Condition Monitoring for Rotational Machinery

Condition Monitoring for Rotational Machinery

By
Daniel Volante, B. Eng.

A Thesis
Submitted to the School of Graduate Studies
in Partial Fulfilment of the Requirements for the Degree

Master of Applied Science
Department of Computing and Software
McMaster University

MASTER OF APPLIED SCIENCE (2011) McMaster University
(Software Engineering) Hamilton, Ontario

TITLE: Condition Monitoring for Rotational Machinery

AUTHOR: Daniel Volante, B. Eng. (McMaster University)

SUPERVISOR: Dr. Martin v. Mohrenschildt

NUMBER OF PAGES: xii, 101

Abstract

Vibrating screens are industrial machines used to sort aggregates through their high rotational accelerations. Utilized in mining operations, they are able to screen dozens of tonnes of material per hour. To enhance maintenance and troubleshooting, this thesis introduces a vibration based condition monitoring system capable of observing machine operation. Using acceleration data collected from remote parts of the machine, software continuously detects for abnormal operation triggered by fault conditions. Users are to be notified in the event of a fault and be provided with relevant information.

Acceleration data is acquired from a set of sensor devices that are mounted to specified points on the vibrating screen. Data is then wirelessly transmitted to a centralized unit for digital signal processing. Existing sensor devices developed for a previous project have been upgraded and integrated into the monitoring system. Alternative communication technologies and the utilized Wi-Fi network are examined and discussed.

The condition monitoring system's hardware and software was designed following engineering principles. Development produced a functional prototype system, implementing the monitoring process. The monitoring technique utilizes signal filtering and processing to compute a set of variables that reveal the status of the machine. Decision making strategies are then employed as to determine when a fault has occurred.

Testing performed on the developed monitoring system has also been documented. The performance of the prototype system is examined as different fault scenarios are induced and monitored. Results and descriptions of virtual simulations and live industrial experiments are presented. The relationships between machine faults and detected fault signatures are also discussed.

Acknowledgments

I would first like to express my gratitude to my supervisor Dr. Martin v. Mohrenschildt. His guidance and expertise has provided me opportunity to further develop my engineering experience. I would also like to thank my examination committee, Dr. Khedri and Dr. Leduc, for providing me with additional feedback and insight.

I want to acknowledge the engineers and technicians at the sponsoring company. Their industrial knowledge was vital not only to my work, but also in my professional growth.

A special thanks to Dr. Jay Parlar, whose original work was the foundation on which I have built. I also want to thank all the graduate students who resided in ITB 135, as they have offered companionship and encouragement over the years.

Finally, I would like to gratefully thank my significant other, Brianne Nicholls. Her love and support has keep me grounded throughout my studies and has allowed me to flourish.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Thesis Motivation

Vibrating screens are industrial machines used to sort aggregates through their high rotational accelerations. Utilized in mining operations, they are able to screen dozens of tonnes of material per hour. McMaster University's Department of Computing and Software was approached by a manufacture of these machines to develop their next generation of vibration analysis tool. The tool was designed to aid technicians in both maintenance and fault detection of rotating machinery [21]. After four years of development, the product was a system able to measure and analyze three axes of vibration data from eight simultaneous sensors. Technicians are presented with numerical and graphical data on the machine's operation in real time. A post processing software package was also constructed for further analysis.

With the initial project a success, the company wants to expand its vibration analysis tool into a permanently installed condition monitoring system. The new system is intended to monitor the operation of a vibrating screen and notify users when abnormal operation or faults are detected. This would warn users of malfunctions and impending failure events before machines becomes inoperable. Instead of relying on reactionary troubleshooting for maintenance, the vibrating screens would be actively monitored.

It is the intention of the thesis to take the acquired knowledge from the existing vibration analysis tool and apply it to condition monitoring. Previously developed electronics and software strategies are to be modified and integrated into the project. Additional software is to be developmed to complete the remainder of the monitoring system.

1.2 Thesis Objective

The goal of this thesis is to develop and construct an actual condition monitoring system for use on vibrating screens. The system should be able to acquire acceleration data from specified points on a vibrating screen and transmit it to a centralized processing unit. Waveform data is to be filtered so that monitored variables can be calculated, such as peak frequencies and average g-forces. The monitored variables will then be compared against the baseline profile. The baseline is observed when the vibrating screen is in good health, and its profile is to be comprised of the variables.

Monitored variables that deviate from the baseline profile would be indicators of undesired operation or machine faults. Users will specify tolerance thresholds on the

monitored variables and algorithms are used to determine when a fault has occurred. When a machine fault is identified, users are to be notified. The system is also to record vibration data in periodic intervals, as well as data that triggers a fault condition.

In summary, the condition monitoring system should be capable of the following:

- acquire synchronized acceleration data from a set of measurement locations
- detect short time and sustained machine abnormalities by examining vibration waveforms and corresponding frequency content
- notify users of abnormal operation and specify the triggered fault condition
- periodically record the monitored vibration data, and record data pertaining to fault events

1.3 Contributions

Contributions provided this thesis support the completion of the thesis objective in constructing a condition monitoring system for vibrating screens. The following list separates the contributions into individual components:

- design and assemble the hardware for a permanently installed monitoring system
- design and implement software to execute continuous condition monitoring
- system assesment through virtual simulations and live industrial experiments

1.4 Assumptions

The thesis and corresponding condition monitoring system have the following assumptions:

- frequencies of interest appear as resonance frequencies and peak-like in nature
- vibrating screens are subject to Gaussian noise, which is mainly due to material flow

1.5 Thesis Overview

The thesis is divided up into the following chapters:

- Chapter 2 provides a literature review on condition monitoring and fault diagnosis
- Chapter 3 illustrates some computational limitations of the fast Fourier transform (FFT)
- Chapter 4 presents the system overview, a high level summary of the condition monitoring system
- Chapter 5 introduces the existing sensor device used for data acquisition and the modifications made to incorporate it into the current system
- Chapter 6 outlines the design of the condition monitoring software
- Chapter 7 describes the monitoring software implementation
- Chapter 8 covers the monitored variables used to determine the machine's condition in the monitoring process
- Chapter 9 details the condition monitoring technique used to track abnormal operation and detect faults
- Chapter 10 contains various simulation experiments using the condition monitoring software
- Chapter 11 presents condition monitoring system testing on a vibrating screen
- Chapter 12 discusses project conclusions and suggestions for future work

2 Literature Review

2.1 Condition Monitoring

Condition monitoring is a maintenance technique that monitors the condition or health of machinery or structures and advises when upkeep is necessary. It consists of collecting system data through sensor equipment and then processing it into meaningful information. Decision making strategies are then employed to determine when maintenance is required [12]. Fault diagnoses is at the core of condition monitoring. Implemented techniques are utilized to detect fault conditions and even identify specific machines faults.

Due to the wide variety of systems, the literature on condition monitoring is very diverse. Condition monitoring is widespread in industry, with applications in automation, predictive maintenance and quality control [17]. Monitoring has been developed for motors [13, 19], circuit breakers [10], cantilever structures [4], individual bearing components [20, 32] and various other machinery [15, 25, 28]. System information can be extracted from various types of collected data: electrical, vibrational, thermal, environmental, results from oil analysis, etc. [12]. However, as this project focuses on monitoring rotational machinery, vibration data will prove the most valuable. As stated by researchers from the Buckinghamshire Chilterns University College,

> Vibration is probably the most important indicator of the mechanical integrity of rotating machinery. Like the heartbeat of humans, vibration within rotating machinery tells a great deal about the health of that machinery. Machinery health monitoring, through condition monitoring, can detect faults before they become serious, optimize maintenance activities during planned shutdowns and unscheduled outages. [17]

With the recent advances in micro-machining, it is possible to purchase low cost acceleration sensors as to construct remote vibration monitoring devices [34]. Multiple sensors have proved to be much more useful, as a single sensor cannot provide enough data when monitoring a complex system [12]. Understanding the relationships between multiple sets of signals reveals even more information for fault diagnostics [25].

Another consideration of condition monitoring is whether to monitor the target system continuously or periodically. Currently, the most common practice is to monitor in periodic intervals [35]. By choosing to use less data, it is possible to utilize advanced filtering techniques combined with detailed analysis to ensure accurate diagnostics [12]. Models have even been developed to optimize the monitoring intervals

through cost functions [9, 35]. Another technique determines the monitoring interval given the system's risk of failure [8]. Any periodic monitoring, however, has the possibility to miss failure events which could provide insight on the corresponding fault.

Accordingly, an obvious advantage of continuous monitoring is being able to observe all the failure events. In one particular project, the authors wanted to expand the periodic monitoring of a circuit breaker with continuous vibration based monitoring [10]. A clear expectation of the continuous monitoring system was that it must have a false alarm rate much lower than the targeted system's failure rate. Noise proved to be the most challenging issue causing numerous spikes. A proposed method of fast noise filtering was to ignore the vibrations where the existence of noise is predefined. Additionally, a hardware solution suggested the use of differential amplifiers to reduce system noise.

Continuous monitoring projects have been utilizing modern computing, and are optimizing analysis with respect to resolution, frequency bands and computational complexity. Some of these systems are able to monitor rotating machinery by looking for specific indicators of faults [5, 36]. Another monitoring project implemented a continuous wavelet transform, a detailed yet expensive analysis [17]. Development was spent trading off accuracy and frequency ranges for reduce computational complexity. To compensate for the accuracy loss, additional signal enhancing techniques were introduced, primarily chosen for their high efficiency.

2.2 Fault Diagnoses

Fault diagnoses is used to detect and identify machine faults. The first step, fault detection, is used to generate an alarm signal when a fault condition has been breached. This is accompanied by fault isolation and identification; processes which attempt to determine the source and severity of the fault [30].

Machines operating with faults can have reduced efficiency and even accelerated deterioration leading to system failure. Faults can be classified as intermittent or permanent. Intermittent faults persists for a bounded period of time, although can alter system operation after their presence subsides. Once permanent faults manifest, they continue to exist until maintenance is performed [30].

A review on fault diagnoses is presented, divided into four categories. Techniques are classified into signal-based diagnoses, model-based diagnoses, neural network-based diagnoses, and expert systems.

2.2.1 Signal-Based Diagnoses

Signal-based fault diagnoses is the most traditional approach to fault detection. It relies on signal processing of system measurements and its comparison against normal operational trends [30]. Especially in vibration based fault detection, waveform analysis is the most common form of data interpretation. Standardized forms of equations and processing techniques can allow the calculation of recognized variables. The three main categories of waveform analysis are time domain, frequency domain and time-frequency analysis [12].

Different characteristics can describe a time domain waveform such as period, peak, mean and standard deviation [13]. Higher order statistics such as root-mean-square, skewness and kurtosis have been used as well [12]. All of these reveal different aspects of the waveform, however are generally unable to extract all underlying signal information.

Frequency domain analysis begins by converting a time domain waveform into its frequency domain equivalent. The most common means of conversion is through the fast Fourier transform (FFT), as it can perform the transform with ease [5]. Once the frequency spectrum is available, it is then possible analyze to the entire signal or specific frequency regions of interest. With the use of peak detection, dominant frequencies can be identified and singled out for further analysis [5]. When the dominant frequencies of a system have been established, special attention is given to any other emerging frequency content, as it is likely to be the signature of fault. Envelope analysis is the most utilized technique in cases where the fault frequencies are all known or pre-estimated [20]. However, it is not suitable to monitor a wide frequency spectrum or a vibration spectrum when the signal-to-noise ratio is low. Additional characteristics observable in the frequency spectrum are the spacing of sidebands, and the presence of harmonics.

Time-frequency analysis combines both the time domain waveform and the corresponding frequency spectrum. This enables the examination of transient features, such as impacts and fault events, as well the ability to monitor frequency content over time [12]. As a result, time-frequency analysis is the most popular method for non-stationary signals [25]. The Short-time Fourier transform (STFT) is a common technique, where the signal is divided up into short-time segments, and then a FFT is applied to each window [12]. It is computationally efficient, however it provides constant resolution for all frequencies in the window [17]. This makes it difficult to examine both low and high frequency content with high resolution. The Wigner-Ville distribution (WVD) overcomes this resolution limitation, however it suffers from interference terms forced by the transform itself [12]. This can lead to difficult in-

terpretation of the analysis. Improved transforms, such as Choi-Willams distribution and cone-shaped distribution, have been developed to further advance time-frequency analysis [25]. Taken together, while each transform is designed to excel in a specific manner, the required trade-off to do so limits another merit.

The wavelet transform is another form of time-frequency analysis that has been receiving much attention. Like the STFT, the wavelet transform is able to convert time domain waveforms into frequency content over time by use of windowing. Conversely, the wavelet transform uses variably size windows, allowing for the acquisition of better resolutions. [20]. Large time windows are used to obtain precise resolution for low frequencies, while shorter windows give precise time information for high frequencies. The wavelet transform has also received praise for its ability to reduce noise in raw signals [12].

With modern instrumentation and control systems technology, vast amounts of system data can be collected for analysis. While systems can be data rich, they can also be information poor [18]. In these situations, feature extraction techniques are used to deal with excessive amounts of redundant data. One of the most widely used forms of feature extraction is Qualitative trend analysis (QTA). This data-driven technique works by first extracting important features or trends from measurement signals. It then provides the features to a trend interpretation algorithm, where conclusions can be made regarding the health of the system. The Hidden Markov Model is a common algorithm used in conjunction with feature extraction techniques [33].

2.2.2 Model-Based Diagnoses

Model-based fault diagnoses relies on the construction of a mathematical model of the target system. Residual generation techniques capture the differences between the model of a normal system and its current operation. Various residual generation methods can be used with the system models, such as observers, detection filters, parameter identification and parity space [6]. The residuals are then used detect and identify faults.

Different classifications of models can be used to identify a system, such as linear or nonlinear, discrete or continuous, and deterministic or stochastic. Being able to produce an accurate mathematical model is a necessity for this form of fault detection. It can be very difficult or even impossible to build models for complex systems [12].

One paper investigated different modeling techniques for the fault diagnosis of rolling element bearings in rotating machinery. It compared autoregressive modeling techniques, using the Box-Jenkins linear autoregressive model, backpropagation neural networks, and radial basis functions. It was reported that while the back-

propagation neural network proved to be superior in terms of accuracy, this was only true when monitoring slower waveforms. As faster waveforms require higher sampling rates to monitor, the increased number of data points greatly lengthens computational time. One of the largest drawback to predictive modeling is the need for substantial amounts of data to train and test the system model [1].

Another approach to model-based fault detection is in the field of discrete event systems (DES). DES use methods like finite state machines or Petri nets as their modeling formalism. Events change the system state, and track the system history through its progression across the states. Faults are represented in their own states, and are detected when the the system enters a particular fault state [26].

2.2.3 Neural Network-Based Diagnoses

Artificial neural networks (ANNs) are mathematical models inspired by the brain. ANNs have been used in fault diagnoses for residual generation, pattern matching and classification [24]. They are various types of neural-network models, each with their own structure.

ANNs can model complex processes with multiple inputs and outputs. They learn or train by observing the input and outputs, and adjusting internal weights. Training can be achieved through supervised or un-supervised learning. Supervised learning requires a priori knowledge about the system or its outputs. Data from normal operation or faults would be input into the ANN, allowing it to then classify subsequent inputs. In unsupervised learning an ANN would attempt to learn by itself using new available information. ANNs would identify patterns that can be later used for classification purposes [12].

ANNs have been successfully applied to fault diagnosis of rotating machinery. One project relied on ANNs to learn from fault signals and categorize subsequent faults. Typical physical modeling and the current knowledge base was not sufficient enough for alternative methods [2]. Another paper continues to expand ANNs by integrating them with the qualatative approach from fuzzy logic. Aside from fault diagnosis, these hybrid systems have been implemented in the medical and chemical fields [15].

2.2.4 Expert Systems

Fault diagnosis systems that are provided with explicit fault information are considered expert systems. Information is provided in the mapping from the measurement space to machine faults in the fault space. This knowledge base is traditionally complied from field experts through observing and comparing graphical tools such

as frequency spectrums [12]. It is possible to provide an existing knowledge base to a fault detection system, allowing it to identify the given faults from a machine's operation.

The Vibration analysis Expert System (VES) relates physical faults to the frequencies they would emit. The use of digital signal processing enables the system to filter the vibration data and convert it to the frequency domain. Peak frequencies are identified, and then mapped to a fault using confidence factors based on the exact frequency and its corresponding amplitude. Gear faults, bent shafts and overloading were successfully identified with the system [5].

Another expert system, VIBEX, was given vibrations symptoms and the corresponding causes. After typical monitored variables are computed, a Bayesian algorithm is used to obtain confidence factors which could then be used by decision trees for fault diagnosis. Some of the faults included are machine unbalance, misalignment, looseness and bearing damage [36].

Fuzzy logic has been used in expert systems to apply a human-like way of think. It is a multivalued logic that can express system states in a more qualitative manner. Instead of the conventional true (1) or false (0) approach to identifying faults, it is able to provide a degree of defectiveness in a machine. When describing a system, it replaces differential equations with expert knowledge [27]. This technique has been used various systems such as gas turbine engines [14], motors [7] and industrial robots [29].

2.3 Cross-Correlation as a Filter

Cross-correlation has been used in condition monitoring and fault diagnostics in the form of a matched filter or optimal detector. As correlations show the similarity between two signals, it can be an excellent detector of a specific signal when contained within a noisy system [3]. One paper investigated this technique by performing correlations on time domain vibration, current and voltage signals [13]. It was found that the signals from a healthy machine would have strong correlations with other signals from that healthy machine. When a faulty bearing was introduced into the machine, its signals had weak correlations against the healthy machine. Although this would be a useful tool in condition monitoring, it was then has shown that performing the correlation in the frequency domain provides more information regarding the signals' similarities [13]. Accordingly, it is possible to perform frequency domain analysis on the correlated signal to determine the nature of the relationship.

Recent work in vibration fault diagnostics has presented the use of cross-correlation in a novel way. The fault detection system is able to localize machine faults through

the use of multiple sensor. The detection system cross-correlates eight sets of sensor data with one another and converts the results into the frequency domain. Peak frequencies are then detected, which reveal what frequencies are shared amongst the sensors and at what strengths. Any identified peak frequencies not present in a healthy machine are then further examined, as they are likely to be the source of a fault. By observing where the strongest relationships are for a given frequency, it is possible to localize the fault on the machine [21].

A beneficial characteristic of the discussed detection system is that it is computationally efficient. No additional filtering is required, as random noise will not correlate amongst the data sets and be filtered out [21]. This quality has extra importance when being used for continuous condition monitoring, as processing techniques need to be optimized. The system currently exists as a post-processing tool apart of a larger vibration analysis system. The corresponding paper promoted the expansion of the detection system for continuous real time monitoring, as well providing it with the fault knowledge of an expert system.

3 FFT Amplitude Computation

3.1 Overview

The fast Fourier transform (FFT) is a technique used to convert time domain waveform data into the corresponding frequency content. It is useful when examining periodic signals, like the ones produced from the vibrating screens. However due to the nature of the FFT computation, the result is a discrete set of frequencies and amplitudes. Ultimately, contained frequency content is assigned to the closest FFT frequency bin. While this reduces the accuracy of measurable frequencies, it also creates error in the bin's amplitude. The further away a frequency is to its respective bin frequency, the less its contributed amplitude is represented. This form of error is commonly referred to as FFT leakage, since portions of the lost contribution are represented in the amplitudes of neighbouring bins. Since the FFT is to be utilized in the condition monitoring system, this topic will be further examined.

3.2 Numerical Example

To demonstrate the effect of FFT leakage a numerical experiment was performed. The amplitude of the frequency corresponding to the kth bin was observed for various input sinusoidal waves. A wave with frequency of the k-1 bin was first created and then passed into an FFT. As expected, the amplitude of the kth bin remained at zero, since the frequency matches up with the k-1 bin frequency. The input sinusoidal's frequency was slightly incremented and the kth bin amplitude was examined. This repeated until the amplitude of the kth bin returned to zero, as the sinusoidal's frequency matched up with the k+1 bin. Figure 3.1 plots the amplitude of the FFT's kth bin output for the range of input frequencies.

It can be seen that the maximum expressed amplitude occurs when the input frequency is equivalent to the kth bin frequency. The further away the input frequency is from this point, the less its amplitude is expressed in the FFT output.

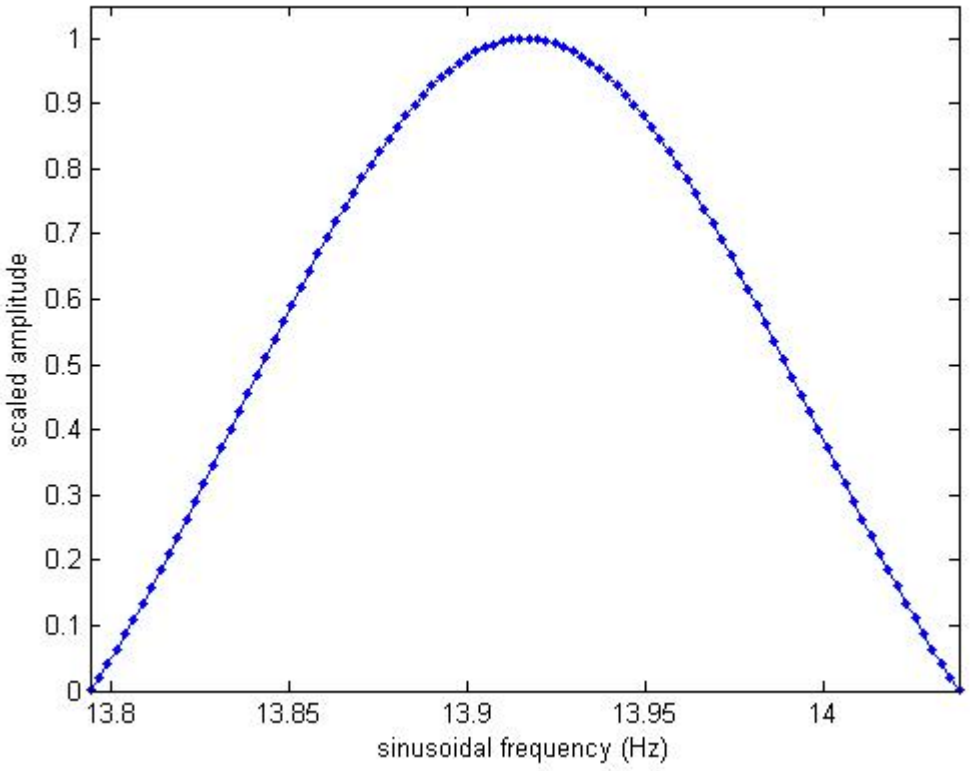

Figure 3.1: FFT kth bin amplitudes

3.3 Theoretical Model

To understand the effect of the FFT leakage, the computation of the discrete Fourier transform is reviewed. The kth bin will be examined, assuming the frequency of the input sinusoidal is within that bin. The sinusoidal can be expressed as,

$$\begin{aligned} x(n) &= \cos(\omega_o n) \\ &= \frac{1}{2}(e^{i\omega_o n} + e^{-i\omega_o n}) \end{aligned}$$

Computing the bin amplitude,

$$\begin{aligned} X_k &= \sum_{n=0}^{N-1} x(n) e^{-ink\omega_s} \\ &= \sum_{n=0}^{N-1} \frac{1}{2}(e^{i\omega_o n} + e^{-i\omega_o n}) e^{-ink\omega_s} \\ &= \frac{1}{2} \sum_{n=0}^{N-1} (e^{i\omega_o n} e^{-ink\omega_s} + e^{-i\omega_o n} e^{-ink\omega_s}) \end{aligned}$$

Ignoring the left hand plane due to symmetry, and simplifying,

$$\begin{aligned} X_k &= \frac{1}{2} \sum_{n=0}^{N-1} e^{i\omega_o n} e^{-ink\omega_s} \\ &= \frac{1}{2} \sum_{n=0}^{N-1} e^{-in(k\omega_s - \omega_o)} \\ &= \frac{1}{2} \sum_{n=0}^{N-1} (e^{-i(k\omega_s - \omega_o)})^n \end{aligned}$$

Using the following identity,

$$\sum_{m=0}^{N} b^m = \frac{1 - b^{N+1}}{1 - b}$$

The summation can be expressed,

$$X_k = \frac{1}{2}(\frac{1-a^N}{1-a})$$

Where,

$$a = e^{-i(k\omega_s - \omega_o)}$$

The simplified X_k expression reveals the amplitude increases as a increases. This is determined by the difference of $k\omega_s$ and ω_o. The closer a frequency is to the center of the bin, the more its amplitude will be properly expressed. When a frequency perfectly aligns with its center bin frequency, a would have a value of 1 (e^0). This would set the denominator of the simplified X_k to zero, setting X_k to infinity or NaN (not a number). This conflicts with the observed result in the previous numerical experiment. A plot of the simplified X_k function for the entire range of the kth bin is presented in Figure 3.2.

