

SMARTWALKER – ROLLING WALKER INSTRUMENTATION AND DATA
ACQUISITION SYSTEM DEVELOPMENT TO MONITOR, VISUALIZE
AND STORE ROLLING WALKER USAGE DATA

by

MAURICIO JAGUAN NIEVES

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2016

Copyright © by Mauricio Jaguan Nieves 2016

All Rights Reserved



Acknowledgements

I would like to thank the entire MAE department at the University of Texas at Arlington that has prepared and supported me all throughout my academic career and has made possible the development of this work. I owe all the professors, TAs classmates and staff during my undergraduate and graduate education years the knowledge and skills gained during this period of time. I would also like to thank the Simon Bolivar University, institution that started this path properly and gave a strong base to continue my education in UTA.

I have to give special recognition to my supervising professor Dr. Haiying Huang who welcomed and gave me the opportunity to join the Advanced Sensor Technology Laboratory during my graduate studies. She guided me with wisdom, patience and understanding all throughout the development of this project and pushed me in the right direction when I needed it the most.

In the ASTL laboratory I had the fortune of sharing with and knowing many great engineers that made all the work performed more enjoyable and fruitful. Here I would like to mention soon to be Dr. Jun Yao, Jeremiah Sanders, Abhay Singh and Kranthi Balusu all great friends and helping hands during the time we shared together. They have given me their valuable input during the development of this work and helped elevating its overall quality.

Finally I have to thank my family that has supported me and is always present. My father that pushed me to find a future in UTA, my mother that always gave me an example of work ethic and the pursue of perfection in every new challenge. I am grateful of my grandparents that always gave me unconditional love and support, and my Uncle Abraham, person that made possible to seek graduate studies.

November 28, 2016

Abstract

SMARTWALKER – ROLLING WALKER INSTRUMENTATION AND DATA ACQUISITION SYSTEM DEVELOPMENT TO MONITOR, VISUALIZE AND STORE ROLLING WALKER USAGE DATA

Mauricio Jaguan Nieves, MS

The University of Texas at Arlington, 2016

Supervising Professor: Haiying Huang

The Smart Walker project was designed to fill a necessity of monitoring in real time the use of rolling walkers (RW) and study the causes that contribute to the high rate of fallings among its users. The main objectives of the project were to measure the major forces applied by RW users in real time and store it safely for further analysis. The first prototype of the Smart Walker includes the measurement of axial load, torque and gripping force applied on the handle as well as acceleration and rotation angles while it is being used. The axial load was measured using strain gages installed on each leg of the rolling walker allowing the measurement of the overall axial load and its distribution. The torque and gripping force applied to the handles were also measured using strain gage rosettes and Force Sensitive Resistors (FSR) respectively. To measure the acceleration and angles of rotation a 6 (Degrees of Freedom) DOF Inertial Measurement Unit (IMU) was implemented. The (Data Acquisition) DAQ system was developed using Arduino boards and Xbee antennas. Arduino boards offer a reliable and cost effective option for DAQ and were successfully implemented in this project. Furthermore, consistent and secure wireless transmission of data was required and achieved using Xbee antennas. A user interface (UI) was developed using LabVIEW that obtained the readings from the

Arduino board, showed the measurements graphically in real time and stored the data for further analysis. This work serves as a reference for strain gage measurement, low-cost DAQs, wireless transmission of data using Xbee antennas. It also covers the use of Finite Element Method (FEM) to assist in the design of strain gage systems and aims to close the gap between the Arduino and LabVIEW interaction. The work has been presented so readers can replicate each phase of the project and adapt it to their specific needs.

Table of Contents

Acknowledgements	iii
Abstract	iv
List of Figures.....	ix
List of Tables	xiv
Chapter 1 Introduction: Smart Walker Project Overview	15
Chapter 2 Rolling Walker Mechanical Model.....	18
Analytical Model.....	18
FEM Model	24
Chapter 3 Axial Load Measurement	29
Strain Gage Measurement Overview	29
Resistance Strain Gages.....	29
Wheatstone Bridge	32
Non-Inverting Amplifier	35
Analog to Digital Conversion	36
Amplifier Design Considerations	38
Estimation of Resolution of Overall System	38
True Gain Test.....	44
Calibration.....	45
Effect of Load Applied on Opposite Handle	49
Chapter 4 Torque Measurement.....	52
Strain Gage Measurement Overview for Torque Measurement	52
Basic Theory – Full Bridge	52
Circuit Design	56
True Gain Test.....	57

Calibration.....	58
Effect of Axial load on Torque Measurement	62
Chapter 5 Gripping Force Measurement	64
FSR Sensor Review	64
Circuit.....	66
Calibration.....	67
Chapter 6 IMU Accelerometer & Gyroscope.....	69
IMU 6DOF Board Review	69
Installation.....	70
Chapter 7 Data Acquisition, Wireless Transmission and User Interface	73
Hardware Overview	73
Arduino MEGA as DAQ.....	73
Resolution.....	76
Sampling Rate	76
XBee® 802.15.4 Antennas for Wireless Data Transmission	77
User Interface & LabVIEW/Arduino Connectivity	80
Arduino Code.....	80
User Interface	87
Initialization Tab	87
Data Visualization Tab	89
LabVIEW Code.....	91
A – Calibration Reference Voltage.....	93
Initial Visa read	95
File Path.....	99
Extract Integers from array (Axial Load).....	100

Extract Integers from Array (Torque & FSR)	100
B – Data Acquisition from Walker sensors & overall Data Storage	101
Torque and Grip VISA to Indicators.....	102
Axial Load VISA to Indicators	103
Walker IMU VISA to User Indicators	105
C – Data Acquisition & Storage from Belt Sensors (Time Stamps).....	106
Belt IMU VISA to indicators	107
Data Visualization.....	109
Chapter 8 Conclusions.....	114
Appendix A Walker FEM Study	117
Geometry and Model Setup.....	118
Mesh	119
Results.....	121
Appendix B Smart Walker Arduino Code	123
Appendix C Axial Load Calibration Data Analysis Mathcad Worksheet.....	129
Appendix D Torque Calibration Data Analysis Mathcad Worksheet	136
Appendix E Smart Walker PCB Arduino Shield (Initial Design)	147
Appendix F Miscellaneous Pictures and Drawings	150
References.....	155
Biographical Information	158

List of Figures

Figure 1-1 Smart Walker 3-D model showing location and placement of sensors.....	17
Figure 2-1 Walker CAD model (a) and simplified geometry model (b)	19
Figure 2-2 Normalized shear and moment diagrams of the Rolling Walker's simplified analytical models	21
Figure 2-3 Maximum total strain results from analytical model of the walker	23
Figure 2-4 FEM analysis result performed in ANSYS (a) vs. actual walker with installed strain gage under the point of maximum strain (b).....	26
Figure 2-5 Total Strain vs. Applied Load for Analytical and FEM model	27
Figure 3-1 Construction of a typical metallic foil strain gauge [2]	31
Figure 3-2 Indicated Strain vs. peak strain measured by Strain Gage [4]	31
Figure 3-3 MR-Series Bridge Completion Modules [9]	32
Figure 3-4 Wheatstone bridge (quarter bridge) [2]	33
Figure 3-5 Differential shunt balance arrangement for balancing bridge circuits [2]	33
Figure 3-6 Non-Inverting amplifier schematic	35
Figure 3-7 Binary quantification and saturation [2]	37
Figure 3-8 Estimated bridge deflection from analytical and FE model	38
Figure 3-9 Estimated voltage response of the system with amplifiers of different gains to applied axial load	40
Figure 3-10 Non-Inverting op-amp schematic (a) and Texas Instruments PDIP-14 package (b).....	43
Figure 3-11 Fabricated Non-Inverting operational amplifier	43
Figure 3-12 True gain test results for axial load sensor amplifier. Channel 1 (a), 2 (b), 3 (c), 4 (d).	44
Figure 3-13 Application of the load for axial load calibration	45

Figure 3-14 Experimental setup for axial load calibration.....	46
Figure 3-15 Axial load Calibration Flow Diagram	47
Figure 3-16 Axial load sensors calibration curves. Corresponding to legs (a) Front/Left, (b) Front/Right, (c) Back/Left, (d) Back/Right	48
Figure 3-17 Effect on the left side legs when load is applied on right handle. Front Leg(a), Back Leg (b).....	49
Figure 3-18 Effect on the right side legs when load is applied on left handle. Front Leg(a), Back Leg (b).....	50
Figure 4-1 Strain gage configuration for torque measurement. [3].....	53
Figure 4-2 General Purpose Strain Gages - Shear/Torque Pattern [5]	53
Figure 4-3 Installed Rosettes on left handle	54
Figure 4-4: Expected Full-Bridge output (Torque)	55
Figure 4-5 Zeroing circuit for torque sensor in main circuit board	56
Figure 4-6 Fabricated Amplifier used in torque sensor (G=1000)	56
Figure 4-7 Circuit schematic of torque sensor	57
Figure 4-8 Torque amplifier true gain test results. Channels 1(a), 2(b), 3(c), 4(d)	57
Figure 4-9 Zeroing circuit & Amplifier true gain test.....	58
Figure 4-10 Clamp 3D printed used to apply torque for calibration	59
Figure 4-11 Experimental setup of torque calibration	59
Figure 4-12 Calibration curve of torque sensor in left handle	60
Figure 4-13 Expected response from analytical model.....	60
Figure 4-14 Calibration curve of torque sensor in right handle.....	61
Figure 4-15 Left handle torque sensor's response due to axial load	62
Figure 4-16 Right handle torque sensor's response due to axial load	63

Figure 5-1 ForceSensitive Resistor used to measure gripping force [20].....	64
Figure 5-2 FSR Resistance vs. Force curve [20].....	65
Figure 5-3 FSR Adafruit installation tutorial for Arduino [20]	66
Figure 5-4 FSR circuit schematic (a) - Fabricated Circuit (b)	67
Figure 5-5 Block diagram of Calibration setup.....	67
Figure 5-6 (a) Calibration setup showing FSR and alignment beams & (b) setup while force is applied to the FSR sensing area.	68
Figure 5-7 FSR Calibration curves.....	68
Figure 6-1 6DOF IMU Board [12].....	69
Figure 6-2 Connection of IMU card to Arduino (UNO) board [12], and block diagram of connection to PC	71
Figure 6-3 Front Panel of VI to acquiring data from IMU and plotting accelerometer data	71
Figure 7-1 Arduino MEGA 2560 [10].....	73
Figure 7-2 XBee [®] 802.15.4 antennas [26]	78
Figure 7-3 SparkFun Xbee shield [15]	79
Figure 7-4 Xbee Antenna + SparkFun Shield + Arduino board assembly [15].....	79
Figure 7-5 SparkFun Xbee Explorer Dongle [15].....	79
Figure 7-6 Smart Walker User Interface (“Initialize” Tab)	89
Figure 7-7 Smart Walker User Interface (“DAQ” Tab)	90
Figure 7-8 Main Smart Walker LabVIEW block diagram	92
Figure 7-9 Smart Walker main code (A: Calibration Reference Voltage)	93
Figure 7-10 Initial VISA read SubVI (Initialize)	95
Figure 7-11 Visa Read SubVI	96
Figure 7-12 Wait for Bytes subVI	97

Figure 7-13 String to Array of Integers SubVI.....	98
Figure 7-14 File Path and Data file title creation subVI	99
Figure 7-15 Extract Integers from array of integers (Axial load)	100
Figure 7-16 Extract Integers from array of Integers (Torque and FSR).....	100
Figure 7-17 Smart Walker main code (B – Data Acquisition from Walker sensors & overall Data Storage)	101
Figure 7-18 Torque and FSR VISA readings to UI Indicators subVI	102
Figure 7-19 Torque and FSR calibration curves subVI.....	103
Figure 7-20 Axial load and VISA readings to UI Indicators subVI	103
Figure 7-21 Axial Load Calibration Curves subVI.....	104
Figure 7-22 Average and percentage calculations for Axial load UI Indicators	105
Figure 7-23 Walker IMU readings to clusters for waveform chart subVI	105
Figure 7-24 Smart Walker main code (C – Data Acquisition & Storage from Belt Sensors (Time Stamps).....	106
Figure 7-25 Belt Arduino VISA read & cluster to waveform charts.....	107
Figure 7-26 String array from integers to be stored in .txt file.....	108
Figure 7-27 Stored data from data acquisition (.txt file).....	109
Figure 7-28 Axial Load Readings.....	111
Figure 7-29 Torque and Gripping Force Readings	112
Figure 7-30 IMU Readings from the walker	113
Figure A-1 CAD model, pressure applied and supports	118
Figure A-2 First Mesh.....	119
Figure A-3 Second Mesh [5mm element]	120
Figure A-4 Third Mesh [3mm element]	120
Figure A-5 Equivalent Strain (close-up 3rd Mesh)	121

Figure A-6 Total Deformation [mm] (3 rd Mesh)	121
Figure C-1 Data files format to be imported to the worksheet	130
Figure D-1 Data files format to be imported to the worksheet	137
Figure E-1 Initial Smart Walker Arduino shield design	148
Figure E-2 Smart Walker Arduino Shield first design (Stencil)	149
Figure E-3 Smart Walker Arduino Shield initial design (upper side).....	149
Figure E-4 Smart Walker Arduino Shield initial design (lower side)	149
Figure F-1 Smart Walker (a) CAD model (b) 1 st prototype	151
Figure F-2 First PCB prototype fabricated to test axial load and torque strain gages	151
Figure F-3 Smart Walker Xbee antenna on USB dongle.....	152
Figure F-4 Belt Arduino UNO with IMU and Xbee shield/antenna.....	152
Figure F-5 Smart Walker Circuit board - 1st prototype (Arduino Side).....	153
Figure F-6 Smart Walker Circuit board - 1st prototype (circuit Side).....	153
Figure F-7 Smart Walker 1st prototype circuit board case mod CAD model.....	154
Figure F-8 Smart Walker 1st prototype circuit board mounted on bottom case side.	154

List of Tables

Table 3-1 DAQ system resolution depending on N-bit of A/D converter and reference voltage in [mV/bit count]	40
Table 3-2 Estimated sensitivities and system resolution for different amplifier gains and reference voltages with 10-bit A/D converter	40
Table 3-3 Operational Amplifier LT1014CN Specifications	42
Table 3-4 Summary of results from true gain test of axial load sensor amplifiers	45
Table 3-5 Sensitivity and overall resolution of measurement	47
Table 3-6 Slope of calibration curves and overall effect on sensor readings when the load is being applied on the opposite handle	50
Table 4-1 Components of Torque sensor's circuit	56
Table 4-2 Summary of linear fit coefficients.....	61
Table 5-1 FSR sensor technical specifications [20].....	65
Table 7-1 Arduino MEGA 2560 Technical Specifications [10].....	75
Table 7-2 XBee® 802.15.4 Technical Specifications [27]	78
Table 7-3 Arduino Code for Smart Walker (Analog read, IMU card read, Xbee connectivity)	80
Table 7-4 Comparison between Decimal and Hexadecimal Strings	86
Table 7-5 Analog Voltage output string format & reading example	86
Table 7-6 Output string format & reading example of 6DOF IMU card readings.....	86
Table 7-7 Acquired data regarding Axial load measurement.....	109
Table 7-8 Acquired data regarding Torque and Gripping Force measurement.....	110
Table 7-9 Acquired data regarding IMU readings from Walker	110
Table 7-10 Acquired data regarding IMU readings from Belt	111
Table A-1 Result Summary & percent change for convergence	122

Chapter 1

Introduction: Smart Walker Project Overview

This thesis describes the development and characterization of the first prototype of a smart walker. The Smart Walker project envisioned the design and fabrication of a device to monitor the use of rolling walkers and study the high rate of falling of elderly patients [1]. The rate of falling is nearly 40% among the rolling walker users. The risk of falls when using rolling walkers is greatly associated with incorrect RW height, inappropriate user posture, changes in gait patterns and poor maintenance of RW tips (wheel). Currently there are no instruments or devices that allow clinicians to monitor how the RW is used for daily mobility. The objective of the Smart Walker project was to measure, store, transmit and visualize the orientation, inclination and main forces applied to a conventional RW in real time during daily mobility usage. The measurements would be carried out in two local senior communities [1]. The data obtained during the studies will serve as a base for recommendations regarding the aforementioned factors that increase the risk of falls.

The forces to be measured by this prototype are the axial load, torque and gripping force applied on the handles. Strain gages were used to measure the axial load and torque. The gripping force is measured by a force sensitive resistor (FSR). The orientation and inclination is measured with a six degree-of-freedom (DOF) Inertial Measurement Unit (IMU) board containing an accelerometer (ADXL345) and a gyroscope (ITG-3200).

The readings from each sensor are acquired by the Arduino MEGA, an open source I/O board based on the ATmega1280 [10]. This board has 16 analog inputs that allow the strain gage and FSR measurements to be sampled with a 10 bit analog to digital converter. It also includes 54 digital input/output pins with the possibility of having

I²C connections that allow the integration of the accelerometer and the gyro. Other important specialized digital pins are available for serial connections. Serial connections permit the incorporation of Xbee antennas, which are responsible for transmitting the data acquired by the Arduino and receiving it at the serial port of a personal computer (PC). Xbee antennas communicate wirelessly using the IEEE 802.15.4 protocol for fast and secure peer-to-peer networking. Continuous, secure and uninterrupted transmission of data was achieved with an open field range of 100 m and indoor range of 30 m [24].

The data is received by a PC through serial connection and it is interpreted, visualized and stored using LabVIEW. There are open source add-ons for LabVIEW such as LIFA (LabVIEW interface for Arduino) and LINX that allow interacting with Arduino directly from LabVIEW without the need to program the Arduino board [17]. These platforms could not be used due to particular needs of the project. These platforms are created around the Arduino UNO, which differs from the MEGA in various aspects but more importantly, in the amount of analog and digital inputs allowed to be used. These platforms limit the number of analog inputs to only six (the ones available in the Arduino UNO) and the Smart walker requires eight analog inputs. Arduino serial communication with the Xbee antenna is also a challenge using the add-ons. Furthermore, the maximum sampling rate achieved with the LINX add-on is less than the one achieved by programming the Arduino board. This is because the LINX firmware includes functions to control most of the Arduino board's capabilities, using memory space that a program written to perform only the needed tasks would not use.

Due to the reasons presented above and also to have more control over the whole system, an Arduino and a LabVIEW program were written to acquire, interpret, visualize and store readings sent by the Arduino through the serial port with the Xbee antenna.

The aforementioned prototype allows for continuous data acquisition and real time visualization and storage of the measurands (axial load, torque, gripping force, acceleration and walker rotation). In addition to the sensors installed on the walker, an IMU board was installed in another Arduino board to be worn by the user during the usage of the RW. This extra IMU allows having more information regarding the posture of the patient.

The next steps in the project are (1) to fabricate three more smart walkers using similar circuitry and sensors and (2) to perform reliability analysis of the walkers to finally perform tests with patients. As part of the efforts for the next steps, a Printed Circuit Board (PCB) Arduino shield was being designed for faster reproduction in a smaller size of the circuit, the progress is shown in the appendix.

The figure below shows the Smart Walker model with the sensors used and installation location. Strain gages are placed on each leg, torque rosettes and FSRs on each handle and the IMU as part of the circuit board containing the necessary circuits and components for the data acquisition and wireless transmission.

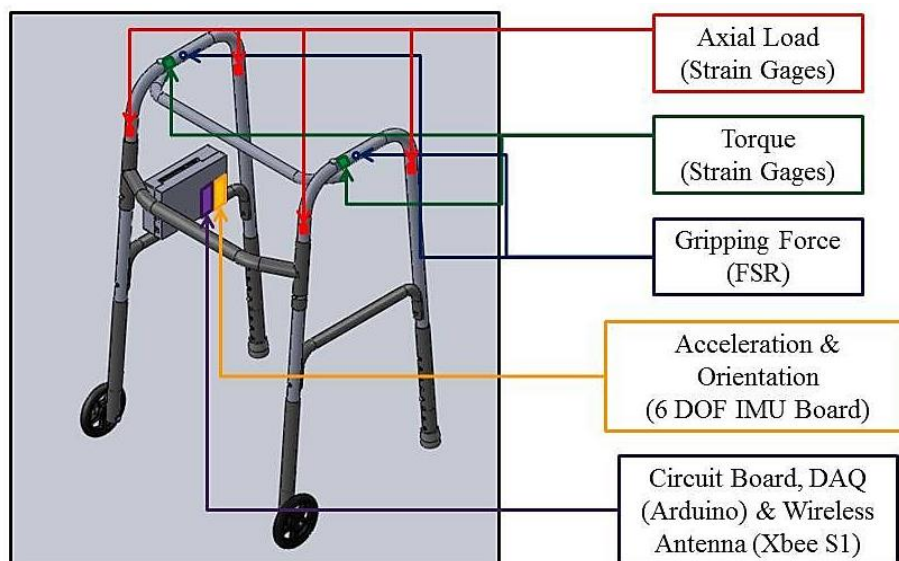


Figure 1-1 Smart Walker 3-D model showing location and placement of sensors

Chapter 2

Rolling Walker Mechanical Model

The first efforts taken to predict the Rolling Walker's (RW) strain at the legs to an applied axial load on the handles was to develop a mechanical model of the walker. Predicting the strain caused by an applied axial load, allows designing properly the axial load sensors for the required resolution of the system. Chapter 3 will describe in more detail the considerations taken to design the axial load sensor using the strain predictions in this chapter. First, a simplified analytical model of the main RW's structural members was developed in order to estimate the maximum strain felt at the curvature formed between the legs and the handle when an axial load is applied. Later, a CAD model of the walker was made in order to perform a Finite Element Method (FEM) analysis of the RW. The FEM analysis allowed having a more detailed and accurate prediction of the strain at the curvature, and helped to determine the location of the strain gages for axial load measurement on the legs.

It is required to have a voltage output from the axial load sensor of sufficient magnitude to monitor the force applied by the user. The strain caused by the loads is used to predict the resistance change of the strain gage, thus predicting the voltage output from the Wheatstone bridge and the necessary amplification to have a measurement resolution of at least 1 lb/mV.

Analytical Model

A side view of the CAD model of the walker is shown in Figure 2-1 (a). The simplified geometry used in the analytical model is also shown in Figure 2-1 (b) for comparison. The analytical model consists of vertical elements (legs), and a horizontal element (handle). The legs are set fixed replicating the calibration conditions. The overall strain is calculated at the extremes of the horizontal member (handle) fixed at both ends

as an approximation of the actual model. The moment is not being calculated on the vertical members of the simplified model (legs), but rather at the handle, where the points of interest are B and C shown in Figure 2-1 (b). These points would represent the points of maximum strain at the curvature joining the handle and the legs. Results will show that this assumption yields an accurate prediction of the maximum strain encountered.

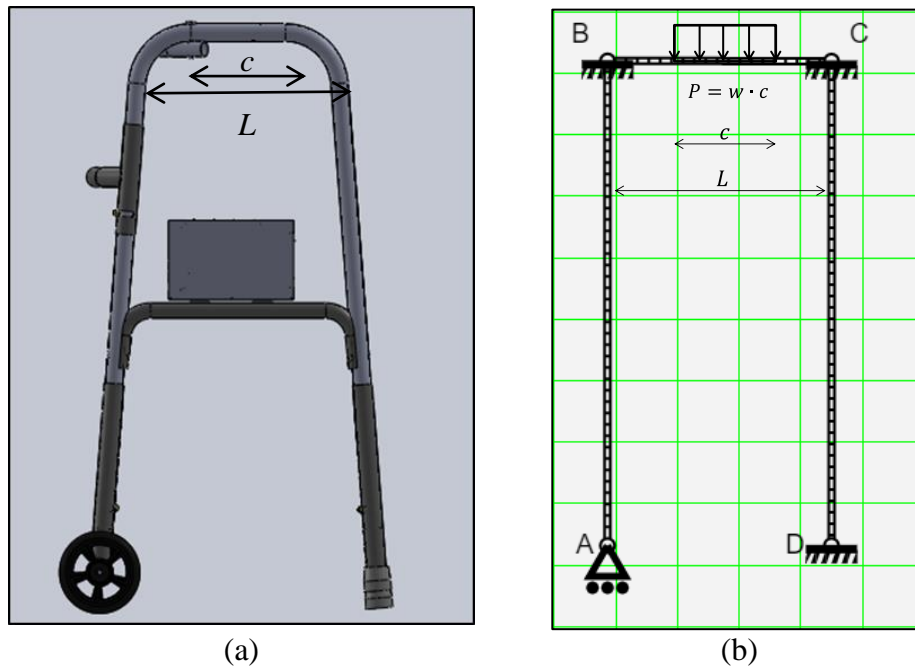


Figure 2-1 Walker CAD model (a) and simplified geometry model (b)

In Figure 2-2 the simplified model and resulting normalized shear and moment diagrams of the described analytical model are shown. The shear and moment equations along the length of a beam fixed at both ends with a uniform distributed load can be found in [32]. Eq. 2-1 to Eq. 2-4 show the general equations found in [32]. They have been simplified for the current case where the distributed load is centered along the length of the beam.

The reaction forces for the distribute load model, as shown in Figure 2-2 are given by Eq. 2-1.

$$R_A = R_B = \frac{w \cdot c}{2}. \quad \text{Eq. 2-1}$$

The reaction moments for the distributed load model are given by Eq. 2-2.

$$M_A = M_B = \frac{w \cdot c}{24L} (3L^2 - c^2). \quad \text{Eq. 2-2}$$

The shear equation is given by Eq. 2-3.

$$V(x) = \begin{cases} R_A & \text{if } x \leq \left(\frac{L-c}{2}\right) \\ R_A - w \left(x - \frac{(L-c)}{2}\right) & \text{if } \left(\frac{L-c}{2}\right) < x < \left(\frac{L+c}{2}\right) \\ -R_B & \text{if } x \geq \left(\frac{L+c}{2}\right). \end{cases} \quad \text{Eq. 2-3}$$

The bending moment equation is given by Eq. 2-4

$$M(x) = \begin{cases} -M_A + R_A x & \text{if } x \leq \left(\frac{L-c}{2}\right) \\ -M_A + R_A x - \frac{w}{2} (x - a)^2 & \text{if } \left(\frac{L-c}{2}\right) < x < \left(\frac{L+c}{2}\right) \\ -M_A + R_A x - wc \left(x - L/2\right) & \text{if } x \geq \left(\frac{L+c}{2}\right). \end{cases} \quad \text{Eq. 2-4}$$

A simpler model can be used. Instead of a distributed load a concentrated force P is applied at the center of the beam (handle). The equations for this model are shown from Eq. 2-5 to Eq. 2-8 [32] and their respective shear and moment diagrams are shown in Figure 2-2 as well. The reaction forces in this case are given by Eq. 2-5.

$$R_A = R_B = \frac{P}{2}. \quad \text{Eq. 2-5}$$

The reaction moments are given by Eq. 2-6.

$$M_A = M_B = \frac{PL}{8}. \quad \text{Eq. 2-6}$$

The shear and moment equations for this model are Eq. 2-7 and Eq. 2-8. Notice that the shear curve is being forced to pass through zero at $L/2$ as if the concentrated load was a very narrow distributed load.

$$V(x) = \begin{cases} R_A & \text{if } x < \left(\frac{L}{2}\right) \\ 0 & \text{if } x = \left(\frac{L}{2}\right) \\ -R_B & \text{if } x > \left(\frac{L}{2}\right) \end{cases} \quad \text{Eq. 2-7}$$

$$M(x) = \begin{cases} -M_A + R_A x & \text{if } x \leq \left(\frac{L}{2}\right) \\ -M_A + R_A (L - x) & \text{if } x > \left(\frac{L}{2}\right) \end{cases} \quad \text{Eq. 2-8}$$

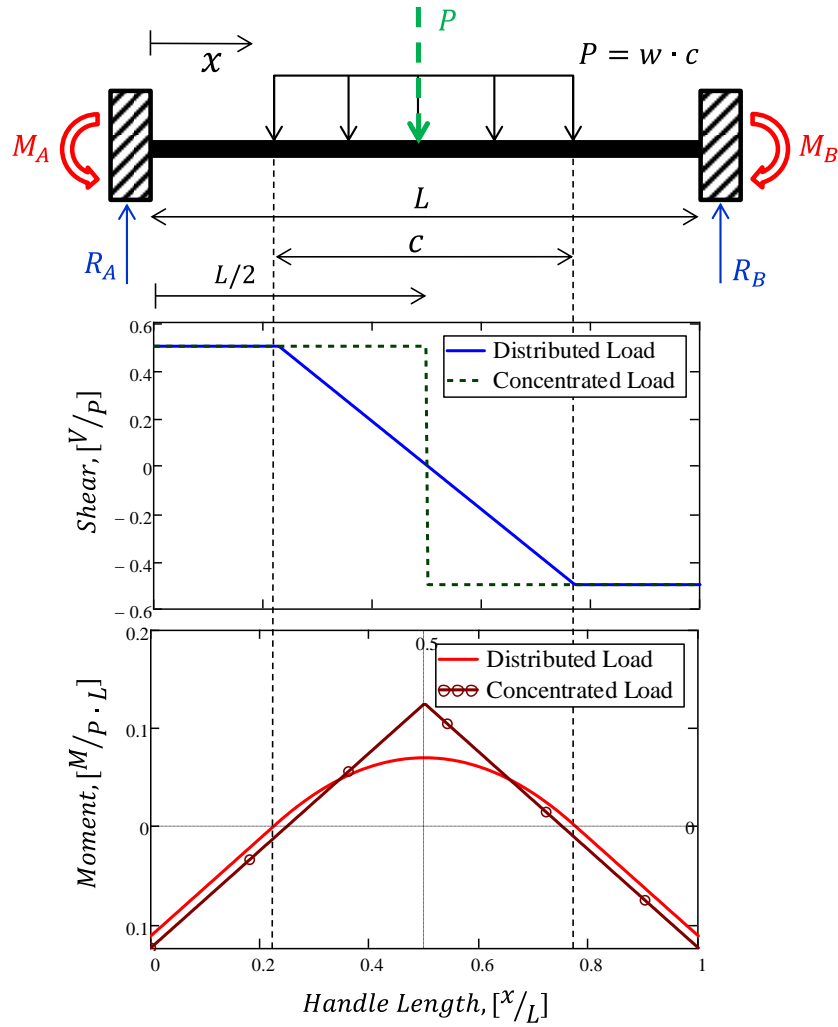


Figure 2-2 Normalized shear and moment diagrams of the Rolling Walker's simplified analytical models

Notice that the reaction forces for both concentrated and distributed load models are the same and the reaction moments are very close to each other. The model with a distributed load has reaction moments of about 90% the magnitude of the one with a centered concentrated force.

In the analytical calculations, the right and left sides of the walker are treated as if they were not connected to each other. It will be shown that this assumption is valid at this point because almost all of the strain on each leg is caused by the force applied to the handle of the respective leg. Later, the effect of axial loads applied on the opposite side is quantified experimentally in order to have a more reliable measurement. This simplified analytical model is expected to quantify the maximum strain at the curvature between the handles and legs of the walker. In the results shown in Figure 2-2, the length of the centered distributed load c is 15 cm and the total beam length L is 27.7 cm.

Axial stress on the legs is calculated using Eq. 2-9.

$$\sigma_{axial}(x) = \frac{V(x)}{A}. \quad \text{Eq. 2-9}$$

The hollow circular cross sectional area A is 1.042 cm^2 with an inner radius of 1.137 cm and outer radius of 1.275 cm.

The Axial strain is obtained using Eq. 2-10.

$$\epsilon_{axial} = \frac{\sigma_{axial}}{E_{al}}. \quad \text{Eq. 2-10}$$

The axial stress is given by σ_{axial} and the aluminum's modulus of elasticity (79.9 GPa) is E_{al} .

The Bending strain at the curvature is obtained as shown in Eq. 2-11,

$$\epsilon_{bend} = \frac{\sigma_{bend}}{E_{al}}. \quad \text{Eq. 2-11}$$

The Bending stress σ_{bend} is given in Eq. 2-12.

$$\sigma_{bend}(x) = \frac{-MR}{I_z}. \quad \text{Eq. 2-12}$$

M is the moment of the horizontal member shown in Figure 2-2, and R is the outer radius of the circular cross section.

The second moment of inertia about the neutral axil is given in Eq. 2-13 and has a value of $7.6\text{e-}9 \text{ m}^4$ using the outer and inner radii mentioned above.

$$I_z = \frac{\pi(R^4 - r^4)}{4}. \quad \text{Eq. 2-13}$$

The total strain is obtained adding both axial and bending components and has been plotted vs. applied loads in Figure 2-3. As expected from the moment diagram, the strain is higher for the concentrated load model. These two models are compared to the FEM model in the following section, where a distributed load was applied at the handles of the roller walker.

$$\varepsilon_{total} = \varepsilon_{axial} + \varepsilon_{bend}. \quad \text{Eq. 2-14}$$

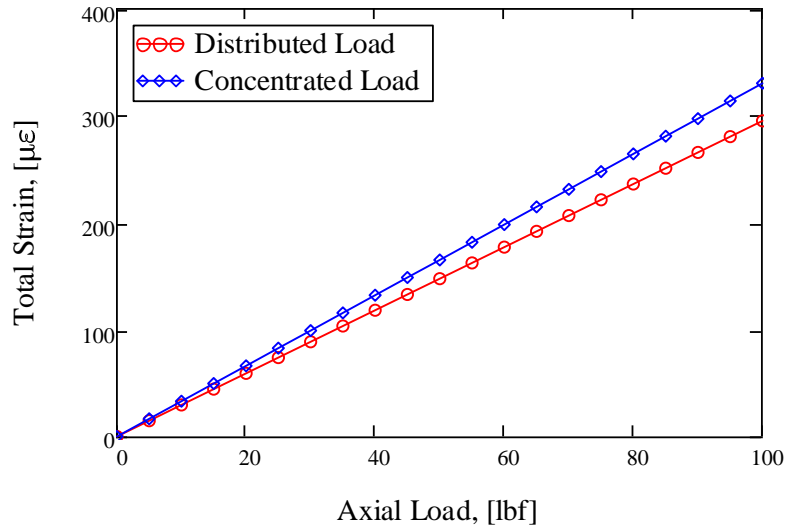


Figure 2-3 Maximum total strain results from analytical model of the walker

FEM Model

A CAD model of the walker was constructed to make a Finite Element Method (FEM) model using ANSYS [23] and predict the strain at the curvature of the leg of the walker when an axial load is applied at the handle. This approach was taken to observe the results on a more realistic geometry than the simplified geometry used in the analytical calculations. This model allows having a representation of the strain distribution in the curvature area that is closer to the real case to help assessing the placement of the strain gages. It is noticeable in Figure 2-4 that the strain is largest at the curvature between the vertical and horizontal members that form each side of the walker's structure. The resultant strain in this area has components of both axial and bending forces. From these results the location where the strain gages were to be installed was determined so that a significant strain would be felt by the strain gages, also allowing for easier and stronger bonding of the gage to the surface than at the point of maximum strain [4][7][8]. Contrarily to the analytical model, the strain at the curvature does not occur at a single point between the vertical and horizontal member but rather is distributed around the curvature area as shown in Figure 2-4.

It is necessary to perform experiments to know the true behavior of the system. FEM can be very accurate but it is limited by the used assumptions and approximations in the model; experimental results are not expected to be exactly the same as results from the FEM analysis for three main reasons:

1. The strain gage tends to integrate, or average, the strain over the area covered by the grid (Figure 3-2) [4]. Knowing that there is strain gradient at the placement area, the measured strain will not be equal to the strain recorded at a specific point from the FEM model.
2. The CAD model is a close but not an exact representation of the walker.

3. When installing the gages it is very difficult to place them exactly at the same area as the one in the FEM model.

The FEM model was set similar to the analytical model described above. The surfaces in contact with the ground were fixed to replicate closely the calibration conditions. Also, axial distributed loads on the handle were applied rather than concentrated loads. Moreover, all the joints in the model were set as bonded. Since computational power was not of big concern when running the analysis the geometry was not simplified. The FEM analysis was performed using three-dimensional tetrahedral elements with the help of the meshing tools provided by ANSYS. This type of elements is useful for stress analysis of general three-dimensional bodies that require a more precise analysis than is possible through two dimensional and/or axisymmetric analysis [31], although in some cases a 2D or a 1D models can be more accurate than a 3D models. In this case a 3D model was used because it was easier to mesh all the parts and connectors joining all the member of the walker. A 2D model would have been a valid alternative as well.

The model was subjected to a convergence analysis. Three different mesh sizes were used for convergence study and they are shown in Appendix A. The finest mesh used had a size of 3 mm and experienced a variation of 6.6% in maximum strain and 1.5% in total deformation from the previous one which had varied 14.6% and 16.3% in maximum strain and deformation respectively. The model was one element thick since the wall thickness is about 1 mm.

The results from the FEM model were taken at two points close to the grid area where the strain gages were chosen to be placed (the strain gage chosen has a 5mm grid) and these results were averaged.

