Development of a low-cost vibration protection device for industrial gearboxes

by

Rudolph Christoffel Kroch

Dissertation submitted in partial fulfilment of the requirements for the degree Master of Engineering in the Faculty of Engineering, Built Environment and Information Technology, the University of Pretoria

December 2014

Abstract

A market for a low-cost vibration protection device in the rotating machine industry has been identified that satisfies the needs of small firms unable to afford and sustain a condition monitoring operation.

In this project, a system is developed that satisfies the need for a low-cost, conservative, configurable and intuitive device that can perform vibration measurements on a range of gearboxes and make an inference as to the level of vibrations coming from the bearings on the shafts.

The inference made by the device, derived from the frequency content of the measured signal, may be used by the operator of the gearbox to make a judgment of whether to have the gearbox investigated by a competent authority. In order to assist this investigation, a vibration history of the device is stored, both in time and frequency domain formats, as well as a full history of the relevant diagnostic information.

To reach this point of maturity, the project evolved through three different hardware configurations. The various iterations were tested within the scope for which they were designed and the lessons learned after each test was incorporated into the next iteration. The final iteration incorporated all the refinements of the system up to that point as well as the anticipated scope of further development into the commercial realm.

To verify the inference credibility of the device, the results of the final specification of the device was evaluated against data obtained from the condition monitoring department of SASOL in Secunda. The results were analysed on two accounts. Firstly the signal reproduction accuracy was evaluated, which established how accurately the signal was digitized and how the processing algorithms performed. Secondly, the inference accuracy was gauged against the practices of SASOL. On both accounts, the final device performed satisfactorily.

The end result of this project is considered a ‘near-commercial ready’ prototype with all the hardware on-board for user interaction, signal processing, 3rd party viewing of the data and future expandability.

Acknowledgements

I would like to thank the following people in the industry for their help in completing this project and research:

- Prof. Stephan Heyns from the University of Pretoria
- Herman Booysen from University of Pretoria
- Johan Pretorius from SASOL
- Hannes Smit from Deman Manufacturers
- Willem Sullivan from our Industrial Partner
- Dumisani Mhlope from Periseo
- Kevin Schlorke from Periseo

In addition, I would like to extend my sincerest appreciations to the following people for their continued love and support in my studies:

- My Creator
- My wife, Karin Kroch
- My Parents, Danie and Magriet Kroch

Glossary

Acronyms

ADM	:	Advanced Development Model
BPFI	:	Ball Pass Frequency of the Outer race
BPFO	:	Ball Pass Frequency of the Inner race
BSF	:	Ball Spin Frequency
CBM	:	Condition Based Maintenance
DFT	:	Discrete Fourier Transform
DIF	:	Decimation in Frequency
DIT	:	Decimation in Time
DMA	:	Direct Memory Access
DSC	:	Digital Signal Controller
DSP	:	Digital Signal Processor
FIR	:	Finite Impulse Response (filter)
FTF	:	Fundamental Train Frequency
GMF	:	Gear Mesh Frequency
I/O	:	Input and Output
IC	:	Integrated Circuit
ICD	:	In Circuit Debugger
IF	:	Instantaneous Frequency
IS	:	Instantaneous Speed
MCU	:	Micro Controller Unit
NI	:	National Instruments (Brand)
PCB	:	Printed Circuit Board
PGA	:	Programmable Gain Amplifier
PIC	:	Programmable Interface Controller
PLL	:	Phase-Locked Loop
PSD	:	Power Spectral Density
RPM	:	Revolutions Per Minute
SD	:	Secure Digital (Card)
SPI	:	Serial Peripheral Interface
SRAM	:	Static Random Access Memory
UI	:	User Interface
ASCII	:	American Standard Code for Information Interchange
XDM	:	Experimental Development Model

Symbols

B_d	:	Ball or roller diameter of anti-friction bearing
F_s	:	Sampling Frequency
N_b	:	Number of balls or rollers in anti-friction bearing
P_d	:	Bearing pitch diameter
T_s	:	Sampling Time
θ	:	Anti-friction bearing contact angle

Contents

List of Figures

List of Tables

1. Introduction

1.1 Background

One of the most common classes of machines encountered in the industry today is rotating machines (Heng, et al., 2009). This places the maintenance of rotating machinery at the forefront of plant expenses and indeed, plants in the United States spent more than $600 billion in 1981 to maintain their operationally critical equipment and that figure had doubled by 2001 (Mobley, 2004).

One can reasonably expect that figure to rise even further, seeing that typical advanced and expensive machinery represents an investment for a company and therefore demands ever more sophisticated maintenance strategies. The reasoning for this is purely economic: A plant operating at maximum efficiency allows one to produce more and therefore sell more. To this end, a properly implemented maintenance plan can significantly reduce maintenance costs by reducing the number of unnecessary maintenance operations.

Against this background, it is clear that 'Breakdown maintenance' (running a machine until it breaks) and time based maintenance (servicing a machine, without prior knowledge of its state of repair) results in a haemorrhage of funds and that more efficient maintenance strategies are required to assist the situation that so many companies find themselves in (Jardine, et al., 2006).

Many large companies have indeed recognised this need and are trying to address the issue. It is not uncommon for a contemporary medium to large scale company to have a number of professionals working exclusively on monitoring the condition of the equipment operating on the plant. This does, of course, require a significant investment for the start-up company and many smaller firms simply cannot afford to make the leap to a fully online condition monitoring strategy. These firms are inevitably stuck in the time based maintenance strategy, or worse, the breakdown maintenance strategy due to the large capital investment required.

Even though the current reality remains that an online condition monitoring scheme is a heavy investment to make for a start-up or small company, an argument can be made that a niche exists for a low-cost automated protection system.

A clear definition of such a system can be formulated as follows (Ma & Jiang, 2011):

"Contemporary plant information systems collect and archive plant-wide measurement data. Real-time and historical data can be analysed for plant performance monitoring, an abnormal event can be swiftly delivered to pertinent plant personnel for subsequent actions"

It is important to note the distinction between a protection system and a condition monitoring system. A condition monitoring system is better thought of as a strategy which exhibits the following three elements (Heng, et al., 2009):

- The measurement and storage of data
- Processing of the measured data (i.e. signal conditioning and feature extraction)
- Making a *technically-oriented* diagnosis based on the processed data

These elements may well be accomplished by different means removed from each other such as data loggers, computer software and engineers.

This differs from a protection system in both philosophy and execution. A protection system is considered, in this document, to mean a self-contained system capable of the following:

- Collection and storage of data
- Processing of the results and archiving
- Making a recommendation on an *administratively-oriented* course of action based on the processed data (such as arranging for specialists to conduct a detailed investigation, assisted by the recorded data).

What sets it apart is that it gives a small company the capability to obtain an inexpensive system that they can use to evaluate their machinery and provide a warning that specialists should be consulted if there is a likelihood that a problem might exist. Importantly however, it does not deliver a diagnosis, but a course of action – i.e. contact specialists.

By using the data collected by the ADC portion of the device and using rudimentary signal processing techniques, the device should be capable of making a conservative inference regarding the level of vibrations experienced by the gearbox – or specifically, the bearings in the gearbox.

Typically, the inference may take the following forms, *in order of increasing likelihood* of damage:

- Vibration normal
- Vibration caution
- Vibration warning

When the system does warn the user that a problem might exist, external help (such as the OEM or other specialists) can be contacted. The third party can then review the archived data and archived analysis results to spot a trend and make a diagnosis and prognosis – thereby completing the condition monitoring process of Data collection – Processing – Diagnosis, without the need for a dedicated department.

Bearing the objective of a vibration protection system in mind, this project aimed to deliver a low cost, handheld, vibration protection system. A device of this description would require:

- A means to record vibration data
- A measure of signal processing capability
- Non-volatile storage
- Portability
- A user interface (UI)
- A means to interact with the UI

Technically, this implies:

- An ADC microchip
- A medium powered microprocessor, preferably a DSP
- An SD card

- A rechargeable battery with recharging circuit
- An LCD screen
- Buttons

From this intuitive specification, a device capable of successfully performing the vibration protection role, as outlined in this section, was developed. Due to the low-cost nature of the protection system, an advanced CPU with a complex circuit layout was neither possible nor required. To this end, a basic DSP type processor was used to process the data and perform the rudimentary algorithms. However, to achieve practical levels of performance (in terms of time and results), the algorithms were optimised as much as possible.

The system evolved through three prototypes gaining in complexity and maturity. These prototypes were tested in the laboratory and the field. The results of each test then influenced the next prototype, with the final prototype being near production ready.

This document details the development lifecycle of the project. It starts by exploring the present gearbox condition monitoring environment as background to factors taken into account during this project. A study is then undertaken to obtain suitable solutions to address these challenges, both in hardware and software.

1.2 Vibration monitoring fundamentals

1.2.1 General gearbox condition monitoring

In gearboxes, both gearing faults (local and distributed) and rolling element bearing faults (usually local) are usually encountered. When these faults are investigated, it is usually the case that fluctuating external load is not considered (Bartelmus & Zimroz, 2009) (Stander & Heyns, 2002). Additionally, one would normally assume that only one anomaly exists in a gearbox at a time and that the change in the condition of the gearbox is as a result of the development of the anomaly.

There are several reasons for anomalous gearbox vibrations which can be classified into four different categories (Bartelmus, 2008), namely: Design, Production technology, Operation and Change of condition. A chart detailing the different mechanisms for gearbox vibrations is given in Figure 1.

From a vibration condition monitoring point of view, the following simplistic model for a gearbox can be used to get an understanding for the environment in which the gearbox will typically operate (Bartelmus, 2008):

- System of elements
  - Prime mover (often electric motor)
  - Gears
  - Bearings
  - Shafts
  - Coupling
  - Driven machinery

- Factors influencing vibration
    - Design
    - Production
    - Operation
    - Condition change

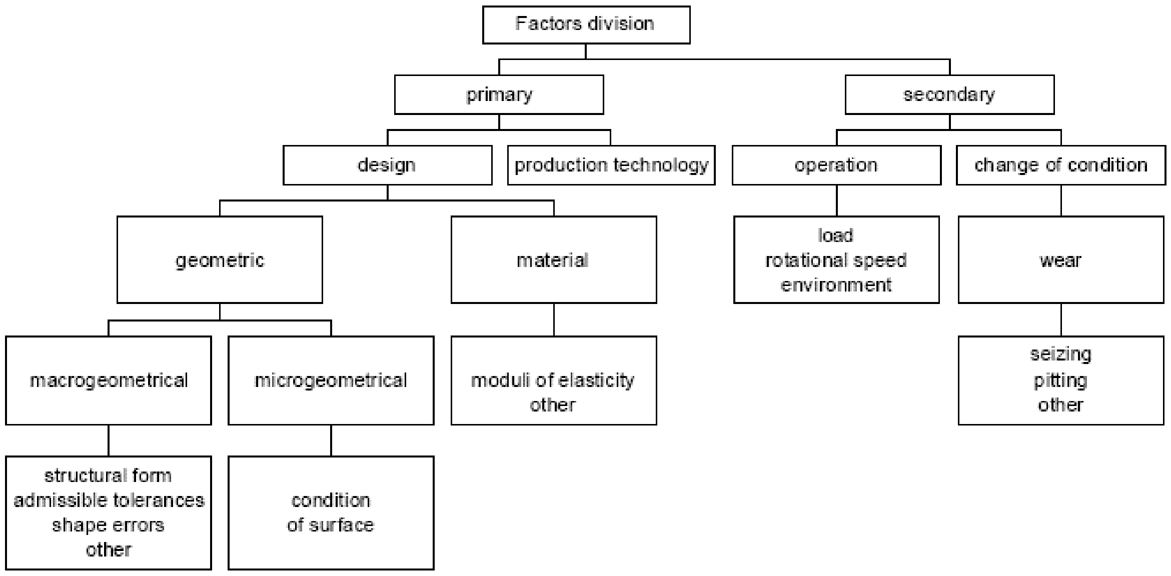


Figure 1: Bearings are some of the most common component failures in rotating machinery (Bartelmus W., 2008)

1.2.2 Speed measurements

Determining of the instantaneous speed (IS) is crucial in the condition monitoring environment and is currently one of the most important tasks in many industrial applications (Combet & Zimroz, 2009) (Combet & Gelman, 2007) (Radoslaw, et al., 2011). In automatic monitoring systems, this task is even more important as the automated algorithm will not work properly if the IS is computed incorrectly. That is because the IS is used to find the characteristic frequencies of components such as shafts, bearings, gears, etc.

In practice, however, it is often very difficult to obtain this information, usually requiring additional hardware and associated wiring. Compounding the problem, tachometers or shaft encoders may not be a viable solution due to the shafts (either input or output) being inaccessible. When this is the case, obtaining the IS can often be obtained indirectly for vibration measurements on the gearbox casing.

When vibration signals are used to estimate the IS, the fundamental frequency can often be tracked and thus the issue becomes one of tracking the IF (Instantaneous Frequency). However, in the case of a change in speed or load during the measuring time, smearing can occur, thus reducing the accuracy.

In the case of steady state operation (which is the focus of this project), the method of gauging the IS from the IF is considered to be accurate enough for the purposes of this project.

The reader is referred to Section 5.2.3 where this problem is addressed in the algorithm.

1.2.3 Bearing faults

By their very nature, bearings are one of the most common components in rotating machinery (Kiral & Karagülle, 2003) and therefore, one of the most often replaced. Additionally, it has a finite lifespan and is often the subject of abuse (McInemy & Dai, 2003).

Therefore, with the deliverable of this project being a prototype, it was decided to focus on bearing damage first. In addition, due to the subtle nature of bearing damage, it was considered to be the most challenging.

The type of damage typically seen on roller bearings are as follows (Ganeriwala, 2010):

- Ball damage
- Inner race defect
- Outer race defect
- Cage damage

Each of these faults generates a distinct frequency which can be obtained when converting a time signal to the frequency domain. These frequencies are dependent on the geometry of the bearing, the speeds of the inner and/or outer races and, under certain circumstances, the change in load applied to bearing (if a bearing is both radially and axially loaded, the contact angle and thus the frequency – see equation below – will be affected). An example of one such an equation (for outer race defect) is given below and additional fault frequencies can be found in the same reference (Bloch & Geitner, 1999):

$$\mathrm{BPFO} = \frac{\mathrm{N_B}}{2}\left(1 - \frac{\mathrm{B_d}}{\mathrm{P_d}}\cos\theta\right) \times \mathrm{RPM} \qquad \text{[Eq. 1]}$$

Where N_b is the number of balls or rollers, B_d is the ball or roller diameter, P_d is the bearing pitch diameter and Θ is the contact angle. A time and frequency domain representation of some of these faults can be represented as follows:

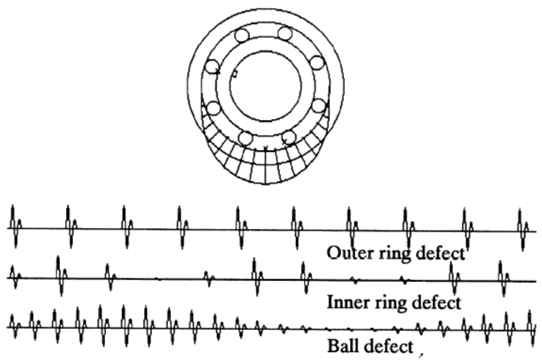


Figure 2: Time domain response of defective bearing, inner race rotating and outer race stationary (Kardushin, 1991)

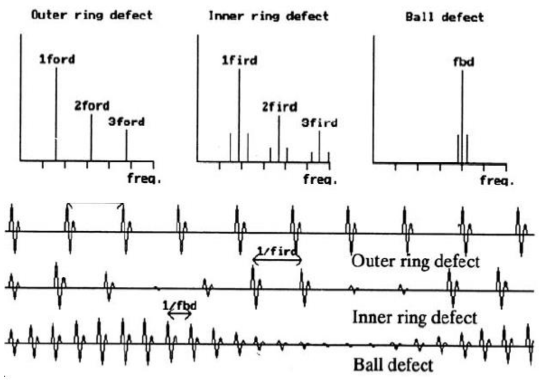


Figure 3: Frequency domain response showing sidebands, inner race rotating and outer race stationary (Kardushin, 1991)

In bearing frequency calculations the following assumptions generally hold (Ganeriwala, 2010):

- Equal diameter balls
- Pure rolling contact
- No slippage between shaft and bearing

In practice, pure rolling contact and no slippage will not always be maintained, however the error introduced in practice should not usually be sufficiently large to be of concern.

1.2.4 Research trends

There are historically several methods how potential bearings faults were detected starting with the most basic technique of the screwdriver to the ear (Bloch & Geitner, 1999). This technique worked remarkably well when applied by a skilled artisan. However, much more sophisticated techniques became available and research is still continuing. A good starting point for a protection system will often be overall vibration measurements such as RMS, CF, etc. (Jardine, et al., 2006). These techniques are however not considered reliable enough for the current application, as their reliability is compromised in the absence of significant impulsiveness, as found under certain circumstances of distributed bearing failures. However, frequency domain techniques tend to be more sensitive and provide an earlier indication of possible. Indeed, many modern techniques are frequency domain based.

In the research domain, several advanced techniques are used to identify bearing damage. The most common techniques are listed below (Ganeriwala, 2010):

- Time waveform analysis
- Frequency spectral analysis
- High Frequency detection
- Stress wave analyser or spike energy
- PeakVue
- Enveloping

Of these, the time and frequency domain techniques are very well established and are used, in one form or another, throughout the industry (Karacay & Nizami, 2009). However, many of these techniques work only within a certain set of circumstances, which can limit their application to more general problems.

A proven technique that can work consistently (albeit sometimes with less accuracy) is the time and frequency domain analysis (Taylor & Kirkland, 2004). Using this technique within a rule based diagnostic system, most machinery problems can be identified. In fact, this technique can be especially useful for rolling element bearing fault detection as the fault frequencies need only be computed (from bearing and operating info) and comparing it to the frequency spectrum (Taylor & Kirkland, 2004). Due to the fact that rolling element bearings in good condition only create a random noise when in operation (Tandon & Choudhury, 1999), the presence of a fault frequency in the frequency spectrum is a strong indicator of bearing damage (Taylor & Kirkland, 2004) (Kiral & Hira, 2003). However, it has been noted that in some instances that damaged bearings exhibit a Gaussian probability distribution (Mathew & Alfredson, 1984).

Some of the very latest techniques being researched and used today include the following (Ganeriwala, 2010):

- Adaptive Noise Cancellation
- Self-Adaptive Noise cancellation
- Spectral Kurtosis
- Discrete random separation
- Cyclostationary signal analysis
- Hilbert-Huang Transform
- Entropy

One must also take cognisance of the High Frequency Resonance Technique (HFRT), which is very popular in bearing diagnostics (Nelwomondo, et al., 2006). This technique is based on the high frequency resonances in the component structure as documented in one of the earliest papers on bearing diagnostics (Balderston, 1969) and has been very successfully applied (Randall & Antoni, 2011).

A common use for the enveloping technique is the identification of cracks in the outer race, inner race and rolling elements of anti-friction bearings (Konstantin-Hansen, 2003) and is excellent for diagnosing cracks and spalls. It can be very effective in conjunction with other signal processing techniques as part of a complete maintenance program.

These techniques represent some of the latest available research and can achieve a high degree of accuracy in discovering and diagnosing damage in bearings.

However, considering the purpose of the project, to develop a low cost vibration protection device, these techniques are not within the current scope as this project does not seek to develop a system capable of delivering accurate diagnoses. Rather a handheld system, likely to be carried by a technician or built into a gearbox, capable of conservative estimates on which to judge whether professional assistance is required. Furthermore, although cognisance is taken of the latest modern techniques, the processing power available is not sufficient for practical use (typically MCU speeds of less than 100 Hz and a few kilobytes worth of RAM).

It should be mentioned that it is obvious from many sources of bearing failure analysis that although methods proposed seems to work well enough, the tests are performed on single bearings alone (often in a plumber block like arrangement). When these tests are performed in the industry while working in a machine many sources of noise are present, including:

- Electric motors
- Fluid or other couplings
- Conveyor belts
- Vibrating structures
- Mills
- Vibrating screens
- Crushers
- Vehicular noise

The de-noising techniques when performing bearing fault detection is fairly difficult with current computational resources. The case of synchronous averaging is often used in gear and shaft condition monitoring (Hochmann & Sadok, 2004) to eliminate noise. This technique requires a signal synchronous with the shaft speed, such as a tachometer signal. Because the Gear Mesh Frequencies (GMFs) are integer multiples of the shaft rotation frequency, the GMF will also be synchronous with the shaft rotation frequency and therefore averaging the shaft rotations will average the GMFs as well. However, the bearing fault frequencies are *not* synchronous with the shaft rotation frequencies (see Table 1 for details) and therefore averaging the shaft rotations will have the effect of *averaging out* the bearing frequencies.

Table 1: Selected gearbox frequencies (from a sample gearbox, as tested in the project)

Shaft rotation frequency	Gear Mesh Frequency	Bearing frequencies
24.67 Hz	493.33 Hz	12.12 Hz (FTF)
		64.95 Hz (2 × BSF)
		161.99 Hz (BPFO)
		232.68 Hz (BPFI)

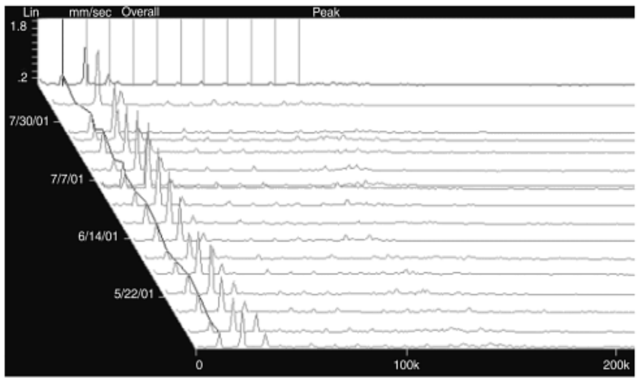


Figure 4: Typical waterfall plot (Girdhar & Scheffer, 2004)

As can be seen in Table 1, the GMF is exactly 20 times the shaft rotation frequency (due to there being 20 teeth on the particular gear). However, none of the bearing frequencies are integer multiples of the shaft rotation frequency and therefore, synchronous averaging cannot be used to de-noise the signal.

A number of other de-noising techniques exist, like non-synchronous averaging and spectral averaging (which is recognised to not being strictly noise reducing, but noise averaging). But these place a high premium on execution time in the embedded environment (with relatively little programming power and very little memory).

Another common practice is that of trending a series of vibration signals (see Figure 4). The premise of this practice is that a series of vibration measurements is made throughout the lifetime of the machine. When viewing spectral plots of these measurements in sequence, as in Figure 4, one can see when machinery faults start to occur and monitor their evolution.

This is a technique that is commonly used in the industry and , indeed, at the SASOL plant in Secunda where this technique is being applied to the monitoring of the gearboxes, fluid couplings and electric motors driving the conveyor belts as well as the bearings supporting the rollers of the conveyor belts.

Even though the algorithms do not use trending to diagnose a fault, it will be seen later in this report that the final version of the hardware does save time and frequency domain data in order for a human analyst to make a judgement either on a routine basis or when the hardware gives a warning that a fault might exist.

1.3 Electronic application considerations

1.3.1 Synopsis

In section 1.2.4 several advanced techniques are mentioned. Some of which, such as HFRT, are used very successfully to locate and diagnose bearing damage at a very early stage.

Due to the envisioned market positioning of the device, the objective of the device is not to locate bearing damage at a very early stage, but to alert an operator that there is a possibility that damage is already present on the to the device and that expert assistance should be sought.

The proposed market positioning therefore influenced the hardware specification of the system and dictated the use of fairly simple and electronic development platforms.

1.3.2 Processor type

Due to the convenient software package offered as well as the support availability it was decided to seriously consider the Microchip range of products. Microchip has on offer a range of 3 main types of Programmable Interface Controller (PIC) microcontrollers (Microchip, 2010). These are:

1. 8-bit PIC Micro Controller Units (MCUs)
2. 16-bit PIC MCUs and dsPIC Digital Signal Controllers (DSCs)
3. 32-bit PIC MCUs

The first device family listed above, the 8-bit PIC, represents the entry level microcontroller on offer by Microchip and represents the bottom range of functionality and performance. In contrast to this, the 32-bit PIC microcontroller family is the flagship range of microcontrollers on offer by Microchip. These microcontrollers represent a fairly new type of microcontroller (released in 2007) and have many advanced features, fast operation and generally the most memory. It was the midrange type of microcontrollers that seemed most attractive for this project. Specifically the 16-bit dsPIC DSCs seemed the most applicable to this project due to their Digital Signal Processor (DSP) capabilities.

The principal attraction for DSP is the fact that their hardware, software and instruction sets are optimised for numerical processing applications (Skolnick & Levine, 1997), crucial for rapidly processing digitised data as they have hardware multiplication capabilities, rapidly speeding up the execution of any algorithms requiring multiplication intensive routines.

1.3.3 Software implementation

1.3.3.1 Development rationale

The algorithms were first coded and tested in the Matlab environment. This was deemed an ideal environment for developing the routines due to its flexibility and ease of use. The Dynamics Systems Group of the University of Pretoria has several licenses for this software package as well as several toolboxes. This provided a number of advanced routines that proved valuable in the development of the algorithms.

However, this language is not readily supported by the chosen hardware architecture (see section 2) and therefore had to be translated to a more suitable language. The chosen language was ANSI C. This is a very powerful language for imbedded programmers and a very brief survey will convince any reader that it is used widely throughout the industry.

Therefore a license for MicroC Professional was obtained. This is an ANSI C based programming language used to program the Microchip series of MCUs.

1.3.3.2 Algorithm efficiency primer

Due to limited processing power available (compared to a PC), a conscious effort was made to select the most efficient algorithms.

In the context of this project, code efficiency is accepted to be execution time – which directly relates to the number of computations required for a routine to execute as well as the type of digital number used for the computation (such as floating point numbers, 8-bit integers, 16 bit integers, to name a few).

The easiest way to increase efficiency was then to use integers as much as possible and this philosophy was applied at every opportunity. Secondly, a conscious effort was made to select algorithm implementations with as few computation steps as possible. The way this was quantified was with the well-known “big-oh” notation, as is often used in computer science (Avigad & Donnelly, 2004).

Big-oh notation forms part of the asymptotic analysis branch of mathematics and allows one to observe the behaviour of a mathematic function when it approaches 0, a constant, or infinity. It does so by supressing lesser order terms which become inconsequential when the tending towards the given boundary.

Formally, it is defined as (Graham, et al., 1994):

$$f(n) = O\big(g(n)\big) \quad \text{as } n \to a \qquad \text{[Eq. 2]}$$

which implies that:

$$f(n) \le C|g(n)| \quad \text{as } n \to a \qquad \text{[Eq. 3]}$$

In words this means that when *n* tends towards a set limit, the function *f(n)* is at most a constant times the absolute value of a related function *g(n)*. For example (Graham, et al., 1994):

$$f(n) = \frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n \qquad \text{[Eq. 4]}$$

There exists the function:

$$g(n) = n^3 \qquad \text{[Eq. 5]}$$

for which can be written:

$$f(n) \le C|g(n)| \quad \text{as } n \to \infty \qquad \text{[Eq. 6]}$$

Because when *n* approaches infinity, the lesser order terms (the n^2 and n terms in this case), becomes insignificant and one can find a (unspecified) number C which will satisfy the condition. Broadly speaking, in the example above, the function *f(n)* and *g(n)* will behave in a similar manner when approaching infinity.

The manner in which this concept will be used in this project is to evaluate the efficiency of algorithms by comparing the number of computational steps in their execution. In this context, the function *f(n)* represents the actual number of steps required to perform the algorithm and n represents the length of the vector which serves as the algorithm input. Therefore, *g(n)* represents a simplified measure of the steps required the perform the algorithm when the vector length is large.

A simple example of this is the sorting algorithm discussed below. The heapsort algorithm is a $O(n\log_2 n)$ algorithm, whereas the elementary bubblesort algorithm is a $O(n^2)$ type algorithm. This then implies that as the vector size starts getting large, the number of computational steps approximately squares with vector size in the case of bubblesort, yet with the heapsort a logarithmic relationship exits. This has a direct impact on execution time (which is even more pronounced when dealing with floating point numbers) and the heapsort algorithm would be the better choice.

1.3.3.3 Sorting algorithm

Sorting algorithms, such as the examples given above, are very commonly used in the programming industry (Press, et al., 1995) and has therefore received a significant amount of attention. In the algorithms deployed in this project, the sorting function was used extensively. It was mostly used to find a maximum and minimum value as well as the median value (along with the index) within a vector.

The MikroC Pro software package offers both a max and min function in its library, but these functions come with the severe limitation that they only work on integer vectors. This renders them useless for the specific application. Therefore, a sorting algorithm was used to sort the array and then pick the last element in the rearranged array (the arrays are sorted in ascending order). A separate vector containing the indexes was created and manipulated so as to reflect the sorting process.

From the research it was obvious that two candidates were suitable: Quicksort and heapsort. Although heapsort was on average slower that Quicksort, its worst case was only 20% slower than the average (Press, et al., 1995) and is a true $n\log_2 n$ algorithm in addition, it is an in place sorting algorithm requiring no extra memory space (very attractive proposition for an embedded system with very limited storage space). Whereas Quicksort's worst execution time was orders of magnitude slower than its average and its implementation is much more involved. On balance, it was therefore decided to implement the heapsort algorithm for its simplicity, in place properties and its consistency in execution time. The specific algorithm employed is a somewhat modified version of the code given in Press, et al. (1995)

1.3.3.4 Fast Fourier Transform performance

1.3.3.4.a Basic Fast Fourier transform

Because the Discrete Fourier Transform (DFT) algorithm is, computationally speaking, very expensive (Lai, 2004) – it is a N^2 type algorithm – the much more efficient FFT algorithm is normally used in computer applications.

This algorithm, requiring only about $N\log_2 N$ operations (Grover & Vollmer, 2010), developed by Cooley and Tukey, basically decomposes a N-point DFT into two N/2-Point DFTs. These are then broken down into N/4-point DFTs and so on until a DFT of size 2 requires computing which is trivial (Grover & Vollmer, 2010). These DFTs are combined in a recursive way to then form the N-point DFT of the original series. This is a so-called Decimation In Time (DIT) implementation - as opposed to Decimation In Frequency (DIF) - and the results are in bit reversed ordering (MikroC, 2010).

A key concept with this implementation is the so-called "Twiddle Factors" (Press, et al., 1995). These are the coefficients used to recombine the decomposed DFTs. The implication of this is computing a 15-bit FFT would require 32 768 twiddle factors. The standard MicroC library only goes up to a 9-bit FFT.

It was therefore decided use one of the clearest derivations of the FFT (Press, et al., 1995) formulated by Danielson and Lanczos in 1942. This derivation still uses the DIT FFT algorithm, but instead of the twiddle factors being pre-stored in non-volatile memory, they are calculated and used as part of the running routine. The algorithm found in (Press, et al., 1995) was extensively modified for use in an embedded application.

Lastly, the algorithm described can only be used on data samples having a size of any power of two. There are formulations, notably of the Winograd Fourier transform types (Press, et al., 1995), that are fast for any size of data samples, but these use a complex indexing system and is not a true in-place algorithm which makes it unsuitable for an embedded system in which memory comes at a premium. There also exist radix 3 algorithms which can often be 20 to 30% faster (Press, et al., 1995) than the radix 2 implementation, but these can easily devolve into a N^2 type of algorithm instead of the $N\log_2 N$ type algorithm (Press, et al., 1995). The likely small increase in speed was therefore deemed not worthwhile when evaluated against the risk of a possible severe decrease in speed.

1.3.3.4.b Fast Fourier Transform optimisation

The source for this optimisation, implemented in the Beta prototype, comes from the same publication containing source code and the entire mathematical philosophy. However its implementation is somewhat involved, especially as the code had to be adapted for embedded use, which is why it was not included in the Experimental Development Model (XDM) and Alpha prototype.

The FFT function (as explained in section 1.3.3.4.a above) previously used in the XDM and Alpha prototype is valid for cases of purely real data, purely imaginary data or real and imaginary data. However, this leaves much room for optimisation if one is aware of the type of data to be used. It mostly happens that one will use only real input data (Smith, 1998), as the input signal such as from an accelerometer is a real analogue voltage signal, which is then digitised.

In this section only the means by which it is accomplished will be explained, not the working itself – for an in depth study into the matter the referenced literature is recommended.

Because the C language (used in the code of the hardware) does not have native imaginary number support (vital for frequency domain techniques), one is forced to make the convention that every second memory location will denote an imaginary component of the first real component. Thus one element of an array (for example) will have two memory locations, one for the real part of the element and one for the imaginary part. When one then starts off with purely real numbers, the implication is that every second memory location is set as zero. As such, one is forced to utilise the full complex FFT algorithm. This proves to be inefficient both in terms of storage and execution time.

The potential for optimisation comes from two *properties of the* Fourier Transform: The *first property* is that for purely real (or in fact imaginary) data certain symmetries exist, notably (Press, et al., 1995):

If h(t) is the time domain data and H(f) its frequency domain transform, then
*For purely real h (t): H(-f) = H(f)**

This states that the complex value of a negative frequency component is equal to the complex conjugate of the positive frequency if the input data is purely real. This implies that the magnitudes of the positive and negative frequencies are the same and forms a “mirror image”, therefore half the resulting spectrum is redundant. As a matter of fact, symmetry exists for purely imaginary input data as well; this will be used later in the explanation (Press, et al., 1995):

*For purely imaginary h(t): H(-f) = -H(f)**

The *second property* that has been alluded to previously, is that for an in-place FFT implementation one has to leave space for the imaginary components that will result from the FFT computation, therefore in the input data string, every second memory location (representing imaginary data) has to be set to zero.

With these two areas of optimisation a computational scheme can be formed in which one can fill all the memory locations with input data (instead of every second memory location). This will yield twice the data with the same memory available (as every second location is not initially set to zero, but filled with meaningful data); however one would then usually end up with inadequate memory (two times too little) space as there is not enough space for the imaginary components of the result. This space is accounted for with the fact that the magnitude spectrum of a real function is double sided and thus redundant, thereby the other half of the memory space required is made up for. A graphical illustration of this is given below:

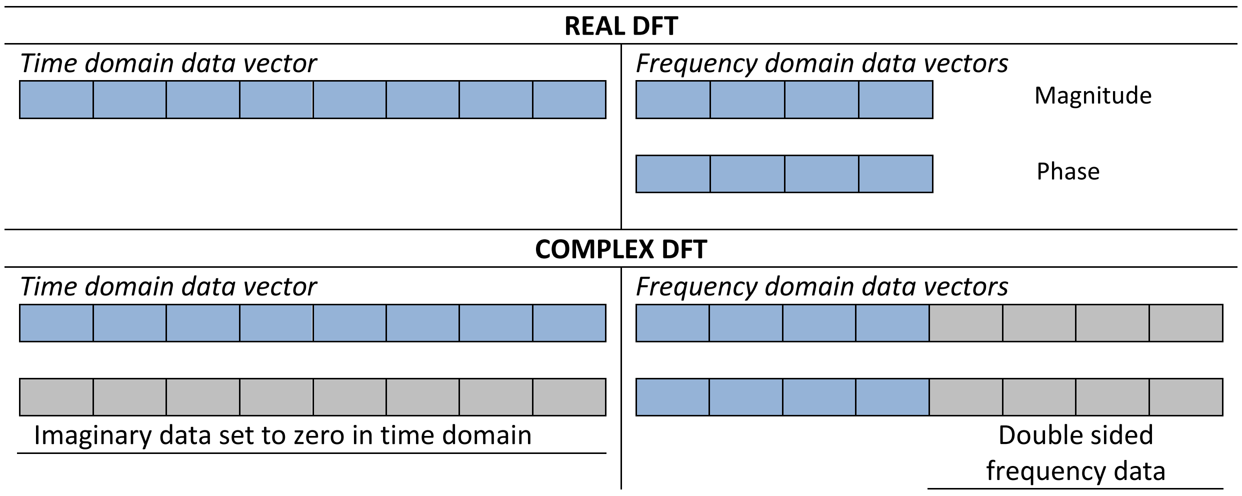

REAL DFT
Time domain data vector
Frequency domain data vectors
Magnitude
Phase
COMPLEX DFT
Time domain data vector
Frequency domain data vectors
Imaginary data set to zero in time domain
Double sided frequency data

Figure 5: Visualisation of the difference between the optimised and non-optimised FFT algorithm

In the figure above the cells represent memory locations. Blue cells represent useful data points while the grey cells represent wasted data (and thus wasted memory) and is to be eliminated by optimization. This wasted data is either in the form of zeros (in the time domain, because the real input has no imaginary components) or mirror imaged data (in the frequency domain, as real input data has a mirror imaged output as a result). It is very important to note that this optimisation is valid only for purely real input data, as complex input data will not result in zero imaginary (time domain) input data and a mirror imaged output (frequency domain) data. These two consequences of purely real input data is the key to make the real DFT algorithm work.

The result of this optimisation is thus twofold. 1) The first benefit from using this optimisation is that one can use two times less memory or, more commonly, twice more data for a given memory size – very important for memory limited devices such as DSCs. This results in a frequency resolution that is now twice as fine as without the optimisation. 2) The second benefit is that per useful data point, the algorithm operates twice as fast. This results from the fact, that the FFT algorithm still only processes the same number of data points as before. The only difference is that all the data points that are processed, are *meaningful data points*. The inefficiency of the non-optimised algorithm was that it still processed the zero valued imaginary data points, even though they are not meaningful towards the output of the algorithm. It must nevertheless be understood that both algorithms processes the same number of data points. This can be understood better by looking at Figure 5. As can be seen, the number of useful data points (blue cells) on Figure 5 is equal for the Real DFT (i.e. the optimised algorithm) and the Complex DFT (the non-optimised algorithm). However, the complex DFT contains just as much redundant data in the form of imaginary zero values. Even though these values are completely redundant, they still *have* to be processed, resulting in an equally redundant double sided spectrum. This is a property of unoptimised (for real data) algorithm. Of course, for imaginary input data, the initial zeros in the time domain will be filled with meaningful data and the resultant spectrum will not be double sided.

While the specific internal workings of the real DFT algorithm is beyond the scope of this literature, it is easy to see from this illustration that a 2× memory saving can be made if the real DFT is used instead of the complex DFT in the case of purely real input data.

For the sake of comprehensiveness, it should be mentioned that another optimisation exists, which works on a similar principle. Instead of filling the data string in the memory entirely with single real input data, it is also possible to interweave the data string with two sets of data. With this method, it is possible to perform the transform of both sets of data with one operation. Therefore, instead of doing both sets of data separately (without the single-sided optimisation, thereby taking twice as much memory and time), one can do both at the same time with twice as little memory and twice as fast than would be required using the non-optimised algorithm.

1.3.3.5 Window performance

When experimentation was conducted, both in the laboratory and in the field, it was found that the frequency spectrum is fairly course with leakage still interfering between closely spaced frequencies. This interferes somewhat with the spectrum and the median computations used in the damage detection algorithms. This is partly due to somewhat low resolution of the Alpha prototype (0.48Hz), but optimising the window was deemed an inexpensive way to get a better quality spectrum, specifically distinguishing between closely spaced frequencies.

1.3.3.5.a Spectral resolution

With the optimised real FFT algorithm in-place, it is possible to fully utilise the memory capacity available. The algorithm (and its memory saving capabilities) will allow a 16-bit FFT to be performed on the data (yielding 32768 discrete frequencies). A sampling frequency of 6.25 kHz will yield a Nyquist frequency of 3.125 kHz. The spectral resolution will therefore be:

$$\text{Res} = \frac{3\ 125\ \text{Hz}}{32\ 768\ \text{bins}} \approx 0.095\ \text{Hz/bin} \qquad \text{[Eq. 7]}$$

This compares to the (best case) resolution SASOL uses in their condition monitoring department of 0.25Hz. The frequencies spaced closest together in typical data are about 1Hz. Unfortunately this is not the complete picture, due to the phenomenon of Minimum Resolution Bandwidth, as explained below.

1.3.3.5.b Minimum resolution bandwidth (MRB)

This is a measure of the minimum separation required (in bins) between two adjacent frequency peaks (of equal magnitude) to fully distinguish them apart (Bores Signal Processing, 2009). According to the previously cited source, the rule of thumb for MRB is: to distinguish two frequencies of equal magnitude, the spacing (in bins) between them is equal to the half power points (-3dB) of the window's frequency response.

However, this assumes incoherent addition of the frequency components, whilst the FFT output is the coherent addition of the frequency components (Bores Signal Processing, 2009). Therefore, again by the previously cited source, the -6dB defines the MRB.

This parameter is important when considering spectral resolution and should be taken into account.

1.3.3.5.c Coherent gain

It is common during the computation of the frequency spectrum to take the coherent gain into account when using a window. Coherent gain is introduced when using a window as the very act of multiplying the time domain signal with the window introduces a distortion effect that alters the amplitude of the signal (National Instruments, 2009). It is computed simply:

$$\text{Coherent Gain} = \sum_{n=0}^{N} w(n) \qquad \text{[Eq. 8]}$$

Where
N - Number of discrete points in a window

Simply stated, it is the sum of all the discrete amplitude points in the time domain window (Bores Signal Processing, 2009). In order to get the correct amplitude in the frequency domain, one then divides the signal in the time domain by the coherent power gain (National Instruments, 2009).

1.3.3.5.d Window comparison

1.3.3.5.d.i Spectral comparison

Highly influential parameters, in terms of leakage and signal-to-noise ratio (Bores Signal Processing, 2009), were found to be highest side lobe level and worst-case processing loss. The best performing windows in these cases were (Harris, 1978):

- Blackman-Harris
- Dolph-Chebyshev
- Kaiser-Bessel

The more common Tukey, Poisson, Hanning and Hamming windows were all found to be inferior.

The frequency response of some various windows considered are given on the next page, along with a uniform window for reference. This plot illustrates the leakage effect in terms of magnitude vs. affected bins.

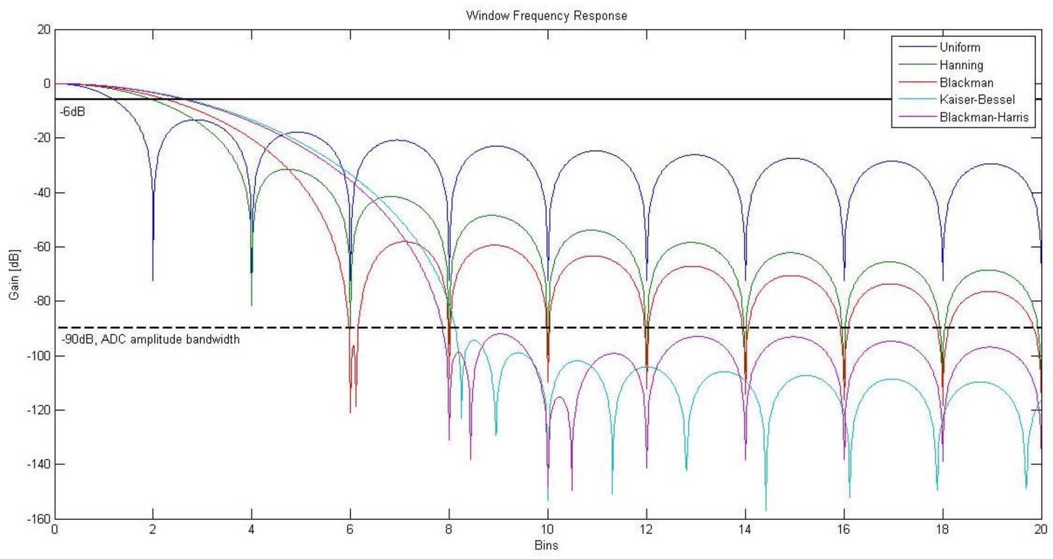


Figure 6: Leakage comparison of considered window functions

In the figure, the -6dB boundary is demarcated with a solid black line; in addition the ADC amplitude bandwidth is illustrated by a dashed black line (the ADC has an amplitude bandwidth of about 90dB, below which it falls below the minimum resolvable value of the ADC). The trade off in window performance is evident in the figure above: Main lobe width versus side lobe height.

Firstly, it is visible that the side lobes of the Kaiser-Bessel and Blackman-Harris fall below the ADC amplitude bandwidth giving them the best possible noise leakage limitation from further from the frequency peak. One can also see the uniform window has the smallest side lobe width, but very high (and thus undesirable) side lobe height.

The best window in this case would be the one that gives sufficient MRB and leakage suppression for the application while providing for the fastest computation time.

The following figures illustrate the frequency response in practice. It shows several plots of two frequencies of equal amplitude subjected to different windows in each plot (the time series was divided by the coherent gain of the windows so that the give the same spectral amplitude, as explained in the section 1.3.3.3.3).

It is useful to compare the window effects this way, as it allows one to easily visualise the ultimate effect of the window functions when used in practice.

Figure 7 and Figure 8 below show the frequency spectrum of two frequencies (conveniently chosen 45 and 46 Hz) of equal amplitude separated by 1Hz, as is the minimum of frequency separation in the tested data. The test shares the spectral resolution of the optimised hardware, namely 0.09Hz.

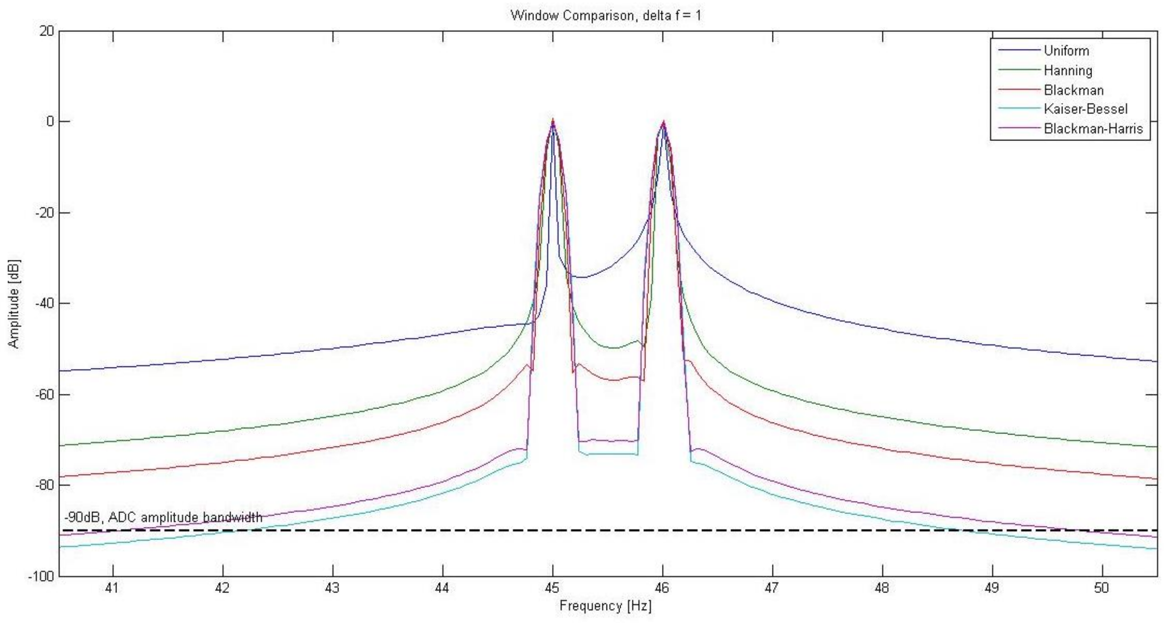


Figure 7: Sample spectrum of two closely spaced signals (frequency difference of 1Hz)

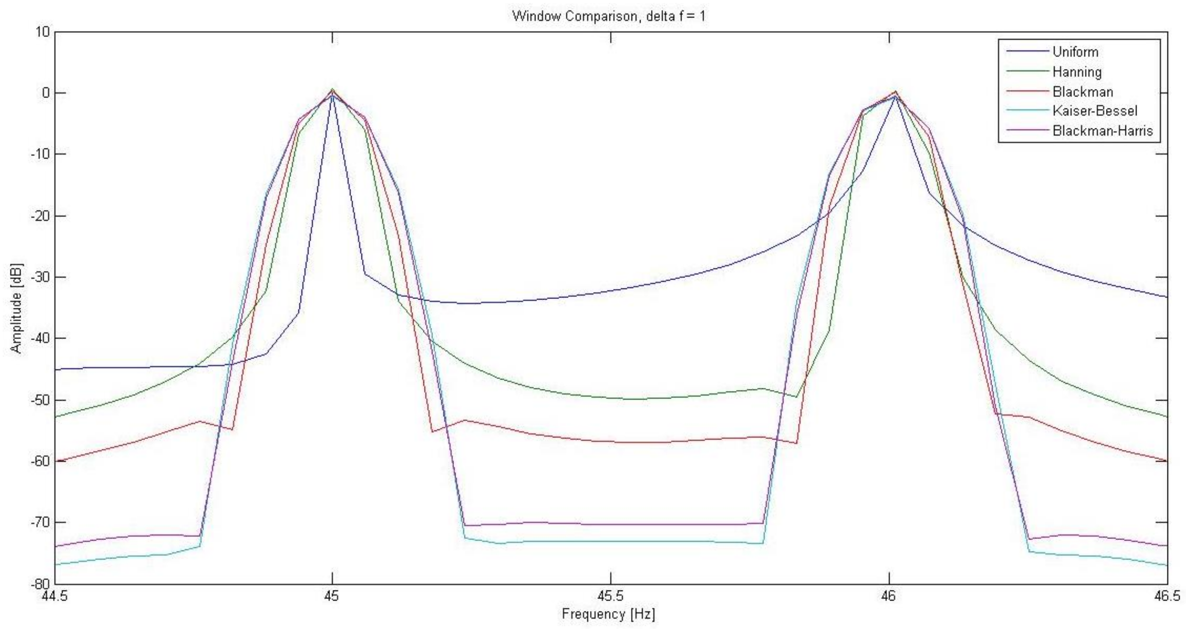


Figure 8: Sample spectrum of two closely spaced signals (frequency difference of 1Hz, narrower band)

Details of interest are the peak width and the noise level. The top figure shows the general leakage level in the vicinity of the peaks. For reference, one can immediately see that not using a window (Uniform) produces frightening leakage. In terms of peak width, not using a window produces the sharpest peak, but at the expense of insufficient drop-off further from the peak – with the effect of imprecise peaks. The Hanning does somewhat better with fair attenuation and drop-off, then the Blackman, Blackman-Harris and Kaiser-Bessel. However, with a larger drop-off and attenuation comes an increase in peak width, which has an adverse effect on MRB. Though at this frequency separation, both peaks are clearly distinguishable. The question now becomes, how close can two frequencies get before they can be separated? One can look to the MRBs of the windows to determine this, from Figure 6 and National Instruments (2009), both giving approximately the same results, we get:

Table 2: MRB of the various windows

Window	MRB (bins)
Uniform	1.21
Hanning	2.00
Blackman	2.30
Kaiser-Bessel	2.73
Blackman-Harris	2.65

The figures below illustrate how the different windows affect signals as they get closer together. From upper left to lower right, the separation between their frequencies are: 2 bins, 3 bins, 5 bins and 10 bins.

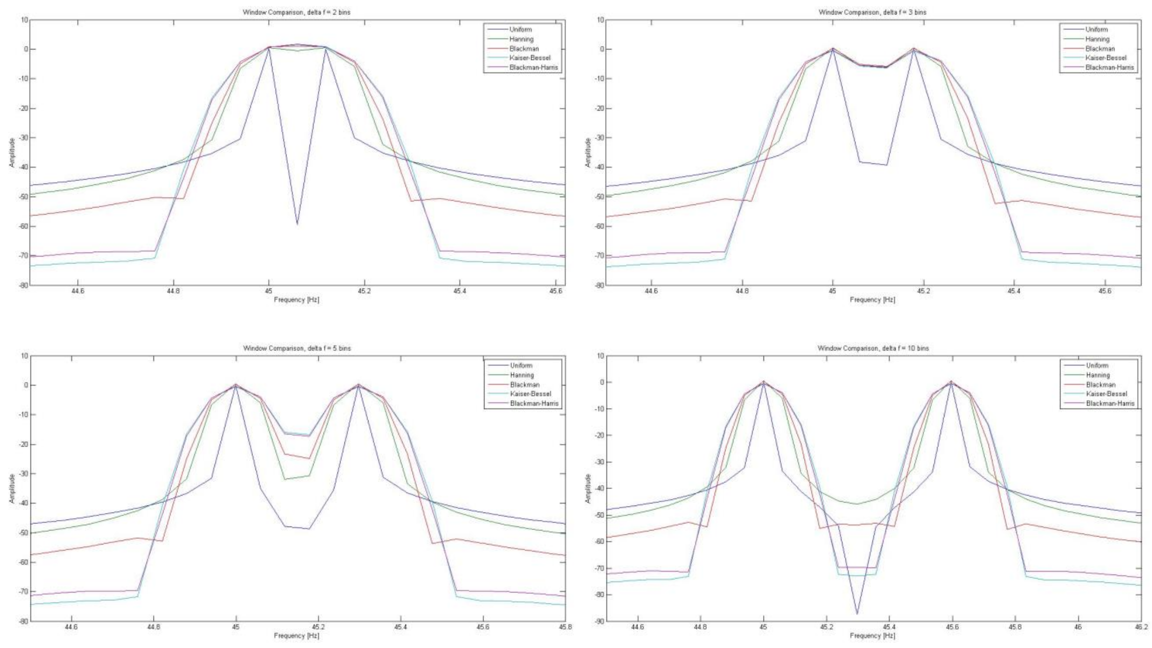


Figure 9: Comparison of window effect when two closely spaced frequencies are considered

It is obvious that not using a window, one gets the sharpest frequency peaks, and frequencies separated by just two bins are distinguishable. However, it is also evident (especially from Figure 9) that the amount of leakage is completely unacceptable. For three bin separation, the windows are about equal in their resolving ability. With discrete peaks barely separable (this is because all of the windows have a MRB of less than 3). For 5 bins, the Hanning window separates the peaks the best, followed by the Blackman, Blackman-Harris and Kaiser-Bessel. At 10 bins the situation is reversed, the Kaiser-Bessel separates frequencies the best, followed by the Blackman-Harris, Blackman and Hanning.

