CribSafe Baby Monitor

Project Three: *Track* and Field — Written Proposal

Team 7 | Mathieu Chenier, Dabeer Abdul-Azeez, Trevor Tung, Junhyeong Lee

Date | February 24th, 2020

Academic Integrity Statement

The student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is my own and adheres to the Academic Integrity Policy of McMaster University.

Submitted by [Dabeer Abdul-Azeez, 400261347]

Dabeer

The student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is my own and adheres to the Academic Integrity Policy of McMaster University.

Submitted by [Mathieu Chenier, 400239909]

Mat Chenier

The student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is my own and adheres to the Academic Integrity Policy of McMaster University.

Submitted by [Trevor Tung, 400261916]

The student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is my own and adheres to the Academic Integrity Policy of McMaster University.

Submitted by [Jun Hyeong Lee, 400202808]

Executive Summary

In the United States, approximately 10, 000 children are treated in the ER every year for injuries associated with their crib [1,2]. Most of these injuries are caused by toddlers who fall out of or into their crib, attempting to escape. Further, in 2017, 1360 infants died from Sudden Infant Death Syndrome (SIDS) in the United States due to unsafe crib environments or sleeping positions [3]. Crib-related accidents and injuries as such should be prevented to promote child safety and well-being.

Given the available technology of the 21st century, any crib-related deaths should be unacceptable. The CribSafe Baby Orientation Monitor (CBOM) is a wearable solution that provides crib safety for infants and toddlers, providing families with peace of mind while they and their children are asleep.

The CBOM is a wearable orientation monitor intended to be worn in the front pocket of an infant's pyjamas, provided to them with purchase, while they are asleep in a crib. In a pocket, the CBOM will be inaccessible to infants, allowing the device to be tamper-proof. A child may remain in their crib until the age of three and, therefore, the CBOM is intended to be used for children up to this age [4].

To prevent crib accidents, the CBOM constantly measures a child's rotation and acceleration. By measuring a child's 'side-to-side' rotation, the CBOM can detect if an infant rolls onto their stomach, indicating an unsafe sleeping position. In the prone position, infants up to one year were found to be five times more likely to experience SIDS [5]. The monitoring of a child's 'back-to-front' rotation allows the CBOM to detect if a child stands up in their crib, which may predict an escape attempt. This allows the CBOM to prevent crib injury before it occurs. Lastly, the CBOM measures a child's acceleration to detect if the user falls. This function acts to escalate the response that the CBOM provides to parents and operates as a fail-safe, in case the previous two functions malfunction.

The CBOM notifies parents if their child is in an unsafe condition through an app, connected to the device via Bluetooth. The app would use audio warnings to notify parents of their child's condition. If the family does not have access to a cellular device, the CBOM also houses an onboard speaker that can alert parents.

The CBOM is intended to be a replacement or addition to traditional audio or video baby monitors. Traditional baby monitors require parents to be vigilant and conscious while watching their child, however, these devices provide no use without an active observer. Instead, the CBOM can passively detect any unsafe condition that a child may put themselves in and alert parents whether they are sleeping or awake. The full-night functionality and peace of mind the CBOM provides gives consumers an incentive to make its purchase.

Table of Contents

Design Objectives

The CribSafe Baby Orientation Monitor strives to fulfill the following criteria:

- Lightweight
- Easy to use
- Ergonomic
- Durable
- Sterile
- Adjustable for different children
- Wearable
- Biocompatible (bioinert)
- Babyproof
- Collect data
- Responds to data
- Comfortable
- Waterproof
- Non-choking hazard

The CribSafe Baby Orientation Monitor seeks to fulfill the following **need statement**:

Design a wearable device for babies and toddlers that notifies their parents of their safety while they sleep in a crib to prevent crib-related injuries and deaths.

Background and Research Summary

The target audience for this device is caregivers of young babies ranging from newborn to three years old. The CBOM was designed largely as an alternative to traditional sound and/or video baby monitors. These devices require the parents to be actively watching or listening to the monitor in order to observe their baby, while the CBOM allows for parents to passively monitor their baby as the device will notify them of the status of their child. The CBOM may also be used as a supplement to a traditional audio baby monitor as a combination of the two can keep a parent better informed about their baby without costing as much as some smart baby monitoring systems (*see Market Analysis*). This device could be used to monitor babies in a hospital setting as well. It could aid nurses working in nurseries to monitor multiple newborns simultaneously, alerting them when a baby is awake. With many possible use cases, extensive research was conducted on crib safety, materials, and miniaturization in order to design a suitable product.

The main focus of the CBOM is to keep infants safe while in their cribs. One of most prominent concerns regarding a young child's health while they sleep is the occurrence of Sudden Infant Death Syndrome (SIDS). This is the leading cause of death between babies between the ages of 1 month to a year, and research has yet to show the exact mechanism by which this occurs [6]. Research has shown, however, that the risk of SIDS is significantly decreased if a baby is placed on their back to sleep compared to when they sleep on their stomachs or sides [6]. On a similar note, the importance of crib safety is growing in demand as over 10,000 kids are taken to the emergency room each year due to crib injuries in the United States [2]. Thus a device to monitor whether a child is sleeping and how they are sleeping can be of great use to parents of children from ages of newborn to 3 years old. Currently, the growth in the global market for baby monitoring technology is projected to increase from $929m in 2016 to $1.63bn by 2025 [7]. With such an intense rise in the demand for baby monitoring technology, the CBOM seeks to target the market with an approach differing from the typical video and/or audio baby monitors: namely, to offer parents the option to passively monitor their child without requiring constant supervision of a video or audio monitor.

The material selection is critical to the function of the CBOM as flexing or other distortions to the casing may damage internal electronics, therefore impairing the longevity and accuracy of the device. It is vital that the product withstand regular usage and sanitations without having the device compromise its structural integrity. The body of the device will be composed of Polyvinylidene Fluoride (PVBF) due to its high mechanical strength, abrasion resistance, durability [8]. This material was selected to give the device a low profile while maintaining its toughness. Any item worn by a young child is also very susceptible to saliva and other bodily fluids, so a material with exceptional chemical resistance and impermeability was chosen.

All electrical components featured in the physical computing prototype, including the buzzer, orientation sensor, Raspberry Pi, and LEDs would be miniaturized in the final design. The miniaturization can cut back on excess materials, reducing the weight and size of the device to increase the suitability of placing it on a child. It is notable, however, that as electronic components get smaller, the cost to manufacture increases, so the device would only be made smaller to the point where it could maintain a cheap market price, as one of CribSafe’s goals is to offer a low-cost alternative to other smart monitors. Due to the low-power nature of the CBOM, issues that tend to arise in miniaturization of electronics, such as thermal distribution, raise little concern for its usage [9].

An analysis of the current market indicates a clear need for parents to monitor their children and prevent them from injuring themselves while they sleep. With appropriate resources dedicated to manufacturing and testing, the CBOM can ease the lives of countless parents whose children are at risk of SIDS or other crib injuries each night they go to sleep.

Market Analysis

Introduction

With increasingly miniaturized technology, the prospect of wearable devices for parents to place on their young children is becoming more feasible, whereas before there was the risk of such technology being too bulky, uncomfortable, or unsafe to place near unwary young ones. In return for accuracy and an increased number of functions available to monitor your child, however, current smart baby monitoring devices are often very expensive, so a parent must decide what exactly they value in a smart baby monitor.

Owlet Smart Sock

The Owlet Smart Sock is a wearable sock that tracks a baby’s oxygen levels, heart rate, and sleep, transmitting this information to parents through an app [10]. Currently, it is $540 CAD for the entire monitoring system (see **Figure 1**), which includes a camera, the sock itself, and a base station which can also provide visual notifications to the parent. The system can be purchased without the camera for $399. The sock has an embedded pulse oximeter, and the camera has a motion and temperature sensor, which together can record the child’s oxygen levels, pulse, movement, sleep and temperature. Similar to the CBOM, this is a wearable device which tracks a child’s status while in their crib. It is lightweight, easy to use, and transmits data to an app for the parents.

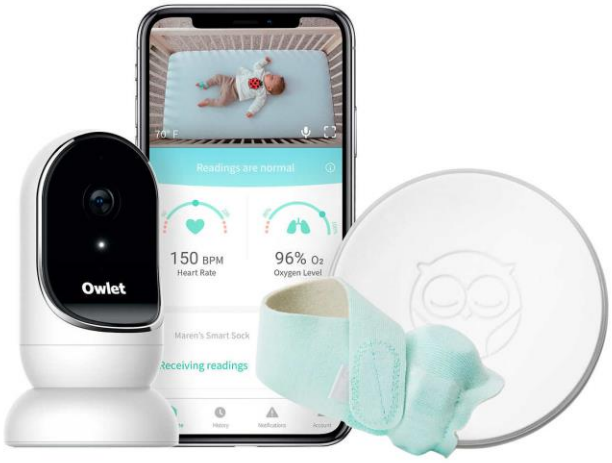


Figure 1: Owlet Smart Sock 2 + Camera Bundle

The main contrast of the Owlet design when compared to the CBOM is its focus on monitoring the baby's vitals rather than their position in the crib, acting less as a passive monitor and preventative measure for injuries and more as a high-tech real-time monitor. As seen in the Owlet's price, the sensors and technology which measure pulse, oxygen levels, motion, and temperature are more expensive than the single orientation and acceleration sensor in the CBOM. Due to the increased demand in the last decade for sensors such as accelerometers and gyroscopes, price erosion has driven down the cost of these sensors making them cheaper alternatives [11]. Furthermore, the Owlet's sensors cannot tell if the child has rolled over, if they are trying to climb out of their crib, or if the child has fallen. While the Owlet system can provide notifications for abnormal vital readings, when vitals are normal it relies on the parent actively checking their phone to have a sense of how their child is doing. The CBOM does not provide data on vitals, but it acts more as a preventative measure for children who might hurt themselves by climbing out of their crib or rolling onto their front. It can inform parents of their child's position in the crib using much fewer resources, though it is limited in providing parents with specific data like the Owlet sock. On another note, the Owlet Sock is more adaptable than the CBOM as it can be worn by any child with no special clothing required.

While the Owlet system is slightly different in its scope versus the CBOM, it was chosen for this market analysis as a representation of how expensive smart baby monitors can become, especially if parents want to monitor more data about their child such as heart rate and breathing rate.

MonBaby Smart Movement Monitor

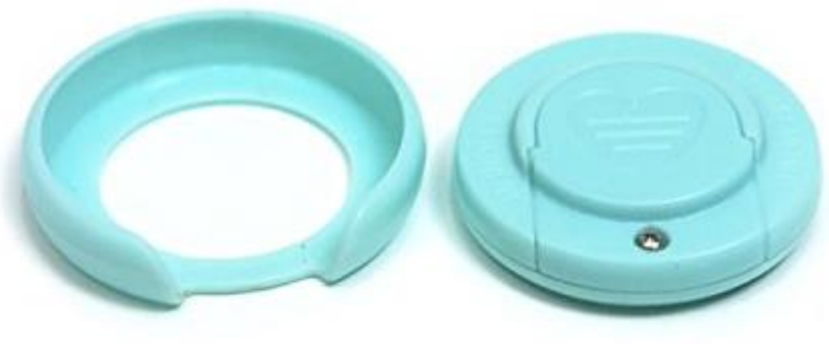


Figure 2: MonBaby Smart Button Monitor

Clasps onto a baby's clothing with one piece on the outside of the clothing and one piece on the inside

The MonBaby Monitor (see **Figure 2**) is a two-piece button design which is worn on the front of a child's clothing and which records useful data about sleeping position, breathing, and proximity to a parent's smartphone. It can measure whether the child is on their front during sleep, whether their breathing is normal, and whether they are too far away from their parents' phone. It is a relatively cost-effective option compared to other baby monitoring systems available on the market: the cheapest model is $50 [12]. There is no mention of whether the device can check if the child is standing up in their crib or whether they have fallen down, but the company website does advertise the ability to track sleeping position and breathing movements, so it is likely that an accelerometer and/or gyroscope exist within the device.

The MonBaby monitor is similar to the CBOM with the added bonuses of being a one-size-fits-all design and having the ability to measure breathing to check on the baby's vitals. Though it has different features and does not measure exactly what the CBOM does, with a software update and modified electronics, the device might be able to encompass the CBOM's functionalities. The MonBaby Monitor targets similar design objectives to the CBOM, being a cost-effective alternative to expensive baby monitors which require active monitoring.

A Comment on the Choice to Smartly Monitor One's Child

Parents have the choice to monitor and look after their child how they please, within reason. Many smart baby monitors make use of this fact in order to increase prices drastically while advertising various sensors and data that it can track. This brings up a subjective question about what aspects of a young child's sleep—or life, for that matter—a parent truly values and to what extent they value the extra benefits which smart monitors offer over traditional baby monitors. We at CribSafe wanted to introduce a product that could better compete with traditionally cheap two-way speaker systems instead of trying to target an expensive and niche smart monitor market. We wanted to offer to parents of young children an affordable solution with only the essentials because while different sensors can measure more aspects of a young child's vitals and provide more information to the parent about their child's health, no baby monitors available on the market currently are classified as medical devices as that is not their intended purpose [13].

We wanted to provide information to the parents only if that information would be difficult to acquire without excessive supervision of their child. For example, the position sensor on the CBOM can inform parents if their child is awake in their crib without them needing to watch their child constantly—through a video monitor, app, or otherwise. By contrast, we find that temperature sensors on smart baby monitors provide little extra benefit to the parent, as they would eventually notice whether their child was cold or had an abnormal temperature when they went to go check on them. Granted, there are some arguably important data which certain on-the-market solutions measure that CribSafe does not (see *Design Critique, Discussion, and Recommendations* for more): MonBaby's monitor, for example, can check for irregular or stopped breathing—something which a parent would not easily be able to check at night unless they watched their child very closely. In that regard, we at CribSafe agree that smart devices can provide useful information to parents, but we do not advocate the implementation of expensive sensors in such devices that do not provide a worthwhile advantage over a selection of cheaper sensors that can accomplish the same task of passively keeping parents aware of a child's health.

Summary

Market analysis of possible competitors to the CBOM reveals that there exists a wide variety of smart baby monitors which differ in what they measure and how much they cost. In order for CribSafe to remain competitive in this market, it should keep costs and prices low while also providing enough useful information to the parent to justify purchasing it over a traditional two-way baby monitor system.

