Real-time Electrocardiogram Monitoring

Final Report

Nicholas Clark, Edward Sandor, and Calvin Walden

Advised By: Drs. In Soo Ahn and Yufeng Lu

Department of Electrical and Computer Engineering
Bradley University, Peoria IL
May 10, 2017

BRADLEY University
Department of Electrical
and Computer Engineering

Abstract

An arrhythmia is an irregular heartbeat that occurs when the electrical signals controlling the heart's muscular contractions become malformed. Patients who suffer from these symptoms are often given a Holter monitor to wear, which records electrocardiogram (ECG) data. During a subsequent healthcare provider visit, the patient's ECG data is analyzed, and if an arrhythmia was successfully recorded, the underlying condition can be diagnosed.

This project aimed to develop a wearable medical device for real-time arrhythmia detection, which acquires ECG data through a three-lead ECG sensor. It performs ECG signal processing and immediately alerts the patient's health care provider of an arrhythmia via wireless messaging.

At the current stage of this project, a common form of arrhythmia known as premature ventricular contractions (PVCs) are identified using the Pan-Tompkins and the wavelet-based Template-Matching algorithms. When three or more consecutive PVCs are detected, the device sends urgent report email to a patient’s health care provider. In the experimental study, the design has been successfully validated using benchmark records from the MIT-BIH arrhythmia database.

A low-cost digital signal processor evaluation kit, the Texas Instruments TMS320C5515 eZdsp USB stick, and an embedded Linux system, the Raspberry Pi 3 Model B, were chosen to be the hardware platform for this project. This study suggested a viable, low-complexity solution for real-time heart monitoring and arrhythmia detection.

Table of Contents

I. Introduction

Arrhythmias are irregular heartbeats that occur when the electrical signals controlling the heart's muscular contractions become malformed[1]. While these occur occasionally in healthy people, frequent occurrences can be a symptom of heart disease. The most common form of arrhythmia is premature ventricular contraction (PVC)[2]. When experienced in succession, PVCs cause a patient's heart to fail to circulate the necessary volume of blood through the body. Figure 1 shows an example of ECG data with two PVC spikes.

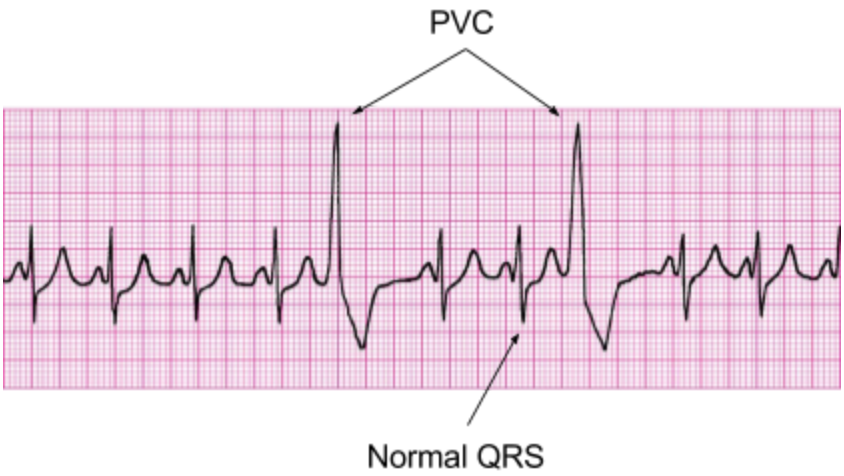

Figure 1: ECG data with two PVC spikes[3]

Holter monitors allow care providers to use electrocardiograms (ECGs) to monitor a patient's heartbeat and diagnose irregularities[4]. However, with present technology, the ECG data must be analyzed after it is acquired, and arrhythmias cannot be detected as they are occurring[5].

Previously, the project *Real-time Heart Monitoring and ECG Signal Processing*[6] addressed the need for real-time ECG signal processing, and successfully implemented a PVC detection algorithm. However, the system was not equipped with sensors to acquire ECG data real-time. It can only test off-line benchmark ECG data from the MIT-BIH arrhythmia database. While the final product could process ECG data directly acquired from a sensor in real-time, it was only used to test pre-recorded, benchmark ECG data from the MIT-BIH arrhythmia database.

This project aimed to implement a system that can be realized as a standalone medical device that could be comfortably worn by an outpatient. Taking this project closer to this goal involves re-evaluating the implementation of the PVC detection algorithms and the choice of embedded platform for operability, connectivity, and battery life.

A. Objectives

The current standard of care used to detect arrhythmias is the Holter monitor. It is a portable ECG device worn by a patient outside the hospital. It records patient ECG data for 24 to 48 hours, after which an external device must copy and process the data, which requires the patient to return to the care provider's office. This device cannot notify a patient's care provider of cardiac events, because it does not process and detect arrhythmias in real-time.

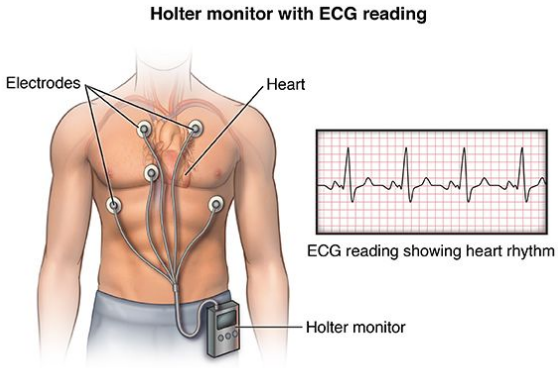

Figure 2: Example of a Holter monitor[7]

Because of the shortfalls of the Holter monitor, this project could improve upon the standard of care. To do this, three primary objectives were identified:

- Develop a portable, mobile device with ECG sensors.
- Implement an embedded system with ECG algorithms to monitor ECG signals in real-time.
- Wirelessly notify a patient's care provider of PVC.

B. Constraints

Based on the objectives, a number of design constraints were also identified and considered throughout the project. Because of the size constraints of the device, embedded hardware with limited processing power and memory is the optimal choice. Wireless connectivity must be reliable and, for the most part, constant to ensure that communication between the device and a care provider is possible. Additionally, battery life must be comparable to that of current Holter monitors to ensure operability throughout the patient's day.

II. Related Work

In 2016, the *Real-time Heart Monitoring and ECG Signal Processing*[6] project was completed by Bamarouf, Crandell, and Tsuyuki. This project looked into the application of detecting arrhythmias in real-time. It aimed to improve the Holter monitor by implementing the Pan-Tompkins and Template Matching algorithms on the low-power microcontroller. The portable and mobile system demonstrated that it could successfully transmit an alert message wirelessly in the event of an arrhythmia.

A. System Design

The system was centered around the Texas Instruments TI CC3200 Internet of Things-enabled microcontroller development platform, shown in Figure 3. Due to the limited memory and computing power in the chosen platform, some modifications to the ECG signal processing algorithms were made. This was due in part to the combined processor load of the real-time signal processing, wireless communication, and data logging on the device.

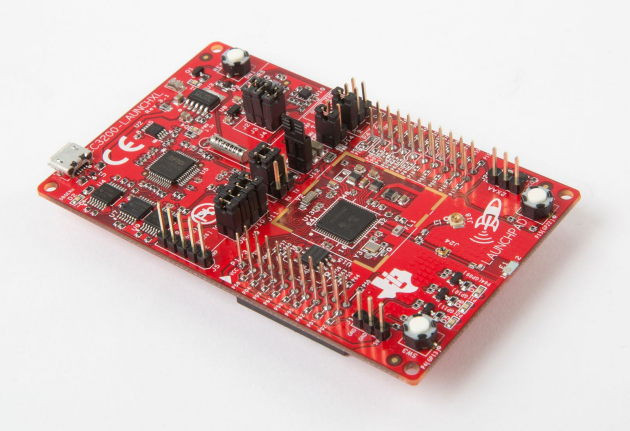

Figure 3: TI CC3200 IoT enabled MCU[8]

B. Results and Analysis

The previous project initially was planned to design a complete system to process ECG data acquired in real-time from sensors. Due to the complexity of system and the unknowns of exploring ECG signal processing algorithms, a sensor interface was not pursued. Benchmark ECG data demonstrating various arrhythmias was used instead. The system successfully acquired and processed ECG data in real-time, and wirelessly transmitted instances of ventricular tachycardia, which are instances of three or more consecutive PVC. Figure 4 shows the wirelessly transmitted plot.

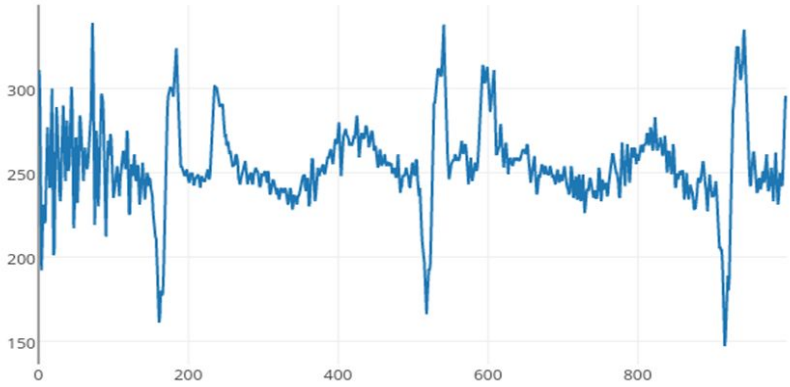

Figure 4: Wirelessly transmitted plot of VT via Plotly[6]

For the benchmark ECG data, the MIT-BIH arrhythmia database was used. This database offers hundreds of ECG samples to test and compare ECG algorithms. Some benchmark tests were extremely successful at not only detecting the QRS peaks, but also the PVC occurrences. The results from these tests are shown in Table 1.

Table 1: *Real-time Heart Monitoring and ECG Signal Processing* Benchmark Results[6]

Record	QRS Sensitivity	QRS Positive Predictivity	PVC Sensitivity	PVC Positive Predictivity
116	0.988	0.999	0.972	0.954
119	1.000	1.000	1.000	1.000
201	0.973	0.979	0.864	0.665
203	0.991	0.982	0.854	0.548
205	0.998	1.000	0.958	0.986
208	0.938	0.994	0.826	0.972

III. Implementation

A. System Block Diagram

The system block diagram of the *Real-Time Electrocardiogram Monitoring* device is shown in Figure 5. The whole design is centric around the system controller, which is responsible for managing and logging data received from the DSP. If the DSP's algorithm detects a PVC, it sends an urgent report message to the configured care provider email address. If configured to do so, it can also send periodic ECG report messages to a care provider or patient.