1.3.3.5.d.ii ***Comp***utational comparison

In an embedded system, where computational resources come at a premium – unlike a PC – it is important to consider the complexity of the algorithms due to the fact that a more complex algorithm might take significantly longer to execute. Therefore, the running time of the windows will now be compared and weighed up against the spectral advantages.

As a reference, the mathematical equations that describe the windows are given below. In all the cases (Harris, 1978) (Ifeachor & Jervis, 1998):

$$0 \leq \mathrm{n} < \mathrm{N}$$

N is the number of discrete points that comprise the window.

Hanning

$$\mathrm{w(n)}_{\mathrm{Hanning}} = 0.5\left(1 - \cos\frac{2\pi \mathrm{n}}{\mathrm{N}-1}\right) \qquad \text{[Eq. 9]}$$

Blackman

$$w(n)_{\text{Blackman}} = a_0 - a_1 \cos\frac{2\pi n}{N-1} + a_2 \cos\frac{4\pi n}{N-1} \qquad \text{[Eq. 10]}$$

where, typically:

$$a_0 = 0.42 \qquad a_1 = 0.5 \qquad a_2 = 0.08$$

Kaiser-Bessel

$$w(n)_{\text{Kaiser-Bessel}} = \frac{I_0\left(\pi\alpha\sqrt{1-\left(\frac{n-N/2}{N/2}\right)^2}\right)}{I_0(\pi\alpha)} \qquad \text{[Eq. 11]}$$

where:

$$I_0 = 1 + \sum_{L=0}^{\infty} \frac{\left(\frac{x}{2}\right)^{2L}}{(L!)^2} \qquad \text{[Eq. 12]}$$

I_0 is the modified Bessel function of the first kind, with the range of L defined as going from zero to infinity. Typically though, bounding L between zero and 32 is sufficient and α typically 4 (Ifeachor & Jervis, 1998).

Blackman-Harris

$$w(n)_{\text{Blackman}} = a_0 - a_1 \cos\frac{2\pi n}{N-1} + a_2 \cos\frac{4\pi n}{N-1} - a_3 \cos\frac{6\pi n}{N-1} \qquad \text{[Eq. 13]}$$

where, typically (Mathworks, n.d.):

$$a_0 = 0.35875 \qquad a_1 = 0.48829 \qquad a_2 = 0.14128 \qquad a_3 = 0.01168$$

One can see by comparing the Hanning, Blackman and Blackman-Harris that all of them involve an increasing number of cosine terms: with one for the Hanning, two for the Blackman and three for the Blackman-Harris. This corresponds to the times of execution with Blackman and Blackman-Harris being two and three times as long, respectively, as the Hanning window.

The results for the time of execution are given below. The window was applied to a vector containing only ones and applied to vectors of varying lengths. The time taken to execute is given in the table below.

Table 3: Time to execute the various windows

Window	**Time to execute (s)**				
	N = 512	N = 2048	N = 8096	N = 327678	N = 65 536
Hanning	2.8	4.2	10.7	36.3	71.0
Blackman	3.2	6.1	18.7	69.3	135.3
Kaiser-Bessel	>120	>120	>120	>120	>120
Blackman-Harris	3.6	8.2	27.3	103.3	203.9

It should be mentioned that the tests above show the results based on the standard 10MHz speed of the processor, it can be completed much more rapidly (8×) by the Beta prototype that operates at 80MHz.

Firstly, it was realised during the very first test that the Kaiser-Bessel window would execute extremely slowly and clearly was not suitable to an embedded application. When it was tested and did not finish after two minutes when merely computing 512 points, the test was aborted. It was therefore decided from the outset that it was not a feasible window and not tested further.

It can be seen that the Hanning window is the least computationally effective, followed by the Blackman, Blackman-Harris and the Kaiser-Bessel. The Blackman-Harris is about 50% more computationally intensive than the exact Blackman, and looking at the equations, it is easy to see why: it has one more multiplication term (keep in mind floating point multiplication uses the largest amount of resources) – totalling three, whereas the Blackman has only two.

Seeing that the Blackman is still relatively fast (albeit approximately 2× slower than the Hanning) and has a somewhat better leakage attenuation, but about the same main-lobe width, it is considered superior. The Blackman-Harris and definitely the Kaiser Bessel are considered too computationally expensive for the application, although these windows have very good leakage attenuation. Their main-lobe widths however are inferior to both the Hanning and Blackman and further detracts from their attractiveness. Thus, the Blackman window is the window of choice and was chosen for this project.

1.3.3.6 Achieving higher ADC resolution using oversampling and decimation

For the Beta prototype implementation, it was decided to use the on-board ADC of the MCU in conjunction with an oversampling decimation routine. What follows is a summary about the theory of operation. It is heavily based on the Microchip application note AN1152 (Microchip Technology Inc, 2008).

The process of quantising an analogue signal into digital words introduces quantization noise, the smaller the word length, the greater the noise introduced. The signal-to-quantization noise ratio is defined as:

$$\mathrm{SNR_Q} = 6.02\mathrm{N} + 4.77 + 20\log_{10}(\mathrm{L_F})\ [\mathrm{dB}] \qquad \text{[Eq. 14]}$$

where N is the number of bits and L_F is the loading factor, defined as the ratio of the RMS value of the input analogue voltage to the peak ADC voltage. For a sine wave L_F = 0.707 and the equation become:

$$\mathrm{SNR_Q} = 6.02\mathrm{N} + 1.77\ [\mathrm{dB}] \qquad \text{[Eq. 15]}$$

From the above equation it can be seen that the SNR_Q improves by 6.02 dB per bit and the higher the number of bits, the better the SNR_Q becomes. The SNR_Q of a 12bit ADC is about 74.01 dB and that of a 16bit ADC is about 98.09 dB. It will now be shown how the SNR_Q can be improved without increasing the word length of the ADC.

The Power Spectral Density (PSD) of the quantization noise with a flat spectrum is given by:

$$\mathrm{PSD_{QN}} = \frac{(\mathrm{lsb\ value})^2}{12\mathrm{fs}} \qquad \text{[Eq. 16]}$$

One can therefore see that to decrease the PSD of the quantization noise, it is necessary to either decrease the LSB value (which means increase the word length) or increasing the sampling frequency – which leads to oversampling. The SNR$_Q$ improvement after oversampling is now given by:

$$\mathrm{SNR_{oversampling}} = 10\log\frac{\mathrm{f_{os}}}{\mathrm{f_N}} \qquad \text{[Eq. 17]}$$

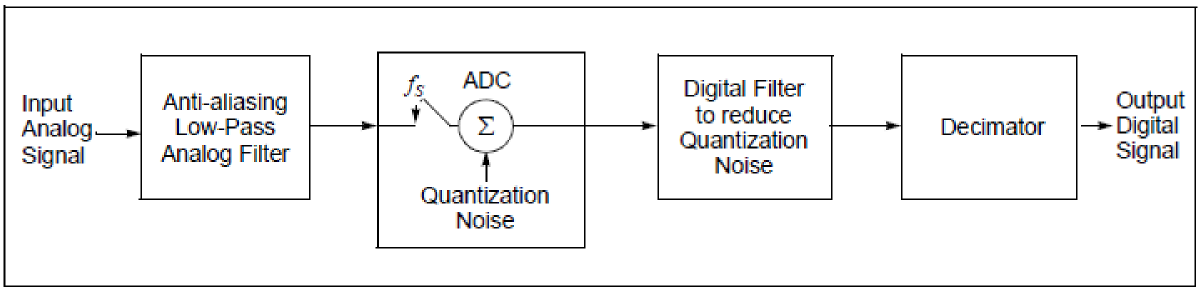
where F$_{OS}$ represents the sampling frequency (when oversampling) and F$_N$ represents the Nyquist frequency. Therefore, the overall SNR is:

$$\mathrm{SNR_{overall}} = 6.02\mathrm{N} + 1.77 + 10\log\frac{\mathrm{f_{os}}}{\mathrm{f_N}} \qquad \text{[Eq. 18]}$$

Suppose we have a P-bit ADC and a Q-bit ADC with Q>P, the sampling factor is calculated as follows:

$$\frac{\mathrm{f_{os}}}{\mathrm{f_N}} = 10^{0.602(\mathrm{Q-P})} \qquad \text{[Eq. 19]}$$

The following block diagram shows the stages of the data acquisition process:



Input Analog Signal
Anti-aliasing Low-Pass Analog Filter
f_S
ADC
Σ
Quantization Noise
Digital Filter to reduce Quantization Noise
Decimator
Output Digital Signal

Figure 10: Block diagram of signal acquisitioning and processing (Microchip Technology Inc, 2008)

The analogue signal is oversampled and an anti-aliasing filter applied. The remaining signal is then further subjected to a digital low-pass filter to suppress the higher frequency quantization noise and to negate the effect of aliasing that may arise after down sampling. After filtering, the signal is then further passed through a decimator to downgrade the rate. At this point the sampled points can be used in the DSP. The signal that is obtained at the end of this process has the SNR of Q bit ADC even though a P bit ADC was used.

It will now be calculated what the effective number of bits will be after the oversampling algorithm is employed.

$$\mathrm{SNR_{overall}}\ \text{of an N bit system} = \mathrm{SNR_{overall}}\ \text{of a 12 bit system with oversampling} \qquad \text{[Eq. 20]}$$

$$6.02\mathrm{N} + 1.77 = 6.02(12) + 1.77 + 10\log\frac{(100\ 000)}{(6\ 250)}$$

$$6.02(\mathrm{N}-12)=10\log\frac{(100\,000)}{(6\,250)}$$

Solving for N:

$$\mathrm{N}=14$$

As can be seen, for the resolution of the original system has been partly regained. The benefits of this technique are now simpler and more efficient source code, using a simpler hardware layout and a system that is marginally less expensive. In addition, the full operational capability of the DSP functionality in the MCU can now be used. This opens up new avenues in future expandability of the algorithm.

1.4 Scope of Research

The progress of this project followed four developmental phases, as described below.

1.4.1 Literature survey

As a starting point, a literature survey was undertaken to explore the current trends in the industry and academic research.

Firstly, this provided a clearer background of what is currently relevant and required in the industry as well as what the administrative, financial and technical challenges are in the industry. Secondly, it also yielded information as to the capabilities of modern electronics, upon which a judgement could be made regarding suitability of various systems for the application. Finally, this information had to be kept in mind when the algorithms considered for inclusion were chosen considering the processing power available from the chosen micro-electronics as well as bearing in mind the ultimate goal of the project, i.e. a protection system.

1.4.2 Algorithm development

The next phase involved the algorithm development. It started out with a field measurement campaign at the SASOL plant Secunda, as described in section 5.2. That initial measurement campaign yielded a large amount of data from a number of different gearboxes in various states of repair.

With guidance from the literature study and using the data gathered from the measurement campaign, the algorithm was developed in the Matlab environment. This platform is highly suited to rapid development and adaptation.

Although the project started with this phase, it was an ongoing effort throughout the project and ran concurrently with the other phases.

1.4.3 Experimental hardware work

When the choice of MCU architecture was made, an off-the-shelf development system was obtained. This system served as an "electronic development laboratory" and allowed for general familiarisation of the environment and the porting of the algorithm from Matlab into C.

It was clear from the outset that this system would not be sufficient to see the project through, due to its large and impractical dimensions and very limited features. However, since it used the same compiler it served as a valuable test bed while the Alpha prototype was designed and manufactured.

1.4.4 Alpha prototype

The Alpha prototype was the first focussed design with this specific project in mind. It incorporated numerous upgrades and features lacking from the experimental hardware. Importantly, the Alpha prototype was capable of measuring and processing a significant amount of data and giving feedback to the user.

This system was extensively tested in the laboratory as well as at the Sasol plant in Secunda. It also served as a temporary test bed and underwent significant modification while the Beta prototype was being designed and built to test new functionality being built into that hardware.

Due to it being the first system, several critical flaws were discovered which would require a system redesign to adequately address. Therefore, both in the laboratory and in the field, many operational and technical lessons were learned and incorporated into the design of the Beta prototype.

The most important shortcomings were the dependence on a controlling PC, i.e. it was not independent. It also did not have an on-board power supply for the ICP accelerometer, which necessitated the use of an external and unreliable power supply. Lastly, due to limitations of the compiler, it had to operate at one eight of its designed speed, which meant it performed very slowly.

Crucially, the benefit of this system was that the entire ecosystem of routines comprising the algorithm could be tested as a unit. This differs from the experimental system, described in paragraph 1.4.2, in which the routines often had to separated and tested individually. This is primarily due to the fact that Alpha prototype had significantly more memory and the capability to sample an external signal.

Additionally, this system afforded the opportunity to write a number of device drivers, such as the memory and ADC drivers as well as implement the corresponding modifications to the algorithms. Due to the fact that the Beta prototype used the same components, the majority of the drivers could be reused with no or little modification.

Finally, the system was highly modified during the manufacturing phase of the Beta prototype. Several delays were encountered during its manufacture and for the sake of driver programming of some of the newer facilities of the Beta prototype, the Alpha prototype were modified with a number of subsystems of the Beta prototype.

1.4.5 Beta prototype

The specification for the Beta prototype was generated after the Alpha prototype finished with its field testing. Several key areas of improvement were identified to address the shortcomings of the Alpha prototype. The redesign of the circuit layout provided an opportunity to add several useful features.

Areas which were improved upon on the Alpha prototype were:

- On-board power ICP power supply
- Corrected circuit design of LCD
- Corrected circuit design of push buttons
- Elimination of the external ADC and use of the on-board ADC on the MCU
- Combined use of the LCD and push buttons allowed the use of the original design speed, enabling an eight times speed increase over the Alpha prototype
- Combined use of the LCD and push buttons allowed the Beta prototype to be independent of a controlling PC

New features on the Beta prototype are:

- SD-Card reader
- Serial port
- Interfaces for future development
- Lead-acid battery recharge circuit

The result of these modifications produced a near commercial ready system capable of recording and storing external signals, processing and storing the results for later analysis, whilst operating independently of a PC.

It was calibrated in the Sasol Laboratory for Structural Mechanics at the University of Pretoria and extensively tested at the Sasol plant in Secunda, where the data gathered correlated closely with the results measured by Sasol.

The result is a system that performs the role of a 'Vibration protection system' and offers an analyst the tools to make a judgement if an alert is given.

1.5 Summary of chapters

What follows now is a brief guide of the chapters comprising this dissertation.

Chapter two details the design evolution of the Alpha prototype and the Beta prototype as well as the role of the experimental development model. It recounts the role each set of hardware played in the project, the weaknesses of the various hardware systems and they were corrected in the subsequent hardware. This account of the hardware development therefore provides a broad scope of the progression of the hardware.

The development of the algorithm is explained in chapter three. The work in this chapter draws heavily the experimental results of the initial data gathering campaign. The rationale behind the algorithm and certain decisions are explained in this chapter. Finally a comparison is made as to how the algorithm performs. The chapter is laid out in order of how the algorithm works and therefore provides a step by step walkthrough of the device operation.

The fourth chapter explains the laboratory test work performed on the hardware iterations. This work was completed before the field work started in order to verify the operation of the device, be it the Alpha or Beta prototype.

Chapter five details the various field measurement expeditions. The field work done during the initial data gathering exercise as well as the Alpha and Beta hardware is explained. Specifically, this chapter explains the goals of the various field work exercises as well as the procedures followed. The chapter explains why and how field testing was performed without going into the results.

Next, in chapter six, the results of the three different field expeditions explained in chapter five are presented and discussed. This chapter provides an impression of how the algorithm developed from functioning within a computer separate from the data acquisitioning system to being eventually fully integrated into a compact system containing all the subsystems necessary to acquire, process, store and display a result.

Finally chapter seven presents the conclusion in which the summary of findings are listed for each device tested. A cost analysis is also presented along with recommendations and future work. The chapter is presented in a commercial sense and should be read as a general guide to take the final beta prototype to a commercially viable product.

2. Design Philosophy

2.1 Preliminaries

Initial discussions with engineers in various industry sectors led to the investigation into the range of Microchip products and it was decided that the core of both the XDM and the Advanced Development Model (ADM) prototypes would be a Microchip 16-bit dsPIC as mentioned in section 1.3.2.

The dsPIC is a type of Programmable Interface Controller with DSP capabilities. The instruction set and hardware design of these PICs are optimised for numerical processing and they contain a particularly good multiplication capability. See section 1.3.2 for more details.

A choice of three programming languages was considered: Pascal, Basic and ANSI C. The programming software was available in these three languages and it was ultimately decided to go for the C route as it is the most widely used and most flexible programming language of the three, particularly when it comes to program efficiency and memory usage (Kochan, 1988).

The algorithms were developed in a Matlab environment as it is much more flexible for this kind of data processing. However, the Matlab algorithms were subsequently transferred to C to work on the dsPICs.

2.2 Experimental development model (XDM)

2.2.1 Overview

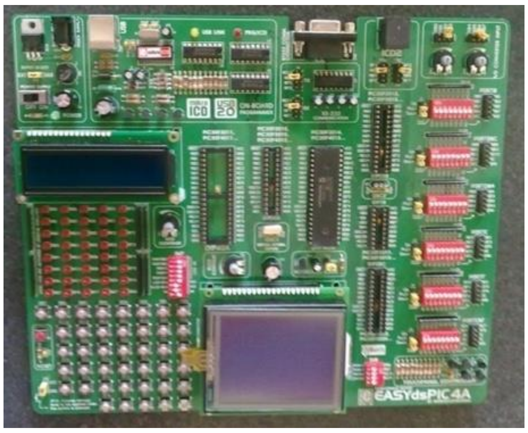


Figure 11: XDM development board

An experimental model on which most of the algorithms' functioning could be verified was obtained in the form of the EasydsPIC4A development system available from Mikroelektronika. Along with this system the appropriate software package, MicroC, was also purchased. As the decision was made to use a specific manufacturer's MCUs throughout the project, this software package was selected and could be used for all development stages in this project. In addition, this package allows programming and in circuit debugging (although hardware dependent).

This hardware features many of the characteristics of the final hardware like a LCD, a 16-bit DSP controller, and an in-circuit debugger.

Apart from the different model Microcontroller (it is of the same family though), it does differ in several very important ways from the ADM prototypes:

1. It cannot capture external analogue signals and therefore cannot record a signal.
2. It only uses on-board RAM – and much less of it, 2kB as opposed to 32kB – rather than both on board and asynchronous RAM to store the data.

3. It has an on-board In Circuit Debugger (ICD) rather than an external one (relevant for development purposes)

4. Uses a USB power supply, rather than a battery.

An explanation of the layout of this board is provided in Appendix A.

2.2.2 Limitations

There were several limitations which prevented the XDM to be used as a fully functional test bed for the project and thus why it could only be used to verify the working of some algorithms, as will be elaborated on soon.

2.2.2.1 ADC functionality

A limitation preventing the XDM from being more useful were the ADC inputs of the board being directly connected to potentiometers (used for testing functionality and basic operation). This severely limited the use of these channels because the potentiometers were not controllable enough to test any form of frequency based technique.

Therefore no usable external data could be generated with the ADC for use in testing the algorithms, even though the ADC operation (from a software writing perspective) could be evaluated.

2.2.2.2 Memory space

The XDM had only the on-board 2kB of on-board RAM memory at its disposal. This memory had to be shared between the data gathered with the ADC as well as program variables. With less than 2kB to work with, the amount of data gathered would not have been sufficient for thorough testing, but it was useful in at least testing the algorithms with generated test data.

2.3 Advanced development model – Alpha prototype

2.3.1 Memory space required for vibration data

A brief study of the general form of algorithms used to compute a FFT of a set of data reveals that it is very common to compute a FFT in-place. That means that the time domain of the data, commonly in the form of a vector of floating point values, is overwritten with the frequency domain values as the algorithm proceeds. However, frequency domain data in its raw form is complex data.

The C compiler cannot directly handle complex data. So it is necessary to leave some memory space for the complex data. An example is given on how the input (real) data is arranged to facilitate the (complex) result of the FFT operation. Elements in the illustration below represents a memory location containing a floating point number, but each adjacent pair jointly forms an imaginary number (blue and white fill), one memory location for the real part and another for the imaginary part. Therefore, a real data string would look as follows:

-0.3	0.0	0.1	0.0	-0.1	0.0	0.5	0.0	-0.3	0.0	0.1	0.0	-0.1	0.0	0.5	0.0

Represents the data string: -0.3, 0.1, -0.1, 0.5, -0.3, 0.1, -0.1, 0.5. Looked at differently, it represents the *imaginary* data string: -0.3+0.0*i*, 0.1+0.0*i*, -0.1+0.0*i*, 0.5+0.0*i*, -0.3+0.0*i*, 0.1+0.0*i*, -0.1+0.0*i*, 0.5+0.0*i*. The Fourier transform of this data would then be calculated as:

-0.4	0.0	0.0	0.0	-0.4	0.8	0.0	0.0	-2.0	0.0	0.0	-0.4	-0.8	0.0	0.0	0.0

This would represent the data string: -0.4+0.0*i*, 0.0+0.0*i*, -0.4+0.8*i*, 0.0+0.0*i*, -2.0+0.0*i*, 0.0-4.0*i*, -0.8+0.0*i*, 0.0+0.0*i*, which would be the Fourier transform of the previous data string. As can be seen, the input data is stored only in every odd element of a memory location (albeit possibly in the form of a data vector).

Consequently, storing 16 samples of data would require 32 memory locations to leave space for the imaginary components of the results. One memory location for the actual data point and another "place holder" memory location reserved for the imaginary part of the eventual Fourier transform result.

Using this information and the information in Table 4, one can calculate the number of samples that the hardware is capable of taking.

Table 4: Alpha prototype ADC parameters

Parameter	**Value**
ADC Resolution	16-bit
Sampling Frequency	15.625 kHz
Sampling time	2 seconds
Double precision floating point memory space required	32 bits (4 bytes)
Asynchronous memory capacity	4096 kB

Keeping in mind the discussion of how complex values are treated (two storage locations for an imaginary number), in that a data point uses two memory locations, then a single data point would require 64 bits.

Therefore, one would be able to store:

$$\text{Samples} = \frac{\text{Total storage}}{\text{storage per sample}} \quad \text{[Eq. 21]}$$

$$= \frac{512\text{K} \times 8}{2 \times 32}$$

$$= 65\ 536 \text{ samples}$$

This would imply that at a sampling frequency of 15.625 kHz, one would have approximately 4 seconds of data (sampling frequency in Hertz):

$$\text{Time} = \frac{\text{Number of samples}}{\text{sampling frequency}} \quad \text{[Eq. 22]}$$

$$= \frac{65\ 536}{15625}$$

$$= 4.2\text{s}$$

As a matter of fact, 32 768 samples would eventually be used, as unfortunately a defective memory chip was being used which exhibited instability when writing significantly more data. This does waste a considerable amount of memory and frequency resolution for this hardware, but it was replaced with a working unit in the Beta prototype of the ADM. For the Alpha prototype however, a sampling time of 2.1 seconds was used.

2.3.2 Spectrum characteristics

2.3.2.1 Lowest frequency component measureable

According to Rayleigh's criterion, this would yield the lowest frequency measureable as:

$$\text{f} = \frac{1}{\Delta \text{T}} \qquad \text{[Eq. 23]}$$

$$= \frac{1}{2.097}$$

$$= 0.48\text{Hz}$$

This is well below the smallest frequency observed (7.4Hz; BSF on the output shaft of a type 4 gearbox, running at 24.5Hz input shaft frequency).

2.3.2.2 Frequency spectrum resolution

A 15-bit FFT (32 768points) will yield 16 384 positive discrete frequency bins (due to the double sided nature of FFTs).

With a Nyquist frequency of 7.8125 kHz, the following frequency resolution will be obtained:

$$\text{Resolution} = \frac{7812.5}{16384} = 0.48\text{Hz/bin} \qquad \text{[Eq. 24]}$$

While investigating the characteristics of the gearboxes, it was found that the closest spacing of bearing frequencies at the operating speeds (input of roughly 1480RPM) was 1.2 Hz. This was found in the output shaft of the type 6 gearbox.

Therefore the frequency resolution of 0.48 Hz/bin was deemed to be admittedly very coarse, but sufficient (2.5 times finer than the smallest difference in frequencies) for the Alpha prototype hardware to capture enough detail.

2.3.3 Architecture

The basic layout of the ADM consists of a core MCU (the DSP variety was selected for this application). This is connected to:

1. LCD screen
2. Keypad
3. Asynchronous memory
4. External ADC (anti-aliasing filter)
5. External programmer and ICD (for debugging purposes)

Various software routines and drivers had to be written to control these components from within the MCU. During the feasibility study, the LCD and keypad were not connected due to circuit board design problems, though these components are not necessary for evaluation purposes. However, in the Beta prototype these design problems was rectified.

2.3.3.1 MCU

As mentioned in the implementation study of chapter one, a DSP type MCU was chosen because of it is optimised for numerical computations, particularly multiplication.

Therefore, the dsPIC33FJ510MC710 was chosen. It is a microchip product and thus seamlessly compatible with the compiler and programmer. The specific variant was chosen due to its high pin count (100 in total, including: IO pins, programmer ports, external device ports, power lines and oscillator pins etc).

In addition, it is a low power device as well as offering very high code security.

2.3.3.2 LCD and keypad

The LCD is an 8 bit parallel unit of resolution 124 × 64 manufactured by Emerging Displays. The drivers for this device were provided by Periseo (a Pretoria firm that was consulted during the project).

The keypad is 4 × 4 push button unit and was purchased of the shelf from Mikroelektronika, the same company that supplied the programmer and compiler. Therefore, it was chosen because the compiler had preinstalled software libraries to control its functions.

2.3.3.3 Asynchronous memory

The asynchronous memory, manufactured by Cypress, is a high performance CMOS Static RAM chip, organised as 512K words by 8 bits. Due to it being parallel high speed memory, a specific design challenge was to ensure that all the address lines to this device were of approximate equal length. This would ensure that no glitches occurred and that the data was reliable.

Being asynchronous memory, it was ideal because of its very high access time. This is crucial, as the ADC operates at a fairly high output frequency and the memory would need to keep up with storing values obtained from it.

2.3.3.4 Analogue to digital converter

The ADC, another Microchip product, was chosen for its high sampling frequency (up to a maximum of 64 kHz) and resolution (16 bits). It is a dual channel, analogue front end delta-sigma type ADC and includes a Programmable Gain Amplifier (PGA) on each channel. This is especially useful in utilising the full 16-bits of the data as the gain can be adjusted according to the signal strength.

It uses a Serial Peripheral Interface (SPI) to communicate with the MCU for which the compiler has a comprehensive library.

The tested response of the ADC after calibration is illustrated below:

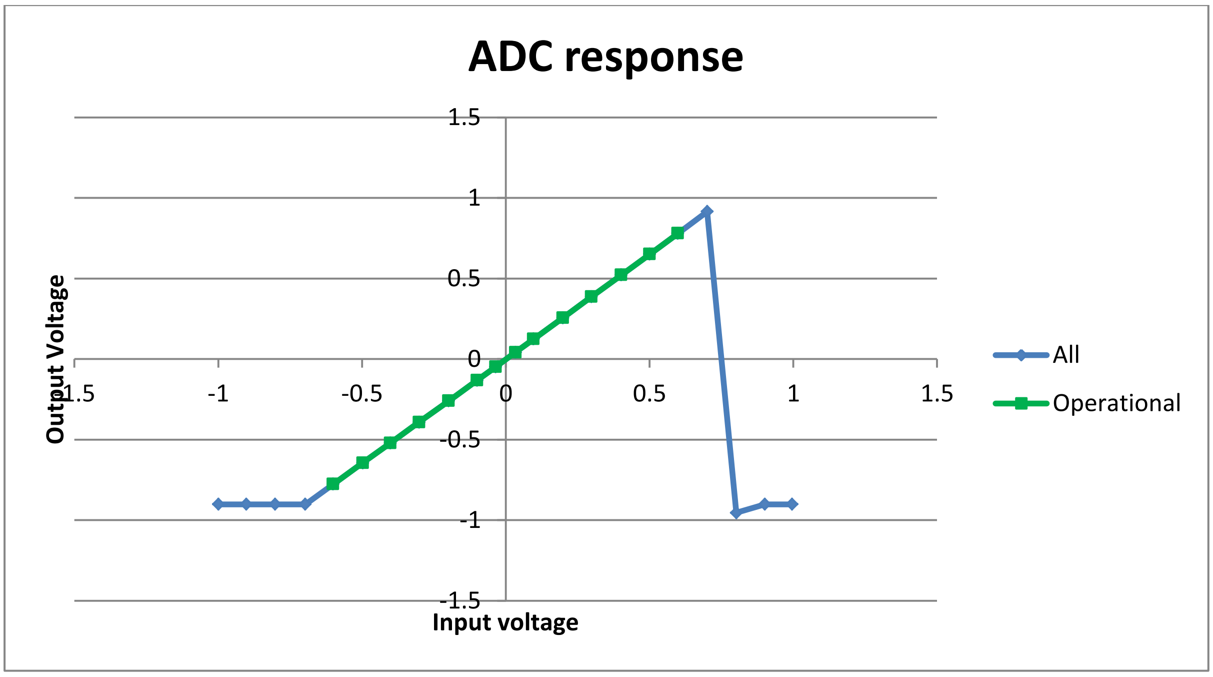


Figure 12: ADC response

The response of the ADC merely involved applying a DC voltage on the channel and noting the output.

A calibration run was performed to see the average offset inherent in the circuit. This value was then stored and subtracted from any following data runs.

It should be noted, that the offset was stable from the very first sample. Therefore, it was not necessary to generate a large number of samples in order to get an accurate offset representation. Nevertheless, for the sake of experimental safety, 20 samples were used and their average taken as the offset in the ADC. The result of this is illustrated in the graph above.

2.3.3.5 External Programmer and ICD

The programmer is used for programming the device. This variant included an ICD which saves a lot of debugging time as it allows one to step through your routine and read the MCU registers real time.

It connects to the circuit board via a 10 pin ribbon cable and to a USB port on a PC on the other end, thus facilitating communication.

2.3.3.6 Anti-aliasing filter

An analogue Butterworth filter, using the Sallen-Key implementation, was used as a low pass filter. It is a fairly low order filter; therefore the cut-off frequency was well below the Nyquist frequency.

It has a cut-off frequency (-3dB) of 2.8 kHz, which is above the specified 2.4 kHz required. This value was derived from the fourth harmonic of the highest frequency of interest, being GMF on the input shaft on the type 4 gearbox. At the Nyquist frequency of 7.8125 kHz the attenuation is approximately -35dB.

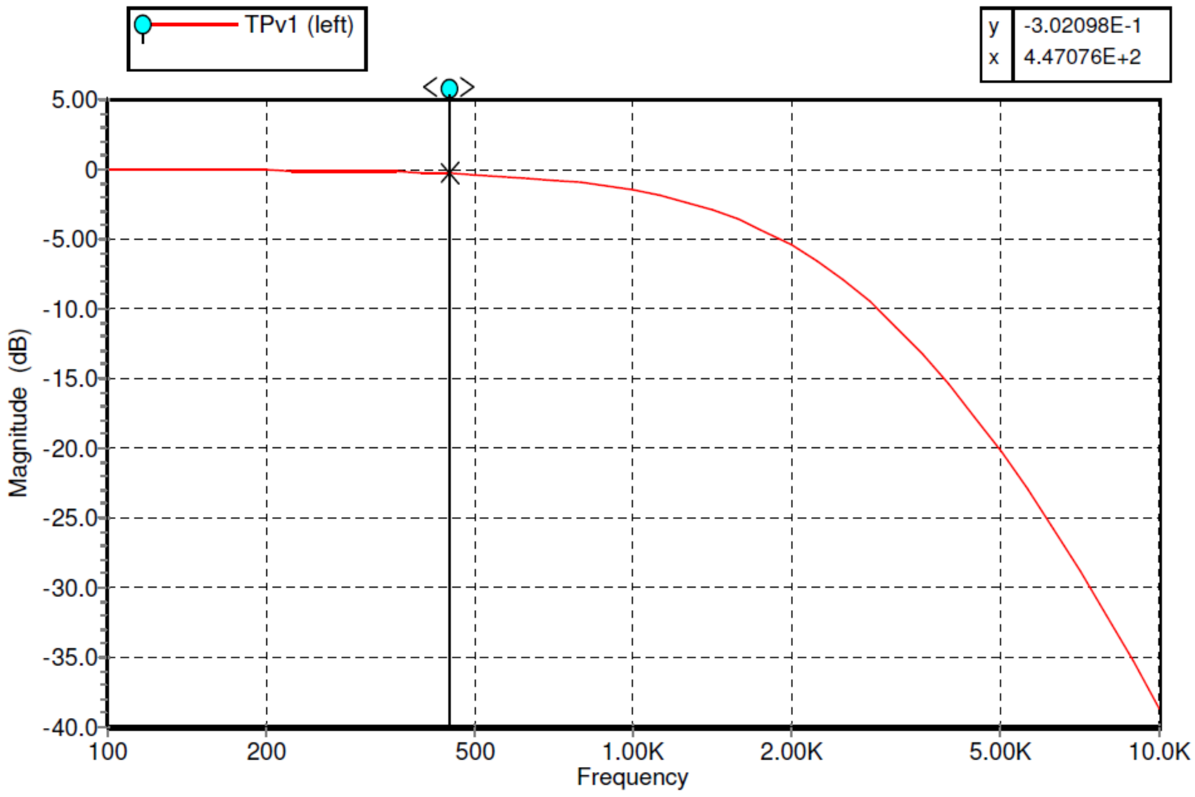


Figure 13: Frequency response of the anti-aliasing filter

This filter was tested by manually connecting a signal generator to the physical hardware input pins of the filter. The output (measured at the physical hardware output pins) was compared the input with an oscilloscope (both the input and output was displayed). This was therefore purely an analogue test of the passive filter circuitry and the active components (MCU, ADC, Memory, etc.) of the device was not tested.

Figure 14 illustrates only the output when a sine sweep was connected to the input. It can be observed that the output signal amplitude remains constant (in time and with frequency) until the increasing signal frequency reaches and surpasses the filter cut-off frequency, whereupon it is visibly attenuated.

Figure 15 illustrates a square of 1 kHz wave passed through the filter. This implies that its harmonics will be attenuated, thereby shaping the signal more to the shape of a sine wave, as can be seen.

Finally, Figure 16 illustrates a signal of 10 kHz, well above the cut-off frequency of the filter, being input and the result again measured after the filter. It is clearly seen how the filter has attenuated the signal amplitude.

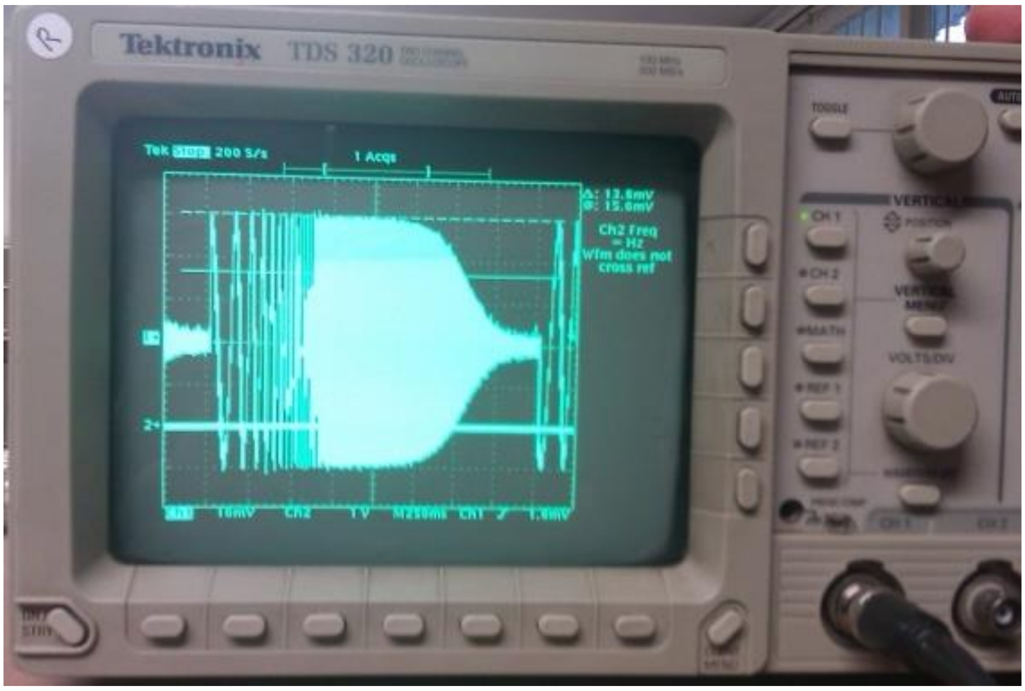


Figure 14: Sine sweep (5Hz to 20 kHz)

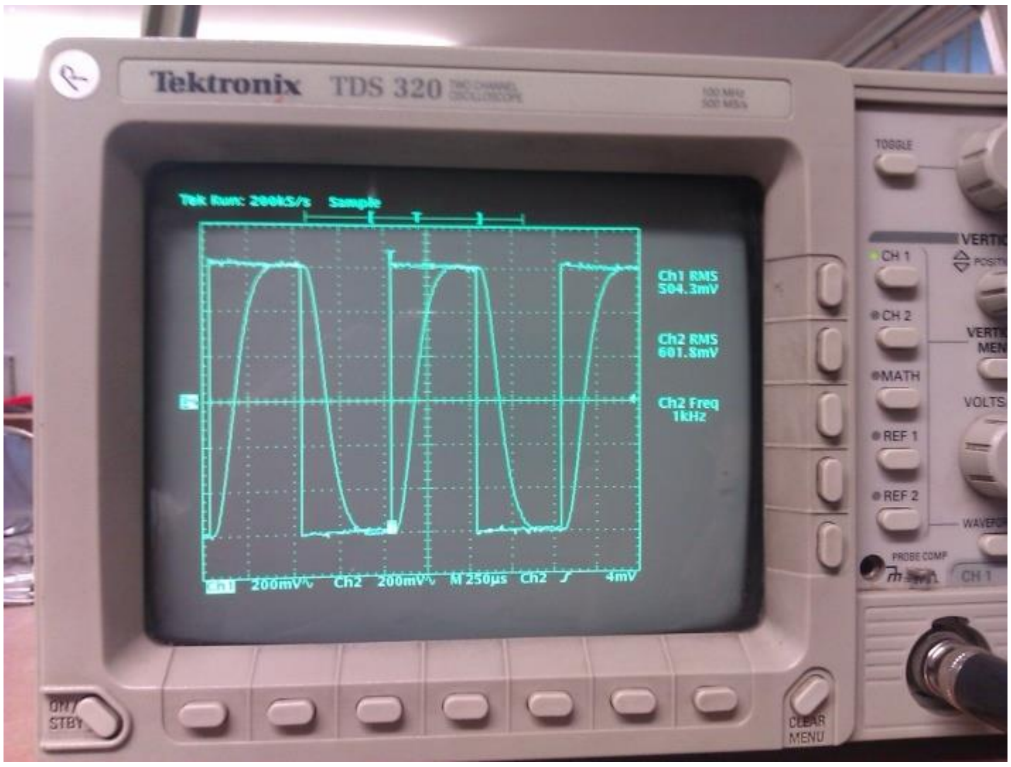


Figure 15: Square wave (overlaid on input) at 1 kHz

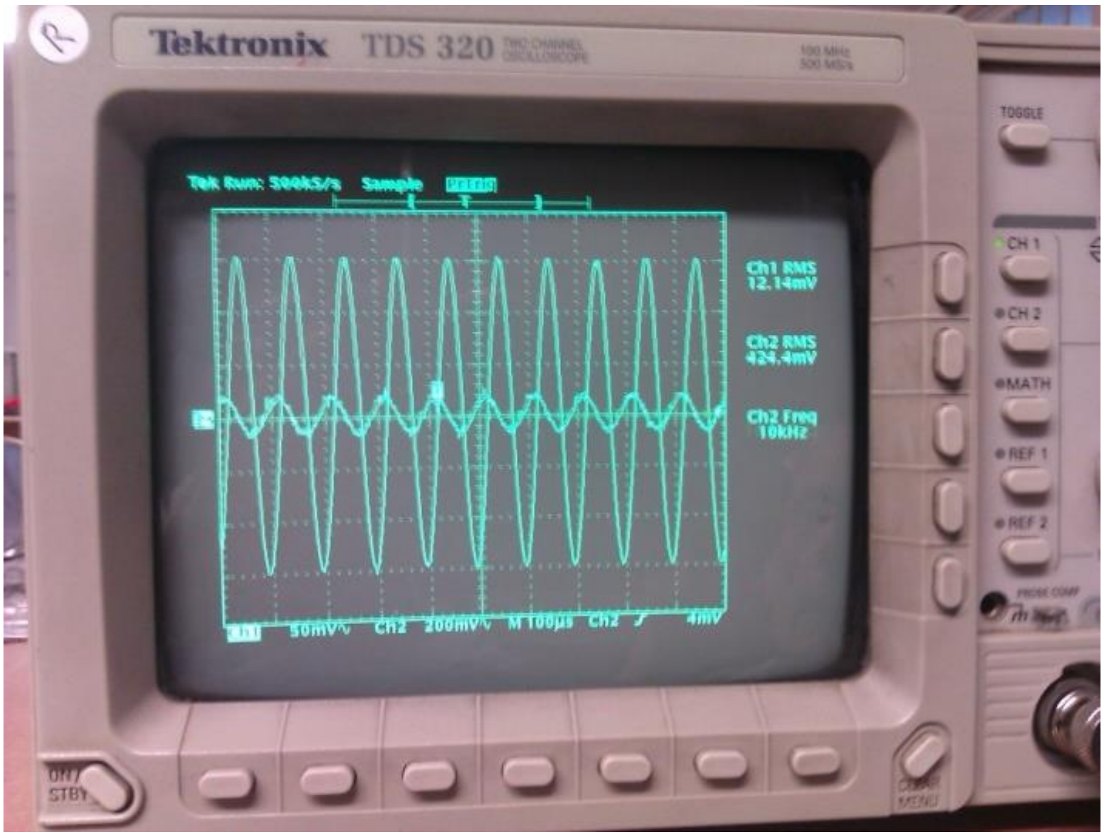


Figure 16: Sine wave at 10 kHz (overlaid with input)

2.3.4 Limitations

2.3.4.1 Data retention

Since the Alpha prototype contained only non-Volatile memory, the data was lost immediately after the device lost power. Coupled to the fact of not being able to download the data to a computer, it was impossible to view the data acquired on a computer. Although the MCU supports this functionality, the design of the circuit board did not allow this.

This shortcoming was duly rectified in the Beta prototype, which was capable of storing the data to non-volatile memory (in the form of and SD-card) as well as transmitting the data via RS-232 connection to a PC.

2.3.4.2 System independence

Unfortunately, due to mistakes made during the design of the circuit board, the Alpha prototype never had the capability to function in a stand-alone capacity. The mistakes related to both the keypad and the LCD screen. Without these components, the device had to be controlled with via a computer.

2.3.4.3 Speed of operation

It was found both during laboratory tests and field tests that the speed of operation was impractically slow – a full analysis of a single measurement would take approximately three minutes.

This implies a total time of twelve minutes for a three stage gearbox (having four shafts). This analysis time includes measurement and the operation of the algorithm (the most taxing operations being detrending, windowing and the Fourier transform).

2.3.4.4 Accelerometer power supply

A hazard was noticed during the field testing regarding the external ICP power supply. Since this power supply was independent of the circuit board power supply, it was found to be very easy to forget to switch it off – resulting in the swift depletion of the three 9V batteries of the unit. Although not a technical problem *per se*, it does have a fairly straight forward technical solution: designing an internal accelerometer power supply as part of the circuit board in the Beta prototype.

2.4 Advanced development model - Beta prototype

What follows is a discussion of the Beta prototype and how it differs from the initial Alpha prototype. The changes were driven by the experien6ces with the Alpha prototype as well as from general optimisation of the design, based on recommendations from our electronics consultant.

See appendix F for illustrations of the beta prototype.

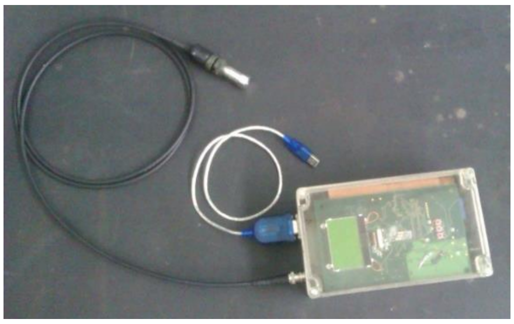


Figure 17: Illustration of the Beta prototype with accelerometer and RS232 cable connected

2.4.1 Clock speed increase

It was found that the Alpha prototype was very slow in its execution speed as the MCU was driven by an external 10 MHz clock, which represents the maximum input for the specific clock type in use.

However, the MCU does have functionality to multiply the clock speed by utilising an on-chip PLL (Phase-Locked Loop) and achieve significantly higher clock speeds (Microchip Technology Inc, 2009). This feature, although available for the Alpha prototype, was not used. This was due to the way in which the Alpha prototype was used: the device was constantly connected to a computer due to the lack of a user interface. This required the system to operate in “Debug mode”. Unfortunately, when using this mode the PLL cannot be activated with the specific compiler in use.

However, as the capacity for the Beta prototype to function independently was added, it was not required anymore to run in “Debug Mode” and the PLL could be activated. This increased the effective clock speed from 10 MHz to 80 MHz.

2.4.2 Addition of external storage and serial communication with a PC

It was found to be a handicap of the system that the gathered data could not directly be shared with a PC for additional computational purposes outside of the on-chip environment.

For this reason, the functionality to send data via a serial RS232 port to a PC was added. This was deemed, at the very least, a useful debugging feature to compare the gathered and computed data on the chip, with commercial data acquisitioning systems, such as the eDAQ and Spider used during this project. In addition, a serial port could serve as an expansion port for future developments such as GPRS transmission and many peripheral applications.

Furthermore, the functionality to store data on an SD card was also added. In this way, the data could be stored on the SD card and saved for later analysis as it is not always possible to have a PC nearby for uploading the data via the serial port. The SD card also served as a "hard drive" as it housed parameter and index files related to the user interface and requirements of the envisaged customer.

2.4.3 Signal acquisitioning and elimination of external ADC

The external ADC was also removed and internal ADC on the MCU was commissioned for data acquisitioning. The reason why an external ADC was used for the Alpha prototype was due to its high resolution: 16 bits. The internal ADC uses only 12 bits, but this deficit was largely recovered by specific use of oversampling, (the theory behind which is explained in section 1.3.3.4.), LCD, push-buttons and a rechargeable battery.

It was mentioned previously that the Alpha prototype had an LCD and push-buttons as well. Unfortunately, the circuit board layout was done incorrectly and these units never functioned. For its purpose, the Alpha prototype could suffice without them, but as the Beta prototype was intended to be near production ready, it was crucial to get these units working.

This was duly accomplished with the Beta prototype, and together with the SD-card formed the user interface, as will be explained in the next section.

As with the Alpha prototype, a battery was used as well. However, with the Beta-prototype, a recharging circuit was added. This enabled the Beta prototype to be recharged from a 220V wall socket, whereas before the Alpha prototype had to be partially dissembled to remove the battery and connected to a pre-configured voltage supply. It was found that a full charge, from empty, takes about 12 hours. However, when used normally during the day, the battery would be fully charged by the next morning when connected overnight to a power outlet.

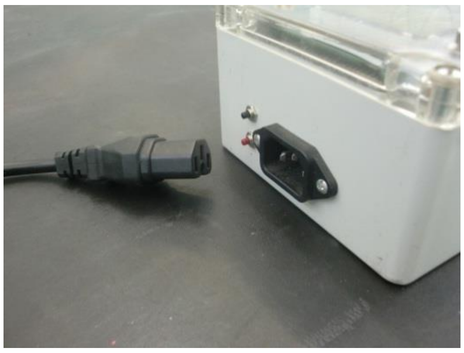


Figure 18: Battery socket and power cord of the Beta prototype

2.4.4 File system

The file system uses a computer-editable SD card and consists of two components:

- The index file (a simple .txt)
- The gearbox folders in which the parameters files resides (also .txt files)

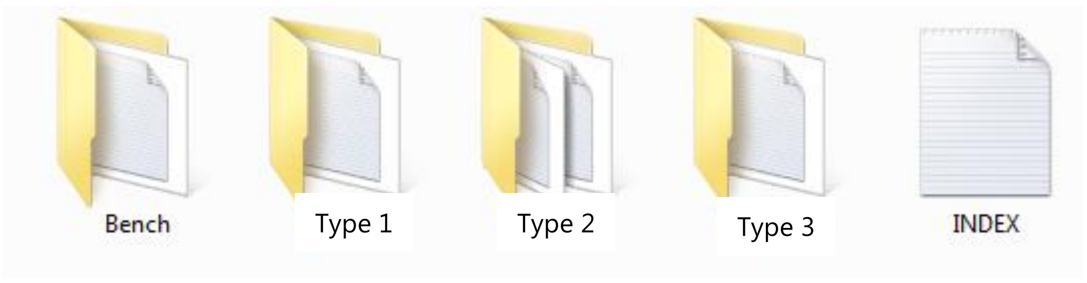
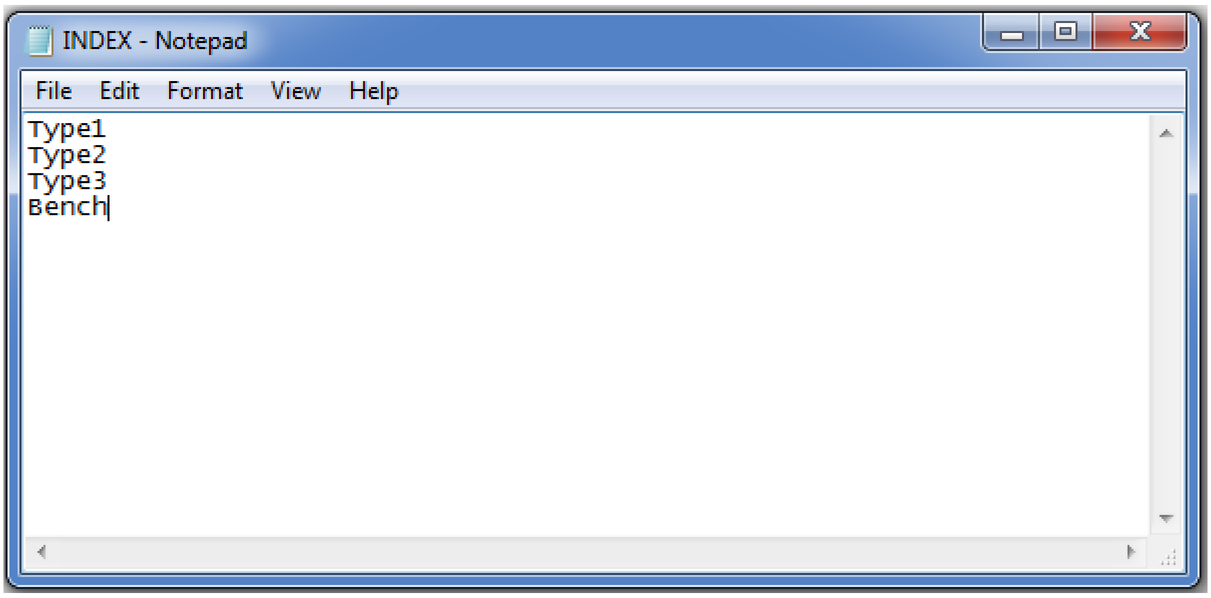


Figure 19: Appropriate contents of the root directory of the SD card for proper functioning

Figure 19 above shows an illustration of how the contents of the SD card look when mounted into a computer. When the Beta prototype boots up, it reads the contents of the file INDEX.txt. This file serves as a means for the device to know what gearboxes are present on the SD card.

