THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

Characterisation of Rechargable Batteries

Addressing Fractional Ultralow-Frequency Devices

by

Vance Farrow

A thesis submitted in fulfilment for the
degree of Master of Engineering

in the
Faculty of Science and Engineering
University of Waikato

September 2020

Declaration of Authorship

I, Vance Farrow, declare that this thesis titled, 'Characterisation of Rechargable Batteries: Addressing Fractional Ultralow-Frequency Devices' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

Acknowledgements

I would like to thank WaikatoLink for their funding and support of this project. I am grateful to have had Professor Jonathan Scott as my supervisor and for his support and guidance to ensure the best possible results from this work. I have enjoyed working alongside the other members of the research group who have provided many hours of entertainment and deep discussion and thank them for their camaraderie. Finally, I extend my unending gratitude to my family and friends who have provided their love and support in abundance.

Abstract

There is a need for an improved method of detecting changes in battery state-of-health. Current methods are inaccurate and lead to early battery retirement and low consumer trust due to public failures of battery management systems such as those in the Nissan Leaf. Recent developments in battery modelling suggest the potential for small signal, ultralow-frequency measurements that will allow the reliable detection of changes in battery state-of-health.

To verify this method of determining state-of-health an experiment has been performed. Systematic measurement and degradation of the state-of-health of a battery through cycling, and regular measurements of the low frequency impedance, has provided conclusive evidence in support of the proposed method of detecting changes in state-of-health. For the purpose of running this experiment, and others like it in the future, tools have been developed for the systematic degradation and measurement of battery state-of-health, and for taking ultralow-frequency measurements to detect changes in state-of-health.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The use of batteries has permeated modern life. Batteries are used in all facets of daily existence, from use in minor conveniences, such as car key remotes, to indespensible assets, such as cars and cell phones, and life saving equipment, such as medical implants and defibrillators. Battery reliability is a prominent issue where the life cycle of critical items is defined by the working life of the battery that powers them [1]. Battery degradation occurs both over time (calendar aging) and with use (cycling aging), and is related to battery chemistry, environmental conditions, and use patterns. Monitoring the state-of-health of batteries allows preventative maintenance, timely replacement, and increased reliability. Modern methods of monitoring battery state-of-health are unreliable, famously so in the case of the Nissan Leaf. The only widely accepted method for reliably and accurately measuring state-of-health requires controlled cycling of the battery. This method cannot be implemented in practical systems however, as it requires the removal of and measurement of the battery independent from the system that would otherwise be using it. To improve battery utilisation by preventing premature discarding of batteries due to poor measurement systems, and to ensure the safety and reliability of systems powered by batteries, new methods of characterising battery degradation must be developed.

Recent developments in the field of battery modelling have produced models that are able to accurately represent extremely low frequency battery impedance data. These models utilise the constant-phase-element to drastically reduce the component count of the model. These models suggest targeted small signal impedance measurements at extremely low frequencies can be interpreted to yield the internal capacity and loss

characteristics of the battery. If this model is correct, then observing a change in these values will directly relate to a change in the state-of-health of the battery.

To verify this method of determining state-of-health, an experiment is proposed. The systematic measurement and degradation of the state-of-health of a battery through cycling, and punctuated by measurements of the low frequency impedance. This will allow the discrimination of any link between the low frequency characteristics of a battery and the state-of-health. The proposed experiement will run for an extended period and will consist primarily of the repetitive cycling of the battery, and impedance measurements that take many days to complete. To aid in the accurate and timely completion of the experiment, specialised tools to automate repetitive and time consuming tasks must be implemented.

1.2 Project goals

The aims of this project are:

(1) To develop an automated battery impedance measurement system.

(2) To develop an automated battery aging and state-of-health measurement system.

(3) To investigate the relationship between battery state-of-health and the ultralow-frequency impedance characteristics of the battery when interpreted through the lens of the proposed constant-phase-element based model.

1.3 Thesis structure

Chapter 1 has explored the need for a new method of determining battery state-of-health. An ultralow-frequency impedance method for characterizing battery state-of-health based on the constant-phase-element model of the battery is proposed, and the desirability of automated tools for assisting with the necessary measurements is stated. The key goal is to determine the viability of the proposed state-of-health measurement method.

Chapter 2 provides a literature review of modern methods of state-of-health measurement as well as a background to the evolution of the constant-phase-element model and an introduction to the mathematics used to described the behaviour of a constant-phase-element.

Chapter 3 introduces the measurement systems created as part of this work. The purpose and major design considerations are described and a verification of performance is demonstrated.

Chapter 4 presents an experiment designed around the measurement systems from Chapter 3 in which a set of three similar batteries are aged and measured at regular intervals. Additional measurements of batteries with different chemistries and capacities are also presented.

Chapter 5 reflects on the measured data and the implications for the proposed measurement system and the model it is based on. Finally, Chapter 6 forms concluding remarks and identifies key areas of focus for future research.

Chapter 2

Modern Methods of Battery Modelling and Measurement

2.1 Background

2.1.1 Estimating Battery State of Health

State-of-health does not have a standardised definition or measurement system. Commonly it is defined as the measure of the capacity of a battery in its present state compared to the capacity when new. For this definition capacity is measured as the net charge moved when fully discharging the battery from fully charged. This is the definition used in this work. State-of-health is distinct from state-of-charge which describes the amount of charge currently stored in the battery as a fraction of the present capacity.

Measuring the state-of-health of a battery, as defined, requires cycling of the battery in order to measure the full discharge capacity. Few applications allow for such a measurement while remaining in service and it is not practical to take a car or pacemaker out of service for the time required to make the measurement. In practice this method of measuring state-of-health is typically only used in research scenarios as a comparison against an alternative method.

Current methods of estimating state-of-health can be separated at a high level into the direct assessment approach, adaptive approach, and data driven approach [2]. Adaptive and data-driven approaches suffer from high computational complexity and a poor suitability for implementation in a battery management system or another practical environment. Direct assessment methods are based on measurements of battery temperature,

voltage, and current. They are preferred for their low cost, computational complexity, and relative ease of implementation [3]. Accurately predicting state-of-health from these measurements is complicated by their non-linear dependence on additional factors such as state-of-charge, current magnitude, and temperature. All of these affect the terminal characteristics in a manner unique to the chemistry and construction of the battery alongside any effects from state-of-health [4]. Despite these challenges a variety of methods for estimating state-of-health have been developed, basing their estimates on battery impedance, power capability, internal resistance, differential voltage curves, charge acceptance, partial discharges, or accumulated estimates. While many of these methods enjoy limited success in lab conditions their overall performance is still unsatisfactory.

2.1.2 Measurement of Battery Impedance

Batteries have a varying impedance across a broad range of frequencies from below $10\,\mu$Hz to above 10 MHz [5]. Battery impedance is also sensitive to many compounding factors including temperature, state-of-charge, and state-of-health. This complicates the development of accurate battery models, as a generalised model fitted to such a wide frequency spectrum and additional factors is non-trivial. Attempts to model this sensitivity to a broad range of effects can result in models that are overly complex yet still ineffective [6], however more commonly the model is fit to only a subset of the above parameters. Scott and Hassan lament that this tends to include reducing the frequency range to no longer include frequencies below 1 mHz [7]. They point to the extremely long-lived transient response of the battery and the daily charging of common appliances which will result in a large stimulus at $11.6\,\mu$Hz to support the relevance of frequencies of this scale.

2.1.3 The Need for an Improved Battery Model

The equivalent circuit model of the battery that is used plays a vital role in determining how effectively it can represent the true behaviour of the battery. In some cases, low complexity models, such as the internal resistance model, are adequate at modelling simple battery behaviour but are wholly inadequate when more complex responses of a battery, such as changes in state-of-charge, state-of-health, and effects of temperature are being evaluated. More complex RC based models such as the Thevenin[8], Partnership for a New Generation of Vehicles (PNGV)[9], and Noshin[10] models attempt to model battery characteristics more accurately at the cost of increased component count and computational complexity. In many cases the increased component count serves

only to reduce the efficacy of fitting the model to a wide range of internal and external factors [11]. Furthermore, the analysis of the impulse response of a battery using Swinglers method by Hassan and Scott[12] reveals that no small number of exponential functions are capable of accurately representing the true behaviour of a battery, a damning rejection of these RC based models. In light of the inadequacy of RC based models alternative models are required.

2.1.4 Constant Phase Element Based Models

It has long been acknowledged that rechargeable batteries exhibit fractional-derivative characteristics; Randles showed this in 1947 by incorporating a Warburg element into his equivalent-circuit impedance model [13] Sabatier et al. have shown fractional derivative based models produce good results in practical applications [14] [15], and more recently a full fractional capacitor or 'constant phase element' has been identified in equivalent-circuit models [7], [16]. The order of the CPE (Constant Phase Element) can be derived from a low frequency measurement and is highly dependent on the battery technology being measured. In the case of lithium-ion batteries incorporating cobalt, the order is closer to or greater than 0.8, compared with the value of 0.5 measured for small lithium-iron-phosphate and various lead-acid batteries [7], [12], [17]. A popular topology used to model the behaviour of an electrode-electrolyte interface [18] that has also gained favour in the battery modelling world is the Randles model with two fractional elements, one representing the non-ideal capacitive behaviour of an electrical double layer and the other representing the faradaic impedance of the internal chemistry. Brug criticises that failing to accurately acknowledge the mechanisms behind these elements causes many analyses of battery impedance data to be internally inconsistent when fitting values to the model [19] and argues instead for a model with a single constant-phase-element which may be modelled as a distributed system. Scott and Hassan also argue for a single constant-phase-element model however their modification to the distributed system greatly improves the fit to measured data [7].

2.1.5 Constant Phase Element Theory

While these concepts were first introduced to model the behaviours of electrodes in salt solution when subjected to alternating currents, for which a detailed history may be found in [18], they have since been found useful for the modelling of medical electrodes inside the human body [Single and Scott 2018] and are seen increasingly in models of batteries. It is assumed that the reader is familiar with the concept of the constant-phase-element, however a brief introduction is provided here. For those interested in

further exploring the mathematics of the constant-phase-element starting with the useful summary by Kartci [20] is recommended.

Introduced by Warburg in 1899 [21] what is now known as the Warburg element is a device similar to a capacitor, however differing in that the terminal current and voltage are related by a one-half power derivative. It is simple to examine this behaviour as an impedance whose magnitude and phase are as follows

$$Z_{\text{warburg}} = \frac{1}{jw^{0.5}C} \tag{2.1}$$

$$\theta_{\text{warburg}} = \frac{\pi}{4} \tag{2.2}$$

The concept introduced by Warburg was further expanded on by Fricke [22] to the generic case now known as a constant-phase-element (CPE) with the properties

$$Z_{\text{CPE}} = \frac{1}{jw^{\alpha}C_F} \tag{2.3}$$

$$\theta_{\text{CPE}} = \frac{\pi\alpha}{2} \tag{2.4}$$

The constant-phase-element can therefore be described fully by the two variables describing the capacity, C_F, and the order, α, where $0 < \alpha < 1$. When compared to the capacitor equations it becomes clear that when $\alpha = 1$ the constant-phase-element behaves as a capacitor and as α decreases towards 0 the constant-phase-element tends towards behaving as a resistor. The behaviour as a sort of 'lossy capacitor' is the basis of the belief that the value of α will decrease with degrading state-of-health of the battery as a greater amount of energy is lost in worn batteries.

2.1.6 Application of Constant Phase Element Based Models

The models suggested by Brug[19], and Scott and Hassan[7] have a remarkably low number of variables and yet are able to fit real data at low frequency with notable accuracy. Observing this it is crucial to realise that there is only one element present that is capable of storing energy in the system; the constant-phase-element described by C_F and α. From this basis it is proposed that any change in the state-of-health of the system must be reflected in a change in either or both of the parameters C_F and α.

Measurement of C_F and α can be made at appropriately low frequencies however the extremely low frequency nature of these measurements requires an automated system to be able to reliably and repeatably make measurements. The low frequency nature of these measurements arises from two properties of the battery system, the large capacity, and the fractional nature of the batteries. The combination of these properties results in an impedance that is dominated by the constant-phase-element at frequencies typically well below 100 μHz.

2.2 Literature review

2.2.1 Types of Battery Models

Battery models are used to predict the behaviour of a circuit or system to ensure safe operation and optimise performance. Terminal characterisics of batteries vary due to a wide range of factors including state-of-charge, state-of-health, and temperature. Due to this wide range of sensitivity a variety of models have been developed, each aiming to satisfy their own requirements for complexity and accuracy. A thorough analysis of these models and their use cases can be found in [11]. A brief summary of the models will be provided here.

Simple models such as the ideal model, internal resistance model, and RC models all provide minimal component counts and low computational complexity. This leaves their performance lacking when modelling some of the more complex responses of a battery such as changes in state-of-charge, state-of-health, and temperature. Despite these flaws they are still adequate for use in systems where those considerations are low priorities or are decoupled from the rest of the system through use of a regulator.

More complex RC based models such as the Thevenin[8], Partnership for a New Generation of Vehicles (PNGV)[9], and Noshin[10] models attempt to overcome these limitations at the cost of increased complexity. These models attempt to characterise the batteries to allow the estimation of state-of-charge and transient responses through measurements of terminal characteristics. Many of these models also take into account the large effect temperature has on battery terminal characteristics to provide a more general model. A common way of adjusting the model parameters to fit real data is through use of a Kalman filter or other computational assistance to estimate appropriate values based on previous measurements. Some methods simply resort to a lookup table to reduce processing requirements, this requires extensive measurements to fit a model and limits the accuracy of the results due to individual variations in battery characteristics. These models are used to simulate the transient response and terminal characteristics

of a battery in EVs and portable power systems. This allows optimisation of the performance of the system for changes due to state-of-charge and temperature variations.

Runtime models separate the effects of identified battery features, such as self discharge, transient response, and varying impedance and terminal voltage with state-of-charge, into subcircuits. These subcircuits are linked together to provide a complete system. This allows the piecewise construction of a battery model by investigating each feature and their relationships independently. These circuits are favoured by some for their 'intuitive nature'[6] however no current models provide for variations due to temperature or state-of-health. The additional complexity of integrating subcircuits that take these variables, and their effects on other subcircuits, into account would likely invalidate this view due to how they interact with all other parts of the system. Runtime models are used for simulation of battery response under dynamic loads with a primary focus on application to EVs.

Impedance models focus on the frequency response of the system. A technique called electrochemical impedance spectroscopy (EIS) is used to measure impedance data over a range of frequencies that is fit to a circuit model[23]. A small-signal alternating current or voltage stimulus is applied to the battery and swept across the desired range of frequencies. It is important for the measurement to stay within the limit for small signal measurements that is dependent on the capacity, geometry, and storage technology of the battery. This is due to the impedance being affected by the charge density at the electrode-electrolyte interface that will be thrown out of equilibrium by large currents in either direction. The impedance is measured and the model fit to this response. This allows the creation of a battery model that precisely fits features of the battery for use with simulations and performance optimisation. A common topology is the Randles circuit featuring a constant-phase-element that is used as the basis for a large amount of models using this technique.

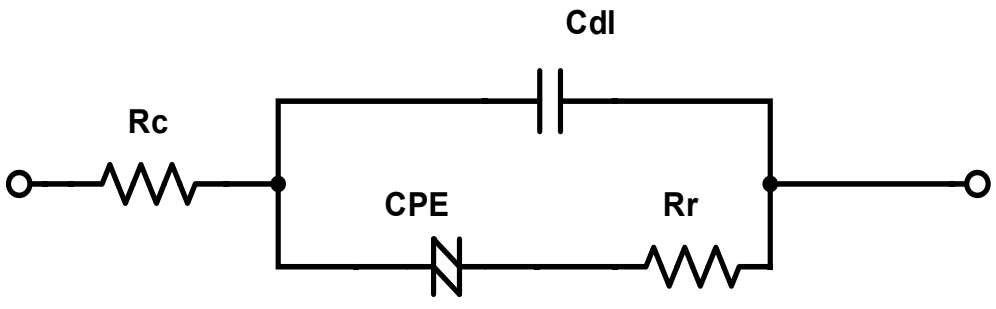


FIGURE 2.1: The circuit topology proposed by Randles as a model of an electrode

2.2.2 Modelling using a Constant Phase Element

The impedance of a battery varies across a broad range of frequencies starting from below 10 μHz to over 10 MHz [5]. Battery impedance is also sensistive to factors such as temperature, state-of-charge, and state-of-health. This complicates the development of accurate battery models as a generalised model fitted to a broad frequency spectrum with multiple compounding factors is a non-trivial endeavour. Attempts to model this sensitivity to a broad range of effects can result in models that are overly complex yet still ineffective as noted by [6]. Commonly the model is fit only to a subset of the above parameters in order to reduce the complexity of the model. Scott and Hassan laments that this tends to include reducing the frequency range to no longer include frequencies below 1mHz [7]. They point to the daily charging of common appliances which will result in a large stimulus at 11.6 μHz to support the relevance of these frequencies. The issue of this collective overlooking of battery characteristics on these timescales is highlighted further by [12]. Their analysis of the open circuit voltage of a battery after load shows there is non-exponential response as the battery recovers for many thousands of seconds after the load is removed, this response is not present in common RC battery models. This long lived transient is a feature of systems containing a constant-phase-element.

Some works do not use a constant-phase-element to model the long lived transient response of batteries, instead using a series of RC components to model the effects. Comparison of the circuits of this form [8] to work by Morrison [24] shows that their proposed model closely resembles the cannonical forms of a constant-phase-element presented by Morrison. Further comparison plainly shows that the general case of the extended Thevenin model becomes a constant-phase-element with minimal effort.

Morrison [24] presents models of a constant-phase-element created from passive RC networks and shows their equivalence. Seshadri and Scott [25] provides necessary corrections to some of the calculations and a method of generating a constant-phase-element for spice simulation. Morrison states that while the mathematical discussion of the models deals with an infinite number of RC element pairs, in practice as few as three RC element pairs can be used to provide an approximate model that covers a two-decade frequency range if "large deviations" from the expected response are permitted.

When modelling batteries it is often necessary to consider the underlying chemical processes. Incorporating often overlooked dynamics Li et al. [26] builds a model designed to model the effects of state-of-charge and battery aging. The model considers the effects state-of-charge on open circuit voltage through the lens of chemical concentration, and the effects of cycling on the growth of the Solid-Electrolyte Interphase (SEI) layer,

focusing on the SEI layer on the anode. The model also the separately models the reactions and reaction rates at the anode and cathode in a manner reminicent of Randles [13] measurements on battery half cells. These factors are considered alongside more common factors including the double layer effects and diffusion dynamics around the electrodes. This approach results in a model with component groups that have a clear relationship with the underlying chemistry, a desirable outcome for investigating how both external and internal factors will affect the terminal characteristics of the battery. Constant-phase-elements are used to model the dispersed capacitance of the electrode-electrolyte interface at the cathode and to model both the SEI film impedance and double layer capacitance at the anode. The model is fit to the data using the Hybrid Multiple Particle Swarm Optimization method. Use of stochastic optimisation methods for model fitting tends to muddy the waters when interpreting the meaning of the fit values and introduces the issue of over-fitting. Due to how the model has been created Li et al. are confident that the resultant values continue to acccurately reflect the underlying processes. Their model is able to fit well to measured battery impedance data between 400 mHz and 4 kHz however the fit becomes poor at lower frequencies with the measurements only going to 25 mHz. This reinforces the importance both of reflecting the process in the model and of the urging from Scott and Hassan to extend the range of models into lower frequencies.

2.2.3 Modelling State of Health

State-of-health is one parameter used to determine state-of-function or remaining-useful-life for a battery, the point where a battery is no longer able to carry out its designed function. Other factors that contribute to these calculations are terminal resistance, maxiumum current draw, and peak output power. Once the state-of-health of a battery has fallen below a limit, typically 0.8, the battery is treated as failed and must be replaced. The lower limit for state-of-health depends on the application, with critical functions such as safety and security systems having a lower tolerance for battery degradation. It is useful to predict when a battery will fail to allow time to organise replacement and to prevent unexpected failures.

It is therefore necessary to make regular measurements of battery state-of-health. Directly measuring state-of-health requires measuring the full discharge capacity of the battery. This requires fully charging, then discharging, and finally recharging the battery. Multiple cycles may be desired to ensure the accuracy of the measurement. This not only removes the battery from service but is also time consuming and wastes the stored energy. In practise this method of measuring state-of-health is typically used only

when the battery will be fully cycled during use and the measurement can be made without removing the battery from service or in research scenarios as a comparison against an alternative method. Measurement of state-of-health by measuring total capacity of a battery while in service reduces the accuracy of the method due to the varying load currents, however, use of filters such as the Kalman filter can improve the result. Alternatively state-of-health can be measured indirectly through its effect on the terminal characteristics of the battery. Accurately predicting state-of-health from measurements of the terminal characteristics is not straightforward due to interferring factors such as state-of-charge and temperature affecting the terminal characteristics alongside any effects from state-of-health [4].

Berecibar et al. [27] and Han et al. [28] both used the differential voltage method to identify artifacts in the charge discharge curve of lithium batteries that are sensitive to state-of-health. This method measures terminal voltage and current during charging and discharging cycles and then calculates $\Delta V \backslash \Delta Q$ at varying state-of-charge. This method works well on battery technologies that have distinctive artifacts such at the LiFePO4 used by Berecibar et al. however some of the batteries characterised by Han et al. did not show any clear artifacts that are sensitive to state-of-health. This heavily limits the method as it is only applicable to certain battery technologies.

Wang et al. [23] attemped to model state-of-health by fitting an impedance model while also considering the effects of temperature and state-of-charge. With their thorough measurement scheme they map the impedance of the battery over a temperature range from 5°C to 45°C and state-of-charge from 10% to 90% while varying state-of-health from 1 to 0.8, the typical life cycle of a battery. Ending up with 200 unique impedance sweeps over the frequency range 10 mHz to 1 kHz. However even with what appears to be a deluge of data their model was vastly inadequate, only managing an error below 10% when estimating state-of-health in a small subset of the region they attempted to map. This highlights the difficulty inherent in attempts to characterise battery state-of-health, especically when expanding the model to include other relevant parameters such as state-of-charge and temperature.

A thorough investigation into the compounding nature of the effects of state-of-charge, temperature, current rate, previous history, and state-of-health on the impedance of lithium ion batteries is provided by Waag et al. [4]. It is clear that attempts to model state of health in a practical application cannot afford to negelect any of these parameters due to their wide ranging interaction with battery impedance. They do not draw any conclusions on how to model these parameters, instead they conclude that any attempt to include them in a model, will need to be specialised to a unique application and cell type.

A review of the literature has demonstrated that while there has been wide ranging and extended effort directed towards development of methods to measure battery state-of-health there has been no significant success in the field. The most promising models of battery behaviour are based on the constant-phase-element and yet all have still failed to investigate frequencies below 100 μHz where it has been identified that measurements of the properties of the constant-phase-element become significant. Attempts to model additional contributing factors such as temperature lead to overly complicated models and poor results. Thus, through the lens of the constant-phase-element based model, it is the focus of this work to investigate battery state of health and the so far uncharacterised interaction with impedance at the neglected extremely low frequencies.

Chapter 3

Tools

Reliable and repeatable measurements are essential to both the sucess of this work and the continued success of the research group with whom it was undertaken. Measurements of battery impedance at sub-hertz frequencies are both time consuming and sensitive to environmental changes. Similarly, the safe aging of a battery through repeated cycles cannot be done quickly. This makes these tasks prime candidates for automation. Tools for measurement of impedances at the frequencies of interest, primarily below 1mHz, exist on the market. A common device used in the literature is the Solartron 1260A however recent works have cast doubt on the reliability of the Solartron 1260A [29], [30], [17]. Instead of purchasing an off the shelf solution with dubious credibility the available lab equipment Table 3.1 is sufficient to make the desired measurements.

TABLE 3.1: Available Equipment

Device	**Output Power**	**Maximum Current**	**Voltage Resolution**	**Current Resolution**	**Suitability Aging**	**Suitability Impedance**
Hameg HM8143	60 W	2 A	10 mV	1 mA	Good	Poor
E5270B/ E5281B	2 W	100 mA	25 μV	50 fA	Poor	Excellent
66332	100 W	5 A	5 mV	1.32 mA	Good	Good
Keithley 2460A	100 W	7 A	100 nV	50 pA	Excellent	Excellent

3.1 A Unified Interface for Measurement Automation

The automation of measuring battery impedance, measuring state of health, and battery aging can all be reduced to the challenge of correctly setting the output of the intended

hardware device and recording the voltage, current, and time to allow analysis of the battery's behaviour. To achieve this, a unified interface was created to translate generic commands into device specific commands for basic tasks such as setting the output and taking a measurement. This allowed the creation of a single generic program for any desired operation that will work on all devices supported by the unified interface. Automated control of the devices requires an additional component however, a communication protocol. Many devices support GPIB, however this is not universal. Another common communication interface is a simple serial port on the device. Regardless of the underlying protocol the communications interface needs to support two basic functions, read and write. This core functionality was also abstracted to a unified interface, allowing the creation of a fully modular system capable of performing predefined operations on any combination of supported devices and communications protocols through a standard interface. This system allowed the creation of two stand-alone programs to both aid in the core experiments of this project and as tools for use by the research group in further research. These programs are an automated battery impedance measurement system, and an automated battery aging and state-of-health measurement system. Both are discussed in detail in Section 3.2 and Section 3.3.

3.1.1 Major Design Considerations

The low frequencies of interest result in time consuming measurements, tying up valuable lab equipment for extended periods of time. This was the driving influence behind the design decision to create a modular system that would allow control of any suitable lab equipment, greatly increasing the flexibility of the measurement system. New hardware can easily be integrated into the system and selection of the targeted device is done when starting the interface. The communications standard GPIB is widely implemented as a method of sending commands to, and receiving data from, instruments. The use of this standard, while common, is not guaranteed, and devices may implement alternative communications standards along with or instead of GPIB. The program allows substitution of the GPIB board number and instrument address with a serial port and baud rate, allowing the use of either GPIB or alternative serial communication ports on instruments.