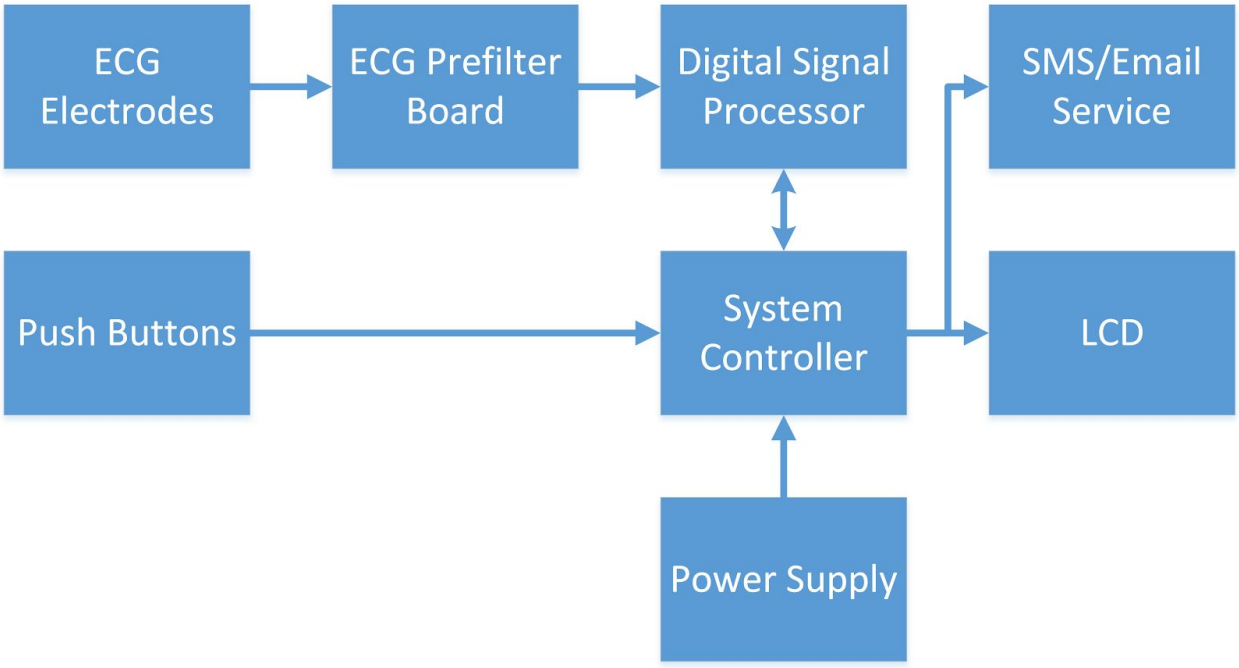

Figure 5: System block diagram

B. Subsystems

Table 2 lists the device's subsystems and the functionality of each of them.

Table 2: Subsystems

Subsystem	**Description**
Digital Signal Processor	Manages data acquisition from ECG sensors and runs arrhythmia detection algorithm. ECG data, analysis results, and instructions are communicated via UART.
System controller	Manages user interface, logs data, sends instructions to DSP, and sends wireless notifications using an SMS/email service. Only component connected directly to the power supply. On-board voltage regulators distribute power to the other subsystems.
ECG Electrodes	Attaches to a patient's chest to sample electrical signal information from the heart using foam adhesive pads.
ECG Prefilter	Amplifies and filters raw signals from ECG electrodes to provide a signal which can be acquired by the DSP's ADC.
LCD	Provides patient with quick information such as the status of the system and heart rate.
Push Buttons	Allows the patient to control the system to perform operations such as starting and stopping monitoring or shutdown of the device.
SMS/Email Service	Communicates ECG analysis to patient or caregiver using an email service.
Power Supply	Selects between a battery or AC power supply and provides each subsystem with clean and regulated power.

1. Digital Signal Processor

The core of the *Real-time Electrocardiogram Monitoring* system is a digital signal processor (DSP), which is responsible for ECG data acquisition and processing to detect arrhythmias in real time. The DSP communicates via the Universal Asynchronous Receiver/Transmitter (UART) protocol to transfer raw data and signal analysis to the system controller for logging, as well as to receive benchmark data and instructions to start or stop processing.

Project Hardware

The TI C5515 DSP was chosen for this project. It is a low-power, 16-bit fixed-point processor with a sufficient amount of built-in RAM for the project's arrhythmia detection algorithm. A TMS320C5515 eZdsp prototyping board, shown in Figure 6, is used to acquire and process ECG data.

The TMS320C5515 eZdsp specifications are as follows:

- 120-MHz 16-bit fixed-point DSP
- 64 KB of DARAM
- 256 KB of SARAM
- 128 KB of ROM
- Up to 16 MB external RAM
- 3 General purpose timers
- 4-input 10-bit ADC
- UART, SPI, and I^2C serial communication

Figure 6: TMS320C5515 eZdsp prototyping board[9]

DSP Software

The software running on the DSP consists of three parts: data acquisition, arrhythmia detection, and UART communication. The main software loop is responsible for running the arrhythmia detection algorithm, transmitting ECG data over UART, and parsing data received over UART. Compared with the operation of data acquisition from the sensor, these three operations are less time sensitive. Therefore, these three operations can be interrupted by higher-priority functions. The flowchart of the main loop is shown in Figure 7.

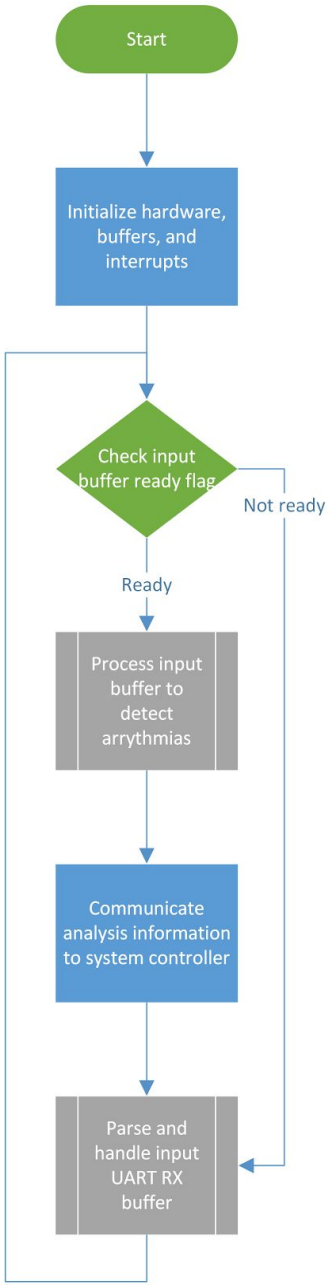

Figure 7: DSP functionality flowchart

To ensure a consistent and accurate sampling rate of 360 samples per second, acquisition is handled in the manner of interrupt. A hardware timer is used to trigger an interrupt service routine triggered every 2.78ms. The flowchart of timer interrupt routine is shown in Figure 8.

The on-board analog-to-digital converter (ADC) is used for data acquisition. If the DSP is configured to acquire ECG data from the sensors instead of using benchmark data, a hardware timer initializes the start of ADC conversion. When the conversion is complete, an interrupt is triggered, and the acquired ECG sample is saved to the system's input buffer. The flowchart of ADC interrupt is shown in Figure 9.

If the DSP is set to process benchmark data instead, the timer interrupt retrieves a single sample of ECG data from the UART receive buffer and saves it to the input buffer of system. When the input buffer is full, a flag is set to run the arrhythmia detection algorithm in the main loop.

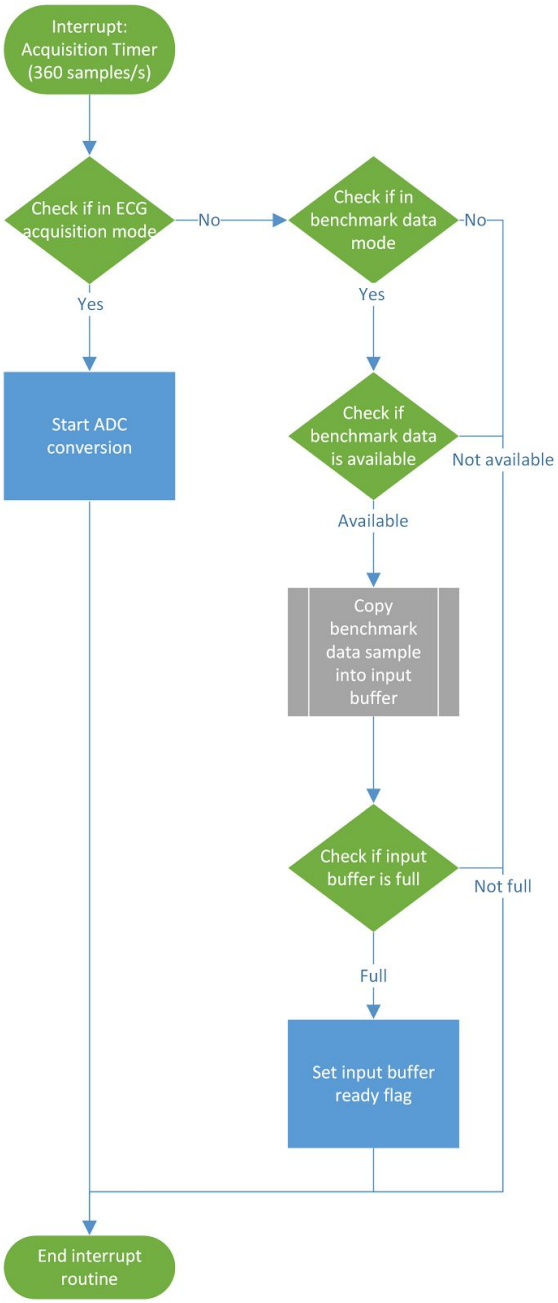

Figure 8: DSP Timer Interrupt Flowchart

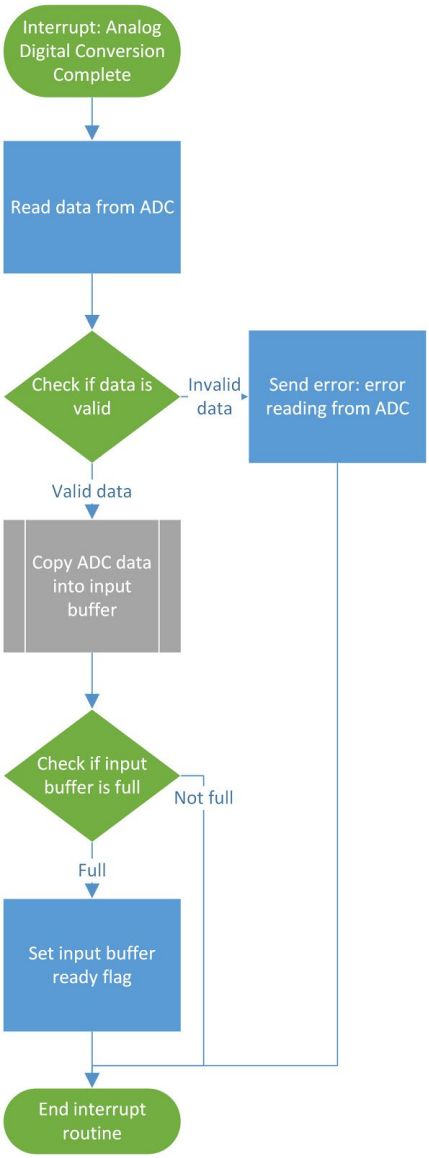

Figure 9: DSP ADC Interrupt Flowchart

Instructions can be sent to the DSP via UART. These instructions are designed to control the DSP operation, change data source, report its status, or perform a system reset. In addition, benchmark data can be sent to the DSP via UART to validate the arrhythmia detection algorithm. Whenever the hardware UART buffer is full, an interrupt is triggered to unload the data from the buffer. In the corresponding interrupt service routine, the data in the hardware buffer is unloaded to a larger software buffer, which is parsed in the main loop. The flowchart of UART interrupt service routine is shown in Figure 10.