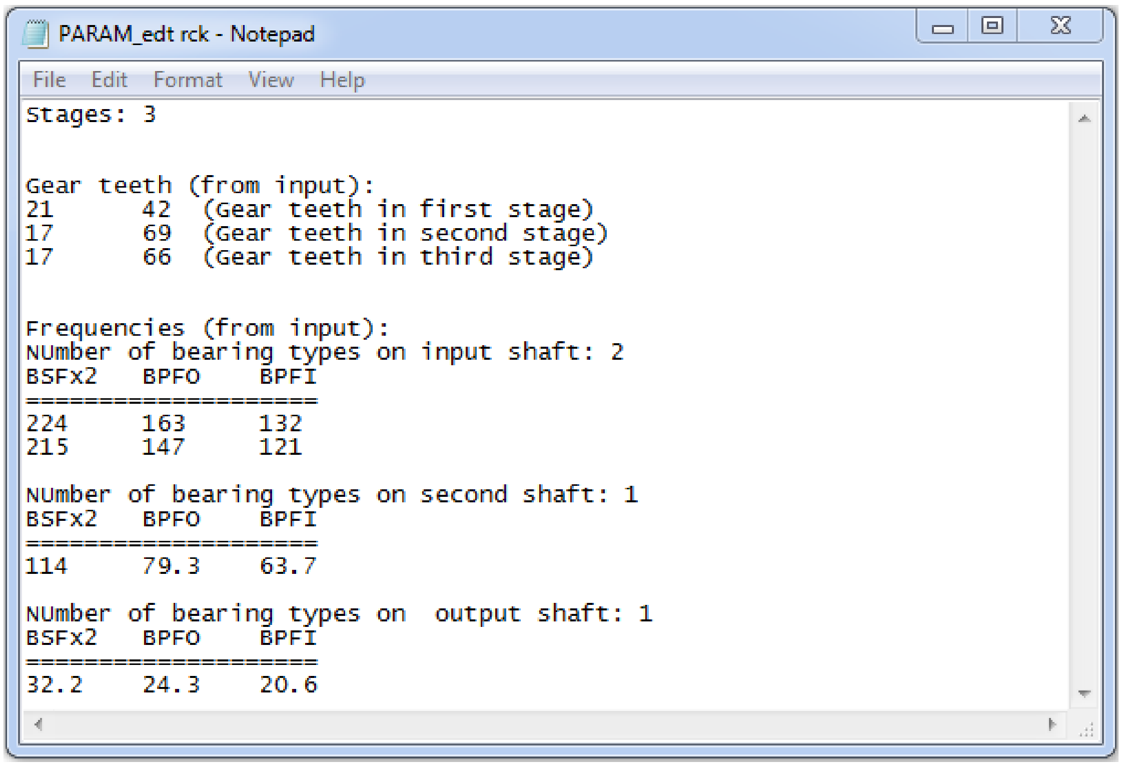
When using the device, the names of the gearboxes in this file are displayed on the screen and the user presses a button to cycle through them. To select a gearbox, the user presses a different button on the device. At that point, the device goes into the folder of the selected gearbox and open the file PARAM.txt. This file contains all the necessary parameters of the gearbox. Figure 21 is an illustration of this file when opened within Windows.

The values within the file PARAM are now be loaded onto the device and starts gathering data and performs the analysis.



```
Type1
Type2
Type3
Bench
```

Figure 20: Contents of the INDEX.txt file



```
Stages: 3


Gear teeth (from input):
21      42  (Gear teeth in first stage)
17      69  (Gear teeth in second stage)
17      66  (Gear teeth in third stage)


Frequencies (from input):
NUmber of bearing types on input shaft: 2
BSFx2   BPFO     BPFI
=====================
224     163      132
215     147      121

NUmber of bearing types on second shaft: 1
BSFx2   BPFO     BPFI
=====================
114     79.3     63.7

NUmber of bearing types on  output shaft: 1
BSFx2   BPFO     BPFI
=====================
32.2    24.3     20.6
```

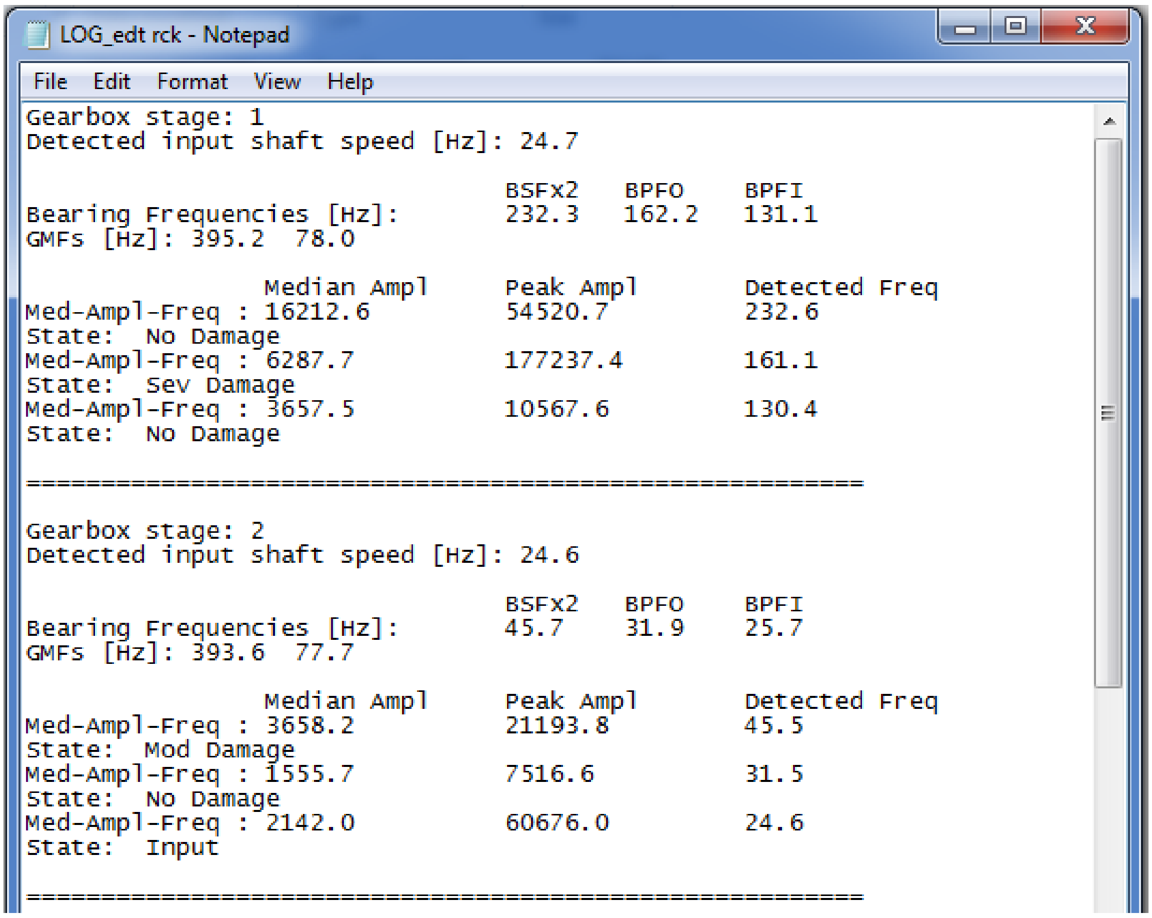
Figure 21: Contents of an example param.txt file illustrating the layout of the file

From then the parameters in the param.txt file are loaded into the working cache memory of the Beta prototype, the following steps occur:

1. Sampling of the data
2. The sampled data is saved to the SD card for later reference
3. The detrending, windowing and FFT algorithms are applied to the data
4. The fault finding algorithm executes
5. The frequency spectrum of the data is saved to the SD card for later reference
6. A log file detailing the findings as well as certain algorithmic parameters are saved, including:
    a. Input shaft frequency detected by the algorithm
    b. Gear mesh frequencies computed
    c. Bearing frequencies computed
    d. Diagnostic information (Figure 22)
        i. Detected bearing frequencies (i.e. as found by the algorithm after computed frequencies were used as a guideline as to where to look)
        ii. Median value of the surrounding noise in that frequency band
        iii. Peak value of the detected bearing frequency

iv. Diagnosis (No damage, Moderate damage, Severe damage)

7. The process repeats for every shaft of the gearbox



```
Gearbox stage: 1
Detected input shaft speed [Hz]: 24.7

                                BSFx2    BPFO     BPFI
Bearing Frequencies [Hz]:       232.3    162.2    131.1
GMFs [Hz]: 395.2  78.0

                Median Ampl     Peak Ampl         Detected Freq
Med-Ampl-Freq : 16212.6         54520.7           232.6
State:  No Damage
Med-Ampl-Freq : 6287.7          177237.4          161.1
State:  Sev Damage
Med-Ampl-Freq : 3657.5          10567.6           130.4
State:  No Damage

========================================================

Gearbox stage: 2
Detected input shaft speed [Hz]: 24.6

                                BSFx2    BPFO     BPFI
Bearing Frequencies [Hz]:       45.7     31.9     25.7
GMFs [Hz]: 393.6  77.7

                Median Ampl     Peak Ampl         Detected Freq
Med-Ampl-Freq : 3658.2          21193.8           45.5
State:  Mod Damage
Med-Ampl-Freq : 1555.7          7516.6            31.5
State:  No Damage
Med-Ampl-Freq : 2142.0          60676.0           24.6
State:  Input

========================================================
```

Figure 22: Illustration of log file (partial window)

The creation of a log file was deemed very valuable for field testing calibration purposes. Using this, one can review the diagnostic information off site. Valuable, because it is often difficult when one is on a noisy site with limited time to spend time judging what the device is trying to tell you. In addition, when the device first enters production, it is anticipated that the first clientele may experience problems and the log file can be used in solving any problems arising as it provides all the diagnostically relevant information.

Additionally, when the device is operational and it indicates a potential problem with the gearbox, the log file can be used by an off-site engineer or technician in conjunction with the time and frequency domain history as recorded by the device. This, in accordance with the requirements for a protection system, allows the analyst to view the history of the device, the machine condition inference that it made, as well as the reasons why it made that inference.

2.4.5 User interface

The device is controlled by using the LCD screen and the pushbuttons. Using these features, the backlight of the device can be toggled, the appropriate gearbox for analysis can be selected and the measurement can be initialised. After each measurement and analysis of every shaft, a brief summary of the state of the bearings present on that shaft is displayed. This process is graphically illustrated in the following figures.

The figure below illustrates the splash screen shown once the device has finished booting and is ready for analysis. Pressing one button toggles the backlight and another moves the routine to the next screen.

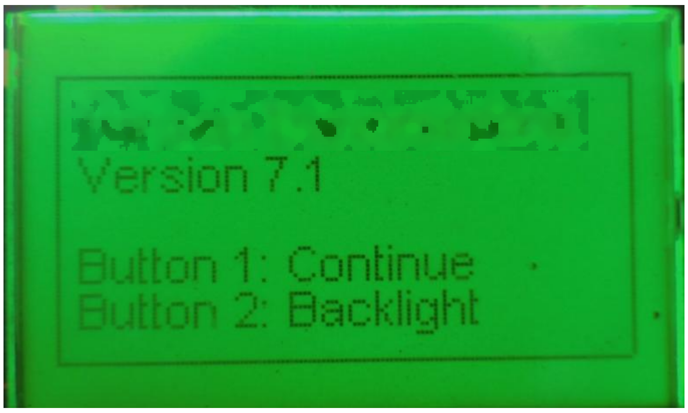


Figure 23: Splash screen displayed after device boot

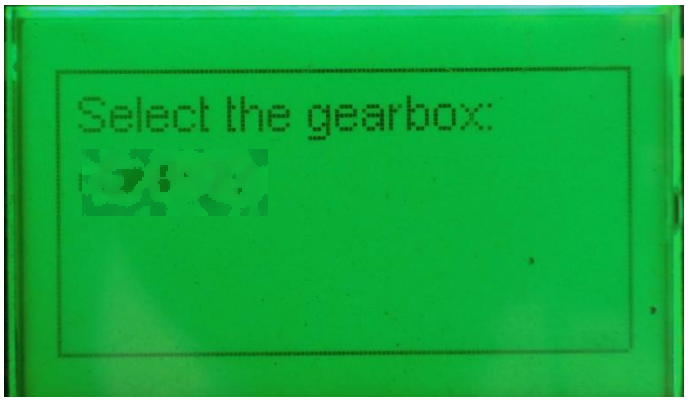


Figure 24: Gearbox selection screen

When arriving at the screen displayed in Figure 24 (by pressing button 1 on the device), the user is prompted to choose the gearbox which is to be analysed. Pressing button 1 will cycle to the next gearbox, while pressing button 2 selects the current gearbox and starts the analysis. If the last gearbox is reached, the lists start over from the beginning.

Once the choice has been made, the device will activate the ADC module and sampling on the first shaft will commence until the required amount of samples have been gathered. Thereafter the ADC unit as well as the screen backlight is shut down in preparation for writing to the SD-Card (all of these units consume a fair amount of power and if not shut down, the device will reset on lower battery levels).

Once sampling has been completed, the Beta prototype writes the time domain data to the SD-card. During this time, the following screen is displayed, which indicates this process. Also displayed on the screen is the maximum time domain value measured.

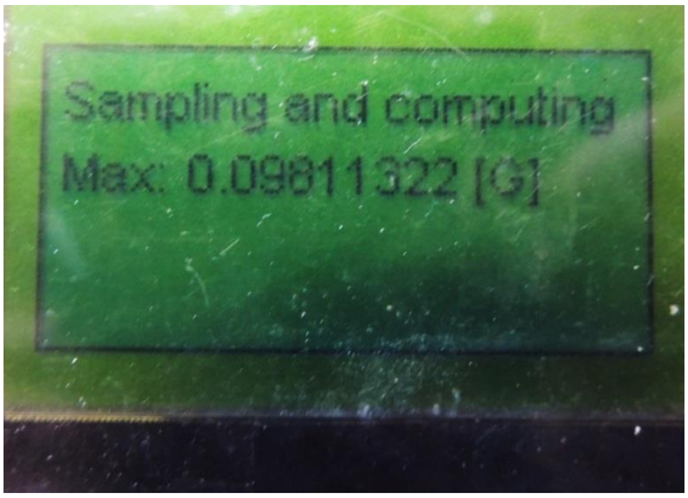
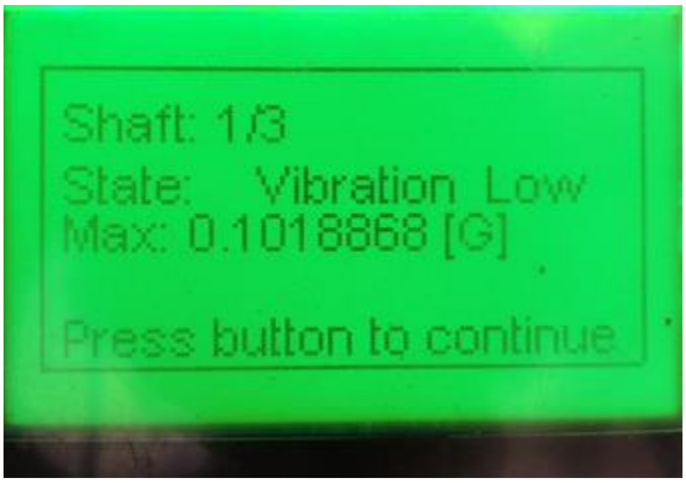

Sampling and computing
Max: 0.09811322 [G]

Figure 25: Progress screen, backlight off

Once the above steps have been completed, the following screen is displayed, proving a summary of the analysis of the measured shaft as well as the maximum value in G of the signal.


Shaft: 1/3
State: Vibration Low
Max: 0.1018868 [G]

Press button to continue

Figure 26: Summary screen

This screen displays the most severe analysis found on the shaft. For example, if a there are two bearings on the shaft then six spectral components will be investigated (three for each bearing). If one or more of these peaks is high enough to trigger a warning, it causes a message to be displayed. Currently, the following warnings are programmed:

Table 5: Screen states corresponding to damage

Incident	Display
Peaks of interest sufficiently low (No damage)	"Vibration Low"
Highest peak >5× median and <10× median (Moderate damage)	"Vibration Moderate"
Highest peak >10× median (High damage)	"Vibration High"

This diagnosis serves as a warning to the operator if an excessively high spectral peak was found and the operator should seek technical assistance.

It is noted that the messages of the analysis in the examples above were very generic in nature, but the following are possibilities (on their own or combined):

- Any message pertaining to the state of the bearings
- Maximum value
- RMS
- Bearing frequencies and peak heights
- Frequency spectrum (not recommended due to low screen resolution)
- Any computed value

When the appropriate button is now pressed, the Beta prototype starts sampling again and the process repeats itself until all the shafts are measured on the gearbox. Of course, the Beta prototype knows how many shafts are present on the gearbox as this information was entered in the parameter file.

3. Algorithm development

3.1 Algorithm premise

As mentioned in section 1.1, it was decided to initially focus on screening the bearings and recommending a course of action based on the findings. The algorithm therefore incorporates a routine that observes the behaviour of the bearings and assesses certain spectral parameters for tell-tale signs of possible damage.

In this regard, it was found in section 1.2.4 that a defect free rolling element bearing does not appear at any of its usual fault frequencies (BPFI, BPFO, 2 × BSF) on a frequency spectrum (the Fundamental Train Frequency – FTF – was not included as it is not often seen in a frequency spectrum a the cage carries very little load, it is more often seen as a modulating signal (Ganeriwala, 2010)), but only generates random noise. The basic premise of the algorithm is therefore based on the fact that if there is a spectral peak at a fault frequency rising above background noise, there is a defect.

The procedure below is explained in sequence as it is performed by the final version of the firmware. The explanation starts at the point where the data has already been captured and resides in the memory of the device, without any prior processing. This explanation thus starts at the beginning of the procedure of data processing.

3.2 Bearing assessment process

3.2.1 Detrending and windowing

The first operation performed on the data is detrending of the data samples. This is done to remove any possible signal drift in the time domain. The possible trend is assumed to be approximately linear.

It is accomplished with a best straight line fit, with the use of the least squares method. A Blackman window is applied to each of the data extracts in order to minimise the effect of leakage. This type of window was chosen for its relatively small computation penalty, good leakage attenuation and good MRB (see section 1.3.3.3).

3.2.2 FFT computation

As previously mentioned, frequency domain techniques were decided upon for use in this project. In the literature, one finds the use of the Power Spectral Density fairly commonplace. It was however deemed less suitable for the application than a normal FFT. This being the case, as a PSD has a tendency to further shrink low amplitude and magnify large ones.

This can be easily conceptualised when considering that a PSD is basically multiplying a Fourier transform with its complex conjugate.

Therefore a standard Cooley-Tuckey algorithm was employed for the FFT computation and the magnitude spectrum used for damage diagnosis.

3.2.3 Input speed

Due to the fact that no speed measurements are available, the algorithm needs to find the speed of the gearbox by using the frequency spectrum (the input speed of the gearbox creates a prominent spectral peak). Bearing this in mind, it was necessary to obtain gearbox input speeds as accurate as possible, due to the compound effect the input speed has on all other frequencies. Indeed, as the algorithm pre-calculates certain frequencies of interest (GMFs, bearing frequencies, shaft frequencies, etc.) based on the input speed, and then looks for spectral peaks at these calculated frequencies; a miscalculated input frequency will cause the algorithm to look for a spectral peak in the wrong place.

In order to understand the methodology which is followed to find the actual shaft frequency, it is first necessary to describe the spectral characteristics created by the motor and gearbox in the vicinity of the fundamental gearbox input shaft frequency. Bear in mind that these characteristics, which will now be summarised, were observed during the study of the gearboxes found at the SASOL plant, implying that the algorithm based on them relies on empirical data:

Due to the fact that a fluid coupling is used to connect the electric motor to the gearbox, a speed differential is created between the motor shaft and gearbox input shaft. This often manifested itself in the frequency domain as two spectral peaks, representing the motor speed and gearbox input speed (with the motor speed having the higher frequency of the two). In most cases, the gearbox input speed was found to have the higher amplitude.

However in a significant number of the observed samples, the motor spectral peaks exhibited a higher amplitude. In the most extreme case observed, the gearbox input shaft peak magnitude was found to be just 13% of the motor peak magnitude. Therefore, the algorithm needed to take this possibility into account when searching for the motor input speed. Importantly, when attempting to identify a low-amplitude gearbox input speed as described, it is necessary to ensure that a stray noise peak is not misidentified as a gearbox input speed, as some noise peaks were observed to have an amplitude of more than 13% of the motor/gearbox shaft. To this end, it was found that if such a low amplitude spectral peak does exist, it was typically at least 20% higher than the surrounding spectral noise. This procedure is illustrated in the following flow-diagram:

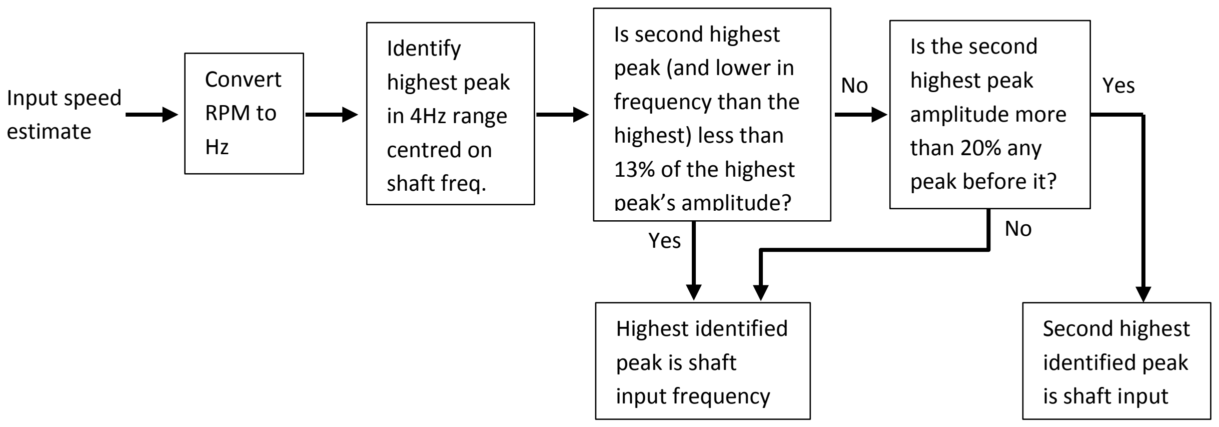

Input speed estimate
Convert RPM to Hz
Identify highest peak in 4Hz range centred on shaft freq.
Is second highest peak (and lower in frequency than the highest) less than 13% of the highest peak's amplitude?
No
Is the second highest peak amplitude more than 20% any peak before it?
Yes
Yes
No
Highest identified peak is shaft input frequency
Second highest identified peak is shaft input

Figure 27: Gearbox input speed methodology work flow

This procedure can be elaborated on as follows:

1. A rough estimate is required of the input speed. A nominal motor speed is sufficient. In the case of the gearboxes under test at SASOL, this approximate speed was 1480 RPM.
2. This converts to approximately 24.7Hz.
3. The algorithm searches for the highest peak within a 4Hz range centred on this frequency range (i.e. between 22.7Hz and 26.7 Hz). The highest peak within this range is then initially taken as the gearbox input speed.
4. In a few cases the shaft speed of the motor has a larger vibration amplitude than the input shaft of the gearbox and the algorithm may therefore wrongly identifies this as the input frequency (see Figure 28). To ascertain this, the second highest peak (of lower frequency – because the motor rotates slightly faster than the gearbox) is identified. If this peak has an amplitude of less than 13.5% (empirical value) of the highest peak, then the highest peak (initially identified) is taken as the gearbox input frequency (Figure 29).

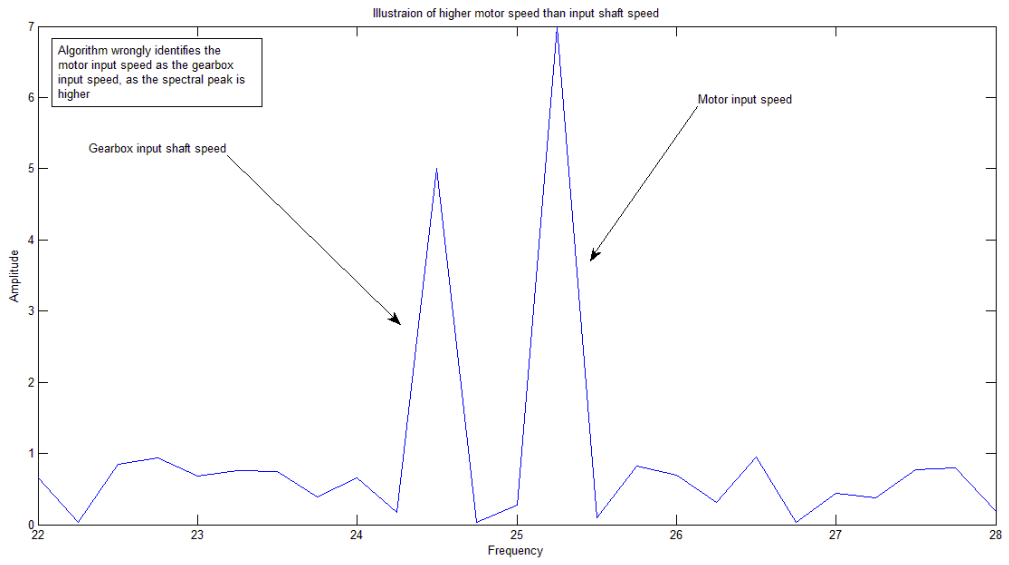


Figure 28: Illustration of a motor frequency having a higher spectral peak than the gearbox input shaft

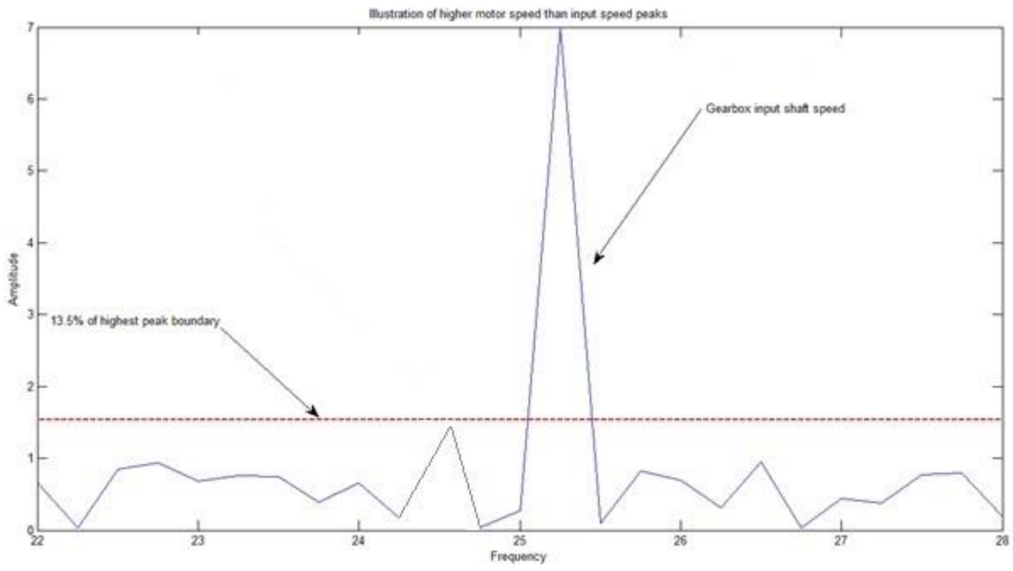


Figure 29: Illustration of the input shaft having a spectral peak more than 13.5% of the second highest peak

5. If the second highest peak has an amplitude of more than 13.5% of the highest peak, there is a probability that it is the input frequency of the gearbox (see Figure 30). However, this peak may well be spectral noise. To test for this, if this peak has a 20% (empirical value) higher value than any peak before it in the frequency range of interest, it – the second highest peak initially identified – is taken to be the gearbox input speed (see Figure 31).

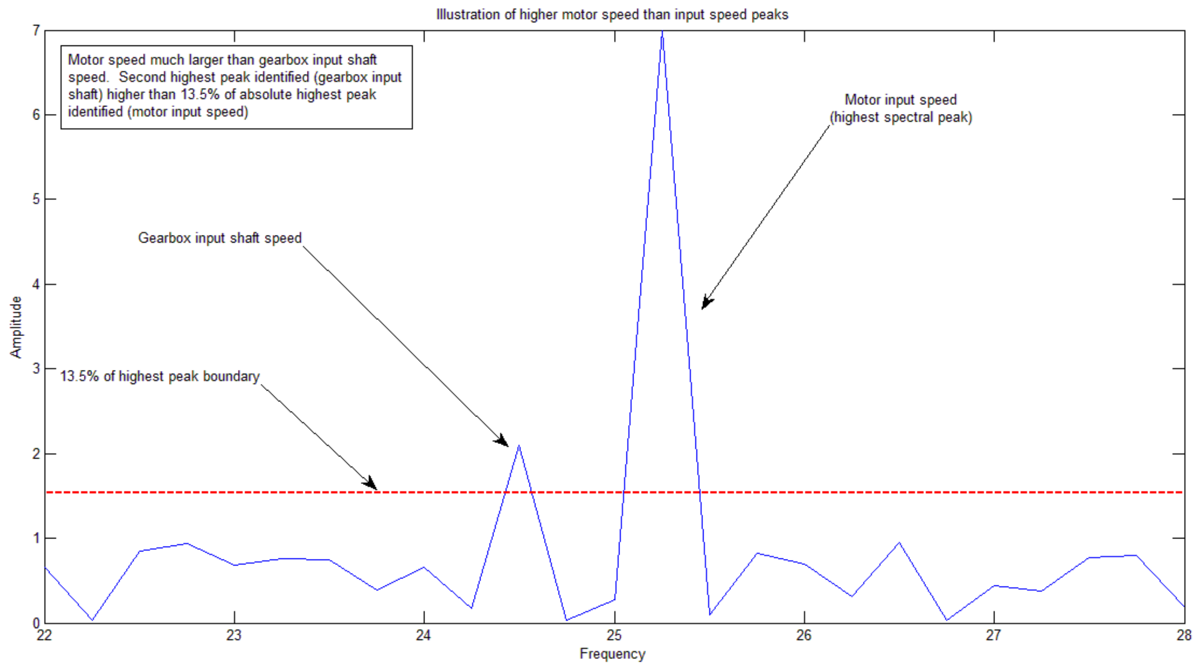


Figure 30: Illustration of a small amplitude peak with a probability of being the gearbox input speed

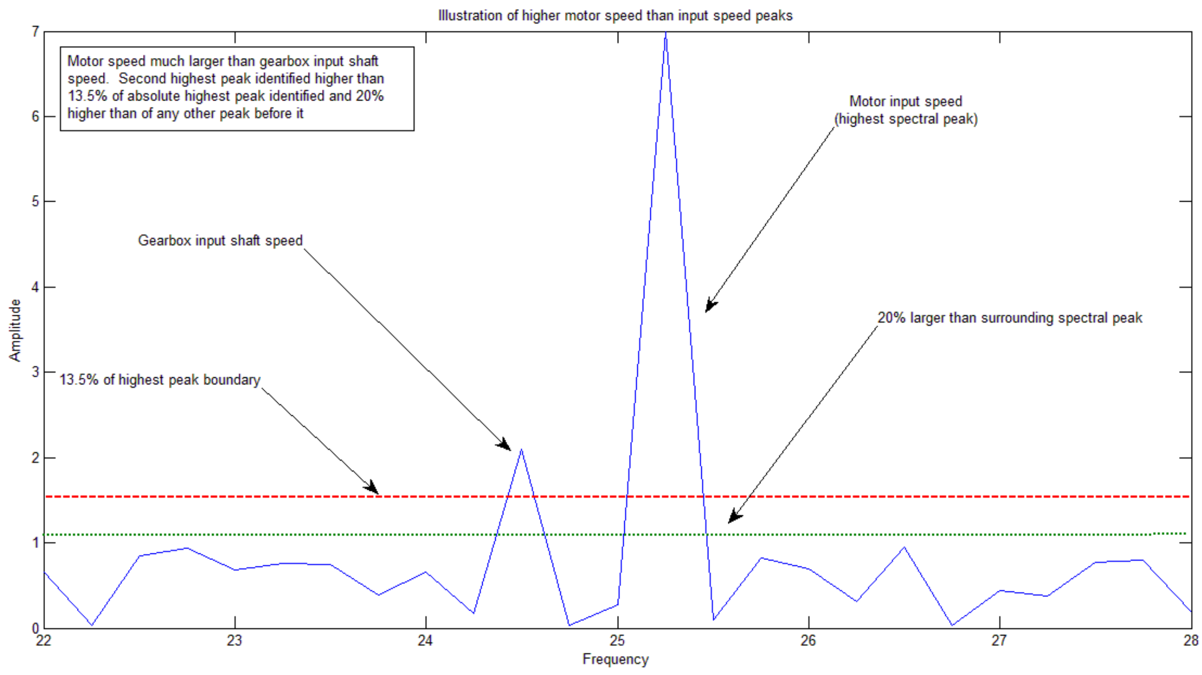


Figure 31: Illustration of an input shaft having a spectral peak higher than 13.5% of the motor input frequency as well being higher than 20% of a peak preceding it

3.2.4 Identifying the relevant spectral peaks

The frequencies of the relevant spectral peaks that can now be calculated are those of the gear mesh frequencies (and harmonics), the harmonics of the input speed and the bearing frequencies. These values are used in the diagnosis of the fault frequencies.

The input speed was calculated in the previous step and its first four harmonics are now calculated and stored.

The GMFs and their harmonics were calculated for each mating gear set. This was done by using the input shaft speed previously calculated and the number of gear teeth on each gear (provided by the industrial partner). By using the supplied gear teeth count on each gear and the input speed, the next shaft speed was calculated and the GMFs applicable to that gear set computed. This process was repeated for each stage in the gearbox until every GMF of that gearbox was computed.

After the bearing part numbers for each shaft on each gearbox were supplied by the manufacturer of the gearboxes, the SKF bearing calculator website (SKF, n.d.) was consulted and the estimated fault frequencies for each bearing were computed (note: only estimated frequencies are computed, the exact frequencies will be found shortly). Importantly these calculations were based on the original estimated input shaft rotation speed (1480 RPM).

To compute the exact frequencies however, one needs to take a look at how these frequencies are computed. Looking at the equation below, it is clear that for a specific bearing design, these frequencies vary linearly with a change in input speed. It is therefore easily possible to algebraically manipulate the SKF computed frequencies so that they reflect the actual frequency instead of the estimated frequencies. This process is illustrated below with the example of the BPFI frequency:

$$\mathrm{BPFI} = \frac{\mathrm{N_b}}{2}\left(1 + \frac{\mathrm{B_d}}{\mathrm{P_d}}\cos\theta\right) \times \mathrm{RPM} \qquad \text{[Eq. 25]}$$

Because variables like the number of rollers, roller geometry and contact angle are generally assumed to be constants in a bearing application, the following can be stated:

$$\mathrm{BPFI} = \mathrm{k} \times \mathrm{RPM} \qquad \text{[Eq. 26]}$$

Therefore, one can derive any new frequency of the same bearing at a new speed as follows:

$$\mathrm{BPFI_2} = \mathrm{BPFI_1} \cdot \left(\frac{\mathrm{RPM_2}}{\mathrm{RPM_1}}\right) \qquad \text{[Eq. 27]}$$

Using this principle, the actual bearing frequencies (and their harmonics) were computed by compensating for the approximation by dividing with the estimated input speed and multiplying by the actual input speed.

3.2.5 Bearing state assessment

Once the frequency spectrum has been generated, the shaft speed computed and the bearing frequencies are computed, and a survey on the state of the bearings can be attempted. The bearing is assessed by considering the amplitudes at its fault frequencies. The algorithm operates on the premise that if a high enough spectral peak at one of these frequencies is present, then the bearing vibrations are assessed to be anomalous and the operator need to warned of potential damage. The relative height of the spectral peak above the noise is used as the assessment parameter that dictates the type of alert that is communicated to the user of the device.

To this end, the first procedure to be done is to find the index of the frequency. At this point a problem arises again with the discrete nature of the frequency spectrum. The computed frequency of the bearing anomalies usually lies between two discrete values of the frequency vector. The value nearest the computed frequency is then used henceforth.

Due to this frequency still being only an approximation (because any variation in contact angle, any slippage of either rings or an inexact shaft input speed may cause the computed bearing frequency to deviate from the actual bearing frequency) a search for a peak in the general vicinity of this frequency is undertaken (+/- 1 Hz in either direction of the computed frequency – see section 3.2.8 for further information). The highest peak within in this 2 Hz boundary (with the computed frequency as the centre frequency) is then taken to be the bearing fault frequency and the amplitude of the signal at that frequency is noted.

The fault frequency's amplitude now compared to the median of the surrounding frequency components (the median of all the frequency components 5Hz either side of the identified fault frequency). When the amplitude of the fault frequency (as per the previous paragraph) is between or above certain boundary values (multiples of the median value – see section 3.2.6) an assessment is made, with a relevant message communicated to the user.

3.2.6 Statistical parameter (median) search boundaries

If a spectral peak is detected in the search boundaries, it is compared to the median of the surrounding frequencies, as explained in the previous section. The reasoning being that if a peak stands out significantly above the surrounding median of the noise, it is likely a definite signal originating from the gearbox.

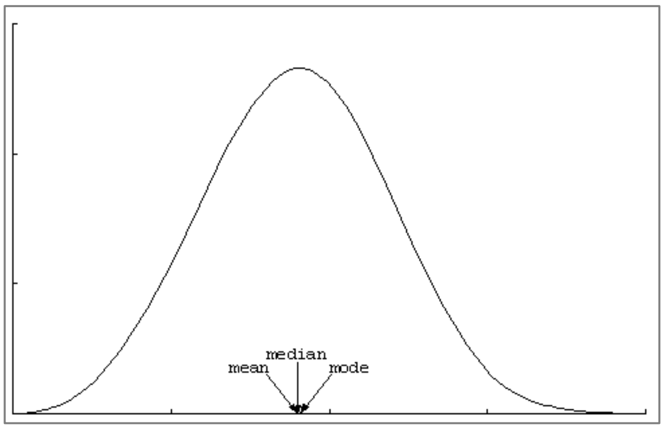


Figure 32: The bell curve, illustrating the relationship between mean, median and mode (not applicable) in a Gaussian distribution (von Hippel, 2005)

As mentioned, the median of the surrounding noise is used instead of the mean. This is simply because the median often gives a more accurate representation of the noise level because a stray peak of a nearby signal component would raise the mean to a level unrepresentative of the noise levels. The median, being merely the middle value in a sample (with an equal number of samples being larger and lower than it), often gives a much more representative figure of the noise level and was not as badly affected by stray peaks.

In addition, the median and mean of Gaussian noise is exactly the same, so in the case of no stray peaks, the answer would be unaffected (von Hippel, 2005). See Figure 32.

For the purposes of this project, if a fundamental spectral peak of a bearing damage frequency was found having an amplitude of 5 times the median value of the surrounding noise, a 'Vibration Caution' message is communicated to the user. An amplitude such as this was considered significantly high enough above the background noise to cause some concern as a bearing fault might be present. This is due to the fact that a bearing in good condition is accepted to emit random noise, as found previously. In this case, the user is advised to contact specialist help in the form of consultants, on site specialists or the OEM as a precautionary measure.

In a more severe case, if a peak with an amplitude of more than 10 times the median was found, a 'Vibration Warning' message is communicated to the user. A peak, as distinguished from the background noise as this, increases the probability that a defect may exist to the point that a warning is given to the user. In this case, the user should seek professional help from specialists or the OEM as a matter of urgency.

Of course, when a peak value of less than 5 times the median value is detected, there is assumed to be a minimal risk of bearing damage. In this case, a simple message of 'Vibration normal' is communicated to the user.

The values of 5× and 10× are considered convenient ballpark figures based on known states of the measured gearboxes. A larger measurement campaign using the Beta prototype to various other measurement venues (in several different industrial markets) will yield significantly more data to be used for calibration purposes to find more exact thresholds. See the section 7.4.5.

As the method used will however likely use spectral analysis, this method was considered sufficient for the purpose of this project, which is to build a working prototype of a system.

3.2.7 Exclusion criteria

If another frequency coincides with a bearing frequency, it is impossible to make a judgement using the current technique and a false assessment will result. For that reason, the computed fault frequencies are checked against other known peaks (GMFs and their harmonics and input shaft speed and its harmonics) in the frequency spectrum before a diagnosis is made. If another frequency coincides with a fault frequency, the user of the device would be notified that an interfering signal is present and notified what the source of the signal is.

Although this currently prevents a diagnosis from being made, further work is intended to remedy this and several additional methods are being considered. See the section 7.4.6 "Recommendations – Signal interference compensation".

3.2.8 Fault frequency search boundaries

As stated, the search routine takes the computed frequency and finds an element in the frequency vector that is the nearest match for this frequency and from this value as the centre of the search boundaries, approximately 1 Hz either way is searched for the fault frequencies by looking for the highest peak in that part of the spectrum.

The boundaries will not be exactly ± 1Hz, though. The reason for this is now explained. Reviewing the coding rationale in the section 3.2.5, it can be seen that the search boundaries are defined by multiplying the upper and lower limits of the search boundaries (±1Hz, in this case) with a factor that converts this frequency boundaries into magnitude- and frequency vector element boundaries when working within the software. This is done to tell the program to search only the elements of the vector that corresponds to the search boundaries. However, after this multiplication, the answer is rarely an integer value representing a vector element and is usually a real number and has to be rounded.

In order to slightly increase the search boundaries, the upper boundary is rounded up and the lower boundary is rounded down. Therefore, the search boundaries are slightly more than 1Hz.

To find an appropriate boundary width, a study was done to see what the general difference in frequency was between the original frequency (the centre frequency in the search range) and the found frequency within the search range. Based on what the difference is, and taking some extreme values into account (there were practical limits to how large this range could be), the boundary values were decided upon.

These practical limits were instances like bearing frequencies near each other on the same shaft and the same bearing with fault frequencies near each other. This occurred in the cases summarised in Table 6.

Table 6: Close frequency spacing

Gearbox	Problem	Comments	Value

Type 4 – shaft 3	Two bearings on the same shaft with fault frequencies close to each other	BPFIs: 38.1 and 40.7
		BPFOs: 26.6 and 28.4
			Δf = 2.6 Hz
			Δf = 1.8 Hz

Type 6 – shaft 3	Bearing fault frequencies near each other	BPFI: 13.9127
		BPFO: 11.8737
		2 × BSF: 10.6744
			
			Δf = 2.039
			Δf = **1.1993**

This implies a maximum search boundary extreme value of roughly 1.2 Hz (identified in bold).

The study of the search boundaries revealed that the average difference between the original value (centre frequency) and the final selected value was 0.64 Hz. However, with a standard deviation of 0.45 Hz, this value was not selected as the extreme values of the search boundaries as there was too much variation in the results. Instead, the maximum difference was looked at. This was found to be 1.11 Hz (the extra 0.11Hz outside the search boundary being the result of the rounding, as explained above).

Therefore, it would logically be possible to increase the search boundaries by an additional 0.1 Hz to a total value of 1.2 Hz. However, to maintain a certain margin between frequencies and search boundaries, it was decided to leave the search boundary at ± 1Hz and allowing for an additional 0.1Hz due to the rounding.

4. Laboratory testing

4.1 Overview

Both the Alpha and Beta prototypes were laboratory tested before they commenced field work. However, the nature of their tests differed substantially.

The Alpha prototype underwent a test primarily to verify the interaction between the hardware and algorithm, as the combination had not been tested in its entirety up to that point. It was therefore crucial to verify that all the components, especially the MCU, ADC and memory units integrated well and that the processed data were reliable.

Seeing as the same basic hardware were used in the Beta prototype, verifying this functionality to the extent of the Alpha prototype was unnecessary and could be done with simple shaker tests. This was accompanied by amplitude calibration.

4.2 Alpha prototype laboratory testing

4.2.1 Exploratory test

4.2.1.1 Description

Before the alpha prototype was field tested, it was deemed a good idea to do an exploratory test on the test setup with laboratory equipment. Two test bearing outer races were used (an undamaged one and a damaged one) and the test was to confirm that the seeded damage was picked up correctly by the test equipment. In addition, the tests certified that no other mechanical faults (misalignment, looseness, etc.) or damage would interfere with the test.

The Alpha prototype then applied the algorithm (described in the entirety of section 3.2) to the test data in order to verify it is working on the test setup.

4.2.1.2 Test setup

The data acquisition equipment used during the test was the Spider data logger, an ICP accelerometer (with power supply) and speed sensor (for real time viewing only, not for logging). The Spider was connected to a computer and the measurements saved for post processing whilst the shaft speed was noted.

The test hardware used consisted of the following (illustrated in Figure 33):

Table 7: List of test setup mechanicals

Label	Description	Further details
1	AC Electric motor	0.15 kW
2	Motor speed control	
3	Shaft coupling	Spider type, flexible coupling
4	Gear mesh	1:1 ratio, 72 teeth per gear
5	Support bearings	Deep groove 16006
6	Test bearing	Taper roller - 320/22X

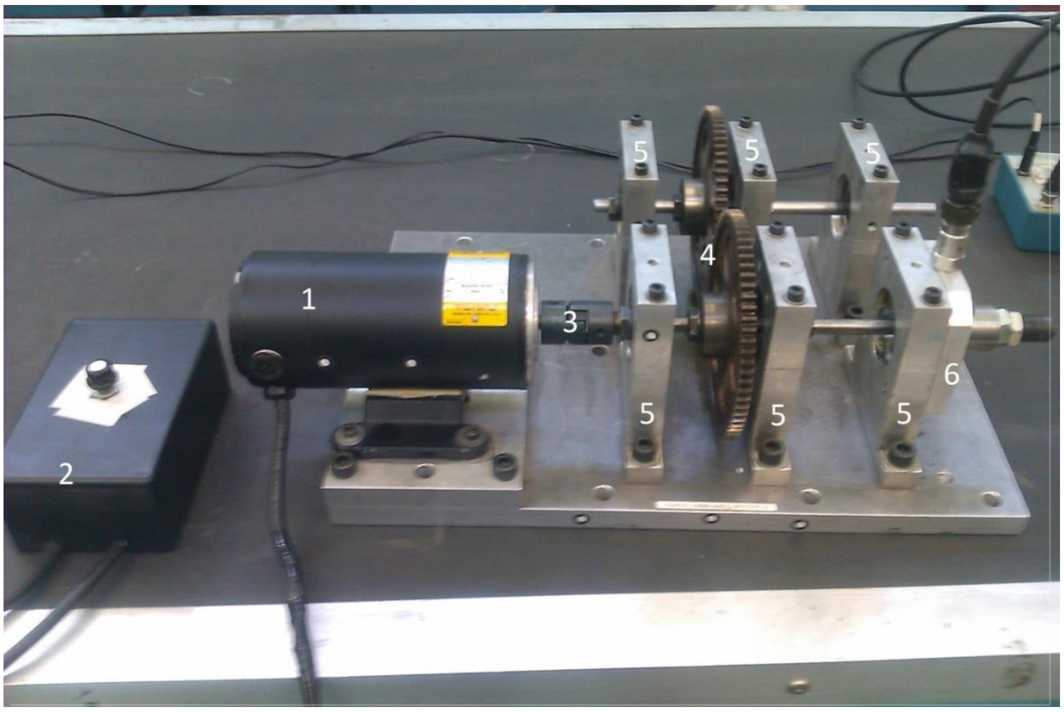


Figure 33: Hardware test setup

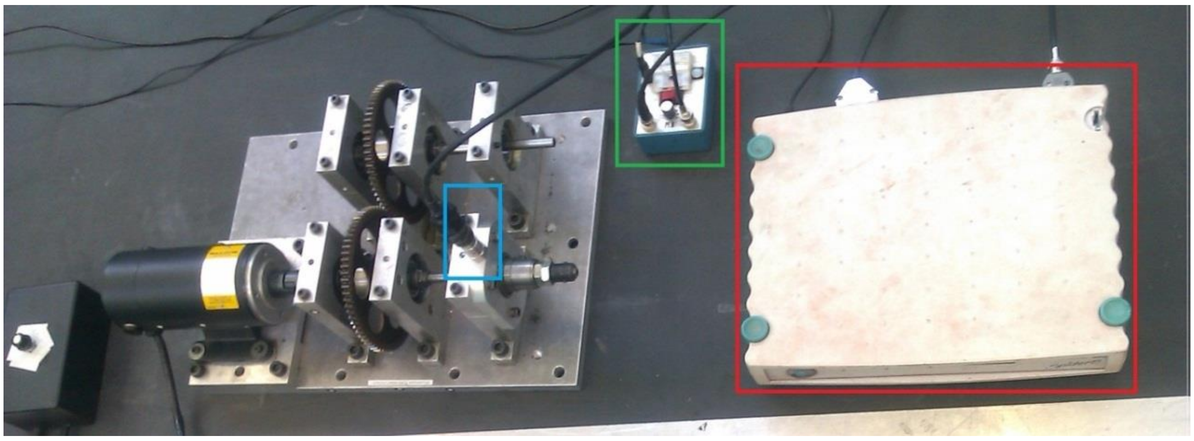


Figure 34: Figure highlighting the data acquisition equipment

Figure 34 key:

Green:	ICP Power supply
Red:	Spider
Blue:	ICP Accelerometer

It consisted of an electric motor with speed control. The power was delivered to the setup via a flexible, spider type, shaft coupling. The primary shaft goes through three support bearings and the test bearing. The shaft also has a gear attached to it that drives a secondary shaft. In addition, a reflective strip was also attached to the shaft (visible on the figure, but not numbered).

The secondary shaft is a by-product of the test bench's previous usage as an unbalance apparatus. In that application unbalance weights were attached to the both shafts (which spun in opposite direction, facilitated by the 1:1 single gear set). Due to this, the secondary shaft has no load on it (apart from the relatively feeble friction torque from its support bearings).

4.2.1.3 Test bearings

Two taper roller (320/22X) test bearings were used and were grease lubricated. One of the bearings were damaged and one remained undamaged (see Figure 35 and Figure 36). In the case of the damaged bearing, a cut was made to the outer race of the bearing.

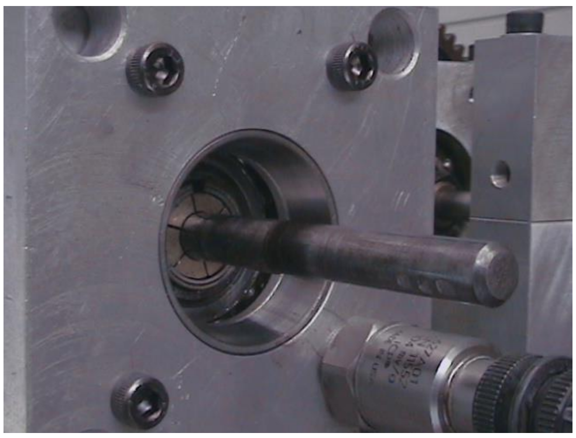


Figure 35: Damaged test bearing

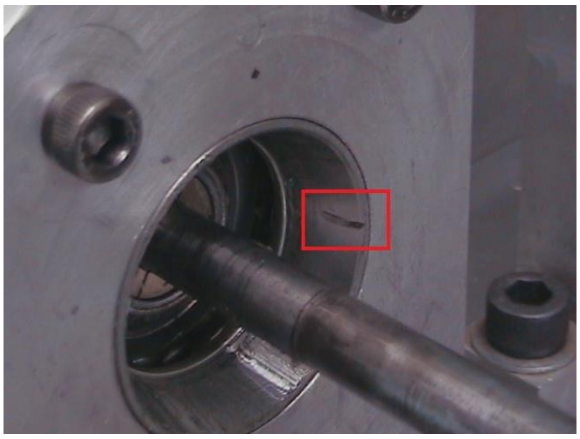


Figure 36: Undamaged test bearing

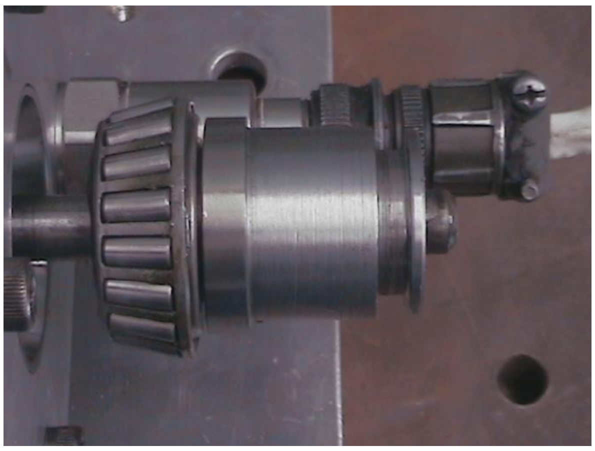


Figure 37: Test bearing inner race bushing assembly

Furthermore, the inner race was press fit onto a bushing assembly which had a 10mm "loose" interference fit drilled into it (diameter 10.02mm) so that it fit snugly onto the 10mm shaft. Not visible in Figure 37 is the part of the bushing onto which the bearing inner race is press fit onto. The bushing consisted of two parts, both of which are visible. The first is the outer part onto which the bearing is press fit, and the second is the inner part (just visible sticking out on the right) which locates it accurately on the shaft.

