## ECG Project Final Report

Zhimingyuan Liu

University of Washington

Electrical Engineering  
1561224

Caesar Gu

University of Washington

Electrical Engineering  
1530980

**ABSTRACT**

In this project, we built a Electrocardiograph with a amplifier circuit, LCD display, Arduino teensy and a bluetooth module. It can take 30 seconds data of heart rate and display the real time ECG analysis including heart rate, RR time interval, QRS, PR interval. It helps users to detect their heart rate situation and get analysis to determine whether their heart is in good condition.

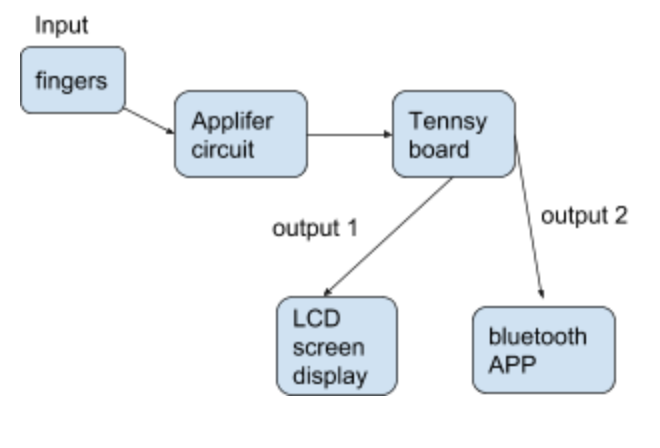
# INTRODUCTION

The purpose of our project is to build a fully functional ECG that can take heart rate data from fingers, analyze the data to get the general health condition of heart. We took data from the amplifier circuit, pass it through the ADC on teensy to get digital data of heart rate. Then the data will go through multiple filters to reduce noises. After coming out from filters, we applied QRS detection algorithm. This algorithm will measure the RR time interval to get heart rate. We can use this result to determine whether the patient has Bradycardia or Tachycardia. The result will be displayed on the LCD screen after 30 seconds. The algorithm will also measure the QRS and PRI so that we can determine whether the patient has Premature Ventricular Contraction or Premature Atrial Contraction. If PVC or PAC is being detected during the measure procedure, unlike Bradycardia and Tachycardia which are determined after 30 seconds. As long as PVC or PAC is detected, the LCD display will show it to warn the patient. After the end of data taking, we can also scroll back the recorded waveform so that we can find which specific QRS wave piece has problem.

