

REMAINING USEFUL LIFE PREDICTION AND LIFECYCLE OPTIMIZATION OF LITHIUM-ION BATTERIES

*Submitted in partial fulfillment of the
Requirement for the award of the degree of*

BACHELOR OF TECHNOLOGY IN ELECTRICAL ENGINEERING BY

Aditya Ray (20117006)
Pankaj Agrawal (20117068)
Pragya Churendra (20117071)

Under the guidance of
DR. LALIT KUMAR SAHU
Assistant Professor



NOVEMBER 2023

DEPARTMENT OF ELECTRICAL ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY
RAIPUR 492010 (INDIA)

CERTIFICATE

I hereby certify that the work which is presented in B.Tech. Major Project Report entitled “**Remaining Useful Life Prediction and Lifecycle Optimization of Lithium Ion Batteries**” in partial fulfillment of the requirements for the award of the Bachelor of Technology in Electrical Engineering and submitted to the Department of Electrical Engineering of National Institute of Technology Raipur is an authentic record of my own work carried out during a period from July 2023 to November 2023 under the supervision of DR. Lalit Kumar Sahu, Assistant professor, Department of Electrical Engineering, NIT Raipur. The matter in this thesis has not been submitted by me for the award of any other degree elsewhere.

This is to certify that the above statement made by the candidates is correct to the best of my knowledge

Signature

Dr. (Mrs.) Anamika Yadav
Head of Department
Electrical Engineering
National institute of Technology Raipur

Signature of Supervisor

Dr. Lalit Kumar Sahu
Assistant Professor
Electrical Engineering
National institute of Technology Raipur

DECLARATION BY THE CANDIDATES

We, the undersigned, solemnly declare that the report of the major project entitled “**Remaining Useful Life Prediction and Lifecycle Optimization of Lithium Ion Batteries**”, Submitted to the **Department of Electrical Engineering, National Institute of Technology, Raipur, November 2023** for the **Bachelor of Technology** degree in **Electrical Engineering** is based on our work carried out during the course of our study under the supervision and guidance of **Dr. Lalit Kumar Sahu**. We further declare that to the best of our knowledge and belief the thesis does not contain any part of any work which has been submitted for the award of any other degree or certificate either in this institute or any other university without proper citation.

Mr. Aditya Ray
(20117006)

Ms. Pragya Churendra
(20117071)

Mr. Pankaj Agrawal
(20117068)

PLAGIARISM DECLARATION

We understand that plagiarism is defined as any one or the combination of the following:

- Uncredited verbatim copying of individual sentences, paragraphs or illustrations (such as graphs, diagrams, etc.) from any source, published or unpublished including the internet.
- Uncredited improper paraphrasing of pages or paragraphs (changing a few words or phrases or rearranging the original sentence order).
- Credited verbatim copying of a major portion of a paper (or thesis chapter) without clear declaration of who did wrote what (Source: IEEE, The Institute, and Dec.2004)

We have made sure that all the ideas, expressions, graphs, diagrams, etc., that are not a result of our work, are properly credited. Long phrases or sentences that had to be used verbatim from published literature have been clearly identified using quotation marks. I affirm that no portion of my work can be considered as plagiarism, and I take full responsibility if such a complaint occurs. I understand fully well that the guide of the thesis may not be in a portion to check for the possibility of such incidences of plagiarism in this body of work.

Mr. Aditya Ray
(20117006)

Ms. Pragya Churendra
(20117071)

Mr. Pankaj Agrawal
(20117068)

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and a deep sense of respect to our beloved Director, for his continuous support and guidance in completing this certification course at the institute.

We express our heartfelt thanks to our Head of the Department, **Dr. (Mrs.) A. Yadav**, for providing us with the necessary infrastructure and giving us the freedom to carry out the certification course.

Special appreciation goes to our project guide, **Dr. Lalit Kumar Sahu**, for his invaluable guidance and mentorship throughout the course.

We would also like to express our gratitude to **Mr. Saket** for their valuable insights and contributions to our project.

Finally, a big thank you to all the teaching and non-teaching staff of the department for their valuable suggestions and **Drives Lab** for their support and resources .

Mr. Aditya Ray
(20117006)

Ms. Pragya Churendra
(20117071)

Mr. Pankaj Agrawal
(20117068)

ABSTRACT

In the pursuit of eco-friendly transportation, electric vehicles (EVs) face significant hurdles like limited range, slow charging, and expensive battery replacements. This project strives to revolutionize EVs by harnessing the capabilities of machine learning to optimize the efficiency and lifespan of lithium-ion batteries, the core components of electric cars. The primary focus lies in smartly managing charging and discharging cycles through the implementation of advanced algorithms. By fine-tuning these cycles, we aim to enhance battery performance, extending both range and longevity. Additionally, the project endeavors to predict the remaining useful life of lithium-ion batteries, a critical aspect for users planning and budgeting for replacements. To achieve these objectives, a dedicated testbed will be established, generating a robust dataset of operational parameters. This dataset will serve as the foundation for training machine learning models, offering accurate insights into battery health. Through this amalgamation of data-driven precision and cutting-edge algorithms, the project aspires to make electric vehicles more reliable, cost-effective, and enticing for a broader audience, contributing to the global shift towards sustainable and green transportation.

CONTENTS

CERTIFICATE	II
DECLARATION BY THE CANDIDATE	III
PLAGIARISM DECLARATION	IV
ACKNOWLEDGEMENT	V
ABSTRACT	VI
 CHAPTER 1: INTRODUCTION TO BATTERY TECHNOLOGY	 1-16
1.1 Introduction to DC Storage	1
1.2 Different Types of Batteries	2
1.3 Batteries Terminologies	3
1.4 Lithium Ion Battery	5
1.4.1 Chemical Aspects	5
1.4.2 Ideal and Practical Characteristics	6
1.4.3 Applications and Advantages	7
1.5 State Of Charge	
1.5.1 Methods for SOC Estimation	8
1.5.2 More about Coulomb Counting Method	9
1.5.3 Advantages of Coulomb Counting	11
1.5.4 Estimation of SOC in Python	12
1.6 Overview of Ev's	13
1.7 Energy Storage Challenge for EVs	14
1.8 Lithium-ion Batteries for Electric Vehicles	14
1.9 Use of Machine Learning Methods for optimizing Li-ion Batteries	15
 CHAPTER 2: REMAINING USEFUL LIFE ESTIMATION AND LIFE CYCLE OPTIMIZATION	 17-29
2.1 Introduction	17
2.2 Problem Statement	17
2.3 Proposed Solution	17
2.4 Operational Parameter Dataset Generation	19
2.5 Methodology	19
2.6 The Hardware- Lithium-ion Data Logging Device	
2.6.1 Circuit Arrangement	19
2.6.2 Components	20
2.7 The Firmware	22
2.8 Data Collection and Processing in Raspberry-Pi	24
	28
 CHAPTER 3: WEB APPLICATION - BACKEND AND FRONTEND	 30-33
3.1 The Backend	30
3.2 The Database	30

3.2.1 Database Schema and Creation	31
3.3 The Web Server and APIs	33
3.4 The Frontend	33
CHAPTER 4: PREDICTIVE MAINTENANCE AND RUL	35-40
4.1 Overview	35
4.2 Advantages Of Predictive Maintenance	36
4.3 Remaining Useful Life (RUL)	36
4.4 Advantages Of Remaining Useful Life (RUL)	37
4.5 Remaining Useful Life (RUL) Estimation Of Li-Ion Batteries	38
4.6 Life Cycle Optimization Of Li-Ion Batteries	39
CHAPTER 5: COMPARATIVE STUDY OF DIFFERENT MODELS USED FOR RUL PREDICTION	41-52
5.1 Dataset Overview	
5.2 Extra Trees Regressor	41
5.3 Linear Regression	43
5.4 Ridge Regression	44
5.5 Lasso Regression	45
5.6 Elastic Net	45
5.7 Random Forest Regression	46
5.8 Extreme Gradient Boosting (Xgboost)	47
5.9 Support Vector Regressor (SVR)	48
5.10 Results	49
CHAPTER 6: CONCLUSION AND FUTURE SCOPES	53-54
6.1 Conclusion	53
6.2 Future Scopes	54
ANNEXURE 1	
ANNEXURE 2	
ANNEXURE 3	

LIST OF FIGURES

Figure 2.1	The Idea	18
Figure 2.2	Basic circuit arrangement for "Li-Ion Data Logging Device" sensor hub	20
Figure 2.3	Detailed circuit arrangement for "Li-Ion Data Logging Device" sensor hub	21
Figure 2.4	Code for voltage computation	24
Figure 2.5	Code for current computation	25
Figure 2.6	Code for temperature computation	26
Figure 2.7	Sampling of data	26
Figure 2.8	Sampling of data	27
Figure 2.9	Establishing an Serial communication between raspberry pi and arduino	28
Figure 2.10	Unpacking and packing of data with additional information	29
Figure 2.11	Detection of a completed cycle	29
Figure 33.1	Database schema and creation	32
Figure 33.2	Database schema and creation	33
Figure 33.3	The frontend	34
Figure 5.1	Correlation Among Features	42
Figure 5.2	First five rows of dataset	42
Figure 5.3	Info of dataset	43
Figure 5.4	Description of dataset	43
Figure 5.5	Residuals Plot for ET	44
Figure 5.6	Residuals Plot for LR	44
Figure 5.7	Residuals Plot for RR	45
Figure 5.8	Code for Lasso Regression	45
Figure 5.9	Residuals Plot for Lasso Regression	46
Figure 5.10	Residuals Plot for EN	46
Figure 5.11	Residuals Plot for RF	47
Figure 5.12	Code for implementation of XGBoost Model	47
Figure 5.13	Residuals Plot for XGB	48
Figure 5.14	Code for implementation of SVR Model	48
Figure 5.15	Residuals Plot for SVR	49
Figure 5.16	Box plot for MSE MSE Comparison of Models	50
Figure 5.17	Final Predicted RUL	50
Figure 5.18	Discharging voltage vs time curve and current vs time graph	51

