Software Design

2/22/2018

Dr. Fatemah Afghah

Version 2.0

Nathan Payton-McCauslin, Alexander Grzesiak,

James Todd

Data Menders

Table of Contents

1.0 Introduction

This document serves to outline the overall architectural design as well as the design of each major module in our project in order to deliver a complete package and introduce the reader to the necessity and importance of the project. This document is essentially a blueprint for the final product.

Data Menders is a capstone team aimed to help facilitate the reduction of false alarms in intensive care units (ICUs) through the analysis and interpretation of signals to determine the true nature of an alarm. In a typical ICU, nurses are constantly scrambling to tend to fires that may not exist. Many of these non existent fires are false or meaningless alarms. A false alarm is an alarm that is triggered to medical staff about a patient that is interpreted as a danger when it is really benign. These false alarms can create "alarm fatigue" in medical staff since they are forced to respond to numerous alarms throughout the day. In other words, false alarms create a "cry wolf effect" with medical staff, this is to say that nurses become desensitized to alarms after responding to numerous alarms and thus they stop responding in the correct way. False alarms and the resulting alarm fatigue can result in interruption of patient care, depressed immune systems through sleep deprivation and even death. In fact, the Emergency Care Research Institute placed false alarms at number one on their list of the Top 10 Health Technology Hazards for the years 2012, 2013, and 2015. To put this quantitatively, from 2005 to 2008, the FDA database received 566 reports of patient deaths related to alarms of monitoring devices. The idea for the project comes through real life experiences pertaining to our client and sponsor and an obviously glaring weakness in the healthcare field.

Our client and sponsor, Dr. Fatemeh Afghah, is an assistant professor in the School of Informatics, Computing, and Cyber Systems at Northern Arizona University. In addition to teaching, her research areas include: wireless communications, game theoretical optimization and biomedical signal processing. Her current research focuses on developing predictive modeling techniques using game theory and graph theory to optimize the performance of current medical diagnosis methods. Her expertise with biomedical signal processing and medical diagnosis methods are invaluable to our project's success.

Automated monitoring has revolutionized care in modern ICU units around the world because it allows continuous monitoring of patient vital signs such as: blood pressure, pulse and ECG signals. These automated monitoring devices watch these vital signs and alert medical staff if there is an anomaly by way of a medical alarm. This is great news for medical staff as they can become more productive by being able to work on other tasks, however, for all the good automated monitoring does for ICU units, it's also prone to false alarms. False alarms here occur from any number of things: sensors become loose, muscles have spasms, monitors are faulty, incorrect placement of sensors by staff, etc.

