

Fitness Watch Open Sound Control Interface

Final Project Binder

Team Members:

Leo Gama

Darsh Patel

Daniel Samtani

Garrett Witt

Submission Date:

April 17th, 2022

Table of Contents: (Page numbers to be updated when finished)

1. Updated Final Problem Analysis (3)
 - Objective (3)
 - Requirements (3)
 - Constraints (3)
 - Engineering Standards (4)
2. Updated Final Detailed Design (4)
 - Detailed design documentation (4)
 - Design of Dashboard (5)
 - Design of Heart Rate Monitor (6)
 - Design of Volume/Gesture Control (7)
 - Design of Smart Device Control (8)
 - Specifications (8)
 - Test Cases (9)
3. Build & Test (9)
 - Project status/results (9)
 - Test results and analysis (9)
 - Key Accomplishments (10)
4. Delivery (10)
 - Overall Project Performance (10)
 - Customer Satisfaction (10)
 - Deliverable status (11)
 - Build Documentation (11)
 - Data Dashboard (11)
 - Heart Rate Monitor (13)
 - Volume/Gesture Control (13)
 - Temperature/Smart Switch Control (14)
 - Challenges/Issues (14)
 - Schedule and Cost Status (15)
 - Lessons Learned (15)

Updated Final Problem Analysis

Objective

The objective of this design project was to create a control system built using a fitness watch that could send and receive Open Sound Control messages for use with smart devices and artistic media. This started with choosing a watch that is known by people that could be connected using bluetooth to a computer where sensor data could be loaded and used for control of some external devices. The initial objective was for the external devices to be some sort of art media like video or music where movements and heart rate data could be used to produce feedback of the user wearing the watch. With some setbacks this objective was pushed off and the controls for the watch changed to some smart devices like lights that are controlled by a smart switch. But even with this minor change the main objective stayed the same for the technical design: the watch was connected to a computer and Open Sound Control messages were the intermediary between the watch, the sensor data, and controls of external devices. All of these objectives were carefully moderated and agreed upon with our customer, who happens to be our subject matter expert.

Requirements

Most of the requirements for this project were based upon the initial watch selection and all of our decisions were always based on what watch we chose and what we were able to do with the watch. The requirements that we laid out for the watch was that it had to be commercially available, must have some sort of connectivity whether it be bluetooth, ANT+, Wifi, etc., it needed certain sensors including heart rate, magnetometer, accelerometer, and temperature, the watch had to be programmable and the sensors must be accessible. Some requirements that were preferred, but were not met were that the watch had to be commonly used and had a skin resistance sensor. The watch obviously needed to be fast in its response, in other terms the latency had to be low for real time analysis to view real time data. Some requirements for the design of the control system is that it had to use Open Sound Control for relay of sensor data between watch and computer, as well as to external devices. The watch had to use the sensor data for some sort of control system whether that be an artistic display or control of smart devices using gestures, temperature data, or heart rate data.

Constraints

The constraints of the project came down to the watch that was chosen and this was found out very early on in the design process. Once the watch was chosen based on the requirements although it had everything that met the requirements, the first watch chosen had its

constraints. These being the connectability of the watch using bluetooth to a computer and what language was used for the connection. Not everyone was familiar with the languages used to program the watch so it became a learning curve to figure out how to get stuff done with the watch. Time was a big constraint as when the first watch did not pan out most of the time was spent catching up by learning the new watch and then having to decide what should be worked on and what might need to be scrapped given the allotted time that was left after the first semester.

Engineering Standards

Engineering standards were taken into consideration along every step of the design process with this project. We ensured that proper technical criteria, methods, and processes were followed in the design, development, and testing of the project and its components. Specifically given the large scale amount of software development relating to the project, we were able to follow Agile development lifecycle processes and follow effective engineering standards in our communication and work in order to ensure that project progress was properly executed.