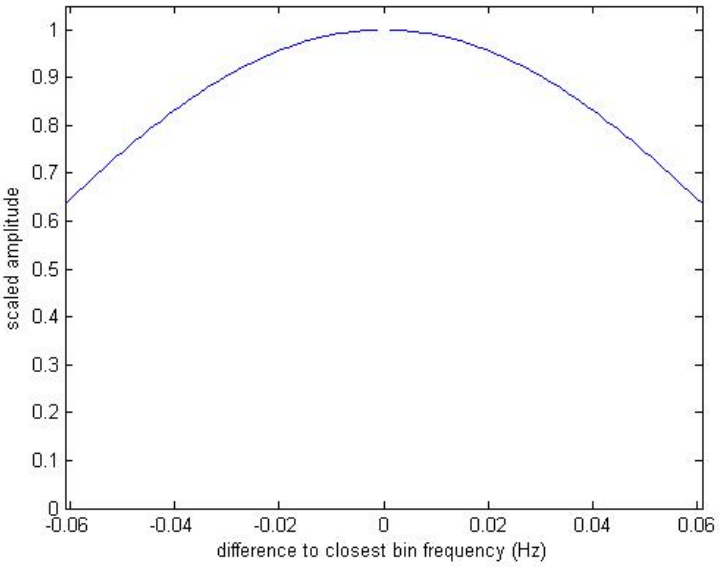

Figure 3.2: Computed kth bin amplitudes

When using the simplified X_k, it should be noted that if a equals 1, then the scaled X_k should assigned a value of 1. Doing so would allow the simplified X_k function to be used for all frequencies within its respective bin.

4 System Overview

4.1 High Level Overview

This chapter presents a high level overview of the condition monitoring system. Sections will discuss the individual components and their functional roles. Figure 4.1 presents physical components and their respective communication methods.

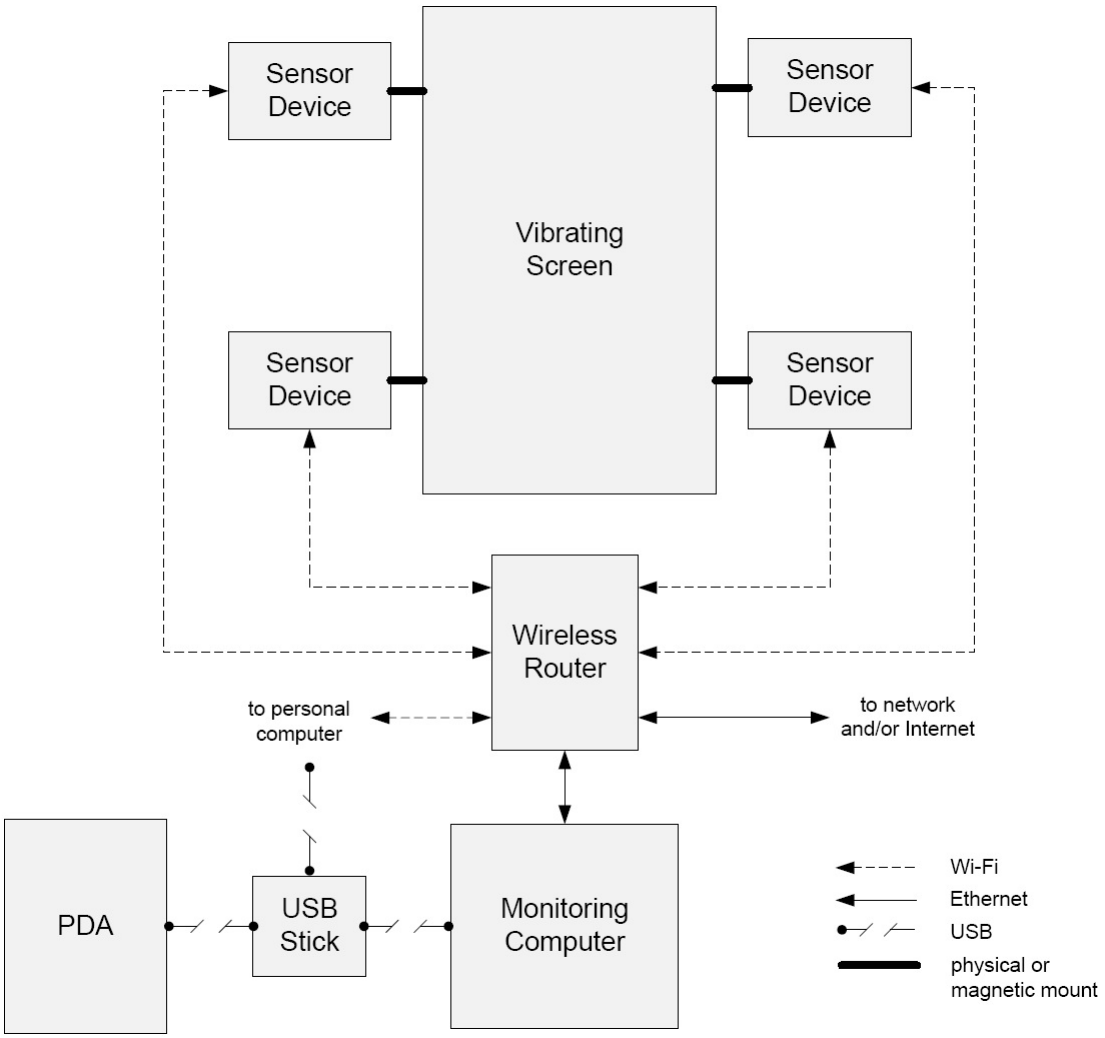

Figure 4.1: High Level System Communication

4.2 Vibrating Screens

The vibrating screen is the monitored plant in the system. These machines screen aggregates through their high rotational accelerations and are able to sort dozens of tonnes of material per hour. Various kinds of vibrating screens exist, operating in either circular, elliptical or linear motions nearing 1000 RPM.

The desired locations to take vibration measurements are depicted in Figures 4.2 and 4.3. Each location has a name associated with it. The first letter, 'L' or 'R', simply represents the 'left' or 'right' side, where designation is given while standing at the feed-end of the machine. The second letter, 'D' or 'F', signifies if the location is at the 'discharge' or 'feed' end. The third letter, 'B' or 'S' indicates whether the location is on the 'body' or 'side-arm' of the machine.

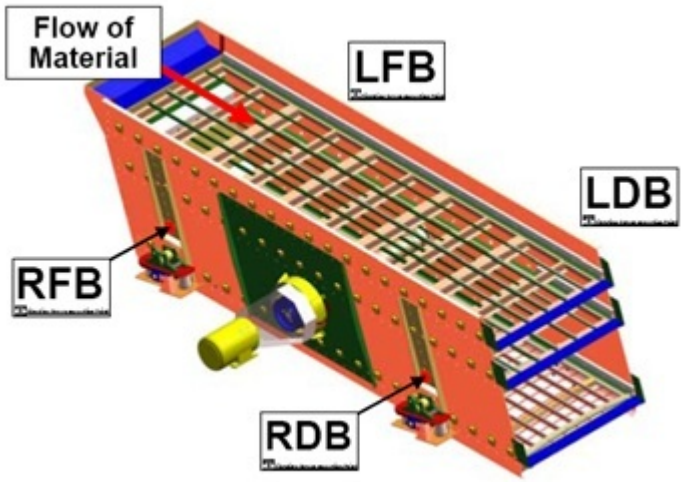

Figure 4.2: Measurement Locations on Two-Bearing Horizontal Screens

4.3 Sensor Devices

The sensor device is an electronic unit developed in-house at McMaster University. It is capable of collecting acceleration data in three axes and transmitting it to a centralized processing unit through a Wi-Fi network (IEEE 802.11). The enclosures are physically or magnetically mounted to the vibrating screen at the desired measurement points. Devices are aligned with the screen to easily allow for a standardized frame of reference.

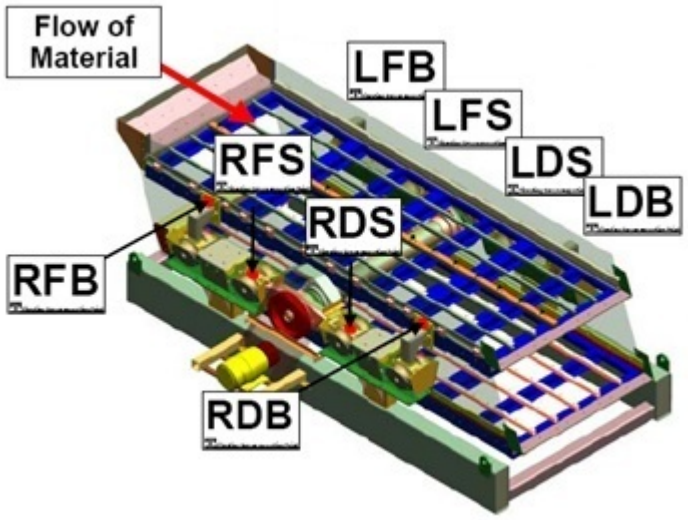

Figure 4.3: Measurement Locations on Four-Bearing Eccentric Screens

The system is able to use all eight measurement locations simultaneously. More sensor devices could be added at the discretion of the wireless router's network stability and the monitoring computer's processing capability. Details of the device are provided in Section 5.

4.4 Monitoring Computer

An industrial computer is used as the main processing unit for the condition monitoring system. The software will continuously monitor the vibrating screen's acceleration data and notify users when abnormal operation is detected. The monitoring software design is presented in Chapter 6, while the implementation is discussed Chapter 7. Ubuntu is chosen as the operating system, primarily for being Linux based and having strong network support.

The monitoring computer is connected to a wireless router, allowing communication with the sensor devices. End users can configure the system remotely through the network, or by a uploading a parameter file generated from the PDA software package. Recorded acceleration data and logs file are saved internally, and can be retrieved remotely or by downloading the files onto a USB stick.

If connected to the Internet, the monitoring software is able to send email notifications to a specified user when machine faults occur. This would also include the

log file of the trigger event. At the end of a monitoring session, the log file can also be emailed so that all triggered fault conditions within the session are known.

4.5 Wireless Router

A standard wireless router is used to support the Wi-Fi network, connecting the monitoring computer to the sensor devices and possibly other networks. Its network capacity should exceed the requirements from the set of sensor devices and other intended wireless access. Section 5.5.2 discusses transmission rates of the implemented sensor devices.

4.6 PDA

Technicians installing and configuring the condition monitoring system have access to a military grade personal digital assistant (PDA). This unit is a key component in a related project, the Vibration Analysis Tool [21]. A software package was created for the PDA so users can select the desired system parameters from a graphical interface and download them on to a USB stick. Users can then upload the file to the monitoring computer, where it is utilized by the monitoring software.

Due to the dust emitting from the vibrating screens, it is undesirable to use conventional electronic devices in the environment. The PDA's sealed touch screen and external buttons allows users to interface with the monitoring system while in harsh conditions.

4.7 USB Stick

As mentioned, it is possible to specify and download the system parameters onto a USB stick using the PDA device, so that it can be uploaded into the monitoring computer. It is also possible to insert the USB stick into the monitoring computer to download the recorded acceleration data and log files produced by the monitoring software.

5 Sensor Devices

5.1 Overview

The sensor devices are the data acquisition units of the condition monitoring system. Mounted at specified points on the vibrating screen, they are able to collect acceleration data and transmit it to a centralized processing unit. The devices were initially developed for a maintenance and troubleshooting tool for vibrating screens. The original device will be briefly discussed, followed by the modifications made to incorporate it into the current monitoring project.

5.2 Existing Devices

The sensor devices were originally used in the Vibration Analysis Tool as the data acquisition units. The device's software and hardware were developed at McMaster University; developmental details are provided in its corresponding paper [21]. The devices are equipped with an accelerometer, allowing them to measure accelerations in three axes up to ± 10 g, where 1 g is equivalent to 9.81 m/s^2. A PIC microprocessor samples the accelerometer at 500 Hz, converting the analog signal into its digital 12 bit representation. The microprocessor then relays the data to the Bluetooth transceiver using RS-232, a point-to-point protocol. Bluetooth wirelessly transmits the data to a PDA, where centralized analysis takes place.

A requirement of the original sensor device was that it must transmit data wirelessly. Devices are magnetically mounted on to vibrating screens for temporary analysis, and then removed when sufficient data was acquired. Wiring would only lengthen the entire maintenance procedure. Also, with the PDA being held by technicians, it is undesirable and against safety regulations to tether the unit to a vibrating screen. As a result of being completely wireless, two AA batteries are used power the device. Figure 5.1 presents the final manufactured prototype sensor device with its enclosure opened.

While the original sensor device was successfully implemented and is still currently in use, it has a few undesired characteristics. These issues arose from the choice of Bluetooth as the wireless technology. A Bluetooth network (piconet) can only support eight devices, including the master. Using the PDA with eight sensor devices would require two Bluetooth networks to sustain the system. The PDA had an internal Bluetooth transceiver, and was equipped with and an additional external one as well to compensate.

The transmission rates of the Bluetooth system were not sufficient enough to

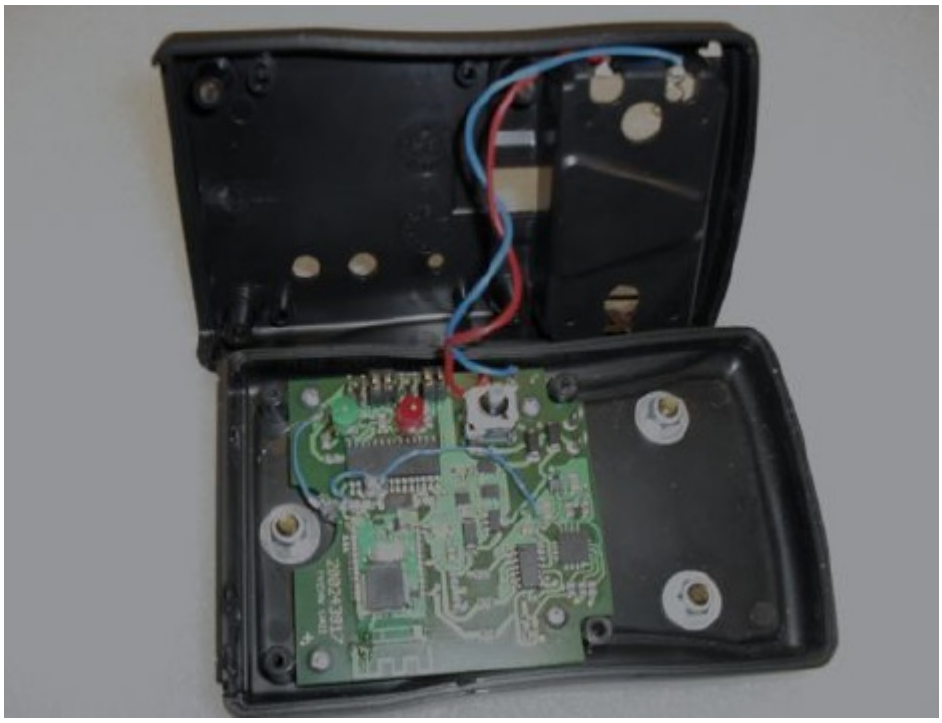

Figure 5.1: Final Manufactured Prototype Sensor Device [21]

support eight sensors sampling acceleration data at 1000 Hz. Sampling was reduced to 500 Hz to significantly reduce the amount of transfered data. The range between the PDA and sensor devices were also of concern, as leaving the immediate area around the machine would disconnect the sensors. It was seen that wireless transmission through the vibrating screens degrades the signal, further reducing the area of the network.

Another Bluetooth issue was the lack of a network broadcast, which would allow the PDA to simultaneously send data acquisition start messages to all devices. It is desirable to have the recorded data from different sensors synchronized so that stronger analysis between sensor data can be achieved. Optimized techniques have only produced a randomized start sequence, where devices have been seen to start up to 3.3 milliseconds apart [21]. To overcome this, the lower level Bluetooth drivers would need to be in accessed as to shorten or regulate the start sequence. Alternatively, additional hardware and software would be required to implement timestamping techniques.

5.3 Requirements

5.3.1 Technical Requirements

The existing sensor devices and devices used in the condition monitoring system have the same technical requirements, as presented below:

- at least eight sensor devices are to be supported by the system
- g-force accelerations are to be monitored in three axes
- g-forces are to be monitored up to ± 10 g, where 1 g is equivalent to 9.81 m/s^2
- frequency spectrums from at least 0.5 Hz to 249.5 Hz are to be examined

5.3.2 Transmission Requirements

As a result of the technical requirements, a minimum transmission rate can be determined. For analysis, eight sensors sampling at 500 Hz will produce the minimum amount of required data. The microprocessor samples three axis of acceleration data, using 12 bit analog to digital conversion. For one device, the transmission rate in bits per second (bps) is:

$$\begin{aligned} \text{Device Rate} &= 500 \text{ (samples/sec)} * 3 \text{ (axes/sample)} * 12 \text{ (bits/axis)} \\ &= 18,000 \text{ bps} \end{aligned}$$

For eight sensors and with no communication overhead, the rate is:

$$\begin{aligned} \text{System Rate} &= 18,000 \text{ (bps/device)} * 8 \text{ (devices)} \\ &= 144,000 \text{ bps} \end{aligned}$$

This produces a minimum transmission rate that must be exceeded to support the sensor network. Networks that marginally exceed this rate are likely to be unstable, as systems should allow ample capacity for communication overhead. This threshold is to be used to dismiss communication technologies that are too slow to meet the bare requirements.

Another requirement placed on the communication system is that it must allow the broadcast of messages. These messages can be received by all devices simultaneously, allowing synchronized device actions to be performed.

5.4 Upgrade Alternatives

In order to satisfy network requirements and further improve transmission rates and distance, new means of device communications were investigated. The new technology had to allow messages to be broadcast, and support a network for at least eight devices and a centralized processing unit. Two main network types were examined: wired and wireless.

5.4.1 Wired Communication

Unlike the original sensor devices, the condition monitoring system does not require the devices to be wireless. It is intended that the devices are permanently installed on to a vibrating screen and so wires could be used for power or communication. While wired networks can be expensive to install on vibrating screens, they allow messages to be broadcast. Since the existing microprocessors are communicating with Bluetooth using RS-232, a multi-point variant, RS-485, was first considered. Modules can be purchased or built to convert the RS-232 to RS-485, and vise-versa.

RS-485, also known as EIA-485, is an electrical standard in defining a multi-point communication network. A maximum of 32 devices can be connected to a single network. RS-485 utilizes differential signaling over twisted pair to provide high noise resistance. Texas Instruments reports that its networks are capable of sending signals up to 2 Mbps when using 50 meter cabling [31].

Ultimately, RS-485 was not chosen because like Bluetooth, more than one network is required to support the eight devices. The limiting factor was the transmission rate of the PIC microprocessor. Testing confirmed that asynchronous rates over 115.2 kbps would result in incorrect data communication [21]. In order to ensure network stability, RS-485 speeds would need to remain under the designated rate. Accommodating the slow speeds requires at least two networks to handle the eight devices. With two networks, the difficulty in obtaining synchronized sensor data is greatly increased. Alternative wired methods then were investigated.

I2C, Inter-Integrated Circuit, is a synchronous multi-point standard able to handle 128 nodes with standard 7 bit addressing. Developed by NXP, I2C networks with buffers are reported to have data rates up to 400 kbps at 100 meters [11]. Two buffers were obtained, a NXP I2C-bus extender and a Texas Instruments dual bidirectional bus buffer. Both chips were tested with the sensor device to determine the I2C network performance.

The NXP I2C-bus extender (P82B715) is a bidirectional buffer with unity voltage gain that can improve the range of an I2C network by increasing cabling impedance.

The chip can operate in networks up to 400 kbps and at 50 meters. A proven quick design multi-point circuit is provided in its datasheet and is presented in Figure 5.2. The design also provides specific information on pull-up resistors given the voltage and number of nodes. The circuit was constructed using 14 meters between nodes in two configurations: master (computer) between two slaves (devices), and master followed by two slaves. Communication up to 400 kbps was achieved in both cases, however the networks were not reliable. Transmission errors were prevalent even after tuning pull-up resistances, transmission rates, packet sizes and delays. The errors would eventually lockup the network, requiring the sensor device's microprocessor to be restarted.

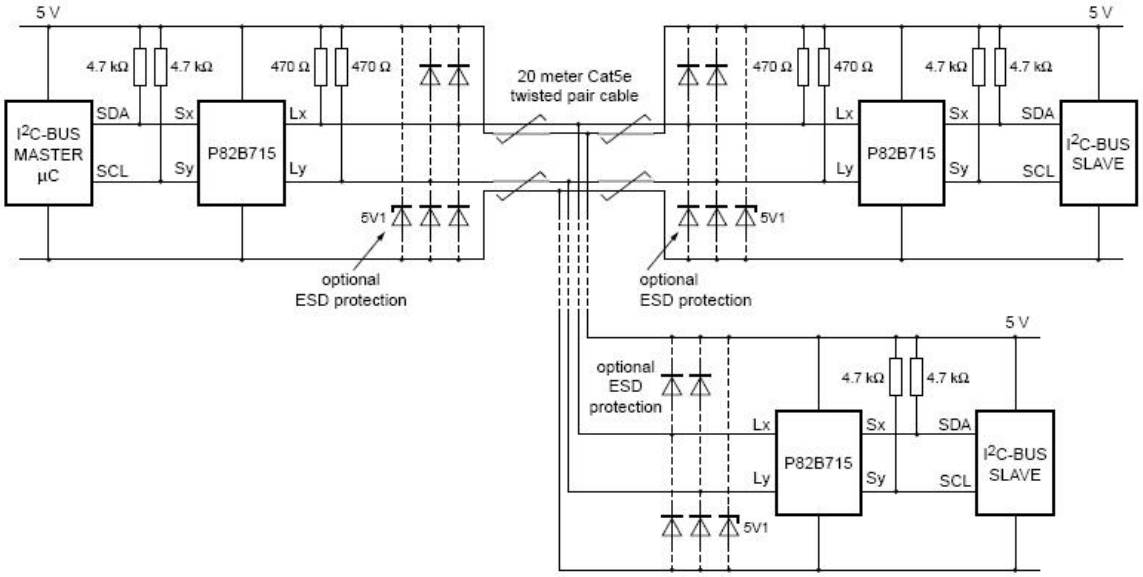

Figure 5.2: Quick Design Multi-Point I2C Circuit

The Texas Instruments dual bidirectional bus buffer (P82B96) can also improve the range of an I2C network, and is able to interface between different logic levels. Sensor devices operating at 3.3 V can have their I2C signals translated up to 15 V, allowing for higher noise resistance. The chip can also translate 15 V signals to the receiving device's logic levels. Like the previous test, the chip was integrated into an I2C network. Communication was achieved, yet network stability was not realized. Even with increased voltage levels, transmission errors led to network failure. Resolving the lockup required the device's microprocessor to be restarted.

