Expanded Telehealth Platform for Android

Phillip Azar
Adarsh Mani
Quan Peng
Jochem Van Gaalen
Daniel Aranki, Ed.

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2015-83
http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-83.html

May 13, 2015

Acknowledgement

With great thanks to Professor Ruzena Bajcsy, Professor Ali Javey, and Daniel Aranki for their assistance in reviewing this work.

2015

Expanded Tele-Health Platform for Android

FINAL CAPSTONE REPORT

PHILLIP AZAR

M.ENG EECS – ROBOTICS AND EMBEDDED SOFTWARE

Table of Contents

1. Introduction

In 2010, $17 billion dollars was spent on patient readmissions in Medicare. (Centers for Medicare & Medicaid Services: 2012). The most preventable readmission factors - and likely the most severe - are improper administration of therapy and medication. To combat this, we are developing a lean, smartphone-based tele medical system that not only provides personal health statistics to the patient, but relays this information to the doctor allowing for a more tailored approach to medicine. By leveraging the latest open source health monitoring algorithms and libraries, we have created a reliable, low-cost and easily implementable solution that has the potential to simultaneously improve patient care and dramatically reduce healthcare expenditure.

The team, advised by Dr. Ruzena Bajcsy and led by Ph.D. student Daniel Aranki, consisted of several members, including four Master of Engineering students (Phillip Azar, Adarsh Mani, Quan Peng, and Jochem van Gaalen) specializing in Robotics & Embedded Software concentration in the EECS department, and engineers Arjun Chopra, Priyanka Nigam, Sneha Sankavaram, Maya Reddy and Qiyin Wu.

The aim of this project is to develop and test a highly configurable, open-sourced Android powered framework that will enable doctors and hospitals to remotely monitor outpatients with chronic health conditions and determine their risk of re-admittance. This platform will transform the ubiquitous smartphone into the backbone of a tele monitoring system which uses various sensors to track different vital signs. These sensors can either be off-the-shelf products (e.g. temperature sensors) or sensors embedded in the smartphone itself (such as gyroscopes, accelerometers, cameras etc.). Once the data is extracted, the smartphone will be used to relay the information, in a fault-tolerant manner[1], to servers where various diagnostic algorithms are run to calculate re-admittance risk factor and other recuperation metrics. Security and privacy procedures will be embedded into these data structures prior to storage.

[1] Fault-tolerance can be thought of as a personal guarantee to the user that the process will be handled despite the presence of faults such as unexpected power downs, disconnection etc.

The team's specific contribution to this project is to develop:

1. Fault-tolerant communication between various off-the-shelf vital sign extractors and the smartphone via Bluetooth.
2. Novel vital sign extraction algorithms for heart rate and blood pressure using the smartphone's camera to serve as a proof of concept to demonstrate the fault-tolerant communication and provide a simpler way to conduct health and vital monitoring.

Over the course of the academic year, an application in Android was built and tested through laboratory validation of individual components, and a pilot study conducted near the end of the project which combined these. The results of this work is outlined in this report, along with challenges faced and potential next steps for this project.

This project was also analyzed from a business perspective, with potential competitors highlighted, and a thorough analysis of the industry and market trends to aid in identifying potential business opportunities.

2. Market Trends Analysis

2.1.1 Motivation & Outline

This section of the paper is dedicated to analyzing the telehealth industry and deriving a potential business strategy for a startup centered about our capstone project. It begins by discussing the broad industry trends that show which direction would yield the best go-to-market results. Next, the importance of adopting a well-defined strategy in terms of maximizing business potential is highlighted. From analyzing the industry as a whole in terms of its value chain, the most optimal place within this market a business should occupy will be determined. This is done by identifying the full chain and finding the strongest link. This is followed by an analysis of Porter's Five Forces, which is used to understand strengths and weaknesses in the chosen position within the value chain. Knowing these strengths and weaknesses is a key factor in shaping the business plan. The conclusions drawn from these forces will be presented along with suggestions for ameliorating any weaknesses identified.

After extensive analysis using these models, the section of the market that was identified to be targeted is the "mediator" segment - a segment whose products would form the platform that will serve as common gateway for devices measuring vital signs to communicate with health analytics services and vice versa. The platform the team is developing will allow for a much easier facilitation of data between entities specializing in either the analysis or measurement of health data. It essentially abstracts away the communication layer of the entire telehealth chain into an easy to use, robust and secure service. The strategy adopted to conquer this segment is to pursue an open-source double licensing model wherein royalty will be collected at a pre-determined rate only when the service or code is used for commercial purposes. Personal use cases will be free of charge.

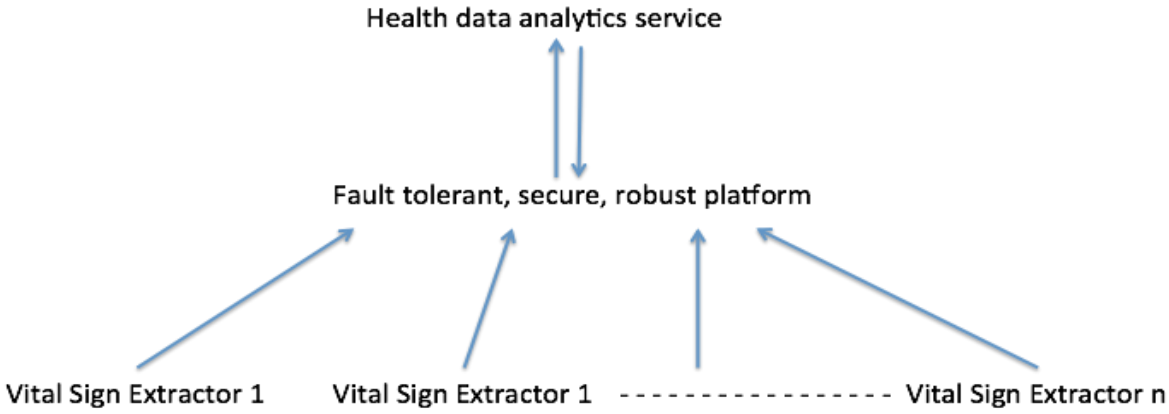

Figure 1: Telehealth Platform Hierarchy

We live in a highly interconnected world. Everything from common household appliances, such as refrigerators, to complex control systems, such as those found in vehicles are connected to information relaying services designed to expand their ordinary functionality. Our lives are becoming more and more dependent on being "connected," and as a result, many major social, technological, and economic trends in the last decade have been fueled by connectivity. Among these trends is the push for telehealth, which since the 1990s, has been a principal force in providing healthcare to underserved communities. (Miller 2007: 133-141) By leveraging major social trends towards connected privacy, availability, and transparency, it is possible to flesh out a telehealth strategy that addresses key impacts and vantage points associated with what will be

collectively known as the "connectivity trends." These connectivity trends are driven by a new, socially aware consumer that can greatly influence a product's adoption rate and popularity.

To provide context for these consumers, it is important to examine contemporary cases of social awareness with respect to connectivity. In recent years, the online health industry has been abuzz with issues of privacy, data storage, and patient protection. So much so, in fact, that entire companies have been spawned out of the necessity for these core three attributes (e.g. Veeva Systems.) Chief among these concerns is that of patient protection. Where as many medical devices remain "analog" with respect to connectivity (i.e. no internet/wireless access,) manufacturers are pushing to create connected equivalents of these products. With this comes the potential for new features, but also the introduction of many risks. In a more extreme (and relevant) example, data from an insulin pump that now communicates via Bluetooth low energy may find itself in the hands of third party advertisers. (Hall & McGraw 2014: 216-221) While this data is protected under the Health Insurance Portability and Accountability Act (HIPAA) passed in 1996, it is very likely that the patient will be unaware of what is occurring, since they are often surrounded by Bluetooth enabled devices with internet access. This creates a very real, and potentially dangerous situation: data can be intercepted and collected from medical devices, which can then be used for almost any purpose. And in effect, what you find is similar in many ways with the motivation behind the Digital Millennium Copyright Act (DMCA,) in which illegally obtained data is so abundant that liability is no longer held against companies who store that data.

From a strategy perspective, the contemporary social awareness of patient privacy, data storage, and protection significantly detracts from attempts at commercialization. The FDA Center for Devices and Radiological Health (CDRH) is the major regulatory arm that governs the development, implementation, and evaluation of products with telehealth features. Since the FDA is a member of the federal government, policymakers, who in turn are influenced by those who they represent (and in many cases, special interest groups,) have the ability to directly impact the regulation of telehealth systems. Negative social acknowledgement can therefore directly harm the commercialization of a new telehealth product by

influencing the implementation of stricter regulation. So the question remains, how can telehealth products hope to compete against an increasingly aware consumer base?

It is not feasible to simply promote ignorance of a very real and tangible problem: the digital age has long since begun, and next to nothing is primarily stored physically. If you cannot simply "wish" the knowledge away, then the next best thing is to approach the problem with the contrapositive: education. And this kind of education, one that targets the socially aware consumers (and in this case, patients,) is not strictly limited to the benefits of a product. In fact, it is critical to convey both sides of the technology. Net neutrality is one modern example where conveying only the benefits of a technology is more subversive than it is helpful. Major internet service providers attempted to only convey the benefits of discarding net neutrality in an attempt to mask its true purpose as a means for generating additional revenue. This had the effect of creating a nearly unanimous voice against the proposition. In conveying both the benefits and downsides to telehealth, we can best leverage the trend of social awareness in connected health care, and in turn, create a more positive public image.

