SmartWatch

Software and Hardware Specification Sheet

Baltej Bal, Jerreh Janneh, Thomas Aziz

April 15, 2020

Declaration of Joint Authorship

We, Thomas Aziz, Baltej Bal and Jerreh Janneh, confirm we are submitting the joint work of our group, expressed in our own words. Any content within this submission authored by a non-group member in any form (ideas, equations, figures, texts, tables, programs), is properly acknowledged at the point of use. A list of the references used is included. The work breakdown is as follows: Each group member provided a sensor or effector, alongside functioning and documented hardware for interfacing with it, designed and implemented in a prior, individual effort. Thomas Aziz provided an AXDL345 accelerometer. Baltej Bal provided an ADS1015 pulse sensor. Jerreh Janneh provided the TMP006 temperature sensor. While our work is a joint effort, each member will lead one specific aspect of the design and integration of our sensors. Baltej is the lead for further development of our mobile application. Thomas will lead hardware development efforts, and Jerreh is the lead for connecting the two via a database system.

Proposal

January 17th 2020

Proposal for the development of SmartWatch

Prepared by Baltej Singh Bal, Jerreh Janneh, Thomas Aziz

Computer Engineering Technology Student

https://github.com/Breezydust/SmartWatch

Executive Summary

As a student in the Computer Engineering Technology program, I will be integrating the knowledge and skills I have learned from our program into this Internet of Things themed capstone project. This proposal requests the approval to build the hardware portion that will connect to a database as well as to a mobile device application. The internet connected hardware will include a custom PCB with the following sensors and effectors: ADXL345 accelerometer, ADS1015 12-bit ADC, SEN-11574 pulse sensor TMP006 IR temperature sensor. The database will store: Username, Password, Timestamp, Temperature Reading, Heart Beat Readings and Steps Walked Readings. The mobile device functionality will include accessing the Timer, Stopwatch, Temperature readings, User Information, Steps Walked Readings, Heart Beat Readings and Customization Features will be further detailed in the mobile application proposal. I will be collaborating with the following company/department: Humber Media Study's. In the winter semester I plan to form a group with the following students, who are also

building similar hardware this term. These are the following group members: Baltej Bal, Jerreh Janneh, Thomas Aziz. The hardware will be completed in CENG 317 Hardware Production Techniques independently and the application will be completed in CENG 319 Software Project. These will be integrated together in the subsequent term in CENG 355 Computer Systems Project as a member of a 2 or 3 student group.

Background

To begin we'd like to thank the Humber College Institute of Technology & Advanced Learning Computer Engineering Technology Capstones for their guidance in our progress. Their experience in hardware and software design is immeasurably helpful as we continue to further our project.

Wearables have become immensely popular devices that utilize Bluetooth connections and serve as integral IoT devices in the daily lives of a large population. The release of the Samsung Galaxy Gear in 2013 – a smart watch with an integrated heartbeat sensor and pedometer - created a whole new market for smart wearable devices. The Apple Watch currently dominates the market, with over 45% market share (Liu, 2019). The Apple Watch provides a swath of very advanced functionality, including tethered messaging, fitness tracking, biometric monitoring and cellular connectivity.

We intend to create a watch that focuses primarily on fitness tracking – an area that the current market leader is not completely focused on. We hope that our watch and accompanying software will allow users to get a more in-depth look into their daily fitness levels by providing them easy-to-read data on movement, heart rate and body temperature.

Our sensor choices will allow us to collect all the data we require, and the application will allow for us to represent the data to the user in a meaningful way.

Each sensor functions as follows:

TMP006 - The TMP006 IR Temperature sensor will allow a contactless analysis of the wearer's ambient body temperature. When integrated into the smartwatch it will let the wearer know his/her temperature and provide a suggestion. These suggestions will be dependent on the recorded temperature between the ranges of -40°C ~ 125°C. (Texus Instruments Inc., 2012)

ADXL345 – The AXDL345 accelerometer is a triple-axis accelerometer that measures the static acceleration of gravity. When integrated into the smartwatch it will allow for the device to record the user's movement during exercise. When coupled with a pulse sensor it will allow us to measure the user's exercise intensity and display suggestions on trends, overall intensity, and calories burned during exercise. (Analog Devices, 2015)

ADS1015 - The pulse/heart-rate sensor amped is a plug-and-play heart-rate sensor for Arduino. The pulse/heart rate sensor can be used to incorporate live heart-rate data and send it to the device's software which will use the pulse/heart-rate sensor to measure the pulses and will alert the user for any drastic changes in the user's vital signs. (Texus Instruments Inc., 2018)

Humber College intends to present a viewing of capstone projects of various designs with the intent of showing in further detail how their capstones work. The smartwatch wearable will be present to demo to potential industry investors.

Existing products on the market include Apple Watch's. I have searched for prior art via Humber's IEEE subscription selecting Institute of International Symposium and have found and read A DIY approach to a pervasive computing for the Internet of Things: a smart watch which provides insight into similar efforts.

In the Computer Engineering Technology program, we have learned about the following topics from the respective relevant courses:

- Java Docs from CENG 212 Programming Techniques in Java,
- Construction of circuits from CENG 215 Digital and Interfacing Systems,
- Rapid application development and Gantt charts from CENG 216 Intro to Software Engineering,
- Micro computing from CENG 252 Embedded Systems,
- SQL from CENG 254 Database with Java,
- Web access of databases from CENG 256 Internet Scripting; and,
- Wireless protocols such as 802.11 from TECH152 Telecom Networks.

This knowledge and skill set will enable me to build the subsystems and integrate them together as my capstone project.

Methodology

This proposal is assigned in the first week of class and is due at the beginning of class in the second week of the fall semester. My coursework will focus on the first two of the 3 phases of this project:

Phase 1 Hardware build.

Phase 2 System integration.

Phase 3 Demonstration to future employers.

Phase 1 Hardware build

The hardware build will be completed in the fall term. It will fit within the CENG Project maximum dimensions of 12 13/16" x 6" x 2 7/8" (32.5cm x 15.25cm x 7.25cm) which represents the space below the tray in the parts kit. The highest AC voltage that will be used is 16Vrms from a wall adaptor from which +/- 15V or as high as 45 VDC can be obtained. Maximum power consumption will be 20 Watts.

Phase 2 System integration

The system integration will be completed in the fall term.

Phase 3 Demonstration to future employers

This project will showcase the knowledge and skills that I have learned to potential employers.

The brief description below provides rough effort and non-labor estimates respectively for each phase. A Gantt chart will be added by week 3 to provide more project schedule details and a more complete budget will be added by week 4. It is important to start tasks as soon as possible to be able to meet deadlines.

Display screen for hardware element required. Materials for creation of the device. Casing for sensors and Raspberry Pi. Additional connectors to link up sensors to one another.

Concluding remarks

This proposal presents a plan for providing an IoT solution. The hardware device is a convenient option for those that want to maintain a healthy life style all while being able to view their current temperature readings, heart rate and steps walked of their body without having to open up external applications. This is an opportunity to integrate the knowledge and skills developed in our program to create a collaborative IoT capstone project demonstrating my ability to learn how to support projects such as the initiative described by [3]. I request approval of this project.

References

[3] M. Schickler *et al*., "Using Wearables in the Context of Chronic Disorders: Results of a Pre-Study," *2016 IEEE 29th International Symposium on Computer-Based Medical Systems (CBMS)*, Dublin, 2016, pp. 68-69.

Abstract

The purpose of this project is to create a device which allows the user to be more active with their healthy living and for those who want to be notified with any changes in their vitals. It will consist of an application which will link up to a physical hardware element via Wi-Fi This technical report will give a thorough analysis of the development process regarding every aspect of the project, ranging from app structure to hardware assembly. Explanations revolving around certain ideals and decisions will be given to provide readers insight on why certain features are present along with their purpose.

Due to uncertain circumstances Test cases may not be available, we will be examined during development to acknowledge mistakes that were made along the road along with the solutions for said mistakes. Any hardships and difficulties will also be documented for prevention purposes in the future of similar circumstances. Hardware explanation and breakdowns can be located in the Requirement Specification portion of the report.

Table of Contents

Illustrations

Introduction

A smartwatch is a wearable computer that provides auxiliary functionality to a smart phone. This functionality can range anywhere from simple telecommunication tools to dedicated, untethered cellular connectivity and biometric tracking. There are a variety of smartwatch devices already on the market each with different features, designs and functionality.

Smartwatch design is focused around three main areas; the physical form of the watch (including the development platform of choice and enclosure design around said platform), the connectivity options of the watch (tethered to a phone, dedicated cellular modem), and the primary software functionality of the watch. The latter two areas follow the first – connectivity and software functionality is contingent on the variety of sensors, effectors and processors used within the watch.

Our smartwatch is primarily a health-focused device and our choices of sensors reflect that. The ADXL345 accelerometer allows for user movement tracking. The ADS1015 pulse sensor will let the device monitor the user’s pulse to help gauge active calories burned during exercise. The TMP006 temperature sensor allows the device to monitor the user’s body temperature and send alerts based on body temperature data. All data will be processed with a dedicated app on a connected Android device, and displayed in-depth within the app. The display of the watch itself will show the user quick tidbits of information over time, and allow for quick changes and modifications to watch functionality.

Project Description

Purpose of Project

Problem

Our generation these days struggle to live an active living style due to obvious reasons. More young adults and teens don’t have the motivation to work out and also the older generation does not know how to check their heartrate. With technology today, an easygoing and agreeable smartwatch is easily wearable. I am working on a wearable device or also known as a smart watch that detects the pulse/heart rate, the body temperature, and the motion detection of the body. The pulse/heart rate sensor amped is a plug-and-play heart-rate sensor for Arduino pulse/heart rate sensor can be used to incorporate live heart-rate data into our project.

