THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENTS OF ELECTRICAL ENGINEERING AND MECHANICAL AND NUCLEAR ENGINEERING

A STATE-OF-CHARGE ESTIMATOR FOR A SEMI-AUTONOMOUS ELECTRIC WHEELCHAIR

CHRISTOPHER X. MILLER
SPRING 2016

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree in Electrical Engineering
with interdisciplinary honors in
Electrical Engineering and Mechanical Engineering

Reviewed and approved* by the following:

Timothy Kane
Professor of Electrical Engineering
Honors Adviser
Thesis Supervisor

Sean Brennan
Associate Professor of Mechanical Engineering
Honors Adviser

Constantino Lagoa
Professor of Electrical Engineering
Faculty Reader

*Signatures are on file in the Schreyer Honors College.

Abstract

According to the U.S. Census, nearly three million individuals in the United States rely on wheelchairs, a large portion of which are electric wheelchairs, in order to regain lost mobility. Consequently, these individuals depend on a reliable power system; if a wheelchair′s battery power depletes without the user being aware, the individual may become stranded, further limiting his or her freedom of mobility and potentially placing the user in a harmful situation. This research seeks to develop a State-of-Charge (SOC) estimator for the batteries of an electric wheelchair. A second-order equivalent circuit battery model is developed and parametrized for a wheelchair′s lead-acid battery pack. The inputs to the algorithm are battery voltage and current and the output of the algorithm is the battery pack′s estimated state of charge. To simplify the SOC estimation, this algorithm models a vehicle′s fuel gauge. When a vehicle′s fuel tank is nearly full or nearly empty, a fuel gauge presents the user with a full or empty reading. Outside of these regions, the fuel gauge varies directly with the fuel remaining in the vehicle′s tank. Similar to a vehicle′s fuel gauge, the algorithm yields the least accurate estimates of the wheelchair′s SOC in the maximum and minimum SOC regions. These extrema are defined by the non-linearities present in the Open Circuit Voltage (OCV) SOC curve. Consequently, a coulomb accumulator is incorporated to estimate energy usage in these regions. A Kalman filter is incorporated to estimate SOC in the linear region of the OCV-SOC curve.

This thesis presents the development of an autonomous wheelchair platform and the subsequent implementation of the aforementioned battery state estimator.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I owe a great deal of thanks to the many people who helped me complete this thesis and my undergraduate degree. Aside from offering technical advice, many picked me up when I was down, supported me when I was lost, or, in one case, funded my college career. I didn't single handedly complete this research, rather, I'm simply the guy who kept all of the paper in order and glued together the many elaborate, electrified pieces of this puzzle.

I'd like to extend my thanks to the Student Space Programs Lab and the Applied Research Laboratory for affording me the opportunity to use their electronic load banks and other test equipment necessary in characterizing my circuits and battery packs.

I'd like to thank Dr. Constantino Lagoa for serving as a thesis reader and for providing meaningful feedback.

I need to thank Jari Safi for helping me debug the battery model, giving me the idea for the fuel gauge method and, most importantly, for reminding me to keep my sense of humor and faith in Timmy.

I would like to thank Kelilah Wolkowicz for mentoring me throughout my entire time in the lab, tolerating my awful sense of humor, working along side me for the entire duration of this project, and for being one of the best colleagues for which one could ever ask.

I need to thank my mentors Chris 'Mik' McKay and Abhi 'Dave Witson' Chivukula, for believing in me when I had no faith, encouraging me when I was lost, and for teaching me to sometimes break the rules. Mik, thank you for showing me that "... [I'm] better than that." Abhi, thank you for teaching me the true definition of excellence.

I very much thank my aunt Joanne for being a very patient and sharp sounding board, teaching me I can surprise myself, forcing me to think through many major decisions, and helping me keep it together during my many 'impossible' moments.

I need to extend a word of thanks to my academic adviser, Dr. Tim Kane, for his many bits of life and academic advice, reminding me to take life less seriously, for keeping me from imploding both physically and mentally, and for allowing me to discover my path in college and beyond.

I exorbitantly thank Dr. Sean Brennan for teaching me to be a good researcher, a good experimentalist, and a great engineer. Further, for driving me to work harder than ever before, to think in ways not previously known, for showing me that I can balance having a life and a career, and for showing me an example of a person I aspire to be.

I thank the late mad scientist I knew as my dad, Chris (June 1963 - March 2009), for inspiring me to become an electrical engineer, leaving behind a basement full of tools, parts and memories, many words of wisdom, challenging me as a kid and as an adult, and supporting me even in death. I'm sorry for the past and thank you for the present and the future. I hope I've made you proud.

Finally, I extend the most of my thanks to my mom, Mary, for not only financing my college career, but for teaching me the most important lessons in life: resilience, strength, kindness, hard work, and patience. I know I'm a difficult person with which to deal; I'm harsh, I have numerous odd idiosyncrasies, and I can be a jerk, but you stand by me regardless. To articulate my gratitude would take a lifetime. Without you, none of these thanks, this thesis, my college education, or my career would be possible. Thank you, Mom. I love you.

Chapter 1

Introduction

1.1 Motivation

Nearly 3.3 million Americans depend on a wheelchair to meet their daily mobility needs [1] and a large percent of all wheelchair users rely on an electric wheelchair. Furthermore, wheelchair use is expected to increase drastically with the advancing age of the baby boomer generation; some even believe usage may double [2, 3]. Users range from young adults temporarily bound to a wheelchair to recover from an athletic injury to the highly disabled coping with diseases such as Cerebral Palsy or amyotrophic lateral sclerosis (ALS). Of interest to this research is are those using electric wheelchairs, more specifically, the highly disabled and their interface with electric wheelchairs.

According to [4], up to 10% of electric wheelchair users find electric wheelchairs extremely difficult to use. Further, in a review of electric wheelchair trends, Simpson *et al.* stipulates that nearly 61 to 91 percent of wheelchair users would benefit from a smart wheelchair [5] some of the time. In other words, users would not necessarily need the intelligent features at all times, but nearly all users would benefit from smart capabilities at least some of the time. While significant research has been conducted since the 1980s in the area of smart wheelchair technology, wheelchair energy usage estimation remains an area of little focus over the past 30 years. This is a critical issue to electric wheelchair users, especially those using smart wheelchairs to overcome their inability to operate a chair: without a proper estimate of the energy remaining on the battery pack, users risk running out of energy and becoming stranded.

The most widely cited wheelchair power estimation paper, A battery state-of-charge indicator for electric wheelchairs by Aylor *et al.* [6], is from 1997. Due to increasing adoption of electric vehicle technologies over the past two decades, power estimation methods have emerged that

advance on this seminal work. Specifically, current state-of-charge estimation techniques increasingly use a model-based approach to predict the charge remaining on the battery whereas [6] uses an open-circuit voltage technique. Further, since 1997, only a handful of papers have been written furthering the field of energy state estimation as applied to electric wheelchairs; discussion of these is presented in the literature review chapter of this thesis. A large majority of published research focuses on novel techniques of supplying energy to powered wheelchairs (e.g. hydrogen fuel cells) and skims the topic of estimating the electric range of the wheelchair.

1.2 Goals

The first goal of this research is to design and build an expandable, robotic electric wheelchair platform for smart wheelchair research. A donated electric wheelchair was completely stripped of its components, excluding the frame, its batteries, and its motors. The wheelchair was retrofitted with new power, computational, sensing, input, and drive-control systems. These modifications enable the power analysis presented in this thesis, and also provides other researchers with both a simple ground robot for algorithm development and also a physical platform to mount and test smart wheelchair technologies.

The second goal of this research is to develop a battery state-of-charge estimator for the electric wheelchair. To meet this goal, hardware was developed to monitor the power usage by the wheelchair. This hardware was integrated on to the aforementioned wheelchair platform. Next, the wheelchair batteries were characterized and an equivalent circuit model was developed. Then, a state-of-charge estimation algorithm modeling the batteries range similar to a vehicles fuel gauge - was designed and simulated. Finally, the charge estimation algorithm was realized, verified, and bench marked on the wheelchair platform.

1.3 Outline of Remaining Chapters

The remainder of this thesis is organized as follows: Chapter 2 provides an overview of smart wheelchair technologies, energy estimation literature related to powered wheelchairs, and a brief discussion of relevant battery state-of-charge estimation techniques. In Chapter 3, the design and development of the wheelchair platform is presented, and a detailed discussion of the power usage monitoring hardware is given in Chapter 4. Chapter 5 presents the mathematical equations used to develop the battery model used in this thesis. Chapter 6 briefly discusses the processes used to characterize and parameterize the wheelchair batteries to fit the model. State-of-charge esti-

mation methods are expanded upon and discussed in Chapter 7. Testing procedures and system realization are discussed in Chapter 8 and the estimators results are discussed in Chapter 9. Finally, conclusions and future works derived from this research are detailed in Chapter 10.

Chapter 2

Literature Review

2.1 Introduction

The aim of this chapter is to provide a review of past and ongoing research related to the design, guidance, and automation of electric and/or smart wheelchair technologies. The summary focuses particularly on power estimation in electric wheelchairs, as there is not yet widespread use of model-based energy monitoring for electric wheelchairs. Beginning with section 2.2, a review of existing smart wheelchair technology will be presented. Then, in section 2.3, a review of energy usage monitoring techniques in electric wheelchairs will be discussed. Finally, section 2.4 reviews the methods used in battery modeling and State-of-Charge (SOC) prediction.

2.2 Overview of Wheelchair Technology

One can readily find research starting in the late 1980s focused on autonomous and smart wheelchairs. In this time, various research groups have also constructed numerous smart wheelchairs for the purpose of realizing sophisticated navigation and localization algorithms designed for ground robotics [4]. Starting in early 2000s, it appears that a paradigm shift began; namely, research in smart wheelchairs is shifting from autonomy algorithms to shared controlled strategies and new human interfaces such as pupil tracking and brain-computer interfaces. This shift is a result of users desire to retain as much of their autonomy for as long as possible [4]. Shared control systems allow the user to maintain control of their wheelchair system while the onboard computer prevents dangerous maneuvers and aids users in navigating difficult situations [7, 8].

2.2.1 Madarasz *et al.* of Arizona State University

Madarasz and co-authors are regarded in this thesis as the first published instance of an automated electric wheelchair; the Madarasz electric wheelchair was built by the Arizona State University in the late 1980s [9]. This wheelchair was designed to transport people between rooms within an active office environment when provided a known destination given prior knowledge of its present location. Obstacle avoidance algorithms to avoid collisions with persons and other possible obstructions encountered in a typical office were implemented on the platform to allow for use in a real world environment. These results are very advanced for the time of publication.

The technology needed for implementation of the above algorithms is revealing in that the same components are used as today, but of more limited quality due to technological challenges of the day. The wheelchair platform was an electric wheelchair equipped with an IBM Portable PC, a 128 x 128 pixel digital camera fitted with a wide angle lens, and an ultrasonic rangefinder capable of scanning a full 360 field-of-view in 3 intervals.

2.2.2 NavChair

Developed in the late 1990s by the University of Michigan, the NavChair demonstrates an advancement in smart wheelchair research from the Madarasz wheelchair [10]. The NavChair was capable of operating in three modes: general obstacle avoidance, wall following, and door passage. The algorithms implemented on the NavChair were notably focused on obstacle avoidance, including an implementation of vector field histogram (VFH) and minimum vector field histogram (MVFH) methods; these methods push and pull the wheelchair to or from obstacles to avoid collisions. The NavChair was ultimately developed to explore shared-control strategies, i.e. methods where both the human and the computer collaborate in decision making and the computer is capable of adjusting to the humans inputs. This is a nascent development of a new goal of smart wheelchair research, shifting from the view of a wheelchair as simply a robot conveying a person to one that interacts with the user.

Technologically, the NavChair was an advancement over the work of the prior decade. It consisted of an electric wheelchair retrofitted with a DOS-based computing system, the original joystick, 12 front-facing ultrasonic, and a module containing the necessary support circuitry. The NavChair is one of the first systems to offer assistance to a user with disability rather than perform complete autonomy and it is one of the first systems to utilize many parallel sensors for more advanced decision making.

2.2.3 LURCH

Developed in 2012 by, A. Bonarini et al. at the Polytechnic University of Milan, LURCH is a smart wheelchair utilizing a number of human interface and autonomy modes [11]. The wheelchairs architecture consists of a localization module, a planning module reliant on a modified A* navigation algorithm, and a control module which uses fuzzy logic to implement trajectory planning. This platform represents an advancement to the state-of-the-art due to its numerous interface methods. First, the wheelchair utilizes a joystick interface which allows for both manual driving and shared-control strategies (e.g. a user may not drive into an obstacle). Second, LURCH utilizes a touch-screen interface for those lacking the dexterity required to use a typical joystick. The touch screen allows for the selection of low-level commands (e.g. forward, backward, etc.) and high-level commands (e.g. living room, office, etc.). Third, the system realizes an electromyographic (EMG) interface system that allows users to send high-level commands to the wheelchair using facial muscles. Finally, LURCH uses a P300-based brain computer interface to allow highly disabled persons send high-level commands to the wheelchair using their thoughts alone. This wheelchair system is one of many [7, 8] that demonstrates the migration in assistive technologies from using ground robotics techniques for control to collaboratively-controlled robotics.

The LURCH platform also shows technological advances over the 1990s wheelchair models: it consists of a modified electric wheelchair retrofitted with a LiDAR for obstacle detection, an upward-facing camera for indoor localization, wheel encoders for odometry measurements, a touch screen interface, an OCZ-brand game controller to serve as the EMG interface, a BCI system, and a customized computing system to process sensor data and implement navigation algorithms.

2.2.4 RobChair

The wheelchair developed by Lopes *et al.* at the University of Coimbra, Polo II in Portugal in the early 2010s focused on the implementation and refinement of a brain-computer interface [12]. Their ultimate goal was to improve the state of collaborative control research in smart wheelchairs. The Lopes wheelchair implements a-priori occupancy grid map of the environment for navigation. To navigate around obstacles encountered, the wheelchair utilizes an enhanced vector field histogram method, similar to that implemented by the NavChair. To find its location within the map, a grid Markov localization system was realized; this system relies on a grid of posterior, discrete probabilities updated with sensor data collected as the RobChair moves. A P300-based brain-computer interface provided seven possible user-selected commands as inputs for the wheelchair. A two-layer controller was used to provide inputs to the wheelchair. The first layer, a Virtual-

Constraint Layer (VCL), constrains possible user selections based upon given situations (e.g. obstacles). The second layer, the intent matching layer, predicts the users next input based upon possible options, as described in the VCL, localization data, and prior user input.

The RobChair design reflects the current approach of using a very large number of parallel sensors on smart wheelchairs. It consists of an electric wheelchair retrofitted with 12 IR proximity sensors, 12 IR rangefinders, an ultrasonic rangefinder system, a LiDAR system, low-resolution cameras, a magnetic sensor ruler, and the brain-computer interface system [13].

2.2.5 The Matt Barnes Wheelchair

In response to the need for a robotic platform and the desire to do autonomous wheelchair research in the Intelligent Vehicles and Systems Group at Penn State, Matt Barnes constructed another autonomous wheelchair platform [14]. This platform consisted of an onboard computer system running ROS, wheel encoders, a Hokoyo brand LiDAR module, and an xPC. This wheelchair provided the basis for the wheelchair to be presented in this research. This wheelchair was one of the first wheelchairs relying on ROS to manage algorithm execution, sensor interfaces, and data capture.

2.3 Overview of Energy Estimation in Electric Wheelchairs

As demonstrated in section 2.2, significant research has been conducted in the area of intelligent electric wheelchairs. Notably absent from the literature is a large amount of research aimed at energy estimation and power wheelchair electric range. Exceptions include the research conducted by Cooper *et al.* to determine the driving habits of electric wheelchair users. Their research showed that average powered-wheelchair users may travel less than 8 km per day [15]. However, [15] did not include information regarding the electric range or number of recharges for a given day; only estimates based upon a users daily driving habits were presented. In a follow-up work, Cooper *et al.* estimates the electric range of multiple wheelchairs; however, no general consensus is presented as the ranges vary from 23.6 km to 57.7 km [16].

Aylor *et al.* designed a simple approach to estimate the State-of-Charge (SOC) of the battery by measuring the open circuit voltage (OCV) of a wheelchairs battery [6]. For the early 1990s, their estimator yielded results comparable to industrial battery fuel gauges for level surfaces. However, on sloped surfaces, this technique lost some of its initially determined accuracy. The methods developed by Aylor *et al.* are presently the most widely cited methods in electric wheelchair battery

SOC estimation. Chen *et al.* developed a system to estimate the remaining SOC and electric range on a wheelchair battery using fuzzy logic and neural networks [17]. Their results indicated these methods are feasible on electric wheelchairs; however, these methods are atypical in the area of battery research which typically prefers a model-based approach for energy estimation. Further, Chen *et al.* used a lithium ion battery as their energy source, whereas most wheelchairs use lead acid batteries as energy sources.

In recent years, additional methods of extending battery capacity in wheelchairs have been presented. Bouquain *et al.* presented a method to extend the range of an electric wheelchair by using a hydrogen fuel cell and a DC-to-DC converter to provide constant power to a wheelchair with slowly changing dynamics (e.g. constant velocity, straight line motion) [18]. As dynamics increase (e.g. sudden turns), a lead acid battery will source power to the wheelchair while the fuel cell builds up the desired power. Yang *et al.* presented a different hybrid hydrogen fuel cell and battery wheelchair power supply [19]. The authors proposed a system where a primary battery is sourcing power to the wheelchairs drive train while a secondary battery is either idling or being recharged by the hydrogen fuel cell. When the primary battery′s voltage decreases below a particular cutoff voltage, the battery packs switch roles; the secondary battery sources current to the drive train and the primary battery pack is recharged by the hydrogen fuel cell.

2.4 Battery State Estimation Techniques

Since the advent of smart phones, portable devices, and electric vehicles, both consumers and researchers alike have yearned for improved means of predicting the remaining state of charge on a battery pack. As a result, noteworthy literature in the area of battery state estimation exists. Battery estimation literature tends to focus on methods of modeling, parameter determination and, estimation techniques. For accessibility, this section will be split into sections discussing modeling, estimation, and parameterization. It must be noted that this section is not a meant to provide a comprehensive review, but rather a brief synopsis to present some of the methods necessary in understanding battery estimation.

2.4.1 Modeling

In the area of battery modeling, there exists two primary types of models: physics-based models and equivalent circuit-based models. Physics-based models attempt to predict the molecular interactions occurring within the battery, whereas equivalent circuit models treat batteries as a

combination of series-connected, passive circuit components. For the past fifteen years, circuit-based models have been the most widely used models in battery estimation [20]. For the purpose of this research, only equivalent circuit models will be investigated.

Zhu *et al.* proposed a 4th-order equivalent circuit model for modeling lead-acid batteries [21]. This model consisted of an ideal power source whose voltage would drop linearly with depletion in SOC and three passive circuit networks. The first, purely resistive network, modeled the battery′s internal resistance. The final two networks, comprised of a parallel connected capacitor and resistor, modeled the packs first and second order dynamics. Coleman, *et al.*, proposed an identical circuit design for modeling lead acid batteries [22]. Figure 5.1 presents an example of an equivalent circuit model.

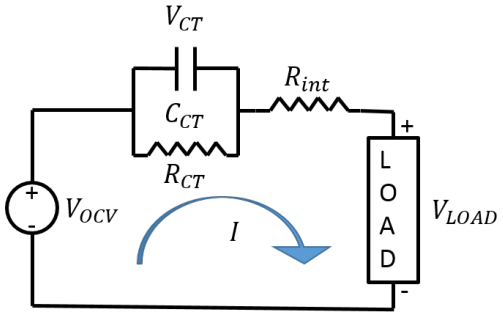


Figure 2.1: An example equivalent circuit battery model, specifically a second order model.

Zhu *et al.* further suggested to improve system dynamics, more series-connected, parallel RC pairs could be added to the model. Weng *et al.* investigated the higher-order models and discussed the negligible improvements and exponential computational costs incurred as more RC pairs are included [23]; [24] agrees with these stipulations.

Plett proposed a simplified model to that suggested in [21]; this model consists of one fewer series-connected RC component but proposes the use of a variable-impedance term to model hysteresis [24, 25, 26]. A later paper by Coleman *et al.* suggest a model identical to [24, 26], but ignores the hysteresis term [27] as they stipulate that an advanced estimator, such as an Extended Kalman Filter, may compensate for the dropped hysteresis term. Fortunately, [20, 24, 26, 28] agree that one of the largest sources of SOC estimator error is the estimator performance in the presence of noise and un-modeled dynamics, and thus, if a high-performing estimator is available, the hysteresis term may be ignored allowing lower-order models to be used while still obtaining sufficient accuracy.

2.4.2 Parameterization

After resolving a battery model, the models parameters need to be identified. According to [24], the first parameter to resolve is the battery′s Open Circuit Voltage (OCV) curve. The OCV curve is the voltage measured across the battery pack when the pack experiences little-to-no load after a long resting period. This load is typically defined as 0.1C, where C is the capacity of the battery under its typical load as defined by its Peukert curve. The OCV curve of a battery is a decreasing, non-linear function of SOC; the OCV can be determined by drawing a constant, low current from the battery pack and measuring the voltage across the pack until a cutoff voltage is reached [24]. Figure 2.2 presents an example OCV-SOC curve.