The importance of influencing the socially aware consumer cannot be understated: coordinating a strategy of education rather than marketing can mean the difference between revolutionizing and downplaying the entire telehealth industry. With the advent of free online educational resources, it is simply not good enough to assume that a market segment will be ignorant about a product or feature. A good strategy will leverage this education and in some ways seek to exploit it, feeding a growing demand for high tech while at the same time appealing to the scientific side of a market segment. There are, however, many more sides to this "good" strategy that cannot be addressed by analyzing the consumer alone.

2.2.1 Strategy: What it is and why it is required

Strategy is both the art and science involved in the formulation and evaluation of key-decisions that will give an entity an edge over its competitors, attain a sustainable advantage and help achieve its long-term objectives. Delineating a strategy also helps the organization formalize objectives, and hence give it concrete direction. It helps one decide how to position a product and what decisions to make to maximize growth and profitability, as it is just as important when trying to enter a saturated market as it is when trying to enter a

nascent one. It is just as useful when trying to consolidate one's lead in a market as when one is trying to establish it.

The telehealth industry is still relatively nascent, but is rapidly expanding. Industry revenue was projected to grow $320.2 million in the five years leading to 2014, including revenue growth of 23.1% in 2014. From 2014 through 2019, the industry stands to reap the benefits of demographic, structural and legal factors. These include a growing aging population, advances in telecommunication and wearable technology, and statutes like the Affordable Care Act. As a result, industry revenue is expected to increase at an annualized 49.7% to $2.4 billion in the five years to 2019 (Morea 2014:9).

Although the top two companies, GlobalMed and InTouch Technologies, occupy 22% of the market share (Morea, 2014), the industry itself is rapidly expanding, and thus there is ample opportunity for this technology to enter the market and seize a sizable share.

However, given that the project is a potential spin off from an academic endeavor, and doesn't have (at least presently) the financial clout or business savviness of companies already in the market, adopting the right strategy becomes crucial to conquer and succeed in this market. Where the team temporarily lacks in business authority, it excels in technical skill and prowess. While it is acknowledged that this will have to change in the future, the team believes it can use this technical superiority to get a foothold in market.

2.3.1 The Strategy

As mentioned in the previous section, the team plans to leverage its technical prowess to its advantage. This involves building a robust easy to use, easy to integrate platform, and then treading the open-source route. In the current market, existing solutions are mostly "walled-gardens" with players providing proprietary solutions to all fields in telehealth (vital-sign extraction, collection, logging, analysis and result distribution). However, with the impending explosion of the internet of things, several third-party developers of medical hardware used for constantly monitoring vital sign data are expected (Lars 2014). The number of connected devices is expected to reach 50 billion mark by 2020 (Swan 2012). This data emphasizes a need for

an open platform to handle the software/communication part of the system. It is this opportunity that the strategy is designed to seize.

2.4.1 Industry Value Chain Analysis

The first requirement one must consider when creating a new business is how it should be positioned within the industry. For example, there are many levels involved in delivering medical devices to hospitals. A company wanting to produce a device must first buy supplies, and then usually contract out the labor to a manufacturer. Once the devices have been manufactured, the company will sell these medical devices to a distributor who carries the device in a catalog. Hospitals partnered with these re-sellers will then purchase these devices, process them, and finally hand them over to doctors who will use them as a part of the care they provide.

We can see that the value chain is long, and in many cases, not quite linear. Various market forces also make the chain asymmetric in that the value that is realized in each part of the chain is not evenly distributed. Many markets often have a particularly 'fat', or highly profitable part of the chain while the other sections fight for the leftovers. An example of this can be seen in the technology industry. In delivering cell phones to consumers, Apple, the company that designs and markets its products, clearly makes the most money, even though there are many other businesses involved in the value chain such as the phone manufacturers (e.g. Foxconn), distributors (e.g. FedEx) and carriers (e.g. AT&T). Unlike the consumer technology industry which arguably has started to reach maturation, telehealth is still a fledgling industry, which is still being shaped by early startups, new government regulations and many new technological innovations. Despite this, the telehealth industry value chain is no different in that the 'fat' part of the chain can clearly be identified.

From the IBISWorld report on telehealth, the industry value chain is shaped with several 'demand' players such as data processing and hosting services, hospitals, physical therapists and health & welfare funds to name a few (Morea 2014:14) . On the other side of the chain, there are several 'supply' players providing infrastructure within which telehealth services can operate. These include businesses such as mobile carriers, software publishers and hardware manufacturers.

Because of the wide variety of players in the industry, both supplying and pulling technologies to and from telehealth services, a simplified linear model was developed to illustrate the supply chain. This is shown below in Figure 2.

Figure 2: Industry Value Chain

From here, we see that the chain is longer than one might expect in that there are several intermediate steps involved with getting a device into a hospital. It is important to note that this value chain is a simplification of the actual chain, which is not linear and has many more steps involved within each major stop in the chain presented.

Due to the established nature of similar industries requiring the same suppliers and hardware manufacturers as the medical devices industry, suppliers and hardware manufacturers are well developed and highly competitive (Kahn 2015:24). Thus, the fat part of the value chain is located with the medical device designers. A good example of the power of this part of the chain are medical devices like Medtronic and Siemens AG - companies whose revenues exceed $17 billion USD**,** which is where we would like to position a potential startup. It is important to note however that there remain several large pitfalls in which a business situated as a device designer could become wildly unprofitable. Many of these have to do with regulatory approvals (Morea 2014:21) such as meeting FDA requirements (Medina, Kremer & Wysk 2012). However, once met, these businesses will be well situated within the chain. The team's particular business would circumvent many of these pitfalls by taking on the 'mediator' role within the medical devices industry. As alluded to earlier, this would allow participating medical devices companies to safely gather data from patients and provide additional services derived from this.

By identifying the best location within the industry chain, we can now analyze which parties in the medical industry would have interests driving the adoption of new medical devices. These parties would be ideal candidates to which a startup business could offer services. Performing this analysis also has the added

benefit of being able to predict the demand of the industry, which is a function of the size and strength of desire of its customers.

2.5.1 Potential Stakeholders

By its very nature, there are a plethora of potential stakeholders with key interests in the success of this platform. First, insurance companies would have an interest in the rise of tele-health monitoring systems, as having the ability to better assess the risk of patients could drastically change the way that insurance is calculated. From this, it follows that insurance payers and patients would see substantial benefits in the way that medicine is applied in that increased information about the patient will allow for more accurate diagnoses and more targeted treatments. This will ultimately help in lengthening and improving the quality of life.

Furthermore, the ability to take measurements of a patient's vital signs opens the door for alternative options to physically visiting the doctor – a practice continued for centuries. This is an enormous market currently valued at approximately $1.151 billion USD and growing at approximately 2.2% yearly (IBISWorld Report: Number of Physician Visits 2014:1) which presents a huge financial opportunity for the telehealth medicine. While certainly not a complete substitute, the telemedicine industry could replace large swathes of routine checkups and minor cases. This could serve as a boon not only for businesses looking to save on costs associated with granting time off to see a doctor but also for understaffed hospitals to relieve labor-related costs.

Governments and regulatory bodies such as the World Health Organization (WHO) would benefit from enhanced data collection in order to present new legislation or seek out cost cutting opportunities in relation to providing health care, and to help model and eventually mitigate the effects of contagious new diseases. With recent legislative moves seeking to expand health coverage in the US such as the Affordable Care Act, an increasing amount of financial support will flow into the telehealth industry contributing to the estimated 49.7% growth in industry revenue (Morea 2014:5) as a reaction to making care affordable to a wider swath of the population.

From this analysis, we can see that there are many large and powerful stakeholders with keen interests to either increase profits, or reduce costs with the aid of telehealth technology. This further validates

the team's judgment in the Porter's power of buyer's analysis. With so many parties interested, and relatively weak internal industry rivalry, starting a business in this sector of the industry should have the highest chance of success.

Now that the stakeholders have been identified, it is crucial to understand the role that the team's technology would play in commercial applications. Investors in a startup are investing primarily in the technology, and the team, so without a clear vision of how the technology will function, it would become impossible to market it correctly or draw up a business model.

2.6.1 Relation between Technology and Commercial Product

The ability to monitor vital sign information in real time has direct applications to commercial technology. The team aims at providing a common platform to developers with some sample vital sign extraction methods embedded with the idea that other companies build on top of this platform. This platform will enable developers to work on producing technology more focused on data extraction and/or analysis without having to spend inordinate effort establishing fault tolerant protocols, server security, and data privacy. The idea here is to position the platform similarly to how the Android operating system is marketed. Android is an open-source cellular operating platform that comes with several basic functionalities, but is meant as a platform where third-party developers can extend its functionality.

Different vital sign extraction techniques are analogous to new apps developed specifically for this platform. Revenue would be collected from licensing agreements between device makers seeking to utilize the platform for profitable ventures. Such a strategy would have a benefit of rapid growth due to its flexibility, but could have the potential downside of being slow to grow revenue. In all likelihood, due to the nature of the technology industry to rapidly grow in new sectors, this would be the best approach as opposed to a strategy that would involve keeping the platform closed and competing rather than collaborating with medical devices and technology companies.

With a defined position within the value chain, and an idea of how the technology in this capstone project will interact with the final product, we can now analyze the remainder of Porter's Five Forces to assess potential threats in the industry.

2.7.1. Porter's Five Forces

In his seminal article in the Harvard Business Review, Michael E. Porter outlines five major factors that should influence the approach one should espouse to be successful (Porter 2008). These are:

1. Bargaining power of buyers
2. Bargaining power of suppliers
3. Threat of new entrants
4. Threat of substitutes
5. Rivalry amongst existing competitors

Each of these forces will be analyzed in detail in how they pertain to our chosen industry in the following sections.

