**Partial Discharge Classification using AI Techniques**

Final Report for CS39440 Major Project

*Author*: James Slater ([jas38@aber.ac.uk](mailto:jas38@aber.ac.uk))

*Supervisor*: Dr Neil MacParthalain ([ncm@aber.ac.uk](mailto:ncm@aber.ac.uk))

6th March 2012

Version 1.0

This report is submitted as partial fulfilment of a BSc degree in  
Artificial Intelligence and Robotics (GH7P)

Department of Computer Science

Aberystwyth University

Aberystwyth

Ceredigion

SY23 3DB

Wales, UK

**Declaration of originality**

In signing below, I confirm that:

* This submission is my own work, except where clearly indicated.
* I understand that there are severe penalties for plagiarism and other unfair practice, which can lead to loss of marks or even the withholding of a degree.
* I have read the sections on unfair practice in the Students’ Examinations Handbook and the relevant sections of the current Student Handbook of the Department of Computer Science.
* I understand and agree to abide by the University’s regulations governing these issues.

Signature ……………………………………………. (James Slater)

Date …………………………………………………

**Consent to share this work**

In signing below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Signature ……………………………………………. (James Slater)

Date …………………………………………………

**Acknowledgements**

I am grateful to…

I’d like to thank…

**Abstract**

The aim for this project is to create an Android Application to classify data from a test instrument, which detects Partial discharge in power distribution networks. The detection instrument will be provided by EA Technology (the industry partner for this project).  Partial discharge signatures are used in order to detect the deterioration of the insulation properties relating to cabling, switching, transformers or other components of an electrical network. Partial discharge has the effect of causing a progressive deterioration of insulating materials, which will eventually lead to complete electrical breakdown. The effects of partial discharge within high voltage cables and equipment can be very serious, ultimately leading to a very high cost of replacement. The early detection of insulation breakdown in networks therefore offers a great number of advantages.

The App proposed in this project is designed for engineers to use for high voltage asset (cabling and infrastructure network) management; the instrument will provide data to the App in order to determine if an asset (cable, transformer, switching network, etc.) is deteriorating by examining the noise signature. If time permits, additional functionality will be added such that raw audio signal can be sent to the Android device, and processed in real-time, otherwise it will use saved data. Artificial Intelligence techniques will be used to develop a means of processing and classifying the demodulated ultrasonic (audio) waveforms.

The conclusion is that the App will classify different noise signatures and be able to discriminate between Partial Discharge, background noise or other interference (i.e. transformer noise).

**Contents**

1. Background, Analysis & Process 7

1.1. The problem 7

1.1.1. Partial Discharge Basics 7

1.1.2. Ways of detecting Partial discharge 8

1.1.3. Android Device capabilities and limitations 8

1.1.4. Features extraction 9

1.1.5. Artificial Intelligence Classifiers 10

1.2. Analysis 10

1.3. Process 12

2. Design 13

2.1. Overall Architecture 13

2.2. Detailed Design 13

2.2.1. Structural View 15

2.2.2. Implementation View 16

2.2.3. Behavioural View 16

2.3. User Interface Design 17

2.4. Other Relevant Sections 19

3. Implementation 21

3.1. Final Application 21

3.1.1. Recording 21

3.1.2. Classification 21

3.2. Spike work 24

3.2.1. Understanding Android 4.4+ 24

3.2.2. Loading WAV Audio files in Android 25

3.2.3. Feature extraction 25

3.3. Issues that arose 26

4. Testing 28

4.1. Overall Approach to Testing 28

4.2. Automated Testing 28

4.2.1. Unit Tests 28

4.2.2. User Interface Testing 28

4.2.3. Stress Testing 29

4.3. Integration Testing 30

4.4. User Testing 30

5. Critical Evaluation 32

6. Appendices 34

A. Third-Party Code and Libraries 34

6.1. Project Specification 34

6.1.1. Android App 34

6.2. Gantt Chart 34

6.3. UML Diagrams 34

6.3.1. Class Diagrams 34

6.4. UML Creator Visual-Paradigm 37

http://www.visual-paradigm.com/ 37

6.5. Feature Spike work 37

6.5.1. NON Partial Discharge 37

6.5.2. Partial Discharge 38

6.6. jAudio Features Analysis 39

6.7. Testing Tables 39

6.7.1. Load the application. 39

6.7.2. Main Menu Screen 39

6.7.3. Record screen 40

6.7.4. Save As Screens 41

6.7.5. File Chooser Screen 41

6.7.6. View Data screen 42

6.8. Final Screen Shots 43

B. Code Samples 43

6.9. Graph View 43

6.10. jAudio 2.0 43

6.11. Libsndfile 43

6.12. Recording WAV format audio in Java 43

6.13. YAAFE 44

6.14. Stack Overflow 44

7. Annotated Bibliography 46

# Background, Analysis & Process

This project will be undertaken in conjunction with a company called EA Technology, who work in the electrical industry offering services and instruments to help manage the state of the High/Medium voltage assets. Their aim is to enable customers to manage their assets more efficiently, with fewer failures and outages. The client base of EA Technology are involved in electricity transmission, distribution and generation, together with operators of electrical assets in petrochemicals, mining, process engineering, manufacturing, transport infrastructure, health and defence. They also work closely with government and commercial organisations to enable the implementation of smart grids and new low carbon technologies.

I have a Higher National Diploma in Electronic and Electrical engineering and have always been interesting in this area. Working at EA Technology over my year in industry provided me with the motivation and background knowledge for this project.

## The problem

Partial discharge is one of the leading causes of failures of electrical assets in a power system. Examples of an electrical asset are cables, transformers or switching networks. As an example, a transformer converts between different voltage levels and is one of the most critical aspects of the system. Many transformers today are in service past their designed life cycle, and keeping them in service makes economical sense considering the cost of repairs and replacements. A transformer failure can result in a small explosion damaging surrounding transformers, a loss of power to a portion of the population and ensue a fine to the electrical provider. Consequently it is essential to be able to detect the condition of an asset and to plan it’s service life.

Several methods can be used for monitoring changes in the electrical asset; an extensive list of available methods is available in [12]. I will be focusing on using the ultrasonic sound released by partial discharge as my data and find a way of classifying the state of an asset.

### Partial Discharge Basics

The Partial Discharge User Group defined partial discharge as follows, “Partial discharge is an electrical discharge or spark that bridges a small portion of the insulation between two conducting electrodes. Partial discharge activity can occur at any point in the insulation system, where the electric field strength exceeds the breakdown strength of that portion of the insulating material”[1]

Partial discharge has a deteriorating effect and over time it reduces the lifetime of an insulation system. Partial discharge, either on the surface or inside an electrical insulation, incurs high-energy electrons or ions to deteriorate the insulation material. This may result in chemical decomposition of the insulation material. [18]

Partial discharge is a sign that the electrical asset has begun to wear and may fail, hence it is very important to be able to detect and monitor the condition of electrical assets to better understand the wear and assess if an asset will need replacing.

A paper published by Cigre interestingly looked at the many forms of partial discharge via an oscilloscope [14].  I will keep this research in mind as I work through my own work, however as I am using ultrasonic data an oscilloscope will not be of use to me. Having said this, the research provides a better understanding of how each sub group of partial discharge happens and their effects on electrical assets.

### Ways of detecting Partial discharge

**Electrical Detection**

Two types of sensors are commonly used for partial discharge measurement in transformers: capacitive and inductive coupling sensors. Capacitive sensors work by measuring changes in an electrical property, called capacitance. Capacitance describes how two conductive objects with a space between them respond to a voltage difference applied to them. Inductive sensors measure inductance of an electromagnetic field in the close surroundings of the sensing surface of the electrical asset. Inductive sensors are common measuring devices for rapidly varying currents for example in transformers.[16]

**Acoustic detection**

The acoustic wave, audible or not, is due to the expansion of gases near the partial discharge, which propagates as a pressure wave. The main frequency used for acoustic detection is between 10 kHz to 1000 kHz and usually ultrasonic partial discharge detectors are tuned at 40 kHz. Acoustic detection has the advantage of being able to localise the partial discharge source. It calculates the time difference for ultrasonics to reach multiple successive sensors within the transformer tank, thus pinpointing the location.

**Chemical detection**

Detection of chemical by-products produced by partial discharge activity is one of the simplest methods for partial discharge detection. When the electrical charge exceeds the field strength of air a spark is created that arcs across a void in the insulating material, which is an example of partial discharge. This breaks down the surrounding insulator into it’s chemical components, which can then be detected chemically. The two primary chemical tests employed by power companies today are dissolved gas analysis (DGA) and high performance liquid chromatography (HPLC).[17]

This type of detection is very well established, immune against noise and relatively easy to measure. It cannot say, however, much about the type of defect, location and intensity of partial discharge, and it also involved taking the asset offline.

### Android Device capabilities and limitations

Android is the chosen operating system of the customer EA Technology. Android is available worldwide on a wide range of devices, hence accessible to all of EA Technology’s customers.

Android was established in 2008, and the last 7 years have seen many iterations of Android versions. Each version was named after a confectionary item, for example the current version 5.0 has the name Lollipop.

It is possible to make an App backwards compatible, and after some conversation with EA Technology it was decided to making all devices running 4.4+ onwards compatible with the app.

The minimum specifications for running Android 4.4 is 512 RAM, see [22] for the full list of hardware requirements. I will be designing and testing the app on a Samsung S4, which has a specification as follows [20]: -

* Screen - 5.0 inches
* Processor - Quad-core 1.6 GHz Cortex-A15 & quad-core 1.2 GHz Cortex-A7
* Sensors - Accelerometer, gyro, proximity, compass, barometer, temperature, humidity, gesture
* Memory 2 GB RAM, 8GB Internal storage
* Battery - Li-Ion 2600 mAh battery

### Features extraction

All Artificial Intelligence classification techniques are dependent on the selection of features available. Partial discharge has three main data criteria:

* Phased-Resolved data
* Time-Resolved data
* Data without Phase/Time information

Phase-Resolved data is the acoustic data captured with relation to the Alternating Current (AC) test voltage. The test voltage is considered to be constant and the phase angle is split up to plot any partial discharge activity within the divided phase angle. Phase angle is used to represent an electrical impedance, in this instance the we refer to the phase angle as the phase difference between the voltage applied to the impedance and the current driven through it. Partial discharge is most likely to occur as the voltage builds, in other words when the phase is approaching a peak.

Time-Resolved data displays the true shapes of the individual partial discharge pulses. The test voltage is treated as a constant, as seen in Phase-Resolved data. This data pattern has attractive advantages, since there are some direct relationships between the physics of the defect and the shape of the partial discharge signal.

Data without Phase/Time information captures a typical partial discharge pulse where time and phase data are not recorded as a reference.

**Statistical methods for feature extraction**

Each Waveform Audio Format (WAV) audio file can be represented as an array of values between 1 and -1, therefore many statistical functions can be applied to study the audio file.

For example: Let N be the number of values in the array.

The average of all the values can show a "measure of central tendency", i.e. a statistic that represents a typical value from the data set.

The variance measures the spread of the data set. A variance of zero indicates that all the values are identical.

Skewness is a measure of the symmetry of the data, which can distinguish noise in an audio file.

These statistics are a small sample of features, which can be collated into a vector to summarise an instance of the audio file, therefore allowing the possibility of comparison of multiple instances.

**Signal processing tools**

Signal processors can be used to extract features from the audio data, such as Fourier transform, wavelet transforms and Haar transforms.

A Fourier transform breaks down a waveform into a summation of sine’s and cosines, therefore finding the frequencies that compose the original signal. A Fast Fourier transform is an algorithm to compute the Fourier transform, which can show the relationship between the test AC voltage and the partial discharge activity. Fourier transforms only consider the frequency and not time domain, which is a disadvantage because we could find that two signals have an impulse at the same frequencies even though they are different in a time domain. To overcome this, wavelet transforms can be used however at present, there is limited research into this area.

### Artificial Intelligence Classifiers

**Distance/Decision function classifiers**

Distance/Decision function classifiers use the existing information as reference for classifying new data, by considering the distances between the two data sets.

Examples of these classifiers are Minimum Distance and Nearest Neighbour. Minimum Distance classifier partitions the data so that we can quantitatively determine which class a feature belongs too. Therefore the minimum distance of the feature to the class implies maximum similarity to that class. Nearest Neighbour classifier is similar to the Minimum Distance classifier, however it allows for the possibility that the data can fall into multiple classes.

**Statistical Classifiers**

This approach uses statistical features to separate the data, for example Bayes Classifier and Support Vector Machines.

Bayes Classifier predicts the values of features for members of that class. Examples are grouped in classes because they have common values for the features.

Support Vector Machines find the optimal hyperplane for linear separable data. It can extend to patterns that are not linearly separable by transformations of original data.

**Neural Networks**

For the AI approach to classify the acoustic data, I looked into Artificial Neural Networks (ANN), as these are efficient classifiers with a very low computational overhead.

Artificial Neural networks have been very successful in pattern recognition and classification problems. The basic advantage of ANN over other classifiers is its ability to learn from examples. Knowledge in the training set is extracted and stored in the connection weights and neuron biases during the learning phase [23]. However once an ANN has been taught it cannot learn new instances and thus having to re teach a new ANN.

## Analysis

Feature extraction from the acoustic data can be highly problematic, as noise and other background sounds can interfere with the classification of the recording. With this in mind the feature extraction will be one of the main focuses of this project.

Upon extraction and analysis of the features, a classifier can be constructed using a statistical approach, which will provide a wider scope for further understanding of partial discharge. A paper on Partial Discharge pattern recognition, [19], demonstrates multiple effective approaches to understanding the waveform, which I have taken and adapted for this project. From this I hope to either construct a rule base classifier or a decision tree, depending on my findings.

Considering I only have access to limited data, I am therefore restricted to which Artificial Intelligence techniques will be appropriate to use to classify the audio files. However I intend to construct my programme with a view to potentially building upon the system in future. For example, by adding a higher level of complexity, such as a Fuzzy Logic Algorithm, I could improve the success rate of the correct classifications.

Fuzzy Logic systems are a branch of logic that uses degrees of membership in sets rather than a strict true/false membership. It is a form of many-valued logic that deals with approximate, rather than fixed and exact reasoning. Compared to traditional binary logic (where variables may take on true or false values), Fuzzy Logic variables may have a truth-value that ranges in degree between 0 and 1.

To seek classifications, as humans we often apply literary phrases to circumstances rather than opting for a numerical system, and this can be suitable to a lot of situations. For example using hot or cold for evaluation of temperature as opposed to the exact temperature in degrees. Fuzzy Logic applies this same idea, and in the case of Partial Discharge the classifications can be relatively ambiguous, however we could still achieve a clear classification.

Within the umbrella of Partial Discharge lie multiple different forms which have a slight affect on the results of ultrasonic detection. For example, Corona and Surface Discharge are two very similar types of Partial Discharge and their acoustic thumbprint differs only very slightly, which is explored in the Cigre paper [14]. The domain experts at EA Technology are currently simply listening to the recording of the audio files and detecting the difference aurally, which has obvious limitations. The difficulty with such fine differences in detection is noise, which can play a major factor in altering the classification of the captured acoustic data.

At EA Technology, they record the data with an instrument that uses a band pass filter to a 40 KHz heterodyne process to convert the raw audio to a digital signal, in order to save it as a WAV audio file. This process can be reversed to obtain the original raw data which, under analysis, may provide more information than the compressed data.

In order to create the classification algorithm, it was essential to research how to read audio files using Java. Normal playback proved to be very simple, however converting the binary data of the WAV files to a double array within the Android environment was much more challenging.

EA Technology uses a library called Libsndfile, a C/C++ library that reads audio files including WAV. As the WAV files can take many different forms, from varying sample rate to channels recorded (up to 4), using a fully tried and tested library will save time and be much more efficient as it has a community behind it for debugging and help on any problems. EA Technology also offered me the use of a server to process the features on, so that I would not be constrained by the limitations of the Android device.

