A Telemonitoring Solution to Long-Distance Running Coaching

Hannah Sarver
Carlos Asuncion
Uma Balakrishnan
Lucas Serven
Eugene Song
Daniel Aranki, Ed.
Ruzena Bajcsy, Ed.
Ali Javey, Ed.

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2016-85
http://www.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-85.html

May 13, 2016

Acknowledgement

Daniel Aranki, University of California, Berkeley
Professor Ruzena Bajcsy, University of California, Berkeley
Professor Ali Javey, University of California, Berkeley
Dr David Liebovitz, MD, University of Chicago Medicine

BERKELEY TELE-MONITORING

A Telemonitoring Solution to Long-Distance Running Coaching

Master of Engineering Capstone Report 2016

Hannah Sarver

with Carlos Asuncion, Eugene Song, Lucas Serven, and Uma Balakrishnan

Prepared for Professors Ruzena Bajcsy and Ali Javey

Contents

Introduction **2**

1 Individual Contributions **7**

1.1 Initial Application Design Iteration 7

1.1.1 Motivation for Application Design Process 7

1.1.2 Design Mockups 7

1.2 Application Components Overview 13

1.3 GPS Data Extraction 14

1.3.1 GPS Data Collection and Distance Calculation in Marathon Application 17

1.3.2 GPS and Distance Testing Data 18

1.4 Heart Rate Data Extraction and Bluetooth Work 19

1.4.1 Bluetooth Stack Connection Lifecycle Modifications 20

1.4.2 Incorporation Into Marathon Training Application 22

1.4.3 Bluetooth Connection Lifecycle Testing Results 25

1.5 Post Run Results Summary Screen 25

1.6 Conclusion 28

2 Engineering Leadership **29**

2.1 Industry and Market Analysis Overview 29

2.2 Market Analysis 29

2.3 Porter's Five Forces Analysis 30

2.3.1 Bargaining Power of Buyers 30
2.3.2 Bargaining Power of Suppliers 30
2.3.3 Threat of New Entrants 31
2.3.4 Threat of Substitutes 31
2.3.5 Rivalry Amongst Existing Competitors 31
2.4 Technology Strategy 33

List of Figures

1 The Telemonitoring Cycle 4

2 Work breakdown of tasks among team members 5

1.1 (a) Settings Navigation Drawer Mockup (b) User Settings Screen Mockup . . 9

1.2 (a) Home Screen Mockup (b) Run Screen Mockup 10

1.3 Post-Run Cadence and Route Detail Screen Mockups 12

1.4 Non-Visual Intervention Mechanisms 13

1.5 Simplified flow of GPS data from sensor to extractor to be encapsulated and transmitted to the server in a serialized fashion 16

1.6 (a) Google Maps Plot of Collected GPS Datapoints (b) Google Maps Distance Measurement of Start to End Coordinates 18

1.7 Bluetooth Connection Lifecycle 21

1.8 Screenshot of Bluetooth device scanning and selection screen with details of devices found upon BLE scan for nearby signals, and confirmation dialog allowing the user to confirm her selection of a heart rate monitor (HRM) device 23

1.9 Run screen showing data being collected by a connected Bluetooth HRM device 24

1.10 (a) Heart rate data collected across a device disconnect and re-connection. Outlying low datapoints result from the Jarv device beginning data collection. (b) Heart rate data collected across multiple disconnects and re-connections over a few minutes. 26

1.11 Results screen showing summary of a completed run, accessible for the latest finished run via the left navigation drawer menu. 27

Introduction

This capstone project consisted of expanding the existing Berkeley Telemonitoring to improve its overall functionality, which we accomplished by creating an application to suit the needs of a marathon trainee with design input from University of Chicago Medicine's Doctor David Liebovitz. The Telemonitoring cycle, depicted in Figure 1, allows for data from a remote environment, for instance a marathon runner, to be transmitted to a centralized server for processing, possibly viewed and acted upon by a doctor or coach, and responded to by the system's sending intervention back to the remote environment in real-time.

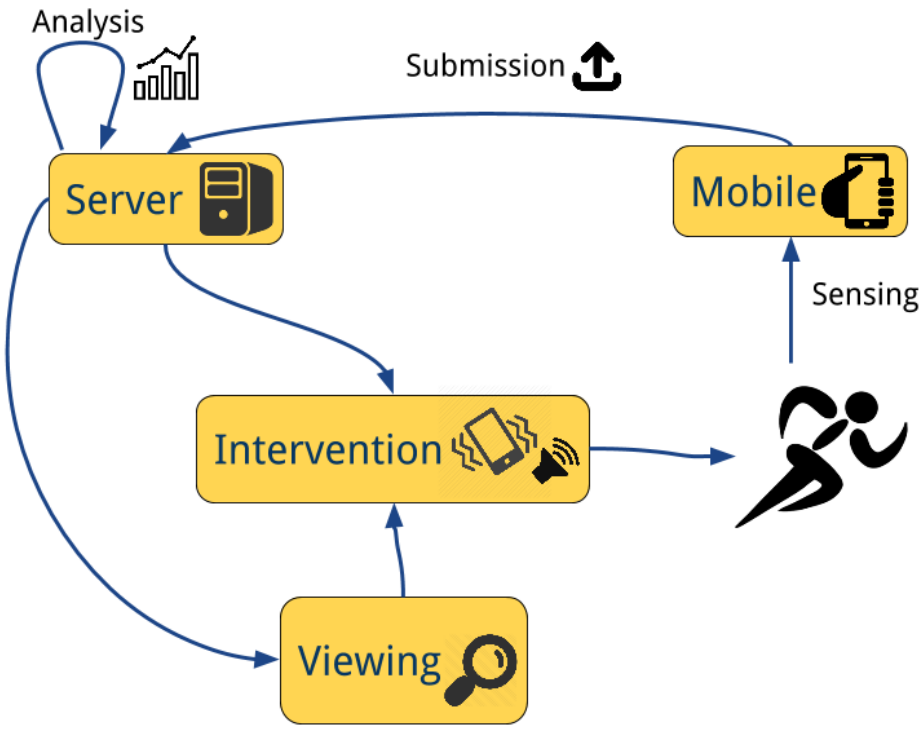


Figure 1: The Telemonitoring Cycle

To the end of validating and strengthening the capacity of the framework to fulfill this full Telemonitoring cycle, the ultimate deliverable of our project was a fully functional and well-tested mobile application that tracks a trainee's cadence and other parameters throughout a training run and provides real-time audio feedback to tell the runner her proximity to a specified target cadence. In order to achieve this goal our team worked to extend various aspects of the existing framework (Aranki et al. 2016) including adding support for data analysis at scale, generating capability to extract data from the phone's on-board GPS and accelerometer sensors, improving the robustness of the Bluetooth communications stack, and enabling non-visual interventions to be communicated to the application user. Our overall task breakdown with work distribution is displayed in Figure 2.

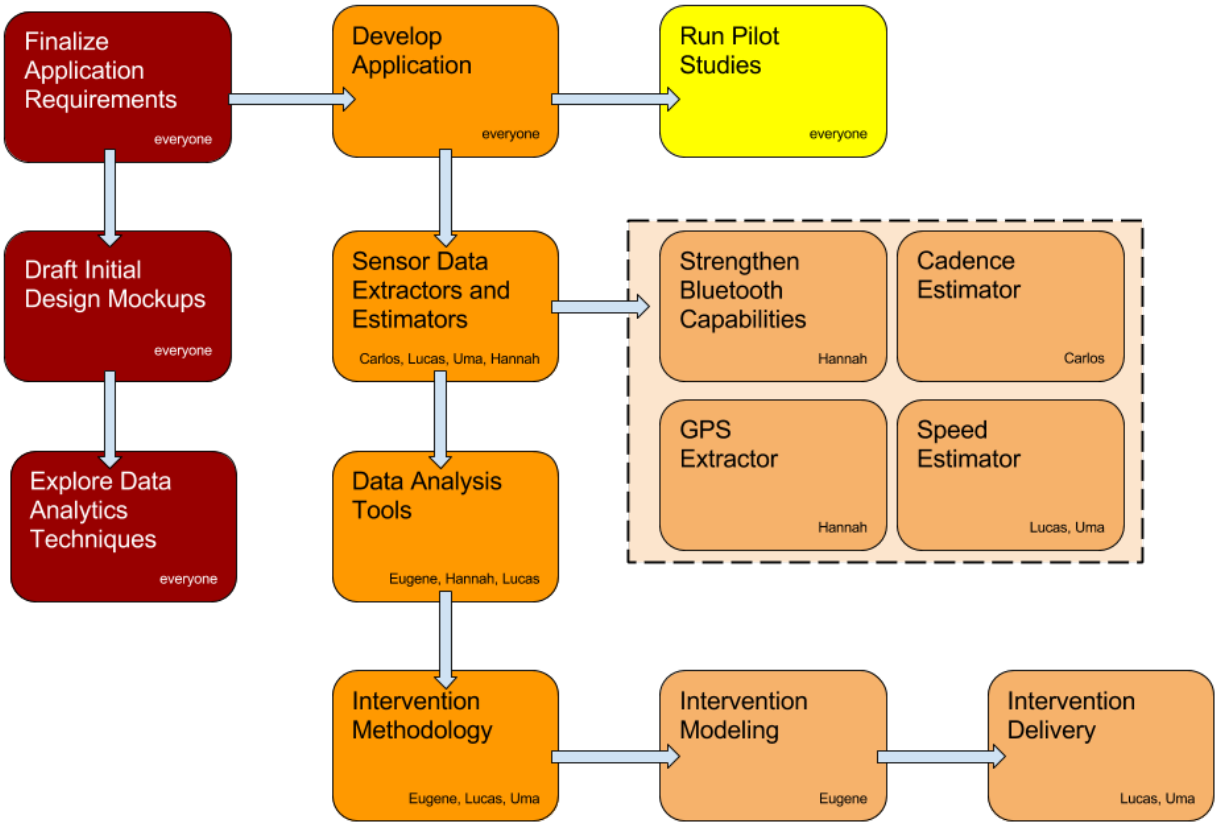


Figure 2: Work breakdown of tasks among team members

Many tasks were shared between all team members, as the design aspect of the project and general understanding of the required data analysis techniques were important for everyone