Figure 5.19 Charging voltage vs time

51

Figure 5.20 Charging Current vs time

52

LIST OF TABLES

Table 1.1	Comparison of Different Energy Storage Technologies	08
Table 2.1	Test battery specifications	22

CHAPTER ONE

INTRODUCTION TO BATTERY TECHNOLOGY

1.1 INTRODUCTION TO DC STORAGE

The history of DC storage involves the development of various technologies to store direct current (DC) for electrical power. Early batteries, like the voltaic pile invented by Alessandro Volta in 1800, marked the beginning. Over time, advancements led to lead-acid batteries, which were widely used in the 19th century. The 20th century saw innovations like nickel-cadmium and nickel-metal hydride batteries.[1] In recent years, lithium-ion batteries have become dominant due to their high energy density, powering a wide range of devices from small electronics to electric vehicles. The history of DC storage can be summarized as follows:

- I. **Voltaic Pile (1800):** Italian physicist Alessandro Volta invented the voltaic pile, an early form of battery that produces a continuous and stable electric current using alternating layers of zinc and copper separated by an electrolyte.
- II. **Lead-Acid Batteries (1859):** French physicist Gaston Planté invented the lead-acid battery, which became the first rechargeable battery. Lead-acid batteries were widely used in the 19th and early 20th centuries for various applications, including electric vehicles.
- III. **Nickel-Cadmium Batteries (1899):** Swedish scientist Waldemar Jungner patented the nickel-cadmium (NiCd) battery, another rechargeable battery technology. NiCd batteries were commonly used in early portable electronic devices.
- IV. **Nickel-Metal Hydride Batteries (1967):** Researchers at the United States Energy Research and Development Administration (ERDA) developed the nickel-metal hydride (NiMH) battery, an improvement over NiCd batteries with higher energy density and less environmental impact.
- V. **Lithium-Ion Batteries (1980s):** The development of lithium-ion (Li-ion) batteries marked a significant advancement in DC storage technology. Sony commercialized the first practical Li-ion battery in 1991, and these batteries have since become ubiquitous in portable electronics, electric vehicles, and renewable energy systems.
- VI. **Advancements in Lithium-Ion Technology (ongoing):** Continuous research and development efforts have led to improvements in lithium-ion battery technology, including enhanced energy density, longer cycle life, and increased safety. These batteries remain at the forefront of DC storage solutions in the 21st century.

1.2 DIFFERENT TYPE OF BATTERIES

- I. Alkaline Batteries:**
Composition: Zinc and manganese dioxide.
Common Use: Household devices like remote controls, flashlights.
- II. Lead-Acid Batteries:**
Composition: Lead dioxide cathode, sponge lead anode, sulfuric acid electrolyte.
Common Use: Automotive batteries, uninterruptible power supplies (UPS).
- III. Lithium-Ion Batteries:**
Composition: Lithium-cobalt oxide cathode, graphite anode.
Common Use: Portable electronics (laptops, smartphones), electric vehicles, renewable energy storage.
- IV. Nickel-Cadmium (NiCd) Batteries:**
Composition: Nickel oxide hydroxide cathode, cadmium anode.
Common Use: Power tools, emergency lighting.
- V. Nickel-Metal Hydride (NiMH) Batteries:**
Composition: Nickel oxide hydroxide cathode, metal hydride anode.
Common Use: Hybrid vehicles, rechargeable AA and AAA batteries.
- VI. Zinc-Carbon Batteries:**
Composition: Zinc chloride cathode, carbon rod anode.
Common Use: Low-drain devices like clocks, remote controls.
- VII. Silver Oxide Batteries:**
Composition: Silver oxide cathode, zinc anode.
Common Use: Watches, hearing aids, small medical devices.
- VIII. Zinc-Air Batteries:**
Composition: Interaction of oxygen with zinc and an electrolyte.
Common Use: Hearing aids, certain medical devices.
- IX. Lithium Polymer Batteries:**
Composition: Similar to lithium-ion but with a different electrolyte and packaging.
Common Use: Thin and flexible form factors, used in smartphones, tablets, wearables.
- X. Flow Batteries:**
Composition: Electrolyte solutions stored in external tanks.
Common Use: Large-scale energy storage for renewable energy systems.
- XI. Sodium-Ion Batteries:**
Composition: Sodium cathode, typically a carbon anode.
Common Use: Still in research; potential for grid energy storage.

These batteries vary in terms of energy density, lifespan, cost, and suitability for different applications.

1.3 BATTERIES TERMINOLOGIES

Understanding battery terminologies is crucial for evaluating and comparing different types of batteries[1]. Here are some key battery terminologies:

I. Voltage (V):

The electrical potential difference between two points in a circuit.

Measured in volts. ($V = I * R$)

In batteries, voltage indicates the electrical potential difference between the positive and negative terminals. In electrical circuits, it determines the rate of flow of electric current. Different devices and components in a circuit may require specific voltage levels for proper operation.

II. Capacity (Ah or mAh):

The amount of electric charge a battery can store.

Measured in ampere-hours (Ah) or milliampere-hours (mAh). ($Q = I * t$)

In general, the battery capacity is dependent on discharge rate. There are two ways of indicating battery discharge rate: C rate is the rate in amperes, while nC rate will discharge a battery in 1/n hours. For example, a rate of C/2 will discharge a battery in 2 hours, and a rate of 5C will discharge a battery in 0.2 hours. For a 2 Ah battery, the C/5 rate is 400 mA, while its 5C rate is 10 A.

III. Current (I):

The flow of electric charge in a circuit.

Measured in amperes (A). ($I = V/R$)

There are 2 types of electric current:

Direct current - In direct current, electric charges flow in one direction. Batteries and many electronic devices, such as cell phones and laptops.

Alternating current - In alternating current, electric charges periodically change direction. AC is a type of current commonly used in household and industrial power distribution systems.

IV. Power (W):

The rate at which energy is transferred or converted.

Calculated as the product of voltage and current ($W = V * I$). This equation indicates that power is the product of voltage and current in an electrical circuit. It also highlights that there are multiple ways to achieve a certain level of power in a circuit—either by having a high voltage with low current or vice versa.

V. Energy (Wh or kWh):

The total amount of work a battery can perform. Measured in watt-hours (Wh) or kilowatt-hours (kWh). ($E = P * t$). The principle of conservation of energy states that energy cannot be created or destroyed, only transformed from one form to another. This principle is fundamental to understanding the behavior of energy in physical systems.

VI. Cycle Life:

The number of charge and discharge cycles a battery can undergo before its capacity significantly degrades. Cycle life is typically specified by the number of cycles a battery can endure before its capacity drops to a certain percentage of its initial capacity (e.g. 80% of initial capacity). It is an important factor in determining the lifespan and durability of a rechargeable battery.

VII. Self-Discharge Rate:

The rate at which a battery loses its charge when not in use. Discharging rate, often referred to as discharge rate or current, is the rate at which a battery releases electrical energy. It is the flow of electric charge from the battery to the connected load.

VIII. C-Rate:

A measure of the charge or discharge current relative to the capacity of the battery. The charging rate is an important parameter when considering the time it takes to charge a battery or device and how quickly it can be brought to its full or desired charge level. The higher the charging rate, the faster the charging process. However, it's important to note that the charging rate is often subject to limitations and recommendations to ensure the safety and health of the battery.

For example, a 1C rate for a 1,000 mAh battery is 1,000 mA.

IX. Open Circuit Voltage (OCV):

The voltage of a battery when it is not connected to any load, it is the voltage measured across the terminals of the battery when no current is flowing. OCV represents the potential difference between the positive and negative terminals of the battery in ideal and no load condition.

X. Nominal Voltage:

The average voltage of a battery during its discharge cycle. The nominal voltage of a battery is a commonly specified average or rounded value that is used to label or identify the battery. It is not the exact voltage, but a convenient reference value.

XI. Specific Energy:

The energy stored per unit weight or volume of a battery. It is commonly expressed in units of energy per unit mass, such as joules per kilogram (J/kg) or watt-hours per kilogram (Wh/kg), or in units of energy per unit volume, such as joules per cubic meter (J/m³) or watt-hours per liter (Wh/L). Specific energy is a useful metric for comparing the energy storage capabilities of different materials or systems.

XII. State of Charge (SOC):

The current capacity of a battery as a percentage of its maximum capacity.

The SOC is a measure of the residual capacity of the battery. It is a crucial parameter in the management of rechargeable batteries and is often used in electrical vehicles, renewable energy systems and portable electronic devices.

XIII. Depth of Discharge (DoD):

The percentage of a battery's capacity that has been used during a discharge cycle.

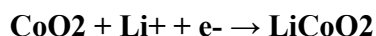
Monitoring the depth of discharge is crucial for battery management, as it helps prevent over-discharging, which can lead to reduced battery lifespan and performance. Many rechargeable batteries, especially lithium-ion batteries, have recommended depth of discharge limits to ensure longevity.

1.4 LITHIUM ION BATTERY

A lithium-ion (Li-ion) battery is a type of rechargeable battery that has become ubiquitous in various electronic devices due to its high energy density, relatively low self-discharge, and lightweight. Here are key features and aspects of lithium-ion batteries:

1.4.1 CHEMICAL ASPECTS

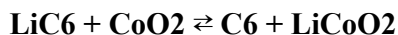
The cathode is usually composed of materials such as lithium cobalt oxide (LiCoO₂), lithium manganese oxide (LiMn₂O₄), lithium iron phosphate (LiFePO₄), or other lithium-based substances. On the other hand, the anode is typically constructed from graphite. Within a lithium-ion battery, oxidation-reduction (Redox) reactions occur, with the reduction process transpiring at the cathode. Specifically, cobalt oxide reacts with lithium ions at the cathode, resulting in the formation of lithium-cobalt oxide (LiCoO₂) [1].