Enabling native code to run within the Android environment also required research, for which I found some appropriate tutorials, which allowed me to make a small sample app. This app passed strings and integers from the Java classes to C files, providing a solid basis for me to work upon for the rest of the project. Also I began carrying out some basic statistical analysis on the audio files, which I compared against the given classification for each file. For example Skewness proved to be a useful measure for how noisy the data is. Skewness evaluates how the data lies in relation to the overall picture; if the higher values are all at the start than the end of the audio file the Skewness will show this, and vice versa. Alternatively if the audio files have noise in the background it will be present throughout and thus Skewness will provide no feedback.

Using a Neural Network is not appropriate for this project as it will prohibit the future possibility of more complex analysis inside of the current algorithm. Also given the limited information available, a Bayes Network would not be suitable as only half of the system could be implemented. I considered the possibility of designing an image recognition approach to the plot of the audio files. However the processing power needed for such an approach would be very high, thus not viable across the current Android mobile range.

## Process

EA Technology monitors the progression of their projects by using the Waterfall model, which is sequentially designed so that progress is mapped downwards (like a waterfall), from conception through to implementation. I created a Gantt chart (see section 6.2) to devise a timescale for the exchanges of data between EA Technology and myself.

For the programming aspect of the project, instead of using the Waterfall model, I intend to use a template from the Agile model and extreme programming (XP), as it offers many values needed for the project. The Agile manifesto states these core values [ref]:

**Individuals and interactions** over processes and tools

**Working software over** comprehensive documentation

**Customer collaboration** over contract negotiation

**Responding to change** over following a plan

The latter three values all apply to this project; hence Agile is an appropriate model. XP is compatible with the values and principles of Agile, through offering disciplines in order to develop a project within a set timescale. In this project only some of the XP disciplines apply, for example Small Releases and Continuous Integration. These two disciplines adhere to the ‘Working Software’ value in Agile.

Small Releases is the process of creating a release and getting feedback from the customer at the end of an iteration, which leads directly to the next value Customer Collaboration.

Continuous Integration ensures that the system is kept in a deployable state by using automated testing which is created at the same time as the class.

Responding to change will be essential in this project as I anticipate a lot of feedback from EA Technology, and therefore an ever-changing set of requirements.

In XP each requirement is transformed into a story, and from there we accumulate the stories into the objective of the application. I believe a hybrid of the two models will be successful in my project.

# Design

## Overall Architecture

The application will be built for the operating system Android, for which the main language is Java. Android is also capable of supporting native C/C++ code compiling it for the operating system and using it as an external library.

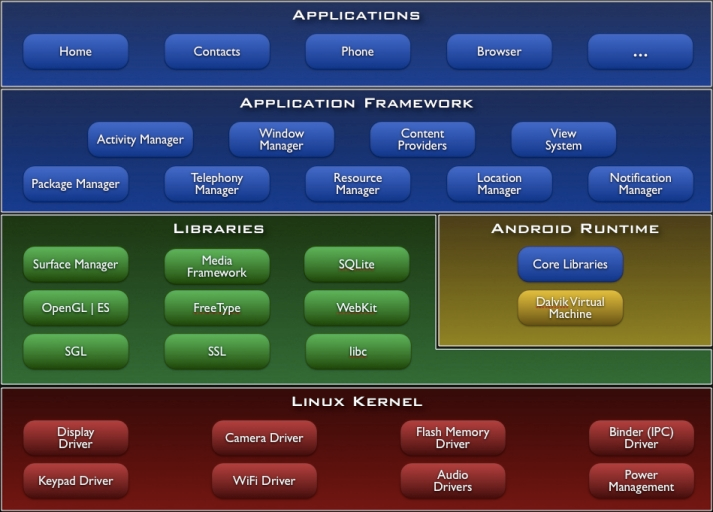


Figure -- [24]

Figure 1 demonstrates the full architecture of the Android operating system, in a hierarchical fashion, for example the modules in the Application Framework are all incorporated into all applications. For the application, we compile the C/C++ files using the Native Development Kit (NDK) to create a library, which can be loaded as part of the application from the Libraries section.

Google, the creators of Android, have designed their own Integrated Development Environment (IDE) called Android Studio, which is where I will be creating, building and debugging the app. Android Studio is not preinstalled with the native compiler, thus it will need to be downloaded and deployed within the structure of the IDE so that it will be able to compile all code simultaneously. I decided to use Android Studio over other IDEs, such as Eclipse or IntelliJ IDEA Ultimate, as I have experience in using it before thus I can quickly and efficiently create an Application. Being created by Google, Android Studio has many features that are already built into the IDE, for example the ability to create a virtual device. Other IDE’s do not have this feature and will need plugins.

## Detailed Design

Detailed design is the in’s and outs of the system using unified mark-up language (UML) to present and describe the structural, behavioural and the user interface designs.

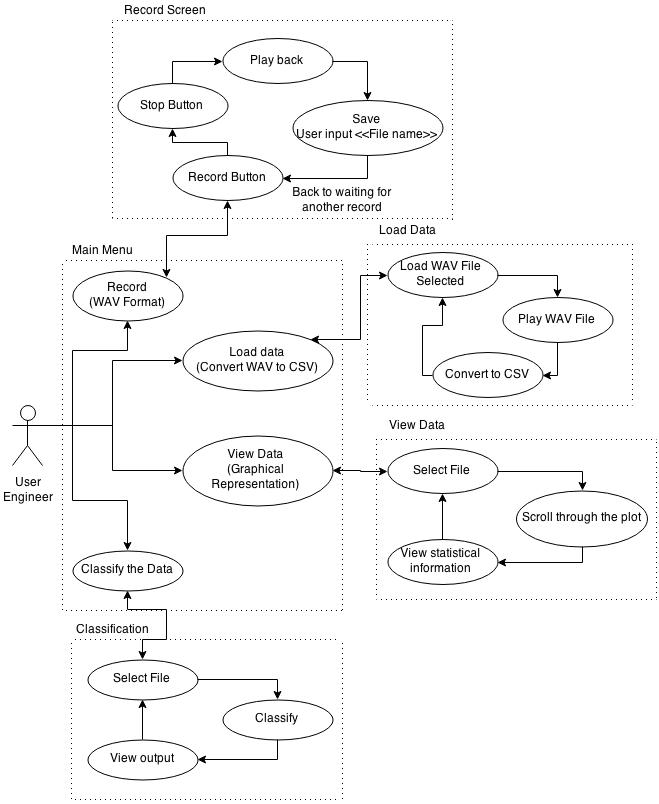
**Use Case Diagram** 

Figure -- Use case displaying how an engineer will be using the App

Figure 2 demonstrates the four principal functions of the App, which can all be accessed from the Main Menu. The classification function is the core requirement (see section 6.1) stated by EA Technology.

**Recording**

Once the EA instrument is attached to the Android device, the user can select the recording screen from the main menu and begin recording. The app will prompt the user for the reference/title of the audio file, which is then automatically stored internally on the device. It then resets itself in order to record a new audio file.

**Loading/Converting the Data**

This function is separate from View Data and Classification to clearly demonstrate the different sub-functions, and also to simplify the testing stages. In this screen, the user will be able to search the files stored on the device, listen to them and convert them into a Comma Separated Values (CSV) formatted file which will be stored in a separate location on the device.

**Viewing the Data**

This functionality allows the user to view the converted data in a graphical format and to see the statistical data for each audio file. If the audio file has already been classified this will also be shown here.

**Classifying the data**

With a similar file selection system to the other functions on the app, this screen will return confirmation of if the audio file shows signs of partial discharge or not.

### Structural View

The structural view describes the static structure of a system, without focussing on the behaviour of the application. A Class diagram is one of many forms in which one can view a system structurally. Class diagrams consider the interactions, including the exchange of data, between classes.

**Class Diagram**

See 6.3.1 Class Diagram for the full Class diagram. These diagrams have been made via Visual-Paradigm see section 6.4.

The Class diagram illustrates a static view of the overall system, which has been split into two sections; the main system and the extracted features. This is because the features section uses source code from jAudio, which has been altered to suit the project, however the code for the main system has been written for this project.

Each Activity within the user interface extends the action bar from Android’s core system, which allows the possibility of a menu bar for settings and other important user options. MainActivity is the principal class, and is the first be to opened upon starting the Application.

The MainActivity’s corresponding Extensible Mark-up Language (XML) file holds the user interface to be loaded by the class. The two buttons in the XML file point to two methods in the MainActivity class, the user then selects one of two buttons, either FileChooser or Record.

The FileChooser function will be where the user selects the audio file that has been saved; this will pass the selected file name as a String to the LoadData class.

The LoadData class will start a thread to load the data and to initialise a new AudioStruct. A new thread is created, as the load time could potentially be lengthy, therefore keeping the user interface responsive for the user. The AudioStruct is passed to the classifier that holds all the known average values for both partial discharge and non-partial discharge. Then the closest matched instance for the new audio data is found, this is the classification. AudioStruct keeps all the relevant data together, thus allowing the system to access the information at any point.

The purpose of the Recording class is to allow the user to record and save an audio file; to do this it calls the ExtAudioRecording class, see section 6.6 for more detail on this class. This class will take the input from the microphone and save it to a temporary file called ‘tmp.wav’. By saving a temporary file we allow the user to playback the audio before deciding if it needs to be saved.

Once the audio has been recorded the user can call the SaveAs method by clicking the relevant button, this will ask the user for the reference/ title as a text input and to confirm before changing the temporary file to the new name. The text input will have to be checked to prohibit the user entering anything harmful; in Android you can limit the characters entered within the XML file.

### Behavioural View

A Behavioural view of the system demonstrates how the application changes state from external influence.

**Sequence Diagram**

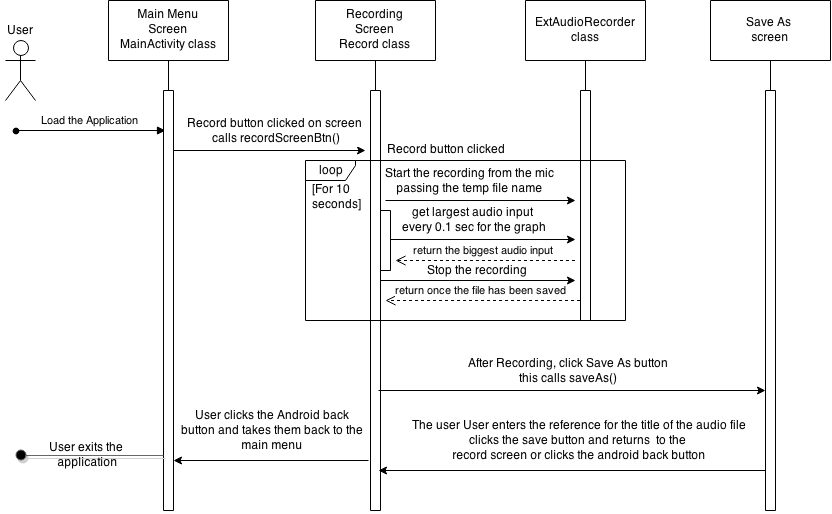


Figure - A sequence diagram showing the user recording new audio data

Figure 3 illustrates the user’s interactions from starting the application to recording and saving new data. This diagram is an ideal interaction, though the user may choose a different path, for example by clicking the back button at any time, which causes the application to be able to handle these different interactions.

The user having started a recording and the countdown begins, a feedback loop of the volume levels detected by the mic is also provided so that the user can check the audio is being recorded. If the user hits the back button mid record, that recording will be lost and a new recording will have to be initiated. At the end of the 10 second recording, the user is asked for the reference/ title, clicking the back button here would take the user back to the recording screen and again the just recorded information will be lost.

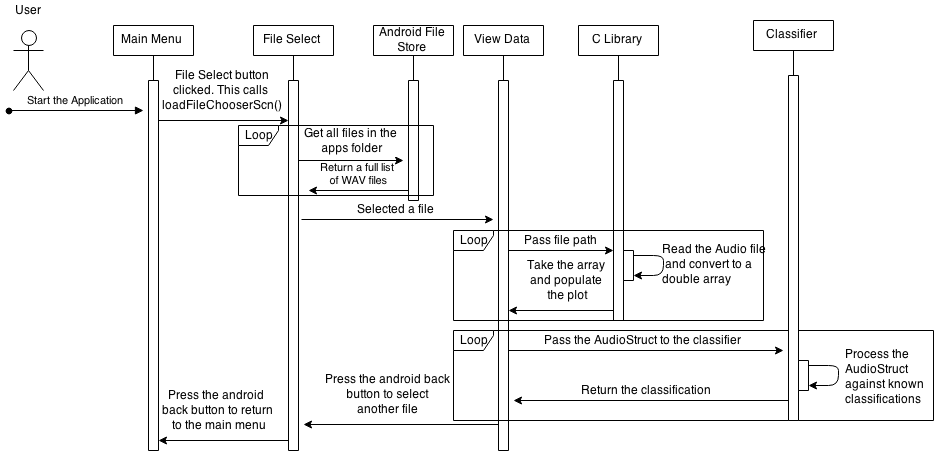


Figure - A sequence diagram of the user viewing the data and classification

Figure 4 shows the preferred flow on how to start the application, select the audio file and view the data with the classification. Once the user has selected ‘File Chooser’ from the main menu he/she will be presented with the list of WAV audio files contained in the applications folder, and they can then select the desired file. If there are too many files to show simultaneously in one screen, then the user can scroll through the files with a swipe of their finger or the scroll bar. Having selected the file, the file path and name gets passed to the C Library where it is then read and converted to a double array. All the information of the audio file is stored in an AudioStruct, after the loading of the file it will be sent to the classifier, where it is matched to the closest already classified instance. The classification is then shown to the user on screen.

For both sequence diagram Figure 3 and Figure 4 the default Android back button is used. The Android best practises state that the developer should always use the default back button as opposed to creating another. This allows for a universal flow of any application across all devices to keep with the users expectations.

## User Interface Design

**Mock Screen shots**

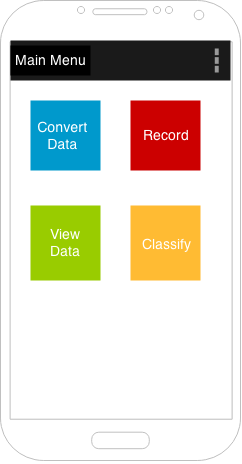


Figure - Mock screen shot of the main menu

Figure 5 is showing the mock main menu of the application, the design had to be clean and simple as the end user will be an engineer will be in the field focusing on other tasks. The engineer will be in full health and safety gear one of which will be gloves this is the reasoning for the big clear colour coded buttons.

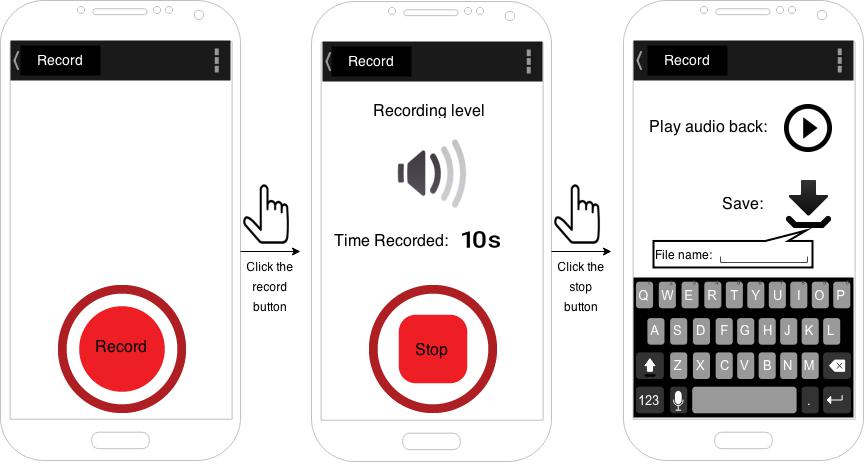


Figure - Mock screen of the recording process

Figure 6 is demonstrations the process to record the audio and how the users interactions will change the screen. The record button is as big as it can be on the android screen; this is for the ease of use for the end user. By having the speaker to provide a feedback, once the user has clicked record it allows the user to confirm the volume level of the recording. This feedback gives the user the confidence that the application is works as expected.

