2019


UNIVERSITY of LIMERICK

OLLSCOIL LUIMNIGH


# Using 24GHz RADAR for Contactless Heart Rate Monitoring

SPONSORED BY: ANALOG DEVICES, INC. – AUTOMATED TRANSPORT AND SAFETY DEPARTMENT

WRITTEN BY: Oisin Watkins, 15156176

SUPERVISOR: Colin Fitzpatrick

ADI CO-ORDINATOR: Padraic O'Reilly

COURSE: Electronic & Computer Engineering; General Stream (LM118)

DEPARTMENT: ECE Department

# Contents



# Figures



# Tables

# Nomenclature

| Acronym | Meaning |
|---|---|
| UL | University of Limerick |
| ATS | Automated Transport & Safety |
| ADI | Analog Devices, Inc. |
| RADAR | Ranged Detection and Ranging |
| IP | Intellectual Property |
| GUI | Graphical User Interface |
| DBF | Digital Beam Forming |
| FMCW | Frequency-Modulated Continuous-Wave |
| MIMO | Multiple Input, Multiple Output |
| HRV | Heartrate Variability |
| FFT | Fast-Fourier Transform |
| DSP | Digital Signal Processing |
| guide | Graphical User Interface Design Environment |
| MPE | Mean Percentage Error |
| MAD | Mean Absolute Deviation |
| MSE | Mean Square Error |
| RMS | Root Mean Square |
| MAPE | Mean Absolute Percentage Error |

Author: Oisin Watkins
Department: *ATS Design Evaluation*
Organisation: *Analog Devices, Inc.*
City, Country: Limerick, Ireland
Email: 15156176@studentmail.ul.ie /
oisin.watkins@analog.com

***Abstract*—This project was specified in consultation with and partly funded by the Automated Transport and Safety (ATS) department of Analog Devices, Inc. (ADI) Limerick, with a major part of the testing being conducted on University of Limerick (UL) campus. The goal is to use the ADI Demorad 24GHz RADAR platform to measure Cardiopulmonary movement at a distance of 1 meter from the subject with an accuracy of 90% or higher. The first task was to program a control script for this platform which would measure this movement and calculate the heartrate therefrom on the host PC. Once this script proved satisfactory in initial tests, a series of measurements would be carried out in an Anechoic chamber, where the amount of Radio Frequency noise would be minimalized. In the resulting script, 4 digital filters were designed, in the interest of inspecting how well various filters performed when attempting to remove noise due to Heart-Rate Variability. Of the 4 designs, 3 were tested at 2 different resolution settings, one filter was only tested at the lower resolution setting. 3 of the 7 variations proved successful under testing, at best achieving 5% Mean Absolute Percentage Error.**

## I. Introduction

This project title was first specified in consultation with and partly funded by the Automated Transport and Safety (ATS) department of Analog Devices, Inc. (ADI) Limerick. The goal come the end of the project is to demonstrate a system which is capable of measuring the heart rate of passengers in a car (one or multiple) with an accuracy of 90%. This system would find its place in intelligent vehicles, more specifically in the automated emergency services contact system of intelligent vehicles. At present certain vehicles are fitted with systems which contact the emergency services in the event of a collision, however if more sophisticated passenger monitoring systems were incorporated it would be feasible to send a more detailed report to the emergency services, informing them as to what equipment they will require on site. Furthermore, these systems can retain information from the cockpit of the car from before an incident. This information would then be compiled into a report detailing the condition of the driver/passengers leading up to an incident.

To aid this project ADI Limerick have provided an ADI *Demorad* 24GHz Radio Detection and Ranging (RADAR) platform, as well as a selection of their Intellectual Property (IP) which pertains to this research. The largest majority of the development for this project took place in their European Research and Development Centre, while a large amount of the testing and optimization took place on the campus of the University of Limerick. The IP provided by *ADI* was presented in both the Python 3.6 and MATLAB R2017b scripting languages. In both cases a style of Object-Oriented coding was employed, however during the initial setup of the project the MATLAB scripts ran far smoother and with fewer bugs, hence this was the platform chosen for the remainder of the project. Additionally, MATLAB provides an intuitive graphical user interface (GUI) builder, a tool which became very useful at later stages of the project such as testing and optimizing.

Sections III and IV of this report will be a quick look at the device being used, a detailing of the code being used as well as explanations behind the mathematics found therein and a brief look at the GUI designed.

## II. Literature Survey

The use of RADAR to measure heart and respiratory rates date back to the 1970's [1][2], which used commercially available waveguide X-band Doppler transceivers. In recent years, more research has been conducted to experiment with creating enclosed RADAR systems specifically with the goal of accurate heart-rate measurement, the most recent of which was in 2013 [3, 4]. These experiments achieved error rates of 1.7% and less given optimum range and angle to target for heart rate measurements. Other experiments were also conducted, and these will be listed in the references section of the report, however a popular design choice in these systems is the use of Doppler RADAR. This is a configuration whereby range to a target is ascertained by transmitting a Continuous Wave (i.e. the frequency does not change during transmission) and monitoring the phase angle difference between the transmitted and received signal. This methodology does offer exceptionally high resolution to range measurements, however the it presents challenges when it comes to measuring heart rates from multiple patients simultaneously. Høst-Madsen has shown that mathematically it is possible to measure heart rate from multiple subjects using this technique and has simulations to demonstrate this in the cited paper, however no actual tests were conducted. [3]

The use of Continuous Wave RADAR was not possible for this project, due to the limited IP provided. However, the antenna on the Demorad support Digital Beam Forming (DBF) quite well, which to date has not been applied to this application. DBF is an application of Frequency-Modulated Continuous Wave (FMCW) RADAR, which provides a range profile of the area in the device's field of view. Objects in the field of view are detected by monitoring the beat frequency (a.k.a. the baseband frequency) on all 4 receive channels. A high-level description of the code which achieved this is provided later in the paper, however in this field one generally states that the Demorad utilizes a "Multiple Input, Multiple Output" (MIMO) arrangement. Similar methodology has been used within ADI Limerick to develop a program which can track the movement of objects within the field of view of this device. Based on the theory, provided in section III, this project aims to extend that functionality to also read the

heartrate of the object moving through the field of view, assuming the object has physiological processes.

In recent years, others have investigated the potential of MIMO Doppler RADAR setups to measure cardiopulmonary movement, as well as the effectiveness of FMCW RADAR for the same purpose [5, 6]. This project will investigate the prospect of using MIMO FMCW RADAR to achieve cardiopulmonary movement measurement. Achieving this will require a detailed understanding of the chest wall movement, as detailed in section III.

## III. Theory and Mathematics

While it is fair to state that there are several contributing factors to the movement of a human's chest wall, the two of interest for this particular project are the effects of respiratory action (breathing) and cardiopulmonary movement (heartbeat). Much is known about how these processes affect chest wall movement, here will be a brief outlining of the relevant details as well as how these will affect the measurement-taking method used in this application.

Viewing the biological process of interest in the frequency domain will provide the most benefit, for reasons explained later in this section. Respiratory motion typically remains in the 0.1-0.8Hz range, with exceptions occurring when a person is under physical stress or subject to certain physiological conditions. Likewise, the cardiopulmonary movement typically remains within the 0.8-2Hz range, subject to external/individual conditions. The measured respiratory movement will be a far stronger signal than that of the heartbeat, however it is difficult to characterize. This is mainly due to there being multiple frequencies present in a single breath, all remaining within that 0.1-0.8Hz band. As respiration is of little interest in this project it is mostly removed by a type of band-pass filtering.

The remaining signal contains information about the heartbeat of the patient. Heartbeat is described as a "nearly periodic signal" [3], in that the period of each heartbeat can vary from on beat to the next. This is termed "Heartrate variability" (HRV). HRV can be modelled as a random process with strong periodicity [7]. While this characteristic of heartrate makes it somewhat easier to measure (via the application of band-pass filtering), one must keep in mind that the relative weakness of the signal can introduce considerable error in the measurement.

Bearing this in mind, multiple possible solutions exist to the same design problem. The solution which was chosen as the focus for this project is as follows: Use the RADAR platform to ascertain the range to the patient's chest wall. Monitor this movement over time and perform a Fast Fourier Transform (FFT) on this recorded movement. Band-pass the resulting power spectrum into the bands listed above and find the corresponding peaks in each band. The peak will correspond to the frequency of movement which is most highly present in each band; respiration and heartrate respectively. To counteract the likely measurement error a type of digital filter may be applied to the heartrate measurements, as is common in most heartrate measuring devices. The next step is to inspect the equipment at hand and begin working towards this solution.

## IV. Outline Design

### A. The ADI Demorad

**At the offset of this project there was no one device/platform available to the ATS group which completely met the requirements of the task.** To prevent any interruption to progress ADI Limerick provided a set of evaluation boards for the ADF4159, ADF5901 and ADF5904, a PLL, Transmitter and Receiver/Mixer chip respectively. During the first months the LabVIEW automation language was employed, testing various configurations of each device listed as well as for the ADAR7251 ADC provided on the Evaluation board, and making the first steps in developing software for the task at hand. However eventually custom firmware for the microprocessor being used (the ADSP-BF700 "Blackfin") was required, a task which according to experts is enough to dedicate a whole FYP to. At this point the supervisors on site at ADI worked to find a replacement setup for the evaluation kits.

During the first stages of the project, before the firmware issue was encountered, another group within the ATS department, the Applications group, had worked to get access to the new RADAR platforms being designed by ADI. The Demorad platform was one of the platforms ADI was given access to. The primary components of the platform were the very chips which comprised the evaluation setups the project began with, so the initial work done was translated from LabVIEW to MATLAB R2017b, the reasons for which are explained in the introduction. As of that point, which was late November 2017, a second Demorad platform was commissioned to this project as the first platform was damaged beyond repair. Figure 1 illustrates the platform, in the references section will be links to the datasheets for each of the primary chips mentioned in this section [8-12]

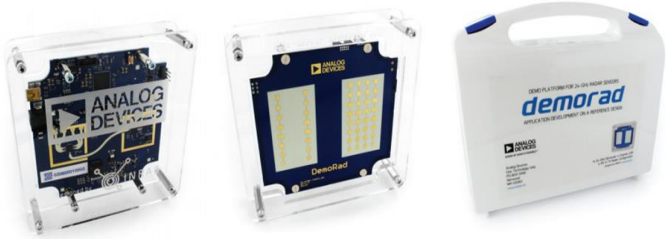



Figure 1: An image of the ADI Demorad 24GHz RADAR platform [13]

### B. The Algorithm

**This section will discuss (sometimes in brief and sometimes in detail) the various elements that make up the program which has been designed to run on the Demorad**. Certain steps cannot be detailed due to copyrighting of ADI IP; however, every operation of major interest will be detailed in as much as can be done.

Much of the initial code used cannot be shared as it details certain classes needed to make the code functional and other copyrighted code belonging to ADI, however what follows in this section will be a high-level overview of the algorithm designed for this project.

Figure 2 shows a simple chain of events which occurs at the beginning of the script. Once the connection to the board is established, the FMCW frequency ramp is initialised, space is stored in the computer's memory for the

measurement info to follow, the maximum duration of the test is saved into memory and the measurement then begins..

