University of Glasgow | School of Computing Science

Honours Individual Project Dissertation

A Model-Based Predictive Battery Monitoring System for Multirotor Drones

Karl Hartmann

March 22, 2024

Abstract

Drones are ever increasing in popularity, but still rely on volatile lithium-ion batteries. While some commercial companies make use of complex battery monitoring systems to minimise risk, these are expensive, and largely proprietary, so many drones are still sold with no advanced battery monitoring. This project aims to create an affordable, minimal solution to give users early warning in cases where battery performance is worse than expected, while monitoring cell differences to warn of cell-specific issues. This project makes use of a model-based approach, running an onboard battery simulation while monitoring the real performance, and comparing these values to estimate the battery state. This approach is able to quickly build an estimation of battery performance, compared to continuous learning-based approaches, meaning it can be integrated into the drone, rather than being required for each battery. This leads to a lower cost when compared to existing solutions, while maintaining functionality.

Education Use Consent

I hereby grant my permission for this project to be stored, distributed and shown to other University of Glasgow students and staff for educational purposes. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

Signature: Karl Hartmann Date: 20 February 2024

Contents

Acronyms

ADC Analog to Digital Converter.
Ah Ampere-hours.

BMS Battery Management System.

IO Input/ Output.

mAh Milliampere-hours.
mWh Milliwatt-hours.

OCV Open-Circuit Voltage.
OS Operating System.

SOC State of Charge.
SOH State of Health.

Wh Watt-hours.

1 Introduction

1.1 Motivation

Drones are currently, and have been over the last several years, rapidly increasing in popularity (Kapustina, Larisa et al. 2021). Multi-copter drones are especially popular, due to their manoeuvrability and ease of use if equipped with suitable flight control hardware and software. However, as they make use of fixed-pitch propellers, and often a single battery pack, they are vulnerable to power loss, which would lead to a complete loss of control. This creates a potential danger as the drone may be flown over objects/ people which could be damaged by the drone. Due to this danger, the onboard battery must be accurately monitored to ensure it is able to provide the required energy until the planned end of the flight.

It is necessary to monitor the State of Charge (SOC) of the battery throughout a flight to ensure safety (Plioutsias et al. 2018), because the battery is the most critical component of a drone, as found by Shafiee et al. (2021). This is often performed by a Battery Management System (BMS), with some mass-market multi-copters being sold with "smart batteries", which integrate a BMS into each individual battery, allowing for monitoring of performance over time, counting of age/ charge cycles, and additional safety features such as cell balancing and over voltage protection during charging. While this solution is effective in terms of increased safety, it also increases the cost of replacement batteries significantly, increases battery weight, and allows manufacturers to serialise their batteries, limiting user repairability. Due to these drawbacks, and a lack of affordable and available hardware/ software, many drones still omit the use of a BMS, such as lower cost, or DIY models. These drones may rely on direct battery readings such as voltage, or basic coulomb-counting methods such as used Milliampere-hours (mAh) to estimate remaining flight time, which are effective in rough estimations, and in notifying the user when a battery is fully exhausted, but are unlikely to detect variations early in the discharge cycle. This means that if a battery is performing below expectation this may not be noticed until late into the flight, which could lead to an accident if this is unexpected.

This project aims to create a prototype battery monitoring system which will be drone-mounted, lightweight, low-power and low-cost, while still being able to give users early warning of decreased battery performance, through specific warnings relating to the type of battery malfunction. Making the project drone-mounted ensures that the price is kept low, since only one monitor is required, as opposed to one per battery. This is also helpful for smaller drones, where adding monitoring hardware to a battery is infeasible. The project makes use of a model-based battery simulation, which is then compared to real performance to estimate performance, then from this estimation further parameters are estimated such as remaining flight time. Since the model-based approach requires initial battery parameters, this project also aims to create a simple methodology for finding the required parameters in an easy and widely accessible way. This will allow a wider range of drones to be equipped with necessary battery monitoring equipment, as it is more feasible to use for individual users, smaller drones, and drones focusing on low-cost.

1.2 Aim and Objectives

This project aims to create a low-cost, lightweight, easy to use drone-mounted battery monitor for multi-rotor drones, which can be integrated within existing drone systems for enhanced battery monitoring.

Quick Estimation The prototype should be able to quickly find lack of performance: Given the risks of embarking on a long flight with a potentially underperforming battery, in cases where the flight time is impacted by more than 20%, this should be clearly estimated during the first 20% of the flight.

Cell Difference Identification The prototype should be able to warn of differently performing cells: In batteries where a single cell has more than 10% higher internal resistance or 10% lower capacity than the average cell, this should be clearly visible in output results.

Specific Prognosis The prototype should be able to differentiate between high resistance and low capacity: If a battery is under performing, it should be possible to user the outputs of the monitoring system to clarify whether this is due to a higher internal resistance than expected, or a lower capacity than expected.

2 Background

2.1 Lithium Battery Background

The primary power source for most drones are lithium-ion batteries, as these provide high energy and power density compared to other common battery technologies such as Nickel-Cadmium or Lead-Acid, while remaining relatively low in cost due to their wide usage leading to extensive mass-manufacturing. This battery technology does however have some drawbacks, especially in safety and performance-critical applications. The information in this chapter is based primarily on the books *Lithium-Ion Batteries: Basics and Applications* by Korthauer (2018), and *Battery Reference Book* by Crompton (2000).

2.1.1 Lithium Battery Basics

Lithium-ion batteries are made up of individual cells, which have several important properties that depend on the manufacture of the cell. These include nominal voltage, nominal capacity, energy, discharge rate, and internal resistance.

The voltage of a cell decreases during a discharge cycle, and should typically be constrained to ensure the minimum safe voltage is not passed. Nominal voltage is equal to the mean voltage of a cell as it is discharged. The lithium-ion polymer cells typically used in drones have a fully charged voltage of 4.2V, a nominal voltage of 3.7V and a minimum voltage of 3V, though this should not be approached to extend lifespan.

Nominal capacity is the total amount of Ampere-hours (Ah) released from the battery when discharged from full to empty, with one Ah being equivalent to a discharge of one ampere for one hour.

Discharge rate is usually given as a C-rating, where one C is equal to one times the capacity, in Amps. So an 8Ah, 20C battery would be able to provide a maximum of

$$8A * 20C = 160A \tag{2.1}$$

of instantaneous discharge current. This parameter can be used to calculate maximum battery power, by multiplying the maximum current by the current voltage. Due to changing voltage during discharge, the available power decreases later in the discharge.

This C value depends on the internal resistance of the battery, which is the effective electrical resistance of the battery. This means that a higher internal resistance leads to more wasted energy through conversion to heat, and therefore lower battery efficiency and current output capability.

Battery energy can be calculated by multiplying the nominal voltage by the nominal capacity, which gives a result in Watt-hours (Wh). One Wh of energy is equivalent to using 1 watt for one hour. This parameter is more useful for estimating battery life than the nominal capacity since most battery applications require a certain amount of power, not current.

Batteries are made up of several individual cells which are connected together in either series or parallel, increasing either the total voltage, or capacity (though the total energy is the same regardless of connection type).

2.1.2 Factors Affecting Usable Capacity

Environmental Factors The performance of lithium-ion batteries is highly dependent on the operating conditions, with varying temperature, and discharge currents having the greatest effects. As the temperature of the battery decreases, the effective capacity also decreases, and the internal resistance increases, as shown by Lu et al. (2019). These changes both contribute to a reduced battery life, as a higher internal resistance means that even in cases where capacity is left in the battery, current being drawn may cause the voltage may decrease beyond a usable or safe threshold earlier than usual.

Discharge Current Internal resistance is inherent to batteries, and whilst lithium-ion batteries typically have lower resistance compared to other battery types, it is still present. In practical terms, this means that the battery acts as a resistor, wasting some energy through heat, with this amount of heat increasing as the current increases.

While in lead-acid batteries, discharging at higher currents directly reduces the battery capacity due to Peukert's law, this effect has been found not to be present in lithium batteries, as seen in Doerffel and Sharkh (2006). As such the only actual loss of energy due to internal resistance in lithium-ion batteries is that which is converted to heat.

Despite actual capacity loss due to internal resistance being generally quite low, internal resistance also causes voltage drop, as a result of Ohms law, where the voltage drop is equal to the current multiplied by the resistance:

$$V = IR, \tag{2.2}$$

where V is the voltage drop, I is the current, and R is the internal resistance. Due to this voltage drop, the useful capacity of the battery could be considerably lower than the nominal capacity, in cases where the current demand is high, and/ or the internal resistance is high. As such, the internal resistance must be taken into consideration when estimating discharge time, as multi-copter drones typically have quite high current requirements.

Aging Lithium batteries suffer from several different types of aging, which have differing effects on the internal structure of the cells, and their performance. These are caused by either the chemicals within the cells reacting with each other, additional layers forming on the barrier on the negative electrode, or physical stress on the battery. The specifics of these aging mechanics are beyond the scope of this project, but their practical effects are highly relevant.

The severity of battery aging depends on many factors, such as the typical depth of discharge, typical maximum charge voltage, storage voltage, storage temperature, and calendar age. In practice, these forms of aging can increase the internal resistance of the battery, and/ or decrease the battery capacity. In both cases, usable capacity is decreased as described previously.

Since lithium batteries age even without usage, especially if stored at a voltage above or below the safe storage voltage, or in high temperatures, they may unexpectedly perform worse than expected. This is a greater risk in batteries which are rarely used, or those in cases where the user is unaware of safe storage procedures.

Overall, these batteries are generally performant, but can be unpredictable, potentially leading to disastrous consequences in the case of drones. As such it is necessary to accurately estimate their remaining usable capacity to detect any anomalies as early as possible.

2.1.3 Battery State of Charge

An effective metric for estimating the remaining useful life of a battery is the SOC. This is defined as the amount of energy available from a battery divided by the energy in the battery when fully charged. Energy is defined as described previously, and uses Wh as a unit. This is more useful than Ah, since it gives the real usable energy, so an SOC difference of 20% is equal to 20% of the

total battery energy, whereas a difference of 20% available capacity is equal to a different amount of energy depending on the voltage at the start and end of the capacity measurements.

2.1.4 Battery State of Health

In order to effectively calculate SOC, the energy available from a fully charged battery must be known, but this changes over time as described in the aging section. Therefore, this maximum energy must also be tracked, using a metric known as State of Health (SOH). This does not have a fixed definition, as it is an abstracted metric based on both the available energy, and the internal resistance of the battery, since these both change during aging and decrease the useful life of the battery.

2.2 Drone Specific Background

Drones primarily make use of lithium-ion batteries due to their high energy and power density, meaning they are able to store a lot of energy in a small volume and mass, and are also able to output a high amount of power relative to their size and capacity. For drone batteries, cells are usually connected in series, as the motors require a certain voltage to be able to spin quickly enough for flight. As the energy within a battery decreases over a flight, its voltage decreases, and to ensure battery longevity and safety, this voltage should never drop below a certain threshold (typically 3V per series cell in lithium-ion polymer batteries commonly used in drones). This voltage also drops further as more current is drawn from the battery, though this voltage drop decreases when less power is used, even at the same battery energy state, as it depends partially on the internal resistance.

The cells in a battery with series connections must be matched in terms of type and performance, as otherwise a certain cell(s) may perform worse than others, leading to an imbalance in voltage, and therefore an imbalance of power taken from each cell, which could cause some cells to overheat due to the increased power demand. Due to this, per-cell monitoring is necessary to be able to find these imbalances before they cause issues.

2.2.1 Risks to Drone Flight

Since drones require a relatively constant amount of power to remain in flight, it is important to consider how lithium-ion batteries react throughout a discharge cycle. The constant power requirements of a drone mean that as voltage drops, current increases, since power is defined as:

$$P = IV, \tag{2.3}$$

where P is power, I is current, and V is voltage. Due to this, there is a feedback loop as voltage decreases, with lower voltage leading to a higher current requirement, again leading to a lower voltage due to the battery's internal resistance. At higher battery energy levels this voltage drop quickly plateaus as the requested power is achieved at a lower amperage, and therefore the voltage drop is less, but as the voltage decreases near the minimum safe threshold, it can continue quickly dropping, causing a total power loss when the voltage drops below the minimum operating voltage of the motors, motor controllers, or any other vital operating components. This can occur earlier than expected if the battery performance is worse than expected, which may be due to battery aging, excessively high or low temperatures, or any other performance-inhibiting condition. It could also occur if one cell has lower voltage than the others, as the overall voltage may initially appear typical, but as the low cell approaches minimum voltage, it will quickly drop in voltage, leading to a sudden greater load on the other cells, which in turn causes more voltage drop.

Due to these potential battery issues, it is vital to be able to accurately estimate the battery SOC on a drone to ensure safe operation. An accurate SOC estimation can then be used to mitigate risks of low voltage in several ways.

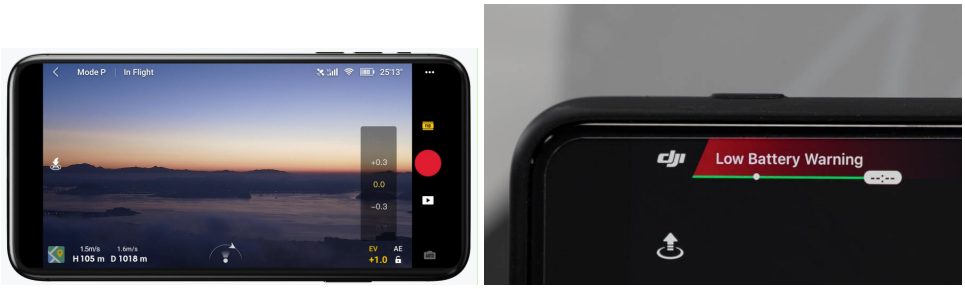
2.2.2 Mitigating Risk of Low Voltage

Feedback to User If the drone pilot is aware of the SOC, they can make informed decisions on the operation of the drone, such as how far away to fly, which speed to fly at, and the potential risk of high energy maneuvers. Providing the user with this awareness is very important in drones which are primarily user controlled (not automated) to ensure safe operation.

Automated Drone Response Many consumer drones are equipped with automated flight features (Hildebrand 2019), where through the use of an onboard flight controller equipped with several sensors, such as a gyroscope, accelerometer, barometer, magnetometer, and GPS receiver, the drone is able to perform maneuvers without user intervention, such as automatically flying to certain coordinates, or returning to the takeoff point and landing.

Combining this ability with an accurate SOC estimation can allow the drone to perform automatic actions based on battery status. For example when a certain minimum SOC threshold is reached the drone can return to the takeoff point automatically, ensuring it performs a safe landing close to the user before losing power. Several other mitigations could also be put in place, such as limiting the horizontal or vertical speed of the drone to minimize power usage, helping to keep the battery voltage at a safe level for the remainder of the flight.

DJI, the worlds largest consumer drone manufacturer, employ both of these methods in their drones, with a flight time estimation provided to the user in the main control UI, and "low battery RTH" automatically returning the drone to the takeoff point if the battery reaches a low threshold footnoteDJI. Accessed 2024-07-03.



(a) DJI Fly app showing battery percentage and remaining flight time in top right. [a]

(b) DJI Go 4 app showing low battery warning once RTH has been triggered. [a]

[a] DJI. Accessed 2024-07-03

[a] DJI. Accessed 2024-07-03

Reducing Other Battery Loads Some drones include payloads which have high energy requirements, such as those performing edge computing, especially when utilising AI, or transferring large amounts of data. A study (Kim et al. 2021) found that running a deep learning algorithm onboard a drone edge computer to perform sensing on an image took 7.09 joules of energy per operation, and that offloading the data for computing took 3.36 joules per operation. Assuming a frame rate of 24fps, this is equivalent to 170.16 watts for local computation, or 80.64 watts for offloading.

This presents considerable power usage, causing increased battery power usage. However, in cases where payload operation is not required for flight, this could be disabled depending on the battery state, such that a safe return flight is possible, and prioritised over any additional tasks. This could happen automatically or feedback could be provided to the user regarding battery state, from which they could decide to disable additional power-using features.

