Electrocardiogram Final Report

Fiona Zeng
1565731
linglz3@uw.edu

Joseph Shin
1143310
js97@uw.edu

ABSTRACT

Heart disease is one of the most significant causes of death in humans [1]. Therefore, monitoring one's heart is imperative for early detection of potentially fatal conditions. The various electrical signals emitted from the body contain vital information pertaining to the operation of the heart. These signals can be processed to determine various statistics and information about the heart being analyzed, via an electrocardiogram. This electrocardiogram utilizes the Teensy 3.1 microcontroller as its main processing unit. First, a user must place his/her fingers on the electrodes of the constructed amplifier circuit. The Teensy then reads and analyzes the analog output signal of the amplifier circuit to determine vital statistics about the signal, including waveforms, beats-per-minute, QRS-intervals, and various potential conditions. The information is then displayed on an LCD screen that is also controlled by the Teensy. The information can also be sent to a phone via Bluetooth.

1. INTRODUCTION

The purpose of this project is to implement an electrocardiogram device that takes in a person's electrical signals through two leads on the front end circuit, graphs the person's electrocardiograph, and detects the heart rate and QRS interval of the person. The device can also detect potential heart conditions such as Bradycardia and Tachycardia. All resulting informations are shown on a liquid crystal display. Additional UI features such as a data scrolling bar and a start/stop button are also added on the display to ensure a more user-friendly experience. The intended use of this project is to aid in the detection of potentially detrimental heart conditions, as well as garner various statistics about one's heart.

2. DISCUSSION

2.1 Hardware Specifications

The electrocardiogram consists of four main components; the Teensy 3.1 microcontroller, an analog front-end amplifier circuit, an ILI9341 display, and an Adafruit Bluetooth module,

Teensy 3.1 Microcontroller

The Teensy 3.1 microcontroller is based on the MK20DX256 processor, which is an ARM Cortex-M4 processor running at 48 MHz. It has a 256K flash memory, 64K RAM, and 2K EEPROM. It contains 34 I/O pins, including 21 high resolution analog inputs. The analog to digital converter is capable of a resolution of 16 bits, with 13 bits being usable.

Analog Front-End Amplifier

The analog front-end is an amplifier circuit that utilizes INA128U instrumental amplifier, a MCP6004 operational amplifier, various resistor and capacitor components, and two dog tags as the electrodes.

ILI9341 Display

The ILI9341 is a liquid crystal display with a resolution of 240 by 320 with 262K color. It also contains a touch functionality and communicates using the SPI interface.

Adafruit Bluetooth Module

The Bluetooth module used is the Adafruit Bluefruit LE. It communicates using the SPI interface.

2.2 Hardware Theory-of-operation

Teensy 3.1 Microcontroller

The Teensy 3.1 Microcontroller is used to run the main electrocardiogram software, and connects the various peripherals, such as the display, Bluetooth adapter, and front-end amplifier circuit, together.

Analog Front-End Amplifier

The analog front-end amplifier measures the heart's electrical activity via two electrodes, which in our case were metal dog tags.

The signal is then amplified via the INA128U instrumental amplifier. The amplified signal is passed through the MCP6004 operational amplifier, which acts as a series of various filters. These filters remove various sources of noise, including the noise from the power supply, which is an AC signal that operates at 60Hz. The output of these filters in then wired directly to the analog input pin of the Teensy.

ILI9341 Display

The ILI9341 Display is used to display the analysis information received from the Teensy, which it communicates to via the SPI interface. The waveform and other statistics pertaining to the heart (beats-per-minute, QRS interval, bradycardia, tachycardia) are displayed on the screen in real-time.

Adafruit Bluetooth Module

The Adafruit Bluetooth module is used to send the beats-per-minute information to a Bluetooth enabled smartphone device. The smartphone device then utilizes the nRF Toolbox application to display and graph the received beats-per-minute information. It communicates to the Teensy via the SPI interface.

2.3 Software Theory-of-operation

The software utilized for the functionality of the electrocardiogram consists of three main aspects, the UI, the sampling of the signal, and the filtering of that signal.

Software UI

The user interface for the electrocardiogram consists of several prompts for the user. When the program begins, a prompt is given for the user to connect to the electrocardiogram via Bluetooth.