2.7.2 Bargaining power of Suppliers

The bargaining power of suppliers refers to the ability of business providing materials, data, IP etc. to a business further down the chain to ask for increased compensation or to switch to supplying a direct competitor. Due to the fact that this platform is largely based off the heavily subsidized, and highly competitive mobile phone market, the suppliers in this regard have little power to demand higher prices were the telemedicine industry to grow and increase their demand. Furthermore, the ubiquity of Android-based phones (International Data Corporation 2014) in the consumer market further weakens dependence on suppliers as our platform does not require a proprietary phones. Rather, a patient can just use a personal phone to use the monitoring platform.

2.7.3 Bargaining power of Buyers

The bargaining power of buyers refers to the ability of customers of the business to demand extra features, lower prices, or switch to a competitor. Due to the suggested licensing strategy of this product, the buyers in this case would be device manufacturers and potentially research laboratories and regulatory bodies looking to do research on the aggregate dataset of patient data. It is without doubt that the sale or licensing of data would come with some sort of public resistance, but if done in an identity sensitive way (and this is

already ingrained into the platform itself from its inception) this could open the door to many licensing opportunities. Since there are few alternatives to this system, the power of buyers to simply switch to another platform would be weak.

2.7.4 Threat of New Entrants

New entrants bring with them fresh ideas and a tenacity ideas to gain a share of the market from existing players. Ease of entrance, characterized by barriers to entry, will encourage more new comers especially if the industry is nascent and has potential for being lucrative. This precisely characterizes the telehealth market. We live in a connected world and with the Internet of Things verging on explosion, telehealth is one of the many industries that will receive a significant boost in market size (Lars, 2014.)

In such a scenario, it is anticipated that many players will prop up to get a piece of the pie. It is, therefore, of utmost importance to increase the barriers to entry to dissuade potential entrants. The team has done this by increasing barriers to entry in the following ways:

1. As the team is building a platform that will be leveraged by third-party manufacturers of health monitoring devices, the platform has to be robust, complete and easily joinable. Developing a solution that offers all three features requires a lot of developmental effort and time, which in turn raises the barrier of entry.
2. Once adopted, it'll be very difficult for our customers/partners to switch to a new platform.
3. The open source model works in the team's favor. Open source provides for collaboration and cooperation as opposed to competition. As the platform gains more adoption, a larger community would organically grow, further fuelling adoption.

By raising the barrier to entry for new entrants, the team has ensured a sustainable "economic moat" can be built.

2.7.5 Rivalry

Rivalry is greatest in an industry if competitors are roughly of equal size, industry growth is slow or if exit barriers are high (Porter, 2008). It can become particularly entrenched if the rivals are competing on

price. Size and financial might of the competitors play a major role in determining who comes out victorious in market share wars. The current major players in the telehealth market are GlobalMed and InTouch Technologies who together own 22% of the market (Morea 2014:25). However, the industry itself is rapidly growing and hence there is abundant opportunity for entry.

In the telehealth industry, the basis of competition is primarily based on product quality, which is determined by factors like product functionality, features, speed and ease of use. Other media of competition include breadth of services, marketing, customer services and training offered to clients. (Morea 2014:24)

Given this, the biggest threat for the team comes from the threat of rivals. Being a potential startup spun off from an academic endeavor, the team is definitely at a disadvantage in terms of the amount of money that can be spent on marketing and customer service.

However, the open-source strategy being adopted will help us overcome this hurdle. The team aims at revolutionizing the telehealth industry the way Linux revolutionized the mainframe operating system industry or the Android revolutionized smartphone operating system industry. By providing a free to use platform that can be leveraged by third party developers, the growth of a community of support and innovation is encouraged. This will be particularly successful because:

1) The proliferation of cheaper hardware and sensors is spurring the growth of independent original equipment manufacturers who can use the team's platform to build a complete solution.
2) With cheaper networking solutions and the growth of internet of things, several startups are developing devices each with their own proprietary communication protocols. Currently, there isn't a single platform that would allow these disparate devices to talk to each other. This reduces interoperability, which can stall growth of the industry a whole (Ghosh et al., 2011). As adoption increases, the open source platform the team is developing has the potential to allow for greater interoperability, thus further spurring the industry growth.
3) Existing companies have their proprietary platforms which can be licensed by third party developers at a hefty fee. By providing an open source solution, the team can win over more developers, thus gaining market share.

2.7.6 Threat of Substitutes

A substitute product is one that that offers similar benefits to customers of a company from other industries. The threat of substitutes describes the possibility that substitutes can replace the company's products. If the company is faced with a high-level threat of substitutes, it loses the control over the price that it can set to sell to customers, since customers can easily switch to its substitutes if the price is set too high. Therefore, it is important to analyze the threat of substitutes given it can affect profitability of the company and the industry. In this section, we will discuss the problem our technology tries to solve, what substitutes there are that solve the same problem, and what the level of threat the substitutes pose on our technology.

Health care is by far one of the most expensive industries in the United States. Readmissions lead to a huge hospital costs. According to Agency for Healthcare Research and Quality, there were 3.3 million readmissions in the United States, which cost hospitals about $41.3 billion USDs (Heins et al 2014). By developing a system that can reduce trips to the hospital, helping people get in touch with their own health, and mediate the link between patient and physician, we can cut down the waste in health care. Specifically, the problem that our technology tires to solve is to reduce hospital readmissions by reporting patients' vital signs (including heart rate, blood pressure, energy expenditure and body temperature) to physicians, who will in return give patients' health advice remotely to keep healthy.

Except our technology, traditional medical devices and modern technologies may achieve the same purpose and become our substitutes. Traditional medical devices are single functional devices such as thermometer, sphygmomanometer, EE monitors and etc. Patients can use those devices to record their vital signs by themselves and remotely consult physicians for feedback with recorded data.

Traditional medical devices have advantage on price over our product. They are relatively cheaper to Bluetooth-enabled medical devices required by our technology, which may lead patients to adopt traditional medical devices instead of our technology. Another weak point of commercializing our technology is switching cost. Since our technology is open-source, we should obey open source philosophy, which states that open-source software should be freely used, changed and shared (Open Source Initiative 2015:1). As a

result, there is no reason to charge termination fees to patients when they decide switching to substitutes. However, our technology differentiates from these substitutes in three aspects which can stick patients to our product:

1. Instant data transmission.

   Our technology helps patients to transfer health data to physicians. If the patients use traditional medical devices to monitor their vital signs, they have to record the data and call or email physicians for advice. This procedure is both time consuming and boring. The patients may give up tracking their health conditions after several trials. With our technology, the health data gathered by Bluetooth-enabled devices will transmit to smartphones immediately and then transfer to physicians by cellular network. The instant health data transmission saves time for patients and physicians.

2. Easy access to health data.

   Our technology provides convenience for patients to check their vital signs record. Traditional medical devices are separate. To check vital signs history, patients need to check on individual devices. Our technology is based on smartphone. All measured vital signs will store on the phone. As a result, patients can check their health data anywhere anytime with their phone.

3. Expandable functionalities.

   Our technology is expendable because we built an open-source platform for telemedicine. Developers can write more applications using our platform. Therefore, in the future, our technology has potential to provide more useful functionalities as well as benefits to patients and physicians by the contributions of third-party developers.

In addition to traditional medical devices, modern technologies could also pose a threat to the platform we are developing. Following are two existing modern technologies serve as substitutes of our product.

1. Motorola - Moto 360™

Motorola has recently released a smartwatch called the Moto 360™. Other than being an accessory to a smartphone, it can give basic biomedical data such as skin temperature, heart rate and energy expenditure. This platform could be developed further to come closer to the product we envision, but this would require substantial modifications to the current product.

2. Microsoft – Kinect

Microsoft's Kinect system has cameras which can be used to detect heart rate, energy expenditure and even mood by sensing the changes in flood flow in the face (Zhang 2012:4-10). Microsoft is currently focused on integrating the Kinect system with its Xbox One gaming console. However, this does not stop third-party developers from using this system to provide in-house medical diagnostics such as respiratory surface motion tracking (Alnowami et al 2012), 4D thermal imaging system (Skala 2011 et al:407-416), and otoneurological examinations (Dolinay et al 2014).

With considerations about price, switching cost, and differentiators with respect to traditional medical devices and discussion about modern technologies, the threat of substitutes of our product is moderate.

2.8.1 Major Conclusions

In this report, it was concluded that the best part of the industry for a startup centered about the technology being developed in our capstone is as a mediator between medical device companies and data services. This is the fat part of the chain. A Porter's Five Forces analysis was also conducted, and it was determined that with small tweaks to the planned dual-licensing strategy, all of the forces can be reduced to weak or moderate in strength. It is important to note however that this favorable analysis in no way guarantees success. As will be discussed in papers to come, the largest hurdles lie not within the placement of the startup, but with issues like avoiding IP prosecution, gaining the necessary industry approvals, and building up a reputation for being protective about sensitive health information. These issues are not easy, and some will be addressed specifically in future papers.

3. Intellectual Property Strategy

3.1.1 Introduction.

This section of the report is dedicated to explaining the strategy the team would like to adopt with respect to intellectual property. While the final goal is to create an open source product, the strategy is to attempt to patent the system architecture in order to safeguard the platform against patent trolls. We will first provide a brief outline of the technology, and then dive into the strategic reasoning behind patenting this particular part of the project in terms the competitive and strategic advantages it would give the team. We will then talk about what the risks of not patenting are and how these hurdles can be overcome.