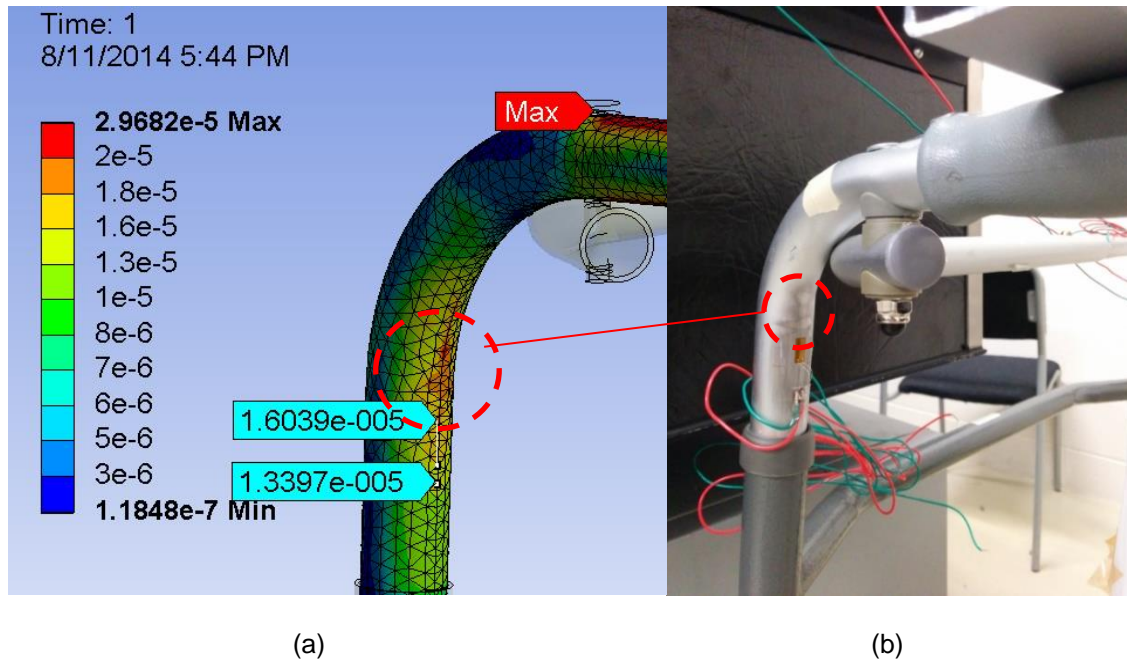


Figure 2-4 FEM analysis result performed in ANSYS (a) vs. actual walker with installed strain gage under the point of maximum strain (b)

It is interesting to notice in Figure 2-4 that the general shape of the strain distribution at the curvature between the leg and handle of the walker in the result from the FEM model coincide with the shades seen on the surface of the tube after the surface preparation process for gage installation on the walker [8]. Figure 2-5 shows the results from the FEM analysis and how it compares to the results from the analytical model.

The maximum total strain estimated in the analytical models and the one found in the FEM analysis are very close as expected. Moreover, the model with the distributed load approaches the FEM results very closely. The calculated strains with this analytical model are 99.6% the maximum strains found in the FEM model. On the other hand, the analytical model with the concentrated load overestimated the maximum strain by 11.3% compared to the FEM model results.

Most of the strain felt at the extremes of the horizontal member corresponds to bending strain. If only the axial component of the strain is measured, the resultant change in the resistance of the strain gage would be very small, resulting in a small voltage response from the bridge, thus a small signal-to-noise-ratio. A larger amplification of the signal implies amplifying the noise as well. If the signal-to-noise-ratio is small the resulting amplified signal would have high noise also. In order to have a larger signal-to-noise-ratio, the strain gage is installed at the curvature formed by the handle and legs where the magnitude of the strain is higher (includes both axial and bending components of the strain due to the applied axial load). The response would still follow a linear relation with load and it would be more easily quantifiable. The downside is that if the resultant load is applied at different locations of the handle, the moment felt at the curvature changes. This means that to maintain a reliable measurement, the location at which the handle is held has to be fixed. The assumption taken, as shown, is that a uniform distributed load is applied on the area of the handle.

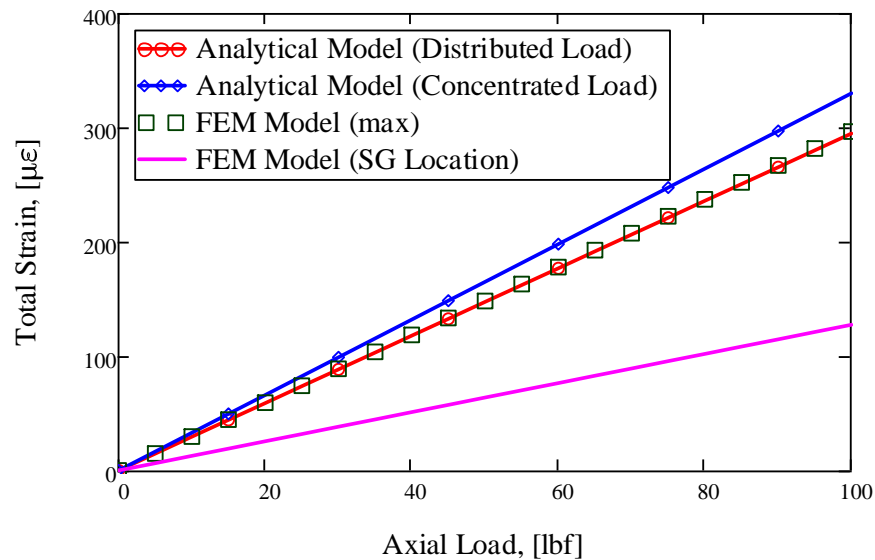


Figure 2-5 Total Strain vs. Applied Load for Analytical and FEM model

The strain gages were chosen to be installed at a location below the location of maximum strain at curvature as shown in Figure 2-4. In this area the response would still be bigger than only measuring axial strain but it would sense a lesser bending component. This location would also help on the strain gage installation because the surface is less curved and easier to prepare for installation. More details of the FEM model can be found in Appendix A.

Chapter 3

Axial Load Measurement

Strain Gage Measurement Overview

A brief overview of the main concepts needed to design the axial load sensor using strain gages is given in this section. A fundamental theoretical overview on resistance strain gages is presented first, followed by a basic depiction of Wheatstone bridges, non-inverting operational amplifier and finally main analog to digital conversion concepts necessary to understand the data acquisition system. The understanding of these concepts is essential for the measurement and data acquisition of strain using resistance strain gages.

Very often the measurement of physical quantities consists in the transduction of the phenomena to an electric voltage and then the measurement of this voltage which is correlated back to the physical phenomena in question. In the case of strain measurement, the strain causes a change in resistance in the gage which then causes an unbalance in the Wheatstone bridge, resulting in a proportional voltage output [2]. The voltage output of the bridge is amplified to be measured by a Multimeter or acquired using a data acquisition system. In order to properly design each component of the system, it is necessary to understand their basic principles behind them. These basic principles are described below.

Resistance Strain Gages

Bonded resistance strain gages have been widely used in the past decades to measure strain. They meet basic necessary characteristics [2] [4]:

1. High spatial resolution
2. Low sensitivity to changes in ambient conditions
3. High frequency response for dynamic strain measurement

They consist of a metallic grid specially designed to be securely bonded on the surface where the strain is to be measured. The strain gage is bonded using a special adhesive that ensures that the gage deforms with the test object. This adhesive varies according to the material and ambient effects at which the measurement is performed. The phenomenon that allows the use of resistance strain gages for this purpose is the change in electrical resistance of metallic and semiconductor materials when subjected to strain. This change in electrical resistance is what is being measured and correlated to the applied strain [2].

To depict how strain gages work let us consider a conductor with uniform cross-sectional area A_c and length L made of a material with an electrical resistivity ρ_c . The resistance of this conductor would be

$$R = \frac{\rho_c L}{A_c}. \quad \text{Eq. 3-1}$$

If the conductor is subjected to a normal load, both the length and the area change, which causes a variation in the total resistance of the conductor. This change in geometry is what allows us to measure the strain of the object. When the resistance of an object changes due to applied mechanical strain, it is known to have called piezoelectric behavior [2].

The variables mentioned in Eq. 3-1 can change significantly with temperature. If temperature fluctuations are not accounted for in the measurement, it can affect the reliability of the measurement. That is why many strain gages have temperature compensation, which allows for minimizing the effect of temperature on the measurement [2] [4]. Temperature doesn't represent a major factor for the Smart Walker because the measurements are thought to be carried out around room temperature; nevertheless the strain gages used have temperature compensation for aluminum (material of the walker).

Figure 3-1 shows a typical strain gauge construction [2]. It has a metallic pattern sandwiched in-between a plastic backing material. Choosing the right pattern dimensions is fundamental to ensure a proper measurement. The strain gage averages the measured strain over the grid area. Many times the maximum strain is the quantity of interest. Having a long gauge length in high strain gradient locations can result in error due to averaging. Figure 3-2 shows a representation of the averaging within the area of the gage.

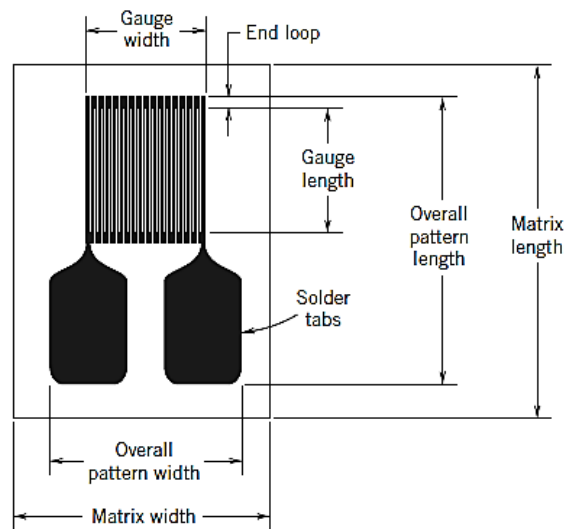


Figure 3-1 Construction of a typical metallic foil strain gauge [2]

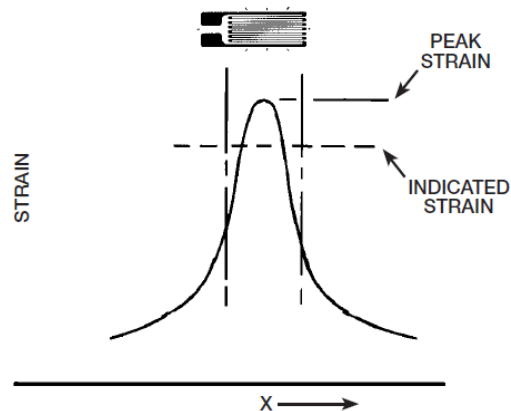


Figure 3-2 Indicated Strain vs. peak strain measured by Strain Gage [4]

The change in resistance is expressed in terms of a parameter called the gauge factor (GF) [2]. This is usually given by the strain gage manufacturer and it is defined as,

$$GF = \frac{\delta R / R}{\delta L / L} = \frac{\delta R / R}{\epsilon_a} \quad \text{Eq. 3-2}$$

Wheatstone Bridge

The Wheatstone bridge is one of the most common circuits used to detect small changes in resistance. It is widely used for strain gage measurement. Equipment is commercially available for this purpose, which can measure changes in resistance of less than $0.0005 \, \Omega$. In this case a bridge completion module MR1-350-127, shown in Figure 3-3, has been used because of its small size and simple installation.

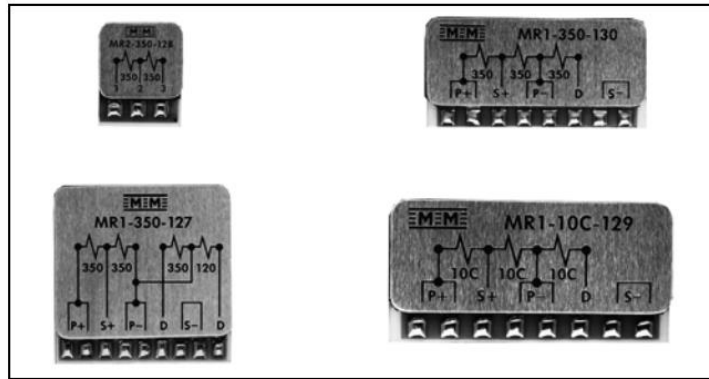


Figure 3-3 MR-Series Bridge Completion Modules [9]

The Wheatstone bridge is fed with an excitation voltage, shown in Figure 3-4 as E_i and the output voltage varies according to the change in resistance of the strain gage.

Figure 3-4 shows a quarter-bridge configuration. In this configuration only one of the four resistances in the bridge varies and the others remain fixed. As such, the change in resistance of only one of the resistors (i.e. the strain gage) is measured. In many situations, such as measuring bending strain, the change in resistance of two resistances is desired in order to obtain a higher response from the bridge. The full bridge

configuration will be shown later for measuring torque, in which all of the resistors in the bridge change and contribute to allow for a larger voltage output and ultimately better response to an applied torque.

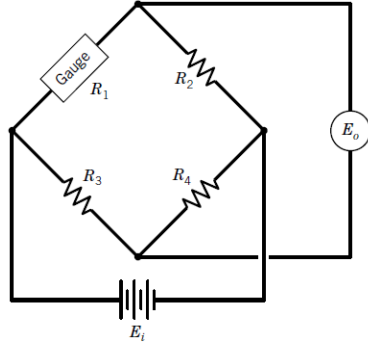


Figure 3-4 Wheatstone bridge
(quarter bridge) [2]

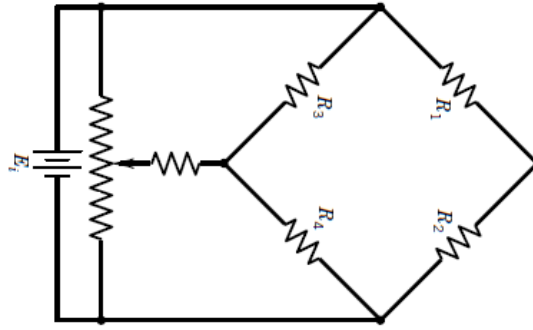


Figure 3-5 Differential shunt balance
arrangement for balancing bridge circuits [2]

$$E_o + \delta E_o = E_i \frac{(R_1 + \delta R) R_4 - R_3 R_2}{(R_1 + \delta R + R_2)(R_3 + R_4)} \quad \text{Eq. 3-3}$$

Eq. 3-3 shows the expected bridge voltage output for a quarter bridge configuration. It displays the initial voltage output with no bridge deflection (E_o), the voltage change (δE_o) due to the change in resistance of the strain gage (δR) as well as its relation to the other fixed resistances ($R_{2,3,4}$).

Often the bridge needs to be balanced in order to ensure its best response to the change in the resistance of the gage and to fine-tune the output voltage at initial conditions. One main reason to change the output voltage at initial conditions is to make sure that the amplifier following the bridge can sense the bridge deflection. That means that the bridge output at initial conditions, E_o be bigger than the input offset voltage of the operational amplifier used. Potentiometers are often used for this purpose. A balancing scheme is shown in Figure 3-5.

$$E_o + \delta E_o = E_i \frac{(R_1 + \delta R)R_4 - R_3R_2}{(R_1 + \delta R + R_2)(R_3 + R_4)} \quad \text{Eq. 3-3}$$

Eq. 3-3 and Eq. 3-2 reduce to Eq. 3-4 under the assumption that $\delta R/R \ll 1$,

$$\frac{\delta E_o}{E_i} = \frac{GF\varepsilon}{4 + 2GF\varepsilon} \approx \frac{GF\varepsilon}{4} \quad \text{Eq. 3-4}$$

For a multiple gauge bridge arrangement, the bridge constant κ is used to predict the bridge response. The bridge constant is defined as the ratio of the actual bridge output to the output that would result from a single gage sensing the maximum strain. When more than one gage is used, Eq. 3-4 becomes

$$\frac{\delta E_o}{E_i} = \frac{\kappa \delta R / R}{4 + 2\delta R / R} = \frac{\kappa GF\varepsilon}{4 + 2GF\varepsilon} \approx \frac{\kappa GF\varepsilon}{4} \quad \text{Eq. 3-5}$$

More detailed derivations of this equation can be found in Figliola [2].

Non-Inverting Amplifier

The output signal of the bridge is amplified using a low noise amplifier. In this case a non-inverting amplifier was used. Figure 3-6 shows the schematic of the non-inverting amplifier with connections to voltage source and bridge outputs. It is necessary to ensure that there is a DC path to earth for the very small input current that is needed, for that reason R_3 is included. R_4 is included for impedance matching purposes at the input of the amplifier [28].

The gain of the amplifier is given by,

$$G_A = 1 + \frac{R_1}{R_2} . \quad \text{Eq. 3-6}$$

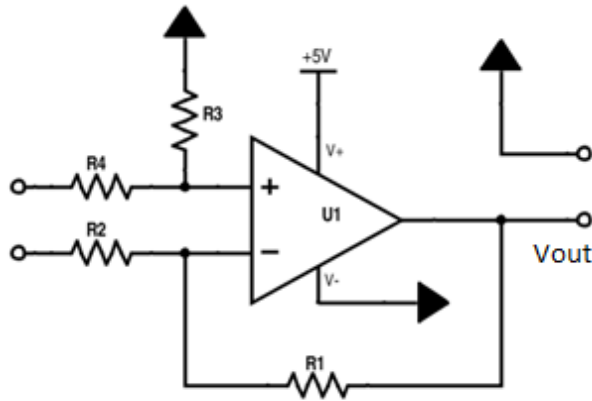


Figure 3-6 Non-Inverting amplifier schematic

Eq. 3-5 becomes Eq. 3-7 when the gain from the non-inverting amplifier is included Eq. 3-6,

$$\frac{\delta E_o}{E_i} = \frac{G_A \kappa (\delta R / R)}{4 + 2(\delta R / R)} = \frac{G_A \kappa G F \varepsilon}{4 + 2 G F \varepsilon} \approx \frac{G_A \kappa G F \varepsilon}{4} . \quad \text{Eq. 3-7}$$

Analog to Digital Conversion

The final goals are to acquire the voltage in real time using a data acquisition system, and to store and visualize the data. This requires an analog to digital (A/D) conversion of the signal. To obtain the best possible measurement from the system, the upper and lower voltage limits of the A/D converter need to be known.

An analog to digital converter discretizes a specific range of voltages according to its resolution and converts the analog signal to a digital quantity; this is called *quantization* [2]. To optimize the resolution of the acquired data from a given system, it is important to use as much as possible of the voltage range used by the A/D converter.

Converters in general have a reference voltage V_{ref} that defines the upper voltage limit of the measurement. The lower limit is usually ground. If the input signal is below the lower limit the reading is null (lowest binary number) and if it goes beyond the upper limit, the result is a saturated output (highest binary number). The amount of divisions possible in the measurement is given by the number of bits the A/D registers. An M-bit A/D converter outputs 2^M binary numbers. For example a 10 bit analog to digital converter with a V_{ref} of 5 V would be able to discretize the 5 V range with $2^{10} = 1024$ binary values; this means a resolution of 4.89 mV per binary value.

If the maximum output voltage from the amplifier is known, the reference voltage can be adjusted to get the maximum possible resolution from the given converter. Or, if a specific reference voltage is known to be used, the signal can be amplified accordingly to closely cover the range of conversion.

Another factor to take into account is the sampling rate of the converter. A/D converters sample the signal periodically. From a continuous signal, a discretized signal in time is obtained. Depending on the application and physical measurement to be performed, the required sampling rate varies. For example, when mechanical vibrations

are to be measured, it is fundamental to ensure that the maximum natural frequency of a given structure is being acquired by the system. The Nyquist criterion states that the minimum sampling frequency needed to capture the information from an oscillating analog signal is twice the frequency of such signal. For example, when sampling sound it is common to find sampling frequencies in the order of 40 kHz, which is about twice the upper limit of frequencies range of the human ear.

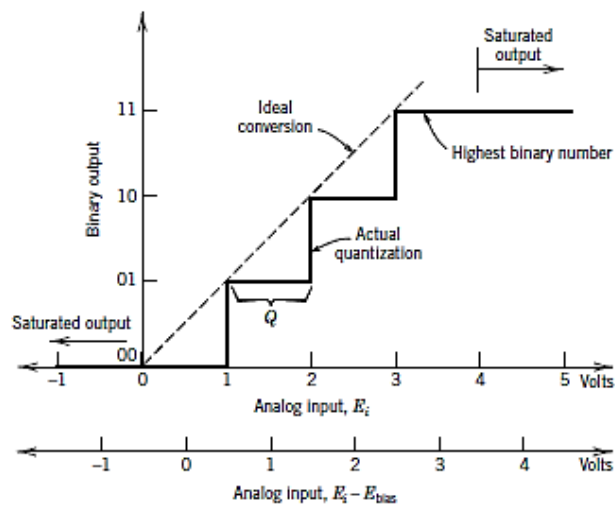


Figure 3-7 Binary quantification and saturation [2]

Having a higher than needed sampling rate can unnecessarily complicate data transmission and storage. A higher sampling rate can translate for example in more power consumption of the system and more overall storage needed. This is why it is necessary to know the nature of the physical phenomena to be measured and the information needed to be measured from it.

This system will use an Arduino MEGA which has a 10 bit A/D converter and has a maximum reference voltage of 5 V [10]. With this reference voltage, a resolution of 4.89 mV is possible. More specific detailed explanation of the data acquisition system (DAQ) used in the smart walker will be given in Chapter 7.

Amplifier Design Considerations

Estimation of Resolution of Overall System

The voltage output from the bridge due to an applied axial load to the handles was estimated using the analytical models and the FEM model. The results are shown in Figure 3-8. At the moment the amplifier was being fabricated, the goal was to be able to get readings of up to 100 lbf per side with a sensitivity of at least 1lbf/mV. It was indicated previously that the precision of the 10-bit A/D converter in the Arduino MEGA with a 5 V reference voltage is about 4.8 mV per bit count.

Knowing the precision of the A/D converter, the required response from the system is known to achieve a specific measurement resolution. In the case of a 4.8 mV resolution, the needed response has to be ≥ 4.8 mV/lbf. Also, another approach would be to reduce the reference voltage of the A/D converter. The lowest possible reference voltage in the Arduino MEGA without having to use the physical V_{ref} pin is 1.1 V. For this reference voltage the resolution is 1.08 mV per bit count.

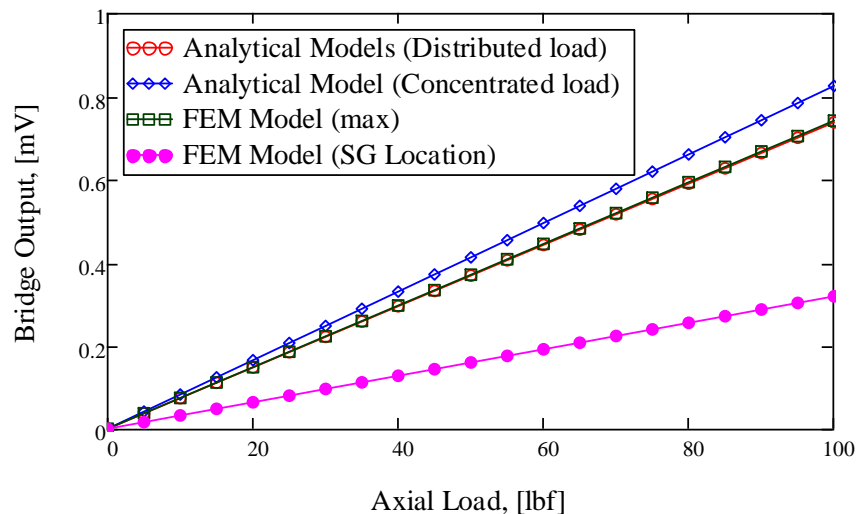


Figure 3-8 Estimated bridge deflection from analytical and FE model

Figure 3-8 shows the expected bridge deflection using the strain calculations from the analytical models (simplified geometry) and the FEM model. This was calculated using Eq. 3-5 where the gage factor used for the calculation is $GF=2$, which is a usual gage factor for the type of strain gages used. As it was indicated before, the strain in the analytical model is expected to be higher than at the location where the strain gage was installed because in the actual walker the strain distributes in the curvature area rather than being at a single point where a vertical and horizontal member connect.

Since the location of the strain gage was chosen from the results of the finite element model, these results are used to estimate the deflection of the bridge and the voltage output from amplifiers of different gains. Eq. 3-7 is used where the excitation voltage E_i is 5 V, and since it is a quarter bridge, κ is 1. Figure 3-9 shows the estimated output voltage with an output voltage at initial conditions of 100 mV.

Knowing the precision of the A/D converter, the actual resolution of the system can be estimated. Table 3-1 shows the precision of an A/D converter depending on the number of bits of the A/D converter and the reference voltage used. The reference voltages used in the table are the ones that can be set through software in the Arduino MEGA. Table 3-2 shows the sensitivities that would be obtained from various amplifications and also the overall resolution of the system (using Arduino MEGA's 10-bit A/D converter) for various gains and reference voltages of 1.1 V and 5 V.

From the criterion of being able to measure at least 1lb/mV, a gain of 1000 or larger would suffice for a reference voltage of 1.1 V. Similarly for a reference voltage of 5 V a gain of 2000 or larger would be sufficient.

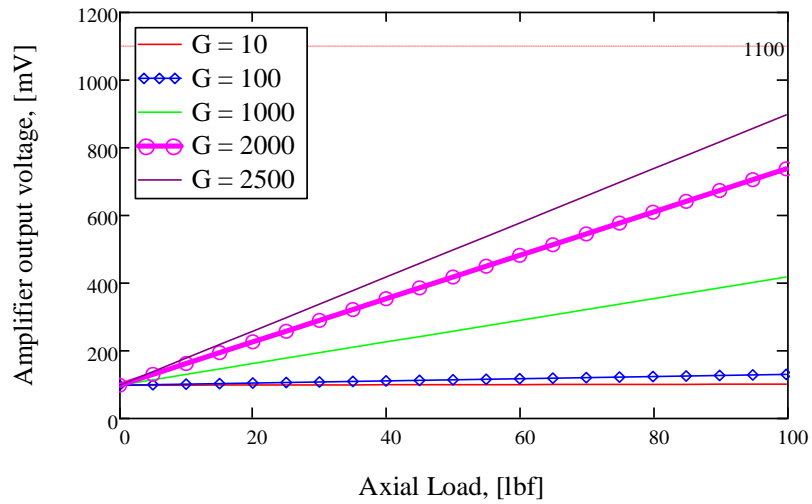


Figure 3-9 Estimated voltage response of the system with amplifiers of different gains to applied axial load

Table 3-1 DAQ system resolution depending on N-bit of A/D converter and reference voltage in [mV/bit count]

N-Bit	DAQ Reference Voltage		
	1.1 V	2.56 V	5 V
8	4.30	10.00	19.53
<u>10</u>	<u>1.07</u>	<u>2.50</u>	<u>4.88</u>
12	0.27	0.63	1.22
16	0.02	0.04	0.08

Table 3-2 Estimated sensitivities and system resolution for different amplifier gains and reference voltages with 10-bit A/D converter

Gain	Sensitivity, [mV/lbf]	System Overall Resolution, [lbf/bit count]	
		$V_{ref} = 1.1 \text{ V}$	$V_{ref} = 5 \text{ V}$
10	0.032	33.67	153.07
100	0.319	3.37	15.31
1000	3.190	0.34	1.53
<u>2000</u>	<u>6.380</u>	<u>0.17</u>	<u>0.77</u>
2500	7.975	0.13	0.61

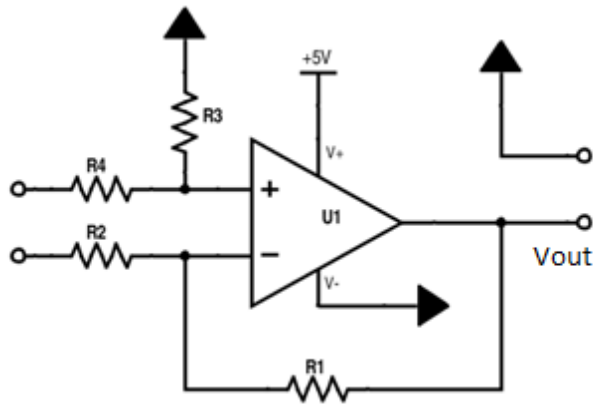
From the other criterion of needing a range of measurement of at least 0 to 100 lbs, it is noticeable that all of the gains presented would be sufficient without saturating the A/D using a reference voltage of 1.1 V. Nevertheless, if a gain of 1000 would be used, although it would meet both requirements, a significant amount of the resolution would be sacrificed noticing that for 100lbs applied it would not even reach half of the possible maximum voltage (1.1 V). Moreover, a gain of 2500 would have the best of the resolutions presented, but the gain of 2000 is chosen for the prototype to avoid applying extra amplification to a system that does not require it, which would amplify the present noise. As mentioned earlier, it is also not recommended to have such high gains with only one operational amplifier. Therefore, a gain of 2000 was chosen to maintain simplicity of the design and avoid having to cascade two or more operational amplifiers. This being said, as per the data sheet of the op-amp used LT1014CN, the maximum allowed gain is 10,000. Setting the gain to 2000 also lets us measure more than 100 lbs per side of the walker (close to 150 lbs).

Non-Inverting Amplifier Design & Fabrication

For the non-inverting amplifier, the op-amp to be used is the Texas Instruments LT1014CN. It is a 4-channel precision amplifier with very low input bias voltage and current, capable of dual supply operation in a PDIP-14 package. The main specifications are shown in Table 3-3. With this op amp it is possible to amplify the signal of each axial load sensor on the walker legs.

Table 3-3 Operational Amplifier LT1014CN Specifications

Amplifier Type:	precision amplifier
Number of Channels:	4 channel
GBP – Gain Bandwidth Product:	1 MHz
SR – Slew Rate:	0.4 V/us
CMRR – Common Mode Rejection Ratio :	97 dB
I_b – Input Bias Current:	30 nA
V_{os} – Input Offset Voltage:	300 uV
Supply Voltage – Max:	44 V
Supply Voltage – Min:	5 V
Maximum Gain	10,000
Operating Supply Current:	2.2 mA
Maximum Operating Temperature:	+ 70 °C
Minimum Operating Temperature:	0 °C
Package	PDIP-14



(a)



(b)

Figure 3-10 Non-Inverting op-amp schematic (a)
and Texas Instruments PDIP-14 package (b)

The schematic of the amplifier is shown in Figure 3-10 (as it was shown previously). The resistances used are $710\text{ k}\Omega$ for R1 and R3, and $350\text{ }\Omega$ for R2 and R4, to get an expected gain of 2030. The gain is calculated using [2]; the fabricated circuit is shown below in Figure 3-11.

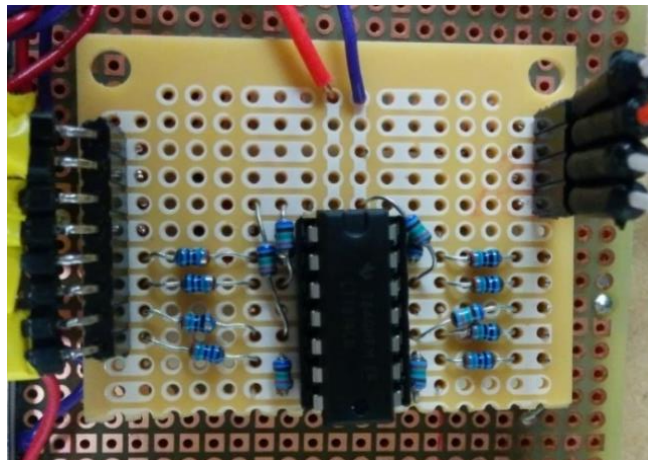


Figure 3-11 Fabricated Non-Inverting operational amplifier

True Gain Test

Due to the tolerances in the components of the amplifier, the true gain is always off the theoretical expected value. Also, to test that the amplifier is operating properly, a true gain test was made on each one of the channels. The results are shown below in Figure 3-12 and the gains measured as the slope of the curves shown in Figure 3-12 are listed in Table 3-4.

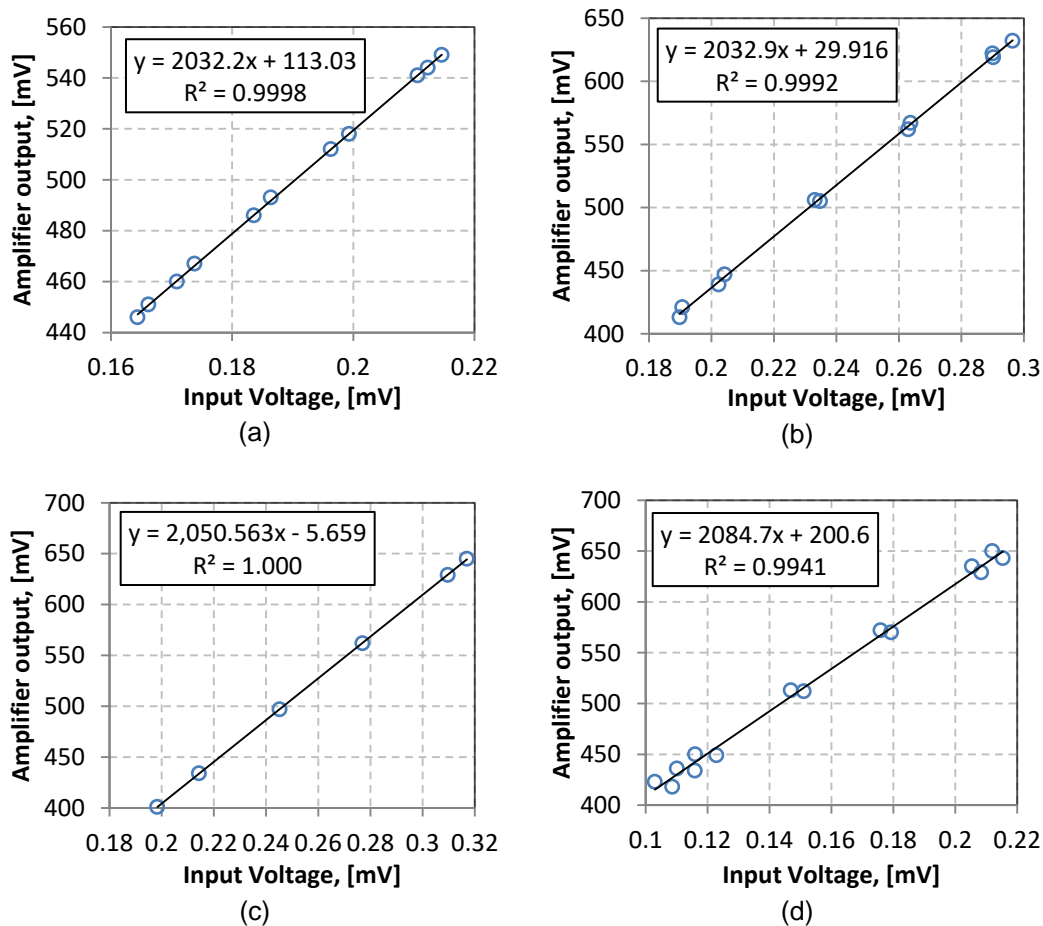


Figure 3-12 True gain test results for axial load sensor amplifier.

Channel 1 (a), 2 (b), 3 (c), 4 (d).

Table 3-4 Summary of results from true gain test of axial load sensor amplifiers

Channel	True Gain
1	2032.2
2	2032.9
3	2050.6
4	2084.7

Calibration

The calibration of the axial load sensors was performed by applying known weights to the handles in increments of 1 lbf, 5 lbf, and 20 lbf in a range from 0 to 40 lbf. When the calibration was performed, the maximum load needed to be measured per side was lower than initially thought, reason why the range of measurements for the calibration went only up to 40 lbs. Nevertheless, the system is able to measure up to 150 lbs per side (maximum allowed by the RW's manufacturer) if needed.



Figure 3-13 Application of the load for axial load calibration

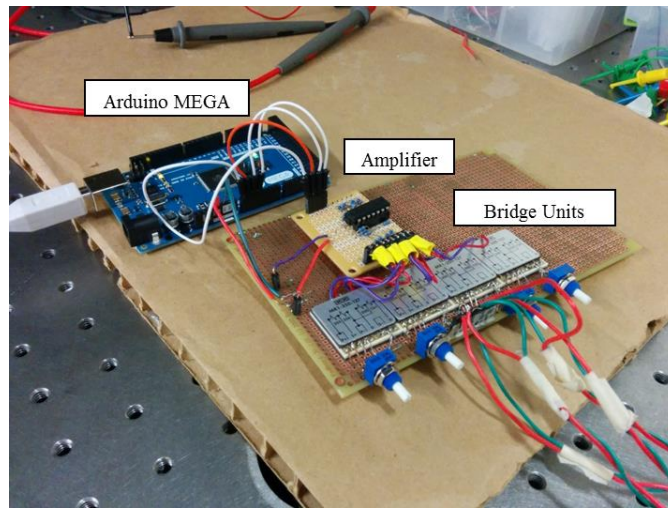


Figure 3-14 Experimental setup for axial load calibration

The flow diagram showing the components of the experimental setup is shown below in Figure 3-15. The measurements were performed by reading voltages from multimeters (analogically) and acquiring the signal using the Arduino MEGA. This was done to compare readings and test the DAQ.

The Arduino MEGA 5V pin supplied the excitation voltage for the bridge and was the voltage source for the op-amp. The voltage output from the bridge was measured using a FLUKE 8845A precision multimeter. The output from the amplifiers was acquired using the Arduino MEGA and also monitored using a multimeter. Part of the experimental setup can be seen in Figure 3-14.

The calibration curves of each strain gage are shown in Figure 3-16. The first strain gage observed in Figure 3-16 (a) experiences high hysteresis compared to the other two strain gages. Since this strain gage has been loaded several times what seems to be causing this behavior is poor bonding. This strain gage has been replaced, but it serves as example of the behavior of the gages with poor bonding [6] - [8].

