Mobile On-Call: Design of a Non-Invasive, Non-Intrusive, Personal Vital Signs Monitor

by

Kirsten Zernask-Cebek

Electrical and Biomedical Engineering Design Project (4BI6)
Department of Electrical and Computer Engineering
McMaster University
Hamilton, Ontario, Canada

Mobile On-Call: Design of a Non-Invasive, Non-Intrusive, Personal Vital Signs Monitor

by

Kirsten Zernask-Cebek (0562587)

Department of Electrical and Computer Engineering
Faculty Advisor: Prof. Deen

Electrical and Biomedical Engineering Project Report
Submitted in partial fulfillment of the requirements
for the degree of Bachelor of Engineering

McMaster University
Hamilton, Ontario, Canada
March, 2010

ABSTRACT

Currently, there are limited resources in place for patients who are recovering from illness and surgeries to be monitored at home while trying to resume or maintain a normal lifestyle. The Mobile On-Call system is a low-cost, low power device developed to address these issues. This device is intended to enable greater independence for those patients, while still offering real-time monitoring of their vital signs, and an immediate connection to healthcare. This project delivers a cell phone based program to actively monitor and assess a user's health based on three vital signals; electrocardiogram, blood pressure and body temperature. The results of the analysis are presented to the user via a graphical interface, and communicated to a physician or healthcare practitioner if deemed necessary, over an available cell phone network. It implements algorithms adapted to a mobile phone platform to compress and store the data efficiently.

Keywords: wireless, Bluetooth, data compression, lossless, biomedical monitoring

ACKNOWLEDGEMENTS

Kirsten would like to acknowledge her advising professor Dr. Deen for his continued support and guidance throughout the duration of this project. She would also like to thank Dr. Doyle for being available to discuss algorithm implementation designs and approaches.

Kirsten would like to thank her group members Dhvani Parekh and Sandra Escandor for their support and collaboration over the year, and for being so understanding when things went wrong.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

RLE – run-length encoding, a type of data compression

ECG – electrocardiogram, a description of the electrical activity of the heart

Compression ratio - a reduction in size needed to represent data; defined as (compressed size)/(original size)

J2ME – Java MicroEdition for mobile applications

RF – radio frequency

ASCII – American Standard Code for Information Interchange, a code for representing English letters numbers and symbols

GUI – graphical user interface

Entropy – a measure of redundancy in data, and the theoretical maximum lossless compression possible

1 Introduction

1.1 Motivation

Currently, there are limited resources in place for patients who are recovering from illness and surgeries to be monitored at home while trying to resume or maintain a normal lifestyle. It is difficult for patients to be able to closely monitor their own health on a regular basis from the comfort of their home. A device to accomplish this goal would be advantageous for health professionals as well, as they can check their patients' vitals frequently without requiring an appointment at a hospital or clinic.

1.2 Objectives

The goal of this project is to design a low cost, low power device that continuously and non-invasively measures and stores three vital signals of a patient at home, characterizes the patient's overall health status based on their vitals, and can immediately communicate the patient's condition to a physician or healthcare professional if necessary.

However, continuous monitoring comes at the cost of large volumes of data. A continuous ECG signal alone can require up to 1.3 Gb to store 24 hours of data. [1] Considering we will be monitoring 3 signals, there is a great need to decrease the amount of space required to store the same data. The primary focus of this project is to design data compression algorithms to maximize the storage of the vital signals, and consequently the length of time the device can be worn. A secondary part of this project is to write a user interface for the device that implements the data storage and communication while providing a graphical display for the patient.

1.3 Overall Methodology

The general approach to our project can be broken down into several components, outlined in Figure 1.

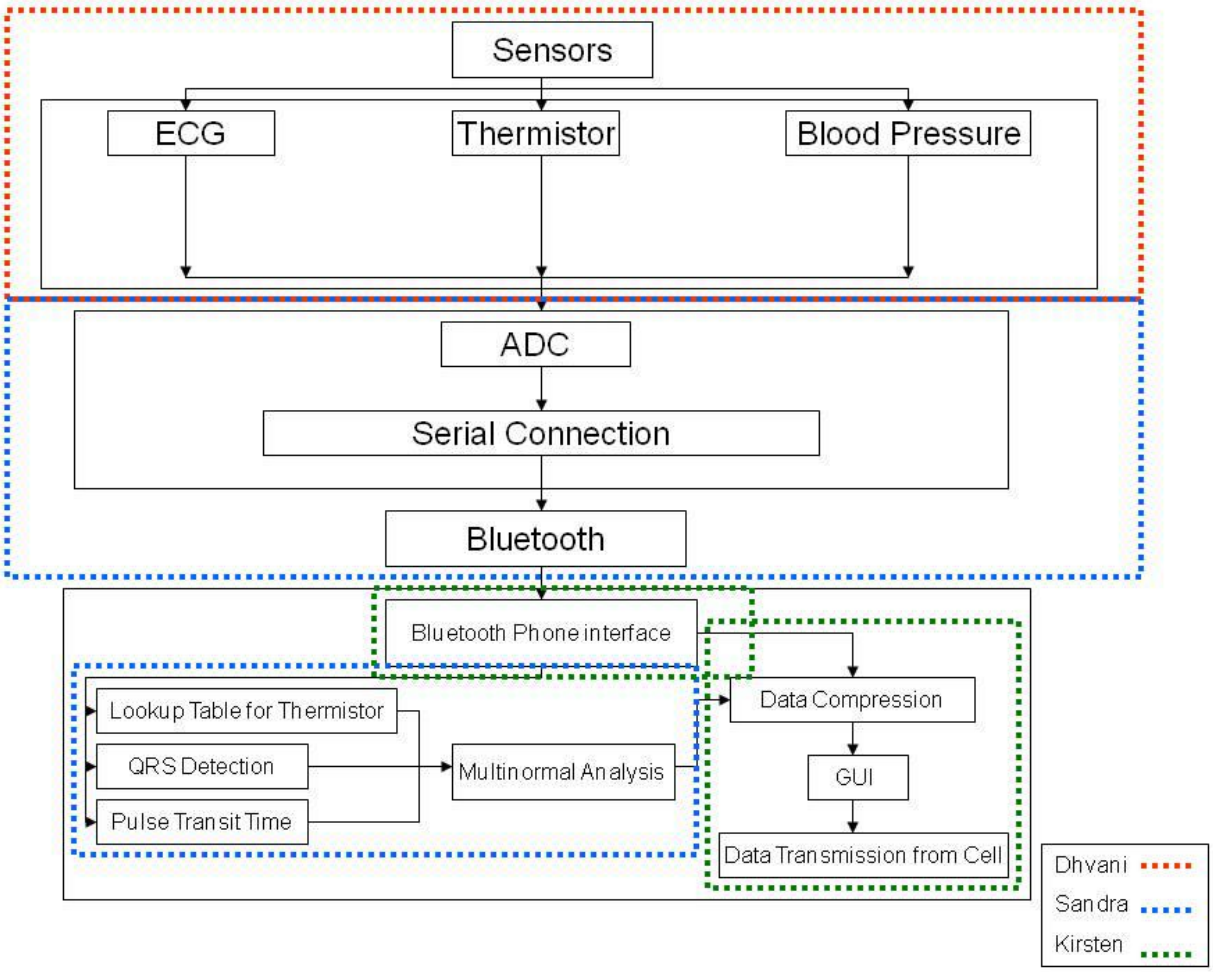

Figure 1. Project Overview and Component Breakdown

The hardware for the acquisition and analog of the 3 physical signals, ECG, body temperature and blood pressure was designed by Dhvani Parekh. The analog to digital conversion of the signals and transmission via Bluetooth was designed by Sandra Escandor, as well as characterization of the health status of the patient via a statistical classifying algorithm called multivariate normal analysis. I was responsible for the Bluetooth interface on the cell phone side, presenting a graphical user interface for the application, determining an action plan for whether or not to notify a physician based on Sandra's classification of the patient's health, and the design of an efficient algorithm for data compression and storage.

1.4 Scope

A person's heart rate, temperature and blood pressure can vary significantly based on emotional state or physical activity. Also, the trends in data between groups of people of different ethnicities can be diverse. The scope of this project is limited to patients who are, for the most part bed ridden, are at home, and have access to a Bluetooth compatible cell phone registered on an available network and are capable of operating simple functions on it.

2 Literature Review

In this section I present a brief survey of several existing devices that address issues similar to those of this project, specifically two commercial devices and two described in research journals.

2.1 Comparable Commercially Available Systems

2.1.1 Acuity Central Monitoring System

The Micropaq Wearable Monitor[1] and accompanying Acuity Central Monitoring System[2] by Welch Allyn, shown in Figure 2, are designed to provide a wearable and wireless monitoring solution and are intended to be able to track up to 60 patients simultaneously in a clinical or a mobile setting. It monitors patient waveforms, has an arrhythmia detection algorithm, and a warning alarm for patient vital signs.

The small Micropaq monitor is hooked up to the patient, and displays heart rate, SPO2 and pulse rate on an LCD screen. It is also built to be drop resistant, and communicates with the base station over a 2.4GHz wireless LAN network.

The system as a whole has several key benefits. It is equipped with an oxygen monitor, can control and observe data from multiple patients wearing Micropaqs, shows patient waveforms on a large computer screen at the base station and will even give an alert when a Micropaq goes out of range of the wireless network.

Modification of allowable ranges for patient vitals is permitted, but the inexperienced user runs the risk of setting them incorrectly. Other drawbacks include a 24 hour limit on storage of patient data, a requirement of extra hardware for a network access point and a loss of functionality out of range of the base station because the device has no data storage ability, and requires that all data be stored on the base computer.

---

[1] http://www.welchallyn.com/products/en-us/x-11-ac-100-0000000001100.htm

[2] http://www.welchallyn.com/documents/Patient%20Monitoring/Continuous%20Monitoring/Acuity/usermanual_20070323_acuitymobileltwireless.pdf

The most substantial drawback is the cost. A base station costs approximately $40 000, and a single Micropaq $3300, which makes it unfit as a low cost monitoring solution.