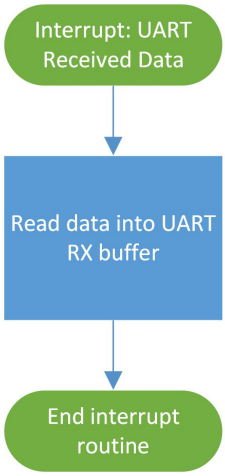

Figure 10: DSP UART Received Data Interrupt Flowchart

2. System Controller

Hardware

A Raspberry Pi 3 Model B, shown in Figure 11, is used as the system controller. It is a mobile and portable Linux-based platform with integrated wired and wireless networking, as well as a 40-pin GPIO expansion port. The operating system chosen for this project is Raspbian 8.0, a Debian Linux derivative. Its specifications are as follows:

- 1.2 GHz Quad-Core ARM Cortex CPU
- 1 GB LPDDR2 RAM
- SD card slot as boot media and local storage
- 27 GPIO pins, including UART and I^2C
- 10/100 BaseT Ethernet wired LAN
- 802.11b/g/n wireless LAN

Figure 11: Raspberry Pi 3 Model B[10]

Software Design

All of the system controller software components are written in C and are built using the GNU toolchain. These pieces of software make use of several open-source libraries and projects to ease the development workload. These include:

- `libconfig`, a library that handles the storage and parsing of plain text configuration files. The Real-Time ECG software shares the configuration file `/etc/rtecg/rtecg.conf` to store information such as patient details, patient and care provider email addresses, and paths for generated files and external binaries.
- `wiringPi`, a library that allows direct control of the 27 GPIO pins of the Raspberry Pi, including UART and I^2C lines.
- `gnuplot`, a command-line graphing utility that can generate intricate and detailed plots. The Communication Daemon uses this to generate plots of ECG data to attach to report emails.

Acquisition Daemon

Data sent over UART from the DSP is first received by the Acquisition Daemon, `rtecg_acquisition`. This program waits for serial data and accumulates several DSP buffers' worth in memory before writing it to a time-stamped file in `/var/rtecg/logs/`. Its functionality is visualized in Figure 12. In the case that an arrhythmia is detected and flagged by the DSP, it will also signal the Communication Daemon to report this to the patient or care provider, depending on configuration.

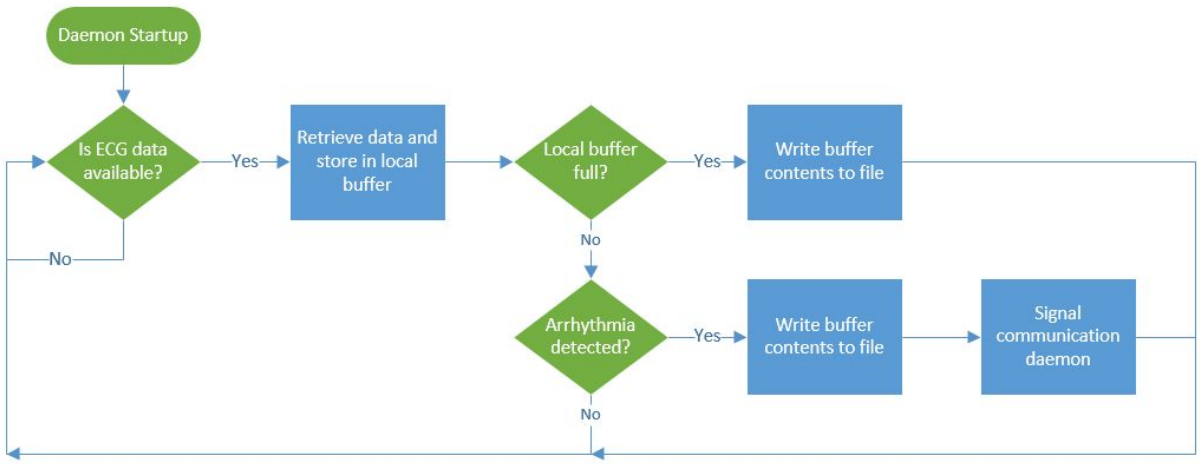

Figure 12: Acquisition Daemon functionality flowchart

Communication Daemon

The second stage of ECG data management in the system controller is performed by the Communication Daemon, `rtecg_comm`. Its functionality is visualized in Figure 13.

Upon receiving a signal from `rtecg_acquisition` or another source, this program finds the most recently created log file in `/var/rtecg/logs/`, and pipes its contents into `gnuplot` to generate a plot as a PNG image file. It then composes a timestamped and descriptive message, and sends it using `mail` to the email addresses specified in the configuration file, with both the complete log file and newly generated plot attached.

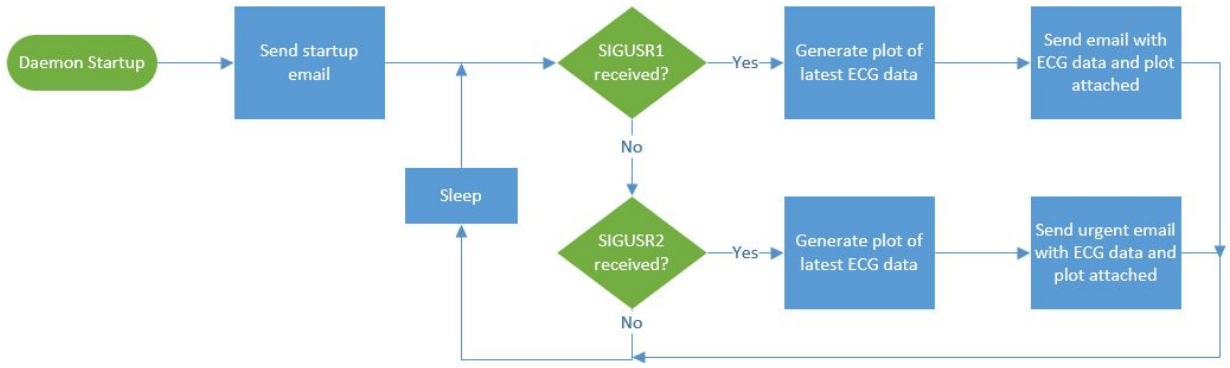

Figure 13: Communication Daemon functionality flowchart

Upon startup, this daemon sets up a POSIX signal handler for both of the user-defined signals, SIGUSR1 and SIGUSR2. While these both trigger the same events, SIGUSR2 is reserved for the event where the DSP's algorithm detected an arrhythmia and will generate an urgent message, whereas SIGUSR1 is intended to be used to trigger a routine ECG monitor report that a care provider or patient may desire. An example of the former is shown in Figure 29.

Additionally, at system startup, the communication daemon sends a message to the care provider email address, containing system information, including network interface IP addresses and local storage free space. An example of this is shown in Figure 30.

3. Electrocardiogram Sensors and Prefilter Board

ECG Electrodes and Hardware

A three-channel ECG was determined to be sufficient for the ECG data acquisition portion of this project. To acquire an ECG signal, the patient wears three foam electrodes that are secured to their chest with an adhesive. During the development of this project, 3M Red Dot™ electrodes

were used. These electrodes are then connected to the device using a three-conductor cable with color-coded electrode connectors on one end and a 3.5mm TRS connector on the other.

An AD8232 ECG prefilter board, shown in Figure 15, is used to amplify and filter the raw ECG signal from the electrodes in order to generate a clean analog signal which may be acquired by the ADC of the DSP. The analog signal, driven by an op-amp, ranges from 0-3.3V, but the DSP's ADC has a maximum input voltage of 1V. In order to interface the prefilter with the DSP, a voltage divider is used.

In addition to an analog ECG signal, the prefilter board also has digital outputs that indicate if the ECG leads are connected to electrodes on the patient's chest. This information can be used to pause processing on the DSP until the electrodes are reconnected and alert the patient of an error. Another digital pin causes the prefilter board to be shut down to save power.

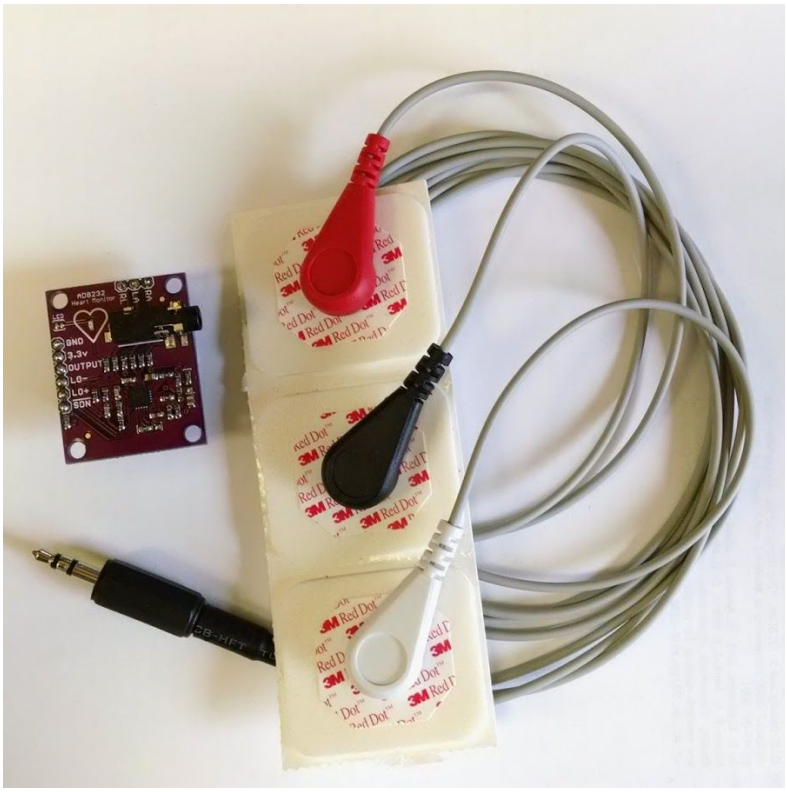

Figure 14: AD8232 Prefilter Board, ECG leads, and electrodes

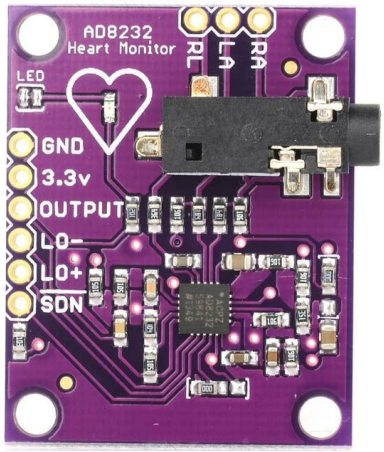

Figure 15: Close-up of the AD8232 prefilter board[11]

During the development of this project, each group member experimented with acquiring ECG data from their own heart to test the DSP algorithm and data acquisition software of the device. Electrodes were placed on their chests using Figure 16 as a reference.