involved. My primary individual contributions to the project consisted of working on implementation of data analytics methods for our framework along with teammates Lucas Serven and Eugene Song (described in Eugene's paper), incorporation of GPS data extraction into the framework, and refactoring the lifecycle of connections within the Bluetooth communications stack in order to support robust data collection from a peripheral heart rate monitor. In this paper, I will describe the team's initial strategy for developing a design of the marathon application itself and detail the components to which I contributed most significantly as an individual, including GPS data extraction and Bluetooth stack improvements.

Chapter 1

Individual Contributions

1.1 Initial Application Design Iteration

1.1.1 Motivation for Application Design Process

The design of the functionality and interface of our target marathon training application was crucial to planning out the implementation and testing work that constituted our project. As this application development is central to our goal of expanding and testing the Telemonitoring framework, we determined that the most benefit to the framework would be derived from thinking through the features that users would desire in this application and then during the implementation phase of the project ensuring that the framework could support them sufficiently. The initial application design process was completed as a team, and delegated to me for the purposes of this written deliverable.

1.1.2 Design Mockups

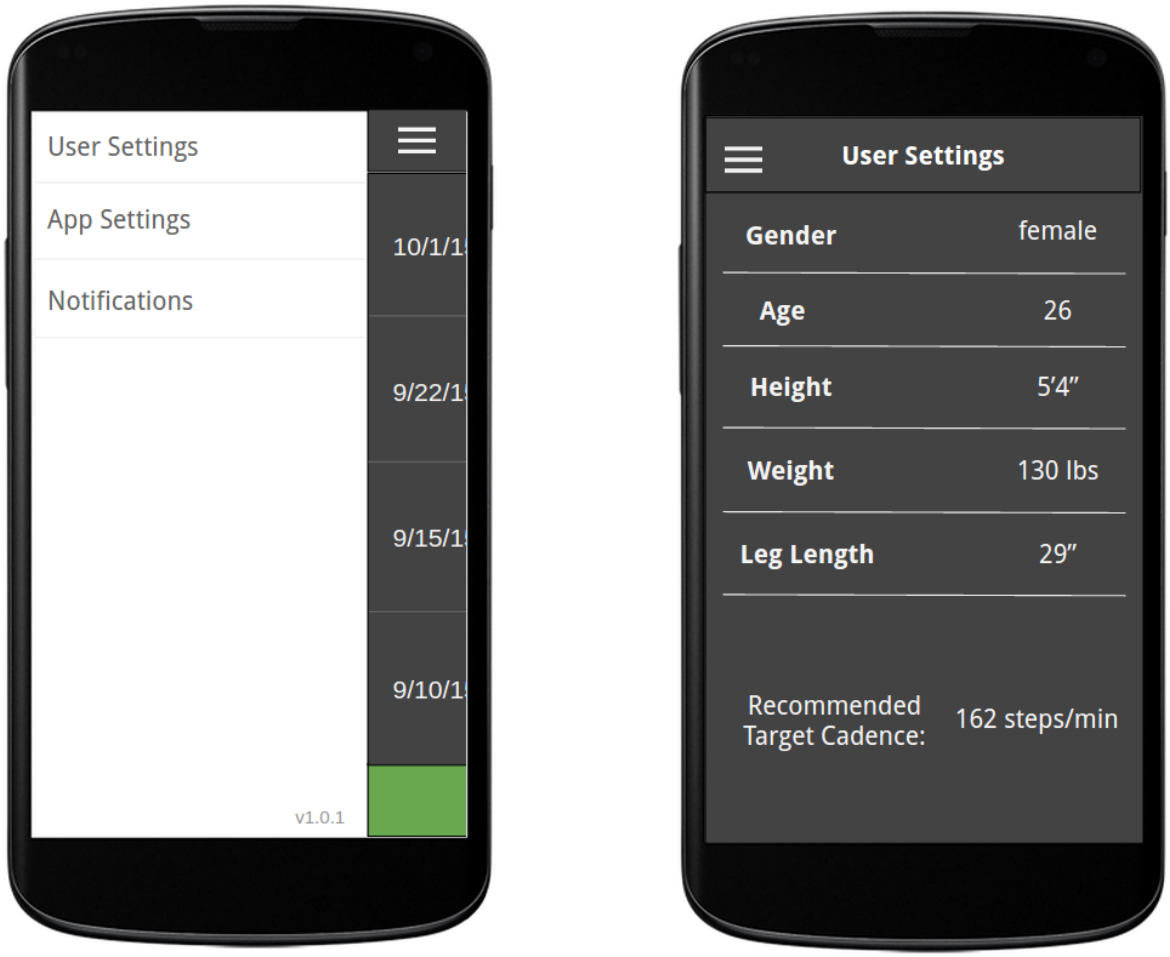
Given a general description of desired functionality from Dr. Liebovitz, we drafted visual mockups of the application we would be creating. Starting with simple, fungible visual prototypes allowed us to easily follow an iterative design process, soliciting feedback from our stakeholders and potential users and altering our design without spending unnecessary time and effort implementing full functionality for early design versions. This type of iterative

design process is well-studied and widely accepted in academia and industry. According to researcher Jakob Nielsen the use of iterative refinement of interfaces yields significant usability improvement; in his study of four cases of iterative design change with user feedback the median usability improvement from first to last version was 165% (Nielsen 1993).

Our mockups depicted the various screens that the user would see while using our application, and thus helped inform the functions that we needed to implement in order to collect and convey appropriate information to the user. Our next design iteration consisted of moving from these visual prototypes to the actual code implementation of these screens in the functional prototype application, including design changes necessary to support additional functionality we determined would be useful to the runner. These implemented screens are distributed through the reports of my teammates for detailed discussion; this report includes the newly added screen on which users scan for Bluetooth devices and choose one to collect data from, detailed later in the Bluetooth Connection Lifecycle section. I'll now show a full user interaction flow through the screens that we designed in our first iteration and describe in detail what value each screen gives to the user and what it dictated with regard to our implementation work. For brevity, I will refer to the runner using our application as "she" although the application is not targeted at any gender in particular.

User Settings Input

During initial setup of the application, the user enters some detailed physical information into a settings screen (Figure 1.1b) in order for our data-informed server to estimate an appropriate target cadence for her to work towards. We will ultimately use the data analytics algorithms described in Eugene's report to determine what is a realistic good target based on data from other users, so in our design mockups a generated target is shown depicting this ideal future feature although in our functional application for pilot testing we instead allow the user to choose a target cadence value and improvement trajectory to reach this goal. If the runner wishes to later change any of her entered parameters or training trajectory she can reach the settings screen through a side navigation drawer (Figure 1.1a).



(a) (b)

Figure 1.1: (a) Settings Navigation Drawer Mockup (b) User Settings Screen Mockup

Home Screen

From the main home screen (Figure 1.2a) the user can see her history of past runs and select one to view in more detail. She can also press the Start button to begin tracking data for a new training run, or access the application settings in the upper left corner to specify physical parameters and other setup information.

Run Screen

Upon pressing the Start button from the home screen, the runner will be taken through two options to collect a starting heart rate estimate using the camera sensors on the phone. After their completion, she will arrive at the run screen (Figure 1.2b) where she sees her current

cadence and proximity to her target, as well as total time and distance of the current run and, if applicable, her current heart rate as monitored by a peripheral device. There is a clear indication of whether, given the training regime she is following (additional detail on training models presented in Eugene's paper), she is sufficiently close to her target cadence for the given day. The cadence section of the screen changes color to indicate whether she should be going faster or slower to match this target. From this screen she can also pause or end her run.

(a) (b)

Figure 1.2: (a) Home Screen Mockup (b) Run Screen Mockup

Cadence Detail Screen

If the user selects a particular completed run from the home screen, a detailed view of cadence information from that run will appear (Figure 1.3), showing measured cadence over time with minimum, maximum, and average cadence clearly indicated. These data will be useful to the runner in terms of tracking performance over time and planning future training runs.