3.2.1 The Aspect: System Architecture

Being the platform that forms the intermediary between data collection and analysis mandates robustness, fault tolerance and ease of use. For widespread adoption, not only is it necessary to be first to market, it is also necessary to provide an easy to integrate solution. The application programming interfaces that would be provided should be seamless enough for people in the adjoining industries to build upon. The product's system architecture caters to all these requirements. We believe the innovation that went into meeting the constraints imposed by the requirements puts the team in good stead to warrant a patentable system architecture. This architecture allows for an instant state recall, clever storage and persistent reconnection as broad means to implement fault tolerance and robustness - aspects that are quite novel and developed from needs specific to handling sensitive health data.

3.3.1 Problems with Software Patents

Software patents are notoriously difficult to obtain due to the difficulty involved in proving the novelty and obviousness of them, unless they are extremely specific. Common problems include

1. Deeming software is math, and math is not patentable.(Jones, 2009)

2. Software programs being different than electromechanical devices because they are designed solely in terms of their function. While the inventor of a typical electromechanical device must design new physical features to qualify for a patent, a software developer need only design new functions to create a working embodiment of the program.(Plotkin, 2002)
3. Computers are deemed to be the "designers" and “builders” of the structure of executable software. Software developers do not “design” the executable software's physical structure, but merely provide the functional terms.(Plotkin, 2002)
4. The large number of micro-niches in software and relative few examiners mean patent examiners seldom have sufficient know how to recognize if the technology disclosed is innovative or novel enough.(Bessen, Meurer, 2008)

For the reasons stated above, it may not be possible for a small team of engineers to mount a serious patent application. In order to overcome this, we plan on leveraging U.C. Berkeley’s Office of transfer office, the Office of Technology Transfer, to help design and formulate a strong application to provide the best chance of being awarded a patent.

3.4.1 The Strategy

Applying for a patent is best in this case as it would protect the business from the threat of patent trolls and also would set a legal basis by which the business can protect its work. Forgoing patents could spawn copycat businesses that would patent this architecture, and could ultimately lead to the business being coerced into paying licensing fees for technology it originally developed. Applying for the patent protects the product irrespective of the outcome of the application for two reasons:

1. Approval of the patent will ensure that other competitors do not develop similar technologies, and thus raise barriers to entry. While it is true that the business may ultimately pursue an open-source strategy to encourage an organic growth of a strong third party community (which in itself raises barrier to entry), being awarded the patent ensures that the economic moat the team aims at building is deeper and wider.

2. Failure to get approval of the patent reassures the business that the same technology likely will not be awarded to a patent troll, giving peace of mind.

It is true that established competitors are much larger and financially capable than a startup this capstone group can produce, but the playing field can be leveled by leveraging the resources of the university to apply for a patent. This will both ease the financial burden, and cast a much larger enforcement net over the patent then could be done alone.

3.5.1 Risks of Not Patenting

If the backend architecture of the platform is not patented during the development phase, the team would leave itself open to the threat of having the architecture stolen and patented by another entity. As a consequence, the team could lose revenues from licensing, or even have to pay for technology the team invented in the first place. This could happen if some other group invented similar back-end architecture and patents it before the group's project is finished and released as an open source software. In addition, due to the highly collaborative nature of these projects, there is a risk that external developers would file a patent for the technology themselves. This would distract the team with crippling legal proceedings rather than focusing on further innovations.

Because of the risks of not patenting, it is advisable that patentable parts of the project are quickly pursued. To achieve this, an internal search utilizing university resources such as the Office of Technology Licensing (OTL) must be conducted to discover any prior art that may stand in the way of granting a patent. Legal experts should be then involved as quickly as possible to explore the potential patentability of the system architecture. Upon a thorough literature review by these experts, their advice will be taken and put into action.

In the case that it is found that a patent would be unlikely to be granted, the group would focus on making the platform open source as soon as possible in order to reduce the risk of infringement of patent rights. This establishes a record of prior art which can be used as a legal defense against patent trolls. A dual-

licensing model will then be adopted to produce revenue. With dual-licensing, clients can use the technology under the condition that they redistribute their technologies as open source as well. If these entities wish to commercialize technologies utilizing this technology however, they will be required to pay a licensing fee on a per-use basis.

In conclusion, patenting the backend architecture provides a shield from legal problems and enables safe cooperation with external developers to speed up the development of the platform. In the unfortunate case that the patent is not granted, the only way to reduce legal risk is to release the platform as quickly as possible to establish prior art against potential litigation.

4. Technical Overview & Context

The nascent field of telemedicine is one with enormous potential to dissociate quality health care from the confines of top notch hospitals. Tele monitoring is the practice of providing electronic and remote access to medical equipment, procedures, and diagnostic tools. Residents of remote areas that do not have access or cannot afford quality health care can be provided with services and diagnostics through tele medical platforms that previously would not have been achievable with traditional systems. In essence, telemedicine is the future of medical infrastructure, providing an inclusive, inexpensive platform by which health care be distributed to all socioeconomic tiers. There is, however, a major barrier to entry for new technologies that is pervasive in the healthcare industry: cost effectiveness. Telemedicine is no exception to this barrier: attempts to quantitatively determine the cost effectiveness of deploying tele medical systems have been largely subjective and difficult to generalize. This lack of generalizability makes telemedicine a feasible option for high end hospitals, but can prevent deployment in more rural, underdeveloped areas (Whitten 2002.) The project seeks to work around this barrier and leverage preexisting technologies to create an easy to use, robust, and fault tolerant tele medical system. Additionally, the project seeks to unlock the potential for research studies in telemedicine by exploiting wide availability of smartphones.

The core philosophy of the capstone project focuses on three principles: fault tolerance, robustness, and ease of use. These principles manifest themselves in every aspect of the project, and they guide the design and implementation of the various software modules that comprise the current system. One of the major challenges that the team has faced in ensuring these core philosophies are present in all work we contributed, however, emerges from the desire to leverage the ubiquitous nature of smartphones in the developed world. In 2009, smartphone market penetration in the United States was 25% - a quarter of the mobile phone market. (Falaki 2010) By 2012, this number had increased to 46% of adults in the U.S. owning a smartphone. Of this 46%, 20% of individuals surveyed indicated they owned an Android device. (Smith 2012.) As such, it was decided that the debut platform for the project would be the Android operating system. We would provide a tele monitoring programming library that Android developers could utilize to create tele medical applications that exhibited the core design philosophies. Android as a system, however, does not necessarily adhere to these design principles, which is where the Masters of Engineering team comes in.

Each member of the team was responsible for expanding or improving on some portion of the Android operating system to create a new module that adhered to the principles of the project. These include the creation of a custom Bluetooth stack, the expansion of a Heart Rate sensing algorithm using face detection, and the creation of a blood pressure sensing algorithm. The first of these, the custom Bluetooth stack, effectively enables the integration of Bluetooth enabled sensors and actuators which can be used in validating results obtained by the latter. Thus, it plays a critical role in validating the pilot system as a whole, and puts the "monitoring" in tele monitoring. For the duration of the capstone project, Adarsh Mani and I, Phillip Azar, were tasked with the design and implementation of this custom Bluetooth stack. Since both Adarsh and I contributed to this task, this paper will be written from my (Phillip Azar's) perspective, and focus on the design and implementation of the various packages within our Bluetooth stack. My work began by analyzing different implementations for tele medical sensor networks, and how they might apply to our project.

5. Knowledge Domains and Background

This project is an immediate continuation of the works conducted by Aranki *et al.* in 2014, wherein a smartphone-based system for remote real-time tele-monitoring of chronic heart failure patients was proposed (Aranki *et al.*, 2014.) The pilot study, which was conducted in collaboration with the Feinberg School of Medicine at Northwestern University, assessed the feasibility of tracking energy expenditure of patients with chronic heart failure using accelerometer and GPS tracking data available from the smartphone. Data collected from the patients was transmit securely to a server, where said data was processed and made available to physicians responsible for care of the participating patients. This process was not without challenges, as the paper illustrates towards the conclusion.

Through the pilot study, Aranki *et al.* identified several key challenges that remain a focal point for our works in this project. Firstly, Aranki *et al.* identify battery life as a primary concern for smartphone-based tele-monitoring. Since data was being collected on the phone in real-time from various sensors simultaneously and subsequently transferred to a server via internet, the processor on the smartphone is in a constant awake state. I aim to incorporate low energy technologies and designs in order to alleviate this issue. Another primary concern discussed by Aranki *et al.* is the need for a more systematic approach to tele-monitoring to make large scale deployment feasible and solve some of the other challenges identified in the paper. Incorporating this systematic approach is possible through the tele-monitoring library, where we can control the design of each and every piece of the proposed system that the application developer interacts with. Moving forward, it is necessary to examine how other tele-monitoring systems approach smartphone integration.

Various tele medical systems integrating a smartphone as the primary telematics hub, similar in design to our own, have been proposed (Seto et al. 2012)(Sultan, Salys, Mohan 2009)(Gund et al. 2008.) Smartphones are a desirable choice for a telematics hub due to their wide variety of sensors, communication electronics, and widespread social acceptability. Systems such as these are important because they represent a not-so-distant future of telemedicine: one dominated by low cost, low footprint, and connected communication devices. One such communication device within nearly all modern smartphones, Bluetooth,

has become a *defacto* standard for peripheral and peer to peer communication. These peripherals can range anywhere from external audio speakers to headsets to store devices. Additionally, an evolutionary successor of the Bluetooth protocol, Bluetooth Low Energy (BLE,) has found a special home in the Medical industry, due to its low footprint and low power dissipation (Omre, Keeping 2010)(OTT 2010: Socket Mobile, Inc. White Papers.) When the team was deciding on a communication protocol to move forward with in the tele medical library, I had to consider not only the past work done on a specific protocol, but also the practical implications of tuning that protocol to the libraries principles. Bluetooth, although promising at surface value, proved to be less desirable than other options.