Description of Design (Design Specifications)

Description of Tangible Prototype

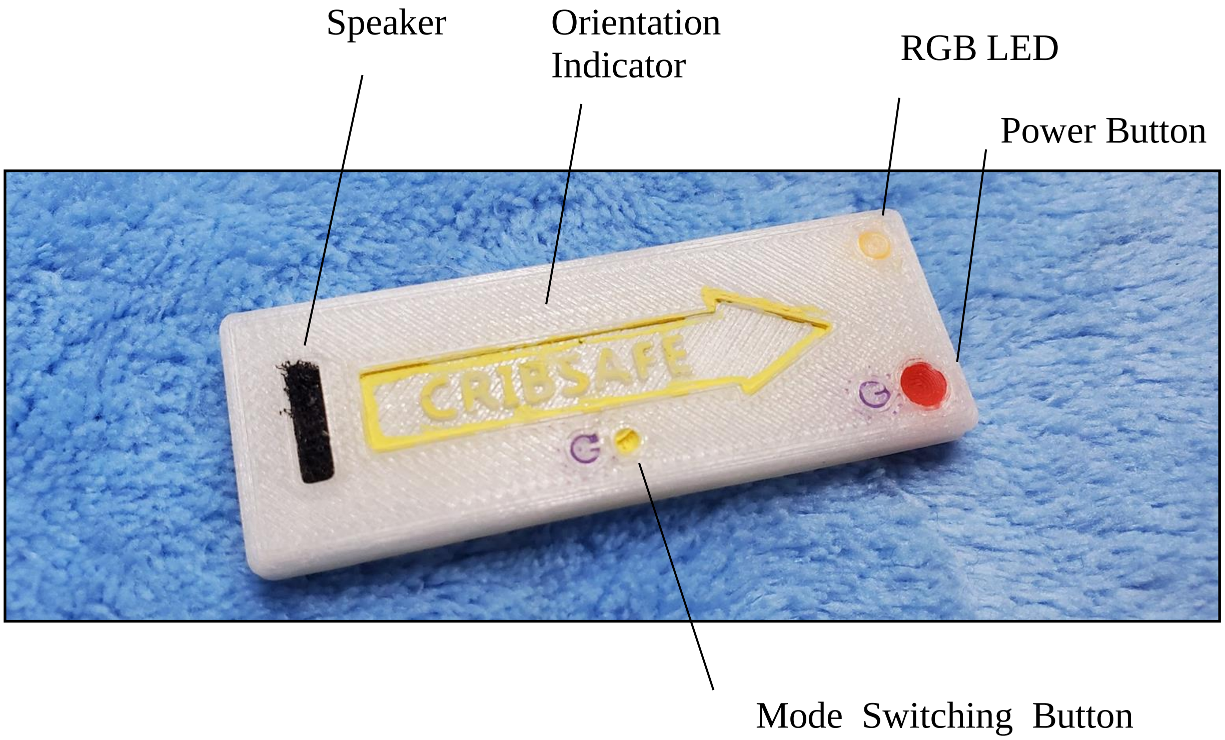


Figure 3: Tangible prototype for the CBOM

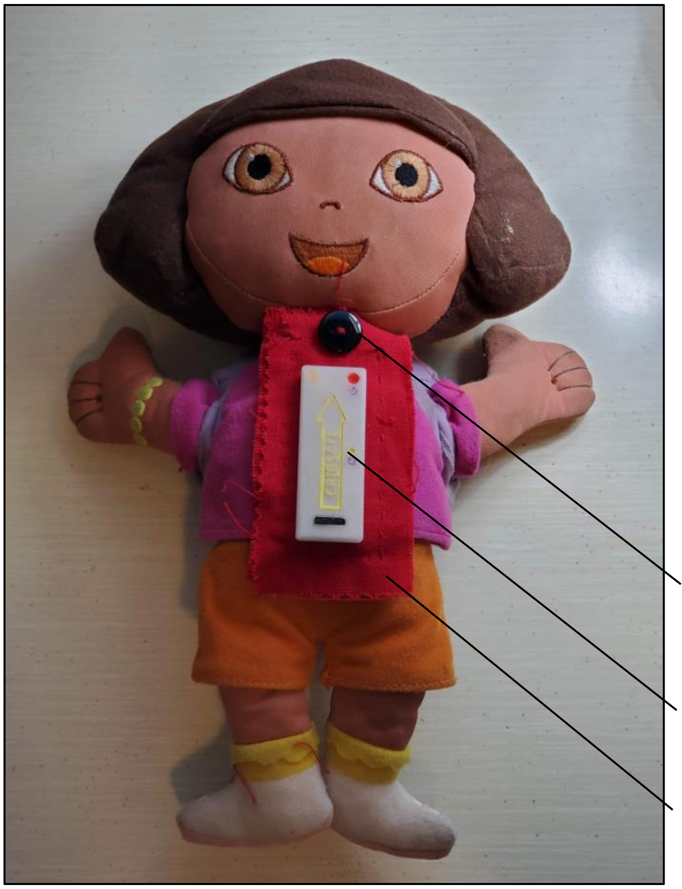


Figure 4: Prototype of pocket to hold the CBOM, glued onto a doll.

NOTE: Pocket was not made to scale with doll, it was made to scale with the CBOM prototype. In reality the pocket would be much smaller on a baby's clothing.

Through an iterative design process, the CBOM (see **Figures 3 and 4** for tangible prototype) has been designed as a wearable device to be placed in a specifically designed pocket attached onto a baby's pajamas while they sleep in their crib at night. An orientation sensor is used to record the baby's orientation and acceleration. The orientation is measured with respect to predefined spatial axes using euler angles, and thus the device must be worn in the position instructed for the position to be calculated properly. To turn on the device, the power button must be pushed and held for three seconds, turning on a light to indicate its status. The same action is used to power off the device. Once on, the device begins to record and monitor data to determine if the baby is standing up, lying on their stomach, or falling. If any of the three scenarios are recognized, a buzzer will be triggered with varying intensities, corresponding to the different events. The device also contains a second button to enable/disable the feature that detects the "side-to-side" orientation of the baby, as children more than a year old can safely sleep in the prone position. If an alarm is triggered it will continue and can only be disabled by turning the device off then back on again.

Part	**Description**
Speaker	The speaker on the device sounds when the system detects potential hazards (i.e. baby lying on front, standing up, or falling down)
Mode Switching Button	The device has two modes: a mode for users under 1-year-old and another mode for users older than 1-year-old. For the under 1-year mode, the rotation alarm is on, allowing the device to sound when the baby is on their front. For the other mode, the rotation alarm is off, so only standing detection and fall detection wil be enabled. The button allows the user to switch between the two modes.
Power Button	The button was designed to be held for three seconds in order to toggle the device on or off. This minimizes accidental interaction from the child.
Orientation Indicator	The arrow is supposed to point towards the baby's face, thus aligning the CBOM with the child's spine (the calculations are dependent on this assumption).
Main Frame	The main frame was designed to be too large to present a serious choking hazard if it were to be removed from its pocket. The rounded edges prevent discomfort while the device is on the user. To be manufactured out of PVDF (see *Research Summary* for more information).
Pocket & Clothes Button	The pocket with a clothes button protects the device from being tampered with or bitten by the child.

Regarding the Final Commercialized Design

In the final commercialized product, the computational system of the device would support Bluetooth connectivity to an app, providing advanced monitoring functionality and enhanced convenience in terms of interaction with device. A small coin cell battery would be included inside the device alongside miniaturized circuit components, making the device much more compact than the physical computing prototype we designed. The dimensions of the device will still be relatively consistent with the tangible prototype displayed in this document—specifically 6.0cm x 2.2cm x 0.5 cm—though the thickness of the device might increase slightly depending on how thin the internal electronics can be made for a low price.

Description of 3-D Printed Assembly for Physical Computing Prototype

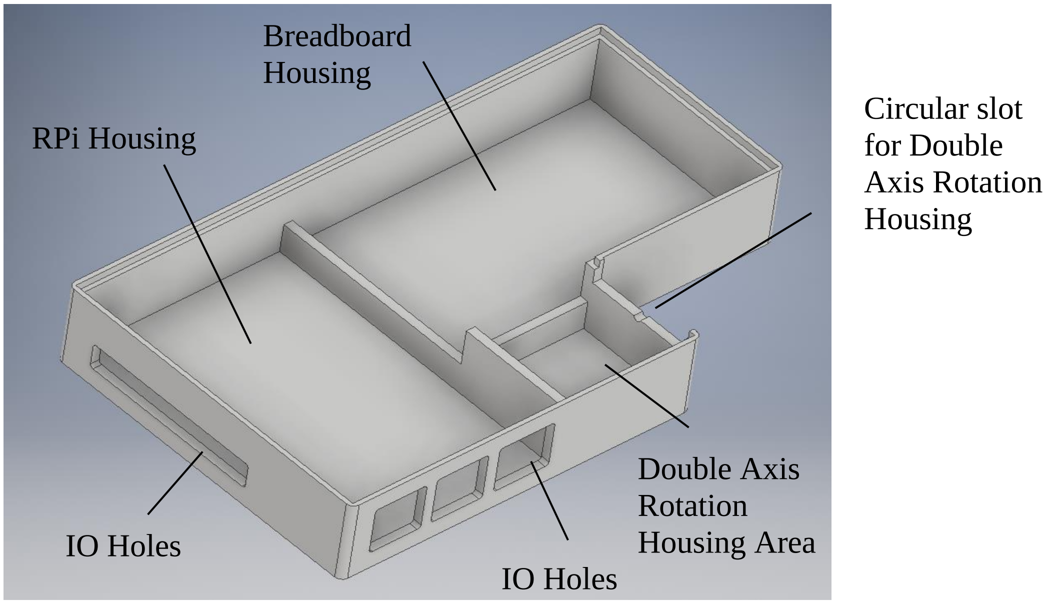

Breadboard Housing
RPi Housing
Circular slot for Double Axis Rotation Housing
Double Axis Rotation Housing Area
IO Holes
IO Holes

Figure 5: Physical Computing Prototype Main Housing CAD Model

The 3-D printed assembly consists of three components: the main housing (see **Figure 5**), the lid (see **Figure 6**), and the Double Axis Rotation Housing (see **Figure 7**).

The main housing holds the Raspberry Pi, sensor, and breadboard circuitry. It is composed of three main sections: the Raspberry Pi housing, the Double Axis Rotation Housing area, and the mini-breadboard housing. All the sections were dimensioned slightly larger than the actual size of the components they were intended to hold in order to allow for smooth assembly and disassembly of the physical computing prototype. Initially, the design made space for a full sized breadboard (see **Figure 16** in Appendix), but a mini breadboard was chosen instead to save on space because the circuitry involved (see **Figure 11**) was not too complex. The wall separating the Raspberry Pi and the breadboard circuitry was lowered to allow extra space for the cables connecting the components. IO holes on the side of the Raspberry Pi housing section were made to allow ethernet and peripheral cable connections. The Double Axis Rotation Housing Area has a slot where the lever of the Double Axis Rotation Housing will fit in.

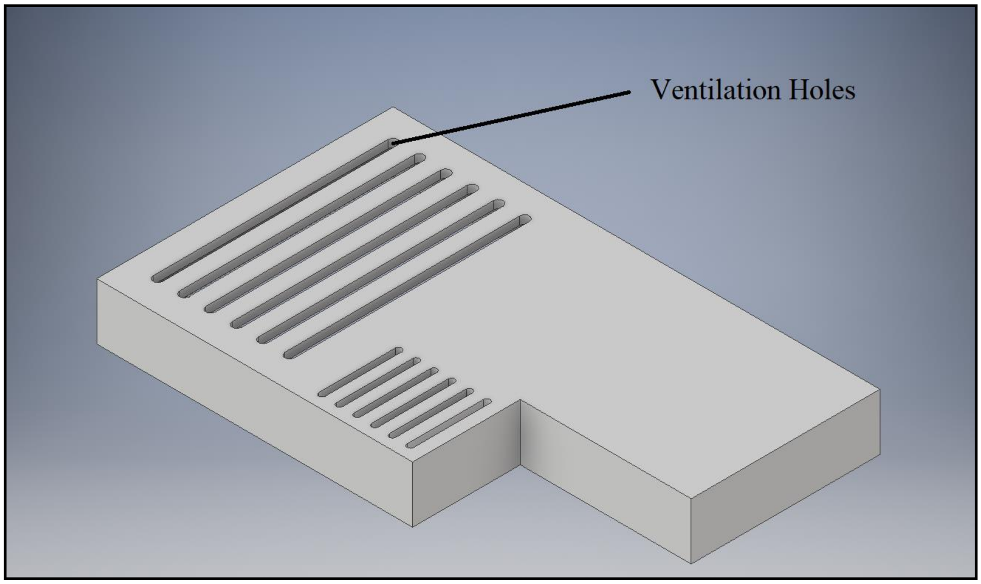


Figure 6: Physical Computing Prototype Lid CAD Model

The housing lid (see **Figure 6** and **Figure 9**) was designed to allow secure storage and transport of the computing prototype. Ventilation holes were manufactured above where the Raspberry Pi and sensor sit in the main housing to improve airflow into the device and reduce material use.

The Double Axis Rotation Housing (DARH, see **Figure 7** right) has a sensor housing that fits the sensor exactly in one of two orthogonal positions, which in conjunction with the Rotation Lever allows for

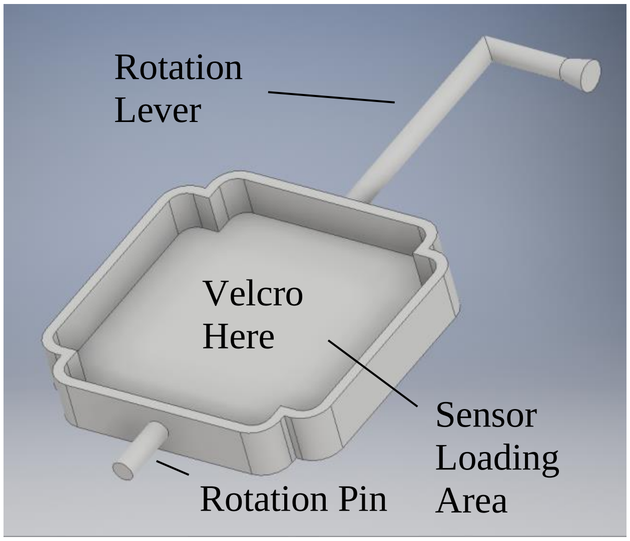


Figure 7: Physical Computing Prototype Double Axis Rotation Housing

the testing of two axes of rotation (see *Verification of Physical Computing Prototype*). The Sensor Loading Area has velcro attached on the real model which is meant to attach to a piece of velcro stuck onto the orientation sensor (see **Figure 8**). The rotation pin and long axis of the rotation lever are placed in the appropriate slot on the main housing to allow for rotation (see **Figure 5** for CAD model, **Figure 10** for 3D printed version).

Fabricated Assembly

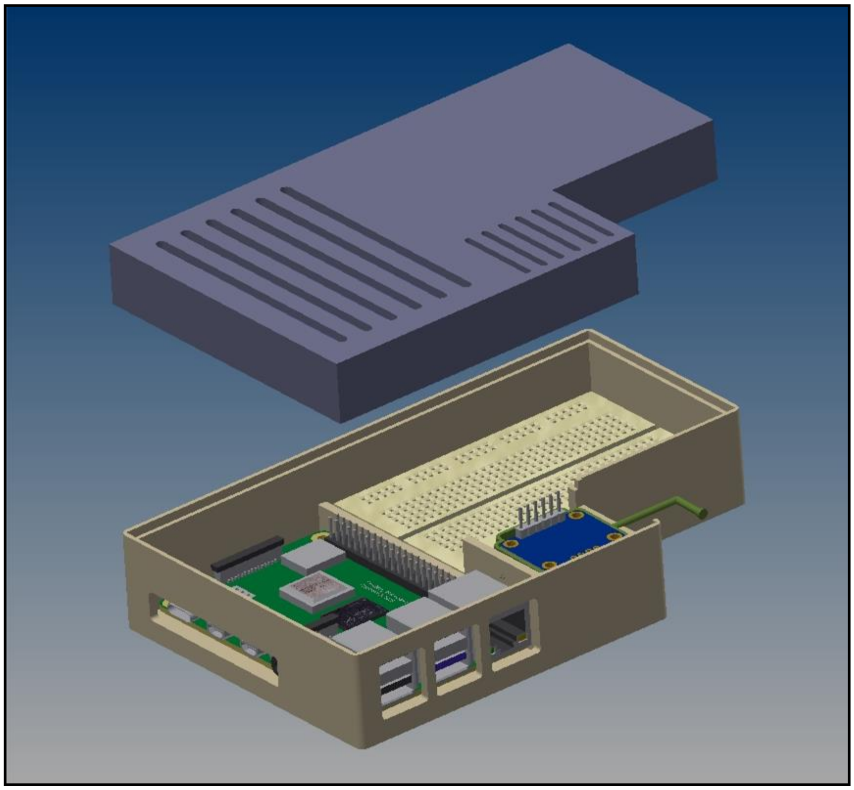


Figure 8: Computational Prototype Housing CAD Assembly

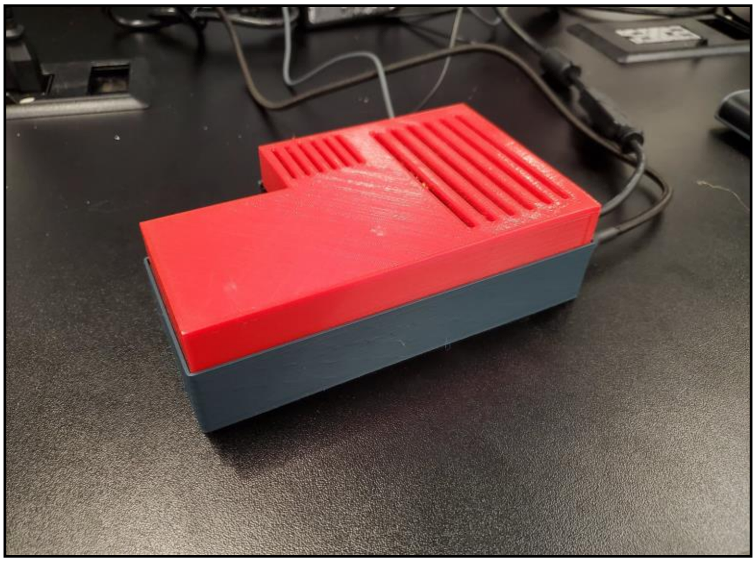
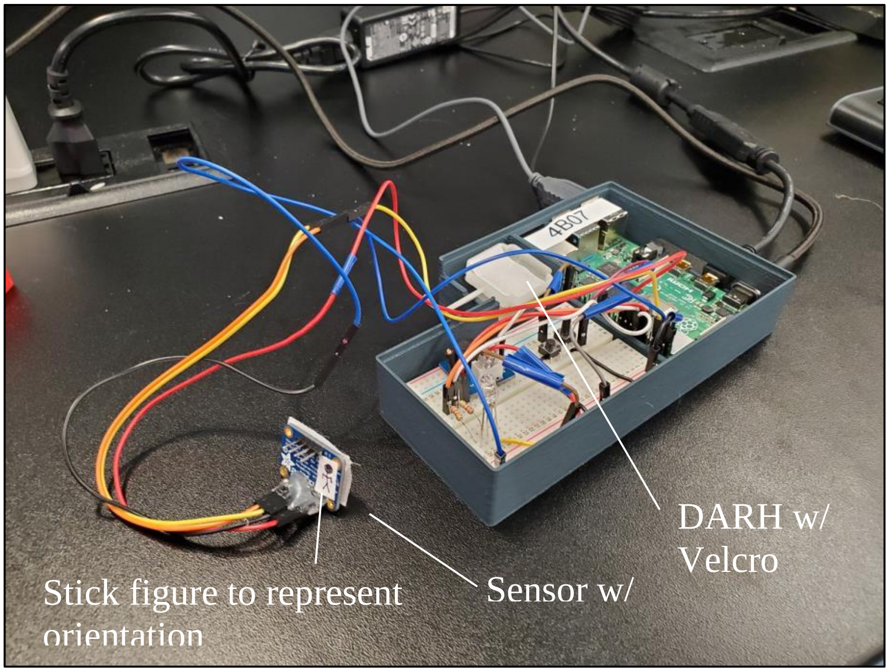


Figure 9: Computational Prototype Housing 3D Printed Assembly



Note how the DARH is placed in the main housing.

