Wireless EKG

By
Chin Chao
Galen Wu
Eric Burstedt
Jim Kerman

Mechanical Engineering Department
California Polytechnic State University
San Luis Obispo
2018
Statement of Confidentiality

The complete senior project report was submitted to the faculty coach and sponsor. The results of this project are of a confidential nature and will not be published at this time.
Statement of Disclaimer

Since this project is a result of a class assignment, it has been graded and accepted as fulfillment of the course requirements. Acceptance does not imply technical accuracy or reliability. Any use of information in this report is done at the risk of the user. These risks may include catastrophic failure of the device or infringement of patent or copyright laws. California Polytechnic State University at San Luis Obispo and its staff cannot be held liable for any use or misuse of the project.
Senior Project, Final Design Review: Wireless EKG Device

Project sponsored by: Dr. Ahmad Nooristani, New Vision Telemedicine

Final Design Review prepared by Cal Poly undergraduate students:

Jim Kerman
Galen Wu
Chin Chao
Eric Burstedt

Project Advisor: Dr. Elhantry Elghandour, Cal Poly ME Dept. Faculty
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table of Contents</td>
<td>2</td>
</tr>
<tr>
<td>List of Figures</td>
<td>3</td>
</tr>
<tr>
<td>List of Tables</td>
<td>4</td>
</tr>
<tr>
<td>Abstract</td>
<td>5</td>
</tr>
<tr>
<td>Chapter 1: Introduction</td>
<td>6</td>
</tr>
<tr>
<td>Chapter 2: Background Research</td>
<td>6</td>
</tr>
<tr>
<td>Chapter 3: Objectives</td>
<td>10</td>
</tr>
<tr>
<td>Chapter 4: Concept Design Development</td>
<td>12</td>
</tr>
<tr>
<td>Chapter 5: Final Design</td>
<td>15</td>
</tr>
<tr>
<td>Chapter 6: Manufacturing</td>
<td>29</td>
</tr>
<tr>
<td>Chapter 7: Design Verification</td>
<td>32</td>
</tr>
<tr>
<td>Chapter 8: Project Management</td>
<td>34</td>
</tr>
<tr>
<td>Chapter 9: Conclusion and Recommendations</td>
<td>36</td>
</tr>
<tr>
<td>References</td>
<td>38</td>
</tr>
<tr>
<td>Appendices</td>
<td>39</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1: Fitbit’s Charge 2 Device
Figure 2: AliveCor’s Kardia Device
Figure 3: Contec’s PM10 User Diagram
Figure 4: SHL Telemedicine’s SmartHeartPro System
Figure 5: Five Lead Holter Monitor
Figure 6: Sketch of Box EKG with Pads
Figure 7: Sketch of Box EKG with Clamps
Figure 8: Sketch of Box EKG with Sockets
Figure 9: Cardboard Prototype of Wrist EKG
Figure 10: Sketch of Rod EKG
Figure 11: CAD Views of Assembled EKG Housing
Figure 12: Exploded CAD View of EKG Housing
Figure 13: CAD View Inside of Assembly without PCB
Figure 14: CAD View Inside of Assembly with PCB
Figure 15: 1D Model Used for Heat Transfer Hand Calculations
Figure 16: Temperature Distribution According to Solidworks Simulation Thermal Study
Figure 17: BioPac Systems Ag-AgCl Electrode
Figure 18: PCB Schematic
Figure 19: PCB Layout
Figure 20: Connection Screen for the iOS App
Figure 21: EKG Monitor Screen of iOS App
Figure 22: EKG Email Screen for iOS App
Figure 23: CSV File Opened in Excel Along with Add-On Graph
Figure 24: Ultimaker 3D Printer
Figure 25: Brass Heat Set Inserts Manufacturing Process
Figure 26: Heart Rate Plot After Moving Average Signal Filtering
Figure 27: Results from Thermal Tests
Figure 28: Front Side of Completed Final Prototype
List of Tables

Table 1: Product Specifications Table
Table 2: Budget Table
Abstract

In 2016, Dr. Ahmad Nooristani identified a need for a personal EKG that is cheaper and simpler than hospital grade systems, but more accurate than personal devices currently on the market. Two previous teams did significant background research, prototyping, focusing on ease of use, portability, and physical specifications. However, neither team was able to produce a device with electrodes that could successfully collect an EKG signal and transmit it to a phone or tablet, one of the main requirements from the Dr. Nooristani.

This team focused on designing a six lead, five electrode EKG device to transmit EKG data from a user device to an iOS application. The device needed to be lightweight, inexpensive, and easy for a mobility-impaired individual to use. Additionally, a focus was placed on creating a device that would be relatively intuitive to use.

This project resulted in a wireless EKG device that was able to successfully transmit data to an iPhone application in real time, and then email a .csv file of the data to a physician. Challenges in manufacturing the PCB led to the prototype only being able to handle a single lead. However, this project resulted in a prototype that can be further expanded upon moving forward. Many of the critical requirements were met, such as using dry electrodes, making the prototype at a reasonable cost, and making a user-friendly product.
Chapter 1: Introduction

Objective of Research & Project History

Dr. Nooristani is an internist and affiliated with several hospital in the San Luis Obispo area. From his experience he has identified a gap in the current medical market for a personal hospital grade electrocardiogram (EKG). For the past two years, Dr. Nooristani has developed a concept with two groups of engineering students to create a portable and wireless EKG that utilizes four dry electrodes. This system will allow both hospital staff and patients to effortlessly monitor a heart rate without having to perform the time consuming setup and operation that a traditional EKG would require. This freedom will reduce the time required for doctors to identify heart defects and ultimately save lives.

This senior project will investigate the design of this EKG to specification. This year’s team is currently composed of two CPE students, Galen Wu and Chin Chao, an EE, Ryan Blaalid, and two ME students, Jim Kerman and Eric Burstedt. Galen and Ryan will focus on the electrical side of the EKG. Their goal is to provide a clean and consistent signal from the electrode to the phone. Chin will focus on creating an iOS app that will interpret the incoming signal and generate a CSV of the waveform. The packaging and construction of the EKG will be led by Jim and Eric. Ultimately, this team’s composition is aligned with the required skill set of the project to effectively create a working prototype.

Chapter 2: Background Research

Customer Needs

The self-administered EKG device shall be primarily used by patients with heart condition, with focus toward the elderly. Most of them have stated they have trouble using complex technology without professional guidance. The customers would prefer an EKG device and app that are simple to use. The app will have a simple user interface for them to access the EKG information. Some customers may become anxious if the device displays information regarding their health condition, especially if the information states they have a certain health issue. The app should display only the basic heart information for the customers to prevent unnecessary anxiety. They will only need to be aware of their basic health condition. The hospital staffs shall be the one to determine the exact condition for the customers, through interpreting the EKG data sent by the app.

The device’s main use will be to monitor the customers’ daily heart condition. It will obtain the user’s heart rate within a minute, so customers can quickly access their heart data and send it to hospital staff for immediate response. This will be essential for urgent situations such as myocardial infarction.

Existing, Consumer-Ready Personal EKG’s

There are several personal EKG designs on the market today. They vary widely in number of leads, functionality, available features, price, etc. Because these devices are meant for personal use, almost all of them prioritize size and ease of use. Most of them also feature smartphone connectivity and dry
electrode sensors.

The simplest of these devices is Fitbit’s Activity Tracker. A simple wristband, with one electrode, the Fitbit is only able to track as user’s heat rate, but for a very cheap price and small package. It also operates as a lifestyle device, tracking exercise and sleep.

![Figure 1 - Fitbit’s Charge 2 device](image)

Slightly more complex are 2-electrode, single lead EKG systems such as AliveCor’s Kardia system. These devices allow for the detection of atrial fibrillation - a condition in the heart that can lead to a stroke. These systems are fairly inexpensive and feature smartphone connectivity, but can only capture a single EKG lead, limiting their functionality as a medical diagnostic device.

![Figure 2 - AliveCor’s Kardia device](image)

Similar devices such as Contec’s PM10 and PM80 series Portable ECG devices add some functionality to this system by allowing for multiple measurement modes to collect up to 4 different leads. However,
data can only be collected from one lead at a time, meaning this method is equally limited in its ability to diagnose. A more complete EKG tracing is needed.

Measurement modes

![Limb Leads I](image1)
![Chest Leads V](image2)
![Limb Leads I](image3)
![Limb Leads II](image4)

Figure [3] - Contec’s PM10 user diagram

On the other end of the portable EKG spectrum, SHL Telemedicine’s SmartHeartPro is one of the most comprehensive devices available to consumers today. This system provides a FDA approved, hospital grade 10 electrode, 12 lead tracing with live and continuous monitoring options. Like the other systems, it is smartphone connected and can upload readings to the internet. More accuracy comes at a price though. The SmartHeartPro costs more than $1000 and must be strapped to the patient’s chest, limiting its use and convenience.

![SHL Telemedicine’s 10 electrode, 12 lead SmartHeartPro system](image5)

Figure [4] - SHL Telemedicine’s 10 electrode, 12 lead SmartHeartPro system
The Holter monitor is an EKG device designed to collect heart data over time. It typically has between 3 and 8 sticky leads, and can cost between $200 and $800 dollars. Designed in the 1960s, it has proven reliable but has not been updated for the 21st century.

Applicable Standards
Current EKG systems require FDA testing and approval in order for the system to be sold or used on patients. Testing covers the time interval consistency and amplitude readings of the wave tracing. Approval is granted when the testing meets the current EC11 standard or any other standard with requirements. Devices should also be tested and verified for in-vitro safety, which includes environmental impacts, software, electrical, and electromagnetic compatibility safety considerations.

Traditional EKG monitoring systems require the user to apply pastes or gels to the contact skin in order to establish a reliable electrical connection. New EKG technology introduces a miniature, low-noise, capacitive sensor that does not require contact to the skin and the results are comparable to modern EKG machines. Eco is a new sensor that provides a reliable, untethered way to sense heart activity.

Due to how sensitive EKG signals are, filtering is an important aspect of the circuitry. The signal voltage range is as low as 0.5mV to 5mV and is highly susceptible to noise. Additionally, the read frequency falls into a range of 0.05 to 100Hz which is again susceptible to noise. Muscle movements and ambient electromagnetic interference can affect the EKG readings. A variety of different filtering techniques are required in order to obtain the correct signal at the correct frequency.
Chapter 3: Objectives

Scope of Research

Due to most current EKG devices (e.g. Traditional 12-lead EKG device) being too large to carry around and too complex for most customers to understand and utilize, the patients will need a more efficient and convenient method to monitor their heart rate. A portable and wireless EKG will allow doctors to monitor their patient remotely without direct professional assistance.

After discussing the functional prototype requirement with Dr. Nooristani, the self-administered EKG shall be a handheld device with dry, wireless electrodes and five-electrode six-lead EKG reading. It should be simple to operate and require a minimum of strength and dexterity. The EKG data shall be received by an iOS app, which will display the user’s heart rate. The app should display basic heart information for the users, such as the user's heart rate and whether the heart rate is regular or irregular. The app shall allow the information to be sent to the hospital staffs for deeper interpretation of the data. The EKG device and the app shall communicate through Bluetooth and 3G if feasible.

The self-administered EKG device for this project must have the following basic requirements:

- The device needs to be handheld for user portability
- The device needs to require a minimum and strength and dexterity to operate
- The device needs to have dry wireless electrodes for customers’ comfort in usage
- The app needs to receive the EKG data wirelessly and send the data to the hospital staffs for interpretation (Functional at least in either Android or iOS)
- The EKG device and the app can be used by the customers without professional assistance
- The app will only display the heart rate and it is regular or irregular

QFD process, Specifications Table, discussion of specifications

QFD was satisfied using the House of Quality approach. Technical analysis of the design was facilitated using the given template. By inputting values for the importance of each customer requirement and the relevant engineering specifications, as well as the performance of the team’s current design and the designs of competitors, a relative weight can be assigned to each requirement and specification. This allows the team to allocate the proper time and resources to each need to ensure customer satisfaction. The use of the HoQ showed that the most important needs for the customer were for it to have specific design choices such as having 5 wireless, dry electrodes and for it to be portable and easy to use.
Table [1] - Specifications Table

<table>
<thead>
<tr>
<th>Spec. #</th>
<th>Spring Qrt: Phase II</th>
<th>Fall Qrt: Phase III</th>
<th>Tolerance</th>
<th>Risk</th>
<th>Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cost</td>
<td>$400</td>
<td>Maximum</td>
<td>3</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>Size</td>
<td>16 in²</td>
<td>Minimum</td>
<td>2</td>
<td>A, I</td>
</tr>
<tr>
<td>3</td>
<td>Battery Life</td>
<td>3 Hours</td>
<td>Minimum</td>
<td>1</td>
<td>T, A</td>
</tr>
<tr>
<td>4</td>
<td>User Interface</td>
<td>Simple</td>
<td>N/A</td>
<td>1</td>
<td>I, S</td>
</tr>
<tr>
<td>5</td>
<td>User Operation</td>
<td>Simple</td>
<td>N/A</td>
<td>1</td>
<td>I, S</td>
</tr>
<tr>
<td>6</td>
<td>Signal Processing and Filtering</td>
<td>Effective</td>
<td>N/A</td>
<td>3</td>
<td>T, A, S</td>
</tr>
<tr>
<td>7</td>
<td>Bluetooth Compatibility</td>
<td>Yes</td>
<td>N/A</td>
<td>0</td>
<td>I</td>
</tr>
<tr>
<td>8</td>
<td>5 Electrode 6 Lead Design</td>
<td>Yes</td>
<td>N/A</td>
<td>1</td>
<td>I</td>
</tr>
</tbody>
</table>

- Cost is a non-critical, high risk specification. In order to be successful it needs to be less expensive than the more complex alternatives, but the focus should be on the design and operation.
- The size of the design should be no larger than what would comfortably fit in two hands. This is critical to the portability and ease of use.
- Battery life is not critical for the prototype, though in the future, the design’s portability will make this specification more important.
- It is very important for the application user interface to be simple. With the goal being self-use, the design must allow input and output with a minimum of confusion to users. It should be compared to the effectiveness of competitors designs.
- It is equally important for the User Operation of the design to be intuitive. Independent use is a priority.
- Effective signal processing and filtering must be achieved for the design to be a success. This area poses the most critical risk as it is where both previous teams have struggled.
● The design must have Bluetooth compatibility in order to communicate with the iPhone application.
● The design must also follow the established 6 lead - 5 electrode specification.

Chapter 4: Concept Design Development

Before starting the circuit design process, developing the functionality and form factor of the EKG is required. In order to sample and collect the necessary data, the EKG needs to make contact with the user’s upper chest and hands. Depending on how the packaging is designed, this maneuver may be difficult to elderly users. The users require an EKG that can accommodate limited mobility and dexterity. In addition to the basic five electrode design, we created a list of possible form factor solutions.

The first concept is to have a box EKG with finger pads for the user. One side of the box would have the pads and the other side would have contact plates for the user to push against their chest. This fulfills the contact and functionality requirement but may be difficult to use for a long period of time due to a lack of support.

![Figure 6 - Sketch of Box EKG with Pads](image-url)

The second concept is very similar to the design above but with finger clamps. The back of the box would have the chest contact plates and the sides of the box would have clamps to hold the users fingers. The clamps provides a more accurate, reliable reading but may be uncomfortable for the user. Similar to the issue above, it wouldn’t provide enough support to comfortably to press the EKG against the chest.
The next concept is again similar to the previous ideas, but utilizes finger sockets. There would be two finger sockets on each side of the box and be large and tapered so that it will fit all finger sizes. In addition, the sockets would also support the EKG because it will rests on top of the user’s fingers.

One problem with the past three concepts is that it measures from the user’s fingers. Measuring from the wrist would provide more consistent and accurate data. Our fourth idea was to have a sloped design where the user can rest both their wrists against the EKG. The EKG would also press against the chest to fulfill the 5 electrode design. This concept was prototyped by constructing a cardboard equivalent and was not intuitive to use. In additional, it would also be difficult to store and travel with an EKG of that shape.
The final concept is an EKG with a rod handle attached to a slim box. The box would contain two leads of the EKG that will be pressed against the chest. The rod will measure the user’s palms and provide support for the EKG. This design is the most intuitive and comfortable but it also lacks in portability.

In order to make the best decision for the form factor, we created a decision matrix, found in Appendix B. From this, it helped us to decide to use the box with sockets design. This design provides easy use for users to hold and apply the electrodes. In addition, the sockets of the design would result in more accurate data due to how secure the contact is to the fingers.

During Week 8 of Winter quarter, we built a wooden prototype of the form. After testing it on different people, we are generally pleased with the design and the ease of use. One adjustment we would add on the next iteration is to change the positioning of the sockets. By moving the socket positions so that it’s around one corner, it would decrease the use of the shoulders, requiring less flexibility, and be easier to support. Another piece of feedback was to add LEDs to signal the user if the electrode has the correct contact and when the device has completed its reading.
We met with the sponsor to receive feedback regarding our final form factor design. He is happy with the design and recommended some changes. One thing is to have only one socket per side rather than two. He suggests that any unnecessary or redundant aspects of the EKG be removed to provide simplicity and cost efficiency as well as reduce the overall size. He also suggested that we use more than two electrodes for the chest so that we can get more relevant heart information. We will apply these changes to our design as we continue to develop and prototype next quarter.