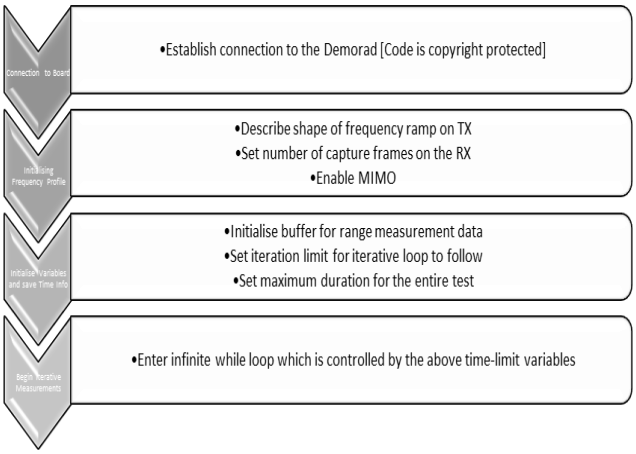



Figure 2: Initial steps to Heart Rate measurement

The following while loop contains nearly all the main functionality of this script. One iteration of the while loop corresponds to one measurement of heart rate. Note however that one measurement of heart rate requires a predefined number of data collections from the board. This number of measurements impacts the resolution of the measurement. As part of the FFT function in MATLAB the sampling rate of the range measurement is required. This is initialised in the first step of the loop. Once a stream of the beat signal, who's length is defined previously as 256 samples long, is captured the waveform is filtered using the "smoothdata" command. To take full advantage of the MIMO functionality the beat signal data must first be organised into information received from Tx1 and Tx2. This organised information can be used to create a range profile of the space in front of the device.

```
for Idx = 1:(Cfg.NrFrms-1)
    tic
    if Idx > 1
        Ts = ((timerValStop) + Ts)/2;   % Sampling period for the range profile
    else
        Ts = 0.1;       % Initial guess for Ts, likely inaccurate
    end

    Data            =   Brd.BrdGetData();
    Data = smoothdata(Data);

    %-------------------------------------------------------------------
    % Creat virtual 8 channels
    %-------------------------------------------------------------------
    DataTx1         =   Data(1:Cfg.N,:);
    DataTx2         =   Data(Cfg.N+1:end,:);
```

Once the range profile is created and the data is stored into the variable "JNorm" a differencing algorithm is run on it. This effectively copies the data into a new variable, "Frame_Odd" or "Frame_Even", and subtracts the frame captured on the previous run of the iterative loop from the newest frame. In doing so any static object will be ignored and only moving targets will be tracked.

```
%-------------------------------------------------------------------
% Copy Range profile magnitudes to frames then preform differencing
% algorithm. Provides movement information
%-------------------------------------------------------------------
if Idx == 1
    Frame_Odd = JNorm;
    Diff = JNorm;
elseif (rem(Idx,2) == 0)
    Frame_Even = JNorm;
    Diff = Frame_Even - Frame_Odd;
else
    Frame_Odd = JNorm;
    Diff = Frame_Odd - Frame_Even;
end
```

Once complete, this differenced range profile then has a threshold applied to remove some low-level noise. Finally, the software focuses on the peak value found in that profile. This is taken to be the patient. Once found the software will make note of the current range to the target. On the next run of the for loop another measurement will be taken, and so on and so forth. This range information is then saved into a buffer for digital signal processing (DSP) provided later on.

```
%-------------------------------------------------------------------
% Filter data based on size of movement (Threshold) and based
% on location of movement relative to the device
%-------------------------------------------------------------------
maximum = -10000; % Sufficiently small, most magnitudes will be <0
Range_Of_Interest = 0;
for x = 1:233
    for k = 1:256
        if abs(Diff(x,k)) < 0.5 % Threshold the detected movement
            Diff(x,k) = 0;
        end
        if Diff(x,k) > maximum
            maximum = Diff(x,k);
            Range_Of_Interest = x;
        end
    end
end

%-------------------------------------------------------------------
% Measurement of interest. Save to first index of buffer
%-------------------------------------------------------------------
buffer = circshift(buffer,1);
buffer(1) = vRangeExt(Range_Of_Interest,1);
```

An FFT is then performed on the differenced range profile. The FFT data is searched for the respiratory and cardiopulmonary information. Note the respiration data is only a rough estimate as it is not the focus of this project. The only difference with the heart rate measurement is that one of multiple filter designs will be applied to the measurements to account for HRV. This is in fact how many optical heart rate trackers operate, such as the Sp02 Assistant device which will be used as a reference in the testing procedure. The exact variations of filter being tested will be listed and explained in full detail under the Results section.

```
P2 = fft(buffer);
P1 = P2(1:((length(buffer))/2)+1);
P1(2:end-1) = 2*P1(2:end-1);
P1 = smoothdata(P1);
f = double((1/Ts)/(length(buffer)))*double((0:((length(buffer))/2)));

%-------------------------------------------------------------------
%Find the HR info in here
%-------------------------------------------------------------------
HR_is_in = real(P1(int32(0.8*(Ts*(Cfg.NrFrms-1))):int32(2*(Ts*(Cfg.NrFrms-1))))); % 0.8Hz - 2Hz
[peaks, freq] = findpeaks(HR_is_in);

Highest_peak = max(peaks);
for idx = 1:length(peaks)
    if peaks(idx) == Highest_peak
        break
    end
end
```

The remainder of the code is simply controlling the timing of the program, printing the results of the above code to the GUI and saving the information into an Excel sheet. A simplified visualisation of this loop is given in Figure 3.


Figure 3: Chain of events inside the main While loop of the code

### C. *The Graphical User Interface (GUI)*

**Figure 4 is a screenshot of the Graphical User Interface developed in MATLAB R2017B's graphical user interface development environment (guide).** In the lower right corner is the differenced range profile generated using code listed above. In the upper right corner is the plot of the FFT performed on the movement data gathered over time. In the upper left corner is the control panel, where the user inputs the number of measurements required (if known), the resolution of the FFT desired (which impacts both the accuracy of the measurement and the time taken for a single measurement by linearly varying the number of frequency bins present in the FFT), the maximum duration for the entire test in minutes and a checkbox which will allow the algorithm to run continuously for a maximum time entered in the maximum duration variable. Also, in the control panel is the output terminals for both the respiration and heartrate measurements. Below the control panel is a string input box, allowing the user to specify the save directory of the measurement files. During testing only two resolution settings were attempted: 0.1Hz and 0.05Hz (meaning that during any test there were either 10 or 20 bins per Hz).

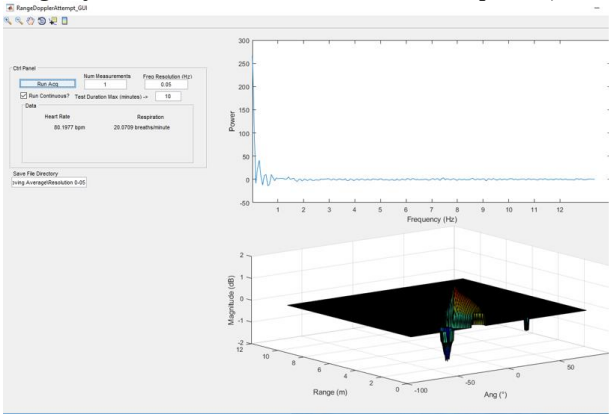


Figure 4: The GUI developed to allow easy use and monitoring of the above detailed algorithm.

## V. Experiment Outline and Setup

With the application designed, the next task is to design a test procedure with acquires all relevant data with maximum efficiency. Given that this experiment requires testing on a live human subject, all safety requirements as outlined in Ireland's microwave transmission legislation must be met. The maximum power level permitted for unlicensed 24GHz microwave transmission is 1mW (0 dBm), though most domestic appliances are well below this limit. When checked on site at ADI's European R&D Centre the Demorad's peak power output averaged at roughly -10dBm. Adhering to this power limit also keeps the subject safe during testing, as the transmitted signals do not have the power needed to penetrate skin.

Found under Appendix A is the experiment outline as submitted to the Department of Electronic and Computer Engineering, UL requesting the use of the Anechoic chamber on site. All parameters and risks were listed and evaluated in the conclusion of this document. In the conclusion section of this document is listed a recommendation of how best to spread out the experiments across 3 weeks, however due to time constraints it became more feasible to lump the experiment runs into 2 days. Given this reduced timeframe the experiment setup had to allow for fast construction and dismantling in between experiment runs.

In Figure 5 is a simplified diagram of the ideal setup. Both the Demorad and the reference fingertip measurement will send their measurements to their respective software on the host PC (the reference measurement sends to SpO2 Manager, the Demorad to MATLAB). After each experiment the data will be labelled and marshalled, as well as sorted into a designated folder for easy access. At a later date the information is compiled into a series of Excel sheets where comparison and analysis of performance can be conducted.

This experiment layout strictly follows the layout of a "Method Comparison Study", wherein a new measurement acquisition method is compared to a "Gold Standard" measurement. This type of study comes with its own typical evaluation method; however, this study could also easily be classified as time-series analysis, which typically involves different statistical evaluations. These will all be discussed under the Discussion section

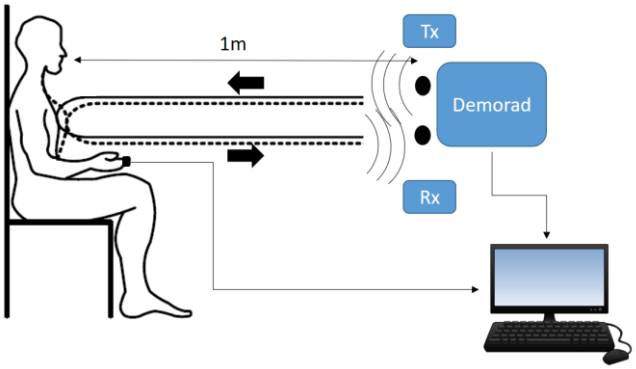



Figure 5: Diagram of the Experiment Setup

From this simplified schematic the experiment can be built in the Anechoic chamber. Here the environment

becomes more challenging, as source of noise and/or error must be noted, and their impact evaluated. Finding these sources was relatively straightforward: the MATLAB script was run with no subject in the chamber. What was noted was a very low level "hum" on the Rx channel. After some quick experimentation, it was found that the transformer connected to the Demorad's power supply was emitting very low-level EM noise. To account for this, the real setup (as seen in Figures 6-8) had the transformer placed behind the Demorad. Given that the chamber effectively cancelled reflective noise, this reduced the hum to such a low level as to be insignificant. The extension cable which the Demorad drew power from could not be moved behind the device, however its noise contribution was even lower than that of the transformer, so this source of error was allowed. Worth noting also was the length of USB cable present in the chamber. Each device was connected to a roughly 4.5m long USB cable, which under some circumstances can behave as an antenna. No noise was detected when the cables were in direct view of the Demorad however, so these were not taken to be sources of error. Out of paranoia the lights were also turned off during testing. No additional noise was detected from their being turned on, however given that they were fluorescent lights every precaution was taken.

No (extra) metal could be allowed into the chamber, so seating the subject and mounting the RADAR become the next setup choice. Inside the chamber were stacks of foam, so the heavier stack was made into a stool for the subject, the lighter one used as the stand for the Demorad. The walkway in the chamber had a small step down partway into the chamber, so to meet the ideal setup the subject's stool was placed 1m away from the step on the lower part of the walkway. The stand for the RADAR was on the edge of the step on the higher part of the walkway. This ensured that the Demorad would be at chest height during the experiment.