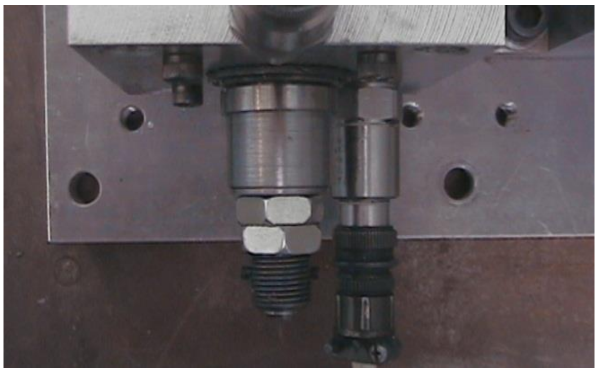


Figure 38: Test bearing load mechanism

An axial load was generated on the bearing under test by a tensioning nut (in conjunction with a locknut). This mechanism required the use of a M16 stud and a 10mm hole drilled axially through it. The hole was drilled on a lathe and was of a loose fitting interference fit (shaft diameter is 10mm and hole diameter is 10.02 mm). In addition, a radial hole of 3mm diameter was drilled into the tensioning bolt in order to accommodate two M3 grub screws (one from either side).

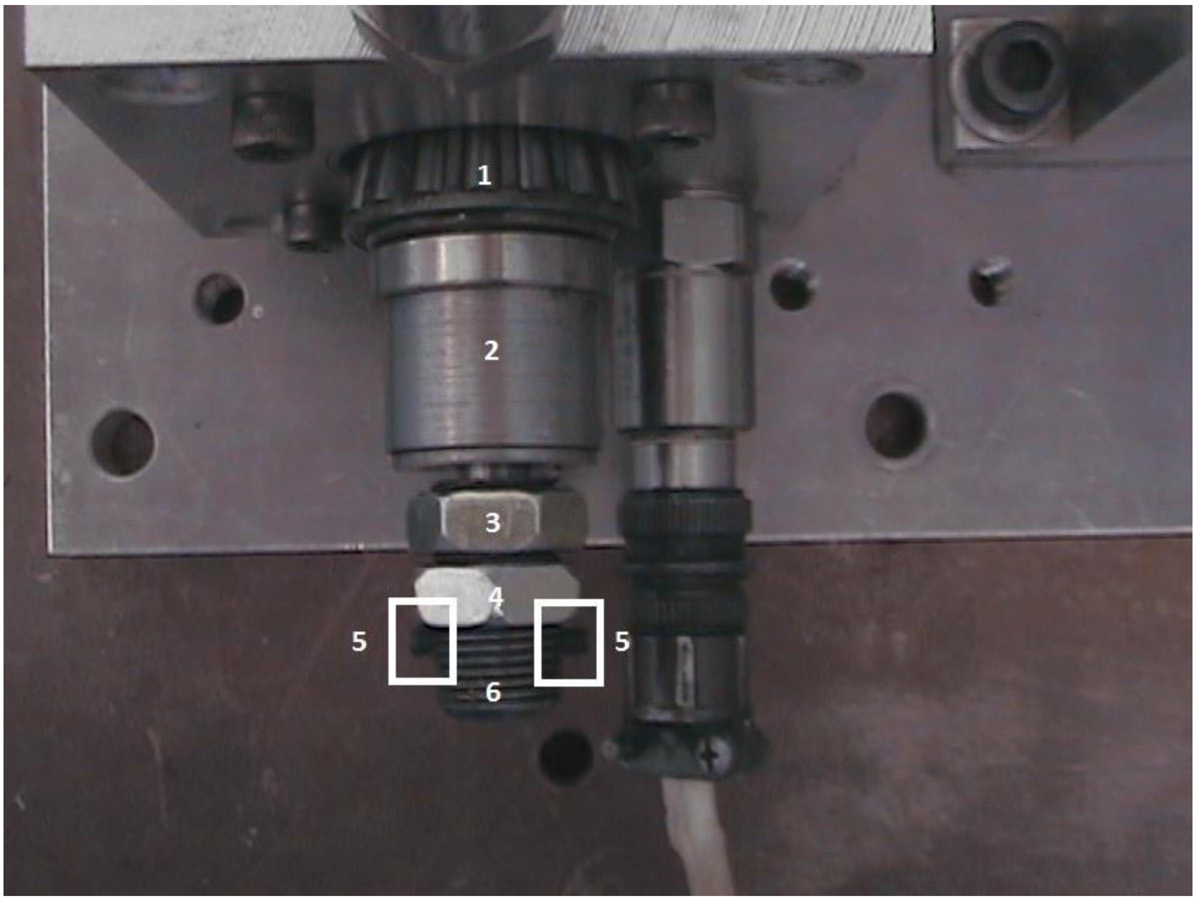


Figure 39: Disassembled view of the load mechanism

In the disassembled view of the load mechanism, one can see the following:

1. Test bearing inner race and cage
2. Load bushing
3. Loading nut
4. Locknut
5. M3 grub screws (within rectangles)
6. M16 stud, axially and radially drilled (M3 and ø10mm, respectively)

The load mechanism worked simply by firstly tightening the grub screws mated to the stud (against indentations made on the shaft) and then tightening the locking nut as required and then locking it in place with the locknut.

4.2.1.4 Diagrammatical illustration

Figure 40 below shows a schematic of the test setup along with some relevant parameters pertinent to the test. The test setup was assembled in the University of Pretoria's SASOL Laboratory for Structural Mechanics. In the figure the layout of the test setup becomes clear and one can understand how the different components work together.

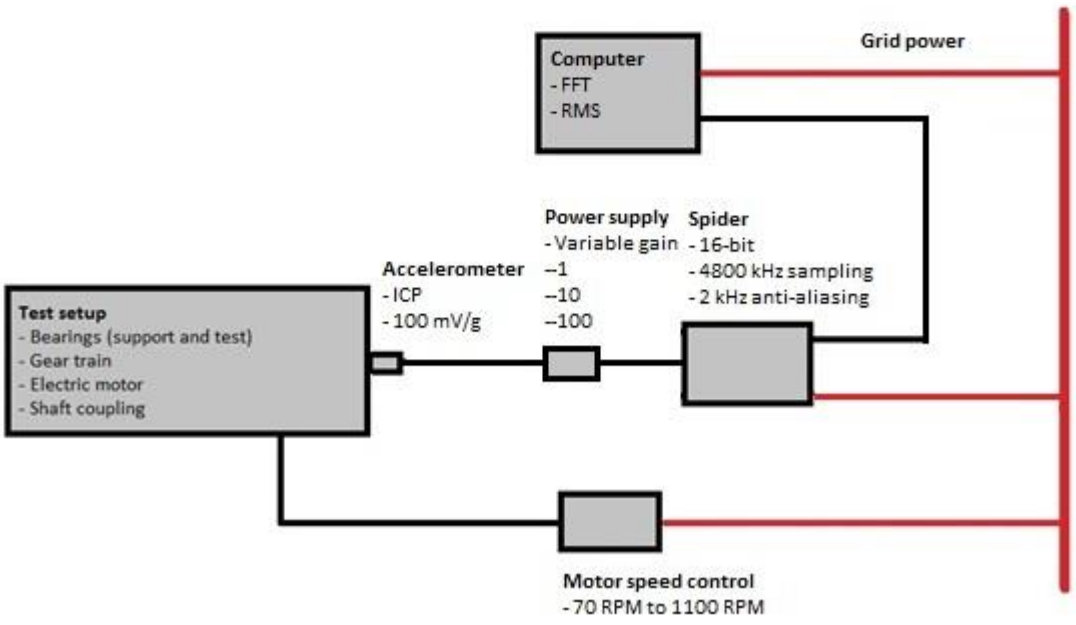


Figure 40: Synoptic diagram of test setup

4.2.1.5 Results

The signal processing parameters of note in these results are as follows:

- 16-bit ADC resolution
- 4.8kHz Sampling frequency
    - 2.4kHz Nyquist frequency
    - 2kHz Anti-aliasing filter
- 8500 samples
- 8192 point double sided FFT
    - 4096 point single sided FFT

It will be noted that the above parameters give a somewhat course spectral resolution of about 0.58Hz. These parameters were chosen deliberately in order to simulate the results that one would expect from the developed hardware, which it can be seen gives a spectral resolution of 0.48Hz as can be seen in Section 2.3.

4.2.1.5.a Comparison: Damaged/Undamaged

Firstly, to evaluate the success of the test setup, a spectral comparison was made between the damaged and undamaged bearings at various speeds; the table below lists the frequencies shaft and speeds that was investigated. The frequencies highlighted in red marks those that were positively identified in the spectra.

The GMFs were never properly distinguished, as there was no load on them and the gears were just freewheeling. In addition, the motor used produced an unvarying 2×line frequency of very high magnitude at a constant 100Hz with a modulation frequency of the shaft speed.

It is also worth noting that as expected, the amplitude of the BPFO component in the spectra increased from about 0.0063g when running at 500 RPM to about 0.023g in the 1000RPM plot (this increasing trend is amplitude is visible throughout the speed range considered).

Table 8: Frequencies (Hz) at the corresponding shaft speed

Input (RPM)	500	600	700	900	1000
Shaft	8.21	9.96	11.72	15.24	16.41
GMF	590.76	717.34	843.84	1083.60	1200.24
Taper – BPFI	80.50	96.60	112.70	145.38	161.00
Taper – BPFO	**60.95**	**73.26**	**84.98**	**109.60**	**121.3**
Taper – 2×BSF	59.00	70.80	82.60	106.55	118.00
D. Groove – BPFI	57.50	69.0	80.50	103.85	115.00
D. Groove – BPFO	42.55	51.06	59.57	76.85	85.10
D. Groove – 2×BSF	54.50	65.40	76.30	98.43	109.00
2×line frequency	**100**	**100**	**100**	**100**	**100**

Two spectra at opposite ends of the speed scale (500RPM and 1000RPM) will be shown below in their damaged and undamaged states.

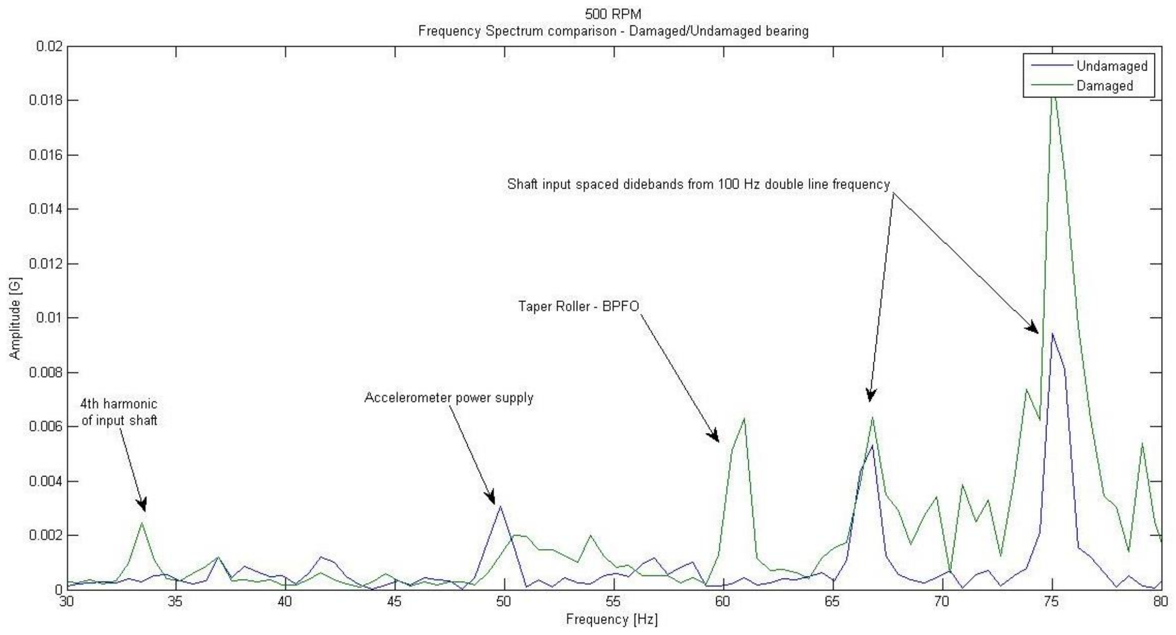


Figure 41: Frequency spectrum at 500RPM input shaft

Several peaks are identified in the frequency spectrum of the data measured at 500 RPM. Peaks identified are the BPFO of the taper roller bearing which is present in the damaged state, but not in the undamaged state. The other strong peaks are sidebands of 2×motor line frequency at a 100Hz (sidebands spaced at motor speed – 8.2Hz in this case). There is also a slightly weaker peak (in the undamaged case) at 50Hz, it was ascertained that this was the accelerometer power supply (at that stage the battery ran out of power and it was connected to the wall socket, at which point that peak appeared). One can also see in the damaged spectrum a peak at about 33Hz corresponding to the 4th harmonic of the input shaft.

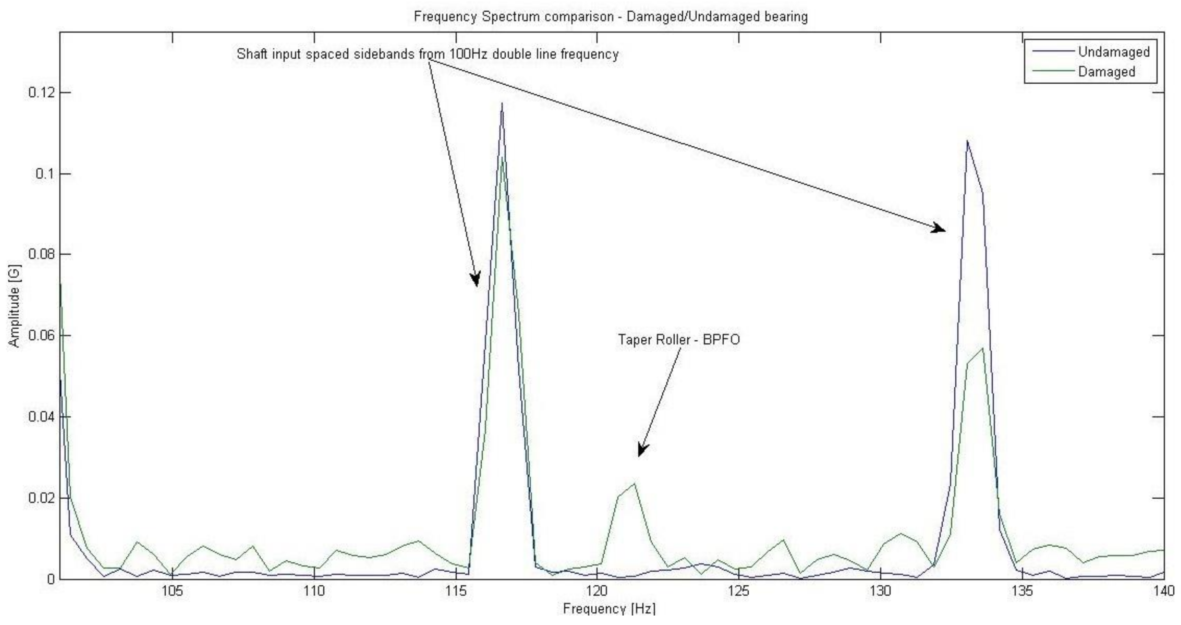


Figure 42: Frequency spectrum at 500RPM input shaft

As with the 500 RPM data, the signal components visible in these spectra are the BPFO of the taper roller bearing as well as sidebands of the 2×line frequency of the electric motor (100Hz) with the sidebands spaced at motor input frequency (16.67Hz). The BPFO is again only visible in the spectrum derived from the damaged bearing as expected.

Further tests, not explicitly shown here, show a similar trend of taper roller bearing BPFO damage in the spectra derived from the damaged bearing, but not the undamaged bearing. In the 800 RPM spectra, it is the case that the BPFO damage is concealed by the powerful 100Hz 2× line frequency and therefore is of no use and not included. The frequency plot illustrating the 2×line frequency is given below:

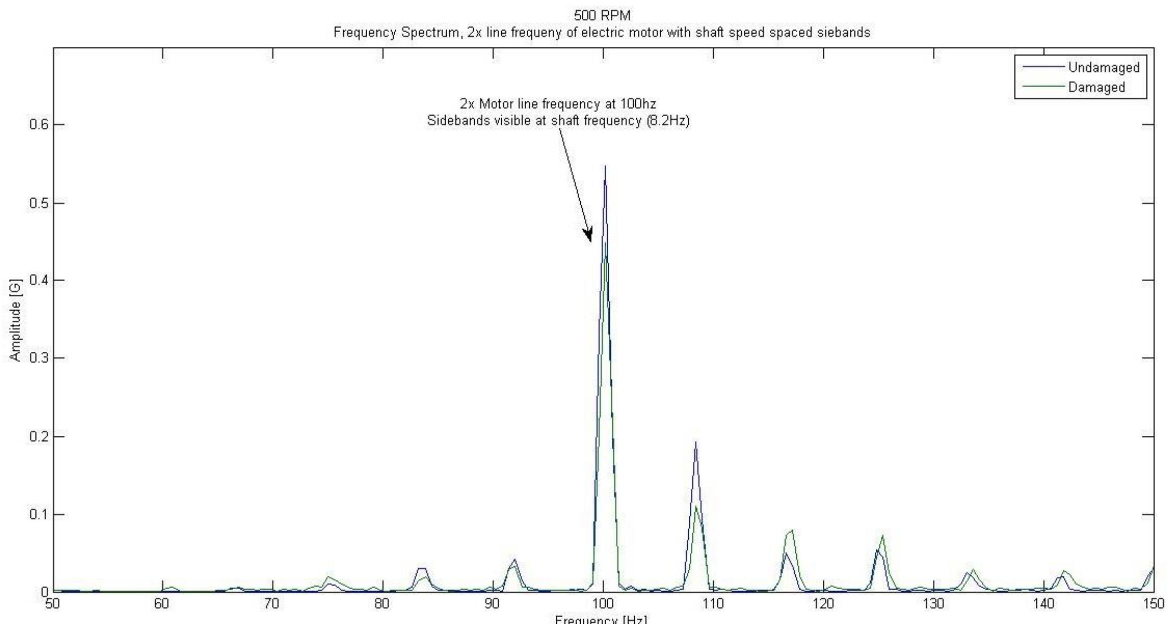


Figure 43: Spectrum illustrating 2x line frequency of motor with modulation at shaft frequency

4.2.1.5.b Algorithm testing

The algorithm, discussed in Section 3, was tested on the test gearbox data for calibration purposes as well as getting an idea of the frequency spectrum characteristics and tendencies. Calibration changes that were made included the following:

- Input frequency searching parameters
- Gear ratios
- Bearing frequencies

A limitation of the test setup was found to be the distinction with which the test bearing defects were detected in the frequency spectrum – that is to say, peaks at the fault frequencies. Compared to the gearbox tests at Secunda, where the peaks in the frequency spectrum of damaged bearings registered above 10× of the median of the noise around the peak, the damaged bearing of the test setup registered usually around 4 times the median. This is likely due to the damage being seeded partially outside the loading zone for a tapered roller bearing in pure axial load (as can be seen in Figure 36) and therefore is not an accurate representation of the typical values one would see in practice.

Therefore, judging the bearing based on the previous parameters (spectral peak value of 10×median and 5×median of the surrounding noise) does not be an accurate representation of the test as the bearing is clearly damaged but registers mostly in the 4×median range. So, based on the tests the following parameters could be ascertained regarding the undamaged and damaged cases.

Table 9: Peak/Median ratios at different speeds for the undamaged and damaged bearing

Speed	**Undamaged**	**Damaged**
RPM	Ratio: Peak/median	
500	0.92	8.45
600	1.31	3.22
700	1.30	3.03
900	1.61	4.99
1000	1.72	5.03

Comparing ratios between peak and median for the damaged and undamaged cases reveals that there is not a large enough difference to be able to distinguish between the cases of "Normal vibration", "Vibration caution" and "Vibration warning", as discussed in section 3.2.6. Instead, one can only really distinguish between the cases of "Normal vibration" and "Vibration warning". The boundary for "Vibration warning" case will be set at twice the average of the undamaged ratios above. The average being 1.37, therefore the boundary condition will be set as approximately twice that, namely 2.7 (actual twice value 2.74).

As with the comparison between the damaged and undamaged bearing in the spectra above, the cases of 500RPM and 1000RPM serve as examples of the working of the algorithm as well. Note that the diagnosis boundaries are illustrated as well and they are plotted across the range for which they are computed (±5Hz). The identified frequency is plotted with a red dot. It is worth comparing the spectra illustrating working of the algorithm below with the spectra illustrating the cases of the damaged and undamaged bearings in the previous section.

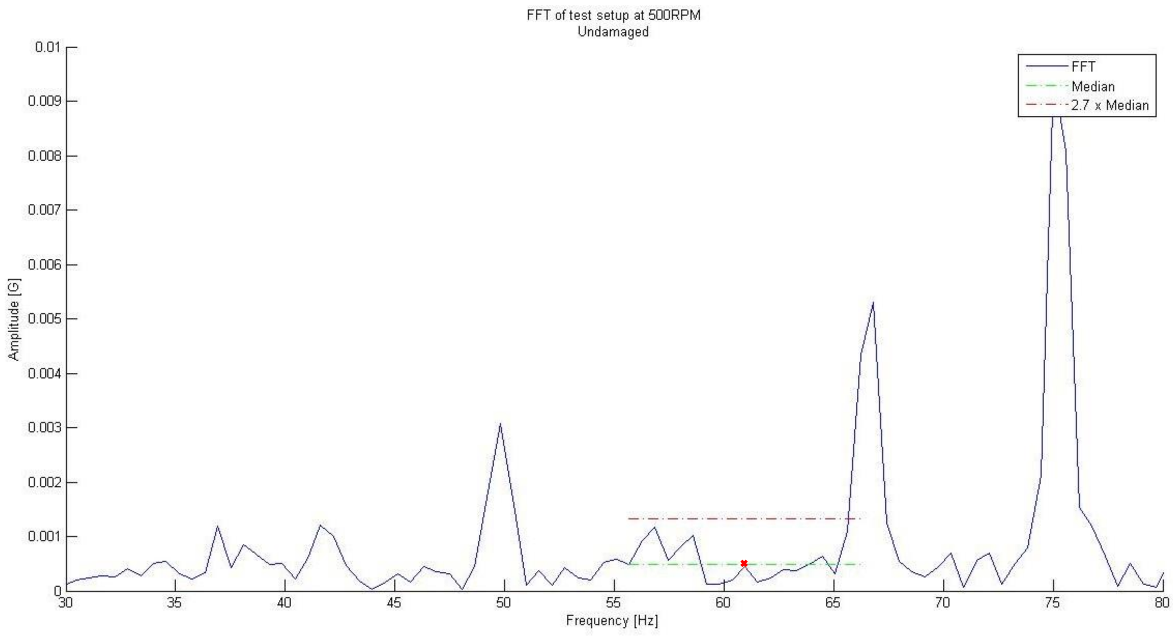


Figure 44: 500RPM undamaged bearing

The first case above is for the undamaged bearing. Looking at the spectrum above one can easily identify sidebands of the 2× motor line frequency at around 67Hz and 75Hz (identified in Figure 41) as well as the accelerometer power supply at 50Hz. However, comparing the figure above to Figure 41, one can see that the prominent peak at BPFO frequency is absent and also note the median and 2.7×median boundaries plotted as well. As a matter of fact, the peak identified (marked with a red dot) is slightly below the median value, having a ratio of 0.92. Therefore, as there is no peak in the expected location, the finding is “Vibration Normal”.

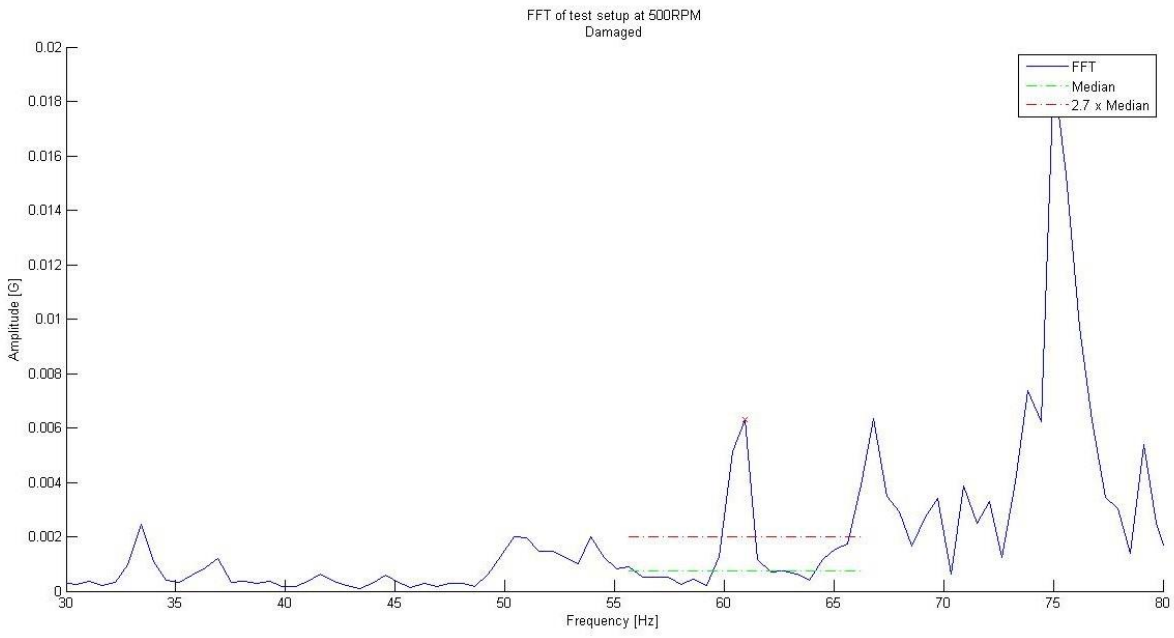


Figure 45: 500RPM damaged bearing

Observing the spectrum of the damaged bearing however, the BPFO peak is clearly identified at about 61Hz. Comparing the spectrum above with Figure 41 and Figure 44, one can clearly see the peak of the BPFO and how it stands out above the median and 2.7×median boundaries. In fact, the ratio of the peak to the median is a substantial 8.45, which is the largest seen in the test and not really typical, as with the current setup, peaks are somewhat lower, as will be demonstrated with the 1000 RPM spectrum. Also visible is the 4th harmonic of the input speed at around 33Hz and again the now familiar sidebands at shaft frequency around the 100Hz 2× line frequency.

The case for 1000 RPM will be illustrated below:

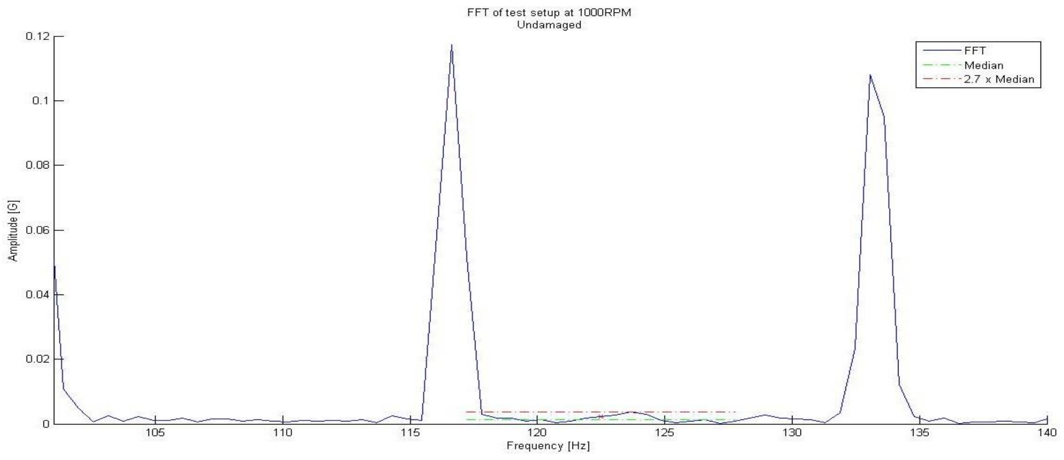


Figure 46: 1000RPM undamaged bearing

At first sight, the undamaged case of 1000RPM is a somewhat misleading, as it appears that a fairly broad and low amplitude peak was partially identified. Yet comparing the spectrum above with Figure 47, one can see that the actual BPFO peak lies lower in the frequency spectrum than the shallow low peak seen in the figure above and that the activity between 116Hz and 130Hz is merely noise. Incidentally, those peaks are the familiar shaft frequency spaced sidebands surrounding the 100Hz 2× line frequency from the motor. As the identified peak is only 1.72 times the value of the median and thus the finding is “Normal vibrations”. Below the damaged case is discussed.

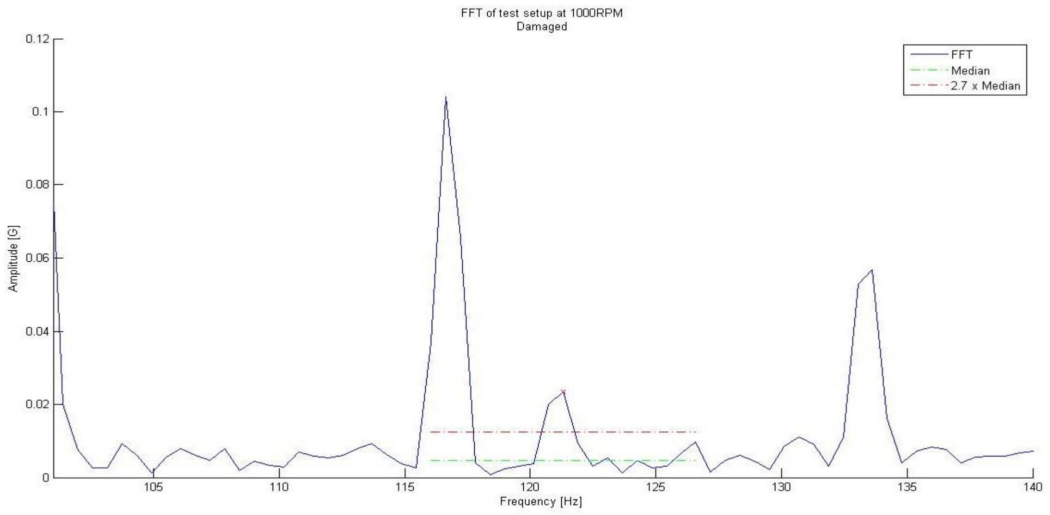


Figure 47: 1000RPM damaged bearing

Compared to the undamaged case (Figure 46) and the overlay of the damaged and undamaged cases (Figure 42), one can clearly see the peak of the damaged bearing at the BPFO frequency of about 121Hz in addition to the sidebands around 100Hz. The amplitude of the peak is sufficient to trigger a "Vibration warning" finding as the amplitude of the peak is about 5 times that of the median and thus above the 2.7× boundary.

4.2.1.5.c Exploratory test conclusion

As can be seen in the results of the test in the SASOL lab, the test bench does not produce a significant peak even with the damaged bearing. As stated, this is likely due to the fact that the seeded damage was not properly in the loading zone. Nevertheless, it is still possible to make out the cases of the damaged and undamaged bearing.

Because the amplitude peaks in the spectrum is not as clearly defined as it is with the tests done at the SASOL plant in Secunda, it was decided not to include an intermediate warning stage (previously defined as "Vibration caution"). Instead only "Vibration warning" and "Normal vibration" states were evaluated for the purpose of this test bench as explained earlier. These parameters are now substituted into the developed hardware.

4.2.2 Developed hardware testing

4.2.2.1 Setup

In the figure below, one can see the test setup. It comprised of the accelerometer, accelerometer power supply, developed hardware, ICD programmer and a laptop.

Due to the LCD not being operational, all the parameters had to be read directly from the registers on the MCU, which required the Laptop.

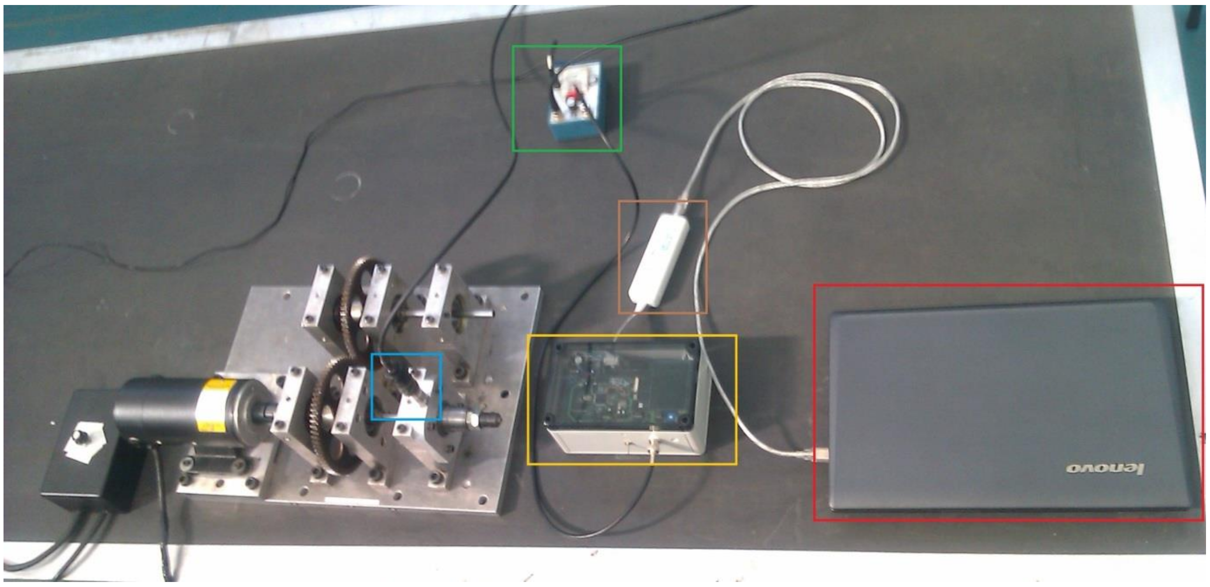


Figure 48: Developed hardware test setup

Figure 48 key:

Green:	ICP Power supply
Red:	Laptop
Yellow:	Alpha prototype
Brown:	ICD Debugger and programmer
Blue:	Accelerometer

A diagrammatical illustration of the test setup is given below. Notice the difference to the tests setup of the spider tests. In this case, the laptop is not used to perform any form of signal analysis, rather it is used purely to read what the MCU is doing (as the screen of the Alpha prototype was not operational). All the signal processing was done by the MCU. In addition, it uses battery power, rather than grid power. The accelerometer still used an external power supply however, which is something that was addressed in the Beta prototype of the ADM.

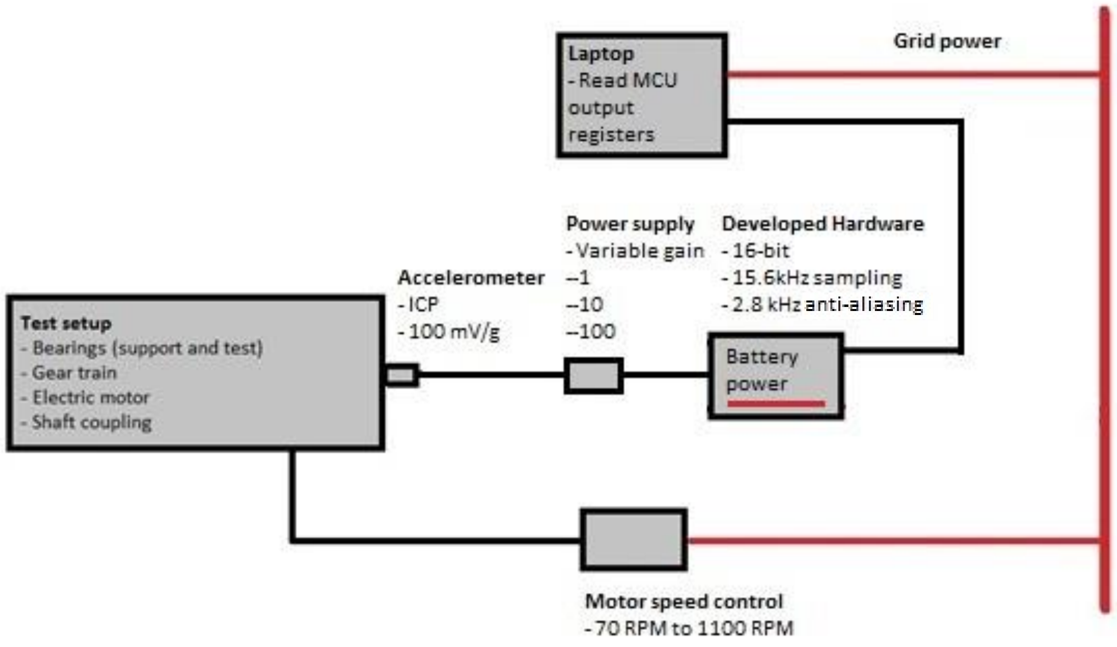


Figure 49: Synoptic diagram of the test setup for the developed hardware

4.2.2.2 Tests results

Next, the developed hardware was tested back to back with the Spider in conjunction with computer software to see if the results correlated. Ideally one would have liked to download the data from the hardware memory and compare the frequency plots, however this was not possible as the Alpha prototype hardware does not have this functionality.

Therefore, the way this was done, was to compare in the damaged and undamaged cases at all the applicable speed settings and to see whether the ratios of peak/median correlated, as this was ultimately the deciding factor in the algorithm and these values would be a good indicator if the Alpha prototype was working correctly. Absolute values were not compared because the data from the ADC on the Alpha prototype was not calibrated as this would have taken up unnecessary processing time and absolute values are not important in the fault finding algorithm, only relative values.

In the tables below the results are summarised. Firstly the undamaged cases for both the Spider and Alpha prototype, followed by a similar table for the damaged cases.

Table 10: Ratio results comparison over different speeds (undamaged)

Speed	Ratio: Spider	Analysis	Ratio: Hardware	Analysis	Difference
500	0.92	Vibration normal	1.26	Vibration normal	0.34
600	1.31	Vibration normal	1.32	Vibration normal	0.01
700	1.30	Vibration normal	1.13	Vibration normal	0.17
900	1.61	Vibration normal	1.71	Vibration normal	0.10
1000	1.72	Vibration normal	2.04	Vibration normal	0.32

In Table 10 above, one can see the ratios of the peaks to the median values. Although the absolute values will be vastly different, it does not matter as only relative values are used. These can be seen compare very favourably with the average difference being 0.19, the smallest difference being 0.01 and the largest difference between the ratios being 0.34. In addition, all these ratios of both the Spider and the hardware are smaller than the 2.7×median boundary and thus all the cases are analysed as "Vibration normal".

Table 11: Ratio results comparison over different speeds (damaged)

Speed	Ratio: Spider	Analysis	Ratio: Hardware	Analysis	Difference
500	8.45	Vibration warning	8.06	Vibration warning	0.39
600	3.22	Vibration warning	2.86	Vibration warning	0.36
700	3.03	Vibration warning	2.84	Vibration warning	0.19
900	4.99	Vibration warning	4.28	Vibration warning	0.71
1000	5.03	Vibration warning	4.55	Vibration warning	0.48

Table 11 shows the results for the damaged bearing for both the Spider and the hardware. Once again, the results correlate well. The variance in ratios are somewhat larger than with the undamaged bearing, still somewhat less than the actual ratios (typically about 10%, and at most 16% of the actual ratios), the largest being for the case of 900RPM for which the difference between the Spider data and the developed hardware is 0.71. The larger variance may be attributed to more noise present when the damaged bearing was tested, thereby directly influencing the median value and the corresponding 2.7× damage boundary. Nevertheless, all the values are above the 2.7 ratio and were all classified as "Vibration warning" both in the spider tests and in the developed hardware.

Two cases with the developed hardware, 600 and 700RPM came fairly close to the boundary value as their ratios were about 2.8. Similar behaviour was seen in the SASOL tests in Secunda where some values were near the boundary of a damage case. This is certainly a limitation of a damage diagnosis system where so few steps in the diagnosis are involved, in this case only 2 ("Vibration normal" and "Vibration warning") and with the SASOL setup 3 ("Vibration normal", "Vibration caution" and "Vibration warning").

4.2.3 Conclusion

The premise for evaluating the working of the Alpha prototype in comparison with the Spider tests was simply to gain experience using an embedded system on an actual test setup and eliminate any firmware instabilities. Seeing as the software routines on the Beta prototype are very similar, the tests were not repeated for the Beta prototype. The only anomaly that could therefore occur was if the hardware was faulty, which would be detected during the normal 'testing and debugging' phase of any hardware development.

Regarding the firmware, it was also useful to ascertain that the signal processing routines performed as expected. An analysis was thus also attempted on a test bearing after confirming that the signal recorded by the Alpha prototype and the spider hardware are consistent (giving credibility to the results of the processed signal and its analysis).

By that token, the hardware can be seen to work correctly and correctly detected the bearing as being damaged. The test setup however proved not be ideal as the damage detection boundaries had to be modified somewhat. This though is a function of the test setup and not the developed hardware, which functioned well nevertheless.

4.3 ADM Beta prototype calibration

4.3.1 Summary

Although the current algorithm only uses relative measures in the frequency domain, it is anticipated that time domain techniques might be used at some point. It may also be useful to display (on the LCD) or log time domain data and parameters, such RMS, CF etc. In order to avoid confusion when a third party observes the recorded data, accurate calibration is a necessity.

It is considered prudent to perform a manual calibration by means of a test (as opposed to an analytical derivation) for the current device as well as future production devices for the following reasons:

- Variances in sensitivities of sensors
- Variances in electronic components
- Variances in cable lengths of sensors
- Variances in internal circuitry of the devices

For these reasons, it was decided to perform a manual calibration of the device. This was done by measuring vibrations of known amplitude both with the Beta prototype as well as a reference instrument and comparing the results.

The measured signal was created by shaking the sensor with a hydraulic actuator. The following displacement signal was constructed and used as an input to the actuator:

Table 12: Calibration signal

Parameter	Value
Signal form	Sinusoidal
Amplitude	10 mm
Frequency	9.54 Hz
Duration	15 s
ADC sampling rate	1 kHz

The parameter considered first for this signal was the frequency. It was primarily chosen so as to exactly fall on a frequency bin (in this case, the 100th). It also needed to fall within the overlap of the usable bandwidth of the accelerometer (between 2 Hz and 10 kHz) as well as what the actuator was capable of. The amplitude that resulted was a function of dynamic response envelope of the actuator. The resulting acceleration was verified beforehand to ensure no clipping of the data in case of excessive acceleration.

4.3.2 Test setup

The test setup used two controller cards: the National Instruments (NI) PCI-6733 DAC card and the PCI-4474 ADC card. As the name implies, these devices consist of PCI cards inserted into a PC from which it is controlled as well (via Matlab, in this case). Each card has two channels.

The signal, constructed using Matlab, was input to one of the DAC channels of the NI controller that controls the actuator.

The resulting motion of the actuator was measured in three ways: the internal LVDT of the actuator measured the displacement of the piston (using an ADC channel of the National Instruments ADC card), a reference accelerometer measured the acceleration at the tip of the rod (using the remaining ADC channel of the NI ADC controller), and the Beta prototype measured the acceleration at the tip of the rod as well.

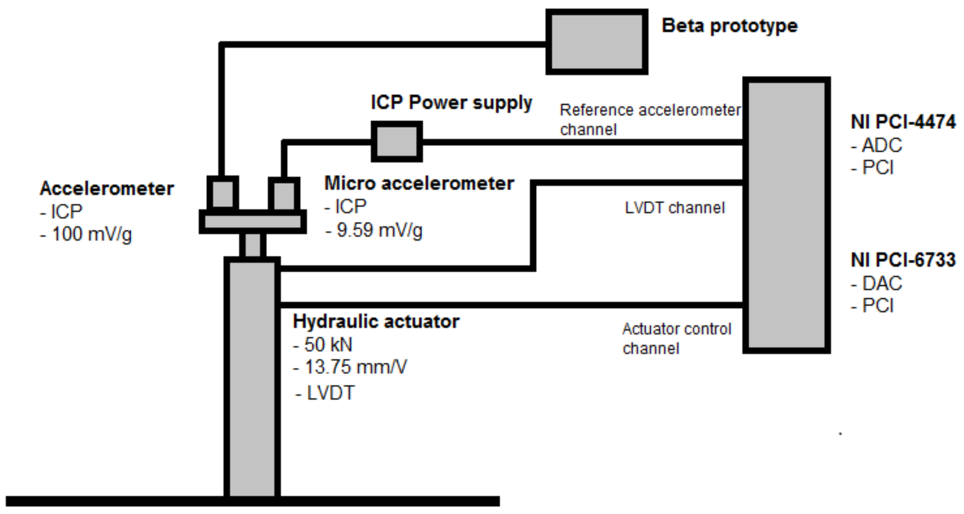


Figure 50: Beta prototype calibration setup

4.3.3 Procedure

The procedure involved sending the constructed displacement signal through the NI DAC card to the actuator. The ADC card immediately started recording the LVDT and reference accelerometer signals. The Beta prototype started sampling once the actuator started (initiated by hand).

The sinusoidal signal was played for 15 seconds, during which the measurements, described above, were taken. The signal from the Beta prototype was downloaded unto a PC from the SD card and compared with the measured signals from the NI ADC card.

The acceleration signal from the Beta prototype could be directly compared to the reference acceleration signal. As the LVDT measures the displacement of the piston, the double derivative of the signal could be compared to the acceleration signals of the reference accelerometer and the Beta prototype.

4.3.4 Calibration Philosophy

Several properties were considered for use as a calibration factor. The maximum value was considered, it is however based on a single data point in each set being the largest in their respective sets. This was not considered the best as the single data point may not be representative of an entire signal.

The best parameter was thought to be RMS. This parameter takes into account the entirety of the recorded signal and is less susceptible to instantaneous values.

The RMS of the reference accelerometer was compared to the RMS of the Beta prototype. The double differentiated LVDT signal was used for verification purposes, as it is not a direct measurement of the accelerometer mounting plate, but a displacement measurement of the actuator piston.

The following calibration value was then used:

$$\mathrm{cal} = \frac{\mathrm{Ref}_{\mathrm{RMS}}}{\mathrm{ADM\ beta}_{\mathrm{RMS}}} \qquad \text{[Eq. 28]}$$

This is to say, the calibration value consisted of a single multiplier generated by dividing the RMS value from the reference accelerometer signal with the RMS value from the ADM beta prototype signal. To calibrate the ADM Beta prototype now simply involved multiplying the time domain values measured by its ADC with the calibration factor.

4.3.5 Results

The results are shown below in Figure 51. The blue plot is the double differentiated LVDT, the green plot is the reference accelerometer and the red line the ADM Beta prototype after calibration.

It is noted that the double differentiated LVDT acceleration signal looks significantly noisier; the sources of this may include the following:

- Dithering noise on the hydraulic valves of the actuator
- Numerical errors originating from the fact that the double derivation of the LVDT to obtain the acceleration signal
- The LVDT only measures the displacement of the piston, whereas the accelerometers (reference and Beta prototype) measures acceleration at the rod end.

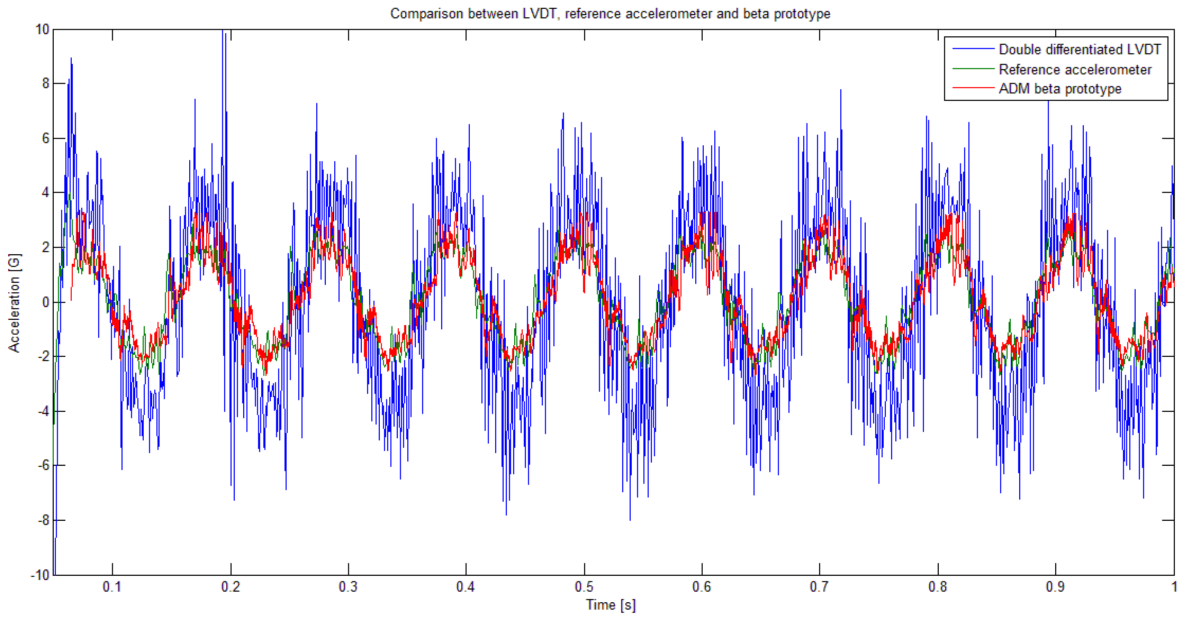


Figure 51: Comparison between LVDT (after double differentiation), the reference accelerometer and the Beta prototype (after calibration)

A further check was done to ensure that the values are in the correct ballpark. This merely involved hand calculating the amplitude of the double differentiation of a sine wave of the same magnitude and frequency as the input signal and checking that the result is similar to the magnitude of the reference accelerometer. This is performed below.

With:

S	0.01 m
f	9.54 Hz
c	2π×f

$$s(t) = S\sin(ct) \qquad \text{[Eq. 29]}$$

$$v(t) = \frac{d}{dt}s(t) = c \cdot S\cos(ct) \qquad \text{[Eq. 30]}$$

$$a(t) = \frac{d}{dt}v(t) = -c^2 \cdot S\sin(ct) \qquad \text{[Eq. 31]}$$

The amplitude is therefore:

$$|-c^2 \cdot S| \qquad \text{[Eq. 32]}$$

$$= |-(2 \times \pi \times 9.54)^2 \times 0.01|$$
$$= 35.91\ \mathrm{m/s^2}$$

Converting this value of m/s^2 to a G value yields the following:

$$35.91 \div 9.81 = 3.66\ \mathrm{G} \qquad \text{[Eq. 33]}$$

As can be seen in Figure 51, the acceleration signals correspond very well with this value, thereby adding confidence that the calibration was done correctly.

The calibration ratio that resulted was:

$$\mathrm{cal} = \frac{\mathrm{Ref_{RMS}}}{\mathrm{ADM\ beta_{RMS}}} = 6866.03 \qquad \text{[Eq. 34]}$$

This value would then include:

- Analogue amplification of the signal in the signal conditioning circuitry
- The quantization procedure (assigning a voltage on the pin of the ADC to a bin number)
- Volt to G conversion (basically the specific sensitivity of the accelerometer)
- Various electronic variances discussed in the beginning of this section

5. Field testing

5.1 Measurement chronology and rationale

The field measurements described in this section, were all performed at the Sasol plant in Secunda. The plant utilises various types of gearboxes and being driven by electrical motors of various power outputs. It therefore provided an opportunity to gather a significant amount of data from various types of gearboxes.

A shortcoming of the test site was however the fact that all the gearboxes operated in a single application – conveyor belt drives. For this reason, it is considered vital that the system under development be exposed to different sites and different applications. For this project however, the single site was deemed sufficient.

The first set of field measurements, described in Section 5.2 below, was conducted before the hardware specification began. A large number of gearboxes were measured to determine the type and nature of the gearbox faults – specifically the bearing faults. From the results of these measurements, the algorithm set and hardware specification of the Alpha prototype was generated.

The second field measurement campaign was executed after the Alpha prototype was manufactured and tested in the laboratory. It was a verification exercise to observe the behaviour of both the hardware and software in the field. The lessons gleaned from the laboratory testing and especially the field testing of the Alpha prototype were incorporated into the specification and design of the definitive hardware of this project – the Beta prototype.

Finally, the Beta prototype was field tested. It contained a considerable amount of revisions compared to the Alpha prototype and had to be thoroughly tested. The tests were somewhat more extensive, due to the fact that laboratory tests were not as exhaustive as performed on the Alpha prototype. As with the Alpha prototype, the tests were performed to test the hardware and software of the system. Qualitative lessons regarding the practical use of such a system was also learned and discussed in the conclusion and further work section.

5.2 Initial data gathering campaign

5.2.1 Conveyor belt drive layout

As mentioned previously in this document, the field tests took place at the SASOL plant in Secunda. The measurements were made on several gearboxes in the Coal Supply section of the plant. These gearboxes formed an integral part of the drive line of the conveyor belts transporting coal to the production section of the plant.