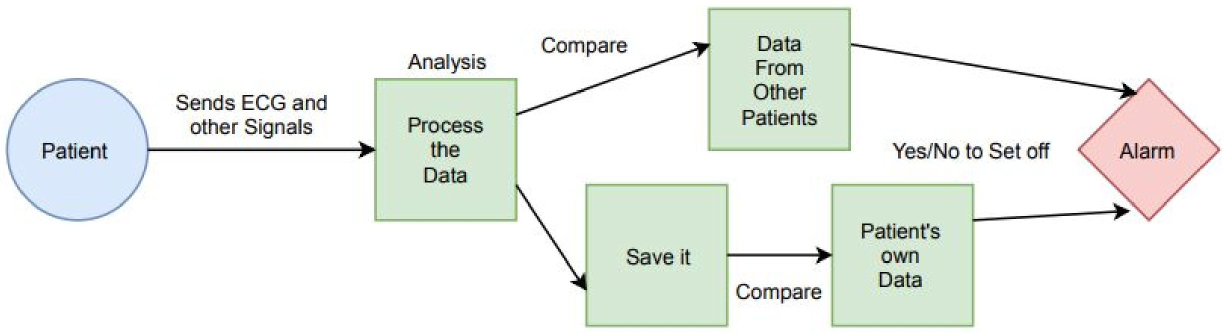
2.0 Implementation Overview

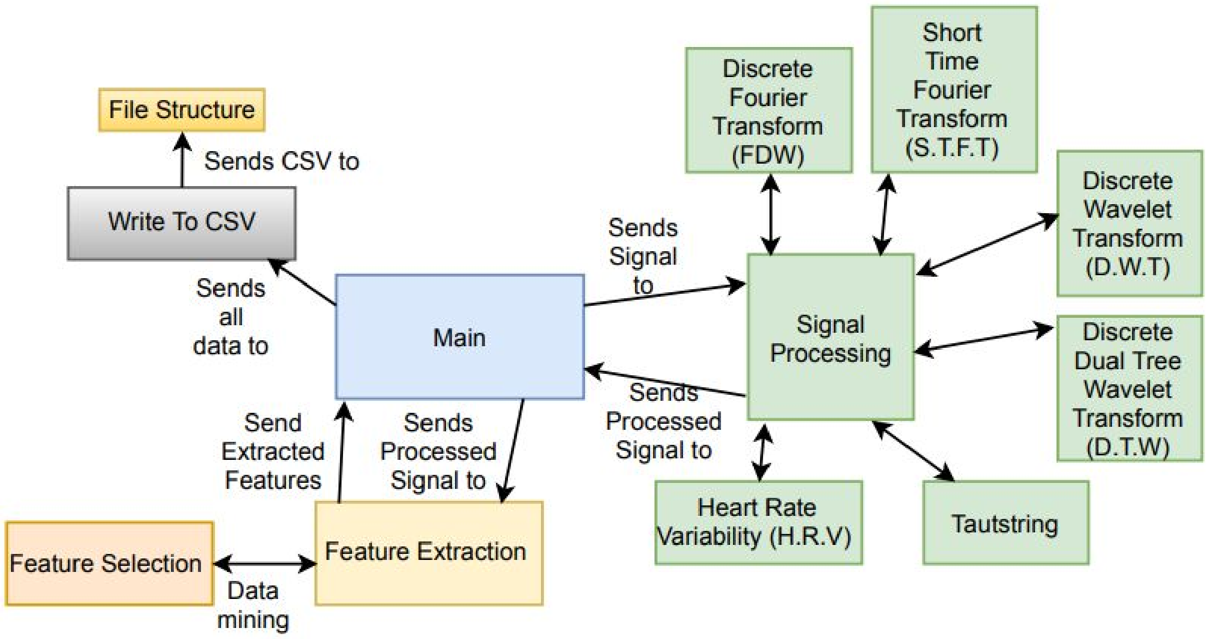
As you can imagine, false alarms have been addressed by medical professionals and engineers, and attempts have been made to reduce them. However, the majority of attempts so far have been focused on improving hardware such as improving monitors or increasing the accuracy of sensors. The problem with this approach is that, as mentioned earlier, people are imperfect. Our project focuses on the much more realistic idea that regardless of the sophistication of the hardware, patients will still move around and staff won't always hook everything up correctly and thus false alarms will be triggered. This is where a software package would come in handy to analyze multiple signals from a patient and determine what kind of care is called for.

Our software packages will alleviate the stress put on ICU medical staff and patients and as a result may aid in saving lives In order to accomplish these goals there are several things that need to be done, we need to: record signal inputs, put the signals through pre-processing, extract features from the pre-processed signals, select the features that are more useful in a given situation, run those select features through machine learning algorithms, create a user interface front end and create a user manual for easy operation of the software (an in depth explanation of each of these are given in section 4 of this document). In order to achieve this, we are using the MATLAB programming language with and without the library WFDB for pre-processing and WEKA for machine learning.

The WFDB MATLAB library gives us a greater ability to manipulate patient signals and extract important features. It also gives us a code framework for which to build our programs onto without having to recreate the basic tools. WEKA is a machine learning framework that analyzes patterns in sets of data to predict solutions to new sets of data.

3.0 Architectural Overview




File Structure
Sends CSV to
Write To CSV
Sends all data to
Main
Sends Signal to
Sends Processed Signal to
Send Extracted Features
Sends Processed Signal to
Feature Selection
Data mining
Feature Extraction
Signal Processing
Discrete Fourier Transform (FDW)
Short Time Fourier Transform (S.T.F.T)
Discrete Wavelet Transform (D.W.T)
Discrete Dual Tree Wavelet Transform (D.T.W)
Heart Rate Variability (H.R.V)
Tautstring

Our project will consist of four critical parts. Main, this will drive the program. Signal Processing, this will take in signal data and process it using one or multiple techniques. Feature Extraction, this will obtain the important points of data from the processed signal. Feature Selection is the ability to learn what Features are more useful. Finally, Write to CSV, this will take data from signal processing and feature extraction and write it to CSV files.
Main is the driver of the whole program. Main does interact with the user to see what they want, and will also need an input file passed into it. This input file will be called config.txt, the contents of this file will have names of other files that are CSVs. Within these files will be signal data. This is so that Main can do batch file processing.

The Signal Processing part has multiple parts connected to it. This is because some signal processing algorithms are more complex than others, thus needing their own files to make the code cleaner. Once the passed in signal has been processed Signal Processing will send it back to Main. The algorithms used in Signal Processing are as follows: Discrete Fourier Transform (DFT), Short Time Fourier Transform (STFT), Discrete Wavelet Transform (DWT), Discrete Dual Tree Wavelet Transform (DTW), Taut String, and Heart Rate Variability (HRV). These algorithms will be discussed in more depth in other sections of this paper.