Updated Final Detailed Design

Detailed design documentation

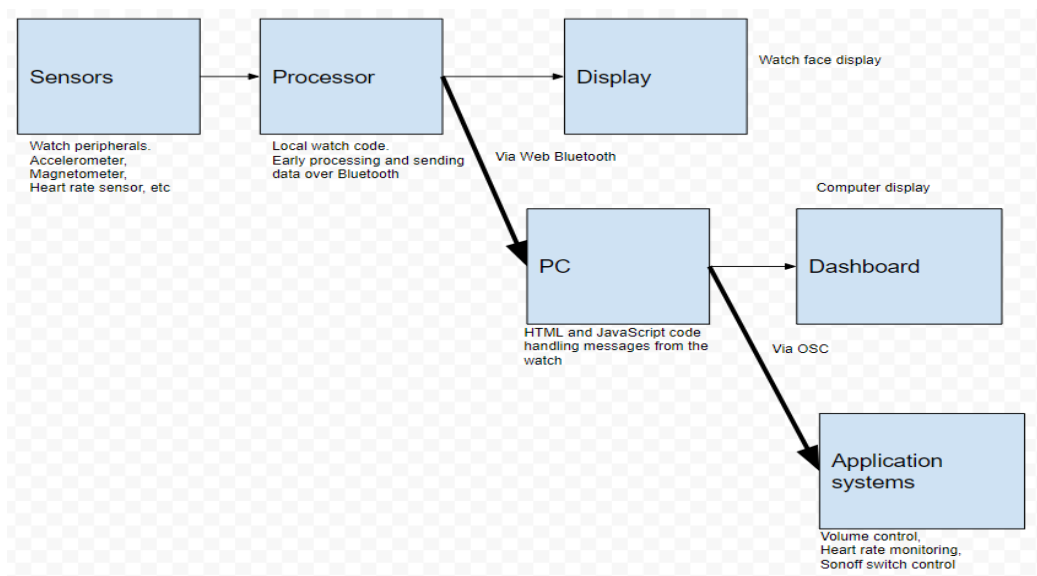


Figure 1: Block diagram devised with customer that defines the system flow design

In Figure 1 you can see the block diagram that was decided upon with the customer/SME that lays out how the system needed to be designed in order to have a control system between a

watch and some external devices. The first step is to load the sensor data to code that is directly loaded onto the watch. For the watch used, the Bangle.js, in order to upload any code onto the watch it had to be connected to a computer over bluetooth and HTML code had to be used to program the watch. This code that was uploaded to the watch was mainly to access the sensor data and also use the peripherals. Once the watch was connected to a computer over bluetooth and sensor data was being taken a UDP server could then be set up so that the use of Open Sound Control messages could be sent over this transport containing the sensor data. This UDP server was set up using NodeJS and some open source code. Once the server is set up and the watch code is written so that whenever sensor data is coming in it is sent using Open Sound Control messages over UDP to a CSV file where the data could be used for signal processing like in the case of the heart rate monitor. For the dashboard the sensor data is stored in local variables, which the React app can then access and publish to the web application that was created. More details about each module can be found below. After the sensor data is stored wherever it needs to be it is then processed for some sort of controls using external devices. The controls are all based on sensor data and each control for different sensors is laid out for each module below.

Design of Dashboard

When beginning to design the dashboard, a springboard was used to ideate and identify the specific attributes necessary for a comprehensive interface system with the Bangle smart watch. We started by first determining that in order to work with the Bangle smart watch in creating an interface, a system that could connect via bluetooth and be hosted via the web would be ideal for portability. To match these initial requirements we settled on developing a web application using the React framework. Using React we were able to use numerous libraries available to the framework and were able to program a fully functioning smart dashboard to connect and communicate with the Bangle watch all using JavaScript, HTML, CSS, and Python.

The dashboard works in having the front-end code (display side and functionality) written in a group of javascript files, additionally the code sent to the Bangle watch in order to communicate and stream data to the dashboard is written in HTML and CSS and sent via an API called Puck.js. Using the data streamed to the dashboard, the dashboard is also able to control external smart devices using a backend which uses API calls in order to trigger the external devices. Further information on this process is described in the design of smart device control section.