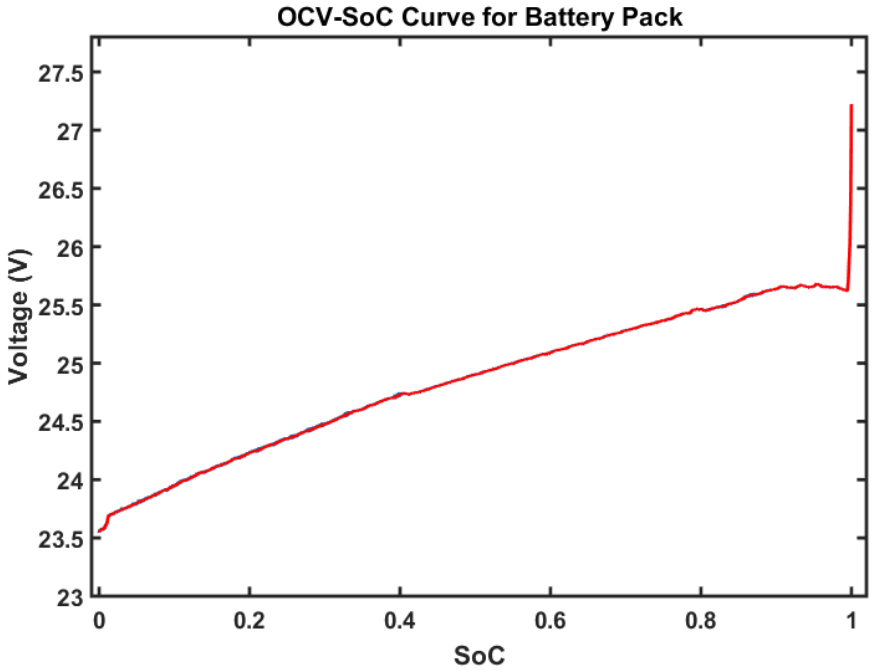


Figure 2.2: An example Open Circuit Voltage – State-of-Charge Curve. This figure presents an OCV-SOC curve for a lead-acid battery.

The remaining parameters, particularly the dynamic terms and internal resistance, may be resolved using a current pulse test. In a pulse test, an at-rest battery is forced to experience a large, sudden, draw of current for a short period of time. After the burst duration, the load is removed and the battery rests until changes in its voltage are negligible. Then, the battery is placed under load again and the process is repeated until the test has been completed. During this entire process, the voltage across the battery pack is sampled and stored until the test concludes. Idaho National Labs proposed the Freedom Car model to characterize a hybrid vehicle′s battery pack [29]; this

test suggested many quick bursts of current draw over a long period of time. On the other hand, Coleman *et al.* suggested a two-pulse test where the duration of rest period between the pulses was equal to the duration of the pulses′ duration period [27]. After measuring battery dynamics, many use a linear least squares regression to fit the parameters based on the responses observed [24, 26, 30]. These dynamics are likely related to the internal ion diffusion within the battery as the battery is perturbed from equilibrium. The dynamics of this behavior are infinite-order, yet can be approximated as first-order terms through an equivalent RC circuit within the battery model [24].

Many times temperature compensations are considered in battery modeling [31]. Since temperature conditions surrounding the battery pack are rather constant in this research, the battery′s behavior dependence on temperature shall not be addressed.

2.4.3 Estimation

Following the formation of a battery model, a method of estimating the charge remaining on the battery is to be developed. Early systems, such as that proposed in [6] suggested measuring the battery pack voltage alone. These systems are subject to drift with battery age, can accumulate errors, and show large transient errors due to a lack of dynamics. Following these methods, many investigated and continue to research coulomb counters which focus on measuring the charge units removed from the cell as a function of time [24, 32]. These methods, however, are subject to drift due battery dynamics and age.

To compensate for battery dynamics and age, many create a battery model and compare the model′s output voltage with the batter′s measurement. Then, using an estimator, the error between the model and measurement is minimized, correcting the errors accumulated in the model [24, 33]. A present trend is to use an Extended Kalman Filter to estimate the battery pack′s SOC [24, 26, 28] as an Extended Kalman Filter (EKF) is capable of linearizing a non-linear system such as a battery and namely its OCV-SOC curve. However, EKFs are slow to execute and can be computationally intensive due to linearization [33]. Others have proposed using an Unscented Kalman Filter (UKF) to estimate SOC [34]. While UKFs estimators yield exceptional results, like EKFs, they are computationally intensive and therefore costly to implement on many systems [33]. In order to balance computational cost and accuracy, some have proposed the use of a piecewise estimation scheme; Codeca *et al.* presented a mixed algorithm that combined Coulomb counting and estimation via a model-based approach [35]. This research treated the OCV-SOC curve as a piecewise function. In the non-linear region of the curve, the authors used Coulomb counting for SOC estimation. In the linear part of the curve, a model-based approach was used to estimate SOC.

Chapter 3

Wheelchair Hardware Design

3.1 Overview

This chapter will discuss the development of a custom semiautonomous wheelchair platform. The physical structure, power and safety architecture, computational capabilities, and sensor systems will be outlined in this chapter. This chapter aims to provide an overview of the present capabilities of the wheelchair constructed for the purpose of this thesis.

3.2 Wheelchair Hardware

The wheelchair platform used is a modified Jazzy Pride 6 manufactured by Pride Products Corp. USA and the starting platform is shown in Figure 3.1. The original wheelchair consisted of a metal wheelchair frame and two 24VDC motors whose maximum current draw did not exceed 30A per channel. To power the wheelchair, two series-connected UB12350 35Ah, 12V batteries were included. Further, the unmodified wheelchair included a joystick, a 24V DC battery charger, a proprietary DC motor controller capable of interpreting joystick commands and supplying power to the drive motors.

Figure 3.1: The Jazzy Pride 6 Electric Wheelchair by Pride Products Corp., USA.

To retrofit the wheelchair with all of the desired hardware, it needed to first be stripped down to remove components that were difficult to interface (the motor controller, for example), were unreliable (power systems), or whose input/output behavior was nonlinear (the joystick). First, all of the wheelchairs external plastics, electronics, and coverings were removed, leaving only the frame, batteries, motors, and seat. A new battery box was built around the batteries to create more storage. A shelving unit was designed and built on the back side of the wheelchair using the 80/20 aluminum framing system and acrylic. Two pieces of 80/20 were mounted vertically from the shelf to serve as mounts for a GPS and a LiDAR module. Around the wheelchair′s edges, an 80/20 frame was installed to allow for more sensors to be mounted. Figures 3.2 and 3.3 display completed views of the wheelchair. More pictures of the wheelchair can be found in Appendix A.

Figure 3.2: Side view of the autonomous wheelchair platform

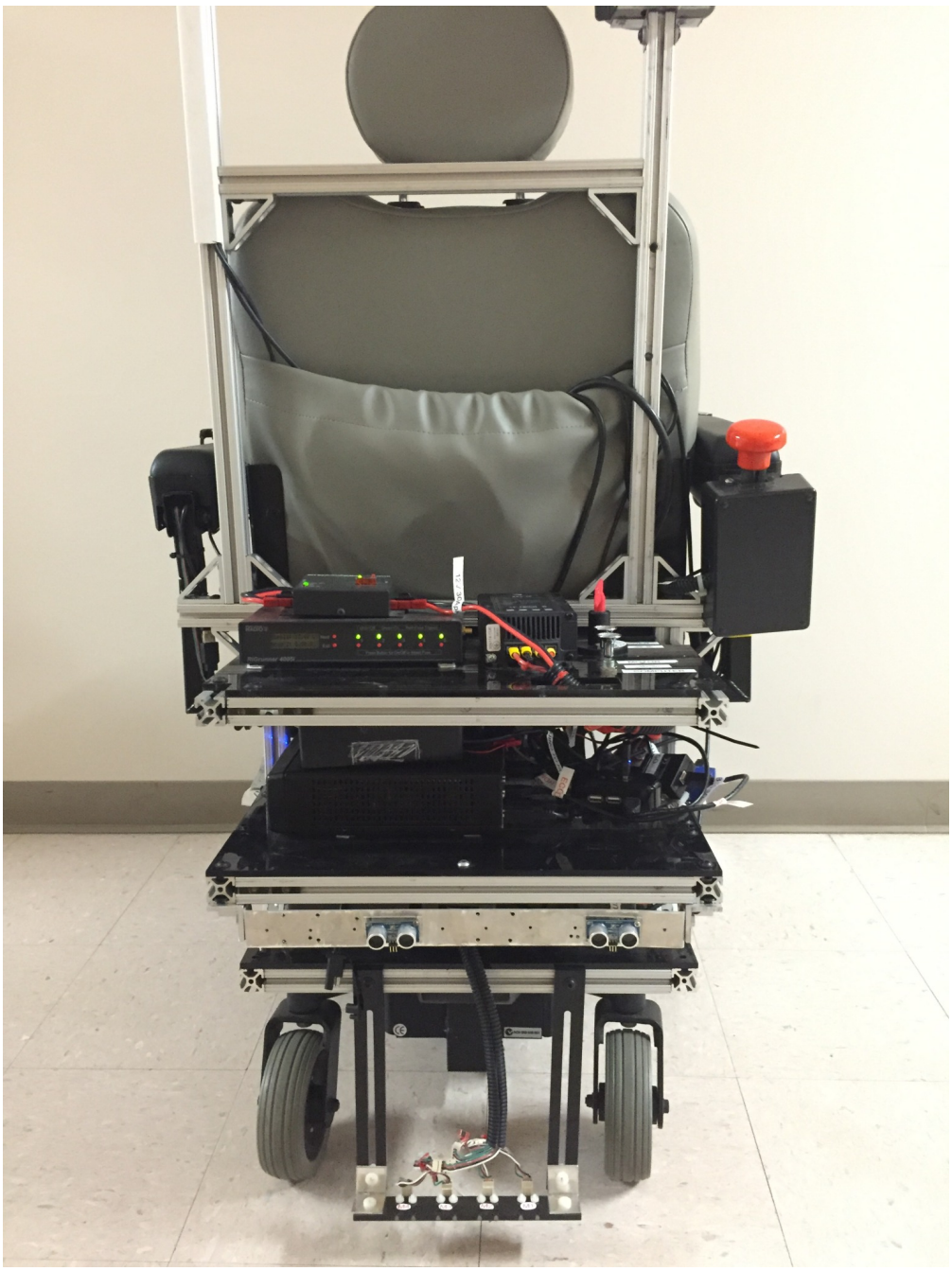


Figure 3.3: Rear view of the autonomous wheelchair platform

3.3 Power and Safety Architecture

The original wheelchair relied entirely on a single 24V rail provided by the battery system. The modified wheelchair required both a 24V rail and a 12V rail. Further, this power system needed to

automatically select to draw power from the battery pack or from a DC power supply as to allow for computational work without complete battery depletion. To implement the aforementioned system, two 12V series-connected batteries were installed in the wheelchair. A switch was placed between the batteries to break the series connection and to prevent parasitic currents from being drawn when the chair is not in use.

In parallel with the output of the battery pack, the original 24V lead-acid battery charger was installed. In series with the primary output of the battery pack, two 50A fuses were installed to limit current draw should there be a short within the power system. A current sensor was series-connected to the output of the battery pack to monitor the current leaving the pack. To monitor the voltage across the pack, a voltage sensor was installed[1]. Then, the output of the battery pack was split to source current for the 24V rail and the 12V rail. To simplify this discussion, the remaining power system will be described in two parts: the computing and sensing (12V) and the drivetrain (24V) systems. Figure 3.4 displays the power system with the aformentioned modifications.

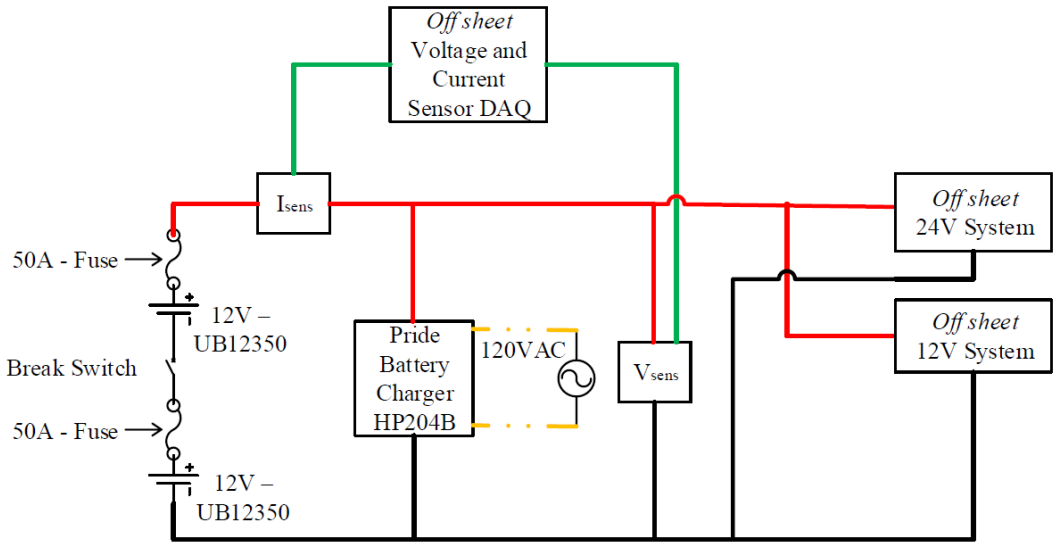


Figure 3.4: High-level schematic of the power system depicting the batteries, battery pack charger, and power subsystems

[1]Since chapter 4 focuses on the design and characterization of the voltage and current sensors, chapter three will not further describe these two components

3.3.1 Drivetrain

A key function of the power system is to supply current to the wheelchairs drive train. The output of the batteries is series-connected to a 45A fuse, an enable switch, and an emergency power shut off switch. Figure 3.5 highlights this emergency switch.

Figure 3.5: The emergency stop switch. This switch cuts power to the drivetrain system while leaving the computing system fully powered on.

A diode reverse-wired is parallel-connected to the 45A fuse to prevent fly-back currents. A 2k resistor is parallel-connected to both switches. The resistors and diodes were installed to allow for regenerative current from the motor controller to flow to ground during an emergency stop. The motor controller, a Roboteq MDC2230 shown in Figure 3.6, was series-connected to the output of the emergency power shut-off switch.

Figure 3.6: The Roboteq MDC2230 dual-channel brushed DC motor controller. Each channel can source a peak continuous current of 50A.

Both motors were connected to the motor controller, each via its own independently controlled channel. The MDC2230 motor controller requires an isolated 12V power source to power its logic circuitry. This allows the motor controller to function even in the event of a fault on the 24V line.

Since drawing from the 12V rail would create a ground fault loop, a second battery was installed on the wheelchair. This external 12V battery was series-connected to a 1A fuse, a 1A switch, and the motor controller. Figure 3.7 depicts the drivetrain power system diagram.

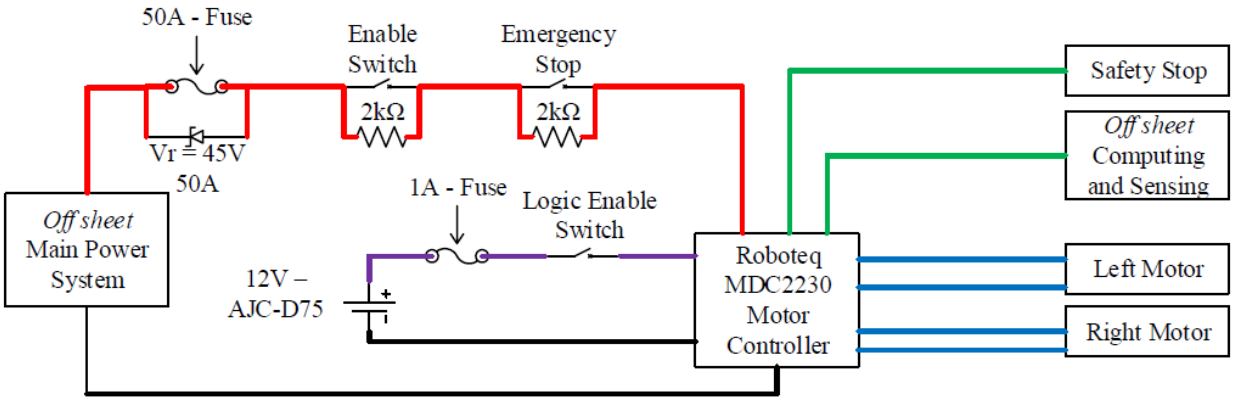


Figure 3.7: Schematic of the drivetrain depicting the safety switches, the motor controller, the secondary battery, and the motors.

3.3.2 Computing and Sensing

The second function of the wheelchair's power system is to supply power to the computing and sensing resources on the chair. The output of the 24V battery pack is series-connected to a Samlex SDC-30 24V to 12V DC-DC converter to step down the voltage from 24V to 12V. The output of the converter connects to the battery input on the ISOpwr by West Mountain Radio, shown in Figure 3.8.

Figure 3.8: The ISOPwr by West Mountain Radio, USA. This device is able to switch between the power supply and the battery pack. Whenever the power supply is energized, the supply sources the system current, else, current is drawn from the battery pack.

A 12V, 40A DC power supply was mounted on the wheelchair and connected to the PS input on the ISOpwr. The ISOpwr draws current from the energized power supply and automatically switches to battery power when the power supply is shut off. The ISOpwr automatically switches back to the power supply when the supply is re-energized. The output of the ISOpwr is series-connected to a 30A fuse, a 30A enable switch, and a PWRGuard Plus by West Mountain Radio. The ISOpwr was used to allow researchers to work on the computing and sensing system without draining the battery; in other words, the wheelchair could draw current from a wall socket or from the battery pack depending on the needs of the moment. The PWRGuard Plus, depicted in Figure 3.9, prevents over and under voltage conditions from damaging the computational and sensing system by disabling its output if the voltage falls below or above 11V and 15V, respectively.

Figure 3.9: The PWRGuard Plus by West Mountain Radio, USA. This device provides over current protection as well as over and under voltage protection to the computing and sensing system.

A West Mountain Radio RigRunner 4500i is series connected to the output of the PWRGuard Plus. The RigRunner, displayed in Figure 3.10, distributes the power from the 12V rail to the computer system, a powered USB port, and pair of DC-DC converters providing 3.3VDC and 5VDC to sensors. Figure 3.11 provides a schematic of this system.

Figure 3.10: The RigRunner 4000i by West Mountain Radio, USA. This device acts as a circuit breaker, power distribution, and power monitor for the computing and sensing system.

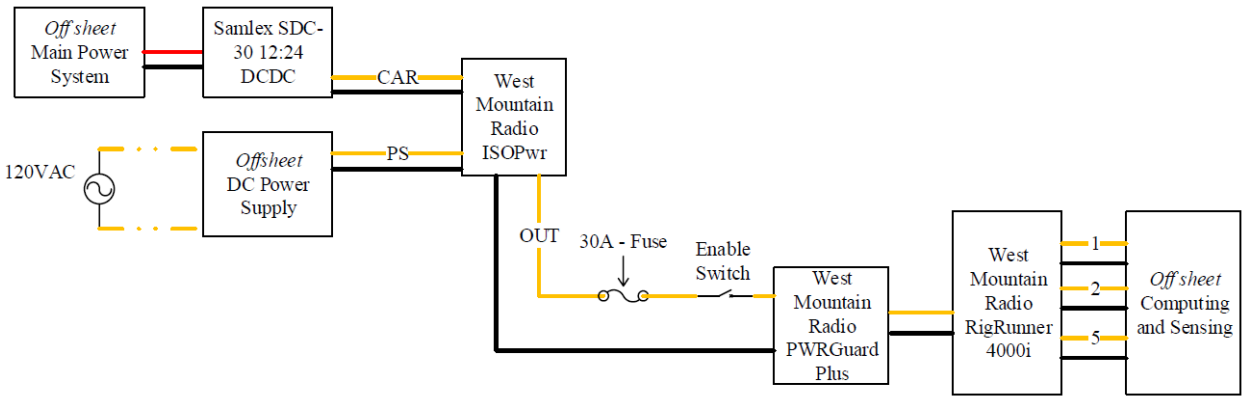


Figure 3.11: Schematic of the computing and sensing system depicting the enable switches, external power supply, voltage and current protection, and power distribution.

3.4 Computational

To control the wheelchair, a custom-built small form factor computer system was installed. This computer system′s specifications are described in Table 3.1.

Hardware Specification	Value
Processor	Intel Core i5, Haswell Quad Core 2.9GHz
RAM	16GB DDR3 RAM
Hard Drive	180GB SSD
GPU	Intel Integrated Graphics
USB Ports	2x USB 3.0, 5x USB 2.0
Other Interfaces	2x RS232, 2x RJ45, 1x DVI, 1x HDMI, 1x VGA
Power Supply	180W

Table 3.1: Specifications for the custom-built computer on the wheelchair platform

Ubuntu 13.04 LTS, a version of Linux, was installed on the wheelchair′s computer. Ubuntu was selected because ROS, the Robotic Operating System, was developed to run on Ubuntu. ROS, is a widely used, open-source software developed by Stanford University and Willow Garage; ROS provides a simple frame work for sensors to communicate with a computer system and allows users an easy method of quickly testing complex robotic platforms [36]. ROS is comprised of a series of regularly updated open-source libraries that serve as the systems framework. A user defines a series of nodes, or small programs, to perform a variety of tasks within ROS. Nodes can

transmit data by publishing ROS messages and read data by subscribing to a given publisher. This publisher-subscriber architecture also allows for users to easily read data from a given node for debugging or save published data for later processing. ROS nodes are written in either Python or C++, thus allowing easy modifications of the wheelchair control software.

To sample analog data from sensors and receive data from sensors reliant on GPIO buses or communication protocols such as I2C or SPI, Arduino Uno microcontrollers were used. The Arduino, shown in Figure 3.12, is a simple, open-source, hobbyist microcontroller with an onboard 10-bit ADC, 5-52 digital I/O lines, serial communication capabilities, and a simple-to-use programming environment [37].

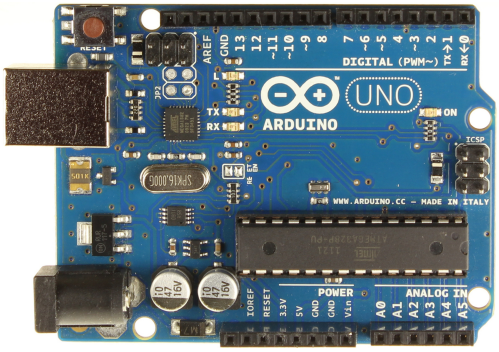


Figure 3.12: The Arduino Uno Microcontroller, a low-cost hobbyist microcontroller.

The Arduino′s simple interface and wide gamut of capabilities made it perfect for use with the wheelchair platform. Further, the Arduino can be directly interfaced to the ROS host, thus allowing the Arduino to act as either a low-cost data acquisition system or interface module for sensors.