Figure 10: Physical Computing Prototype (Lid Open)

Description of Physical Computing Prototype

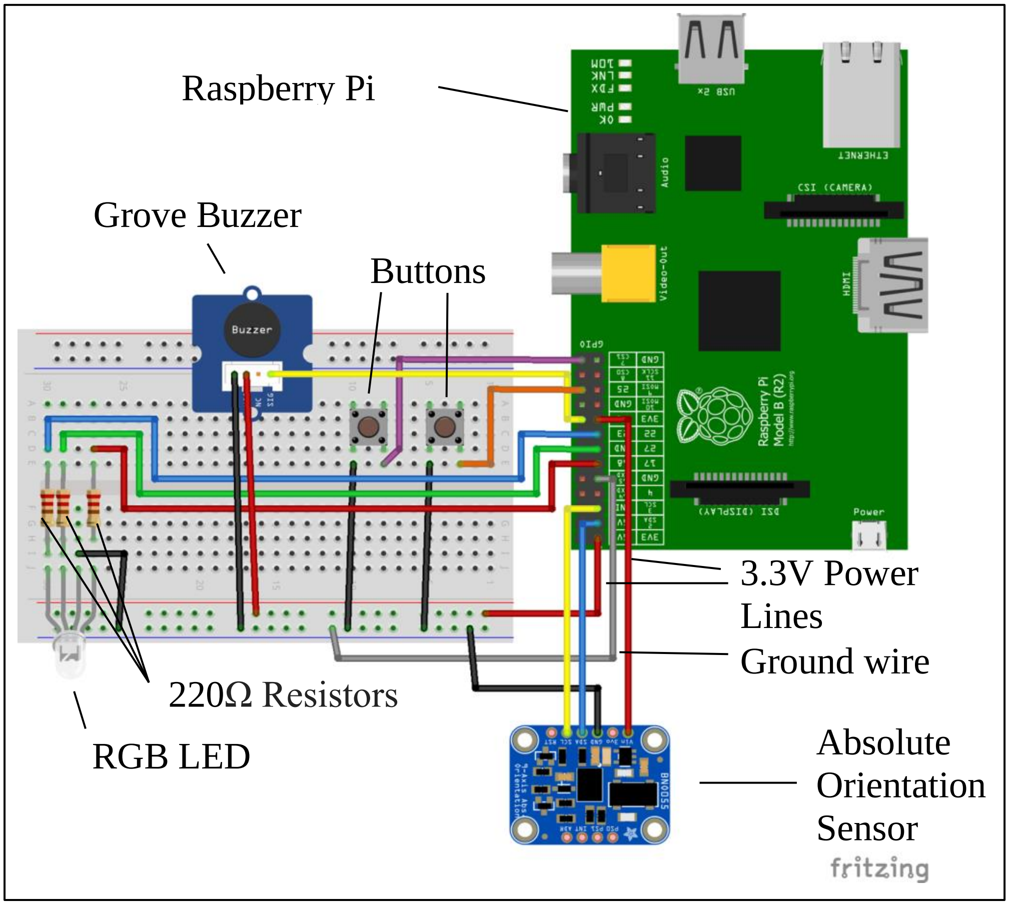


Figure 11: Circuit diagram of physical computing prototype

Circuit diagram designed using *fritzing*. Older RPi model was used for the diagram due to *fritzing*'s limitations.

The physical computing prototype (see **Figure 11** for circuit diagram) was built to demonstrate the feasibility of the CBOM concept. It operates using a Raspberry Pi 4B, an Adafruit BNO055 Absolute Orientation Sensor, an RGB LED, a Grove Buzzer v1.2, and two buttons. The two buttons operate exactly as detailed under *Description of Tangible Prototype*.

Once powered on, the sensor constantly records data on its orientation and acceleration. When the sensor detects that it is in a dangerous orientation for too long or it feels rapid acceleration, it will trigger the buzzer with a specific buzzing pattern, related to the event occurring. The software also provides print statements and a GUI which both update constantly to provide additional information to the user about what state the sensor is in.

Note that the Absolute Orientation Sensor and the buzzer have their own sources of 3.3V Power. This is because testing revealed that the buzzer sounds became somewhat muted when it shared power with other loads.

Physical Computing Prototype Input/Output Chart

Item	**Input/Output**
Adafruit BNO055 Absolute Orientation Sensor	input
Short button (mode toggle)	input
Tall button (power)	input
Grove Buzzer v1.2	output
RGB led (common cathode)	output

Design Verification

Test Plan of Python Program

The python program processes input values of euler angles (angle_x, angle_y, angle_z) and acceleration (accel_x, accel_y, accel_z) which are recorded by the sensor. These tests were conducted outside of the Design Expo on our own time. Case 3 failed during the Design Expo for no discernible reason. We rebooted the program multiple times and it worked on the third try. It is something we need to look out for in future revisions.

Case #	Input(s)	(and explanation of inputs)	Expected Output(s)	Actual Output	P/F
1	(0, 0, -9.8) (0, 0, -4,5) (0, 0, 0)	Large change in acceleration (over short time, ~1s)	High alarm	High alarm	P
2	(0, 0, -9.8) (0, 0, -9.0) … (0, 0, 0)	Large or small change in acceleration (over long time, ~10s)	No alarm	No alarm	P
3	(0, 0, -9.8) (0, 0, -7.0) (0, 0, -11.0)	Small changes in acceleration over short time (e.g. walking with sensor, rolling over in bed)	No alarm	No alarm	P
4	(None, None, None)	None outputted by sensor (acceleration or euler angles)	"Sensor error"	"Sensor error"	P
5	(0, 0, -180) (0, 0, -120)	Decrease in z-euler angle (baby stands up)	Low alarm	Low alarm	P

	(0, 0, -105) (0, 0, -100)				
6	(0, 0, -180) (0, 0, -120) (0, 0, -175)	Sudden decrease in z-euler angle (sensor inaccuracy)	No alarm	No alarm	P
7	(0, -90, -180) (0, -110, -120) (0, -110, -105) (0, -120, -100)	Decrease in y-euler angle (baby rolls over)	No alarm	No alarm	P
8	(0, -90, -180) Rotation button pressed (0, -110, -120) (0, -110, -105) (0, -120, -100)	Decrease in y-euler angle (baby rolls over) but ‘activate rotation alarm’ button is pressed	Low alarm	Low alarm	P
9	(0, -90, -180) (0, -150, -120) (0, -90, -105) (0, -90, -100)	Sudden decrease in y-euler angle (sensor inaccuracy)	No alarm	No alarm	P
10	Power button held for 1 second	(at program start)	No output, Power LED remains off	No output, Power LED remains off	P
11	Power button held for 3 seconds	(at program start)	Power LED turns on, main loop starts	Power LED turns on, main loop starts	P
12	Power button held for 1 second	(while main loop is working)	No change in buzzer output, Power LED remains on	No change in buzzer output, Power LED remains on	P

13	Power button held for 3 seconds	(while main loop is working)	Power LED turns off, main loop stops, no printing to console	Power LED turns off, main loop stops, no printing to console	P
14	Power button held for 6 seconds	(at program start)	Main loop turns on and then off	Main loop turns on and then off	P
15	Rotation button pressed	(before power button held, before main loop)	No change	No change	P
16	Rotation button pressed	(during main loop)	Rotation LED turns on	Rotation LED turns on	P
17	Wiring disconnected during testing	-	“Error, please check your wiring!”	“Error, please check your wiring!”	P

Verification of Physical Computing Prototype

In order to verify that the physical computing prototype functioned properly, the following procedures were followed:

Orientation Testing (Objectives 2 and 4 of Program)

This procedure tests either the rotation or standing up alarm.

1. Place the sensor on the DARH (see **Figure 7**) with the flat plane of the housing facing parallel to the ground (i.e. with the sensor 'face up')
2. Depending on the configuration of the orientation sensor on the DARH (refer to the stick figure drawn on the sensor) this procedure tests for either standing up or rolling over. Note that when testing the rolling over feature, it must be enabled by switching to the appropriate mode using the mode-switching button (RGB LED should turn blue).
3. In either scenario, rotate the DARH using the lever by approximately 45 degrees in either direction and hold for a few seconds until a buzzer sounds, alerting of either a 'standing' baby or a 'face-down' baby (see GUI for more information)
    a. If the sensor is quickly tipped over and returned back to its original position, no alarm should sound—this feature prevents sensor inaccuracies from accidentally triggering the alarm

Acceleration Testing (Objective 3 of Program)

This procedure tests the fall detection alarm.

1. Take the sensor outside of the DARH and raise it 1 foot above the ground
2. Drop the sensor onto a bed of bubble wrap. A more intense buzzer sound should start, indicating a fall has occurred. Confirm with the GUI that a fall was detected.
    a. The bubble wrap is to protect the sensor, though it is not absolutely necessary for the program to recognize the fall

Sample Calculations

The following are sample calculations for certain functions written in the program. The remainder of the program operates using various other methods of processing (e.g. boolean comparisons) which do not warrant sample calculations.

Average function

```python
def vector_list_average(vector_list):
    """Returns averaged vector from a list of vectors (with any number o
    dimensions)"""
    return [round(sum(i)/len(i), 2) for i in zip(*vector_list)]
```

This function zips the inputted list of *n*-dimensional vectors into *n* lists containing all entries from each "column" of the vector list. It then calculates the average of the entries in each column vector by summing the entries and dividing by the column length. The averages for each column are outputted to a tuple. Example shown below:

$$input = \begin{matrix} [(10 & 10 & 10), \\ (20 & 20 & 20), \\ (30 & 30 & 30), \\ (40 & 40 & 40)] \end{matrix}$$

Plugging into the average_vector_list function…

$$\underbrace{\begin{matrix} [(10 & 10 & 10), \\ (20 & 20 & 20), \\ (30 & 30 & 30), \\ (40 & 40 & 40)] \end{matrix}}_{input} \quad \xrightarrow{\text{zip columns}} \quad \begin{matrix} [(10 & 20 & 30 & 40), \\ (10 & 20 & 30 & 40), \\ (10 & 20 & 30 & 40)] \end{matrix} \quad \xrightarrow{\text{average each row}} \quad \underbrace{\mathbf{(25, 25, 25)}}_{\text{result}}$$

V_sum function

```python
import math

def v_sum(vec):
    """Returns magnitude of inputted vector (with any number of dimensions)"""
    return math.sqrt(sum(i**2 for i in vec))
```

This function takes in an input vector of *n* dimensions and squares each component before summing all the squares and taking the square root of the sum.

$$input = (3, 4, 5, 6, 7)$$

Plugging into v_sum function…

$(3, 4, 5, 6, 7)$	Square each term and append to a temporary list
$(9, 16, 25, 36, 49)$	Sum up items in squared list
$= \mathbf{135}$	**Result**

Amplitude function

```python
def amplitude(data_set):
    """Determines amplitude from a list of data"""
    return abs(max(data_set)-min(data_set))
```

This function calculates the ‘amplitude’ of a data set by calculating the difference between the maximum and minimum of that data set. Normally, amplitude calculations have a division by 2, but to make the parameters for calculations more intuitive with regards to changes in acceleration, we decided against it. Specifically, we noted that a deceleration of around 9.8m/s^2 would indicate a fall, so we wanted to have the tolerance for the amplitude of the acceleration be around 9.8 instead of around 4.9. We decided to call the function an amplitude calculator regardless for simplicity’s sake.

$$\underset{\boldsymbol{input}}{(0,4,2,8,10,3)} \quad \underset{\rightarrow}{\text{find max and min}} \quad \begin{matrix}\max = 10\\ \min = 0\end{matrix} \quad \underset{\rightarrow}{\text{amplitude} \; = \; (\max - \min)} \quad \textbf{amplitude} = \mathbf{10} \;\; (\textbf{result})$$

Design Critique, Discussion, and Recommendations

Where we were several weeks ago

After many preliminary sketches for the device (see **Figures 13-15** in Appendix), we visualized the product to be a magnetic clip (see **Figure 12**) which would attach to a child's clothing similar to a magnetic name tag with the front piece being on the outside of the clothing and the back piece on the inside. It would monitor the rotation of the child to see if they were on their front or back, and it would also measure rapid acceleration to alert the parents of a fall. Unlike the final design, it did not measure whether the baby was standing up or lying down. After the design review, our mentors raised some concerns and questions about our design which we have since addressed in our final design.

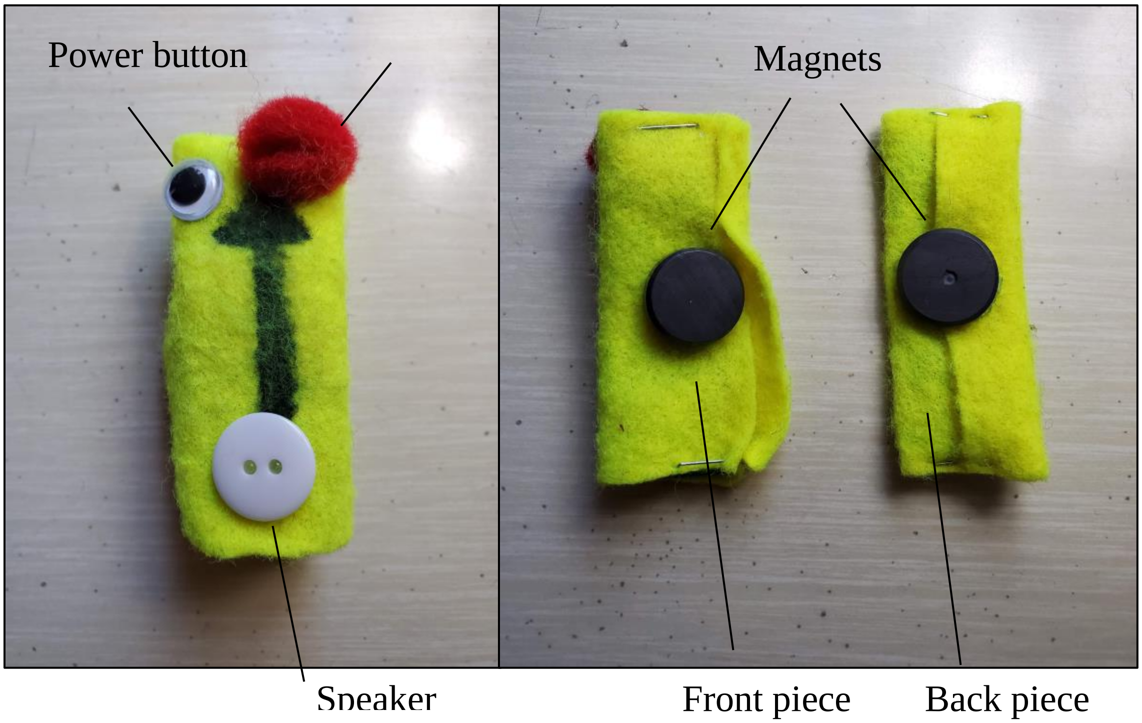


Figure 12: Preliminary prototype for baby orientation monitor (magnetic clip design)

Our mentors emphasized two major concerns they had: that the miniature and easily removable design could present a choking hazard and that the device only seemed to notify parents after the fact and could not prevent a fall from the crib. Regarding the choking hazard, they informed us that young children would, whether intentionally or not, try to remove a wearable device if they had one on their person. They suggested that we develop a method to keep the wearable device away from the child’s reach. This was a difficult task seeing as a wearable device had to be worn on the user in some manner. The mentors suggested that the device be integrated into the child’s nighttime clothing in a comfortable manner. We decided to follow this advice, and the final design is now of a onesie with a buttoned front pocket which houses the low-profile device inside, providing a comfortable and baby proof solution to the placement of the device on the child. As for the lack of preventative measures, the mentors suggested that we make use of the acceleration sensor’s multiple axes to determine whether the child was standing up in their crib or not. After a team discussion, we agreed that the device could trigger an alarm if it found the child to be standing up, and this could warn the parents of the child’s activity in the crib before they attempted to escape and hurt themselves. We kept the fall detection aspect of the design as we found it was still important information to relate to the parents, and we modified the program to link a fall with a louder alarm.