Feature Extraction can still grow a little. With this part of the project, Feature Extraction will analyze the code that Main got from Signal Processing. Then once finished, this will be sent back in to Main. The way that Feature Extraction can grow is by more complex methods of feature extraction, much like Signal Processing.

Another important part to this project, but only talked about briefly, is Feature Selection. This is when the data from Feature Extraction is analyzed and is able to tell what features from a signal are more important for false alarm detection. By doing this, the features that are more important will be extracted and the less important ones will not be. This will be an integrated process, meaning it will be done within the overall program. This will be done by using a tool called Weka. Weka uses data mining techniques to find the important feature that will be used to determine if an alarm is false or not.

Write to CSV will be responsible for converting the given data into a CSVs file type. Then given what kind of data it is, the CSV will be placed in a folder. The folder will be named after the kind of data it is. Because there is Signal Process data, and there is Feature Extraction data, there will be a folder Signal Process and a Feature Extraction.

The File Structure will hold data that Main will process, meaning Main will make CSVs for each kind of data that is made throughout the program. The File structure will organize the data, so when accessing, it will be nice and easy to do so. The data saved within the File Structure will be used to help predict false or true alarms with a given patient

With the final product, there will also be a user interface interacting with Main itself. The goal for this interface is to allow the use to have a nice time interacting with the program.

The expected flow of this will be to pass in a config.txt file in to Main. Within this config.txt file, there will be a list of CSV file names where each file contains signal data. Once Main knows all of the files that need to be processed, Main then asks the User what they want done in signal processing. Once the User has answered the questions, Main will then call the respective functions in Signal Processing. Signal Processing will then send back the results it got from processing the given signal, and in Main. Main will make a CSV for each kind of signal processed in this way for each signal. Once all of the Signal Processing is done, Main will then call Feature Extraction for all of the processed signals. At that point, once Feature Extraction is finished analyzing the signals and has send it back to Main, Main will write that data into CSVs like it does for Signal Processing. Once all of the names within the config.txt file have been processed, then the program has finished running. This is the normal flow of the program.

4.0 Module and Interface Description

4.1 Pre Processing

The first component a patient's data will encounter is the pre-processing module of the program. Here transformations will be applied and a patient's ECG signal will be broken down into points of interest such as the R peaks of the ECG. These pre-processing techniques only serve to break the signal into as many subdomains as possible. This is because one processing technique may be able to offer more insight into the signatures of a certain heart condition than others.

4.1.1 Removing Artifacts

In an ICU environment there are many sources of noise that make their way into an ECG signal. The problem with removing some of these sources of noise is it's hard to remove only the noise. Some sources of noise have very definitive frequencies and could be easily removed. The problem is however, there may be useful signal data that shares the same frequency as the noise so removing the noise also causes a loss in data. For this reason very conservative noise reduction if any is used. This being said, there is one source of noise that is easily removed. Baseline wander is the way an ECG signal may fluctuate throughout time causing it to take on a wavy structure. Since the baseline wander may only contribute 1 or 2 full waves throughout the course of a 30 minute ECG signal, we can remove this with a great deal of confidence that we won't remove useful data.

The function 'remove_BW' is a function that removes this baseline wander so later transformation and feature extraction is working with more accurate data. The function takes in an ECG signal and the sampling frequency of the signal. The signal is run through the discrete fourier transform to get all the frequencies that make up the ECG signal. Now, utilizing the fact that baseline wander contributes such a low frequency to the signal, we can safely remove all frequencies under .5 Hz and in some cases 1 Hz. After this is done, the inverse fourier transform is used to get the ECG signal back and the new signal without the baseline wander is returned.

4.1.2 SignalTransformations

One of the first methods that will get called is the one which takes an ECG signal and performs several signal transformations to obtain different ways of representing the same data. The signal transformations include Discrete Fourier Transform (DFT), Discrete Wavelet Transform (DWT), Discrete Dual Tree Wavelet Transform (DTW), Taut String, and Short Time Fourier Transform (STFT). Each of these transforms offers something different that may prove valuable to future pieces of the program. For example, DFT is great at showing what frequencies are present in a a signal without regard to where they occured. DWT builds on this and offers the same insight into the frequencies present in a signal but also gives the time at which the signals occurred. DTW is very similar to DWT except it effectively has two times the information as DWT as alluded to in the name of the transformations. Taut String is a smoothing function which, depending on an epsilon value, will take a signal and remove variability in the signal eventually leaving only the most radical changes. STFT uses the same underlying frequency analysis as DFT but instead of applying it to the entire signal, STFT breaks the signal into windows and performs a fourier transform on the windows. This gives greater detail into when the frequency occurred but only within the limits of the window.