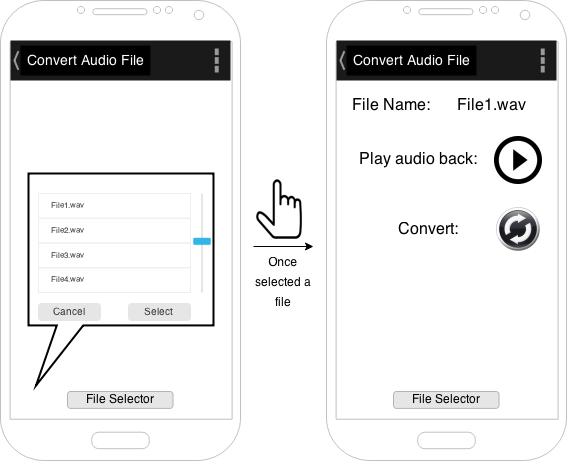


Figure – Mock Convert Audio file to data Screen

Figure 7 is showing how the user can convert the data from the audio WAV format to a CSV file. The file selector screen will be the same across all screens and will provide the user with the list of files save on the device.

Once the user has selected the file it takes them to the convert screen that allows the user to playback the audio file or converts it.

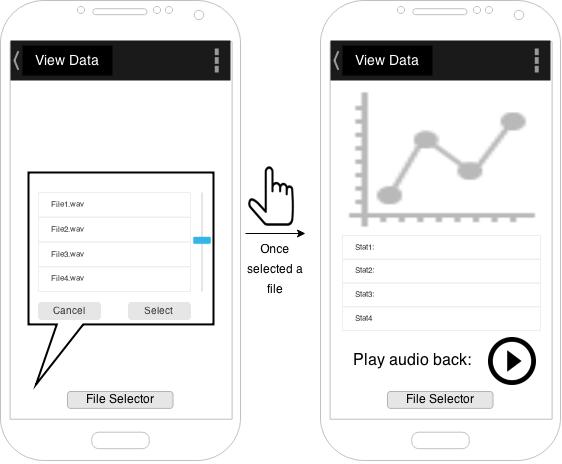


Figure - View data screen

Figure 8 is how to show the data once converted. The select file window will only show the converted data (file ending in .csv). It will show the plot of the data, the features extracted and allow the user to playback the audio and select a new file to load.

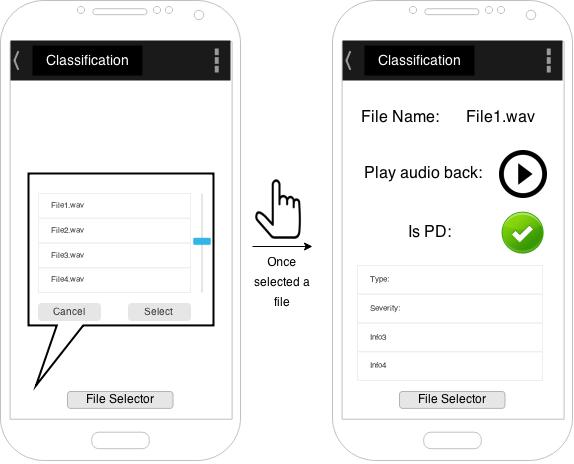


Figure - Classification screen

Figure 9 is the classification of the audio file selected. It will show the user the list of file that is available to classify, then display the classification along with the extra data extracted. For example the type and severity of partial discharge detected. It also allows the user to play back the recording.

## Other Relevant Sections

**Data storage**

The recorded audio from the application will be saved within the applications’ own folder created by the Android operating system upon the install. The purpose of this feature is for security ensuring that the data is not corrupted by other applications.

Android offer an SQLite database, which is implemented inside the Android operating system. The SQLite database is, however, superfluous to this particular project.

**Data structures**

Due to the large quantity of information to be stored whilst loading an audio file, it is most efficient to collate this information into a structure. Some of the variables to be stored include: file name, sample rate, channels, an array of the audio, window size, and features extracted from its data. All of these variables will need getters and some of which will have setters.

# Implementation

This section will describe how the Android application was implemented and the issues encountered along the way. The section is approached from the top down, by first discussing the final system, then considering the issues that arose and the spike work that this resulted in.

## Final Application

The final application followed many iterations and feedback from EA Technology. From the original conceptual design, the final outcome on the user interface and the classification method were both very different.

EA Technology received the initial mock User Interface very warmly, although they had some feedback requiring some minor alterations to be made. The main menu was accurate to their requirements as the buttons were clear, well separated and large enough for an end user with gloves to be able to use (the original user requirement specification). The minor alterations that were fed back from EA Technology were regarding Recording and Classification. These have been discussed in further detail below.

### Recording

The modifications to the recording screen that were suggested were;

* To ensure that the screen portrays feedback to show the user when it is recording audio.
* To give the user the option to record for a set time frame or an automatic 10 seconds.
* To amend the option of how to save and when to save the audio file once recorded.

To expand further on the audio screen feedback; originally the Application was built with a volume feedback system where there were three levels indicating the volume were inputted. After working through a number of options for the user to review and choose from, a decision was made that a plot of the incoming audio would be best, rather than the speaker image. The main reasons for this was that the engineer could potentially also see Partial Discharge as well as listen to it. See section 6.8 for the final screen shots of the recording screens.

After solving the issue of the audio screen feedback the next modification to approach was the option of the recording length for the user. Previously the user could record for any amount of time, which pushed the limitations of the Android device when reading and classifying the data. After a small discussion with EA Technology a decision was made to only record for 10 seconds, therefore the user would only be concerned with starting the recording and was then free to focus on other jobs. Also this modification ensures that the Application is not limited to running on certain devices because it reaches the minimum specification for all Android devices running 4.4 and above.

Once the other modifications were integrated in the application I then turned to modifying the different save options for users. Initially, after each ten second recording the user was required to enter the reference/title of the file regardless of whether or not they wished to save the file. To overcome this, the recording was amended so it would be saved as a temporary file called ‘tmp.wav’ which allows the user to playback the audio and then decides if they wanted to save it.

It was beneficial to have EA Technology’s feedback and their quick response to any questions that arose; this allowed for the modifications to be concluded on time even with the difficulties encountered.

An example of an unforeseen difficulty is that the audio file was required to be in WAV format, which can be saved in many different sample rates and have up to 4 channels. My knowledge on how to handle this within the Android framework was lacking but after some spike work I found a tutorial and an open source library in java to use, called ExtAudioRecorder. The library was comprehensive and offered many functions, of which I only needed to use the recording class. Because of these setbacks, implementing the Recording function proved to be much harder I initially thought whilst creating the Gantt chart, though I still managed to complete it within the allocated time frame.

ExtAudioRecorder class needed to be edited to be implemented in the current project, however, the edits required were only minor. For example, the ExtAudioRecorder required various pieces of data to create a new recording such as file path, name and format. Since the scope of this project is limited to only one format (WAV), I disabled the irrelevant code. I chose not to delete it altogether as this function may be useful in later versions of the Application.

### Classification

Unfortunately due to the Confidential Disclosure Agreement (CDA) with EA Technology, I experienced a major setback as I received the data 5 weeks later than originally anticipated. Therefore I had to prioritise what could be achieved in the remaining weeks of my project.

Despite not having all the data for such a long time, I had been given one single data set, which I could do some preliminary spike work on. For example I was able to look into how to read the audio files and extract some statistics. Once I had received all the data I was able to fully analyse the features I could extract.

After doing a Fast Fourier Transform (FFT), the Skewness of the data gave a particularly interesting outcome. If noise was present then the FFT plot was constant, while if partial discharge were present then the plot would be larger down at the lower frequencies. See section 6.5 for the plots and descriptions. While these results looked promising each result was taking a long time to calculate.

Following a communication with my supervisor he had suggested that I try and find a feature extractor tool to help accelerate the progression of my project. I researched the options for audio feature extractor tools and subsequently chose jAudio, which allowed me to read in the audio files and select which features to extract. This extractor tool has features from which I chose 9. This allowed the average and standard deviation for each to be calculated, ending up with 18 features in total.

jAudio works by having a window size of the array. The Fast Fourier Transform requires the window size to be a power of 2, and after some consideration I chose 512. This is because 1024 would have resulted in less overall windows and thus a lower resolution of clarity with the feature results. On the other hand 256 would create windows that were too small to provide a comprehensive overview of the data. If the final window did not match the required size it would be extended with zeros.

The 9 features are:

1. Compactness works by comparing each window to its adjacent neighbours, and computes how similarly valued they are. A smaller compactness implies a better match of windows. It also considers the magnitude spectrum of the neighbouring windows to more accurately compare the windows. Much like Skewness, compactness provides a good indicator of noise.
2. Root Means Squared (RMS) is the measure of the power of the signal. This can be very helpful in finding the partial discharge over noise within the audio file.
3. Spectral Centroid is the centre of mass of the power spectrum. It is often interpreted with a connection to the ‘brightness’ of a sound file.
4. Spectral roll off point is the fraction of bins in the power spectrum at which 85% of the power is at lower frequencies. It is a measure of the right-Skewness of the power spectrum.
5. Spectral Variability is a measure of the variance of a signal's magnitude spectrum, and it is calculated from the standard deviation.
6. Strongest Frequency Via FFT Max is the strongest frequency component of a signal, in Hz, found via finding the FFT bin with the highest power. This needs the FFT Bin Frequency Labels and the power spectrum.
7. Strongest frequency via spectral centroid is similar to the Strongest Frequency Via FFT Max but using the spectral centroid instead of the FFT bin with the highest power. It needs both the spectral centroid and magnitude spectrum.
8. Strongest frequency via zero crossings is the strongest frequency component of a signal, in Hz, found via the number of zero-crossings. This requires the zero crossing value of the window.
9. Zero Crossing is the number of times the waveform changed sign. It can be an indication of frequency as well as noisiness.

Some of these features require extra information such as the power spectrum or magnitude spectrum. Upon calculation of the Fast Fourier Transform, the Magnitude Spectrum Value for each bin is found by first summing the squares of each component. The Magnitude Spectrum in the square root of this divided by the number of bins. The Power Spectrum is the simply the Magnitude Spectrum Value divided by the number of bins. The FFT bin frequencies are the bin labels for each power spectrum and magnitude bin.

Each feature is extracted for each window within the WAV file; from this array we can calculate the average value and the standard deviation, this makes up the 18 features.

With these features now extracted we split the data up into 60% training data and 40% testing data, so that the classifier does not fully over fit to the training data. The 60% training data will be split into Partial Discharge and non-Partial Discharge, the aim was to put the values into an excel spread sheet to plot a graph and see if there is any separation between them, see Figure 10.

Figure -- The Average of features of partial discharge against non-partial discharge

After inspection I found that the best feature to separate the values was the compactness average. To help further analyse this data I took an average of the other individual features to get rid of any outlying data, when comparing one of the testing audio files it was still clear if it was Partial Discharge or not. See Figure 11 below.

Figure -- plotting an unclassified audio file against the known classification features

Unfortunately given the time constraints on the project I have been unable to fully research all of these features and how they may impact my work. However jAudio allowed me to use features I would otherwise not have been aware of thus providing a better platform to classify the audio files. Since jAudio was not my own code, I kept it separate in my project. One of the most difficult aspects of working with this new code was how to order the feature extraction, as the features all worked off an abstract class from which all other features extended. I had to ensure that each variable required in the consequential formulae was already calculated. For example the Strongest Frequency Via Spectral Centroid feature depends on the Spectral Centroid and Power Spectrum but the Spectral Centroid requires the Power Spectrum, so it was pointless to work out the Strongest Frequency Via Spectral Centroid first.

The classifier was constructed using the main idea of minimum distance from the average value for each feature of the new data, i.e. for both Partial Discharge and Non-Partial Discharge. With the average values hard coded into the classifier, the minimum distance to each new audio feature is calculated, and whichever class the data is closer to provides the feature sub-classification. The 18 features supply 18 sub-classifications, which are then totalled to classify the data.

The certainty of the sub-classifications compares the distance between the Partial and Non-Partial Discharge average values with the distances between the two averages and the test data. With these distances you can compare, as a percentage, whether the new data is very close to one of the averages, so a high certainty, or only slightly nearer one average than the other, so a lower certainty. If the new data lies outside of the range of both averages, then the certainty is 100% to the nearest classification.

For example given a feature X, with average and standard deviation for both Partial and Non-Partial Discharge given in the table below, the algorithm can be computed with certain test data to classify the file. Assume that the new audio instance passed in to be classified has an average and standard deviation as below.

|  |  |  |
| --- | --- | --- |
| Partial Discharge |  | 50 |
|  | 25 |
| Non-partial Discharge |  | 60 |
|  | 15 |
| New Instance Feature |  | 40 |
|  | 19 |

First the algorithm calculates the closest match (minimum distance) to the average data:

The first function creates a tally system wherein the feature with the smallest difference is given a value of 1. In this example, adds 1 to Partial Discharge and adds 1 to Non-Partial Discharge. This is repeated for the remaining 16 features.

Next the certainty is calculated for both features, which requires the difference between range of each feature. In this example:

The certainty value is finally calculated by dividing the by the .

A certainty of 1 is equal to 100%. This is as expected as the of the new instance is beyond the Partial Discharge value, which can be seen on the graph, Fig \*\*\*\*\* (Simple table of the xAvg values).

For the standard deviation certainty in this example:

Therefore

Let us now assume that we have calculated these values for each of the other 16 features and arrived at an overall tally and certainty.

If the tally gave 15 values in Partial Discharge and 1 in Non-Partial Discharge, then combining this with the 1 tally for each from the average and standard, then the final tally is 16:2. Therefore the file is classified as Partial Discharge.

The certainty for all Partial Discharge features is summed and divided by the total number of features, which is 16 in this example. Note the certainty will always lie between 0 and 1.

If the tally gave us 9 Partial Discharge and 9 Non-Partial Discharge values, then in order to seek a classification the algorithm considers the certainty. In the rare circumstance where the certainties are equal then the algorithm returns the classification of Non-Partial Discharge.

See section 6.15 for the pseudo code of this algorithm.

## Spike work

This section will cover the early spike work I progressed as a necessary step to create the final Android application.

### Understanding Android 4.4+

My previous experience has been in Android 4.2, so understanding the differences in the later version 4.4 + was necessary. One of the main differences is the storage framework, where the application can only access its own folder. This was the first challenge I addressed, and I was able to call the Android built in path method to the internal storage that meant it always return to the applications folder. This meant that the impact was minimised and only a small change to the code did not affect the overall function.

### Loading WAV Audio files in Android

I started by loading the audio data into Scilab, and saving the arrays as CSV files. These CSV files were then available to be read in the Android application. This allowed me to create the foundations for the application as I could use the data as a double array.

The next step was to try and read the WAV file in Java to convert it to a double array. I expected this to be simple as playing audio files in Android uses the media player, which is already built in. However the media player was unable to convert the file, and Android had no other pre-existing functions capable of conversion. EA Technology uses a C library called Libsndfile to read the WAV files. Therefore I had to add C library support into the Android Application.

I managed to get a very basic application working, this just passed information backwards and forwards between C and Java. It worked by making an extra folder called ‘jni’ and building/compiling the code within ‘jni’ with the Native Development Kit. To compile the C code I had to create a makefile and run this in a terminal, the debugging feedback was very limited in this method. Sometime the C code would compile and then the application would stop running and give no indication on how it failed, which caused some delays to the development of the Application. The make file Android.mk holds the library name, the files to compile, and a list of dependencies to create a library for the Java code. When building the native code there is a function that creates the C header file that the Java code calls, therefore creating the functions was simple.

I then needed to add the library Libsndfile into the Application. This was difficult as Libsndfile is an external library to be deployed into the Android Operating System. Fortunately Libsndfile has a very large community of users thus I was not the first to encounter this issue. Another developer released their code under the GNU lesser General Public License, which resolved this issue. I was able to implement this code within my Application, which meant that I didn’t need to adapt my previous spike work to read WAV files with the Libsndfile C library.

I also discovered that I no longer needed to convert the data into a CSV file because reading the audio file took longer to process into doubles seeing as it is first read as a String before being converted. I discovered I could convert the WAV file directly into a double, which is much quicker as a WAV file is in binary form and therefore the conversion is a singular process. This process accelerated the file loading by 10 times, meaning the data classification was possible on each load.