Solution

The idea of the S-watch was adapted from similar smart watch’s using similar integrated features that can already be found on the market. Apple watch, Fitbit, Galaxy Watch, are just some of the smart watch’s already on the market with the same technological features that we are going to have on the S-watch. These watch’s use accelerometer which counts the users' motion otherwise known as steps walked, distance travelled, and calories burned. Our S-watch device makes for a unique design compared to our competitors as we will use the pulse/heart rate sensor to measure the heart rate and will alert the user for any drastic changes in the user’s vital signs.

Requirement Specifications

Hardware

The hardware portion of this project will be a joint effort between each member of the group as there are many responsibilities in order for everything to function as intended. In terms of the hardware design, enclosure, PCB, and Soldering, it will be handled by jerreh. The functionality of each sensor will be tested and operational mainly by Thomas with help from Jerreh when required. Connections between sensors and the Raspberry Pi will be accomplished by Jerreh. The Integration of components may require additional help from every member due to problems that may occur during development.

The project utilizes many hardware components such as the Raspberry Pi Zero W, ADXL345 accelerometer, ADS1015 Analog to digital converter also the SEN-11754 pulse sensor, and a TMP006 temperature sensor and a pyboard Color LCD. These sensors will be enclosed using 3D printed and laser cut materials. The new inclusion for the project is the Display Screen as it is crucial to display the information from the application. The android smartphone’s role will act as the device’s remote as it can communicate with the smartwatch through Wi-Fi. An 8GB micro SD card will be used as storage as it can store the installation of the Raspbian OS and reading and writing values from the smartwatch. The PCB (printed circuit board) acts as the structure and support of the system for the sensor connections.

Software

The android application will be developed and maintained and Add-on’s and additional functionality will be incorporated by Baltej. The app is mostly complete in its current state. The only things that are left to work on is Wi-Fi functionality and debugging. The app needs to respond to the hardware in order to display desired information from the application.

The project utilizes a smartphone capable of running Android API 28 or higher. An up to date version of Android Studio was used to build the mobile application. A Raspberry Pi Zero W was implemented with connection between the hardware and application. Updating the Raspbian OS to its newest version was used throughout the project. The mobile application will be used to work alongside the hardware components. Firebase real-time readings is going to be used for communication such as storing user’s, temperature readings, heart rate readings and steps walked readings.

Database

The database will be designed, created and upheld by Jerreh. The database connection is established and connected to the mobile application. Reading and writing from the sensor to the database are also required. The database utilizes user-authentication to allow maximum security and protection for the user’s information. In order to read and write to the different sensors, the user must be registered using an email and password through authentication processing.

Build Instructions

For microcontrollers without an analog-to-digital converter or when you want a higher-precision ADC, the ADS1015 provides 12-bit precision at 3300 samples/second over I²C. The chip can be configured as 4 single-ended input channels, or two differential channels. It even includes a programmable gain amplifier, up to x16, to help boost up smaller single/differential signals to the full range. This ADC because it can run from 2V to 5V power/logic,

Figure 1 – ADS1015 12-Bit ADC

can measure a large range of signals and its super easy to use.

The ADXL345 is a low-power, 3-axis accelerometer with both I2C and SPI interfaces. The sensor is soldered on to a custom-designed PCB that allows for plug-and-play operation with my chosen development platform – the Raspberry Pi 3.

This sensor works best with objects that are good emitters of infrared radiation. Black anodized aluminum or cast iron are pretty good emitters. Polished metal surfaces are very poor emitters, but can usually be turned into a good emitter with a bit of flat-black paint. The TMP006 works with 3v to 5v, so it can be used with most microcontrollers without the need for a level

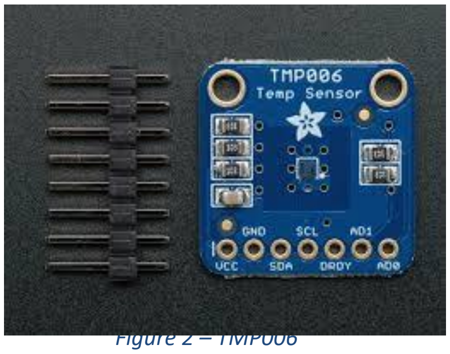


Figure 2 – TMP006

shifter. It connects via the i2c bus and is addressable so you can have up to 8 TMP006 sensors on the same bus.

The Pulse Sensor Amped is a plug-and-play heart-rate sensor for Arduino. It can be used by students, artists, athletes, makers, and game & mobile developers who want to easily incorporate live heart-rate data into their projects. It essentially combines a simple optical heart rate sensor with amplification and noise cancellation circuitry making it fast and easy to get reliable pulse readings. Also, the Pulse Sensor is easy to use just place your finger tip and plug it into your 3* or *5 Volt input.

Figure 4 – Pulse Sensor

Introduction

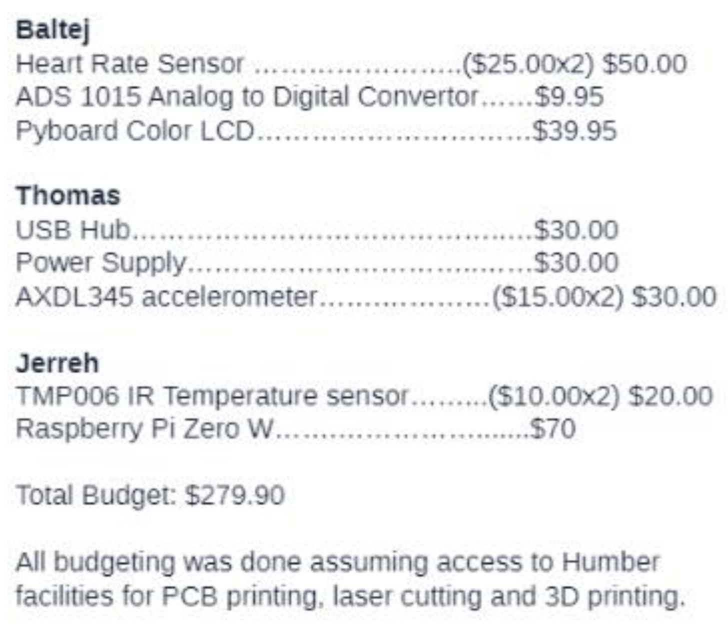
Our smartwatch is primarily a health-focused device and our choices of sensors reflect that. The ADXL345 accelerometer allows for user movement tracking. The ADS1015 pulse sensor will let the device monitor the user's pulse to help gauge active calories burned during exercise. The TMP006 temperature sensor allows the device to monitor the user's body temperature and send alerts based on body temperature data. All data will be processed with a dedicated app on a connected Android device, and displayed in-depth within the app. The display of the watch itself will show the user quick tidbits of information over time, and allow for quick changes and modifications to watch functionality.

Repository

The repositories consist of all the files that were made throughout the production of the smartwatch. One repository strictly consists of android code to replicate the smartwatch MyBeat mobile application. Inside the documentation repository are folders that categorize how each of the work was handled. From the previous semester CENG319, a schedule was predetermined on how the work was going to be laid out. This work was accomplished by project and excel 2016. The schedule, report, proposal, and budget were being updated as we came upon obstacles that interfered within the plans. In the repository are the 5 status reports that were to be reported during this semester for the Capstone Project. The updates were rotated by students A, B, and C but uploaded to a main repository by student b. These reports were to verify if the work is being done properly and if the project is on task as required. There are 5 folders in the repository that consists of 5 different parts, the electronics, the firmware, the mechanical, and images. The electronics folder handles the breadboard and the pcb. These were made using the Fritzing program, the files were merged into one zip file containing Gerber files to create the custom PCB. The firmware folder handles the code for all the different sensors and the combined file with all three sensors working together. All the code of this project is available to download directly as this project is completely open source for anyone to use. The mechanical folder handles the smartwatch case which was made using CorelDRAW to create the laser cut case and also Sprint3D with was used to create the 3D part of the case. The images folder handles the pictures which consist of all the progress made throughout the stage of the smartwatch project. From the skeleton of the hardware, sensors, raspberry pi, and PCB are all contained within the folder. unfortunately, due to circumstances we were unable to provide screenshots of code working with the sensor all together, this would have been with the temperature sensor, heart rate sensor and steps walked sensor all being viewed through a screen. Additionally, there are also pictures of the different views of the hardware and PCB which were taken to display accurate visuals on how the pin sockets were soldered.

Budget for Materials Required

The required materials and budget for this project can be found in the documentation folder in the repository.