This function is called with a patient's ECG signal as the only input and after all the signal transformations have been applied a variable length output array is returned. Each transformation returns its own structure of data and to be sure the calling function has every bit of the data, the return values are vastly different in the information they carry. For example, DFT only returns a 1D matrix corresponding to the amplitude of some frequency. This differs from the output of DTW which is a struct containing the levels of the wavelet transformation that were used as well the actual data from these levels. It is up to the calling function to decide which information it deems relevant from the returned array.

4.1.3 Heart Rate Variability (HRV)

Heart Rate Variability (HRV) is useful to us for analyzing a patient’s heart rate over a period of time for anomalies. HRV is a signal that is generated from an ECG signal and provides many more insights into the patient's heart than a pure ECG does. For example, using an HRV signal you are able to calculate a patient's heart rate as well as the patients heart rate standard deviation, average, domain, and many more.

Extracting an HRV signal from a regular ECG signal is done by obtaining QRS peaks from the ECG signal and taking the differences between them to obtain an interval.The inverse of these obtained intervals make up the new HRV signal.

4.2 Feature Extraction

After the signal has been broken down into different signals using data transformations and HRV has identified the attributes of all the heart rhythms, features about the new data have to be identified in order to attempt to find trends among both healthy patients and unhealthy ones. If trends are only identified from one of these groups, it will not be possible to know which attributes of an unhealthy persons ECG signal caused it to be classified as unhealthy.

There are different kinds of features that can be taken from a set of transformations. Statistical features such as mean, median, standard deviation, and harmonic mean are some that can be extracted from just about any signal. The extractFeatures function takes in a variable number of inputs depending on the use. Of course it needs an input signal but it can also take in information to save the features in a file such as a flag to say if output to file is desired, a file name, and the patient name since there is usually multiple transformations for any given patient and their condition. Whether or not the information gets saved to file, it will always get returned to the calling function. The order of the features can be found in a cell array that is also returned.

4.2.1 Extracting HRV Features

The HRV signals our package generates requires a different set of features be extracted. In addition to the statistical features mentioned above, we want to get distances between each peak rather it be R1-R2, P1-P2, Q1-Q2, etc. We also want to get other simple arithmetic features of these peaks such as R1/R2, P1*P2, etc. and more advanced features like (P1-P2) / (Q1-Q2) and power functions. This information allows us to analyze the patterns in signals of healthy and unhealthy patients to determine what specific point in a given feature causes problems. Overall, we extract 700 features from the HRV signals, not all are significant. Each patient's HRV features are copied as its own row in a .csv file where they can be easily compared to each other and analyzed.

4.3 Feature Selection

Once a signal has gone through feature extraction there are hundreds of features from statistical features to unique HRV features. These features are useful information but may not all be needed when trying to find the set of features that gives the highest level of accuracy for a given condition. To do this, the features that come out of statistical analysis from signal transformations as well as the statistical and peak feature from the HRV are analyzed to determine which features are best used at detecting certain heart conditions.

This can be done with the Weka API. Since the features that any given heart condition are used for multitudes of of patients and rarely change, features can be coded into the alarm classification. To make sure the set of features are as accurate as possible, a training set is used consisting of healthy patients to get a baseline of what features expect to look like as well as unhealthy patients to give data on the specific features where abnormalities are seen.

4.4 Alarm Classification

The features that are taken from a patient's ECG signal are originally compared to healthy patients data to look for abnormalities meaning a heart condition. The features that are used to look for any number of conditions are continuously monitored as new ECG data comes in. A function called detectConditions will use selected features from previous steps to compare to an aggregate of known features that signify a heart condition. If the program determines that the patient's ECG features are outside a certain threshold, it determines that an alarm should sound.