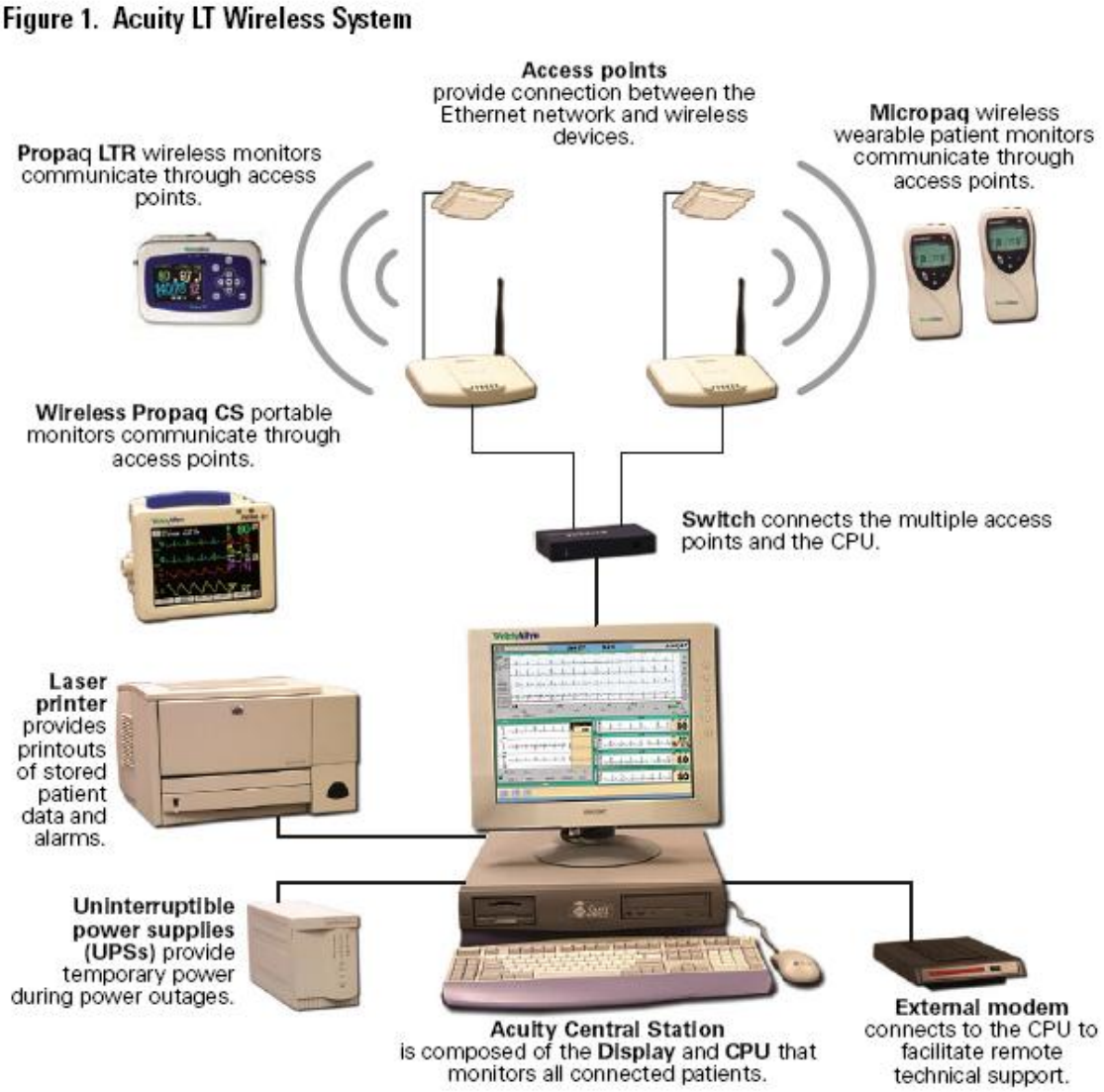

Figure 2. Welch Allyn Monitoring System

2.1.2 Mark of Fitness Blood Pressure Monitor

A second commercial device is the Mark of Fitness MF-77 Wrist Blood Pressure Monitor[3]. It is a relatively low cost blood pressure monitoring system designed for personal use, and has even been adapted by IBM research to communicate with a Sony Ericsson P900 cell

[3] http://www.bloodpressuremonitor.bz/mark_of_fitness_mf77_blood_pressure_monitors.asp

phone via a Bluetooth link[4]. This device has built in graphing to show blood pressure on its screen, shows histograms and correlation charts on data, and only costs $194.90 for the monitoring system and computer interface package.

However, it doesn't automatically connect to a computer or a cell phone to store data but requires an interface cable, uses an intrusive cuff to take measurements and measures only blood pressure and pulse per minute.

2.2 Comparable Research Devices

2.2.1 iCalm

Another new device the iCalm, developed by researchers at MIT, proposes a low cost, low power design of a wearable and wireless system to monitor electrodermal activity, temperature, motor activity and blood volume pulse. [2] The device continuously analyses these signals, and attempts to determine the patient's affective emotional state. It is designed for outpatient and long term studies, and addresses the need to monitor a patient over a long period of time to be able to customize its analysis. The iCalm uses non-traditional placement of electrodes, to increase comfort for the user, but gives a significantly lower output result compared to commercial devices of its class, and does not provide any worthwhile data for the first 15 minutes every time it is used.

2.2.2 Prognosis

Finally, the device Prognosis is aimed at provided continuous ambulatory monitoring of patients who are elderly or who suffer from long term chronic diseases and incorporates wearable sensors, as well as sensors embedded in the home, to collect data unobtrusively. [3] It provides the opportunity for determination of individual user baselines, which integrate with processing algorithms to autonomously discover health trends and concerns, inform medical professionals, and integrate with emergency responder's equipment. The device also includes speech recognition, to obtain feedback from the user about other symptoms they may be experiencing.

[4] http://domino.research.ibm.com/comm/pr.nsf/pages/news.20031112_mobilehealth.html

3 Statement of Problem and Methodology of Solution

This section focuses on the problems that have to be addressed in order for the program to function as a whole. The project can be sectioned into several major blocks, outlined in Figure 3.

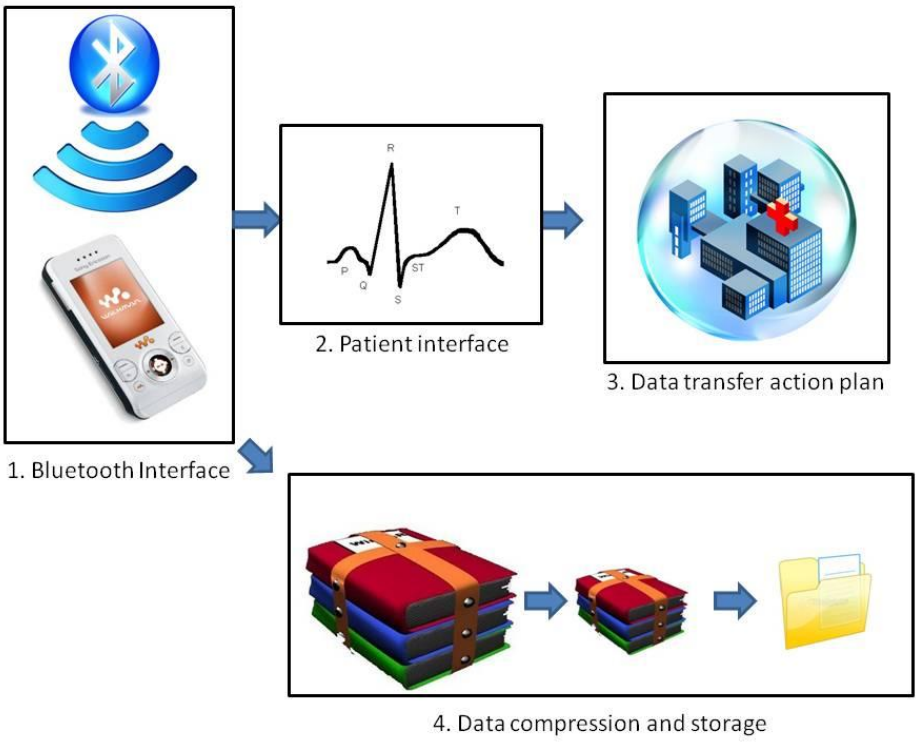

Figure 3. Basic blocks of problem

The first block describes the problem of the wireless link from the acquisition device to the cell phone via Bluetooth, the second deals with the graphical user interface developed for the patient to operate the device, the third details the control of data management and communication from the device to a physician, and the fourth section is a discussion of how to effectively store a large amount of continuous data on a memory limited platform.

3.1 Platform Selection and Bluetooth Interface

Before I discuss the specifics of the methodology, I present a description of the cell phone and programming platform. In the design of a mobile phone based application, there are numerous options for implementation; I have chosen to build my application on a Sony Ericsson W580i. This selection is not nearly as critical as the selection of the operating platform. I am designing my application to run on a J2ME platform. Because virtually all cell phones and PDAs these days are capable of running Java applications, this application will ultimately be able to run on any mobile device that has a built in Bluetooth functionality. This portability removes the constraint on the operating device, which means most patients who own a cell phone should be able to use the monitoring device, effectively reducing the cost of the overall apparatus.

The selection of Bluetooth as a means to wirelessly transmit data from the wearable device to the mobile phone is in keeping with the goal of portability and reduced cost and power. Other transmission options exist, such as the low power ZigBee [4], or other RF based transmitters and receivers, but they require extra hardware for receivers attached to the cell phone which adds to the size and cost, and detracts from the ability to transfer the device between phone types. Currently, most cell phones have built in Bluetooth receivers that are accessible by custom applications making it an effective and convenient solution for a wireless link.

3.2 Patient Interface

To control the operation of the device on the cell phone, a graphical user interface is provided for the patient. The interface provides a means for the patient to keep track of their own health status, and displays their vital signs with an indication of where the values fall within their normal range. It offers the option to view the ECG waveform in real time, and alerts the patient if the status of their health becomes a concern.

3.3 Data Management and Communication

To ensure each block of the software functions together, a logical flow of data has to be designed and maintained. The overall control structure has to pass the data through the multivariate normal analysis to determine the patient’s health, and then design an action plan for

the data based on the results. In all cases, the data is displayed for the user, with gauges indicating where the current signal falls within the normal range. In the case that the user's vitals become critical, the program implements a function to immediately notify a physician of the patient's status and alerts the patient.

Another issue that needs to be addressed is what to do if the phone receives an incoming call or text message. The application has to be set up to maintain the Bluetooth connection and continue analysing and storing the data until it regains control of the phone.

3.4 Data Compression

3.4.1 Background

For an application based on a mobile phone, available memory is limited, and can become an issue for storage. This project requires that a large amount of data be saved continuously, so there is a great need to maximize storage capability. An obvious solution to this problem may appear to be simply adding more memory. However, while it is possible to expand the available external memory on a cell phone a certain amount, via larger memory cards, there is a limit to the extent this is possible. In the case of the Sony Ericsson W580i, the largest memory card available is 2GB. [5] Due to the limited options with regard to memory expansion, there is a need for another solution to maximize storage.