3.5 Sensors

The wheelchair was retrofitted with a variety of sensors to interpret its environment. To monitor wheelchair odometry, one US Digital HB6M 10,000 count-per-revolution optical encoder, as depicted in Figure 3.13, was installed on each motor′s rear axel.

Figure 3.13: The US Digital HB6M 10,000 CPR optical encoder selected to be mounted on the rear axle of the wheelchair′s motors.

To read encoder data a custom interface based on an Arduino was used. These encoders were selected for their simple electrical and mechanical interfaces, rugged casing, and high precision [38].

To map the environment for subsequent navigation, a Hokoyu URF-04LX LiDAR was selected and mounted on the wheelchair. Figure 3.14 depicts this device.

Figure 3.14: The Hokoyu URF-04LX LiDAR module used to map the environment around the wheelchair.

The LiDAR was mounted at a slight downward facing angle on the taller of two masts installed on the wheelchair. The URG-04LX has a detectable range of up to 4 meters, a scanning rate of 10Hz, and a 240° field of view with 0.36° angular resolution [39]. Figure 3.15 depicts the location of the LiDAR and other sensors on the wheelchair.

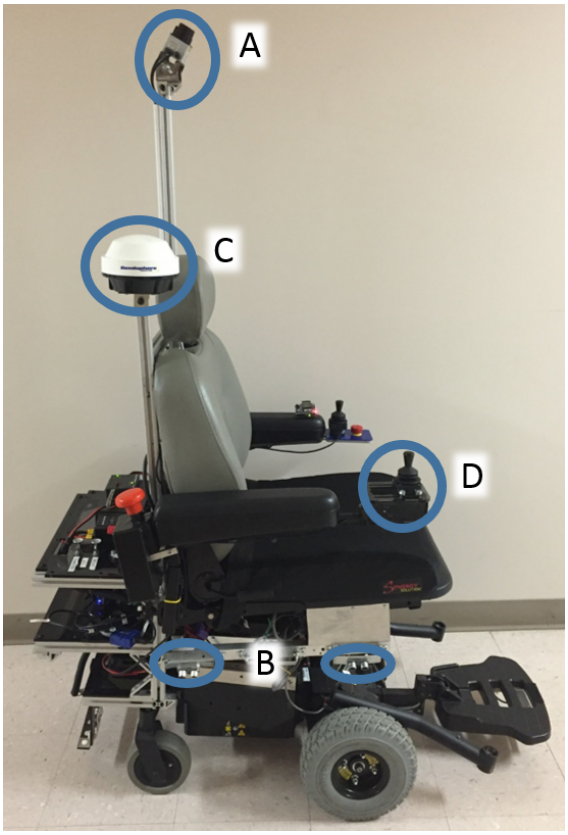


Figure 3.15: This side view of the wheelchair depicts A) the LiDAR′s position, B) the PING)))′s location, C) the position of the Hemisphere GPS, and D) the location of the Apem joystick

An array of Parallax PING))) ultrasonic range finders were installed around the wheelchair′s perimeter to detect positive obstacles such as people and furniture or negative obstacles such as curbs or potholes. These sensors, shown in Figure 3.16 , rely on a 40 kHz ultrasonic pulse and are capable of detecting objects within the range 3cm to 3m [40]. These sensors were selected for their aforementioned capabilities, low-cost, and their simple interface. The mounting of some of these sensors are depicted in Figure 3.15.

Figure 3.16: The Parallax PING))) Ultrasonic rangefinder used to detect small obstacles and objects in areas not visible to the LiDAR.

For outdoor localization, a Hemisphere A325 Smart GPS, shown in Figure 3.17, was installed on the wheelchair. The Hemisphere was installed on the shorter of the two masts on the wheelchair, as shown in Figure 3.15.

Figure 3.17: The Hemisphere A325 Smart GPS used for outdoor localization and navigation.

The Hemisphere A325 is capable of providing real-time kinematic localization data with precision of 2 cm. This GPS was installed on the wheelchair for its technical specifications and to allow for research to be conducted outdoors; many smart wheelchairs have been equipped for indoor use and few have been retrofitted for outdoor use [41].

Since the included joystick was unable to be re-engineered and had a large nonlinear dead-zone about the zero joystick input range, a new joystick was selected and installed on the wheelchair. The joystick selected, shown in Figure 3.18, was a HF11S10U, 2-axis Hall Effect USB joystick by Apem Inc.

Figure 3.18: The Apem 2-axis Hall Effect USB joystick used for manual control and navigation.

This joystick was selected for two reasons: first, the APEM joystick relies on Hall Effect sensors, therefore making it more accurate and less susceptible to noise when compared to its potentiometer-based counterparts. Second, this joystick relies on the Human Interface Device (HID) standard for communication with the computer system, thus making it very reliable and easy to integrate [42]. The location of the joystick is highlighted in Figure 3.15.

For a detailed design of the wheelchair′s sensor architecture, including a few sensors not relevant to this research, refer to Appendix B. Further, Appendix B includes further detail of the schematics presented in Figures 3.4, 3.7, and 3.11.

Chapter 4

DAQ Hardware Design

4.1 Overview

Critical to this research was the development of a high-fidelity power monitoring system. To monitor the wheelchair′s State of Charge (SOC), it was necessary to monitor the current leaving the battery pack and the voltage across the battery pack. This chapter details the specification of the Data Acquisition system (DAQ) developed to sample and send the data from the voltage and current sensors to the wheelchair′s computer. Furthermore, this chapter details the development and characterization of the voltage and current sensors.

4.2 Data Acquisition Hardware

As discussed in chapter 3, an Arduino was selected to serve as the interface between both analog and digital sensors. Initial testing indicated that the Arduino was capable of transferring data from its Analog-to-Digital (ADC) converted to the wheelchair′s computer at desired speeds; however, testing also indicated that the Arduino′s ADC was not yielding enough precision. As prior mentioned, the Arduino uses a 10-bit ADC, meaning the Arduino could resolve analog voltages up to +/- 0.0049V.

To improve this accuracy, an external ADC was selected. The Texas Instruments ADS1115 ADC, populated on a breakout board by Adafruit Inc., was selected. This breakout board is shown in Figure 4.1.

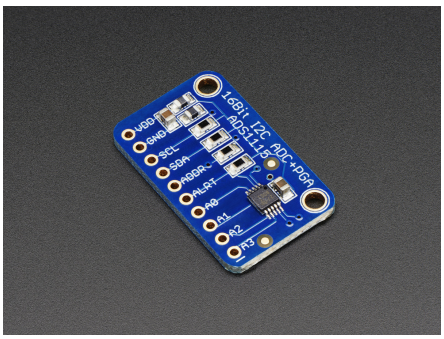


Figure 4.1: The Texas Instruments ADS1115 Analog-to-Digital Converter mounted on a Pololu brand breakout board.

The ADS1115 was selected for its high precision, pre-written Arduino drivers, speed, and number of channels. Table 4.1 presents the specifications and options used on the ADS1115 and Figure 4.2 presents a schematic describing the ADC and the other sensors.

DAQ Specification	**Value**
Resolution	16-bits, 7E-5V
Channels	4
Sampling Frequency	80Hz

Table 4.1: Specifications for the power monitoring DAQ, focused on the ADS1115 ADC specifications

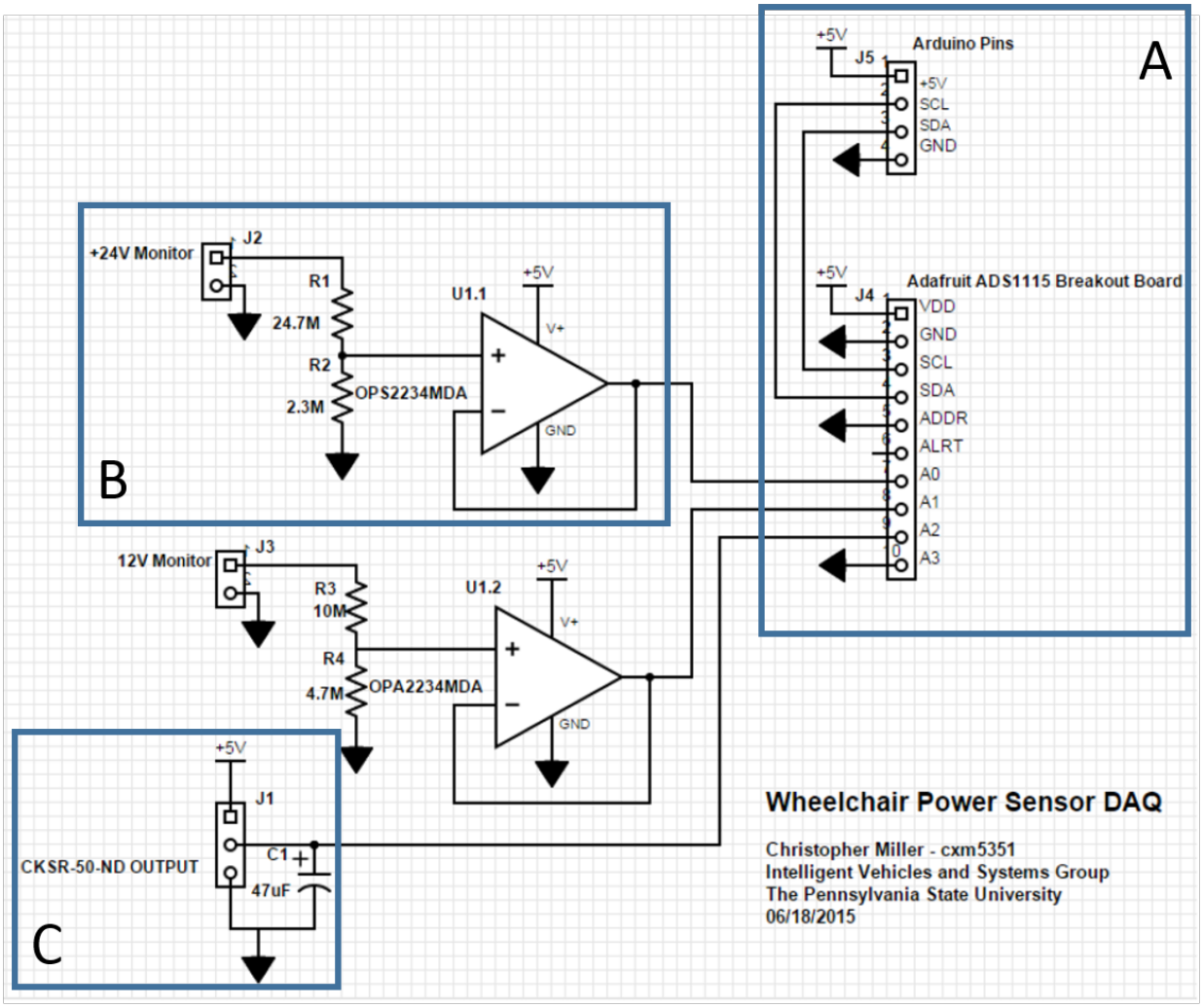


Figure 4.2: The power monitoring system schematic. A) Depicts the ADC, B) depicts the voltage sensor, C) depicts the current sensor.

4.3 Voltage Sensor

The ADS1115 can sample voltages from 0-5V in configuration described in Figure 4.2; therefore, the voltage needed to be scaled from 0-26V to 0-5V. To scale the voltage, a voltage divider was designed. The output of the voltage divider was connected to a single-rail op amp to remove any loading effects created by the ADC. This circuitry is highlighted in Figure 4.2. To characterize the voltage sensor, a calibrated DC power supply was connected to the input of the sensor. Then, the sensor was characterized over 0.00V to 30.00V in 0.50V steps. At each step, 250 samples were saved using the DAQ described in section 4.2. The results of this test can be found in Figure 4.3.

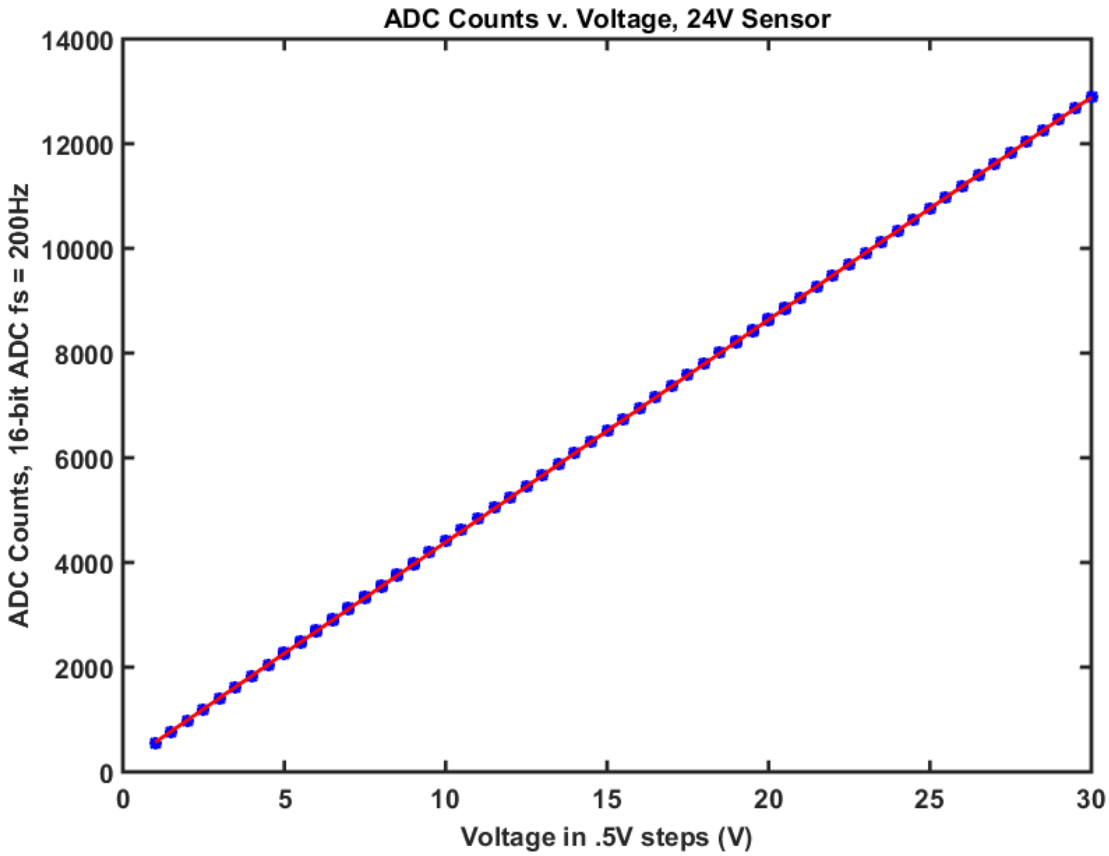


Figure 4.3: Calibration results and linear regression fit for the voltage sensor

To relate sampled voltage and raw ADC values, a linear regression was used and the results of this regression are displayed in (4.1).

$$V = \frac{sample - 139}{425}[V] \tag{4.1}$$
$$r^2 = 0.99$$

The code used to process the voltage sensor data can be found in Appendix C.1.

4.4 Current Sensor

To measure the current leaving the battery pack, the LEM CKSR50NP was selected; Figure 4.4 displays this sensor. This sensor was selected for low noise margins, high response time, simple interface, and common usage within industry. Figure 4.2 highlights this section.

Figure 4.4: The LEM CKSR50NP current transducer (sensor), unmounted.

Similar to the voltage sensor, the current sensor required calibration. To calibrate the current sensor, the sensor was connected in series to a DC current source and an electronic load. The sensor was characterized from 0.00A, to 30.00A in 0.25A steps. At each step, 250 samples were saved using the DAQ described in section 4.2. The results of this test can be found in Figure 4.5.

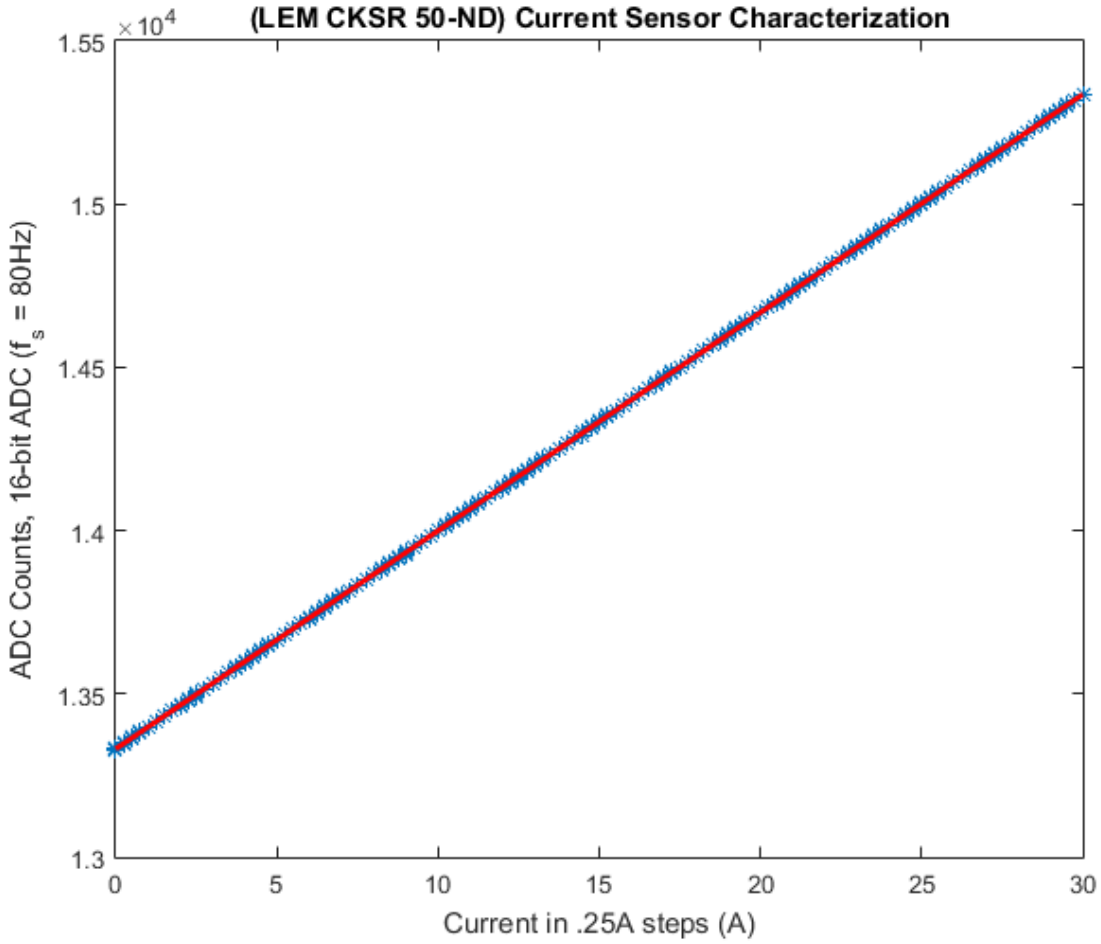


Figure 4.5: Calibration and linear regression fit for the LEM CKSR50NP current transducer

To relate sampled current and raw ADC values, a linear regression was used and the results of this regression are displayed in (4.2).

$$I = \frac{sample - 13318}{69}[A] \tag{4.2}$$
$$r^2 = 0.99$$

The code used to process the current sensor data can be found in Appendix C.2.

Figure 4.6 depicts an image of the completed circuit containing all of the hardware described in Figure 4.2.

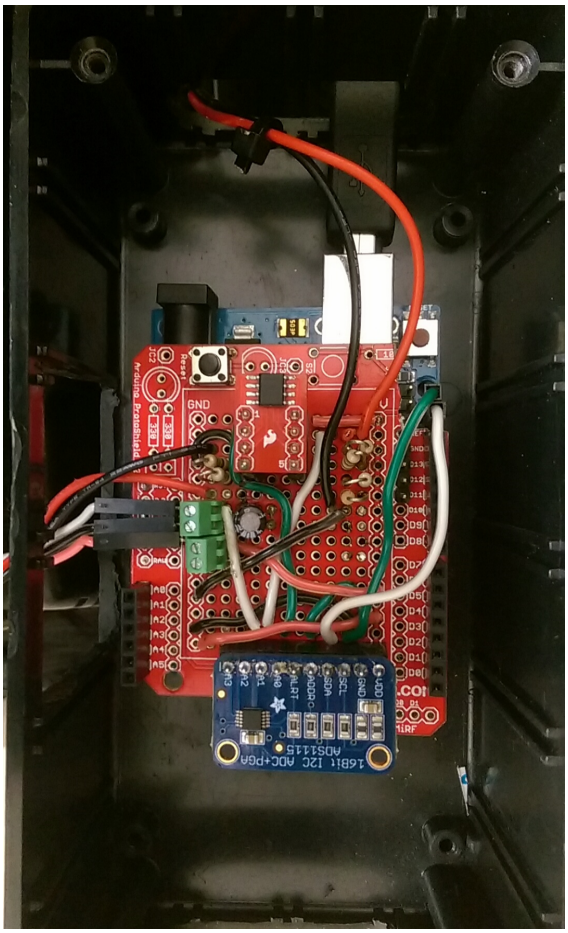


Figure 4.6: Constructed DAQ system. Not pictured: current sensor

Chapter 5

Battery Models

5.1 Overview

A battery′s SOC cannot be physically observed in practical applications. Therefore, to observe SOC, a battery model is necessary. This chapter will review the implemented battery model and the derivations necessary to transform the model into state-space form.

5.2 Battery Model

An equivalent circuit model, based on those seen in [24, 27], was used to model the battery pack. For this model, an ideal power source is series-connected to a resistor, a parallel-connected resistor-capacitor network, and the load. The ideal power source′s voltage is assumed to be algebraically dependent on the battery′s SOC. Figure 5.1 presents a schematic drawing of this model.