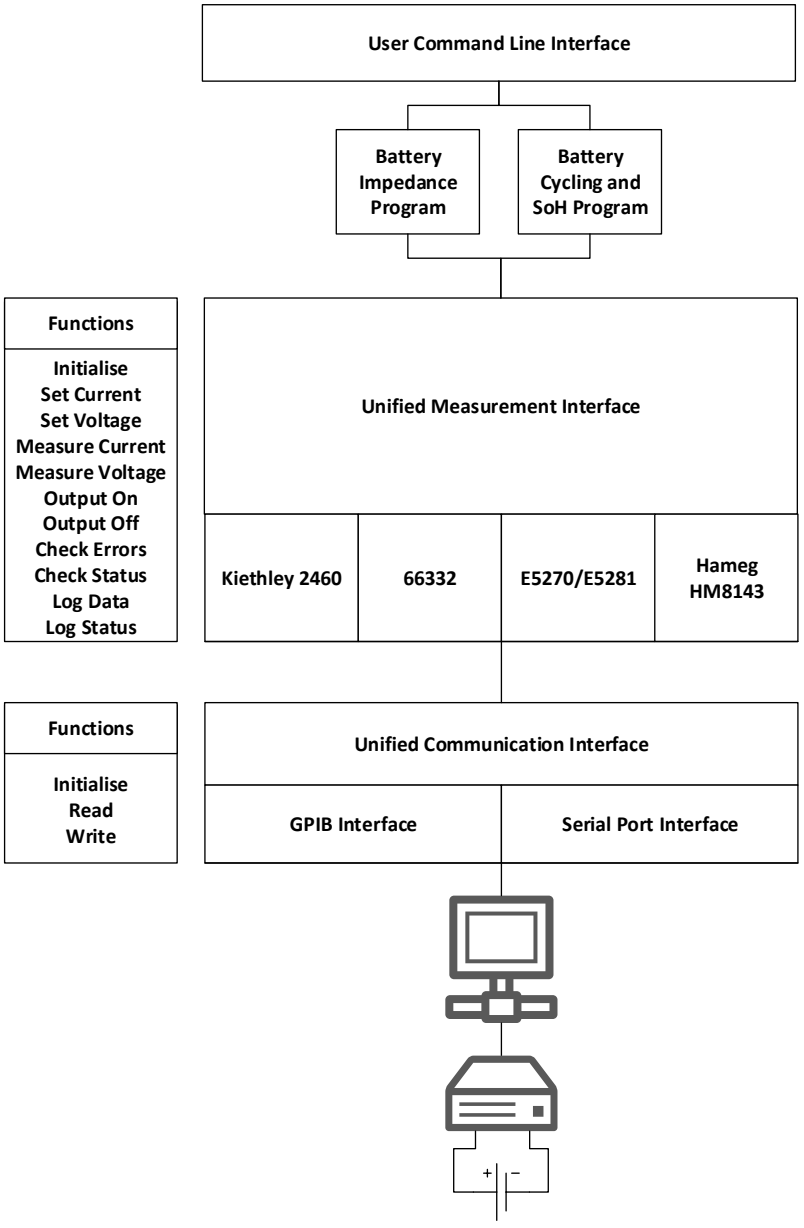


FIGURE 3.1: Simplified achitecture of the unified measurement interface

3.2 Software for Battery Impedance Measurement

Measurement of battery impedance over a wide range of frequencies is crucial to ongoing efforts to fit accurate models. The frequencies of interest are in the range of 10 μHz to 10 MHz as described by Macdonald [5]. Modern instruments used to measure impedance at high frequencies exist and are readily available, however the extremely low frequencies are not supported by modern measurement instruments. These low frequencies, especially those less than 100 μHz, are the primary interest of this work and require a custom solution to provide reliable and repeatable measurements. Due to the long running nature of the desired measurements, the system must run safely without intervention for extended periods of time after being started. With the creation of the unified interface for controlling the available hardware a generic algorithm for making the desired measurements could be devised. This was written as a function integrated into the unified interface and used as a core component of the battery impedance measurement system.

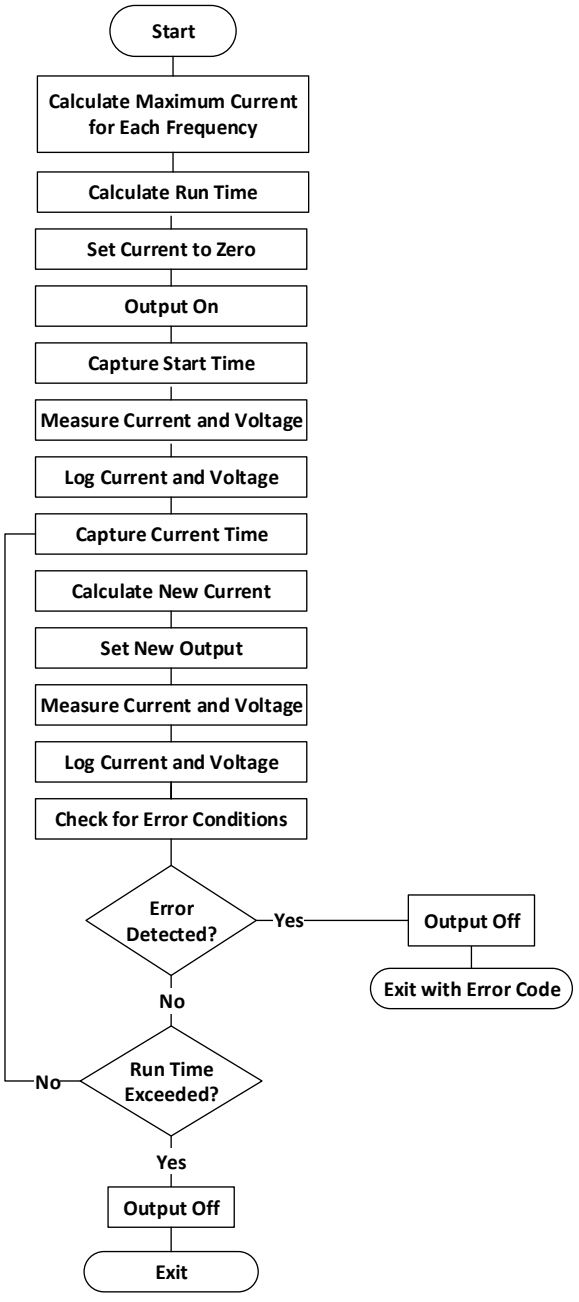


FIGURE 3.2: Simplified architecture of the battery impedance algorithm

3.2.1 Major Design Considerations

A command line interface is used to run the battery impedance measurement system. This allows the creation of script files that automatically run sequences of measurements alongside the battery aging program also created for this work. This allows for the efficient use of the limited time available. The program displays real-time progress in the command window, and creates a log file containing details of major events, and data files with time-stamped voltage and current measurements for further processing into impedance data. The raw data is not processed by this program into impedance data. This allows inspection of the data quality and reduces the complexity of the individual program and improves maintainability.

The frequency and number of periods of the measurement are given to the program through the command line interface. A list of up to 16 frequencies can be entered, negative frequencies can be entered to introduce a 180° phase shift. An optional parameter forces measurements to be taken one frequency at a time, performed in the order given. This allows sweeps of associated measurements to be done with a single run of the program, ensuring all the measurements share the same parameters. Without the optional parameter a multi-tone measurement is performed whereby all given frequencies are superimposed. This is to allow every frequency to be measured simultaneously, greatly reducing the time taken to perform a sweep of measurements. In the multi-tone case, the time taken is that of the requested number of periods of the lowest frequency given. In order to ensure the safe and reliable nature of the measurements, when multi-tone measurements are requested the amplitude of each frequency is scaled down by the number of contributing tones from the size it would normally be if the same sweep were to be single tone. The ensures compliance with the maximum current limit.

The probing signal is designed to be a sinusoidal current waveform. While driving a sinusoidal current will result in no net charge being moved after a complete period of the waveform, the peak deviation from the initial state-of-charge occurs after one half period. Low frequencies can move significant amounts of charge during a half period. This is a concern as the relationship between battery terminal characteristics and state-of-charge is non-linear. For example, a 50mA peak signal at 10 μHz applied to a 2000mAh battery will shift the SoC by 22% from the initial state during the measurement. This would both distort the measurement and risk overcharging the battery if the initial state-of-charge was too high.
To manage this risk two parameters are used, one to set the maximum current limit and one to set the maximum ΔQ, to ensure the integrity and safety of the measurement. Additionally, maximum and minimum terminal voltage limits are enforced to further ensure safe and reliable measurements.

3.2.2 Verification of Performance

It is of utmost importance that the impedance measurement system produces repeatable and reliable results. Repeated measurements were made to confirm the repeatability of the system. A series of measurements were made on the Keithley 2460 showing the variation of identical measurements to be near non-existant as shown in Figure 3.3. Furthermore, from Figure 3.4 there is no noticeable effect on the impedance due to changing the order in which the measurements are taken. Figure 3.5 shows that the effect of increasing the number of cycles measured has diminishing returns for larger numbers of cycles. Finally the same measurement was made across multiple devices with each providing high reliability at low frequencies as seen in Figure 3.6.

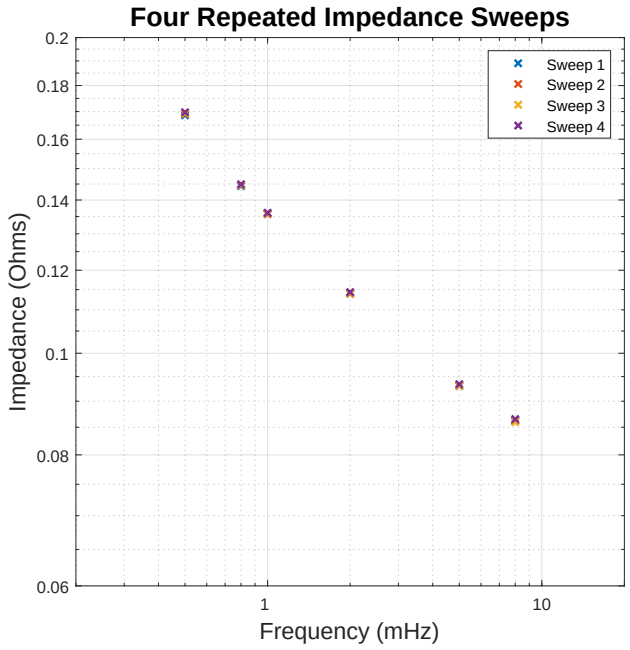


FIGURE 3.3: The same measurments repeated 4 times show no significant deviation and cannot be distinguished from each other

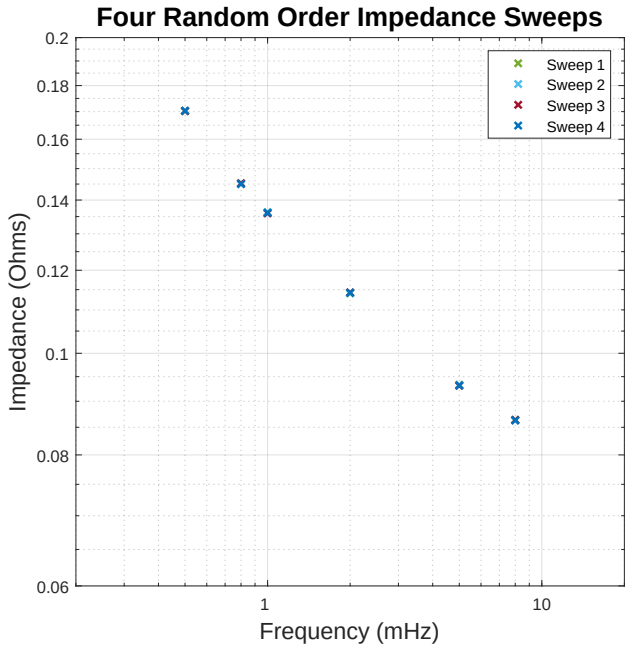


FIGURE 3.4: The order of measurments does not affect the results

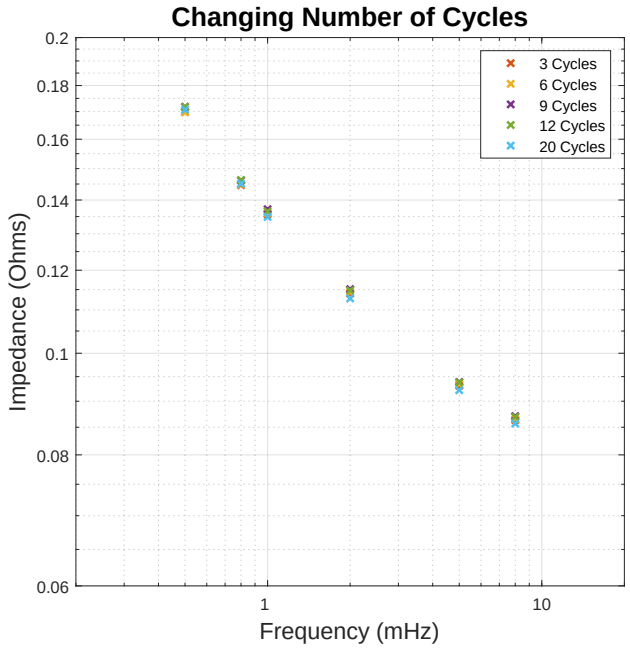


FIGURE 3.5: The number of cycles used does not greatly affect the measured results, measurements with more cycles may be affected by external factors such as the temperature change at night. This is likely why the 20 Cycle measurement does not follow the trend of the other measurements.

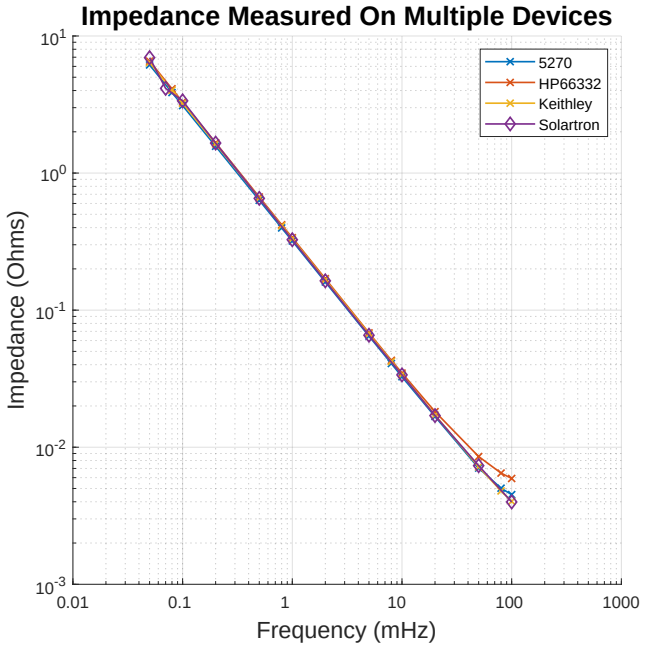


FIGURE 3.6: Measurements across devices are consistent. The Solartron is starting to deviate from the expected behaviour at the lowest frequencies

3.3 Software for Battery Cycling and State of Health Measurement

Accelerating the rate of aging of batteries through cycles of charging and discharging will allow the simultaneous measurement and degradation of battery state-of-health. Aging the batteries in a timely manner requires an automated system to safely charge and discharge the batteries. The system created implements the constant-current-constant-voltage (CCCV) battery charging system and allows the user to set the upper and lower voltage limits, constant current limit, constant voltage minimum current limit, and constant voltage time limit. The charge moved during each CCCV charge and discharge cycle is measured and the battery can be discharged to a user defined percentage of this value to ensure the battery is left at a consistent state-of-charge after cycling. Additionally the system can be set to continue measuring the battery voltage with zero current draw for a user defined period after all other actions are performed to allow obervation of the characteristic relaxation curve of the battery.

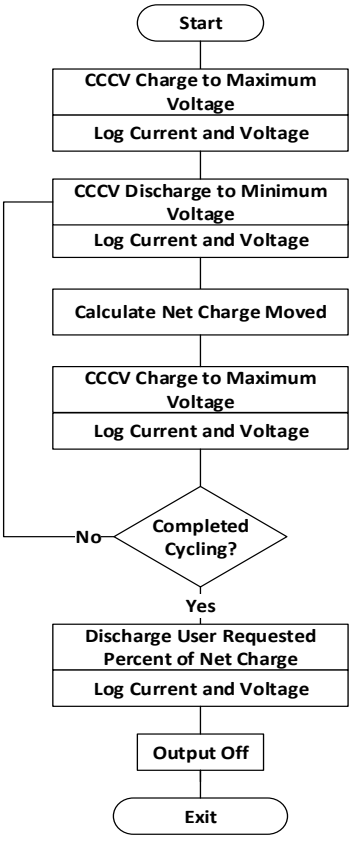


FIGURE 3.7: Simplified architecture of the battery cycling and state of health measurment algorithm

3.3.1 Major Design Considerations

Measurements of the capacity of each cycle and the ability to return the battery to a known state of charge were important requirements to judge the state of health of the battery and ensure consistency across each impedance measurement. This system would also need to safely run for long periods of time without intervention and not cause undue damage to the battery.

To achieve this the user must give both a maximum and minimum voltage limit and a maximum current limit. The system will use these as the boundaries for cycling the battery. These limits, given by the user, are assumed to be correct and safe for the battery.

While cycling the battery between voltage limits it is useful to be able to sit at these limits for a set duration during which the terminal voltage will remain constant while the current magnitude decays. Two limits to this current decay are specified by the user; a lower current limit, the lowest current which the system will decay to before continuing with the next stage of cycling, and the dwell time, the maximum amount of time the system will spend allowing the current to decay towards the lower current limit before continuing with the next stage of cycling. With this set of limits the user can cycle the battery using constant current constant voltage (CCCV) cycling.

The number of cycles is specified by the user, the system will complete this number of cycles according to the scheme outlined in Figure 3.7. After the final charge is completed the system will then discharge the battery to a given percentage of 'full' defined as the amount of charge moved when last discharging the battery. A relaxation time can be given which will set the system to zero current and continue making measurements of the battery. This allows the observation of the equilibriation of the battery.

3.3.2 Verification of Performance

The safety and reliability of the battery cycling and state-of-health measurement system was of utmost importance. Testing of the system to ensure compliance with the set limits and safe behaviour in fault conditions gave the confidence to proceed with further automated testing. Initial testing was done with a large capacitor and progressed to a NiMH battery. Only once satisfied that the system was both safe and reliable were tests using lithium batteries started.

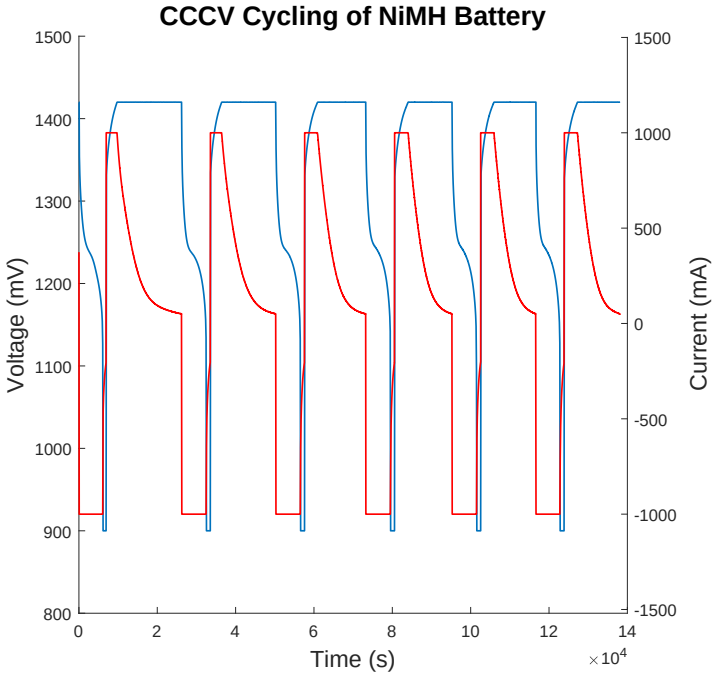


FIGURE 3.8: Cycling NiMH battery using the battery cycling program to prove functionality before testing on lithium batteries

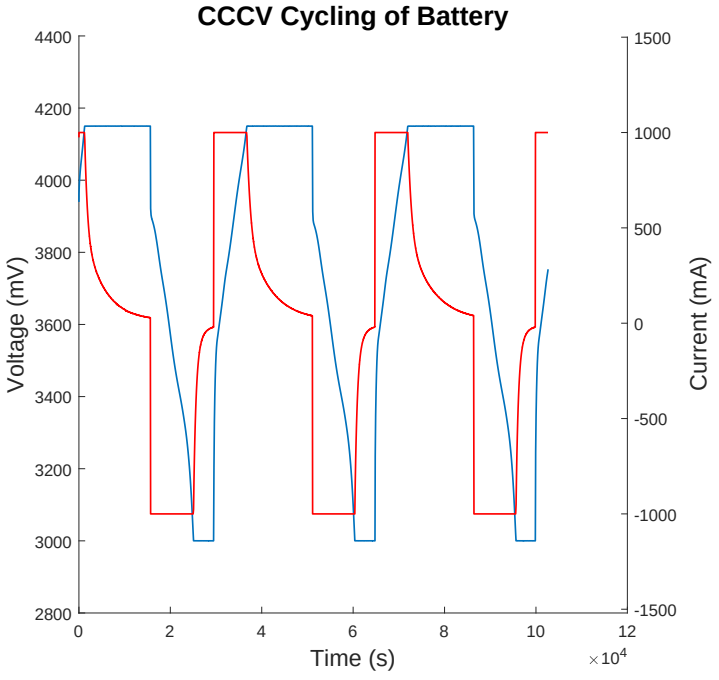


FIGURE 3.9: Cycling lithium-NMC battery using the battery cycling program

3.4 Single Board Battery Impedance Measurement

The system created for automated impedance measurement works extremely well but requires the dedication of both a computer and an expensive piece of laboratory equipment for the duration of the measurement. This limits the number batteries that can be tested at any time to the amount of free lab equipment that can be reserved for days or possibly weeks in a row. To ease the demand on equipment and to allow portable measurements, work on a self-contained impedance measurement system has been progressing. Initial work by Chris Dunn as a final year engineering project [31] led to promising results for the creation of a cheap, independent system, able to make impedance measurements at predetermined frequencies. This concept has been further refined as part of the final year engineering project of Ben Finer, still in progress at the time of submission. The contributions of Gordan Wildschut to the PCB layout and from Professor Jonathan Scott to both circuit design and the algorithm for impedance calculation are gratefully recognised as crucial components of the final system. While still requiring further refinement, initial tests have shown the concept is possible, and the proof of concept for the creation of a low cost, independent system for measuring low frequency battery impedance has been a resounding success. Circuit schematic and code listing are attached in Appendix B

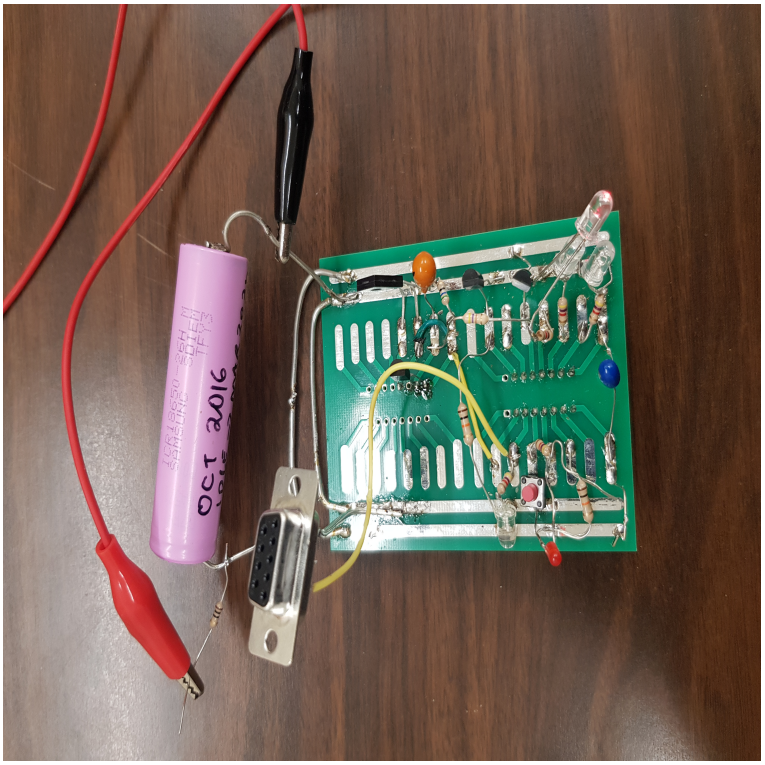


FIGURE 3.10: Single board battery impedance measurement prototype measuring a lithium-NMC 18650 battery

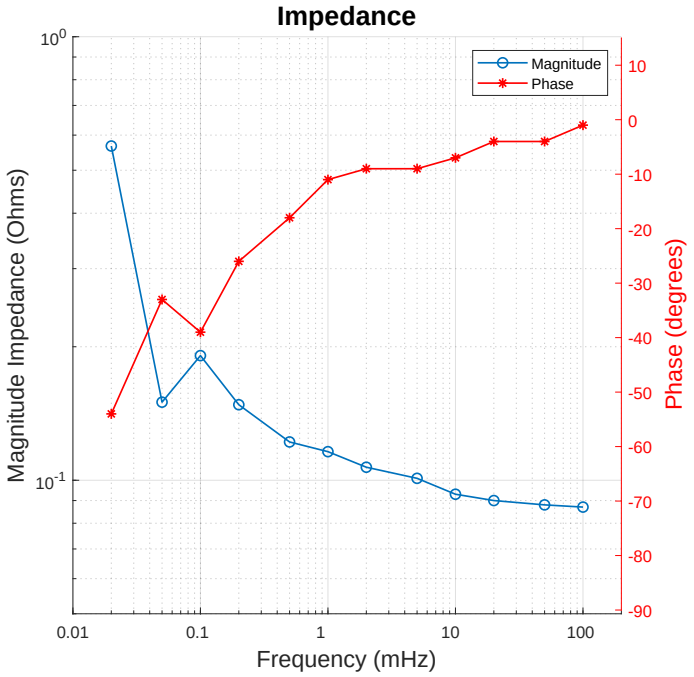


FIGURE 3.11: Impedance of lithium-NMC battery measured by the single board impedance measurement system, while not as clean as the results from the measurement program from Section 3.2 the results show sufficient similarities in shape and magnitude to justify futher investment into refining the prototype.