Baltej
Heart Rate Sensor ……………………..($25.00x2) $50.00
ADS 1015 Analog to Digital Convertor……$9.95
Pyboard Color LCD…………………………$39.95

Thomas
USB Hub……………………………………..$30.00
Power Supply…………………………….….$30.00
AXDL345 accelerometer………………..($15.00x2) $30.00

Jerreh
TMP006 IR Temperature sensor………($10.00x2) $20.00
Raspberry Pi Zero W…………………........$70

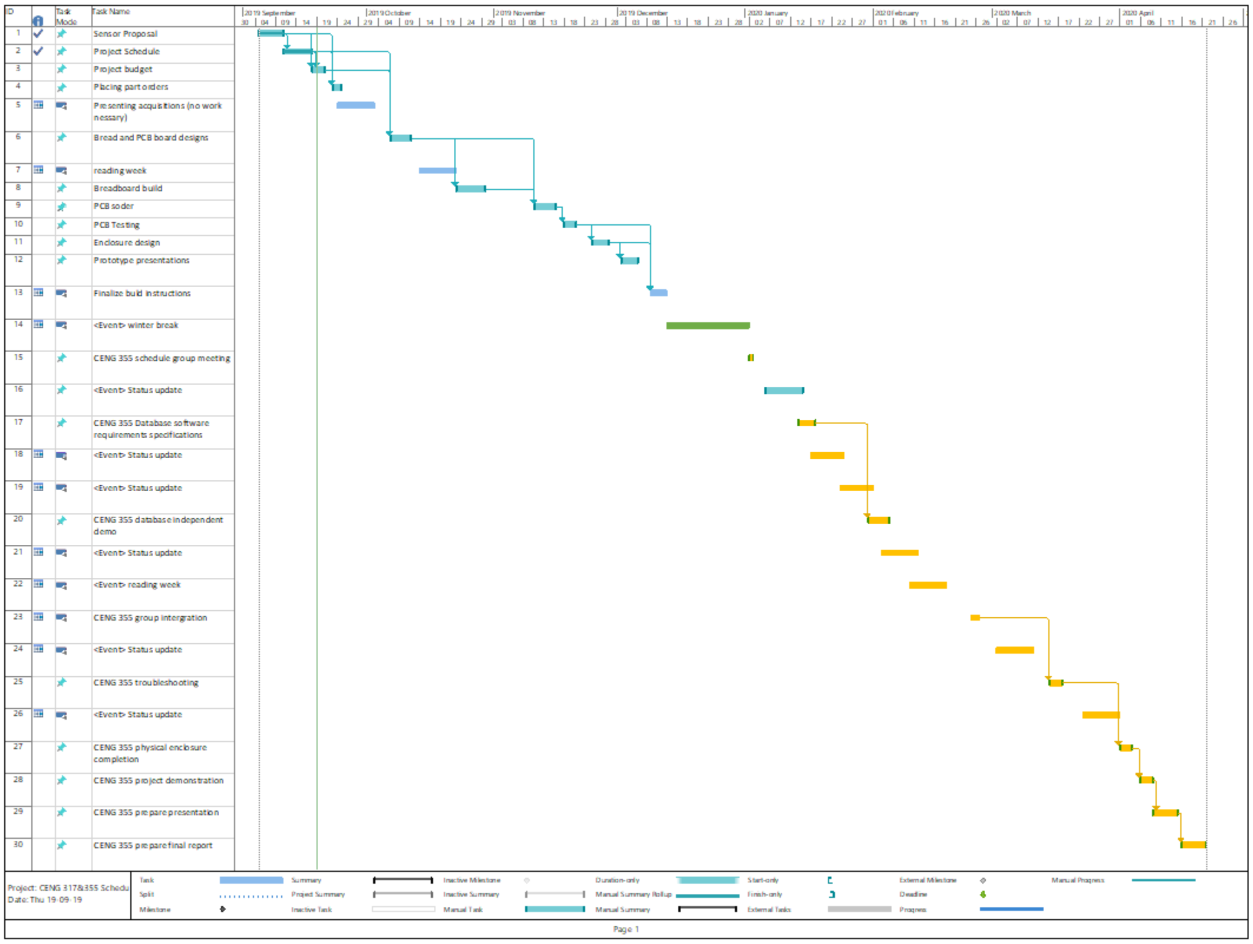
Total Budget: $279.90

All budgeting was done assuming access to Humber facilities for PCB printing, laser cutting and 3D printing.

Figure 5 – Budget

Time Schedule

Realistically, this project should take around 3-4 weeks to complete if all materials and facilities are available to you. The materials themselves might take a week to arrive due to shipping, but the actual process of assembling and programming should not take longer than a week if proper time is given. A couple of hours each day can be dedicated towards the different aspects of the project to make time usage more efficient and effective. For us, this project took around 2 whole semesters (8 months) to finish along with an average work time of around 2.5 hours a week. Here is the time schedule we followed:



ID		Task Mode	Task Name
1	✓		Sensor Proposal
2	✓		Project Schedule
3			Project budget
4			Placing part orders
5			Presenting acquisitions (no work nessary)
6			Bread and PCB board designs
7			reading week
8			Breadboard build
9			PCB soder
10			PCB Testing
11			Enclosure design
12			Prototype presentations
13			Finalize build instructions
14			<Event> winter break
15			CENG 355 schedule group meeting
16			<Event> Status update
17			CENG 355 Database software requirements specifications
18			<Event> Status update
19			<Event> Status update
20			CENG 355 database independent demo
21			<Event> Status update
22			<Event> reading week
23			CENG 355 group intergration
24			<Event> Status update
25			CENG 355 troubleshooting
26			<Event> Status update
27			CENG 355 physical enclosure completion
28			CENG 355 project demonstration
29			CENG 355 prepare presentation
30			CENG 355 prepare final report

Project: CENG 317&355 Schedu
Date: Thu 19-09-19

Page 1

Figure 3 - Schedule

Assembly of Pi

These steps will cover how to set up the Raspberry Pi Zero W properly so that you have the ability to log in and test your sensors capabilities.

1. Format an SD card with a minimum of 8GB to be used for the OS of the Pi. You can use the following link to download a SD card formatting software: https://www.sdcard.org/downloads/formatter_4/index.html

2. Download and unzip the latest version of the OS for the Raspberry Pi to your SD card. Download NOOBS in the link as that will ensure that you will have almost everything required when starting: https://www.raspberrypi.org/downloads/noobs/

3. Once the image is on the SD card, remove it from the pc and insert it in the Pi. Now plug in a separate monitor, mouse, keyboard, HDMI, Ethernet cable, and power supply to the Pi in their corresponding ports. The Pi turns on automatically when the power is plugged in.

4. Upon the boot up session, select Raspbian as the operating system for the Pi and follow the instructions as they appear. You may also change the keyboard layout on the bottom during initial boot. The US layout is highly recommended.

5. Once installation is completed, you should be brought to the desktop. Connect yourself to either Wifi or wired connection in order to perform the next few steps.

6. Now it is time to set up a VNC connection so that you can access your Pi on any computer screen. From the Start Menu, go -> Preferences->Raspberry Pi Configuration->Interfaces, then set VNC to Enabled. Now on the desktop in the top

right corner, you should see a VNC logo. When you click it, you should see an IP address for your Pi which will be used to connect it via the VNC software. Download the software on any computer you wish to communicate with the Pi: https://www.realvnc.com/en/connect/download/vnc/

7. Once the software is installed, connect the Ethernet cable from the Pi to your computer of choice to have a direct connection. Now you can simply input the same address you found in the Pi in the VNC software and it should connect.

8. To turn off the Pi, type sudo power down in the terminal.

If you are still unsure or struggling with a part in particular, this video provides a step by step explanation for everything required:
https://www.youtube.com/watch?v=xBlxuf_LSCM

Wiring

Before wiring each sensor to the breadboard, it is important to solder the pins that come included to the sensors corresponding pin layouts. Additionally, make sure to cut the excess pins that come included that will not be required for each sensor.

When soldering, make sure you have safety glasses equipped along with having proper ventilation that contains an extractor arm for the fumes. A soldering toolkit is also required which is available in most labs. Here is a great soldering tutorial to help with those unsure.

https://www.youtube.com/watch?v=3230nCz3XQA

You can wire the sensors to the Raspberry Pi using the following charts:

TMP006 Temperature.

Device Pin	Pi
1 (3.3v)	[3.3v]
6 (GND)	[GND]
3 (SDA)	[GPIO 2]
5 (SCI)	[GPIO 3]

ADXL345.

1 (3.3v)	[3.3v]
6 (GND)	[GND]
3 (SDA)	[GPIO 2]
5 (SCI)	[GPIO 3]

ADS1015.

1 (3.3v)	[3.3v]
6 (GND)	[GND]
3 (SDA)	[GPIO 2]
5 (SCI)	[GPIO 3]

SEN-11754.

Middle pin (3.3v)	[3.3v]
Left pin (GND)	[GND]
Right pin (Signal of ADS1015)	[A0]

This is the layout for the pins of the Raspberry pi for guidance on where certain pins are located:

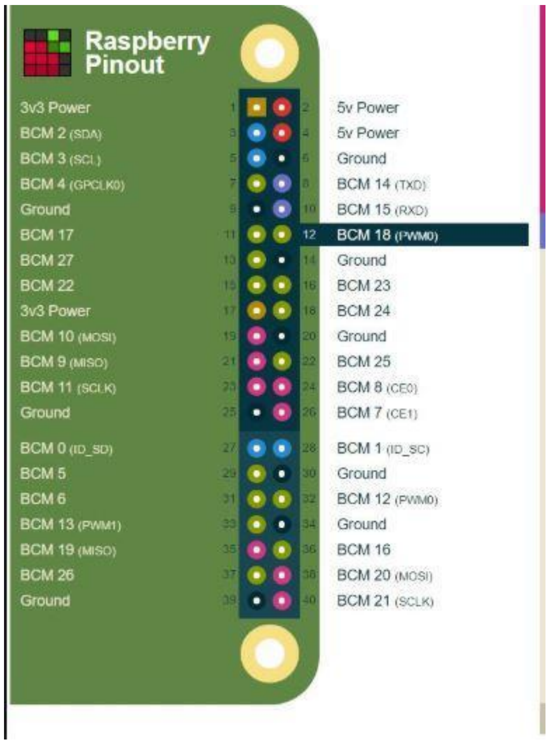


Figure 4 - Raspberry Pi Zero W Pinouts

PCB Design Files

PCB implementation required multiple connections via copper pathways catered to three sensors across both surfaces of the PCB while avoiding any crossovers, this allows the electrical currents demanded of the sensors to reach where they need to go. These sensors being the aforementioned TMP006, ADXL345 and MCP3008. The beginning stages of the PCB involved creating a breadboard and schematic to map out what connections are necessary and to illustrate how it should work to readers. In order to develop the PCB design files, the application Fritzing is required along with the Corresponding sensors file which must be added to the application under MyParts. The files can be located here: https://github.com/adafruit/Fritzing-Library/tree/master/parts Once the Sensor is added to parts, you can create a fritzing diagram for the wiring of the pi and sensor. It should look similar to this.