The anode undergoes oxidation, where the graphite intercalation compound LiC₆ transforms into graphite (C₆) and releases lithium ions[7].



Here is the full reaction



Operation:

1. Discharge: During use, lithium ions move from the anode to the cathode through an electrolyte, creating an electric current.
2. Charge: When charging, the process is reversed, with lithium ions moving from the cathode back to the anode.

1.4.2 IDEAL AND PRACTICAL CHARACTERISTICS

Ideal Characteristics of a Lithium-Ion Battery:

- I. **High Energy Density:** Ideal batteries should store a large amount of energy relative to their size and weight.
- II. **Long Cycle Life:** The ability to undergo a large number of charge and discharge cycles without significant capacity degradation.
- III. **Fast Charging and Discharging:** Rapid charge and discharge capabilities without compromising battery health.
- IV. **Low Self-Discharge Rate:** Minimal loss of charge when the battery is not in use, allowing for longer shelf life.
- V. **Wide Operating Temperature Range:** Performance should remain efficient in a variety of temperature conditions, both high and low.
- VI. **High Voltage:** A higher voltage allows for more power output and is beneficial for many applications.
- VII. **Safe Operation:** Safety is paramount; ideal batteries should not pose a significant risk of overheating, catching fire, or exploding.[5]

Practical Characteristics of Lithium-Ion Batteries:

- I. **High Energy Density :** Lithium-ion batteries have achieved a high energy density compared to many other types of batteries, making them suitable for a wide range of applications.
- II. **Cycle Life :** Advances have been made in enhancing the cycle life of lithium-ion batteries, but they still experience some capacity degradation over time.
- III. **Fast Charging and Discharging :** Ongoing research aims to improve the speed of charging while maintaining battery health.
- IV. **Low Self-Discharge Rate:** Lithium-ion batteries generally have a low self-discharge rate, allowing for longer storage periods.

- V. **Wide Operating Temperature Range :** While lithium-ion batteries operate well in moderate temperatures, extreme conditions can still impact performance.
- VI. **High Voltage :** Lithium-ion batteries provide a high voltage output, contributing to their efficiency in powering various devices.

1.4.3 APPLICATIONS AND ADVANTAGES

Lithium-ion batteries have proven highly practical for a diverse array of applications, and ongoing research and development continue to address challenges and enhance their characteristics.

Advantages:

1. High Energy Density: Li-ion batteries offer a high energy density, providing a lot of energy in a compact and lightweight package.
2. Low Self-Discharge: Li-ion batteries have a relatively low self-discharge rate compared to other rechargeable batteries.
3. No Memory Effect: Unlike some other rechargeable batteries, Li-ion batteries do not suffer from memory effect, so they can be charged at any state of charge without affecting performance.

Applications:

1. Portable Electronics: Commonly used in smartphones, laptops, tablets, cameras, and other portable electronic devices.
2. Electric Vehicles (EVs): Power the propulsion systems in electric cars and hybrid vehicles.
3. Renewable Energy Storage: Used in conjunction with solar panels and wind turbines for energy storage in residential and commercial applications.

Challenges:

1. Temperature Sensitivity: Li-ion batteries can be sensitive to high temperatures, and exposure to extreme heat can degrade their performance and lifespan.
2. Safety Concerns: While rare, thermal runaway (a rapid, uncontrolled increase in temperature) can occur, leading to safety concerns. Battery management systems are employed to mitigate these risks.[7]

Advancements:

Ongoing research aims to improve the energy density, safety, and cost-effectiveness of Li-ion batteries. Advances in materials and design continue to drive progress in battery technology.

Lithium-ion batteries have revolutionized portable electronics and are playing a crucial role in the transition to electric vehicles and renewable energy solutions. Their widespread use is a result of their excellent balance between energy density, power output, and rechargeability.

Comparison of Different Energy Storage Technologies

Table 1.1 Comparison of Different Energy Storage Technologies[11]

Storage Technology	Cycle Life	Efficiency (%)	Specific power (W/Kg)	Specific energy (Wh/kg)
Lead acid battery	500-800	50-92	150-400	30-40
Li-ion battery	400-1200	80-90	300-1500	150-250
Nickel metal hydride battery	500-1000	66	250-1000	30-80
Ultracapacitor	1000000	90	1000-9000	0.5-30

1.5 STATE OF CHARGE

Accurate estimation of the State of Charge (SoC) holds paramount importance in efficiently managing and utilizing battery systems. SoC refers to determining the remaining capacity of a battery as a percentage of its full capacity, playing a crucial role in maximizing battery lifespan, optimizing performance, and ensuring the safe and effective operation of various applications[2].

The state of charge (SoC) of cells is defined as the available capacity in ampere-hours (Ah) expressed as a percentage of its rated capacity. This SoC parameter serves as a thermodynamic measure, offering insights into the potential energy stored in a battery. Additionally, evaluating the state of health (SOH) of a battery becomes significant, representing the battery's capacity to store and deliver electrical energy in comparison to a new battery.

While SoC stands as a key parameter for batteries, its definition presents various challenges. In essence, the SoC of a battery is generally defined as the ratio of its current capacity $Q(t)$ to the nominal capacity $Q(n)$, where the nominal capacity, provided by the manufacturer, denotes the maximum charge the battery can store.

1.5.1 METHODS FOR SOC ESTIMATION

Accurate estimation of the State of Charge (SoC) in batteries is crucial for optimizing performance, extending battery life, and ensuring safe and efficient operation. Various methods are employed for SoC estimation, each with its strengths and limitations.

I . Voltage-Based Methods:

Voltage-based approaches utilize the connection between a battery's open-circuit voltage (OCV) and its State of Charge (SoC). The Open Circuit Voltage Method establishes a correlation between the battery's voltage and SoC when it is neither charging or discharging actively. Alternatively, a State of Charge Voltage Curve is generated by mapping the battery's voltage at various SoC levels during charging and discharging processes. Despite their simplicity, these methods may exhibit reduced accuracy in dynamic operational scenarios[14].

II . Current Integration (Coulomb Counting):

Current integration, commonly referred to as Coulomb counting, involves the computation of the total charge transferred into or out of a battery over a specific period. While the concept behind this method is straightforward, its implementation necessitates accurate current measurements and is susceptible to cumulative errors, particularly in the presence of factors such as self-discharge[19].

The Coulomb counting approach assesses the discharging current of a battery and integrates this current over time to derive an estimate of the State of Charge (SOC). The SOC at a given time $SOC(t)$ is determined based on the discharging current $I(t)$ and previously estimated SOC values $SOC(t-1)$. The calculation of SOC is expressed by the following equation:

$$SOC(t) = SOC(t-1) + I(t)n\Delta t$$

III. Impedance-Based Methods:

Impedance-based techniques focus on assessing the internal resistance or impedance of a battery, a parameter that varies with State of Charge (SoC). While this method often demands additional hardware and sophisticated algorithms, it can yield accurate estimates, particularly when complemented by other approaches. The methodology involves utilizing battery voltage and current to measure the internal resistance, with voltage variations observed during small durations (less than 10 ms). The ratio of voltage and current changes produces DC resistance, indicating the battery's capacity in DC. The brief interval is crucial to capture the ohmic effect and mitigate the impact of transfer reactions and acid diffusion. However, prolonged durations lead to errors in estimated resistance. Notably, this method exhibits good adaptability and high accuracy in SoC estimation, particularly during the latter stages of discharging. Despite its precision challenges due to low values (in the milliohm range) and limited usability for

estimating SoC, the internal resistance's slight variations over a broad SoC range pose challenges for observation. Consequently, DC internal resistance is seldom employed for SoC estimation.

IV. Kalman Filtering:

Kalman filtering is a mathematical algorithm that integrates data from various sources, including voltage, current, and temperature, to gauge the State of Charge (SoC). Widely employed in control systems, this method excels in delivering precise and resilient SoC estimations, especially in dynamic operating conditions.

Utilizing real-time road data measurements to assess the State of Charge (SOC) of a battery is typically challenging or costly. The application of the Kalman filter method, however, offers a reliable means of estimating battery SOC through real-time state estimation. Yatsui and Bai introduced a SOC estimation method for lithium-ion batteries based on the Kalman filter, and experimental outcomes affirm the efficacy of this approach during online application. Barbarisi et al. proposed an extended Kalman filter (EKF) to gauge the concentrations of key chemical species averaged over the thickness of the active material. This allows for SOC determination based on terminal current and voltage measurements. Introducing a novel SOC estimation method based on unscented Kalman filter (UKF) theory and a comprehensive battery model, another study demonstrated that the UKF method surpasses the extended Kalman filter in accurately estimating SOC for batteries. Sun et al. presented an adaptive UKF method designed to estimate SOC for lithium-ion batteries in electric vehicles[16].

V. Machine Learning:

Machine learning techniques, such as neural networks, support vector machines, and regression models, are gaining popularity for SoC estimation. These methods learn from historical data and can adapt to different battery chemistries and operating conditions. Machine learning approaches are particularly useful when dealing with complex, nonlinear battery behaviors.

A Neural Network employs a mathematical algorithmic model to capture intricate characteristics and parallel processing capabilities of complex neural networks. It excels in processing data and discerning relationships among various initially intricate factors. Among the various Neural Network algorithms, the Backpropagation (BP) Neural Network stands out for its ability to solve nonlinear systems while maintaining a simpler topology compared to conventional Neural Network methods. The BP Neural Network's structure comprises three layers: the input layer, the hidden layer, and the output layer. The input layer encompasses battery voltage, current, resistance, and ambient temperature. The number of hidden layers is determined by the system's precision requirements. The output layer of this system generates an estimated State of Charge (SOC) value.