3.5 Single Board Battery Cycling

Automated battery cycling is time consuming. While the system created for this purpose is able to both cycle a battery and measure the battery capacity, the information gathered from a whole 100 cycles is not a large amount more valuable than the information that can be gained from analyzing a handful of cycles from the start and the end. This means that valuable lab equipment that is committed to cycling a battery any large number of times is being poorly utilised. Further, the battery cycling system is only able to utilise 2 or 4 quadrant supplies, further straining the list of available equipment. To utilise readily available single quadrant power supplies for battery cycling a solution has been created. The single board battery cycler measures and controls up to two batteries, selectively connecting them to either a connected load or power supply to charge or discharge the batteries. The user can easily select the number of cycles, the upper and lower voltage limits, the ratio of the input voltage divider, and the number of connected batteries. Appropriately setting up the load and power supply allows the user to easily cycle multiple batteries in a safe and manageable manner without expensive lab equipment. Circuit schematic and code listing are attached, Appendix C

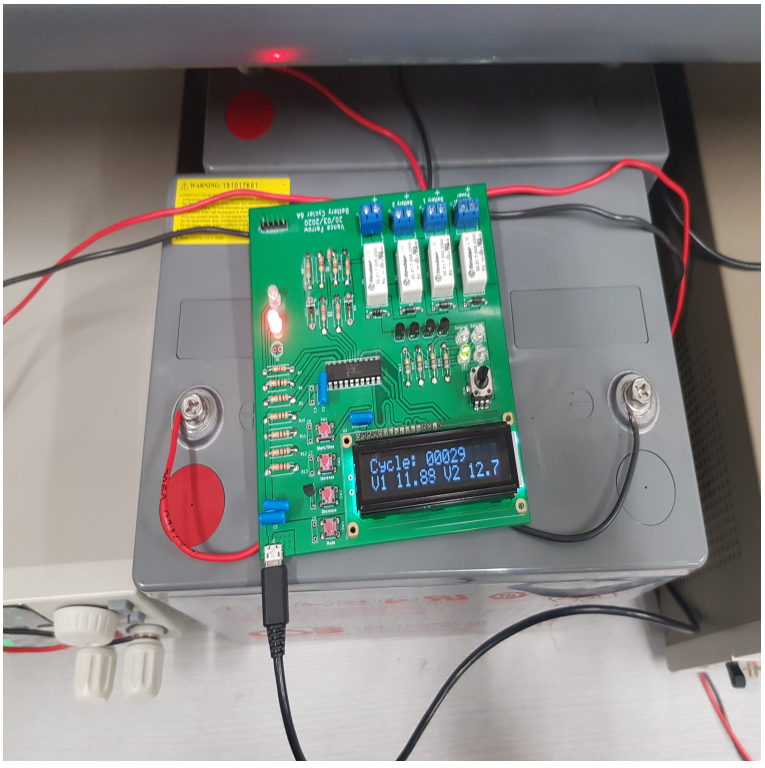


FIGURE 3.12: The first version of the battery cycling board running on a pair of lead-acid batteries

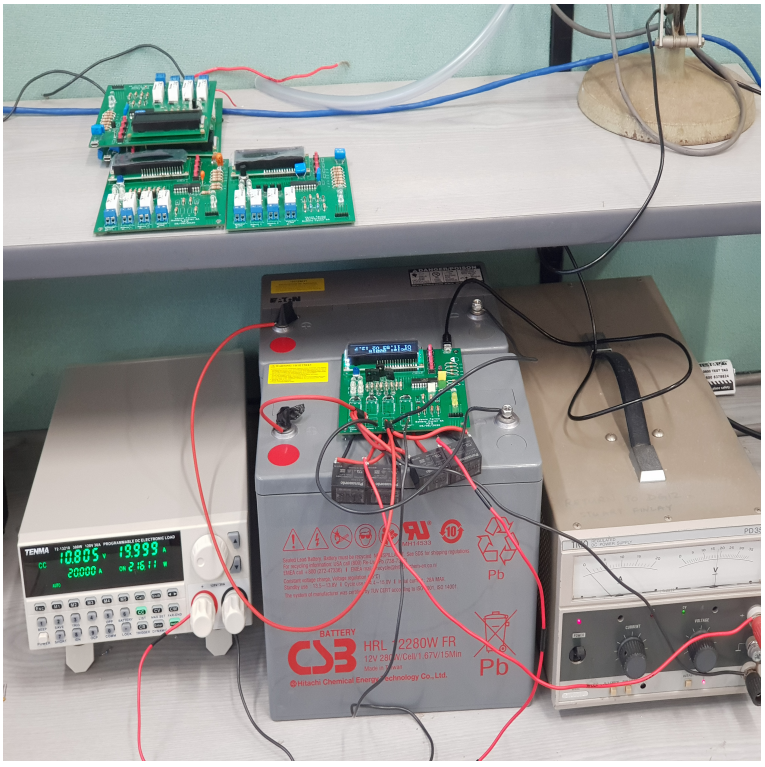


FIGURE 3.13: A later version of the cycler board modified to use 20A relays to rapidly cycle large lead-acid batteries.

Chapter 4

Experiments and Measurements

4.1 Automated Aging and Measurement of Battery Cells

Now armed with the appropriate tools from Chapter 3 an experiment to investigate the effect of battery aging on low frequency impedance characteristics was created. This section details the experimental setup and resulting measurements.

4.1.1 Experimental Setup

Three identical 2400 mAh lithium-NMC 18650 batteries were purchased. Two were kept in an air conditioned room at 23°C and the last was kept in an incubator at 40°C. The temperature of the battery was raised to 40°C to accelerate the aging process as suggested by the literature [32] [33].

The battery at 40°C was connected to a 66332 two quadrant power supply. This allows both cycling and measuring the impedance of the battery with a single device through use of the tools created in Chapter 3. One of the batteries at 23°C was connected to the Keithley 2460. This device is a capable of both cycling the battery and measuring the impedance with great accuracy. The final battery at 23°C was connected to both the Hameg 8143 and the E5270/E5281. This was arranged so the Hameg could cycle the battery and the 5270 could measure the low frequency impedances.

Informed by initial testing and restricted by time limitations it was decided the impedance of each battery would be measured a single tone at a time at the following frequencies. Each frequency measurement would consist of 6 periods of the single frequency. $500, 200, 100, 80, 50, 20, 10, 8, 5, 2, 1, 0.8, 0.5, 0.2, 0.1, 0.08, 0.05,$ and 0.02 mHz.

A maximum current of 100 mA and a maximum charge displacement of 42 mAh were enforced to ensure that neither the current nor the charge displacement were high enough to interfere with accurate measurements. After an initial measurement sweep the automated battery cycling system was used to age the batteries. A batch of 50 cycles between 4.2 V and 3.0 V at 1 A were performed. These limits were chosen to ensure the safety of the experiment by not subjecting the battery to abnormally high currents or voltages. After the 50 cycles were completed the battery was discharged to 60% capacity and rested for an hour to ensure a consistent SoC after each batch of cycles. Following the batch of cycling another impedance sweep was measured to detect changes in the low frequency impedance. This cycle was to be repeated until the battery had been cycled 1000 times. The system of automated cycling and measuring impedance was written into batch scripts and executed for each battery and its corresponding set of hardware.

After 150 cycles the state-of-health of the battery raised to 40°C was inspected. Noticing that the battery had not degraded significantly, and believing that this battery would age the fastest of all the batteries, the testing scheme was altered to produce results in the limited remaining time. A large set of 150 cycles was performed on all the batteries, followed by an impedance sweep. Following this, sets of 100 cycles were made between impedance sweeps.

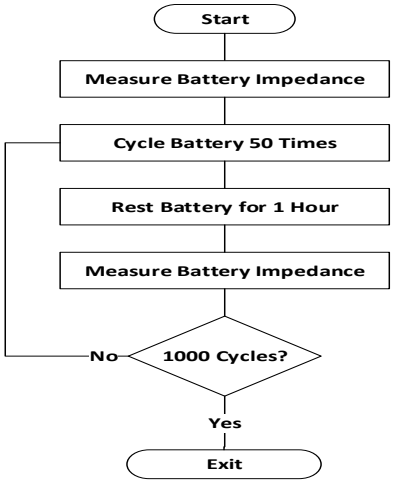


FIGURE 4.1: Flow diagram of initial testing schema

4.1.2 Major Design Decisions

A selection of three identical 2400 mAh lithium-NMC 18650 batteries was used. This battery technology was chosen because of its high popularity with applications ranging

from cars to cellphones, its permeation of the rechargeable battery market and high availability make it an excellent choice. Three batteries were used to provide sufficient data to allow strong conclusions to be drawn.

It is well documented that changes in temperature will cause changes in battery impedance. Two batteries were kept in an air-conditioned room at a constant 23°C and the last kept in an incubator at 40°C. The temperature was raised with the intention of accelerating the aging process as there is evidence that raised temperatures have a negative effect on battery lifespan [32] [33].

Preliminary tests showed promising results were obtainable without needing to measure below 20 μHz. Frequencies were selected to optimise both the quality of the results and to reduce the time taken for each measurement. At the time the experiment was started preliminary work with multiple simultaneous measurements was underway, and the reliability of the system was not confirmed. For this reason, the measurements in this experiment were taken a single frequency at a time. While this greatly increased the time taken to complete a set of low frequency impedance measurements, it was the most tested and reliable method available at the time.

Limiting the maximum current and charge displacement during an impedance measurement was crucial to ensuring the reliability of the measured values. A current limit set too high would no longer be a small signal measurement and may be affected by non-linear behaviour, this will mostly affect higher frequencies. Similarly, a charge displacement set too high will vary the state-of-charge of the battery during a measurement sufficiently to change the impedance and pollute the measurement, this is mostly an issue faced by lower frequency measurements.

4.1.3 Safety Considerations

Safety has always been a high priority and the potential danger posed by lithium-NMC batteries catching fire must be considered. All batteries were inspected for physical damage that may compromise their safe performance under load. Appropriate maximum and minimum voltages of 4.2 V and 3.0 V were chosen to prevent damage to the battery due to overcharging or completely flattening the battery. A maximum current for cycling the batteries of 1 A was chosen to aid in temperature management and to further ensure the safety of the experiment. These precautions will see the battery operating well within the scope of what can be considered normal use for batteries of this kind and reduce the risk of catastrophic failure. Further safety precautions were taken to minimise the risk posed by a fire. All batteries were placed on fire resistant fabric for the duration of the experiment and flammable items such as loose paper were removed from their

proximity. A fire alarm and an appropriate fire extinguisher were installed in the room as final precautions.

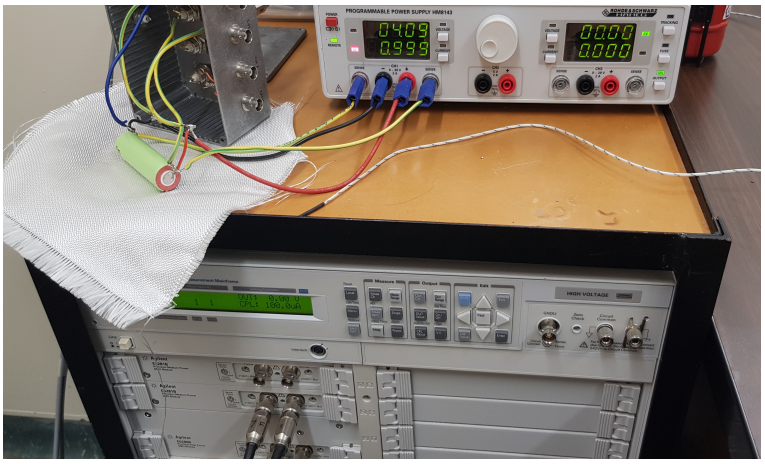


FIGURE 4.2: Experimental setup of the 5270 and the Hameg. The 5270 is used for measuring the impedance and the Hameg is used for the cycling of the battery. The 4 wire measurement ensures resistive loss in the wires does not affect the measurements of battery voltage and impedance. Each system is able to present a high impedance when inactive allowing alternating operation of both devices.

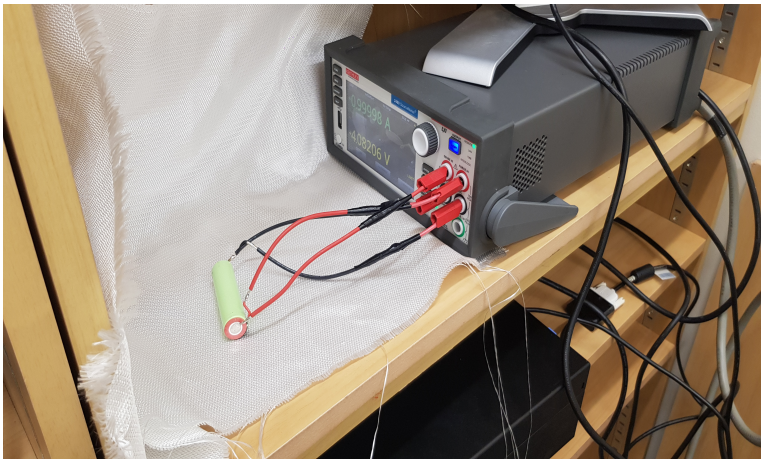


FIGURE 4.3: Experiemental setup of the Keithley 2460. Fire resistant fabric lines the shelf.

FIGURE 4.4: Experimental setup of the 66332 on top of the Polar 1000 incubator set to 40°C. The 66332 output is on the reverse of the device.

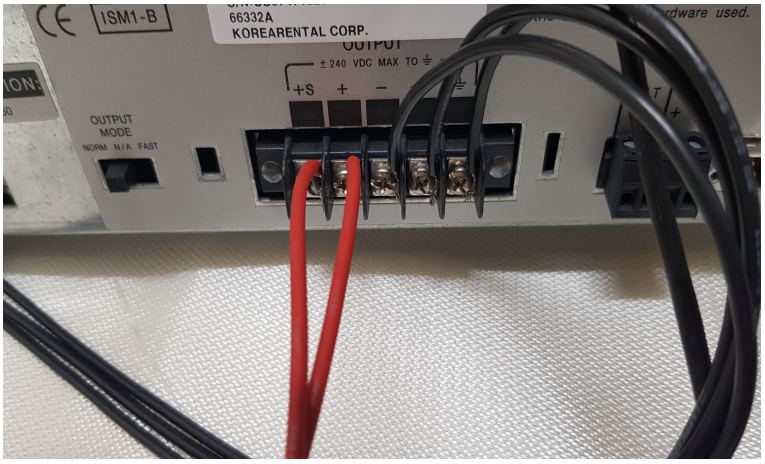


FIGURE 4.5: The output of the 66332 connecting to the battery in the incubator. The 4 wire measurement ensures resistive loss in the wires does not affect the measurements of battery voltage and impedance.

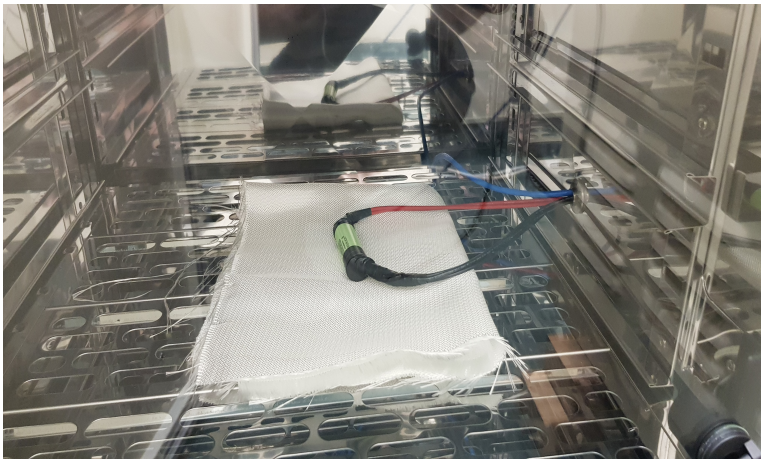


FIGURE 4.6: The battery connected to the 66332 sitting inside the Polar 1000 incubator. There is an aperture in the side of the incubator allowing the wires access into the heated space without affecting the incubators ability to maintain the set temperature. Fire resistant fabric lines the shelf and insulation tape is used to prevent short circuits.

4.1.4 Measurements using the Hameg and 5270

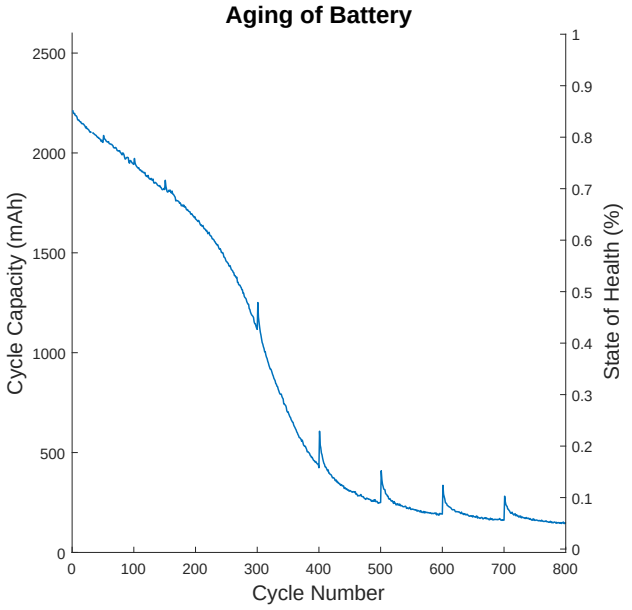


FIGURE 4.7: The battery connected to the Hameg and 5270 was aged for 800 cycles between 4.2 V and 3.0 V at 1 A using the Hameg. The regular spikes in battery capacity occur when the battery is allowed to relax while the impedance is measured. This recovery of capacity decays rapidly when cycling is resumed.

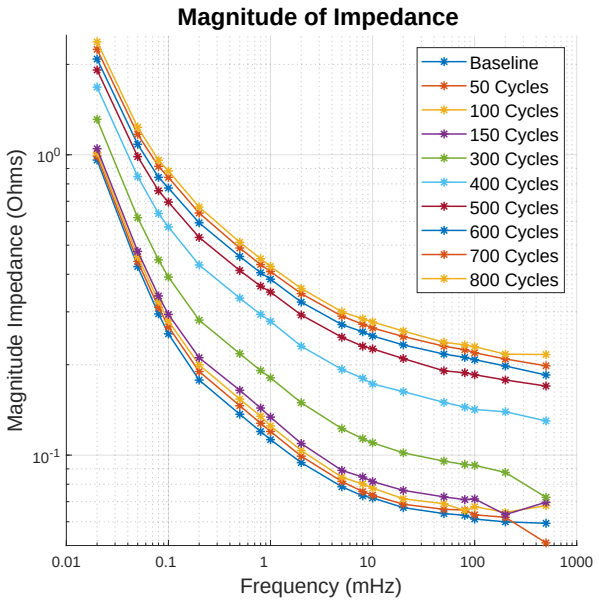


FIGURE 4.8: The impedance of the battery connected to the Hameg and 5270 was measured between periods of cycling. The impedance of the battery was measured using the 5270. The magnitude of the impedance increases with the number of battery cycles, with a larger relative change measured at higher frequencies and a larger abolute change at lower frequencies. The data measured at frequencies above 100 mHz is inconsistent due to low sampling rate.

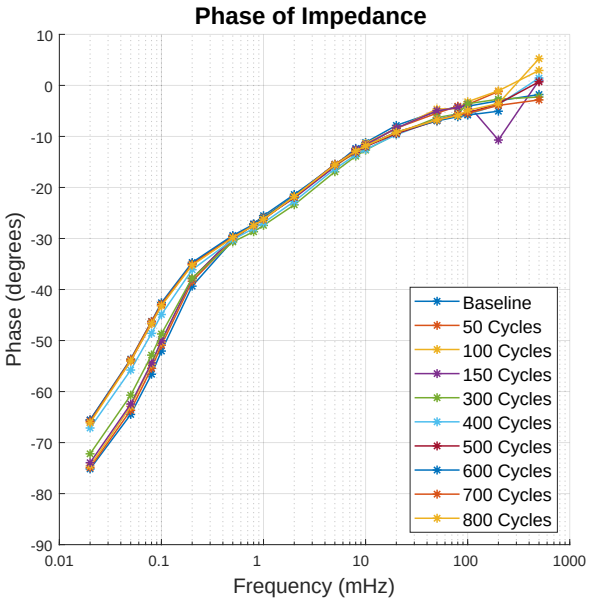


FIGURE 4.9: The impedance of the battery connected to the Hameg and 5270 was measured between periods of cycling. The impedance of the battery was measured using the 5270. The phase of the impedance changes with the number of battery cycles, the phase at decreases at higher frequencies and increases at lower frequencies. This is consistent with a decrease in Alpha. Alpha is related to the low frequency phase asymptote through Equation 2.3 however it is clear that measurements at lower frequencies are needed to obtain Alpha through this method. The inconsistency in data measured at frequencies above 100 mHz due to low sampling rate is pronounced.

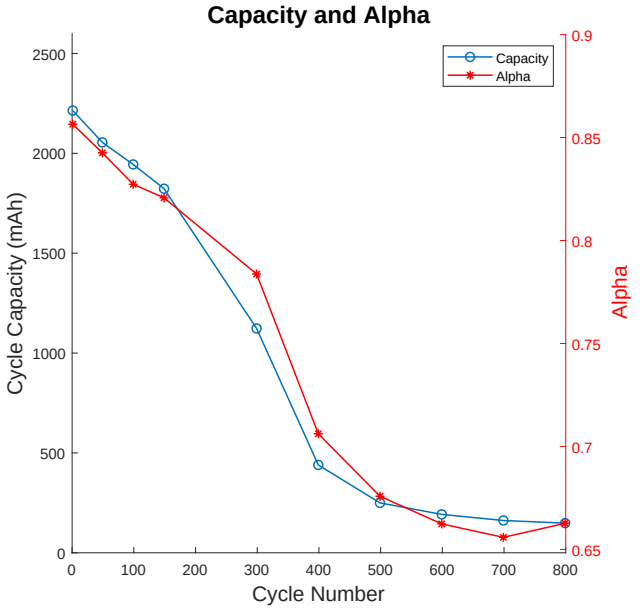


FIGURE 4.10: This battery was connected to the Hameg and 5270. The Alpha value is calculated from lowest frequencies of the impedance measured using the 5270 and the capacity was measured using the Hameg. There is a pronounced relationship between Alpha and capacity providing evidence for the viabilty of inferring a change in capacity from a change in Alpha.

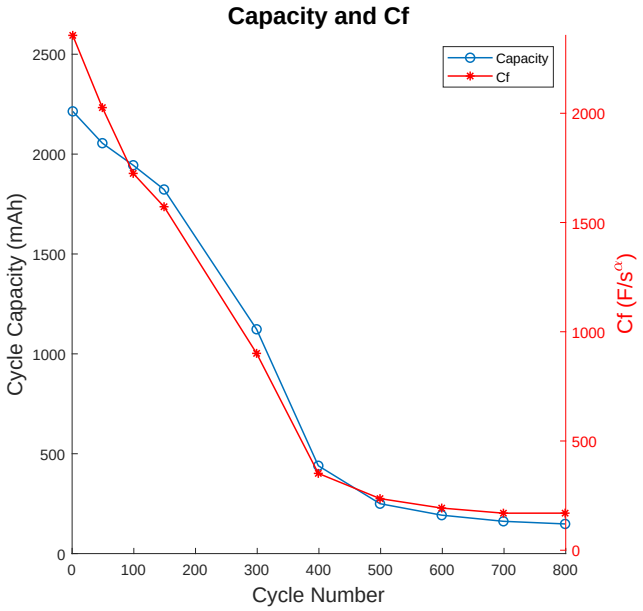


FIGURE 4.11: This battery was connected to the Hameg and 5270. C_F value is calculated from the impedance measured at 50 μHz using the 5270 and the capacity was measured using the Hameg. There is a clear relationship between C_F and capacity with both declining in unison over the extended cycling process. This observation provides evidence supporting the viabilty of inferring a change in capacity from a change in C_F.

4.1.5 Measurements using the Keithley 2460

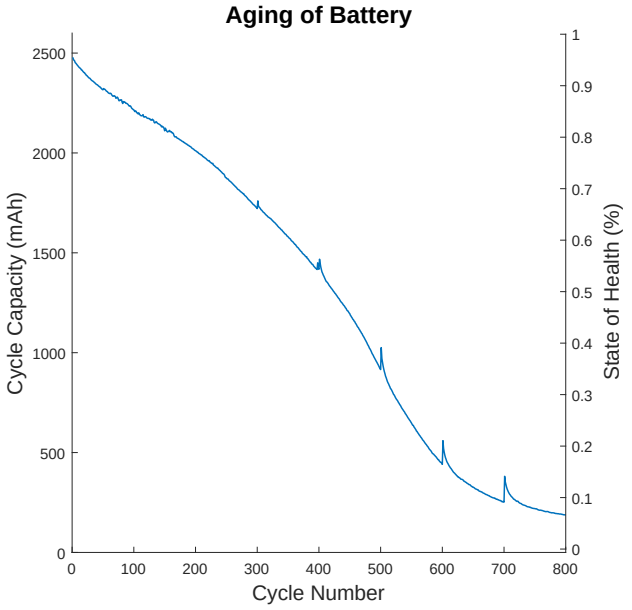


FIGURE 4.12: The battery connected to the Keithley was aged for 800 cycles between 4.2 V and 3.0 V at 1 A. The regular spikes in battery capacity occur when the battery is allowed to relax while the impedance is measured. This recovery of capacity decays rapidly when cycling is resumed. An artifact from external interference is noticable just prior to 400 cycles.