After this I will only need two options from the main menu screen, ‘Record‘ and ‘Classify’. This will cut half the program as I no longer have to convert, save and read CSV files. However I still kept all the code in case this feature is needed at some point in the future. For example the user may wish to export more than just the audio as an array, such as the classification and other captured information.

### Feature extraction

Initially, EA Technology was able to provide one audio file and informed me this contained partial discharge recorded. With this sole file it was possible to test some of the theory which I had researched. The file had very prominent pattern in the audio see Figure 12, which I was then able to analyse.

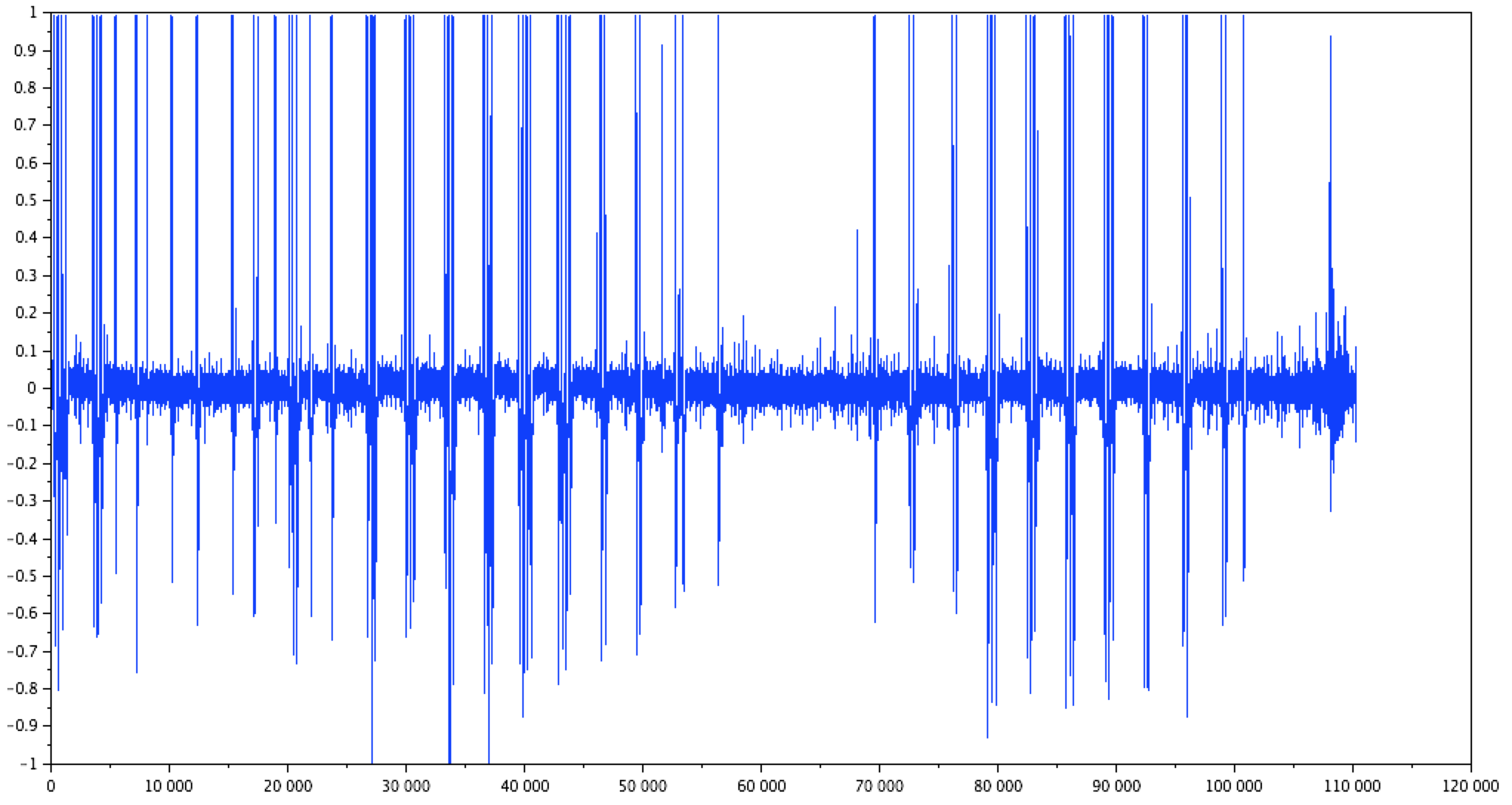


Figure - Plot of the test.wav file

I calculated the distance between spikes and the duration of the full spike, and following this I also calculated the average, variance and Skewness. However without any other files to compare to, I was unable to progress this analysis further.

Upon receiving the rest of the files from EA Technology I began analysing other features which I could use to classify the data. I ran the training data through the same initial code which I had created with the first audio file, in an attempt to see if any were linearly separable. Unfortunately this was unsuccessful, so I attempted the same analysis after applying a Fast Fourier Transform on the audio files. A Fast Fourier Transform shows the frequency response and helps to separate partial discharge from noise.

Scilab is an open source program for numerical computation providing a powerful computing environment for engineering and scientific applications. Scilab has many built in functions to help process the audio data as it can read in in the audio WAV files, run an FFT and display the results. Scilab allowed me to quickly test some of my ideas rather than first spending time programming these features in C or Java. The results from this experiment see section 6.5, show that the result of an FFT can help distinguish between Partial and Non-Partial Discharge. However it provided slightly different results to that I expected, for example Skewness did not provide an adequate classification feature. Unfortunately, this meant a lot of the initial spike work was rendered useless once I had received the rest of the data from EA Technology.

## Issues that arose

In order to accelerate the analysis into feature extraction, I used an existing program with extraction tools built in. Once I had found a program that offered all the functions I required, I was able to fully analyse the difference between Partial and Non-Partial discharge which I could then integrate this into my existing Application.

Whilst trying to find an appropriate program, I assessed many different options and extraction tools. For example I downloaded YAAFE (Yet Another Audio Feature Extractor) which is a C++ library. YAAFE uses many dependencies to run, one of which is Libsndfile, the library which reads the audio files that I have used in the Application. A full list of features that YAAFE extracts can be found in section 6.9. Unfortunately some of the dependencies YAAFE required for the feature extraction would not port over to the Android environment, so I could not use it in the Application.

The website Stack Overflow (section 6.14) was vital in overcoming many of the minor issues I encountered with my code. For example, as part of my Application, I had to be able to search for the WAV files a folder. Stack Overflow provided a detailed example of this function, in which a given directory is searched for the correct format, which in this case is ‘.wav’, and returning the matches as a list.

For my other coding issues, if Stack Overflow was unable to provide a solution then I was still able to use the website as it often includes very similar scenarios which I could adapt to my own issues.

My experience in creating Android applications helped enormously throughout this project. However I had not previously encountered adding Native Support into an Application, which I needed to be able to use C libraries such as Libsndfile. Thus I began by using resources on the Android developer website [15] to create a simple application that supports Native language.

Adding in the library and the dependencies proved to be a greater challenge, as there was only limited debugging information whilst building inside the terminal. At first, the build would complete but when it came to running the application, it would close with a null pointer. The lack of debugging feedback meant that this process took longer than I originally anticipated.

I was able to break down the functions in the Libsndfile code to assess small portions at a time, thus overcoming the errors more quickly. I first focussed on passing the data across from Java to C. As a String is not a set type in C, I had to use a char array to hold the information. I called a function from the environment, called GetStringUTFChars, which returns the String as a char array and can then be accessed in C code.

# Testing

## Overall Approach to Testing

Testing is an integral part of any software program, and as a result Google have provided many testing features into Android. The Android framework offers JUnit, instrumentation, test case and assertion testing, which collectively ensure that the application will not fail, stop or behave unexpectedly. The XP processes get the user to create the tests when creating the classes, this process has been integrated into this project.

Considering this project mostly consists using Java, JUnit testing was incredibly useful in testing the logic of the algorithms, the return values expected from the third party code and the user interface components.

Libsndfile comes with built in test functions, thus I didn’t need JUnit for testing the C code. However as my Java code interacts with the C library, I was able to use JUnit to test the process and return values.

## Automated Testing

### Unit Tests

Writing the test code concurrently to writing the Java code meant that I could test my code as I was writing it. The IDE Android Studio allows the test classes to be made automatically, it creates individual tests for each method. These tests allow for continuous integration when either refactoring or implementing new functionality.

When creating a test class it can extend many different way of testing, for example we can test the UI variables by extending InstrumentationTestCase or the general logic by extending AndroidTestCase.

I split up the tests created for the features and the rest of the system; this was done as both these parts were in separate packages. The tests are run on the Samsung S4, the device the application has been developed on.

For JUnit testing, the setUp() method was used to create a double array 80,000 in size. This size was chosen as it is the size of a 10 second WAV recording at 8,000 sample rate which is a default setting. Each array is populated with the value 1.0 to allow the features to be easily calculated and to check the correct return value. While this does not test real values expected I was able to create the answers in Scilab and confirm the answers with assertEqual() methods.

Some features require extra information, such as Compactness needs the magnitude spectrum, so I pass the array twice. Other features return an array of the processed data, as in the FFTBinFrequenciesTest class. Because all the calculations in the array are identical, checking the first array value was equivalent to checking the whole array.

After testing the array at maximum size, i.e. 80,000, I then tested it at the minimum size of 1. In reality the minimum size is zero but this requires no testing.

A limitation to JUnit 3.0 is that it does not handle exception testing; passing a null array to one of the features throws an exception which can then not be tested.

### User Interface Testing

I created some User Interface Unit Tests to ensure that the local variables are assigned and not null pointers. Therefore whichever screen the user is on, all the variables and listeners are created.

One of the problems I encountered during this process was trying to create the correct environment for the tests to run in. My Activities extend ActionBarActivity to allow the Application to use a pop up menu, thus when unit testing the user interface the onCreate method is called before the theme was set. This returned an ‘IllegalStateException’, with the explanation:

*“You need to use a Theme.AppCompat theme (or descendant) with this activity”*

All that was required was the theme being set before onCreate, and I discovered two solutions to this error. The first was to change ActionBarActivity to Activity, which removed the action bar theme and options, and set up as a blank Activity to then inflate the XML file. Despite this solving the issue of having a null theme, this solution was impractical because it would remove the whole action bar, including the title and the settings option, from each screen.

The second solution was far more simple and elegant. In JUnit testing we can build a ContextThemeWrapper, in which we can set the theme to the context. Applying this before setting and starting the Activity resolves this issue thus ‘IllegalStateException’ is no longer returned and all tests pass successfully.

### Stress Testing

To stress test the application in JUnit test we set the full array size to 80,000. While this is just 10 seconds of recording at 8KHz sample rate the features only get the window size of this (default is 512), by passing the full array we give each feature its full capacity.

To maximise the success of stress testing, it is ideal to test the application on multiple devices. Unfortunately I only have access to a Samsung S4, however Android Studio allows you to create a virtual Android device and set the hardware specification to your own requirements. The testing on multiple virtual devices raised no issues, though it ran much slower on lower specification devices.

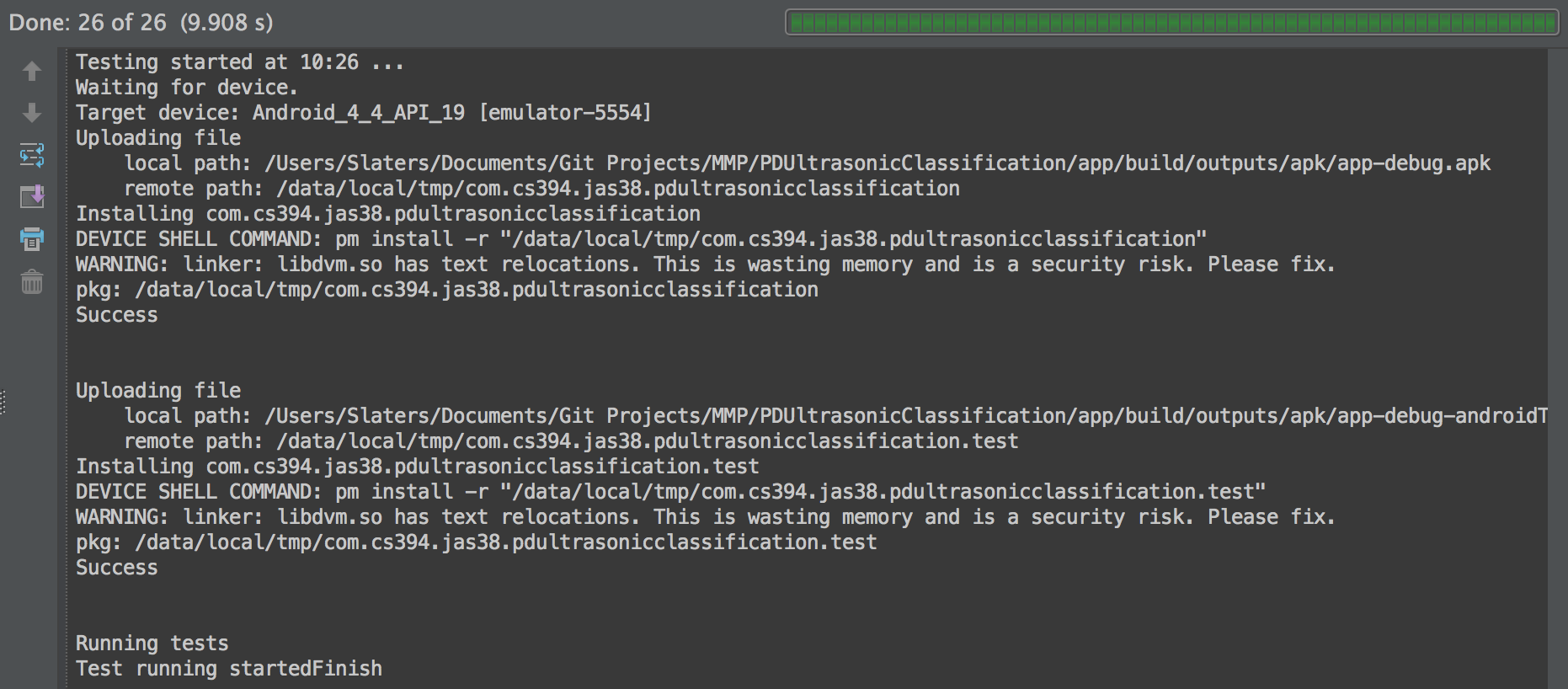


Figure - Android 4.4 API 19 Emulator

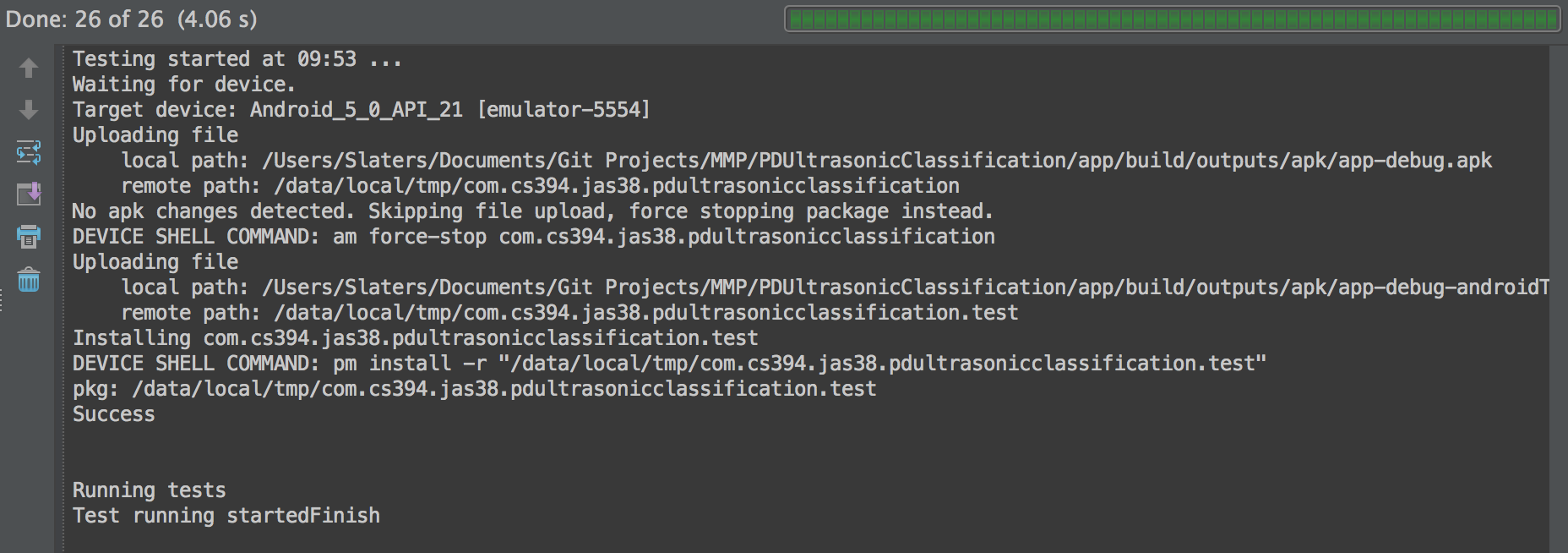


Figure -- Android 5.0 API 21 emulator

Figure 13 is running Android 4.4 on a Nexus 5 x86 processor emulated device. With only 256MB of RAM and 200MB of internal memory, it is the minimum specification for Android 4.4 to run. All the tests ran and passed though took a lengthy 9.9 seconds to complete due to the lower specification.

