Development of a Coal Wagon Monitoring and Control System

A dissertation submitted by

Adam Roberts-Apps

in fulfilment of the requirements of

Courses ENG4111 & ENG4112 Research Project

towards the degree of

Bachelor of Engineering (Mechanical)

Submitted: October, 2013
Abstract

The project analyses and tests the potential use of low cost hobbyist electronics to monitor and control mechanical faults in Coal Wagons. An Arduino Due microcontroller board and sensors similar to those found in smart devices have been integrated into the initial concept. Low level hardware will make it financially viable to instrument wagons on a mass scale.

Instrumentation to permanently monitor and control the mechanical condition of Coal Wagons is not commercially available; while the value of fixed track side systems is debatable. The idea behind this project was to use proven fixed plant condition monitoring techniques, low cost consumer level electronics and ‘the internet of things’ to create a synergized solution to overcome the shortfalls of other systems.

The objective of this project was to build and program a controller unit to process and record data, while keeping the design open for later integration of an ABS system. A Failure Modes, Effects and Criticality Analysis (FMECA) was conducted on Coal Wagons to identify the mechanical componentry in need of monitoring. Research was conducted on low weight algorithms for processing data, sensor types, communication protocols and C/C++ programming.

Currently the unit can process and log vibration, temperature, acceleration, tilt, displacement, axle speed and strain data from sensors to an SD card, while averages are sent via a wireless 3G router to an internet cloud service called Xively. GPS stamps are assigned to each data set which allows for tracking and mapping of faults on the rail corridor, either in real time as a Xively meta-data feed, or
by downloading the data from the SD card and displaying it in 2D or 3D using Google Maps or Google Earth Pro respectively.

More code debugging is required to improve program runtime and to add a greater degree of error checking capability. To take the design from the bread board/concept phase and house the controller and sensors in weather proof enclosures for fitting to Coal Wagon, with future plans to integrate ABS control to limit the frequency of wheel flat.

The proposed micro-controller based platform holds promise for further development to meet the requirements of those responsible for maintaining rolling stock assets.
Limitations of Use

The Council of the University of Southern Queensland, its Faculty of Health, Engineering and Sciences, and the staff of the University of Southern Queensland, do not accept any responsibility for the truth, accuracy or completeness of material contained within or associated with this dissertation.

Persons using all or any part of this material do so at their own risk, and not at the risk of the Council of the University of Southern Queensland, its Faculty of Health, Engineering and Sciences or the staff of the University of Southern Queensland.

This dissertation reports an educational exercise and has no purpose or validity beyond this exercise. The sole purpose of the course pair entitled “Research Project” is to contribute to the overall education within the student’s chosen degree program. This document, the associated hardware, software, drawings, and other material set out in the associated appendices should not be used for any other purpose: if they are so used, it is entirely at the risk of the user.

Executive Dean
Faculty of Health, Engineering and Sciences
Certification of Dissertation

I certify that the ideas, designs and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.

I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

ADAM ROBERTS-APPS

0061004459

____________________________________
Signature

____________________________________
Date
Acknowledgments

I would like to thank my supervisor Prof John Billingsley for explaining the correct configuration of resistors for my Instrument Amplifier. A special thanks goes out to Patrick Torok for providing Coal Wagon design information and Kaine Rathbone for giving me general guidance with the project.

Adam Roberts-Apps

University of Southern Queensland

October 2013
Contents

Abstract i

Acknowledgments v

List of Figures xi

List of Tables xiv

Chapter 1 Project introduction 1

1.1 Introduction ........................................... 1

1.2 Objective and scope .................................... 1

1.3 Dissertation overview .................................. 2

Chapter 2 Literature review 4

2.1 Background ............................................ 4

2.2 Current technology ..................................... 4

2.3 Wagon construction .................................... 6

2.3.1 Hopper/chassis .................................... 7
<table>
<thead>
<tr>
<th>CONTENTS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.2 Bogie</td>
<td>8</td>
</tr>
<tr>
<td>2.3.3 Braking system</td>
<td>9</td>
</tr>
<tr>
<td>2.4 Embedded systems</td>
<td>10</td>
</tr>
<tr>
<td>2.4.1 Micro-controllers</td>
<td>11</td>
</tr>
<tr>
<td>2.4.2 Micro-electro-mechanical systems elements</td>
<td>13</td>
</tr>
<tr>
<td>2.4.3 Piezoelectric transducers</td>
<td>17</td>
</tr>
<tr>
<td>2.4.4 Load cells, strain gauges and thermocouples</td>
<td>19</td>
</tr>
<tr>
<td>2.4.5 Proximity and displacement sensing</td>
<td>21</td>
</tr>
<tr>
<td>2.4.6 Global positioning system modules</td>
<td>24</td>
</tr>
<tr>
<td>2.5 Data processing</td>
<td>27</td>
</tr>
<tr>
<td>2.5.1 Pulse timing</td>
<td>27</td>
</tr>
<tr>
<td>2.5.2 Parsing</td>
<td>28</td>
</tr>
<tr>
<td>2.5.3 Transient signals</td>
<td>29</td>
</tr>
<tr>
<td>2.5.4 Cycle counting</td>
<td>35</td>
</tr>
<tr>
<td>2.5.5 Alarming</td>
<td>36</td>
</tr>
<tr>
<td>2.5.6 Linear quadratic estimation</td>
<td>37</td>
</tr>
<tr>
<td>2.6 Digital communications</td>
<td>40</td>
</tr>
<tr>
<td>2.6.1 Universal asynchronous reviver/transmitter</td>
<td>41</td>
</tr>
<tr>
<td>2.6.2 Inter-integrated circuit</td>
<td>42</td>
</tr>
<tr>
<td>2.6.3 Serial peripheral interface bus</td>
<td>42</td>
</tr>
</tbody>
</table>
CONTENTS

2.6.4 Controller area network bus .......................... 43
2.6.5 Modbus ............................................. 44
2.6.6 Internet suite ........................................ 44
2.6.7 Ethernet and WiFi ................................... 47
2.6.8 ZigBee ................................................. 48

Chapter 3  Failure modes and counter measures 52

3.1 FMECA .................................................... 52
3.2 Wheel flats .............................................. 53
3.3 Bearing faults ........................................... 54
3.4 Draft gear fatigue and dynamics ....................... 57
3.5 Brake pressure monitoring ............................... 58
3.6 Wagon body dynamics .................................. 59

Chapter 4  Concept design 60

4.1 Proposed layout ......................................... 60
4.2 Hardware selection ...................................... 62
  4.2.1 Micro-controller .................................... 64
  4.2.2 Sensors ............................................. 66
  4.2.3 Displacement sensor ................................ 73
4.3 Scripting ............................................... 74
## CONTENTS

Appendix C  Risk assessment  

Appendix D  OTSI incidents  

Appendix E  FMECA  

Appendix F  Coal Wagon information  

| F.1  | SKF rolling element bearings   | 128  |
| F.2  | SKF engineering calculator     | 129  |
| F.3  | Braken wheel set               | 130  |
| F.4  | Typical draft gear design      | 131  |
| F.5  | Bogie spring defection graph   | 132  |
| F.6  | NHRH wagon details manual      | 133  |

Appendix G  Electronic schematics  

| G.1  | Arduino Due                    | 136  |
| G.2  | IMU                            | 137  |

Appendix H  Third party source code  

| H.1  | C++ complex number handler     | 139  |
| H.2  | Processing FFT analyser        | 148  |
| H.3  | Hacked Python Karlman filter code | 151  |
List of Figures

2.1 LEADER cabin display (New York Air Brake 2013) . . . . . . . . . 5
2.2 NHRH Coal Wagon in current use . . . . . . . . . . . . . . . . . . 7
2.3 Unloading doors and opening mechanism . . . . . . . . . . . . . 7
2.4 Annotated bogie model . . . . . . . . . . . . . . . . . . . . . . . . 9
2.5 Air brake system . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
2.6 Arduino Due board (Arduino 2013) . . . . . . . . . . . . . . . . . . 13
2.7 The five motion that MEM sensor can measure (O’Reilly & Weinberg 2010) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
2.8 Workings of a MEMS sensor (Young 2006) . . . . . . . . . . . . . 17
2.9 Low cost piezoelectric sensor (MSI 1999) . . . . . . . . . . . . . 18
2.10 Quarter, half and full bridge configuration from left to right re- spectively (Matsuno, Adachi, Nakayama & Watanabe 1993) . . . 20
2.11 Capacitive proximity sensor block diagram (Repas 2006) . . . . 23
2.12 Differential GPS concept (Morgan-Owen & Johnston 1995) . . . 26
2.13 Example of aliasing (Billingsley 2010) . . . . . . . . . . . . . . 30
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.14</td>
<td>Oversampling (a) 2.56x and (b) 20x (Anders 2012)</td>
<td>31</td>
</tr>
<tr>
<td>2.15</td>
<td>Fourier series principal (Anders 2012)</td>
<td>33</td>
</tr>
<tr>
<td>2.16</td>
<td>Example FFT</td>
<td>34</td>
</tr>
<tr>
<td>2.17</td>
<td>Time domain of common windows (Courtois 2011)</td>
<td>35</td>
</tr>
<tr>
<td>2.18</td>
<td>Machine monitoring alarm types</td>
<td>38</td>
</tr>
<tr>
<td>2.19</td>
<td>$I^2C$ bus (Robot Electronics 2013)</td>
<td>42</td>
</tr>
<tr>
<td>2.20</td>
<td>SPI layout (Byte Paradigm 2013)</td>
<td>43</td>
</tr>
<tr>
<td>2.21</td>
<td>XBee (Digi 2013)</td>
<td>49</td>
</tr>
<tr>
<td>3.1</td>
<td>Rolling element bearing components</td>
<td>56</td>
</tr>
<tr>
<td>4.1</td>
<td>Concept data logger layout</td>
<td>61</td>
</tr>
<tr>
<td>4.2</td>
<td>Proposed design to be implemented in the field</td>
<td>61</td>
</tr>
<tr>
<td>4.3</td>
<td>Equipment used to test the inductive sensor</td>
<td>69</td>
</tr>
<tr>
<td>4.4</td>
<td>Quarter strain bridge circuit reference</td>
<td>70</td>
</tr>
<tr>
<td>4.5</td>
<td>GPS Bee</td>
<td>71</td>
</tr>
<tr>
<td>4.6</td>
<td>Arduino Ethernet shield</td>
<td>72</td>
</tr>
<tr>
<td>4.7</td>
<td>Telstra’s coverage of the Hunter Network</td>
<td>73</td>
</tr>
<tr>
<td>4.8</td>
<td>Testing of the tachometer code</td>
<td>79</td>
</tr>
<tr>
<td>4.9</td>
<td>512Hz bandwidth FFT sample</td>
<td>86</td>
</tr>
<tr>
<td>4.10</td>
<td>Data sampled using the Listing 4.12</td>
<td>88</td>
</tr>
</tbody>
</table>
4.11 Xively alarm set up pane ........................................ 90

5.1 SD card data imported into Google Fusion charts ............... 93

5.2 Result of Xively **datastreams** and **location** feeds .......... 93

5.3 All Xively channel feeds ........................................... 94

D.1 El Zorro derailment (OTSI 2012\textit{a}) .......................... 114

D.2 Gulgong failure (OTSI 2012\textit{c}) ................................. 115

D.3 Broggabri bridge derailment (OTSI 2012\textit{b}) ................... 116

D.4 Clifton axle failure (OTSI 2011) .................................. 117

F.1 Bearing fault frequencies for a 177 mm ID tapper roller ....... 129
## List of Tables

2.1 RMC reference ................................................. 51

3.1 Proposed countermeasures .............................. 53

4.1 Summary of the Arduino Due (Arduino 2013a) .......... 65

4.2 MPU-9150 gyro features (InvenSense 2012a) .......... 67

4.3 MPU-9150 Accelerometer features (InvenSense 2012a) 68

4.4 MPU-9150 Temperature sensor and general specifications (InvenSense 2012a) ................................. 68

4.5 Summary of the 4010045 inductive proximity sensor ............................... 69

4.6 Micro-Measurements C2A-06-062LW-350 strain gauge specifications ................................. 70

4.7 GPS Bee specifications ................................. 72
Chapter 1

Project introduction

1.1 Introduction

Efficient and reliable rail is a key point of interest for mining companies and the services industries they employ. Our ability to deliver minerals to port gives Australia a competitive advantage over other resource rich nations with inferior infrastructure networks. Therefore, to maintain and/or improve Australia’s competitiveness in this sector, we need to find innovative ways to improve the availability of ageing rail assets.

There are many ways this can be achieved however; this project investigates one option, that being the possible application of hobbyist micro-controllers and sensors to form a instrument that can aid rolling stock operator/owners improve their programmed maintenance and overall asset reliability.

1.2 Objective and scope

To prototype a cost effective instrument that can measure, report and output controls based on input signals generated by Coal Wagon conditions; however; it is hopeful that this project can be easily adapted to other applications and
1.3 Dissertation overview

industries.

As stated in the Project Specification under Appendix A, the sub-objectives of the project are explained below:

- To research information related to the types of rolling stock failures, current technology and capability of similar instruments, available hardware, programming languages and data processing methods
- Design a system architecture
- Build a prototype
- Write software/firmware for the controller and computer
- Be able to display data in a user friendly format

1.3 Dissertation overview

This dissertation is divided into seven chapters. Chapter 1 introduces the research topic to the reader and explains on the objectives, scope of the research and the project methodology. Chapter 2 will summarize the details on the literature reviewed, before commencing on the design stage. Among the topics covered in this chapter; identifying the need for a continuous wagon monitoring system, current technologies, wagon construction, micro-controller hardware, computer programming and data processing algorithms.

Chapter 3 focuses on the outcomes of a FMEA undertaken on a generic wagon. A detailed breakdown of a Coal Wagon’s sub-assemblies and their possible failure modes can be found here. Counter measures to incorporate into the controller are discussed and selected based on a risk assessment.

The actual prototype device and coding will be discussed in Chapter 4. Moreover, this chapter will give all the relevant calculations and processes used so the device
1.3 Dissertation overview

can input and manipulate data. After the actual prototyping in Chapter 5, all the results from testing the device is shown and discussed. Finally, the conclusion will be addressed in Chapter 6 by discussing the project’s outcomes, shortfalls and further work.
Chapter 2

Literature review

2.1 Background

Structural and mechanical failures have plagued the rail industry for decades. The Office of Transport Safety Investigation (OTSI), has documented numerous incidents involving rolling stock, which have either caused risk or diminished the health and safety of rail workers or public, not to mention the costs of investigations, fines, repair of assets and damage to property.

To put some sort of perspective to the frequency and type of incidents, a brief summary has been compiled from reports published by OTSI of recent incidents which can be viewed in Appendix E

2.2 Current technology

Currently there are only a few devices that can continuously monitor the condition of rolling stock with the majority of these still in development. They are known as the Train Dynamics Monitor (TDM), Wheel Impact Load Detector (WILD), Locomotive Engineer Assist Display & Event Recorder (LEADER) and TrainStar systems, to mention a few.
The TDM system measures the coupler force and body acceleration of the wagons using an array of accelerometers and load cells. The data collected from the sensors is processed, relayed back to the driver via a LED panel (Gorman, Roach & Cole 2000). The information displayed on the LED panel allows the driver to make informed decisions about the current dynamics of the trailing wagons and make any necessary adjustments to the controls.

The TDM is a simplified version of the earlier developed LEADER system. LEADER can monitor and display several more parameters, namely:

- Horizontal view of track, track curvature representation, overhead view of track, bridge, signal, crossing, draft buff forces, brake pipe pressure, brake cylinder pressure head end information section, feed valve setting, fuel consumed, consist length, speed limit, brake pipe reduction, safe reduction indication and acceleration (New York Air Brake 2013).
Minimal information could be found on the TrainStar system although it was mentioned in the paper (Cole, Bosomworth, Hayman, McLeod & Croce 2006). Also mentioned in this paper was a system called the Intelligent Train Monitor, still under development. However it claimed to extend the capabilities of the LEADER and TrainStar system by being able to predict the dynamics of the train fifty seconds in advance, using pre-recorded track data, GPS and control sequences.

2.3 Wagon construction

Coal Wagons are pretty basic machines which have not fundamentally changed since the days of the horse and cart, with the major innovation focused around speeding up unloading times and load carrying capacities.

Wagons have maintained three major components, with those being:

- Hopper/chassis
- Bogie and
- Braking system

Each major component has several sub-assemblies, which are discussed in more detail below.

As can been seen in Figure 2.2 it is not obvious looking at the apparatus which end is the front and back of the wagon. Therefore, industry loosely follows a standard naming convention which relies of the orientation of the brake cylinder piston and position of the hand brake. The piston points towards the handbrake wheel, which is nearly always fitted to the ‘B’ end of the wagon, although some exception do exist (NSW Rolling Stock 2013). Without surprise the other end is call the ‘A’ end.
2.3 Wagon construction

2.3.1 Hopper/chassis

The hopper is an open container which holds the coal loading in the dump station. Coal is loaded via the opening on top of the wagon while the consist slowly creeps forward, filling the wagon from front to rear. At the dump station coal is unloaded through doors on the bottom of the hopper.

This component has received the most attention over the years, with the initial designs requiring manual labour or mechanical means to remove the coal from the wagon. Also significant work has been done to improve the geometry of the hopper, with gains in volumetric and dumping efficiency.
2.3 Wagon construction

2.3.2 Bogie

Bogies and their sub-assemblies form the major mechanical item on a Coal Wagon, and therefore a focus area for this project. Modern bogies are a three piece design although others do exist. As can been see in Figure 2.4 there are obviously more than three components; however the side frame, bolster and wheel sets comprise the three items where is get its name.

The hopper is mounted onto the centre-bowl with a stub shaft for centering which allows angular movement between the bogie and hopper for cornering. Side bearers provided lateral support for the hopper which reduces centre-bowl wear.

The load springs mounted between the bolster and side-rails serve two important functions - suspension system similar to that found in cars, and a switching mechanism to control the brake pressure.

Each wheel set has two flanged steel wheels shrink fitted onto the axle. Rolling element bearings found fitted to the axle’s journals are encapsulated by a bearing adapter, typically called a cartridge. The rim of each wheel is tapered to prevent the wheel flange and rail from coming into contact and also damps out oscillations (Billingsley 1989). Wheel flat spots and bearing fatigue are big ticket maintenance concerns for rail freight operators, with flat spots causing significant secondary damage - literature suggests a reduction in the potential service life of 37 times (Sun, Cole & Bosomworth 2010a) if not detected and repaired.

Wheels have a nominal diameter ranging from 720mm to 940mm (depending on the wagon model) when new; however wear and machining out flat spots changes these dimensions. The rolling element bearings vary from model to model although they are most often cup and cone with opposing taper roller type bearings. For wheel and bearing specifications refer to Appendix F.

The brake beam can also be see in Figure 2.4; however this component forms part of the braking system which is covered in another section.
2.3 Wagon construction

Excluded from any of the figures is the draft gear. This component is a stub beam device with a coupler at the end to join wagon together to form a consist. The draft gear is connected to the bogie pin connection - refer to Appendix F.4 for a detailed drawing.

Figure 2.4: Annotated bogie model

2.3.3 Braking system

Nearly all wagons run the Westinghouse WF relaid freight brake system, which uses pneumatics to activate and apply braking force. The brake pipe provides the active low signal to the WF5 triple valve - currently being replaced by the ECP system. Charged air from the pressure vessel passes through the VTA change over valve which essentially is two state pressure control valve. The two state condition is ‘loaded’ and ‘unloaded’; unloaded the wagon only requires a fraction of the braking force, which is proportional to the brake pressure.

As mentioned in Section 2.3.2, the VTA valve is switched by bogie spring deflection when the wagon is loaded. The VTA valve is normally mounted on the side of the bolster and a striker plate aligned with the valve fixed to the side frame, so
when the springs compress the VTA piston changes state, passing higher pressure air to the brake cylinder.

The brake applicator, a shoe type one, is driven from the cylinder via a network of levers/linkages. The brake shoe is pressed against the rim of the wheel, creating frictional force opposing the rotation of the wheel. The brake shoe always remains in contact with the wheel due to the slack adjuster which compensates for shoe wear.

Figure 2.5: Air brake system

### 2.4 Embedded systems

There are many hardware options available which would be suitable for wagon monitoring applications. Products similar to National Instruments ComctRIO (National Instruments 2013) are specifically designed for this field, offering high level capability, modular configuration and quicker development times. However, high level instrumentation comes with a price tag to match, and therefore is not suitable for mass scale deployment or self funded student projects.
2.4 Embedded systems

In order to keep the cost of this project within budget, it was necessary to look elsewhere for hardware options which are not specifically designed for the industrial, scientific or academic sectors. The ‘makers community’ is an example of such a market which uses adaptable hardware with consumer level pricing. Therefore, the review has been limited to hardware available for consumer purchase and appropriate for use in a wagon monitoring and control project.

2.4.1 Micro-controllers

Micro-controllers are dedicated computers. Most devices which interact with the user are controlled by a micro-controller. They can be found in almost every device around the house and in the car.

Micro-controllers, as do most computers, follow basic building blocks: central processing unit (CPU) that executes a program, random access memory (RAM) where variables are stored, I/O pins for attaching other controllers, sensors or electro-mechanical devices and read only memory (ROM) where the program is stored. The main difference between the micro-controllers and the PC is computational power. It is not uncommon for a PC to have gigabytes of RAM and gigahertz of processor speeds. Typically micro-controllers on the other hand have a couple of kilobytes of RAM and have CPU’s that operate in the megahertz band. Although micro-controllers appear to be at a significant disadvantage compared to a PC, they operate on less power and are more robust making them perfect for industrial and remote applications.