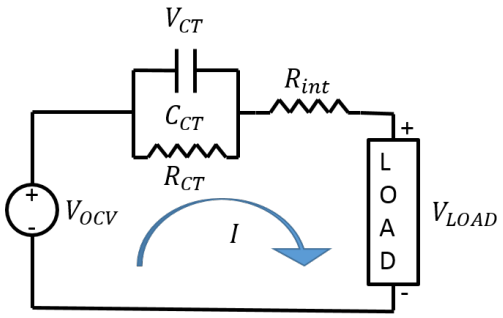


Figure 5.1: The second order equivalent circuit battery model

The ideal power source models the drop in voltage associated with the loss of charge on the

battery. The single resistor models the internal battery resistance due to the wire, electrochemical resistance, and hardware connecting the cells within battery pack. The RC pair models the battery pack′s dynamics. Finally, the load component models the system drawing current from the pack. The initial use of this model does not consider the battery′s non-linearities; rather, it is initially assumed that the battery′s dependence on SOC is linear, which is shown later to be a good approximation for most of the operating range of the battery. Chapter 7 will detail the methods used to consider the non-linearities not described by this model which become pronounced during the extreme high and low levels of a battery′s SOC curve.

Measuring at the positive and negative terminals of the load is akin to measuring the voltage across the positive and negative terminals of the battery while the pack is installed in the wheelchair system. This model provides a simple method to represent battery behavior; it does not indicate the difficulty measuring behavior of the components within a battery pack, nor the idealization of complex behavior into specific components. As a result, it must be emphasized that only the voltage across the load can be measured and the other components must be estimated.

Using Kirchoff's Voltage law on the model, the voltage, V_{load}, is presented in (5.1).

$$V_{load} = V_{OCV} - R_{int}I(t) - V_{CT} \tag{5.1}$$

Where $I(t)$ represents the current flowing from the battery pack to the wheelchair system and V_{CT} represents the voltage across the RC network.

V_{OCV} is a function of the battery's SOC and (5.2) presents this function.

$$V_{OCV}(SOC) = SOC\alpha + \mu \tag{5.2}$$

Where α relates SOC to voltage and μ represents the cutoff voltage of the battery pack. Since only current and voltage can be directly measured, a function relating SOC to current is presented in (5.3).

$$SOC = 1 - \frac{1}{Q_0}\int_{t_0}^{t} I(\tau)d\tau \tag{5.3}$$

5.3 State-Space Representation of the Battery Model

To practically implement the model described by (5.1), (5.2), and (5.3), these functions must be transformed into State-Space representation, as described in standard form shown in (5.4-5).

$$\dot{x} = Ax + Bu \tag{5.4}$$

$$y = Cx + Du \tag{5.5}$$

To transform the functions into the form presented in (5.4-5), the states, or energy storage components for this model, must be identified. For this simplified model, the derivative of the energy states in this system are: $S\dot{O}C$ and $\dot{V}_{CT}$. The first state represents SOC, or remaining energy, of the battery. The second state, V_{CT}, represents the voltage across the RC pair, or the energy stored in the capacitor.

Equation (5.3) represents the SOC state. The state equation, (5.6), is found by differentiating (5.3).

$$S\dot{O}C = -\frac{1}{Q_0}I(t) \tag{5.6}$$

It must be noted that this state equation only depends on the input to the system and does not depend on any other states. In the system dynamic model, this represents a pure-integrator dynamic.

To derive the $\dot{V}_{CT}$ state, one must look at the parallel RC pair in Figure 5.1. The voltage across the resistor and the capacitor remains the same, but the current does not. Therefore it is useful to follow Kirchoff′s Current Law about the parallel pair (5.7).

$$I(t) = I_{R_{CT}}(t) + I_{C_{CT}}(t) \tag{5.7}$$

Next, substituting the definitions for current through a resistor and a capacitor into (5.7), one can obtain equation (5.8).

$$I(t) = \frac{V_{CT}}{R_{CT}} + \dot{V_{CT}}C_{CT} \tag{5.8}$$

Through algebraic manipulation, the state equation is derived from (5.8) and presented in (5.9).

$$\dot{V_{CT}} = -\frac{V_{CT}}{\tau_{CT}} + \frac{I(t)}{C_{CT}} \tag{5.9}$$

Where τ_{CT} represents the product of C_{CT} and R_{CT}.

The output equation of the system was presented in (5.1), and substituting values into (5.1) from (5.2) and, yields equation (5.10).

$$V_{load} = \alpha SOC + \mu - R_{int} I(t) - V_{CT} \tag{5.10}$$

The state vector, x, is displayed in (5.11), the input function, $u(t)$, is presented in (5.12), and the output variable, y, is defined in (5.13).

$$x = \begin{pmatrix} V_{CT} \\ SOC \end{pmatrix} \tag{5.11}$$

$$u(t) = \begin{pmatrix} I(t) \\ \mu \end{pmatrix} \tag{5.12}$$

$$y(t) = V_{load} \tag{5.13}$$

Now, that the state equations and output equations have been derived, they can be transformed into state-space form, as presented in (5.14-15).

$$\dot{x} = \begin{pmatrix} -\frac{1}{\tau_{CT}} & 0 \\ 0 & 0 \end{pmatrix} x + \begin{pmatrix} \frac{1}{C_{CT}} & 0 \\ -\frac{1}{Q_0} & 0 \end{pmatrix} u(t) \tag{5.14}$$

$$y(t) = \begin{pmatrix} -1 & \alpha \end{pmatrix} x + \begin{pmatrix} -R_{int} & 1 \end{pmatrix} u(t) \tag{5.15}$$

Before implementing the model shown in (5.14-15) within a Kalman Filter, the equations are first discretized. Since sampled data serve as the inputs to this model, a discrete-time model was derived using a Zero Order Hold (ZoH); equation (5.16-19) presents the results of this method [43]

$$A_d = e^{AT} \tag{5.16}$$

$$B_d = (\int_{\tau=0}^{T} e^{A\tau} d\tau) B \tag{5.17}$$

$$C_d = C \tag{5.18}$$

$$D_d = D \tag{5.19}$$

where T represents the sampling period of the DAQ.

Chapter 6

Battery Parametrization

6.1 Overview

In order for the model derived in Chapter 5 to be usable, the parameters within the model needed to be identified. This process can be described in two stages: capturing the open-circuit voltage State-of-Charge curve and determining dynamic model coefficients via the current pulse test. This chapter aims to describe the aforementioned processes and how they were used to identify the battery models parameters.

6.2 The Open-Circuit Voltage-State of Charge Curve

Equation (5.14-15) presents a relationship between the battery′s SOC and its open-circuit voltage. To find this relationship, the battery pack was first fully charged. Next it was slowly discharged at a constant rate until the cutoff voltage has been met. The cutoff voltage is defined by the absolute lowest voltage a battery pack may reach before damage to the pack or significant losses in capacity are incurred. Typically, the cutoff voltage is provided by the battery′s manufacturer in the datasheet.

To perform a discharge test, the battery pack was connected to an electronic load. A relay circuit was series-connected between the battery and the load to ensure that discharging of the battery would cease once the voltage fell below the cutoff voltage. Finally, the DAQ described in Chapter 4 was connected to the pack for data acquisition. Figure 6.1 presents a diagram describing the hardware configuration for this test.

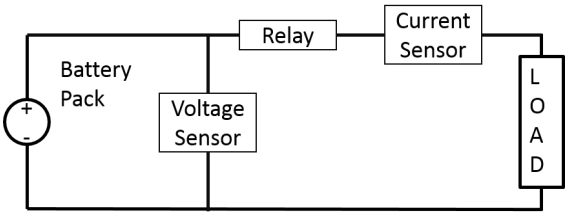


Figure 6.1: Rig used to characterize the battery′s OCV-SOC curve.

The DAQ sampled the pack′s voltage at 80Hz until the completion of the test. Finally, all interconnects between devices were tightly bound and coated in a dielectric grease to ensure the best possible connection.

The discharge testing requires caution: a discharge too slow will take too long for the test, whereas a discharge that is too fast can cause excitation of battery dynamics or even damage to the battery. To determine an appropriate discharge rate, the battery′s predicted capacity can be used; for the wheelchair battery, this is approximately 28 Ah [44]. The battery must be discharged a rate of 0.1 times the capacity or less to properly sample the voltages needed to generate the pack′s OCV-SOC curve [24]. A discharge rate of 1A was selected to discharge the battery pack, as this rate balanced an acceptable discharge time versus fidelity in measuring the OCV battery behavior. To capture the data needed to form the OCV versus SOC curve, the battery pack was fully charged and allowed to rest. Then, the pack was connected to the system described in Figure 6.1 and the discharge began. The discharge ended when the cutoff voltage was met and the relay opened the circuit. Finally, the data captured was saved and stored for processing. This test was performed three times and Figures 6.2, 6.3, and 6.4 present the results of the three discharge tests.

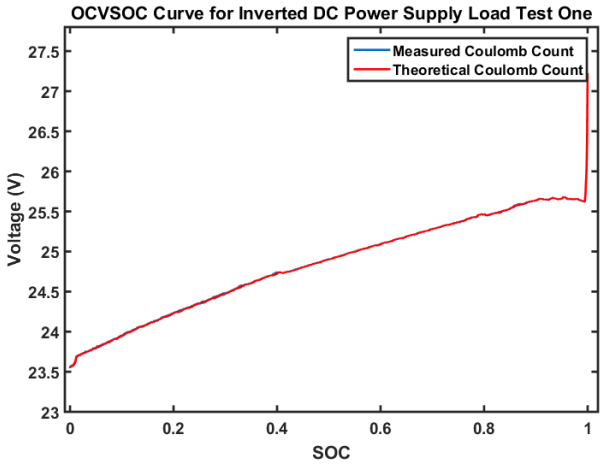


Figure 6.2: First discharge OCV-SOC curve.

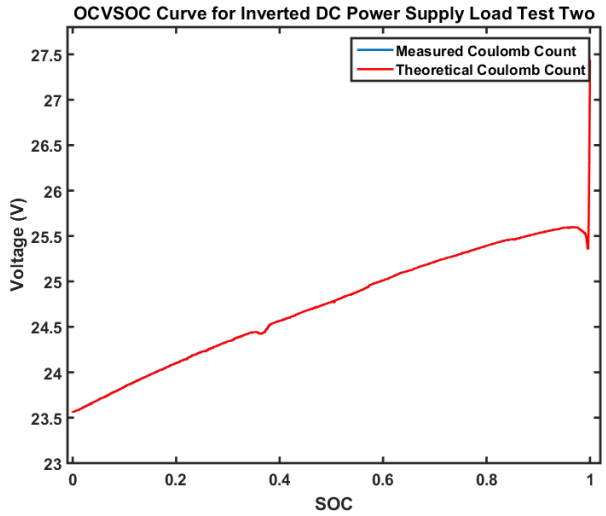


Figure 6.3: Second discharge OCV-SOC curve.

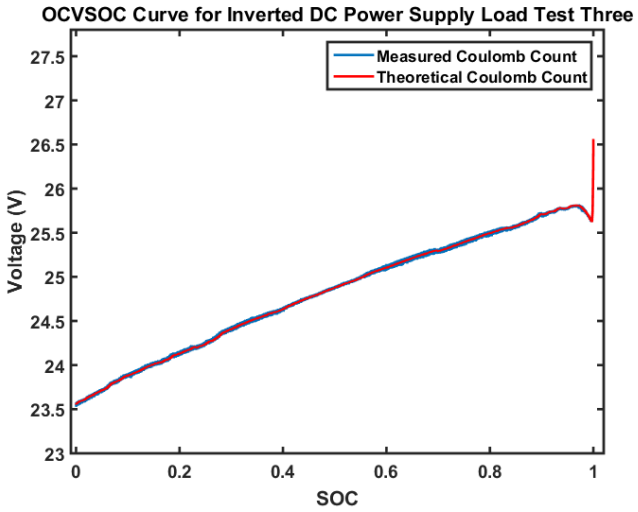


Figure 6.4: Third discharge OCV-SOC curve.

As mentioned, only the linear region of the battery′s OSC-SOC curve is used in later models for estimation. As a result, the non-linear regions of the battery were ignored when determining the parameters discussed in equations (5.14-15). By inspection, the linear region of the OCV-SOC curve was defined as the region between 0.1 and 0.9 SOC. After truncating the data for the three data sets between these SOC ranges, a linear regression was performed to find α and μ. Figure 6.5 presents the linear region of the three curves and the resulting regression line.

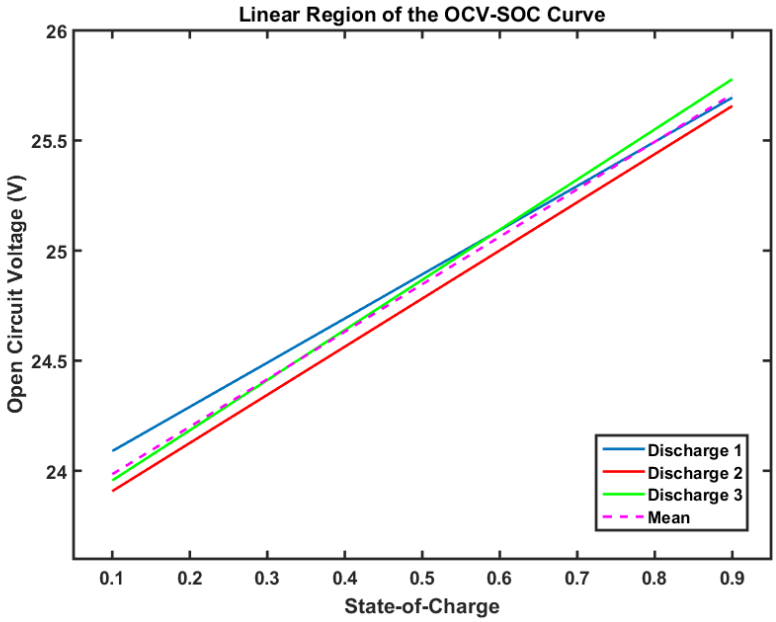


Figure 6.5: Presented are the linear regions of the three OCV-SOC discharge tests and the average of the three curves.

Since the battery was discharged from fully-charged to complete-discharge, the battery′s capacity was also determined. This capacity value, Q_0 considers the entirety of the OCV-SOC curve and is necessary for SOC estimation in both the linear and non-linear regions. Table 6.1 presents the regression′s parameters and the battery capacity.

Parameter	**Value**
α	2.1569 V/SOC
μ	23.7689 V
r^2	0.9968
Q_0	1.1070E+05 Coulombs

Table 6.1: Battery model parameters derived from the linear region of the OCV-SOC Curve

Appendix D.1 presents the MATLAB code used to process the OCV data, perform the regression, and generate the plots shown in Figures 6.2 – 6.4.

6.3 The Current Pulse Test and Least Squares Regression

The parameters in section 6.2 describe the battery′s transient response while operating in the linear region of the OCV-SOC curve. For low, constant-current draw systems, the regression parameters presented in Table 6.1 would be the only terms necessary for SOC estimation. However, the wheelchair system does not draw power via low and constant currents; the wheelchair′s current draw will vary significantly with wheelchair velocity, surface incline, and surface type among other things.

Because the battery voltage will react to any sudden changes in current, it is important to characterize the battery′s response to sudden changes in current draw, i.e. the battery dynamics. To characterize the battery dynamics, a current pulse test is used in this study. A pulse test holds the battery at rest for an extended period, then suddenly changes the current draw of the battery from no current draw to a current draw equal to a fraction of the pack′s typical load. This pulse duration lasts for a set fraction of the battery′s SOC; this allows for the capture of impulse dynamics. The battery is then allowed to rest for a period of time to characterize settling dynamics. Some pulse tests suggest using two to three closely-spaced, equal-magnitude pulses of large current to capture the dynamics [27] whereas others suggest a series of small pulses of similar magnitude spread across the battery′s entire SOC [26, 29]. To characterize the battery pack in this study, a combination of both methods was used.

The pulse test as implemented in this thesis starts by first shifting the battery into its linear region by discharging the pack until its SOC is equal to 80%. The pack is allowed to rest for an extended period after being discharged into the linear region. Then the pack is discharged using five pulses: two pulses of 10A and three pulses of 5A spread out across the battery pack′s SOC. These values were selected because they closely model the rest current draw and average driving current draw of the wheelchair. Figure 6.6 presents a diagram describing this profile.

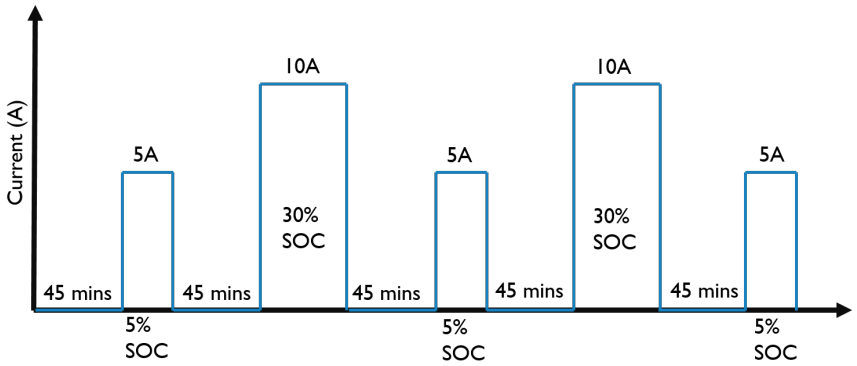


Figure 6.6: The discharge profile used to resolve remaining the battery parameters, specifically, the pack′s dynamics.

An HP 6050A 1800 Watt electronic load was used to control the current draw from the battery pack using the profile described in Figure 6.6. The DAQ discussed in CH 4 sampled and stored the data captured during these tests. Figure 6.7 depicts the hardware used to realize the aforementioned current profile. Figure 6.8 presents the results of this test.

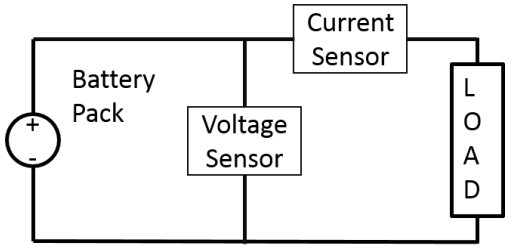


Figure 6.7: A diagram depicting the hardware used to run the pulse test.

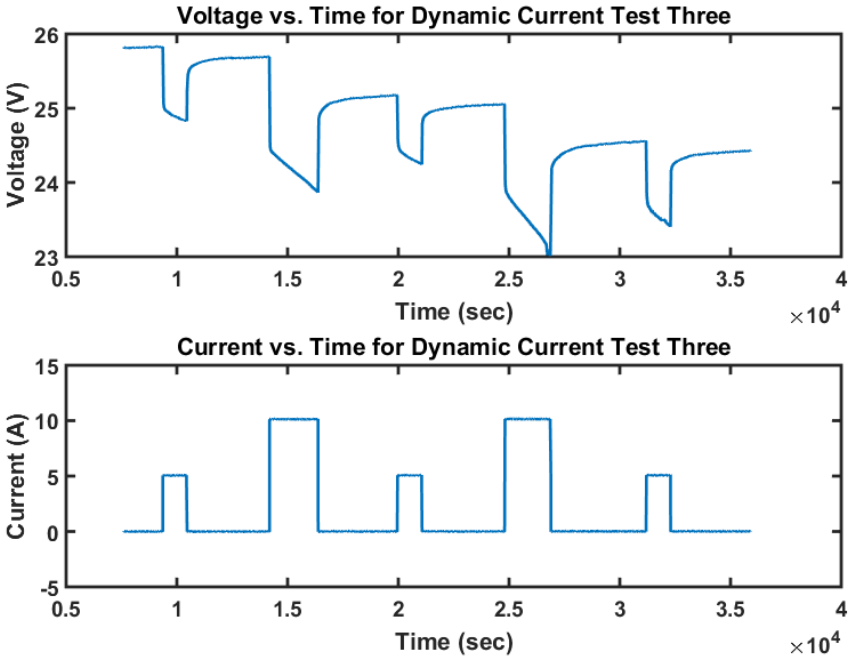


Figure 6.8: The measured voltage (top) and current (bottom) from the current pulse test.

Using a linear least squares regression and the data presented in Figure 6.8, the remaining free parameters of the model described in Chapter 5 were fitted. Figure 6.9 presents the results of this model fit and Table 6.2 defines these parameters.

Parameter	**Value**
τ	305.77 s
R_{int}	0.108 Ω
C_{CT}	11994 F

Table 6.2: Battery model parameters derived from the least squares regression performed on the current pulse data

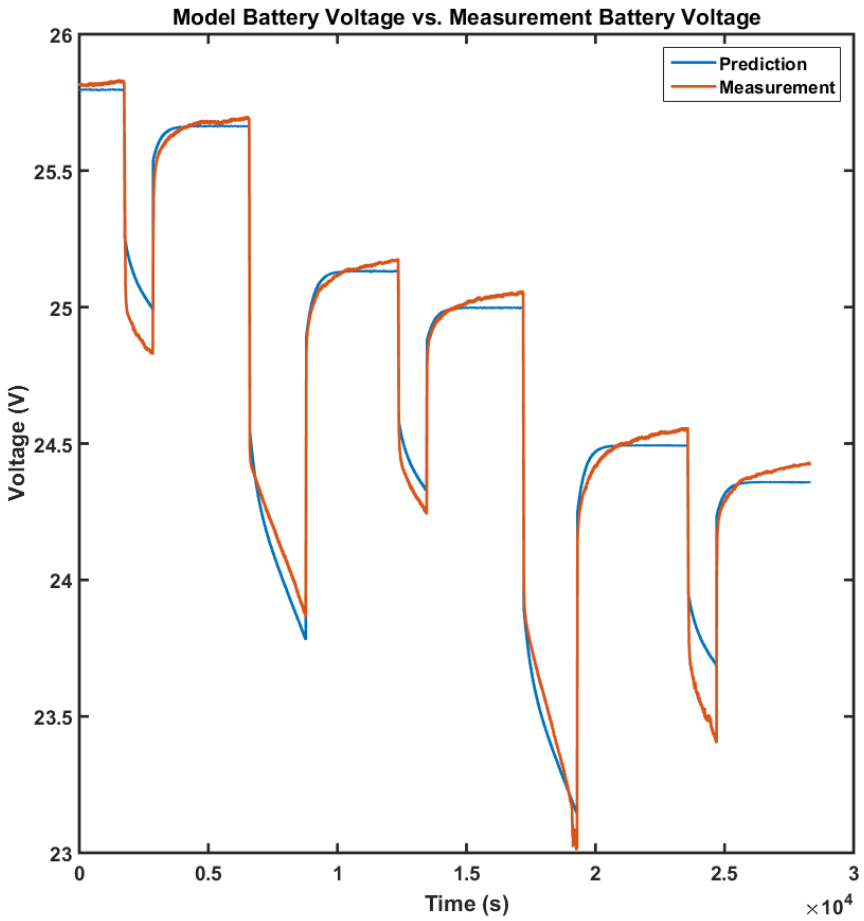


Figure 6.9: The least squares regression fit of the dynamic terms of the battery model.

