

Copyright is owned by the Author of the thesis. Permission is given for a copy to be downloaded by an individual for the purpose of research and private study only. The thesis may not be reproduced elsewhere without the permission of the Author.



**MASSEY UNIVERSITY**  
**ENGINEERING**

**SCHOOL OF ENGINEERING AND  
ADVANCED TECHNOLOGY**

**HARDWARE AND SOFTWARE DEVELOPMENT  
TOWARDS LAMENESS DETECTION OF  
CATTLE**

A thesis presented in partial fulfilment of the requirements  
for the degree of

Master of Engineering  
in  
Mechatronics

at Massey University, Manawatu, New Zealand.

**Johann Nel**

**2015**

**SUPERVISORS**

- a. Gourab Sen Gupta
- b. Ken Mercer
- c. John Gawith

## Abstract

A platform comprising four individual sections has been designed and built to determine a dairy cow's weight, hooves' position, the duration each hoof is on the section, and the stride length. The developed hardware and software is geared towards building a complete system to detect lameness in cattle, the ultimate aim of the project. Each section is an independent unit and consists of four ASB1000 shearbeam load cells, an AD7193 which is a 24-bit sigma-delta analogue-to-digital converter (ADC), and an ATmega328 microcontroller. The AD7193 ADC communicates with the microcontroller via the serial peripheral interface (SPI). Because each section contains its own microcontroller, an Arduino Mega 2560 has been used as the master microcontroller. This handles communication between the computer and all the sections. The master and sections communicate on a RS-485 half-duplex bus. The load cell values are transmitted from the master microcontroller to the computer via serial communication. The individual load cell value is then recorded and further processed where the data can be plotted, and the cow's average weight, stride length, hooves' position and duration can be calculated. The user also has the ability to render the data to a video file and to split cow data.

Laboratory testing was conducted to find the accuracy of the sections using a laser cut jig and a 20kg point load calibration weight. It was found that the X-position mean error is  $1.0 \pm 2.2\text{mm}$ , the Y-position mean error is  $0.8 \pm 1.8\text{mm}$ , and the total weight on the section has a maximum error of 0.4%. The mainframe to which the sections bolt to is 3000mm long and 540mm wide, while the individual sections measure 650mm long by 500mm wide. When the platform is assembled, the platform is 100mm high and has a walking surface width of 400mm. The platform sections are adjustable between the ranges of  $700 \pm 50\text{mm}$  to find the optimal stride length. The platform has been galvanized for protection against the elements. Experimental field testing was conducted at Massey Dairy Farm Number 1 where the signal signatures of 60 cows were recorded for further analysis. The recorded data was used as the basis for all the software tools that were developed; more field testing would be required to make the software more robust to different cow behaviours to see whether cow's weight, hoof position, duration of each hoof and stride length can be successfully and accurately calculated.

## Acknowledgements

I would like to express my very great appreciation to Associate Professor Gourab Sen Gupta and Ken Mercer from the School of Engineering and Advanced Technology at Massey University for their valuable and constructive suggestions during the planning and development of this project. I would also like to thank John Gawith for his valuable ideas on the platform design. The willingness of my supervisors to give their time so generously has been very much appreciated.

Secondly I would like to thank Aaron Dalbeth (Honours student also working on the project) who contributed towards the mechanical, electronics design and manufacturing.

I would also like to thank the workshop technicians from Massey University, and Tru-Test for supplying all the resources necessary to develop this project.

Finally, I wish to thank my family and friends for their support and encouragement throughout my studies.

## Table of Contents

Abstract .....	I
Acknowledgements .....	II
Chapter 1 Introduction and Project Aims .....	1
1.1. Overall Project Aims .....	1
1.2. Current Project Aims .....	2
1.3. Project Objectives .....	2
Chapter 2 Preliminary Survey and Research .....	3
2.1. Lameness Research .....	3
2.1.1. The New Zealand Dairy Industry .....	3
2.1.2. Identifying Lameness .....	3
2.1.3. Cause of Lameness .....	4
2.1.4. Cost of Lameness .....	4
2.2. Current Lameness Detection Systems .....	5
2.2.1. The GAITWISE System .....	5
2.2.2. The StepMetrix System .....	6
2.2.3. Intellectual Property .....	7
2.3. Technical Research .....	8
2.3.1. Tru-Test Products .....	8
2.3.2. Ground Reaction Forces .....	9
2.3.3. Load cell .....	9
2.3.4. Communication Interfaces .....	11
2.3.5. Serial Peripheral Interface Bus .....	13
2.3.6. I <sup>2</sup> C Interface .....	15
2.3.7. Averaging Techniques .....	16
Chapter 3 Electronic System Design and Development .....	18
3.1. Specifications .....	18
3.2. Functional Blocks of the System .....	18
3.2.1. Resources Required for Development .....	19
3.3. Electronic Development .....	20
3.3.1. Objective .....	20
3.3.2. Initial Prototype .....	20
3.3.3. Communication .....	30
3.3.4. Final Prototype PCB .....	34
Chapter 4 Software Development .....	40
4.1. Microcontroller Programming .....	40

4.1.1. AD7193 Programming.....	40
4.1.2. MATLAB Programming.....	48
4.1.3. C++ Programming .....	56
4.1.4. Python Programming .....	62
Chapter 5 Mechanical Design and Development.....	91
5.1. Specifications .....	91
5.2. Initial Prototype Design .....	91
5.2.1. Testing.....	92
5.3. Final Prototype Platform Design.....	98
5.4. Manufacturing of Components.....	99
5.5. Assembly and Integration.....	100
Chapter 6 Experiments and Results .....	103
6.1. Laboratory Testing .....	103
6.1.1. Calculating the Total Weight on the Platform .....	103
6.1.2. Impulse Testing.....	104
6.1.3. AD7193 Averaging.....	105
6.1.4. Platform Accuracy .....	106
6.1.5. Dynamic Response.....	107
6.1.6. Human Walking Signals .....	108
6.1.7. Field Testing .....	109
6.1.8. Validation of Average Weight Algorithms.....	111
Chapter 7 Recommendations and Future Work.....	113
7.1. Electronic Improvements .....	113
7.2. Software Improvements .....	114
7.3. Mechanical Improvements .....	114
Chapter 8 Conclusions .....	116
References.....	118
Appendices.....	121
Appendix 1: Critical Component Datasheets.....	121
AD7193 Datasheet .....	121
REF5040 Datasheet .....	122
AD8656 Datasheet .....	123
MAX487 Datasheet .....	124
ATmega328 Datasheet.....	125
ASB1000 Datasheet .....	126
Appendix 2: Experimental Results .....	127
Load Cell Calibration Experiment .....	127

Load Cell Serial Numbers and Positions .....	128
Test Results from First Prototype Platform (Weight/Position).....	130
Test Platform Positional and Weight Data.....	132
Results from Validating Average Weight Algorithms.....	133
Appendix 3: Final PCB Schematic .....	137

## Chapter 1

### Introduction and Project Aims

Over the past 30 years, the New Zealand dairy industry has experienced huge growth, and is now the country's biggest export earner, worth \$14 billion a year [1]. It contributes about 2.8% of New Zealand's GDP [2]. The top three biggest economic losses to the dairy industry are from diseases and causes such as mastitis, sub-fertility and lameness. Lameness affects the well-being of the cow, impacts its productivity, reduces milk production, and decreases fertility [3]. The occurrence of lameness is on the rise and a cause of concern.

The most common methods to identify cattle lameness involve observing and noticing the walking pattern of the cattle; typical signs being cattle walking slower and with an irregular stride. This method of detection is time consuming, labour intensive and inefficient. Since the introduction of milk shed automation, lameness cases have increased; this is because the contact time between the farmer and cattle has decreased significantly. Early stages of lameness are now easily missed and lameness is only noticed once the cow has become moderately to severely lame, costing the farmer about \$500 to treat the cow. Therefore, there is a need to be able to automatically detect lameness in its early stages.

Making use of electronics to automatically detect lameness is a reasonably new concept. Currently the only company that sells a commercial product is Bou-Matic; they have developed the StepMetrix system which helps detect cattle lameness automatically. Their product has been investigated to help generate ideas on how to develop an automated cattle lameness detection system; this includes using an array of load cells on separate sections. There are three main variables used to determine lameness: these are the force, location and duration of each leg.

The electronics used in this project are based on receiving the analogue signals produced by the load cells and converting these to digital values. An amplifier circuit will be required to amplify the signals from the load cells, and an ADC to convert the analogue signals to digital signals. A microcontroller is required to interface with the ADC and to transfer the digitized load cell values to a computer for further processing. The proposed platform will consist of four sections; therefore, a communication protocol is required to acquire the data from each section sequentially.

The mechanical aspects of the project involved creating and manufacturing an initial platform section to test the accuracy of the load cells. These results were then taken into account and the final platform was designed and manufactured. Laboratory testing was conducted to test the validity of the platform and to observe healthy and lame human gait signal signatures. Once it was proven that the platform functioned as intended, the platform was installed at the Massey Dairy Farm Number 1 milking shed. After the platform was installed, dairy cows walked over the platform with their signal signature being recorded.

Various software tools were then developed to plot the recorded signals, calculate the positional coordinates, calculate average weight, calculate hoof duration, calculate stride length, split cow data by Electronic Identification (EID) tags, and finally to convert the recorded data to a video.

#### 1.1. Overall Project Aims

The overall project aims are listed below:

- To test the hypothesis that a lame cow will produce a distinct signal signature and potentially have a different stride length compared to a healthy cow.
- Replace current walkover weighing scales with an electromechanical detection system that not only weighs cows but is also capable of determining if a cow is starting to show indications of lameness.



## **1.2. Current Project Aims**

The aims of the current project, which is a sub-set of the overall project, are listed below:

- To design and manufacture a platform that uses an array of load cells and captures the data produced by the load cells.
- To process the data produced by the load cells; this includes being able to plot the data as a function of time, calculate and plot the positional coordinates, calculate the average weight, calculate the hoof duration, and calculate stride length; also the ability to split cow data, and render that data to a video for further processing.

## **1.3. Project Objectives**

The project objectives are listed below:

- Interface multiple load cells to a microcontroller
- Implement communication protocols that can be used to transfer data between multiple microcontrollers and also to the computer for further processing.
- Design and manufacture a prototype platform. This platform will be used to determine whether force, duration and position of an applied load anywhere on the platform can be calculated with precision.
- Integrate the electronics, software and mechanical components and install the final prototype into a dairy shed for testing on cattle.
- Record data and use it to help develop software tools that can be used to help identify lameness.

## Chapter 2

### Preliminary Survey and Research

In this chapter, we will look at how the New Zealand Dairy Industry is doing, how to identify lameness, what causes lameness, and the cost of lameness. This is followed by what a load cell is, the different types of load cells, where load cells are used, and how load cells work. The various communication protocols used in this project are then explored; this includes RS-232, RS-485, the Serial Peripheral Interface Bus, I<sup>2</sup>C interface. Various averaging techniques are then explored, which are used to calculate the average weight of the cow.

#### 2.1. Lameness Research

##### 2.1.1. The New Zealand Dairy Industry

As of 2013, the New Zealand dairy cattle population reached 6.6 million [4] with an average herd size of 413 cows per farm [5]. In Figure 1: Trends in the Number of Dairy Herds and Average Herd Size it can be seen that over the past 30 years in New Zealand, the average herd size has been increasing steadily, while the number of herds have been decreasing. The average herd size increase can be due to automation technologies becoming more readily available and the farm area growth occurring on the South Island [5].

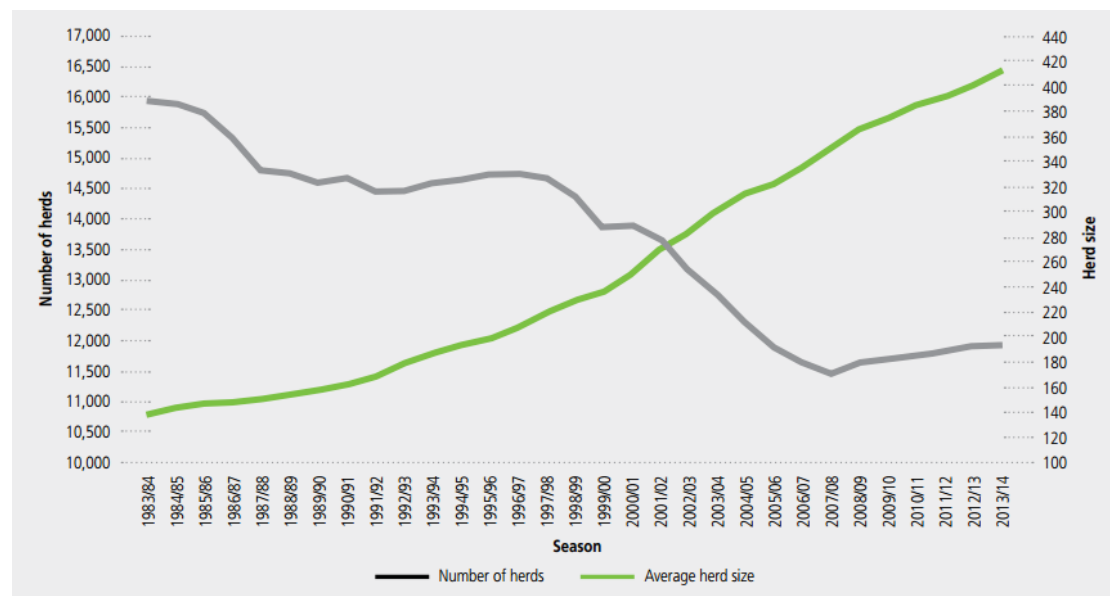


Figure 1: Trends in the Number of Dairy Herds and Average Herd Size [5]

##### 2.1.2. Identifying Lameness

There are various scoring systems to identify lameness, the most common being the Locomotion Scoring System. When using this scoring system, the farmer observes the cattle walking, with emphasis on the head bob and stride length. The occurrence of lameness is best determined using a simple 4-point Locomotion Scoring System, with 0 being normal and 3 being severely lame; there is a 3-point and 5-point scoring system as well [6]. The scoring system is shown in Table 1.

**Table 1: Locomotion Scoring Criteria [6]**

<b>Score</b>	<b>Description</b>
<b>0</b>	Cow walks with a level back and long strides. Walks rapidly, confidently and no apparent signs of lameness.
<b>1</b>	Cow shows no apparent signs of limping; however, the cow will take shorter strides and have a slightly arched back.
<b>2</b>	Cow's head carried low or bobbing up and down. Signs of obvious arched back and limping affected limb(s).
<b>3</b>	Cow has a very noticeable arched back, difficulty turning; moves slow and only applies partial weight to affected limb(s).

### **2.1.3. Cause of Lameness**

There are various reasons why cattle become lame. The general reasons are because of poor cow tracks, poor quality floors in cattle housing, cows being forced to stand on hard surfaces for an excessive amount of time, cubicles being of poor design, ineffective foot trimming, contagious diseases and malnutrition [7].

Milking sheds typically have concrete floors, which can be rough and very abrasive to the cow's hooves. Claw disorders account for 90% of lameness [8]. Although the cow's front hooves carry up to 75% of the cow's weight, lameness occurs 70% of the time in the hind limbs [8]. White line disease is one of the main causes of lameness and relates to the handling of cattle. As seen previously in Figure 1, the herd sizes have been increasing over the past few years. Because of this, the farm sizes have increased and the cattle need to walk further to get to the milking shed. This can cause extra wear and tear to their hooves while being handled and can increase the likelihood of the cows becoming lame. The stock herders now also need to spend more time handling the cattle; this can result in them being impatient and pushing the cattle too hard, increasing incident rates.

### **2.1.4. Cost of Lameness**

Cow lameness cases in New Zealand on dairy farms are around 10%, but can be as high as 15%. The average lameness incident costs the farmer \$500 [3]. As seen in Figure 1, the average herd size is 413; this means that annually the farmer can expect between 41 and 62 cases (the same cow can get lame multiple times) of lameness. The result of this is an annual cost of between \$20,500 and \$31,000. These figures clearly show the importance of monitoring each cow individually and detecting cases of mild lameness (score of 1) and treated before they become moderately (score of 2) or severely lame (score of 3), consequently decreasing the average lameness incident costs.

The proposed lameness detection platform is estimated to cost between \$7,500 and \$10,000 to manufacture. It is expected that the platform could be sold for at least \$15,000 and is expected to last several years. If the proposed platform can successfully detect mild lameness cases, cattle can be treated earlier and the farmer can save money as treatment costs would be less. The cattle will also be more productive throughout the year and be in less pain.

## 2.2. Current Lameness Detection Systems

Currently there are only two competitors on the market that provide an automated solution to detect cattle lameness force measurement systems. The first system is called the GAITWISE system, and the second system is called the StepMetrix system.

### 2.2.1. The GAITWISE System

The GAITWISE system is being developed in Belgium at the Institute for Agricultural and Fisheries Research (ILVO). The GAITWISE system has been in continuous development over the past 6 years and displaying very promising results [9]. The GAITWISE system however is not available on the market as of 2015.

The system has a 1m x 6m pressure sensitive mat; the system measures the position and relative force of hooves with respect to time. The system is able to monitor the cow's gait using four dimensions (one temporal, one force and two spatial). The system analyses 10 specific variables with these being stride length, stride time, stance time, step overlap, abduction, symmetry in step width, step length, step time, stance time and force. The system operates in real-time and is fully automatic and is able to accurately classify lameness 88% of the time [9].

The pressure sensitive mat uses an array of 384 sensor elements. The pressure sensitive mat is protected by multiple layers; the first layer consists of a 1mm thick ethylene propylene diene monomer flexible water and manure cover, the second layer is a 10mm thick rubber mat to provide mechanical protection and skid resistance. The system has a sampling rate of 60Hz [9].

Testing was done on a farm in Belgium where a sample herd of 80 dairy cows were milked twice daily. A video camera that records at 30fps was mounted in the dairy shed to record the cows as they walked over the platform. The video footage was then handed to a trained veterinarian to score the lameness level (as discussed in Section 2.1.2) of each cow. The GAITWISE system score was then compared to that of the trained veterinarian [9].



Figure 2: GAITWISE System Installed on a farm in Belgium [9]

### 2.2.2. The StepMetrix System

The StepMetrix system is a commercial product developed by Bou-Matic. The StepMetrix system consists of three main components:

- Step sensor Platform that is permanently installed in the return lane of the milking shed.
- SMX Score Controller that analyses each cow's steps, generates and transmits SMX scores to a computer.
- StepMetrix Management Software that is installed on the computer. The software allows the user to analyse SMX scores and generate reports.

The StepMetrix system has an average score of 85% accuracy of detecting lameness in individual cows; this includes cows that have out-of-balance hooves. The system was able to detect lameness for weeks before humans could observe any signs of lameness in the cows. The system currently retails for approximately \$60,000 [10].

The system consists of an array of single-axis load cells that are mounted under two parallel floor-plates. Each floor plate consists of four load cells, so a total of 8 load cells are used. The system has a sampling rate of 100Hz [11].



Figure 3: StepMetrix System by Bou-Matic [10]

### **2.2.3. Intellectual Property**

Various development companies and universities have realised the opportunity to develop a platform that detects lameness and numerous patents have been filed worldwide, therefore it is important not to infringe on any of these patents.

A patent (WO 2013052001 A1) by Delaval Holding Ab was filed in 2012 and is the only patent in the New Zealand registrar that mentions lameness detection. The claims from this patent refer to making use of image processing and positioning of cameras to detect lameness. As no image processing or cameras are involved in the scope of this project, the patent will not be infringed.

In the US, a patent has been granted for the StepMetrix System (US 6699207 B2). The developed system is similar to this project. The developed system makes use of multiple load cells to assist with detecting lameness, however the claims from this patent relate more to the computer based diagnostic system and do not protect the use of multiple load cells, therefore the patent will not be infringed.



## 2.3. Technical Research

### 2.3.1. Tru-Test Products

Tru-Test is the main stakeholder for this project and will be funding the project. A technical meeting took place on the 1<sup>st</sup> of May 2014 at Tru-test in Auckland to gather technical knowledge about their current walkover weigh system and how it works. A typical walkover weigh system consists of the following:

- Platform. The platform has two load bars with each bar only having a half-bridge strain gauge. These two bars are then wired together to form a complete bridge.
- EID Antenna and EID Reader. The EID Reader has an ARM Cortex M3 microcontroller. This interfaces with a single channel 24-bit ADC with a sampling rate of 50Hz.



Figure 4: Walkover Platform (Top) [12], EID Antenna and EID Reader (bottom) [13]

An important piece of information the engineers mentioned was that the cows could generate signals up to five times their average weight when walking over the platform. Because of this, they recommended using 1000kg rated single-point shearbeam load cells, and also to use the ASB-1000 by PT Global. The engineers also suggested waterproofing the electronics, and to ensure that no high pressure water came into direct contact with the load cells as this has caused load cell failure in the past. It is interesting to note that although the load cells are IP-67 rated, these failures still occurred.

### 2.3.2. Ground Reaction Forces

The GAITWISE and StepMetrix systems both rely on ground reaction forces that are produced by cattle walking over the platform. By capturing these forces, it is possible to calculate the weight, time and hoof location of the cow walking over the platform, all which are major variables in being able to determine whether a cow is starting to show signs of lameness.

There are various force transducers on the market, some of these are listed below:

- Load bar (used in walkover weigh platforms by Tru-Test)
- Pressure-sensitive mat (used in the GAITWISE system)
- Load cell (used in the StepMetrix system by Bou-Matic)
- Tactile sensors

Any of these transducers could be used to measure the ground reaction forces, however Tru-Test suggested to use ASB-1000 load cells by PT Global, as load cells are more robust and used more commonly in the production of scales. By using load cells the overall platform cost would also be less and make it more affordable.

### 2.3.3. Load cell

A load cell can be described as a sensor that detects force (mass, torque, etc.). When a force is applied to a load cell, the force is converted into an electrical signal. Load cells are also known as “load transducers”.

There are various kinds of load cells, the strain gauge load cell being the most dominant. Therefore, when we say “load cell”, we generally refer to a strain gauge load cell.

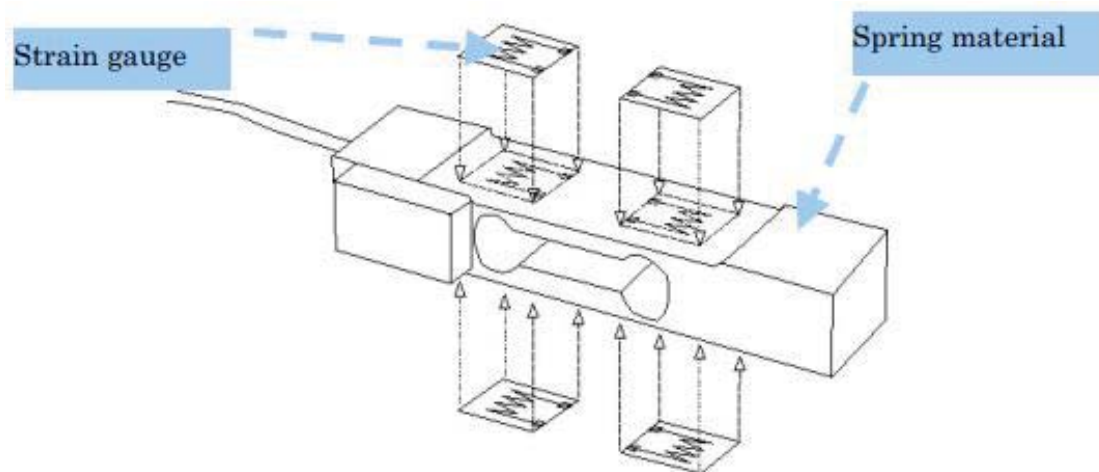


Figure 5: Strain Gauge Load Cell [14]

Load cells are generally made from metals such as aluminium, iron, or stainless-steel. The fatigue life indicates the number of times a rated capacity can be loaded. Sudden shock or applying a force that exceeds the rated capacity for a long time will damage the load cell. With proper usage, maintenance, and protection, a load cell can be used for numerous years [15].

The most common arrangement for a load cell is a Wheatstone bridge configuration which consists of four strain gauges. Cheaper and less accurate load cells are available with half bridge or quarter bridge strain gauges.



### 2.3.3.1. Characteristics of a Strain Gauge Load Cell

Listed below are a few of the characteristics of the strain gauge load cell:

1. Able to provide extremely precise measurements without being affected by temperature changes.
2. As the output is an electrical signal, long distance communication is possible. It is easy to do processing and calculations with a computer.
3. It is of small size compared with other types of load cells.
4. As there are no moving parts or any parts that generate friction, maintenance is easy and it has a long operating life.
5. Because of the sensor's simple operation principle and small number of components, production is easy.
6. As long as the device is not overloaded, the load cell has excellent fatigue characteristics and its performance can be maintained semi-permanently.
7. The deflection due to the deformation of the spring material is small, and the spring material's natural frequency is high. Therefore, it is possible to shorten the measurement time [16].

### 2.3.3.2. Types of Load Cells

Load cells can be divided into four main types: beam load cells, column load cells, "S" load cells and finally diaphragm load cells.

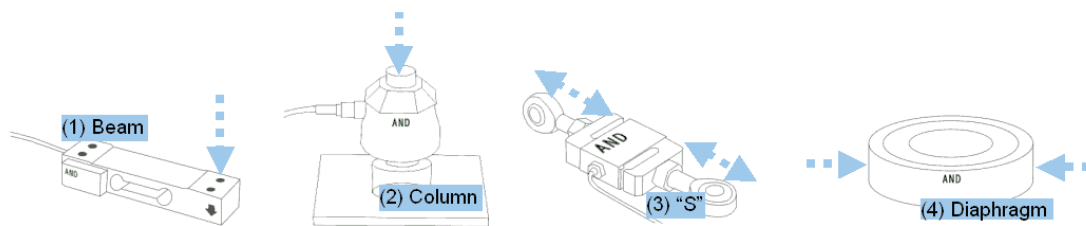


Figure 6: Different Types of Load Cells [17]

It is vital to use the load cell with the capacity and structure appropriate to the position where it will be used [17].

### 2.3.3.3. Where Are Load Cells Used?

Wherever there is a "force measurement" required, we can make use of load cells. We seldom come into direct contact with load cells, but they are used in various measuring instruments such as electronic platform scales, bathroom scales, industrial scales, testing machines etc. [18]



Figure 7: Example of Bathroom Scale (Left) [19] and Electronic Platform Scale (Right) [20]

The shape of the load cell also differs depending on the type of load cell and application it is being used for.

#### 2.3.3.4. How a Load Cell Works

A load cell typically consists of four strain gauges in a Wheatstone bridge configuration. The four strain gauges on the load cell will measure the bending distortion of the load cell as weight is applied to it. Two of the strain gauges measure tension, while the other two measure compression. A strain gauge is typically made of very fine wire, or foil and setup in a grid pattern in such a way that there is a linear change in electrical resistance when strain is applied. A Wheatstone bridge configuration is used to measure small changes in resistance and turning it into something more measurable. A differential amplifier is then used to take the output from the Wheatstone bridge (typically a few millivolts) and amplifying the signal to a more useful voltage [21]. The actual deformation of the load cell is so small that it is very difficult to observe visually [22].

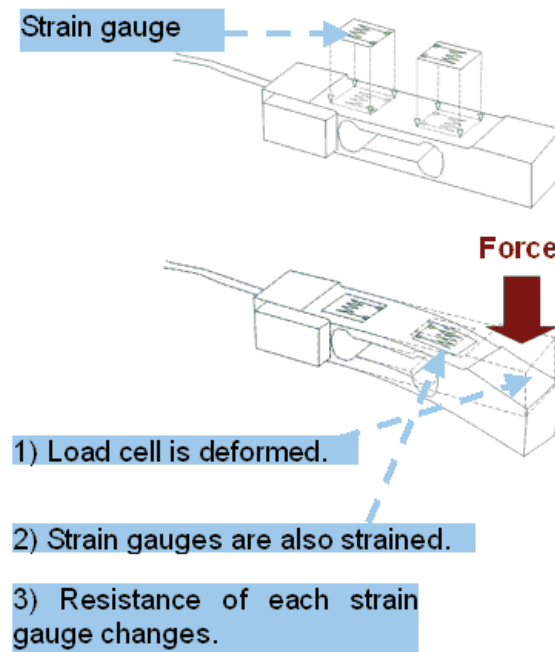


Figure 8: Demonstration of Load Cell Deforming [22]

#### 2.3.4. Communication Interfaces

##### 2.3.4.1. Balanced and Unbalanced Systems

When selecting a data communication interface, the choice between balanced and unbalanced transmission lines is an important factor. The RS-232 standard is an unbalanced standard, whereas the RS-485 standard is a balanced standard.

When only one wire carries the signal voltage, the system is known to be 'unbalanced'; RS-232 is an unbalanced system. RS-232 also has a 'signal common' wire or sometimes referred to as the signal ground wire. The voltage between the signal voltage wire and the ground wire is known as the transmitted signal.

Two conductors are required by the RS-485 interface standard to transmit each signal. The voltage difference between these two wires is measured at the receiving end; this is known as a differential or balanced system. Many interference problems associated with the 'signal common' wire were eliminated because of this.

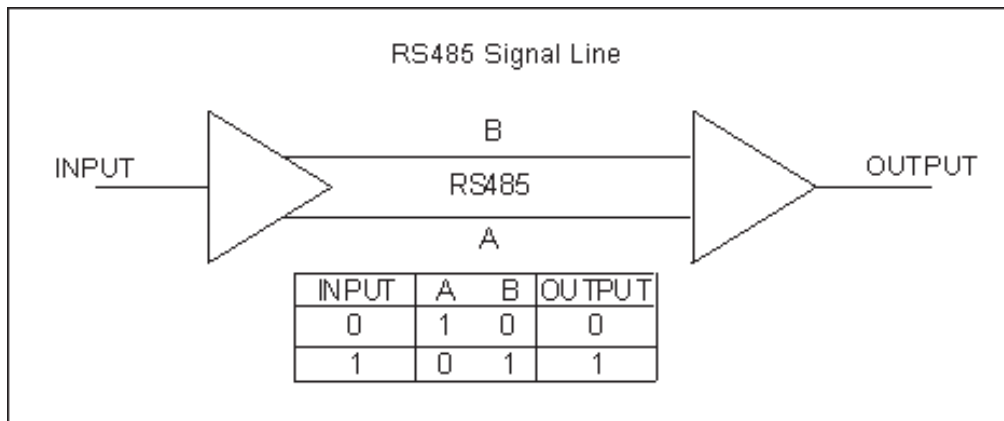


Figure 9: Logic of RS-485 Communication [23]

By making use of a balanced transmission line, it is possible to use higher data transfers over longer distances. A balanced communication protocol such as RS-485 is favoured in industrial applications where noise can be a major problem. One disadvantage of a balanced system is that the system requires two conductors for every signal [24].

#### 2.3.4.2. RS-232 Standard

The RS-232 interface standard specifies the method of connection between two devices: the Data Terminal Equipment (DTE) and the Data Circuit Terminating Equipment (DCE). The DTE for example can be a computer, and the DCE can be a modem, but in this project the DCE is the microcontroller [24].

Equipment that uses the RS-232 standard have the following features:

- Point-to-Point Communication
- Full Duplex Communications
- Suitable for serial, binary and digital data communication
- Asynchronous, meaning there is a fixed timing between data bits, but variable time between character frames.

It is possible to achieve reliable communication up to a distance of about 15 metres; this depends on the type of cable used and it is possible to achieve data rates of 160kbps [24].

#### 2.3.4.3. RS-485 Standard

The RS-485 standard is balanced or differential standard. Because RS-485 uses two wires, it permits a 'multidrop' network communication, see Figure 10.

RS-485 allows reliable serial communication for:

- Data rates up to 10 Mbps
- Distances of up to 1200m
- Up to 32 line drivers on the same line
- Up to 32 line receivers on the same line

RS-485 lines can operate in three states; this is known as tri-state operation:

- Logic 0
- Logic 1
- High-Impedance

When the line driver is in high impedance, virtually no current is drawn and the line driver appears not to be present on the line, also known as the 'disabled' state. This state can be initiated by a signal on a control pin on the line driver integrated circuit (IC). When using a 'multidrop' network it is important to allocate a unique address to each device in the network; this is to avoid confliction with other devices. RS-485 does include current limiting in cases where confliction would occur between devices [24].

RS-485 communication is useful for a system that requires communicating with several devices or controllers connected to the same lines/wires. Special care however needs to be taken to co-ordinate which device on the 'multidrop' network can become active, else conflicts can occur. Typically, a computer or microcontroller is used to control which device is active at any one time [24].

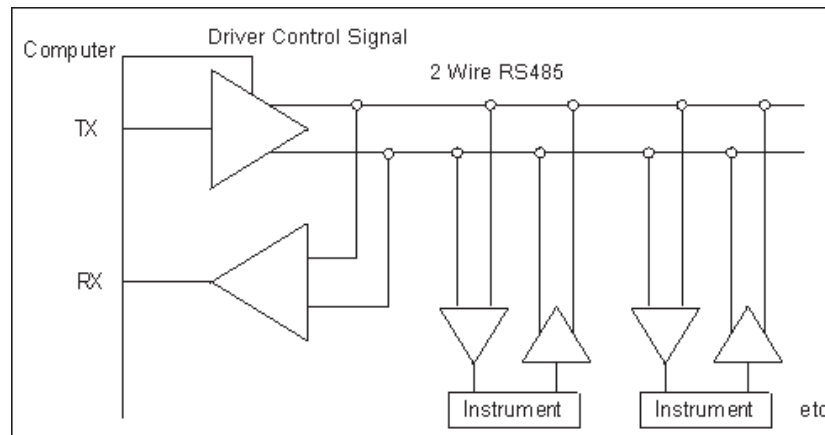


Figure 10: Half-Duplex System: Sending and Receiving Data over 2 Wires [23]

Normally RS-485 does not require any special termination, however, when using long wires, the leading trailing edges of the data pulses can be much sharper if fail-safe biasing resistors approximately equal to the characteristics impedance of the line are fitted at the extreme ends [24].

### 2.3.5. Serial Peripheral Interface Bus

The Serial Peripheral Interface (SPI) is an interface bus commonly used to send data between a microcontroller and small peripherals such as sensors, analogue-to-digital converters, shift registers, and SD cards. The SPI interface uses separate clock and data lines, along with a select line to choose the device you wish to communicate with [25].

SPI is a “synchronous” data bus; this means it uses separate lines for data and a “clock” signal that is used to keep both sides in synchronization. The clock signal is an oscillating signal that indicates to the receiver when to sample the bits that are on the data line. This can occur on the falling (high to low) or rising (low to high) edge of the clock signal. The datasheet of the device you want to interface with will specify whether to sample on the falling or rising edge of the clock signal. Once the receiver detects the edge, it will sample the data on the data line (see the arrows in Figure 11) [25].

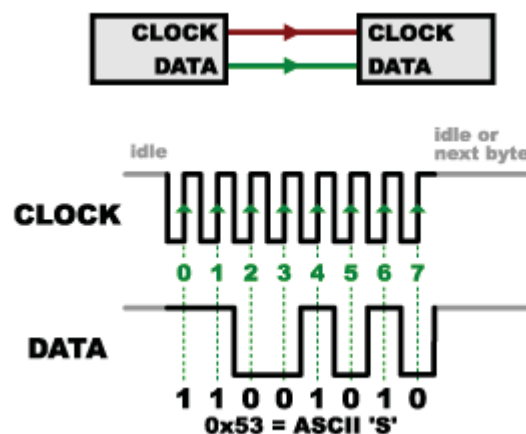


Figure 11: When the Receiver Detects the Edge, It Will Immediately Look at the Date Line (See the Green Arrows) [25]

When using SPI, a clock signal is required; this is usually known as CLK or SCK for Serial Clock. The clock signal is generated by the “Master” and the other side is called the “Slave”. When using SPI there is always only one master, which is normally a microcontroller [25].

When data is sent from the Master to the Slave it is sent on the MOSI (Master Out / Slave In) line. If the Slave needs to send data back to the Master, the master will continue to generate a prearranged number of clock cycles and put the data onto the MISO (Master In / Slave Out) line [25].

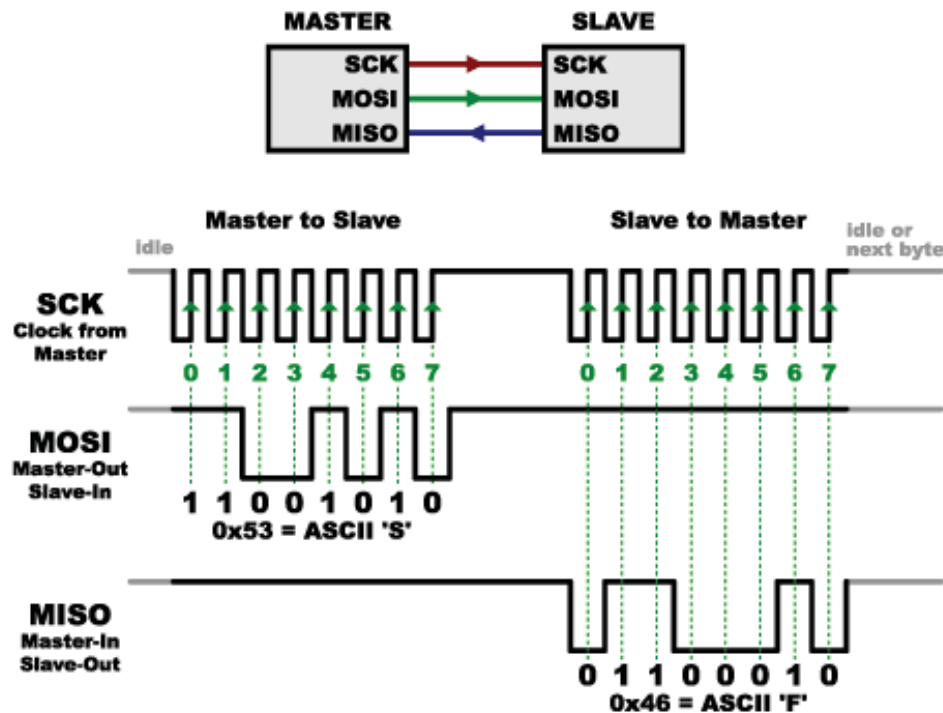


Figure 12: Example of Data Being Send On the MOSI and MISO Lines [25]

Earlier the word ‘prearranged’ was used, this is because the master always generates the clock signal and it needs to know in advance whether the slave is going to return data to the master and how much data will be returned. SPI is used to communicate with sensors that have a very specific command structure. For example, if you send the command “read data” to a device, you know that the device will for example always return one byte [25].

SPI has a separate send and receive line, therefore SPI is known to be “full-duplex”, this makes it possible to transmit and receive data at the same time [25].

It is possible to have a “Master” communicate with multiple “Slave” devices. Because of this SPI has another line known as Slave Select or SS for short. This line is used to communicate with the “Slave” device that it should wake up and communicate with the “Master” and to select which “Slave” you would like to communicate with. Only one “Slave” device should be active at any one time [25].

The SS line is normally active low, which means if the line is held high, the device is disconnected from the SPI bus. To activate the slave device, the line is held low, enabling the master to communicate with the slave device. The SS line is normally brought back to high again after you are done communicating with the slave device [25].

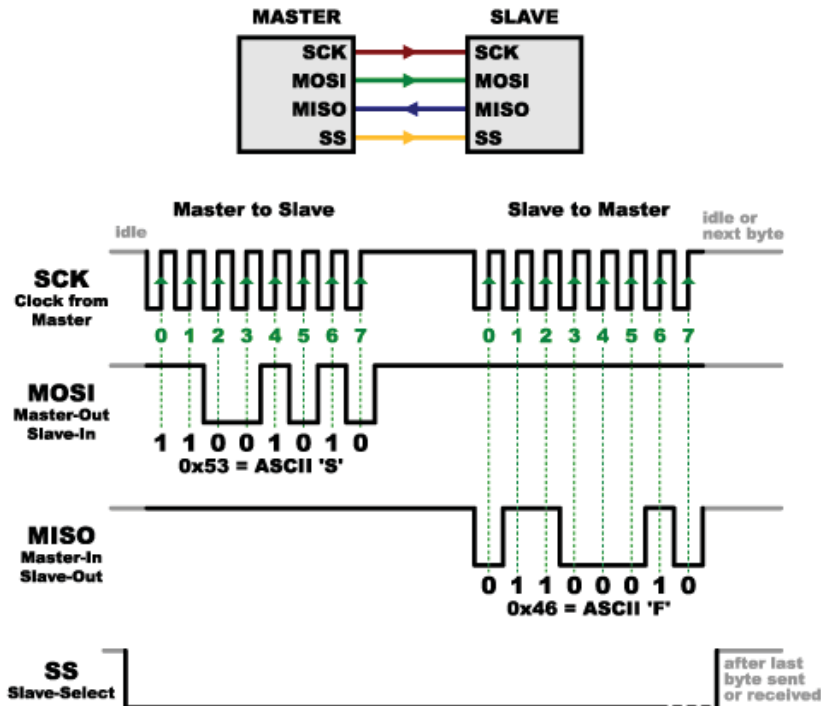


Figure 13: Example of Data Being Send On the MISO and MOSI Lines While SS Is Held Low [25]

The SS line is often used to signal the beginning of a data stream and to “latch” the data at the end of the stream.

### **Advantages and Disadvantages**

Listed below are some of the advantages and disadvantages of the SPI bus.

Some advantages include:

- It can support multiple slaves.
- The receiving hardware can be as simple as a shift register.
- Faster than asynchronous serial communication [25].

Some disadvantages include:

- More signal lines required compared to other communication methods.
- Communication must be well-defined in advance (it can't send random amounts of data whenever you feel like it).
- Master must control all the communication (slaves aren't able to communicate directly to each other).
- Each slave device requires its own separate SS line, which can be problematic if several slaves are required [25].

### **2.3.6. I<sup>2</sup>C Interface**

One of the most obvious drawbacks of SPI is the number of pins required. Simply connecting a single master to a single slave device with the SPI bus requires four lines, and on top of that, each additional slave requires one additional SS pin on the master [26].

The Inter-Integrated Circuit (I<sup>2</sup>C) is ideally suited for typical microcontroller applications. The I<sup>2</sup>C protocol allows the systems to interconnect up to 128 different devices using only two bi-directional bus lines: one for clock (SCL) and one for data (SDA). These are 16 reserved addresses, meaning it is only possible to connect a maximum 112 devices using this protocol. The transfer rate varies from 10Kb/s (low speed) to 100Kb/s [27] and operates well for distances less than 3 metres. The only external hardware needed to implement the bus is a single pull-up resistor for each of the I<sup>2</sup>C bus lines. All devices connected to the bus have

individual addresses, and mechanisms for resolving contention are inherent in the I<sup>2</sup>C protocol.

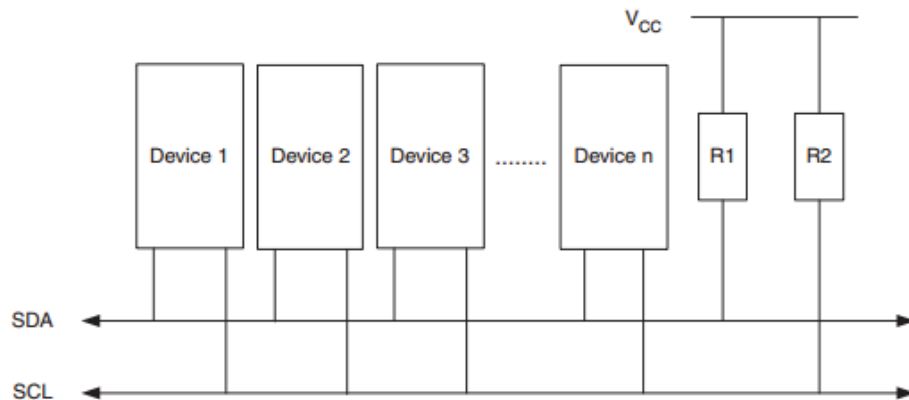


Figure 14: I<sup>2</sup>C Bus Interconnection [27]

The I<sup>2</sup>C protocol defines the concept of master and slave devices. A master device is the device that is in charge of the bus, and controls the clock and generates the START and STOP signals. Slave devices listen to the commands sent by the Master and respond to them [27].

### 2.3.7. Averaging Techniques

It is important to discuss the various averaging techniques as these are used to aid with the calculations of the average weight of the cow.

#### 2.3.7.1. Running Average

A running average (also known as a moving average or rolling average) is calculated by taking the arithmetic mean of a given set of values [28]. For example, to calculate a 10-point moving average of various weights, you would add up all the weights and divide it by 10.

**78.6kg 77.2kg 79.6kg 77.6kg 78.2kg 78.7kg 77.3kg 77.9kg 79.1kg 78.4kg**

$$78.6 + 77.2 + 79.6 + 77.6 + 78.2 + 78.7 + 77.3 + 77.9 + 79.1 + 78.4 = 782.6\text{kg}$$

Average:  $782.6 \div 10 = 78.26\text{kg}$

At this point of time, this doesn't really look any different than just regular mean; it will become more apparent as more weights are added to weight list as to why it is called a running average [28]. After adding new data to the weight list, the average is calculated as follows:

**78.6kg 77.2kg 79.6kg 77.6kg 78.2kg 78.7kg 77.3kg 77.9kg 79.1kg 78.4kg 79.6kg**

$$78.6 + 77.2 + 79.6 + 77.6 + 78.2 + 78.7 + 77.3 + 77.9 + 79.1 + 78.4 + 79.6\text{kg} = 783.6\text{kg}$$

Average:  $783.6 \div 10 = 78.36\text{kg}$

It is now more apparent as to why it's called a running average. As new data is becoming available, the oldest data point is dropped and the new data point comes in to replace the old data point. Therefore, that data set is constantly "running" to account for new data as it becomes available. This method ensures that only current information is accounted for [28].

### 2.3.7.2. Weighted Average

A weighted average is where each quantity to be averaged is assigned a multiplier constant. The multiplier constant determines the importance of each quantity on the weighted average. The weighted average acts as a low pass filter. To calculate a weighted average, the following steps are performed:

1. Multiply each value by its multiplier constant.
2. Add up the weighted values.
3. Add up the multiplier constant for each value.
4. Divide the total of the weighted value by the total of the multiplier constants.

This is best explained by doing an example:

Data (kg)	78.60	77.20	79.60	77.60	78.20	78.70	77.30	77.90	79.10	78.40
Multiplier Constant	0.00	0.25	0.50	0.75	1	1	0.75	0.50	0.25	0.00
Weighted Value (kg)	0.00	19.30	39.80	58.20	78.20	78.70	57.98	38.95	19.78	0

Total Weighted value: 390.91kg

Total of multiplier constants: 5

Weighted Average:  $390.91/5 = 78.182\text{kg}$



## Chapter 3

### Electronic System Design and Development

#### 3.1. Specifications

Listed below are the main specifications the electronics must adhere to:

- An ADC that can interface with at least four load cells is required as each section will consist of four load cells.
- The chosen ADC must be highly accurate and stable as the electrical signal produced by a load cell is only a few millivolts.
- The ADC must have an adjustable sampling rate as it might be useful when determining whether a cow is starting to show indications of lameness or not.
- As Tru-Test's current weighing platform measures at 50Hz, the chosen ADC should be capable of measuring at a frequency equal or greater of 50Hz. The higher the sampling rate, the more data can be captured and some hidden data might be revealed that indicates cattle lameness.
- The microcontroller must be able to send digitized data from the load cells to a computer for further processing. This can either be via serial communication or wireless communication.

#### 3.2. Functional Blocks of the System

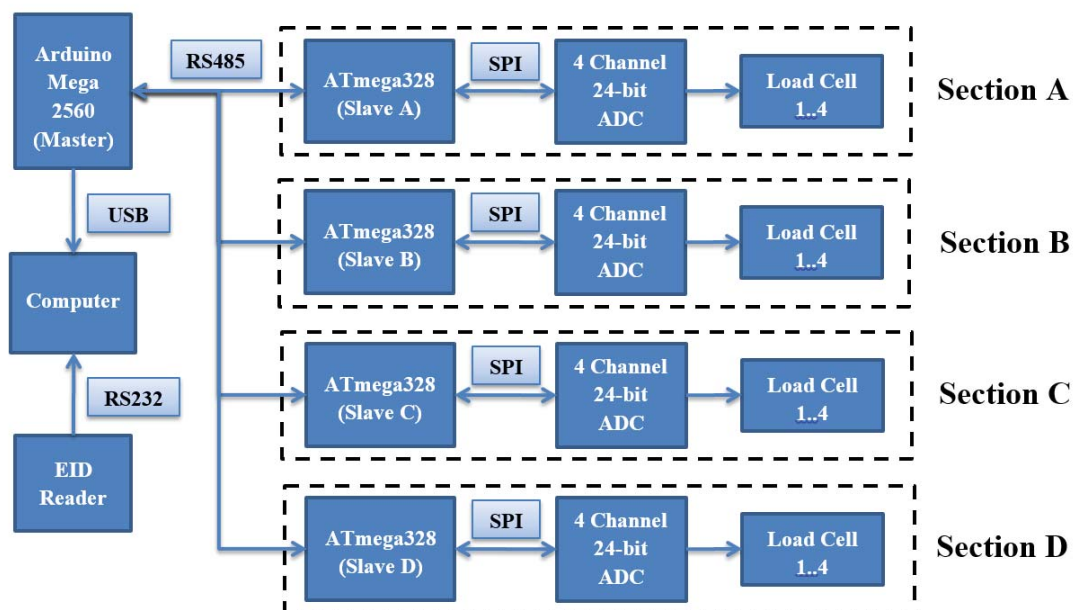


Figure 15: Functional Block Diagram of System

In Figure 15: Functional Block Diagram of System it can be seen that the platform overall consists of four independent sections, with each section having its own unique ID. Each section contains the following hardware: four ASB1000 single-point shearbeam load cells, which converts the force into an electrical signal; a 24-bit four-channel AD7193 Analogue-to-Digital Converter (ADC), which converts the electrical signal obtained from the load cells to a digital signal, which is transmitted to a microcontroller via the SPI bus; and finally an ATmega328 microcontroller acting as a slave device. The Arduino Mega 2560 acts as the Master device and controls all communication. It will query each slave device in-turn, i.e. Section A, Section B, Section C, Section D, Section A, Section B etc. The master device communicates with the slave devices via the RS-485 communication bus. The data received from any slave device is transmitted serially to a computer for further processing. An EID reader is also connected to the computer to help identify which cow walked over the platform, and the ID of the cow is transmitted serially to the computer.

### **3.2.1. Resources Required for Development**

To make the project a success the following resources are required:

Hardware:

- Sixteen Load Cells
- Four AD7193 ADCs
- Four High Precision Voltage Regulators
- Four Op-amps
- Four ATmega328 Microcontrollers
- An Arduino Mega 2560
- A Computer
- An EID Reader

Software:

- Altium Designer
- Fritzing
- Arduino IDE
- Python IDE
- Qt Creator

### 3.3. Electronic Development

#### 3.3.1. Objective

The objective of the electronics is to take the electrical signal produced by each load cell and to digitize it; the digitized data then needs to be transmitted from the microcontroller to a computer for further processing.

#### 3.3.2. Initial Prototype

The aims of the initial prototype were to find load cells that could withstand cow weights, design a breakout board that would fit directly onto an Arduino Uno and be able to directly interface with four load cells, and the digitized load cell data that is obtained be able to be transmitted to a computer for further processing.

##### 3.3.2.1. Component Selection

In this project, single-point shearbeam load cells were used. The load cells used in the project were ASB1000 manufactured by PT Ltd and could withstand weights up to 1000kg [29]. Their low cost made them a very attractive option, each load cell costs around \$120 (New Zealand Dollars). By reading the data sheet of the ASB1000, the full scale output was found to be  $2\text{mV/V} \pm 0.1\%$  (see Appendix 1: Critical Component Datasheets, therefore when exciting the load cells, it was of utmost importance to use a highly stable voltage regulator.

The two main components required for the breakout board design were a highly stable voltage reference (to excite the load cells) and an ADC (to digitize the analogue signals produced by the load cells). Surface mount components were selected to ensure a minimum footprint; this allowed a breakout board to be designed to fit within the Arduino Uno header pins.

The high grade REF5040I by Texas Instruments was chosen for the stable voltage regulator to excite the load cells as it has low-noise ( $3\mu\text{V}_{\text{pp}}/\text{V}$  (max)) and low-drift ( $3\text{ppm}/^\circ\text{C}$  (max)) characteristics [30]. It also has an output voltage of  $4.096\text{V} \pm 0.05\%$  and an output current of  $10\text{mA}$  [30]. As the voltage regulator outputs  $4.096\text{V}$  and the load cells have a full output scale of  $2\text{mV/V}$  [29], if  $1000\text{kg}$  is placed on a load cell, an electrical signal of  $8.192\text{mV}$  will be produced by the load cell, meaning a very high resolution and low noise ADC will need to be chosen to digitize the signal.

By inspecting the datasheet of the ASB1000 load cell, it was found that the input resistance was  $410\Omega$  (see Appendix 1: Critical Component Datasheets. Therefore, the current drawn by a single load-cell was calculated to be about  $10\text{mA}$ . As there are four load cells, the required current is  $40\text{mA}$ . To increase the current, a voltage follower circuit was employed. Therefore, the output current of the highly stable voltage regulator was an unimportant factor as the op-amp would supply the necessary current at the same voltage [30].

The voltage follower circuit required a high-precision op-amp that had low offset voltage drift characteristics and was able to supply enough current. An AD8656 precision CMOS amplifier by Analog Devices was chosen as it is able to retain a low offset voltage drift ( $0.4\mu\text{V}/^\circ\text{C}$ ) and is able to supply a maximum current of  $220\text{mA}$ , which would be more than suitable for the application [31].

It was important to choose an ADC with a very high resolution and low noise due to the fact that the load cells output only  $8.192\text{mV}$  at the maximum load of  $1000\text{kg}$ , so the slightest changes had to be detected. The AD7193 by Analog Devices was chosen as it would fulfil the main specifications and can directly interface with four load cells. The AD7193 requires an analogue reference voltage between  $3\text{V}$  and  $5\text{V}$  [32] and the ASB1000 load cell recommends a voltage between  $5$  and  $12$  volts [29] considering these facts, a reference voltage of at least  $3\text{V}$  was required. The REF5040I voltage regulator was chosen as it is able to supply  $4.096\text{V}$ . The AD7193 communicates with the ATmega328 via the SPI-bus. As the AD7193 has a lot of features, it was investigated in-depth.

### 3.3.2.2. AD7193 ADC Investigation

Listed below are some of the features of the AD7193:

- Fast Settling Filter Option
- 24-bit sigma-delta ADC with 4 differential/8 pseudo differential input channels
- Up to 22 noise-free bits (gain = 1)
- Internal or External Clock
- Very low gain drift ( $\pm 1$  ppm/ $^{\circ}\text{C}$ ) and offset drift ( $\pm 5$  nV/ $^{\circ}\text{C}$ )
- Multiplexor with automatic channel sequencer, simplifying communication and a buffer
- Simultaneous 50Hz/60Hz rejection and programmable filters
- Programmable variable output data rate between 4.7Hz and 4.8kHz
- Programmable gain (1 to 128)
- Averaging (2 to 16) [32]

From the listed features of the AD7193 it is evident that a considerable amount of time was spend to understand how to interface with the device.