In evaluating the latency surrounding our data streaming, after numerous testing using the watch in real-time we were able to determine that a stream rate of 1000 ms or 1 second as ideal in ensuring that the data which is being streamed is easily readable and accurate. In addition to having the data streamed to the dashboard from the Bangle watch via web bluetooth in real-time,

we were also able to store the data into a historical database. The data is available at the bottom of each tile in the dashboard, sectioned off by each sensor's data type.

The final step we took was to host the web application on a web page so that it was available not just locally on a single or select few computers but to anyone with access to the world wide web. As later defined in the build documentation, the application is currently hosted on a website linked here: (<https://darsh97229.github.io/dashboard/>).

Design of Heart Rate Monitor

The first step in designing the heart rate monitor was to get the sensor data to send to a python file through Open Sound Control messages. This meant there had to be code on the watch JavaScript side that could send Open Sound Control messages and code on the python side that could receive the Open Sound Control messages. Luckily for both JavaScript and Python there are open source modules that allow for each of these actions to be done in a few lines of code and are referenced below. Once there was a free flow of sensor data between these two sources the sensor data was stored in a CSV file where a separate python script could be used to signal process the data. When the original heart rate data was visualized using the dashboard it was quickly realized that there needed to be some filtering to clean up the unreliable heart rate data that was coming through. Since the watch not only let out heart rate data in beats per minute, but also in raw photoplethysmography(PPG) signals the raw signals were used. The python script read the data from the CSV file along with timestamps for when the data was being sent from the watch, in order to calculate the sampling frequency, and passed it to a 8th order Butterworth Bandpass filter with cutoffs of 0.7 and 3.0Hz. The signals were taken initially over a 10 second period and then a sliding window every second updates the heart rate for what happened in the last second. The cutoff frequencies were determined after reading research papers on different filter designs for raw PPG data taken at 10 Hz, which is close to our sampling frequency which was around 8-10 Hz varying. After the signals were filtered they were resampled at ten times the sampling rate because the algorithm used to calculate beats per minute from the raw data needed a higher sampling rate than the watch sent data. The algorithm that was used is cited below and gave out a bunch of different data from the raw signals, but ultimately we just took the beats per minute heart rate data. This data was then sent over Open Sound Control messages to the dashboard in JavaScript and published to the web application so that the data could be visualized.

The heart rate monitor can also be used for a lot of different control mechanisms as well as just visualizing the data. Right now with time constraints the only control it has is that it will make the watch vibrate if the user's heart rate is over 100 BPM. The future goal that envisioned for use of heart rate was to have a range decided upon for a healthy heart rate and as long as your heart rate stayed in that range a green led would be lit up on the watch and if out of the range a red led would be lit. Another nice feature is using the other data from the algorithm that was used

like breathing rate, it could be determined if a user is falling asleep. So say a user is wearing the watch in class or while they're driving and they turn the feature on if they are falling asleep the watch will vibrate vigorously in order to wake the user up to prevent car accidents or missing something in class. The heart rate could also be used to visualize people's reactions to artistic things like a play to see how people are reacting to certain things in real time. This or at a concert the possibilities are endless with using the heart rate in order to control or visualize things.

Design of Volume/Gesture Control

As mentioned above, once OSC messages are sending sensor data to a python script an application can be written that uses the sensor data. In this case the accelerometer data is controlling volume on the computer that the python script is running.

The python application receives OSC messages into separate channels and functions for each of the data types. For example, the accelerometer is received in a channel called “/accel” and is pushed to a function called “accelhandler”. This means that each of the data types is processed as fast as they come in and independently of every other data type.

The accelerometer on the watch produces five data types, X, Y, Z, magnitude, and difference, representing the value of the X, Y, and Z parts of the vector, the magnitude of the change from the last sample, and the total difference of the change from the last sample. In this case, the X, Y, and Z values range from -100 to 100. Controlling the volume primarily uses the Y parameter. The idea is that changing the volume is mimicking the motion used to turn the knob on a stereo. So that would mean turning the wrist clockwise for up and counter clockwise for down.

There are multiple methods to trigger the motion control. The simplest method is pressing button three on the watch. That would both activate and deactivate the volume control. The second method is using the magnitude parameter. The control can be activated when the user flicks their wrist, causing the magnitude to be large, and then begins changing volume when the magnitude returns to a normal value. In this case the magnitude value chosen was 75. Through testing it was deemed high enough to not be triggered when the user didn't mean to but also low enough that it could be triggered reliably. To deactivate the volume control the user would still have to press button two on the watch since a system that uses motion would also move the volume.