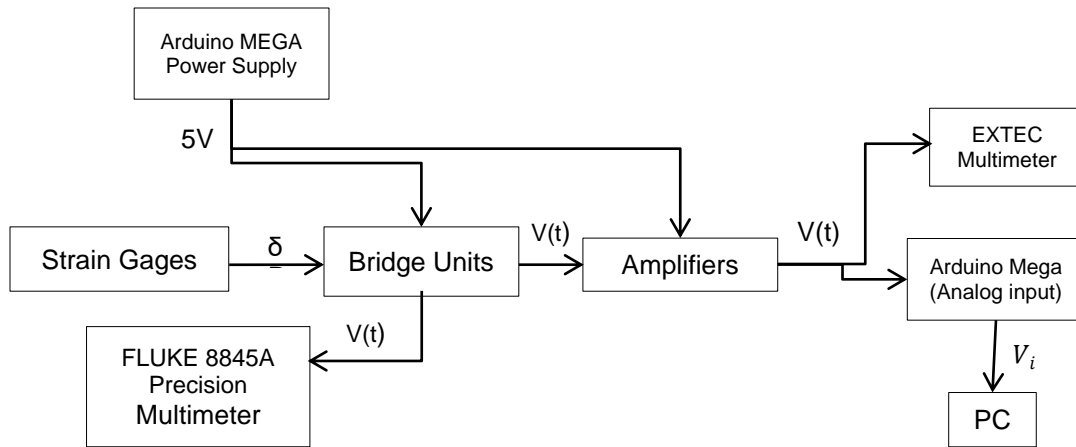


Figure 3-15 Axial load Calibration Flow Diagram

As it was seen on the FEM model, the sensitivity of the sensors is highly dependent on the location of installation; experimentally it is seen also that the strain gages that are closer to the curvature experience a higher response to loading.

Table 3-5 summarizes the sensitivity of each of the sensors as well as the resolution of the measurement using the 10-bit A/D converter in the Arduino MEGA with a reference voltage of 1.1 V.

Table 3-5 Sensitivity and overall resolution of measurement

Strain gage (leg)	Sensitivity [mV/lb]	Overall Resolution [lb/bit count]
Back/Right	2.48	0.43
Back/Left	5.14	0.21
Front/Right	6.37	0.17
Front/Left	6.01	0.18

The resolution of the front leg sensors are almost exactly as predicted from the FE model and bridge and amplifier estimations. Although the gauge on the back leg has a lower resolution, it is still within the acceptable margin of 1 lb per bit count. This

difference is because the location of the strain gage is further away from the curvature of the leg, thus it doesn't sense as much of the bending strain.

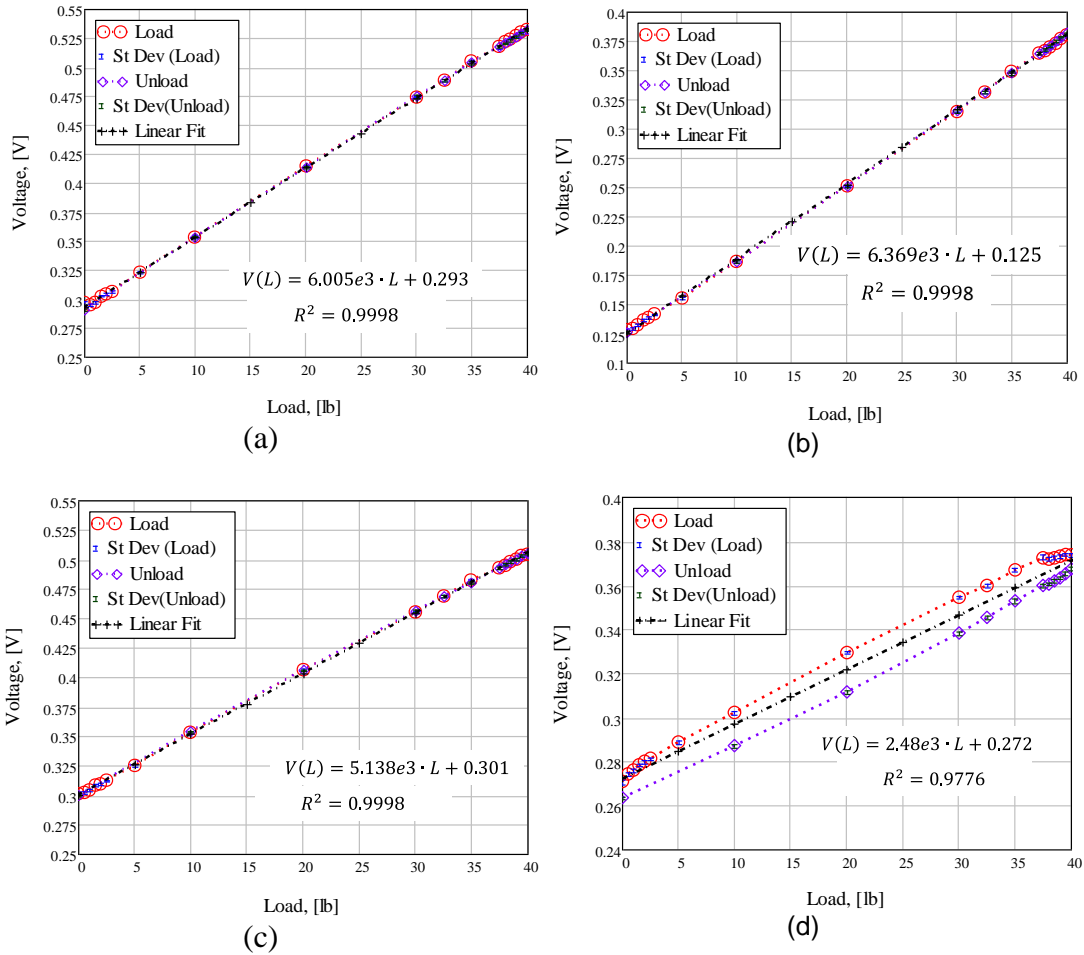


Figure 3-16 Axial load sensors calibration curves. Corresponding to legs (a) Front/Left, (b) Front/Right, (c) Back/Left, (d) Back/Right

Effect of Load Applied on Opposite Handle

It was noticed experimentally that the sensors on the side where the load was not being applied still experienced some strain. This strain was not quantified analytically or with the FE model because it would have been cumbersome to replicate actual joint conditions of the members of the walker. The effect will be quantified experimentally and a relation will be found to include this effect on the overall calibration curve in the final user interface. The experimental setup is the same as in the calibration process described above, except that the loads are being applied on the opposite side of the walker where the sensor is installed.

The effects on the sensors when a load is being applied on the opposite side handle are shown below in Figure 3-17 Figure 3-18.

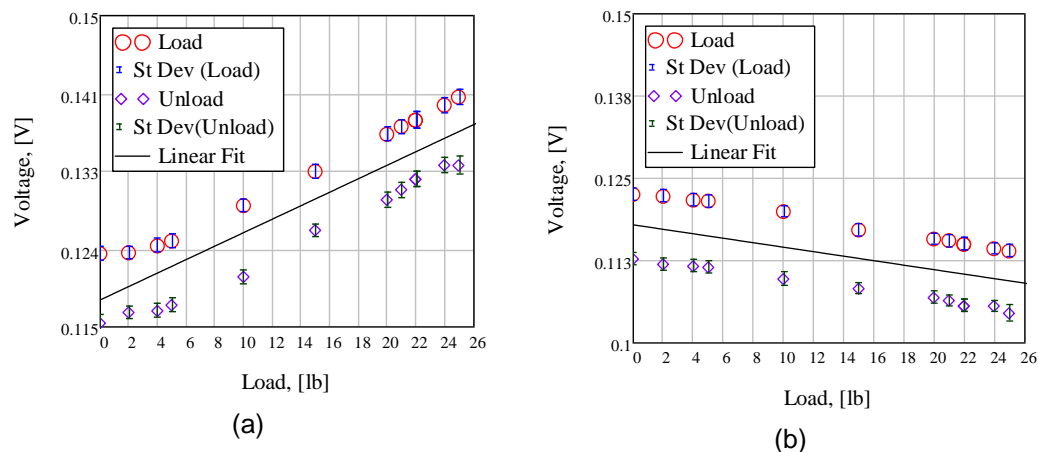


Figure 3-17 Effect on the left side legs when load is applied on right handle.

Front Leg(a), Back Leg (b)

Notice that the slope is negative for the sensors in the back, and positive for the ones in the front. Also the slope is steeper for the sensors on the left side; they are more sensitive to loads applied to the opposite side. The linearity on the left side is much better

than on the right side. Because the bar that connects both sides of the walker is not completely rigid and fixed, it is difficult to find a linear relationship.

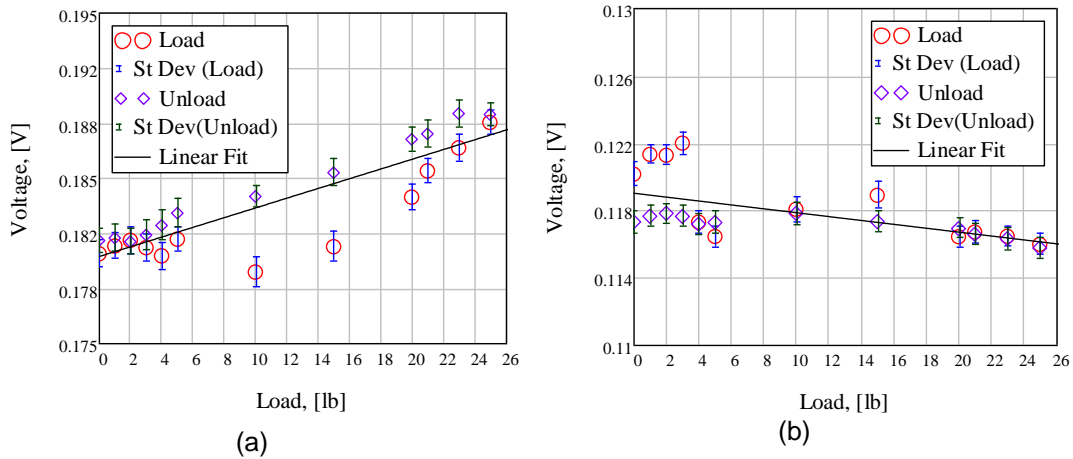


Figure 3-18 Effect on the right side legs when load is applied on left handle.

Front Leg(a), Back Leg (b)

Table 3-6 Slope of calibration curves and overall effect on sensor readings when the load is being applied on the opposite handle

	Slope [mV/lbf]	Effect on reading [lbf per lbf applied on opposite side]
Front/Left	0.76	0.13
Back/Left	-0.34	-0.07
Front/Right	0.29	0.05
Back/Right	-0.12	-0.05

Notice also that linearity wasn't achieved in the results of the right side after numerous attempts, not because the response is not linear by nature, but because the members of the walker are not completely bonded and during testing, joints that were practically fixed experienced release or friction and shifted the voltage readings.

The effect on the sensors when loads are being applied on the opposite side was measured to be relatively small but not negligible. For example on the front left gage an offset of 1.3 lbf would be felt on every 10 lbf applied on the right side.

The characterization of this effect is not simple. When loads were applied on both sides, the effect that an applied load on the opposite side where the strain gages are installed was observed to diminish. It is recommended for following prototypes to bond the joints connecting all the member of the walker to help the characterization of this effect.

Chapter 4

Torque Measurement

Another main objective in this project is to be able to measure the torque applied by the user to the handle. Strain gages will be used as well for this purpose, measuring the shear strain at the surface caused by the torque. In this case rosettes with grids rotated 45 degrees as shown in Figure 4-1 were used in order to have four varying resistances in a full bridge configuration. A similar design process to the axial load sensor was carried. Analytical calculations were performed to assess the proper gain that would be needed to have a proper measurement resolution. This time an existing amplifier with a gain of 1000 was available. The full-bridge configuration response from an applied torque was predicted to ensure that the amplification would be sufficient. Later the system was calibrated to obtain actual calibration curves.

Strain Gage Measurement Overview for Torque Measurement

Basic Theory – Full Bridge

In Chapter 3 a closed form solution to predict the voltage output from a Wheatstone bridge was shown in Eq. 3-5.

In torque measurement using strain gages a full-bridge configuration is used. Four strain gages, rotated in a 45 degree angle are installed as shown in Figure 4-1. All four gages (4) contribute to the bridge deflection and so Eq. 3-5 with $\kappa = 4$ becomes

$$\frac{\delta E_o}{E_i} = \frac{4\delta R / R}{4 + 2\delta R / R} = \frac{4GF\varepsilon}{4 + 2GF\varepsilon} \approx GF\varepsilon . \quad \text{Eq. 4-1}$$

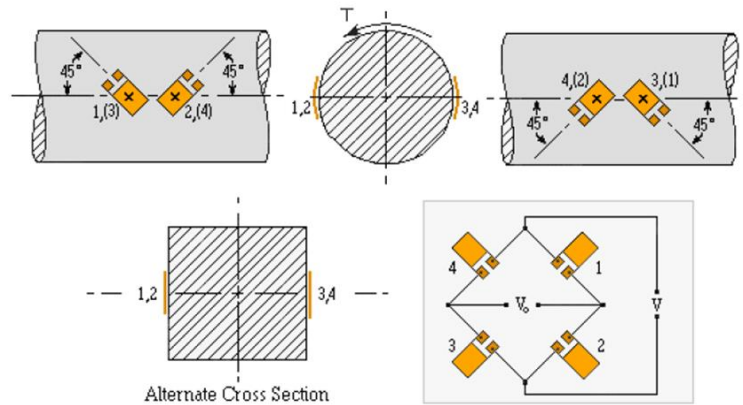


Figure 4-1 Strain gage configuration for torque measurement. [3]

For simplicity of installation and accuracy of relative angle between gages, a rosette with two grids rotated 45° was used.

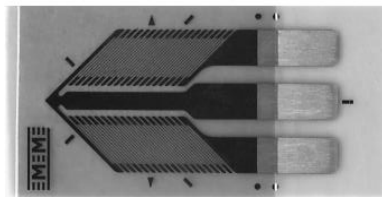


Figure 4-2 General Purpose Strain Gages - Shear/Torque Pattern [5]

The bonding and surface preparation is the same as that for the axial load strain gages, although the installation is a little more difficult for special constrains.

It is evident that because the four (4) grid patterns are the four varying resistances forming the full bridge, there is no need for a bridge completion unit. The voltage output from the bridge is directly amplified and then acquired by the Arduino MEGA.

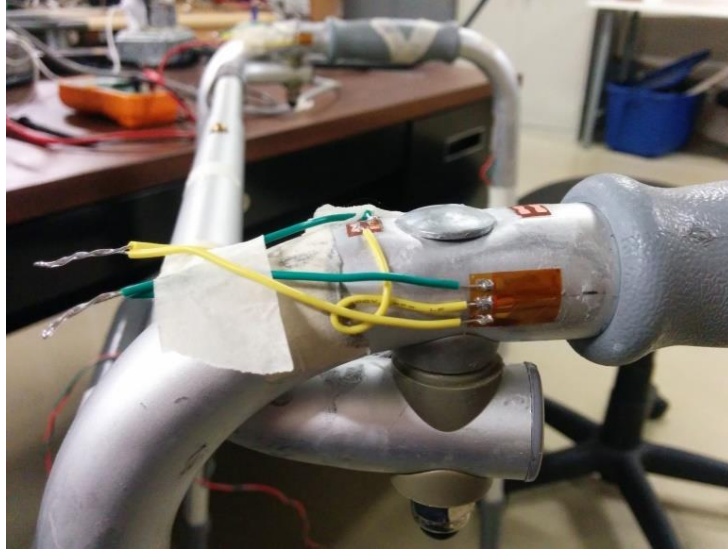


Figure 4-3 Installed Rosettes on left handle

The maximum shear stress on the surface of a cylindrical member subjected to torque is given by,

$$\tau_{\max} = \frac{TR}{J} \quad \text{Eq. 4-2}$$

Where, T is the applied torque, R is the outer radius and J is the polar moment of inertia.

The polar moment of inertia of a hollowed circular cross section is,

$$J = \frac{\pi \left((2R)^4 - (2r)^4 \right)}{32} \quad \text{Eq. 4-3}$$

R and r are the outer and inner radius respectively.

From here the maximum shear strain is,

$$\gamma_{\max} = \frac{\tau_{\max}}{G} = \tau_{\max} \frac{2(1+\nu)}{E} \quad \text{Eq. 4-4}$$

Where G is the Shear modulus, E is Young's modulus and ν is the Poisson's Ratio.

Finally, the expected bridge response as a function of torque applied is given by,

$$\frac{\delta E_o}{E_i} \approx GF \gamma_{\max} = GF \frac{32TR}{\pi((2R)^4 - (2r)^4)} \frac{2(1+\nu)}{E} . \quad \text{Eq. 4-5}$$

Based on the walker's handle dimensions and aluminum's properties the expected signal from the bridge for an excitation voltage of 5 V would be as shown in Figure 4-4. With this we can estimate the output signal from an amplifier as well. In this case the approach was different than for the axial load sensors. There was an amplifier available with a gain of 1000 that was fabricated earlier. With this model we can predict the output and see if it meets the required resolution. It is easy to visualize from the Figure 4-4 that the output from the amplifier will be about 0.3 V for a torque of 1 N-m or 8.85 lb-in t. The expected sensitivity of the system would be 35.6 mV/lb-in, which results in a resolution of 0.03 lb-in per bit count and sufficient for this purpose. Also, considering that torque is being measured in both directions, the range of voltages available is 550 mV on each direction. Using 1.1 V as a reference voltage (as explained in Chapter 3) for the A/D conversion, would result in a maximum reading of about +/- 16 lb-in.

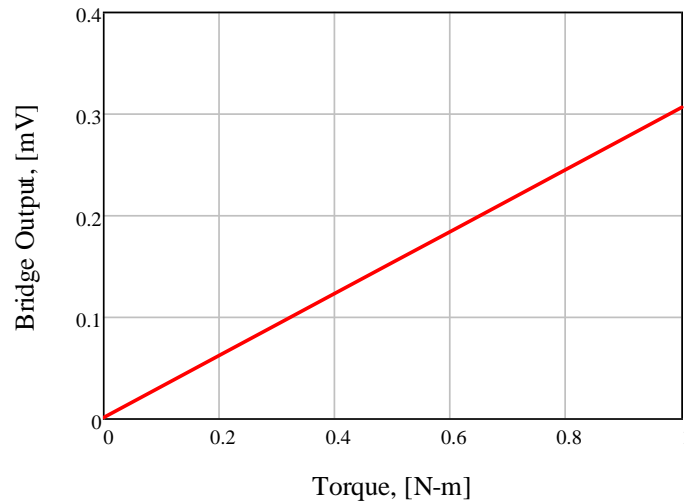


Figure 4-4: Expected Full-Bridge output (Torque)

Circuit Design

On the same circuit board where the axial load sensors were connected, the zeroing circuit for the torque sensor was added. It is very important for this circuit to have a zeroing function. The measured output voltage with no load applied is needed to be at the center of the voltage range of measurement, this is 550 mV. This zeroing capability does not affect the performance of the bridge; it maintains its response no matter what value the voltage takes at initial conditions. Figure 4-7 shows the schematic of the circuit. It includes the zeroing circuit, the full-bridge and the non-inverting amplifier.

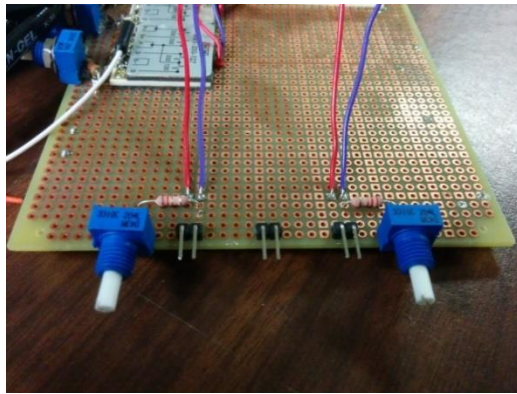


Figure 4-5 Zeroing circuit for torque sensor in
main circuit board

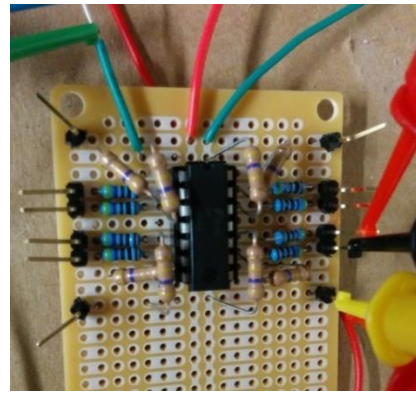


Figure 4-6 Fabricated Amplifier used in
torque sensor ($G=1000$)

Table 4-1 Components of Torque sensor's circuit

Component	Description
V_{ext}	Excitation Voltage (5 V)
R_{pot}	Potentiometer
R_{null}	Nulling Resistor
$R_1 \cdots R_4$	Strain Gages (350Ω)
R_5, R_6	470Ω
R_7, R_8	$470k\Omega$
J_1	Arduino analog input port

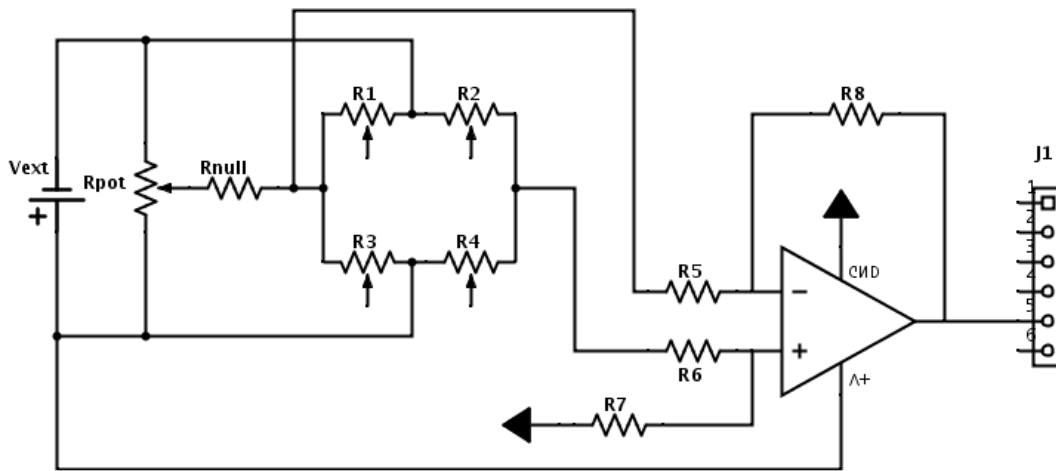
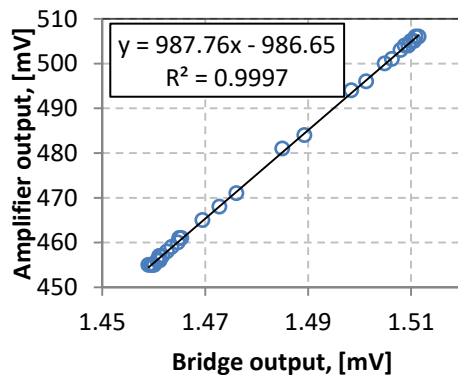
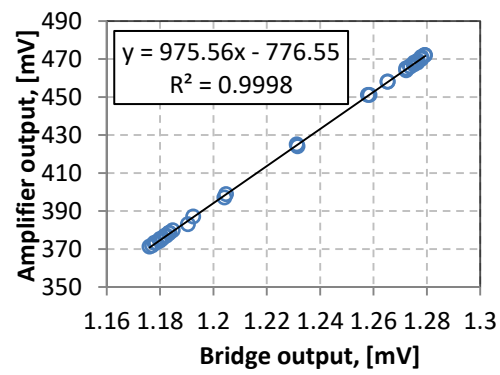


Figure 4-7 Circuit schematic of torque sensor

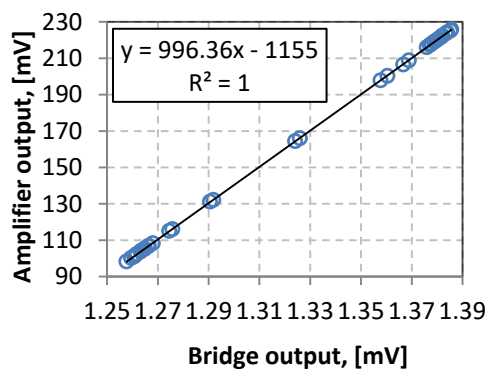
True Gain Test



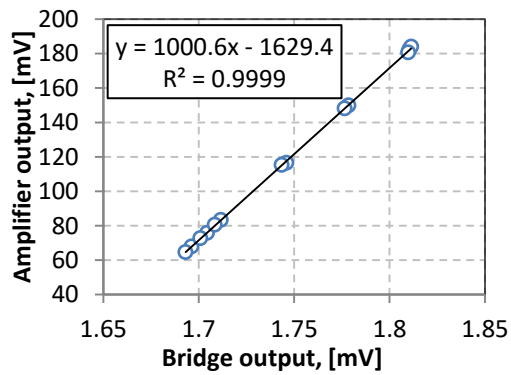
(a)



(b)



(c)



(d)

Figure 4-8 Torque amplifier true gain test results. Channels 1(a), 2(b), 3(c), 4(d)

The amplifier to be used was tested to obtain the true gain of each of its channels. Although only two channels are needed, because the chip had capability of four channels all of them were made. Above are the results from the true gain test.

The gains of the amplifiers are as expected. Although the input voltage is very small and difficult to measure, linear relations were found.

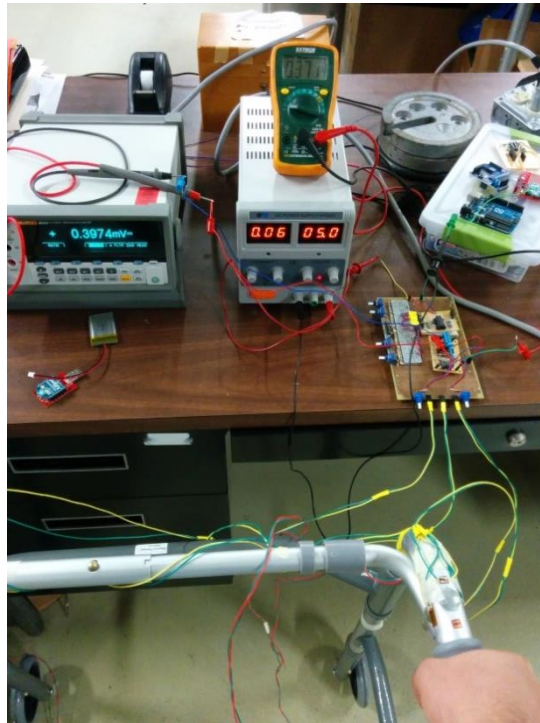


Figure 4-9 Zeroing circuit & Amplifier true gain test

Calibration

Finally to complete the torque sensor it is necessary to measure the actual response to applied torque and obtain calibration curves. For this purpose the simplest way found to apply known torques to the handle was by 3D printing a clamp and rod and hang measured weights at known distances from the center of the handle. Figure 4-10 shows the printed clamp used for the calibration.



Figure 4-10 Clamp 3D printed used to apply torque for calibration

The experimental setup is similar to the one used for the axial load sensor calibration. It is shown below.

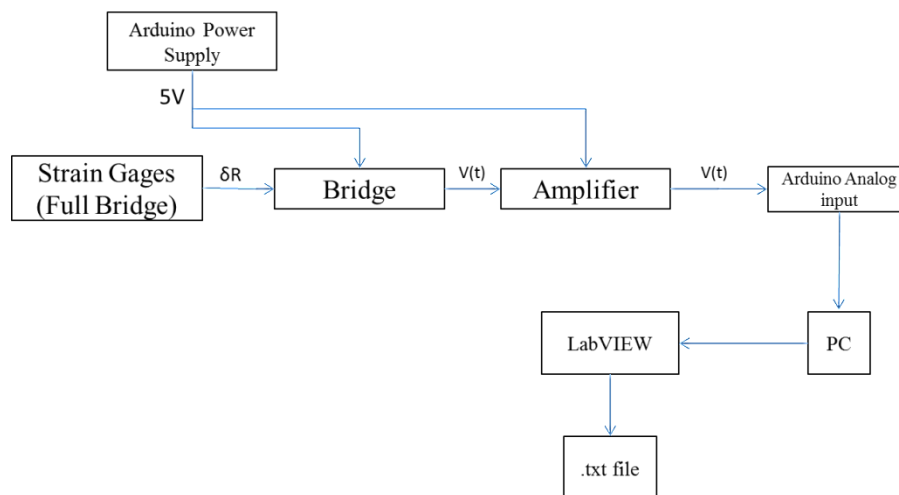


Figure 4-11 Experimental setup of torque calibration

The clamp with a 6in bar was used to apply torque to the handles as shown in the Figure 4-10. Two tests were performed hanging a mass of 0.5lb and 1lb every inch from the base of the bar, having torques applied of [0.5, 1, 1.5, 2, 2.5, 3] lb-in and [1, 2, 3, 4, 5, 6] lb-in. This test was performed in both directions of twist to make sure the voltage response is measured in both directions. About 30 seconds of data was acquired per

load applied and the average of the reading was used to get each point and construct the calibration curve.

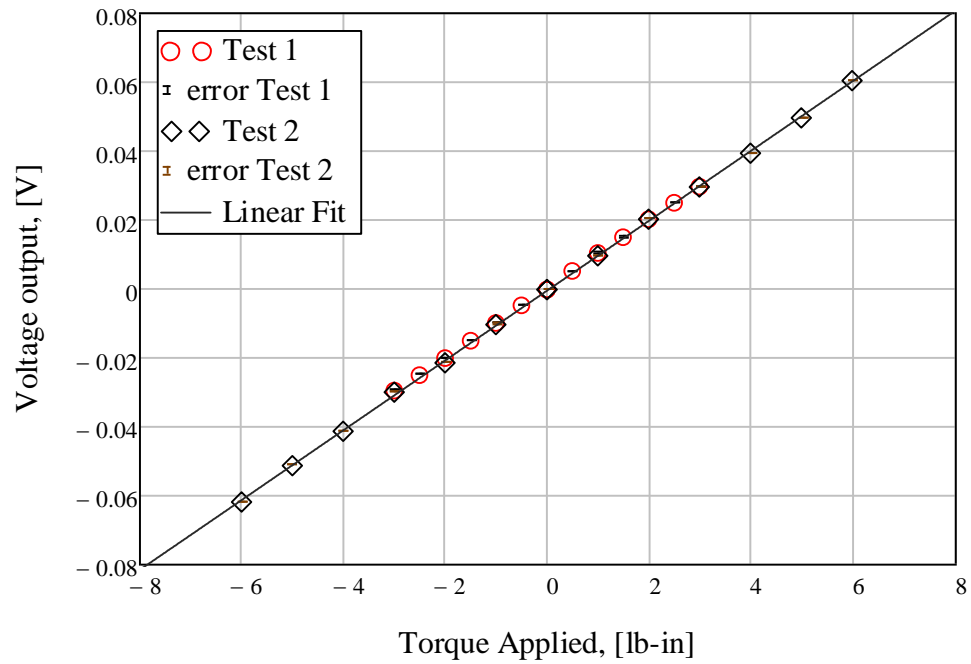


Figure 4-12 Calibration curve of torque sensor in left handle

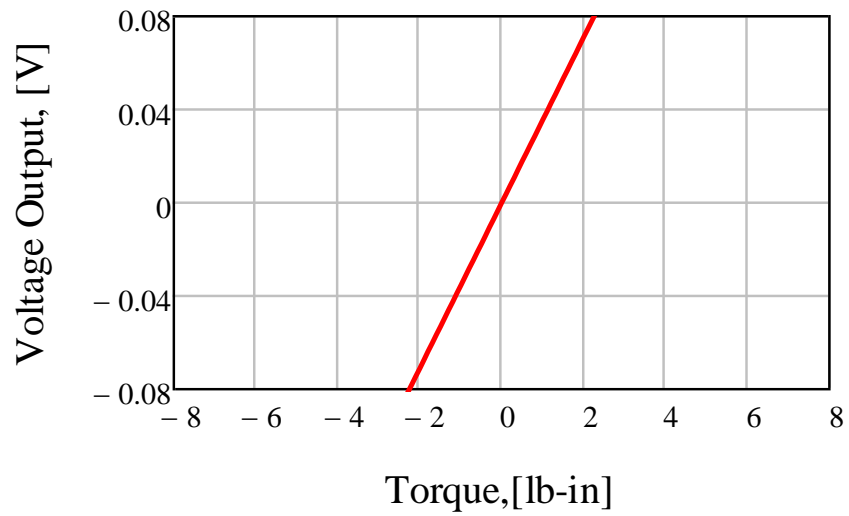


Figure 4-13 Expected response from analytical model

Table 4-2 Summary of linear fit coefficients

	Left Handle	Right Handle
Slope [mV/lb-in]	10.1	10.6
Intercept [V]	0.00036	-0.00001
R²	0.9998	0.9995

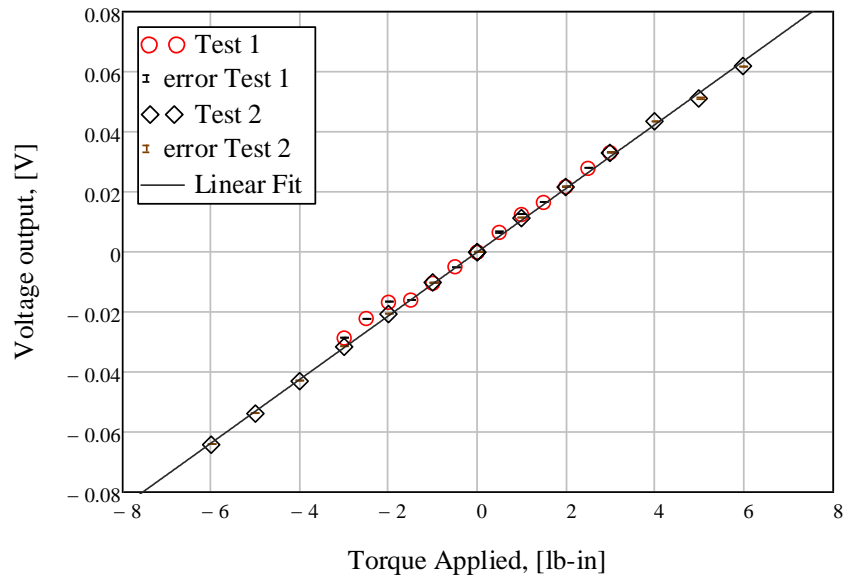


Figure 4-14 Calibration curve of torque sensor in right handle

Observe that the measurements acquired with the Arduino have very low noise and the sensor responds linearly (as expected) to applied torque.

The slope obtained is about three times less of what was expected from the analytical model (Figure 4-13). The reason could be that the actual geometry is different than the ideal model used as well as misalignments in the rosettes. Consider also that the inner and outer radius were taken at the end of the legs, at the handle dimensions could be different. Since the sensitivity gives a resolution of measurement of about 0.1lb-in per bit count this mismatch from the theoretical model did not really affect negatively the ability of the sensor to measure applied torque. The sensor has an acceptable resolution and a maximum range of measurement that allows for bigger torques to be

measured. The sensor can measure up to +/- 54 lb-in (left handle) and 56 lb-in (right handle) if the voltage with no load is set at 550 mV. Still, such high applied torques are not expected from elderly patients in a nursing home.

Effect of Axial load on Torque Measurement

There is strain that is caused by axial loads at the location where the torque rosettes were installed. This was observed experimentally noticing torque readings when only axial load was being applied. This effect was measured to subtract it from the final reading of the torque sensor. Curves were obtained by performing the same procedure as with the axial load sensor calibration. The data points do not follow a single curve because the effect changes according to the position at which the axial load is being applied but the correlation is still acceptable to have an accurate torque measurement.

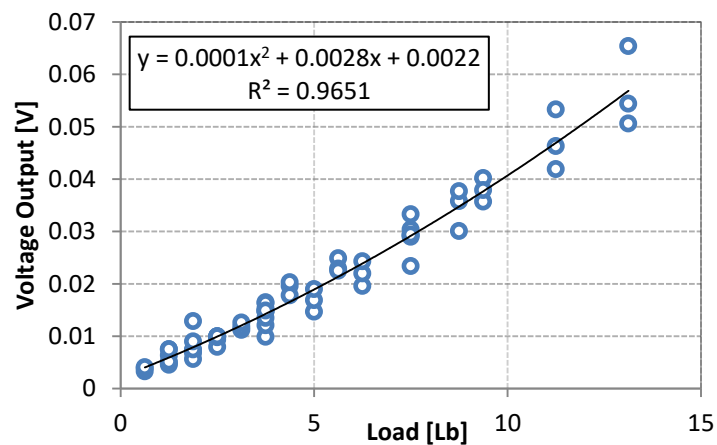


Figure 4-15 Left handle torque sensor's response due to axial load

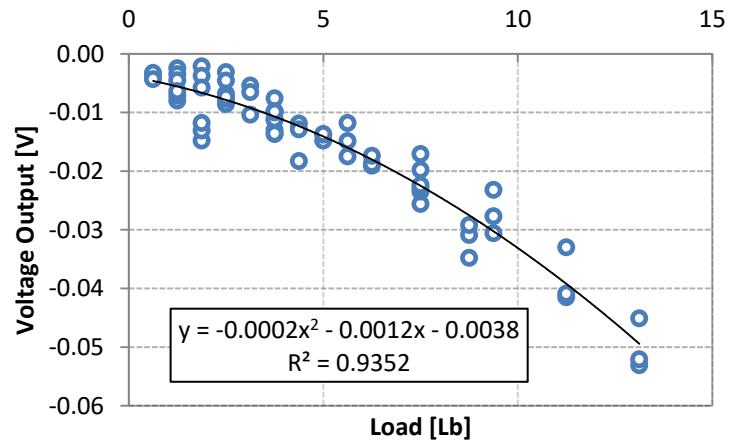


Figure 4-16 Right handle torque sensor's response due to axial load

From these figures it is noticeable that the response is not linear and it can be approximated with a quadratic fit. These curves include various test runs with axial load applied at different locations of the handle.