Our mentors also commended us for the thought we put into our design: we had many functions that would collectively provide a detailed representation of the child’s status in the crib, and we had a form which fit the function of our device neatly, namely that a low-profile design would increase comfort for the child.

Where we are now

Further on during the development of our product, we received many questions from inquisitive teaching assistants and other students to which we did not have fully fleshed out answers. Slowly, we incorporated more and more features into our design to increase its robustness. For example, we incorporated a waterproof housing to prevent the device from being susceptible to washing machines and bodily fluids, we allowed the rotation monitor to be turned off in case the child was of age where they could turn onto their back without parental assistance, and we added LEDs on the device to display more information about what functions were on or off. We find that our current design has improved upon many of the flaws we had before, but there is still room for improvement.

For future implementation of our device, we had some features that we would want to add. Firstly, a smartphone app for the parents, which would allow the buzzer notification to directly reach them through an alarm notification rather than scare the child if they were to wake up in the middle of the night or accidentally roll over or fall in their crib. One Design Expo viewer commented that a future design might also be able to connect to a normal two-way audio baby monitor. We decided that a speaker on the device itself was necessary in case access to a smartphone was limited, but with a proper phone app or link to other baby monitors, we could add the ability to mute the buzzer sound on the device itself so that the child is not scared by it. Secondly, we considered adding the ability for the device to alert emergency services if a fall has been detected and the device has not been turned off, similar to a Series 4 Apple Watch [14] which detects falls for elderly users. Thirdly, we could add a microphone on the CBOM which would remove the need for another monitor to check on the child.

A major component of our design on which we had to compromise was the ease of access. More specifically, our design requires that the parent also purchase a specialized piece of clothing with an inbuilt pocket for the sensor. This means that parents cannot easily use their existing child's clothing unless they wanted to sew their own pocket onto it. This was an avenue which we were not able to explore much further due to time constraints, but we weighted the safety of the child over ease of access using our design criteria, which is why the pocket design was chosen. Reflecting on market analysis, further research could be done to develop a locking mechanism similar to the MonBaby Monitor (see *Market Analysis*).

In short, we feel that our design effectively targets the most important design criteria which we sought to fulfill. With further development on the software, miniaturized electrical components, a more accurate sensor, we feel that this device could readily be used in today's market.

References

[1] E. S. Yeh, L. M. Rochette, L. B. McKenzie, and G. A. Smith, “Injuries Associated With Cribs, Playpens, and Bassinets Among Young Children in the US, 1990–2008,” *American Academy of Pediatrics*, 01-Mar-2011. [Online]. Available: https://pediatrics.aappublications.org/content/127/3/479.short. [Accessed: 19-Feb-2020].

[2] B. Goodman, “Cribs Frequent Cause of Injury for Babies, Toddlers,” *WebMD*, 16-Feb-2011. [Online]. Available: https://www.webmd.com/parenting/baby/news/20110216/cribs-frequent-cause-of-injury-for-babies-toddlers#1. [Accessed: 20-Feb-2020].

[3] “Fast Facts About SIDS,” *Eunice Kennedy Shriver National Institute of Child Health and Human Development*. [Online]. Available: https://safetosleep.nichd.nih.gov/safesleepbasics/SIDS/fastfacts. [Accessed: 20-Feb-2020].

[4] A. A. Williamson, E. S. Leichman, R. M. Walters, and J. A. Mindell, “Caregiver-perceived sleep outcomes in toddlers sleeping in cribs versus beds,” *Sleep Medicine*, vol. 54, pp. 16–21, Feb. 2019.

[5] G. A. D. Jonge, A. C. Engelberts, A. J. Koomen-Liefting, and P. J. Kostense, “Cot death and prone sleeping position in The Netherlands.,” *BMJ*, vol. 298, no. 6675, pp. 722–722, 1989.

[6] F. R. L. Jr., Ed., “Sudden Infant Death Syndrome (SIDS) (for Parents) - Nemours KidsHealth,” KidsHealth, Feb-2017. [Online]. Available: https://kidshealth.org/en/parents/sids.html. [Accessed: 22-Feb-2020].

[7]N. Mancall-Bitel, “The 'smart' baby technology raising today's children,” *BBC Worklife*, 29-Nov-2018. [Online]. Available: https://www.bbc.com/worklife/article/20181128-the-smart-baby-technology-raising-todays-children. [Accessed: 23-Feb-2020].

[8] “What is Kynar® Plastic? - PVDF Plastics for Flow Control Parts,” *Industrial Specialties Mfg.*, 2015. [Online]. Available: https://www.industrialspec.com/about-us/blog/detail/what-is-kynar-pvdf-plastic-definition-description. [Accessed: 22-Feb-2020].

[9] G. Pinkerton, *StackPath*, 22-Dec-2002. [Online]. Available: https://www.electronicdesign.com/technologies/boards/article/21763572/many-technologies-contribute-to-miniaturization. [Accessed: 22-Feb-2020].

[10]“Smart Sock 2,” *Owlet Canada*. [Online]. Available: https://owletcare.ca/products/owlet-smart-sock-2-baby-monitor. [Accessed: 22-Feb-2020].

[11]“MonBaby Essential,” *MonBaby Smart Movement Monitor*, 2019. [Online]. Available: https://monbaby.com/products/. [Accessed: 22-Feb-2020].

[12] J. Carbone, “Expect Sensor Prices to Fall,” *DigiKey*, 18-Dec-2013. [Online]. Available: https://www.digikey.com/en/articles/techzone/2013/dec/expect-sensor-prices-to-fall. [Accessed: 22-Feb-2020].

[13] J. Swearingen, “Making My Baby a Smart Baby Was a Mistake,” *Intelligencer*, 10-May-2018. [Online]. Available: https://nymag.com/intelligencer/2018/05/why-using-smart-wearable-baby-monitors-was-a-mistake.html. [Accessed: 22-Feb-2020].

[14] “Use fall detection with Apple Watch,” *Apple Support*, 21-Nov-2019. [Online]. Available: https://support.apple.com/en-ca/HT208944. [Accessed: 22-Feb-2020].

Design Critique, Discussion, and Recommendations

Tangible Prototype Documentation

Preliminary Sketches for Baby Orientation Monitor Idea

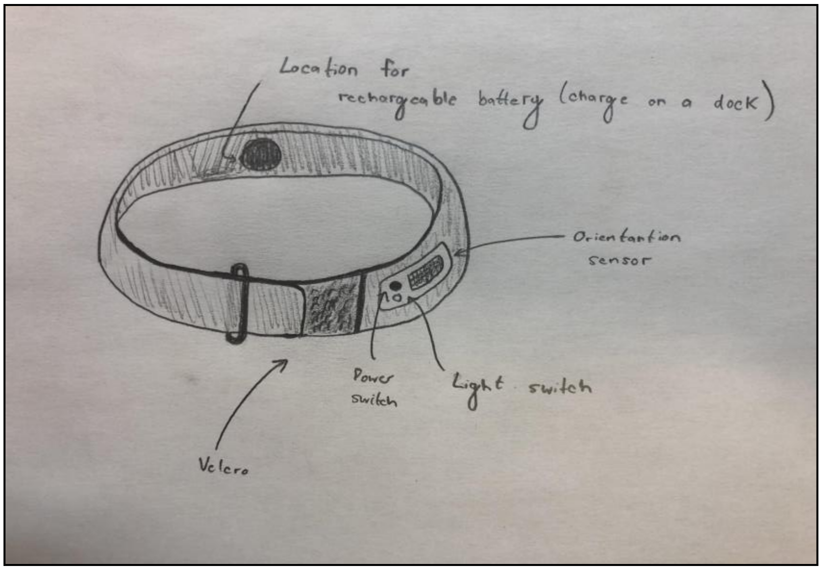


Figure 13: Baby Orientation Monitor (Bracelet form)

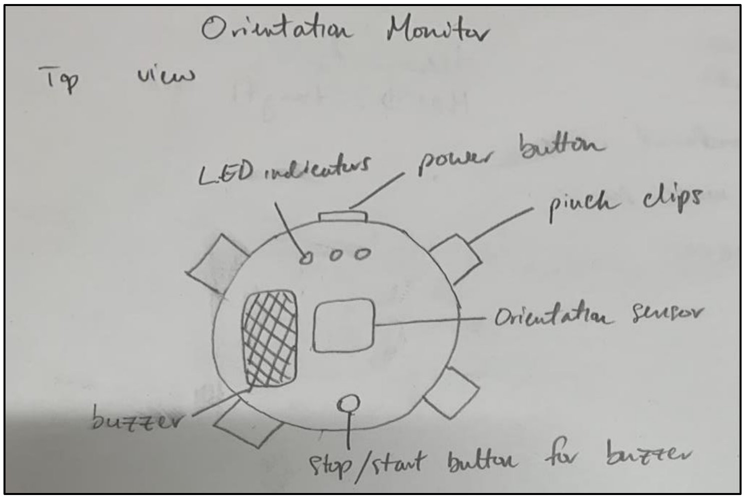


Figure 14: Baby orientation monitor (clip on attachment)

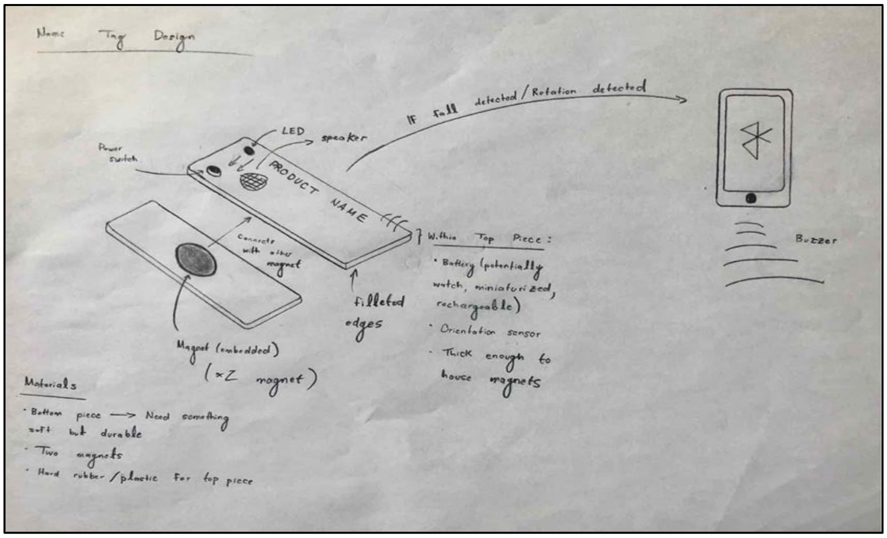
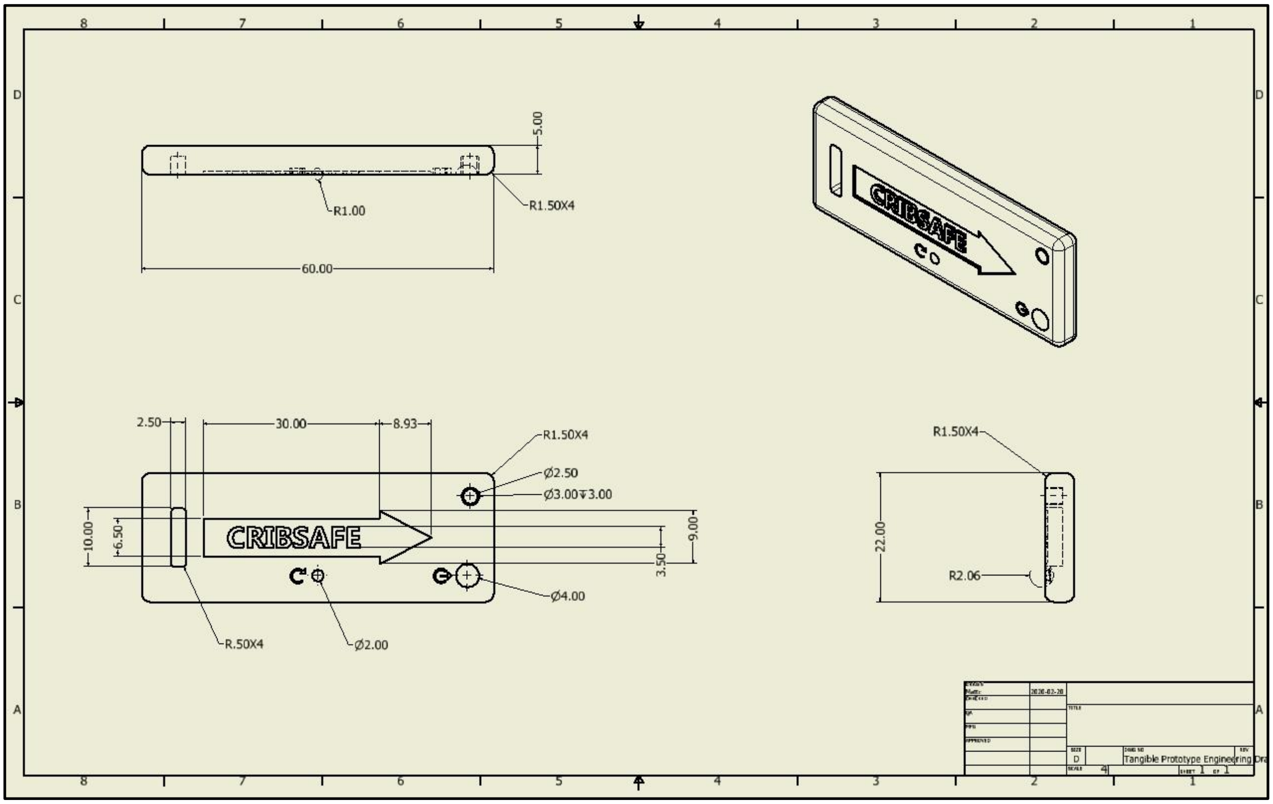

Name Tag Design
IF fall detected / Rotation detected
LED
speaker
Power switch
PRODUCT NAME
Connects with other magnet
filleted edges
Magnet (embedded)
(x2 magnet)
Within Top Piece:
• Battery (potentially watch, miniaturized, rechargeable)
• Orientation sensor
• Thick enough to house magnets
Buzzer
Materials
• Bottom piece —> Need something soft but durable
• Two magnets
• Hard rubber/plastic for top piece

Figure 15: Baby Orientation Sensor (magnetic name-tag)

Tangible Prototype Engineering Drawings



Tangible Prototype Bill of Materials

Parts	Type of Part	Qty.	Cost per Part	Extra Info
Prusa Pla	Printer Plastic (Main Frame)	6.6cm^3	$0.32	
Yellow Acrylic Paint	Decal	1	$5.49(per bottle)	
Red Acrylic Paint	Decal	1	$5.49(per bottle)	
Orange Play Doh	Decal	1	$2.50	Used to give power button colour
Purple Play Doh	Decal	1	$2.50	Used to give power and rotation symbols colour
Foam	Speaker	1	$0.50	
Butto	Custom Onesie	1	$0.10	
Felt	Custom Onesie	1	$0.25	
Thread	Custom Onesie	1	$0.01	

Physical Computing Prototype Documentation

Python Program Code

DISCLAIMER: This code has been formatted and edited to fit properly on 8.5" x 11" paper; attempts to directly copy the below code into a python file may be unsuccessful. Also, the code was split into four different python files (on top of the given sensorlibrary.py file), so it has been separated as such below.