Using OSC the application is also able to send OSC messages backwards for the purpose of displaying information to the user.

A python library called “pycaw” was used to get and set the computer's volume. Specifically the SetMasterVolumeLevelScalar and the GetChannelVolumeLevelScalar methods.

Another python library called “pythonosc” was used to send and receive OSC messages to and from the python application.

Design of Smart Device Control

The first step of integrating smart devices into the system was to find available smart devices that would be compatible with Javascript or Python. After some research, Sonoff smart switch, an easy to work, simple and reliable WiFi smart switch with a Javascript compatible API called eWeLink was chosen. The eWeLink API communicates and interacts directly with the eWeLink app that connects to the smart switches using regular credentials. It offers and enables it to be run on web browsers, node scripts or serverless environments.

The API works by sending a request to the eWeLink connected devices ID in this case a Sonoff smart switch, after that it is possible to send a toggle command which will toggle the device state. Therefore, if the device is off it will toggle on and vice versa. In addition, the API is not only limited for smart switches, other devices from the Sonoff range of products are able to communicate with the API by sending power state, humidity, and temperature readings.

Currently the eWeLink API is running as Node scripts in the backend of the Dashboard, to demonstrate the use and application we are using the light bulbs plugged to Sonoff smart switches that can be triggered by data received from the watch sensors. There are two different colored bulbs, one blue representing cold and one red representing hot. In the Dashboard scripts we can set watch temperature thresholds that trigger each bulb accordingly, when the threshold is hit the eWeLink node scripts are called and the devices are toggled.

Other sensors can also be used to toggle the different devices, working the same way but with different thresholds or different conditions. Devices can be controlled also using watch buttons.

Specifications

In terms of our specifications we worked to ensure that we meet our goals in terms of bluetooth connectivity, wifi connectivity, processing speed, battery life, gesture detection, iot, access to all sensors, app connectivity, able to turn off functionality, aesthetically pleasing, and lightweight. Choosing the Bangle watch and developing our project in the manner we chose achieved significant progress towards satisfying each of these specifications.

Test Cases

Use of smart devices to determine if gesture control and other sensor data was controlling external devices. Fingertip pulse oximeter to verify that the filtered heart rate data was being processed correctly. Another fitness watch was also used to verify the heart rate data provided by the bangle watch and filtered heart rate data that was being processed. Additionally, another way we were able to verify that build specifications met our requirements via testing was to utilize more than one physical watch to verify all of our results. We used a total of three bangle watches in order to determine accuracy of each watch against each other and to verify the applications and deliverables that we developed and completed.

Build & Test

Project status/results

The project in relation to its deliverables has been completed. A web interface was developed to communicate fitness watch data to a web application communicating smart watch data to an application from which user controlled actions can take place. One specific user controlled action is the control of a PC's volume based on rotation of one's wrist while wearing the fitness watch. A database of live streamed data, heart rate, temperature, accelerometer, and magnetometer was also cached in a manner where the user can see their own sensor data's progression over a period of time when connected to the interface. Control of external devices via Wi-Fi connected Sonoff switches was also implemented to demonstrate functionality of a user to control any powered device via control of their fitness watch.

In terms of future work there is a wide array of ways to continue this project. Some of which include, expanding the interface to integrate other more popular smartwatches onto our dashboard (ex. Apple Watch, Samsung Gear, Fitbit) and developing group interaction via Open Sound Control to control volume via watch sensors and data.