The support vector machine (SVM) has found application in classifying different patterns across various domains in pattern recognition. Additionally, it has been utilized for regression tasks, which are inherently more challenging than classification problems. When employed as a nonlinear estimation system, SVM exhibits greater robustness compared to a least squares estimation system, as it remains relatively insensitive to minor alterations[13].

Ultimately, the choice of SoC estimation method depends on the specific application, battery chemistry, and the desired balance between accuracy, complexity, and resource requirements.

1.5.2 MORE ABOUT COULOMB COUNTING METHOD

Coulomb counting is an approach utilized to gauge the State of Charge (SoC) in a battery by assessing the overall electric charge transferred to or from the battery across a period. This technique relies on integrating the current moving into or out of the battery with respect to time. The fundamental concept involves tracking the accumulated charge entering or leaving the battery, commencing from a known initial state of charge.

In its most straightforward manifestation, the Coulomb counting method entails monitoring the current entering or exiting the battery and integrating this current over time. The SoC is then approximated by dividing the integrated charge by the nominal capacity of the battery. In mathematical terms, it can be articulated as [20]:

$$\text{SOC}(t) = \text{SOC}(t-1) + I(t)n\Delta t$$

Where:

(SoC(t)) is the State of Charge at time (t),

(Initial SoC(t-1)) is the SoC at the starting time (t-1),

(I(t)) is the current at time (t),

(n) is the efficiency.

Modified Coulomb Counting Method

To enhance the Coulomb counting method, a novel approach known as the modified Coulomb counting method is suggested. This modified technique incorporates a refined current, referred to as the corrected current, to enhance the precision of the estimation. The corrected current is determined based on the discharging current, and there exists a quadratic correlation between the corrected current and the battery's discharging current. Through the application of experimental data, the corrected current is computed using the following formula[21]:

$$I_c(t) = k_2 I(t)^2 + k_1 I(t) + k_0$$

In the modified Coulomb counting method, the State of Charge (SoC) is determined using the following equation, where the constants k_2 , k_1 , and k_0 are derived from practical experimental data:

$$\text{SOC}(t) = \text{SOC}(t - 1) + I(t)Q\Delta t$$

While Coulomb counting is a straightforward and conceptually simple method, it does have limitations. One significant challenge is that it requires accurate measurements of current, and any errors in the measurement can accumulate over time, leading to inaccuracies in the SoC estimation. Additionally, factors such as temperature changes, aging effects, and variations in the battery's internal resistance can introduce uncertainties.

Despite its limitations, Coulomb counting is widely used in practical battery management systems due to its simplicity and real-time applicability. It is often used in conjunction with other SoC estimation methods or correction techniques to improve accuracy, especially in applications where precise SoC information is critical for optimal battery performance and longevity.

1.5.3 ADVANTAGES OF COULOMB COUNTING

I . REAL TIME APPLICABILITY

Coulomb counting provides a real-time estimation of the State of Charge, making it suitable for applications where immediate and continuous monitoring of battery status is crucial.

II . SIMPLICITY

The concept of Coulomb counting is straightforward and easy to implement. It involves integrating the current flowing in or out of the battery over time, and the SoC is calculated based on this accumulated charge.

III. LOW COMPUTATIONAL OVERHEAD

Coulomb counting involves simple arithmetic operations, which results in low computational overhead. This makes it computationally efficient and suitable for resource-constrained systems.

IV. COMPATIBILITY WITH VARIOUS BATTERIES CHEMISTRY

Coulomb counting is applicable to a wide range of battery chemistries. It doesn't rely on specific electrochemical reactions and can be used for different types of batteries, such as lithium-ion, lead-acid, nickel-metal hydride, etc.

V. INTEGRATION INTO BATTERY MANAGEMENT SYSTEM

Coulomb counting is often integrated into sophisticated Battery Management Systems (BMS) that monitor and control the charging and discharging of batteries. BMS systems use Coulomb counting as one component of a comprehensive strategy for managing battery health and performance.

1.5.4 ESTIMATION OF SOC IN PYTHON

Estimating State of Charge (SoC) in Python involves implementing algorithms or methods that use data from the battery, such as voltage and current measurements, to calculate the remaining capacity as a percentage of the total capacity[14]. Here's a simple example using Coulomb counting:

```
class Battery:
    def __init__(self, capacity):
        self.capacity = capacity # Total battery capacity in ampere-hours
        self.soc = 100.0 # Initial State of Charge in percentage

    def update_soc_coulomb_counting(self, current, time):
        # Coulomb counting method
        # Update SoC based on current and time passed
        delta_soc = (current * time) / self.capacity
        self.soc -= delta_soc
        # Ensure SoC stays within 0% and 100%
        self.soc = max(0.0, min(100.0, self.soc))

# Example usage:
battery = Battery(capacity=100.0) # Initializing a battery with a capacity of 100 Ah

# Simulate discharging the battery with a current of 5A for 2 hours
current_discharge = 5.0 # in Amperes
time_discharge = 2.0 # in hours
battery.update_soc_coulomb_counting(current_discharge, time_discharge)

print(f"Current State of Charge: {battery.soc}%")
```

This is a basic example and does not consider factors like temperature effects, aging, or complex battery behaviors. For more accurate SoC estimation advanced algorithms, machine learning models, or combine multiple estimation methods are used.

The accuracy of SoC estimation heavily relies on the precision of measurements and the characteristics of the battery being monitored. Advanced battery management systems often employ more sophisticated algorithms and sensor data to achieve precise SoC estimation.

1.6 OVERVIEW OF EVs

Electric vehicles (EVs) have emerged as a groundbreaking solution to the environmental challenges posed by traditional internal combustion engine vehicles. Unlike their fossil fuel counterparts, EVs are powered by electricity stored in advanced battery packs. These batteries, often utilizing cutting-edge lithium-ion technology, have seen significant improvements in energy density, resulting in longer driving ranges and enhanced overall performance. The electric motor, a core component of EVs, offers immediate torque, providing a unique driving experience characterized by smooth acceleration. This departure from conventional vehicles aligns with global efforts to reduce carbon emissions and combat climate change[3].

A pivotal aspect of the electric vehicle ecosystem is the evolving charging infrastructure. Ranging from standard home outlets to fast-charging stations, these facilities play a crucial role in the widespread adoption of EVs. Innovations in charging technology aim to address concerns about "range anxiety," offering more convenient and accessible solutions for users. Moreover, governments worldwide are incentivizing the transition to electric transportation through subsidies and tax credits, fostering a supportive environment for both consumers and manufacturers.

As the electric vehicle industry advances, it not only contributes to environmental sustainability but also challenges preconceived notions about electric cars. Modern EVs showcase impressive ranges, high-performance capabilities, and innovative features, reshaping perceptions of what electric mobility entails. Continuous research and development investments from automakers further drive the sector's growth, promising a future where electric vehicles play a central role in fostering a cleaner, greener, and more technologically advanced transportation landscape.

1.7 ENERGY STORAGE CHALLENGE FOR EVs

The vitality of electric vehicles (EVs) lies in their battery technology, predominantly utilizing lithium-ion configurations for their high energy density. Central to the evolution of EVs is the ongoing quest to extend battery range, alleviating concerns of "range anxiety" and making these vehicles more practical for everyday use. Simultaneously, a key focus is on enhancing battery durability to ensure a sustainable and cost-effective ownership experience. Scientists are exploring methods to reduce degradation over time and establish efficient recycling processes.

Charging speed is another crucial facet, prompting advancements in fast-charging capabilities for greater convenience. Government incentives are driving significant investments in research,

fostering collaborations between automakers and battery manufacturers to accelerate progress. The pursuit of sustainability extends to materials used in battery production, aiming to minimize environmental impact[3].

In essence, the future of EVs is intricately linked to continuous improvements in battery technology. From extending driving ranges to enhancing durability and charging efficiency, these advancements are poised to redefine the landscape of transportation, making electric vehicles a more compelling and environmentally friendly choice for consumers.

1.8 LITHIUM-ION BATTERIES FOR ELECTRIC VEHICLES

Lithium-ion batteries (Li-ion) have become the backbone of electric vehicle (EV) technology, gaining popularity for their high energy density, lightweight design, and rechargeable nature. The widespread adoption of Li-ion batteries in EVs gained momentum in the early 21st century, aligning with the surge in interest and investment in electric mobility. Their popularity stems from their ability to store and release electrical energy efficiently, providing the necessary power to propel vehicles while maintaining a relatively compact and lightweight form factor.

The use of Li-ion batteries in EVs is driven by their superior energy-to-weight ratio, enabling longer driving ranges and improving overall performance. The lightweight nature of these batteries also contributes to the efficiency and agility of electric vehicles, counteracting the added weight traditionally associated with battery-powered propulsion.

However, Li-ion batteries come with their set of challenges. One prominent issue is their limited lifespan, as they tend to degrade over time with repeated charge and discharge cycles. Efforts are underway to enhance battery durability through innovations in electrode materials and manufacturing processes. Additionally, concerns exist regarding the environmental impact of extracting and processing lithium, cobalt, and other materials used in these batteries. Recycling initiatives and the exploration of alternative materials aim to address these sustainability challenges[9].

Furthermore, safety considerations surround Li-ion batteries, with rare instances of thermal runaway leading to overheating and fires. Ongoing research focuses on improving battery management systems and incorporating advanced safety features to mitigate these risks.