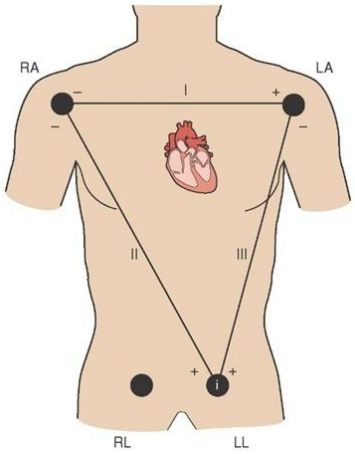

Figure 16: Reference figure for 3-lead ECG electrode placement[12]

4. User Interface

In order to provide the patient with simple information and configuration options, a user interface (UI) was developed. The primary component of the UI is the liquid crystal display (LCD) and button interface. A web server running on the Raspberry Pi also allows for additional customization and configuration of the device.

Hardware

The Adafruit 16x2 Character LCD + Keypad for Raspberry Pi was the chosen display and pushbutton interface for this project. It offers the most flexibility by utilizing the I^2C and GPIO buses on the first 26 pins of the Raspberry Pi 3's GPIO header. The notable components of this package are the MCP23017 I^2C port expander chip and the HD44780 LCD Controller. Button press events are signaled to controlling software on the Raspberry Pi via its GPIO buses.

Figure 17: Adafruit 16x2 LCD and Keypad[13]

LCD Software

The LCD process, `rtecg_ui`, is a Linux daemon written in C using the Adafruit-RPi-LCD library that handles direct control of the LCD display and onboard keypad. The daemon waits for updates to specific file descriptors and button-press events, and updates local variables and LCD text as needed.

In order to update the LCD, commands must be sent using a specific packet protocol used by the Microchip MCP23017 I/O Expander. This component adds additional GPIO pins for the Hitachi HD44780 LCD controller over the Raspberry Pi's I^2C bus.

Upon startup, the LCD software initializes the GPIO pins that it uses and sets up its menu options. It then runs in a loop, looking for changes to file descriptors that point to network information or piped data from the webserver. If changes are detected, the menus are updated accordingly. It also tests for detected button presses and denounces them in software before handling them.

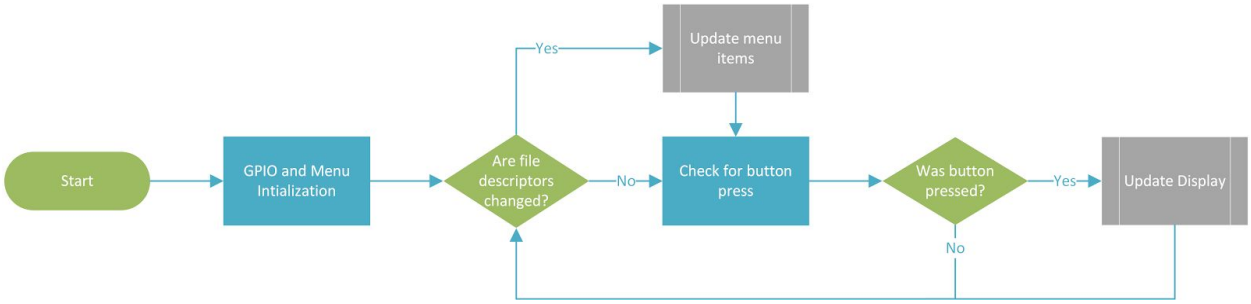

Figure 18: Flowchart of LCD process

Web Server

A web server was also implemented on the Raspberry Pi. For a responsive web interface, PHP5 is used alongside a common gateway interface (CGI). The web interface is served by Lighttpd, a lightweight, secure, and open source web server. This web interface allows the patient to access device information and configure it remotely.

5. Printed Circuit Board

To connect the Raspberry Pi 3, the DSP, and the ECG prefilter board, a printed circuit board using the Raspberry Pi "Hat" formfactor was designed and fabricated. Its complete schematic can be found in Appendix A. The final, fabricated circuit board can be seen in Figure 19.

The circuit board incorporates the 40-pin Raspberry Pi 3 header, the 60-pin TMS320C5515 eZdsp expansion edge connector, a header for the prefilter board, and the voltage divider used to convert the ECG signal to the range 0 to 1 volts. The DSP power bus and UART and I^2C signals are connected using jumpers. Additionally, the board has debug headers for the ECG signal, Raspberry Pi GPIO, and DSP GPIO signals to serve as accessible test points and to aid in future work.

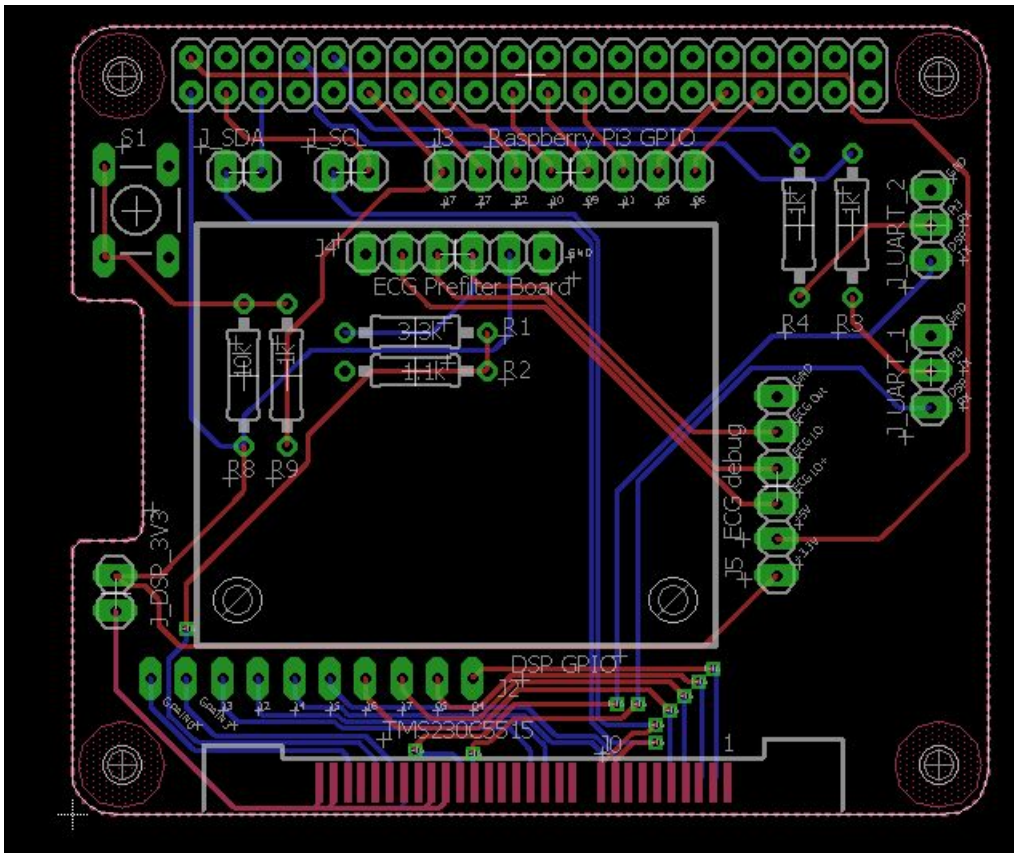

Figure 19: Layout of PCB to interface system components

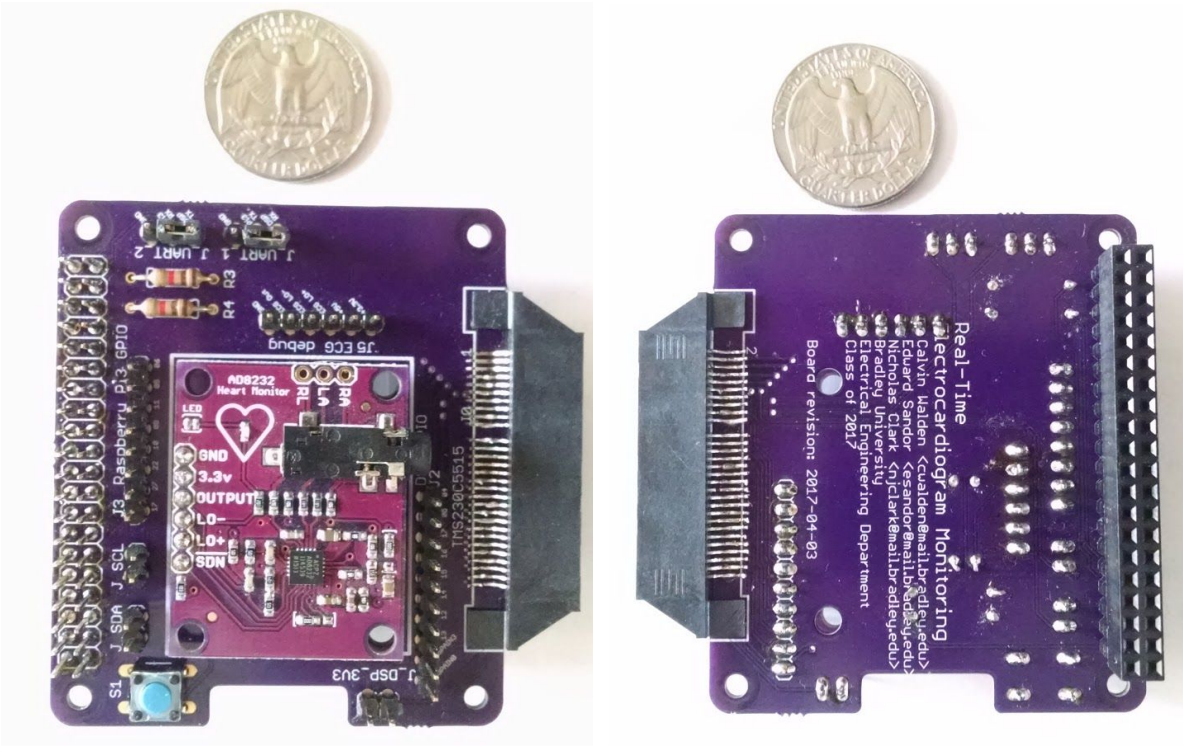

Figure 20: Top and bottom of fabricated circuit board

C. Arrhythmia Detection Algorithm

The arrhythmia detection algorithm used in this project uses C code written by Bamarouf, Crandell, and Tsuyuki for their *Real-time Heart Monitoring and ECG Signal Processing* project. Only the arrhythmia detection algorithm portion of their code is applicable to the system developed for this project. The algorithm works in two stages, first QRS peaks are detected then the locations are used with a template matching algorithm.