To compensate for the I2C network lockup, additional circuitry and software would be required. The network would need to be continuously monitored to ensure that nodes do not hold communication lines at logic low. Nodes doing so restrict any other device from being able transmit data, rendering the network unusable. An automated routine would be required to restart the devices' microprocessors, and notify the network master in such a failure event. However, any communication interruption is undesirable as it is not be possible to perform condition monitoring without sensor data. It was concluded that due to poor reliability and transmission rate of the microprocessor, using a wired network would be unadvantageous.

5.4.2 Wireless Communication

When the existing sensor device was developed, the alternatives to Bluetooth were ZigBee and Wi-Fi. ZigBee was ruled out for being a niche product, as components were difficult to obtain. Wi-Fi was not originally chosen because of its power requirements. While it had high data rates and ranges, the power draw was too much for a wireless device [21]. Technology has advanced over the last few years and new Wi-Fi products are available. As these Wi-Fi transceivers now have comparable power consumption to Bluetooth, they are reexamined as a wireless technology.

The Roving Networks WiFly GSX (RN-131G) is a complete wireless LAN module for embedded systems. The module's size is depicted in Figure 5.3. It is qualified for 802.11b/g networks and is able to achieve 54 Mbps transmission rates. Documentation states that lowering the rate increases range, so tests were performed at 6 Mbps, which is still well above the minimum requirements.

The Wi-Fi module was integrated in to the sensor device for testing. Like Bluetooth, the Wi-Fi module interfaced with the microcontroller using a simple RS-232 connection at 115.2 kbps. An additional button was added to the circuitry so remote configuration could be utilized. When the button is held down on start up, the module would create its own adhoc network. Connecting to this network from Telnet can allow remote configuration of the transceiver.

A computer equipped with a wireless router was responsible for hosting the Wi-Fi network. The sensor devices were able to join the network, and successfully communicate with the host using UDP (User Datagram Protocol). Transmission errors were rare on early prototypes, but the devices could be easily resynchronized by remotely restarting communications. Aside from exceeding the requirements, Wi-Fi was chosen for its robustness and ease of integration.

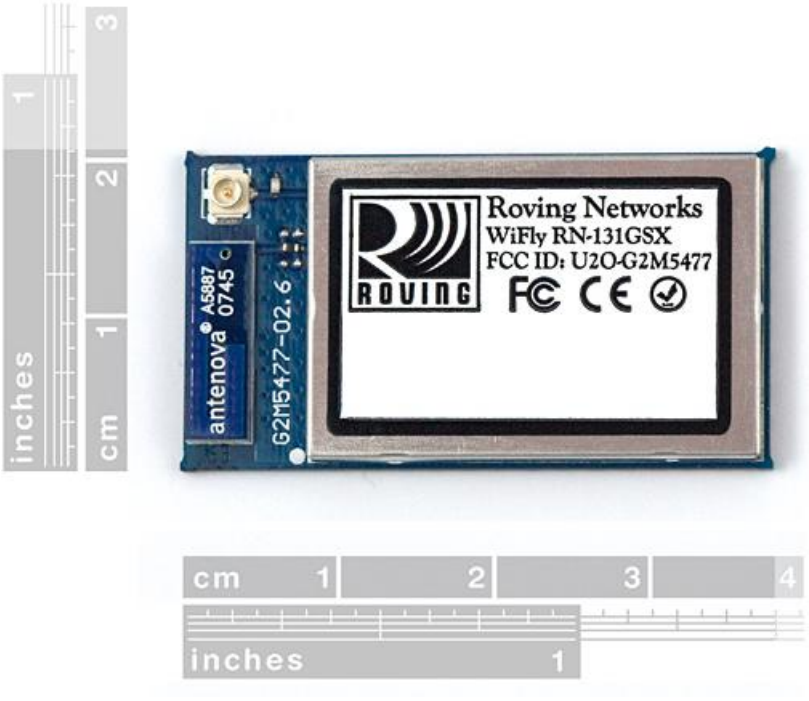

Figure 5.3: Roving Networks WiFly GSX [16]

5.5 Communication Upgrade

Wi-Fi was chosen as the new communication technology, and so the sensor devices were fully integrated with the WiFly GSX modules. The internal functionality of the device remains the same as the existing one, however the communication protocol has been modified. As a result, the revised protocol will be discussed.

5.5.1 Control Protocol

The same control characters from the existing device are used, along with an additional two. The host computer now initiates the wireless communication by requesting an echoed test character. Acknowledgments are also now used after each received packet, as to be aware of data delivery. While brief descriptions are provided in Table 5.1 and in the following paragraph, full functional details are provided in original paper [21].

The start command is simultaneously sent to all devices, commencing their data acquisition. Packets of sampled acceleration data are sent to a centralized unit until

ASCII Character	Description	Host Broadcast	Device Reply
{	Start data acquisition	yes	none
}	Stop data acquisition	yes	0xFFFFFFFF
?	Read calibration values	no	echo
#	Write calibration values	no	echo
^	Initiate/test communication	no	echo
a	Packet acknowledge	no	none

Table 5.1: Control Characters

the stop command has been issued. Devices will send their stop reply if they were acquiring data, but would remain silent if idle.

Calibration values are unique to each device, as they are individually calculated and then internally saved. The calibration procedure is a one time process that produces a mapping from the binary representation of an acceleration sample to its g-force value. Calibration values are read on system initialization and used for signal processing within the monitoring software.

5.5.2 Data Transmission Procedure

Each acceleration datum originates from the accelerometer, is sampled by the microprocessor and is sent to the Wi-Fi module for wireless transmission to a centralized unit. The wireless transmission rate is determined by the rate at which data is being collected and how it is buffered in the device. The structure of transmitted data, known as a data packet, will be also be presented.

The accelerometer is sampled every 500 Hz, producing 12 bit acceleration samples for each of the three axes. Buffered data is stored in bytes (8 bits), and so one acceleration sample is simply stored over two bytes. To reduce transmission overhead, data is sent in sequences of 200 sample triplets. The structure of the packet is controlled by the microcontroller, as presented in Figure 5.4.

The header bytes are used to uniquely identify the start of packets, as data samples cannot exceed 12 bits (0x0FFF). The packet counter is the footer, and is repeated to ensure identification. To detect dropped packets, packets from a device are sequentially numbered and counted. The counter value has modulo of 0xFF, so it remains a one byte value and does not appear as the start header.

The 1200 data sample packet size is chosen with consideration to the WiFly GSX specifications. The chip has two buffers of 1460 bytes; one is for transmitting data,

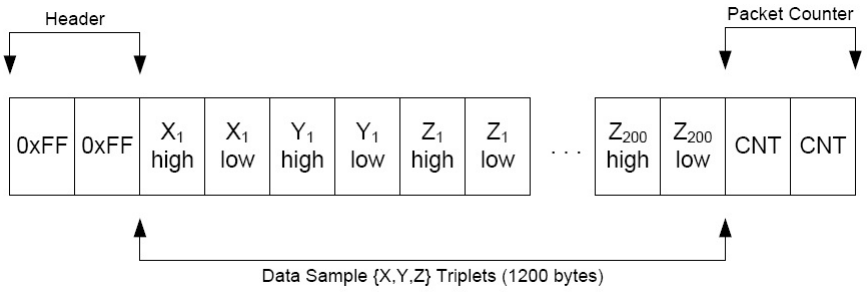

Figure 5.4: Packet Transmission Protocol; each box represents on byte

and the other is for buffering the next packet. Data is transmitted following UDP, a simple transmission model. The Wi-Fi module is configured with UDP Retry, such that it will continually repeat the transmission of a packet if no acknowledgment is received within 250 ms of sending. However, it will begin to transmit the next packet when it is completely buffered.

The time between packets is determined by the amount of data samples within a packet, and the corresponding sampling rate. The amount of time required to collect one packet of data is:

$$\begin{aligned} \text{Packet Period} &= 200 \text{ sample triplets} \,/\, 500 \text{ Hz} \\ &= 400 \text{ ms} \end{aligned}$$

This timing allows for at most one UDP Retry to be utilized. Since the full 1460 byte buffer would not allow more than one opportunity for this packet structure, no additional attempts can be achieved by increasing the size. It should be noted that in order to utilize UDP Retry at 1000 Hz sampling, sampled data would need to be put into larger packets and compressed.

The time between subsequent packets is the duration that the packet is available to be transmitted. If a module is unable to send a packet by the time the next one is completely buffered, the initial packet will be lost. To ensure no data is lost, network capacity should be allocated for each sensor device, the possibility of UDP retry, overhead and additional network usage.

$$\begin{aligned} \text{Device Rate} &= 2 * 1204 \text{ (bytes/packet)} * 8 \text{ (bits/byte)} \,/\, 400 \text{ (ms/packet)} \\ &= 48,160 \text{ bps} \end{aligned}$$

For eight sensors and with no communication overhead, the rate is:

$$\begin{aligned} \text{System Rate} &= 48,160 \text{ (bps/device)} * 8 \text{ (devices)} \\ &= 385,280 \text{ bps} \end{aligned}$$

The implemented sensor devices operate at 6 Mbps, allowing for over 1500% of communication overhead. With the excessive capacity, each device will have ample network opportunity to transmit its packets.

6 Monitoring Software Design

6.1 Design Model

The condition monitoring software design is influenced by Parnas' functional documentation standards [23]. The principle concept is to create a software specification with minimal technical details. Mathematical mappings are defined between different classifications of variables. In our software design these variables are: measured, monitored, internal and controlled. The variable-based model is presented in Figure 6.1.

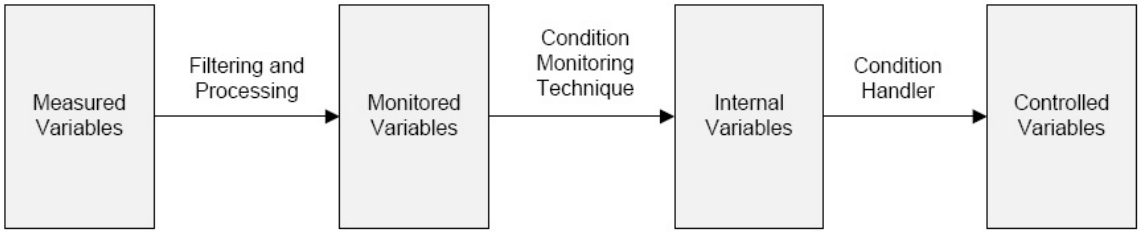

Figure 6.1: Condition Monitoring Software Model

Measured variables are comprised of acquired acceleration data from the sensor devices. Filtering and processing converts the measured data into monitored variables. Collectively, monitored variables depict the profile of a vibrating screen. The condition monitoring technique observes the monitored variables and compares them to a profile obtained during normal operation, known as the baseline profile. The baseline profile is an internal variable, along with the points awarded by the algorithm used to track deviating operation and detect faults. Internal variables are then observed by the condition handler, where controlled variables are set. The controlled variables are connected to information modules so users can be notified at machine fault events.

6.2 Variables

This section lists and describes the variables in their respective classifications. To quickly identify a variable's group, prefixes are used on all names. Measured variables are prefixed by a_, monitored by m_, internal by i_, and controlled by c_.

Variables have N unique instances, where N is the number of sensor devices used in the system. Alternatively, variables reflecting cross-correlated data will have P instances, as to reflect the number of cross-correlated device pairs. The relationship

between N and P is as follows:

$$P = (N-1) + (N-2) + ... + 1$$

6.2.1 Measured Variables

Measured variables are collected acceleration data from the sensor devices. There are N streams of raw data entering the software system. The variable has an equivalent definition for all three axes, as presented in Table 6.1.

Name	Type	Description
$\texttt{a\_accD}_{\texttt{k,k=x,y,z}}$	**integer**	raw acceleration datum

Table 6.1: Measured Variables

The interval between subsequent data from the same sensor and axis is inversely related to the sensor's sampling rate. Acceleration data is to be sampled at 500 Hz, so the interval between subsequent data is 2 milliseconds.

6.2.2 Monitored Variables

The machine's condition is observed through the monitored variables. They are the result of filtering and processing the measured variables. Their implemented definitions are presented Chapter 8. Each of the N sensor devices has a profile depicting the machine's operation. The device profile type is presented in Table 6.2.

Name	Type	Description
`m_mainGForce`	**real**	main g-force
$\texttt{m\_averageGForce}_{\texttt{k,k=x,y,z}}$	**real**	average g-force
$\texttt{m\_operatingFrequency}_{\texttt{k,k=x,y}}$	**real**	operating frequency in Hz
$\texttt{m\_peakList}_{\texttt{k,k=x,y,z}}$	**peakList**	list of peak frequencies

Table 6.2: Monitored Variables in **deviceProfile**

Regarding cross-correlated data, only the peak frequencies are monitored. There are P instances of a cross-correlated profile. Its type is presented in Table 6.3.

The definition of the **peakList** type is given below. It is followed by the elements used in its construction, as presented in Table 6.4.

peakList = sequence of pairs: $\langle$`m_frequency`, `m_amplitude`$\rangle$

Name	Type	Description
$\texttt{m\_peakList}_{\texttt{k,k=x,y,z}}$	**peakList**	list of peak frequencies

Table 6.3: Monitored Variables in **crosscorrelatedProfile**

Name	Type	Description
`m_frequency`	**real**	peak's frequency in Hz
`m_amplitude`	**real**	peak's scaled relative amplitude

Table 6.4: Monitored Variables in **peakList**

A final type definition is given, encompassing all instances of the monitored variables. The `m_machineProfile` variable is type **machineProfile**, and contains all relevant machine information from the individual devices and cross-correlated pairs. Its elements are presented in Table 6.5.

machineProfile = sequence of profiles,
($\texttt{m\_deviceProfile}_1$, ... , $\texttt{m\_deviceProfile}_N$,
$\texttt{m\_crosscorrelatedProfile}_1$, ... , $\texttt{m\_crosscorrelatedProfile}_P$)

Name	Type	Description
$\texttt{m\_deviceProfile}_{\texttt{n,n=1...N}}$	**deviceProfile**	device profile
$\texttt{m\_crosscorrelatedProfile}_{\texttt{n,n=1...P}}$	**crosscorrelatedProfile**	paired profile

Table 6.5: Monitored Variables in **machineProfile**

Some of the monitored variables make use of a frequency spectrum, which requires specific mathematical computations such as the FFT. An axial sequence of data is required for the calculation, and a of length 2^x terms optimizes its performance. A sequence of 4096 samples will be used, where samples are acquired at 500 Hz. As a result, monitored variables will be updated at the following rate:

$$\begin{aligned}\text{Monitored Variable Update Rate} &= 4096 \text{ samples} / 500 \text{ Hz} \\ &= 8.192 \text{ sec}\end{aligned}$$

6.2.3 Internal Variables

Internal variables are affected by the monitored variables through the condition monitoring technique. One such internal variable, the baseline profile, has the **ma-**

chineProfile type as defined in the previous section. This profile contains the normal machine operational signatures used to designate healthy conditions. Figure 6.6 defines the internal variables.

Name	Type	Description
`i_baseProfile`	**machineProfile**	profile of baseline operation
$\texttt{i\_incidentPoint}_{\texttt{m}}$	**real**	numerical indicator of irregularity
$\texttt{i\_faultFlag}_{\texttt{m}}$	**boolean**	flag indicating a fault condition

Table 6.6: Internal Variables

The $\texttt{i\_incidentPoint}_{\texttt{m}}$ variable contain incident points, indicators of operational irregularity for a specific monitored variable. The $\texttt{i\_faultFlag}_{\texttt{m}}$ flag has a Boolean value indicating a breached fault condition for a specific monitored variable. Every monitored variable with a **real** type has these two corresponding internal variables. The individual sets of the two internal variables have a bijective relationship with the set of all **real** monitored variables. All other monitored variables are populated by these **real** values.

The `i_baseProfile` is only instantiated once, and then updated at the discretion of the user. Changing machine functionality would alter its baseline, and so a new baseline profile would be needed. The other internal variables are to be updated at the same rate of the monitored variables. No new decisions can be made without the new information provided by an updated set of monitored variables.

6.2.4 Controlled Variables

The controlled variable in the condition monitoring software is used to indicate a fault event, as presented in Figure 6.7. It is influenced by the internal variables and modified by the condition handler. The controlled variable is to govern a display module, as to communicate the presence of a machine fault to users.

Name	Type	Description
`c_faultDetect`	**boolean**	flag indicating fault conditions

Table 6.7: Controlled Variables

The controlled variable will complete its update cycle at the rate the internal variables are being updated. It can be modified at any time when the condition handler detects a fault condition, or a return to normal operation.

6.3 Functions

Functions in the model are given a set of variables, and produce the subsequent set of variables. Functions and their requirements are discussed in this section.

6.3.1 Filtering and Processing

The purpose of filtering and processing is to produce the monitored variables from measured variables. Triaxial acceleration data from a sensor device, $\texttt{a\_accD}_{\texttt{k,k=x,y,z}}$, are used in sequences to instantiate a **deviceProfile** type. Two synchronized sequences of data from different sensors are used for the **crosscorrelatedProfile**. Input and output relationships are presented below.

A new type definition is given, as to reflect the sequences of triaxial acceleration data. An **accBuffer** type is defined as:

$$\textbf{accBuffer} = \text{three sequences: } [\langle \texttt{a\_accD}_{\texttt{x}} \rangle, \langle \texttt{a\_accD}_{\texttt{y}} \rangle, \langle \texttt{a\_accD}_{\texttt{z}} \rangle]$$

For each sensor device used, an **accBuffer** is instantiated. Each acquired data sample will belong to one of the N **accBuffer**, in a surjective mapping. Every datum has two samples corresponding to the same time and sensor location, but unique regarding axis. When enough data has been acquired in an **accBuffer**, the contents are filtered and processed, emptying the buffer for the next sequence of upcoming data samples.

All monitored variables are contained in a **machineProfile** type, within the **deviceProfile** or **crosscorrelatedProfile** type. As a result, defining the relations to these two types provides a mapping from the measured to monitored variables.

- The set of all instantiated **deviceProfile** types has a bijective relation to the set of all instantiated **accBuffer** buffers
- The set of all instantiated **crosscorrelatedProfile** types has a bijective relation to the set of all instantiated $\{i, j\}$ **accBuffer** pairs, where $i \neq j$

6.3.2 Condition Monitoring Technique

The condition monitoring technique observes the monitored variables and updates the internal variables to track the machine's condition and faults. Monitored variables are first compared to their baseline profile equivalent, and if the difference exceeds their respective tolerance threshold, ϵ, an incident event occurs. The presence or absence of an incident event pertaining to an a monitored variable is then passed

along to the remainder of the monitoring process. The condition of a monitored variable is tracked using the incidents, and is responsible for distinguishing between machine faults and normal operation.

The existence of an incident will be expressed as a Boolean, in the `incidentFlag` flag. Each of the monitored variables with a **real** type are used to check for corresponding incidents, as defined below.

For a sensor n $[1, N]$,

$$\begin{aligned}&\forall m : (m, \_, \_, \_) \in \texttt{m\_machineProfile},\\&(\exists m_b : (m_b, \_, \_, \_) \in \texttt{i\_baseProfile},\\&(|m_b - m|/m_b \geq \epsilon_m) \Leftrightarrow \texttt{incidentFlag})\end{aligned}$$

For a sensor n $[1, N]$, and for an axis k $\{x, y, z\}$,

$$\begin{aligned}&\forall g : (\_, g, \_, \_) \in \texttt{m\_machineProfile},\\&(\forall g_b : (\_, g_b, \_, \_) \in \texttt{i\_baseProfile},\\&(|g_b - g|/g_b \geq \epsilon_g) \Leftrightarrow \texttt{incidentFlag})\end{aligned}$$

For a sensor n $[1, N]$, and for an axis k $\{x, y\}$,

$$\begin{aligned}&\forall p : (\_, \_, p, \_) \in \texttt{m\_machineProfile},\\&(\exists p_b : (\_, \_, p_b, \_) \in \texttt{i\_baseProfile},\\&(|p_b - p| \geq \epsilon_p) \Leftrightarrow \texttt{incidentFlag})\end{aligned}$$

For a sensor n $[1, N]$ or cross-correlated pair p $[1, P]$, and for an axis k $\{x, y, z\}$,

$$\begin{aligned}&\forall l : (\_, \_, \_, l) \in \texttt{m\_machineProfile}, \forall \langle f, a \rangle \in l,\\&(\neg\exists l_b : (\_, \_, \_, l_b) \in \texttt{i\_baseProfile}, \forall \langle f_b, a_b \rangle \in l_b,\\&((|f_b - f| < \epsilon_f) \wedge (|a_b - a|/a_b < \epsilon_a)) \Leftrightarrow \texttt{incidentFlag})\end{aligned}$$

After the `incidentFlag` is assigned a Boolean value, the respective modifications are made to the corresponding $\texttt{i\_incidentPoint}_\texttt{m}$ or $\texttt{i\_faultFlag}_\texttt{m}$ internal variables. While there are no specific requirements on this process, the set of all $\texttt{i\_incidentPoint}_\texttt{m}$ contains the indicators representing the incident events. The set of all $\texttt{i\_faultFlag}_\texttt{m}$ flags identifies when the algorithm has identified a fault condition on a specific monitored parameter.

6.3.3 Condition Handler

The condition handler modifies the controlled variables in response to the internal variables. The detection of any one fault, corresponding to a particular monitored variable, will activate the machine fault flag. The Boolean value of this controlled variable, `c_faultDetect`, is simply defined as follows.

$$(\exists u \in \texttt{i\_faultFlag}_{\texttt{m}},\ u) \Leftrightarrow (\texttt{c\_faultDetect})$$

7 Monitoring Software Implementation

7.1 Overview

The condition monitoring software is a required subsystem apart of the monitoring project. The software supports the completion of the thesis objectives, Section 1.2, and follows the framework provided in its design model, Chapter 6. The software is implemented in the C programming language and is executed by the system's monitoring computer.

The program has a command line interface that allows users to initiate the monitoring process and auxiliary modules. A PDA application has been developed to allow users to generate a parameter file containing information and settings, which is then uploaded via USB stick to the monitoring software.

Figure 7.1 presents the main modules that comprise the software, and the corresponding data flow between them. Modules will be listed in a module guide, presented with their software loops, and have individual details discussed in the following sections.

7.2 Module Guide

This section contains a Module Guide of the software system as defined by Parnas [22]. Its purpose is to ensure separation of concerns and assist in future maintenance of the modules. Parnas wrote,

> It defines the responsibilities of each of the modules by describing the design decisions that will be hidden (encapsulated) by that module (its secrets). [22]

The name, service and secret of each module are presented below. The Module Guide spans across Tables 7.1, 7.2 and 7.3.