Micro-controllers support digital I/O and many others support analog input/output via their analog to digital converter (ADC) and digital to analog converters (DAC) respectively. Also supported are the I²C and SPI protocols often used for communications between micro-controllers or sensors. A few offer CAN bus although it is primarily used in the automotive industry.

In recent years companies like Arduino, PIC (Microchip), Freescale Semiconductor and Beagleboard (Texas Instruments) to mention a few, have flooded the
market with consumer ready development boards which offer easy access to the controller’s I/O pins via headers soldered onto the PCB. Each company offers a range of boards to suit a variety of projects.

At the heart of an Arduino one can find a Atmel 8-bit AVR micro-controller though they have recently released a 32-bit ARM board which is the platform used in this project (Arduino 2013b). Both the ARV and ARM boards are programmed using the C/C++ programming language in Arduino’s own IDE and compiler.

PIC and Freescale Semiconductor development boards are similar to the Arduino but use a alternative layouts and programming languages. Their user community and support base is not as large as Arduino’s.

The Beagleboard platform offers the user access to PC level computing power, and runs a Linux operating system. Beagleboard’s can be programmed using a number of different languages and environments. However, their I/O pins only operate at 1.8V logic (Beagleboard.org 2013) making them hard to directly integrate with sensors. Currently there is only a small community of developers using this platform.

Expansion modules called ‘shields’ for Arduino or ‘capes’ for Beagleboard are readily available making it easier to adapt common technologies into microcontroller based projects. These module are available in the following extensions:

- GSM, Ethernet, WiFi, ZigBee, CAN bus RS 248 and 485 communications
- Servo, stepper, DC motor drivers and actuator control
- TFT LCD screen and video/audio
- Flash memory
- Relays and switching
- Battery and solar energy
2.4 Embedded systems

2.4.2 Micro-electro-mechanical systems elements

Micro-electro-mechanical system (MEMS) technology has been around for decades in airbag deployment and automotive pressure sensors although it has taken motion-sensing smart phones to showcase the capability of this technology. MEMS sensors can now be found in many industrial applications like robotic arms and GPS positioning systems.

The idea persists that inertial sensors are useful mainly when the end product has a need to detect acceleration and deceleration. This is true enough from a purely scientific view, yet it ignores many of the expanding uses of MEMS accelerometers and gyroscopes in areas that include medical devices, industrial equipment, consumer electronics, and automotive electronics.

Acceleration, vibration, shock, tilt and rotation are actually different manifestations of acceleration over different periods of time. Considering each mode separately helps to envision more possibilities.
2.4 Embedded systems

Acceleration measures the change in velocity in a unit of time. Velocity, expressed in meters per second ($m/s$), includes both the rate of displacement and direction of movement. It follows that acceleration is measured in meters per second per second ($m/s^2$).

If one considers acceleration over various periods of time then vibration can be thought of as acceleration and deceleration that happens quickly and in a periodic manner. Similarly, shock is acceleration that occurs instantaneously but, unlike vibration, a shock is a non-periodic function that typically happens once or intermittently. When an object is moved to alter its tilt, or inclination, some change in position with respect to gravity is involved. This movement tends to happen rather slowly compared to vibration and shock.

Because these first four modes of motion sensing each involve certain aspects of acceleration, they are measured by g force, the unit of force that gravity exerts on an object on the Earth i.e $9.81m/s^2 = 1g$. A MEMS accelerometer detects tilt by measuring the effect of the force of gravity on the different axes of the accelerometer. In the instance of a 3-axis accelerometer, three separate outputs measure acceleration along the X, Y, and Z axes of motion.

The accelerometers with the largest share of the market today use differential capacitors to measure g loading, which is then converted into volts or bits via a ADC and then passed to a microprocessor. Recent advances in technology have made it possible to manufacture tiny MEMS accelerometers in low-g and high-g sensing ranges with much wider bandwidth than previously, increasing the field of potential applications. A low-g sensing range is less than $20g/s$ and deals with motion a human can generate. High-g is useful for sensing motion related to machines and other industrial applications.

Thus far linear motion has only been considered, the type of motion that includes acceleration, vibration, shock, and tilt. Rotation is a measure of angular rate motion. This mode differs from the others because rotation may take place without any change in acceleration. For example; if a 3-axis inertial sensor with the sensor’s X and Y axes parallel to the Earth’s surface and the Z axis pointing
toward the centre of the Earth. In this position, the Z axis measures 1 g; the X and Y axes register 0 g. Now rotate the sensor to move only about the Z axis. The X and Y planes simply rotate, continuing to measure 0 g while the Z axis still measures 1 g. MEMS gyroscopes are used to sense this rotational motion and may be integrated in an inertial measurement unit (IMU) that embeds a multi-axis gyroscope, multi-axis accelerometer (O’Reilly & Weinberg 2010), and other non-MEMS sensing elements either for MEMES compensation purposes, additional function or both. Generally a IMU comes with an accelerometer and gyroscope at a minimum. The sensing elements can be of a 1, 2 or 3-axis transfiguration. MEMS also come in individual sensing element options; standalone accelerometer or gyroscope.

MEMS technology can achieve accuracies comparable to other much more expensive sensor types. An article comparing MEMS gyroscopes to the FOGD found the relative error to be in the order of 20 – 30%, caused by bias, scale factor and non-orthogonalises in the MEMS IMU. Although this may seem like a significant differential, one needs to consider cost of FOGS to be 10 times that of MEMS (Goodall, Carmichael & Scannell 2013). For high accuracy applications, MEMS sensors need to be calibrated for acceptable results. Moreover testing indicates MEMS performance to be dependant upon changes in temperature (Aggarwal, Syed, Niu & El-Sheimy 2008). Papers such as Aggaral et al. propose the need for “an accurate, reliable and efficient temperature model” to increase system performance. MEMS sensors nowadays can come with an inbuilt temperature sensor to compensate for environmental influences. The temperature and MEMS signal outputs are fed through the IC’s processor and filters before the acceleration signal is delivered to the appropriate read registers or analog pin. The temperature signal is generally available via a temperature register or analog pin on most IMU with temperature compensation.

MEMS sensors are available in two types; analog and digital. Analog versions output a voltage from proportional to acceleration for a set range. For example; a ±2 g accelerometer would output 0 V for −2 g and \( V_{CC} \) for +2 g. A few offer a selectable range and test pin to check calibration. Digital MEMS com-
municate to the master or slave via the I²C or SPI protocols and generally have more functionality compared to analog. While not being as user friendly from a programmers perspective, digital can offer pre process data, adjust calibration offsets, collect simultaneous samples, reduced noise, higher resolutions and better power management.

MEMS sense motion through the use of micro-fingers in close proximity to anchored electrodes. When the assembly is subjected to vibrations the fingers are deflected and the distance between them and the electrodes changes. This causes a change in the capacitance between the anchor and the electrodes according to the following equation:

\[ C = \varepsilon_0 \varepsilon_r \frac{A}{d} \]  

(2.1)

Where

- \( C \) capacitance (F)
- \( \varepsilon_0 \) permittivity of free space (F/m)
- \( \varepsilon_r \) dielectric constant of the insulator
- \( A \) area of each plane electrode (m/s²)

Change in capacitance is converted to a change in the output signal. “The fixed plates are driven by 180° out of phase square waves. Acceleration deflects the beam and unbalances the differential capacitor, resulting in an output square wave whose amplitude is proportional to acceleration” (Young 2006). This can be used to produce an analog signal which will continuously indicate the acceleration of the sensor. If there is no acceleration; that is the sensor is at a constant velocity, the signal will be constant. If, however, the sensor is subjected to continuous but possibly varying accelerations, as it would be when mounted on a vibrating object, the signal will be directly proportional to the actual acceleration. Figure 2.8 is a picture of a MEMS accelerometer showing the fixed electrodes and the deflecting fingers.
2.4 Embedded systems

2.4.3 Piezoelectric transducers

Transducer materials convert one form of energy into another, and are widely used in sensing applications. The tremendous growth in the use of microprocessors has propelled the demand for sensors in diverse applications. Today, Piezoelectric polymer sensors are among the fastest growing of the technologies within the sensor market.

Piezoelectric materials generate charge when squeezed. The amplitude and frequency of the signal is directly proportional to the mechanical deformation of the piezoelectric material. The resulting deformation causes a change in the surface charge density of the material so that a voltage appears between the electrode surfaces. When the force is reversed, the output voltage is of opposite polarity. A reciprocating force thus results in an alternating output voltage.

Piezo film, like all piezoelectric materials, is a dynamic material that develops an electrical charge proportional to a change in mechanical stress. Piezoelectric materials are not suitable for static measurements due to their internal resistance. The electrical charges developed by piezo film decay with a time constant that is determined by the dielectric constant and the internal resistance of the film, as well as the input impedance of the interface electronics to which the film is connected. Practically speaking, the lowest frequency measurable with piezo film is in the order of $0.001 \text{Hz}$ (MSI 1999). There are methods to achieve true DC response, but these require using the piezo film as both an actuator and sensor, monitoring change in the actuation resulting from the DC event.
The fundamental piezoelectric coefficients for charge or voltage predict, for small stress or strain levels, the charge density \((V/m^2)\) or voltage field \((V/m)\) developed by the piezo polymer.

Piezo film transducers offer wide dynamic range and are also broadband. These wide band characteristics (near DC to \(2GHz\)) and low charge are partly attributable to the polymer softness. As audio transmitters, a curved piezo film element, clamped at each end, vibrates in the length mode. Piezo film is a very high fidelity tweeter, also used in novelty speakers for toys, inflatables and apparel. Some configurations can also used for air ultrasound ranging applications up to frequencies of about \(50kHz\).

Many properties of piezo film change with excitation frequency and temperature. These properties are reversible and repeatable with either frequency or temperature cycling. Long term exposure to temperatures around \(70°C\) causes decay in the strain constant (Eorn & Trolier-McKinstry 2012).

The fidelity of a shielded piezo film sensor in musical instruments led to the development of vibration sensors for machines. In its simplest mode, piezo film vibration sensors behave essentially like dynamic strain gauges. The film does not require an external power source, yet typically generates signals greater than strain gauges after amplification. A typical piezo film sensor produces four orders of magnitude higher voltage signal than a foil-type strain gauges, and two orders higher than semiconductor types. The frequency response of the piezo film strain gauges is also superior.
The high sensitivity is due to the form of the piezo film material. The low thickness of the film results in a very small cross sectional area. Thus very small longitudinal forces create very large stresses within the material.

Piezo film sensors can be affixed to a vibrating surface and monitor the amplitude and frequency of the vibrating structure. The sensors can cover larger areas than normal strain gauges so any direct comparisons should be performed in uniform strain fields for meaningful results. Obviously, point type transducers may be used where required, although the low capacitance of the small sensor area will require additional consideration. Operation down to fractions of Hz can be achieved by either conventional charge amplifiers or, since signal levels are relatively high, simple high impedance FET buffer circuits.

A shielded piezo film sensors have been used to monitor bearings for wear and evidence of spalling. The sensors are permanently affixed to the outer surface of the bearing race with epoxy. The low mass and thin profile allow its use as a built-in non-destructive testing sensor, rather than the use of accelerometers for periodic condition monitoring.

### 2.4.4 Load cells, strain gauges and thermocouples

Load cells, strain gauges and thermocouple (sensing elements) measure force, strain and temperature respectively. These sensing elements are commonly set-up in a Wheatstone bridge to measure voltage signals proportional to the parameter of interest. Usually this requires an excitation signal and a few external components. But IC like the MAX1452 minimise the number of components simplifying circuitry (Parson & Budimir 2004).

A Wheatstone bridge is the name given to a voltage divider; sensing element change value and become unbalanced between each bridge arm. There are four (variable resistance) sensing elements in a full bridge although there are quarter and half bridge configurations (Matsuno et al. 1993). Issues can arise when bridge resistance changes with load, and current overrides built-in sensitivity networks.
2.4 Embedded systems

(Parson & Budimir 2004). A workaround is to use a current excitation drive or IC similar to the MAX1452.

Offset voltage can be affected by temperature changes reducing performance. Sensor manufactures recommend compensating for this by placing temperature sensitive resistance into one arm of the bridge. Another way to compensate for temperature drift is to amplify the output voltage either bridge node by passing the signal through a differential or instrument amplifier. The second option is independent of bridge balance and will not reduce the sensitivity (Matsuno et al. 1993). Resistance of the lead wires may cause imbalance of the bridge, depending on their length and excitation voltage.

DC voltage is normally used to excite the bridge, but AC can be used if the amplitude and frequency is kept constant. The uses of AC excitation helps overcome offset drift, noise, thermocouple effects and common-mode line noise from other electrical sources (Titus 2008). In essence, the bridge modulates the AC signal.

The voltage differential between the bridge node is normally small, requiring boosting by passing the signal through a op or instrument amplifier - same case for DC and AC excitation. For the DC case, output from the amplifier is fed to a a DSP or controller. AC excitation requires more bridge signal processing circuitry post amplification. The amp output and bridge excitation is wired into a synchronous demodulator converting the AC to a rectified signal before it is converted to DC by a low pass filter. The demodulator and low pass filter can be purchased in a single IC and therefore does not need to be built or done in software. Instead of amplifying the signal, high resolution ADC’s can give fine enough dynamic range to measure the smallest voltage variations. ADC’s with a
resolution between $12b$ and $16b$ are suitable.

Load cells (electronic types) and strain gauges operate in similar ways; a metallic foil element when loaded or strained deforms increasing or decreasing resistance. Each gauge measures only one direction in-line with its active length. There are multi-axis strain gauges, these are only multiple single axis gauges bonded to a common carrier and angled at $45^\circ$ or $90^\circ$ relative (Furman 2006). In order to collect precision strain data it is critical to use the correct gauge type, therefore there are many methods of gauge application and types to suit different stresses, surface temperatures and materials. Gauges should come with a calibration sheet stating the gauge factor, resistance and transverse sensitivity so the bridge voltage can be scaled for the unit of measure; micro-strain or Newtons.

Thermocouples are typically configured in a half bridge or a single voltage divider arm. Obviously temperature compensation is not required other than compensated resistor. A thermocouple is a two wire lead of dissimilar metals soldered together at the point of temperature measurement. Thermocouples produce a voltage when heated i.e. not a change in resistance like the force or strain gauges (ASTM 1990). Thermocouples come in a variety of types to suit a particular temperature range - K, E, J and N. K type are the general purpose thermocouples used in industry with a sensitivity of approximately $42\mu V/^\circ C$ and working range of -200$^\circ C$ to +1350$^\circ C$. Thermocouples do exhibit non-linearity above the Curie point which is approximatively 350$^\circ C$ for K type.

### 2.4.5 Proximity and displacement sensing

Proximity sensors detect the presence of an object with the need to be in physical contact. They are often used for counting, positioning, detection and inspection in control systems for automated plant (Fargo Controls 2013). A proximity sensor output is digital; DC or AC voltage depending on the sensor supply. The digital output is either PNP or NPN; when a object is detected the PNP sensor will pull high and NPN’s pull low.
Displacement sensors often use the same technology as proximity sensors and are sometimes called prox sensors; however, displacement sensors output a signal proportional to the distance an object is away from the probe - analog signal.

Inductive sensors are in common use and come in digital and analog outputs (Udelhoven 2009). As the name suggests, inductive sensors sense the metallics only and do this by generating a high frequency radio waves which generate a magnet field in front of the probe. When a metallic object passes through the field, a disturbance is detected by a change in reactance. A key advantage is they do not have any moving parts, as opposed to limit switches and are designed for harsh environments. Inductive sensor can have hysteresis < 0.05mm making them suitable for precision measurement; however, they have a relatively short range. A review of Allen-Bradley catalogue reveals the largest to be 30mm yields and up to a range of 40mm (Rockwell Automation 2010). Digital output types are relatively low cost devices ($100 each) due to their abundance of manufacture and simple design. Analog on the other hand are quite expensive ($2k each), because they are made to order and are calibrated for specific target material.

Unlike inductive prox probes, capacitive sensors work on all solids - metallic and non-metallic materials. There two types; contact and non-contact. Contact types are found in touch screens devices - not practicable for use in the project other than having a TFT interface with the unit, which is beyond the scope of work. Two metallic electrode plates are used as the sensing elements which form a capacitor in a feedback loop of a high frequency oscillator. An object approaching the probe will change the dielectric constant between the plates and raises the capacitance. Changes in capacitance varies the amplitude of the oscillation which in turn changes the output level detection circuit (Repas 2006).

Hall effect (magnetic) sensors detect the presence of magnetic fields, therefore the object needs to be magnetised by an appropriate means. This can be done in several ways; gluing a magnetised target onto the object, passing electric current through the object or hysteresis effects. Hall effect is typically used for presence/absence detection in speed, timing and positioning applications because of
their superior response time. The Hall effect is severely non-linear making them difficult to use for measuring linear displacement, their non-linear characteristics can be compensated for by electronics or software. Some Hall effect configurations can approach linearity namely the ‘two pole fixed gap method’ though difficult to apply and not without technical issues caused by end effects (Ramsden 2006). Hall effect sensors are available as IC’s which fall in four categories - bipolar, unipolar, latching and differential (Rakes 1999) used for digital sensing.

Infra red (IR) sensors, like those above, are another non-contact sensor. They have two main components; an emitter and a receiver which holds true for both digital and analog; however the receivers differ between the two. The analog receiver provides a voltage proportional to displacement of an object and covers a majority of the IR spectrum. Digital on the other hand, are tuned to a centering frequency which requires the IR emitter to be pulsed - 38kHz appears to be the nominal frequency between manufactures. Therefore, analog types do pick up interference from ambient IR light. IR beam spread is significant and needs to be focused by an aperture. The reflective ratio of the IR beam is dependant on the materials but mainly the colour of the target object, which in turn affects the sensitivity of the sensor. IR sensors are well suited for a range spanning between 400mm to 1500mm - depending of their focal length. Resolutions as fine as 5mm can be achieved; however the target object can be no closer than 50mm. Analog IR sensors are generally non-linear so correction curves are normally provided by the manufacturer. A significant disadvantage, like all optic sensors, is that they can detect matter suspended in the medium between the sensor and target material producing false readings.
Laser proximity sensors are another type of optical proximity/displacement sensor; however are too costly to be considered for this project.

Similar to the IR sensor, ultrasonic range finders have an emitter and receiver; piezoelectric transducers can be both an emitter and receiver, therefore only need one sensing element. As their name implies, ultrasonic sound waves are used instead of IR light. Comparing costs to IR sensor, ultrasonic range finders are only slightly more expensive. Instead of using the strength of the emitted signal (like IR sensors do), ultrasonics range finders use a time of flight method (TOF). An ultrasonic wave is propagated from the emitter at a fixed frequency. Timers are used to measure the period elapsed from the pulse to receiving the echo reflected from an object. There are three primary variables effecting the usability of ultrasonic range finders; decibel level of the emitted sound, impedance properties of the target object and transfer media since the speed of sound is unique for each material. The analog type are available in two types of output; proportional voltage and Pulse Width Modulation (PWM) - a variable frequency square wave signal. Digital outputs are available in NPN and PNP.

Linear displacement resistive sensor, otherwise known as ‘pots’ use a sliding contact against a fixed resistive element to create change in resistance. Wired up to a DC voltage divider, they produce a proportional voltage for high impedance loads. Pots are easy to use and require little electronic support. However, because pots are a electromechanical sensor, they suffer from large hysteresis, repeatability issues and tend to deteriorate over time/repeated use caused by wear, especially when exposed to vibrations (Schaevitz 2004).

2.4.6 Global positioning system modules

Global positioning system (GPS) is a pace-based radio-navigation system that broadcasts highly accurate navigation pulses to users on or near the Earth. A GPS module measures the precise time (< 100ns) it takes radio waves to travel from at least four satellites to the module’s location. Time is then converted into
distance for each satellite and from this longitude, latitude and elevation can be
determined.

The principle behind GPS is the process called triangulation. The time taken for
the signal to reach Earth, generally $< 0.1 s$, is multiplied by $300\, Mm/s$ (velocity
of a radio wave) to determine the distance (Logsdon 2013). This puts the GPS
module at a circumference on a sphere relative to the satellite. The same process
is used for the remaining three satellites required for the fix. Where all four
spheres intersect the location is positioned. The in-built computer converts this
data into GPS coordinates. Although mathematically three satellites are only
required, the fourth one is used to correct clock errors. In addition, GPS modules
calculate velocity using the Doppler effect shifts created for the motion of the four
satellites.

Civilian GPS has an horizontal accuracy of 10$m$ and vertical of 20$m$. Velocity
can be determined to an accuracy of about 1$m/s$. The military GPS service
has a accuracy within 3$m$ (Logsdon) (Morgan-Owen & Johnston 1995). The
civilian GPS accuracy can be significantly improved by differential techniques. In
short, differential GPS uses a fixed GPS module with known coordinates, acquires
the GPS signal and subtracts the two which yields the correcting factor. The
correction factor is sent to other GPS devices to improve precision positioning.
Accuracies approaching the military system can be achieved using this technique.

Most GPS modules communicate with the micro-controller or computer by stream-
ing data to the master. Manufactures often have their own propitiatory formats;
however it appears most follow the standards published by the National Marine
Electronics Association (NMEA). The NMEA 0183 standard uses ASCII serial
communications to stream data to the listener in the string class otherwise know
as a character array.