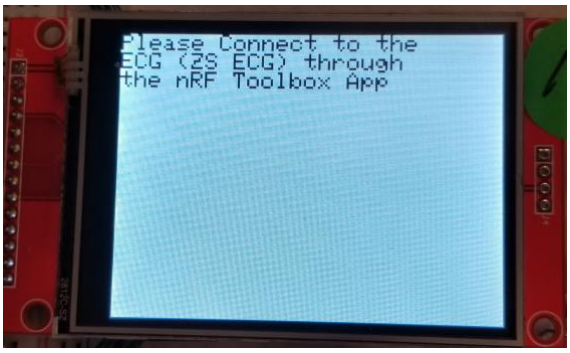

Figure 1: Bluetooth Prompt Screen

Following the prompt, a welcome screen is shown. The user can then begin the program utilizing the onscreen "start" button.

Figure 2: ECG Program Welcome Screen

The program then attempts to find an adaptive threshold, and using that displays the relevant information (waveform, beats-per-minute, QRS interval, tachycardia, and bradycardia) for 30 seconds.

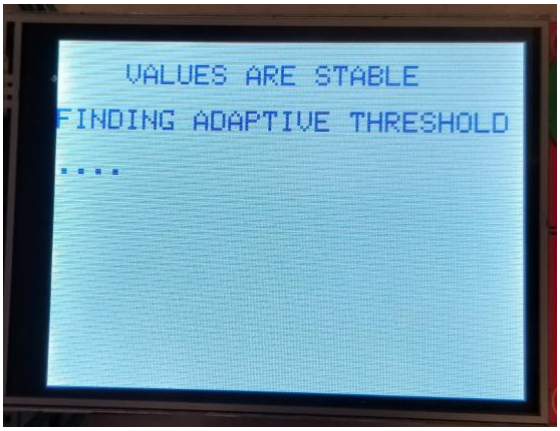

Figure 3: Adaptive Threshold Phase Screen

Figure 4: Main Program Screen

After the time has elapsed, the user can then scroll through the recorded data using the scroll bar on the bottom.

Sampling Rate

The sampling rate utilized in the electrocardiogram was 100Hz. It was set in the PDB (programmable delay block) of the Teensy 3.1, which triggers the ADC (analog to digital converter) to read 100 samples per second.

The sampling rate was determined based on the maximum observed heart rates. Generally, the maximum heart rate for humans is around 220 beats per minute [2]. This corresponds to a frequency of 3.66 Hz. Using the Nyquist Sampling theorem, this corresponds to a Nyquist frequency of 7.32 Hz; 100 Hz is more than adequate for our purposes of detecting a human heart rate.

Filtering

The output of the ADC contained extreme amounts of noise that hid many integral features of the wave. As a result, we utilized extensive filtering on the raw signal to achieve a usable signal. The raw signal is filtered by a low-pass filter, a high-pass filter, a differentiator, a squaring filter, and a moving window integrator. The filters were implemented according to the algorithms found in the given *Real Time ECG Feature Extraction and Arrhythmia Detection on a Mobile Platform* [3] document.

Low-Pass/High-Pass Filters

The raw signal contains many different sources of noise. Noise can emanate from the 60 Hz power line, muscular activity, and general interference from other electronic devices. Therefore, filters must be implemented to remove unwanted data.

This first filter utilized was a 2nd order Butterworth low-pass filter, with a cutoff of 18 Hz. The lower frequency noise was eliminated with a 2nd order Butterworth high-pass filter with a cutoff of 3 Hz. These filters were then combined in software using the Filter Design Tool [4] in a single 2nd order Butterworth bandpass filter, as seen below.

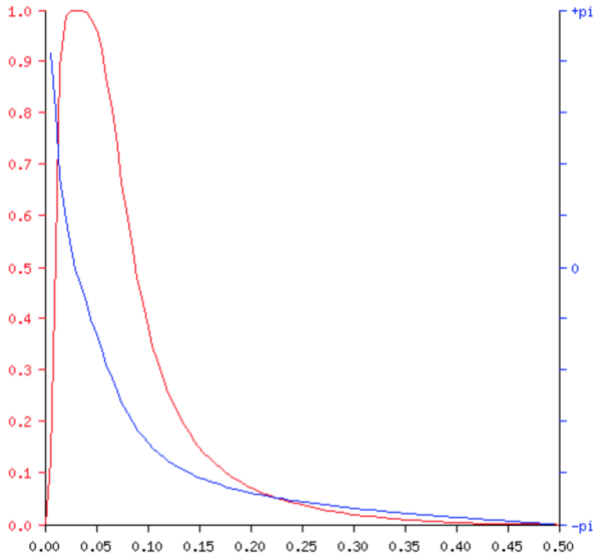

Figure 5: Magnitude (red) & phase (blue) vs frequency

Differentiator Filter

The output of the bandpass filter then goes into a differentiator filter to provide information regarding the QRS slope. The transfer function is as follows: $H(z) = 0.1 (2 + z^{-1} – z^{-3} - 2 z^{-4})$