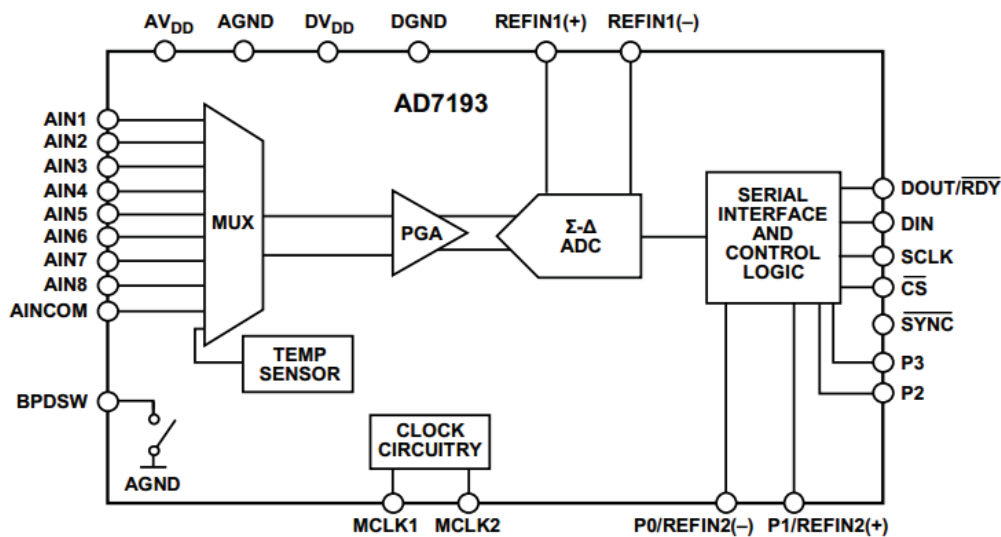


Figure 16: Functional Block Diagram of AD7193 [32]

The fact that the AD7193 features differential inputs is a significant feature as this means the load cells can interface directly with the AD7193. This also means less components on the PCB as there is no need to put any external differential amplifiers up front before interfacing with the ADC, consequently simplifying the electronics and lowering the failure rate of the electronics.

An internal block diagram of the AD7193 is shown in **Error! Reference source not found.**. The ASB-1000 load-cells are connected to AIN1 through to AIN8 and is excited from REFIN(+) and REFIN(-); the analogue reference voltage (4.096V) produced by the REF5040.

Four wires are required to communicate with the AD7193 as it communicates via SPI, these being:

- $\overline{\text{DOUT/RDY}}$ : Master In/Slave Out (MISO). It functions as a serial data output pin to access the output shift register of the ADC. The output shift register can contain data from any of the on-chip data or control registers. In addition,  $\overline{\text{DOUT/RDY}}$  operates as a data ready pin, going low to indicate the completion of a conversion.
- DIN: Master Out/Slave In (MOSI) receives data from the microcontroller to configure internal registers of the AD7193.
- $\overline{\text{CS}}$ : Chip Select (active low) is used to select the AD7193. In this case, this line will always be low to have this component selected.
- SCLK: The serial clock which can be internal or external. The serial clock input is for data transfers to and from the ADC.

The  $\overline{\text{SYNC}}$  pin is pulled high through a pull-up resistor as no synchronisation with other devices is required.

Various features of the AD7193 can be configured by configuring the registers as specified in the datasheet. A summary of these registers can be seen in Table 2: Summary of AD7193 Registers.

**Table 2: Summary of AD7193 Registers**

<b>Name</b>	<b>Read/Write</b>	<b>Register Size</b>	<b>Summary</b>
Commutations Register	Write-Only	8-bits	The data written to the communication register determines whether the next operation is a read or write and which register this operation occurs.
Status Register	Read-Only	8-bits	Indicates the status of the AD7193
Mode Register	Read/Write	24-bits	This register is used to select the operating mode, the output data rate, and the clock source.
Configuration Register	Read/Write	24-bits	This register is used to configure the ADC for unipolar or bipolar mode, to enable or disable the buffer, to enable or disable the burnout currents, to select the gain, and to select the analogue input channel.
Data Register	Read-Only	24/32-bits	The conversion result from the ADC is stored in this data register. Upon completion of a read operation from this register, the RDY pin/bit is set. When the DAT_STA bit in the mode register is set to 1, the contents of the status register are appended to each 24-bit conversion.
ID Register	Read-Only	8-bits	The identification number for the AD7193 is stored in this register.
GPOCON Register	Read/Write	8-bits	This register is used to enable the general-purpose digital inputs.
Offset Register	Read/Write	24-bits	The offset register holds the offset calibration coefficient for the ADC.
Full-Scale Register	Read/Write	24-bits	The full-scale register is a 24-bit register that holds the full-scale calibration coefficient for the ADC.

No pre-written library to interface with the AD7193 using an Arduino Uno could be found. Therefore, it was required to study the datasheet thoroughly to gain a better understanding of how to configure each register and enable/disable the features required. Once the datasheet was studied, software was developed to communicate with the AD7193.

Flow diagrams of how the AD7193 is programmed can be found in Section 4.1.1.

- The Arduino Uno header pins supply 5V and ground to the entire breakout board. The 5V line is connected to the REF5040 which produces the stable 4.096V which is used to excite the load cells and power the AD7193.
- The current required by the load cells is provided by the voltage follower circuit using an AD8656.
- The AD7193 interfaces with the load cells and converts electrical signals to a digital signal, which is then transferred to the computer via the serial line of Arduino Uno.
- Four male headers on the PCB used to connect the load cells.
- 100nF filtering capacitors on all input channels of the AD7193 (recommended by the AD7193 datasheet).
- A low impedance bead is used between the analogue and digital ground to separate the high frequency switching on the digital line. This was done to help smooth the input analogue signal.



### 3.3.2.4. PCB Design

Once the schematic was designed, it was sent to Ken Mercer for verification before the PCB design commenced. After it was verified by Ken Mercer, PCB design began. As there were no standard footprints available in Altium Designer for the AD7193, REF5040, and AD8656, these had to be custom designed. The PCB design was then sent to the Massey University Electronics Technicians to be fabricated. The designed PCB breakout board can be seen in Figure 18: Arduino Uno Breakout Prototype Board.

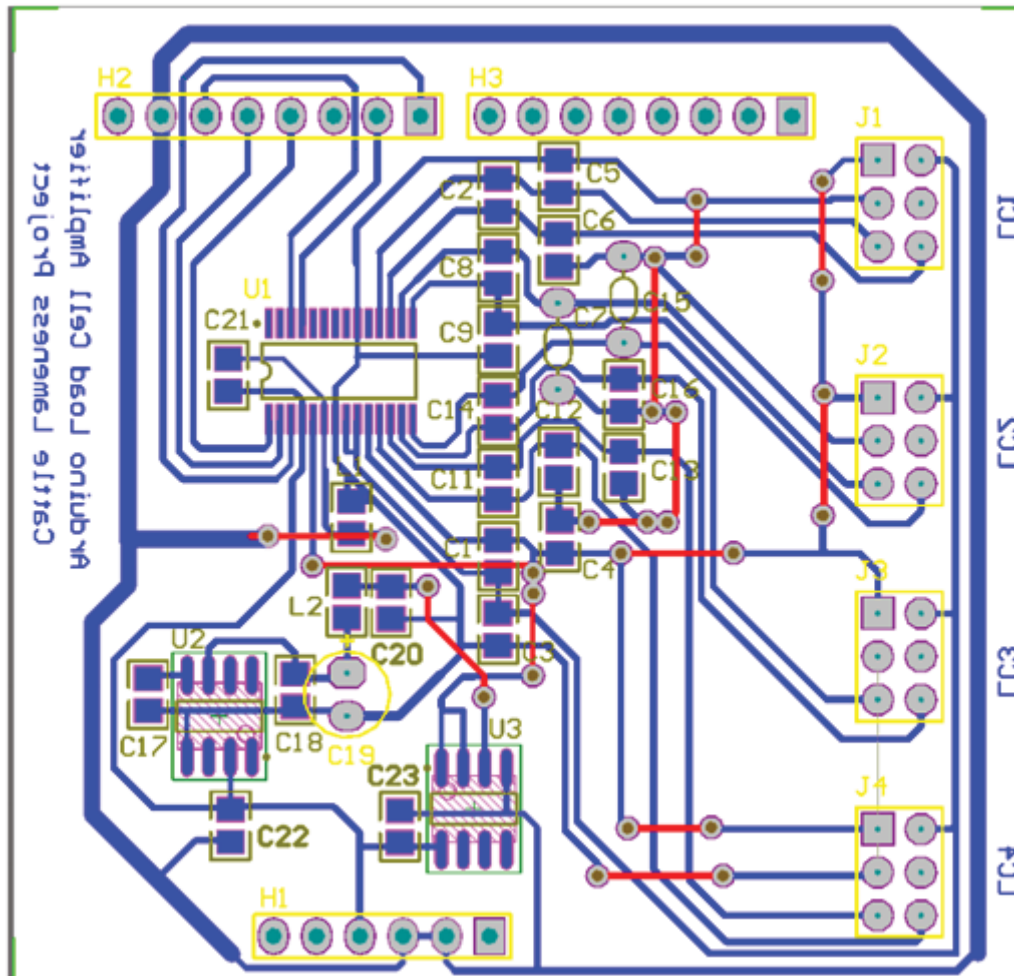


Figure 18: Arduino Uno Breakout Prototype Board

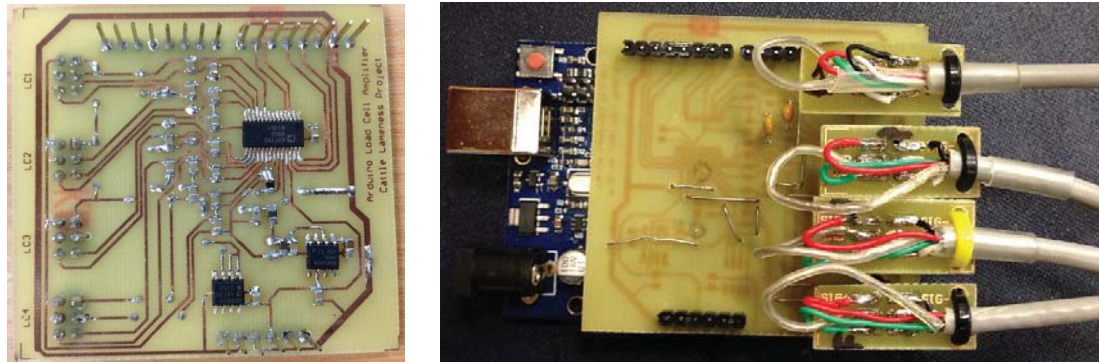
The following design guidelines and component layouts were implemented on the PCB:

- Filtering capacitors were placed as close as possible to AIN1 – AIN8 of the AD7193 as recommended by the datasheet.
- Decoupling capacitors across all ICs as per good practice.
- Ground line on the outside of the PCB to help remove noise.
- Surface mount component pads were 12 mil as this was the smallest print size available at Massey University.
- As Massey is unable to produce double sided PCBs, the components were positioned to ensure minimum amount of jump wires required.

After the PCB was fabricated, components were soldered onto the PCB in multiple stages. This was done to ensure each component was functioning as intended. The header pins and REF5040 voltage regulator were soldered onto the PCB first. The PCB was plugged into the Arduino Uno and power. The Vout pin of the REF5040 was tested and found to be 4.096V as expected. Next, the AD8656 and all capacitors were soldered onto the PCB. The output of the AD8656 was tested and found to be 4.096V as expected. Finally, the AD7193 was soldered onto the PCB. All soldering was done by hand and the small footprint of the AD7193 was

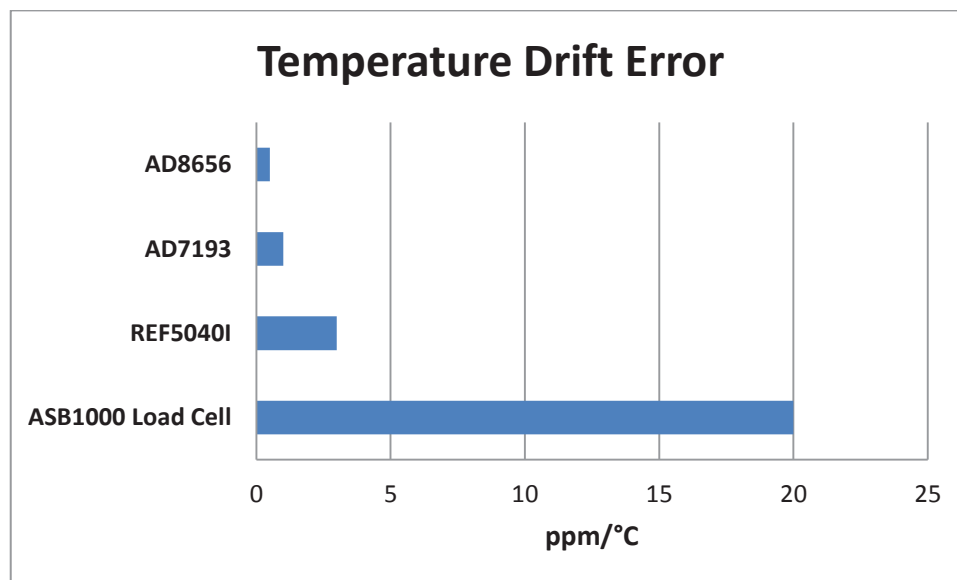
placed under a magnifying glass to ensure no lines were shorting out and all pads were soldered on correctly. As there was no software written for the AD7193 at this time, the AD7193 would be tested later.

The completed prototype breakout board can be seen in Figure 19: Completed Prototype Breakout Board PCB. Left: Underneath of Breakout Board PCB. Right: Top of Breakout board and plugged into Arduino Uno, and four load cells connected..



**Figure 19: Completed Prototype Breakout Board PCB. Left: Underneath of Breakout Board PCB. Right: Top of Breakout board and plugged into Arduino Uno, and four load cells connected.**

The contributed temperature drift error by the four main components was found in their respective datasheet and can be seen in Figure 20: Temperature Drift of Various Components. It can be seen that the largest contribution is from the load cell (20ppm/°C). By adding the temperature drift of all four components, a total combined temperature drift of 24.5ppm/°C was calculated. This means that for every degree Celsius the temperature changes, an error of 24.5 grams will be produced. This is not of much concern as temperature varies slowly and the load cells will be calibrated automatically every time the platform is powered.



**Figure 20: Temperature Drift of Various Components**

### 3.3.2.5. Validation of SPI Communication to AD7193

After the prototype breakout board was completed, software had to be written to communicate with the AD7193 to validate whether communication to the AD7193 was working or not.

By inspecting the datasheet, it was found that when the AD7193 powers up or resets, the ADC is in the default state waiting for a write operation to the communications register. It was



also found that the ID returned by the AD7193 should be 0x02 on power-on/reset. A bitmask of 0x0F was therefore used to test whether the ID returned from the AD7193 was in fact 0x02. A simple program was then written to get the ID from the AD7193.

```

//=====
// A simple function to read the ID-Register of the device
// Expect a value of 0xX2 (xxxx 0010)
//
//=====
void read_ID_Register()
{
    SPI.transfer(0x60);    // Write to the communication register that
                          // the next operation will be a read

    // Select the ID-register
    while (digitalRead(DOUT) == 1); // DOUT goes low to indicate the
                                  // completion of conversion

    ID = SPI.transfer(0xFF);    // Store the value read from
                              // AD7193 in the variable called
                              // ID
}

//=====
// A simple function to check whether the ID is valid or not
//
//=====
void valid_ID()
{
    if (ID & 0x0F == 0x02)    // Do bitmasking to test whether
                          // result is 0x02 or not
        Serial.println("Valid ID");
    else
        Serial.println("Invalid ID");
}

```

**Figure 21: Code snippet to verify communication with AD7193 by reading ID-Register**

The result obtained from the ID register is stored in a variable called “ID”. The result is then verified, and a message is printed on the serial monitor to indicate whether a valid or invalid ID was found. It was found that a valid ID was returned and printed on the serial monitor; this confirmed that communication was happening successfully.

Now that it was confirmed that communication between the Arduino Uno and the AD7193 was happening successfully, the datasheet of the AD7193 was inspected again to see how to configure each register.

Register	Register Size	Hex	Binary	Configuration
Mode Register	24-bits	0x18 0x20 0x20	0001 1000 [23:16] 1010 0000 [15:8] 0001 0000 [7:0]	<p>[23:16] Put the ADC in continuous conversion mode. In continuous conversion mode, the ADC continuously performs conversions and places the result in the data register.</p> <p>The DAT_STA bit is set and the contents of the status register are transmitted along with each data register read. This function is useful when several channels are selected because the status register identifies the channel to which the data register</p>

				<p>value corresponds. The internal 4.92MHz clock is selected and Pin MCLK2 is tristated. Fast Settling Mode is Disabled.</p> <p>[15:10] The sinc<sup>3</sup> is selected. The benefit of this is lower settling time. The parity bit is enabled. Set clock divide-by-2, this bit must be set when AV<sub>DD</sub> is less than 4.75V. Single cycle conversion disabled. Notch Filter disabled.</p> <p>[9:0] – Filter output data rate select bits Because the 4.92MHz clock is selected, the output rate of the AD7193 can vary from 4.69Hz to 4.8kHz. An output rate of 300Hz was selected.</p>
Configuration Register	24-bits	0x00 0x0F 0x17	0000 0000 0000 1111 0001 0111	<p>[23:16] Chop is disabled External reference applied between REFIN1(+) and REFIN(-). Configure AD7193 to have four differential analog inputs.</p> <p>[17:8] These bits select which channels are enabled on the AD7193. Enable CH3, CH2, CH1 and CH0.</p> <p>[7:0] Disable burnout current. Disable the reference detect function. Enable the buffer on the analog inputs, allowing the user to place source impedances on the front end without contributing gain errors to the system. Polarity bit is cleared, meaning bipolar operation is selected. The gain is set to a value of 128.</p>
GPOCON Register	8-bits	0x64	0100 0000	<p>[7:0] Set bridge-down switch BPDSW to AGND. This is so REFIN- has a reference point.</p>

Because several analogue inputs were enabled, the datasheet suggested setting the DAT\_STA bit in the mode register to 1. This enables the contents of the status register to be appended to each 24-bit conversion in the data register. The four LSBs of the status register (CHD3 to CHD0) identify from which channel the conversion originated [32].

See Figure 22: Flowchart of AD7193 Programming for a flowchart of the AD7193 program logic.

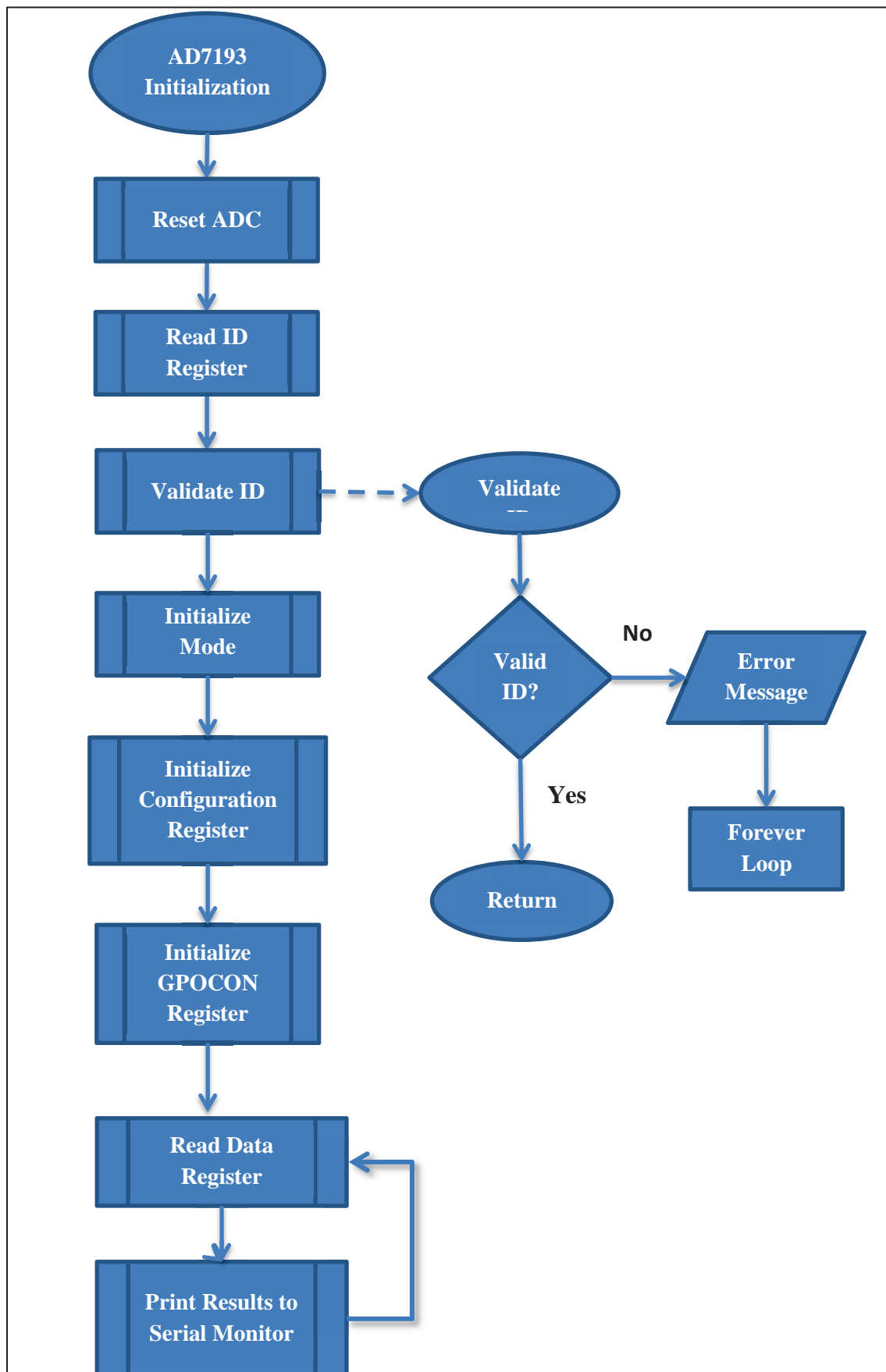


Figure 22: Flowchart of AD7193 Programming

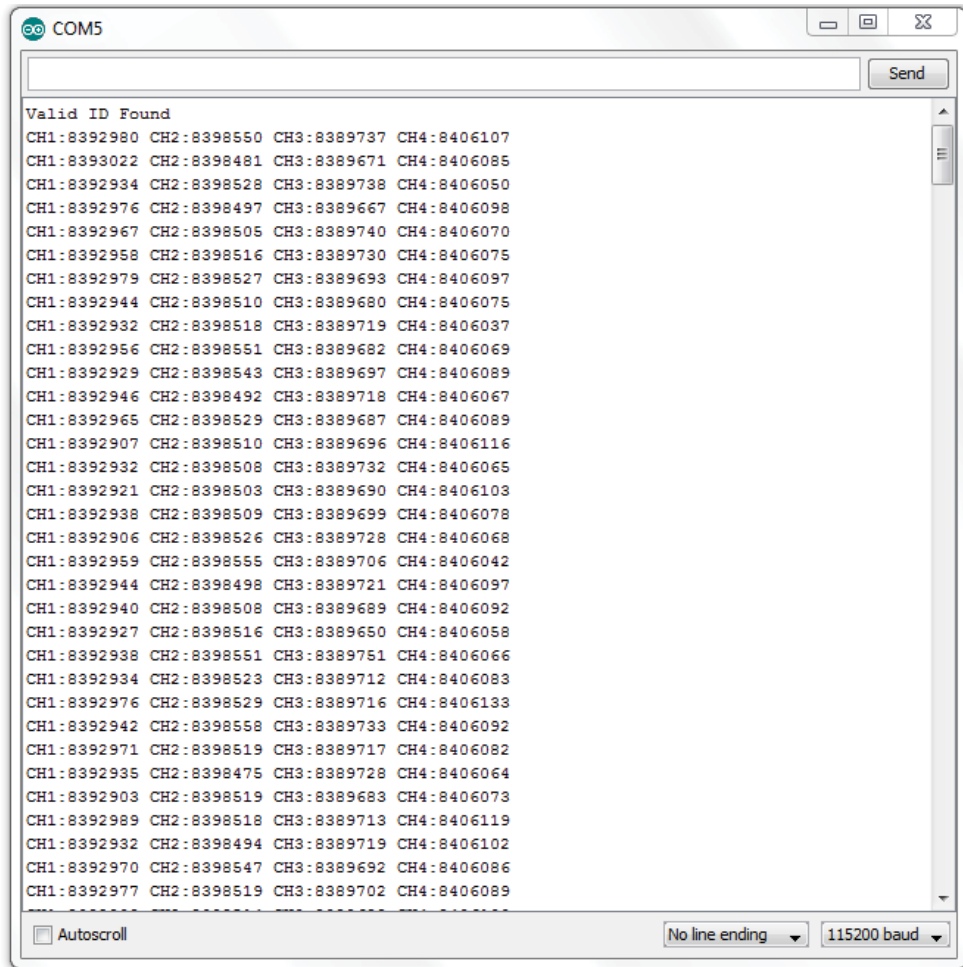


Figure 23: Output from Arduino Uno printed on Serial Monitor

It can be seen from Figure 23 that data was being successfully transmitted from the Arduino Uno to the computer via the serial line. A MATLAB program was then written (see Section 4.1.2.3) to scale (in kilograms) the raw digitized values from the AD7193 (running at a sampling rate of 150Hz) and to plot it in real-time. A known weight of 2kg was applied to each load cell, the results of the MATLAB program can be seen in Figure 24: 2kg Weight Placed On Each Load Cell to Test Accuracy.

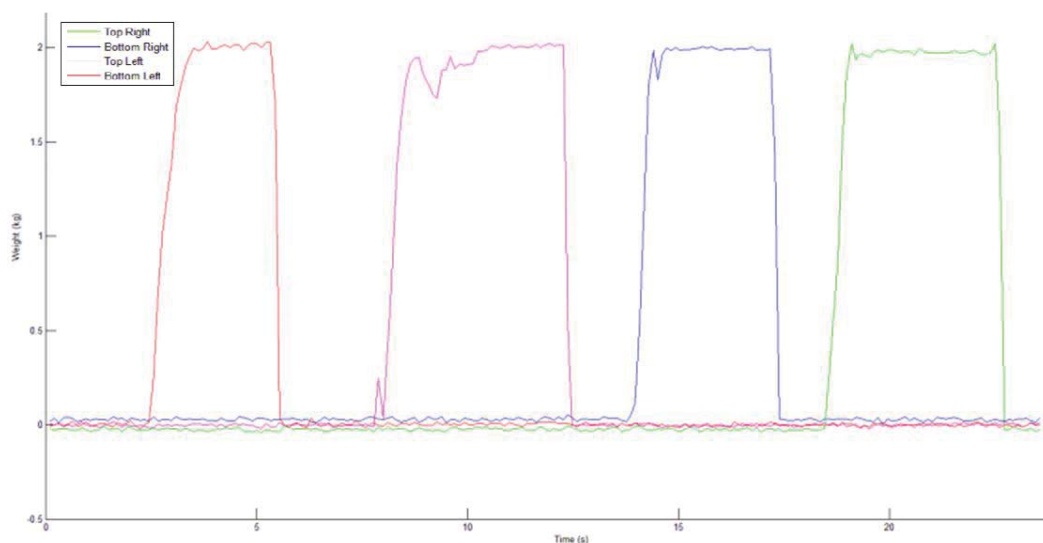


Figure 24: 2kg Weight Placed On Each Load Cell to Test Accuracy

### 3.3.3. Communication

As the overall system was to consist of four individual sections, some form of communication needed to take place. Two types of communication interfaces were investigated that would allow for connecting multiple devices (four slave microcontrollers – each section having its own microcontroller, and a master microcontroller to control all communications and a computer at an unknown distance), with these being RS-485 and I<sup>2</sup>C (see Section 2.3.4 and Section 2.3.6 respectively).

RS-485 was chosen over I<sup>2</sup>C for this system mainly because RS-485 has superior noise immunity, faster data transfer speeds, supports greater distances, and is an industrial standard. RS-485 line drivers/receivers are required (to increase voltage levels) for each device operating on the data lines. The MAX487 by Maxim Integrated was found to be suitable for the task at hand. The MAX487 transceiver has two communication lines (A and B), two switchable pins to set whether the transceiver should be in transmit or receive mode, and two serial data lines.

A small PCB was designed containing three MAX487 components for testing purposes (to connect two Arduinos and one USB to TTL UART Bridge). The USB to TTL UART Bridge was connected to a computer to see what was being transmitted on the data line. The enabled pins (Receiver Output Enabled and Driver Output Enabled) to be tied together; this was done so only one pin is required from the microcontroller to set the device in transmit or receive mode.

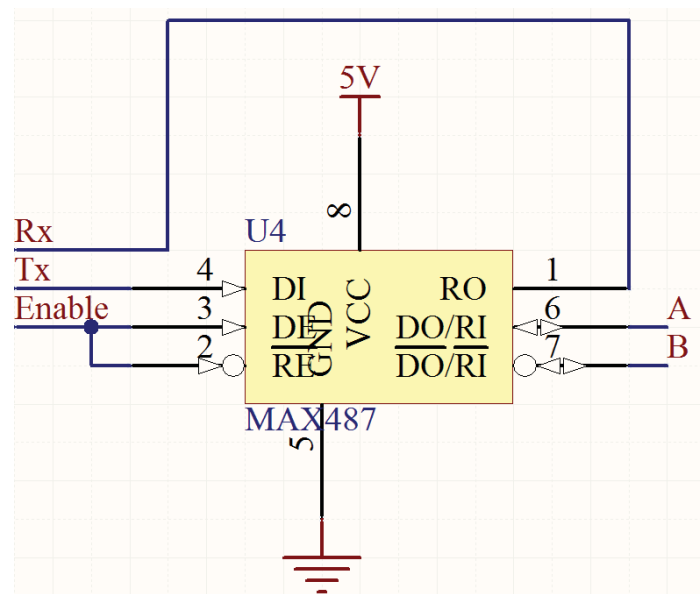


Figure 25: MAX487 Connection Diagram of MAX487

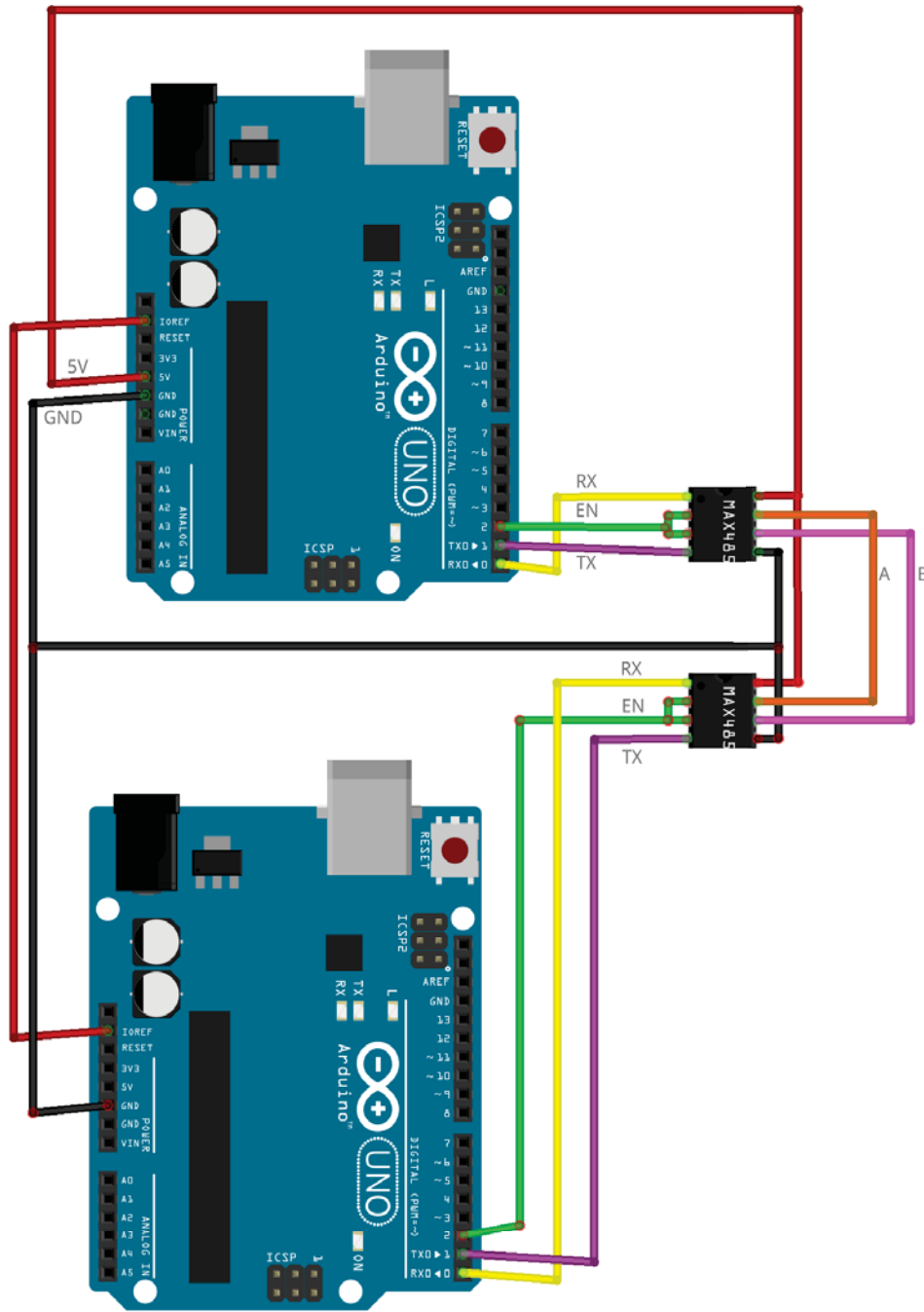
#### 3.3.3.1. One-Way Communication Testing

The main reason for doing one-way communication testing first was to become familiar with driving the MAX487 chip. The required resources for this test are:

- Two Arduino Uno boards
- Two MAX487 ICs

Each Arduino Uno board is connected to its own MAX487 IC. The common lines shared between the Arduino Uno boards and MAX487 ICs are the A and B lines, 5V line and ground. One of the Arduino Uno boards was configured to act as the master device while the other Arduino Uno was configured to act as the slave device (always receiving data).

A connection diagram can be seen in Figure 26: Connection Diagram for One-Way Communication.



**Figure 26: Connection Diagram for One-Way Communication**

To verify that the slave is receiving data from the Master device, it will simply flash an LED when it receives the expected character. Flowchart diagrams for the basic operation of the one-way communication between the master and slave devices are as shown in Figure 27: Flowchart of One-Way Communication between Master and Slave. An oscilloscope was connected to inspect the communication lines A and B when the character 'A' was being transmitted. By placing the probe of the oscilloscope on line A (see Figure 28: Character 'A' being displayed on the non-inverting (left side) line and inverting (right side) line.), a signal in the form of 0100 0001 should be present (non-inverted signal) and on line B (see Figure 28: Character 'A' being displayed on the non-inverting (left side) line and inverting (right side) line.), the inverted signal should be present. This is to be expected when using RS-485 communication as it is a balanced line and is the reason why it has high noise immunity.

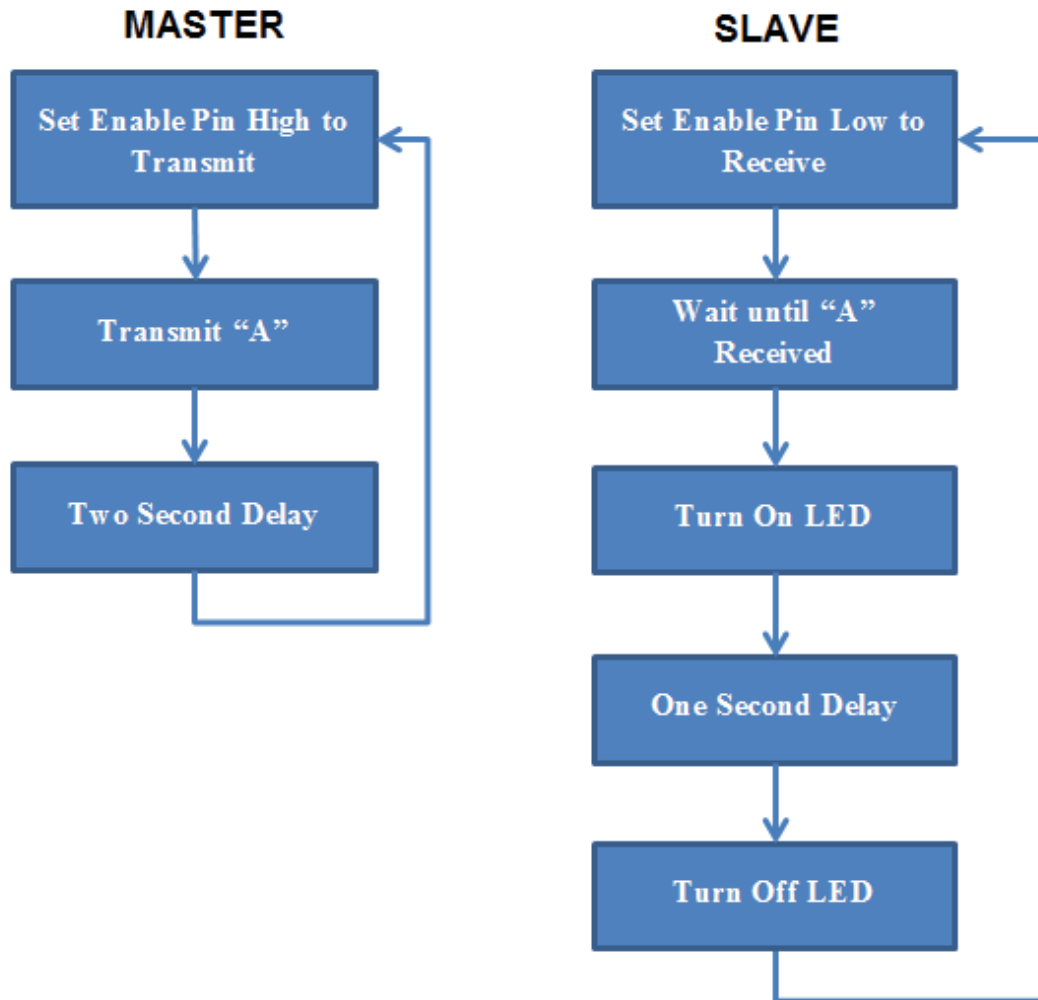


Figure 27: Flowchart of One-Way Communication between Master and Slave

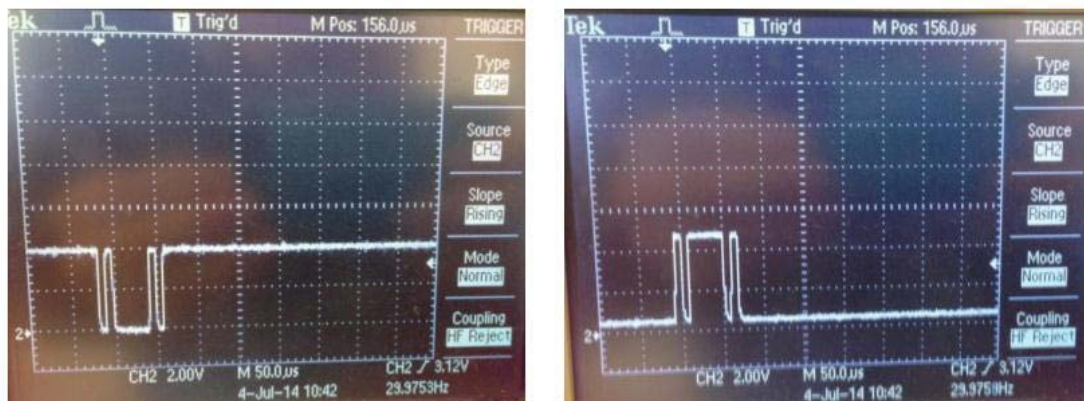


Figure 28: Character 'A' being displayed on the non-inverting (left side) line and inverting (right side) line.

### 3.3.3.2. Two-Way Communication Testing

The main purpose of this test was to get the Master Arduino Uno and the Slave Arduino Uno to communicate with each other. The USB to UART TTL Bridge with its own MAX487 was used to spoof line A and line B to observe whether communication between the two devices was happening successfully. The results were displayed on a Serial Monitor.

The wiring diagram is shown in Figure 29: Two-Way Communication Wiring Diagram. It is up to the user to decide which Arduino Uno they want to program as the Master and Slave device.



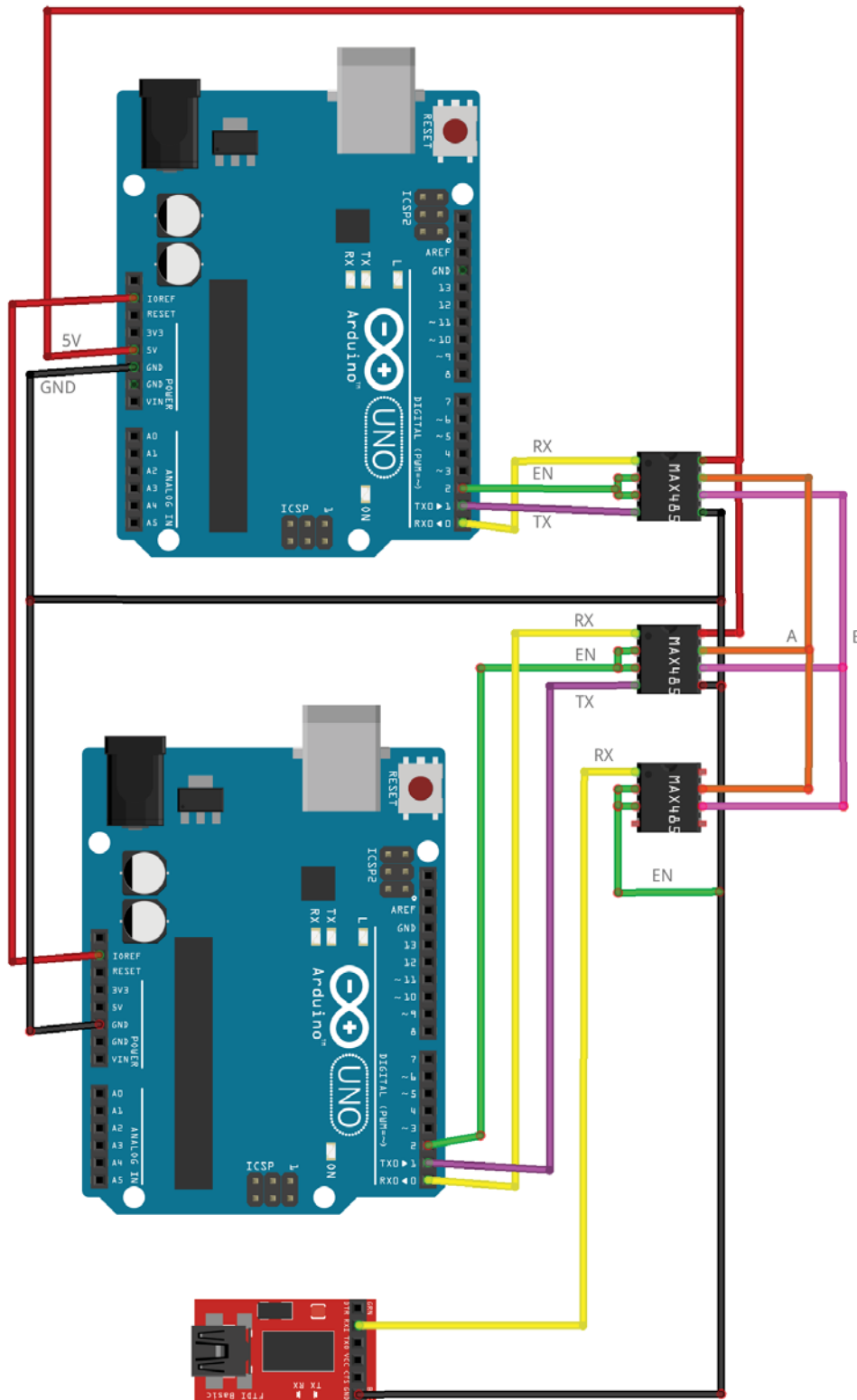


Figure 29: Two-Way Communication Wiring Diagram

To verify whether the Master and Slave devices are communicating with each other, the Master sends a character "S" to the slave device, the slave then prints "SLAVE". The slave should then respond with the character "M". The Master will then receive this and print "MASTER" and respond with "S". This sequence continues until the power is turned off. Flowchart diagrams for the basic operation of the two-way communication between the

master and slave devices as shown in Figure 30: Flowchart of Two-Way Communication between Master and Slave.

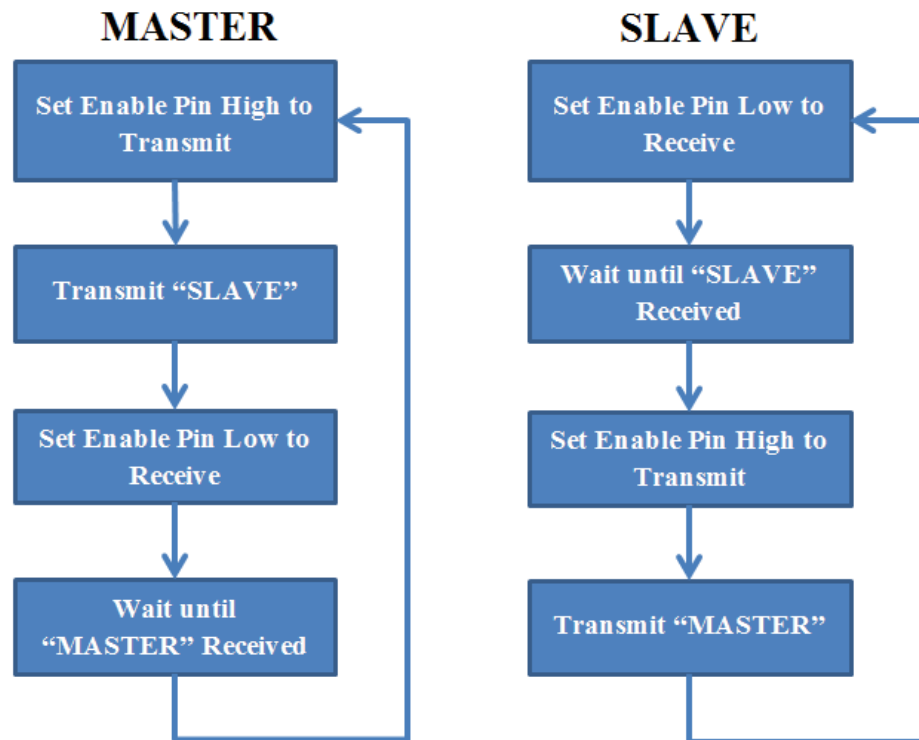


Figure 30: Flowchart of Two-Way Communication between Master and Slave

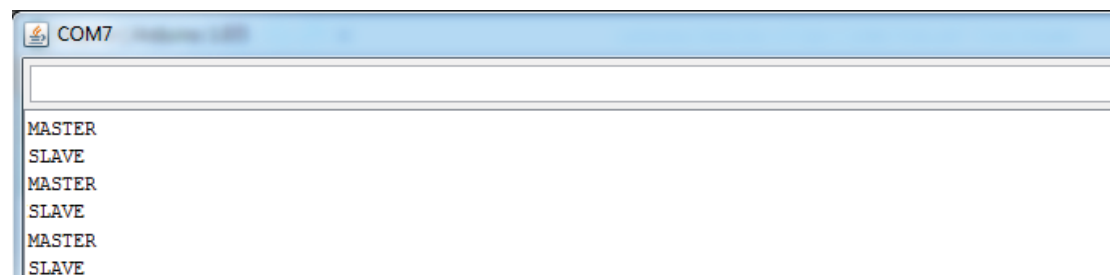


Figure 31: Proof of Two-Way Communication Working

Once it was confirmed that the two-way communication was working fine, the prototype breakout board was plugged into the Slave device and the program was modified for the Master to request data from the Slave, which will respond with the values from the load cells. The results were printed to the Serial Monitor as shown in Figure 31: Proof of Two-Way Communication Working.

After it was proven that two-communication was working, the breakout board designed in Section 3.2.3.4 was plugged into the slave device. The master and slave device code was modified in such a way that the master could query the slave device and the slave device would respond with the data of the AD7193.

### 3.3.4. Final Prototype PCB

The results obtained from the first prototype platform section (see Section 5.2.1 for more details) were very successful. It was also possible at this stage to get two-way communication working between a single Master device and a Slave device which reports load cell values. Therefore, it was decided to design a final Prototype PCB to be housed inside each section of the platform because:

- This will make repairing and fault find easier as each section will have its own unique ID.

- Sections are able to be assembled and tested individually.
- There will be less redesign work compared to a single PCB interfacing with 16 load cells.

The final manufactured prototype PCB (see Figure 34: Bottom View of Final Prototype PCB Assembled and Figure 35: Top View of Final Prototype PCB Assembled) to be housed inside each section was designed to house its own ATmega328 microcontroller acting as a slave device. This is the same microcontroller used on the Arduino Uno, which meant minimal changes to the code already written for the hardware. As the platform will be used in a harsh farm environment, the electronics will need to be waterproof, especially because high-pressure hoses are used to wash the platform after the cows walked over it. Therefore, the PCB was designed to be housed inside a waterproof box (see Figure 36: Final Prototype PCB Mounted inside Waterproof Case with Load Cell Wiring Connected). The load cells connect to the PCB via waterproof cable glands and a 4-wire power and communications cable is connected with an IP-68 plug and socket for easy removal. A daisy chained parallel configuration was used so each PCB can simply connect the next PCB using one cable.

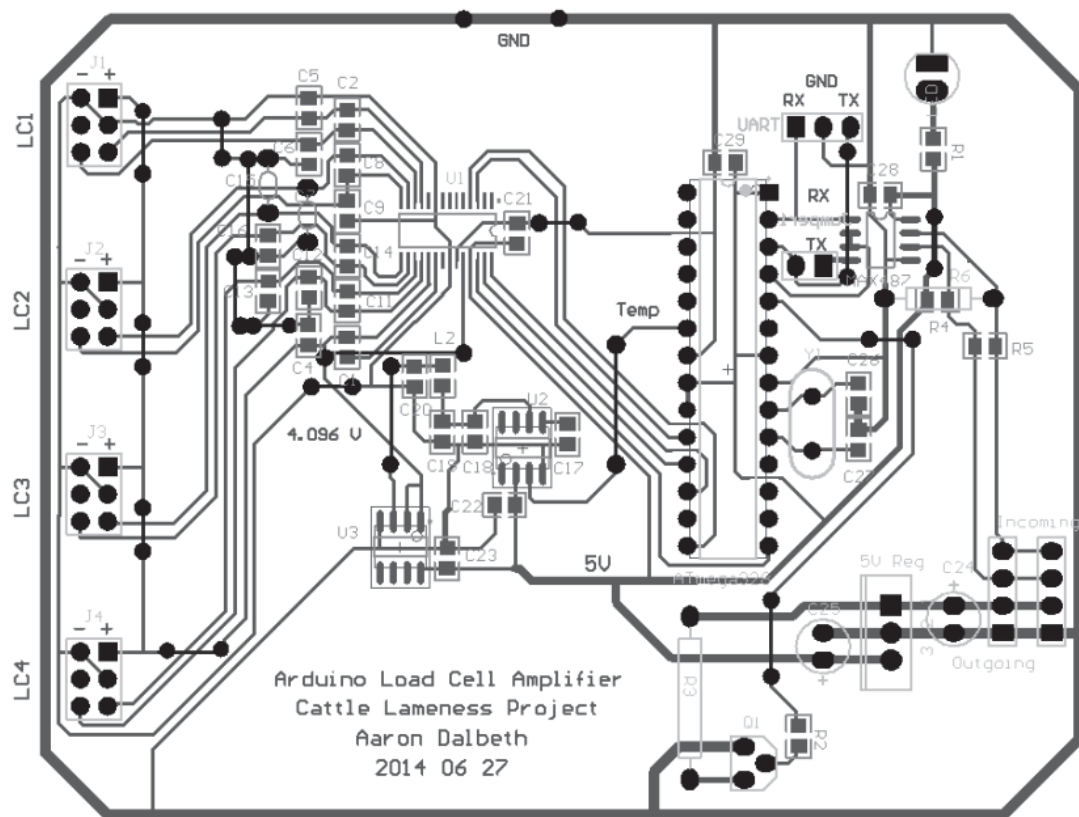


Figure 32: Final Prototype PCB Designed by Aaron Dalbeth

Some new features were added to the final PCB compared with the initial breakout board. These include:

- An LED to indicate whether the PCB is being supplied with power or not.
- A transistor controlled 1W heating circuit as condensation might occur inside the waterproof box.
- As it has a temperature line, a REF5040 was connected to the ATmega328 to monitor the temperature inside the enclosure.
- A 5V voltage regulator to ensure the board can support a wide range of voltage ranging from 7VDC to 36VDC.
- A MAX487 IC for RS-485 communication.
- Daisy-chain communication.
- Load cell headers were given more space
- A 16MHz crystal oscillator as the clock source for the ATmega328.
- Fail-safe biasing arrangement for RS-485 communication.

The schematic of the final prototype can be found in Appendix 3: Final PCB Schematic

As an additional feature, a heater circuit was added to help remove any moisture that might occur inside the waterproof box. A small 1W resistor heating circuit was used to increase the temperature inside the waterproof box. Assuming the platform is being powered from a car battery (typically 12V) and 1W heating is required, then the required resistance was calculated to be  $144\Omega$ . Two standard  $270\Omega$  0.5W resistors will be used to create the 1W heating “element” in parallel to create an effective resistance  $135\Omega$ , meaning the power dissipated through the resistors would be 1.06W and current consumed when active would be 89mA.

It was recommended when using RS-485 communication to apply fail-safe biasing to terminate the furthestmost point. Three resistors are used to create the fail-safe biasing arrangement, as seen in Figure 33: Fail-safe Biasing Arrangement. The main purpose of these resistors are to remove undefined state on a standard RS-485 communication line and replace this with a differential voltage between  $\pm 200\text{mV}$  so no false triggering can occur. This arrangement was included on the final prototype PCB; however, these resistors are only soldered onto the PCB furthest away from the Master device as discussed in Section 2.3.4.3.

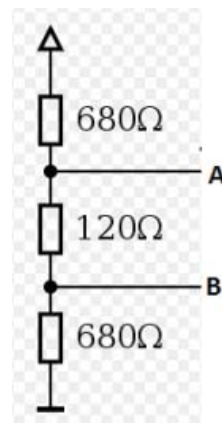


Figure 33: Fail-safe Biasing Arrangement

Four of these PCBs had to be fabricated and assembled. The PCBs were assembled and tested in multiple stages to ensure components were functioning as intended. The testing stages were the same as the initial prototype breakout board, apart from the added components (DIP socket for ATmega328, voltage regulator, LED, heating circuit and MAX-485) having to be soldered on. To test whether communication was working properly, each microcontroller was flashed with a program, plugged back into the socket where the microcontroller plugs into, and the Master microcontroller would then query the slave device to see whether it got any information from the AD7193.