1. QRS Peak Detection

In order to identify most arrhythmias, some form of QRS peak detection is required. Two sets of research have identified the Pan-Tompkins (Hamilton-Tompkins) algorithm as the simplest and most efficient algorithm to use on embedded devices[14][15]. This algorithm cleans the raw ECG signal using a bandpass filter then emphasises the QRS signal using differentiation, squaring, and an moving average filter. The algorithm then applies a set of rules to set an adaptive threshold to detect QRS based on the bandpass filtered signal and the QRS emphasised signal.

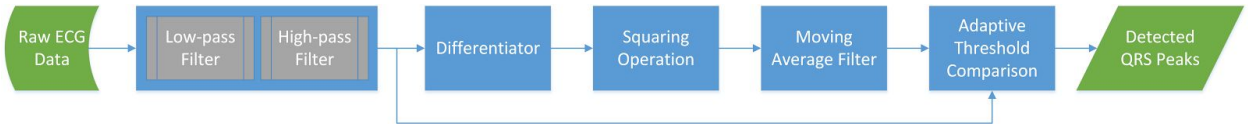

Figure 21: Flowchart of the Pan-Tompkins Algorithm

2. Template Matching

After QRS peaks have been detected, a template matching algorithm is used to determine if a beat is healthy or is an arrhythmia[15]. This algorithm performs a wavelet transform on the raw data. When first started, the algorithm creates a template for a healthy heartbeat for the particular patient based on the best of the first beats detected. When a QRS peak is detected by the Pan-Tompkins algorithm, the raw signal is correlated with the template and the beat is determined to be healthy based on a threshold.

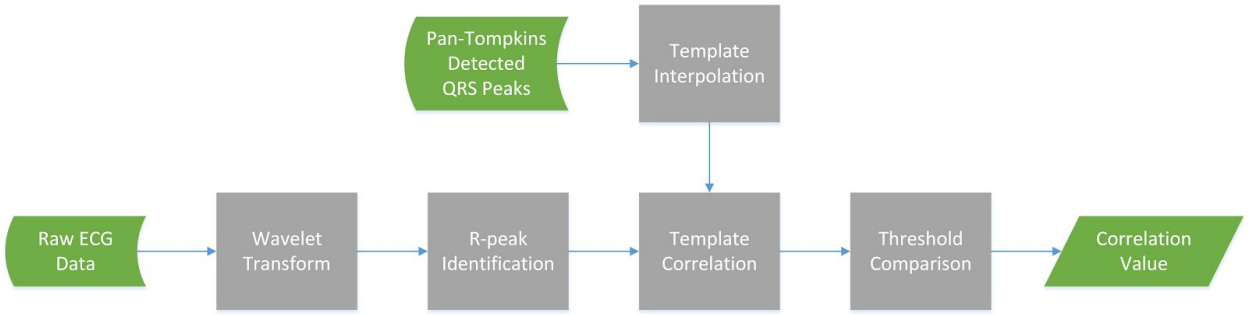

Figure 22: Flowchart of the Template Matching Algorithm

3. Conversion to Platform Independent Code

The complete arrhythmia detection algorithm was extracted from their project and made to be platform independent, and callable as a standalone C function. A few functions were rewritten to remove dependencies on floating point operations and libraries specific to their original platform. The advantage of being platform independent is the code can be recompiled for different platforms and therefore easily ported if different hardware is selected. Platform independence also allowed the code to be directly simulated using MATLAB by creating a MEX file; Bamarouf, Crandell, and Tsuyuki had a separate implementation that had to be maintained for simulation. Any change made to the platform independent version algorithm changes simultaneously on the DSP and in the MATLAB simulation reducing the chance of discrepancies between hardware and simulation results.

4. Implementation on the DSP

The code by Bamarouf, Crandell, and Tsuyuki did not fit on the C5515 DSP as it required a large amount of RAM. In order to make the code compatible, some of the larger buffers were converted to circular buffers and their sizes were reduced.

IV. Results and Discussion

A. Digital Signal Processor Performance

1. Acquisition

The first major responsibility of the DSP is to acquire ECG information from a patient. The acquisition setup is described in section III.A.3. The DSP successfully samples the ECG prefilter board's output with 10-bits of resolution at a rate of 360 samples per second. Figure 23 shows a section of acquired raw ECG data with a normalized magnitude. At this stage, a strong instrumental noise is pronounced in the acquired data.

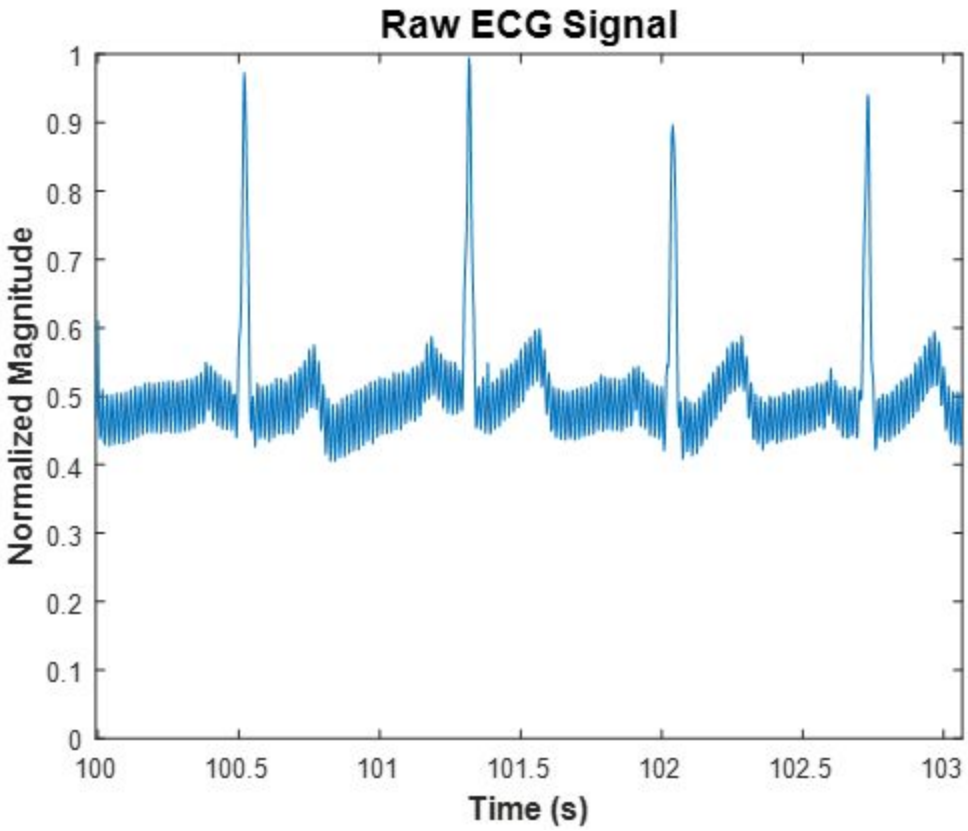

Figure 23: Raw ECG data acquired by DSP

2. Processing Time

The second major responsibility of the DSP is to process the ECG signal and communicate analysis data to the system controller in real-time to detect arrhythmias. The measurement of

processing time starts when an output GPIO pin on the DSP is set which launches the arrhythmia detection algorithm and the input buffer is accessed. The measurement of processing time ends when the GPIO pin is cleared upon completion of detection algorithm.

Figure 24 shows an oscilloscope screenshot of monitor the DSP's activity. The green trace indicates that the DSP is processing buffered ECG data. It takes 21 ms to process a 550 ms buffer of ECG data. Next, the red trace shows it takes 120 ms to send a 550 ms buffer of data from the DSP to the Raspberry Pi. Finally, the blue trace shows the ECG output from the prefilter board.

Figure 24: Oscilloscope monitoring of DSP activity. From top to bottom: pulse indicating arrhythmia detection processing time, DSP transmitting analysis data, raw ECG signal output from prefilter board

Our profiling results show that approximately 3.8% of the DSP's main loop is dedicated to arrhythmia detection and approximately 22% of the main loop is dedicated to communicating analysis. The DSP is idle approximately 74% of the time. It suggests that the system could be set in hibernate or power-saving mode to reduce the power consumption of the system. In addition, the idle time could be utilized to implement a more sophisticated detection algorithm on the DSP. Currently all data are transferred in the format of blocks. Multithreading or

asynchronous hardware operations could be used to further improve the system performance in terms of execution time.

B. Arrhythmia Detection Algorithm Performance

Performance of the arrhythmia detection algorithm has been evaluated using the same C code compiled for the DSP as well as compiled for MATLAB. Evaluation on the DSP primarily verifies execution on the embedding platform and real-time processing. Evaluation using MATLAB verifies accuracy and consistency of the detection algorithm. Results are shown to be comparable by observing the same output from both the DSP with MATLAB. Because the same code is compiled for both platforms, modifications to the codes are reflected in the output from both platforms.

1. Pan-Tompkins Algorithm

The Pan-Tompkins algorithm has been verified using both the DSP and MATLAB. Figure 25 shows the output from the DSP demonstrating bandpass prefiltering between 5 and 11 Hz for the Pan-Tompkins algorithm.