Appendix D.2 presents the MATLAB code used to process the current-pulse data, perform the regression, and generate the plots shown in Figures 6.8 and 6.9.

Chapter 7

State of Charge Estimation

7.1 Overview

This chapter presents the methods used to estimate the wheelchair battery pack′s SOC when the wheelchair is operating. The estimators are treated differently depending on whether the SOC is in either the linear and non-linear regions of the battery pack′s OCV-SOC curve; this idea of treating the OCV-SOC curve as a piecewise function for estimation was first presented by Codeca *et al.* in [35]. First, an analog for the model will be presented. Then a discussion for SOC estimation in both the linear and non-linear regions of the curve will be discussed.

7.2 The Fuel-Gauge Model

The method presented by [35] can be described using an automotive fuel-gauge as an analog. A vehicle′s fuel gauge remains at full for a long time after the tank has been filled with gasoline. This occurs to compensate for the non-linearities that occur when measuring fluid volume in a full container, as discussed in [45]. After a given threshold, the fuel measurements begin to decrease with a direct relation to remaining fuel volume. Finally, given a second threshold, the fuel gauge will present empty even as some fuel remains in the tank. This occurs to account for consumers who prefer to drive their vehicle with little fuel remaining and to account for the non-linearities associated with measuring fluid volume as the tank is nearly empty.

The fuel-gauge model of a battery SOC model uses a similar approach; a Coulomb counter estimates the SOC of the battery pack in the first non-linear region (100% to 90% SOC) of the battery pack, using only current measurements. Figure 7.1 depicts a diagram presenting this method.

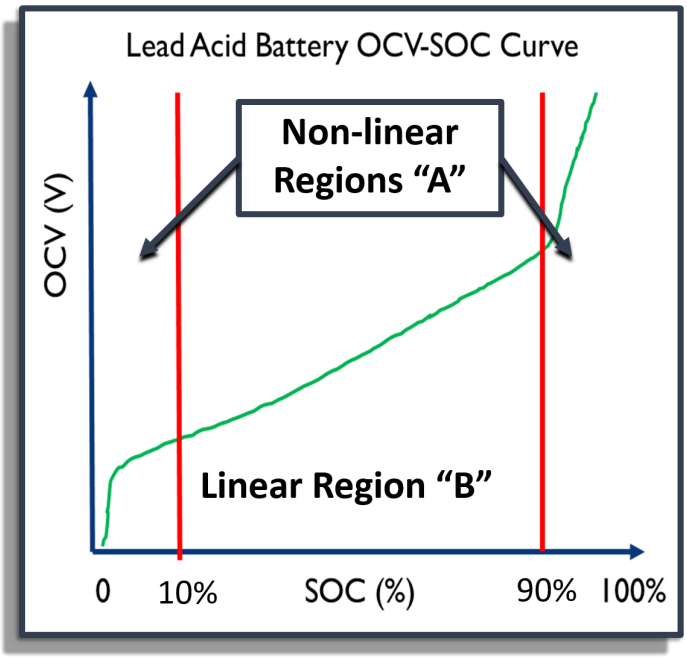


Figure 7.1: The fuel gauge model: A) represents the non-linear regions in which Coulomb counting will be implemented for SOC estimation and B) represents the linear regions where a Kalman filter will be realized for SOC estimation

In the linear region of the pack (90% to 10% SOC), a Kalman Filter predicts the SOC state from the model presented in Chapter 4, which utilizes both voltage and current measurements. For this stage of the estimator, the initial guess for the SOC state originates from the aforementioned Coulomb counter. Finally, when the SOC state falls below the 10% threshold and enters the final non-linear region (10% to 0% SOC), SOC is again estimated using a Coulomb counter, using only current. The goal of this gas-gauge model is to effectively avoid using voltage measurements in the nonlinear region of the OCV-SOC curve, when the voltage to capacity relationship is difficult to measure or utilize.

7.3 The Coulomb Counter

In the nonlinear regions of the OCV-SOC curve (SOC $>$ 90% SOC or SOC $<$ 10%), Coulomb counting is used in the fuel-gauge model to estimate the SOC on the battery. Coulomb counting does not depend on the OCV-SOC curve; it only depends on current draw, battery capacity, and initial state to predict SOC. This method, while very simple, is very susceptible to drift as it does not compensate prediction changes due to battery dynamics. The equation shown in (7.1) presents the Coulomb counter used in this estimator.

$$SOC = 1 - \frac{1}{Q_0}\int_0^t I(\tau)d\tau \tag{7.1}$$

In the first non-linear region, the Coulomb counter′s purpose is to provide the user with a rough estimate of the SOC of the wheelchair and to provide the initial guess of the SOC state to the Kalman filter. In this non-linear region, the user can be told they have full charge until the linear region is entered, just as is done in a typical vehicle fuel gauge.

In the third and final region of the SOC curve, the Coulomb counter will estimate the remaining charge on the battery. With time, the battery will age and, as a result, its parameters will change thus yielding a lower overall SOC. In this final SOC region, the user may be told their battery pack is empty, similar to the fuel gauge, and the SOC estimate can be used to help the user predict when they should return to a wall charger. Driving in this region of the OCV-SOC curve would be similar to driving a car when the fuel gauge reads empty.

7.4 The Kalman Filter

In the linear region of the OCV-SOC curve (90% - 10% SOC), this thesis utilized a linear Kalman Filter as described by [33]. Equations (7.2-7.4) present the prediction and equations (7.5-7.9) present the measurement update.

$$\hat{x}_{k|k-1} = A\hat{x}_{k-1|k-1} + Bu_{k-1} \tag{7.2}$$

$$P_{k|k-1} = AP_{k-1|k-1}A^T + Q \tag{7.3}$$

$$Q = diag(\sigma^2_{V_{CT}}, \sigma^2_{SOC}) \tag{7.4}$$

$$K_k = P_{k|k}C^T(CP_{k|k-1}C^T + R)^{-1} \tag{7.5}$$

$$\tilde{y_k} = y_k - (C\hat{x}_{k|k-1} + Du(k)) \tag{7.6}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k\tilde{y_k} \tag{7.7}$$

$$P_{k|k} = (I - K_kC)P_{k|k-1} \tag{7.8}$$

$$R = \sigma^2_{sens} \tag{7.9}$$

The performance of the Kalman filter described in (7.2-7.9) depends on the Q and R matrices and assumes the noise found within the system is Gaussian. The Q matrix, or the process noise,

contains the standard deviations associated with each states′ noise. A state′s noise may come from, among other things, model mismatch or unforseen system behavior. This matrix is a square matrix of size n x n where n is the number of states. The R matrix is the measurement noise, or the noise associated with each sensor. This matrix is assumed to be diagonal, contains the standard deviations associated with each sensor′s noise, and is of size m x m where m is the number of inputs.

To develop the R matrix, the noise characteristics of the voltage sensor – the single measurement in this system – needed to be determined. Using the DAQ described in, the voltage sensor measured the potential of a constant DC supply for a minute after selecting the constant potential. The RMS error between the true voltage of the supply and the measured potential allowed for the calculation of the single term in the R matrix. This value was further refined experimentally while testing the Kalman Filter. Table 7.1 presents the R matrix.

The R matrix further describes the trust in the measurement versus the trust in the model.

Parameter	**Value**
σ^2_{sens}	5.2365

Table 7.1: R matrix values; this value represents the measurement noise

Since the noise for the states in this system is uncolored, the Q matrix is diagonal and each term along the diagonal describes the noise associated with that state. The noise associated with the SOC state was derived by calculating the mean RMS error between the derived OCV-SOC curve presented in Section 6.2 (Table 6.1) and the linear regions of each OCV-SOC curve presented in Figures 6.2, 6.3, and 6.4. To calculate the error associated with the V_{CT} state, the square of the difference between the model and the measurement′s best and worst 'dynamic' (e.g. the response during a rest period just after a sudden impulse or change) was used. Specifically, these data were found in Figure 6.9. Similarly, the aforementioned values were further refined experimentally while testing the Kalman Filter.

The Q matrix diagonals present not only the noise of each respective state, but also the trust in the model versus trust in the measurement. The lower the value, the more trust in the model and the less trust in the measurement. Table 7.2 presents the values along the diagonal of the Q matrix.

Parameter	**Value**
σ^2_{SOC}	0.0011
$\sigma^2_{V_{CT}}$	3.356

Table 7.2: Q matrix values; These values represent the noise associated with the model′s states

Chapter 8

Implementation of Estimator

8.1 Overview

In order to verify the functionality of the estimator design, it was realized on the wheelchair platform and tested. The purpose of this chapter is to present the nuances observed when realizing the algorithm and to discuss the tests designed to verify the algorithms functionality.

8.2 Implementation of Estimator

The estimator defined in chapter 7 exists as two functions: the coulomb counter and the Kalman filter. This algorithm was realized using, a series of case statements checking the SOC state to switch between the coulomb counter and the Kalman filter.

The estimator was simulated and realized in MATLAB. Voltage and current data was captured from the wheelchair using a ROS bag file. This data was subsequently parsed into an *.csv file for MATLAB processing.

Appendix E presents the Kalman filters implementation in MATLAB.

8.3 Tests Designed

After fully charging the wheelchair′s battery pack, the wheelchair′s computing hardware was powered using the external power source in order to have a realistic power load for the wheelchair′s SOC estimator. Then, once the wheelchair was fully powered on, the estimator was started and the external power cut. To discharge the batteries, the wheelchair was driven around an office

environment and a hallway environment. Figure 8.1 depicts a sample hallway environment.

Figure 8.1: This image depicts the hallway outside of 320 Reber building on The Pennsylvania State University Main Campus. Hallways such as these were used to verify the estimator′s functionality

These environments consist of smooth, level surfaces commonly found in office buildings. To best model the behavior of a wheelchair user in an office environment, the wheelchair drove up and down the hallway environment and drove into the office environment to rest. During the rest period, the only current draw is from the computing system. The wheelchair traversals in the hallway were not all uniform, as a human user was moving the wheelchair in a hallway. This illustrates that power draw may vary for a wheelchair depending on average velocity, the presence of nearby people, operator distractions, or sudden obstacles. The rest periods in the office models a user working at their desk for extended periods.

Chapter 9

Estimator Results

9.1 Overview

This chapter presents and discusses the results of the estimation algorithm. Further, this chapter analyzes the effectiveness of the estimator.

9.2 Discussion of Results

After simulating part of an average wheelchair user's day, as discussed in Chapter 8, the voltage and current information was processed. The resulting SOC estimate is presented in Figure 9.1. The result of the fuel gauge estimator is presented in (blue) and, for comparison, an SOC estimate using Coulomb counting (red) alone is presented. First, it must be noted that the SOC estimate does not extend from full charge to full discharge. This occurred because, when the battery was placed under driving load, the battery voltage would fall below the pack's cutoff voltage. Since the estimator was not realized on the wheelchair and run externally the test ended when the majority of pack voltage readings were below cutoff while driving to prevent damage to the battery pack.

Along the same vein, one will notice variance in slope of the SOC curve. The regions with the largest slope, namely those around 100 minutes, 250 minutes, and just before 300 minutes, are instances where the wheelchair was driven. During other time frames, the wheelchair was at rest or being driven at low speeds.

The estimate from the Coulomb counter and the estimator are very similar; of greatest note, both SOC estimates end at nearly the same value. The current sensor used presents very low noise margins and, as a result, little drift. Coulomb counting can, with high fidelity sensors, present a

reasonable SOC estimate. However, Coulomb counting will not compensate for model mismatch due to battery aging whereas a Kalman filter is able to compensate for both battery age and lower fidelity sensors.

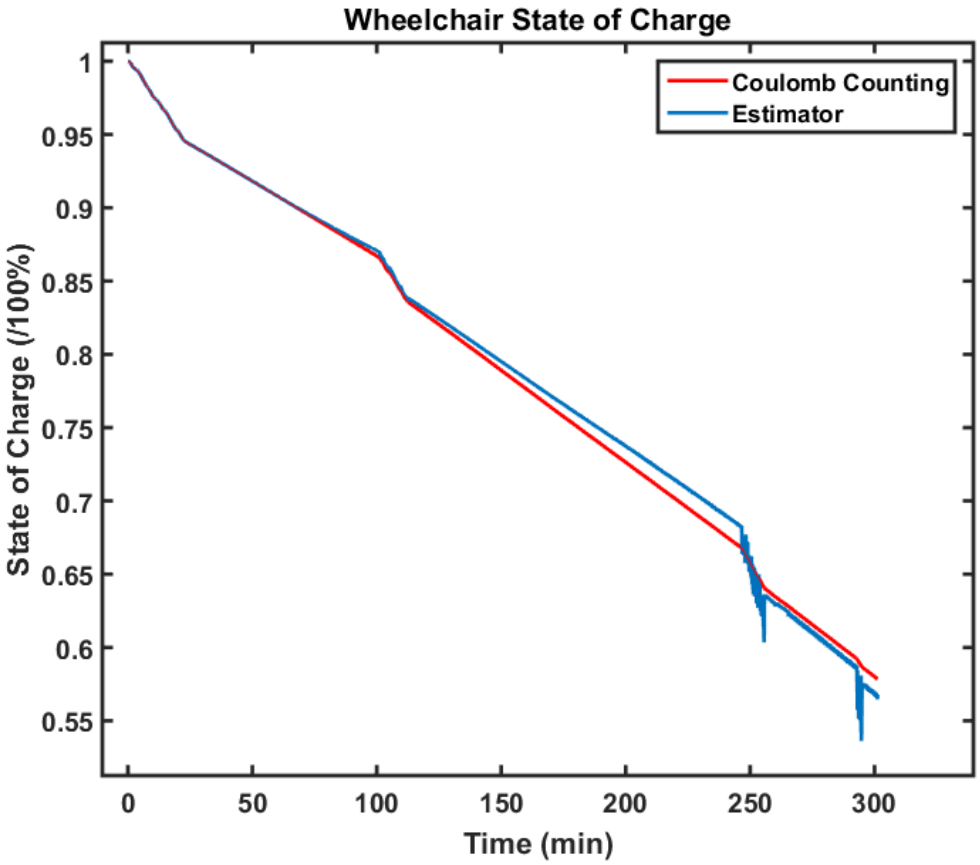


Figure 9.1: This figure presents the SOC estimate when using Coulomb Counting (red) and the fuel gauge model (blue).

Finally, reviewing the estimator's SOC it is observed that the Coulomb counter's SOC curve is smoother; the estimator's SOC curve contains spikes near the end of the discharge. These spikes are a result of the Kalman filter's inability to completely correct for unmodeled battery dynamics. If a higher-order, non-linear model were used to estimate battery SOC and an Extended Kalman Filter (EKF) were implemented instead of a linear Kalman Filter, these noise artifacts would be significantly mitigated if not entirely eliminated.

The estimator is able to track the battery pack voltage with great accuracy, as demonstrated in Figure 9.2. Clearly evident, the voltage measurement and estimate overlap significantly. However, the tracking is not perfect. The imperfections arise from the Kalman Filter's inability to predict

exact state values; the Kalman Filter is only able to estimate state values and, as a result, will be unable to fully track any given system.

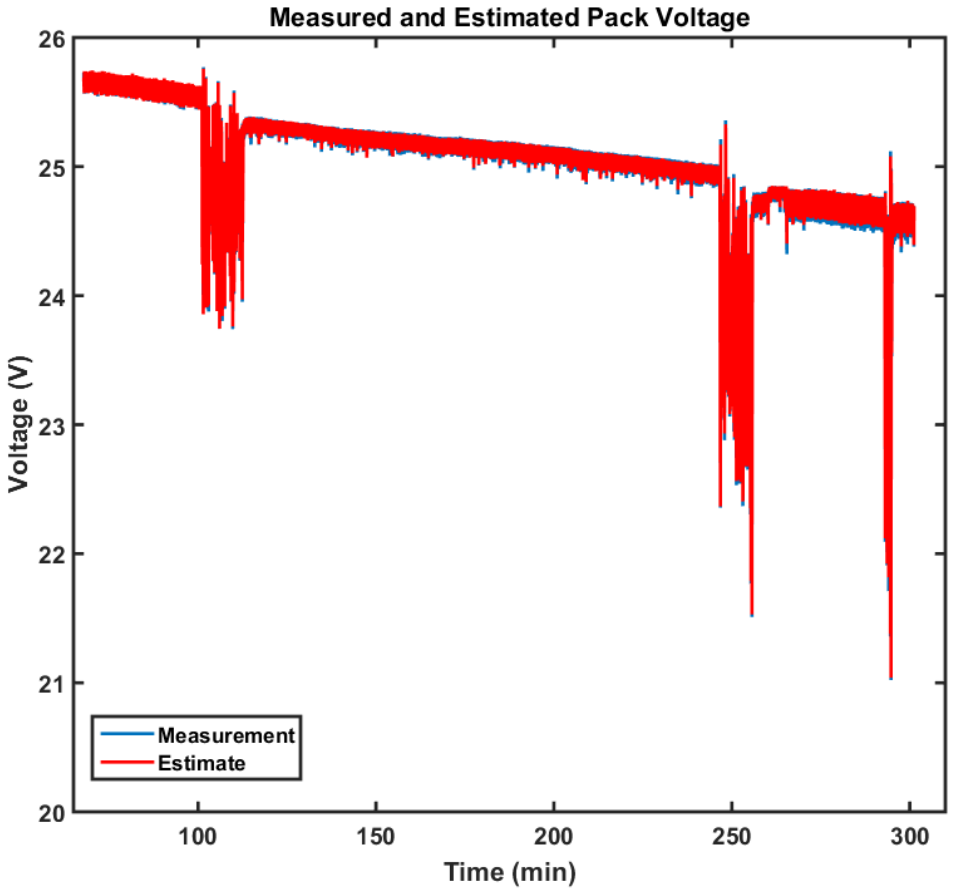


Figure 9.2: This figure presents the battery pack voltage estimate (red) and the voltage measurement (blue).

Finally, Figure 9.3 presents both the SOC and V_{CT} states on top of one another for direct comparison. Reviewing the behavior of the V_{CT} state and considering typical capacitor behaviors, the overall behavior of the V_{CT} state is as predicted by circuit theory. At first, the V_{CT} state's voltage is, indeed, negative; this voltage is corrected with time as the Kalman filter continues to estimate; this correction confirms Kalman filters' estimates improve over time. Finally, it must be noted that estimating V_{CT} is very difficult as the voltage across this parallel RC pair has little physical meaning on the battery pack. There is no RC pair in the battery pack that can be probed and used for comparison.

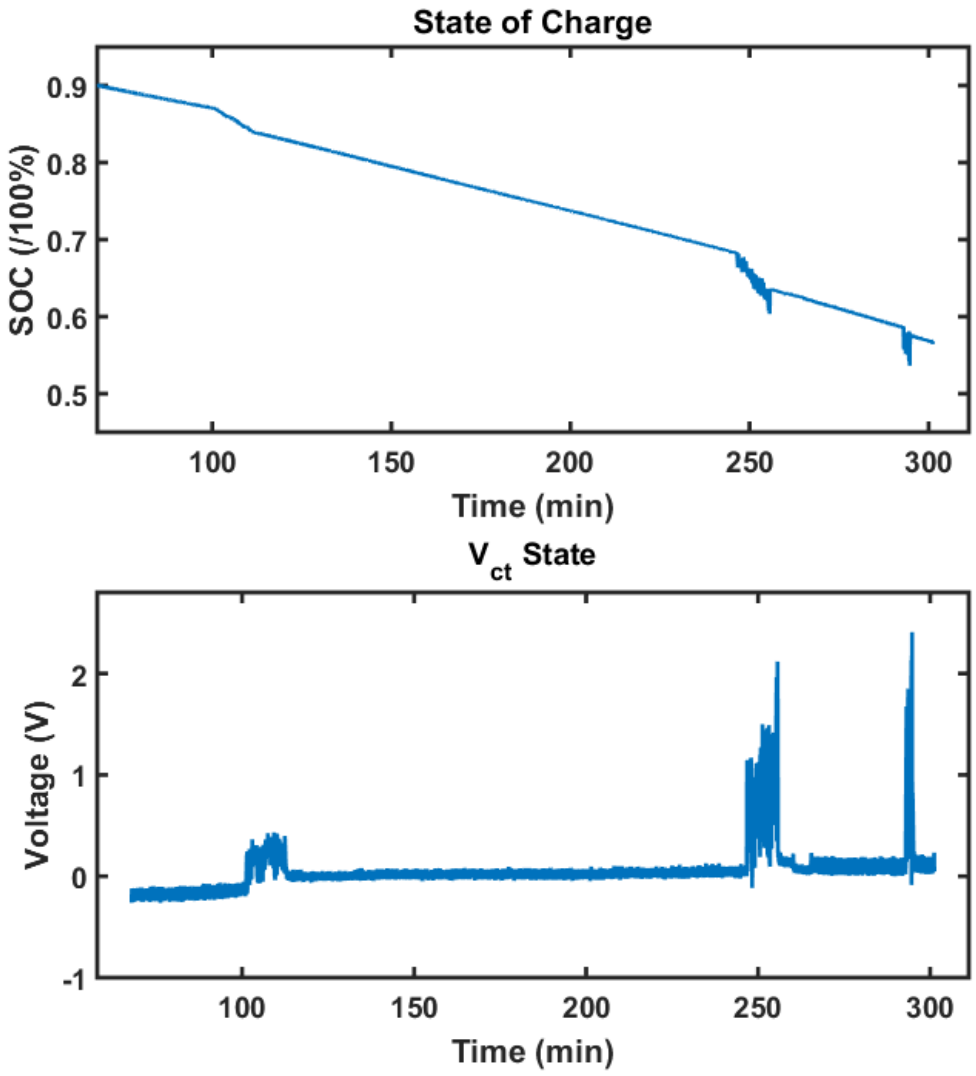


Figure 9.3: This figure presents the SOC (upper) and V_{CT} (lower) state estimates with respect to time.

