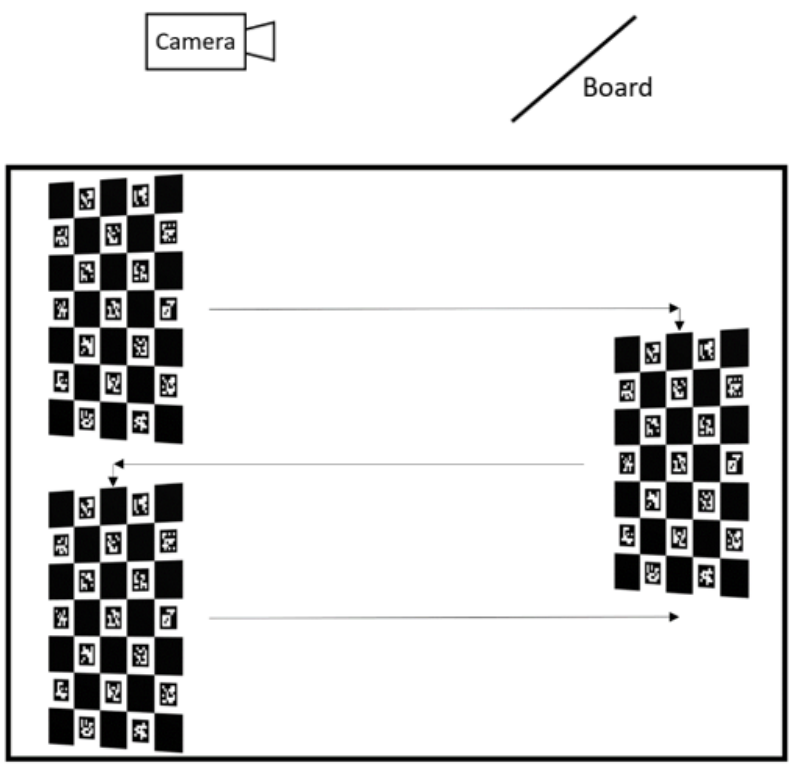



Table of Contents

### Phase 3: Negative Yaw

Shift the board so it has about a 45-degree angle such as in the diagram below, and repeat the process of moving it through the camera's field of view:

**Top View:**

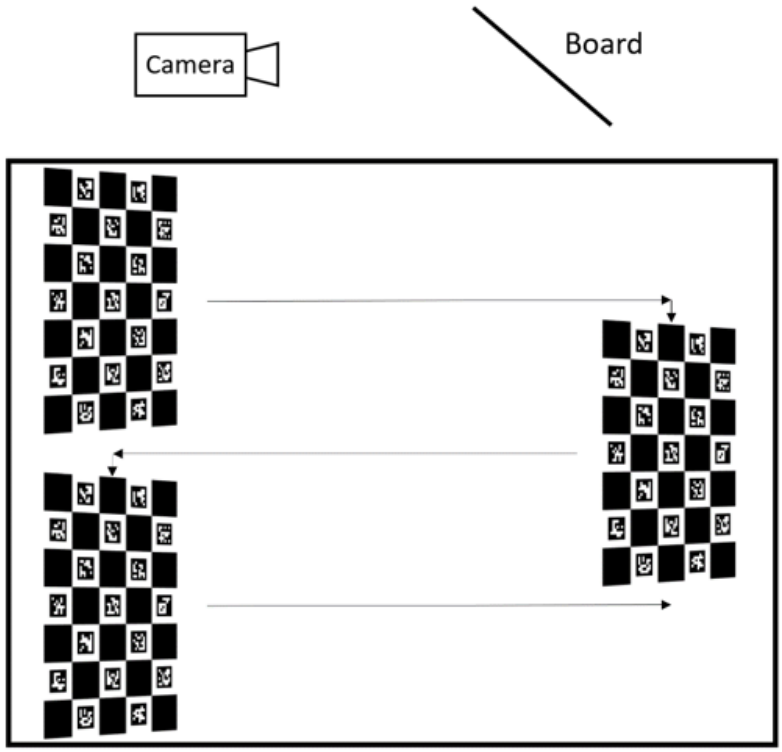



Table of Contents

### Phase 3: Positive Pitch

Shift the board so that it forms a 45-degree angle to the camera as shown below, and move it in the indicated pattern:

**Side View:**

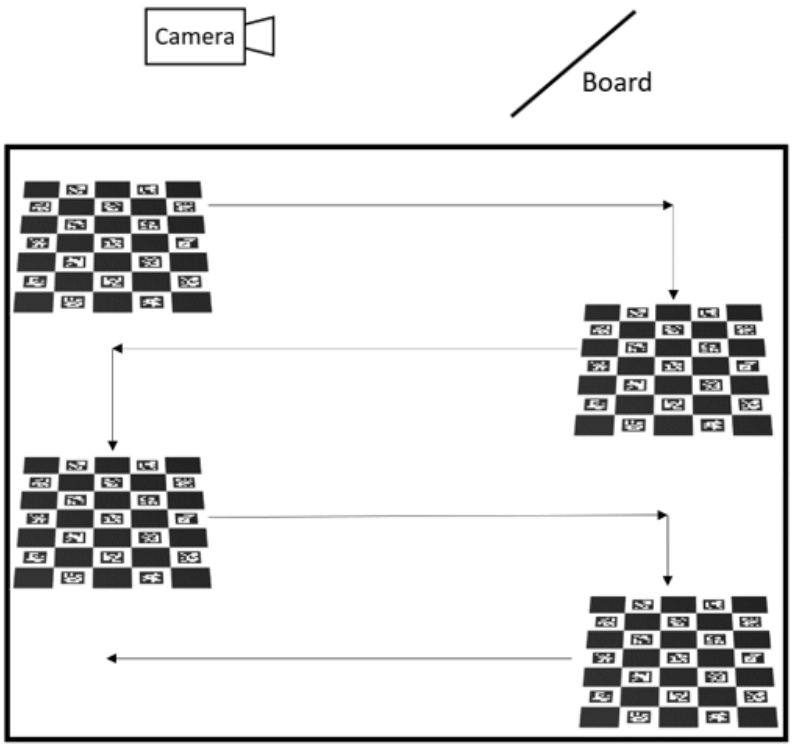



Table of Contents

### Phase 4: Negative Pitch

Shift the board once more to form a 45-degree angle as shown below, and move it in the indicated pattern:

**Side View:**

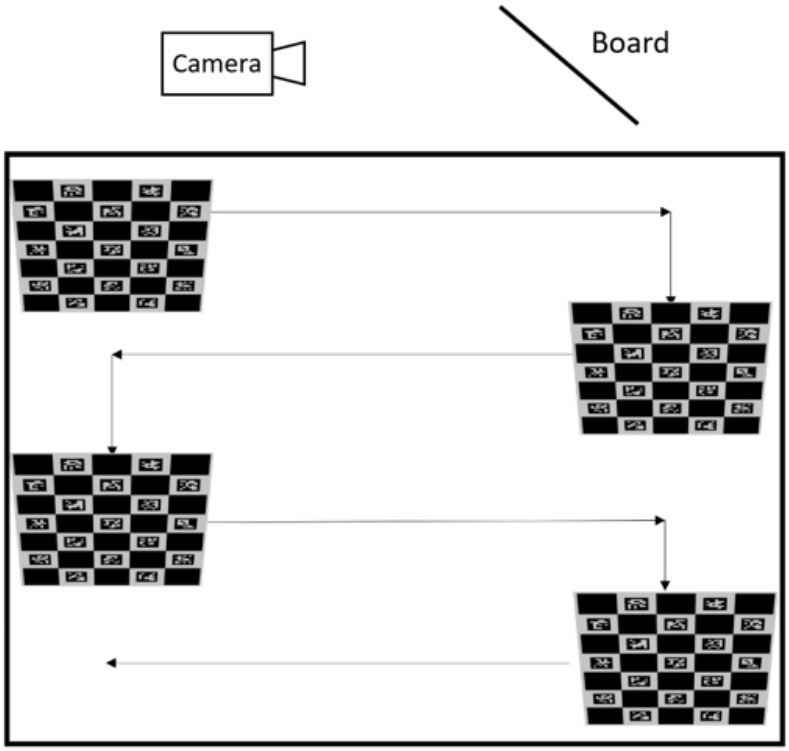



Table of Contents

### Phase 5: Maximum Perspective

Move the board so that one corner is closest the screen and the opposite corner is furthest from the camera. Repeat this process for all four corners.

**NOTE:** If not enough images have been captured at this point, continue moving the board as described above until the system automatically finishes the calibration process.

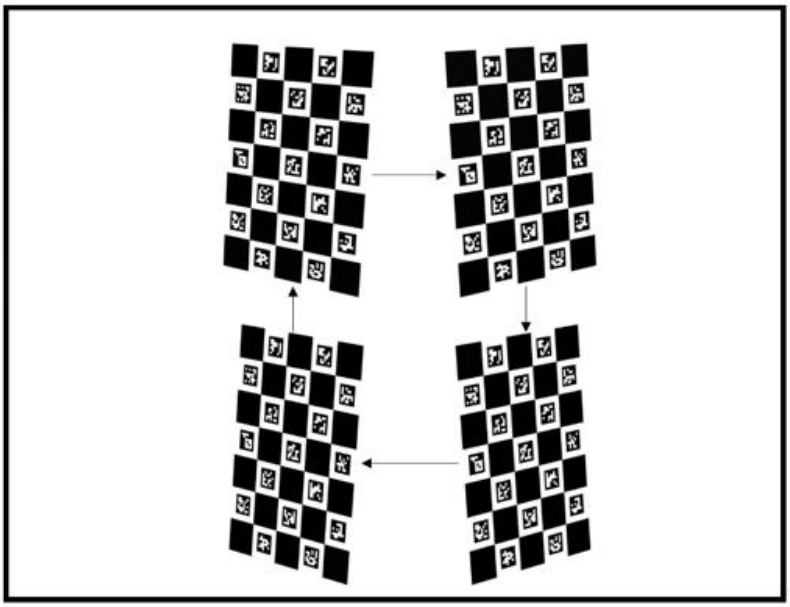


Table of Contents

## Marker Installation Instructions

To set up this system for your school, you must first assign names to some of the markers you will be posting around your school. There is a pre-set library that you may edit to accommodate the needs of your school, but you may create your own. There are 1000 markers available to identify objects within your school, and all you need to do is name them within your school's database. An example of markers that have been assigned labels are in *Table 3: Marker Naming Scheme*.

*Table 3: Marker Naming Scheme*

| ID# | Label |
|---|---|
| 1 | Sink |
| 2 | Toilet |
| 3 | Urinal |
| 4 | Water Fountain |

An example marker database that demonstrates how the marker dictionary's IDs can be assigned to locations and facilities within a school are in *Table 4: Example Marker Database for School*.

*Table 4: Example Marker Database for School*

| ID# Range | Labels |
|---|---|
| 0-199 | Common Objects and Locations (Bathroom door, water fountain, etc.) |
| 200-499 | Classroom doors |
| 500-999 | Locker Numbers |

The specific name of each label can be viewed and changed as needed through your school's administrative account. To print any given marker, please ensure the scale of the page is set to 100% and it is printed on standard 8.5" × 11" paper. To check the marker is the correct size, measure the bar located at the bottom of the page. If it is the same length as labeled, you have printed it correctly. If not, you will need to adjust your printer's settings and try again.

When the marker is placed within the building, make sure the arrow is pointing up and the marker is level on the wall. This will prevent any problems that may occur when finding the marker with the vision system.

Table of Contents

An example of an ArUco marker after it has been printed and is ready to post on the wall is in *Figure 17: ArUco Marker Labeled Printout*.

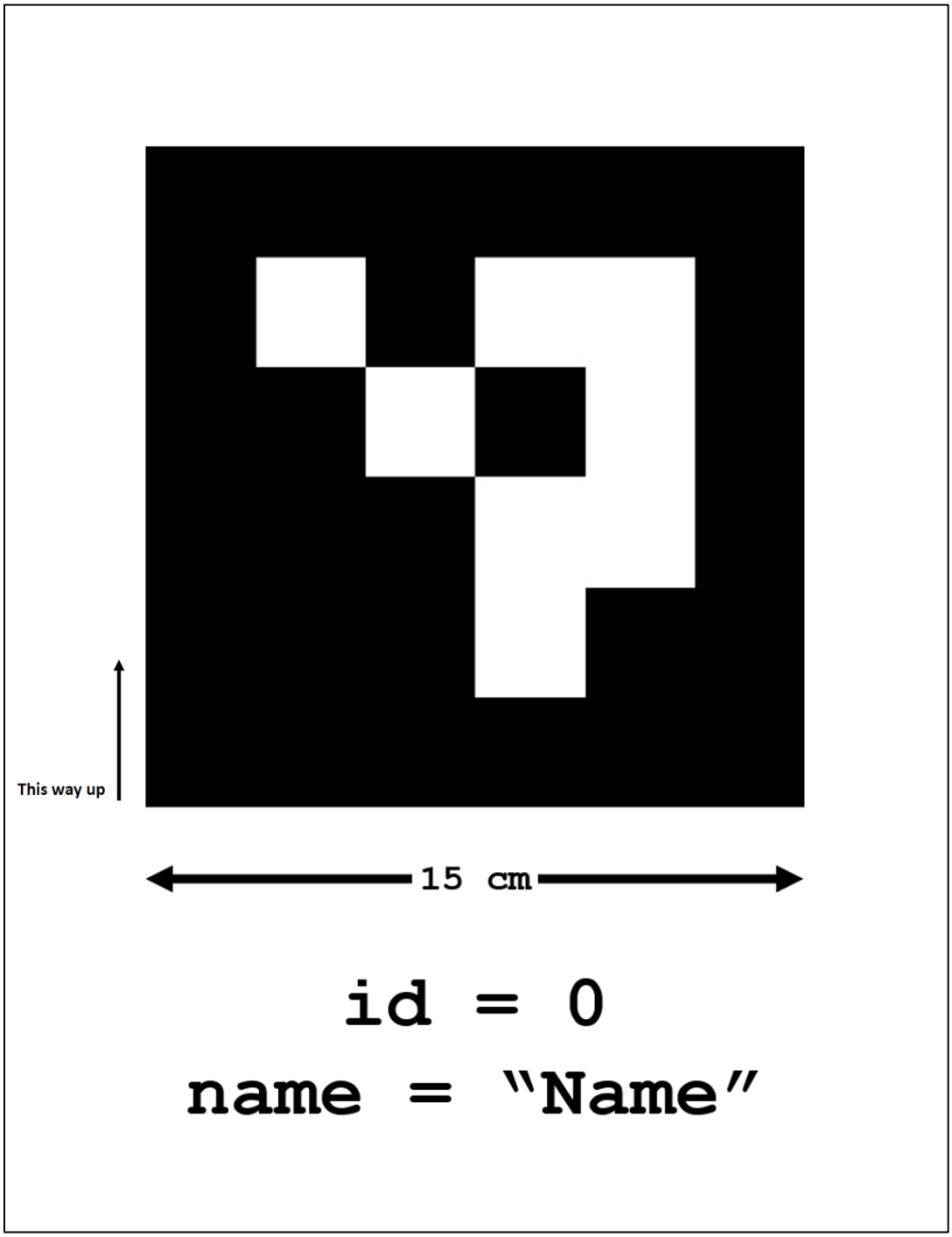



*Figure 17: ArUco Marker Labeled Printout*

Table of Contents

## Vision System Software

The operational vision system (excluding the initial one-time camera calibration process) was designed using C++ object-oriented programming techniques (as seen in *Figure 18: UML Class Diagram of Vision System*). With a VideoCapture device opened, a *Frame* object is constructed from each image and optionally with a specified ArUco marker dictionary. The constructor for a "Frame" object performs image processing (e.g., converting to grayscale), scans the grayscale image for single ArUco markers, stores relevant information in private member variables, performs pose estimation, and generates display images.