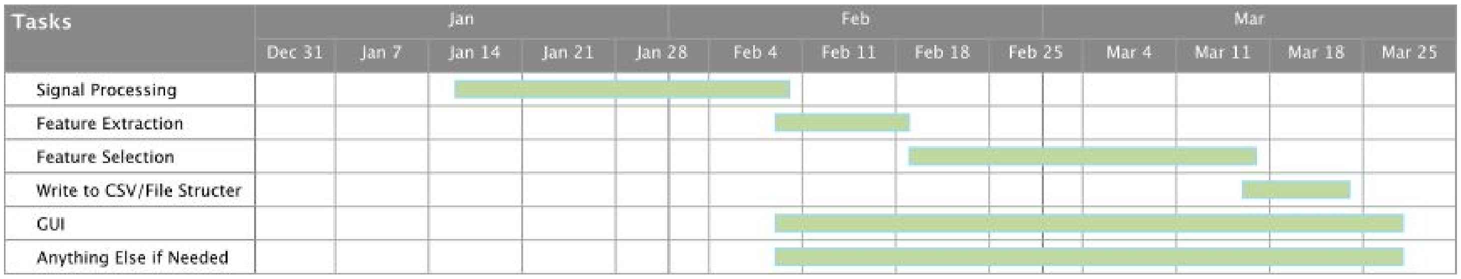
However, as a patient spends time in an ICU, features that once were determined to cause a false alarm can be saved so the next time the once abnormal feature values are seen, an alarm may not need to be triggered. This is aided through two functions. The first is called isAlarmForPatient and deals with determining whether or not a set of features should sound an alarm for a patient. The function first references a database of features and known conditions that those features correspond to. If an alarm condition is determined and there is no previous patient data, an alarm will be triggered. But, if there is patient data it will then be referenced to check if the set of features have been previously marked as normal for the patient. To do this the function takes in an array of features that are being examined, a reference to the generic patient database, and a possible reference to the patient's specific database. The function will do its evaluations of the the patient databases and its output will be an alarm result.

If no alarm has been determined, the next ECG rhythm will be analyzed, but if an alarm goes off there is the option for it to be classified as a false alarm. In this case, input from a nurse will cause a call to the second main function savePatientData. As input, this function takes in the features that caused an alarm to be triggered, the alarm that was triggered, and a reference to the patient's existing specific database. The call to this function adds or updates the database of the patients data which will serve as “normal” for a patient. To ensure there is record of the patient's unique “normal” feature values, each call to this function will update a file that will maintain a storage copy. To mitigate the risk of a few outlying features that may be present for a single alarm but not truly indicative of the false alarm, the saved data about one specific alarm will build up to the point where only features that can show a strong connection to a false alarm are given prominence.

4.5 Utility Functions

There are numerous functions that don't fall under any particular category due to their use throughout the program.

5.0 Implementation Plan



0This Gantt chart shows the order on how parts of this project will be done.   Something to keep in mind is that each part has a substantial amount of work, thus this is a high level overview of how things will be completed in time.

The first part will be Signal Processing. All of the logic to process signals and to send their data back to who call them will be implemented here. There are six kinds of signal processing techniques that will be used, which are: Discrete Fourier Transform (DFT), Short Time Fourier Transform (STFT), Discrete Wavelet Transform (DWT), Discrete Dual Tree Wavelet Transform (DTW), Taut String, and Heart Rate Variability (HRV). Throughout obtaining the different signals for Signal Processing, they will be tested one by one to insure data integrity. Once that is finished, the other parts of the project can start.

The next section that can start, after Signal processing is finished, is Feature Extraction. The Feature Extraction is needed to be finished for the Feature Selection part of this project, because Feature Selection cannot be started without it. The Feature Extraction will also test that the signal processing part was done right because the features extracted from the processed signal will either be good or bad. This is tested by our mentor.

With the GUI part of this project it can start once Signal processing is finished. This is because the GUI can call the different algorithms for Signal Processing, and save that data in the same directory of the program for now. As other parts are finished, more things will be added to the GUI.
The Feature Selection will be able to start after the Feature Extraction ends. With Feature Extraction's data, it can be determined what data points are more important and needed for predicting false alarms, thus this part is important. Once this part is finished, Feature Extraction will be updated so that only the important features are obtained.

Once all of the data components are being finished, the Wire to CSV/File Stricter will begin. This is so that all of the data can be organized for use.

Also, the section Anything Else will be going through the majority of the project to add things here and there when needed.

6.0 Conclusion

To conclude this document, the false alarms that the ICU have to deal with are costing people their lives and stressing people out. The goal of this project is to help reduce that. Since false alarms can cause nurses to get a "cry wolf effect", the plan is to reduce that by analyzing the inputs of the devices that the patients are connected to. By having a device look at all of the inputs, a more accurate alarm can be set. This document talked about how the software package will come together to accomplish this goal. The parts of this package seem simple to complete, but due to the nature of this project, each part will be looked into appropriately for correctness and neatness of the code. This ensures correct outcomes of false alarms. The organization of this project will help further research for others in similar areas because of the GUI. In the end, the goal of this project is to help reduce false alarms in the ICU. By doing so, we can reduce the stress of the ICU staff and help save lives.