The design process this quarter helped us determine our final form design of the EKG, which was the box design with finger sockets. We created several prototypes of different materials to help us find issues and gather feedback. The current design has two finger sockets, one of the side and one on the bottom, and three chest contacts. With the feedback from our colleagues and the sponsor, we will implement user feedback through LEDs, different position of the finger sockets to increase ease of use, and a reduction of sockets for simplicity.

**Chapter 5: Final Design**

*Overall Description and Layout*

The updated design seen in Figure 11 has three chest electrodes on the front, one finger electrode on the side, and one finger electrode on the bottom. The housing contains two small slots on the side so a wrist strap can be connected to the device. The device has an external on/off switch on the side. An external LED was included that turns on a green light when the user powers on the device with the external on/off switch. The LED serves as a visual confirmation that the device is indeed powered on. The LED will emit a steady yellow color when the battery needs to be recharged. The device pictured measures 6 inches, 4 inches tall, and 2 inches thick. Due to the nature of prototypes and the frequency of which last minute changes are often needed with hardware, ample room has been provided on the interior of the device for extra wires, components, and assembly space. A “clearance space” with the footprint of the PCB and a height of 0.75” was used in the CAD models to ensure the extra space was maintained throughout design changes.
Detailed Design Description

An exploded view of the assembly can be seen in Figure 12. The housing is broken into five components, each of which was 3D printed. The orientation of the component on the 3D printer build plate was the main consideration affecting how the overall housing was broken into its five parts. Effort was taken to avoid any overhanging faces or features during the 3D print. Additionally, a clearance space of 0.02” was included on the 3D printed slots and rails that interacted with other 3D printed parts. Due to the nature of the manufacturing process, plastic parts can slightly contract or warp after they’ve cooled down, making this clearance space essential.

The entire assembly is held together by four 8-32 stainless steel Philips screws. The screws attach the back plate to the front plate and prevent the side and bottom panels from sliding out of their designated slots. The stainless steel screws interface with brass heat set inserts, which were melted into holes in the front piece. The brass inserts improved the reliability of the part since the other option would be tapping the PLA plastic, producing threads that could be easily stripped. The PCB, the biggest component inside the housing, was screwed into brass heat-set inserts that had been melted into raised posts. This allowed the battery and battery charging board to be placed underneath the PCB. Four 2-56 plastic screws were used to hold the battery charging board and PCB in place. Plastic screws were chosen to avoid any issues with conductivity. The 2-56 screws were big enough so the head of the screw wouldn’t slip through the holes on the PCB and were small enough so the diameter of the threading was able to fit through the holes on the charging. Using the same screw size to hold both components helped to reduce the cost and complexity of manufacturing. The detailed drawings of each part can be seen in Appendix E.
The holes for the chest electrodes, LED light, and On/Off switch were all counterbored so that the component would be flush with the outer surface of the housing, yet unable to be accidentally pushed into the housing. The On/Off Switch was designed as a press-fit device, meaning no additional means were required to hold the component in place. The chest electrodes and LED light all required the use of epoxy on the inside of the housing to prevent the components from falling out.
Figure [13]: CAD view of inside of assembly without PCB.

Figure [14]: CAD view of inside of assembly with PCB.
Analysis Description and Results
The primary mechanical analysis required for the design was an evaluation of the device’s ability to dissipate heat. Due to the nature of the device, a consumer medical device held by users against their chest, a stress analysis was not warranted. Drop tests can be performed to ensure the device won’t break if dropped from a reasonable height. A load analysis for this would likely prove to be inaccurate due to the difficulty associated with estimating the time required for the device to come to a stop after impacting the ground, which greatly affects the load calculated. Since manufacturing the PCB and housing took a considerable amount of time, drop tests were not conducted, for the sake of not damaging the one working prototype. However, were this device to be manufactured on a larger scale, drop tests would be recommended.

The thermal analysis was conducted to ensure the electronics would not overheat. The analysis was conducted via hand calculations and a thermal study in Solidworks Simulation. For the hand calculations, the model was assumed to be one dimensional. The convection from the electronics to the inside air within the housing, the convection from the inside air to the inside of the wall, the conduction through the PLA wall, and the convection from the outside wall to the outside air were all considered and can be seen in Figure 13.

![Figure 13: One dimensional model used for heat transfer hand calculations.](image)

It was assumed that no convection would occur from the outside of the housing to the outside air on the front of the device since the user will be holding the front of the device to their chest. Efforts were taken throughout the analysis to ensure the calculations were performed with a degree of conservatism. The coefficient of convection of air, a value which can range 10 to 100 W/(m^2 K), was chosen to be 10 W/(m^2 K) for the sake of these calculations (6). A higher value would tend to be used for forced convection, whereas this type of application has free convection, requiring a lower coefficient that reflects the reduced rate at which heat is transferred per unit surface area and unit degree. For reference, a value ranging up to 25 W/(m^2 K) could conceivably be used for free convection with air. The electronics were assumed to dissipate 0.5 watts, a rate which should be higher than the actual value. The thermal conductivity of PLA was taken to 0.1 W/(m K), a published value.
The hand calculations indicated the highest value the electronics would reach is 317 Kelvin, or 111 Fahrenheit. The Solidworks Simulation study showed the maximum value to be 318 Kelvin, or 113 Fahrenheit, seen in Figure 14.

![Temperature distribution according to Solidworks Simulation thermal study.](image)

These temperatures will present no problems for the electronics. Another important temperature to consider is the maximum temperature of the PLA wall. A high temperature could cause significant thermal expansion, potentially causing large shearing stresses within the housing. However, the maximum wall temperature was calculated to be just 306 Kelvin, or 91 Fahrenheit. The coefficient of thermal expansion for PLA is $72 \times 10^{-6} \text{ m}(\text{m K})$, meaning any thermal expansion and thermal stresses will be negligible (7). Additionally, the fact that the entire structure will be made of the same material means the whole structure will thermally expand at the same rate, reducing the chances for any large internal shearing stresses.

The tests proved that the device would operate in a safe-temperature range while in use. The analysis was very conservative but gave an early indicator that temperature wouldn’t be an issue while operating.
Cost Analysis

The majority of the costs associated with building this prototype were associated with purchasing the electronics. The cost breakdown can be seen in Table 2.

Table [2]: Budget

<table>
<thead>
<tr>
<th>ITEM</th>
<th>DESCRIPTION</th>
<th>SUGG. MFG.</th>
<th>QTY</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Evaluation Kit for the MAX30003 Biopotential AFE</td>
<td>Maxim</td>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>Ultra-Low Power, Single-Channel Integrated Biopotential (ECG, R to R Detection) AFE</td>
<td>Maxim</td>
<td>10</td>
<td>5.263</td>
</tr>
<tr>
<td>3</td>
<td>Bluetooth 3.002.15.1 Development Tools Adafruit Feather M0 Bluefruit LE</td>
<td>Adafruit Industries LLC</td>
<td>2</td>
<td>30.99</td>
</tr>
<tr>
<td>4</td>
<td>Battery Packs Lithium Ion Polymer Battery 3.7V 1200mAh (1 piece)</td>
<td>Adafruit Industries LLC</td>
<td>2</td>
<td>9.09</td>
</tr>
<tr>
<td>5</td>
<td>B-32 0.312&quot; Lg Brass Heat Set Insert for Plastic</td>
<td>McMaster Carr</td>
<td>1</td>
<td>8.62</td>
</tr>
<tr>
<td>6</td>
<td>B-32 7/16&quot; Lg Steel Flat Head Phillips Screw</td>
<td>McMaster Carr</td>
<td>1</td>
<td>3.94</td>
</tr>
<tr>
<td>7</td>
<td>0.025&quot; Thick Aluminum Sheet Metal</td>
<td>McMaster Carr</td>
<td>1</td>
<td>16.5</td>
</tr>
<tr>
<td>8</td>
<td>Ag-AgCl Electrodes</td>
<td>Biotelemetry Systems</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>9</td>
<td>ABS Print Material</td>
<td>Miltus</td>
<td>1</td>
<td>24.99</td>
</tr>
<tr>
<td>10</td>
<td>Adafruit Micro-Lipo Charger for LiPo/Universal Batt with MicroUSB Jack</td>
<td>Adafruit Industries LLC</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Oscillator XO 32768kHz CMOS SMD</td>
<td>Doodles Incorporated</td>
<td>1</td>
<td>2.48</td>
</tr>
<tr>
<td>12</td>
<td>Conn Header Female 16POS 1&quot; Tin</td>
<td>Sullins Connector Solution</td>
<td>2</td>
<td>0.90</td>
</tr>
<tr>
<td>13</td>
<td>IC Reg Linear 1 6V 20mA SOT23-5</td>
<td>Maxim Integrated</td>
<td>3</td>
<td>1.95</td>
</tr>
<tr>
<td>14</td>
<td>CAP CER 0.1 uf 10V XFR 0402</td>
<td>TDK Corp</td>
<td>10</td>
<td>0.05</td>
</tr>
<tr>
<td>15</td>
<td>RES SMD 499K Ohm 1% 1/10 W 0603</td>
<td>Panasonic</td>
<td>10</td>
<td>0.085</td>
</tr>
<tr>
<td>16</td>
<td>RES SMD 0.033uf Jumper 1/8W 0603</td>
<td>Vishay Dale</td>
<td>20</td>
<td>0.044</td>
</tr>
<tr>
<td>17</td>
<td>RES SMD 499K Ohm 0.1% 1/8W 0805</td>
<td>Panasonic</td>
<td>10</td>
<td>0.302</td>
</tr>
<tr>
<td>18</td>
<td>RES SMD 10K Ohm 0.5% 1/10W 0805</td>
<td>Susumu</td>
<td>5</td>
<td>0.11</td>
</tr>
<tr>
<td>19</td>
<td>CAP CER 10pF 1KV COG/NPO 0805</td>
<td>Kemet</td>
<td>10</td>
<td>0.3</td>
</tr>
<tr>
<td>20</td>
<td>CAP CER 1000pF 50V XFR 0402</td>
<td>Murata Electronics</td>
<td>10</td>
<td>0.035</td>
</tr>
<tr>
<td>21</td>
<td>CAP CER 1uf 10V XFR 0402</td>
<td>Samsung</td>
<td>20</td>
<td>0.332</td>
</tr>
<tr>
<td>22</td>
<td>CAP CER 10uF 6.3V XFR 0603</td>
<td>Samsung</td>
<td>20</td>
<td>0.309</td>
</tr>
<tr>
<td>23</td>
<td>CONN Header Female 12 Pos 0.1&quot; Tin</td>
<td>Sullins Connector Solution</td>
<td>2</td>
<td>0.73</td>
</tr>
<tr>
<td>24</td>
<td>CONN Header Female 4 Pos 0.1&quot; Tin</td>
<td>Sullins Connector Solution</td>
<td>2</td>
<td>0.45</td>
</tr>
<tr>
<td>25</td>
<td>Stencil</td>
<td>Bay Area Circuits</td>
<td>1</td>
<td>130</td>
</tr>
<tr>
<td>26</td>
<td>#6073 (2 Layer Instant Quota)</td>
<td>Bay Area Services</td>
<td>2</td>
<td>114.5</td>
</tr>
<tr>
<td>27</td>
<td>Shapely 4 Flutes Putty Scrapers</td>
<td>Mentor Computer Co</td>
<td>1</td>
<td>7.99</td>
</tr>
<tr>
<td>28</td>
<td>MG Chemicals 6341 No Clean Flux Paste</td>
<td>Amazon.com Services</td>
<td>1</td>
<td>10.95</td>
</tr>
<tr>
<td>29</td>
<td>Solder Paste sns42/65 B/Ag4 No-Clean Lead-Free</td>
<td>Chip Quik</td>
<td>1</td>
<td>15.95</td>
</tr>
<tr>
<td>30</td>
<td>MG Chemicals 8397CM Thermal Conductive Adhesive, Medium Cure</td>
<td>Amazon.com Services</td>
<td>1</td>
<td>13.87</td>
</tr>
<tr>
<td>31</td>
<td>Gorilla Ultra Control Super Glue Gel, 15g, Clear</td>
<td>Amazon.com Services</td>
<td>1</td>
<td>6.49</td>
</tr>
</tbody>
</table>

Total = $ 711.92

The final prototype cost worked out to be approximately $720. This cost included extra screws, heat set inserts, electrodes, and PLA filament, meaning multiple iterations of the housing were made at no extra cost.

At the conclusion of manufacturing, there was still a lot of leftover solder, epoxy, and mechanical components. If several more iterations were to be produced, the cost per iteration would come down significantly. Also, part of the reason for the relatively high cost of the PCB was due to the fact that a
short lead time was required. If this team had a schedule that accommodated waiting 6 weeks for a PCB to arrive, then the unit price would have been much lower.

While it is impractical to extrapolate the potential unit cost for mass-producing the device, it is reasonable to say that the cost could be cut in half, if not more. Mass-producing the device could definitely bring the costs down to a point that could make it relatively affordable to anyone.

*Material, Geometry and Component Choices*

The 3D printed housing is made out of polylactic acid (PLA) filament. Each 3D printed part was designed to avoid overhanging material, thus avoiding the need for support material. The overall housing design is more function than aesthetic based. As the internal electronics are finalized, the dimensions of the housing can be adjusted accordingly, and any desired aesthetic modifications implemented.

One of the biggest component choices to still be finalized is the type of electrode to be used. This group has decided to compare several different electrode types, including Ag-AgCl electrodes purchased from BioPac Systems, aluminum 6061 electrodes manufactured by this team from sheet metal, and 99.9% silver electrodes manufactured in the same manner. Due to the popularity of wet electrodes and various issues associated with dry electrodes, there is a lack of dry electrodes available online for purchase. Research was done into electroplating, a process involving covering a material in the desired net shape of the electrode with a coating of another, conductive material. However, this tends to be a somewhat complicated procedure that often produces somewhat mediocre results. Additionally, the exact steps of the process tend to be proprietary information and are not commonly shared. Developing and tweaking an electroplating process to give the required properties of the electrode could potentially take months to do and would likely merit its own project.

One of the bigger problems encountered with dry electrodes are their susceptibility to motion artifact. Motion artifact is noise in data produced by respiratory, muscular, or other movements from a patient. Electrodes made of conductive polymers or foams exist and have the ability to change shape as the body and skin move, reducing unwanted noise. However, finding commercially available versions is challenging and their overall signal quality and durability may not meet the expectations required on this project. Manufacturing such an electrode would be beyond this group’s capabilities and resources. A common way to address motion artifact is to include pointed bumps on the electrode, as seen in the electrode being purchased from BioPac Systems, shown in Figure 15. These bumps locally compress the patient’s skin underneath each electrode, creating a mechanical and conductive interface with the skin (8). This type of shape is likely manufactured by metal casting, with a mold shape creating the desired bumps. Therefore, the electrodes manufactured by this team will not feature these bumps. If this product were to be mass-produced, it would be highly advised, as well as financially viable, to include pointed bumps on any manufactured electrodes.
With any dry electrode, the lack of a wet conductive gel means the electrode-to-skin impedance will be somewhat poor compared to wet electrodes. Dry electrodes will have higher electrical impedance, or resistance to current flow under an applied voltage. Different research papers indicate that a “settling time” of a few minutes may be required to achieve a readable signal. This settling time allows for perspiration, which acts as a pseudo-gel (8). Experimentation with the various electrodes should shed more light on this.

**Flowcharts, Schematic, Pseudo-Code, Wiring Diagrams**

Rather than designing the analog front-end of the EKG, this team will use a pre-packaged and tested integrated circuit from Maxim Integrated. The MAX30003 is a ultra-low power, single-channel integrated biopotential analog front end IC designed specifically for EKG single lead measurements. The data obtained from the MAX30003 will provide a single lead so in order to create a six lead system, three MAX30003 chips are required.

In addition to having a system capable of collecting six lead heart data, it’ll also need to be Bluetooth compatible with iOS devices. Apple’s MFi Program is an extensive licensing program to validate hardware and software to match with iPhones, iPods, and iPads. Because of this, there are few microcontrollers that can pair with iOS devices and match the specs of the project. Fortunately, Adafruit’s Feather M0 Bluefruit LE is a slim microcontroller with low powered Bluetooth functionality that has been approved by Apple. With this, the schematic shall include one M0 Feather and three MAX30003 ICs, as seen in Figure 16.
The PCB’s schematic and layout was both designed in KiCad. Female headers would be used to mount the microcontroller directly to the PCB. From there, the microcontroller was then routed to its respected areas, connecting all three MAX30003s together. Additional ports were also added to create easy access for testing and debugging purposes. U1 is a linear regulator to properly manage the power of the three MAX30003s. The lock would be supplied by the microcontroller by outputting the clock signal through pin 15.
After creating and verifying the schematic, the layout was then constructed in KiCad. Mounting holes were placed on the corners as specified by the mechanical team. Seven smaller mounting holes were also added to the edge of boards to connect the wires of the electrodes to. Ground stitches were added to connect the two ground floods together. After reviewing both the schematic and layout, gerber files were then created and sent for DFM of the PCB manufacturer. Once the DFM changes were added and verified, the PCB was ready to be sent to be manufactured by Bay Area Circuits.