The pose estimation is converted from axis-angle (i.e., Euler angle) representation to a Rodrigues (3×3) rotation matrix, which is then put through an RQ Decomposition to obtain the Pitch, Yaw, and Roll of the camera. The distance between the camera and marker is also computed. All this information is displayed in the console window for debugging. The Yaw and Euclidean distance will be given to the Haptic Subsystem so it can provide directional feedback to the user.

Table of Contents

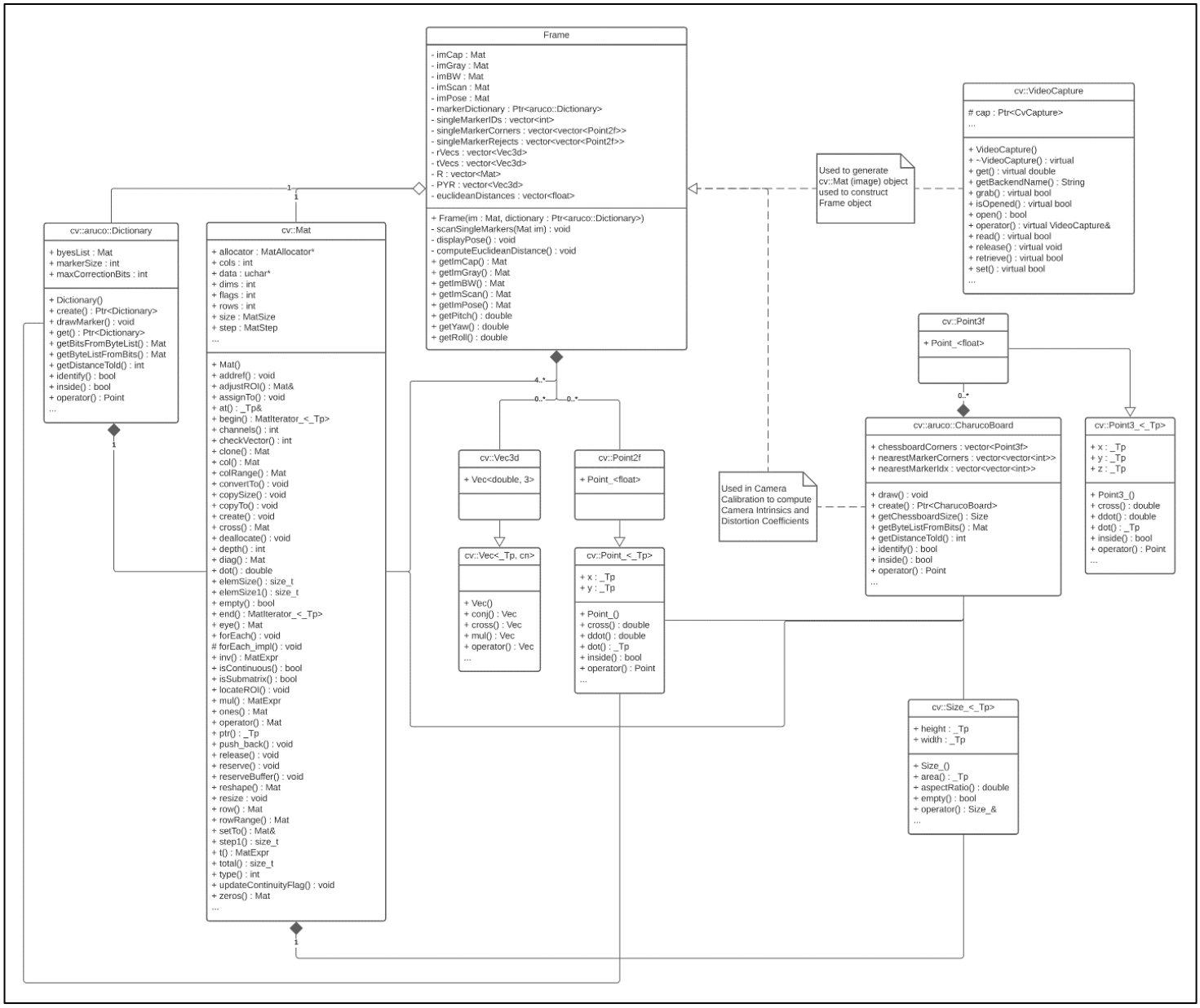



*Figure 18: UML Class Diagram of Vision System*

Table of Contents

## Frame.h

```
#pragma once
#include "opencv2/core.hpp"
#include "opencv2/imgcodecs.hpp"
#include "opencv2/imgproc.hpp"
#include "opencv2/highgui.hpp"
#include "opencv2/aruco.hpp"
#include "opencv2/calib3d.hpp"

#include <sstream>
#include <iostream>
#include <fstream>

using namespace std;
using namespace cv;

// Global Variables:
extern const float singleMarkerSideLength;
extern Mat cameraIntrinsics;
extern Mat distortionCoeffs;

class Frame
{
public:
	Frame(Mat im, Ptr<aruco::Dictionary> dictionary = aruco::getPredefinedDictionary(aruco::PREDEFINED_DICTIONARY_NAME::DICT_4X4_50));

	// Getters:
	Mat getImCap()
	{
		return imCap;
	}
	Mat getImGray()
	{
		return imGray;
	}
	Mat getImBW()
	{
		return imBW;
	}
	Mat getImScan()
	{
		return imScan;
	}
	Mat getImPose()
	{
		return imPose;
	}
	double getPitch(int i)
	{
		return -PYR[i][0];
	}
	double getYaw(int i)
	{
		return -PYR[i][1];
	}
	double getRoll(int i)
	{
		return -PYR[i][2];
	}

private:
	// Member Functions:
	void scanSingleMarkers(Mat im);

	void displayPose();

	void computeEuclidianDistances();

	// Images:
	Mat imCap;
	Mat imGray;
	Mat imBW;
	Mat imScan;
	Mat imPose;

	// Marker Information:
	Ptr<aruco::Dictionary> markerDictionary;
	vector<int> singleMarkerIDs;
	vector<vector<Point2f>> singleMarkerCorners, singleMarkerRejects;
	vector<Vec3d> rVecs, tVecs;
	vector<Mat> R;
	vector<Vec3d> PYR;
	vector<double> euclidianDistances;
};
```

## Frame.cpp

```
#include "Frame.h"

using namespace std;
using namespace cv;
```

Table of Contents

```
Frame::Frame(Mat im, Ptr<aruco::Dictionary> dictionary)
{
	markerDictionary = dictionary;
	// Image Processing
	imCap = im;
	cvtColor(im, imGray, COLOR_BGR2GRAY);
	//adaptiveThreshold(imGray, imBW, 255, ADAPTIVE_THRESH_GAUSSIAN_C, THRESH_BINARY, 11, 2);
	cvtColor(imGray, imScan, COLOR_GRAY2BGR);
	cvtColor(imGray, imPose, COLOR_GRAY2BGR);
	// Scan grayscale image for ArUco single markers and perform Pose Estimation on any detected markers
	scanSingleMarkers(imGray);
	// Display Pose Estimation data and draw marker axes
	displayPose();
}

void Frame::scanSingleMarkers(Mat im)
{
	Ptr<aruco::Dictionary> markerDictionary = aruco::getPredefinedDictionary(aruco::PREDEFINED_DICTIONARY_NAME::DICT_4X4_50);
	Ptr<aruco::DetectorParameters> detectParams = aruco::DetectorParameters::create();
	aruco::detectMarkers(im, markerDictionary, singleMarkerCorners, singleMarkerIDs, detectParams, singleMarkerRejects,
cameraIntrinsics, distortionCoeffs);
	aruco::drawDetectedMarkers(imScan, singleMarkerCorners, singleMarkerIDs);
	aruco::estimatePoseSingleMarkers(singleMarkerCorners, singleMarkerSideLength, cameraIntrinsics, distortionCoeffs, rVecs, tVecs);

	// Convert Axis-Angle representation of pose to Rodrigues Rotation Matrix, then perform RQ Decomposition to obtain Pitch, Yaw, and
Roll
	Mat Ri = Mat::zeros(Size(3, 3), CV_16F);
	Vec3d PYRi;
	Mat mtxR = Mat::zeros(Size(3, 3), CV_16F);
	Mat mtxQ = Mat::zeros(Size(3, 3), CV_16F);
	for (int i = 0; i < singleMarkerIDs.size(); i++)
	{
		Rodrigues(rVecs[i], Ri);
		R.push_back(Ri);
		PYRi = RQDecomp3x3(Ri, mtxR, mtxQ);
		PYR.push_back(PYRi);
	}

	// Calculate Euclidian distance from Translation Vector
	computeEuclidianDistances();
}

void Frame::displayPose()
{
	system("CLS");
	cout << "------------------------------------------------" << endl;
	cout << "\tDetected Markers = " << singleMarkerIDs.size() << endl;
	cout << "------------------------------------------------" << endl;
	// Display Pose Estimation Values and Axes
	for (int i = 0; i < singleMarkerIDs.size(); i++)
	{
		aruco::drawAxis(imPose, cameraIntrinsics, distortionCoeffs, rVecs[i], tVecs[i], 0.07f);
		cout << " ID = " << singleMarkerIDs[i] << endl;
		cout << " Rot Vec   = " << rVecs[i] << endl;
		cout << " Trans Vec = " << tVecs[i] << endl;
		cout << " Euclidian = " << euclidianDistances[i] << endl;
		cout << " Rot Matrix:" << endl << R[i] << endl;
		cout << " Pitch = " << getPitch(i) << endl;
		cout << " Yaw   = " << getYaw(i) << endl;
		cout << " Roll  = " << getRoll(i) << endl;
		cout << "------------------------------------------------" << endl;
	}
}

void Frame::computeEuclidianDistances()
{
	for (int i = 0; i < singleMarkerIDs.size(); i++)
	{
		euclidianDistances.push_back(sqrt(pow(tVecs[i][0], 2) + pow(tVecs[i][1], 2) + pow(tVecs[i][2], 2)));
	}

}
```

## Main.cpp

```
#include "opencv2/core.hpp"
#include "opencv2/imgcodecs.hpp"
#include "opencv2/imgproc.hpp"
#include "opencv2/highgui.hpp"
#include "opencv2/aruco.hpp"
#include "opencv2/calib3d.hpp"

#include <sstream>
#include <iostream>
#include <fstream>
#include <math.h>

#include "Frame.h"

using namespace std;
using namespace cv;

// Global Variables:
const float singleMarkerSideLength = 0.15f; //meters
Mat cameraIntrinsics = Mat::eye(3, 3, CV_64F);
Mat distortionCoeffs;

// Camera Calibration
bool loadCameraCalibration(string name, Mat& cameraIntrinsics, Mat& distortionCoeffs)
{
	// initialize data stream from calibraiton file
	ifstream inStream(name);

	// if the stream opens successfully (i.e. the file is found), then unpack info for distortions and camera matrix
	if (inStream)
	{
		uint16_t rows;
		uint16_t columns;

		// Camera Matrix
		inStream >> rows;
		inStream >> columns;

		cameraIntrinsics = Mat(Size(columns, rows), CV_64F);

		cout << "Camera Intrinsics:" << endl;
		for (int r = 0; r < rows; r++)
		{
			for (int c = 0; c < columns; c++)
			{
				double read = 0.0f;
				inStream >> read;
				cameraIntrinsics.at<double>(r, c) = read;
				cout << cameraIntrinsics.at<double>(r, c) << "\n";
			}
		}

		//Distortion Coeffs
		inStream >> rows;
		inStream >> columns;

		distortionCoeffs = Mat::zeros(rows, columns, CV_64F);

		cout << "Distortion Coefficients:" << endl;
		for (int r = 0; r < rows; r++)
		{
			for (int c = 0; c < columns; c++)
			{
				double read = 0.0f;
				inStream >> read;
				distortionCoeffs.at<double>(r, c) = read;
				cout << distortionCoeffs.at<double>(r, c) << "\n";
			}
		}

		// close input stream
		inStream.close();

		// return successfully completed task
		return true;
	}

	return false;
}

void createArucoMarkers(Ptr<aruco::Dictionary> markerDictionary =
aruco::getPredefinedDictionary(aruco::PREDEFINED_DICTIONARY_NAME::DICT_4X4_1000))
{
	Mat outputMarker;

	for (int i = 0; i < 1024; i++)
	{
		aruco::drawMarker(markerDictionary, i, 1000, outputMarker, 1);
```

```
                    ostringstream convert;
                    string imageName = "ARUCO_ORIGINAL_";
                    convert << imageName << i << ".png";
                    imwrite(convert.str(), outputMarker);
          }
}

int main()
{
          // Open VideoCapture device:
          VideoCapture cap(0);
          if (!cap.isOpened())
          {
                    cout << "Error: Camera already opened." << endl;
                    return -1;
          }

          // Open Windows:
          namedWindow("imCap", WINDOW_AUTOSIZE);
          namedWindow("imGray", WINDOW_AUTOSIZE);
          //namedWindow("imBW", WINDOW_AUTOSIZE);
          namedWindow("imScan", WINDOW_AUTOSIZE);
          namedWindow("imPose", WINDOW_AUTOSIZE);

          // Load Camera Parameters:
          loadCameraCalibration("CharCaliOut", cameraIntrinsics, distortionCoeffs);

          // Initialize Current Capture:
          Mat currentCap;

          while (true)
          {
                    // if no frame returned from webcam (unable to get info from webcam)
                    if (!cap.read(currentCap))
                              break;

                    // Create Frame object:
                    Frame currentFrame(currentCap);

                    // Display to Windows:
                    imshow("imCap", currentFrame.getImCap());
                    imshow("imGray", currentFrame.getImGray());
                    //imshow("imBW", currentFrame.getImBW());
                    imshow("imScan", currentFrame.getImScan());
                    imshow("imPose", currentFrame.getImPose());

                    // WaitKey:
                    if (waitKey(1) >= 0) break;
          }

          return 1;
}
```

# Interface

## Bluetooth Communications

The Arduino Uno will have a serial Bluetooth adapter attached; this adapter will communicate with the built in Bluetooth adapter in an Android device. These adapters will serve as the interface for the serial communications between the two devices. The serial protocol is detailed below.