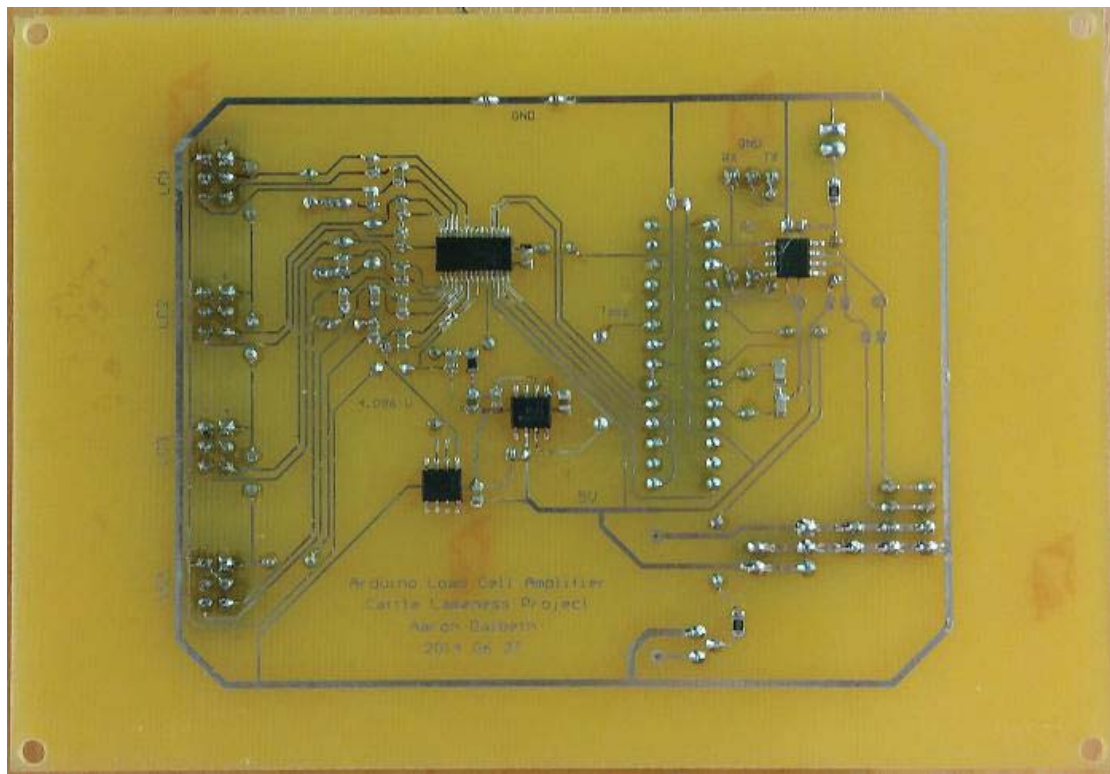


Figure 34: Bottom View of Final Prototype PCB Assembled

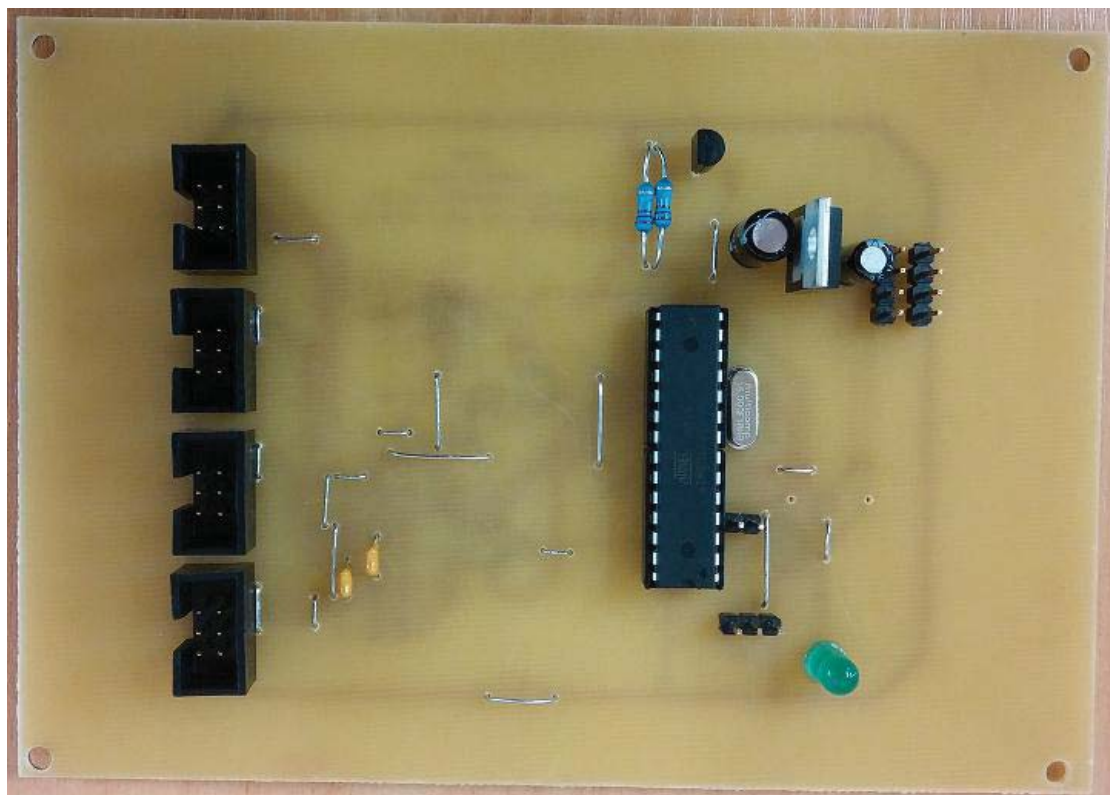
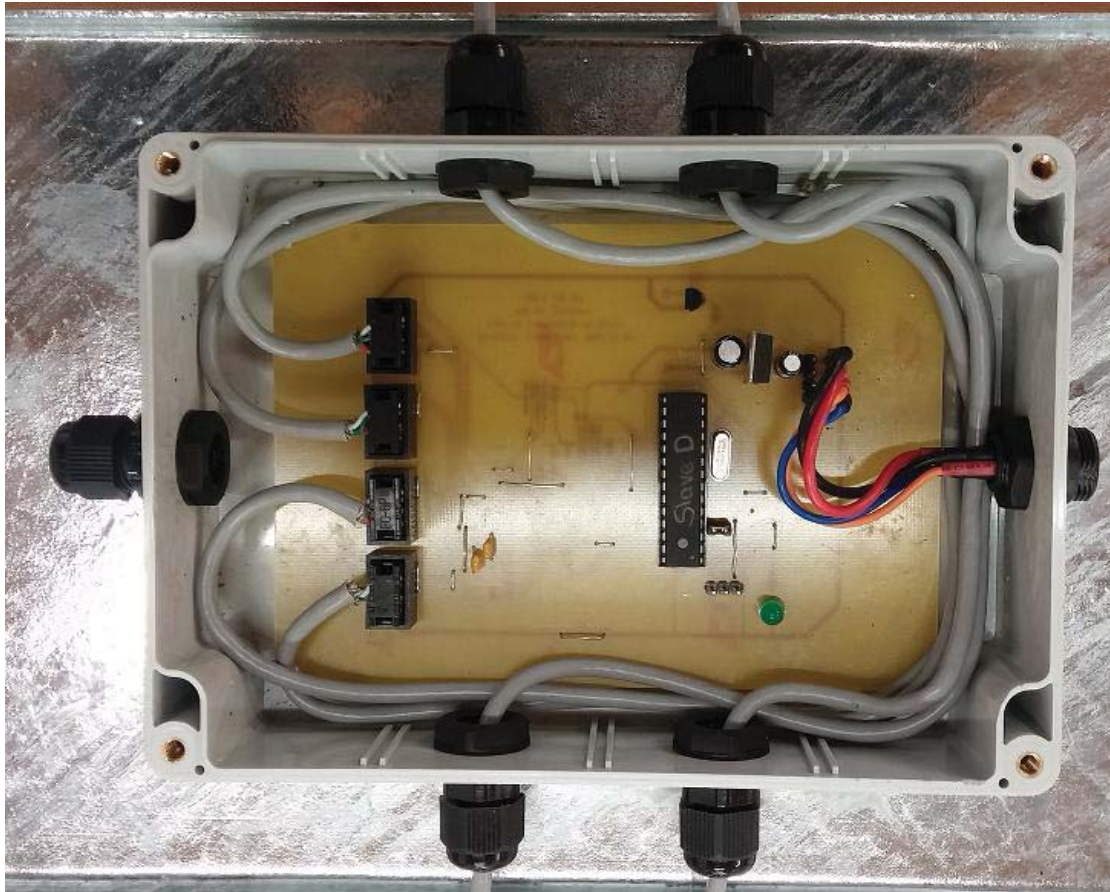


Figure 35: Top View of Final Prototype PCB Assembled





**Figure 36: Final Prototype PCB Mounted inside Waterproof Case with Load Cell Wiring Connected**

A breakout board (see Figure 37: Breakout Board Connected to Arduino Mega Showing USB Cable which is connected to the PC, power adaptor to power system and security cable to carry power and data bus lines) containing a MAX487 IC was also designed to plug into the master microcontroller, tidying up the electronics. A power adaptor is connected to the DC power jack on the Arduino Mega. The power can be accessed via the  $V_{in}$  pin on the Arduino Mega and is used to power the platform.



**Figure 37: Breakout Board Connected to Arduino Mega Showing USB Cable which is connected to the PC, power adaptor to power system and security cable to carry power and data bus lines**

A 4-core security cable was used to carry the power and data bus lines (see Figure 38: 4-Core Security Cable showing the connection which connects to Master Breakout Board and Connection to first slave device) from the master to the first slave device.



**Figure 38: 4-Core Security Cable showing the connection which connects to Master Breakout Board and Connection to first slave device**

## Chapter 4

### Software Development

Various software tools were created to help detect lameness; this included capturing the load cell values, processing them, and displaying the weight, position and duration. The data is transferred on the RS-485 interface between the master microcontroller and the slave devices, and once the data is obtained from a slave device, the master transmits the data to the computer via RS-232. The main tasks the software had to perform were:

- Able to capture the ADC values from each section and translate it into useful data.
- Remove offset on individual load cells.
- Transfer data from the master microcontroller to the computer.
- Plot the force vs time signal in real-time.
- Plot the centre of pressure on the platform in real-time.
- Record data for post-processing; calculating average weight, stride length, hoof duration, rendering data to a video file and graphing.
- Able to adjust the sampling rate of the AD7193.

#### 4.1. Microcontroller Programming

The microcontroller programming was done in the Arduino IDE, and is hardware-orientated. The main purpose of the microcontroller programming was the capability to interface with the AD7193 (get a digitized signal of the load cell) and the MAX487 (transmitting the data via RS-485) devices. There are three types of communication interfaces used, with these being: SPI (communication between ATmega328 and AD7193), RS-485 to communicate between master and slave devices, and serial communication between the master and computer.

##### 4.1.1. AD7193 Programming

As mentioned in Section 3.2.3.2, there was no software available for the AD7193 and the complete process of configuring, reading and communicating with the AD7193 had to be developed. The overall layout of the program to capture data from the load cells can be seen in Figure 39: Overall Program for AD7193 (Left), SPI Initialization Process (Middle), I/O Initialization (Right). When the program starts, there are multiple initializations that take place.

The first is the serial initialization, where the baud-rate is set for serial data transmission.

When the SPI initialization takes place, SPI communication is started and the data mode is set, which was found to be mode 3 after inspecting the datasheet. The clock divider was set to 4MHz, and the bit order was set to output and input most significant bit first.

Next was the I/O initialization, where the SS pin on the ATmega328 was set to be an output, the SS of the AD7193 was pulled low and kept low as this was the only device on the SPI bus, and always selected. The sync pin of the AD7193 was set high as there was no need to synchronize it with other devices.



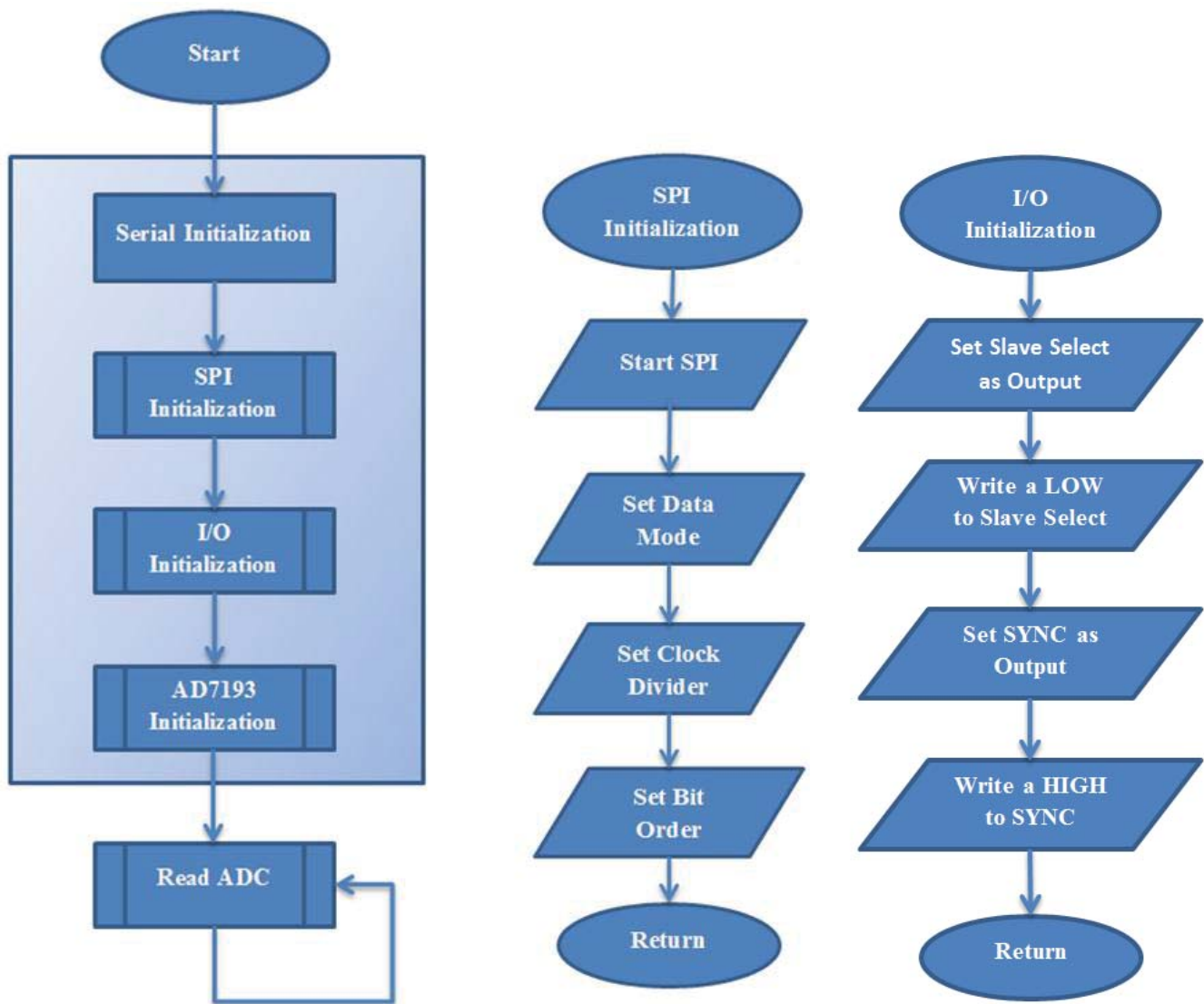
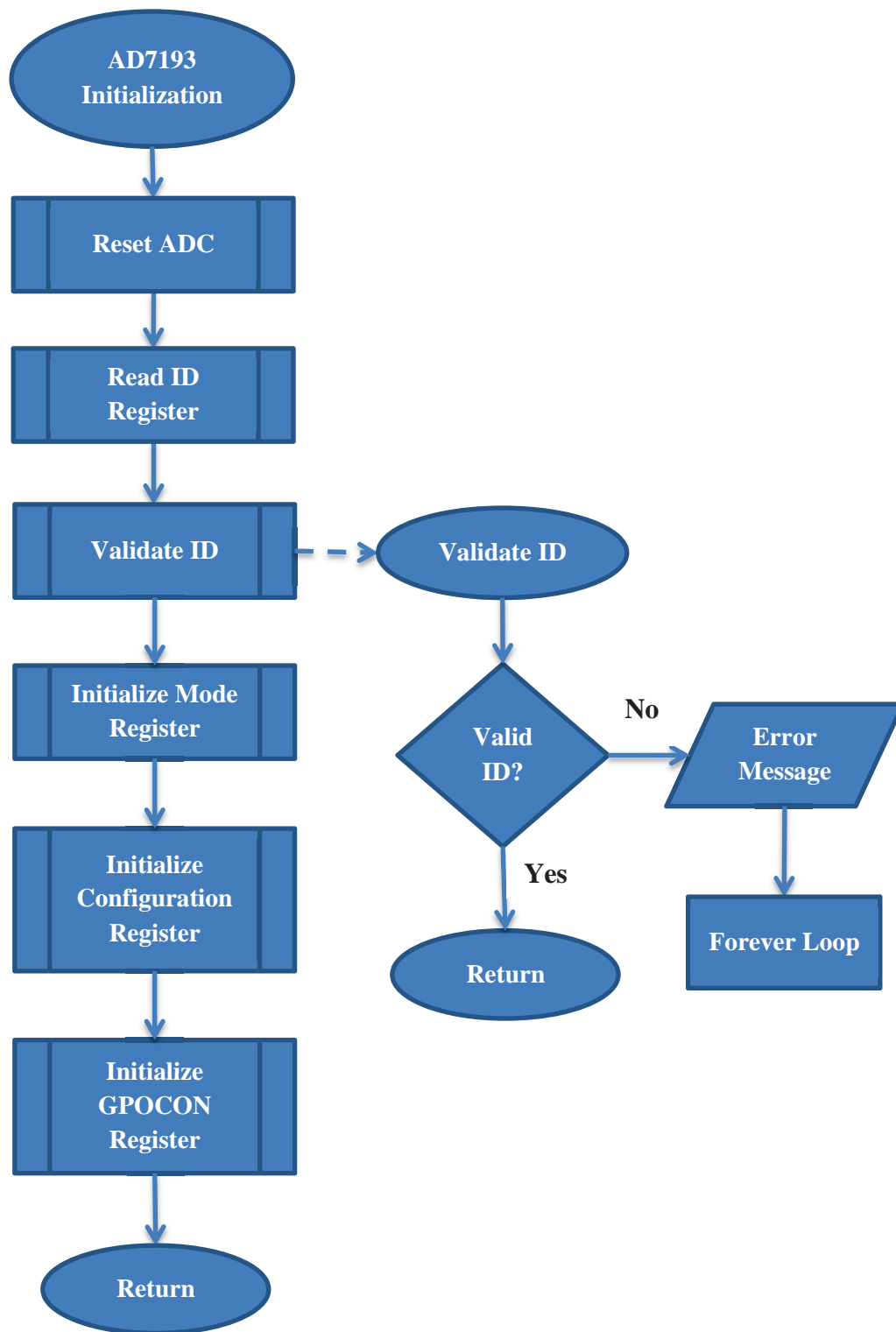
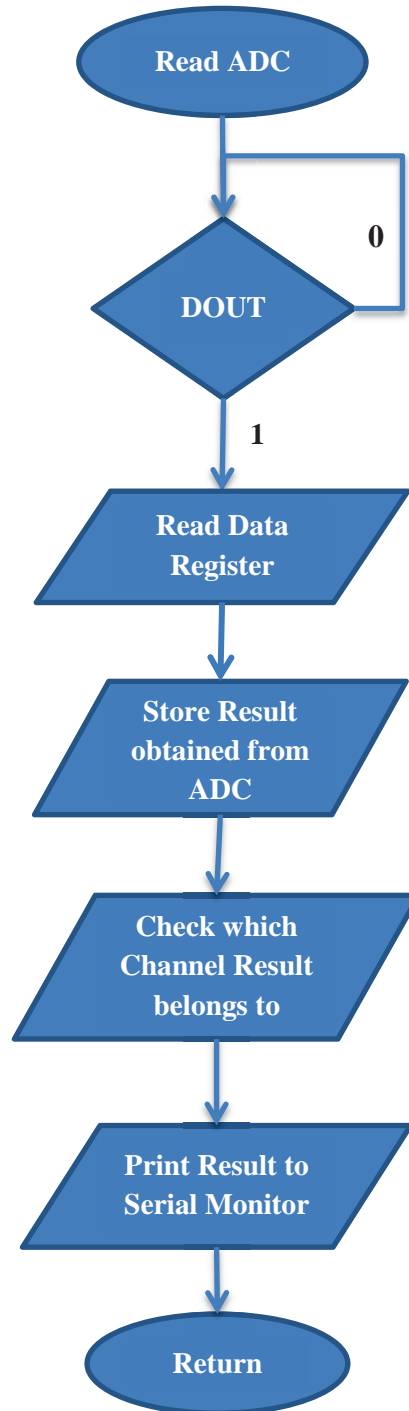


Figure 39: Overall Program for AD7193 (Left), SPI Initialization Process (Middle), I/O Initialization (Right)



**Figure 40: AD7193 Initialization Process**

Finally, the AD7193 initialization was done (see Figure 40: AD7193 Initialization Process). Once all the initialization was completed, the only part left was to read the data register of the AD7193 and print the results to the serial monitor on the computer (see Figure 41: Read ADC and Transmit to Computer).



**Figure 41: Read ADC and Transmit to Computer**

The true sampling rate when multiple channels are used depends on the number of enabled channels. Equation 1 is used to determine the output frequency per channel:

$$\text{Sampling Rate} = \frac{\text{AD7193 Data Rate}}{\text{Number of Enabled Channels}} \quad (1)$$

Now if for example all four channels are enabled and the AD7193 output data rate is set to 300Hz, the actual sampling rate per channel is 75Hz.

#### 4.1.1.1. Communication Protocol

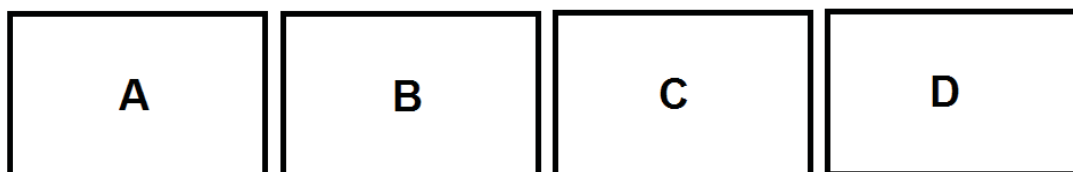


Figure 42: Platform with Four Sections

The master will query each section in turn i.e. A, B, C, D, A, B, C etc, (this is also known as round-robin) as only one device is able to communicate on the RS-485 bus at a time. The individual sections will always be in receiving mode until the master transmits a packet; at which stage the corresponding section will receive the packet and go into transmit mode, respond to the master with the corresponding data, and then go back into receiving mode. Therefore, it can be seen that it was important to have some protocol between the master and slave devices.

Each slave section has four load cells connected to it. The master will query the section, where the slave will respond to the master with the corresponding data. The master must be capable of selecting individual sections and set/query the section to perform certain functions, this include:

- Being able to set sampling rate of the AD7193.
- Being able to request data from slave device; this can be the digitized values of the load cells, or the current temperature the REF5040 is reporting.
- Being able to turn on the heating circuit if required.

Various pre-existing RS-485 protocols were investigated but were found to be complex. Consequently, it was decided to make our own custom data transfer protocol as this allowed us to design a protocol specific for this system. The designed protocol was named the AJ convention (after our first names) and a packet consisted of three characters.

A diagram of an AJ packet transmitted by the master microcontroller to the slave microcontrollers is shown below and is transmitted as ASCII characters:

Slave ID	Command	Termination
----------	---------	-------------

Where:

- Slave ID is either, A, B, C, D or E, with E being all slaves selected
- Command
  - o R = Read Data (Slave will respond with digitized values).
  - o Fx = Change Frequency (set the sampling rate of AD7193), x would specify the sampling rate (see Table 3: Various Modes to Set Sampling Rate of AD7193 on Page 44). See Example 2 for more details.
  - o T = Read REF5040 Temperature.
  - o H = Turn heating circuit on or off.
- Termination is simply a new-line character '\n'

A diagram of an AJ packet transmitted by a slave microcontroller to the master microcontroller is shown below and is transmitted as ASCII characters:

Slave ID	Data	CRC-32	Termination
----------	------	--------	-------------

Where:

- Slave ID is the ID of the device that is responding to the master.
- Data, this is either the digitized values of the load cells (CH1:xxx CH2:xxx CH3:xxx CH4:xxx), where xxx is the AD7193 values or the temperature of the REF5040.
- CRC-32 used for data integrity (CRC only gets transmitted when AD7193 reports values)
- Termination is simply a new-line character '\n'

Every time a slave responds to the master, it also transmits the character 'M'. The master microcontroller uses this as a mechanism to query the next slave device.

Several examples are given how the communication protocol works for the various commands.

**Example 1:** In this example, the master will request data from each slave in turn.

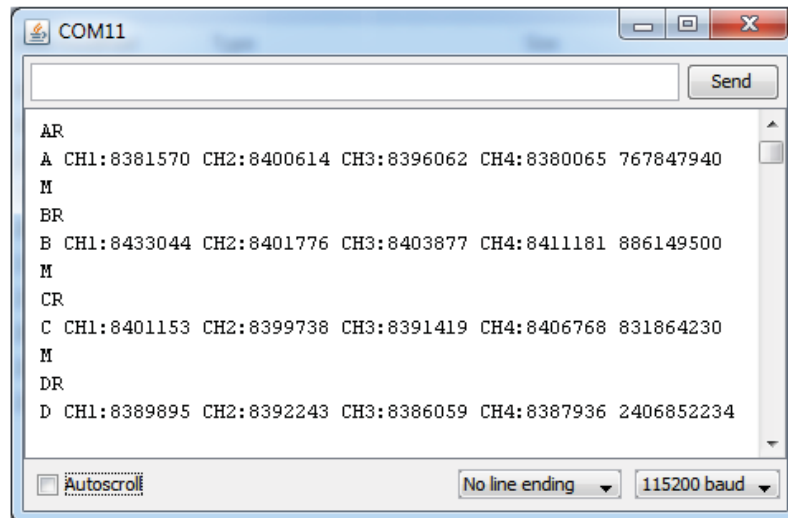


Figure 43: Master Requesting AD7193 Values from All Slave Devices

In Figure 43: Master Requesting AD7193 Values from All Slave Devices the master transmits the command "AR", which means select slave A and read data. The slave device responds to the master with the AD7193 values and also transmits the character 'M'. Once the character 'M' is received, the master will select the next slave device and request data from it, in this case slave B. This continues until it reaches Slave D. Once data has been received from slave D, the master will request data from Slave A again and will follow the same pattern described earlier. The slave reports its own ID so it makes it easier to know what slave device the data belongs to when it comes to processing the data on the computer. Note that when the Master transmits the data to the computer, it only transmits the data the slave responded with (see Figure 53: Data Master Transmits to Computer via Serial Port).

**Example 2:** In this example, the master will set the frequency of each slave device (see Figure 44: Master Sending Commands to Set the Frequency of Slave Devices).

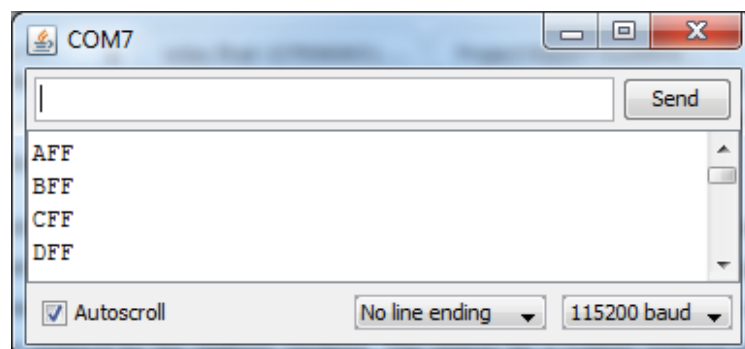


Figure 44: Master Sending Commands to Set the Frequency of Slave Devices

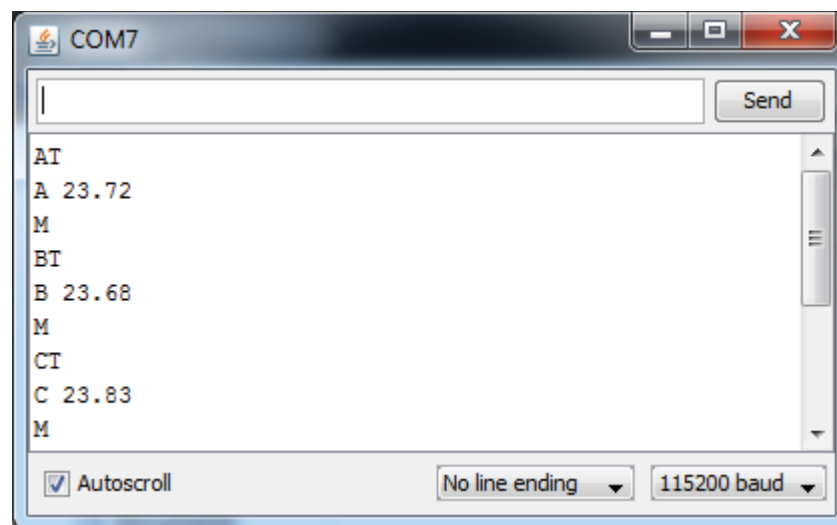
If the master transmits the command "AFF", "BFF", "CFF" and "DFF", this sets the sampling rate of the AD7193 of Slave A, B, C and D to Mode F (see Table 3: Various Modes to Set Sampling Rate of AD7193), which is a frequency of 2400Hz. In this case it would have been able to simply send the command "EFF" as all slaves are being set to use the same sampling rate, but was done separately to show it is possible to set the sampling rate of each section to

a different sampling rate. If the user for instance wanted to set the sampling rate to 960Hz of section C, the command “CFE” would be transmitted from the master.

**Table 3: Various Modes to Set Sampling Rate of AD7193**

Mode	Sampling Rate of AD7193
A	50Hz
B	60Hz
C	150Hz
D	300Hz
E	960Hz
F	2400Hz
G	4800Hz

Example 3: In this example, the master will request the current ambient temperature of the PCB enclosure from the slave devices.



**Figure 45: Master Requesting Temperature from Slave Devices**

The slave device gets the temperature from the REF5040's analogue temperature pin which is converted to a digital value by the ATmega328's internal 10-bit ADC. The voltage value is scaled by the conversion factor of 2.3mV/°C which can be found in the REF5040's datasheet. Knowing the temperature is not an essential component of the system, however it could be useful information.

The communication protocol that was designed was extensively tested to ensure communication happens between the master device and slave devices at all times when the system is powered. Each slave device is given its own unique ID, and its ID was written in pencil on top of it to distinguish between the slave devices. The four PCBs as seen in Figure 46: RS-485 Test Setup of Master Communicating with Four Slave Devices are the ones that will be used in the system. A bench-top power supply was used to power the electronics, the power supply was set to 12V and the current limit was set to 400mA. The power supply was connected to the first PCB (Slave A), because of the daisy-chain configuration, the other PCBs was also being powered. The communication lines are also daisy-chained. The Arduino Mega 2560 is connected to its own MAX487 IC and connected to the A and B lines; this is done so the master can communicate with the slave devices. An Arduino Mega 2560 was chosen as it features three serial communication ports. Only two of the serial ports are used; one port is used to communicate with the slave devices, and the another port is used to transfer the data to the computer for further processing. To see communication happening between the devices, a serial monitor program such as Terminal was used. This was done to fault find and observe the packets being sent between the master and slave devices.

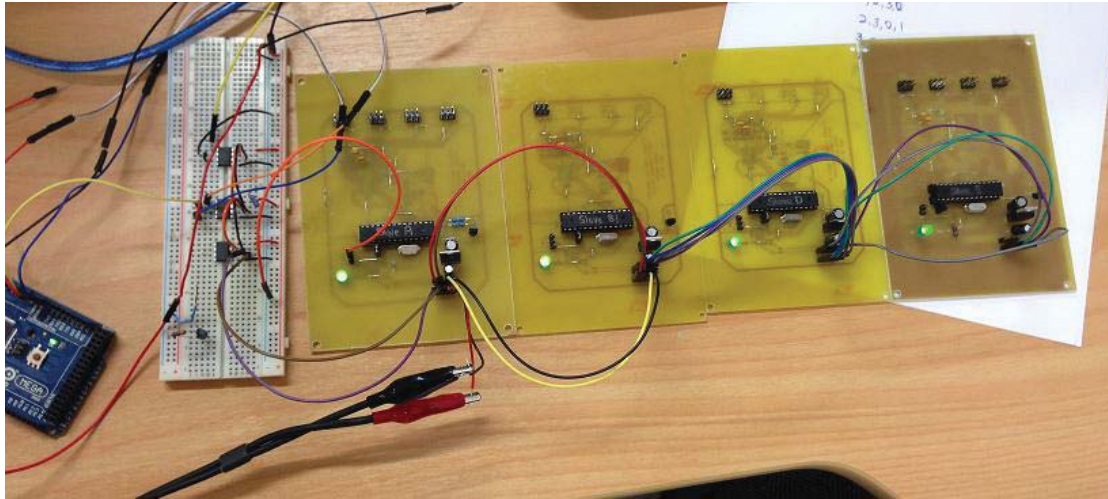


Figure 46: RS-485 Test Setup of Master Communicating with Four Slave Devices

The AD7193 is configured to use the sinc<sup>3</sup> filter therefore the time required to read all enabled channel is given by the following formula [32]:

$$\frac{(3 \times \text{Number of Enabled Channels})}{f_{ADC}} \quad (2)$$

Where:

- $f_{ADC}$  is the frequency which the ADC is set at

Therefore, for example if the ADC frequency was set to 300Hz and four load cells are connected, the effective output of the ADC per channel would only be 25Hz.

The RS-485 communication operates in half-duplex mode (data can't be transmitted and received simultaneously). Therefore, the master has to request data from one slave device at a time as only one device can use the data bus at a time, which can potentially slow down the overall time at which data is received from slave devices. To test the frequency at which data was received at a pin on the Arduino Mega (master) was programmed to toggle each time data was being received. The results are shown in Table 4: Comparison of Calculated and Measured Frequency of Rate at Which Data is Being Received from Slave Devices.

Table 4: Comparison of Calculated and Measured Frequency of Rate at Which Data is Being Received from Slave Devices

AD7193 Data Rate (Hz)	Calculated Frequency (Hz)	Measured Frequency (Hz)
50	4.167	4.717
60	5.000	5.837
150	12.500	6.051
300	25.000	20.470
960	80.000	60.510
2400	200.000	100.300
4800	400.000	130.400

It can be seen from Table 4 that the calculated frequency and measured frequency did not match. This might be due to the RS-485 bus running too slow, although this is unlikely as a baud rate of 115200 was used. It was later discovered that a bug existed within the software that limited the performance of the AD7193. The AD7193 would not continuously read the channels but only once the Master requested data from it, which slowed down performance at higher frequencies.

It should be noted that the measured frequency for 50Hz and 60Hz is faster than the calculated frequency, this might be due human error when using the oscilloscope to measure the values.



#### 4.1.1.2. Simulation of Load Cell Data

A program was developed in the Arduino IDE to simulate the data the master microcontroller would be sending to the computer. This was done so software development and testing could continue without actually having to be connected to the real platform. The software will transmit a slave ID followed by four random values ranging from 20 to 100,000 followed by a CRC (see Figure 48: Simulated Load Cell Data including CRC). To ensure random data is being generated, the analogue input 0 pin is used to seed a random number.

```
// If analog input pin 0 is unconnected, random analog
// noise will cause the call to randomSeed() to generate
// different seed numbers each time the sketch runs.
// randomSeed() will then shuffle the random function.
randomSeed(analogRead(0));
```

Figure 47: Code Snippet to Generate Different Seed Number

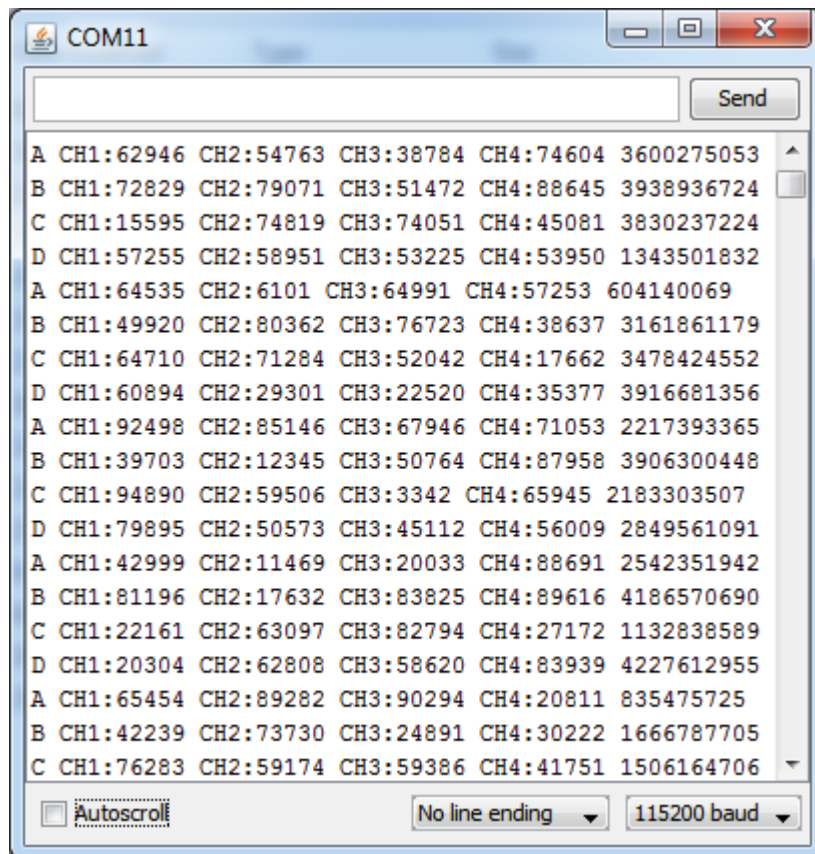


Figure 48: Simulated Load Cell Data including CRC

#### 4.1.2. MATLAB Programming


MATLAB was used to calculate and display the data being received in real-time. Separate m-files were created to plot the weight (kg) vs time (seconds) and the position (mm) in the x-y directions. Both programs had the following in common:

- Open a serial port on a specified port at the baud rate of the master microcontroller.
- Scan the expected string and extract the digitized load cell values.
- Tare each load cell channel.
- Calibrate the load cells (weight is scaled to kilograms)



#### 4.1.2.1. Load Cell Calibration

When purchasing a load cell from the manufacturers, the manufacturer supplies a calibration certificate (see Figure 49: ASB-1000 Load Cell Calibration Certificate) which states the tested strain gauge characteristics. One of the main characteristics on this certificate is the "Full Scale Output" which is used to determine the scaling factor. Each load cell certificate contains the serial number of the load cell which is unique to that load cell; consequently, it is important to not lose these certificates as each load cell's characteristics can vary from load cell to load cell. The serial number is engraved into the load cell. From the 16 load cells purchased, it was found that the full scale output varies from 1.998mV/V to 2.002mV/V. Because of this it was important to calculate the scaling factor for each load cell. See the following example on why this is important.


S/N : 1370133	PT Limited P.O. Box 102041 NSMC Auckland, New Zealand P.O. Box 7205 Baulkham Hills BC, NSW 2153, Australia sales@ptglobal.com	
Model : ASB		
Capacity : 1000 kg		
mV/V : 1.999		
Type : Aluminium Shear Beam		

### CALIBRATION CERTIFICATE

Full Scale Output(mV/V) : 1.999	Recommended Excitation(V) : 5 to 12V
Zero Load Output(%FS) : 0.84	Operating Temperature(C) : -30 to 70
Non Repeatability(%FS) : <0.010	Thermal Zero TC(%FS/C) : <0.0020
Non Linearity (%FS) : <0.025	Thermal Span TC(%FS/C) : <0.0015
Creep(%FS in 30min.) : <0.035	Input Resistance(ohms) : 411.9
Combined Error (%FS):<0.030 (per VDI/VDE 2637)	Output Resistance(ohms) : 352
	Insulation Res.(Mohm @50V) : >5000
	Year of manufacture : Nil
	IP Rating (IEC 529:1977) : 67

#### WIRING CODE

RED	: + Excitation
BLACK	: - Excitation
GREEN	: + Signal
WHITE	: - Signal
BROWN	: +Sense
BLUE	: -Sense



Caution: Altering the length of a 4 core cable will affect calibration.  
Specifications are subject to change without notice.  
[www.ptglobal.com](http://www.ptglobal.com)

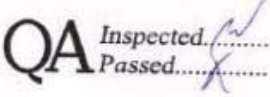


Figure 49: ASB-1000 Load Cell Calibration Certificate

Example: A load cell has a full-scale output of 2.001mV/V and it was calculated that the scaling factor is 0.000467308 (this will be discussed in more depth later). This scaling factor was then used for all the load cells to calculate the true weight. A 50kg weight was placed on each load cell and the ADC value was recorded.

**Table 5: Scaling Factors**

Full Scale Output (mV/V)	Scaling Factor	ADC Reading	Calculated Weight (kg)
1.998	0.000467308	105800	49.441
1.999	0.000467308	106500	49.768
2.000	0.000467308	106850	49.931
2.001	0.000467308	107150	50.072

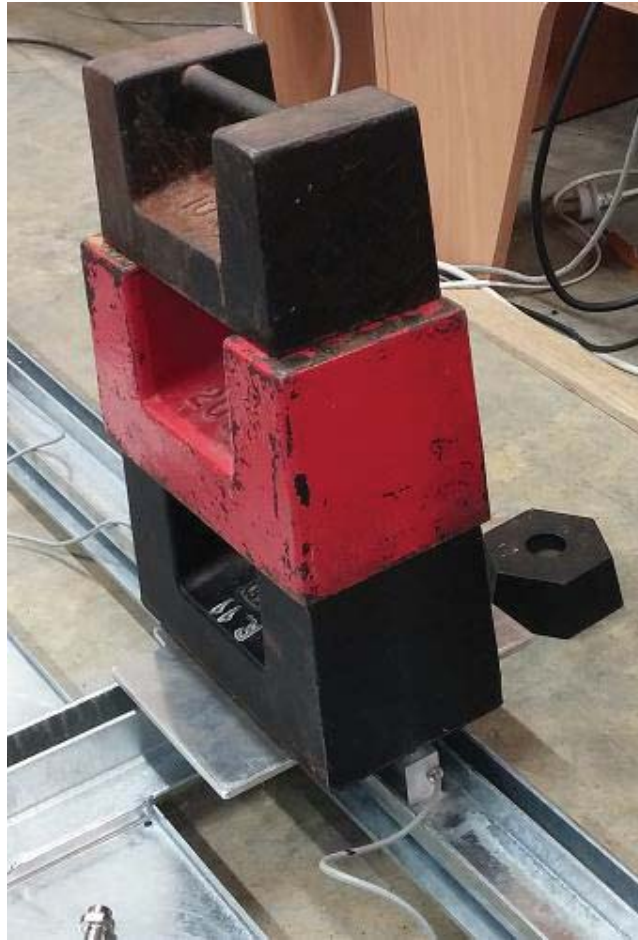
From Table 5: Scaling Factors it can be seen that using the scaling factor from a load cell with a full-scale output of 2.001mV/V on a load cell with a full-scale output of 1.998mV/V produces an error of 631 grams. Now assuming some linear relationship, if a 1000kg was placed on the load cell, the error would be about 12kg.

Consequently, five separate scale factors were calculated to make the system as accurate as possible.

The five scaling factors were determined experimentally making use of a range of known weights varying from 1kg to 50kg. An aluminium plate (150mm x 150mm x 6mm) was made in the workshop which could bolt into the load cell where the load button would normally sit, see Figure 50: Load Cell Support Plate Used During Scaling Factor Experiment. The plate's purpose was to support the calibration weights (see Figure 51: Weight Packed onto Load Cell Support Plate).



**Figure 50: Load Cell Support Plate Used During Scaling Factor Experiment**



**Figure 51: Weight Packed onto Load Cell Support Plate**

MATLAB was used to capture and plot the ADC readings from each load cell when a calibration weight was placed on the load cell. The figure produced by MATLAB was then enlarged and analysed to find the average ADC reading for each calibration weight. The average was used as there is a small percentage of noise present due to the operation of the sigma-delta ADC. The average values obtained from the load cells with various full scale output characteristics at various calibration weights were entered into Excel and the slope of the line was calculated.

An example of the experimental results for load cells with a full scale output of  $2.000\text{mV/V}$  is shown in Figure 52: ADC Reading vs Weight (kg) Producing Linear Slope Table 6. A linear slope is expected (see Section 2.3.3.4) because of the strain gauge characteristics, see Figure 52. The scaling factor is the slope of the graph.

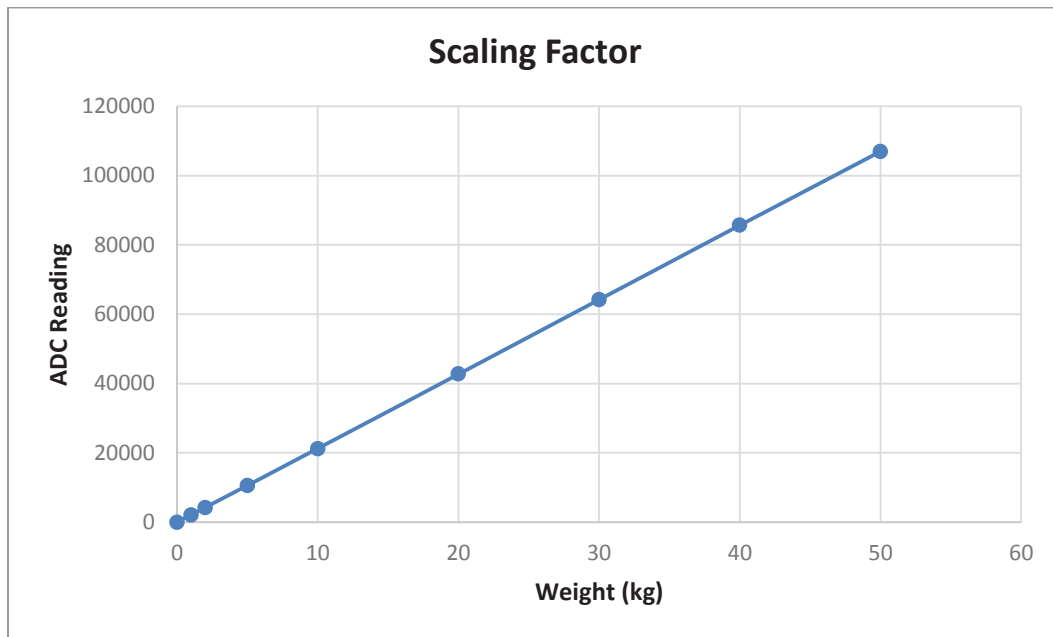


Figure 52: ADC Reading vs Weight (kg) Producing Linear Slope

Table 6: Experimental Calibration Results for Full Scale Output of 2.000mV/V

Full Scale Output of 2.000mV/V	
Calibration Weight (kg)	ADC Reading
0	11
1	2100
2	4200
5	10600
10	21200
20	42750
30	64200
40	85700
50	106850
Scaling Factor	0.00046715

#### 4.1.2.2. Taring the Load Cells of the Platform

It is important to tare the load cells; this means to zero a load cell with a load on top of it. It is especially important to tare the load cells as covers which weigh relatively heavy will be resting on them and needs to be zeroed out. At the start, the taring was done manually by gathering a few values from each load cell. The average was then calculated for each load cell and was used to tare the load cells. An opportunity was spotted to develop an algorithm to automatically tare the sections of the platform.

The input from the master will look like that as seen in Figure 53. The algorithm takes about 400 readings before it has completed its taring. The reason why it takes 400 readings is because there are 4 sections; now this equals to 100 readings per section. The algorithm will scan and look for a certain pattern on the incoming data "%c CH1:%d CH2:%d CH3:%d CH4:%d\n". The algorithm then determines which section the incoming data belongs to and stores the result in their respective arrays. Once the 400 readings are taken, a mean of the array is calculated and the offset value for each specific channel is then removed from the incoming data. This is useful when calculating the weight on the section.



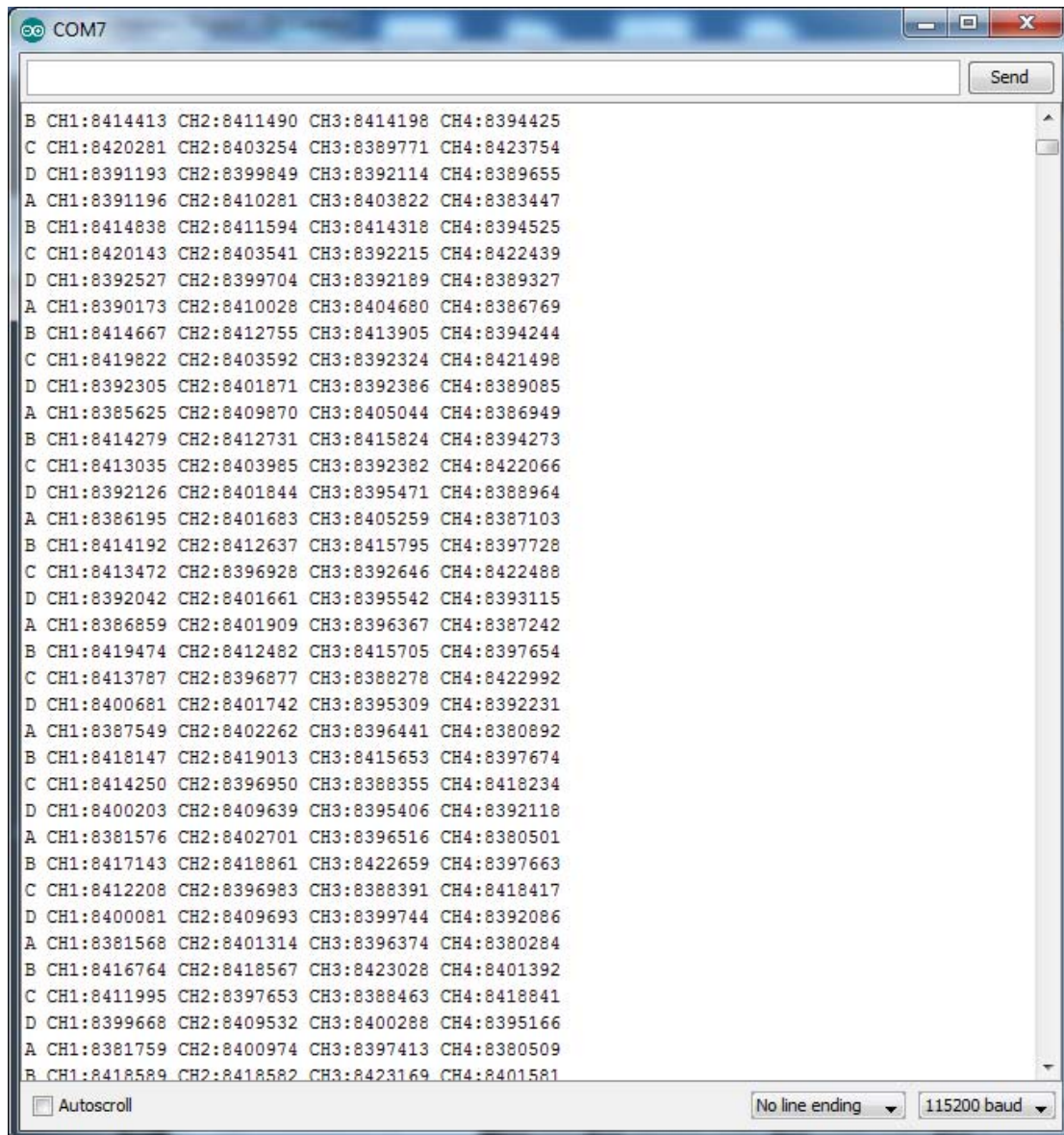


Figure 53: Data Master Transmits to Computer via Serial Port

```
while (x < 400)
    [dataC cntC msgC] = fscanf(serialObjectC, '%c CH1:%d CH2:%d
CH3:%d CH4:%d\n');
    # Check whether data belongs to slave A
    # Store the load cell value in the respective array
    if (dataC(1) == 'A')
        CHA1_MEAN = [CHA1_MEAN, dataC(2)];
        CHA2_MEAN = [CHA2_MEAN, dataC(3)];
        CHA3_MEAN = [CHA3_MEAN, dataC(4)];
        CHA4_MEAN = [CHA4_MEAN, dataC(5)];
    end
    x = x + 1;
end
# Calculate the mean of each load cell value
CHA1_MEAN = mean(CH A1_MEAN);
CHA2_MEAN = mean(CH A2_MEAN);
CHA3_MEAN = mean(CH A3_MEAN);
CHA4_MEAN = mean(CH A4_MEAN);
```

Figure 54: Code Snippet on How to Calculate Weight on Section A

#### 4.1.2.3. Calculating Weight

After the mean of each section and channel is calculated, it is possible to calculate the weight that is currently being experienced on the section. To calculate the weight being experienced on each section, the mean of each channel is deducted from the current channels ADC value and multiplied by the scaling factor, see Equation 3.

$$LC = (ADC_{Value} - LC_{Mean}) \times Scaling\ Factor \quad (3)$$

Where:

<b>LC</b>	is the load cell value scaled in kilograms
<b>ADC<sub>Value</sub></b>	is the raw incoming ADC value for a specific channel
<b>LC<sub>Mean</sub></b>	is the mean value calculated from the raw ADC values under no load condition

Below is a snippet of code of how this was achieved:

```
If (dataC(1) == 'A')
    CH1A(countC) = (dataC(2) - CH1_MEAN) * SCALING_FACTOR_LC1;
    CH2A(countC) = (dataC(3) - CH2_MEAN) * SCALING_FACTOR_LC2;
    CH3A(countC) = (dataC(4) - CH3_MEAN) * SCALING_FACTOR_LC3;
    CH4A(countC) = (dataC(5) - CH4_MEAN) * SCALING_FACTOR_LC4;
```

The resultant force experienced on each load cell in each section will now be scaled to be in kilograms. Four individual plots were created for each section and the force being experienced on each load cell was then plotted. In Figure 55 is an example of one of these plots.