3 Related Work

To ensure safe drone flight, accurate battery monitoring is required, since without this the user or flight system will not be aware of the remaining safe flight time, which could lead to in-flight power loss. The main values required for battery monitoring are:

- State of charge
- State of health
- Internal resistance
- Cell differences

There are several different existing methods for measuring or estimating these values, each with specific advantages and disadvantages.

3.1 State of Charge Estimation

3.1.1 Open Circuit Voltage

The most simple method for SOC estimation is Open-Circuit Voltage (OCV) measurement, as described in Pattipati et al. (2014). OCV is the voltage of the battery measured when it is not connected to any other components/ when no power is being drawn from the battery. This method makes use of the fact that the voltage of a lithium-ion battery decreases throughout a discharge, and as such a lower voltage corresponds to a lower SOC. This method does however have several drawbacks.

Required Parameters The voltage decrease of a battery during a discharge is non-linear, meaning that the SOC value of a certain voltage cannot be directly obtained without prior information. This information is known as an OCV-SOC characterization curve, and is an "offline" measurement, meaning it is not measured during actual usage of the battery, but must be taken beforehand. To make this measurement, the capacity of the battery must be known, to enable accurate SOC measurement during discharge using a technique known as coulomb counting, described later in this section. The discharge rate must also be constant, and minimal, to minimise the voltage drop due to current draw.

Open Circuit Requirement Due to the definition of OCV, there must be no power drawn from the battery when the measurement is taken. This alone means that this method is not usable during a drone discharge, where power is constantly required, but even if the power could be temporarily decreased to zero, a battery's voltage does not immediately recover to OCV due to an effect known as hysteresis. This is the effect that the history of the battery usage affects the current OCV, with the OCV appearing lower if the battery was previously discharged, or higher if it was previously charged (Qiu et al. 2011). This effect can take a long time to disappear, often taking hours (Meng et al. 2018). This means that the battery must be at zero power for some time before an accurate measurement of OCV can be taken. Some approaches have been taken to minimize this time, which have proven effective, but require injection of AC currents into the battery (Densmore and Hanif 2015) (Galeotti et al. 2015), which requires further hardware, and still require the battery to be at open circuit conditions for several minutes.

Primarily due to the open circuit requirement, this method is not usable on drones, but could be implemented to estimate the SOC in cases where the OCV during different states of charge is known and the power can be reduced to near zero for some time. In this project, OCV measurement was initially used to estimate the SOC of the battery before its usage in flight, but the generated OCV-SOC curve was found to be too inaccurate, and that if a battery with different properties was used, this method led to misleading data once the battery was being used in flight.

3.1.2 Coulomb Counting

Coulomb counting is a well-established method for estimating SOC in various batteries, with Ah counting at one point being the most common SOC measurement method (Piller et al. 2001). This is especially useful for lithium-ion batteries, where the capacity is not directly impacted by power usage as discussed previously. This method uses several measurements combined with initial condition parameters to calculate the remaining energy within the battery. The values required are:

- Instantaneous battery voltage (Volts)
- Instantaneous battery current (Amperes)
- Total battery energy (Wh)
- Used battery energy (Wh)

This method works by integrating the instantaneous power, calculated from the voltage and current, over some time, giving a value for energy used. This value is then added to a stored value of total energy used, which can be divided by the total battery energy to deduce a SOC. Theoretically this method should give accurate SOC readings, but it has several limitations, and potential areas for inaccuracy. Some of these are described by Movassagh et al. (2019).

Voltage/ Current Measurement Inaccuracy If the current or voltage sensors provide inaccurate measurements, the power calculation will lead to inaccurate results, leading to an SOC estimation which is either too low if the calculated power is higher than reality, or too low if the calculated power is lower than reality.

Cumulative Errors Since the used energy parameter is a running total, any inaccuracy in the power estimation will cause an increasingly large cumulative error over time. Jeong et al. (2014) attempts to mitigate this error by making use of the OCV measurement method described previously. This paper resets the used battery energy during short times of no battery usage, by estimating the battery hysteresis from two voltage measurements a short time apart, then calculating where on the hysteresis curve the measurement lies as seen in 3.1. This method was found to be effective in estimating the true SOC in car power usages, where times of zero usage occur relatively frequently, but these times are not found on multi-copter drones, where once in flight, power is never reduced to zero.

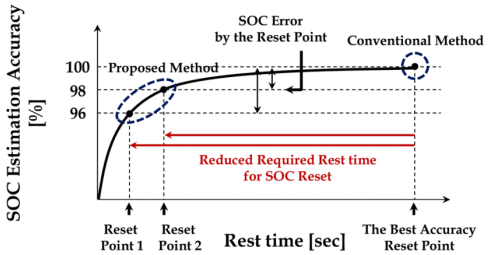


Figure 3.1: *Hysteresis curve estimation from Jeong et al. (2014).*

Initial SOC Inaccuracy Whenever the initial state of the battery is not 100% charged, the amount of energy taken out so far must be known. This is most easily estimated using OCV measurement, but this suffers from the inaccuracies mentioned previously. In cases where this initial SOC is inaccurate, the results for the rest of the SOC measurements will be wrong by this inaccuracy, but this inaccuracy is not additive, so will not get worse over time.

Total Battery Energy Inaccuracy Due to the battery aging described previously, the total battery energy parameter changes over time, leading to an inaccurate SOC estimation, even if the used energy is accurate. Also, despite the OCV SOC relationship remaining similar during aging, when this ratio is used to assume the initial energy for use in coulomb counting, it will lead to excessively high values. E.g., if a 10Wh battery has 80% of its original capacity, measuring SOC via OCV to be 50% will lead to an initially estimated 5Wh used, when in reality only 4Wh have been used, since the overall energy is 8Wh, not 10. Due to these issues, battery SOH must be accurately measured to ensure accuracy in coulomb counting, as described in Movassagh et al. (2021).

Current Integration Error Since current is constantly changing, and measurements cannot truly be taken instantaneously, the change in current between measurements is not taken into consideration, leading to an integration error (Movassagh et al. 2021) (Movassagh et al. 2019).

In this project, this error is minimised by making use of a separate, independent Analog to Digital Converter (ADC) chip which supports onboard averaging, to build up an average of very quick readings which is then sent to the main processor when it is ready to process this data.

3.2 State of Health Estimation

To create accurate measurements using any method, an accurate SOH metric is required, since both techniques previously mentioned would otherwise suffer from inaccuracy, as described in their specific sections. In practice, many advanced monitoring methods include SOH estimation alongside SOC estimation, creating a complete battery state estimation method. There exist a wide range of different methods, but most of these can be generally grouped into three categories, with some overlap:

3.2.1 Data-Driven Approaches

Data-driven approaches for battery state estimation have been successfully demonstrated multiple times in the past, with low error. These make use of various forms of artificial intelligence to "learn" the battery parameters given historical data. This is done by making use of types of artificial neural networks, which are able to accurately model non-linear functions (Haykin 1999), such as a battery discharge.

Charkhgard and Farrokhi (2010) uses an artificial neural network trained using charging data taken from a specific charging method where current is pulsed, allowing for better parameter estimation. This data is taken from different stages of the charge cycle, under different current conditions. This leads to quite a complex initial setup requirement, and considerable data processing to extract the training data. This trained model is then used to implement an extended kalman filter (an algorithm for estimating non-linear systems (Julier and Uhlmann 2004)), which is used to generate an estimation of SOC.

Tian et al. (2022b) makes use of a convolutional neural network which is easier to train, requiring arbitrary data from a 1C charge (where the charge current is equal to 1 times the battery capacity in amperes). This was found to be highly accurate when provided with 1C charging data, but requires this charging data specifically to create an estimation of the battery parameters, thus making it unsuitable for online measurement during variable discharge. It also requires the

network to be trained in order to generate estimations, making it computationally infeasible to use in real time on a microcontroller.

Tian et al. (2022a) focuses on battery estimation during "dynamic usage", which is more in line with typical drone discharge conditions. This method combines coulomb counting in the form of Ah counting combined with a neural network SOC estimation using a kalman filter (similar to extended kalman filter mentioned above, but for linear outputs (Julier and Uhlmann 2004)), to attempt to combat issues with using exclusively a neural network in dynamic conditions. While the method is shown to be effective, it requires complicated training, in the form of previous data from expected discharge conditions. As with the other data-driven approaches, the computation required to run this approach in real time is beyond that possible on a microcontroller.

While these approaches are generally successful in their results, they require higher computational power than is available on a microcontroller, as well as initial training of various degrees, with some requiring a considerable amount of training data to produce accurate outputs, considerably complicates setup requirements. Due to these reasons, they are not suitable as an online measurement for a small, lightweight, easy to use application.

3.2.2 Model Based

Tian et al. (2022a), mentioned previously, makes use of what is known as an equivalent circuit battery model to improve its outputs. This method of battery modelling uses knowledge of the underlying chemistry within batteries and their performance to create circuit models made up of electrical components which mirror the performance of batteries. These are set up via parameters extracted from battery performance information, and are overall more simple when compared to electrochemical models, such as that used in Uddin et al. (2016). Electrochemical models aim to directly simulate the underlying chemistry within a battery, and require significant background knowledge to create, and many parameters to instantiate. This makes equivalent circuit modelling more attractive for a user friendly implementation.

Thevenin Model While there are many battery models available, the Thevenin model is considered to be the most widely used (Deng et al. 2023). It requires relatively few parameters compared to electrochemical models, but still has some friction in initial setup, requiring a pulsed charge-discharge experiment to properly identify parameters (Wang et al. 2021).

Tremblay Battery Model The Tremblay battery model was first presented in Tremblay et al. (2007), and presented a model for "dynamic simulation" of hybrid electric vehicles while focusing on ease of use. Since hybrid vehicles have highly volatile power profiles, this paper focuses on modelling dynamic loads, similar to those found in drones. The model itself is based on the electrochemical Shepherd model (Shepherd 1965), but is modified to ensure there is no "algebraic loop", where outputs from the model are required for the next step of the simulation. The parameters required for this model can be easily extracted from a discharge curve often provided by manufacturers, and as such usually does not require user testing of a battery. This project makes use of the updated version of this battery model, presented in Tremblay and Dessaint (2009), where a modified version of the original model is presented, along with dynamic test results in a hybrid application, showing an absolute error generally below 5%.

This model is quick to run as it is represented by a mathematical equation, making it usable on microcontrollers. The parameters are easily extracted from a basic discharge curve, making it easy to set up, and the only required input is the current amount of charge used in Ah, and the current, both of which can be found by implementing current sensing. The only difficulty in finding these inputs is finding the initially used Ah when a new battery is plugged in, but since the model is set up for a specific battery, an estimation for this can be deduced from the initial battery voltage given a current measurement. For the aforementioned reasons, this is the model used in this project.

Mansouri et al. (2017) states that model-based approaches "tend to be computationally complex making their applicability to real life settings questionable", and that they "[rely] on the expert knowledge of the underlying mechanisms". This refers to electrochemical models, however equivalent circuit models are not mentioned in this paper. As described, the Tremblay model is minimally computationally intensive, and once implemented, does not require significant underlying knowledge to instantiate for a specific lithium-ion battery.

The paper itself focuses specifically on online remaining useful life estimation (usable energy stored on the battery), making its goals similar to this project. However, the techniques explored make use of machine learning, which as previously described is infeasible on a small microcontroller. The paper assumes a full onboard computer is available, making use of an Intel NUC computer, but this equipment is not typical in many drones, and when available, it is likely it will have purposes other than battery modelling. Also, the presented results of the paper show considerable errors, with the best method having a mean absolute percentage error of 96.57.

3.3 Cell Difference Estimation

Piao et al. (2015) presents a cell balancing method based on real-time outlier detection. Whilst cell balancing itself is beyond the scope of this project, the method for finding outliers was used to estimate individual cell performance. This works by calculating the z-score of each cell, a representation of its distance from the mean, and then checking whether the overall difference between cells is greater than a defined limit. If so, clustering is used to define cells as outliers or non-outliers. This is slightly adapted within this project to update a running average of cell outlier values, clarifying between positive outliers (where the voltage is above the mean), and negative outliers. This is helpful for diagnosing problems within a series battery pack, as if one cell is consistently below mean voltage it is effectively limiting the performance of the pack, whereas if one cell is consistently above the mean the remaining cells are impacting performance.

4 Design

4.1 Analysis of Requirements

To ensure safety in drone flight, battery monitoring is required, and should be able to give an accurate estimation of the remaining flight time. This should update depending on the performance of the battery, such that in cases where the battery does not perform as expected, the user and/ or flight computer is aware of this lack of performance early in the flight, so that the flight plan can be adjusted accordingly. The values required for this are covered next.

To calculate this flight time, a minimum safe voltage should be defined, which represents the lowest voltage at which the drone should be landed. Alongside this input, the following values must be known:

- Battery Voltage
- Battery Current
- Battery Power
- Battery SOH
- Battery SOC

These must be accurately monitored, and from these, the flight time estimation should be updated based on a typical power profile of the drone in flight.

The SOC and remaining flight time should be somehow communicated to the drone flight computer or user where they will be interpreted and acted upon. Battery SOH is here represented by two values, the maximum capacity of the battery in Ah, and the internal resistance in ohms.

Since capacity used throughout a discharge is non-linear as described in the background chapter, the capacity must be converted to an energy value in Wh. This value can then be used to define the state of charge as the used energy divided by the maximum stored energy.

The initial used energy must be accurately estimated as otherwise even with an accurate SOH estimation the remaining flight time will be skewed.

Battery internal resistance must be estimated as the voltage drop during power usage is dependent on this, meaning that if this is different to the expected value, the flight time will also be affected. This is because the flight time estimation is based on a minimum voltage, which will be reached sooner if the internal resistance is increased, as power usage will remain similar during flight.

Since coulomb counting is required for keeping track of used energy, the processing which takes place must be kept to a minimum to minimise the time between voltage and amperage readings, as this will lead to more reliable power measurements, since the power usage is constantly changing on an infinitely small timescale.

As well as performance estimation, per-cell tracking should be implemented. This involves somehow measuring the performance of individual cells within the battery in order to calculate their individual states, and from this notifying the user of abnormalities within the battery. The knowledge of this can then be used to either swap out a cell, or stop using a battery due to the dangers associated with differing cell performances.

These cover the basic required outputs of the monitor for general drone battery monitoring. There are also several requirements for the project more generally to ensure it is fulfils the current gap in available technology. There are several large categories of drones which are currently not widely equipped with battery monitors. These include:

- Cheap Commercial Drones: These are often built with minimizing cost in mind, and as such tend to omit the use of a battery monitor, since existing solutions are largely proprietary, and developing a new one would be costly.
- Custom Built Drones: Drones which are built from components rather than purchased often do not have battery monitors onboard either, as there is no widely available and easy to use solution available, and adding a monitor would increase weight, decreasing performance.
- Small Drones: Drones are available in very small sizes, and these often fall into one of the two previously mentioned categories leading to a lack of monitoring system.

These categories of drones lead to a range of project requirements:

The system should be easily adaptable to a range of different batteries, to ensure it is usable on drones of different sizes/ types. Whichever monitoring/ estimation technique is used should thus be easy to adapt, and this modification should be possible by the end user.

In terms of size, it should be possible to build the project on a single circuit board, though the prototype may be larger/ on more boards due to usage of development boards which allow for easier prototyping and modification of the project. The single board requirement is so that it can be mounted with minimal difficulty, and to minimise additional wires required to install.

The weight should be kept minimal, if possible to around 10 grams or less, as this would allow the monitor to be used on a wider range of drones, as weight is one of the factors which most impacts flight time since this directly increases the power required for flight. This weight can again be more on the prototype, as this may use larger components than required for ease of construction, but the minimal individual components required should target the lower weight.