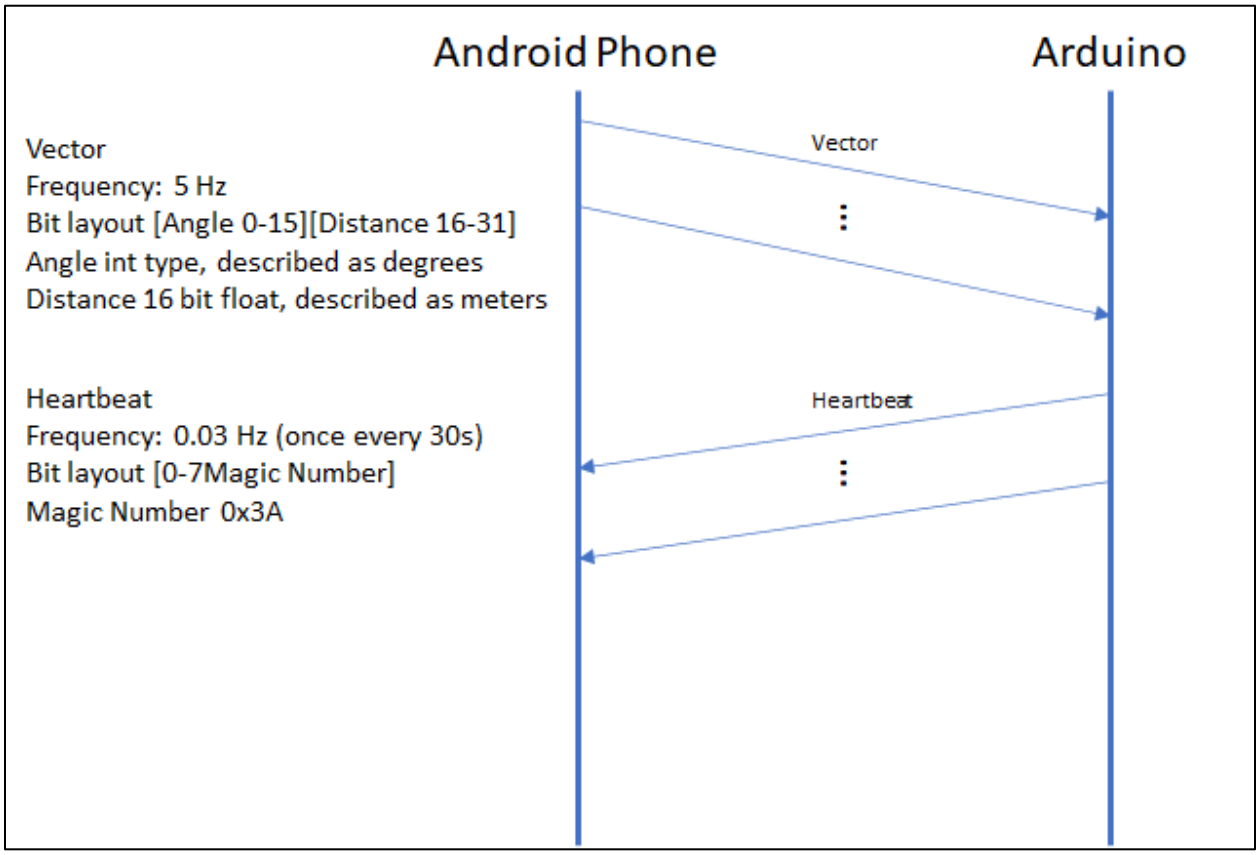



*Figure 19: Bluetooth Communications Diagram*

The vector is sent at a rate of 5Hz because that is the highest rate that the Arduino will update when the user is closest to the marker. Any Frequency above this would be a waste of bandwidth and power, as the data would go unused.

The heartbeat sent from the Arduino is meant to confirm connectivity between the smartphone and Arduino for the user.

Table of Contents

## Android Application

The Android Application serves as the user interface between the ArUco marker detection library and the Arduino haptic system. The application provides a user interface for connecting the smartphone's Bluetooth to the Arduino's Bluetooth, calibrating the camera for use by the ArUco library, and for selecting the desired point of interest. These three aspects are crucial to being able to find a point of interest in a room. Android has developed a suite of accessible design guidelines to help developers create more accessible applications [27]. The user interface was designed to only have the required elements present, as too many features could be confusing when using the accessibility functions on Android. The colours and aesthetics of the application are subject to change as they are not considered critical functional aspects of the design, and we do not have a graphic designer.

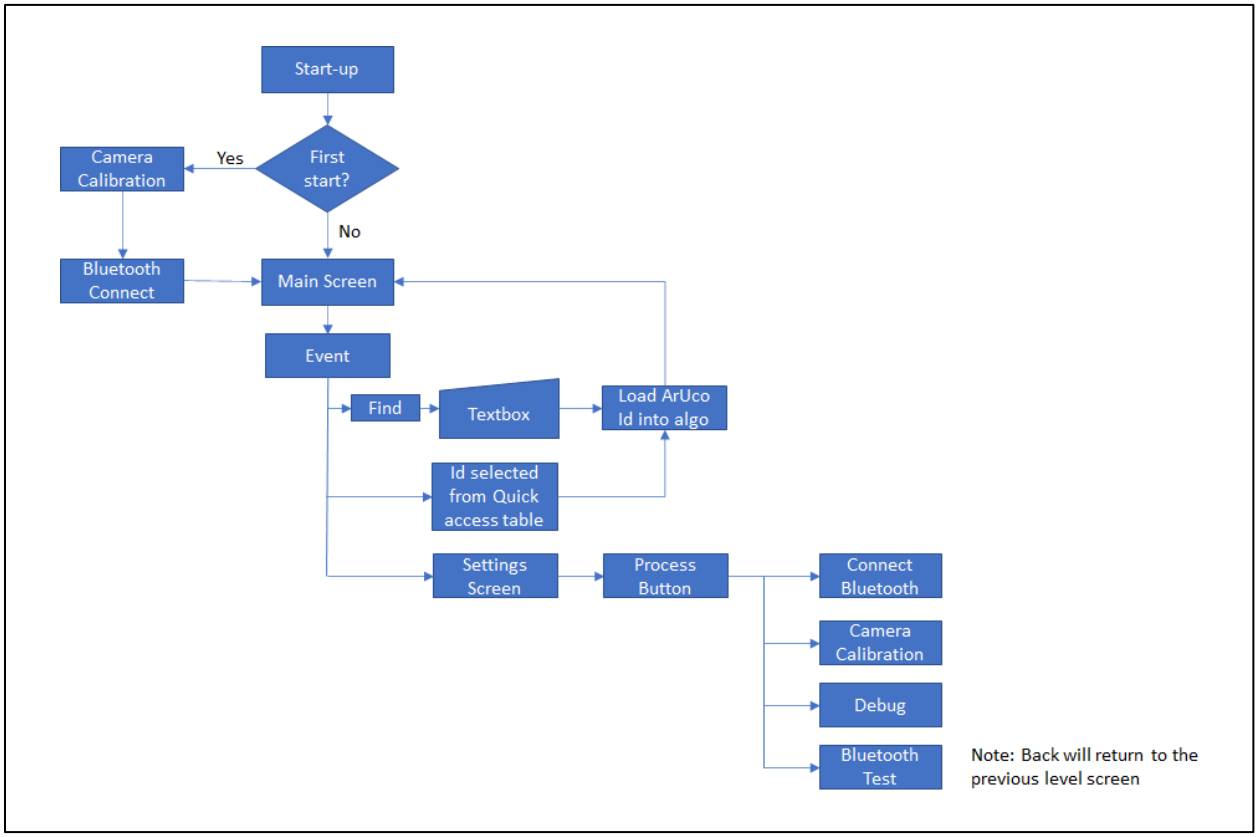
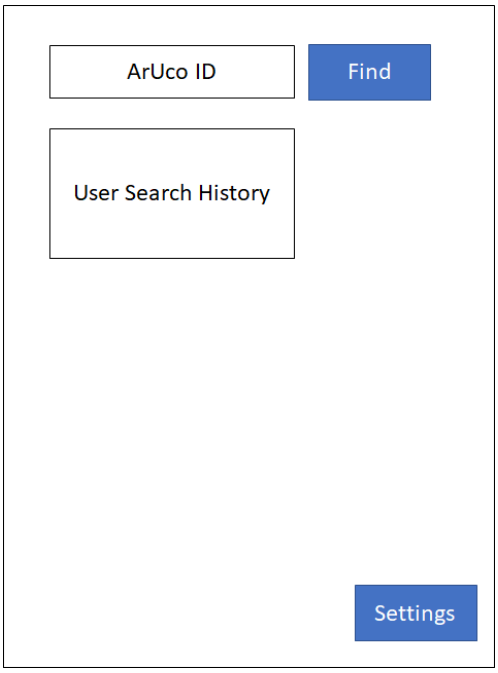



*Figure 20: Android Application Flowchart*

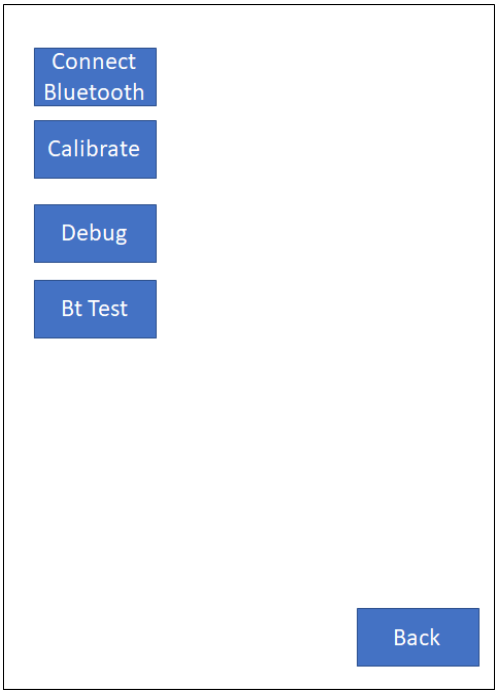
Table of Contents

## Main Screen




**ArUco ID**: A plain English identifier for the desired point of interest. E.g. “locker 200”
**Find:** Loads the ID in the “ArUco ID” textbox into the algorithm to find the desired ID
**User Search History:** A table of user searches, selecting a marker from this list will load the ID into the algorithm.
**Settings:** Navigates to the settings screen

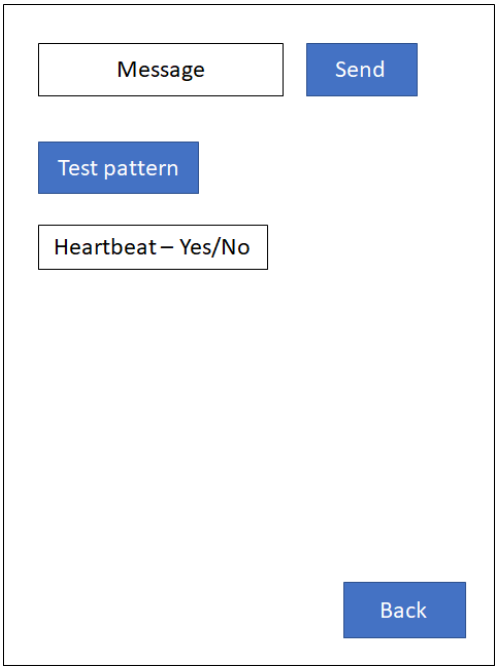
## Settings Screen




**Connect**: Will connect to the Arduino via Bluetooth
**Calibrate**: Runs the calibration program for ArUco
**Debug**: Goes to the debug screen (prototype only)
**Bt Test:** Goes to the “Bt Test Screen”
**Back:** Go back to the “Main Screen”

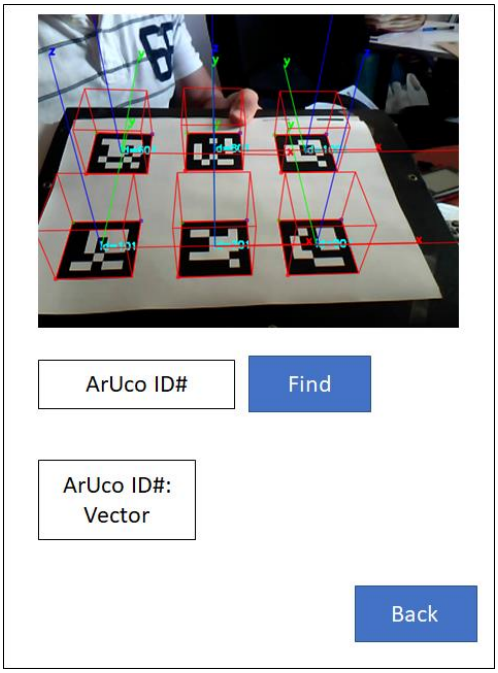
Table of Contents

## Bluetooth Test Screen




**Message:** Desired message to be displayed on Arduino's serial port
**Send:** Send the message in the "Message" textbox
**Test pattern:** Send a request to the Ardiuno to output a test pattern on the haptic wearable
**Heartbeat:** Confirms if there has been a heartbeat received or not
**Back:** Goes back to the settings screen

## Debug Screen




**ArUco ID**: A plain English identifier for the desired point of interest. E.g. "locker 200"
**Find:** Loads the ID in the "ArUco ID" textbox into the algorithm to find the desired ID
**ArUco ID#: Vector:** Displays the current ArUco ID vector, will also draw it on the video screen.
**Back:** Goes back to the "Settings Screen"