Figure 6: Anechoic Chamber Setup – View 1

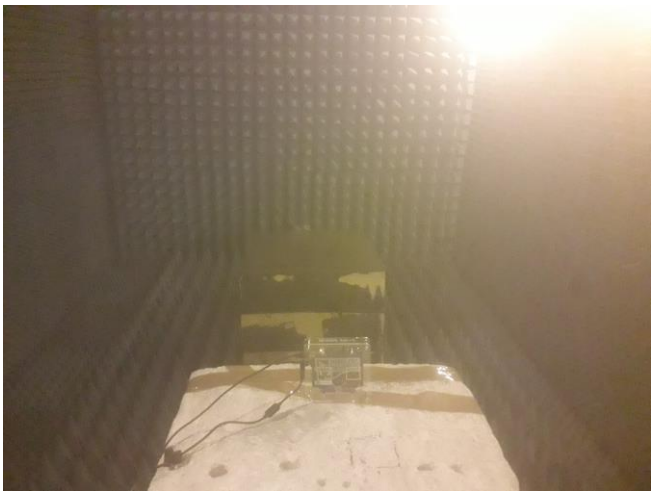


Figure 7: Anechoic Chamber Setup – View 2

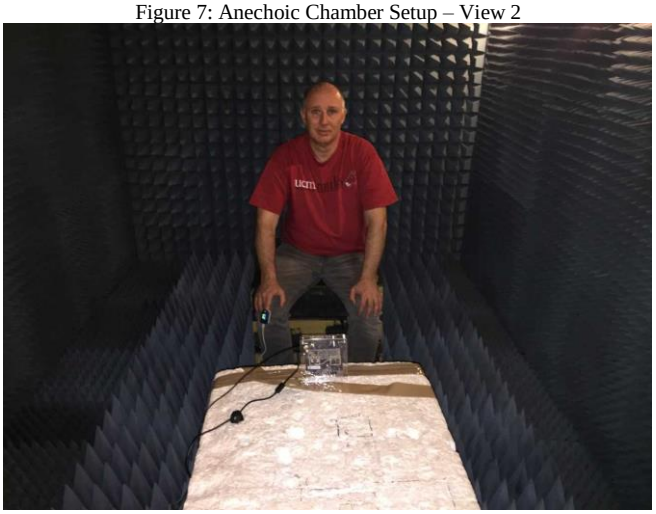


Figure 8: Anechoic Chamber Setup – Featuring Volunteer Dr. Strunz

Finally, the PC must be organised such that the running of the experiment was as smooth as possible. There was limited space outside the chamber for a full setup, so a makeshift desktop setup was put together with the screen lying flat on the table and the keyboard and mouse located on either side of the screen. Luckily only a few keystrokes and button clicks were required to run the experiment, so this setup proved effective. In Figure 11 is a sample screenshot from an experiment mid-run.


Figure 9: Anechoic Chamber Setup – PC arrangement

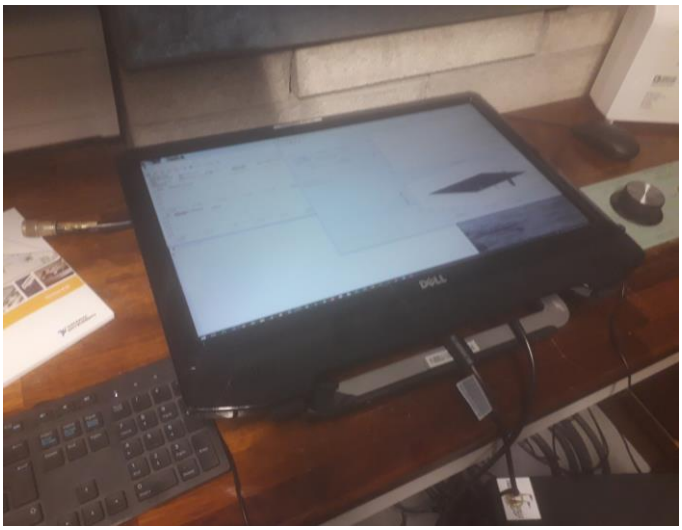


Figure 10: Anechoic Chamber Setup – Desktop Layout

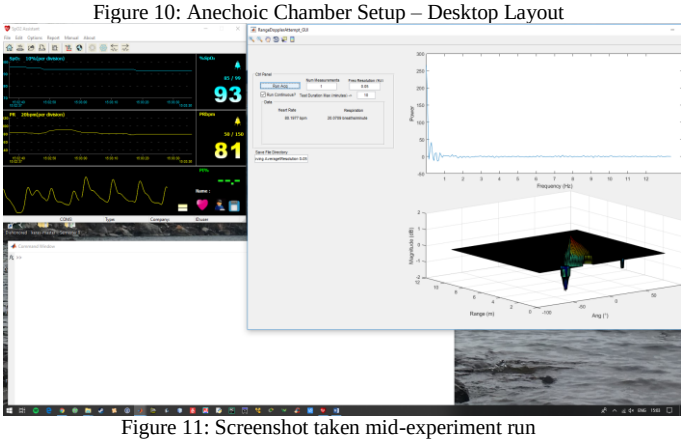



Figure 11: Screenshot taken mid-experiment run

## VI. Results

At multiple points during this section reference will be made to various figures and tables (Figures 12-18, Tables 1-7). These are to be found at the end of this section given their own dedicated pages. Four designs were engineered and then test, and as will be seen two of these designs demonstrate high accuracies under test conditions, while two of them show little promise.

For the purposes of defining a passing and a failing design it is best to define what error metrics constitutes a pass, however given that this experiment can fall under 2 different categories of test, different metrics for error exist. During the initial data compilation stage 3 metrics were taken: Maximum Error, Average Error and Standard Deviation, where each value is given as a percentage. The "Average Error" calculated for each design actually corresponds to a value known as the Mean Percentage Error (MPE) for that design. A more comprehensive detailing of how best to define a success or failure is in the Discussion section, however any one of these metrics remain mostly useless without the other two, hence all three must be measured.

The first design attempted was the 8-Tap Moving Average Filter. The operation of the filter is simple in idea: have a buffer which is 8 samples long and completely refill it after every measurement. This in practice means waiting for 8 new measurements to arrive before operating on the newest data. This idea originally seemed logical, by taking such a large sample of data and performing linear averaging, the overall effect of HRV would be nullified. However, the failing of this design came from its very low measurements/minute count. Each individual measurement with this design took around 64 seconds to update (even at the reduced resolution of 0.1Hz), meaning that the algorithm never had a chance to account for the error in measurement, which explains the long flat sections of the experimental plot in Figure 12. Every error metric proved far too high to be acceptable, as read from Table 1. Maximum error and average error show this design to be a failure. Given that this resolution setting was the fastest of the two specified, the slower resolution setting was not attempted for this design.

Second was the Gaussian Filter design. Slightly modified from the previous 8-Tap design, now the buffer contains the newest measurement and the 7 previous measurements. This is inherently faster, as there is no need to wait for the buffer to refill. Once full, simply apply a Gaussian filter kernel to the data. Given HRV's Gaussian nature this seemed appropriate. Under testing it proved volatile at best, responding surprisingly poorly to the presence of noise. It is likely that by altering the variance of the kernel this response could have been improved, however by examining both the 0.1Hz resolution and the 0.05Hz resolution experiment data, neither setting seemed to improve performance drastically. Oddly the average error for this design was the lowest recorded for any design tested, however both the maximum error and standard deviation of said error show this to be a volatile design at best. Figures 13-14 show the experimental data at their respective resolution settings, Tables 2-3 contain the error metrics for each test.

The failing of the 8-Tap Moving Average filter was its low measurement/minute count. Increasing this count should, in theory, improve the performance of the design. The operation of the 4-Tap Moving Average filter is exactly as before: a buffer stores the last 4 measurements taken, and these measurements are linearly averaged. Quick glances at figures 15-16 show immediate improvements across the board with respect to the error characteristics of the design. The tracking ability of the design also vastly improved. Tables 4-5 also show that these designs were the first to pass the experiment based on the error criteria outlined in the abstract. Just how well they passed remains to be seen. Given that this design still requires that the buffer be filled before updating its output, there remains the long flat sections of the experimental plots, although they are half the length of the flat sections in the 8-Tap design. The question remains, if greater speed can be achieved would it improve performance?

This fourth design attempts to answer that question. The Gaussian filter was by far the fastest design thus far, updating its output once per measurement cycle (roughly once per 8 seconds on the PC used). The 1-Tap sliding Window filter replaces the Gaussian filter kernel with a simple averaging kernel. The buffer remains 8 samples long, with the value at index 1 being the newest measurement. Providing the best compromise between speed and accuracy, this design performed very well under both resolution settings, as seen in Figures 17-18. Likewise, Tables 6-7 confirm that these designs have the best error performance of any filter type tested. Note the average error recorded for

the 0.05Hz resolution setting was negative. This only means that the experimental readings were lower on average than the reference measurements.

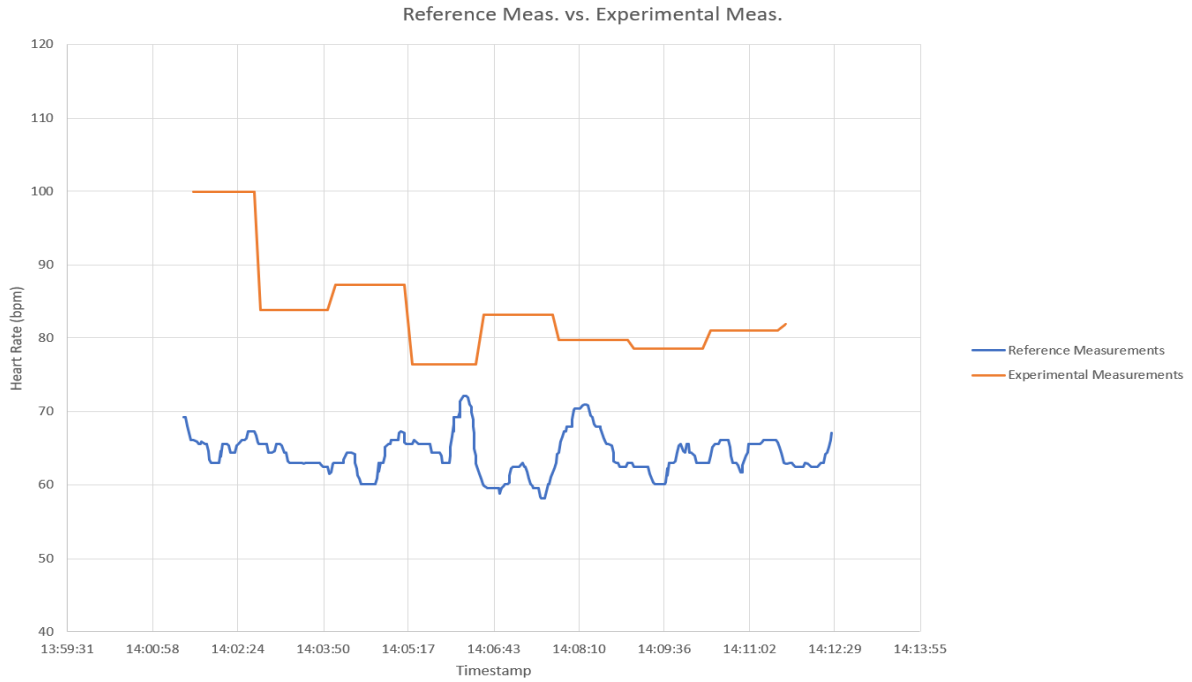



Figure 12: 8-Tap Moving Average Filter – Resolution 0.1Hz

**Table 1: 8-Tap Moving Average Filter – Resolution 0.1Hz**

| Maximum Error % | Average Error % | Standard Deviation % |
|---|---|---|
| 58.4161 | 30.249 | 11.7456 |