NMEA 0183 supports bidirectional communications between devices; however
GPS messages are the only data type of interest for this project though is does
support several other identifiers like RADAR, sound scanning, time keeping and
weather instruments to mention a few. In order for messages to comply with
the NMEA 0183 standard, the following set of rules apply (National Marine Electronics Association 2001):

- Start with the dollar symbol ($)
- Next five characters identify the talker (first two) and message type (last three)
- Each field is separated by a comma (,)
- If no data is available for a particular field a blank space is created
- An asterisk (*) precedes checksum if used
- Following the asterisk is a two digit hexadecimal number which represent the number of bits between the dollar and asterisk symbols
- A carriage return end the message

The talker identifier for GPS is GP - next two characters following the dollar symbol. There are numerous message types; however data contained in RMC suits most applications. The RMC sentence structure appears as follows:
2.5 Data processing

Listing 2.1: RMC format

$GPRMC, hhmmss , status , latitude ,N, longitude ,E, spd , cog , ddmmyy 
, mv, mvE, mode*cs<CR><LF>

Listing 2.2: Example refer to Table 2.1

$GPRMC, 083559.00, A, 4717.11437, N, 00833.91522, E 
, 0.004, 77.52, 091202 , , , A*57

2.5 Data processing

Processing techniques are required in order to manipulate data into meaningful information. Some digital sensors can output data that would normally need processing in a controller or computer; however most sensor provide analog or digitised analog information. For analog data, at the very least, values read from the sensor need to be converted to the relevant unit. Data provided by a analog temperature probe is a good example - data read for the sensor is matched against the nearest ADC division and scaled according to the sensor sensitivity. Nonconformity, data read from most sensors requires more processing than that mentioned above. Techniques for processing such data is discussed below.

2.5.1 Pulse timing

In order to measure angular velocity, tachometer is required. A proximity sensor is used in conjunction with a one or several targets. The proximity sensor is fixed in position relative to the targets, which are attached to rotating object.

When using a one marker tachometer, it is a case of timing the period between
2.5 Data processing

two pulses - refer to Equation 2.2.

\[ f = \frac{1}{T} \]  \hspace{1cm} (2.2)

Where
\( f \) cycles rate \((Hz)\)
\( T \) period \((s)\)

One marker tachometers are normally used for high speed steady state machines.
however for applications that exhibit transient behaviour or need finer resolution
multi-targeted tachometers are better suited.

Data captured from multi-targeted tachometer can be processed in two ways.
The first method requires a counter to be implemented. Once the counter value
equals the number of targets, the time between the first and last pulse is used
to determine the frequency of revolution. The other approach is to measure the
period between two pulses and multiply the frequency by the number of targets.
The second approach has greater accuracy; however requires more processing
power.

Angular acceleration can be computed once the speed data is obtained using the
finite difference method - refer to Equation 2.3.

\[ f'(x) = \lim_{h \to 0} \frac{f(x + h) - f(x)}{h} \]  \hspace{1cm} (2.3)

2.5.2 Parsing

Output data from most GPS modules are character arrays and already in a read-
able format, without any scaling or processing required. However, GPS NMEA
strings contain a lot of information demanding temporary memory resources of
the host.
Instead of storing all the input data in a buffer, state machines can parse the data while the stream is arriving. In order to implement a state machine, one needs to know the data sequence prior to the host receiving it.

A state machine is a system of states. The fist state is special - the initial state. One or more other states are the finial states. If an event causes one state to change to another state then a state machine exits (Thomas & Hunt 2002). States are connected by transitions which are named by their input. When an event occurs, we follow a transition from the current to the new state.

State machines are useful when a program has one input event and multiple events are dependant on that event. From a programming perspective, a variable is needed to store the current state and is updated as events occur. Typically event handling is done with the case statement.

### 2.5.3 Transient signals

Vibration data requires multiple processes for analysis. Unlike most parameters, reading a value from an accelerometer is meaningless. Vibration and sound data requires many time stamped values to be stored in a buffer. The size of the buffer is determined from the number of samples collected, which in turn is a function of the sample rate and time.

Nyquist sampling theorem is fundamental to all digital signal analysis. The sampling theorem says that an analog signal $x(t)$ can be uniquely represented by discrete samples, if, and only if, the sampling frequency is twice that of the desired bandwidth (Billingsley 1989). If this can be achieved, then the analog signal can be reconstructed using Equation 2.4 (Anders 2012). For practical applications the sample rate needs to be $> 2$ times the highest frequency ($f_{max}$); $2.56 f_{max}$ has been adopted as the standard in industry because of its relationship
2.5 Data processing

with binomial mathematics.

\[ x(t) = \sum_{n=-\infty}^{\infty} x(n) \sin c f_s(t - n\Delta t) \quad \text{and} \quad \sin c(x) = \frac{\sin(\pi x)}{\pi x} \quad (2.4) \]

Where \( f_s > 2(f_2 - f_1) \) discrete sampling frequency

If one does not consider the sampling theorem, aliasing or frequency fold can occur (Billingsley 2010). Put simply, aliasing is the under-sampling of an analog signal so that appears in the digitised signal at a lower frequency. An example of aliasing can be seen in Figure 2.13. If \( f_{\text{max}} \) is unknown or the sample rate of the hardware is too slow to reach \( f_{\text{max}} \) then low pass filters can be used to remove any high frequency signals.

![Figure 2.13: Example of aliasing (Billingsley 2010)](image)

Another problem of under-sampling is that of resolution. Depending on the transient characteristics of the signal, oversampling at 2.56 may not be enough to accurately represent the time domain signal. Comparing the two wave forms in Figure 2.14, it is evident that the waveform oversampled at 2.56\( f_{\text{max}} \) appears to jump between sample points while 20\( f_{\text{max}} \) signal is a good representation. Despite the 2.56 oversampling representation appearing inaccurate, it still contains all information about the signal i.e. adequate enough to do a frequency domain analysis.
2.5 Data processing

Signal filters play a large role in signal analysis and are often described by their frequency response, called the filter characteristics. There are three primary filter types; low pass, high pass and band pass. As their names imply, low pass filter remove frequencies higher than their cut off; high pass remove frequencies lower than their cut off and band pass is a low and high pass filter combined. Whilst these and many more filters are used in processing vibration signals, the first-order is the most common and often used in vibration sensors and signal conditioning units (Anders 2012) - also can remove DC offsets.

Filtering can be done before or after digitising a analog signal. If done before, the filtering achieved using hardware before the ADC - a resistor capacitor circuit. Otherwise discrete data filtering is done in software. The same principles apply to either hardware of software filtering. Not yet mentioned, filters have a cut off frequency and is typically denoted by $f_c$ in most texts. This is the frequency where the filter starts to take effect if using a high pass or end if using a low pass. Equations 2.6 & 2.7 can be used for software filtering.

\[ f_c = \frac{1}{2\pi \tau} \quad \text{and} \quad \alpha = \frac{\tau}{\tau + \Delta t} \]  

(2.5)

\[ y_i = \alpha x_i + (1 - \alpha)y_{i-1} \]  

(2.6)
2.5 Data processing

\[ y_i = \alpha y_{i-1} + \alpha (x_i - x_{i-1}) \]  

(2.7)

Where \( \tau \) time constant (t)

Once the signal is filtered and scaled, it is often convenient to integrate or differentiate the signal. When the signal is from an accelerometer, the only option is integration. There are several methods of integration although some are better than others. Anders (2012) suggests the difference equation (Equation 2.8) is accurate, which is simply the trapezoidal rule. Equation 2.3 can be applied if a differentiated signal is wanted.

\[ y_n = y_{n-1} + \frac{\Delta t}{2} x_n + \frac{\Delta t}{2} x_{n-1} \]  

(2.8)

Since dynamics signals cannot be summarised by a single value, often it is useful to have a measure for comparison. The most common measure used is the root mean squared or RMS value. The RMS value of a dynamic signal \( x(n) \) can be defined by

\[ x_{RMS} = \sqrt{\frac{1}{n} \sum_{n=1}^{n} x^2(n)} \]  

(2.9)

Converting a dynamic signal from the time domain to the frequency domain is an essential tool for vibration analysis. Fourier series states that every periodic signal can be represented by a combination of sinusoidal waveforms, diagrammatically shown in Figure 2.15. The Discrete Fourier Transform (DFT) is the method of converting signals into spectra - frequency domain. It is done so by passing a signal array \( x(n) = x(n\Delta t) \) of size \( N \), where \( N \) is normally an integer power of two; that is \( N = 2^p \), where \( p \) is an integer, through the DFT shown in Equation 2.10. The DFT then produces an array \( X(k) = X(k\Delta f) \), a block of frequency data. There are also analog Fourier transforms although seldom used because nearly all
2.5 Data processing

device are now digital. The inverse transform can be also calculated; that is, a block of frequency data can be converted back into a time domain signal though; that is of little use for this project.

![Figure 2.15: Fourier series principal (Anders 2012)](image)

Some important points of the DFT:

- Computed from a discrete number of samples
- Scaled in the same unit as the signal block
- Calculated in a non-symmetrical way, from \( n = 0 \) to \( n = N - 1 \)

While the DFT is a very useful tool, a lot of the computations are not necessary. By observing the \( e^{-j2\pi kn/N} \) term, it becomes evident that this process can be made faster. The Fast Fourier Transform (FFT) is that solution. As the name implies, it is just a faster version of the DFT - about one hundred times faster (Chernenko 2007). The main speed-up happens by noting \( e^{-j2\pi kn/N} \) yields only two values; one number for all odd values of the exponent \((n)\) and another for the even.

\[
X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N} \quad (2.10)
\]

The FFT cannot be passed through one equation and yield a result like the DFT; its sequential series of processes rather than a single equation. A FFT algorithm must follow these steps (Chernenko 2007):
2.5 Data processing

1. Sort the time signal into convenient order to for summation

2. For every summation level:

   (a) For every exponent factor $\pi$

      i. Compute factor

      ii. For very sum of this factor:

         A. Compute the product of the factor and the second term of the sum

         B. Compute sum

         C. Compute difference

The workings of each step in the FFT process is shown in Section 4.3.6.

![Example FFT](image)

Figure 2.16: Example FFT

(Courtois 2011)

More likely than not, real data does not start and finish at zero. Passing real (filtered or unfiltered) data through the FFT at this stage would produce spectra with poor resolution. The spectra would suffer from leakage caused by edge effects of the time domain signal. In order to minimise leakage, time signals need windowing. There are several windowing functions, each having their pros and
2.5 Data processing

cons; however the Hann windowing function (Equation 2.11) offers the best all round performance (ISO 2008). Windowing does cause other issues like increased number of frequency bins and a higher first lobe, although, these compromises are considered acceptable for increased spectral resolution (Courtois 2011).

Windowing functions are independent of the nature of the time domain signal. Therefore, window’s only needs to be calculated once assuming the number of samples in the signal block is constant - an inspection of Equation 2.11 shows this to be true.

\[
w(n) = 0.5 \left(1 - \cos\left(2\pi \frac{n}{N}\right)\right), \quad 0 \leq n \leq N
\]  

(2.11)

2.5.4 Cycle counting

There are several method for performing fatigue analysis to predict the life of a component including stress-life, strain-life, crack propagation and spot weld methods (NI 2013). Stress-life is widely used for design and shares close similarities with the strain-life method (ANSYS 2013).

Strain gauge data can be processed to produce a load spectrum which is a
very useful tool for life estimation of a component. ASTM E1049-85 (ASTM 2011) standard lists the procedures for cycles counting; ranging the simple Level-Crossing method to the most complex Rainfow methods. These methods determine load cycles by

- Counting the number of times the signal amplitude crosses a set level or passes within a range or
- Counting the peaks and valleys above a reference load or
- Comparing the load range in consideration to the previous range and weighting the event count based on severity of change

### 2.5.5 Alarming

Setting limits to measured variables is a useful machine monitoring aid. Large plants can have hundreds of the individual machines and even more process or machines health monitoring parameters being sampled. Therefore, it is not efficient to have personnel monitoring all this data as it comes in, especially considering most of the time the data is not indicating anything meaningful, other than machinery is operating normally. This is where computer generated limits or alarms come into their own, as they can send an alert to the control system or operator to take action.

There are several alarms to suit different variable types. Overall alarm limits (Figure 2.18(a)) are used for monitoring variables which can be represented by a single value, otherwise known as trend data i.e. RMS vibration, temperature, velocity, static strain and pressure. These alarm types can also be used to monitor maxima and minima of transient variable blocks (Figure ??) when alarming from frequency domain data is not suitable.

Banding and envelopes are other ways to set up alarms, but only used for frequency domain (blocked) data i.e. transient pressure, vibration spectra and supply network monitoring to mention a few. Bands (Figure 2.18(c)) are similar
to overall alarms; however there are typically several per variable block broadly monitoring internals of the grouped data sets - refer to Figure 2.18(c). Enveloping as shown by the red trace in Figure ??, is the most advanced method of the three; alarm limits are set for every frequency contained in the variable block.

In recent years, computer algorithms have been designed to learn the normal characteristics of the monitored machine or process, and can detect small changes before the condition deteriorates (Jack & Nandi 2002) causing secondary damage or triggers an alarm based trip. These technologies are still relatively new and not available in most supervisor control and data acquisition systems (SCADA) systems.

Alarms can be set statistically, by manufacturer’s recommendations, documented standards or experience. Manufacturer tend to be too conservative and standards generally class machinery in too broad a range creating uncertainty with the selected level(s) (Jacobs 2008). Setting alarms statistically or by experience either requires historical data or knowledge of the same/similar machine or process and operating conditions.

Generally alarms are rarely singular...often there are a system of alarms like level 1, level 2, level 3, or some use word based system e.g. warning, alert, alarm, fault, upper fault etc. Limits are set to a predefined number of standard deviations from the mean, based on the same theories of statistical quality control. For a three level alarm system the first alarm is typical set to one standard deviation; second level, two deviations and third level, three deviations; however this is only a rule of thumb and should be tuned to achieve reliability targets.

2.5.6 Linear quadratic estimation

The linear quadratic estimation algorithm better know as the Kalman filter is a method for statistically filtering noise and other inaccuracies. The algorithm works by a two step process; time update and measurement update.
The Karlman filter estimates the current state variables along with the uncertainties. Once the next sample of raw data (with noise and other uncertainties) is known, the previous estimate is updated with a weighted average, with a higher weight being given to a higher uncertainty.

One benefit of the Karlman filter is that it can run in real time because the recursive nature of the algorithm - only needs the past state and current measurement. However, the theory behind the algorithm assumes a normal distribution of noise (Welch & Bishop 2006) which could lead to error if the data being sampled does have this characteristic.

From a mathematical perspective the Karlman filter addresses the general prob-
2.5 Data processing

The problem of trying to estimate the state \( x \in \mathbb{R}^n \) of a discrete-time controlled process that is governed by the linear stochastic difference equation (Equation 2.12) as described by Welch & Bishop (2006).

\[
x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}
\]

(2.12)

with a measurement \( z \in \mathbb{R}^m \) that is

\[
z_k = Hx_k + v_k
\]

(2.13)

Where

- \( w_{k-1} \) process noise
- \( v_k \) measurement noise
- \( A \) \( n \times n \) matrix relating the previous time step \( k - 1 \) to the current \( k \)
- \( B \) \( n \times 1 \) vector relates the optional control input \( u \in \mathbb{R}^1 \) to state \( x \)
- \( H \) \( m \times n \) matrix relates the state \( x \) to the measurement \( v_k \)

While Equations 2.12 & 2.13 are the grounding formula used to describe the Kalman filter, it is not clear how they can be applied in a program - a set of discrete equations are needed. As mentioned earlier, the Kalman filter is a two set process which implies there are two groups of equations - one for prediction and the other for correction.

\[
\hat{x}_k^{-} = \hat{A}x_{k-1} + B_{k-1}
\]

(2.14)

\[
P_k^{-} = AP_{k-1}A^T + Q
\]

(2.15)

\[
K_k = P_k^{-}H^T \left( HP_k^{-}H^T + R \right)^{-1}
\]

(2.16)
\[ \hat{x}_k = \hat{x}_k^- + K_k \left( z_k - H\hat{x}_k^- \right) \quad (2.17) \]

\[ P_k = (I - K_k H P_k^-) \quad (2.18) \]

Where

- Equations 2.14 & 2.15 time updates
- Equations 2.16, 2.17 & 2.18 measurement updates
- \( x_k \) & \( x_k^- \) prior and post estimate state
- \( P_k \) & \( P_k^- \) prior and post estimate error covariance matrix respectively
- \( R \) prior and post estimate measurement error covariance matrix
- \( K_k \) Kalman filter gain matrix
- \( H \) measurement prediction matrix
- \( I \) identity matrix

Equations 2.14 to 2.18 can be scripted in a loop to compute the smoothed data. Depending on the application, many terms can be eliminated or represented by a single value reducing the complexity when implementing the code. In order to initiate the filter one value for \( x_k \) is needed, normally the initial sample, and assumed initial conditions for \( Q, P_{k-1} \) and \( R \). Optimum performance from the filter is achieved when the initial assumption for \( Q \) is slightly > 1; however it is acceptable to set to zero. \( P_0 \neq 0 \) otherwise the filter will believe the initial sample; \( x_k \) remains constant - any value > 0 will eventually converge. The square of the measurement variance \( R \) is selected by analysing the noisy signal.

### 2.6 Digital communications

Devices and their peripherals need to communicate in order to transfer data. Similar to languages, there are a variety of different types of communication protocols between hardware platforms. It is important to know the details of
2.6 Digital communications

each to make good use of the applicable method. Therefore, a brief review of
digital communication standards is needed.

In order to inform train drivers or control operators of issues, data sampled from
the unit must be able to be sent to a terminal for analysis. Therefore, a review
has been conducted into common networking technologies.

Since it would be ideal to have multiple wagons communicating to the terminal,
those methods which only support paired network topology have been excluded
from the review.

2.6.1 Universal asynchronous reviver/transmitter

A universal asynchronous reviver/transmitter (UART) is an IC that translates
data between the parallel and serial forms, and often used in conjunction with
RS-232, RS-422 and RS-485 standards.

UART transmits eleven bits of data at a time; a 1\text{b} (bit) start and 2\text{b} stop. The
other 8\text{b} form a 1\text{B} (byte) of data containing the information. The start bit is a
logic low which signals to the receiver that a new piece of information is going to
arrive. The stop bits are logic high, meaning that there is always going to be a
level shift in a transfer.

Data speeds are selectable, depending on the maximum data transfer rate sup-
ported by the slowest device, which is related to the UART IC clock frequency,
and must match. On each clock cycle, data placed in shift registers is transferred
which may trigger an interrupt in the host device, depending on the size of the
first-in first-out buffer, to make sure data is not over written before it is sent. The
receiving device sends a flag to the transmitter to notify it has finished processing
the last bytes and is ready for more.
2.6 Digital communications

2.6.2 Inter-integrated circuit

The Inter-integrated circuit (I\textsuperscript{2}C) protocol is a single-end bus for attaching low speed peripherals to devices. I\textsuperscript{2}C is a two wire system; has a serial data (SDA) and serial clock (SCL) lines between the master and slave device. There is not a hard set standard for the logic levels though +5\textit{V} or +3.3\textit{V} are the most common and is active low.

![Figure 2.19: I\textsuperscript{2}C bus (Robot Electronics 2013)]

I\textsuperscript{2}C has two references for for the address size; 7\textit{b} or 10\textit{b}. Bus speed’s are 400\textit{kb/s} in normal mode (Byte Paradigm 2013) and 10\textit{kb} in low-speed mode. The number of slave devices is limited only to the address space and bus capacitance which should be restricted to 400\textit{pF}, making communication limited to a couple of meters.

2.6.3 Serial peripheral interface bus

Serial peripheral interface bus is a four wire system; clock (CLK), master-out slave-in (MOSI), master-in slave-out (MISO) and a chip select (CS) lines - refer to Figure 2.20. SPI is a single master system and can support as many slaves as there are CS pins. The SPI standard or lack thereof, is rather loose. Clock frequencies can be in the range of 10\textit{kHz} to 100\textit{MHz} (Byte Paradigm 2013). CS can be logic high or low depending on the slave and data is not limited to a set number of bits per transmission.

SPI has full duplex communication which allows for higher data rates compared to I\textsuperscript{2}C with more flexibility. However it requires more pins, supports only one master, can be prone to noise spike causing communication errors and can only
2.6 Digital communications

handle very short distance between devices.

![Diagram of SPI layout](Image)

(a) Single slave  (b) Multiple slaves

Figure 2.20: SPI layout (Byte Paradigm 2013)

2.6.4 Controller area network bus

The controller area network bus or CAN bus for short, is a robust serial bus which can communicate with a throughput of up to $1\text{Mb/s}$ (CAN C) (CAN Bus n.d.). CAN bus is similar is to $I^2C$; that is, it only requires two wires (minimum of one wire) and uses an addressing system to communicate. For that reason it was introduced to the automotive industry to reduce wire looms.

There are four types of CAN bus; CAN A, B and C, from the lowest throughput to the highest respectively. The higher throughput the more likely it will suffer from network error for noise. The four different types can be used on the same network assuming they meet at a central point called a gateway, which is basically a transistor (Meiser 2009). CAN is a a multi master network which provides peer to peer communication though only one message can be sent at a time; the one with the highest priority will go first, determined by the header codes (Warner 2003).
2.6 Digital communications

2.6.5 Modbus

Modbus is another serial communication protocol widely used for communications between programmable logic controllers (PLC’s) and SCADA systems. Modbus was specifically designed for industrial application making it robust. It is open source meaning no licensing fees or vendor lock in.