Route Detail Screen

A neighboring tab on the cadence detail screen (shown second in Figure 1.3) shows information about the route taken for that particular run based on collected GPS data and the runner's total distance covered during the run, as derived from the GPS data points.

Due to time and resource constraints during implementation of the functional application, we determined that while conveying basic information on the average and target cadence of each run is important to the core purpose of the application, the graphical depiction of these data was of a lower priority. Thus for our pilot study application implementation we decided to combine these cadence and route detail screens from our initial mockups into a unified run summary screen showing pertinent information more concisely.

Non-Visual Intervention

From our first mockup design iteration we received feedback from our project advisor, Daniel Aranki, to include other sources of intervention feedback rather than only a visual display of the measured cadence data and proximity to the target cadence, as this is not an effective way to communicate information to a runner during a run. This problem identified with our design encouraged us to focus more on the end user experience of our application and update the interface to better suit the needs of our users. In order to provide a useful intervention while the user is currently running, we identified a few non-visual feedback options: have the phone vibrate in a specified pattern to indicate the runner should run faster or slower to match her target, have the application give explicit audio prompts telling the user what to

Figure 1.3: Post-Run Cadence and Route Detail Screen Mockups

do, or integrate with the phone's music player to increase or decrease the beats per minute (BPM) of the music playing to indicate the appropriate change in the runner's behavior.

For the initial application implementation, we decided to include only the vibration cues as the non-visual communication mechanism. We chose this in order to keep the application simple and minimally invasive to the runner's existing routine, as well as to ensure that our technical work would be achievable and robust. The spoken audio and music BPM feedback methods are left as a future improvement to the application, and are discussed further in Uma's paper.

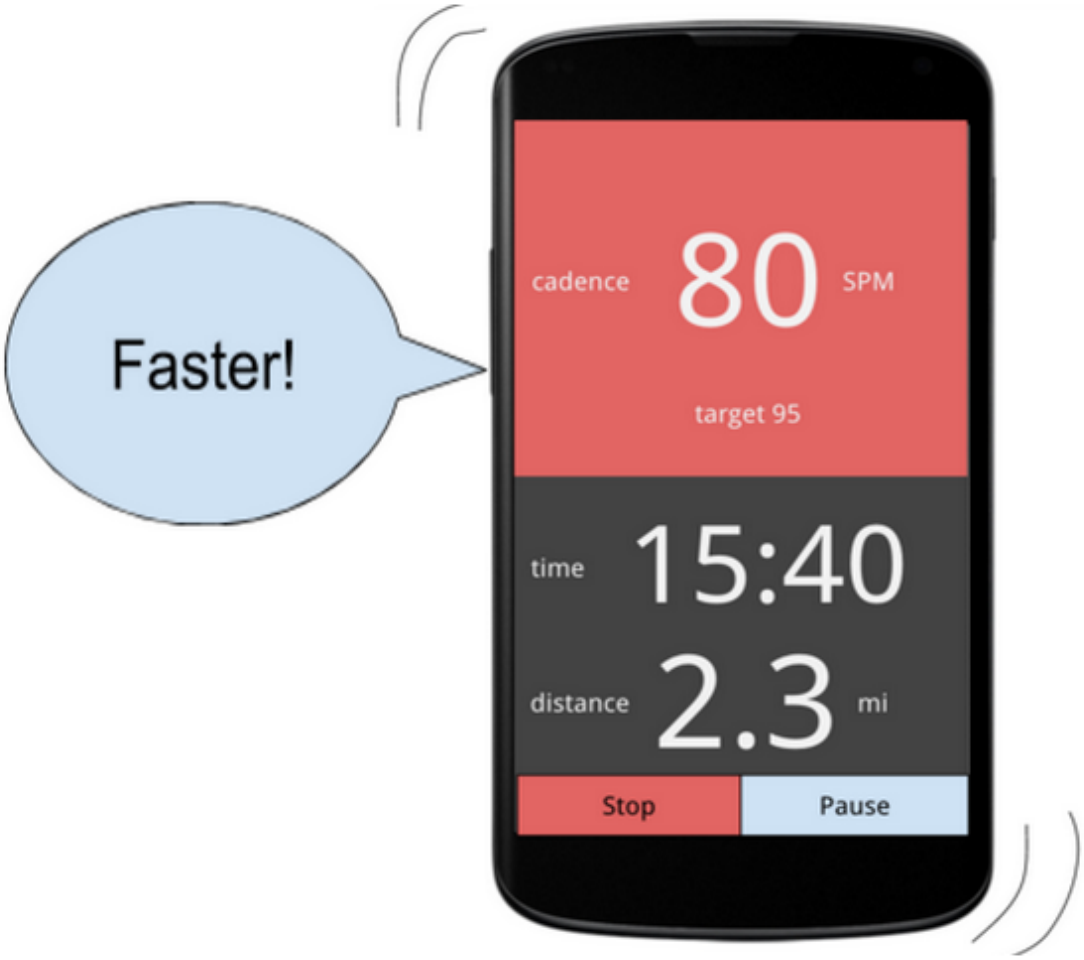


Figure 1.4: Non-Visual Intervention Mechanisms

Having an overall design for our application helped us to plan out and track our implementation progress towards our final functional project deliverable, and provided a framework for us to maintain focus on usability as we developed and tested the features of our application.

1.2 Application Components Overview

Starting from these initial design mockups, the team began implementation of the framework additions and application-specific code components necessary to provide the required functionality. The primary elements involved are the data extractors and estimators, the performance training model, the data analytics support, and the user interface including

data display and non-visual intervention techniques. The estimators for cadence and speed along with the mathematical basis and communication to the user thereof are discussed in Carlos's, Uma's, and Lucas's papers. The training model and data analytics built into the framework which will support assignment of a training target to the user based on entered parameters as well as support for some mathematical operations needed for speed and cadence estimation are discussed in Eugene's paper. I will describe in detail the extraction and display of GPS data and of heart rate information via Bluetooth in the following sections.

1.3 GPS Data Extraction

Information from the Global Positioning System (GPS) is one of a few types of data that are provided by sensors on the phone but not supported by the existing Telemonitoring framework prior to this project. The GPS is a spaceborne positioning system "conceived as a ranging system from known positions of satellites to unknown positions on land, sea, in air and space" (Hofmann-Wellenhof, Lichtenegger, and Collins 1997) through which devices capable of receiving signals from GPS satellites can determine their own locations and track this information over time. Collecting coarse location data provided by the phone's GPS receiver, often augmented by locationing information based on proximity to cell towers and wireless network signals, allows us to calculate an approximate distance covered during each run as described following. Additionally, one of the methods for estimating speed that we researched [1] uses GPS data to calibrate the model for estimation (Altini et al. 2014), so if in the future it is determined that the implemented speed estimation algorithm is insufficient we will have the flexibility to change to that method or another that requires GPS data integration. Finally, for future applications outside the marathon training space that will be implemented using the Telemonitoring framework, GPS data may be critically useful, and having support already built in will decrease development time for these applications.

The process of adding support for GPS data extraction involved understanding the frame-

[1] In the method discussed in Self-Calibration of Walking Speed Estimations Using Smartphone Sensors Estimations by Altini et al. 2014, GPS data is used in conjunction with accelerometer data to personalize the speed estimations generated by their model.

work's data extraction and storage models and writing new code to support these actions specifically for GPS data, as well as testing the functionality of this new code by incorporating it into the existing sample application before we had begun the implementation phase of our marathon coaching application. In order to write the new software classes necessary for the application to stream GPS location data to the server, I needed to research and understand how Android applications access the data from the onboard GPS sensor. The infrastructure for developers using the Android operating system makes it very easy to obtain a user's location within a software application (Shu, Du, and Chen 2009). The Android libraries provide support for reactionary event-based data exchange, causing an action to happen only when the phone changes location, and for location data retrieval on demand, based on the latest known coordinates if available.

One challenge involved in this effort was determining how best to fit these access patterns into the Telemonitoring framework's standardized process surrounding sensor data extraction, which is mainly designed around timed periodic reading of sensors. The general structure that is followed in data extraction within the framework is for each type of data to have its own Encapsulator class which can store a datapoint of that type, and then an Extractor or Estimator class which collects or estimates the data of the specified type to generate standardized data points to be stored in instantiated encapsulators and sent to the server for remote processing (see Figure 1.5 for clarification of the dataflow). On consultation with the project advisor, I included code to support extraction of GPS coordinates in both periodic and event-based manners, re-using the storage container code for the data since datapoints can be stored in the same format in both cases. Although we will likely only use one of these extraction methods in our application development this year, this initial effort will leave the framework better able to support flexibility in future applications. One reason that maintaining a minimum time between collecting datapoints, either through using the periodic extraction method or setting a minimum span between events handled, is appealing for this application is that use of the GPS sensor can dramatically increase the rate of power usage on a cellular phone (Carroll and Heiser 2010), and given that we expect runners to

be continuously using our application for durations of an hour or more we do not want to unnecessarily drain their phones' batteries due to too-frequent location tracking.