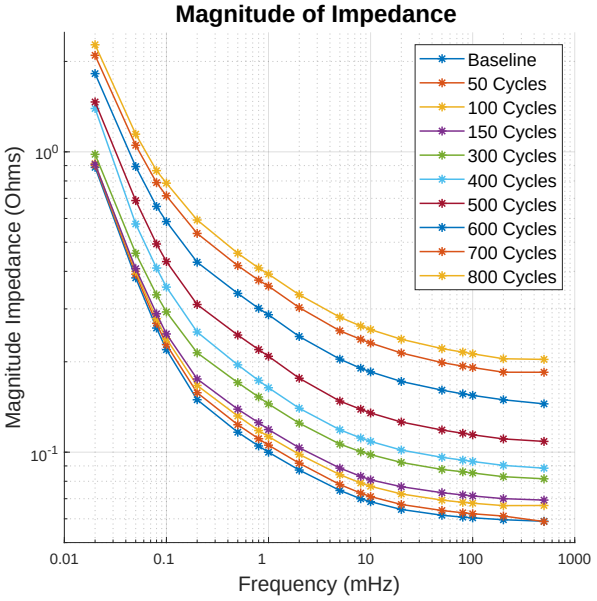


FIGURE 4.13: The impedance of the battery connected to the Keithley 2460 was measured between periods of cycling. The magnitude of the impedance increases with the number of battery cycles, with a larger relative change measured at higher frequencies and a larger abolute change at lower frequencies. The data measured at frequencies above 100 mHz has minor inconsistencies due to the low sampling rate. The 20 μHz measurement at 400 cycles has been corrupted due to an unknown source of errror external to the experiment.

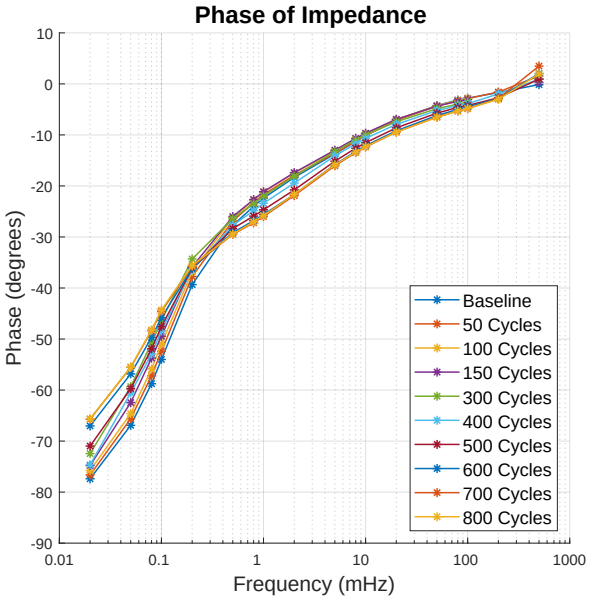


FIGURE 4.14: The impedance of the battery connected to the Keithley 2460 was measured between periods of cycling. The phase of the impedance changes with the number of battery cycles, the phase at decreases at higher frequencies and increases at lower frequencies. This is consistent with a decrease in Alpha. Alpha is related to the low frequency phase asymptote through Equation 2.3 however it is clear that measurements at lower frequencies are needed to obtain Alpha through this method. The error in the 20 μHz measurement at 400 cycles is noticable.

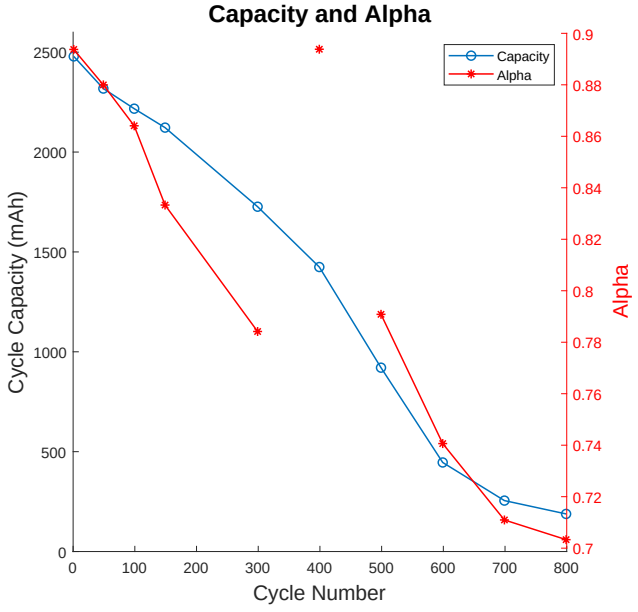


FIGURE 4.15: This battery was connected to the Keithley 2460. The Alpha value is calculated from lowest frequencies of the impedance measured using the Keithley 2460 and the capacity was also measured using the Keithley 2460. There is an apparent relationship between Alpha and capacity. The abnormallity of the Alpha measurement at 400 cycles is clear.

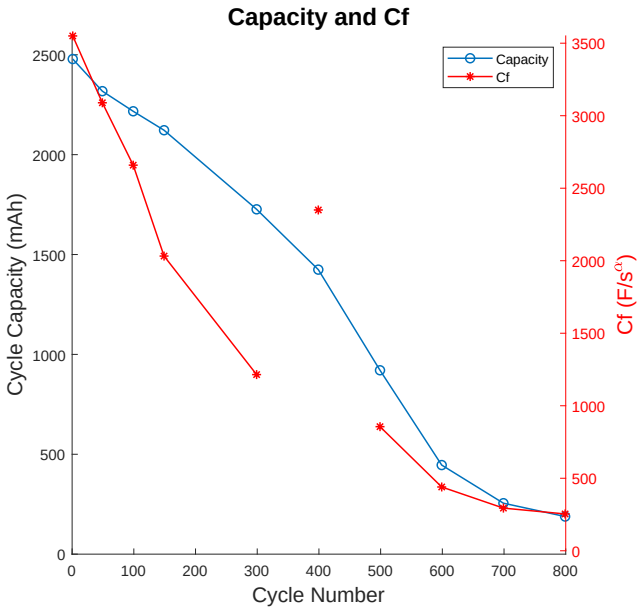


FIGURE 4.16: This battery was connected to the Keithley 2460. C_F value is calculated from the impedance measured at 50 μHz using the Keithley 2460 and the capacity was also measured using the Keithley 2460. There is an apparent relationship between C_F and capacity. The abnormallity of the C_F measurement at 400 cycles is clear.

4.1.6 Measurements using the 66332

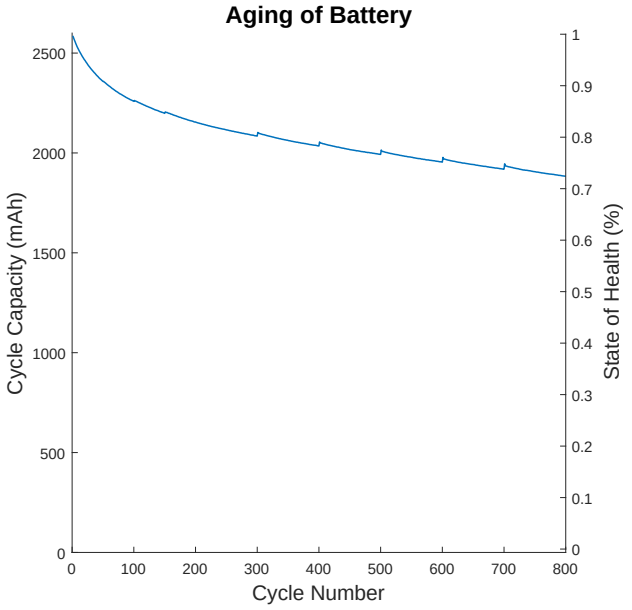


FIGURE 4.17: The battery connected to the 66332 was aged for 800 cycles between 4.2 V and 3.0 at 1 A while at a temperature of 40°C. The regular spikes in battery capacity occur when the battery is allowed to relax while the impedance is measured. The battery capacity did not decline as rapidly as the batteries measured using the other systems.

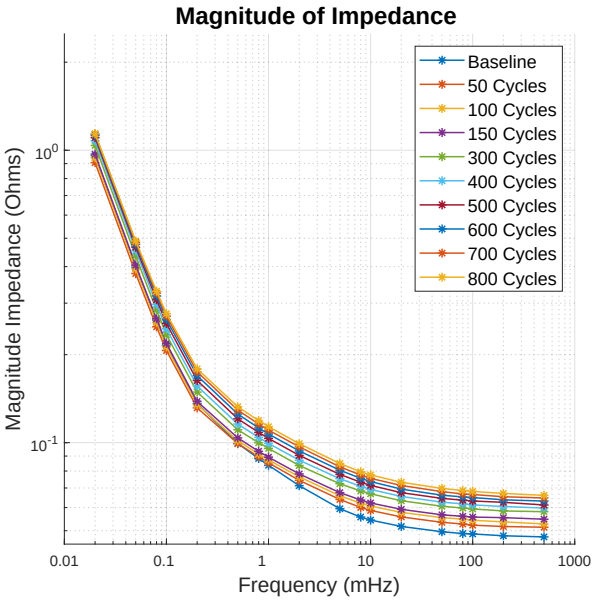


FIGURE 4.18: The impedance of the battery connected to the 66332 and raised to a temperature of 40°C was measured between periods of cycling. The magnitude of the impedance increases with the number of battery cycles but at a much slower rate than seen in the batteries at room temperature. Alpha is calculated using the slope of a straight line fit to the impedance at the three lowest frequencies. C_F is calculated from the impedance at 50 μHz.

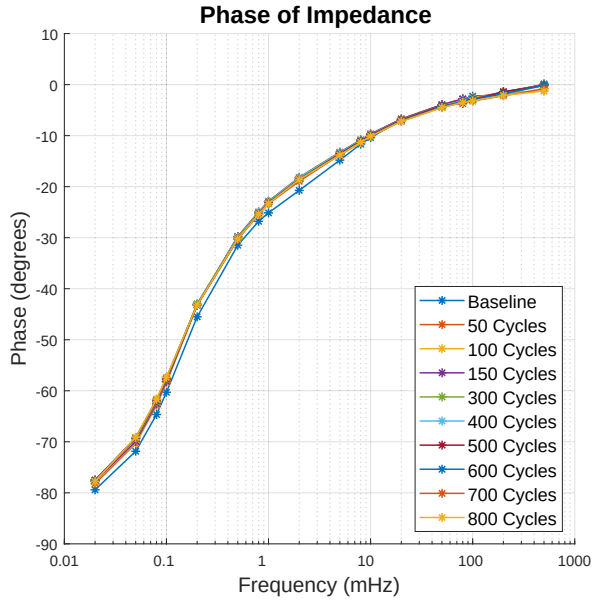


FIGURE 4.19: The impedance of the battery connected to the 66332 was measured between periods of cycling. The phase of the impedance does not seem to change with cycling. This is consistent with little to no change in Alpha. Alpha is related to the low frequency phase asymptote through Equation 2.3 however it is clear that measurements at lower frequencies are needed to obtain Alpha through this method.

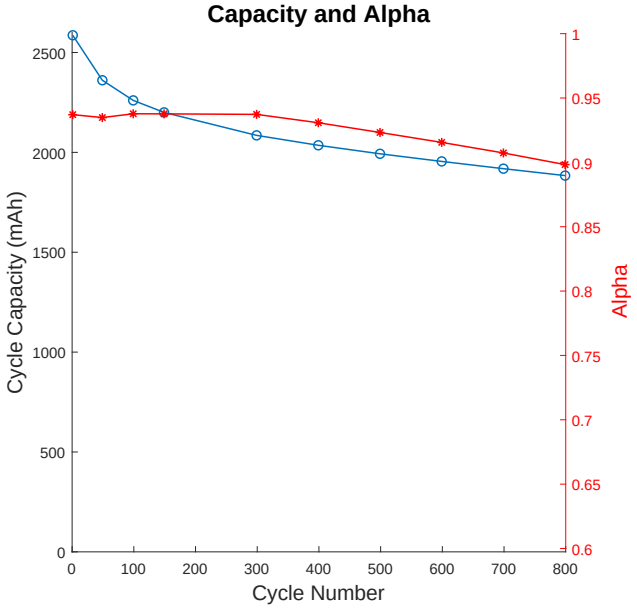


FIGURE 4.20: This battery was connected to the 66332 and raised to a temperature of 40°C. Alpha value is calculated from lowest frequencies of the impedance measured using the 66332 and the capacity was also measured using the 66332. There is a clear relationship between Alpha and capacity with both slowly declining over the extended cycling process. This observation provides evidence for the viabilty of inferring a change in capacity from a change in Alpha.

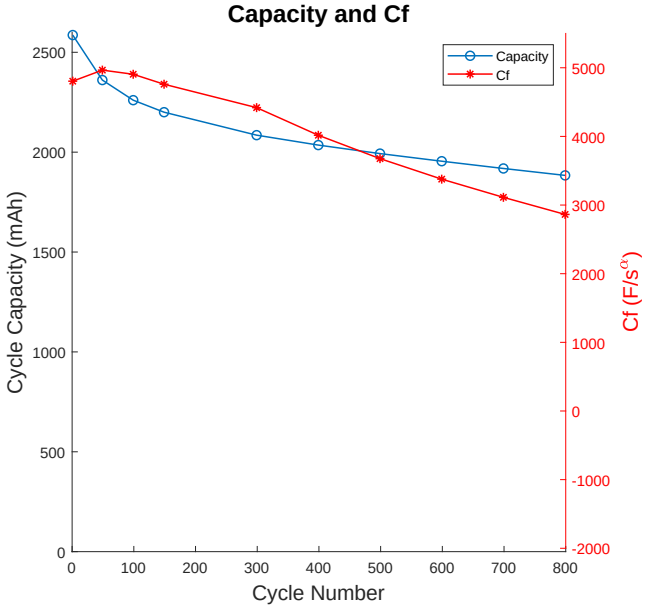


FIGURE 4.21: This battery was connected to the 66332 and raised to a temperature of 40°C. C_F value is calculated from the impedance measured at 50 μHz by the 66332 and the capacity was also measured using the 66332. There is an apparent relationship between C_F and capacity with both slowly declining over the extended cycling process. This observation provides evidence for the viabilty of inferring a change in capacity from a change in C_F.

4.2 Additional Measurements

4.2.1 Investigating Retired Lead Acid Batteries

In collaboration with industry partners, measurements of large lead acid batteries were made using the created tools. These measurements were made to compare the performance of existing age characterisation technology to the proposed method. The contribution of Professor Jonathan Scott to the making of these measurements and documents is recognised as contributing key information and valuable assistance. Measurements of the capacity and low frequency impedance were made on a pair of 280W and 540W lead acid batteries. In each pair one battery had been removed from service due to age and the other battery was brand new. Full manuscripts are attached in Appendix D and Appendix E.

In the case of the 280W batteries it is clear that the decommissioning of the retired battery was unnecessary as there has been no significant degradation to charge capacity or power capacity. These findings appear consistent with the measured Alpha values which vary slightly between each battery however without initial measurments of the aged batteries strong conclusions cannot be reached. The retired 540W battery shows significant degradation in capacity and likely should have been removed from service sooner than it was. The low frequency impedance is signicantly higher in the retired 540W battery however only a small change in Alpha is observed. Again without initial measurements of the aged batteries strong conclusions cannot be reached. In each case, while there is minor evidence in support of the proposed method it is clear that the performance of the existing method is lacking.

4.2.2 Investigating Battery Relaxation and Multi-tone Measurements

The proposed battery model is linear and suggests the potential for accurately measuring the impedance at multiple frequencies simultaneously using a multi-tone probing signal. This would greatly improve the time taken to make measurements at multiple low frequencies as required to determine Alpha and C_F. Initial testing showed that the impedance of the battery when measured with a single tone at a time and using the multi-tone method were different. Due to time limitations, and the testing already performed on the single tone measurements, multi-tone measurements were not used for the experiment in Section 4.1. Professor Jonathan Scott has used the measurement systems created in this work in Section 3.2 and Section 3.3 to investigate the cause of this disparity. He found cycling of the battery upset the internal equilibirum which returned over an extended period of relaxation, during which the measured impedance was affected. His results are attached in Appendix F for their relevance to the measurements and conclusions made in this work and as proof of the functionality of the created systems and their use by other members of the research group.

Chapter 5

Discussion

The data measured in the battery aging experiment in Section 4.1 shows a clear link between the both the values of Alpha and C_F, and the state-of-health of the measured batteries, with degradadation in the state-of-health being reflected in simultaneous reduction in both Alpha and C_F. This is most pronounced in Figure 4.10 and Figure 4.11 however the trend is also observed in the other measurements made. While it may be tempting to deem the lack of a drastic reduction in the state-of-health of the battery raised to 40°C a failure, it has provided a highly valuable set of measurments for comparison against the other data. This resilience to degradation in state-of-health is also reflected in a greatly reduced change in Alpha and C_F, as seen in Figure 4.20 and Figure 4.21. These observations combined form a weighty sum of evidence in support of the viability of quantifying a change in state-of-health in a battery through successive measurements of Alpha and C_F.

Although all three lithium-NMC batteries were purchased at the same time from the same supplier there is noticable variation in their properties. The inital values for capacity, Alpha, and C_F are all very different. While raising the temperature of one battery has undoubtably affected its measured properties, the variation seen in the room temperature batteries stands as testament to the variability introduced in the manufacturing process. Despite this inherent variation, a clear relationship between battery state-of-health and low frequency impedance characteristics is observed across all three batteries.

The reliability of the measurement systems created is shown through repeated testing and measurements across multiple battery technologies. The investigation into multi-tone measurements provides evidence in support of the constant-phase-element based battery model used as the basis of these experiments, and with it, support for the presented interpretation of the measured results. Overall the measurements made, and

the integrity of the conclusions based thereon, are believed to be sound as appropriate measures were taken to mitigate known sources of error and ensure the repeatability and reliablity of each measurement.

5.1 Limitations

The later measurements of the lithium-NMC batteries are polluted by factors not considered when designing the experiment. This effect is most pronounced in the two batteries measured at 20°C. The effect of recent activity on battery impedance is noticeable for a much longer period than realised when beginning the experiments. This effect becomes more prominent as alpha is reduced as the time taken for this effect to diminish increases. These properties, combined with the order of the single tone measurements in the experiment, will have polluted the measurements by artificially lowering the measured impedance at higher frequencies. This will have the effect of artificially *increasing* the Alpha value as seen in the later measurements as we do not have a method to correct for the introduced error. The impact of this limitation on the results is negated in part as the data points most affected are those measured well after the battery state-of-health had diminished past any reasonable measure of serviceability. Additionally, since the trend we are observing is that Alpha decreases alongside state-of-health, this suggests that when properly taken into account there will be an even stronger response to battery degradation than measured in our data.

The adjustments to the testing scheme were based on the behaviour of the battery in the incubator at 40°C. The assumption that this battery would age the fastest, based on the literature, lead to changing the timing of all the experiments being run. Sadly, this proved to not be the case and the timing of the change was such that potentially the most valuable data points on the Keithley and Hameg/5270 batteries were lost due to this adjustment. While all of the data continues to support a link between Alpha and C_F, and battery state-of-health, the conclusions we can draw would have benefitted from inclusion of those missing measurements.

The measurements made using the Keithley 2460 show an external interference with the experiment just prior to the 400 cycles mark, this is visible in the state-of-health measurements Figure 4.12 and is clearly evident in both the impedance, Figure 4.13, and the derived Alpha and C_F values, Figures 4.15 and 4.16. The nature of this interference is unknown and occured during the Covid-19 pandemic lockdown and so could not be detected or corrected for. While this interference has degraded the quaility of results measured on the Keithley 2460 the other measurements are sufficient to form the basis of our conclusions.

Chapter 6

Accomplishments

6.1 Conclusions

Battery reliability is a prominent issue where the life cycle of critical items is defined by the working life of the battery that powers them. Current methods of measuring the state-of-health of a battery are sorely lacking in performance. This leads to batteries being prematurely replaced to avoid dire consequences due to undetected failure. Recent advancements in battery modelling have lead to investigating a new approach to measuring battery state-of-health using small signal, low frequency, impedance measurements.

To verify this method of determining state-of-health an experiment has been performed. Systematic measurement and degradation of the state-of-health of a battery through cycling, and regular measurements of the low frequency impedance, has provided conclusive evidence in support of the proposed method of measuring state-of-health. To this end, tools for measurement of low frequency battery impedances and for battery cycling and state-of-health measurement have been successfully created and verified. Each tool has been tested rigourously across multiple devices and battery technologies and has performed to a high standard. The experiment ran continuously for nine months without issue or failure using these tools.

The results of the experiment show a clear link between the battery state-of-health and the low frequency impedance characteristics, Alpha and C_F. This link is evident at multiple temperatures, and in multiple battery technologies, and may form the basis of a state-of-health measurement system more reliable than current methods.

While the physical cause of these properties requires further investigation there is mounting evidence in suppport for the constant-phase-element based model upon which they

are based. This, along with the congruent results from the highly successful measurement tools, provides confidence in the given interpretation of the results, and satisfies the goals of this work

6.2 Future work

Examining the properties of the proposed constant-phase-element based model and the measurements presented in this work suggests the following promising avenues for future research.

(1) Since a definitive link between Alpha, C_F, and state-of-health has been established, further investigation into the effects of state-of-charge, temperature, and current rate on the measurements should be undertaken. This would provide the neccessary foundation for future refinements to the model to account for varying environmental conditions and application in practical systems.

(2) The proposed constant-phase-element based battery model accurately fits low frequency impedance measurements, however does not account for high frequency battery behaviours. Creation and verification of a combined model to allow accurate simulation would enable the design and manufacture of battery powered devices to maximise battery utilisation.

(3) Measurements of low frequency battery impedance are now known to be affected by large stimulus to the battery for many thousands of seconds after the stimulus has been removed. This has the potential to heavily limit the application of low frequency impedance measurements in systems with frequent or irregular use patterns due to disturbances to the measurement. Investigating methods to compensate for this interference will be vital for creating a product for in-situ battery state-of-health measurements.

(4) The frequencies measured were only able to reach down to 20 μHz due to time limitations. Using multi-tone measurements as seen in Section 4.2.2 will allow full sweeps of measurements with more individual frequencies, and reaching to much lower frequencies, to be taken within the same time span. A categorisation of the low frequency impedance of batteries across multiple battery technologies, with both lower minimum frequencies and higher data densities, is recommended to build a database of knowledge for reference for future research.

(5) The success of multi-tone measurements suggests the potential to pursue probing signals of arbitrary shape. Investigating battery impedance measurements using

step functions, delta functions, and pseudorandom signals will pave the way to make measurements of battery impedance with greatly enhanced frequency resolution.

Appendix A

Unified Measurement Interface and Battery Measurement Tools

The unified interface for measurement automation is available from a public github repository https://github.com/Rockscanfly/NinjaMasters

Additionally a command line fourier transform program that can be used to analyse the impedance data is available https://github.com/Rockscanfly/DFTp

The project is tested for both Windows and Linux and is set up to build the impedance measurement and battery cycling tools by default. The build system is managed using CMake. Compiling with gcc/g++ is recommended.

A.1 Battery Impedance Measurement

By default the project creates an executable called 'bz' when compiled. This is the low-frequency impedance measurement tool. Renaming of the executable is permitted.

Running the executable without input arguments or with incorrect arguments will print the following help message

Usage: bz ⟨device⟩ ⟨GPIB#⟩ ⟨gpib_addr⟩ ⟨Vmax⟩ ⟨Vmin⟩ ⟨Imax⟩ ⟨QMax⟩ ⟨Ncycles⟩ [S—s—Single] ⟨Frequency List⟩ [filestring]
Usage: bz ⟨device⟩ ⟨serial_port⟩ ⟨baud⟩ ⟨Vmax⟩ ⟨Vmin⟩ ⟨Imax⟩ ⟨QMax⟩ ⟨Ncycles⟩ [S—s—Single] ⟨Frequency List⟩ [filestring]
Parameters in square brackets [ ] are optional

bz HP66332 2 3 ... connects to a HP66332 at GPIB2::3::INSTR
bz HP66332 GPIB2 3 ... connects to a HP66332 at GPIB2::3::INSTR
bz Keithley COM20 9600 ... connects to a Keithley at COM20 with a baud rate of 9600
bz Hameg /dev/ttyS20 115200 ... connects to a Hameg at /dev/ttyS20 with a baud rate of 115200

This program requires a device connected to the host machine accessible at the specified GPIB address or serial port, and the visa driver interface to be installed on the host

The program is designed to measure the impedance of a battery attached to the device at the specified frequencies. The program expects the voltage and current ranges to accurately represent the safe operating values of the battery and provides no guarantee it will not damage any attached device

Data and log files will be created in the same directory as where the program was called from
The optional argument [filestring] will overwrite the default file names DataFile.tvi and LogFile.log and results will be stored in [filestring].tvi and [filestring].log The .tvi file stores measurements in three tab separated columns time(s) volts(V) current(A)
The .log file stores time-stamped plain english descriptions of milestones reached by the program

⟨device⟩ is the hardware to be used to measure imepedance with. Options are:
Keithley Hameg HP66332 E5270
⟨GPIB#⟩ is the GPIB address value of the primary switch that the device is attached to
⟨gpib_addr⟩ is the GPIB subaddress value of the device attached to the GPIB switch at

⟨GPIB#⟩
⟨serial_port⟩ is the alternative serial port to connect to ie. COM20 or /dev/ttyS20
⟨baud⟩ is the baud rate when using a non-GPIB serial port
⟨vmax⟩ is the maximum voltage, in Volts, up to which the battery will be charged, the program will exit if this limit is reached during cycling
⟨vmin⟩ is the minimum voltage, in Volts, down to which the battery will be discharged, the program will exit if this limit is reached during cycling
⟨imax⟩ is the maximum current, in Amps, that will be sourced or sunk to force a waveform into the battery
⟨qmax⟩ is the maximum charge, in Amphere Hours, that will sourced or sunk to force a waveform into the battery
⟨ncycles⟩ is the number of sinusoidal current cycles, at the lowest given frequency, the battery will experience
[Single] is an option string to specify that the given list of frequencies should be iterated over individually instead of simultaneously to use this option, type the word "Single" BEFORE listing frequencies
⟨frequency list⟩ is a list of frequencies, in Hz, used to create the current waveform that will be produced, limit 16 frequencies, maximum 1Hz
[filestring] sets the base name of the target .tvi and .log files, NOTE existing files with the same name will be OVERWRITTEN
Data files with the name format [filestring]_[first frequency in uHz]uHz[# of frequencies]tone.tvi will be made for multitone sweeps
Data files with the name format [filestring]_[frequency in uHz]uHz.tvi will be made for single tone sweeps

A.2 Battery Cycling and State-of-Health Measurement

By default the project creates an excutable called 'bc' when compiled. This is the battery cycling and state-of-health measurement tool. Renaming of the exectuable is permitted.