Table of Contents

# Haptic Subsystem

## Mechanical

### Armband

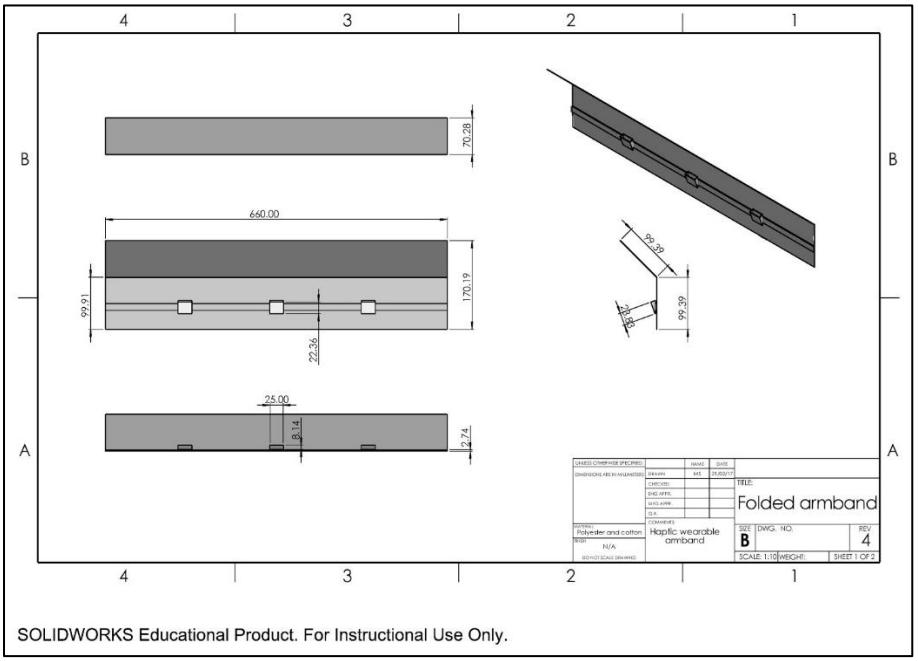



*Figure 21: Folded Armband with Actuator Pouches CAD Drawing*

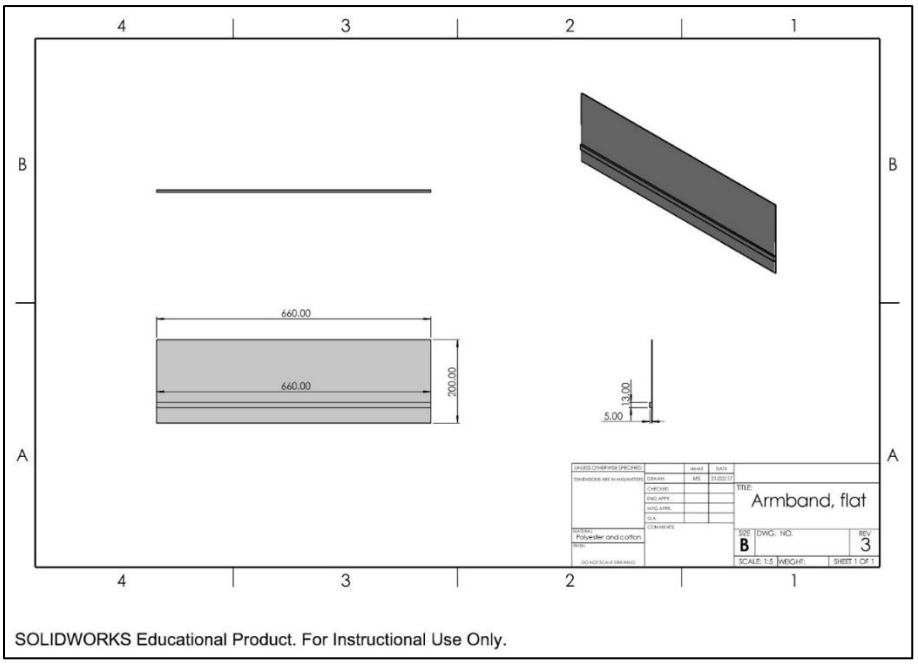



*Figure 22: Flat Armband without Actuator Pouches CAD Drawing*

Table of Contents

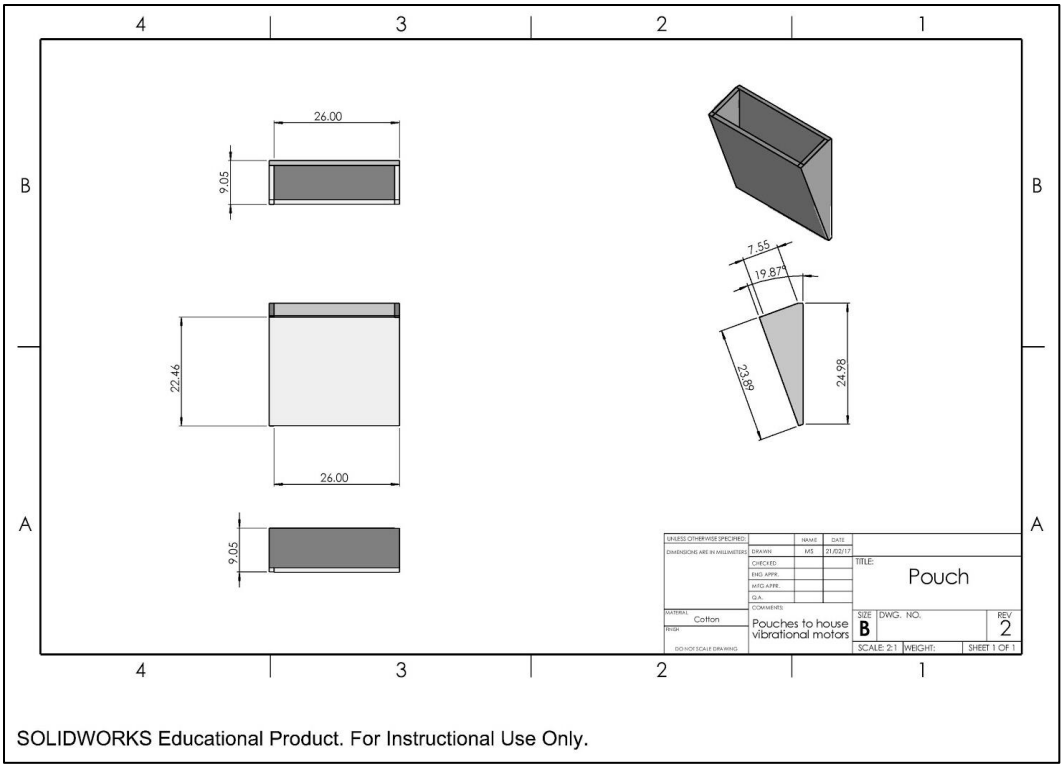



*Figure 23: Actuator Pouches CAD Drawing*

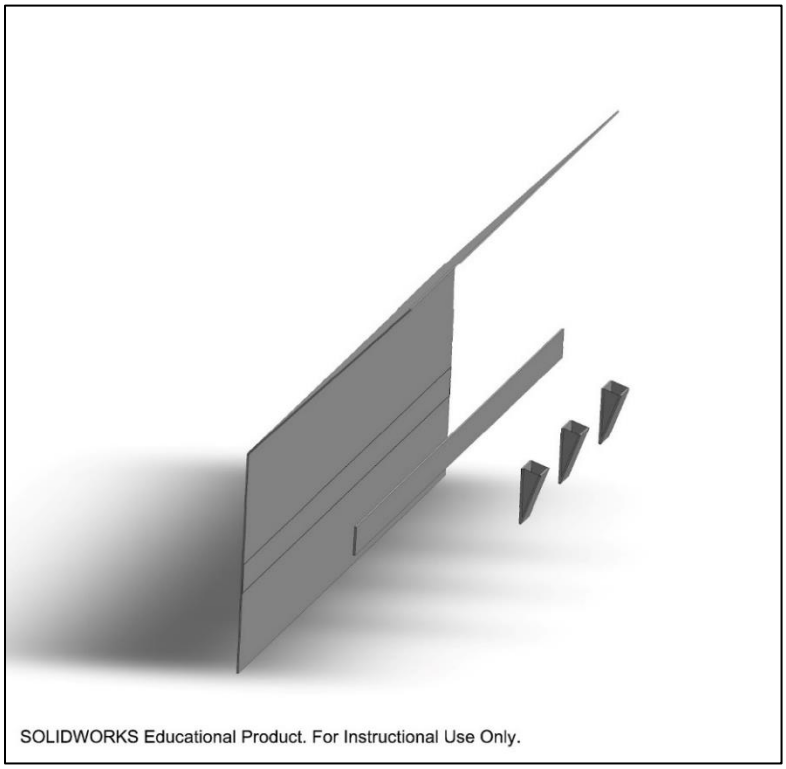



*Figure 24: Folded Armband with Actuator Pouches Exploded View*

Table of Contents

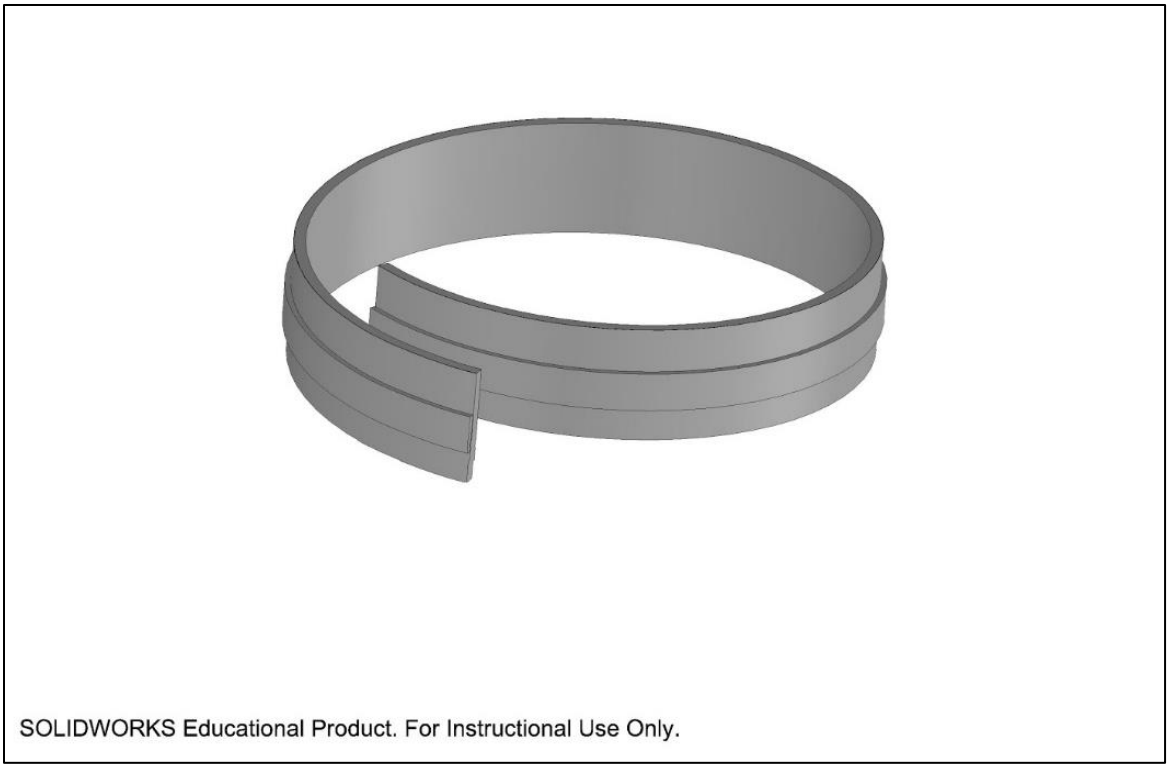



*Figure 25: Haptic Armband While Worn View*

## Assembly Instructions for Haptic Wearable Armband

1. Cut a length of fabric 660mm by 200mm
2. Glue, sew, or otherwise adhere soft Velcro along the 660mm length of the fabric, 50mm above the bottom of the fabric
3. Fold the fabric in half at 100mm and sew the two ends together
4. Cut two squares of fabric 26mm by 26mm. Repeat two more times for each actuator
5. Layer the two squares together and sew three edges together. Repeat two more times
6. Glue, sew, or otherwise adhere hard Velcro to the back of the three created pouches
7. Peel sticker off the back of the vibrational actuators to expose their adhesive side and insert each actuator into its pouch, pressing down on them to secure it with the adhesive
8. Place the pouches along the soft Velcro of the arm band in their desired positions, adhering to the 3cm minimum distance requirement for each actuator
9. Tie the armband to the upper arm
10. Readjust actuator pouches as needed
11. Connect the 2pin connectors from the actuator pouches to the enclosure

Table of Contents

## Belt Enclosure

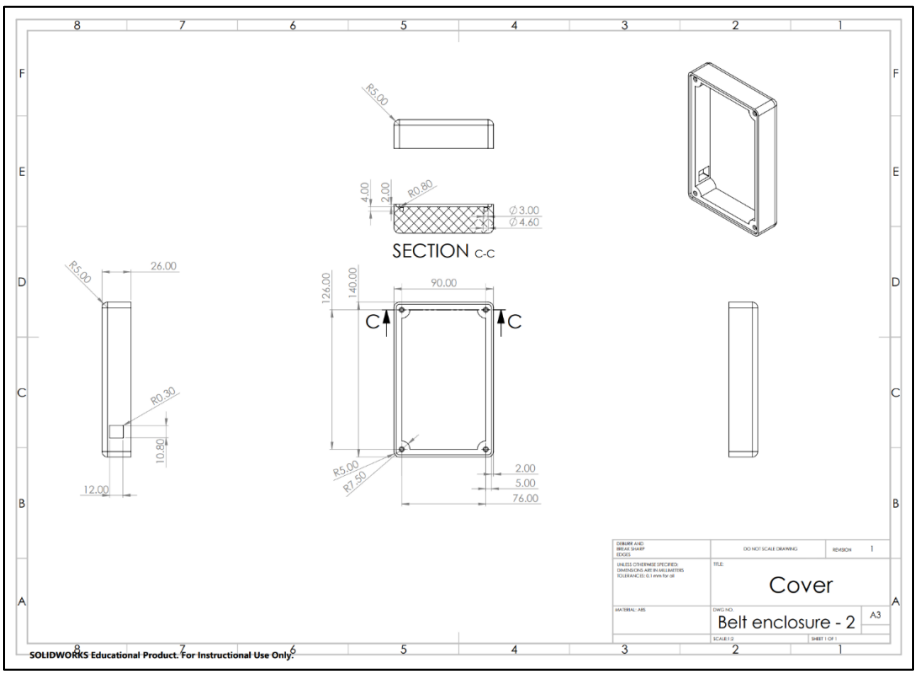



*Figure 26: Belt Enclosure side 1 (without clip)*

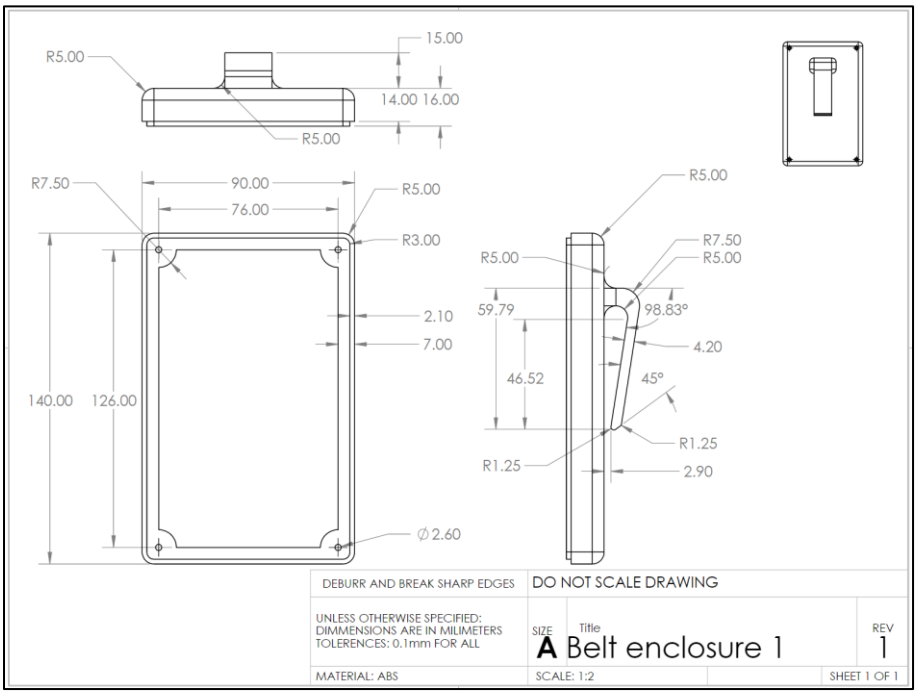



*Figure 27: Belt Enclosure side 2 (with clip)*

Table of Contents

### Manufacturing/Assembly Instructions for Belt Enclosure

The enclosure is manufactured via injection molding of ABS, ABS PC, or a similar plastic.