One downside to using Bluetooth in the library surfaced during preliminary research. As I mentioned above, BLE is an evolutionary successor of the Bluetooth communication protocol. Its' protocol stack maintains the same structure as its ancestor, having two components: a controller and a host. In terms of the network stack (i.e. the 7 layer model,) the controller comprises the physical layer and the link layer. These two layers are effectively the "low level" components that handle much of the initial networking. The host comprises the higher level functions, and is typically run on the application side. Bluetooth is similar, but unfortunately, is incompatible with BLE (Gomez, Oller, Paradells 2012.) This makes implementing both on a client side an arduous task. Since BLE remains a fairly nascent successor to Bluetooth, the team decided that it would be better to first implement and tune a regular Bluetooth stack, and then work towards a BLE stack, which could address the second concern: energy consumption.

Another downside to using Bluetooth over other protocols such as ZigBee and Z-Wave was the high energy consumption of Bluetooth. In the project, I had to make many design decisions under the assumption that the phone I would have access to was in no ways a modern smartphone. In fact, the test device was using Android 2.1 Éclair, released in 2010. Many of the optimizations that came from later Android releases, such as Kit Kat (4.4) and Lollipop (5.0,) could not be relied on for communication and processing efficiency. The primary reason for utilizing this earlier version of Android was to ensure that the tele-monitoring library could be applied on as many devices as possible. Bluetooth, while not consuming as much energy as Wi-Fi or CDMA/LTE/GSM radios, can still be highly inefficient if left on or transmitting for long periods of time

(Balani 2007.) Competing technologies, such as ZigBee and Z-Wave, do not consume as much energy as Bluetooth, due to factors such as low data rates and short transmitting range. It would seem then, that in order to effectively utilize Bluetooth in a tele-monitoring system, one would have to address two major concerns: energy consumption and backwards compatibility.

Android does a very good job at handling the latter concern with respect to functionality. Backwards compatibility is one of Android's major advantages, allowing later API's (application program interface) to seamlessly interact with previous API's. While some classes may be deprecated as API's continue to evolve, there is still very little in the way of headache when reconfiguring an Android project to a more recent API. With immediate respect to Bluetooth, however, this concern is not so readily addressed by the operating system. I had to make a choice to either develop Bluetooth or BLE from the onset of the project. Doing both initially would make the effort intractable for the scope of the project. After discussion, it was decided that classic Bluetooth would be the best starting point, with an eventual development of BLE for the pilot study. With respect to energy consumption, developing and optimizing with Bluetooth as opposed to BLE would ensure that if and when the transition to BLE was made, the energy consumption of the application would have already been characterized with a much more power hungry mode. Additionally, the Bluetooth stack in Android is much more flexible then the BLE stack: I could fine tune and tailor the Android implementation to suit the projects core principles of fault tolerance, robustness, and ease of use. This can be of significant use to other tele medical teams with similar goals, addressing the need for a communication protocol that is self-contained and resistant to a fault.

One such application follows from a recent study conducted in 2008 by Gund *et al.* They mention that one challenge they faced was converting patients who were not comfortable with technology to their system. This is a major challenge associated with smartphone based tele medical systems, as some individuals have no desire to be constantly monitored. Another issue mentioned was cost: as hardware becomes more sophisticated to improve user interaction, the cost of this hardware increases drastically. (Gund et al. 2008: EMBS) These two challenges represent an excellent case for the implementation and design to handle. Bluetooth, and technologies that utilize Bluetooth, are not generally expensive. Much of the cost associated

with using Bluetooth results from implementing it properly, i.e. developing the software that will allow Bluetooth to be utilized correctly. As this software becomes more and more complex, the minimum hardware required becomes more expensive. The system will implement Bluetooth in a manner that can address these issues by remaining as simple as possible without sacrificing functionality. Thus, individuals who are not comfortable with technology can be presented with the minimal amount of hardware possible - keeping costs down and encouraging participation.

After the team established the base of the communication stack and decided on Bluetooth, designing and implementing the module was the majority of my work. All subsequent research I conducted was aimed at determining the various forms in which candidate "sensors" transmit data. Here I define a sensor as follows: any device which can communicate via Bluetooth or BLE to deliver information about some physical phenomenon. Two such candidate sensors are the UA767-PBT and UA651-BT blood pressure monitors by A&D Medical. The former is a Bluetooth enabled blood pressure monitor can be used with a smartphone application that supports wireless data transfer (which is essentially what Bluetooth does) to transmit up to 40 readings at a time. The latter is the evolutionary successor, utilizing BLE (marketed as Bluetooth Smart) to communicate with smartphones and store/transmit up to 30 readings at a time. The Bluetooth stack will be designed to function in a heavily multithreaded environment, where many function calls are asynchronous, i.e. they do not block, and instead utilize a *callback* to inform the application developer when an event of interest occurs. The environment is also networked insofar that the system is constantly communication over internet with a server, which allows for the storage of both data and backup information.

6. Methods and Materials

To go about designing and implementing the Bluetooth module was not a trivial task. In this particular software system, where many components are interrelated and communicating, there is no single point at which one can begin development and end development. The traditional paradigm of *imperative programming*, which is often seen as a top down approach, is not as effective. Instead, I can utilize an *object oriented* approach, which greatly simplifies the development of our system. As I mentioned in the previous

section, the Bluetooth stack has two primary functional components: the controller, and the host. In terms of software, these can be thought of as separate objects, with different properties, purposes, and capabilities. In fact, a telemedicine system as a whole lends itself to an object oriented programming approach. Each component in the tele medical platform, from the communication protocols to the physical sensors, can be represented in software as an object. These objects interact with each other and can be programmatically manipulated to fulfill the requirements of the system as a whole. This is (while greatly simplified,) the essence of object oriented programming. And this is the guiding philosophy behind how I proceeded to implement and design the Bluetooth module.

Bluetooth, as discussed on the Android Developers website, has a lifecycle that essentially begins and ends with the Bluetooth Adapter (Google, 2015.) The Bluetooth Adapter is the primary point of contact for the application developer when accessing Bluetooth features through the operating system. The adapter is switched on by asking the Android operating system to power on the hardware. Then, a software representation of the default adapter for that specific phone is obtained statically. This adapter can then be used to scan for and pair with new, unpaired devices, or to connect to known, paired devices. The adapter can also initiate un-pairing with paired devices. Every Bluetooth device will initially begin as an *unpaired* device that is found via scanning. Then, once the pairing process has been completed, it will be a *paired* device. The Android operating system does not make a distinction in software between a paired and unpaired Bluetooth device – instead, every device is a "Bluetooth Device" object that is either listed as paired or unpaired. Once connected to a paired device, the adapter choreographs data transfer between the client device and the host device over a radio frequency communication (RFCOMM) channel. When communication ceases, the adapter will close the connection. Any Bluetooth Device objects that were created remain, albeit in an unconnected state. When all is said and done, the application developer can choose to switch off Bluetooth by requesting it from the Android OS.

I began with an example program that helped me understand the dynamics of Bluetooth communication in Android in more depth. This simple application, dubbed "BluetoothChatApp," was the beginning of the Bluetooth module. I chose to start with an example program because of two major reasons.

First, I knew next to nothing about Bluetooth at the beginning of last semester. This application would give me a chance to dive in headfirst. Second, building on examples is an excellent method of engineering. There is never a good reason to reinvent the wheel - this is especially true when working with software. The example consisted of two *objects*: a BluetoothChatService, and a BluetoothChatActivity. The chat service was the primary functional unit of the sample application, and consisted of all of the purported "back end" programming. This included turning on Bluetooth, establishing a new connection, maintaining existing connections, and closing existing connections. The chat activity was the "front end" of the sample application. It consisted of everything the user could see, including messages sent from one device to another, which device was connected, and the duration of the connection. These two basic subdivisions outlined the functional hierarchy of the module: I would create a backend "manager," and a frontend "representative." In software engineering, this is a common method of designing applications - many professional teams of engineers are split into three classes of developers: frontend, backend, and full stack (which is a combination of the two.) I would essentially approach the development of the Bluetooth module from a full stack approach, building both the back and front end in tandem.

This development strategy had two significant implications for the resulting structure of the module. The first of these implications was a challenge in ensuring a functional and philosophical separation between the test application and the module code. Fundamentally, I was developing a module that stood alone as a part of a larger library. In Android, this means that much of the information and *context* that may come from the application (user) side is more or less invisible to the module. While I can request *context* from the application, I cannot design any dependencies on the application in the module. The module had to completely generalized to any application, in a "familiar too many but specific to none" type implementation. The second implication was that testing the module could be somewhat biased. In designing a library of modules, it may be desirable to test that a specific module functions in a wide range of situations and applications. This would ensure *robustness*, one of the core principles of the tele medical library. Since I had essentially one test application throughout the development cycle, I only had one point of view for testing. This is fine initially, but in terms of an eventual deployment to a broader audience, more thorough testing

would be required. Following all of the aforementioned considerations, I was able to construct the module with help of my advisor.