The executing the program without input arguments or with incorrect arguments will print the following help message

Usage: bc ⟨device⟩ ⟨GPIB#⟩ ⟨gpib_addr⟩ ⟨ncycles⟩ ⟨Vmax⟩ ⟨Vmin⟩ ⟨Imax⟩ ⟨Iend⟩ ⟨Qend⟩ ⟨timeout⟩ ⟨relaxtime⟩ [filestring]
Usage: bc ⟨device⟩ ⟨serial_port⟩ ⟨baud⟩ ⟨ncycles⟩ ⟨Vmax⟩ ⟨Vmin⟩ ⟨Imax⟩ ⟨Iend⟩ ⟨Qend⟩

⟨timeout⟩ ⟨relaxtime⟩ [filestring]
Parameters in square brackets [ ] are optional

bc HP66332 2 3 ... ⟨ connects to a HP66332 at GPIB2::3::INSTR
bc HP66332 GPIB2 3 ... ⟨ connects to a HP66332 at GPIB2::3::INSTR
bc Keithley COM20 9600 ... ⟨ connects to a Keithley at COM20 with a baud rate of 9600
bc Hameg /dev/ttyS20 115200 ... ⟨ connects to a Hameg at /dev/ttyS20 with a baud rate of 115200

This program requires a device connected to the host machine accessible at the specified GPIB address, and the visa driver interface to be installed on the host

The program is designed to cycle a battery attached to the device between the specified maximum and minimum voltages at the specified maximum current. The program expects the voltage and current ranges to accurately represent the safe operating values of the battery and provides no guarantee it will not damage any attached device

Data and log files will be created in the same directory that the program was called from
The optional argument [filestring] will overwrite the default file names DataFile.tvi and LogFile.log
and results will be stored in [filestring].tvi and [filestring].log
The .tvi file stores measurements in three tab separated columns time(s) volts(V) current(A)
The .log file stores time-stamped plain english descriptions of milestones reached by the program

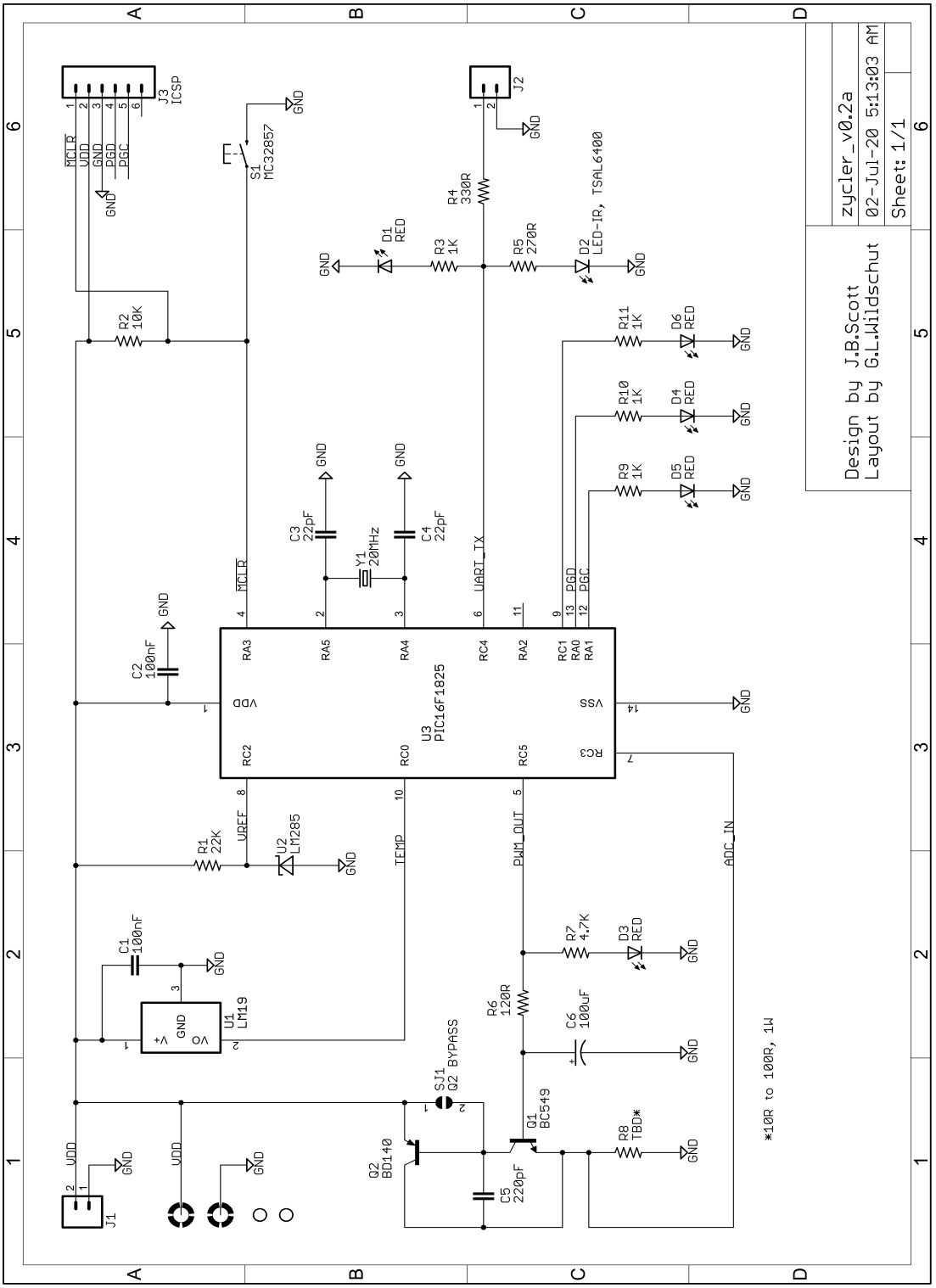
⟨device⟩ is the hardware to be used to cycle the battery with. Options are:
Keithley Hameg HP66332 E5270
⟨GPIB#⟩ is the GPIB address value of the primary switch that the device is attached to
⟨gpib_addr⟩ is the GPIB subaddress value of the device attached to the GPIB switch at ⟨GPIB#⟩
⟨serial_port⟩ is the alternative serial port to connect to ie. COM20 or /dev/ttyS20
⟨baud⟩ is the baud rate when using a non-GPIB serial port
⟨ncycles⟩ is the number of times the attached battery will be fully discharged and then fully charged
⟨vmax⟩ is the maximum voltage, in Volts, up to which the battery will be charged
⟨vmin⟩ is the minimum voltage, in Volts, down to which the battery will be discharged

⟨imax⟩ is the maximum current, in Amps, that device will source or sink to charge or discharge the battery
⟨iend⟩ is the minimum current, in Amps, flowing when the program will decide the battery is fully charged or discharged
⟨qend⟩ is the final fraction of charge left in the battery as a percent of fully charged, expected values are [100 - 0]
⟨timeout⟩ is the maximum amount of time, in Seconds, the current will be allowed to decay after falling below ⟨imax⟩ before the program decides the battery is fully charged or discharged
⟨relaxtime⟩ is the time, in Seconds, the voltage will be allowed to relax after all cycling is complete and the charge level has reached ⟨qend⟩, zero current will flow but the device will continue measurements for this time
[filestring] sets the name of the target .tvi and .log files, NOTE existing files with the same name will be OVERWRITTEN

Appendix B

Single Board Battery Impedance Measurement

B.1 Schematic



B.2 Code Listing

```
// program to find Z(f) of attached battery on Zycler PCB
// July 2020, Jonathan Scott
// jbs-main.c   PIC16F1825 version (not PIC16F18426 version)
// Written for XC8; we saved a lot of memory with v2.2 compared to v1.45

// Circuit connections
// ------------------------------------------------------------------------
// pin 2 - OSC1 Xtal oscillator
// pin 3 - OSC2 Xtal oscillator
// pin 4 - PB active low (RA3)
// pin 5 - P1A PWM output to current DAC & LED
// pin 6 - serial Tx (RC4)
// pin 7 - AN7 (RC3) measuring current
// pin 8 - AN6 (RC2) measuring Vref (1.21 or 2.50V) to determine Vsupply value
// pin 9 - Error LED (active high) (RC1)
// pin 10- AN4 (RC0) measuring LM19 temperature
// pin 11- LED progress/heartbeat (RA2)
// pin 12- SPARE (not working... RX?) (RA1)
// pin 13- LED data available DAv (RA0)
// ------------------------------------------------------------------------
// diagnostic LED oututs:
// Heartbeat is one blink in a bunch per cycle undertaken;
// When Data AVailable, DAV LED blinks at sample rate
// ERR LED blinks on before heartbeat if deltaTemperature excessive
// ERR LED blinks on with heartbeat if delta voltage is excessive
// duration of ERR LED blink is 10*Temperature(C) milliseconds long
// after boot, all LEDs stay on through whole stimulus cycle (minutes?)
// Pressing PB for 0.5s starts serial output, cancels DAV indicator
// Pressing PB for 2s turns serial data OFF, cancels ERR indicator
// Pressing PB for 10s reboots

// thing to be done, 18426 version (when there is more memory)
// do not forget changes (CONFIG, ISR, register initialisation) for 20MHz crystal
// switch to 32-bit floating point
// track time-modulo for each tone individually
// if full 32 bit too slow (1Hz?) might need fast sin approximation, or float-sin(double)
// Make output strings more human-readable, add Temperature and Voltage limits
// Put data into NVRAM, remember last few measurements
// add serial Rx channel to request diagnostic readout of values, past results, etc

// PIC16F1825 Configuration Bit Settings
// #pragma config statements should precede project file includes.
// So that we use project enums instead of #define for ON and OFF.
// CONFIG1
#pragma config FOSC = INTOSC    // Oscillator Selection (INTOSC oscillator: I/O function on CLKI
//#pragma config FOSC = HS        // Oscillator Selection (HS Oscillator, High-speed crystal/res
#pragma config WDTE = OFF       // Watchdog Timer Enable (WDT disabled)
#pragma config PWRTE = ON       // Power-up Timer Enable (PWRT enabled)
#pragma config MCLRE = OFF      // MCLR Pin Function Select (MCLR/VPP pin function is digital in
#pragma config CP = OFF         // Flash Program Memory Code Protection (Program memory code pro
#pragma config CPD = OFF        // Data Memory Code Protection (Data memory code protection is d
#pragma config BOREN = ON       // Brown-out Reset Enable (Brown-out Reset enabled)
#pragma config CLKOUTEN = OFF   // Clock Out Enable (CLKOUT function is disabled. I/O or oscilla
```

```
#pragma config IESO = ON        // Internal/External Switchover (Internal/External Switchover mo
#pragma config FCMEN = ON       // Fail-Safe Clock Monitor Enable (Fail-Safe Clock Monitor is en
// CONFIG2
#pragma config WRT = OFF        // Flash Memory Self-Write Protection (Write protection off)
#pragma config PLLEN = OFF      // PLL Enable (4x PLL enabled)
#pragma config STVREN = ON      // Stack Overflow/Underflow Reset Enable (Stack Overflow or Unde
#pragma config BORV = LO        // Brown-out Reset Voltage Selection (Brown-out Reset Voltage (V
#pragma config LVP = OFF        // Low-Voltage Programming Enable (Low-voltage programming enabl

#include <xc.h>
#include <string.h>
#include <math.h>
#include <stdio.h>

// !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
// these are set at compile time for each application
#define NOMMV   3300                    // nominal cell voltage
#define VERROR  3200                    // the "flat" voltage value
#define dTERROR 4                               // the thermal shift limit in Celcius
#define VREF    2502                    // the actual value of LM285 (~2500 or ~1220)
#define RS      56                                      // actual current sense resistor value
#define NCYCLES 6                               // number of cycles of lowest frequency
#define NF 10                                   // number of tones
#define PERIOD1 1000000L                // number of 10ms periods in PERIOD of lowest frequency
const double freq[NF]={200e-3, 100e-6, 200e-6, 400e-6, 600e-6, 100e-3, 1e-3, 2e-3, 10e-3, 50e-3}
// !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//#define PERIOD1 100000L               // 10x value for testing, results in ~100 minutes
//const double freq[NF]={1000e-6, 2000e-6, 4000e-6, 6000e-6, 10e-3, 20e-3, 50e-3, 100e-3, 200e-3
// !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!


#define PB RA3                                  // the press button, toggles serial, resets flag
#define HBLED RA2                               // 1Hz heartbeat, pulse lengthens with cycle pro
#define DAVLED RA0                              // indicates new data available (measurement com
#define ERRLED RC1                              // voltage or temperature may invalidate data
#define SPARE RA1                               // spare pin, not working?

#define ON 1
#define OFF 0
#define TRUE 1
#define FALSE 0
#define IMAG 1
#define REAL 0
#define VSELECT ADCON0=0b00011001;              // AN6, ON, idle
#define ISELECT ADCON0=0b00011101;              // AN7
#define TSELECT ADCON0=0b00010001;              // AN4
#define TWOPI 6.28318530718                             // radians/cycle
#define PI 3.1415926539                                 // the constant

// globals, access in ISR and main
unsigned long int t10ms;                        // time since start in multiples of 10ms
unsigned char cycle;                            // progress in cycle number
char msg[20];                                           // line + LF + NULL termination
signed short int degC;                          // the temperature in Celcius
signed short Tmin=1000;                         // low T limit
```

```
signed short Tmax=0;                            // high T limit
signed short Vmin=5500;                         // low Vb limit
signed short Vmax=0;                            // hi Vb limit
bit serial;                                                 // sending out data stream?
bit reboot;                                                 // kill while loop, warm boot
bit dav;                                                    // data available flag
bit overDeltaT;                                     // temp has shifted too many degrees
bit badvoltage;                                     // battery went flat?
bit sync;                                                   // ISR sync flag
bit stimcalc;                                       // set once per loop, when setting curre

//**************************************************************************
//Functions
//**************************************************************************

void ltomsg(long unsigned int n, char* str)     // long to ASCII
{
        char c;
        char* cptr;
        long unsigned int ltmp=n;

        cptr=str+2;                                                 // pointer into string s
        do{ltmp/=10;cptr++;}while(ltmp>0);      // how many decimal chars needed?
        *cptr-- = 0;                                                // NULL terminate, step back
        *cptr-- = '\n';                                             // LF, step back
        ltmp=n;                                                     // reload number
        do{
                c=ltmp%10;                                          // remainder after /10 (
                ltmp/=10;                                           // like >>1 in decimal
                c+='0';                                             // convert to ASCII
                *cptr-- = c;                                        // put in msg, step back
        } while(ltmp>0);                                            // until all needed digits done
        str[0]=' ';                                                 // add starting space (o
        return;
}

// ISR tasks:
// keep time in 10ms chunks
// if UART empty and buffer not empty, add byte
// poll, de-bounce, and action PB (ack DAV, dis/enable serial, reboot)
// update error, heartbeat/progress, and DAV LEDs

void interrupt ISR(void){

        static unsigned short debounce; // PB de-bounce counter
        static unsigned char ctr;               // heartbeat counter
        static unsigned char mptr;              // output string pointer
        char ct;

        if(TXIE && TXIF){                               // TXIE is enabled and TXREG is empty
                if(msg[mptr]==NULL){            // end of msg
                        mptr=0;                                     // pointer to start
                        TXIE=0;                                     // disable EUSART interrupts
                        return;
                }
```

```
                TXREG=msg[mptr++];                      // load next char to Tx (will clear TXIF
                // do not return from here; allow next instructions for TXIF to clear
        }

    if(TMR1IF){                     // 20MHz Xtal, 5MIPs, 0.2us clock
        TMR1IF=0;
//        TMR1=(65535-49986);                   // set so ISR rate is 10ms/100Hz from 20MHz osc
        TMR1=(65535-39986);                     // set so ISR rate is 10ms/100Hz from 16MHz osc
                // value of this constant should be set using the lab DFM
        t10ms+=1;                   // increment time in 10ms intervals
                sync=1;                                         // ISR running

                if(PB==0){                                      // button pressed
                        if(++debounce>50){              // half sec press...
                                dav=0;                          // clear DAV flag
                                serial=ON;                      // start sending data
                        }
                        if(debounce>200){               // 2 sec press...
                                serial=OFF;                     // stop sending data
                                if(overDeltaT){         // Temperature-related error
                                        Tmin=1100;              // reset low T limit
                                        Tmax=0;                 // reset high T limit
                                }
                                if(badvoltage){         // voltage-related error
                                        Vmin=5500;              // reset low Vb limit
                                        Vmax=0;                 // reset hi Vb limit
                                }
                                overDeltaT=0;           // clear T error
                                badvoltage=0;           // clear Vb error
                        }
                        if(debounce>1000){              // 10 sec press
                                reboot=1;
                        }
                }else{debounce=0;}                      // clear de-bounce counter

                // control of information LEDs
                if(++ctr > 99){ctr=0;}
                if((ctr&2)&&(ctr<(cycle<<2))){HBLED=ON;}else{HBLED=OFF;}// cycle blink
                if((dav||!PB)&&stimcalc){DAVLED=ON;}else{DAVLED=OFF;}   // show DAV
                ct=(unsigned)100-degC;
                if( (badvoltage&&(ctr<degC)) ||
                                (overDeltaT&&(ctr>ct)) )
                {ERRLED=ON;}else{ERRLED=OFF;}
        }
}

void integrate(double* Rsum, double* Isum, char i, long tn, long tnm1, double yn, double ynm1){

        double kay,ang,slope,thiscos,thissin,lsin,lcos;

        kay=(tn-tnm1)/100.00;
        slope=(yn-ynm1)/kay;

        kay=TWOPI*freq[i];
        ang=kay*tnm1/100.0;                                             // convert time into sec
```

```
        lcos = cos(ang);
        lsin = sin(ang);
        ang=kay*tn/100.0;                                          // convert tn into secon
        thiscos = cos(ang);
        thissin = sin(ang);

        Isum[i] += slope*(thissin-lsin)/(kay*kay);
        Isum[i] -= (yn*thiscos-ynm1*lcos)/kay;
        Rsum[i] += slope*(thiscos-lcos)/(kay*kay);
        Rsum[i] += (yn*thissin-ynm1*lsin)/kay;

}
/* from Scott & Parker 1995
                lsin = sin(0.0);
                lcos = cos(0.0);
                kay = TWOPI * j * fundamental;
                for(i=1; i<ndatl; i++) {
                        ang = (xjbs[i]-xjbs[0])*kay;
                        width = xjbs[i]-xjbs[i-1];
                        slope = (yjbs[i]-yjbs[i-1])/width;
                        thiscos = cos(ang);
                        thissin = sin(ang);
                        imag += slope*(thissin-lsin)/(kay*kay);
                        imag -= (yjbs[i]*thiscos-yjbs[i-1]*lcos)/kay;
                        real += slope*(thiscos-lcos)/(kay*kay);
                        real += (yjbs[i]*thissin-yjbs[i-1]*lsin)/kay;
                        lsin = thissin;
                        lcos = thiscos;
                        }

                        Bhaskara's Sine approximation
                        sin(x) ~ 16x(pi-x)/(5pi^2-4x(pi-x)) 0<x<pi error <1.5% (2 *, 1 /)
                        error for 7th-order poly is 3ppm, so error <.0003% (10 *)
                        sin(x) ~ x-x^3/3!+x^5/5!-x^7/7!
                        arctan(x) ~ see Rajan etal., Efficient Approximations for the Arctangent

 */

void main(void){

        unsigned char i;                                           // loop counter
        unsigned long int nseq=-1;                        // counts samples
        unsigned char nvar;                                        // what variable to output
        unsigned long tstamp, prevtstamp;       // measurement timestamp
        unsigned long tmodulo;                                     // measurement time, modulo LCM period
    signed short int VbmV=NOMMV, prevVb;// supply in millivolts
    signed short int IbcuA, prevIb;             // load current in multiples of 100uA
        signed short int adc;                                      // reading register
        long signed int ltmp;
        double dtmp,dtmp2;
        double win;                                                // phase/value of raised
        long unsigned int nseg=0;                                  // number of samples/integration points
        double Zmag[NF], Zpha[NF];                                 // the target outputs
        double ReVsum[NF], ImVsum[NF];          // running sums in real & imaginary
        double ReIsum[NF], ImIsum[NF];          // running sums in real & imaginary
```

```
    double Vbwin, Ibwin;                        // I and V values, windowed
    double prevVbwin, prevIbwin;          // previous I and V values, windowed
    double sumOfSines;                          // generator value
    float stimax, stimin;                       // max & min stimulus values
    signed short int pwm;                       // analog output via PWM in ECCP

    OSCCON     = 0b01111000;                    // 16MHz
    TRISA      = 0b00001000;                    // all outputs, but RA3 always an input
    TRISC      = 0b00001101;                    // RC0, RC2, RC3 inputs (AN4, AN6, AN7))
OPTION_REG = 0b00000100;                  // TMR0 set for /32
    TMR0IE     = 0;                             // no TMR0 interrupts
T1CON      = 0b00000001;                  // TMR1 ON
    T1GCON     = 0;                             // no TMR1 gate functions
    TMR1IE     = 1;
    PEIE       = 1;                             // enable TMR1 interrupts

    PR2        = 0xFF;                          // may be redundant
    CCP1CON    = 0b00001100;                    // P1A active high, PWM mode
    T2CON      = 0b00000100;                    // T2 on, 1:1 pre and post

    TXSTA      = 0b00100000;                    // async, 8-bit, enable, low rate
    RCSTA      = 0b10000000;                    // enable serial SPEN
    BAUDCON    = 0b00011000;                    // inverted TX data, 16-bit SPBRG (BRG16
    SPBRGL     = 0x05;                          // gives baud rate (4167=0x10:0x
    SPBRGH     = 0x0D;                          // of 300
    //APFCON0=0;                                // resets to zero
    //APFCON1=0b0001000;                  // resets to zero

    ADCON0     = 0b00011001;                    // ADC ON, channel AN6 (VREF)
    ADCON1     = 0b10100000;                    // Right-justified, fosc/32
    ANSELC     = 0b00001101;                    // ANalog on RC0, RC2, RC3
    WPUC       = 0x00;                          // defaults on
    ANSELA     = 0b00000000;                    // reset sets them ON!

    HBLED=DAVLED=ERRLED=1;                      // all LEDs ON
    for(i=0;i<NF;i++){                          // for each frequency
            Zmag[i]=0.0;                        // zero the data
            Zpha[i]=0.0;
    }

    stimin=2*NF; stimax=0.00;
    for(tmodulo=0;tmodulo<PERIOD1;tmodulo+=33){         // for each time point @3Hz
            dtmp2=(0.01*TWOPI)*tmodulo;                         // 2.pi.t
            for(sumOfSines=0.0,i=0;i<NF;i++){                   // for each tone
                    dtmp=dtmp2*freq[i];                                 // 2*pi*
                    sumOfSines+=sin(dtmp+NF);                   // sum sin(2.pi.
            }
            sumOfSines+=NF;                                             // now b
            if(sumOfSines>stimax){stimax=sumOfSines;}   // find max reached
            if(sumOfSines<stimin){stimin=sumOfSines;}   // find min reached
    }

    GIE=1;                                              // start ISR, hand over

    msg[0]='B';
```

```
        msg[1]='0';
        msg[2]='0';
        msg[3]='T';
        msg[4]='\n';
        msg[5]=0;
        TXIE=TRUE;

        // main loop tasks:
        // measure Vref to find Vsupply value
        // grab timestamp
        // measure current (which is not changing so timing not critical)
        // measure temperature, check for big shifts
        // compute and set current using PWM output
        // compute window multiplier
        // integrate each Fourier segment, 1 per frequency
        // check if N cycles done, update output variables
        // if serial enabled and buffer empty, refill

    while(!reboot){
                nseq++;                                                 // count points
                VSELECT;                                                // point ADC to
                sync=0;                                                 // clear ISR syn
                while(!sync){;}                                     // wait until time is mu
                GO_nDONE=1;                                             // start ADC
                GIE=0;tstamp=t10ms;GIE=1;                       // atomic grab of timestamp
                while(GO_nDONE){;}
                adc=ADRESH*256+ADRESL;                          // reading into unsigned int
                ISELECT;                                                // point ADC to
                prevVb=VbmV;                                        // current to previous v
                VbmV=(1024L*VREF)/adc;                          // VREF=VbmV*(adc/1024);
                GO_nDONE=1;                                             // start reading
                if(VbmV<Vmin){Vmin=VbmV;}                       // capture min
                if(VbmV>Vmax){Vmax=VbmV;}                       // capture max
                if(VbmV<VERROR)badvoltage=TRUE;         // flat error
                if(Vmax-Vmin>150)badvoltage=TRUE;       // deltaV error
                prevIb=IbcuA;                                           // store previous
                while(GO_nDONE){;}
                adc=ADRESH*256+ADRESL;                          // I reading into unsigned int
                TSELECT;                                                // point ADC to
                IbcuA=(short)(-10L*adc*VbmV/1024L/RS); // scale battery current
                GO_nDONE=1;                                             // read Temperat
                while(GO_nDONE){;}
                adc=ADRESH*256+ADRESL;                          // T reading, ~12mV/C
                degC=(VbmV*adc/1024L);                          // now in mV
                degC=(int)((1866.3-degC)/11.69);        // T ~ (1.8663-V0)/11.69e-3;
                if(degC<Tmin){Tmin=degC;}
                if(degC>Tmax){Tmax=degC;}
                if(Tmax-Tmin>dTERROR){overDeltaT=TRUE;}

                // preserve precision by using modulo time in trig calculations
                tmodulo=tstamp;
                cycle=1;
                while(tmodulo>=PERIOD1){
                        tmodulo-=PERIOD1;
                        cycle++;
```

```
}

// update current drive
stimcalc=TRUE;
dtmp2=(10e-3*TWOPI)*tmodulo;                                    // 2.pi.t
for(sumOfSines=0.0,i=0;i<NF;i++){                               // for each tone
        dtmp=dtmp2*freq[i];
        sumOfSines+=sin(dtmp+NF);                                       // sum o
}
sumOfSines += NF;

#define OFFSET 130
sumOfSines -= stimin;                                                   // now 0
sumOfSines /= (stimax-stimin);                                  // now 0 to 1
sumOfSines *= (1024-OFFSET);                                    // now 0 to (102
pwm = (signed short int)sumOfSines;                             //
pwm += OFFSET;

if(pwm>1023){pwm=1023;}                                                 // shoul
if(pwm<OFFSET){pwm=OFFSET;}                                             // shoul
CCPR1L=(char)(pwm>>2);                                                  // pwm M
if(pwm&0x02){DC1B1=1;}else{DC1B1=0;}                    // pwm 2LSB
if(pwm&0x01){DC1B0=1;}else{DC1B0=0;}                    // pwm LSB
stimcalc=FALSE;

// integrate each Fourier segment
if(tstamp>(PERIOD1*NCYCLES)){                           // gone past the end
        // integrate last chunk
        tstamp=(PERIOD1);                                               // about
        for(i=0;i<NF;i++){
                integrate(ReVsum,ImVsum,i,tstamp,prevtstamp,0.00,prevVbwin);
                integrate(ReIsum,ImIsum,i,tstamp,prevtstamp,0.00,prevIbwin);
        }
        for(i=0;i<NF;i++){                                              // outpu
                dtmp  =(ReVsum[i]*ReVsum[i]);
                dtmp +=(ImVsum[i]*ImVsum[i]);
                dtmp2 =sqrt(dtmp);                                      // sqrt(
                dtmp  =(ReIsum[i]*ReIsum[i]);
                dtmp +=(ImIsum[i]*ImIsum[i]);
                dtmp  =sqrt(dtmp);                                      // sqrt(
                Zmag[i] =10.0*dtmp2/dtmp;                       // |Z| in "mV/(1
                dtmp =atan2(ImVsum[i],ReVsum[i]);
                dtmp-=atan2(ImIsum[i],ReIsum[i]);       // dtmp is phase differe
                dtmp/=TWOPI;
                dtmp*=360.0;                                            // arg(Z
                while(dtmp<0.0)dtmp+=360.0;                     // range 0->360
                Zpha[i]=dtmp;                                           // pos a
                dav=TRUE;
        }
        nseq=0;
        overDeltaT=FALSE;                                               // clear
        badvoltage=FALSE;
}
if(nseq==0){                                                            // start
        GIE=0;t10ms=0L;GIE=1;                                   // clear master
```