In conclusion, while lithium-ion batteries have revolutionized the electric vehicle industry, ongoing efforts in research and development are crucial to addressing challenges related to lifespan, sustainability, and safety, ensuring a more robust and sustainable future for electric mobility.

1.9 USE OF MACHINE LEARNING METHODS FOR OPTIMIZING LI-ION BATTERIES

Remaining Useful Life (RUL) Prediction

- Machine learning (ML) methods are employed to predict the Remaining Useful Life of Li-ion batteries. This involves estimating the time until a battery reaches the end of its operational life, aiding in proactive maintenance and replacement strategies.
- Regression algorithms, such as Random Forests or Long Short-Term Memory (LSTM) networks, are commonly used for RUL prediction. These models analyze historical battery performance data to forecast when degradation might necessitate replacement.
- RUL models require labeled datasets with information on battery health over time. Continuous monitoring of battery parameters, including voltage, current, and temperature, contributes to building robust datasets for training and validation.

Lifecycle Optimization

- ML is employed for optimizing the entire lifecycle of Li-ion batteries, considering factors such as charging strategies, usage patterns, and environmental conditions to extend overall performance and reduce degradation.
- Reinforcement learning algorithms, like Q-learning, can be applied to determine optimal charging and discharging policies. These models learn from interactions with the battery system, adapting strategies to maximize efficiency and lifespan.
- Lifecycle optimization requires diverse datasets encompassing various operating scenarios. Simulated and real-world data, incorporating different charging rates, temperature variations, and usage patterns, helps train models to generalize well[23].

Data Generation for ML in Battery Optimization

- Data for ML models can be sourced from on-board sensors in EVs or stationary battery systems. This includes voltage and current measurements, temperature data, and information about charging and discharging cycles.
- In scenarios where real-world data is limited, synthetic data generation methods, such as physics-based models or data augmentation techniques, can be employed. This ensures ML models have a diverse set of examples to learn from. ML models for battery optimization often include anomaly detection mechanisms. Training datasets must encompass instances of battery malfunctions or unusual behavior to enhance the model's ability to detect and address potential issues.

CHAPTER TWO

REMAINING USEFUL LIFE ESTIMATION AND LIFECYCLE OPTIMIZATION

2.1 INTRODUCTION

As we have got a proper context of the background, now is the right time to get versed with the problem we are trying to solve and the solution proposed. In this chapter we will be taking a peek at the methodology which will be expanded upon in the coming chapters.

2.2 PROBLEM STATEMENT

Li-ion batteries are currently one of the most popular technologies for energy storage due to its high energy density and performance. The life and efficiency of these batteries vary on the basis of various factors such as charging and discharging current, operating temperature etc over the course of its life. To optimize the battery for higher performance and life cycles and to accurately predict the remaining useful life, use deep learning methods using a training and testing dataset generated using Internet of Things.

2.3 PROPOSED SOLUTION

In figure 3.1 the whole idea of our solution is presented in the form of a block diagram. Now we will discuss the components in brief.[6]

The whole solution can be divided into two components i.e. dataset generation and machine learning which are linked by a communication link. The flow is as follows:-

1. Using a device called **sensor hub** we will be collecting all the data from different sensors which will provide operational parameters.
2. The sensor hub which is a microcontroller will also do some preprocessing at a basic level.
3. Serial communication will be established with an onboard computer where the data will be nourished and processed to make a reliable dataset.
4. The data will be displayed on a frontend which will also enable to control and monitor the whole system.
5. The dataset will be sent to a machine learning algorithm which will generate insights about the battery like remaining useful life and recommendations for better use.

These points will be discussed with every intricacy in the coming chapters.

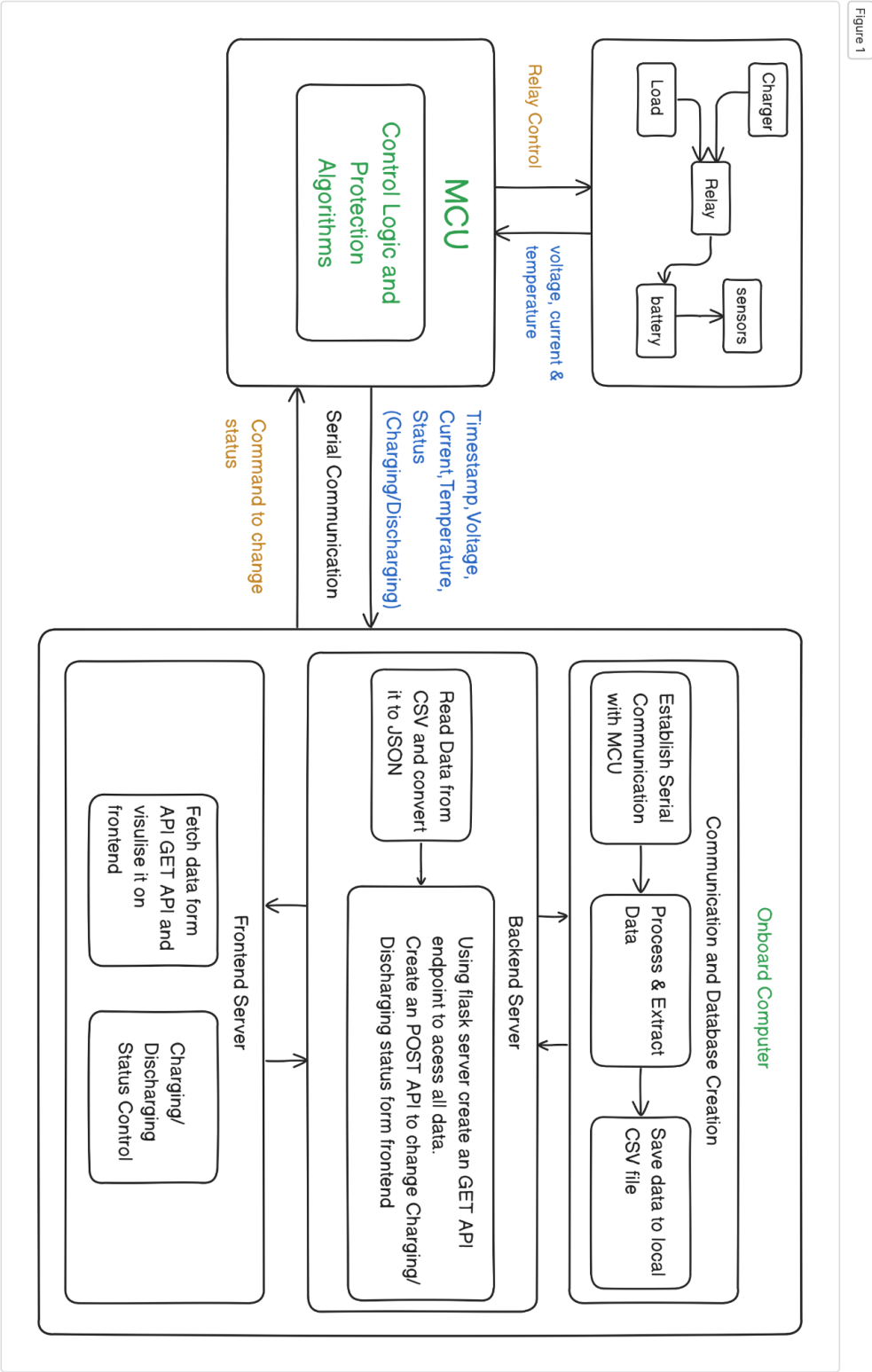


Figure 2.1 The Idea

2.4 OPERATIONAL PARAMETER DATASET GENERATION

In the world of batteries, predicting how long they'll last (Remaining Useful Life or RUL) and understanding their health (State of Health or SOH) is a big deal. This project uses a Raspberry Pi and Arduino together to create a detailed dataset for this purpose.

The Raspberry Pi is a small but powerful computer, and the Arduino is great for collecting real-time data. Together, they make an excellent choice for gathering information from lithium-ion batteries. This dataset includes important parameters like voltage, current, temperature and SoC basically, all the factors that help figure out how well a battery is doing.

A data logger device made using above technologies makes it possible for generating quality data which is precise and in the real time. The dataset created will be used as a library for machine learning algorithms to get important insights about the battery.[23]

2.5 METHODOLOGY

The dataset generation methodology for this project involved a systematic process to capture a comprehensive understanding of lithium-ion battery behavior. The hardware setup utilized Raspberry Pi and Arduino for efficient data acquisition. Arduino, functioning as a real-time sensor interface, gathered essential parameters like voltage, current, and temperature from the batteries. This data was then transmitted to the Raspberry Pi for processing and storage.

To ensure dataset diversity, batteries underwent various operational scenarios, encompassing different charge-discharge cycles and temperature conditions. This approach aimed to simulate real-world usage and provide a holistic view of battery performance. Stress testing was also incorporated, mimicking extreme conditions to assess the batteries' resilience.

Data preprocessing played a crucial role in refining the dataset. Raw data underwent cleaning, normalization, and feature extraction to eliminate noise and inconsistencies. This meticulous preparation ensured the dataset's quality and suitability for subsequent analysis.

Additionally, a logging mechanism was implemented to timestamp each data point, enabling chronological tracking of battery behavior. This temporal aspect facilitated the observation of trends and changes over time, contributing valuable insights into battery performance.

2.6 THE HARDWARE - LI-ION DATA LOGGING DEVICE

The hardware setup for dataset generation combined Raspberry Pi and Arduino technologies. Arduino, serving as a real-time sensor interface, collected critical parameters such as voltage and temperature from lithium-ion batteries. This information was transmitted to the Raspberry Pi for

processing and storage. The synergy between these platforms not only enabled efficient data acquisition but also established a seamless connection between hardware and software, forming the foundation for a comprehensive dataset reflective of diverse battery operational conditions.