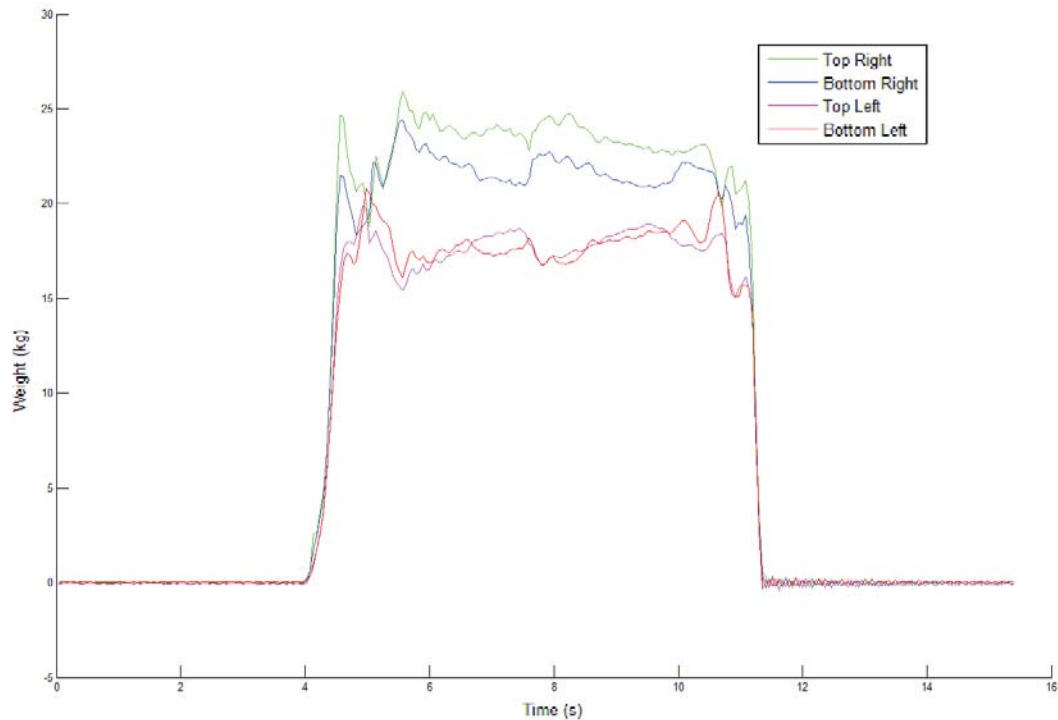
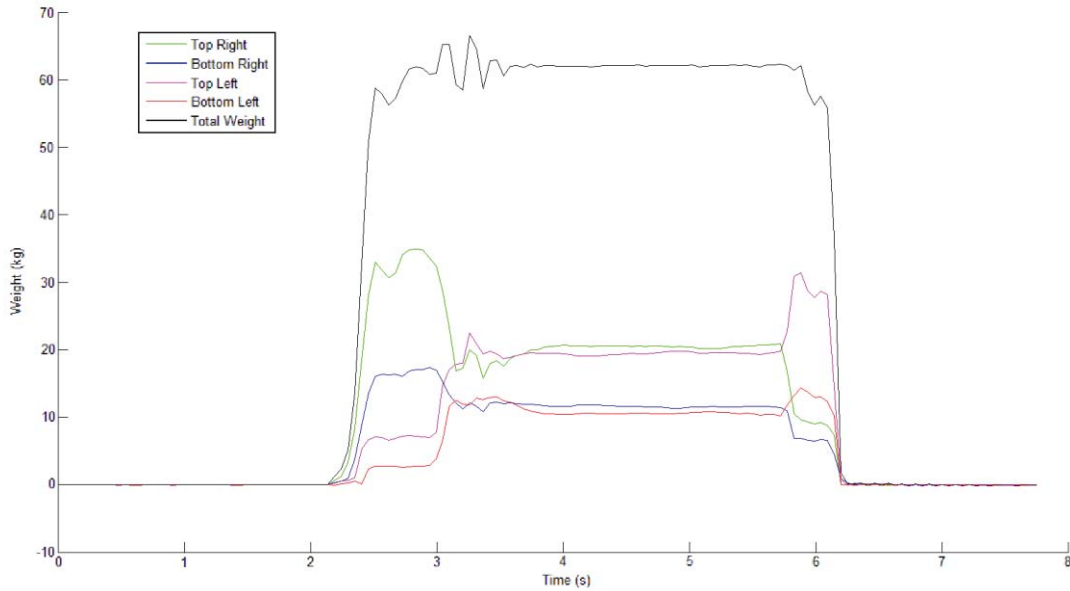


Figure 55: Example of Force Being Experienced On the Four Load Cells by Standing Almost in the Centre of the Section

To calculate the total weight being experienced on a section, the force experienced on each load cell is added together, see Equation 4.

$$Total\ Weight = \sum_{n=1}^4 LC_n \quad (4)$$

An example of someone stepping onto and off a section is shown in Figure 56. It can be seen from the figure that once someone steps on the section, the forces experienced are quite unsteady. A steady weight is achieved once you have settled and stand still. Finally, when you step off, the forces experienced on each load cell are unsteady.



**Figure 56: Stepping onto the Centre Location of a Section**

It should be noted in Figure 55 and Figure 56 that there appears to be a large difference of around 20% to 50% between some of the load cells when standing almost in the centre of the section. This is due to a human standing in the centre and our feet aren't point loads, therefore it is difficult to stand exactly in the centre and we observe these large differences.

#### **4.1.2.4. Calculating Centre of Pressure**

A significant variable to correctly classify lameness is related to the position of the force applied. The position can be used in various ways to determine irregularities in stride length and step width, which are two of the four most highly correlated variables used [9].

An algorithm was developed to determine the centre of pressure; this includes the X and Y position using the four load cells. When the load cells are placed under pressure, reaction forces are generated. These reaction forces are F1, F2, F3 and F4 (as seen in Figure 57). The distance between F1 & F2 and F3 & F4 is the Width, and the distance between F1 & F3 and F2 & F4 is the Length. The total force being experienced on the platform is  $F_{Total} = F1 + F2 + F3 + F4$ .

The origin has been defined to be at the lower left corner of the section. When calculating the X-position (see Equation 5) F2 and F4 are summed together and divided by the total force ( $F_{Total}$ ) as the force is shared by all the load cells. This results in a value less than one, and is then multiplied by the width (distance between F1 and F2) to find the X-position.

To calculate the Y-position (see Equation 6), F3 and F4 are summed together and divided by the total force, and then multiplied by the Length (distance between F2 and F4).

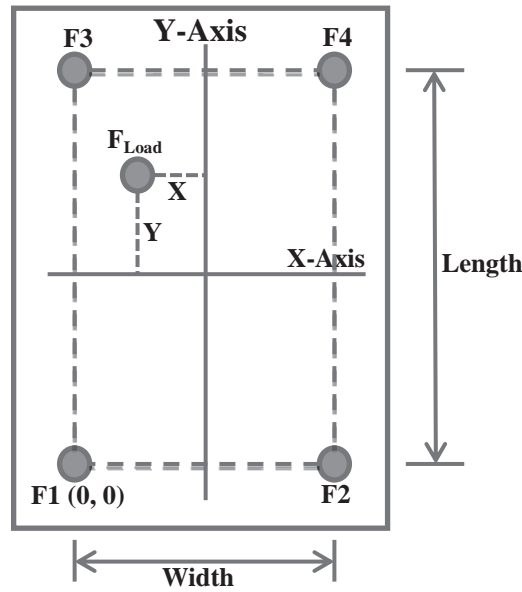


Figure 57: Calculating Centre of Pressure

$$X = \frac{(F2 + F4)}{F_{Total}} \times Width \quad (5)$$

$$Y = \frac{(F3 + F4)}{F_{Total}} \times Length \quad (6)$$

A snippet of code on how to calculate the centre of pressure is shown below. The value of 425mm was the width between the load cells, and 600mm was the length between the load cells in the section. The testing of the algorithm and the accuracy of the co-ordinates can be found in Section 5.2.1.

```
x(countC) = ((CH2(countC)+CH4(countC))/sum(countC))*425;
y(countC) = ((CH3(countC)+CH4(countC))/sum(countC))*600;

scatter(x,y);
```

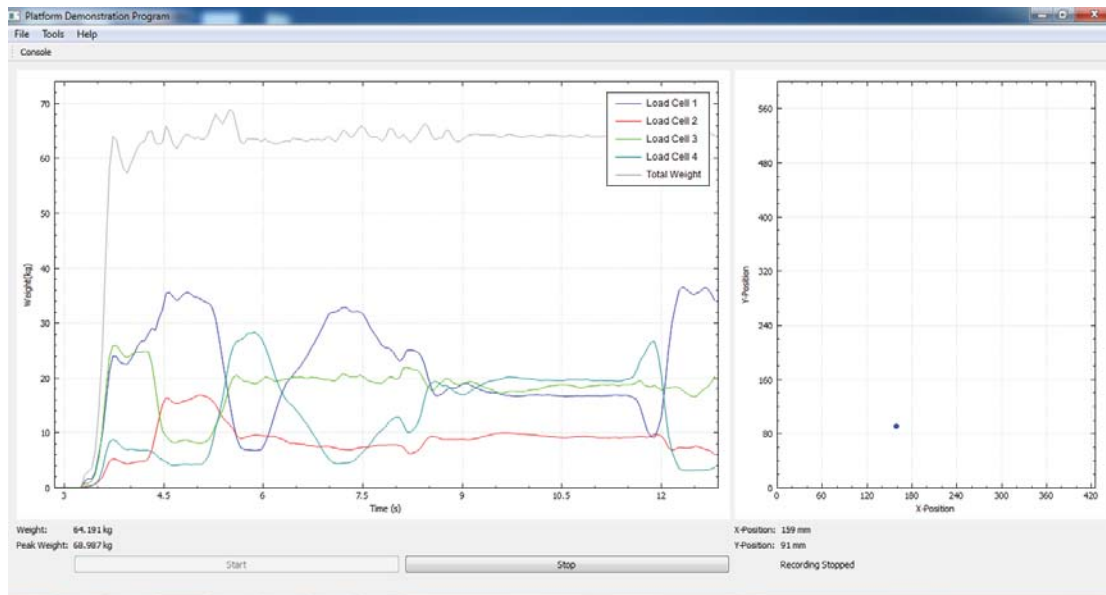
#### 4.1.3. C++ Programming

Now that it was proven that the total weight and centre of pressure could successfully be calculated in MATLAB, the same functionality was implemented in C++. All the C++ programs were developed in Qt Creator. It was decided to write dedicated C++ programs as MATLAB requires special licensing fees.

##### 4.1.3.1. GUI Program for Prototype Demonstration Platform

A C++ program with a GUI was developed to demonstrate the prototype demonstration platform (see Section 5.2). The algorithms used in this application are identical to that explained in the MATLAB programming section, the only difference being it was now being implemented in C++ (see Figure 58 for screenshot of the application).





**Figure 58: Screenshot of the Demonstration Program GUI**

When the user presses the start button, the program automatically detects the COM port to which the microcontroller is connected to. Once the serial port is successfully opened, the load cells are tared. Only once all the load cells are tared, will the program start plotting. This includes:

- A plot of weight vs time of all four load cell signals including the total weight in real-time.
- A plot showing the centre of pressure on the platform in real-time.
- Various labels which show the current weight on the platform, the X and Y position on the platform, what the peak weight was, and whether it is currently recording data to a file or not.

When the program is set to record data, it simply records the incoming data from the microcontroller.

Another useful feature added to this application was the ability to open a serial console to view the incoming data from the microcontroller. This was especially handy for fault finding. An error message is printed to the console window when the program can't successfully open the COM port. An example of the serial window can be seen in Figure 59 displaying the ADC values received from the microcontroller.

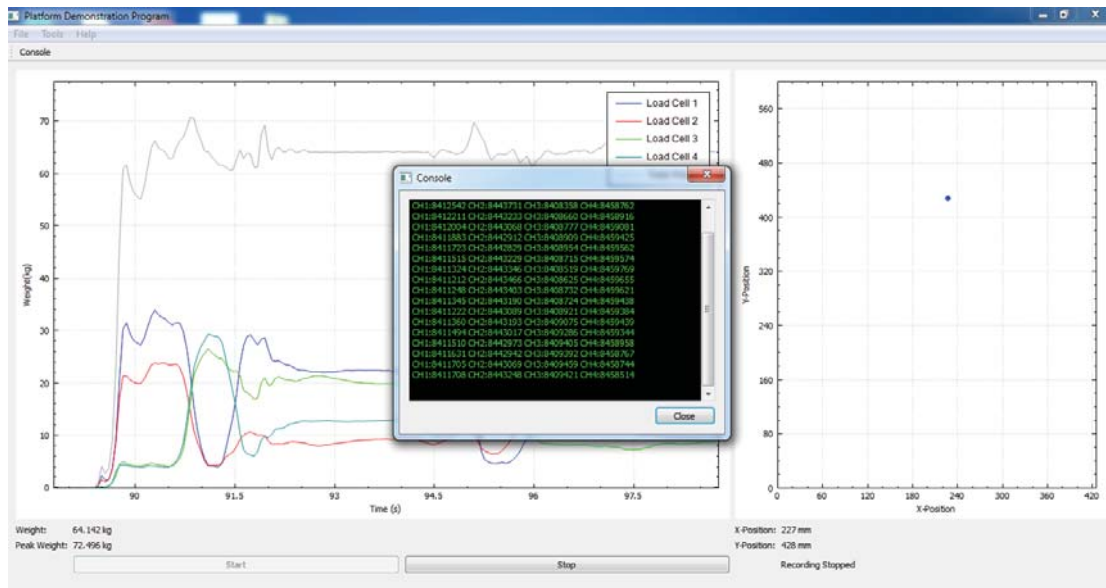


Figure 59: Demonstration Program Serial Console Window Displaying Data Received from Microcontroller

Note that only a single Arduino Uno and the prototype breakout board are used when running this program.

#### 4.1.3.2. Real-Time Plot of Individual Sections of the Platform

A program to view individual sections (display individual load cell signals, total weight and centre of pressure) in real-time was developed. This was especially useful to observe and test whether each section of the platform was responding and functioning correctly. The program itself consisted of multiple tabs, allowing the user to view one section at a time. A tab allowing the user to view all four sections at once was implemented, but was removed due to the graphs being cramped as seen in Figure 60. Once the program starts, it will automatically scan all the COM ports and look to which the Arduino Mega (Master) is connected to. If it is found, it will calibrate the platform and start plotting the incoming data. If it can't connect to the Arduino Mega, an error message is printed to the console window.

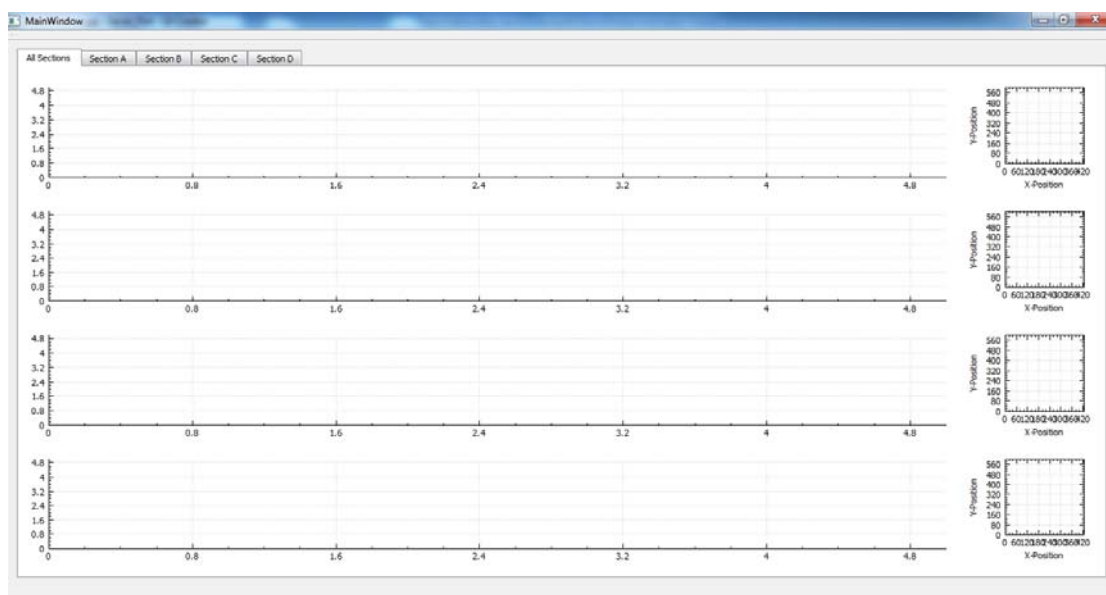


Figure 60: Example of Program Displaying All Sections and Multiple Tabs to Select Each Section Individually

As there are four sections in the platform, meaning a total of 16 load cells (4 load cells per section), one can imagine there would be lots of variables involved. The power of C++'s

object-orientated features were taken to create a class that simplified the code. It was decided to develop a class giving it various properties and methods. Therefore, when I need to add a new section, I simply have to create a new instance of the class instead of having to create lots of new variables associated with the section when adding a section.

Various variables were used in the class, including the number of readings to take when taring the platform, variables to hold the mean value of each channel, variables to hold the raw channel information being received from the load cells, a variable to hold the total weight, and finally, variables to hold x and y position of the centre of pressure.

Various functions were implemented such as setting and getting each channel individually, the ability to calibrate (scale in kilograms) each channel individually and getting each channel's calibrated value individually, set the ID of the section, calibrate the section, calculate the weight, calculate the x-position, and calculate the y-position. It is also possible to calculate the x-position or y-position with a custom width or length.

```
int platform::getChannel1()
{
    return channel1;
}

void platform::setChannel1(int ch1)
{
    channel1 = ch1;
}
```

```
void platform::setIdentity(QString id)
{
    identity = id;
}

QString platform::getIdentity()
{
    return identity;
}
```

```
// This function calibrates the platform, appends 50 readings to a
// list for each channel
// it then calculates the mean for each channel
//void platform::calibratePlatform(int ch1, int ch2, int ch3, int
//ch4, bool reset = false)
void platform::calibratePlatform()
{
    // We ignore the first ten readings from the microcontroller
    if (counter > 10 && counter < (NO_READINGS + 10))
    {
        ch1_list.append(channel1);
        //qDebug() << ch1;
        ch2_list.append(channel2);
        ch3_list.append(channel3);
        ch4_list.append(channel4);
        counter++;
    }
    else if (counter <= 10)
        counter++;

    // qDebug() << counter;

    if (counter == (NO_READINGS + 10))
    {
```

```

        ch1_mean = calculateSum(ch1_list)/ch1_list.size();
        ch2_mean = calculateSum(ch2_list)/ch2_list.size();
        ch3_mean = calculateSum(ch3_list)/ch3_list.size();
        ch4_mean = calculateSum(ch4_list)/ch4_list.size();
        counter++; // Increment counter again so this part doesn't
get executed everytime
    }
}

```

```

double platform::calculateWeight(int ch1, int ch2, int ch3, int ch4)
{
    if (ch1_mean != 0 && ch2_mean != 0 && ch3_mean != 0 && ch4_mean
!= 0)
    {
        // Get the weight on each load cell and scale it
        ch1_cal = (ch1 - ch1_mean) * 0.0004673;
        ch2_cal = (ch2 - ch2_mean) * 0.0004661;
        ch3_cal = (ch3 - ch3_mean) * 0.0004693;
        ch4_cal = (ch4 - ch4_mean) * 0.0004673;

        return ch1_cal + ch2_cal + ch3_cal + ch4_cal;
    }
    return 1;
}

```

```

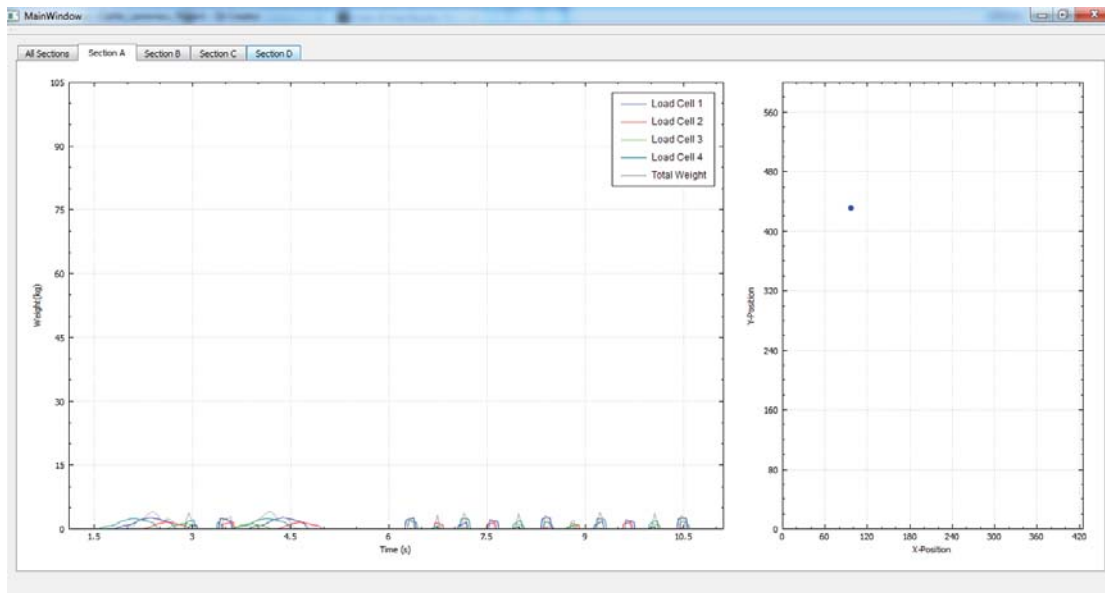
double platform::calculateXPosition()
{
    if (ch1_cal != 0 && ch2_cal != 0)
        x_position = ((ch1_cal + ch2_cal)/weight)*425;

    return x_position;
}
double platform::calculateXPosition(int width)
{
    if (ch1_cal != 0 && ch2_cal != 0)
        x_position = ((ch1_cal + ch2_cal)/weight)*width;

    return x_position;
}

```

An instance was created for each section all the variables were updated accordingly and then plotted as seen in Figure 61.

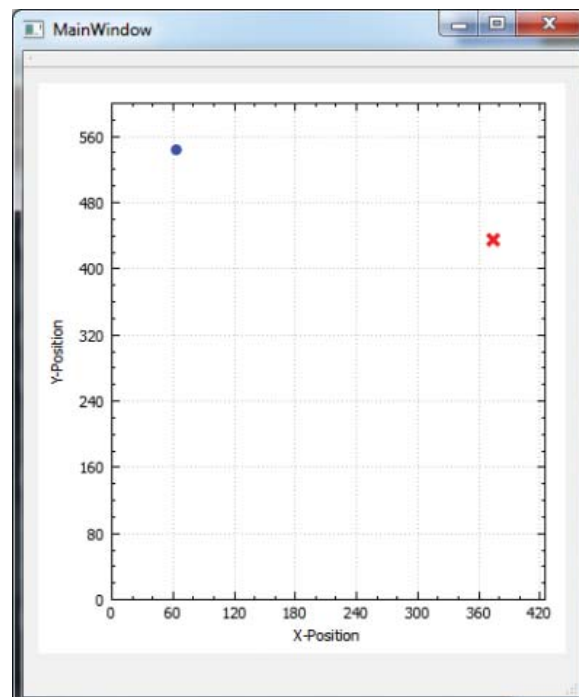


**Figure 61: Data Plotted for Section A in Real-Time**

In Figure 61 it can be seen that noise is present on Section A; this was due to using a switched-mode power supply to power the platform. This caused false readings to occur even when no weight was being applied, hence a positional dot is displayed in the upper left corner of the positional data figure. All sections were experiencing this issue. This issue was solved by using a linear regulated power supply instead to power the platform.

#### **4.1.3.3. Game Exploration**

A simple game was written to explore using the demonstration platform (see Section 5.2) as a gaming device. This gives the user some form of exercise while playing a game, also known as 'exergaming'. A red cross will appear at a random location on a scatterplot. The user is the blue dot, and the objective of the game is to reach the red cross. Once the red cross is reached, it will disappear and a new cross will appear at a random location. This results in the user having to physically move around on the platform to reach the target, giving them some exercise.



**Figure 62: Simple UI Displaying the User (Blue Dot) and Their Target Location (Red Cross)**

It was found that there is potential to use the demonstration platform as a gaming platform. Future work would be to include a scoring system and different game modes.

#### 4.1.3.4. *Animate Recorded Data*

A program was developed which allowed the user to select data that was recorded (see Section 4.1.4.1 on how data was recorded for the platform) and play it back to get an idea of how the cow walked over the platform. The user is able interact with the plot such as dragging the plot along its x-y axis and zooming in and out. A label is used to display to the user what file they are currently viewing.

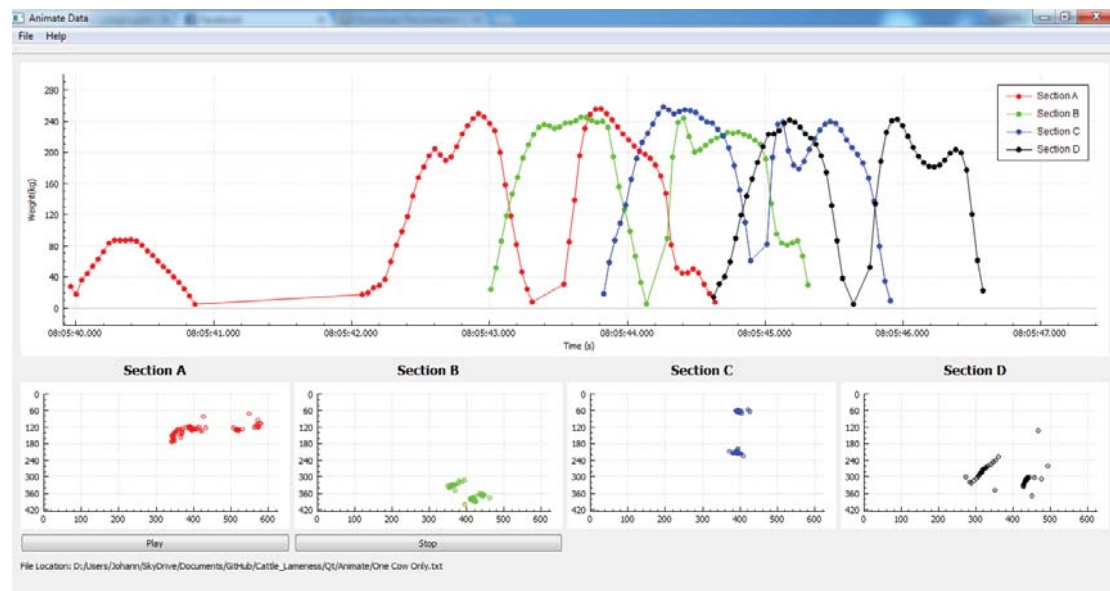


Figure 63: UI of Animate Data Program

As seen from the user interface, a label is used display to the user what file is currently selected

#### 4.1.4. Python Programming

A GUI looking program was implemented in Python, allowing the user to record data, analyse the data, or make use of various tools such as rendering data to a video file, opening a serial monitor, and viewing which COM ports are available.

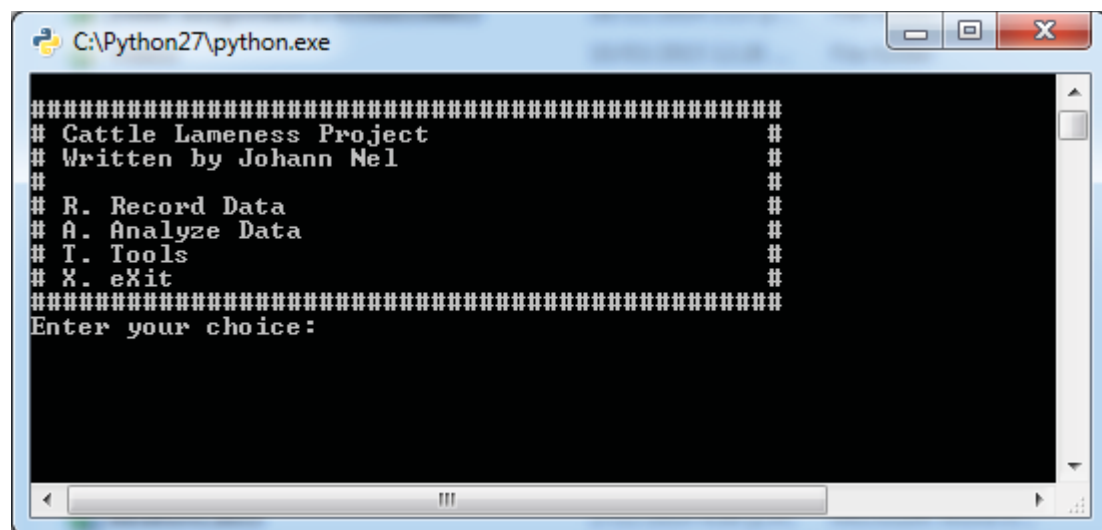


Figure 64: Console Interface of Python Program

#### 4.1.4.1. Record Data

When the user enters the choice 'R', the user is able to record data to a text file. The user is able to select between recording without the EID reader or with the EID Reader. The data is stored in a folder called "Data".

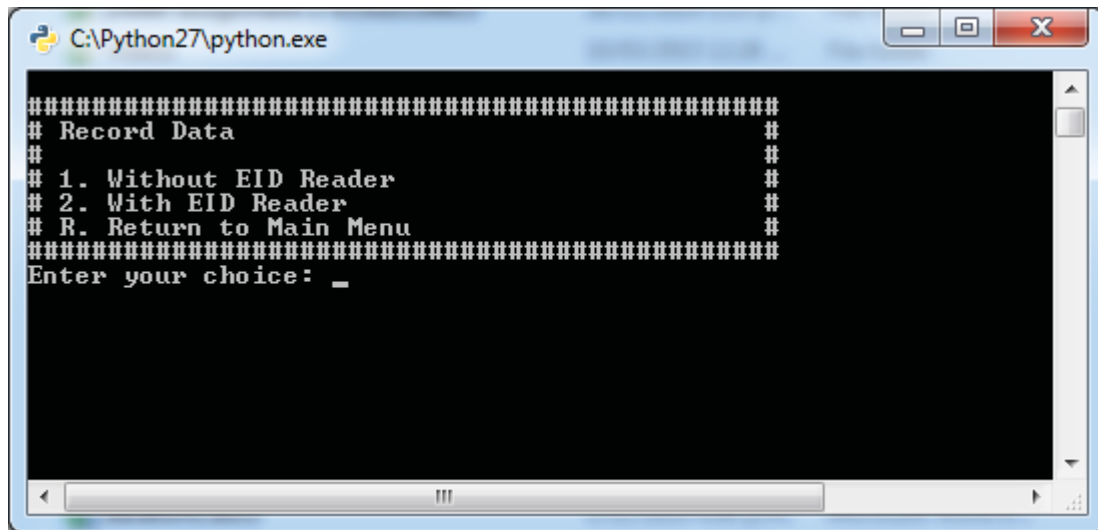


Figure 65: Record Menu GUI

The data recorded to the text-file will be in the following format and is comma delimited:

Time, Slave, CH1, CH2, CH3, CH4, Total Weight, X-Position, Y-Position, Peak

Where:

- Time is the time the data is written to the file
- Slave is the ID of the section
- CH1 – 4 is the force (calibrated in kilogram) experienced on each load cell
- Total Weight is the total weight experienced on the section
- X-Position/Y-Position is where the centre of pressure is on the section
- Peak will indicate whether a new peak weight has occurred or not. A 'P' is written to indicate a new peak occurred, otherwise an 'o' is written to the file.

Given below is an example of how the data is stored in the text-file.

```
TIME,SLAVE,CH1,CH2,CH3,CH4,WEIGHT,X,Y,PEAK
10:32:06.692000,A,2.48,3.78,1.79,0.02,8.08,339.39,303.67,P
10:32:06.723000,A,3.99,9.62,4.08,1.00,18.71,318.80,247.95,P
10:32:06.770000,A,11.42,20.10,4.72,3.26,39.52,349.44,234.47,P
10:32:06.801000,A,17.91,30.00,5.99,6.55,60.47,347.13,226.93,P
10:32:06.848000,A,20.49,33.66,6.44,7.21,67.81,349.76,228.03,P
10:32:06.879000,A,19.50,32.91,6.41,7.27,66.10,347.31,225.06,o
10:32:06.926000,A,18.55,31.30,6.38,7.04,63.28,345.09,226.18,o
10:32:06.973000,A,18.07,30.39,6.39,7.08,61.94,342.73,226.71,o
10:32:07.004000,A,16.87,29.13,5.65,5.97,57.63,349.59,224.39,o
```

Examining the first line from the example it can be seen that data was captured around 10:32:06.692000, the data was coming from section A, the total weight was 8.08kg, the centre of pressure was at 339mm (x) and 303mm (y) and was the peak weight indicated by the 'P'.

When a file is created, the program will automatically name it to the date the program is executed i.e. "11-03-2015.txt". When data is written to a file, the program will always write the header "TIME,SLAVE,CH1,CH2,CH3,CH4,WEIGHT,X,Y,PEAK" followed by the data. When data is recorded multiple times on the same day, the program will simply append to the file as it is important to not overwrite any pre-existing data (it will first write the header again and



then the actual data). This makes it easier to spot when data has been recorded multiple times to the same file. Data will only be recorded if more than 5kg of force is experienced on any section; this is to ensure that if the cows were to leave any faeces behind, that it won't get recorded. This threshold can be changed if required. Data will also only be recorded if the received CRC matches the calculated CRC.

```

=====
Received Data: A CH1:8386207 CH2:8403741 CH3:8398665 CH4:8384308 2196105900
Received CRC: 2196105900
Calculated CRC: 2196105900
Correct CRC
=====
Received Data: B CH1:8416521 CH2:8414655 CH3:8417632 CH4:8397608 1925092709
Received CRC: 1925092709
Calculated CRC: 1925092709
Correct CRC
=====
Received Data: C CH1:8415104 CH2:8398822 CH3:8390117 CH4:8421490 2216405813
Received CRC: 2216405813
Calculated CRC: 2216405813
Correct CRC
=====
Received Data: D CH1:8394437 CH2:8402462 CH3:8396080 CH4:8392364 2396108788
Received CRC: 2396108788
Calculated CRC: 2396108788
Correct CRC
=====

```

Figure 66: Proof that Received CRC and Calculated CRC are Identical

The precision of each load cell, total weight, and centre of pressure is up to two decimal points. When the user is done recording data, it is important to hit “Enter” as this will close the serial port and the file that is being written to.

At this point of time, it was impossible to distinguish between different cows walking over the platform, therefore an EID reader was implemented to read the ear tags from the cows as they walk over the platform. The EID reader is placed at the end of section D (see Figure 67).

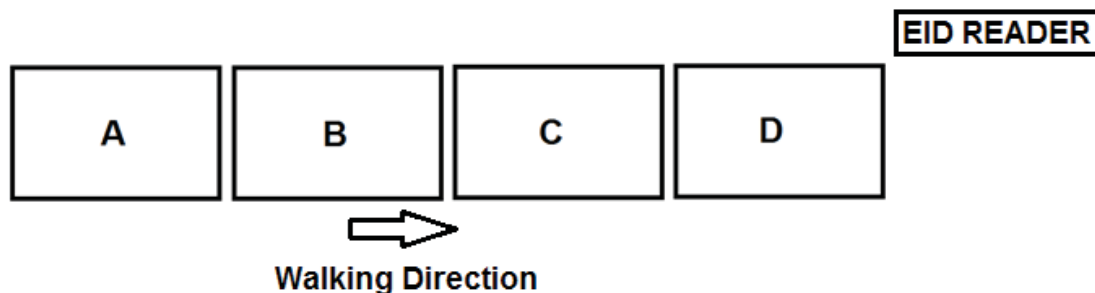


Figure 67: Placement of EID Reader

An XRP2 EID Reader (see Figure 68) and large RF antenna was supplied by TruTest, allowing us to read the ear-tags from cows as they walk over the system. The EID reader was connected serially (RS-232) to the computer, and a serial monitor was opened to determine the data the EID reader transmits over the serial port when an ear-tag is passed by the antenna (see Figure 69). It was determined that the EID reader uses a baud-rate of 9600, transmits 8 data bits, no parity and 1 stop bit. The serial output was in the format of a 3-digit manufacturer's code followed by a unique 12 digit number which corresponds to a specific ear-tag.



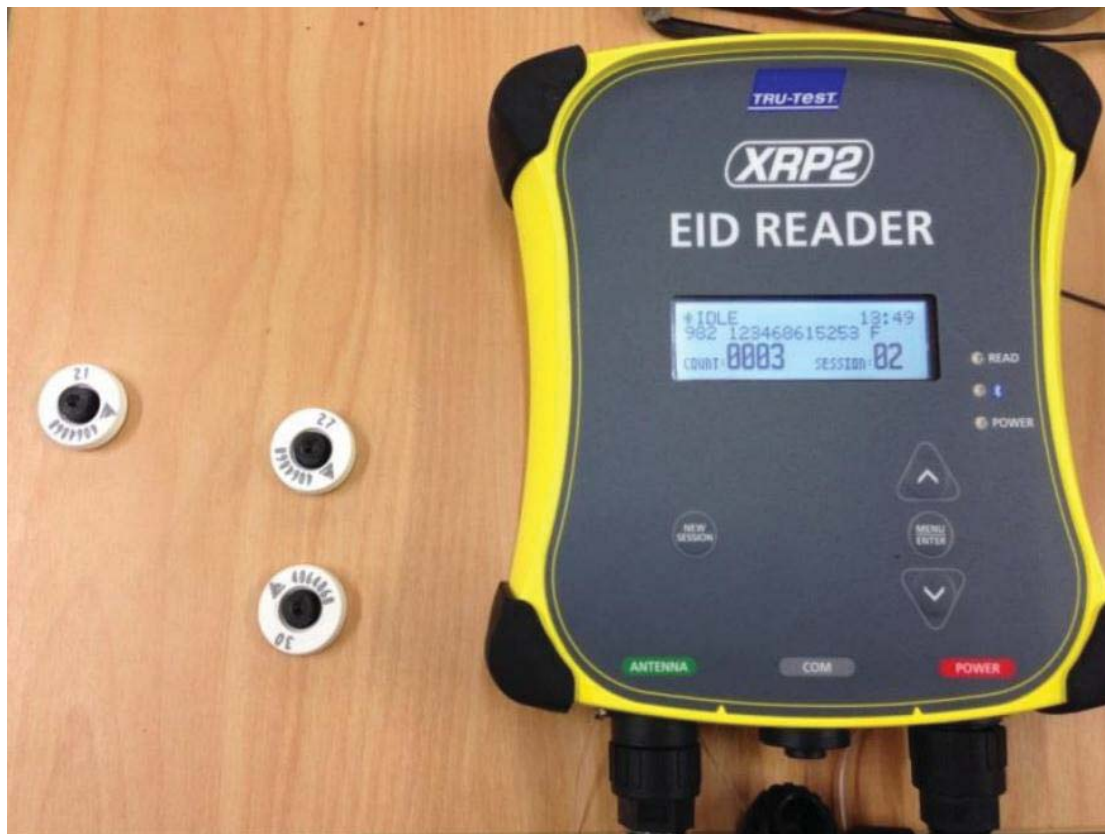


Figure 68: EID Reader Used to Detect and Display Ear-tag Numbers

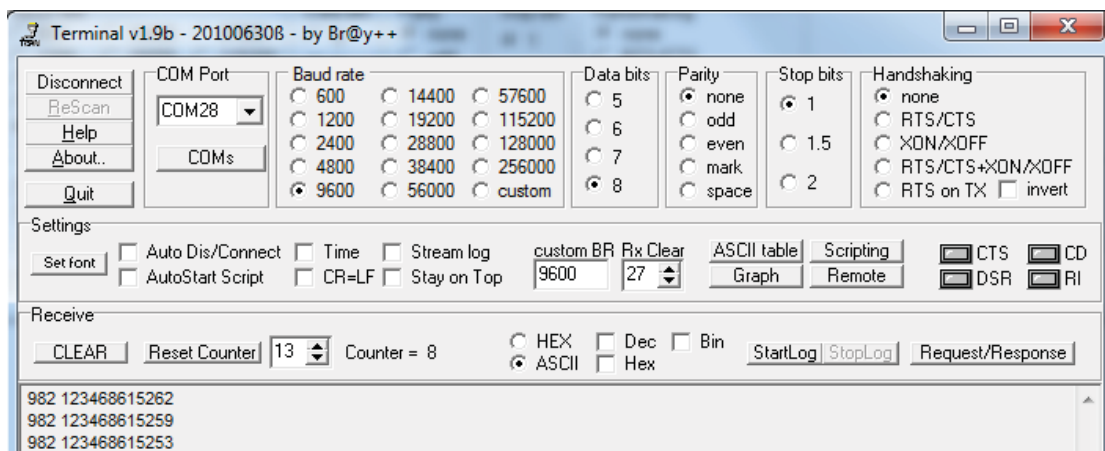


Figure 69: Serial Data Received from EID Reader

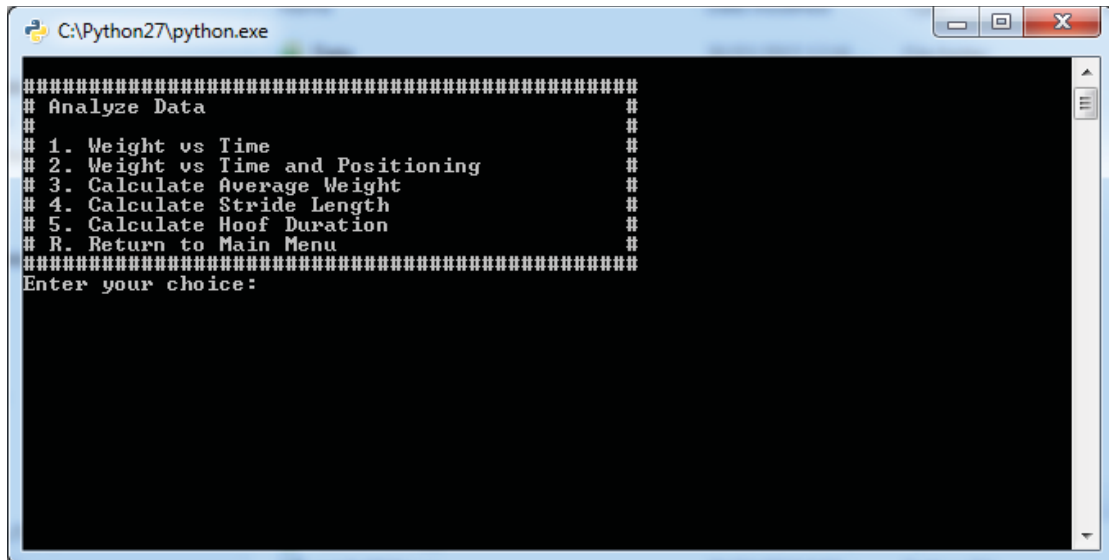
Now when a cow walks over the platform, the data will be recorded as described earlier. The only difference now is that once the cow walks past the EID reader, an ID is written to determine which cow the data belongs to.

```
11:11:12.516000,D,30.14,49.45,4.58,8.77,92.95,375.06,214.45,P
11:11:12.547000,D,31.52,50.92,4.84,8.85,96.15,375.58,217.07,P
11:11:12.578000,D,29.45,46.56,4.34,8.06,88.42,376.53,219.37,o
ID: 982 123468615262
11:11:12.625000,D,26.30,40.78,3.64,7.06,77.79,377.72,220.95,o
11:11:12.656000,D,22.78,35.72,2.83,5.76,67.11,381.86,219.12,o
11:11:12.703000,D,20.37,31.89,2.23,5.04,59.54,384.45,217.97,o
```

Figure 70: Code snippet of Data to Expect in File "11-03-2015.txt"

#### 4.1.4.2. Analyse Data

When the user enters the choice 'A', the user is able to analyse data files that have been recorded. The user can plot the data, weight vs time or weight vs time and position; or the user can calculate the average weight, the stride length or the duration each hoof was on a section.



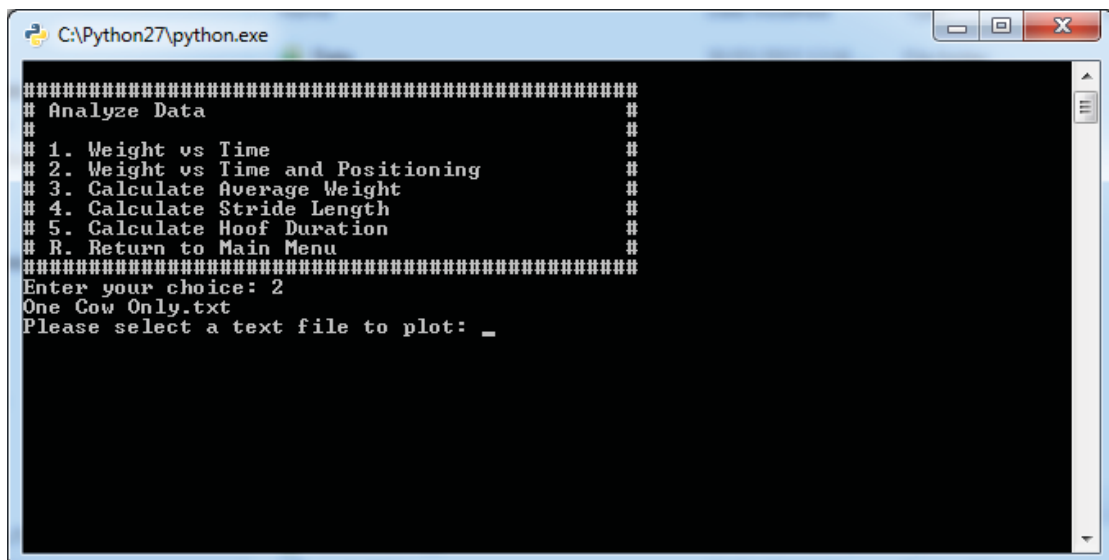
```

C:\Python27\python.exe
#####
# Analyze Data                                     #
#                                                  #
# 1. Weight vs Time                               #
# 2. Weight vs Time and Positioning                #
# 3. Calculate Average Weight                     #
# 4. Calculate Stride Length                       #
# 5. Calculate Hoof Duration                       #
# R. Return to Main Menu                           #
#####
Enter your choice:

```

Figure 71: Analyze Data Menu

When any of the options are selected, the program will print all the available text files found in the "Data" folder and ask the user which file to use.



```

C:\Python27\python.exe
#####
# Analyze Data                                     #
#                                                  #
# 1. Weight vs Time                               #
# 2. Weight vs Time and Positioning                #
# 3. Calculate Average Weight                     #
# 4. Calculate Stride Length                       #
# 5. Calculate Hoof Duration                       #
# R. Return to Main Menu                           #
#####
Enter your choice: 2
One Cow Only.txt
Please select a text file to plot: _

```

Figure 72: Program Prompting the User to Select a File to Plot

##### 4.1.4.2.1. Algorithms Used to Analyse Data

When analysing the data, various algorithms had to be developed. These algorithms include being able to split the peaks and remove any invalid peaks when processing the data.

Split Peaks:

As seen in Figure 73 when the cow walked over the platform, various peaks occurred. The first peak that occurs on each section is the front left/right hoof and the second peak that occurs on each section is the back left/right hoof of the cow. Therefore, it is important to be

able to split the peaks. Note that the first peak that occurred at the beginning was because the cow was hesitant to walk over the platform as it was unfamiliar to the cow.

Two methods were developed to split the peaks; the first method was to make use of the x-axis i.e. time. By observing Figure 73 it can be seen that when the cow is placing pressure on the section, lots of data points are sampled for a brief period before the second peak occurs. It can be seen that there is a big time gap between “Peak 1” and “Peak 2”, it can also be seen that there is a gap between “Peak 2” and “Peak 3”. Because of the fast sampling rate of the system, this observation was used to my advantage to split the peaks.

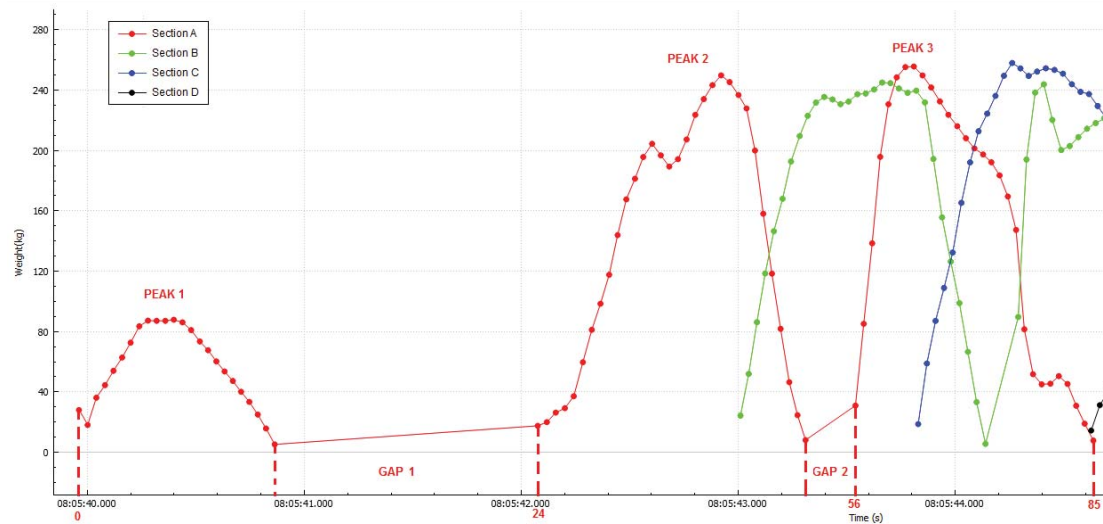


Figure 73: Example of How to Split Peaks of Section A Based on Time

In this example, we will explore how the peaks that occurred on section A were split. By observing Figure 73, it can be seen that there are two gaps that occur on Section A annotated as GAP 1 and GAP 2. The array containing all the data of section A is passed to a function that will find the index of where each peak starts. The algorithm simply works by looking at the previous time reading value and the current time reading value. A counter is used to keep track of where we are in the array. If it is found that the time value between the current and the previous reading is greater than a certain time threshold, the counter value gets stored in an array. The previous reading and current reading values are then updated. This continues until the end of the array is reached. A flowchart of how this algorithm works can be found in Figure 75.

By manually counting the points, it was found that the expected value returned from the function should be [0, 24, 56, 85]. The algorithm developed returned the expected value, see Figure 74.

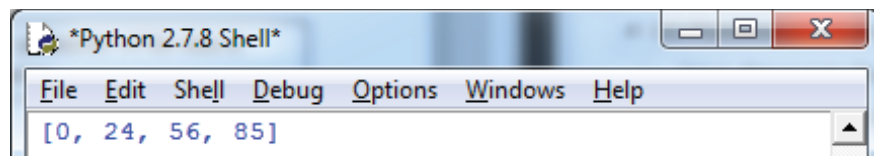


Figure 74: Output from Running Split Peaks Algorithm

The array will always begin with 0 and end with the length of the array, in this case 85. The number 24 indicates where peak 1 starts, and the number 24 indicates where peak 2 starts (as seen in Figure 74). Now to extract peak 1, take the second index [0, 24, 56, 85] and subtract one. Now this means that all the data between index 0 and 23 belongs to peak 1. To extract peak 2 simply subtract one from that start value of peak 3 i.e. [0, 24, 56, 85], this means all the data between index 24 and 55 belong to peak 2. And finally all the data between index 56 and 85 belong to peak 3.

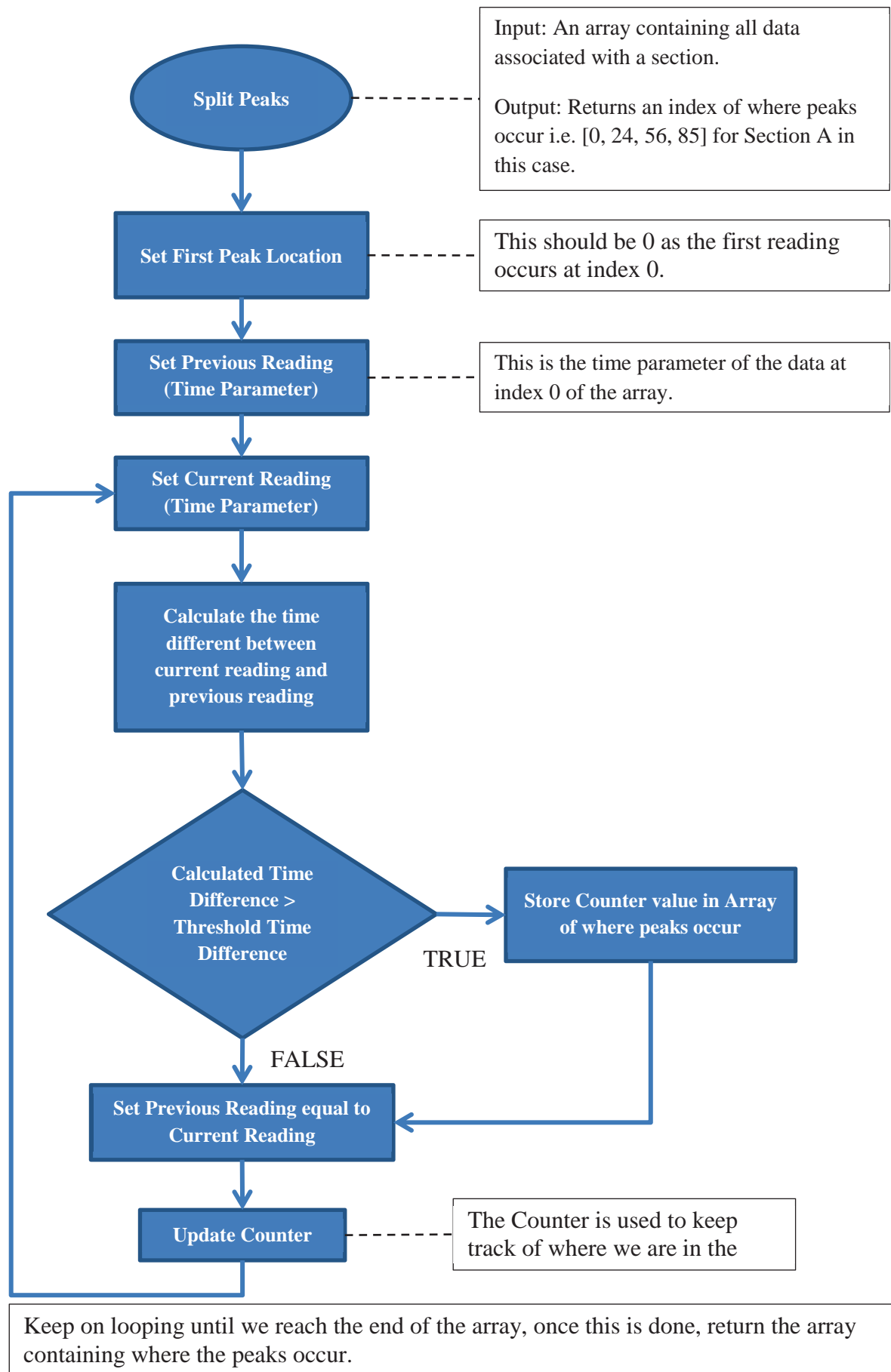
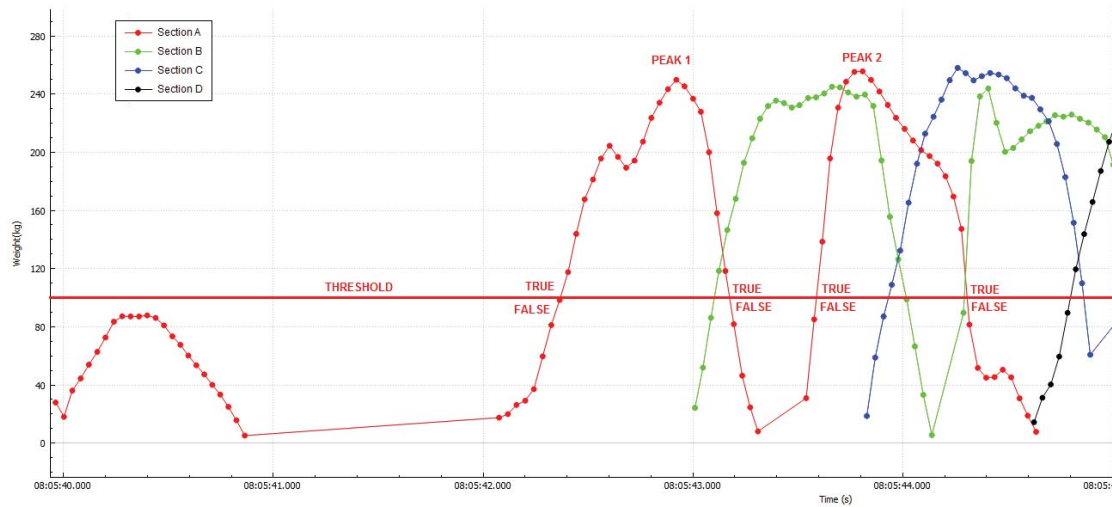


Figure 75: Flowchart of Splitting Peaks based on Time

The second method developed to split the peaks make use of the y-axis i.e. weight of the peaks that occur.