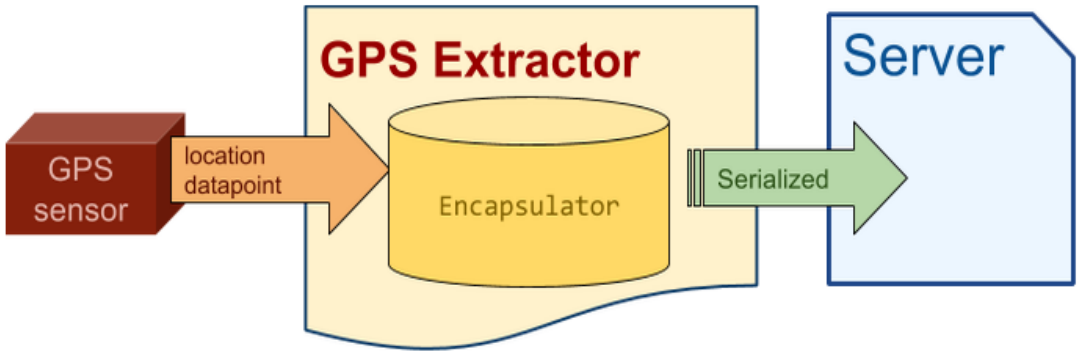


Figure 1.5: Simplified flow of GPS data from sensor to extractor to be encapsulated and transmitted to the server in a serialized fashion

Once the encapsulator class for storing GPS location data and the extractor class for retrieving these data from the phone's sensor were written, they needed to be tested in the context of a running Android application built on the Telemonitoring framework. For this initial testing, I added to the existing test application in our shared codebase. This effort involved adding GPS monitoring to the functionality of the application, and preference control to the user interface to allow the user to turn on and off GPS monitoring. To run the test application and verify that data were streaming properly, it was necessary to familiarize myself with the entire software pipeline of our framework, including compiling the updated server code, copying it to our remote server, and starting it running there, as well as launching the client application on one of our lab-owned Android devices. Then with debug logging I could verify that the data were being collected on the client side, and checking the logs on the server I could ensure that it was correctly being synchronized for remote access.

Another major challenge to effectively extracting GPS location data is the extent of the phone's ability to contact GPS satellites and maintain clear signal reception. While testing I found that GPS was frequently inaccessible when indoors or when the phone had just come out of sleep or powered-off modes. This limitation required a slight re-design of the extractor structure, as the code for other sensor extractors was written with the expectation

of definitely retrieving a datapoint at each periodic cycle of checking for data. When the GPS sensor cannot access the satellites' signal and additionally does not have a stored most recent location it returns a null value, which cannot be stored in the expected latitude and longitude format and provides no useful information to an application. So this case needed to be handled without causing an error, which I accomplished by ignoring null values and sending no data to the server in these cases. This maintains functionality of the application, but means that any algorithm that we implement in the future that relies on GPS data will have to be designed to be robust to receiving no data on startup, and potentially less often than the specified period should the sensor stop being able to retrieve data.

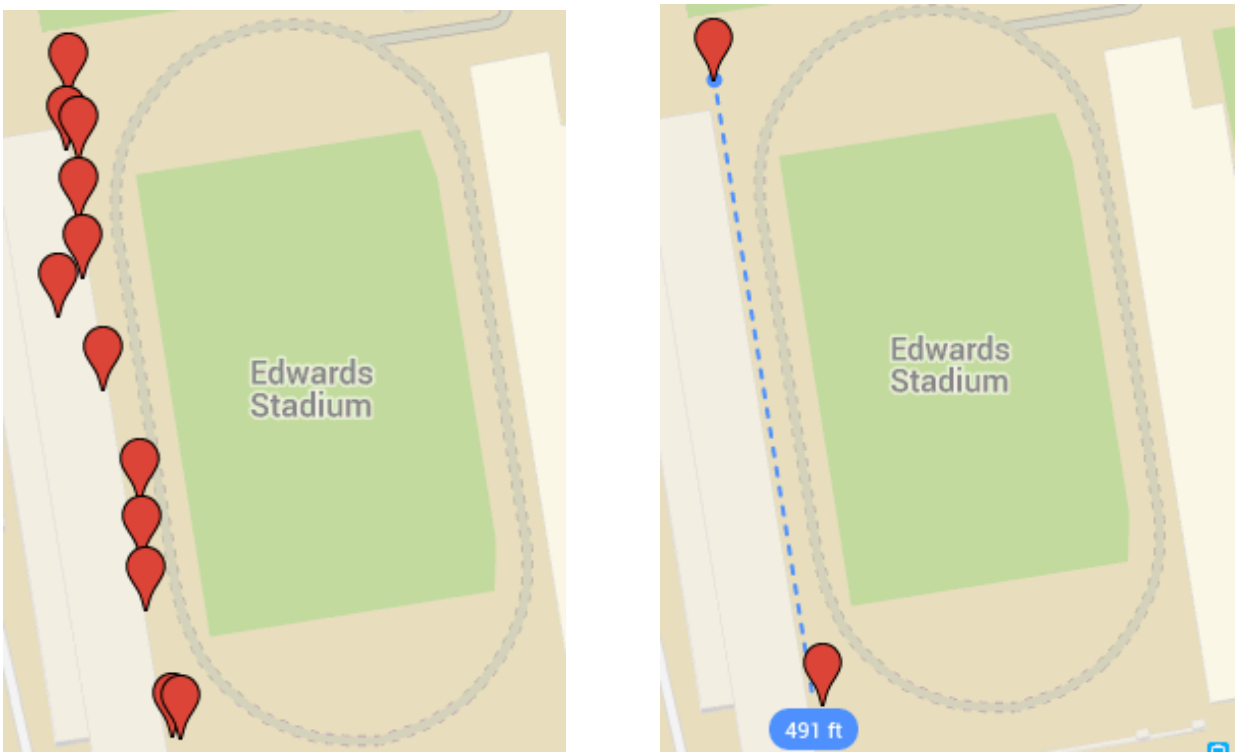
1.3.1 GPS Data Collection and Distance Calculation in Marathon Application

Once GPS data extraction was implemented and tested on its own, the task remained to incorporate this functionality into the marathon training application. While we did want to send GPS coordinates collected during each run to the server, we recognized that maintaining easily visible location information on the phones of our pilot study participants posed a risk to their privacy should their phones be lost or stolen. Thus we made the design decision to collect GPS datapoints to send to the server but to store minimal accessible GPS data on the phone and within the user interface to show only a calculated distance for the run based on change between collected GPS datapoints. For the distance calculation, I implemented the haversine formula (Equation 1.1) commonly used in navigation to determine the great-circle distance between two points given their latitudes and longitudes and the radius r of the Earth (Mwemezi and Huang 2011). Although for this running training application it is unlikely that taking the Earth's curvature into account would have significance, this code could potentially be reused in future applications built on the framework that involve GPS coordinates significantly further apart.

$$distance = 2r \cdot arcsin(\sqrt{sin^2(\frac{lat_2 - lat_1}{2}) + cos(lat_1)cos(lat_2)sin^2(\frac{lng_2 - lng_1}{2})}) \quad (1.1)$$

1.3.2 GPS and Distance Testing Data

We tested the location collection functionality within the marathon training application by plotting the collected GPS datapoints from a brief test run using Google Maps, shown in Figure 1.6a. Datapoints were logged upon location-change events up to every 10 seconds as configured in the application code, and showed up on the map quite close to our expectations. We additionally checked our distance calculation approximately using Google Maps' visual distance measurement tool, as shown in Figure 1.6b. This measurement indicated a total run distance of 491 feet as compared to our application's calculation of a cumulative distance of 517 feet. A likely reason for the small discrepancy between these distances is that our distance calculation is accumulated by change between each subsequent pair of GPS coordinates collected, so variation away from a straight line between the starting and ending points of the total run will increase the calculated distance above the simple start-to-end value.



(a) (b)

Figure 1.6: (a) Google Maps Plot of Collected GPS Datapoints (b) Google Maps Distance Measurement of Start to End Coordinates

1.4 Heart Rate Data Extraction and Bluetooth Work

Along with cadence, speed, and location, we wanted to provide functionality for the runner using our application to be able to easily record her heart rate throughout each of her training runs. Although the existing Telemonitoring framework already included support for two algorithms to detect and record a heart rate reading using the camera sensors on the phone[2], these methods require the user to be actively engaged in the activity of recording these data and remain relatively still during the process and thus are not appropriate for tracking heart rate continuously while the user is running (van Gaalen et al. 2015; Peng et al. 2015). Given this limitation we decided to use a peripheral hardware monitor which can transmit sensed heart rate data to the runner's phone during her run via the Bluetooth Low Energy protocol. After researching several external hardware heart monitors, we chose the Jarv sensor as a relatively inexpensive option that implements the standard Bluetooth Heart Rate Health Device profile (Bluetooth SIG 2016). This means that the data transmitted by the sensor follow a known format that is recognized by other systems that conform to the Bluetooth standard, including having some basic support within the Android operating system (Mosa, Yoo, and Sheets 2012) upon which our framework is built. This native Android support in conjunction with the abstractions of the Bluetooth stack and health device connection process added to the Telemonitoring framework last year made it relatively straightforward to incorporate the Jarv into our marathon training application.