Typically such a drive system would consist of the following:

- Electric motor
- Fluid coupling
- Gearbox
- Conveyor drive roller

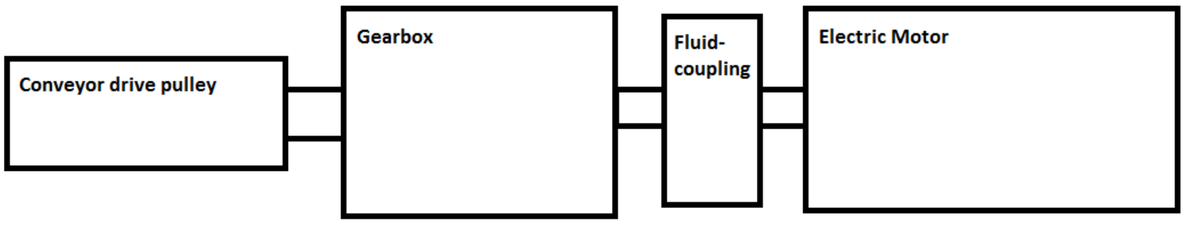


Figure 52: Conveyor belt drive arrangement

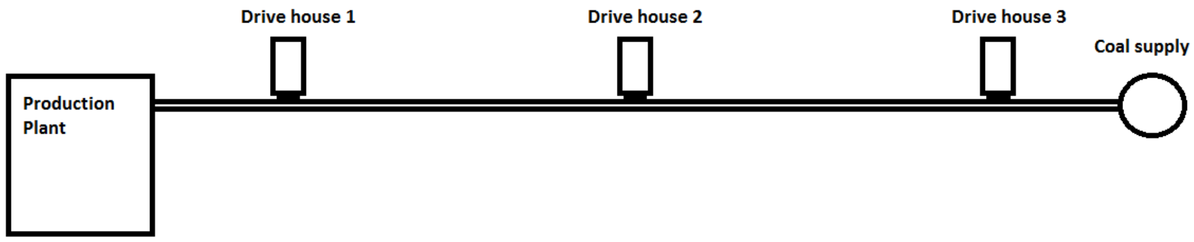


Figure 53: Conveyor belt arrangement within plant

Several (typically four) such arrangements would be present in a single drive house and several such drive houses would service a conveyor belt line, often spanning several kilometres.

5.2.2 Test setup instrumentation

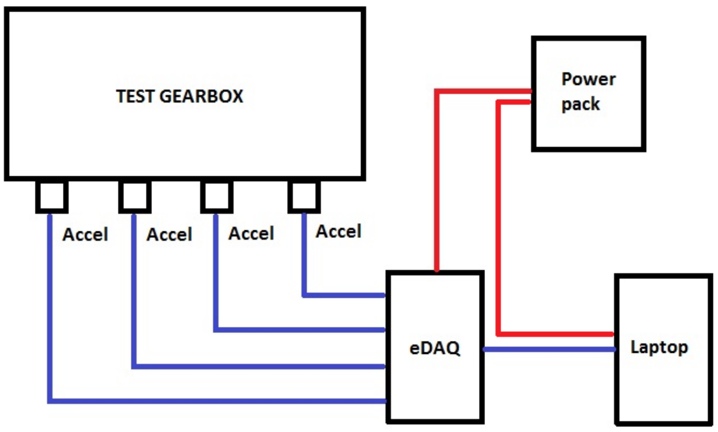


Figure 54: Test setup diagram

The test setup used to develop the software package will now be described.

This setup chiefly comprised of a commercial hardware data acquisition package, the capabilities of which far exceeded the final system's capabilities. This measure was deemed necessary to explore the boundaries of what was necessary for the final system

The test setup is depicted in Figure 54. The main part of the data acquisition system consisted of a Somat eDAQ lite. To this were connected four ICP 100mV/g accelerometers. The eDAQ was then connected to a laptop which contained the necessary control software, required for the data gathering operation of the eDAQ. As the appropriate power supply was not available, both the eDAQ and the Laptop were powered by a portable power supply.

The figures below show the actual setup of the equipment on a typical gearbox. As can be seen, measurements were taken in both the axial and radial directions. The accelerometers were attached to the gearbox via an aluminium plate (this was done to improve high frequency vibration transmission, as this often suffers with the industry standard magnetic base). This plate was glued to the gearbox casing and the accelerometers were, in turn, screwed onto the plate. The accelerometers were placed on the casing so as to make the transmission path most direct to the bearings. This involved placing it directly on the housing when measuring in the radial direction and on a blank flange edge in the axial direction (see the figures below).

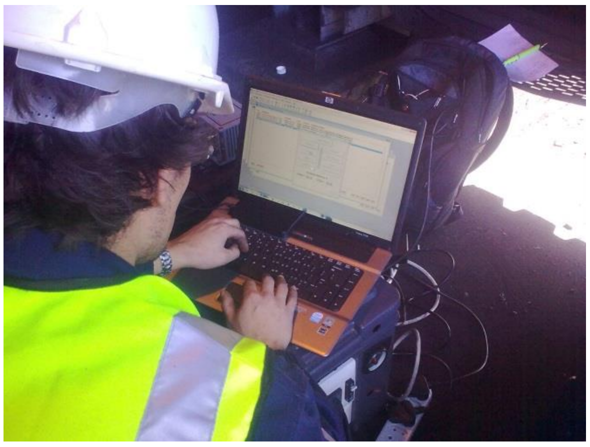


Figure 55: eDAQ unit with which data was recorded

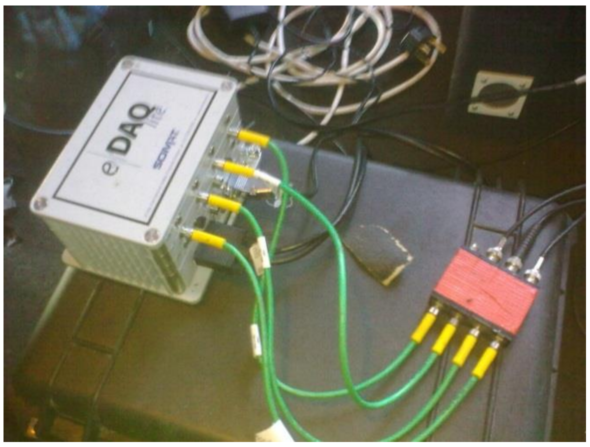


Figure 56: Laptop on which data was recorded

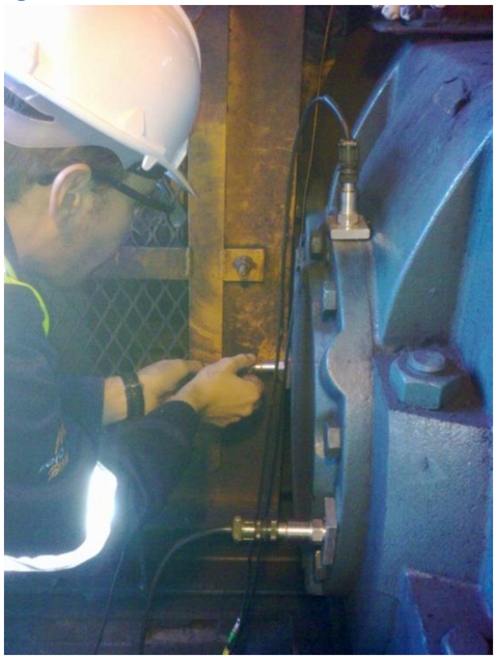


Figure 57: Accelerometers on gearbox

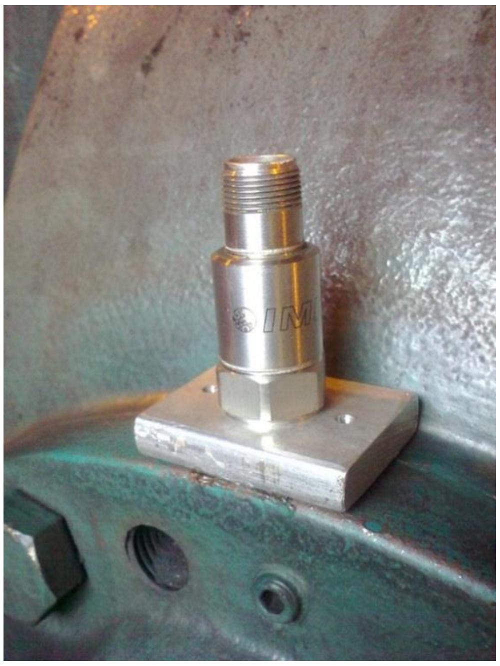


Figure 58: Detail showing accelerometer mounting

Although it would have been desirable to get rotation speed measurements as well, this was not possible due to extremely restricted access to both the input or output shafts of the gearboxes (see Figure 59). However, it was found that the input speed of the gearbox was clearly visible on the vibration spectrum as long as steady state operation was maintained.

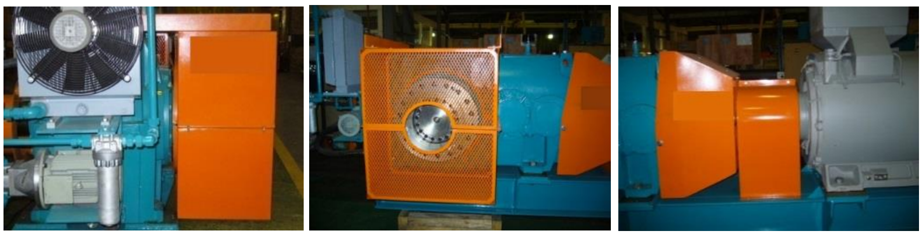


Figure 59: A typical gearbox from our industrial partner, showing the output- and input shafts' protective shrouds

5.2.3 Signal flow

In the signal flow diagram below, the signal of a single accelerometer is displayed. As all the accelerometers' signals follow same paths, it is unnecessary to show all.

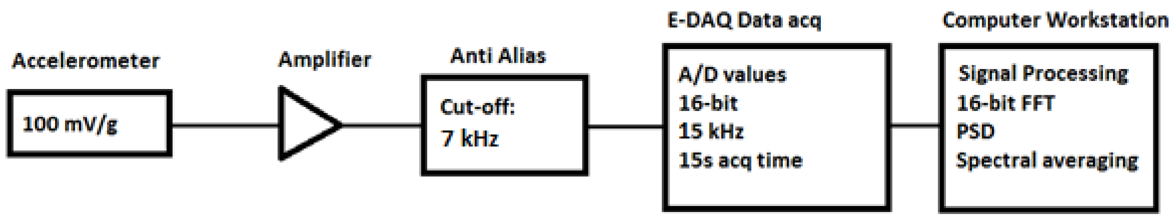


Figure 60: Test setup signal flow

The signal enters an accelerometer (100mV/g) where it is amplified and subjected to an anti-aliasing filter. From there the data gets converted to digital format by the eDAQ data acquisition unit and is temporarily stored until it is downloaded onto a computer where it is post-processed.

5.2.4 Test procedure

Tests were performed on the following gearboxes:

Table 13: List of gearboxes which was tested

Gearbox	Stages	Ratio
Type 1	2	14:1
Type 2	3	25:1
Type 3	2	16:1
Type 4	3	22.4:1
Type 5	2	20:1
Type 6	2	20:1

Measurements were taken on each gearbox listed in the axial and radial directions as close as possible to the shaft and being mindful of the best transmission path. This often resulted in the accelerometers being placed on the bolts securing the blank flanges at the shaft ends. Consultation with the condition monitoring department of coal supply confirmed this as being suitable positions as they often use them for their own measurements as well.

The measuring procedure started by cleaning the surfaces on which the accelerometers would be placed by industrial alcohol to remove any coal dust and oil as well as cleaning the intermediary mounting blocks.

These small aluminium plates were used as a mounting surface between the accelerometers and the gearbox surface (the accelerometer was screwed to one side of the plate while the other side was glued to the gearbox surface) and were glued to the surface of the gearbox with superglue. Before an aluminium plate was glued to a new gearbox surface, the residue of the previous use was cleaned off thoroughly.

After all the aluminium plates were secured to the gearbox and the accelerometers attached to them, typically 2 measurements, one axial and one radial, were made on two shafts at a time until the entire gearbox was measured. Three complete measurement sets were made per gearbox, ensuring that at least 10 minutes had passed between each measurement to allow sufficient time for any change in operating conditions (speed variations, loads, shocks, etc.) to take effect. In addition, only gearboxes that served loaded and running conveyor belts were measured.

The A/D values that were used were:

- 15s sample length
- 16-bit resolution
- 15kHz sampling frequency (limited by accelerometer resonance)
- Anti-aliasing filter at 7kHz cut-off frequency

5.3 ADM Alpha prototype field testing

5.3.1 Test description

After the laboratory tests (explained at length in Section 4.2), the Alpha prototype was tested at the SASOL plant in Secunda to see how the system would perform in a real plant. Several key observations were made regarding its performance. In addition, the commercial eDAQ was taken along as well for comparative tests.

5.3.2 Test setup

5.3.2.1 eDAQ system

As the initial data gathering test used the eDAQ as well, the test setup with regard to the eDAQ was exactly the same as depicted in Figure 54. Once again, four ICP 100mV/g accelerometers were used, while the eDAQ received power from a portable power pack. The eDAQ was connected to, and controlled by, a laptop. After each test, the test data was downloaded unto the laptop for investigation. As before it was not possible to measure the gearbox speed at either the input or output shaft. The analogue to digital conversion parameters of interest, with regard to the eDAQ system, was as follows:

Table 14: ADC parameters used by the eDAQ during the measurements at SASOL

Parameter	Value
Resolution	16-bit
Sampling frequency	10kHz
Sampling time	8s
Anti-aliasing filter cutoff frequency	3kHz
Full scale	±2G

The photos below illustrate the test setup.

In the figure to the left the electric motor and fluid coupling (underneath a protective shroud – indicated by the blue box) is shown. This assembly stands on a large metal ladder-frame structure.

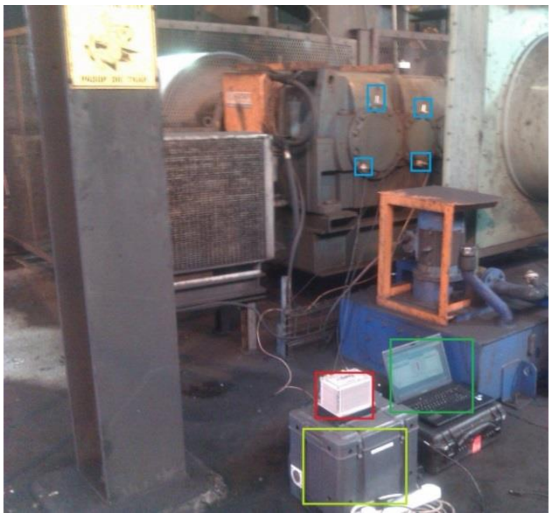


Figure 61: Test setup

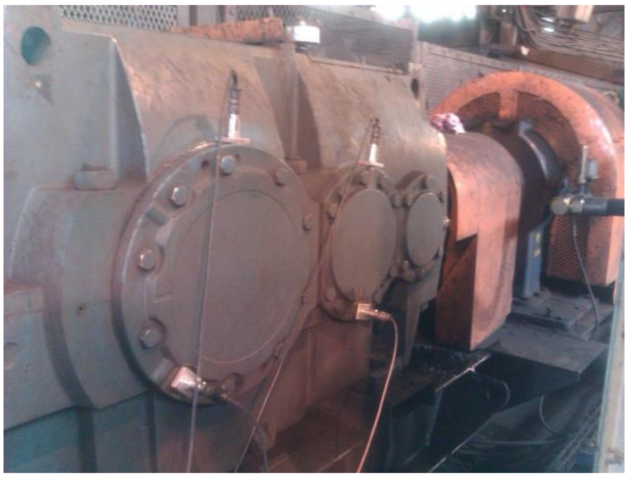


Figure 62: Accelerometer placement

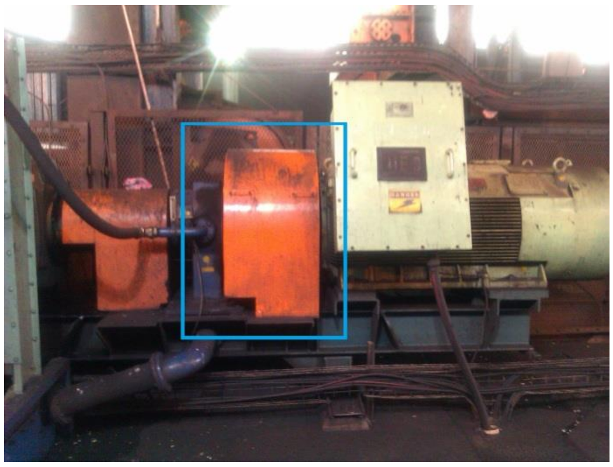


Figure 63: Typical fluid coupling in a gearbox drive system

Figure 61 key:

Green:	Laptop
Red:	eDAQ
Yellow:	Power pack
Blue:	Accelerometers

Figure 63 key:

Blue:	Fluid coupling

5.3.2.2 ADM Alpha Prototype

The testing of the Alpha prototype involved measuring the vibrations of the gearboxes one position at a time (the hardware only has one channel). The exact same positions on the gearbox were measured with the Beta prototype as with the eDAQ. This was done using the same pickup points (comprising aluminium plates glued to the gearbox into which the accelerometers were screwed). The prototype was controlled by the laptop, but all the data acquisitioning and processing was done by the prototype and the result passed back to the laptop afterwards. The analogue to digital conversion parameters were as follows:

Table 15: ADC parameters of the Alpha prototype

Parameter	Value
Resolution	16-bit
Sampling frequency	15.625kHz
Sampling time	Approximately 2s
Anti-aliasing filter cut-off frequency	2.8kHz
Full scale	±10Gs (±1V)

Figure 64 key:

Green:	Laptop
Red:	Beta prototype
Yellow:	Power pack
Blue:	ICP Power supply

In, the test setup of the developed prototype is illustrated. As with the eDAQ, the developed prototype was permanently connected to the laptop, as the required hardware to make it fully independent was not operational. However, only the results were relayed back to the laptop and all the data acquisitioning and computation was done on the hardware. Figure 65 condenses this information into the signal flow of the Alpha prototype.

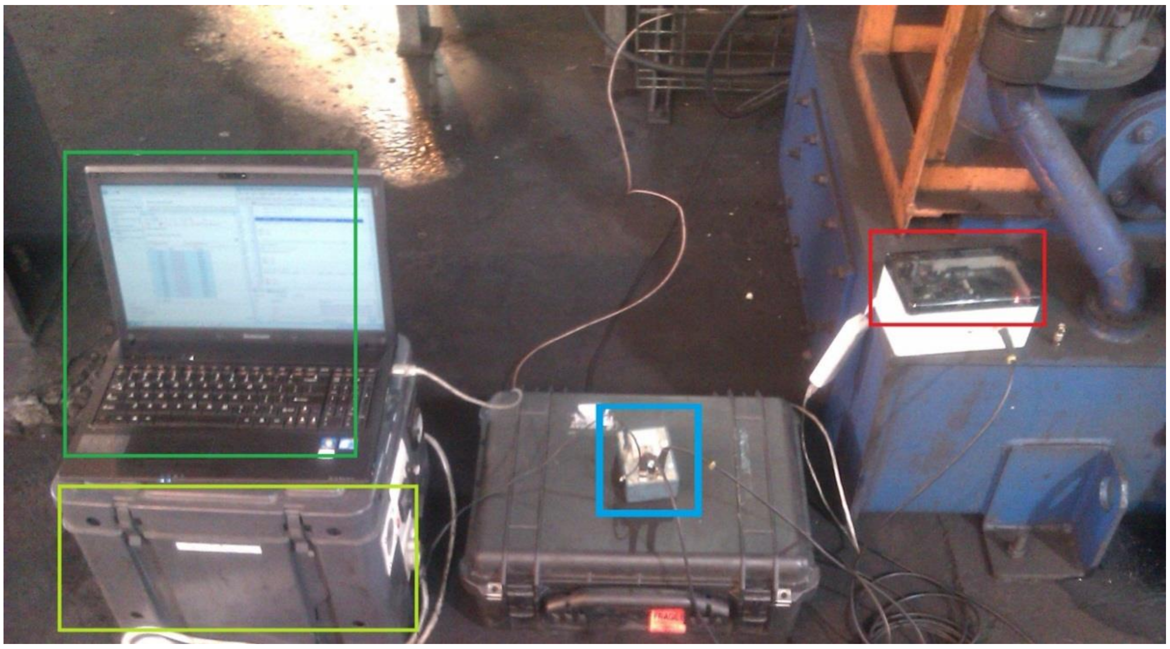


Figure 64: Test setup of Alpha prototype during test at SASOL

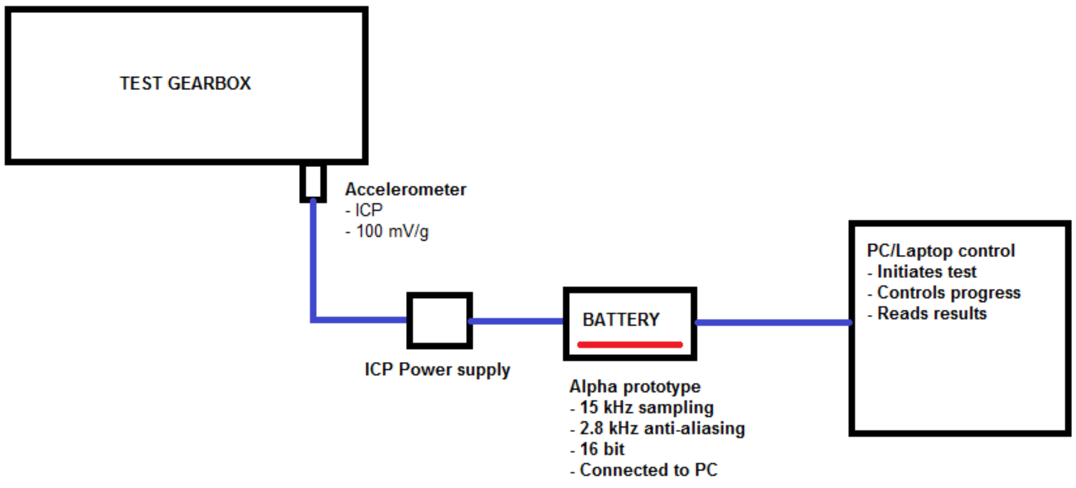


Figure 65: Signal flow of the Alpha prototype

5.4 ADM Beta prototype field testing

5.4.1 Test description

After the calibration laboratory tests, the Beta prototype and a reference instrument was taken to the SASOL plant in Secunda, as was done in the initial measurements and the Alpha prototype. The reference instrument was once again the proven eDAQ system.

During the test, the eDAQ as well as the Beta prototype was used to measure the vibrations of four different gearboxes. Signal quality (a problematic facet of the Alpha prototype) as well as diagnostic accuracy were tested and is discussed in Section 6.3.

5.4.2 Test setup

5.4.2.1 eDAQ system

As with the previous tests, the eDAQ system was used as a basis for comparison, as it is an industrial and well proven system. The test setup was therefore very similar to previous measurements of the project. Figure 66 and Figure 67on the next page details the test setup.

The procedure for the test was similar to previous tests as well. After the conditions were deemed to be steady state, the measurement run was initiated for 5 minutes (300 seconds). After the measurements, the data was downloaded and saved to the laptop. The following table lists the A/D parameters:

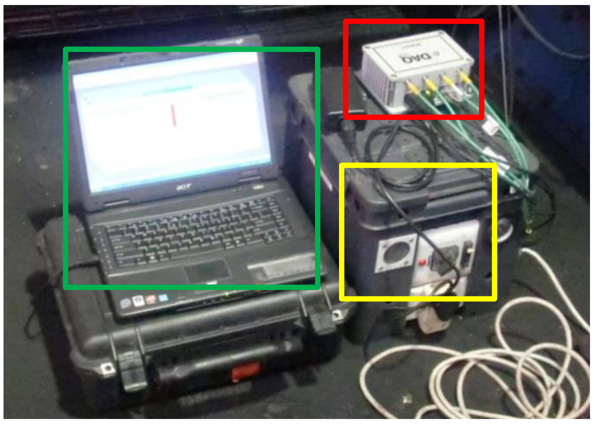


Figure 66: eDAQ test setup

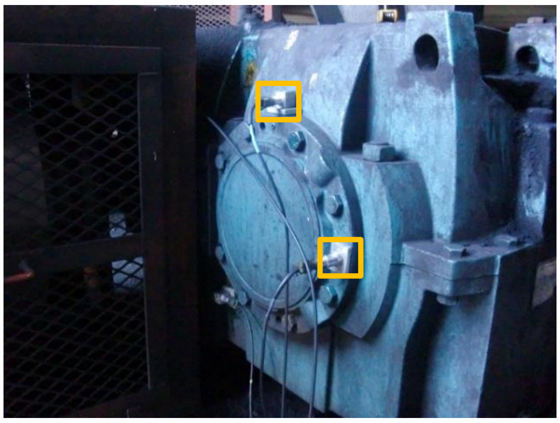


Figure 67: eDAQ test setup (Cont.)

Figure 66 key:

Green:	Laptop
Red:	eDAQ
Yellow:	Power pack
Orange:	Accelerometers

Table 16: ADC parameters as used by the eDAQ

Parameter	Value
Resolution	16-bit
Sampling frequency	20 kHz
Sampling time	300s
Anti-aliasing filter cut-off frequency	7.9 kHz
Full scale	±10G

5.4.2.2 Prototype

After the benchmark test was done on gearbox using the eDAQ system, the same measurement points (using the same exact same sensor per measurement point as well) were tested using the Beta prototype. It was attempted to get as close as possible to the same state of the drive line, i.e. steady state running with as close as possible to the same amount of load on the conveyor belts. The photographs below illustrate the test setup.

Figure 68 key:

Green:	Accelerometer
Red:	Beta prototype

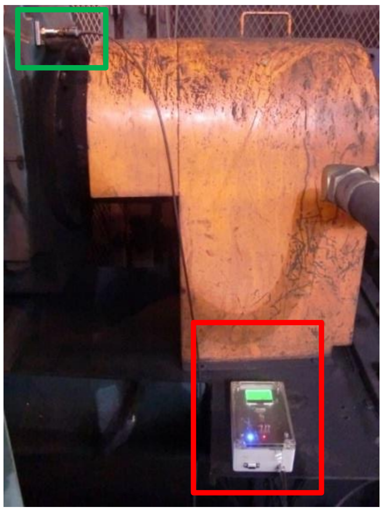


Figure 68: Beta prototype hardware setup

The ADC values of the Beta prototype were hard programmed into the firmware to be as follows:

Table 17: ADC parameters as used by the eDAQ

Parameter	Value
Resolution	12-bit
Sampling frequency	6.25 kHz
Sampling time	10.5 s
Anti-aliasing filter cutoff frequency	2.1 kHz
Full scale	±16 G

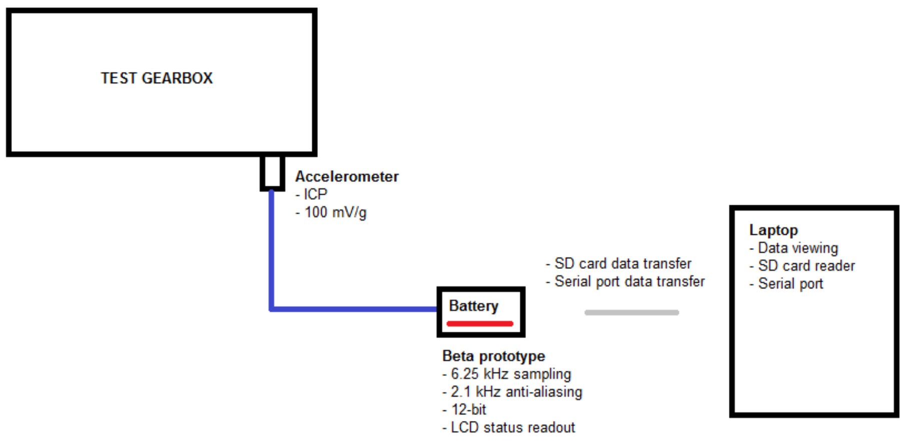


Figure 69: Diagrammatical illustration of the Beta prototype test setup

The signal flow of the Beta prototype is illustrated in the figure above. As can be seen when comparing the signal flow diagrams of the eDAQ test setup (Figure 54) and the Alpha prototype test setup (Figure 65) the signal flow of the Beta prototype (Figure 69 above) is progressively simplified. This is due to the system becoming more self-contained as the project continued. Due to the deliverable of the project being a hand held and highly portable device, this aspect was absolutely necessary. Indeed, the Laptop in the diagram above does not need to be present when the device is in the field and its presence is only required when the data is downloaded. Therefore, when in the field, only the device is required is the Beta prototype along with the attached accelerometer.

6. Field testing results and interpretation

6.1 Chronological discussion

This section discusses the data gathered during the field measurements of this project, as described in section 5. The data processing was performed, where applicable, according to the algorithm as described in section 3.

Each set of data had an effect on the hardware used to gather the next set. In practice, this meant that the first data gathering campaign had an effect on the specification of the Alpha prototype and the results from the Alpha prototype had an effect on the specification of the Beta prototype.

The experimental development model (XDM), as described in section 2.2, was never meant to be field tested and was a bridging hardware between the initial data gathering campaign and the Alpha prototype.

6.2 Initial data gathering campaign

During the test, six different types of gearboxes were tested as per the technique described in Section 5.2. The data gathered was processed and the results analysed. The analyses performed involved a spectral analysis approach to bearing damage and formed the basis on which the eventual firmware was based. Important factors were:

1. Evaluation of bearing damage in the form of spectral peaks
2. Detection of input speed and harmonics
3. Detection of GMFs and harmonics
4. Determining if there is any interfering factors (coincident frequencies, etc.).

Below are a few noteworthy examples of the numerous samples are discussed and the relevant conclusions drawn.

6.2.1 Vibration analysis definition

Before the results are presented, it is necessary to define the analysis terminology in this text. Three stages of bearing vibrations were selected for this project. These stages are defined as follows:

Table 17: Definition of the stages of bearing damage

Analysis outcome	Definition
Vibration normal	The vibration emitted by the bearings on the shaft are judged to be of a low enough amplitude not be of concern
Vibration caution	The vibration emitted by the bearings on the shaft are judged to be of large enough amplitude to be cautious. Close attention should be paid to subsequent measurements and professional help is prudent
Vibration warning	The vibration emitted by the bearings on the shaft are judged to be of large enough amplitude to be concerned and the data should be investigated immediately. Professional help is strongly advised.

6.2.2 Time domain analysis

Some of the historical time domain methods of detecting bearing damage were tested during the development of the algorithms, namely Crest factor and Kurtosis (Patil, et al., 2010). Judged according to frequency domain techniques (which are deemed to be more accurate) and the historical feedback from SASOL, they were evaluated.

Several signal samples were taken from bearings which were qualitatively judged to be undamaged, moderately damaged and severely damaged and their kurtosis and crest factor values compared. These measurements were made on 6 different types of gearboxes. In both measurements directions (axial and radial) per measurement point were used to gather the data (the most dominant case is reported here). The following is a table of the gearboxes measured as well as the time domain results.

Table 18: Time domain analysis results

	Location*	Run	Direction	Crest Factor	Kurtosis
No Damage					
Type 4	2nd shaft	1	Vertical	4.138	2.642
			Horizontal	**4.325**	**2.979**
		2	Vertical	3.710	2.719
			Horizontal	4.156	2.725
Type 2	3rd shaft	1	Vertical	4.329	2.828
			Horizontal	5.038	2.959
		2	Vertical	4.236	2.821
			Horizontal	4.037	2.879
Moderate Damage					
Type 5	3rd shaft	1	Vertical	4.137	3.145
			Horizontal	4.822	2.894
		2	Vertical	4.885	3.084
			Horizontal	4.076	2.752
Type 5	2nd shaft	1	Vertical	4.723	3.051
			Horizontal	4.971	2.879
		2	Vertical	5.879	3.207
			Horizontal	4.048	2.730
Severe Damage					
Type 6	1st shaft	1	Vertical	4.625	3.003
			Horizontal	5.047	3.031
		2	Vertical	4.635	3.059
			Horizontal	4.762	3.039
Type 3	2nd shaft	1	Vertical	4.428	2.922
			Horizontal	4.760	2.976
		2	Vertical	4.416	2.925
			Horizontal	4.868	3.009
			Maximum	4.544	2.927
			Mean	3.710	2.642
			Minimum	5.879	3.207

**Referenced from the input shaft*

Signals with a crest factor of more than 3.5 and a kurtosis of more than 4.0 would tend to allude to bearing damage (Norton & Karczub, 2003).

Examining the signals above one can see that the minimum crest factor is in fact about 4.5 which would indicate bearing damage as it is larger than the accepted 3.5. However, this is obviously false as the first eight of the bearing samples are of undamaged bearings. Therefore crest factor can be ruled out as an indicator of bearing damage in this application

In addition, the maximum kurtosis value in the sample range is 3.2 which would indicate no damage in any of the bearings as it is smaller than the minimum kurtosis of 4.0 for damaged bearings. This again is false as the next eight samples are of data with moderately damaged bearings and the final eight samples are of data with severely damaged bearings. This would also indicate that kurtosis is not a suitable indicator of bearing damage in this application as well.

The unreliability of the crest factor value can be explained simply by the enormous array of signals composing each measurement. Present in the signals are components from:

1. The motor
2. The fluid coupling
3. Multiple shaft frequencies
4. Multiple GMFs
5. Potential bearing faults
6. Any external noise emanating from the
    a. Conveyor belts
    b. Vibrating structures on which the motor and gearbox are mounted
    c. Vibrations from the coal chutes
7. Harmonics of all the above

All these signals combined creates a noisy environment in which signals with crest factors as high as those observed is created.

The kurtosis value can be explained by its very nature. Kurtosis, being the fourth statistical moment, provides information about the "peakedness" of the sample (Samual & Pines, 2005). However, as the peaks in the signal sample becomes spread out and distributed, the kurtosis value will become lower. The many spectral components present in the signal will likely create a diversified series of peaks within the time domain leading to the lowered kurtosis value.

To summarise, the two most prominent time domain techniques commonly employed in the industry is ineffective at detecting bearing damage within the noisy environment created by the gearbox and its surroundings. These techniques will likely be more effective for isolated bearings in plumber blocks (for example) in which the frequency content is less cluttered with external signals or at least where the hardware allows some scope for filtering. This route is not possible in this project, because in embedded hardware using digital filtering for this purpose would be impractical as there would be too many frequencies to account for.

6.2.3 Frequency domain analysis

6.2.3.1 Initial remarks

Frequency domain analysis was always considered as the most promising analysis technique and it was developed as described in Section 3 “Algorithm Development”. The algorithm was initially programmed in Matlab and then applied to the recorded data.

The algorithm was applied to a measurement of every shaft of every gearbox. These gearboxes had been diagnosed previously with the help of Johan Pretorius from SASOL using their Condition Based Maintenance (CBM) techniques. As noted in the before, three stages of damage were defined, samples of which are given below. These spectra were derived from data recorded using the eDAQ data logger.

In this section, the gearboxes are described as having no damage, moderate damage and severe damage. This assessment was made by the condition monitoring department at SASOL. Therefore, during the development of the algorithm, the results of the tests were known and the algorithm adapted to best fit those results.

This is contrary to the rest of this chapter, where definitive diagnoses are not explicitly provided, but a course of action is recommended. The course of action is naturally based on the results of the algorithm developed in conjunction with known results. This course of action was verified according to known states of the gearboxes provided by SASOL.

6.2.3.2 No damage

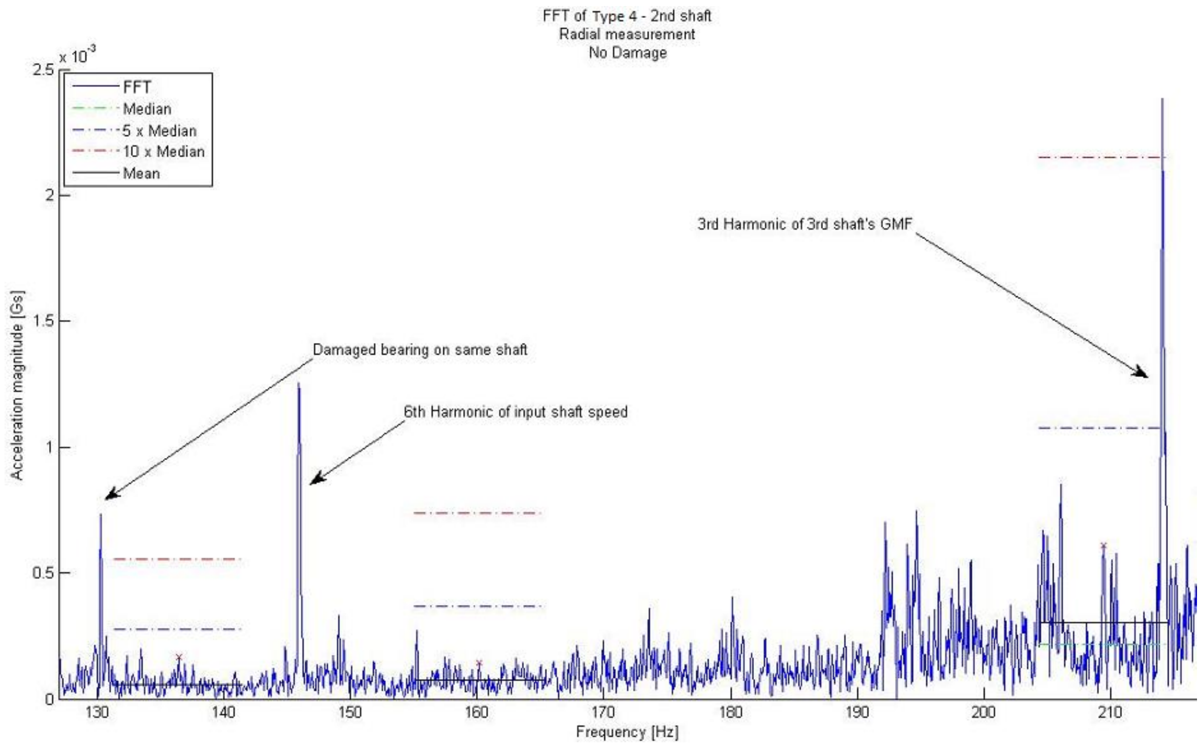


Figure 70: FFT of undamaged gearbox bearing (Type 4, 2nd shaft)

The figure above provides an illustration of an undamaged gearbox. Marked frequencies are

Table 19: Frequency components

Origin	Frequency (Hz)
A separate, but damaged, bearing on the same shaft	130.3
A harmonic of the input shaft	145.9
The third harmonic of the 3rd shaft's GMF	214.1

The frequencies of the bearing under investigation are not clearly found because no damage is present, as evidenced by the fact that the spectral peaks found are well below the boundary values, see the legend.

Also visible on the figure (and legend) is the black "Mean" line. This line, in conjunction with the "Median" line gives a visual idea of the relation between the Median and Mean. This relationship, though not important in the algorithm, is interesting to note. It lends credence to the fact that the median is a better indication of background noise than mean, due to the observation that the mean value is easily influenced by irrelevant stray peaks in the range.

One can see how the algorithm adapts according to the noise levels within the search domain of the spectrum. In the BPFO and 2x BSF regions, the noise is relatively insignificant. However, in the BPFI region, the noise is a somewhat more and therefore the search boundaries go up accordingly with the increased amount of noise.

Nevertheless, no damage is present on the bearing under investigation.

6.2.3.3 Moderate Damage

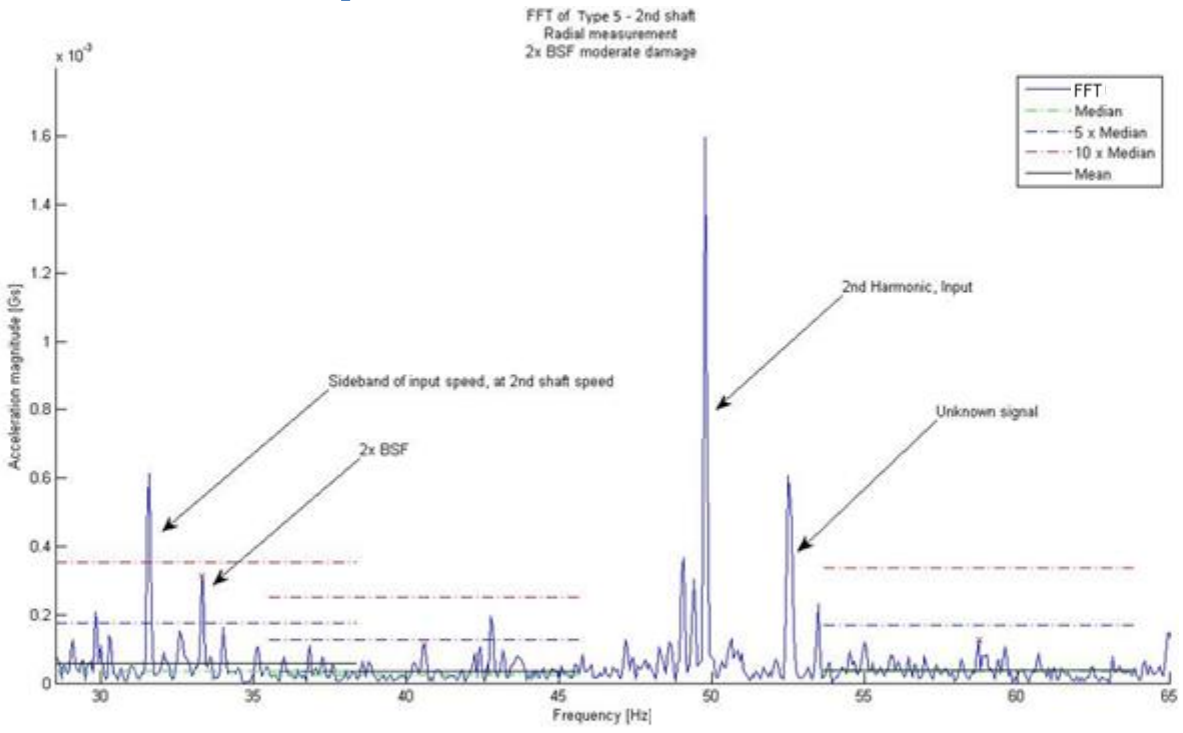


Figure 71: FFT of moderately damaged gearbox bearing (Type 5, 2nd shaft)

The figure above is a typical illustration of a moderately damaged bearing on a gearbox shaft. This particular sample was found on the second shaft of Type 5 gearbox and a ball was damaged generating a 2×BSF frequency. Including this, the following frequencies are also present:

Table 20: Frequency components

Origin	Frequency (Hz)
Sideband of input shaft speed, at 2nd shaft speed	31.58
2×BSF bearing fault frequency	33.29
2nd Harmonic of input shaft speed	49.8
Unknown signal	52.5

In this figure one can see the 2×BSF frequency peak above the 5×Median boundary indicating moderate damage to the rollers of the bearing. Again one gets an idea of how the algorithm adapts to the noise levels within the vicinity of the signal component under investigation; increasing the bounds when the noise increases.

In addition, there is an unknown signal component within the signal, appearing as a spectral peak at 52.5 Hz. The known signal components in the spectrum are:

Table 21: Known frequency components

Origin	Frequency (Hz)
Motor input	25.0
Shafts • **Gearbox input** • **Intermediate shaft** • **Output shaft**	 • 24.80 • 6.28 • 1.23
GMF 1	100.47
GMF 2	496.05
Bearing 1 • **BPFI** • **BPFO** • **2×BSF** **Bearing 2** • ***No info***	 • 59.22 • 41.23 • 33.18 • *52.5*

It is very likely that this unknown component of the frequency spectrum originates from this unknown bearing. In addition, there were many other sources of noise and external signals present in the vicinity of the test which may have an influence on the frequency spectrum including all signals from the conveyor belts, vibrating structures, etc. It does not appear to be a harmonic of any other signal as there seems to be no integer correlation between it and the other signals and neither is any geometric pattern visible on the spectrum itself.

6.2.3.4 Severe Damage

Figure 72 illustrates the case of a gearbox with a severely damaged bearing (inner race damage in this case). The measurement was taken in the axial direction on the 3rd and final shaft of the gearbox. Clearly visible on this spectrum is the BPFI peak at around 14Hz and notice that it is somewhat above the 10× Median boundary.

For the sake of clarity, because the signals are so closely spaced at low rotation frequencies the frequency markers and the boundary conditions were highlighted for the bearing spectral component under investigation, namely the BPFI.

In addition, another unknown spectral component is visible in this spectrum at 17.8Hz. This signal does not correlate with any of the known frequencies (or their harmonics) in the signal and the only conclusion that can be made is that it originates from outside the gearbox, perhaps from the vibrating structure or from the conveyor belts.

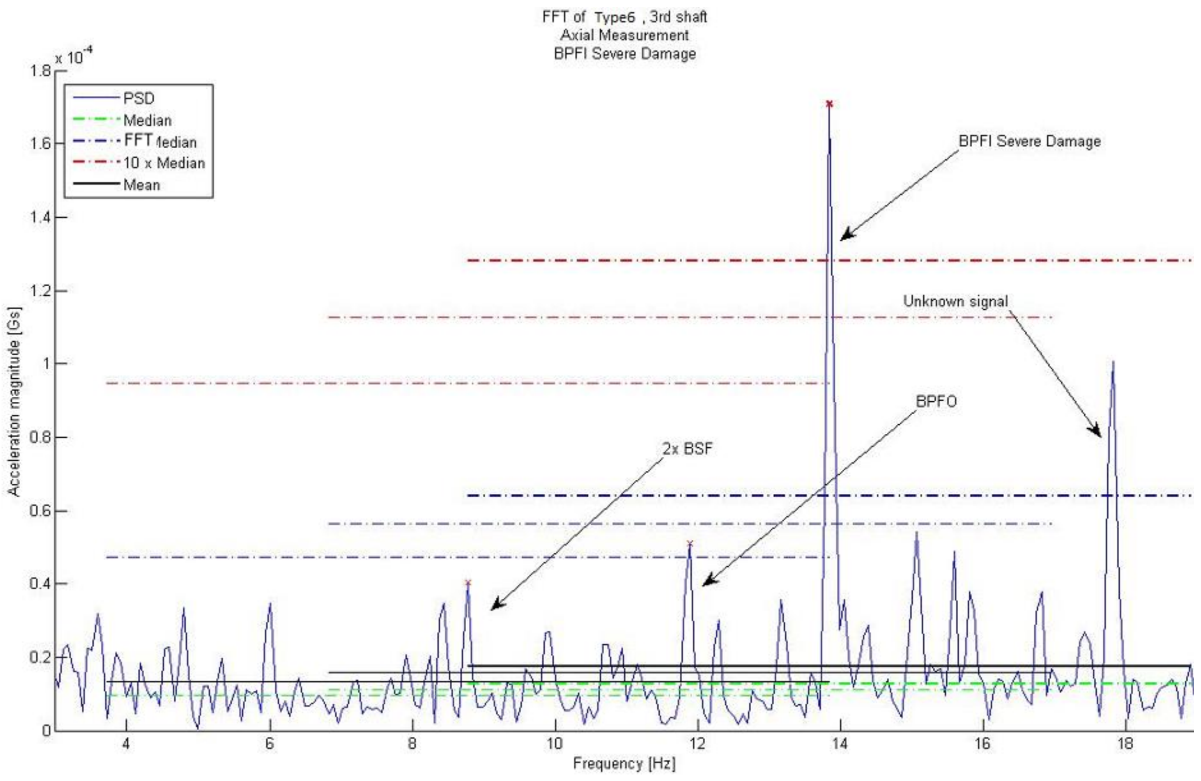


Figure 72: FFT of severely damaged gearbox bearing (Type 6, 3rd shaft)

6.2.4 Success rate

During the process of testing the algorithm, all the shafts on all six gearboxes were tested both in the radial and axial directions (some faults appeared to show up somewhat better in the axial direction). Each of the diagnoses were then evaluated with the guidance of the CBM team at Secunda and their protocols.

The success of the tests was measured according to how many diagnoses were deemed to be correct. This is defined in the context of an operator using the hand held device: if the operator measures a gearbox with a bearing that is considered – by the analysis presented in this section – as being damaged in some way, would the device convey the appropriate sentiment.

This sentiment is given in this project a direct correlation to the perceived damage state of the bearing as follows:

• No damage	–	Vibration normal
• Moderate damage	–	Vibration caution
• Severe damage	–	Vibration warning

This correlation was deduced from the philosophy employed by a number of institutions that a mechanical component still has a certain useful life span, even after initial damage is detected. That is to say, a mechanical component is not discarded after the first sign of damage, but can often operate for a definite time frame as long as the behaviour of the component is scrutinised on a regular basis.

By that token, a bearing whose vibration signature leads one to believe that it is moderately damaged can still function, as long as the user proceeds with caution – therefore a 'Vibration caution' message is given with the implicit recommendation that a specialist or OEM is involved in its observation over time.

A bearing whose vibration signature leads one to believe that it is severely damaged, possibly after a time of being judged as moderately damaged, should be attended to as soon as possible. A 'Vibration warning' is therefore issued and the user is implored to seek professional help.

Table 22: Success statistics

	Number	**Percentage**
Total number of cases	158	
Successes	127	80.4
Failures	11	6.9
Ambiguous	20	12.7

In this review of algorithm accuracy, whether the assessment itself is "Vibration normal", "Vibration caution" or "Vibration warning" was irrelevant, if it was the correct assessment (as described in the two paragraphs preceding Table 23) it was counted as a success.

Table 23 summarises the results and shows an over 80% success ratio of the algorithm. The cases marked "Ambiguous" were usually when other spectral components interfered with the assessment. This often occurred when harmonics of a different signal overlapped with the bearing frequency.

These assessments were delivered on a first attempt basis, though the problem with the ambiguous signals could often be resolved by waiting a short period and taking a measurement again. In the tests performed three measurements were taken on each shaft (approximately 30 minutes apart) of each gearbox and therefore alternative data was available.

In the case of the failed assessments there are several reasons why they did not convey the appropriate sentiment. The following table illustrates the number and cause of the failures:

Table 23: Failed diagnosis summary

Number	**Cause of incorrect diagnosis**
6	Boundary case
4	Interference
1	Too much noise in vicinity of signal
Total incorrect: 11	
Grand total: 158 (6.9%)	

In the table above, one can see that 6 of the 11 incorrect assessments were simply because the signal peak was very near the edge of the diagnosis boundary (the 5 × median or 10 × median lines). Stated differently, the peak was just barely on the wrong side of the boundary to produce a false assessment. In these cases, as with the ambiguous assessments, taking another measurement a few minutes later would usually produce a different assessment.

The second reason for failed assessments were interference, often in the form of sidebands or harmonics of a different signal (in which case, the actual backup check for harmonic interference barely failed to detect this). This happened four times and was as the result of the bearing frequency being the same as a harmonic or a sideband of either the gear train or the input frequency to the gearbox.

Lastly, a case was encountered in which the bearing frequency was located within a haystack of signals originating from a loose shaft. The many high peaks within the region therefore greatly amplified the median values within the region as well as the signal at the particular frequency so as to induce a “Vibration caution” assessment instead of a “Vibration warning” assessment.

6.3 Field testing of the Alpha prototype at SASOL

6.3.1 Results and analysis

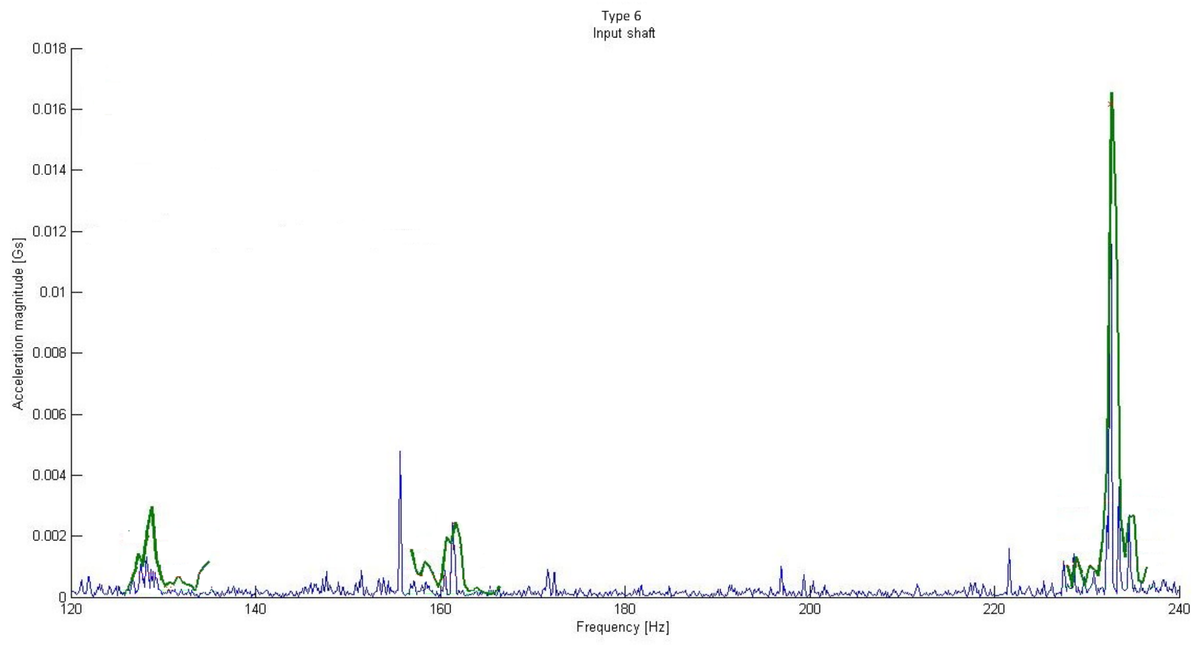


Figure 73: Spectral comparison between the Alpha prototype and the eDAQ system

The figure above is a typical example of what was found during the tests. The blue spectrum is what was measured by the eDAQ and the green overlay was the selected sections of the spectrum as measured by the prototype. These sections correspond to the 2×BSF, BPFO and BPFI of the test gearbox on its input shaft.