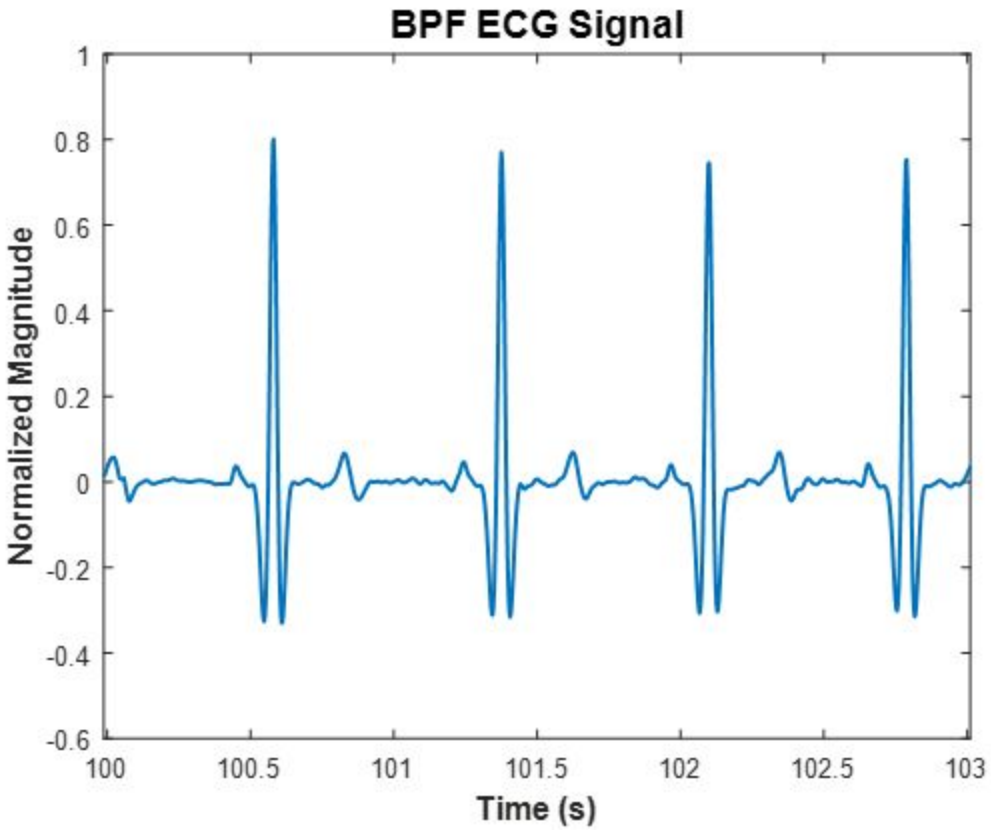

Figure 25: Bandpass filtered data output from DSP

After prefiltering, QRS complex and heart beats are marked by the Pan-Tompkins algorithms. Markings are used to identify heartbeats for the template matching algorithm. Figure 26 shows a bandpass filtered ECG signal with heartbeats marked. A similar output has been observed from the codes running on the DSP.

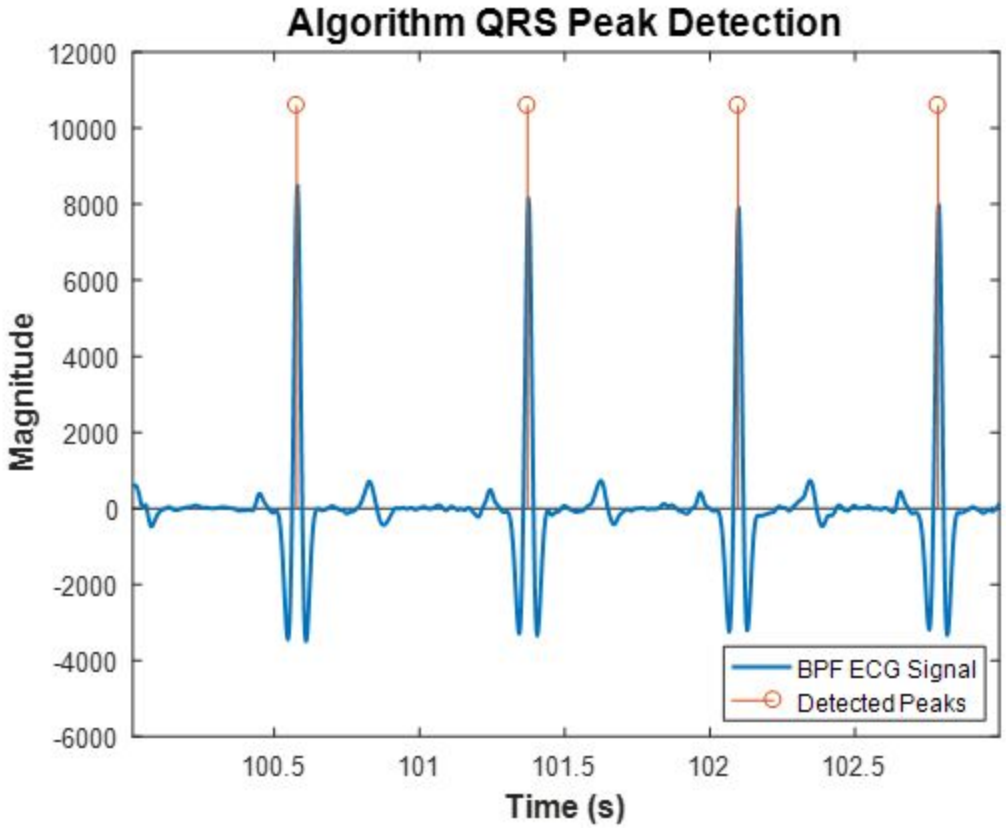

Figure 26: Detected QRS peaks by arrhythmia detection algorithm run using MATLAB

2. Template Matching Algorithm

The template matching algorithm generates two templates for a patient's normal heartbeat by monitoring average amplitude and duration during the first two minutes of operation. A beat closely matching the average parameters is used as a template for arrhythmia detection. Two templates, T1 and T2, are used. Template T1 is the QRS portion of a heartbeat and template T2 is the interval between R peaks of a heart beat. Figure 27 and Figure 28 are examples of obtained templates T1 and T2 through the MATLAB simulation of the algorithm.

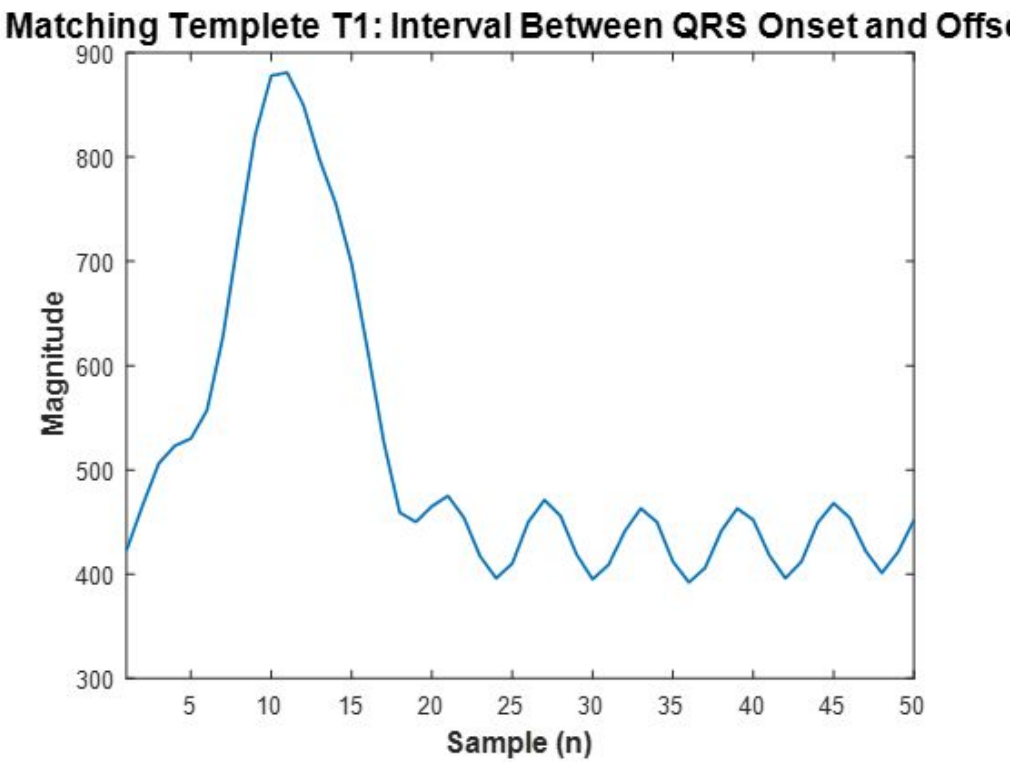

Figure 27: Template of QRS peak used by template matching algorithm

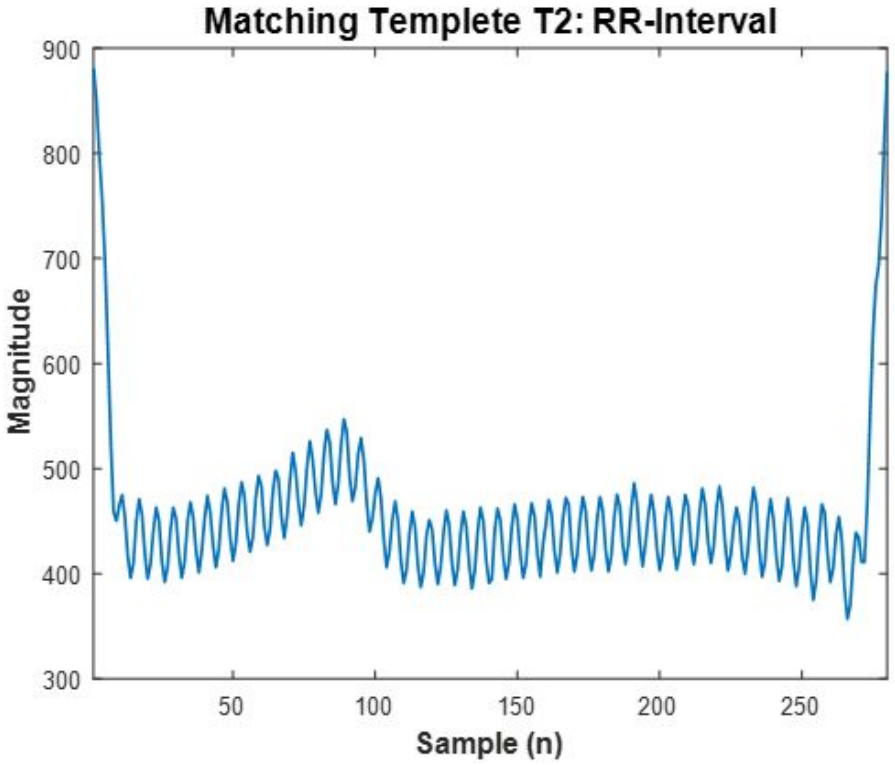

Figure 28: RR interval template used by template matching algorithm

3. Overall Algorithm Performance

The ECG signal processing algorithm on the DSP successfully prefilters data for the Pan-Tompkins algorithm, identifies heartbeats using the Pan-Tompkins algorithm, and generates template for the template matching algorithm. No major changes were made to the detection technique completed by the group from previous project [6]. There are incorrect identification of arrhythmias after compiling either the DSP or MATLAB version of the modified codes. It generates inaccurate correlation values in the final stage of the template matching algorithm which eventually results in incorrect identification of arrhythmias. Further analysis and refinement of codes is needed. A detailed evaluation of the overall algorithm performance can be seen in the reference *Real-time Heart Monitoring and ECG Signal Processing*[6].