Although last year's capstone team performed significant work to provide support in the Telemonitoring framework for abstracting away from the developer the inner working of the Bluetooth and Bluetooth Low Energy (BLE) communications in order to allow easy handling of devices implementing a Bluetooth Health Device Protocol, there remained some effort necessary to incorporate the Jarv heart monitor device into our marathon training application. As mentioned in last year's team's reports, the system had only been tested with a few device types and needed additional testing with new device types (Mani et al.

[2]These methods involve estimating a person's heart rate based on coloration changes due to bloodflow which are detected in sequential video frames of her face or finger. The implementation of these algorithms in the Telemonitoring framework is detailed at length in the papers of last year's team (van Gaalen et al. 2015; Peng et al. 2015). Their usage in our application before and after a training run is discussed in detail in Uma's paper.

2015; Azar et al. 2015). As the framework's Bluetooth support had not previously been tested with a device using the Bluetooth Heart Rate Health Device profile, I first had to make some minor code changes in order for this type of device to be recognized by the system. The bulk of the work in effectively including the Jarv device in our application lay in improving upon last year's integration of Bluetooth functionality into the framework to offer configurable flexibility in the lifecycle of applications' connections to Bluetooth and BLE devices. Finally, it was necessary to include the required scanning, connection, and data collection code into our application in order to record heart rate datapoints from the Jarv device during each training run if that preference is specified by the runner.

1.4.1 Bluetooth Stack Connection Lifecycle Modifications

A significant component of the effort to include BLE heart rate sensing in our application was a redesign and corresponding implementation of the Bluetooth connection lifecycle in the framework. Previously, the behavior of the system in terms of retrying a new connection to a Bluetooth device as well as its behavior in reconnection attempts upon a lost connection was non-apparent and non-configurable to the developer using the framework to build an application with a Bluetooth device connection component. As suggested in the continuing work sections of last year's team's papers, it is central in development of this framework's Bluetooth communications stack to ensure maintainable code that addresses both Bluetooth and BLE devices (Azar et al. 2015; Mani et al. 2015). In order to make this development experience more intuitive and flexible I worked with advisor Daniel Aranki to generate a clear design of the possible desired behaviors within the context of Bluetooth connection lifecycle and a developer configuration process including a set of reasonable default options to maintain as much abstraction as possible for the developer using our framework. The diagram in Figure 1.7 depicts the possible paths for a Bluetooth device from connection attempt to connection close, with optional paths aligning with configuration settings that include connection and reconnection retry attempts.

In order to allow developers to specify whether to try multiple connection or reconnec-

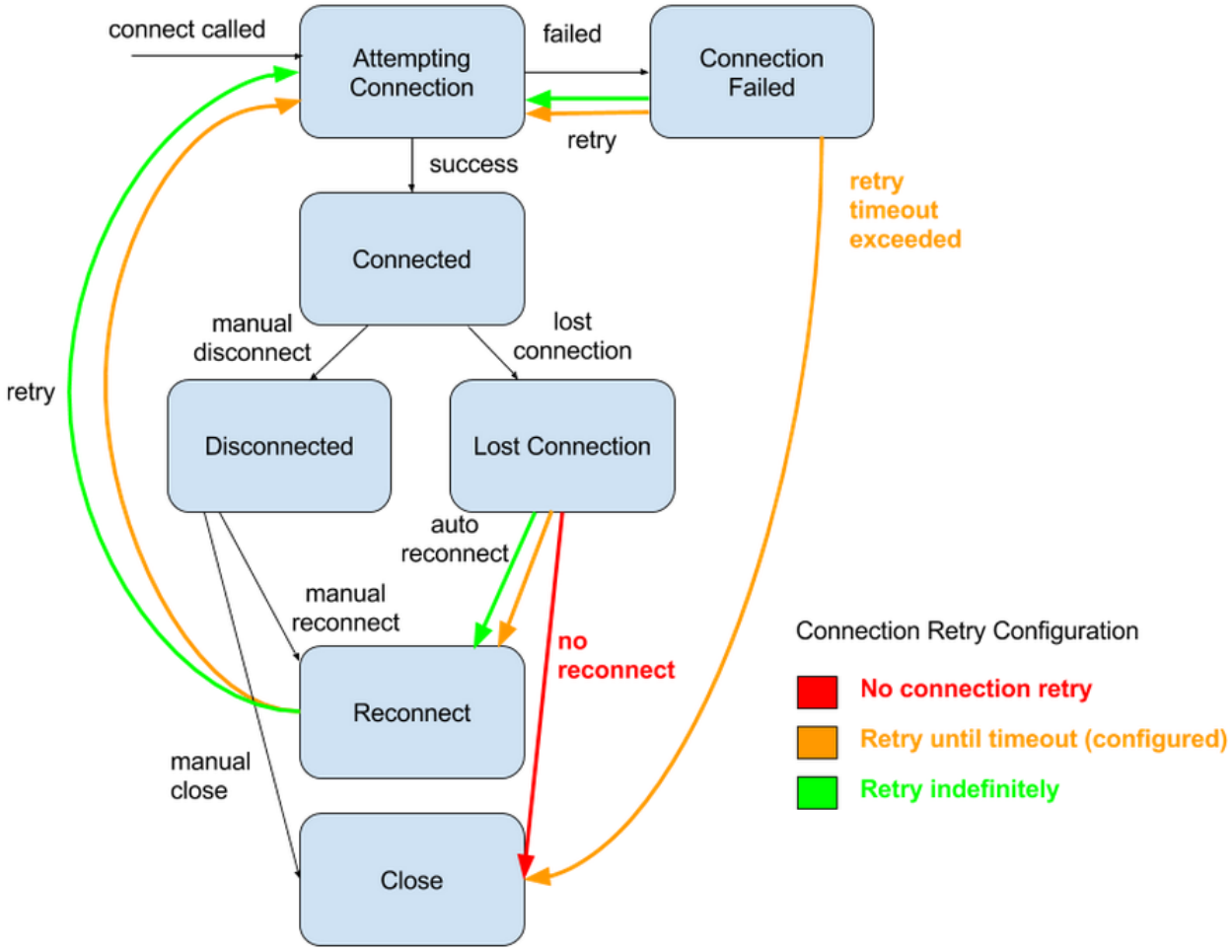


Figure 1.7: Bluetooth Connection Lifecycle

tion attempts to Bluetooth devices in their applications, we designed a class to represent Bluetooth Connection Configuration, which can be instantiated and passed to a Bluetooth device object in order to dictate connection and reconnection behavior. In this new connection lifecycle paradigm, by default a given Bluetooth device will attempt a direct connection when its `connect` method is called, and if this attempt fails the connection will close without making another attempt at success. If the connection is successful but is then later lost unexpectedly, the connection will close without making any attempt at reconnection. If a developer wishes to override this default behavior, she may simply create a Bluetooth Connection Configuration object with parameters set for either or both of initial connection and reconnection on lost signal and call the `applyConnectionConfigurationSettings` method to apply these settings to the device before attempting connection. She can specify

for either of these whether to not retry at all and immediately close the connection, retry up to a specified time interval before closing, also configurable, or to retry indefinitely until success or an explicit call to `close`.

Implementation of this configuration mechanism and the underlying modifications of the Bluetooth stack within the framework to support retry logic on initial and lost connections required a deep dive into the existing Bluetooth stack in the Telemonitoring framework and a testing effort to verify that the additional functionality was working correctly. Future work to be done includes testing of the new connection configuration behavior with other devices beyond the BLE Jarv heart rate monitor, and potential expansion of this connection retry logic into a more abstract approach that could be applied to provide standardized connection configuration not only for Bluetooth devices but for connections of other types such as WiFi or Zigbee.

1.4.2 Incorporation Into Marathon Training Application

Scanning and Device Connection Screen

Within the user interaction flow of the marathon training application, if the runner would like to specify a Bluetooth or BLE heart rate monitor device to collect heart rate data during runs the first step is to enter the Bluetooth connection screen (Figure 1.8) in order to find the device in a scan of nearby transmitting Bluetooth devices and select it to indicate that the application should store the specified device identifier to enable automatic connection and data collection upon starting any training run in the future.

Run Screen with Bluetooth Connection Indication

After initially selecting a heart rate monitor device, when the user later starts a new run and is wearing the device its broadcast heart rate datapoints will immediately begin being recorded and sent to the server. In addition, should the runner have the phone screen on she will see an indication appear on the run screen displaying her current heart rate, as shown in Figure 1.9.