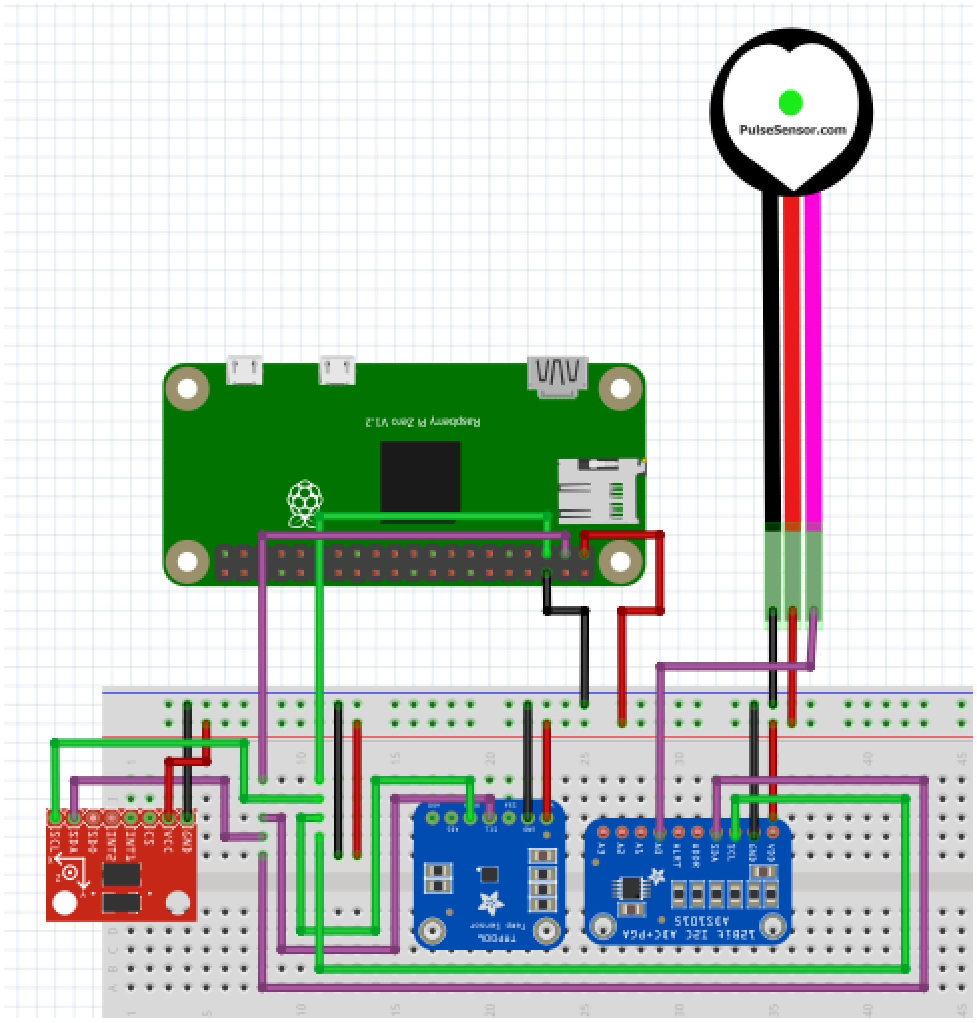


Figure 5 - Breadboard Layout

From here you can create the PCB design from the wiring you just designed. The PCB layout should look similar to this.

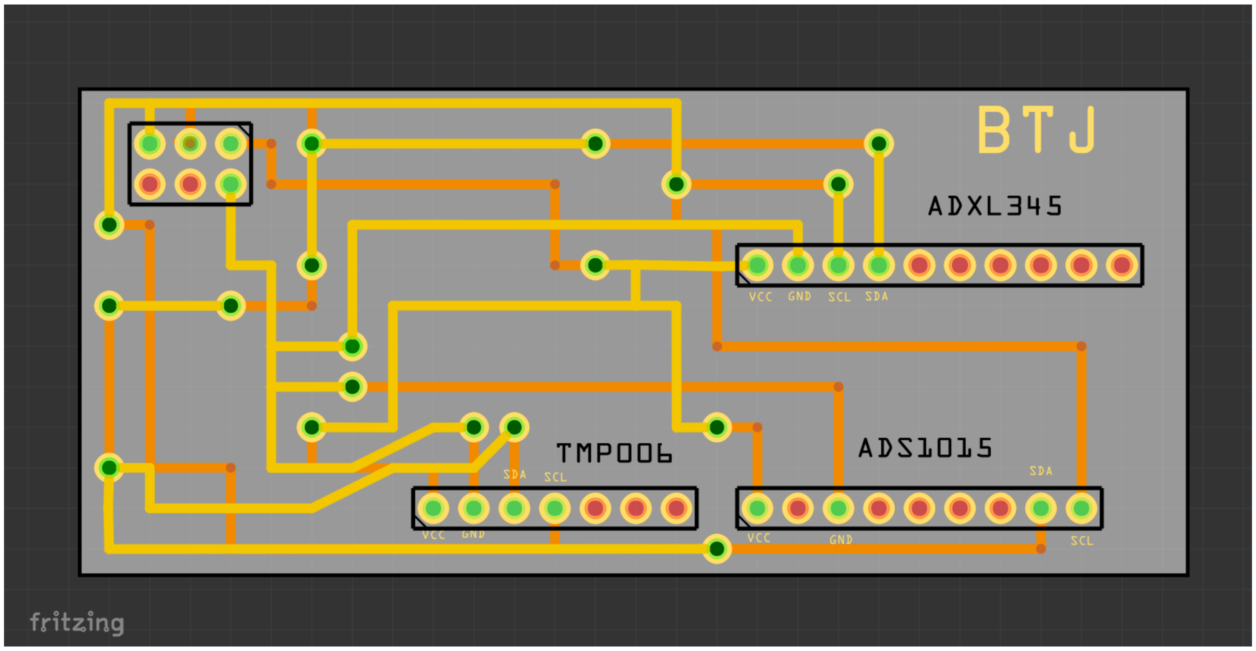


Figure 6 - PCB Design

The copper layers were designed avoiding the connections with each other with the use of vias 0.9 mm diameter holes that allow a copper layer to connect to the other side of a PCB when soldered. the vias were especially useful given the placement of the sensors with the analog to digital converter for the ADXL345 with which was a priority as due to the distance between the raspberry pi zero and the PCB board being so short and the top surface having the other two sensors that would result in a cumbersome placement of parts, the sensor needed to be soldered directly into the board. With these now ready, you can put together your Gerber files and create your PCB using a laser cutter machine.

The PCB uses a combination of sockets purchased from amazon whereas the PCB itself was created by Humber college’s in-house prototype lab, this lab did the work of ensuring the substrate, solder mask layer and silkscreens are properly affixed to the board as well as ensure the copper layers and specified dimensions matched the Gerber files, which enclosed as the dimensions and specifications requested. The Gerber files are located in the repository:

https://github.com/Breezydust/SmartWatch/tree/master/Electronics

PCB Soldering

Using the same rules as when soldering the Sensors, solder pieces of wire in between the vias on the PCB board. Once that's done, solder the 20-pin socket to the PCB board to the corresponding holes for where the pi would connect. For the remaining pin sockets, you have, solder in the respective headers for each sensor in the appropriate locations. Your final board should look similar to this.

Top view:

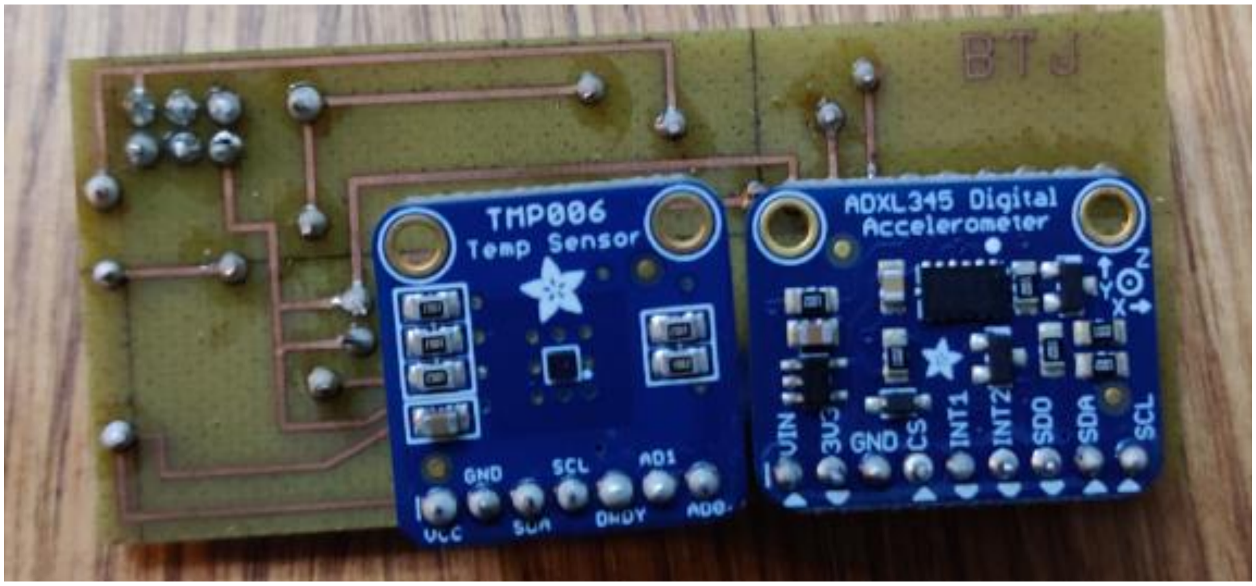


Figure 7 - PCB Top View

Bottom view:

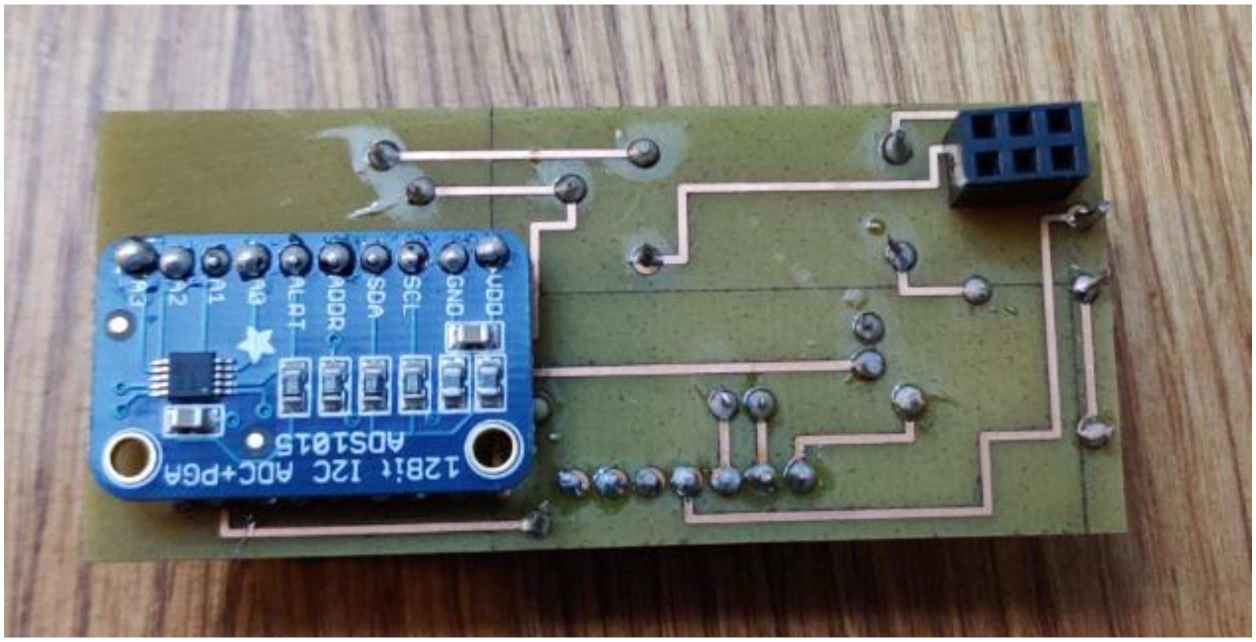


Figure 8 - PCB Bottom View

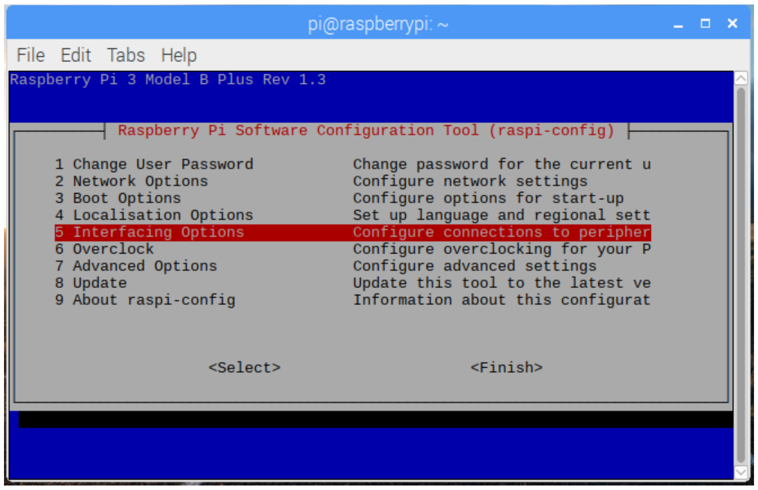
Power Up

In this section, we will now see if everything works, this works on whether you have soldered your PCB or you normally wired it onto your circuit board. Once connected boot up the Raspberry Pi, open the terminal window and follow these steps:

1. This will bring you into the configuration tool

sudo rasp-config

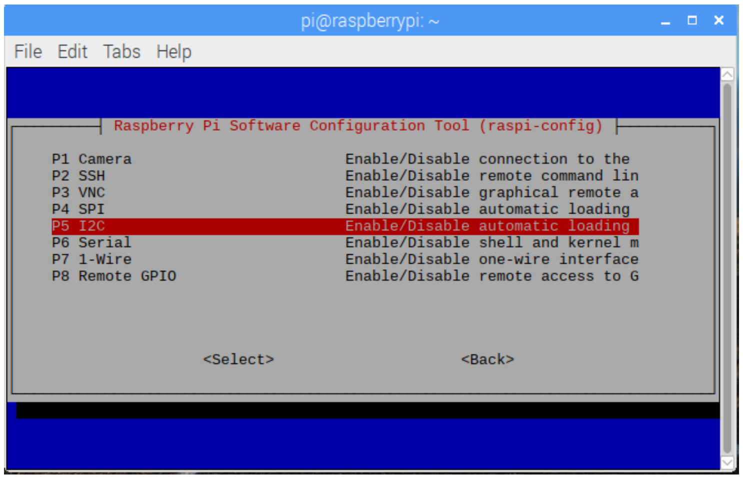
1. Use your arrows keys to go down and select "Interfacing Options"



```
Raspberry Pi 3 Model B Plus Rev 1.3

         Raspberry Pi Software Configuration Tool (raspi-config)

     1 Change User Password          Change password for the current u
     2 Network Options               Configure network settings
     3 Boot Options                  Configure options for start-up
     4 Localisation Options          Set up language and regional sett
     5 Interfacing Options           Configure connections to peripher
     6 Overclock                     Configure overclocking for your P
     7 Advanced Options              Configure advanced settings
     8 Update                        Update this tool to the latest ve
     9 About raspi-config            Information about this configurat

                  <Select>                     <Finish>
```

Figure 9 - Interfacing Menu

2. Select I2C and submit yes. It should display ARM I2C is enabled.

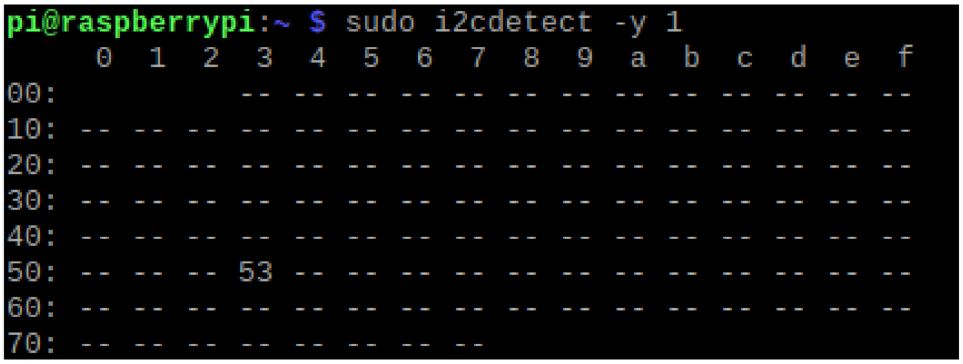


```
Raspberry Pi Software Configuration Tool (raspi-config)

P1 Camera                        Enable/Disable connection to the
P2 SSH                           Enable/Disable remote command lin
P3 VNC                           Enable/Disable graphical remote a
P4 SPI                           Enable/Disable automatic loading
P5 I2C                           Enable/Disable automatic loading
P6 Serial                        Enable/Disable shell and kernel m
P7 1-Wire                        Enable/Disable one-wire interface
P8 Remote GPIO                   Enable/Disable remote access to G

              <Select>                     <Back>
```

Figure 10 - I2C Option

3. Exit by selecting the finish option. By using the command below, it should your address for the sensors being (0x40) and (0x70).

sudo i2cdetect -y 1

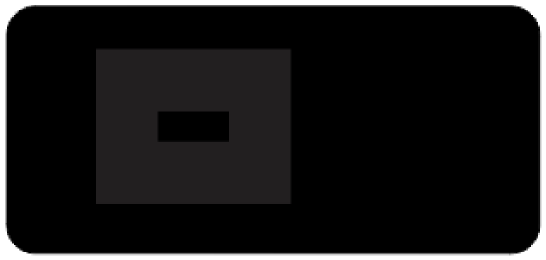


```
pi@raspberrypi:~ $ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- 53 -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Figure 11 - I2C Addresses

Case Design

The case was created using CorelDRAW software. The file can be located in the repository linked above. This is the top of the case side which was carefully measured to ensure everything fit in to the corresponding locations with the design of the 3D portion of the case. It should look like this:



The other part of the case was created using the Sprint3D software. The file can be located in the repository linked above. This is the side and bottom of the case side which was carefully measured to ensure everything fit in to the corresponding locations with the design of the laser cut portion of the case. It should look like this:

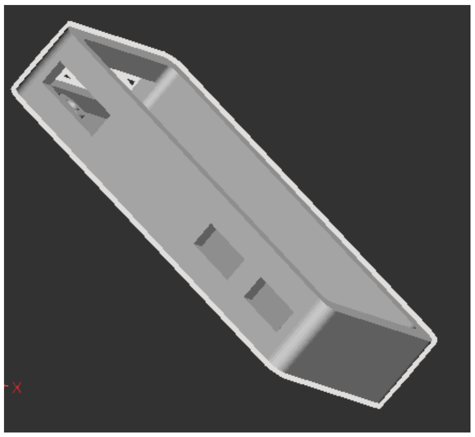


Figure 12- Case Design

For the case, we decided to mount the display screen, and the pulse sensor on top of the case as this project is a watch and in order to give our project more appeal in terms of its design rather than simply being a box that contains 3 sensors within. The temperature sensor was mounted on the bottom of the PCB so that it can receive more accurate readings. The step counter and the ADC are mounted on top of the PCB. In terms of port accessibility, all the ports are open as the others are necessary for the project to function.

Installing Circuit Python

For this project a lot of code will be used with Circuit Python so here is the installation process for it.

The following tests were experimented using Adafruit:

1. Run the update commands for the Raspberry Pi.

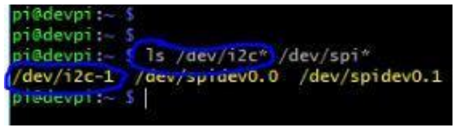
   sudo apt-get upgrade

2. When done installing, run the command line for the python tools

   sudo pip3 install --upgrade setup tools

3. Verify you have I2C Enabled

   ls /dev/i2c*