As can be seen, the prototype correctly detected the major peaks at the right places, indicating the correct functioning of the ADC and FFT algorithm. However, it is obvious that the finer details of the spectrum were lost during the process and only broader outline of the spectrum was captured without sufficient details to observe fine nuances such as sidebands and other closely spaced peaks. The reasons for this are as follows:

1. The somewhat low spectral resolution of 0.5Hz coupled with the minimum resolution bandwidth of the Hanning window smears the finer details of the spectrum.
2. Although a window function decreases leakage, it does not eliminate it. As leakage is a parameter that affects the amplitude accuracy of discrete bins within an FFT, the lower the spectral resolution the higher the interval of frequencies will be effected, as every bin that is affected by leakage covers a larger frequency interval.
3. As all electronic components generate noise, there is some scope for improving the layout of the circuitry so that a minimum of this noise reaches the ADC portion of the circuit. Therefore, keeping that portion of the circuit away from the rest and generally following good circuit layout practices, the noise added to the signal can be reduced.

With the quality of the results as they are illustrated in Figure 73, it was nearly impossible to distinguish nearby frequencies with any accuracy. In addition, because the spectral leakage was so high, the median value in the vicinity of the frequency of interest rose as well to levels that desensitised the algorithm.

The success rate of the algorithm could therefore not be ascertained any more accurately than during the laboratory tests and certainly could not be compared to the initial measurements from which the algorithm was developed (as explained in the Section 5.2).

The test was therefore used rather as a means to gain operational experience with embedded hardware and the useful deliverables of the test were testing of system stability and identifying operational weaknesses, as explained in the next section.

6.3.2 General comments about the performance of the hardware during the test

The purpose of the Alpha prototype was to build a first iteration device capable of acquiring and processing a signal. Once the signal was processed, an inference regarding the signal was made. During this time, several lessons were learned in preparation for the definitive Beta prototype.

The first of which was the use of an external power supply for the accelerometer. This proved to be somewhat troublesome as the units were not very reliable when in a constant on-off environment. Their batteries (three 9V units) also suffered to a large extent, needing regular replacement – especially when the power supplies were left on; and seeing how the power supply is separate from the Alpha prototype, it was an easy mistake to make.

In addition, the unit takes a considerable amount of time to process the data and yield a result, up to 5 minutes per measurement. This is considered much too long in a practical environment.

As the device did not have a dedicated means to export the data to a PC, interpreting the results was often challenging. It was possible to read the data from the internal registers of the MCU, viewable from a debugging application in the programming software – as was done in Figure 73 – but this proved very cumbersome and impractical on a large scale.

However, the system is very robust and compact. Although it was not ready to operate on its own (it was connected to a laptop throughout the tests), the functionality of a keyboard and display were emulated on the Laptop and it was very easy to use. Indeed, it did not require the skill set of an engineer to use.

All of these comments were taken into consideration for the next phase of the project, namely the Beta prototype. This hardware system incorporated remedies to all of the vices mentioned and added additional functionality as well, as discussed in Section 2.4.

The results from the field testing of that device are discussed in the next section.

6.4 Field testing of the Beta prototype

Due to being the definitive hardware and final deliverable of this project, the results from the field tests of the Beta prototype is discussed at length. Due to the poor signal reproduction accuracy of the Alpha prototype, this aspect of the hardware was at a premium and is discussed separately from the algorithm results.

The algorithm results are presented slightly differently than in Section 6.2. In that section, the results were analysed in conjunction with known gearbox condition data from the condition monitoring department of SASOL, in order to formulate an algorithm. The results in this section were not based on that of the condition monitoring department, but were verified against them.

6.4.1 Signal reproduction accuracy

The inferences made by the algorithm are based, at this time, completely on the shape of the spectra, hence why accurate correlation between the reference data (from the eDAQ) and the Beta prototype is of paramount importance. Furthermore, it is entirely possible that as this system is developed, absolute values of the data may be important as well. For that reason, the correct amplitude scaling was also considered important.

Comparing the data, it is important to note that the Beta hardware has the signal post-processing programmed into it, so when the signals were downloaded onto a PC, it was ready for comparison. The eDAQ hardware output is only the raw signal, so a certain amount of signal processing had to be done before the signals could be compared. The signal processing performed (be it on a PC afterward, in the case of the eDAQ or as part of the algorithm as in the case of the Beta prototype) was exactly the same and explained at length in Section 3 “Algorithm development”. Frequency domain results are easier to compare than time domain results and a sample is illustrated below. The graph represents samples that were measured both with the Beta prototype and the eDAQ system under similar circumstances of the same gearbox.

Comparing the spectra, one can see the different peaks correspond very well both in terms of peak amplitude and shape. Importantly, one can easily distinguish individual closely spaced peaks, such as the peak and sidebands around 24 Hz. Also visible in the figure below is that the relationship between the background noise level and the spectral peaks compares well in the two sets of data –

this is also a very important factor for the algorithm as the inference boundary values are based on this relationship.

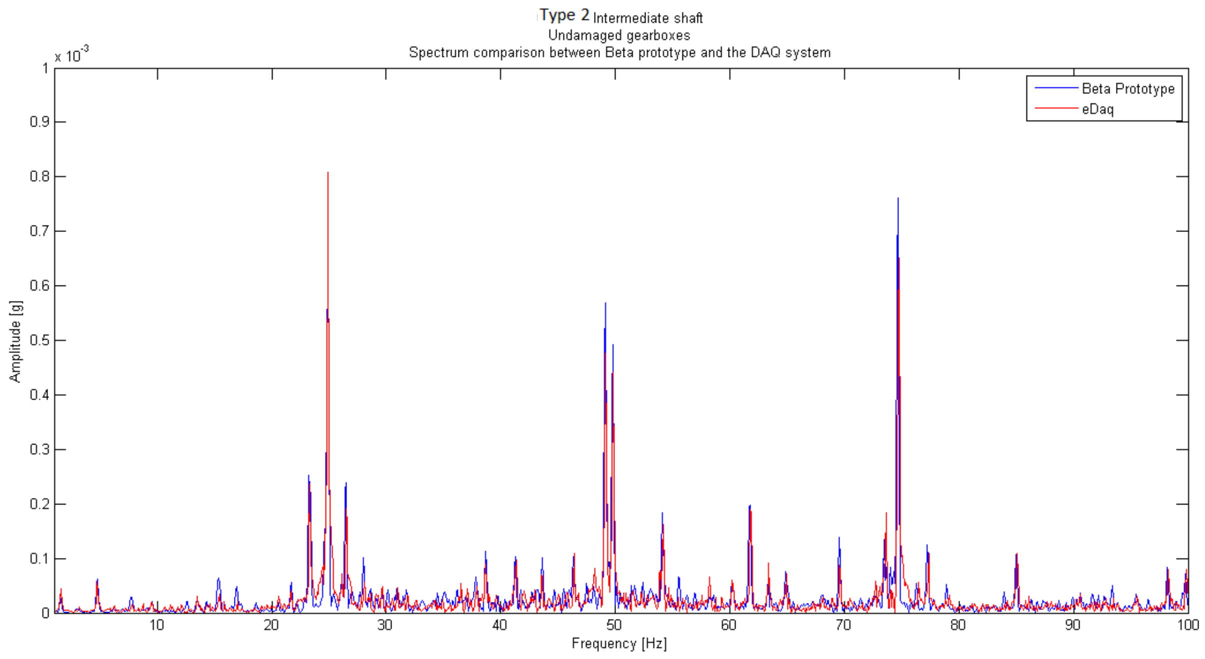


Figure 74: Comparative spectrum of the Beta prototype and eDAQ

It is now worthwhile to compare the spectrum in Figure 74 above to the spectrum of the Alpha prototype in Figure 73. Comparing these figures, the effect of the measures taken in the Section 2.4 "Advanced Development Model - Beta prototype" is taken and judged to be successful in significantly increasing signal quality.

6.4.2 Comparison between the Beta prototype and SASOL data

A comparison is now made between the results obtained by the Beta prototype and the commercial system used by the condition monitoring department of the Sasol Plant in Secunda. Regarding the system in use by SASOL (CSI AM/RBM Suite), after gathering the vibration data using a logger, the data is processed to the frequency domain where relevant data peaks are highlighted for inspection. Time domain plots and techniques are also available. Finally, the results of each gearbox are archived and can be arranged into a waterfall plot to observe the changes.

As mentioned, the suite plots the data in either time or frequency domain and assists the observer in identifying relevant information by overlaying markers where events are expected (such as peaks on a frequency spectrum, or impulses spaced at a period in the time domain plot). After a history of the machine in question is generated a judgment is then made based on changes in the data – be it time or frequency domain.

Figure 75 illustrates an example plot that compares the data from the Beta prototype to the system in use by SASOL. An important part of the spectrum is illustrated in which three prominent components are visible: The second and third harmonics of one of the gear sets as well as a signal coming from a damaged outer race of a bearing.

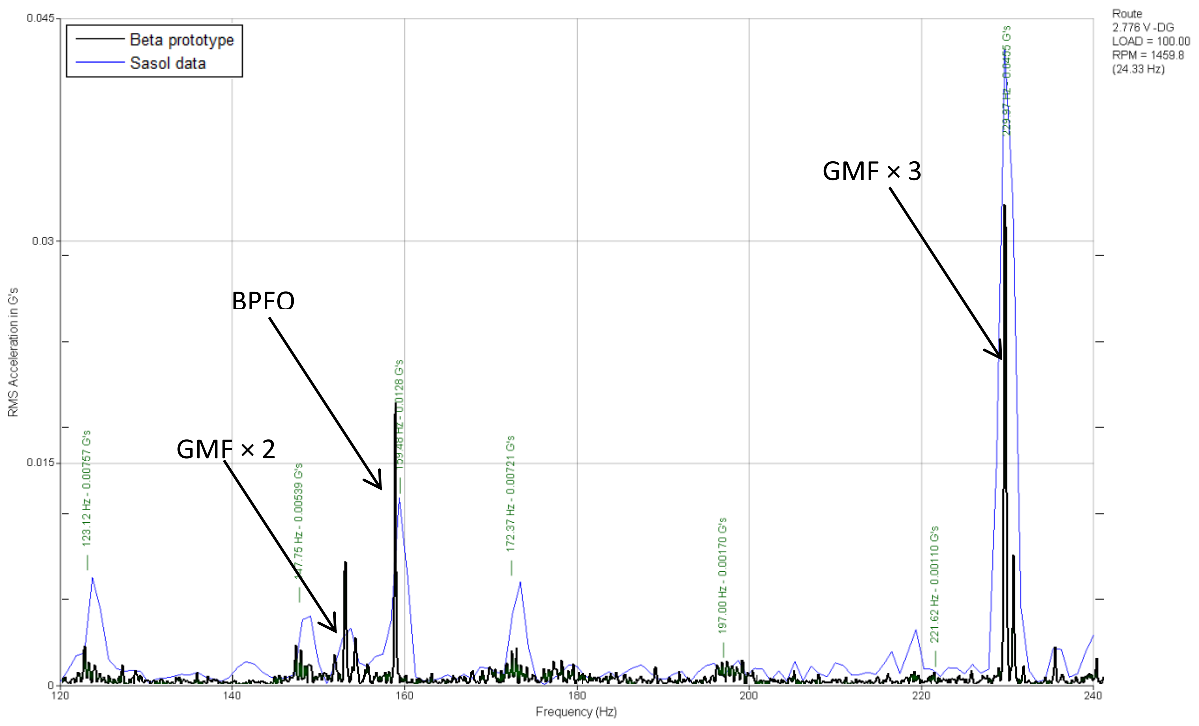


Figure 75: Overlaid broadband plot of the Beta prototype and SASOL data

In the figure, one can clearly see how the spectral components correspond very well in terms of frequency and amplitude. As the software package used by SASOL only has a somewhat coarser frequency resolution as produced by the Beta prototype, frequency components from the Beta prototype are significantly more distinguishable. Spectral clarity is further aided by the optimized window used by the Beta prototype that results in less leakage into adjacent bins as explained in section 1.3.3.

It is mentioned that as the results obtained from Sasol was scaled in RMS amplitude, the results from the Beta prototype had to be scaled to RMS as well in post processing. Even though this is not the normal operation of the Beta prototype and the scaling had to be done in post processing, the results are still considered to be valuable as only the magnitude of the spectrum is changed and not the shape.

6.4.3 Inference accuracy

With the signal integrity of the Beta prototype verified by comparing it to a commercial system as well as the system in use by SASOL, the accuracy of the inference can now be verified. During the Beta prototype test, four gearboxes were measured, totalling 26 different bearings of 8 different types. Two of the gearboxes had bearing problems large enough for SASOL to schedule replacement. The other two were not sufficiently damaged for SASOL to consider replacement.

6.4.3.1 Samples of damaged bearings

The two gearbox samples below are both from the same type of gearbox. A specific bearing on the input shaft of this type of gearbox was found to be susceptible to damage in this application. Both these gearboxes were identified by SASOL as requiring a new bearing on the input shaft.

The data presented here are the very spectra from the signals as measured by the Beta prototype and obtained from the saved data. And as can be seen in the frequency spectra below, both these bearings produced pronounced peaks at the outer race fault frequency (159 Hz on CV2203–1 and 161 Hz on CV3011; the slight difference is due to slightly different operating speeds).

In fact, the bearing on CV 2203 produced a frequency peak at the inner race fault frequency as well (which was flagged by the Beta prototype). It was found at a later stage that there was indeed a fault at the inner race even though at the time of measurement it was not known yet. An encouraging indication was that the Beta prototype detected this fault without prior knowledge.

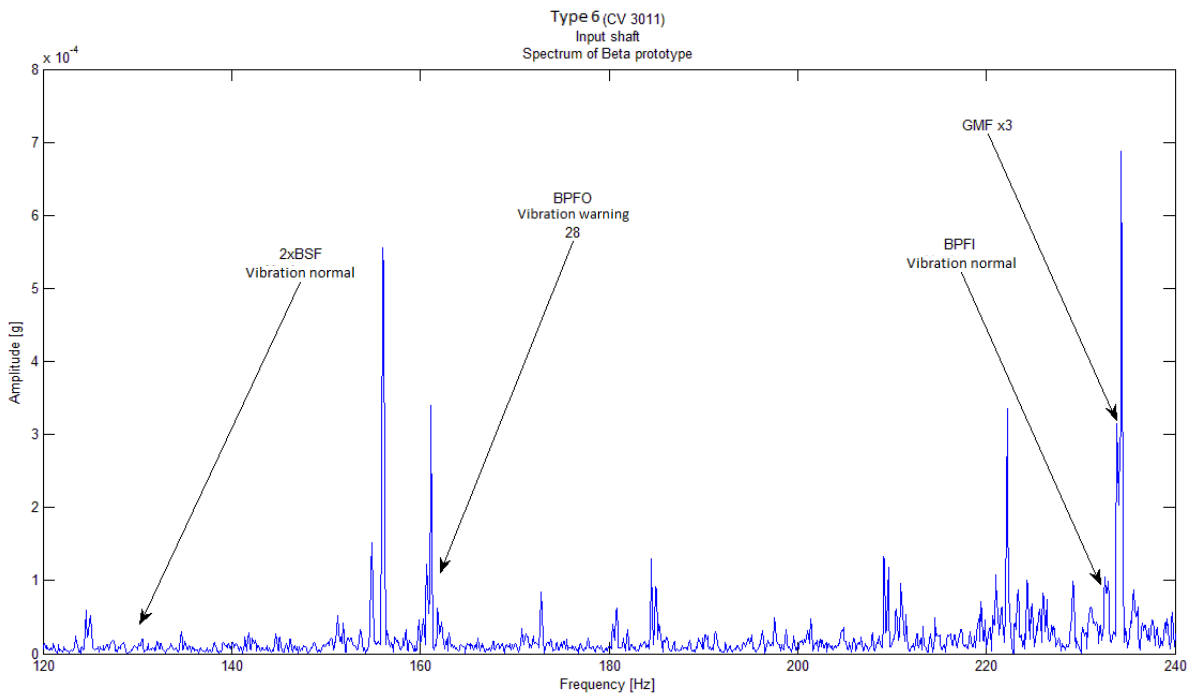


Figure 76: Frequency spectrum of input shaft of gearbox CV 3011 with damaged outer race

The spectrum above is from the input shaft of gearbox CV 3011. The three fault frequencies are included in this plot and are:

Table 24: Identified bearing frequency components of Figure 76

Signal component	Frequency [Hz]
BPFI	232.1 Hz
BPFO	161.9 Hz
2× BSF	131.2 Hz

A sharp peak is clearly distinguishable on the plot at 161.2 Hz, which was flagged by the Beta prototype as being possible outer race damage. The 28 in the caption indicates the peak-median ratio was 28, thus exceeding the 10× boundary and flagged as a 'Vibration Warning'. The other two bearing frequencies, BPFI and 2×BSF had ratios of less than 5 and were thus not flagged.

Table 25: Other frequencies identified in Figure 76

Signal component	Frequency [Hz]
2× $GMF_{2\text{-}3}$	156.3 Hz
9× Shaft 1	222.5 Hz
3× $GMF_{2\text{-}3}$	234.5 Hz

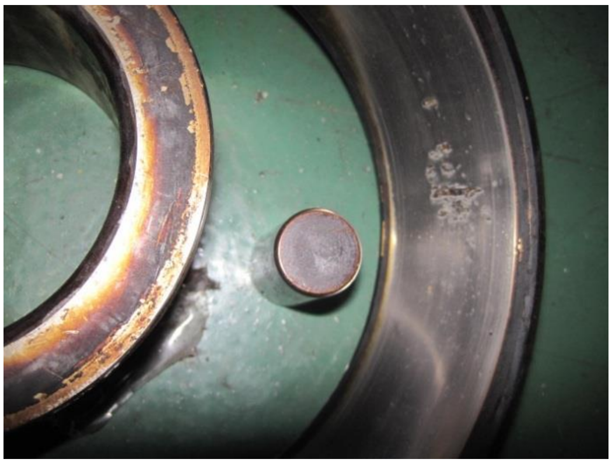


Figure 77: Inspected bearing with damage

The bearing in question was removed not long after the tests and Figure 77 shows the pitting on the outer race as detected up by the Beta prototype and as well as SASOL.

As mentioned, the gearbox under consideration is susceptible to damage of the input shaft bearings in this application. The next sample is from the same type of gearbox and it suffered the same type of bearing damage.

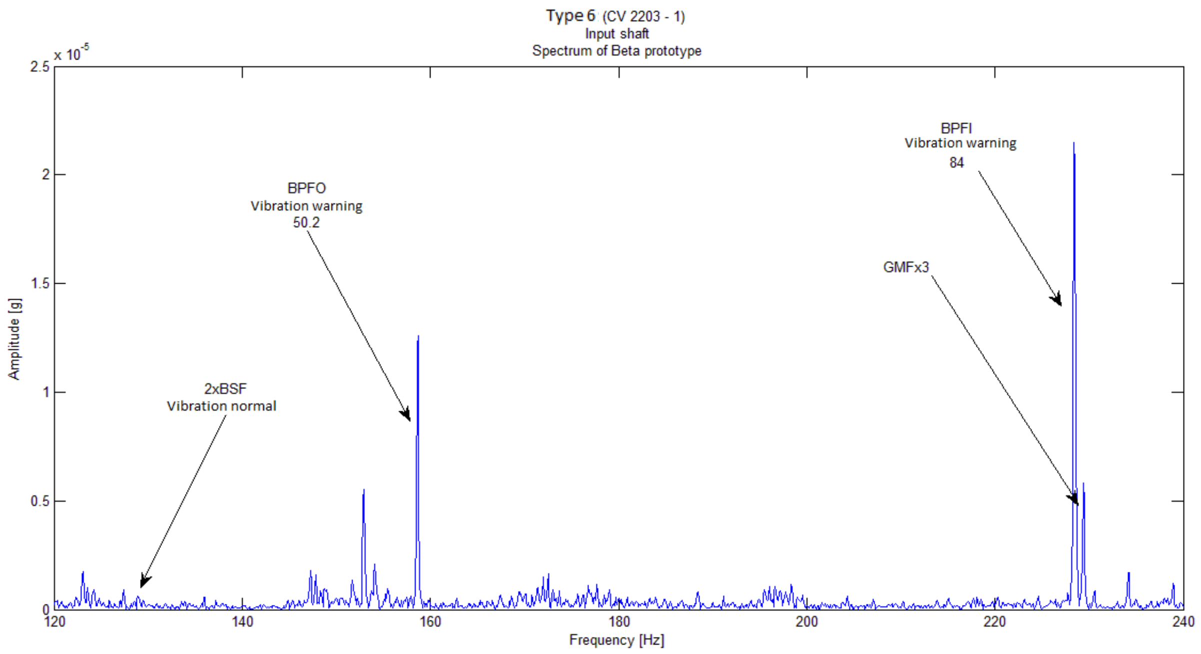


Figure 78: Frequency spectrum of input shaft of gearbox CV 2203-1 with damaged outer and inner race

Visible in the spectra of Figure 78, the same fault is present on the outer race as in the previous example and indicated by an arrow and BPFO caption. A peak-to-median ratio was obtained of 50.2 which far exceed the 10× threshold. In addition, a fault is present on the inner race as well. To the observer, this fault appears very close to the 3rd harmonic of the shaft 2-3 GMF. However, the Beta prototype successfully distinguished this peak from the GMF harmonic and flagged this fault as having severe damage as it had a peak-to-median ratio of 84.

6.4.3.2 Samples of undamaged bearings

A few samples will now be given of shafts with undamaged bearings (as confirmed by SASOL). These samples may come from the same types of gearboxes discussed already, but which illustrates how the signals from undamaged bearings look in the frequency domain and how the Beta prototype interpreted the signals.

The first sample under consideration is of gearbox CV 3301 – 2 at SASOL. The spectrum shown below is a two stage gearbox, implying that three shafts are present. The spectrum is of shaft 2 (the intermediate shaft). This shaft has two types of bearings, giving 6 potential fault frequencies. The spectrum was drawn over a range that covers all these frequencies, summarised in the table below (the numeral in the name differentiates the bearings)

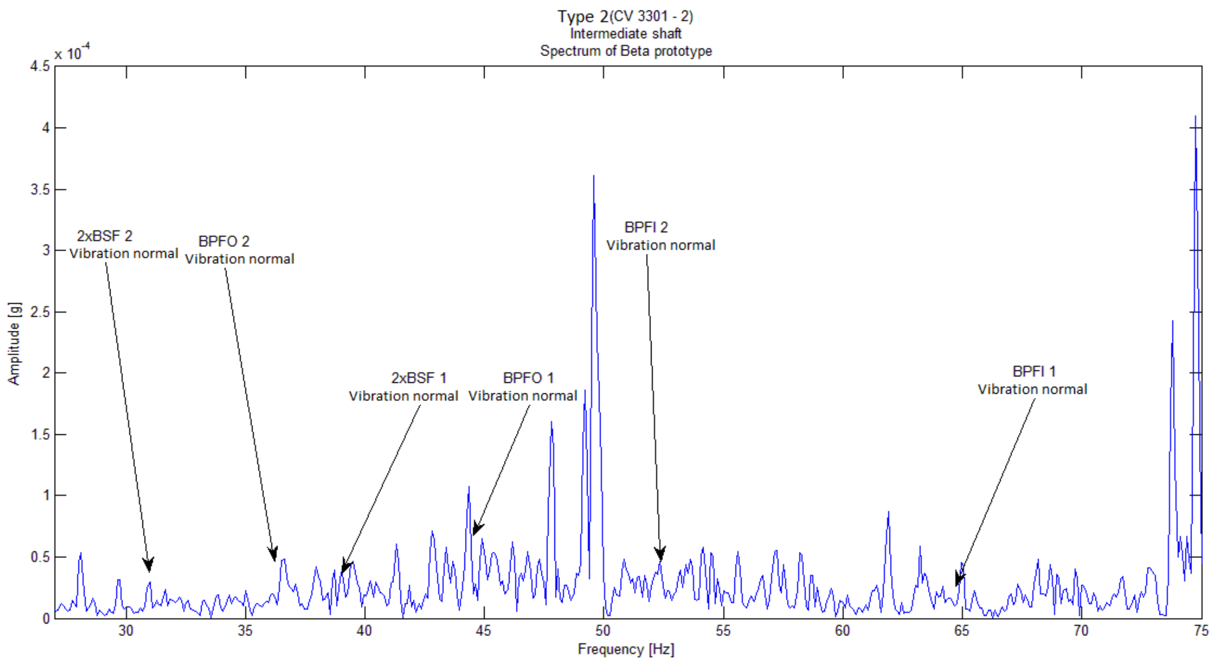


Figure 79: Spectrum illustrating healthy bearings on an intermediate shaft

Table 26: Identified frequencies in Figure 79

Signal component	Frequency [Hz]
BPFI 1	65.0 Hz
BPFO 1	44.4 Hz
2× BSF 1	39.0 Hz
BPFI 2	52.3 Hz
BPFO 2	36.5 Hz
2× BSF 2	31.0 Hz

As can be seen in the plot of Figure 79, none of these frequencies really created a distinct peak, apart from the BPFO 1 peak. However, this peak was not large enough to trigger a flag, as the surrounding activity decreased the peak-to-median ratio.

It is important to note, that the condition monitoring department at SASOL does not consider any of these bearings to be faulty, unlike the previous samples. This corresponds to the fact that no spectral activity was found by the Beta prototype in at these frequencies.

The next sample is of the output shaft of a SASOL gearbox. On this shaft is two bearings of the same type, therefore if one was damaged it is possible that one would find a peak in the frequency domain. These frequencies are marked in the spectrum of Figure 80 below (along with other nearby peaks) and the frequency values given in the table below.

Table 27: Spectral components of interest in Figure 80

Signal component	Frequency [Hz]
BPFI	15.4 Hz
BPFO	10.8 Hz
2× BSF	9.16 Hz

As with the previous example, the spectrum gives no indication of there being a bearing problem as there are no distinct peaks at the frequencies one would expect to see them. Once more, this is expected as SASOL reported no bearing faults on this shaft.

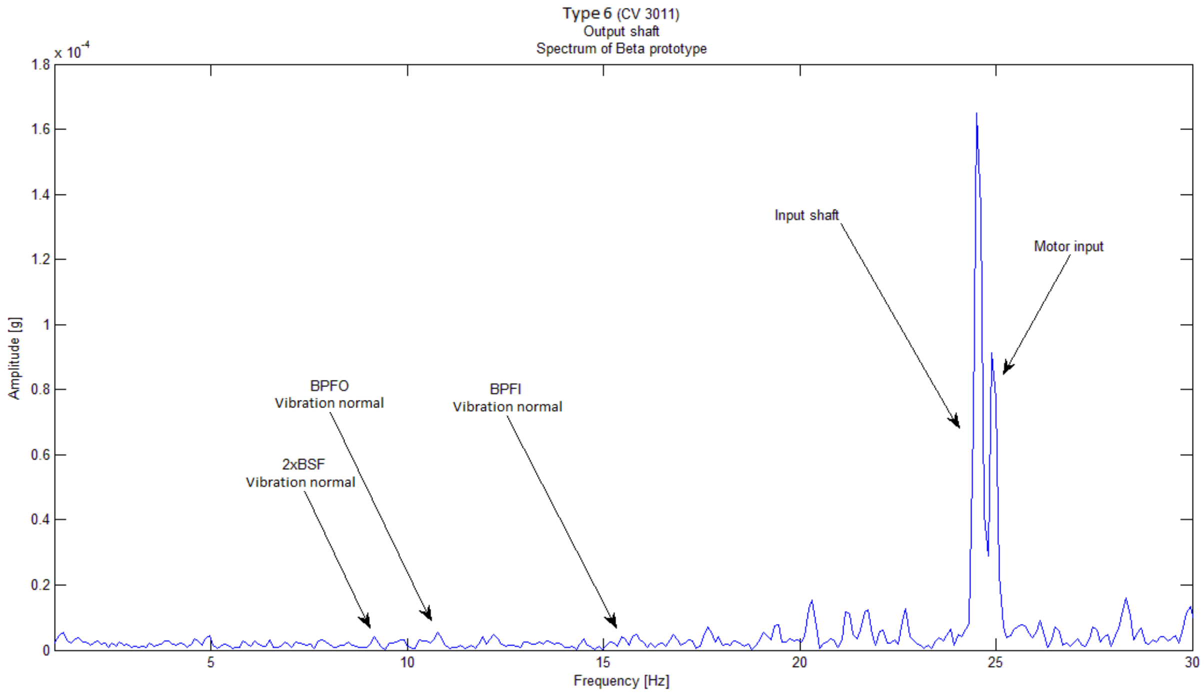


Figure 80: Spectrum illustrating healthy bearings on an output shaft

6.4.3.3 Misdiagnoses

On a number of occasions, it was found that the current methodology for evaluating the potential damage on bearings in a gearbox was conservative, i.e. flagging bearing damage when the bearing was not considered to be damaged by SASOL. An illustration of this happening is given in the spectrum below.

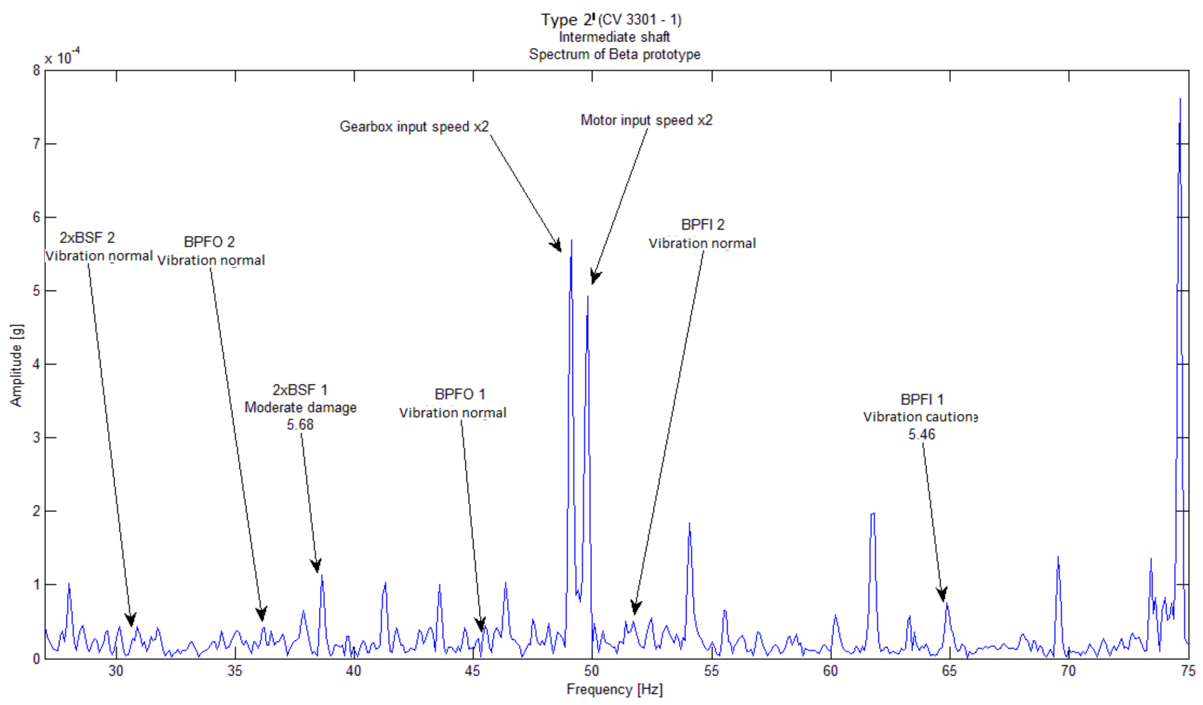


Figure 81: Spectrum of the first misdiagnosis example

Figure 81 illustrates a range in the frequency spectrum of the intermediate shaft where the fundamental frequencies are located of the two types of bearing present on the shaft. Using the arrows and captions, one can see that the Beta prototype flagged the 2×BSF and BPFI of the first bearing as 'Vibration caution', with a 5.68 and 5.46 peak-to-median ratio (exceeding the 5× limit for moderate damage).

However, SASOL advised that their monitoring program did not indicate any damage in the bearing and they consider it completely healthy, even though some peaks were distinguishable at the predicted frequencies. Therefore, in actuality no need for caution existed.

The most severe case in which a misdiagnosis occurred is illustrated in Figure 82 below. The first type of bearing on the shaft produced peaks on the outer race (BPFO) and inner race (BPFI) frequencies, in addition to the 2×BSF frequency coinciding with the shaft 2-3 GMF.

The BPFO frequency had a peak-to-median ratio of almost 16 and the BPFI had a ratio of about 18, but as before, SASOL indicated that it did not consider the bearings to be particularly damaged nor did they intend to replace the bearings soon.

It also has to be said that the 2×BSF frequency identified happens to be the third sideband (modulation by shaft 3; rotating at the 1.7Hz spacing frequency) of the shaft 2-3 GMF. Even though it did not flag this peak as being even moderately damaged (the surrounding spectral activity increased the median value enough to decrease the peak-to-median ratio), the frequency was still wrongly identified, as the algorithm does not include sidebands as exclusion criteria.

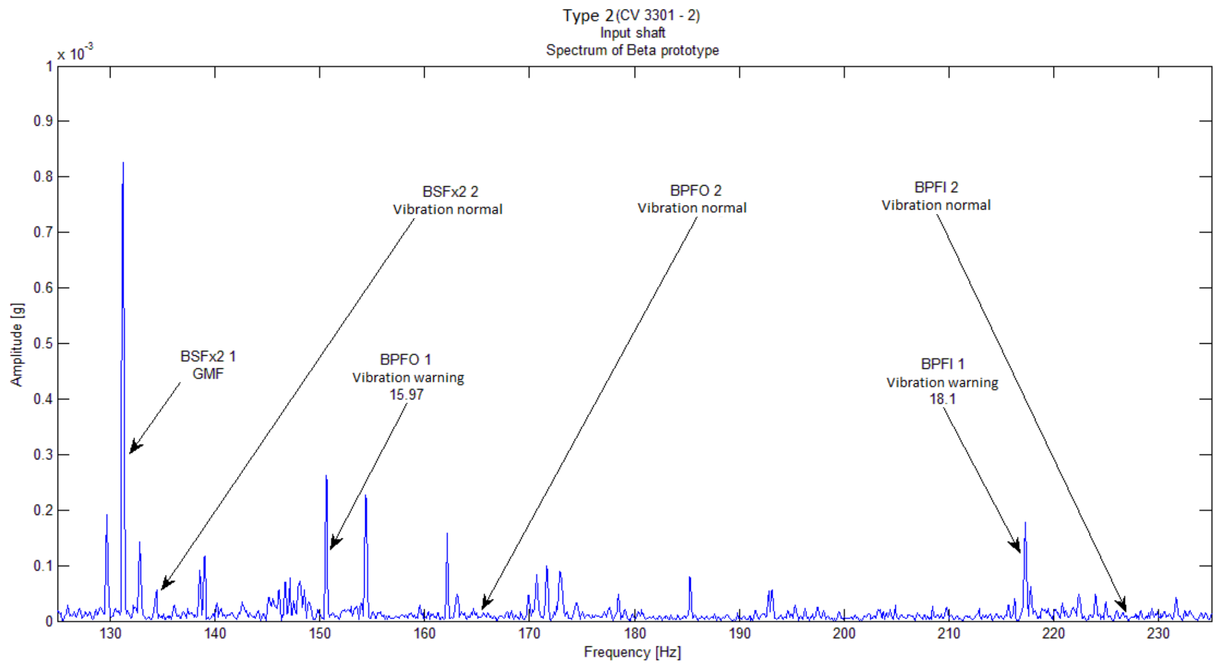


Figure 82: Spectrum of the second misdiagnosis example

6.4.3.4 Discussion

Based on the different bearings observed in the various gearboxes, it is concluded that the system is conservative, which is accordance with the projected behaviour of a 'Protection system'. Its behaviour is such that although it sometimes happens that the bearings are overly conservatively evaluated as having higher levels of vibration (implying higher risk of failure) than they actually have, it never declared a bearing as 'Vibration normal' when there was damage present. Based on the four gearboxes sampled, and taking every bearing's diagnoses as a sample, the following figures can be compiled regarding the inference accuracy:

Table 28: Inference accuracy figures

Parameter	Figure
Number of correct inferences	40
Number of incorrect inferences	8
Total number of cases	48
Success percentage	83.3%

This is figure of 83.3% is very similar to the 80.4% quoted the Section 6.2.4 where the algorithm was tested on the measured data of the first measurement exercise. This consistency is taken as a positive sign that translation of the algorithm from the Matlab environment, to the Alpha prototype and finally the Beta prototype was a success.

From the project philosophy point of view, the 83.3% success ratio is considered adequate as the system was not designed to replace human judgement, but to supplement it. As the system has proven to be conservative, this behaviour paves the way for the laymen wielding the finished product (the product being the commercial successor of the Beta prototype) to inspect his consignment of gearboxes. When the product flags a potential problem, he can hand over the data to an engineer or technician to make the judgement of whether to replace the bearing or to leave it for a while yet, safe in the knowledge that the system will not fail to catch a damaged bearing.

7. Conclusion

7.1 Cost analysis

The entire premise of the project was to develop a low cost condition monitoring device. A target budget of R5000 was set and a summary of the costs involved are given below. As is often the case, costs per part differ for the quantity of a part ordered. The target was to get the total cost of manufacturing one unit below the target budget, with any savings for bulk an added bonus. Where possible, the price for both 1 unit and 100 units are given, otherwise the price for 1 unit was used and the entry was underlined. All prices in ZAR are inclusive of VAT.

Table 29: Prices of components and manufacturing

Part/Task	Supplier	Price (1)	Price (100)
MCU	RS Components	136.52	66.88
Memory	Avnet	21.06	19.27
LCD	Nu Vision	96.95	<u>96.95</u>
Battery	Communica	78.50	<u>78.50</u>
Accelerometer (including cables)	Anderson and Hurley	1334.94	<u>1334.94</u>
Miscellaneous electronic components	Farnell	360.25	310.69
Casing	Routertec	877.00	237.00
PC Hardware (once off)	Mikroelektronika	647.26	0
High speed SD Card	Orms	499.99	<u>499.99</u>
Serial to USB converter	Communica	112.00	<u>112.00</u>
Circuit board printing	Deman Manufacturing	800 (approx)	300 (approx.)
Circuit board populating	Deman Manufacturing	90	18
Total		5054.47	3074.22

The price column for 1 unit lists all of the materials and manufacturing costs in addition to auxiliary costs such as PC hardware (once off), casing design and die manufacturing (incorporated into the Casing category) and the initial programming and setup of the circuit board printing and populating machinery. Therefore, it includes the initial investment required for production.

Contrary to this, the price column for a 100 units assumes the necessary initial investments (as described) have been made, so these items are not included. Furthermore, this price column includes all manufacturer and dealer discounts available to large orders.

As the project budget is R5000, the additional funding available if production quantities are made (100 units or more), will be well spent on a higher quality accelerometer. In addition, a better case design (such as an IP66 compliant case) may also be considered.

7.2 Project review

The aim of this project was to develop a low cost vibration protection device.

An initial data gathering campaign was therefore undertaken to characterise a variety of gearboxes in the industry. During this excursion, several different types of gearboxes were measured in various states of health.

Considering the information gleaned from the measured data, a literature study was undertaken during which typical signal processing techniques were investigated and considered. A fairly simple frequency domain technique (developed in the robust Matlab environment) was chosen as the most promising avenue of work. This is in line with the market positioning for a low-cost protection system, which does not give precise diagnoses, which require a powerful processor, but recommendations on a course of action. The technique employed was based on the results from the initial data gather campaign and evaluated against it.

In parallel with this, the XDM was acquired. This allowed familiarisation with the embedded environment as well as serving as a vessel for translating and testing the developed algorithm from Matlab to ANSI C. This was however the scope of usefulness for this system, as it was very large in dimensions, could not store a large amount of data and unable to gather analogue signals.

This was corrected with the Alpha prototype of the ADM, this was a custom designed system capable of holding more data (in volatile memory) and measuring an analogue signal. It was however still very slow and lost the data when switched off. It was tested at the SASOL plant and found to be slow and it is data of poor quality. It also had to be connected to a PC and was thus not an independent system.

The definitive hardware specification was of the Beta prototype of the ADM. This device is considered to be pre-production and incorporated many improvements to aid signal quality as well as incorporating an LCD, pushbuttons, a user interface, non-volatile memory, future expandability and increased speed.

The algorithm proved to be fairly accurate and successfully made a large percentage of correct inferences. However it made several misdiagnoses and is considered to be a priority improvement. The system also remains to be tested outside of SASOL.

A requirement of the project was for the final device to cost R5 000 or less. The final system cost depends on how many devices are manufactured. For a single device, the cost is R 5054.47. This is marginally more than the target budget, but includes a R1 287.26 once-off start-up cost. For a hundred units, the cost is much less, at R3 074.22. The surplus is recommended for a better accelerometer, faster SD Card or an IP66 case.

The goals of this project is therefore deemed to be met as the device is small enough to be operated by hand or mounted onto a machine. It is capable of performing the necessary procedures to fulfil the role of vibration protection device. This includes:

- Acquiring and archiving data from various different gearboxes
- Applying rudimentary techniques to obtain a conservative state of the bearings within the gearbox
- Notifying the operator of the appropriate course of action based on the findings
- Storing a history of the time and frequency domain of the various gearboxes as well as a full test log of every test for later evaluation by experts

This is considered to fulfil the requirements for the current interpretation of the vibration protection device.

7.3 Summary of findings

7.3.1 Experimental development model

The experimental development model was never intended for either field or laboratory testing. Its sole purpose was to serve as a test bed having the same family of processor and using the same compiler.

It was therefore mainly used to translate the developed algorithms from the Matlab programming language to the C programming language. Further use was excluded due to limitations on its ADC module and very limited memory.

It was used primarily in the time during which the Alpha prototype was being manufactured.

7.3.2 Advanced development model – Alpha prototype

The advanced development model was the first custom designed hardware for the project. It contained many improvements to the hardware and addressed the ADC and memory limitations of the XDM. It also incorporated an LCD screen, although it never worked. Importantly, it used a higher specification of processor.

It performed reasonably well in laboratory and field tests, but several limitations existed. Data could not be downloaded onto a PC, the system had to be connected to a controlling PC at all times (due to the LCD not working), it was very slow in its operation and it had an un-ergonomic power supply that made it easy to forget on – thus depleting the battery.

The measurement results indicated a severe lack of spectral resolution. The root cause of this was found to be a sub-optimal window function, very high sampling rate (in conjunction with finite memory) and a problematic memory chip.

The prototype found a second use during the time in which the Beta prototype was being used. During this time, some modifications made to the hardware allowed the development of drivers which was used in the Beta prototype.

7.3.3 Advanced development model – Beta prototype

The Beta prototype was the final specification hardware of this project and included a number of further improvements upon the Alpha prototype in addition to correcting its flaws.

The changes between the Alpha and Beta systems involved the use of the PLL (which multiplied the operating speed by 8), a working LCD screen, push-buttons (providing a user interface and independence from a PC), RS232 connectivity, the use of an SD card for non-volatile storage, on-board accelerometer power supply and a rechargeable battery.

In addition, many algorithmic improvements were made during the transition which included a much optimised FFT which better utilises the memory in order to provide twice the number of samples. A study was conducted about window functions, and an optimised window function was chosen.

Field tests confirmed a vastly improved frequency spectrum. The results were compared to samples from SASOL in Secunda and the Beta prototype yielded better spectral resolution. Regarding the inference accuracy, more 80% of the inferences were deemed to be correct. This was deemed to be a success.

7.4 Recommendations

With the final deliverable of the project being a prototype, there are several aspects of the hardware that can be improved to make a better product. These items are listed below.

7.4.1 Input speed and calibration amplitude inclusion to the param.txt file

Currently, the SD card only serves to house gearbox design data. This includes number of gear teeth, number of stages and bearing frequency. The inclusion of the input speed will make for a system that is more modular.

The reason for its present omission is due to the fact that all the gearboxes measured in the industry operated at approximately the same input speed. However in practice this will obviously not be the case and a more robust scheme to change this parameter will be required.

It is noted that the firmware currently works with the input frequency. This may be somewhat confusing to operators used to working with input speeds in units of RPM. As there is a simple ratio between input frequency (in units of Hertz) and input speed (in units of Rotations per minute) it is logical to have the operator use the input speed and let the firmware do the conversion internally, as this would eliminate the possibility of an error.

The calibration amplitude would serve the envisioned auto calibrate routine described below.

7.4.2 Auto calibrate routine

As has been mentioned in Section 4.3 where the calibration procedure is explained, there are several factors that can cause erroneous variances in the amplitude logged by the device. For this reason, it is considered prudent to include an auto calibrate routine in the firmware to take care of this and reduce the amplitude error that may exists from machine to machine.

This routine may comprise of a simple option that exists in the menu of the device that would initiate the auto calibrate routine in the device. The routine would follow the same procedure as outlined in Section 4.3. When this routine is initialised, the operator would have to measure a signal of known amplitude (included in the param.txt file). The routine would then compare the internal amplitude logged by the device with the known calibration amplitude and correct for it with a calibration factor.

7.4.3 Gear and shaft fault finding

Currently, the firmware on the Beta prototype only looks for possible bearing problems. The scope of the project was intentionally limited to these components due to the fact that bearing problems are the most common on gearboxes found at SASOL. Furthermore, being a prototype it was deemed sufficient to prove the validity of the system.

However, bearing defects are of course not the only defect type that exists in a gearbox. Shaft and gear defects are also found on gearboxes and can be found using vibration based techniques (Bloch & Geitner, 1999). In the industry, these problems will arise from time to time and it would add immense value to the product if these defects can be detected, especially since the hardware infrastructure already exists for this.

7.4.4 Spectral averaging, higher quality SD card and improved SD card routines

This is another common vibration monitoring technique that would add value to the system as it would likely suppress stray peaks in the frequency spectrum and generally smooth the noise level (it is important to note however that the technique does not reduce spectral noise, but merely averages it).

Currently the hardware itself is not capable of storing enough data to store more than one frequency spectrum, unless the bandwidth is reduced. The SD card can however store this data, as when it saves the spectra for further analysis.

However, this procedure is currently taking too long (in the order of a minute and even then it only saves up to 500 Hz). However, it is readily admitted that the SD card routines have much potential for optimization.

In addition, the SD card in use is not of the highest quality and at higher writing speeds becomes unstable. The use of a higher quality, higher speed SD-card would sever to improve the writing speeds.

Improving the SD card write speeds, through the use of optimized SD card routines and a higher quality SD card, will make spectral averaging feasible by temporarily storing successive spectra on the SD card and the averaging them. This will likely improve the damage prediction performance and ease third party consultation on the data as the spectra will appear ‘cleaner’.

7.4.5 Algorithm calibration

Another important aspect task is calibration of the algorithm. The current peak to median ratios of 5× and 10× in the frequency spectrum was effective enough to prove the concept and be effective at SASOL, but further data will be needed to verify if these figures are appropriate for other sites and applications as well.

This can only be achieved by putting further models in the field and monitoring their performance. Fortunately, the coding infrastructure is already in place, in the form of the log file, to prepare for this exercise.

7.4.6 Signal interference compensation

It was mentioned in the Section 3.2.7 “Algorithm development – Exclusion criteria” that when there is interference (in the form of frequency coincidence) that the algorithm ignores this spectral component.

This is creates a vulnerability in the system, as that component may well be damaged, even though this damage is masked by the interfering component. Further research is necessary to minimise this problem, but a scheme in which the system looks to time domain parameters such as maximum value for a backup plan.

It is realised that whatever scheme is put in place will likely be less sensitive to problems, but this is considered to be better than nothing. In addition, the system fortunately incorporates the functionality to save data, which means third party consultants or our Industrial partner can offer assistance in interpreting the data.

References

Avigad, J., & Donnelly, K. (2004). Formalizing O notation in Isabelle/HOL. Basin and Rusinowitch: Springer Verlag.

Balderston, H. l. (1969). The detection of incipient failure in bearings. *Material Evaluation*(27), 121-128.

Bartelmus, W. (2008, April 4). Root cause and vibration signal analysis for gearbox condition monitoring. *BINDT Insight Journal, 50*(4).

Bartelmus, W., & Zimroz, R. (2009). Gearbox condition degradation models. *The sixth International Conference on Condition Monitoring and Machinery Failure Prevention Technologies*, (pp. 1-10). Dublin.

Bloch, H., & Geitner, H. (1999). *Machinery Failure Diagnosis and Troubleshooting* (4th Edition ed.).

Bores Signal Processing. (2009, 06 02). *FFT window functions: Limits on FFT analysis.* Retrieved 10 10, 2010, from Bores signal processing: http://www.bores.com/courses/advanced/windows/files/windows.pdf

Bores Signal Processing. (2009, 2 6). *FFT windows: Coherent power gain*. Retrieved 04 17, 2011, from Advanced DSP: http://www.bores.com/courses/advanced/windows/10_cpg.htm

Combet, F., & Gelman, L. (2007). An automated methodology for performing time synchronous averaging of a gearbox signal without speed sensor. *Mechanical systesms and signal processing, 21*, 2590-2606.

Combet, F., & Zimroz, R. (2009). A new method for the estimation of the instantanous speed relative fluctuation in a vibration signal based on the short time scale transform. *Mechanical Systems and Signal Processing, 23*, 1382-1397.

DLIengineering. (2010). Review of Techniques for Bearings & Gearbox Diagnostics. Richmond, Virginia: SpectraQuest, Inc. Retrieved 03 21, 2010, from http://www.dliengineering.com/vibman/gloss_bearingtones1.htm

Ganeriwala, S. (2010). Review of Techniques for Bearings and Gearbox Diagnosis. *IMAC Conference*, (pp. 1-37). Jacksonville FL.

Girdhar, P., & Scheffer, C. (2004). *Practical Machinery Vibration Analysis and Predictive Maintenance* (1st ed.). Oxford: Elsevier.

Graham, R. L., Knuth, D. E., & Patashnik, O. (1994). *Concrete Mathematics: A foundation for computer science* (2 ed.). Reading: Addison-Wesley Publishing Company.

Grover, D., & Vollmer, M. (2010). *Fast Fourier Transform (FFT) FAQ*. Retrieved March 17, 2010, from http://www.dspguru.com/dsp/faqs/fft

Harris, F. J. (1978). On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE*, *66(1)*, pp. 51-84.

Heng, A., Zhang, S., Tan, A. C., & Mathew, J. (2009). Rotating machinery prognostics: State of the art, challenges and opportunities. *Mechanical Systems and Signal Processing*(23), 724-739.

Hochmann, D., & Sadok, M. (2004). Theory of Synchronous Averaging. *IEEE Aerospace conference proceedings.* Vergennes, USA: Aerospace Conference.

Ifeachor, E. C., & Jervis, B. W. (1998). *Digital Signal Processing: A practical approac.* Harlow: Addison Wesley.

Jardine, A. K., Lin, D., & Banjevic, D. (2006). A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*(20), 1483-1510.

Karacay, T., & Nizami, A. (2009). Experimental diagnostics of ball bearings using statistical and spectral methods. *Tribology Internation*(42), 836-843.

Kardushin, D., 1991. *3rd International Machinery & Diagnostics Conference.* Las Vegas, Union College.

Kiral, Z., & Hira, K. (2003). Simulation and analysis of vibration signals generated by rolling element bearing with defects. *Tribology International*(36), 667-678.

Kiral, Z., & Karagülle, H. (2003). Simulation and analysis of vibration signals generated by rolling element bearing with defects. *Tribology International*(36), 667 - 678.

Kochan, S. G. (1988). *Programming in ANSI C* (1st ed.). Indianapolis: Hayden Books.

Konstantin-Hansen, H. (2003). Application Note 7773: Envelope Analysis for Diagnostics of Local Faults in Rolling Element Bearings. *Brüel & Kjær Application note*, pp. 1-8.

Lai, E. (2004). *Practical Digital Signal Processing for Engineers and Technicians.* Oxford: Newnes.

Ma, J., & Jiang, J. (2011). Applications of fault detection and diagnosis methods in nuclear power plants: A review. *Progress in Nuclear Energy*(53), 255-266.

Mathew, J., & Alfredson, R. J. (1984). The condition monitoring of rolling element bearings using vibration analysis. *ASME Transactions - Journal of Vibration, Acoustics, Stress and Reliability in Design*(106), 447-453.

Mathworks. (n.d.). *Minimum 4-term Blackman-Harris window - MATLAB*. Retrieved 05 03, 2011, from Mathworks: http://www.mathworks.com/help/toolbox/signal/blackmanharris.html

McInemy, S. A., & Dai, Y. (2003). Basic Vibration Signal Processing for Bearing Fault Detection. *IEEE Transactions on Education, 46*(1), 149 - 156.

Microchip. (2010). *Microchip PIC product page*. Retrieved February 13, 2010, from http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2551

Microchip Technology Inc. (2008, January 22). *AN1152: Achieving Higher ADC Resolution Using Oversampling.* Retrieved from Microchip application notes:

http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1824&appnote=en533730

Microchip Technology Inc. (2009). *dsPIC33FJXXXMCX06/X08/X10 Data Sheet.* United States.