The Bluetooth module consisted of a main package that contained the core backend classes. These backend classes were responsible for maintaining Bluetooth uptime, recognizing paired and unpaired devices, managing connections, and transferring data. The BluetoothService class is the main functional component of the main package, handling the majority of functionalities that the module conducts. It has no dependencies, is a singleton, and can stand alone, not requiring any other class in the library to function in full capacity. There are two ways in which the BluetoothService class implements the core principles of the library. The first is in the manner by which it maintains incoming and outgoing connections. In the traditional Android Bluetooth stack, connections are handled on a case by case basis - when one connection is established, the recommended implementation on the Android Developer website does not provide support for handling more connections. I added a system of state variables and cyclic connection acceptance to allow the service to continually service new connections from different devices. This allows the service to be fault tolerant and facilitate reconnections to devices that have poor connectivity. Additionally, the service can use reflection to manually open connections on certain ports to facilitate connections to older medical devices. For example, in our tests I determined that the UA767-PBT from A&D is only able to communicate over RFCOMM channel 5. The reason for this constraint is unknown, but nonetheless the module can accommodate for this and communicate with the device. The service can then be made more robust and functional without completely overhauling the design.

Bluetooth itself is not very complex in the Android API - rather the complexity I faced arose from implementing Bluetooth in a manner that satisfies the core principles of the library. To this end, I avoided utilizing any design that might necessitate an overhaul in future works. One example of a dangerous design might be a singular device class that represents all known devices. This prevents extension into a particular subset of devices, for example, unpaired blood pressure devices, because the singular device class will contain all information pertinent to both. Essentially, dead-end designs largely violate the principles of object oriented design. In the main package, to represent devices, I used two classes: PairedBluetoothDevice and

UnpairedBluetoothDevice. These classes essentially wrapped the Android standard BluetoothDevice class with extra functionality. This design both allowed the fulfillment of core functionality requirements, but also allowed for extended modularity should a developer choose to make more specific device classes. One aspect of the main package has not yet been discussed, and it pertains directly to data communication. While I have discussed the higher level mode of communication with external devices, internal communication remains undisclosed.

When data has been acquired by the module, the application contains all relevant information in a structure known as a BluetoothSendable. The BluetoothSendable is a generic wrapper object that contains a data structure that represents the data that has been received in its original format. When data is received from an external device, the paired device object associated with the external device uses an internally stored *converter* class to properly decode the information from a byte stream to a formatted data structure. Once the data has been formatted and stored in a sendable, the module can pass the sendable freely between various classes, and the developer/user can be assured that the data will be secure, uncorrupted, and consistent. The means by which these sendables are passed throughout the library follows a programming model known as *event driven programming*. In this model, each core class in the main package has a corresponding *handler*, which maintains a list of interested parties, known as *listeners*. When an event occurs that necessitates notification of these listeners, the handler (which, in our case, is the class itself,) is called upon to notify the listeners that the event has occurred in an asynchronous manner.

In the Bluetooth package, each handler is one in the same as its corresponding class - that is, the service and device objects are in fact handlers. Initially, I followed a paradigm that was adopted elsewhere in the library, utilizing an event handler interface to contractually obligate any implementing classes to utilize a method known as notifyListeners. This design decision was made in order to remove the burden of notifying many listeners in the registered class - an operation that could potentially take a large amount of time to complete. After revisiting this decision, I came to the conclusion that a single listener was a much more reliable implementation in the case of both the service and the device classes. Figure 1 below highlights these two designs.

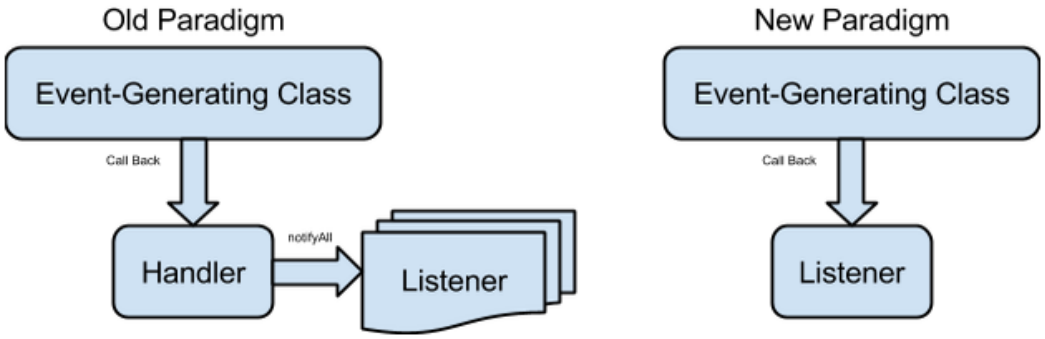

Figure 1: Handler/Listener vs. Single Listener design patterns.

In the case of the service, there are a number of scenarios in which many listeners would potentially cause problems. These scenarios are largely in the domain of user decision. For example, when the Bluetooth Service received an incoming connection, the application developer would receive a callback to indicate whether or not they want to accept this connection. In a many listener design, there could be a conflict in the answers relayed back to the Service, which creates a "voting" scenario that is not supported in the current design. A single listener eliminates this potential problem - and simplifies the communication of events. In the case of the listeners, I maintained the same design as previously discussed, in which I provide an interface to implement each listener. This allows developers to create listeners that are event specific rather than device specific by leaving certain methods empty

Previously, I used an event response paradigm, where events are encapsulated into objects and distributed to listeners asynchronously, as opposed to discrete event callbacks implemented in the listeners. There were a few reasons why I chose this approach. The first was backwards compatibility. I could easily add new events without changing the internal dynamics of the handlers and listeners. The event responses could simply contain new information and be passed through to the listeners in the same fashion as they were before. The second benefit is one of encapsulation. By using event response objects, I was able to succinctly store all relevant information into one data structure rather than having to pass each piece of information separately. This greatly reduces organizational complexity when notifying many listeners with many objects. Reduced organizational complexity therefore lends itself to reduced chance for internal fault, which is a core principle of the library. The problem with these event response objects arose during the final push towards a

complete Bluetooth stack. When the handler/listener paradigm shifted away from multiple listeners, the event response objects quickly lost their novelty. There was no longer a need to encapsulate events into objects for distribution, since each handler only had one listener. Additionally, in moving towards a more interactive service class, certain callback methods needed to return values to the service, while other did not. The event response objects did not facilitate this, as their distribution through one notify method resulted in all events generating return values. This can be seen in figure 2 below.

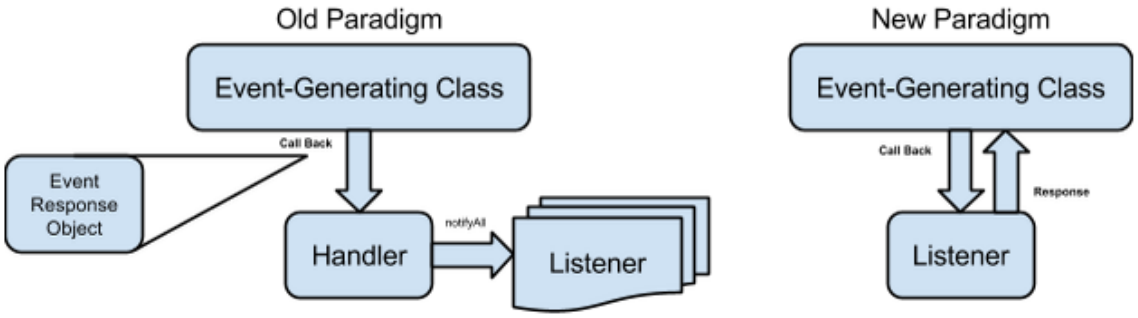

Figure 2: Event Response Object vs. Discrete callback paradigms

The event response object on the left hides the details of the event from the original calling class by passing it to the handler via a generic function call. On the right, I can call discrete callbacks with known return values to require feedback from the application directly. This design effectively replaced the event response package, and thus completed our Bluetooth stack.

Completion of the Bluetooth Stack was a critical milestone that allowed our team to determine a few key facts. First, it quickly became clear that classic Bluetooth was not widely used for health devices. While the utility of implementing a classic Bluetooth stack that follows the library design principles is known, its relevance to health devices is limited, as many no longer use the classic Bluetooth protocol. The primary reasons for this are the large power consumption of classic Bluetooth, coupled with the synchronous nature of the Bluetooth stack. The second key fact was that the computational paradigm of Bluetooth in Android was a complicated mix of both asynchronous and synchronous behavior. In our current design, the user is required to maintain two sets of threads to maintain data transfer and connection acquisition. This generates

excess CPU usage that could be avoided through the use of asynchronous callbacks or notifications. This is the point our team began to develop BLE for the project.

The design of BLE was guided by a need to eliminate some of the major problems associated with classic Bluetooth. This includes data transfer that is both synchronous and asynchronous as well as large power consumption. In BLE, once a device is found, a persistent connection is not needed between the client and server to transfer data. Instead, once a connection has been established with the BLE server device, a notification request can be registered by the client device for a particular piece of data. Then, when this piece of data (known as a Bluetooth Characteristic) changes, the client device will be notified by the server device, and the information can then be sent to the client device. This type of asynchronous communication utilizes callbacks, which continues to follow the event driven programming paradigm I followed in classic Bluetooth. The difference here, however, is that now there is no synchronous component. BLE is a completely asynchronous protocol, with callbacks used for interactions between the client and the server. The benefits of this event driven programming are well characterized (Frank et al. 2002,) over a threaded paradigm. Many resource constrained embedded platforms utilize event driven programming models. In our case, callbacks were implemented at each point of the BLE package. Upon completing BLE, I began to develop the final piece to the puzzle, which was the Bluetooth Health Device package. This package, which further abstracted the listener implementation for BLE, would serve to further simplify interacting with health devices that support BLE.