Chapter 5

Gripping Force Measurement

FSR Sensor Review

Similarly to strain gages and other sensors, the FSR sensor experiences changes in resistance according to the measured physical quantity, in this case force. These changes in resistance are translated into an output voltage that then is correlated to the physical quantity in question.

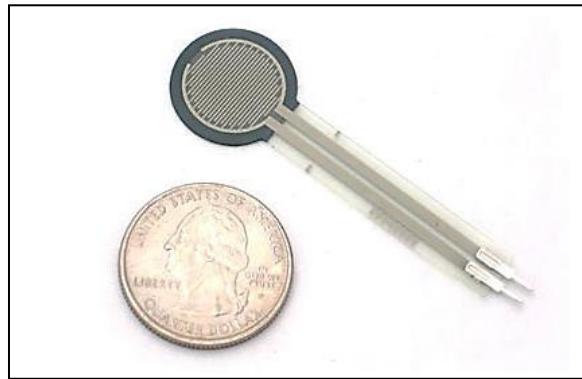


Figure 5-1 ForceSensitive Resistor used to measure gripping force [20]

Because the change in resistance with applied force in the FSR sensor is large, a Wheatstone bridge is not used to translate this change in resistance to voltage, as it is used with strain gages. With strain gages, the resistance is maintained within a value and changes slightly with applied loads, reason why a bridge is needed to capture this slight change in resistance. The FSR sensor has an infinite resistance when no load is being applied (open circuit) and with force applied the resistance reduces progressively until it is very small. The figure below shows the non-linear change of resistance with applied force.

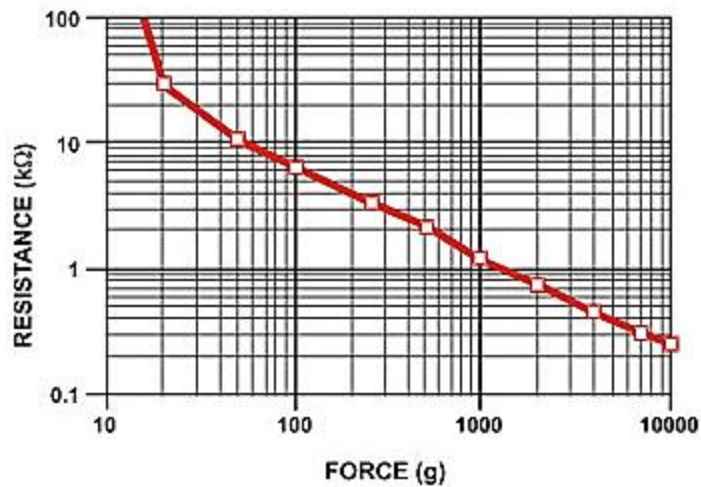


Figure 5-2 FSR Resistance vs. Force curve [20]

The FSR sensor is fairly low cost and easy to use and install but they are rarely accurate [20]. Definitely not even close to the accuracy of Strain gages, but for the purpose of this measurement its accuracy is enough. Ranges of responses are expected more than an exact force measurement.

The FSR sensor has the specifications shown below,

Table 5-1 FSR sensor technical specifications [20]

Diameter [in]	0.5
Resistance Range [Ω]	Infinite (no force) to 200 Ω max force
Force Range [lb]	0 to 20 lb applied over 0.125 in ² of surface area
Power Supply	Less than 1 mA od current

Circuit

The FSR sensors are very easy to install and use. The easiest way to measure force is to connect one end to Power and the other to a pull down resistor to ground as shown below [20]. This basically makes the sensor act as a voltage valve. When there is no force applied the valve is closed and no voltage output is observed, when the FSR is pushed the valve is open and voltage is felt at the analog input. In Figure 5-3 the 5 V supply is used, so the range of voltage will be between 0 and 5 V. In the Adafruit reference given it is possible to find a tutorial to complete the acquisition with an Arduino Board.

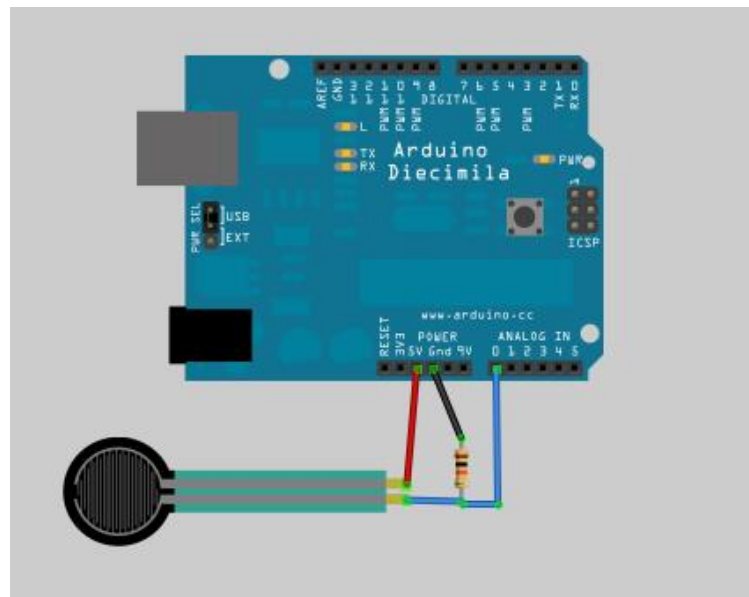


Figure 5-3 FSR Adafruit installation tutorial for Arduino [20]

Because the reference voltage for the A/D converter in this system has been set for the axial load and torque measurements to 1.1 V, the range of voltage output from the FSR sensor has to be reduced from 5 V to 1.1 V. The overall recommended circuit in Adafruit was slightly modified and a voltage divider circuit was implemented for this purpose, the schematic and components used are shown in Figure 5-4.

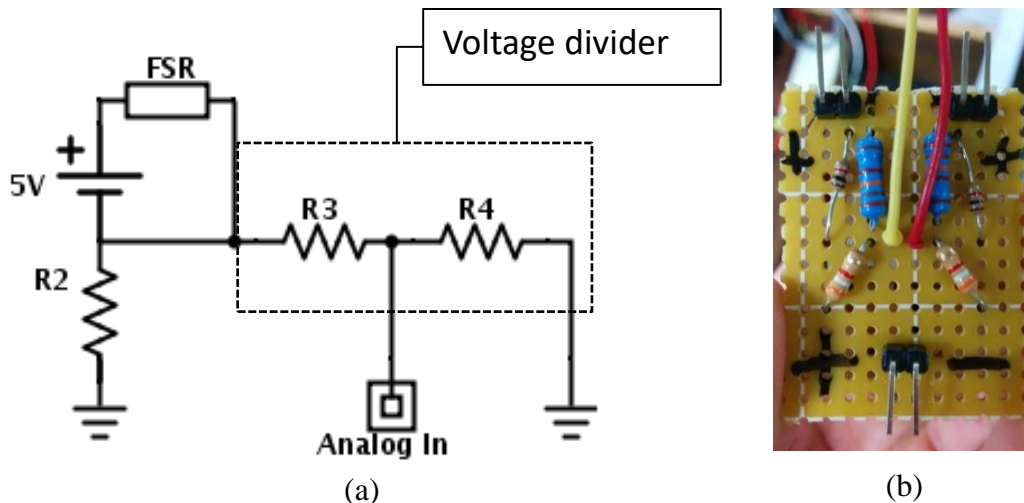


Figure 5-4 FSR circuit schematic (a) - Fabricated Circuit (b)

The values of the resistors used in the schematic shown in Figure 5-4 are 1 k Ω for R₂, 11.3 k Ω for R₃, 3.9 k Ω for R₄.

Calibration

After the circuit was made and the voltage response was observed in the wanted voltage range a calibration curve was obtained. The voltage output was observed using a multimeter.



Figure 5-5 Block diagram of Calibration setup

Known weights were applied to the 0.125 in² sensing area as shown in the experimental setup picture Figure 5-6.

The voltage response found from the sensor is logarithmic having a maximum voltage of about 1 V when 22 lb are being applied. The correlation found has good R² values although it was not possible to fit all of the points with only one function. The data was subdivided in 3 sub-sections in order to fit better the functions to the data. Figure 5-7 shows the data and calibration curves obtained.

After the calibration was performed the sensors were installed on the handle. One sensor on the inside of every handle was installed. The measuring area of the FSR was set at the handle curvature where the hand lays, same point where the axial load is expected to be applied.

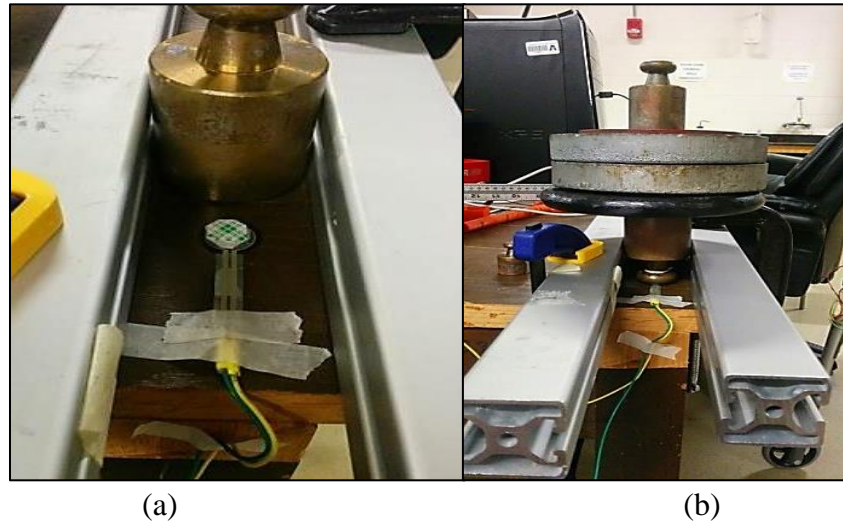


Figure 5-6 (a) Calibration setup showing FSR and alignment beams & (b) setup while force is applied to the FSR sensing area.

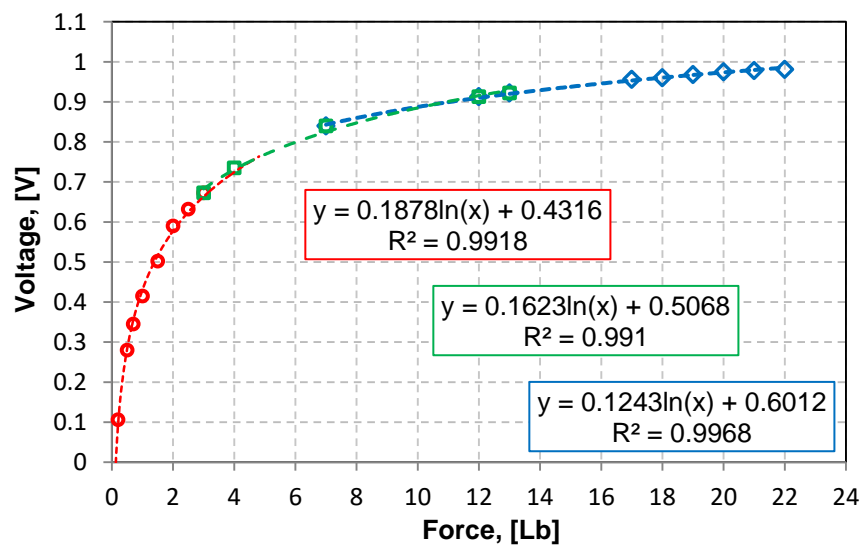


Figure 5-7 FSR Calibration curves

Chapter 6

IMU Accelerometer & Gyroscope

Additionally to the axial load, torque and gripping force measurements, it is needed to know the inclination and orientation of the walker during its use. It is also required to know the velocity at which the patient is moving. For this purpose accelerometers and gyroscopes are used. They were installed both in the walker and also in a separate unit that the patient would wear. An off the shelf IMU 6-DOF (Figure 6-1) board was used containing both an accelerometer (ADXL345) and a gyroscope (ITG3200). This section will briefly describe these sensors, how to connect them to an Arduino board to obtain measurements and sample obtained measurements.

IMU 6DOF Board Review

The Sparkfun 6 DOF IMU digital combo board includes an accelerometer (ADXL345) and gyroscope (ITG3200). The sensors communicate over I2C and pins are available to easily connect them to an Arduino Board or any board that permits I2C communication.



Figure 6-1 6DOF IMU Board [12]

The ADXL345 accelerometer is a small, thin, low power, 3-axis accelerometer with high resolution (13-bit) measurement at up to $\pm 16g$ [6]. It measures the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion or shock.

The ITG-3200 gyro is a single-chip, digital-output, 3-axis gyro. It features enhanced bias and sensitivity temperature stability, reducing the need for user calibration [12].

One of the main advantages of using this board is the ease of installation and data acquisition and available public domain information provided by Sparkfun and various Arduino enthusiasts. Much of the efforts needed to interpret the sensor's readings were helped by available Arduino libraries with supporting documentation [22] [12].

Installation

Sparkfun provides in very useful sources to install and use this IMU board with Arduino Boards. They provide the Arduino code necessary to get the data from the sensors and also a code to visualize it [12]. This was done initially to make sure that the board was working but the code was modified in order to include the readings from the analog inputs and send the data via serial connection to be read by the LabVIEW user interface.

Figure 6-2 from the mentioned tutorial [12] shows the I2C connection between the IMU board and the Arduino board. The SDA and SCL ports are the analog ports 4 and 5 in the Arduino UNO and in the Arduino Mega the SDA and SCL ports are the digital ports 20 and 21 respectively.

The Arduino code used to acquire the measurements is in the appendix and the LabVIEW code to read the data sent by the Arduino board to the PC is explained in Chapter 7. A very detail explanation of the models implemented in the libraries used in the Arduino code can be found in [22]. It was not necessary to perform any calibration. The sensors measured immediately acceleration and orientation and it was easily verifiable. Gravity was able to be observed as well as the orientation. This board however is not accurate to measure yaw. The Yaw reading is decent initially but with time it

increases even when the board maintains its same orientation. So the board can only measure effectively 5 degrees of freedom.

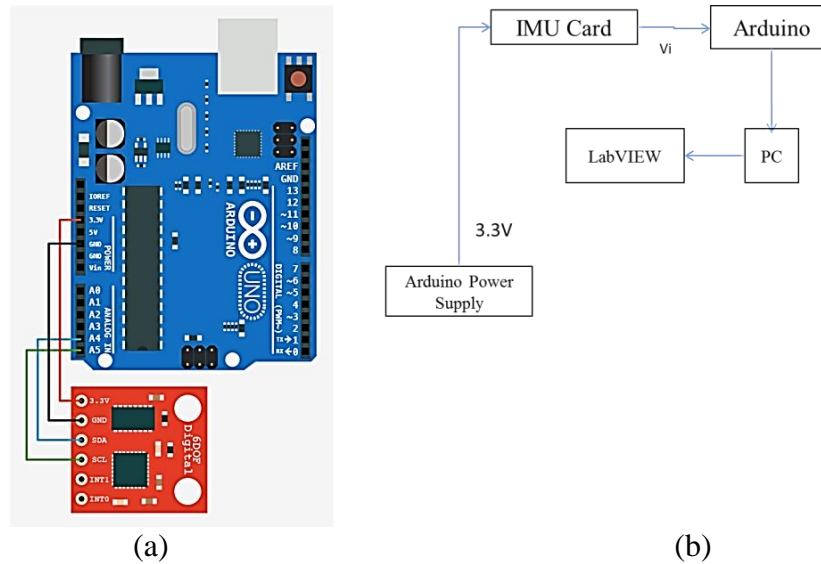


Figure 6-2 Connection of IMU card to Arduino (UNO) board [12], and block diagram of connection to PC



Figure 6-3 Front Panel of VI to acquiring data from IMU and plotting accelerometer data

Figure 6-3 shows a screen shot once the IMU board was successfully sending data through the Arduino board to LabVIEW. The data shown is random data with the IMU placed on a table; it does not represent data acquired during the usage of the walker.

Chapter 7

Data Acquisition, Wireless Transmission and User Interface

Hardware Overview

Arduino MEGA as DAQ

Arduino is an open-source project founded by Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, and David Mellis. It was built with the effort to have in the market an open-source electronic platform with easy-to-use hardware and software intended for anyone making interactive projects [10]. Arduino boards are able to read inputs (sensors, switches, digital commands, etc.) and turn them into outputs (activating a motor, turning on an LED, sending information, etc.). Figure 7-1 shows the top view of the Arduino Mega where all the inputs and outputs are displayed. The microcontrollers serves as the brain of the board and it is programmed using the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

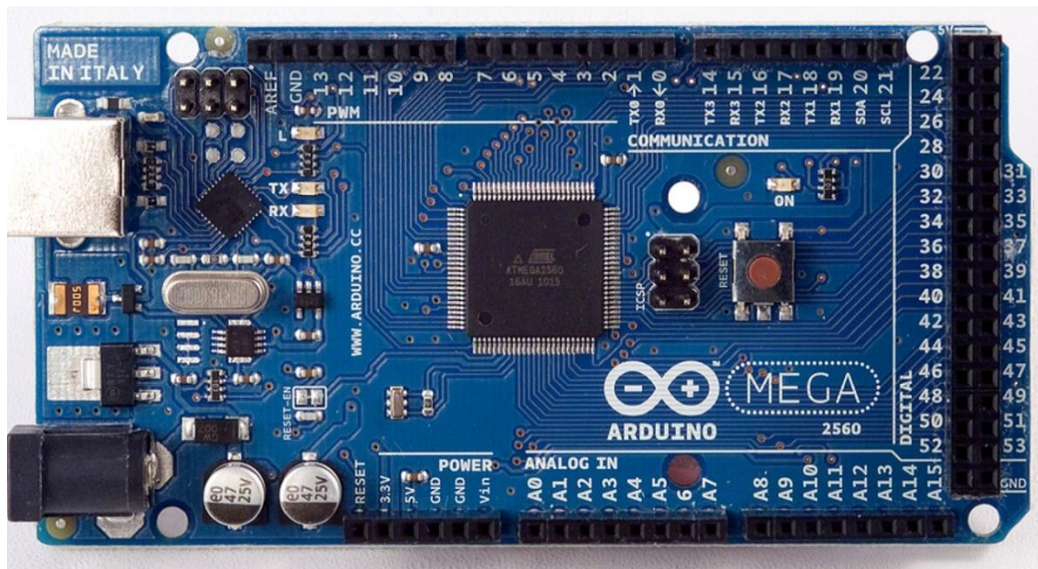


Figure 7-1 Arduino MEGA 2560 [10]

Arduino presents many advantages when building projects like the Smart Walker. Because of its language, anyone with basic knowledge of C++ can easily program the

board and link the interactions between sensors and user interface. Additionally, it has been widely used for thousands of open source projects, reason why there are many sources to look up to for similar implementations and solutions to encountered problems. Also there is a variety of sensors and circuits specifically designed to interact with Arduino (some of them used in the Smart Walker), and many open-source user-tested libraries that lessens the necessity to fabricate and write needed equipment and code from scratch. Compared to other microcontroller platforms Arduino boards are relatively inexpensive and its software (IDE) runs on all mayor operating systems (Windows, Macintosh OSX and Linux).

Arguably the most commonly used board among all the Arduino Products is the Arduino UNO. This board is very versatile and was the first in a series of USB Arduino boards. It is based on the ATmega328P microcontroller. Provides the user with 14 digital input/output pins, 6 analog inputs, a 16MHz quartz crystal, a USB connection, a power jack, a ICSP header and a reset button. This board allows for numerous possibilities and applications. Unfortunately, for the Smart Walker, the amount of analog inputs on the Arduino UNO was not enough. Including the 4 (four) channels for axial load measurement, 2 (two) for torque measurement and 2 (two) for gripping force a total of 8 analog inputs are needed in order to perform the reading with only one board. For that reason the Arduino MEGA 2560 was used.

The Arduino MEGA is based on the ATmega2560 microcontroller, and is designed for more complex projects (MEGA) [10]. It has 54 digital input/output pins, 16 analog inputs and a larger flash memory than the UNO.

Table 7-1 shows all de major technical specifications of the Arduino MEGA board. The MEGA board is the one recommended for 3D printers and robotics projects. For the

Smart Walker it serves mainly for data acquisition. In this regard the MEGA board has one extra benefit on top of the Arduino UNO regarding easily achievable resolution.

Table 7-1 Arduino MEGA 2560 Technical Specifications [10]

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by boot- loader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
Length	101.52 mm
Width	53.3 mm

Resolution

The overall achievable resolution of an analog reading in the Arduino has the possibility to be modified or maximized depending on the application. As mentioned in Chapter 3 in the Analog to Digital Conversion section, the resolution depends on the quantization of a specific sampled voltage range. The Arduino MEGA has a 10-bit A2D converter and its default reference voltage is 5 V. This results in a resolution of,

$$\frac{5V}{2^{10}-1} = 4.89mV .$$

This resolution can be maximized if the high end of the measured voltage range is known. The Arduino MEGA has the possibility to configure the reference voltage by code to three different values 5 V (default), 2.56 V, and 1.1 V. Also, Arduino UNO and MEGA give the possibility to configure the reference voltage externally if the wanted reference voltage is applied to the AREF pin (in the range 0 to 5 V only). Because the reference voltage has to be the same for all inputs, the Smart Walker the sensors were designed to output a maximum voltage of ~1 V and the reference voltage set to 1.1 V. Resulting in a resolution of,

$$\frac{1.1V}{2^{10}-1} = 1.075mV .$$

Sampling Rate

Both Arduino MEGA and UNO have one 10-bit A2D converter that samples the analog signal of all of their analog inputs. Although the clock speed is 16 MHz the highest sampling rate achievable when only reading one analog input is about 1 kHz for a number of reasons. First, some performance sacrificed in order to have the ease of the Arduino programming language. Since it is only one “brain” reading the analog signal, processing the data and sending it to the serial port, the amount of operations and speed at which this information is sent is directly related to the overall sampling rate of the system. The biggest baud rate possible is 115200 bits/sec. In order to achieve the

maximum possible sampling rate with this system, the program and internal operations have to be very mindfully considered. When sampling 8 analog inputs instead of one, the overall sampling rate is shared among these inputs because there is only one A2D converter. So in reality the analog channels are read one after the next, not all at the same time.

In the Arduino code written for the Smart Walker the analog inputs are read one after the next and then appended in one big string with the reading from the IMU card. Once the string is formed, the array of characters is sent through the serial port. A more detailed explanation can be seen in the Arduino Code section.

Including all of the mentioned factors an overall sampling rate of about 40 Hz per channel was achieved. Again, the effect of the inputs being read progressively and not at the same time is neglected. The timestamp taken as the time of the reading is the instant when the complete string of readings is read by the LabVIEW program. In the stored data the readings from all sensors are recorded at the same time. It is important to mention as well that the time interval between readings is not fixed but varies from reading to reading and on average they result in an overall sampling rate of about 40 Hz. For this application at the sampling rate achieved, these conditions and assumptions are acceptable but if the application requires a meticulous determination of the time stamps and equally spaced time intervals from reading to reading, the Arduino MEGA would not be the best option.

XBee® 802.15.4 Antennas for Wireless Data Transmission

Xbee modules are off-the-shelf embedded solutions providing wireless end-point connectivity to devices [26]. They use IEEE 802.15.4 networking protocol for fast peer-to-peer networking. These modules are ideal for low-power, low-cost applications [24].

Table 7-2 shows the main technical specifications of the XBee® 802.15.4 antennas and Figure 7-2 shows a picture of the antennas.

Table 7-2 XBee® 802.15.4 Technical Specifications [27]

Indoor/Urban Range	Up to 100.ft (30m.)
Outdoor Range	Up to 300 ft. (100m.)
Transmit Power Output (software selectable)	1mW (0 dBm)
RF Data Rate	250,000 bps
Serial Interface Data Rate (software selectable)	1200 - 115200 bps (non-standard baud rates also supported)
Receiver Sensitivity	-92 dBm (1% packet error rate)
Operating Frequency	ISM 2.4 GHz
Dimensions	0.960" x 1.087" (2.438cm x 2.761cm)
Addressing Options	PAN ID, Channel and Addresses



Figure 7-2 XBee® 802.15.4 antennas [26]

Xbee modules allow reliable and simple communication between microcontrollers, computers, or anything with serial ports. They are ideal for Arduino projects. Arduino shields are readily available to implement wireless communication using Xbee antennas. Sparkfun electronics provides a series of products to ease the use and implementation of Xbee antennas [15], many of them used in the Smart Walker. Figure 7-3 to Figure 7-5 show the Arduino shield to connect the antenna to the Arduino

board an explorer dongle to connect the antenna to the PC and the Arduino/Shield/Xbee assembly.

For the Smart Walker, one antenna is needed connected to the Arduino board and one connected to the PC where the data is being sent to. SparkFun Xbee shield was used to connect one Xbee antenna to the Arduino and SparkFun Xbee Explorer Dongle to connect the other connected Xbee antenna to the serial port of the PC.

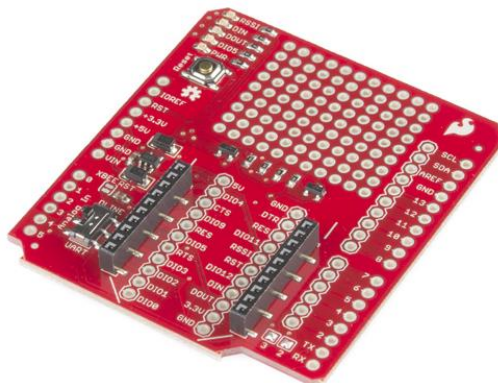


Figure 7-3 SparkFun Xbee shield [15]

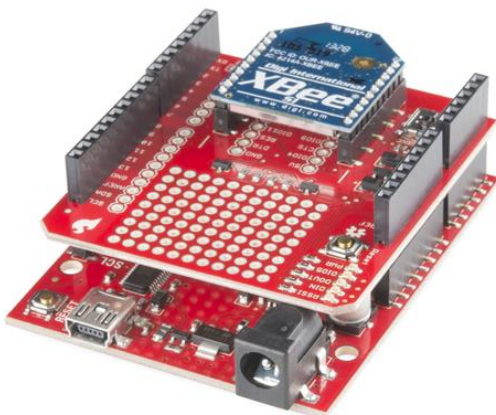


Figure 7-4 Xbee Antenna + SparkFun
Shield + Arduino board assembly [15]



Figure 7-5 SparkFun Xbee Explorer
Dongle [15]

User Interface & LabVIEW/Arduino Connectivity

Arduino Code

A detailed explanation of the Arduino code written for the Smart Walker is shown below in Table 7-3. The code was written in order to read the analog signals from the axial load, torque and pressure sensors as well as the readings from the 6DOF IMU board and wireless transmission with the Xbee antenna. Tutorials for connectivity with the 6DOF IMU and Xbee antenna can be found in [12] and [15]. The full code can be found in Appendix B.

It is important to mention that the Baud Rate has to match in the Arduino code, Xbee antenna and LabVIEW code (user interface). Any mismatch would result in communication problems between XBee antennas.

Table 7-3 Arduino Code for Smart Walker
(Analog read, IMU card read, Xbee connectivity)

Code Section	Explanation
#include <FreeSixIMU.h>	//FreeSixIMU library was made from the code FreeIMU developed by the people at Versano under GPL License specifically to accommodate to the needs of the 6DOF IMU card used in the Smart Walker [12] since it was created for a 9DOF chip originally. Their models were based on the positioning algorithms found in [22].
#include <FIMU_ADXL345.h>	//This library is for the 3-axis high resolution accelerometer chip used in the 6DOF IMU Card.
#include <FIMU_ITG3200.h>	//This library is for the 3-axis high resolution gyroscope chip used in the 6DOF IMU Card

Table 7-3—Continued

#include "CommunicationUtils.h"	//This library is needed to communicate with the 6DOF IMU Card
#include <Wire.h>	This library allows to communicate with I2C / TWI devices (6DOF IMU Card communicates through I2C with Arduino)
#include <SoftwareSerial.h>	//This library is for the serial communication between the Arduino board and the Xbee antenna [13].
//Variables for IMU reading float Acc[4]; float gyr[4];	//Initially the readings from the IMU board are taken as floats. To reduce the size of each reading the float readings will be transformed into integers in order to get a 4 integer long value.
int Acci[4]; int Acc_abs[4]; int Gyi[4]; int Gy_abs[4];	//Variables to create the truncated readings mentioned before.
String Accs[4]; String Gys[4]; String A_str_tot, G_str_tot;	//Once the reading is of the desired length as an integer, the IMU readings are transformed to a string in order to concatenate them with the analog inputs and have a final string output with all the readings.
FreeSixIMU my3IMU = FreeSixIMU();	//Set IMU object.
SoftwareSerial Xbee(2,3);	//Setup digital port to print to Xbee. The ports are chosen depending on the ports in the Arduino board.

Table 7-3—Continued

<pre>//USB Serial Connection Serial.begin(57600); // the setup routine runs once when you press reset: void setup() { //Xbee Serial Connection XBee.begin(57600);</pre>	<pre>// Initialize serial communication at 57600 bits per second for both through the USB port of the Arduino and Serial connection to Xbee. During regular operations of the walker the USB connection will not be used, but for debugging purposes the line of code is kept. This is not the fastest Baud Rate possible by the Xbee or Arduino (115200 bits/second) but it was noticed that when using the fastest Baud Rate the final string arrived at the Xbee with some errors and although the overall speed of the system was faster, the fidelity of the readings was not perfect.</pre>
<pre>analogReference(INTERNAL1V1);</pre>	<pre>//Reference Voltage for Analog reading is set to 1.1V. This is an option that only Arduino MEGA provides. It could be set to 2.56V and 5V through the code, or to any desired voltage plugged in the Vref pin. [14]</pre>
<pre>Wire.begin();</pre>	<pre>//Initiate I2C connections (IMU Board)</pre>
<pre>delay(5); my3IMU.init(); delay(5); }</pre>	<pre>//Initiate connection with the IMU board</pre>
<pre>int analog[8]; float Vf[8]; int Vi[8]; String Vs[8];</pre>	<pre>//Define variables for analog readings (Axial load, Torque and FSR). Similar to the IMU readings the float values are converted into integers and then into strings.</pre>

Table 7-3—Continued

<pre>// the loop routine runs over and over again forever: void loop() { String V_tot_string = "";</pre>	<pre>//Placeholder for the final String sent through the Xbee Antenna.</pre>
<pre>for (int x = 0; x<8 ; x++){ // read the input on analog pin 0-7: analog[x] = analogRead(x);</pre>	<pre>//Readings are bits from 0 to 1023 (integers).</pre>
<pre>// Convert to Float Vf[x] = analog[x] * (1.1 / 1023.0)*1000;</pre>	<pre>//(1.1/1023) conversion factor is used to convert the reading from bits to Volts and the 1000 multiplier to convert to mV. This value is a float.</pre>
<pre>//Convert back to integer Vi[x] = int(Vf[x]);</pre>	<pre>//Since the float was in millivolts. When is converted to an integer all the decimal values are dropped. As it has been mentioned before, the resolution of the readings is a little over 1mV which makes this readings more than adequate.</pre>
<pre>//Convert to String if (Vi[x]>=256){Vs[x] = String(Vi[x],HEX);}; if (Vi[x]<256){Vs[x] = "0" + String(Vi[x],HEX);}; if (Vi[x]<16){Vs[x] = "00" + String(Vi[x],HEX);};</pre>	<pre>//Integer values are converted to hexadecimal strings in order to reduce the length of the total string. As it can be seen in Table 7-4 decimal values of from 256 to 4095 can be represented with only three characters in the hexadecimal system. In this case the length has been reduced by one character per reading. It might seem a minor gain but including all the analog channels used, the overall reduction in the output string (8 characters) and thus the increase is overall speed is noticeable. To maintain a constant length of the output string no matter the magnitude of the reading, zeroes are appended at the beginning of the reading according its magnitude. Every channel's output string is 3 characters long no matter what voltage is read.</pre>

Table 7-3—Continued

<pre>//Create full string from analog inputs V_tot_string = V_tot_string + Vs[x]; }</pre>	<p>//The full string format of the readings from the Analog voltage readings are shown in Table 7-5 Table 7-6 . The string has a total of 24 characters, 3 per channel.</p>
<pre>A_str_tot = ""; G_str_tot = "";</pre>	<p>//Placeholder for accelerometer and gyroscope readings</p>
<pre>//Obtain values from IMU my3IMU.getValues(Acc); my3IMU.getYawPitchRoll(gyr); // Normalize Acceleration readings for(int x = 0; x<4 ; x++){ Acci[x] = int(Acc[x]*1000);</pre>	<p>//Accelerometer readings obtained are in g's and are converted to mili-g's.</p>
<pre>} //Get absolute values to avoid continuous unnecessary conversions when transforming to strings for(int x = 0; x<4 ; x++){ Acc_abs[x] = abs(Acci[x]); Gy_abs[x] = abs(gyr[x]); } //convert IMU readings to strings (4 characters) for(int x = 0; x<4 ; x++){ if (Acci[x]<0){ if (Acc_abs[x] >= 256){Accs[x] = "-" + String(Acc_abs[x],HEX);}; if (Acc_abs[x] < 256){Accs[x] = "-0" + String(Acc_abs[x],HEX);}; if (Acc_abs[x] < 16){Accs[x] = "-00" + String(Acc_abs[x],HEX);}; } if (Acci[x]>=0){ if (Acc_abs[x] >= 256){Accs[x] = "0" + String(Acc_abs[x],HEX);}; if (Acc_abs[x] < 256){Accs[x] = "00" + String(Acc_abs[x],HEX);}; if (Acc_abs[x] < 16){Accs[x] = "000" + String(Acc_abs[x],HEX);}; } } if (x==0){ if (gyr[x]<0){ if (Gy_abs[x] >= 256){Gys[x] = "-" + String(Gy_abs[x],HEX);}; if (Gy_abs[x] < 256){Gys[x] = "-0" + String(Gy_abs[x],HEX);};</pre>	<p>//In the case of the IMU, since the readings can be negative or positive notice that the conversion is a little less simple. Also, one extra character is always present to be able to know the sign of the reading. If it is a positive reading then instead of a negative sign, that place would be replaced by a zero. Again, the purpose is to maintain a constant length of output string to reduce fluctuations in overall sampling rate.</p>

Table 7-3—Continued

<pre> if (Gy_abs[x] < 16){Gys[x] = "-00" + String(Gy_abs[x],HEX);}; } if (gyr[x]>=0){ if (Gy_abs[x] >= 256){Gys[x] = "-" + String(Gy_abs[x],HEX);}; if (Gy_abs[x] < 256){Gys[x] = "00" + String(Gy_abs[x],HEX);}; if (Gy_abs[x] < 16){Gys[x] = "000" + String(Gy_abs[x],HEX);}; } } if (x!=0){ if (gyr[x]<0){ if (Gy_abs[x] < 256){Gys[x] = "-" + String(Gy_abs[x],HEX);}; if (Gy_abs[x] < 16){Gys[x] = "-0" + String(Gy_abs[x],HEX);}; } if (gyr[x]>=0){ if (Gy_abs[x] < 256){Gys[x] = "0" + String(Gy_abs[x],HEX);}; if (Gy_abs[x] < 16){Gys[x] = "00" + String(Gy_abs[x],HEX);}; } } } </pre>	
<pre> for(int i =0; i<3; i++){ A_str_tot = A_str_tot + Accs[i]; G_str_tot = G_str_tot + Gys[i]; } </pre>	<p>//Table 7-6 shows the format of the strings with the readings from the accelerometer and gyroscope in the 6DOF IMU Card.</p>
<pre> String Tot_string = V_tot_string + A_str_tot + G_str_tot; </pre>	<p>//Complete output string with the format as shown in Table 7-5 and Table 7-6 is created by concatenating the analog voltage in string to the one of the IMU board readings (//The complete string to be sent to the LabVIEW user interface would have the format shown in the Table below.</p>

Table 7-3—Continued

//Print String to Arduino USB port //Serial.println(Tot_string);	//notice it is commented out; this line was used when the code was being tested without the Xbee through the USB port.
//Print to Xbee XBee.println(Tot_string);	//Print string to the serial connection to the Xbee antenna
delay(16); } }	//This delay is an effort to maintain the sampling rate of the system consistent. The sampling rate without the delay function is about 40Hz. But even using the delay function (which input is in milliseconds) the overall sampling rate still fluctuates.