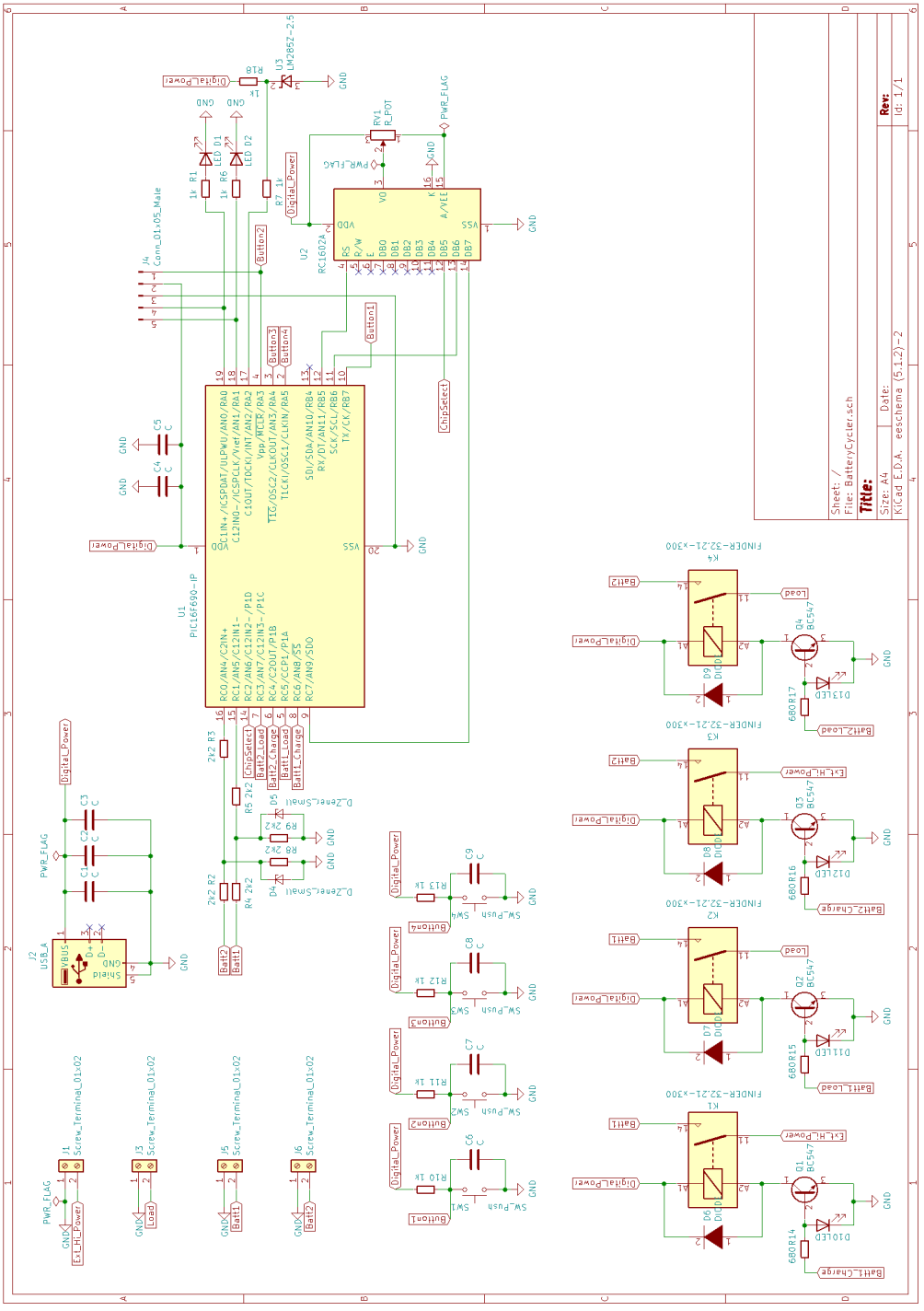
```
                        tstamp=prevtstamp=0;                                    // clear timesta
                        prevVbwin=0.00;                                                 // set p
                        prevIbwin=0.00;                                                 // zero
                        for(i=0;i<NF;i++){                                              // for e
                                ReVsum[i]=0.00;                                         // clear
                                ReIsum[i]=0.00;
                                ImVsum[i]=0.00;
                                ImIsum[i]=0.00;
                        }
                }else{
                        // compute Hann window multiplier, across whole duration, PERIOD1*NCYCLE
                        win = (double)tstamp/(PERIOD1*NCYCLES);
                        win *= TWOPI;
                        win = 1.00-cos(win);                                            // Hann
                        prevVbwin=Vbwin;
                        prevIbwin=Ibwin;
                        Vbwin = (VbmV-NOMMV)*win;                                       // subtr
                        Ibwin = (IbcuA+(5*NOMMV/RS))*win;                       // average aroun
                        // offset around zero reduces large values in float accumulators
                        // now we have 2 windowed tvi data, prev & new, for I & V, at NF frequen
                        for(i=0;i<NF;i++){
                                integrate(ReVsum,ImVsum,i,tmodulo,prevtstamp,Vbwin,prevVbwin);
                                integrate(ReIsum,ImIsum,i,tmodulo,prevtstamp,Ibwin,prevIbwin);
                        }       // have now integrated across one slice of the V & I waveforms
                        prevtstamp=tmodulo;
                }

                // want to send header, freq/mag/pha
                if(serial && TXIE==FALSE){                              // sending data but buff
                        if(++nvar>=NF*3)nvar=0;                         // send 3 numbers/freque
                        i=(unsigned)(nvar/3);                           // the frequency index
                        switch(nvar%3){                                         // the freq/mag/
                                default:
                                case 0:
                                        ltmp=(long)(1e6*freq[i]);                       // send
                                        break;
                                case 1:
                                        ltmp=(long int)(1e3*Zmag[i]);           // send mOhms
//ltmp=(long int)(fabs(ReVsum[i])+0.5); // diagnostic view of V accumulators
//ltmp=(long int)(fabs(ReIsum[i])+0.5); // diagnostic view of I accumulators
//ltmp=(long int)VbmV;  // diagnostic voltage check
                                        break;
                                case 2:
                                        ltmp=(long int)(Zpha[i]+0.5);                   // send
// ltmp=(long int)(fabs(ImVsum[i])+0.5);        // diagnostic view Im V accumulators
// ltmp=(long int)(fabs(ImIsum[i])+0.5);        // diagnostic view Im V accumulators
//ltmp=(long int)(tmodulo);     // diagnostic readout of tmodulo
//ltmp=(long int)IbcuA; // diagnostic current check
                                        break;
                        }
                        ltomsg(ltmp,msg);                                       // fill msg with
                        if(nvar%3==0)msg[0]='u';                        // frequency has leading
                        TXIE=TRUE;                                              // enabl
                }
```

```
// this code modelled on NIXIE project... too big to fit in 1825
//#define SECSSOFAR 0
//#define PROGRESS 1
//#define VMINMAX 2
//#define IMINMAX 3
//              if(serial && TXIE==FALSE){                              // sending data but buff
//                      if(++msgline>NF+4){msgline=0;}  // which message to send
//                      switch(msgline){
//                              case SECSSOFAR:
//                                      strcpy(msg,"time ");
//                                      ltomsg(tstamp,msg+5);
//                                      break;
//                              case PROGRESS:
//                                      strcpy(msg,"% complete ");
//                                      ltmp=100*tstamp/(PERIOD1*NCYCLES);
//                                      ltomsg(ltmp,msg+11);
//                                      break;
//                              case VMINMAX:
//                                      strcpy(msg,"Vrange ");
//                                      ltmp=Vmin;
//                                      ltomsg(ltmp,msg+7);
//                                      strcat(msg,"/");
//                                      ltmp=Vmax;
//                                      ltomsg(ltmp,msg+strlen(msg));
//                                      break;
//                              case IMINMAX:
//                                      strcpy(msg,"Trange ");
//                                      ltmp=Tmax-Tmin;
//                                      ltomsg(ltmp,msg+7);
//                                      break;
//                              default:
//                                      i=(msgline-4);
//                                      ltmp=1e6*freq[i];
//                                      ltomsg(ltmp,msg);
//                                      strcat(msg,"uHz\n");
//                                      ltmp=(long int)(1000.0*Zmag[i]);
//                                      ltomsg(ltmp,msg+strlen(msg));
//                                      strcat(msg,"mR <");
//                                      ltmp=(long int)Zpha[i];
//                                      ltomsg(ltmp,msg+strlen(msg));
//                                      break;
//                      }
//                      TXIE=TRUE;                                                              // resta
//              }

    }

}
// end of file
```

Appendix C

Single Board Battery Cycling

C.1 Schematic



C.2 Code Listing

C.2.1 Main

```
/*
 * File:   BatteryCycler.c
 * Author: Vance Farrow
 *
 * Combine reads into a squence of read vref read batt convert
 *
 */

// CONFIG
#pragma config FOSC = INTRCIO   // Oscillator Selection bits (INTOSCIO oscillator: I/O function
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled and can be enabled by
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT disabled)
#pragma config MCLRE = OFF      // MCLR Pin Function Select bit (MCLR pin function is digital in
#pragma config CP = OFF         // Code Protection bit (Program memory code protection is disabl
#pragma config CPD = OFF        // Data Code Protection bit (Data memory code protection is disa
#pragma config BOREN = OFF      // Brown-out Reset Selection bits (BOR disabled)
#pragma config IESO = ON        // Internal External Switchover bit (Internal External Switchove
#pragma config FCMEN = ON       // Fail-Safe Clock Monitor Enabled bit (Fail-Safe Clock Monitor

// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.

#include <xc.h>

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdint.h>

#include "Util.h"
#include "LCD.h"

void __interrupt() ISR(void) {
    if (T0IF) { // ~1ms ticks
        T0IF = 0;
        if (b1timeoutstart) {
            b1timeout++;
        }
        if (b2timeoutstart) {
            b2timeout++;
        }
    } // T0IF

    if (TMR1IF) {
        TMR1IF = 0;
        refeshdisplay = 1;
        TMR1 = 15535; // leave 50ms in the timer, refresh rate 20Hz
        if (!StartStop) {
            if (startstoplast)
                startstopcount++;
```

```
        if (startstopcount >= 250) // prevent overflow, limit to 250
            startstopcount = 250;
        startstoplast = 1;
    } else {
        if (!startstoplast) { // Extra bounce protection for important function
            startstoplatch = 0;
            startstopcount = 0;
        }
        startstoplast = 0;
    }
    if (!Increase) {
        if (increaselast)
            increasecount++;
        if (increasecount >= 250) // prevent overflow, limit to 250
            increasecount = 250;
        increaselast = 1;
    } else {
        increasecount = 0;
        increaselast = 0;
    }
    if (!Decrease) {
        if (decreaselast)
            decreasecount++;
        if (decreasecount >= 250) // prevent overflow, limit to 250
            decreasecount = 250;
        decreaselast = 1;
    } else {
        decreasecount = 0;
        decreaselast = 0;
    }
    if (!Mode) {
        if (modelast)
            modecount++;
        if (modecount >= 250) // prevent overflow, limit to 250
            modecount = 250;
        modelast = 1;
    } else {
        if (!modelast) { // Extra bounce protection for important function
            modelatch = 0;
            modecount = 0;
        }
        modelast = 0;
    }

    if (Run == STOPSTATE) { // only allow modification of important parameters when stopped
        if (Input == VMAX) {
            updown(&voltmax);
        }
        if (Input == VMIN) {
            updown(&voltmin);
        }
        if (Input == CYCLES) {
            updown(&ncycles);
        }
        if (Input == RATIO) {
```

```
                updown(&inputratio);
            }
        }
    } // TMR1IF

    //    if (TMR2IF) {  // timer 2 used to time LCD commands
    //        TMR2IF = 0;
    //    } // TMR2IF

}

uint32_t adc2mvolt(uint32_t tmp) {
    uint32_t tmpvolt = 0;
    tmpvolt = tmp * (10000); // multiply by 2.5 for vref and 10000 for input ratio
    tmpvolt = tmpvolt / (uint32_t) vref;
    tmpvolt = tmpvolt * 2500;
    tmpvolt = tmpvolt / (uint32_t) inputratio;
    return tmpvolt;
}

/*
 *
 */
int main(void) {

    PORTA = 0x00;
    PORTB = 0x00;
    PORTC = 0x00;
    init();
    waitms(20);
    LCDclear();
    LCDsend(' ', 1); // prime the SPI bus
    LCDinit(); // turn on the LCD.

    firststart = readEEPROM(ADDR_FIRSTSTART);
    GIE = 0;
    if (firststart != 0xFF) { // recover previous state after shutdown

        IntToBytes.unsigned8[0] = readEEPROM(ADDR_VMAX1);
        IntToBytes.unsigned8[1] = readEEPROM(ADDR_VMAX2);
        voltmax = IntToBytes.unsigned16;

        IntToBytes.unsigned8[0] = readEEPROM(ADDR_VMIN1);
        IntToBytes.unsigned8[1] = readEEPROM(ADDR_VMIN2);
        voltmin = IntToBytes.unsigned16;

        IntToBytes.unsigned8[0] = readEEPROM(ADDR_CYCLES1);
        IntToBytes.unsigned8[1] = readEEPROM(ADDR_CYCLES2);
        ncycles = IntToBytes.unsigned16;

        IntToBytes.unsigned8[0] = readEEPROM(ADDR_INPUTRATIO1);
        IntToBytes.unsigned8[1] = readEEPROM(ADDR_INPUTRATIO2);
        inputratio = IntToBytes.unsigned16;

        batt2en = readEEPROM(ADDR_B2EN);
```

```
        Run = readEEPROM(ADDR_RUN);

        B1State = readEEPROM(ADDR_B1ST);
        B2State = readEEPROM(ADDR_B2ST);

        IntToBytes.unsigned8[0] = readEEPROM(ADDR_B1CYCLES1);
        IntToBytes.unsigned8[1] = readEEPROM(ADDR_B1CYCLES2);
        cyclesb1 = IntToBytes.unsigned16;
        IntToBytes.unsigned8[0] = readEEPROM(ADDR_B2CYCLES1);
        IntToBytes.unsigned8[1] = readEEPROM(ADDR_B2CYCLES2);
        cyclesb2 = IntToBytes.unsigned16;

        if(B1State == CHARGING)
        {
            BATT1_POWER = 1;
            nop(); // wait 1 clock cycle to ensure output changes successfully
        }
        if(B2State == CHARGING)
        {
            BATT2_POWER = 1;
            nop();
        }
        if(B1State == DISCHARGING)
        {
            BATT1_LOAD = 1;
            nop();
        }
        if(B2State == DISCHARGING)
        {
            BATT2_LOAD = 1;
            nop();
        }

    }

    GIE = 1;

    for (;;) {
        if (refeshdisplay) {
            LCDmovecursor(1, 0);
            LCDsendline(line1);
            LCDnextline();
            LCDsendline(line2);
            refeshdisplay = 0;
        }

        if (Run == STOPSTATE) {
            if ((startstopcount > 60) && (startstoplatch == 0) && (modecount == 0)) {
                Run = GOSTATE;
                writeEEPROM(Run, ADDR_RUN);
                writeEEPROM(0x00, ADDR_FIRSTSTART);


                IntToBytes.unsigned16 = voltmax;
                writeEEPROM(IntToBytes.unsigned8[0], ADDR_VMAX1);
```

```
        writeEEPROM(IntToBytes.unsigned8[1], ADDR_VMAX2);

        IntToBytes.unsigned16 = voltmin;
        writeEEPROM(IntToBytes.unsigned8[0], ADDR_VMIN1);
        writeEEPROM(IntToBytes.unsigned8[1], ADDR_VMIN2);


        IntToBytes.unsigned16 = ncycles;
        writeEEPROM(IntToBytes.unsigned8[0], ADDR_CYCLES1);
        writeEEPROM(IntToBytes.unsigned8[1], ADDR_CYCLES2);

        IntToBytes.unsigned16 = inputratio;
        writeEEPROM(IntToBytes.unsigned8[0], ADDR_INPUTRATIO1);
        writeEEPROM(IntToBytes.unsigned8[1], ADDR_INPUTRATIO2);

        writeEEPROM(batt2en, ADDR_B2EN);

        cyclesb1 = ncycles;
        cyclesb2 = ncycles;

        IntToBytes.unsigned16 = cyclesb1;
        writeEEPROM(IntToBytes.unsigned8[0], ADDR_B1CYCLES1);
        writeEEPROM(IntToBytes.unsigned8[1], ADDR_B1CYCLES2);

        IntToBytes.unsigned16 = cyclesb2;
        writeEEPROM(IntToBytes.unsigned8[0], ADDR_B2CYCLES1);
        writeEEPROM(IntToBytes.unsigned8[1], ADDR_B2CYCLES2);

        startstoplatch = 1;
        LED1 = 0;
        LED2 = 1;
        B1State = TOCHARGE;
        B2State = TOCHARGE;
        writeEEPROM(B1State, ADDR_B1ST);
        writeEEPROM(B2State, ADDR_B2ST);
    }
    if (modecount > 4) {
        if ((Input == VMAX) && !modelatch) {
            Input = VMIN;
            modelatch = 1;
        }
        if ((Input == VMIN) && !modelatch) {
            Input = CYCLES;
            modelatch = 1;
        }
        if ((Input == CYCLES) && !modelatch) {
            Input = B2EN;
            modelatch = 1;
        }
        if ((Input == B2EN) && !modelatch) {
            Input = RATIO;
            modelatch = 1;
        }
        if ((Input == RATIO) && !modelatch) {
            Input = VMAX;
```

```
            modelatch = 1;
        }
    }

    if (startstopcount > 100 && modecount > 100 && increasecount > 100 && decreasecount
        voltmax = 4200;
        voltmin = 3000;
        ncycles = 20;
        inputratio = 8696;
    }

    if (Input == VMAX) {
        memset(line1, ' ', 16);
        nchar = sprintf(line1, "Maximum Voltage");
        line1[nchar] = ' ';

        uint16_t tmp1 = voltmax / 1000;
        uint16_t tmp2 = voltmax - tmp1 * 1000;
        memset(line2, ' ', 16);
        nchar = sprintf(line2, "%d.%03dV", tmp1, tmp2);
        line2[nchar] = ' ';

    }
    if (Input == VMIN) {
        memset(line1, ' ', 16);
        nchar = sprintf(line1, "Minimum Voltage");
        line1[nchar] = ' ';

        uint16_t tmp1 = voltmin / 1000;
        uint16_t tmp2 = voltmin - tmp1 * 1000;
        memset(line2, ' ', 16);
        nchar = sprintf(line2, "%d.%03dV", tmp1, tmp2);
        line2[nchar] = ' ';

    }
    if (Input == CYCLES) {
        memset(line1, ' ', 16);
        nchar = sprintf(line1, "Number of Cycles");
        line1[nchar] = ' ';

        memset(line2, ' ', 16);
        nchar = sprintf(line2, "%d", ncycles);
        line2[nchar] = ' ';
    }
    if (Input == B2EN) {
        if (increasecount > 0) {
            batt2en = 1;
        }
        if (decreasecount > 0) {
            batt2en = 0;
        }

        memset(line1, ' ', 16);
        nchar = sprintf(line1, "Enable Battery 2");
        line1[nchar] = ' ';
```

```
                memset(line2, ' ', 16);
                if (batt2en) {
                    nchar = sprintf(line2, "enabled");
                } else {
                    nchar = sprintf(line2, "disabled");
                }
                line2[nchar] = ' ';
            }
            if (Input == RATIO) {
                memset(line1, ' ', 16);
                nchar = sprintf(line1, "Input Ratio");
                line1[nchar] = ' ';

                memset(line2, ' ', 16);
                nchar = sprintf(line2, "0.%04d", inputratio);
                line2[nchar] = ' ';
            }

        }
        if (Run == GOSTATE) {
            {
//                ANS3 = 1; // Set VREF/Decrease pin as analog input when making a measurement V
                mvoltb1 = 0;
                mvoltb2 = 0;
                uint32_t tmp = 0;


                tmp = 0;
                vref = 0;
                for (int i = 0; i < 16; i++) {
                    ADCON0 = (ADCON0 & 0b11000011) | (0x02 << 2); // select AN2 VREF pin as anal
                    nop(); // ensure settling
                    nop(); // ensure settling
                    nop(); // ensure settling
                    nop(); // ensure settling
                    nop(); // ensure settling
                    GO_nDONE = 1;
                    while (GO_nDONE) {
                        ;
                    }
                    vref += (ADRESH << 8) | (ADRESL);

                    ADCON0 = (ADCON0 & 0b11000011) | (0x04 << 2); // Select battery 2 as next an
                    nop(); // ensure settling
                    nop(); // ensure settling
                    nop(); // ensure settling
                    nop(); // ensure settling
                    nop(); // ensure settling
                    GO_nDONE = 1;
                    while (GO_nDONE) {
                        ;
                    }
                    tmp += (ADRESH << 8) | (ADRESL);
                }
```

```
                mvoltb2 = adc2mvolt(tmp); // division by 16 already done by vref summed 16 times

                tmp = 0;
                vref = 0;
                for (int i = 0; i < 16; i++) {
                    ADCON0 = (ADCON0 & 0b11000011) | (0x02 << 2); // select AN2 VREF pin as anal
                    nop(); // ensure settling
                    nop(); // ensure settling
                    nop(); // ensure settling
                    nop(); // ensure settling
                    nop(); // ensure settling
                    GO_nDONE = 1;
                    while (GO_nDONE) {
                        ;
                    }
                    vref += (ADRESH << 8) | (ADRESL);

                    ADCON0 = (ADCON0 & 0b11000011) | (0x05 << 2); // Select battery 1 as next an
                    nop(); // ensure settling
                    nop(); // ensure settling
                    nop(); // ensure settling
                    nop(); // ensure settling
                    nop(); // ensure settling
                    GO_nDONE = 1;
                    while (GO_nDONE) {
                        ;
                    }
                    tmp += (ADRESH << 8) | (ADRESL);
                }
                mvoltb1 = adc2mvolt(tmp);
//                ANS3 = 0;
            }

            if (!batt2en) {
                B2State = FINISH;
            }
            if (((B1State == FINISH) && (B2State == FINISH)) || ((startstopcount > 60) && (start
                Run = STOPSTATE;
                writeEEPROM(Run, ADDR_RUN);
                startstoplatch = 1;
                BATT1_POWER = 0;
                nop(); // wait 1 clock cycle to ensure output changes successfully
                BATT2_POWER = 0;
                nop();
                BATT1_LOAD = 0;
                nop();
                BATT2_LOAD = 0;
                LED1 = 1;
                LED2 = 0;
            }

            if (mvoltb1 >= voltmax && B1State == CHARGING) {
                b1timeoutstart = 1;
                if (b1timeout > 10000) { // timeout at end state for 10 seconds
                    B1State = TODISCHARGE;
```

```
            BATT1_POWER = 0;
            cyclesb1 -= 1;
            IntToBytes.unsigned16 = cyclesb1;
            writeEEPROM(IntToBytes.unsigned8[0], ADDR_B1CYCLES1);
            writeEEPROM(IntToBytes.unsigned8[1], ADDR_B1CYCLES2);

            if (cyclesb1 <= 0) {
                B1State = FINISH;
            }
            b1timeoutstart = 0;
            b1timeout = 0;
        }
    }
    if (mvoltb2 >= voltmax && B2State == CHARGING) {
        b2timeoutstart = 1;
        if (b2timeout > 10000) { // timeout at end state for 10 seconds
            B2State = TODISCHARGE;
            BATT2_POWER = 0;
            cyclesb2 -= 1;
            IntToBytes.unsigned16 = cyclesb2;
            writeEEPROM(IntToBytes.unsigned8[0], ADDR_B2CYCLES1);
            writeEEPROM(IntToBytes.unsigned8[1], ADDR_B2CYCLES2);

            if (cyclesb2 <= 0) {
                B2State = FINISH;
            }
            b2timeoutstart = 0;
            b2timeout = 0;

        }
    }
    if (mvoltb1 <= voltmin && B1State == DISCHARGING) {
        b1timeoutstart = 1;
        if (b1timeout > 10000) { // timeout at end state for 10 seconds
            B1State = TOCHARGE;
            BATT1_LOAD = 0;
            b1timeoutstart = 0;
            b1timeout = 0;
        }
    }
    if (mvoltb2 <= voltmin && B2State == DISCHARGING) {
        b2timeoutstart = 1;
        if (b2timeout > 10000) { // timeout at end state for 10 seconds
            B2State = TOCHARGE;
            BATT2_LOAD = 0;
            b2timeoutstart = 0;
            b2timeout = 0;
        }
    }
    if ((B1State == TOCHARGE) && (B2State != CHARGING)) {
        B1State = CHARGING;
        BATT1_POWER = 1;
    }
    if ((B2State == TOCHARGE) && (B1State != CHARGING)) {
        B2State = CHARGING;
```

```
            BATT2_POWER = 1;
        }
        if ((B1State == TODISCHARGE) && (B2State != DISCHARGING)) {
            B1State = DISCHARGING;
            BATT1_LOAD = 1;
        }
        if ((B2State == TODISCHARGE) && (B1State != DISCHARGING)) {
            B2State = DISCHARGING;
            BATT2_LOAD = 1;
        }

        { // push scope when using temp local variables
            memset(line1, ' ', 16);
            nchar = sprintf(line1, "Cycle: %05d", cyclesb1);
            line1[nchar] = ' ';

            uint16_t tmp1 = mvoltb1 / 1000;
            uint16_t tmp2 = mvoltb1 - tmp1 * 1000;
            tmp2 = tmp2 / 10;
            uint16_t tmp3 = mvoltb2 / 1000;
            uint16_t tmp4 = mvoltb2 - tmp3 * 1000;
            tmp4 = tmp4 / 10;

            memset(line2, ' ', 16);
            nchar = sprintf(line2, "V1 %d.%02d V2 %d.%02d ", tmp1, tmp2, tmp3, tmp4);
            line2[nchar] = ' ';
        }

        if (B1State != readEEPROM(ADDR_B1ST)) {
            writeEEPROM(B1State, ADDR_B1ST);
        }
        if (B2State != readEEPROM(ADDR_B2ST)) {
            writeEEPROM(B2State, ADDR_B2ST);
        }
    }// Run == GOSTATE
  } // main loop
} // main()
```