The Bluetooth Health Device package essentially implements the IEEE 11073 protocols defined for various health devices communicating over Bluetooth. The main class, Bluetooth Health Device, essentially defines a device that could be either a classic Bluetooth device or a BLE device, and contains any number of health listeners. In the library, I implement three basic health listeners: heart rate, blood pressure, and temperature. The difference from the standard device classes in the classic Bluetooth package is such that the Bluetooth Health Device class can contain any number of health listeners. Health listeners, in a nutshell, represent an interested party for the acquisition of that specific health data point. A Bluetooth Health Device does not contain an explicit list of these listeners. Instead, an encapsulated class known as Bluetooth Health

Device Router maintains the list of listeners. The routers job, then, is twofold: to be responsible for maintaining the list of health device listeners in a thread safe manner, and to ensure that when a data point is received, it is routed to the appropriate callbacks. One challenge I faced was synchronizing access to the list of listeners. Since the addition of listeners is an external function, and the routing of a data point can happen concurrently with this if a data point is received, neglecting synchronization can create scenarios where behavior is inconsistent with a single threaded model. Figure 3 below illustrates this erroneous behavior:

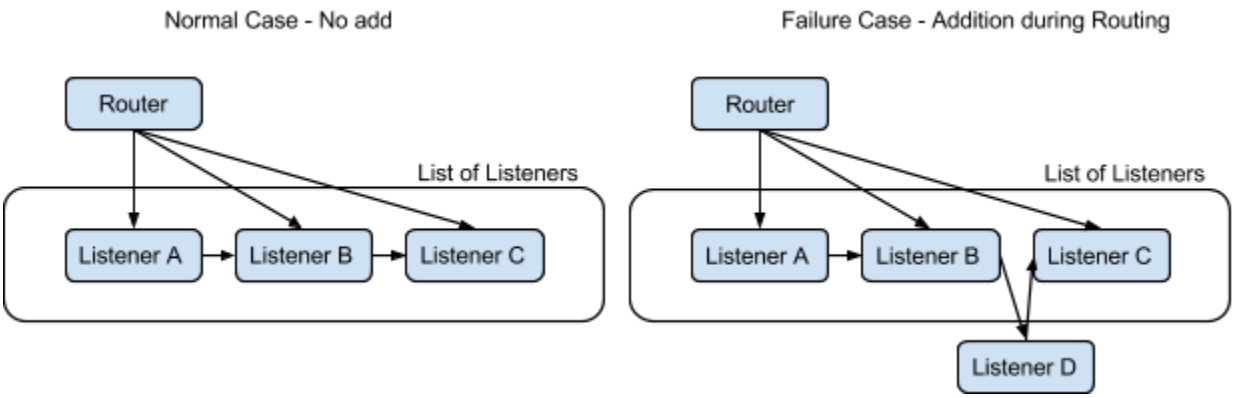

Figure 3: Non-deterministic health router behavior when using unsynchronized listener lists.

In this case, consider if access were unsynchronized. What would happen to listener C on the right? Would it receive the data point, or would it be ignored? The answer is uncertain. To prevent this, for adding listeners, I synchronize on the associated list. For callbacks, I synchronize both on the router and the list of listeners. These synchronizations will guarantee deterministic behavior of the router, since now the list cannot be traversed and modified simultaneously. I considered the usage of a ConcurrentLinkedQueue construct in java, which has a weakly consistent iterator and thread safe implementation. I chose against this because of the consistency of the iterator. Weakly consistent indicates that it is not guaranteed that during iteration of the list, the copy used by the iterator may not reflect the current state of the list. This nondeterministic behavior is particularly dangerous in a health application, as it can cause notification failures, wherein one listener is not notified of a new data point. Overall, the focus of the Health Device package is no different from the classic device classes: it seeks to simplify interaction with health devices and maintain fault tolerant, robust connections with those devices.

The application layer is the final piece of the puzzle, where all of the above design decisions and implementations come together. I contributed to the development of the pilot study application, which implemented the BLE stack and allowed the pilot application to communicate and retrieve data from the UA651-BT blood pressure monitor. The pilot study, which our team conducted, contained a number of key lessons learned, and continues to guide iteration of our design for future studies.

7. Results and Discussion

For the overall project, the metric of success was the ability to implement a pilot application on Android for use in an internal study with human subjects. This would see the smartphone realized as a main hub for all tele medical activities, including remote sensing, data storage and collection, and client/server communication with the cloud. These "trials" would then be analyzed and used to correct issues discovered with the library as a whole, and essentially, lead the project into the next steps. For the scope of the capstone project, the metric of completion was to design, implement, and test the Bluetooth module to meet this end. The first phase of tests involved implementing the classic Bluetooth stack with the UA767-PBT. The final phase of tests began immediately after completing the overall implementation. Using the UA651-BT, I implemented the BLE stack into the pilot application and conducted the first pilot study.

The first phase of testing helped me to discover a number of issues in the Bluetooth module, and also identify many areas of improvement. The first issue I discovered was a pragmatic one. Many sensors, if not all, do not "speak" Java, or even Android. In fact, when they send information, there is generally no specific IO pattern used. Previously the module had used a construct known as an Object Input Stream to read data incoming from external devices, and the complement, an Object Output Stream, to write data to external devices. This, however, immediately caused a class of exceptions known as StreamCorruptedException. This exception results in the event that the constructed IO stream is not consistent with the IO stream on the other end. In more detail, the read operation of the Object Input Stream attempts to identify a header that java places on objects written to an Object Output Stream. If this header is not found, the StreamCorruptedException is thrown. External devices cannot be guaranteed to

"speak" Java or Android, and thus I needed to use the base class, InputStream, in order to communicate with the devices. This fundamentally changed the inner workings of the converter classes. The converter classes now had to contend with raw data that had no level of preprocessing (i.e., primitive byte streams.) Instead, they now had to convert raw bytes into the appropriate data structure, which mimics more traditional implementation of protocols. Another insight that the test application provided us was the inability of some devices to speak "in general" to a smartphone. Whereas in the Android Bluetooth stack, RFCOMM channels are established on an *as requested* basis, many devices were hardcoded to communicate over one channel. The UA767-PBT is no exception to this rule, and I was forced to circumvent the standard Android API using reflection in order to establish communications with the device. This circumvention is interesting because it lends the question "should the Android Bluetooth stack be more managed?" In its' current state, there is no way to determine how Android handles many connections with different devices, and even less so if those devices use an older standard of Bluetooth. The test application makes a good point that if RFCOMM channels were managed according to incoming requests, the overall complexity of the resulting application would be reduced. As I continue to add more and more external sensors and equipment, it will become clearer if this is truly worth investigating.

The first phase of testing also helped us identify BLE as a more appropriate solution for interacting with health devices. Leading into the final phase of testing and the pilot application, large overhauls of the design of the classic Bluetooth package took place to further mimic the BLE stack's asynchronous behavior. Our pilot study provided us with a number of key insights into the inner workings of the final BLE stack. One such insight was that of consistent functionality. I experienced large problems when testing the BLE stack over a range of devices. These devices were listed as BLE capable with Bluetooth V4.0, but failed to receive any data whatsoever when equipped with the pilot application. Other phones did not have this issue. There can be a number of potential reasons why this inconsistency was seen in the pilot study, but none have been investigated as of yet. Another insight was that of generalization. The Bluetooth Health Device class currently only supports two forms of listeners: heart rate, and blood pressure. In addition, this support is hardcoded into the class. In the future, I will have to add support for additional listeners in a way that is much

more robust and modular. That way, in the event an application developer decides they want to create their own listener, they would be able to specify it and add it simply, using an interface.

Finally, our team utilized a pre and post study survey to examine participant acceptance of the technology, as well as to identify any particular pain points. We saw that among our first pilot study participants, acceptance was generally high, with no participant indicating strong aversion to using the technology. Through our survey we saw that most participants, while comfortable sharing health and location information with their families and physicians, they were unanimously uncomfortable sharing this information with private and insurance companies. This trend persisted to the post study survey. Of particular note, we did not see any change in response from the pre and post study survey. In terms of usability, no participants detailed difficulties they had with the interface or application. On the other hand, participants indicated that they regularly used smartphones, Bluetooth enabled devices, and health tracking software. Based on the survey results, we concluded that the participants were surveyed were generally more technically apt then the average smart phone user, and therefore many of the conclusions we could make regarding the interface and pilot application may not be as applicable, due to sample bias.

In the end, I was able to extract the raw data from the UA767-PBT using the Bluetooth module *as is* during the first testing phase, with no implemented converter or specific device class for the blood pressure monitor. This is significant for two reasons. The first is that the library can function according to specifications with no prior knowledge of what it is connecting to. Although I was forced to make changes accordingly after I had the chance to work with the UA767-PBT, the core structure and functionality of the library remained unchanged. The second is that the module remained independent from other modules during this testing, proving its functionality as a standalone product. This displays the real beauty behind the library, as each component is functionally isolated, and can operate without any of the others. Therefore if another engineer on the project wanted to use the module, they would not need to import any of the test environment or use the library module versions that I used. They could simply extract the module and use it immediately, with no overhead. In the second testing phase, and the pilot study, I was able to successfully extract, store, and backup data from the UA651-BT using the pilot application. I was also able to combine the

functionality of the BLE package with the other packages of the library and use them in tandem without many errors. This is a milestone, as it indicates many of the decisions I made with regard to design are not negatively interfering with other packages in the library. There are still more pilot studies ahead, but the initial results are promising. The next steps would be to characterize more specific data - such as battery life consumption and communication consistency metrics - and begin to quantitatively analyze the performance of the classic Bluetooth and BLE package.