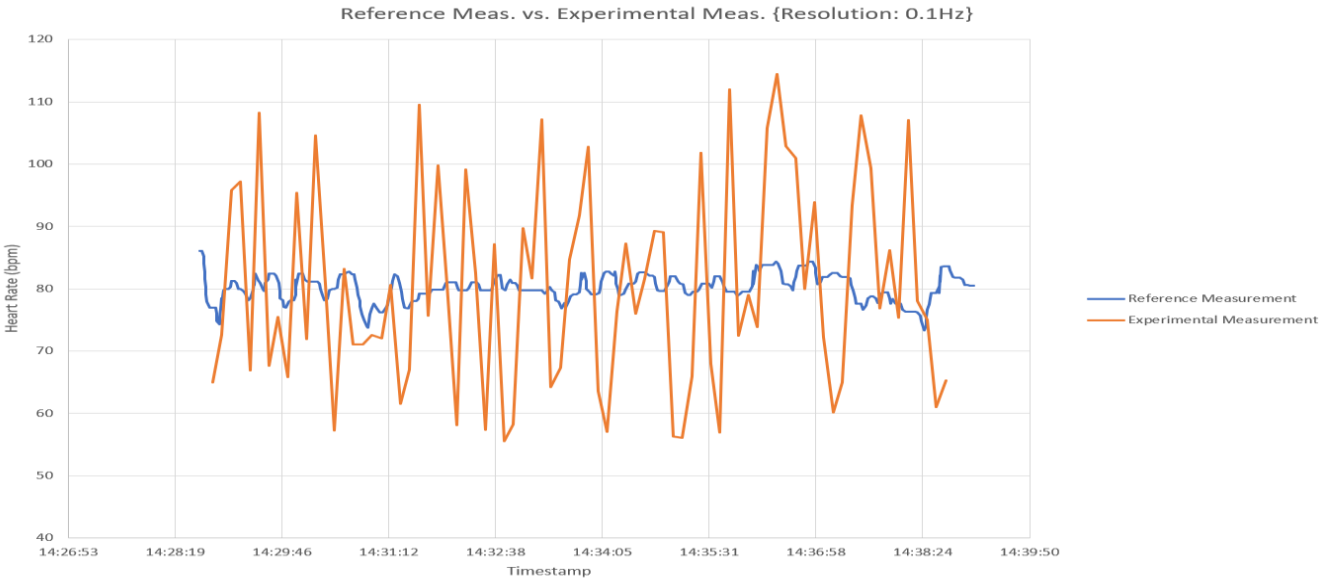



Figure 13: Gaussian Filter – Resolution 0.1Hz

**Table 2: Gaussian Filter – Resolution 0.1Hz**

| Maximum Error % | Average Error % | Standard Deviation % |
|---|---|---|
| 40.7561 | 0.2864 | 20.6596 |

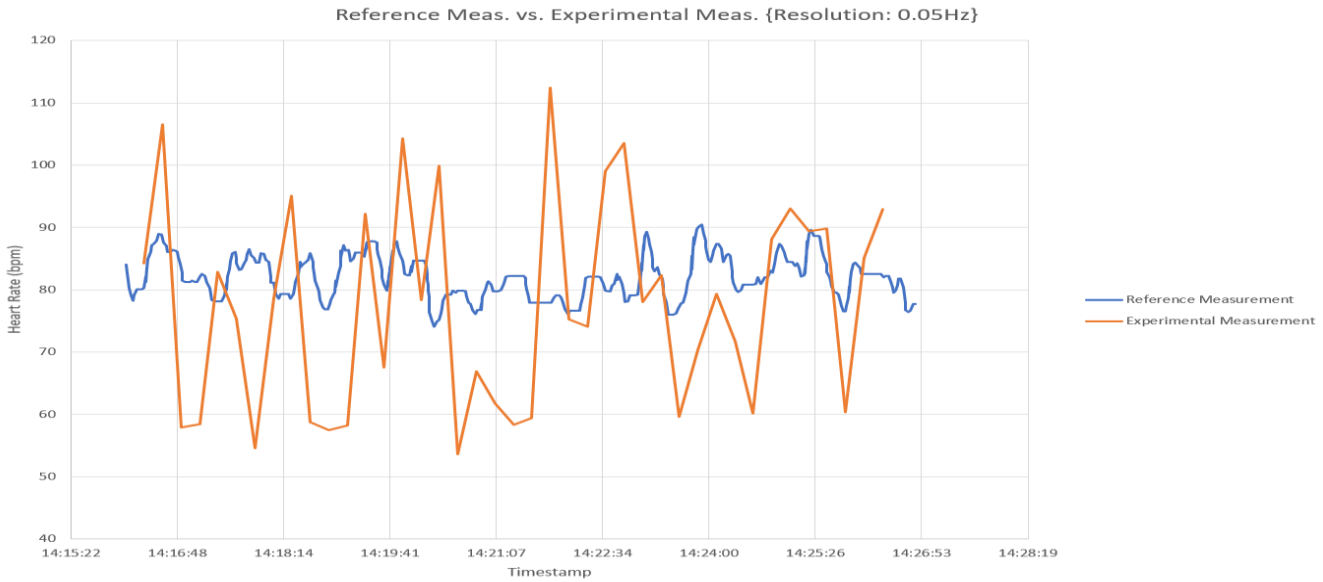



Figure 14: Gaussian Filter – Resolution 0.05Hz

**Table 3: Gaussian Filter – Resolution 0.05Hz**

| Maximum Error % | Average Error % | Standard Deviation % |
|---|---|---|
| 44.2768 | -5.6783 | 20.5086 |

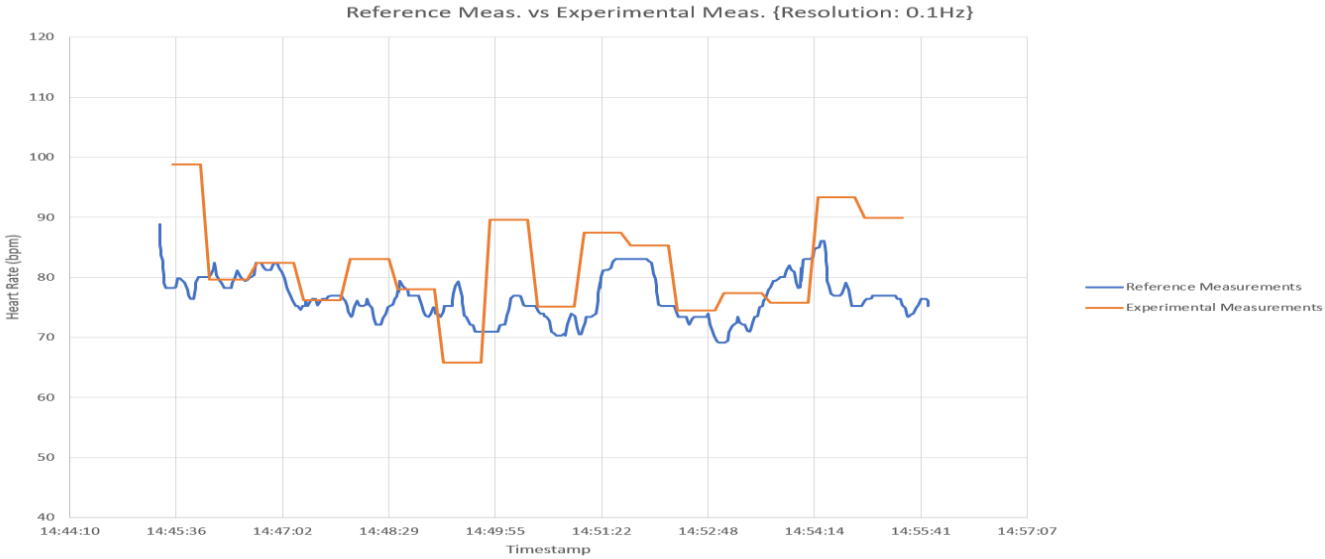



Figure 15: 4-Tap Moving Average Filter – Resolution 0.1Hz

**Table 4: 4-Tap Moving Average Filter – Resolution 0.1Hz**

| Maximum Error % | Average Error % | Standard Deviation % |
|---|---|---|
| 26.7827 | 6.9336 | 10.2765 |

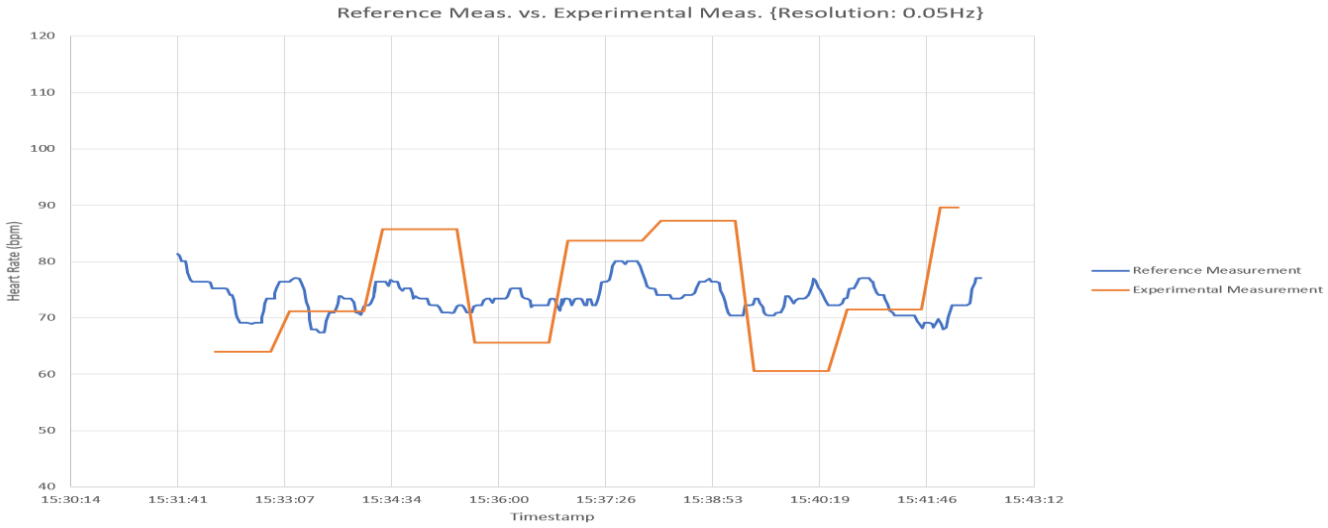



Figure 16: 4-Tap Moving Average Filter – Resolution 0.05Hz

**Table 5: 4-Tap Moving Average Filter – Resolution 0.05Hz**

| Maximum Error % | Average Error % | Standard Deviation % |
|---|---|---|
| 29.5172 | 1.7969 | 13.9173 |