The cost of replicating the project should be kept to a minimum by using widely available components, whilst ensuring that quality is not compromised on. The measurements should be accurate and this should be prioritised over cost, however if it is possible to reduce cost this should be done, as this increases the possible adoption of the monitor by making it more viable to include it in cheaper drones or by people with smaller budgets for their projects.

The hardware chosen should also be able to communicate externally, as this ensures that it is possible to integrate it within wider drone flight systems. The communication should be possible with a widely-used communication protocol which is likely to be found on drone flight controllers.

Overall the project should be able to create an accurate estimation of battery performance and a prediction of flight time, and be able to output this data in a usable format, while remaining small, lightweight, cheap, and easy to use to ensure compatibility and suitability for the drones which are most likely to make use of the system.

4.2 Overall Structure

This project consists of monitoring hardware which is able to read the required values and perform processing on them, and two different sets of firmware for the hardware.

The hardware is designed to be connected in series between the main battery connector and the drone power input, and to also connect to the battery balance connector (a smaller connector which contains wires to each cell of the battery). The main series connection allows monitoring of overall battery voltage and current, and the balance connector allows independent monitoring

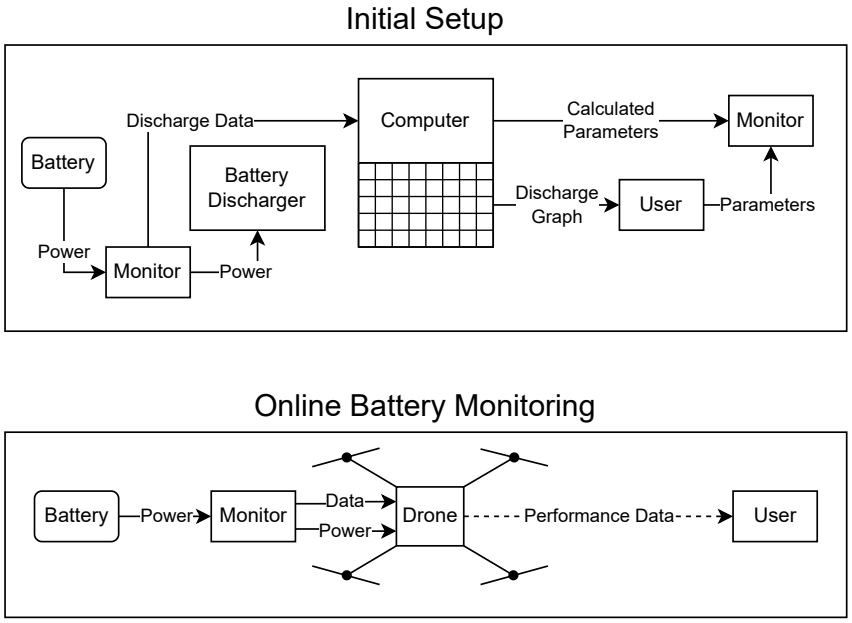


Figure 4.1: *Block diagram of setup and usage of battery monitor.*

of each cell's voltage. The hardware must be able to perform adequate processing for the required firmware described next, and remain very low power to ensure no impact on flight time is caused by excess power usage.

The first firmware is designed to store data directly to a local computer via a program on the computer which reads and saves the data. This data is stored throughout a discharge cycle (one discharge of the battery from full to empty), after which the data which has been stored is used to generate some parameters for the second firmware, and generate a visual discharge graph from which the user can easily derive the remaining required parameters.

The second firmware is used when the monitor is mounted on a drone, to estimate performance in real time. It makes use of a battery model, which is set up based on the data acquired by using the first firmware, as well as an internal resistance estimation and a cell difference monitor. These components are run in real-time during the drone flight, and are combined to create an estimation of the total battery capacity, and a flight time estimation based on this. The monitor then logs these values, and can communicate them to the drone flight controller to be transmitted to the user or directly acted upon.

4.3 Hardware

The hardware chosen for this project must fit the overall project goals of minimising weight, power usage, and cost, while retaining high accuracy, reliability and functionality.

4.3.1 Microcontrollers

As a main processing device, a microcontroller is used. This was chosen for several reasons.

While microcontrollers are much slower than a typical computer, they do not run a full Operating System (OS), and as such have much lower overhead. Due to the decrease in compute power and lack of OS, they also use very little energy, ensuring that the monitor can be powered by

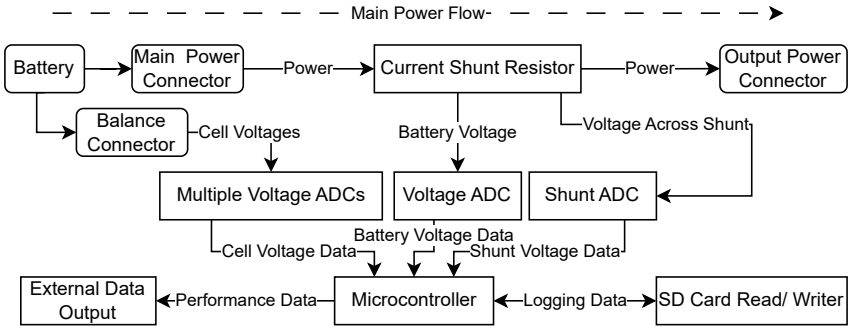


Figure 4.2: *Hardware layout of monitor design.*

the drone battery with minimal impact on battery performance. Another advantage of the lack of OS is that this avoids any extended boot time, since the microcontroller runs its firmware directly with a minimal bootloader rather than requiring an OS to be loaded before performing any tasks. This helps to ensure that there is no added wait time to being able to fly the drone, removing any inconvenience of having the monitor installed.

Using a microcontroller makes integration with existing flight control systems easier, since these systems already often make use of microcontrollers, and as such will primarily support similar communication protocols designed for microcontroller use.

Since microcontrollers are designed for embedded applications, they tend to have much more low-level Input/ Output (IO). This is necessary for this project as this ensures that low-level IO devices such as external ADCs can be used with minimal latency and communication overhead.

Microcontrollers are also much smaller than a typical computer setup, packaging most of the components required into a single integrated chip, rather than having separate chips for CPU, RAM etc. This makes a microcontroller easier to integrate in a system with limited space, such as a drone.

4.3.2 Voltage Measurement

Accurate voltage measurement is integral to the accuracy of the outputs as described previously, and as such it is necessary to include hardware capable of this measurement. There are two possible ways to approach this, both of which could be appropriate for different drone setups and microcontrollers.

Onboard Microcontroller ADC Many microcontrollers include ADCs, meaning that analogue voltages can be read directly without the need for external hardware. Whilst this is the easiest approach and requires no extra hardware, it does have several drawbacks.

The onboard ADC in Microcontrollers typically has several limitations:

- Limited Resolution
- Limited Input Voltage
- Limited Number of ADCs

The resolution of onboard ADCs varies between microcontroller and manufacturer, so some microcontrollers may contain high-resolution ADCs, but several popular models have relatively low resolution. The Arduino Uno for example only has a 10-bit ADC[1], which gives it a step size of 0.016V for a measurement of up to 16.8V (the fully charged voltage of a typical 4-cell lithium-ion battery). This resolution could impact the accuracy of the SOC model, as this

[1] `https://www.best-microcontroller-projects.com/arduino-adc.html`. Accessed 2024-03-13

means a measurement could be as far as 0.008V removed from the true voltage, affecting power calculation. However, if the voltage used is lower, such as if the drone makes use of a 2-cell battery, the resolution could effectively be twice as high, provided that the voltage is scaled correctly for the ADC input. As such, the resolution of onboard ADCs may not be an issue in some setups, and some have higher resolutions, such as the 12-bit ADC found in the ESP32 microcontroller[2].

The voltage range of onboard ADCs also varies, though if this is a limiting factor it can be mitigated. The ADC onboard the Arduino Uno for example has a maximum voltage of 5V[1], however this can be increased by making use of a voltage divider. A voltage divider makes use of two different resistors in series with the input voltage. This then splits the voltage across each resistor by the ratio of its resistance to the total resistance:

$$V_o = \frac{R_2}{R_1 + R_2} \cdot V \tag{4.1}$$

where V_o is the voltage measured across the second resistor, R_x is the resistance of resistor x, and V is the input voltage.

So for example to measure 16.8V with the Arduino Uno, two resistors with resistances 75kOhm and 25kOhm could be used. This would lead to a maximum voltage of 4.2V over the second resistor, safely within the voltage range of the onboard ADC.

Using a voltage divider does however have several drawbacks. As the voltage source must be connected to ground to measure across the resistors, it introduces wasted energy through the resistors. This can be minimised by choosing high resistance values but will still be present. More importantly, this method will introduce some inaccuracy, as the resistance of the resistors will not be perfectly accurate to their ratings, meaning the ratio between them is not exactly as expected. Since the resolution of the ADC is also constant, the effective resolution (the number of volts per ADC step) decreases as the voltage is divided more.

If several cells are being measured this can cause bigger issues. Since the system operates on a single electrical ground, the cell voltages must all be read from 0V, which means that in a series battery the cell voltages are read as the total voltage from 0V to the currently read cell in the series (e.g., for a 4 cell battery which each cell at 3.7V the readings would be 3.7V, 7.4V, 11.1V, 14.8V). This means that if an ADC which cannot read the battery voltage is used, different levels of division must take place on each cell. Due to the inaccuracies mentioned previously, this would lead to inaccurate individual cell performance estimation, potentially giving erroneous outputs.

Finally, microcontrollers may not contain enough ADCs to measure voltages simultaneously. For example while the ESP32 can read analogue voltage on many different IO pins, it only has two actual ADCs[2]. While this is not a problem when reading channels sequentially, it removes the ability to perform more than two readings simultaneously, which means that the time at which readings at some cells are taking may be slightly offset, potentially leading to inaccuracy in comparisons.

External ADC The alternative to an onboard ADC is to use an external one, which connects to the microcontroller using a communication protocol, such as I2C in the case of the INA226[3] While this has the drawback of adding components, and thus increasing size and weight, it means that the performance of the ADC does not depend on the microcontroller, allowing for a wider range of resolutions and input voltages by choosing a different ADC. Using an external ADC chip also allows for several ADCs to be integrated in one chip, and then sending the data from each ADC at the same time using the communication channel. This is demonstrated on the

[2] `https://docs.espressif.com/projects/esp-idf/en/release-v4.4/esp32/api-reference/peripherals/adc.html`. Accessed 2024-03-13

[3] `https://www.ti.com/lit/ds/symlink/ina226.pdf`. Accessed 2024-03-13.

INA3221, which contains 6 different ADCs, with with two different voltage ranges and high resolutions[4].

Overall, both methods could be used depending on the application, for example in small drones powered by a 1-cell battery, an onboard ADC capable of 5V would be adequate if it has a high resolution and would reduce weight, whereas a larger drone using a 4-cell battery would benefit from an external, high voltage, multi-ADC.

This project makes use of several external ADCs, as this allowed for higher voltage and resolution of readings, as well as simultaneous readings of multiple cells.

4.3.3 Current Monitoring

There are two widely available methods available for current monitoring, both of which could be used for this project.

Shunt-Based A current sensing shunt resistor is a conductor with a very low but known resistance. This allows it to be placed in series with a circuit, at which point the current flowing in the circuit can be calculated using Ohm's law:

$$I = \frac{V}{R} \tag{4.2}$$

where I is the measured current, V is the voltage across the shunt resistor, and R is the resistance of the shunt resistor.

This method is quite simple, reliable, and cheap, but does have some drawbacks. Similarly to the issue with voltage dividers described previously, a slight variation in resistance could lead to incorrect readings, in this case of current. This can however be calibrated by using a known accurate current measuring device, and programming an offset and multiplier for the current based on the measured voltage. Resistors also vary in resistance slightly depending on temperature, and since the primary drone current is flowing through the shunt, it will heat up slightly, which could skew readings slightly over a flight.

Hall Effect-Based A Hall effect current sensor functions by the principle that a magnetic field is generated when current is flowing through a wire, which is directly proportional to the magnitude of current. This magnitude of this magnetic field is then measured by an analogue Hall effect sensor, which converts magnetic field strength to voltage, and this measurement is then converted to a usable reading by an integrated circuit. An example chip for this is the Allegro Microsystems ACS712, which can read up to 30A accurately[5] This does add another chip to the hardware, but the additional chip may add functionality, such as temperature compensation. Hall effect current sensors are also typically more expensive, and the power cable must be mounted correctly directly next to the sensor for it to function properly.

Again, both of these sensing techniques are viable for different applications of the monitor, for example in a cheaper setup a current shunt could be used, whereas in a more expensive setup performing longer flights a hall effect sensor would lead to more reliable readings for the duration of the flight.

This project makes use of a shunt-based current sensor, as the tested setup requires very high current sensing capability, and there are no readily available and appropriately sized development boards available with high current hall sensors.

[4] `https://www.ti.com/lit/ds/symlink/ina3221.pdf`. Accessed 2024-03-13

[5] `https://www.allegromicro.com/en/Products/Sense/Current-Sensor-ICs/Zero-To-Fifty-Amp-Integrated-Conductor-Sensor-ICs/ACS712`. Accessed 2024-03-14.

4.3.4 Logging

In order to diagnose battery issues, logging of performance is very helpful. For this, some form of storage must be added, as the internal flash storage found on microcontrollers is typically limited in terms of write cycles and storage space.

External Flash Storage It is possible to add an external flash storage chip to a microcontroller which communicates using a standard serial protocol. This has the advantage of being very small in size and fast to read/ write. An example chip is the W25Q64FV, which contains 64Mb of storage space and communicates over SPI, a high speed serial protocol[6]. The problem with using a flash storage chip is that it presents significant friction in trying to read the data for observation, since the chip is permanently mounted to the monitor, so it would have to be read by the microcontroller and then somehow transmitted to the user.

SD Card Chip A more user-friendly alternative is adding an external chip which can interface with SD cards. While this may be slightly bigger, making use of a removable and widely used storage medium ensures that users can easily extract the logged data when required, by simply removing the SD card and using a reading device.

This project makes use of an external SD card chip as this much better fits the project requirement of ensuring the monitor is easy to use, as users will need no advanced knowledge to retrieve their performance data.

4.3.5 Communication

While logging is useful for post-flight analysis, the primary goal of the monitor is to make use of the in-flight performance data. Since most drones are already equipped with a flight controller which handles flight status and communication, the most widely adaptable method for applying the performance estimation is to relay this information to the flight controller, which can then act upon it and/ or relay it back to the user. As a microcontroller is used, there are several possible communication protocols depending on hardware and software, with three main protocols available when using the Arduino software.

Serial Peripheral Interface (SPI) SPI is a protocol based on a main/ sub device structure, with one device selecting from all connected sub-devices which to communicate with. This protocol requires 4 wires not including power, and is capable of very high communication speeds. It is full-duplex, meaning that both devices can transmit and receive data simultaneously.

Inter-integrated Circuit (I2C) I2C similarly uses a main/sub device structure, however only requires 2 wires for data. As a result, it is somewhat slower, but it is still widely used onboard drones for external sensors. It is also a half-duplex protocol, meaning only one device can send data at any point, which is not an issue with sensors.

Universal Asynchronous Receiver-Transmitter (UART) UART functions asynchronously, unlike the other protocols, and can only connect to one device. It only requires 2 data wires, but since it can only make one connection, several different UART ports are required for several simultaneous connections. This is widely implemented for peripherals in drone flight controllers, as it is full-duplex, and requires less connections than SPI.

As UART is widely implemented for peripherals, and only requires two wires, this is used in this project. This allows for wide compatibility, and since it is full-duplex, future expansions are possible, such as allowing the flight controller to change monitor settings in flight during operation.

[6] `https://www.winbond.com/hq/product/code-storage-flash-memory/serial-nor-flash/?__locale=en&partNo=W25Q64FV`. Accessed 2024-03-14.