Table 3: *Real-time Heart Monitoring and ECG Signal Processing* Benchmark Results[6]

Record	QRS Sensitivity	QRS Positive Predictivity	PVC Sensitivity	PVC Positive Predictivity
116	0.988	0.999	0.972	0.954
119	1.000	1.000	1.000	1.000
201	0.973	0.979	0.864	0.665
203	0.991	0.982	0.854	0.548
205	0.998	1.000	0.958	0.986
208	0.938	0.994	0.826	0.972

C. System Controller

1. ECG Monitor Report Messages

The ultimate responsibility of the system controller is to alert the patient or patient's health care provider in the event when an arrhythmia is detected in the ECG signal. Figure 29 shows a sample alert message, as seen on a care provider's smartphone. The message includes patient information, a timestamp, and attachments of a plot of the patient's ECG data, as well as the complete ECG data log.

Additionally, Figure 30 shows the message to a health care provider or device administrator upon system startup. It includes the name and ID of the device, its IP address, and the available space of the Raspberry Pi's SD card for ECG data storage.

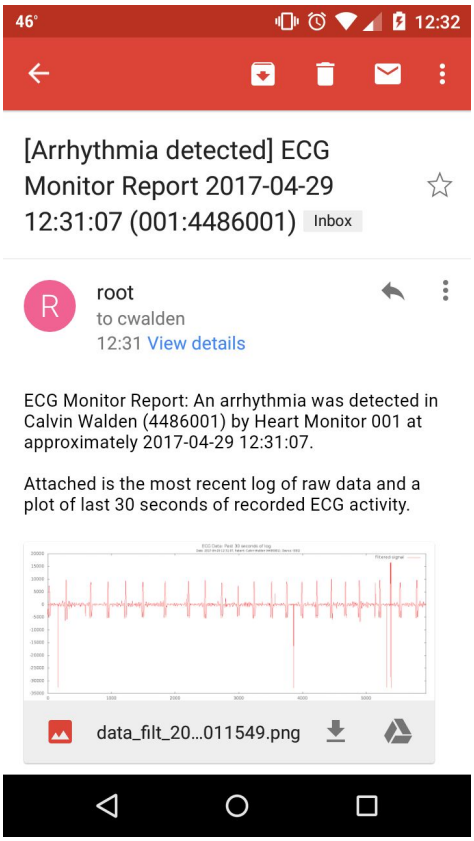

Figure 29: Screenshot of an ECG monitor alert email on a smartphone, indicating that the running system detected that its patient experienced an arrhythmia.

Figure 30: Communication Daemon startup notification email

2. Performance

One of the goals of this project is to improve upon the algorithm performance. Here system resource usage on this device is used as a metric to evaluate the performance.

With all of the *Real-Time Electrocardiogram Monitoring* software loaded and running, the one-minute load average, as reported by `uptime`, was consistently between 0.30 and 0.50. It can be interpreted as 7.5% to 12.5% total resource capacity, given the Raspberry Pi 3 being a quad-core system. The Acquisition Daemon is the most resource-intensive process of all of the project's software, using between 5% and 9% of CPU time. It could be attributed to frequent serial port polling in the daemon.

Memory usage, again measured with all of the project's software running, is consistently found to be about 20% of its capacity, which is about 200 MB of the total available 923 MB system memory.

The low resource usage figures suggest that in a future iteration of this project, improvements can be made either to harness the unused computing power to perform additional ECG information acquisition and processing, or to invoke power-saving modes to reduce power consumption and improve battery life.

D. User Interface

1. LCD

The LCD component of the user interface displays a number of pieces of important information across five menu options. These include the project title, patient name, device's IP address, LCD color options, and power options. The user can scroll through the menu items using the *up* and *down* buttons, and toggle between options on certain menu items using the *left* and *right* buttons. In addition, the *select* button can be used to select an option when applicable.

Currently, this LCD serves to convey important information quickly to the patient. The 32-character dual-line LCD restricts the information to be displayed. It also limits the amount of interactivity. Text-only information can be displayed, not graphic data. In a future iteration of this project, this component could be further developed to add more options to the menu, such as a display for heart rate and more configuration options. Or an app running on smartphone could be used for configuration.

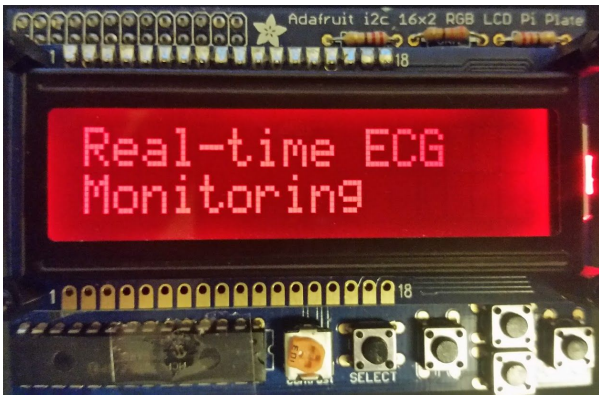

Figure 31: LCD displaying project title

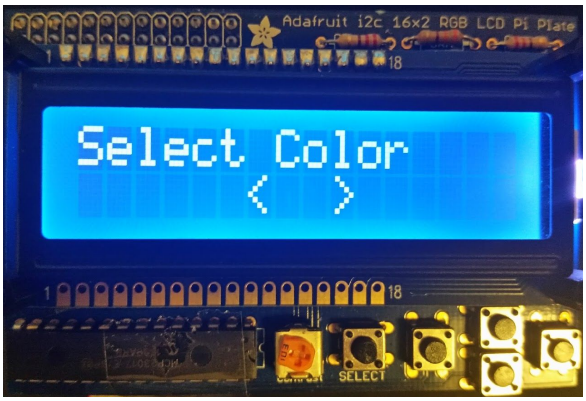

Figure 32: LCD displaying color menu option

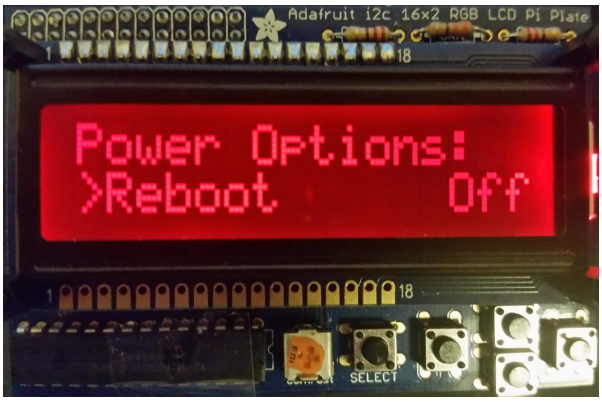

Figure 33: LCD displaying power options menu item

Figure 34: LCD in screensaver mode after 20 seconds of inactivity

2. Web Server

The web server allows the patient to access important patient and device information. In the patient information section, the patient name and identification number are displayed. In the user configuration section, the patient can see the email recipient information for the notification emails. In the device configuration section, the device name and identification number are listed. There is also a field to upload a new configuration file for remote configuration. There are also power options listed including poweroff and reboot. The network information shows the current IP address, as well as the gateway and subnet information. The final two sections include download links for logs and files for patient access.

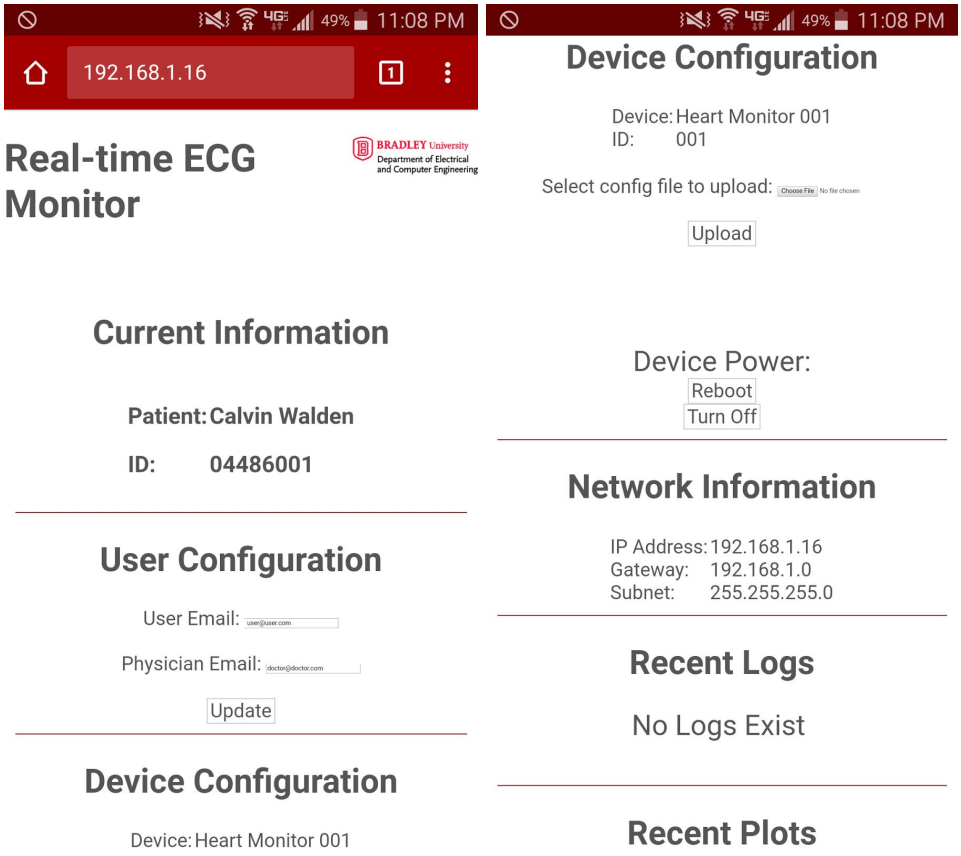

Figure 35: Web interface on a mobile phone

E. Power Consumption

An experiment has been performed to have a rough estimate of the average power consumed by the entire system. The power consumption is estimated by measuring the time required to discharge a Anker Astro E1 portable lithium polymer battery. Here the battery has a built-in voltage regulator that makes the battery pack usable as a 5 volt power source.