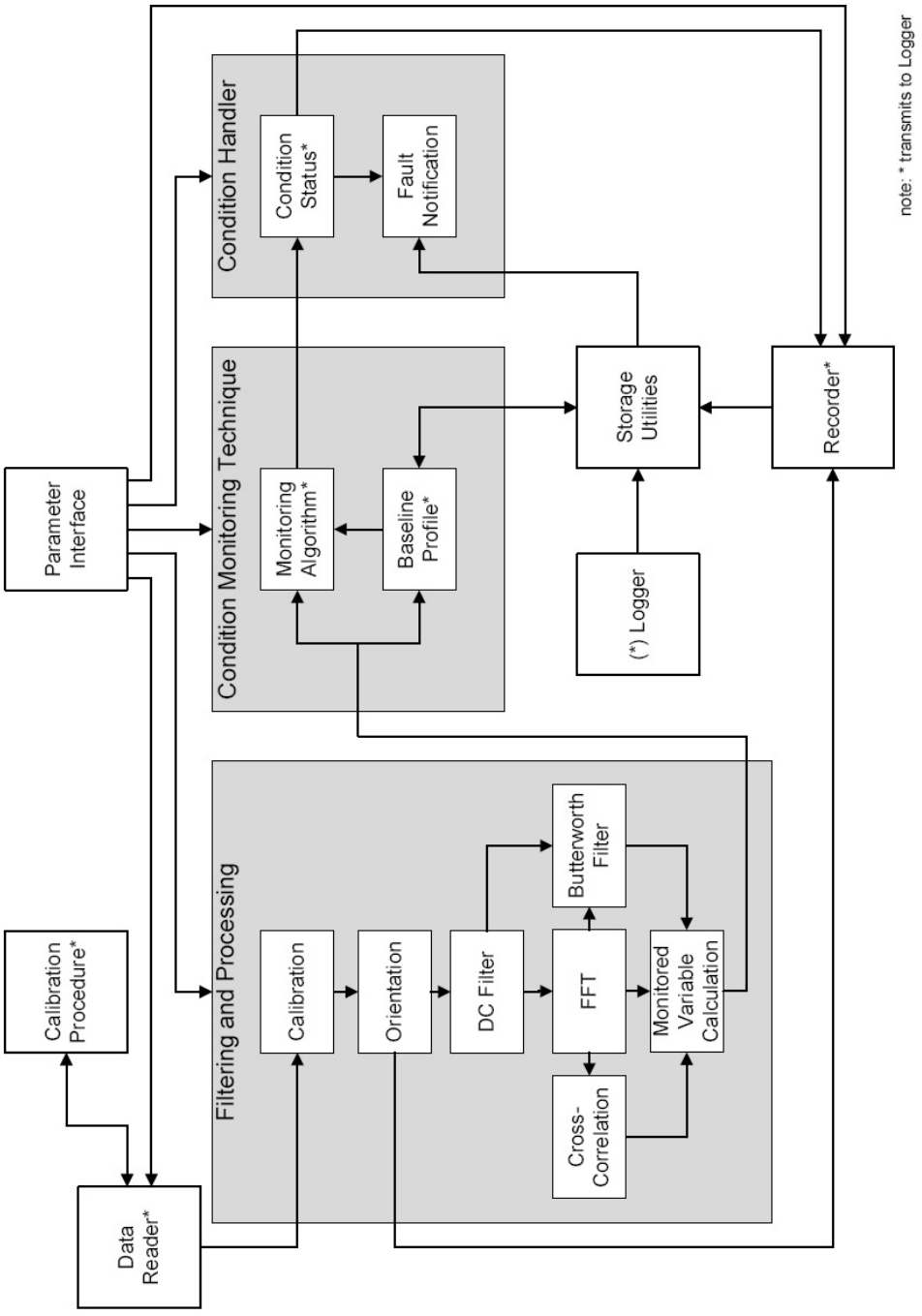

Figure 7.1: Software Module Data Flow Diagram

Name	Parameter Interface
Service	Acquire and distribute system parameters
Secret	Parameter acquisition and distribution implementation

Name	Data Reader
Service	Receive transmitted sensor device data
Secret	Techniques and protocols for handling device communication

Name	Filtering and Processing
Service	Filter and process raw sensor data into monitored variables
Secret	Filtering, processing and calculation techniques

Name	Calibration
Service	Convert raw data into g-force values
Secret	Mappings from raw data to g-forces

Name	Orientation
Service	Orient data to a standardized frame of reference
Secret	Orientation implementation

Name	DC Filter
Service	Remove DC (constant) components from waveform data
Secret	DC Filter coefficients

Name	FFT
Service	Calculate frequency spectrums
Secret	Algorithm parameters and implementation

Table 7.1: Module Guide - part 1 of 3

Name	Butterworth Filter
Service	Adaptive bandpass filter
Secret	Filter coefficient and implementation

Name	Cross-Correlation
Service	Cross-correlate frequency spectrums
Secret	Implemented cross-correlation algorithms

Name	Monitored Variable Calculation
Service	Calculate monitored variables
Secret	Utilized equations and algorithms

Name	Condition Monitoring Technique
Service	Detect deviating monitored variables
Secret	Monitoring technique

Name	Baseline Profile
Service	Calculates the baseline profile
Secret	Baseline profile calculation technique

Name	Monitoring Algorithm
Service	Track and detect deviating monitored variables
Secret	Implemented monitoring algorithm

Name	Condition Handler
Service	Detect and respond to machine faults
Secret	Machine fault detection and response procedures

Table 7.2: Module Guide - part 2 of 3

Name	Condition Status
Service	Detect machine faults from internal variables
Secret	Fault detection algorithm

Name	Fault Notification
Service	Notify users of machine fault conditions
Secret	Notification process

Name	Recorder
Service	Record data periodically and on fault events
Secret	Implemented recording procedure

Name	Logger
Service	Log system events to file
Secret	Implemented logging procedure

Name	Storage Utilities
Service	Save and retrieve data and files
Secret	Implemented save and load procedures

Name	Calibration Procedure
Service	Calculate calibration values and save to device
Secret	Implemented calibration technique

Table 7.3: Module Guide - part 3 of 3

7.3 Software Loops

While the software was developed on a single-core processor, it utilizes multiple loops to perform its repeated tasks. Each loop is executed as its own thread. The loops and associated modules are presented in table 7.4.

Loop	**Name**	**Module**
1	Main	Parameter Interface
2	Reader	Data Reader Logger
3(a)	Process	Filtering and Processing Condition Monitoring Technique Condition Handler Recorder Storage Utilities Logger
3(b)	Calibrate	Calibration Procedure Logger

Table 7.4: Software System Loops

The Main loop gathers and distributes system parameters, and is responsible for handling the command line interface. Parameter updates, calibration procedures, and monitoring are initiated in this loop. During monitoring, the loop runs to ensure users may quit the application when desired.

The Reader loop is dedicated to the Data Reader module, as to ensure timely reception of incoming acceleration data. The Logger module is accessed as the discretion of the Data Reader module when it requires messages to be logged. This loop is constantly executed, as to be prepared for new incoming data. Through a circular queue, collected data is forward to one of two next loops.

The Process loop is the main operational loop which contains filtering, processing, and monitoring techniques. Auxiliary modules are also executed in this loop, such as the recording and saving of data. This loop is only executed once it has buffered a time window's worth of data for a sensor device. The last device to provide its time window's worth of data initiates the cross-correlated calculations, which require data sets from all the device pairs.

The Calibration loop is used in place of the Process loop when the system is being used to calibrate new sensors. The Calibration Procedure module contains the required methods to utilize sensor data for the calculation of calibration values.

7.4 Filtering and Processing

Filtering and Processing contains the procedures to manipulate raw acceleration data into meaning information. The result is a collection of monitored variables, specific to individual devices and device pairs. Since the frequency spectrum of the sensor devices' axes are computed every time window, all monitored variables are updated at the same rate. This simplifies the continuous monitoring process into the monitoring of discrete time windows. As a result, every module will process data in buffers corresponding to the amount of acceleration data pertaining to a single time window.

Raw acceleration data is introduced into the software by the Data Reader module, and then stored into circular queues specific to a device and axis (X, Y, Z). The Filtering and Processing module buffers this data into sized sequences, and then passes the data through the various filters and processing techniques.

7.4.1 Calibration

At system initialization, the Calibration module is provided with calibration values unique to each sensor device and axis. Since each device stores these values internally, they are acquired by the Data Reader module. Each sensor axis has two calibration values associated with it, m_k and b_k. This provides a linear mapping from a raw acceleration datum to a g-force value.

$$y(n) = m_k x(n) + b_k$$

The stored calibration values are calculated within the Calibration Procedure module. Details of this calibration technique are outlined in Section 7.7.6.

7.4.2 Orientation

All sensor devices have the same three dimensional Cartesian coordinate system. Devices are mounted with their X-axis parallel with the direction of material flow, however mounting on different sides of the vibrating screen and at different device orientations (side or top) creates different device frames of reference. The Orientation module is responsible for transforming each device's frame of reference to the standardized machine frame of reference. Figure 7.2 presents the vibrating screens frame of reference. The utilized transformations are summarized in Table 7.5.

After acceleration data have been orientated, it is forward to the DC Filter module for further processing. A copy of the data is provided to the Recorder module so that

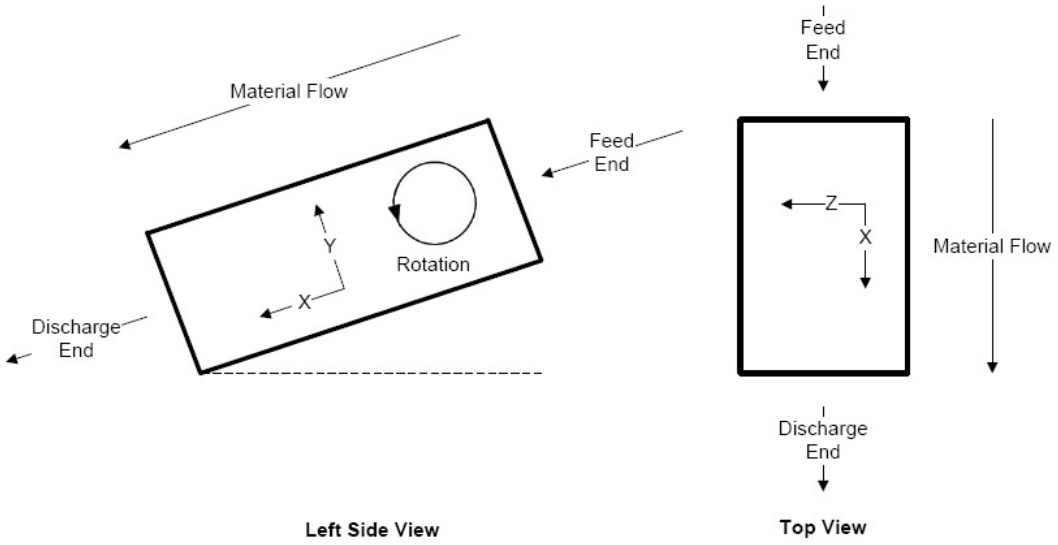

Figure 7.2: Vibrating Screen - Frame of Reference

Machine Side	Device Orientation	Transformation
right	side or top	negate X-axis
left	side	negate Z-axis
right	top	flip -Y and Z-axis
left	top	flip Y and Z-axis

Table 7.5: Sensor Device to Vibrating Screen Frame of Reference

the data can be saved to file when desired. It was decided to record acceleration data after minimal processing, while not requiring additional information about the data sequences such as calibration values and device orientations.

7.4.3 DC Filter

The DC Filter module removes constant components from the sensors acceleration data. The X and Y-axis are subject to gravity, which does not reflect the machine operation. In order to ignore these constant g-forces, each acceleration datum, with the previous datums's original and scaled value, are pass through the following filter.

$$y(n) = x(n) - x(n-1) + Ry(n-1)$$

The value of R dictates the aggressiveness of the filter, and values between 0.9 to 1.0 are typically used. Smaller values accommodate drifting DC components, but since gravity does not change, a higher value was chosen. The Vibration Analysis Tool found that a relatively high value, $R = 0.98$, was able to completely remove gravity from vibrating screen's data after a few hundred samples [21]. This value of R is also used in the current implementation.

The more the DC filter is iterated, the more it can better filter out constant components. As a result, the first couple time windows of data are filtered, yet not used in the monitoring process. This allows the construction of the baseline profile and comparison techniques to commence after the filters has completely stabilized.

7.4.4 FFT

The FFT module converts time domain acceleration data into its frequency domain equivalent. The result is a discrete list of frequencies, and corresponding amplitudes depicting signal contribution or strength. The distance between discrete frequencies is determined by the sensor sampling rate and size of the input data sequence. It is commonly referred to as the *bin* size and in the current implementation it is:

$$\begin{aligned} \textit{bin} \text{ size} = f(n+1) - f(n) &= \text{(sampling rate) / (data length)} \\ &= 500.0 \text{ Hz} / 4096 \\ &= 0.12207 \text{ Hz} \end{aligned}$$

The usable frequency content is limited to half the sampling rate, known as the Nyquist frequency. Content at and above this limit are subject to aliasing, which can introduce frequency based errors into the monitoring process. For the current implementation, the Nyquist frequency is simply:

$$\begin{aligned} \text{Nyquist frequency} &= \text{(sampling rate) / 2.0} \\ &= 500.0 \text{ Hz} / 2.0 \\ &= 250.0 \text{ Hz} \end{aligned}$$

At the core of this module is the fast Fourier transform (FFT), provided by an open source library, Fastest Fourier Transform in the West (FFTW). Developed at MIT, this transform is popularized by its speed, portability and usability. At system initialization, the input data size is provided to the library where it then creates an optimized routine for the given length. The FFTW provides complex results, but since phase information is not required, the absolute magnitude of the complex values

are taken.

The computed frequency spectrum is provided to the Cross-Correlation and Monitored Variable Calculation modules for further processing. The frequency corresponding to the largest amplitude, the operating frequency, is provided to the Butterworth Filter module. The Butterworth filter filters around a specific frequency, and so an argument of the maximum function (*argmax*) within the FFT module provides it with this value.

7.4.5 Butterworth Filter

The Butterworth Filter module contains the implemented band-pass filter used to 'smooth' time domain acceleration data. It is primarily used to remove high-frequency content, so that subsequent processing can be carried out without the presence of noise. The filter is centered around the operating frequency, provided by the FFT module, and any frequency content outside of the bandwidth is attenuated.

The implemented Butterworth filter follows the design from the Vibration Analysis Tool, where noise was successfully removed from vibrating screens' acceleration data [21]. Characteristics of the filter are presented in Table 7.6.

Center Frequency	dynamic
Center Frequency Tolerance	±2.0 Hz
Bandwidth	±5.0 Hz
Order	4th
Coefficients	9

Table 7.6: Butterworth Filter Characteristics

The Butterworth filter utilizes computed coefficients in the filtering process, where coefficients are updated when the the center frequency changes. To regulate updates, new coefficients are only calculated when the the center frequency exceeds its tolerance.

Butterworth filters also require a settling time in order to produce meaningful results. Filter outputs start at zero and oscillate with increasing amplitudes until after a few hundred data points when it stabilizes. To ensure the filter has settled before monitoring calculations have begun, the first couple time windows of data are completely filtered, and then discarded.

7.4.6 Cross-Correlation

The Cross-Correlation module uses pairs of device frequency spectrums in a certain axis, as to produce a frequency spectrum depicting common frequency components. The result is a frequency spectrum like the FFT module, however information pertains to two sensor devices instead of just one. When performed for all device pairs, frequency relationships can be easily observed. Also, as noise is random it is highly unlikely to correlate between devices. This allows the produced spectrums to reveal frequency content experienced by multiple sensors, and ignore individual sensors' noise.

Cross-correlations can be performed in the time domain, where the result can then converted into the frequency domain for analysis. However, one paper had shown that performing the cross-correlation in the frequency domain can significantly reduce computational requirements. The proof and full explanation of the concept were presented in [21].

Since the provided frequency spectrum amplitudes are positive and real, the two sets of frequency content can be easily cross-correlated by multiplying corresponding amplitudes. Given two frequency spectrums' amplitudes, X_i and X_j, the cross-correlated amplitude for each frequency is simply calculated by:

$$y(n) = x_i(n)x_j(n)$$

7.4.7 Monitored Variable Calculation

The Monitored Variable Calculation module requires various data sets in order to compute the monitored variables. The required inputs are presented in Table 7.7. One set of filtered time domain g-forces and frequency spectrum can produce the monitored variables for an individual sensor device. One cross-correlated frequency spectrum can allow the calculation of the monitored variables pertaining to a cross-correlated device pair. The equations and algorithms used to compute the monitored variables are provided in Chapter 8.

Input Data	**Providing Module**
Filtered Time Domain G-Forces	Butterworth Filter
Frequency Spectrums	FFT
Cross-Correlated Frequency Spectrums	Cross-Correlation

Table 7.7: Monitored Variable Calculation Inputs

7.5 Condition Monitoring Technique

The Condition Monitoring Technique contains two modules: the Baseline Profile and Monitoring Algorithm module. Monitored variables computed from the Filtering and Processing module are used to populate the baseline profile, and then later used to compare against this profile. The Monitoring Algorithm conducts the comparison, and provides results to the Condition Handler module. Full details of the Condition Monitoring Technique are presented in Chapter 9.

At the start of a new monitoring session, the previously used baseline profile can be retrieved from the Storage Utilities module. Alternatively, a new baseline profile can be constructed. Both new and old baseline profiles are summarized through the Logger module. However, only new baseline profiles are saved in full detail using the Storage Utilities module.

7.6 Condition Handler

The Condition Handler contains the procedures for responding to the Monitoring Algorithm results. This is divided into two parts: identifying the machine status, and notifying users accordingly.

7.6.1 Condition Status

The Condition Status module identifies the condition of the vibrating screen given the operational status of each monitored variable. For these machines, there is currently no knowledge base matching specific operational signatures to particular machine faults. As a result, the machine is considered to have a fault if at least one of its monitored variables has a breached fault condition. Alternatively, a machine has healthy or normal operation only if all monitored variables have fault-free statuses.

The Condition Status module provides the machine status to the Recorder and Fault Notification modules. The Recorder uses the machine status to trigger fault recordings and determine the rate and duration of periodic recordings. As described in the following section, the Fault Notification module requires the machine status to trigger its user notification procedure.

7.6.2 Fault Notification

The Fault Notification module is responsible for contacting users in the event of a machine fault. Currently, two types of user notifications exist. The first is an on screen fault notification, with the corresponding triggered fault condition. Secondly,

if the software is provided with an email address and the system with Internet access, a fault message with attached log file is emailed to the specified user.

The automatically generated emails with attachment are sent using the Mutt, a small but powerful text-based email client. As it requires an email server, an open source option, Postfix, was chosen. It follows the Simple Mail Transfer Protocol (SMTP) to traverse the message through the Internet to its end destination. With the advent of mobile computing, namely smartphones, technicians and industrial personnel can access their emails remotely and immediately.

Currently, there exists future plans to interface the monitoring system with a programmable logic controller (PLC), and a remote file server with a database. The intention is to create standardized external connections, as to interface fault notifications with other systems. It is the Fault Notification module that would need to be expanded in order to implement both of these upgrades. The details of future work are discussed in Section 12.2.

7.7 Functional Modules

Functional modules are not and do not contain submodules, but rather function and interact with other modules independently. This section contains descriptions on these types of modules.

7.7.1 Parameter Interface

The Parameter Interface module downloads parameter files from a USB stick, and packages the parameters for module distribution. Users can create a parameter file from a graphical user interface in the PDA software package. Transferring this file to the system's monitoring computer specifies certain information required by the condition monitoring software. The parameter file can be updated as desired to better reflect the machine's details and system's operation.

Information contained in the parameter file is categorized in Tables 7.8, 7.9, 7.10, 7.11, 7.12 and 7.13. Specific module parameter requirements are also identified.

Parameter	Type	Required by Module
Customer Name	string	Fault Notification
Email Contact	string	Fault Notification
Machine Name	string	Fault Notification
Equipment Number	string	Fault Notification
Serial Number	string	Fault Notification
Machine Model	string	Fault Notification
Number of Bearings	integer	Fault Notification

Table 7.8: Parameters - Machine Information

Parameter	Type	Required by Module
Number of Devices	integer	Data Reader, Filtering and Processing, Recorder, Condition Monitoring Technique
Device Names	list of string	Recorder
Device Orientations	list of string	Orientation
Device Numbers	list of integer	Data Reader
Network Address	string	Data Reader

Table 7.9: Parameters - Sensor Device Information

Parameter	Type	Required by Module
Recording Duration	real	Recorder
Recording Interval	real	Recorder
Fault Recording Duration	real	Recorder
Fault Recording Interval	real	Recorder

Table 7.10: Parameters - Recording Information

Parameter	Type	Required by Module
Peak Amplitude	real	Monitored Variable Calculation
CC Peak Amplitude	real	Monitored Variable Calculation

Table 7.11: Parameters - Minimum Peak Amplitudes

Parameter	**Type**	**Required by Module**
Main G-Force	real	Monitoring Algorithm
Average G-Force X	real	Monitoring Algorithm
Average G-Force Y	real	Monitoring Algorithm
Average G-Force Z	real	Monitoring Algorithm
Operating Frequency X	real	Monitoring Algorithm
Operating Frequency Y	real	Monitoring Algorithm
Peak Frequency	real	Monitoring Algorithm
Peak Amplitude X	real	Monitoring Algorithm
Peak Amplitude Y	real	Monitoring Algorithm
Peak Amplitude Z	real	Monitoring Algorithm
CC Peak Amplitude X	real	Monitoring Algorithm
CC Peak Amplitude Y	real	Monitoring Algorithm
CC Peak Amplitude Z	real	Monitoring Algorithm

Table 7.12: Parameters - Tolerance Thresholds

Parameter	**Type**	**Required by Module**
Increment	real	Monitoring Algorithm
Start	real	Monitoring Algorithm
Normal	real	Monitoring Algorithm
Fault	real	Monitoring Algorithm
Maximum	real	Monitoring Algorithm

Table 7.13: Parameters - Algorithm Parameters

7.7.2 Data Reader

The Data Reader module is responsible for handling all communications with the sensor devices, as to receive the measured acceleration data. At startup, network sockets are created so incoming data is directed to the monitoring software. Given the parameter list of sensor devices and corresponding Internet Protocol addresses, the module then tests device network connectivity. If all devices are on the network, internally stored calibration values are individually requested, and then given to the Filtering and Processing module for later use. When the rest of the monitoring software is ready, devices are simultaneous sent an acquisition start message. This begins one of the main loops in the condition monitoring software.

The reader loop is contained in its own thread, as to ensure timely reception of the incoming data. When a device packet is received, an acknowledge message is returned. This message is receipt for the sensor device, and without it the device will resend the packet. The packet's structure is then examined and if correct, data is buffered to queues specific to a device and axis (X, Y, Z). These are the output queues where the Filtering and Processing module obtains data to be processed.

Within the Data Reader loop, malformed or missing packets trigger a transmission error routine. Durations between received packets are also monitored, as to handle devices that have not transmitted a packet after 2 seconds. Device silence for this duration is indicative of a dropped packet or network failure. All these events are documented through the Logger module. To compensate for missing data and to keep devices synchronized, the stop data acquisition is broadcast to all devices. Devices are reconnected, and only those that are properly networked are eligible to receive the data acquisition start broadcast. The Filtering and Processing module is notified of this procedure, so it can reinitialize itself for the restarted data streams.

The Data Reader module also communicates with the Calibration Procedure module by collecting data when requested, and sending calibration values to sensor devices. The start, stop and reconnect routines are utilized on a specific device as supplied by the Calibration Procedure module.

7.7.3 Recorder

The Recorder module governs the recording of calibrated and orientated acceleration data. It is given four parameters, the duration and frequency of data recordings for normal and faulty machine operation. Acceleration data is saved to file when due for a periodic recording or when the Condition Status module has identified a machine fault. The Storage Utilities module contains the specific procedures for saving

the data in its appropriate directory.

When a fault event has been identified, a recording is initiated capturing the waveform data responsible for triggering fault conditions. The fault recordings' length is set by the fault duration parameter. Subsequent data recordings would then proceed at the fault interval parameter. If normal machine operation is later identified, the original recording parameters will take effect once again.