Squaring Filter

The output of the differentiator is then squared. This is done so that all of the data points are positive, and amplifies the higher frequencies for easier detection.

Moving Window Integrator

Finally, after the signal is squared, it is then put through a moving window integration filter. This is done to obtain additional information concerning the slope of the R wave. It is implemented with the following equation: y[n] = (x[n - (N-1)] + x[n - (N-2)] + … + x[n]) / N, with N being the number of samples. The final equation utilized was: y[n] = .8 * x[n] + .2 * y[n].

2.4 QRS/Arrhythmia Theory-of-operation

Detecting the QRS peaks is imperative for a functional electrocardiogram. This detection is required for realtime analysis pertaining to arrhythmia.

QRS

The QRS peaks are detected by reading the filtered signal. When the program begins, an adaptive threshold is set for the R peak. The threshold is found by continuously modifying a threshold so the program can detect 5 consecutive beats within a normal heart range.

After the threshold is found, the R peak is detected by comparing each value of the filtered signal to the R peak. If the signal surpasses the threshold, an R peak is found. The interval between the R peaks is where the beats-per-minute measurement is found. To prevent false positives, after an R peak is found, the program waits a minimum time before searching for another R peak. This prevents accidentally reading an R peak, or other noise peaks, twice.

For the QRS interval, the Q and S peaks are found in a similar way. A smaller threshold range is utilized to accurately detect the Q and S. To prevent false positives in determining the QRS interval, the S peak is only searched for after the R peak is found.

Arrhythmia

The detection of various arrhythmias is found directly from the beats-per-minute measurement. For our purposes, bradycardia is detected if the average heart rate was under 60 beats-per-minute. Tachycardia is detected if the average heart rate exceeds 110 beats-per-minute.

3. FUTURE WORK

This electrocardiogram implementation has much room for improvement. Currently, it can detect heart rate, QRS interval, bradycardia and tachycardia. Additional features such as the PR interval, and Premature Ventricular Contraction/Premature Atrial Contraction detection could be added. In terms of the front-end amplifier itself; whilst software did remove much of the noise, perhaps better filtering could be implemented physically to further remove other sources of noise.

4. CONCLUSION

Our final product was a success in that in can take in a user's raw electrocardio signal and effectively filter out noise. Using an adaptive threshold, the system can accurately detect the R peak of a particular waveform, further identify the QRS interval and calculate the heart rate. With the heart rate data, the system can decide whether the user has underlying heart conditions such as Bradycardia and Tachycardia. The UI button and scroll bar added on the liquid crystal display enables the user to easily interact with the data. Overall our electrocardiogram is low-cost, and can accurately detect basic information about the heart.

The design did not entail really any trade-offs. All of the functionality was built upon the analog signal received from the front-end amplifier circuit. Perhaps in future designs, a more robust circuit could be utilized, allowing for better, more accurate detection of all of the peaks, and consequently the detection of other heart conditions.

5. REFERENCES

[1] Mayo Clinic. Heart Disease: Symptoms & causes. Retrieved from https://www.mayoclinic.org/diseases-conditions/heart-disease/symptoms-causes/syc-20353118.

[2] Mayo Clinic. Healthy Lifestyle: Fitness. Retrieved from https://www.mayoclinic.org/healthy-lifestyle/fitness/in-depth/exercise-intensity/art-20046887?pg=2.

[3] Abhilasha M. Patel, Pankaj K. Gakare, A. N. Cheeran. 2012. *Real Time ECG Feature Extraction and Arrhythmia Detection on a Mobile Platform.* International Journal of Computer Applications.

[4] University of New York, Department of Computer Science. Interactive Digital Filter Design. Retrieved from http://www-users.cs.york.ac.uk/~fisher/mkfilter/.