|  |  |
| --- | --- |
|  | |
|  | |
| Interim Report | |
| **Module code:** | **EG3005** |
| **Module name:** | **Third Year Project** |
|  | |
| 9th January 2019 | |
|  | |
| **Author(s):** | **Nishat Tarannum** |
| **Student ID(s):** | **169018358** |
| **Degree:** | **MEng Software and Electronic Engineering** |
| **Tutor/Project supervisor:** | **Mr. Andrew Norman** |
|  | |
| **SUPERVISOR’S COPY/EXAMINER’S COPY [For 3rd/4th year projects only - delete as appropriate]** | |

By submitting this report for assessment I confirm that this assignment is my own work, is not copied from any other person's work (published or unpublished), and that it has not previously been submitted for assessment on any other module or course.

I am aware of the University of Leicester’s policy on plagiarism, and have taken the online tutorial on avoiding plagiarism.  I am aware that plagiarism in this project report may result in the application of severe penalties up to and including expulsion from the University without a degree.

|  |
| --- |
|  |
| Summary |
| The aim of this project is to make a portable, heart rate monitor which will be able to diagnose any abnormal heartbeat patterns such as arrhythmias. This device will enable people to know immediately if they are suffering from such heart conditions without going to a doctor. The device shall be compact and easy to use to allow people of most ages to use it comfortably. This report covers the work done, decisions taken, and the various challenges faced so far. It further details any future plans and improvements. |

Contents

[1 Introduction and Background 2](#_Toc534758163)

[1.1 Types of Irregular Heart Rhythms 2](#_Toc534758164)

[1.2 Heart Rate Variability 2](#_Toc534758165)

[1.3 Importance of Filtering Noise in Signals 3](#_Toc534758166)

[2 Method and Decisions 3](#_Toc534758167)

[3 Results 6](#_Toc534758168)

[4 Future Plans 8](#_Toc534758169)

[References 9](#_Toc534758170)

[Student Self-reflection on performance 10](#_Toc534758171)

# Introduction and Background

The purpose of this project is to make a device which is able to detect any irregularities in the heart beat pattern. Users will be able to diagnose any issues immediately (such as arrhythmias) without consulting a doctor or having to do an electrocardiogram (ECG). The device will be portable, it will be implemented using a Raspberry Pi, an Arduino and a photoplethysmographic (PPG) sensor which will be the substitute for ECG electrodes. This will increase usability as people of most ages will be comfortable in using the device with minimal technical knowledge.

ECG electrodes were not considered because for an ECG three points of different voltage are required to produce the ECG trace. Furthermore, a conducting gel has to be applied to the body to reduce resistance between skin and electrodes. This means that overall this method is less user friendly and requires other components whereas the PPG method does not need anything other than the sensor.

## Types of Irregular Heart Rhythms

An arrhythmia is defined as a breakage of the rhythm (of the heart beat) either in timing or shape (of the ECG trace). If the shape of the ECG trace is different to a normal ECG trace, it means that the electrical activation didn’t occur in the sequence it should have. The mechanical action will also be out of sequence thus the shape of the trace will be different. This is a cardiac arrhythmia.

One type of arrhythmia is called atrial fibrillation. This causes the heart rhythm to be irregular, in which case sometimes the electrical impulses are propagated to the ventricles and the ventricles beat. Other times the atrial contraction which are p-waves in the ECG trace, is not seen because there is a limited time in which the waves can be captured. Consequently, the beat to beat timing interval in atrial fibrillation is chaotic, with no regular pattern in timing.



Figure 1 - ECG Trace of Person with Atrial Fibrillation [1]

## Heart Rate Variability

A healthy heart beat contains irregularities. Heart rate variability (HRV) means the variation in time between consecutive heartbeats. It is universally accepted as a non-invasive marker of autonomic nervous system (ANS) activity. Variability is different to the heart rate, as it increases when doing relaxing activities and decreases when in stress. Thus, HRV is generally greater when the heart rate is low and smaller when the heart rate is high. The relationship between HRV and heart rate is inversely proportional [4]. Variations are detected in an ECG trace by measuring the intervals between heartbeats are called R-R intervals. The R peak is the large spike at the beginning of a QRS complex as shown below in Figure 2:

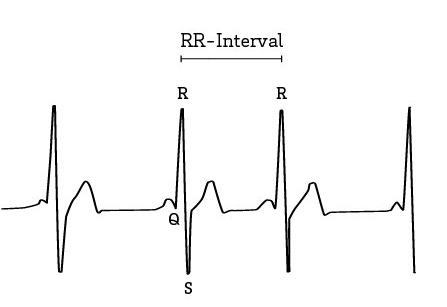


Figure 2 - QRS complex of ECG Trace [5]

To calculate the amount of variation in the intervals between heartbeats rMSSD (Root Mean Square of the Successive Differences) can be used which is the most commonly used HRV formula [5]. Patients with a history of atrial fibrillation record episodes HRV is significantly reduced under rMSSD (<30ms) [7].