C.2.2 LCD

```
/*
 * File:   LCD.h
 * Author: vance
 *
 * Created on 28 May 2020, 4:46 PM
 */

#ifndef LCD_H
#define LCD_H

#ifdef  __cplusplus
extern "C" {
#endif
```

```
    void LCDsendline(char line[]);
    void LCDsend(char byte, uint8_t rs);
    void LCDclear(void);
    void LCDnextline(void);
    void LCDmovecursor(int row, int col);
    void LCDinit(void);

#ifdef  __cplusplus
}
#endif

#endif  /* LCD_H */
```

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdint.h>
#include <xc.h>

#include "LCD.h"
#include "Util.h"

void LCDsendline(char line[]) {
    for (int8_t i = 0; i < 16; i++) {
        LCDsend(line[i], 1);
    }
}

void LCDsend(char byte, uint8_t rs) {
    RS = rs;
    CHIPSELECT = 0;

    WCOL = 0;
    SSPBUF = byte;
    waitus(40);

    CHIPSELECT = 1;
    RS = 0;
}

void LCDclear(void) {
    LCDsend(0x01, 0);
    waitms(2);
}

void LCDnextline(void) {
    LCDsend(0xC0, 0);
}

void LCDmovecursor(int row, int col) {
    uint8_t addr = 0x80;
    addr += col;
```

```
    if (row == 2) {
        addr += 0x40;
    }

    LCDsend(addr, 0);
}

void LCDinit(void) {
    waitms(1000); // wait to ensure LCD is idle

    //    0b001(DL)NFXX
    LCDsend(0x38, 0); // Function Set
    waitus(250);

    //    0b00001DCB
    LCDsend(0x0D, 0); // Display ON/OFF
    waitus(120);

    LCDsend(0x01, 0); // Clear Display
    waitms(25);

    //    0b000001(I/D)S
    LCDsend(0x06, 0); // Entry Mode Set
    waitms(20);
}
```

C.2.3 Util

```
/*
 * File:   Util.h
 * Author: vance
 *
 * Created on 28 May 2020, 4:45 PM
 */

#ifndef UTIL_H
#define UTIL_H

#define LED1 RA0 // RA0
#define LED2 RA1 // RA1
#define LED3 RA2 // RA2 VERSION 1 ONLY

#define VREF RA4 // RA4/AN3 VERSION 1 ONLY // RA2/AN2

#define StartStop RB7 // RB7
#define Increase RA3 // RA3
#define Decrease RA4 // RA4
#define Mode RA5 // RA5

//// PIN ORDER FOR VERSION 1 ONLY
//#define BATT2_LOAD RC3 // RC3
//#define BATT2_POWER RC4 // RC4
//#define BATT_1LOAD RC5 // RC5
```

```
//#define BATT1_POWER RC6 // RC6

// PIN ORDER FOR V3+
#define BATT1_POWER RC3 // RC3
#define BATT1_LOAD RC4 // RC4
#define BATT2_LOAD RC5 // RC5
#define BATT2_POWER RC6 // RC6

#define CHIPSELECT RC2 // RC2
#define RS RB5 // RB5

#define nop() {asm("NOP");} // Nicer looking NOP assembly instruction

//states for state machines

enum BState {
    TOCHARGE,
    TODISCHARGE,
    CHARGING,
    DISCHARGING,
    FINISH
};

enum StartState {
    STOPSTATE,
    GOSTATE,
    FIRSTSTARTSTATE
};

enum InputState {
    VMAX,
    VMIN,
    CYCLES,
    B2EN,
    B1EN,
    RATIO,
    TIMEOUT,

};

// Globals
volatile int8_t startstoplast = 0;
volatile int8_t increaselast = 0;
volatile int8_t decreaselast = 0;
volatile int8_t modelast = 0;

volatile uint8_t startstopcount = 0;
volatile uint8_t increasecount = 0;
volatile uint8_t decreasecount = 0;
volatile uint8_t modecount = 0;

volatile uint8_t startstoplatch = 0;
volatile uint8_t modelatch = 0;

volatile enum BState B1State = TOCHARGE;
```

```
volatile enum BState B2State = TOCHARGE;

volatile enum InputState Input = VMAX;
volatile enum StartState Run = STOPSTATE;

volatile int8_t batt1en = 1;
volatile int8_t batt2en = 0; // default single battery operation

volatile uint16_t b1timeout = 0;
volatile int8_t b1timeoutstart = 0;
volatile uint16_t b2timeout = 0;
volatile int8_t b2timeoutstart = 0;

volatile uint16_t voltmax = 13800;
volatile uint16_t voltmin = 10800; // default voltage limits for a lithium 18650
volatile uint16_t ncycles = 20;
volatile uint16_t inputratio = 3190; // 2200/2200+330

volatile uint8_t firststart = 0xFF;


volatile uint32_t vref = 0;
volatile uint32_t mvoltb1 = 0;
volatile uint32_t mvoltb2 = 0;

volatile uint16_t cyclesb1 = 0;
volatile uint16_t cyclesb2 = 0;

volatile uint8_t refeshdisplay = 1;

char line1[20] = {' '};
char line2[20] = {' '};
volatile int8_t nchar = 0;

union {
    volatile uint16_t unsigned16;
    volatile uint8_t unsigned8[2];
} IntToBytes;

#define ADDR_FIRSTSTART 0x00
#define ADDR_RUN 0x01
#define ADDR_VMAX1 0x04
#define ADDR_VMAX2 0x05
#define ADDR_VMIN1 0x06
#define ADDR_VMIN2 0x07
#define ADDR_CYCLES1 0x08
#define ADDR_CYCLES2 0x09
#define ADDR_INPUTRATIO1 0x0A
#define ADDR_INPUTRATIO2 0x0B
#define ADDR_B2EN 0x0C
#define ADDR_B1CYCLES1 0x10
#define ADDR_B1CYCLES2 0x11
#define ADDR_B2CYCLES1 0x12
#define ADDR_B2CYCLES2 0x13
#define ADDR_B1ST 0x14
```

```
#define ADDR_B2ST 0x15



// // Useful debugging tools
//union {
//    int32_t signed32;
//    uint32_t unsigned32;
//    int16_t signed16[2];
//    uint16_t unsigned16[2];
//    int8_t signed8[4];
//    uint8_t unsigned8[4];
//} LongIntToBytes;
//
//#define BYTE_TO_BINARY_PATTERN "%c%c%c%c%c%c%c%c"
//#define BYTE_TO_BINARY(byte)  \
//  (byte & 0x80 ? '1' : '0'), \
//  (byte & 0x40 ? '1' : '0'), \
//  (byte & 0x20 ? '1' : '0'), \
//  (byte & 0x10 ? '1' : '0'), \
//  (byte & 0x08 ? '1' : '0'), \
//  (byte & 0x04 ? '1' : '0'), \
//  (byte & 0x02 ? '1' : '0'), \
//  (byte & 0x01 ? '1' : '0')
//
//sprintf(line1, BYTE_TO_BINARY_PATTERN BYTE_TO_BINARY_PATTERN, BYTE_TO_BINARY(byte1), BYTE_TO_B
#ifdef  __cplusplus
extern "C" {
#endif


    void updown(volatile uint16_t* num);
    void waitms(uint16_t ms);
    void waitus(uint16_t us);
    uint8_t readEEPROM(uint8_t address);
    uint8_t writeEEPROM(uint8_t data, uint8_t address);
    void init(void);

#ifdef  __cplusplus
}
#endif

#endif  /* UTIL_H */
```

```
#include <stdint.h>
#include <xc.h>
#include "Util.h"

void updown(volatile uint16_t* num) {
    uint16_t tmp = *num;

    if ((increasecount > 0) && (increasecount < 60)) {
        *num += 1;
    }
    if ((increasecount > 59) && (increasecount < 200)) {
```

```
        *num += 5;
    }
    if ((increasecount > 199)) {
        *num += 10;
    }
    if ((decreasecount > 0) && (decreasecount < 60)) {
        *num -= 1;
    }
    if ((decreasecount > 59) && (decreasecount < 200)) {
        *num -= 5;
    }
    if ((decreasecount > 199)) {
        *num -= 10;
    }


    if (decreasecount && (tmp < *num)) {
        *num = 0;
    }
    if (increasecount && (tmp > *num)) {
        *num = 65535;
    }
}

void waitms(uint16_t ms) {
    while (ms > 0) {
        TMR2IF = 0;
        TMR2 = 0x00;
        TMR2ON = 1;
        while (!TMR2IF) {
            ;
        } // wait for LCD instruction to finish
        ms--;
    }
    TMR2ON = 0;
    TMR2 = 0x00;
    TMR2IF = 0;
}

void waitus(uint16_t us) {
    if (us > 1000) {
        waitms(us / 1000);
    } else {
        us /= 4;
        TMR2IF = 0;
        TMR2 = 0x00;
        TMR2ON = 1;
        while (TMR2 < us) {
            ;
        }
    }
}

// reads from EEPROM at address, returns read data
```

```
uint8_t readEEPROM(uint8_t address) {
    EEADR = address;
    EEPGD = 0; // read data not program memory
    RD = 1; // start EEPROM read
    nop();
    return EEDAT;
}

// writes data value to EEPROM address, returns written data

uint8_t writeEEPROM(uint8_t data, uint8_t address) {
    GIE = 0; // prevent interrupts while writing EEPROM
    EEADR = address; // EEPROM address to write to
    EEDAT = data; // data to write to EEPROM
    // required sequence to start EEPROM write: section 10.3 PIC16F684 datasheet
    EEPGD = 0; // write data not program memory
    WREN = 1; // enable EEPROM write

    EECON2 = 0x55; // Unlock EEPROM with 0x55 then 0xAA
    EECON2 = 0xAA;
    WR = 1;
    while (WR) {
        ; // wait for EEPROM to finish write
    }
    WREN = 0; // disable EEPROM write
    GIE = 1; // enable interrupts
    return readEEPROM(address); // return value at address
}

void init(void) {
    //IRCF0 = 1; // Fosc 8MHz to get 500kHz clock on LCD bus

    // OPTION_REG
    nRABPU = 1; // PORTA/PORTB pullup disabled
    INTEDG = 1; // Interrupt on RA2 rising edge
    T0CS = 0; // Timer0 clock source Fosc/4
    T0SE = 0; // Timer0 edge select NOT USED
    PSA = 0; // Assign prescaler to Timer0
    PS2 = 0; // Set prescaler to 4 to get a 4us resolution TMR0
    PS1 = 0; // ~1ms range
    PS0 = 1; //

    // T1CON
    T1GINV = 0; // Timer1 gate active low NOT USED
    TMR1GE = 0; // Timer1 always counts when on
    T1CKPS1 = 0; // Timer1 prescaler value
    T1CKPS0 = 0; // 1:1 prescaler 65ms range
    T1OSCEN = 0; // LP oscillator off NOT USED
    nT1SYNC = 0; // Ignored as TMR1CS 0 NOT USED
    TMR1CS = 0; // Use internal clock Fosc/4
    TMR1ON = 1; // Timer1 on

    // T2CON
    TOUTPS3 = 0; // set timer 2 postscaler to 1
    TOUTPS2 = 0; //
```

```
TOUTPS1 = 0; //
TOUTPS0 = 0; //
TMR2ON = 0; // Timer2 off
T2CKPS1 = 0; // set timer 2 prescaler to 4
T2CKPS0 = 1; // ~1ms range, 4us resolution

// INTCON
GIE = 0; // Global interrupt disable during setup
PEIE = 1; // Peripheral interrupt enabled
T0IE = 1; // Timer0 overflow interrupt enabled
INTE = 0; // External interrupt RA2 disabled
RABIE = 0; // PORTA change interrupt disabled
T0IF = 0; // Clear Timer0 interrupt flag
INTF = 0; // Clear RA2 interrupt flag
RABIF = 0; // Clear PORTA/B change interrupt flag

// PIE1
ADIE = 0; // ADC interrupt disabled
RCIE = 0; // EUSART receive interrupt disabled
TXIE = 0; // EUSART transmit interrupt disabled
SSPIE = 0; // Synchronous Serial Port interrupt disabled
CCP1IE = 0; // CCP1 interrupt disabled
TMR2IE = 0; // Timer2 match PR2 interrupt disabled
TMR1IE = 1; // Timer1 overflow interrupt enabled

// PIE2
OSFIE = 0; // Oscillator fail interrupt disabled
C2IE = 0; // Comparator C2 interrupt disabled
C1IE = 0; // Comparator C1 interrupt disabled
EEIE = 0; // EE write complete interrupt disabled

// PIR1
ADIF = 0; // Clear ADC complete flag
RCIF = 0; // Clear EUSART receive flag
TXIF = 0; // Clear EUSART transmit flag
SSPIF = 0; //  Clear SSP flag
CCP1IF = 0; // Clear Timer1 capture flag
TMR2IF = 0; // Clear Timer2 PR2 match flag
TMR1IF = 0; // Clear Timer1 overflow flag

// PIR2
OSFIF = 0; // clear oscillator fail interrupt flag
C2IF = 0; // Clear comparator 2 change flag
C1IF = 0; // Clear comparator 1 change flag
EEIF = 0; // Clear EE write complete flag

// TRISA
// Port A IO configuration. 0 output, 1 input
TRISA5 = 1; // Button input
TRISA4 = 1; // Button input & VREF INPUT VERSION 1 ONLY
TRISA3 = 1; // Button input
TRISA2 = 1; // Status LED3 output VERSION 1 ONLY // VREF 2.5V input V3+
TRISA1 = 0; // Status LED2 output
TRISA0 = 0; // Status LED1 output
```

```
// WPUA
WPUA5 = 0; // disable weak pull up
WPUA4 = 0; // disable weak pull up
//WPUA3 = 0; //
WPUA2 = 0; // disable weak pull up
WPUA1 = 0; // disable weak pull up
WPUA0 = 0; // disable weak pull up

// IOCA
IOCA5 = 0; // disable interrupt on change?
IOCA4 = 0; // disable interrupt on change?
IOCA3 = 0; // disable interrupt on change?
IOCA2 = 0; // disable interrupt on change
IOCA1 = 0; // disable interrupt on change
IOCA0 = 0; // disable interrupt on change

// TRISB
TRISB7 = 1; // Button input
TRISB6 = 0; // SCK output
TRISB5 = 0; // output
TRISB4 = 1; // Not connected

// WPUB
WPUB7 = 0; // disable weak pull up
WPUB6 = 0; // disable weak pull up
WPUB5 = 0; // disable weak pull up
WPUB4 = 1; // enable weak pull up

// IOCB
IOCB7 = 0; // disable interrupt on change?
IOCB6 = 0; // disable interrupt on change
IOCB5 = 0; // disable interrupt on change
IOCB4 = 0; // disable interrupt on change

// TRISC
// Port C IO configuration. 0 output, 1 input
TRISC7 = 0; // SDO output
TRISC6 = 0; // Batt1 Charge output
TRISC5 = 0; // Batt1 Load output
TRISC4 = 0; // Batt2 Charge output
TRISC3 = 0; // Batt2 Load output
TRISC2 = 0; // Chip Select output
TRISC1 = 1; // Batt1 Level input
TRISC0 = 1; // Batt2 Level input

// ANSELH
ANS11 = 0;
ANS10 = 0;
ANS9 = 0;
ANS8 = 0;

// ANSEL
ANS7 = 0; // Set pins to digital IO
ANS6 = 0;
ANS5 = 1; // Set Batt1 Level pin as analog input
```

```
    ANS4 = 1; // Set Batt2 Level pin as analog input
//    ANS3 = 0; // Set VREF pin as analog input when making a measurement VERSION 1 ONLY
    ANS3 = 0;
//    ANS2 = 0; // VERSION 1 ONLY
    ANS2 = 1; // Set VREF pin sa analog input V3+
    ANS1 = 0;
    ANS0 = 0;

    // ADCON0
    ADFM = 1; // right justify ADC result (Fill ADRESL)
    VCFG = 0; // voltage reference Vdd
    CHS3 = 0; // analog channel select start at AN4/Batt2
    CHS2 = 1;
    CHS1 = 0;
    CHS0 = 0;
    GO_nDONE = 0; // don't start a conversion yet
    ADON = 1; // enable ADC

    // ADCON1
    ADCS2 = 0; // set adc clock to
    ADCS1 = 0; // Fosc/8 2us cycle
    ADCS0 = 1; // ~26us to capture

    // SSPSTAT
    SMP = 0;
    CKE = 1; // Data transmit on falling edge (CKP = 1)

    // SSPCON
    WCOL = 0;
    SSPOV = 0;
    SSPEN = 1; // Enable LCD
    CKP = 1; // Clock idle high
    SSPM3 = 0; // LCD Master mode, CLK Fosc/16
    SSPM2 = 0;
    SSPM1 = 0;
    SSPM0 = 1;