4.4 Software

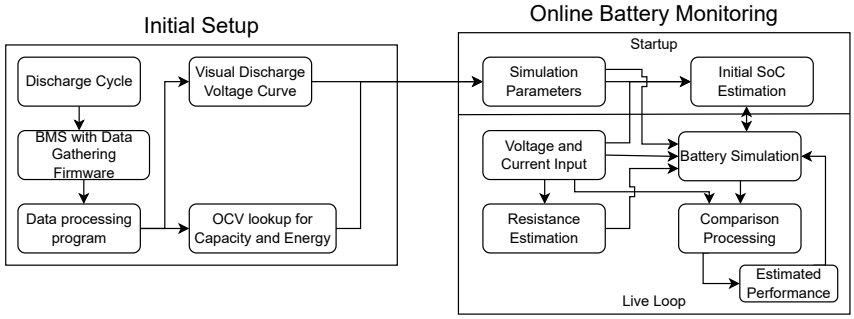


Figure 4.3: *Software component layout.*

The software for this project is split into two sections, as described earlier in the design chapter. One of these sections is for gathering the data required for instantiating the battery model, and the second then makes use of this model in real-time to estimate performance information.

The first section consists primarily functions as a data-logger, reading from sensors and storing and/ or transmitting the values. This makes use of the voltage and current measurements described previously during a discharge cycle, where a battery discharger is used to fully discharge the battery from full. These values are stored as data points including voltage, current, and time. In this project the data is sent via serial connection to a paired program is run simultaneously on a computer, and this program then stores the data in real time in a local CSV file. This ensures that no data is lost in case of any issues with the monitor, and that the data is then immediately ready to be processed.

Once the discharge is complete, the monitor can be disconnected, and the CSV file stored on the computer will have the complete information. This data is then opened in another program on the computer which then plots the data graphically, first applying several filters to ensure that the plotted data is readable. The parameters required for instantiating the battery model are then visually derived from this by a method described later in the battery model section. This program also generates two arrays which correspond to the used battery capacity and energy at certain voltages. These were initially intended to be used as a lookup for directly finding initial SOC, however it was found that the power drawn from the battery, even when the current is kept minimal, has a large enough impact that the results of the discharge test cannot be used as an OCV lookup. Despite this, these arrays are still used for the initial energy consumption estimation, as the updated method for initial SOC estimation only outputs used capacity in mAh, and as such the two arrays can be used in conjunction to find the used energy at a certain amount of used capacity. This is helpful since as described previously, energy usage depends on voltage as well as current, and while the total energy can be calculated from the capacity and nominal voltage, this is not possible for sections of the discharge. This is because as the nominal voltage is the average voltage over the full discharge, if part of the capacity was used from fully charged, the nominal voltage for this section would be higher, and thus for example the energy used in the first half the battery discharge by capacity is more than that in the latter half.

The second section of the software is that which runs in real-time during flight. It is significantly more advanced than the first section, and makes use of several interconnected software components to generate an accurate performance estimation. The most important of these sections is the battery model, which all comparisons/ estimations are based upon.

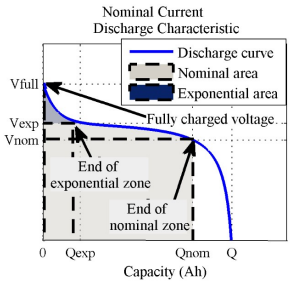


Figure 4.4: *Parameters derived from discharge graph, image from Tremblay and Dessaint (2009).*

4.4.1 Battery Model

The battery model used in this project is the updated Tremblay battery model described by Tremblay and Dessaint (2009). This model focuses on ease-of-use whilst maintaining accuracy, and was tested in the paper to be accurate to within ±5% of the true battery voltage in a range of tests when between 100-20% SOC, and within around ±10% between 20-0%. The decrease in accuracy at lower SOC is mentioned to be "acceptable because it is not recommended to fully discharge a battery", which also applies to drones, where the high power requirements will often cause the voltage to drop below usable levels even above 20% SOC, meaning the accuracy of this section is not as relevant. More importantly, the tests run on the model made use of highly dynamic simulated conditions based on a hybrid fuel-cell car, meaning the model is validated on data with a similar volatility to that found in drone discharging.

Required Parameters In order to set up the battery model, several pieces of information about the battery must be entered. These are derived from a constant-current discharge curve of the battery, which is suggested to be available from manufacturers in the model paper, but in this case can also be generate locally using the steps describe previously. This enables modelling of batteries which may not have manufacturer discharge curves available.

From this discharge curve, several parameters must be entered, as described by Tremblay and Dessaint (2009):

- Fully charged voltage
- Capacity at end of exponential zone
- Voltage at end of exponential zone
- Capacity at end of nominal zone
- Voltage at end of nominal zone
- Maximum capacity

These are derived from the curve as shown in figure 4.4. The fully charged voltage is the OCV of the battery at 100% SOC. The end of the exponential zone is where the voltage of the battery stops initially dropping exponentially and remains relatively linear, with the capacity and voltage corresponding to the x and y axes of the discharge graph. The end of the nominal zone is the point where the voltage drop goes back from linear to exponential, as the battery near zero SOC. Maximum capacity is the total battery capacity available, as represented by the length of the x-axis on the discharge graph.

As well as these parameters, the internal resistance of the battery must also be entered, as this is used to accurately model battery voltage drop during increased power demand. This value may again not be given by all manufacturers, and as such must be found locally. Fortunately, many lithium battery chargers are able to estimate this value during charging, and as such it is quite possible to obtain a good estimation of this value with minimal effort, and with a limited budget, as even if the user does not already own a charger, a new smart charger with internal resistance

estimation is available for less than £40 at the time of writing[7]. Using these entered parameters, a series of equations are used to calculate the values required to run the battery model equation.

4.4.2 In-Flight Simulation Adjustments

The battery model is instantiated twice and run in real-time during the drone flight, and the measured voltage is compared to the estimated voltage from the models. The second model is updated in real time to attempt to fit the real battery performance. This makes use of the voltage and current measurements, as well as the real-time internal resistance estimation.

If the difference between the measured and estimated voltage are greater than a threshold, the second model is updated to use the estimated internal resistance and rerun. If this voltage is within the threshold range, the battery can be assumed to have a different internal resistance to the expected, as changing this setting in the simulation lead to the expected voltage matching where the default model did not.

if the difference between the battery voltage and the second simulation, which makes use of the updated resistance estimate, is still greater than the threshold, the maximum capacity value within the simulation is modified. This is changed to be either higher if the voltage is positively outwith the threshold, and lower if it is negatively outwith.

This forms the basis for the battery capacity estimation, as this process is repeated every time a new measurement is made, thus constantly iterating upon the capacity estimation until the second simulation outputs voltages within a threshold of error of the real voltage readings.

This second simulation, which should after some iterations fit the true performance of the battery, is then used to create a capacity estimate, and from this, derive the expected flight time. This is done by running the simulation with an estimated typical input current, which is created by using a rolling average, and then increasing the used capacity input until the output estimated voltage reaches the minimum flight voltage. The used capacity input is iteratively modified to derive an estimation, increasing if the simulation output voltage is higher than the minimum and decreasing otherwise.

This input capacity represents the estimated total usable capacity until the minimum voltage is reached at typical flight power requirements, based on the rolling average described previously. This capacity number is then converted to an energy value in Wh (by multiplying by the battery nominal voltage), which can then be used in conjunction with the typical power requirements to estimate the flight time in hours, by dividing the power in watts by the energy in watt-hours. This can then be converted as required to minutes or seconds.

4.4.3 Cell Difference Estimation

As individual cell voltages are monitored, it is possible to estimate individual cell performance. This is implemented by making partial use of the method described by Piao et al. (2015). This functions by first calculating checking whether all cells are equal, and if not, calculating the mean of all cells, followed by individual cell z-scores. In the paper the methods are described as using both individual cell voltage and SOC, however as cell SOC is not calculated within this project, all calculations were based only on cell voltage.

Once cell z-scores have been calculated, the "outlier value" of each cell is calculated, by finding the sum of the absolute differences between the current cell and all other cells. In the paper, this is the Euclid-distance as there are two parameters in use, but since only voltage is used in this project it is simply the absolute distance between the two voltage z-scores.

[7] `https://www.hobbyrc.co.uk/toolkitrc-m8p-600w-multifunctional-charger`. Accessed 2024-03-15.

From these outputs, an "abnormality range" is calculated, which is defined as the mean of all cell differences. If the difference between the outlier values of the cell with the lowest and the cell with the highest value ("the range of outlier values" (Piao et al. 2015)) is greater than the abnormality range, then the cells are considered unbalanced.

In the original paper, all cells are assigned to two clusters based on the least and most outlying cell at this point, after which the mean of clusters is repeatedly updated, and cells redistributed based on distance. This then leads to two clusters, one representing the balanced cells, and the other the outlying cells.

In this project however, to minimise computation time and since the total number of cells onboard drones is typically lower than the 40 used in the paper (Piao et al. 2015), the initial most and least outliers are simply used as representative values for typical/ outlying cells, and the other cells are then compared to both of these values, being labelled as an outlier/ typical depending on closer to which value they are.

If a cell is counted as an outlier, its outlier value is added to a rolling total, which can then be divided by the total sum of outlier values to find the average outlier value of the cell compared to all other cells, serving as a good indication of performance, with a high negative outlier value representing a cell which is consistently below the mean cell voltage of the battery, and a positive value representing a cell which is typically higher.

4.4.4 Logging and Communication

Several different output modes are available for the monitor, with three independently selectable modalities. These modalities each have different use cases, and can be used simultaneously to send a range of input and output data from the monitor. In the discharge testing version of software, the only processing performed is basic coulomb counting, so the parameters available for logging are limited to basic outputs.

Serial Communication Serial communication using the UART protocol (as described in 4.3.5) is one modality, which functions by setting a timeout, and making use of a custom handshake. When a timeout above zero is set, the monitor will attempt to connect using this handshake for the duration of the timeout. If it is successful, data will be sent via this connection, otherwise, the serial connection is ignored.

Using this connection for communication with the onboard drone flight controller enables potential further functionality, depending on how the flight controller interprets the data. Since drones often contain some method of relaying data back to the user, the flight controller could include the estimated performance in this data, allowing the user to act based on the performance. Alternatively, if the drone has autonomous features, the flight controller could be programmed to automatically cut a flight short by returning safely to the takeoff point within the estimated available flight time.

OLED Display Small monochrome OLED displays compatible with microcontrollers are widely available and cheap, making them a useful output modality for debugging or ground testing. These communicate using either I2C or SPI (described in 4.3.5), and require low power. The highest resolution widely available is 128×64, which is a relatively safe resolution in terms of processing power and working memory requirements. However, due to this low resolution, only a limited number of parameters can be displayed at once, but these can be changed in code to allow for checking of various sections of the program.

SD Logging Onboard logging is achieved by using the connected SD card reader described in 4.3.4. When enabled, a logging file is created on the SD card, and this is then repeatedly opened, written to, and closed while the monitor is powered. This logging file can then be copied to another device after flight to be able to examine the battery status and performance.

5 Implementation

In this chapter, the specific details of implementation for the prototype monitor created are covered, as well as why certain choices were made, and how the project evolved over time.

5.1 Hardware

The hardware choices were made based on the requirements and design presented in 4. As mentioned in the design chapter, the prototype created is not required to meet all requirements in its current state, but it should be possible to construct it using the same underlying components in a form where the requirements are met.

5.1.1 Microcontroller Choice

The microcontroller used in this project is the Espressif Systems ESP8266. This was chosen as it is widely available, low-cost, well documented, and has the hardware required to interface with all other hardware required for the monitor. This microcontroller is used in a development board format, specifically the NodeMCU, as this has many of its IO pins made available externally, simplifying interfacing and experimentation.

5.1.2 Voltage Sensing Method

As described in 4, there are several possible methods for voltage measurement, however the implemented prototype makes use of several external ADCs for several reasons.

The ESP8266 has one onboard ADC, which has an input voltage range of 0-3.3V, and a 10-bit resolution[1], allowing up to 1024 levels to be read. As this voltage range is very low (unable to read the voltage of even a single cell), and the resolution is considerably limited especially when using a voltage divider, making use of this built-in ADC for accurate monitoring is infeasible.

As such, an external ADC with higher voltage and resolution was required. In this case, two different ADCs were used to enable per-cell voltage measurement of up to four cell batteries, and battery current measurement. The ADCs used were the Texas Instruments INA226 and INA3221 integrated chips. These are chips with similar functions, but the INA3221 has 3 separate channels where the INA226 only has one. These chips are designed for use as power monitors, with the ability to measure both bus voltage (the total input voltage), and voltage drop over a shunt for current measurement. The resolution of the ADC onboard these chips is 16 bit on the INA226 an 13 bit on the INA3221, and the input range for voltage is 0-36V on the INA226[2], and 0-26V on the INA3221[3]. This voltage range is sufficient to measure up to 6-cell lithium batteries, which is adequate for the testing performed in this project. Given the input voltage range this leads to a step size of less than 0.01V for both chips, and with typical error ratings

[1] `https://docs.espressif.com/projects/espressif-esp-faq/en/latest/software-framework/peripherals/adc.html`. Accessed 2024-03-16.

[2] `https://www.ti.com/lit/ds/symlink/ina226.pdf`. Accessed 2024-03-16.

[3] `https://www.ti.com/lit/ds/symlink/ina3221.pdf`. Accessed 2024-03-16.

of 0.002% [2] and 0.01% [3] respectively, this puts any error from the readings well within half of the typical 5% error of the battery simulation, ensuring that any reading errors will not cause a change in estimated capacity. Both of these ADCs are connected to the ESP via I2C, as this minimises wires, and allows for several device connections.

5.1.3 Current Sensing Method

As previously mentioned, the INA226 has a 16-bit ADC which is used both for bus voltage measurement and current sensing via a shunt resistor. These two ADC inputs have different voltage capabilities, as the voltage drop across a shunt voltage is typically minimised by using a low-resistance shunt resistor to minimise wasted energy as current flows through the shunt. Due to this, the INA226 has a much lower input voltage for the shunt differential voltage measurement of ±81.92mv or ±0.08192v. This range is both positive and negative to allow for current flowing in either direction to be read. Using a lower voltage range for this allows for a higher resolution, which in this case allows for a step size of 2.5μv. This allows high resolution readings to be taken, but as a result a suitable shunt must be chosen to ensure the maximum input voltage is not reached at the maximum possible current of the drone.

For testing, this project made use of a high-power drone which may use current in excess of 120A, therefore a suitable shunt was required which would be able to handle this current without overheating or significantly changing in temperature during usage, and have a voltage drop of no more than the ADC limit of 81.92mv at this amperage.

Shunt resistors are available in a range of different resistance and current ratings, with higher current ratings often being larger in size to allow for heat to sufficiently dissipate, and/ or to reduce resistance, minimising power loss. The selected shunt resistor did not mention a resistance rating on the sales page [4], instead only listing the current rating, and a voltage rating, presumably derived from the resistance and maximum current.

The chosen current/ voltage ratings were 200A and 75mv, as 200A is safely above the potential maximum of 120A, with some margin added to minimise generated heat. 75mv is safely within the ADC limit of 81.92mv, but still relatively close to it, ensuring that the resolution of the current readings is high, since if the maximum shunt voltage difference is much lower than the measurable ADC voltage, only part of the total domain of readable ADC values is used, effectively lowering the true resolution. For example if the readable voltage was 100mv, and a 50mv shunt was used, the effective step size for current readings would be halved in comparison to if a 100mv shunt was used.

From the shunt specifications, the shunt resistance can be calculated, which is then used in the code to calculate the current given a certain voltage drop. Since the shunt is rated for 75mv and 200A, the resistance is calculated using Ohm's law to be 0.375mOhm.

For the initial discharge testing section of operation, a different shunt was used, as the discharge test was at a much lower current, and as such the resolution would be inadequate using a 200A compatible shunt. For this, an INA226 development board with an onboard shunt was used, which had a resistance of 0.1ohm, meaning it was able to read up to a maximum of only 0.8A, but reduced the step size from 0.0067A to 0.000025A, ensuring accuracy at the lower currents in use.