The results presented indicate that the estimator designed provides a usable estimate of battery pack SOC. Furthermore, these results indicate that, if one has a very high fidelity current sensor, this sort of estimator may not be necessary. However, outside of the research environment where such sensors may not be cost effective, the estimator derived in this research will be able to provide the an SOC estimate with accuracy and precision comparable to a high-fidelity sensor. Furthermore, this estimation will be able to account for drift as a result of battery model mismatch, battery aging, and non-regular current demands.

Chapter 10

Conclusions

10.1 Overview

This project successfully implemented a model -based SOC estimator on an electric wheelchair. The estimator was realized in a manner similar to a cars fuel gauge, where the fuel estimates are purposefully less accurate at the extrema of the fuel levels (e.g. full and empty). This chapter will present conclusions and possible future expansions.

10.2 Conclusions

This research sought to fill a gap in assistive technology literature in the area of energy prediction for electric wheelchairs. This research may be beneficial to both electric wheelchair users and researchers alike. Electric wheelchair users will be able to better estimate their electric wheelchairs range and make informed decisions about when to recharge their batteries. Researchers may use this work to develop better SOC estimators for electric wheelchairs or use this work to develop energy-cognizant smart wheelchairs.

One advancement of this work would be to realize an Extended Kalman Filter (EKF) to estimate the SOC on the battery pack across the entire range of the SOC values. The EKF linearizes a system on every iteration, thus accounting for non-linearities. This approach was not used on this wheelchair as the aim of this work was to develop an estimator that could be realized on most electric wheelchairs; most electric wheelchairs have limited computational power and the aforementioned model could be realized on a simple microcontroller, similar to those found on electric wheelchairs today.

10.3 Future Work

There still exists a large amount of possible work in the area of energy prediction for electric and smart wheelchairs. First, algorithms to predict remaining electric range may be investigated to help users predict remaining distance as opposed to remaining charge. Another possible expansion of this research involves helping a user not only predict electric range, but also predict if they could return to their starting point given their current location. This process is known as retro-traversal.

Researchers and smart wheelchair developers may use this research to develop navigation algorithms that account for energy efficiency, traveling distance, and traveling time. Finally, institutions may build upon this research by using this system to measure the power needed to travel and retro-traverse common paths and generate maps displaying powered wheelchair energy requirements. Such maps may be useful to users so that they may make informed decisions about the path they would like to use to move between two locations, such that energy-aware choices can be made.

Appendix A

Appendix A

The following is a collection of images of the wheelchair system. They are in no particular order.

Figure A.1: Side view of the wheelchair

Figure A.2: Semi-oblique view of the wheelchair

Figure A.3: Front-on view of the wheelchair

Figure A.4: Seat of the wheelchair

Figure A.5: Wheelchair joystick and mount

Figure A.6: Emergency stop, LCD indicator, and secondary joystick (for show only).

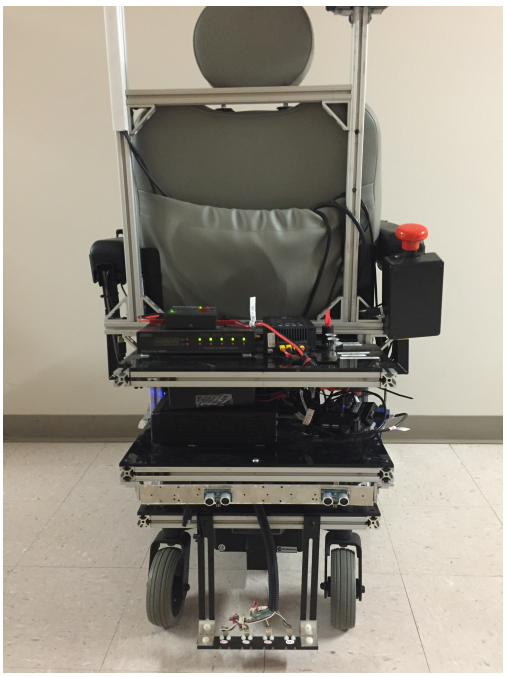


Figure A.7: Back view of the wheelchair

Figure A.8: Top-down view of the wheelchair's top shelf

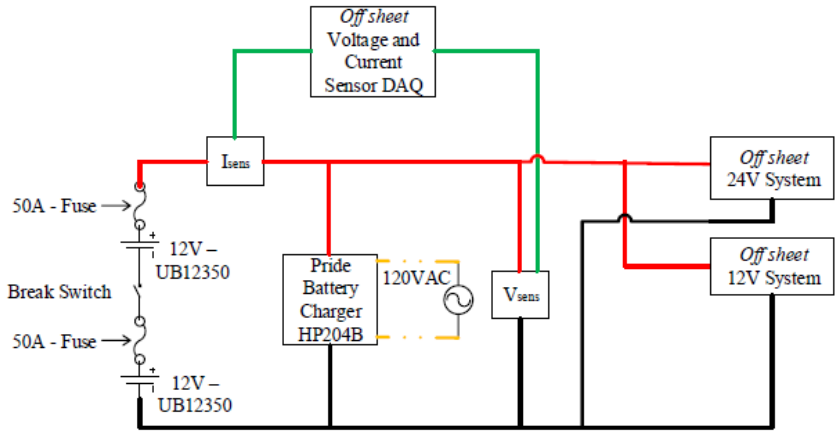
Figure A.9: Side view of the wheelchair's shelf (computer powered on)

Appendix B

Appendix B

The following are the schematics designed for the power and sensor architecture to run the wheelchair. Each schematic has a unique key.

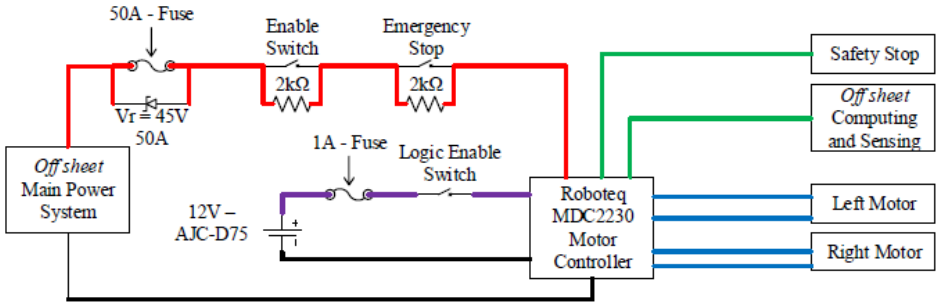
Each schematic contains a unique key to identify the the components and wiring type. The first schematic is the main power system schematic, the second contains information for the drive train power system, the third describes the power system for the computing and sensing system, and the fourth schematic breaks down the sensor architecture.



Main Power System Design Rev 5
Intelligent Vehicles and Systems Group
The Pennsylvania State University
By: Christopher Miller

KEY
+24V
GND
Data
120VAC

Figure B.1: Main power system schematic



Drivetrain (24V) Power System Design Rev 5
Intelligent Vehicles and Systems Group
The Pennsylvania State University
By: Christopher Miller

KEY
+24V
GND
Data
+24V motor
+12V (secondary)

Figure B.2: Drive train power system schematic

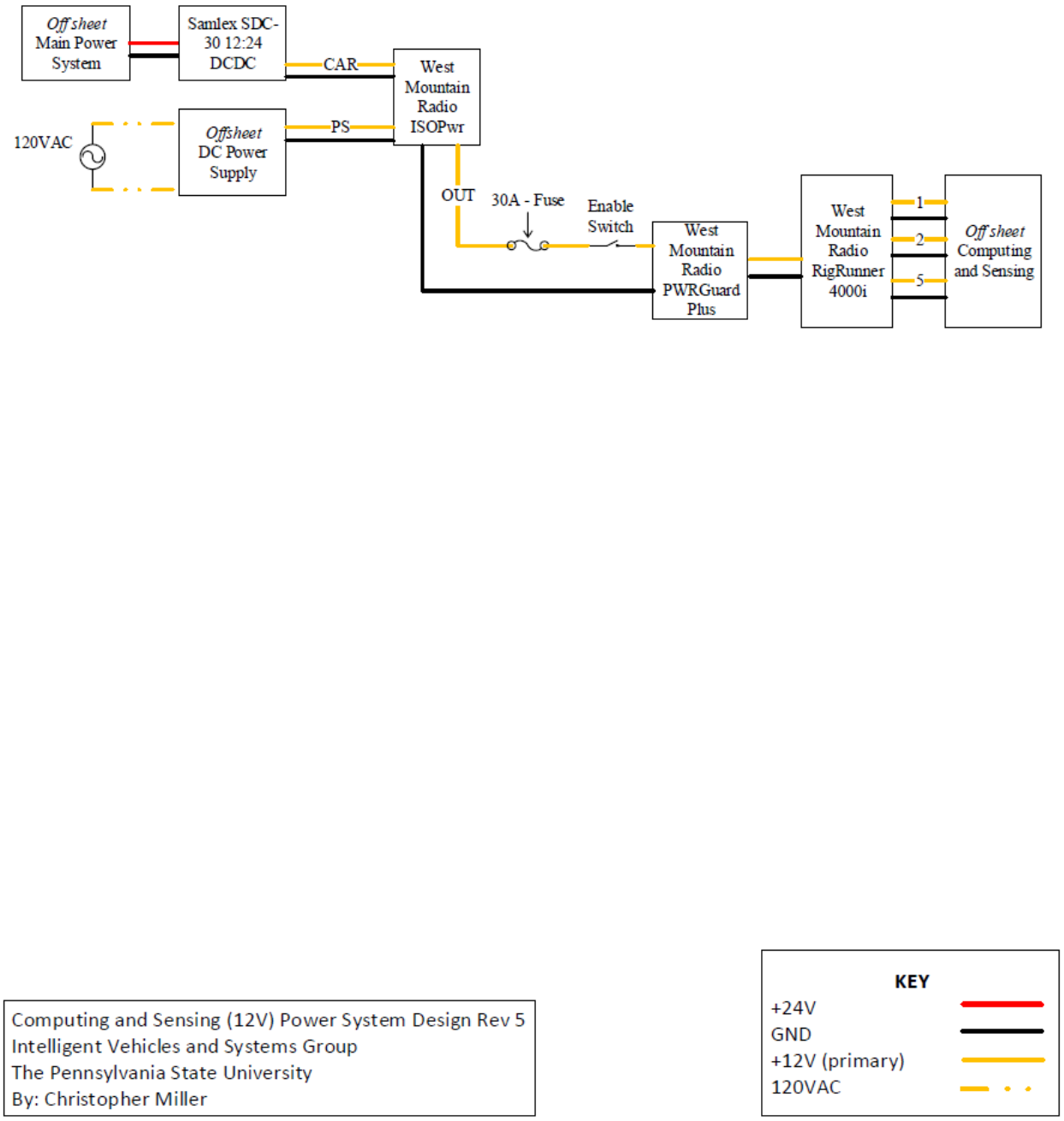


Figure B.3: Computing and sensing power system schematic

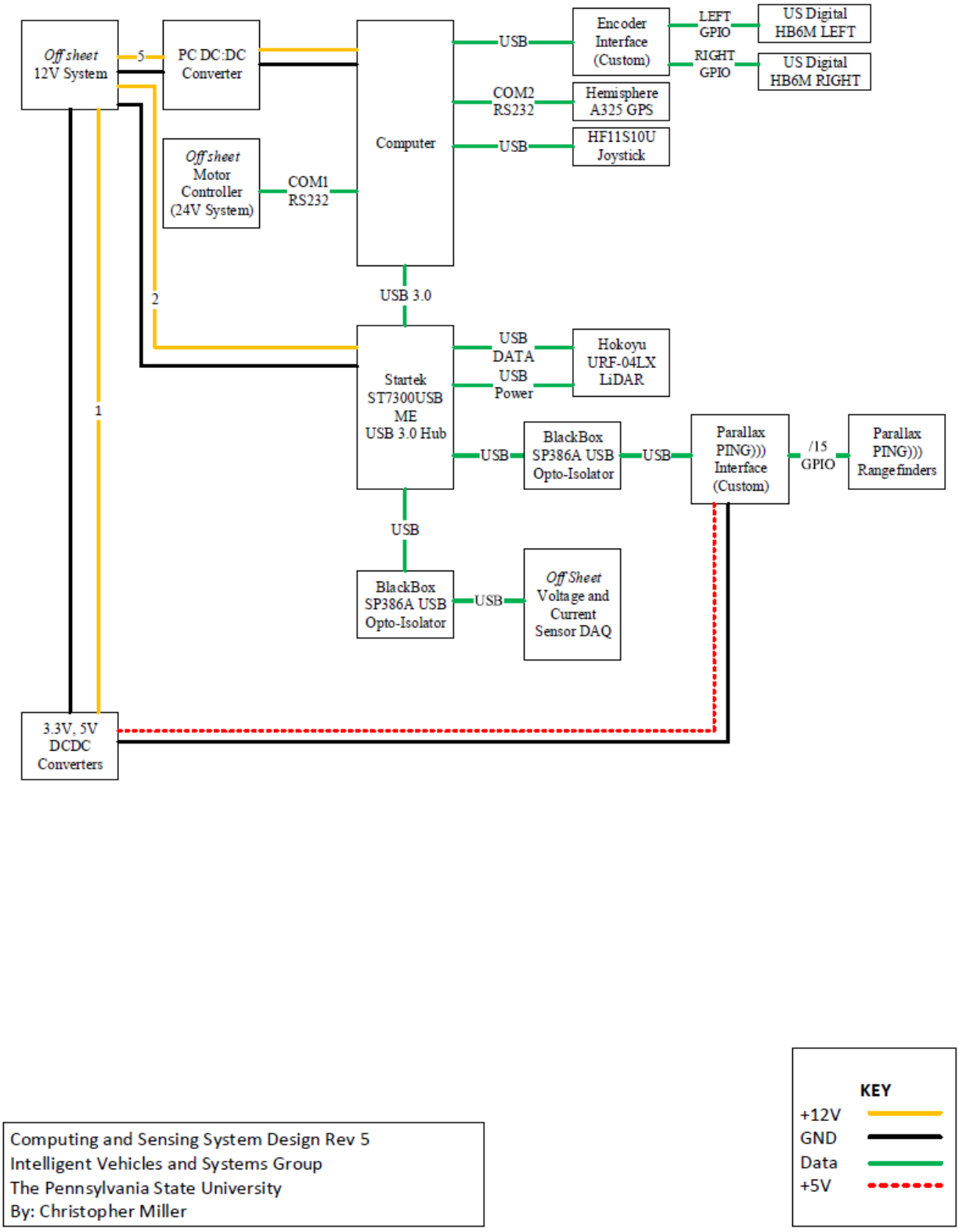


Figure B.4: Sensor interconnect schematic

Appendix C

Appendix C

The following includes the code used to characterize the voltage and current sensors. This section also includes the code used to capture the data; the capture code varied slightly depending on the sensor. The code provided was used to capture data from the current sensor. .

C.1 Voltage Sensor

The following includes the code used to capture and characterize the current sensor.

```
% Data Procesing for Characterization
% Christopher Miller
% Created on: 6/16/2015
% Modified: 6/16/2015

clc
close all
clear all
clc

%Read in the data file
filename = '2015_06_15_rawcounts24VSensor.csv';
data = csvread(filename);

%Convert the file to a 2xn vector

dim1 = 250; %Number of points
```

```
18 dim2 = 118; %Number of measurements
19
20 %Variables to store data
21 raw_vals = zeros(dim1*dim2/2,2);
22
23 %counters
24 x = 1;
25 j = 1;
26
27 %Transform the csv file
28 while j  < dim2
29     for i = 1:dim1
30         raw_vals(x,1) = data(i, j);
31         raw_vals(x,2) = data(i, j + 1);
32         x = x + 1;
33     end
34     j = j + 2;
35 end
36
37 %Calculate std devs per measurement value
38 stddevs = zeros(dim2/2,2);
39 covars = zeros(dim2/2,2);
40
41 %Counters and variables for parsing the data
42 j = 1;
43
44 %Measurement step sizes
45 init_val = 1;
46 max_val = 30;
47 step_size = .5;
48 val = init_val;
49
50 %min/max value for std/cov windows
51 window_size = dim1;
52 min = 1;
53 max = window_size;
54
55 %Calculate sensor mean
56 %sen_mean = round(mean(1 - raw_vals(min:max,1)));
57
58 while j < dim2 / 2 + 1
```

```
%Calculate standard deviation and covariance
stddevs(j,1) = val;
stddevs(j,2) = std(raw_vals(min:max ,1));
covars(j,1) = val;
covars(j,2) = cov(raw_vals(min:max ,1));

%Increment the counters
val = val + step_size;
min = min + window_size;
max = max + window_size;
j = j + 1;
end

%Average statistics
unfixed_stddev = mean(stddevs(:,2));
unfixed_covars = mean(covars(:,2));

%Plot the standard deviations for the Voltage sensor values
figure(1);
hold on;
plot(stddevs(:,1), stddevs(:,2), '*')
title('Standard Deviations v. Voltage (24V Sensor)');
xlabel('Voltage in .5V steps (V)')
ylabel('Standard Deviation (unitless)')
mean_val = num2str(unfixed_stddev);
s = strcat('Average Standard Deviation: ', mean_val);
text(5,.95,s);

%Plot the covariances for the voltage sensor values
figure(2);
plot(covars(:,1), covars(:,2), '*')
hold on;
title('Covariances v. Voltage (24V Sensor)');
xlabel('Voltage in .5V steps (V)')
ylabel('Covariance Value (unitless)')
mean_val = num2str(unfixed_covars);
s = strcat('Average Covariance: ', mean_val);
text(5,.95,s);

%Plot the raw data
figure(3)
```

```
plot(raw_vals(:,2), raw_vals(:,1), '*')
title('Raw ADC Counts v. Voltage');
xlabel('Voltage in .25V steps (V)')
ylabel('ADC Counts, 16-bit ADC fs = 200Hz')

%Remove the mean from the data
fixed_vals(:,1) = raw_vals(:,1); % - sen_mean;
fixed_vals(:,2) = raw_vals(:,2);

%Counter for fixed values...
counter = 0;

%Assuming the first point is okay else this code breaks...
for i = 2:(dim1*dim2)/2 - 1

    diff1 = abs((fixed_vals(i,1) - fixed_vals(i+1,1)) / fixed_vals(i,1));
    diff2 = abs((fixed_vals(i,1) - fixed_vals(i-1,1)) / fixed_vals(i,1));

    %If the value is > 25% out of line...
    if diff1 > .25 && diff2 > .25
        fixed_vals(i,1) = round((fixed_vals(i-1, 1) + fixed_vals(i+1,1))/2);
        counter = counter + 1;
    end

end

%Plot the processed data
figure(5)
plot(fixed_vals(:,2), fixed_vals(:,1), '*')
title('ADC Counts v. Voltage, 24V Sensor (Outliers Removed)');
xlabel('Voltage in .5V steps (V)');
ylabel('ADC Counts, 16-bit ADC fs = 200Hz');
counts_val = num2str(counter);
%mean_val = num2str(sen_mean);
s = strcat('Values Removed (< 25% difference) from prior value:', counts_val);
s2 = 'Removed values replaced with the midpoint of the values prior and post.';

text(5,1000,s);
text(5,500,s2);

%Find a trend line...
```

```
141 p = polyfit(fixed_vals(:,2), fixed_vals(:,1),1);
142
143 %Round to ints; the ADC only returns ints
144 %p = round(p)
145
146 %Fit the line...
147 yfit = polyval(p, fixed_vals(:,2));
148
149 %R^2 Computation
150 yresid = fixed_vals(:,1) - yfit;
151 SSresid = sum(yresid.^2);
152 SStotal = (length(fixed_vals(:,1)-1)*var(fixed_vals(:,1)));
153
154 rsq = 1 - SSresid/SStotal;
155
156 %Save the constants...
157 csvwrite('Voltage24SensorConstants.txt', p);
158
159 %Plot the fitted data
160 figure(6)
161 plot(fixed_vals(:,2), fixed_vals(:,1), '*')
162 hold on
163 plot(fixed_vals(:,2), yfit, 'r', 'LineWidth', 2)
164 title('ADC Counts v. Voltage, 24V Sensor (fitted Line and Collected Data)');
165 xlabel('Voltage in .5V steps (V)');
166 ylabel('ADC Counts, 16-bit ADC fs = 200Hz');
167 m_val = num2str(p(1));
168 b_val = num2str(p(2));
169 rsq_val = num2str(rsq);
170 s = strcat('Regression: y = ', m_val, ' * x + ', b_val);
171 s2 = strcat('r^2 = ', rsq_val);
172 s3 = 'Parameters rounded ints; ADC only returns ints';
173
174 %Place text onto plot
175 text(5,1650,s);
176 text(5,1100,s2);
177 text(5,650,s3);
178
179 %Statistics using the cleaner data
180
181 %Counters and variables for parsing the data
```

```
182 j = 1;
183
184 %Measurement step sizes
185 init_val = 1;
186 max_val = 30;
187 step_size = .5;
188 val = init_val;
189
190 %min/max value for std/cov windows
191 window_size = dim1;
192 min = 1;
193 max = window_size;
194
195 while j < dim2 / 2 + 1
196 %Calculate standard deviation and covariance
197 stddevs_clean(j,1) = val;
198 stddevs_clean(j,2) = std(fixed_vals(min:max ,1));
199 covars_clean(j,1) = val;
200 covars_clean(j,2) = cov(fixed_vals(min:max ,1));
201
202 %Increment the counters
203 val = val + step_size;
204 min = min + window_size;
205 max = max + window_size;
206 j = j + 1;
207 end
208
209 %Average statistics
210 fixed_stddev = mean(stddevs_clean(:,2));
211 fixed_covars = mean(covars_clean(:,2));
212
213 %Plot the standard deviations for the Voltage sensor values
214 figure(7);
215 hold on;
216 plot(stddevs_clean(:,1), stddevs_clean(:,2), '*')
217 title('Standard Deviations v. Voltage without Extreme Outliers (24V Sensor)');
218 xlabel('Voltage in .5V steps (V)')
219 ylabel('Standard Deviation (unitless)')
220 mean_val = num2str(fixed_stddev);
221 s = strcat('Average Standard Deviation: ', mean_val);
222 text(5,.95,s);
```

```
223
224 %Plot the covariances for the Voltage sensor values
225 figure(8);
226 plot(covars_clean(:,1), covars_clean(:,2), '*')
227 hold on;
228 title('Covariances v. Voltage without Extreme Outliers (24V Sensor)');
229 xlabel('Voltage in .5V steps (V)')
230 ylabel('Covariance Value (unitless)')
231 mean_val = num2str(fixed_covars);
232 s = strcat('Average Covariance: ', mean_val);
233 text(5,.95,s);
```