MyThread.py

```
import threading


class MyThread(threading.Thread):
    '''Created a subclass of threads from the Threading module with an instance
       variable 'val' that can be updated and read'''
    def __init__(self, target):
        super().__init__(target=target)
        self.val = [0,0]
```

GUI.py

```
'''
README:
If you run this file directly, a 'proof of concept' GUI will pop up (not connected to
a sensor, so can be used for demonstration purposes if the Pi is not available).
'''

import tkinter as tk
from tkinter.font import Font
from time import sleep
import random
import threading

WIDTH = 850
HEIGHT = 500

root = tk.Tk() # Initialize main window
root.title("CribSafe GUI")

def GUI_loop_random():
    """Updates GUI with random values every second"""

    while True:
        on_off_var["text"] = str(bool(random.getrandbits(1)))
        facingup_var["text"] = str(bool(random.getrandbits(1)))
        standingup_var["text"] = str(bool(random.getrandbits(1)))
        is_fallen_var["text"] = str(bool(random.getrandbits(1)))

        alarm_var["text"] = random.choice(["High", "Low", "Off"])
        avg_accel["text"] = "(" + ",".join([str(random.randint(-3,3)) for i in
                            range(3)]) + ")"
        avg_euler["text"] = "(" + ",".join([str(random.randint(50,150)) for i in
                            range(3)]) + ")"

        sleep(1)
        root.update()
```

objectives.py

```
import math
import random
from time import sleep
import threading
from MyThread import *
from gpiozero import Button, TonalBuzzer

STAND_UP_TOL = 45   # Degrees from horizontal
FACING_UP_TOL = 45  # Degrees from horizontal
BUTTON_TIME = 3     # Number of seconds to hold button for toggle on/off
PWR_BUTTON_PIN = 7
AMPLITUDE_TOL = 9   # Magnitude of acceleration amplitude to trigger buzzer


power_button = Button(PWR_BUTTON_PIN)
# Power button initialized in this file because status function below requires it


def v_sum(vec):
    """Returns magnitude of inputted vector (with any number of dimensions)"""
    return math.sqrt(sum(i**2 for i in vec))


def vector_list_average(vector_list):
    """Returns averaged vector from a list of vectors (with any number of
       dimensions)"""
    return [round(sum(i)/len(i), 2) for i in zip(*vector_list)]

def amplitude(data_set):
    """Determines amplitude from a list of data"""
    return abs(max(data_set)-min(data_set))


def countingbooleans(boolean, boolcounter, tol, alarm_value):
    """Determines if enough outputs of a specific boolean have been received in order
       to warrant switching the buzzer boolean to its 'alarm value'
       (i.e. the value that would trigger an alarm, so 'TRUE' for 'Standing Up' but
       'FALSE' for 'Facing Up'"""
    try:
        boolean_b = not alarm_value
    except TypeError:
        print("Tried inverting a non-boolean!")
    if boolean != alarm_value and boolcounter < tol:
      # Check for 'tol' number of 'boolean' triggers in order to update the ‘buzzer
      # boolean’ which is linked to the buzzer thread and the GUI
        boolcounter = 0
    else:
        boolcounter += 1

    if boolcounter >= tol:
      # Once the buzzer boolean has been triggered, ensure that the alarm remains on
      until the parent switches off the device
        boolean = targetval
        boolean_b = targetval

    return boolean, boolcounter, boolean_b


def obj1_status(is_on):
    """(Status) Checks to see if user is trying to long-press the power button (i.e.
      trying to toggle device on/off)"""
    button_counter = 0

    while power_button.is_pressed:
        button_counter += 1
        sleep(1)

        if button_counter >= BUTTON_TIME:
            button_counter = 0
            is_on = not is_on # Flip 'on' status

            if is_on: # Display appropriate message for shutdown or startup, since
                        this function is used for both purposes
                print("Beginning main process")
            else:
                print("Shutting down")

            return is_on

    return is_on


def obj2_baby_stand_up(avg_euler_z):
    """(Notification) Determines if baby is standing up (according to alignment of
      spine relative to horizontal)"""
    return False if STAND_UP_TOL < abs(avg_euler_z) - 90 else True
    # 90 degrees = standing straight up


def obj3_fall_detection(recent_accels):
    """(Escalation) Determines if baby has fallen (according to the magnitude of the
      amplitude of recent acceleration values)"""
    accel_magnitudes = [v_sum(i) for i in recent_accels]
# Determines magnitude of each acceleration vector in the recent accelerations list
    accel_amplitude = amplitude(accel_magnitudes)
    return True if accel_amplitude > AMPLITUDE_TOL else False
# Checks if amplitude of recent accelerations vectors is too high
# (indicating a fall / large acceleration)

def obj4_facing_up(avg_euler_y):
    """Determines if baby is face down"""
    return False if abs(avg_euler_y) > (90 - FACING_UP_TOL) else True
    # 90 degrees = rolled onto left side; -90 degrees = rolled onto right side
```

main.py

```
### DP3 TEAM 7 PYTHON PROGRAM ###
### Computing Subteam: Dabeer Abdul-Azeez (abdulazd) & Trevor Tung (tungt1)
### Date Submitted: February 14th, 2020

'''''
DISCLAIMER:

This python program has multiple threads to allow for pieces of code to run
concurrently:
    - One for the main loop
    - One for the GUI
    - And one for the buzzer output

The submitted folder includes multiple python files:
    - A main file (main loop, variables, multithread-functions)
    - An objectives file (all functions which are not targeted by their own thread,
including all four objectives)
    - A MyThread file (a threading subclass definition, used in the main file to run
the GUI and buzzer in parallel with the main while loop)
    - A GUI file (which initializes the GUI that is updated in the main file by its
own thread)

    - There is also a logo image which is used by the GUI, feel free to open it up in
full resolution and save it to your desktop
    - and some .idea / venv folders which were made by PyCharm... we didn't want to
delete them just to be safe

A bug we've noticed...
    - Sometimes the 'fall detection' detects false positives when the device is still.
We think this is due to sensor inaccuracies.
      Please restart the program if you face such challenges.

'''



### SETUP ###
```

```
from gpiozero import Button, RGBLED, TonalBuzzer
from sensor_library import *
from objectives import * # Import some functions
from MyThread import * # Import a threading subclass

# Constants
LIST_LENGTH = 10  # Length of list of most recent data points recorded from sensor
REFRESH_RATE = 20 # Refresh rate of GUI and main loop (in Hertz)

STANDING_TOL = 10 # <-- These tolerances correlate to how long an environmental
FALLING_TOL = 6   # condition (i.e. facing down) must be recognized by the program
FACING_TOL = 10   # for it to update the buzzer and actually trigger the alarm

ROT_BUTTON_PIN = 21 # Pins for GPIO devices
BUZZER_PIN = 18
RGBPINS = (17, 27, 22)

# Variables and lists
is_on = False       # Boolean for whether the monitor is on or off
rotation_on = False # Boolean for whether the rotation alarm is on or off

facing_up = True    # Booleans which represent the baby's position
standing_up = False
is_fallen = False

facing_up_b = True    # Extra booleans to be used by the buzzer and GUI (triggered
is_fallen_b = False   # only if above 'normal' booleans are triggered for long
standing_up_b = False # enough (to prevent erratic sensor data from turning
                      # on an alarm

counter_r = 0
counter_f = 0
counter_s = 0

recent_accels = []  # Initialize lists to keep recent readings
recent_eulers = []
recent_averaged_accels = []
avg_accel = ["No input detected"] # Initialize average values for the GUI
avg_euler = ["No input detected"]

# Initialize input/output devices, with 'off' as initial value as necessary
try:
    sensor = Orientation_Sensor()
    rotation_button = Button(ROT_BUTTON_PIN)
    buzzer = TonalBuzzer(BUZZER_PIN, initial_value=None)

# Power button (not seen here) initialized in objectives file as it is not needed in
this file

    status = RGBLED(red=RGBPINS[0], green=RGBPINS[1], blue=RGBPINS[2],
             initial_value=(0,0,0))
except ValueError:
    print("Please check the wiring of your sensor / buzzer / LED / buttons!")
    print("This program will now exit")
    sleep(2)
    exit()


### FUNCTIONS TO BE RUN BY THREADS ###

def buzzer_choice():
    """Plays appropriate buzzer pattern depending on value of appropriate booleans"""

    tune = 0  # Initial value

    while True:
        tune = thread_buzz.val[0] # Read the tune value from the buzzer thread

        # Play appropriate tune based on 'tune' value (no tune played for tune value
of 0)
        if tune == 1:  # Facing down -- Low alarm
            thread_buzz.val[1] = "LOW" # Set buzzer thread's 'alarm' value to "LOW"
            buzzer.play("C5")          # (for use in print statements / GUI)
            sleep(1)
            buzzer.stop()
            sleep(1)

        elif tune == 2:  # Standing Up -- Low alarm
            thread_buzz.val[1] = "LOW"
            buzzer.play("C5")
            sleep(1)
            buzzer.stop()
            sleep(1)

        elif tune == 3:  # Fallen down -- High alarm
            thread_buzz.val[1] = "HIGH"
            buzzer.play("A5")
            sleep(0.1)
            buzzer.play("A4")
            sleep(0.1)
            buzzer.play("A5")
            sleep(0.1)
            buzzer.play("A4")
            sleep(0.1)
            buzzer.stop()
            sleep(0.5)

        sleep(0.2)

def GUI_mainloop():
    from GUI import colors # Import colors dictionary from GUI file
    import GUI

# Imported GUI file inside GUI_mainloop (i.e. inside this 'thread') because of
# Tkinter's limited multithreading capabilities. Specifically, "mainloop()" (called
# below) must be started inside the same thread where Tkinter is imported, meaning
# that 'import GUI' must be done within this GUI_mainloop() function because the GUI
# file itself imports Tkinter

    COL2WIDTH = 6 # Formatting for GUI columns
    COL4WIDTH = 14

    def GUI_update():
        """Updates GUI with new values at the end of each second"""
        # Defined this function inside GUI_mainloop so i) 'after' function below can
        # call it and ii) so it can read/write to GUI labels
        # (imported one level above)

        while True:
            GUI.on_off_var.config(text= "ON" if is_on else "OFF")

            if is_on: # Update GUI labels with appropriate text based on the
                      # values of relevant variables
                if rotation_on:
                    GUI.facingup_var.config(text=("YES" if facing_up_b else
                                            "NO").center(COL2WIDTH),
                                            fg= colors['green'] if facing_up_b else
                                            colors['red'])
                else:
                    GUI.facingup_var.config(text="OFF", fg='white')
                GUI.standingup_var.config(text=("YES" if standing_up_b else "NO"),
                                          fg= colors['red'] if standing_up_b else
                                          colors['green'])
                GUI.is_fallen_var.config(text=("YES" if is_fallen_b else "NO"),
                                       fg= colors['red'] if is_fallen_b else
                                       colors['green'])
                GUI.alarm_var.config(text=thread_buzz.val[1].center(COL4WIDTH))

                if GUI.dev_var.get() == 1: # Hide or show acceleration / euler values
                                           # based on whether 'Developer Options'
                                           # checkbox is ticked
                    GUI.sensoraccel.config(text="SensorAccel:")
                    GUI.avg_accel.config(text=str(avg_accel))
                    GUI.sensoreuler.config(text="SensorAccel:")
                    GUI.avg_euler.config(text=str(avg_euler))

                else:
                    GUI.sensoraccel.config(text="")
                    GUI.avg_accel.config(text="")
                    GUI.sensoreuler.config(text="")
                    GUI.avg_euler.config(text="")

            else: # Show blank lines if program is 'off' according to is_on boolean
                GUI.facingup_var.config(text="-----",fg='white')
                GUI.standingup_var.config(text="-----",fg='white')
                GUI.is_fallen_var.config(text="-----",fg='white')

                GUI.alarm_var.config(text="".center(COL4WIDTH, "-"),fg='white')

                if GUI.dev_var.get() == 1:
                    GUI.sensoraccel.config(text="SensorAccel:")
                    GUI.avg_accel.config(text="".center(COL4WIDTH, "-"))
                    GUI.sensoreuler.config(text="SensorEuler:")
                    GUI.avg_euler.config(text="".center(COL4WIDTH, "-"))

                else: # Hide sensor accel / euler entries if 'Developer Options'
                      # checkbox is not ticked
                    GUI.sensoraccel.config(text="")
                    GUI.avg_accel.config(text="")
                    GUI.sensoreuler.config(text="")
                    GUI.avg_euler.config(text="")

            GUI.root.update()

            sleep(1/REFRESH_RATE)

    GUI.root.after(1000, GUI_update)
    # Runs GUI_update function 1000ms after this line of code is run, allowing
    # for i) mainloop() (below) to be initiated and ii) while True function (in
    # GUI_update) to be run

    GUI.root.mainloop() # Starts main loop for GUI (gets the GUI running)

# Creating / Starting of Threads
thread_buzz = MyThread(buzzer_choice) # Initialize buzzer thread (running
                                      # 'buzzer_choice' function)
thread_buzz.start()
thread_buzz.val = [0, "OFF"]  # thread_buzz.val = ['tune #', 'alarm level']

thread_GUI = threading.Thread(target=GUI_mainloop) # Initialize GUI thread (running
                                                   # 'GUI_mainloop' function)
thread_GUI.start()



### MAIN FUNCTION ###

def main():
    global is_on # Import global variables for modification within the main function
    global rotation_on
    global facing_up_b
    global standing_up_b
    global is_fallen_b
    global facing_up
    global standing_up
    global is_fallen
    global counter_r
    global counter_f
    global counter_s

    global recent_accels
    global recent_eulers
    global recent_averaged_accels
    global avg_accel
    global avg_euler

    print('''''Welcome to the 'CribSafe' Baby Monitoring System! After placing your
          child to sleep, place the device in your child's clothing pocket with the
          arrow pointing up.

    With this device, you can:
    - Press and hold the power button for three seconds to turn the device on
    - Press and hold the power button for three seconds to turn the device off (once
       on)
    - Turn on/off the side-to-side rotation alarm (LED will turn blue when rotation
       alarm is on)
    - Monitor whether your child is standing up or lying down
    - Monitor whether your child has experienced a fall''')

    while True:

        try:

            while not is_on:
                button_counter = counter_f = counter_s = counter_r = 0
                # Reset counters
                thread_buzz.val = [0, "OFF"] # Default values for buzzer thread
                status.off()

                is_on = obj1_status(is_on) # Check if user is trying to power on the
                                           # device

            while is_on: # Main loop if device has been turned on

                if rotation_on: # Green colour for LED if rotation alarm is off,
                                # blue if on
                    status.color = (0, 0, 1)
                else:
                    status.color = (0, 1, 0)
                # Check if user is trying to turn toggle the baby rotation alarm
                if rotation_button.is_pressed:
                    rotation_on = not rotation_on
                    sleep(0.4)  # Have a short sleep to prevent 'rotation_on' from
                                 # toggling back and forth too fast

                # Extract sensor information to variables for further calculations
                accel = sensor.accelerometer()
                euler = sensor.euler_angles()

                # List management (restrict data set to 10 most recent valid sensor
                # inputs)
                if accel[0] is not None:  # Prevents 'None's being read from the
                                          # sensor (Nones preclude proper avg.
                                          # calculations)
                    recent_accels.append(accel)

                if euler[0] is not None:
                    euler = [abs(i) for i in euler] # Read absolute values of euler
                                                    # angles (negative values
                                                    # unnecessary
                    recent_eulers.append(euler)     # for our program, and absolute
                                                    # values improve calculation
                                                    # accuracy)

                if len(recent_accels) > LIST_LENGTH:
       # Limits list length by deleting surplus entries from the beginning of the list
                    diff = len(recent_accels) - LIST_LENGTH
                    del recent_accels[0:diff]

                if len(recent_eulers) > LIST_LENGTH:
                    diff = len(recent_eulers) - LIST_LENGTH
                    del recent_eulers[0:diff]

                # Take average of the most recent readings
                avg_accel = vector_list_average(recent_accels)
                avg_euler = vector_list_average(recent_eulers)

                recent_averaged_accels.append(avg_accel)
       # List of recent averaged accelerations to be used for objective 3 (calculation
       # of amplitude needs multiple data points)

                if len(recent_averaged_accels) > LIST_LENGTH:
                    diff = len(recent_averaged_accels) - LIST_LENGTH
                    del recent_accels[0:diff]

                # Run through objectives
                try:
                    standing_up = obj2_baby_stand_up(avg_euler[2])
                    is_fallen = obj3_fall_detection(recent_averaged_accels)
```