Device numbers are limited to 247 devices per bus, each having a unique address. Depending on the hardware infrastructure, Modbus can be a master slave or peer to peer configuration. Serial only supports master slave while Ethernet can do peer to peer communications.

There are four Modbus frames works; RTU, ASCII and TCP (open mbus). The RTU and ASCII format share close similarities with their functions and both support error checking. TCP on the other hand is completely different, has nil error checking data and can be prone to unauthorised commands and data interception.

2.6.6 Internet suite

The World Wide Web uses Hypertext Transfer Protocol (HTTP) as the foundation for communications applies for cellular and cables networks. Like those protocols mentioned above, HTTP adopts the master slave configuration but assigns them different names; client and server respectively. A device can wear both hats; that is it can act as a server and client, but only one at a time.

HTTP requests are sent to sever by the client and the server responds with a message. The response contains status information indicating 200 OK for a successful connection followed by the requested data or an error code which indicates the possible cause..

HTTP is only an application of lay protocol and requires a transportation layer like TCP or UDP. The option will be dependant on the desired transfer speed and
error checking functionality. Instead of sending bytes at a time, these transport layers packet the data up and sends it all at once.

The HTTP/1.0 specification defines three request methods; GET, POST and PUT (W3C 2013). After a server has received the request, it also expects some headers containing information specific to the API required for the site. A typical request may look similar to the example below:

Listing 2.3: Example of a HTTP PUT request

```
PUT /v2/feeds/1951066047.json HTTP/1.0
Host: api.pachube.com
X-APIKey: OqeB68UHfHYClShV4tuGHHqBMWcN0KrTz1SND7JRpmuDWhP
User-Agent: Arduino/1.0.5
Content-Length: 178
```

PUT command example notifies the server to expect data from the client. The string of characters states the API version and site credentials. Next the data format, in the example JSON, although there are others formats available like XML and CSV.

The header host, user-agent and content-length are the minimum content required by HTTP/1.0 specification in order to have a successful connection.

As mentioned earlier, data encapsulated in a post can be formatted in three different ways - hybrids of these three also exist but are not as widely used. JSON is the newest format orientated around JavaScript notation which is easily readable by both computer and humans alike. XML like JSON supports all the common data types (numbers, strings, boolean, arrays, objects, null); however for a human perspective it is much harder to follow, involving significantly more operators to comply with the syntax. CSV is the easiest of the three to use and read; however it only supports the exchange of strings and number data types therefore making it limited to basic information.
Listings 2.4, 2.5 & 2.6 are examples of the same data packet formatted in JSON, XML and CSV.

Listing 2.4: JSON

```json
{
    "version": "1.0.0",
    "datastreams": [{
        "id": "example",
        "current_value": "333"
    },
    {
        "id": "key",
        "current_value": "value"
    },
    {
        "id": "datastream",
        "current_value": "1337"
    }
}
```

Listing 2.5: XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<eeml>
    <environment>
        <data id="example">
            <current_value>333</current_value>
        </data>
        <data id="key">
            <current_value>value</current_value>
        </data>
    </environment>
</eeml>
```
2.6 Digital communications

Listing 2.6: CSV

element, 333
key, value
datastream, 1337
<key>, <value>

2.6.7 Ethernet and WiFi

Ethernet and WiFi are the common computer networking technologies for local area networks (LAN’s), standardised by IEEE 802.3 and 802.11 standards respectively. Ethernet has a throughput of around 100Gb/s and WiFi11Mb/s - slow throughput is to be expected of a wireless transmission method.

Both Ethernet and WiFi divide data stream into smaller pieces called either packets or frames. Each frame contains a preamble, Ethernet header with source and mac address, payload data with header for other protocols and finishes with a 32b check sum.

Each device on the network has a MAC address which is unique and generally static unless explicitly changed by the user. If multiple networks are interconnected, an additional layer called the Internet Protocol (IP) is required on top of the standard Ethernet layer and yet another address - IP address (Contemporary Control Systems 2011). The IP address is not unique to each device unless an address has been designated for a particular device on the network. Normally each time a device connects to a network an IP address is assigned by the switch/router.
2.6 Digital communications

although it does depend of the network configuration - differences between static and Dynamic Host Configuration Protocol (DHCP).

There are two types of IP address; local/private and internet-side/public. As mentioned earlier, the switch/router assigns the IP address; that is, a local IP address. The internet-side IP is set by the service provider and normally does not change unless the connection is refreshed.

The maximum range between repeater for Ethernet is about 100m and about 80m between WiFi extenders. These ranges vary between cable type for Ethernet or broadcasting band (2.4\,GHz or 5\,GHz) and signal strength of the router for WiFi. But with most networks, the overall range is unlimited, assuming one uses enough repeater or WiFi extenders.

Ethernet does not require security in the normal computer sense because an attacker would need a physical connection to the network - assuming a Secure Socket Layer (SSL) is used for all internet communications. WiFi on the other hand can be hacked remotely. A standardise encryption (WPA2) has been adopted among the router manufactures to increase the security of the wireless signal.

2.6.8 ZigBee

ZigBee is a radio networking protocol built on top of the IEEE 802.15.4 layer - a standard that defines power management, addressing, error correction, message formats, and other point to point specifics. The ZigBee layer adds three more things (Faludi 2011):

1. **Routing**: so messages can be passed from through from radio to radio

2. **Ad-hoc network creation**: an automatic method for creating network on the run

3. **Self-healing mesh**: another automatic process that can detect missing radio node and reconfigure the network to repair any broken routes
ZigBee supports pair, star, mesh and cluster tree topologies if setup with a coordinator, router and/or end device radios. Each radio has a three level addressing scheme; address, PAN ID and channel. The address in itself is broken down into three parts; 64\text{b} address unique to that radio, 16\text{b} address assigned by the coordinator unique on that network and a optional node identifier so the radio can be given a string name.

Each radio is configured and programmed via a serial terminal using AT commands which support two different modes; transparent and command modes. In transparent mode (default), messages are passed through the radio similar to a router and in command mode the communications directed to the radio.

Security can be included to the network via a 128\text{b} encryption key. There are two types of these keys; network and link keys. Network keys encrypt data before sending it to the next radio node where it is decrypted and encrypted again before sending it off. This process is cycled from node to node until the message reaches the destination radio. This leaves the door open for the packet to be compromised at the nodes in between the sender and receiver. Link keys do not decrypt the data at the nodes in between making for a more secure network.

XBee, along with other ZigBee radio manufacturers, have several wireless RF module; however most are not strictly ZigBee i.e. some are 802.11n, 802.15.4,
proprietary protocol or Digimesh. The ZigBee transmits on the 2.4GHz band at a rate of 250kb/s. The primary differentiator between ZigBee radios is transmission range. They come in 90m to 3.2km ranges - line of sight. The power consumed operate can be scaled to the inverse square law. Hence, longer the distance, the more power is required. However if correctly meshed their range becomes endless assuming enough node radio are chained together.
Table 2.1: RMC reference

<table>
<thead>
<tr>
<th>Field No.</th>
<th>Example</th>
<th>Format</th>
<th>Name</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$GPRMC</td>
<td>string</td>
<td>$GPRMC</td>
<td></td>
<td>Message ID, RMC protocol header</td>
</tr>
<tr>
<td>1</td>
<td>083559.00</td>
<td>hhmmss.sss</td>
<td>hhmmss.ss</td>
<td></td>
<td>UTC Time, Time of position fix</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>character</td>
<td>Status</td>
<td></td>
<td>Status, V = Navigation receiver warning, A = Data valid</td>
</tr>
<tr>
<td>3</td>
<td>4717.11437</td>
<td>ddmn.mmmm</td>
<td>Latitude</td>
<td></td>
<td>Latitude, Degrees + minutes</td>
</tr>
<tr>
<td>4</td>
<td>N</td>
<td>character</td>
<td>N</td>
<td></td>
<td>N/S Indicator, hemisphere N=north or S=south</td>
</tr>
<tr>
<td>5</td>
<td>00833.91522</td>
<td>dddmm.mmmm</td>
<td>Longitude</td>
<td></td>
<td>Longitude, Degrees + minutes</td>
</tr>
<tr>
<td>6</td>
<td>E</td>
<td>character</td>
<td>E</td>
<td></td>
<td>E/W indicator, E=east or W=west</td>
</tr>
<tr>
<td>7</td>
<td>0.004</td>
<td>numeric</td>
<td>Spd</td>
<td>knots</td>
<td>Speed over ground</td>
</tr>
<tr>
<td>8</td>
<td>77.52</td>
<td>numeric</td>
<td>Cog</td>
<td>degrees</td>
<td>Course over ground</td>
</tr>
<tr>
<td>9</td>
<td>091202</td>
<td>ddmmyy</td>
<td>date</td>
<td></td>
<td>Date in day, month, year format</td>
</tr>
<tr>
<td>10</td>
<td>-</td>
<td>numeric</td>
<td>mv</td>
<td>degrees</td>
<td>Magnetic variation value, not being output by receiver</td>
</tr>
<tr>
<td>11</td>
<td>-</td>
<td>character</td>
<td>mvE</td>
<td></td>
<td>Magnetic variation E/W indicator, not being output by receiver</td>
</tr>
<tr>
<td>12</td>
<td>-</td>
<td>character</td>
<td>mode</td>
<td></td>
<td>Mode Indicator</td>
</tr>
<tr>
<td>13</td>
<td>*57</td>
<td>hexadecimal</td>
<td>cs</td>
<td></td>
<td>Checksum</td>
</tr>
<tr>
<td>14</td>
<td>-</td>
<td>character</td>
<td>&lt;CR&gt;&lt;LF&gt;</td>
<td></td>
<td>Carriage Return and Line Feed</td>
</tr>
</tbody>
</table>
Chapter 3

Failure modes and counter measures

3.1 FMECA

Failure Modes, Effects and Criticality Analysis (FMECA) has been used to identify potential Coal Wagon failure modes, and to rank their severity. Based on the ranking, a list of countermeasures has been devised.

The countermeasures for this application are focused around engineering measurement for monitoring and control of the Coal Wagon. Other options may be available as countermeasures; however they can be excluded from the analysis because they hold little relevance to the development of the project.

A systematic procedure has been followed during the course of the FMECA, similar to that recommended by Australian Standards (Standards Australia 2008). The analysis highlighted the need for the countermeasures listed in Table 3.1.

As it is neither cost effective, nor practical to measure and control every variable, only those countermeasures repeatedly mentioned in the FMECA will be considered for inclusion into the concept design.
### Table 3.1: Proposed countermeasures

<table>
<thead>
<tr>
<th>No.</th>
<th>Parameter or control</th>
<th>Detection or fix</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Vibration</td>
<td>Wheel flat spots, bearing faults, poor sections of track, natural frequencies, loose, worn or broken suspension components</td>
</tr>
<tr>
<td>2</td>
<td>Temperature</td>
<td>Bearing friction</td>
</tr>
<tr>
<td>3</td>
<td>Pressure</td>
<td>Air leaks and brake application</td>
</tr>
<tr>
<td>4</td>
<td>RPM</td>
<td>Wheel lock</td>
</tr>
<tr>
<td>5</td>
<td>Strain</td>
<td>Fatigue and stress on the major structural elements</td>
</tr>
<tr>
<td>6</td>
<td>Displacement</td>
<td>Wagon loaded/unloaded</td>
</tr>
<tr>
<td>7</td>
<td>Tilt</td>
<td>Wagon dynamics</td>
</tr>
<tr>
<td>8</td>
<td>Location</td>
<td>Identify sections of poor track</td>
</tr>
<tr>
<td>9</td>
<td>ABS</td>
<td>Flat spots - reduction in associated vibrations/shocks and repair costs</td>
</tr>
</tbody>
</table>

A detailed analysis can be found in the Appendix E.

### 3.2 Wheel flats

Shock loading due to wheel flats are particularly pernicious since they impart high loads on the rail and wheel with each revolution, leading to very high frequency loads (Sun, Cole & Bosomworth 2010b). A wheel flat can be thought of as loss of surface roundness normally caused by abrasive wear when the wheel ceases to rotate while the wagon has momentum - wheel lock.

There have been several advancements to the braking systems which have improved the frequency of wheel lock-ups. The WILD track-side monitoring system has been specifically developed to detect flats by measuring rail acceleration and strain. During the course of development the WILD system, it became evident that acceleration measurement was the best parameter for flat detection, because
In Australia flats larger than 50mm in depth are not permissible. Y Sun et al. (2010b) have modelled the effects of 10mm and 70mm flats with a wagon speed of 80km/hr. They simulated the sensitivities between rail mount and wagon mounted accelerometers. Wagon acceleration measurement was significantly better at detecting flats. The wagon mounted accelerometers were able to detect a 20 mm flat, while the rail mounted system could not distinguish the small response over the rail irregularities. The simulation noted a significant reduction in load as the wagon speed was reduced, with the wagon mounted accelerometer was not able to detect wheel impact when the wagon speed was < 60km/hr. In order to detect the impacts, in the time domain over the noise, a bandpass filter was needed with cut-off frequencies between 10Hz and 200Hz.

3.3 Bearing faults

Vibration analysis and temperature measurement have been used for years to monitor the condition of bearings - sound and lubrication analysis can also be a good condition identifier, if and where is can be applied. Coal Wagons use a sealed bearing so live lubrication analysis is not possible and there are already track-side systems such as RailBAM which uses sound. Microphone pick-up from windage effects would make sound analysis difficult for a wagon mounted system.

A temperature rise in a bearing can be indicative of possible failure, lack of lubrication or overloading. Ambient temperature and minimal internal radial clearances, associated with new bearings, can have an effect on bearing temperature, resulting in false indicators. After all this, bearing temperatures generally increase in the latter stages of failure so a method of early detection is preferred. Studies have shown monitoring of the bearing cage yields the greatest sensitivity and earliest detection, as far as bearing temperature monitoring is concerned. The theory behind this is thought to be due to the interaction between the rolling
elements and the races being of pure rolling motion; while the interaction between the rolling elements and the cage is based on sliding motion. Therefore, a greater heat generation is anticipated in the rolling elements and cage contact than in the rolling elements and races. The mass of the cage is smaller than that of the outer race. Also the cage is not in direct contact with any other part of the bearing. Consequently, the rate of thermal dissipation from the cage should be relatively low. The bearing cage, therefore, should respond faster to an increase in heat input than other bearing components. Unfortunately, it is difficult to measure the cage since it is rotating and behind mechanical seals.

Henao-Sepulveda et al. (2005) have designed a wireless temperature sensor to measure cage temperatures. Their results showed that the cage reach steady operating temperature within 200s for both the 25lb and 45lb load case. However, they failed to measure the bearing race temperature and the bearing used for testing was not sealed so their sensor and results are of little practical importance. Measuring the bearing temperature off the housing still appears like the most suitable application.

Vibration analysis is inherently more reliable. Bearing wear, outer and inner bearing race, ball/roller and cage defects can all be determined by vibration because each fault has a unique signature allowing identification of potential faults. Time-domain acceleration data is captured, processed and transformed to produce a FFT.

The FFT can identify the bearing component frequencies and the severity judged from the amplitudes. The time-domain data is also used in the analysis to verify the FFT data and is a better measure to gauge the energies developed by the vibrations. The amplitude has units of displacement, velocity or acceleration depending on the order of the frequencies being analysed. In ideal circumstances, frequencies between; 0 to $10Hz$ use displacement, $10Hz$ to $1000Hz$ use velocity and $>1000Hz$ use acceleration. This may not always be possible because it is not recommended to double integrate an acceleration measurement because of noise amplification effects.
3.3 Bearing faults

In order for accurate diagnosis of bearing faults, the ISO bearing number or geometries must be known so the frequencies can be calculated from the component fault orders. The component frequencies are simply named after the bearing components which are identified in Figure 3.1.

![Figure 3.1: Rolling element bearing components](SKF 1996)

As mentioned earlier, each bearing component generates an unique frequency and can be calculated using Equations 3.1 to 3.5. These equations yield the orders of the fault, which is a scaler number that can be multiplied by the shaft’s rotational speed to give the frequency in Hertz or any other time based unit (Taylor & Kirkland 2004).

\[
BPFO = \frac{N}{2} \left(1 - \frac{B \cos \alpha}{P}\right) \tag{3.1}
\]

\[
BPFI = \frac{N}{2} \left(1 + \frac{B \cos \alpha}{P}\right) \tag{3.2}
\]

\[
BSF = \frac{P}{2B} \left(1 - \left(\frac{B \cos \alpha}{P}\right)^2\right) \tag{3.3}
\]

\[
FTFi = \frac{S}{2} \left(1 - \frac{B \cos \alpha}{P}\right) \tag{3.4}
\]
$$FTF_O = \frac{S}{2} \left( 1 + \frac{B \cos \alpha}{P} \right)$$ \hspace{1cm} (3.5)

Where

- $BPFO$ outer race fault frequency
- $BPFI$ inner race fault frequency
- $BSF$ roller defect frequency
- $FTF_I$ cage frequency if the inner race is rotating
- $FTF_O$ cage frequency if the outer race is rotating

These equations are considered the basics for bearing vibration analysis. There are several other bearing specific analysis like Peakvue, shock-pulse and g-sm to mention a few, that sample at very high frequencies to detect faults before they appear in the normal spectrum. These high level techniques require sophisticated equipment and therefore extend beyond the physical specifications of the hard platform intended to be used for this project.

### 3.4 Draft gear fatigue and dynamics

There is little in way of literature, specific to monitoring draft gears. Several papers have been published for the aeronautical sector for monitoring fatigue in landing gears. Daniel, Luo & Sifniotopoulos (2007) propose to use an acoustic emission technique as a suitable method for monitoring fatigue in components with similar loading conditions. Acoustic techniques require specialist equipment and a outage for testing. Therefore, this or similar methods are not suitable for this project.

Civil engineers have been using strain measurements to monitor bridge truss members for fatigue and cracking. Draft gears, similar to truss members, are designed for axial loading making strain measurement a viable option to monitoring the load cycles placed on this component. Sartor, Culmo & DeWolf (1999) noted
when strain gauging a bridge member that was designed for uni-axial loading, there was some strain due to bending in the data. The possibility of strain due to bending could be a real issue and trigger false loading events to be recorded in the data. Configuration of the strain gauges need careful consideration so strain due to bending is nullified. Counting the load cycles for comparison with the design data is proposed to be useful for determining when the component is due to be replaced. A method similar to that mentioned in Section 2.5.4 should be applicable for this application.

Measuring the tilt of the draft gear may also yield useful information. Since the draft gear is hinged to allow movement in the vertical plain, strains due to uneven load may occur. It is hypothesised that such an event could only occur if two adjoining wagons had different ride heights, causing the coupling draft gears to compensate by tilting. Under a braking event this could cause the lower wagon to apply a upward force to the trailing wagon, potentially resulting in a derailment.

3.5 Brake pressure monitoring

Localised pressure drops and air signal delay is believed to be major contributor to wheel flats. Monitoring the brake pressures would give some insight into the possible cause, so a decision can be made on the appropriate actions.

Most rolling stock operators are currently in the process of replacing their air control valve for ECP. Therefore, designing a system around an old model would be rather pointless.

The best option would is to gather data on the frequency of wheel flat before and after the installation of the ECP units. If the review does not indicate any significant change in frequency, the option to install pressure sensors in the air line becomes a higher priority. Installing pressure sensors on an in service wagon could cause issues in itself. If a pressure sensor mechanically fails, air pressure would drop (depending on the size of the leak), the brake applied, causing downtime,
3.6 Wagon body dynamics

Providing feedback to the locomotive driver is useful so he/she can adjust their driving style accordingly. In a full consist of wagons, the driver has little idea of what is going on behind the locomotive. Track gradients, wagon accelerations, lean angles, speed, track condition, mechanical status of the equipment and the number of empty or loaded wagons are important variables for the driver.

For example, in the case of an empty wagon in a loaded consists will have significance when applying the brakes. The ride height of the empty wagon will be higher than that of the adjoining wagons. Under a braking event, loaded wagons behind have more momentum and reduced braking efficiency, due to the extra mass, which has the tendency to apply derailing forces to the unloaded wagon (Cole 1998). If a system could notify the driver of such as case they could reduce the braking effort to minimise these forces.
Chapter 4

Concept design

4.1 Proposed layout

The concept shown in Figure 4.1 was used for development purpose only. It includes all the sensors types expected to be included in the final layout. The evolution from the concept to field version should be a matter of scale rather than a drastic change to hardware or software.

The layout as shown in Figure 4.2 is expected to be similar to the design used for field testing. There will likely be two modules per wagon; one for each bogie. Each wagon will share a common link/network to the locomotive where data can be viewed via a cabin display.

The router would likely be positioned in the locomotive with all the data piped to the master node via Ethernet, WiFi or ZigBee. The data would then be uploaded to an online database.

The proposed device is intended to be considered more as a maintenance tool. It can also be an aid for the operator; however, this is considered to be a secondary function.
4.1 Proposed layout

Figure 4.1: Concept data logger layout

Figure 4.2: Proposed design to be implemented in the field
4.2 Hardware selection

As discussed in Section 2.4, there is a vast array of low cost controller boards and sensors available for the ‘maker community’ which can be used for this project. However, before one starts choosing bits and pieces, the physical requirements must be determined to form a selection criteria. An analysis of the technical requirements is discussed below.

**Temperature:** maximum permissible bearing temperature is $120^\circ C$ (SKF 1996) for a typical wheel bearing. The grease temperature is often the limiting factor. Housing temperature of no more than $82^\circ C$ is recommended. Minimum temperatures could drop to just below zero during winter when the wagon is stationary. Therefore, one could expect a temperature range between $-10^\circ C$ and $+120^\circ C$.

**Acceleration:** the change in mass between a loaded and empty wagon is 100t. The bogie’s spring stiffness is $0.57t/mm$ resulting in a difference between the ride heights of $60mm$. The span between the draft gear pivots is approximately $1200mm$ which would result in a maximum tilt of $2.9^\circ$ - assuming the coupling is rigid. Irregularities in the track and wagon dynamics would cause small changes to the angle, anticipated to be in the order of $0.5^\circ$.

The highest ‘monitored’ vibration frequency would be generated by the bearing inner race. Fault frequency data for the tapper roller bearing used in the BT2-8609 series assembly could not be attained. However, the 36990/36920 tapper roller is of a similar bore and should generate frequencies of the same order. A wagon travelling at top speed ($100km/hr$), having a nominal wheel diameter of $840mm$ would rotate the axles at 632$RPM$. At this speed the inner race fault would produce a first fundamental frequency at $287Hz$. The expected shock due to a wheel flat should be $<2g$ according to Cole’s (1998) simulations. The $g’s$ load caused by a bearing fault is an unknown quantity as it would depend on the type of bearing fault and speed, although thought be $>2g$, taking into account the shock signal from a wheel flat travels up through the rolling elements of the bearing - a source of damping. A bearing fault signal, say on the outer race, is
directly transferred to the housing.

*Strain:* the range of expected amplitude values will differ greatly. There are two primary variables which are going to effect the strain level, wagon and gauge position

Firstly, the amount of load placed on the draft gear varies depending on where the wagon sits relative to the locomotive. For example, the wagon connected directly to the locomotive has the load of accelerating and overcoming friction of all the trailing wagons. Thus, loads reduce proportionally along the consist towards the rear.

Secondly, draft gears do not have a uniform cross-section and have a damper to absorb a lot of the shock during acceleration and breaking. Thus, principles of shear flow and rigid body kinematics come into play, which would require a finite element analysis for an accurate estimation. A trial and error approach would be simpler.

*Displacement:* maximum spring compression is reasoned to be 60mm. Track irregularities would cause dynamic changes to this value as the load springs absorb some of the shocks - likely in the order of millimetres over a couple of milliseconds duration.

*RPM:* mentioned earlier, the maximum wheel speed would be approximately $632\text{RPM}$ which implies a minimum cycle time of $95\text{ms}$, assuming one reference is used. If the number of reference markers are increased, the minimum cycle time becomes $t = 95/n$ where $n$ in the number of markers.

*Location:* sections of poor track can be localised to only several meters. Therefore, location data accurate to $< 3m$ should be adequate. At $100\text{km/hr}$ the wagon will cover $27.7m/s$ making the sampling frequency more critical than the horizontal accuracy. In order to get the resolution to match the accuracy, samples must be collected at a rate $> 9.2H\text{z}$.
4.2 Hardware selection

4.2.1 Micro-controller

The transient signal processing component of this project will place the most demand on the CPU. In order to capture the inner face bearing fault at 287\,Hz would require the signal to be sampled at 734\,Hz - Nyquist frequency plus 0.56. To meet this criteria the controller must be able to perform a pin read in $<\,1.7\,ms$.

After the sample speed criteria has been met, comes the issue of temporary storage. The size of each time-domain array is a function of the required spectral resolution. Since RAM memory in all micro-controllers is scarce, it becomes a balancing act between need and want. The closest two frequencies which need to be resolved are the inner and outer race faults. At 100\,km/hr these two frequencies are parted by 18\,Hz. However when the wagon reduces speed, these two frequencies will tend to merge requiring more spectral resolution. An average wagon speed of 60\,km/hr will be used as the reference for setting the resolution. At the reference, inner and outer race frequencies are parted by 10.8\,Hz which requires a spectral resolution of 5\,Hz. Assuming floating point data (4\,B), implies there must be approximately 480\,B for the spectral array and 960\,B for the time signal before writing to flash or EEPROM.

The analysis above only takes into account one channel of vibration. The proposed prototype layout will need to sample eight channels of vibration and temperature, one analog channel for strain, eight tachometers along with GPS data. The way in which the source code is written can have a large effect on the memory required. For example; when sampling more than one channel of the same parameter i.e. temperature or vibration, a variable can be allocated for each channel or one variable for all channels nested in a \texttt{loop}, each time the cycle starts a new channel is selected and over rides the previous channel’s data. The later option only needs a fraction of the memory; however it comes at a cost of speed. It is estimated 1.5kB of RAM is needed to store the raw data. The estimate has not taking into account the space required for variables contained in methods and libraries, which can be a significant portion of the consumed RAM. Therefore, controllers with $>\,20kB$ of RAM will only be considered, thus,
4.2 Hardware selection

immediately excluding any boards based on the 8b AVR architecture.

The number of available pins should not be an issue. I²C communications only need two pins to communicate with the five sensors, one analog pin for strain, four I/O pins for the tachometers and one UART for serial debugging/GPS streaming.

The Arduino Due (Figure 2.6) based on the 32b ARM architecture meets the selection criteria. Other platform like the Beaglebone or Arduino Tre are even more capable but run an operating system and have higher processor speeds, far beyond the needs of this project. Their inbuilt ADC’s are only 1.8V tolerant compared to the Arduino Due’s 3.3V, benefiting strain gauge sensitivity. Using the Beagle, Tre (or similar) would result in more complexity and consume more energy for basically the same result.

Table 4.1: Summary of the Arduino Due (Arduino 2013a)

<table>
<thead>
<tr>
<th>Item</th>
<th>Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Micro-controller</td>
<td>AT91SAM3X8E</td>
</tr>
<tr>
<td>Operating Voltage</td>
<td>3.3V</td>
</tr>
<tr>
<td>Input Voltage (recommended)</td>
<td>7 – 12V</td>
</tr>
<tr>
<td>Input Voltage (limits)</td>
<td>6 – 16V</td>
</tr>
<tr>
<td>Digital I/O Pins</td>
<td>54 (12 provide PWM output)</td>
</tr>
<tr>
<td>Analog Input Pins</td>
<td>12</td>
</tr>
<tr>
<td>Analog Outputs Pins</td>
<td>2 (DAC)</td>
</tr>
<tr>
<td>Total DC Output Current on all I/O lines</td>
<td>130mA</td>
</tr>
<tr>
<td>DC Current for 3.3V Pin</td>
<td>800mA</td>
</tr>
<tr>
<td>DC Current for 5V Pin</td>
<td>800mA</td>
</tr>
<tr>
<td>Flash Memory</td>
<td>512kB</td>
</tr>
<tr>
<td>SRAM</td>
<td>96kB</td>
</tr>
<tr>
<td>Clock Speed</td>
<td>84MHz</td>
</tr>
</tbody>
</table>
4.2 Hardware selection

4.2.2 Sensors

Inertial motion unit

The digital inertial motion unit (IMU) MPU-9150 was selected for its accuracy, robustness and versatility - temperature, acceleration and orientation data can be collected on one sensor. The MPU-9150 was also stated to have a DSP engine offering fused motion data that can be directly read from the registers; however this has turned out to be false. The DSP was one of the main reason for selecting the MPU-9150 sensor, with the hope it would save time coding and reduce processing load on the controller.

There are three separate 3-axis sensing elements in the MPU-9150 making it a 9 DOF sensor; 3-axis accelerometer, 3-axis gyroscope, 3-axis manometer and a temperature sensor. Hence, one unit can take care of the draft gear tilt, bearing temperature and vibration measurements. Only dealing with one sensor type for these three measurements will reduce the effort required to develop the source code. Unlike many MEM’s the MPU-9150 is temperature compensated which significantly reduces drift compared to non-compensated MEM’s. A summary of each sensing element feature can be found in the tables provided.

Inductive proximity probe

An 4010045 inductive proximity sensor will be used to provide the pulse from a target placed on the wheel - a piece of steel protruding from the wheel face. The justification for selecting this inductive sensor was price driven. Consequently this came at the expense of manufacturing quality and documentation, although should be suitable for at least testing purposes.

An inductive sensor is the preferred option for the following reasons:

- Hall effect sensor requires magnetic targets which tend to loose their residual magnetism over time which could require recalibration regularly
### 4.2 Hardware selection

**Table 4.2: MPU-9150 gyro features (InvenSense 2012a)**

- 3-axes with selectable ranges of $\pm 250$, $\pm 500$, $\pm 1000$ and $\pm 2000^\circ/s$
- $16b$ simultaneous sampling ADC
- $\pm 0.04^%/^\circ C$ over a temperature range of $-40^\circ C$ to $+125^\circ C$
- Total noise $0.06^\circ/s - RMS$
- Programmable low pass filter with minimum and maximum cut frequencies $5Hz$ and $256Hz$ respectively
- Maximum output data rate $8kHz$
- Setting time $30ms$

- Coal debris and other foreign matter could work its way in between the sensor face and wheel which will effect capacitive and optical sensing methods
- Ultrasonics sensors have too poor a response time
- Inductive sensors are often used for tachometer pick-ups in industrial applications

NB: most of the data contained in Table 4.7 has been determine by experimentation - refer to Figure 5.1. An adjustable power supply was used to feed the sensor 12V. The probe sensitivity was determined by measuring the offset between the probe face and flat metal surface when the LED on the back of the probe shun. The voltage between high and low was noted on the multimeter.

**Strain gauge bridge**

A Micro-Measurements general purpose C2A-06-062LW-350 linear strain gauge was donated to the project and would be similar, if not the same type, as the
4.2 Hardware selection

Table 4.3: MPU-9150 Accelerometer features (InvenSense 2012a)

- 3-axes with selectable ranges of ±2, ±4, ±8 and ±16°/s
- 16b simultaneous sampling ADC
- ±0.02%/°C over a temperature range of −40°C to +125°C
- Total noise 4mg – RMS
- Programmable low pass filter with minimum and maximum cut frequencies 5Hz and 256Hz respectively
- Maximum output data rate 1kHz
- Self test input
- Tap detection with programmable interrupt
- High-g interrupt

Table 4.4: MPU-9150 Temperature sensor and general specifications (InvenSense 2012a)

- Range −40°C to +125°C
- Tolerance ±1°C
- $V_{DD}$ range between 2.4V to 3.5V
- 1.8V logic
- Nominal operating current with the magnetometer disabled
- User programmable power saving modes
4.2 Hardware selection

Figure 4.3: Equipment used to test the inductive sensor

Table 4.5: Summary of the 4010045 inductive proximity sensor

<table>
<thead>
<tr>
<th>Item</th>
<th>Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calibrated material</td>
<td>Iron and stainless steel</td>
</tr>
<tr>
<td>Input Voltage (limits)</td>
<td>6 – 36V</td>
</tr>
<tr>
<td>Logic type</td>
<td>NPN</td>
</tr>
<tr>
<td>Logic high</td>
<td>2.5V</td>
</tr>
<tr>
<td>Logic low</td>
<td>2.3mV</td>
</tr>
<tr>
<td>Wiring colour code</td>
<td>$V_{CC}$(brown), signal(blue) and $GND$(black)</td>
</tr>
<tr>
<td>Maximum detection distance</td>
<td>8mm</td>
</tr>
<tr>
<td>Dimensions</td>
<td>7mm x 70mm</td>
</tr>
</tbody>
</table>

one(s) to be applied to the draft gear when time comes for field testing. For balance, three Micro-Measurements 350Ω±0.01% precision resistors used to form a quarter bridge circuit.

$$
\epsilon = \frac{-4V_r}{GF (1 + 2V_r)} \left(1 + \frac{R_i}{R_g}\right) \left(\frac{1}{G}\right)
$$

(4.1)
4.2 Hardware selection

Table 4.6: Micro-Measurements C2A-06-062LW-350 strain gauge specifications

<table>
<thead>
<tr>
<th>Item</th>
<th>Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resistance</td>
<td>350Ω ± 0.6</td>
</tr>
<tr>
<td>Gauge factor @24°C</td>
<td>2.125 ± 0.5%</td>
</tr>
<tr>
<td>Transverse sensitivity</td>
<td>(+0.9 ± 0.2)%</td>
</tr>
<tr>
<td>Strain range</td>
<td>±3% over −50°C to +80°C</td>
</tr>
<tr>
<td>Dimensions</td>
<td>6.40mm x 4.32mm</td>
</tr>
</tbody>
</table>

Where

\[ V_r = \frac{(V_{OUT}/V_{IN})_{\text{strained}} - (V_{OUT}/V_{IN})_{\text{unstrained}}}{GF} \]

- \( V_r \) is the voltage output of the bridge circuit
- \( GF \) is the gauge factor
- \( R_l \) is the lead wire resistance (Ω)
- \( R_g \) is the gauge resistance (Ω)
- \( R_l/R_g \) term nearly always approaches zero unless very long lead wires are used
- \( G \) is the amplifier gain

The strain output of a quarter bridge is determined by Equation 4.1, normally multiplied by \( 10^6 \) to convert the units into micro-strain.

\[ V_r = (V_{OUT}/V_{IN})_{\text{strained}} - (V_{OUT}/V_{IN})_{\text{unstrained}} \]

Figure 4.4: Quarter strain bridge circuit reference

As mentioned in Section 2.4.4, the voltage signal from the bridge is small and in differential form. The in-built ADC in the Arduino can only handle signals between 0V and +3.3V. Therefore the signal needs to pass through an instrument amplifier (Texas Instruments INA126P) to boost the signal and give it a reference to ground. The INA uses the +3.3V provided by the Arduino’s voltage regulator which is also used for bridge excitation. Configuring the circuit in this manner reduces the chances of the INA providing a potentially damaging voltage to the
4.2 Hardware selection

analog pin. The maximum possible bridge current is $18.9mA$ - well within the supply capacity of the voltage regulator. The gain of the INA is set to 5 - open circuit on pins 1 & 8.

The bridge nodes are wired to the INA’s pins 2 & 3 - input signal pins. The INA’s pin 6 is wired to an analog pin of the Arduino. Two $1k\Omega$ resistors form a voltage divider between $+3.3V$ and ground so the strain signal is limited between the voltage levels of the Arduino’s analog pins.

\[
G = 5 + \frac{80k\Omega}{R_G}
\]  

(4.2)

Where $R_G$ is the gain resistor value ($k\Omega$) on pin 1 & 8 - $\infty$ in this case

GPS module

The GPS Bee has been used for the GPS receiver. The GPS Bee runs a u-blox NEO-5 series module that has the highest refresh rate and horizontal accuracy out of all Arduino compatible receivers. Plus, the GPS Bee pin configuration is compatible with the standard XBee shield designs, which made it easier to connect to the Arduino than most other modules.
Table 4.7: GPS Bee specifications

<table>
<thead>
<tr>
<th>Item</th>
<th>Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. channels</td>
<td>50</td>
</tr>
<tr>
<td>Time to first fix</td>
<td>29s (cold start) &lt; 1s (hot start)</td>
</tr>
<tr>
<td>Sensitivity</td>
<td>$-160dB$</td>
</tr>
<tr>
<td>Horizontal accuracy</td>
<td>$&lt; 2m$</td>
</tr>
<tr>
<td>Velocity accuracy</td>
<td>$0.1m/s$</td>
</tr>
<tr>
<td>Update rate</td>
<td>1Hz (default) 4Hz (config. max)</td>
</tr>
<tr>
<td>Interfaces</td>
<td>$6.40mm \times 4.32mm$</td>
</tr>
</tbody>
</table>

Routing and storage

A NetComm NTC-6908 industrial router was used in conjunction with an Arduino Ethernet shield to communicate with the internet. The Ethernet shield also provides an SD card slot so data can be written to the permanent memory via SPI commands.

![Arduino Ethernet shield](image)

The mobile network may not cover the full area of the rail network and therefore there may be holes in the data reported to the cloud service. To account for this situation, the program will need to check for a client connection to the service. If the connection fails, data will need to be written to a separate file on the SD card and when reconnected the data from that file uploaded to the cloud.

The initial field testing is planned for Hunter Network and according to Telstra,
4.2 Hardware selection

90% of the rail corridor is covered by a 3G signal - refer to Figure 4.7.

Figure 4.7: Telstra’s coverage of the Hunter Network

4.2.3 Displacement sensor

A Sharp GP2Y0A02YK0F IR sensor was used to develop the code to determine if the wagon is loaded or empty. It is unlikely that this sensor will not see active duty. A linear actuator would be more suitable; however they both provide a similar signal so the code should be able to be ported over to either sensor without too much modification.

The displacement sensor is intended to measure the compression of the load springs on the bogie. When loaded the gap between the bolster and side rails should change by approximately 50\text{mm}. A construct of two \texttt{if} statements would be sufficient to output the load state.

Sensitivities between the IR and linear displacement sensor will be different and therefore the limits in the \texttt{if} condition will need to be calibrated to suit.
4.3 Scripting

The Arduino scripting language uses C/C++ variation. Unlike typical C code their is no `main`; instead, Arduino breaks the body into two pieces - `setup` and `loop`. The `setup` method only executes once while the `loop` method is continuous unless explicitly exited. In order to reduce the volume script contained in this report, only those lines of code with significance have been copied into the report and discussed. Variable declarations, `setup`, `loop`, comments and the like add little value to the content of this report and therefore have been omitted.

NB: considering this project was largely a programming exercise, it is considered appropriate to include the blocks of scripts in the body of this report. The comment leader followed by three dots has been use to represent either unrelated, similar or repeated code. Unlike most program based projects, the results/output of the code is rather arbitrary; that is, the program either produces, stores and communicates data or fails to do so. Therefore, an explanation of the program on a functional level has has been considered significant, relative to analysing the data. The term ‘method’ is equivalent to ‘function’ in most other programming languages.

4.3.1 Temperature sampling

Temperature was one of the simpler parameters to code. It was only a matter of reading data from the MPU-9159 temperature registers and scaling the result by the equation given in the product sheet (InvenSense 2012b).

The code in Listing 4.1 was used to communicate with the MPU-9159 temperature registers. The Arduino `Wire` library was needed to handle the I²C component. Like most I²C sensors, two registers are read. The first register contains the most significant bit (MSB) and the second, the least significant bit (LSB). The MSB and LSB are combined using binary operators to form the final 16b value, which is then scaled.
4.3 Scripting

Listing 4.1: Temperature sampling code