The mobile app is currently developed in iOS. The mobile app will connect to the EKG device (Adafruit Arduino) through wireless Bluetooth Low Energy and receive the data from the device once the connection is established. The app will perform Moving Average filtering to reduce the signal noise in the data. It will show the heart rate plot with signal filtering on the app. Once the app obtained a readable heart rate data, it should be able to export the result to the physicians for them to interpret more in details.

The user interaction (UI/UX) for the mobile app should include a connection screen to perform EKG search and connection, a main monitor screen to show the heart rate plot, an export screen to send the result to the physicians. Due to the main focus on developing a functional EKG monitor app, there will be minimal UI for ease of usage and simplicity in app development.
The connection screen shown in Figure 20 currently include a basic description on connection and searching tutorial, a connect button that will appear when there is a compatible EKG device available during searching mode, and a refresh button at the top right to restart searching. The current searching interval is 17 seconds. Once the searching mode ends, the refresh button can be used to starts searching again or stop the current attempt to connect if it takes too long. When the connect button is clicked and the connection is successful, the app will go into the monitor screen.

![Connection Screen](image)

*Figure [20]: The Connection Screen of the iOS App*

The monitor screen shown in Figure 21 currently include a chart to show the heart rate plot with signal filtering (Moving Average of 5 points), a Start/Stop button at the bottom left to start or stop plotting data, an email button at the bottom right to prepare exporting the result, and a disconnect button at the top left to disconnect the EKG device from the app and go back to the connection screen. The email button will appear when there are EKG data stored in the app and the app stops plotting. The EKG data in the app along with the chart will be cleared whenever the app starts plotting again. The moving average filter in the app is developed to allow the filter point range to be modified if needed.
The export screen shown in Figure 22 currently shows an email view that contains the result in CSV format for export. The user can choose who to send the email to, rewrite the email content descriptions, and either cancel or send the email.

Once the email is sent, the CSV file from the email can be opened in Excel sheet for view. The original CSV file contains only two columns of information, the plot index (Time) and the data value (Data). The
Excel sheet shown in Figure 23 was created before the moving average filtered was set up in the app. The Excel sheet allows moving average filter to be performed on existing data plot, which created the third column’s data (Moving Average). The figure shows the impact of the 5-point moving average filter on the original data. The signal filtering reduces the noise signal while making sure the heart rate value is not affected severely.

![Excel sheet shown in Figure 23](image)

Figure [23]: The CSV File Opened in Excel Along with the Add-On Graph (Filtered and Unfiltered)

Safety, Maintenance, and Repair Considerations
This device contains no moving parts, no pinch points or sharp edges and doesn’t weigh that much. It does operate with electric power, and precautions have been taken to isolate the end user from risks associated with electricity. However, the total power involved in the device is so low that the risk of injury is already very slight.

The robust plastic casing should do a good job protecting the sensitive electronic components from damage. If something were to break, the bolted assembly and exposed circuitry should allow the prototype to be fairly simply repaired and maintained. On the other hand, because the electronic components are very small and fairly complex, repairs may not be possible beyond simple fixes like re-soldering wires or adjusting pins. We’ve purchased extra parts in case we end up needing to replace whole components.
Chapter 6: Manufacturing

*Procurement*

All of the purchased components were procured online, through Amazon and the various manufacturers’ websites. They were shipped to our location and assembled with the rest of the materials.

*Electrical*

After receiving the PCB from the manufacturer, it was realized that the bottom solder mask was missing. The KiCad layout did not have a bottom solder mask layer so the bottom solder mask was not created. Fortunately, this inconvenience did not greatly affect the development of the project, but would add difficulties such as advanced soldering. Construction and surface mounting of the components proved to be difficult due to a lack of proper equipment such as precise soldering tips and thin solder. Because of the size of the current soldering tips, it made it nearly impossible to solder the surface mount components directly to the PCB. Ultimately, without the necessary expertise and equipment, construction of the PCB was halted.

In order to replace the need of the PCB, the team decided to lower the specification of the project. For prototyping purposes, the team created a prototype EKG that will transmit one lead of heart data. A MAX30003WING was used to house one MAX30003 chip. The Wing is a development board for the MAX30003 that can directly connect to the Feather. The casing was then adapted to support the Feather and the attached Wing. A cable was soldered from the electrodes and attached to the Wing’s 3.5mm jack. With the current adaption to the design, the team successfully demonstrated the functionality of the project.

*Manufacturing*

Several of the components were self-manufactured. The casing itself was initially 3D printed using a FlashForge Creator Pro Dual Extrusion 3D printer owned by one of the team members. However, various iterations failed to yield a completed housing of acceptable quality. The first test print yielded a front part where the top layers of filament began delaminating from the rest of the part. Later iterations featured parts that had too much warping, meaning the pieces could not be assembled together. Further research was done to improve the print quality, discussed below. The final iterations were done through the Cal Poly Innovation Sandbox, a campus club which provides free 3D printing services.
In an effort to improve the print quality, the websites of 3D printer manufacturers Ultimaker and Makerbot were consulted. This lead to the team switching from using acrylonitrile butadiene (ABS) to PLA. PLA has a lower shrinkage factor than ABS, making it less likely to warp as it cools down. In addition, the heated build plate temperature was increased so that it was at a temperature just below the glass transition temperature (solidifying point) of the plastic filament. This helped to prevent the individual layers of filament from contracting before the entire part had printed. A raft, or a thin layer of plastic between the build plate and part, was used to help distribute the heat more evenly to the part. Additionally, later iterations were printed at a cooler nozzle extrusion temperature, allowing for a smaller temperature difference between the filament being currently extruded and the filament already on the build plate, improving the interlayer adhesion. Later iterations were also printed with a lower infill, or density, which served to decrease the amount of internal tension exerted by the plastic as the part cooled.

To prevent delamination from occurring, the extruder head was calibrated to obtain a material flow rate in accordance with the 3D model slicer data. Delamination typically occurs when not enough deposited material is on the plate, so the adjacent threads are unable to stick to one another. The early iterations likely had a nozzle flow rate that was too low.

In order to improve the accuracy of angles in the print, angles were smoothed into curved features in the CAD model when possible. 3D printers often struggle to print angles accurately as the nozzle head slows down as it works its way through the corner and accidentally deposits too much material. Increasing the jerk, or speed which the nozzle moves through the corner, was a solution to improve the corner quality.
The brass heat set inserts were melted into the PLA structure with a soldering iron, seen in Figure 25. The soldering iron was applied to the brass insert, causing the PLA to locally melt, at which point the soldering iron was removed and the brass insert fell into the plastic hole. As the part cooled, the PLA reformed around the external grooves of the insert, locking it into place. A pair of tweezers was used to help guide the brass insert into place and make minute adjustments to its position. While manufacturing the first prototype, the brass inserts sometimes drifted left or right and failed to stay centered within their designated hole. The brass was so hot that it would melt the plastic all around it and often then fall sideways as it was pushed into place by the soldering iron, producing a brass insert that wasn’t flush with the surface of the housing. To prevent this in the later iterations, hole chamfers were included to help guide the brass into place, which proved to be a successful strategy.

![Figure 25: Brass heat set insert manufacturing process.](image)

The finger electrodes and a set of test electrodes were manufactured in house. The blanks were cut out of aluminum 6061 sheet metal using the abrasive water-jet cutter in the Industrial Technology Packaging Lab. After burr removal and buffing, the blanks were formed into shape. To facilitate soldered connections to the aluminum electrodes, the copper tabs were removed from a set of strain gauge terminals and fixed to each electrode using a small amount of conductive epoxy. The connection wires would then be soldered directly to the copper tabs. However, soldering these connections proved difficult and the conductive epoxy was used again to secure the wires. Eventually it was decided to use the purchased Biopac AG/AgCl as the primary chest electrodes. There were issues soldering these as well due to the low melting temp of the silver, but a solid connection was eventually made by melting the wire into the inside surface of the electrode. After completing assembly of the electrodes, they were connected to the board and epoxied into place on the casing.
Assembly

Before assembly of the casing, several of the 3D printed parts had to be filed down in order to reduce their height. After this had been done, the brass heat set inserts were soldered into place, components epoxied to the housing, and plastic housing parts inserted and screwed into place. The slots on the main part of the housing helped to guide the side panels into place.

Chapter 7: Design Verification

DVP&R

To validate the design, several different tests were conducted. The first test was an end-to-end testing to make sure that both the EKG device and the mobile app were functional when used together. The test first involved connecting to the EKG device. Then a user held the EKG device properly to his or her chest to get the heart rate signal. The app should be able to show a visible heart rate on the chart and export that result through email. Then the CSV file is opened in Excel to make sure the data closely matches the one shown on the app. Finally, the app is disconnected from the EKG. The end-to-end testing was done by the team and should not encounter any issues like freezing, crashing or EKG malfunction.

The end-to-end testing was mostly successful. The app is able to connect and disconnect wirelessly to the EKG device. The app is able to show a visible heart rate waveform, such as Figure 26. The result was sent through email and viewed on Excel sheet. The plot generated on the Excel matched the one on the app.
However, throughout the testing, the app encountered various crash or freeze issues due to the concurrent data array update and access for the chart. All of these known problems were resolved with correct thread setup to properly update UI in real time and prevent invalid access. But in rare occasion, the app still crashes due to an unknown issue.

The second test involved giving users the device and app and seeing if they could figure out how to use it without any prior instruction. If not, then the user interface and user experience may be too complicated. Of the five people tested, none had any issues turning on the device, collecting data, and emailing the results. However, all of these subjects were college-aged and further testing would need to be conducted with an older population sample.

In order to validate the thermal analysis, thermal tests were performed once the prototype construction had finished. A TEGAM 132F TRMS Multimeter with a thermocouple was used to measure the temperature. The thermocouple was placed inside one of the finger electrodes holes and the device was powered on for 5 minutes. The temperature was recorded every 30 seconds, and the test was repeated three times. The results can be seen in Figure 27. The app was connected to the device as well so that the device was operating in a maximum power-usage scenario. The test results showed the internal temperature reaching a maximum temperature of 77 degrees Fahrenheit, a 5 degree increase from the room-temperature value the device started at. The temperature appeared to be approaching an asymptote by the end of each thermal test, indicating that the temperature wouldn’t rise much more had the test gone on longer. However, 5 minutes is longer than the device will need to run for a typical
test. 77 degrees is far less than the critical temperature for the electronics and the PLA casing, as well as the “worst case scenario” thermal model calculation.

![Thermal Test Results](image)

**Figure 27:** Results from thermal tests showing temperature versus time as device is operated for a five minute period.

Finally, a drop test was not conducted as it was determined by inspection that the prototype would not survive any such test, making the test redundant and detrimental to the progress of the project.

**Chapter 8: Project Management**

This project involved three main components. Each member of the team was assigned to specific components of the project as denoted below:

- **EKG Filtering and PCB Development**: Galen Wu and Ryan Blalid focused on the electrical engineering aspects of the project, which included the EKG circuit design and hardware, PCB board layout, and analog-to-digital conversion of the data for the app.

- **Physical EKG Development**: Jim Kerman and Eric Burstedt focused on the mechanical engineering aspects of the project, which included the physical design of the EKG device such as the shape, size, materials and the development of the device to fit the EKG circuit board.

- **Mobile App Development**: Chin Chao focused on the software engineering aspects of the project, which included the mobile app design and development for viewing and sending the EKG data, and communication between the device and the mobile app through Bluetooth to receive EKG data.
All members participated in the design, manufacture, and testing of each component of the project for the test platforms and final prototype. This allowed everyone to stay up-to-date on all aspects of the project and each component to be successfully combined and interfaced with the others. Each quarter of the project was focused on different parts of the project design process.

Winter Quarter 2018 was focused on the research and preliminary design. The key deliverable for this quarter was a concept design of the form factor. The design idea was tested for feasibility with a concept prototype. The design was flexible to allow change if needed, but the team tried to limit it to only minor modifications so the project would not be set back by sudden changes.

The Spring Quarter 2018 was focused on the prototype design and the development of each component along with the Critical Design Review (Week 4). The key deliverables of this quarter were:

- **Critical Design Review** - April 2018
- **Structural prototype of the EKG device form** - May 2018
- **EKG filtering circuit design** - May 2018
- **Mobile app prototype** - June 2018

The structural prototype and EKG filtering design needed to be completed by May so the finalized device packaging could be determined and the PCB and all physical casing parts could be ordered and completely manufacturing before the end of the quarter. The risk and safety concerns were taken into account, such as the standard circuit safety design for the PCB, so the final device is safe for users.

Design of the PCB ended up taking longer than expected, delaying the final packaging design as well as PCB manufacture until early Fall. Spring quarter was also used for exploring the mobile app development including the Bluetooth interface and whether a cross-platform or Android/IOS app would be preferable. The basic prototype including the user interface and basic data transmission was completed by the end of this quarter.

Fall Quarter 2018 was focused on the final design and development of each component and the final design review (Week 9). The key deliverables of this quarter were:

- **EKG filtering circuit test** - 4th week (10/20/18)
- **EKG device assembly** - 4th week (10/20/18)
- **Complete preliminary mobile app** - 6th week (10/28/18)
- **Complete EKG adafruit development** - 6th week (10/28/18)
- **Hardware Testing (Interface adafruit arduino with mobile app)** - 7th week (11/04/18)
- **Complete EKG device + final mobile app** - Thanksgiving Week (11/18/18)
- **Final hardware test + senior project expo presentation** - 9th week (11/25/18)
The basic PCB board arrived early Fall quarter after the design work was completed over the summer. However, significant roadblocks in the installation of the hardware due to inadequate tooling delayed and ultimately suspended the use of the PCB in our final prototype. The PCB may be completed at a later date. Circuit tests were conducted with the development board during week 4. The setbacks to the PCB also delayed the manufacture of the final casing prototype from week 4 to week 7. The final EKG device was assembled and completed final testing in week 9. The mobile app features were completed by week 5 so all the app features including Bluetooth communication could be tested in week 6. Week 7 and 8 were used to polish the app and remove bugs. Full testing of the app and device together occurred in week 9 for the Final Hardware Test. The device, with one lead and the development board, was ready for the Project Expo.

![Figure 28. Front side of completed Final Prototype](image)

**Chapter 9: Conclusion and Recommendations**

As discussed with Dr. Nooristani, the main purpose of this project was to create a functional prototype of a handheld self-administered EKG device that can be used by patients at any place without professional assistance. The device was to provide an efficient way for patients to monitor their heart rate constantly while still allowing hospital staff to monitor the heart rate through information sent by the mobile app.

After comparing the research from the existing EKG products and the basic requirements for the prototype, an EKG device with a five-electrode six-lead design and wireless Bluetooth connection to a
mobile app that received and displayed the EKG data was decided to be the best approach. The device and app had an emphasis on a simple but effective interface for the patients to have ease of access to their heart data.

The final product ended up being a three-electrode device. However, at the time of writing the report, the PCB was not completed due to the difficulty in manufacturing and in assembling the device. Hence, the current prototype can only handle a single lead. Manufacturing challenges were also encountered with producing the finger electrodes. When it became apparent that the PCB would not be finished in time, work on the finger electrodes was stopped in order to focus on other aspects of the project.

The final product is somewhat user friendly - the LED turns green when the device is on and becomes yellow when the battery starts running low. The wrist-strap helps to prevent users from accidentally dropping the device. After explaining to users that the electrodes on the front of the device were supposed to be placed over the heart, the majority of people were able to figure out by themselves how to properly hold the device.

The simple layout of the app made it so everyone this team asked to use the device were able to successfully connect the app via Bluetooth to the device, collect the data, and send the email with the data at the end. Although the user interface of the app is somewhat sparse, its simple layout makes it easy to use. There were some unresolved bugs in the app that caused it to crash. More work is required to prevent this from being an ongoing issue. While a moving average was used to filter the incoming data, a cardiologist would be needed to verify that this is the best way to filter the data to provide the clearest signal.

If more iterations of the prototype were carried out, the overall size of the device would be reduced. Additionally, the placement of the finger electrodes could be readjusted further. Many people had varying opinions on the best placement of the finger electrodes so that an elderly patient with movement issues could still easily use the device. Perhaps a limitation of this team was not being able to thoroughly understand how a mobility-impaired individual would be able position their arms or hands.