5.2 Software

The ESP8266 supports several software development kits (SDKs), allowing for a range of languages to be utilised. This project makes use of the Arduino SDK, as this is widely used in

[4] `https://www.aliexpress.com/item/4000102337577.html?spm=a2g0o.order_detail.order_detail_item.2.66c3f19cFolp7E`. Accessed 2024-03-16.

```
step_mAh_charged = (current_mA * (loop_time * 0.001) * A_ms_to_A_h);
mAh_charged += step_mAh_charged;
mWh_charged += step_mAh_charged * busVoltage_V;
```

Listing 5.1: *The code for per-reading coulomb counting.*

microcontroller projects, and has a considerable existing package library available, simplifying much of the communication within the project. It runs on C++, which was used to implement a class-based program structure, separating distinct sections of software into separate files, and then connecting them in a single main file. The basic structure of an Arduino program consists of two void functions, setup() and loop(). The setup function is run once, allowing any initial setup to be completed, such as instantiation of classes, variable initialisation, communication setup, and any initial computations. After this function has fully executed, the loop function is run infinitely, where the main functionality of the program is typically located. This continues running until the system is powered off.

Platformio was used to flash the firmware to the ESP, and as a package and project management tool. This allowed the simple addition of packages as required, easy uploading of firmware, and splitting of the main program into two separate firmwares which both have access to the same packages. Using this, the initial discharge testing software was implemented as a separately selectable program in Platformio, making it simple to upload this or the monitoring firmware with no extra setup configuration.

For user settings, there is a file titled "defines.h", which contains the information for the battery simulation, and any user-defined values, as well as some system functions and constants.

5.2.1 Discharge Test Software

The first section of software functioned primarily as a logging tool as described in 4. This functions by first instantiating all required classes and setting up the selected communication channels and measurement devices in the setup function. Once this is complete, the loop function is entered. In the loop, the values of the external ADCs are read, which are stored directly in their respective variables, with the current being calculated via the method described previously.

Once all voltages and the current have been read, the coulomb counting is performed, using the measured voltage, current, and the time since the last reading, which is measured in microseconds to maximise the coulomb counting accuracy. This is performed using the code shown in 5.1. As the inputs are in μs and mA, directly multiplying them gives an output in milli-ampere-micro-seconds, which should be converted to a more usable unit type. First, the time is multiplied by 0.001, to convert to milliseconds, after which the overall result is multiplied by a constant, "A_ms_to_A_h", which is equivalent to 1 divided by 1000, then divided by 3600, thus converting milliseconds to hours. As the input current is in milli-amperes rather than amperes, this gives a milli-ampere-hour output. A constant is used here to remove any division from the calculation, increasing execution speed. This resulting value is then added to a running total, "mAh_charged", and multiplied by the current battery voltage to calculate the equivalent mWh used, which is then added to a separate running total, "mWh_charged".

Once the coulomb counting has been performed, the data read from the ADCs, and the calculated used capacity and energy are written to all selected communication methods, using either UART connection, SD logging, or an OLED display, as described in 4.4.4. This process then loops repeatedly, reading data throughout the battery discharge.

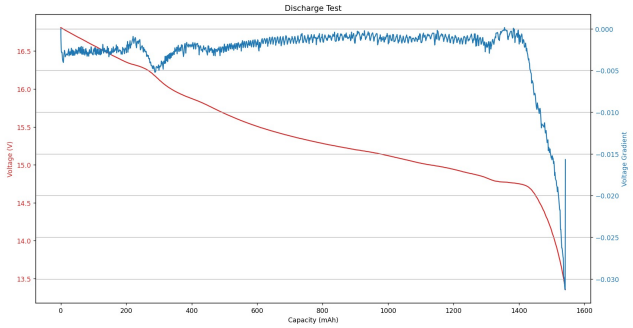


Figure 5.1: *Generated discharge graph after data processing.*

There is a paired program for storing this data on a computer in real time, which makes use of the UART output of the monitor. This is written in python, and tries to connect to the monitor in serial mode. If successful, the data received is repeatedly written to a local .csv file, until either the connection fails or the program is stopped.

Once the data for a discharge test has been obtained, this is processed by another python program, which performs multiple forms of filtering on the input data to crop and smooth it, after which the lookup tables required are extracted, the average discharge current is calculated, and a graph is generated, such as that shown in 5.1, which, using the method shown in 4.4 can be used to extract the required battery parameters. The graph has a second line showing the gradient of the voltage line throughout, to assist the user in finding where the initial and final exponential zones start/ end.

As the discharge current in the test may not be completely constant, and as depending on the time taken for discharge, many thousand values could be stored, filtering this data is required before plotting. This is achieved by using several functions within the plotting program. First, the data is truncated from the beginning to the point where a voltage higher than the minimum battery voltage was detected, to ignore any values taken before the battery was plugged into the monitor, for cases where the monitor was first powered via USB to establish a serial connection before beginning the test. Then, the voltage values are all passed through a simple low pass filter, implemented as a moving average, with a reaction time which can be user-set, allowing for modification if data is more/ less consistent. The output array is then reduced in size by averaging every x values throughout the array, where x is also user selectable, and should be changed depending on the size of the input data. The data is then ready to be graphed, and the lookup tables are generated next.

The two lookup tables are arrays which have length maximum voltage (mV) - minimum voltage (mV), meaning that there is one value for every 0.01v. Two slices of the data array are taken as inputs, the battery voltage, and the used capacity or energy depending on which lookup is being generated. The voltage array is iterated through, and the value of the output array which corresponds to the current voltage in mV is set to the input capacity/ energy at that position. If a position (voltage) is repeated, the output value at that position is the average of all values at that position. If one or more voltage values are skipped, the skipped values are set to the next available value. Finally, if the maximum or minimum voltage were not found in the input, the missing edge output values are set to the closest available voltages. These generated lookup tables can then be input into a user defines file which is later used in the main monitor software.

Once this processing is complete, the graph is generated, and the lookup tables and average current throughout the discharge are printed. This is everything required to set up the battery model, except for the battery internal resistance, which can be found by using a charger as described in 4.4.1.

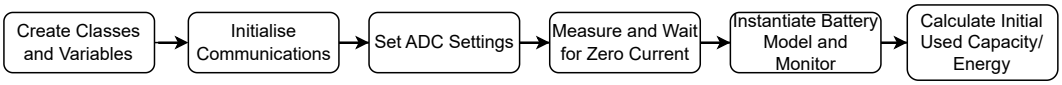


Figure 5.2: *Flow of monitor program setup function initialisation.*

5.2.2 Monitor Program Structure

The second section of the software makes up the battery monitoring and performance prediction. This is also structured using the typical Arduino layout of a setup() function followed by a loop().

The setup function instantiates all necessary classes and objects, sets up the required communication methods, and estimates the initially used capacity by making use of the battery model, as shown in 5.2.

Used Capacity Estimation Initially, the lookup tables generated in the setup stage were used to attempt to directly estimate the used capacity, by finding the value corresponding to the initial voltage, and assuming that the current would be low enough in both the discharge test and initial measurements that it would not majorly affect the battery voltage. However, it was found that due to battery hysteresis, even low discharge currents would have a notable impact on battery voltage. As such this method was modified to instead make use of the battery simulation, as this is able to accurately model the voltage drop at different currents.

To do this, first the program waits until the current is below a user-defined threshold for a user-defined amount of time, to ensure the current is relatively low, minimising error due to difference in internal resistance between the model and the real battery, and to allow the voltage to stabilise. Once the current has been low enough for the time required, the initial used capacity estimate is iteratively increased, and the battery model is run with this value, and the current draw at the current time, until the simulated voltage is less than or equal to the real voltage. At this point the used capacity estimate is considered complete, and is then also converted to a used energy estimate by using the two lookup tables defined during discharge testing. It iterates through the capacity lookup until the capacity in the lookup table is close to the estimated capacity used, at which point the estimated energy used is set to the other lookup table value found at the same index. This gives an estimation of the initial used energy and capacity, but these may be inaccurate if the battery has a different capacity to that which the simulation is based on, so within the later capacity estimation code, the initial estimated used energy and current can be changed based on true performance.

For the last step of the setup, the rolling average values of total current and wattage are initialised to be based on previous data, when available. This data is stored on the SD card in two files, whose titles may be modified by the user.

Once the setup has completed running, the main loop begins, which first reads all ADC values, checking whether any readings have been missed using a method described in TODO: REFER TO WHERE I TALK ABOUT ADC STUFF. Once these values have been read in, logic to decide whether or not the drone is currently in flight is run.

This is based on a user-defined variable, "MIN_FLYING_AMPS", which represents the minimum expected current while the drone is in flight. If the internal state of the monitor is currently not flying and current is above this minimum, the flying state is updated to true. If the state is flying, and the current is below this threshold, a timer is started, which checks if the current is below the threshold for longer than another user-defined variable "MAX_ZERO_CURRENT_TIME_MS". If so, the internal state is again set to not flying.

If the drone is found to be in flight, the rolling average for current and wattage is updated, after which the monitor class is updated, which performs coulomb counting, battery simulation, resistance estimation, and cell difference checking. The simulation is then run locally, using the

estimated usable current, and the estimated current at the minimum usable voltage, calculated by using the rolling average wattage, and dividing this by the minimum usable voltage, to estimate the current requirements when the voltage is at the minimum state. The estimated usable capacity is then updated based on whether the result of this simulation was a voltage above or below the minimum. As the entire loop section runs repeatedly, an estimation is quickly built up without using a local loop here. Based on the usable capacity estimation, a flight time estimation is created, by converting the estimated total capacity to Wh, then subtracting the used capacity from this, and dividing the result by the rolling average wattage. A total flight time is also estimated, by using only the estimated energy value without subtracting the energy used.

After this, the results of the estimation and measurements are sent to all selected output methods, and the loop repeats.

5.2.3 Simulation Implementation

As previously described, this project makes use of the battery model presented by Tremblay and Dessaint (2009). A major part of this project was implementing this model as described in the paper.

The paper presents several different equations for different battery types, but since drones almost exclusively make use of lithium-ion batteries, only the relevant equations are implemented. The model is primarily made up of a single equation, from which several other equations are derived. The model equation requires several inputs and parameters. As an input, it only requires current, filtered current, and used capacity, enabling the simulation to be used to estimate capacity as described previously.

Filtering Since a filtered current is required as an input, this must be derived from the instantaneous measured current over time. The paper does not clarify which type of filtering was used, but does mention that it involves a "time constant" representing the reaction time of the battery. For this project, an exponential moving average was implemented, using the following formula:

$$(1 - \alpha)x + \alpha y \tag{5.1}$$

Where α is the filter constant, x is the initial average, and y is the new value. This filter is configured to make use of a time constant by recalculating the filter constant based on the time delta. The filter constant is then calculated by dividing the time delta by the time constant. The filter is implemented as a class, with the filtering method itself shown in 5.2.

Simulation Parameter Calculation Based on the simulation input values, several parameters need to be calculated to run the simulation. The equation to calculate the battery voltage given inputs i (current), $i*$ (filtered current), and it (used capacity) requires the following parameters on top of the previously described and calculated total capacity, resistance, and curve-derived values:

- A = exponential zone amplitude
- B = exponential zone time constant inverse
- E_0 = battery constant voltage
- K = polarisation constant

as described in Tremblay and Dessaint (2009).

These are calculated by using equations derived from the main voltage estimation equation:

$$V_{batt} = E_0 - R \cdot i - K\frac{Q}{Q - it} \cdot (it + i^*) - K\frac{Q}{Q - it} \cdot it + Aexp(-B \cdot it) \tag{5.2}$$

(Tremblay and Dessaint 2009). There was initially some confusion, as the same letters are used for parameters in the earlier paper (Tremblay et al. 2007), and since the later paper does not

```
float LPF::Update(unsigned long time_delta, float current) {
   float filter_alpha = (float) time_delta / (float) reaction_time_;
   if (filter_alpha >= 1) {
      filtered_current_ = current;
   } else {
      filtered_current_ = filtered_current_ * (1 - filter_alpha) + current *
          filter_alpha;
   }

   return filtered_current_;
};
```

Listing 5.2: *The code for the time-based exponential moving average. filtered_current_ and reaction_time_ are class variables.*

as clearly state the equations required to derive some of these parameters, notably E_0 and K, I initially attempted to use the equations presented in the earlier paper, not realising they had different meanings. This difference was found after some time spent attempting to identify the cause of the difference in simulation outputs in my implementation compared to those presented in either paper. The difference is in the K (polarisation constant) parameter, which was defined differently in the earlier paper, but the original model was found to "produce false results when the current varies" (Tremblay and Dessaint 2009), so the model was updated in the later paper to "extend its validity for variable [...] current" (Tremblay and Dessaint 2009).

The exponential zone amplitude, A, is trivial to calculate, simply being the voltage at the end of the exponential zone subtracted from the full battery voltage. B, the "exponential zone time constant inverse" "can be approximated to 3/Qexp since the energy of the exponential term is almost 0 (5%) after 3 time constants", also making it simple to calculate given Qexp, the exponential zone capacity derived from the discharge graph. Calculating E_0 and K was more complicated, as equations for these are not given directly, rather equations containing them (derived from the discharge equation given certain conditions) are given which must then be rearranged to solve for the relevant parameter. The equations were rearranged as shown:

Original	$V_{full} = E_0 - R \cdot i + A$ (Tremblay and Dessaint 2009)
Rearranged	$E_0 = V_{full} + R \cdot i - A$
Original	$V_{exp} = E_0 - K\frac{Q}{Q-Qexp} \cdot (Qexp + i) - R \cdot i + Aexp(\frac{-3}{Qexp} \cdot Qexp)$ (Tremblay and Dessaint 2009)
Rearranged	$K = \frac{(Q-Qexp)\cdot(E_0 - R\cdot i + Aexp(-B\cdot Qexp) - Vexp)}{Q(Qexp+i)}$

All parameters and inputs have now been covered, except the i term in the above equations, which would be equal to the current during a discharge simulation, but as these equations are calculating parameters its value is not clear. In this case, as the equations are being created based on a constant current discharge graph, i is equal to the constant current used in the graph. There is no filtered current ($i*$) term in either equations, as the first equation is based on full, voltage, where current has only just started flowing, and as such the filtered current is zero, and in the second, as the current is constant, by the end of the exponential zone where the other equation values are based, the filtered current will have reached the constant current (Tremblay and Dessaint 2009).

Structure The battery model was implemented in a class "BattModel", which takes in all the user-defined parameters required for the model during instantiation, and then calculates all derived parameters using the equations described previously. It is then possible to call the "Simulate" function, which takes the used capacity, current, and a filtered current as required, and returns

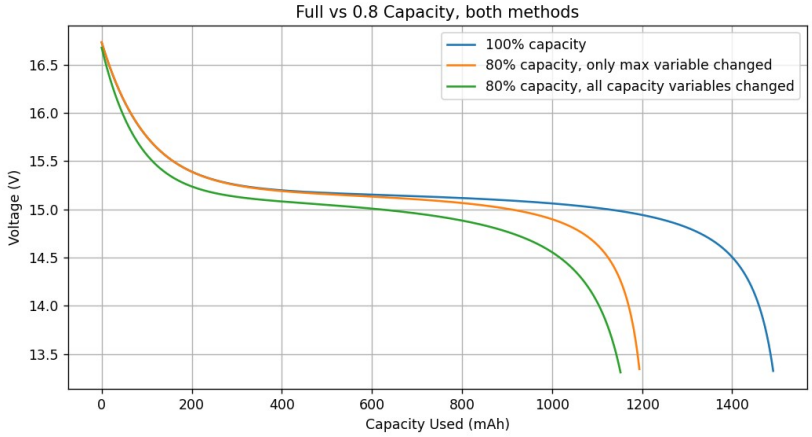


Figure 5.3: *Impact of improperly vs properly changed capacity in simulation of 6.5A discharge on 1500mAh battery.*

the simulated voltage calculated using equation 5.2. The current filtering is kept separate from this class to enable the model to be tested with different currents if required.