```c
float readMPU_Temp()
{
    Wire.requestFrom(MPU_Temp_I2C_ADDRESS, 2);
    byte MSB = Wire.read();
    byte LSB = Wire.read();
    return (float)(((short)(MSB << 8) | LSB) / 340.0 + 35.0);
}
```

The code constructs used to access the temperature data is very similar to how the vibration and gyroscope data is accessed. The I\(^2\)C address and scaling equations is the only notable modifications made to the method above to access acceleration and gyroscope registers. Therefore, the sensor read code will not be repeated in Sections 4.3.5 & 4.3.6.

### 4.3.2 Load condition

As described in Section 4.2.3 the following code is not likely to change apart from the condition values assigned to variables, loaded and empty, which is dependent on the sensor sensitivity.

The logic follows that if the wagon is loaded the ADC value will be greater than if the wagon was empty. If the measured ADC value is somewhere in between what has been the ADC values for the empty and loaded wagons, that implies that the wagon is either in the process of being loaded or there is a sensor fault.

Listing 4.2: Tachometer sensor pin configuration and wheel revolution counter

```c
char loadCase()
{
    //...
```
4.3 Scripting

```c
disp = analogRead(dispPin);
if(disp > loaded)
{
    return 'Loaded';
}
else if(disp < empty)
    return 'Empty';
else
{
    return 'Being loaded/sensor error';
}
}
```

4.3.3 Tachometer

The input assigned for the tachometer signal was configured as an external interrupt. The interrupt pin is continuously monitored for a falling signal which triggers a cycle of the `revCounter` method. Each time the counter method is executed the revolution counter variable is indexed by one.

Listing 4.3: Tachometer sensor pin configuration and wheel revolution counter

```c
// ...
pinMode(tachoPin, INPUT);
digitalWrite(tachoPin, HIGH);
attachInterrupt(tachoPin, revCount, FALLING);
// ...

void revCount()
{
    revTracker++;
}
```
4.3 Scripting

The **axleKinematics** method works by timing the period between two low signals. Equations 2.2 & 2.3 are used to calculate the axle RPM and change in RPM (delta-RPM) depending on the input argument.

The method has been set up for one *RPM* marker on the wheel or axle. Higher resolution data can be achieved with more markers and polled less frequently by the main **loop**; however slower response times would result. Increasing the condition greater than one revolution in the **if** statement would also have the same effect.

This method suffers from jittery performance at low *RPM*. Additional conditions should to be placed on the method when called in the main **loop**; a 100*RPM* threshold should be sufficient. Increasing the number of markers on the wheel should also increase the tachometer performance at low *RPM*.

The **Serial** lines of code are only an error handling function.

Listing 4.4: Pulse timer

```c
float axleKinematics(char varSelect)
{
    //...
    if (revTracker >= 1)
    {
        deltaTime = millis() - oldTime;
        RPM[1] = RPM[2];
        RPM[2] = revTracker*c1/(float)deltaTime;
        deltaRPM = (RPM[2] - RPM[1])*c2/(float)deltaTime;
        oldTime = millis();
        revTracker = 0;
    }
    switch(varSelect)
    {
        case 'V':
```
4.3 Scripting