1. Fill up the battery holder with 6 × BH12-2300 Batteries
2. Press the 4 x M2 inserts into their respective positions in each corner of the clipless board
3. Fish the bundle of cable connecting to the motors through the hold on the side with a clip (left in the below diagram)
4. Apply heat resistant epoxy resin to the breakout board, microcontroller and charging board.
5. Place these components in their respective positions, as shown by the diagram below:

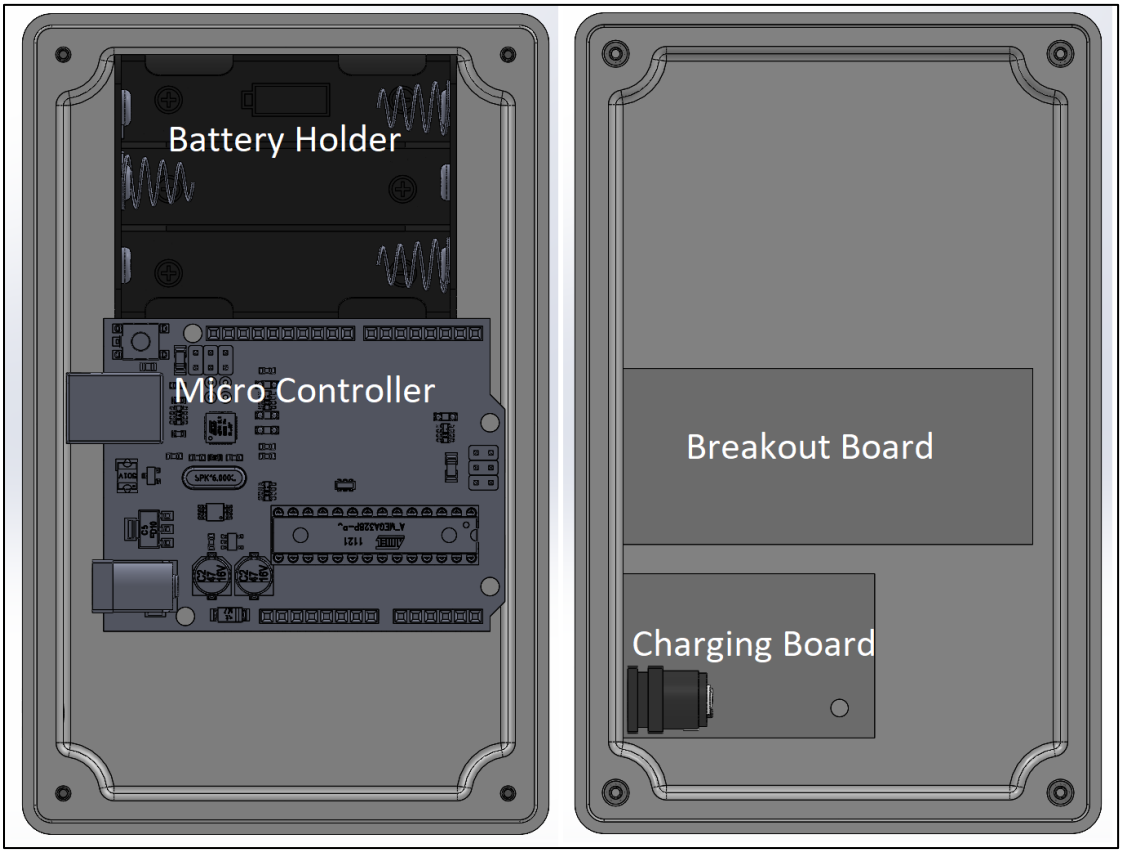



*Figure 28: Interior Layout of the belt enclosure, clip is left, no clip is right*

6. The specific position of each component is not critical, with the exclusion of the charging board, which needs to be positioned such that the connector is aligned with the open hole
7. Allow the Epoxy resin to set
8. Fit the two sides of the enclosure together and fasten them with 4 × M2 screws

# Electrical

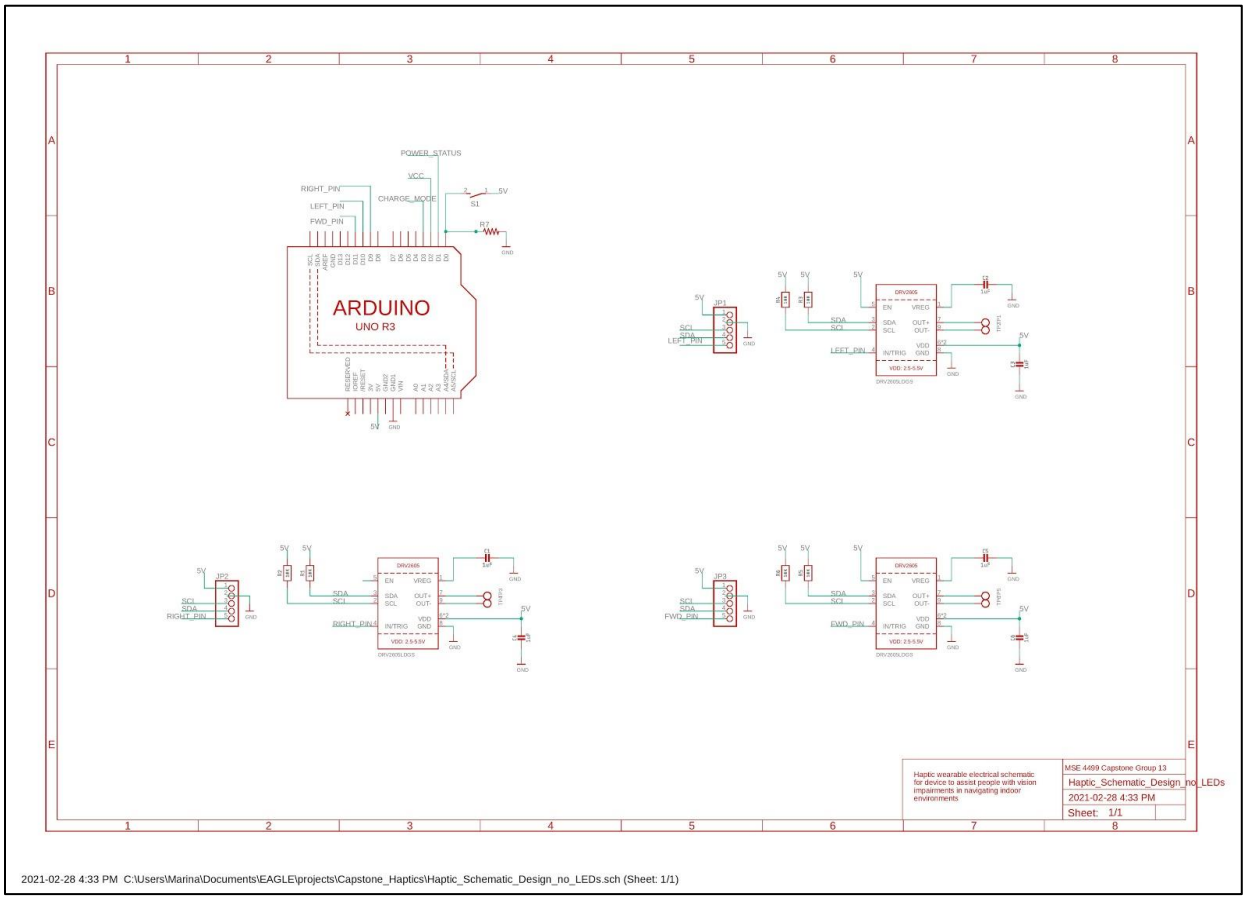



*Figure 29: Haptic Subsystem Electrical Schematic*

Table of Contents

## Charging Board

The charging board is placed in the case with the Arduino and battery pack. It charges the batteries using an external wall adapter and routes power to the microcontroller. Below is the bill of materials for the board used to charge the batteries.

*Table 5: Charging Board BOM*

| Part Name | Manufacturer | Part Number | Quantity |
|---|---|---|---|
| **Charging Board** | | | |
| MAX712CSE+T | Maxim Integrated | MAX712CSE+TTR-ND | 1 |
| **Resistors** | | | |
| 1Ω | TE Connectivity | A129236TR-ND | 1 |
| 150Ω | TE Connectivity | A126421TR-ND | 1 |
| 68kΩ | TE Connectivity | A103234TR-ND | 1 |
| 22kΩ | TE Connectivity | A103178TR-ND | 1 |
| 10kΩ | TE Connectivity | A103142TR-ND | 1 |
| **Capacitors** | | | |
| 1μF | Murata Electronics | 490-8043-2-ND | 1 |
| 0.01μF | Taiyo Yuden | 587-1114-2-ND | 2 |
| 10μF | AVX Corporation | 478-0805YC106KAT2ATR-ND | 1 |
| **Diode** | AVX Corporation | 478-7800-2-ND | 1 |
| **Transistor** | ON Superconductor | 2N6107GOS-ND | 1 |
| **Charging Port** | CUI Devices | CP-048H-ND | 1 |
| **Wall Adapter** | Outtag | PA-30150W-ZMX | 1 |
| **PCB** | PCB Way | - | 1 |
| **Button** | Omron Electronics | SW402-ND | 1 |

All surface-mount capacitors and resistors use the 0805 (metric 2012) form factor to minimize the volume of the PCB. All components have been specified as per the documentation for Maxim Integrated's MAX712CSE+TTR-ND control board. The transistor and charging port will be attached via through-holes on the PCB. All other components will mount to the surface of the PCB.

Table of Contents

This is the circuit schematic of the charger PCB. This board takes 10V DC from a wall converter and charges the batteries. When charging is completed, power is routed to the microcontroller from the batteries through the charging board. The output pins ‘LOAD-’ and ‘LOAD+’ connect to the corresponding positive and negative ports of the microcontroller, the pins ‘BATT-” and ‘BATT+’ connect to the corresponding negative and positive ports of the battery pack, and ‘VCC+’, ‘CHRG_MD’, and ‘PWR_ST’ are connected to the microcontroller and used to monitor the status of the charging board.

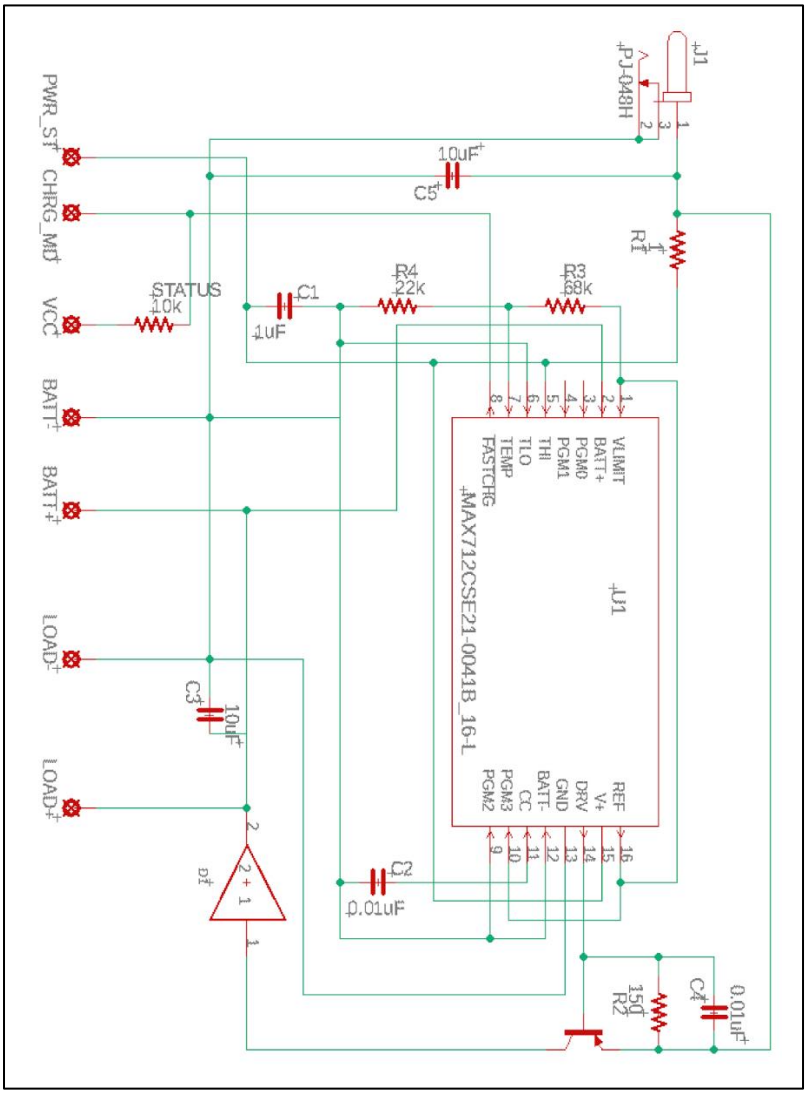



*Figure 30: Charger PCB Circuit Schematic*

The PCB layout is below. The final board is 43mm x 28mm and features a through hole with a diameter of 3.3mm. A screw will be placed through this hole to secure the board to the device case. This will prevent the board from turning or becoming displaced, which would prevent the device from being charged. This PCB will have two layers and will be 1.6mm thick. The through holes for the connections to the microcontroller and battery pack have a hole diameter of 1mm and have a pad diameter of 2.3mm.