Table 7-4 Comparison between Decimal and Hexadecimal Strings

# of Characters in HEX String	DEC	HEX
1	1	1
2	16	10
3	256	100
4	4096	1000
5	65536	10000
6	1048576	100000

Table 7-5 Analog Voltage output string format & reading example

Axial Load												Torque						FSR					
A0			A1			A2			A3			A4			A5			A6			A7		
2	8	3	1	2	D	3	0	B	2	3	C	3	A	B	1	5	C	3	8	4	2	7	3

Table 7-6 Output string format & reading example of 6DOF IMU card readings

Accelerometer												Gyroscope											
X				Y				Z				Pitch				Roll				Yaw			
-	A	1	0	0	4	3	8	-	1	B	2	0	1	C	1	0	2	F	4	-	3	7	E

User Interface

The user interface was created to monitor the readings of the walker and visualize them graphically and numerically. The main screen includes two controls to define the VISA inputs. One is to define the input from the Walker's Arduino and the other from the Belt's Arduino.

The interface, as shown in Figure 7-6 and Figure 7-7 has a main block with an "Initialize" tab and a "DAQ" tab. The first one serves mainly to visualize the initial voltage read from the axial load and torque sensors and use them as reference voltages for when the calibration curves are applied to the signals during the DAQ portion. Also, this tab is where the file path of the .txt file where the data is to be stored is defined. After the reference voltages are set the user must press the "continue" button and go to the "DAQ" tab where all the physical measurements are visualized. After the "continue" button is pressed the system starts storing the read data.

In the right side of the screen, two waveform charts have been set to show the readings from the accelerometers in the walker's and belt's IMU. Also, numerical indicators were placed to show not only the readings from the accelerometer but also the gyroscope. The "STOP" button in the upper right corner of the screen stops the data acquisition at any point. The code was made making sure that if the data acquisition process is broken inappropriately, the data until that point is still stored in the .txt file previously defined.

Initialization Tab

This tab, as mentioned above, serves two main purposes, to have initial readings of the sensors and to define the file path where the data is to be stored.

The initial readings are necessary in order to verify that readings are being taken from all sensors prior to start the data storage and if the voltage readings from the

sensors that use Wheatstone bridges (axial load and torque sensors) are not within the voltage range set for the Arduino (in this case [0-1100 mV]), it allows the user to adjust the zeroing potentiometers to bring the reference voltage within the proper voltage range.

The recommended reference voltage for the axial load sensors is 100 mV because it lets us know that the minimum required voltage of the amplifier has been met, allows some room for tension readings and gives a range of 1000 mV for compression readings (which is the main purpose of the sensors). For the torque sensors the reference voltage recommended is in the middle of the voltage range (550 mV) in order to have the same range of measurement for both directions of torque applied to the handles.

In this initial tab the user can input his/her weight to later on have a percentage of the body weight being felt by the walker. This of course can be done with the stored data once the measurement is taken but it is a useful indicator to have while the walker is being used.

This Tab also allows knowing the sampling rate of each input prior to the start of the data acquisition and lets the user to troubleshoot in case there are connectivity problems. Another indicator that has been kept on this Tab that was of much help during the development of the code is the "String Reading" indicator. This indicator shows exactly the string that is being read through the VISA resource. Below this indicator is an array on integers created from the read string and below this indicator the voltage readings from axial load, torque and FSR sensors.

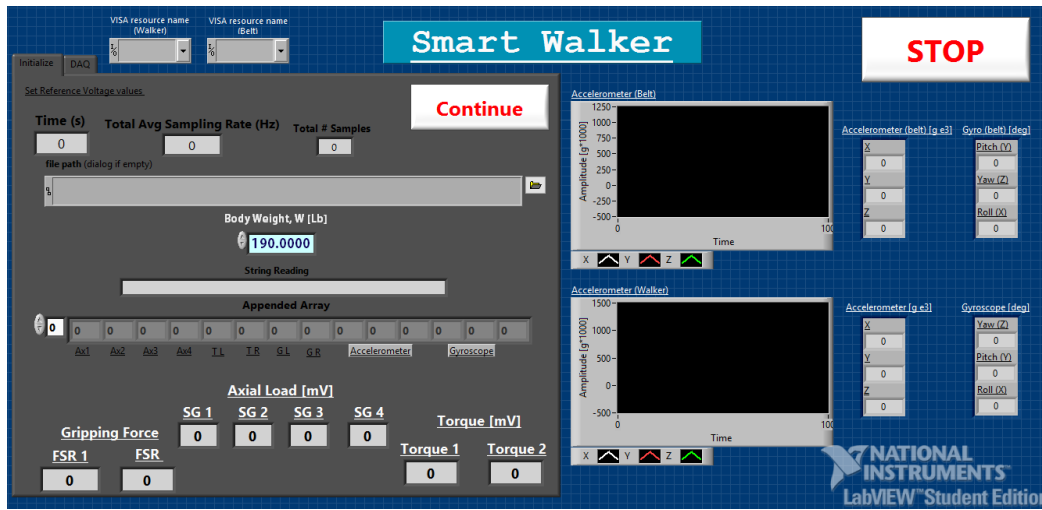


Figure 7-6 Smart Walker User Interface ("Initialize" Tab)

Data Visualization Tab

This tab is the one where all the measured forces can be monitored through bar and numerical indicators as well as waveform charts for the accelerometer readings.

In this screen the time of measurement (in seconds), the average sampling rate of each channel and the total number of samples are indicated.

The sensor indicators are organized according to the location of the sensors in the walker. The right and left side of the UI include the axial load readings in a numerical indicator from the strain gage on each leg (see burgundy numerical indicators in Figure 7-7). Moreover, the axial load readings from each side are averaged and shown in a bar indicator to display the average axial load applied on each side of the walker (see blue bar indicators in Figure 7-7). Also, it is shown the percentage of the total force applied on each side of the walker. Furthermore, these two averages are then added to display the overall weight of the user on the walker (see the red bar indicator in the middle of the block and the numerical indicator below the bar indicator). Finally, regarding the axial load measurement the percentage of the body weight being felt by the walker is shown in a numerical indicator as well.

Both, gripping force and torque applied on each handle are shown in bar indicators as well (see green and yellow indicators respectively). The torque indicators show positive readings when the torque is applied to the right side of the user and negative readings when the torque is applied to the opposite direction.

Waveform charts and numerical indicators are shown to indicate readings from the IMU in the walker and in the belt worn by the user. Accelerometer readings in X, Y and Z directions are shown where Z is normal to the floor, X in the forward direction and Y to the right side of the walker according to the right-hand rule. Pitch, yaw and roll readings from the gyroscope are shown only as numerical indicators pitch as rotation of Y axis, yaw rotation of the Z axis and roll of the X axis.

When the measurement is done the stop button is to be pressed to stop reading and storing data from the VISA ports.

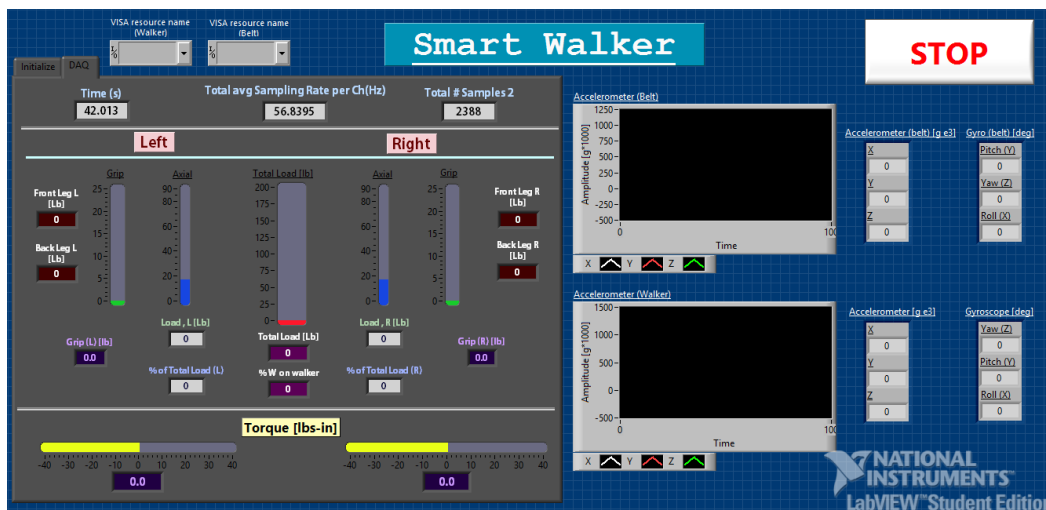


Figure 7-7 Smart Walker User Interface ("DAQ" Tab)

LabVIEW Code

This section will explain the LabVIEW code to read, interpret, show and store the data sent by the Arduino through the VISA ports. The code consists of two major sections in a flat sequence structure, as shown in the previous section. The first sequence, linked to the “initialize” serves the purposes explained previously and shown as section “A” in Figure 7-8. The second sequence is linked to the “DAQ” tab and IMU indicators in the right side of the screen shown as sections “B” and “C” in Figure 7-8. Section “B” contains the code for the walker sensors and section “C” contains the one for the Belt IMU.

The purpose of this section is to show the LabVIEW code developed to create this user interface and show each one of the written subVIs. Each subVI will be explained in individual sections and will contain subsections with internal subVIs. Various subVIs are repeated throughout the code. The first appearance will be explained and following appearances will refer to the section where it first appeared. If a subVI is a variation of a previously explained subVI, it will be described superficially and will refer the reader to the previously explained subVI.

The Arduino board is programmed to read each sensor connected to it and send read values through the Xbee antenna continuously. The LabVIEW code obtains these readings sent to the Xbee antenna connected to the USB port of the computer, separates, interprets, shows and records these readings.

Before starting, the user is asked to determine the USB ports (VISA resource name) where the Xbee antennas for the Walker and Belt readings are to be taken. The highest baud rate with stable readings is 57600 bits/sec and has been set both in the Arduino and LabVIEW codes. The initialize screen allows the user to set the reference voltage for the strain gage sensors, test that the other voltages are being read and select the file path where the data is to be stored. After the voltages are in the proper range, the

user weight has been set and the file path the “Continue” button is pressed to start the data acquisition and storage.

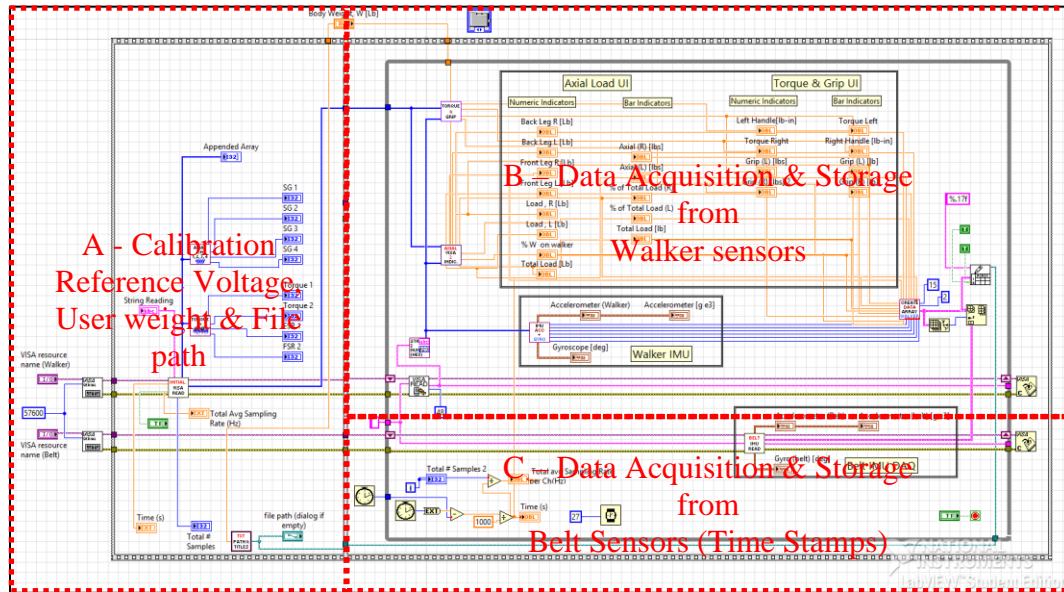


Figure 7-8 Main Smart Walker LabVIEW block diagram

Once the setup is done in section A, Section B & C (second part of the flat sequence) reads the data, shows it in the user interface and stores it in the file set in Section A. When the data acquisition is finished, the STOP button is to be pressed to end data acquisition.

A – Calibration Reference Voltage

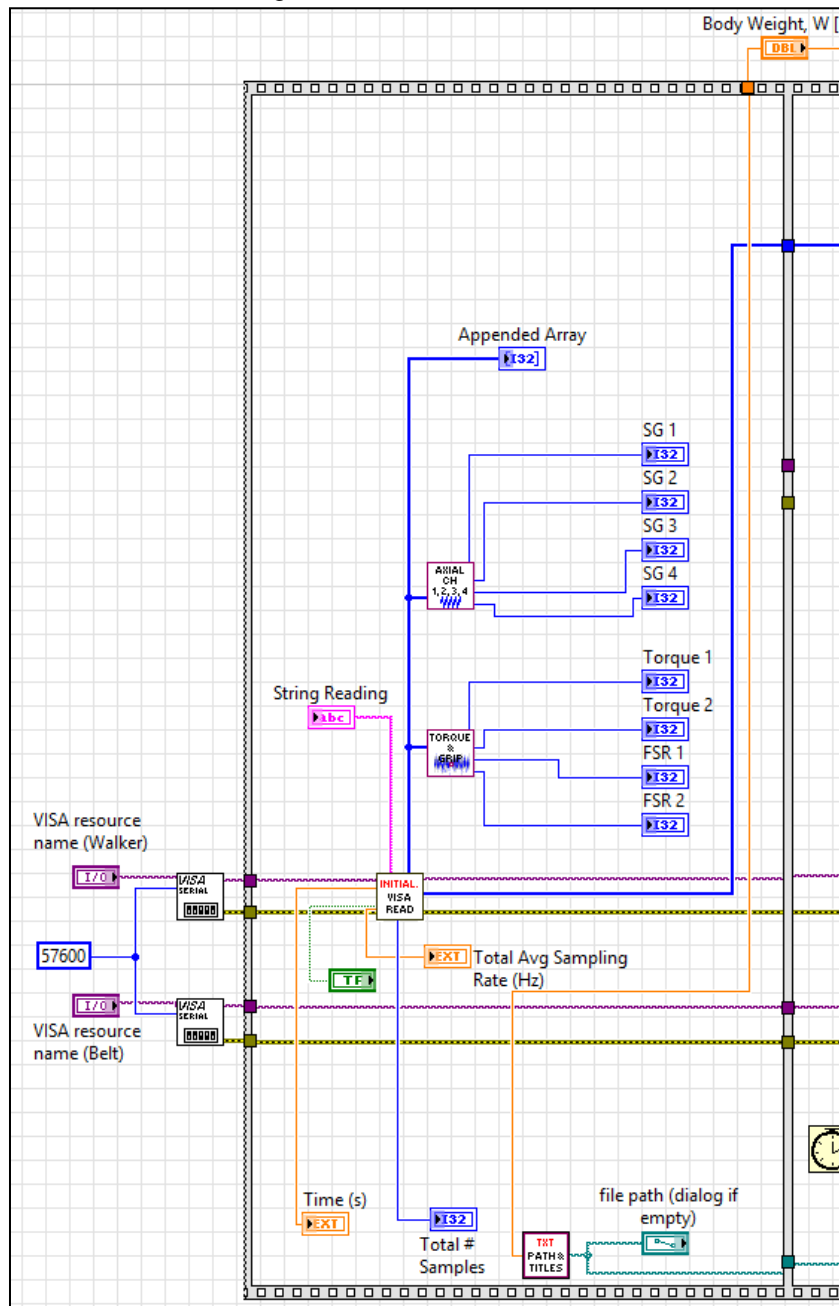


Figure 7-9 Smart Walker main code (A: Calibration Reference Voltage)

The following main tasks are performed in this section of the code shown in

Figure 7-9:

1. VISA serial ports are configured.
 - a. User interface control allows user to set VISA resource names.
 - b. Baud rate is set to 57600 bit/sec.
2. Walker VISA serial port configuration connects to Initial VISA Read SubVI.
 - a. Outputs total average sampling rate
 - b. Time (in seconds)
 - c. Total samples acquired
 - d. Readings from the Arduino board (both, raw string being sent to serial port and array of integers after hexadecimal to decimal conversion).
 - i. The appended array of integers is separated in individual indicators to show individual voltage readings for each sensor.
3. User weight is input (in pounds).
4. File path where data is to be stored is set.
 - a. The title to the text file and to the arrays for the readings of each sensor is created.
5. After "Continue" button is pressed the following is sent to the next flat sequence event:
 - a. Voltage readings are sent to be the reference voltages in the calibration curves
 - b. File path where created .txt file is located
 - c. VISA port configuration cables

Initial Visa read

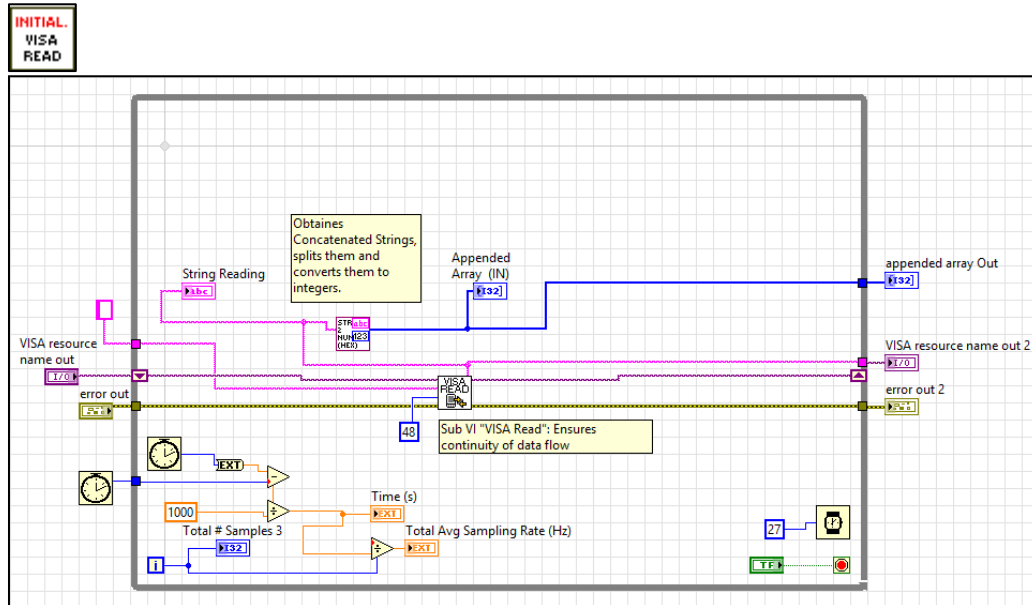


Figure 7-10 Initial VISA read SubVI (Initialize)

Initial Visa Read subVI shown in Figure 7-10 is the main component to acquire and monitor analog voltage readings using the Arduino Board and LabVIEW. In this subVI all the component to read data through the VISA ports are present as well as other additions to monitor performance and assist for debugging purposes.

The initial Visa Read Sub VI is a while loop that on each iteration obtains the readings at the VISA port, interprets the hexadecimal characters, transforms them to integers and indicates the values. Moreover it includes indicators for number of samples taken, total average sampling rate and time elapsed since the start of the data acquisition process. The “while loop” is forced to wait 27 milliseconds (in this case) as an attempt to make sampling rate constant. The maximum sampling rate per channel was close to 40Hz when this code was developed and various “wait times” close to this sampling frequency were tried until the sampling rate behaved more stable; in this case the resulting sampling rate was close to 37Hz.

Visa Read

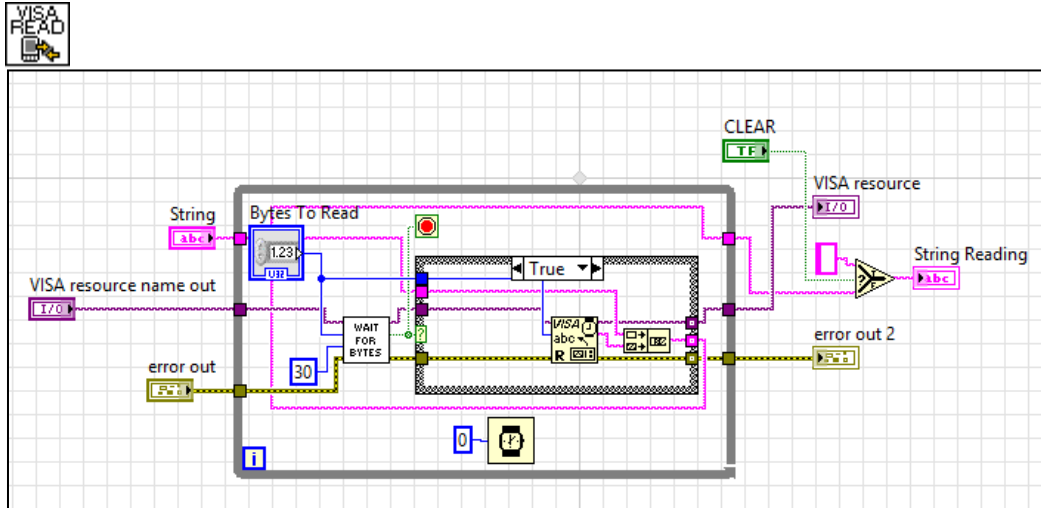


Figure 7-11 Visa Read SubVI

The Visa Read Sub VI, shown in Figure 7-11, shown above is the responsible of allowing reading the string coming from the Arduino board through the VISA resource port continuously and reliably. A main necessity that was added to this subVI was the “Wait for Bytes” subVI which solved the lack of fluidity of data. Before this subVI was added the data acquisition would occur for some time and crash eventually when no data was encountered at the VISA port.

The main components of the VISA read function block are present and a control for the number of bytes to read was added to allow for changes in case the string to be read would change.

Wait for bytes

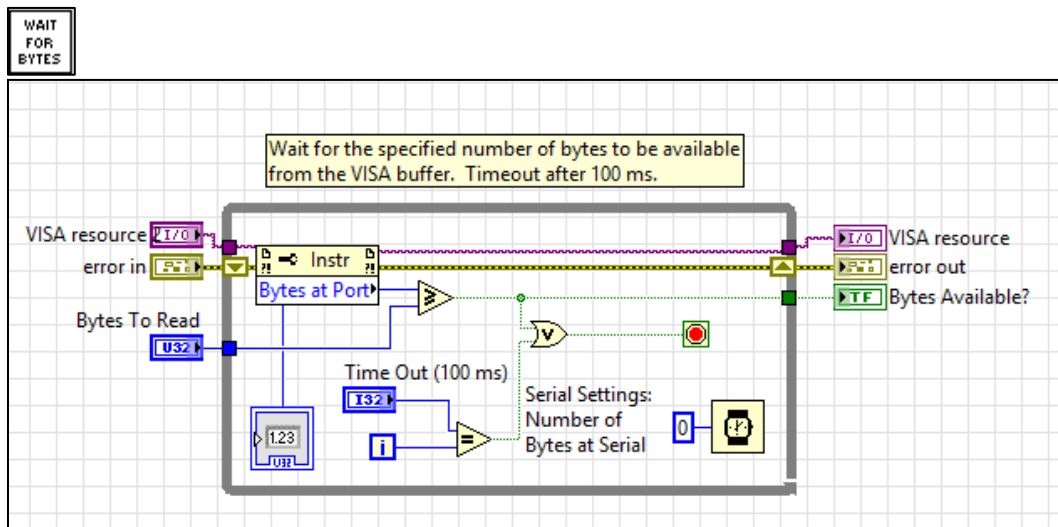


Figure 7-12 Wait for Bytes subVI

The Wait for bytes sub VI shown in Figure 7-12 allows the VISA Read subVI to – as the name indicates – wait until enough data is available to be shown and then proceed to read it. Prior to the addition of this subVI the program would crash every time not enough data was available to be read and the continuous data acquisition was impossible. The number of bytes expected to be read have to be indicated. The Time Out (time at which the process would stop if not enough data is encountered) period was defined as 100 ms, large enough compared to the looping period of the system (27 ms).

String to array of Integers

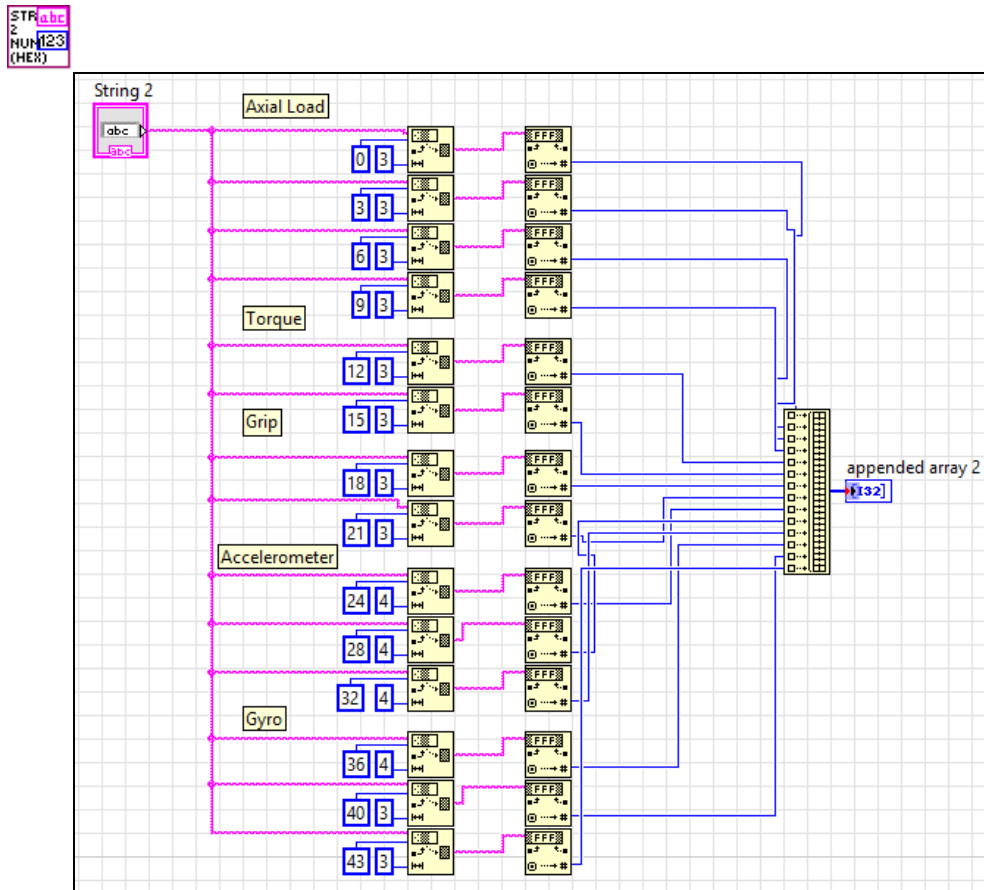


Figure 7-13 String to Array of Integers SubVI

As the name describes, the String to Array subVI shown in Figure 7-13 takes the original string sent by the Arduino, slices it according to the length of each sensor reading, converts the hexadecimal string to decimal integer and appends the integers in one big array of integers. In other word, this subVI is the interpreter from hexadecimal Arduino output to usable and visual decimal integers.

File Path

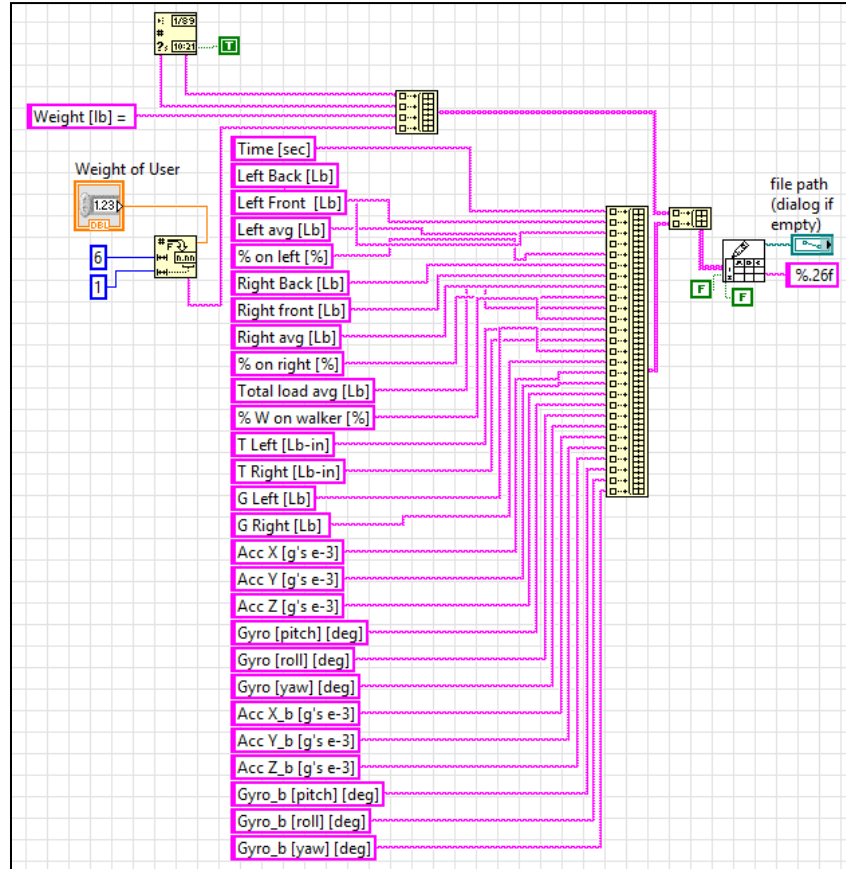


Figure 7-14 File Path and Data file title creation subVI

The file path subVI shown in Figure 7-14 creates the title of the text file where the data is to be stored. This title consists of the date and time of the beginning of the measurement and the weight of the user. Additionally it includes the title for each one of the columns of data stored. Each column represents one of the values indicated in the user interface.

Extract Integers from array (Axial Load)

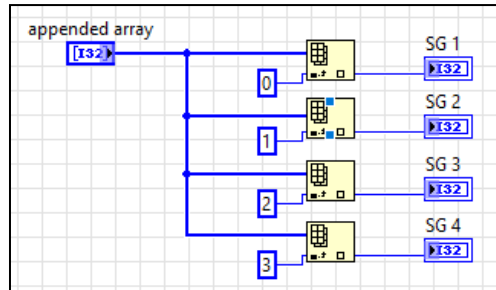


Figure 7-15 Extract Integers from array of integers (Axial load)

The “Extract Integers from array” subVI, shown in Figure 7-15, is a simple one with the only purpose of extracting the individual integer values needed from the appended array of integers created by “String to array of Integers” subVI. Here the one for axial load readings is shown, but throughout the code variations of this subVI are used for other sensors.

Extract Integers from Array (Torque & FSR)

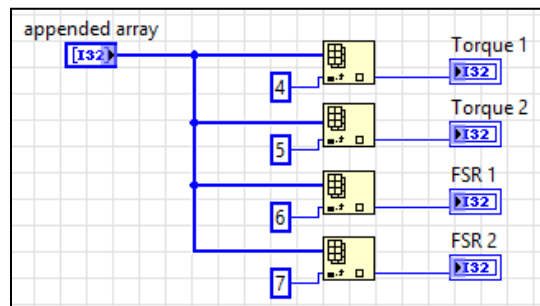


Figure 7-16 Extract Integers from array of Integers (Torque and FSR)

The Extract Integers from Array SubVI shown in Figure 7-16 is almost identical to the one shown in Figure 7-15 except that at other position of the array of integers.

B – Data Acquisition from Walker sensors & overall Data Storage

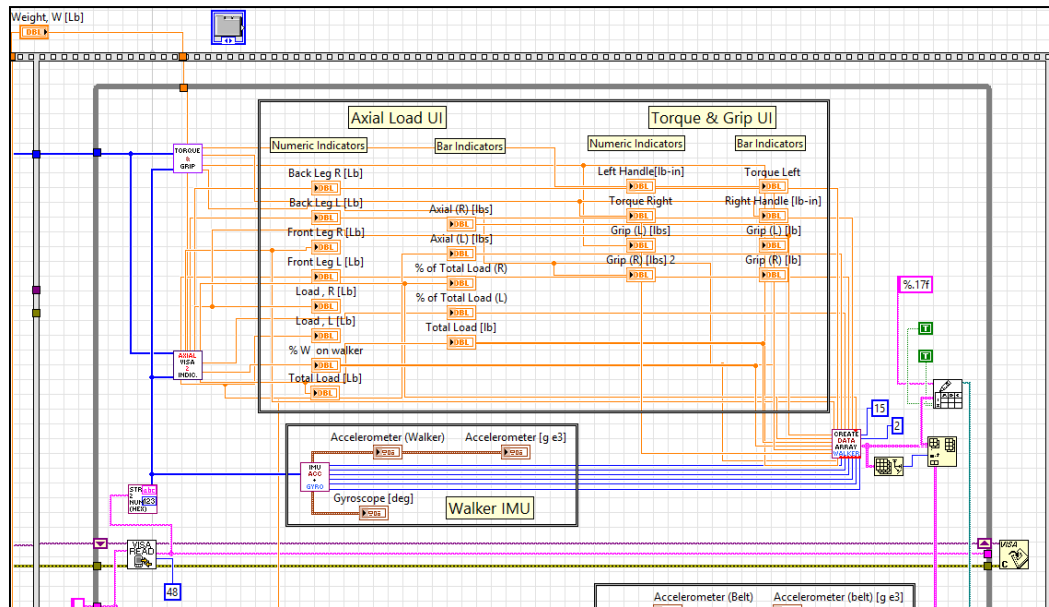


Figure 7-17 Smart Walker main code

(B – Data Acquisition from Walker sensors & overall Data Storage)

The code shown in Figure 7-17 flows as follows:

1. Reference voltage from section A of the code is used to apply calibration curves to voltage readings and show axial load, torque and gripping force measurements. The physical measurements shown are:
 - a. Axial Load:
 - i. Felt by the strain gage of each leg (numerical indicator)
 - ii. Axial load felt by each side of the walker (Average of front/back on each side)
 - iii. Overall applied to the walker and what percentage of body weight it represents.
 - b. Torque (both handles of the walker - numerical and bar indicators)

- c. Gripping force measurements (Both handles of the walker – numerical and bar indicators)
 - d. IMU readings in waveform chart and numerical indicators including
 - i. Acceleration in X,Y and Z axis
 - ii. Pitch, roll, yaw
2. Data is stored in the file created in part A
- a. Axial Load (each leg, average of each side, overall axial load, overall weight on the walker)
 - b. Torque from each side
 - c. Gripping force from each side
 - d. IMU from walker and belt

Torque and Grip VISA to Indicators

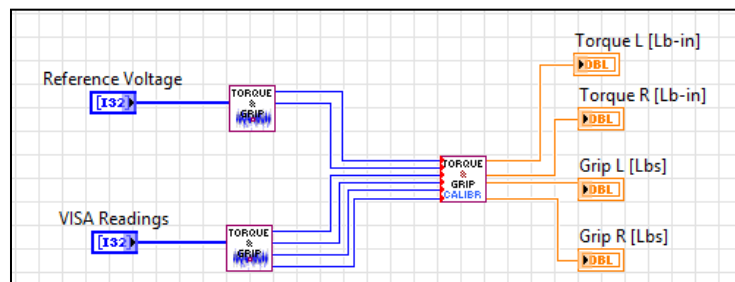


Figure 7-18 Torque and FSR VISA readings to UI Indicators subVI

The Torque and Grip VISA to Indicators subVI shown in Figure 7-18 serves as a connector to extract the relevant integers from the reference voltage array of integers and readings and use them to apply the calibration curves developed in Chapter 4 and Chapter 5.

Torque and FSR calibration curves

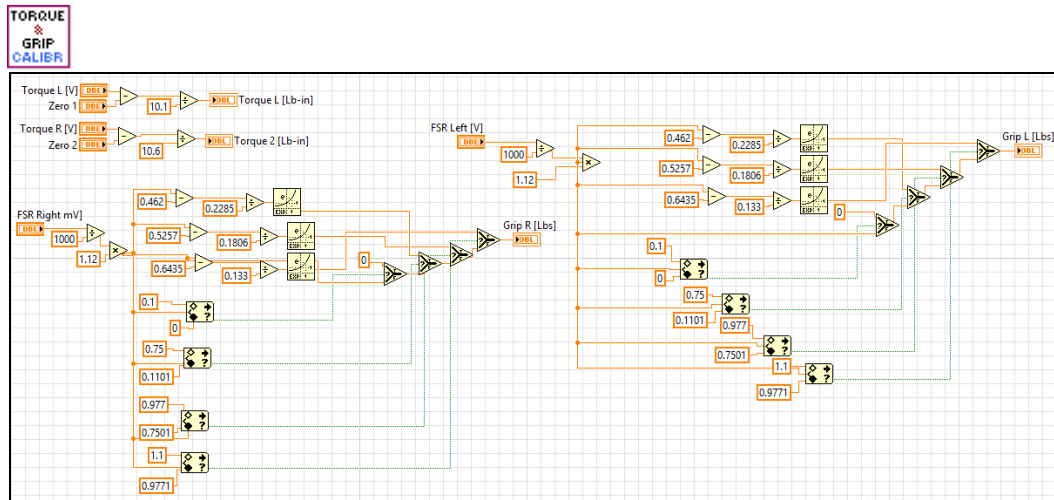


Figure 7-19 Torque and FSR calibration curves subVI

The Torque and FSR calibration curves subVI shown in Figure 7-19 applies the linear calibration curve to the torque voltage readings (Upper left corner) and the exponential calibration curve for the FSR sensors. The FSR calibration curve, in order to obtain a closer fit to the experimental results, was broken down to three main regions (as shown is Chapter 5).