To allow simulation adjustment as required for the performance estimation, two public modification functions are implemented; "SetCapacity" and "SetInternalResistance". "SetInternalResistance" functions as a direct setter for the simulation internal resistance, and then recalculates the derived parameters, as this changes the values of E_0 and K. "SetCapacity" contains slightly more logic, as simply changing the maximum capacity would lead to the simulation's discharge curve changing unevenly, since the exponential and nominal zone capacities would remain unchanged. As such, this method calculates the ratio of change between the previous and updated capacity, and then applies this ratio to the exponential and nominal zone capacities also, before recalculating the derived parameters to update given the new inputs. The impact of this method is shown in 5.3.

5.2.4 Internal Resistance Estimation

To be able to differentiate between lack of performance due to low capacity and lack of performance due to high internal resistance, an online internal resistance estimate must be calculated. While there exist previous research papers relating to this (Chiang et al. 2011; Remmlinger et al. 2011), they focus on electric vehicles, and present entire systems for SOC or SOH estimation, meaning their full implementations would be unnecessary. A more related methodology is used by the Ardupilot project[5], an open-source flight controller software which is available for a variety of vehicles, and is widely used onboard multirotor drones. This software has some battery monitoring implemented in the form of basic coulomb counting, and voltage drop compensation via internal resistance estimation. As this code is specifically designed for drones and similar vehicles, and is open-source under the GPL V3 license, allowing reuse, I decided to make use of the code within this project for resistance estimation, using a modified version of the "update_resistance_estimate" function from the "AP_Battmonitor_Backend.cpp" file.

This function works by storing a low-pass filtered reading of the battery current and voltage, and comparing this to instantaneous readings, with the difference between these being used to calculate the effective internal resistance[6]. This function was modified to make use of a time

[5] `https://github.com/ArduPilot/ardupilot/`. Accessed 2024-03-17.

[6] `https://github.com/ArduPilot/ardupilot/blob/master/libraries/AP_BattMonitor/AP_BattMonitor_Backend.cpp`. Accessed 2024-03-17.

delta input rather than an internal timer, as this project already has a main timer for the battery readings, and to add conversion from microseconds to seconds rather than milliseconds as in the original function. Otherwise, the function was left largely untouched, other than replacing functions defined in other sections of the software with either inline or local functions. A class was then created around this function, allowing for class variables to store the required filtered values, and to minimise interference with other sections of the program using similar variables. The class is instantiated without any required inputs, and then takes amperage, voltage, and time delta as an input to update the resistance estimate. A getter is used to extract the resistance, which is then used to update the modifiable simulation as required for the performance estimation.

5.2.5 Cell Difference Estimation

The cell difference estimation is implemented by partially using the algorithm described by Piao et al. (2015), as explained in 4.4.3. This is included in the "BattMonitor" class, which performs all monitoring operations within the main loop, such as coulomb counting, running the battery simulation, updating estimated capacity, running resistance estimation, and calculating cell difference as required.

5.2.6 Real Time Operation

The external ADC chips used support limited forms of internal configuration and calculation, which are used in this project to minimise time where readings are not being taken, by offloading averaging of values onto the ADCs, allowing for the required estimation operations to be performed in the time where the offboard averaging is taking place. The INA chips have several selectable options, including enabling/ disabling certain measurements, setting the "conversion time", which is the time which the ADC takes to measure the voltage, with higher times leading to more reliable readings, and the average number, which is how many readings are averaged before returning a value. Using a combination of these settings, I optimised the time taken to receive a new reading to be low, but high enough to allow all onboard processing to be completed between new readings sent from ADCs.

As the readings taken from the INA226, the total battery voltage and current, are the most important for the estimation code, the timing of the program is primarily based around these. The conversion time for this is set to 2116μs, and the averaging is set to 16 values. As only one ADC is onboard, this means that 32 total readings are taken before a value is available, leading to a total time of 68ms per reading. The INA chips have onboard registers storing the status of the conversion, allowing this value to be checked. At the beginning of each main loop, the loop is blocked until this reading is available, and if it is already available when this point of the loop is reached, a counter for missed values is iterated, as this means that the time for the loop to run was longer than the conversion time, meaning that a reading may have been missed. This general operation is illustrated in 5.4. I tried to implement interrupt-based catching of missed reading by using the INA alert pin, which can be connected to an IO pin on the ESP to trigger an interrupt in real time, however running an interrupt function during other IO operations such as writing to the SD card or display caused errors with the communication methods, presumably as this interrupted the clock of the synchronous SPI and/ or I2C bus.

The INA3221 was configured with different settings, as it was not used to measure current, and as such the timings of its reading were less important, since the coulomb counting did not depend on this. As such, this was set to a lower reading time than the INA226, by using an 1100μs conversion time, with 16 averages. Since the current reading is not required, this can be disabled in the onboard configuration, reducing the total number of reading channels from 6 to 3, meaning a total of 48 readings were taken per average, leading to a total reading time of 53ms, less than the INA226 but still with a high enough conversion time to take accurate readings. This was read after the INA226, ensuring that a reading would always be ready when

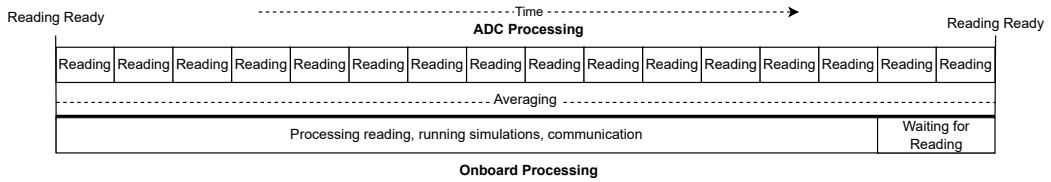


Figure 5.4: *Visualisation of timing with INA226 on-chip averaging, each reading represents reading both bus and shunt voltage.*

required, removing the need to check if it is ready, and ensuring that no execution time is spent additionally waiting for the INA3221 to complete conversion.

These settings gave an effective time "budget" of 68ms to run the complete main loop once. After testing the execution speed of the simulation, it was found to take consistently less than 3.5ms, and as such could be run several times within the loop as required. Recalculating the parameters also took less than 3ms, meaning simulation adjustments could be easily made in time, and as the structure of the program only makes up to 2 changes to the simulation per loop, as an iterative approach is taken, these timings fall well under the maximum available. The serial communication was configured to use a bitrate of 115200, meaning that the amount of data sent through this, less than 2000 bits, would be sent in less than 20ms. Initially, using SD card logging caused sufficient slowdown that readings were consistently missed, and this did not improve with any change in settings. However after some research, it was found that the default Arduino SD library made use of an outdated version of another library, SDFat. Changing the package used to the updated version of this package allowed for a manual speed setting of the SPI bus used to communicate with the SD card, limited to less than half of the total chip clock speed. With the ESP running at 80Mhz, setting this to 39.9Mhz reduced the time taken to write to the SD card sufficiently to complete the loop within the required time, even with all communication methods being in use.

6 Evaluation

6.1 Experimental Setup

To test the implemented prototype, real drone flights were performed, utilising several different batteries to find whether the various predictive functions worked as intended. The package versions used for these tests were as follows.

- SdFat V2.2.2 by Bill Greiman, repository link, used for all SD operations.
- U8g2 V2.35.9 by oliver, repository link, used for OLED display operations.
- INA226_WE V1.2.9 by Wolfgang Ewald, repository link, used for INA226 communication.
- INA3221 V0.0.1 by Tinyu, repository link, used for INA3221 communication.

6.1.1 Test Platform

The platform used for testing was a custom-built racing drone, which made use of 4 series cell lithium batteries. The drone consisted of the following parts:

- Atto Sniper 5 Frame (Product link, Accessed 2024-03-18.)
- iFlight SucceX-E Mini F4 Flight Controller + ESC (Product link, Accessed 2024-03-18.)
- Emax Eco 2207 2400kv Motors (Product link, Accessed 2024-03-18.)
- Runcam Racer 2 FPV Camera (Product link, Accessed 2024-03-18.)
- Happymodel OVX300 Video Transmitter (Product link, Accessed 2024-03-18.)
- Happymodel EP1 Radio Receiver (Product link, Accessed 2024-03-18.)

This presents a typical setup capable of high-power and highly dynamic operation, testing the dynamic capabilities of the monitor whilst using various batteries. The drone is also large and powerful enough to be minimally impacted by the additional weight of the prototype.

6.1.2 Tested Batteries

To test the different functionalities, batteries with different specifications and levels of wear were tested. The batteries were given numbers for testing, to differentiate the results and test consistency among the outputs when using the same battery. There were 2 different models of battery tested, one from the manufacturer Dinogy, with a rated capacity of 1600mAh, and one from ChinaHobbyLine (CNHL), with a rated capacity of 1500mAh. 5 batteries in total were tested, more were planned to be tested however due to a hardware failure further data readings were unable to be recorded. Two of these batteries were the Dinogy batteries, which represented aged batteries in the tests, as these were purchased roughly 4 years prior, and have since been used often, meaning they have been aged both through calendar aging and cyclic aging. Despite their initially slightly higher rated capacity, these now hold less capacity, and have a higher internal resistance than the CNHL batteries. The CNHL batteries were purchased much more recently, and have been minimally used, with at most 2 discharge cycles, so they represent the expected performance. The battery numbers are specified in 6.1.

Battery Number	Battery Type	Battery Status
1	Dinogy 1600mAh 4S	Purchased 2020, well used
2	Dinogy 1600mAh 4s	Same as 1
3	CNHL 1500mAh 4s	Purchased December 2023, almost new
4	CNHL 1500mAh 4s	Same as 3
5	CNHL 1500mAh 4s	Same as 3, model instantiated based on this battery

Table 6.1: *Batteries used in testing.*

Figure 6.1: *Charger showing per-cell internal resistance estimate after charging.*

6.1.3 Test Procedure

To begin with, the model parameters had to be gathered, for which I ran a discharge cycle using a CNHL battery (battery number 5), discharging via an ISDT Q6 Plus smart charger[1]. The discharge current was set to the lowest possible setting of 0.1A, as at this point the lookup was still intended as OCV, but the charger pulsed this current, so the true average was closer to 0.06A. This generated the discharge graph shown in 5.1, as well as the required lookup tables. The battery was then charged again to find the internal resistance via the charger, shown in 6.1. From these results, the following parameters were set on the monitor:

- Internal resistance: 0.0208 Ohms
- Capacity: 1.55 Ah
- Nominal voltage: 14.8 V (from manufacturer information, 3.7V per cell)
- Nominal zone voltage: 14.75 V
- Nominal zone capacity: 1.42 Ah
- Exponential zone voltage: 15.26 V
- Exponential zone capacity: 0.8 Ah
- Current from discharge graph: 0.06340690796660907 A
- Max voltage: 16.8 V (Maximum during discharge test which was used to generate lookups.)
- Min voltage: 13.3 V (Minimum during discharge test which was used to generate lookups.)
- Reaction time: 30 s (Found to be typical by Tremblay and Dessaint (2009).)

The other user-defined parameters were set as follows:

- Max resting amps (maximum amps during initial voltage stabilisation): 1 A
- Min flying amps (minimum amps required to assume a flying state): 1 A
- Capacity step percentage (how far to move the estimated capacity in each modification step): 1 %

[1] `https://www.isdt.co/q6-plus.html?lang=en`. Accessed 2024-03-18.

- Max voltage variance (the allowable difference between real and simulated voltage before modifying capacity): 0.2 V
- Short decay seconds (time divisor for short term low pass filter): 180 s
- Long decay seconds (divisor for long term low pass filter): 1800 s
- Min flying voltage (the minimum allowable voltage in flight): 13.3 V

TODO ADD current calibration methodto implementation. The onboard current shunt resistor was calibrated by applying differing power draws to the battery by using it to charge another battery, then measuring the current with a known accurate multimeter, and writing down the values measured by the monitor and the multimeter. These values were then entered into a spreadsheet designed for current calibration, which calculates the exact voltage multiplier for shunt voltage, and the current offset. The resulting values for the 200A shunt were an offset of −982.815879mA, and a multiplier of 3.7834625mV/10A.

Once all parameters had been set, the monitor was attached to the drone as shown TODO ADD PICTURE, and several test flights were performed with SD logging, to gather results. These flights were typical racing drone flights, following a set course of gates at high speed, with significant current variation, and moments of high power usage. The conditions during these flights were relatively cold, around 7∘C. The flying style and track followed was kept consistent through all flights, though there is some variation in current profile to be expected due to the variable nature of the flights. Some flights were cut short due to crashes, and as such returned only partial discharge data.

6.1.4 Mocking Data

To allow for testing of new software without requiring physical tests to be repeatedly run, mocking was used to feed in existing data to the main program. This was implemented using two mocking classes, one for each external ADC.

The INA226 mocking class reads a CSV file containing time, battery voltage, and shunt voltage from the SD card, and behaves similarly to the real INA226 class from the INA226_WE Arduino package used in the project. This does not implement all functions, but all necessary for the project are implemented, allowing a direct substitution. The mocking class is able to block until a new reading is "available", using the time stamps from the existing data to set the time per reading. It then returns the ADC values at that time stamp, as the real INA would.

The INA3221 mocking class is simpler, as the main loop only calls this INA for reading once, and does not base any timing around it. As such the class reads from a file only containing the first 3 cell voltages, and converts them to the total voltage at that cell, which would otherwise be provided by the INA, returning the current row when called, and moves to the next line once all values on one line are read, assuming these will all be read for each reading, which is done in the main program.

These classes both make use of the same SdFat object from the main program, meaning no unnecessary additional objects are created. The classes are implemented in the main program using C header code to check whether mocking files are defined by the user, and based on this either instantiating the real or mocking classes.

6.2 Results

6.2.1 Capacity Estimation

Battery Num	Expected Performance	Rated Capacity (Ah)	In-Flight Average Estimated Capacity (Ah)	First 20% In-Flight Average Estimated Capacity (Ah)	In-Flight Average Estimated Capacity (Ah) (new sim)	First 20% In-Flight Average Estimated Capacity (Ah) (new sim)
1	Poor	1.6	~2.08	~1.31	~1.14	~1.22
2	Poor	1.6	~2.52, ~2.15	~1.67, ~1.51	~3.94, ~1.05	~1.83, ~1.19
3	As Rated	1.5	~2.44	~1.56	~1.86	~1.53
4	As Rated	1.5	~2.29	~1.48	~1.73	~1.50
5	As Rated	1.5	~2.37	~1.53	~1.48	~1.50

Table 6.2: *Capacities estimated during tests compared to ratings and expectations, (new sim) represents tests on modified simulation.*

6.2.2 Resistance Estimation

Battery Num	Resistance Measured by Charger (Ohms)	In-Flight Average Resistance Measured (Ohms)	First 20% In-Flight Average Resistance Measured (Ohms)
1	0.0339	~0.026	~0.031
2	0.0618	~0.044, ~0.042	~0.051, ~0.072
3	0.0192	~0.017	~0.022
4	0.0208	~0.017	~0.020
5	0.0208	~0.019	~0.027

Table 6.3: *Resistance estimations from charger compared to in-flight estimations. Column 3 is averaged over the whole flight, and column 4 averaged over first 20% of flight values.*

6.2.3 Cell Difference Estimation

Battery Num	Per-Cell Resistance from Charger (milliOhms)	Cell Difference Values
1	8.3, 7.3, 8.9, 9.4	-0.88, 0, 0.04, -0.03
2	15.9, 15.2, 15.6, 15.1	-0.75, 0.1, 0, -0.06 -0.06, 0.34, -0.08, -0.34
3	5.9, 4.1, 6.6, 2.6	-0.02, -0.05, 0.09, -0.52
4	6.9, 4.7, 1.7, 7.5	-0.03, 0, 0.12, -0.44
5	5.0, 5.1, 5.7, 5.0	-0.01, -0.03, 0.35, -0.6

Table 6.4: *Per-cell resistance estimated by charger compared to cell difference values output from monitor.*

6.3 Discussion

6.3.1 Inaccuracies and Model Modification

After initial testing, the capacity estimation results were found to be wildly inaccurate, even for battery 5, which the model was based on, as seen in column 4 of 6.2.1. After inspecting the

real voltage compared to the simulated voltage visually by using a model implemented locally in python with the voltage and current readings found during testing, it was found that the simulated voltage entered the final zone of exponential voltage decrease much earlier than the real voltage, and that the voltage decreased much more than expected during the nominal zone, as seen in 6.2a. Fortunately, the voltage fluctuation as current rapidly changed did seem accurate, validating the general accuracy of dynamic behaviour of the battery model.