Figure 14 runs the same test on the same Nexus 5 emulated device but on Android 5.0. The specification is much higher and completes the tests in 4 seconds. Android 5.0 is the applications targeted deployment environment and since the tests pass on the minimum targeted operating system version API 19 shows it is a stable application.

## Integration Testing

Integration testing is the software testing phase in which individual software modules are combined and tested as a group.

Using the model integration testing from extreme programming was a continuous process throughout the Application’s creation. Unit testing was beneficial to this model as it tested the individual parts and how they interact and pass data. Using Github for my version control allowed the ability to create a branch to add the next functionality or story to the application. I endeavoured not to check out a branch for too long as that would dramatically increase the likelihood of errors when integrating it back with the main branch.

Once the code was running successfully, the next aim was to make it more efficient both in run time and enabling future development/maintenance. I had a number of refactors within the code, for example taking the local variables and moving it into a structure. Again the unit testing help make sure what I changed did not affect the passing of data.

To maximise the test efficiency, the integration tests are all automated. To further test the application after integrating either a branch or after a refactor, I created some test tables, see section 6.6. This testing flagged up the areas of the Application that the unit tests were unable to test, for example some of the issues when updating the graph after implementing the AudioStruct did not work, but all unit tests passed.

## User Testing

User testing involved gathering feedback from engineers at EA Technology, which happened after most major iterations of my code. The Application was unable to be taken out into the field, so the feedback largely consisted of returning the test tables that I had sent them.

I was able to send the end users an APK\* file to run on the device. An APK file is the Android Application, much like a java program, that can be compiled into an executable jar file. Then EA Technology could run the application on their Android devices, unfortunately I have yet to hear further feedback regarding its performance.

# Critical Evaluation

This section is my critical evaluation on the project, assessing the strengths, weaknesses and the approach taken for the project overall. I will focus on the key aspects of the project and critically evaluate each, for example some of the key aspects considered include; were the requirements correctly identified? were the design decisions correct? were the tools I used right for this project?

## Requirements

Having EA Technology established as a client meant the requirements were already correctly identified. EA Technology and myself did have to have a meeting to go through what would be possible in the time frame of the project. A strength for the project was good relationship and understanding I had with EA Technology which reduced the time it took to get a finalized version of the requirements as both EA Technology and I knew what was required.

In any project the client can provide the data needed for the project, rather than the project manager finding and getting the data. For my project this was a strength as EA Technology were able to provide the data, which saved time and allowed myself to focus on other key aspects of the project.

A major weakness was the limited time factor for the project resulting from the slow CDA approval, however I was able to manage most of the other aspects to time and the only functionality I have not been able to add was the conversion of the file.

## The Client / Customer

My decision to use EA Technology as a client for this project was beneficial. It was very useful having a familiar Company supporting me  with clear goals to aim for which was better for me.

A Strength for the project was the client. The client, EA Technology, made it very clear they were more focused on the classifier and to have someone approach the problem with a new take and see if a different improved solution can be found. Looking back to the start of the project, having a good client making the specification helped as they knew what they wanted and the only major input I needed to give was to plan it in the time frame.

However EA Technology needed a CDA agreement to be signed between the University and themselves. This was to protect them from the expertly analysed data being released or used against them. I understand the need for this agreement to be sorted out and to protect EA Technology’s assets, however when starting a project with EA Technology I would of never expected the length of delay that ensued. Arranging a CDA for my project was a weakness and it adversely impacted on my timeline.

It took a total of 7 weeks for the CDA to be agreed and signed. As a result I finally got the data on the 13th March 2015, however to address the delay I worked with EA Technology, who shared some information before the agreement was signed. This allowed me to make some limited progress and to minimise the impact. However at one stage the project was so delayed by this setback that I held conversations with both my supervisor and the module coordinator about possibly starting another project as the scope of what was needed by the end and the time left was not enough. In addition the domain expert at EA Technology my point of focus, acting as the Customer for EA Technology, also changed a number of times throughout the project. The internal changes at EA Technology made the task of providing a constant up to date client relationship very hard to manage. Each new customer expert needed to be briefed and updated. By the end of this project I could not name my first point of contact at EA Technology due to their internal restructuring. This meant my requests for more data to test the classifier wasn’t met and even an EA Technology Android icon has not been supplied.

On reflection, I am pleased to have continued with the original project with EA Technology, as it has been satisfying and rewarding. The final outcome was a good working model and EA Technology did answer any question I asked and provided feedback very quickly. Whilst there were some difficult aspects to overcome in the project, by working hard and having a good communication line with my supervisor I managed to deliver an application that does the job and it allows for a good base to build upon for future work.

## Design Choices

### Tools Used

The tools used to build the final application were readily available very right for the job at hand. Android Studio was the IDE of choice and I was able to design, build, test and deploy the Android application.

A strength of the Project, followed my decision to use my final choice of the IDE. I chose Android Studio as the IDE.  As I have had previous experience with the beta versions of Android Studio and the version used from the concept of this project was the first official release of the ‘app’ it required no rework and there were no issues.  I discounted other ‘apps’ such as Eclipse, as I did not want to have to deal with all the plug-in’s that were needed for Eclipse. My evaluation showed that while Eclipse is probably the stronger IDE with a bigger community backing it setting the environment up would take longer than just downloading Android Studio.

A weakness was my inexperience in using audio feature extractor. Following the delay in data being provided, I had to use an audio feature extractor to get the features from each audio file. I have never used one of these tools before and I needed to learn how very quickly. I feel having to use a feature extractor tool was very unfortunate as what resulted in one of the main features to distinguish Partial discharge from Non-partial Discharge was an adapted look on Skewness.

With time being precious I understood the need to speed up the feature extraction and the guidance my supervisor gave me. I decided on jAudio as the program to do this as it allowed me to select the which features to be calculated. jAudio split the audio file into windows which allowed a better analysis for the overall picture. The list of features was very extensive but I decided on 9 and then calculate the average value and the standard deviation from the values each window created. Being able to take these values and find a linear separation made using the audio feature extractor worthwhile indeed, I was able to extract all the features needed for all files in a matter of minutes.

I was not able to research into if there was a better audio feature extractor available the only other tool I found offered something similar however did not offer the window size to break up the audio file. I feel jAudio was the right choice as for the amount that was accomplished in a short space of time and looking forward the ability to adapt and build upon by adding more features.

The Android device I used to develop on was the Samsung S4, I had to use this as it was the only device available to me. This device was my old phone and was very familiar to me and already set up to develop on, I was also able to use the tools in Android Studio for testing on a virtual device. This allowed me to have a full range of testing done and see how the application works finally across all Android versions, device specifications and screen sizes.

### Application Flow

The initial design for recording was to allow the User to decide on how long they wanted the audio file to be. This would have proved very problematic for storage and classification. My first attempt was wrong but following good communication with EA Technology I was able to make progress allowing the development of the recording to be quick and precise. I feel from the initial requirements my understanding of how the engineer would be using the application wasn’t good enough and this was the reason for the initial design misunderstanding. This is a problem that can happen in any project and shows that using the Agile methodology when creating the software was the correct one.

The requirements stated that the output has to be read in a WAV file.  I discovered that to do this in Java was so much harder than first anticipated and this was the reason I changed the approach to converting the WAV to a CSV. By doing this I could continue to make progress with the project as reading a CSV file was very straightforward.

However on reflection seeing how I handled the problems, this was both a weakness and strength for my project. It allowed me to continue making progress but if I had focused just on the reading of WAV it would have benefits as the load time for WAV is less and it would have reduced the amount of code written down and thus saving time overall.

### User Interface

My initial design of the android application was naive, and in the fact that I tried to separate all the functionality into separate screens. This is not what an Engineer needs nor wants. It needs to be a lot simpler, so when in the field, the Engineer just wants to record and obtain a classification. A strength of my project was my flexibility and ability to change to meet the User requirements, I made a number of upgrades to my project and noting the difference from the mock design to the final application, demonstrates to me that the application is on the correct track to be a useful tool in aiding engineers in the field of Partial Discharge.

## Android Application

An issue was the CDA required to be signed covering the information and data from EA Technology and Aberystwyth University. As previously outlined this agreement took longer than I had planned. I understood that these agreements can take time and I had allowed some extra time when planning this in the Gantt chart. However, taking an extra 5 weeks longer than planned was a real set back, and the project was very close to being completely changed because of this reason.  As discussed above the delays were addressed and mitigated where possible but the impact of the delay meant that some of the functions, such as, the classification needed in the application will not be as efficient as possible or even complete and could be better with more time.

When reviewing the final application and seeing that it meets the user requirements stated at the start of the project, I feel that the functions presented match the specification. While the user interface went through many iterations the functionality was already comprehensively completed and this helped when building the application. Only the flow changed and these were minor changes and easily implemented in an iteration.

The process used to create the application was an Agile values using extreme programming processes. Small Releases, Continuous integration, Coding standard, Sustainable Pace, Refactoring and Simple Design are the main processes I used to develop this application on time. I found the process I used to be a real strength as it allowed me to understand what was needed with some spike work, allow and welcome change to the requirements, keep the architecture of the application simple with the right amount of complexity and having small releases to provide feedback from the customer.

### Testing

Testing the application has been a success.

Due to the need for continuous integration during the project, any errors were identified at an early stage and addressed.  This process showed up the lines of code that either were not tested or the functions of the application that I had broken via a refactor.

Having the ability to use JUnit testing was a strength because of the ease of use, and the speed of the testing and the in depth analysis it provides.

Being able to split up each class to test the individual methods, making sure the logic is correct for the maths in each feature or how the data is processed from reading in a WAV. I used JUnit 3.0 to build the unit tests, this allows for testing the correct return values and making sure variables are not null. However a weakness to using JUnit 3.0 was it did not allow testing of the boundary cases, as it would not test the exception caused or if any test throws an exception. In hindsight I should have used JUnit 4.0 which allows to test the exceptions.

A weakness I have discovered from my review is that when evaluating the tests created, I have not tested/ checked to see if there is enough space to save an audio recording before initiating a recording. This check could be vital and could create a scenario which would make the application quit unexpectedly. Carrying out some additional research, Android do offer a way to get total free space on the internal or external storage and this is how I would make the check before starting a recording. To test if this is possible I would create a virtual device with less storage than needed and thus create the scenario.

### Classifier

It is positive the Classifier works for the data provided.

However it is a weakness of the project that I only have 20 files of data to work with. It would be a requirement to check the Classifier further against more data which would reaffirm that it is able to classify whether an audio file is showing signs of partial discharge or non-partial discharge is required as further work.

Using a 3rd party program for the feature extraction was really good advice from my supervisor as it allowed me to process all the test data I had quickly rather than searching for my features to extract. jAudio 2.0 is currently in a development stage and it permitted me to get the source code.  This allowed me to implement the features into the Android application and create the feature extractor.

I informed my supervisor that I had found the features and from my analysis I could distinguish between partial and non-partial discharge, and together we reviewed Classifier. He did his own analysis on the data and although I had decided to use all 18 features, we found that only a few (3) features actually can tell the data part.

The main feature was the compactness average, as this showed the signal against the noise present much like Skewness. From my supervisors analysis using a 3rd party program WEKA, running PART, decision trees to neural networks using data I had extracted the Compactness Average was always used almost 100% of the time to tell partial discharge from non-partial discharge. See Figure 15 below of the summary from a PART decision evaluation using WEKA.

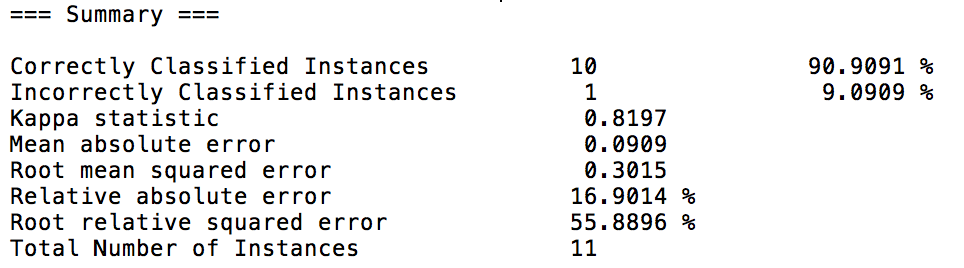


Figure -- WEKA summary of a PART Analysis

In summary of Figure 15 it shows it is 90+% effective in classifying partial discharge from non-partial discharge. In Figure 16 below it shows the rule it creates to manage this. As long as the Compactness Average in below or equal to the value 1536 it is classed as partial discharge. I do believe this outcome is really due to only having a total 11 audio file to create the rule base from.

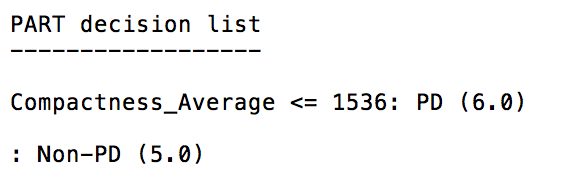


Figure -- The PART Rule created

Knowing this I still decided to use all 18 features as for now may be only Compactness is the useful feature but for the future work trying to take this a step further and being able to separate what type of partial discharge is being detected, the other features may be needed.

Each feature has a non-partial discharge and partial discharge value that is compared against any new instance, however I came across a problem while analysing the application and testing the classifier.

The difference in the stored values for both partial and non-partial discharge used for the certainty evaluation, if this is a very small difference (lets say 0.01) and the  value of the new instance is close to the middle, the certainly may come out at 100% in theory but in practise this is far too close to be able to say with 100% certainty.

A solution to this problem would be to add weights to each feature. This would allow each feature to show their importance to a classification.

### Degree Scheme

Selecting the degree scheme Artificial Intelligence and robotics there were so many modules that helped to understand, design and create the final application. Some of the modules are, Machine Learning, The Artificial Intelligence Toolbox Part 1 and part 2, Program Design, Data Structures And Algorithms and User Centred Design And Human Computer Interaction. These all played a part in me being able to do this project while there are other modules that I have not named that helped me develop overall but not directly help towards the project. The degree scheme really helped me in being able to finish the project and I have no regrets in the module choices I made, for example the Agile module in the 3rd year helped with the overall model and picking the relevant processes to help make this project a success, the User Interface module really helped me to understand and think how the end user will interact and how to make a universal feel so all users in any country.

The degree also had a Sandwich Year in which I sent my year at EA Technology, I was able to learn and experience so much while there. One of the main things I took away was to keep a diary of my day-to-day activities. The diary really helped when making the final report so I decided to do the same for this project. I found a real strength in having this diary when creating the report, it was very beneficial as I was more editing what I had written rather than having to write the report from scratch.

### Improvements

The current application while works well, could have minor improvements made. For example there is a problem that can occur and to implement the solution would improve the stability and efficiency of the classifier.

Another improvement would be to put the check in storage available before recording new audio file. This is a simple update and again offers more stability when running and using the application.