7.7.4 Logger

The Logger module controls access to the log files, where system events are documented. Every time the system is started, a new log file is created so that monitoring details of each session can be examined. A semaphore is used to ensure messages are written in their entirety before a subsequent message is logged. Software modules and their logged messages are presented in Table 7.14.

Module	**Logged Messages**
Data Reader	active devices, device transmission errors
Recorder	periodic and fault recording times
Baseline Profile	baseline profile summary
Monitoring Algorithm	monitored variables changing fault status
Condition Status	return to normal machine operation
Calibration Procedure	calibration procedure details

Table 7.14: Modules using Logger

7.7.5 Storage Utilities

The Storage Utilities module contains the routines pertaining to long term data storage. This module interacts with the computer's file system by saving and loading data recordings, logs, baseline profiles and other required files. Directories categorize the saved file types, while subdirectories are used to designate the date and time of each new monitoring session, recording or calibration procedure.

Currently, system data can be exported in one of two ways: as an email attachment via the Fault Notification module, and though an USB stick upload. The upload procedure simply copies the saved directory structure onto an inserted USB stick. It is also possible to clear all saved data at the discretion of the user.

7.7.6 Calibration Procedure

The Calibration Procedure module controls the process of calculating and saving calibration values to a sensor device. It directs users to manipulate a device so that data readings can be taken at various orientations. The device's enclosure allows it to rest on a flat surface perpendicular to the ground, allowing each axis ($\pm X$, $\pm Y$, $\pm Z$) to be parallel with gravity. For the axes, data is collected and averaged corresponding to -1 g, 0 g and +1 g resting accelerations. Linear least squares is then performed, where linear mappings are found from raw acceleration data to g-force values. The calibrations values, m_k and b_k, are saved for each axis, where the mappings are expressed by the equation:

$$y = m_k x + b_k$$

The linear mapping is possible because the accelerometer's datasheet specified its output is linear in a defined range. Exceeding the maximum range, $\pm$ 10 g, does not guarantee output linearity.

Each sensor device requires at least one successful calibration procedure to be functional in the system. The calibration values are crucial to processing the acceleration data. It is possible to recalibrate a device if the accelerometer orientation has shifted or other sensor characteristics have changed over time. A poorly performed calibration procedure can result in reduced accuracy of the corresponding sensor device.

8 Monitored Variables

8.1 Overview

This chapter discusses the monitored variables used in the condition monitoring process. These variables are calculated from the filtered waveform signals, along with the individual and cross-correlated frequency spectrums discussed in Section 7.4.

8.2 Average G-Force

The average g-force is a monitored variable that reveals the acceleration of a machine. Its unit is in g-forces, where 1 g equals 9.81 m/s^2. It is computed for every sensor and for all three axes. Given a time window of filtered acceleration data (X, Y or Z), it is calculated as follows:

$$\texttt{m\_averageGForce}_{\texttt{x}} = (|\max(X)| + |\min(X)|)/2$$

8.3 Main G-Force

Another form of average g-force is known as the main g-force. It uses filtered acceleration data from the two main drive axes, X and Y, to determine the largest g-force experienced in the corresponding plane. It is calculated as follows:

$$\texttt{m\_mainGForce} = \max(\sqrt{x_i^2 + y_i^2}, \forall x \in X, y \in Y)$$

8.4 Operating Frequency

The operating frequency describes the speed of a periodic system in cycles per second, commonly known as Hertz (Hz). This variable is only calculated for the X and Y axes, as they are the only axes being driven in the vibrating machines. Since the FFT only returns a discrete set frequencies (F) and their amplitudes (A), polynomial interpolation is used to increase accuracy.

Given a FFT result of a time window, the operating frequency is calculated as follows:

1. Find the largest bin amplitude (a_i) and corresponding center frequency (f_i)
2. Perform polynomial interpolation using the following points, (f_{i-1}, a_{i-1}), $(f_i, a_i), (f_{i+1}, a_{i+1})$, to determine the frequency corresponding to the largest amplitude in the polynomial, which is $\texttt{m\_operatingFrequency}_{\texttt{k}}$

While the interpolation technique is discussed in detail in [21], the implemented calculation is expressed as follows:

$$\alpha = f_{i+1}a_{i-1} - f_{i-1}a_{i+1} - f_i a_{i-1} + f_i a_{i+1} - f_{i+1}a_i + f_{i-1}a_i$$
$$\beta = f_{i-1}^2 a_{i+1} - f_{i-1}^2 a_i - f_i^2 a_{i+1} + f_i^2 a_{i-1} - f_{i+1}^2 a_{i-1} + f_{i+1}^2 a_i$$
$$\texttt{m\_operatingFrequency}_\texttt{k} = -\beta \ / \ 2\alpha$$

Industrial end users tend to request the operating frequency in revolutions per minute, known as RPM. Since Hertz is defined as revolutions per second, the conversion is simply:

$$\text{RPM (rev/min)} = \texttt{m\_operatingFrequency}_\texttt{k} \text{ (rev/sec)} * 60 \text{ (sec/min)}$$

8.5 Peak Frequencies

8.5.1 Peak Detection

Peak frequencies are frequencies with amplitudes that produce a local maxima in the frequency spectrum. This process of selecting frequencies based on their amplitude is known as peak detection. In the monitoring system, frequencies and their amplitudes are selected from the discrete set of FFT results for a particular axis and time window. The FFT output either reflects the frequency content of a particular sensor's acceleration waveform, or is the cross-correlation of two sensors' FFTs.

The implemented detection algorithm requires a minimum peak amplitude, for which all frequencies with amplitudes less than this value are deemed noise and thus ignored. This threshold would need to be adjusted depending on the amount of noise in the monitored system.

The algorithm runs over each frequency bin, and a peak is detected if it passes both of the following criteria:

1. The amplitude is greater than the minimum peak amplitude
2. The amplitude is greater than the four neighbouring frequencies' amplitudes, two from each side (greater than or equal to the subsequent bin)

This simple algorithm has been successful in identifying peaks from frequency spectrums. However, a side effect of the second criteria is that two peaks may not be within three bin widths of one another. Figure 8.1 presents two successfully identified peaks. This demonstrates an example of the closest two peaks can be to one another, while both being uniquely identified.

Figure 8.2 depicts a similar example with two local maxima. The peaks are within three bin widths of one another, and as a result only the largest one will be identified. It is undesirable to have all neighbouring peaks identified, as secondary peaks within a few bins widths of the primary peak are attributed to FFT leakage or sidelobes from the primary peak [21].

In the Vibration Analysis Tool developed at McMaster University, a slope based peak detection was implemented, but the concept of ignoring peaks too close together was utilized [21]. The corresponding paper empirically testing the tool, and determined a peak criteria stating that peaks would only be identified if they are larger than the next four neighbouring amplitudes. The particular tool was used in noisy systems, but tuning all the system parameters resulted in the elimination of false positive peaks. The peak detection algorithm used in the monitoring system could also be adjusted to accommodate noisier systems, by increasing the frequency band between allowable peaks.

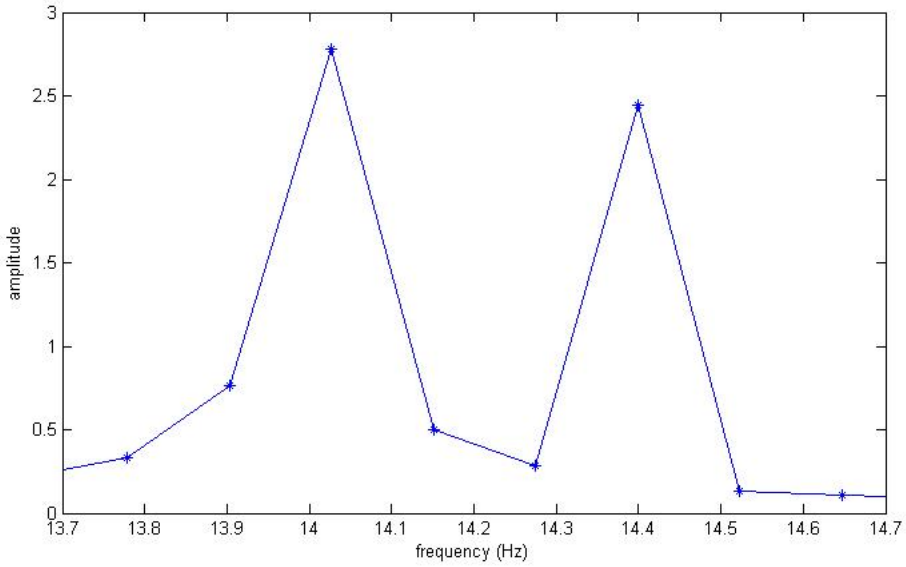

Figure 8.1: Detecting two peaks, at 14.0 Hz and 14.4 Hz

8.5.2 Peak Assignment

Once a peak had been identified, its frequency and amplitude are then computed. Like the operating frequency computation in Section 8.4, polynomial interpolation is

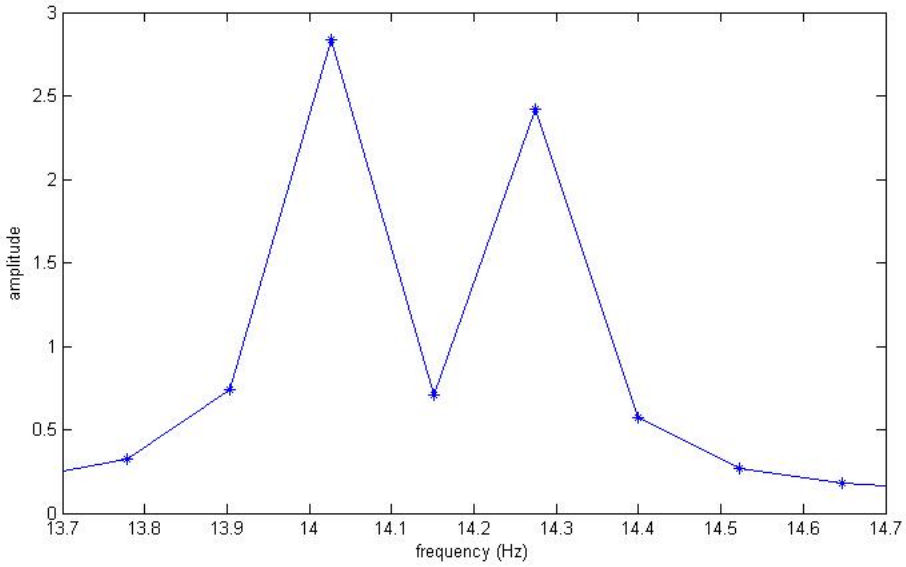

Figure 8.2: Detecting one peak, at 14.0 Hz

used to provide more accuracy on the frequency of interest contained in the respective FFT bin. Interpolation is performed using the detected peak bin values, and the two neighbouring points. The is resulting peak frequency is assigned to `m_frequency`.

As discussed, a side effect of the FFT is that expressed amplitudes are reduced the further away they are from their center bin frequency. An expression was derived in Chapter 3 to determine the amount of amplitude reduction based on this frequency distance. Having access to the peak and bin's center frequency, their difference is input into the inverse of the found expression. This function amplifies amplitudes so that the effect of FFT leakage is negated. Figure 8.3 presents an example of the scaling function for an entire bin width.

A peak's amplitude, `m_amplitude`, is the product of the its bin's amplitude and the appropriate scaling factor. Scaling allows peaks to be better represented, regardless of their location with respect to their center frequency bins. This allows peaks with a drifting frequency to have consistent amplitudes as they moves across the frequency spectrum. Ultimately, it improves the monitorability of peaks by reducing deviations caused by error.

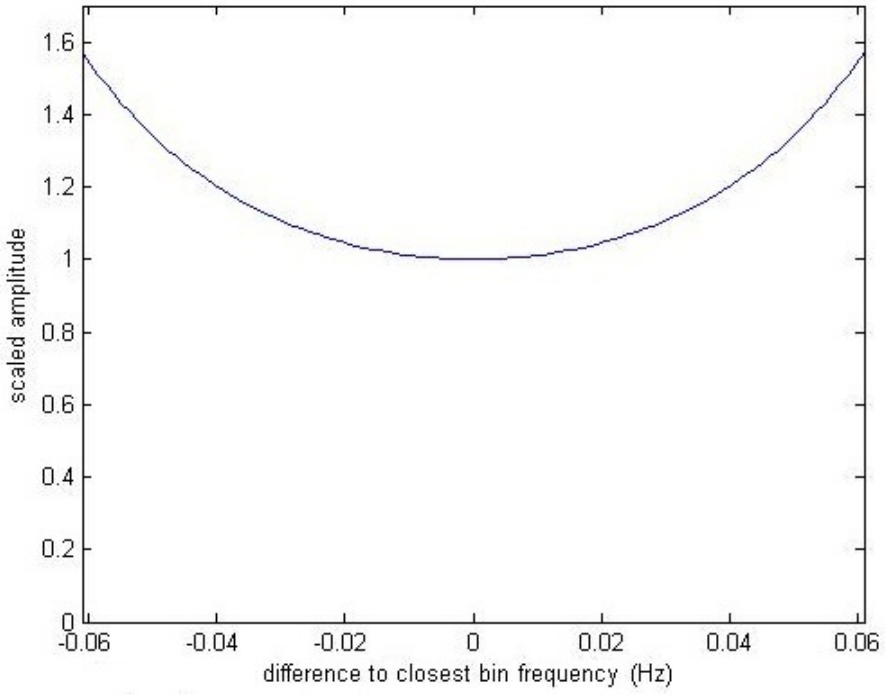

Figure 8.3: Peak Amplitude Scaling Function

9 Condition Monitoring Technique

9.1 Overview

This chapter discusses the implemented condition monitoring process. The technique observes monitored variables calculated from waveform data, and compares them to their baseline equivalent. Algorithm parameters are tuned to refine the monitoring process. This chapter contains the procedure and decision making techniques that attempt to determine the condition of the monitored variables.

9.2 Baseline Profile

Baselines are associated with normal or desired machine operation. In condition monitoring, signals observed to deviated from the machine's baseline typically suggest abnormal functionality. If a baseline is acquired from a machine with an existing fault, that fault would be assumed to be apart of the machine's normal operation. This would allow the fault to go undetected, until the fault's signature differs from how it was observed in the baseline. Consequently, it is assumed that baselines will be taken from machines without faults, or at least with faults in early stages that are too difficult to detect using the current available technology. Technicians have access to the Vibration Analysis Tool [21], enabling them to analyze the current status of the machine and ensure no detectable faults are present.

For the monitoring system, the baseline profile is comprised of the monitored variables as discussed in Chapter 8. This allows for direct comparison between the current variables of the machine and its baseline, one variable at a time. To ensure the baseline profile properly represents the operation of the machine, the machine is monitored for multiple time windows. This allows averaging techniques to be utilized, ultimately increasing the accuracy and precision of the baseline profile.

The baseline profile for a sensor's average g-forces, $\texttt{m\_averageGForce}_\texttt{k}$ and $\texttt{m\_mainGForce}$, are determined by taking the mean of the respective monitored variables for each time window of the baseline. Since the g-forces variables already represent averages, noise does not have a large impact on their values. Averaging across the time windows further reduces the effect of system noise in their profile.

The baseline operating frequency, $\texttt{m\_operatingFrequency}_\texttt{k}$, and peak frequencies, $\texttt{m\_peakList}_\texttt{k}$, are not averaged directly like the g-forces. Instead, the FFT results from the individual sensors as well as the cross-correlated sensor pairs are averaged across their respected time windows. Operating frequencies are then calculated from the spectrums, using the documented methods. The frequency spectrums are then

passed through the peak detection algorithm to identify peak frequencies. Since noise is assumed to be Gaussian it affects each time window differently, so averaging the FFTs from multiple windows attenuates this noise. This procedure provides a better representation of not only the amplitudes dominated by noise, but also the amplitudes of meaningful frequency content.

9.3 Monitoring Algorithm

The implemented condition monitoring algorithm is generic in nature, able to be used in a variety of situations. It is a numeric point based system, where points are awarded for incidents, and withdrawn when tolerances are not exceeded. An incident event occurs when a monitored variable exceeds its assigned threshold tolerance in a time window. The points were documented as $\texttt{i\_incidentPoint}_{\texttt{m}}$, and known as incident points (IP). The monitoring algorithm can possibly incremented or decremented the IPs every time window. Each monitored variable has its own IP variable associated with it for every sensor device and axis.

The algorithm has parameters, which are tuned to achieve the desired monitoring response. These parameters dictate the usage of the monitoring system. Other values used in the algorithm are predefined constants, and are references when determining the parameters. Table 9.1 presents the constants and their values, while Table 9.2 presents the parameters and their corresponding restrictions.

Name	Symbol	Value
`Decrement`	P_{dec}	-1
`Minimum`	P_{min}	0

Table 9.1: Algorithm Constants

Name	Symbol	Restriction
`Increment`	P_{inc}	$P_{\text{inc}} > 0$
`Start`	P_{start}	$P_{\text{min}} < P_{\text{start}} \leq P_{\text{max}}$
`Normal`	P_{norm}	$P_{\text{min}} \leq P_{\text{norm}} < P_{\text{fault}}$
`Fault`	P_{fault}	$P_{\text{norm}} < P_{\text{fault}} \leq P_{\text{max}}$
`Maximum`	P_{max}	$P_{\text{max}} \geq P_{\text{fault}}$

Table 9.2: Algorithm Parameters

The constants and parameters are reference points and triggers for the IP values corresponding to the monitored variables. Their application is depicted in the algorithm's framework, as presented in Figure 9.1. As previously mentioned, the monitoring process begins after the baseline profile has been determined. Monitored variables' IP values are then initialized to `Minimum`, which is also the lower bound for IP values. The fault flag, $\texttt{i\_faultFlag}_{\texttt{m}}$, is also initialized to false, indicating that the machine has healthy or normal operation. When a monitored variable is calculated for a new time window, it is then compared to the baseline profile as outlined in Section 9.4. The control flow then follows one of the two main branches, determined by whether the tolerance threshold was exceeded or not.

If the tolerance was exceeded, and the corresponding IP value was at the `Minimum` value, the IP would be assigned the `Start` value. Otherwise, the IP value would be incremented by the `Increment` value. It is possible to assign the same value to both the `Start` and `Increment` value, which would result in the equivalent IP value when originating from the `Minimum`. However, if the user wants to take full advantage of the continuous monitoring, the `Start` value would be given a value equal to or greater than the `Fault` value. Doing so would allow a single deviation from the baseline to immediately trigger a fault condition (IP $\geq P_{\text{fault}}$), given that the corresponding fault condition is not already in process. This results in the documentation of any abnormal machine event.

Due to the presence of noise in a monitored system, it may not be desirable to allow a single incident to activate a fault condition. In this situation, `Start` would be assigned a value less than `Fault`. In order to trigger a fault, a monitored variables's tolerance would need to be breached over multiple time windows. The minimum number of time windows required to indicate a fault ($MTW_{\text{min-fault}}$) can be calculated as follows:

$$MTW_{\text{min-fault}} = \lceil (P_{\text{fault}} - P_{\text{start}})/P_{\text{inc}} \rceil + 1$$

This number is designated as a minimum because it is not a requirement that the tolerances be exceeded in consecutive time windows. It is possible that a monitored variable causing incidents would not trigger its tolerance for a few time windows within its progression to activating the fault condition. In such time windows, the control flow would follow the second of the two main branches.

The second main branch in the monitoring algorithm manages the incident points in time windows where a monitored variable does not deviate from the baseline. The simplest case for the branch is when a monitored variable's IP value is at the `Minimum`, and no action needs to be taken until the next time window. Alternatively, an IP

with a value greater than the `Minimum` will be altered by `Decrement`. Note that values cannot be decremented below `Minimum`, as enforced by the algorithm.

Unlike the `Increment` value, the `Decrement` has a constant value of -1. The significance of `Decrement`'s magnitude is equivalent to the duration of one time window. For any given monitored variable, the summation of the acquired IP represents the minimum number of time windows that variable must remain within its tolerance in order to return to `Minimum`. This is also considered a minimum number because consecutive equivalent actions are not required. More importantly, if a monitored variable has an activate fault condition, the minimum number of non-deviating time windows required to signify a return normal operation is simply:

$$MTW_{\text{fault-norm}} = \lceil P_{\text{fault}} - P_{\text{norm}} \rceil$$

This presents the purpose of the `Normal` value, a trigger threshold to set the fault flag to false when the current IP is less than or equal to its value. Note that the corresponding monitored variable must also have an active fault condition, indicated by the fault flag being true.

It is possible that abnormal machine operation is temporary, and so the monitoring algorithm will identify when fault symptoms are alleviated beyond detection. After a fault is detected, the capacity to return to normal operation is expressed through the difference between the current IP value and `Normal`, denoted by $MTW_{\text{IP-norm}}$. The upper bound to this minimum number of non-deviating time windows is restricted by the `Maximum` value. A monitored variable whose IP value exceeds the fault trigger can still be incremented up until `Maximum` if the fault continues to persist. As a result, the upper bound to $MTW_{\text{IP-norm}}$ can be determined by:

$$\max(MTW_{\text{IP-norm}}) = \lceil P_{\max} - P_{\text{norm}} \rceil$$

The collection of minimum time windows ($MTWs$) provides additional numerical indicators on how the algorithm is being utilized. This can aid users and technicians by emphasizing the relationships between the chosen parameters.

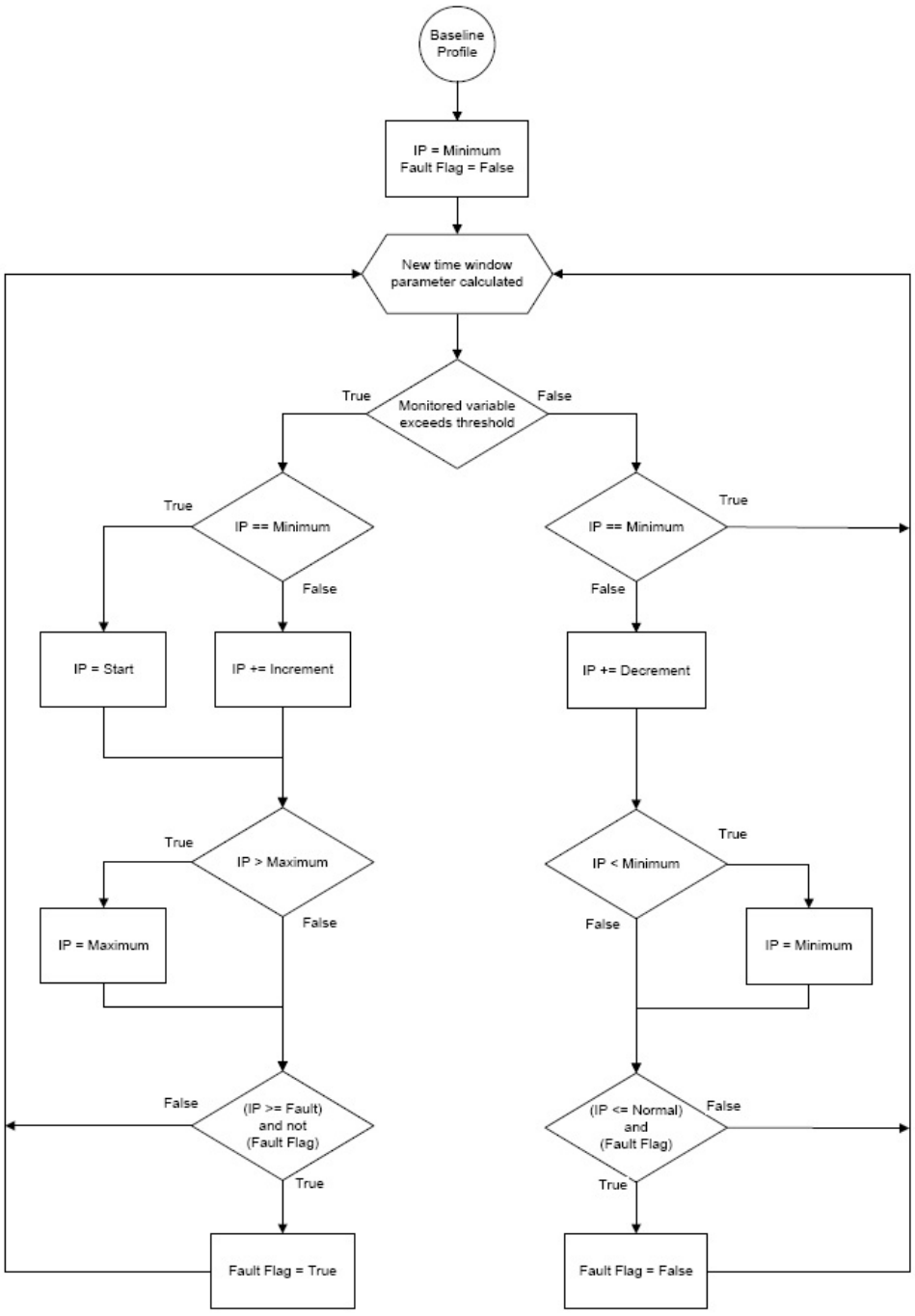

Figure 9.1: Condition Monitoring Algorithm

9.4 Tolerance Thresholds

Following the calculation of a baseline profile, the monitored variables of each subsequent time window are compared to their respective baseline value. This step simply determines if a variables's current time window value is within an acceptable deviation from the baseline profile. For g-forces and frequency amplitudes, a relative difference is used, whereas for frequencies an absolute difference is utilized.