Data compression provides an effective solution as it involves decreasing the memory requirement for a certain amount of data. By reducing the amount of space needed to store the same amount of data, it becomes possible to store significantly more, with the same available memory. It is important to note the difference between compression algorithms that are optimized for immediate data transmission versus those optimized for storage. Immediate transmission requires an algorithm to operate in real time, whereas a storage application can allow for several passes through the gathered data. Although the data acquired in this project will be stored in the memory on the cell phone, it may need to be accessed immediately, and as such, a real time compression is required. To justify the selection of the algorithms implemented in this project, a discussion of the types of data compression and several algorithms that were considered, but ultimately rejected is included.

3.4.2 Lossy Compression

To begin, there are two main classifications of data compression, lossy and lossless. Lossy compression stores data in a way that it can be recreated approximately, but not exactly. Since the entire data is not saved precisely, lossy compression can often achieve a higher compression ratio than what is referred to as lossless compression, which is described later. Thus, for an application where the data needs only to be represented approximately, lossy compression is ideal. Depending on the application, a lossy algorithm may be adequate for the compression of an ECG waveform. For example, in the characterization of arrhythmia, a condition of an abnormal heart rhythm, [6] information about where the beats occur is sufficient.

3.4.3 Lossless Compression

For the purpose of this application, a lossy compression is not feasible. The intent of the monitoring device is to identify abnormalities of as many types as possible and is not limited to a specific characteristic of the ECG waveform, so the loss of integrity of any of the data is unacceptable. Thus, lossless compression is necessary. Lossless compression, as the name suggests, exactly maintains the original data, without any loss of integrity. [7]

3.4.4 Huffman Compression

The original algorithm I intended to implement is that of static Huffman coding, which is able to achieve close to the best possible compression ratio for a long set of data with a relatively small set of distinct symbols. [7][8] In fact, it is a benchmark to which other data compression techniques may be compared, and is the basis for many of their designs. [7] Although I eventually abandoned this method for reasons examined later, it is important to include a discussion of how it works, and I will use it as a benchmark for comparison for the techniques I implement.

Huffman coding is a form of entropy coding, and takes advantage of the probability model of a message. Essentially, once all the data is acquired, a library of all the unique symbols found in the data is compiled, and the frequency of occurrence of each symbol is determined. Intuitively, for a sufficiently large sample, the frequency of each symbol occurring is proportional to its probability. Once this library is assembled, a unique code is assigned to each

symbol, with the length of the code inversely proportional to its frequency. Each symbol in the original data is converted to its corresponding code and then stored. Because the most common symbols have the shortest code, the resulting collection of coded data is significantly smaller than the original. [7][9]

An example of Huffman coding that clearly illustrates both its function and significance is Morse code. In fact, Morse code predates the computer by around 100 years [8], the first telegram being sent in 1844. [10] A precursor to the numerous types of wireless communication today, the telegraph was finally discontinued in 2006, after 162 years. [10] At the base of this revolutionary technology is the idea of Huffman coding.

The basic library of symbols for Morse code is the English alphabet. Figure 4 presents this library and the unique code assigned to each letter, and Figures 5 and 6 present the English alphabet according to frequency. The full Morse code library includes the set of numbers 0-9 as well as some punctuation and several other symbols, but these are not necessary for the sake of this discussion.

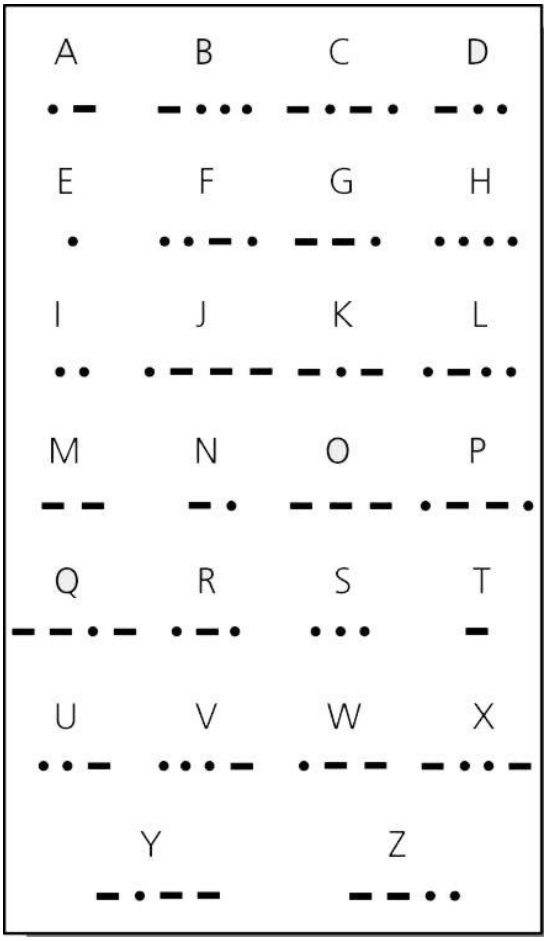

Figure 4. Morse Code [11]

e t a o i n s r h l d c u m f p g w y b v k x j q z

Figure 5. English alphabet arranged according to frequency of occurrence [12]

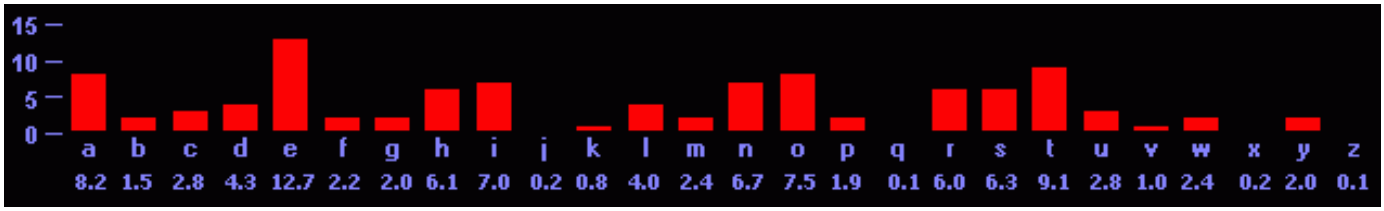

Figure 6. Frequency percentage of letters in the English language [13]

The most frequently occurring letters, E and T are represented by a single dot and a single dash, respectively, where as the least frequent Q and Z have a combination of four dots

and dashes each. Treating each dot or dash as if it were a single bit, to represent this library with a fixed length code for each word would require the equivalent of 5 bits per letter. As a simple example, the phrase ‘data compression’ would require 75 bits. If it were represented using the variable length Morse code, it would only require 38 bits, a compression of about 50%.

While they are the same in principle, one vital difference between Morse code and Huffman coding is the separation of symbols. When transmitting a signal via telegraph, a pause is used to indicate separation between letters. Alternatively, Huffman coded data is saved as a continuous stream of bits in a file. Thus, to clearly distinguish between symbols in a file, each code must be unique in that no one symbol is contained within another. [14] For example, the sequence of the Morse code letter A can be found in the code for the letters C, F, J, K, L, P, Q, R, V, W, X and Y, and would be indistinguishable if there was no separation of symbols. Huffman coding overcomes this issue by designing a library in which the code for any symbol is not contained in the code for any other symbol.

There are a few reasons that led me away from Huffman coding. Primarily, as aforementioned, the signal has to be processed in real time. Unfortunately, Huffman encoding requires two passes through the data; one to build the library and assign codes corresponding to frequency, and second to encode all the data. An alternate solution that satisfies the real time requirement is to create a library based on an initial sampling of data, and use this library for all consecutive data. However, an ECG signal varies over time and a static library is unlikely to provide efficient compression. Adaptive Huffman coding extends the original algorithm to adapt its library continuously with changing data and only requires a single pass through the data. [7]

Although these algorithms would work well if implemented on a computer, they are not feasible for a mobile application. Any implementation of a Huffman compression algorithm requires enough overhead memory to store the entire library of symbols and assigned codes, and the adaptive method requires extra computational power, both of which are limited resources for a cell phone. For these reasons, I selected two much simpler algorithms that require minimal processing power and overhead memory while still significantly compressing the data. Due to the nature of the signals, I implement Run-Length Encoding to store blood pressure and body temperature, and Delta Encoding to store the ECG signal.

3.4.5 Run-Length Encoding (RLE)

Run-Length Encoding is a simple, computationally minimal type of data compression that is effective for sets of data with strings of repeated elements.[9] Signals like blood pressure and body temperature do not change instantaneously and since they are sampled many times a second the consecutive readings are constant over short periods of time. Thus RLE is an incredibly effective solution to compress these signals.

The basic RLE algorithm is straightforward and elegant. A sequence of identical consecutive symbols is replaced by the count of symbols, and one instance of the symbol itself. [10] For example, the string 'AAAABBCCCCA' if represented in 8-bit ASCII, requires 88 bits. If compressed using RLE, the string would become '4A2B4C1A' which requires only 64 bits, a compression of 72%. The compression ratio becomes much better as the lengths of repeated symbols increases. Unfortunately, the inverse is also true, and for data that has little to no consecutive repeated elements, the resulting files could actually increase in size. An ECG signal varies rapidly with respect to blood pressure and body temperature, and is not likely to have any consecutive repeated elements, making it a very poor candidate for RLE.

3.4 6 Delta Encoding

Delta or difference-value encoding is another straightforward compression type that is much more suitable for an ECG signal than RLE. Figure 7 depicts the basic idea behind delta encoding.

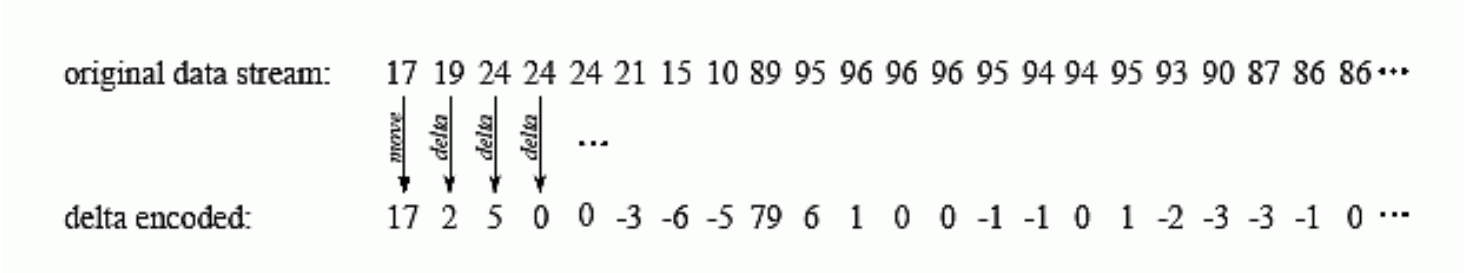

Figure 7. Delta Encoding [14]

In delta encoding, the first value in the compressed file is equal to the first data point and successive samples are the variation between the current and last sample in the original file.