Test results and analysis

Testing the latency of sending a message from the computer to the watch and back to the computer. The difference ranged between 50 - 90 ms but averaged 70 - 80ms for the connection between the computer and the watch and back to the computer.

sent: 1638556994471

recv: 1638556994545

Figure 2: Sample times from the latency test

We tested how the sensors reacted when the watch was not moving to see how consistent the data is. Shown in the following figure is the magnetometer data

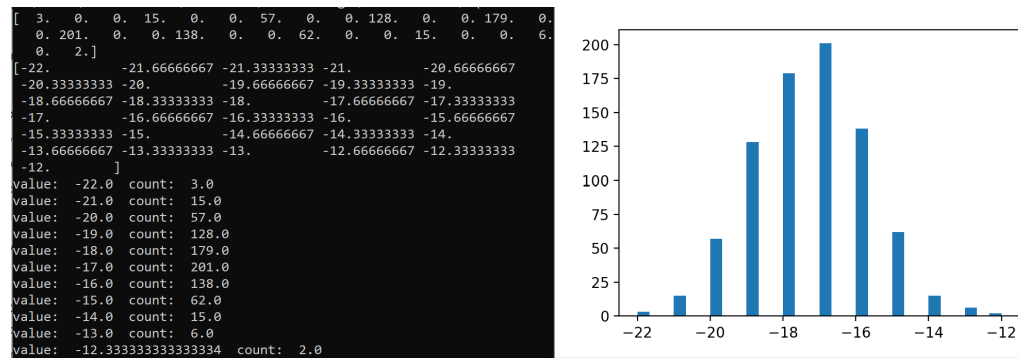


Figure 3: Graph showing the fluctuation in data with a motionless watch

To test the reliability of the systems, once each system was demonstrated to work in a general setting they were tested under as many conditions as possible.

Key Accomplishments

Our key accomplishments throughout this project include being able to pivot from one watch to an entirely different one in order to achieve the end desired result and best satisfy our customer, working to provide a successful prototype to our customer, working in full-stack technologies we had no prior experience in, and finally being able to work with numerous applications using connectivity of our watch such as being able to control external smart devices.

Delivery

Overall Project Performance

After tests and results we can conclude that the overall performance is really good, we were able to meet all the projected deliverables. Furthermore, we were able to deliver extra features like simultaneously watch pairing to the dashboard. In addition, we delivered a working demo showing the core features of the project.

Customer Satisfaction

From multiple meetings with the customer it seems our customer is pleased with what deliverables he will receive and knows there is much progress that can be made from what the

team has delivered. More testing will be done in order to ensure that the products that are delivered work all the time and just need to be extended and not altered. The customer feels that given our deliverables he will be able to fund a team to continue on our progress to develop more additions to the project.

Deliverable status

Data Dashboard: React web application that displays all real time sensor data including heart rate monitor that uses filtered heart rate sent by Open Sound Control messages from python scripts.

Heart Rate Monitor: Python script that takes data sent by watch into a CSV file and then signal processes the raw PPG signals that the watch sends. Python script sends filtered heart rate every second to the dashboard using OSC and if the heart rate goes above 100 BPM the watch will vibrate

Volume/Gesture Control: Python script containing methods that increments, decrements, sets, and gets the computer's volume. Python script that receives OSC messages containing data from the watch and controls the computer's volume based on accelerometer data.

Temperature/Smart Switch Control: Node script that makes API calls to devices connected to the desired eWeLink account, based on or set by sensor data received by OSC to the dashboard.

Build Documentation

Data Dashboard

Running the dashboard is a very simple process, starting with visiting a website where the web application has been hosted on: <https://darsh97229.github.io/dashboard/>. Once visiting this page, a site similar to the following figure will be loaded.