Table of Contents

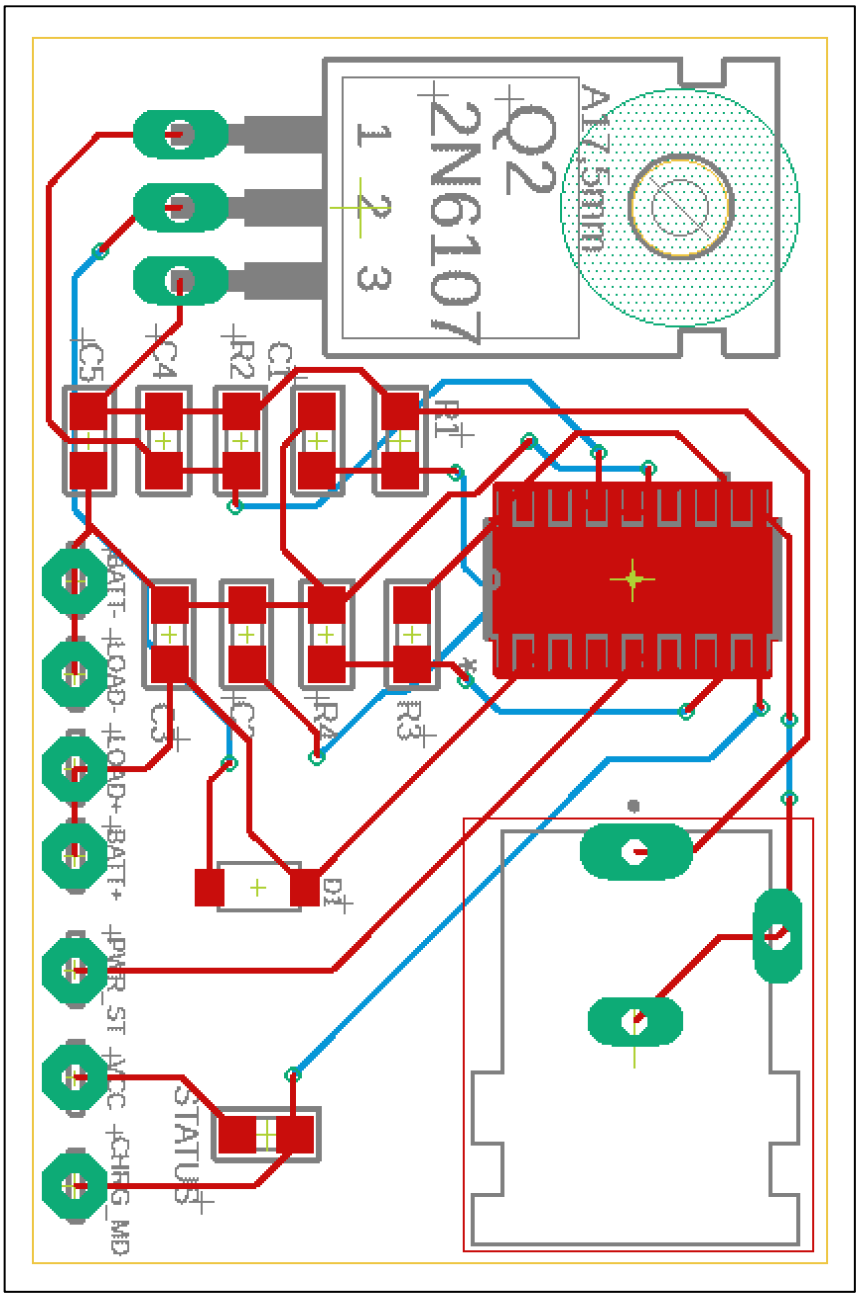



*Figure 31: Charger PCB Layout*

This diagram shows how the power is routed through the charging board to the microcontroller, and how the charging board connects to the microcontroller to report its status:

Table of Contents

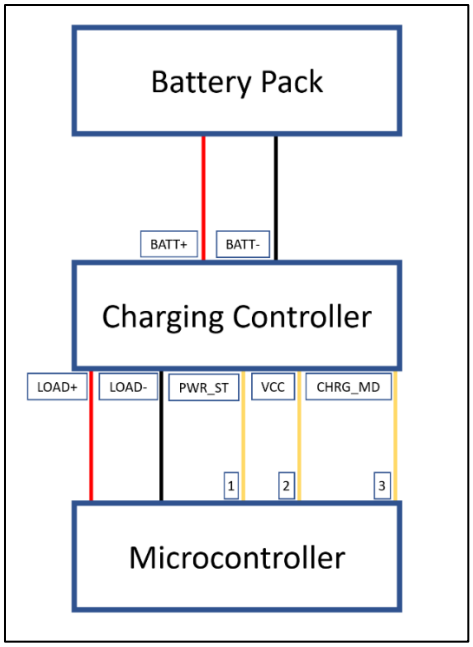



*Figure 32: Power Supply Routing*

A section of the code on the Arduino used for the haptics system checks the status of the charging board and alerts the user. The following flow chart describes its behaviour:

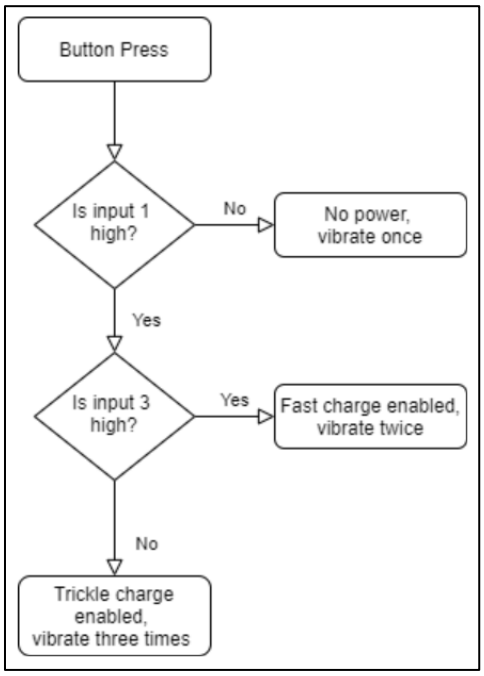



*Figure 33: Arduino Battery Charging Status Program Flowchart*

Table of Contents

## Software

The following diagram outlines the general structure of the simple code that was developed to perform the guidance, from the data provided by the Bluetooth module.

### Haptic Subsystem Functional Flowchart

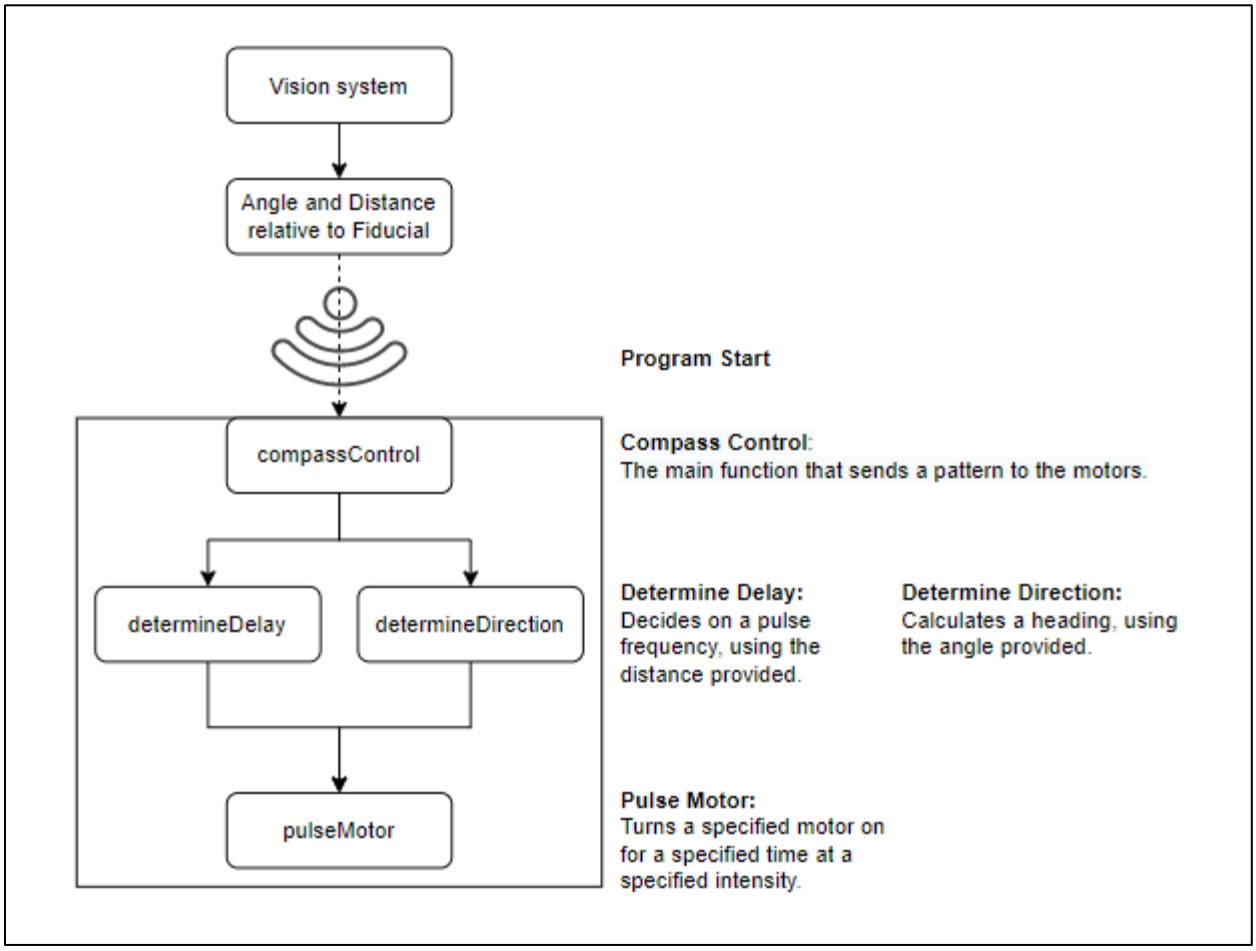



*Figure 34: Haptic Subsystem Arduino Code Functions Flowchart*

Table of Contents

## Haptic Subsystem Arduino Code

```
#include "Wire.h"
#include "Adafruit_DRV2605.h"
#include "math.h"
#define TCAADDR 0x70
extern "C" {
#include "utility/twi.h"  // from Wire library, so we can do bus scanning
}

  int northPin = 11;
  int westPin = 10;
  int eastPin = 9;
  int off = 100;
  int baseBpm = 100;
  int standardDistance = 5;

void setup() {
  // put your setup code here, to run once:
  //MOVED FROM OUTSIDE SETUP TO INSIDE |--->
  Adafruit_DRV2605 drv;

  // <---|
  pinMode(eastPin, OUTPUT);
  pinMode(northPin, OUTPUT);
  pinMode(westPin, OUTPUT);
  pinMode (0, INPUT);
  pinMode (1, INPUT);
  pinMode (2, INPUT);
  pinMode (3, INPUT);

  Serial.begin(9600);
  drv.begin();
  drv.setMode(DRV2605_MODE_PWMANALOG);

  // ac coupled input, puts in 0.9V bias
  drv.writeRegister8(DRV2605_REG_CONTROL1, 0x20);

  // analog input
  drv.writeRegister8(DRV2605_REG_CONTROL3, 0xA3);
  Serial.print("In setup");
  Serial.println();

  // turn them all off to start
  analogWrite(eastPin, off);
  analogWrite(northPin, off);
  analogWrite(westPin, off);
}

int determineDelay(double bpm, double distance, double unalteredDistance, double noteDelayDenominator){//Bpm, distance in meters,
distance where bpm= stays the same, and the factor which the delay is going to be divided by. 1=quarter, 2=eighth.

double distanceDiff = unalteredDistance - distance;
double bpmScale = 50*distanceDiff;
double bpmTotal = bpm+bpmScale;
double beats = 60000/bpmTotal;
double delayItNow = beats/noteDelayDenominator;

Serial.print(distanceDiff);
Serial.println();
Serial.print(bpmScale);
Serial.println();
Serial.print(bpmTotal);
Serial.println();
Serial.print(beats);
Serial.println();
Serial.print(delayItNow);
Serial.println();
return (int)delayItNow;

//return ((60, 000 / bpm) + (bpmScalingFactor * (unalteredDistance-distance))/noteDelayDenominator);
}

int determineDelay(int bpm, int distance, int unalteredDistance){ //Bpm, distance in meters, distance where bpm= stays the same.
QuarterNote unless extra param is specified. {
int bpmScalingFactor = 100;
return ((60, 000 / bpm) + (bpmScalingFactor * (unalteredDistance - distance)));
}

int compassControl(int onIntensity, double angle, int distance, int currentTime) {
  //angle in degrees, distance in cm, currentTime in ms
  int tunedDelay = determineDelay(baseBpm, distance, standardDistance,2);
  Serial.print("Tuned Delay: ");
  Serial.print(tunedDelay);
  Serial.println();
  String comDirection = determineDirection(angle);
  Serial.print("In compass");
  Serial.println();

  for (int i = 0; i < comDirection.length(); i++)
  {
    Serial.print(comDirection.charAt(i));
    Serial.println();
    switch (comDirection.charAt(i)) {
```

Table of Contents

```
      case 'N':
        pulseMotor(northPin,onIntensity,tunedDelay);
        delay(tunedDelay);
        break;
      case 'E':
        pulseMotor(eastPin,onIntensity,tunedDelay);
        delay(tunedDelay);
        break;
      case 'W':
        pulseMotor(westPin,onIntensity,tunedDelay);
        delay(tunedDelay);
        break;
      default:
        delay(2*tunedDelay);
        break;
    }
  }

}

void pulseMotor(int pin, int intensity, int dTime){
int realintensity = intensity*(155/100) + 100;
analogWrite(pin,realintensity);
delay(dTime);
analogWrite(pin,off);

}

String determineDirection(double angle) {
  //takes angle and determines closest direction.
  // ZERO FOR NOW IS TO TURN OFF. CHANGE BACK TO >= LATER.
  if ((angle > 0 && angle < 11.25) || (angle > 348.75 && angle < 360))
  {
    return "EEE_";
    Serial.print("Going EEE");
    Serial.println();
  } else if (angle >= 11.25 && angle < 33.75)
  {
    return "EEN_";
    Serial.print("Going EEN");
    Serial.println();
  } else if (angle >= 33.75 && angle < 56.25)
  {
    return "NE__";
    Serial.print("Going NE");
    Serial.println();
  } else if (angle >= 56.25 && angle < 78.75)
  {
    return "NNE_";
    Serial.print("Going NNE");
    Serial.println();
  } else if (angle > 78.75 && angle < 101.25)
  {
    return "NNN_";
    Serial.print("Going NNN");
    Serial.println();
  } else if (angle >= 101.25 && angle < 123.75)
  {
    return "NNW_";
    Serial.print("Going NNW");
    Serial.println();
  } else if (angle >= 123.75 && angle < 146.25)
  {
    return "NW__";
    Serial.print("Going NW");
    Serial.println();
  } else if (angle >= 146.25 && angle < 168.75)
  {
    return "WWN_";
    Serial.print("Going WWN");
    Serial.println();
  } else if (angle > 168.75 && angle < 270)
  {
    return "WWW_";
    Serial.print("Going WWW");
    Serial.println();
  } else if (angle >= 270 && angle < 348.75)
  {
    return "EEE_";
    Serial.print("Going EEE");
    Serial.println();
  }
  //If no given heading this error string will be returned
  return "XxxX";
}


void loop() {
  double inputAngle;

  int cTime = millis();
  int onIntensity=100;
  int distance;

  //take angle and distance
```