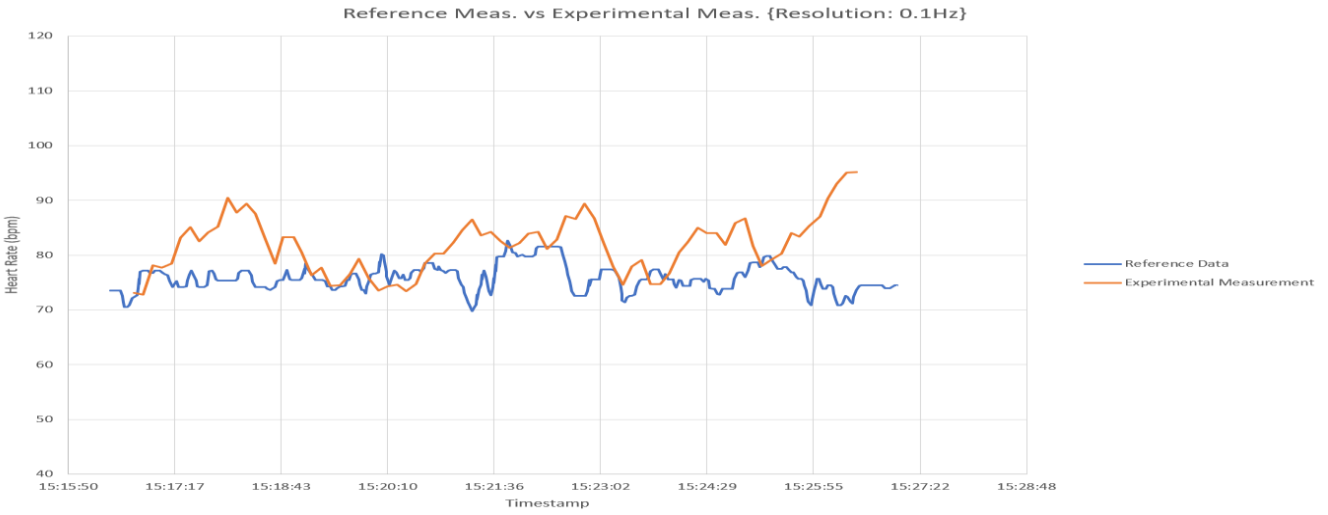



Figure 17: 1-Tap Sliding Window Filter – Resolution 0.1Hz

**Table 6: 1-Tap Sliding Window Filter – Resolution 0.1Hz**

| Maximum Error % | Average Error % | Standard Deviation % |
|---|---|---|
| 33.7621 | 7.8724 | 8.3567 |

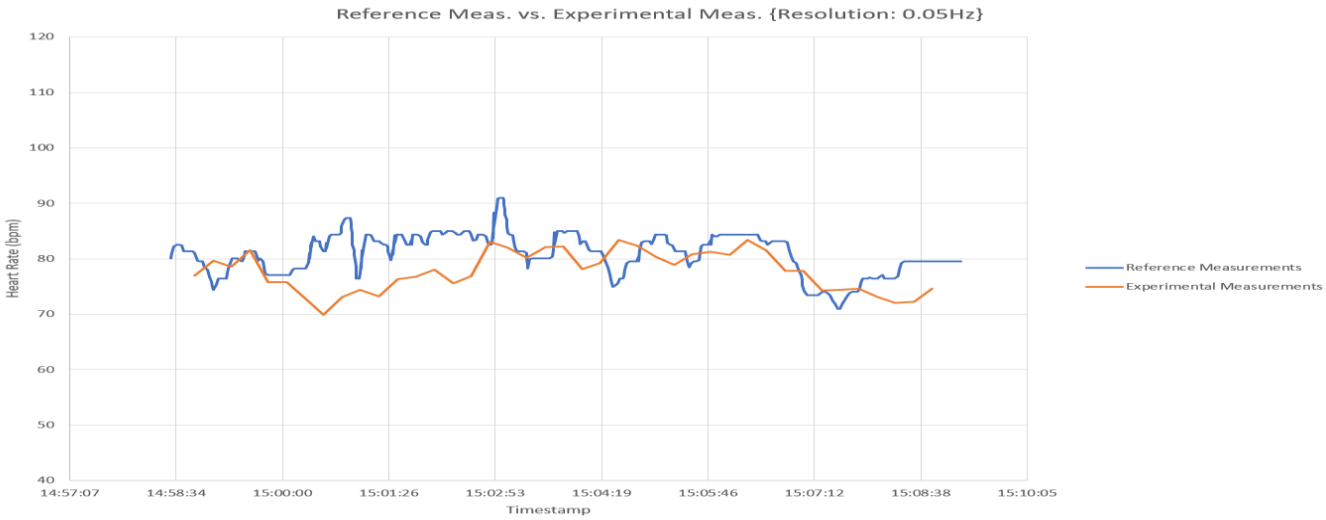



Figure 18: 1-Tap Sliding Window Filter – Resolution 0.05Hz

**Table 7: 1-Tap Sliding Window Filter – Resolution 0.05Hz**

| Maximum Error % | Average Error % | Standard Deviation % |
|---|---|---|
| 15.9021 | -3.715 | 5.334 |

## VII. Discussion

Before discussing each result set in any detail, it would first be prudent to examine the experiment types that this study falls under. As mentioned in the Experiment Outline and Setup section, this data gathered from this experiment can either be reviewed as time-series data or as a Method Comparison Study. The analysis performed under either data type provides very useful insights into how the experimental measurement performed under test conditions, however it would be simplest to choose a single data type and choose a single best design based on that type.

Method Comparison Studies essentially compare 1 or more new data acquisition types with a "Gold Standard" acquisition method. This gold standard is typically a method which has been tested and characterised extensively in the past, meaning the data gathered using this method is trusted to follow the true value so closely as to have negligible error. In this case, the fingertip optical measurement was used to form this reference. Typically, the data gathered from just such experiments is compiled into scatter plots, wherein one axis denotes the measurements taken using one method and the other axis denotes the other method (e.g. x = Gold Standard, y = Experimental Method). Plots of this nature tend to provide very useful information regarding the performance of the experimental method under various input types and over wide ranges of input values. Ideally the best-fit-line of this scattered data will come to x = y, indicating that on average both methods perform roughly the same under the same test conditions. Other common behaviours would include conical responses, where the experimental error gets larger as the input values get larger.

Time-series data is reviewed slightly differently. As in the Results section, data of this type is typically plotted over time, although usually not exactly this way. Time-series are more often than not used for predictions, in this example however it is only used as a measurement. There are multiple well-founded versions of error metrics for such data: Mean Absolute Deviation (MAD), Mean Square Error (MSE), Root Mean Square Error (RMS Error), Mean Absolute Percentage Error (MAPE) and so on. In the Results section the average error was treated as the primary error metric. This is similar in computation to the MAPE metric, however mean percentage error can become inaccurate if the error centres on zero despite having poor a standard deviation. MAD and MAPE are the most intuitive by design, the first being measured in beats/minute and the second being measured as a percentage. Both MSE and standard deviation are measure in $(\text{beats/minute})^2$, which loses meaning when comparing performance, given that it isn't an intuitive unit to understand. MAPE can come into difficulty if the reference measurement ever falls to 0, however in this example this is not a concern. Both MAD and MAPE provide unified expressions for the magnitude and spread of the error across the measurements, which provides essential characterisation data.

In deciding which representation would suit analysis best, two key points became apparent. First that the measurements from both devices all centre around roughly 80bpm, meaning that any scatter plots made would not have such a variety of data to work with and thereby limit the amount of information that can be read from the plot. Second that this design is still in its infancy, quite a bit more work can (and in fact should) be done before performing Method Comparison Study analysis. At this early stage it is more useful to the future of this research to represent this data as Time-series data and classify each filter design from worst to best using MAPE (given that success/failure is gauged using percentage errors). More on what the next steps in this project should be in the Further Research section. For reference, the formula for MAPE is given below:

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{Exp.-Ref.}{Ref.}\right|$$

In Table 8 is listed the MAPE for each design in order from worst to best. Interestingly, despite looking terrible, the Gaussian filter response was not the worst of the sampled designs. As would be expected the 4-Tap and 1-Tap designs performed the best, with the higher resolution setting performing better in 1-Tap case and worse in the 4-Tap case. Remembering that the outlying goal was to have an accuracy of 90% or higher (error being <10%), 3 designs manage to pass this test, while the remaining 4 fail by varying margins.

**Table 8: Rank – Ordering Filter Designs**

| Filter Design | Mean Absolute Percentage Error (%) |
|---|---|
| **8-Tap Moving Average (0.1Hz)** | 30.249 |
| **Gaussian Filter (0.1Hz)** | 17.1244 |
| **Gaussian Filter (0.05Hz)** | 17.7495 |
| **4-Tap Moving Average (0.05Hz)** | 12.0955 |
| **4-Tap Moving Average (0.1Hz)** | 9.247 |
| **1-Tap Sliding Window (0.1Hz)** | 8.6673 |
| **1-Tap Sliding Window (0.05Hz)** | 5.1508 |

## VIII. Conclusion

The degree of success or failure for any filter design largely depends on its speed of operation. There is little hope for improvement with the 8-Tap moving average filter. No matter what is done with it, it will always be the slowest design. Perhaps with some clever use of computing or with faster hardware it could become viable, but the data do not indicate any real potential for improved performance with this filter. The Gaussian filter is interesting. It certainly failed in these experiments, however close review of the figures show that the filter actually followed the data rather well, only with a far to volatile response. Perhaps some fine tuning would see an improvement in the performance. As seen in the code given in Appendix B, the Gaussian filter was left to run with its default value for $\text{sigma}^2$, perhaps attempting different values would prove useful. As they are, both the 4-Tap and 1-Tap moving average are ready to be pushed into the Further Research stage. While the experiments conducted satisfied the scope of this project, there is quite a lot of benefit to be gotten from continuing

this research further. So far as this project outline goes, 3 separate designs have been made which meet the requirements set out in the abstract.

## IX. Further Research

As far as further design work goes, a greater variation on filter type and design is certainly called for in any future experiments. The gaussian filter idea warrants further investigation, and all passing filter designs warrant optimisation and refining. The major choke-point of these designs was the speed of data acquisition, largely affected by the fact that the algorithm was always run on a host PC, rather than on the RADAR itself. The microprocessor on the board, The ADI Blackfin, is more than capable of handling the Digital Signal Processing carried out in MATLAB. The largest amount of benefit stands to be found in treating the Demorad as an application specific board and porting the algorithm over to the board itself.

After that, testing each design on a larger (potentially much larger) sample size would provide far more concrete evidence of the success/failure of the design as a whole. Also varying the conditions for each sample individual would provide useful data, potentially allowing for a Method Comparison Study to be undertaken once the design has been verified.

If all the afore mentioned work is carried out and this on-board algorithm continues to hold promise, the very last piece of this project would be to finally attempt to amalgamate the software which tracked multiple targets moving through the Demorad's field of view with this algorithm. Doing so successfully could push this project to being market ready, however there naturally would be quite a lot of legislation to go through before such a device could be sold on the market. Nevertheless, this is the defined end-goal of this project.

## X. Acknoledgements

First and foremost, my family deserve gratitude for proof reading my reports and listening to my practicing the presentation for this project.

My supervisor Colin Fitzpatrick deserves thanks for volunteering so readily to be my supervisor, and for providing much needed advice in technical report writing and presentation giving.