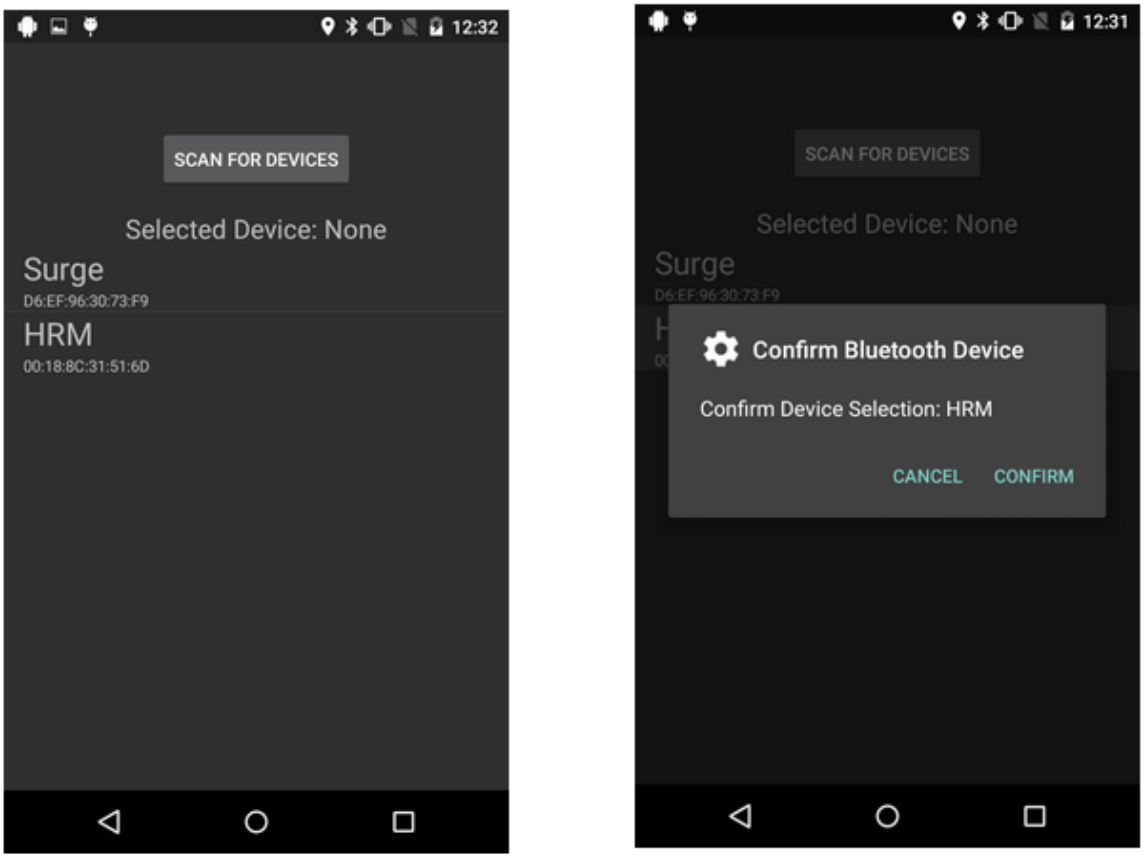


Figure 1.8: Screenshot of Bluetooth device scanning and selection screen with details of devices found upon BLE scan for nearby signals, and confirmation dialog allowing the user to confirm her selection of a heart rate monitor (HRM) device

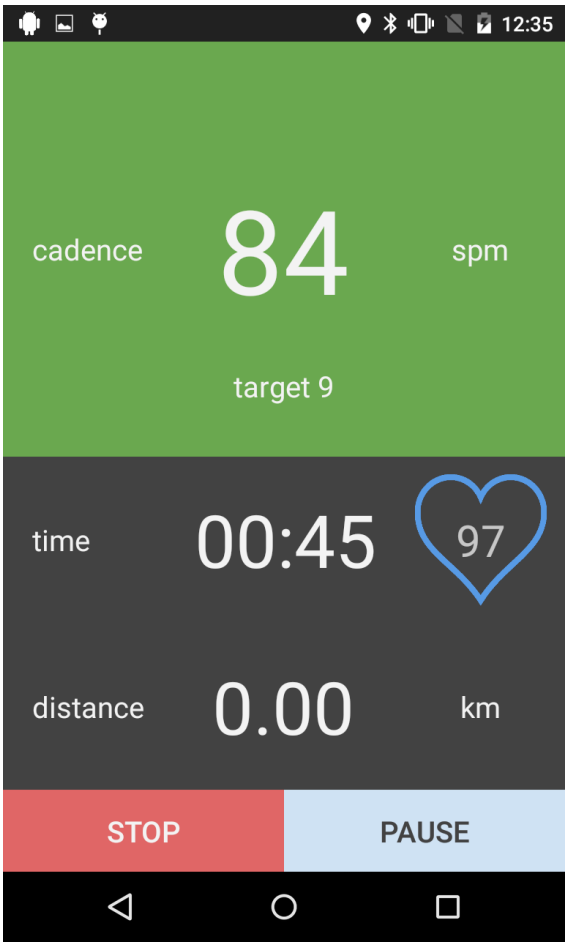


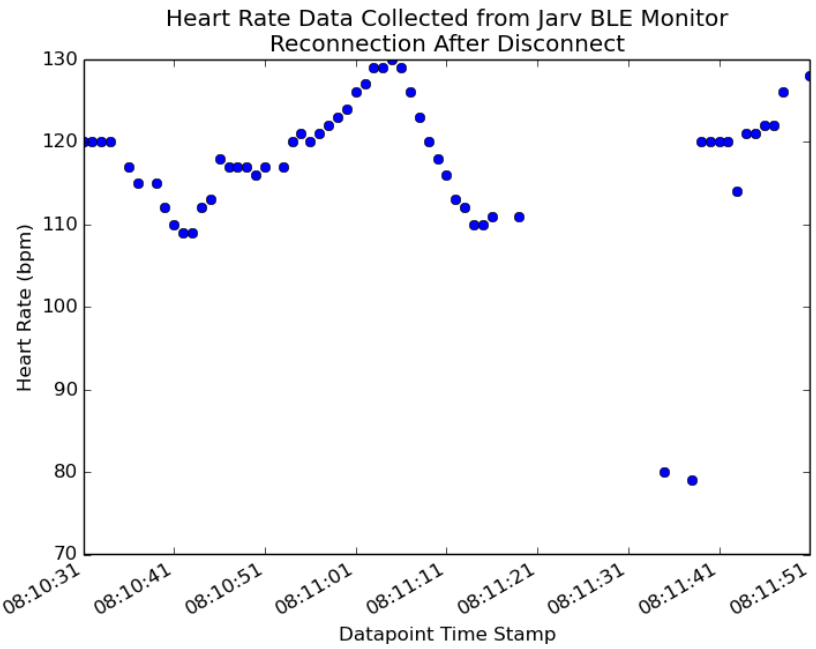
Figure 1.9: Run screen showing data being collected by a connected Bluetooth HRM device

1.4.3 Bluetooth Connection Lifecycle Testing Results

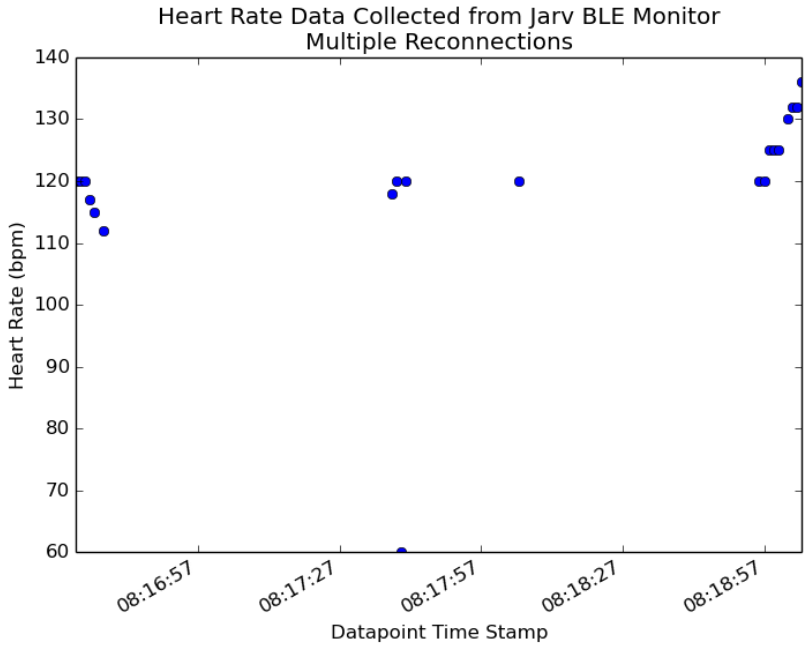
We were able to test some of the modified Bluetooth connection and reconnection behavior in the field in the context of the marathon training application, using the Jarv BLE heart rate monitor device. With connection to the Jarv device configured to attempt connection and reconnection indefinitely, we saw successful reconnection from the application to the device after a few seconds to a minute of being disconnected, as shown in Figure 1.10. In controlled manual tests, we were also able to confirm expected behavior of the retry timeout and no retry logic configurations, as well as observing reconnection attempts in the range of twenty to thirty minutes of being disconnected with indefinite retries configured.

1.5 Post Run Results Summary Screen

In addition to the GPS and heart rate data extraction, I contributed to some of the other less-complex application elements including the screen to display the results of a run to the application user in summary. This screen is shown immediately after the conclusion of a run and can be accessed for the latest run from the left navigation bar as shown in Figure 1.11, or when a previously-completed run is selected from the home screen. This screen shows the target and average value for this particular run of each of the main types of data collected during runs, so that the runner can easily tell how close she was to her goals for that run.



(a)



(b)

Figure 1.10: (a) Heart rate data collected across a device disconnect and re-connection. Outlying low datapoints result from the Jarv device beginning data collection. (b) Heart rate data collected across multiple disconnects and re-connections over a few minutes.