Table of Contents

```
  //determine

  if (Serial.available() > 0) {
    inputAngle = Serial.parseInt();
    Serial.print("Input Angle: ");
    Serial.print(inputAngle);
    Serial.println();
    delay(1000);
  }
  if (Serial.available() > 0) {
    distance = Serial.parseInt();
    Serial.print("Distance: ");
    Serial.print(distance);
    Serial.println();
    delay(1000);
  }
  Serial.print("I live.");
  Serial.println();
  compassControl(onIntensity, inputAngle, distance, cTime);

  //check battery charge
  checkCharge = digitalRead(0);

  if (checkCharge == HIGH){
    if (digitalRead(1)==LOW){
      // buzz once
      pulseMotor(northPin,onIntensity,tunedDelay);
      pulseMotor(eastPin,onIntensity,tunedDelay);
      pulseMotor(westPin,onIntensity,tunedDelay);
    }
    else if (digitalRead(3) == LOW){
      // buzz twice for fast charge
      pulseMotor(northPin,onIntensity,tunedDelay);
      pulseMotor(eastPin,onIntensity,tunedDelay);
      pulseMotor(westPin,onIntensity,tunedDelay);
      delay(50);
      pulseMotor(northPin,onIntensity,tunedDelay);
      pulseMotor(eastPin,onIntensity,tunedDelay);
      pulseMotor(westPin,onIntensity,tunedDelay);
    }
    else {
      // buzz three times for trickle charge
      pulseMotor(northPin,onIntensity,tunedDelay);
      pulseMotor(eastPin,onIntensity,tunedDelay);
      pulseMotor(westPin,onIntensity,tunedDelay);
      delay(50);
      pulseMotor(northPin,onIntensity,tunedDelay);
      pulseMotor(eastPin,onIntensity,tunedDelay);
      pulseMotor(westPin,onIntensity,tunedDelay);
      delay(50);
      pulseMotor(northPin,onIntensity,tunedDelay);
      pulseMotor(eastPin,onIntensity,tunedDelay);
      pulseMotor(westPin,onIntensity,tunedDelay);
      delay(50);
    }
  }
}
```

# Bill of Materials

| Quantity | Part Number | Part Name | Manufacturer |
|---|---|---|---|
| 1 | A000066 | Arduino Uno R3 ATmega328P Eval | Arduino |
| 3 | DRV2605L | Haptic Driver for ERM/LRA | Adafruit Industries LLC |
| 3 | 1201 | Vibrating Disc Motor | Adafruit Industries LLC |
| 1 | PRT-12795 | Jumper Wire Male to Male 6" 20 per Pkg | SparkFun Electronics |
| 2 | | cotton & polyester cloth | Fabricland |
| 1 | | velcro | Fabricland |
| 1 | 736503141 | Coats & Clark All Purpose Polyester Thread (123m) | Coats & Clark |
| 3 | R9332 | 22AWG 2pin connector cable 15cm0 | Electrical & Electronic Supply Inc |
| 1 | S2030 | 5x7cm tinned PCB double sided | Electrical & Electronic Supply Inc |
| 1 | S2029 | 3x7cm tinned PCB double sided | Electrical & Electronic Supply Inc |
| 1 | 1528-2182 | Solderless Breadboard Terminal Strip | Adafruit Industries LLC |
| 1 | M000006 | USB 2.0 Cable Type A/B | Arduino |
| 1 | BH26AASF | Battery Holder (Open) AA 6 Cell 9 Volt Snap Fasteners | Memory Protection Devices |
| 1 | MAX712CSE+TTR-ND | Controller MAX712CSE+T | Maxim Integrated |
| 1 | A129236TR-ND | 1 Ohms ±5% 0.5W, 1/2W Chip Resistor 0805 | TE Connectivity Passive Product |
| 1 | A126421TR-ND | 150 Ohms ±1% 0.333W, 1/3W Chip Resistor 0805 | TE Connectivity Passive Product |
| 1 | A103234TR-ND | 68 kOhms ±0.1% 0.1W, 1/10W Chip Resistor 0805 | TE Connectivity Passive Product |
| 1 | A103178TR-ND | 22 kOhms ±0.1% 0.1W, 1/10W Chip Resistor 0805 | TE Connectivity Passive Product |
| 1 | A103142TR-ND | 10 kOhms ±0.1% 0.1W, 1/10W Chip Resistor 0805 | TE Connectivity Passive Product |
| 1 | 490-8043-2-ND | 1µF ±10% 16V Ceramic Capacitor X7R 0805 | Murata Electronics |
| 2 | 587-1114-2-ND | 10000pF ±10% 50V Ceramic Capacitor 0805 | Taiyo Yuden |
| 1 | 478-0805YC106KAT2ATR-ND | 10µF ±10% 16V Ceramic Capacitor X7R 0805 | AVX Corporation |
| 1 | 478-7800-2-ND | Diode Schottky 20V 1A (DC) Surface Mount 0805 | AVX Corporation |
| 1 | 2N6107GOS-ND | Bipolar (BJT) Transistor PNP 70V 7A 10MHz 40W Through Hole TO-220AB | ON Semiconductor |
| 1 | CP-048H-ND | Power Barrel Connector Jack 3.00mm ID (0.118"), 6.50mm OD (0.256") Through Hole, Right Angle | CUI Devices |
| 1 | PA-3015OW-ZMX | Outtag 12W Universal AC Adapter 3V 4.5V 5V 6V 7.5V 9V 12V Multi Voltage Switching Power Supply Cord | Outtag |
| 5 | | PCBs 43mm x 28mm | PCBway |
| 1 | B3F-1020 | SWITCH TACTILE SPST-NO 0.05A 24V | Omron Electronics |
| 4 | 91698A713 | Pack of 25 Black-Oxide 18-8 Stainless Steel Phillips Flat Head Screws M2x0.4mm Thread, 20mm long | McMaster-Carr |
| 4 | 93738A263 | Pack of 10 Barbed Inserts for Plastic Brass M2x0.40mm Thread, 3.9mm Installed Length | McMaster-Carr |
| 1 | WRL-12580 | Bluetooth adapter | SparkFun Electronics |
| | | | |
| | | | |
| **Equipment** | | | |
| 1 | 052-0052-2 | Autoranging digital multimeter | Mastercraft |
| 1 | | 60/100W hot glue gun with 15pcs white glue sticks | SOONAN |
| 1 | JX1420 | Brother JX1420 Mechanical Sewing Machine | Brother |
| 1 | 058-6305-4 | Mastercraft 25W Soldering Iron | Mastercraft |
| 1 | S1030 | De-soldering pump | Electrical & Electronic Supply Inc |
| 1 | | MARTISAN Sheet Music Stand Holder/Portable Folding Music Stand Super Sturdy Adjustable Height Tripod Base Metal Music Stand, Lightweight & Compact for Storage Martisan | |
| 1 | B07NWC3L95 | Selfie Stick Tripod, UBeesize 51" Extendable Tripod Stand with Bluetooth Remote for Cell Phones and Cameras, Heavy Duty Aluminum, Lightweight | UBeesize |
| 1 | PW313 | AVerMedia Live Streamer CAM 313 1080p HD Webcam | Avermedia |
| 1 | 11200 | Westcott 6-Inch Plastic 180 Degree Protractor, Clear | Westcott |
| 1 | | | |
| 1 | | Etekcity Lasergrip 1080 Laser Thermometer Digital Infrared Thermometer Temperature Gun for Kitchen Cooking BBQ Grill and Bath Water, -58°F~1022°F (-50°C~550 Etekcity | |

*Figure 35: Bill of Materials Part 1*

| Cost per item | Description | Reference |
|---|---|---|
| $30.43 | Microcontroller | https://www.digikey.ca/en/products/detail/arduino/A000066/2784006 |
| $11 | Breakout board | https://www.digikey.ca/en/products/detail/adafruit-industries-llc/2305/5356831 |
| $2.70 | ERM | https://www.digikey.ca/en/products/detail/adafruit-industries-llc/1201/5353637?s=N4IgTCBcDaIIYBM4DMBOBXAlgFwAQDk |
| $2.70 | jumpers | https://www.digikey.ca/en/products/detail/sparkfun-electronics/PRT-12795/5993860 |
| $15 | cloth | |
| $5 | velcro | |
| $1.87 | spool of thread | https://www.walmart.ca/en/ip/coats-clark-all-purpose-polyester-thread-white/6000197237011?cmpid=sem_google_en_pla |
| $1.10 | clips (between breakout board and vibrators) | |
| $2.35 | timed PCB | |
| $1.55 | timed PCB | |
| $7.36 | breadboard | https://www.digikey.ca/en/products/detail/adafruit-industries-llc/64/7241427?utm_adgroup=General&utm_source=google |
| $3.95 | Arduino cable | https://store.arduino.cc/usa/usb-2-0-cable-type-a-b |
| $2.88 | battery pack | https://www.digikey.ca/en/products/detail/mpd-memory-protection-devices/BH26AASF/470766 |
| $9.21 | controller | MAX712CSE+T Maxim Integrated | Integrated Circuits (ICs) | DigiKey |
| $0.54 | resistor | CRGS0805J1R0 TE Connectivity AMP Connectors | Resistors | DigiKey |
| $0.16 | resistor | CRGH0805F150R TE Connectivity Passive Product | Resistors | DigiKey |
| $0.87 | resistor | CPF0805B68KE1 TE Connectivity Passive Product | Resistors | DigiKey |
| $0.87 | resistor | CPF0805B22KE1 TE Connectivity Passive Product | Resistors | DigiKey |
| $0.87 | resistor | CPF0805B10KE1 TE Connectivity Passive Product | Resistors | DigiKey |
| $0.37 | capacitor | GCM219R71C105KA37D Murata Electronics | Capacitors | DigiKey |
| $0.50 | capacitor | UMK212SD103KD-T Taiyo Yuden | Capacitors | DigiKey |
| $1.44 | capacitor | 0805YC106KAT2A AVX Corporation | Capacitors | DigiKey |
| $0.51 | diode | SD0805S020S1R0 AVX Corporation | Discrete Semiconductor Products | DigiKey |
| $1.26 | transistor | 2N6107G ON Semiconductor | Discrete Semiconductor Products | DigiKey |
| $1.20 | power jack | N4JgICBcoEwJxVAYygMwIYBsQQBTANCAPZQDaIAzAOxhgASVIAuoQA4AuUJAyuwE4BLAHYBzEAF9CYOAFY6JECkgYcBYnRBQAE |
| $18.99 | power plug | Smartphone/dp/B07VBKPWQL/ref=sr_1_4?dchild=1&keywords=AC+wall+adapter+10V+3mm&qid=1614476458&s=industri |
| $31.20 | PCB | https://www.pcbway.com/orderonline.aspx |
| $0.40 | push button | B3F-1020 Omron Electronics Inc-EMC Div | Switches | DigiKey |
| $0.39 | screws | https://www.mcmaster.com/91698A713/ |
| $0.71 | inserts | https://www.mcmaster.com/93738A263/ |
| $37.00 | Arduino module | https://www.sparkfun.com/products/12580 |
| | | |
| | | |
| | | |
| | | |
| | | |
| $69.99 | multimeter | https://www.canadiantire.ca/en/pdp/autoranging-digital-multimeter-0520052p.0520052.html?ds_rl=1283573&ds_rl=12835 |
| $21.99 | hot glue gun | https://www.amazon.ca/Sticks-Professional-Repairs-Festival-Decoration/dp/B07P46J49B/ref=sr_1_9?dchild=1&keywords=h |
| $86.97 | sewing machine | https://www.walmart.ca/en/ip/brother-jx1420-sewing-machine/6000198888634 |
| $24.99 | soldering iron | https://www.canadiantire.ca/en/pdp/mastercraft-25w-soldering-iron-0586305p.0586305.html?gclid=CJ0KCQiApsiBBhCKARI |
| $7.25 | desolder pump | |
| $21.00 | Music Stand | Carrying/dp/B0832TFMZRN/ref=sr_1_2_sspa?dchild=1&gclid=CJ0KCQiA- |
| $33.00 | Tripod - Webcam | Lightweight/dp/B07NWC3195/ref=sr_1_7?dchild=1&gclid=CJ0KCQiA-OeBBhDIARIsADyBcEF7Ug7JF9IAGCxuoQQSB- |
| 100 | Webcam | AVerMedia Live Streamer CAM 313 1080p HD Webcam (PW313) | Best Buy Canada |
| 0.88 | Protractor | https://www.amazon.ca/Westcott-6-Inch-Plastic-Degree-Protractor/dp/B000F8MFSW |
| 8 | Wall trim tool | |
| 48.44 | Thermometer | and Bath Water, -58°F~1022°F (-50°C~550°C), Yellow and Black: Amazon.ca: Home & Kitchen |

*Figure 36: Bill of Materials Part 2*

# Prototype Structure, Fabrication Plan, and Cost

## Prototype Design

The prototype aims to put together the two subsystems and create a fully integrated haptic experience for the user. It will allow the user to define what points of interest they are looking for and receive haptic feedback bringing them to the point of interest. To accomplish this goal, there are two major systems: the Arduino haptic controller and the Android application that will use the smartphone's camera to run the OpenCV ArUco library and provide positional data to the Arduino.

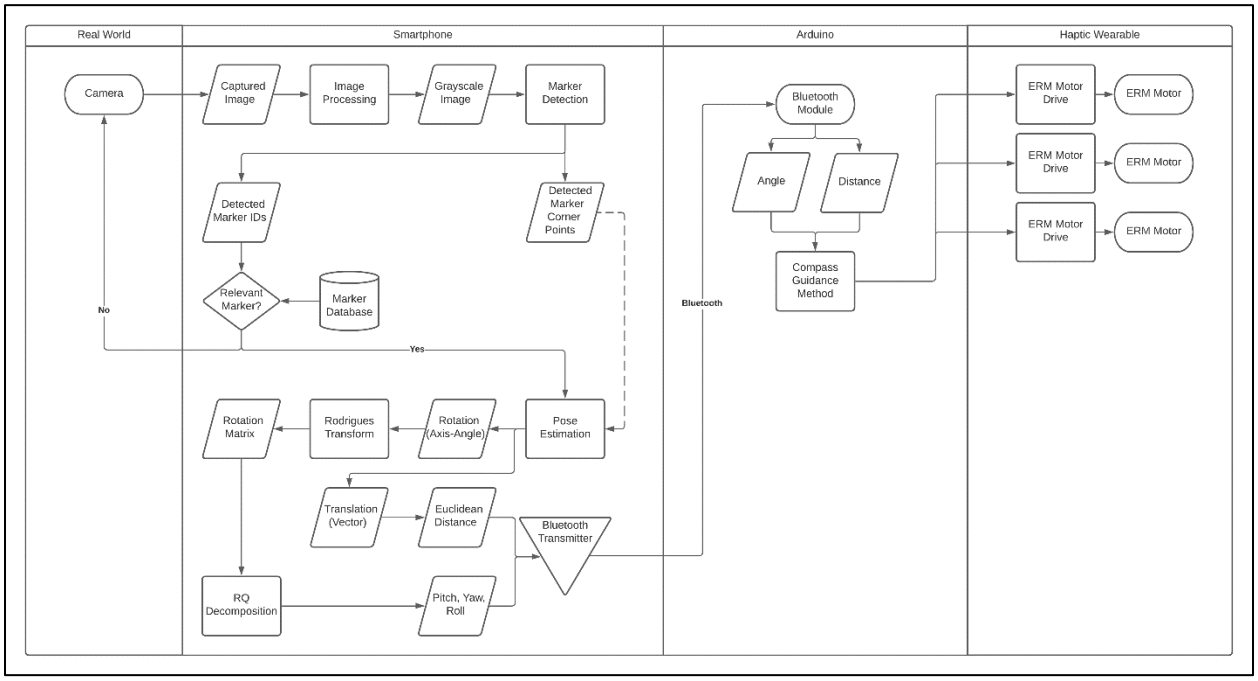



*Figure 37: System Overview Block Diagram*

# Prototype Plan

The prototype was broken into three different steps to test and validate the various components of our design and product. Testing phase is where the initial fabrication and testing of the haptic system and vision system is done as two separate systems. Performance data is collected in the testing phase to validate initial designs. In the integration phase, a laptop is used to connect the ArUco system to the Arduino haptic system through a USB port. This will allow testing of how the system works and how well it can provide feedback to the user. The communications between the two systems are tested, along with how effectively it can guide a user to a point of interest. In the final phase, the ArUco system is fully working on the smartphone and the communications are done wirelessly. The Bluetooth communications will be tested (heartbeat, vectors), as well as the user interface on the smartphone. The final phase of the prototype will not include the final enclosure, or the battery charging circuit. These are excluded because they are not required to validate the final product functionality.