Essentially, the first value of the original data is saved as a starting point, and each subsequent point is saved as the difference between consecutive samples. This method works very well for continuously changing data, as shown in Figures 8 and 9 which depict a digitized audio signal, and delta encoded version of the signal respectively.

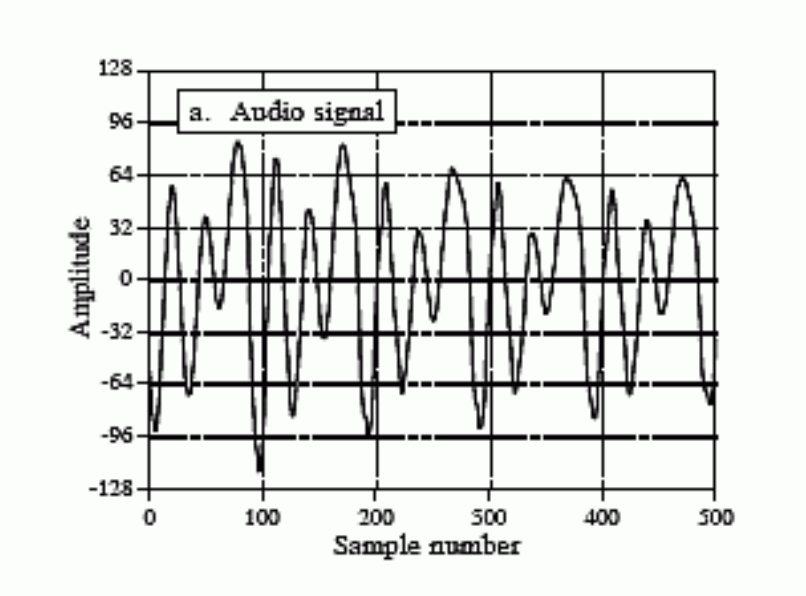

Figure 8. Effectiveness of delta encoding for a continuous signal [14]

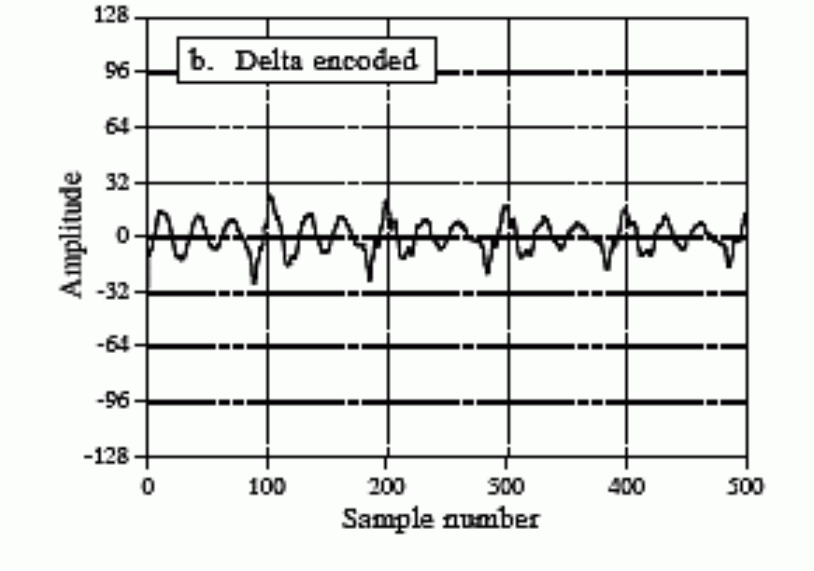

Figure 9 . Delta Encoded Version [14]

Although an audio signal is represented, an ECG waveform functions in a similar manner because they are both continuous physical signals that vary frequently with time. Fundamentally, the overall size of the data is significantly reduced. Interestingly enough, the entropy of the delta compressed file is much smaller than that of the original signal, [14] so provided with extra memory and processing power, a Huffman compression could be run to achieve an even greater amount of compression.

4 Experimental and Design Procedures

4.1 Design of Bluetooth Interface

The flowchart for the design of the Bluetooth interface is represented in Figure 10.

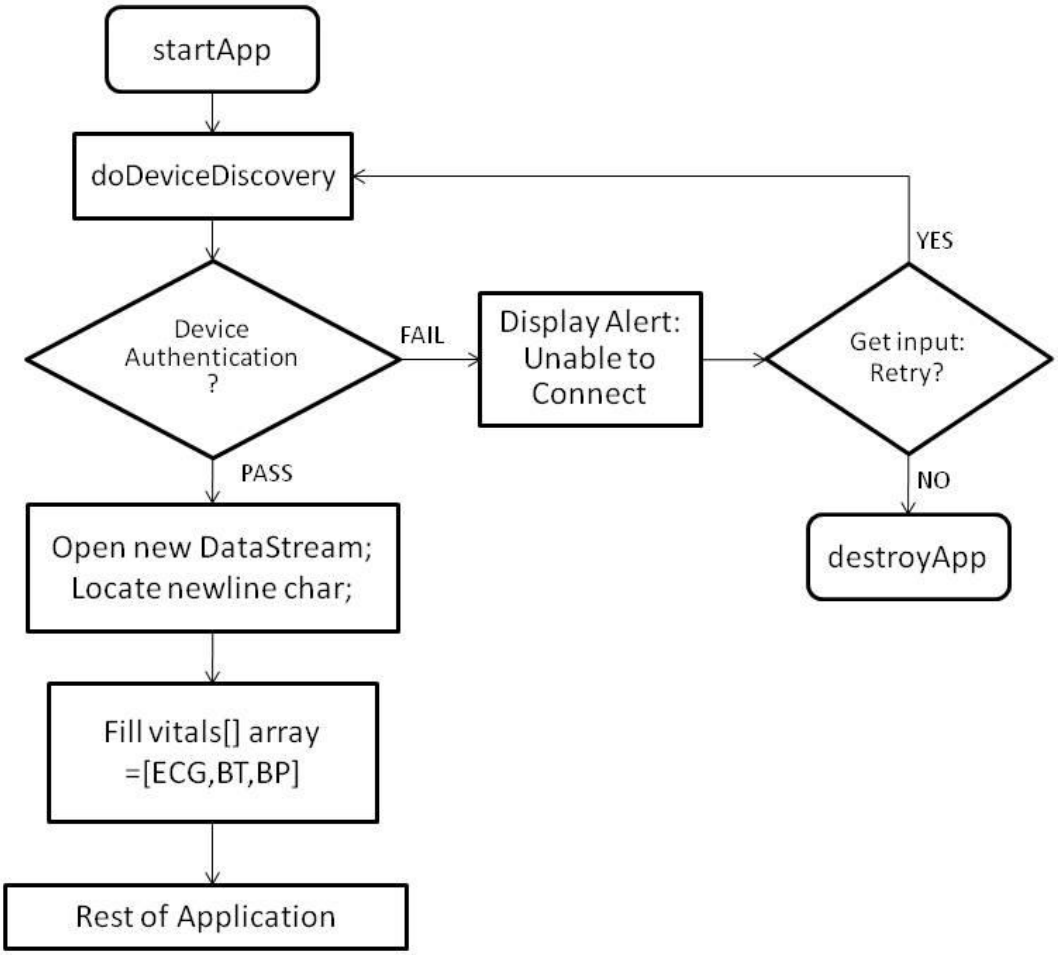

Figure 10. Flowchart for design of Bluetooth interface

As depicted in the flowchart, upon start up the application performs a search to discover the Bluetooth device, and attempts to connect to it. If the connection is not successful, the application alerts the user, and presents an option to retry. If the user chooses not to retry, the application is terminated, otherwise the application reattempts device discovery. Once the device is connected, a new instance of a data stream is invoked. This data stream is set to contain the

information being sent from the Bluetooth transmitter. The transmitted data is represented in comma separated channels, using ASCII code for each character of data. It is sent with four channels, for ECG, body temperature, blood pressure and a new line symbol respectively. Thus, once the data stream is started, the ASCII characters are read until a new line symbol is found. The next channel is then for ECG, and the array vitals[], containing one time point of each of the three signals can be compiled.

4.2 Design of GUI

The flowchart for the design of the graphical user interface is represented in Figure 11.