Capacity is estimated by measuring the time required to discharge the battery through a resistive load. An Arduino Nano, powered by a separate power supply, is used to measure the amount of time required to discharge the battery. The experiment was run two times with two different loads to estimate the battery's capacity. In the first experiment, the battery is discharged through a 10 ohm resistive load. In the second experiment, a 50 ohm resistive load is used to discharge the battery. The estimated capacity of battery is approximately 14.6 Wh. The fully-charged battery is used to run the system developed for this project. The system operates 8.7 hours under the battery power. The estimated system power consumption is approximately 1.7 W.

The Raspberry Pi 3 is reported to use approximately 1.3 W alone[16]. It can be seen that the Raspberry Pi 3 is a power-hungry unit in the whole system. An alternate embedded system with low power rating to replace the Raspberry Pi would greatly reduce the system power consumption.

V. Project Management

A. Division of Labor

Table 4 lists the division of labor for the project. In general, Ed Sanders is the primary contributor to DSP and arrhythmia detection algorithm development, Calvin Walden is the primary contributor to the system controller software, and Nick Clark is the primary contributor to the user interface of the system. The bolded name next to each task denotes the primary contributor.

Table 4: Tasks divided amongst group members

Task	**Contributor(s)**
DSP and algorithm evaluation	**Ed**
System Controller acquisition and communication software	**Calvin**
Printed circuit board design	**Calvin**
PCB review	Calvin, **Ed**, Nick
System Controller user interface software	**Nick**
Poster, presentation, and reports	Calvin, Ed, **Nick**
Documentation	**Calvin**, Ed
ECG electrode testing	Calvin, Ed, Nick

B. Project Schedule

During its development, this project followed its original, proposed scale reasonably well. While its goal was to complete all lab work in March of 2017, system testing and tuning continued into April of 2017. When we decided to do this, each part of the system was functional, but the team members felt that more refinement was necessary. As a result, some time originally reserved for written deliverables was reallocated for more lab time.

Table 5 shows the updated schedule, with reallocated tasks stuck out and amendments and added tasks in red.

Table 5: Updated Project Schedule

Week Of	Work To Be Completed
11/21/16	• MATLAB simulation of Pan-Tompkins algorithm
11/28/16	• Project Proposal Presentation (12/1) • Determine complexity of Pan-Tompkins algorithm
12/5/16	• Project Proposal (12/5) • Receive ordered parts
Winter Break	• Begin software development for embedded computer and DSP
1/16/17	• Implement initial Pan-Tompkins algorithm on choice platform • Begin interfacing board for embedded computer • Interface embedded computer with DSP
1/23/17	• Refine Pan-Tompkins algorithm, add Template Matching algorithm • Trial wireless communication and SMS
1/30/17	• Trial ECG data from MIT-BIH database • Refine wireless communication and SMS
2/6/17	• Refine Pan-Tompkins and Template Matching algorithms • Trial ECG data from MIT-BIH database • Begin embedded computer serial communication
2/13/17	• Begin real-time ECG testing • Add LCD and pushbutton interface to interfacing board • Begin UI development
2/20/17	• Refine Pan-Tompkins and Template Matching algorithms • Continue real-time ECG testing • Continue UI development
2/27/17	• ~~Complete real-time ECG testing~~ • Complete UI development
3/6/17	• Progress Evaluation • Continue system testing and tuning
Spring Break	• Continue system testing and tuning
3/20/17	• ~~Complete all lab work~~ • Continue written deliverables • Continue system testing and tuning
3/27/17	• Continue written deliverables • Update Student Scholarship Expo poster • Continue system testing and tuning
4/3/17	• Finalize final report draft • Continue system testing and tuning

4/10/17	• Final Report Draft (4/10) • Finalize draft of presentation slides • Finalize Student Scholarship Expo poster (4/10) • Event: Student Scholarship Expo (4/11) • Oral Presentation Preparation (4/13) • Continue system testing and tuning
4/17/17	• Complete all written deliverables • Oral Presentation Preparation (4/18) • Continue system testing and tuning
4/24/17	• Event: Project demonstration • Event: Poster Presentation (4/28) • Complete all lab work
5/1/17	• Event: Presentation of project (5/2) • Project Website Verification (5/2) • Complete all written deliverables

VI. Conclusion

This project group concluded that this project is a success because it met all of its primary objectives:

- Develop a portable, mobile device with ECG sensors.
- Implement an embedded system with ECG algorithms to monitor ECG signals in real-time.
- Wirelessly notify a patient's care provider of PVC.

This project also met secondary objectives that would not detrimental to the project if not completed. These include logging data and plots and providing a simple user interface. At the end of the project timeline, the device was capable of processing, recording, and transmitting acquired and benchmark ECG data in real-time. The primary weakness at this time is the power consumption of the device, but this can be addressed in future iterations of the project.

This project built on the work of the *Real-time Heart Monitoring and ECG Signal Processing* project. Strides were made in the ECG processing component as existing code was optimized for cross-platform use. The system tasks were spread across a number of platforms with the introduction of a Linux-based system controller which is better capable of logging and handling external actions such as wireless notification. A user interface was added to make the device more usable for a care provider or patient, and to provide access to logged information.

This project group is pleased with the outcome, but acknowledges that there is much room for improvement. Like the additions to the previous project, more can be done to enhance this system, and ultimately produce a marketable product in the future.

VII. Recommendations for Future Work

This project was successful in completing its primary objectives, and included a variety of additional functionalities not initially accounted for in its original proposal. This project has much room for improvement, and if a new group were interested in taking it further, this project group has several suggestions:

- **Reduce power consumption** - According to simulated design testing, the system is currently only capable of an 8 hour battery life using a 15 Wh battery. The next suggestion explores one way to reduce power consumption.
- **Integrate more components onto single PCB** - Integrating device components including the DSP and embedded computer (if applicable) on the same board will remove development kits with unnecessary power consuming components from the design. This may also allow for increased accuracy and speed of the device. It may also involve finding a lower power embedded computer or removing the embedded computer from the system entirely.
- **Add new algorithms to detect additional types of arrhythmias** - This would increase the scope of the device, and make it more marketable in the future. There is additional processing time available in the system for additional detection algorithms to process data.
- **Make transmission and storage of patient data secure** - If this is to be a marketable product, patient information must be secure. Look into storage and transmission encryption.

VII. References

[1] “About Arrhythmia.” [Online]. Available: http://www.heart.org/HEARTORG/Conditions/Arrhythmia/AboutArrhythmia/About-Arrhythmia_UCM_002010_Article.jsp.

[2] “Why Arrhythmia Matters.” [Online]. Available: http://www.heart.org/HEARTORG/Conditions/Arrhythmia/WhyArrhythmiaMatters/Why-Arrhythmia-Matters_UCM_002023_Article.jsp.

[3] “pvc_21349464440513.jpg (1425×537).” [Online]. Available: http://classconnection.s3.amazonaws.com/833/flashcards/1119833/jpg/pvc_21349464440513.jpg

[4] M. AlGhatrif and J. Lindsay, “A brief review: history to understand fundamentals of electrocardiography,” J Community Hosp Intern Med Perspect, vol. 2, no. 1, Apr. 2012.

[5] J. A. Z. Justo, R. A. G. Calleja, and A. M. Diosdado, “Acquisition software development for monitor Holter prototype signals and its use for pre-diagnosis of cardiac damage based on nonlinear dynamic techniques,” in AIP Conference Proceedings, 2016, vol. 1747, p. 90001.

[6] F. Bamarouf, C. Crandell, and S. Tsuyuki, “Real-time heart monitoring and ECG signal processing,” Bradley University, May 2016.

[7] “acardio20140402v0005.jpg.” [Online]. Available: https://api.kramesstaywell.com/Content/ebd5aa86-5c85-4a95-a92a-a524015ce556/medical-illustrations/Images/acardio20140402v0005.jpg.

[8] “ee0e66c4-ff50-42e2-85b5-cdf9f64d6208.jpg.” [Online]. Available: http://www.multivu.com/players/English/70647514-add-wi-fi-to-anything-with-ti-s-internet-on-a-chip-new-simplelink/gallery/image/ee0e66c4-ff50-42e2-85b5-cdf9f64d6208.jpg.

[9] “med_tmdx5515ezdsp_c5515_ezdsp_board_72.jpg (350×225).” [Online]. Available: http://www.ti.com/diagrams/med_tmdx5515ezdsp_c5515_ezdsp_board_72.jpg.

[10] “913XYU1VtjL._SX355_.jpg (355×228).” [Online]. Available: https://images-na.ssl-images-amazon.com/images/I/913XYU1VtjL._SX355_.jpg.

[11] “sku_389334_1.jpg (700×700).” [Online]. Available: http://img.dxcdn.com/productimages/sku_389334_1.jpg.

[12] “tmp14285_thumb1.jpg (372×480).” [Online]. Available: http://what-when-how.com/wp-content/uploads/2012/04/tmp14285_thumb1.jpg.

[13] “1115-00.jpg (1200×900).” [Online]. Available: https://cdn-shop.adafruit.com/1200x900/1115-00.jpg.

[14] H. Khamis, R. Weiss, Y. Xie, C. W. Chang, N. H. Lovell, and S. J. Redmond, “QRS detection algorithm for telehealth electrocardiogram recordings,” IEEE Transactions on Biomedical Engineering, vol. 63, no. 7, pp. 1377–1388, Jul. 2016.

[15] N. M. Arzeno, Z.-D. Deng, and C.-S. Poon, “Analysis of first-derivative based QRS detection algorithms,” IEEE Trans Biomed Eng, vol. 55, no. 2, pp. 478–484, Feb. 2008.

[16] "Power Consumption," Raspberry Pi Dramble, [Online]. Available: https://www.pidramble.com/wiki/benchmarks/power-consumption.

Appendix A: Applicable Standards

Standards are guidelines agreed upon in an industry that ensure understandable and successful implementation and interfacing of systems. There are two applicable standards that have been identified for this project.

1. IEEE 802.11

This IEEE standard outlines a wireless networking topology, in this case, WiFi. This project is based on a Raspberry Pi 3 embedded platform, which is IEEE 802.11n compliant at 2.4GHz.

2. MISRA C

The Motor Industry Software Reliability Association (MISRA) has developed coding standards for C language. This ensures safety, security, portability, and reliability of C code across embedded systems. Developers across numerous technical industries have adopted these standards, including the medical device industry.

Appendix B: Interface Circuit Schematic