```cpp
return RPM[2];
break;
case 'A':
return deltaRPM;
break;
default:
    Serial.print(varSelect);
    Serial.println(" argument was received but not an available option.");
    Serial.println("Try V for RPM or A for Delta-RPM.");
    break;
}
}
```

Only very basic testing of the tachometer be carried out i.e. passing a metal object in front of the probe and counting the time in between cycles. It was difficult to maintain a constant frequency; however the RPM appeared to trend with changes to the period - refer to Figure 4.8.

### 4.3.4 GPS parsing

A state machine was used to parse the GPS data. There are two components to the parser code; one section handles the current state and the other section, the previous state.

The `processStream` method analyses each character byte by byte and checks the character against each state in the `case` statement, before being passed to the `previousState` method.

`isdigit` checks if the character being streamed in from the serial port is a numerical one, and if it is true the data variable concatenates each digit as it arrives before
then next state is reached and passed to a formatting method.

Listing 4.5: Current state parser

```c
void processStream(const byte c)
{
    if (isdigit(c))
    {
        data *= 10;
        data += c - '0';
    }
    else
    {
        previousState();
        switch (c)
        {
```
case 'G':
    state = got_G;
    break;

case 'P':
    state = got_P;
    break;

case 'R':
    state = got_R;
    break;

case 'M':
    state = got_M;
    break;

case 'C':
    state = got_C;
    break;

case ',':
    state = got_COM1;
    break;

//...

default:
    state = none;
    break;

}
}

The `previousState` method handles the state passed to it by the `processStream` until the piece of data being parsed arrives; in this case the time formatted in hhmmss. The cycle then continues to the next expected state until all the useful
4.3 Scripting

data (time, longitude, latitude, speed and data) from the GPRMC sentence has been processed.

Listing 4.6: Pulse timer

```c
void previousState()
{
    switch (state)
    {
    case got_0:
        break;
    case got_G:
        break;
    case got_P:
        break;
    case got_R:
        break;
    case got_M:
        break;
    case got_C:
        break;
    case got_COM1:
        time(data);
        break;
    //...
    }
    data = 0;
}
```
4.3 Scripting

4.3.5 Tilt measurement

Angular measurement of the draft gear can be done using the MPU-9150. The output from the accelerometer can be used to calculate the tilt angle, as can gyroscopic data. The values in the accelerometer registers can be thought of as vector and therefore can be used in Euler angle equations as shown in first line of Listing 4.7. Another approach is to numerically integrate the gyroscopic data.

Listing 4.7: Euler angles

\[
\begin{align*}
tilt\text{Accel} &= \text{atan2}\left(\frac{\text{accelY} - \text{biasAccelY}}{256.0}, \frac{\text{accelZ} - \text{biasAccelZ}}{256.0}\right) \times 360.0 / (2 \times 3.142); \\
tilt\text{Gyro} &= tilt\text{Gyro} + \left(\frac{\text{gyroX} - \text{biasGyroX}}{16.384}\right) \times \text{deltaTime};
\end{align*}
\]

It is well documented that accelerometers output a noisy signal while gyroscopes suffer from drift. Therefore a one dimensional Karlman filter was used to combine the benefits of the accelerometer and gyroscope. The Karlman algorithm filters the drift of the gyroscope by a weighted average determined from the accelerometer data.

The one dimensional Karlman filter in Listing 4.8 sourced from Glover (2010) and ported from Python to C.

Listing 4.8: Karlman filter

\[
\begin{align*}
tilt\text{Predict} &= tilt\text{Predict} + \left(\frac{\text{gyroX} - \text{biasGyroX}}{16.384}\right) \times \text{deltaTime}; \\
Pxx &= Pxx + \text{deltaTime} \times 2 \times P xv + \text{deltaTime} \times P vv; \\
P xv &= \text{deltaTime} \times P vv; \\
Pxx &= \text{deltaTime} \times \text{giroVar}; \\
P vv &= \text{deltaTime} \times \text{deltaGiroVar}; \\
kx &= Pxx \times \left(\frac{1}{(Pxx + \text{accelVar})}\right);
\end{align*}
\]
4.3 Scripting

\[ kv = \frac{P_{xv}}{P_{xx} + \text{accelVar}}; \]

\[ \text{tiltPredict} += (\text{tiltAccel} - \text{tiltPredict}) \times k_x; \]

\[ P_{xx} *= (1 - k_x); \]
\[ P_{xv} *= (1 - k_x); \]
\[ P_{vv} -= kv \times P_{xv}; \]

4.3.6 Vibration signal processing

Sampling vibration data from the accelerometer has turned out to be one of the most processing inventive tasks of this project. The working program has yet to include the FFT capabilities and only outputs the RMS vibration. When attempted to include the FFT methods to the main code, hook-ups occurred and corrupted the data from the other sensors. However, the FFT works as a stand alone program and will be discussed in this section.

Before the vibration data is handled by the following methods, it is sampled at 1kHz in the loop by using the `delayMicroseconds` to time the spacing between samples - set to 1ms. A total of 512 samples \( (N) \) are collected and stored in a buffer of the complex class. While each sample is being stored in the array it is scaled to \( g' \)s. The scale is dependant on the selected range - \( 2g \) in this case.

The first step of the process is to remove the DC offset by passing the buffered data through a high pass filter which can be seen in Listing 4.9 derived from Equation 2.7 where `deltaTime` is equivalent to `delayMicroseconds` and `Tconst` is adjusted so as to allow frequencies just above 10Hz to pass.

Listing 4.9: Removing the DC offset using a high pass filter

```cpp
float highPass(float x[])
{
    alpha = deltaTime/(T_const + deltaTime);
```

4.3 Scripting

```c
// ...
for (short i = 1; i <= N; i++)
{
    y[i] = alpha*y[i - 1] + alpha*(x[i] - x[i - 1]);
}
return y;
}
```

Meaningful data can now be extracted from the waveform i.e. RMS vibration and frequencies. The following are the algorithms used to transform the filtered vibration buffer to RMS value or the frequency-domain.

Listing 4.10: RMS algorithm

```c
float RMS(float hpBuffer[])
{
    // ...
    for (short i = 1; i <= N; i++)
    {
        sumSquare += hpBuffer[i] * hpBuffer[i];
    }
    return sqrt(sumSquare/N);
}
```

The FFT listing has been ported over from Librow (2013) using the Cooley-Tukey method. This algorithm requires the use of the complex class which can be found in Appendix H.1. A plot of the ‘real’ part of the output from the algorithm can be found in Figure 4.9 from a shaker cycling at 45 Hz. The buffer was windowed (code not shown) before passing it to the FFT method using the Hann equation.

Listing 4.11: FFT algorithm

```c
boolean FFT(const complex *const Input, complex *const Output, const unsigned int N)
```
4.3 Scripting

```c
{
    // ...
    if (!Input || !Output || N < 1 || N & (N - 1))
    {
        return false;
    }

    unsigned int Target = 0;
    for (short i = 0; i < N; i++)
    {
        Output[i] = Input[i];
        unsigned short Mask = N;
        while (Target & (Mask >>= 1))
        {
            Target &= ~Mask;
        }
        Target |= Mask;
    }
    // ...
    for (short Step = 1; Step < N; Step <<= 1)
    {
        const unsigned int Jump = Step << 1;
        const double Sine = sin(delta * .5);
        const complex Multiplier(-2.0*Sine*Sine, sin(delta));
        complex Factor(1.0);
        for (unsigned int Group = 0; Group < Step; ++Group)
        {
            for (unsigned int Pair = Group; Pair < N; Pair += Jump)
            {
                const unsigned int Match = Pair + Step;
                const complex Product(Factor*Data[Match]);
            }
        }
    }
```
4.3 Scripting

Data[Match] = Data[Pair] - Product;
Data[Pair] += Product;
}

Factor = Multiplier*Factor + Factor;
}

return true;
}

Figure 4.9: 512Hz bandwidth FFT sample

4.3.7 Strain and cycle counter

The rawStrain method samples the amplified strain signal on the nominated analog pin. The ADC resolution is set to 12b using the analogReadResolution function prior to sampling the signal. The first operation shifts the ADC signal so it is balanced around the zero and then converted to a voltage scale based on the number of 12b deviations between 0V and +3.3V and ± the balanceOffset
measured after the gauge is applied to the surface of the test object. Using
Equation 4.1, one can then can calculate the strain.

The strain frequency is expected to be relatively low and may be effected by
higher frequency influences. In order to eliminate these other factors, a discrete
low pass filter is used. It is assumed that the strain frequency should not exceed
5Hz; however in order to have some margin for safety the frequency cut off will
be higher - set at 10Hz. The algorithm used in Listing 4.12 was derived from
Equation 2.6.

Listing 4.12: Raw strain acquisition

```c
float rawStrain()
{
    alpha = deltaTime/(T_const + deltaTime);
    //...
    adc = map(analogRead(strainPin), 0, 4096, -2048, 2047);
    volts = E*0.001612 + balanceOffset;
    x = (-4.0*abs(volts))/(2.2125*(1.0 + 2.0*abs(volts))
         *5.0)*1000000;
    y = alpha*x + (1 - alpha)*y;
    return y;
}
```

A cycle counting algorithm has yet to be implemented in the Ardunio language;
however one can export the strain data either from the cloud service or SD card
and use solvers available in Excel or MATLAB. The Level-crossing Counting
method as described in ASMTE1049-85 (2011) has been carried out on a sample
of strain data collected using Listing 4.12 then converted from strain to a load
level, where a load level three is equivalent to 2.5V, level two 2.2V, level one 1.9V
and so on. The result can be viewed in Figure 4.10(b).
4.3 Scripting

(a) Smoothed & levelled strain signal  
(b) Level crossing method histogram

Figure 4.10: Data sampled using the Listing 4.12

4.3.8 HTTP feed

In order to transfer data from the Arduino to the cloud service (Xively), the data needed to be encapsulated in a TCP wrapper and formatted according to the API of the site. HTTP headers similar to that found in Listing 2.3 were used. The main **loop** checks for a client connection and then proceeds to develop the data feed using the methods in Listings 4.13 below.

Listing 4.13: HTTP PUT and connection error handling method

```cpp
void feed()
{
    int dataStreamLength = jsonBodyData().length();
    int locationStreamLength = jsonBodyLocation().length();
    for(int i = 0; i <= 1; i++)
    {
        if(client.connect(serverName, 80))
        {
            client.print("PUT /v2/feeds/");
            client.print(FEED_ID);
            client.println(".json HTTP/1.1");
```
The cloud service used for this project has an Arduino library available which is simple to use; however it does not support the full functionality of the site, namely meta data. Therefore two feeds were coded; one for the condition data and the other for the meta data (GPS waypoints).

Anything up to 255 separate data channels can be passed to the service for each registered device, although only eight were used during testing - 3 RMS vibration, 1 temperature, 1 tilt, 1 strain channel and displacement. Conditions can be placed on each datasreams channel to trigger alarms based on the overall level of the incoming data - refer to Figure 4.11.

Listing 4.14: JSON formatted monitored data and connection error handling method