My ADI co-ordinator Padraic O'Reilly, for giving advice in how to properly run the experiment in the chamber and for assisting in keeping the project on course.

Dr. Bob Strunz for providing the Anechoic Chamber for my use, and for so willingly volunteering to be the test subject for this experiment.

The Analog Devices Automated Transport and Safety Design Evaluation lab for providing all necessary hardware, insight and assistance at every turn and general support in pursuit of this project.

## XI. Appendices

### A. *Anechoic Chamber Experiment Outline*

***Name:*** Oisín Watkins
***Student ID:*** 15156176
***Date:*** 08/10/2018
***Proposal Title:*** Heart Rate Measurements in the Anechoic Chamber.

Herein is listed the goals, parameters, settings and risks for an FYP experiment proposal in the Anechoic Chamber located in the main building, C-block, floor 0 of UL Campus. This missive is sent with the purpose of requesting permission to use the anechoic chamber for the experiment outlined below.
Within this document reference is made to "The Subject", meaning the volunteer who agreed to take part in this experiment. Bob Strunz has agreed to step in as the subject for this experiment. Given his experience using the chamber this also removes the need for a third supervisor to be present.

**Goals:**
The FYP title I am working towards is: "An Application of 24GHz RADAR as a means of Contactless Heart Rate Monitoring". Both the RADAR platform being used for this project as well as the software/firmware required to accomplish this task have been provided by Analog Devices Inc. Limerick. A working prototype is now fully engineered and ready to be tested. To provide ideal-scenario measurements an anechoic chamber is required. The goal of this experiment is to acquire data from live Human volunteers (both using my non-contact RADAR measurements and using an SpO2 fingertip measurement as a reference) from within a noiseless environment.

**Parameters:**
This experiment will measure only two metrics: The Heart Rate of the subject using the RADAR method, and the Heart Rate of the subject using the SpO2 fingertip sensor. No other devices will be necessary. The RADAR will be positioned 1m away from the subject at chest height with the antenna parallel to the subject's chest wall. The fingertip sensor will be attached to the subject's right index finger. Both devices will be connected to a laptop being operated immediately outside the chamber.

**Settings:**
The RADAR will be positioned as specified in the Parameters section. The subject will be seated on a plastic chair placed in the middle of the catwalk in the chamber. The subject will be facing in the direction of the door, with the RADAR between the subject and the door.

**Risks:**
Medical risks are at a minimum for this experiment. The maximum power level permitted for unlicensed 24GHz microwave transmission is 1mW (0 dBm), though most domestic appliances are well below this level. When last checked on site in Analog Devices Inc. the Demorad's

power output averaged at -10dBm (though no image of this was saved. This is worth measuring on campus if possible, to confirm).
There is also the possibility of damage to the foam lining of the chamber itself. Supervision from an experienced user of the chamber will be necessary to minimize this risk.

***Conclusion:***
Given the above experiment description I would like to request permission to use the anechoic chamber to run 10 experiments on the subject, who has been briefed on the nature of the experiment, each experiment lasting a maximum of 10 minutes. The most efficient organization of the total 100 minutes would be to spread them out over the course of 3 weeks at most.