Users and technicians must select deviations for each type of monitored variable and axis in `m_machineProfile`. These deviation allowances are known as trigger or threshold tolerances. When a monitored variable's value exceeds its tolerance in a time window, the event is referred to as an incident and does not necessarily imply abnormal machine operation. Rather, this is determined by the implementation of the monitoring algorithm, as outlined in the previous section.

It should be noted that the current implementation attempts to minimize the number of parameters as to simplify usability for end users. The monitoring system's algorithm could be revamped so that frequency peaks with different amplitudes are assigned different tolerance thresholds. Absolute tolerance thresholds could also be adopted for peak amplitudes, but would also require additional tuning for the various amplitudes.

9.5 Baseline Profile Stability

As mentioned, it may be desirable to select parameters such that a single violation of a tolerance threshold (an incident), would not trigger a fault condition. The threshold would instead have to be exceeded for at least the minimum number of time windows to indicate a fault, $MTW_{\text{min-fault}}$. The benefit of the described usage is in the reduction of potential false positive faults caused by noise. As noise is random, it is unlikely that it would cause persistent deviation of a specific monitored variable from its baseline profile across multiple time windows.

In noisy environments it is expected that false positive incidents would occur when the system is given a set of tolerance thresholds with a close bound to the baseline profile. Desired system performance can be achieved provided noise based incidents do not exceed the weighted ratio of the `Decrement` to `Increment`, as reflected by:

$$|P_{\text{dec}}| : P_{\text{inc}}$$

With a 1:1 ratio, time windows that do not deviate from the baseline would counteract incidents, using an equivalent weighting. Despite an evenly distributed 50 % in-

cident rate, a monitored variable's incident point value would remain near the `Minimum` value. For fault detection, consistance readings across time windows would be heavily favoured.

Using a different ratio where P_{inc} is greater than $|P_{\text{dec}}|$ would function like previously discussed yet allow for a larger impact from incidents. In situations where most noise deviations are contained within the tolerance threshold, a large `Increment` value would acknowledge the significance of a subsequent incident. Specifically, it would require `Increment` time windows of normal operation to completely counterbalance one incident.

It should be mentioned that `Start` can be assigned a different value than `Increment` so that the first of a potential train of incidents can have its own weighting. It is up to technicians to decide if this weighting should smaller, larger, or simply equivalent to any other incident.

In general, if noise is high enough to trigger a fault condition, the monitoring system would require tuning for successful operation. End user action from at least one of the following would need to be taken:

1. Increase the tolerance threshold of the corresponding monitored variable, as to reduce the frequency of noise triggering the threshold
2. Decrease `Start` so that isolated incidents are not as heavily weighted
3. Decrease `Increment` so that more consistency is required for fault detection
4. Increase `Fault` to reduce the likeliness of the fault condition being breached by clusters of incidents, by allowing more opportunity for non-deviating time windows to counteract the noise-based incidents
5. Increase the minimum detectable peak amplitude so that the baseline profile would disregard smaller peaks encompassed by noise

10 Monitoring Simulation

10.1 Overview

This chapter presents a simulation of a vibrating machine that is monitored by the condition monitoring system. A stable baseline profile will be first be determined. Faults will then be introduced into the machine to determine the effectiveness of the software system. The system will operate at 500 Hz sampling and use 4096 points in the FFT calculations. Only the X-axis will be discussed, as analogous methodologies are utilized for the Y and Z axes.

10.2 Simulated Signals

The simulated signals of a healthy vibrating machine are constructed using sinusoidal waves of various frequencies and amplitudes. Like the real vibrating machines, a dominant frequency is present representing the operating frequency. Noise is also included into the signals to reflect potential inaccuracies of the sensors as well the material flow over the machine. Gaussian random noise with a mean of zero and standard deviation of one, $\mathcal{N}(0, 1)$, was chosen for this role. The experiment is conducted using four virtual sensors, where the amplitudes of the sinusoidals in each signal are presented in Table 10.1.

	Frequency / Amplitude			
Sensor Location	14.0 Hz	23.0 Hz	35.0 Hz	42.0 Hz
RFB	3.00	1.00	0.50	0.25
RDB	3.20	0.80	0.56	0.23
LFB	2.80	1.10	0.53	0.24
LDB	2.90	0.70	0.54	0.20

Table 10.1: Simulated Healthy Machine Signals

A quarter time window of the generated waveform for the RFB sensor is illustrated in Figure 10.1. The corresponding frequency spectrum of the time window up to 100 Hz follows in Figure 10.2.

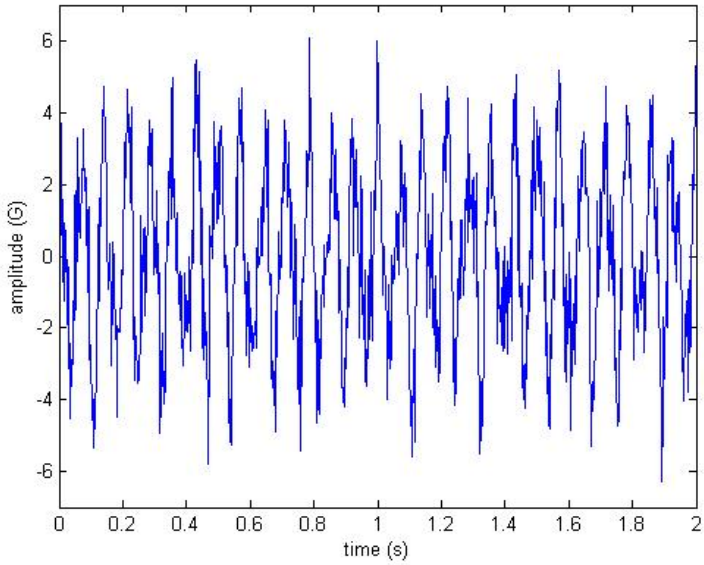

Figure 10.1: Simulated RFB Vibration Waveform

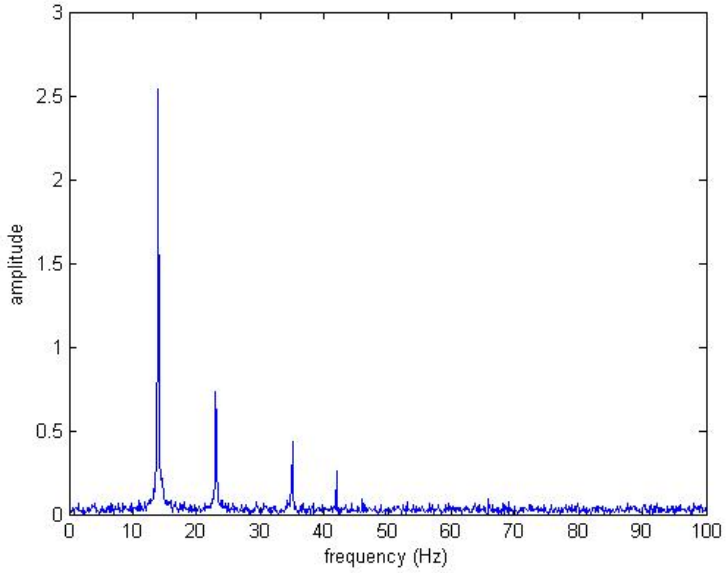

Figure 10.2: RFB Frequency Spectrum, from the simulated baseline

10.3 Baseline Profile

The first assessment on the monitoring system was ensuring a successful construction of the baseline profile for the simulated machine. Half an hour of generated waveform data (900,000 points) was passed into the monitoring system. The first minute of data is used to establish the baseline profile, while the remaining data would be monitored to confirm its stability. The minimum peak amplitudes, tolerance thresholds and algorithm parameters chosen to identify a stable profile are presented in Table 10.2. Empirical testing was used to refine these values. The selected parameters were not exhaustively optimized for the given monitored machine, as to better reflect the practical use of the system by end users.

Minimum Peak Amplitudes	
Individual Peaks	0.10
Cross-Correlated Peaks	0.006

Tolerance Thresholds	
Average G-Force	5.0%
Operating Frequency	0.050 Hz
Peak Frequency	0.061 Hz
Peak Amplitude	10.0%
Cross-Correlated Peak Amplitude	30.0%

Algorithm Parameters	
`Increment`	1
`Start`	1
`Normal`	0
`Fault`	8
`Maximum`	8

Table 10.2: Selected System Parameters

The numeric values of the parameters reveals the fairly conservative use of the monitoring system as detailed in Section 9.3. A single violation of a tolerance threshold in a time window, known as an incident, would not trigger the fault condition. Rather, it would have to be exceeded for at least the minimum number of time windows to indicate a fault, $MTW_{\text{min-fault}}$. For the simulation, this duration is eight time windows, corresponding to approximately a minute. The implementation is targeted towards the monitoring of sustained abnormalities and faults.

The individual sensors' baseline profiles of the simulated machine are presented in Table 10.3. The frequencies of interest were all successfully identified, without the inclusion of any extraneous frequency content. All sensors also independently and correctly identified 14.03 Hz as the operating frequency of the machine. The baseline profile is also comprised of peak frequencies detected from the cross-correlated frequency spectrums, as provided in Table 10.4.

	Peak Frequency / Amplitude				
Sensor	14.03 Hz	22.97 Hz	35.03 Hz	41.99 Hz	**Average G-Force**
RFB	2.58	0.81	0.46	0.26	3.61 g
RDB	2.74	0.64	0.50	0.23	3.76 g
LFB	2.42	0.88	0.49	0.25	3.42 g
LDB	2.50	0.55	0.49	0.21	3.46 g

Table 10.3: Simulated Machine's Baseline Profile

	Peak Frequency / Amplitude			
Cross-Correlated Sensors	14.03 Hz	22.97 Hz	35.03 Hz	41.99 Hz
RFB - RDB	6.93	0.47	0.22	0.059
RFB - LFB	6.12	0.65	0.22	0.064
RFB - LDB	6.33	0.41	0.22	0.053
RDB - LDB	6.71	0.32	0.24	0.047
LFB - RDB	6.48	0.51	0.24	0.056
LFB - LDB	5.92	0.44	0.24	0.051

Table 10.4: Simulated Machine's Baseline Profile - Cross-Correlated Peaks

After establishing the baseline profile, no faults conditions were triggered while monitoring the remainder of the normal machine operation. However, it was seen from the incident point values in the monitored variables that incidents did occur. The most susceptible variables to noise based incidents were the peak frequencies with low amplitudes, specifically the 42.0 Hz peaks. With a 10 % peak amplitude tolerance threshold, small amplitudes are not given as much of an absolute deviation from the baseline profile as compared to larger peak amplitudes. If this did cause baseline profile instability, Section 9.5 presents different approaches to supressing the system noise.

10.4 Emerging Frequency Content

The first set of simulated faults introduce additional frequency content into the machine's normal operation. As discussed in Section 2.2, emerging peak frequencies are evidence of various machine failures in gears, bearings, shafts, springs or other components that contribute vibrations.

10.4.1 Visible Content

In this experiment a sinusoidal wave of 28.0 Hz is added to each sensor's signal, with the corresponding amplitudes shown in Table 10.5. The fault induced frequency spectrum of the RFB signal is presented in Figure 10.3, where the new peak frequency is visible.

	Frequency / Amplitude
Sensor Location	27.97 Hz
RFB	0.60
RDB	0.80
LFB	1.00
LDB	0.40

Table 10.5: Simulated Fault - Additional Sinusoidal

In the machine's signals, the fault persisted for 20 time windows. All four sensors and six cross-correlations were able to identify an emerging peak frequency at 27.97 Hz. Table 10.6 contains the reported amplitudes, which indicate the fault had the greatest effect near the LFB sensor location. If this was an unknown fault on a real machine, it would be advised technicians investigate the LFB area of the machine for a damaged component.

	Frequency / Amplitude
Sensor Location	28.0 Hz
RFB	0.48
RDB	0.67
LFB	0.82
LDB	0.33

Table 10.6: Simulation Results - Additional 28.0 Hz Sinusoidal

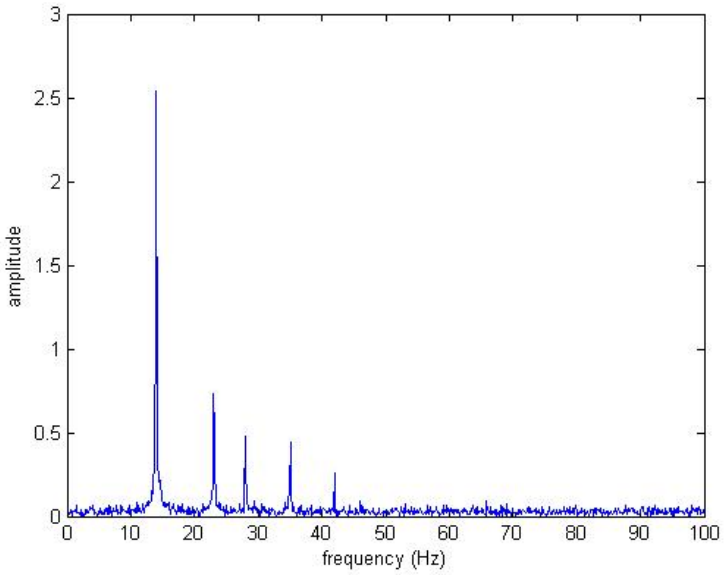

Figure 10.3: RFB Frequency Spectrum, containing simulated 28.0 Hz fault

After successful identification of the fault, its signature was removed from the machine's signal to reflect a temporary abnormality or machine maintenance. The experiment was continued to ensure the monitoring system was able to detect the absence of the fault. Shortly after eight time windows, as determined by $P_{\text{fault-norm}}$, the monitoring system correctly identified that the simulated machine was running under normal operation once again.

10.4.2 Buried Content

Similar to the previous experiment, frequency content was introduced into the machine's healthy signals. A 57.0 Hz sinusoidal wave was used, with amplitudes presented in Table 10.7. The entire monitored frequency spectrum of the sensors containing the largest fault amplitudes, RDB and LDB, are presented in Figures 10.4 and 10.5. In this situation, it is unlikely technicians would be aware of the emerging peaks through visual inspection of the frequency spectrum. Each sensors' amplitude corresponding to 57.0 Hz are visually concealed by the noise of the system.

The 57.0 Hz sinusoidal fault signals were introduced for 50 time windows. No individual sensor was able to detect the emerging frequency content for the duration of the experiment. However, peak detection from the cross-correlated frequency

	Frequency / Amplitude
Sensor Location	57.0 Hz
RFB	0.068
RDB	0.090
LFB	0.045
LDB	0.072

Table 10.7: Simulated Fault - Additional 57.0 Hz Sinusoidal

spectrums were able to identify the peak frequency of interest in two instances, as presented below in Table 10.8. The common element between the two cross-correlated frequency spectrums is the RDB sensor, indicating the fault likely originates in that region of the machine. When the fault signature was removed from the machine's signal, the monitoring system successfully identified normal machine operation.

	Frequency / Amplitude
Sensor Location	57.0 Hz
RFB - RDB	0.009
RDB - LDB	0.008

Table 10.8: Simulation Results - Additional 57.0 Hz Sinusoidal

Faults identified solely through these means do not appear critical, as their contributed vibration amplitudes is minimal. However, they are indicators of future machine health and an advisory requesting maintenance. Once fault signatures are recognized, an expert system could give end users detailed recommendations regarding the severity of the particular fault.

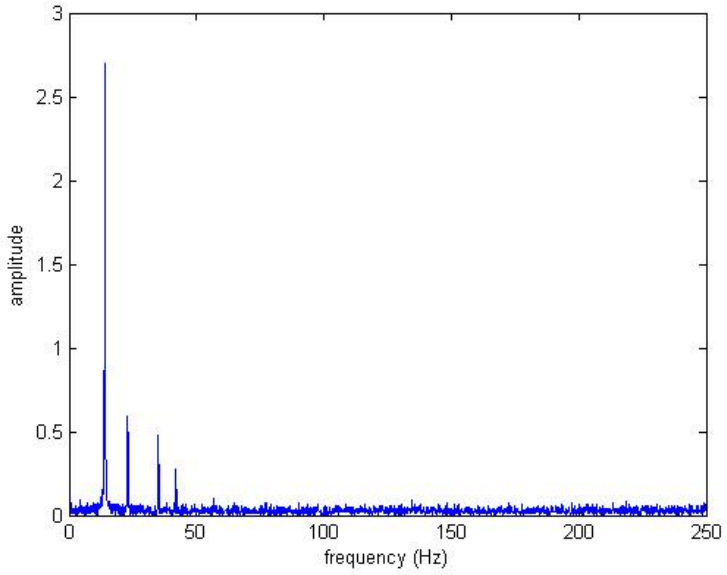

Figure 10.4: RDB Frequency Spectrum, containing simulated 57.0 Hz fault

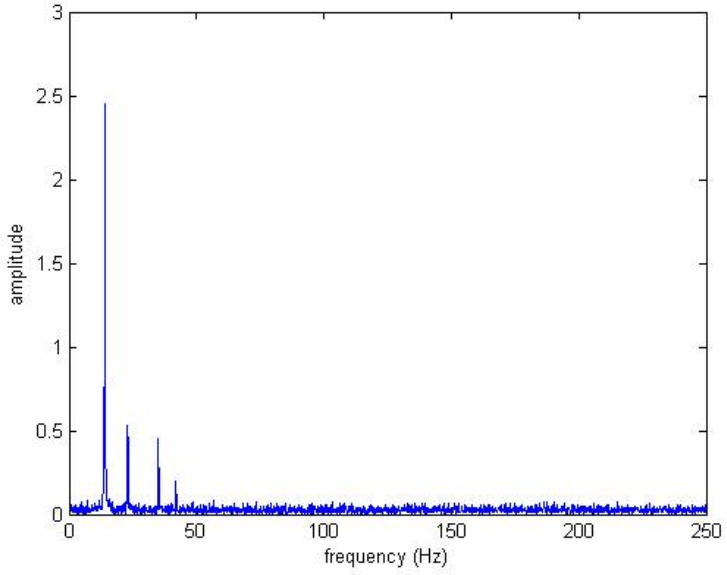

Figure 10.5: LDB Frequency Spectrum, with simulated 57.0 Hz fault

10.5 Lost Frequency Content

The simulated fault in this experiment is the withdrawal of the 35.0 Hz peak frequency from the RFB's normal signal. This signifies the failure of an auxiliary component, where the machine continues to operate with the loss of some functionality. Due to the rigid body of the vibrating machines, the 35.0 Hz peak was not completely removed, but rather only 25 % of the assigned amplitude remained. The residual vibration caused from similar functioning components on other parts of the machine would induce some amplitude of this frequency content.

The abnormal frequency spectrum is presented in Figure 10.6. The peak frequency was reduced for 10 time windows, and then reintroduced at its corresponding baseline amplitude. The fault condition was triggered when the system identified a 76 % amplitude reduction of the 35.0 Hz peak at the RFB location. Such a reduction would be cause for a machine inspection at the indicated location. When the 35.0 Hz peak returned to its proper amplitude, the system declared normal operation.

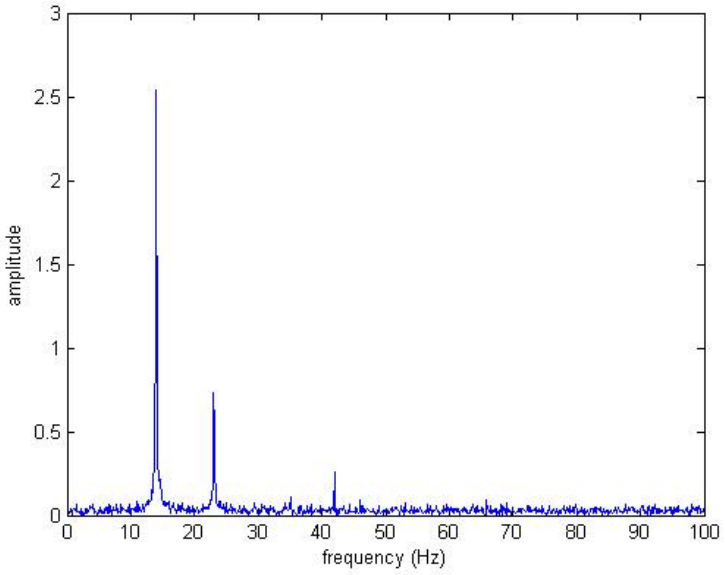

Figure 10.6: RFB Frequency Spectrum, with a reduced 35.0 Hz peak

10.6 Changing G-Forces

This experiment simulates another form of undesired operation, machine unbalance. When vibrating screens are properly weighted, acting g-forces are consistent throughout the machine. This allows the entire surface of the screen to be equally effective. If properly balancing is not achieved, different portions of the screen experience different g-forces. The result of the unexpected g-forces is irregular material screening and potential damage to the machine.

To simulate this fault scenario, the RFB location has an increased operational amplitude while LDB has a reduced amplitude. Table 10.9 presents the absolute and relative amplitude changes from 14.0 Hz sinusoidals used in the production of the baseline signals. Frequency spectrums from the fault induced sensors are given in Figure 10.7 and 10.8.

	Difference from Baseline Profile	
Sensor Location	Absolute (g)	Relative (%)
RFB	0.25	8.33
LDB	-0.25	-8.62
RDB	0	0
LFB	0	0

Table 10.9: Simulated Unbalance - Modified Operational Amplitude

The fault condition was triggered when monitoring the 20 time windows of abnormal operation. The average g-force tolerance threshold was exceeded by both the RFB and LDB sensors. Table 10.10 presents the observed fault signature. No specific frequency was identified, but it is known that large deviations amongst sensors' g-forces are associated with unbalance. Adding weight to specific parts of the machine can allow for a more uniform operation.