While the 3D printed structure provided a good structure for the prototype, using a plastic extrusion process would provide much better accuracy, lower costs, and quicker manufacturing. However, there is also a significant up-front cost and lead time associated with producing the dies needed for plastic extrusion. Mass production of this device would make this a viable and attractive option.

As the project proceeded, it became apparent that the biggest challenge would be with designing and manufacturing the PCB. As the design proceeded, issues with the PCB kept cropping up, causing the project to be delayed by several weeks. However, designing the PCB was the most complicated and
crucial aspect of the project, so doing it correctly was the main focus. If more time could have been allocated for the design portion of the project, then later parts wouldn’t have been delayed as much.

One of the issues from the design of the PCB was picking a fabricator that aligned with our timeline and meeting their design for manufacturing (DFM) requirements. The team decided to go with Bay Area Circuits as the fabricator because they are local to Ryan for him to pickup and inspect the board in person. They had strict DFM requirements that proved to be difficult to reach. A lot of our design had to be changed or tinkered with in order to meet them. This process took longer than the team expected delaying the project by at least two weeks.

Once the board was successfully fabricated, another issue was the construction of it. The current equipment provided to us did not meet our needs. Fortunately, we did have access to a microscope and reflow oven which was useful during the manufacturing phase. When surface mounting the components to the board, some of the components would also need to be soldered directly. This was extremely difficult because of our large solder tips. Because of our lack of expertise and equipment, we were not able to successfully solder the surface mount components directly to the board. It was then decided for Ryan to take the incomplete board and to finish building it at his workplace, Apple, where they have more precise and specialized equipment.

To continue to improve upon this project, more work would need to be done on assembling the PCB. However, due to the precision and equipment required for this, a skilled technician would likely be required to carry this out.

The team will be turning over all project materials to Ryan Blaalid, who will continue to work on it during Winter Quarter 2019.

Overall, this team was able to design and build a device that could wirelessly transmit EKG data to a phone, from where it could be emailed to a physician. Although there were shortcomings in the project, many important goals were met. The project served as a strong learning experience for those involved. This team is thankful for the support of Dr. Nooristani and Dr. Elghandour for giving their time and resources to help the team through the design process, from beginning to end.

References


[8] N Meziane et al 2013 Physiological Measurement 34 R47

Appendices
[A] QFD House of Quality
[B] Decision Matrix
[C] EKG Code
[D] Initial Analysis
[E] App Code
[F] Concept Layout Drawing
[G] Complete Drawings Package
[H] Purchased Parts Detail
[I] Budget Procurement List
[K] Safety Hazard Checklist, FMEA
[L] Gantt Chart
# Appendix [A]: QFD House of Quality

## QFD House of Quality

**FDR: Wireless EKG Device**

**Date:** 12/06/18

### Table: QFD House of Quality

<table>
<thead>
<tr>
<th>Customer Requirements</th>
<th>Technical Requirements</th>
<th>Quality Function Deployment (QFD)</th>
<th>Environmental Factors</th>
<th>Leadership Factors</th>
<th>Overall Importance</th>
<th>Performance Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Battery Life</td>
<td>Power Efficiency</td>
<td>Power Consumption</td>
<td></td>
<td>Safety</td>
<td>2</td>
<td>0.8</td>
</tr>
<tr>
<td>Comfort</td>
<td>Comfort</td>
<td>Comfort</td>
<td></td>
<td>Design</td>
<td>3</td>
<td>0.9</td>
</tr>
<tr>
<td>Accuracy</td>
<td>Accuracy</td>
<td>Accuracy</td>
<td></td>
<td>Usability</td>
<td>4</td>
<td>1.0</td>
</tr>
<tr>
<td>Ease of Use</td>
<td>Ease of Use</td>
<td>Ease of Use</td>
<td></td>
<td>Maintainability</td>
<td>5</td>
<td>1.2</td>
</tr>
</tbody>
</table>

### Diagram: QFD Matrix

The diagram illustrates the relationship between customer requirements, technical requirements, and overall importance, showing how each requirement affects the product design and its performance.

---

**UNDA**

FDR: Wireless EKG Device

12/06/18 Page [40]
Appendix [B]: Decision Matrix

Team: Wireless EKG
Members: Jim Kerman, Galen Wu, Chin Chao

Table 1: Weighted Decision Matrix of EKG Form

<table>
<thead>
<tr>
<th>Forms</th>
<th>Size</th>
<th>Usability</th>
<th>Accuracy</th>
<th>Accessibility</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Box with Clamps</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>Bow with Sockets</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>27</td>
</tr>
<tr>
<td>Box with Pads</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>21</td>
</tr>
<tr>
<td>Wrist and Handle</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>24</td>
</tr>
<tr>
<td>Weight</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

It was determined to use the Box with Sockets design from the Weighted Decision Matrix.
/*
 * @Author: Galen Wu and Ryan Blaalid
 * @Date: 2018-04-27 21:49:09
 * @Last Modified time: 2018-04-27 22:18:48
 * *
 * Description:
 * Helper functions to init, read, and write
 * to the MAX30003
 */

#include <arduino.h>
#include <SPI.h>
#include "MAX30003.h"

void MAX30003_begin(void){
    max30003_sw_reset();

    MAX30003_Reg_Write(CNFG_GEN, 0x081217); // 0x081007
    delay(100);
    MAX30003_Reg_Write(CNFG_ECG, 0x835000); // d23 - d22 : 10 for 250sps, 00:
    delay(100);

    MAX30003_Reg_Write(CNFG_RTOR1, 0x3FC600);
    delay(100);

    MAX30003_Reg_Write(MNGR_INT, 0x180014);
    delay(100);

    MAX30003_Reg_Write(CNFG_CAL, 0x720000);
    delay(100);

    MAX30003_Reg_Write(EN_INT, 0x800003);
    delay(100);

    MAX30003_Reg_Write(MNGR_DYN, 0x3F0000);
    delay(100);

    MAX30003_Reg_Write(CNFG_EMUX, 0x000000);
    delay(100);

    max30003_synch();
}

void max30003_synch(void){
    MAX30003_Reg_Write(SYNCH, 0x000000);
    delay(100);

}

void max30003_sw_reset(void){
    MAX30003_Reg_Write(SW_RST, 0x000000);
    delay(100);
}

uint32_t MAX30003_Reg_Read(uint8_t regAddr){
    uint32_t data;
    uint8_t addr = (regAddr << 1) | RREG;

digitalWrite(MAX30003_CS_PIN, LOW);

SPI.transfer(addr); //Send register location

for (int shift = 16; shift >= 0; shift -= 8){
    data |= (SPI.transfer(0xFF) << shift);
}

digitalWrite(MAX30003_CS_PIN, HIGH);
    return data;
}

void MAX30003_Reg_Write(uint8_t writeAddr, uint32_t data){
    // now combine the register address and the command into one byte:
    uint8_t dataToSend = (writeAddr << 1) | WREG;

    // take the chip select low to select the device:
digitalWrite(MAX30003_CS_PIN, LOW);

delay(2);
SPI.transfer(dataToSend); //Send register location
SPI.transfer((UPPER_BYTE_MASK & data) >> 16); //number of register to write
SPI.transfer((MID_BYTE_MASK & data) >> 8); //number of register to write
SPI.transfer(LOWER_BYTE_MASK & data); //Send value to record index

delay(2);

    // take the chip select high to de-select:
digitalWrite(MAX30003_CS_PIN, HIGH);
}
#ifndef MAX30003_H_
#define MAX30003_H_

#include <arduino.h>

#define SERIAL_BAUD_RATE 115200

#define UPPER_BYTE_MASK 0x00FF0000
#define MID_BYTE_MASK 0x0000FF00
#define LOWER_BYTE_MASK 0x000000FF

#define EINT_STATUS_MASK (1 << 23)
#define FIFO_OVF_MASK 0x07
#define FIFO_VALID_SAMPLE_MASK 0x00
#define FIFO_FAST_SAMPLE_MASK 0x01
#define ETAG_BITS_MASK 0x07

#define WREG 0x00
#define RREG 0x01

#define MAX30003_CS_PIN 9
#define EINT_PIN 10

/* MAX30003 Registers Addresses */
#define NO_OP 0x00
#define STATUS 0x01
#define EN_INT 0x02
#define EN_INT2 0x03
#define MNGR_INT 0x04
#define MNGR_DYN 0x05
#define SW_RST 0x08
#define SYNCH 0x09
#define FIFO_RST 0x0A
#define INFO 0x0F
#define CNFG_GEN 0x10
#define CNFG_CAL 0x12
#define CNFG_EMUX 0x14
#define CNFG_ECG 0x15
#define CNFG_RTOR1 0x1D
#define CNFG_RTOR2 0x1E
#define ECG_FIFO_BURST 0x20
#define ECG_FIFO 0x21
#define RTOR 0x25
#define NO_OP 0x7F
void MAX30003_begin(void);
void max30003_synch(void);
void max30003_sw_reset(void);
uint32_t MAX30003_Reg_Read(uint8_t);
void MAX30003_Reg_Write(uint8_t, uint32_t);

#endif
/*
 * @Author: Galen Wu and Ryan Blaalid
 * @Date:   2018-04-27 21:49:09
 * @Last Modified time: 2018-04-27 21:56:39
 *
 * Description:
 * Setup and main loop for the M0 Feather board
 * and MAX30003 IC used in our EKG Senior Project
 */

#include <arduino.h>
#include <SPI.h>
#include "MAX30003.h"

int eint_flag = 0;

void EINT_ISR(){
  eint_flag = 1;
}

void setup(){
  Serial.begin(SERIAL_BAUD_RATE);      //Serial begin
  pinMode(MAX3003_CS_PIN, OUTPUT);
  digitalWrite(MAX3003_CS_PIN, HIGH); //disable device

  SPI.begin();
  SPI.setBitOrder(MSBFIRST);
  SPI.setDataMode(SPI_MODE0);
  SPI.setClockDivider(SPI_CLOCK_DIV4);

  MAX3003_begin();

  pinMode(EINT_PIN, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(EINT_PIN), EINT_ISR, FALLING);
}

void loop(){
  uint32_t ecgFIFO = 0, readECGSamples = 0, idx = 0, ETAG[32], status = 0;
  int16_t ecgSample[32];

  while (true){

    while (eint_flag){
      eint_flag = 0;


status = MAX30003_Reg_Read(STATUS);  // Read the STATUS register

// Check if EINT interrupt asserted
if ((status & EINT_STATUS_MASK) == EINT_STATUS_MASK){
  readECGSamples = 0;  // Reset sample counter

  do {
    ecgFIFO = MAX30003_Reg_Read(ECG_FIFO);
    ecgSample[readECGSamples] = ecgFIFO >> 8;
    ETAG[readECGSamples] = (ecgFIFO >> 3) & ETAG_BITS_MASK;
    readECGSamples++;

    // Check that sample is not last sample in FIFO
  } while (ETAG[readECGSamples - 1] == FIFO_VALID_SAMPLE_MASK ||
           ETAG[readECGSamples - 1] == FIFO_FAST_SAMPLE_MASK);

  // Check if FIFO has overflowed
  if(ETAG[readECGSamples - 1] == FIFO_OVF_MASK){
      MAX30003_Reg_Write(FIFO_RST, 0);  // Reset FIFO
  }

  // Print results
  for(idx = 0; idx < readECGSamples; ++idx){
      Serial.print("sample: ");
      Serial.println(ecgSample[idx]);
  }
}
}
Hand Calculations

Assume Four Parts: (1) Convection from electronics to ambient air (2) Convection from ambient air to inner wall of housing (3) Conduction of heat from inner surface of housing to outer surface. (4) Convection from outer surface to air in surroundings.

\[ Q = hA(T_4 - T_5) \]

\[ Q = kA(T_3 - T_4)/L \]

Inner Surface Areas | Outer Surface Areas

| A (front) \([\text{m}^2]\) | 0.01161288 | 0.01548384 |
| A (back) \([\text{m}^2]\)  | 0.00290322 | 0.00516128 |
| A (left) \([\text{m}^2]\)  | 0.00290322 | 0.00516128 |
| A (right) \([\text{m}^2]\)  | 0.01741932 | 0.00860213 |
| A (total) \([\text{m}^2]\)  | 0.01741932 | 0.00860213 |

Part (4): \[ Q = hA(T_4 - T_5) \]

\[ Q = 0.5 \text{ W} \]

\[ h = 10 \text{ W/(m}^2 \text{ K)} \]

\[ A \text{ (outer)} = 0.00860213 \text{ m}^2 \]

\[ T_5 = 298 \text{ K} \]

\[ T_4 = 303.8 \text{ K} \]

Part (3): \[ Q = kA(T_3 - T_4)/L \]

\[ k = 0.1 \text{ W/(m K)} \]

\[ A \text{ (inner)} = 0.01741932 \text{ m}^2 \]

\[ L = 0.0635 \text{ m} \]

\[ T_4 = 303.8 \text{ K} \]
<table>
<thead>
<tr>
<th>Part (2): $Q = hA(T_2 - T_3)$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$ (inner) = 0.01741932 m$^2$</td>
<td></td>
</tr>
<tr>
<td>$h = 10$ W/(m$^2$ K)</td>
<td></td>
</tr>
<tr>
<td>$T_2 = 308.5$ K</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Part (1): $Q = hA(T_1 - T_2)$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$ (electronics) = 0.00548386 m$^2$</td>
<td></td>
</tr>
<tr>
<td>$h = 10$ W/(m$^2$ K)</td>
<td></td>
</tr>
<tr>
<td>$T_1 = 317.6$ K</td>
<td></td>
</tr>
<tr>
<td>$T_1 = 44.6$ C</td>
<td></td>
</tr>
<tr>
<td>$T_1 = 112.3218353$ F</td>
<td></td>
</tr>
</tbody>
</table>
import UIKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {
        // Override point for customization after application launch.
        return true
    }

    func applicationWillResignActive(_ application: UIApplication) {
        // Sent when the application is about to move from active to inactive
        // state. This can occur for certain types of temporary interruptions
        // (such as an incoming phone call or SMS message) or when the user
        // quits the application and it begins the transition to the background
        // state. Use this method to pause ongoing tasks, disable timers, and
        // invalidate graphics rendering callbacks. Games should use this method
        // to pause the game.
    }

    func applicationDidEnterBackground(_ application: UIApplication) {
        // Use this method to release shared resources, save user data, invalidate
        // timers, and store enough application state information to restore your
        // application to its current state in case it is terminated later.
        // If your application supports background execution, this method is called
        // instead of applicationWillTerminate: when the user quits.
    }

    func applicationWillEnterForeground(_ application: UIApplication) {
        // Called as part of the transition from the background to the active state;
        // here you can undo many of the changes made on entering the background.
    }

    func applicationDidBecomeActive(_ application: UIApplication) {
        // Restart any tasks that were paused (or not yet started) while the
        // application was inactive. If the application was previously in the
        // background, optionally refresh the user interface.
    }

    func applicationWillTerminate(_ application: UIApplication) {
        // Called when the application is about to terminate. Save data if
        // appropriate. See also applicationWillEnterForeground:.
    }
}
import CoreBluetooth

// Universally Unique Identifier for Bluetooth Low Energy
// for Adafruit Feather M0 Bluefruit
let kBLEService_UUID = "6e400001-b5a3-f393-e0a9-e50e24dcca9e"
let kBLE_Characteristic_uuid_Tx = "6e400002-b5a3-f393-e0a9-e50e24dcca9e"
let kBLE_Characteristic_uuid_Rx = "6e400003-b5a3-f393-e0a9-e50e24dcca9e"
let MaxCharacters = 20

let BLEService_UUID = CBUUID(string: kBLEService_UUID)

let BLE_Characteristic_uuid_Tx = CBUUID(string: kBLE_Characteristic_uuid_Tx)

let BLE_Characteristic_uuid_Rx = CBUUID(string: kBLE_Characteristic_uuid_Rx)
1 // Moving Average for Signal Filtering
2 class MovingAverage {
3     var samples: Array<Double> // Sample array
4     var sampleCount = 0 // Total sample gathered already
5     var period = 5 // Sample period (array size) for moving average calculation
6
7     // Initialize moving average filter with specified sample period
8     // Default is period of 5 points
9     init(period: Int = 5) {
10         self.period = period
11         samples = Array<Double>()
12     }
13
14     // Average of the current samples
15     var average: Double {
16         let sum: Double = samples.reduce(0, +)
17
18         if period > samples.count { // array not fill up yet
19             return sum / Double(samples.count)
20         } else { // array fills up to period already
21             return sum / Double(period)
22         }
23     }
24
25     // Add new sample into the array and return the moving average
26     func addSample(value: Double) -> Double {
27         let pos = sampleCount % period // get next sample position in array
28         sampleCount = sampleCount + 1 // increment total sample count
29
30         if pos >= samples.count { // array not fill up yet
31             samples.append(value)
32         } else { // replace old sample
33             samples[pos] = value
34         }
35
36         return average
37     }
38
39     // Restart the moving average samples
40     func reset() {
41         samples.removeAll()
42         sampleCount = 0
43     }
44 }
import Foundation
import UIKit
import CoreBluetooth

var txCharacteristic : CBCharacteristic?
var rxCharacteristic : CBCharacteristic?
var blePeripheral : CBPeripheral?
var heartrates = [UInt8]()

class BLECentralViewController : UIViewController, CBCentralManagerDelegate, CBPeripheralDelegate {

    // Data
    var centralManager : CBCentralManager!
    var data = NSMutableData()
    var peripherals: [CBPeripheral] = []
    var timer = Timer()
    var characteristics = [String : CBCharacteristic]()

    // UI
    @IBOutlet weak var refreshButton: UIBarButtonItem!
    @IBOutlet weak var connectButton: UIButton!
    @IBOutlet weak var pendingIndicator: UIActivityIndicatorView!