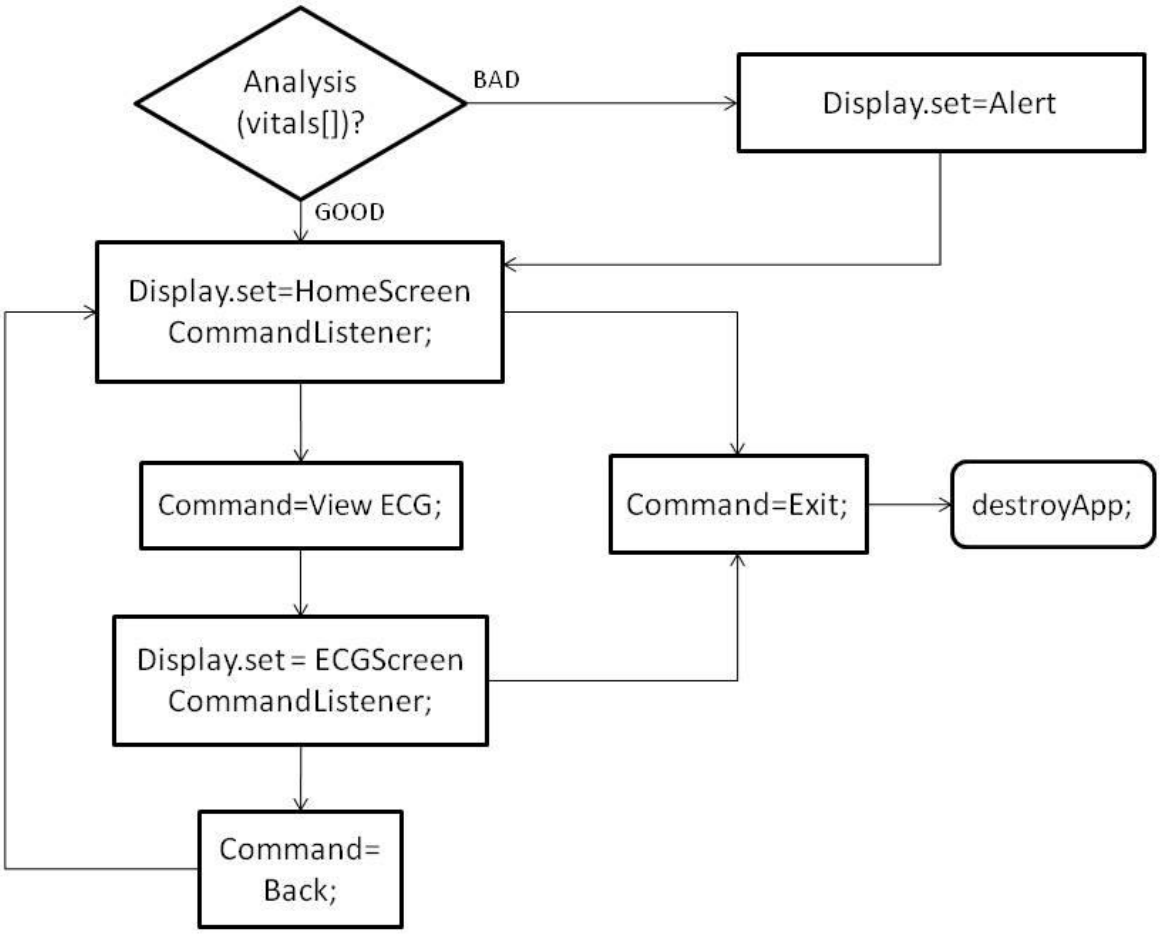

Figure 11. Flowchart of graphical user interface

After the application runs the set of three data points through the multivariate normal analysis, the resultant data points are displayed on the screen. Underneath each reading is a gauge, indicating where each data point falls within the patient's normal range for a particular signal. If the results of the analysis indicate a problem in the patient's condition, an alert is displayed on the screen, before the home screen is displayed again. The code for each Display, or screen object implements a command listener, which recognizes key strokes input by the user that correspond to specific menu options. When a command is detected, the program takes appropriate action. In my application, the home screen has two available commands, Exit and View ECG, while the ECG screen has a Back command and an Exit command. In both cases, the Exit command invokes the destroyApp{} function, which closes the application. The View ECG command makes the ECG screen the visible display, and the Back command reverts back to the Home screen.

4.3 Design of Data Management and Communication

The flowchart for the design of the data flow is represented in Figure 12.

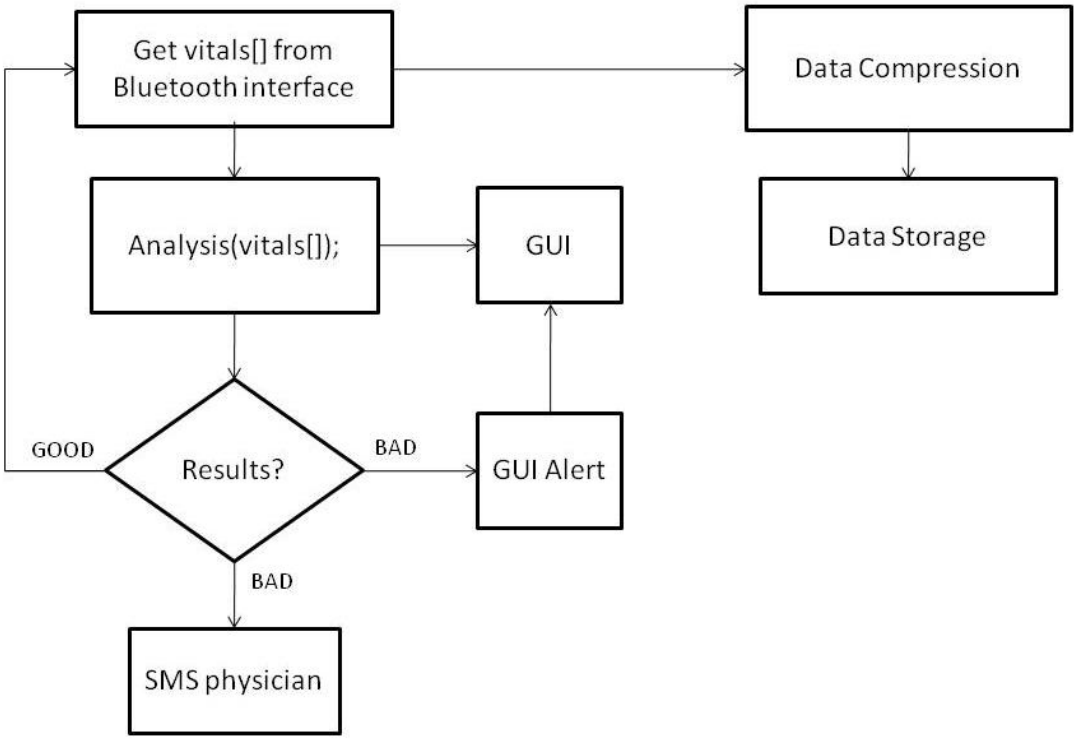

Figure 12. Flowchart for data management and communication

After acquiring the set of vital signals, the program passes them to data compression and storage. It also passes them simultaneously to the analysis algorithm, and displays the results on the screen. If the results indicate the patient's health is normal, the process is repeated for the next set of data. However if it is out of the normal limits, the patient and the physician are alerted. If the results of the analysis show that the data is significantly out of normal limits, and the patient is in critical condition, an SMS (or short messaging service) message is sent immediately to a physician. However, if the vitals have been hovering close to an extreme limit of the range of a patient's normal signal, something may be wrong even though it does not seem initially critical. Figure 13 presents detail on how the decision determining an action plan is to send caution data to a physician is made.

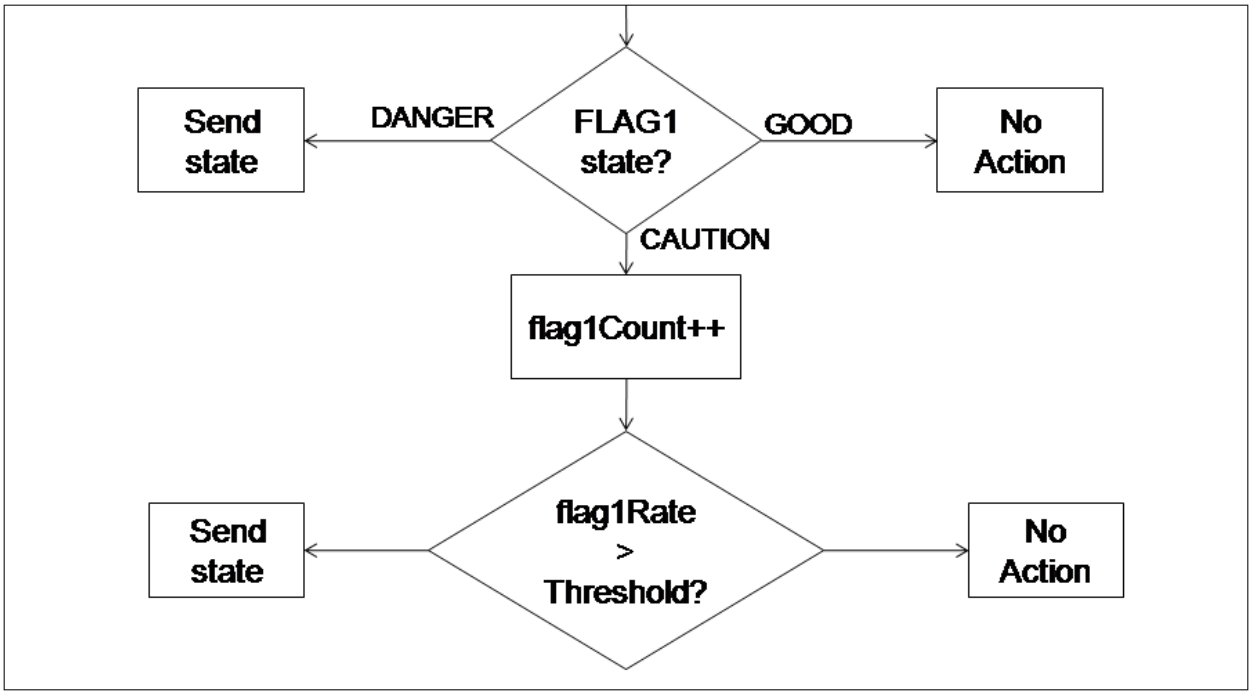

Figure 13. Action plan decision

For each signal, the program keep a count of consecutive values that are very close to the outer bound of the normal range by setting a flag each time a value comes close, and increasing its count. If this occurs for too long (determined by a set threshold), a new and potentially harmful trend may be occurring. An SMS message is sent to the physician to advise them take a look into the patient's data just in case something is wrong.

4.4 Design of Data Compression

The flowchart for the run length encoding algorithm used to compress the body temperature and blood pressure waveforms is shown in Figure 14.