```
pi@devpi:~ $
pi@devpi:~ $
pi@devpi:~ $ ls /dev/i2c* /dev/spi*
/dev/i2c-1  /dev/spidev0.0  /dev/spidev0.1
pi@devpi:~ $ |
```

Figure 13 – Verification

4. Begin to install the Python Libraries

   pip3 install RPI.GPIO

5. Use the following command to install adafruit-blinka:

   pip3 install adafruit-blinka

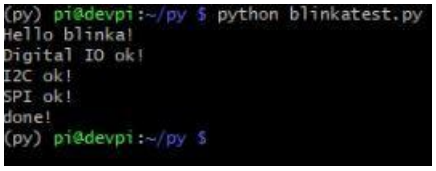
To test if Python works, open python in the Raspberry Pi (it should be installed at this point), and write an example file to sample output.

```
import board

import digitalio

import busio


print("Hello blinka!")


# Try to great a Digital input

pin = digitalio.DigitalInOut(board.D4)

print("Digital IO ok!")


# Try to create an I2C device

i2c = busio.I2C(board.SCL, board.SDA)

print("I2C ok!")


# Try to create an SPI device

spi = busio.SPI(board.SCLK, board.MOSI, board.MISO)

print("SPI ok!")


print("done!")
```

Save it, then run it on the command line by typing

```
python3 blinkatest.py
```

The following should be seen



```
(py) pi@devpi:~/py $ python blinkatest.py
Hello blinka!
Digital IO ok!
I2C ok!
SPI ok!
done!
(py) pi@devpi:~/py $
```

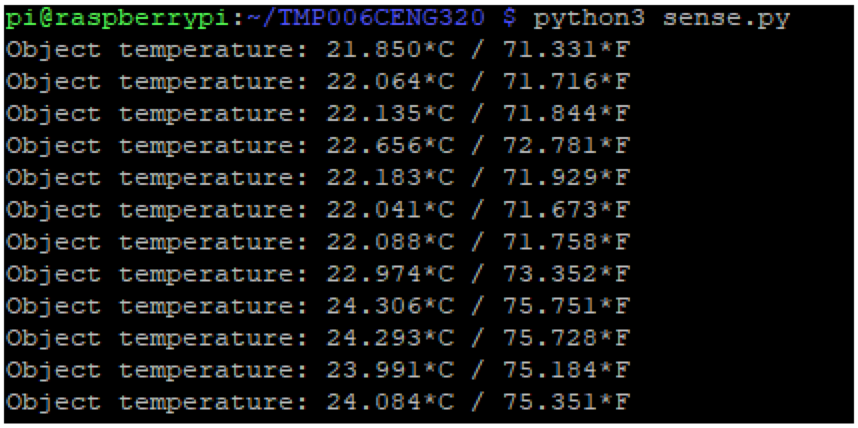
Figure 14 - Blinatest Output

Code for Sensors

TMP006 Temperature/Humidity Sensor:

```
import time

import board

import busio

import adafruit_tmp006


# Define a function to convert celsius to fahrenheit.

def c_to_f(c):

return c * 9.0 / 5.0 + 32.0




# Create library object using our Bus I2C port

i2c = busio.I2C(board.SCL, board.SDA)

sensor = adafruit_tmp006.TMP006(i2c)


# Initialize communication with the sensor, using the default 16 samples per
conversion.

# This is the best accuracy but a little slower at reacting to changes.
```

```
        # The first sample will be meaningless

        while True:

        obj_temp = sensor.temperature

        print(

        "Object temperature: {0:0.3F}*C / {1:0.3F}*F".format(obj_temp,
        c_to_f(obj_temp))

        )

    time.sleep(5.0)
```



```
pi@raspberrypi:~/TMP006CENG320 $ python3 sense.py
Object temperature: 21.850*C / 71.331*F
Object temperature: 22.064*C / 71.716*F
Object temperature: 22.135*C / 71.844*F
Object temperature: 22.656*C / 72.781*F
Object temperature: 22.183*C / 71.929*F
Object temperature: 22.041*C / 71.673*F
Object temperature: 22.088*C / 71.758*F
Object temperature: 22.974*C / 73.352*F
Object temperature: 24.306*C / 75.751*F
Object temperature: 24.293*C / 75.728*F
Object temperature: 23.991*C / 75.184*F
Object temperature: 24.084*C / 75.351*F
```

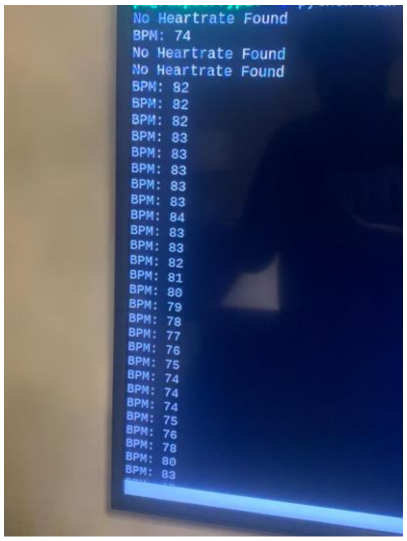
Figure 15 - Temperature Readings

ADS1015 ADC + Heart Rate Sensor Code**:**

```
 Import
 time
           # Import the ADS1x15 module.
           import Adafruit_ADS1x15


           if __name__ == '__main__':
```

```
    adc = Adafruit_ADS1x15.ADS1015()
    # initialization
    GAIN = 2/3
    curState = 0
    thresh = 525  # mid point in the waveform
    P = 512
    T = 512
    stateChanged = 0
    sampleCounter = 0
    lastBeatTime = 0
    firstBeat = True
    secondBeat = False
    Pulse = False
    IBI = 600
    rate = [0]*10
    amp = 100

    lastTime = int(time.time()*1000)

    # Main loop. use Ctrl-c to stop the code
    while True:
        # read from the ADC
        Signal = adc.read_adc(0, gain=GAIN)   #TODO: Select the correct
ADC channel. I have selected A0 here
        curTime = int(time.time()*1000)

        sampleCounter += curTime - lastTime;      #                  # keep track of
the time in mS with this variable
        lastTime = curTime
        N = sampleCounter - lastBeatTime;     #  # monitor the time since the
last beat to avoid noise
        #print N, Signal, curTime, sampleCounter, lastBeatTime

        ##  find the peak and trough of the pulse wave
        if Signal < thresh and N > (IBI/5.0)*3.0 :  #       # avoid dichrotic noise
by waiting 3/5 of last IBI
            if Signal < T :                        # T is the trough
              T = Signal;                        # keep track of lowest point in pulse
wave

        if Signal > thresh and  Signal > P:          # thresh condition helps avoid
noise
            P = Signal;                            # P is the peak
```

```
                                        # keep track of highest point in pulse wave

        #  NOW IT'S TIME TO LOOK FOR THE HEART BEAT
        # signal surges up in value every time there is a pulse
       if N > 250 :                              # avoid high frequency noise
         if  (Signal > thresh) and  (Pulse == False) and  (N > (IBI/5.0)*3.0)  :
           Pulse = True;                            # set the Pulse flag when we think
there is a pulse
           IBI = sampleCounter - lastBeatTime;        # measure time between
beats in mS
           lastBeatTime = sampleCounter;              # keep track of time for
next pulse

           if secondBeat :                       # if this is the second beat, if
secondBeat == TRUE
             secondBeat = False;                  # clear secondBeat flag
             for i in range(0,10):           # seed the running total to get a
realisitic BPM at startup
               rate[i] = IBI;

           if firstBeat :                    # if it's the first time we found a beat, if
firstBeat == TRUE
             firstBeat = False;                 # clear firstBeat flag
             secondBeat = True;                   # set the second beat flag
             continue                          # IBI value is unreliable so discard it