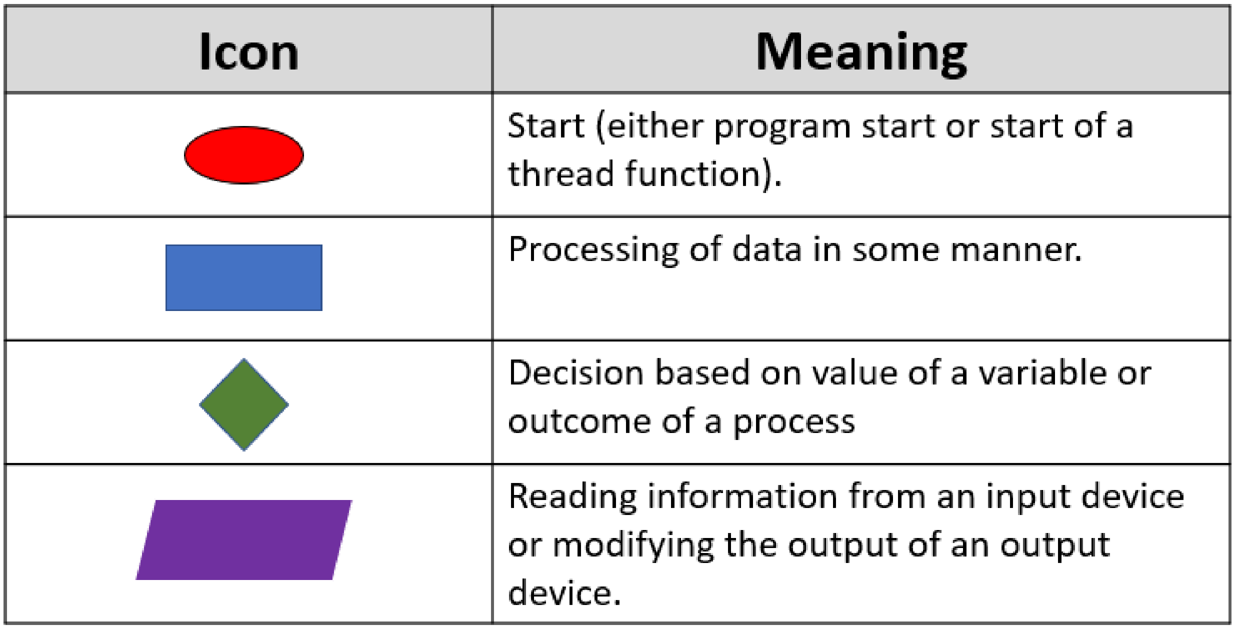
```
                    if rotation_on:
                        facing_up = obj4_facing_up(avg_euler[1])

                except IndexError: # Index Error randomly occurred during testing,
                                   # possibly due to sensor malfunction
                    print("Sensor index error...retaking data.")

                # Check if standing_up / is_fallen / facing_up checks have been
                # triggered for long enough to
                # warrant activating the buzzer and updating the GUI

                standing_up, counter_s, standing_up_b =
                          countingbooleans(standing_up, counter_s, STANDING_TOL, True)
                is_fallen, counter_f, is_fallen_b =
                          countingbooleans(is_fallen, counter_f, FALLING_TOL, True)
                facing_up, counter_r, facing_up_b =
                          countingbooleans(facing_up, counter_r, FACING_TOL, False)

               # Determine appropriate tune to play from value of buzzer booleans and
               # assigned hierarchy
               # (i.e. 'Fallen down' alarm is more important than the 'standing up' or
               # 'facing down' alarms)
                if not facing_up_b:
                    tune = 1
                if standing_up_b:
                    tune = 2
                if is_fallen_b:
                    tune = 3
                if facing_up_b and not standing_up_b and not is_fallen_b:
                    tune = 0

                # Pass the buzzer thread the value of the tune that should be played
                thread_buzz.val[0] = tune

                # Print Statement
                print(f"On?: {is_on} | Rotation Alarm On?: {rotation_on} | Face Up?:
                       {facing_up} | Standing Up?: {standing_up} | Fallen?:
                       {is_fallen} | BUZZER: {thread_buzz.val[1]}")

                is_on = obj1_status(is_on)
                # Check if user is trying to turn off the device

                sleep(1/REFRESH_RATE)
                # Sleep for a bit so as not to overwork the system.

        except OSError or ValueError:

            print("Please check your wiring again and restart the program!")

main()
```

Python Program Flow Chart

Note i): Due to the complexity of the program, the main ideas had to be condensed for the flowchart. Some specifics (e.g. function names, try/excepts) are not included in the flowchart to keep it clean.

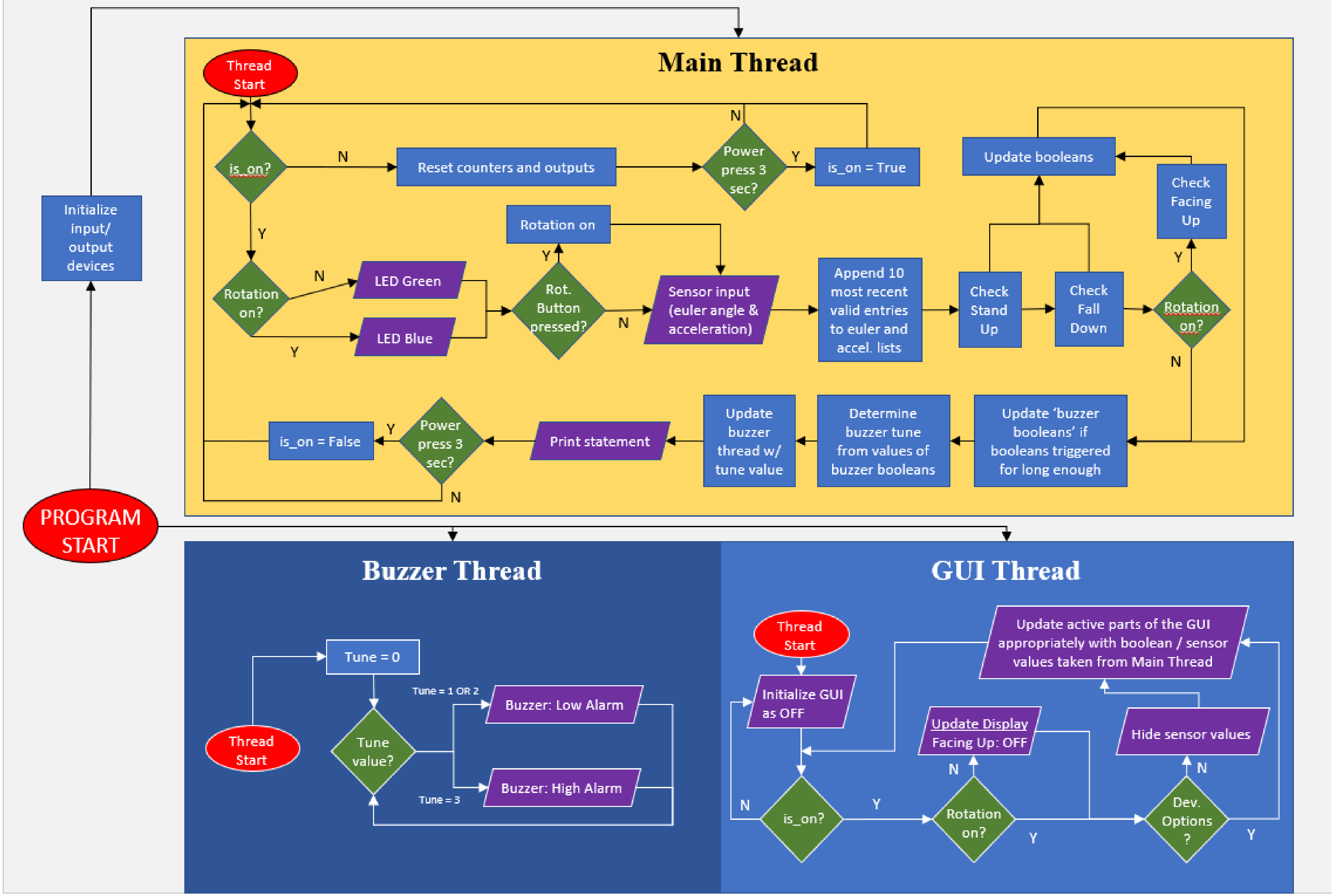
Note ii): Because the program is multithreaded, the flowchart has been split into three separate sections or threads. Each thread can be thought of as its own miniature program, which is why a ‘thread start’ button has been included for each thread in the flow chart.

Legend:



Icon	Meaning
🔴	Start (either program start or start of a thread function).
🟦	Processing of data in some manner.
◆	Decision based on value of a variable or outcome of a process
▰	Reading information from an input device or modifying the output of an output device.

DP3 Team 7 Python Flowchart

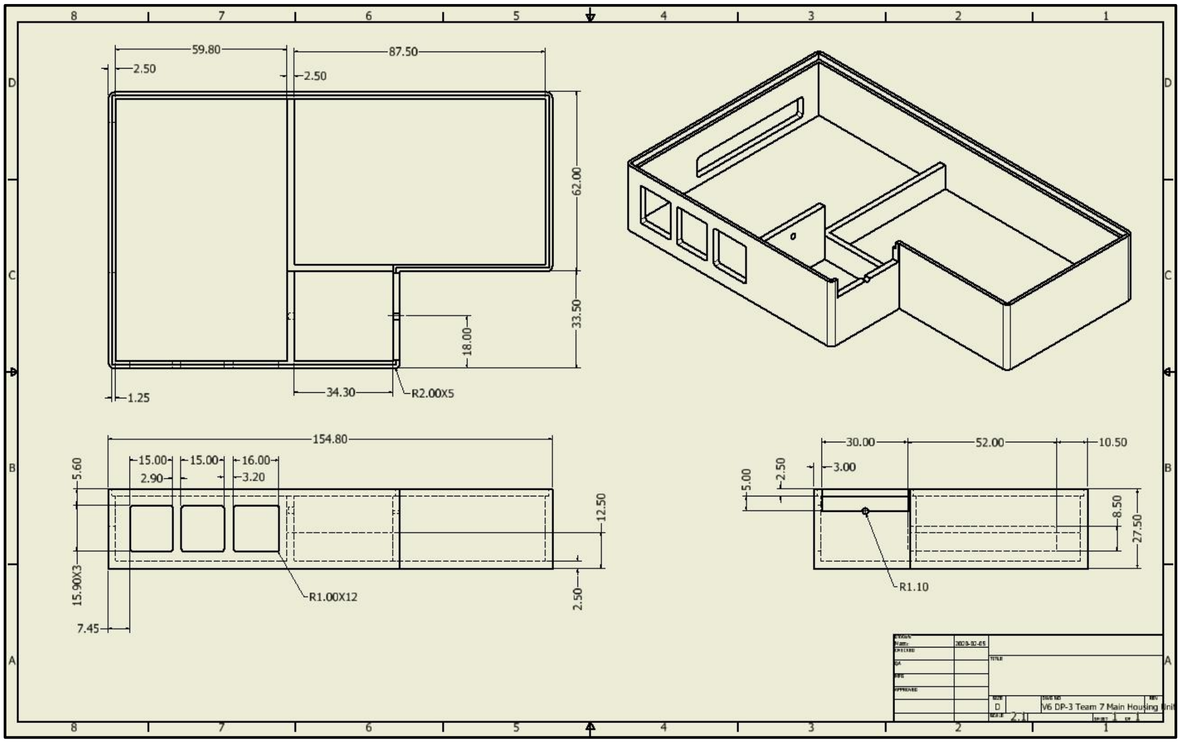


Physical Computing Prototype Bill of Materials

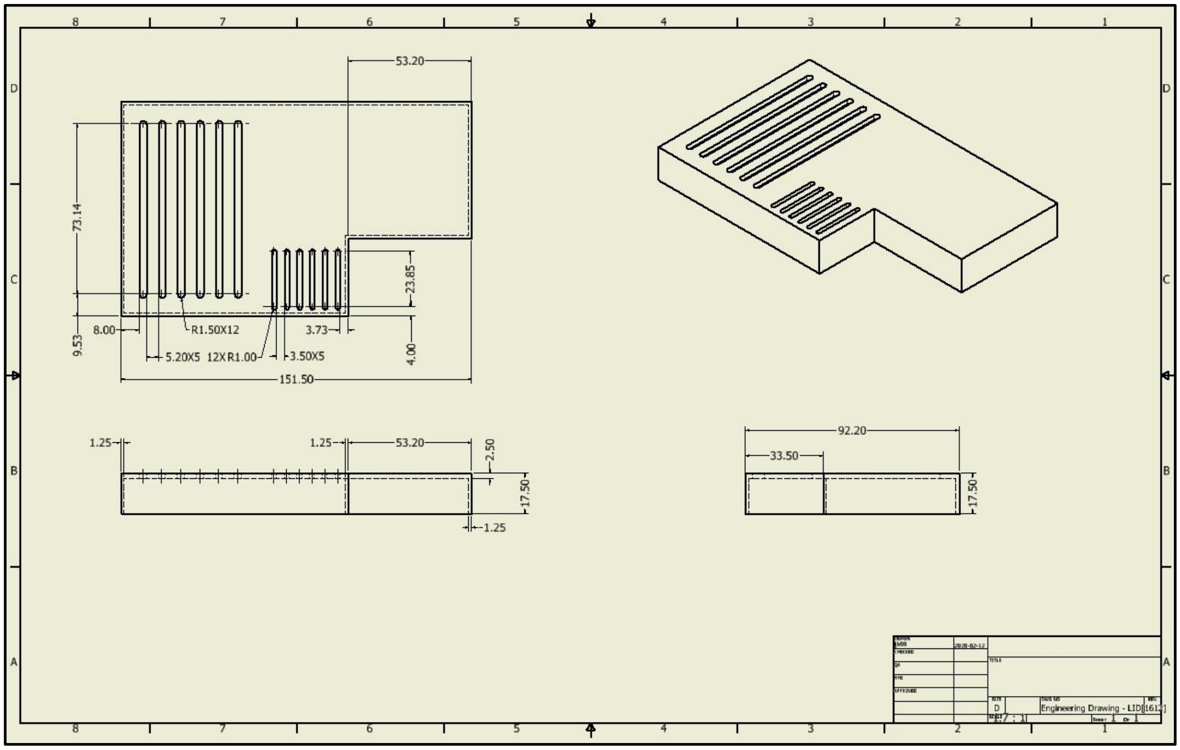
Parts	Type of Part	Qty.	Cost per Part	Total Cost	Extra Info
Raspberry Pi 4B	Electronic	1	$90.00	$90.00	Needs support Frame
Mini-Breadboard	Electronic Housing	1	$3.00	$3.00	Needs support Frame
Monitor	Peripheral	1	$50.00	$50.00	
Mouse	Computation	1	$5.00	$5.00	
Keyboard	Computation	1	$10.00	$10.00	
USB-C Power Cable (for Raspberry Pi)	Power/Cable	1	$6.00	$6.00	
Wall Charger	Power/Cable	1	$4.00	$4.00	
HDMI Cable	Cable	1	$7.00	$7.00	
Grove Buzzer Module	Output	2	$1.50	$3.00	In Breadboard
RGB LED (common cathode)	Output	1	$1.50		In Breadboard
220 Ohm Resistor	Electronic	3	$1.00	$3.00	In Breadboard
Adafruit BNO055 Absolute Orientation Sensor	Sensor	1	$59.00	$59.00	In Breadboard
Push Button	Input	2	$2.00	$4.00	In Breadboard
Male to Male Jumper Cables	Cable	16	$0.15	$2.40	

Main Support Frame	Housing	1	N/A	N/A	3D printed
Support Frame Lid	Housing	1	N/A	N/A	3D printed
Double Axis Rotation Housing	Housing	1	N/A	N/A	3D printed