**Figure 76: Example of How to Split Peaks of Section A Based on Weight**

This is achieved by setting a high threshold and considering any weight above this threshold as true, and any weight below this threshold as false and will be ignored. By observing Figure 76: Example of How to Split Peaks of Section A Based on Weight, it can be seen that something unique is occurring; at the start of a peak, the transition from false to true always occurs, and at the end of the peak, the transition from true to false occurs. This observation was used to create the algorithm to split the peaks. Two Boolean values are used: one to hold what the Boolean value was in the past, and one to hold the current Boolean value. If the previous Boolean value was true and the current Boolean value is false, that means the end of the peak has occurred.

A flowchart of how this algorithm works can be found in Figure 77.

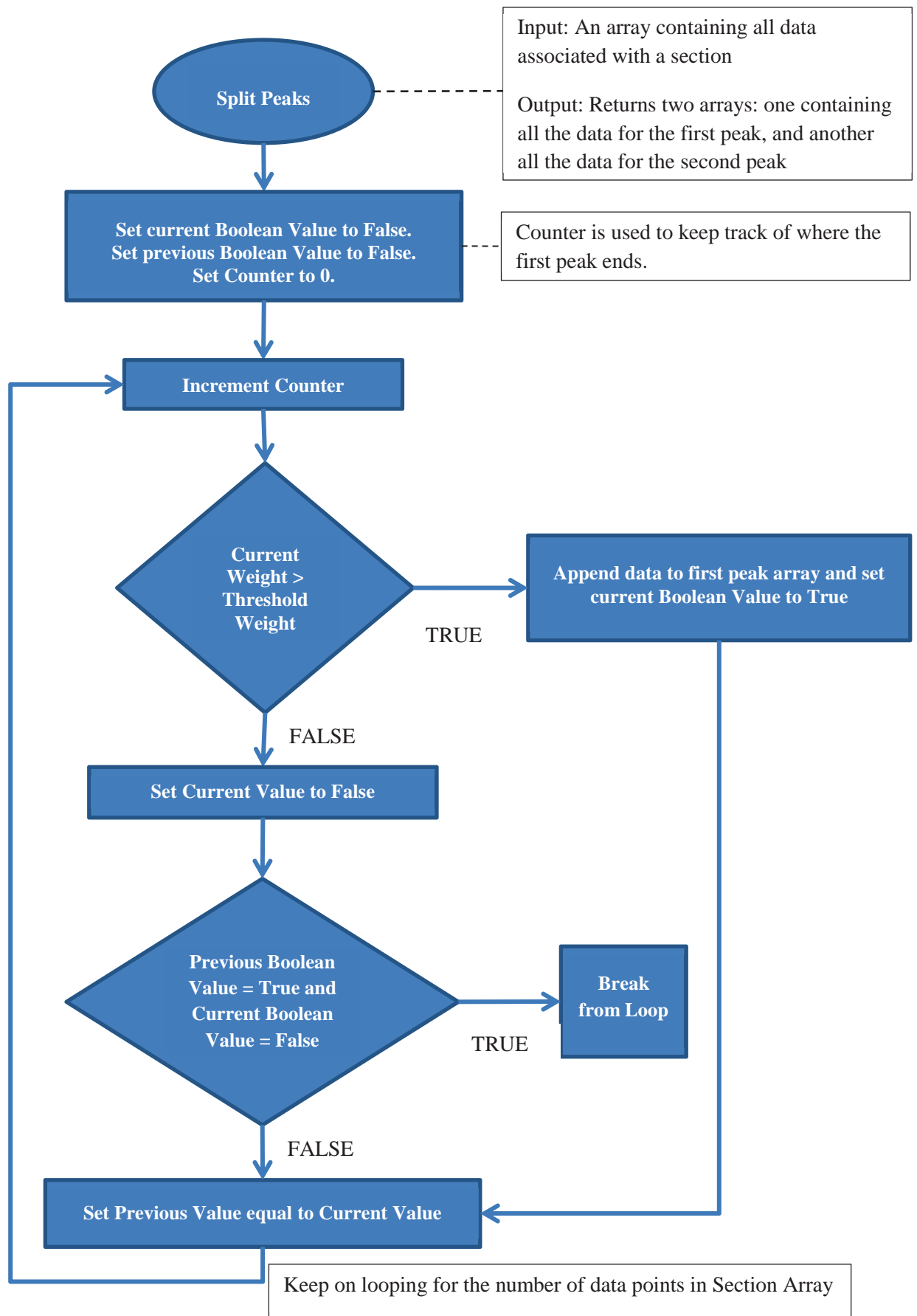
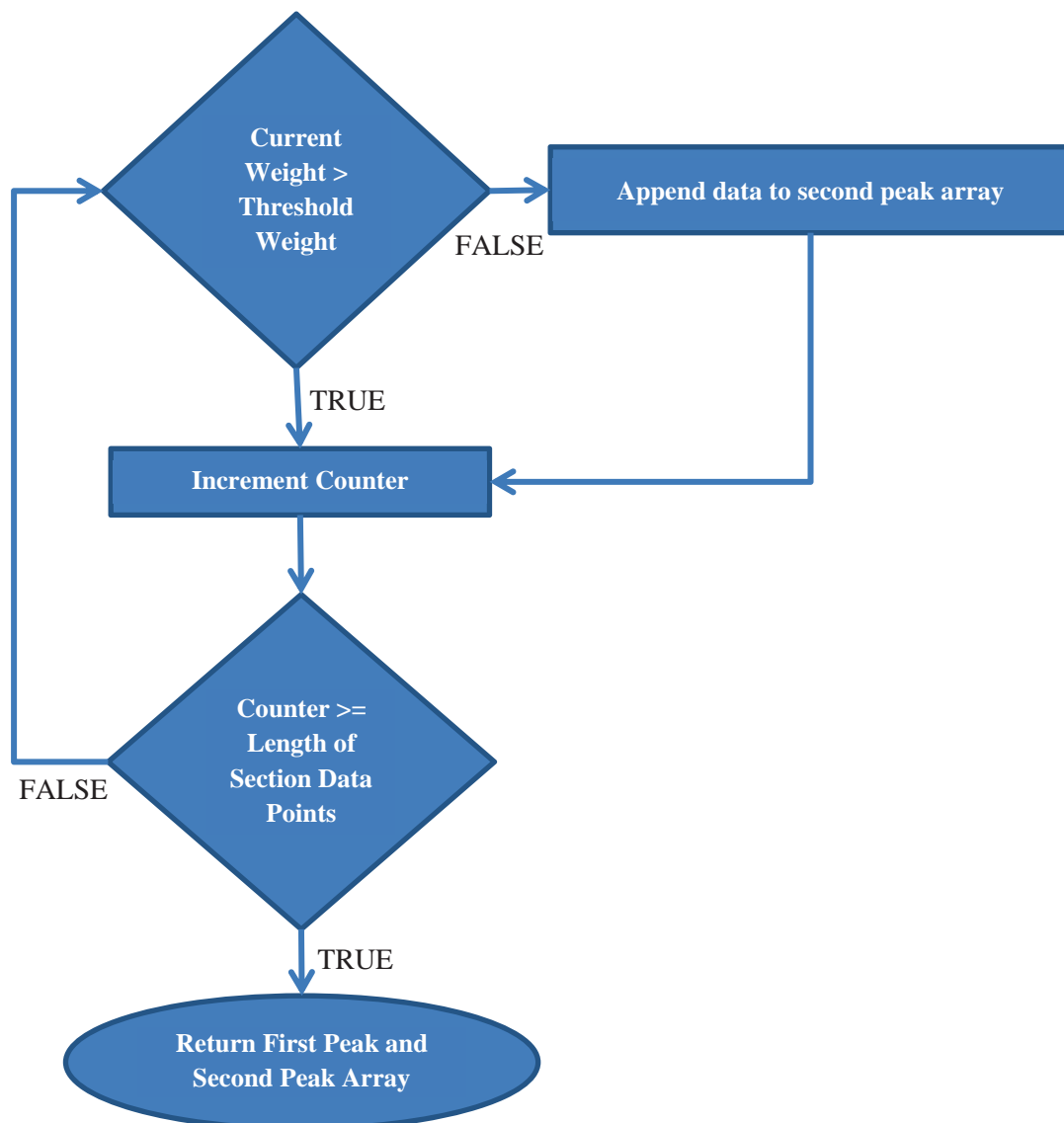


Figure 77: Flowchart of Extracting First Peak Based on Weight

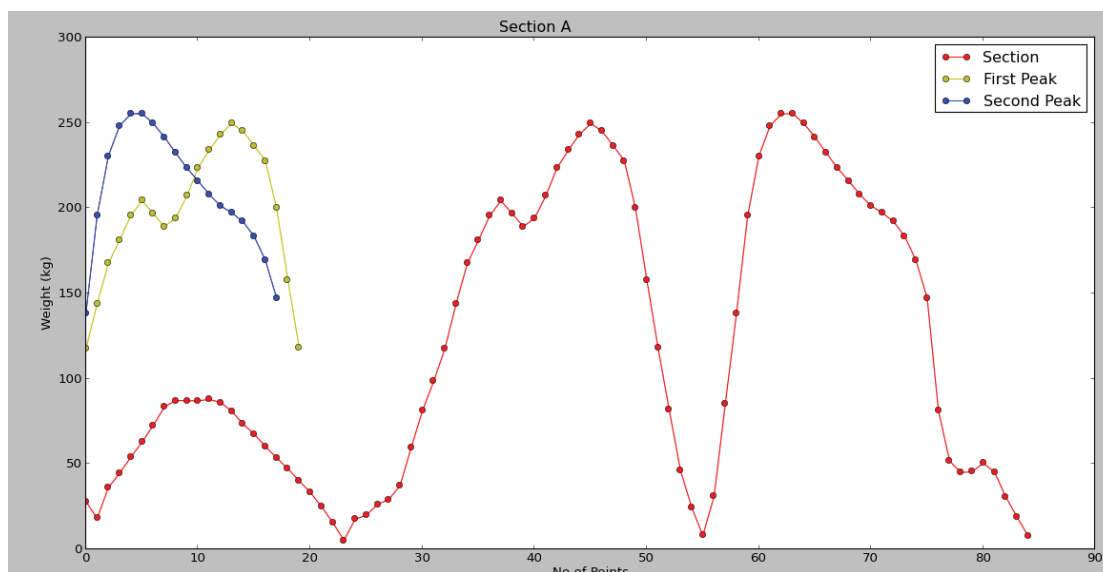
The first peak is now extracted. The only part left now to do is to iterate over the remaining data and extract the second peak, see Figure 78. The loop will continue from the value currently stored in counter.



**Figure 78: Flowchart of Extracting Second Peak Based on Weight**

To prove that the algorithm is working as intended, the original data was plotted along with the extract first and second peak data.





**Figure 79: Comparing Extracted Peaks to That of Original Signal**

From Figure 79: Comparing Extracted Peaks to That of Original Signal it can be seen that the first peak (yellow) has the same shape as that of the first peak of the original signal (red), and the same with the second peak (blue). A threshold value of 100kg was used, filtering out all readings below 100kg.

Being able to extract the peaks will definitely be helpful when calculating the stride length of the cow, the average weight of the cow, and the time the cow spends on each hoof while walking over the platform. This information can prove to be helpful in determining whether a cow is becoming lame or not.

### Removing Invalid Peaks:

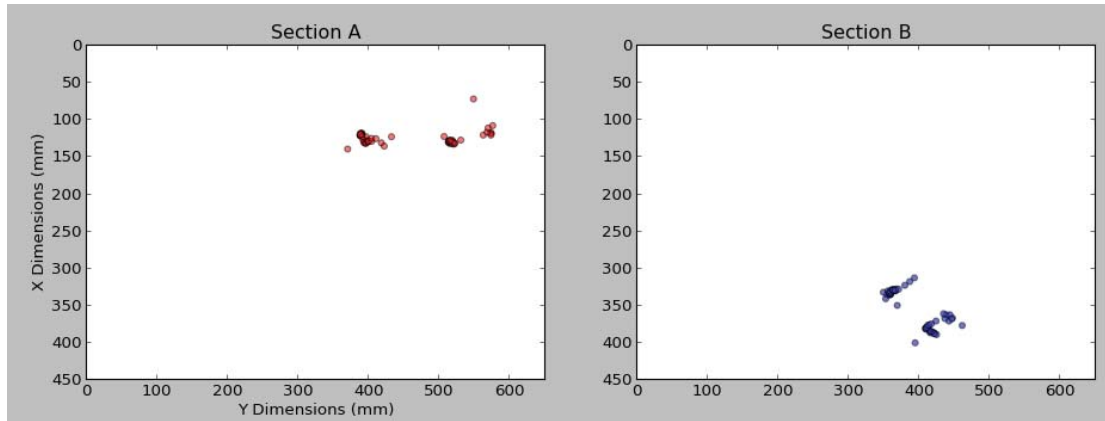
As seen in Figure 73: Example of How to Split Peaks of Section A Based on Time, when the cow walked over the platform, at the start it was very hesitant to walk over it as it was unfamiliar it. This resulted in a small peak occurring at the start when the cow walked over. This created an opportunity to develop an algorithm to remove any invalid peaks from the data.

Now that it is possible to find the where peaks start (method 1 of find peaks), an algorithm was developed to verify whether a peak is valid or not. The points of where the peaks occur and the section data itself is passed to a function that validate whether the peaks are valid or not. So for instance in the example of Section A, the peaks occur at [0, 24, 56, 85], as seen in method 1 of find peaks algorithm.

A threshold weight value of say 100kg is set, and the algorithm then simply looks at all the points between 0 and 23 (remember 24 is where the second peak starts). If it is found that the peak weight between those values is less than the threshold value, the start of the peak is removed from the peaks array, or else it is kept. Then it will look at value between 24 and 55 to find whether the peak weight is greater than the threshold weight. In this instance it is found to be greater and is kept in the array of where the peaks occur. The algorithm then looks at all the points between 56 and 85 and evaluates it is a valid peak as well. The result would be [24, 56, 85].

### Detection of Left or Right Hoof:

As the positional data is also recorded when the weight is recorded, it is possible to determine whether it was a left or right hoof on the section. Because it is possible to split the peaks, the average X-position of each peak can be calculated. The first peak that occurs on a section is always the front hoof.



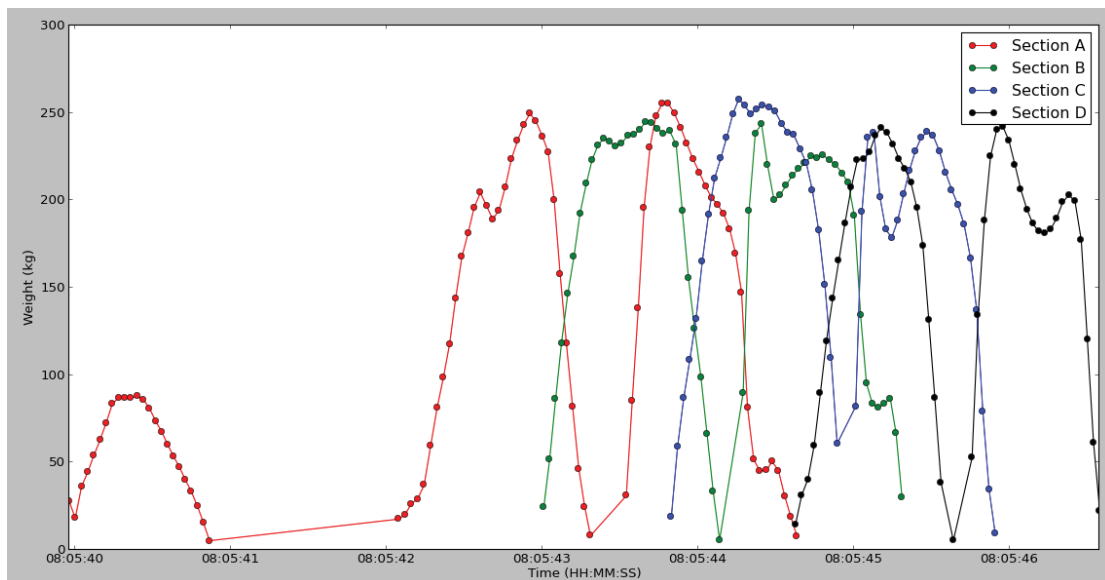
**Figure 80: Positional Data of Section A and Section B**

To determine whether it is the left or right front hoof, the average X-position data of section A and the average X-position data of section B is taken.

If it is found that the average X-position data of Section A is less than the average X-position data of Section B, it can be concluded that the left front hoof was on Section A and the right front hoof on Section B; else the right front hoof was on Section A and the left hoof on Section B. This calculation is only done on Section A and Section B, as the same walking pattern will occur on Section C and Section D (the cow can't change its leg orientation once its stepped onto the platform). Only the front hoof X-positional data is used to determine left or right hoof as the back hoofs will land around the same position as the front hoofs.

#### 4.1.4.2.2. Weight vs Time

When the user selects option 1 "Weight vs Time", the program will prompt the user which file to plot. The file's contents will then be read and plotted as a function of weight vs time as seen in Figure 81: Example of Plotted Cow after Walking Over the Platform. The data will be plotted as it is in the file, and no invalid peaks will be removed.



**Figure 81: Example of Plotted Cow after Walking Over the Platform**

The weight displayed for each section is the total weight experienced on that section. The program reads the text-file and determines whether the data belongs to section A, B, C, D. The program then simply extracts the time and weight for the section the data is associated with and plots it (see Figure 81: Example of Plotted Cow after Walking Over the Platform).

#### 4.1.4.2.3. Weight vs Time and Positioning

When the user selects option 2 “Weight vs Time and Positioning”, the program will plot the data from the text-file as a function of weight vs time and plot where on the platform the centre of pressure occurred. This can be seen in Figure 82. The data will be plotted as it is in the file, and no invalid peaks will be removed.

The program reads the text-file and determines whether the data belongs to section A, B, C, or D. The program then simply extracts the time, weight, x and y positions for the section the data is associated with, and plots it.

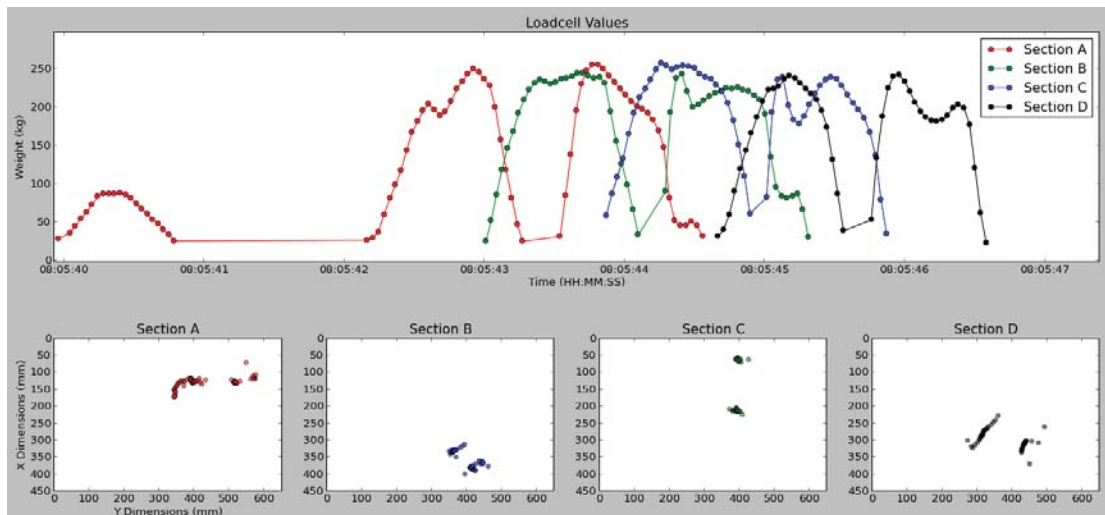


Figure 82: Cow Data Plotted as Weight vs Time and Centre of Pressure on each Section

#### 4.1.4.2.4. Calculate Stride Length

When the user selects option 3 “Calculate Stride Length”, the user is presented with three options. These three options are based on various techniques on how to calculate the stride length.

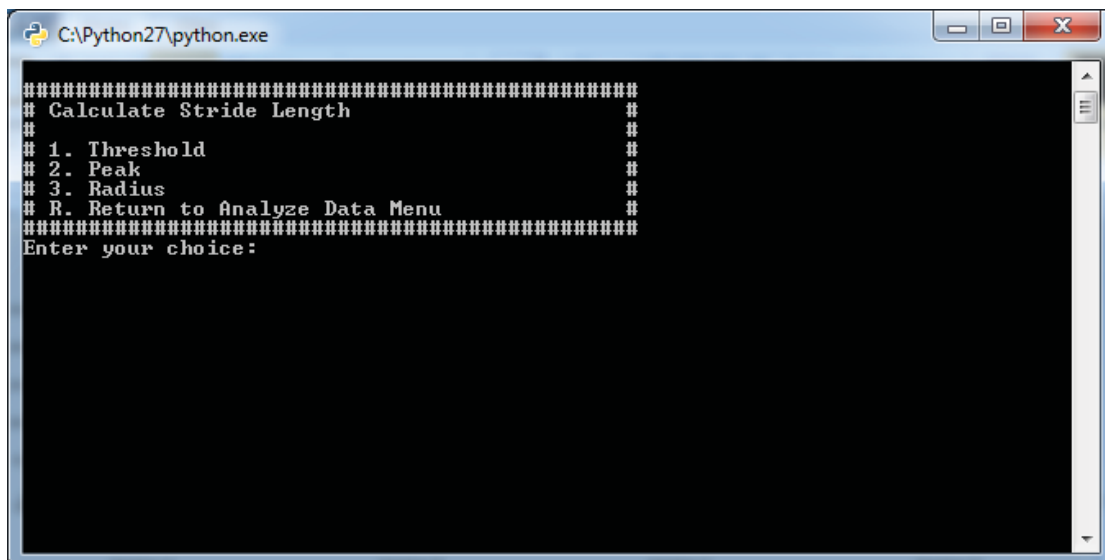


Figure 83: Calculate Stride Length Menu

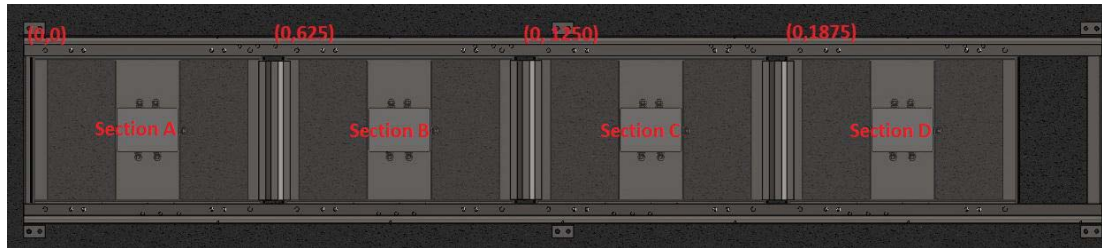


Figure 84: Platform showing Locations of First Load Cells of Each Section

To calculate the stride length:

- Loadcell 1 of section A was considered to be (0,0)
- Loadcell 1 of Section B was considered to be (0, 625)
- Loadcell 1 of Section C was considered to be (0, 1250)
- Loadcell 1 of Section D was considered to be (0, 1875)

To calculate the stride length for the front left hoof for instance, the location for section C (left front hoof) will be deducted from the location of section A (left front hoof). The software can automatically detect whether it is a left/right front hoof or left/right back hoof. The first peak that occurs on any of the sections will always be the front hooves of the cow and the second peak will be the back hooves of the cow. The location of where the hoof is on the platform would be where the centre of pressure was the greatest; hence it is important to filter out the data that occurred at low weights.

**Please note that the stride length is in millimetres (mm).**

Three methods to calculate stride length were developed:

1. The first method to calculate the stride length is to set a high threshold to narrow down where the cow stood on the platform. It can clearly be seen in Figure 85 that a lot of the outliers have been filtered out compared to Figure 82. An average of the remaining location data was then taken to calculate the stride length.

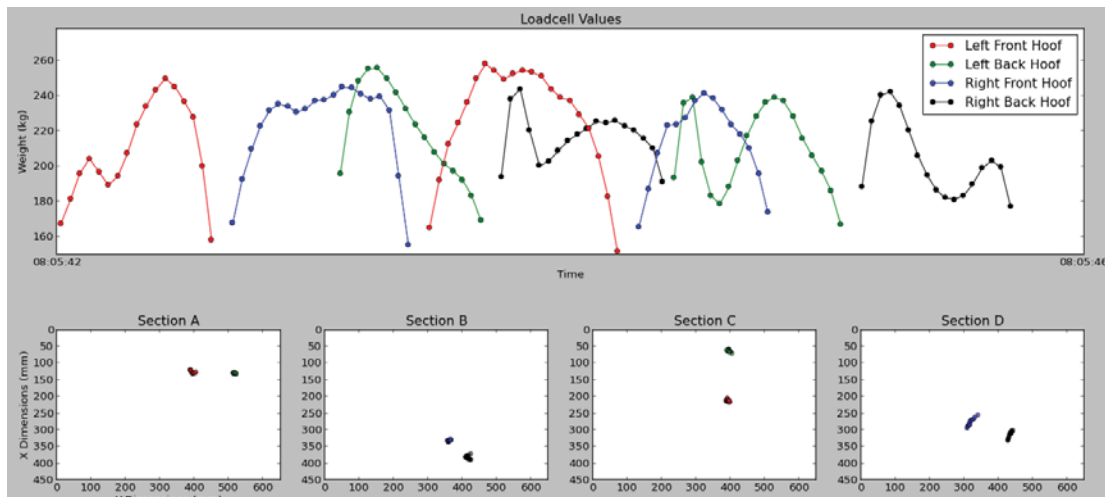


Figure 85: Result from using a Threshold of 150kg

```

Stride A->B Front Hoof: 592.913239559
Stride B->C Front Hoof: 656.144796071
Stride C->D Front Hoof: 552.512213429
Stride A->B Back Hoof: 571.682507648
Stride B->C Back Hoof: 615.795846974
Stride C->D Back Hoof: 667.83666375
Stride A->C Front Hoof (left side of cow): 1249.05803563
Stride A->C Back Hoof (left side of cow): 1187.47835462
Stride B->D Front Hoof (right side of cow): 1208.6570095
Stride B->D Back Hoof (right side of cow): 1283.63251072

```

Figure 86: Stride Length Result from using a Threshold of 150kg

Two types of strides were calculated: the first being the stride between the left and right hooves, and the stride of each individual hoof.

From the results obtained (see Figure 86: Stride Length Result from using a Threshold of 150kg), it can be seen that stride of the front hooves, left front hoof (occurred on section A, red colour) and right front hoof (occurred on section B, blue colour) was found to be 592mm. The stride of the back hooves, left back hoof (occurred on section A, green colour) and right back hoof (occurred on section B, black colour) was found to be 571mm.

Now the stride of left front hoof (red colour), from Section A to Section C was found to be 1249mm.

Caution needs to be taken when setting the threshold value. If the threshold was for instance set to 200kg, unexpected results would occur. This is because the algorithm works by splitting the peaks by weight as discussed in Section 4.1.4.2.1.

Observing Figure 87, it can be seen with the first peak that a dip occurs as the cow stepped onto the section. Because of this dip, the algorithm will split this signal into two peaks and treat it as the two peaks that occurred on Section A; therefore, the result would look like that as seen in Figure 88 and present the user unexpected results.

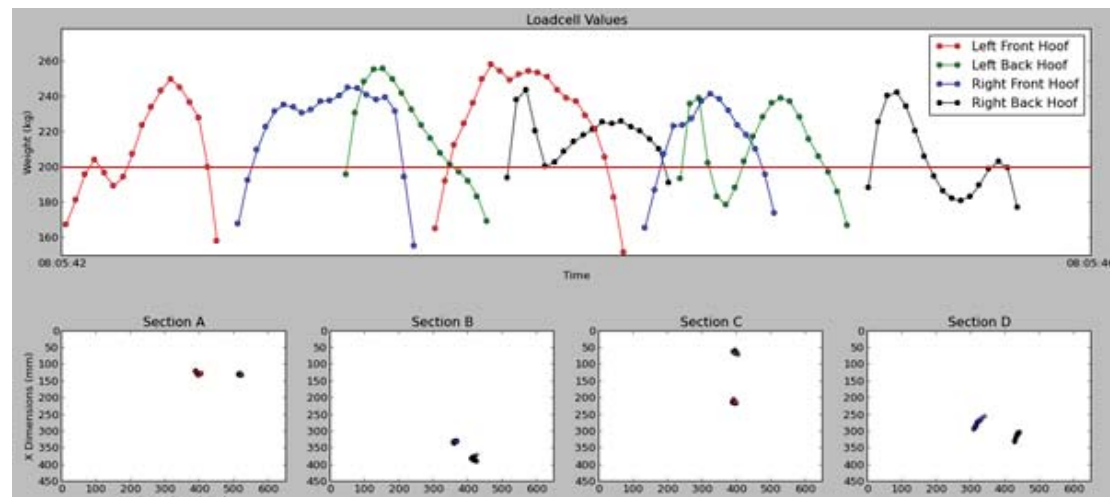


Figure 87: Setting Threshold Value of 200kg

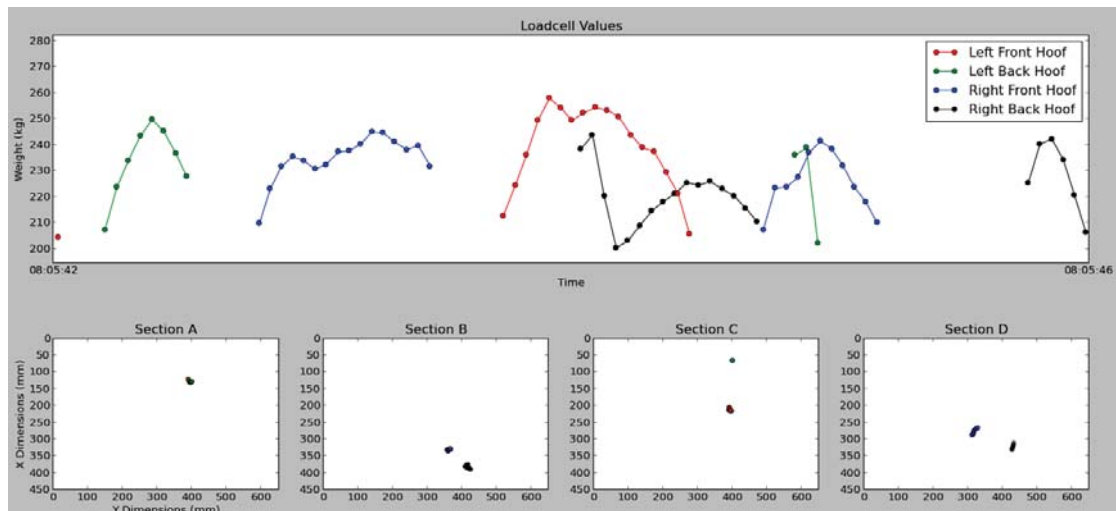


Figure 88: Result of Setting Threshold Value of 200kg

2. The second method to calculate the stride length is to find the peak value of each peak and then take a certain number of points around the peak value. This is an okay method to calculate the stride length of the cow. If one was to use too many points on either side of the peak, the second peak's points will start making use of the first peak's points to calculate the average location or vice versa. This is because the cow tends to put weight a lot quicker on its back hoof compared to its front hoof.

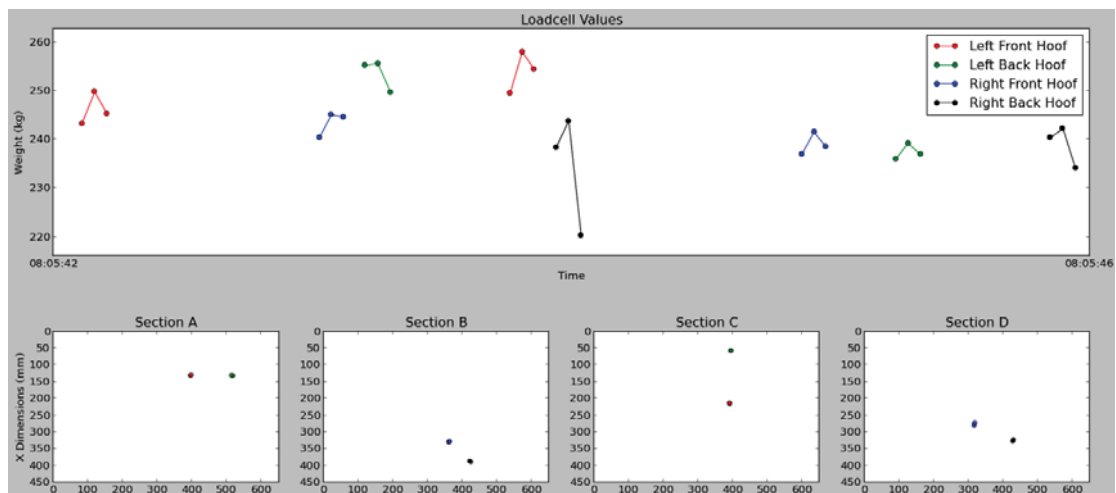


Figure 89: Result from Using 1 Point on Each Side of Peak Value

```
Stride A->B Front Hoof: 590.407453333
Stride B->C Front Hoof: 654.261973333
Stride C->D Front Hoof: 551.584953333
Stride A->B Back Hoof: 531.997603333
Stride B->C Back Hoof: 597.17045
Stride C->D Back Hoof: 659.318533333
Stride A->C Front Hoof (left side of cow): 1244.66942667
Stride A->C Back Hoof (left side of cow): 1129.16805333
Stride B->D Front Hoof (right side of cow): 1205.84692667
Stride B->D Front Hoof (right side of cow): 1256.48898333
```

Figure 90: Stride Length Result from using 1 Point on Each Side of Peak Value

3. The third method to calculate the stride length was to calculate the average location of each peak. A radius would then be set around this average location and all the data that is inside this radius will be kept. An average is then run again to calculate the stride length.



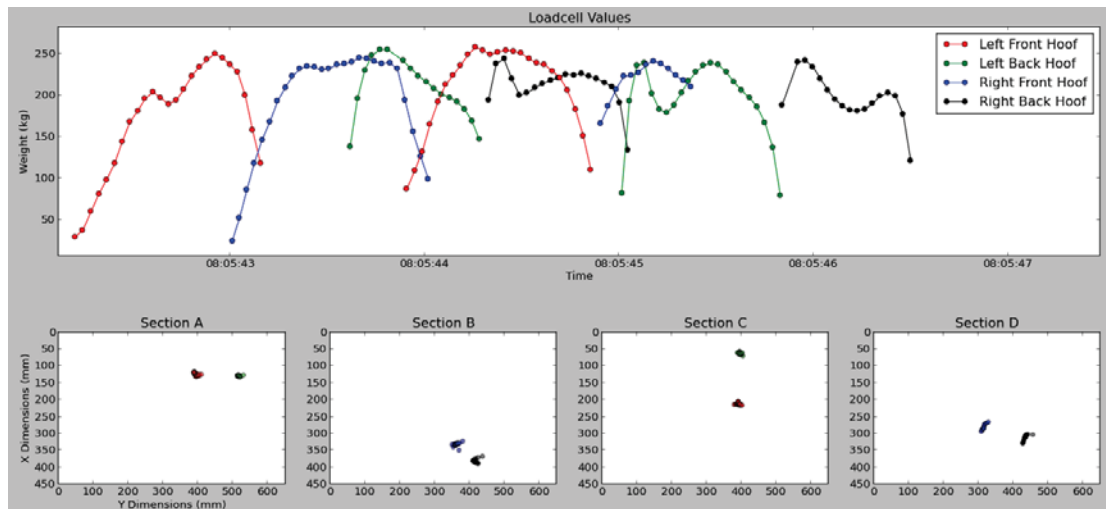


Figure 91: Result from using a Radius of 20mm

```

Stride A->B Front Hoof: 592.702821422
Stride B->C Front Hoof: 655.186565855
Stride C->D Front Hoof: 552.485418846
Stride A->B Back Hoof: 521.472314912
Stride B->C Back Hoof: 603.826047057
Stride C->D Back Hoof: 665.084983128
Stride A->C Front Hoof (left side of cow): 1247.88938728
Stride A->C Back Hoof (left side of cow): 1125.29836197
Stride B->D Front Hoof (right side of cow): 1207.6719847
Stride B->D Back Hoof (right side of cow): 1268.91103019

```

Figure 92: Stride Length Result from using a Radius of 20mm

Below is a comparison of all three methods.

Table 7: Results from All Three Methods Compared

	Method 1 (High Threshold)	Method 2 (Peak Only)	Method 3 (Radius)	Mean	Standard Deviation
Stride A->B Front Hoof	592	590	592	591.33	1.15
Stride A->B Back Hoof	571	531	521	541.00	26.46
Stride B->C Front Hoof	656	654	655	655.00	1.00
Stride B->C Back Hoof	615	597	603	605.00	9.17
Stride C->D Front Hoof	552	551	552	551.67	0.58
Stride C->D Back Hoof	667	659	665	663.67	4.16
Stride A->C Front Hoof (Left Side)	1249	1244	1247	1246.67	2.52
Stride A->C Back Hoof (Left Side)	1187	1129	1125	1147.00	34.70
Stride B->D Front Hoof (Right Side)	1208	1205	1207	1206.67	1.53
Stride B->D Back Hoof (Right Side)	1283	1256	1268	1269.00	13.53

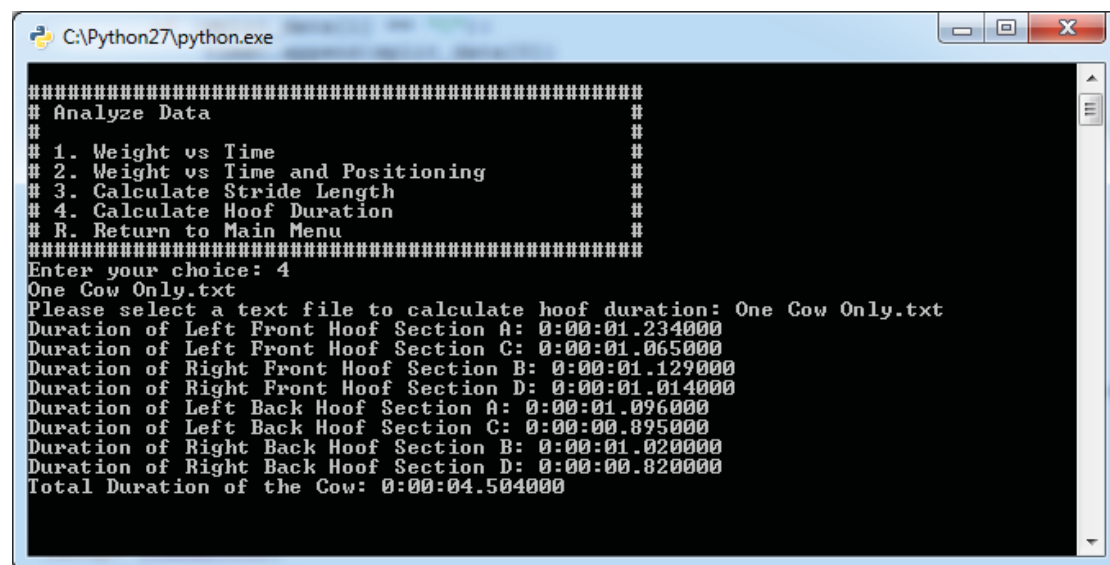


From Table 7 it can be seen that the standard deviations for Stride A->B Front Hoof, Stride B->C Back Hoof, Stride A->C Back Hoof (Left Side), Stride B->D Back Hoof (Right Side) are very large. Looking at these closer, it can be seen that the results from Method 2 (Peak Only) and Method 3 (Radius) are very similar to each other and that Method 1 (High Threshold) is causing the standard deviation to be very large.

More trials will need to be done to see how the strides vary from day to day. It will be interesting to see whether different breeds of cows have the same stride length when walking across the platform, and how these stride lengths will change compared to that of a lame cow walking across the platform.

#### 4.1.4.2.5. Calculate Hoof Duration

When the user selects option 4 “Calculate Hoof Duration”, the program will prompt the user to select a file and it will calculate the duration of each hoof on each section. A plot is created and the results are printed to the console window.



```

C:\Python27\python.exe
#####
# Analyze Data
#
# 1. Weight vs Time
# 2. Weight vs Time and Positioning
# 3. Calculate Stride Length
# 4. Calculate Hoof Duration
# R. Return to Main Menu
#####
Enter your choice: 4
One Cow Only.txt
Please select a text file to calculate hoof duration: One Cow Only.txt
Duration of Left Front Hoof Section A: 0:00:01.234000
Duration of Left Front Hoof Section C: 0:00:01.065000
Duration of Right Front Hoof Section B: 0:00:01.129000
Duration of Right Front Hoof Section D: 0:00:01.014000
Duration of Left Back Hoof Section A: 0:00:01.096000
Duration of Left Back Hoof Section C: 0:00:00.895000
Duration of Right Back Hoof Section B: 0:00:01.020000
Duration of Right Back Hoof Section D: 0:00:00.820000
Total Duration of the Cow: 0:00:04.504000

```

Figure 93: Output from Calculating Hoof Duration of Each Hoof

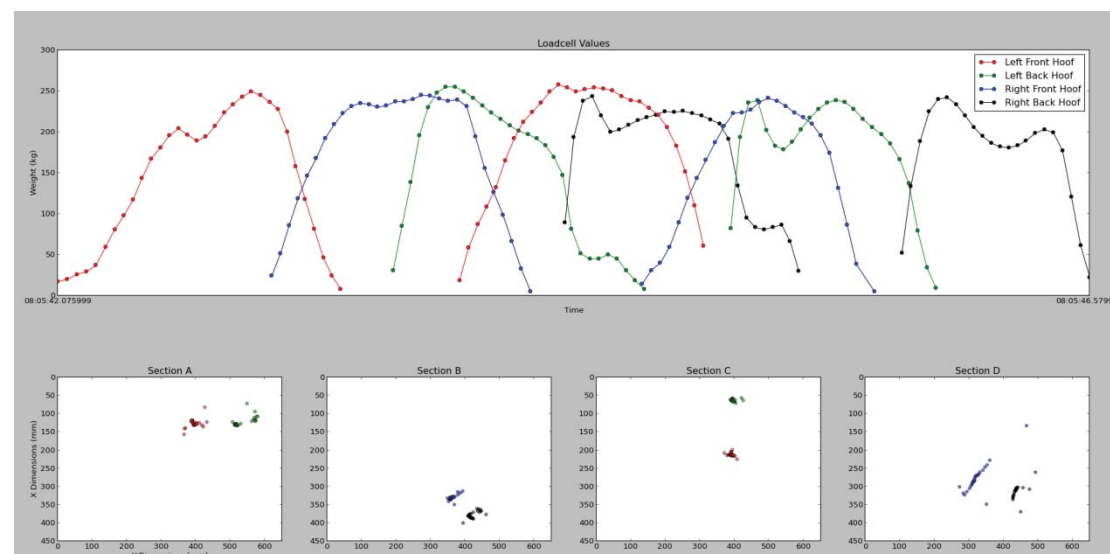


Figure 94: Cow 1 Weight vs Time and Positional Data Plotted

As it is possible to split the peaks, each peak is analysed individually to calculate the duration of each hoof. Each point has a time associated with it. To calculate the duration of the hoof, simply take the last point's time and subtract it from the first point's time. To verify whether the correct results were being produced, I zoomed in on the graph and looked at the time difference between the last and first point and found it to be the same as the results that were produced.

Below is an example of looking at the left front hoof Section C

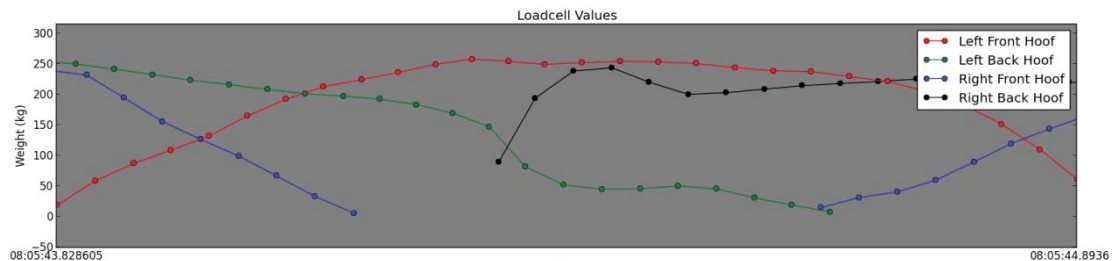


Figure 95: Example of Time Duration for Left Front Hoof on Section C

The time difference was calculated and found to be 1.065 seconds, which is the same as the results produced.

The total duration of how long the cow was on the platform for is simply the last point of Section D (black colour) subtracted from the first point of Section A (red colour).

#### 4.1.4.2.6. Calculate Average Weight

When the user selects option 4: "Calculate Average Weight", the user is presented with two options: "Running Average" or "Weighted Average" as seen in Figure 96: Calculate Average Weight Menu.

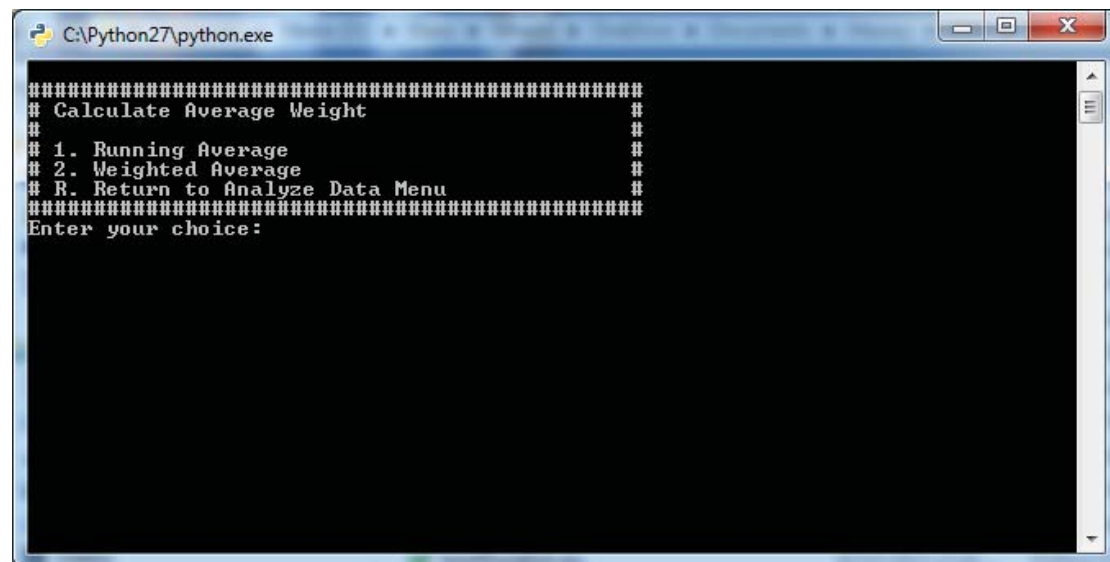
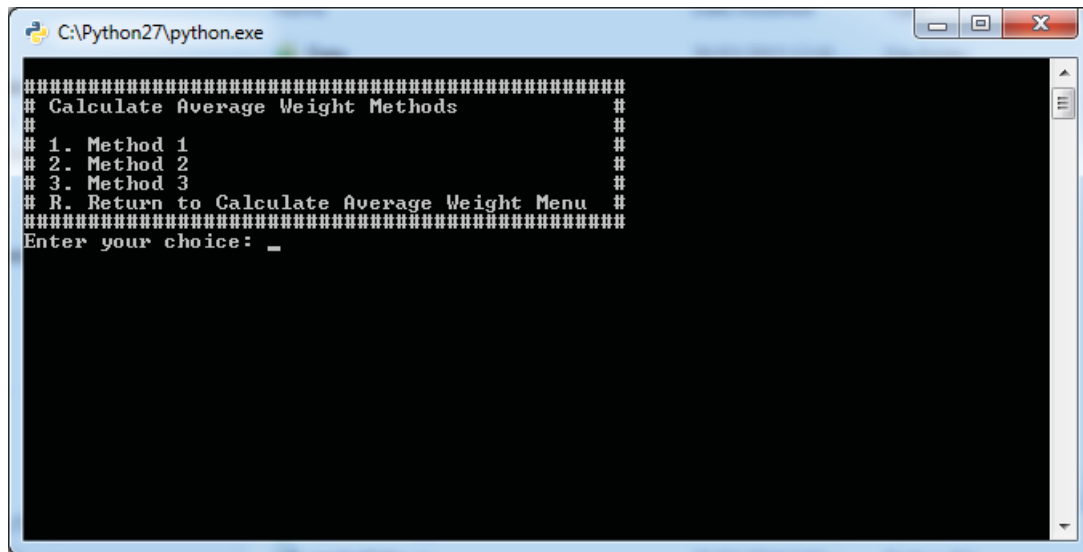


Figure 96: Calculate Average Weight Menu

The user will then be presented with another three options based on three methods (discussed over the next few pages) developed to calculate the average weight as seen in Figure 97: Prompting the User Which Method to Use to Calculate Average Weight. The menu displayed for either option will look the same.

When doing a weighted average, it was decided to give the data in the middle of the dataset the most weight and that on the edges the least weight; this is because in the centre of the dataset, the data would be the most stable if someone was to step on and off the platform section.

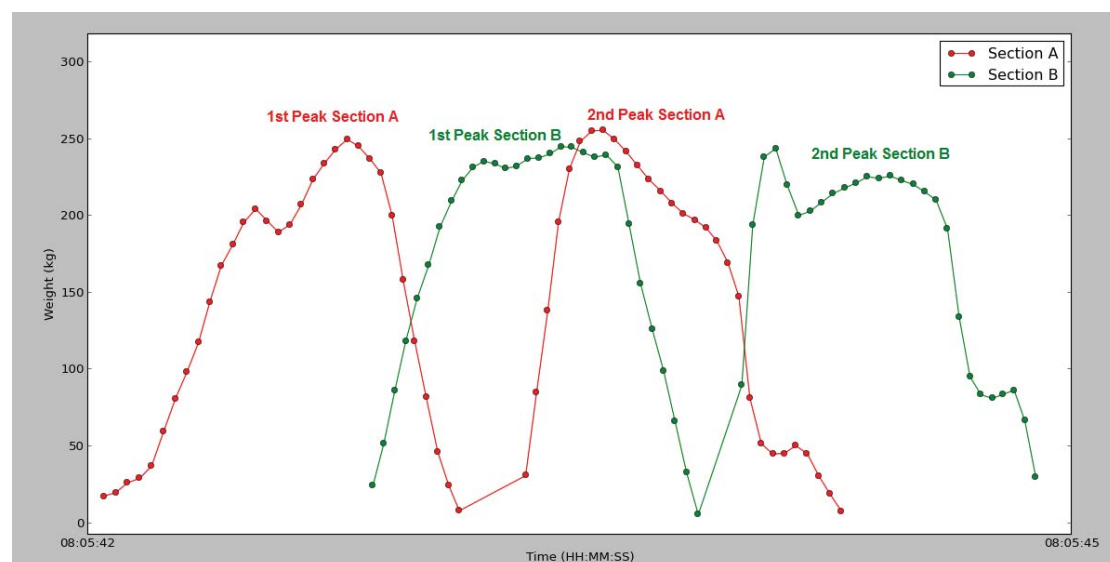


**Figure 97: Prompting the User Which Method to Use to Calculate Average Weight**

In all three methods developed, only two sections are taken at a time, as all the cow's weight will be on any two sections at any time. I look at the weight on Section AB, Section BC, and Section CD. The user is prompted after selecting which method to use as to what two sections they would like to use to calculate the average weight of the cow. It is important to note here that the method used here to split the peaks is the Method 2 as discussed in Section 4.1.4.2.1.

Method 1:

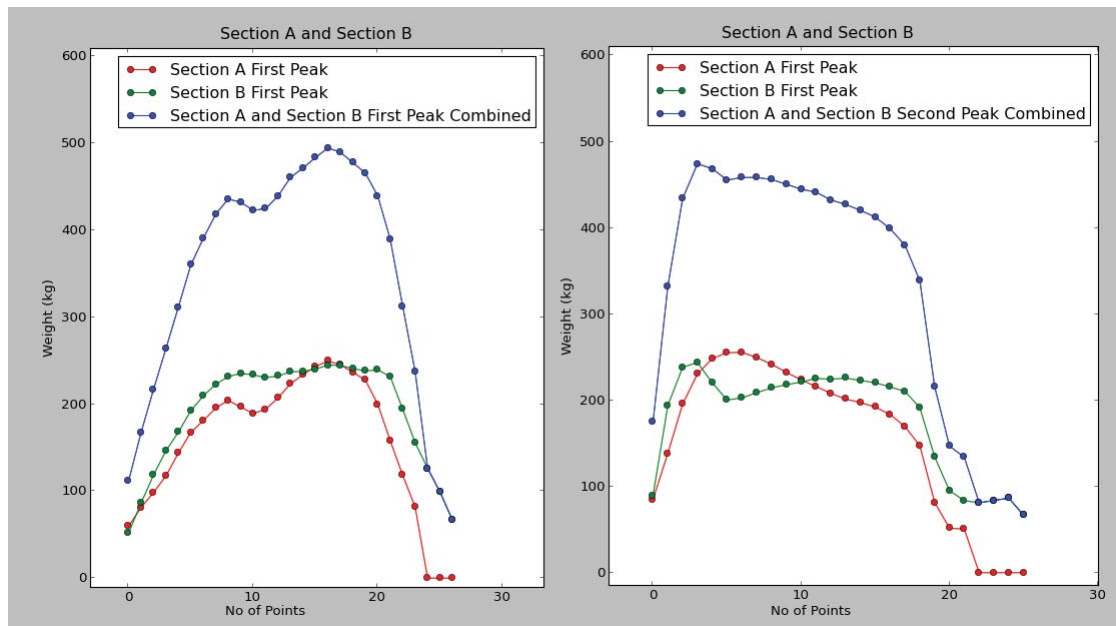
In Method 1 of calculating the average weight of the cow, the peaks of two sections are split; the first peak (the first peak represent the front hoof of the cow) that occurred on section A is added with the first peak of Section B, and the second peak (the second peak is back hoof of the cow) of Section A is also added with the second peak of Section B, see Figure 98: .