Exporting was not on the requirements however could be a really good function to improve the application. As the CSV write code is part of the application just not used exporting the results would be possible at not to much expense, and sending an email in Android is also very straightforward and again be quick to integrate this function into the application.

When evaluating improvements they all could have been achieved by better time management, and this is then something else I could have improved on. While the main reason for being time constrained was out of my hands, spending less time on functions like reading and converting an audio file to CSV when in the end all the function is doing is reading the audio file, would have meant more time on these improvements.

* 1. **Starting Again**

A very good and valid challenge regards any project is; ‘If you were starting again, what would you do differently?’

If I had the chance to start everything again, sadly I would pick a different project.

I only feel this as the difficult situation I found myself in at the halfway point suffering significant programme delays due to the CDA approval between Client and University. By picking a different project, one that did not involve a Client would have achieved this and allowed me to focus totally on the Project details.

However if I re-started this project a big weakness of the project was the CDA, and I would have made the CDA be signed before starting the project and this would have made the weakness a strength. A CDA agreement can always potentially take time to solve and in the Gantt chart I had allowed for a delay and knew what the next job was and did manage to fill the lost time with another appropriate job.

This would have put more pressure on myself over an exam period but would have saved me up to 7 weeks in the project.

Plus a positive from progressing a CDA demonstrated an element of industrial realism and helped me understand the hurdles presented to projects

I would want to focus on the separate part

* 1. **Future Work**

The next big steps in the project would have two aspects to it, Real-time Classification and to be able to classify which type of Partial Discharge is being detected or noise being detected.

Both these functions would take some time to implement, though the real-time classification would be best to do first.

This would be possible if we take the raw input as a double array and have a running average of a features being extracted, then pass them to the classifier. This would be very computationally high for an Android device as there would be a lot of data to process, this may even make this unachievable currently. Some further research into this matter would be needed into this area to better understand what would be needed.

To be able to classify what type of Partial Discharge is being detected would take more already classified data and the analysis of an domain expert. This would require EA Technology further assistance in both providing the data and to provide the expert analysis.

For the application alone it now needs to go into Beta testing on the Google Play Store, and allow select end users test it further. EA Technology have a developers account that will be available to provide this functionality, however the classifier will need to be further tested with a wider range of data.

* 1. **Overall**

Having now completed the application and the final report, looking back I can say, that overall I did enjoy this project, despite the challenges it presented. I was able to learn new skills and apply them to a real world situation, for example dealing with a customer who provided information, data required and feedback when needed, and enduring the highs and lows that can inevitably happen during a project.

Neil MacParthalain [ncm] my supervisor has been a great help all the way through this project, offering guidance and support where and when needed. I do not believe this project would have come this far without Neil’s guidance, and I owe Neil a big thank you for this. Neil enabled me to focus on set tasks and guided where it is best to focus more of my efforts.

The Agile way is to keep the application as simple as needed, and implementing the Artificial Intelligence technique Minimum Distance allowed me to have the correct amount of complexity to solve the classification problem. Looking back using a hybrid of the two models Waterfall and Extreme Programming really helped get the job/functions done and finished within the set timeframe.

This project has allowed me to delve into the Android development of application even further, and I have learnt many new skills, for example applying the support for native C/C++ code into an app and being able to call this code from the Java classes.

The final project version of the App is successful and delivers assistance to the Engineer, hence I achieved my initial goal.

# Appendices

## Project Specification

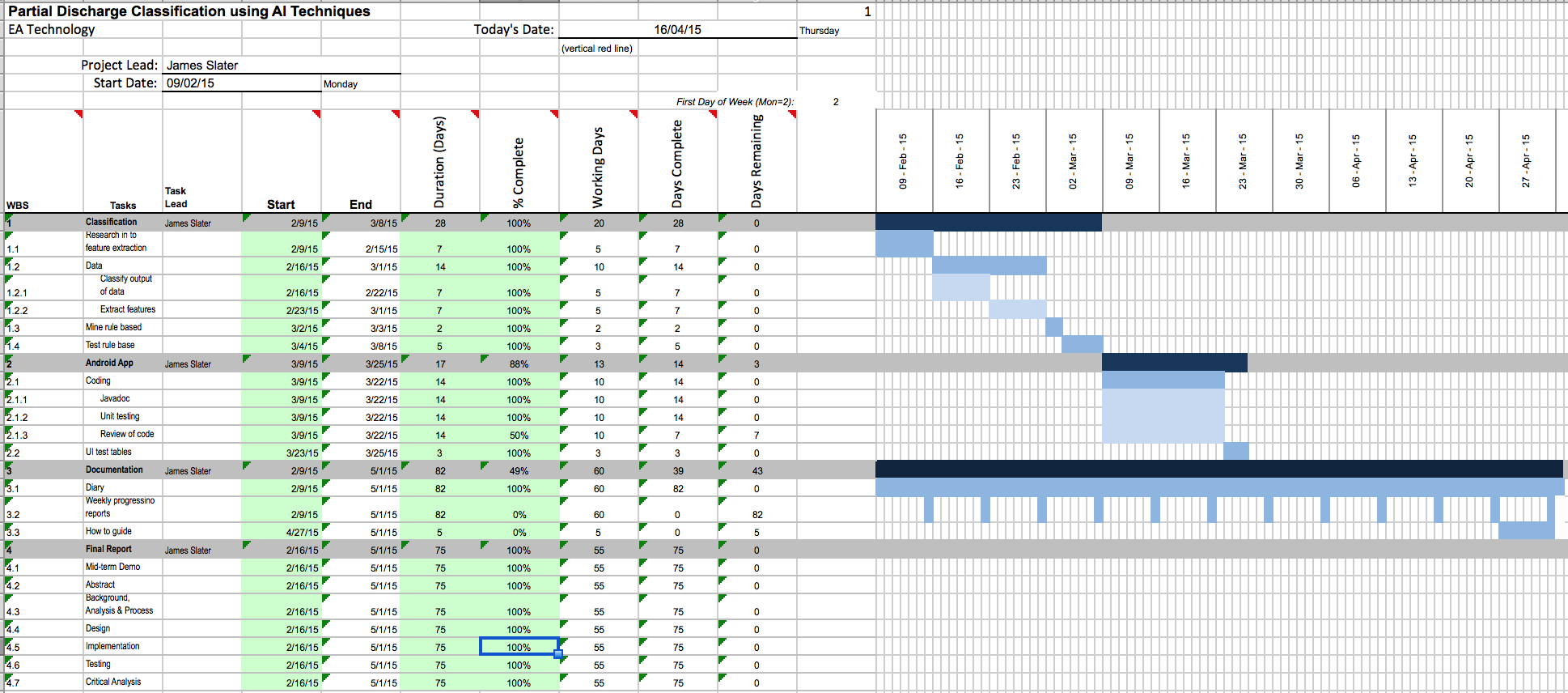
For the data, I will need to decide which features are important and decide how they will impact on the required functionality of the App. As stated by Sahoo and Salama, “The first step in any recognition process is to consider the problem of what discriminatory features to select and how to extract these features from the patterns.”[4] I will use this as my starting point. In order to do this, I will need to process the data into a more relevant and more manageable form, I will need to do some additional research in order to fully understand how to accomplish this in the most straightforward manner.

In order to design a compact classifier it will be best to use data mining tools (e.g. WEKA) to extract rules based on the training data. This rule base could then hard-coded in the App and used to classify new data.

### Android App

* The Android app will work on Android 4.4+ upwards
* It will be able to record audio given
* Javadoc *“Javadoc is a tool for generating API documentation in HTML format from doc comments in source code”.*
* Working executable *APK files are a type of archive file, specifically in zip format packages based on the JAR file format, with .apk as the filename extension.*
* Testing *“The Android testing framework, an integral part of the development environment, provides an architecture and powerful tools that help you test every aspect of your application at every level from unit to framework.”*
* Test Tables for UI While I can use some testing already built into the Android API and testing tools there will be some aspects of the Application that cannot be tested and will need a table.

## Gantt Chart



## UML Diagrams

### Class Diagrams

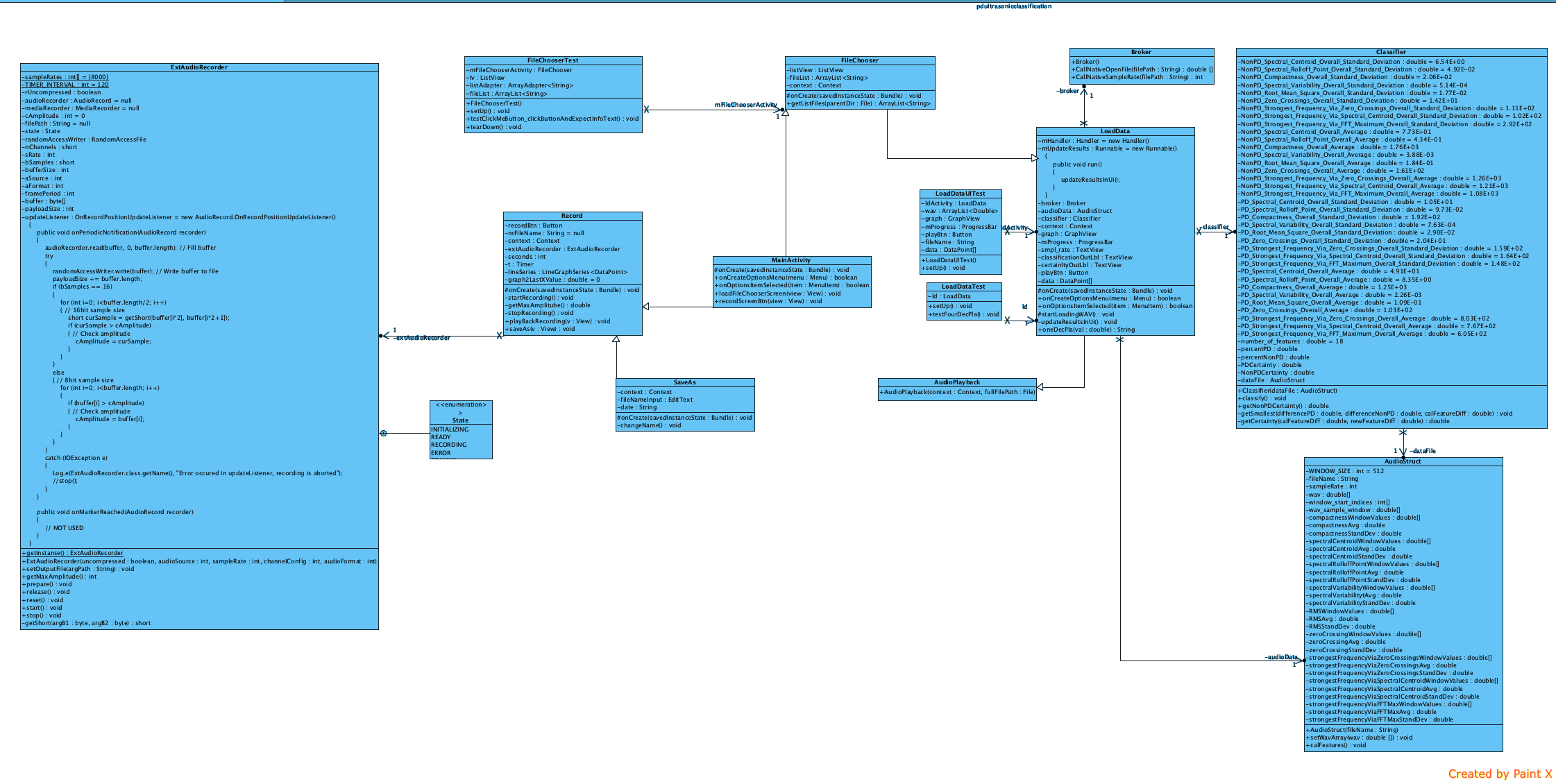
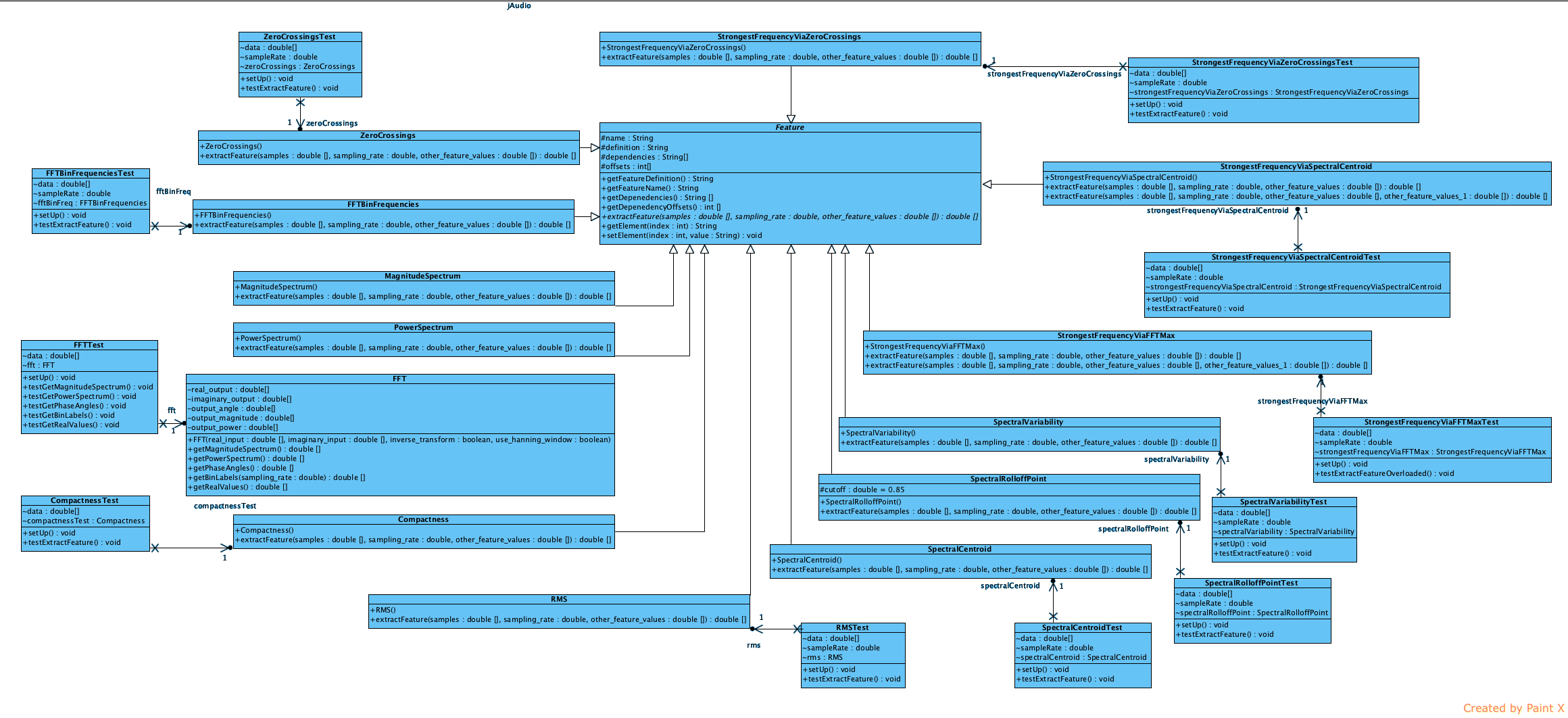