New studies also showed that low-resting short-term heart rate variability was associated with elevated incidence of atrial fibrillation. Higher risk for a higher risk for atrial fibrillation was linked to lower overall heart rate variability. A baseline heart rate of < 60 beats per minute was associated with an increased risk for atrial fibrillation [6]. Thus, these measures could be used to find arrhythmias.

## Importance of Filtering Noise in Signals

Filtering is a process that removes unwanted components from a signal. This is especially important in these types of applications where any noise or interference could lead to incorrect peak detections and thus can influence when measuring the heart rate. Different types of filters could be used for heart rate processing however, the most common filter used is the Butterworth filter as it has maximum flat frequency response. Furthermore, low pass or bandpass filters would be best for this type of application as for an ECG trace values beyond 100Hz will probably be unnecessary for detecting arrhythmias. In a low pass filter, high frequencies are attenuated. For a band pass filter on the other hand, frequencies in a specific frequency band are passed. The cut-off frequency is also an important factor for filtering it is the frequency beyond signals will not pass through it [8].

# Method and Decisions

After the components had arrived, the first thing done was setting up the Raspberry Pi. The operating system (Rasbian) was downloaded onto the SD card and installed onto the Pi as well as other software which would be required for the project, such as the Arduino IDE etc. Next, the Arduino was set up and the photoplethysmographic (PPG) pulse sensor was connected. This sensor has to either be clipped to the ear lobe or wrapped around a finger using the Velcro strap provided.

The pulse sensor uses photoplethysmography to measure the heart rate and this is displayed on the Arduino serial monitor using some code which is provided in an Arduino library specifically for the pulse sensor called PulseSensorPlayground. This library contains basic programs which can plot the heart rate trace and display the heart rate.

Photoplethysmography (PPG) works through the use of low-intensity infrared green light at high precision. Various tissues in the body absorb light when light penetrates through the tissues. It is more strongly absorbed by the blood compared to other tissues such as bone etc. This variation in absorption means that PPG sensors are able to detect changes in blood flow, by using the changes in light intensity. The amount of blood flow through the blood vessels is proportional to the PPG voltage signal. PPG can even be used to detect small changes in the volume of blood, this means it is able to provide a higher resolution (for more information please check [2]).

Purely for testing the functionality and quality of the pulse sensor, the provided library code was used to display the heart rate trace on the Arduino serial monitor. The sensor was quite sensitive to movement which could be seen from the trace, as trace was a bit erratic when the heart rate was measured while moving the finger. Furthermore, there was some noise visible in the signal, shown in Figure 3, in the results section below. Thus, it was decided to filter the signal before doing using the data to find any irregularities in the heart rhythm.

Before filtering, the original data was written to a text file using some Python code shown in Figure 6. This was done so that testing for heart rhythm problems could be done more conveniently as this file could be used as required instead of having to acquire new data when it was needed. This code wrote values to a text file sent from the Arduino (which was reading the sensor data); the Arduino was sampling the sensor values at a rate of 100 Hz. To keep the sampling frequency constant, timer interrupt coding was used on the Arduino, the code is shown in Figure 5.

The Python code in Figure 6, listens to the port that the Arduino is sending information to and it writes each value read by the Arduino to the text file; each value is written on a new line until the program is terminated or there is an error. The Arduino timer interrupt code in Figure 5, basically sets up the timer, the 64-bit prescaler and finally the match compare register so that every 100 Hz there is an interrupt. The baud rate for this is set to the maximum value of 115200 to ensure that serial communication is faster than the data sampling rate. The code reads the value of the pulse sensor and assigns the value to a variable, this occurs 100 times each second.

Once the data was written to the text file, it was then pre-processed by filtering. First, the text file including the data was converted into an array in Python, so that filtering could be done using various Python libraries such as SciPy and MatPlotLib was used for plotting the signal. A Butterworth filter was used after researching through many different filters. This was mainly done to remove noise and eventually to do a Fourier transform to separate the heart beat frequencies. This would allow for detecting peaks and eventually the heart rate. A low pass Butterworth filter (response is maximally flat which means there are no ripples) was used to remove noise and to smoothen the signal. The higher the order of the filter the closer it is to an ideal filter that has unity gain in the passband [3]. A sixth order filter was used, and the cut-off frequency was set to 0.5 (as this is a digital filter ωn had to be between 0 and 1). These values reduced the noise and smoothen the signal to the degree required for Fourier transforms as shown in Figure 4 and code is shown in Figure 7.