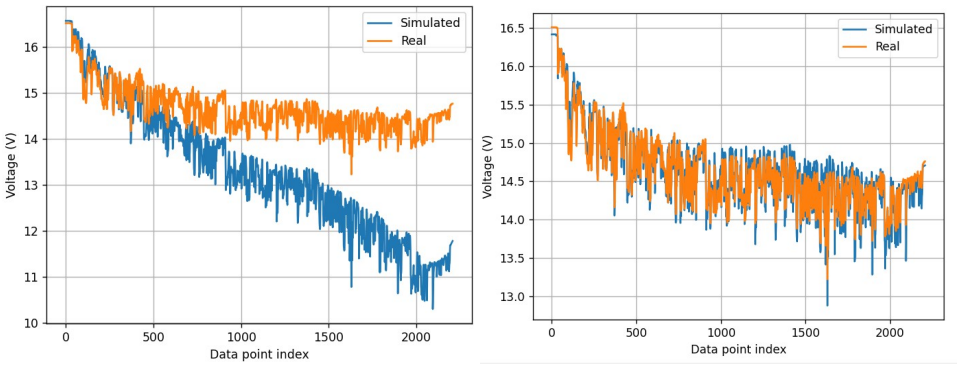
By tweaking the simulation values in this local model, and repeatedly running the simulation, I found values which better fit the real discharge of the modelled battery. The values updated were:

- Nominal zone capacity: from 1.42 to 1.5
- Exponential zone capacity: From 0.8 to 0.3
- Current from discharge graph: From 0.06 to 7

After changing these values, the simulated discharge matched the real discharge much more closely, as seen in 6.2b. I believe the reason that the initial values did not lead to an accurate simulation is due to the excessively low discharge current used in the initial discharge test. Lithium batteries exhibit different behaviours as current differs, which is known and implemented by Tremblay and Dessaint (2009), however, the behaviour seems to vary in different ways to those described in the paper during very low current discharge relative to the battery capabilities. This can be seen in 6.4, where the 0.2C discharge curve appears to have a significantly longer exponential zone, extending to almost 800mAh, where in the 2C discharge, the nominal zone begins clearly before 400mAh. The lower current graphs also slowly decrease for a short section around 400mAh, which is not modelled in the battery model, as this focuses on 3 distinct regions which are assumed to have fixed characteristics. This voltage "bump" is visible in the discharge test performed in this experiment, as seen in 5.1, suggesting that perhaps the discharge current was too low to be accurately used as an input for the Tremblay battery model, as all the example discharge graphs shown in the paper (Tremblay and Dessaint 2009) do not have this voltage bump. This is possibly as the batteries tested were designed for lower discharge rates, as the drone batteries are designed for very high C ratings, having a manufacturer claimed rating of 100C, where many commercial lithium batteries have much lower C ratings, such as the Molicel P28A, which has a C rating of roughly 13.4 [2]. As such, it is possible that the expected input would be a constant discharge graph with a current closer to the typical battery usage than 0.06A on a battery rated to be capable of 150A, as used in this experiment. The exponential/ nominal zone changes lengthen the nominal zone, and minimise the initial exponential zone, whilst changing the current from the discharge graph increases the expected performance of the simulation under higher current usage, ending up with a model which is significantly better matched to the real battery.

Inaccuracy in Test 1 of Battery 2 Even after the modification, the first test involving battery 2 remained highly inaccurate in all figures. After visually inspecting the voltages measured during the discharge of this battery, it was found that a loose connection had caused the voltage of the first cell to repeatedly drop to zero or near-zero values. This in turn caused the total voltage to drop considerably at certain points, which in this test caused the initial voltage to be far lower than the true battery voltage at that point. This meant that the estimated initially used capacity was far higher than reality, and that during the flight, the used capacity exceeded the modelled capacity, causing undefined behaviour. The repeated fluctuation of voltage due to the loose connection also majorly impacted the cell difference value results, as due to the first cell dropping voltage, its calculated score indicated major issues with performance, where it was actually performing similarly to other cells, as seen in the second test of the battery (6.2.3).

[2] `https://www.molicel.com/wp-content/uploads/INR18650P28A-V1-80093.pdf`. Accessed 2024-03-19.



(a) *Graph before modifying simulation.* ***(b)*** *Graph after modifying simulation.*

Figure 6.2: *Graphs showing simulated and real voltage, showing error in model compared to real performance of battery which the model is based on.*

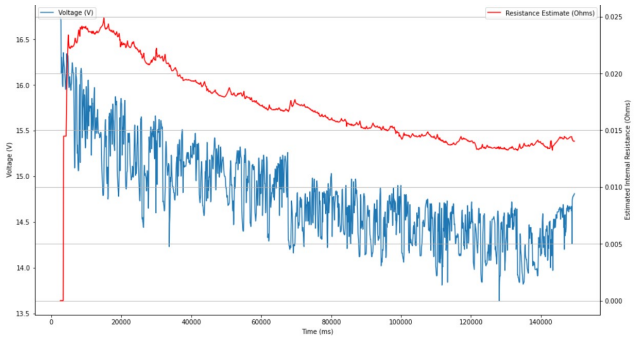


Figure 6.3: *Resistance estimation during discharge, battery 3.*

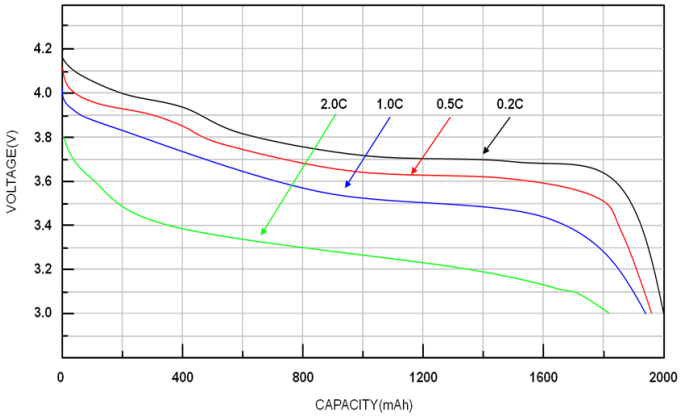


Figure 6.4: *Voltage over discharge using varying current amplitudes. From richtek.com. Accessed 2024-03-19.*

6.3.2 Capacity Estimation Performance

When using the originally generated battery model, the capacity estimation was highly inaccurate, estimating a value far higher than accurate or realistic in all tests. Despite this general inaccuracy, the aged Dinogy batteries (numbers 1 and 2) were generally predicted to be lower capacity than the newer CNHL batteries after the full test, as can be seen in column 4 of 6.2.1. The exception to this is the first test of battery 2, which was estimated as higher than any other battery, however this is likely due to the inaccurate initially used capacity estimation described in 6.3.1, as if the initial used capacity estimation is higher than reality, the battery will appear to perform better throughout the test, since its initial SOC is higher than estimated, skewing the entire test. In the first 20% of a flight, the capacities estimated were generally similar, even for battery 2 during the second test without connection issues. Only battery 1 was initially predicted to be lower than the modelled battery by just under 20%. As such the initial version failed to complete objective 1 (1.2), as it did not accurately warn of lacking capacity within the first 20% of the flight, even on highly aged batteries which perform much worse than the expected model.

Once the model changes had been implemented however, the capacity estimation improved drastically, giving more accurate values in every test, except test 1 of battery 2 due to the issues with this test data. While the estimations for batteries 3 and 4 are still higher than reality, they are closer to real values than before. Visually inspecting this data shows that these batteries appear to have longer exponential zones than the new simulation, and maintained slightly higher voltages during the nominal zone. As such it seems that the capacity estimation may not function well in cases where the general shape of the battery discharge graph does not match the original battery, even when capacity and resistance is similar. Despite this issue, the capacity estimation on the underperforming batteries worked well, estimating a capacity more than 20% less than the model in all cases (except battery 2 test 1), meaning this version successfully completes objective 1 (1.2) by providing early warning in cases of lack of battery capacity. It is also noteworthy that in the first 20% of the flights, all CNHL batteries were estimated to have very similar capacities as expected, showing that perhaps the manually updated model still contains inaccuracies during later sections of the discharge, but performs as expected in the initial exponential discharge section.

6.3.3 Internal Resistance Estimation Performance

The overall in-flight resistance estimated generally matched the expected values in terms of between-battery differences, as the CNHL batteries (3,4,5) all showed similar estimations which were lower than the Dinogy batteries (1,2). Battery 2 was estimated to have a higher internal resistance than battery 1 also, which was expected given the charger measured resistance. The absolute values were not entirely accurate, with the average values during the first 20% of the flight being closest to the real values. This is possibly due to the temperature of the batteries during flight, as increased battery temperatures decrease the internal resistance, and given the high current draw, the batteries heat up considerably during the flight. Visually inspecting the resistance estimate backs up this theory, as the internal resistance estimation peaks initially, before dropping for the rest of the flight, as seen in 6.3. The initial peak in some tests is much higher than the rest of the discharge, likely an inaccurate estimation due to limited data at the beginning of discharge, but the continued decrease in internal resistance after this is found in every discharge test, except for test 1 of battery 2, due to the issues mentioned previously.

As the monitor is accurately able to consistently differentiate batteries of higher or lower resistance, and the profile of the internal resistance matches the expected change due to temperature, objective 3 (1.2) was successfully completed, as by checking the estimated capacity and resistance value, it is clear which is further from the expected value, and which is likely to be causing limited performance. To further confirm this, a test with a battery with a lower capacity but similar internal resistance could be performed, and this was planned, with flight tests completed, but due

to the hardware failure mentioned previously no data was stored.

6.3.4 Cell Difference Estimation Performance

To measure cell performance before testing, each battery was fully charged using the ISDT Q6 Plus smart charger mentioned previously, which provided internal resistance estimations of each cell. These are shown in column 2 of 6.2.3. As I had no way to measure the individual capacity of each cell, this was the only per-cell metric available to measure.

The results generally do not align with the resistance values, with cell 1 of battery 1 estimated to perform worse despite having a similar internal resistance, cell 2 to perform worse and cell 4 to perform better in battery 2 despite similar resistance values, and further unexpected results in batteries 3, 4, and 5. Test 1 of battery 2 again shows highly different results, which do however match the connection issues, as cell 1 is found to have significantly worse performance, as expected due to its repeated zero voltage readings.

However, when comparing estimated performance to mean cell voltage instead, the difference values are more closely correlated, matching the order of cell performance values in all cases. Some of these seem to match better than others, such as cell 4 in battery 5 being considerably lower than the others, whilst others do not match as well, with cell 1 in battery 1 being rated considerably lower than cell 4, despite only being 0.01V lower on average. This could however be due to the typical behaviour of the cell, for example if the cell 4 voltage dropped more during high power draw but was otherwise closer to the other cells in voltage, its mean would be similar whilst the cell would be less of an outlier. This can be seen in 6.5, where cell 1 is generally at a lower voltage, yet cell 4 drops more during power. This suggests cell 1 may have a lower capacity, but a lower internal resistance.

Overall, the cell difference estimation seems so at least partially function, but further testing is required with more knowledge about the individual cell characteristics before the test such as per-cell capacity. As such, the success of objective 2 (1.2) is inconclusive.

Battery Num	Mean Cell Voltages (V)	Cell Difference Values
1	3.6, 3.64, 3.66, 3.61	-0.88, 0, 0.04, -0.03
2	3.42, 3.56, 3.54, 3.47 3.47, 3.53, 3.5, 3.42	-0.75, 0.1, 0, -0.06 -0.06, 0.34, -0.08, -0.34
3	3.7, 3.7, 3.71, 3.68	-0.02, -0.05, 0.09, -0.52
4	3.7, 3.71, 3.72, 3.68	-0.03, 0, 0.12, -0.44
5	3.67, 3.67, 3.69, 3.63	-0.01, -0.03, 0.35, -0.6

Table 6.5: *Mean cell voltages during discharge tests.*

TODO move the: Maybe briefly touch on requirements fulfilled? to implementation!

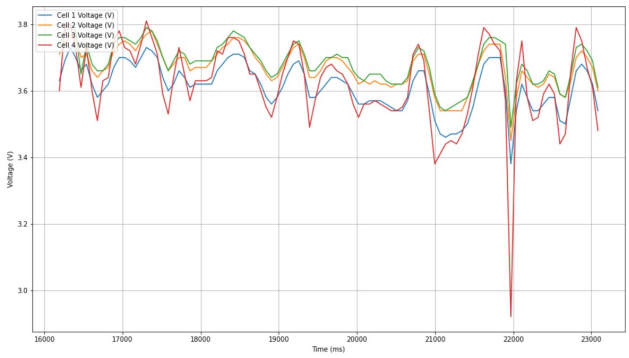


Figure 6.5: *Individual cell voltages during part of battery 1 test.*

7 Conclusion

7.1 Evaluation of Project

This project aimed to create a drone-mounted online battery monitor to accurately estimate performance relative to the expected performance, whilst remaining lightweight, low-cost, and easy to use.

The prototype developed consisted of battery monitoring hardware, two programs for this hardware, and two accompanying computer programs.

The hardware created was relatively large due to making use of primarily development boards rather than a single circuit board housing all required components, however all parts chosen were low-cost, generated accurate outputs, and were lightweight. If the components were to be combined into a single circuit board they would fall below the weight requirement, ensuring that the monitor could be mounted to a range of drones of different sizes. The total price of components for the monitor was £7.88, well below the £10 requirement, and if reduced to bare chips rather than development boards, this price could be even lower. As such, the hardware used did meet all requirements for this project.

The first program for the monitor, used to measure a constant current discharge test, functioned well, accurately measuring voltage and current throughout a discharge, and accurately coulomb counting both capacity and energy used.

The accompanying computer program to read the data and write it to a local file in real time via a serial connection also worked as required, consistently connecting successfully using the defined serial handshake, and storing all data values without error even after running for over 20 hours during a discharge test.

The computer program to process this data also worked well, creating an easily readable graph of this data, after processing it to ensure readability. The lookup tables were also generated as required, and functioned as required for initial used energy estimation onboard the monitor. This program is generally easy to use, but is still python based, requiring the program file to be opened to change settings and select the file path.

The second monitor program, which runs the online battery performance estimation, initially did not work as expected, producing inaccurate outputs in all cases for estimated battery capacity (representing SOH), thus giving inaccurate SOC estimations, however the resistance estimation worked well, matching expected values and accurately identifying the difference in resistance between batteries, thus completing objective 2. The per-cell performance estimation testing was inconclusive, due to a lack of per-cell battery data available to compare with, however it appeared to differentiate well between cells which only dropped voltage during high power compared to those which were at a lower SOC or had a lower capacity.

Once the simulation parameters were manually modified rather than directly taking those generated by the first program, the accuracy of the battery model was improved, and as such the estimations for SOC were much more accurate, and the monitor was able to quickly give accurate estimations of batteries under-performing, successfully completing objective 1. This did however require manual modification of the battery model by checking previous discharge data

and comparing it to a local version of the simulation, something which may not be simple for end-users.

As such, the project was overall successful after manually changing the simulation parameters by comparing real battery performance to the simulation generated by the first program, and iteratively modifying values for best fit. While this increases difficulty of use, it would still be possible for a typical end-user or manufacturer to learn how to manually modify the simulation, and as this only has to be performed once for a certain drone, it would not represent a large hurdle during general use of the monitor.