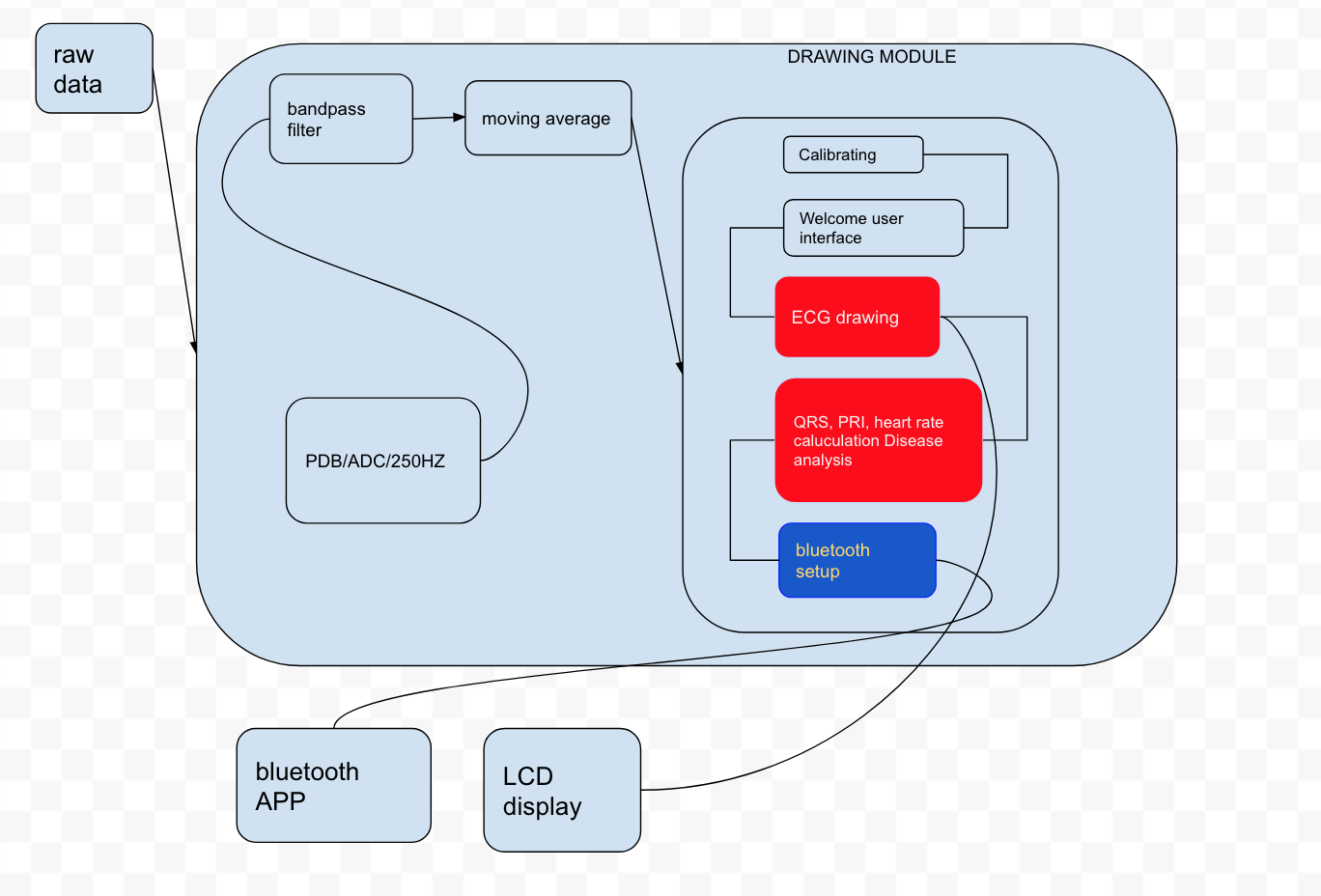
* **SOLUTION DESCRIPTION**

**Hardware implementation**

bellowing is the general implementation of hardware, which we put fingers on amplifier circuit. The data go through Teensy microcontroller. After processing the data the output shows on LCD screen and bluetooth APP



Bellowing is the hardware implementation within teensy board combined with PDB; bandpass filter ECG display;moving average for filter out data; QRS; PRI; heart rate calculation; and bluetooth setup.



# 

# Amplifier circuit

We followed the instruction on the website and built the amplifier circuit. We used oscilloscope to see the output waveform of the circuit to make sure it can correctly generate the heart rate signal. Then we connected it to the arduino teensy. Since the output of this circuit is an analog signal. We used what TA taught us to build a ADC on the teensy. Then we tried to directly display the data out of ADC on the LCD display. The output of ADC is from 0 to 4096 but the display is at most 240. Thus we add a map to adjust the data. It matched what we saw on the oscilloscope so we knew that the circuit is working with the ADC.

**Stabilizing**

Our device has the function that it can detect whether user’s finger has been put on the pad. After the fingers are on, it will keep tracking the input data and determine if the signal is stable. As long as the signal is stable, the entire ECG process will start to work. We achieve this by tracking the variance of the input data. We build a data buffer with size of 100 and keep calculating its variance. When the fingers haven’t been put on, the input data is stable so we set a small variance threshold. When the variance is larger than the threshold, the finger should be put on and it enters next stage. When the fingers are first put on, there are some really big noise, then the signal will be generally stable. Our second stage is detecting whether the variance reduces from a large level to a relatively level. Thus we set a large threshold here. As long as the variance is smaller than this threshold, it enters the main stage of ECG. During all these stages, the LCD display will print which stage the user is in.

**PDB/ADC (Software)**

Instead of using analogread, we use PDB to read data into out system. We set our PDB frequency as 250 HZ, so every 4 ms, there is data read in from channel. Based on code from lab5, we delete all DMA clock, and within the adc0\_isr method, we read value in

void adc0\_isr() {

adcValue = ADC0\_RA;

}

# Bandpass filter (Software)

The original data out of ADC still has a lot of noise, which would affect our QRS detection. To avoid unnecessary error, we needed to build a filter based on software to get rid of the noise. As were taught, the frequency of ECG signal is between 0.5 Hz to 40 Hz. A bandpass filter is suitable for this condition. We used the method TA gave us to build the bandpass filter.[1] This bandpass filter consists of a low pass filter and a high pass filter. Low pass filter will get rid of all the noise that is with frequency that is higher than 40Hz. Then the high pass filter will get rid of the noise that is with frequency that is lower than 0.5 Hz. The output of bandpass filter is from -2048 to 2048. Then we mapped it to 240 again. Then we applied an average structure to smooth the data. and display it on the LCD display . The waveform is much more smooth than before. There is much fewer noise than the old display. After finishing this part, the display of ECG is actually almost done. What is left is detection and analysis.

## Heart Rate detection (Software)

For heart rate detection, we use the simplest logic: detect the peak of the wave. Although peak can definitely be defined as a R point, the value of each peak is different. Some peak is relatively small and some is big. Thus it is necessary to set a self-adjust threshold to accurately detect the peak. We first build a buffer to store different peak value. We use the first 1 - 2 seconds to fulfill the buffer then we took an average of them as the initial peak threshold. Then we start to determine the peaks that we will use to calculate the heart rate. If a value is larger than threshold - 6 and smaller than threshold + 6, it will be defined as a peak. A timer will be set at that instant. And this peak value will be add to the threshold buffer so that we can calculate a new real time threshold. When next peak is detected, the timer will stop and we will get a RR interval time(ms). This interval time will be add to a RR interval time buffer and we will applied the average calculation again to smooth the result. Then we will use 60000/current RR interval time(ms) to get a real time heart rate result. This result will later be displayed on the LCD screen.