} // END OF INIT
```

Appendix D

Analysis of New HRL12280 and Retired EATON12280 Lead-acid Batteries

The data in the following document has been measured and prepared with the assistance of Professor Jonathan Scott using the measurement systems developed within this work as part of internal communications with the research group.

Analysis of New HRL12280W and Retired EATON12280W Lead-acid Batteries

Jonathan Scott and Vance Farrow

***Abstract*—This manuscript examines two batteries, essentially identical except for age. The HRL12280W is new, while the EATON12280W is "retired" from service. The retired battery has little or no decrease in charge capacity, which is reflected in the CPE parameters. We find that the old battery is in excellent condition except for an increase in high-frequency resistance, and probably has a long remaining service life.**

I. Introduction

A full and fair test of any measure of ageing would require observation of the subject battery for its shelf life, which is impractical. In this test, a new battery was purchased and is compared to a "retired" battery. The type and ratings of the new battery were chosen to be as close as possible to those of the retired battery.

II. Measurements

In both cases the battery was cycled twice and then its impedance measured. It was finally cycled twice more.

Figure 1 shows the charge and discharge terminal voltage of the EATON (retired) battery as a function of delivered charge. It has a capacity of 60–62 Ampere-hours. This was measured at 5A, which turns out to be approximately the standard C/10 rate, at 23C, cycling between 13.7V and 11.1V, with 2 hours soak time at the end points.

Figure 3 shows the charge and discharge terminal voltage of the new HRL battery as a function of delivered charge. It has a capacity of 62–66 Ampere-hours. This was similarly measured at 5A between 13.7V and 11.1V, with 2 hours soak time at the end points. The retired battery shows 5% less charge capacity. It is impossible to say whether this represents a reduction, since the new capacity of this battery was never measured. Nevertheless, we can say that this is not indicative of any significant capacity loss.

The EATON battery was also load tested. At 80% charge, at 23C, it delivers approximately 11.75V at 43A after 10 seconds into an 0.275Ω, 500 Watt resistor. This is almost double its continuous rated power capacity, although below the maximum rated peak current. See figure 2.

The plots of magnitude of impedance appear in figures 4 and 5. It is clear that the very low frequency impedance of the two batteries is almost indistinguishable. The impedance of the older battery is slightly higher at 0.2 Hz, and appears to be levelling off a little higher.

The authors are with the School of Engineering, the University of Waikato, Hamilton, New Zealand. e-mail: jonathanscott@ieee.org

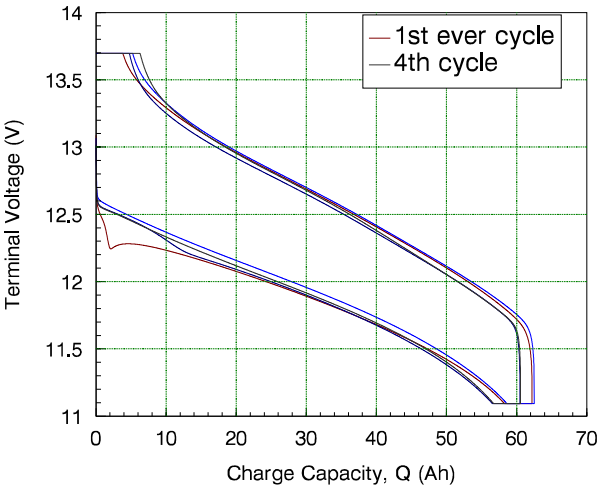


Fig. 1. Cycle data for the EATON 280W 12 Battery. Note kink in the "first ever" cycle after we received the battery, presumably after a long life mostly in storage at full charge. Capacity is 60–62Ah, dependent upon recent history.

Fig. 2. Picture of EATON 280W 12 Battery load test. At 80% charge, it delivers approximately 11.73V at 42.5A after 10 seconds.

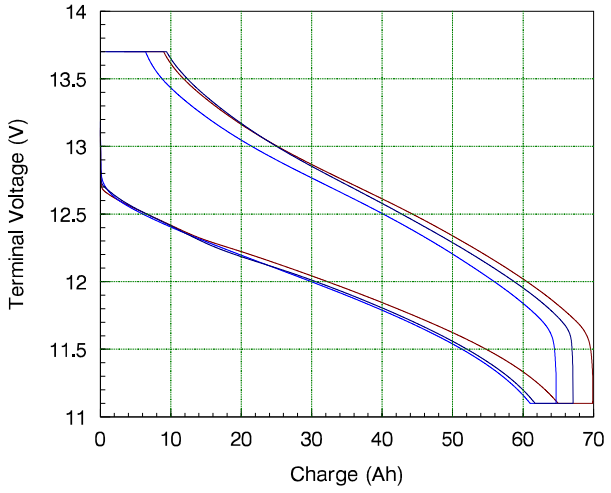


Fig. 3. Cycle data for the HRL 280W 12V battery. Capacity is 62–66Ah, dependent upon recent history.

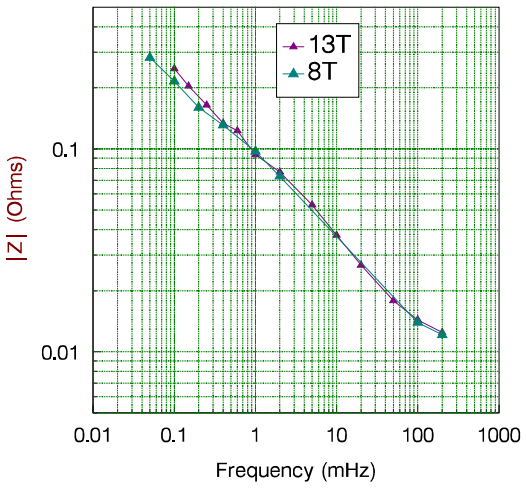


Fig. 4. Measured low-frequency impedance of the EATON 280W 12V battery. Two measurements, both multitone, deliver similar results.

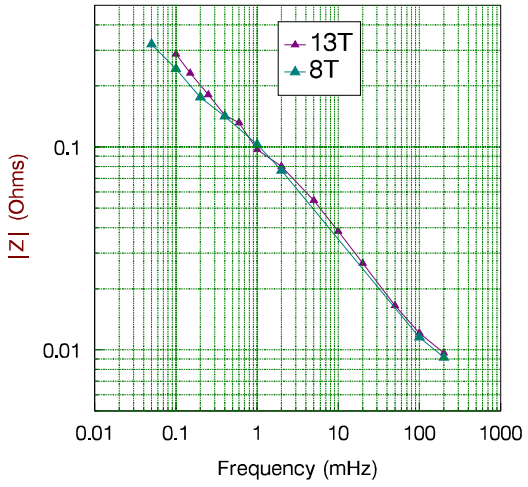


Fig. 5. Measured low-frequency impedance of the HRL 280W 12V battery. Two measurements, both multitone, deliver similar results.

III. Conclusion

The main conclusion is that the EATON battery appears to be perfectly healthy. The change in impedance above 0.2 Hz is not indicative of reduced capacity. The increased high-frequency resistance does not prevent the battery meeting or exceeding specification.

The order of the CPE is $\alpha \approx 0.44$ for the HRL and $\alpha \approx 0.40$ for the EATON. These numbers are consistent with previous measurements of healthy LA batteries.

The old battery has a slightly smaller order (α), about 10% reduced, indicating reduced (or designed lower) power density, and possibly some degradation.[1] If using the CPE parameters as indicators of age, we would say that ageing is perceptible, but capacity not significantly reduced, and power capacity not yet significantly impaired.

[1] We have no proof that reduction in α correlates with calendar age, but we observe it correlates with cycle age.

Appendix E

Analysis of New HRL12540 and Retired HAZE61 Lead-acid Batteries

The data in the following document has been measured and prepared with the assistance of Professor Jonathan Scott using the measurement systems developed within this work as part of internal communications with the research group.

Analysis of New HRL12540 and Retired HAZE61 Lead-acid Batteries

Jonathan Scott and Vance Farrow

***Abstract*—This manuscript examines two batteries, essentially identical except for age. The HRL12540 is new, while the HAZE61 is "retired" from service. The retired battery has greatly reduced charge capacity, which is reflected in both the high and low frequency impedance. We find that the old battery is in poor condition and potentially should have been retired sooner as its capacity has decreased below nominal levels for end of life.**

I. Introduction

A full and fair test of any measure of ageing would require observation of the subject battery for its shelf life, which is impractical. In this test, a new battery was purchased and is compared to a "retired" battery. The type and ratings of the new battery were chosen to be as close as possible to those of the retired battery. Due to similarities in physical dimension both batteries are assumed to be of the same or similar specification with regards to power capability and charge capacity

II. Measurements

In both cases the battery was cycled twice and then its impedance measured. It was finally cycled twice more.

Figure 1 shows the charge and discharge terminal voltage of the HAZE (retired) battery as a function of delivered charge. The first cycle, notable for the characteristic discharge curve, has a capacity of 83–85 Ampere-hours. Subsequent cycles show a rapid decline in capacity. This was measured at 5A, at 23C, cycling between 13.7V and 11.1V, with 2 hours soak time at the end points.

Figure 2 shows the charge and discharge terminal voltage of the new HRL battery as a function of delivered charge. It has a capacity of 127–131 Ampere-hours which does not decrease significantly after cycling, however only the first two cycles are available. This was similarly measured at 5A between 13.7V and 11.1V, with 2 hours soak time at the end points. The retired battery shows 35% less charge capacity in the initial cycle and higher loss of capacity with subsequent cycles. It is impossible to say whether this is the true value of the capacity reduction, since the new capacity of this battery was never measured. However, we can say that the rapid decline of capacity with cycling is characteristic of a battery with significant wear supporting the claim of such a large reduction in capacity. In mission critical applications such as security and safety a common lower threshold for decline in capacity is a 20% decrease. This puts the HAZE61 far beyond the acceptable limit for such applications.

The authors are with the School of Engineering, the University of Waikato, Hamilton, New Zealand. e-mail: jonathanscott@ieee.org

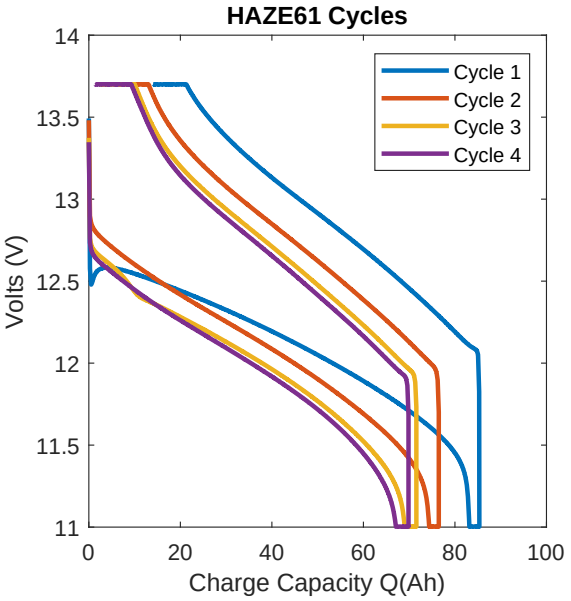


Fig. 1. Cycle data for the HAZE61 Battery. Note kink in the "first ever" cycle after we received the battery, presumably after a long life mostly in storage at full charge. Capacity is 85–65Ah, dependent upon recent history.

A comparison of the magnitude of impedance appear in figures 3 and 4. It is clear that the HAZE61 battery has a higher impedance at all frequencies. The logarithmic scaling of 3 shows the relative change in impedance is greater at high frequencies. The same data is shown in 4 with linear scaling for the magnitude of the impedance showing that not only is the magnitude of the impedance greater, the magnitude of the change in impedance is also greater.

III. Conclusion

The main conclusion is that the HAZE61 battery appears to be severly worn based on all metrics. The reduction in capacity by 35% is beyond the allowable degradation for many applications, especially those mission critical in nature. The low frequency impedance shows a larger change in magnitude and is reccomended as a preferred way of detecting this change in capacity.

The order of the CPE is $\alpha \approx 0.45$ for the HAZE61 and $\alpha \approx 0.48$ for the HRL12540. These numbers are consistent with previous measurements of LA batteries however their significance cannot be stated without knowing the individual value of the order for each battery when brand new.

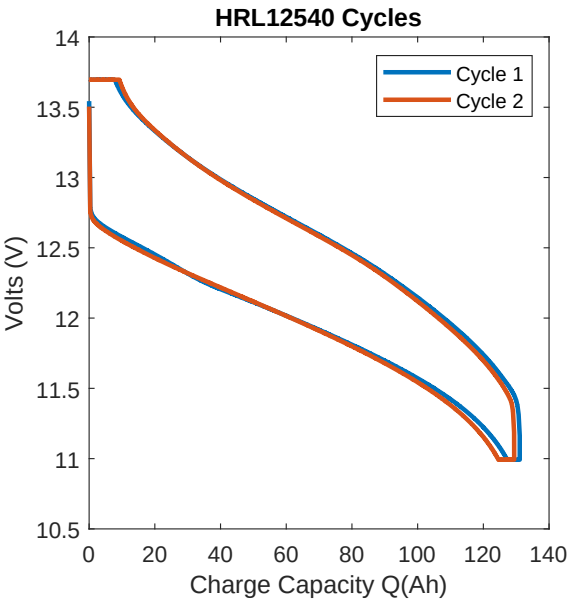


Fig. 2. Cycle data for the HRL12540 battery. Capacity is 127–131Ah, dependent upon recent history.

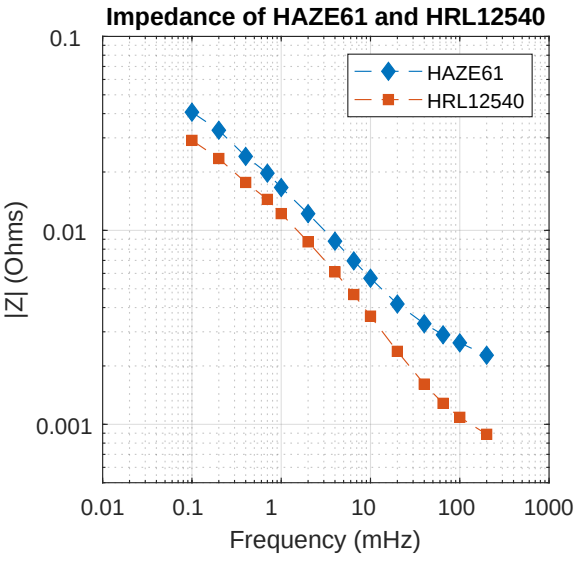


Fig. 3. Measured low-frequency impedance of the HRL12540 and HAZE61 batteries. Two measurements, both multitone showing HAZE61 with an increased impedance across all measured frequencies.

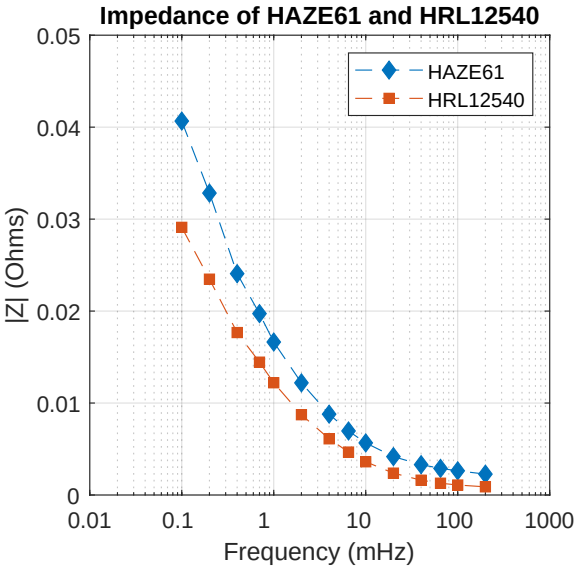


Fig. 4. Measured low-frequency impedance of the HRL12540 and HAZE61 batteries with linear magnitude showing a greater increase in impedance at low frequencies.

Appendix F

Measurement of Multi-tone and Equilibriation

The following data has been measured and document prepared by Professor Jonathan Scott in support of the work in this thesis using the measurement systems developed within this work.

Measurement of battery impedance
&
Equilibration after large-signal operation

Abstract: This document presents evidence that (1) multitone complex impedance measurement is at least as good as single-tone, stepped impedance measurement, and that (2) significant confusion and error can arise because the impedance of a battery changes greatly as it equilibrates after large signal operation, such as cycling.

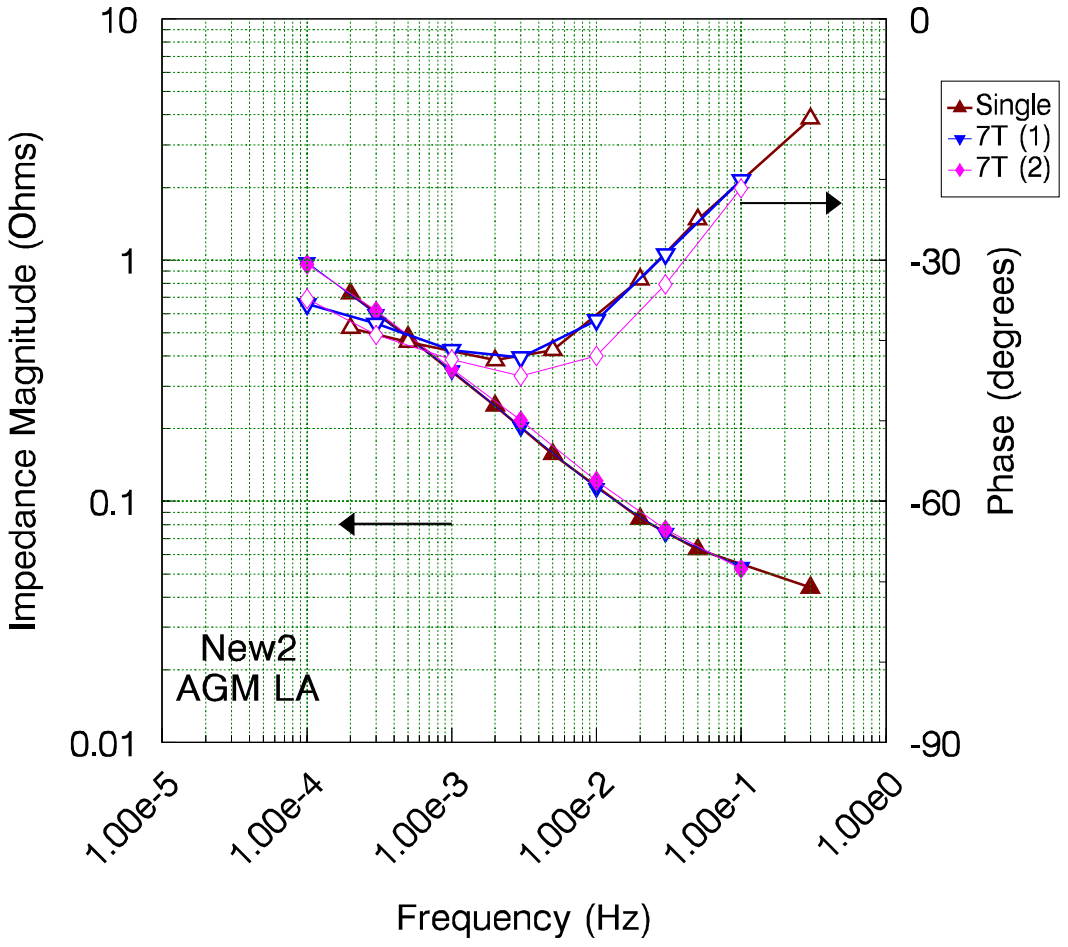
Measurement Method: The battery used here is a 30Ah AGM lead-acid battery intended for use in light-duty traction applications. The battery has been labelled "New2" as it was the second of a set purchased to replace in-use batteries in a small scooter. Although slightly used, it is in near-new condition.
Impedance is measured by subjecting the battery to a current signal, and measuring both current and terminal voltage using Kelvin connections by means of an HP66332A 2-quadrant, 5A programmable power supply controlled from a dedicated linux computer. Measurements were made using Vance Farrow's "bc" and "bz" software packages. A typical measurement script looks like this, with 'New2' given as the script parameter $1:

```
./bz HP66332 /dev/ttyUSB0 9600 14 11.0 200e-3 1.0 6 100e-6 200e-6 -500e-6 -1e-3 100e-3 $1_C5T
./bc HP66332 /dev/ttyUSB0 9600 1 14.1 10.8 5 .5 80 300 300 $1_cyc
./bz HP66332 /dev/ttyUSB0 9600 14 11.0 200e-3 1.0 6 100e-6 200e-6 -500e-6 -1e-3 100e-3 $1_C5T_post1
./bz HP66332 /dev/ttyUSB0 9600 14 11.0 200e-3 1.0 6 100e-6 200e-6 -500e-6 -1e-3 100e-3 $1_C5T_post2
./bz HP66332 /dev/ttyUSB0 9600 14 11.0 200e-3 1.0 6 100e-6 200e-6 -500e-6 -1e-3 100e-3 $1_C5T_post3
```

and a script such as this, with various filenames given as the parameter:

```
#!/bin/bash
dftv $1 100e-3 1 0 D F C 3 >hold
dftv $1 100e-3 1 0 D F L >>hold
dftv $1 1000e-6 1 0 D F C 3 >>hold
dftv $1 1000e-6 1 0 D F L >>hold
dftv $1 500e-6 1 0 D F C 3 >>hold
dftv $1 500e-6 1 0 D F L >>hold
dftv $1 200e-6 1 0 D F C 3 >>hold
dftv $1 200e-6 1 0 D F L >>hold
dftv $1 100e-6 1 0 D F C 3 >>hold
dftv $1 100e-6 1 0 D F L >>hold
awk -f iv2Z.awk hold >>ZF-$1.fmp
```

was used to extract the impedance values from the ".tvi" file containing data triples of time-voltage-current" provided by bz.

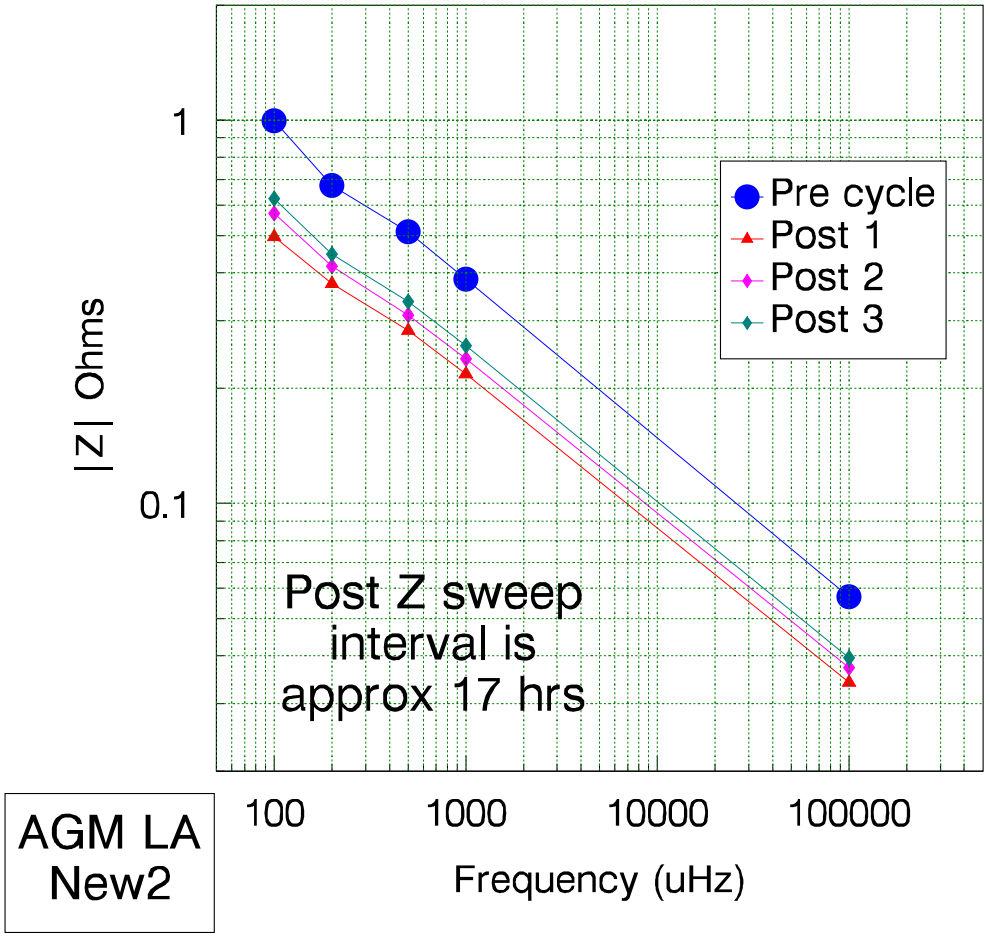


IMPEDANCE OF NEW2 MEASURED BY SINGLE AND MULTITONE METHODS. TWO 7-TONE SWEEPS ARE USED TO CHECK THAT THE RESULT IS STABLE.

Results:

The first plot compares single-tone with multitone measurement. The measurement was repeated at 2x higher and 3x smaller current levels to ensure that the measurement was "small signal". The spectrum of the voltage was observed to check that there were no IMD products visible. With a 40dB+ SNR no spurious tones were detected.
The conclusion is that, while there is some deviation, ***the multitone agrees with the single-tone data***.

The next plot presents a series of identical 5-tone measurements. The first (pre-cycle) was made after the battery was rested for 1 week, but before the battery was charged, discharged, recharged to full, all at a current of 5A, and finally discharged to 80% capacity. After the cycling three further sweeps (post 1, post 2, and post 3) were made. (The script used is the example script included above.) Each sweep took approximately 17 hours to run, meaning that the post traces are separated by three-quarters of a day each.
The conclusion is that ***the battery impedance continues to "drift" as it internally equilibrates,*** and this equilibration takes a relatively long time. It must now be recommended that all impedance measurements on fractional elements be carried out after a substantial period of rest.



MULTIPLE IMPEDANCE MEASUREMENTS MADE USING THE MULTITONE METHOD AND 5 FREQUENCIES.

Bibliography

[1] M. Woody, M. Arbabzadeh, G. M. Lewis, G. A. Keoleian, and A. Stefanopoulou, "Strategies to limit degradation and maximize Li-ion battery service lifetime - Critical review and guidance for stakeholders," *Journal of Energy Storage*, vol. 28, p. 101231, Apr. 2020.

[2] M. H. Lipu, M. Hannan, A. Hussain, M. Hoque, P. J. Ker, M. Saad, and A. Ayob, "A review of state of health and remaining useful life estimation methods for lithium-ion battery in electric vehicles: Challenges and recommendations," *Journal of Cleaner Production*, vol. 205, pp. 115–133, Dec. 2018.

[3] M. Berecibar, I. Gandiaga, I. Villarreal, N. Omar, J. Van Mierlo, and P. Van den Bossche, "Critical review of state of health estimation methods of Li-ion batteries for real applications," *Renewable and Sustainable Energy Reviews*, vol. 56, pp. 572–587, Apr. 2016.

[4] W. Waag, S. Käbitz, and D. U. Sauer, "Experimental investigation of the lithium-ion battery impedance characteristic at various conditions and aging states and its influence on the application," *Applied Energy*, vol. 102, pp. 885–897, Feb. 2013.

[5] J. R. Macdonald, "Impedance spectroscopy," *Annals of Biomedical Engineering*, vol. 20, pp. 289–305, 1992.

[6] M. Chen and G. Rincon-Mora, "Accurate Electrical Battery Model Capable of Predicting Runtime and I–V Performance," *IEEE Transactions on Energy Conversion*, vol. 21, pp. 504–511, June 2006.

[7] J. Scott and R. Hasan, "New Results for Battery Impedance at Very Low Frequencies," *IEEE Access*, vol. 7, pp. 106925–106930, 2019.

[8] Y. Cao, R. C. Kroeze, and P. T. Krein, "Multi-timescale Parametric Electrical Battery Model for Use in Dynamic Electric Vehicle Simulations," *IEEE Transactions on Transportation Electrification*, vol. 2, pp. 432–442, Dec. 2016.

[9] X. Liu, W. Li, and A. Zhou, "PNGV Equivalent Circuit Model and SOC Estimation Algorithm for Lithium Battery Pack Adopted in AGV Vehicle," *IEEE Access*, vol. 6, pp. 23639–23647, 2018.

[10] N. Omar, D. Widanage, M. Abdel Monem, Y. Firouz, O. Hegazy, P. Van den Bossche, T. Coosemans, and J. Van Mierlo, "Optimization of an advanced battery model parameter minimization tool and development of a novel electrical model for lithium-ion batteries: NOSHIN'S MODEL," *International Transactions on Electrical Energy Systems*, vol. 24, pp. 1747–1767, Dec. 2014.

[11] G. Saldaña, J. I. San Martín, I. Zamora, F. J. Asensio, and O. Oñederra, "Analysis of the Current Electric Battery Models for Electric Vehicle Simulation," *Energies*, vol. 12, p. 2750, July 2019.

[12] R. Hasan and J. B. Scott, "Fractional behaviour of rechargeable batteries," (Victoria University of Wellington, New Zealand), pp. 111–114, Electronics New Zealand Inc, Nov. 2016.

[13] J. E. B. Randles, "Kinetics of rapid electrode reactions," *Discussions of the Faraday Society*, vol. 1, p. 11, 1947.

[14] J. Sabatier, M. Aoun, A. Oustaloup, G. Grégoire, F. Ragot, and P. Roy, "Fractional system identification for lead acid battery state of charge estimation," *Signal Process.*, vol. 86, p. 2645–2657, Oct. 2006.

[15] J. Sabatier, M. Merveillaut, J. M. Francisco, F. Guillemard, and D. Porcelatto, "Fractional models for lithium-ion batteries," in *2013 European Control Conference (ECC)*, (Zurich), pp. 3458–3463, IEEE, July 2013.

[16] R. Hasan and J. Scott, "Extending Randles's Battery Model to Predict Impedance, Charge–Voltage, and Runtime Characteristics," *IEEE Access*, vol. 8, pp. 85321–85328, 2020.

[17] Jianbang Liu, *Automatic Impedance Measurement at Ultralow Frequencies*. Masters Thesis, The University of Waikato, Waikato, 2019.

[18] Mark H. Jones, *The Electrical Properties of Interfacial Double Layers*. PhD Thesis, The University of Waikato, Waikato, 2015.

[19] G. Brug, A. van den Eeden, M. Sluyters-Rehbach, and J. Sluyters, "The analysis of electrode impedances complicated by the presence of a constant phase element," *Journal of Electroanalytical Chemistry and Interfacial Electrochemistry*, vol. 176, no. 1, pp. 275 – 295, 1984.

[20] A. Kartci, N. Herencsar, J. T. Machado, and L. Brancik, "History and Progress of Fractional-Order Element Passive Emulators: A Review," *Radioengineering*, vol. 29, pp. 296–304, June 2020.

[21] E. Warburg, "Ueber das verhalten sogenannter unpolarisirbarer elektroden gegen wechselstrom," *Annalen der Physik*, vol. 303, no. 3, pp. 493–499, 1899.

[22] H. Fricke, "The theory of electrolytic polarization," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 14, pp. 310–318, Aug. 1932.

[23] X. Wang, X. Wei, and H. Dai, "Estimation of state of health of lithium-ion batteries based on charge transfer resistance considering different temperature and state of charge," *Journal of Energy Storage*, vol. 21, pp. 618–631, Feb. 2019.

[24] R. Morrison, "RC Constant-Argument Driving-Point Admittances," *IRE Transactions on Circuit Theory*, vol. 6, no. 3, pp. 310–317, 1959.

[25] J. B. Scott and S. Seshadri, "Corrections to Compact Nonlinear Model of an Implantable Electrode Array for Spinal Cord Stimulation," *IEEE Transactions on Biomedical Circuits and Systems*, vol. PP, no. 99, 2018. Publisher: Institute of Electrical and Electronics Engineers.

[26] S. E. Li, B. Wang, H. Peng, and X. Hu, "An electrochemistry-based impedance model for lithium-ion batteries," *Journal of Power Sources*, vol. 258, pp. 9–18, July 2014.

[27] M. Berecibar, M. Garmendia, I. Gandiaga, J. Crego, and I. Villarreal, "State of health estimation algorithm of LiFePO4 battery packs based on differential voltage curves for battery management system application," *Energy*, vol. 103, pp. 784–796, May 2016.

[28] X. Han, M. Ouyang, L. Lu, J. Li, Y. Zheng, and Z. Li, "A comparative study of commercial lithium ion battery cycle life in electrical vehicle: Aging mechanism identification," *Journal of Power Sources*, vol. 251, pp. 38–54, Apr. 2014.

[29] R. Hasan and J. Scott, "Measurement for fractional characteristic of Lithium batteries," in *2019 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, (Auckland, New Zealand), pp. 1–5, IEEE, May 2019.

[30] David MacCallum, *"Automatic Measurement of Implantable Electrode Characteristics using a Single Impedance Analyser.* Honours research thesis, Faculty of Sciene and Engineering, The University of Waikato, 2017.

[31] C. Dunn, *Deducing Battery Impedance From Vechicle Data*. Honours research thesis, Faculty of Sciene and Engineering, The University of Waikato, 2019.

[32] F. Leng, C. M. Tan, and M. Pecht, "Effect of Temperature on the Aging rate of Li Ion Battery Operating above Room Temperature," *Scientific Reports*, vol. 5, p. 12967, Oct. 2015.

[33] I. Baghdadi, O. Briat, J.-Y. Delétage, P. Gyan, and J.-M. Vinassa, "Lithium battery aging model based on Dakin's degradation approach," *Journal of Power Sources*, vol. 325, pp. 273–285, Sept. 2016.