## Testing Phase

### Vision System

To test and optimize the functionality of the vision system, a webcam paired with a computer was used. This allowed the vision system team to make changes and test the system quickly. Using a wired webcam also allowed the team to test the different outputs of the system (roll, pitch, yaw, and distance) without having to move a bulky computer and risk damaging it. The wired webcam is also relatively light and small, like a smartphone, and was therefore a suitable stand-in for testing. Using a computer instead of a smartphone allowed the team to utilize more time perfecting the vision system and eliminated the time it would take to move the program onto a smartphone every time a revision was made. The computer had more computing power, which reduced the amount of time spent calculating the calibration characteristics of the camera and finding the position vector of the maker in space. This method allowed the team to benchmark system utilization of the computer and investigate whether a smartphone could effectively run the software.

The computer used in conjunction with a webcam is an appropriate substitute for a smartphone at this stage. The team is not familiar with creating applications for Android or Apple devices, so this process eliminates any prototyping delays that would result. The program being run on the computer displays the rotation and distance values that would output directly from the finalized vision system and can therefore be used to test the haptic system.

### Haptic Feedback System

Initially, eccentric rotating mass (ERM) vibrational actuators and linear resonant actuators were tested on the user to determine which actuators to use in the system. ERM actuators were selected because of their stronger amplitude. The haptic feedback system is constructed using an Arduino and is outlined in the below figures. The next stage of testing involved testing out different haptic feedback patterns to determine what is an effective way to communicate direction to a user. This includes the frequency and intensity of the vibrations and is discussed in Design Analysis.

The prototype was constructed by cutting a piece of fabric, folding it in half, and sewing the two ends secure. Soft Velcro is glued to the centre of one of the halves to allow the pouches holding the actuators to be moved to suit varying arm sizes for different users. The actuators were soldered to their breakout boards via 2-pin connector clips. For preliminary testing, the breakout boards were placed on a breadboard, but for the final prototype and design they will be soldered to a tinned PCB protoboard. The cloth can then be wrapped and tied around the arm, as depicted in *Figure 38: Haptic Armband Prototype Flat* through *Figure 41: Haptic Armband Worn and Exposed*, and testing of the feedback patterns with differing frequencies and intensities began.

*Figure 38: Haptic Armband Prototype Flat*

*Figure 39: Haptic Armband Prototype Opened*

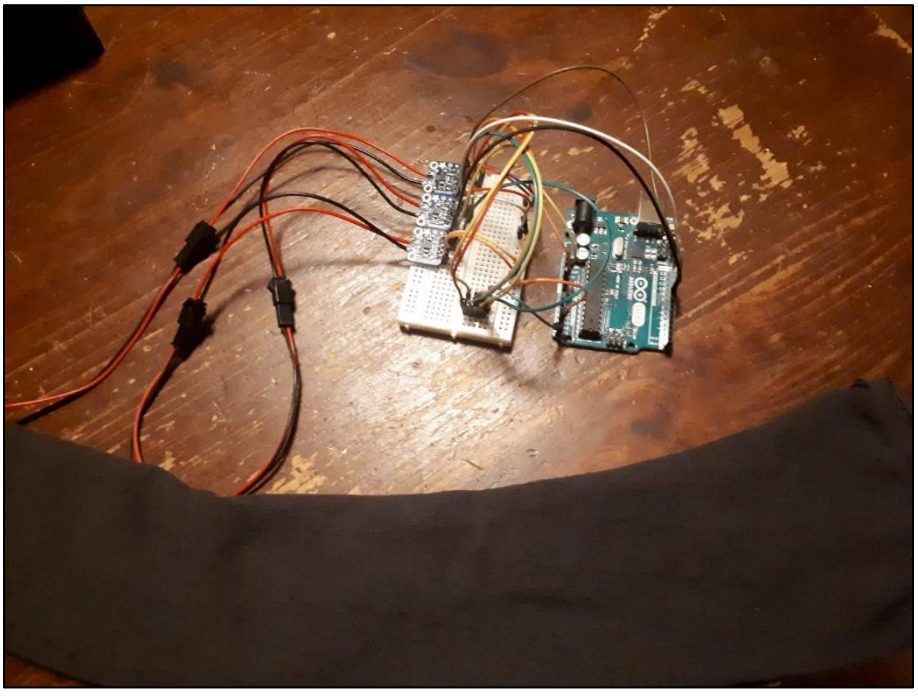


*Figure 40: Haptic Subsystem Prototype Circuit*

Table of Contents

*Figure 41: Haptic Armband Worn and Exposed*

Table of Contents

## Integration Phase

In this phase a serial link will be set up using a USB cable from the Arduino device and the computer running the ArUco software. The user will use a webcam held in their hand to simulate the feel of the smartphone. When the system is initiated, it will guide them to predetermined ArUco markers throughout a room. The primary function of this phase is to test and validate the user experience in using a handheld camera to locate a point of interest through haptic feedback. This step is crucial because it comes very close to the final product in terms of necessary functionality.

## Final Phase

In this phase the Android app will be the primary user interface between the ArUco system and the haptic feedback system. The smartphone's camera will be used to run the vision system, and the required user rotation and distance to reach the desired marker will be passed to the haptic system through a Bluetooth connection. The heartbeat functionality will also be tested, providing feedback for Arduino connection. The Android application is also tested in this phase, where the user will be required to use the app in full accessibility mode.

## Lockdown Mitigation

For the duration of this project, the team practiced social distancing and minimized contact between team members. To create the prototype, the team split the project into three different parts, each having specified team members. The haptic system required more than one person to construct, and the team delegated the task to members who were living together. The vision system was software based, and team members already had webcams available, so this part of the project was able to be completed remotely. Creating the smartphone application and testing the Bluetooth communications was also completed remotely. Most of the components for this project were ordered online and delivered without contact. The few materials that did require pick-up and on-site selection were retrieved while the team member practiced the appropriate health and safety protocols.

The armband for the haptic system prototype required tools that the haptics team did not have. To work around the lockdown, the individual with the proper tools outside the team gathered the appropriate materials and fabricated the armband. To avoid direct contact, the device was passed back and forth between the haptics team and the individual using respective mailboxes until the armband was complete and met prototype requirements.

During the lockdown, the prototyping labs were unavailable, and alternative methods for testing had to be investigated. Therefore, prototype testing was completed using objects that members had readily available. For example, when the vision system was tested to measure the accuracy of the distance and rotation calculations, a music stand, protractor, camera tripod, and measuring tape were used. These materials were readily available and addressed the requirements of the testing that was performed while allowing the individual to observe lockdown protocol.

# Prototype Cost

The prototype cost consists of parts required to complete fully functional testing. For this reason, the charging board for the batteries is not included since the final product will operate similarly with or without rechargeable batteries. Likewise, the final enclosure is not included in the prototype cost. Thus, the parts needed to create the fully functional prototype are those included in the prototype BOM. Materials used to construct the prototype that are not on the BOM were sourced through recycling or scavenging of materials. This was done to keep prototyping cost low. Group members were selected to do certain tasks based on equipment they already owned, because with the lockdown in place, the sharing of equipment was impractical. The materials included in this BOM include the components that were used for initial testing for the prototype. These were crucial in the testing and validation of our design and as such are included in the prototype cost.

*Table 6: Prototype BOM*

| **Part Description** | **Part Number** | **Qty** | **Individual Cost (CAD)** | **Total** |
|---|---|---|---|---|
| **Fabric** | | 1 | $20.00 | $20.00 |
| **Bluetooth Adapter** | WRL-12580 (SparkFun) | 1 | $37.00 | $37.00 |
| **ERM Device** | 1528-1177-ND | 12 | $2.70 | $32.40 |
| **Haptic Motor Driver** | 1528-1346-ND | 12 | $11.00 | $132.00 |
| **LRA Device** | 1670-1034-ND | 8 | $4.38 | $35.04 |
| **I2C Mux** | 1528-1363-ND | 1 | $9.61 | $9.61 |
| **2-pin connector cable** | R9332 | 3 | $1.10 | $3.30 |
| **3x7cm tinned PCB** | S2029 | 1 | $1.55 | $1.55 |
| **De-solder pump** | | 1 | $7.25 | $7.25 |
| | | | | ***$278.15*** |

# References


[1] BlindSquare, "BlindSquare," BlindSquare, October 2020. [Online]. Available: https://www.blindsquare.com. [Accessed 16 November 2020].

[2] I. H. A. I. A. W. Muhammad Shoaib, "Adaptive Auditory Feedback: A New Method for Desktop Assistance of the Visual Impaired People," in *2018 ACM International Joint Conference and 2018 International Symposium*, 2018.

[3] D. Maggiacomo, Interviewee, *Principal of W. Ross Macdonald School for the Blind.* [Interview]. 13 November 2020.

[4] S. K. I. S. Nurcan Yabaci, "The relationship between height and arm span, mid-upper arm and waist circumferences in children," *Annals of Human Biology*, vol. 37, no. 1, 2010.

[5] "OpenCV.org," OpenCV Team, 2021. [Online]. Available: http://opencv.org/. [Accessed 1 March 2021].

[6] "CMake.org," Kitware, 2021. [Online]. Available: http://cmake.org/. [Accessed 1 March 2021].

[7] "Microsoft Visual Studio 2019," Microsoft, 2021. [Online]. Available: http://visualstudio.microsoft.com/. [Accessed 1 March 2020].

[8] "ArUco Marker Detection," OpenCV Team, 2020. [Online]. Available: https://docs.opencv.org/4.5.1/d9/d6a/group__aruco.html. [Accessed 1 March 2021].

[9] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280-2292, 2014.

[10] "Detection of ArUco Markers," OpenCV Team, 2020. [Online]. Available: http://docs.opencv.org/4.5.1/d5/dae/tutorial_aruco_detection.html. [Accessed 1 March 2021].

[11] M. W. Spong, S. Hutchinson and M. Vidyasagar, "Appendix E: Vision-Based Control," in *Robot Modeling and Control*, West Sussex, John Wiley & Sons, Ltd, 2020, pp. 555-560.

[12] "Intel® Core™ i5-4590 Processor," Intel Corporation, [Online]. Available: http://ark.intel.com/content/www/us/en/ark/products/80815/intel-core-i5-4590-processor-6m-cache-up-to-3-70-ghz.html. [Accessed 1 March 2021].

[13] "Galaxy A5 (2017)," Samsung, [Online]. Available: https://www.samsung.com/ph/smartphones/galaxy-a/galaxy-a5-black-32gb-sm-a520fzkdxtc/. [Accessed 1 March 2021].

[14] Arduino, "Power Consumption Arduino," Arduino, 4 August 2010. [Online]. [Accessed 20 February 2021].

References


[15] Texas Instruments, "DRV2605L 2 to 5.2V Haptic Driver for LRA and ERM With Effect Library and Smart-Loop Architecture," September 2014. [Online]. Available: https://cdn.sparkfun.com/datasheets/Robotics/drv2605l.pdf. [Accessed 20 February 2021].

[16] Adafruit, "Product Specification Datasheet," 10 October 2011. [Online]. Available: https://cdn-shop.adafruit.com/product-files/1201/P1012_datasheet.pdf. [Accessed 20 February 2021].

[17] Energizer Holdings Inc., "LR6CL_EU," [Online]. Available: https://data.energizer.com/PDFs/LR6CL_EU.pdf. [Accessed 20 February 2021].

[18] Adafruit, "Lithium Ion Polymer Battery 3.7V 2500mAh," Adafruit, [Online]. Available: https://www.adafruit.com/product/328. [Accessed 20 February 2021].

[19] Duracell, "MN1604," [Online]. Available: https://d2ei442zrkqy2u.cloudfront.net/wp-content/uploads/2016/03/MN1604_6LP3146_US_CT1.pdf. [Accessed 20 February 2021].

[20] Duracell, "MN1300," [Online]. Available: https://web.archive.org/web/20120521213503/http://www.duracell.com/media/en-US/pdf/gtcl/Product_Data_Sheet/NA_DATASHEETS/MN1300_US_CT.pdf. [Accessed 20 February 2021].

[21] Y. Z. Z. D. B. J. T. Wenhua H. Zhu, "Energy efficiency and capacity retention of Ni-MH batteries for storage applications," *Applied Energy*, vol. 106, pp. 307-313, 2013.

[22] Renfrew County District School Board, "School Day Schedule/Periods," [Online]. Available: https://fhs.rcdsb.on.ca/en/ourschool/school-day-schedule-periods.asp. [Accessed 20 February 2021].

[23] Contrado, "Types of Fabrics," Contrado, 2021. [Online]. Available: https://www.contrado.com/types-of-fabrics. [Accessed 27 February 2021].

[24] D. Shooter, "Use of two-point discrimination as a nerve repair assessment tool: preliminary report," *ANZ Journal of Surgery*, vol. 75, no. 10, pp. 866-868, 2005.

[25] Science World, "Tactile Sensitivity," Science World, 2021. [Online]. Available: https://www.scienceworld.ca/resource/tactile-sensitivity/. [Accessed 24 February 2021].

[26] H. Bajwa and Y. A. Khalili., "Physiology, Vibratory Sense," 1 3 2021. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK542288/.

[27] A. Studios, "developer.Android," Google, [Online]. Available: https://developer.android.com/guide/topics/ui/accessibility/index.html. [Accessed 24 February 2021].


Table of Contents