Eventually, a Fourier transform was done however the results were not satisfactory. There was a large peak at 0 and no other prominent frequencies could be seen from the transform. So that method was abandoned. The Fourier transform didn’t work because an ECG signal is a periodic signal however it is not a signal that can be reconstructed via sine and cosine waves. Hence, that method was unsuccessful for splitting the waveform into different frequencies.

As an alternative, peaks were identified to work out the heart rate and this was done by coding a different function shown in Figure 8. This code takes the sampling frequency, the array of the filtered signal and a threshold value to above which peaks will be identified. E.g. the threshold value could be 530, this is dependent on the values captured in the filtered data from the sensor and the threshold should be a suitable value. This value can be chosen by plotting the signal and identifying above which value do most peaks occur. The function itself looks at each value in the filtered signal individually and checks if the current value is greater than the previous value, the next value and also greater than the threshold. If it is true, then a peak is identified, and it is added to a variable called beat\_count. A few calculations are done to eventually get the beats per minute and it is rounded to return a whole number from the function.

Recently, a toolkit for processing heart related data was found and some initial processing was done using the functions in that toolkit, purely for testing purposes and to find out if the functions in that toolkit could assist in the project. The library was called HeartPy was installed. The code used some measures that related to heart rate variability such as rMSSD, and also used a different peak detection and rejection mechanism to calculate the heart rate. To detect peaks, it used a fitting function in which the standard deviation between successive differences was reduced. This method looks for the optimal threshold of the data by looking at the regularity of the heart rate. Some incorrect peaks which were detected may still remain due to various factors, these peaks were rejected based on a thresholded value for the RR-intervals.

The code in Figure 9, takes a set of data and it processes the data and returns a measures dictionary containing results. So, the function named process, in the code, takes the following arguments: an array of data, the frequency at which the data is sampled, and it also shows the time for processing the data. The next part prints the values of some values of the measures dictionary such as heart rate in beats per minute, sdsd (standard deviation between successive differences) and rMSSD. However, nothing more was done with this information as the semester had finished.

It was noticed that the previous function in Figure 8 gave a heart rate of 66 beats per minute and the value of the bpm from the measures dictionary was 68 beats per minute (Figure 10), which showed that the temporary function is promising and can be further developed into something similar to the HeartPy function which uses a fitting function to choose the best threshold.