2.6.1 CIRCUIT ARRANGEMENT

The circuit arrangement comprised crucial components for precise data collection from lithium-ion batteries. The ACS712 sensor measured current, a potential divider gauged voltage, and an LM35 captured temperature. A relay switch, strategically positioned, alternated between charging and discharging states. This configuration facilitated real-time data acquisition, providing a detailed dataset reflective of diverse battery operational conditions.

The device hardware consists of a test battery, ammeter, voltmeter, load, charging source and a relay acting as a switch to transition from charging to discharging. This will help us to gather information such as current, voltage and temperature. This part we are calling the **sensor hub**. A basic circuit arrangement of it is given in figure 2.1.

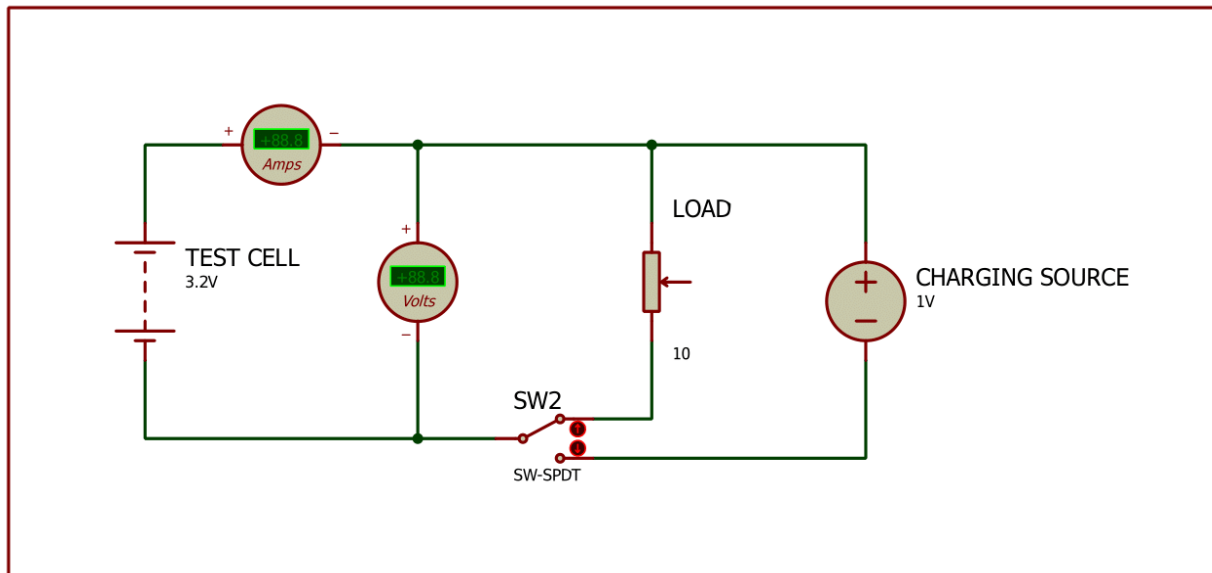


Figure 2.2 Basic circuit arrangement for "Li-Ion Data Logging Device" sensor hub

After setting up the basic circuit we need a microcontroller to gather data with their timestamps. Also the microcontroller will work as a charging and discharging controller for the test battery. A more detailed circuit arrangement with all real components can be found in figure 2.2[29].

The test cell for generation of databases for presenting the proof of concept was bought from the company orange.

Although this experiment used this particular cell, it can be tested with any battery pack with following specifications.

Max Current - 30A

One thing to note here is although all the data we need is gathered at one place using this hardware, it still needs to be stored in some database for processing and analyzing. For this we will be using an onboard powerful computer **raspberry pi**. More about it later in this chapter[31].

Cell Specifications

Table 2.1 Test battery specifications

Model No.	IFR32650 6000mAh
Capacity (mAh)	6000
Output Voltage (VDC)	3.2
Discharge Voltage	2V
Charging Voltage	3.65V
Standard Charge	Constant Current: 0.2 C Constant Voltage: 3.6 5V Cut-off: 0.02 C
Standard Discharge	Constant Current: 0.2 C End Voltage: 2.0V
Fast Charge	Constant Current: 1C Constant Voltage: 3.65V Cut off: 0.02C
Fast Discharge	Constant Current: 1C End Voltage: 2.0V
Max. Continuous Charge Current	1C
Max. Continuous Discharge Current	3C
Operation Temp. Range	Charge: 0 ~ 45°C Discharge: -20 ~ 60°C

2.6.2 COMPONENTS

I . Arduino Uno

The Arduino Uno is a popular microcontroller board, serving as a foundation for countless DIY electronics projects. Developed by Arduino.cc, it features the ATmega328P microcontroller, offering digital and analog inputs/outputs, making it versatile for a range of applications. With a user-friendly interface and an extensive community, the Uno is an ideal choice for beginners and experts alike. Its open-source nature encourages innovation, allowing users to create interactive

devices, robots, and more. The Uno's simplicity, affordability, and broad compatibility with sensors and actuators make it a cornerstone in the world of prototyping and experimentation.

II. ACS712 Current Sensor

This Hall-effect-based device precisely measured the electrical current flowing through the circuit, providing crucial insights into the battery's operational characteristics. Its sensitivity and reliability made it an essential component, enabling the system to capture real-time current data with precision, contributing to the creation of a comprehensive dataset for subsequent analysis and understanding of the battery's performance.

The ACS712 sensor measures current using the Hall-effect principle. Inside the sensor, there's a thin strip of conductive material through which the current flows. Perpendicular to this current path, a magnetic field is generated when current passes through the strip. The Hall sensor, positioned nearby, detects the magnetic field's strength, which is directly proportional to the current flowing through the strip. This principle allows the ACS712 to convert the current into a voltage output, providing an accurate representation of the current magnitude. The sensor's ability to precisely convert electrical current into measurable voltage makes it a valuable tool for current sensing applications in various electronic systems.

<https://www.sparkfun.com/datasheets/BreakoutBoards/0712.pdf>

III. 0-25 V DC Voltage Sensor

The 0-25 V DC voltage sensor in this setup employs a potential divider mechanism to measure voltage accurately. This sensor comprises two resistors: a fixed resistor and a variable resistor (potentiometer). The fixed resistor establishes a known voltage ratio, while the potentiometer allows adjustment for precise calibration. The input voltage is divided proportionally across these resistors, creating a fraction of the total voltage that corresponds to the measured input. This scaled-down voltage is then fed to the system for monitoring. The adjustable nature of the potential divider ensures flexibility and accuracy in voltage measurement, contributing to the overall reliability of the data acquisition system.

<https://www.datasheethub.com/wp-content/uploads/2022/10/arduino-25v-voltage-sensor-module.pdf>

IV. LM35 Temperature Sensor

The LM35 temperature sensor is a versatile and widely utilized device in electronics. Renowned for its precision, it delivers accurate temperature readings in Celsius with a linear voltage output. Its straightforward interface, low power consumption, and compact design contribute to its popularity in various applications. From climate control systems to industrial processes, the LM35's reliability and ease of integration make it an indispensable tool for monitoring and

regulating temperatures in electronic circuits, offering a cost-effective and efficient solution for temperature-sensitive projects[12].

<https://www.ti.com/lit/ds/symlink/lm35.pdf>

2.7 THE FIRMWARE - LI-ION DATA LOGGING DEVICE

The embedded firmware was written in C/C++ programming language. Utmost care was given in making the system failsafe with rigorous testing of 100+ hours to test out any possible scenario.


Responsibilities of the firmware

- Over Charge protection of the battery.
- Automated Cutoff of charging if battery is critically low.
- Over Current protection of the hardware.
- Receiving Commands from frontend application to change its charging/ discharging state.
- Extraction and calculation of sensor data for onboard analog to digital converter.
- Encapsulation of all data in a format which can be forwarded for further processing.
- Establishing a serial communication with raspberry pi.

The whole firmware is provided in annexure 1. Here we can understand its working with the help of a few code snippets which will pave a way to the clarity of data sensing, processing and shipping.

I . Computation of Voltage

The code for voltage calculation is given in figure 2.4.



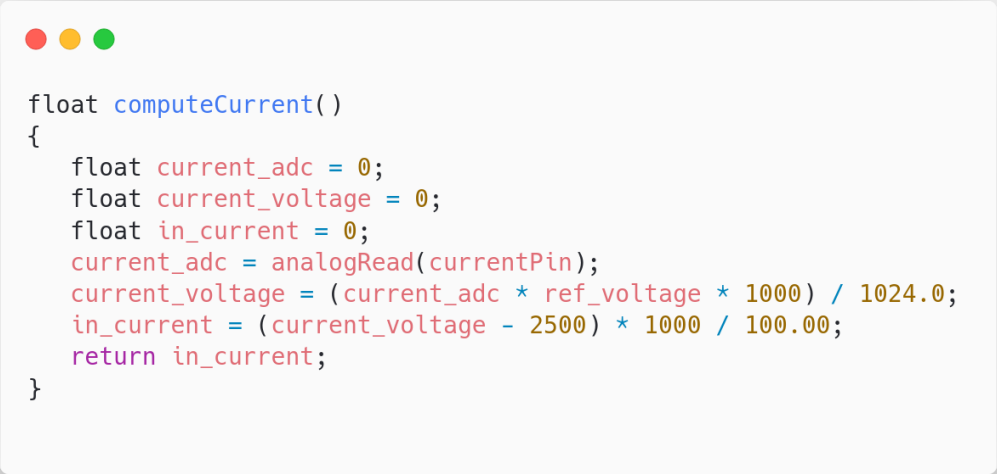
```
float computeVoltage()
{
    float R1 = 30000.0;
    float R2 = 7500.0;
    float adc_voltage = 0.0;
    float in_voltage = 0.0;
    int adc_value = 0;
    adc_value = analogRead(voltagePin);
    adc_voltage = (adc_value * 5) / 1024.0;
    in_voltage = adc_voltage / (R2 / (R1 + R2));
    return in_voltage;
}
```

Figure 2.4 Code for voltage computation

This code defines a function **computeVoltage()** to measure the voltage using an Arduino analog sensor connected to **voltagePin**. It assumes a voltage divider setup with resistors R1 (30,000 ohms) and R2 (7,500 ohms). The function reads the analog value from the sensor, converts it to voltage (assuming a 5V reference), and then calculates the actual input voltage considering the voltage divider formula. The final result, **in_voltage**, represents the voltage applied to the sensor.