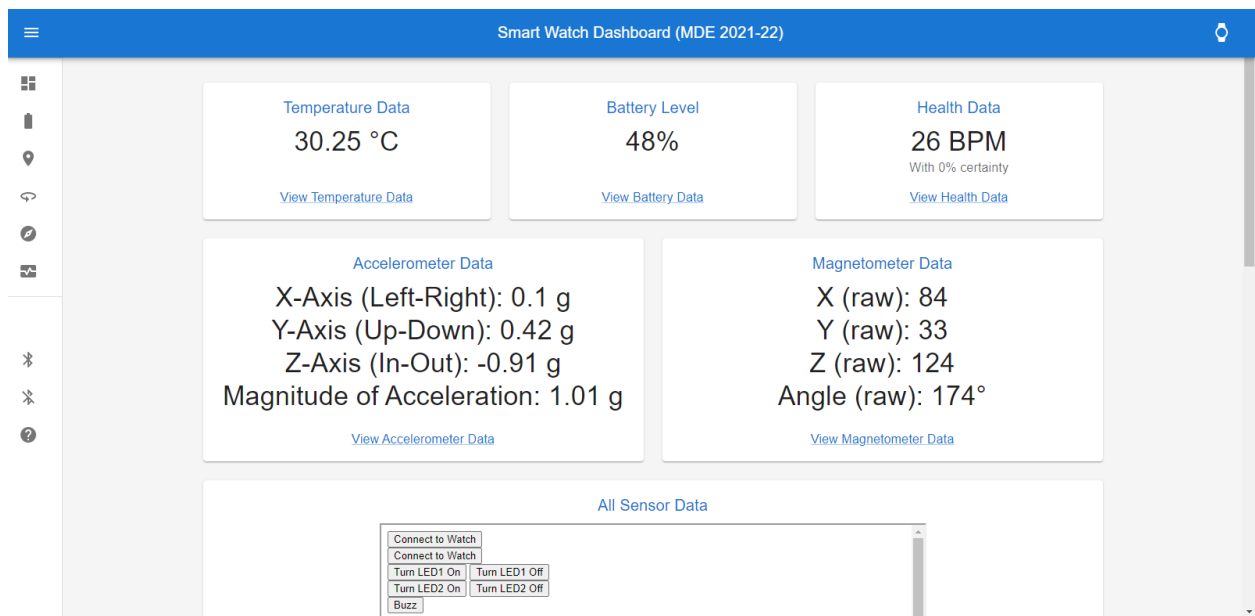


Figure 4: Dashboard on initial loading of web site

The next step is to click on the connect to watch button near the bottom of the page. Once clicked a popup will appear indicating that a bangle smart watch can be connected. The below figure shows the connection popup including a bangle watch on the list ready to be paired.