# Results

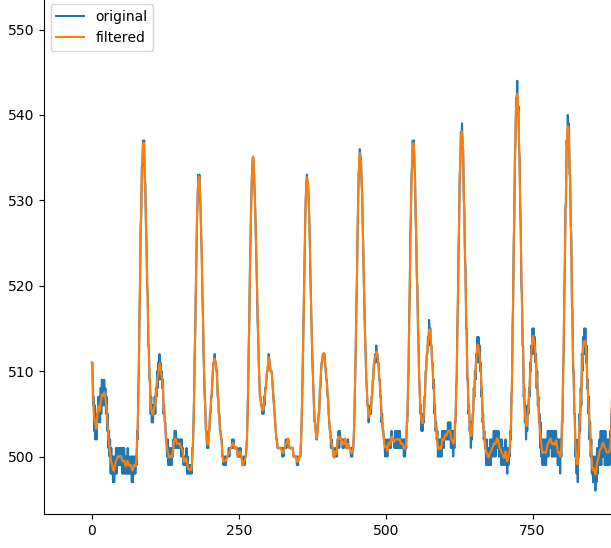


Figure 4 - Filtered and unfiltered signal comparison

Figure 3 - Original signal showing noise

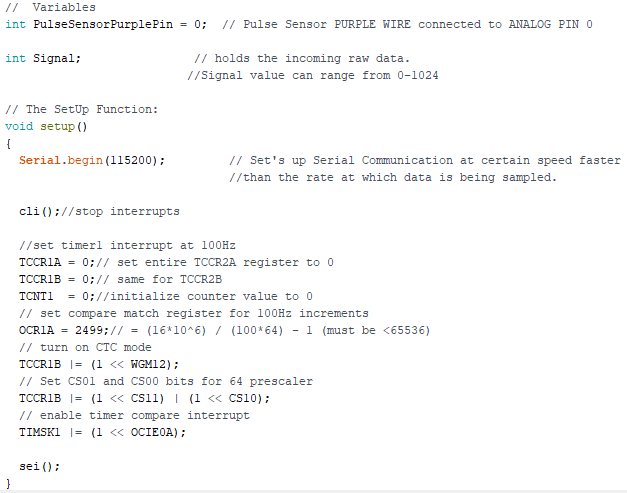
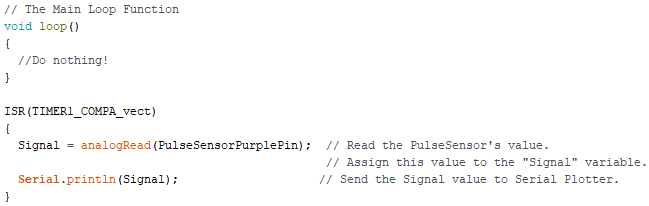
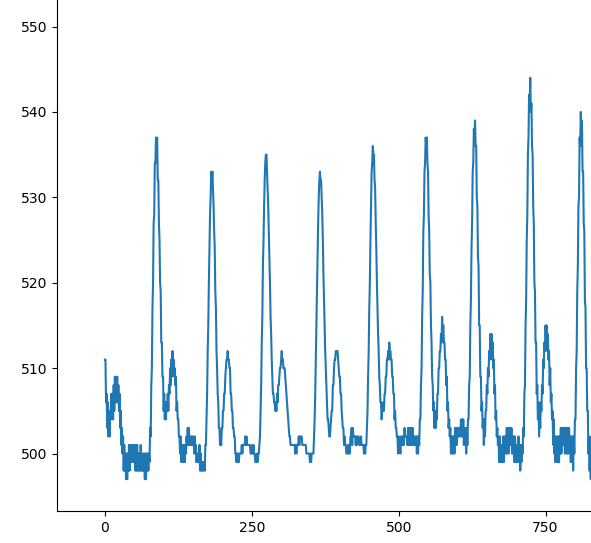


Figure 5 - Timer Interrupt Arduino Code

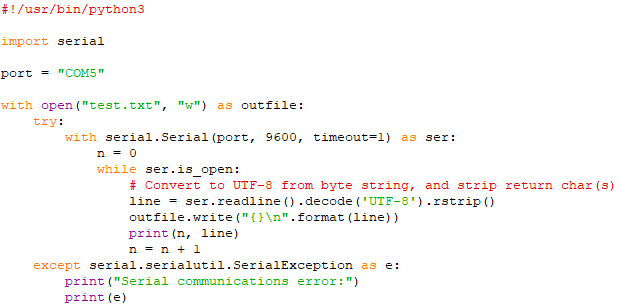


Figure 6 - Python code for writing data sent from Arduino to text file

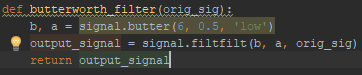


Figure 7 – Python Butterworth Filter Code

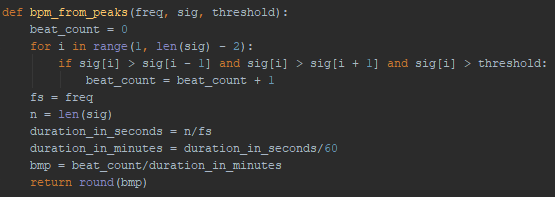


Figure 8 - Python Code that Measures the Heart Rate

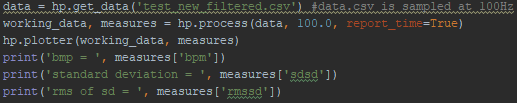


Figure 9 - HeartPy example code

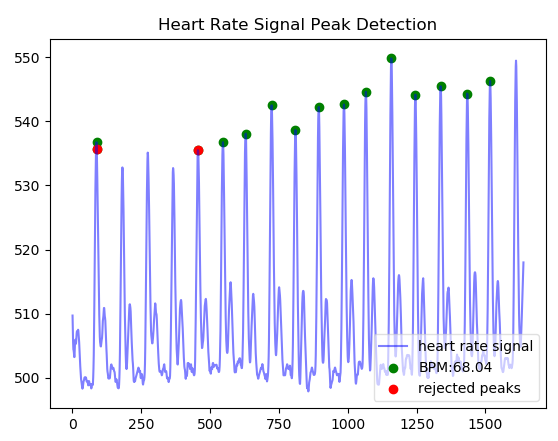


Figure 10 - HeartPy plot showing detected and rejected peaks and heart rate

# Future Plans

Currently, what has been done hasn’t totally been to plan according to the Gantt chart in the proposal. Setting up the Raspberry Pi and waiting for components to arrive etc, these things have led to being slightly behind schedule as well as some time was spent on Fourier transforms.