**Figure 98: First Peak on Section A is added with the First Peak on Section B, and The Second Peak That Occurred on Section A is added with the Second Peak That Occurred on Section B.**

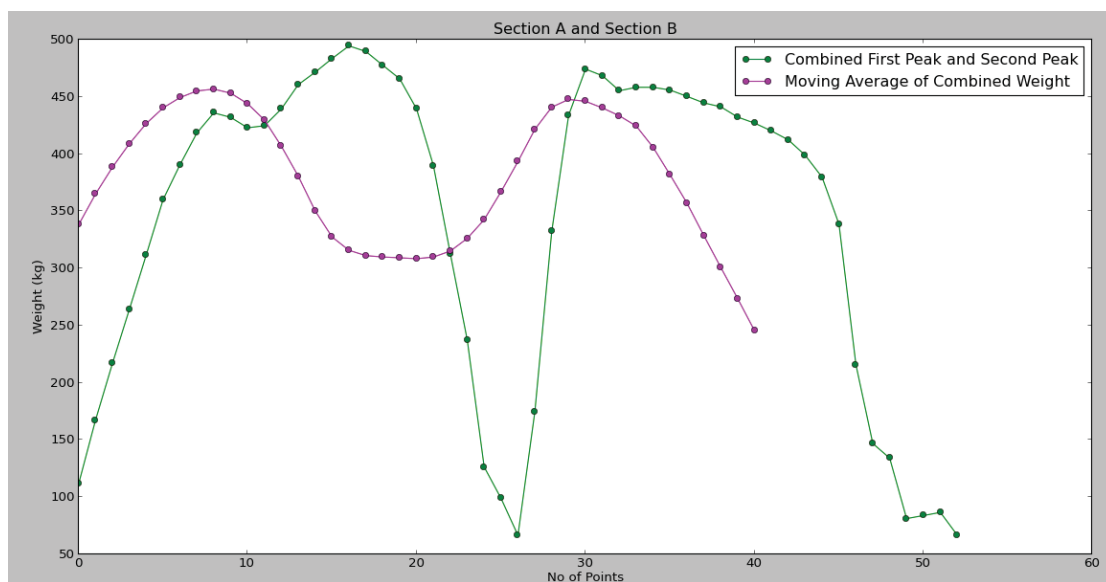
Because the first peak of Section A and the first peak of Section B can vary in length (the amount of data points), it is important to normalize the two arrays to be the same length before adding them together. The same applies when adding the second peak of Section A

and Section B together; this can be seen in Figure 99: Proof That Peaks Are Being Normalized.



**Figure 99: Proof That Peaks Are Being Normalized and added together (blue).**

When the first peaks are combined and the second peaks are combined, the data is merged again. A running average/weight average is then performed to this data to find the average weight of the cow, see Figure 100: First Peak and Second Peak Combined (Green). Moving Average Done of Combined Weight (Magenta). (moving average) and Figure 101: First Peak and Second Peak Combined (Green). Weight Average Done of Combined Weight (Magenta). (weighted average).



**Figure 100: First Peak and Second Peak Combined (Green). Moving Average Done of Combined Weight (Magenta).**

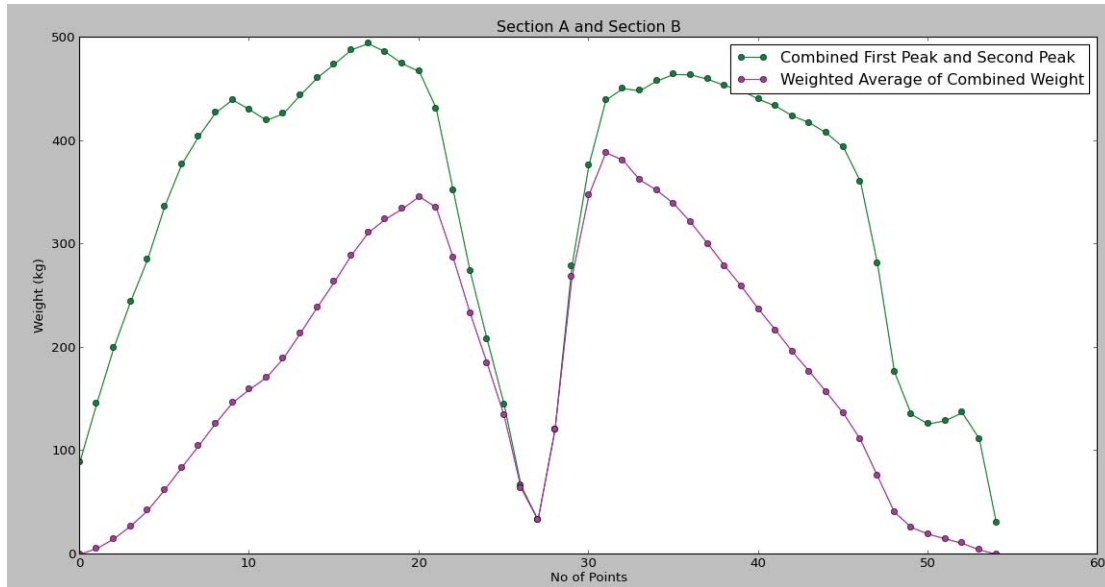


Figure 101: First Peak and Second Peak Combined (Green). Weight Average Done of Combined Weight (Magenta).

Method 2:

In method 2 of calculating the average weight of the cow, the peaks of the sections are split, the first peak and the second peak that occurred on Section A are added together, and the first and second peak that occurred on Section B are added together. Because the peaks will be different lengths again, it is important to normalize them before adding them together, see Figure 102: Proof that Peaks Are Being Normalized.

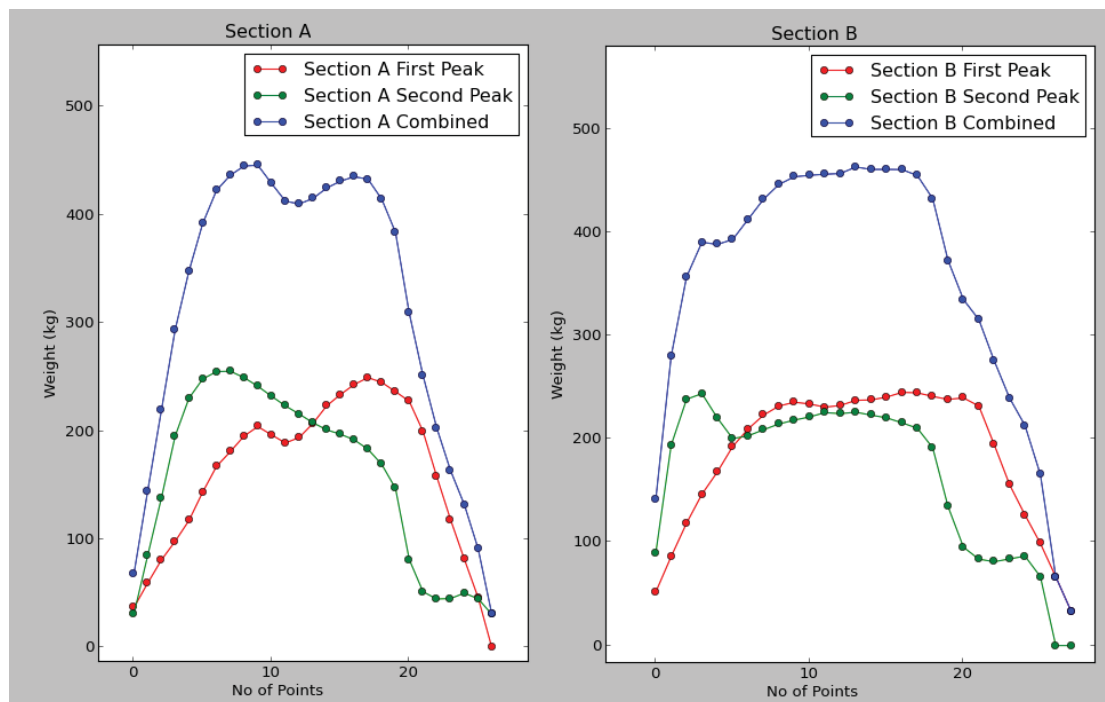


Figure 102: Proof that Peaks Are Being Normalized, and added together (blue)

A running average/weighted average is then taken on Section A to find the average weight, and a running/weighted average is then taken on Section B to find the average weight, see Figure 103: Moving Average of Section A (Left), Moving Average of Section B (Right) (moving average) and Figure 104: Weighted Average of Section A (Left), Weighted Average of Section B (Right) (weighted average).

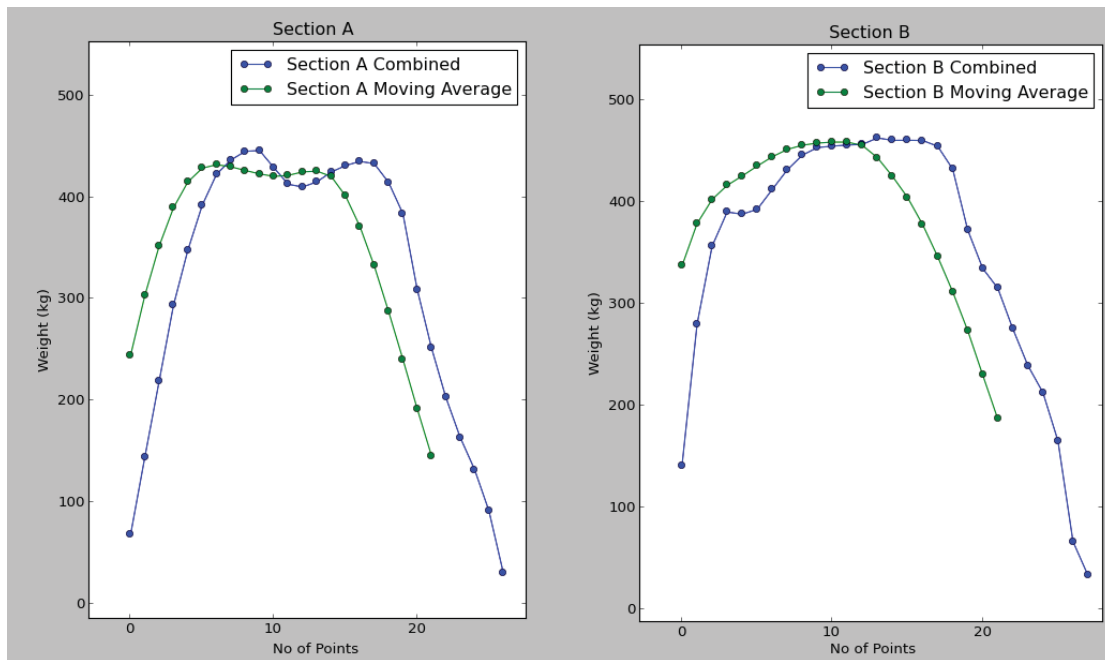


Figure 103: Moving Average of Section A (Left), Moving Average of Section B (Right)

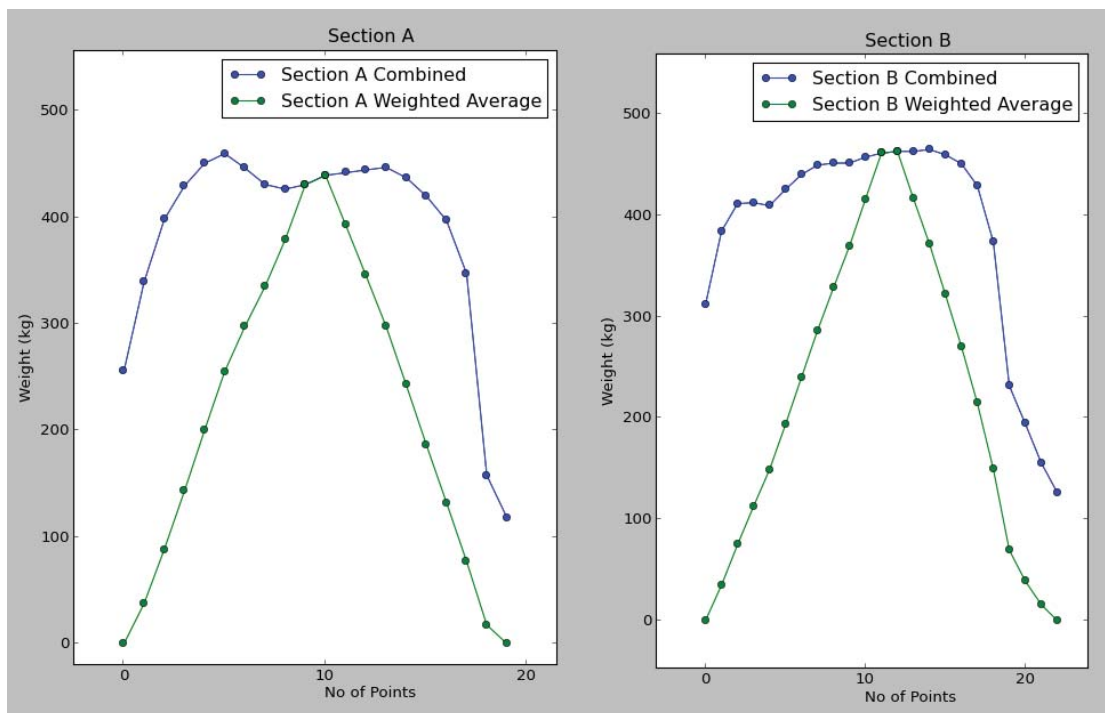
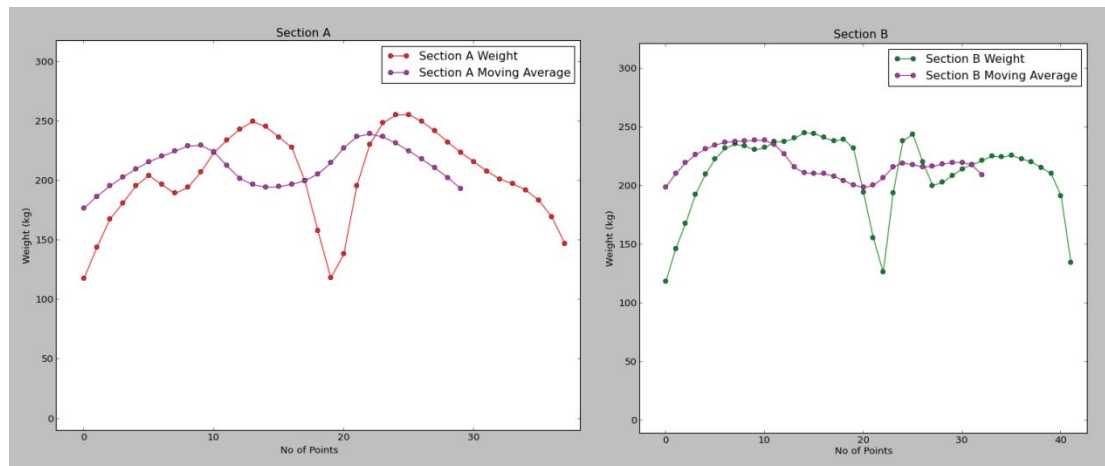


Figure 104: Weighted Average of Section A (Left), Weighted Average of Section B (Right)

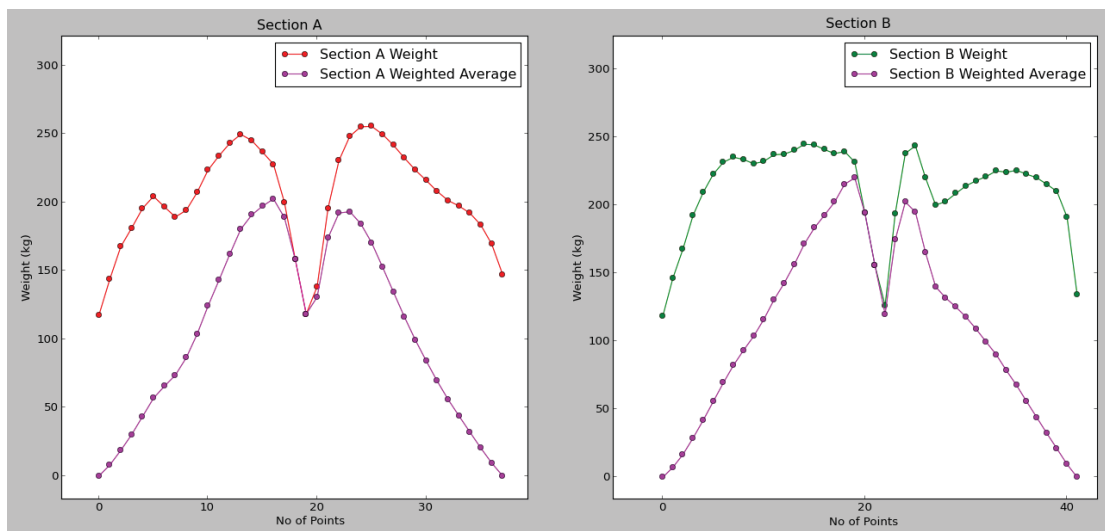
The result from each section is then added together and divided by two (as the weight would now be double seeing the peaks were added together, so it is important to divide by two) to find the average weight of the cow.

Method 3:

In Method 3 of calculating the average weight of the cow, a threshold is simply set, the data that remains from Section A is then taken and a running/weight average is performed. The same procedure is applied to Section B.



**Figure 105: Moving Average of Section A (Left), Moving Average of Section B (Right). A Threshold value of 100kg was used.**



**Figure 106: Weighted Average of Section A (Left), Weighted Average of Section B (Right). A Threshold value of 100kg was used.**

The results from Section A and Section B are added to find the average weight of the cow.

The user is able to perform the average weight for Section AB, Section BC or Section CD.

#### 4.1.4.3. Tools

When the user enters the choice 'T', the user is presented with various tools as seen in Figure 107: Tools Menu GUI.



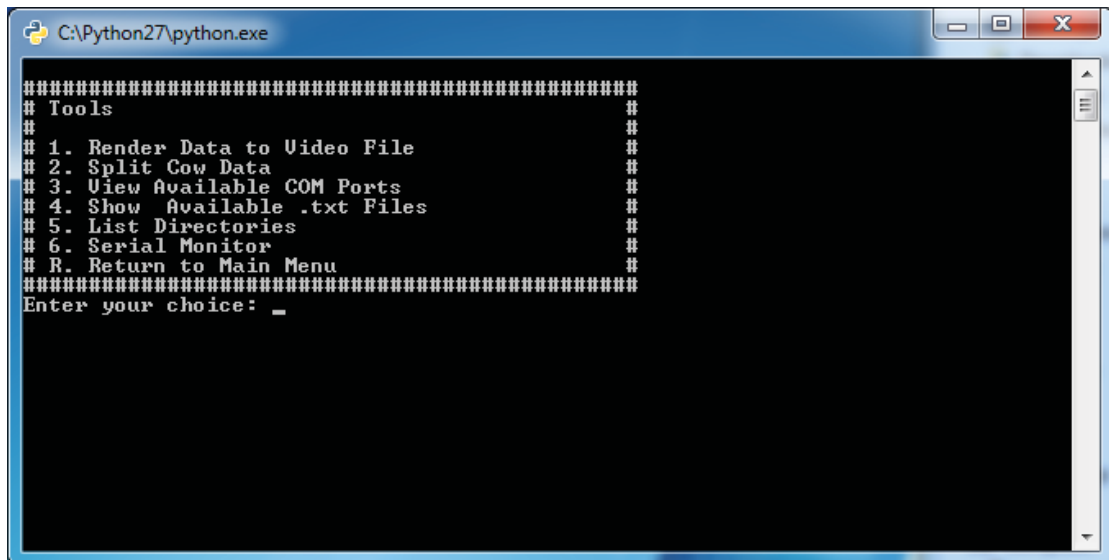


Figure 107: Tools Menu GUI

#### 4.1.4.3.1. Render Data to Video File

This option enables the user to select a text file that contains data, and render the data to a video. This might be handy if the user wants to examine the data to see if any anomalies are occurring.

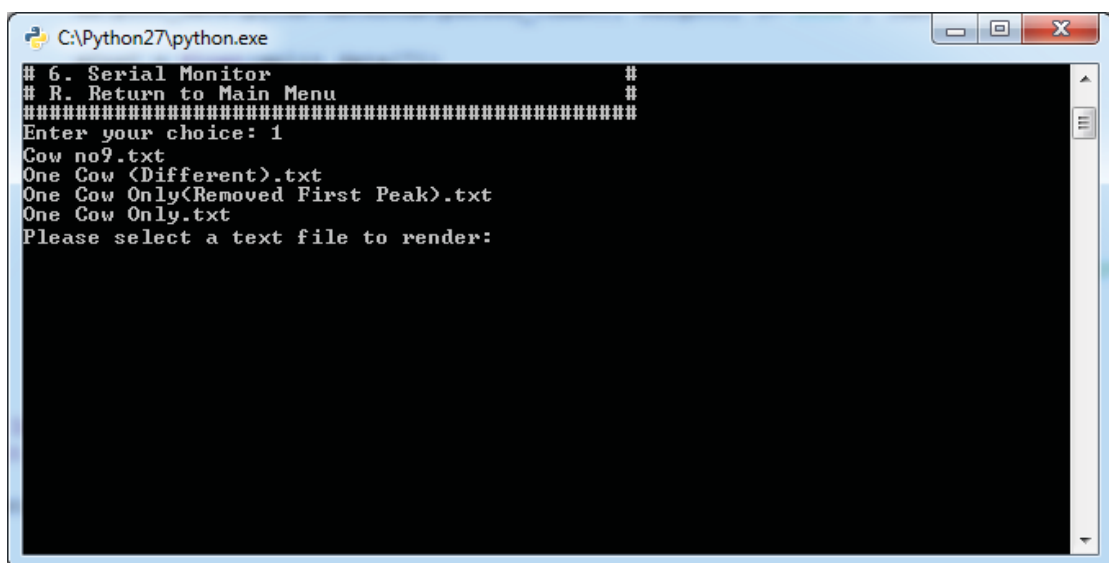


Figure 108: GUI Prompting the User to Select a Text File That Is Stored In the Data Folder

A list containing all the text files stored in the Data folder is presented to the user (see Figure 108: GUI Prompting the User to Select a Text File That Is Stored In the Data Folder). The program will then prompt the user to select a file to render. The file is rendered 24fps with a bitrate of 3000kbps and a resolution of 1600x900. All these settings can be changed to suit the user's needs.

```
# Save the file as whatever the text file is called xxxxx.mp4
ani.save(filename=inputFile + '.mp4', writer="ffmpeg", fps=24,
dpi=100, bitrate = 3000)
```

A message is printed to the user informing them that the file is currently being rendered to a video file, and also where the file is being stored. Once the file is completed rendering to a video file, a message is printed that informs the user that the rendering is complete (see Figure 109: GUI Displaying That It Has Completed Rendering the Data to a Video File). The

file is stored with the same name as the text file, but apart from having a .txt extension, it will now have a .mp4 extension, i.e. if the file was called 'data.txt' and the user renders this to a video file, it will be stored as 'data.mp4'.

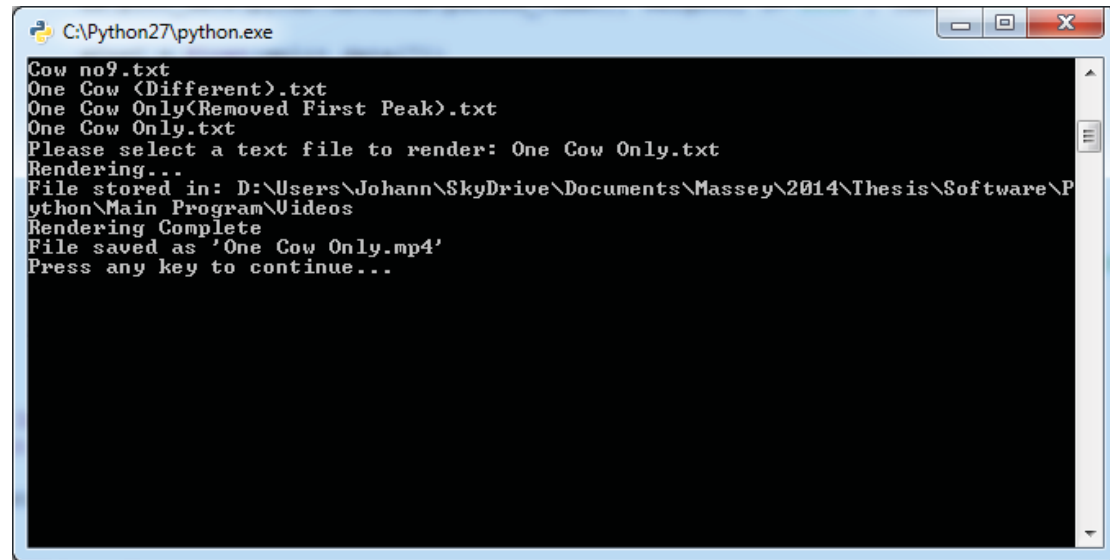


Figure 109: GUI Displaying That It Has Completed Rendering the Data to a Video File

The file can be played in a video player such as VLC Media Player or any other media player that support .mp4 files.

#### 4.1.4.3.2. Split Cow Data

This option allows the user to split the cow data by EID Number. The program simply reads the text-file line-by-line, and each line that is read is stored into a temporary array. This will continue until the line being read starts with "ID". If this happens, a file is created named by the ID (in this case the file will be called "982123468615262.txt") and the data that is stored in the temporary array is written to the file. The only issue with this (as shown Figure 110) is the ID might get written but the back hoofs might still be on section D, ultimately missing out on some data while splitting the data. This issue has not yet been resolved as more trials need be done in the future to determine how to go about solving this issue.

The date is written to the file, followed by the header and then the actual data. The date written to the file is the date the data was recorded. This is done to make it easier to build a history of the cow and to see what parameters are changing which can potentially indicate whether the cow is becoming lame.

```
Date:21-11-2014
TIME,SLAVE,CH1,CH2,CH3,CH4,WEIGHT,X,Y,PEAK
14:01:21.933000,A,32.95,36.84,1.72,13.11,84.64,76.80,338.84,P
14:01:21.973000,A,36.30,37.22,2.60,13.24,89.37,77.66,324.10,P
14:01:22.013000,A,38.01,37.21,2.96,13.14,91.33,77.26,316.48,P
Date:11-03-2015
TIME,SLAVE,CH1,CH2,CH3,CH4,WEIGHT,X,Y,PEAK
11:11:27.872000,D,3.98,7.00,18.31,43.15,72.45,66.38,176.62,o
11:11:27.918000,D,4.01,7.23,18.91,44.57,74.74,65.95,176.08,o
11:11:27.950000,D,3.90,7.15,19.41,44.26,74.73,64.81,179.12,o
```

Figure 110: Data to Expect in "982123468615253.txt"

A different file called "Sequence.txt" is also created to display what sequence the cows walked over the platform. The date the date cow walked over the platform is recorded and the actual sequence.

```

Date:21-11-2014
ID: 982 123468615262
ID: 982 123468615259
ID: 982 123468615259
ID: 982 123468615253
Date:11-03-2015
ID: 982 123468615262
ID: 982 123468615259
ID: 982 123468615253

```

Figure 111: Data to Expect in File "Sequence.txt"

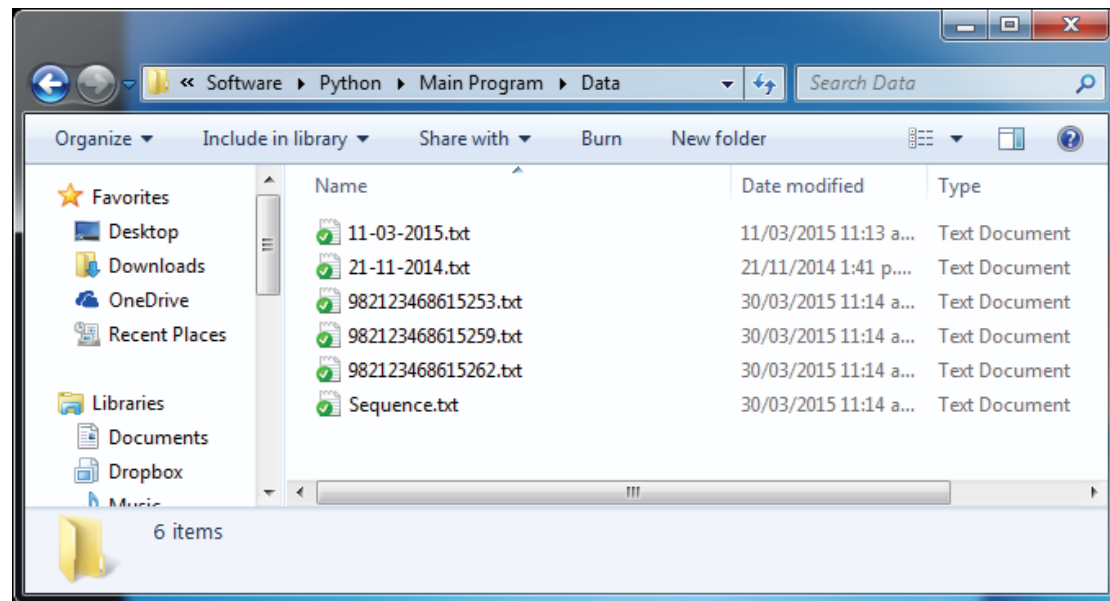


Figure 112: A Screenshot Displaying All the Different Files Created

#### 4.1.4.3.3. View Available COM Ports

This option enables the user to see what devices are currently connected to the computer. This is helpful for debugging purposes and to ensure whether the computer is detecting the Arduino, see Figure 113: GUI Printing Available COM Ports.

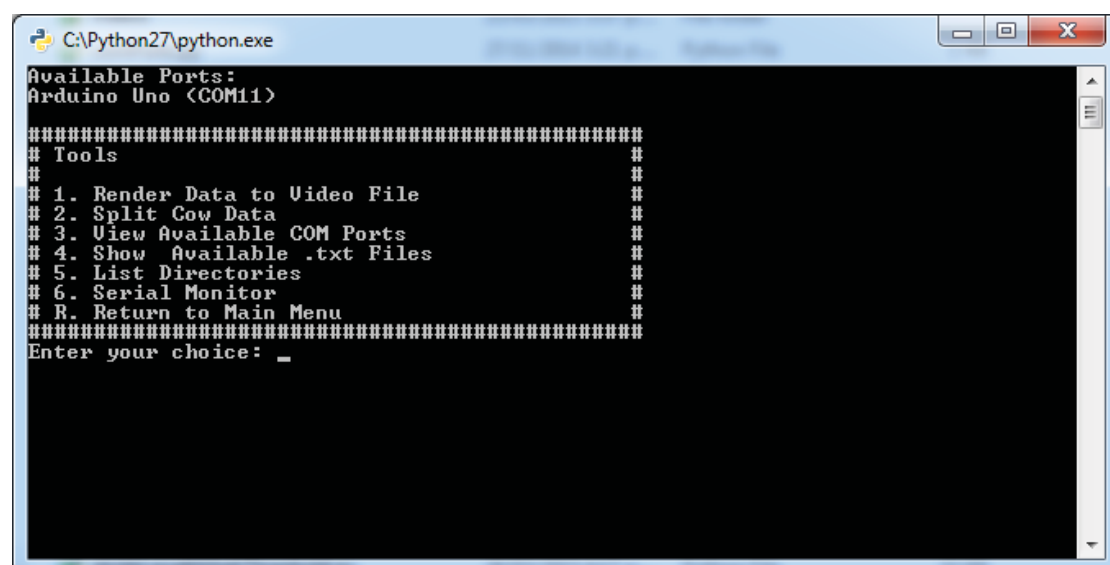
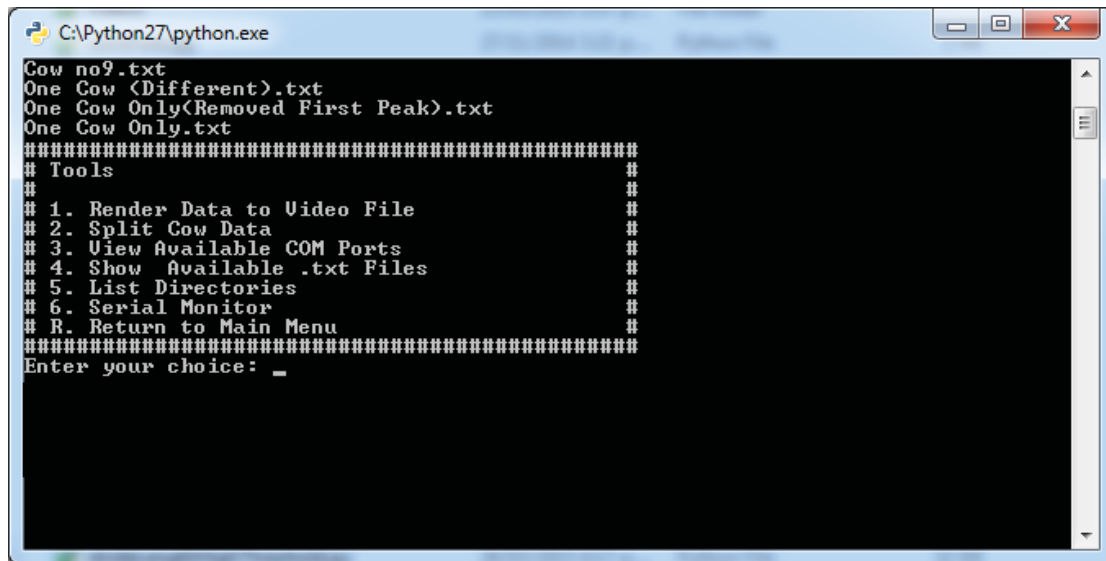


Figure 113: GUI Printing Available COM Ports

#### 4.1.4.3.4. View Available .txt Files

This option enables the user to see what text files are currently stored in the “Data” folder, see Figure 114: GUI Print All Available Text Files In the “Data” Folder.



```

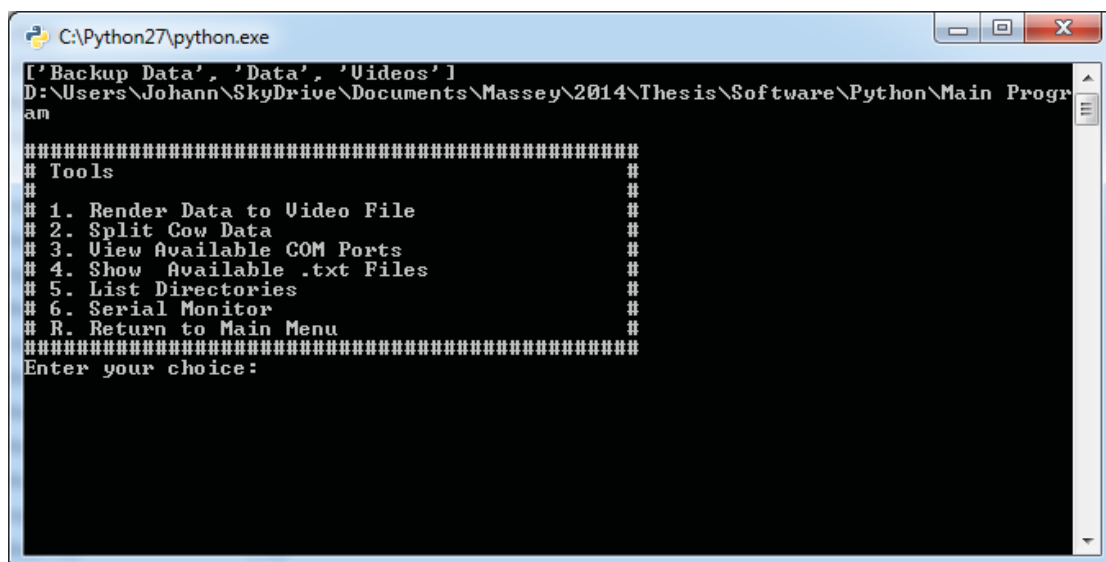
C:\Python27\python.exe
Cow no9.txt
One Cow <Different>.txt
One Cow Only<Removed First Peak>.txt
One Cow Only.txt
#####
# Tools                                     #
#                                         #
# 1. Render Data to Video File           #
# 2. Split Cow Data                     #
# 3. View Available COM Ports            #
# 4. Show Available .txt Files           #
# 5. List Directories                    #
# 6. Serial Monitor                     #
# R. Return to Main Menu                 #
#####
Enter your choice: _

```

Figure 114: GUI Print All Available Text Files In the “Data” Folder

#### 4.1.4.3.5. List Directories

This option enables the user to view all available directories. There should only be two directories i.e. “Data” and “Videos”, but the user might have some additional directories where the program is kept. The program will also print the current directory where the main program is stored, see Figure 115: GUI Printing All Available Directories and Where the Main Program is Stored.



```

C:\Python27\python.exe
['Backup Data', 'Data', 'Videos']
D:\Users\Johann\SkyDrive\Documents\Massey\2014\Thesis\Software\Python\Main Program
#####
# Tools                                     #
#                                         #
# 1. Render Data to Video File           #
# 2. Split Cow Data                     #
# 3. View Available COM Ports            #
# 4. Show Available .txt Files           #
# 5. List Directories                    #
# 6. Serial Monitor                     #
# R. Return to Main Menu                 #
#####
Enter your choice:

```

Figure 115: GUI Printing All Available Directories and Where the Main Program is Stored

#### 4.1.4.3.6. Serial Monitor

This option enables the user to connect to a serial device. All the available COM Ports are printed to the command window; the user then selects which COM port to connect to. If the user selects an invalid COM Port, an error message is printed and the user is prompted to try again. Once the user selects a valid COM Port, the data then simply prints to the command window. The user simply has to hit ‘Enter’ to close the serial communication. An example can

be seen in Figure 116: GUI Print Available COM Ports. User Selects Invalid COM Ports, Error Messages Are Printed. The User Finally Selects a Valid COM Port and Data Is Printed To the Command Window.

```

#####
# Tools                                     #
#                                           #
# 1. Render Data to Video File             #
# 2. Split Cow Data                       #
# 3. View Available COM Ports              #
# 4. Show Available .txt Files             #
# 5. List Directories                     #
# 6. Serial Monitor                       #
# R. Return to Main Menu                  #
#####
Enter your choice: 6
USB-SERIAL CH340 (COM13)
Arduino Uno (COM11)
Select COM Port to Connect to i.e. COM7: COM7
Invalid COM Port, please try again
Select COM Port to Connect to: COM10
Invalid COM Port, please try again
Select COM Port to Connect to: COM11
Successfully Connected to COM Port
Hit 'Enter' to Close Serial Connection

A CH1:91388 CH2:28005 CH3:30817 CH4:89717 2308080555
B CH1:25855 CH2:90352 CH3:41083 CH4:47714 1619343882

```

Figure 116: GUI Print Available COM Ports. User Selects Invalid COM Ports, Error Messages Are Printed. The User Finally Selects a Valid COM Port and Data Is Printed To the Command Window

## Chapter 5

### Mechanical Design and Development

As the project mainly revolves around cows walking to detect lameness, a platform needed to be designed that could capture ground reaction forces that are produced as the cattle walk over the platform. The platform consists of four individual sections as seen in Figure 117: Platform Concept with four Sections.



Figure 117: Platform Concept with four Sections

Each section is assigned its own unique ID (A, B, C, D), making it easier to identify which hoof was on which section at any one time. It was decided to make use of four sections of the way the cows walked; doing it this way also means that only one hoof will be on the section at any one time.

#### 5.1. Specifications

During the concept development stage, general design specifications were established which the platform must comply to, these are listed below:

- As the three most common dairy cattle breeds in New Zealand are between 400kg and 490kg [5], the platform must be able to withstand at least 500kg.
- The optimal stride length is between 700mm  $\pm$  50mm [33], therefore each section needs to be easily adjustable to find the optimal stride length.
- The ability to withstand high pressured water blasts as the platform will be washed down twice daily in a milking shed environment. Therefore, it is important that the electronics is waterproof and protected from direct high pressured water blasts.
- Be able to fit within the standard width of a cattle race in a milking shed.
- The platform needs to be as low to the ground as possible. This is to ensure cattle walk over the platform in a more natural way. If not, signals produced might not be useful for determining lameness.
- There are no bolts visible on the walking surface of each section.
- There are no small gaps for stones or foreign objects to accumulate.
- It is dimensionally similar to current Tru-Test weighing platforms (100mm high, 700mm overall width and 400mm walking surface width).

#### 5.2. Initial Prototype Design

An initial full sized single section was designed and manufactured at Massey University's workshop. The section was designed to test how the load cells responded with a human standing on the section and whether their centre of pressure could be accurately determined. The initial prototype was designed to have similar dimension of the sections that will be used in the final prototype platform. The initial prototype included adjustable sliders to move the load cells to find the optimal position. It was found that having the load cells as close to the corners as possible gave the best results as this covered more surface of the section. The material used for the initial prototype should be similar to the material that will be used in the final prototype platform.

A finished construction of the initial prototype design can be seen in Figure 118.



Figure 118: Constructed Prototype Section

### 5.2.1. Testing

After the initial prototype was constructed, it was required to test the section with all the electronics embedded onto the section. The test involved testing how accurately the system could measure the weight of an object placed on the section, and whether the centre of pressure could be calculated correctly. Listed below are the resources required to conduct the test:

- Arduino Uno R3
- Prototype Breakout Board with AD7193
- 4x ASB1000 Load Cells
- Prototype Section
- Point Load Stand
- 20kg Weight
- Computer running MATLAB



Figure 119: 20kg Weight Placed on Top of Point Load Stand (Left), Centre lines drawn on sides to help align weight with grid (Right).



### System Configuration

- The AD7193 was configured to have an output rate of 300Hz. Because four channels are enabled, the effective output rate per load cell is only 75Hz.
- The AD7193 was also configured to have a gain of 128, therefore the RMS noise is 85nV [32].
- The load cells were adjusted to be as close to the corners as possible in order to give the maximum surface area.
- A grid of 425mm x 600mm with a spacing of 25mm was drawn onto the platform. This was done to ensure the point load stand would be placed on known locations. The centre of the point load stand was also found, and this was aligned with the grid to ensure accuracy during the experiment.

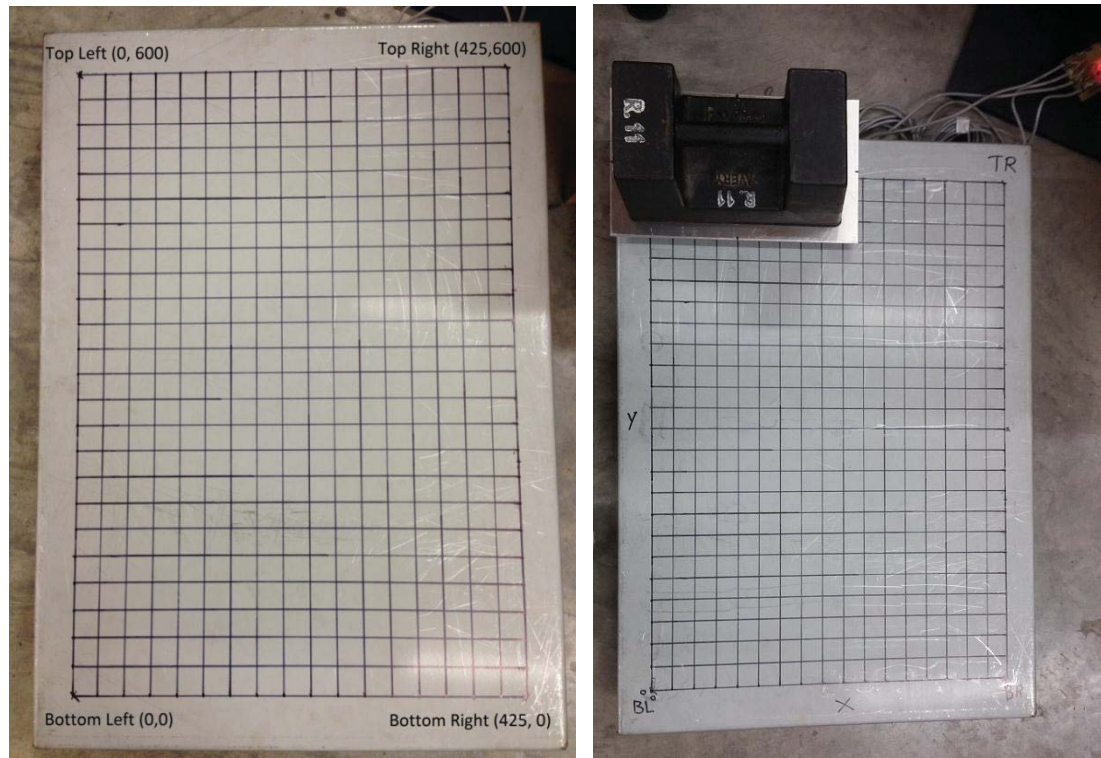


Figure 120: Platform Test Grid (Left), 20kg Point Load Placed on Prototype Section (Right)

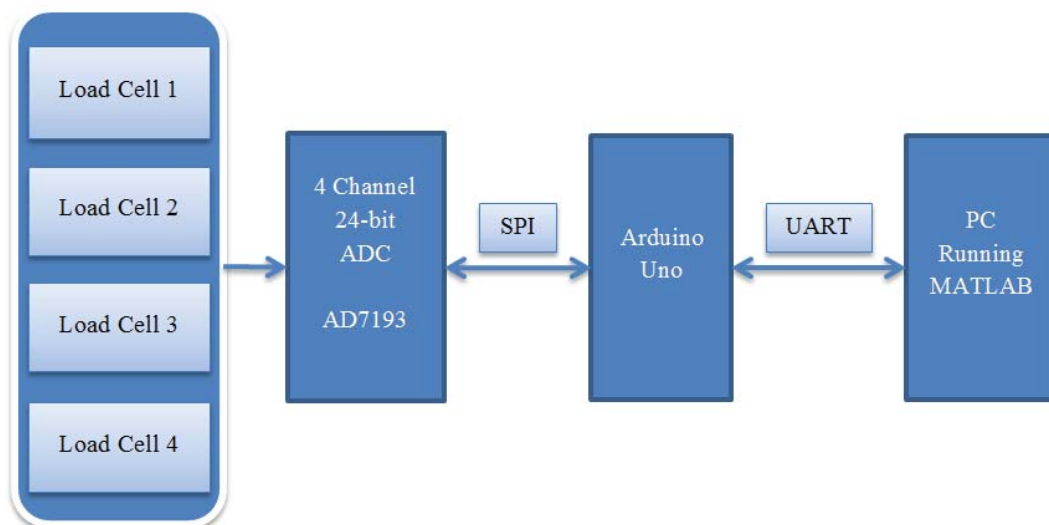


Figure 121: Block Diagram of Test System

From Figure 121, it can be seen that all four load cells interface with the AD7193. The AD7193 and the Arduino Uno communicate using the SPI bus. The Arduino then simply forwards the results to the computer for further processing. The computer runs MATLAB to do the number crunching.

## Results

**Note:** The total weight shown on Figure 122 to Figure 126 is not correct as the weight of the point load stand was not deducted when the graph was being plotted. The results on the left hand side of the graph however are correct.

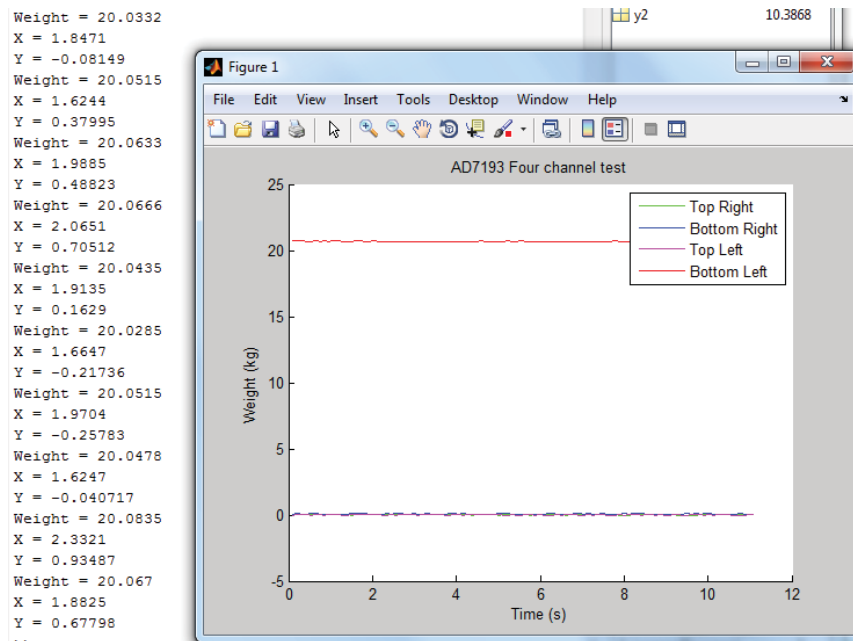


Figure 122: Load Placed on Bottom Left Corner (0, 0)

In Figure 122: Load Placed on Bottom Left Corner (0, 0), it can be seen that the bottom left load cell measured a weight of about 20kg and the X-coordinate was approximately 1mm, and the y-coordinate was approximately 0.5mm. This was expected as the weight was placed in the bottom left corner.

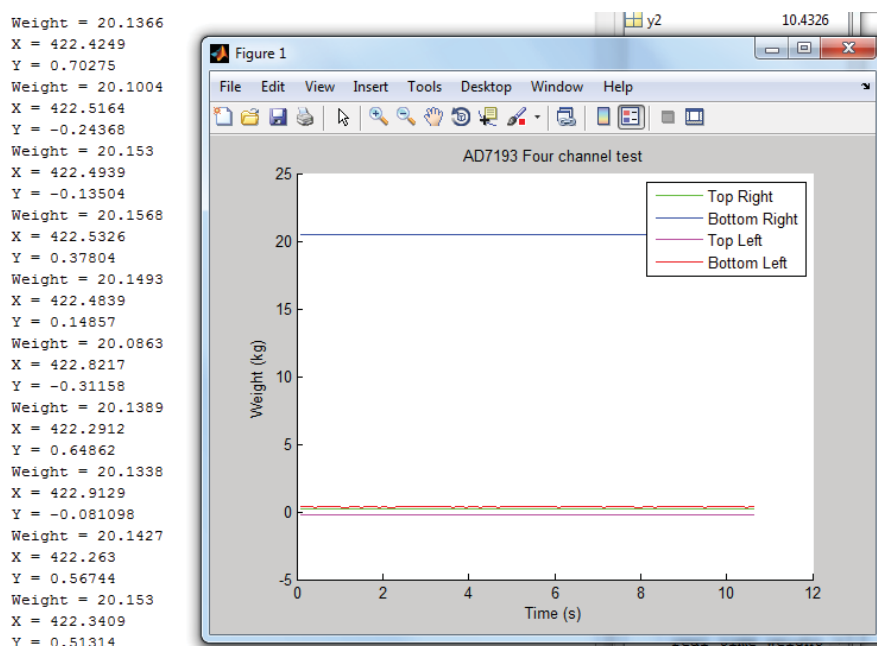


Figure 123: Load Placed on Bottom Right Corner (425, 0)

In Figure 123: Load Placed on Bottom Right Corner (425, 0) it can be seen that the bottom right load cell measured a weight of about 20kg and the X-coordinate was approximately 422mm, and the y-coordinate approximately 0.5mm. This was expected as the weight was placed in the bottom right corner.

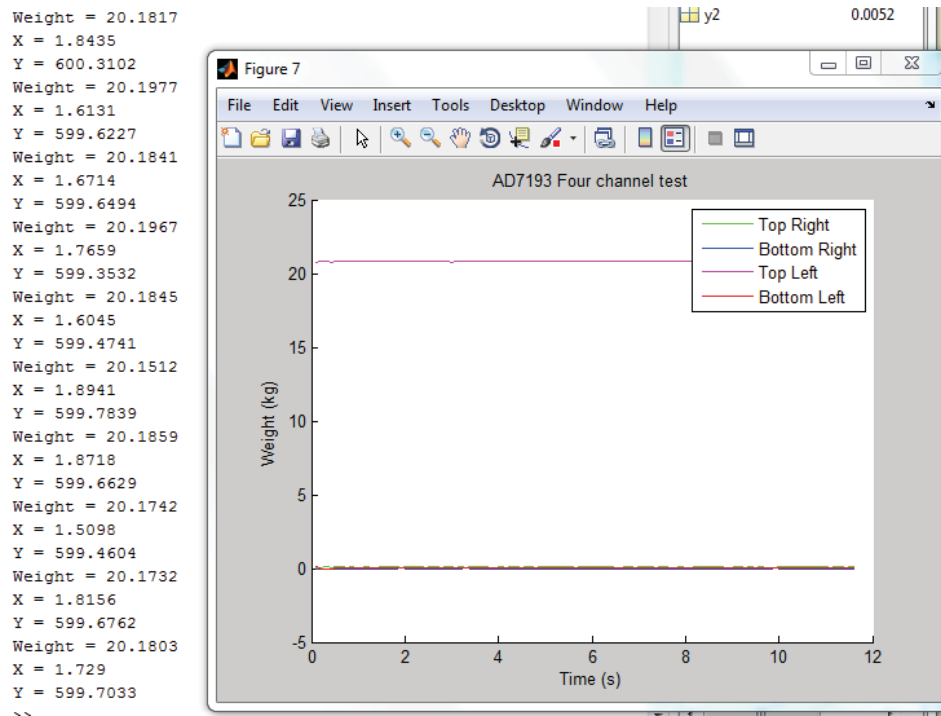


Figure 124: Load Placed on Top Left Corner (0, 600)

In Figure 124: Load Placed on Top Left Corner (0, 600) it can be seen that the top left load cell measured a weight of about 20kg and the X-coordinate was approximately 1.8mm, and the y-coordinate approximately 600mm. This was expected as the weight was placed in the top left corner.

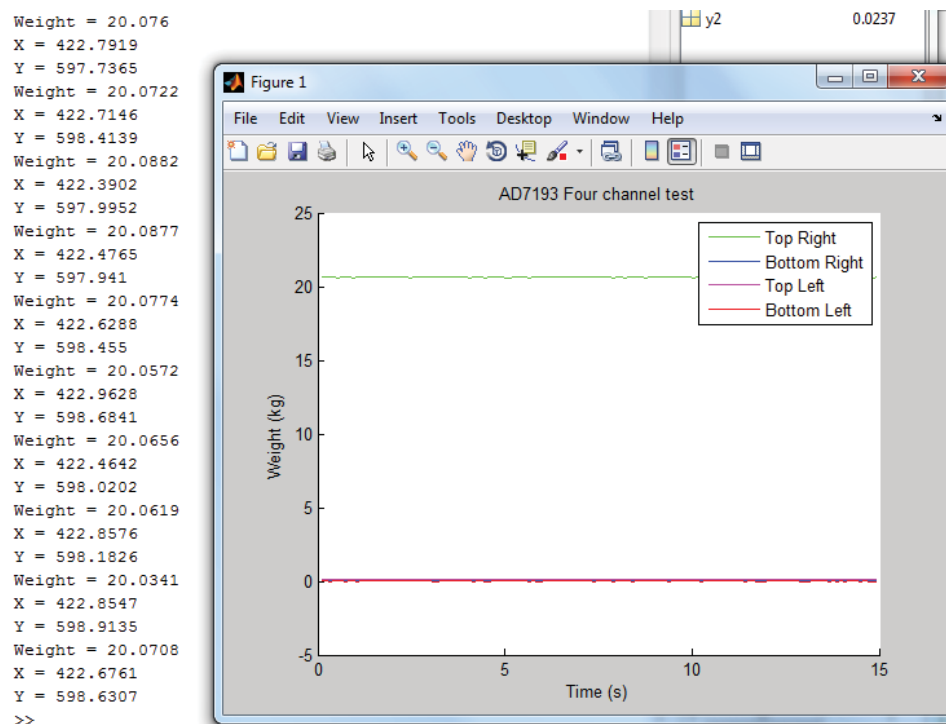


Figure 125: Load Placed on Top Right (425, 600)

In Figure 125: Load Placed on Top Right (425, 600) it can be seen that the top right load cell measured a weight of about 20kg and the X-coordinate was approximately 422mm, and the y-coordinate approximately 598mm. This was expected as the weight was placed in the top right corner.