## QRS detection (Software)

For QRS detection, we use the algorithm different from the paper. Due to the fact that our QRS is quite smooth, and there is basically no noise, except the heart rate frequency. We detect QR wave and RS wave individually by using the derivative between two points. In order to get standard derivative for QR wave, we first find the rapid change point, and record its (time - 20) (due to the fact that Q actually starts from last point), then we continuously find the point where the derivative become 0, and record the time. Then R point is founded. then we go through the loop to find where the derivative become 0 again. After the R point, it will decrease until it become flat again. According to that algorithm, we find the QR interval and RS interval. then we use buffer size of 10 to shift each data interval. and get average value from it, so the QRS interval will not fluctuate so much.

## PRI detection (Software)

For PRI detection, we use the algorithm that is similar to QRS detection. The difference between P wave and Q wave is that Q wave has a really rapid increase while P wave is much smaller. Thus we adjust the threshold for detecting the derivative of P wave. Since our signal is really smooth, as long as there is a continuous increase, we can define it as a special wave point. Although we use a small threshold for P wave, it is still accurate enough. After we detect P wave, we set a timer and use QRS detection algorithm to find next Q point. Then we calculate the time interval between P and Q. It is not 100 percent PR, but it is really close and we can add some constant to adjust it to be closer to PR. All the result of calculation will enter a shift data buffer and get a average to smooth the change. Then we will display it on the LCD.

**Bluetooth Setup (Software/Hardware)**

For the bluetooth part, our goal is to print heart rate on APP. After including the library for bluetooth, we write some if else logic for initializing bluetooth, and we build void to print BMP on APP. Finally, we download nRF Toolbox from APP store. Our bluetooth function is only in charge of heart rate, instead of ECG waveform.

**Scrolling (Software)**

For scrolling, we firstly design our scroll function happened after 30 seconds ECG display, and all the ECG analysis has already finish. In that way, it will not influence the current running data.At first we define boolean istouched, and every time we touch screen and scroll, we calculate the x axis change between point we current touch and point we first detect. then we scale down the point difference by 5 and we can get the starting index and ending index of draw buffer and draw the data on the screen. once we leave the LCD untouched, and we touch again, the initial point for change reference of other points will also updated, until we leave our fingers on screen.

* **FUTURE IMPROVEMENTS**

**QRS, PRI detection improvement**

For QRS and PRI detection, we can use the A REAL-Time QRS Detection Algorithm by JIAPU PAN and WILLIS J. TOMPKINS. It has the sequence of bandpass filter, derivative function, squaring function, and integration function, which will give us more accurate calculation for threshold voltage, heart rate, and QRS, PRI.

**BLUETOOTH APP update time and ECG analysis update time on LCD**

In our design, the ECG data is showing on the same page as ECG display. We make our ECG data analysis updating speed reduce as 5 times slow as ECG drawing, because if it is too fast the value of ECG drawing data will be influenced a lot (wave move really slow, and the peak is hard to detect). but when we implementing the bluetooth, the heart rate updating speed is much more faster than heart rate data on LCD, which is not quite synchronized. In the future, we may put more effort on synchronized bluetooth data and ECG analysis data on LCD.

**USER INTERFACE DESIGN**

Our user interface is a single senquency, with Welcome page; bluetooth setup, ECG display, ECG analysis, and finally scrolling. In the future, we may design multiple sequences programs. For example, users can choose whatever they want to review the data, and scrolling, and they can select whether to connect with bluetooth device. After scrolling, users can detect heart rate again, if they want. There will be a option let them to play it again.

**Calibrating Process**

In our design, we only go through the calibration process, when it first start to draw ECG graph. However, in the real world, if the noise is too much, the system will automatically go through calibrating process again. We should design out system with a dynamic calibrating function, which keep calculating the standard deviation with the raw data in. Once it is too much, it should return the program to the first page.

* **CONCLUSION**

We finally make all the functions we intended to make work. The accuracy of our ECG device is good but not perfect. Sometimes there is still some unstable condition appearing. Its ability to resist to external interference is relatively week. We were assuming our user will hold their finger peacefully. But as long as it meets our pre-condition, the measure and analysis will be accurate enough. The ECG waveform, heart rate, QRS interval, PR interval, RR interval and ECG analysis can be displayed clearly and beautifully on the LCD display. The ECG waveform record can be tracked through scrolling smoothly. It is a successful result for this project.

* **Reference List**

[1] Butterworth / Bessel / Chebyshev Filters：  
http://www-users.cs.york.ac.uk/~fisher/mkfilter/trad.html