    @IBAction func refreshAction(_ sender: AnyObject) {
        disconnectFromDevice()
        pendingIndicator.stopAnimating()
        self.peripherals = []
        startScan()
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        connectButton.isHidden = true
        pendingIndicator.stopAnimating()

        /* CBCentralManager objects are used to manage discovered or connected remote
         peripheral devices (represented by CBPeripheral objects), including
         scanning for, discovering, and connecting to advertising peripherals. */
        centralManager = CBCentralManager(delegate: self, queue: nil)
        let backButton = UIBarButtonItem(title: "Disconnect", style: .plain,
                                         target: nil, action: nil)
        navigationItem.backBarButtonItem = backButton
    }

    override func viewDidAppear(_ animated: Bool) {
        disconnectFromDevice()
        super.viewDidAppear(animated)
        refreshScanView()
        print("View Cleared")
    }

    override func viewWillDisappear(_ animated: Bool) {
        super.viewWillDisappear(animated)
        print("Stop Scanning")
        centralManager?.stopScan()
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        connectButton.isHidden = true
        pendingIndicator.stopAnimating()

        /* CBCentralManager objects are used to manage discovered or connected remote
         peripheral devices (represented by CBPeripheral objects), including
         scanning for, discovering, and connecting to advertising peripherals. */
        centralManager = CBCentralManager(delegate: self, queue: nil)
        let backButton = UIBarButtonItem(title: "Disconnect", style: .plain,
                                         target: nil, action: nil)
        navigationItem.backBarButtonItem = backButton
    }

    override func viewDidAppear(_ animated: Bool) {
        disconnectFromDevice()
        super.viewDidAppear(animated)
        refreshScanView()
        print("View Cleared")
    }

    override func viewWillDisappear(_ animated: Bool) {
        super.viewWillDisappear(animated)
        print("Stop Scanning")
        centralManager?.stopScan()
    }
}
func startScan() {
    peripherals = []
    connectButton.isHidden = true
    print("Now Scanning...")
    self.timer.invalidate()
    centralManager?.scanForPeripherals(withServices: [BLEService_UUID], options: [CBCentralManagerScanOptionAllowDuplicatesKey: false])
    Timer.scheduledTimer(timeInterval: 17, target: self, selector: #selector(self.cancelScan), userInfo: nil, repeats: false)
}

/* Stop scanning for peripherals */
@objc func cancelScan() {
    self.centralManager?.stopScan()
    print("Scan Stopped")
    print("Number of Peripherals Found: \(peripherals.count)")
}

/* Refresh the connect screen */
func refreshScanView() {
    pendingIndicator.stopAnimating()
    connectButton.isHidden = false
}

// Terminate all Peripheral Connection
/* Cancels any subscriptions if there are any, or straight disconnects if not. */
func disconnectFromDevice () {
    if blePeripheral != nil {
        centralManager?.cancelPeripheralConnection(blePeripheral!)
    }
}

/* Restores Central Manager delegate if something went wrong */
func restoreCentralManager() {
    centralManager?.delegate = self
}

/* Central manager discovers a peripheral while scanning. */
func centralManager(_ central: CBCentralManager, didDiscover peripheral: CBPeripheral, advertisementData: [String : Any], rssi RSSI: NSNumber) {
    blePeripheral = peripheral
    self.peripherals.append(peripheral)
    peripheral.delegate = self
    if blePeripheral != nil {
        print("Found new pheripheral devices with services")
        print("Peripheral name: \(String(describing: peripheral.name))")
        print("****************************")
        print ("Advertisement Data : \(advertisementData)")
    }
    connectButton.isHidden = false
}

// Connect to device
func connectToDevice () {

connectButton.isHidden = true
pendingIndicator.startAnimating()
remote?.connect(blePeripheral!, options: nil)
}

// A connection with a peripheral is successful. */
func centralManager(_ central: CBCentralManager, didConnect peripheral: CBPeripheral) {
    print("*****************************")
    print("Connection complete")
    print("Peripheral info: \\
        \(String(describing: blePeripheral))")
    // Stop scanning
    centralManager?.stopScan()
    print("Scan Stopped")
    // Erase data that we might have
    data.length = 0
    // Discovery callback
    peripheral.delegate = self
    // Only look for services that matches transmit uuid
    peripheral.discoverServices([BLEService_UUID])
    // Move to new view controller to manage incoming heart rate data
    let storyboard = UIStoryboard(name: "Main", bundle: nil)
    let uartViewController = storyboard.instantiateViewController(withIdentifier: "UartModuleViewController") as! UartModuleViewController
    uartViewController.peripheral = peripheral
    navigationController?.pushViewController(uartViewController, animated: true)
}

// Central manager fails to create a connection with a peripheral. */
func centralManager(_ central: CBCentralManager, didFailToConnect peripheral: CBPeripheral, error: Error?) {
    if error != nil {
        let alertVC = UIAlertController(title: "Connection Failure", message: "EKG Bluetooth Connection Failure, Please Retry", preferredStyle: UIAlertControllerStyle.alert)
        let action = UIAlertAction(title: "OK", style: UIAlertActionStyle.default, handler: { (action: UIAlertAction) -> Void in
            self.dismiss(animated: true, completion: nil)
        })
        alertVC.addAction(action)
        present(alertVC, animated: true, completion: nil)
        return
    }
}

// Discover the peripheral’s available services. */
func peripheral(_ peripheral: CBPeripheral, didDiscoverServices error: Error?) {
    if ((error) != nil) {
        print("Error discovering services: \\
            \(error!.localizedDescription)")
        return
    }
    guard let services = peripheral.services else {

// We need to discover the all characteristic
for service in services {
    peripheral.discoverCharacteristics(nil, for: service)
}
print("Discovered Services: \(services)")
}

/* Discover the characteristics of a specified service. */
func peripheral(_ peripheral: CBPeripheral, didDiscoverCharacteristicsFor
service: CBService, error: Error?) {
    print("*******************************************************")
    if ((error) != nil) {
        print("Error discovering services: \(error!.localizedDescription)")
        return
    }
    guard let characteristics = service.characteristics
    else {
        return
    }
    print("Found \(characteristics.count) characteristics!")
    for characteristic in characteristics {
        // looks for the right characteristic
        if characteristic.uuid.isEqual(BLE_Characteristic_uuid_Rx) {
            rxCharacteristic = characteristic
            // Subscribe to the this particular characteristic
            peripheral.setNotifyValue(true, for: rxCharacteristic!)
            peripheral.readValue(for: characteristic)
            print("Rx Characteristic: \(characteristic.uuid)")
        }
        if characteristic.uuid.isEqual(BLE_Characteristic_uuid_Tx){
            txCharacteristic = characteristic
            print("Tx Characteristic: \(characteristic.uuid)")
        }
        peripheral.discoverDescriptors(for: characteristic)
    }
}
/* Getting Values From Characteristic */
func peripheral(_ peripheral: CBPeripheral, didUpdateValueFor characteristic:
CBBCharacteristic, error: Error?) {
    guard let data = characteristic.value
    else {
        print("Value not updated!")
        return
    }
    if characteristic == rxCharacteristic {
        heartrates = [UInt8](data)
        NotificationCenter.default.post(name: NSNotification.Name(rawValue:"Notify"), object: nil)
    }
}
/* Discover the descriptors for specific characteristic */
func peripheral(_ peripheral: CBPeripheral, didDiscoverDescriptorsFor
characteristic: CBBCharacteristic, error: Error?) {

if error != nil {
    print("
    \\
    \(error.debugDescription)"
    return
}

if ((characteristic.descriptors) != nil) {
    for x in characteristic.descriptors!{
        let descript = x as! CBDescriptor?
        print("function name: DidDiscoverDescriptorForChar ",
               "\(String(describing: descript?.description))")
        print("Rx Value \(String(describing: rxCharacteristic?.value))")
        print("Tx Value \(String(describing: txCharacteristic?.value))")
    }
}

/* Update notification state for a specific characteristic */
func peripheral(_ peripheral: CBPeripheral, didUpdateNotificationStateFor
characteristic: CBCharacteristic, error: Error?) {
    print("*******************************************************")
    if (error != nil) {
        print("Error changing notification state:",
               "\(String(describing: error?.localizedDescription))")
    } else {
        print("Characteristic's value subscribed")
    }
    if (characteristic.isNotifying) {
        print("Subscribed. Notification has begun for: \(characteristic.uuid)")
    }
}

/* Central manager disconnects from the peripheral */
func centralManager(_ central: CBCentralManager, didDisconnectPeripheral
peripheral: CBPeripheral, error: Error?) {
    print("Disconnected")
}

/* Write valu for a specific characteristic */
func peripheral(_ peripheral: CBPeripheral, didWriteValueFor characteristic:
CBCharacteristic, error: Error?) {
    guard error == nil else {
        print("Error discovering services: error")
        return
    }
    print("Message sent")
}

/* Write value for a specific descriptor */
func peripheral(_ peripheral: CBPeripheral, didWriteValueFor descriptor:
CBDescriptor, error: Error?) {
    guard error == nil else {
        print("Error discovering services: error")
        return
    }
    print("Succeeded!")
@IBAction func startConnect(sender: AnyObject) {
    if peripherals.count != 0 {
        blePeripheral = peripherals[0]
        connectToDevice()
    } else {
        let alertVC = UIAlertController(title: "No Connection Available",
                                        message: "There is no EKG connection currently available!",
                                                preferredStyle: .alert)
        let action = UIAlertAction(title: "OK", style: .default,
                                    handler: { (action: UIAlertAction) -> Void in
                                                self.dismiss(animated: true, completion: nil)
                                    })
        alertVC.addAction(action)
        present(alertVC, animated: true, completion: nil)
    }
}

/* Central manager’s state is updated. Ex: Bluetooth turns on or off */
func centralManagerDidUpdateState(_ central: CBCentralManager) {
    if central.state == .poweredOn { // Bluetooth is on
        print("Bluetooth Enabled")
        startScan()
    } else { // Bluetooth is off or in other states
        print("Bluetooth Disabled- Make sure your Bluetooth is turned on")
        let alertVC = UIAlertController(title: "Bluetooth is not enabled",
                                        message: "Make sure that your bluetooth is turned on",
                                                preferredStyle: .alert)
        let action = UIAlertAction(title: "ok", style: .default,
                                    handler: { (action: UIAlertAction) -> Void in
                                                self.dismiss(animated: true, completion: nil)
                                    })
        alertVC.addAction(action)
        self.present(alertVC, animated: true, completion: nil)
    }
}
import UIKit
import CoreBluetooth
import Charts
import MessageUI

class UartModuleViewController: UIViewController, CBPeripheralManagerDelegate, MFMailComposeViewControllerDelegate {

    // UI
    @IBOutlet weak var lineChart: LineChartView!
    @IBOutlet weak var emailButton: UIButton!
    @IBOutlet weak var startStopButton: UIButton!
    @IBOutlet weak var scrollView: UIScrollView!

    // Data
    var peripheralManager: CBPeripheralManager?
    var peripheral: CBPeripheral!
    private let interval = 640 // Max data points shown on chart
    private var numbers = [Double]() // Heart rate data array
    private var pause = true // Pause data gathering

    // thread safety queue for add and access heart rate data and update chart
    private let queue = DispatchQueue(label: "thread-safe", attributes: .concurrent)
    private let avg = MovingAverage(period: 5) // Moving Average Setup (Period of 5 data)

    override func viewDidLoad() {
        super.viewDidLoad()

        pause = true
        startStopButton.setTitle("Start", for: .normal)
        startStopButton.backgroundColor = .green
        startStopButton.setTitleColor(.red, for: .normal)
        emailButton.isHidden = true

        // Line chart initial setup
        lineChart.xAxis.labelTextColor = .green
        lineChart.xAxis.granularityEnabled = true
        lineChart.xAxis.granularity = 1
        lineChart.leftAxis.labelTextColor = .green
        lineChart.leftAxis.drawZeroLineEnabled = true
        lineChart.rightAxis.labelTextColor = .green
        lineChart.legend.textColor = .red
        lineChart.chartDescription?.text = "Heart Rate Monitor"
        lineChart.chartDescription?.textColor = .red
        avg.reset()

        self.navigationItem.backBarButtonItem = UIBarButtonItem(title: "Back", style: .plain, target: nil, action: nil)
        // Create and start the peripheral manager
        peripheralManager = CBPeripheralManager(delegate: self, queue: nil)
        // Notification for updating the text view with incoming text
        updateIncomingData()
    }

    override func viewWillAppear(_ animated: Bool) {
    }

    override func viewDidAppear(_ animated: Bool) {
    }

    override func viewDidDisappear(_ animated: Bool) {
        peripheralManager?.stopAdvertising()
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        pause = true
        startStopButton.setTitle("Start", for: .normal)
        startStopButton.backgroundColor = .green
        startStopButton.setTitleColor(.red, for: .normal)
        emailButton.isHidden = true

        // Line chart initial setup
        lineChart.xAxis.labelTextColor = .green
        lineChart.xAxis.granularityEnabled = true
        lineChart.xAxis.granularity = 1
        lineChart.leftAxis.labelTextColor = .green
        lineChart.leftAxis.drawZeroLineEnabled = true
        lineChart.rightAxis.labelTextColor = .green
        lineChart.legend.textColor = .red
        lineChart.chartDescription?.text = "Heart Rate Monitor"
        lineChart.chartDescription?.textColor = .red
        avg.reset()

        self.navigationItem.backBarButtonItem = UIBarButtonItem(title: "Back", style: .plain, target: nil, action: nil)
        // Create and start the peripheral manager
        peripheralManager = CBPeripheralManager(delegate: self, queue: nil)
        // Notification for updating the text view with incoming text
        updateIncomingData()
    }

    override func viewDidLoad() {
    }

    override func viewDidAppear(_ animated: Bool) {
    }

    override func viewDidDisappear(_ animated: Bool) {
        peripheralManager?.stopAdvertising()
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        pause = true
        startStopButton.setTitle("Start", for: .normal)
        startStopButton.backgroundColor = .green
        startStopButton.setTitleColor(.red, for: .normal)
        emailButton.isHidden = true

        // Line chart initial setup
        lineChart.xAxis.labelTextColor = .green
        lineChart.xAxis.granularityEnabled = true
        lineChart.xAxis.granularity = 1
        lineChart.leftAxis.labelTextColor = .green
        lineChart.leftAxis.drawZeroLineEnabled = true
        lineChart.rightAxis.labelTextColor = .green
        lineChart.legend.textColor = .red
        lineChart.chartDescription?.text = "Heart Rate Monitor"
        lineChart.chartDescription?.textColor = .red
        avg.reset()

        self.navigationItem.backBarButtonItem = UIBarButtonItem(title: "Back", style: .plain, target: nil, action: nil)
        // Create and start the peripheral manager
        peripheralManager = CBPeripheralManager(delegate: self, queue: nil)
        // Notification for updating the text view with incoming text
        updateIncomingData()
    }

    override func viewWillAppear(_ animated: Bool) {
    }

    override func viewDidAppear(_ animated: Bool) {
    }

    override func viewDidDisappear(_ animated: Bool) {
        peripheralManager?.stopAdvertising()
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        pause = true
        startStopButton.setTitle("Start", for: .normal)
        startStopButton.backgroundColor = .green
        startStopButton.setTitleColor(.red, for: .normal)
        emailButton.isHidden = true

        // Line chart initial setup
        lineChart.xAxis.labelTextColor = .green
        lineChart.xAxis.granularityEnabled = true
        lineChart.xAxis.granularity = 1
        lineChart.leftAxis.labelTextColor = .green
        lineChart.leftAxis.drawZeroLineEnabled = true
        lineChart.rightAxis.labelTextColor = .green
        lineChart.legend.textColor = .red
        lineChart.chartDescription?.text = "Heart Rate Monitor"
        lineChart.chartDescription?.textColor = .red
        avg.reset()

        self.navigationItem.backBarButtonItem = UIBarButtonItem(title: "Back", style: .plain, target: nil, action: nil)
        // Create and start the peripheral manager
        peripheralManager = CBPeripheralManager(delegate: self, queue: nil)
        // Notification for updating the text view with incoming text
        updateIncomingData()
    }

    override func viewDidLoad() {
    }

    override func viewDidAppear(_ animated: Bool) {
    }

    override func viewDidDisappear(_ animated: Bool) {
        peripheralManager?.stopAdvertising()
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        pause = true
        startStopButton.setTitle("Start", for: .normal)
        startStopButton.backgroundColor = .green
        startStopButton.setTitleColor(.red, for: .normal)
        emailButton.isHidden = true