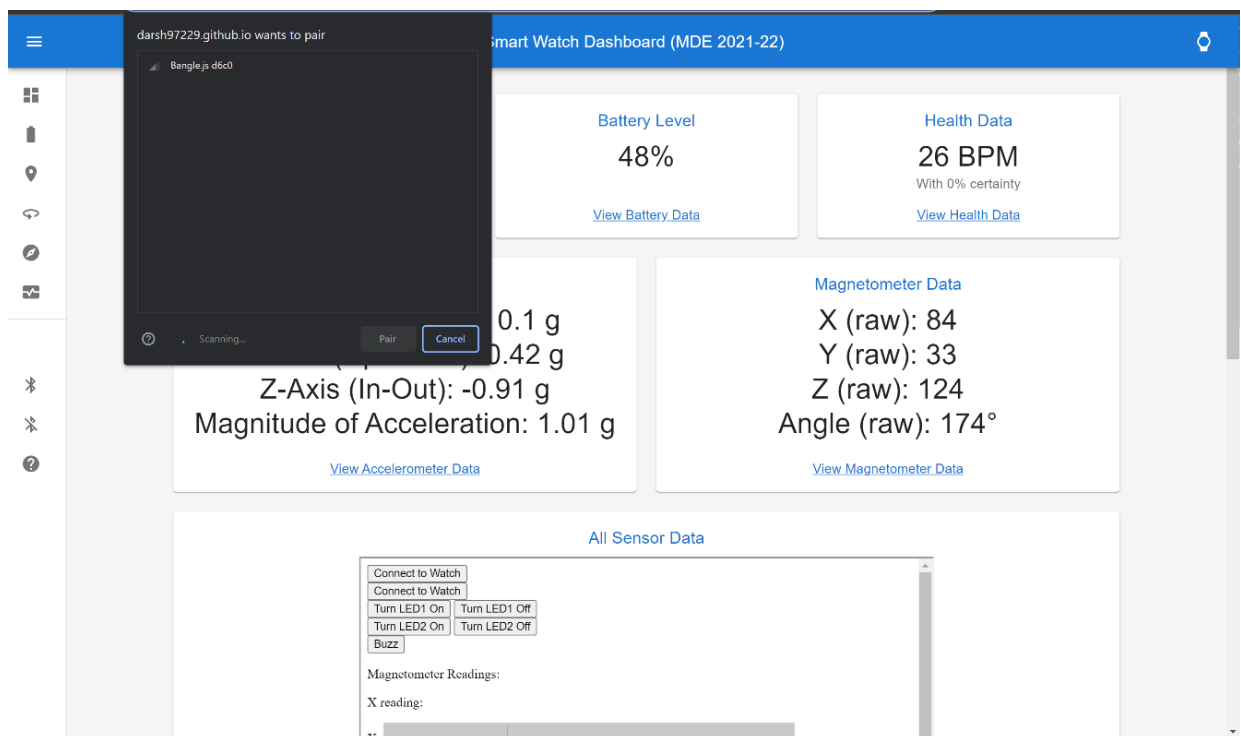


Figure 5: Connectivity screen with one Bangle watch able for pairing

After the watch has been paired, the dashboard will become active and data values will be streamed in real time and displayed, this can be seen in something as simple as the rotation of the watch and watching the accelerometer data change to reflect the watch.

The functionality of the dashboard allows for external device control based on the temperature data, if the temperature reading detected by the watch is too high or too low then a red or blue light bulb will turn on to reflect such temperature levels. In addition to this, the dashboard will also be able to detect heart rate data and if it is too high or low, turn on LEDs on the watch itself to indicate to the user such information.

Heart Rate Monitor

In order to run the heart rate monitor with signal processing there are a few things that need to be done. The first is to start in the main directory of the project and run `python -m pip install heartpy` in a command prompt. The next step is to run `node .` in order to start an OSC server so that OSC messages can be sent and received over a UDP connection. Next you have to connect the watch over bluetooth to either the dashboard or some HTML file like done with the Volume/Gesture Control. Next run `ThePythonSideOfThings.py` file and click the left side of the screen to start taking heart rate data into a CSV file. Once data is being taken into the CSV file you can run `watch_filtering.py` and it will start calculating the heart rate in BPM and the data will print to the command prompt and if you are connected to the dashboard you will be able to see it being updated in there as well.

Volume/Gesture Control

There are three parts to running the volume control. But before running anything, in the directory with the main python file (`ThePythonSideOfThings.py`) to install the dependencies you will need to run `npm install node-osc`, and `pip install pythonosc`. The first thing to run is `node .` in the same directory. This starts an OSC server to carry the messages. Next run the `ThePythonSideOfThings.py` file in a new command window and open the watch HTML file in a browser. You should see a new connection appear in the window with the OSC browser. You can see if the connection is working by clicking the “say hello” button on the webpage and seeing if the python window says hello.

Next you can connect your watch with the connect button on the webpage. Note it may take a few seconds for your system to see the watch. Once it is connected you can touch the left side of the watch screen to collect heart rate data or the right side of the screen to start looking to control the volume. To actually control the volume flick the wrist wearing the watch or press button three. The state of the system can be monitored on the window running the python.

Temperature/Smart Switch Control



To run the temperature smart control does not require much since already it is integrated into the dashboard. First we are required to connect the Sonoff smart switches to the eWeLink servers using their smartphone app, that is done by pressing and holding the button on the Sonoff switches for five seconds and releasing, this need to be done two times, which should make the light one the switch blink constantly showing that it is in pairing mode, then click to add a device in the eWeLink app and follow the guided steps.

On the code and scripting side, it is required to install the eWeLink API package by running “npm install ewelink-api”. After that the desired temperature thresholds need to be set and simply run the dashboard.