           # keep a running total of the last 10 IBI values
           runningTotal = 0;                  # clear the runningTotal variable

           for i in range(0,9):               # shift data in the rate array
             rate[i] = rate[i+1];               # and drop the oldest IBI value
             runningTotal += rate[i];             # add up the 9 oldest IBI values

           rate[9] = IBI;                      # add the latest IBI to the rate array
           runningTotal += rate[9];                # add the latest IBI to
runningTotal
           runningTotal /= 10;                  # average the last 10 IBI values
           BPM = 60000/runningTotal;               # how many beats can fit into
a minute? that's BPM!
           print 'BPM: {}'.format(BPM)
       if Signal < thresh and Pulse == True :   # when the values are going down, the beat is over
         Pulse = False;                   # reset the Pulse flag so we can do it again
         amp = P - T;                     # get amplitude of the pulse wave
         thresh = amp/2 + T;                  # set thresh at 50% of the amplitude
         P = thresh;                      # reset these for next time
         T = thresh;

       if N > 2500 :                        # if 2.5 seconds go by without a beat
         thresh = 512;                        # set thresh default
         P = 512;                           # set P default
         T = 512;                           # set T default
         lastBeatTime = sampleCounter;          # bring the lastBeatTime up to date
         firstBeat = True;                    # set these to avoid noise
         secondBeat = False;                     # when we get the heartbeat back
         print "no beats found"

       time.sleep(0.005)
```



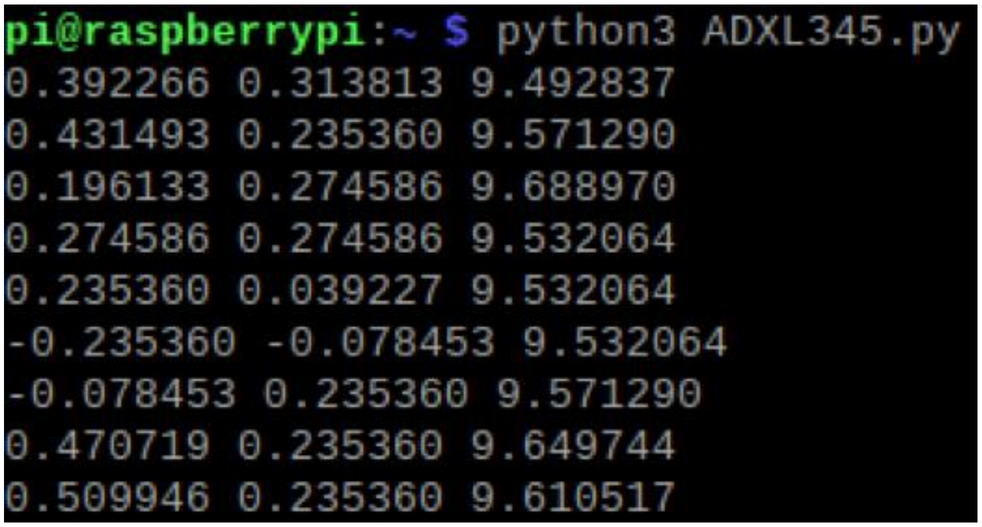
```
No Heartrate Found
BPM: 74
No Heartrate Found
No Heartrate Found
BPM: 82
BPM: 82
BPM: 82
BPM: 83
BPM: 83
BPM: 83
BPM: 83
BPM: 83
BPM: 84
BPM: 83
BPM: 83
BPM: 82
BPM: 81
BPM: 80
BPM: 79
BPM: 78
BPM: 77
BPM: 76
BPM: 75
BPM: 74
BPM: 74
BPM: 74
BPM: 75
BPM: 76
BPM: 78
BPM: 80
BPM: 83
```

Figure 22 – ADS1015

ADXL345:

```
#LibraryImports
                        import time
                        import board
                        import busio
                        import adafruit_adxl34x

                        #Sensor initialization
                        i2c = busio.I2C(board.SCL, board.SDA)
                        sensor = adafruit_adxl34x.ADXL345(i2c)


                        mode = input("Simple or Advanced Measurements? ")


                        if mode.lower() == "advanced": #enable more sensor functionality
                                print("Enabling advanced measurements")
                                sensor.enable_freefall_detection(threshold=10, time=25)
                        #enable freefall detection with a threshold of 625mg of force
                        (10*62.5mg) and a time scale of less than 125ms (25x5)
                                sensor.enable_motion_detection(threshold=18) #enable
                        motion detection with a threshold of 1125mg (18x62.5mg) of force
                                sensor.enable_tap_detection(tap_count=1, threshold=20,
                        duration=50, latency=20, window=0) #enable tap detection.
                        detects single taps, tap intensity threshold is 1250mg, minimum
                        duration is 50ms, latency to register single tap is 20ms
                        else:
                                print("Using only simple measurements")

                        while True:
                                print("X: %f Y: %f Z: %f"%sensor.acceleration) #print X, Y
                        and Z coordinates and any acceleration
                                if bool(sensor.events["freefall"]): print("Sensor dropped!")
                                #advanced funtionality - print sensor dropped if freefall is
                        detected
                                if bool(sensor.events["tap"]): print("Sensor tapped!")
                                #advanced functionality - print sensor tapped if tap is
                        detected
                                if bool(sensor.events["motion"]): print("Sensor moved!")
                                #advanced functionality - print sensor moved if movement
                        is detected
```

```
time.sleep(1.0)
```

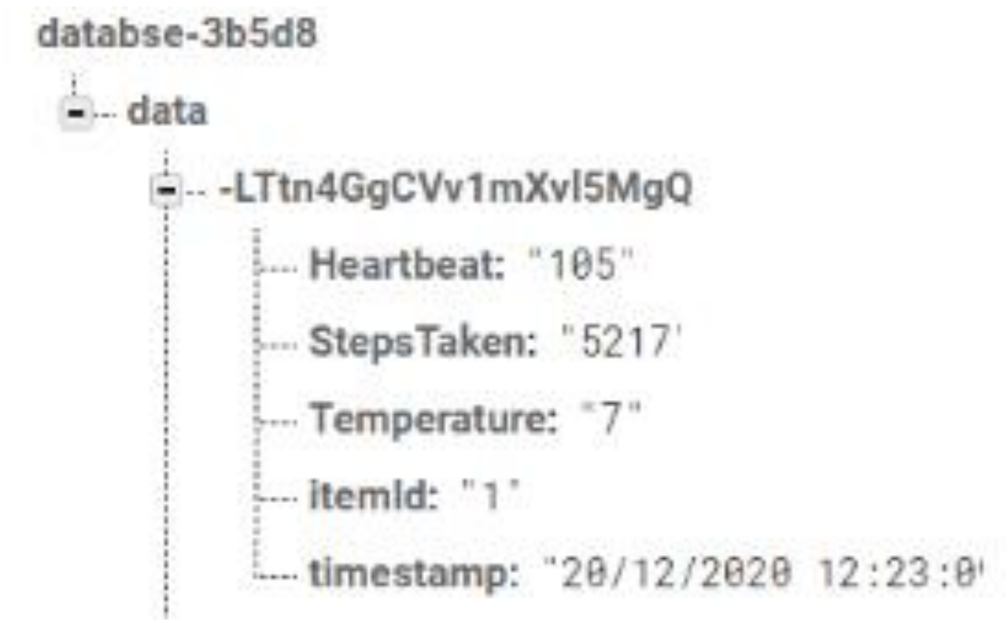


```
pi@raspberrypi:~ $ python3 ADXL345.py
0.392266 0.313813 9.492837
0.431493 0.235360 9.571290
0.196133 0.274586 9.688970
0.274586 0.274586 9.532064
0.235360 0.039227 9.532064
-0.235360 -0.078453 9.532064
-0.078453 0.235360 9.571290
0.470719 0.235360 9.649744
0.509946 0.235360 9.610517
```

Figure 163 – ADXL345

Database Design

The database connection is established and connected to the mobile application. Reading and writing from the sensors to the database are also required. The database utilizes user-authentication to allow maximum security and protection for the user's information. In order to read and write temperature, the user must be registered using a username and password through authentication processing. The database connection works in conjunction with firebase.



```
databse-3b5d8
 ⊟-- data
      ⊟-- -LTtn4GgCVv1mXvl5MgQ
           |--- Heartbeat: "105"
           |--- StepsTaken: "5217'
           |--- Temperature: "7"
           |--- itemId: "1"
           └--- timestamp: "20/12/2020 12:23:0'
```

Figure 24 - Database Layout

The first structure (Data) gets the readings from all the users. The second data structure (username) is the users Id that is currently logged in with their saved sensor reading and a timestamp.

Mobile Application

The mobile application upon startup will provide the login screen. Through user authentication, the application will transfer to the main menu. We designed the user interface to be clean and simple so that it would feel inviting for the user rather than being complex and intimidating. Having the 5 main features of the app (History, Timer, Steps Walked, temperature, and heartrate) on the homepage via tab fragments would make it quick and easy for the user to traverse without having to flip through various menus and options. The data that the app uses such as the stopwatch count would be provided by user input so that they could choose whatever they desire. The mobile app also includes a section where it connects to the firebase where it has the sensor readings. In terms of hardware connectivity, the user will be able to select which sensor they want to have working at the moment from the app and that information will then be sent to the database. The sensor data will then be sent to the hardware component, which will ultimately trigger the hardware to go off.

Test Cases

Authentication:

This is important as it gives the user access to the database. The first thing they will see is if they can login, for new users they would have to register. If the user attempts to login without registering, a message will appear noting that the account is not registered. The register page consists of an email and password. When the user inputs an incorrect email address such as @email.com, it would not work due to the authentication recognizing that this is not a proper email address. Therefore, the user would have to use either Gmail, Hotmail, Yahoo, etc. Once everything is met, the account is now registered and saved onto the firebase authentication. They can now log in, if any mistakes are made a warning will appear prompting the user that there is a mistake. A successful login message will appear, if they meet all the necessary credentials and they will move on the to the home screen.

Firebase Read/Write:

In order to read and write to the database, the user first has to be logged in. If the user is not logged in, they will not be able to read or write to the database. The database is structured so that it will always use the UID as the primary key and then the 3 sensors and timestamp under it. So, if a new user saves a temperature, a new structure of their UID, temperature readings, and timestamp will be displayed on the database. This in turn can always have accurate reads of timestamp and temperature for different selected users and different selected sensors.

Android Components

Many android components and libraries are used during development of the app, each with their own specific purpose.

Major methods used for general overall use:

- **Intent (this, MainActivity.class)** - this method is often called when switching between screens (Often used with buttons or TextView links)
- **findViewById (int id)** - this method is used frequently to locate and interact with views found from layout resource files that are attached to the current activity
- **toastmakeTest(applicationContext, text, duration) -** this method is used often to send feedback to the user when they do a certain task like logging in.
- **setOnClickListener(new View.OnClickListener)** - When a button is placed this allows the button to be set. In doing so it becomes it creates a function called onClick(View v). As a result, the system allows all executeable code that is under onClick(View) once the button is pressed.
- **onCreate()** - When an activity is first created it comes with onCreate. Everything containing the activity is under the onCreate method. The onCreate method allows things such as create views, bind data to lists, strings, etc.
- **onOptionsItemSelected(MenuItem item)** - An item is hook each time a item in the option menu is selected it will run the item's selected output. MenuItem is the item that was selected and will execute, the item can never be NULL.

Main Methods for all Firebase Activities:

- **ChildEventListener ()** - Listens for any activities to the children.
- **ValueEventListener ()** - Looks for data changes in specific location of the database.
- **onChildAdded ()** - This method allows the android device to pull data from the data structure and read from the database. In turn when a new child is added to the location to which it was added. For example, the temperature readings from Firebase.
- **onChildedChanged ()** - This method will update the data when a child location has changed. Whenever there is a new temperature reading.
- **onChildRemoved ()** - An event when the child is removed. Whenever content is being removing from the database.
- **onChildMoved ()** - This method is used when the location of the child changes. Used to sort data on the database.
- **onDataChange ()** - Similar to onChildedChanged, but will reads the static snapshot of the information at the given path at the timed event.
- **writeData()** - Will write the database with whatever the user is saving. This also includes the current user ID.

Main methods for Fragments:

- **onCreateView()** - similar to onCreate, but allows the view of other associated fragments.

Main methods for user authentication:

- **signInWithEmailAndPassword -** this method is called when the user gains access to their respective account once the proper email and password are entered
- **getCurrentUser() -** this method retrieves the data from the current user logged into the server, thereby accumulating further input they enter and save to the database

Main methods of Stopwatch Fragment:

- **Chronometer** - A class that implements a timer. You can set when the timer should start counting and also change it to a countdown timer.

Hardware Code

ADS1015 Hardware Code:

```
adc = Adafruit_ADS1x15.ADS1015()
# initialization
GAIN = 2/3
curState = 0
thresh = 525  # mid point in the waveform
P = 512
T = 512
stateChanged = 0
sampleCounter = 0
lastBeatTime = 0
firstBeat = True
secondBeat = False
Pulse = False
IBI = 600
rate = [0]*10
amp = 100
```

TMP006 Function Code:

```
import board

import busio

import adafruit_tmp006


i2c = busio.I2C(board.SCL, board.SDA)

sensor = adafruit_tmp006.TMP006(i2c)
```

ADXL345 Function Code:

```
import time
import board
import busio
import adafruit_adxl34x

#Sensor initialization
```

```
i2c = busio.I2C(board.SCL, board.SDA)
sensor = adafruit_adxl34x.ADXL345(i2c)
```

Problems Encountered

During the development of this project, we ran into many problems and roadblocks which halted progression and created extra hurdles for us to overcome. These ranged from database communication errors to hardware malfunctions.

Hardware Difficulties

When it came to designing a suitable PCB for the project, a few issues were encountered. The biggest issue was spacing – due to the small footprint and the size restrictions imposed by the development platform (Raspberry Pi Zero), we found that we didn’t have much room to work with in terms of placing the sensors and having reliable traces through the board. It took a lot of experimenting and modified designs to get one that was good enough for production.

Database Issues

There were quite a few database issues that hindered progress. The most immediate database issues that have shown itself was early in the database development we ran into issues with the History activity of the application these being the problem of crashing the first of which would allow the application return to the login signing out the current user any concurrent crash would have the user prompted with a message explaining application failure there were also some early-stage issue in the form of parsing the JSON when it displays in the list view of the application.

Application Errors

When working with the application itself, there were many instances where a (R.) error code occurred in android studio. This had to do with strings that were created upon making new xml and java pages that had similar or identical ids. We also had errors when we add the firebase auth to the project as android studio had to be updated.

Case Design Problems

Another area during the projects development where several issues occurred was with the cases design. The case was designed using a software tool called Corel Draw also Sprint3D and it was created by Baltej. It was very difficult to use as we had little to no experience using such software which resulted in many attempts when creating said case. The problem stemmed from measurement errors and incorrect placements for the connectors. Every sensor had to be mounted so cutouts for each device was needed to be accurately measured and positioned on the sides. As all the measuring was done by hand, it was prone to human error and resulted in many attempts to get right.

Solution

Hardware Fix

Because of our size restraints (a smartwatch cannot be too large, and we wanted to fit around the specifications of the Raspberry Pi Zero), we knew that simply making the board bigger was not an adequate solution. Instead, we had to rely on multiple board designs and a multi-layered board to get to a working state. Once we were able to utilize both sides of the board using a series of vias, we were successful in creating a working board.

Database Fix

The first issue of crashing was remedied with proper updating of dependencies and changes to the layout of the history activities, to explain in further detail views that used to exist created more problems than they were worth. These views have been removed without compromising visuals. As for the fix for the JSON parse, this was fixed when we exported the JSON and edited it in notepad++.

Application Fix

In order to fix the (R.) issue, we need to navigate to the strings.xml file and check change string ids that were the same. This did not always fix the error initially however; a successful reboot would clear the errors that were still present. We also updated out android studio to allow for the newest dependences.

Case Design Fix

The case design was eventually fixed after a lot of trial and error however the dimension tool in CorelDraw assisted greatly with the final product. The main issue was that after altering certain sides, others became distorted or their respective cutouts became enlarged. This was difficult to notice as the change was minuscule but still important. The dimension tool helped with keeping track of the distance for each respective part so that when one thing was altered, the other sides can be fixed respectfully.

Future Planning

Plans that are to be determined in the future can go a long way. Be it the hardware, or the mobile application. For starters, the alteration of volume can be implemented. Using these variations of volume can further alter the sound to the user's desires. Another plan that was made throughout the project is the effectiveness of the done timer done sound. Having more than 1 timer done in the mobile application will increase the difference and allow for more variety in sound overall. With this feature, teens can set their stopwatch or timers to have their favorite songs playing when they are working out. As we want the different sounds to be at its full functionality, we plan to implement Bluetooth to the watch as Wi-Fi is not always accessible. If someone is on vacation and/or is far away from technology, the Bluetooth feature can allow communication with the mobile application and hardware without the use of internet. This way, the phone and smartwatch can be used almost anywhere without the worry of having to use Wi-Fi via internet. Another plan that we set forth to the future is the effectiveness of the smartwatch. Since our goal is to allow the user to have a activity healthy living style, adding another hardware to force the user to physically be active would make the smartwatch more efficient than ever. As a result of this feature, the consumers will experience a tougher time to turn off the smartwatch which will keep them in a more "active" state. With this plan, the smartwatch will keep on ringing until the goal is achieved. Another plan that can be set into motion is to make it more difficult to access the snooze button. As the user wants to snooze the smartwatch to stop the disruptive noise, we can implement a tougher way for the user to get into the application to ironically stall their goal.

Reproduction of Product

A feature that was planned during the production of the project is to produce multiple ringtones/sounds to emit from the application when the stopwatch or the timer is done. To further enhance this option, having the user to implement their own sound or ringtone can be added to the mobile application. On the other hand, the mobile application will additionally add various volumes such as bass, tone, tempo, etc. To make the mobile and hardware device communicate with each other without internet via Wi-Fi, Bluetooth can be installed and accessed within the mobile application and hardware device. Other features that can be implemented to the project in the future is a option where the user gets notified when the users temperature is to high or to low also we could implement a inbuilt heart rate sensor with constantly measures the pulse and if the pulse is low it notifies the user or an emergency contacts. For example, when the users heart rate or temperature drops to a curtain number the watch displays a warning and if the numbers get to a critical point the watch automatically sends a location and calls the emergency contact or emergency response to the location. Simply adding these various features to the smartwatch will make it better than the watches on the market already.

Conclusion

In Conclusion, the smartwatch is an android based project designed to give users ease of access to anything health related in a simple and clean form factor. The project consists of mainly layers in order to function as designed. It utilizes 3 distinct sensors being the tmp006 temperature sensor, ADS1015 heart rate sensor and the ADXL345 step counter. All sensors communicate with our database in firebase which allow interactivity with the android app. Users have the ability to check their temperature, heartrate, the number of steps walked on a daily bases and many more functions on the app that are related to a healthy active living. They can also view/save local sensor readings all in one condensed form factor.

Bibliography

Adafruit Industries. “ADS1015 12-Bit ADC - 4 Channel with Programmable Gain Amplifier.” *Adafruit Industries Blog RSS*, www.adafruit.com/product/1083.

Adafruit Industries. “Contact-Less Infrared Thermopile Sensor Breakout - TMP006.” *Adafruit Industries Blog RSS*, www.adafruit.com/product/1296.

Adafruit Industries. “ADXL345 - Triple-Axis Accelerometer (-2g/4g/8g/16g) w/ I2C/SPI.” *Adafruit Industries Blog RSS*, www.adafruit.com/product/1231.

Adafruit Industries. “Pyboard Color LCD Skin with Resistive Touch.” *Adafruit Industries Blog RSS*, www.adafruit.com/product/3498.

Google. (n.d.). Read and Write Data on the Web| Firebase Realtime Database | Firebase. Retrieved April 22, 2020, from https://firebase.google.com/docs/database/web/read-and-write

Udayan Kumar. “Heart Rate Monitor Using Raspberry Pi and Pulse Sensor.” *Udayan Kumar*, 17 May 2016, udayankumar.com/2016/05/17/heart-beat-raspberry/.

[3] M. Schickler *et al*., "Using Wearables in the Context of Chronic Disorders: Results of a Pre-Study," *2016 IEEE 29th International Symposium on Computer-Based Medical Systems (CBMS)*, Dublin, 2016, pp. 68-69.

Millennium Circuits Limited. “Printed Circuit Board Manufacturing Process.” *MCL*, Millennium Circuits Limited, 10 Apr. 2020, www.mclpcb.com/pcb-manufacturing-process/.