        // Line chart initial setup
        lineChart.xAxis.labelTextColor = .green
        lineChart.xAxis.granularityEnabled = true
        lineChart.xAxis.granularity = 1
        lineChart.leftAxis.labelTextColor = .green
        lineChart.leftAxis.drawZeroLineEnabled = true
        lineChart.rightAxis.labelTextColor = .green
        lineChart.legend.textColor = .red
        lineChart.chartDescription?.text = "Heart Rate Monitor"
        lineChart.chartDescription?.textColor = .red
        avg.reset()

        self.navigationItem.backBarButtonItem = UIBarButtonItem(title: "Back", style: .plain, target: nil, action: nil)
        // Create and start the peripheral manager
        peripheralManager = CBPeripheralManager(delegate: self, queue: nil)
        // Notification for updating the text view with incoming text
        updateIncomingData()
    }

    override func viewWillAppear(_ animated: Bool) {
    }

    override func viewDidAppear(_ animated: Bool) {
    }

    override func viewDidDisappear(_ animated: Bool) {
        peripheralManager?.stopAdvertising()
// self.peripheralManager = nil
super.viewDidLoad()
NotificationCenter.default.removeObserver(self)
}

/*@ Update incoming new heart rate data */
func updateIncomingData() {
    NotificationCenter.default.addObserver(forName: NSNotification.Name(rawValue: "Notify"), object: nil, queue: nil) {
        notification in
            if !self.pause { // stop updating when pause
                // thread safety in background queue, add and access heart rate data
                // array, and update the line chart
                self.queue.async(flags: .barrier) {
                    var lineChartEntry = [ChartDataEntry]()

                    // Parse the incoming data into readable Double values
                    // Perform moving average filter on incoming data before storing
                    // them in the array
                    if let characteristicStringValue = String(bytes: heartrates, encoding: .utf8) {
                        let lineStrings = characteristicStringValue
                            .replacingOccurrences(of: "\r", with: "")
                            .components(separatedBy: "\n")
                        for lineString in lineStrings {
                            let valuesStrings = lineString.components(separatedBy: CharacterSet(charactersIn: ",;\t"))
                            for valueString in valuesStrings {
                                if let value = Double(valueString) {
                                    self.numbers.append(self.avg.addSample(value: value))
                                }
                            }
                        }
                    }
                    // Update the line chart entries
                    // Make sure total entries don't surpass interval value
                    var count = self.numbers.count - self.interval
                    if count < 0 {
                        count = 0
                    }
                    for i in count..<self.numbers.count {
                        let num = self.numbers[i]
                        let value = ChartDataEntry(x: Double(i), y: num)
                        lineChartEntry.append(value)
                    }
                    print("Not a valid UTF-8 sequence")
                }
            } else {

            }
        }
    }

    // Set the line chart data set
    let line = LineChartDataSet(values: lineChartEntry, label: "Heart Rate")
    line.drawCirclesEnabled = false
    line.colors = [.yellow]
    let data = LineChartData()
data.addDataSet(line)

    DispatchQueue.main.async { // Update the chart UI
        self.lineChart.data = data
    }
}
self.lineChart.data?.notifyDataChanged()
self.lineChart.notifyDataSetChanged()

/* Check whether the peripheral manager updates state */
func peripheralManagerDidUpdateState(_ peripheral: CBPeripheralManager) {
    if peripheral.state == .poweredOn {
        return
    }
    print("Peripheral manager is running")
}

/* Check when someone subscribe to our characteristic, start sending the data */
func peripheralManager(_ peripheral: CBPeripheralManager, central: CBCentral, didSubscribeTo characteristic: CBCharacteristic) {
    print("Device subscribe to characteristic")
}

/* Check whether the peripheral manager starts advertising */
func peripheralManagerDidStartAdvertising(_ peripheral: CBPeripheralManager, error: Error?) {
    if let error = error {
        print("\n(error)\n")
        return
    }
}

/* Start and stop button action */
@IBAction func startorstop(sender: AnyObject) {
    if pause {
        // Start plotting data on chart, turn start to stop button
        startStopButton.setTitle("Stop", for: .normal)
        startStopButton.backgroundColor = .red
        startStopButton.setTitleColor(.green, for: .normal)
        emailButton.isHidden = true // Can send email now
        numbers.removeAll() // Clear the old data array
        avg.reset() // Restart the moving average filter
        lineChart.data = nil // Empty out the old data
    } else {
        // Stop plotting data, turn stop to start button
        startStopButton.setTitle("Start", for: .normal)
        startStopButton.backgroundColor = .green
        startStopButton.setTitleColor(.red, for: .normal)
        emailButton.isHidden = false // Can't send email
    }
    pause = !pause
}

/* Export the EKG data through email in CSV file format */
@IBAction func export(sender: AnyObject) {
    let filename = "EKG.csv"
    let path = NSURL(fileURLWithPath: NSTemporaryDirectory()).appendingPathComponent(filename)
    var csvText = "Time,Data\n"
    let messagebody = "From ME Senior Project EKG Team\n" + "\nThe EKG data is attached in this email as a "
+ "CSV file for testing purpose"

// Make sure there is EKG data to send out
if numbers.count > 0 {
    for (index, number) in numbers.enumerated() {
        let newline = "\(index),\(number)\n"
        csvText.append(contentsOf: newline)
    }
    do { // Set up the email template and go into email view to send
        try csvText.write(to: path!, atomically: true,
        encoding: String.Encoding.utf8)
        if MFMailComposeViewController.canSendMail() {
            let emailController = MFMailComposeViewController()
            emailController.mailComposeDelegate = self
            emailController.setToRecipients([])
            emailController.setSubject("EKG Data Export")
            emailController.setMessageBody(messagebody, isHTML: false)
            emailController.addAttachmentData(try NSData(contentsOf: path!) as Data, mimeType: "text/csv", fileName: "EKG.csv")
            present(emailController, animated: true, completion: nil)
        } catch { // CSV file fails to be created
            print("Failed to create file")
            print("\(error)")
        }
    }
    else { // No EKG data
        let alertVC = UIAlertController(title: "No Data",
        message: "There is no EKG data exists",
        preferredStyle: UIAlertControllerStyle.alert)
        let action = UIAlertAction(title: "OK",
        style: UIAlertActionStyle.default, handler: {
            (action: UIAlertAction) -> Void in
            self.dismiss(animated: true, completion: nil)
        })
        alertVC.addAction(action)
        present(alertVC, animated: true, completion: nil)
    }
}

/* Setup after email view is gone (Send or cancel email) */
func mailComposeController(_ controller: MFMailComposeViewController,
    didFinishWith result: MFMailComposeResult, error: Error?) {
    controller.dismiss(animated: true, completion: nil)
    numbers.removeAll() // Restart the data after email sent
    avg.reset() // Restart the moving average
    lineChart.data = nil // Empty out the data
    pause = true // Resume EKG data gathering
    startStopButton.setTitle("Start", for: .normal)
    startStopButton.backgroundColor = .green
    startStopButton.setTitleColor(.red, for: .normal)
    emailButton.isHidden = true
}
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
  <dict>
    <key>CFBundleIconFiles</key>
    <array>
      <string>heart.jpg</string>
    </array>
    <key>CFBundleDevelopmentRegion</key>
    <string>$\{DEVELOPMENT\_LANGUAGE\}</string>
    <key>CFBundleExecutable</key>
    <string>$\{EXECUTABLE\_NAME\}</string>
    <key>CFBundleIdentifier</key>
    <string>$\{PRODUCT\_BUNDLE\_IDENTIFIER\}</string>
    <key>CFBundleInfoDictionaryVersion</key>
    <string>6.0</string>
    <key>CFBundleName</key>
    <string>$\{PRODUCT\_NAME\}</string>
    <key>CFBundlePackageType</key>
    <string>APPL</string>
    <key>CFBundleShortVersionString</key>
    <string>1.0</string>
    <key>CFBundleVersion</key>
    <string>1</string>
    <key>LSRequiresIPhoneOS</key>
    <true/>
    <key>UILaunchStoryboardName</key>
    <string>LaunchScreen</string>
    <key>UIMainStoryboardFile</key>
    <string>Main</string>
    <key>UIRequiredDeviceCapabilities</key>
    <array>
      <string>armv7</string>
    </array>
    <key>UIRequiresFullScreen</key>
    <true/>
    <key>UISupportedInterfaceOrientations</key>
    <array>
      <string>UIInterfaceOrientationPortrait</string>
    </array>
    <key>UISupportedInterfaceOrientations~ipad</key>
    <array>
      <string>UIInterfaceOrientationPortrait</string>
      <string>UIInterfaceOrientationPortraitUpsideDown</string>
      <string>UIInterfaceOrientationLandscapeLeft</string>
      <string>UIInterfaceOrientationLandscapeRight</string>
    </array>
  </dict>
</plist>
<?xml version="1.0" encoding="UTF-8"?>
    version="3.0" toolsVersion="14460.31" targetRuntime="iOS.CocoaTouch"
    propertyAccessControl="none" useAutolayout="YES" launchScreen="YES"
    useTraitCollections="YES" useSafeAreas="YES" colorMatched="YES"
    initialViewController="01J-lp-oVM">
    <device id="retina4_7" orientation="portrait">
        <adaptation id="fullscreen"/>
    </device>
    <dependencies>
        <deployment identifier="iOS"/>
        <plugIn identifier="com.apple.InterfaceBuilder.IBCocoaTouchPlugin"
            version="14460.20"/>
        <capability name="Safe area layout guides" minToolsVersion="9.0"/>
        <capability name="documents saved in the Xcode 8 format" minToolsVersion="8.0"/>
    </dependencies>
    <scenes>
        <!--View Controller-->
        <scene sceneID="EHf-IW-A2E">
            <objects>
                <viewController id="01J-lp-oVM" sceneMemberID="viewController">
                    <view key="view" contentMode="scaleToFill" id="Ze5-6b-2t3">
                        <rect key="frame" x="0.0" y="0.0" width="375" height="667"/>
                        <autoresizingMask key="autoresizingMask" widthSizable="YES" heightSizable="YES"/>
                        <subviews>
                            <imageView userInteractionEnabled="NO"
                                contentMode="scaleToFill" horizontalHuggingPriority="251"
                                verticalHuggingPriority="251" image="heartrate"
                                adjustsImageSizeForAccessibilityContentSizeCategory="YES"
                                translatesAutoresizingMaskIntoConstraints="NO" id="RxE-oR-iEb">
                                <rect key="frame" x="0.0" y="20" width="375" height="647"/>
                            </imageView>
                            <label opaque="NO" userInteractionEnabled="NO"
                                contentMode="left" horizontalHuggingPriority="251"
                                verticalHuggingPriority="251" text="Mobile EKG"
                                textAlignment="natural" lineBreakMode="tailTruncation"
                                baselineAdjustment="alignBaselines"
                                adjustsFontSizeToFit="NO" translatesAutoresizingMaskIntoConstraints="NO"
                                id="1wK-sD-Fvg">
                                <rect key="frame" x="112" y="42" width="151" height="36"/>
                            </label>
                            <fontDescription key="fontDescription" type="system"
                                pointSize="30"/>
                            <color key="textColor" white="1" alpha="1" colorSpace="custom"
                                customColorSpace="genericGamma22GrayColorSpace"/>
                            <nil key="highlightedColor"/>
                        </subviews>
                    </view>
                </viewController>
            </objects>
        </scene>
    </scenes>
</document>
Chin Chao, Galen Wu,
Jim Kerman, Eric Burstedt
Sponsor: Dr. Nooristani
Main.storyboard (XML)

```xml
<?xml version="1.0" encoding="UTF-8"?>
toolsVersion="14460.31" targetRuntime="iOS.CocoaTouch"
propertyAccessControl="none" useAutolayout="YES" useTraitCollections="YES"
useSafeAreas="YES" colorMatched="YES" initialViewController="HAd-LH-a4B">
<device id="retina4_7" orientation="portrait">
<adaptation id="fullscreen"/>
<dependencies>
<deployment identifier="iOS"/>
<plugIn identifier="com.apple.InterfaceBuilder.IBCocoaTouchPlugin"
version="14460.20"/>
<capability name="Safe area layout guides" minToolsVersion="9.0"/>
<capability name="documents saved in the Xcode 8 format" minToolsVersion="8.0"/>
</dependencies>
<scenes>
<!--Navigation Controller-->
<scene sceneID="NwV-U9-Qic"/>
<objects>
<navigationController id="HAd-LH-a4B" sceneMemberID="viewController">
<navigationBar key="navigationBar" contentMode="scaleToFill"
id="xT5-r3-7Tx">
<rect key="frame" x="0.0" y="20" width="375" height="44"/>
</navigationBar>
<connections>
<segue destination="h1i-nR-FbJ" kind="relationship"
relationship="rootViewController" id="k0C-Vq-C3C"/>
</connections>
</navigationController>
<placeholder placeholderIdentifier="IBFirstResponder" id="CKM-aB-ldA"
userLabel="First Responder" sceneMemberID="firstResponder"/>
<point key="canvasLocation" x="-567" y="-3"/>
</objects>
</scene>
<!--Bluetooth EKG-->
<scene sceneID="ell-Uq-3hI"/>
<objects>
<viewController storyboardIdentifier="BLECentralViewController"
automaticallyAdjustsScrollViewInsets="NO" id="h1i-nR-FbJ"
customClass="BLECentralViewController" customModule="BLEEKG"
customModuleProvider="target" sceneMemberID="viewController">
<view key="view" contentMode="scaleToFill" id="DdB-KE-78G">
<rect key="frame" x="0.0" y="0.0" width="375" height="667"/>
</view>
<autoresizingMask key="autoresizingMask" widthSizable="YES" heightSizable="YES"/>
<subviews>
</subviews>
</viewController>
</scene>
</scenes>
</document>
```
When the Connect Button appears, there is an EKG available to connect. Click the top right refresh button to search for EKGs again or if the current connection attempt is pending for too long.
ME Senior Project

Chin Chao, Galen Wu

Jim Kerman, Eric Burstedt

App Developer: Chin Chao
secondItem="jp2-Jn-dEk" secondAttribute="centerX"
    id="cj5-JE-huX"/>
  <constraint firstItem="3uE-VX-gmu" firstAttribute="leading"
    secondItem="DdB-KE-78G" secondAttribute="leadingMargin"
    id="dTD-K9-wMa"/>
  <constraint firstItem="ipw-hq-CD1" firstAttribute="top"
    secondItem="yQI-33-obC" secondAttribute="bottom"
    constant="8" id="ezs-WR-mUg"/>
  <constraint firstItem="jp2-Jn-dEk" firstAttribute="centerX"
    secondItem="kAD-ux-bYh" secondAttribute="centerX"
    id="fRl-Km-PKj"/>
  <constraint firstItem="yQI-33-obC" firstAttribute="top"
    secondItem="jp2-Jn-dEk" secondAttribute="bottom"
    constant="8" id="nq2-nb-Paw"/>
  <constraint firstItem="ipw-hq-CD1" secondAttribute="centerX"
    constant="0" id="uJS-F6-wDR"/>
  <constraint firstItem="kAD-ux-bYh" secondAttribute="centerX"
    constant="0" id="v0d-a2-prn"/>
</constraints>
</viewLayoutGuide>

<viewkey="view" content_mode="scaleToFill" id="Q3A-V2-Pk9">
  <autoresizingMask key="autoresizingMask" width_sizable="YES"
    height_sizable="YES"/>
  <point key="canvasLocation" x="229.59999999999999" y="-3.1484257871064472"/>
</scene>

<!-- Uart Module View Controller -->
<scene sceneid="j6Z-p2-tKk">
  <objects>
    <viewController storyboardIdentifier="UartModuleViewController"
      useStoryboardIdentifierAsRestorationIdentifier="YES"
      id="3e9-OX-pZO" customClass="UartModuleViewController"
      customModule="BLEEKG" customModuleProvider="target"
      sceneMemberID="viewController">
      <view key="view" contentMode="scaleToFill" id="Q3A-V2-Pk9">
        <rect key="frame" x="0.0" y="0.0" width="375" height="667"/>
        <autoresizingMask key="autoresizingMask" width_sizable="YES"/>
        <point key="canvasLocation" x="229.59999999999999" y="-3.1484257871064472"/>
      </view>
    </viewController>
</scene>
heightSizable="YES"/>