Axial Load VISA to Indicators

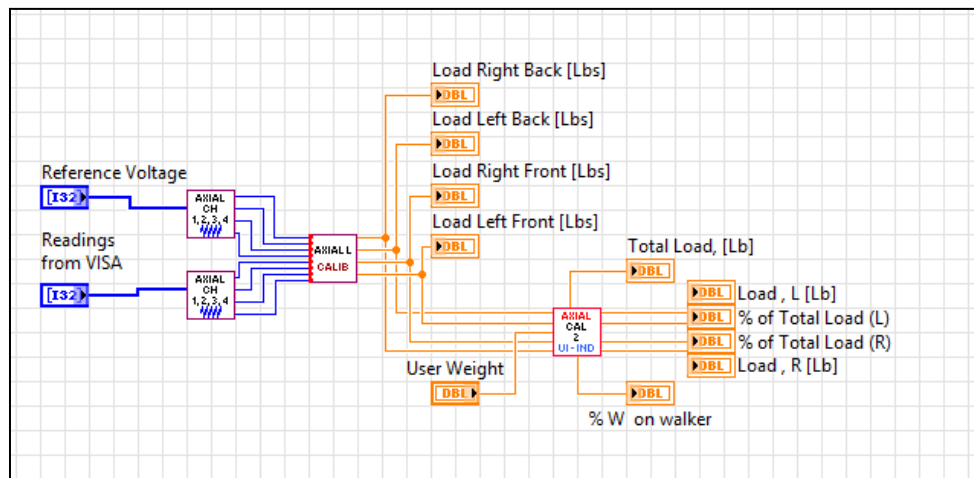


Figure 7-20 Axial load and VISA readings to UI Indicators subVI

Similarly to the section describing the torque and gripping force indicators before, the Axial Load VISA to Indicators section, shown in Figure 7-20 uses the reference voltage values and the continuous readings, applies the calibration curves developed in section Chapter 3 and indicates the physical values of the axial load applied to each strain gage. Moreover, additional calculations are performed to show the average axial load applied to each side, the overall axial load applied to the walker, percentage of total axial load applied to each side and percentage of user weight on the walker.

Axial load Calibration Curves

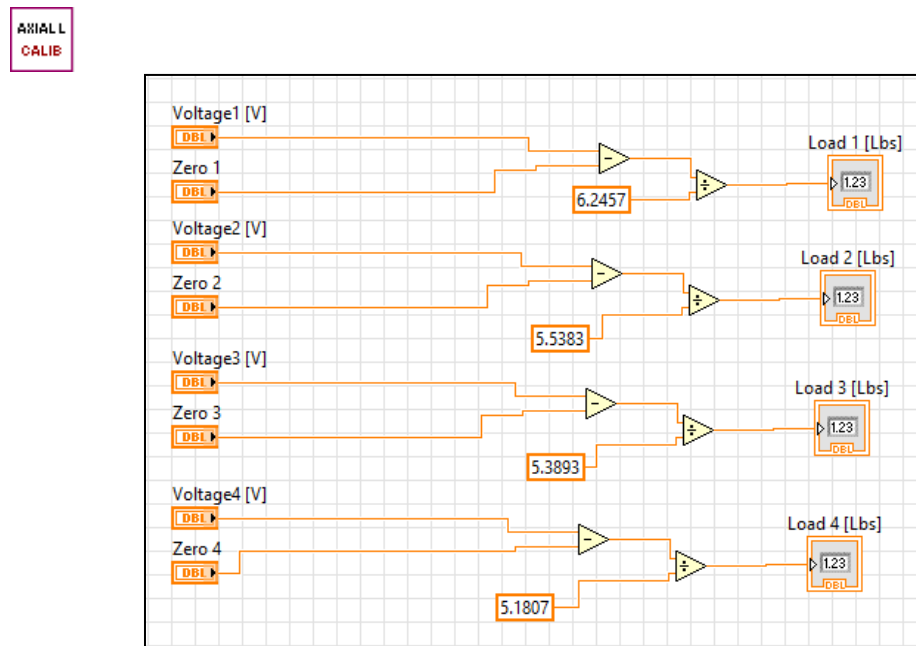


Figure 7-21 Axial Load Calibration Curves subVI

The Axial load Calibration Curves subVI shown in Figure 7-21 applies the calibration curves found in Chapter 3 to the samples sent by the Arduino.

Operations for User Interface Indicators

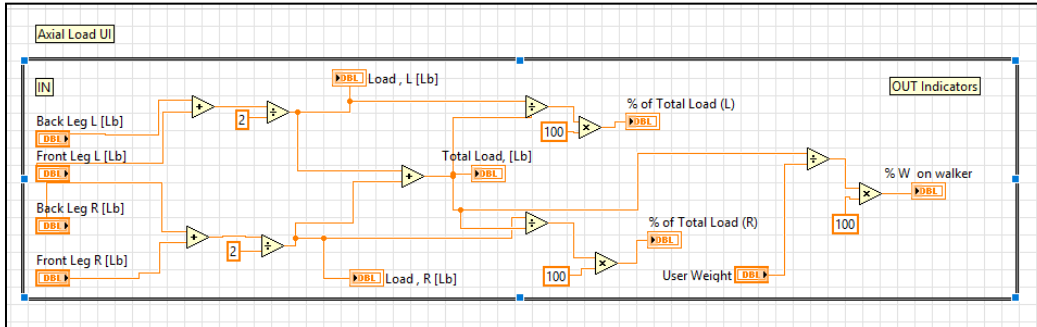


Figure 7-22 Average and percentage calculations for Axial load UI Indicators

The Operations for User Interface Indicators subVI shown in Figure 7-22 includes the average and percentage calculations needed for all the values going to the axial load indicators. Including:

- Average of left and right side axial load
- Overall axial load applied (addition of the left and right side averages)
- % of load on each side of the walker
- % of body weight applied to the walker

Walker IMU VISA to User Indicators

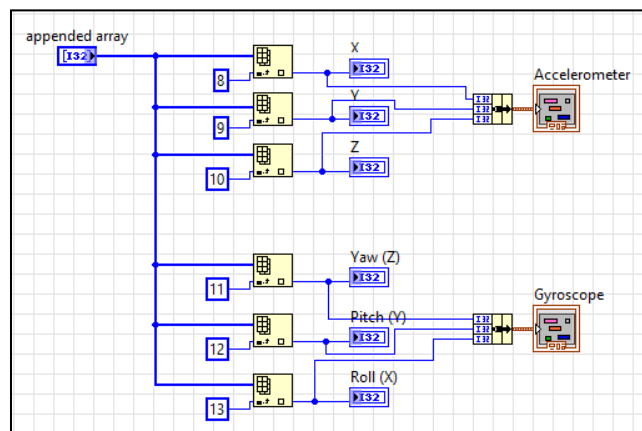


Figure 7-23 Walker IMU readings to clusters for waveform chart subVI

Since the IMU readings don't need to be calibrated because the Arduino libraries do all the work to output accelerometer and gyroscope readings, this subVI extracts the integers from IMU readings and clusters them in order to send them to a waveform chart as shown in Figure 7-23.

C – Data Acquisition & Storage from Belt Sensors (Time Stamps)

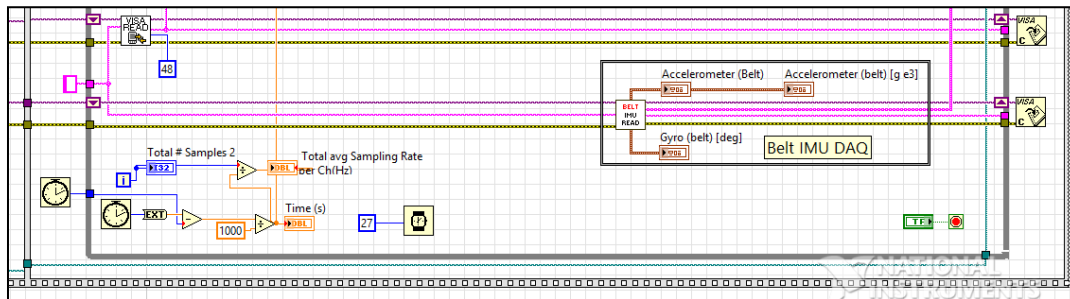


Figure 7-24 Smart Walker main code

(C – Data Acquisition & Storage from Belt Sensors (Time Stamps))

This section of the code is part of the data acquisition section of the two part flat sequence. Another VISA read subVI is implemented but only to read IMU values from an Arduino installed on the User's belt and be able to monitor the movement of the user simultaneously with the one of the walker.

Similarly to the Initial VISA Read subVI, this section includes the time elapsed starting at the moment of data acquisition/storage, average sampling rate and overall samples stored as shown in the lower left corner of Figure 7-24.

Belt IMU VISA to indicators

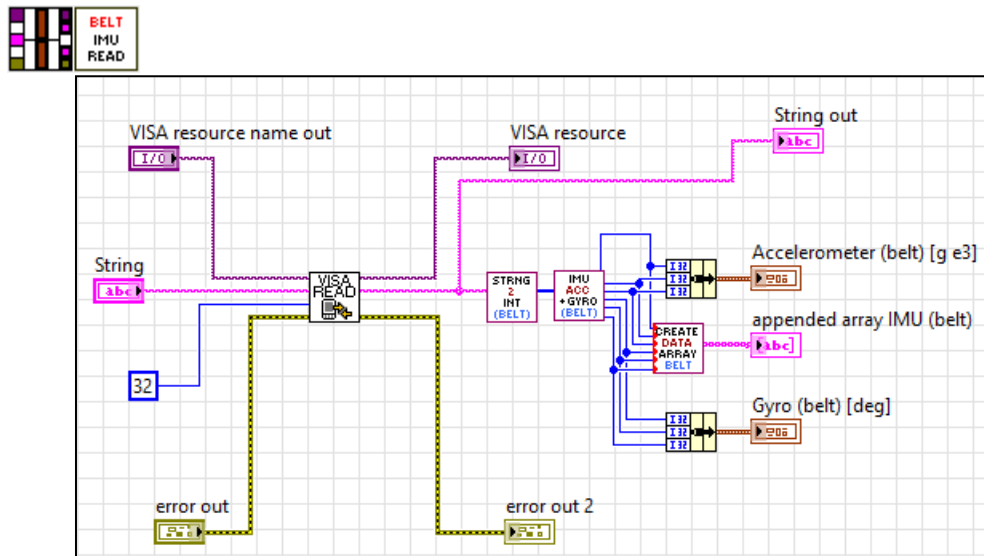


Figure 7-25 Belt Arduino VISA read & cluster to waveform charts

The Belt IMU VISA to indicators subVI shown in Figure 7-25 is a modification of previously mentioned subVIs but specifically written for the readings coming from the Arduino worn by the user. It essentially consists of:

1. Visa Read
2. String to integers
3. Integers to cluster
4. Cluster to waveform charts
5. Integers to data arrays to be sent to .txt file

Create String Array to be stored in created .txt file

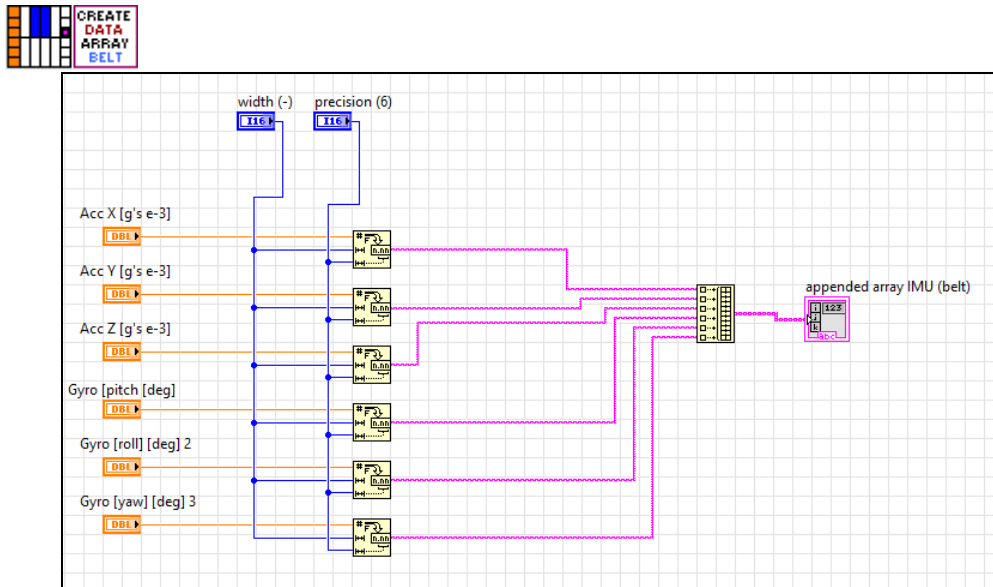


Figure 7-26 String array from integers to be stored in .txt file

For simplicity the subVI shown in Figure 7-26 has been used to present the code used to compile the readings from the sensors, convert them to strings and append them in array of strings so they can be sent to a .txt file in an organized manner and create columns of data. A similar subVI was used for the Walker portion in section B of the code and looks very similar to this except that with more input and a bigger output array of strings.

Data Visualization

The resulting text file resulting from the data acquisition looks like is shown in

Figure 7-27 (more columns to the right are present).

4/24/2015	4:29:17 PM	Weight [lb] = 190.0								
Time [sec]	Left Back [Lb]	Left Front [Lb]	Left avg [Lb]	% on left [%]	Right Back [Lb]	Right front [Lb]	Right avg [Lb]	% on right [%]	Total	
0.00	0.00	0.19	0.10	-126.46	-0.16	-0.19	-0.17	226.46	-0.08	
0.04	0.18	-0.39	-0.10	100.00	0.00	0.00	0.00	-0.00	-0.10	
0.07	0.36	0.00	0.18	-3616.98	0.00	-0.37	-0.19	3716.98	-0.00	
0.12	0.00	-0.19	-0.10	20.64	0.00	-0.74	-0.37	79.36	-0.47	
0.15	-0.36	0.00	-0.18	63.12	0.16	-0.37	-0.11	36.88	-0.29	
0.20	-0.36	-0.19	-0.28	74.92	0.00	-0.19	-0.09	25.08	-0.37	
0.23	-0.36	-0.19	-0.28	74.92	0.00	-0.19	-0.09	25.08	-0.37	
0.28	0.00	-0.39	-0.19	26.65	-0.32	-0.74	-0.53	73.35	-0.72	
0.31	0.00	0.00	0.00	-0.00	0.00	-0.74	-0.37	100.00	-0.37	
0.36	0.18	-0.19	-0.01	2.71	0.48	-0.93	-0.22	97.29	-0.23	
0.39	0.00	0.19	0.10	54.66	0.16	0.00	0.00	45.34	0.18	
0.44	0.00	-0.77	-0.39	50.99	0.00	-0.74	-0.37	49.01	-0.76	
0.47	0.18	-0.39	-0.10	18.55	-0.16	-0.74	-0.45	81.45	-0.55	
0.52	-0.18	0.19	0.01	-2.40	-0.16	-0.37	-0.27	102.40	-0.26	
0.55	-0.18	-0.39	-0.28	51.61	-0.16	-0.37	-0.27	48.39	-0.55	
0.60	0.00	-0.39	-0.19	31.20	-0.48	-0.37	-0.43	68.00	-0.62	
0.63	0.36	-0.19	0.08	-24.60	-0.48	-0.37	-0.43	124.60	-0.34	
0.68	0.00	0.00	0.00	-0.00	-0.32	-0.19	-0.25	100.00	-0.25	
0.71	0.00	-0.19	-0.10	17.62	-0.16	-0.74	-0.45	82.38	-0.55	
0.76	0.36	-0.39	-0.01	2.62	0.00	-0.93	-0.46	97.30	-0.48	
0.79	0.00	-0.19	-0.10	26.65	-0.16	-0.37	-0.27	73.35	-0.36	
0.84	0.00	-0.19	-0.10	47.78	0.16	-0.37	-0.11	52.22	-0.20	
0.87	-0.18	-0.19	-0.19	42.48	-0.32	-0.19	-0.25	57.52	-0.44	
0.92	0.36	-0.19	0.08	-22.89	-0.16	-0.74	-0.45	122.89	-0.37	
0.95	0.36	-0.19	0.08	-94.67	-0.16	-0.19	-0.17	194.67	-0.09	
10.00	0.18	0.19	0.19	107.31	0.16	-0.19	-0.01	-7.31	0.17	
10.03	0.00	0.00	0.00	-0.00	-0.16	0.00	-0.08	100.00	-0.08	
10.08	0.18	-0.19	-0.01	3.25	0.00	-0.37	-0.19	96.75	-0.19	
10.11	0.00	-0.19	-0.10	50.99	0.00	-0.19	-0.09	49.01	-0.19	
10.16	0.00	0.00	0.00	-0.00	0.00	-0.19	-0.09	100.00	-0.09	
10.19	0.36	0.19	0.28	100.00	0.00	0.00	0.00	0.00	0.28	
10.24	0.18	-0.77	-0.30	63.12	-0.16	-0.19	-0.17	36.88	-0.47	
10.27	0.18	0.39	0.10	18.55	-0.16	-0.74	-0.45	81.45	-0.55	

Figure 7-27 Stored data from data acquisition (.txt file)

After importing the data a table like the one below can be obtained. The table showing first couple of samples acquired has been divided in Table 7-7 to Table 7-9 to fit the width of the page.

First the section of the data regarding axial load measurement is shown, then torque & gripping force sensors and finally IMU readings from Walker and user belt.

Table 7-7 Acquired data regarding Axial load measurement

Time [sec]	Left Back [Lb]	Left Front [Lb]	Left avg [Lb]	% on left [%]	Right Back [Lb]	Right front [Lb]	Right avg [Lb]	% on right [%]	Total load avg [Lb]	% W on walker [%]
0	-0.36	2.32	0.98	84.05	0	0.37	0.19	15.95	1.16	0.61
0.05	-1.26	3.86	1.3	49.99	0	2.6	1.3	50.01	2.6	1.37
0.1	-1.08	4.44	1.68	53.06	0	2.97	1.48	46.94	3.16	1.66
0.14	-0.9	4.63	1.86	64.63	0	2.04	1.02	35.37	2.89	1.52
0.19	-0.72	5.4	2.34	69.64	0	2.04	1.02	30.36	3.36	1.77
0.24	-2.17	6.18	2.01	59.03	0	2.78	1.39	40.97	3.4	1.79

Table 7-8 Acquired data regarding Torque and Gripping Force measurement

Time [sec]	T Left [Lb-in]	T Right [Lb-in]	G Left [Lb]	G Right [Lb]
0	0.99	1.6	0	0.41
0.05	0.5	0.38	0	0.42
0.1	0	-0.75	0	0.42
0.14	0.3	-0.19	0	0.42
0.19	0.4	-0.09	0	0.43
0.24	0.4	-0.75	0	0.43
0.29	0	-0.94	0	0.43

A standard code was written in MATLAB to be able to visualize the data after it has been acquired. Plots obtained are shown in Figure 7-28 to Figure 7-30.

It is noticed that if long periods of data acquisition are needed some of the columns that are derivative from the main measurements can be omitted, thus producing a smaller size data file, and the information then can be computed after the data has been acquired. Columns such as the one showing average axial load on each side, or percentage load on each side can be taken out and computed in MATLAB if needed.

Table 7-9 Acquired data regarding IMU readings from Walker

Time [sec]	Acc X [g's e-3]	Acc Y [g's e-3]	Acc Z [g's e-3]	Gyro [pitch] [deg]	Gyro [roll] [deg]	Gyro [yaw] [deg]
0	669	119	811	40	39	0
0.05	657	65	742	40	39	0
0.1	638	3	807	40	39	0
0.14	642	0	784	40	39	0
0.19	646	-3	773	40	39	0
0.24	650	-7	761	40	39	0
0.29	634	-7	780	40	39	0

Table 7-10 Acquired data regarding IMU readings from Belt

Time [sec]	Acc X_b [g's e-3]	Acc Y_b [g's e-3]	Acc Z_b [g's e-3]	Gyro_b [pitch] [deg]	Gyro_b [roll] [deg]	Gyro_b [yaw] [deg]
0	-7	30	961	0	-117	1
0.05	-7	23	969	0	-117	1
0.1	-11	30	965	0	-117	1
0.14	-7	26	961	0	-117	1
0.19	-7	26	969	0	-117	1
0.24	-7	34	969	0	-117	1
0.29	-7	26	965	0	-117	1

The IMU readings figure & code would be the same for the readings coming from the walker's IMU and from the user belt's IMU.

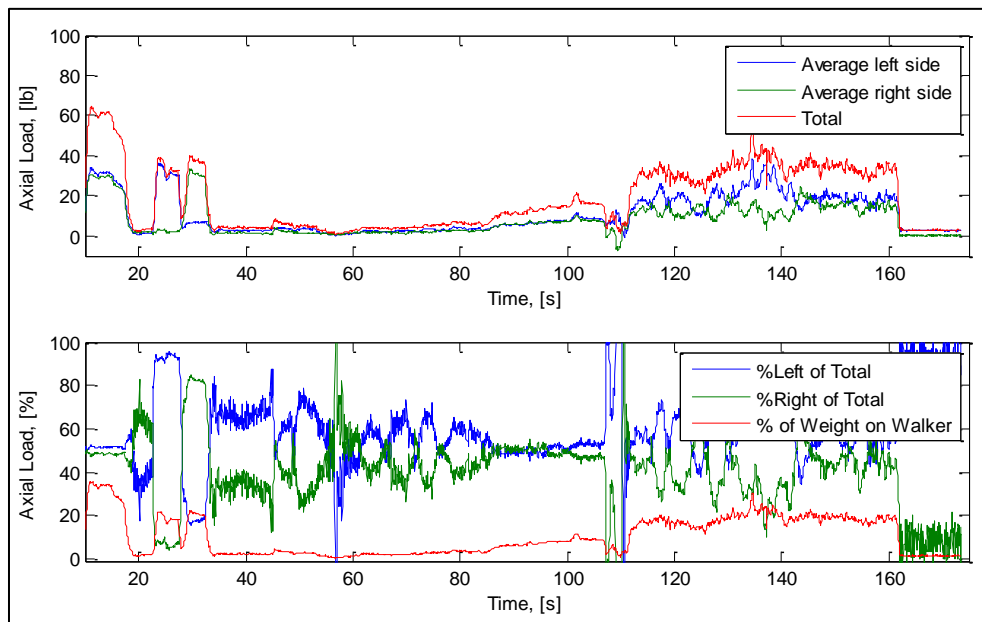


Figure 7-28 Axial Load Readings

Figure 7-28 shows the results for axial loading. It can be seen that from seconds 20 to 40 a load to each side of the walker was applied. Then the user walked normally

using the walker and at about second 110 the user started applying bigger forces purposely.

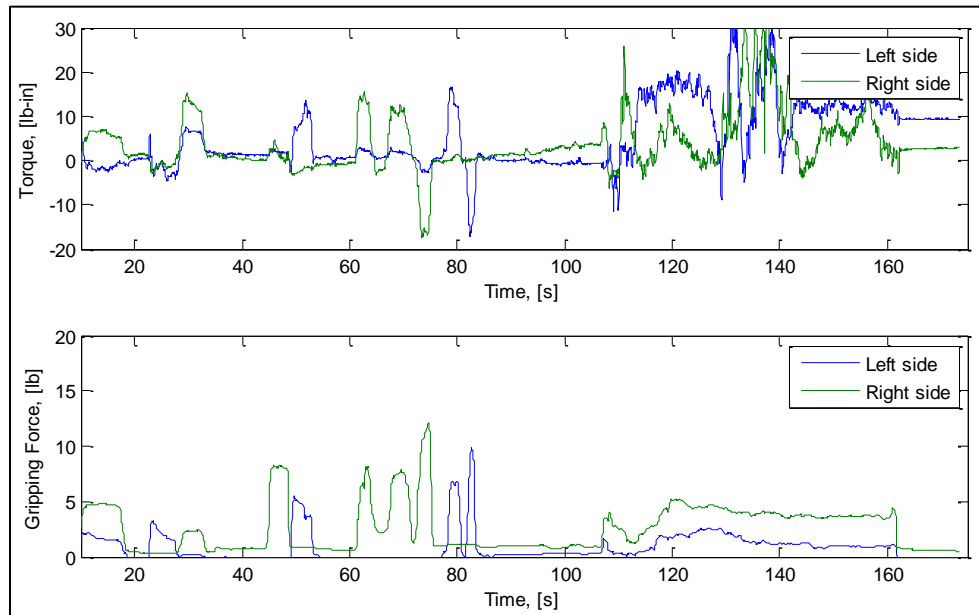


Figure 7-29 Torque and Gripping Force Readings

Figure 7-29 shows the torque and gripping force sensor readings. The user continuously applied torque to both sides in both directions. Similarly the gripping force was tested and gave reasonable results according to the user's experience.

Figure 7-30 shows accelerometer readings. Notice that the acceleration has fluctuations of a little bigger amplitude after the user starts walker normally. When the user starts applying more force, it shakes the walker with more intensity than before and it is felt by the accelerometer. Notice that the accelerometer during this test was not fixed in a specific direction so it shown readings in it x, y and z axes from the vibration of the whole IMU board. The gyroscope readings are also shown. See that what is plotted as pitch is the one direction that is not maintained reliably after the user starts to walk and increases although the user had not changed its orientation. At about second 120 the

user rotates in what is plotted as pitch. The pitch, roll and yaw can be defined according on the testing necessities.

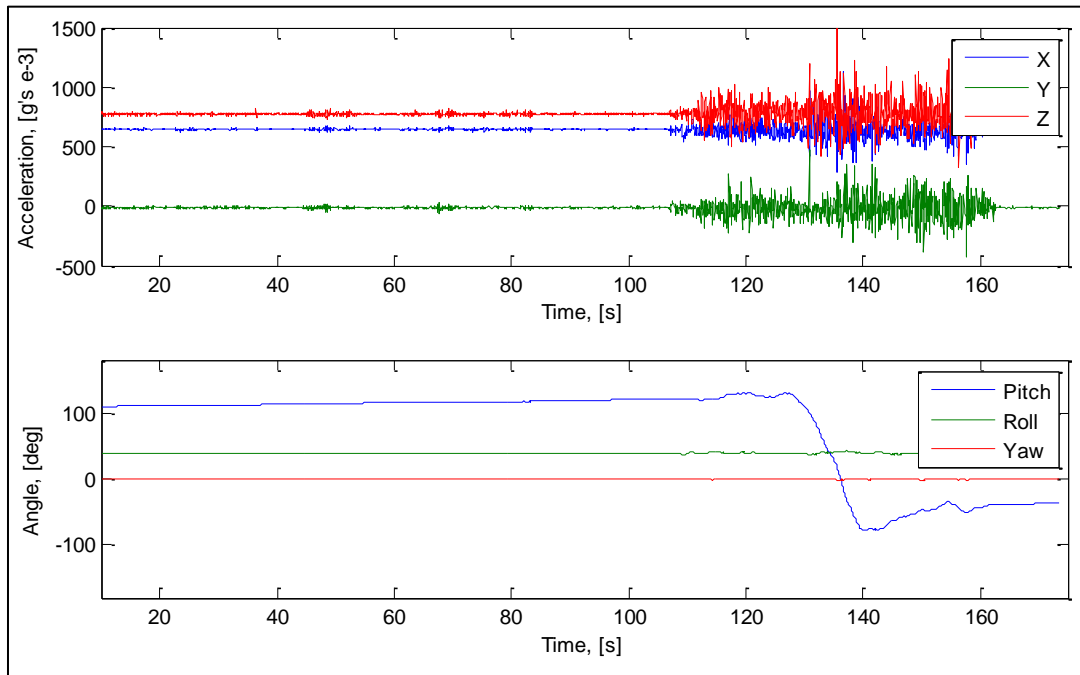


Figure 7-30 IMU Readings from the walker

Chapter 8

Conclusions

The smart walker project is a device that permits walker users and therapists to observe and store readings of the forces applied to a rolling walker during its use. The visualization of the forces applied to the walker help to determine and quantify usage and posture problems that lead to the falling of patients. The goal of reliably acquire digital readings of the main relevant forces applied to the walker was achieved by using reasonably priced and commonly used components, devices and software was successfully achieved.

Essentially it was necessary to know the axial load applied to the walker and its distribution, the torque and the gripping force applied to the handles. Moreover it was necessary to know when and how it is moving, reason why an IMU was included in the design. Finally, the relationship between the forces and the movement of the patient itself needed to be correlated so another IMU was used to be placed at the patient's belt.

The axial load was read by the use of 1-axis metallic foil strain gages. One strain gage was carefully placed on each leg of the walker at a location carefully studied with a FEM model of the walker. It is worth mentioning that the results obtained from the FEM study and the strain gage readings agreed very closely. The torque was also measured using strain gages, in this case, 2 axis strain gages specially designed with patterns perpendicular to each other and 45deg from the horizontal. These patterns allow setting 2 2-axis strain gages in a full bridge configuration that measures torque of a circular shaft (walker handle) reliably and without the need of high amplification.

The gripping force was measured by using an FSR sensor. Similarly to strain gages, this sensor allows quantifying the gripping force by measuring the voltage output due to resistance changes. Contrarily to strain gages, FSRs experience resistance

changes from infinity to – in this case – 200 Ω , resistance at which the maximum measurable force is reached. This is not a linear sensor, its obtained calibration curve was better fitted with three logarithmic curves (Chapter 5). This sensor is not very accurate, but it does allow us to quantify the intensity at which the patient is grabbing the handle. The circuitry involved to setup this sensor is very simple. This sensor basically acts like a valve that allows current flow (the more it is pressed, the less resistance it has and the more voltage is felt). The “valve” is followed by a voltage divider to reduce the maximum voltage output from 5 V (voltage source) to around the limit set in the Arduino program (1.1 V).

All these sensors were read and sampled with the Arduino MEGA board, a very commonly used piece of equipment. The Arduino environment helped immensely to develop the project, not only because its language is very intuitive but also because of the massive amount of resources in the Arduino community offered that help solving very quickly most of the problems encountered. Additionally, there is a plethora of products that are compatible and easy to install to the Arduino, like the Xbee antenna and related accessories. The Arduino MEGA has proven itself to be a very useful device to be used for data acquisition for sampling rates between 20 and 1000 Hz. Because of the fact that it only has 1 A2D converter that is shared by all of its analog inputs, the more inputs it has, smaller is the maximum achievable sampling rate. For the case of the Smart Walker, a sampling rate of 30 Hz to 60 Hz was enough for the therapist to observe and study the patient's use of the walker.

Another main requirement for this project was to ensure a reliable wireless connection between the Smart Walker system and the computer where the data was to be observed and stored. The Xbee antenna served this purpose perfectly, allowing the Smart Walker to have a reliable constant connection to the computer with a very useful

range for the application. The monitor station could be placed up to 100 ft. of the walker (indoor) and 300 ft. outdoor ensuring a fluid, secure and reliable connection.

Finally the system was able to be monitored with a user interface developed in LabVIEW that received the data sent by the Arduino through the Xbee antennas. LabVIEW allowed to read and interpret the data, showed the data in an easy-to-visualize manner and stored it to be able to analyze it afterwards. The use interface showed the axial load and its distribution on the walker, the percent body weight on the walker, torque and gripping force measurements and IMU readings from the walker and belt of the user.

Future work planned after this prototype was completed was to develop a PCB design of the all the circuits in the walker and arrange them as an Arduino shield for easy assembly, fabrication, stability and reduce the space occupied by the system. In the Appendix the initial PCB design in eagle is shown.

Beyond the specific application of the Smart Walker, this project was written as a good resource for anyone attempting to measure axial load, torque or punctual forces with FSRs, develop a cost effective data acquisition systems, utilize IMUs and develop LabVIEW user interfaces to work with Arduino boards and wireless connectivity using Xbee antennas. It is wished that it helps future students, engineers or anyone that wants to make use of the material presented.

Appendix A
Walker FEM Study

A simple Cad model of the walker was made. A static structural simulation of the model was performed with ANSYS in order to predict the strains of the principal structure of the walker when perpendicular incremental pressures on the handles are applied. This study helped to have a better perspective on the positioning of the strain gages by avoiding areas with large strain gradients..

Geometry and Model Setup

Only the essential parts were considered. The contacts were set as bonded for this study. The material used was the general structural aluminum in ANSYS. Figure A-1 shows the supports and applied pressure on the handles. This model setup was the one that approached more closely to the conditions of calibration.

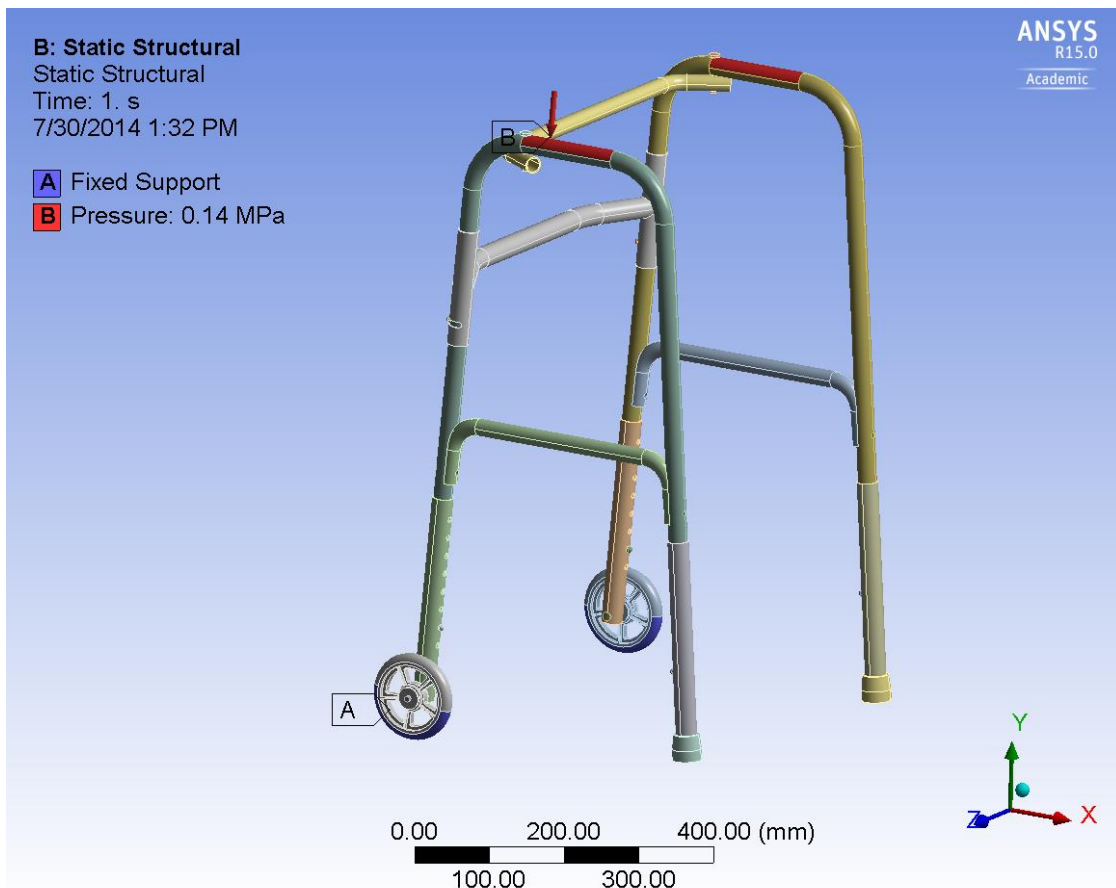


Figure A-1 CAD model, pressure applied and supports

Mesh

Three meshes were used for convergence analysis of the results and are shown in Figure A-2, Figure A-3 and Figure A-4. Three dimensional tetrahedral elements were used. The first the one generated by ANSYS by default, the second one with a 5 mm element size and the third one 3 mm element size.