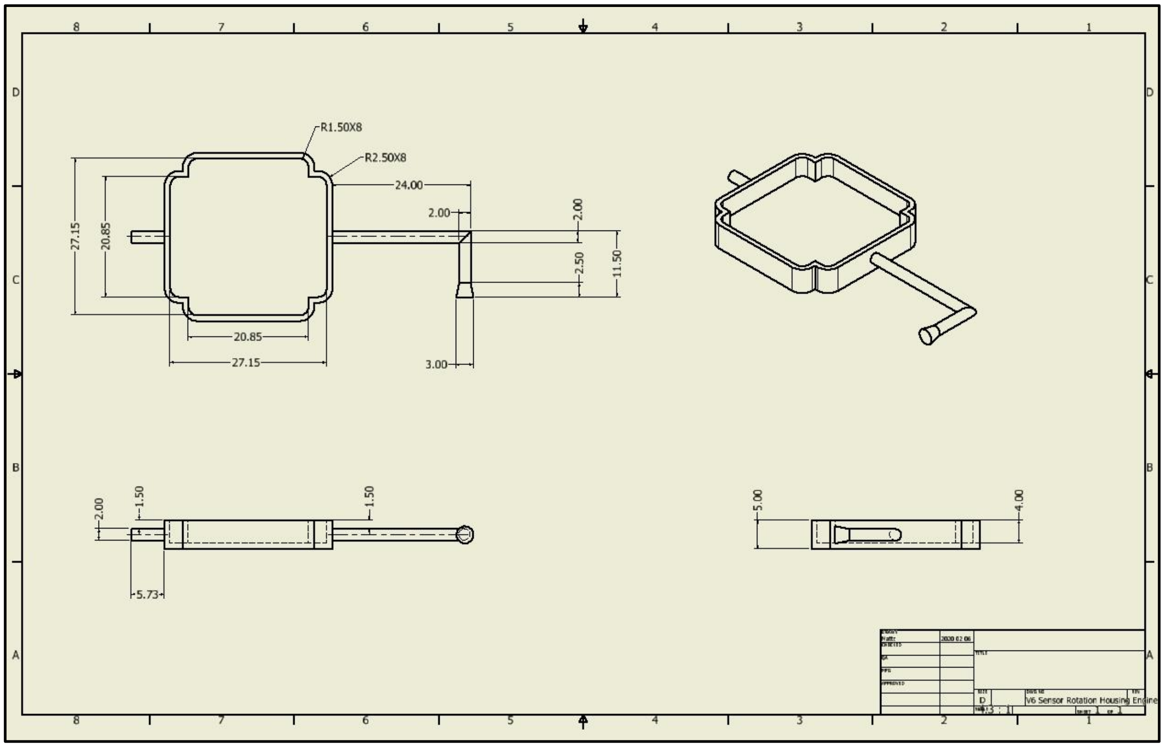
Support Frame Engineering Drawings



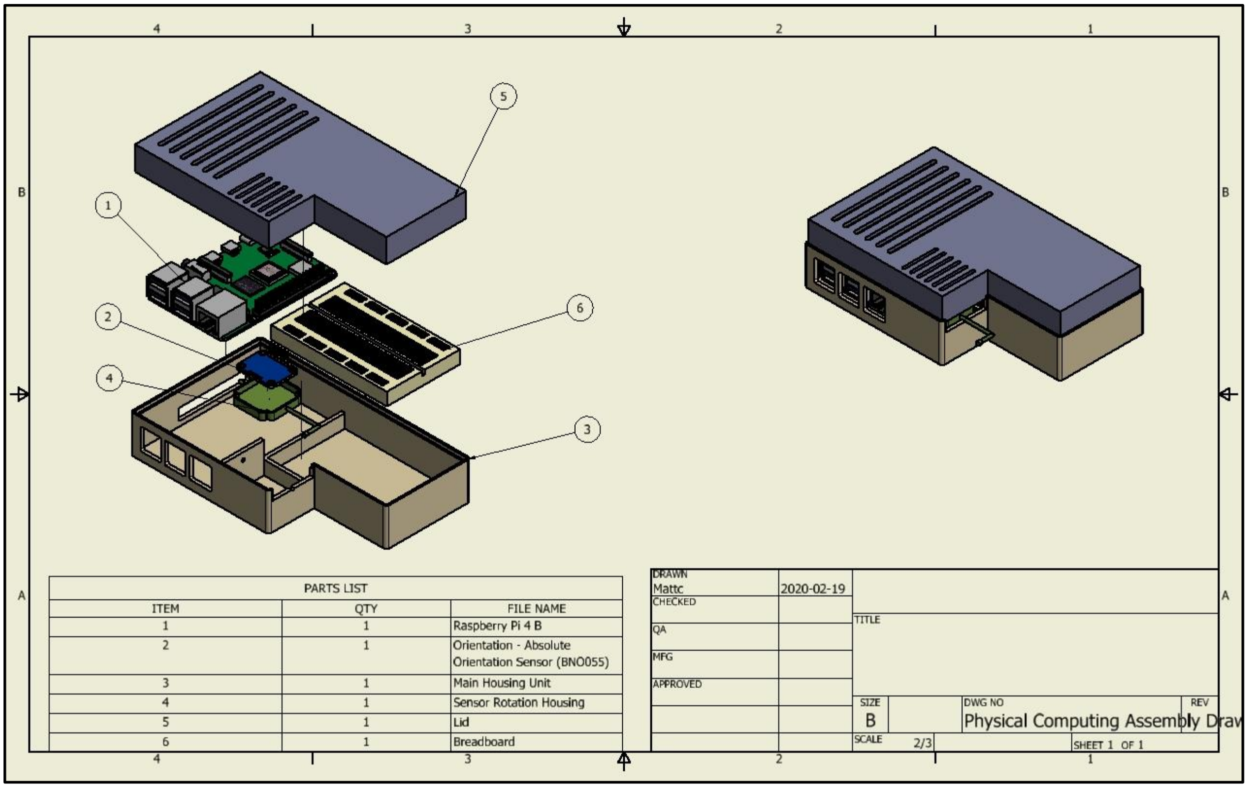
Main Housing Unit



Main Housing Lid



Double Axis Housing System



PARTS LIST		
ITEM	QTY	FILE NAME
1	1	Raspberry Pi 4 B
2	1	Orientation - Absolute Orientation Sensor (BNO055)
3	1	Main Housing Unit
4	1	Sensor Rotation Housing
5	1	Lid
6	1	Breadboard

DRAWN Mattc	2020-02-19	
CHECKED		TITLE
QA		
MFG		
APPROVED		SIZE B — DWG NO Physical Computing Assembly Draw — REV
		SCALE 2/3 — SHEET 1 OF 1

Physical Computing Prototype Assembly

<u>Support Frame: Early Prototype</u>

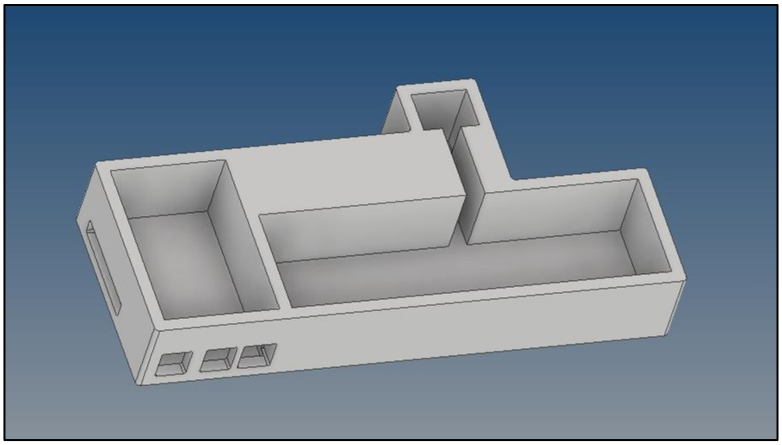


Figure 16: The early design of the main housing unit was less efficient in PLA use with thicker walls and larger dimensions due to the original intention to use a large breadboard. No consideration was given to the incorporation of design verification (i.e. sensor rotation) in the model.

Additional Documents Pertaining to Team’s Progress

Team Contributions

Mathieu Chenier	Dabeer Abdul-Azeez	Trevor Tung	Junhyeong Lee
• CAD modelling for physical computing prototype • CAD modelling for tangible prototype • Engineering Drawings • Manager	• Code for GUI and multithreading • Objectives 2 and 4 of the code (notification and rotation objective) • Main loop for code • Code commenting • Administrator	• Objectives 1 and 3 of the code (status and escalation) • Main loop for code • Code commenting • Soldering, wiring of physical computing prototype • Sewing for tangible prototype • Subject Matter Expert	• CAD modelling for physical computing prototype •Engineering Drawings •Design of the test functionality in the main housing • Coordinator

Summary of team meetings

2020-01-11: Completion of milestone 1: Confirmation of the End-user, Formation of Need Statement, Preparation for milestone 2: Discussion of preliminary designs

2020-01-18: Discussion about the Market research, preparation for milestone 3: Discussion of Refined sketches

2020-01-24: Discussion of CAD file prototype

2020-01-25: Brief explanation about the python file (Computation sub-team) and briefing about the CAD files of housing (Modelling sub-team)

2020-01-26: Completion of the low-fidelity prototype and discussion for the presentation, question preparation for milestone 4-A

2020-01-29: Modelling team meeting: hinge and lid-main housing modification

2020-01-30: Computation sub-team discussion: working on computing prototype

2020-02-07: Completion of the engineering drawings for the modelling files, installation of the computing prototypes and discussion towards the completion of the computation processing

2020-02-09: Discussion about what is due 2020-02-14: engineering drawing, prototype, assembly, computing prototype, design expo (understanding of both computing and CAD process, market research), written proposal

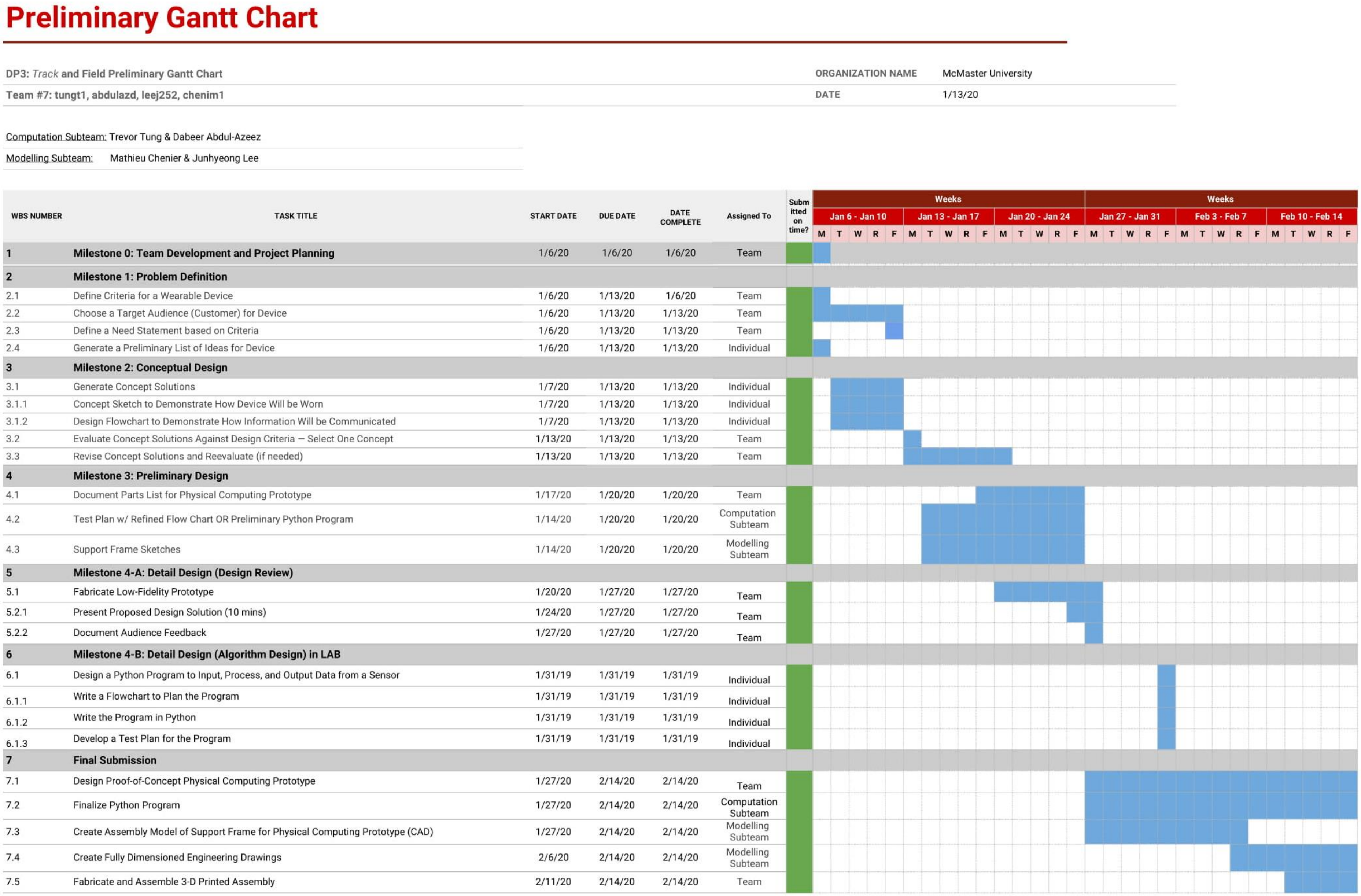
2020-02-11: Modelling sub-team Meeting: Detailed Review of modelling process for Design expo

2020-02-13: Review of tangible prototype and computing prototype, final preparation for the design expo, Recording of the demo video

2020-02-17: Discussion for the completion of the written proposal, division of parts of the written proposal, final verification of the final deliverables that are due 2020-02-24

2020-02-22: Interim Review of the written proposal for final submission.

Preliminary Gantt Chart



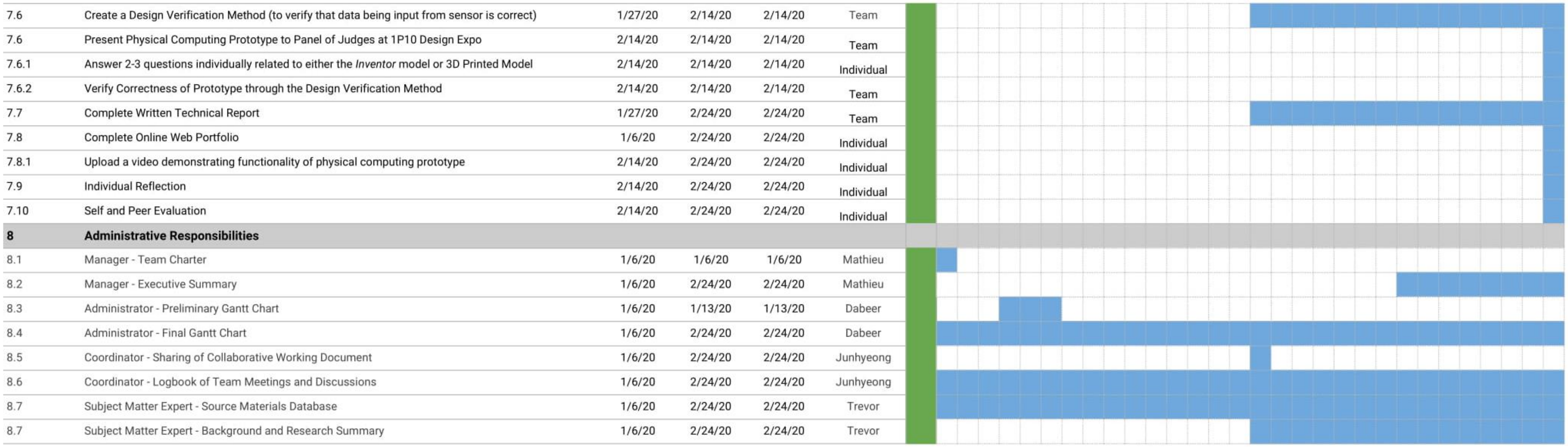
Preliminary Gantt Chart

DP3: *Track* **and Field Preliminary Gantt Chart**		ORGANIZATION NAME	McMaster University
Team #7: tungt1, abdulazd, leej252, chenim1		DATE	1/13/20