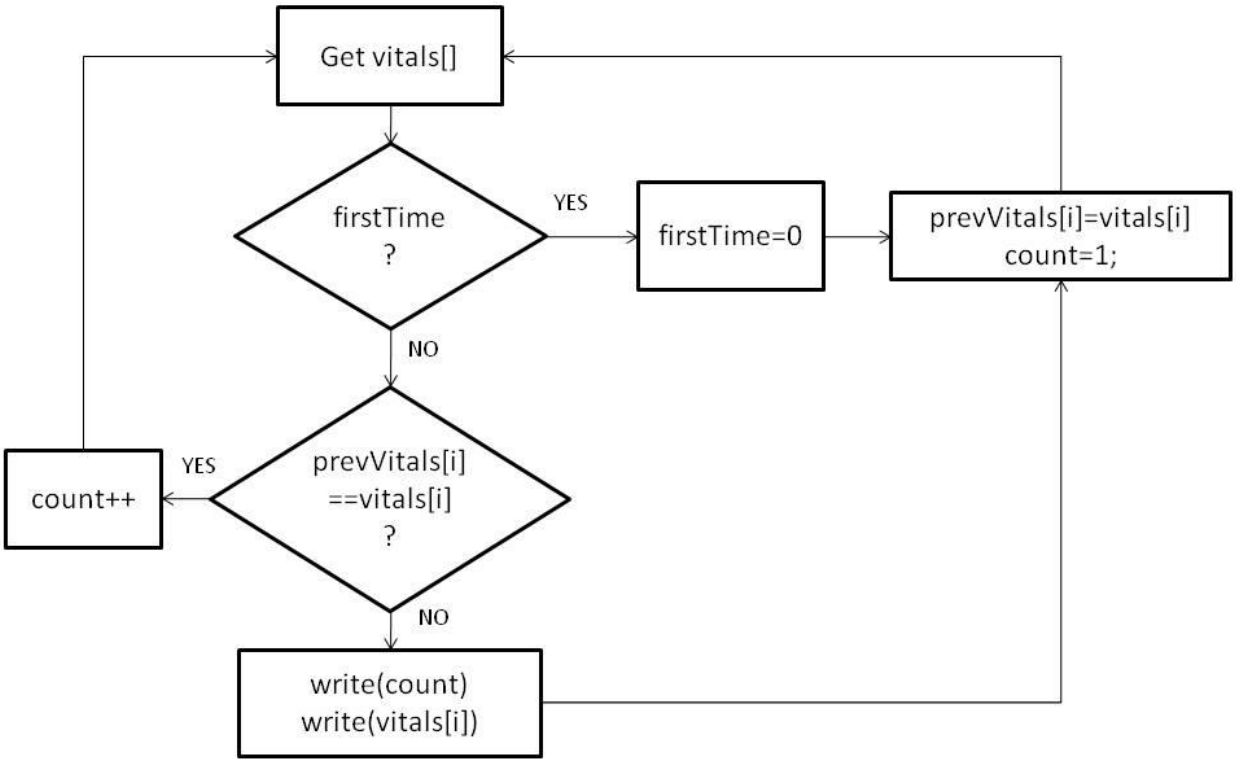

Figure 14. Flowchart for Run-Length Encoding algorithm

The RLE algorithm takes the current set of vitals[] and determines whether or not this is the first point in the file. FirstTime is a Boolean that is initialised to 1, and then set to zero after the first time running through the code. The index i is equal to 1 for body temperature, and 2 for blood pressure. If the current value is equal to the previous value, the code increments a variable to count the number of identical consecutive values. If it is different, the code writes the current count value and the corresponding vital sign to the file. It should be noted that the file name in the final java implementation includes a timestamp indicating the time of the first data point. Since the sampling rate is known, the time of all subsequent points can be inferred thus eliminating the need to store a timestamp for each data point within the file, and significantly

decreasing the overall file size. The code for the Matlab implementation of RLE is included in Appendix A2.

Figure 15 depicts a flowchart of the algorithm for delta encoding used to compress the ECG waveform.

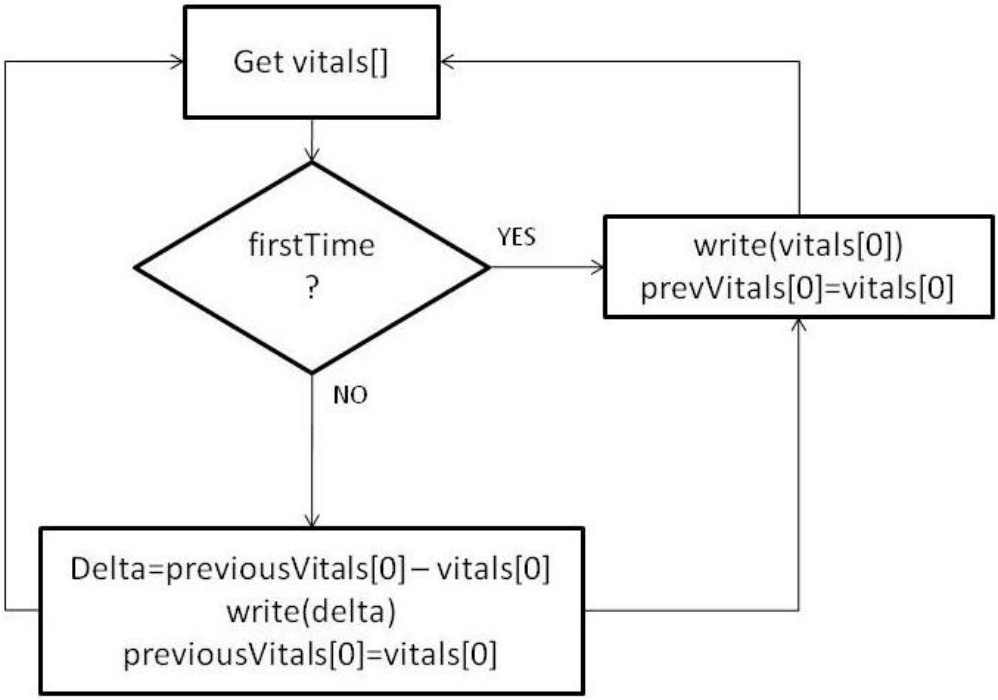

Figure 15. Flowchart for Delta Encoding algorithm

The delta encoding algorithm deals only with the value in vitals[0], the ECG signal. It first checks whether or not this is the starting point. If it is, the value of the data point is written to the file. If it is not the first point, the difference between the current point and the previous point is written to the file. The current data point is then stored as the previous point for the next set of data. The code for the Matlab implementation of delta value encoding is included in Appendix A3.

4.5 Experimental Procedures

To test the proficiency of my data compression algorithms, I have developed a design procedure that compares the results of each of my algorithms when applied to several sets of sample data. My goal was to achieve a minimum reduction of 50%, which effectively doubles the length of time the device can be used.

For the sake of uncomplicated analysis, I initially implemented both my algorithms in MATLAB. I ran the algorithms on several sets of simulated data as well a set of sample data recorded using a sensor developed by Dhvani. I generated sets of sample data for blood pressure using a method similar to Sandra. The sample sets can be varied by length, and generate a Gaussian distribution centered around the normal systolic blood pressure 120mmHg. The code for the sample generator is included in Appendix A4.1.

I also combined my two implementations to run consecutively. I compressed the sample with the delta value algorithm first, and then applied run-length encoding to the compressed file to see if I could achieve better results than just one algorithm. Although I run them consecutively, the two algorithms could be combined such that only one pass through the data is necessary to run both simultaneously, which would make it a viable solution to implement on the cell phone.

5 Results and Discussion

In this section I discuss the performance of my data compression algorithms on several sets of generated sample blood pressure data. The following table summarizes the results I have obtained using the sample data for each compression algorithm. Each set of sample data contains 10000 data points, representing 50 seconds of data when sampled at a rate of 200Hz.

Table 1. Summary of Data Compression Results

Data Set	**Original File**	**Run-Length Encoding (RLE)**		**Delta-Value Encoding (DV)**		**RLE after DV**	
	Size (Kb)	Size (Kb)	Ratio	Size (Kb)	Ratio	Size (Kb)	Ratio
BP_Sample1	49	25	50.9	32	65	20	41
BP_Sample2	49	24.7	50.63	31.6	64.89	18.9	38.57
BP_Sample3	48.8	24.1	49.36	31.6	64.76	18.6	38.11
BP_Sample4	48.8	24.7	50.62	31.6	64.84	18.9	38.57
Average Compression Ratio			**50.38**		**64.87**		***39.06***

5.1 Results of Run-Length Encoding

As shown in Table 1, I achieved an average compression ratio of 50.38 with my run-length encoding algorithm. Figure 16 shows the result of RLE compression graphically for the data set BP_Sample1.txt (included in Appendix A4.2).

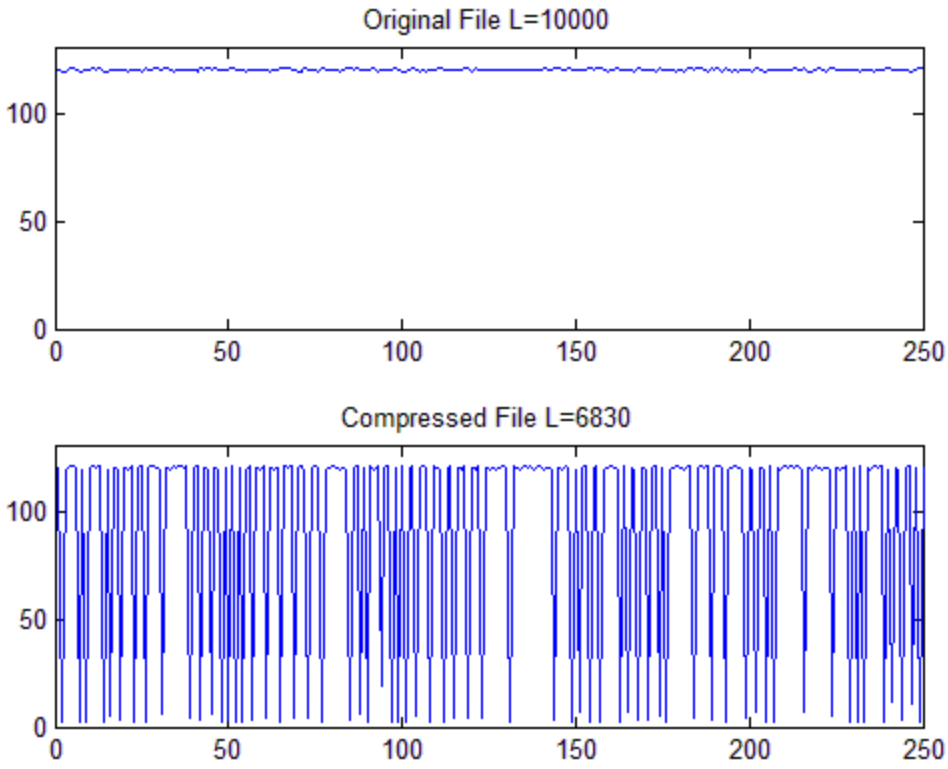

Figure 16. Run-Length Encoding Compression Results

From Figure 16 it is clear from the plots of the first 250 data points that while the original file has an average value around 120, the compressed file frequently has much smaller values around 1 and 2, which requires less memory than a value of 120. Also we see that the overall length of the compressed file is 6830 elements versus the original sample size of 10000. Since many data points in the compressed file require less memory to store than the data points in the original file and the number of data points for the compressed file is significantly less than the original, the compressed file is half the size of the original.

5.2 Results of Delta-Value Encoding

As shown in Table 1, I achieved an average compression ratio of 64.87 with my delta-value encoding algorithm. Figure 17 shows the result of delta-value compression for the same data set, BP_Sample1.txt.