	Exceeded Average G-Forces	
Sensor Location	Amplitude (g)	Relative Difference (%)
RFB	3.88	7.51
LDB	3.24	-6.22

Table 10.10: Simulation Results - an Unbalanced Machine

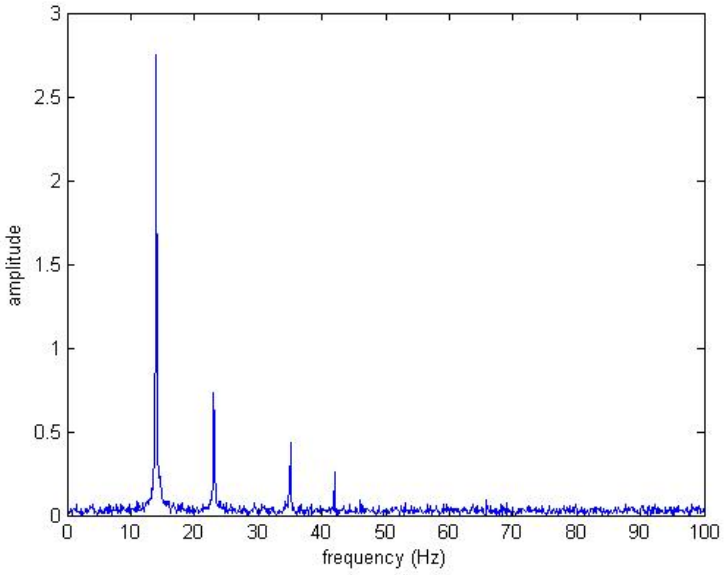

Figure 10.7: RFB Frequency Spectrum, with increased 14.0 Hz amplitude

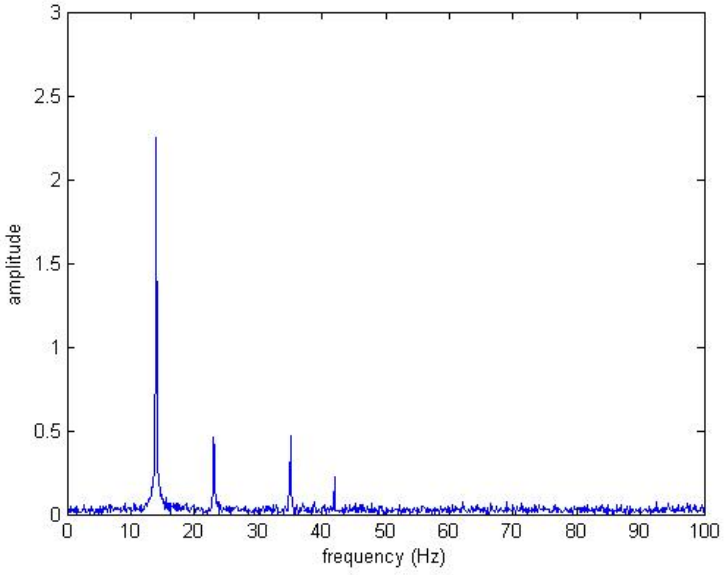

Figure 10.8: LDB Frequency Spectrum, with decreased 14.0 Hz amplitude

It should be noted that the operating frequency's peak amplitude did not trigger a fault condition. The 14.0 Hz peak seen in Figure 10.7 is visibly larger than in the baseline, as seen in Figure 10.2. However, the difference remained consistently less than the selected 10 % tolerance threshold and so no corresponding fault condition was triggered. If it was desired to have this variable trigger a fault at the given amplitude, one would have to decrease the peak frequency amplitude tolerance threshold. This would allow smaller deviations to be considered incidents for all the corresponding axial peak amplitudes.

10.7 Drifting Operating Frequency

The simulated fault in this experiment is a drift in the operating frequency. A 0.095 Hz increase is induced into the vibrating machine for 25 time windows. This abnormality is large enough to violate the selected tolerance threshold put on the operating frequency.

When performing the experiment, fault conditions were successfully triggered when the monitoring system identified a consistent change in the operating frequency for all sensors. Table 10.11 presents the unanimous input and detected operating frequency for the baseline profile and current experimental fault signal.

	Operating Frequency (Hz)	
Simulation	Input	Detected
Baseline Profile	14.0	14.03
Fault Experiment	14.095	14.08

Table 10.11: Simulation Results - Drifting Operating Frequency

The RFB frequency spectrum with drifted operating frequency is presented in Figure 10.9. When visually compared to the baseline's spectrum, Figure 10.2, a large reduction in the operating frequency's peak amplitude is seen. This is further evidence of the FFT leakage discussed in Chapter 3, as the input amplitude was not reduced but rather shifted in the frequency axis. If the system did not account for this as in Section 8.5, then it would have also concluded there was a reduction in amplitude.

After the fault was detected, the operating frequency was returned to the observed baseline profile. The system was able to identify the return to the original operating frequency for all sensors, and then declared normal machine operation.

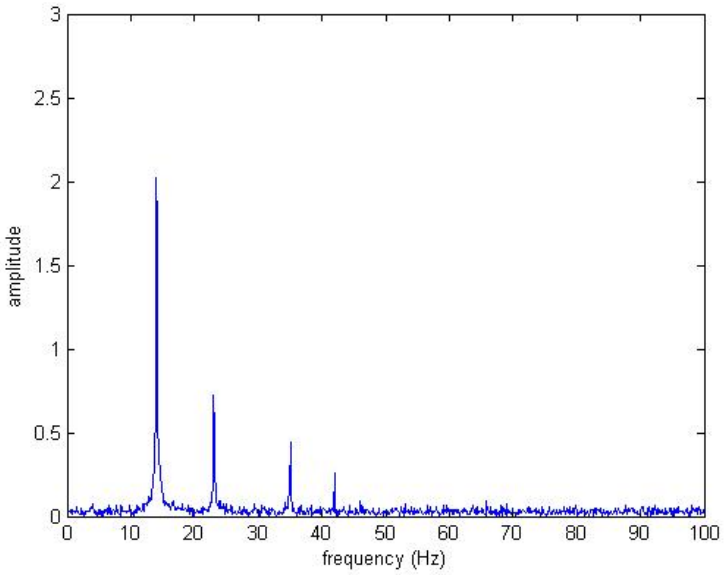

Figure 10.9: RFB Frequency Spectrum, with drifted operating frequency

10.8 Noise-Based Faults

In this last experiment set, the fault signals remain the same as the simulated baseline signals with the exception of increased noise. For a vibrating machine increased levels of noise are expected signatures of screening larger aggregates, structural deterioration from extensive operation, or active environmental conditions. To reflect such, the initial Gaussian random noise, $\mathcal{N}(0,1)$, was amplified as presented in Table 10.12.

	Gaussian Random Noise		
Simulation	Notation	Mean	Standard Deviation
Baseline Signal	$\mathcal{N}(0,1)$	0 g	1.0 g
Experiment 1	$\mathcal{N}(0,2)$	0 g	2.0 g
Experiment 2	$\mathcal{N}(0,3)$	0 g	3.0 g
Experiment 3	$\mathcal{N}(0,5)$	0 g	5.0 g

Table 10.12: Noise Experiments - Gaussian Random Noise

10.8.1 Noise Experiment 1

For all three noise experiments, half an hour of simulated waveform data was monitored. The first experiment uses signals with $\mathcal{N}(0, 2)$ noise, and the corresponding frequency spectrum up to 100 Hz for a time window is presented in Figure 10.10.

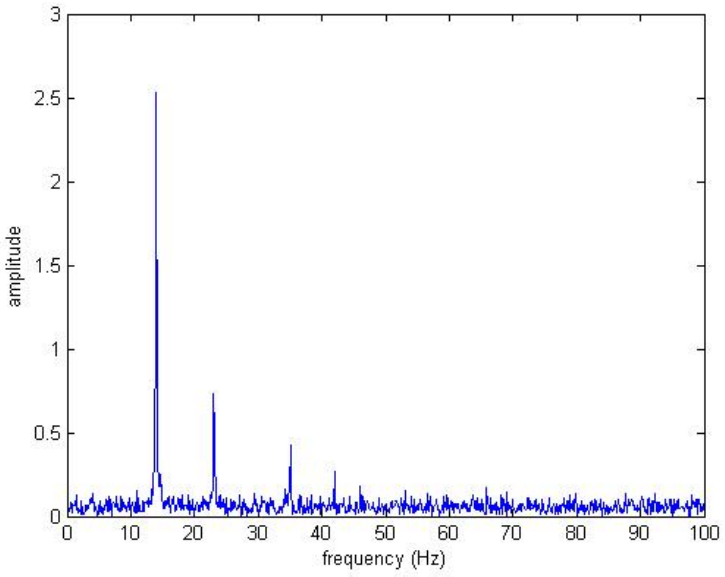

Figure 10.10: RFB Frequency Spectrum, with $\mathcal{N}(0, 2)$ noise

The monitoring system detected increased average g-forces for all sensors, as presented in Table 10.13. For this monitored variable, all sensor locations experienced a similar effect, ranging from a 13 to 16 % increase. Increasing the noise can increase the energy of the system, allowing it to output more g-forces.

	Exceeded Average G-Forces	
Sensor Location	Absolute (g)	Relative Difference (%)
RFB	4.08	13.1
LDB	3.91	13.0
RDB	4.30	14.3
LFB	3.96	15.8

Table 10.13: Simulation Results, Average G-Forces - $\mathcal{N}(0, 2)$ Noise

Another type of detected fault condition was the deviation of the smallest peak amplitude from the baseline profile. Table 10.14 presents the corresponding 42.0 Hz peak amplitudes. The RFB sensor location has multiple values recorded, as over the course of the experiment it returned to acceptable levels, as denoted by 'a'. The RFB 42.0 Hz peak amplitude triggered its fault condition three times throughout the experiment. This repeated fault triggering, especially with the amplitude being too large then too small, and accompanying increased g-forces are strong evidence of noise.

	Peak Amplitude Relative Difference (%)
Sensor Location	42.0 Hz
RFB	{10, a, -14, a, -20}
LDB	11
RDB	-21
LFB	13

Table 10.14: Simulation Results, 42.0 Hz Peak Amplitudes - $\mathcal{N}(0, 2)$ Noise

10.8.2 Noise Experiment 2

In this experiment, the same procedure will be carried out as in the previous test. However, noise will be increased to $\mathcal{N}(0, 3)$. Figure 10.11 presents the frequency spectrum up to 100 Hz for a time window of the signal.

As expected, all average g-forces exceeded the tolerance threshold. Table 10.15 presents the increased averages, ranged from 26 to 32 %, roughly doubling the relative differences from the previous experiment.

	Exceeded Average G-Forces	
Sensor Location	Absolute (g)	Relative Difference (%)
RFB	4.56	26.2
LDB	4.37	26.4
RDB	4.85	28.9
LFB	4.49	31.5

Table 10.15: Simulation Results, Average G-Forces - $\mathcal{N}(0, 3)$ Noise

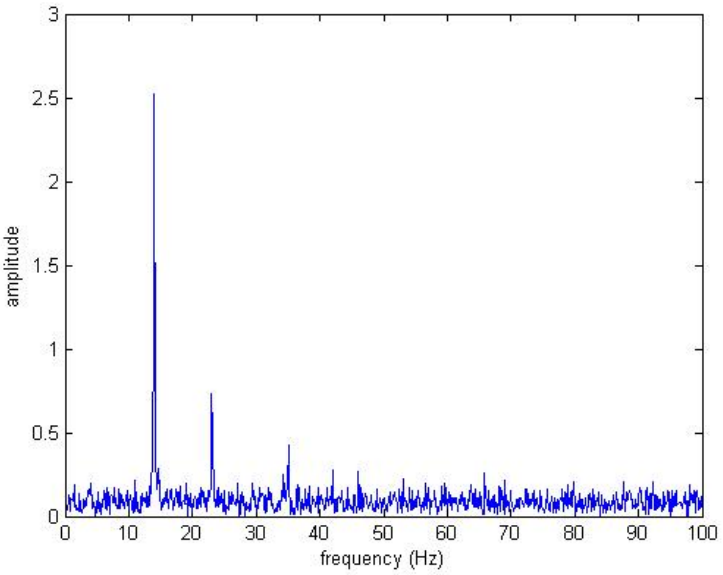

Figure 10.11: RFB Frequency Spectrum, with $\mathcal{N}(0, 3)$ noise

Individual and cross-correlated peak amplitudes were also seen deviating from their baseline profile. Table 10.16 presents the individual sensors' breached fault conditions. It was the three smallest peaks from the baseline profile that were most affected by the noise. All deviated amplitudes for the 23 and 35 Hz peaks returned to acceptable levels, where some went on to trigger a fault condition again. Alternately, all the 42 Hz peaks were completely influenced by noise, continuously maintaining their fault status.

	Peak Amplitude Relative Difference (%)		
Sensor Location	23.0 Hz	35.0 Hz	42.0 Hz
RFB	-	{16, a, 11}	21
LDB	{26, a, 16}	{15, a}	-18
RDB	{-12, a}	{-11, a}	22
LFB	-	{15, a}	32

Table 10.16: Simulation Results, Peak Amplitudes - $\mathcal{N}(0, 3)$ Noise

10.8.3 Noise Experiment 3

The final noise experiment further increases the used noise to $\mathcal{N}(0, 5)$. Figure 10.12 depicts the strength of the noise across the frequency spectrum of a time window.

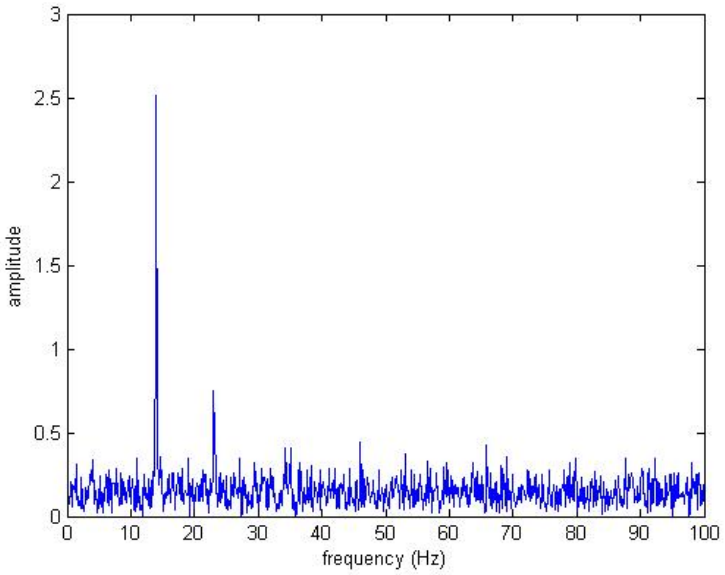

Figure 10.12: RFB Frequency Spectrum, with $\mathcal{N}(0, 5)$ noise

When monitoring the noisy signal, average g-forces exceeded relative tolerances in the range from 55 to 63 %. Table 10.17 presents these findings.

	Exceeded Average G-Forces	
Sensor Location	Absolute (g)	Relative Difference (%)
RFB	5.58	54.7
LDB	5.36	55.0
RDB	5.99	59.1
LFB	5.58	63.3

Table 10.17: Simulation Results, Average G-Forces - $\mathcal{N}(0, 5)$ Noise

Again, the three smallest individual and cross-correlated peak amplitudes were seen deviating from the baseline profile. Some of the peaks were undetectable as a

result of the noise. Table 10.18 presents the individual sensor results, denoting undetectable peaks with an 'L', to indicate they are lost frequency content. Noise has either diminished their amplitudes below the minimum peak amplitude, or neighbouring noise-based peaks are dominating their presence.

	Peak Amplitude Relative Difference (%)		
Sensor Location	23.0 Hz	35.0 Hz	42.0 Hz
RFB	{12, a, -11}	20	128
LDB	L	66	L
RDB	14	14	67
LFB	{16, a, -12}	{11, a}	L

Table 10.18: Simulation Results, Peak Amplitudes - $\mathcal{N}(0, 5)$ Noise

Some of the peak amplitudes were identified as too large, but would return to acceptable operational amplitudes. This was followed by two other triggers, indicating amplitudes were reduced beyond the tolerance.

Like the other noise-based experiments, similar fault signatures are seen. It was also observed that increasing the levels of noise increases the effect of the corresponding symptoms. In summary, signatures of increased levels of noise are:

- increased average g-forces
- repeated triggering of peak amplitudes
- peak amplitudes that are too large, becoming too small (and vise-versa)
- smaller baseline profile peaks being undetected

11 System Testing

11.1 Overview

This chapter documents the system testing performed on an in-house vibrating screen. After a stable baseline profile is constructed experiments are carried out, each imitating a fault condition. The monitoring software in combination with the system network and sensor devices will be examined in an industrial setting. The system will operate at 500 Hz sampling and use 4096 points in the FFT calculations.

11.2 Test Machine

The machine monitored in the testing process is a linear vibrating screen. This type of machine uses linear motion to quickly screen large amounts of material. Figure 11.1 presents a side view, while Figure 11.2 presents a discharge end view.

Figure 11.1: Linear Vibrating Screen - Side View

The vibrating screen is driven by two motors, as seen from the discharge end view. The machine is also equipped with a variable-frequency drive (VFD). VFDs control the frequency of electrical power supplied to a motor, which in turn affects the

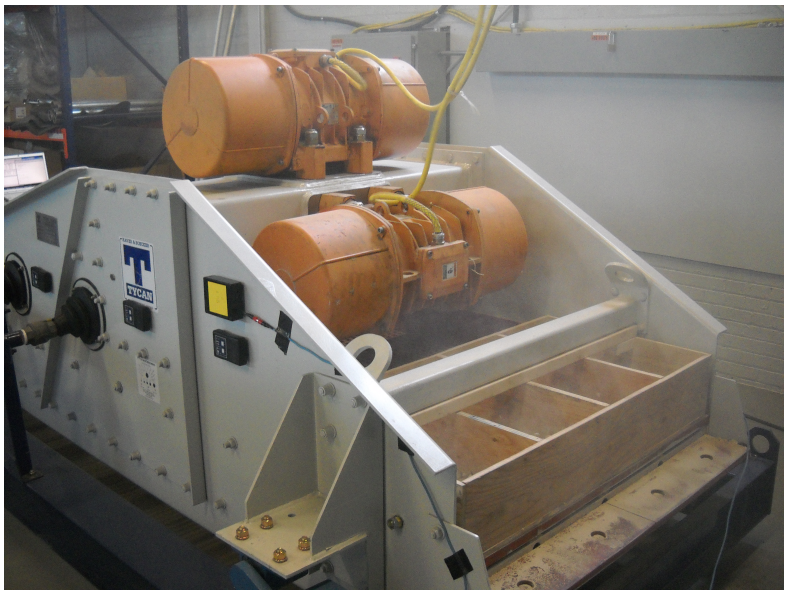

Figure 11.2: Linear Vibrating Screen - Discharge End View

rotational speed of the motor. This allows the operational frequency of the machine to be tuned while running.

Also visible from the discharge end view is a slotted wooden box bolted to the vibrating screen. During testing it housed rocks to imitate the material flow. The box itself was seen vibrating, contributing additional noisy content.

Only two Wi-Fi prototype sensor devices were constructed, so the number of sensors used in system testing was limited. The sensor devices were placed on both sides of the discharge end body, denoted as RDB and LDB.

11.3 Baseline Profile

While constructing a stable baseline profile for the X and Y axes was relatively simple, the Z-axis proved to be more challenging. The vibrating screen's motors rotates in the XY plane, governing motion in those axes. Figure 11.3 presents the X-axis acceleration for a quarter time window. Figure 11.4 contains the corresponding baseline frequency spectrum. Both sensors' X and Y axes appear similar in nature; a single operating frequency peak with little noise and no other spectrum content.

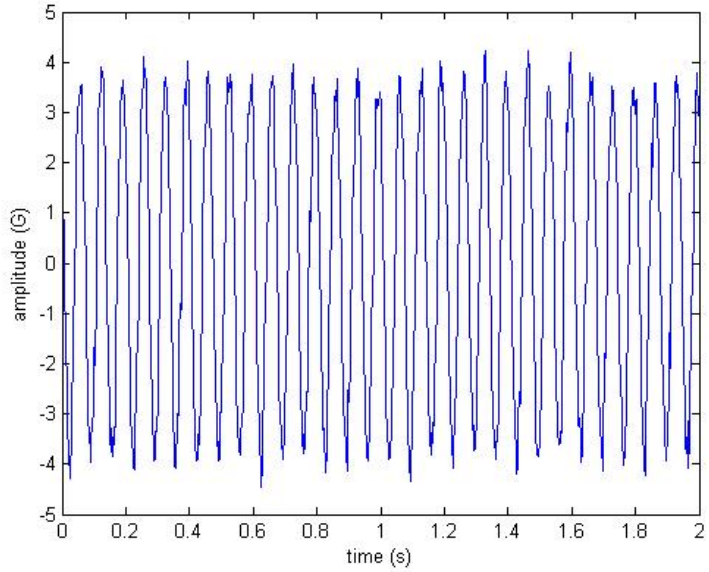

Figure 11.3: Baseline LDB X-Axis Waveform

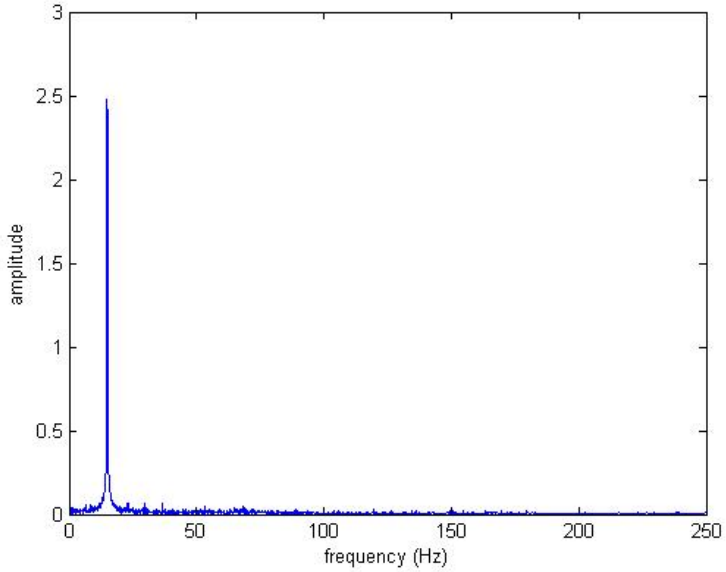

Figure 11.4: Baseline LDB X-Axis Frequency Spectrum

Noise and other frequency content was observed in the Z-axis. The acceleration waveform of the LDB Z-axis is presented in Figure 11.5. The sensor devices' frequency spectrums reveals the contained content, as seen in Figures 11.6 and 11.7. Many of the visible Z-axis peaks were not consistently present, and would trigger a fault condition when monitored. This required the minimum peak amplitude to be greater than the inconsistent peaks.

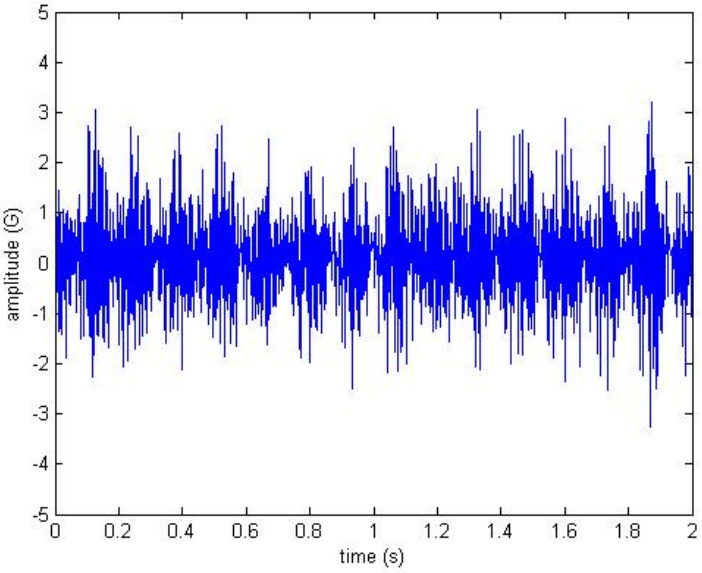

Figure 11.5: Baseline LDB Z-Axis Waveform

The selected parameters used to construct the stable baseline profile are presented below in Table 11.1. Empirical testing refined these values over a few attempts. The resulting baseline profile is presented over Tables 11.2 and 11.3. All sensors devices reported 14.94 Hz as the operating frequency of the vibrating screen.