*B. MATLAB Script*

```
function varargout =
RangeDopplerAttempt_GUI(varargin)
% RANGEDOPPLERATTEMPT_GUI MATLAB code for
RangeDopplerAttempt_GUI.fig
%      RANGEDOPPLERATTEMPT_GUI, by itself,
creates a new RANGEDOPPLERATTEMPT_GUI or
raises the existing
%      singleton*.
%
%      H = RANGEDOPPLERATTEMPT_GUI returns
the handle to a new RANGEDOPPLERATTEMPT_GUI
or the handle to
%      the existing singleton*.
%
%
RANGEDOPPLERATTEMPT_GUI('CALLBACK',hObject,ev
entData,handles,...) calls the local
%      function named CALLBACK in
RANGEDOPPLERATTEMPT_GUI.M with the given
input arguments.
%
%
RANGEDOPPLERATTEMPT_GUI('Property','Value',..
.) creates a new RANGEDOPPLERATTEMPT_GUI or
raises the
%      existing singleton*.  Starting from
the left, property value pairs are
%      applied to the GUI before
RangeDopplerAttempt_GUI_OpeningFcn gets
called.  An
%      unrecognized property name or invalid
value makes property application
%      stop.  All inputs are passed to
RangeDopplerAttempt_GUI_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools
menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response
to help RangeDopplerAttempt_GUI

% Last Modified by GUIDE v2.5 24-Jan-2019
10:16:54

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',
mfilename, ...
                   'gui_Singleton',
gui_Singleton, ...
                   'gui_OpeningFcn',
@RangeDopplerAttempt_GUI_OpeningFcn, ...
                   'gui_OutputFcn',
@RangeDopplerAttempt_GUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback =
str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] =
gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT



% --- Executes just before
RangeDopplerAttempt_GUI is made visible.
function
RangeDopplerAttempt_GUI_OpeningFcn(hObject,
eventdata, handles, varargin)
% This function has no output args, see
OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a
future version of MATLAB
% handles    structure with handles and user
data (see GUIDATA)
% varargin   command line arguments to
RangeDopplerAttempt_GUI (see VARARGIN)

% Choose default command line output for
RangeDopplerAttempt_GUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
% UIWAIT makes RangeDopplerAttempt_GUI wait
for user response (see UIRESUME)
% uiwait(handles.figure1);



% --- Outputs from this function are returned
to the command line.
function varargout =
RangeDopplerAttempt_GUI_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output
args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a
future version of MATLAB
% handles    structure with handles and user
data (see GUIDATA)

% Get default command line output from
handles structure
varargout{1} = handles.output;
```

```
% --- Executes on button press in RunButton.
function RunButton_Callback(hObject,
eventdata, handles)
% hObject    handle to RunButton (see GCBO)
% eventdata  reserved - to be defined in a
future version of MATLAB
% handles    structure with handles and user
data (see GUIDATA)

%-----------------------------------------
------------------------------
% Include all necessary directories
%-----------------------------------------
------------------------------
CurPath = pwd();
addpath([CurPath,'/../../DemoRadUsb']);
addpath([CurPath,'/../../Class']);

%-----------------------------------------
------------------------------
% Define Constants
%-----------------------------------------
------------------------------
c0 = 3e8;

Brd     =   Adf24Tx2Rx4();

%-----------------------------------------
------------------------------
% Reset Board and Enable Power Supply
%-----------------------------------------
------------------------------
Brd.BrdRst();

%-----------------------------------------
------------------------------
% Software Version
%-----------------------------------------
------------------------------
Brd.BrdDispSwVers();

%-----------------------------------------
------------------------------
% Read calibration data
% Generate cal data for Tx1 and Tx2
%-----------------------------------------
------------------------------
CalDat          =   Brd.BrdGetCalDat();
CalDat          =   CalDat;
CalDatTx1       =   CalDat(1:4);
CalDatTx2       =   CalDat(5:8);

%-----------------------------------------
------------------------------
% Enable Receive Chips
%-----------------------------------------
------------------------------
Brd.RfRxEna();

%-----------------------------------------
------------------------------
% Configure Transmitter (Antenna 0 - 2, Pwr 0
- 100)
%-----------------------------------------
------------------------------
Brd.RfTxEna(1, 100);

%-----------------------------------------
------------------------------
% Configure Up-Chirp
% TRamp is only used for chirp rate
calculation!!
% TRamp, N, Tp cannot be altered in the
current framework
% Effective bandwidth is reduced as only 256
us are sampled from the 280 us
% upchirp.
% Only the bandwidth can be altered
%-----------------------------------------
------------------------------
Resolution =
str2double(get(handles.FreqRes,'String'));

Cfg.fStrt           =   24e9;
Cfg.fStop           =   24.25e9;
Cfg.TRampUp         =   280e-6;
Cfg.Tp              =   284e-6;
Cfg.N               =   256;
Cfg.NrFrms          =
int32(1/(0.05*Resolution)); % 0.05 is an
estimate for the sampling period of the for
loop below on this machine
Cfg.StrtIdx         =   0;
Cfg.StopIdx         =   2;
Cfg.MimoEna         =   1;

%-----------------------------------------
------------------------------
% Configure Mimo Mode
% 0: static activation of TX antenna no
switching
% 1: TX1/Tx2
%-----------------------------------------
------------------------------
Brd.RfMeas('Adi', Cfg);

%-----------------------------------------
------------------------------
% Read actual configuration
%-----------------------------------------
------------------------------
NrChn           =   Brd.Get('NrChn');
N               =   Brd.Get('N');
fs              =   Brd.Get('fs');

%-----------------------------------------
------------------------------
% Configure Signal Processing
%-----------------------------------------
------------------------------
% Processing of range profile
Win2D           =   Brd.hanning(N,2*NrChn-1);
ScaWin          =   sum(Win2D(:,1));
NFFT            =   2^12;
kf              =   (Cfg.fStop -
Cfg.fStrt)/Cfg.TRampUp;
vRange          =   [0:NFFT-
1].'./NFFT.*fs.*c0/(2.*kf);

Freq            =   [0:NFFT-1].'./NFFT.*fs;

RMin            =   0.5;
RMax            =   10;
```

```
[Val RMinIdx]    =    min(abs(vRange - RMin));
[Val RMaxIdx]    =    min(abs(vRange - RMax));
vRangeExt        =    vRange(RMinIdx:RMaxIdx);

% Window function for receive channels
NFFTAnt          =    256;
WinAnt           =    Brd.hanning(2*NrChn-1);
ScaWinAnt        =    sum(WinAnt);
WinAnt2D         =
repmat(WinAnt.',numel(vRangeExt),1);
vAngDeg          =    asin(2*[-
NFFTAnt./2:NFFTAnt./2-1].'./NFFTAnt)./pi*180;

% Calibration data
mCalData         =    repmat(CalDat.',N,1);

% Positions for polar plot of cost function
vU               =    linspace(-1,1,NFFTAnt);
[mRange , mU]    =    ndgrid(vRangeExt,vU);
mX               =    mRange.*mU;
mY               =    mRange.*cos(asin(mU));
VirtData         =    zeros(Cfg.N, 7);

mCalDatTx1       =
repmat(CalDatTx1.',Cfg.N,1);
mCalDatTx2       =
repmat(CalDatTx2.',Cfg.N,1);

%-----------------------------------------
---------------------------------
% Initialising data for the measurement
buffer, number of measurements,
% maximum duration of the test and the save
directories of all files
% generated.
%-----------------------------------------
---------------------------------
buffer = zeros(1,Cfg.NrFrms-1);

itr =
str2double(get(handles.NumMeas,'String'));
current = 0;

MaxDur = str2double(get(handles.TestDur,
'String'));
Time_of_Test = clock;
Time_of_Test_Int = int32(Time_of_Test);
Day_Hour_Minute_Second = strcat('
',num2str(Time_of_Test_Int(:,1)),'
',num2str(Time_of_Test_Int(:,2)),' ',
num2str(Time_of_Test_Int(:,3)),'
',num2str(Time_of_Test_Int(:,4)),'-
',num2str(Time_of_Test_Int(:,5)),'-
',num2str(Time_of_Test_Int(:,6)));
CurrentDuration = 0;

% FileInit = strcat('C:\Users\Oisin
Watkins\Desktop\FYP\Contactless HR\Matlab and
Python Scripts\Matlab AN24_07-
AH\Matlab\AdfTx2Rx4\AppNotes\Wave
Files\BeatSigData_',Day_Hour_Minute_Second);

HR_File = strcat(get(handles.SaveDirect,
'String'),'\HeartRateMeas_',Day_Hour_Minute_S
econd,'.xlsx');
Heart_Info_Array = [];
HR_buffer = [0];
Heart_Rate = 0;

%-----------------------------------------
---------------------------------
% Clear all data printed to the terminal and
begin measurement
%-----------------------------------------
---------------------------------
clc
while true
    for Idx = 1:(Cfg.NrFrms-1)
        tic
        if Idx > 1
            Ts = ((timerValStop) + Ts)/2;    %
Sampling period for the range profile
        else
            Ts = 0.1;       % Initial guess
for Ts, likely inaccurate
        end

        %-----------------------------------
---------------------------------
        % Record data for Tx1
        %-----------------------------------
---------------------------------
%         num = num2str(Idx);
%         FileName1 = strcat(FileInit,
'_Rx0_', num, '.wav');
%         FileName2 = strcat(FileInit,
'_Rx1_', num, '.wav');
%         FileName3 = strcat(FileInit,
'_Rx2_', num, '.wav');
%         FileName4 = strcat(FileInit,
'_Rx3_', num, '.wav');

        Data             =    Brd.BrdGetData();
        Data = smoothdata(Data);

        %-----------------------------------
---------------------------------
        % Creat virtual 8 channels
        %-----------------------------------
---------------------------------
        DataTx1          =    Data(1:Cfg.N,:);
        DataTx2          =
Data(Cfg.N+1:end,:);

        %-----------------------------------
---------------------------------
        % Generate data of virtual uniform
linear array
        % use mean of overlapping element
        %-----------------------------------
---------------------------------
        VirtData(:,1:3)      =
DataTx1(:,1:3);
        VirtData(:,4)        =
0.5*(DataTx1(:,4) + DataTx2(:,1));
        VirtData(:,5:end)    =
DataTx2(:,2:end);

        %-----------------------------------
---------------------------------
        % Calculate range profile including
calibration
        %-----------------------------------
---------------------------------
```

```
        RP          =
fft(VirtData.*Win2D,NFFT,1).*Brd.FuSca/ScaWin
;
        RPExt       =
RP(RMinIdx:RMaxIdx,:);

        %-----------------------------------
---------------------------------
        % calculate fourier transform over
receive channels
        %-----------------------------------
---------------------------------
        JOpt        =
fftshift(fft(RPExt.*WinAnt2D,NFFTAnt,2)/ScaWi
nAnt,2);

        %-----------------------------------
---------------------------------
        % normalize cost function
        %-----------------------------------
---------------------------------
        JdB         =   20.*log10(abs(JOpt));
        JMax        =   max(JdB(:));
        JNorm       =   JdB - JMax;
        JNorm(JNorm < -18)  =   -18;

        %-----------------------------------
---------------------------------
        % Copy Range profile magnitudes to
frames then preform differencing
        % algorithm. Provides movement
information
        %-----------------------------------
---------------------------------
        if Idx == 1
            Frame_Odd = JNorm;
            Diff = JNorm;
        elseif (rem(Idx,2) == 0)
            Frame_Even = JNorm;
            Diff = Frame_Even - Frame_Odd;
        else
            Frame_Odd = JNorm;
            Diff = Frame_Odd - Frame_Even;
        end

        %-----------------------------------
---------------------------------
        % Filter data based on size of
movement (Threshold) and based
        % on location of movement relative to
the device
        %-----------------------------------
---------------------------------
        maximum = -10000; % Sufficiently
small, most magnitudes will be <0
        Range_Of_Interest = 0;
        for x = 1:233
            for k = 1:256
                if abs(Diff(x,k)) < 0.5 %
Threshold the detected movement
                    Diff(x,k) = 0;
                end
                if Diff(x,k) > maximum
                    maximum = Diff(x,k);
                    Range_Of_Interest = x;
                end
            end
        end

        %-----------------------------------
---------------------------------
        % Measurement of interest. Save to
first index of buffer
        %-----------------------------------
---------------------------------
        buffer = circshift(buffer,1);
        buffer(1) =
vRangeExt(Range_Of_Interest,1);

        clc
        timerValStop = toc;
    end

    %-----------------------------------
---------------------------------
    % Preform FFT and plot to GUI inside a
try-catch
    %-----------------------------------
---------------------------------
    try
        P2 = fft(buffer);
        P1 = P2(1:((length(buffer))/2)+1);
        P1(2:end-1) = 2*P1(2:end-1);
        P1 = smoothdata(P1);
        f =
double((1/Ts)/(length(buffer)))*double((0:((l
ength(buffer))/2)));

        axes(handles.axes1)
        plot(f,P1);
        xticks([1:1:12]);
        xlabel('Frequency (Hz)');
        ylabel('Power');

        %-----------------------------------
---------------------------------
        % Print Differenced Range Profile for
context
        %-----------------------------------
---------------------------------
        axes(handles.axes2)
        surf(vAngDeg, vRangeExt, Diff);
        xlabel('Ang (°)');
        ylabel('Range (m)');
        zlabel('Magnitude (dB)');
        colormap('jet');

        %-----------------------------------
---------------------------------
        %Find the breath info in here
        %-----------------------------------
---------------------------------
        Breath_is_in =
real(P1(int32(0.2*(Ts*(Cfg.NrFrms-
1))):int32(0.8*(Ts*(Cfg.NrFrms-1)))+1)); %
0.2Hz - 0.8Hz
        [peaks, freq] =
findpeaks(Breath_is_in);

        Highest_peak = max(peaks);
        for idx = 1:length(peaks)
            if peaks(idx) == Highest_peak
                break
            end
        end

        Respiratory_Rate = f(freq(idx));
```

```
        %-------------------------------------
-------------------------------------
        %Find the HR info in here
        %-------------------------------------
-------------------------------------
        HR_is_in =
real(P1(int32(0.8*(Ts*(Cfg.NrFrms-
1))):int32(2*(Ts*(Cfg.NrFrms-1))))); % 0.8Hz
- 2Hz
        [peaks, freq] = findpeaks(HR_is_in);

        Highest_peak = max(peaks);
        for idx = 1:length(peaks)
            if peaks(idx) == Highest_peak
                break
            end
        end

          % 8 Beat Sliding Window
%         if HR_buffer(1) == 0
%             HR_buffer(1) =
f(int32(freq(idx)) +
int32(0.8*(Ts*(Cfg.NrFrms-1)))); % place new
data at 1st index
%         elseif length(HR_buffer) <= 8
%             HR_buffer(length(HR_buffer)+1)
= 0; % add new index in array
%             HR_buffer =
circshift(HR_buffer,1); % shift data over 1
index
%             HR_buffer(1) =
f(int32(freq(idx)) +
int32(0.8*(Ts*(Cfg.NrFrms-1)))); % place new
data at 1st index
%         else
%             HR_buffer =
circshift(HR_buffer,1); % shift data over 1
index
%             HR_buffer(1) =
f(int32(freq(idx)) +
int32(0.8*(Ts*(Cfg.NrFrms-1)))); % place new
data at 1st index
%         end

%-----------------------------------------
-------------------------------
% Choose one filter design out of the two