II. Computation of Current

The code for current computation is given in figure 2.5. This code defines a function **computeCurrent()** for measuring current using an Arduino analog sensor connected to **currentPin**.



```
float computeCurrent()
{
    float current_adc = 0;
    float current_voltage = 0;
    float in_current = 0;
    current_adc = analogRead(currentPin);
    current_voltage = (current_adc * ref_voltage * 1000) / 1024.0;
    in_current = (current_voltage - 2500) * 1000 / 100.00;
    return in_current;
}
```

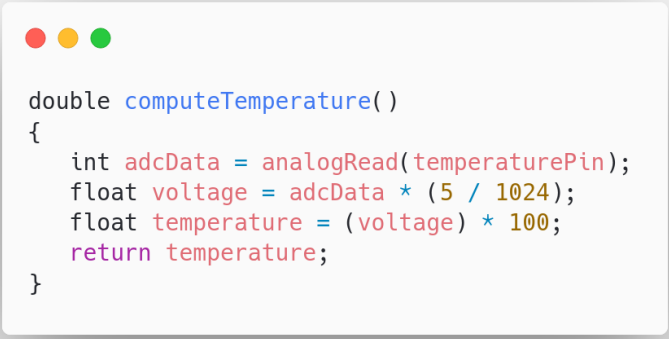
Figure 2.5 Code for current computation

It reads the analog value, converts it to voltage based on a reference voltage **ref_voltage**, and then calculates the input current. The calculation involves adjusting for a 2.5V offset and scaling by a factor of 100. This function is suitable for accurately obtaining current readings from a sensor, providing a useful tool for various electronic applications that involve current monitoring in milliamperes. The code assumes a 5V reference voltage and includes the necessary conversions for accurate current measurement.

III. Computation of Temperature

The code for temperature calculation is given in figure 2.6. [13] This code defines a function **computeTemperature()** for measuring temperature using an Arduino analog sensor connected to **temperaturePin**. It reads the analog value, converts it to voltage based on a 5V reference, and

then calculates the temperature assuming a linear relationship. The conversion from analog to voltage is done by multiplying the ADC reading by the ratio of 5V to 1024 (the ADC resolution).

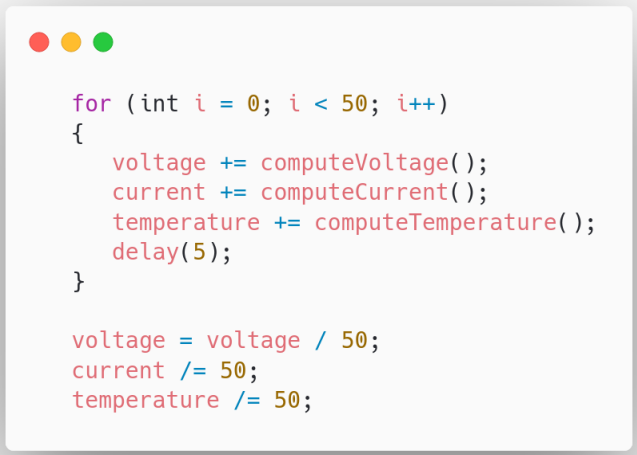


```
double computeTemperature()
{
    int adcData = analogRead(temperaturePin);
    float voltage = adcData * (5 / 1024);
    float temperature = (voltage) * 100;
    return temperature;
}
```

Figure 2.6 Code for temperature computation

The temperature calculation scales the voltage by a factor of 100. However, it's worth noting that without specific calibration information or a sensor model, the temperature result may not be accurate in real-world scenarios.

IV. Sampling of Data



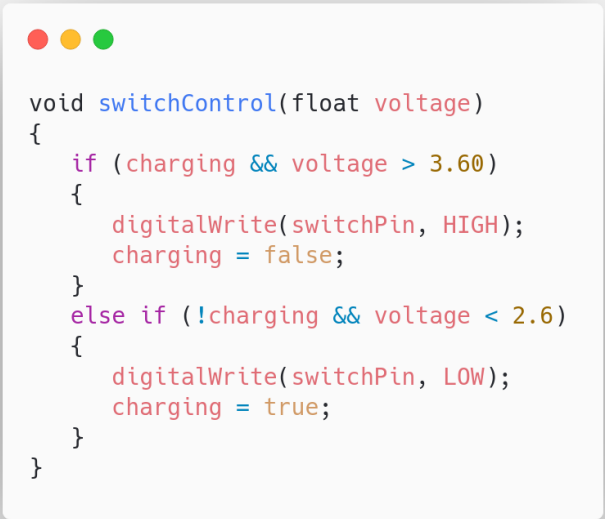
```
for (int i = 0; i < 50; i++)
{
    voltage += computeVoltage();
    current += computeCurrent();
    temperature += computeTemperature();
    delay(5);
}

voltage = voltage / 50;
current /= 50;
temperature /= 50;
```

Figure 2.7 Sampling of data

For better accuracy of data the average value of 50 samples taken with delay of 5ms each is taken. Figure 2.7 shows its implementation.

V. Charging Logic



```
void switchControl(float voltage)
{
    if (charging && voltage > 3.60)
    {
        digitalWrite(switchPin, HIGH);
        charging = false;
    }
    else if (!charging && voltage < 2.6)
    {
        digitalWrite(switchPin, LOW);
        charging = true;
    }
}
```

Figure 2.8 Sampling of data

The function is designed to control a switch based on the voltage level, presumably for charging a battery. It assumes the existence of a boolean variable named `charging` and uses a specific pin **switchPin** to control the switch.

Charging Condition - if (charging && voltage > 3.60)

If the system is currently in a charging state **charging** == **true** and the voltage exceeds 3.60 volts, the code inside this block is executed.

It sets the **switchPin** to HIGH, indicating the activation of the switch.

State Update: It updates the **charging** status to false, indicating that the system is no longer in a charging state.

Discharging Condition - else if (!charging && voltage < 2.6)

If the system is not currently charging == **false** and the voltage drops below 2.6 volts, the code inside this block is executed. It sets the **switchPin** to LOW, indicating the deactivation of the switch.

State Update - It updates the **charging** status to true, indicating that the system is now in a charging state.

The logic manages the charging process by dynamically controlling a switch based on voltage thresholds. When the battery voltage is above 3.60 volts and it's currently in a charging state, the charging is considered complete, and the switch is turned off. Conversely, when the voltage

drops below 2.6 volts and it's not currently in a charging state, the logic assumes it's time to start charging, and the switch is turned on. This simple logic helps protect the battery by controlling the charging process based on voltage conditions.

2.8 DATA COLLECTION AND PROCESSING IN RASPBERRY PI

After data is gathered in arduino uno it is transmitted to a python program which does some processing and computes some additional necessary data such as the time at which an data list is received (generally data list comes every 5-10 seconds) and the state of charge of the battery pack.

The program can be divided into several steps which are explained in detail.

Step 1 - Establishing a Serial Communication with Arduino



Figure 2.9 Establishing an Serial communication between raspberry pi and arduino

This Python script establishes a serial connection with a device using the `serial` module. Encapsulated within the main block, it initializes a serial object (`ser`) with parameters specifying the serial port (`/dev/ttyACM0`) and a baud rate of 9600. This connection facilitates communication between the script and the connected device. The script can be further extended to send and receive data over this serial link, making it a foundational setup for projects involving serial communication with external hardware through the specified port and baud rate.

Step 2 - Unpacking data and adding additional information.

In figure 2.10 the data is decoded and stripped. Then the current timestamp is retrieved. Apart from this the most important data of our analysis i.e state of charge is calculated using coulomb counting algorithm. Finally all the data is packed together and sent to the database and saved there which can be retrieved at any point of time according to our needs.

```

import time

while True:
    try:
        line = ser.readline().decode().strip()
        print(line)
        voltage, current, temperature, status = map(float, line.split(','))
        soc = get_soc(status,current)
        timestamp = datetime.now().strftime("%d/%m/%Y %H:%M:%S")
        insert_to_instant_db(timestamp,voltage,current,temperature,soc,status)

```

Figure 2.10 Unpacking and packing of data with additional information

Step 3 - Detection of Completed Cycle

Whenever a charging-discharging cycle is completed we are calculating all the required points and inserting them to the database. For this the algorithm used is shown in figure 2.11.

```

prev_mode = get_prev_mode()
if prev_mode == None:
    prev_mode = 0
print(prev_mode, int(status))
cycle = 1 if prev_mode and not int(status) else 0
if cycle:
    print("hello")
# if cycle as completed do feature calculation
    cycle_data = cycle_calculations()
# insert cycle data row to database
    insert_to_cycle_db(cycle_data)

```

Figure 2.11 Detection of a completed cycle

CHAPTER THREE

WEB APPLICATION - BACKEND AND FRONTEND

3.1 THE BACKEND

Backend software serves as the engine behind applications, handling server-side operations, data storage, and business logic. It ensures seamless communication between the user interface and the database, enabling dynamic and responsive functionality. Typically written in languages like Python, Node.js, or Java, backend systems manage user requests, process data, and execute algorithms. They play a pivotal role in supporting the frontend, handling complex tasks, and maintaining data integrity, contributing to the overall functionality and performance of web and mobile applications.