Figure A-2 First Mesh

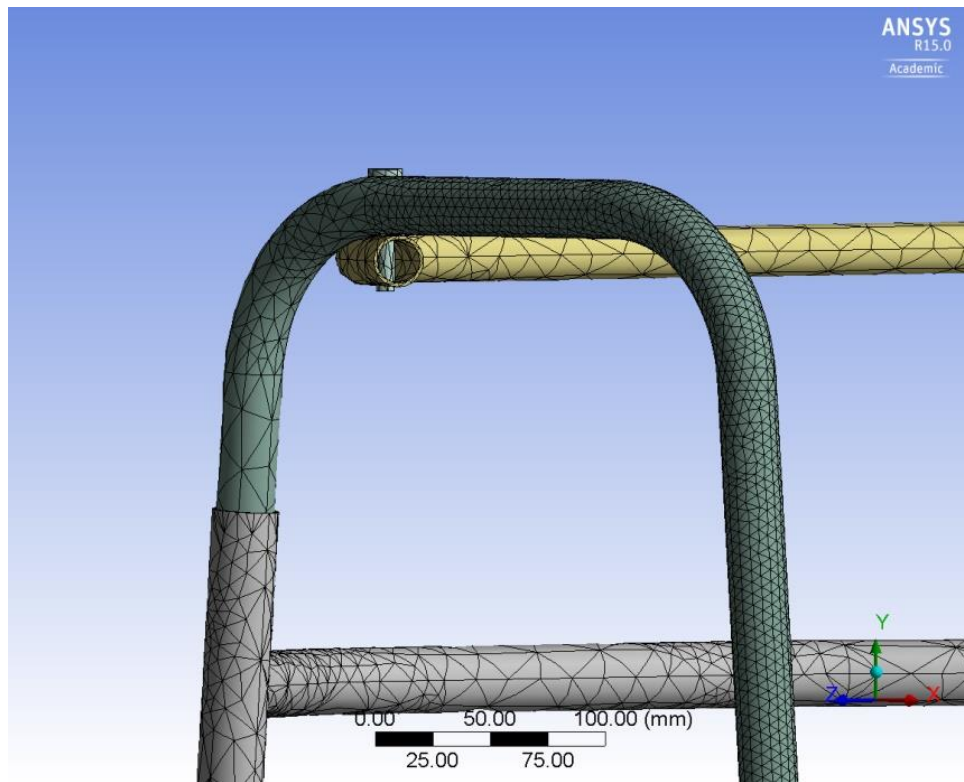


Figure A-3 Second Mesh [5mm element]

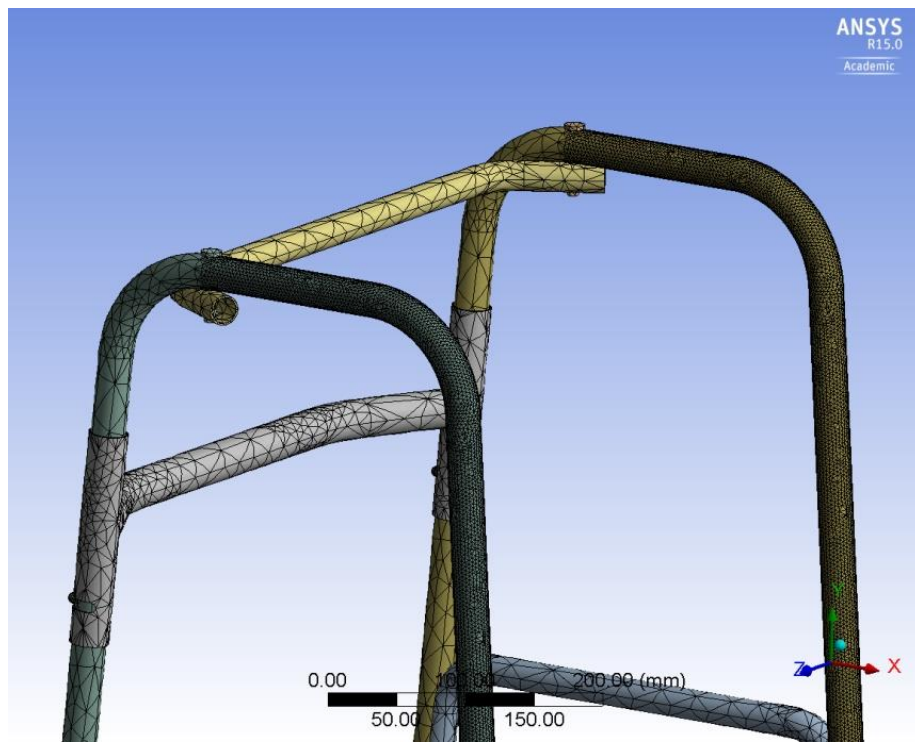


Figure A-4 Third Mesh [3mm element]

Results

Table A-1 shows the values of equivalent strain, and total deformation for their respective mesh. Reasonable convergence is observed for the maximum values. For the regions of the walker where the strains are wanted the values are very stable. Figure A-5 and Figure A-6 show screenshots of the results of the equivalent strain and total deformation results obtained from the model.

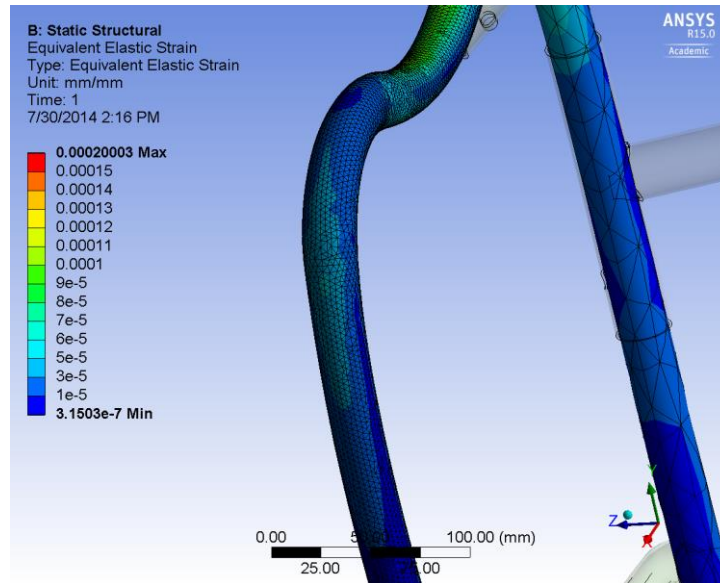


Figure A-5 Equivalent Strain (close-up 3rd Mesh)

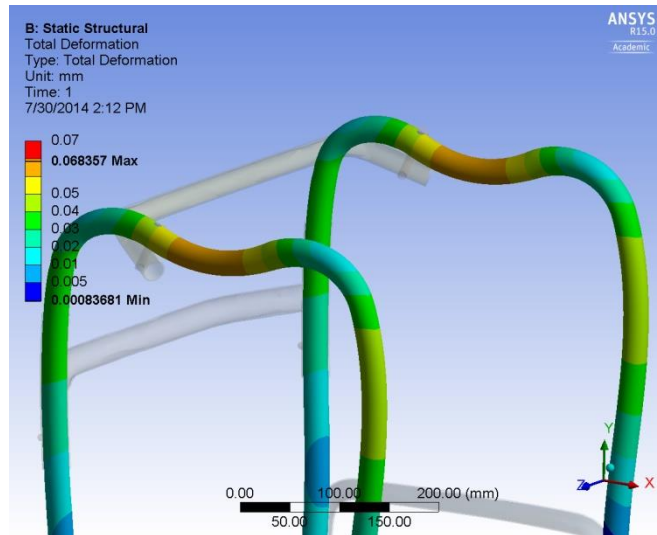


Figure A-6 Total Deformation [mm] (3rd Mesh)

Table A-1 Result Summary & percent change for convergence

	Max equivalent strain [$\mu\epsilon$]	$\Delta\%$	Total Deformation [mm]	$\Delta\%$
1st Mesh	219.8		0.080	
2nd Mesh	187.6	14.6	0.067	16.3
3rd Mesh	200.0	6.6	0.068	1.5

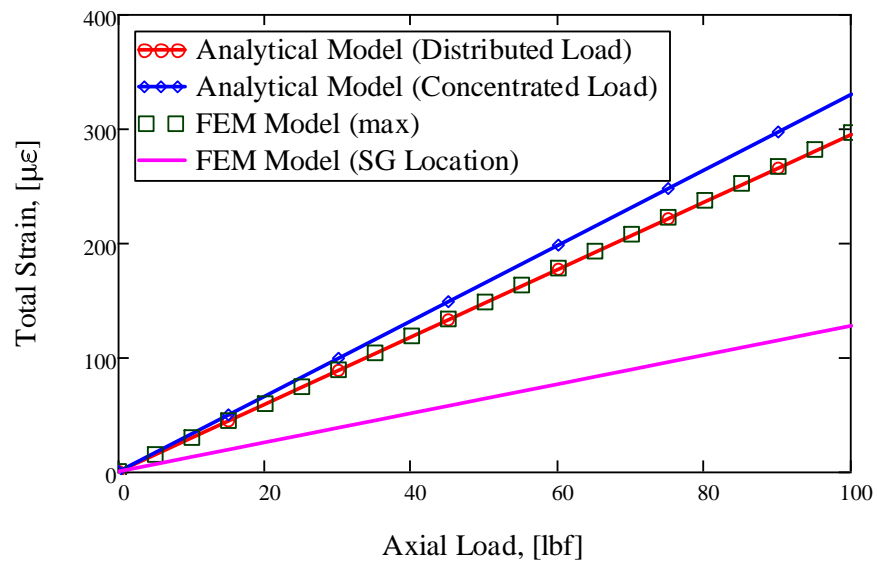
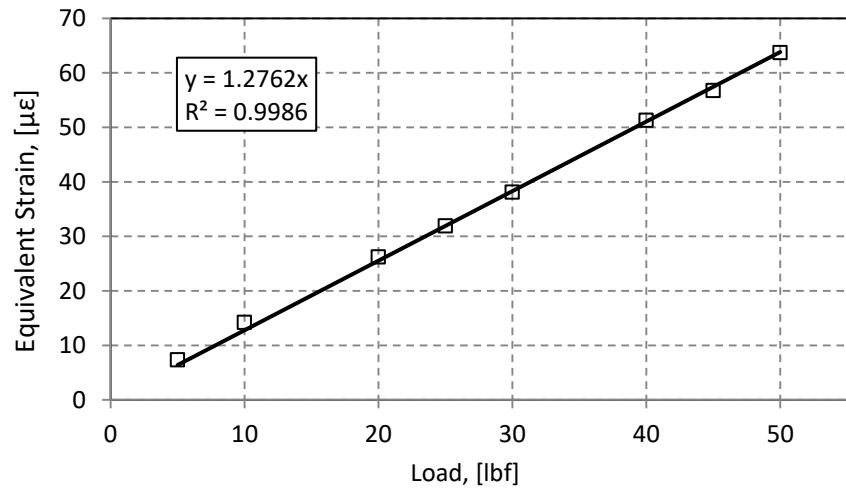


Figure 2-5 Total Strain vs. Applied Load for Analytical and FEM model

Appendix B
Smart Walker Arduino Code

```

//Smart Walker Arduino Code

//This code acquires 8 analog voltage inputs and IMU readings through I2C connection,
compiles the readings in an array of characters and sends them through the serial port
connected to the Xbee antenna

//ASTL Lab – The University of Texas at Arlington

//Mauricio Jaguan Nieves


#include <FreeSixIMU.h>
#include <FIMU_ADXL345.h>
#include <FIMU_ITG3200.h>
#include "CommunicationUtils.h"
#include <Wire.h>
#include <SoftwareSerial.h>


//Variables for IMU reading
float Acc[4];
float gyr[4];
int Acci[4];
int Acc_abs[4];
int Gyi[4];
int Gy_abs[4];
String Accs[4];
String Gys[4];
String A_str_tot, G_str_tot;
FreeSixIMU my3IMU = FreeSixIMU();

```

```

SoftwareSerial Xbee(2,3);

//USB Serial Connection
Serial.begin(57600);

// the setup routine runs once when you press reset:
void setup() {
  //Xbee Serial Connection
  XBee.begin(57600);
  analogReference(INTERNAL1V1);
  Wire.begin();
  delay(5);
  my3IMU.init();
  delay(5);
}

int analog[8];
float Vf[8];
int Vi[8];
String Vs[8];

// the loop routine runs over and over again forever:
void loop() {
  String V_tot_string = "";
  for (int x = 0; x<8 ; x++){
    // read the input on analog pin 0-7:
    analog[x] = analogRead(x);

```

```

// Convert to Float

Vf[x] = analog[x] * (1.1 / 1023.0)*1000;


//Convert back to integer

Vi[x] = int(Vf[x]);

//Convert to String

if (Vi[x]>=256){Vs[x] = String(Vi[x],HEX);};
if (Vi[x]<256){Vs[x] = "0" + String(Vi[x],HEX);};
if (Vi[x]<16){Vs[x] = "00" + String(Vi[x],HEX);};

//Create full string from analog inputs

V_tot_string = V_tot_string + Vs[x];

}

A_str_tot = "";
G_str_tot = "";

//Obtain values from IMU

my3IMU.getValues(Acc);

my3IMU.getYawPitchRoll(gyr);

// Normalize Acceleration readings

for(int x = 0; x<4 ; x++){

    Acci[x] = int(Acc[x]*1000/260);

}

//Get absolute values to avoid continuous unnecessary conversions when transforming to
strings

for(int x = 0; x<4 ; x++){

    Acc_abs[x] = abs(Acci[x]);

```

```

    Gy_abs[x] = abs(gyr[x]);
}

//convert IMU readings to strings (4 characters)
for(int x = 0; x<4 ; x++){
    if (Acci[x]<0){
        if (Acc_abs[x] >= 256){Accs[x] = "-" + String(Acc_abs[x],HEX);};
        if (Acc_abs[x] < 256){Accs[x] = "-0" + String(Acc_abs[x],HEX);};
        if (Acc_abs[x] < 16){Accs[x] = "-00" + String(Acc_abs[x],HEX);};
    }
    if (Acci[x]>=0){
        if (Acc_abs[x] >= 256){Accs[x] = "0" + String(Acc_abs[x],HEX);};
        if (Acc_abs[x] < 256){Accs[x] = "00" + String(Acc_abs[x],HEX);};
        if (Acc_abs[x] < 16){Accs[x] = "000" + String(Acc_abs[x],HEX);};
    }
    if ( x==0 ){
        if (gyr[x]<0){
            if (Gy_abs[x] >= 256){Gys[x] = "-" + String(Gy_abs[x],HEX);};
            if (Gy_abs[x] < 256){Gys[x] = "-0" + String(Gy_abs[x],HEX);};
            if (Gy_abs[x] < 16){Gys[x] = "-00" + String(Gy_abs[x],HEX);};
        }
        if (gyr[x]>=0){
            if (Gy_abs[x] >= 256){Gys[x] = "-" + String(Gy_abs[x],HEX);};
            if (Gy_abs[x] < 256){Gys[x] = "00" + String(Gy_abs[x],HEX);};
            if (Gy_abs[x] < 16){Gys[x] = "000" + String(Gy_abs[x],HEX);};
        }
    }
}

```

```

if ( x!=0 ){
  if (gyr[x]<0){
    if (Gy_abs[x] < 256){Gys[x] = "-" + String(Gy_abs[x],HEX);};
    if (Gy_abs[x] < 16){Gys[x] = "-0" + String(Gy_abs[x],HEX);};
  }
  if (gyr[x]>=0){
    if (Gy_abs[x] < 256){Gys[x] = "0" + String(Gy_abs[x],HEX);};
    if (Gy_abs[x] < 16){Gys[x] = "00" + String(Gy_abs[x],HEX);};
  }
}

for(int i =0; i<3; i++){
  A_str_tot = A_str_tot + Accs[i];
  G_str_tot = G_str_tot + Gys[i];
  String Tot_string = V_tot_string + A_str_tot + G_str_tot;
  //Print String to Arduino USB port
  //Serial.println(Tot_string);

  //Print to Xbee
  XBee.println(Tot_string);
  delay(16);
}
}

```


Appendix C
Axial Load Calibration
Data Analysis Mathcad Worksheet

The following Mathcad worksheet was created to automate the axial load strain gage calibration data analysis process for further walkers. It takes voltage readings acquired with the Arduino for a reasonable time (in the case of the example shown ~20 sec). The Worksheet requires inputting the names of the files to be read with the load applied as title. It extracts the data from the .txt file and averages the voltage readings over the time it was acquired, calculates the standard deviation, plots linear calibration curves and shows the coefficients of the curves. It also includes standard deviation plots to show the uncertainty of the reading that comes from the noise of the signal.

The calibration process to use this worksheet is as follows:

1. Acquire sets of data from the Arduino with voltage readings of all strain gages for a period of 10 to 20 seconds. Data sets need to follow the format shown in Figure C-1.

9/12/2014	3:03:28 PM			
Time [sec]	Volt SG 1 [V]	Volt SG 2 [V]	Volt SG 3 [V]	Volt SG 4 [V]
1.037	0.3151	0.3933	0.2451	0.4152
1.037	0.3079	0.3878	0.2427	0.4159
1.037	0.3200	0.3976	0.2463	0.4153
1.037	0.3327	0.4077	0.2504	0.4146
1.037	0.3342	0.4083	0.2507	0.4139
1.037	0.3317	0.4054	0.2493	0.4135
1.037	0.3309	0.4050	0.2492	0.4135
1.037	0.3307	0.4065	0.2503	0.4135
1.037	0.3299	0.4066	0.2503	0.4133
1.037	0.3295	0.4052	0.2495	0.4130

Figure C-1 Data files format to be imported to the worksheet

2. Name each file "load applid.txt" i.e. "80.txt"
3. Separate the readings taken while loading and unloading the walker.
Notice there is a directory for loading and one for unloading. This is to make sure hysteresis is taken into account.
4. Match the string columns in the worksheet with the titles of your files.
Make sure to input the proper directory where files have been stored.
5. Calibration curves should be calculated automatically

Strain gage Calibration Data Analysis**Read Files****File Location and Name**

LocationL := "C:\Users\Mauricio\Google Drive\Universidad\ASTL\4 Strain gage test - Axial load\Calibration\Load\"

LocationU := "C:\Users\Mauricio\Google Drive\Universidad\ASTL\4 Strain gage test - Axial load\Calibration\UnLoad\"

Vnum2str(M) :=
$$\begin{cases} \text{for } i \in 0 \dots \text{length}(M) - 1 \\ V_i \leftarrow \text{num2str}(M_i) \\ V \end{cases}$$

Load :=
$$\begin{pmatrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 10 \\ 20 \\ 40 \\ 60 \\ 65 \\ 70 \\ 75 \\ 76 \\ 77 \\ 78 \\ 79 \\ 80 \end{pmatrix}$$

Loadtxt := Vnum2str(Load) =

	0
0	"0"
1	"1"
2	"2"
3	"3"
4	"4"
5	"5"
6	"10"
7	"20"
8	"40"
9	"60"
10	"65"
11	"70"
12	"75"
13	"76"
14	"77"
15	...

Unload :=

$$\begin{pmatrix} 0 \\ 20 \\ 40 \\ 60 \\ 65 \\ 70 \\ 75 \\ 76 \\ 77 \\ 78 \\ 79 \\ 80 \end{pmatrix}$$

UnLoadtxt := Vnum2str(Unload) =

	0
0	"0"
1	"20"
2	"40"
3	"60"
4	"65"
5	"70"
6	"75"
7	"76"
8	"77"
9	"78"
10	"79"
11	"80"

Vmean & Standard dev calculations (Read Files in folder)

Vmean(Loc,Mtxt) :=
$$\begin{cases} \text{for } i \in 0 \dots \text{length}(Mtxt) - 1 \\ \text{filename} \leftarrow \text{concat}(\text{Loc}, Mtxt_i, ".txt") \\ M \leftarrow \text{READFILE} \left[\text{filename}, "delimited", \begin{pmatrix} 10 \\ 350 \end{pmatrix}, 1 \right] \\ \text{for } j \in 1 \dots 4 \\ Vmean_{i,j-1} \leftarrow \text{mean}(M^{(j)}) \\ Vmean \end{cases}$$

Loc --> Location of the text files to be read
Mtxt--> Name of the specific text files as strings

Vmean_Load := Vmean(LocationL,Loadtxt)

Vmean_Unload := Vmean(LocationU,UnLoadtxt)

Vmean_total(sg) := stack(Vmean_Load^(sg-1), Vmean_Unload^(sg-1))

Smart Walker Project

$X := \text{stack}(\text{Load}, \text{Unload})$

Mauricio Jaguan Nieves

Sdev(Loc, Mtxt) :=

for $i \in 0 \dots \text{length}(\text{Mtxt}) - 1$

filename $\leftarrow \text{concat}(\text{Loc}, \text{Mtxt}_i, ".\text{txt}")$

$M \leftarrow \text{READFILE} \left[\text{filename}, "delimited", \begin{pmatrix} 10 \\ 350 \end{pmatrix}, 1 \right]$

for $j \in 1 \dots 4$

$V\text{Stdev}_{i,j-1} \leftarrow \text{Stdev}(M^{(j)})$

VStdev

Sdev_Load := Sdev(LocationL, Loadtxt)

Sdev_Unload := Sdev(LocationU, UnLoadtxt)

Sdev_total(sg) := stack(Sdev_Load^(sg-1), Sdev_Unload^(sg-1))

sg→strain gage

Linear Fit Lines

Intercept

$b(\text{sg}) := \text{intercept}(X, \text{Vmean_total}(\text{sg}))$

Slope

$a(\text{sg}) := \text{slope}(X, \text{Vmean_total}(\text{sg}))$

R^2

$R_2(\text{sg}) := \text{corr}(X, \text{Vmean_total}(\text{sg}))^2$

Plots

$\text{Volt}(\text{sg}, x) := b(\text{sg}) + a(\text{sg}) \cdot x$

$x := 0, 0.5 \dots 80$

Plots

Error Bars

sg→strain gage

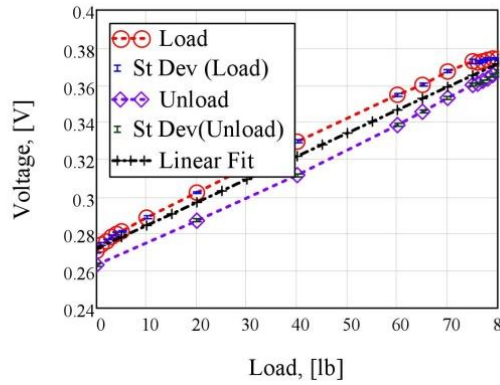
eu_load(sg) := $\text{Vmean_Load}^{\langle \text{sg}-1 \rangle} + \text{Sdev_Load}^{\langle \text{sg}-1 \rangle}$

eu_unload(sg) := $\text{Vmean_Unload}^{\langle \text{sg}-1 \rangle} + \text{Sdev_Unload}^{\langle \text{sg}-1 \rangle}$

ed_load(sg) := $\text{Vmean_Load}^{\langle \text{sg}-1 \rangle} - \text{Sdev_Load}^{\langle \text{sg}-1 \rangle}$

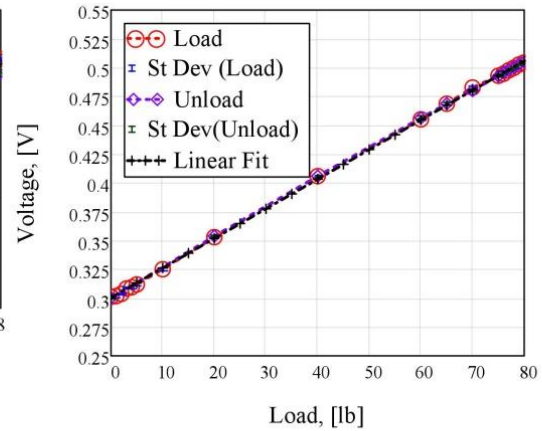
ed_unload(sg) := $\text{Vmean_Unload}^{\langle \text{sg}-1 \rangle} - \text{Sdev_Unload}^{\langle \text{sg}-1 \rangle}$

Strain Gage 1



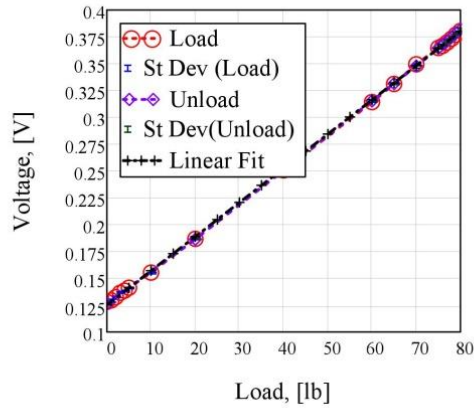
Intercept $b(1) = 0.272$
Slope $a(1) \cdot 1000 = 1.24 \frac{\text{mV}}{\text{lb}}$
R^2 $R_2(1) = 0.9776$

Strain Gage 2



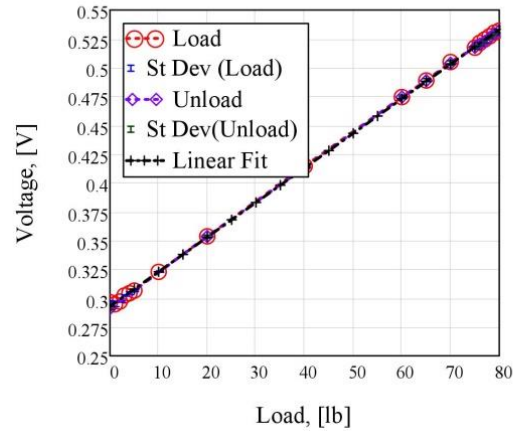
Intercept $b(2) = 0.301$
Slope $a(2) \cdot 1000 = 2.569 \frac{\text{mV}}{\text{lb}}$
R^2 $R_2(2) = 0.9998$

Strain Gage 3



Intercept $b(3) = 0.125$
Slope $a(3) \cdot 1000 = 3.185 \frac{\text{mV}}{\text{lb}}$
R^2 $R_2(3) = 0.9998$

Strain Gage 4

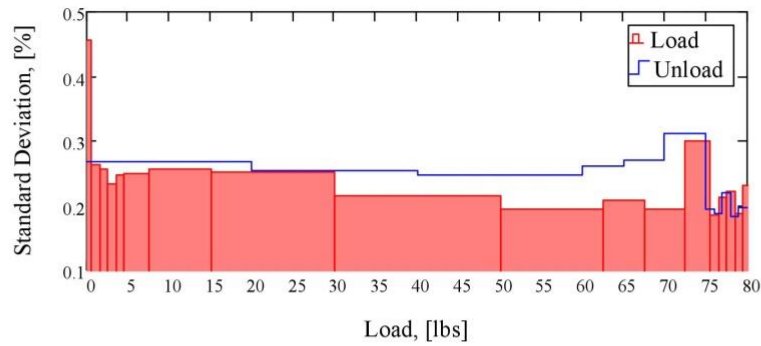
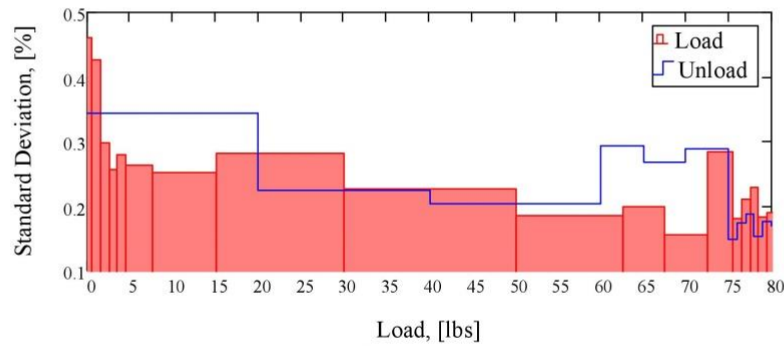
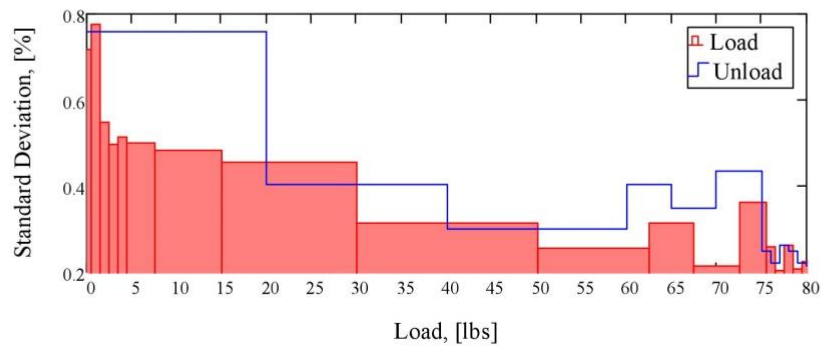


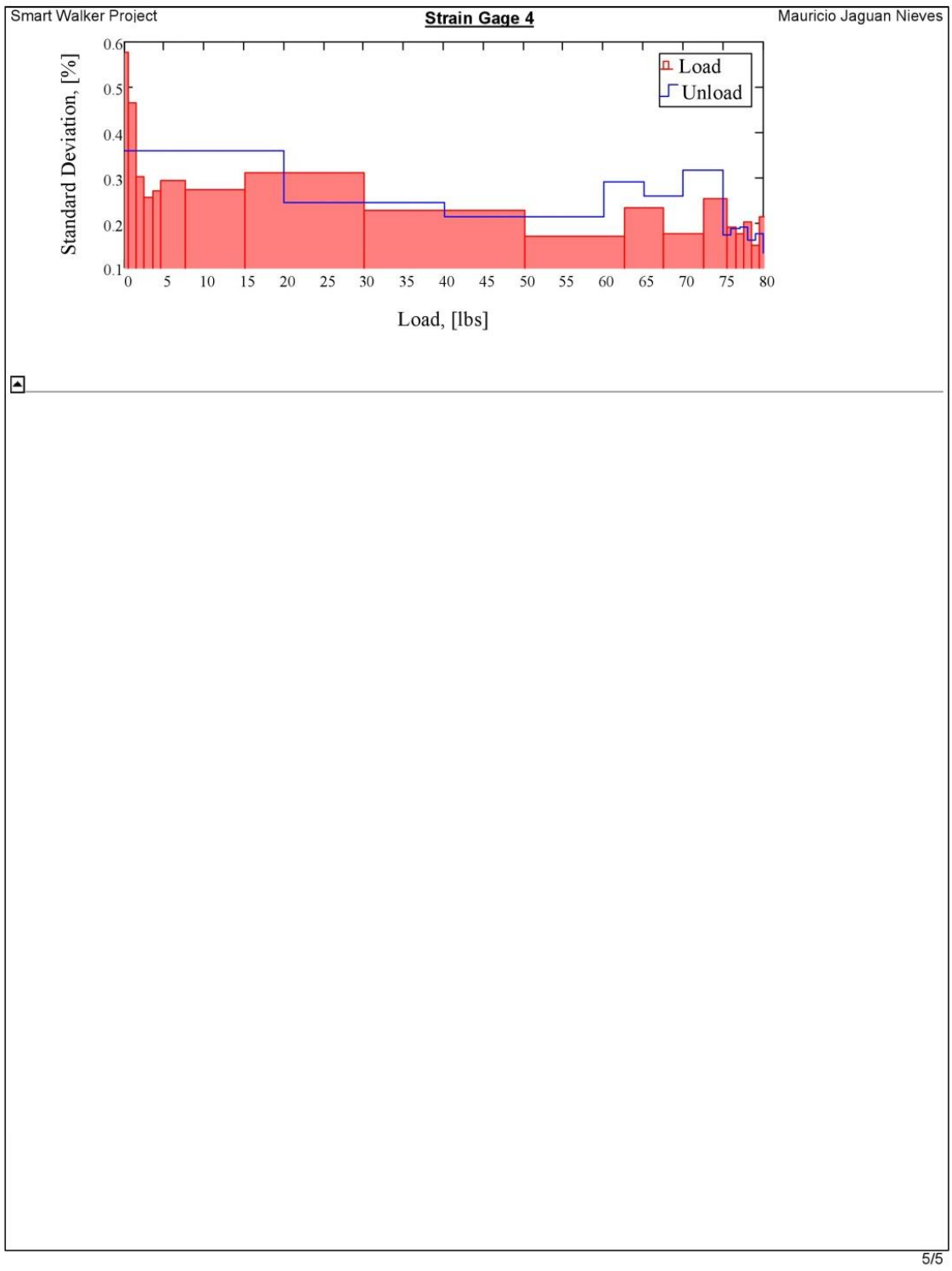
Intercept $b(4) = 0.293$
Slope $a(4) \cdot 1000 = 3.003 \frac{\text{mV}}{\text{lb}}$
R^2 $R_2(4) = 0.9998$

Standard Deviation Plots

$$\text{eplotL}(\text{sg}) := \frac{\text{Sdev_Load}^{\langle \text{sg}-1 \rangle}}{\text{Vmean_Load}^{\langle \text{sg}-1 \rangle}} \cdot 100$$

$$\text{eplotU}(\text{sg}) := \frac{\text{Sdev_Unload}^{\langle \text{sg}-1 \rangle}}{\text{Vmean_Unload}^{\langle \text{sg}-1 \rangle}} \cdot 100$$

Strain Gage 1**Strain Gage 2****Strain Gage 3**



Appendix D
Torque Calibration
Data Analysis Mathcad Worksheet

The following Mathcad worksheet similarly to the one for the axial load sensor was created to automate the torque sensor calibration data analysis process for further walkers. It takes voltage readings acquired with the Arduino for a reasonable time (in the case of the example shown ~20 sec). It extracts the data from the .txt file and averages the voltage readings over the time it was acquired, calculates the standard deviation, plots linear calibration curves and shows the coefficients of the curves.

The calibration process to use this worksheet is as follows:

1. Acquire sets of data from the Arduino with voltage readings of all strain gages for a period of 10 to 20 seconds. Data sets need to follow the format shown in Figure D-1.

12/23/2014	4:00:22 PM					
Time [sec]	Volt SG 1 [V]	Volt SG 2 [V]	Volt SG 3 [V]	Volt SG 4 [V]	Torque L [V]	Torque R [V]
8.697	0.1457	0.0285	0.0154	0.0024	0.1198	0.1223
8.697	0.1463	0.0290	0.0159	0.0023	0.1200	0.1226
8.697	0.1460	0.0287	0.0157	0.0022	0.1204	0.1222
8.697	0.1458	0.0285	0.0157	0.0022	0.1204	0.1215
8.697	0.1461	0.0288	0.0162	0.0022	0.1203	0.1213
8.697	0.1466	0.0291	0.0168	0.0021	0.1203	0.1213
8.697	0.1469	0.0292	0.0170	0.0021	0.1200	0.1214
9.219	0.1466	0.0290	0.0165	0.0021	0.1198	0.1218
9.219	0.1467	0.0291	0.0165	0.0021	0.1197	0.1219

Figure D-1 Data files format to be imported to the worksheet

2. Creates vectors to read the defined directories and separate data according to handle and magnitude and direction of the torque applied.
3. This worksheet also can read accelerometer readings from data in columns adjacent to the ones shown above for torque.
4. Obtains voltage average of the time the data was acquired for each torque applied as well as standard deviation to have a sense of the noise in the signal.
5. The calculated averages are compiled to create vectors and create linear trends and calculate the calibration curves.

Smart Walker Sensor Calibration

Torque

Read Files

File Location and Names

Location := CWD = "C:\Users\Mauricio\Google Drive\Universidad\ASTL\Torque Calibration\Calibration (Ax+To+acc)\Torque\"

Vnum2str(M) :=	for i ∈ 0 .. length(M) - 1 V _i ← num2str(M _i) V
----------------	--

-----> Converts Vector of integers in a vector of strings

Nb -----> Number of divisions along X (bar)

Nb := 7

Nh -----> Number of divisions along Y (Handle)

Nh := 3

Nl -----> Number of Tests (Constant loads)

Nl := 2

Test

1 --> Load = 1lb
2 --> Load = 0.5lb

Xposition :=

for i ∈ 0 .. (Nb - 1)	X _i ← i
X	

Xposition_txt := Vnum2str(Xposition) =

"0"
"1"
"2"
"3"
"4"
"5"
"6"

Vectors of Strings of addresses and names of the text files to be read.