Figure - Class diagram of the Android application

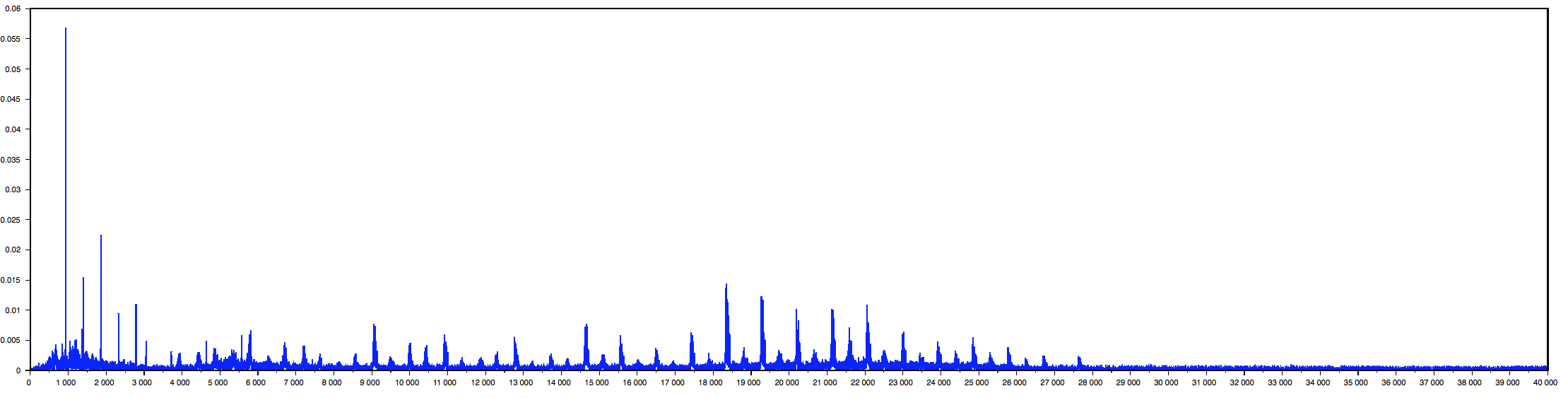


## UML Creator Visual-Paradigm

### [http://www.visual-paradigm.com/](http://www.visual-paradigm.com/sdeij.php)

## Feature Spike work

### NON Partial Discharge

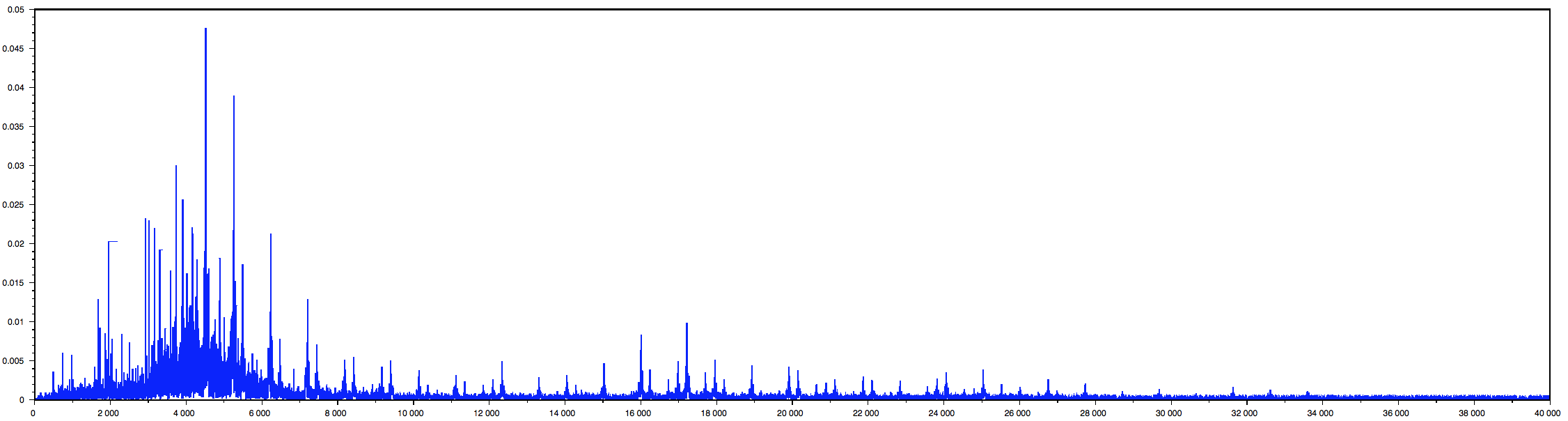
**Ref\_05\_WSP\_Util\_swbd\_feeder 1**

Notes - Very regular, probably not PD

Skewness = 10.929229

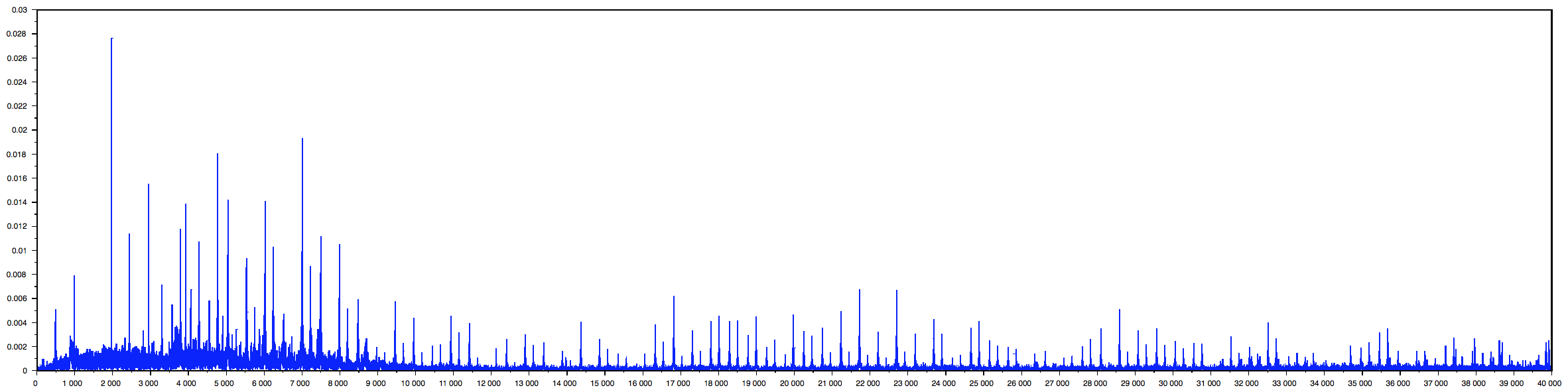
Kurtosis = 451.34994

**Ref\_01\_Sub\_1\_TR\_61\_CD\_1**



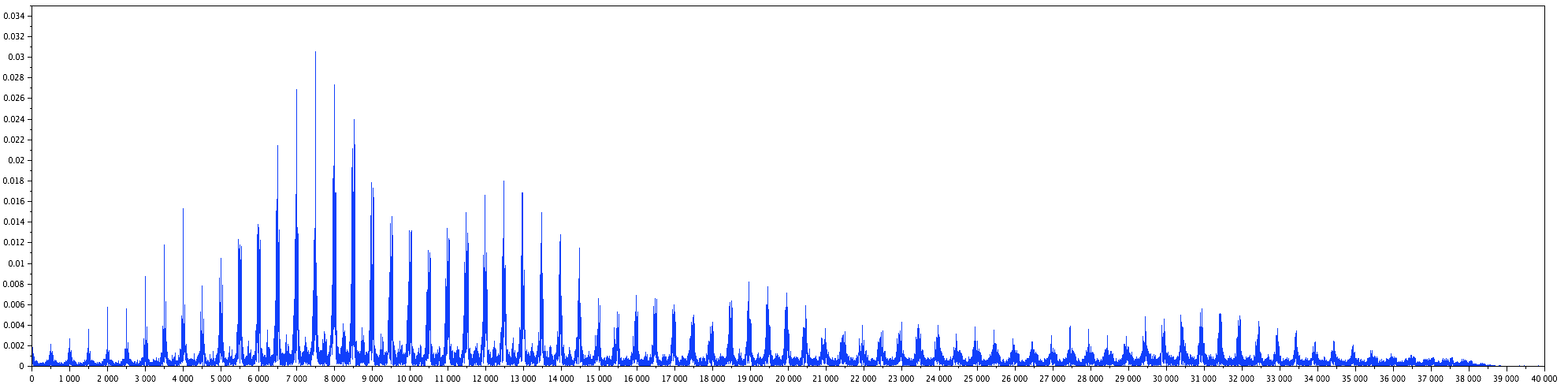
Notes - Very repetitive.

**Ref\_07 Sub 9 42P-371**



Notes - Too regular

**Ref\_11 Central \_ Tx No. Tx01 CHP Building**



Notes - Too regular

Skewness = 5.0396758

Kurtosis = 39.860225

**Conclusion of information**

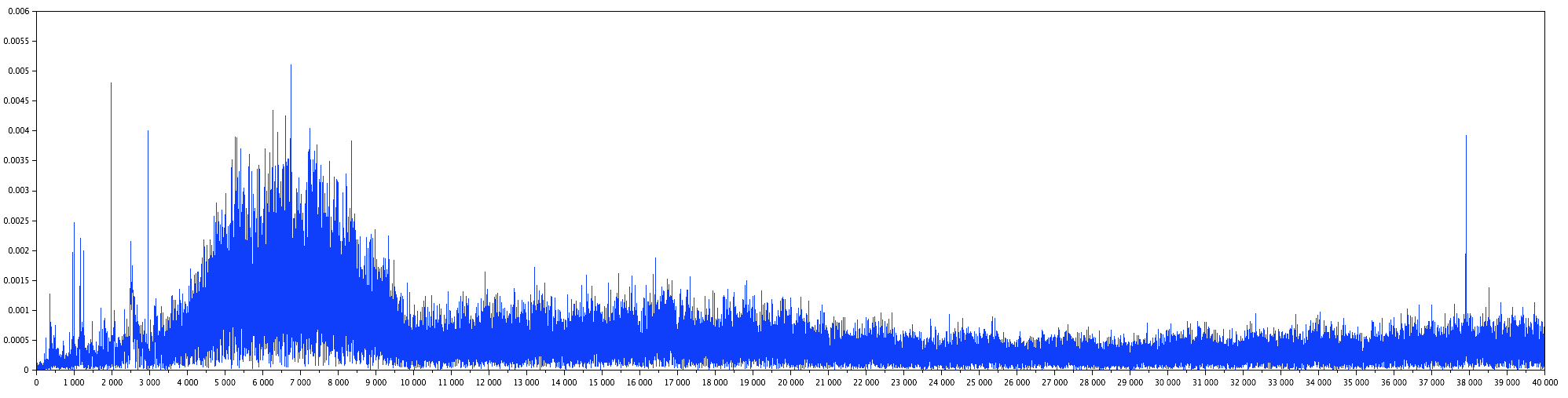
The plots are of the wav form after FFT and only showing the absolute values.

Find a way to find small peaks and if they are regular and full defiantly not PD.

As this is signs of Transformer vibrations/noise

### Partial Discharge

**Ref\_09\_sub9rec2**

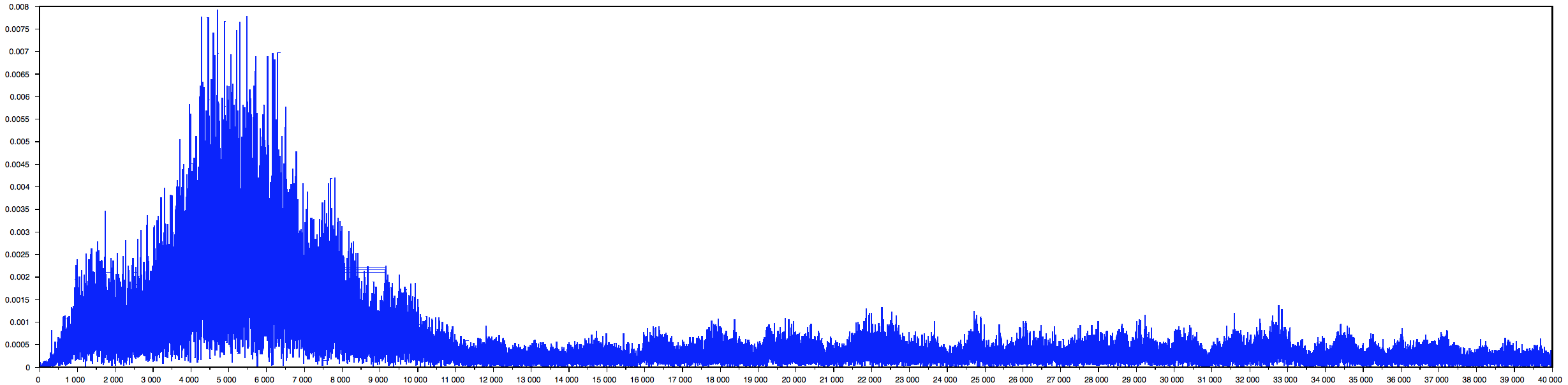


Notes - the spikiness of the noise, and it's dropout

Skewness = 2.7725939

Kurtosis = 13.791916

**Ref\_10 dss k (2)**

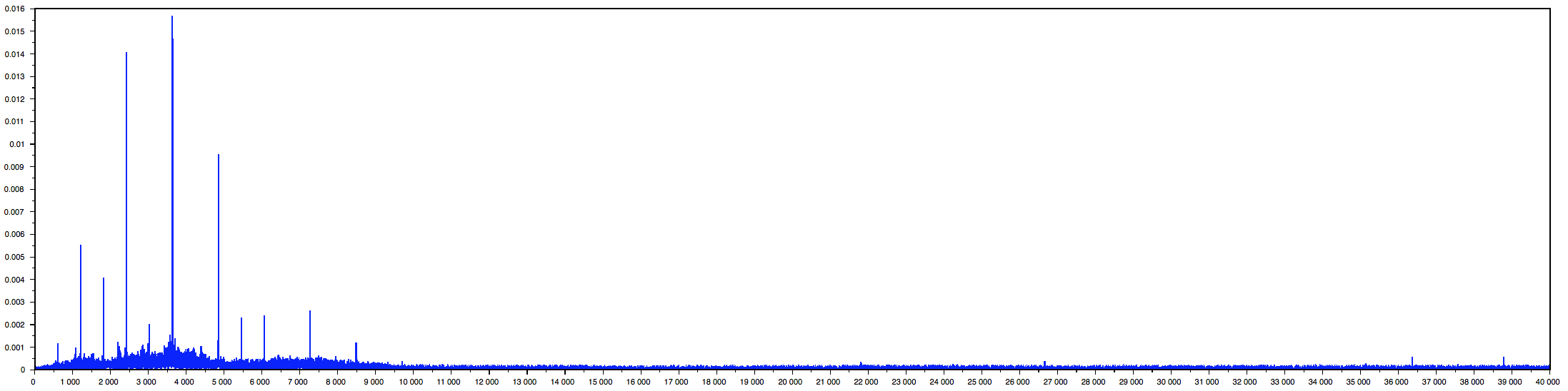


Notes - Characteristic crackle, wind noise ambient adds complication for analysis

Skewness = 3.3425825

Kurtosis = 16.76697

**Ref\_16 Cherokee lhs aux**

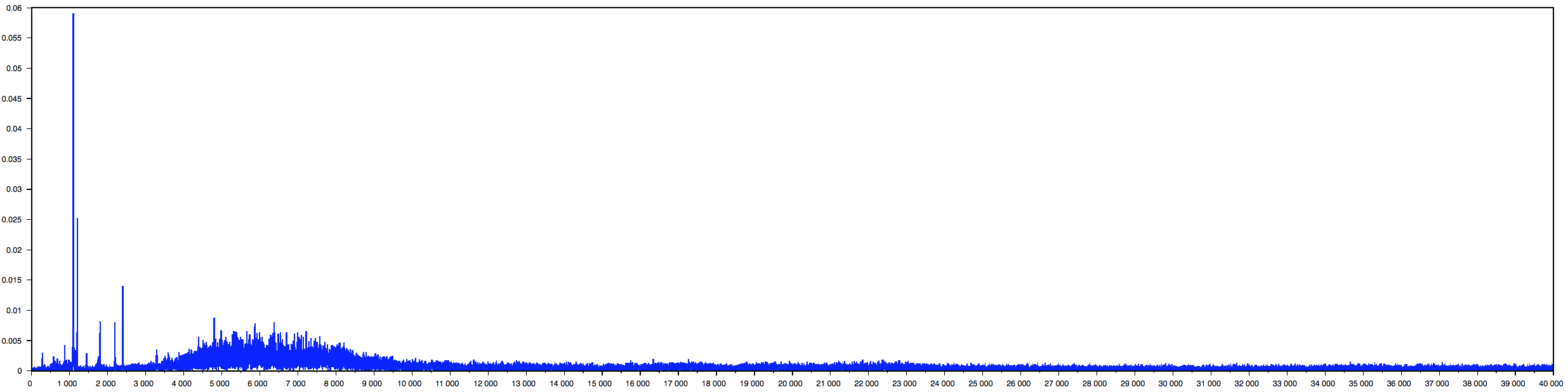


Notes - Crackling noise in background - Possibly PD

Skewness = 38.799082

Kurtosis = 2612.5601

**Ref\_17 emerson**



Notes - Something like across a too small air gap. Vicious sawing noise

Skewness = 16.444505

Kurtosis = 810.7896

**Conclusion of information**

All seem very one sided, still a lot of noise in some of them

Skewness could be a very good factor to determine against vibration noise

Many classical statistical tests and intervals depend on normality assumptions. Significant Skewness and kurtosis clearly indicate that data are not normal.