C.2 Current Sensor

```
% Data Procesing for Characterization
% Christopher Miller
% Created on: 6/16/2015
% Modified: 10/15/2015

clc
close all
clear all
clc

%Read in the data file
filename = '2015_02_21_rawcountsLEMCurrentSensor.dat';
data = csvread(filename);

%Convert the file to a 2xn vector

dim1 = 250; %Number of points
dim2 = 242; %Number of measurements

%Sampling period/freq
f_s = 82;
T_s = 1/f_s;

%Variables to store data
raw_vals = zeros(dim1*dim2/2,2);

%counters
x = 1;
j = 1;

%Transform the csv file
while j  < dim2
    for i = 1:dim1
        raw_vals(x,1) = data(i, j);
        raw_vals(x,2) = data(i, j + 1);
        x = x + 1;
    end
    j = j + 2;
```

```
39 end
40
41 %Calculate std devs per measurement value
42 stddevs = zeros(dim2/2,2);
43 covars = zeros(dim2/2,2);
44
45 %Counters and variables for parsing the data
46 j = 1;
47
48 %Measurement step sizes
49 init_val = 0;
50 max_val = 30;
51 step_size = .25;
52 val = init_val;
53
54 %min/max value for std/cov windows
55 window_size = dim1;
56 min = 1;
57 max = window_size;
58
59 %Calculate sensor mean
60 sen_mean = round(mean(raw_vals(min:max,1)));
61
62 while j < dim2 / 2 + 1
63 %Calculate standard deviation and covariance
64 stddevs(j,1) = val;
65 stddevs(j,2) = std(raw_vals(min:max ,1));
66 covars(j,1) = val;
67 covars(j,2) = cov(raw_vals(min:max ,1));
68
69 %Increment the counters
70 val = val + step_size;
71 min = min + window_size;
72 max = max + window_size;
73 j = j + 1;
74 end
75
76 %Average statistics
77 unfixed_stddev = mean(stddevs(:,2));
78 unfixed_covars = mean(covars(:,2));
79
```

```
%Plot the standard deviations for the current sensor values
figure(1);
hold on;
plot(stddevs(:,1), stddevs(:,2), '*')
title('Standard Deviations v. Current');
xlabel('Current in .25A steps (A)')
ylabel('Standard Deviation (unitless)')
mean_val = num2str(unfixed_stddev);
s = strcat('Average Standard Deviation: ', mean_val);
text(5,300,s);

%Plot the covariances for the current sensor values
figure(2);
plot(covars(:,1), covars(:,2), '*')
hold on;
title('Covariances v. Current');
xlabel('Current in .25A steps (A)')
ylabel('Covariance Value (unitless)')
mean_val = num2str(unfixed_covars);
s = strcat('Average Covariance: ', mean_val);
text(5,100000,s);

%Plot the raw data
figure(3)
plot(raw_vals(:,2), raw_vals(:,1), '*')
title('Raw ADC Counts v. Current');
xlabel('Current in .25A steps (A)')
ylabel('ADC Counts, 16-bit ADC fs = 200Hz')

%Remove the mean from the data
fixed_vals(:,1) = raw_vals(:,1);% - sen_mean;
fixed_vals(:,2) = raw_vals(:,2);

%Counter for fixed values...
counter = 0;

%acceptable error
err = .25;

%Assuming the first point is okay else this code breaks...
for i = 2:(dim1*dim2)/2 - 1
```

```
121
122     diff1 = abs((fixed_vals(i,1) - fixed_vals(i+1,1)) / fixed_vals(i,1));
123     diff2 = abs((fixed_vals(i,1) - fixed_vals(i-1,1)) / fixed_vals(i,1));
124
125     %If the value is > 25% out of line...
126     if diff1 > err && diff2 > err
127         fixed_vals(i,1) = round((fixed_vals(i-1, 1) + fixed_vals(i+1,1))/2);
128         counter = counter + 1;
129     end
130
131 end
132
133 %Plot the processed data
134 figure(4)
135 plot(fixed_vals(:,2), fixed_vals(:,1), '*')
136 title('ADC Counts v. Current; LEM CKSR50NP Sensor (Serial Outliers Removed)');
137 xlabel('Current in .25A steps (A)');
138 ylabel('ADC Counts, 16-bit ADC (f_s = 80.21Hz)');
139 counts_val = num2str(counter);
140 mean_val = num2str(sen_mean);
141 s = strcat('Values Removed (< 25% difference) from prior value:', counts_val);
142 s2 = 'Removed values replaced with the midpoint of the values prior and post.';
143 s3 = strcat('Constant Mean Value: ', mean_val);
144
145 %Place info on plot
146 text(2,1.45e4,s);
147 text(2,1.44e4,s2);
148 text(2,1.43e4,s3);
149
150 %Find a trend line...
151 p = polyfit(fixed_vals(:,2), fixed_vals(:,1),1)
152
153 %Fit the line...
154 yfit = polyval(p, fixed_vals(:,2));
155
156 %R^2 Computation
157 yresid = fixed_vals(:,1) - yfit;
158 SSresid = sum(yresid.^2);
159 SStotal = (length(fixed_vals(:,1)-1)*var(fixed_vals(:,1)));
160
161 rsq = 1 - SSresid/SStotal;
```

```
162
163 %Plot the fitted data
164 figure(5)
165 plot(fixed_vals(:,2), fixed_vals(:,1), '*')
166 hold on
167 plot(fixed_vals(:,2), yfit, 'r', 'LineWidth', 2)
168 title('ADC Counts v. Current for LEM CKSR 50-ND (Fitted Line)');
169 xlabel('Current in .25A steps (A)');
170 ylabel('ADC Counts, 16-bit ADC (f_s = 80.21Hz)');
171 m_val = num2str(p(1));
172 b_val = num2str(p(2));
173 rsq_val = num2str(rsq);
174 s = strcat('Regression: y = ', m_val, ' * x + ', b_val);
175 s2 = strcat('r^2 = ', rsq_val);
176 %s3 = 'Parameters rounded ints; ADC only returns ints';
177
178 %Place text onto plot
179 text(2,1.45e4,s);
180 text(2,1.44e4,s2);
181
182 %Statistics using the cleaner data
183
184 %Counters and variables for parsing the data
185 j = 1;
186
187 %Measurement step sizes
188 init_val = 0;
189 max_val = 30;
190 step_size = .25;
191 val = init_val;
192
193 %min/max value for std/cov windows
194 window_size = dim1;
195 min = 1;
196 max = window_size;
197
198 while j < dim2 / 2 + 1
199 %Calculate standard deviation and covariance
200 stddevs_clean(j,1) = val;
201 stddevs_clean(j,2) = std(fixed_vals(min:max ,1));
202 covars_clean(j,1) = val;
```

```
203 covars_clean(j,2) = cov(fixed_vals(min:max ,1));
204
205 %Increment the counters
206 val = val + step_size;
207 min = min + window_size;
208 max = max + window_size;
209 j = j + 1;
210 end
211
212 %Average statistics
213 fixed_stddev = mean(stddevs_clean(:,2));
214 fixed_covars = mean(covars_clean(:,2));
215
216 %Plot the standard deviations for the current sensor values
217 figure(6);
218 hold on;
219 plot(stddevs_clean(:,1), stddevs_clean(:,2), '*')
220 title('Standard Deviations v. Current without Extreme Outliers');
221 xlabel('Current in .25A steps (A)')
222 ylabel('Standard Deviation (unitless)')
223 mean_val = num2str(fixed_stddev);
224 s = strcat('Average Standard Deviation: ', mean_val);
225 text(5,.5,s);
226
227 %Plot the covariances for the current sensor values
228 figure(7);
229 plot(covars_clean(:,1), covars_clean(:,2), '*')
230 hold on;
231 title('Covariances v. Current without Extreme Outliers');
232 xlabel('Current in .25A steps (A)')
233 ylabel('Covariance Value (unitless)')
234 mean_val = num2str(fixed_covars);
235 s = strcat('Average Covariance: ', mean_val);
236 text(5,.5,s);
237
238 %10/15/2015 - Added code to properly calculate sigma-sqed value
239
240 %Define the linear regression vars
241 alpha = p(1);
242 mu = p(2);
243
```

```
%Convert the current to
current_hat = (fixed_vals(:,1) - mu) / alpha; %Convert from cnts to Amps
%'Truth', verified with a calibrated DC power supply from EE dept.
current = fixed_vals(:,2);

%Innovations
err = current - current_hat;

%Length of the vector (num pts)
n = size(current);

%Calc sigma squared (square of the RMS)
sig_sq = (1/n(1))*sum(err.^2);

disp('Sig Squared for Current Sensor: ');
sig_sq

%Save the values as a *.csv
x = [p, sig_sq];
csvwrite('CurrentSensorConstants.txt', x);


%End script 10/15/2015
```

C.3 Data Capture

The following includes the code used to capture and characterize the current sensor.

```
%Intelligent Vehicles and Systems Group
%LEM CKRS 50-NP Current Sensor Characterization data capture file
%Utilizes an Arduino UNO and the MATLAB serial drivers
%to pull current 16-bit ADC counts into MATLAB for processing and analysis
%Arduino Uno tied to COM4

%Script written by Christopher Miller (chris.x.miller@psu.edu)
%On 6/2/2015
%Last Edited 7/21/2015

%Current Sensor Characterization Script

%Clear and close stuff...
clc
clear all
close all
clear
clc

%Serial Stuff...
disp('Init the Serial Port');
%Open the serial port
s = serial('COM4', 'BaudRate', 57600);

%Set the timeout to 25ms
s.Timeout = 25;

%Make the input buffer smaller
%s.InputBufferSize = 8;

disp('Open the Serial Port');
%Open the port
fopen(s);
disp('Port Open!');

%Initialize lengths
max_val = 30; %Upper Bound
```

```
step_size = .25; %Increment size (units) (Delta)
samples = 250; %Number of samples per step
steps_total = max_val / step_size + 1; %Total number of steps
%Recording counts from the ADC, not voltages
data_arr = zeros(samples,2,steps_total); %Data Arr
measure = 0; %initial value of measurement (units)

%initialize the array
for i = 1:steps_total
    for j = 1:samples
        %initalize the current value being recorded
        data_arr(j, 2, i) = measure;
        j = j+1;
    end
    i = i+1;
    measure = measure + step_size;
end

%Get the samples into the data vec!
measure = 0;

%setup a counter...
j = 1;

%Prompt the user when ready to begin test...
r = input('Press enter when ready to begin.');
for i = 1:steps_total
    %Prompt user to set input
    disp('Set measured value to ');
    disp(measure);
    r = input('Return when ready to step');
%     %Flush the input buffer then collect
    flushinput(s);
%     disp('Waiting...');
%     pause(5);
    %Formatting...
    disp('Sampling...');
    disp(' ');

    %Sampling routine
    while(j <= samples)
```

```
79          %initalize the current value being recorded
80          val = str2num(fgets(s)); %data reading here..
81          %If it's empty, ignore it
82          if(isempty(val))
83              disp('Empty Val');
84          %else, save it
85          else
86              data_arr(j,1,i) = val(1);
87              %increment the ounter
88              j = j+1;
89          end
90
91      end
92      %Increment the counters...
93      measure = measure + step_size;
94      %zero out the counter
95      j = 1;
96  end
97
98  %Close the serial port, save the data...
99  disp('Test Complete.');
100 disp('Closing Serial Port...');
101 fclose(s);
102 disp('Port Closed. Saving data to .csv file...');
103 csvwrite('2015_02_21_rawcountsLEMCurrentSensor.dat', data_arr);
104 disp('CSVfile Saved.');
105 disp('Sampling Fre = 5ms');
```

Appendix D

Appendix D

The following includes the code used to process the battery discharge test results to determine the battery's parameters. The first section presents the code needed to determine the characteristics of the OCV-SOC curve for the wheelchair's lead acid batteries. The second section includes the code to fit the dynamics from the current pulse test.

D.1 OCV-SOC Parameters

```
%Intelligent Vehicles and Systems Group
%Penn State University
%Parse file to read in bag file from the current sensor, voltage sensor on
%on the wheelchair to generate OCVSOC Curve.
%Removes outliers and the stopping point(when the relay triggered)

%Script Written by Christopher Miller (chris.x.miller@psu.edu)
%Written on: 7/8/2015
%Edited on: 9/10/2015

%This file is for the second run of the OCVSOC curve using the fixed
%sensors, lower frequency, and higher current...

close all
clear all
clc

```

```
% Read in power data
% bagfile = '2015-07-26-01-55-56-OCV_SOC_DC_Load_newbats.txt';
% bagfile = '2015-08-25-17-53-09-OCVSOC1A2.txt';
 bagfile = '2015-09-01-13-38-26-OCVSOC1A3.txt';
power_data = csvread(bagfile);
power_12V = power_data(:,1); % Battery voltage, should be 12V
power_24V = power_data(:,2); % Battery voltage, should be 24V
power_current = power_data(:,3); % Battery current
stop_vec = power_data(:,4); %Record value (1 or 0)

%Stopping index (manual read of raw data)
len = 1;

while(1)
    if(stop_vec(len) == 0 && stop_vec(len+ 5) == 0 && ...
            stop_vec(len + 10) ==0)
        break
    end
    len = len + 1;
end

%Stopping index (manual read of raw data)
%cut the last two points (they tend to be flaky)
len = len-2;% len - 2;

%Sampling frequency
f_s = 79.73; %Hz
T_s = 1/f_s; %s

%Shorten vectors,
parsed_power_24V = power_24V(1:len);
parsed_power_current = power_current(1:len);

%Remove mega outliers (+/- 10%) for voltage; assumes first, last points are
%is "okay"
counter_24V = 0;
err = .05;
for i = 2:len-1

    diff1 = abs((parsed_power_24V(i,1) - parsed_power_24V(i+1,1)) / ...
        parsed_power_24V(i+1,1));
```

```
59      diff2 = abs((parsed_power_24V(i,1) - parsed_power_24V(i-1,1)) /...
60          parsed_power_24V(i-1,1));
61
62       %If the value is > err out of line... (this is the worst error...)
63      if diff1 > err && diff2 > err
64          parsed_power_24V(i,1) = round((parsed_power_24V(i-1, 1) ...
65                      + parsed_power_24V(i+1,1))/2);
66          counter_24V = counter_24V + 1;
67      end
68  end
69
70  %Remove mega outliers (+/- 25%) for voltage; assumes first, last points are
71  %is "okay"
72  counter_current = 0;
73  for i = 2:len-1
74
75      diff_1 = abs((parsed_power_current(i,1) - parsed_power_current(i+1,1)) / ...
76          parsed_power_current(i,1));
77      diff_2 = abs((parsed_power_current(i,1) - parsed_power_current(i-1,1)) /...
78          parsed_power_current(i,1));
79
80       %If the value is > 25% out of line...
81      if diff_1 > err && diff_2 > err
82          parsed_power_current(i,1) = round((parsed_power_current(i-1, 1) +...
83              parsed_power_current(i+1,1))/2);
84          counter_current = counter_current + 1;
85      end
86  end
87
88  %Convert to voltage, current
89  voltage = (parsed_power_24V - 139)/425;
90  current = (parsed_power_current - (13300.4779))/66.8083;
91
92  %Charge for SOC
93  coluombs = zeros(len,1);
94  coluombs_const = zeros(len,1);
95  cur = mean(current); %Amps, constant current
96  dT = T_s; %1/f_s or T_s
97
98  %Initial conditions
99  coluombs_const(1) = cur*dT; %current(1)*dT;
```

```
100 coluombs(1) = current(1)*dT;
101
102 for(i = 2:len)
103    coluombs_const(i) = coluombs_const(i-1) + cur*dT;
104    coluombs(i) = coluombs(i-1) + current(i)*dT;
105 end
106
107 %Gen SOC Vecs
108 for(i = 1:len)
109     SOC(i) = coluombs(i)/coluombs(end);
110     SOC_const(i) = coluombs_const(i)/coluombs_const(end);
111 end
112 %Remove internal resistance voltage....
113 r_int = 0.055;
114
115 %Compensate for internal resistance values
116 for i = 1:len
117     volts_const_cur(i) = voltage(i) + r_int*cur;
118     volts_dyn_cur(i) = voltage(i) + r_int*current(i);
119 end
120
121 %Generate time vec
122 for i = 1:len
123     time(i) = (i*T_s)/3600;
124 end
125
126 %PLOTS PLOTS PLOTS EVRYBODY
127 figure(1)
128 %Plot the curve vs charge
129 plot(coluombs(1:end), volts_dyn_cur(1:end));
130 title('OCVSOC Curve for Inverted DC Power Supply Load,...
131     R_i_n_t Added, Wheelchairs 16-bit ADC, Third Discharge');
132 xlabel('Charge (C)');
133 ylabel('Voltage (V)');
134 axis([0 1.16e5 23 27.8])
135
136 figure(3)
137 plot(SOC, fliplr(volts_dyn_cur))
138 hold on
139 plot(SOC_const, fliplr(volts_const_cur), 'r')
140 title('OCVSOC Curve for Inverted DC Power Supply Load Test Three');
```

```
141 xlabel('SOC');
142 ylabel('Voltage (V)');
143 axis([-.01 1.02 23 27.8])
144 legend('Measured Coulomb Count', 'Theoretical Coulomb Count')
145
146 disp('Coluombs Count:')
147 coluombs(end)
148 disp('Const and integrated difference:')
149 coluomb_count_diff = coluombs(end) - coluombs_const(end)
150 disp('Percent Error: ' );
151 err = (coluomb_count_diff/coluombs(end))*100
152
153 %Plot linear region of curve
154 figure(4)
155 volts_linear = fliplr(volts_dyn_cur(round(.2*len):.8*len));
156 coluombs_linear = SOC(round(.2*len):.8*len)';
157 plot(coluombs_linear,volts_linear)
158 hold on
159
160 %region 2 regrssion and plotting...
161 p = polyfit(coluombs_linear, volts_linear', 1);
162 yfit = p(1)*coluombs_linear + p(2);
163 plot(coluombs_linear, yfit, 'r')
164 title('Linear Region of OCVSOC Curve for OCVSOC Test Three')
165 xlabel('SOC');
166 ylabel('Voltage (V)');
167 legend('Actual','Fit');
168 axis([.19 .81 23 27.8])
169
170 %Calculate r^2
171 yresid = volts_linear - yfit';
172 SSresid = sum(yresid.^2);
173 SStotal = (length(volts_linear)-1) *var(volts_linear);
174 rsq = 1 - SSresid/SStotal;
175 disp('r^2 for the fit is...');
176 rsq
177 disp('Coeffs of the curve fit...');
178 p
179
180 disp('Average Current:');
181 mean(current)
```

D.2 Dynamic Parameters

D.2.1 Runner File

```
%Intelligent Vehicles and Systems Group
%Penn State University
%Parse file to read voltage and current data recorded on the wheelchair
%Parses the data and finds a model fit.

%This file assumes the data read in occurs after the first 20% of the
%batteries have been discharged to place the system into the linear region
%of the OCV-SOC curve. This code will not work if this has not been
%completed.

%Script Written by Christopher Miller (chris.x.miller@psu.edu)
%Written on: 9/10/2015
%Edited on: 9/20/2015

close all
clear all
clc

% Read in power data
bagfile = 'ModelFitData.txt'; % Date and time stamp of desired bag file
power_data = csvread(bagfile);
time = power_data(:,1); % Read in the time vector, convert to seconds
voltage = power_data(:,2); % Battery voltage, should be 24V
current = power_data(:,3); % Battery current

%Time_init_Adjust
init_time = time(1);
time = time - init_time;

%length of data vectors
len = length(time);

%Read in other constants
misc_constants = csvread('miscvals.txt');

%Total Charge
```

```
Q_0 = misc_constants(1); %C

%Internal Resistance
R_int = 0.16;% misc_constants(2); %Ohms

%Sampling frequency
f_s = misc_constants(3); %Hz
dT = 1/f_s; %s

%Read in the SOC Information
OCVSOC_info = csvread('OCVSOCConstants.txt');

%Define alpha, SOC to OCV
alpha = OCVSOC_info(1);
%Define constant mean, mu
mu = OCVSOC_info(2);

%Time to fit the model....

%Since 20% has been discharged, SOC_0 is .8
SOC_0 = .8;
SOC = zeros(len,1);
SOC(1) = SOC_0;

%Generate a SOC vector...
for(i = 2:len)
    %Depleating SOC with time...
    SOC(i) = SOC(i-1) - ((current(i)*dT)/Q_0);
    %the SOC prior will be greater than the present SOC. It will decrease
    %by the % of charge lost. % charge = charge used in timestep/total
    %charge. Charge used in timestep = sampled current * dT (timestep)
end

%Initial guess for the two terms
init_guess = [.5 1.2e5 alpha mu .156];
% [ -1/tau 1/C alpha, mu, R_int]

Q_0_dt = 8.8266e+06;

%FIT THE MODEL