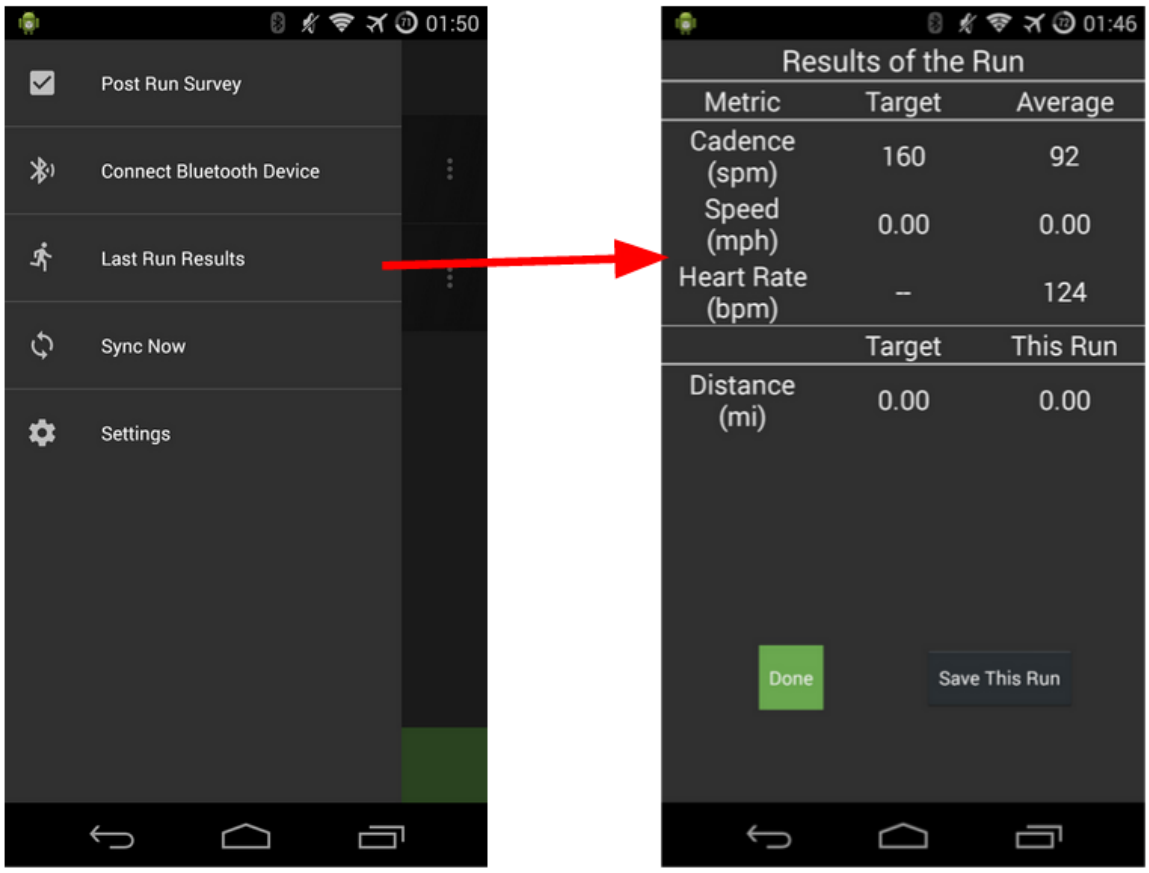


Figure 1.11: Results screen showing summary of a completed run, accessible for the latest finished run via the left navigation drawer menu.

1.6 Conclusion

Although the project deliverable for our team is a standalone application for marathon training, our more significant engineering contributions are in the form of expansion of functionality of the Berkeley Telemonitoring framework presented by Aranki et al. 2016. Thus for those continuing to build from our work and that of our predecessors on the framework, there is great opportunity both in continuing to expand this framework, which is open-source and available online (The Berkeley Telemonitoring Project 2016), to support additional uses and in creating applications using the framework to address real health needs of patients that can benefit from telemonitoring of their conditions. By testing existing functionality of the framework through creation of our new application and adding support for new sensors, data analytics techniques, and feedback intervention models, we have increased the potential for this framework to serve medical professionals and other health-oriented application developers in the future, a need motivated in part by a previous study on real-time monitoring of patients with chronic heart failure (Aranki et al. 2014). In addition, we created a pilot study plan that has been approved by Berkeley's Institutional Review Board (IRB) to complete a larger-scale test of our application and its efficacy, which will allow significant further testing and development of the functionality we have added to the telemonitoring framework.

Chapter 2

Engineering Leadership

2.1 Industry and Market Analysis Overview

While there exist several smartphone based platforms that can be used to create applications for various purposes, such as Canvas, Appery.io and Mobile Rodie (Smith 2013) most of these offer limited functionality and access to sensors. Additionally, these products lack the ability to easily build predictive models for automated generation of personalized interventions. More generally, there are no such platforms that cater to the issue of telehealth. Our telemonitoring framework, targeted towards doctors and coaches, addresses this unmet need.

To guide the expansion of the framework, we will consider the design and implementation of new features in the context of a commercial application that would be used by marathon trainees. To this end, it is useful to perform a market and industry analysis on existing fitness tracking technology. By examining consumer behavior and industry offerings, we can better understand what functionality is missing and what features athletes desire.

2.2 Market Analysis

According to surveys (Running USA 2015) in 2014 there were more than 1200 marathon events held within the US, with a total of 550,637 finishers. These are both all-time high

statistics, with the number of marathon finishers growing about 1.8% from 2013 to 2014. A survey of marathon runners showed that 74% of them relied on wearable technology for training and 88% of them relied on said technology for motivation (Freescale 2014). Between 2014 and 2015, the number of wearables purchased is said to have nearly tripled from 17.6 million to 51.2 million (GfK 2015). Of Internet users who exercise between the ages of 18-34, 38% of males and 21% of females use wearable fitness trackers (Mintel 2014). Wearable technology has clearly entered into the mainstream, especially in the area of fitness training with fitness trackers. Marathon runners are no exception. With their ubiquity and proclivity for training technology, they represent an acceptable target market for our application.

2.3 Porter's Five Forces Analysis

We will now conduct a Porter's Five Forces analysis of our mobile application for marathon runners to contextualize it in the industry and develop a strategy for differentiating and promoting it (Porter 2008).

2.3.1 Bargaining Power of Buyers

Buyers have strong bargaining power only when they are consolidated. Consumers of fitness tracking products are numerous, but diffuse in their buying patterns. Buyers are many and demand is great, weakening the bargaining power of buyers.

2.3.2 Bargaining Power of Suppliers

The power of suppliers refers to the power of businesses that provide materials to another business to demand increased prices for materials (Porter 2008). The application is developed for the Android platform, and cell phones have become a commodity, indicating a weak bargaining power.

2.3.3 Threat of New Entrants

New entrants have the potential to bring new ideas to a market. The market of activity monitors poses few barriers and connected fitness trackers are projected to grow from $2 billion to $5.4 billion from 2014 to 2019 (Parks Associates 2015). With the burgeoning of the Internet of Things, it is expected that there will be new players in many applications of telemonitoring. Thus, the threat of new entrants is perceived to be strong.

2.3.4 Threat of Substitutes

A product from a different industry that offers the same benefits to consumers is referred to as a substitute product. Substitute products pose a threat because there is possibility of replacement for a company's product (Porter 2008). One substitute product for runners training for marathons is meeting one-on-one or in small groups with dedicated professional trainers and coaches. There is an approximately $1.5 billion industry existing in intensive personal athletic training in the United States (Witter 2015). This includes firms and independent individuals who provide services granting personalized attention to athletes training for sports seasons or upcoming events such as marathons. However, human trainers conducting in-person training generate problems not seen in the activity monitor/trainer application. For example, scheduling is a factor for this substitute, as the athlete would need to train according to the trainer's schedule and location. Having a human trainer is also significantly more expensive than using an activity monitor. The application does not come with these added cost and conditions. For these reasons we believe that the threat of substitutes is weak.

2.3.5 Rivalry Amongst Existing Competitors

Rivalry can pose a great threat when the rivals compete over price instead of features. The market for tracking and training of fitness, including endurance running, is a crowded one. In this market, our application will be competing with a variety of technologies, such as smartphone apps and specialized fitness tracking hardware. We will need to ensure that

our feature offerings are differentiated in order to avoid significant threat from price-based rivalry.

Wearable fitness tracking devices have seen widespread adoption among runners and other athletes. There are several subcategories of device functionality in this area, ranging in metrics measured, accuracy of these metrics, and price. These include step counters such as the Fitbit One or Nike+ FuelBand at the lower end of functionality and price, GPS-based speed and distance monitors like the Garmin ForeRunner 620 or TomTom Runner at the higher end, and multi-functional devices like smartwatches, such as the Apple Watch or Pebble Time that have some built-in fitness features (Carter 2013).