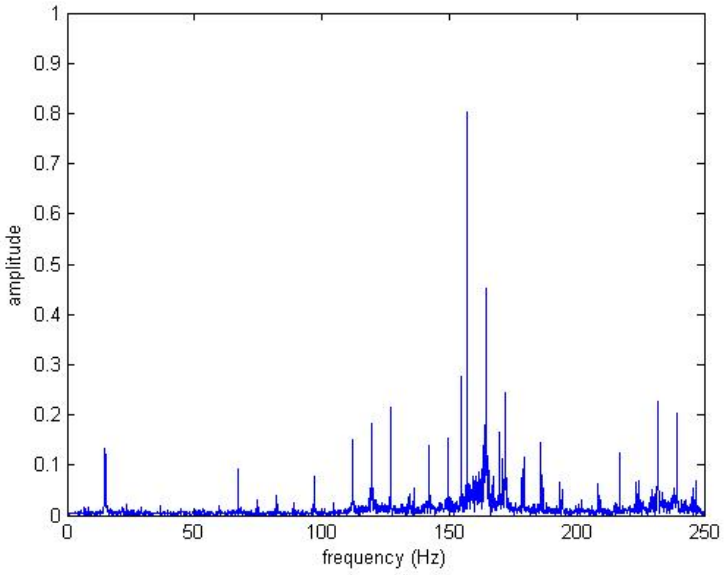

Figure 11.6: Baseline LDB Z-Axis Frequency Spectrum

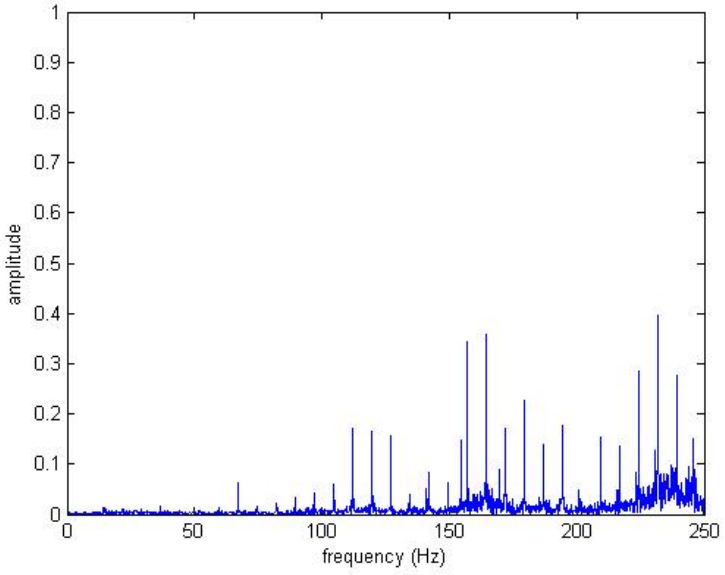

Figure 11.7: Baseline RDB Z-Axis Frequency Spectrum

Minimum Peak Amplitudes	
Individual Peaks	0.60
Cross-Correlated Peaks	0.30

Tolerance Thresholds	
Main G-Force	5.0%
Average G-Force (X, Y)	5.0%
Average G-Force (Z)	30.0%
Operating Frequency	0.050 Hz
Peak Frequency	0.122 Hz
Peak Amplitude	30.0%
Cross-Correlated Peak Amplitude	30.0%

Algorithm Parameters	
`Increment`	1
`Start`	1
`Normal`	0
`Fault`	8
`Maximum`	8

Table 11.1: Selected System Parameters

	Main	**Average G-Forces**		
Sensor	**G-Force**	X	Y	Z
RDB	5.34	3.91	3.68	0.114
LDB	5.50	4.01	3.82	0.301

Table 11.2: Baseline Profile, G-Forces

11.4 Loose Components

While the constructed baseline profile was deemed stable, the LDB Z-axis peak at 157.98 Hz occasionally trigged its fault condition when the machine was restarted between experiments. At times the particular baseline profile peak would disappear, while a new peak would emerge nearby. Peaks up to 164.54 Hz were seen emerging replacing the original peak. Figure 11.8 presents the altered LDB Z-axis spectrum.

		Peak Frequency / Amplitude	
Sensor	**Axis**	14.94 Hz	157.98 Hz
RDB	X	3.51	-
	Y	3.28	-
	Z	-	-
LDB	X	3.42	-
	Y	3.35	-
	Z	-	0.77

Table 11.3: Baseline Profile, Peak Frequencies

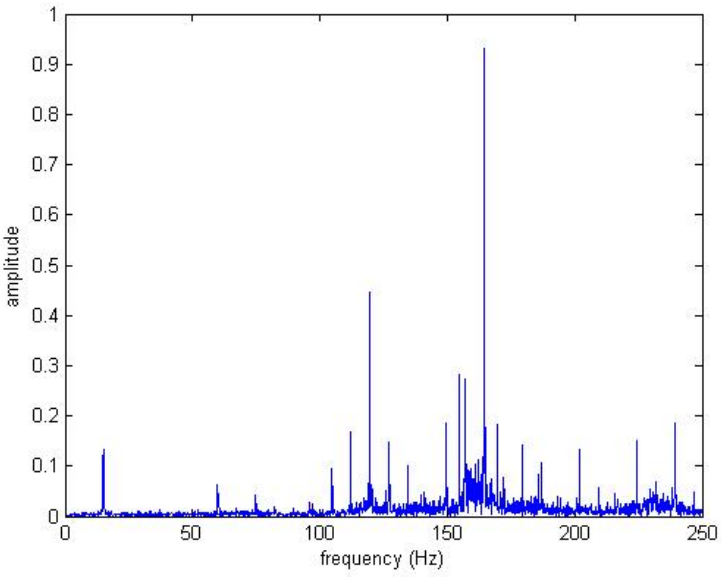

Figure 11.8: LDB Z-Axis Frequency Spectrum, with loose component

Such a shift in frequency indicates a loose component on the machine that is susceptible to changing its rotational path. Since the test machine was in good health aside from the induced faults, the unknown fault condition was attributed to the mounted wooden box containing the aggregate. Unlike any other part of the machine, the bolts mounting the box were loose and able to vibrate. Nonetheless, it provided an experimental example of signatures pertaining to loose components.

11.5 Induced Impacts

In this experiment the vibrating screen was subject to impacts caused by a wooden block placed in its path. The impacts occurred on the left discharge end body, near the LDB sensor. Such impacts could be caused by loose or damaged components. Alternately, when aggregate material builds up it can overflow and obstruct machine movement.

The impacts were introduced until the fault signature was revealed. As a result of the impacts, the LDB sensor experienced increased noise, as presented in Figure 11.9. The triggered fault condition was due to increased average g-forces, as presented Table 11.4.

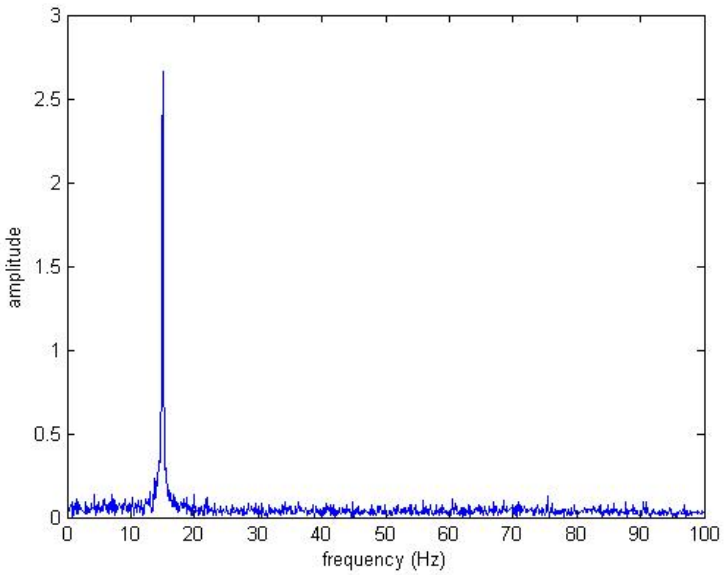

Figure 11.9: LDB X-Axis Frequency Spectrum, with induced impacts

		Exceeded Average G-Forces	
Sensor	**Axis**	Amplitude (g)	Relative Difference (%)
LDB	Main	5.97	8.54
	X	4.64	15.71
	Z	0.729	142.19

Table 11.4: Fault Conditions - Induced Impact

After the faults were detected, the wooden block was removed from the vibrating screen's path. The system was able to correctly identify that the vibrating screen was back to normal operation.

11.6 Motor Failures

As mentioned, the linear vibrating screen is driven by two electric motors. If a failure occurs on one of the two motors, the screening efficiency would be drastically reduced. If both motors fail, the vibrating screen ceases to move. This experiment set monitors single and double motor failures.

11.6.1 Single Motor Failure

The single motor failure was induced by simply removing power from the top mounted motor. There were numerous triggered fault conditions; Table 11.5 presents the g-force based faults conditions. The average g-forces in the Y-axis were diminished, yet the X-axis g-forces were slightly increased. Ultimately, less vibrational g-forces were being produced.

		Exceeded Average G-Forces	
Sensor	**Axis**	Amplitude (g)	Relative Difference (%)
LDB	X	4.27	6.48
	Y	3.35	-12.30
RDB	X	4.18	6.91
	Y	3.21	-12.77

Table 11.5: Fault Conditions - Single Motor Failure

Another set of experienced fault conditions was the unanimous detection of a drift in operating frequency by -0.05 Hz. It is intuitive that the loss of a motor would slow

down the machine's movement. After the faults were realized, power was restored to the motor where the monitoring software quickly identified the return to normal operation.

11.6.2 Double Motor Failure

In this experiment, power to both motors on the vibrating screen was removed. In this state, the machine is inoperative. All baseline profile peaks were no longer detectable, and experienced average g-forces all reported minimal values.

Interestingly, the LDB sensor reported an operating frequency of 60.04 Hz, while the RDB reported 177.0 Hz. These values are either the product of noise in the sensor circuitry or the industrial environment.

11.7 Machine Unbalance

A technique used to imitate a spring failure involves unbalancing the vibrating screen. This was achieved by raising the base frame of the machine on three corners and letting the fourth corner sag. For this experiment, the LDB corner remained on the ground.

The only detected fault condition was the absence of the 156.98 Hz peak from the LDB Z-axis. While it was mentioned that this peak was susceptible to movement across the frequency spectrum, no other experiment had this content completely removed. Figure 11.10 presents the modified Z-axis frequency spectrum of the LDB sensor.

11.8 Drifting Operating Frequency

This experiment drifts the operating frequency of the vibrating screen while it is active. An altered operating frequency could be attributed to changing dynamics of the vibrating screen, and is also an indicator of reduced efficiency. Machine vibrations are tuned for certain materials and screen sizes, and shifting away from the desired characteristics only degrades performance.

To drift the operating frequency, the variable-frequency drive (VFD) was used to increase the frequency of the power signal. The inputs and unanimously detected operating frequencies are presented in Table 11.6.

The operating frequency was slowly decreased until the monitoring system identified a breached fault condition. The first operating frequency to trigger a fault, 15.0 Hz, exceeded the threshold tolerance placed on the baseline profile.

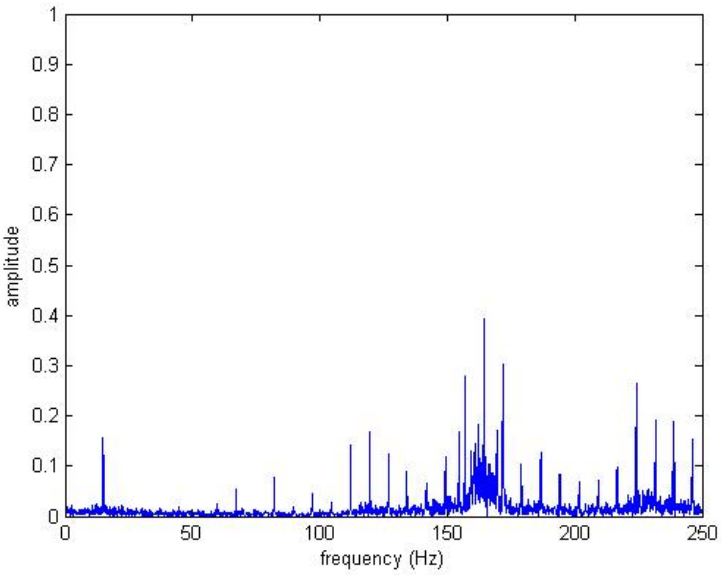

Figure 11.10: LDB Z-Axis Frequency Spectrum, with machine unbalance

	Frequency (Hz)	
Test	Electrical Power	Machine's Operation
Baseline Profile	50.0	14.94
Fault Experiment	40.0	15.00

Table 11.6: Results - Drifting Operating Frequency

After the fault notification, the original operating frequency was restored. The monitoring system was able detect the operation had returned to the baseline profile.

11.9 Network Performance

Over the course of the fault experiments, the communication network between the sensor devices and monitoring computer was observed. Only in two instances throughout the testing day did a sensor send a malformed packet. The monitoring system was able to automatically restart communications within seconds, quickly resuming the monitoring process. The monitoring system's network proved to be reliable and robust.

It is expected that manufactured sensor devices with surface mount components

will have increased performance compared to the current prototype. Electrical paths can be optimized, and contacts between components will be enhanced. As suggested by the WiFly datasheet, an improved layout would maximize distance between the Wi-Fi module and any metallic mounting bracket.

12 Conclusion

12.1 Discussion

This thesis presented the design, implementation and testing of a condition monitoring system for use on vibrating screens. The system was able to collect acceleration data from remote sensor devices as to construct a baseline profile of a machine. Subsequent data from operation was then compared to this profile for fault detection purposes. Fault signatures were detected and recorded through various experiments. A discussion will be carried out evaluating the performance of the hardware and software system components.

The upgraded hardware utilized in the monitoring system proved to be successful. The sensor devices equipped with Wi-Fi modules performed better than their Bluetooth equivalents. The wireless network hosted by the router allowed for increased transmission rates over larger distances, with an increased node capacity. Transmissions errors were very rare, and the system was able to easily resynchronize communications.

The monitoring computer was able to reliably communicate with the sensor devices through the network, allowing acceleration data to be supplied to the monitoring software. The computer was easily interfaced with other networks and computers, and when given Internet access could autonomously send email fault notifications providing users with a log file.

The monitoring software successfully constructed stable baselines profiles and detecting various fault conditions. For both virtual and live industrial experiments, minor tuning of the system parameters achieved the desired monitoring response. System testing revealed the monitoring technique could be utilized in a practical industrial setting.

Slight modifications to the monitoring software could allow for finer tolerance thresholds to be put on the monitored variables. Due to the nature of the vibrating screen, the X and Y axes accelerations are similar while the Z-axis is very different. As such, the Z-axis should have more independent parameters such as minimum peak amplitudes and the peak frequency range. Smaller deviation bounds could then be placed on the X and Y axes, while Z-axis profiles can remain stable using its own tolerances.

The completed condition monitoring system is a self-contained product. However, additional monitoring and testing on more types of vibrating screens would provide further insight on how to refine the system. The following section contains known suggestions on how to advance the condition monitoring system.

12.2 Future Work

This section provides a list of suggested future work to expand the scope of the implemented condition monitoring system. The following items are recognized and desired by the sponsoring company:

1. Introduce fault notification hardware by interfacing the monitoring computer with a programmable logic controller (PLC). Industrial control rooms commonly use PLCs to control and monitor machinery. Users would have much more flexibility when interfacing with the system.

2. Expand the condition monitoring software by increasing the number of monitored variables used in the machine profile. Ellipse fitting algorithms could provide eccentricity and phase angles of the machine rotation. New monitored variables would require their own tolerance thresholds.

3. Work has already begun creating a website and database so that data recordings and fault notifications could be automatically uploaded to a centralized server over the Internet. This would accumulate data recordings, enabling the opportunity for long term machine analysis. It would also be possible to select system parameters through a web interface, and have the monitoring computer retrieved the desired settings.

4. Once a sufficient knowledge base for fault signatures is compiled, the monitoring system could be provided with a mapping from deviated monitored variables to machine faults. Instead of notifying users of the breached monitored variables, specific faults and their severity could be presented to the user. Such an upgrade would classified the monitoring system as an expert system.

Bibliography

[1] D. C. Baillie and J. Mathew. A comparison of autoregressive modeling techniques for fault diagnosis of rolling element bearings. *Mechanical Systems and Signal Processing*, 10(1):1 – 17, 1996.

[2] Changzheng Chen and Changtao Mo. A method for intelligent fault diagnosis of rotating machinery. *Digital Signal Processing*, 14(3):203–217, 2004.

[3] Seung-deog Choi, B. Akin, M.M. Rahimian, and H.A. Toliyat. Fault diagnosis implementation of induction machines based on advanced digital signal processing techniques. *Twenty-Fourth Annual IEEE Applied Power Electronics Conference and Exposition*, pages 957–963, Feb. 2009.

[4] E.H. Clayton, Bong-Hwan Koh, Guoliang Xing, Chien-Liang Fok, S.J. Dyke, and Chenyang Lu. Damage detection and correlation-based localization using wireless mote sensors. *Intelligent Control, 2005. Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation*, pages 304 –309, jun. 2005.

[5] Stephan Ebersbach and Zhongxiao Peng. Expert system development for vibration analysis in machine condition monitoring. *Expert Systems with Applications*, 34(1):291–299, 2008.

[6] Paul M. Frank. Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy: A survey and some new results. *Automatica*, 26(3), 1990.

[7] D. Fuessel and R. Isermann. Hierarchical motor diagnosis utilizing structural knowledge and a self-learning neuro-fuzzy scheme. *Industrial Electronics, IEEE Transactions*, 47(5), 2000.

[8] K.B. Goode, B.J. Roylance, and J. Moore. Development of model to predict condition monitoring interval times. *Ironmaking and Steelmaking*, 27(1):63 – 68, 2000.

[9] A. Grall, C. Brenguer, and L. Dieulle. A condition-based maintenance policy for stochastically deteriorating systems. *Reliability Engineering and System Safety*, 76(2):167 – 180, 2002.

[10] H.K. Hoidalen and M. Runde. Continuous monitoring of circuit breakers using vibration analysis. *Power Delivery, IEEE Transactions on*, 20(4):2458 – 2465, Oct. 2005.

[11] Jean-Marc Irazabal and Steve Blozi. I2c manual. `http://www.nxp.com/documents/application_note/AN10216.pdf`, March 2003.

[12] Andrew K.S. Jardine, Daming Lin, and Dragan Banjevic. A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, 20(7):1483–1510, 2006.

[13] Saad Ahmed Saleh Al Kazzaz and G. K. Singh. Experimental investigations on induction machine condition monitoring and fault diagnosis using digital signal processing techniques. *Electric Power Systems Research*, 65(3):197–221, 2003.

[14] E.G. Laukonen, K.M. Passino, and V. Krishnaswami. Fault detection and isolation for an experimental internal combustion engine via fuzzy identification. *Control Systems Technology, IEEE Transactions*, 3(3), 1995.

[15] Yaguo Lei, Zhengjia He, and Yanyang Zi. A new approach to intelligent fault diagnosis of rotating machinery. *Expert Systems with Applications*, 35(4):1593–1600, 2008.

[16] Electrofun Ltd. Wifly gsx 802.11b/g serial module. `http://www.electrofun.biz/catalog/product_info.php?products_id=168`, July 2009.

[17] G. Y. Luo, D. Osypiw, and M. Irle. On-line vibration analysis with fast continuous wavelet algorithm for condition monitoring of bearing. *Journal of Vibration and Control*, 9:931–947, 2003.

[18] Mano Ram Mauryaa, Praveen K. Paritoshb, and Raghunathan Rengaswamy. A framework for on-line trend extraction and fault diagnosis. *Engineering Applications of Artificial Intelligence*, 23(6), Sept. 2010.

[19] S. Nandi, H.A. Toliyat, and Xiaodong Li. Condition monitoring and fault diagnosis of electrical motors-a review. *Energy Conversion, IEEE Transactions on*, 20(4):719 – 729, dec. 2005.

[20] Hasan Ocak, Kenneth A. Loparo, and Fred M. Discenzo. Online tracking of bearing wear using wavelet packet decomposition and probabilistic modeling: A method for bearing prognostics. *Journal of Sound and Vibration*, 302(4-5):951–961, 2007.

[21] Jay Parlar. *Vibration Analysis and Vibrating Screens: Theory and Practice*. PhD thesis, McMaster University, 2010.

[22] David Lorge Parnas. Document based rational software development. *Knowledge-Based Systems*, April 2009.

[23] David Lorge Parnas and Jan Madey. Functional documents for computer systems. *Science of Computer Programming*, 1995.

[24] R.J. Patton, C.J. Lopez-Toribio, and F.J. Uppal. Artificial intelligence approaches to fault diagnosis. *Condition Monitoring: Machinery, External Structures and Health*, Apr. 1999.

[25] Z. K. Peng and F. L. Chu. Application of the wavelet transform in machine condition monitoring and fault diagnostics: a review with bibliography. *Mechanical Systems and Signal Processing*, 18(2):199–221, 2004.

[26] A. Ramirez-Trevino, E. Ruiz-Beltran, and I. Rivera-Rangel. Online fault diagnosis of discrete event systems. a petri net-based approach. *Automation Science and Engineering, IEEE Transactions*, 4(1), 2007.

[27] R.SaravanaKumar, K.Vinoth Kumar, and K.K.Ray. Fuzzy logic based fault detection in induction machines using lab view. *IJCSNS International Journal of Computer Science and Network Security*, 9(9), Sep. 2009.

[28] C. Servire and P. Fabry. Blind source separation of noisy harmonic signals for rotating machine diagnosis. *Journal of Sound and Vibration*, 272(1-2):317–339, 2004.

[29] H. Sneider and P.M. Frank. Observer-based supervision and fault detection in robots using nonlinear and fuzzy logic residual evaluation. *Control Systems Technology, IEEE Transactions*, 4(3), 1996.

[30] Ehsan Sobhani-Tehrani and Khashayar Khorasani. *Fault Diagnosis of Nonlinear Systems Using a Hybrid Approach*. Springer, 2009.

[31] Manny Soltero, Jing Zhang, and Chris Cockril. Rs-422 and rs-485 standards overview and system configurations. `http://focus.ti.com/lit/an/slla070d/slla070d.pdf`, May 2010.

[32] Qiao Sun, Ping Chen, Dajun Zhang, and Fengfeng Xi. Pattern recognition for automatic machinery fault diagnosis. *Journal of Vibration and Acoustics*, 126(2):307–316, 2004.

[33] Wei Sun, Ahmet Palazolu, and Jose A. Romagnoli. Detecting abnormal process trends by wavelet-domain hidden markov models. *AIChE Journal*, 49(1), Jan. 2003.

[34] J. Vollmer, Yanting Hu, T. Neumann, and E. Solda. Construction kit for low-cost vibration analysis systems based on low-cost acceleration sensors. *IEEE/ASME International Conference*, pages 463–468, Jul. 2009.

[35] W. Wang. A model to determine the optimal critical level and the monitoring intervals in condition-based maintenance. *International Journal of Production Research*, 38(6):1425 – 1436, 2000.

[36] Bo-Suk Yang, Dong-Soo Lim, and Andy Chit Chiow Tan. Vibex: an expert system for vibration fault diagnosis of rotating machinery using decision tree and decision table. *Expert Systems with Applications*, 28(4):735 – 742, 2005.