```

```
78 opts = optimoptions(@lsqnonlin, 'MaxFunEvals', 10000);
79 vars = lsqnonlin(@(m) batteryFit_est_alpha_mu_R_int(m, voltage, current,....
80     SOC, Q_0_dt,dT), init_guess);
81
82 %Plot model vs measurement
83
84 %Run the simulation
85 volt_compare = batterySim(vars, current, SOC, Q_0_dt, dT);
86
87 %plot the model
88 figure(1)
89 plot(time, volt_compare)
90 hold on
91 plot(time, voltage)
92
93 title('Model Battery Voltage vs. Measurement Battery Voltage');
94 xlabel('Time (s)');
95 ylabel('Voltage (V)');
96 legend('Prediction', 'Measurement');
97
98 %Calculate error
99 err = 100*((voltage - volt_compare)./voltage);
100 %new figure, yo
101 figure(2)
102 plot(time, err);
103
104 title('Model Battery Voltage Error');
105 xlabel('Time (s)');
106 ylabel('% Error');
107
108 mean(err)
109
110 csvwrite('2ndOrderParams.txt', vars);
```

D.2.2 Fitting Function

```
function [ dy ] = batteryFit_est_alpha_mu_R_int( vec, voltage, current, SOC, Q_0, dT)
%UNTITLED4 Function to read in battery data and guesstimate the params
%   Detailed explanation goes here
%Vec(1) = tau
%Vec(2) = C
%Vec(3) = alpha
%Vec(4) = mu

    %Calculate the discrete time SOC from the current vector
    xpart = .80 - (1/Q_0) * cumsum(current)*dT;

    %Calculate the Open Circuit Voltage
    V_ocv = SOC*vec(3) + vec(4);

    %Calculate the direct response (zero state response)
    V_dir = current*vec(5);

    %Exponential for convolution
    exparr = exp(-(dT/(vec(1))*(1:length(current))'));

    %Transient response (due to current...) (Zero input response)
    V_c = ((1/vec(2))*dT*fconv(current, exparr));

    %Cut the extra values from the convolution function...
    V_c = V_c(1:numel(current));

    %Voltage guess (this is from the circuit diagram ... Subtract
    y = V_ocv - V_dir - V_c;
%
    %Calculate the difference between 'truth' and 'guess'
    dy = (voltage - y);
       %Subtrace the constant offset from the voltage, then find the error
       %between "truth" and the estimate.
end
```

Appendix E

Appendix E

The following includes the code used to simulate the battery pack state estimator in MATLAB.

```
%Intelligent Vehicles and Systems Group
%Penn State University
%Reads in real data, simulates the SOC estimator to be implemented on the
%wheelchair

%Script Written by Christopher Miller (chris.x.miller@psu.edu)
%Written on: 02/27/2016
%Edited on: 02/27/2016

%Clear the space
close all
clear all
clc

%Read in the data
data = csvread('2016-02-25-12-15-09.txt');
%data = csvread('2015-11-12-17-11-31-QMatrixData.txt');

%Parse the data
volts_cnts = data(:,2);
current_cnts = data(:,3);

%Length variable
len = length(volts_cnts);

```

```
%Read in constants for conversion
volt_sens = csvread('Voltage24SensorConstants.txt');
cur_sens = csvread('CurrentSensorConstants.txt');

%Convert from counts to...
current_uf = (current_cnts - 13318)/cur_sens(1); %[A]
voltage_uf = (volts_cnts - volt_sens(2))/volt_sens(1); %[V]

%Filter out the BS from
for i = 1:len
    if voltage_uf(i) < 0
        voltage_uf(i) = voltage_uf(i-1);
    end

    if current_uf(i) < 0
        current_uf(i) = current_uf(i-1);
    end
end

current = current_uf;%step(LP_FIR, current_uf);
voltage = voltage_uf;%step(LP_FIR, voltage_uf);

%Read in Discrete-type state matricies
Ad = csvread('A_d.txt');
Bd = csvread('B_d.txt');
Cd = csvread('C_d.txt');
Dd = csvread('D_d.txt');

%System Params
params = csvread('2ndOrderParams.txt');
misc = csvread('miscvals.txt');
f_s = misc(3);%Hz
dT = 1/f_s;%s
SOC_0 = 1.00;% 80% SOC for linear region
V_ct_0 = 0.05; %Assume 0.05V for the capactitive voltage. Seems right
mu = params(4);%DC offset in the battery
Q_0 = misc(1); %Total battery charge

%Noise Params
current_vars = csvread('CurrentSensorConstants.txt')';
cur_sens_sig_sq = current_vars(3); %R matrix stuff
```

```
67
68 %State vector and initial states
69 x_hat = zeros(2, len);
70 x_hat(1,1) = V_ct_0; %V
71 x_hat(2,1) = SOC_0; %*ones(1,len); %
72
73 %Define sensor noise
74 s_noise = 5.23653;
75
76 %Q matrix, by defn
77 sig_Vct = 0.3356*10;
78 sig_SOC = 0.011/10;
79
80 Q = [sig_Vct 0; 0 sig_SOC];
81
82 %Define the R matrix
83 R = s_noise;
84
85 %Initalize the matrix
86 P = zeros(2,2,len);
87 P(:,:,1) = eye(2,2); %eye(2)*.001;
88
89 % Preallocate marices
90 K = zeros(2,len);
91 sqrtP = zeros(1,len);
92 inov = zeros(1,len);
93 y_hat = zeros(1,len);
94 volt_est_hat = zeros(1,len);
95 vct_covar = zeros(1,len);
96 soc_covar = vct_covar;
97
98 %Initalize the Coulomb Counter
99 cmbcnt = zeros(1,len);
100 cmbcnt(1) = 1;
101
102 %Turn on the switches
103 sw_1 = 3;
104 sw_2 = 1;
105
106 %Run the estimator
107 for k = 2:len
```

```
108     if x_hat(2,k-1) >=.9  && sw_1 == 3
109         sw_1 = 1;
110     elseif x_hat(2,k-1) < .9 && x_hat(2,k-1) > .1 && sw_1 == 1
111         sw_1 = 0;
112     elseif x_hat(2,k-1) < .1 && sw_1 == 0
113         sw_1 = 1;
114     end
115
116     if  sw_1 == 1
117         %This simulates the coloumb counter
118         x_hat(2,k) = x_hat(2,k-1) - inv(Q_0)*cumsum(current(k))*dT;
119         sw_1 = 1;
120
121      elseif sw_1 == 0
122         %Do Once
123         if sw_2 == 1
124             %Set the initial y_hat
125             y_hat(k) = voltage(k);
126             %Initial P matrix
127             P(:,:,k-1) = Q;
128             %Initial V_ct state
129             x_hat(1,k-1) = 0.15;
130             %Open the switch
131             sw_2 = 0;
132         end
133
134         %Time Update measurement
135         x_hat(:,k) = Ad*x_hat(:,k-1) + Bd*current(k-1);
136         P(:,:,k) = Ad*P(:,:,k-1)*Ad' + Q;
137
138         %Measurement update the system
139         %Innovations
140         inov(k) = (voltage(k) - mu) - (Cd*x_hat(:,k) + current(k)*Dd);
141         K(:,k) = P(:,:,k)*Cd'*inv(Cd*P(:,:,k)*Cd' + R);
142         x_hat(:,k) = x_hat(:,k) + K(:,k)*(inov(k));
143         P(:,:,k) = (eye(2) - K(:,k)*Cd)*P(:,:,k);
144
145         %For plotting
146         y_hat(k) = Cd*x_hat(:,k) + Dd*current(k);
147         volt_est_hat(k) = y_hat(k) + mu;
148
```

```
149         %Sqrt of trace of P
150         sqrtP(k) = sqrt(trace(P(:,:,k)));
151 
152         %Save covariances
153         vct_covar(k) = P(1,1,k);
154         soc_covar(k) = P(2,2,k);
155     end
156 
157     %Use pure Coloumb counting for comparison
158     cmbcnt(k) = cmbcnt(k-1) - inv(Q_0)*cumsum(current(k))*dT;
159 
160 end
161 
162 figure(1)
163 plot(x_hat(2,:))
164 title ('SOC');
165 
166 figure(2)
167 plot(x_hat(1,:))
168 title ('V_C_T');
169 
170 figure(3);
171 plot(voltage);
172 hold on
173 plot(volt_est_hat, 'r');
174 title('Voltage vs Est. Voltage');
175 legend('Measurement', 'Estimate');
176 
177 figure(4)
178 plot(vct_covar);
179 hold on
180 plot(soc_covar, 'r');
181 title('State Covariances');
182 legend('VCT Covar', 'SOC Covar');
```

Bibliography

[1] U. S. C. Bureau. (2010, May) 20th anniversary of americans with disabilities act: July 26. [Online]. Available: https://www.census.gov/newsroom/releases/archives/facts_for_features_special_editions/cb10-ff13.html

[2] R. R. S. C. by KD Healthcare Company USA. (2015, Janurary) Wheelchair facts, numbers and figures [infographic]. [Online]. Available: http://kdsmartchair.com/blogs/news/18706123-wheelchair-facts-numbers-and-figures-infographic

[3] IBISWorld. (2016, January) Ibisworld business environment profiles: Number of adults aged 65 and older. [Online]. Available: http://clients1.ibisworld.com/reports/us/bed/default.aspx?bedid=22

[4] O. Horn, "Smart wheelchairs: past and current trends; the example of the vahm," in *International Conference on Systems and Computer Science*, vol. 1. IEEE, August 2012.

[5] R. Simpson, E. LoPresti, and R. Cooper, "How many people would benefit from a smart wheelchair?" *Journal of Rehabilitation Research and Development*, vol. 24, no. 1, pp. 53–72, 2008.

[6] J. Aylor, A. Thieme, and B. Johnson, "A battery state-of-charge indicator for electric whelchairs," *IEEE Transactions on Industrial Electronics*, vol. 29, no. 5, pp. 398–409, October 1992.

[7] T. Taha, J. V. Miro, and G. Dissanayake, "Pomdp-based long-term user intention prediction for wheelchair navigation," in *IEEE International Conference on Robotics and Automation*. IEEE, May 2008, pp. 3920–3925.

[8] T. Carlson and Y. Demiris, "Collaborative control for a robotic wheelchair: Evaluation of performance, attention, and workload," *IEEE TRANSACTIONS ON SYSTEMS,MAN, AND CYBERNETICSPART B: CYBERNETICS*, vol. 42, no. 3, pp. 876 – 888, June 2012.

[9] R. L. Madarasz, L. C. Heiny, R. F. Cromp, and N. M. Mazur, "The design of an autonomous vehicle for the disabled," *IEEE Journal of Robotics and Automation*, vol. RA-2, no. 3, pp. 117 – 126, September 1986.

[10] R. C. Simpson, S. P. Levine, D. A. Bell, L. A. Jaros, Y. Koren, and J. Borenstein, "Navchair: An assistive wheelchair navigation system with automatic adaptation," in *Assistive Technology and Artificial Intelligence*, ser. Lecture Notes in Computer Science, vol. 1458. Springer Berlin Heidelberg, 1998, pp. 235–255.

[11] A. Bonarini, S. Ceriani, G. Fontana, and M. Matteucci, "Introducing lurch: a shared autonomy robotic wheelchair with multimodal interfaces," in *In Proceedings of IROS 2012 Workshop on Progress, Challenges and Future Perspectives in Navigation and Manipulation Assistance for Robotic Wheelchairs*. IEEE, 2012, pp. 1–6.

[12] A. C. Lopes, G. Pires, and U. Nunes, "Assisted navigation for a brain-actuated intelligent wheelchair," *Robotics and Autonomous Systems*, vol. 61, no. 3, pp. 245–258, 2013.

[13] G. Pires and U. Nunes, "A wheelchair steered through voice commands and assisted by a reactive fuzzy-logic controller," *Journal of Intelligent and Robotic Systems*, vol. 34, no. 3, pp. 301–314, jul 2002.

[14] M. Barnes, "Autonomous wheelchair control," 2013, matt Barnes' Honors Thesis, Schreyer Honors College, Penn State University. This thesis served as the idea for the wheelchair resrarch.

[15] R. A. Cooper, T. Thorman, R. Cooper, M. J. Dvorznak, S. G. Fitzgerald, W. Ammer, G. Song-Feng, and M. L. Boninger, "Driving characteristics of electric-powered wheelchair users: How far, fast, and often do people drive?" *Archives of Physical Medicine and Rehabilitation*, vol. 83, no. 2, pp. 250–255, 2002.

[16] R. A. Cooper, D. P. VanSickle, S. J. Albright, K. J. Stewart, M. Flannery, and R. N. Robertson, "Power wheelchair range testing and energy consumption during fatigue testing," *Journal of rehabilitation research and development*, vol. 32, no. 3, pp. 255–263, 1995.

[17] P.-C. Chen and Y.-F. Koh, "Residual traveling distance estimation of an electric wheelchair," in *2015 5th International Conference on BioMedical Engineering and Informatics (BMEI 2012)*. IEEE, pp. 790–794.

[18] D. Bouquain, B. Blunier, and A. Miraoui, "A hybrid fuel cell/battery wheelchair - modeling, simulation and experimentation," in *Vehicle Power and Propulsion Conference*. IEEE, oct 2008.

[19] Y.-P. Yang, R.-M. Guan, and Y.-M. Huang, "Hybrid fuel cell powertrain for a powered wheelchair driven by rim motors," *Journal of Power Sources*, pp. 192–204, 2012.

[20] Z. Zou, J. Xu, C. Mi, B. Cao, , and Z. Chen, "Evaluation of model based state of charge estimation methods for lithium-ion batteries," *Energies*, vol. 7, pp. 5065–5082, 2014.

[21] C. Zhu, M. Coleman, and W. Hurley, "State of charge determination in a lead-acid battery: Combined emf estimation and ah-balance approach," in *2004 35th Annual IEEE Power Electronics Specialists Conference*. IEEE, 2004, pp. 1908–1914.

[22] M. Coleman, C. K. Lee, C. Zhu, and W. G. Hurley, "State-of-charge determination from emf voltage estimation: Using impedance, terminal voltage, and current for lead-acid and lithium-ion batteries," *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS*, vol. 54, no. 5, pp. 2550–2557, oct 2007.

[23] C. Weng, J. Sun, and H. Peng, "A unified open-circuit-voltage model of lithium-ion batteries for state-of-charge estimation and state-of-health monitoring," *Journal of Power Sources*, vol. 258, pp. 228–237, jul 2014.

[24] G. L. Plett, *Battery Management Systems*, 1st ed. 685 Canton Street: Artech House, 2015, vol. 1.

[25] ——, "Extended kalman filtering for battery management systems of lipb-based hev battery packs part 1. background," *Journal of Power Sources*, vol. 134, no. 2, pp. 252–261, aug 2004.

[26] ——, "Extended kalman filtering for battery management systems of lipb-based hev battery packs part 2. modeling and identification," *Journal of Power Sources*, vol. 134, no. 2, pp. 262–276, aug 2004.

[27] M. Coleman, W. G. Hurley, and C. K. Lee, "An improved battery characterization method using a two-pulse load test," *IEEE TRANSACTIONS ON ENERGY CONVERSION*, vol. 23, no. 2, pp. 708–713, jun 2008.

[28] G. L. Plet, "Extended kalman filtering for battery management systems of lipb-based hev battery packs: Part 3. state and parameter estimation," *Journal of Power Sources*, vol. 134, no. 2, p. 277292, aug 2004.

[29] I. N. Laboratory and B. E. Alliance, "Battery test manual for plug-in hybrid electric vehicles revision," US Department of Energy: Vehicle Technologies Program, Tech. Rep. 0, 2008, freedom Car Report.

[30] R. A. Jackey, G. L. Plett, and M. J. Klein, "Parameterization of a battery simulation model using numerical optimization methods," The MathWorks, Tech. Rep., 2009.

[31] R. A. Jackey, "A simple, effective lead-acid battery modeling process for electrical system component selection," The MathWorks Inc, Tech. Rep., 2007.

[32] K.-S. Ng, Y.-F. Huang, C.-S. Moo, and Y.-C. Hsieh, "An enhanced coulomb counting method for estimating state-of-charge and state-of-health of lead-acid batteries," in *The 31th International Telecommunications Energy Conference*, vol. 31. IEEE, 2009.

[33] D. Simon, *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*, 1st ed. Wiley-Interscience, jun 2006.

[34] F. Zhang, G. Liu, and L. Fang, "Battery state estimation using unscented kalman filter," in *2009 IEEE International Conference on Robotics and Automation*, IRAS. IEEE, may 2009, pp. 1863–1868.

[35] F. Codeca, S. M. Savaresi, and G. Rizzoni, "On battery state of charge estimation: a new mixed algorithm," in *17th IEEE International Conference on Control Applications, Part of 2008 IEEE Multi-conference on Systems and Control*. IEEE, 2008.

[36] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*. IEEE, 2009.

[37] (2016, jan) Arduino uno product spec. [Online]. Available: https://www.arduino.cc/en/Main/ArduinoBoardUno

[38] U. Digital, *HB6M Hollow Bore Optical Encoder*, US Digital.

[39] *Scanning Scanning Laser Range Finder Laser Range Finder URG-04LX Specifications*, 5th ed., Hokuyo Automatic Co. LTD., apr 2008.

[40] *PING))) Ultrasonic Distance Sensor*, Parallax Inc, jan 2016, https://www.parallax.com/product/28015.

[41] *A325 GNSS Smart Antenna*, 2nd ed., Hemisphere GPS, 2012.

[42] *HF Series; Hall effect joysticks*, Apem Inc.

[43] A. E.-N. Gene F. Franklin, J. David Powell, *Feedback Control of Dynamic Systems*, 7th ed. Pearson Higher Education, 2015.

[44] *UPG Sealed Lead-Acid Battery UB12350*, UPG, 1720 Hayden Drive, Carrollton, TX 75006, 2016.

[45] K. Nice, apr 2001.

CHRISTOPHER X. MILLER

Academic Vita

chris.x.miller@gmail.com

Education

Bachelor of Science in Electrical Engineering — Aug. 2012 – May 2016
The Pennsylvania State University, University Park, PA
Schreyer Honors College, Presidential Leadership Academy

Work and Leadership Experience

National Robotics Engineering Center (NREC), *Carnegie Mellon University; Pittsburgh, PA*
Electrical Engineer (Jun. 2016 – Present)

Intelligent Vehicles and Systems Group, *Penn State University; University Park, PA*
Undergraduate Researcher (Jun. 2013 – May 2016)

- Retrofitted an electric wheelchair with various sensors (encoders, LiDAR, voltage/current monitors, etc.), an onboard Linux-based computing system, safety systems, and a custom power management system for automation
- Designed, built, and debugged a prototype PCB to measure current and voltage from the wheelchair's battery pack
- Publishing a conference paper on a model-based energy-usage prediction algorithm using a Kalman filter to estimate wheelchair electric range; simulated system in MATLAB, realized system in Python on wheelchair
- Designing hardware and algorithms for positive and negative obstacle detection using low-cost rangefinders
- Planning implementation of biological interface control methods such as EEG, EMG, and ocular tracking

Penn State IEEE, *Penn State University; University Park, PA*
President (Apr. 2014 – Apr. 2015)

- Allocated $55,000 (increased from $12,000) in annual corporate sponsorship to increase student involvement in the IEEE and EE department, represented EE student body to department officials, led 12-person officer team, and oversaw 50+ sponsored events (20-250 attendees)
- Transitioned organization from a startup club to a mature organization through managerial restructuring, leadership consolidation, and writing a 5-10 year fiscal plan; established fully-stocked student-run EE lab
- Chaired outreach committee, managing $4000 in funds and leading 15 team members to organize an annual, state-wide, K-12 robotics competition, hosting nearly 100 students

NASA's Jet Propulsion Laboratory (JPL), *Pasadena, CA*
Research Intern (May 2014 – Aug. 2014)

- Developed embedded software in C for a 32-channel, high-speed DAQ based around the PIC32MX; wrote drivers in MATLAB and C++ to interface DAQ with existing computer systems; developed benchmark tests in MATLAB and Python for DAQ verification; and integrated wireless capabilities for DAQ

- Fabricated, calibrated, and tested custom EMG electrodes for the JPL BioSleeve V3: a surface EMG-based gesture recognition system for advanced robotic control

Student Space Programs Laboratory, *Penn State University; University Park, PA*
Undergraduate Researcher (Aug. 2012 – Dec. 2013)

- Updated Command and Data Handling (C&DH) motherboard schematics to better reflect the updated needs of the OSIRIS CubeSat subsystems using Altium Designer
- Led the design and construction of a small rocket payload (1.2-2k ft altitude) to collect telemetry and weather data by serving as the systems engineer of a 14-person team

Technical Skills

Code	**Proficient:** MATLAB, C/C++, Python, ROS, LaTeX; **Basic:** Java, Simulink, LabVIEW
Software	**Proficient:** Linux (Ubuntu), SVN, GIT; **Basic:** Altium Designer, SolidWorks, Multisim, Ultiboard
Hardware	**Proficient:** Soldering (PTH & SMT), PCB fabrication and testing, PIC, mBed, Arduino, SPI, I2C, UART; **Basic:** LiDAR, encoder systems, bio-sensing systems (EMG/EEG)
Other	**Proficient:** Technical writing, Battery characterization and modeling; **Basic:** Kalman filters, particle filters

Honors and Awards

Rodney A. Erickson (Research) Discovery Grant ($5000)	May 2015 – Aug. 2015
Third Place in Engineering, Penn State Undergraduate Research Symposium	Apr. 2015
College of Engineering Research Initiative (CERI) Grant ($5000)	Aug. 2014 – May 2015
PA Space Grant Consortium Summer Research and Travel Grant ($7000)	Jun. 2014 – Aug. 2014
Best Engineered Design, Penn State College of Engineering Design Project Showcase	Dec. 2012
Penn State College of Engineering Research Scholarship ($16,000)	May 2011

Certifications and Memberships

National Instruments CLAD - 100-314-7770	Apr. 2015 – Apr. 2020
Eta Kappa Nu Electrical Engineering Honor Society	Apr. 2014 – Present
Chairperson, Penn State School of EECS Undergraduate Advisory Board	Dec. 2015 – May 2016
FCC Technician Class Ham Radio License - KC3AMP	Apr. 2013 – Apr. 2023
Institute of Electrical and Electronics Engineers	Aug. 2012 – Present