Other competing fitness devices include specialized peripheral hardware, such as chest straps to monitor heart rate, shoe inserts to track impact and step duration, and devices that help athletes recover from training in terms of bloodflow and muscle relaxation, such as the Firefly Recovery System (Alger 2014). These products are more targeted at health monitoring and feedback for runners, which we can compete with by providing without specialized hardware outside of the mobile phone itself.

Additionally, given the demand for personal training, new products which provide personalized feedback, such as the Moov, have already begun to appear. Moov's successful crowdfunding campaign indicates a demand for fitness trackers that can provide this type of feedback (Colao 2014). Major players are pushing for greater personalization. For example, FitBit, a key player in wearable fitness tracking, acquired the startup FitStar in 2015 (Lawler 2015) which provides users with personalized instructional videos. Finally, our application will be competing with a host of other smartphone fitness applications. A huge market for personal fitness tracking exists in the app stores of the smartphones that so many Americans already carry with them daily. A study (Stanfy 2013) estimated that in 2012 there were over 40 thousand health and fitness applications available for mobile phones, reaching over 500,000 active users, and that number has only increased in the past few years. A wide variety of fitness and run tracking, goal-setting and socially competitive, and motivational applications are available. Some of the most popular apps specifically targeted at runners

are RunKeeper, MapMyRun, and Runtastic (Carter 2013). On the more creative side are apps like Zombies, Run which provides audio motivation in a narrative form, taking a runner through customizable missions in a fictional environment.

Given the great number of players in this industry, the threat of existing rivals is strong. However, given the still largely unexplored area of personalized coaching within the crowded space of fitness tracking technology, we believe that this rivalry will primarily be features-based.

2.4 Technology Strategy

Considering our market research and Porter's Five Forces analysis, we have developed a strategy for our product in order to minimize the threats posed to our product. Our strategy revolves around marketing to customers based on the features offered by our product, particularly focusing on measurement and real-time feedback regarding performance metrics, such as speed and cadence. For instance, despite its importance, many fitness tracking solutions do not measure cadence. In addition, the products that do are typically not transparent about the estimation algorithms used and their accuracies. Even for those that do report accuracy, the algorithms used are still unpublished, and the accuracy of specific metrics, such as cadence, are conspicuously missing (Garmin 2016). Our application uses algorithms backed by published scientific literature, and the accuracy of our implementation will be further measured and published. Furthermore, the framework on which the application is built includes a fault-tolerant client-server protocol for secure and convenient data syncing, and a wide library of well-tested data collection and analytics functionality to support our application's features and ensure they remain reliable and easy to use. Raising the standards of information transparency, estimation accuracy, and application reliability would not only allow our application to gain traction in the market if we were to actively promote it, but would also impose barriers to new entrants.

Bibliography

Alger, Kieran. 2014. "Training for a marathon: Tech for half marathon, marathons and beyond". http://www.t3.com/features/training-for-a-marathon-half-marathon-ultra-and-beyond.

Altini, Marco, Ruud Vullers, Chris Van Hoof, Marijn van Dort, and Oliver Amft. 2014. "Self-calibration of walking speed estimations using smartphone sensors". In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on*, 10–18. IEEE.

Aranki, Daniel, Gregorij Kurillo, Adarsh Mani, Phillip Azar, Jochem van Gaalen, Quan Peng, Priyanka Nigam, Maya P. Reddy, Sneha Sankavaram, Qiyin Wu, and Ruzena Bajcsy. 2016. "A Telemonitoring Framework for Android Devices". In *Proceedings of the 1st IEEE Conference on Connected Health: Applications, Systems and Engineering Technologies*. To appear. IEEE.

Aranki, Daniel, Gregorij Kurillo, Posu Yan, David M. Liebovitz, and Ruzena Bajcsy. 2014. "Continuous, Real-time, Tele-monitoring of Patients with Chronic Heart-failure: Lessons Learned from a Pilot Study". In *Proceedings of the 9th International Conference on Body Area Networks*, 135–141. BodyNets '14. London, United Kingdom: ICST (Institute for Computer Sciences, Social-Informatics / Telecommunications Engineering). ISBN: 978-1-63190-047-1. doi:10.4108/icst.bodynets.2014.257036. http://dx.doi.org/10.4108/icst.bodynets.2014.257036.

Azar, Phillip, Adarsh Mani, Quan Peng, and Jochem van Gaalen. 2015. "Expanded Telehealth Platform for Android". MA thesis, EECS Department, University of California, Berkeley. http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-83.html.

Bluetooth SIG, Inc. 2016. "The Bluetooth SIG Health Device Profile (HDP) Data Exchange Specifications". https://www.bluetooth.com/specifications/assigned-numbers/health-device-profile.

Carroll, Aaron, and Gernot Heiser. 2010. "An Analysis of Power Consumption in a Smartphone". In *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference*, 21–21. USENIXATC'10. Boston, MA: USENIX Association. http://dl.acm.org/citation.cfm?id=1855840.1855861.

Carter, Jamie. 2013. "10 best running gadgets: the top tech for training". http://www.techradar.com/us/news/world-of-tech/roundup/10-best-running-gadgets-the-top-tech-for-training-1157180/1.

Colao, J.J. 2014. "Who Needs Kickstarter? Exercise Sensor Moov Raises $1 Million In 15 Days". http://www.forbes.com/sites/jjcolao/2014/03/14/exercise-sensor-moov-raises-1-million-in-15-days-without-kickstarter/#704c414614e3.

Freescale. 2014. "The next evolution in running". https://web.archive.org/web/20141223192158/http://blogs.freescale.com/iot/2014/12/wearables-next-evolution-in-running-marathon/.

Garmin. 2016. "How accurate are the Running Dynamics from the HRM-Run and how was the accuracy tested?" https://support.garmin.com/support/searchSupport/case.faces?caseId=%7B435cc8c0-e2e1-11e3-47b1-000000000000%7D.

GfK. 2015. "GfK forecasts 51 million wearables will be bought globally in 2015". gfk.com/news-and-events/press-room/press-releases/pages/gfk-forecasts-51-million-wearables-sold-globally-2015.aspx.

Hofmann-Wellenhof, Bernhard, Herbert Lichtenegger, and James Collins. 1997. *Global Positioning System: Theory and Practice*. Springer-Verlag.

Lawler, Ryan. 2015. "Fitbit Confirms FitStar Acquisition To Bring Training To Its Fitness Portfolio". http://techcrunch.com/2015/03/05/fitbit-confirms-fitstar-acquisition-to-bring-training-to-its-fitness-portfolio/.

Mani, Adarsh, Phillip Azar, Jochem van Gaalen, and Quan Peng. 2015. "Expanded Tele-heatlh Platform for Android". MA thesis, EECS Department, University of California, Berkeley. http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-110.html.

Mintel. 2014. *Exercise Trends - US - October 2014*. Market report. Mintel.

Mosa, Abu Saleh Mohammad, Illhoi Yoo, and Lincoln Sheets. 2012. "A Systematic Review of Healthcare Applications for Smartphones". *BMC Medical Informatics and Decision Making* 12 (1): 1–31. ISSN: 1472-6947. doi:10.1186/1472-6947-12-67. http://dx.doi.org/10.1186/1472-6947-12-67.

Mwemezi, Jovin J., and Youfang Huang. 2011. "Optimal Facility Location on Spherical Surfaces: Algorithm and Application". *New York Science Journal* 4(7):21–28.

Nielsen, Jakob. 1993. "Iterative user-interface design". *Computer* 26, no. 11 (): 32–41. ISSN: 0018-9162. doi:10.1109/2.241424.

Parks Associates. 2015. “Global revenues from connected fitness trackers to exceed $5 billion by 2019”. http://www.parksassociates.com/blog/article/pr-march2015-whcc.

Peng, Quan, Phillip Azar, Adarsh Mani, and Jochem van Gaalen. 2015. “Expanded Tele-Health Platform for Android”. MA thesis, EECS Department, University of California, Berkeley. http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-90.html.

Porter, Michael E. 2008. “The Five Competitive Forces That Shape Strategy”. *Harvard Business Review* 86 (1): 78–93.

Running USA. 2015. “2014 Running USA Annual Marathon Report”. http://www.runningusa.org/marathon-report-2015.

Shu, Xianhua, Zhenjun Du, and Rong Chen. 2009. “Research on Mobile Location Service Design Based on Android”. *In Wireless Communications, Networking and Mobile Computing*: 1–4. doi:10.1109/wicom.2009.5302615.

Smith, Grace. 2013. “10 Excellent Platforms for Building Mobile Apps”. http://mashable.com/2013/12/03/build-mobile-apps/#SXF0URANsqqg.

Stanfy. 2013. “Fitness in Mobile: Case-study”. http://www.slideshare.net/stanfymobile/fitness-cs-22962137.

The Berkeley Telemonitoring Project. 2016. “The Berkeley Telemonitoring Project - Privacy-Aware Health Monitoring”. https://telemonitoring.berkeley.edu.

van Gaalen, Jochem, Phillip Azar, Adarsh Mani, and Quan Peng. 2015. “Expanded Tele-Health Platform for Android”. MA thesis, EECS Department, University of California, Berkeley. http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-86.html.

Witter, David. 2015. *IBISworld Industry Report 61162: Sports Coaching In The US*. Industry report. IBISworld.