```java
String jsonBodyData()
```
4.3 Scripting

```java
{
   //...
   String body;
   body += "{\n";
   body += "\"version\":"1.0.0","\n";
   body += "\"datastreams\" : [ {\n";
   body += "\"id\" : \""
   body += "Temperature";
   body += "\",\n";
   body += "\"current_value\" : \"";
   dtostrf(temp, 6, 3, decimalToStringBuffer);
   body += String(decimalToStringBuffer);
   //...
}
```

![Figure 4.11: Xively alarm set up pane](image)

The meta data feeds contains the GPS waypoints and some additional information to label the GPS. For example, the `location` tag informs the site that the incoming feed contains meta data and to treat it differently than a `datastreams` feed. The `disposition` can be set to either ‘mobile’ or ‘static’. The mobile descriptor forces the site to look for new waypoints in each new feed. If set to static, the first meta data feed will remain fixed until a new connection is made by the same device. The other descriptors only add text based information to the feed.
String jsonBodyLocation()
{
    //...
    String body;
    body += "{\n";
    body += "\"location\": {\n";
    body += "\"disposition\": "mobile",\n";
    body += "\"name\": "My Trailer",\n";
    body += "\"exposure\": "outdoor",\n";
    body += "\"domain\": "physical",\n";
    body += "\"ele\": "
    dtostrf(gps.f_altitude(), 7, 5, decimalToStringBuffer);
    body += String(decimalToStringBuffer);
    body += "\",\n";
    body += "\"lat\": ";
    dtostrf(lat, 7, 5, decimalToStringBuffer);
    body += String(decimalToStringBuffer);
    body += "\",\n";
    body += "\"lon\": ";
    dtostrf(lon, 7, 5, decimalToStringBuffer);
    body += String(decimalToStringBuffer);
    body += "\"\n";
    body += "}";
    body += "}"
    return body;
}
Chapter 5

Results

5.1 Google Fusion Table map

The program currently has the capability to save the trend data from the sensors (excluding the tachometer) along with GPS waypoints to a CSV log file every second. On a return trip the data stored on the SD card can be loaded onto a computer for analysis.

Once the data resides on the computer, it can be imported into Google Fusion or Google Earth Pro. Google Fusion allows the GPS waypoints and trend data to be visualised on a 2D map as show in Figure 5.1. Google Earth Pro is even more powerful. Data can be visualised in the third dimension with the vertical height of the trend following the course of waypoints.

5.2 Live data feed

Assuming there is a wireless network signal, the same data stored on the SD card is transferred to Xively; the cloud service used for this project for storing data sampled by the Arduino unit.
5.2 Live data feed

The wagon then can be monitored in real time by viewing the current value or graphs of each channel - refer to Figure 5.2 & 5.3 respectively. Alarms can also be set up so constant monitoring is not required by using the trigger function shown in the previous chapter.

Every second the location of the wagon can updated and displayed on a small map for tracking functionality.

![Figure 5.1: SD card data imported into Google Fusion charts](image)

![Figure 5.2: Result of Xively datastreams and location feeds](image)

(a) Xively trend  
(b) Output of meta data feed

Figure 5.2: Result of Xively **datastreams** and **location** feeds
5.2 Live data feed

<table>
<thead>
<tr>
<th>Channels</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>IR-Sensor</td>
<td>Loaded</td>
</tr>
<tr>
<td>Speed</td>
<td>20.4</td>
</tr>
<tr>
<td>Strain-Gauge</td>
<td>-134453</td>
</tr>
<tr>
<td>Temperature</td>
<td>22.500</td>
</tr>
<tr>
<td><strong>Tilt</strong></td>
<td>73.8</td>
</tr>
<tr>
<td>X-Vibration-RMS</td>
<td>0.02</td>
</tr>
<tr>
<td>Y-Vibration-RMS</td>
<td>0.01</td>
</tr>
<tr>
<td>Z-Vibration-RMS</td>
<td>0.47</td>
</tr>
</tbody>
</table>

Figure 5.3: All Xively channel feeds
Chapter 6

Conclusion

6.1 Project summary

The main objective of this project was to develop a Coal Wagon data logging unit using low cost ‘hobbyist’ level hardware. Based on this philosophy, the concept design was built.

Early in the project it was found out that the need for the unit to control the braking system has become redundant since most of the asset owners are in the process of retrofitting ECP braking systems to their wagons.

The literature review found that there is no commercially available instrumentation specifically designed to ‘permanently’ monitor rolling stock assets, possibly due to the lack of available power. A micro-controller based solution, like the one proposed in this report, has very low power consumption (estimated 1.6W based on the power consumption of test configuration) and can be reduced even further by programming the controller to turn off the MEM sensors when not in use. The power requirement is well within the capability of a small solar panel and battery pack.

A review of wagon asset failure modes was conducted by reading incident reports
and undertaking a failure mode and effect analysis. The FEMA identified the components with the highest need for monitoring. An investigation into the fault symptoms of each component resulted in the variables to measure. From this information the electronic hardware could be selected.

After the hardware was selected the C/C++ computer language was learnt in order to program the controller. Significant amounts of time was spent fault finding to track down bugs in the program and hardware.

As a result programming, the controller can now save data to a SD card in a format compatible with most data analysis software packages, at the same time uploading data to a cloud service so monitoring on the wagon can be done in real time.

Testing of the device to a level first planned has not yet prevailed. The risk of damaging the micro-controller and sensors by testing it on a small trailer was too great. Sensors which could be fitted to the internals of a vehicle were tested with successful results.

Although the unit is not yet deployable, the Arduino platform is showing promise and has proven to be a useful prototyping tool.

## 6.2 Further work

Further testing of the tachometer is required. It is proposed to use a lathe where one can control the RPM and calibrate the tachometer program accordingly.

Currently the micro-controller and sensors are exposed and would require industrial enclosures for protection. In order to save costs only one of each sensor has been accounted for in the program. Therefore, the program needs to be scaled to suit the additional eight MPU-9150 and three inductive sensors.

A sustainable power supply needed to be designed. Since there is no power supply
on board the design would need to consider alternative forms of energy like solar, compressed air and mechanical generation.

The program needs further work in the area of error checking. It would be beneficial if it could check for faults on the wagon and also faults in itself e.g. low power, faulty sensor etc.

Find the hook up causing issues with the other parts of the program when the FFT algorithm is called. Once the tachometer and FFT component of the program are working, write code that will order normalise the frequency data and track the bearing frequency bands.

Investigate the possible use of the tap detection feature in MPU-9150 to detect wheel flats.
References


ANSYS (2013), ‘Which fatigue methodology is appropriate - stress-life or strain-life’.


REFERENCES


Byte Paradigm (2013), ‘Introduction to I2C and SPI protocols’.

CAN Bus (n.d.), ‘What is can bus’.


Digi (2013), ‘Need wireless? choose xbee.’.


InvenSense (2012b), MPU-9150 Register Map and Descriptions, 4 edn, InvenSense Inc, Sunnyvale California.


W3C (2013), ‘Http - hypertext transfer protocol’.


Welch, G. & Bishop, G. (2006), An introduction to the kalman filter, Report TR 95-041, Department of Computer Science, University of North Carolina, Chapel Hill.

Appendix A

Project specification
FOR: ADAM ROBERTS-APPS

TOPIC: DEVELOPMENT OF A COAL WAGON MONITORING AND CONTROL SYSTEM

SUPERVISORS: Prof John Billingsley

PROJECT AIM: To prototype a cost effective instrument that can measure, report and output controls based on input signals generated by Coal Wagon conditions or states however; it is hopeful that this project can be easily adapted to other applications and industries.

SPONSORSHIP: Self-funded project

PROGRAMME: Issue A, 24th March 2013

1. Research background information relating to wagon failures, failure modes and effects analysis (FEMA), commercially available wagon monitoring and control systems in the market, electrical and pneumatic circuit design, C programming and data processing.
2. Identify all possible fault conditions including false positives and parameters to be measured from FEMA.
3. Sensor and controller platform selection based on the requirements of Item 2.
4. Design the electrical/pneumatic circuits for analog data sensing, control I/O and communication networks.
5. Program the unit and investigate the possible integration of third party software packages to display the data in a user friendly format i.e. MATLAB, LabView, Java and the like.
6. Test the system on an small automotive vehicle trailer.

As time permits:

7. Make the system physically robust enough for the intended purpose i.e. placing the sensitive components in IS boxes and conduit.
8. Develop a sustainable power source.
9. Extend the system to be multiplex and self-checking.
Appendix B

Project time-line
### Research Project Time-line

<table>
<thead>
<tr>
<th>Jan</th>
<th>Feb</th>
<th>Mar</th>
<th>Apr</th>
<th>May</th>
<th>Jun</th>
<th>Jul</th>
<th>Aug</th>
<th>Sep</th>
<th>Oct</th>
<th>Nov</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preliminary work</td>
<td>Preliminary report</td>
<td>Preliminary report Due</td>
<td>Hardware design</td>
<td>Prototype constructed</td>
<td>Software design</td>
<td>Sketches loaded</td>
<td>Integration testing</td>
<td>Field testing</td>
<td>Final dissertation</td>
<td>Submission</td>
</tr>
</tbody>
</table>

**Legend:**
- Green: Initial stage
- Purple: Mid-stage
- Red: Final stage
- Orange: Milestone
Appendix C

Risk assessment
## Safe work method statement (SWMS)

### Administration Section

<table>
<thead>
<tr>
<th>Project name:</th>
<th>Dev of a coal wagon monitoring and control system</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job / Project no:</td>
<td>ENG4111/2</td>
</tr>
<tr>
<td>SWMS no:</td>
<td>Version no:</td>
</tr>
<tr>
<td>Activity / Task assessed:</td>
<td>Electrical work – building sensor circuits</td>
</tr>
<tr>
<td>Date of assessment:</td>
<td>30/04/2013</td>
</tr>
<tr>
<td>Prepared by:</td>
<td>Adam Roberts-Apps</td>
</tr>
<tr>
<td>Approved by:</td>
<td></td>
</tr>
<tr>
<td>Reviewed at first use by:</td>
<td></td>
</tr>
<tr>
<td>Review date</td>
<td>Controls effective</td>
</tr>
</tbody>
</table>

### Work method assessment

(Must be task specific only, no generic or motherhood statements will be accepted)

<table>
<thead>
<tr>
<th>Step</th>
<th>Task</th>
<th>Hazard</th>
<th>Consequences</th>
<th>Risk Score</th>
<th>Controls</th>
<th>Risk score</th>
<th>Control hierarchy</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tidy work area of any non-related materials</td>
<td>Excessive amount of text book laying on the ground</td>
<td>Slip, trip or fall causing an impact injury</td>
<td>Moderate</td>
<td>Clear access ways</td>
<td>Eliminate</td>
<td>Low</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Set up equipment</td>
<td>Lifting toolbox</td>
<td>Pull back Nerve damage</td>
<td>Moderate</td>
<td>Lift from the knees</td>
<td>Administrate</td>
<td>Low</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Positioning electronic components onto a bread or PCB board</td>
<td>Pushing sharp edged IC’s and pins in boards</td>
<td>Cuts and puncher wounds</td>
<td>Moderate</td>
<td>Wear gloves</td>
<td>PPE</td>
<td>Low</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Soldering</td>
<td>Hot surface</td>
<td>Skin burn</td>
<td>High</td>
<td>Hand tools training</td>
<td>Administrate Engineer</td>
<td>Low</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Turn off soldering iron and allow to cool</td>
<td>480 V mains</td>
<td>Electrocut</td>
<td>Moderate</td>
<td>Use a RCD protected circuits</td>
<td>Engineer Administrate Administrate</td>
<td>Low</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Pack up equipment and parts and clean work area</td>
<td>Stepping on small items</td>
<td>Cut foot</td>
<td>Moderate</td>
<td>Wear shoes</td>
<td>PPE Eliminate</td>
<td>Low</td>
<td></td>
</tr>
</tbody>
</table>

* Note that any hazard that has a consequence of Significant, Serious, Critical or Disastrous is a “Significant Hazard” – NZ only.

---

### Personal protective equipment (P.P.E) required

- Fall arrest equipment
- Steel capped boots
- Hard hat
- Safety glasses
- Sun screen
- Hearing protection
- Respirator
- Gloves
- Disposable overalls
- Hi Visiblity Vest

**Description of the specific PPE that is to be used**

**Other PPE, Please List**

### Training required

**List Mandatory Training Requirements for the Job Site / Job Activity “Check”**

- Company induction
- Site induction
- Industry induction

**Other Training, Please List:**

- Hand tools

### Permits required

**List Permits required for the Job Site prior to commencing work “Check”**

- Isolation & lockout permit
- Excavation permit
- Confined space permit
- Hot work permit
- Permit to work
- Road closure permit
- Other

**Specify:**

**H&S legislation**

- Work, Health and Safety Act 2011
Safe work method statement (SWMS)

Administration Section
Project name: Dev of a coal wagon monitoring and control system
Activity / Task assessed: Preliminary prototype testing

Prepared by: Adam Roberts-Apps
Approved by:

Date
Version

Controls effective
Date of next review: November 2014

Reviewed at first use by:

Work method assessment
(Must be task specific only, no generic or motherhood statements will be accepted)

<table>
<thead>
<tr>
<th>Step</th>
<th>Task Description</th>
<th>Hazard Exact what could cause injury or illness? (one hazard per line)</th>
<th>Consequences What could happen if hazard were not controlled?</th>
<th>Risk Score (SWMS)</th>
<th>Control What are the hierarchy’s used</th>
<th>Risk score Area of control</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Fit instrumentation onto an automotive trailer Drilling Cuttings Torque causing a wrist injury Cutting fingers</td>
<td>Moderate Hand tools training Gloves</td>
<td>Administrate PPE</td>
<td>Low</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Hitch trailer to vehicle and connect rear lights Heavy lifting Pull back out Nerve damage Strains</td>
<td>High Have two or more people helping to lessen the load</td>
<td>Administrate</td>
<td>Low</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Check that everything is secure Lifting Corroded steel Pinch points Crush injuries Tetanus Cuts and grazes</td>
<td>Moderate Wear gloves and watch out for sharp edges</td>
<td>PPE Administrate</td>
<td>Low</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Drive along nominate route for testing Wreck Major injury or death</td>
<td>High Concentrate on driving Licenced drivers only Drive on quiet roads to reduce the chances of having a wreck with another road user</td>
<td>Administrate Eliminate</td>
<td>Low</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Park the trailer and vehicle Potential energy i.e. vehicle rolling away Entrapment Crush injury</td>
<td>Moderate</td>
<td>Administrate Administrate</td>
<td>Low</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Detached and stow equipment, trailer and controller away Heavy lifting Sharp wire ends Trailer tipping over once erect Cuts Muscle tears Impact injury</td>
<td>Moderate Wear gloves Many hands make light work Secure the trailer against the wall</td>
<td>PPE Isolate Administrate</td>
<td>Low</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Note that any hazard that has a consequence of Significant, Serious, Critical or Disastrous is a “Significant Hazard” – NZ only.

Personal protective equipment (P.P.E) required

- Fall arrest equipment
- Steel capped boots
- Hard hat
- Safety glasses
- Sun screen
- Hearing protection
- Respirator
- Gloves
- Disposable overalls
- Hi Visibility Vest

Other PPE, Please List

- Company induction
- Site induction
- Industry induction

Training required

- Other Training, Please List:
  - Hand tools

Permits required

- Isolation & lockout permit
- Excavation permit
- Confined space permit
- Hot work permit
- Permit to work
- Road closure permit
- Other Specify: NSW driver’s license

H&S legislation

- Work, Health and Safety Act 2011
## Safe work method statement (SWMS)

### Risk Rating Matrix

<table>
<thead>
<tr>
<th>Most Likely Consequence</th>
<th>Likelihood</th>
<th>Grade</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disastrous</td>
<td>Very likely will occur</td>
<td>Extreme</td>
<td>Single or multiple fatality</td>
</tr>
<tr>
<td>Critical</td>
<td>Good chance to occur</td>
<td>Extreme</td>
<td>Disabling injury or illness i.e. amputation and / or permanent loss of bodily function, or any kind of permanent health impact</td>
</tr>
<tr>
<td>Serious</td>
<td>Likely to occur</td>
<td>High</td>
<td>Any lost time injury (LTI) resulting in one or more complete days off work or Any (RWI) resulting in more than 1 week off normal duties</td>
</tr>
<tr>
<td>Significant</td>
<td>Unlikely to occur</td>
<td>Moderate</td>
<td>A medical treatment injury (MTI) or A restricted work injury (RWI) i.e. any injury resulting in less than 1 week on alternate duties</td>
</tr>
<tr>
<td>Minor</td>
<td>Very unlikely to occur</td>
<td>Low</td>
<td>Minor First Aid injury or an injury not requiring treatment</td>
</tr>
</tbody>
</table>

### Risk Category

<table>
<thead>
<tr>
<th>Rating</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extreme Risk</td>
<td>Cease the work immediately and notify the most senior line manager responsible for the work. Immediate action required, do not proceed with any work until confirmed safe to do so and re-commencement has been authorised the senior line manager or appropriately qualified and competent person.</td>
</tr>
<tr>
<td>High Risk</td>
<td>Cease the work immediately and notify the most senior line manager responsible for the work. Immediate action required, do not proceed with any work until confirmed safe to do so and re-commencement has been authorised the senior line manager or appropriately qualified and competent person.</td>
</tr>
<tr>
<td>Moderate Risk</td>
<td>Notify Project Leader or Manager and identify control actions and action dates, proceed with work only if confirmed safe to do so.</td>
</tr>
<tr>
<td>Low Risk</td>
<td>Manage by routine procedures.</td>
</tr>
</tbody>
</table>

### Hierarchy of Control

You should attempt remove or control the hazard in the following order:

*If the hazard cannot be eliminated completely then apply any one or a combination controls 2 – 5 in descending order until the work can be done safely.

- **Option 1**
  1. Eliminate
     - Eliminate the hazard at its source i.e. complete removal or termination of whatever is generating the hazard, could be a process, work method, equipment, material or substance etc.

- **Option 2**
  2. Substitute
     - Replace whatever is generating the hazard with a non-hazardous or less hazardous process, work method, equipment, material or substance etc.
  3. Engineer / Isolate
     - Redesign or modify whatever is generating the hazard to control the effects of the hazard or prevent people from coming into contact with it; this includes isolating the hazard to prevent access. Includes all design and engineering solutions (done with engineering solutions, site design and engineering solutions, etc).
  4. Administrative
     - Administrative controls include the use of procedures, training and information, signage, hours of work etc.
  5. Personal Protective Equipment (PPE)
     - Use appropriately designed and properly fitting personal protective equipment where other controls are not practicable. This is not a primary control it is a back up control and should be considered only as a support to the other controls.

---

**Note:** Anything with a consequence of 'Significant' or above needs to be considered as a 'significant hazard' in New Zealand.
Appendix D

OTSI incidents
At 1503 on Wednesday 3 January 2013, El Zorro grain train 5CM7 derailed ten wagons towards the rear of the trains consist near the township of Rennie while travelling between Oaklands and Benalla Figure D.1. There was extensive damage to the rail infrastructure and to the derailed wagons.

Following the derailment a small grass fire broke out requiring the local Rural Fire Service to extinguish it (OTSI 2012a).

At approximately 3.35am on 5 December 2012, Qube Logistics containerised grain service 8448, travelling from Gilgandra to Walsh Island (Newcastle), stopped at Gulgong to facilitate a crew change. As the relieving crew departed Gulgong, the original crew performed a roll-by inspection in accordance with standard safety procedures from the ground beside the Gulgong-Ulan Road level crossing.

As 8448 passed, the crew on the ground noticed that the 20th wagon was sagging in the middle. They immediately called the crew on the train via VHF radio to stop which they did clear of the crossing, beside the platform at Gulgong Station.
The failed wagon, SQDY 00060-G, was loaded with two 20t containers and had a gross mass of 76 tonnes. The wagon was one of a class of 100 wagons, all of which had entered service with Qube within the past 12 months. The wagons were manufactured in the United States of America by American Railcar Industries Inc.

Initial inspection of SQDY 00060-G noted that the sagging was a result of a fracture in the middle of the main structural member of the wagon Figure D.2. The fracture had emanated from a failed welded joint in the bottom plate forming the central longitudinal box section of the member.

Inspection of the rest of the train identified that fracturing was also occurring at the same location in the structural members on a number of other wagons. Damage to level crossings indicated that the wagon had failed approximately 25 kilometres prior (OTSI 2012c).

At 1430 hours on Wednesday 28 November 2012, Pacific National coal train NB942 while travelling over the Coxs Creek Bridge at Boggabri derailed the last six wagons of the trains consist Figure D.3. As a result of the derailment the first five wagons toppled over the bridge and onto the ground below. The last wagon
derailed but remained on the bridge.

Following the derailment a small fire broke out in the grass below the bridge, requiring the attendance of the local Rural Fire Service to extinguish it.

There was extensive damage to the Coxs Creek Bridge and to the six derailed wagons” (OTSI 2012b).

At approximately 11.56pm on 23 November 2011, Pacific National coal service MC92 derailed eight wagons at Clifton. MC92, a 45-wagon train with single locomotives at its front and rear, was fully loaded with coal and was travelling South from the Metropolitan Colliery at Helensburgh to Inner Harbour at Port Kembla where it was to be unloaded. The leading locomotive had just passed Clifton level crossing when an emergency application of the trains brakes occurred automatically and it came to a stand. The rear of the train was still inside the Coalcliff tunnel.

The Driver on the leading locomotive notified the RailCorp Signal Complex at Wollongong that his train was stopped. He then sent his co-driver back to inspect
the train and locate whatever had caused the brakes to apply automatically. The co-driver found that the wagons had derailed North of the level crossing and used his two-way radio to alert the Driver. At 12.08am the Driver notified the RailCorp Signal Complex at Wollongong that the wagons had derailed Figure D.4.

The investigation revealed that the barrel of the No. 3 axle of the eighth position wagon had broken and parted, causing both wheels to derail. As a result, seven wagons following this wagon derailed. The two locomotives and all other wagons remained on the track. Although there were no injuries as a result of the derailment, approximately 470 metres of damaged track needed to be replaced.

The investigation established that the break in the axle was attributable to the propagation of metal fatigue at the site of the fracture. The fatigue fracture was initiated some time prior to the final complete failure of the axle at the derailment site, but due to damage sustained in the derailment, the initiator of the fracture could not be determined (OTSI 2011).

The brief summary above only includes those incidents which occurred in NSW and were reported to the OTSI. There were fifteen other cases of a similar nature not mentioned dating back to 2002. One could only imagine the total number
of incidents if the study examined all states and territories in Australia. As the assets associated with the rail freight industry become older, failure rates are only going to become higher, unless the assets are either replaced for new or maintenance is done more frequently.

Making large capital purchases purely to minimise risk against uncertainty is an expensive exercise if one can avoid it, and the same theory applies to maintenance. Clearly the owner/operator of these assets needs a system than can give them some pre-warning before failure occurs.

Visual inspection and non-destructive testing programs are some of the ways defects can be detected before catastrophic failure occurs (Barnett 2004). Performing these activities normally decreases the risk (if the defects are detected!); however increases the downtime required so inspection personnel can safely and thoroughly perform their duties.

It would be convenient and less costly if the inspections and testing could be performed in-situ, automated and continuous. In order to achieve such a feat, the requirement for people to perform these duties needs to be taken out of the equation. Hence, there is a potential market to develop a device that can be permanently mounted to the wagon and perform the tasks humans would otherwise do and perhaps more frequently.
Appendix E

FMECA
<table>
<thead>
<tr>
<th>Ref.</th>
<th>Sub-system</th>
<th>Failure mode</th>
<th>Potential effects</th>
<th>Potential controls or pre-warning</th>
<th>Existing controls</th>
<th>Recommended action</th>
<th>Revised controls</th>
<th>Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Wheels</td>
<td>Even rim wear</td>
<td>Reduced dead-reckoning causing the wheel to ride up upon the wheel flange - possible derailment</td>
<td>Ride height from axle centre to the top of the rail</td>
<td>8 2 9 144</td>
<td>Mount displacement sensor and monitor change in ride height from the top of the rail</td>
<td>8 1 5 40 104</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Uneven rim wear</td>
<td>Shimming causing higher lateral cyclic loading fatigue</td>
<td>Angle across the bolster and track oscillations over time at the centrebowl</td>
<td></td>
<td>3 5 9 135</td>
<td>Encoder mounted at the centrebowl and monitor oscillations over time</td>
<td>3 4 5 60 75</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Buckled or damaged flanges</td>
<td>Derailment or high shock loading</td>
<td>1x vibration</td>
<td></td>
<td>2 9 8 144</td>
<td>Mount accelerometer on the bearing assembly and measure shaft speed to order normalise data</td>