8. Concluding Reflections

At the onset of the project, our team set out to take the tele monitoring library and expand it such that it could be used for a more extensive clinical study. Following the study conducted last year, we believe that there is enormous potential to be had in the development of an open source, robust, and fault tolerant tele medical API. That being said, we did not achieve every goal we sought out to achieve. Particularly in my case, I was hoping to accomplish much more on the lines of testing before exiting the project. While the benefits of BLE versus classic Bluetooth are well known, quantitative metrics, which we proposed to perform early in the project, have yet to be conducted. Arguably, there are still many more studies that will be conducted before the years end. The pilot study was a great achievement, for both our team and for the project. It demonstrated that conducting a study with the platform is not only feasible, but also beneficial. I was able to verify the behavior of the Bluetooth stack, and I was able to identify key issues that detract from the overall experience. These include device inconsistency, ease of use, and modularity of design. These purely technical insights, however, are not alone - there are also key management insights I gained from this capstone experience.

The first major insight I feel I have gained from this project is with respect to understanding the dynamic of a team of software engineers. Software engineering is a very different experience from other forms of engineering. Whereas in a discipline like mechanical engineering, where results are often tangible and failures are often recognizable, many of the problems that can occur in the software engineering design process are much more insidious. Problems can remain uncovered for months at a time, only to surface in the

midst of a study, or, in the worst case, after release. This encouraged me to adopt a more rigorous, thorough approach to reviewing my colleague's collective works. In a way, I am more scrupulous when it comes to reviewing a design, piece of code, or system as a whole. Another major insight I have gained is one of communication. Many times in the project, there would be miscommunications caused by something as simple as a missing "CC" in an email address, or even a discussion held on the side. These seem harmless, but in my case, they have caused a number of situations where a design is just slightly off, because some insight or discussion was not relayed in time. This wastes time, and arguably, reflects poorly on the team as a whole. In the future, I intend to be much more vigilant to improve in the aforementioned areas.

With regards to future research, I believe that there are many improvements that can be made in the Bluetooth package. Much of the standard Bluetooth API in Android relies on fairly synchronous behavior. While much was done in the way of making the classic Bluetooth stack a more asynchronous experience, much of this was done external to the API. I believe that with Android being an open source software, a much more tailored experience can be achieved by building a new Bluetooth stack that better follows the design principles of the library. Additionally, a large area of future research is enabling communication protocols beyond Bluetooth. Certain phones utilize more esoteric forms of communication, such as NFC and IR, which can be used to further expand the breadth of functionality the library offers. Whether or not there are medical devices that utilize these other forms of communication is another research topic in and of itself. The path, however, is fairly clear - the library still has much more room for communication protocols.

To those who would continue this project, I recommend a fairly hearty knowledge of the ways in which Bluetooth and BLE work. It is only when you can see the differences between the two protocols can you develop a much more robust experience using them. Bluetooth and BLE both have their own use cases - and a mix and match between them would be ideal for a tele monitoring system. Additionally, I would encourage those individuals who continue this work to make an effort in making the Bluetooth stack as maintainable as possible. Code, unfortunately, has the tendency to rot. The more maintainable this stack is, the longer the tele monitoring library will remain functional in the years to come. In conclusion, I feel that I

have accomplished much in the way of developing the tele monitoring library, and I look forward to conducting these final rounds of study before graduation.

9. References

Centers for Medicare & Medicaid Services.
"National Medicare Readmission Findings: Recent Data and Trends." *Office of Information Product and Data Analytics* (2012).

Alnowami, M., et al.
A quantitative assessment of using the Kinect for Xbox360 for respiratory surface motion tracking. *SPIE Medical Imaging*. International Society for Optics and Photonics, 2012.

Banet, Matthew J., et al.
Body-worn sensor featuring a low-power processor and multi-sensor array for measuring blood pressure. U.S. Patent Application 13/346,408.

Banet, Matthew J., et al.
Vital sign monitor for cufflessly measuring blood pressure using a pulse transit time corrected for vascular index. U.S. Patent Application 14/072,305.

Bessen, James and Meurer, Michael J.
2008, "Patent Failure: How Judges, Bureaucrats, and Lawyers Put Innovators at Risk, Princeton University Press, http://press.princeton.edu/titles/8634.html, Accessed March 02, 2015.

Dolinay, Viliam, Lucie Pivnickova, and Vladimir Vasek
Objectivization of traditional otoneurological examinations based on Kinect sensor Hautant's test based on Kinect. *Control Conference (ICCC), 2014 15th International Carpathian*. IEEE, 2014.

Ghosh, Rajib; Heit, Juergen; Srinivasan, Soundararajan
Telehealth at scale: The need for interoperability and analytics
International Conference on Information and Knowledge Management, Proceedings, p 63-66, 2011, *CIKM 2011 Glasgow: MIXHS'11 - Proceedings of the 1st International Workshop on Managing Interoperability and Complexity in Health Systems*; Accessed February 16, 2015

Hall, Joseph L., and Deven McGraw.
For telehealth to succeed, privacy and security risks must be identified and addressed. *Health Affairs* 33.2 (2014): 216-221.

Hines, A. L., Barrett, M. L., Jiang, H. J., & Steiner, C. A.
Conditions with the largest number of adult hospital readmissions by payer, 2011 (2014). http://www.hcup-us.ahrq.gov/reports/statbriefs/sb172-Conditions-Readmissions-Payer.pdf,
accessed February 16, 2015

Jones, Pamela
2009, "An Explanation of Computation Theory for Lawyers"
http://www.groklaw.net/pdf2/ComputationalTheoryforLawyers.pdf Accessed March 02, 2015.

Lars, Nile
2014 Connected Medical Devices, Apps: Are They Leading the IoT Revolution — or Vice Versa?
Wired
http://www.wired.com/2014/06/connected-medical-devices-apps-leading-iot-revolution-vice-vers
Accessed Feb 27, 2015

Miller, Edward Alan.
"Solving the disjuncture between research and practice: telehealth trends in the 21st century." Health Policy 82.2 (2007): 133-141.

Morea, Stephan
2014 IBISWorld Industry Report OD5775: Telehealth Services in the US. http://clients1.ibisworld.com/reports/us/industry/default.aspx?entid=5775, Accessed February 16, 2015

Open Source Initiative
Welcome to The Open Source Initiative. http://opensource.org , accessed February 16, 2015

Plotkin, Robert
Intellectual Property and the Process of Invention: Why Software is Different, 2002, International Symposium on Technology and Society, http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1013821, Accessed March 02, 2015

Porter E., Michael
2008 The Five Competitive Forces That Shape Strategy. Harvard Business School Publishing Corporation. https://hbr.org/2008/01/the-five-competitive-forces-that-shape-strategy. Accessed February 16, 2015

Skala, Karolj, et al.
4D thermal imaging system for medical applications. *Periodicum biologorum* 113.4 (2011): 407-416.

Swan, Melanie
2012 Sensor Mania! The Internet of Things, Wearable Computing, Objective Metrics, and the Quantified Self 2.0, Journal of Sensors and Actuators, Page 217-25, http://www.mdpi.com/2224-2708/1/3/217/htm#sthash.e0RU9ljv.dpuf, Accessed Feb 27,2015

Zhang, Zhengyou.
Microsoft kinect sensor and its effect. *MultiMedia, IEEE* 19, no. 2 (2012): 4-10.

Whitten, Pamela S., et al.
"Systematic review of cost effectiveness studies of telemedicine interventions." *Bmj* 324.7351 (2002): 1434-1437.

Falaki, Hossein, et al.
"Diversity in smartphone usage." *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 2010.

Smith, Aaron.
"46% of American adults are smartphone owners." *Pew Internet & American Life Project* (2012).

Aranki, Daniel, et al.
"Continuous, real-time, tele-monitoring of patients with chronic heart-failure: lessons learned from a pilot study." *Proceedings of the 9th International Conference on Body Area Networks*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2014.

"Bluetooth." Android Developers. Google, Inc., n.d. Web. 10 May 2015.

Seto, Emily, et al.
"Mobile phone-based tele monitoring for heart failure management: a randomized controlled trial." *Journal of medical Internet research* 14.1 (2012): e31.

Sultan, Salys, and Permanand Mohan.
"How to interact: Evaluating the interface between mobile healthcare systems and the monitoring of blood sugar and blood pressure." *Mobile and Ubiquitous Systems: Networking & Services, MobiQuitous, 2009. MobiQuitous' 09. 6th Annual International.* IEEE, 2009.

Gund, Anna, et al.
"Design evaluation of a home-based telecare system for chronic heart failure patients." *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*. IEEE, 2008.

Omre, Alf Helge, and Steven Keeping.
"Bluetooth low energy: wireless connectivity for medical monitoring." *Journal of diabetes science and technology* 4.2 (2010): 457-463.

OTT, LEN.
"The Evolution of Bluetooth® in Wireless Medical Devices." *Socket Mobile, Inc. White Papers* (2010).

Gomez, Carles, Joaquim Oller, and Josep Paradells.
"Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology." *Sensors* 12.9 (2012): 11734-11753.

Balani, Rahul.
"Energy consumption analysis for bluetooth, wifi and cellular networks." *Networked & Embedded Systems Laboratory, NESL Technical Report TR-UCLA-NESL-200712-01* (2007).

Dabek, Frank, et al.
"Event-driven programming for robust software."*Proceedings of the 10th workshop on ACM SIGOPS European workshop*. ACM, 2002.

10. Glossary of Terms

API: Application Program Interface
BLE: Bluetooth Low Energy
Bluetooth Smart: Marketing name for BLE
CDMA/LTE/GSM: Mobile network communication designations
Event Generating Class: Any class which can generate events and relay those events through call backs.
RFCOMM: Radio Frequency Communication
IO: Input/Output
UML: Unified Modeling Language
UA767-PBT: Bluetooth enabled blood pressure monitor by A&D Medical
UA651-BT: Bluetooth Low Energy enabled blood pressure monitor by A&D Medical
The service: The Bluetooth service class