At this point of time, it was evident that the section was measuring the weight and position very accurately. The final point of interest was to place the weight in the centre of the section to determine how accurate the system measures the location, and how well the load cells are sharing the load. The load cells should share the weight equally; 5kg per load cell.

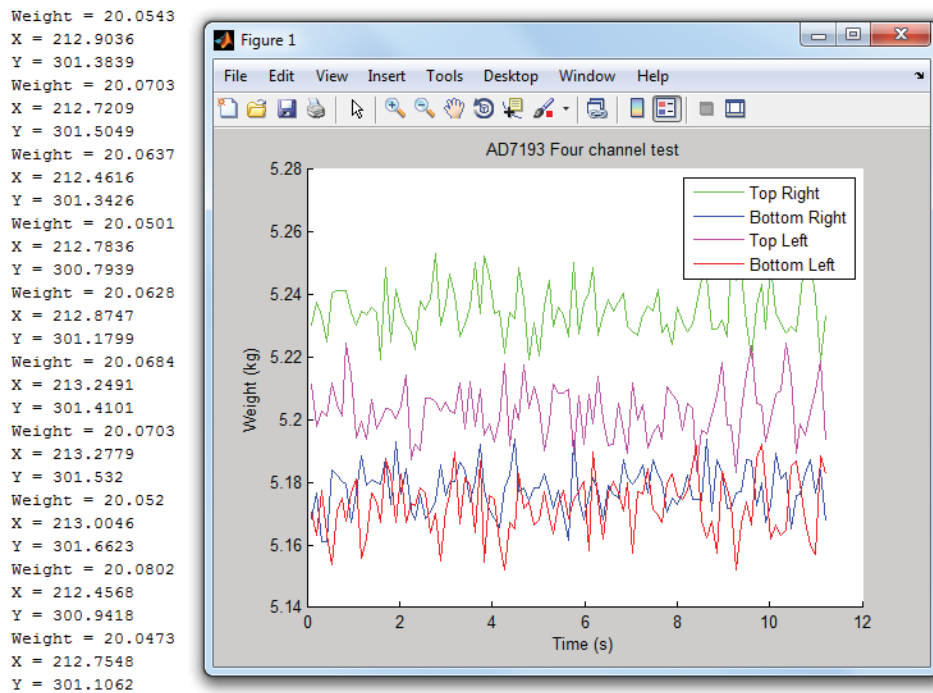


Figure 126: Load Placed in the Centre of the Section

In Figure 126, it can be seen that by having the load placed in the centre of the section, the total load still measured a weight of about 20kg. The X-coordinate was approximately 212mm, and the y-coordinate approximately 301mm. This was expected as the weight was placed in the centre of the section.

The noise present as seen in Figure 126 is due to the AD7193 having a set gain of 128 which caused the AD7193 to have a RMS noise of 85nV [32]. Other contributors to the noise are the load cells which has a signal output of 2mV/V  $\pm$  0.1% [29], and the voltage reference which has a signal output 4.096V  $\pm$  0.05% [30].

Please see Appendix 2 for all the results that were recorded while conducting the experiment.

The statistical results of the centre of pressure and weight accuracies can be seen in Table 8: Mean and Standard Deviation of X & Y and  $0.814 \pm 1.788$  mm.

Table 9: Mean and Standard Deviation of Weight respectively.

Table 8: Mean and Standard Deviation of X & Y Position

X-Position		Y-Position	
Mean Deviation	1.005mm	Mean Deviation	0.814mm
Standard Deviation	2.172mm	Standard Deviation	1.788mm
Minimum	-4.279mm	Minimum	-2.437mm

Maximum	5.377mm	Maximum	3.512mm
---------	---------	---------	---------

It can be seen from Table 8: Mean and Standard Deviation of X & Y Position that the X-position accuracy was calculated to be  $1.005 \pm 2.172$  mm, and the Y-position accuracy was calculated to be  $0.814 \pm 1.788$  mm.

**Table 9: Mean and Standard Deviation of Weight**

<b>Weight</b>	
Mean Deviation	20.087kg
Standard Deviation	0.034kg
Minimum	0.024kg
Maximum	0.180kg

It can be seen from  $0.814 \pm 1.788$  mm.

Table 9: Mean and Standard Deviation of Weight that the weight accuracy was calculated to be  $20.087 \pm 0.034$  kg.

The mean weight error was calculated and found to be 0.44% using a 20kg weight.

### **Conclusion**

It was concluded that the accuracy of the system overall was excellent. The x-position however had a larger standard deviation. This might have been due to human error when aligning the point load stand, vivid marks being too thick or grid pattern not being 100% straight.

Overall the results were very good, and it was decided to move onto the final prototype design of the project.

### 5.3. Final Prototype Platform Design

A number of design changes were implemented based on what had been learnt and observed after testing the initial prototype section, with the main changes being:

- The load cell sliders were removed as it was found the load cells need to be as close to the corner as possible. Therefore, the load cells will be in a fixed position.
- It was decided to remove the top structural frame used in the initial prototype. This was done so debris can't build up between the section and frame.
- Because the top structural frame was removed, the load cell sockets now had to be welded onto the section directly.
- It was decided the torsional strength had to be increased and better waterproof protection be provided for the load cells. This was achieved by changing the bottom structural frame the load cells were mounted to from 5mm angle iron to 5mm C-channel iron.
- The section cover of the initial section was made out of 1.6mm mild steel. This was increased to 3mm steel to lessen the deflection observed from the original section tray.
- The section length was reduced from 700mm to 650mm. This was done to meet the specification that the gait distance could be optimized between the ranges of 700mm  $\pm$  50mm, and to ensure the cattle's natural gait is not altered.

The final prototype platform will consist of four sections and it is possible to change the location of each section. This is to ensure the optimal stride length can be found. Once the optimal stride length has been found, the platform will go through one more iteration in the future.

The final prototype platform consists of a 3 meter long mainframe with pre-drilled locations. Each section has its own sub-frame and simply bolts to the pre-drilled locations, with these locations being 650mm, 700mm and 750mm respectively. The side rails can be attached to the mainframe directly without influencing the load cell signals.

Listed below are the key design aspects of the CAD model seen in Figure 127: CAD Model of Final Platform Design.

- The mainframe has the ability to house four sections at minimum spacing of 600mm and maximum spacing of 750mm.
- A 3 meter long side rail on either side of the platform. This was done as a safety feature and to guide the cow along the platform.
- The platform can house steppers. This is required when adjusting the sections to have different spacing, as you don't want the cow to get its hoof trapped between the gap of the sections.
- The steppers might also get the cows to walk in a certain pattern which might be useful for determining lameness.
- The electrical boxes are mounted underneath the section trays. This is done to protect the electronics from direct high pressure water blasting.
- The overall walking surface width is 400mm and the height is 100mm, which is dimensionally similar to Tru-Test's current weighing platforms.









**Figure 129: Side Rail Construction (3 Separate Sections)**

## **5.5. Assembly and Integration**

After the mechanical parts were manufactured, the parts were sent away to get hot dip galvanized to prevent the steel from rusting while being used in the milking shed. After the mechanical structure was galvanized, it was found section trays became twisted due to the galvanising process; all holes had to be re-tapped as well. The section trays had to be untwisted to ensure the section tray rests perfectly flat on all the load cells. This is important as you want the load cells to share the weight equally and don't want the section to wobble.

Now that the electronic part and all the mechanical parts were manufactured, it was required to assemble the overall system and test it.

M10 bolts were used to fix the load cells onto the sub-frame. The load cell cables were cut to length and wired into the waterproof electrical box using four IP-67 cable glands to secure the load cell cables. The load cell cable ends were fitted with female header pins, and the PCB had male header pins, making it easier to replace a load cell if required. A fifth IP-67 gland was used to secure and connect the outgoing cable to the next electrical box. Each electrical box was fitted with an IP-68 4-pin plug socket, which was used to connect the power and data lines to each section's electrical box; this created a daisy-chain configuration.

This meant that only one cable is required between each electrical box as the data lines and power lines are wired in parallel, and makes it simple to add or remove an electrical box if required.

An assembled electrical box can be seen in Figure 130: Assembled Electrical Enclosure.



**Figure 130: Assembled Electrical Enclosure**

The wiring colour code that was used to connect the plug side and the socket side of the power and data lines can be found in Table 10: Wiring Colour Code Used for Electrical Plugs and Sockets.

**Table 10: Wiring Colour Code Used for Electrical Plugs and Sockets**

Pin Number	Characteristic	Plug Side	Socket Side
1	Positive Voltage	Red	Red
2	Ground	Black	Black
3	Data Line (A)	Green	Orange
4	Data Line (B)	White	Blue



**Figure 131: Wiring of Plug Side (Left), Wiring of Socket Side (Right)**

Tru-test provided an industrial strength 20mm thick rubber mat that was used in milking sheds. The rubber mat was cut to size (650mm x 500mm) and attached to each section tray; this was done prevent the cow from slipping while walking over the platform. The rubber mat was attached to the section by using countersunk rivets, and two stainless steel strips were used to secure the rubber mat in place, as seen in Figure 132: Section Tray with Rubber Mat Attached. Countersunk rivets were chosen as this would create a smooth walking surface as no bolts are allowed to protrude as specified in the main specifications listed on page 89.



**Figure 132: Section Tray with Rubber Mat Attached**

The assembled final prototype platform can be seen in Figure 133: Assembled Platform. The steppers were designed to be directly attached to the side rails and at the same angle; this was done to increase the overall structural support.



**Figure 133: Assembled Platform**

## Chapter 6

### Experiments and Results

#### 6.1. Laboratory Testing

After the platform had been assembled, it was necessary to conduct a range of tests to measure the response and accuracy of the overall platform before installing it at Massey University Dairy Farm Number 1. The following features were looked at:

- The ability to accurately calculate the total weight on the platform.
- The measuring of the impulse response.
- The AD7193 averaging feature.
- The accuracy of weight and positional data with and without industrial rubber mat.
- How the industrial rubber mat affects the dynamic response of a section.
- Human walking signal signatures.
- Simulation of lameness.
- The ability to record the data from each section to a file for further processing.

##### 6.1.1. Calculating the Total Weight on the Platform

The total weight on the platform is calculated by taking the summation of the weights across the four sections. Because of the data being sampled at different rates for each section, an algorithm had to be developed to calculate the total weight on the platform.

The algorithm will make use of a current reading and previous reading. If the current and previous reading is equal to A, append a zero to array B, and vice versa. This is done to ensure that the arrays are of equal length when adding them together. This is done for Section B & C, Section C & D as well.

An example to prove the algorithm works:

**Table 11: Example of Calculating Total Weight Algorithm Works**

Reading number	Slave	Weight (kg)	Array A	Array B	Current Reading	Previous Reading
1	A	10	[10]	[ ]	A	
2	B	11	[10]	[11]	B	A
3	A	12	[10, 12]	[11]	A	B
4	B	13	[10, 12]	[11, 13]	B	A
5	B	14	[10, 12, 0]	[11, 13, 14]	B	B

When adding array A and array B, the result is [21, 25, 14].

This algorithm was then applied to recorded data. Figure 134: Total Weight when stepping between two Sections displays the total weight being applied when stepping between two sections.



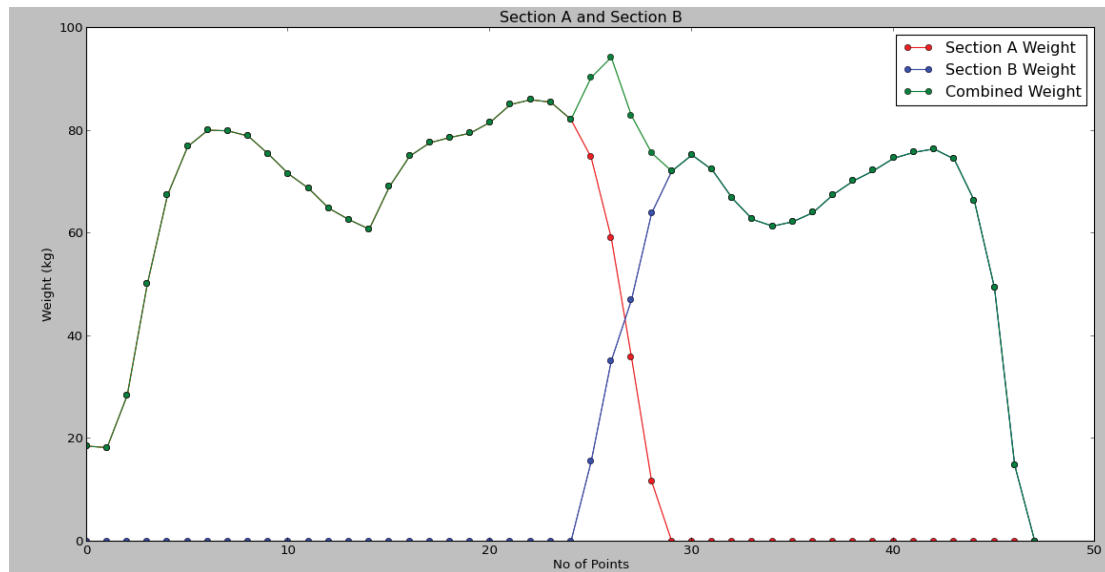


Figure 134: Total Weight when stepping between two Sections

The red signal is the weight experienced on Section A, and the blue signal is the weight experienced on Section B. As expected, the combined weight (the green line) follows the weight of Section A at the start. When the transition from Section A and Section B occurs, it can be seen as one puts more weight when stepping onto Section B (due to the heel and toe impulses that occur when walking). The combined weight then keeps on following the weight of Section B at the end.

### 6.1.2. Impulse Testing

Impulse testing was done to see how the system would respond when a large impulse would occur on a section; this might occur if a cow jumps on the section or is spooked for some reason. This test was done using a human to determine the impulse response of the system. The person would stand on the section and wait for their weight to stabilize, and once his/her weight stabilized, the experimenter gave them the signal to jump on the section. The results can be seen in Figure 135.

From Figure 135 the following can be observed:

1. The subject steps onto section and stabilizes their weight.
2. A signal is given for them to jump, and they jump.
3. The subject is the air, hence no weight present on section.
4. The subject lands on the section, and an impulse of almost 5 times greater than their normal weight occurs.
5. The subject stabilizes their weight and steps off the section.

At our visit to TruTest on the 1<sup>st</sup> of May 2014, TruTest mentioned that cows can produce impulses 5 times their weight. From the test it can be seen that results obtained from this test corresponds to their claim, even though a human was producing the signals. Because the average cow can weigh up to 500kg and can produce impulses 5 times their weight, it was important that the load cells and sections could withstand an impulse of 2500kg. The ASB1000 load cell is rated to withstand up to 1000kg, and because each section has four load cells, the section should be able to withstand weights up to 4000kg, which is 8 times greater than the average weight of a cow, this is only true if the impulse is in the centre of the platform. There is a risk of a cow overloading the load cells if the impulse is not in the centre of the section.

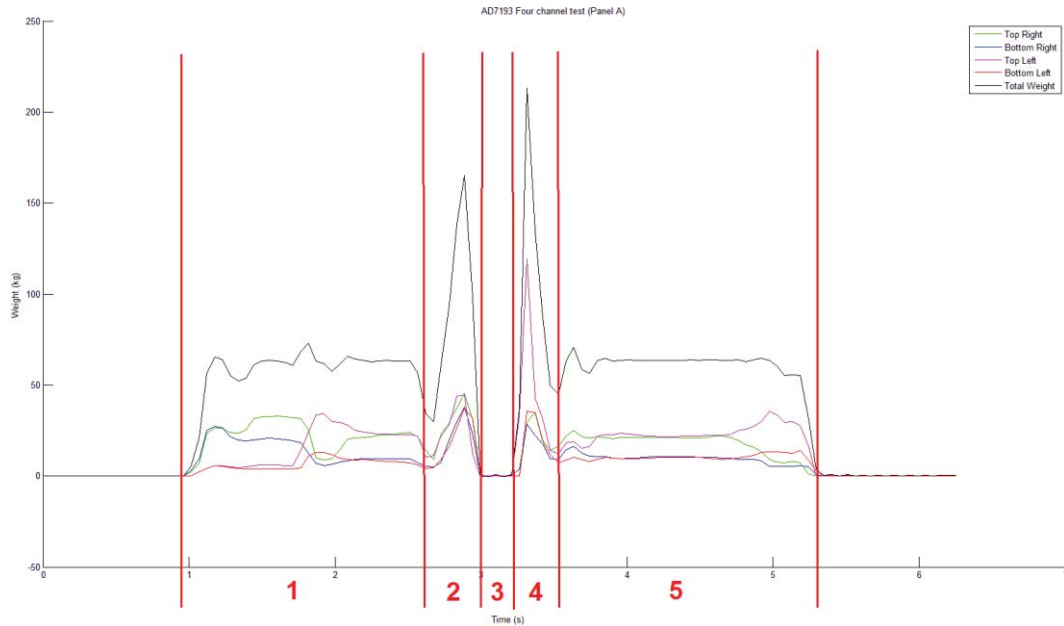


Figure 135: Jumping on Platform to Test Impulse Response

### 6.1.3. AD7193 Averaging

The AD7193 features an inbuilt averaging filter and it was decided to test how well this performed by checking whether the initial impulse that occurs when one steps or jump onto the section can be reduced. The AD7193 have four modes of averaging, this being 0 for no averaging, 2, 8 or 16. Using an average filter of 2 didn't really alter the signal in a significant way or form. It only became more apparent when using an averaging filter of 8 or 16 that the signal was being altered in a noticeable way. Using an averaging value of 16 produced the best results, but the consequence of this was the output data rate dropped significantly. Because of this it was decided to not use the averaging functionality of the AD7193 and rather let the computer do the averaging as it has a lot more resources. Please note the x-axis in Figure 136 when comparing the two images.

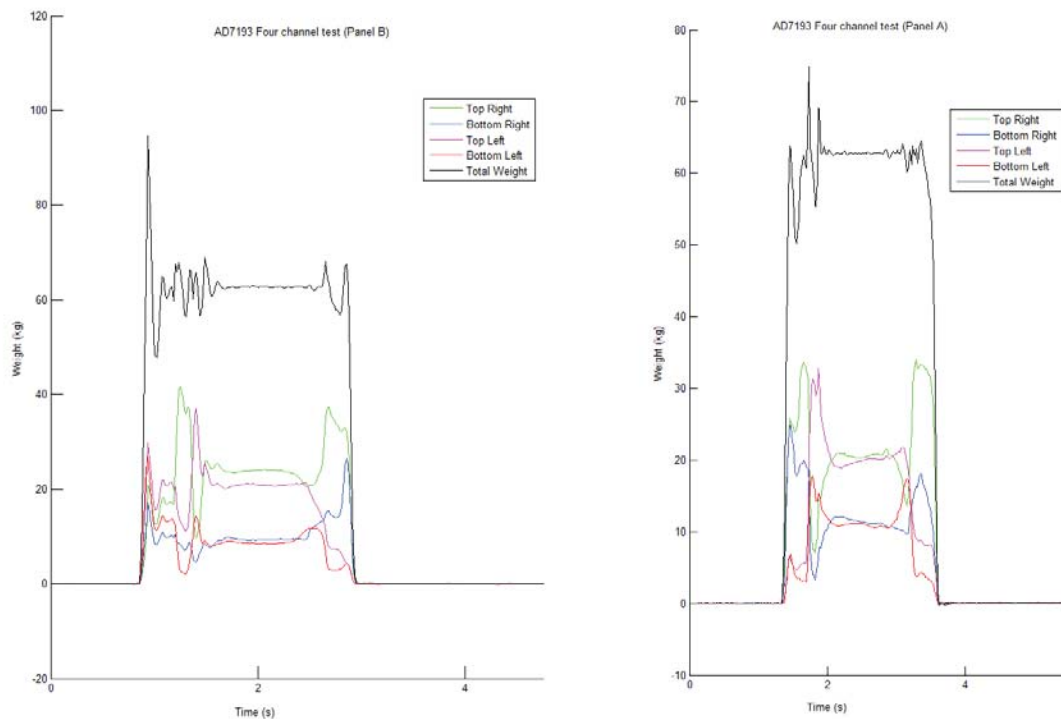


Figure 136: Comparison of Averaging of 2 (left) and Averaging of 16 (right)

#### 6.1.4. Platform Accuracy

After the platform was galvanized, it was important to test the accuracy of the platform again and see how the rubber mats affects the positional and weight accuracy. A testing jig was designed and then manufactured on the laser cutter as seen in Figure 137: Laser Cut Jig to Test Accuracy of Sections. This was done to speed up the accuracy testing, making it easier to move the weight onto known positions and at the same time creating a more precise test grid. The original point load stand and 20kg calibration weight were used from the initial prototype section to test the points on the jig.

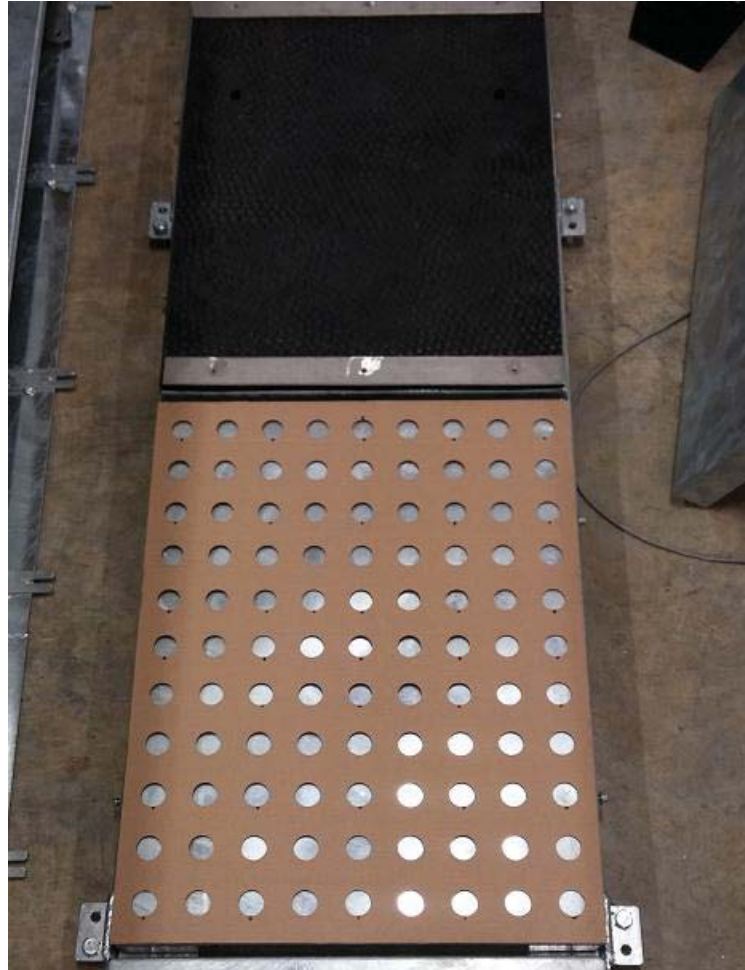


Figure 137: Laser Cut Jig to Test Accuracy of Sections

A summary of the experimental results from the testing can be found in Table 12 and Table 13 and the raw data can be found in Appendix 2. It was interesting to see the difference between the Y-Position mean errors when a rubber mat was used compared to without using a rubber mat. With a rubber mat, the Y-position mean error is  $0.3 \pm 7.2\text{mm}$ , and without a rubber mat the Y-position mean error is  $1.6 \pm 1.7\text{mm}$ . This large standard deviation occurred due to the incorrect measurements of at least 10mm at data points positioned at the front and rear of the section. The galvanized section produced less accurate results compared to the un-galvanized sections, as these sections wouldn't have been deformed.

Table 12: Mean Error of Positional Data and Weight Accuracy (Without Rubber Mat)

	X-Position (mm)	Y-Position (mm)	Weight (kg)
Mean Deviation	-6.60	-1.56	20.160
Standard Deviation	6.08	1.72	0.065
Minimum	15.00	-5.00	20.038
Maximum	6.00	2.00	20.254



**Table 13: Mean Error of Positional Data and Weight Accuracy (With Rubber Mat)**

	X-Position (mm)	Y-Position (mm)	Weight (kg)
<b>Mean Deviation</b>	-5.06	-0.30	19.965
<b>Standard Deviation</b>	8.47	7.20	0.105
<b>Minimum</b>	15.00	-10.00	19.752
<b>Maximum</b>	6.00	12.00	20.089

### 6.1.5. Dynamic Response

A test was performed to test the dynamic response time of the load cells and compared with the signals that were produced when a rubber mat was attached to a section. This test was conducted to see whether the rubber mat would affect the signals and by how much.

Figure 138: Dynamic Response without Rubber Mat shows the signals that were produced by the section when stepping onto the section without a rubber mat attached. It can be seen that the total weight (black signal) has two spikes of 70kg (which is 8kg from the average weight) when stepping onto and off the section.

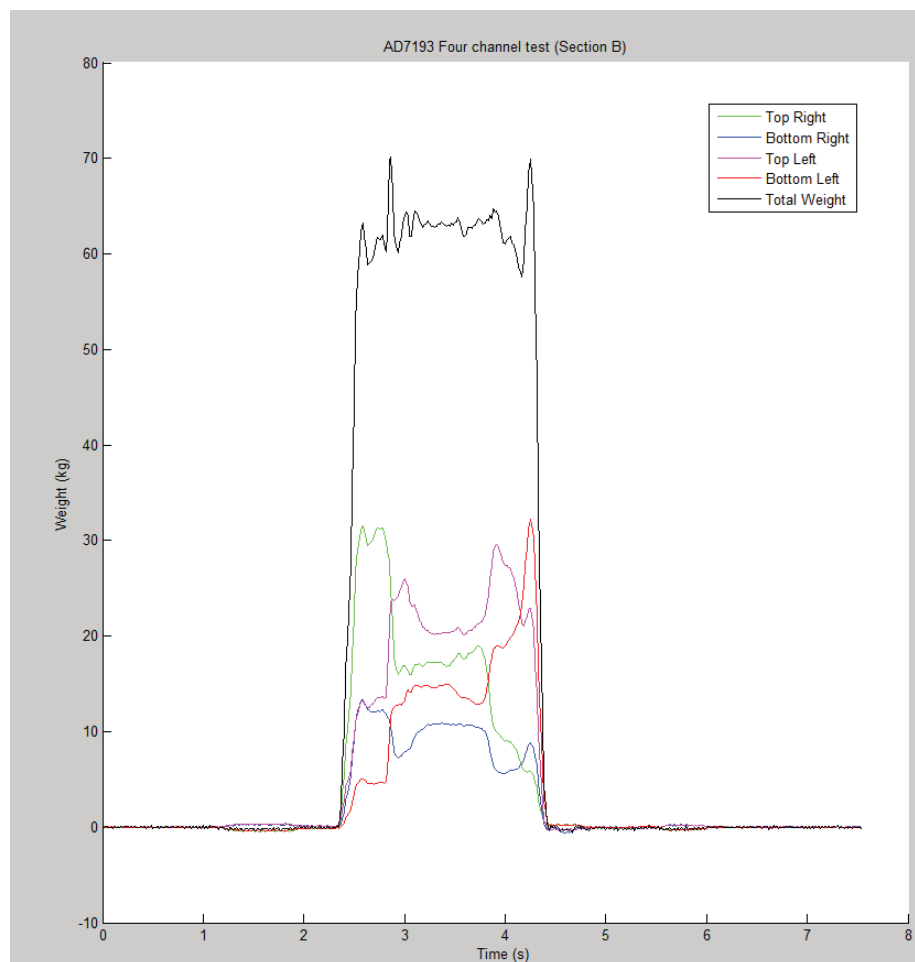
**Figure 138: Dynamic Response without Rubber Mat**

Figure 139: Dynamic Response with Rubber Mat shows the signals that were produced by the section when stepping on the section with a rubber mat attached. It can be seen that the signals produced were somewhat different. The main difference is that the two impulses were reduced to 65kg when stepping onto and off the section. Therefore, it can be concluded that the rubber mat slightly dampens the impulse signals by approximately 7%.

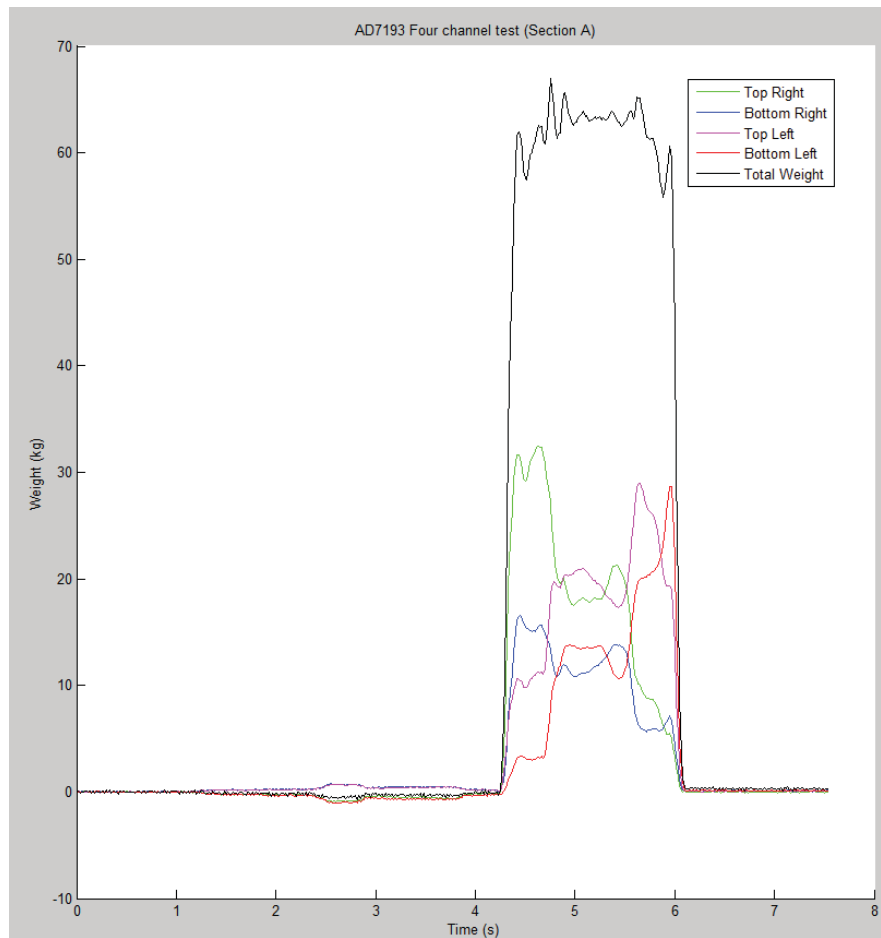


Figure 139: Dynamic Response with Rubber Mat

#### 6.1.6. Human Walking Signals

To test whether the file recorder program worked as intended, the program was started and one would walk over the platform. After it was seen that data was recorded, by simply inspecting the file, the data was taken and plotted and the results can be seen in Figure 140: Human Walking over Platform without Limp.

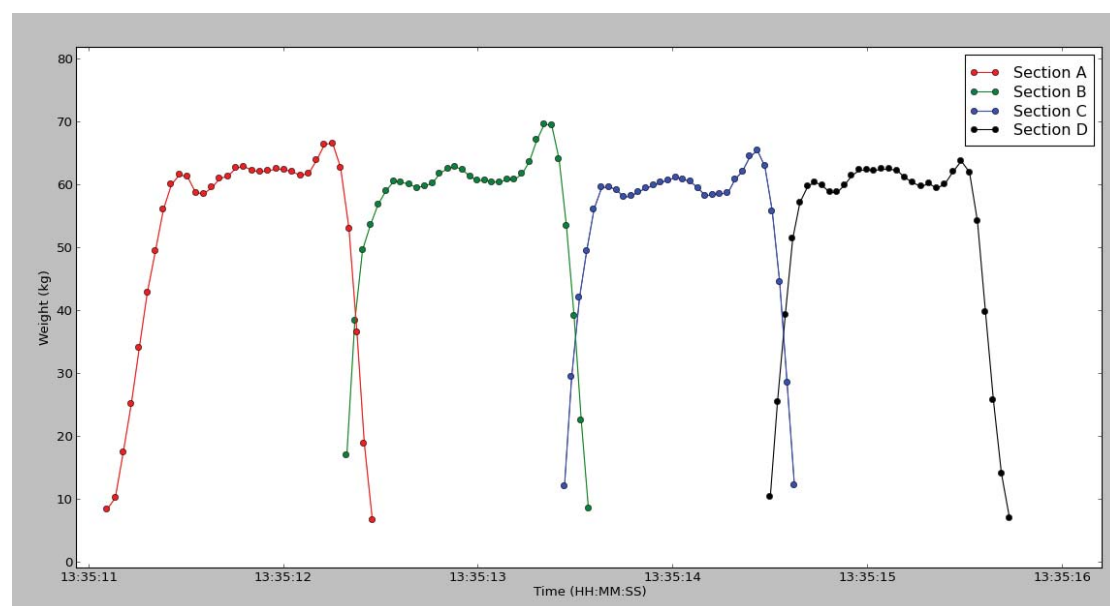
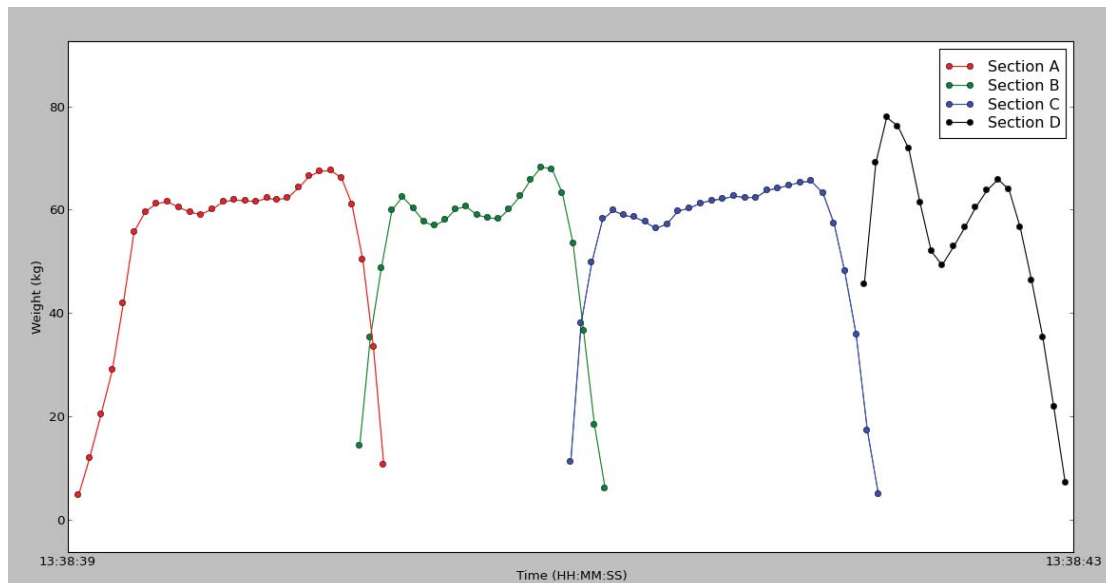


Figure 140: Human Walking over Platform without Limp

The next part of the test was to see whether simulating a lame signal would look much different from a normal walking pattern. This was done by walking over the platform pretending to have a limp in one leg. By comparing Figure 140 and Figure 141, it can be seen that the signals produced on Section A and C are very similar but those of Section B and Section D are quite different and the duration of the signals are also shorter.



**Figure 141: Human Walking Over Platform with Limp**

#### 6.1.7. Field Testing

For the field testing, the platform was taken to Massey Dairy Farm Number One and installed in the cattle crush to collect real data from the cows when they walked over after milking. The cows tended to walk over the platform in groups of 20. This data was captured to be analysed at a later stage. The platform section spacing's were configured to be 650mm; this means no stepper was installed and it allowed the cows to walk over the platform in a natural way. The installed platform can be seen in Figure 142.



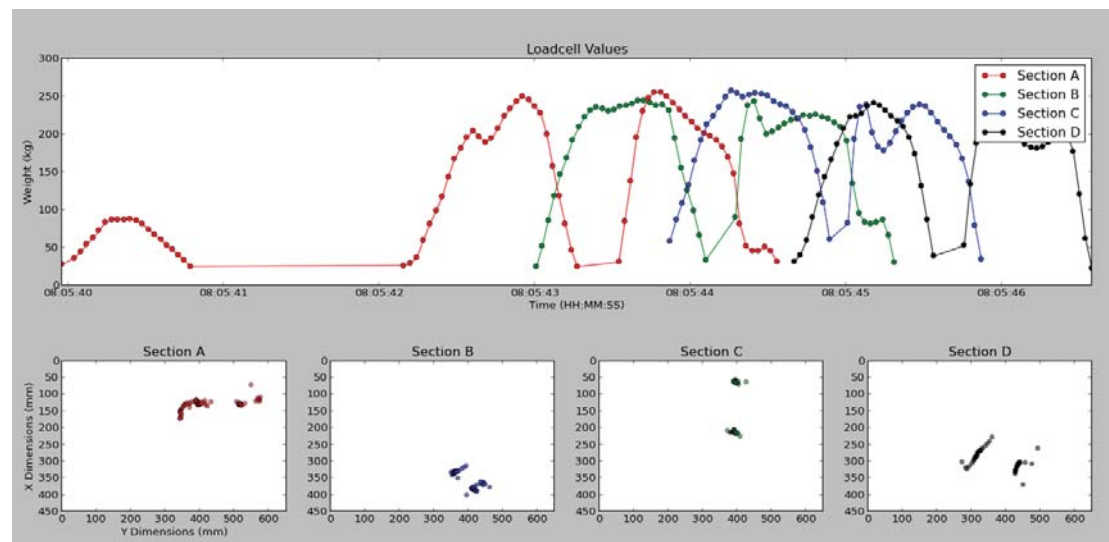
**Figure 142: Installed Platform in Crush at Dairy Farm Number 1**

Although lots of cow data was captured, it was at this point of time very difficult to separate the cow data as it was all combined and no EID reader was implemented at this stage. Therefore only cows that produced clean data such as from the first cow that walked over the platform was used.



**Figure 143: First Cow to Walk over Platform**

The first ever cow to walk over the platform can be seen in Figure 143 and produced some very clean and useful data. This cow's data was mainly used to develop all the algorithms as seen in Section 4.1.4.



**Figure 144: Plotted Data from the First Cow (Weight and Position)**

The first peak that occurred in Figure 144 was from the cow being hesitant to stand on the platform. The top part shows the total weight that was displayed on each section. At the bottom, each figure is used to represent a section and where the cow stood on the section as the cow walked over the platform. The signals are colour coded to make it easier to determine what data belongs to Section A (red), Section B (green), Section C (blue) or Section D (black). Looking at the positional data in Figure 144 it is difficult to determine what data belongs to the front hoof or back hoof on the section, therefore as discussed in Section 4.1.4.3.1; it is possible to render the data to a video file allowing analysis to be easier. The first peak that occurs will always be the front hoof, and the second peak is the back hoof.

Some observations were noticed during the trial:

- The shape of the signals produced by the front hooves is quite different compared to those produced by the rear hooves. This is because more of the cow's weight is on the front.
- The positional coordinates on Section B and Section C show the least variation and fewer outliers compared to Section A and Section D. This is because the cow steps onto the platform at Section A and steps off the platform at Section D.
- The recorded data was used to determine the average weight of the cow. It was calculated that the cow measures around 500kg.
- The stride length of the cow was calculated. The results can be seen in Table 7 on page 76 using the various methods on how to calculate the stride length as seen in Section 4.1.4.2.4.
- Two hoof strikes per section occurred as expected as the cow tends to put its back hoof in the same spot as its front hoof when walking.

#### 6.1.8. Validation of Average Weight Algorithms

It was necessary to test whether the algorithms written to calculate the average weight of a cow walking over the platform were accurate or not. Unfortunately, this test wasn't done with cows; instead two people walked over the platform to simulate the data a cow would produce.

To perform this test, both people stood on one section at a time to determine their total combined weight. Both people would then stand with a foot on two sections. This was done to get a static weight of the two people to have something to compare the dynamic weight to. The static weights can be seen in Table 14.

**Table 14: Static Weight on Section AB, Section BC and Section CD**

	Static Weight	
	50Hz	100Hz
<b>Section A</b>	168.496	167.275
<b>Section B</b>	161.407	163.004
<b>Section C</b>	167.598	167.927
<b>Section D</b>	167.598	166.587
<b>Section AB</b>	165.077	169.217
<b>Section BC</b>	161.831	162.649
<b>Section CD</b>	167.499	169.511

Both people then walked over the platform 5 times and this data was captured and analysed. This test was done at a sampling frequency of 50Hz and 100Hz. The results can be seen in Table 15 and Table 16.

From Table 15 it can be seen that using a higher sampling frequency of 100Hz resulted overall in higher mean and standard deviation errors. It can also be observed that Method 1, which uses a sampling rate of 50Hz, produced the best overall result. Note that the running average technique as discussed in Section 2.3.7 is being used here.

From Table 16 it can be seen that using a higher sampling frequency also resulted in an overall higher mean and standard deviation errors. It can be observed that Method 2, which uses a sampling rate of 50Hz, produced the best overall result. Note that the weighted average technique as discussed in Section 2.3.7 is being used here.

Comparing the running average and weighted average results with each other, it can be seen that overall by making use of a weighted average, the mean and standard deviation error decreased. Therefore, it is expected that the weighted average will be the favourable method when calculating the dynamic weight.

It is possible that 50Hz produced better results overall as the people walked over at a slower pace over the platform. More tests however will need to be done in the future to determine which method produces the best result.

**Table 15: Running Average, Different Method Results Compared to Each Other**

Running Average (50Hz)				Running Average (100Hz)		
	Section AB	Section BC	Section CD	Section AB	Section BC	Section CD
Method 1						
Mean Deviation	7.088	2.733	11.521	11.914	3.551	9.142
Standard Deviation	0.823	1.079	2.509	3.160	1.079	0.658
Minimum	6.032	1.512	7.700	8.066	2.330	8.655
Maximum	8.209	4.221	13.313	14.913	5.039	10.090
Method 2						
Mean Deviation	9.576	5.238	15.024	12.582	7.749	15.010
Standard Deviation	2.440	2.183	4.088	4.189	4.757	2.708
Minimum	7.075	3.168	9.696	8.440	3.113	10.984
Maximum	12.977	8.484	21.063	19.488	15.048	16.855
Method 3						
Mean Deviation	8.017	4.768	12.468	11.175	7.667	11.389
Standard Deviation	4.436	2.785	7.265	6.516	5.282	6.285
Minimum	2.440	2.183	4.088	4.189	3.113	2.708
Maximum	12.977	8.484	21.063	19.488	15.048	16.855

**Table 16: Weighted Average, Different Method Results Compared to Each Other**

Weighted Average (50Hz)				Weighted Average (100Hz)		
	Section AB	Section BC	Section CD	Section AB	Section BC	Section CD
Method 1						
Mean Deviation	5.003	0.401	6.394	8.024	0.608	6.454
Standard Deviation	0.869	1.399	1.012	2.317	2.503	0.439
Minimum	3.726	-0.899	4.816	5.054	-2.292	6.010
Maximum	5.993	2.611	7.366	10.334	3.714	7.060
Method 2						
Mean Deviation	5.273	0.712	-2.528	8.191	-1.682	0.076
Standard Deviation	1.552	4.473	2.451	2.026	3.850	3.414
Minimum	2.532	-5.289	-6.605	5.351	-5.808	-3.922
Maximum	6.201	5.633	-0.109	10.454	2.628	3.629
Method 3						
Mean Deviation	12.265	8.057	13.256	15.002	6.301	11.397
Standard Deviation	2.462	2.321	3.787	3.453	2.954	3.922
Minimum	10.651	4.620	6.862	10.288	2.552	5.634
Maximum	16.597	10.552	16.874	19.929	8.666	16.352



## Chapter 7

### Recommendations and Future Work

Overall, the prototype platform that has been developed functions as intended and meets all the electronic and mechanical specifications specified. There are however still numerous areas for improvement.

#### 7.1. Electronic Improvements

Listed below are various ways in which the electronics can be improved:

- The 8-channel AD7193 can be replaced with a 16-channel ADC. The 16-channel ADC inputs however will need to operate in pseudo differential mode. Differential amplifiers will need to be added up-front before interfacing with the ADC. Only one microcontroller and one ADC will then be required to interface with all 16 load cells used in the platform. The advantage of this is that there will be no need for any RS-485 communication to take place, no need for a communication protocol between master and slave devices. The disadvantage of this design is, a new PCB will need to be designed to accommodate for all 16 load cell headers, the overall reliability of the system will decrease, new code will have to be written to communicate with the ADC, a larger electrical box will be required to house all the electronics, and longer cables will be required to the load cells.
- Design a new PCB that contains a microcontroller and four AD7193 ADCs. The microcontroller then selects each AD7193 in turn and gets the results over the SPI bus. This still eliminates the need for a communication protocol as the microcontroller will simply have to have one of these AD7193 devices active at a time. The disadvantages of this design are the same as discussed earlier.
- Replace the switched-mode power adapter that powers the platform with a linear regulated power supply as it introduces less noise on the power line. The AD7193 is sensitive enough to detect these noise signals and causing peaks up to 10kg, as seen in Figure 145.

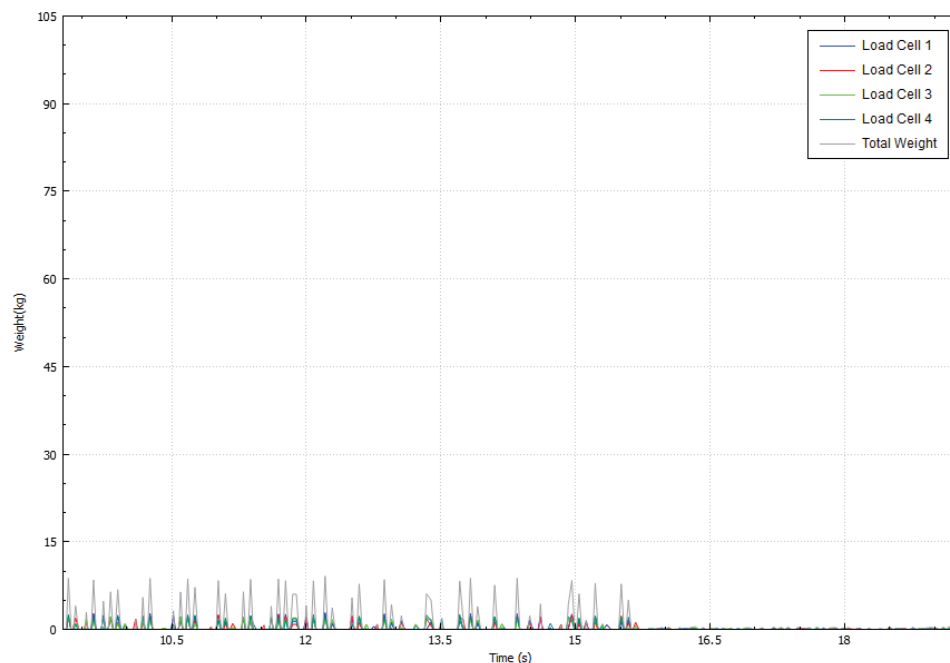


Figure 145: Noise Being Picked Up by AD7193 and Causing Peaks of up to 10kg

- Use a 2-PIN DIP switch, implemented in the electronics to change the slave ID (if one was to stick with the current design, with only 4 sections) by simply toggling the DIP switch. A 7-segment LED display can also be implemented to display the ID of the device making it visually easier to see what the slave ID is.



- Put a real-time clock and microSD card reader on the electronics. With this when the platform is powered, the microcontroller can create a new file on the microSD card named with the date i.e. "15-01-2015.txt". The date is obtained from the real-time clock. The microcontroller can then simply write the load cell values to this file. This design would eliminate the need for the user to have a computer to record data. The disadvantages of this design are that the battery of the real-time clock might go flat and incorrect files can be created, or the microSD card might not be present and data doesn't get recorded (this can be avoided by using some visual indicator such as an LED to indicate whether a microSD card is present or not). A new PCB will need to be designed.

## 7.2. Software Improvements

The algorithms developed to calculate average weight, stride lengths, hoof duration and splitting the code data all function as intended, however there are a few shortcomings. These algorithms assume that only two peaks will occur on each section, therefore further improvement of these algorithms are required. For instance, if a cow was to step on the section three times, the algorithm may discard one of the peaks, discard all of the data, or deem it as invalid data. Therefore, it is important to do more trials to see how these algorithms can be improved. The average weight algorithm can potentially be improved by creating a dynamic threshold value when calculating the weight to give more accurate results. The software can also be further improved by adding more exception handling code, so if some invalid data was passed to the algorithm, it doesn't simply crash.

A potential algorithm that might prove to be useful in determining lameness is to calculate the walking velocity of the cow. A lame cow might walk slower over the platform than a healthy cow.

## 7.3. Mechanical Improvements

A key area would be to optimize the mechanical design to make the platform more robust and use fewer parts. This can be achieved when the optimal stride length of the cow has been found, eliminating the need for steppers for different stride lengths. It also eliminates the need for four separate sub-frames as the load cells could be fixed to the main frame. The main frame can then be redesigned based on the optimal stride length with box section or a similar material with a large second moment of area; in turn this will increase the stiffness resulting in less twisting. This twisting can have an effect of the weight being reported on the other sections as seen in Figure 146.

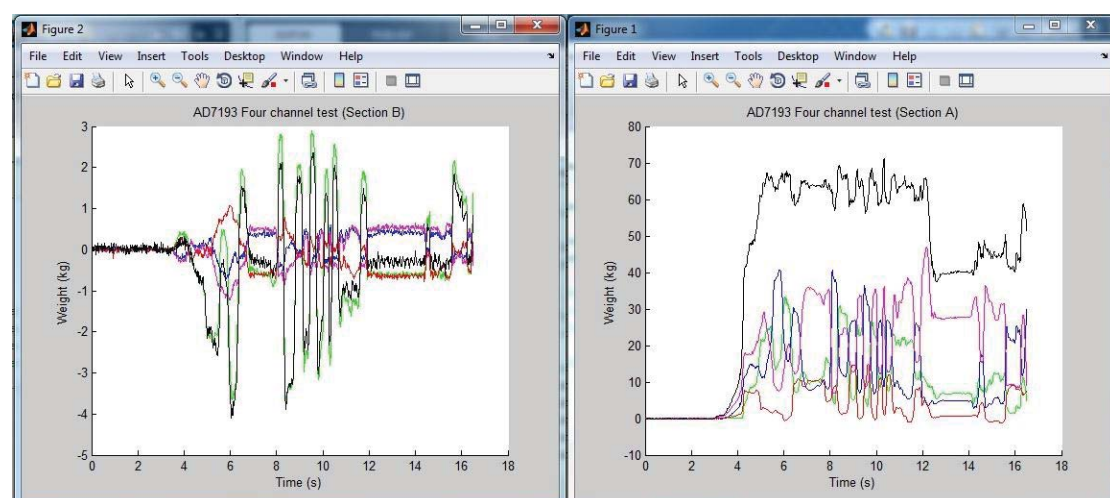


Figure 146: Standing on Section A, but Due To The Twisting effecting the Data reported by Section B

The side rails and base platform can then be folded from one continuous piece of sheet-metal, resulting in a similar product TruTest already manufactures. Another possibility is to experiment with fewer load cells if a hinge principle was developed. The inside edges of each

platform section could be hinged off the previous section and share the same load cell which could potentially reduce the total number of load cells required by six.

In the next stages of field testing the following should be completed:

- Get a control group of cows and take the static weight of each cow. Then get each cow to walk over the platform 10 times. This will enable to test which average weight algorithm as discussed in Section 4.1.4.2.6 is the best.
- Monitor the control group of cows over a certain period and see whether their signature changes or their weight or stride lengths vary. It would be beneficial if one of the cows became lame during the trial as one would be able to see how the cow signature changes. This would create a better indication of what to use to detect lameness, be able to see whether hoof duration is shorter or longer, and how the stride length changes. It would also be interesting to see how many days or times it takes the cows to become familiar with the platform and walk with a natural gait over it.
- Install an EID reader in the shed, as this will make it easier to split the cow data. The EID Splitter already creates a file to show the order the cows walk over; it would be interesting to see whether cows tend to walk over the platform in the same order or not. It might also be that the cow at the back most often is most likely to be lame.
- Install a video camera in the cow shed and record how the cows walk over the platform. This video and the data can then be given to veterinarian experts who are assisting with this project to help inform us of what cows may be showing signs of lameness. The signature of the cow can then manually be examined to see if any significant changes can be observed.

It is also recommended to further the test accuracy of the platform by using the laser cut jig and using a greater weight range to get a more evident picture of how accurate the platform is over a certain weight range and known locations.

## Chapter 8

### Conclusions

A robust platform which consists of an array of load cells has successfully been developed that can be used in the milking shed. The information from each section can be transmitted to the computer for further processing which included plotting the data, calculating the stride length, centre of pressure, hoof duration and average weight. The data can also be split and rendered to a video file.

This was achieved by designing four independent sub-frame sections that were bolted to a mainframe. Each sub-frame section consists of four ASB1000 shearbeam load cells (one in each corner), an AD7193 ADC, an ATmega 328 microcontroller, a MAX-487 chip for RS-485 communication, and a very highly stable voltage regulator to excite the load cells. Each component was researched thoroughly before purchasing to ensure high quality components were used in the system. An Arduino Mega 2560 is used to act as the Master device and to communicate with each section in turn. The data is then stored on the computer for further processing.

Each section is fitted with its own voltage regulator to power the electronics. The voltage regulator supports a wide range of voltages ranging 7 VDC to 36 VDC and converts this to 5 VDC to power all the electronics. This means the farmer can power the platform using a 12V or 24V battery when no power points are around to plug an adaptor into. The platform consumes a total of 10W when a 12V battery is used to power the platform and all the heating resistors are operating.

The analogue signals produced by the load cells are able to be successfully converted to a digital signal using the AD7193 and sent to the computer for further processing. Using MATLAB it is possible to display the data in real-time. If the data is recorded, it is possible to use the software written in Python to analyse the data in various ways such as plotting the data; calculating the stride length, the centre of pressure, hoof duration, and average weight; and rendering this data to a video file.

The demonstration platform developed proved to be very accurate. From the experiments the X-position accuracy was calculated to be  $1.005 \pm 2.172$  mm, and the Y-position accuracy was calculated to be  $0.814 \pm 1.788$  mm. The weight accuracy was calculated to be  $20.087 \pm 0.034$  kg. The demonstration platform had a mean weight error of 0.44% using a 20kg weight. The TruTest walk overweigh platforms has a weight error of about 1%, TruTest did not specify over what weight range. The demonstration platform was so sensitive that if you were to run your finger over the surface of the platform and look at the data in real-time on the computer, the same pattern will appear on the monitor. Essentially the platform could be thought of as an oversized touchpad.

The accuracy of one of the sections of the final platform was also tested and found to be very accurate. A laser cut jig was used to test the accuracy. The laser cut jig allowed for weights to be placed in known locations, the X-position accuracy was calculated to be  $-6.60 \pm 6.08$  mm, and the Y-position accuracy was calculated to be  $-5.06 \pm 8.47$  mm. The weight accuracy was calculated to be  $19.965 \pm 0.105$  kg.