MikroC. (2010). FFT Library help file. *MikroC Pro for dsPIC30/33 and PIC24 Help*. MikroC.

Mobley, R. K. (2004). *Maintenance Fundamentals* (2nd ed.). Burlington: Elsevier.

National Instruments. (2009, 06 08). *he Fundamentals of FFT-Based Signal Analysis and Measurement in LabVIEW and LabWindows/CVI*. Retrieved 04 17, 2011, from National Instruments Developer Zone: http://zone.ni.com/devzone/cda/tut/p/id/4278

Nelwomondo, F. V., Marwala, T., & Mahola, U. (2006). Early classification of bearing faults using hidden Markov models, mel-frequency cepstral coefficients and fractals. *International Journal of Innovative Computing, Information and Control, 2*(6), 1281-1299.

Norton, M., & Karczub, D. (2003). *Fundamentals of noise and analysis for Engineers* (2nd ed.). Cambridge: Cambridge university press.

Patil, M. S., Mathew, J., Rajendrakumar, P. K., & Karade, S. (2010). Experimental studies using response surface methodology for condition monitoring of ball bearings. *Journal of Tribology, 132*, 44505-44511.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1995). *Numerical Recipes in C: "The art of Scientific Computing".* Cambridge: Press Syndicate of the University of Cambridge.

Radoslaw, Z., Urbanek, J., Barszcz, T., Bartelmus, W., Millioz, F., & Martin, N. (2011). Measurement of instantaneous shaft speed by advanced vibrtation signal processing - application to wind turbine gearbox. *Metrology and measurement systems*, 505-722.

Randall, R. B., & Antoni, J. (2011). Rolling elementbearingdiagnostics—A tutorial. *Mechanical Systems and Signal Processing*(25), 485–520.

Samual, P. D., & Pines, D. J. (2005). A review of vibration-based techniques for helicopter transmission diagnostics. *Journal of Sound and Vibration*(282), 475–508.

SKF. (n.d.). *Bearing frequencies*. Retrieved 6 6, 2009, from http://www.skf.com/skf/productcatalogue/calculationsFilter?lang=en&action=Calc6

Skolnick, D., & Levine, N. (1997, December 1). *Why Use DSP?: An introductory course in DSP system design*. Retrieved January 17, 2010, from http://svconline.com/mag/avinstall_why_dsp_introductory_2/

Smith, S. W. (1998). *The Scientist and Engineer's Guide to Digital Signal Processing.* Retrieved 04 20, 2011, from http://www.dspguide.com

Stander, C. J., & Heyns, P. S. (2002). Using vibration monitoring for local fault detection on gears operating under fluctuating load conditions. *Mechanical systems and signal processing, 16*(6), 1005-1024.

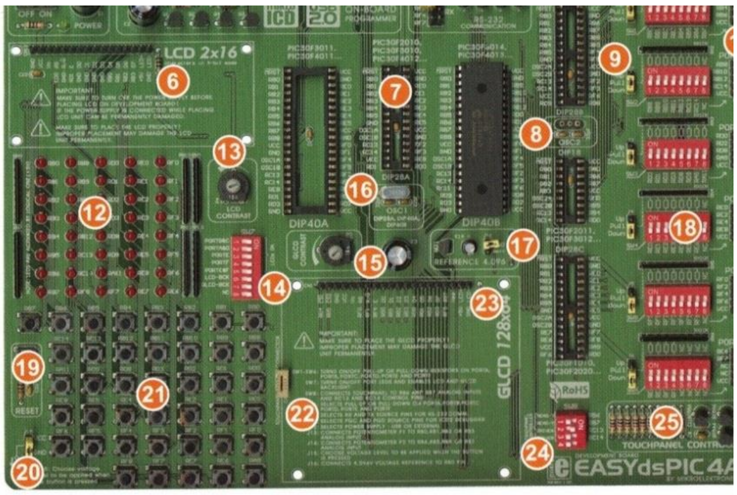
Tandon, N., & Choudhury, A. (1999). A review of vibration and acoustic measurement methods for the detection of defects in rolling element bearings. *Tribology International*(32), 469–480.

Taylor, J. I., & Kirkland, D. W. (2004). *The Bearing analysis handbook: A practical guide for solving vibration problems in bearings.* Vibration Consultants Inc.

von Hippel, P. T. (2005). Mean, Median, and Skew: Correcting a Textbook Rule. *Journal of Statistics Education, 13*(2).

Wilmshurst, T. (2007). *Designing Embedded Systems with PIC Microcontrollers: Principles and applications.* Oxford: Newnes.

APPENDIX A – XDM evaluation board



1. External power supply of 8v to 16v AC/DC;
2. On-Board USB 2.0 programmer with MikroICD (In-Circuit debuger);
3. RS-232 communication port;
4. ICD2 external programmer connector;
5. A/D converter test input potentiometers;
6. 2X16 character LCD display in 4-bit mode conector;
7. EASYdsPIC4A supports microcontrollers in DIP18, DIP28, and DIP40 packages;
8. OSC2 oscillator connector;
9. Jumpers to determine input pin performance in idle state (connected to pull-up/pull-down resistor);
10. Resistor network 8x10K;
11. Direct port access connectors;
12. Each I/O pin corresponds to one LED;
13. LCD contrast potentiometar;
14. Switch group SW7 allowing all LEDs on ports A,B,C,D,E and F to be connected or disconnected from MCU pins. Switches 6 and 7 of the same group enable LCD and GLCD backlight.;
15. GLCD contrast potentiometer;
16. OSC1 crystal (10Mhz);
17. Reference voltage source 4.096V;
18. Switch groups SW1-SW6 enabling pull-up/pull-dowr resistors on port pins;
19. RESET push-button;
20. Jumper J15 is used to select high or low state of pin any button press;
21. 41 push-buttons allowing control of all microcontroller p
22. CN11 touch panel connector;
23. Graphic LCD display (GLCD) connector;
24. Touch panel switch (SW8) enabling/disabling connec between touch panel and microcontroller; and
25. Touch panel controller;

MikroElektronika SOFTWARE AND HARDWARE SOLUTIONS FOR EMBEDDED WORLD

Figure A1: Data sheet of XDM evaluation board

APPENDIX B – ADM Beta prototype hardware specification

Table B1: Hardware specifications of ADM Beta prototype

Parameter	Value	Parameter	Value
Sensitivity (± 20%)	100 mV/g	Storage	2 GB (exchangeable)
Measurement range	± 16g	Package	SD Card
Frequency range	0.95 Hz to 3250 Hz	File type	.txt
Broadband resolution	8 mg	Possible number of records	> 16 million
Non - linearity	±1%	Stored records	Time and Frequency
Transverse sensitivity	≤ 7%		
		Expandability	• RS 232 • I2C • SPI
Size	200 ×120×80		
Weight	510 g		
Battery	6 V (3200 mAh)	LCD	128 × 64
Battery runtime (typical use)	3 days	Selection buttons	3
Charge time	≈ 8 hours	MCU speed	80 MHz
Outlet type	220 V (AC) – kettle connector		
Temperature range	-40°C to 150°C		

APPENDIX C – Electronic fundamentals and configuration

The objective of this section is to provide a brief overview of key electronic principles required for the operation of the electronics in this project, followed by a data flow and explanation of how the system internally operates. Finally in this section the physical electronic design is provided as well as the layout of the system on the Printed Circuit Board (PCB).

C.1 Microcontroller I/O ports primer

Microcontrollers (and indeed the related microprocessors) have a basic necessity that requires them to be useful in practical applications: input and output (I/O) interfaces (Wilmshurst, 2007). These so called I/O ports, provides the device with two way communications to the outside world – often, but not always, by means of integrated circuits (ICs) connected to these ports. Examples of devices typically connected to the ports include:

1. Buttons
2. Rotary dials
3. ADC units
4. Displays (seven segment, LCD)
5. Speakers
6. GPRS units
7. SD Card reader
8. Communication ports (USB, RS232)

As can be seen, some of these devices are for input into the system (the first three items listed), some are for output (items 4 and 5) and some are for two way communication (items 6 to 8). The common denominators of these units are that they use the I/O ports MCU. It is true that some of them use some intermediary circuitry, but in principle, all the signals from peripherals go through the I/O ports.

These ports manifest themselves as a specific series of pins on the MCU device (note however that not all pins are for I/O purposes, and some have a dual purpose) that is connected to the data bus (a figuratively parallel series of data lines from one point to another) of the MCU. Several pins are grouped together to form a port and can be used as a group or individually. A typical MCU has several of these ports. Figure C1 is an illustration of a basic MCU with the I/O ports highlighted.

Information on the pins manifests themselves as voltages. Typically, a voltage on a pin may be 3.3V – which will register as 1, or 0V (ground) which will register as a 0. Depending on whether the pin is configured as an input to the MCU or output to the MCU will determine whether the MCU reads the voltage or sets the voltage. If the pin is configured as an input, the peripheral (such as a button) will cause the pin to reflect a voltage, which is read by the MCU and used. Conversely, if the pin is configured as an output, the MCU will set the voltage, which will be read by the peripheral (such as an SD Card reader).

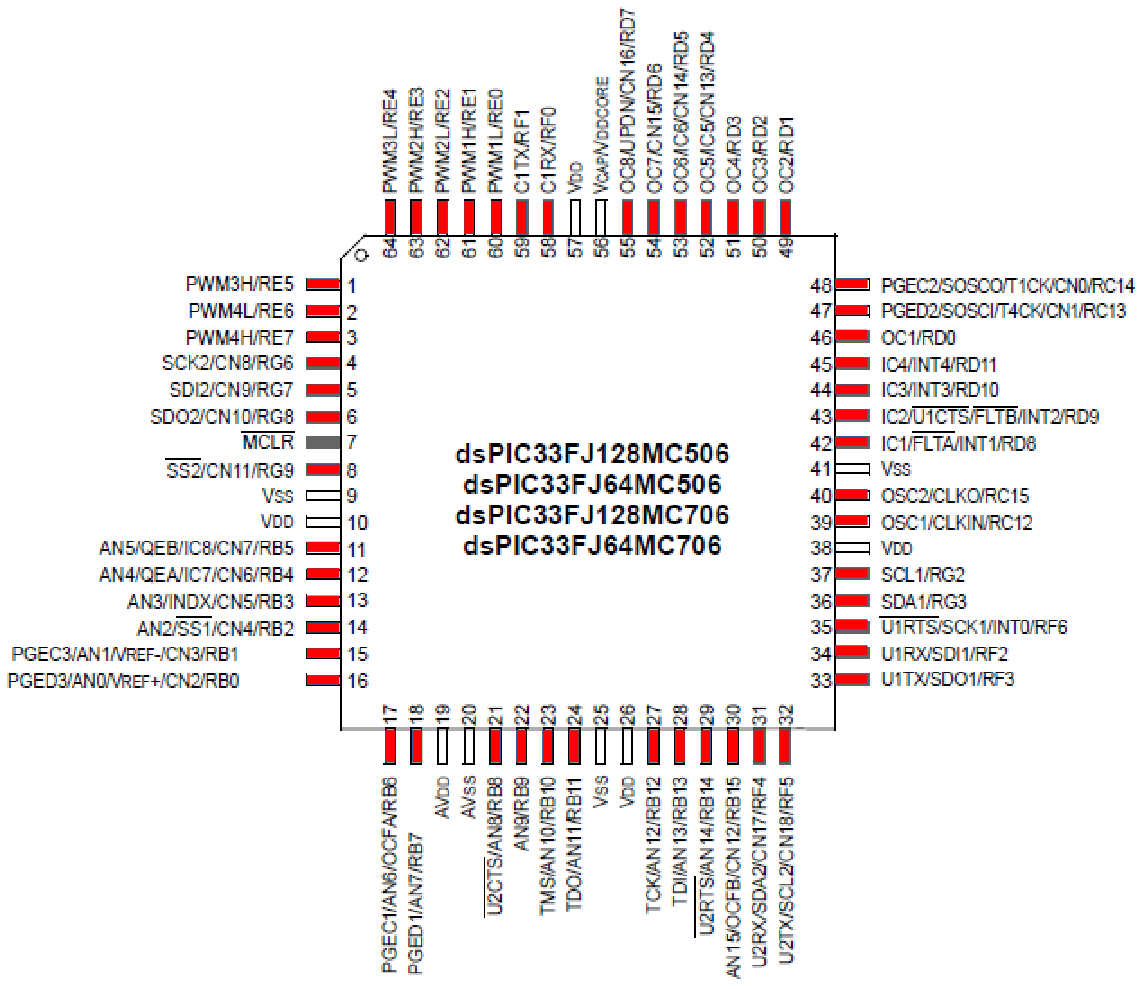


Figure C1: A basic MCU of the Mircrochip 16-bit family (Microchip Technology Inc, 2009)

These ports are useful as peripheral interfaces often require a series of pins that make up, say, a byte. Consider a hypothetical IC peripheral and the MCU below. The 4 pins of the port are all configured as input, meaning that the MCU will read the information from the pins (in the form of a voltage set by the peripheral). These pins are connected to four pins of the peripheral.

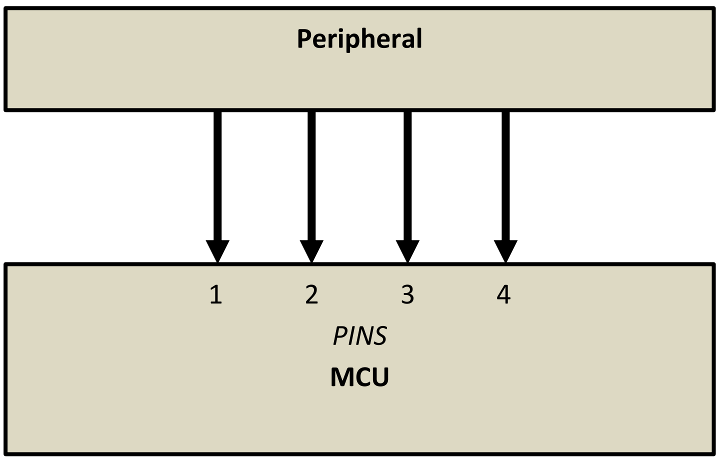


Figure C2: Data flow via pins

Table 31 illustrates how a number is passed in a parallel pin configuration from the peripheral to the MCU. As the pins are configured to be inputs to the MCU, the peripheral has control to set the voltages.

Table C1: Example of how parallel pin voltages relate to information

Pin number	Voltage [V]	Logic
1	3.3	1
2	3.3	1
3	0	0
4	3.3	1
	Binary value passed	13

If the peripheral (say, an ADC) sets the voltages as follows, the 4 pins can be combined into a port to recreate a 4 bit binary number – in this case 13. This number can now be read from port and used in the MCU for computation

It is emphasised that the pins can still be used separately. This does pose the restriction that only two values can be read or written – this is however a common requirement, such as with switches.

C.2 Circuit flow diagram

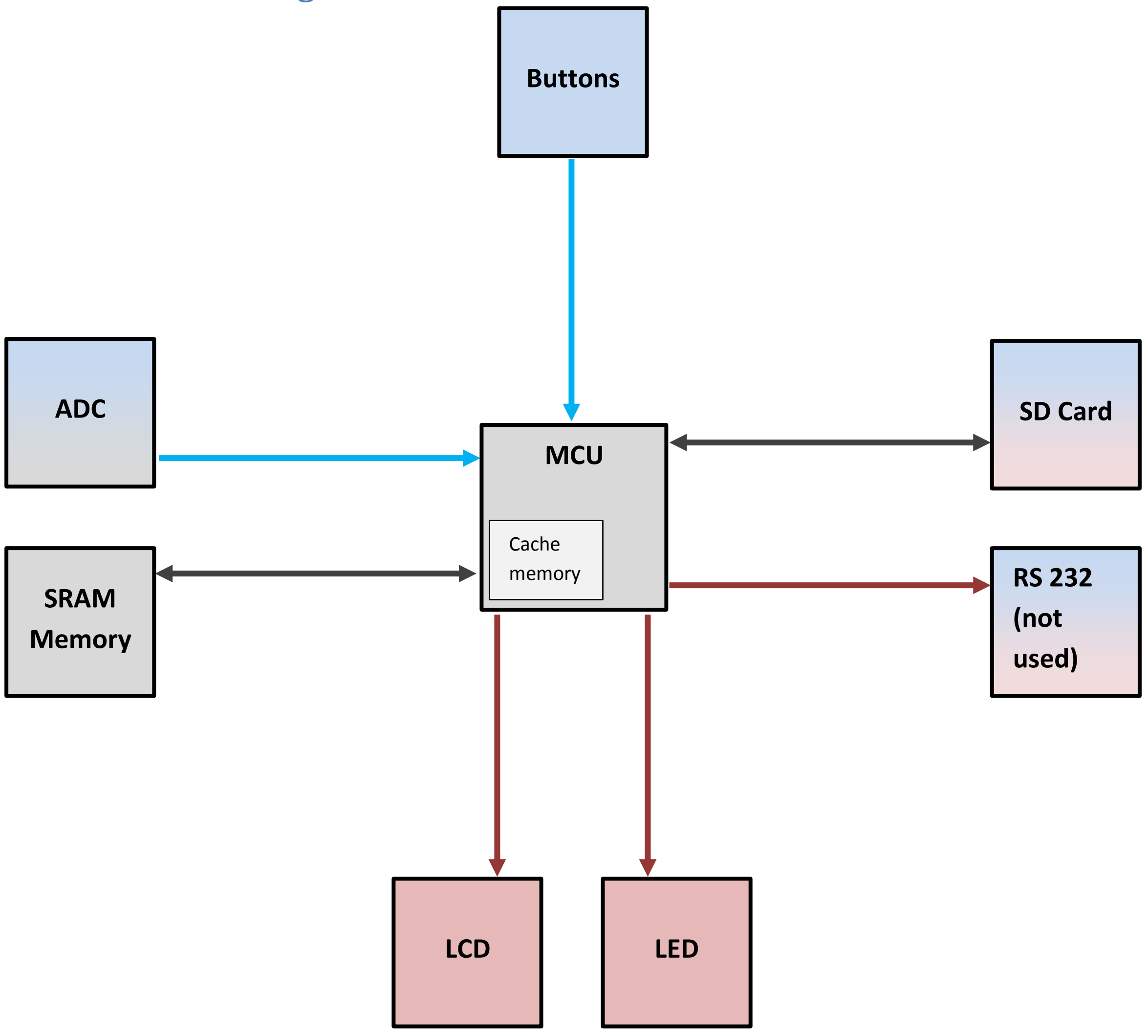


Figure C3: ADM beta prototype information flow diagram

Figure C3 illustrates how information passes between the components of the ADM beta prototype. Blue boxes represent input components, red boxes represent output components and grey boxes represent internal components among which input data is passed for processing, before being sent to the output components. Note the SD card and RS232 units act as both input and outputs, hence they are gradient shaded to reflect this. In addition, the ADC unit is internal to the MCU, but operates separately unit from the main processing unit.

To assist with the distinction between input, output and internal components; the arrows symbolising data flow are colour coded as well: Blue for data going into the MCU, red for data flowing from the MCU and black for bi-directional data.

As expected, input data flows from the buttons (as a means of UI) and the ADC unit (the raw measured data). Additional input data is retrieved from the SD-card. During processing, the information is passed between the MCU and the memory (via the cache memory – where active variables are stored). After the data analysis is complete, outputs are passed to the LCD display, the SD card and potentially the RS232 unit (unimplemented in the final firmware release, but operational). In addition, while the device is active, certain activities may trigger the LEDs, to provide the user with assurance of normal operation.

APPENDIX D – MCU driver operation

Although the focus of this project is how the bearing state assessment algorithm works in conjunction with the signal processing hardware, significant resources were spent on driver programming which allows the various electronic subsystems to communicate with each other through the central MCU.

This section details some of the non-algorithmic routines (hardware drivers) which had to be manually coded in order to make the device function. Each driver is discussed in terms of functional premise and philosophy rather than technical computing. This approach is deemed sufficient considering the focus of this document.

Appendix C provides a background to this section.

D.1 ADC driver

The most recent ADC driver of the beta prototype is now discussed. This will differ from the driver used for the alpha prototype, as the two systems use different ICs for the purpose. In fact, the alpha prototype used an external IC connected to the main MCU whereas the beta prototype uses the on-board system of the MCU, as explained in section 2.4.3. The premise and basic source code was obtained from Microchip but had to be heavily modified to work with the current design and compiler (Microchip Technology Inc, 2008).

Operational premise:

The framework of the driver, along with the oversampling technique (section 1.3.3.6) was obtained from Microchip. Therefore, the entire driver was designed around this technique.

The philosophy involved the use of ‘ping-pong’ buffers. This involved the use of two buffers to which the ADC can write data. The implementation then involved low-pass filtering and decimation on the buffer that was already full, while filling the other buffer with new data.

A FIR filter was used for this driver, which of-course requires a history of previous samples to calculate the next, filtered, data point. This implied that the buffers were intricately linked and care had to be taken to ensure that, during the first few data points in the new buffer, the required history in the old buffer was not yet overwritten (luckily, the MCU speeds are orders higher than the ADC sampling rate, so realistically this was never expected to be a problem). This additionally required the MCU to both read and write to the same buffer at the same – a requirement that necessitated the use of the DMA (Direct Memory Access). As the name implies, this allows that ADC module to bypass the MCU core and communicate directly with the appointed section of memory (specifically, the section that housed the ping-pong buffers), freeing the MCU core to process the data (filtering, decimation, etc.) without requiring it to administrate the ADC-memory data flow.

Preliminary configurations: Interrupt, timer and DMA

During initialisation, the ADC driver was programmed to ready a number of parameters. The most important of which are:

- ADC resolution to 12 bit

- Points the ADC analogue input to the appropriate bit of a port (i.e. a specific pin on the chip)
- Synchronises the ADC conversion period to a general purpose timer and sets the period of the timer to correspond to the desired sampling frequency
- Allocate the addresses of the ping-pong buffers
- Activates the DMA and links the ADC output buffer to the ping-pong buffer
- Sets the necessary flags (such as interrupt flags, buffer select flags and buffer full flags).

Main driver

The procedure followed by the driver is best described by means of a flow diagram, as in figure D1.

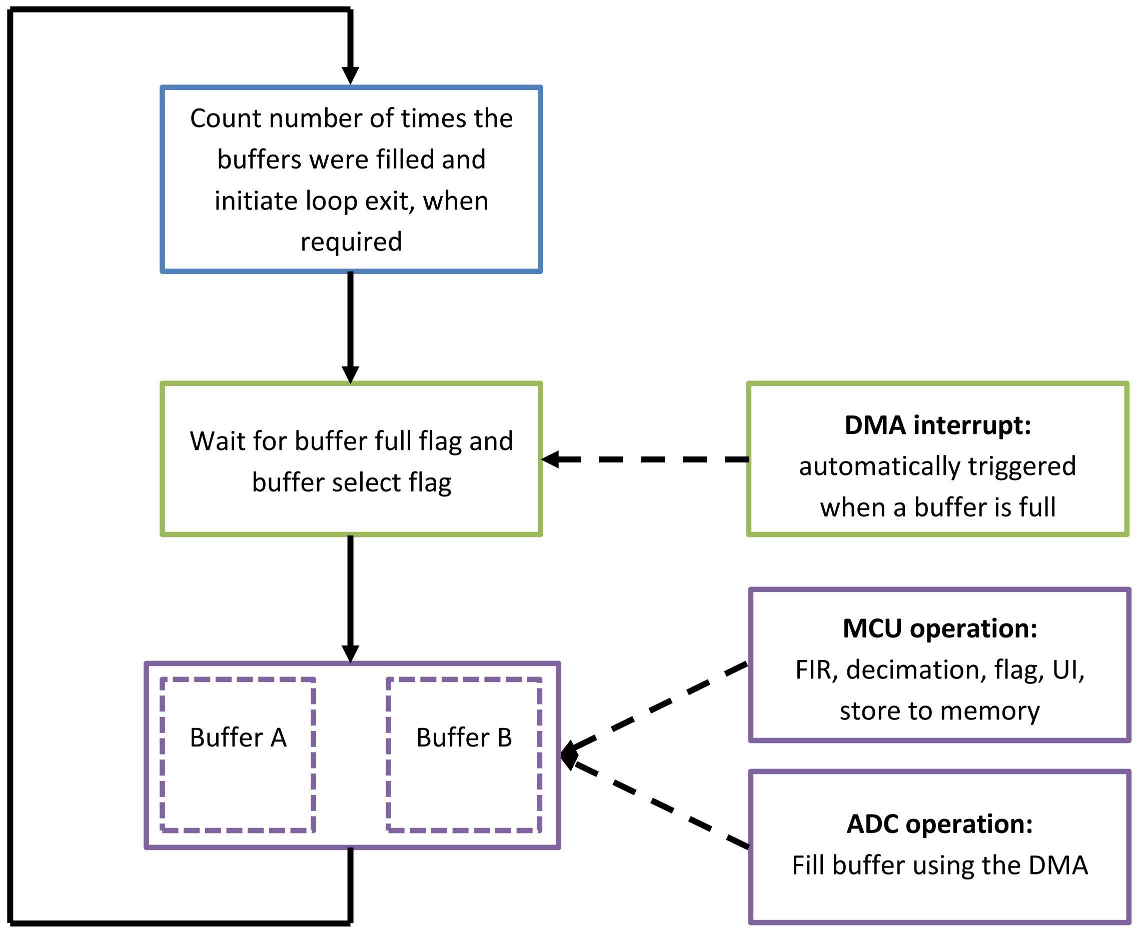

Count number of times the buffers were filled and initiate loop exit, when required
Wait for buffer full flag and buffer select flag
DMA interrupt: automatically triggered when a buffer is full
MCU operation: FIR, decimation, flag, UI, store to memory
Buffer A
Buffer B
ADC operation: Fill buffer using the DMA

Figure D1: ADC driver flow diagram

The entire operation resides within a loop that repeats the number of times that the buffers require to fill the memory with the desired amount of processed data samples (for reference, each buffer was 512 bits long and 32 768 samples). When the buffers had, in turn, filled the memory with the required amount of samples, the loop will exit and the ADC module switched off to save power (not pictured). This counting process is illustrated in the blue box.

Within the loop there also exists a function that monitors whether the one buffer had been completely filled with data from the ADC, after the other had been processed. This is necessary as the MCU operates much faster than the ADC, implying that the MCU will finish processing the one buffer before the ADC can finish filling the other. This function therefore allows the ADC the time required to fill the current buffer it is working on and temporarily suspends the MCU. After the ADC has finished filling its buffer, the function will allow the buffers to switch allowing the MCU to operate on the other buffer and allowing the ADC to refill the buffer that the MCU had been operating on.

Had this function not been in place, a conflict would occur as the MCU would start processing the same buffer that the ADC is filling with data, after it (the MCU) finished with the other buffer. This process is illustrated in the left most green box. The function just explained is assisted by the DMA interrupt (see note at the end of this section for a brief explanation on interrupts) which is triggered when a buffer is full. The interrupt – represented by the right most green box – lets the main function (the left most green box) know that the ADC had finished filling a buffer with data and that switch has taken place.

In the meantime, while the one buffer is being filled with data, the other buffer is being processed according to the requirements of the oversampling algorithm. During this time, the data in the buffer is filtered, decimated and stored in the SRAM memory. The operations on the buffers are illustrated by the purple boxes. The ADC will however still be occupied filling the other buffer. Once it has finished doing so, a buffer full flag will be set.

After this process is finished, the loop would start over again. Therefore, a counter (in the first function – indicated by the blue box) would increase in value and that function will then decide if more samples are required.

Note
An interrupt is a function that automatically executes only when a certain condition is met and is not explicitly called. Sources of interrupts include RS232 connections, timers, SPI interfaces, ADC units, etc.

In this project, the interrupt function was configured to execute automatically when the ADC had finished filling a buffer with data. The function relayed to the main driver that the ADC had finished filling a buffer and also which buffer, specifically.

D.2 Memory driver

Introduction

The volatile SRAM of the alpha and beta prototypes (Static Random Access Memory) were used to store the data samples that is being used by the algorithm. This was necessary as the internal cache memory of the MCU was not large enough to house all the data. SRAM is an asynchronous type of memory, implying that its operation is not synchronised to a clock pulse and can be utilised anytime, as long as it is configured properly. In addition, the memory has 512k addressable locations of 8-bits (one byte) each.

The memory is connected to the MCU via several ports. A control bus runs from one port on the MCU to the memory chip (for turning on the chip, write command, read command), an address bus for setting the address to write to or to be read from and the data bus, where the actual data transfer takes place. The figure D2 illustrates the layout.

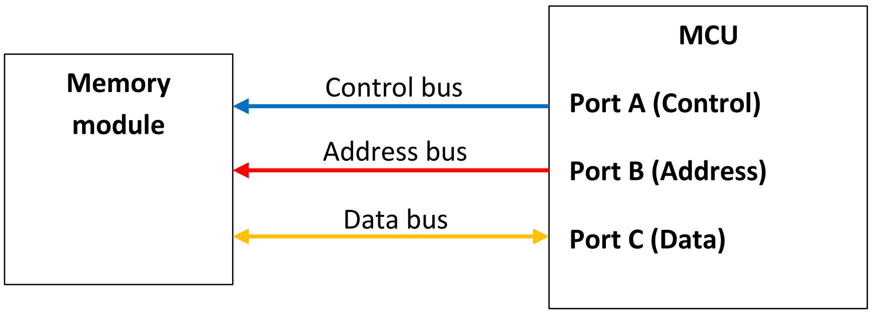


Figure D2: Memory bus connections to the MCU

Table D1: Memory address layout

Addr 0	*Addr 1*	*Addr 2*	*Addr 3*	*Addr 4*	*Addr 5*	*Addr 6*	…
8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	

Digital number formats used

As mentioned, the ADC outputs integer data of 16 bits. Which implies that each sample had to be stored in two separate memory locations. However, the algorithms work strictly with floating point values which are 32 bits in length. Therefore, after all the data had been sampled by the ADC, every data sample was converted to a floating point number of the same value. Thus, if the ADC yielded digitised values of 12, 23, 120, 97, 571... (bear in mind, these values did not reflect values in traditional ADC units, but digitized values based on the input pin from the ADC), they were converted after the sampling process to 12.0000000, 23.0000000, 120.0000000, 97.0000000, 571.0000000....

Driver layout

The driver consist of several functions: Initialisation of the memory module, writing integer numbers, writing floating point numbers, reading integer numbers, reading floating point numbers and converting the integers existing in the memory to floating point numbers.

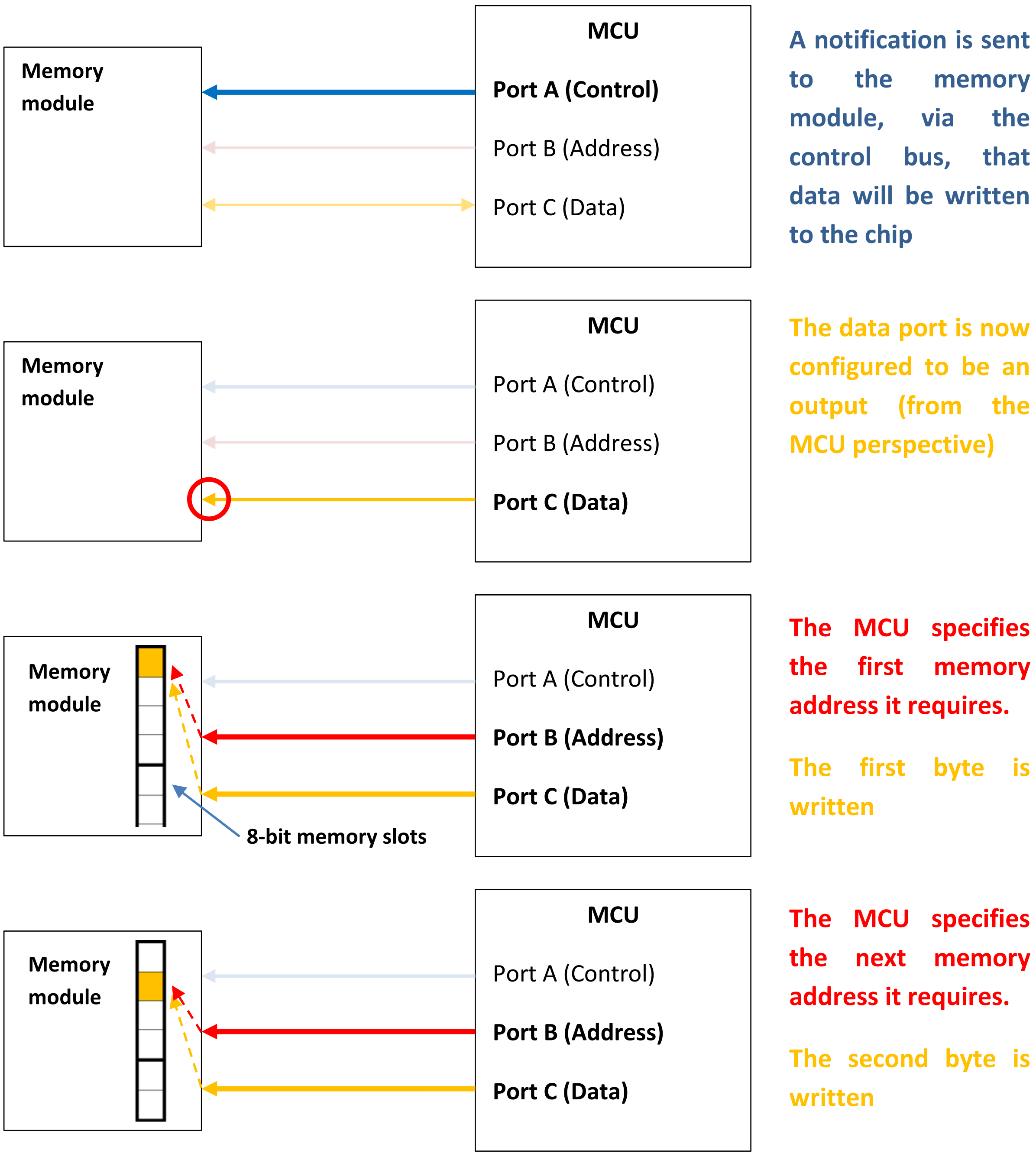
Initialisation

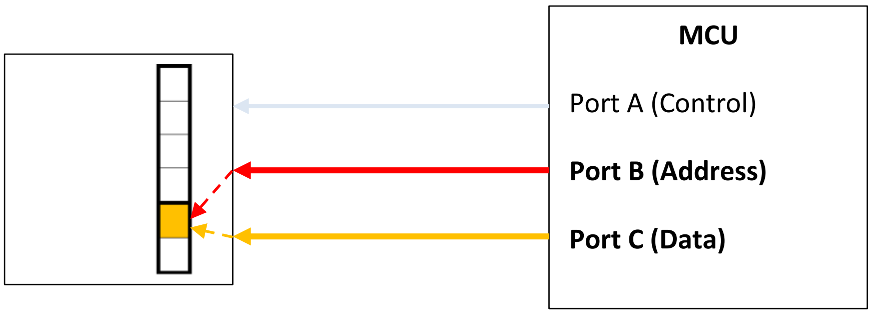
The initialisation routine is executed once during start-up of the device and configures the pins connecting the address bus as output (as the address is always sent to the memory module – whether one needs to read from the address or write to the address and, the command bus as output (commands are always sent to the memory module). In addition, the memory module requires a series of initial commands that needs to be sent to the module before the first operation.

Integer write

The next function that forms part of the driver suite is the integer write function. This function is typically used when the ADC values are written to the memory.

When writing an integer value (of 16 bits), a control command is given to prepare the chip to receive data whilst the data port (connecting to the data bus) is configured as an output. It then performs two write operations. This is because the memory module is divided into 8 bit memory slots. The 16 bit integer therefore had to be separated into a higher and lower byte and stored separately. The operation of the integer write operation is illustrated in the figure below (note that when integers are written, only the first and second - of four - memory slots are used. The third and fourth are reserved for a floating point conversion, as will be explained imminently).


Memory module
MCU
Port A (Control)
Port B (Address)
Port C (Data)
A notification is sent to the memory module, via the control bus, that data will be written to the chip
Memory module
MCU
Port A (Control)
Port B (Address)
Port C (Data)
The data port is now configured to be an output (from the MCU perspective)
Memory module
8-bit memory slots
MCU
Port A (Control)
Port B (Address)
Port C (Data)
The MCU specifies the first memory address it requires.
The first byte is written
Memory module
MCU
Port A (Control)
Port B (Address)
Port C (Data)
The MCU specifies the next memory address it requires.
The second byte is written



The next integer is written four memory locations from the first byte of the previous integer (in case floating point convers is required)

Figure D3: Diagram depicting the steps of integer storage to the SRAM module

Therefore, if the number 14893 (binary number 0011101000101101) had to be stored in memory, the first 8-bit memory location would contain 00111010, and the second would contain 00101101.

Floating point write

As seen in Figure D3, where integer write is illustrated, only the first and second, of four, memory locations are used per write. This was because the integers used are 16 bits long, while each memory location is only 8 bits longs. The integer therefore had to be divided into two 8 bit halves and stored consecutively.

The same principle applies to floating point numbers as well. Only now, each floating point number is 32 bits long. This implies that each number will be divided into four 8 bit quarters and stored consecutively.

Incidentally, this is also the reason why only the first two of every four memory location are used by the integer write routine – to leave space for when that number is later converted into a floating point value.

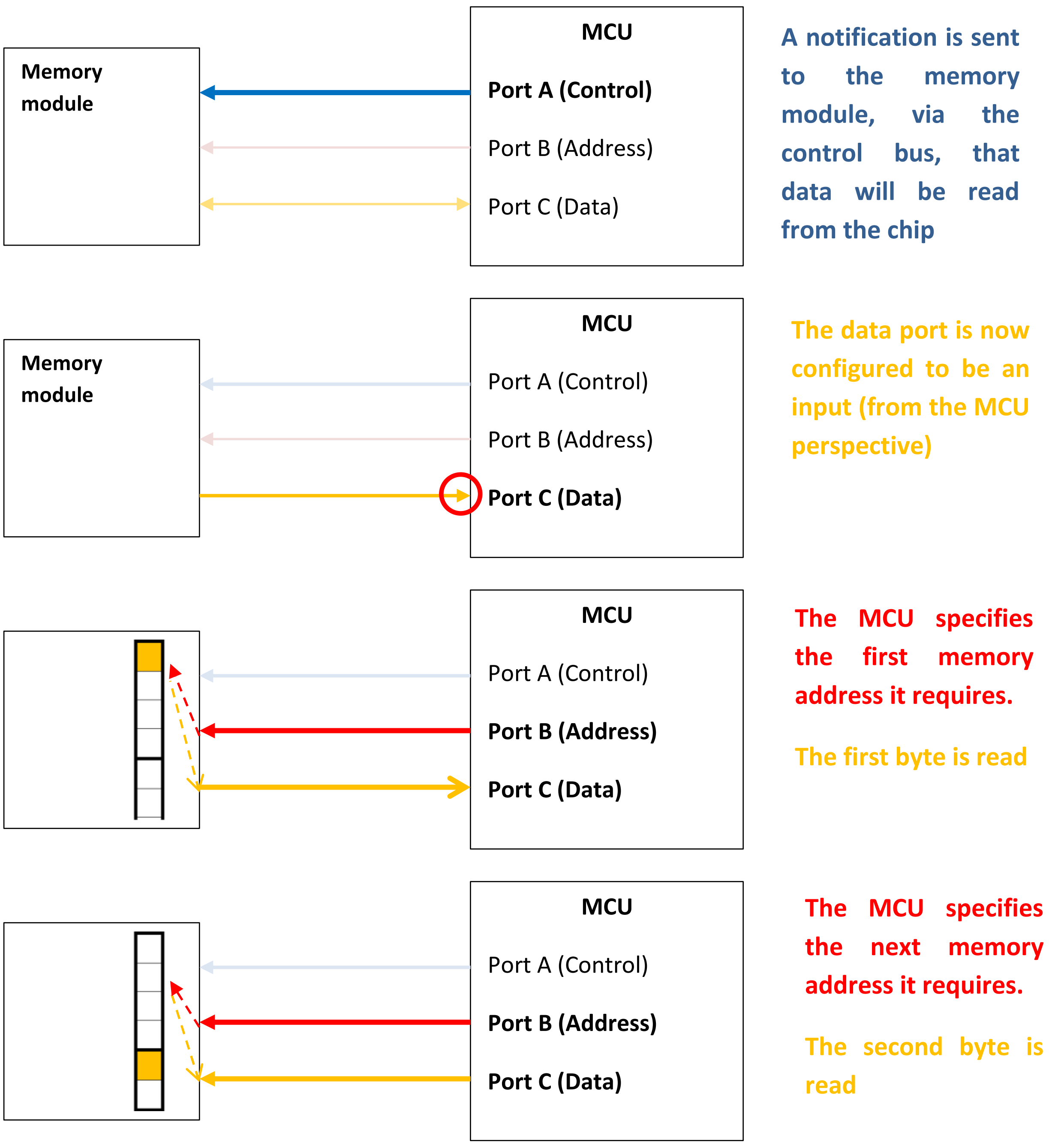
This will be required, as the algorithms work with floating point values, yet the ADC outputs integers. The table below illustrates how the transition to floating point numbers:

Table D2: Integer vs. floating point number layout in 8-bit memory

Data entry number	Memory location	Integer bytes	Floating point bytes
1	0	*Low byte*	*Low byte*
	1	*High byte*	*Middle byte 1*
	2		*Middle byte 2*
	3		*High byte*
2	4	*Low byte*	*Low byte*
	5	*High byte*	*Middle byte 1*
	6		*Middle byte 2*
	7		*High byte*
3	8	*Low byte*	*Low byte*
	9	*High byte*	*Middle byte 1*
	10		*Etc…*

Integer read

In many ways, reading integers from the memory is the reverse of writing them to memory. Firstly, a command is sent via the control bus to the memory that data will be read from the chip. After data port is then configured as an input, two read operations, from consecutive memory slots, are performed and the results combined into a 16 bit integer value. A figure depicting driver operation is provided below.


Memory module
MCU
Port A (Control)
Port B (Address)
Port C (Data)
A notification is sent to the memory module, via the control bus, that data will be read from the chip
Memory module
MCU
Port A (Control)
Port B (Address)
Port C (Data)
The data port is now configured to be an input (from the MCU perspective)
MCU
Port A (Control)
Port B (Address)
Port C (Data)
The MCU specifies the first memory address it requires.
The first byte is read
MCU
Port A (Control)
Port B (Address)
Port C (Data)
The MCU specifies the next memory address it requires.
The second byte is read

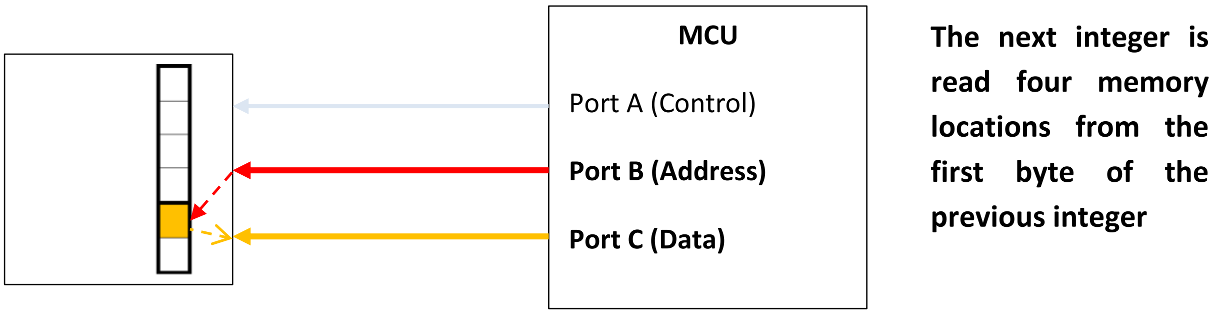

MCU
Port A (Control)
Port B (Address)
Port C (Data)
The next integer is read four memory locations from the first byte of the previous integer

Figure D4: Diagram depicting the steps of integer storage to the SRAM module

Floating point read

The similarities that exist between integer write and floating point write, exist between integer read and floating point read. Assuming the numbers in the SRAM was converted to floating point values, each value will be 32 bits (4 bytes) long – which implies that they will each be distributed across four consecutive 8-bit memory addresses.

The floating point read routine would then follow a similar pattern to the integer read, except that four distinct read cycles would be repeated and the four byte appended to each other to form the number.

Integer to floating point conversion

This function was called after the sampling process was complete and the integer results from the ADC stored in the memory. The function was fairly simple in that it consisted of reading each integer (spread across two memory locations), converting the number to a floating point value, and rewriting the number, overwriting the integer in the process.

D.3 SD Card driver

Introduction

The SD card is utilised in the beta prototype to store parameters of several gearboxes on as well as analysis results.

The SD card reader and circuit communicates with the MCU by means of a Serial Peripheral Interface (SPI). As the name implies, it is a serial interface which implies that data is sent, bit for bit, along a data line. This is opposed to the SRAM, which utilises parallel data lines (a data bus). SPI connection between two devices requires also a shared clock pulse between connected devices.

The complete SPI interface is illustrated below:

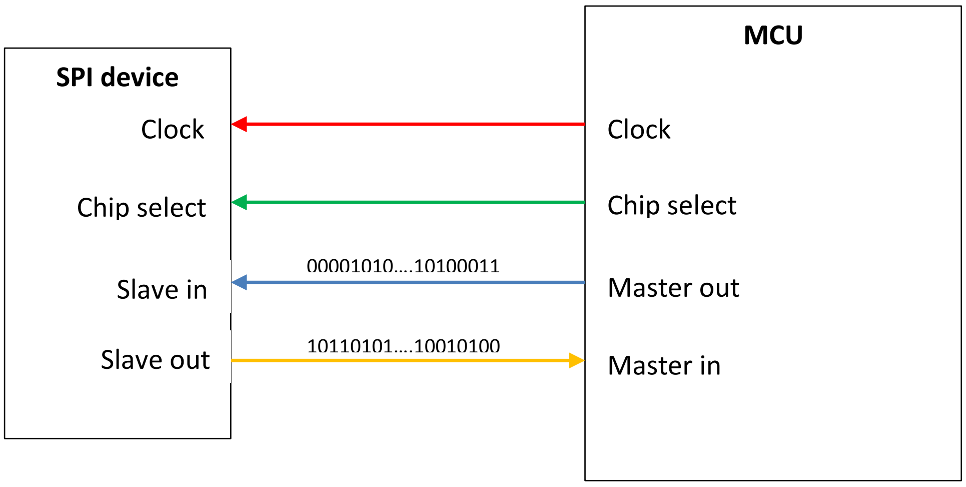


Figure D5: SPI connections

Figure D5 illustrates the connections required for a SPI connection. The clock line synchronises the data transfer between the master (in this application, the MCU) and the slave (in this application the SD Card reader). The master out/slave in line is where data is transmitted from the master to the slave device. This occurs bit for bit on each clock pulse. After 8 bits, the slave will typically recognise that a full data byte has been received, which can then be used. Conversely, the master in/slave out line is where the data is similarly transferred to the MCU.

The chip select line is unnecessary if only one peripheral is connected to the SPI port of the MCU. It is however possible to connect several devices on the same port, in which case each device will have its own chip select port, but share the same clock, master in and master out lines. Since only the SD card is connected to the SPI port, this line is redundant.

Driver layout

Write operations

The compiler library includes a section that deals with SD card operations, including SPI connections. It was soon discovered however that these operations are performed using a 512 bit buffer, regardless of the length of the data that needs to be sent or received. Therefore, if only 16 bits of data is written to the SD card by the user, a full 512 bits will be written with the final 504 bits being zero.

This is not necessarily inefficient, but it does need to be harnessed, as the SPI cycle described above will be repeated 64 times (512 bit buffer, implying 64 eight bit serial writes to the SD card), with every write command.

Therefore, the write function initially stored all the data in a global buffer (of 512 bits long) every time it was called, until the buffer was filled, upon which the entire buffer was written to the SD card. The buffer would then be cleared and the process would start again. The process is illustrated below:

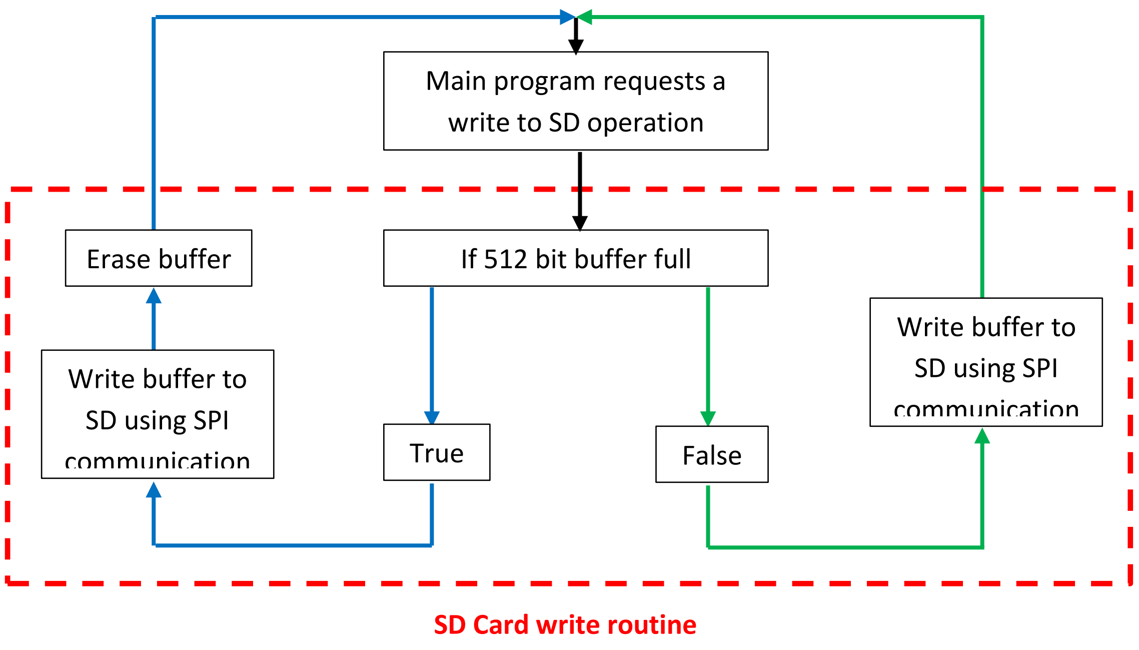

Main program requests a write to SD operation
Erase buffer
If 512 bit buffer full
Write buffer to SD using SPI communication
True
False
Write buffer to SD using SPI communication
SD Card write routine

Figure D6: SD card write flow diagram

Read operation

Reading operations work in a different way to write operations, however. Even though the chip still read a full 512 bit sector with each call of standard SD read library function, it does not really matter, because the algorithm requires very few reads per execution (<100 – it only reads the contents of the parameter file), while thousands of write operations are performed (saving the time and frequency domain outputs of the algorithm as well as the test log file).

With so few reads, it was not considered worthwhile to buffer all the data required from the files on the SD card at once.

However, a distinction was made between what type of data is required from the SD card: integer data, floating point data, or alpha numeric data. Bearing in mind, that from an end-user perspective, the data on the SD card would appear as an ASCII file. A read would then be performed as illustrated in the figures D7 and D8.

If is firstly necessary to briefly describe how an ASCII file is encoded: An ASCII file consists of characters – some of which are printed and some of which are not. The long string of numbers and symbols in figure D7 represent how the ASCII file in figure D8 is encoded (in actuality, it is more complex than illustrated, but for the purposes of understanding the driver, the illustration is sufficient). As can be seen, all the characters in figure D8 are present (including the spaces, illustrated with double apostrophes). However, two extra characters are present: /r and /n. These characters represent the carriage return and newline 'characters'. In essence, the /r character instructs the cursor to go the beginning of the current line and the /n character tells the cursor to go the next line. In light of this, the string in figure D8 can be interpreted.