Computation Subteam: Trevor Tung & Dabeer Abdul-Azeez

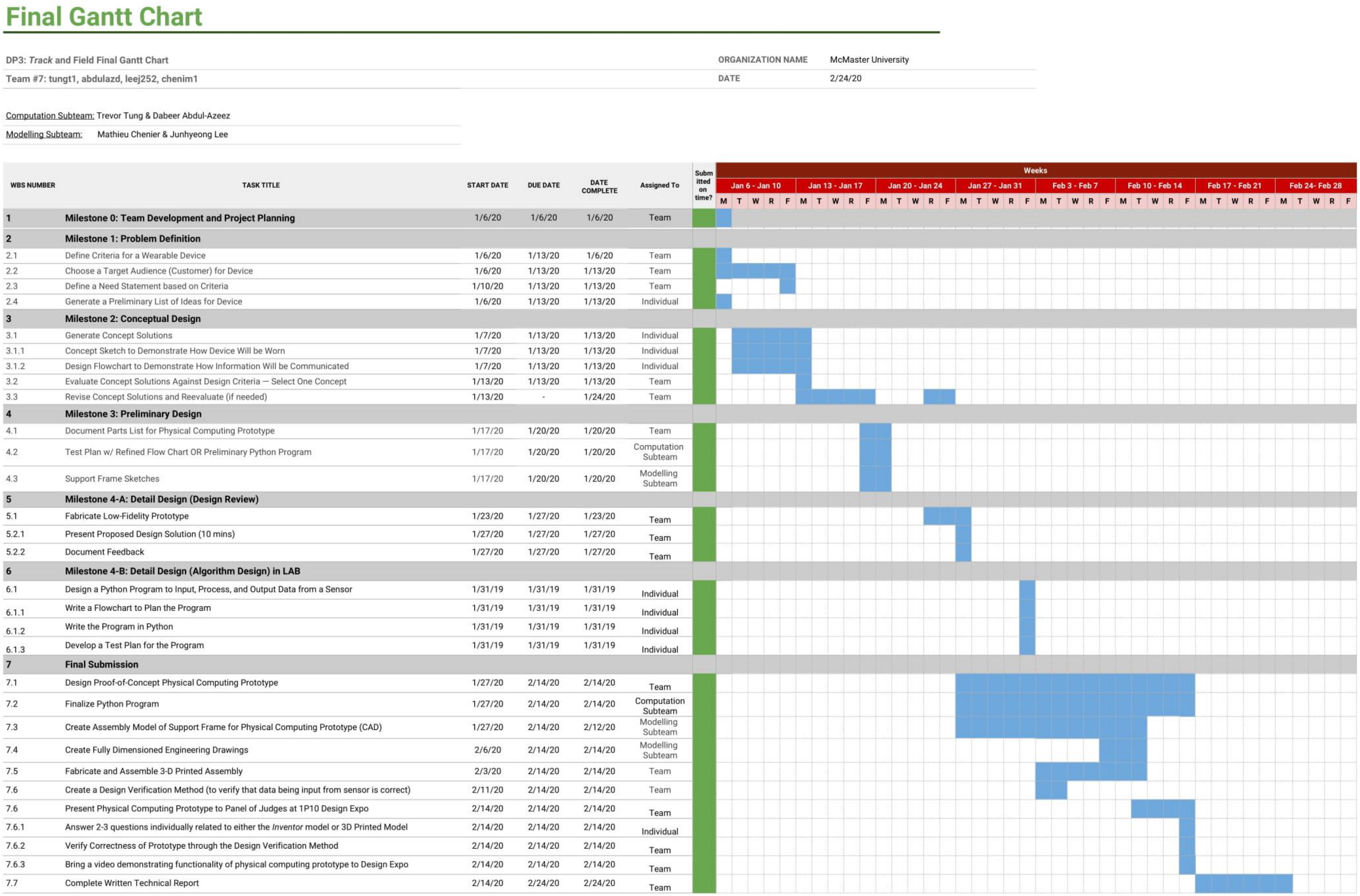
Modelling Subteam: Mathieu Chenier & Junhyeong Lee

WBS NUMBER	TASK TITLE	START DATE	DUE DATE	DATE COMPLETE	Assigned To	Submitted on time?	Weeks															Weeks														
							Jan 6 - Jan 10					Jan 13 - Jan 17					Jan 20 - Jan 24					Jan 27 - Jan 31					Feb 3 - Feb 7					Feb 10 - Feb 14				
							M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F
1	**Milestone 0: Team Development and Project Planning**	1/6/20	1/6/20	1/6/20	Team	🟩	🟦																													
2	**Milestone 1: Problem Definition**																																			
2.1	Define Criteria for a Wearable Device	1/6/20	1/13/20	1/6/20	Team	🟩	🟦																													
2.2	Choose a Target Audience (Customer) for Device	1/6/20	1/13/20	1/13/20	Team	🟩	🟦	🟦	🟦	🟦	🟦																									
2.3	Define a Need Statement based on Criteria	1/6/20	1/13/20	1/13/20	Team	🟩					🟦																									
2.4	Generate a Preliminary List of Ideas for Device	1/6/20	1/13/20	1/13/20	Individual	🟩	🟦																													
3	**Milestone 2: Conceptual Design**																																			
3.1	Generate Concept Solutions	1/7/20	1/13/20	1/13/20	Individual	🟩		🟦	🟦	🟦	🟦																									
3.1.1	Concept Sketch to Demonstrate How Device Will be Worn	1/7/20	1/13/20	1/13/20	Individual	🟩		🟦	🟦	🟦	🟦																									
3.1.2	Design Flowchart to Demonstrate How Information Will be Communicated	1/7/20	1/13/20	1/13/20	Individual	🟩		🟦	🟦	🟦	🟦																									
3.2	Evaluate Concept Solutions Against Design Criteria — Select One Concept	1/13/20	1/13/20	1/13/20	Team	🟩						🟦																								
3.3	Revise Concept Solutions and Reevaluate (if needed)	1/13/20	1/13/20	1/13/20	Team	🟩						🟦	🟦	🟦	🟦	🟦	🟦																			
4	**Milestone 3: Preliminary Design**																																			
4.1	Document Parts List for Physical Computing Prototype	1/17/20	1/20/20	1/20/20	Team	🟩										🟦	🟦	🟦	🟦	🟦	🟦															
4.2	Test Plan w/ Refined Flow Chart OR Preliminary Python Program	1/14/20	1/20/20	1/20/20	Computation Subteam	🟩							🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦															
4.3	Support Frame Sketches	1/14/20	1/20/20	1/20/20	Modelling Subteam	🟩							🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦															
5	**Milestone 4-A: Detail Design (Design Review)**																																			
5.1	Fabricate Low-Fidelity Prototype	1/20/20	1/27/20	1/27/20	Team	🟩											🟦	🟦	🟦	🟦	🟦	🟦														
5.2.1	Present Proposed Design Solution (10 mins)	1/24/20	1/27/20	1/27/20	Team	🟩															🟦	🟦														
5.2.2	Document Audience Feedback	1/27/20	1/27/20	1/27/20	Team	🟩																🟦														
6	**Milestone 4-B: Detail Design (Algorithm Design) in LAB**																																			
6.1	Design a Python Program to Input, Process, and Output Data from a Sensor	1/31/19	1/31/19	1/31/19	Individual	🟩																				🟦										
6.1.1	Write a Flowchart to Plan the Program	1/31/19	1/31/19	1/31/19	Individual	🟩																				🟦										
6.1.2	Write the Program in Python	1/31/19	1/31/19	1/31/19	Individual	🟩																				🟦										
6.1.3	Develop a Test Plan for the Program	1/31/19	1/31/19	1/31/19	Individual	🟩																				🟦										
7	**Final Submission**																																			
7.1	Design Proof-of-Concept Physical Computing Prototype	1/27/20	2/14/20	2/14/20	Team	🟩																🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
7.2	Finalize Python Program	1/27/20	2/14/20	2/14/20	Computation Subteam	🟩																🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
7.3	Create Assembly Model of Support Frame for Physical Computing Prototype (CAD)	1/27/20	2/14/20	2/14/20	Modelling Subteam	🟩																🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦						
7.4	Create Fully Dimensioned Engineering Drawings	2/6/20	2/14/20	2/14/20	Modelling Subteam	🟩																								🟦	🟦	🟦	🟦	🟦	🟦	🟦
7.5	Fabricate and Assemble 3-D Printed Assembly	2/11/20	2/14/20	2/14/20	Team	🟩																											🟦	🟦	🟦	🟦



7.6	Create a Design Verification Method (to verify that data being input from sensor is correct)	1/27/20	2/14/20	2/14/20	Team
7.6	Present Physical Computing Prototype to Panel of Judges at 1P10 Design Expo	2/14/20	2/14/20	2/14/20	Team
7.6.1	Answer 2-3 questions individually related to either the *Inventor* model or 3D Printed Model	2/14/20	2/14/20	2/14/20	Individual
7.6.2	Verify Correctness of Prototype through the Design Verification Method	2/14/20	2/14/20	2/14/20	Team
7.7	Complete Written Technical Report	1/27/20	2/24/20	2/24/20	Team
7.8	Complete Online Web Portfolio	1/6/20	2/24/20	2/24/20	Individual
7.8.1	Upload a video demonstrating functionality of physical computing prototype	2/14/20	2/24/20	2/24/20	Individual
7.9	Individual Reflection	2/14/20	2/24/20	2/24/20	Individual
7.10	Self and Peer Evaluation	2/14/20	2/24/20	2/24/20	Individual
8	**Administrative Responsibilities**				
8.1	Manager - Team Charter	1/6/20	1/6/20	1/6/20	Mathieu
8.2	Manager - Executive Summary	1/6/20	2/24/20	2/24/20	Mathieu
8.3	Administrator - Preliminary Gantt Chart	1/6/20	1/13/20	1/13/20	Dabeer
8.4	Administrator - Final Gantt Chart	1/6/20	2/24/20	2/24/20	Dabeer
8.5	Coordinator - Sharing of Collaborative Working Document	1/6/20	2/24/20	2/24/20	Junhyeong
8.6	Coordinator - Logbook of Team Meetings and Discussions	1/6/20	2/24/20	2/24/20	Junhyeong
8.7	Subject Matter Expert - Source Materials Database	1/6/20	2/24/20	2/24/20	Trevor
8.7	Subject Matter Expert - Background and Research Summary	1/6/20	2/24/20	2/24/20	Trevor

Final Gantt Chart



Final Gantt Chart

DP3: *Track* and Field Final Gantt Chart		ORGANIZATION NAME	McMaster University
Team #7: tungt1, abdulazd, leej252, chenim1		DATE	2/24/20

Computation Subteam: Trevor Tung & Dabeer Abdul-Azeez

Modelling Subteam: Mathieu Chenier & Junhyeong Lee

WBS NUMBER	TASK TITLE	START DATE	DUE DATE	DATE COMPLETE	Assigned To	Submitted on time?
1	**Milestone 0: Team Development and Project Planning**	1/6/20	1/6/20	1/6/20	Team	🟩
2	**Milestone 1: Problem Definition**					
2.1	Define Criteria for a Wearable Device	1/6/20	1/13/20	1/6/20	Team	🟩
2.2	Choose a Target Audience (Customer) for Device	1/6/20	1/13/20	1/13/20	Team	🟩
2.3	Define a Need Statement based on Criteria	1/10/20	1/13/20	1/13/20	Team	🟩
2.4	Generate a Preliminary List of Ideas for Device	1/6/20	1/13/20	1/13/20	Individual	🟩
3	**Milestone 2: Conceptual Design**					
3.1	Generate Concept Solutions	1/7/20	1/13/20	1/13/20	Individual	🟩
3.1.1	Concept Sketch to Demonstrate How Device Will be Worn	1/7/20	1/13/20	1/13/20	Individual	🟩
3.1.2	Design Flowchart to Demonstrate How Information Will be Communicated	1/7/20	1/13/20	1/13/20	Individual	🟩
3.2	Evaluate Concept Solutions Against Design Criteria — Select One Concept	1/13/20	1/13/20	1/13/20	Team	🟩
3.3	Revise Concept Solutions and Reevaluate (if needed)	1/13/20	-	1/24/20	Team	🟩
4	**Milestone 3: Preliminary Design**					
4.1	Document Parts List for Physical Computing Prototype	1/17/20	1/20/20	1/20/20	Team	🟩
4.2	Test Plan w/ Refined Flow Chart OR Preliminary Python Program	1/17/20	1/20/20	1/20/20	Computation Subteam	🟩
4.3	Support Frame Sketches	1/17/20	1/20/20	1/20/20	Modelling Subteam	🟩
5	**Milestone 4-A: Detail Design (Design Review)**					
5.1	Fabricate Low-Fidelity Prototype	1/23/20	1/27/20	1/23/20	Team	🟩
5.2.1	Present Proposed Design Solution (10 mins)	1/27/20	1/27/20	1/27/20	Team	🟩
5.2.2	Document Feedback	1/27/20	1/27/20	1/27/20	Team	🟩
6	**Milestone 4-B: Detail Design (Algorithm Design) in LAB**					
6.1	Design a Python Program to Input, Process, and Output Data from a Sensor	1/31/19	1/31/19	1/31/19	Individual	🟩
6.1.1	Write a Flowchart to Plan the Program	1/31/19	1/31/19	1/31/19	Individual	🟩
6.1.2	Write the Program in Python	1/31/19	1/31/19	1/31/19	Individual	🟩
6.1.3	Develop a Test Plan for the Program	1/31/19	1/31/19	1/31/19	Individual	🟩
7	**Final Submission**					
7.1	Design Proof-of-Concept Physical Computing Prototype	1/27/20	2/14/20	2/14/20	Team	🟩
7.2	Finalize Python Program	1/27/20	2/14/20	2/14/20	Computation Subteam	🟩
7.3	Create Assembly Model of Support Frame for Physical Computing Prototype (CAD)	1/27/20	2/14/20	2/12/20	Modelling Subteam	🟩
7.4	Create Fully Dimensioned Engineering Drawings	2/6/20	2/14/20	2/14/20	Modelling Subteam	🟩
7.5	Fabricate and Assemble 3-D Printed Assembly	2/3/20	2/14/20	2/14/20	Team	🟩
7.6	Create a Design Verification Method (to verify that data being input from sensor is correct)	2/11/20	2/14/20	2/14/20	Team	🟩
7.6	Present Physical Computing Prototype to Panel of Judges at 1P10 Design Expo	2/14/20	2/14/20	2/14/20	Team	🟩
7.6.1	Answer 2-3 questions individually related to either the *Inventor* model or 3D Printed Model	2/14/20	2/14/20	2/14/20	Individual	🟩
7.6.2	Verify Correctness of Prototype through the Design Verification Method	2/14/20	2/14/20	2/14/20	Team	🟩
7.6.3	Bring a video demonstrating functionality of physical computing prototype to Design Expo	2/14/20	2/14/20	2/14/20	Team	🟩
7.7	Complete Written Technical Report	2/14/20	2/24/20	2/24/20	Team	🟩

7.8	Complete Online Web Portfolio	2/20/20	2/28/20	2/28/20	Individual
7.8.1	Upload a video showcasing purpose and motivation of commercialized design to Web Portfolio	2/25/20	2/28/20	2/25/20	Individual
7.8.2	Upload a video demonstrating functionality of physical computing prototype to Web Portfolio	2/25/20	2/28/20	2/25/20	Individual
7.9	Individual Reflection	2/22/20	2/28/20	2/28/20	Individual
7.10	Self and Peer Evaluation	2/22/20	2/28/20	2/28/20	Individual
8	**Administrative Responsibilities**				
8.1	Manager - Team Charter	1/6/20	1/6/20	1/6/20	Mathieu
8.2	Manager - Executive Summary	2/18/20	2/24/20	2/24/20	Mathieu
8.3	Administrator - Preliminary Gantt Chart	1/8/20	1/13/20	1/13/20	Dabeer
8.4	Administrator - Final Gantt Chart	1/8/20	2/24/20	2/24/20	Dabeer
8.5	Coordinator - Sharing of Collaborative Working Document	2/21/20	2/24/20	2/24/20	Junhyeong
8.6	Coordinator - Logbook of Team Meetings and Discussions	1/8/20	2/17/20	2/24/20	Junhyeong
8.7	Subject Matter Expert - Source Materials Database	1/6/20	2/24/20	2/24/20	Trevor
8.7	Subject Matter Expert - Background and Research Summary	2/17/20	2/24/20	2/24/20	Trevor