Side :=

"Left"
"Right"

Direction :=

"Negative"
"Positive"

Test :=

"1"
"2"

C:\Users\Mauricio\Google Drive\Universidad\ASTL\Nov24\Calibration (Ax+To+acc)\Torque\Left\Negative\

Name :=

for h ∈ 0 .. 1	for i ∈ 0 .. 1	for j ∈ 0 .. 1	for k ∈ 0 .. 6	A _k ← concat(Location, Side _j , "\", Direction _i , "\", Test _h , "\", Xposition_txt _k , "_6.txt")
				B _{0,j} ← A
				C _i ← B
				D _h ← C
				D

Name =

{2,1}
{2,1}

Name₀ =

{1,2}
{1,2}

[Name₍₀₎]₀ = ({7,1} {7,1})


Vmean & Standard Deviation



```

Vmean_stdev := M0,0 ← 0                                     concat(Location, Sidej, "\", Directioni, "\", Testh, "\", Xposition_txtk, "_6.txt")
N0,0 ← 0
Acc ← 3 if Accelerometer = 0
Acc ← 0 if Accelerometer = 1
D ← Data
for h ∈ 0 .. 1
  for i ∈ 0 .. 1
    for j ∈ 0 .. 1
      for k ∈ 0 .. 6
        for l ∈ 1 .. 9 - Acc
          Mk,l-1 ← mean[[(Dh)i,j,k]]
          M2k,l-1 ← Stdev[[(Dh)i,j,k]]
          N ← M
          N2 ← M2
          Oj ← N
          O2j ← N2
          Pi ← O
          P2i ← O2
          Qh ← P
          Q2h ← P2
        R0,0 ← Q
        R0,1 ← Q2
      return R

```

$n :=$  $n = 0$

$$Vmean := Vmean_stdev_{0,0} = \begin{pmatrix} \{2,1\} \\ \{2,1\} \end{pmatrix} \quad Vmean_0 = \begin{pmatrix} \{2,1\} \\ \{2,1\} \end{pmatrix} \quad (Vmean_0)_0 = \begin{pmatrix} \{7,6\} \\ \{7,6\} \end{pmatrix}$$

$$\left[(Vmean_0)_0 \right]_1 = \begin{pmatrix} 0.6539 & 0.1317 & 0.1002 & 0.016 & 0.5697 & 0.5198 \\ 0.652 & 0.1303 & 0.0984 & 0.016 & 0.5697 & 0.515 \\ 0.6531 & 0.1316 & 0.099 & 0.016 & 0.5694 & 0.5095 \\ 0.6523 & 0.1307 & 0.0984 & 0.016 & 0.5691 & 0.504 \\ 0.6507 & 0.1303 & 0.0987 & 0.016 & 0.5778 & 0.5032 \\ 0.651 & 0.1311 & 0.0992 & 0.016 & 0.5775 & 0.4978 \\ 0.6492 & 0.1298 & 0.0975 & 0.016 & 0.5788 & 0.4914 \end{pmatrix}$$

$$\sigma := \text{Vmean_stdev}_{0,1} = \begin{pmatrix} \{2,1\} \\ \{2,1\} \end{pmatrix} \quad \sigma_0 = \begin{pmatrix} \{2,1\} \\ \{2,1\} \end{pmatrix} \quad [\sigma_{(0)}]_0 = \begin{pmatrix} \{7,6\} \\ \{7,6\} \end{pmatrix}$$

$$\left[\left[[\sigma_0]_0 \right] \right] = \begin{pmatrix} 0.0078 & 0.0072 & 0.0068 & 0.0001 & 0.0006 & 0.0008 \\ 0.0073 & 0.0067 & 0.0061 & 0.0001 & 0.0005 & 0.0008 \\ 0.0075 & 0.0068 & 0.0065 & 0.0001 & 0.0005 & 0.0008 \\ 0.008 & 0.0074 & 0.0069 & 0.0001 & 0.0006 & 0.0009 \\ 0.0068 & 0.0062 & 0.0058 & 0.0001 & 0.0005 & 0.0008 \\ 0.008 & 0.0075 & 0.007 & 0.0001 & 0.0006 & 0.0009 \\ 0.0081 & 0.0075 & 0.0069 & 0.0001 & 0.0006 & 0.0008 \end{pmatrix}$$



Load Vectors



```

ApTorq :=
  P1 ← 0.5
  P2 ← 1
  for x ∈ 0 .. 6
    Tx,0 ← P1·x
    Tx,1 ← P2·x
  T2 ← stack(-T, T)
  return T2

```

ApTorq =

	0	1
0	0	0
1	-0.5	-1
2	-1	-2
3	-1.5	-3
4	-2	-4
5	-2.5	-5
6	-3	-6
7	0	0
8	0.5	1
9	1	2
10	1.5	3
11	2	4
12	2.5	5
13	3	6



Voltage Vectors

$$V_vect := \left[\begin{array}{l} V \leftarrow Vmean \\ \text{for } h \in 0..1 \\ \quad \text{for } j \in 0..1 \\ \quad \quad \text{for } i \in 0..1 \\ \quad \quad \quad \text{for } k \in 0..6 \\ \quad \quad \quad \quad \text{for } l \in 0..5 \\ \quad \quad \quad \quad \quad M_{k,l} \leftarrow \left[\left[\left[(V_h)_i \right]_{j,k,l} \right] - \left[\left[(V_h)_i \right]_{0,l} \right] \right] \\ \quad \quad \quad \quad \quad N_i \leftarrow M \\ \quad \quad \quad \quad \quad O_j \leftarrow \text{stack}(N_0, N_1) \\ \quad \quad P_h \leftarrow O \end{array} \right] P$$

$$\left[(Vmean)_l \right]_l = \begin{pmatrix} 0.6629 & 0.1244 & 0.0954 & 0.016 & 0.5808 & 0.5287 \\ 0.6638 & 0.1248 & 0.0955 & 0.016 & 0.5814 & 0.5390 \\ 0.6639 & 0.124 & 0.0951 & 0.016 & 0.5824 & 0.5495 \\ 0.6641 & 0.1228 & 0.0942 & 0.016 & 0.583 & 0.5617 \\ 0.6646 & 0.124 & 0.0944 & 0.016 & 0.5835 & 0.5717 \\ 0.6648 & 0.1236 & 0.0947 & 0.016 & 0.5842 & 0.5794 \\ 0.6646 & 0.1226 & 0.0935 & 0.016 & 0.5852 & 0.5907 \end{pmatrix}$$

$$(V_vect) = \begin{pmatrix} \{2,1\} \\ \{2,1\} \end{pmatrix}$$

$$V_vect_0 = \begin{pmatrix} \{14,6\} \\ \{14,6\} \end{pmatrix}$$

$$\sigma_vect := \left[\begin{array}{l} V \leftarrow \sigma \\ \text{for } h \in 0..1 \\ \quad \text{for } j \in 0..1 \\ \quad \quad \text{for } i \in 0..1 \\ \quad \quad \quad \text{for } k \in 0..6 \\ \quad \quad \quad \quad \text{for } l \in 0..5 \\ \quad \quad \quad \quad \quad M_{k,l} \leftarrow \left[\left[\left[(V_h)_i \right]_{j,k,l} \right] - \left[\left[(V_h)_i \right]_{0,l} \right] \right] \\ \quad \quad \quad \quad \quad N_i \leftarrow M \\ \quad \quad \quad \quad \quad O_j \leftarrow \text{stack}(N_0, N_1) \\ \quad \quad P_h \leftarrow O \end{array} \right] P$$

$$V_{\text{vect}} = \begin{pmatrix} \{2,1\} \\ \{2,1\} \end{pmatrix} \quad V_{\text{vect}_0} = \begin{pmatrix} \{14,6\} \\ \{14,6\} \end{pmatrix}$$

$$(V_{\text{vect}_1})_0 =$$

	0	1	2	3
0	0	0	0	0
1	$1.8234 \cdot 10^{-4}$	$9.2211 \cdot 10^{-4}$	$2.8224 \cdot 10^{-4}$	$-7.2384 \cdot 10^{-6}$
2	$-1.9734 \cdot 10^{-3}$	$-1.4557 \cdot 10^{-3}$	$-2.7876 \cdot 10^{-3}$	$-9.3206 \cdot 10^{-6}$
3	$-3.4112 \cdot 10^{-4}$	$3.4832 \cdot 10^{-4}$	$-2.4505 \cdot 10^{-3}$	$-1.8824 \cdot 10^{-5}$
4	$-8.3053 \cdot 10^{-4}$	$-9.3212 \cdot 10^{-4}$	$-3.9753 \cdot 10^{-3}$	$-2.5355 \cdot 10^{-5}$
5	$-2.7406 \cdot 10^{-4}$	$1.7926 \cdot 10^{-4}$	$-4.5901 \cdot 10^{-3}$	$-1.3294 \cdot 10^{-5}$
6	$-9.0757 \cdot 10^{-4}$	$-1.2457 \cdot 10^{-3}$	$-5.8593 \cdot 10^{-3}$	$-1.9531 \cdot 10^{-6}$
7	0	0	0	0
8	$-1.1364 \cdot 10^{-3}$	$-1.9573 \cdot 10^{-3}$	$-2.6528 \cdot 10^{-3}$	$-7.9563 \cdot 10^{-6}$
9	$3.7101 \cdot 10^{-5}$	$-1.2931 \cdot 10^{-3}$	$-2.6471 \cdot 10^{-3}$	$-1.057 \cdot 10^{-5}$
10	$-7.3702 \cdot 10^{-4}$	$-2.3032 \cdot 10^{-3}$	$-2.7263 \cdot 10^{-3}$	$-2.8126 \cdot 10^{-5}$
11	$-6.6499 \cdot 10^{-4}$	$-2.4703 \cdot 10^{-3}$	$-2.8088 \cdot 10^{-3}$	$-5.1832 \cdot 10^{-5}$
12	$-2.1229 \cdot 10^{-4}$	$-2.7128 \cdot 10^{-3}$	$-3.189 \cdot 10^{-3}$	$-5.8708 \cdot 10^{-6}$
13	$-4.3358 \cdot 10^{-4}$	$-3.3532 \cdot 10^{-3}$	$-3.6802 \cdot 10^{-3}$...

$$ApTorq =$$

	0	1
0	0	0
1	-0.5	-1
2	-1	-2
3	-1.5	-3
4	-2	-4
5	-2.5	-5
6	-3	-6
7	0	0
8	0.5	1
9	1	2
10	1.5	3
11	2	4
12	2.5	5
13	3	6

$$(\sigma_{\text{vect}_1})_0 =$$

	0	1	2	3	4
0	0	0	0	0	0
1	$2.3166 \cdot 10^{-3}$	$2.3901 \cdot 10^{-3}$	$2.1986 \cdot 10^{-3}$	$-1.1087 \cdot 10^{-5}$	$-1.2603 \cdot 10^{-4}$
2	$1.9287 \cdot 10^{-3}$	$2.1045 \cdot 10^{-3}$	$1.8874 \cdot 10^{-3}$	$-1.4283 \cdot 10^{-5}$	$-1.3292 \cdot 10^{-4}$
3	$1.0464 \cdot 10^{-3}$	$1.1332 \cdot 10^{-3}$	$1.0073 \cdot 10^{-3}$	$-5.36 \cdot 10^{-6}$	$-1.0632 \cdot 10^{-4}$
4	$5.0462 \cdot 10^{-4}$	$5.0796 \cdot 10^{-4}$	$5.0909 \cdot 10^{-4}$	$-1.0821 \cdot 10^{-5}$	$-1.2675 \cdot 10^{-4}$
5	$1.3039 \cdot 10^{-3}$	$1.4211 \cdot 10^{-3}$	$1.2185 \cdot 10^{-3}$	$-1.3104 \cdot 10^{-5}$	$-1.0358 \cdot 10^{-4}$
6	$3.632 \cdot 10^{-3}$	$3.405 \cdot 10^{-3}$	$3.1342 \cdot 10^{-3}$	$-2.5167 \cdot 10^{-5}$...



Slope, intercepts and R^2 of linear regression (Axial Strain gages)

```

Regress := V ← V_vect
           T ← ApTorq
           for h ∈ 0..1
             for j ∈ 0..1
               side ← 4 if j = 0
               side ← 5 if j = 1
               a_j ← slope[(T^(h)), (V_h)_j^(side)]
               b_j ← intercept[(T^(h)), (V_h)_j^(side)]
               R_j ← corr[(T^(h)), (V_h)_j^(side)]^2
             M_h,0 ← a
             M_h,1 ← b
             M_h,2 ← R
           M

y = a*x + b

Regress = [ [ 9.9456 × 10^-3, 0.0102 ], [ 2.6043 × 10^-4, 1.4365 × 10^-3 ], [ 0.9999, 0.9947 ] ]
           [ [ 0.0101, 0.0106 ], [ -3.6218 × 10^-4, -1.2035 × 10^-5 ], [ 0.9998, 0.9995 ] ]

Slope := Regress^(0) = {2,1}
Interc := Regress^(1) = {2,1}
R2 := Regress^(2) = {2,1}

Slope_0 = [ 9.9456 × 10^-3, 0.0102 ]
Interc_0 = [ 2.6043 × 10^-4, 1.4365 × 10^-3 ]
R2_0 = [ 0.9999, 0.9947 ]

```

Functions for Plots

```

X_Plot1 := ApTorq^(0)
X_Plot2 := ApTorq^(1)

Yvect(Test, Side) := [ α ← 0 if Test = 1
                       α ← 1 if Test = 2
                       β ← 4 ∧ γ ← 0 if Side = "Left"
                       β ← 5 ∧ γ ← 1 if Side = "Right"
                       V ← [(V_vect_α)_γ]^(β) ]
Reg(Test, Side) := [ α ← 0 if Test = 1
                    α ← 1 if Test = 2
                    β ← 4 ∧ γ ← 0 if Side = "Left"
                    β ← 5 ∧ γ ← 1 if Side = "Right"
                    V_0 ← [(Regress^(0))_α_γ]
                    V_1 ← [(Regress^(1))_α_γ]
                    V_2 ← [(Regress^(2))_α_γ]
                    return V ]

Ystdev(Test, Side) := [ α ← 0 if Test = 1
                       α ← 1 if Test = 2
                       β ← 4 ∧ γ ← 0 if Side = "Left"
                       β ← 5 ∧ γ ← 1 if Side = "Right"
                       V ← [(σ_vect_α)_γ]^(β) ]

```


$$Y_{\text{vect}}(1, \text{"Left"}) =$$

	0
0	0
1	$-4.5279 \cdot 10^{-3}$
2	$-9.6739 \cdot 10^{-3}$
3	-0.0148
4	-0.0198
5	-0.0248
6	-0.0292
7	0
8	$5.3701 \cdot 10^{-3}$
9	0.0106
10	0.0152
11	0.0204
12	0.0252
13	0.0298

$$Y_{\text{stdev}}("1", \text{"Left"}) =$$

	0
0	0
1	$-8.3859 \cdot 10^{-5}$
2	$-2.2079 \cdot 10^{-5}$
3	$-4.4756 \cdot 10^{-6}$
4	$-6.4159 \cdot 10^{-5}$
5	$3.2498 \cdot 10^{-5}$
6	$1.8828 \cdot 10^{-5}$
7	0
8	$-4.5369 \cdot 10^{-5}$
9	$-1.641 \cdot 10^{-5}$
10	$1.9455 \cdot 10^{-5}$
11	$-4.3072 \cdot 10^{-5}$
12	$-6.4525 \cdot 10^{-5}$
13	$5.0902 \cdot 10^{-5}$

$$X_{\text{Plot1}} =$$

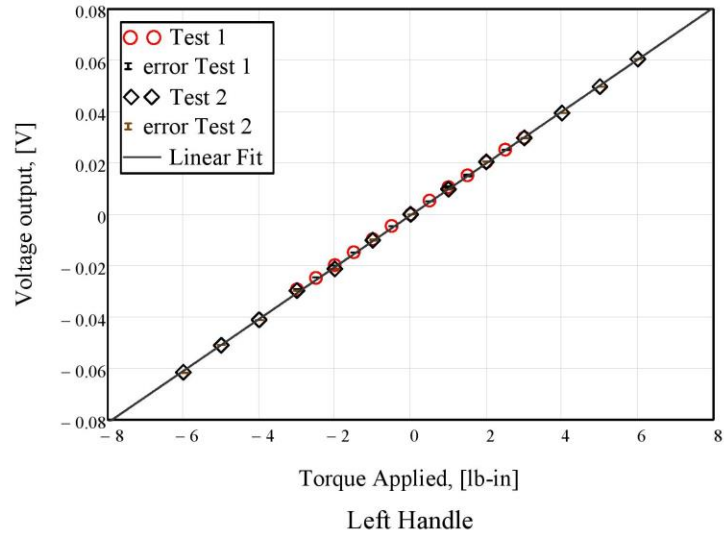
	0
0	0
1	-0.5
2	-1
3	-1.5
4	-2
5	-2.5
6	-3
7	0
8	0.5
9	1
10	1.5
11	2
12	2.5
13	3



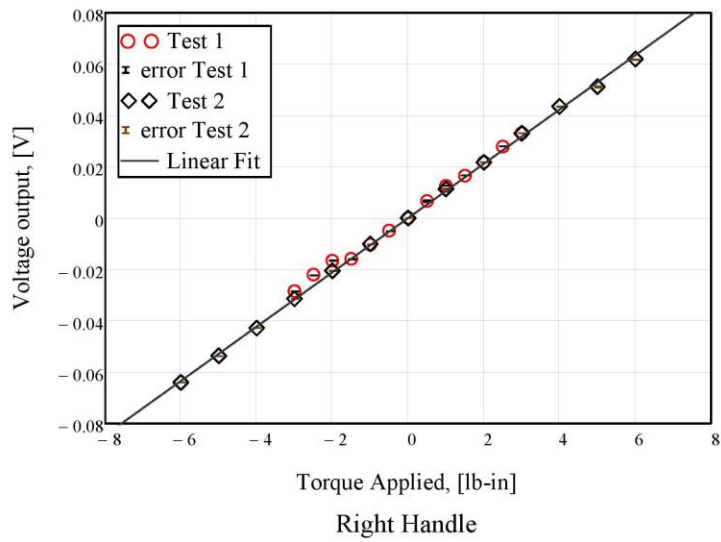
$$\begin{pmatrix} aL \\ bL \\ R2L \end{pmatrix} := \text{Reg}(2, \text{"Left"}) = \begin{pmatrix} 0.0101 \\ -3.6218 \times 10^{-4} \\ 0.9998 \end{pmatrix}$$

$$\text{FitL}(x) := aL \cdot x + bL$$

Plots



$$\begin{pmatrix} aR \\ bR \\ R2R \end{pmatrix} := \text{Reg}(2, \text{"Right"}) = \begin{pmatrix} 0.0106 \\ -1.2035 \times 10^{-5} \\ 0.9995 \end{pmatrix} \quad \text{FitR}(x) := aR \cdot x + bR$$



Appendix E
Smart Walker PCB Arduino Shield
(Initial Design)

After the first prototype was completed, the next step was to create a PCB design of the circuits in the walker to expedite the fabrication of further walkers, reduce the size of the circuit board. This section shows the initial efforts to create an Arduino shield that contained all the circuits described in this thesis (Figure E-1). A major change from the original circuits was the attempt to create Wheatstone bridges with low tolerance resistors rather than using bridge completion units. The main challenge was to fit all the components in the Arduino MEGA surface area knowing that it would be fabricated manually. This design proved to have various problems for manual fabrication due to the small size of the chosen components.

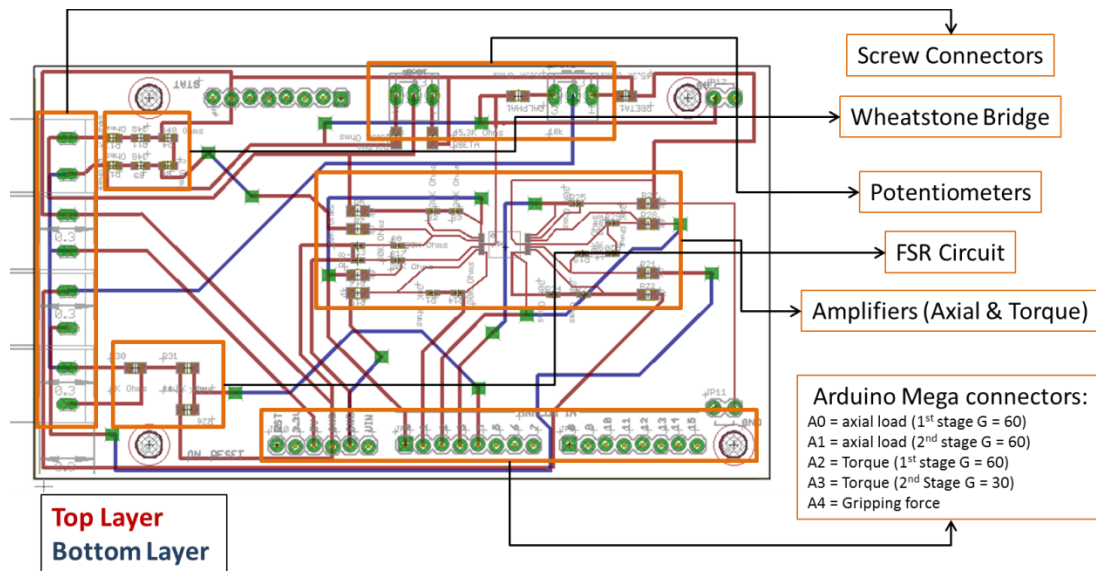


Figure E-1 Initial Smart Walker Arduino shield design

A stencil was made in order to solder the components using solder paste. Stencil is shown below.

Also, a board was etched and ready to place the components. Unfortunately placing the operational amplifier was very difficult and the board was damaged after the attempt.

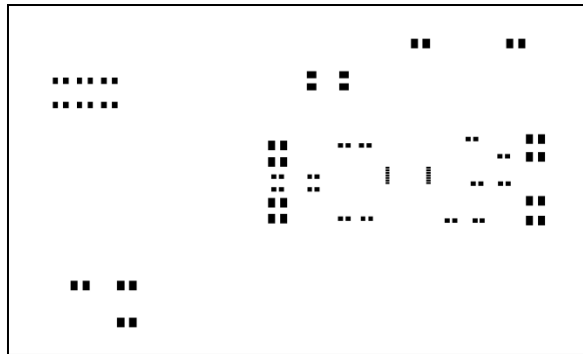


Figure E-2 Smart Walker Arduino Shield first design (Stencil)

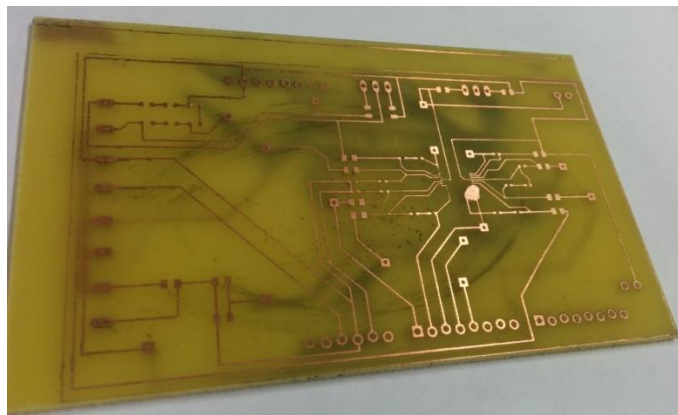


Figure E-3 Smart Walker Arduino Shield initial design (upper side)

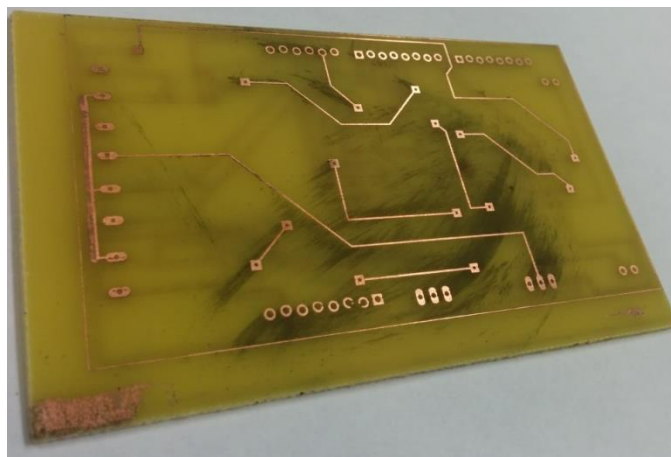


Figure E-4 Smart Walker Arduino Shield initial design (lower side)

Appendix F

Miscellaneous Pictures and Drawings

Some of the pictures of the first prototype of the Smart Walker and the progress of its fabrication and testing are being shown in this section.



(a) (b)
Figure F-1 Smart Walker (a) CAD model (b) 1st prototype

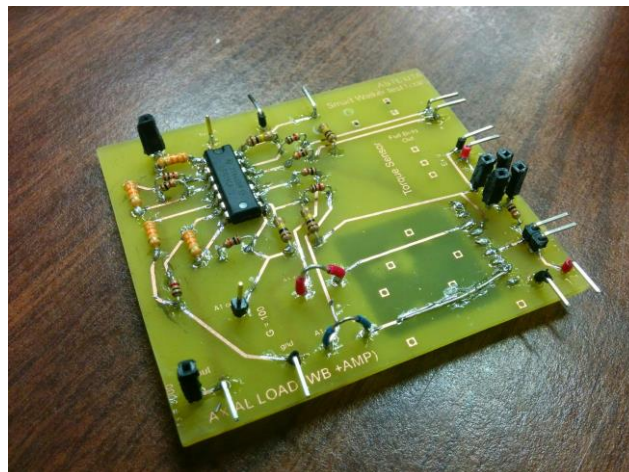


Figure F-2 First PCB prototype fabricated to test axial load and torque strain gages

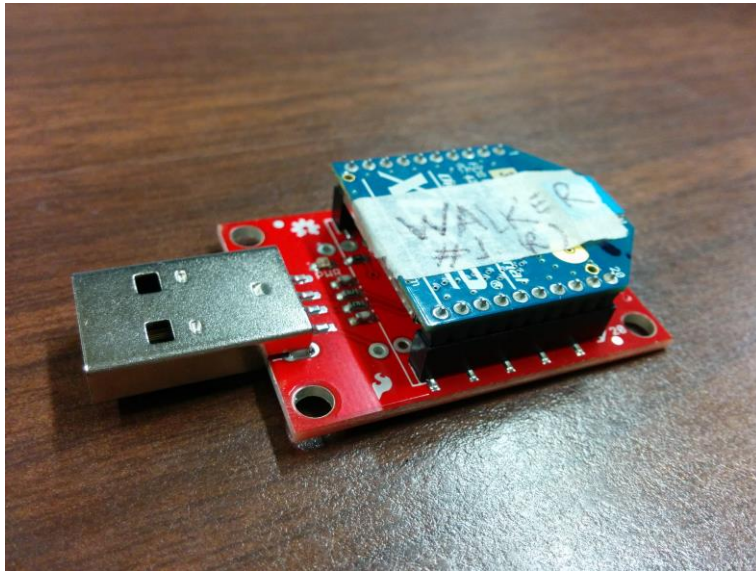


Figure F-3 Smart Walker Xbee antenna on USB dongle

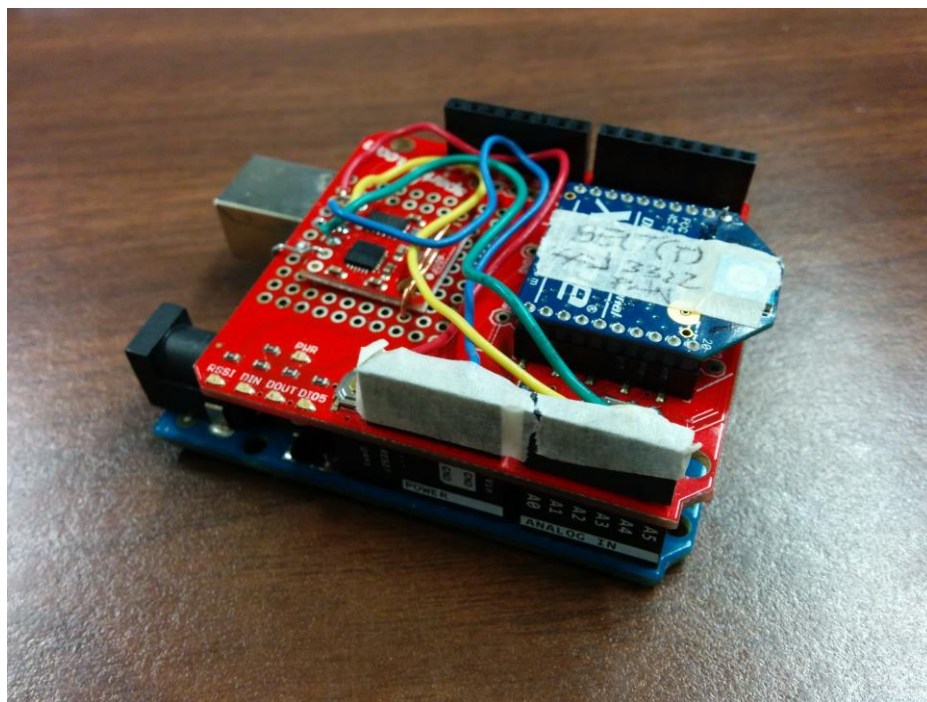


Figure F-4 Belt Arduino UNO with IMU and Xbee shield/antenna

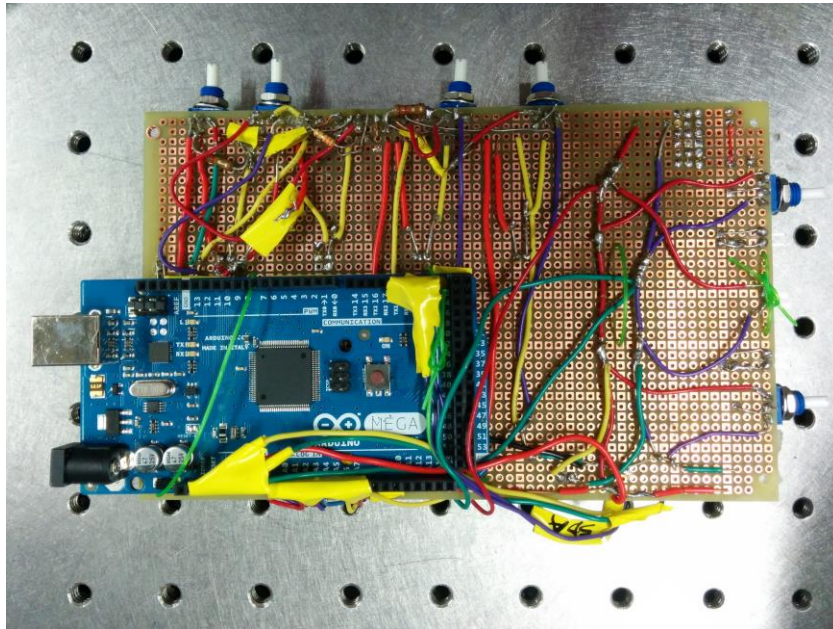


Figure F-5 Smart Walker Circuit board - 1st prototype (Arduino Side)

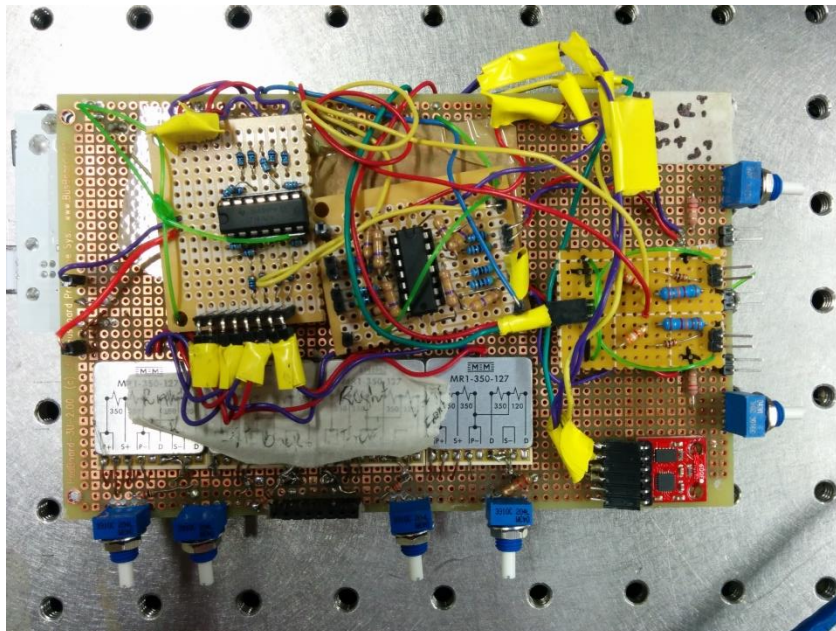


Figure F-6 Smart Walker Circuit board - 1st prototype (circuit Side)



Figure F-7 Smart Walker 1st prototype circuit board case mod CAD model

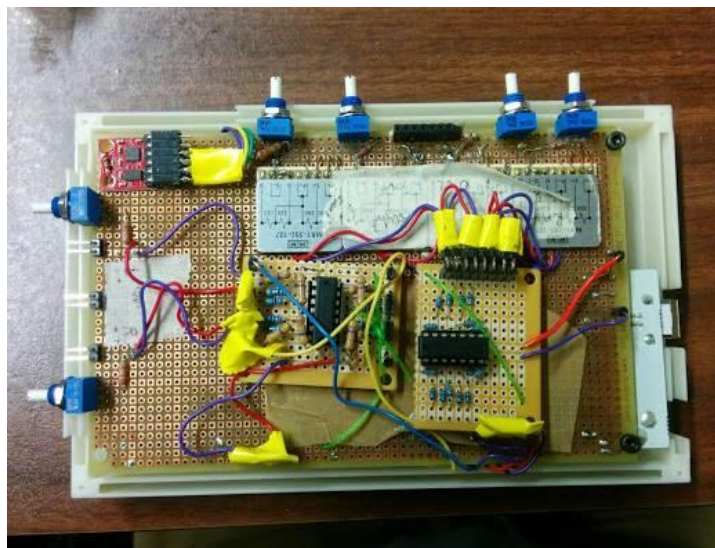


Figure F-8 Smart Walker 1st prototype circuit board mounted on bottom case side.

References

- [1] Liu, H., Huang, H. and Nana A., *SmartWalker – A New Design of Rolling Walker to Reduce Falls and Walker-use Related Side Effects, Texas MRC Program, Proposal Application Form for 2014*, 2014.
- [2] Figliola, R. S. and Beasley D. E., *Theory and design for mechanical measurements*, 5th ed. Hoboken, NJ: Wiley, 2011.
- [3] Furman, B. J., *Force, Torque, Stress, and Strain Measurement*, ME120 Experimental Methods, San Jose State University, 2006
- [4] Micro-Measurements, *Strain Gage Selection: Criteria, Procedures, Recommendations*, TN-505-4, Doc No. 11055, Vishay, Precision Group, 2010.
- [5] Micro-Measurements, *General Purpose Strain Gages – Shear/Torque Pattern*, 187UV, Doc No. 11244, Vishay, Precision Group, 2010.
- [6] Micro-Measurements, *Strain Gage Adhesive, M-Bond 200*, Doc No. 11010, Vishay, Precision Group, 2012.
- [7] Micro-Measurements, *Strain Gage Installation with M-Bond 200 Adhesive, B-127-14*, Doc No. 11127, Vishay, Precision Group, 2011.
- [8] Micro-Measurements, *Surface Preparation for Strain Gage Bonding, B-129-8*, Doc No. 11129, Vishay, Precision Group, 2011.
- [9] Micro-Measurements, *Information and Selection Chart*, MR-Series Bridge Completion Modules, Doc No. 11042, Vishay, Precision Group, 2013.
- [10] Arduino LLC., 2016, "Arduino MEGA 2560", from <https://www.arduino.cc/en/Main/ArduinoBoardMega2560>
- [11] Ochoa, G. J., 2011, "A Swarm of Xbees! Arduino Xbee Wireless & More", from <http://bildr.org/2011/04/arduino-xbee-wireless/>

- [12] Bldr.org, 2012, "Stable Orientation – Digital IMU 6DOF + Arduino", from
<http://bldr.org/2012/03/stable-orientation-digital-imu-6dof-arduino/>
- [13] Arduino LLC., 2016, "SoftwareSerial Library", from
<http://www.arduino.cc/en/Reference/softwareSerial>
- [14] Arduino LLC., 2016, "analogReference()", from
<https://www.arduino.cc/en/Reference/AnalogReference>
- [15] SparkFun Electronics ®, 2016, "XBee Shield Hookup Guide", from
<https://learn.sparkfun.com/tutorials/xbee-shield-hookup-guide>
- [16] Digi International Inc., 2016, "Basic XBee 802.15.4 (Series 1) Chat", from
<http://www.digi.com/blog/xbee/basic-xbee-802-15-4-chat/>
- [17] LabVIEW MarkerHub, "LINX", from
<https://www.labviewmakerhub.com/doku.php?id=libraries:linx:start> (LINX)
- [18] Margolis, M., *Arduino Cookbook*, 1st Ed. Sebastopol, CA: O'Reilly, 2011.
- [19] Evans, B. W., *Arduino Programming Notebook*, 1st ed. San Francisco, CA:
Creative CommonsAttribution-Noncommercial-Share,2007.
- [20] Adafruit ®., 2014, "Force Sensitive Resistor (FSR)", from
<https://learn.adafruit.com/force-sensitive-resistor-fsr/using-an-fsr>
- [21] Padmanapan S., *Workshop on SMT Stencils*, SMTA Chennai, 2014.
- [22] Seifert, K. and Camacho O., *Implementing Positioning Algorithms Using Accelerometers*, , AN3397, Rev. 0. Tempe, AR: Freescale Semiconductor, Inc. 2007.
- [23] Lawrence, K., *ANSYS Workbench Tutorial Release 14*, 1st Ed. Mission, KS: Schroff Development Corporation, 2012.
- [24] Digi International Inc., Demystifying 802.15.4 and ZigBee®, White Paper, 9100991437, A1/208, 2008.

- [25] Faludi, R., *Building Wireless Sensor Networks*, 1st Ed. Sebastopol, CA: O'Reilly, 2011.
- [26] Digi International Inc., *XBee/XBee-PRO® 802.15.4 Professional Kit - Getting Started*, 90002159_A, 2012.
- [27] Digi International Inc., *XBee®/XBee-PRO® RF Modules, Product Manual v1.xEx – 802.15.4 Protocol*, 90000982_P, Minnetonka, MN: 2014.
- [28] University of California Santa Barbra, *Lab Manual “Strain Gage Sensors”*, Department of Mechanical and Environmental Engineering, 2013.
- [29] Hibbeler, R.C., *Mechanics of Materials*, 8th Ed. Boston, MA: Prentice Hall, 2011.
- [30] Norton, R. L., *Machine Design – An Integrated Approach*, 4th Ed. Boston, MA: Prentice Hall, 2011.
- [31] Logan, D. L., *A First Course in the Finite Element Method*, 3rd Ed. Pacific Grove, CA: Brooks/Cole, 2002.
- [32] Young, W. C., Budynas, R. G., *Roark's Formulas for Stress and Strain*, 7th Ed. New York, NY: McGraw-Hill, 2002.

Biographical Information

Mauricio Jaguan Nieves started his undergraduate education in the Simon Bolivar University in Caracas, Venezuela where he coursed basic mathematics and physics courses before transferring to the University of Texas at Arlington. In UTA he graduated magna-cum-Laude as a bachelor of science in Mechanical Engineering and Aerospace Engineering. He immediately pursued further education in Mechanical Engineering and enrolled in a Master of Science program at his alma mater the University of Texas at Arlington. He works currently in Clyde Bergemann Power Group as applications engineer in the Air Pollution and Material Handling division. His main research interests have been in data acquisition systems and currently he seeks further preparation in machine learning for autonomous vehicles.