## jAudio Features Analysis

## Testing Tables

Setup:

* Have an android device running Android 4.4+
* Have the app installed on the android device.
* Load over the test audio file into the apps folder on the internal storage at:
  + “/storage/emulated/0/Android/data/com.cs294.jas38.pdultrasonicclassification/files”

### Load the application.

Setup:

* Find the Application with the Apps section.

|  |  |  |
| --- | --- | --- |
| **Test** | **Successful** | **Comments/ Outcome** |
| Is the Apps name “PD Ultrasonic Classification” | Yes |  |
| Once clicked on the icon, does the app load. | Yes | The main menu screen is loaded. |
| Is the Icon the EA Technology Android logo? | No | It is the default Android icon |

### Main Menu Screen

Setup:

* Orientation - portrait

|  |  |  |
| --- | --- | --- |
| **Test** | **Successful** | **Comments/ Outcome** |
| Is the title “Main Menu” | Yes |  |
| “File Chooser” button is present | Yes | Big clear and green |
| “Record” button is present | Yes | Big clear and red |

Setup:

* Orientation - landscape

|  |  |  |
| --- | --- | --- |
| **Test** | **Successful** | **Comments/ Outcome** |
| Is the title “Main Menu” | Yes |  |
| “File Chooser” button is present | Yes | Big clear and green |
| “Record” button is present | Yes | Big clear and red |

|  |  |  |
| --- | --- | --- |
| **Test** | **Successful** | **Comments/ Outcome** |
| “File Chooser” button is clicked | Yes | A new screen is loaded. A list of files appears. |
| “Record” button is clicked | Yes | The “Record” screen in loaded |
| Click the android back button. | Yes | It takes the screen back to the devices home screen. |

### Record screen

Setup:

* Click on the “Record” button from the home screen.
* Orientation – portrait

|  |  |  |
| --- | --- | --- |
| **Test** | **Successful** | **Comments/ Outcome** |
| The title says “Record” | Yes |  |
| Has label “Count down: ” | Yes |  |
| Has number of 10 next to it | Yes |  |
| Under the label and count is a graph with a title of “Volume Input” | Yes | Title is present however the graph does not show until record starts. |
| A playback button is present | Yes |  |
| A Save as… button is present | Yes |  |
| A record button is present | Yes |  |

Setup:

* Orientation – landscape

|  |  |  |
| --- | --- | --- |
| **Test** | **Successful** | **Comments/ Outcome** |
| The title says “Record” | Yes |  |
| Has label “Count down: ” | Yes |  |
| Has number of 10 next to it | Yes |  |
| Under the label and count is a graph with a title of “Volume Input” | Yes | Title is present however the graph does not show until record starts. |
| A playback button is present | Yes |  |
| A Save as… button is present | Yes |  |
| A record button is present | Yes |  |

|  |  |  |
| --- | --- | --- |
| **Test** | **Successful** | **Comments/ Outcome** |
| Click the button record | Yes | The count down value counts down in 0.1 increments.  The graph shows the sound volume input over time.  After the count hits zero and the graph stops moving along and the count resets. |
| Click on Play back button | Yes | The Audio is played. |
| Click on the Save as button | Yes | Loads the Save as screen. |

### Save As Screens

Setup:

* Make sure there is a recorded audio tmp file.

|  |  |  |
| --- | --- | --- |
| **Test** | **Successful** | **Comments/ Outcome** |
| Reference label is present | Yes |  |
| Edit text input is present | Yes |  |
| Date label is present | Yes |  |
| The current date is shown in the format YYYY\_MM\_DD | Yes |  |
| Save button is present | Yes |  |
| Enter txt in the edit text aspect, test max length is 15 characters | Yes | Limits text to 15 characters |
| Only [a-z A-Z 0-9 \_-] characters can be used | Yes | Can press other characters however nothing is inputted |
| Enter “Testing” into the edit text and click the save button | Yes | Takes you back to the Record screen and a pop up saying saved file “Testing”. |
| Check the file store and see if the file has been saved Testing\_YYYY\_MM\_DD.wav | Yes |  |

### File Chooser Screen

Setup:

* Load the app and click the File Chooser button
* Have the test.wav file loaded within the apps folder or record some audio within the record screen.

|  |  |  |
| --- | --- | --- |
| **Test** | **Successful** | **Comments/ Outcome** |
| Can see a list of files on screen | Yes |  |
| Test.wav file is an option | Yes |  |
| If you change orientation all the same files exist | Yes |  |
| There is a scroll bar to the right, while this may not move anything if the user has many files it will be needed | Yes | It does nothing with only one .wav file in the folder. |
| Each file shown is clickable | Yes | Clicked on test.wav file and it loads View data screen. |
| Click the Android back button | Yes | Takes me back to the main menu. |
| After clicking the test.wav file and the new screen loads click the Android back button. | Yes | Takes me back to the list of files. |

### View Data screen

Setup:

* From the main menu click File chooser, click the test.wav file
* Make sure the volume is turned up to an audible level
* Have in portrait

|  |  |  |
| --- | --- | --- |
| **Test** | **Successful** | **Comments/ Outcome** |
| Title is View Data: test.wav | Yes | The title is the file that was clicked from the previous screen |
| There is a plot present of the audio file | Yes |  |
| The plot y-axis = 1 to -1 | Yes |  |
| The plots x-axis is from 0 to 250 | Yes |  |
| Move your thumb from right to left along the plot. Does the X-axis move along? | Yes | Very responsive. |
| Pinch together over the plot and the X scale will change | Yes | It went from 0-250 to 0-2000. Though the load time of this was very high and after the moving of the plot was very delayed. |
| Under the plot there is a label called Sample rate | Yes |  |
| To the right of Sample rate is the number that is the sample rate it should be one of these values: {44100, 22050, 11025, 8000}; | Yes | The value was 8000 |
| A label situated under the sample rate called “Classification”, the label will be in larger text | Yes |  |
| The value next to the Classification label will be the classification of the Audio file, either PD or Non-PD. For the ‘test.wav file’ the classification will be PD | Yes |  |
| Each Classification comes with a certainty percentage, the one present for the test.wav file will be 95% | Yes |  |
| Button called ‘Play Audio File’ is present? | Yes |  |
| Click on the button ‘Play Audio File’, does the file play? | Yes |  |
| When tilting to a landscape orientation, all labels, buttons and plot still present with the same values? | Yes |  |

## Final Screen Shots

## Graph View

The project is a library for Android to programmatically create  
flexible diagrams. Version 4.0.0 is the latest instalment created by Jonas Gehring, is open source under the GNU general public license, version 2, June 1991. The GNU license see [5]. Full website for more information see [4]. I used the .jar file and incorporated into the final Android app.

## jAudio 2.0

jAudio is a software package for extracting features from audio files as well as for iteratively developing and sharing new features. These extracted features can then be used in many areas of music information retrieval (MIR) research, often via processing with machine learning frameworks. Authors Daniel McEnnis and Cory McKay, I used jAudio 2.0 to do some feature extraction from the data provided. TODO – add more information on how and where I used this software and on licensing

## Libsndfile

Libsndfile is a C library for reading and writing files containing sampled sound (such as MS Windows WAV and the Apple/SGI AIFF format) through one standard library interface. It is released in source code format under the [Gnu Lesser General Public License](http://www.gnu.org/copyleft/lesser.html). I used this library to be able to read wav audio files on the android application; all the code was copied in to the jni folder.

Libsndfile has the following main features:

* Ability to read and write a large number of file formats.
* A simple, elegant and easy to use Applications Programming Interface.
* Usable on Unix, Win32, MacOS and others.
* On the fly format conversion, including endian-ness swapping, type conversion and bit width scaling.
* Optional normalisation when reading floating-point data from files containing integer data.
* Ability to open files in read/write mode.
* The ability to write the file header without closing the file (only on files open for write or read/write).
* Ability to query the library about all supported formats and retrieve text strings describing each format.

The code used was from <https://github.com/michaelwu/libsndfile>

## Recording WAV format audio in Java

<http://i-liger.com/article/android-wav-audio-recording>

## YAAFE

*YAAFE, an Easy to Use and Efficient Audio Feature Extraction Software*, B.Mathieu, S.Essid, T.Fillon, J.Prado, G.Richard, proceedings of the 11th ISMIR conference, Utrecht, Netherlands, 2010.

Available features:

- AmplitudeModulation

- AutoCorrelation

- ComplexDomainOnsetDetection

- Energy

- Envelope

- EnvelopeShapeStatistics

- Frames

- LPC

- LSF

- Loudness

- MFCC

- MagnitudeSpectrum

- OBSI

- OBSIR

- PerceptualSharpness

- PerceptualSpread

- SpectralCrestFactorPerBand

- SpectralDecrease

- SpectralFlatness

- SpectralFlatnessPerBand

- SpectralFlux

- SpectralRolloff

- SpectralShapeStatistics

- SpectralSlope

- SpectralVariation

- TemporalShapeStatistics

- ZCR

Available feature transforms:

- AutoCorrelationPeaksIntegrator

- Cepstrum

- Derivate

- HistogramIntegrator

- SlopeIntegrator

- StatisticalIntegrator

Available Output formats:

- csv

- h5

## Stack Overflow

1. Searching a folder using java for certain files. See - <http://stackoverflow.com/questions/9530921/list-all-the-files-from-all-the-folder-in-a-single-list>
2. Passing String from Java to C code in the Android environment. See - <http://stackoverflow.com/questions/11558899/passing-a-string-to-c-code-in-android-ndk>
3. Solving the Testing error. See - <http://stackoverflow.com/questions/22364433/activityunittestcase-and-startactivity-with-actionbaractivity>

## Pseudo code

### Classification Algorithm

## 

# Annotated Bibliography

1. The PD Academy is an online resource for sharing knowledge on Partial Discharge. Link: <http://www.partial-discharge-academy.com/what-is-partial-discharge>
2. Sinan Si Alhir, UML in a nutshell – A quick reference, First Edition. 1998 O’Reilly & Associates. ISBN 1-56592-448-7   
     
   I used this book for a reference for the UML diagrams used is this report. I found the overall organization of the Diagrams and which ones to put where most helpful.
3. Apache Software Foundation (2004) “Apache License, Version 2.0” (Online) Available at: <http://www.apache.org/licenses/LICENSE-2.0> Accessed: 14th March 2014.
4. Graph View - open source graph plotting library for Android. Version 4.0.0 – Available at: <http://www.android-graphview.org/> Accessed: 28th February 2015
5. GNU General Public License, link - <http://www.gnu.org/copyleft/gpl.html> Accessed: 28th February 2015.
6. Adding native support to the Android app. Link -- <http://www.javaworld.com/article/2077513/learn-java/java-tip-17--integrating-java-with-c--.html>

This is a tutorial I used to be able to understand how to set up the environment and utilise the C/C++ code. It gives an example and very clear diagrams on how issues can arise when using native code in the Android environment.

1. YouTube tutorial Available at <https://www.youtube.com/watch?v=kFtxo7rr2HQ>

Building NDK apps with Android studio is the tutorial title. I used this to help understand how to implement support of native code into the android app. While the tutorial is for an older version of Android it is still relevant.

1. Android NDK Available at <https://developer.android.com/tools/sdk/ndk/index.html>

Here is the NDK to build and compile the C/C++ code. It has instructions on how to install and implement it into the Android SDK.

1. Java Sound Resource Available at <http://www.jsresources.org/>

I used this website for further information into how to record and read wav formatted files. I found this very insightful and in depth regarding how java can handle audio data.

1. Audio file C library used in my Android app <http://www.mega-nerd.com/libsndfile/>

The libsndfile C library is used to read WAV format audio files. I used this library to read the audio file in and process it to a double array. I also got the sample rate from the file as it is needed for the feature extraction. This has been released under the GNU license.

1. H. William, P. E. Bartley, “Analysis of Transformer Failures”, International Association of Engineering Insurers 36th Annual Conference – Stockholm, 2003. Link -- <https://www.hsb.com/TheLocomotive/AnAnalysisOfInternationalTransformerFailuresPart1.aspx>

I used part one and part 2 for information on how transformers fail. It has all the stats and was very helpful in understanding how transformers fail and knowing that insulation failure

1. *Service handbook for transformers-3rd edition*, published by ABB.

I only read chapter 9.3 as this section was on monitoring. This game me an extensive list on how high and low level monitoring of electrical assests.

1. <http://www.fftw.org/index.html>

FFTW is a C subroutine library for computing the discrete Fourier transform (DFT) in one or more dimensions, of arbitrary input size, and of both real and complex data (as well as of even/odd data, i.e. the discrete cosine/sine transforms or DCT/DST). FFTW, which is [free software](http://www.fftw.org/faq/section1.html#isfftwfree) under the GNU General Public License.

1. Cigre - Recognition of Discharges, Electra No 11 page 61-98.

This paper goes through the types of partial discharge that can occur on high voltage assets. It shows the observed discharge and its characteristics of any unwanted disturbances, and some means of diagnosing them. It is not a complete list and all results are shown on an oscilloscope.

1. Android developer website, <http://developer.android.com/index.html>. Accessed through out the project.

The Android developer website has a lot of information regarding the design, the development and the finishing touches on how to create an Android app.

1. PD Electrical sensors, <http://www.fargocontrols.com/sensors/>

I used this website for a further insight into how both capacitive and inductive sensors work.

1. Partial Discharge Detection and Localization in High Voltage Transformers Using an Optical Acoustic Sensor by Alison K. Lazarevich.

This paper has a better understanding of the ways PD can be detected. I used this for an in-depth background into chemical, electrical and acoustic detection of Partial Discharge.

1. Partial Discharge Signatures of Defects in Insulation Systems Consisting of Oil and Oil-impregnated Paper by Mohamad Ghaffarian Niasar, 2012

This paper has an in-depth knowledge into partial discharge and its many forms. I have used this paper to refer to the definitions of the many types of PD and detection methods.

1. Partial Discharge Pattern Recognition of Transformer Based on Electric Signal and Ultrasonic Comprehensive Analysis, by Shutao Zhao, Baoshu Li and Yong Wang.

Abstract - According to the characteristics and disadvantages of PD electrical analysis method, an electro-acoustic and ultrasound combination partial discharge (PD) pattern recognition method is proposed.

I used this paper to see what statistical features can be found from the waveforms.

1. The hardware specification for the Samsung S4 - <http://www.gsmarena.com/samsung_i9500_galaxy_s4-5125.php>
2. History of Android, <http://en.wikipedia.org/wiki/Android_version_history#Android_1.5_Cupcake_.28API_level_3.29>

Accessed: 21th February 2015. I used this for the background on Android.

1. Android 4.4 compatibility pdf file, <https://static.googleusercontent.com/media/source.android.com/en//compatibility/4.4/android-4.4-cdd.pdf>

This document has all the information on the needed hardware and specification of any mobile device running Android 4.4. I only used the hardware section 7, for the storage information on the minimum requirements.

1. Trends in Partial Discharge Pattern Classification: A Survey, by N.C. Sahoo, M.M.A. Salama and R. Bartnikas

Abstract - Partial discharge (PD) detection, measurement and classification constitute an important tool for quality assessment of insulation systems utilized in HV power apparatus and cables.

This paper describes some of the tools used to extract features and compare how different AI approaches perform.

1. Android overall Architecture picture is from <http://theworkshop.screeneros.com/android-source-code-under-miscroscope/>

Access on 6th March, it was created by [Nanik Tolaram](http://theworkshop.screeneros.com/author/nanik-tolaram/), on 02 October 2014.