given below if using sliding
% window
%-----------------------------------------
-------------------------------
%         Heart_Rate = mean(HR_buffer); %
implement moving average filter

        % Gaussian Filter Design
%         HR_buffer = smoothdata(HR_buffer,
'gaussian'); % apply gaussian filter to
sliding window
%         Heart_Rate = HR_buffer(1); % Take
heart rate as newest measured value
%-----------------------------------------
-------------------------------

        % 8/4 Tap Moving Average Filter
Design
        if HR_buffer(1) == 0
            HR_buffer(1) = f(int32(freq(idx))
+ int32(0.8*(Ts*(Cfg.NrFrms-1)))); % place
new data at 1st index
            if Heart_Rate == 0
                Heart_Rate = HR_buffer(1);
            end
        elseif length(HR_buffer) < 4 % Choose
either 8 or 4 here for length of buffer
            HR_buffer(length(HR_buffer)+1) =
0; % add new index in array
            HR_buffer =
circshift(HR_buffer,1); % shift data over 1
index
            HR_buffer(1) = f(int32(freq(idx))
+ int32(0.8*(Ts*(Cfg.NrFrms-1)))); % place
new data at 1st index
            %=> Do not update Heart_Rate
        else
            Heart_Rate = mean(HR_buffer); %
take average of previous 8 measurements
            HR_buffer = [0]; % Reinitialise
HR_buffer for next 8 measurements
        end

        %-------------------------------------
-------------------------------------
        % Print data
        %-------------------------------------
-------------------------------------
        clc
        Respiratory_Rate_Str =
num2str(Respiratory_Rate*60);
        Heart_Rate_Str =
num2str(Heart_Rate*60);

        info1 = strcat(Heart_Rate_Str,'
bpm');
        info2 = strcat(Respiratory_Rate_Str,'
breaths/minute');

        set(handles.HR_Out,'String',info1);
        set(handles.Res_Out,'String',info2);
% Printing info to the GUI
    catch
        info1 = 'No Heart Detected';
        info2 = 'No Breathing Detected';

        set(handles.HR_Out,'String',info1);
        set(handles.Res_Out,'String',info2);
% Printing error to the GUI
    end

    %-------------------------------------
-------------------------------------
    % Control Number of Measurements and save
info to array
    %-------------------------------------
-------------------------------------
    current = current+1;

    RunContinuous = get(handles.RunCtnd,
'Value');
    CurrentTime = clock;
    CurrentDuration = (CurrentTime(1,4) +
(CurrentTime(1,5)/60) +
(CurrentTime(1,6)/3600)) - (Time_of_Test(1,4)
+ (Time_of_Test(1,5)/60) +
(Time_of_Test(1,6)/3600));
```

```
    Heart_Info_Array(current,1:6) =
CurrentTime(1:6); %Saving a timestamp
%     Heart_Info_Array(current,7) =
str2num(strcat(num2str(CurrentTime(4)),':',nu
m2str(CurrentTime(5)),':',num2str(CurrentTime
(6))));
    Heart_Info_Array(current,7) =
Heart_Rate*60; %Saving info to array

    if RunContinuous
        itr = itr+1;
    end

    if current == itr || CurrentDuration >=
(MaxDur/60)
       break
    end

end
%-----------------------------------------
---------------------------------
% Reset board and display reason for ceasing
the measurement
%-----------------------------------------
---------------------------------
Brd.BrdRst();
clc
if CurrentDuration >= (MaxDur/60)
    disp('Timed Out');
else
    disp('Measurement Complete');
end
%-----------------------------------------
---------------------------------
% Save data into Excel
%-----------------------------------------
---------------------------------
xlswrite(HR_File, Heart_Info_Array);

function NumMeas_Callback(hObject, eventdata,
handles)
% hObject    handle to NumMeas (see GCBO)
% eventdata  reserved - to be defined in a
future version of MATLAB
% handles    structure with handles and user
data (see GUIDATA)

% Hints: get(hObject,'String') returns
contents of NumMeas as text
%        str2double(get(hObject,'String'))
returns contents of NumMeas as a double

% --- Executes during object creation, after
setting all properties.

function NumMeas_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to NumMeas (see GCBO)
% eventdata  reserved - to be defined in a
future version of MATLAB
% handles    empty - handles not created
until after all CreateFcns called

% Hint: edit controls usually have a white
background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in RunCtnd.
function RunCtnd_Callback(hObject, eventdata,
handles)
% hObject    handle to RunCtnd (see GCBO)
% eventdata  reserved - to be defined in a
future version of MATLAB
% handles    structure with handles and user
data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle
state of RunCtnd

function FreqRes_Callback(hObject, eventdata,
handles)
% hObject    handle to FreqRes (see GCBO)
% eventdata  reserved - to be defined in a
future version of MATLAB
% handles    structure with handles and user
data (see GUIDATA)

% Hints: get(hObject,'String') returns
contents of FreqRes as text
%        str2double(get(hObject,'String'))
returns contents of FreqRes as a double

% --- Executes during object creation, after
setting all properties.
function FreqRes_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to FreqRes (see GCBO)
% eventdata  reserved - to be defined in a
future version of MATLAB
% handles    empty - handles not created
until after all CreateFcns called

% Hint: edit controls usually have a white
background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function TestDur_Callback(hObject, eventdata,
handles)
% hObject    handle to TestDur (see GCBO)
% eventdata  reserved - to be defined in a
future version of MATLAB
% handles    structure with handles and user
data (see GUIDATA)

% Hints: get(hObject,'String') returns
contents of TestDur as text
%        str2double(get(hObject,'String'))
returns contents of TestDur as a double


% --- Executes during object creation, after
setting all properties.
```

```
function TestDur_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to TestDur (see GCBO)
% eventdata  reserved - to be defined in a
future version of MATLAB
% handles    empty - handles not created
until after all CreateFcns called

% Hint: edit controls usually have a white
background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function SaveDirect_Callback(hObject,
eventdata, handles)
% hObject    handle to SaveDirect (see GCBO)
% eventdata  reserved - to be defined in a
future version of MATLAB
% handles    structure with handles and user
data (see GUIDATA)

% Hints: get(hObject,'String') returns
contents of SaveDirect as text
%        str2double(get(hObject,'String'))
returns contents of SaveDirect as a double



% --- Executes during object creation, after
setting all properties.
function SaveDirect_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to SaveDirect (see GCBO)
% eventdata  reserved - to be defined in a
future version of MATLAB
% handles    empty - handles not created
until after all CreateFcns called

% Hint: edit controls usually have a white
background on Windows.
%       See ISPC and COMPUTER.
if ispc &&
isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

## XII. References


[1] J. C. Lin, J. Kiernicki, M. Kiernicki, P. B. J. I. T. o. M. T. Wollschlaeger, and Techniques, "Microwave apexcardiography," vol. 27, no. 6, pp. 618-620, 1979.

[2] J. C. J. P. o. t. I. Lin, "Noninvasive microwave measurement of respiration," vol. 63, no. 10, pp. 1530-1530, 1975.

[3] A. Høst-Madsen, N. Petrochilos, O. Boric-Lubecke, V. M. Lubecke, B.-K. Park, and Q. Zhou, "Signal processing methods for Doppler radar heart rate monitoring," in *Signal Processing Techniques for Knowledge Extraction and Information Fusion*: Springer, 2008, pp. 121-140.

[4] W. Ser, J. Yu, X. Guo, J. Zhang, and M. E. H. Ong, "Noncontact heart rate measurement using a 24 GHz Doppler radar," in *Microwave Workshop Series on RF and Wireless Technologies for Biomedical and Healthcare Applications (IMWS-BIO), 2013 IEEE MTT-S International*, 2013, pp. 1-3: IEEE.

[5] O. Boric-Lubecke, V. M. Lubecke, A. Host-Madsen, D. Samardzija, and K. Cheung, "Doppler radar sensing of multiple subjects in single and multiple antenna systems," in *Telecommunications in Modern Satellite, Cable and Broadcasting Services, 2005. 7th International Conference on*, 2005, vol. 1, pp. 7-11: IEEE.

[6] L. Anitori, A. de Jong, and F. Nennie, "FMCW radar for life-sign detection," in *IEEE radar conference*, 2009, pp. 1-6.

[7] P. E. McSharry, G. D. Clifford, L. Tarassenko, and L. A. J. I. t. o. b. e. Smith, "A dynamical model for generating synthetic electrocardiogram signals," vol. 50, no. 3, pp. 289-294, 2003.

[8] D. S. ADF, "Direct Modulation/Fast Waveform Generating, 13 GHz, Fractional-N Frequency Synthesizer."

[9] D. S. ADF, "24 GHz VCO and PGA with 2-Channel PA Output."

[10] D. S. ADF, "4-Channel, 16-Bit, Continuous Time Data Acquisition ADC."

[11] D. S. ADF, "4-Channel, 24 GHz, Receiver Downconverter."

[12] D. S. ADF, "Blackfin+ Core Embedded Processor."

[13] G.Images, "https://www.google.ie/search?q=inras+demorad&rlz=1C1CHBF_enIE755IE755&source=lnms&tbm=isch&sa=X&ved=0ahUKEwit8Zn6hZDeAhVkJMAKHSxVBJQQ_AUIDigB&biw=1920&bih=1089#imgrc=AwOthdQyjYj2PM:," 10/09/2018.

# An Application of 24GHz RADAR for Contactless Heart Rate Monitoring


E&CE | Department of Electronic and Computer Engineering

Oisin Watkins

LM118 Electronic & Computer Engineering – General Stream


## Introduction

This project was specified in consultation with and partly funded by the Automated Transport & Safety department of Analog Devices, Inc. Limerick, with a large portion of the testing being conducted on University of Limerick campus. The goal is:

- Use the ADI Demorad 24GHz RADAR platform to measure cardiopulmonary movement at a distance of 1m from the subject with an accuracy of 90% or greater

This system would find its place in intelligent vehicles, more specifically the automated emergency services contact system of intelligent vehicles.

## Aim

Write a prototype MATLAB script to control the Demorad and read Heart Rate data from it. When the prototype has been confirmed to be functional, test it in as noiseless an environment as can be found.

Using the an appropriate error metric, test whether the prototype passes the accuracy requirement outlined in the project definition.

Ideally at the end of this project one or more designs will exist which meet this accuracy requirement. These designs will go on to form the basis for the further research of this project

## Method

This section will detail both the design method and the testing method employed during the course of this project.

### Design Method:

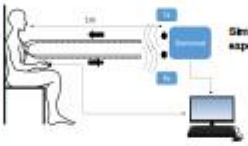
The resulting prototype from this design process will measure the distance from itself to the patient's chest wall over time and use this to discern the patient's heart rate. This design must therefore account for every factor contributing to the movement of the chest wall.

Main while loop of code, where data capture and signal processing in performed. Omitting code that establishes connection to board

## Method

### Testing Method:

In the interest of characterisation, this prototype was tested in the Anechoic chamber found in Main building, C-block, floor 0. Here a version of a Method Comparison study was performed (however the typical analysis performed on a Method Comparison Study was not appropriate for this experiment, as discussed in the results section), where the measurement taken from this prototype was compared against the "Gold Standard" SpO2 fingertip optical sensor.



Simplified diagram of experiment setup

Dr. Bob Strunz in the Anechoic chamber during testing

Contec Pulse Oximeter; Sp02 measurement device. Used as a reference for this experiment.

Table 4: 4-Tap Moving Average Filter – Resolution 0.1Hz

| Maximum Error % | Average Error % | Standard Deviation % |
|---|---|---|
| 26.3827 | 6.0316 | 10.2565 |

Table 5: 4-Tap Moving Average Filter – Resolution 0.05Hz

| Maximum Error % | Average Error % | Standard Deviation % |
|---|---|---|
| 18.5413 | 3.2969 | 11.0123 |

Table 6: 1-Tap Sliding Window Filter – Resolution 0.1Hz

| Maximum Error % | Average Error % | Standard Deviation % |
|---|---|---|
| 11.3625 | 5.8124 | 8.3567 |

Table 7: 1-Tap Sliding Window Filter – Resolution 0.05Hz

| Maximum Error % | Average Error % | Standard Deviation % |
|---|---|---|
| 11.0571 | 5.151 | 5.334 |

Next we define Mean Absolute Percentage Error as:

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{Exp_i - Ref_i}{Ref_i}\right|$$

And chart the MAPE of each design, ordered from worst to best:

Table 8: Rank – Ordering Filter Designs

| Filter Design | Mean Absolute Percentage Error (%) |
|---|---|
| 8-Tap Moving Average (0.1Hz) | 30.249 |
| Gaussian Filter (0.1Hz) | 17.1244 |
| Gaussian Filter (0.05Hz) | 17.7485 |
| 4-Tap Moving Average (0.05Hz) | 12.0955 |
| 4-Tap Moving Average (0.1Hz) | 9.247 |
| 1-Tap Sliding Window (0.1Hz) | 8.8673 |
| 1-Tap Sliding Window (0.05Hz) | 5.1508 |

The 1-Tap Sliding Window design, with frequency resolution set to 0.05Hz performed the best, achieving error of 5.15% (accuracy = 94.85%). Two other designs also passed the test criteria, though with marginally worse performance.

## Results

As mentioned previously, Method Comparison Study analysis was not appropriate for this project. This is because method comparisons need data which has a wide input value range. Here, the heart rate of the subject remained in the 70-90 bpm range, meaning that a scatter-plot drawn from this data would be clustered around one set of values. Performing time-series analysis proved far more useful. From the data gathered in the experiment the maximum error, average error and standard deviation of the error were tabulated for each design. Later, each design was compared based on their Mean Absolute Percentage Error (MAPE).

These are the output plots from the two best designs (1-Tap Sliding Average at 0.1Hz and 0.05Hz Resolution)

Table 1: 8-Tap Moving Average Filter – Resolution 0.1Hz

| Maximum Error % | Average Error % | Standard Deviation % |
|---|---|---|
| 58.4161 | 30.249 | 11.2456 |

Table 2: Gaussian Filter – Resolution 0.1Hz

| Maximum Error % | Average Error % | Standard Deviation % |
|---|---|---|
| 40.7501 | 6.2004 | 10.6446 |

Table 3: Gaussian Filter – Resolution 0.05Hz

| Maximum Error % | Average Error % | Standard Deviation % |
|---|---|---|
| 44.3769 | 5.4705 | 10.3484 |

## Conclusion and personal reflection

As seen above, this prototype design more than achieved the accuracy requirement outlined at the beginning of the project. If this project were to be done again greater variations on test conditions would be attempted: more test subjects under different conditions (some who are deliberately slowing heart rate, others who are speeding it up, etc).

There is also a lot to be said for defining the type of experiment being carried out early on, here the data is gathered and analysed later. A more efficient approach would have been to figure out how to analyse the data immediately, then have that pre-programmed into the script. This would doubtless have saved time, but the end result would have been the same.


## Acknowledgements

My friends and family for proof reading my report and for listening to me practicing my presentation. My supervisor Colin Fitzpatrick for providing report writing advice, my ADI co-Ordinator Padraic O'Reilly for keeping the project on track, Dr. Bob Strunz for providing the Anechoic chamber and volunteering to be the test subject and the ADI Automated Transport and Safety Design Evaluation lab for providing their support.


University of Limerick – Ollscoil Luimnigh

Analog Devices – Analog of what's possible