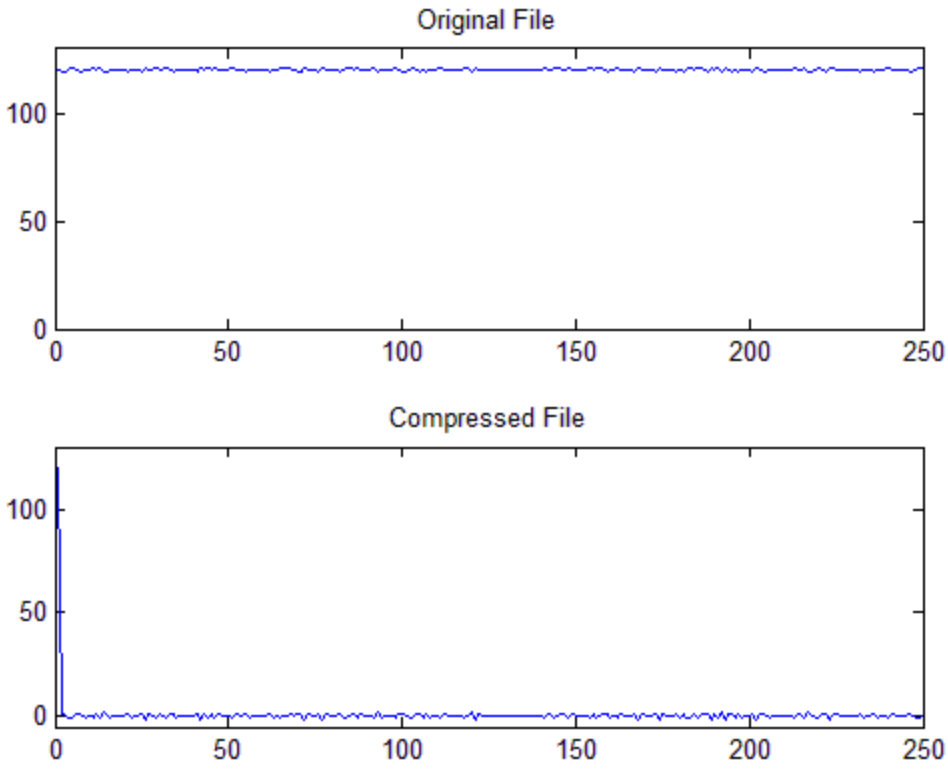

Figure 17. Delta-Value Encoding Compression Results

From Figure 17 it is clear from the plots that while the original file has an average value around 120, the compressed file frequently has much smaller values, requiring less memory per data point than a value of 120. Since all the data points in the compressed file, with the exception of the first point, require less memory to store than the data points in the original file, the compressed file is a fraction of the size of the original.

5.3 Results of Combined Algorithm Compression

As shown in Table 1, I achieved an average compression ratio of 39.06 by combining my two algorithms. Figure 18 shows the result of combined algorithm compression for the same data set, BP_Sample1.txt.

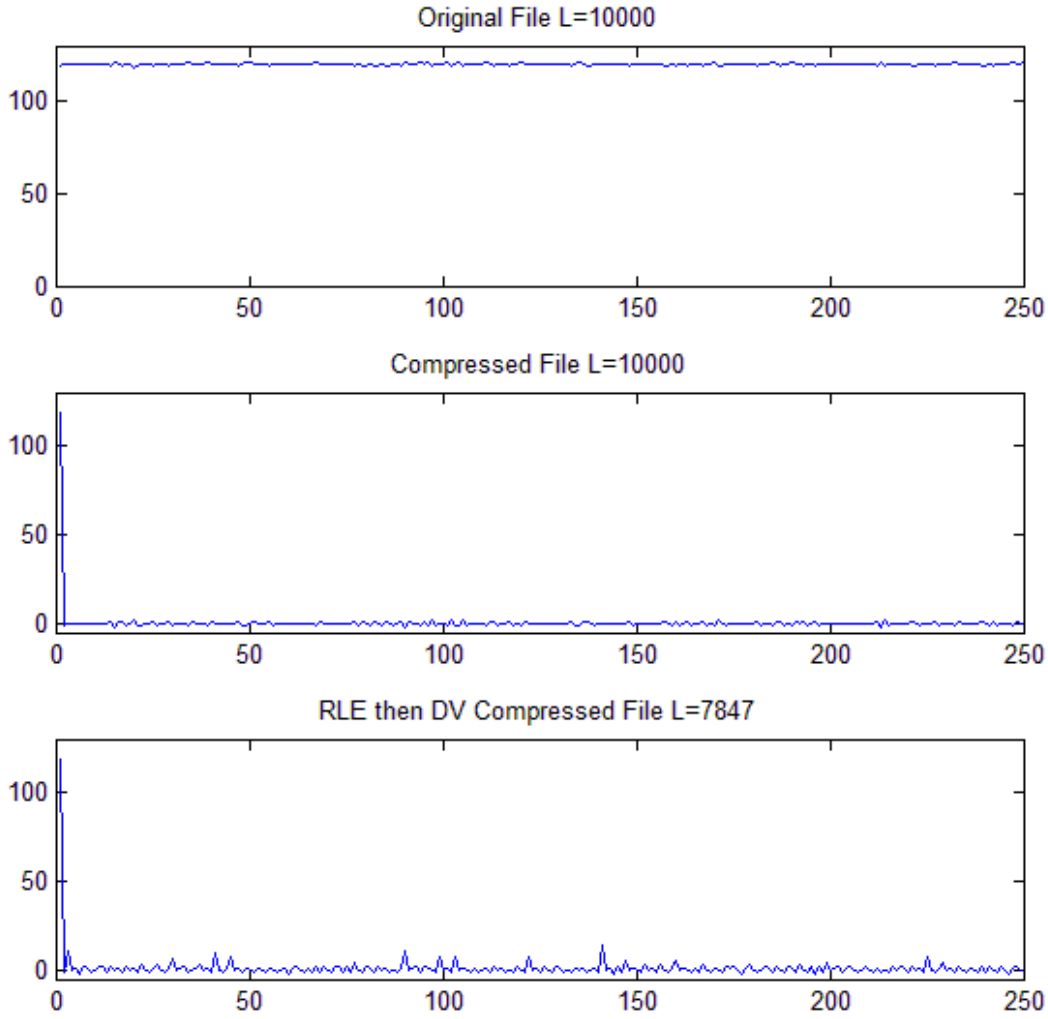

Figure 18. Combined Algorithm Compression Results

To illustrate the effect of each stage of compression on the final result, I show the results of consecutive implementations of the algorithms, and not parallel or simultaneous (Matlab code included in Appendix A5). Figure 18 shows how first applying a delta value compression reduces the average size of the data points, and by applying run length compression after, the total number of data points required is reduced. With a compression ratio of 39, more than double the length of data can be stored versus an uncompressed implementation

6 Conclusions and Recommendations

6.1 Conclusions

The design of the software for the Mobile On-Call system addresses all of the primary goals stated in the objective. The result is a design for a low cost, low power system that continuously monitors three of a patient's vital signs while they are at home, and facilitates a communication link between the patient and a physician in the event that the patient requires treatment.

6.2 Recommendations for Future Work

For the purpose of a beginner classifying program, the scope of this project was limited to a mainly bed-ridden patient. A continuation of the project could involve the extension to an ambulatory monitoring device, with adaptations to accommodate for normal ranges of patient vitals during different types of motion. The Sony Ericsson W580i has a built in accelerometer that could be used to help determine the type of motion the patient is partaking in. With controls for ambulatory monitoring available, there is a large increase in potential uses for the device. One application could be a continuous monitoring system for people identified as being at risk for certain diseases, or suffering from chronic illness. Complications could be identified as soon as, or even before they happen, and be dealt with immediately in either a reactive or preventative manner.

Another extension would be the addition of two way communication, which was not achieved during this project. The design of a remote server to interface with the cell phone application and facilitate the collection of data independent of patient interaction would allow physicians to perform routine check-ups without disrupting the patient. Immediate access to a patient's vital would be possible in the event of a health complication, and a central server based in a clinical setting would easily be able to monitor multiple patients simultaneously.

Finally, in the event of a critical patient complication, functionality for the application to auto-answer a call and place the physician on speaker phone would allow for immediate contact with a patient who may be incapacitated.

Appendix A

A1. List of Computer Programs Used

Microsoft Office 2007 (Word, Excel, Powerpoint), MATLAB, NetBeans IDE

A2. Matlab Implementation of Run-Length Encoding

```
%RLE Compression
clear all;close all;

%open file of sample data
fid=fopen('BP_Sample5.txt');

%open file to write compressed data to
fid2=fopen('rlecompressed.txt','w');

%open file to write original data to compare to file size of compressed
%data
fid3=fopen('rlecompare.txt','w');
[A, length1]=fscanf(fid,'%f');

%%encoding algorithm
%initializes count instead of firstTime flag that is implemented in java
%code
count=0;
size_B=0;
countB=1;
for i=1:length1
    if count==0
        prev=A(i);
        count=1;
    else
        if prev==A(i)
            count=count+1;
        elseif count==1
```

```
            size_B=size_B+fprintf(fid2,'%g\n',prev);
            count=1;
            B(countB)=prev;
            countB=countB+1;
        else
            size_B=size_B+fprintf(fid2,'*%g\n%g\n',count,prev);
            B(countB)=count;
            B(countB+1)=prev;
            countB=countB+2;
            count=1;
        end
    end
prev=A(i);
end

%code to deal with last value not taken care of in loop
if count==1
    size_B=size_B+fprintf(fid2,'%g\n',prev);
    B(countB)=prev;
else
    size_B=size_B+fprintf(fid2,'*%g\n%d\n',count,prev); %to print final
values
     B(countB)=count;
     B(countB+1)=prev;
end

%size of comparison file
size_A=fprintf(fid3,'%g\r\n',A);

%scale sizes from bytes to kilobytes
size_A=size_A/1000;
size_B=size_B/1000;

%calculate compression ratio
compression_ratio=size_B/size_A*100;

%display results of compression
```

```
fprintf('\nRun-Length Compression Results:\n\n')
fprintf('Original File Size = %3.1fKb\n',size_A)
fprintf('Compressed File Size = %3.1fKb\n',size_B)
fprintf('Compression Ratio = %2.1f\n',compression_ratio)

%display results graphically
figure('Name','File Comparison');subplot(2,1,1);plot(A(1:500));
axis([0,500,min(A),max(A)]);
title(['Original File L=',num2str(length(A))]);
subplot(2,1,2);plot(B(1:500));
axis([0,500,(min(B)-5),(max(B)+5)]);
title(['Compressed File L=',num2str(length(B))]);

%close files
fclose(fid);
fclose(fid2);
fclose(fid3);
```