Even though the software is able to calculate stride length and average weight, more trials are required to verify the accuracy of the algorithms and to verify the hypothesis that lame cows produce a distinct signal signature.

It is possible to capture and calculate the main variables to determine lameness (force, location and duration of each leg) using the current platform.

The prototype platform can be used for other applications such as:

- Exergaming – playing a game while physically having to move around (as seen in Section 4.1.3.3), achieved only using a single section.
- Rehabilitation of leg injuries.

- Sport and exercise research.
- Assistance for those with neuromuscular diseases.
- Creating interactive videos.
- Detection of lameness in horses or other animals.
- To monitor animal balance while they are being transported.

As there are currently only small snippets of data available in journal articles from other research projects and no raw data to be found anywhere, the captured data from the prototype platform captured in the milking shed can prove to be very useful for veterinarians and researchers interested in cattle behaviour.

A number of recommendations have been put forward for future improvements of the system. All my project aims were achieved; however, more testing will need to be done to verify the hypothesis that a lame cow produces a distinct signal signature compared to a healthy cow's signal signature. This report reflects the amount of knowledge and understanding I have gained while developing the "Hardware and Software Development towards Lameness Detection of Cattle".

## References

- [1] Ministry of Primary Industries, “Dairy,” Ministry of Primary Industries, March 2015. [Online]. Available: <https://www.mpi.govt.nz/exporting/food/dairy/>. [Accessed 20 April 2015].
- [2] Ministry of Primary Industries, “Dairy,” Ministry of Primary Industries, 19 May 2015. [Online]. Available: <http://archive.mpi.govt.nz/agriculture/pastoral/dairy>. [Accessed 20 May 2015].
- [3] T. Cronshaw, “Cow lameness costs farmers,” Stuff, 16 August 2014. [Online]. Available: <http://www.stuff.co.nz/business/farming/dairy/10387298/Cow-lameness-costs-farmers>. [Accessed 25 March 2015].
- [4] Stuff, “NZ’s dairy cattle population hits 6.6 million,” Stuff, 16 December 2013. [Online]. Available: <http://www.stuff.co.nz/business/farming/dairy/9522856/NZs-dairy-cattle-population-hits-6-6-million>. [Accessed 23 April 2015].
- [5] DairyNZ, “New Zealand Dairy Statistics 2013/2014,” 2015. [Online]. Available: <http://www.dairynz.co.nz/media/1327583/nz-dairy-statistics-2013-2014-web.pdf>. [Accessed 21 April 2015].
- [6] Zinpro, “Step-Up Locomatin Scoring System,” Zinpro, [Online]. Available: <http://www.zinpro.com/lameness/beef/locomotion-scoring>. [Accessed 16 March 2015].
- [7] AHDB Dairy, “Lameness,” AHDB Dairy, 2015. [Online]. Available: <http://dairy.ahdb.org.uk/technical-information/animal-health-welfare/lameness/>. [Accessed 30 April 2015].
- [8] T. Parkinson, J. Vermunt and J. and Malmo, Diseases of Cattle in Australasia, 2010.
- [9] M. Willem, V. Jürgen, B. Jeroen, J. Alexandru, M. Koen C., D. C. Sam, P. Arno, O. Geert, V. W. Stephanie and V. N. Annelies, “Development of a real time cow gait tracking and analysing tool to assess lameness using a pressure sensitive walkway: The GAITWISE system,” *ScienceDirect*, vol. 110, no. 1, pp. 29-39, 2011.
- [10] Boumatic, “StepMetrix,” Boumatic, 2015. [Online]. Available: <http://www.boumatic.com/eu-en/products/view/stepmetrix>. [Accessed 13 June 2014].
- [11] U. Tasch and P. Rajkondawar, “The Development of a SoftSeperator for a lameness diagnostic system,” *Science Direct*, vol. 44, pp. 239-245, 204.
- [12] Tru-Test, “Platforms,” Tru-Test, 2015. [Online]. Available: <http://livestock.tru-test.com/en-nz/platforms>. [Accessed 7 May 2015].
- [13] Tru-Test, “XRP2 Panel Reader and Antennas,” Tru-Test, 2015. [Online]. Available: <http://livestock.tru-test.com/en-nz/readers/xrp2-panel-reader>. [Accessed 7 May 2015].
- [14] A&D, “Definition of the “Load Cell”,” 2014. [Online]. Available: <http://www.aandd.jp/products/weighing/loadcell/introduction/pdf/1-1.pdf>. [Accessed 16 October 2014].
- [15] A&D, “What is the life-span of a load cell?,” A&D, 2015. [Online]. Available: [http://www.aandd.jp/products/weighing/loadcell/introduction/loadcells\\_qa\\_06.html](http://www.aandd.jp/products/weighing/loadcell/introduction/loadcells_qa_06.html). [Accessed 23 March 2015].

- [16] A&D, "Characteristics of the Strain Gauge Load Cell," A&D, 2015. [Online]. Available: <http://www.aandd.jp/products/weighing/loadcell/introduction/pdf/1-2.pdf>. [Accessed 09 June 2014].
- [17] A&D, "What kind of load cells exist?," A&D, 2015. [Online]. Available: [http://www.aandd.jp/products/weighing/loadcell/introduction/loadcells\\_qa\\_04.html](http://www.aandd.jp/products/weighing/loadcell/introduction/loadcells_qa_04.html). [Accessed 10 June 2014].
- [18] A&D, "Where are load cells used?," A&D, [Online]. Available: [http://www.aandd.jp/products/weighing/loadcell/introduction/loadcells\\_qa\\_02.html](http://www.aandd.jp/products/weighing/loadcell/introduction/loadcells_qa_02.html). [Accessed 12 June 2014].
- [19] EatSmart, "Bathroom Scales," EatSmart, [Online]. Available: <http://www.eatsmartproducts.com/products>. [Accessed 12 June 2014].
- [20] Nisbets, "Weightstation Electronic Platform Scale 3kg," Nisbets, 2014. [Online]. Available: <http://www.nisbets.co.uk/weightstation-electronic-platform-scale-3kg/F201/ProductDetail.raction>. [Accessed 12 June 2014].
- [21] Loadstar Sensors, "What is a Load Cell? How do Load Cells Work?," Loadstar Sensors, [Online]. Available: <http://www.loadstarsensors.com/what-is-a-load-cell.html>. [Accessed 13 June 2014].
- [22] A&D, "How does a load cell conduct measurements?," A&D, 2015. [Online]. Available: [http://www.aandd.jp/products/weighing/loadcell/introduction/loadcells\\_qa\\_05.html](http://www.aandd.jp/products/weighing/loadcell/introduction/loadcells_qa_05.html). [Accessed 12 June 2014].
- [23] Windmill Software, "Understanding RS-485 and RS-422," Windmill Software, [Online]. Available: <http://www.windmill.co.uk/rs485.html>. [Accessed 23 August 2014].
- [24] IDC-Online, "Industrial Data Communications - RS-232/RS-485," [Online]. Available: [https://www.idc-online.com/technical\\_references/pdfs/data\\_communications/tutorial\\_2.pdf](https://www.idc-online.com/technical_references/pdfs/data_communications/tutorial_2.pdf). [Accessed 24 August 2014].
- [25] Sparkfun, "Serial Peripheral Interface (SPI)," Sparkfun, [Online]. Available: <https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi#receiving-data>. [Accessed 16 June 2014].
- [26] Sparkfun, "I2C," Sparkfun, [Online]. Available: <https://learn.sparkfun.com/tutorials/i2c>. [Accessed 10 August 2014].
- [27] cornelam, "I2C between Arduino's," Instructables, [Online]. Available: <http://www.instructables.com/id/I2C-between-Arduinos/>. [Accessed 11 June 2014].
- [28] C. Murphy, "Moving Averages," Investopedia, 2014. [Online]. Available: <http://www.investopedia.com/university/movingaverage/movingaverages1.asp>. [Accessed 8 September 2014].
- [29] PT Global, "ASB-1000 1000kg - ASB Shearbeam," PT Global, [Online]. Available: <https://www.ptglobal.com/products/ABM1000A000XXX>. [Accessed 5 June 2014].
- [30] Texas Instruments, "REF5040 Low Noise, Very Low Drift, Precision Voltage Reference," Texas Instruments, 2011. [Online]. Available: <http://www.ti.com/product/ref5040>. [Accessed 2014 18 June].
- [31] Analog Devices, "Low Noise, Precision CMOS Amplifier," [Online]. Available: <http://www.analog.com/media/en/technical-documentation/data->



- sheets/AD8655\_8656.pdf. [Accessed 25 June 2014].
- [32] Analog Devices, “AD7193,” [Online]. Available: <http://www.analog.com/media/en/technical-documentation/data-sheets/AD7193.pdf>. [Accessed 26 May 2014].
- [33] M. Stephenson, “Lameness Detection for Cattle,” Massey University, Palmerston North, New Zealand, 2011.

## Appendices

### Appendix 1: Critical Component Datasheets

#### AD7193 Datasheet



## 4-Channel, 4.8 kHz, Ultralow Noise, 24-Bit Sigma-Delta ADC with PGA

Data Sheet

**AD7193**

### FEATURES

Fast settling filter option  
 4 differential/8 pseudo differential input channels  
 RMS noise: 11 nV @ 4.7 Hz (gain = 128)  
 15.5 noise-free bits @ 2.4 kHz (gain = 128)  
 Up to 22 noise-free bits (gain = 1)  
 Offset drift:  $\pm 5$  nV/ $^{\circ}$ C  
 Gain drift:  $\pm 1$  ppm/ $^{\circ}$ C  
 Specified drift over time  
 Automatic channel sequencer  
 Programmable gain (1 to 128)  
 Output data rate: 4.7 Hz to 4.8 kHz  
 Internal or external clock  
 Simultaneous 50 Hz/60 Hz rejection  
 4 general-purpose digital outputs  
 Power supply  
   AV<sub>DD</sub>: 3 V to 5.25 V  
   DV<sub>DD</sub>: 2.7 V to 5.25 V  
 Current: 4.65 mA  
 Temperature range:  $-40^{\circ}$ C to  $+105^{\circ}$ C  
 28-lead TSSOP and 32-lead LFCSP packages  
 Interface  
   3-wire serial  
   SPI, QSPI™, MICROWIRE™, and DSP compatible  
   Schmitt trigger on SCLK

### APPLICATIONS

PLC/DCS analog input modules  
 Data acquisition  
 Strain gage transducers

Pressure measurement  
 Temperature measurement  
 Flow measurement  
 Weigh scales  
 Chromatography  
 Medical and scientific instrumentation

### GENERAL DESCRIPTION

The AD7193 is a low noise, complete analog front end for high precision measurement applications. It contains a low noise, 24-bit sigma-delta ( $\Sigma$ - $\Delta$ ) analog-to-digital converter (ADC). The on-chip low noise gain stage means that signals of small amplitude can interface directly to the ADC.

The device can be configured to have four differential inputs or eight pseudo differential inputs. The on-chip channel sequencer allows several channels to be enabled simultaneously, and the AD7193 sequentially converts on each enabled channel, simplifying communication with the part. The on-chip 4.92 MHz clock can be used as the clock source to the ADC or, alternatively, an external clock or crystal can be used. The output data rate from the part can be varied from 4.7 Hz to 4.8 kHz.

The device has a very flexible digital filter, including a fast settling option. Variables such as output data rate and settling time are dependent on the option selected. The AD7193 also includes a zero latency option.

The part operates with a power supply from 3 V to 5.25 V. It consumes a current of 4.65 mA, and it is available in a 28-lead TSSOP package and a 32-lead LFCSP package.

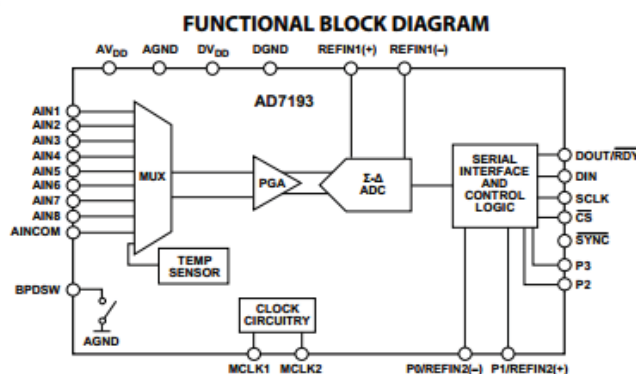


Figure 1.

The full datasheet for the AD7193 24-bit ADC can be found at:

<http://www.analog.com/media/en/technical-documentation/data-sheets/AD7193.pdf> accessed on 26 May 2014.

## REF5040 Datasheet



www.ti.com

REF5010, REF5020  
REF5025, REF5030  
REF5040, REF5045, REF5050

SBOS410F – JUNE 2007 – REVISED DECEMBER 2013

## Low-Noise, Very Low Drift, Precision Voltage Reference

Check for Samples: [REF5010](#), [REF5020](#), [REF5025](#), [REF5030](#), [REF5040](#), [REF5045](#), [REF5050](#)

### FEATURES

- **LOW TEMPERATURE DRIFT:**
  - High-Grade: 3ppm/°C (max)
  - Standard-Grade: 8ppm/°C (max)
- **HIGH ACCURACY:**
  - High-Grade: 0.05% (max)
  - Standard-Grade: 0.1% (max)
- **LOW NOISE:** 3μV<sub>pp</sub>/V
- **EXCELLENT LONG-TERM STABILITY:**
  - 45ppm/1000hr (typ) after 1000 hours
- **HIGH OUTPUT CURRENT:** ±10mA
- **TEMPERATURE RANGE:** –40°C to +125°C

### APPLICATIONS

- 16-BIT DATA ACQUISITION SYSTEMS
- ATE EQUIPMENT
- INDUSTRIAL PROCESS CONTROL
- MEDICAL INSTRUMENTATION
- OPTICAL CONTROL SYSTEMS
- PRECISION INSTRUMENTATION

### DESCRIPTION

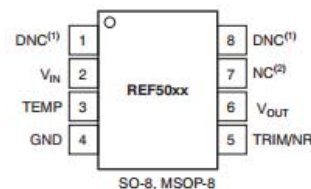
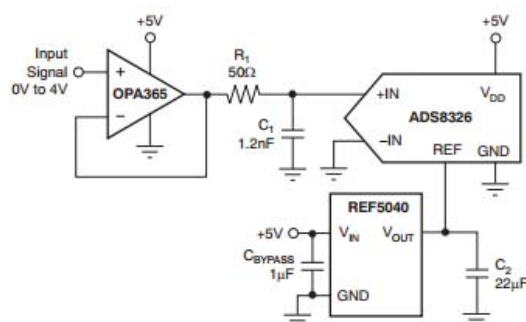
The REF50xx is a family of low-noise, low-drift, very high precision voltage references. These references are capable of both sinking and sourcing, and are very robust with regard to line and load changes.

Excellent temperature drift (3ppm/°C) and high accuracy (0.05%) are achieved using proprietary design techniques. These features, combined with very low noise, make the REF50xx family ideal for use in high-precision data acquisition systems.

Each reference voltage is available in both standard- and high-grade versions. They are offered in MSOP-8 and SO-8 packages, and are specified from –40°C to +125°C.

REF50xx Family

MODEL	OUTPUT VOLTAGE
REF5020	2.048V
REF5025	2.5V
REF5030	3.0V
REF5040	4.096V
REF5045	4.5V
REF5050	5.0V
REF5010	10.0V



NOTES: (1) DNC = Do not connect.  
(2) NC = No internal connection.

The full datasheet for the REF5040 voltage reference can be found at:  
<http://www.ti.com/lit/ds/sbos410f/sbos410f.pdf> accessed on 18 June 2014.



# Low Noise, Precision CMOS Amplifier

Data Sheet

**AD8655/AD8656****FEATURES**

**Low noise:** 2.7 nV/√Hz at  $f = 10$  kHz  
**Low offset voltage:** 250  $\mu$ V max over  $V_{CM}$   
**Offset voltage drift:** 0.4  $\mu$ V/°C typ and 2.3  $\mu$ V/°C max  
**Bandwidth:** 28 MHz  
**Rail-to-rail input/output**  
**Unity gain stable**  
**2.7 V to 5.5 V operation**  
**–40°C to +125°C operation**  
**Qualified for automotive applications**

**APPLICATIONS**

**ADC and DAC buffers**  
**Audio**  
**Industrial controls**  
**Precision filters**  
**Digital scales**  
**Automotive collision avoidance**  
**PLL filters**

**GENERAL DESCRIPTION**

The AD8655/AD8656 are the industry's lowest noise, precision CMOS amplifiers. They leverage the Analog Devices DigiTrim\* technology to achieve high dc accuracy.

The AD8655/AD8656 provide low noise (2.7 nV/√Hz at 10 kHz), low THD + N (0.0007%), and high precision performance (250  $\mu$ V max over  $V_{CM}$ ) to low voltage applications. The ability to swing rail-to-rail at the input and output enables designers to buffer analog-to-digital converters (ADCs) and other wide dynamic range devices in single-supply systems.

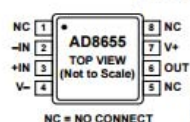
**PIN CONFIGURATIONS**

Figure 1. AD8655  
 8-Lead MSOP (RM-8)  
 8-Lead SOIC (R-8)

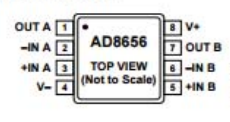


Figure 2. AD8656  
 8-Lead MSOP (RM-8)  
 8-Lead SOIC (R-8)

The high precision performance of the AD8655/AD8656 improves the resolution and dynamic range in low voltage applications. Audio applications, such as microphone pre-amps and audio mixing consoles, benefit from the low noise, low distortion, and high output current capability of the AD8655/AD8656 to reduce system level noise performance and maintain audio fidelity. The high precision and rail-to-rail input and output of the AD8655/AD8656 benefit data acquisition, process controls, and PLL filter applications.

The AD8655/AD8656 are fully specified over the –40°C to +125°C temperature range. The AD8655/AD8656 are available in Pb-free, 8-lead MSOP and SOIC packages. The AD8655/AD8656 are both available for automotive applications.

The full datasheet for the AD8656 amplifier can be found at:

[http://www.analog.com/media/en/technical-documentation/data-sheets/AD8655\\_8656.pdf](http://www.analog.com/media/en/technical-documentation/data-sheets/AD8655_8656.pdf)  
 accessed on 25 June 2014.





## MAX481/MAX483/MAX485/ MAX487-MAX491/MAX1487

### Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

#### General Description

The MAX481, MAX483, MAX485, MAX487-MAX491, and MAX1487 are low-power transceivers for RS-485 and RS-422 communication. Each part contains one driver and one receiver. The MAX483, MAX487, MAX488, and MAX489 feature reduced slew-rate drivers that minimize EMI and reduce reflections caused by improperly terminated cables, thus allowing error-free data transmission up to 250kbps. The driver slew rates of the MAX481, MAX485, MAX490, MAX491, and MAX1487 are not limited, allowing them to transmit up to 2.5Mbps.

These transceivers draw between 120 $\mu$ A and 500 $\mu$ A of supply current when unloaded or fully loaded with disabled drivers. Additionally, the MAX481, MAX483, and MAX487 have a low-current shutdown mode in which they consume only 0.1 $\mu$ A. All parts operate from a single 5V supply.

Drivers are short-circuit current limited and are protected against excessive power dissipation by thermal shutdown circuitry that places the driver outputs into a high-impedance state. The receiver input has a fail-safe feature that guarantees a logic-high output if the input is open circuit.

The MAX487 and MAX1487 feature quarter-unit-load receiver input impedance, allowing up to 128 MAX487/MAX1487 transceivers on the bus. Full-duplex communications are obtained using the MAX488-MAX491, while the MAX481, MAX483, MAX485, MAX487, and MAX1487 are designed for half-duplex applications.

#### Applications

Low-Power RS-485 Transceivers  
Low-Power RS-422 Transceivers  
Level Translators  
Transceivers for EMI-Sensitive Applications  
Industrial-Control Local Area Networks

#### Next Generation Device Features

- ◆ For Fault-Tolerant Applications  
MAX3430:  $\pm 80$ V Fault-Protected, Fail-Safe, 1/4 Unit Load, +3.3V, RS-485 Transceiver  
MAX3440E-MAX3444E:  $\pm 15$ kV ESD-Protected,  $\pm 60$ V Fault-Protected, 10Mbps, Fail-Safe, RS-485/J1708 Transceivers
- ◆ For Space-Constrained Applications  
MAX3460-MAX3464: +5V, Fail-Safe, 20Mbps, Profibus RS-485/RS-422 Transceivers  
MAX3362: +3.3V, High-Speed, RS-485/RS-422 Transceiver in a SOT23 Package  
MAX3280E-MAX3284E:  $\pm 15$ kV ESD-Protected, 52Mbps, +3V to +5.5V, SOT23, RS-485/RS-422, True Fail-Safe Receivers  
MAX3293/MAX3294/MAX3295: 20Mbps, +3.3V, SOT23, RS-485/RS-422 Transmitters
- ◆ For Multiple Transceiver Applications  
MAX3030E-MAX3033E:  $\pm 15$ kV ESD-Protected, +3.3V, Quad RS-422 Transmitters
- ◆ For Fail-Safe Applications  
MAX3080-MAX3089: Fail-Safe, High-Speed (10Mbps), Slew-Rate-Limited RS-485/RS-422 Transceivers
- ◆ For Low-Voltage Applications  
MAX3483E/MAX3485E/MAX3486E/MAX3488E/MAX3490E/MAX3491E: +3.3V Powered,  $\pm 15$ kV ESD-Protected, 12Mbps, Slew-Rate-Limited, True RS-485/RS-422 Transceivers

Ordering Information appears at end of data sheet.

#### Selection Table

PART NUMBER	HALF/FULL DUPLEX	DATA RATE (Mbps)	SLEW-RATE LIMITED	LOW-POWER SHUTDOWN	RECEIVER/DRIVER ENABLE	QUIESCENT CURRENT ( $\mu$ A)	NUMBER OF RECEIVERS ON BUS	PIN COUNT
MAX481	Half	2.5	No	Yes	Yes	300	32	8
MAX483	Half	0.25	Yes	Yes	Yes	120	32	8
MAX485	Half	2.5	No	No	Yes	300	32	8
MAX487	Half	0.25	Yes	Yes	Yes	120	128	8
MAX488	Full	0.25	Yes	No	No	120	32	8
MAX489	Full	0.25	Yes	No	Yes	120	32	14
MAX490	Full	2.5	No	No	No	300	32	8
MAX491	Full	2.5	No	No	Yes	300	32	14
MAX1487	Half	2.5	No	No	Yes	230	128	8

For pricing, delivery, and ordering information, please contact Maxim Direct at 1-888-629-4642, or visit Maxim Integrated's website at [www.maximintegrated.com](http://www.maximintegrated.com).

19-0122; Rev 10; 9/14

The full datasheet for the MAX487 RS-485 Transceiver can be found at: <http://datasheets.maximintegrated.com/en/ds/MAX1487-MAX491.pdf> accessed on 13 June 2014.



## ATmega48A/PA/88A/PA/168A/PA/328/P

### ATMEL 8-BIT MICROCONTROLLER WITH 4/8/16/32KBYTES IN-SYSTEM PROGRAMMABLE FLASH

#### DATASHEET

#### Features

- High Performance, Low Power Atmel® AVR® 8-Bit Microcontroller Family
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 4/8/16/32KBytes of In-System Self-Programmable Flash program memory
  - 256/512/512/1KBytes EEPROM
  - 512/1K/1K/2KBytes Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C<sup>(1)</sup>
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
- Atmel® QTouch® library support
  - Capacitive touch buttons, sliders and wheels
  - QTouch and QMatrix® acquisition
  - Up to 64 sense channels
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
    - Temperature Measurement
  - 6-channel 10-bit ADC in PDIP Package
    - Temperature Measurement
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Byte-oriented 2-wire Serial Interface (Philips I<sup>2</sup>C compatible)
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change

The full datasheet for the ATmega328 microcontroller can be found at:


[http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P\\_datasheet\\_Complete.pdf](http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf) accessed on 30 May 2014.



## ASB1000 Datasheet

## Model ASB

### LOW COST SHEARBEAM LOAD CELL



*A superb low cost weighing solution, yet surprisingly accurate with good long life features.*

*The ASB is a shearbeam load cell with M8 threads on the 125kg model and M12 threads on the balance of the range.*

*This is one of the most cost effective shearbeam cells available.*

*Robustly constructed in design with carefully selected high strength aircraft aluminium, it is an ideal selection for most weighing applications.*

*Used extensively in the agricultural industry it has proved to be a rugged long life load cell.*

*Marine grade anodised the ASB has an excellent protection rating of IP67 and is backed by a three-year warranty.*

*Many of the metric accessories throughout our accessory range can be used with this model.*

#### APPLICATIONS

- Lower cost weighing applications
- Agricultural and harsh environments
- All general industrial weighing
- Both tension and compression




#### FEATURES

- Marine grade anodised
- Robust construction
- Most competitively priced
- Wide capacity range 125kg ~ 1t
- Simple installation
- Suitable for most installation

#### Specifications

Note: All specifications are a maximum, as a % (±) of full load, unless otherwise stated.

<table style="width: 100%; border-collapse: collapse;"> <tr><td>Nominal Capacity</td><td>125kg ~ 1t</td></tr> <tr><td>Signal Output at Capacity</td><td>2mV/V ± 0.1%</td></tr> <tr><td>Linearity Error</td><td>&lt; 0.025% FSO</td></tr> <tr><td>Non-Repeatability</td><td>&lt; 0.010% FSO</td></tr> <tr><td>Combined Error</td><td>&lt; 0.030% FSO</td></tr> <tr><td>Hysteresis</td><td>&lt; 0.010% FSO</td></tr> <tr><td>Creep/Zero Return (30 mins)</td><td>&lt; 0.035% / 0.025% FSO</td></tr> <tr><td>Zero Balance</td><td>&lt; 3.000% Capacity</td></tr> <tr><td>Temperature Effect on Span/10°C</td><td>&lt; 0.015% FSO</td></tr> <tr><td>Temperature Effect on Zero/10°C</td><td>&lt; 0.020% Capacity</td></tr> <tr><td>Compensated Temperature Range</td><td>-10 ~ 40°C</td></tr> <tr><td>Operating Temperature Range</td><td>-30 ~ 70°C</td></tr> <tr><td>Service Load</td><td>100% of Rated Capacity</td></tr> </table>	Nominal Capacity	125kg ~ 1t	Signal Output at Capacity	2mV/V ± 0.1%	Linearity Error	< 0.025% FSO	Non-Repeatability	< 0.010% FSO	Combined Error	< 0.030% FSO	Hysteresis	< 0.010% FSO	Creep/Zero Return (30 mins)	< 0.035% / 0.025% FSO	Zero Balance	< 3.000% Capacity	Temperature Effect on Span/10°C	< 0.015% FSO	Temperature Effect on Zero/10°C	< 0.020% Capacity	Compensated Temperature Range	-10 ~ 40°C	Operating Temperature Range	-30 ~ 70°C	Service Load	100% of Rated Capacity	<table style="width: 100%; border-collapse: collapse;"> <tr><td>Safe Load</td><td>125% of Rated Capacity</td></tr> <tr><td>Ultimate Load</td><td>300% of Rated Capacity</td></tr> <tr><td>Input Resistance</td><td>410Ω Nominal</td></tr> <tr><td>Output Resistance</td><td>352Ω Nominal</td></tr> <tr><td>Insulation Resistance (bridge to ground)</td><td>&gt; 5000 MΩ at 100V DC</td></tr> <tr><td>Excitation Voltage (Recommended)</td><td>5 ~ 12V AC/DC</td></tr> <tr><td>Excitation Voltage (Maximum)</td><td>15V AC/DC</td></tr> <tr><td>Storage Temperature Range</td><td>-50 ~ 70°C</td></tr> <tr><td>Cable Type</td><td>4mm, Screened, PVC Sheath 4 Core x 0.09mm<sup>2</sup> (28 AWG)</td></tr> <tr><td>Cable Length</td><td>3 Metres</td></tr> <tr><td>Material</td><td>Aluminium</td></tr> <tr><td>Finish</td><td>Marine Anodised</td></tr> </table>	Safe Load	125% of Rated Capacity	Ultimate Load	300% of Rated Capacity	Input Resistance	410Ω Nominal	Output Resistance	352Ω Nominal	Insulation Resistance (bridge to ground)	> 5000 MΩ at 100V DC	Excitation Voltage (Recommended)	5 ~ 12V AC/DC	Excitation Voltage (Maximum)	15V AC/DC	Storage Temperature Range	-50 ~ 70°C	Cable Type	4mm, Screened, PVC Sheath 4 Core x 0.09mm <sup>2</sup> (28 AWG)	Cable Length	3 Metres	Material	Aluminium	Finish	Marine Anodised
Nominal Capacity	125kg ~ 1t																																																		
Signal Output at Capacity	2mV/V ± 0.1%																																																		
Linearity Error	< 0.025% FSO																																																		
Non-Repeatability	< 0.010% FSO																																																		
Combined Error	< 0.030% FSO																																																		
Hysteresis	< 0.010% FSO																																																		
Creep/Zero Return (30 mins)	< 0.035% / 0.025% FSO																																																		
Zero Balance	< 3.000% Capacity																																																		
Temperature Effect on Span/10°C	< 0.015% FSO																																																		
Temperature Effect on Zero/10°C	< 0.020% Capacity																																																		
Compensated Temperature Range	-10 ~ 40°C																																																		
Operating Temperature Range	-30 ~ 70°C																																																		
Service Load	100% of Rated Capacity																																																		
Safe Load	125% of Rated Capacity																																																		
Ultimate Load	300% of Rated Capacity																																																		
Input Resistance	410Ω Nominal																																																		
Output Resistance	352Ω Nominal																																																		
Insulation Resistance (bridge to ground)	> 5000 MΩ at 100V DC																																																		
Excitation Voltage (Recommended)	5 ~ 12V AC/DC																																																		
Excitation Voltage (Maximum)	15V AC/DC																																																		
Storage Temperature Range	-50 ~ 70°C																																																		
Cable Type	4mm, Screened, PVC Sheath 4 Core x 0.09mm <sup>2</sup> (28 AWG)																																																		
Cable Length	3 Metres																																																		
Material	Aluminium																																																		
Finish	Marine Anodised																																																		

Manufactured in New Zealand

PT Limited®

The full datasheet for the ASB1000 load cell can be found at:

[http://s3-ap-southeast-2.amazonaws.com/ptglobal-cdn/assets/54/WEB\\_ASB\\_804.pdf?AWSAccessKeyId=AKIAJEAX2FF3ZMX66G3Q&Expires=1433124212&Signature=HxAu84TZ7sm%2BB2r5Qg%2BaTX5byLY%3D](http://s3-ap-southeast-2.amazonaws.com/ptglobal-cdn/assets/54/WEB_ASB_804.pdf?AWSAccessKeyId=AKIAJEAX2FF3ZMX66G3Q&Expires=1433124212&Signature=HxAu84TZ7sm%2BB2r5Qg%2BaTX5byLY%3D) accessed on 15 May 2014.

## Appendix 2: Experimental Results

### Load Cell Calibration Experiment

1.998mV	
Weight (kg)	ADC Reading
0	11
1	2100
2	4250
5	10650
10	21050
20	42400
30	63700
40	84600
50	105800
Scale Factor	0.000472443

1.999mV	
Weight (kg)	ADC Reading
0	11
1	2100
2	4200
5	10600
10	21200
20	42500
30	63800
40	85250
50	106500
Scale Factor	0.000469303

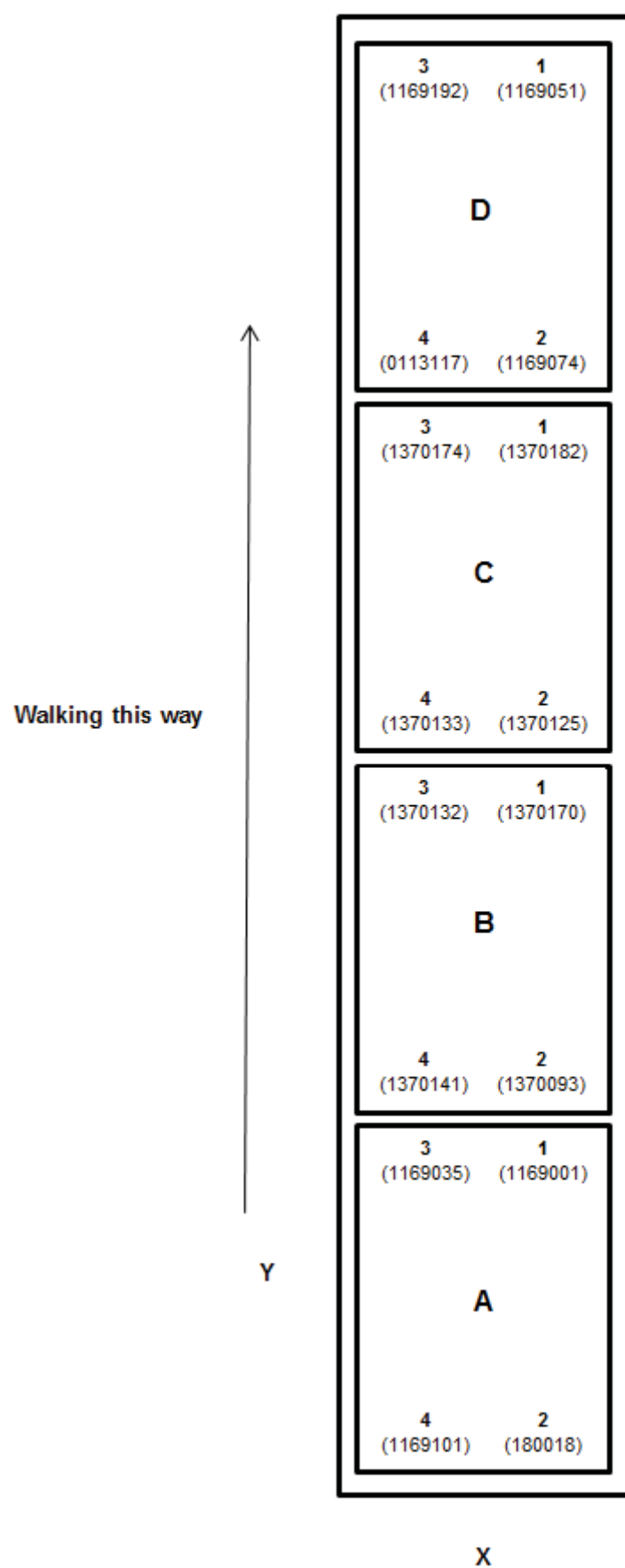
2.000mV	
Weight (kg)	ADC Reading
0	11
1	2100
2	4200
5	10600
10	21200
20	42750
30	64200
40	85700
50	106850
Scale Factor	0.000467150

2.001mV	
Weight (kg)	ADC Reading
0	11
1	2100
2	4200
5	10600
10	21200
20	42700
30	64200
40	85400
50	107000
Scale Factor	0.000467308

2.002mV	
Weight (kg)	ADC Reading
0	11.2705
1	2100
2	4200
5	10600
10	21400
20	43000
30	64500
40	85000
50	106500
Scale Factor	0.000469060

**Load Cell Serial Numbers and Positions**

Section	Loadcell	Serial Number	mV/V	Scaling Factor
A	1	1169001	1.998	0.0004724
	2	180018	2.001	0.0004673
	3	1169035	2.000	0.0004661
	4	1169101	1.998	0.0004724
B	1	1370170	2.000	0.0004661
	2	1370093	1.999	0.0004693
	3	1370132	2.001	0.0004673
	4	1370141	1.998	0.0004724
C	1	1370182	2.000	0.0004661
	2	1370125	2.000	0.0004661
	3	1370174	1.999	0.0004693
	4	1370133	1.999	0.0004693
D	1	1169051	2.002	0.0004691
	2	1169074	2.001	0.0004673
	3	1169192	2.000	0.0004661
	4	0113117	1.998	0.0004724



**Test Results from First Prototype Platform (Weight/Position)**

Position (mm)		Actual Position (mm)		Weight (kg)	Measured Weight (kg)	Difference in Weight (Kg)	Difference in X Position (mm)	Difference in Y Position (mm)
X	Y	X	Y					
0	0	1.972	0.275	20	20.053	0.053	1.972	0.275
50	0	51.585	1.091	20	20.025	0.025	1.585	1.091
100	0	103.636	0.650	20	20.024	0.024	3.636	0.650
150	0	153.848	1.952	20	20.053	0.053	3.848	1.952
200	0	201.532	1.756	20	20.052	0.052	1.532	1.756
212.5	0	211.928	1.027	20	20.045	0.045	-0.572	1.027
250	0	251.429	1.107	20	20.083	0.083	1.429	1.107
300	0	301.392	0.266	20	20.124	0.124	1.392	0.266
350	0	350.581	0.845	20	20.109	0.109	0.581	0.845
400	0	401.170	0.255	20	20.095	0.095	1.170	0.255
425	0	422.549	0.122	20	20.131	0.131	-2.451	0.122
0	50	1.647	53.154	20	20.101	0.101	1.647	3.154
50	50	53.328	52.933	20	20.059	0.059	3.328	2.933
100	50	103.120	53.265	20	20.026	0.026	3.120	3.265
150	50	153.191	51.910	20	20.069	0.069	3.191	1.910
200	50	203.960	52.342	20	20.078	0.078	3.960	2.342
250	50	252.293	53.067	20	20.077	0.077	2.293	3.067
300	50	300.651	53.098	20	20.031	0.031	0.651	3.098
350	50	350.993	52.030	20	20.029	0.029	0.993	2.030
400	50	398.994	53.436	20	20.118	0.118	-1.006	3.436
425	50	420.721	52.466	20	20.126	0.126	-4.279	2.466
0	100	3.332	102.792	20	20.112	0.112	3.332	2.792
100	100	102.306	103.512	20	20.105	0.105	2.306	3.512
200	100	200.498	103.259	20	20.134	0.134	0.498	3.259
300	100	297.897	100.571	20	20.073	0.073	-2.103	0.571
400	100	397.131	102.153	20	20.110	0.110	-2.869	2.153
425	100	422.017	103.445	20	20.093	0.093	-2.983	3.445
0	200	2.025	201.039	20	20.068	0.067	2.025	1.039
100	200	102.132	199.461	20	20.062	0.062	2.132	-0.540
200	200	201.658	203.174	20	20.082	0.082	1.658	3.174
300	200	299.556	202.553	20	20.084	0.084	-0.444	2.553
400	200	398.813	201.912	20	20.080	0.080	-1.187	1.912
425	200	421.803	201.772	20	20.063	0.063	-3.197	1.772
0	300	1.439	301.916	20	20.062	0.062	1.439	1.916
100	300	103.648	301.386	20	20.065	0.065	3.648	1.386
200	300	202.660	298.810	20	20.032	0.032	2.660	-1.190
212.5	300	212.887	301.265	20	20.058	0.058	0.387	1.265
300	300	302.460	298.500	20	20.090	0.090	2.460	-1.500
400	300	401.482	298.051	20	20.103	0.103	1.482	-1.950

425	300	424.841	299.358	20	20.068	0.068	-0.159	-0.642
0	400	5.377	400.723	20	20.126	0.126	5.377	0.723
100	400	104.308	400.390	20	20.102	0.102	4.308	0.390
200	400	202.051	399.015	20	20.097	0.097	2.051	-0.986
300	400	303.057	397.921	20	20.090	0.090	3.057	-2.079
400	400	401.651	398.019	20	20.149	0.149	1.651	-1.981
425	400	425.072	397.563	20	20.093	0.093	0.072	-2.437
0	500	4.392	502.123	20	20.146	0.146	4.392	2.123
100	500	104.034	500.571	20	20.136	0.136	4.034	0.571
200	500	199.232	500.446	20	20.106	0.105	-0.768	0.446
300	500	300.265	498.478	20	20.067	0.067	0.265	-1.522
400	500	398.350	497.744	20	20.109	0.109	-1.650	-2.256
425	500	423.086	498.367	20	20.089	0.089	-1.914	-1.633
0	600	1.731	599.670	20	20.180	0.180	1.731	-0.330
100	600	100.962	601.556	20	20.119	0.119	0.962	1.556
200	600	200.114	598.454	20	20.119	0.119	0.114	-1.546
212.5	600	212.353	600.309	20	20.093	0.093	-0.147	0.309
300	600	298.996	598.805	20	20.104	0.104	-1.005	-1.195
400	600	399.980	597.615	20	20.091	0.091	-0.020	-2.385
425	600	422.682	598.298	20	20.069	0.069	-2.318	-1.702



### Test Platform Positional and Weight Data

#### No Rubber Mat

Actual Position			Measured Position			Difference between Actual and Measured Position		
X (mm)	Y (mm)	Weight (kg)	X (mm)	Y (mm)	Weight (kg)	X (mm)	Y (mm)	Weight (kg)
0	0	20	-7	-1	20.234	-7	-1	0.234
111	0	20	101	-5	20.216	-10	-5	0.216
222	0	20	214	-1	20.135	-8	-1	0.135
333	0	20	327	-2	20.204	-6	-2	0.204
444	0	20	430	-2	20.165	-14	-2	0.165
0	287.5	20	4	287	20.054	4	-0.5	0.054
111	287.5	20	107	284	20.254	-4	-3.5	0.254
222	287.5	20	212	285	20.157	-10	-2.5	0.157
333	287.5	20	321	285	20.158	-12	-2.5	0.158
444	287.5	20	436	286	20.075	-8	-1.5	0.075
0	575	20	0	574	20.038	0	-1	0.038
111	575	20	117	577	20.153	6	2	0.153
222	575	20	212	572	20.165	-10	-3	0.165
333	575	20	328	574	20.152	-5	-1	0.152
444	575	20	429	576	20.242	-15	1	0.242

#### With Rubber Mat

Actual Position			Measured Position			Difference between Actual and Measured Position		
X (mm)	Y (mm)	Weight (kg)	X (mm)	Y (mm)	Weight (kg)	X (mm)	Y (mm)	Weight
0	0	20	-11	-5	19.945	-11	-5	-0.055
111	0	20	101	-6	19.982	-10	-6	-0.018
222	0	20	211	-10	20.056	-11	-10	0.056
333	0	20	323	-8	20.068	-10	-8	0.068
444	0	20	428	-9	19.925	-16	-9	-0.075
0	287.5	20	-3	288	19.892	-3	0.5	-0.108
111	287.5	20	104	286	20.053	-7	-1.5	0.053
222	287.5	20	211	285	20.025	-11	-2.5	0.025
333	287.5	20	328	282	20.089	-5	-5.5	0.089
444	287.5	20	432	288	19.982	-12	0.5	-0.018
0	575	20	18	580	20.086	18	5	0.086
111	575	20	114	583	19.942	3	8	-0.058
222	575	20	223	587	19.921	1	12	-0.079
333	575	20	334	585	19.752	1	10	-0.248
444	575	20	441	582	19.766	-3	7	-0.234

## Results from Validating Average Weight Algorithms

### Data Rate of 50Hz

Running Average (Method 1)	Section AB (kg)	Difference (kg)	Section BC (kg)	Difference (kg)	Section CD (kg)	Difference (kg)
Run 1	157.566	7.511	157.610	4.221	159.799	7.700
Run 2	159.045	6.032	159.421	2.410	157.298	10.201
Run 3	156.868	8.209	158.420	3.411	154.186	13.313
Run 4	158.377	6.700	160.319	1.512	154.228	13.271
Run 5	158.088	6.989	159.722	2.109	154.377	13.122
<b>Mean Deviation</b>	157.989	<b>7.088</b>	159.098	<b>2.733</b>	155.978	<b>11.521</b>
<b>Standard Deviation</b>	0.823	<b>0.823</b>	1.079	<b>1.079</b>	2.509	<b>2.509</b>
<b>Minimum</b>	156.868	<b>6.032</b>	157.610	<b>1.512</b>	154.186	<b>7.700</b>
<b>Maximum</b>	159.045	<b>8.209</b>	160.319	<b>4.221</b>	159.799	<b>13.313</b>
Running Average (Method 2)	Section AB (kg)	Difference (kg)	Section BC (kg)	Difference (kg)	Section CD (kg)	Difference (kg)
Run 1	154.174	10.903	153.347	8.484	157.803	9.696
Run 2	158.002	7.075	156.373	5.458	151.904	15.595
Run 3	157.551	7.526	158.559	3.272	152.469	15.030
Run 4	152.100	12.977	156.021	5.810	146.436	21.063
Run 5	155.678	9.399	158.663	3.168	153.762	13.737
<b>Mean Deviation</b>	155.501	<b>9.576</b>	156.593	<b>5.238</b>	152.475	<b>15.024</b>
<b>Standard Deviation</b>	2.440	<b>2.440</b>	2.183	<b>2.183</b>	4.088	<b>4.088</b>
<b>Minimum</b>	152.100	<b>7.075</b>	153.347	<b>3.168</b>	146.436	<b>9.696</b>
<b>Maximum</b>	158.002	<b>12.977</b>	158.663	<b>8.484</b>	157.803	<b>21.063</b>
Running Average (Method 3)	Section AB (kg)	Difference (kg)	Section BC (kg)	Difference (kg)	Section CD (kg)	Difference (kg)
Run 1	153.352	11.725	154.522	7.309	153.970	13.529
Run 2	153.701	11.376	152.544	9.287	150.938	16.561
Run 3	148.819	16.258	151.342	10.489	152.760	14.739
Run 4	154.543	10.534	157.274	4.557	160.472	7.027
Run 5	154.059	11.018	153.432	8.399	153.061	14.438
<b>Mean Deviation</b>	152.895	<b>12.182</b>	153.823	<b>8.008</b>	154.240	<b>13.259</b>
<b>Standard Deviation</b>	2.321	<b>2.321</b>	2.255	<b>2.255</b>	3.654	<b>3.654</b>
<b>Minimum</b>	148.819	<b>10.534</b>	151.342	<b>4.557</b>	150.938	<b>7.027</b>
<b>Maximum</b>	154.543	<b>16.258</b>	157.274	<b>10.489</b>	160.472	<b>16.561</b>
Weighted Average (Method 1)	Section AB (kg)	Difference (kg)	Section BC (kg)	Difference (kg)	Section CD (kg)	Difference (kg)
Run 1	159.084	5.993	159.220	2.611	162.683	4.816
Run 2	161.351	3.726	162.124	-0.293	160.378	7.121
Run 3	159.478	5.599	160.925	0.906	160.133	7.366

Run 4	160.167	4.910	162.730	-0.899	161.412	6.087
Run 5	160.290	4.787	162.151	-0.320	160.918	6.581
<b>Mean Deviation</b>	160.074	<b>5.003</b>	161.430	<b>0.401</b>	161.105	<b>6.394</b>
<b>Standard Deviation</b>	0.869	<b>0.869</b>	1.399	<b>1.399</b>	1.012	<b>1.012</b>
<b>Minimum</b>	159.084	<b>3.726</b>	159.220	<b>-0.899</b>	160.133	<b>4.816</b>
<b>Maximum</b>	161.351	<b>5.993</b>	162.730	<b>2.611</b>	162.683	<b>7.366</b>
Weighted Average (Method 2)	Section AB (kg)	Difference (kg)	Section BC (kg)	Difference (kg)	Section CD (kg)	Difference (kg)
Run 1	159.129	5.948	156.198	5.633	168.436	-6.605
Run 2	158.876	6.201	157.371	4.460	163.703	-1.872
Run 3	159.510	5.567	161.405	0.426	161.940	-0.109
Run 4	158.958	6.119	163.501	-1.670	164.416	-2.585
Run 5	162.545	2.532	167.120	-5.289	163.299	-1.468
<b>Mean Deviation</b>	159.804	<b>5.273</b>	161.119	<b>0.712</b>	164.359	<b>-2.528</b>
<b>Standard Deviation</b>	1.552	<b>1.552</b>	4.473	<b>4.473</b>	2.451	<b>2.451</b>
<b>Minimum</b>	158.876	<b>2.532</b>	156.198	<b>-5.289</b>	161.940	<b>-6.605</b>
<b>Maximum</b>	162.545	<b>6.201</b>	167.120	<b>5.633</b>	168.436	<b>-0.109</b>
Weighted Average (Method 3)	Section AB (kg)	Difference (kg)	Section BC (kg)	Difference (kg)	Section CD (kg)	Difference (kg)
Run 1	153.450	11.627	154.752	7.079	154.064	13.435
Run 2	153.464	11.613	152.222	9.609	150.625	16.874
Run 3	148.480	16.597	151.279	10.552	152.958	14.541
Run 4	154.426	10.651	157.211	4.620	160.637	6.862
Run 5	154.239	10.838	153.405	8.426	152.932	14.567
<b>Mean Deviation</b>	152.812	<b>12.265</b>	153.774	<b>8.057</b>	154.243	<b>13.256</b>
<b>Standard Deviation</b>	2.462	<b>2.462</b>	2.321	<b>2.321</b>	3.787	<b>3.787</b>
<b>Minimum</b>	148.480	<b>10.651</b>	151.279	<b>4.620</b>	150.625	<b>6.862</b>
<b>Maximum</b>	154.426	<b>16.597</b>	157.211	<b>10.552</b>	160.637	<b>16.874</b>

**Data Rate of 100Hz**

Running Average (Method 1)	Section AB (kg)	Difference (kg)	Section BC (kg)	Difference (kg)	Section CD (kg)	Difference (kg)
Run 1	156.500	12.717	157.610	5.039	159.421	10.090
Run 2	154.496	14.721	159.421	3.228	INVALID	
Run 3	160.065	9.152	158.420	4.229	160.433	9.078
Run 4	161.151	8.066	160.319	2.330	160.856	8.655
Run 5	154.304	14.913	159.722	2.927	160.766	8.745
<b>Mean Deviation</b>	157.303	<b>11.914</b>	159.098	<b>3.551</b>	160.369	<b>9.142</b>
<b>Standard Deviation</b>	3.160	<b>3.160</b>	1.079	<b>1.079</b>	0.658	<b>0.658</b>
<b>Minimum</b>	154.304	<b>8.066</b>	157.610	<b>2.330</b>	159.421	<b>8.655</b>
<b>Maximum</b>	161.151	<b>14.913</b>	160.319	<b>5.039</b>	160.856	<b>10.090</b>
Running Average	Section AB	Difference	Section BC	Difference	Section	Difference

(Method 2)	(kg)	(kg)	(kg)	(kg)	CD (kg)	(kg)
Run 1	156.508	12.709	153.017	9.632	153.441	16.070
Run 2	149.729	19.488	147.601	15.048	INVALID	
Run 3	160.777	8.440	158.136	4.513	153.379	16.132
Run 4	158.842	10.375	159.536	3.113	158.527	10.984
Run 5	157.318	11.899	156.209	6.440	152.656	16.855
<b>Mean Deviation</b>	156.635	<b>12.582</b>	154.900	<b>7.749</b>	154.501	<b>15.010</b>
<b>Standard Deviation</b>	4.189	<b>4.189</b>	4.757	<b>4.757</b>	2.708	<b>2.708</b>
<b>Minimum</b>	149.729	<b>8.440</b>	147.601	<b>3.113</b>	152.656	<b>10.984</b>
<b>Maximum</b>	160.777	<b>19.488</b>	159.536	<b>15.048</b>	158.527	<b>16.855</b>
Running Average (Method 3)	Section AB (kg)	Difference (kg)	Section BC (kg)	Difference (kg)	Section CD (kg)	Difference (kg)
Run 1	149.416	19.801	153.980	8.669	156.712	12.799
Run 2	154.226	14.991	154.525	8.124	163.703	5.808
Run 3	158.898	10.319	160.149	2.500	157.290	12.221
Run 4	155.295	13.922	154.304	8.345	153.238	16.273
Run 5	153.637	15.580	158.955	3.694	159.243	10.268
<b>Mean Deviation</b>	154.294	<b>14.923</b>	156.383	<b>6.266</b>	158.037	<b>11.474</b>
<b>Standard Deviation</b>	3.407	<b>3.407</b>	2.930	<b>2.930</b>	3.837	<b>3.837</b>
<b>Minimum</b>	149.416	<b>10.319</b>	153.980	<b>2.500</b>	153.238	<b>5.808</b>
<b>Maximum</b>	158.898	<b>19.801</b>	160.149	<b>8.669</b>	163.703	<b>16.273</b>
Weighted Average (Method 1)	Section AB (kg)	Difference (kg)	Section BC (kg)	Difference (kg)	Section CD (kg)	Difference (kg)
Run 1	158.896	10.321	159.968	2.681	163.501	6.010
Run 2	158.883	10.334	158.935	3.714	INVALID	
Run 3	164.163	5.054	164.941	-2.292	163.150	6.361
Run 4	162.606	6.611	163.407	-0.758	163.128	6.383
Run 5	161.418	7.799	162.952	-0.303	162.451	7.060
<b>Mean Deviation</b>	161.193	<b>8.024</b>	162.041	<b>0.608</b>	163.058	<b>6.454</b>
<b>Standard Deviation</b>	2.317	<b>2.317</b>	2.503	<b>2.503</b>	0.439	<b>0.439</b>
<b>Minimum</b>	158.883	<b>5.054</b>	158.935	<b>-2.292</b>	162.451	<b>6.010</b>
<b>Maximum</b>	164.163	<b>10.334</b>	164.941	<b>3.714</b>	163.501	<b>7.060</b>
Weighted Average (Method 2)	Section AB (kg)	Difference (kg)	Section BC (kg)	Difference (kg)	Section CD (kg)	Difference (kg)
Run 1	159.703	9.514	166.933	-4.284	166.571	-3.922
Run 2	163.866	5.351	160.021	2.628	INVALID	
Run 3	162.177	7.040	168.457	-5.808	164.119	-1.470
Run 4	160.621	8.596	160.476	2.173	159.020	3.629
Run 5	158.763	10.454	165.769	-3.120	160.580	2.069
<b>Mean Deviation</b>	161.026	<b>8.191</b>	164.331	<b>-1.682</b>	162.573	<b>0.076</b>
<b>Standard Deviation</b>	2.026	<b>2.026</b>	3.850	<b>3.850</b>	3.414	<b>3.414</b>
<b>Minimum</b>	158.763	<b>5.351</b>	160.021	<b>-5.808</b>	159.020	<b>-3.922</b>

<b>Maximum</b>	163.866	<b>10.454</b>	168.457	<b>2.628</b>	166.571	<b>3.629</b>
Weighted Average (Method 3)	Section AB (kg)	Difference (kg)	Section BC (kg)	Difference (kg)	Section CD (kg)	Difference (kg)
Run 1	149.288	19.929	153.983	8.666	156.880	12.631
Run 2	154.202	15.015	154.453	8.196	163.877	5.634
Run 3	158.929	10.288	160.097	2.552	157.289	12.222
Run 4	155.090	14.127	154.202	8.447	153.159	16.352
Run 5	153.568	15.649	159.007	3.642	159.367	10.144
<b>Mean Deviation</b>	154.215	<b>15.002</b>	156.348	<b>6.301</b>	158.114	<b>11.397</b>
<b>Standard Deviation</b>	3.453	<b>3.453</b>	2.954	<b>2.954</b>	3.922	<b>3.922</b>
<b>Minimum</b>	149.288	<b>10.288</b>	153.983	<b>2.552</b>	153.159	<b>5.634</b>
<b>Maximum</b>	158.929	<b>19.929</b>	160.097	<b>8.666</b>	163.877	<b>16.352</b>

### Appendix 3: Final PCB Schematic