Challenges/Issues

The biggest challenge that was faced came in the first semester of the class and early stages of the design process. The first step of the project was to put together a decision matrix to compare the different watches that met the specifications. With the specifications laid out with the different watches and their attributes a watch was chosen, which was the Garmin Vivoactive 4. The initial step with working with this watch was to get it connected to a computer via bluetooth so that sensor data could be streamed via watch to a computer. While the watch promoted bluetooth and it could be connected to a phone and code could be uploaded to it, it took the team too long and wasted too much time figuring out how to get the watch connected to a computer over bluetooth and eventually the group had to move on to a watch that could be connected via bluetooth much easier. This took almost two months of valuable work time off the group's hands as there was no work that could be done until the watch was connected. So once the team gave up on this watch and selected a new one, the new watch was connected within a few days of receiving it. However, at this point the group was very behind and spent a lot of time catching up, so a few requirements were altered, but in the end a viable product was created. However, it would have been nice to get those few months back in order to work on other things other than the connectability, which turned out to be much easier with the right watch. The group should have done a little more intensive research when looking at the watches to see what work could be done with it and not just the specifications. This was a great learning experience though as now it is known what kind of research needed to be done at each stage of design and will hopefully lead to better design decisions in the future.

Schedule and Cost Status

		Name	Duration	Start	Finish
1		Sensor Management Control	45 days	1/18/22 8:00 AM	3/21/22 5:00 PM
2		Sensor Control & Monitoring	21 days	1/18/22 8:00 AM	2/15/22 5:00 PM
3		Implement Control System via React App	21 days	1/18/22 8:00 AM	2/15/22 5:00 PM
4		Sensor Analytics	45 days	1/18/22 8:00 AM	3/21/22 5:00 PM
5		Implement Historical Sensor Data Remote Saving (API)	24 days	1/18/22 8:00 AM	2/18/22 5:00 PM
6		Implement Historical Sensor Data Remote Retrieval (API)	7 days	2/21/22 8:00 AM	3/1/22 5:00 PM
7		Implement Historical Sensor Data Dashboad	14 days	3/2/22 8:00 AM	3/21/22 5:00 PM
8		Temperature Sensor Control	34 days	1/18/22 8:00 AM	3/4/22 5:00 PM
9		Research bluetooth enabled power plugs	5 days	1/18/22 8:00 AM	1/24/22 5:00 PM
10		Use temperature data to control BT power plug	29 days	1/25/22 8:00 AM	3/4/22 5:00 PM
11	 	Gesture Control	42 days	1/18/22 8:00 AM	3/16/22 5:00 PM
12		Hand Rotation (Volume Control)	42 days	1/18/22 8:00 AM	3/16/22 5:00 PM
13		Arm Movement (Pitch Control)	42 days	1/18/22 8:00 AM	3/16/22 5:00 PM
14		Complex Movement - Move arm in circle (On/Off Sound)	84 days	1/18/22 8:00 AM	3/16/22 5:00 PM
15		Signal Processing	54 days?	1/18/22 8:00 AM	4/1/22 5:00 PM
16		FINAL DEMO	62 days?	1/18/22 8:00 AM	4/13/22 5:00 PM

Researching and Testing

- Garmin Vivo Active: \$350
- NRF52-DK Bluetooth Development Kit: \$195
- Bangle.js: \$94.45 per watch
 - 4 watches: \$377.80

Demo

- 2-Pack Hanging Light Cord with Switch: \$15.99
- Sonoff Wifi Switch 3-Pack: \$20.98
- A19 LED Light Bulb 2-Pack Red: \$8.87
- A19 LED Light Bulb 2-Pack Blue: \$8.17

Total Expenditure

- \$976.81

Lessons Learned

Throughout the processing of developing and progressing with our project there were numerous moments of learning and a great many takeaways looking back. One of the largest lessons we learned throughout this project was the significance of proper time management. We were able to learn from past mistakes at points throughout the project to then move forward with

a project plan to better coordinate progress towards deliverables and meet deadlines better. Another lesson we learned was to be more analytical and rigorous in our evaluation of devices and technologies used. Specifically more could have been done while initially evaluating watches to determine that the Garmin Vivoactive would not be a viable option. Another lesson that we learned was that of team management. One member of our group did not actively participate very much at all - once he was removed from the team during the spring of 2022, the team dynamic was improved drastically and there was more headway made with tangible results corresponding to each team member. And the last but not least lesson we learned was that of communication being key. Working together with team members internally whilst also maintaining a constant level of communication along with consistently keeping in contact with our Subject Matter Expert, Mentor, and Customer allowed us to always ensure we were on track and maintaining significant and effective progress over the course of the project.