7.2 Areas for Future Work

7.2.1 Create UI for Model Adjustments

As the model had to be manually modified to obtain accurate outputs, this manual modification could be made easy to use, to enable more novice users to generate accurate simulations. This could be in the form of a GUI based program with sliders to adjust the simulation settings in real time, rerunning the discharge test with each change, and thus making it easy for any users to fit a model well given discharge test data, which could be obtained by SD logging in flight.

7.2.2 Integration with Flight Controller Software

Whilst the monitor outputs the data required for further integration in a protocol implemented by flight controllers, currently no flight controller software supports this specific data, as it must be setup to know which data to expect and in which format. As such, open-source flight controller software such as Ardupilot [1] or Betaflight [2] could be modified to allow this as a direct input on an available UART connection, and these values could then be relayed to the user, or used to allow dynamic flight planning based on real battery performance.

7.2.3 Printed Circuit Board Design

In the current hardware state, the project is difficult to mount, and much larger than the size of the individual components due to each being mounted on a separate circuit board. As such, a next step for this project would be to create a circuit board design to mount all components and connect them as required, whilst having IO pins available for the battery and external communication as required. This would significantly lower weight and size, and could lead to a lower cost as only the individual chips would be required.

[1] https://github.com/ArduPilot/ardupilot/
[2] https://github.com/betaflight/betaflight/

A Appendices

Table A.1: *Full processed outputs from running monitor on real data. (new) denotes runs on modified simulation parameters.*

Test Number	Battery Number	Average Estimated Capacity (Ah)	20% Estimated Capacity (Ah)	Average Absolute Error to Simulation (V)	Average Error to Simulation (V)	Average Internal Resistance Estimate (Ohm)	20% Average Internal Resistance Estimate (Ohm)	Cell Difference Values
1	1	~2.08	~1.31	~1.02	~0.83	~0.026	~0.031	-0.88, 0, 0.04, -0.03
1 (new)	1	~1.14	~1.22	~0.32	~-0.31	~0.026	~0.031	-0.88, 0, 0.04, -0.03
2	5	~2.37	~1.53	~1.39	~1.36	~0.019	~0.027	-0.01, -0.03, 0.35, -0.6
2 (new)	5	~1.48	~1.50	~0.08	~-0.02	~0.019	~0.027	-0.01, -0.03, 0.35, -0.6
3	2	~2.52	~1.67	~0.74	~0.53	~0.044	~0.051	-0.75, 0.1, 0, -0.06
3 (new)	2	~3.94	~1.83	~6.51	~-2.64	~0.044	~0.051	-0.75, 0.1, 0, -0.06
4	3	~2.44	~1.56	~1.45	~1.42	~0.017	~0.022	-0.02, -0.05, 0.09, -0.52
4 (new)	3	~1.86	~1.53	~0.15	~0.12	~0.017	~0.022	-0.02, -0.05, 0.09, -0.52
5	4	~2.29	~1.48	~1.21	~1.16	~0.017	~0.020	-0.03, 0, 0.12, -0.44
5 (new)	4	~1.73	~1.50	~0.20	~0.16	~0.017	~0.020	-0.03, 0, 0.12, -0.44
6	2	~2.15	~1.51	~1.25	~0.74	~0.042	~0.072	-0.06, 0.34, -0.08, -0.34
6 (new)	2	~1.05	~1.19	~0.71	~-0.71	~0.042	~0.072	-0.06, 0.34, -0.08, -0.34

7 Bibliography

M. Charkhgard and M. Farrokhi. State-of-Charge Estimation for Lithium-Ion Batteries Using Neural Networks and EKF. *IEEE Transactions on Industrial Electronics*, 57(12):4178–4187, Dec. 2010. ISSN 1557-9948. doi: 10.1109/TIE.2010.2043035. URL https://ieeexplore.ieee.org/document/5416277. Conference Name: IEEE Transactions on Industrial Electronics.

Y.-H. Chiang, W.-Y. Sean, and J.-C. Ke. Online estimation of internal resistance and open-circuit voltage of lithium-ion batteries in electric vehicles. *Journal of Power Sources*, 196(8): 3921–3932, Apr. 2011. ISSN 0378-7753. doi: 10.1016/j.jpowsour.2011.01.005. URL https://www.sciencedirect.com/science/article/pii/S0378775311000772.

T. R. Crompton. *Battery reference book*. Newnes, Oxford, England;Boston;, 3rd edition, 2000. ISBN 9780750646253;075064625X;9781493302840;1493302841;.

D. Deng, J. Qiao, J. Qi, S. Wang, S. Jin, X. Xiao, X. Hao, and Y. Shang. Chapter 6 - Equivalent modeling and parameter identification of power lithium-ion batteries. In S. Wang, K. Liu, Y. Wang, D.-I. Stroe, C. Fernandez, and J. M. Guerrero, editors, *State Estimation Strategies in Lithium-ion Battery Management Systems*, pages 95–124. Elsevier, Jan. 2023. ISBN 978-0-443-16160-5. doi: 10.1016/B978-0-443-16160-5.00001-9. URL https://www.sciencedirect.com/science/article/pii/B9780443161605000019.

A. Densmore and M. Hanif. Determining battery SoC using Electrochemical Impedance Spectroscopy and the Extreme Learning Machine. In *2015 IEEE 2nd International Future Energy Electronics Conference (IFEEC)*, pages 1–7, Nov. 2015. doi: 10.1109/IFEEC.2015.7361603. URL https://ieeexplore.ieee.org/abstract/document/7361603.

D. Doerffel and S. A. Sharkh. A critical review of using the Peukert equation for determining the remaining capacity of lead-acid and lithium-ion batteries. *Journal of Power Sources*, 155(2): 395–400, Apr. 2006. ISSN 0378-7753. doi: 10.1016/j.jpowsour.2005.04.030. URL https://www.sciencedirect.com/science/article/pii/S0378775305007093.

M. Galeotti, L. Cinà, C. Giammanco, S. Cordiner, and A. Di Carlo. Performance analysis and SOH (state of health) evaluation of lithium polymer batteries through electrochemical impedance spectroscopy. *Energy*, 89:678–686, Sept. 2015. ISSN 0360-5442. doi: 10.1016/j.energy.2015.05.148. URL https://www.sciencedirect.com/science/article/pii/S0360544215007756.

S. S. Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall International, Upper Saddle River, N.J, 2nd international edition, 1999. ISBN 0132733501;9780132733502;.

J. M. Hildebrand. Consumer drones and communication on the fly. *Mobile Media & Communication*, 7(3):395–411, Sept. 2019. ISSN 2050-1579. doi: 10.1177/2050157919850603. URL https://doi.org/10.1177/2050157919850603. Publisher: SAGE Publications.

Y.-M. Jeong, Y.-K. Cho, J.-H. Ahn, S.-H. Ryu, and B.-K. Lee. Enhanced Coulomb counting method with adaptive SOC reset time for estimating OCV. In *2014 IEEE Energy Conversion Congress and Exposition (ECCE)*, pages 1313–1318, Sept. 2014. doi: 10.1109/ECCE.2014.6953989. URL https://ieeexplore.ieee.org/abstract/document/6953989. ISSN: 2329-3748.

S. Julier and J. Uhlmann. Unscented Filtering and Nonlinear Estimation. *Proceedings of the IEEE*, 92(3):401–422, Mar. 2004. ISSN 0018-9219. doi: 10.1109/JPROC.2003.823141. URL http://ieeexplore.ieee.org/document/1271397/.

Kapustina, Larisa, Izakova, Natalia, Makovkina, Elizaveta, and Khmelkov, Michail. The global drone market: main development trends. *SHS Web Conf.*, 129:11004, 2021. doi: 10.1051/shsconf/202112911004. URL https://doi.org/10.1051/shsconf/202112911004.

B. Kim, J. Jung, H. Min, and J. Heo. Energy Efficient and Real-Time Remote Sensing in AI-Powered Drone. *Mobile Information Systems*, 2021:e6650053, Apr. 2021. ISSN 1574-017X. doi: 10.1155/2021/6650053. URL https://www.hindawi.com/journals/misy/2021/6650053/. Publisher: Hindawi.

R. Korthauer. *Lithium-Ion Batteries: Basics and Applications*. Springer, Aug. 2018. ISBN 978-3-662-53071-9. Google-Books-ID: ll1oDwAAQBAJ.

Z. Lu, X. L. Yu, L. C. Wei, F. Cao, L. Y. Zhang, X. Z. Meng, and L. W. Jin. A comprehensive experimental study on temperature-dependent performance of lithium-ion battery. *Applied Thermal Engineering*, 158:113800, July 2019. ISSN 1359-4311. doi: 10.1016/j.applthermaleng.2019.113800. URL https://www.sciencedirect.com/science/article/pii/S1359431118377068.

S. S. Mansouri, P. Karvelis, G. Georgoulas, and G. Nikolakopoulos. Remaining Useful Battery Life Prediction for UAVs based on Machine Learning*. *IFAC-PapersOnLine*, 50(1):4727–4732, July 2017. ISSN 2405-8963. doi: 10.1016/j.ifacol.2017.08.863. URL https://www.sciencedirect.com/science/article/pii/S2405896317313253.

J. Meng, M. Ricco, G. Luo, M. Swierczynski, D.-I. Stroe, A.-I. Stroe, and R. Teodorescu. An Overview and Comparison of Online Implementable SOC Estimation Methods for Lithium-Ion Battery. *IEEE Transactions on Industry Applications*, 54(2):1583–1591, Mar. 2018. ISSN 1939-9367. doi: 10.1109/TIA.2017.2775179. URL https://ieeexplore.ieee.org/abstract/document/8114258. Conference Name: IEEE Transactions on Industry Applications.

K. Movassagh, S. A. Raihan, and B. Balasingam. Performance Analysis of Coulomb Counting Approach for State of Charge Estimation. In *2019 IEEE Electrical Power and Energy Conference (EPEC)*, pages 1–6, Oct. 2019. doi: 10.1109/EPEC47565.2019.9074781. URL https://ieeexplore.ieee.org/document/9074781. ISSN: 2381-2842.

K. Movassagh, A. Raihan, B. Balasingam, and K. Pattipati. A Critical Look at Coulomb Counting Approach for State of Charge Estimation in Batteries. *Energies*, 14(14):4074, Jan. 2021. ISSN 1996-1073. doi: 10.3390/en14144074. URL https://www.mdpi.com/1996-1073/14/14/4074. Number: 14 Publisher: Multidisciplinary Digital Publishing Institute.

B. Pattipati, B. Balasingam, G. V. Avvari, K. R. Pattipati, and Y. Bar-Shalom. Open circuit voltage characterization of lithium-ion batteries. *Journal of Power Sources*, 269:317–333, Dec. 2014. ISSN 0378-7753. doi: 10.1016/j.jpowsour.2014.06.152. URL https://www.sciencedirect.com/science/article/pii/S037877531401026X.

C. Piao, Z. Wang, J. Cao, W. Zhang, and S. Lu. Lithium-Ion Battery Cell-Balancing Algorithm for Battery Management System Based on Real-Time Outlier Detection. *Mathematical Problems in Engineering*, 2015:1–12, 2015. ISSN 1024-123X, 1563-5147. doi: 10.1155/2015/168529. URL http://www.hindawi.com/journals/mpe/2015/168529/.

S. Piller, M. Perrin, and A. Jossen. Methods for state-of-charge determination and their applications. *Journal of Power Sources*, 96(1):113–120, June 2001. ISSN 0378-7753. doi: 10.

1016/S0378-7753(01)00560-2. URL https://www.sciencedirect.com/science/article/pii/S0378775301005602.

A. Plioutsias, N. Karanikas, and M. M. Chatzimihailidou. Hazard Analysis and Safety Requirements for Small Drone Operations: To What Extent Do Popular Drones Embed Safety? *Risk Analysis*, 38(3):562–584, 2018. ISSN 1539-6924. doi: 10.1111/risa.12867. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/risa.12867. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/risa.12867.

S. Qiu, Z. Chen, M. A. Masrur, and Y. L. Murphey. Battery hysteresis modeling for state of charge estimation based on Extended Kalman Filter. In *2011 6th IEEE Conference on Industrial Electronics and Applications*, pages 184–189, June 2011. doi: 10.1109/ICIEA.2011.5975576. URL https://ieeexplore.ieee.org/abstract/document/5975576. ISSN: 2158-2297.

J. Remmlinger, M. Buchholz, M. Meiler, P. Bernreuter, and K. Dietmayer. State-of-health monitoring of lithium-ion batteries in electric vehicles by on-board internal resistance estimation. *Journal of Power Sources*, 196(12):5357–5363, June 2011. ISSN 0378-7753. doi: 10.1016/j.jpowsour.2010.08.035. URL https://www.sciencedirect.com/science/article/pii/S0378775310013534.

M. Shafiee, Z. Zhou, L. Mei, F. Dinmohammadi, J. Karama, and D. Flynn. Unmanned Aerial Drones for Inspection of Offshore Wind Turbines: A Mission-Critical Failure Analysis. *Robotics*, 10(1):26, Mar. 2021. ISSN 2218-6581. doi: 10.3390/robotics10010026. URL https://www.mdpi.com/2218-6581/10/1/26. Number: 1 Publisher: Multidisciplinary Digital Publishing Institute.

C. M. Shepherd. Design of primary and secondary cells: Ii . an equation describing battery discharge. *Journal of The Electrochemical Society*, 112(7):657, jul 1965. doi: 10.1149/1.2423659. URL https://dx.doi.org/10.1149/1.2423659.

J. Tian, R. Xiong, J. Lu, C. Chen, and W. Shen. Battery state-of-charge estimation amid dynamic usage with physics-informed deep learning. *Energy Storage Materials*, 50:718–729, Sept. 2022a. ISSN 2405-8297. doi: 10.1016/j.ensm.2022.06.007. URL https://www.sciencedirect.com/science/article/pii/S2405829722003154.

J. Tian, R. Xiong, W. Shen, J. Lu, and F. Sun. Flexible battery state of health and state of charge estimation using partial charging data and deep learning. *Energy Storage Materials*, 51:372–381, Oct. 2022b. ISSN 2405-8297. doi: 10.1016/j.ensm.2022.06.053. URL https://www.sciencedirect.com/science/article/pii/S2405829722003683.

O. Tremblay and L.-A. Dessaint. Experimental Validation of a Battery Dynamic Model for EV Applications. *World Electric Vehicle Journal*, 3(2):289–298, June 2009. ISSN 2032-6653. doi: 10.3390/wevj3020289. URL http://www.mdpi.com/2032-6653/3/2/289.

O. Tremblay, L.-A. Dessaint, and A.-I. Dekkiche. A Generic Battery Model for the Dynamic Simulation of Hybrid Electric Vehicles. In *2007 IEEE Vehicle Power and Propulsion Conference*, pages 284–289, Sept. 2007. doi: 10.1109/VPPC.2007.4544139. URL https://ieeexplore.ieee.org/abstract/document/4544139. ISSN: 1938-8756.

K. Uddin, S. Perera, W. D. Widanage, L. Somerville, and J. Marco. Characterising Lithium-Ion Battery Degradation through the Identification and Tracking of Electrochemical Battery Model Parameters. *Batteries*, 2(2):13, June 2016. ISSN 2313-0105. doi: 10.3390/batteries2020013. URL https://www.mdpi.com/2313-0105/2/2/13. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.

S. Wang, Y. Fan, D.-I. Stroe, C. Fernandez, C. Yu, W. Cao, and Z. Chen. Chapter 5 - Battery state-of-charge estimation methods. In S. Wang, Y. Fan, D.-I. Stroe, C. Fernandez, C. Yu, W. Cao, and Z. Chen, editors, *Battery System Modeling*, pages 157–198. Elsevier, Jan. 2021. ISBN 978-0-323-90472-8. doi: 10.1016/B978-0-323-90472-8.00009-3. URL https://www.sciencedirect.com/science/article/pii/B9780323904728000093.