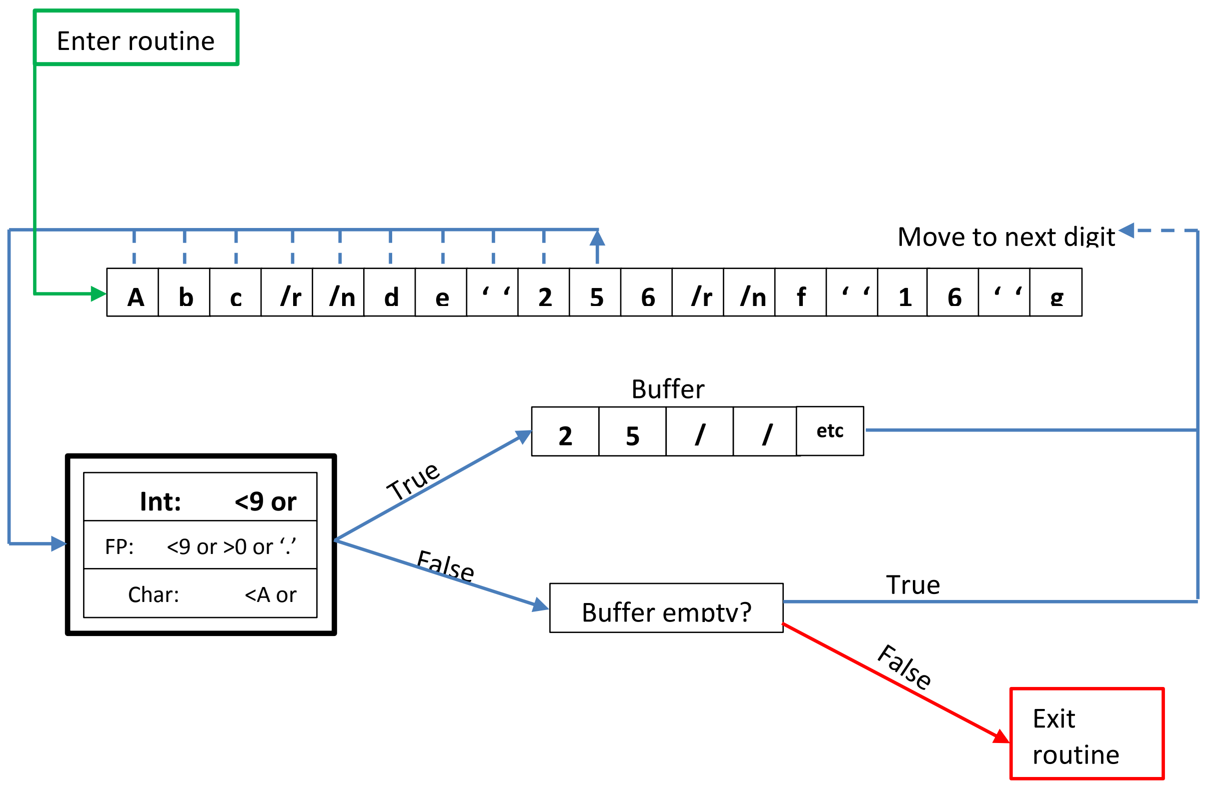
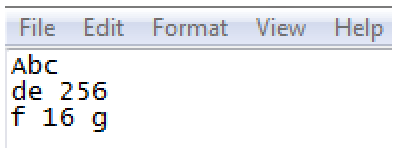


Figure D7: SD read flow diagram (at 10th cycle)

Figure D7 legend:

/0	Empty bit - null
/r	Carriage return
/n	New line



```
Abc
de 256
f 16 g
```

Figure D8: An example ASCII file

With this in mind, the driver operation can be explained (the example of integer read will be explained): The routine starts its first cycle with an empty buffer in which successive numerals will be stored. Keeping this in mind, it commences reading the first character of the ASCII file, which is an 'A'. This value does not fall between the ASCII values of '0' and '9' and, since the buffer is still empty, reads the next character: 'b'. This character also does not fall between '0' and '9', the buffer is still empty, and the next character is therefore read.

This process continues until a character is read that does fall between '0' and '9'. In the example above, this occurs at the 9th cycle of the code where the character '2' is read. This character replaces the first null (empty) bit in the buffer. The cycle then continues. Figure D7 depicts the following cycle, where the character '5' is read and appended to the buffer.

This will continue until another character is read which falls outside the '0' to '9' boundary (by the 12th cycle, in this example). This character would be the carriage return (/r) character. As it falls outside the boundary, and the buffer is not empty anymore (it would contain the values of 256), the routine would exit. A function would now read the buffer up to the first null character and convert the string to an integer, which can be used for computations.

The algorithm works the same when searching for floating points. Only then, the boundary would be '0' to '9' and the '.' character would be tested for as well, and added to the buffer if found. The results would again be passed to a function that reads the buffer up to the first null character. The string would then be converted to a floating point number, which can be used.

The alpha numeric search works in the same way.

D.4 Button driver

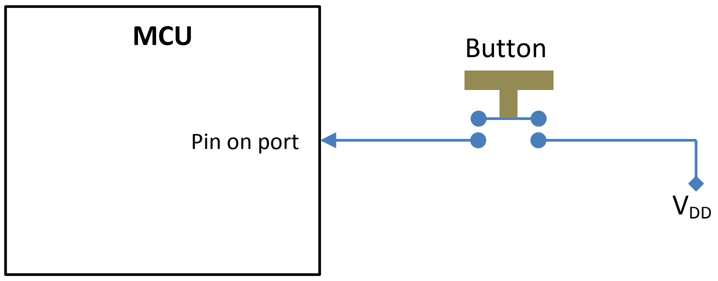


Figure D9: Simplified button circuit

Figure D9 illustrates a simplified button circuit. The button is basically a spring loaded switch that completes an electronic pathway once pressed, allowing the V_{DD} voltage to appear on the pin, to which the circuit is connected.

The driver utilises this by firstly setting the pin in question to be an input, implying that the MCU reads the value on the pin. When the button is pressed, this will have the effect of changing a bit in a standard port register, which is monitored by software when a button press is expected. The flow diagram of driver operation is illustrated below:

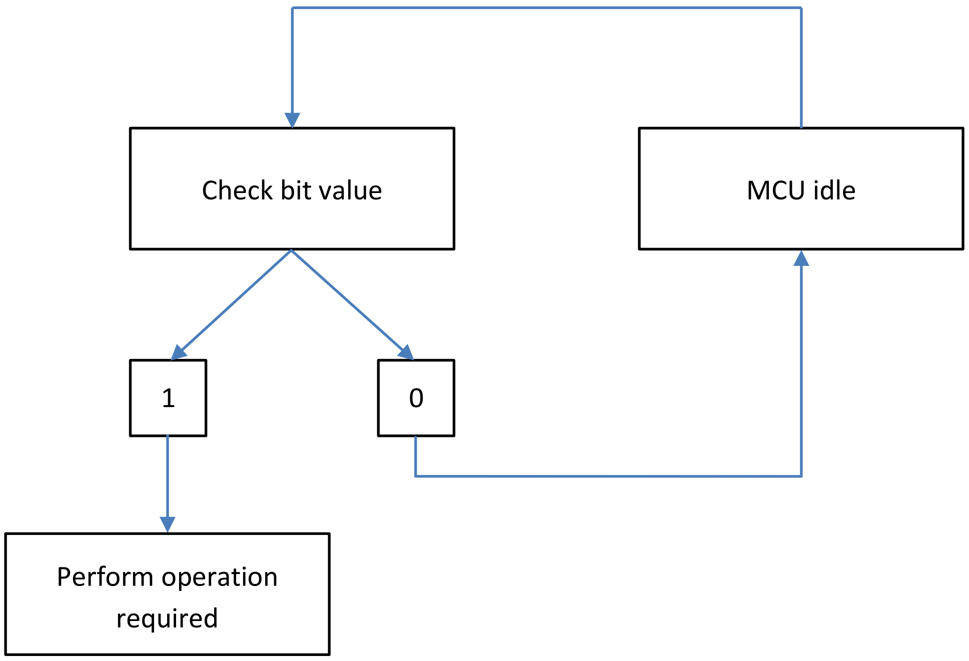


Figure D10: Button driver flow diagram

D.5 LED driver

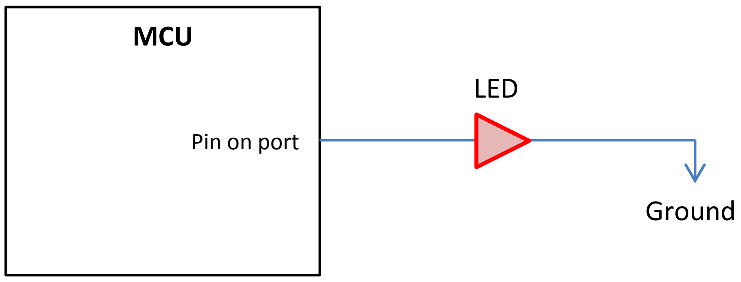


Figure D11: Simplified LED circuit

Figure D11, illustrates a simplified LED circuit. Driver operation is fairly straight forward, in that the pin on the port needs to be configured as an output. When the pin is supplied with a voltage (via a bit in the specific ports register), a voltage difference will exist across the LED, which will cause it to illuminate.

D.7 LCD and RS232

There are two more peripherals that require mention: the LCD and the RS232. Of course, these devices required drivers as well. They were not manually programmed however.

The LCD driver was obtained from Periseo and the adaptation required for it to function was small enough not to warrant its own section.

Likewise, the RS232 driver – although not used in the final release of the firmware – was mostly composed of a library function from the Mikroe compiler.

APPENDIX E – Circuit design and layout

E.1 Circuit design

This section includes the design of the circuitry, i.e. the physical connections between the components (as opposed to their arrangement, as illustrated in section E.2).

Note that this represents the circuit design as sent to that was manufactured. After manufacturing, it was found that several slight mistakes were made, which were repaired after manufacturing and therefore is not included here.

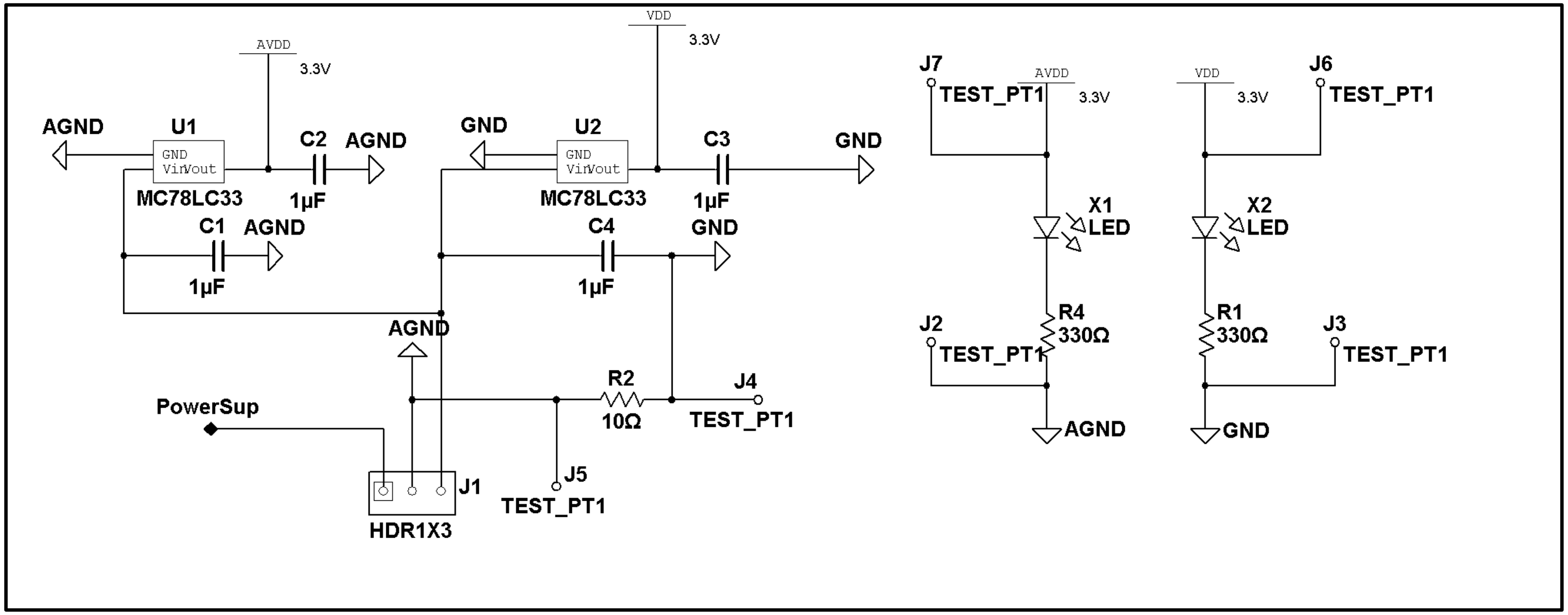


Figure E1: Power supply circuit

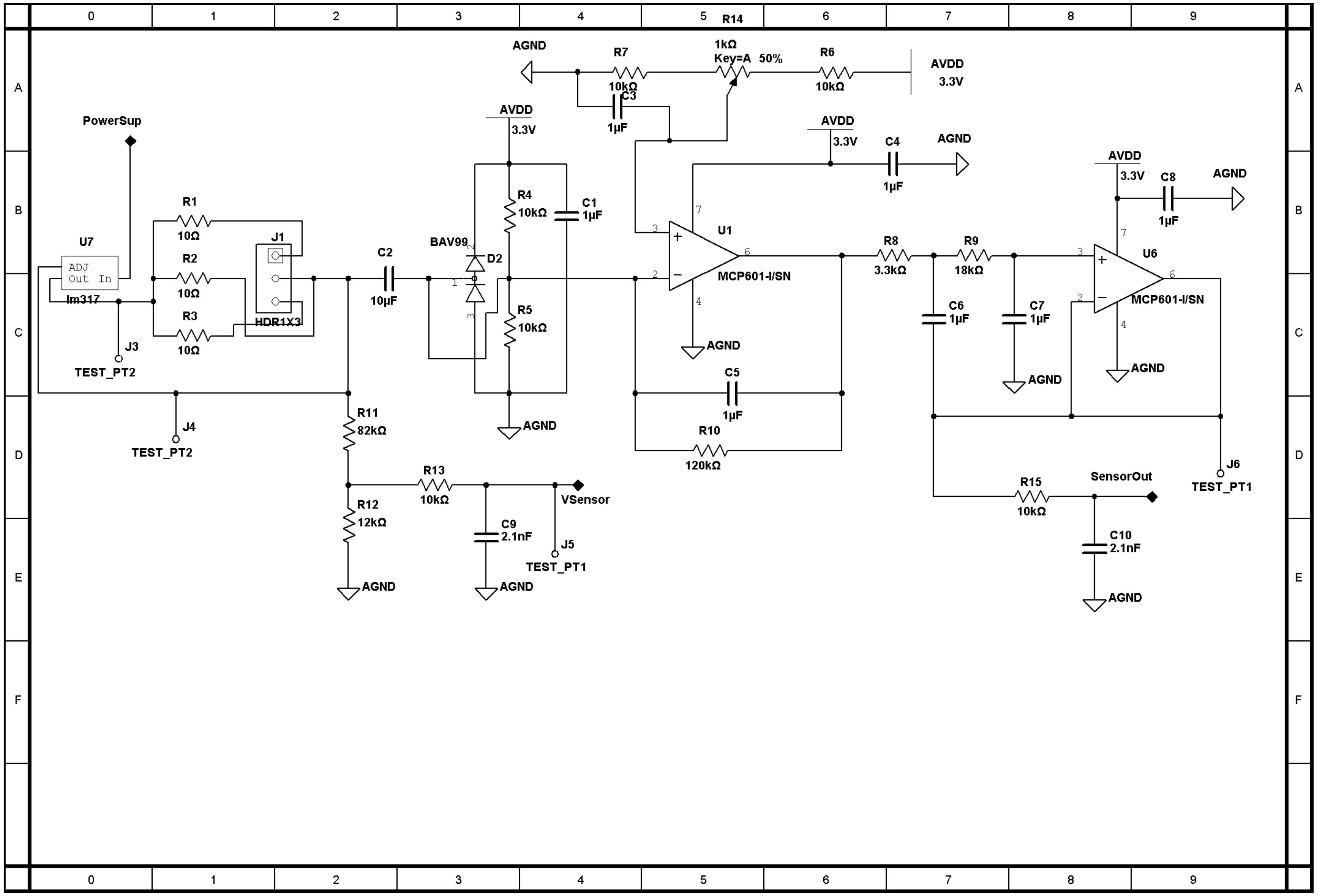


Figure E2: Anti-aliasing filter circuit

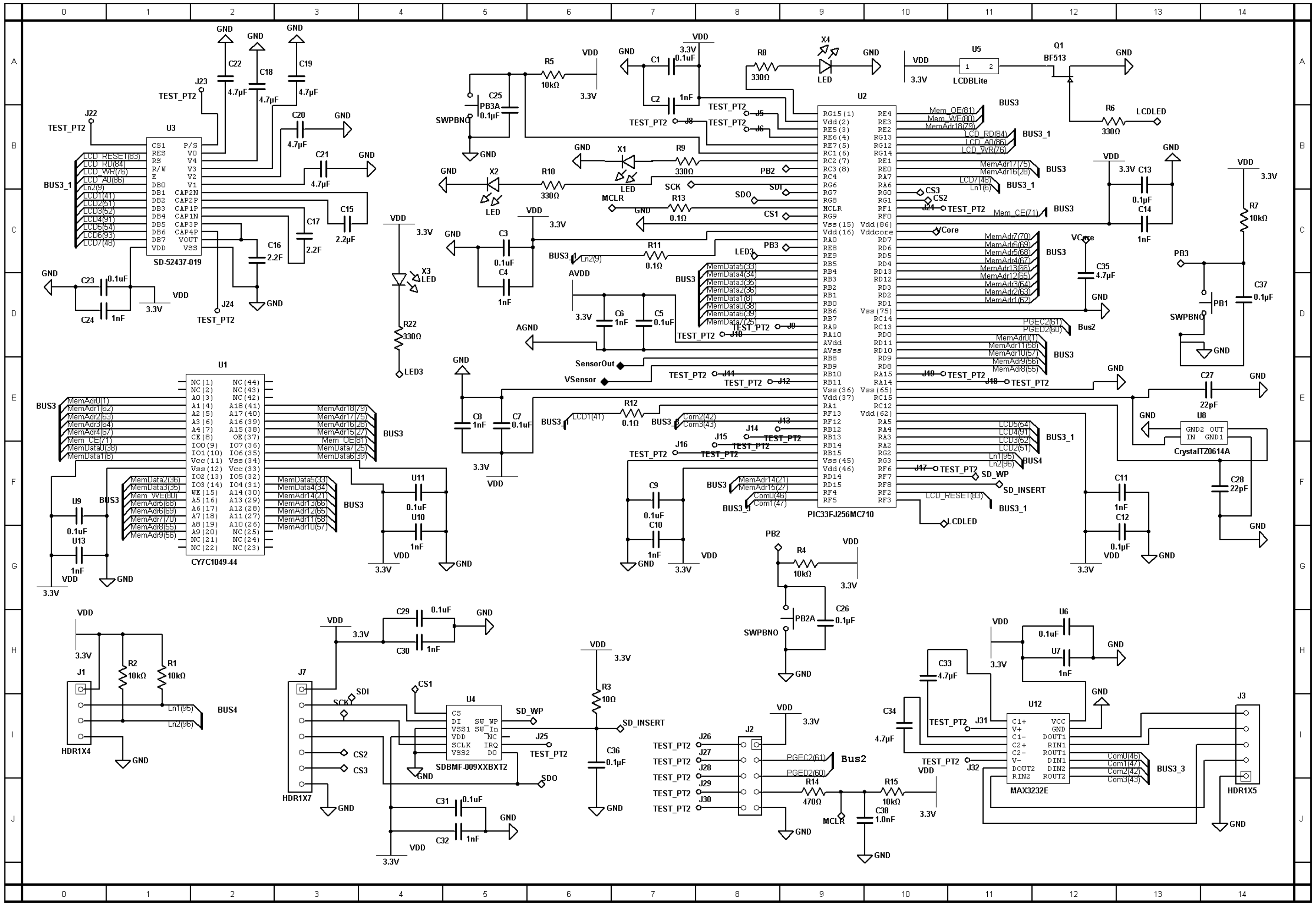


Figure E3: Main circuit board design

E.2 Circuit layout

The circuit board layout is illustrated here. Note that the beta prototype used a four layer PCB. This implies that four separate layers were sandwiched and printed. The electronic components were placed on the top and bottom layers, which also housed the majority of the copper connection lines between components.

A third and fourth layer exists between these outer layers. One contained a common ground and another a common power level. These layers are not pictured.

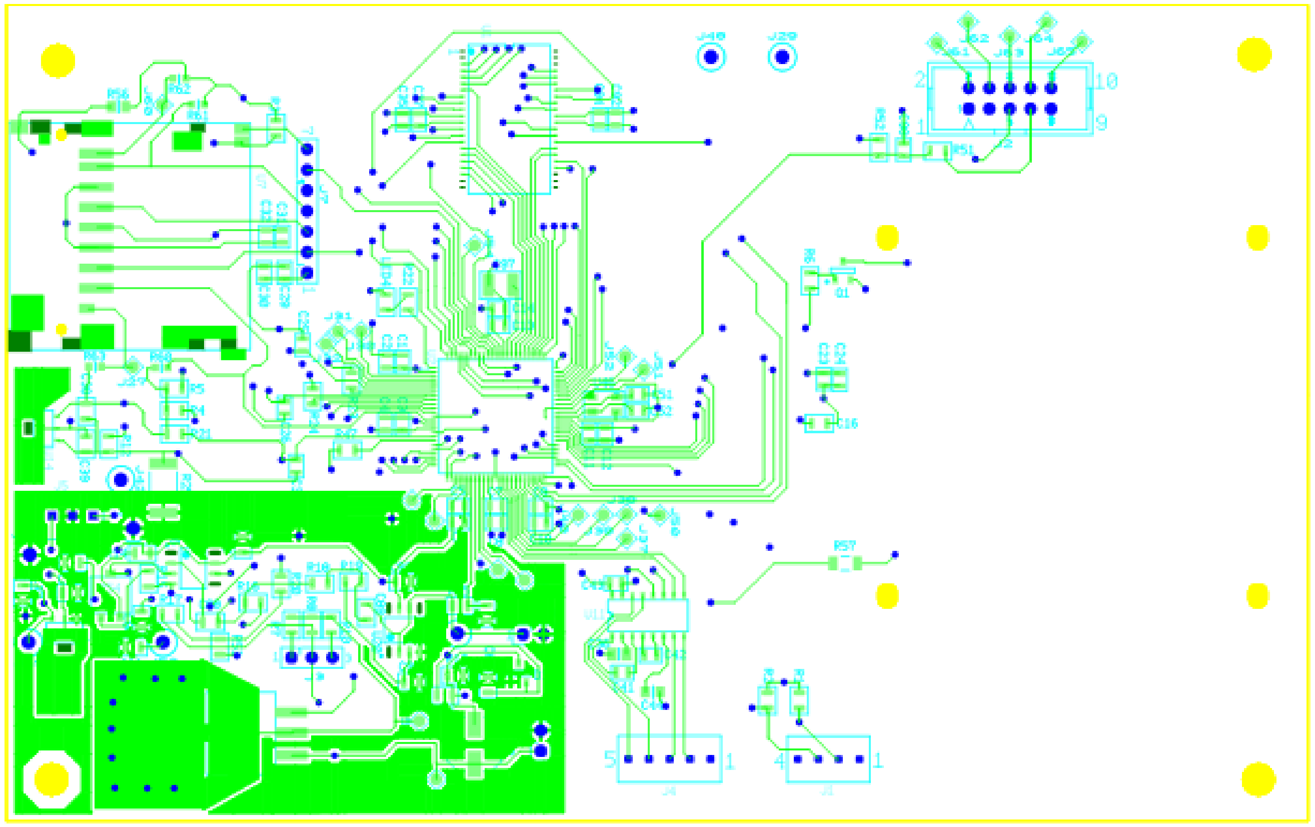


Figure E4: Top printed layer

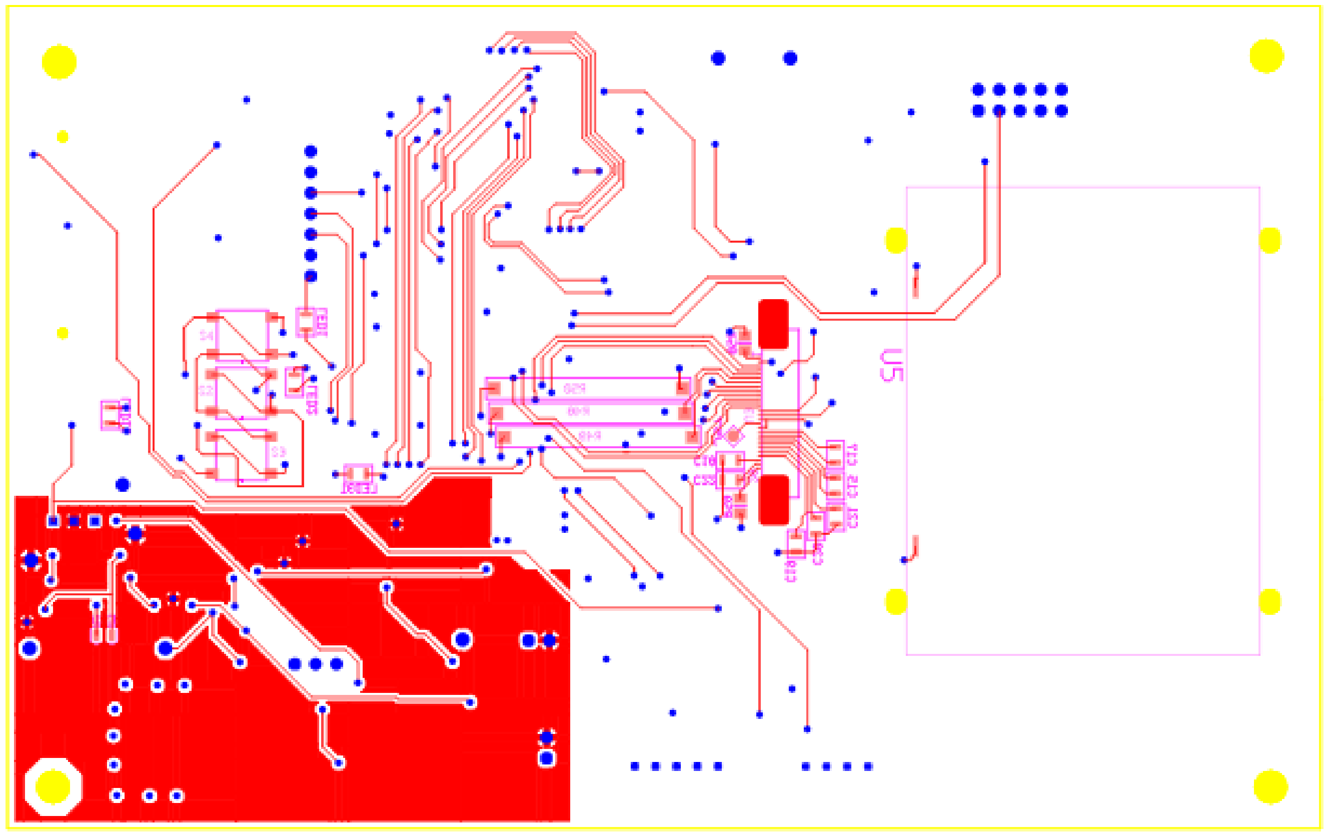


Figure E5: Top printed layer

APPENDIX F – ADM Beta prototype illustrations

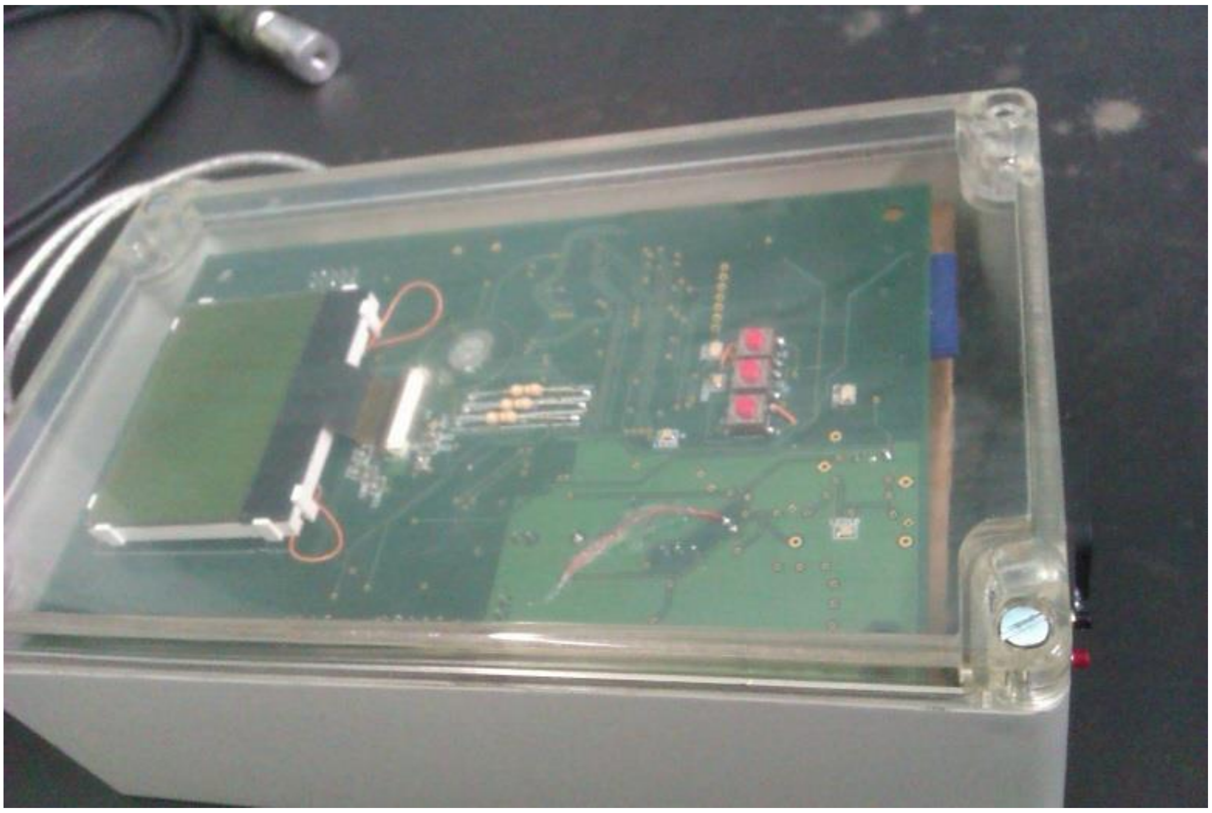


Figure F1: ADM beta prototype photo (switched off)

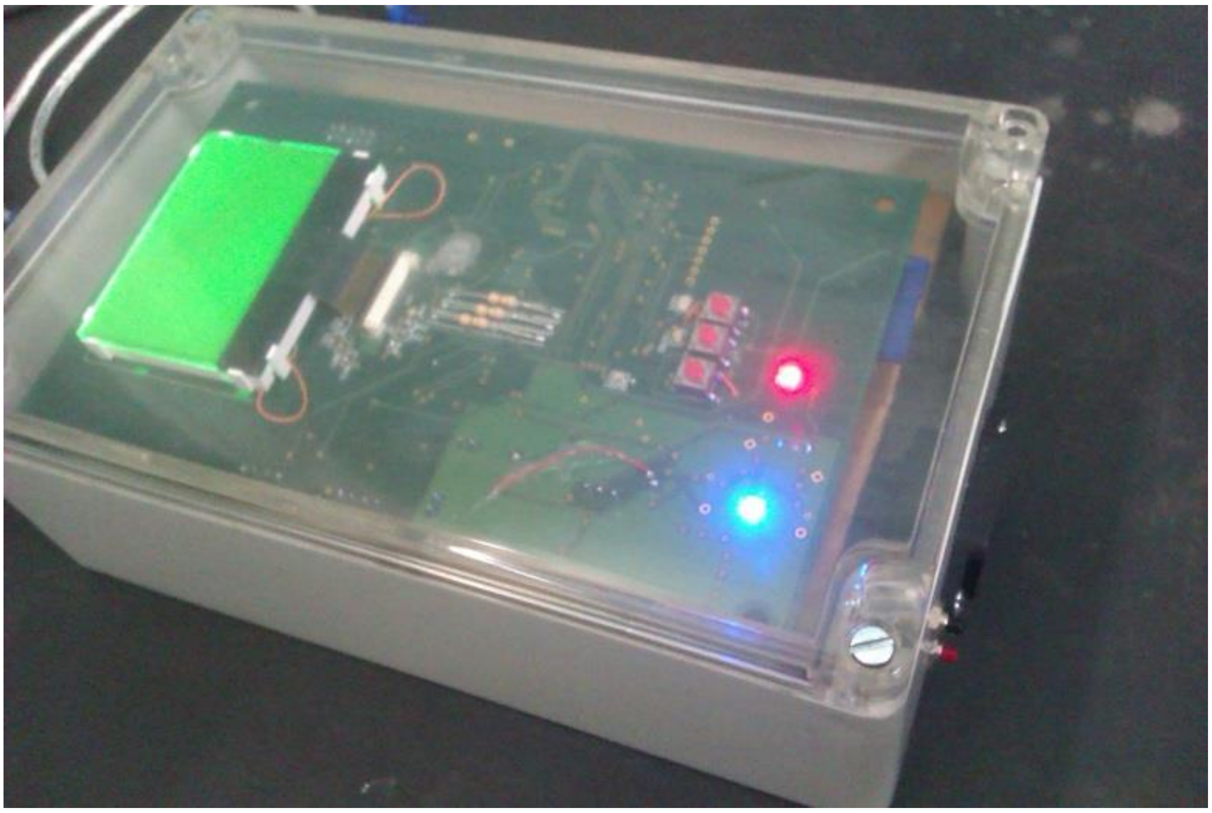


Figure F2: ADM beta prototype photo (switched on)

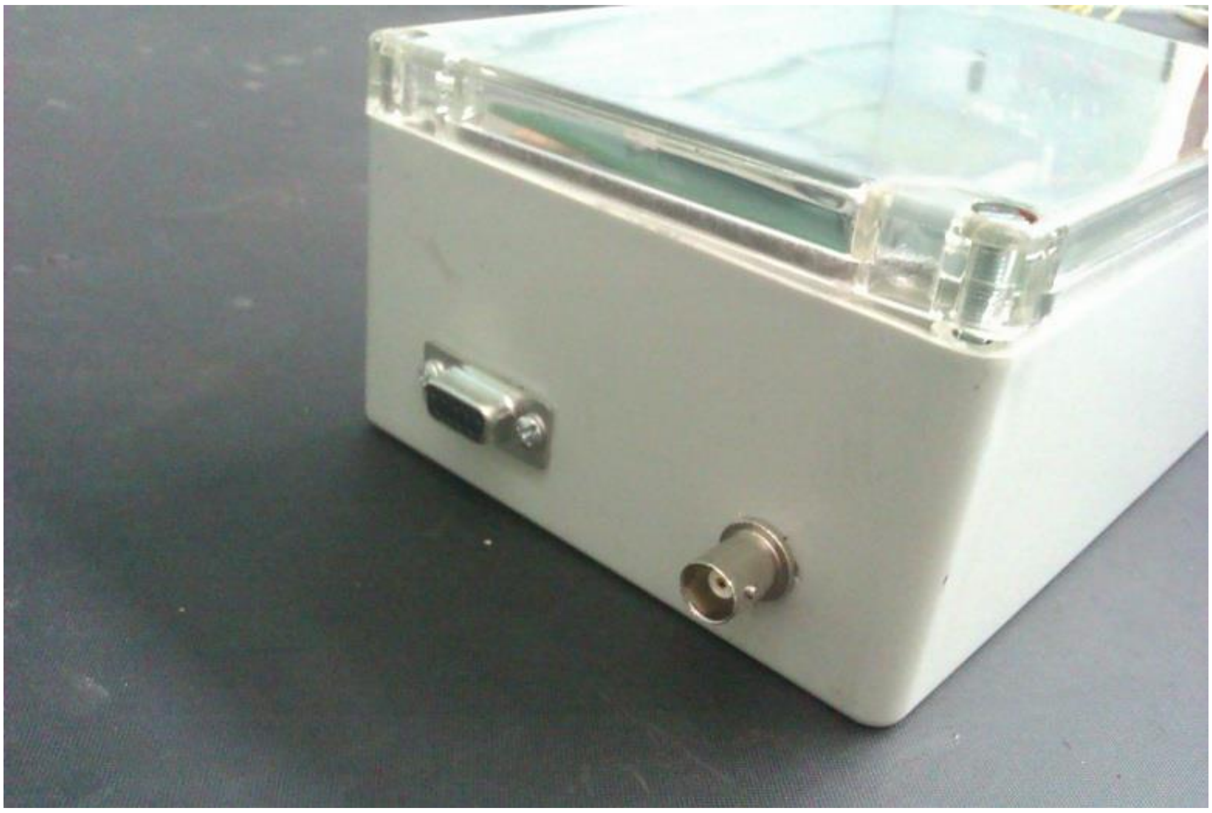


Figure F3: ADM prototype illustrating RS232 and BNC connector

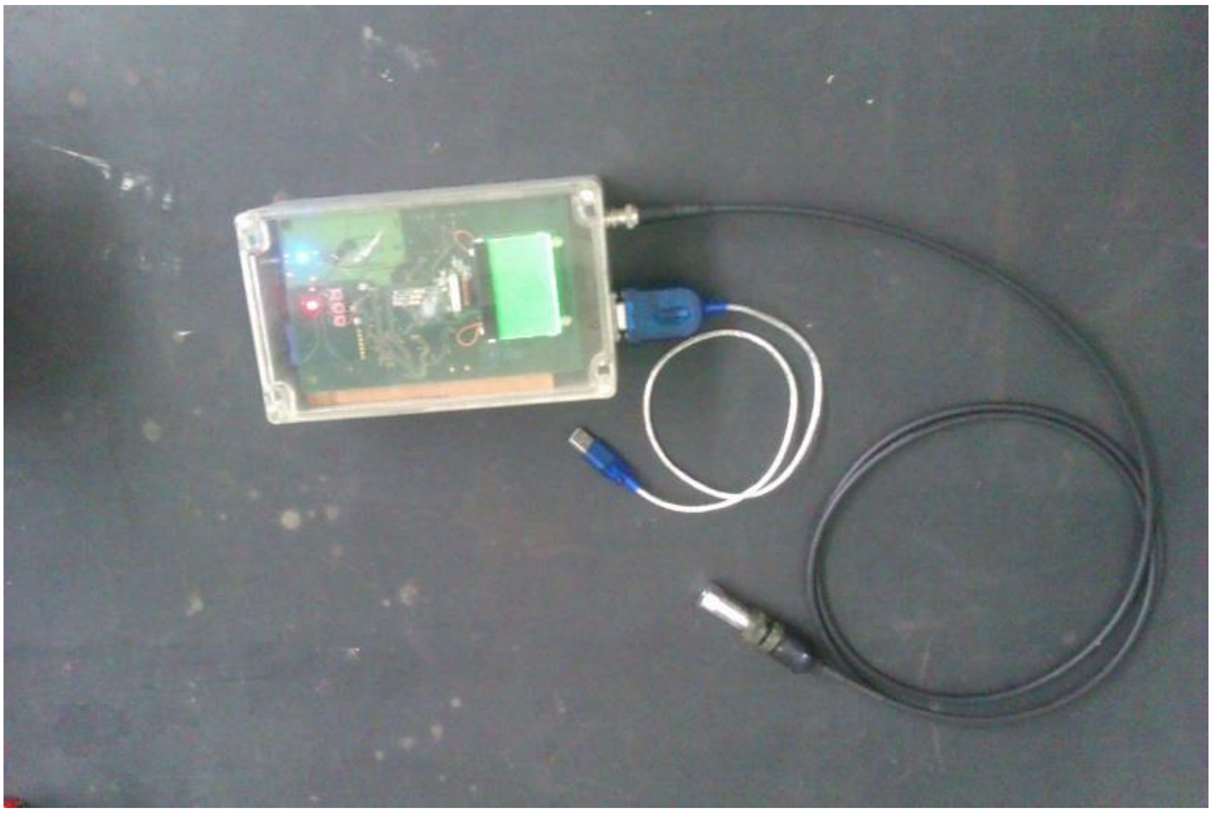


Figure F4: ADM prototype illustrating RS232 and BNC connector with cables

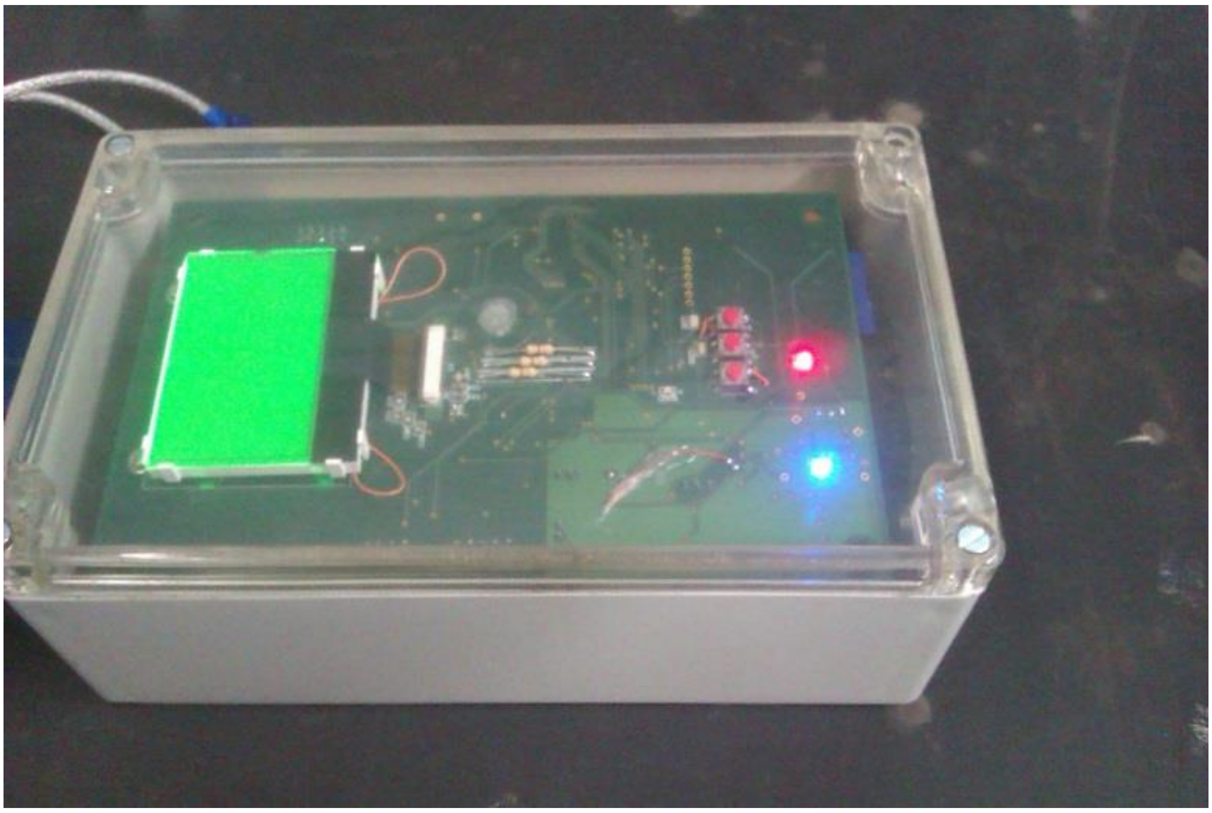


Figure F5: Photo illustrating LCD and LEDs

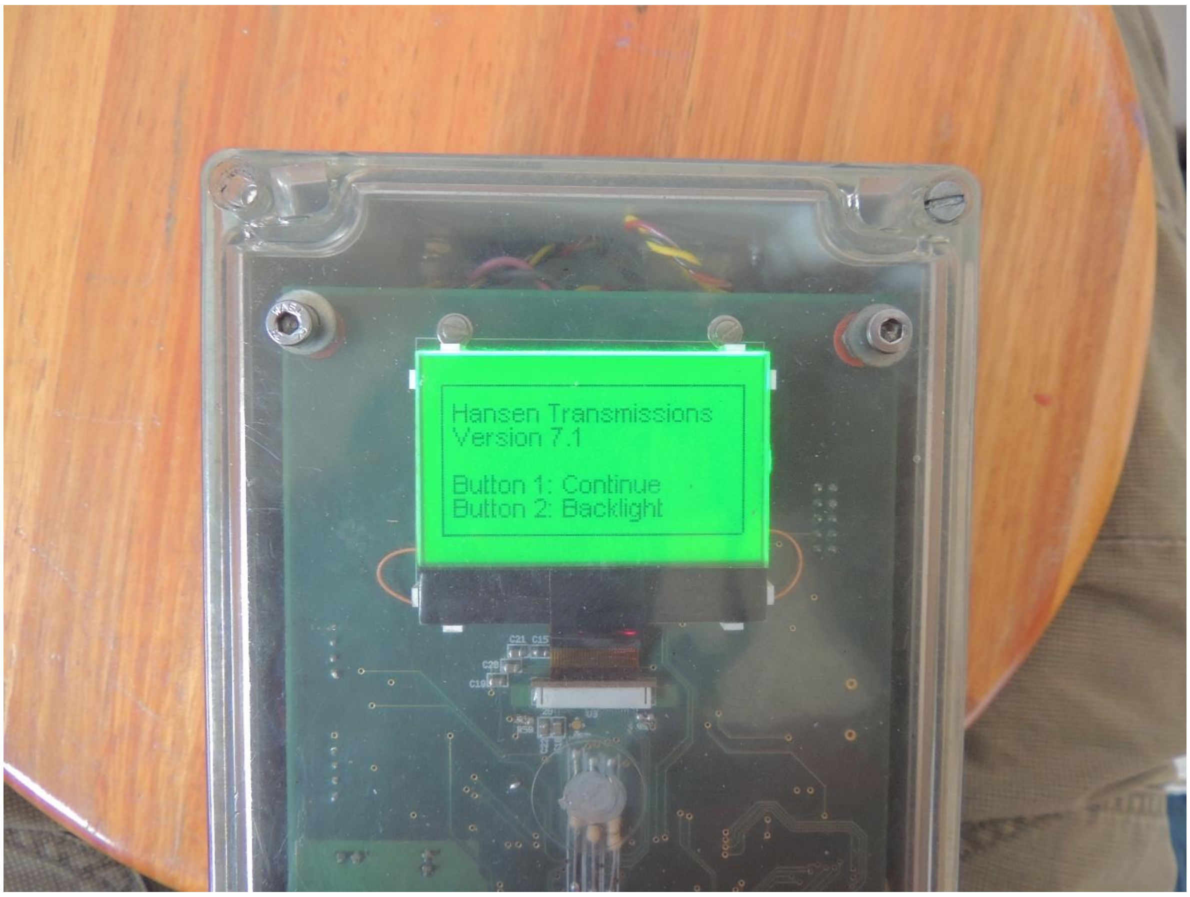


Figure F6: LCD writing

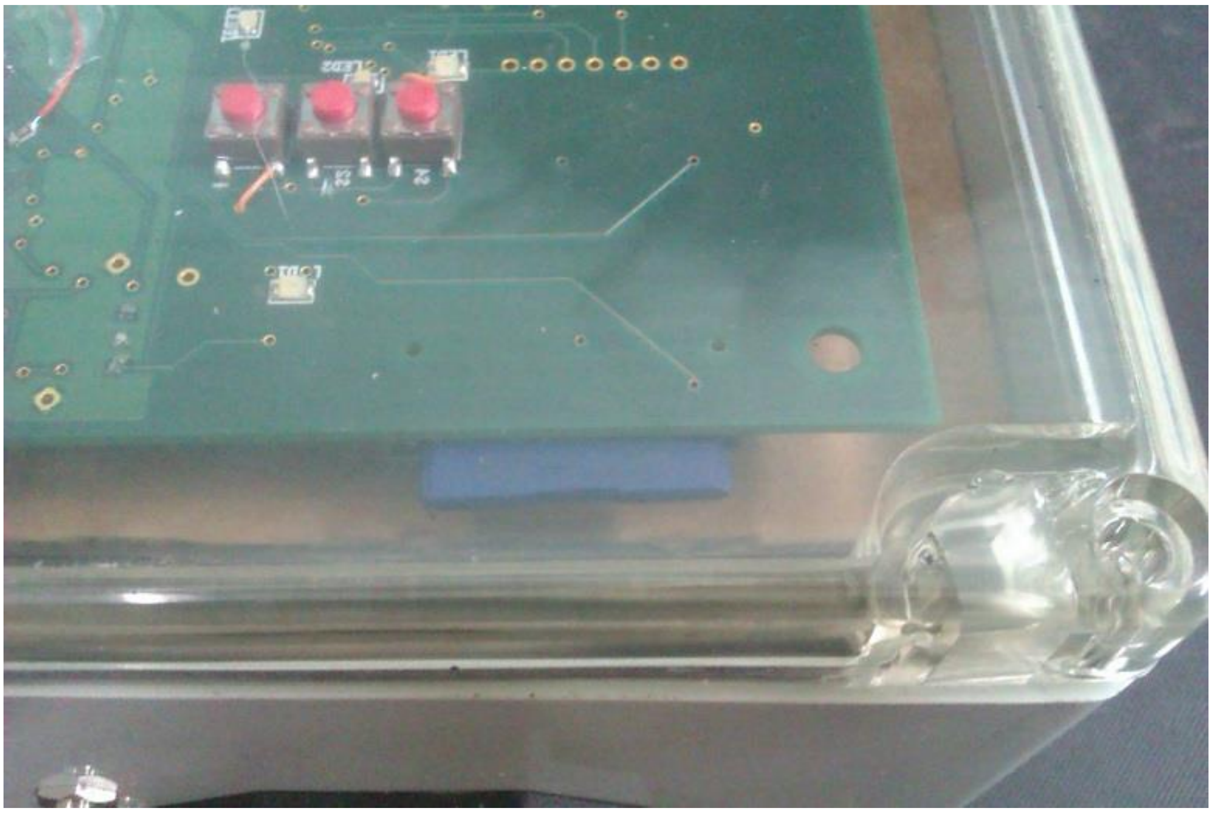


Figure F7: SD card and buttons

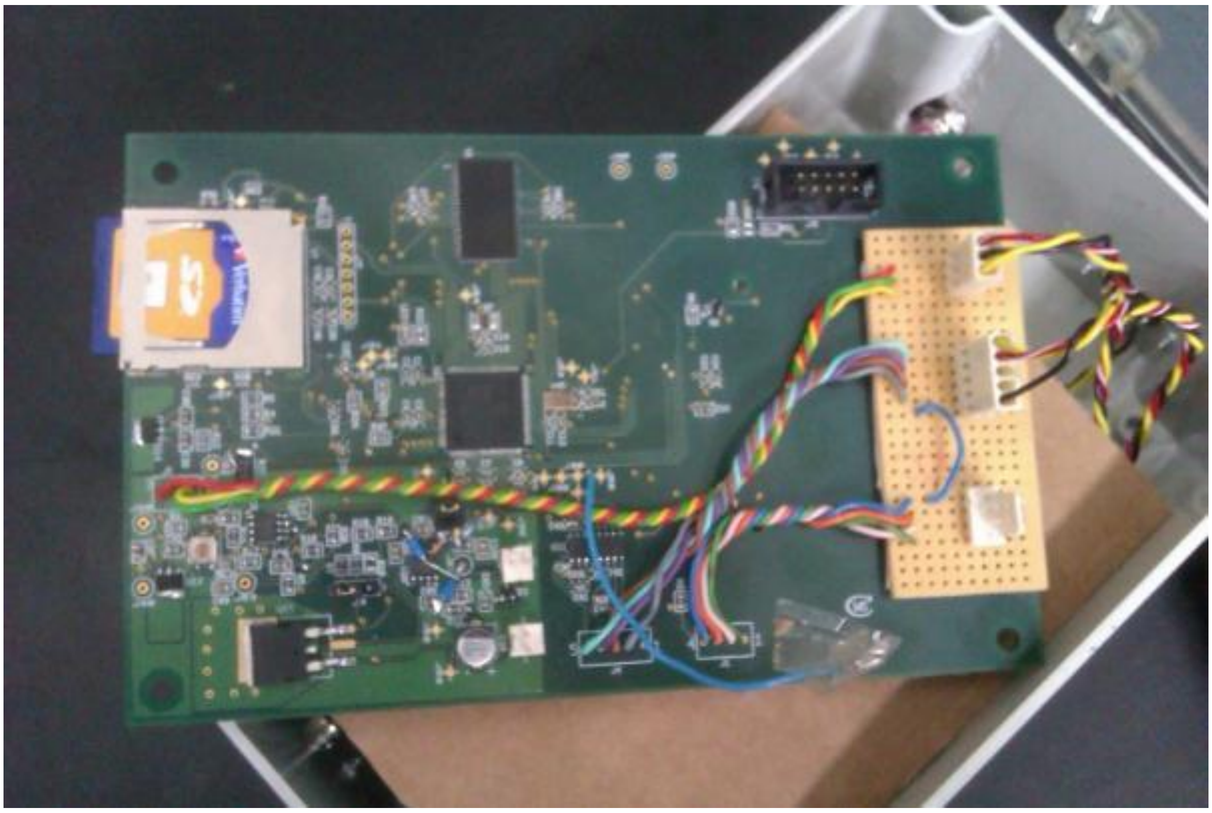


Figure F8: Naked PC board

APPENDIX G – Equipment used

Table G1: Equipment used

Item	Make	Model/Serial number	Specification
EDAQ MEASUREMENT EQUIPMENT			
eDAQ	Somat	762049	N/A
Field test accelerometer	PCB	M627A01 / SN11990	103 mV/g
Field test accelerometer	PCB	M627A01 / SN11991	102 mV/g
Field test accelerometer	PCB	M627A01 / SN1151	105 mV/g
Field test accelerometer	PCB	M627A01 / SN4670	103 mV/g
PROTOTYPE MEASUREMENT EQUIMENT			
Hardware accelerometer	IMI	603C01 / SN178281	95 mV/g
Calibration accelerometer	PCB	M627A01 / SN4670	103 mV/g
LVDT	Instron	PL25N / 2109	10 mm/V