The original Gantt chart is shown below in Figure 11:



Figure 11 – Initial Gantt Chart

Currently, everything has been done until activity 8. The method for looking at heart variance will most probably be root mean square of the standard deviation between two points as that is the most popular method, as found from research. This will probably have to be developed and researched further to make the method more reliable by finding what out what values of rMSSD prove that a person has arrhythmia or a risk to developing arrhythmia. It can be seen that there is evidence to suggest that low heart rate variability is linked to arrhythmia such as atrial fibrillation.

At this moment in time, activity 3 hasn’t been needed because MySQL has not been incorporated into the project as of yet. It is still something to be considered, if the data from the pulse sensor was able to be logged to a database and the user would be able to track their heart rate, how frequent their episodes of arrhythmia were or if they have a specific time of the day when their arrhythmia increases etc. So that is something which may be added to the project after a better peak detection algorithm is made which is able to fit the optimal threshold without the user having to do it manually.

Furthermore, if these issues are resolved then the MySQL database could be connected to a server and users would be able to access their data online via an app or a website similar to the Fitbit. And they would be able to track their data anywhere and anytime.

So, the next steps are to: first resolve how to find if a person has arrhythmia maybe not only use rMSSD, there could be other measures that may help in proving someone has arrhythmia and also create a better peak detection algorithm.

References

[1] Healio.com. (2018). Atrial Fibrillation ECG Review - Criteria and Examples | LearntheHeart.com. [ONLINE] Available at: https://www.healio.com/cardiology/learn-the-heart/ecg-review/ecg-topic-reviews-and-criteria/atrial-fibrillation-review# [Accessed 8 Oct. 2018].

[2] Soulfit. 2018. How Does PPG Technology Works?. [ONLINE] Available at: https://soulfit.io/blog/how-does-ppg-technology-works/ [Accessed 7 January 2019].

[3] StackExchange. 2016. Higher Order Butterworth Filters. [ONLINE] Available at: https://dsp.stackexchange.com/questions/34127/higher-order-butterworth-filters. [Accessed 8 January 2019].

[4] Firstbeat. 2019. Heart Rate Variability. [ONLINE] Available at: https://www.firstbeat.com/en/science-and-physiology/heart-rate-variability/. [Accessed 8 January 2019].

[5] Ouraring. 2017. What Is Heart Rate Variability And What Can You Learn From It. [ONLINE] Available at: https://ouraring.com/heart-rate-variability-basics/. [Accessed 8 January 2019].

[6] Healio. 2017. Low heart rate variability linked to higher AF risk. [ONLINE] Available at: https://www.healio.com/cardiology/arrhythmia-disorders/news/online/%7B44ee2ed9-8cb2-4ca2-a8e5-1307e94e9d01%7D/low-heart-rate-variability-linked-to-higher-af-risk. [Accessed 8 January 2019].

[7] Barauskiene, V., Rumbinaite, E., Karuzas, A., Martinkute, E. and Puodziukynas, A. (2016). Importance of Heart Rate Variability in Patients with Atrial Fibrillation. Journal of Cardiology & Clinical Research, p.1.

[8] Wikipedia. 2018. Filter (signal processing). [ONLINE] Available at: https://en.wikipedia.org/wiki/Filter\_(signal\_processing). [Accessed 8 January 2019].

Student Self-reflection on performance

All students must complete the following sections for every piece of work they submit using this template. The aim of this is to help you use feedback more effectively to improve your marks and your skills as a professional engineer. This section is not formally marked, but your tutor may use it when discussing your work with you.

|  |
| --- |
| Describe how you have used AT LEAST ONE of the following sources of information to improve this piece of work:  1.) (PREFERRED) Feedback from previous assignment(s). This can be from the same module or from a previous module or previous year of study (e.g. comments from 1st year lab formal reports should be used to help improve your 2nd year lab formal reports).  2.) The marking criteria or rubric provided for this assignment.  3.) The Department Technical Writing Handbook for Students. |
| Feedback from first year lab reports and second year design reports have helped in writing this report. Furthermore, the technical writing handbook also supported in referencing. |
| Are there any aspects of this work that you would specifically like the marker to comment/or advise on? For example: “I wasn’t sure if my figure formatting looked professional and would appreciate feedback on this aspect” |
| I wasn’t sure if I have written enough about my project as a whole and if more background information was required, I would like some feedback on this. Thank you. |