<subviews>
  <scrollView clipsSubviews="YES" multipleTouchEnabled="YES"
    contentMode="scaleToFill" scrollEnabled="NO"
    translatesAutoresizingMaskIntoConstraints="NO"
    id="aAH-Lq-FuQ">
    <rect key="frame" x="0.0" y="0.0" width="375"
         height="667"/>
    <subviews>
      <view contentMode="scaleToFill"
            translatesAutoresizingMaskIntoConstraints="NO"
            id="BKx-X3-Gj5" userLabel="ContentView">
        <rect key="frame" x="0.0" y="0.0" width="375"
             height="667"/>
        <subviews>
          <view contentMode="scaleToFill"
                translatesAutoresizingMaskIntoConstraints="NO"
                id="4zk-lt-TXv" customClass="LineChartView"
                customModule="Charts">
            <rect key="frame" x="16" y="28" width="343"
                  height="469"/>
            <color key="backgroundColor" white="1" alpha="1"
                   colorSpace="custom"
                   customColorSpace="genericGamma22GrayColorSpace"/>
          </view>
          <button opaque="NO" contentMode="scaleToFill"
                  contentHorizontalAlignment="center"
                  contentVerticalAlignment="center"
                  buttonType="roundedRect"
                  lineBreakMode="middleTruncation"
                  translatesAutoresizingMaskIntoConstraints="NO"
                  id="PZx-eD-qdG">
            <rect key="frame" x="267" y="545"
                  width="92" height="77"/>
            <color key="backgroundColor"
                   red="0.90236398963730569"
                   green="0.87887527689574751"
                   blue="0.0" alpha="1"
                   colorSpace="custom"
                   customColorSpace="sRGB"/>
            <constraints>
              <constraint firstAttribute="width"
                          constant="92" id="BYI-Es-MDg"/>
            </constraints>
            <fontDescription key="fontDescription"
                             type="system" pointSize="30"/>
            <state key="normal" title="Email"/>
            <connections>
              <action selector="exportWithSender:"
                      destination="3e9-OX-pZO"
                      eventType="touchUpInside"
                      id="aDs-w0-OA6"/>
            </connections>
          </button>
          <button opaque="NO" contentMode="scaleToFill"
                  contentHorizontalAlignment="center"
                  contentVerticalAlignment="center"
                  buttonType="roundedRect"
                  lineBreakMode="middleTruncation"
Picture assets used in the app:
Appendix [F] – Concept Layout Drawings
<table>
<thead>
<tr>
<th>ITEM NO.</th>
<th>PART NUMBER</th>
<th>DESCRIPTION</th>
<th>QTY.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>110</td>
<td>FRONT PANEL</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>111</td>
<td>LEFT PANEL</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>112</td>
<td>BOTTOM PANEL</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>113</td>
<td>BACK PANEL</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>114</td>
<td>FINGER ELECTRODE</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>115</td>
<td>CHEST ELECTRODE</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>116</td>
<td>RIGHT PANEL</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>ADA1904</td>
<td>CHARGING BOARD</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>258</td>
<td>BATTERY</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>Q8F8BZZRYG02E</td>
<td>LED</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>SRB22A2FBNN</td>
<td>SWITCH</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>93365A142</td>
<td>8-32 0.312&quot; LG BRASS HEAT SET INSERT</td>
<td>4</td>
</tr>
<tr>
<td>13</td>
<td>90273A193</td>
<td>8-32 7/16&quot; LG STEEL FLAT HEAD PHILLIPS SCREW</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>94735A176</td>
<td>2-56 0.188IN LG BRASS HEAT SET INSERT</td>
<td>8</td>
</tr>
<tr>
<td>15</td>
<td>92942A706</td>
<td>2-56 3/16&quot; NYLON PAN HEAD SLOTTED SCREW</td>
<td>8</td>
</tr>
</tbody>
</table>
NOTES
UNLESS OTHERWISE SPECIFIED:
1. ALL DIMS. IN INCHES
2. TOLERANCES:
   X.XX = ±0.01
   X.XXX = ±0.005
   ANGLES = ±1°
3. INSIDE TOOL RADIUS 0.02 MAX.
4. BREAK SHARP EDGES 0.02 MAX.
5. MATERIAL: ABS

Cal Poly Mechanical Engineering
ME 430 - FALL 2018
Lab Section: 02  Assignment #N/A  Title: EKG HOUSING - FRONT
Dwg. #: 110  Nxt Asb: N/A  Date: 12/8/18  Drwn. By: J.KERMAN E.BURSTEDT
Scale: 1:1  Chkd. By: N/A
NOTES
UNLESS OTHERWISE SPECIFIED:
1. ALL DIMS. IN INCHES
2. TOLERANCES:
   X.XX = ±0.01
   X.XXX = ±0.005
   ANGLES = ±1°
3. INSIDE TOOL RADIUS 0.02 MAX.
4. BREAK SHARP EDGES 0.02 MAX.
5. MATERIAL: ABS
NOTES
UNLESS OTHERWISE SPECIFIED:
1. ALL DIMS. IN INCHES
2. TOLERANCES:
   X.XX = ±0.01
   X.XXX = ±0.005
   ANGLES = ±1°
3. INSIDE TOOL RADIUS 0.02 MAX.
4. BREAK SHARP EDGES 0.02 MAX.
5. MATERIAL: ABS

Cal Poly Mechanical Engineering
ME 430 - FALL 2018
Lab Section: 02
Dwg. #: 111
Assignment #N/A
Nxt Asb: N/A
Title: EKG HOUSING - LEFT PANEL
Date: 12/8/18
Scale: 1:1
Drwn. By: J.KERMAN E.BURSTEDT
Chkd. By: N/A
NOTES
UNLESS OTHERWISE SPECIFIED:
1. ALL DIMS. IN INCHES
2. TOLERANCES:
   X.XX = ±0.01
   X.XXX = ±0.005
   ANGLES = ±1°
3. INSIDE TOOL RADIUS 0.02 MAX.
4. BREAK SHARP EDGES 0.02 MAX.
5. MATERIAL: ABS
NOTES
UNLESS OTHERWISE SPECIFIED:
1. ALL DIMS. IN INCHES
2. TOLERANCES:
   X.XX = ±0.01
   X.XXX = ±0.005
   ANGLES = ±1°
3. INSIDE TOOL RADIUS 0.02 MAX.
4. BREAK SHARP EDGES 0.02 MAX.
5. MATERIAL: ABS
NOTES
UNLESS OTHERWISE SPECIFIED:
1. ALL DIMS. IN INCHES
2. TOLERANCES:
   X.XX = ±0.01
   X.XXX = ±0.005
   ANGLES = ±1°
3. INSIDE TOOL RADIUS 0.02 MAX.
4. BREAK SHARP EDGES 0.02 MAX.
5. MATERIAL: AL 6061-T6
6. PART WILL BE MANUFACTURED FROM SHEET METAL ROLLED INTO SPECIFIED SHAPE AND WILL HAVE SMALL GAP RUNNING LENGTH OF PART. SIZE OF GAP SHOULD BE MINIMIZED BUT NOT CRITICAL.
NOTES
UNLESS OTHERWISE SPECIFIED:
1. ALL DIMS. IN INCHES
2. TOLERANCES:
   X.XX = ±0.01
   X.XXX = ±0.005
   ANGLES = ±1°
3. INSIDE TOOL RADIUS 0.02 MAX.
4. BREAK SHARP EDGES 0.02 MAX.
5. MATERIAL: Al 6061 - T6
NOTES
UNLESS OTHERWISE SPECIFIED:
1. ALL DIMS. IN INCHES
2. TOLERANCES:
   X.XX = ±0.01
   X.XXX = ±0.005
   ANGLES = ±1°
3. INSIDE TOOL RADIUS 0.02 MAX.
4. BREAK SHARP EDGES 0.02 MAX.
5. MATERIAL: ABS
## Appendix [H] – Purchased Parts Detail

<table>
<thead>
<tr>
<th>ITEM</th>
<th>REFERENCE DESIGNATOR</th>
<th>DESCRIPTION</th>
<th>SUGG. MFG.</th>
<th>SUGG. MFG-PN.</th>
<th>SUGG. DISTRIBUT.</th>
<th>SUGG. DIST. P.N.</th>
<th>LINK</th>
<th>QTY</th>
<th>PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>N/A</td>
<td>Fabricated Circuit Board</td>
<td>Bay Area Circuits</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>1</td>
<td>Price Vary</td>
</tr>
<tr>
<td>2</td>
<td>U1</td>
<td>IC REG LINEAR 1.8V 20MA SOT23-5</td>
<td>Maxim Integrated</td>
<td>MAX1726EUK18+T</td>
<td>Digikey</td>
<td>MAX1726EUK18+T-ND</td>
<td><a href="https://www.amazon.com/StrivedayTM-Flexible-Silicone-electronic-electrics/dp/B01KQ2JNLI/ref=sr_1_3?ie=UTF8&amp;qid=1543800429&amp;sr=8-3&amp;keywords=wires">Link</a></td>
<td>1</td>
<td>$1.95</td>
</tr>
<tr>
<td>4</td>
<td>J8</td>
<td>CONN HEADER FEM 12POS.1&quot; SGL TIN</td>
<td>Sullins Connector Solutions</td>
<td>PPTC121LFBN-RC</td>
<td>Digikey</td>
<td>S6100-ND</td>
<td><a href="https://www.digikey.com/product-detail/en/sullins-connector-solutions/PPTC121LFBN-RC/S6100-ND/807231">Link</a></td>
<td>1</td>
<td>$0.78</td>
</tr>
<tr>
<td>7</td>
<td>C1, C3, C4, C5</td>
<td>CAP CER 10PF 1KV C0G/NP0 0805</td>
<td>KEMET</td>
<td>C0805C100JDGACTU</td>
<td>Digikey</td>
<td>399-7062-1-ND</td>
<td><a href="https://www.digikey.com/product-detail/en/kemet/C0805C100JDGACTU/399-7062-1-ND/3439407">Link</a></td>
<td>4</td>
<td>$0.43</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>9</td>
<td>C8, C17, C26</td>
<td>CAP CER 0.1UF 10V X7R 0402</td>
<td>TDK Corporation</td>
<td>C1005X7R1A104K050BB</td>
<td>Digike</td>
<td>445-4958-1-ND</td>
<td><a href="https://www.digikey.com/product-detail/en/tdk-corporation/C1005X7R1A104K050BB/445-4958-1-ND/2093572">https://www.digikey.com/product-detail/en/tdk-corporation/C1005X7R1A104K050BB/445-4958-1-ND/2093572</a></td>
<td>3</td>
<td>$0.10</td>
</tr>
<tr>
<td>12</td>
<td>R1, R6, R7, R10</td>
<td>RES SMD 49.9K OHM 0.1% 1/8W 0805</td>
<td>Panasonic Electronic Components</td>
<td>ERA-6AEB4992V</td>
<td>Digike</td>
<td>P49.9KDACT-ND</td>
<td><a href="https://www.digikey.com/product-detail/en/panasonic-electronic-components/ERA-6AEB4992V/P49.9KDACT-ND/3075162">https://www.digikey.com/product-detail/en/panasonic-electronic-components/ERA-6AEB4992V/P49.9KDACT-ND/3075162</a></td>
<td>4</td>
<td>0.36</td>
</tr>
<tr>
<td>13</td>
<td>R2, R3, R4, R5, R8, R9, R11, R12</td>
<td>RES SMD 0 OHM JUMPER 1/8W 0805</td>
<td>Vishay Dale</td>
<td>CRCW08050000Z0EA</td>
<td>Digike</td>
<td>541-0.0ACT-ND</td>
<td><a href="https://www.digikey.com/product-detail/en/vishay-dale/CRCW08050000Z0EA/541-0.0ACT-ND/1180267">https://www.digikey.com/product-detail/en/vishay-dale/CRCW08050000Z0EA/541-0.0ACT-ND/1180267</a></td>
<td>8</td>
<td>$0.10</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>R16</td>
<td>RES SMD 10K OHM 0.5% 1/10W 0805</td>
<td>Susumu</td>
<td>RR1220P-103-D</td>
<td>Digikeyp</td>
<td>RR12P10.0KDCT-ND</td>
<td>1</td>
<td>$0.11</td>
<td></td>
</tr>
<tr>
<td>ITEM</td>
<td>REFERENCE DESIGNATOR</td>
<td>DESCRIPTION</td>
<td>SUGG. MFG.</td>
<td>SUGG. MFG-PN.</td>
<td>SUGG. DISTRIBUT.</td>
<td>SUGG. DIST. P.N.</td>
<td>LINK</td>
<td>QTY</td>
<td>PRICE</td>
</tr>
<tr>
<td>------</td>
<td>----------------------</td>
<td>-----------------------------------------------</td>
<td>----------------</td>
<td>------------------</td>
<td>------------------</td>
<td>------------------</td>
<td>----------------------------------------------------------------------</td>
<td>-----</td>
<td>---------</td>
</tr>
<tr>
<td>1</td>
<td>N/A</td>
<td>3D Printing Material PLA</td>
<td>Hatchbox</td>
<td>3D PLA-1KG1.75-WHT</td>
<td>Amazon</td>
<td>B00J0GMMPP6</td>
<td><a href="https://www.amazon.com/HATCHBOX-3D-Filament-Dimensional-Accuracy/dp/B00J0GMMPP6/ref=sr_1_3?si=industrial&amp;ie=UTF8&amp;qid=1543797630&amp;sr=1-3&amp;keywords=pla">https://www.amazon.com/HATCHBOX-3D-Filament-Dimensional-Accuracy/dp/B00J0GMMPP6/ref=sr_1_3?si=industrial&amp;ie=UTF8&amp;qid=1543797630&amp;sr=1-3&amp;keywords=pla</a></td>
<td>1</td>
<td>$19.99</td>
</tr>
<tr>
<td>2</td>
<td>N/A</td>
<td>Disposable / Reusable Dry EEG Electrode</td>
<td>Florida Research Instruments</td>
<td>TDE-200</td>
<td>N/A</td>
<td>N/A</td>
<td><a href="https://fri-fl-shop.com/product/tde-200/">https://fri-fl-shop.com/product/tde-200/</a></td>
<td>1 ten pack</td>
<td>$10.00</td>
</tr>
<tr>
<td>4</td>
<td>N/A</td>
<td>Battery Packs Lithium Ion Polymer Battery 3.7V 1200mAh (1 piece)</td>
<td>Adafruit Industries LLC</td>
<td>258</td>
<td>Amazon</td>
<td>B00J2QET64</td>
<td><a href="https://www.amazon.com/gp/product/B00J2QET64/ref=od_aui_detailpages00?ie=UTF8&amp;psc=1">https://www.amazon.com/gp/product/B00J2QET64/ref=od_aui_detailpages00?ie=UTF8&amp;psc=1</a></td>
<td>1</td>
<td>$15.35</td>
</tr>
<tr>
<td>5</td>
<td>N/A</td>
<td>Adafruit Micro-Lipo Charger for LiPo/LiIon Batt w/MicroUSB Jack</td>
<td>Adafruit Industries LLC</td>
<td>ADA1904</td>
<td>Amazon</td>
<td>B00MJ0HOKS</td>
<td><a href="https://www.amazon.com/gp/product/B00MJ0HOKS/ref=od_aui_detailpages00?ie=UTF8&amp;psc=1">https://www.amazon.com/gp/product/B00MJ0HOKS/ref=od_aui_detailpages00?ie=UTF8&amp;psc=1</a></td>
<td>1</td>
<td>$8.04</td>
</tr>
<tr>
<td>6</td>
<td>N/A</td>
<td>8-32 0.312&quot; Lg Brass Heat Set Insert for Plastic</td>
<td>McMaster Carr</td>
<td>93365A142</td>
<td>McMaster Carr</td>
<td>93365A142</td>
<td><a href="https://www.mcmaster.com/#93365a142?i=1ca5l76">https://www.mcmaster.com/#93365a142?i=1ca5l76</a></td>
<td>1</td>
<td>$8.62</td>
</tr>
<tr>
<td>7</td>
<td>N/A</td>
<td>8-32 7/16&quot; Lg Steel Flat Head Phillips Screw</td>
<td>McMaster Carr</td>
<td>90273A193</td>
<td>McMaster Carr</td>
<td>90273A193</td>
<td><a href="https://www.mcmaster.com/#90273a193?i=1ca6nwr">https://www.mcmaster.com/#90273a193?i=1ca6nwr</a></td>
<td>1</td>
<td>$4.06</td>
</tr>
<tr>
<td>8</td>
<td>N/A</td>
<td>2-56 3/16&quot; Nylon Pan Head Slotted Screws</td>
<td>McMaster Carr</td>
<td>92942A706</td>
<td>McMaster Carr</td>
<td>92942A706</td>
<td><a href="https://www.mcmaster.com/92942A706">https://www.mcmaster.com/92942A706</a></td>
<td>1</td>
<td>$5.95</td>
</tr>
<tr>
<td>9</td>
<td>N/A</td>
<td>0.025&quot; Thick Aluminum Sheet Metal</td>
<td>McMaster Carr</td>
<td>89015K138</td>
<td>McMaster Carr</td>
<td>89015K138</td>
<td><a href="https://www.mcmaster.com/catalog/124/3768">https://www.mcmaster.com/catalog/124/3768</a></td>
<td>1</td>
<td>$17.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>LED Panel Indicator</strong> Green, Red, Yellow 2.2V Green, 2V Red, 2.1V Yellow 20mA Round with Flat Top Solder Lug</td>
<td><strong>APEM Inc.</strong></td>
<td><strong>Q8F8BZZRYG02E</strong></td>
<td><strong>Digikey</strong></td>
<td><strong>679-4074-ND</strong></td>
<td><strong><a href="https://www.digikey.com/products/en?keywords=Q8F8BZZRYG02E">https://www.digikey.com/products/en?keywords=Q8F8BZZRYG02E</a></strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>$14.06</td>
</tr>
</tbody>
</table>

|    |   | Constructed PCB | See PCB BOM | | | | | | |
## Appendix [I] – Budget Procurement List