A3. Matlab Implementation of Delta-Value Encoding

A3.1 Delta Value Compression Algorithm

```
%Delta Value Compression
clear all;close all;

%open file of sample data
fid=fopen('BP_Sample5.txt');

%open file to write compressed data to
fid2=fopen('dvcompressed.txt','w');

%open file to write original data to compare to file size of compressed
%data
fid3=fopen('dvcompare.txt','w');
[A, length1]=fscanf(fid,'%f');
```

```
%%encoding algorithm
prev=0;
size_B=0;
count=1;
for i=1:length1
    if i==1
        size_B=size_B+fprintf(fid2,'%g\r\n',A(i));
        prev=A(i);
        B(count)=A(i);
        count=count+1;
    else
        delta=prev-A(i);
        size_B=size_B+fprintf(fid2,'%g\r\n',delta);
        prev=A(i);
        B(count)=delta;
        count=count+1;
    end
end

%size of comparison file
size_A=fprintf(fid3,'%g\r\n',A);

%scale sizes from bytes to kilobytes
size_A=size_A/1000;
size_B=size_B/1000;

%calculate compression ratio
compression_ratio=size_B/size_A*100;

%display results of compression
fprintf('\nDelta Value Compression Results:\n\n')
fprintf('Original File Size = %3.1fKb\n',size_A)
fprintf('Compressed File Size = %3.1fKb\n',size_B)
fprintf('Compression Ratio = %2.1f\n',compression_ratio)

%display results graphically
```

```
figure('Name','File Compression Comparison');subplot(2,1,1);plot(A(1:500));
axis([0,500,min(A),max(A)]);
title('Original File');
subplot(2,1,2);plot(B(1:500));
axis([0,500,min(B),max(B)]);
title('Compressed File');

%close files
fclose(fid);
fclose(fid2);
fclose(fid3);

%code to extract compressed file, and compare to original - if equal,
%demonstrates accuracy of compression ratio
if(dv_extract(fopen('dvcompressed.txt')))
    fprintf('\nExtraction Results:\nExtracted file is equal to original\n');
end
```

A3.2 Delta Value Extraction Algorithm

```
%Matlab function to extract compressed data from delta value compression,
%and compare to original file to see if the two are equal
%Inputs: file ID of file to be extracted
%Outputs: Boolean value result = 1 if extracted file is equal to original
%                              = 0 if extracted file does not equal
%                              original

function result = dv_extract(fid)

%open file to write to for extracted data
fid2=fopen('dv_extracted.txt','w');

[A, length1]=fscanf(fid,'%f');
fprintf(fid2,'%6g\r\n',A(1));
val=A(1);
for i=2:length1
```

```
    val=val-A(i);
    if abs(val)<0.005
        fprintf(fid2,'0\r\n');
    else
        fprintf(fid2,'%g\r\n',val);
    end
end

fid3=fopen('dvcompare.txt');
B=fscanf(fid3,'%f');
fclose(fid2);
fid2=fopen('dv_extracted.txt');
C=fscanf(fid2,'%f');

%returns 1 if extracted file is equal to original, and 0 if not
result=isequal(B,C);

%graphically compare extracted and original files
figure('Name','File Extraction Comparison');subplot(2,1,1);plot(B(1:500));
axis([0,500,min(B),max(B)]);
title('Original File');
subplot(2,1,2);plot(B(1:500));
axis([0,500,min(C),max(C)]);
title('Extracted File');

%close files
fclose(fid);
fclose(fid2);
fclose(fid3);
```

A4. Matlab Sample Generator

A4.1 Sample Generator Code

```
%sample_generator.m
```

```
%Creates a set of sample blood pressure observation data that can be used to
demonstrate
%functionality of compression algorithms
clc; clear all; close all;

%set length of sample data
SampleSize = 10000;

%generates gaussian random variable for sample
meanBP = 120;
stdDevBP = 0.5;

sampleData = meanBP + stdDevBP.*randn(SampleSize,1); %mean of 120 with
standard dev. of 0.5

% open the file with write permission
fid = fopen('exp.txt', 'w');
fid2=fopen('BP_Sample5.txt','w');

%print sample data to file
fprintf(fid2, '%3.0f\n', sampleData);

%close files
fclose(fid);
fclose(fid2);
```

A4.2 Excerpt of Sample File Created with sample_generator.m

BP_Sample1.txt

```
120
119
119
120
121
120
119
119
120
```

```
120
121
120
121
119
119
120
120
120
120
120
119
120
120
120
119
121
120
120
121
120
120
121
120
119
120
120
120
120
```

A5. Matlab Implementation of Combined Algorithm

```
%Difference Value Compression then Run-Length Encoding
clear all;close all;
fid=fopen('BP_Sample1.txt');
fid2=fopen('dvcompressed.txt','w');
fid3=fopen('dvcompare.txt','w');
[A, length1]=fscanf(fid,'%f');

%%DV encoding algorithm
prev=0;
size_B=0;
count=1;
for i=1:length1
    if i==1
        size_B=size_B+fprintf(fid2,'%g\r\n',A(i));
        prev=A(i);
```

```
        B(count)=A(i);
        count=count+1;
    else
        delta=prev-A(i);
        size_B=size_B+fprintf(fid2,'%g\r\n',delta);
        prev=A(i);
        B(count)=delta;
        count=count+1;
    end
end

size_A=fprintf(fid3,'%g\r\n',A);
size_A=size_A/1000;
size_B=size_B/1000;
orig_size=size_A;
compression_ratio=size_B/size_A*100;
fprintf('\nDelta Value Compression Results:\n\n')
fprintf('Original File Size = %3.1fKb\n',size_A)
fprintf('Compressed File Size = %3.1fKb\n',size_B)
fprintf('Compression Ratio = %2.1f\n',compression_ratio)

figure('Name','File Comparison');subplot(3,1,1);plot(A(1:250));
axis([0,250,0,130]);
title(['Original File L=',num2str(length(A))]);
subplot(3,1,2);plot(B(1:250));
axis([0,250,-5,130]);
title(['Compressed File L=',num2str(length(B))]);

fclose(fid);
fclose(fid2);
fclose(fid3);

%RLE Compression
clear A B;
fid=fopen('dvcompressed.txt');
fid2=fopen('rle_dvcompressed.txt','w');
fid3=fopen('rle_dvcompare.txt','w');
```

```
[A, length1]=fscanf(fid,'%f');
%%encoding algorithm
count=0;
size_B=0;
countB=1;
for i=1:length1
    if count==0
        prev=A(i);
        count=1;
    else
        if prev==A(i)
            count=count+1;
        elseif count==1
            size_B=size_B+fprintf(fid2,'%g\n',prev);
            count=1;
            B(countB)=prev;
            countB=countB+1;
        else
            size_B=size_B+fprintf(fid2,'*%g\n%g\n',count,prev);
            B(countB)=count;
            B(countB+1)=prev;
            countB=countB+2;
            count=1;
        end
    end
prev=A(i);
end

if count==1
    size_B=size_B+fprintf(fid2,'%g\n',prev);
    B(countB)=prev;
else
    size_B=size_B+fprintf(fid2,'*%g\n%d\n',count,prev); %to print final
values
     B(countB)=count;
     B(countB+1)=prev;
end
```

```
size_A=fprintf(fid3,'%g\r\n',A);

size_A=size_A/1000;
size_B=size_B/1000;
compression_ratio=size_B/size_A*100;
fprintf('\nRun-Length Compression Results:\n\n')
fprintf('Original File Size = %3.1fKb\n',size_A)
fprintf('Compressed File Size = %3.1fKb\n',size_B)
fprintf('Compression Ratio = %2.1f\n',compression_ratio)

subplot(3,1,3);plot(B(1:250));
axis([0,250,-5,130]);
title(['RLE then DV Compressed File L=',num2str(length(B))]);

total_compression=size_B/orig_size*100;

fprintf('\nOverall Compression Results:\n\n')
fprintf('Original File Size = %3.1fKb\n',orig_size)
fprintf('Compressed File Size = %3.1fKb\n',size_B)
fprintf('Compression Ratio = %2.1f\n',total_compression)



fclose(fid);
fclose(fid2);
fclose(fid3);
```

References

[1] G. Weber, "A Comparison of Single Lead ECG Data Compression Techniques," Harvard University. [Online]. Available: http://www.hcs.harvard.edu/~weber/HomePage/Papers/ECGCompression/

[2] R. Fletcher et al, "iCalm: Wearable Sensor and Network Architecture for Wirelessly Communicating and Logging Autonomic Activity," *IEEE Transactions on Information Technology in Biomedicine,* Vol. PP, Issue 99, p 1, Jan. 8, 2010. [Online]. Available: http://ieeexplore.ieee.org

[3] A. Pantelopoulos, N. Bourbakis, "Prognosis – A Wearable Health Monitoring System for People at Risk: Methodology and Modeling," *IEEE Transactions on Information Technology in Biomedicine,* Vol. PP, Issue 99, p 1-1, Jan. 29, 2010. [Online]. Available: http://ieeexplore.ieee.org

[4] "ZigBee White Papers," *ZigBee Alliance,* 2010. [Online]. Available: http://www.zigbee.org/LearnMore/WhitePapers/tabid/257/Default.aspx

[5] "White Paper W580," *Sony Ericsson*, Oct. 28, 2007. [Online]. Available: http://developer.sonyericsson.com/wportal/devworld/downloads/download/1.708243?cc=gb&lc=en

[6] "Definition of Arrhythmia," *MedicineNet.com,* May 2000. [Online]. Available: http://www.medterms.com/script/main/art.asp?articlekey=2328

[7] D. A. Lelewer and D. S. Hirschberg, "Data Compression," *ACM Computing Surveys*, vol. 19, no. 3, p. 261-296, 1987. [Online]. Available: http://www.ics.uci.edu/~dan/pubs/DataCompression.html

[8] J. Kennedy, "Huffman Coding," *Santa Monica College*, Nov. 22, 2009. [Online]. Available: http://homepage.smc.edu/kennedy_john/HUFFMAN.PDF

[9] R. Seeck, "Data Compression," *binaryessence.com*. [Online]. Available: http://www.binaryessence.com/

[10] R. R. Britt, "Era Ends: Western Union Stops Sending Telegrams," *Live Science,* Jan. 31, 2006. [Online]. Available: http://www.livescience.com/technology/060131_western_union.html

[11] "Morse Code," *Hila Outdoor Center and Educational Resources*. [Online]. Available: http://hila.webcentre.ca/projects/morse_code/

[12] "Letter Frequency in the English Language," *Alternative Method Research*. [Online]. Available: http://letterfrequency.org/#english-language-letter-frequency

[13] "Letter Frequencies," *The Black Chamber*. [Online]. Available: http://www.simonsingh.net/The_Black_Chamber/frequencyanalysis.html

[14] S. W. Smith, "The Scientist and Engineer's Guide to Digital Signal Processing," *dspguide*.com, 1997. [Online]. Available: http://www.dspguide.com/ch27/4.htm

Vitae

NAME:	Kirsten Zernask-Cebek
PLACE OF BIRTH:	Lindsay, Ontario
YEAR OF BIRTH:	1988
SECONDARY EDUCATION:	Lindsay Collegiate and Vocational Institute (2001-2005)
HONOURS and AWARDS:	McMaster Honour Award 2005

Kirsten Zernask-Cebek is currently completing level IV of Electrical and Biomedical Engineering at McMaster University in Hamilton, Ontario.