<td>2 8 7 112 32</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Flat spots</td>
<td>Shock loading and increased noise</td>
<td>Nx vibration where N = the number of flat spots</td>
<td></td>
<td>7 4 7 196</td>
<td>Mount accelerometer on the bearing assembly and measure shaft speed to order normalise data and control brakes i.e. ABS</td>
<td>7 3 3 63 133</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Sprawling fatigue</td>
<td>Rough ride and increased vibrations i.e. increases fatigue rates and potential secondary damage</td>
<td>Rough vibration signal in the time domain</td>
<td></td>
<td>6 4 8 192</td>
<td>Measure overall vibration level on the wheel or non-contact displacement sensor at the top of the wheel and track spike over time.</td>
<td>6 3 4 72 120</td>
<td></td>
</tr>
</tbody>
</table>

Criticality code:
Occ = Prob. of occurrence x 10
Sev = Severity of effect on 1 - 1- scale
Det = Prob. not detected x 10
APN = Action Priority Number Occ x Sev x Det
<table>
<thead>
<tr>
<th>Ref.</th>
<th>Sub-system</th>
<th>Failure mode</th>
<th>Potential effects</th>
<th>V</th>
<th>Potential controls or pre-warning</th>
<th>Existing controls Occ Sev Det RPN</th>
<th>Recommended action</th>
<th>Revised controls Occ Sev Det APN</th>
<th>Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Axle</td>
<td>Cocked fit</td>
<td>Axial vibration</td>
<td></td>
<td>1x, 2x and higher 3x vibration</td>
<td>1 3 10 30</td>
<td>Mount accelerometer on the bearing assembly and measure shaft speed to order normalise data</td>
<td>1 2 8 16 14</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Loose fit</td>
<td>Wheel coming off the axle causing derailment</td>
<td>Vibration</td>
<td>1 9 5 45</td>
<td>Measure overall vibration levels</td>
<td>1 8 5 40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Axle</td>
<td>Fatigue cracking</td>
<td>Increased stress concentration and higher risk of the shaft shearing causing derailment</td>
<td>Increased vibration at later stages of failure when there would significant shaft deflections. NDT shaft during maintenance intervals or in-situ shaft test using ultrasonic</td>
<td>6 10 6 360</td>
<td>Measure shaft vibration using a proxy probe</td>
<td>6 9 5 270</td>
<td>90</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Worn journals</td>
<td>Wheel coming off the axle</td>
<td>Gauging journals during maintenance interval. Possible fretting characteristic vibrations</td>
<td>4 10 6 240</td>
<td>FFT spectrum vibration analysis</td>
<td>4 9 5 180</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bowed shaft</td>
<td>High radial vibration/fatigue</td>
<td>Radial deflection of shaft from rotational centre and axial displacement of wheels</td>
<td>3 1 8 24</td>
<td>Accelerometer mounted onto the bearing assembly</td>
<td>3 1 6 18</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

**Criticality code:**

Occ = Prob. of occurrence x 10  
Sev = Severity of effect on 1 - 1- scale  
Det = Prob. not detected x 10  
APN = Action Priority Number Occ x Sev x Det
<table>
<thead>
<tr>
<th>Ref.</th>
<th>Sub-system</th>
<th>Failure mode</th>
<th>Potential effects</th>
<th>V</th>
<th>Potential controls or pre-warning</th>
<th>Existing controls</th>
<th>Recommended action</th>
<th>Revised controls</th>
<th>Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Rolling element bearing and adapter</td>
<td>Race damage/fatigue</td>
<td>Bearing vibration and noise with increase in bearing temperature if sprawling is severe. Potential for the bearing to seize causing secondary damage to the shaft of wheel</td>
<td></td>
<td>Vibration frequencies at the BPFI and BPFO frequencies. Increased bearing temperature at later stages of failure</td>
<td>9 7 7 441</td>
<td>Accelerometer and temperature sensor mounted onto the bearing assembly</td>
<td>9 6 2 108 333</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Incorrect or lack of lubrication</td>
<td></td>
<td>Bearing noise elevated bearing temperatures - potential to cause the wheel to lock creating a large flat spot or heat up enough to an ignition source for a fire</td>
<td></td>
<td>High frequency raised noise floor in the frequency domain and increased bearing temperatures</td>
<td>4 7 7 196</td>
<td>Accelerometer and temperature sensor mounted onto the bearing assembly</td>
<td>4 6 2 48 148</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Out of round balls or rollers</td>
<td></td>
<td>Bearing lock-up as above</td>
<td></td>
<td>Vibration at the 2x BSF frequency</td>
<td>4 7 7 196</td>
<td>Accelerometer and temperature sensor mounted onto the bearing assembly</td>
<td>4 6 2 48 148</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cracked cage</td>
<td></td>
<td>Ball/roller dislodging seizing the bearing - causing extreme heat or fire and potential shaft damage</td>
<td></td>
<td>Vibration at the FTF frequency</td>
<td>2 7 7 98</td>
<td>Accelerometer and temperature sensor mounted onto the bearing assembly</td>
<td>2 6 3 36 62</td>
<td></td>
</tr>
</tbody>
</table>

Criticality code:
Occ = Prob. of occurrence x 10
Sev = Severity of effect on 1 - 1- scale
Det = Prob. not detected x 10
APN = Action Priority Number Occ x Sev x Det
<table>
<thead>
<tr>
<th>Ref.</th>
<th>Sub-system</th>
<th>Failure mode</th>
<th>Potential effects</th>
<th>Potential controls or pre-warning</th>
<th>Existing controls</th>
<th>Recommended action</th>
<th>Revised controls</th>
<th>Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Load springs</td>
<td>Wear</td>
<td>Increased friction reduced fuel efficiency</td>
<td>Monitor fuel consumption</td>
<td>5 7 7 245</td>
<td>Accelerometer and temperature sensor mounted onto the bearing assembly</td>
<td>5 6 4 120</td>
<td>125</td>
</tr>
<tr>
<td>4</td>
<td>Load springs</td>
<td>Shearing</td>
<td>Reduced suspension performance</td>
<td>Displacement over time</td>
<td>2 4 7 56</td>
<td>Mount non-contact displacement sensor at the bottom of the bolster and point it towards the top of the spring recess on the side rails</td>
<td>2 3 4 24</td>
<td>32</td>
</tr>
<tr>
<td>6</td>
<td>Fatigue cracking</td>
<td>Fatigue cracking</td>
<td>Change in spring coefficient</td>
<td>Change typical spring displacement from loaded/unloaded state</td>
<td>6 3 10 180</td>
<td>Mount non-contact displacement sensor at the bottom of the bolster and point it towards the top of the spring recess on the side rails</td>
<td>6 2 6 72</td>
<td>108</td>
</tr>
<tr>
<td>1</td>
<td>Eccentric loading/buckling</td>
<td>Eccentric loading/buckling</td>
<td>Change in spring coefficient</td>
<td>Compare the spring displacement of other three sets</td>
<td>1 3 10 30</td>
<td>Mount non-contact displacement sensor at the bottom of the bolster and point it towards the top of the spring recess on the side rails</td>
<td>1 2 4 8</td>
<td>22</td>
</tr>
<tr>
<td>1</td>
<td>Damaged spring ends</td>
<td>Damaged spring ends</td>
<td>Uneven loading</td>
<td>Compare spring end loads</td>
<td>1 2 6 12</td>
<td>Mount non-contact displacement sensor at the bottom of the bolster and point it towards the top of the spring recess on the side rails</td>
<td>1 1 4 4</td>
<td>8</td>
</tr>
</tbody>
</table>

Criticality code:
Occ = Prob. of occurrence x 10
Sev = Severity of effect on 1 - 1- scale
Det = Prob. not detected x 10
APN = Action Priority Number Occ x Sev x Det
<table>
<thead>
<tr>
<th>Ref.</th>
<th>Sub-system</th>
<th>Failure mode</th>
<th>Potential effects</th>
<th>Potential controls or pre-warning</th>
<th>Existing controls</th>
<th>Recommended action</th>
<th>Revised controls</th>
<th>Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>Occ</td>
<td>Sev</td>
<td>Det</td>
<td>RPN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>Brake beams</td>
<td>Bowed members</td>
<td>Uneven brake force</td>
<td>Process brake pressure data and compare to other units</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td>6</td>
<td>Side frame</td>
<td>Fatigue cracking</td>
<td>Reduced service life</td>
<td>NDT inspection and strain gauging</td>
<td>8</td>
<td>9</td>
<td>4</td>
<td>288</td>
</tr>
<tr>
<td>7</td>
<td>Bolster</td>
<td>Fatigue cracking</td>
<td>Reduced service life</td>
<td>NDT inspection and strain gauging</td>
<td>8</td>
<td>9</td>
<td>5</td>
<td>360</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>

Criticality code:
Occ = Prob. of occurrence x 10
Sev = Severity of effect on 1 - 1- scale
Det = Prob. not detected x 10
APN = Action Priority Number Occ x Sev x Det
<table>
<thead>
<tr>
<th>Ref.</th>
<th>Sub-system</th>
<th>Failure mode</th>
<th>Potential effects</th>
<th>V</th>
<th>Potential controls or pre-warning</th>
<th>Existing controls</th>
<th>Recommended action</th>
<th>Revised controls</th>
<th>Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Centrebowl</td>
<td>Sprawling fatigue damage</td>
<td>Jamming of the joint</td>
<td>8</td>
<td>4</td>
<td>7</td>
<td>224</td>
<td>Encoder mounted at the centre bowl and compare reading with other bogie angle</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Side bearers</td>
<td>Flat spots on the rollers</td>
<td>Increased friction on the bottom of the hopper chassis</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>12</td>
<td>Install an EC sensor in the Centrebowl</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Seized roller</td>
<td>Increased friction on the bottom of the hopper chassis</td>
<td>7</td>
<td>1</td>
<td>5</td>
<td>35</td>
<td>Collect thermal images of hot spots</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cracked/sheared pin</td>
<td>Uneven load share</td>
<td>8</td>
<td>1</td>
<td>5</td>
<td>40</td>
<td>Collect thermal images of hot spots</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Hopper</td>
<td>Structural failures</td>
<td>Coal spillage, derailment or track damage</td>
<td>6</td>
<td>10</td>
<td>7</td>
<td>420</td>
<td>Strain gauge rosette mounted on the key stress points</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dump doors not closing or opening properly</td>
<td>Pay load for a given wagon not delivered</td>
<td>8</td>
<td>4</td>
<td>5</td>
<td>160</td>
<td>Limit switch hinge and install load cell on the door jam</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Empty wagon i.e. missed at loading station</td>
<td>Increased risk of derailment during significant braking events</td>
<td>2</td>
<td>7</td>
<td>8</td>
<td>112</td>
<td>Mount non-contact displacement sensor at the bottom of the bolster and point it towards the top of the spring recess on the side rails</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Air lines</td>
<td>Split pipe</td>
<td>Uncontrolled application of the brake</td>
<td>7</td>
<td>5</td>
<td>8</td>
<td>280</td>
<td>Sound and pressure monitoring</td>
<td>7</td>
</tr>
</tbody>
</table>

Criticality code:
Occ = Prob. of occurrence x 10
Sev = Severity of effect on 1 - 1- scale
Det = Prob. not detected x 10
APN = Action Priority Number Occ x Sev x Det
<table>
<thead>
<tr>
<th>Ref.</th>
<th>Sub-system</th>
<th>Failure mode</th>
<th>Potential effects</th>
<th>Potential controls or pre-warning</th>
<th>Existing controls Occ Sev Det RPN</th>
<th>Recommended action</th>
<th>Revised controls Occ Sev Det APN</th>
<th>Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>Worn connection</td>
<td>High compressor loads</td>
<td>Brake pipe pressure</td>
<td>8 3 9 216</td>
<td>Sound and pressure monitoring</td>
<td>8 2 5 80</td>
<td>136</td>
</tr>
<tr>
<td>12</td>
<td>Air receiver</td>
<td>Internal pitting/corrosion causing a leak</td>
<td>Unable to maintained pressure</td>
<td>Receiver pressure</td>
<td>4 2 6 48</td>
<td>Sound and pressure monitoring</td>
<td>4 1 6 24</td>
<td>24</td>
</tr>
<tr>
<td>13</td>
<td>Brake shoe</td>
<td>Accessibly worn</td>
<td>Loss of pressure</td>
<td>Receiver pressure</td>
<td>1 2 5 10</td>
<td>Sound and pressure monitoring</td>
<td>1 1 5 5</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Larger displacements of the brake actuator resulting in delayed brake application</td>
<td>Brake cylinder pressure v deceleration</td>
<td>6 2 6 72</td>
<td>General purpose accelerometer wagon body and cross-referenced with the pressure data</td>
<td>6 1 5 30</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Reduced coefficient of friction resulting in longer stopping distances</td>
<td>Brake cylinder pressure v deceleration</td>
<td>4 3 8 96</td>
<td>General purpose accelerometer wagon body and cross-referenced with the pressure data</td>
<td>4 2 5 40</td>
<td>56</td>
</tr>
</tbody>
</table>

Criticality code:
Occ = Prob. of occurrence x 10
Sev = Severity of effect on 1 - 1- scale
Det = Prob. not detected x 10
APN = Action Priority Number Occ x Sev x Det
Appendix F

Coal Wagon information
F.1 SKF rolling element bearings
**Bearing Frequencies Calculator**

This calculator is used to calculate the different bearing defect frequencies of bearing applications. One can search for an existing SKF bearing or input the different bearing parameters manually. The bearing defect frequencies can be displayed in Hertz, CPM or in orders of the rotational speed.

<table>
<thead>
<tr>
<th>Bearing Data</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>SKF bearing designation*</td>
<td><img src="image" alt="Image of SKF bearing designation" /></td>
</tr>
<tr>
<td>Measurement system</td>
<td><img src="image" alt="Image of Measurement system" /></td>
</tr>
<tr>
<td>Bearing type*</td>
<td><img src="image" alt="Image of Bearing type" /></td>
</tr>
<tr>
<td>Pitch diameter*</td>
<td><img src="image" alt="Image of Pitch diameter" /></td>
</tr>
<tr>
<td>Rolling element diameter*</td>
<td><img src="image" alt="Image of Rolling element diameter" /></td>
</tr>
<tr>
<td>Number of rolling elements (per row)*</td>
<td><img src="image" alt="Image of Number of rolling elements" /></td>
</tr>
<tr>
<td>Contact angle*</td>
<td><img src="image" alt="Image of Contact angle" /></td>
</tr>
<tr>
<td>Rotational speed*</td>
<td><img src="image" alt="Image of Rotational speed" /></td>
</tr>
<tr>
<td>Rotating ring*</td>
<td><img src="image" alt="Image of Rotating ring" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><img src="image" alt="Image of SKF bearing designation" /></th>
<th><img src="image" alt="Image of SKF bearing designation" /></th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Image of Measurement system" /></td>
<td><img src="image" alt="Image of Measurement system" /></td>
</tr>
<tr>
<td><img src="image" alt="Image of Bearing type" /></td>
<td><img src="image" alt="Image of Bearing type" /></td>
</tr>
<tr>
<td><img src="image" alt="Image of Pitch diameter" /></td>
<td><img src="image" alt="Image of Pitch diameter" /></td>
</tr>
<tr>
<td><img src="image" alt="Image of Rolling element diameter" /></td>
<td><img src="image" alt="Image of Rolling element diameter" /></td>
</tr>
<tr>
<td><img src="image" alt="Image of Number of rolling elements" /></td>
<td><img src="image" alt="Image of Number of rolling elements" /></td>
</tr>
<tr>
<td><img src="image" alt="Image of Contact angle" /></td>
<td><img src="image" alt="Image of Contact angle" /></td>
</tr>
<tr>
<td><img src="image" alt="Image of Rotational speed" /></td>
<td><img src="image" alt="Image of Rotational speed" /></td>
</tr>
<tr>
<td><img src="image" alt="Image of Rotating ring" /></td>
<td><img src="image" alt="Image of Rotating ring" /></td>
</tr>
</tbody>
</table>

**Output**
- ![Image of Hertz](image)
- ![Image of CPM](image)
- ![Image of Orders](image)
- Shaft speed frequency: 1.000
- Inner race defect frequency (BRFI): 27.305
- Outer race defect frequency (BIFO): 24.694
- Cage defect frequency (FTF): 0.475
- Ball spin frequency (BSF): 9.528
- Rolling element defect frequency: 19.056

**Figure F.1:** Bearing fault frequencies for a 177 mm ID tapper roller
F.3 Braken wheel set
F.4 Typical draft gear design
F.5 Bogie spring deflection graph
Wagon Type: COAL HOPPER  
Class: NHRH  

<table>
<thead>
<tr>
<th>Fleet Data:</th>
<th>Operational Data:*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous Classes:</td>
<td>Route availability:</td>
</tr>
<tr>
<td>-</td>
<td>ROA Plate B</td>
</tr>
<tr>
<td>Number in Class: 399 (57 x 7 packs)</td>
<td>Unit length (Over Couplers): 113500 mm (7 wagons)</td>
</tr>
<tr>
<td>Date first built: 1992 EPT</td>
<td>Tare mass: 23 tonne</td>
</tr>
<tr>
<td>Usage: Coal</td>
<td>Gross mass: 120 tonne</td>
</tr>
<tr>
<td>Volumetric Capacity:</td>
<td>Payload capacity: 97 tonne</td>
</tr>
<tr>
<td>Comments:</td>
<td>Maximum speed: 100 km/h empty</td>
</tr>
</tbody>
</table>

*Note: Not for operational purposes. Refer Wagon Data for Train Operations

---

Wagon Details Manual

Reference Number: Revision Number
WDM-NHRH_02

Issue No. 2  Issue Date: 22nd September 2003  All printed copies are uncontrolled

Approved By: Wagon Fleet Manager, Operation Services  Section WDM, Page 1 of 2
WAGON DETAILS

GENERAL
Designer: EPT
Builder: EPT
Built For: FreightCorp
Build Date: 1992
Built As: -
Drg No (GA): EPT drgs
AO 15314 Trailing wagon
AO 15313 Intermediate wagon
AO 15312 Lead Wagon
Drg List: A1 21295
Modifications: None
Dates of Mods: -

BOGIES
Bogie Type: 3 piece steering
Std Designation: EAT
Arrangement drg: -
Drawing list: -
Axle load: 30 tonne
Load Springs: Group D7 (9 outer, 5 inner)
Wedge Springs: RC274
Wheel Diameter: 920 mm
Journal Centres: 2006 mm
Wheelbase: 1600 mm
Centre Plate: Flat 406 mm diameter
Journal Bearings: 7" x 12" "G" class
Brake lever ratio: 4:1 side pull
Side Bearers: CCSB
Ride Control / Barber

BRAKES
Brake cylinder: 356 x 305 WF
Mounting: Body
Relay valve: 24/50 WHB
Triple valve: WF4-6M WHB
Load comp.: Automatic VTA valve
Grade control: Fixed exhaust choke
Slack adjuster: DRV2A-600 SAB
Park brake: Transverse
Brake block: High friction
BP bifurcated: No
Brake % tare: 55%
Brake % gross: 22%
Brake % park: Not available

NOTE: Brake % calculated with no losses.
Main reservoir: Yes
“B” wagon: All

COUPLERS
Draftgear spec: AAR M-901 E
Coupler type: A335 Ultra
Coupler length: 680 mm
Drawbar type: Slackless
Drawbar length: -
Artic connector: N/A
Appendix G

Electronic schematics
Appendix H

Third party source code
// complex.h - declaration of class of complex number
// The code is property of LIBROW

#ifndef _COMPLEX_H_
#define _COMPLEX_H_

class complex
{
protected:
   // Internal presentation - real and imaginary parts
   double m_re;
   double m_im;

public:
   // Imaginary unity
   static const complex i;
   static const complex j;

   // Constructors
   complex(): m_re(0.), m_im(0.) {}
   complex(double re, double im): m_re(re), m_im(im) {}
   complex(double val): m_re(val), m_im(0.) {}

   // Assignment
   complex& operator= (const double val)
   {
      m_re = val;
      m_im = 0.;
return *this;

// Basic operations - taking parts
double re() const { return m_re; }
double im() const { return m_im; }

// Conjugate number
complex conjugate() const
{
    return complex(m_re, -m_im);
}

// Norm
double norm() const
{
    return m_re * m_re + m_im * m_im;
}

// Arithmetic operations
complex operator+ (const complex& other) const
{
    return complex(m_re + other.m_re, m_im + other.m_im);
}

complex operator- (const complex& other) const
{
    return complex(m_re - other.m_re, m_im - other.m_im);
}
complex operator* (const complex& other) const
{
    return complex(m_re * other.m_re - m_im * other.
    m_im,
    m_re * other.m_im + m_im * other.m_re);
}

complex operator/ (const complex& other) const
{
    const double denominator = other.m_re * other.m_re
    + other.m_im * other.m_im;
    return complex((m_re * other.m_re + m_im * other.
    m_im) / denominator,
    (m_im * other.m_re - m_re * other.m_im) / 
    denominator);
}

complex& operator+= (const complex& other)
{
    m_re += other.m_re;
    m_im += other.m_im;
    return *this;
}

complex& operator-= (const complex& other)
{
    m_re -= other.m_re;
    m_im -= other.m_im;
    return *this;
}

complex& operator*=(const complex& other)
H.1 C++ complex number handler

{
    const double temp = m_re;
    m_re = m_re * other.m_re - m_im * other.m_im;
    m_im = m_im * other.m_re + temp * other.m_im;
    return *this;
}

complex& operator/=(const complex& other)
{
    const double denominator = other.m_re * other.m_re
        + other.m_im * other.m_im;
    const double temp = m_re;
    m_re = (m_re * other.m_re + m_im * other.m_im) /
        denominator;
    m_im = (m_im * other.m_re - temp * other.m_im) /
        denominator;
    return *this;
}

complex& operator++()
{
    ++m_re;
    return *this;
}

complex operator++(int)
{
    complex temp(*this);
    ++m_re;
    return temp;
}
complex& operator-- ()
{
    --m_re;
    return *this;
}

complex operator-- (int)
{
    complex temp(*this);
    --m_re;
    return temp;
}

complex operator+ (const double val) const
{
    return complex(m_re + val, m_im);
}

complex operator- (const double val) const
{
    return complex(m_re - val, m_im);
}

complex operator* (const double val) const
{
    return complex(m_re * val, m_im * val);
}

complex operator/ (const double val) const
{
    return complex(m_re / val, m_im / val);
}
complex& operator+= (const double val)
{
    m_re += val;
    return *this;
}

complex& operator-= (const double val)
{
    m_re -= val;
    return *this;
}

complex& operator*= (const double val)
{
    m_re *= val;
    m_im *= val;
    return *this;
}

complex& operator/= (const double val)
{
    m_re /= val;
    m_im /= val;
    return *this;
}

friend complex operator+ (const double left, const complex& right)
{
    return complex(left + right.m_re, right.m_im);
}
friend complex operator- (const double left, const complex& right)
{
    return complex(left - right.m_re, -right.m_im);
}

friend complex operator* (const double left, const complex& right)
{
    return complex(left * right.m_re, left * right.m_im);
}

friend complex operator/ (const double left, const complex& right)
{
    const double denominator = right.m_re * right.m_re
    + right.m_im * right.m_im;
    return complex(left * right.m_re / denominator, -left * right.m_im / denominator);
}

// Boolean operators
bool operator== (const complex &other) const
{
    return m_re == other.m_re && m_im == other.m_im;
}

bool operator!= (const complex &other) const
{
    return m_re != other.m_re || m_im != other.m_im;
bool operator==(const double val) const
{
    return m_re == val && m_im == 0.;
}

bool operator!=(const double val) const
{
    return m_re != val || m_im != 0.;
}

friend bool operator==(const double left, const complex & right)
{
    return left == right.m_re && right.m_im == 0.;
}

friend bool operator!=(const double left, const complex & right)
{
    return left != right.m_re || right.m_im != 0.;
}

#endif

// complex.cpp - implelementation of class of complex number

// The code is property of LIBROW

// Include header file
#include "complex.h"

// Imaginary unity constants
const complex complex::i(0., 1.);
const complex complex::j(0., 1.);
import ddf.minim.analysis.*;
import ddf.minim.*;

Minim minim;
AudioPlayer jingle;
FFT fftLin;
FFT fftLog;
float height3;
float height23;

void setup()
{
    size(512, 300, P3D);
    height3 = height/3;
    height23 = 2*height/3;

    minim = new Minim(this);
jingle = minim.loadFile("jingle.mp3", 2048);
    // loop the file
    jingle.loop();
    // create an FFT object that has a time-domain buffer
    the same size as jingle’s sample buffer
    // note that this needs to be a power of two
    // and that it means the size of the spectrum will be
    1024.
    // see the online tutorial for more info.
    fftLin = new FFT(jingle.bufferSize(), jingle.
        sampleRate());
// calculate the averages by grouping frequency bands linearly. Use 30 averages.
fftLin.linAverages(30);
fftLog = new FFT(jingle.bufferSize(), jingle.
sampleRate());
// calculate averages based on a minimum octave width of 22 Hz
// split each octave into three bands
// this should result in 30 averages
fftLog.logAverages(22, 3);
rectMode(CORNERS);

void draw()
{
  background(0);
  // perform a forward FFT on the samples in jingle’s mix buffer
  // note that if jingle were a MONO file, this would be the same as using jingle.left or jingle.right
  fftLin.forward(jingle.mix);

  stroke(255);
  noFill();
  // draw the full spectrum
  for(int i = 0; i < fftLin.specSize(); i++)
  {
    line(i, height3, i, height3 - fftLin.getBand(i)*2);
  }

  noStroke();
  fill(255);
// draw the linear averages
int w = int(width/fftLin.avgSize());
for(int i = 0; i < fftLin.avgSize(); i++)
{
    // draw a rectangle for each average, multiply the value by 5 so we can see it better
    rect(i*w, height23, i*w + w, height23 - fftLin.getAvg(i)*5);
}

// draw the logarithmic averages
fftLog.forward(jingle.mix);
w = int(width/fftLog.avgSize());
for(int i = 0; i < fftLog.avgSize(); i++)
{
    // draw a rectangle for each average, multiply the value by 5 so we can see it better
    rect(i*w, height, i*w + w, height - fftLog.getAvg(i));
}
}

void stop()
{
    // always close Minim audio classes when you are done with them
    jingle.close();
    // always stop Minim before exiting
    minim.stop();

    super.stop();
}
from random import normalvariate

# "Real world" that we're trying to track
class RealWorld:
    def __init__(self):
        self.position = 0.0
        self.velocity = 0.5
        self.time_step = 0.01
        self.time = 0.0
        self.measure = None

        # Noise on the measurements
        self.measurement_variance = 3.0

        # If we want to kink the profile.
        self.change_after = 5
        self.changed_rate = -0.5

    def measure(self):
        if self.measure == None:
            self.measure = (self.position
                        + normalvariate(0,
                        measurement_variance))
        return self.measure

    def step(self):
        self.time += self.time_step
        self.position += self.velocity * self.time_step
if self.time >= self.change_after:
    self.rate_of_change = self.changed_rate

world = RealWorld()

# Model
# Estimates
estimate_position = 0.0
estimate_velocity = 0.0

# Covariance matrix
P_xx = 0.1 # Variance of the temperature
P_xv = 0.1 # Covariance of temperature and rate.
P_vv = 0.1 # Variance of the rate

# Model parameters
position_process_variance = 0.01
velocity_process_variance = 0.01
R = 30.0 # Measurement noise variance

average_length = 30
data = []

for i in range(10000):
    world.step()
    measurement = world.measurement()

    # We need to boot strap the estimates for temperature and
    # rate
if i == 0:  # First measurement
    estimate_position = measurement

elif i == 1:  # Second measurement
    estimate_velocity = (measurement - estimate_position) / world.time_step
    estimate_position = measurement

else:  # Can apply model
    # Temporal update (predictive)
    estimate_position += estimate_velocity * world.time_step

    # Update covariance
    P_xx += world.time_step * (2.0 * P_xv + time_step * P_vv)
    P_xv += world.time_step * P_vv

    P_xx += world.time_step *
    position_process_variance
    P_vv += world.time_step *
    velocity_process_variance

    # Observational update (reactive)
    vi = 1.0 / (P_xx + R)
    kx = P_xx * vi
    kv = P_xv * vi

    estimate_position += (measurement - estimate_position) * kx
    estimate_velocity += (measurement - estimate_position) * kv

    P_xx *= (1 - kx)
\[
P_{xv} *= (1 - k_x) \\
\text{P}_{vv} -= k_v * \text{P}_{xv}
\]

```python
print world.time, world.position, measurement, \
    estimate_position, estimate_velocity
```