<table>
<thead>
<tr>
<th>ITEM</th>
<th>DESCRIPTION</th>
<th>SUGG. MFG.</th>
<th>SUGG. DISTRIBUT.</th>
<th>QTY</th>
<th>PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Evaluation Kit for the MAX39003 Biopotential AFE</td>
<td>Maxim</td>
<td>Maxim</td>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>Ultra-Low Power, Single-Channel Integrated Biopotential (ECG, RR to R Detection) AFE</td>
<td>Maxim</td>
<td>Maxim</td>
<td>10</td>
<td>5,263</td>
</tr>
<tr>
<td>3</td>
<td>Bluetooth / BLE 15.1 Development Tools Adafruit Feather M0 Bluefruit LE</td>
<td>Adafruit Industries LLC</td>
<td>Amazon</td>
<td>2</td>
<td>30.93</td>
</tr>
<tr>
<td>4</td>
<td>Battery Packs Lithium Ion Polymer Battery 3.7V 1200mAh (1 piece)</td>
<td>Adafruit Industries LLC</td>
<td>Amazon</td>
<td>2</td>
<td>9.09</td>
</tr>
<tr>
<td>5</td>
<td>8x20 0.312&quot; Lg Brass Heat Set Insert for Plastic</td>
<td>McMaster Carr</td>
<td>McMaster Carr</td>
<td>1</td>
<td>8.62</td>
</tr>
<tr>
<td>6</td>
<td>0.32 7/16&quot; Lg Steel Flat Head Phillips Screw</td>
<td>McMaster Carr</td>
<td>McMaster Carr</td>
<td>1</td>
<td>3.84</td>
</tr>
<tr>
<td>7</td>
<td>0.025&quot; Thick Aluminum Sheet Metal</td>
<td>McMaster Carr</td>
<td>McMaster Carr</td>
<td>1</td>
<td>16.5</td>
</tr>
<tr>
<td>8</td>
<td>Ag/AgCl Electrodes</td>
<td>BioPac Systems</td>
<td>BioPac Systems</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>9</td>
<td>ABS Print Material</td>
<td>Mitus</td>
<td>Amazon</td>
<td>1</td>
<td>24.89</td>
</tr>
<tr>
<td>10</td>
<td>Adafruit Micro-Lipo Charger for LiPo/LiIon Batt w/MicroUSB Jack</td>
<td>Adafruit Industries LLC</td>
<td>Adafruit</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Oscillator XO 32768kHz CMOS SMD</td>
<td>Dodes Incorporated</td>
<td>DigiKey</td>
<td>1</td>
<td>2.40</td>
</tr>
<tr>
<td>12</td>
<td>Conn Header Female 18POS 1&quot; Tin</td>
<td>SunBus Connector Solution</td>
<td>DigiKey</td>
<td>2</td>
<td>0.98</td>
</tr>
<tr>
<td>13</td>
<td>IC Reg Linear 1.8V 20mA SOT23-5</td>
<td>Maxim Integrated</td>
<td>DigiKey</td>
<td>3</td>
<td>1.85</td>
</tr>
<tr>
<td>14</td>
<td>CAP CER 0.1 uF 10V X7R 0402</td>
<td>TDK Corp</td>
<td>DigiKey</td>
<td>10</td>
<td>0.065</td>
</tr>
<tr>
<td>15</td>
<td>RES SMD 490k Ohm 1% 1/10 W 0603</td>
<td>Panasonic</td>
<td>DigiKey</td>
<td>10</td>
<td>0.065</td>
</tr>
<tr>
<td>16</td>
<td>RES SMD 0 Ohm Jumper 1/8W 0805</td>
<td>Vishay Dale</td>
<td>DigiKey</td>
<td>20</td>
<td>0.044</td>
</tr>
<tr>
<td>17</td>
<td>RES SMD 49k Ohm 0.1% 1/8W 0805</td>
<td>Panasonic</td>
<td>DigiKey</td>
<td>10</td>
<td>0.302</td>
</tr>
<tr>
<td>18</td>
<td>RES SMD 10k Ohm 0.5% 1/8W 0805</td>
<td>Susumu</td>
<td>DigiKey</td>
<td>5</td>
<td>0.11</td>
</tr>
<tr>
<td>19</td>
<td>CAP CER 10pF 1KV DO4NP0 0805</td>
<td>Keneset</td>
<td>DigiKey</td>
<td>10</td>
<td>0.3</td>
</tr>
<tr>
<td>20</td>
<td>CAP CER 100pF 63V X7R 0402</td>
<td>Murata Electronics</td>
<td>DigiKey</td>
<td>10</td>
<td>0.036</td>
</tr>
<tr>
<td>21</td>
<td>CAP CER 1uf 10V Y5V 0402</td>
<td>Samsung</td>
<td>DigiKey</td>
<td>20</td>
<td>0.032</td>
</tr>
<tr>
<td>22</td>
<td>CAP CER 1uf 5.5V Y5R 0503</td>
<td>Samsung</td>
<td>DigiKey</td>
<td>20</td>
<td>0.209</td>
</tr>
<tr>
<td>23</td>
<td>CONN Header Female 12 Pos 0.1&quot; Tin</td>
<td>SunBus Connector Solution</td>
<td>DigiKey</td>
<td>2</td>
<td>0.78</td>
</tr>
<tr>
<td>24</td>
<td>CONN Header Female 4 Pos 0.1&quot; Tin</td>
<td>SunBus Connector Solution</td>
<td>DigiKey</td>
<td>2</td>
<td>0.45</td>
</tr>
<tr>
<td>25</td>
<td>Stencil</td>
<td>Bay Area Circuits</td>
<td>Bay Area Circuits</td>
<td>1</td>
<td>139</td>
</tr>
<tr>
<td>26</td>
<td>46023 (2 Layer Instant Quote)</td>
<td>Bay Area Circuits</td>
<td>Bay Area Circuits</td>
<td>2</td>
<td>114.5</td>
</tr>
<tr>
<td>27</td>
<td>Shaggy 4 Pieces Putty Scrapers</td>
<td>Video Computer Co</td>
<td>Amazon</td>
<td>1</td>
<td>7.89</td>
</tr>
<tr>
<td>28</td>
<td>MG Chemicals 5341 No Clean Flux Paste</td>
<td>Amazon.com Services</td>
<td>Amazon</td>
<td>1</td>
<td>10.95</td>
</tr>
<tr>
<td>29</td>
<td>Solder Paste sm427/67 0/AgO 4 No-Clean Lead-Free</td>
<td>Chip Quik</td>
<td>Amazon</td>
<td>1</td>
<td>15.55</td>
</tr>
<tr>
<td>30</td>
<td>MG Chemicals 8329/110 CM Thermal Conductive Adhesive, Medium Cure</td>
<td>Amazon.com Services</td>
<td>Amazon</td>
<td>1</td>
<td>13.87</td>
</tr>
<tr>
<td>31</td>
<td>Gorilla Ultra Control Super Glue Gel, 15g, Clear</td>
<td>Amazon.com Services</td>
<td>Amazon</td>
<td>1</td>
<td>6.49</td>
</tr>
</tbody>
</table>

**Total = $ 711.82**
Appendix [J]: Operators’ Manual

Parts List
The following devices and mobile app are needed to fully use the Portable EKG effectively:

- The EKG device (Adafruit Feather M0 Bluefruit LE)
- The iOS iPhone or iPad with Bluetooth LE connection capability (Either screen orientation is fine)
- The mobile app developed for this project to interface with the EKG device
- MicroUSB to connect to the Lipo battery (power the device) in order to recharge if needed

EKG Reading
Setup:
1. Turn on the switch on the side of the EKG device
2. Confirm the device is on by checking the green LED light next to the switch
   - **NOTE**: If the light is yellow, then the battery is running out
   - Open the back case and connect the microUSB to the Adafruit to charge the device
   - **WARNING**: Make sure to not loosening any electronics inside, especially the Adafruit

3. Download the EKG mobile app onto an iPhone or iPad
4. Open up the mobile app
5. Confirm a connect button appears at the connect screen when a compatible turn-on EKG is nearby.
   - **NOTE**: If the connect button doesn’t appear, click the refresh button on the top right to start another search (17 seconds interval) again
6. Click the connect button to connect to the EKG wirelessly
7. Confirm the app goes into the EKG monitor screen
   - **NOTE**: If the connection is pending for too long, click the refresh button on the top right to stop the current attempt and start from step 5 again
8. Click the start button on the bottom left to start plotting the heart rate onto the app
9. Confirm the chart is showing visible data and updating continuously (start button turns into stop button)
   - **NOTE**: The current app may sometime crash due to unknown error. If it occurs, simply start from step 4 again
10. Hold the EKG device so the electrodes are facing the user
11. Put the wrist strap onto the right wrist
   ○ **WARNING:** If the wrist strap isn’t used, the EKG can drop to the ground during use and risk damaging the device

12. Put the right index finger to the side opening and the left index finger to the bottom opening to hold the device
13. Press the device against the left side of the chest so the electrodes are physically in contact with the chest
14. Confirm a visible heart rate is showing on the chart
15. Put the EKG down
16. Click the stop button on the bottom left to stop plotting
17. Confirm no more new data appears on the chart (stop button turns into start button)
18. Confirm the email button appears on the bottom right (Should only appear whenever there are EKG data in the app and the app is currently no plotting)
19. Click the email button to export the EKG result
20. Confirm an email view pops up with a CSV file as part of its content
21. Choose a recipient (e.g. physician) to send the email to
22. Confirm the email view disappears (Return to the EKG monitor screen)
   ○ To get another set of EKG data, starts at step 8 again (Old EKG data will be deleted once restarts)
23. Confirm the target email recipient received the email after sending it
24. Click the disconnect button on the top left to disconnect from the EKG
25. Confirm the app goes back to the connect screen
   ○ Starts from step 5 again to gather another set of EKG data
26. Turn off the EKG since it is not used
27. Confirm the LED light is off
28. Download the CSV file from the email to open it in application (e.g. Excel) that can view CSV file
29. Plot the CSV data to confirm the result matches the one from the app
Appendix K – Safety Hazard Checklist

DESIGN HAZARD CHECKLIST

Team: EKG Project
Advisor: Emily Fysh
Date: 2/27/18

☐ Y ☐ N 1. Will the system include hazardous revolving, running, rolling, or mixing actions?
☐ Y ☐ N 2. Will the system include hazardous reciprocating, shearing, punching, pressing, squeezing, drawing, or cutting actions?
☐ Y ☐ N 3. Will any part of the design undergo high accelerations/decelerations?
☐ Y ☐ N 4. Will the system have any large (>5 kg) moving masses or large (>250 N) forces?
☐ Y ☐ N 5. Could the system produce a projectile?
☐ Y ☐ N 6. Could the system fall (due to gravity), creating injury?
☐ Y ☐ N 7. Will a user be exposed to overhanging weights as part of the design?
☐ Y ☐ N 8. Will the system have any burrs, sharp edges, shear points, or pinch points?
☐ Y ☐ N 9. Will any part of the electrical systems not be grounded?
☐ Y ☐ N 10. Will there be any large batteries (over 30 V)?
☐ Y ☐ N 11. Will there be any exposed electrical connections in the system (over 40 V)?
☐ Y ☐ N 12. Will there be any stored energy in the system such as flywheels, hanging weights or pressurized fluids/gases?
☐ Y ☐ N 13. Will there be any explosive or flammable liquids, gases, or small particle fuel as part of the system?
☐ Y ☐ N 14. Will the user be required to exert any abnormal effort or experience any abnormal physical posture during the use of the design?
☐ Y ☐ N 15. Will there be any materials known to be hazardous to humans involved in either the design or its manufacturing?
☐ Y ☐ N 16. Could the system generate high levels (>90 dBA) of noise?
☐ Y ☐ N 17. Will the device/system be exposed to extreme environmental conditions such as fog, humidity, or cold/high temperatures, during normal use?
☐ Y ☐ N 18. Is it possible for the system to be used in an unsafe manner?
☐ Y ☐ N 19. For powered systems, is there an emergency stop button?
☐ Y ☐ N 20. Will there be any other potential hazards not listed above? If yes, please explain on reverse.

For any “Y” responses, add (1) a complete description, (2) a list of corrective actions to be taken, and (3) date to be completed on the reverse side.
<table>
<thead>
<tr>
<th>Description of Hazard</th>
<th>Planned Corrective Action</th>
<th>Planned Date</th>
<th>Actual Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Falling</td>
<td>Minimize device weight. Include safety wrist-strap.</td>
<td>5/17</td>
<td></td>
</tr>
<tr>
<td>Electrical grounding</td>
<td>Will include standard safety designs and implementations for electrical components</td>
<td>6/5</td>
<td></td>
</tr>
<tr>
<td>Electro Static Discharge</td>
<td>Implement safe circuit designs and practices.</td>
<td>6/5</td>
<td></td>
</tr>
<tr>
<td>Emotional Distress</td>
<td>Limit the amount of information provided to patient directly. Important health issues should be discussed by doctor.</td>
<td>5/25</td>
<td></td>
</tr>
</tbody>
</table>
### 2-6 Self-Administered EKG …

<table>
<thead>
<tr>
<th>Category</th>
<th>Required</th>
<th>Progress</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Problem Definition</strong></td>
<td>0h</td>
<td>50%</td>
</tr>
<tr>
<td><strong>Perform Customer/Need Research</strong></td>
<td>0h</td>
<td>100%</td>
</tr>
<tr>
<td>Interview Sponsor</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Interview Users</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Online search for customer compla...</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td><strong>Perform Technical Research</strong></td>
<td>0h</td>
<td>100%</td>
</tr>
<tr>
<td>Develop Search Criteria</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Perform Literature Search</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td><strong>Perform Product Research</strong></td>
<td>0h</td>
<td>100%</td>
</tr>
<tr>
<td>Online search for current products</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Ask sponsor/users about current pr...</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Patent search for similar products</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Write Problem Statement</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Write Customer Needs/Wants List</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Perform QFD</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Create Initial Project Plan</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Write Scope of Work</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Scope of Work Milestone</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td><strong>Conceptualization</strong></td>
<td>0h</td>
<td>100%</td>
</tr>
<tr>
<td>Generate Ideas</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td><strong>Preliminary EKG Design</strong></td>
<td>0h</td>
<td>100%</td>
</tr>
<tr>
<td>Research Existing EKG Design</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Determine Device Placement on th...</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Determine Size Range</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Develop Preliminary EKG Filtering</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Develop Form of the Device</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Concept Prototype</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Evaluate Device Form</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td><strong>Preliminary Mobile App Design</strong></td>
<td>0h</td>
<td>100%</td>
</tr>
<tr>
<td>Research Cross-Platform App Feasi...</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Search for Development Platform</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Design Preliminary App Layout</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Research Bluetooth Compatibility</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td><strong>Existing EKG Product Testing</strong></td>
<td>0h</td>
<td>100%</td>
</tr>
<tr>
<td>Meet Sponsor for Product Testing</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Test Existing Product</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Record Result and Feasibility</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Preliminary Design Review</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td><strong>Detail Design</strong></td>
<td>0h</td>
<td>15%</td>
</tr>
<tr>
<td>Interim Design Review</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Critical Design Review</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Hardware Layout</td>
<td>0%</td>
<td>50%</td>
</tr>
<tr>
<td>Packaging Design</td>
<td>0%</td>
<td>70%</td>
</tr>
<tr>
<td>Complete App Layout</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Complete EKG Filtering Design</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td><strong>Manufacturing</strong></td>
<td>0h</td>
<td>41%</td>
</tr>
<tr>
<td>Send purchase list to sponsor</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Verify sponsor purchased parts</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Manufacture Part 1</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Manufacturing &amp; Test Review</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Hardware Safety Demo</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Iterate Packaging</td>
<td>0%</td>
<td>30%</td>
</tr>
<tr>
<td>Develop Finalized App</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td><strong>Testing</strong></td>
<td>0h</td>
<td>0%</td>
</tr>
<tr>
<td>Setup Test 1</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Perform Test 1</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Project Expo</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Continue Testing</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Fine-tune Design and Packaging</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>