We for this project will be using flask with python as our backend. Flask is justified in web development for its simplicity, flexibility, and rapid prototyping capabilities. With a minimalistic design, it allows quick project initiation and easily integrates with the Python ecosystem. Flask's extensibility and support for building RESTful APIs make it suitable for various applications. Its widespread adoption in the industry ensures a robust community, providing resources and expertise. Overall, Flask strikes a balance between lightweight design and functionality, making it a pragmatic choice for web developers.

3.2 THE DATABASE

A database is a structured collection of data that is organized, stored, and managed to facilitate efficient retrieval and modification. It serves as a digital repository for information, enabling users to store, organize, and retrieve data in a systematic manner. Databases use tables to organize data into rows and columns, providing a relational structure for efficient data management. Database management systems (DBMS) like MySQL, PostgreSQL, and MongoDB play a crucial role in ensuring data integrity, security, and scalability, making them essential tools in modern information technology[31]. In this project we will be using sqlite database due to its responsiveness, simplicity and extensive documentation.

The Sqlite Database

SQLite is a robust, serverless relational database management system (RDBMS) renowned for its lightweight nature and self-contained architecture. Unlike traditional databases, SQLite operates without a separate server process, allowing it to function as a simple yet powerful embedded database engine. It excels in scenarios where a standalone database is necessary, making it a popular choice for mobile applications, embedded systems, and projects with modest data requirements.

One distinctive feature is its file-based storage system, encapsulating the entire database within a single file. This simplifies deployment and management, facilitating seamless integration into various applications. SQLite adheres to standard SQL syntax, supporting transactions and enabling multiple users to access the database concurrently.

Mobile platforms, including Android and iOS, commonly leverage SQLite due to its efficiency and seamless integration. Moreover, desktop applications and small to medium-scale projects benefit from SQLite's speed, cross-platform compatibility, and minimal configuration needs.

3.2.1 DATABASE SCHEMA AND CREATION

Our database mainly consists of two tables which store instantaneous data and cycle data and are called **instant_data** and **cycle_data** respectively. Code snippets to create them also indicating their structure are given in figure 3.1.

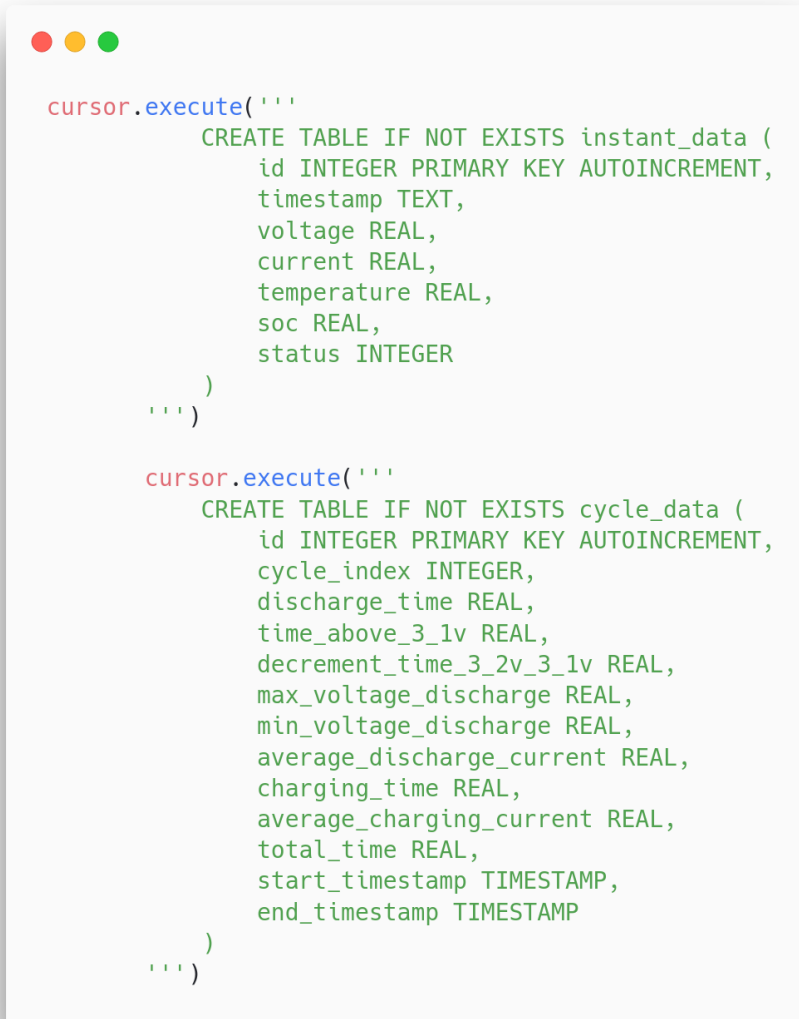
Certainly, let's delve into the significance of each field in the two tables:

1. Instant Data Table:

1. **id** (Primary Key, Autoincrement): A unique identifier for each record, facilitating individual record retrieval.
2. **timestamp`** (Text): Records the timestamp of the data entry, enabling time-based analysis and tracking temporal patterns.
3. **voltage** (Real): Represents the instantaneous voltage of the battery, a crucial metric for monitoring its electrical potential.
4. **current** (Real): Indicates the instantaneous current flowing through the battery, vital for assessing the electrical flow at a specific moment.
5. **temperature** (Real): Captures the temperature of the battery, a critical parameter influencing its performance and lifespan.
6. **soc**(Real): Reflects the state of charge of the battery, providing insights into its energy storage level.
7. **status** (Integer): Denotes the status of the battery at a given moment, conveying information about its operational state (Charging/Discharging).

2. Cycle Data Table:

1. **id** (Primary Key, Autoincrement): A unique identifier for each record, aiding in individual record tracking and retrieval.
2. **cycle_index** (Integer): Represents the index or number assigned to a specific battery cycle, facilitating organization and categorization.



```

cursor.execute('''
    CREATE TABLE IF NOT EXISTS instant_data (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        timestamp TEXT,
        voltage REAL,
        current REAL,
        temperature REAL,
        soc REAL,
        status INTEGER
    )
''')

cursor.execute('''
    CREATE TABLE IF NOT EXISTS cycle_data (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        cycle_index INTEGER,
        discharge_time REAL,
        time_above_3_1v REAL,
        decrement_time_3_2v_3_1v REAL,
        max_voltage_discharge REAL,
        min_voltage_discharge REAL,
        average_discharge_current REAL,
        charging_time REAL,
        average_charging_current REAL,
        total_time REAL,
        start_timestamp TIMESTAMP,
        end_timestamp TIMESTAMP
    )
''')

```

Figure 3.1 Database schema and creation

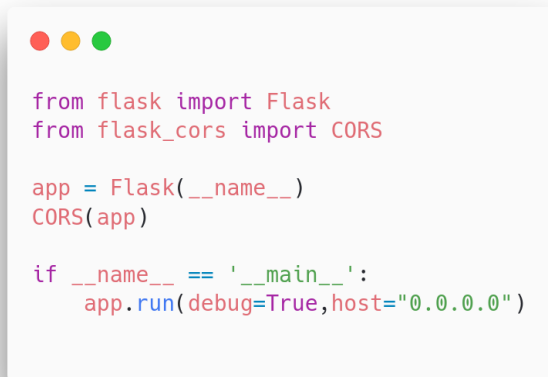
3. Various metrics like `discharge_time`, `time_above_3_1v`, etc.: Record specific performance metrics during a battery cycle, offering detailed insights into discharge and charging characteristics.
4. `start_timestamp` and `end_timestamp` (Timestamp): Indicate the start and end times of a battery cycle, enabling analysis of the duration and temporal aspects of each cycle.

In summary, these tables and their fields collectively provide a comprehensive framework for storing and analyzing data related to lithium-ion battery performance, covering aspects such as status, instantaneous measurements, and detailed cycle-specific information.

3.3 THE WEB SERVER AND APIs

The function of the web server is to host the backend as well as our frontend. The application programming interfaces would aid the communication between the two.

A code snippet to represent the setting up of web server is given in figure 3.2



```
from flask import Flask
from flask_cors import CORS

app = Flask(__name__)
CORS(app)

if __name__ == '__main__':
    app.run(debug=True, host="0.0.0.0")
```

Figure 3.2 Database schema and creation

The full code for backend server and the setup instructions can be found in annexure 2.

3.4 THE FRONTEND

In developing the frontend for the lithium-ion RUL estimation and analysis project, the emphasis was placed on delivering a user experience that seamlessly combines intuition and insight. The focal point of the frontend lies in its dynamic graphs, captivantly illustrating essential metrics such as voltage, current, temperature, and SOC. These graphs serve as an accessible portal into the intricate intricacies of the battery's performance, fostering a nuanced comprehension of its behavior over time.

The incorporation of real-time data updates stands out as a transformative feature. This integration ensures instant feedback on the battery's status, allowing for prompt responses to any performance variations. The real-time functionality elevates the practicality of the frontend, guaranteeing that users remain well-informed about fluctuations and trends as they unfold. In essence, this frontend not only underscores a commitment to clarity and accessibility but also endows users with a dynamic tool for robust RUL estimation and profound battery analysis.

In crafting the frontend for my lithium-ion RUL estimation project, the focus was on an intuitive user experience. Dynamic graphs visually convey vital metrics—voltage, current, temperature, SOC—offering a detailed insight into the battery's performance over time. Real-time data

updates enhance practicality, providing instant feedback for swift responses to performance shifts. This feature ensures users are well-informed about fluctuations and trends, reflecting a commitment to clarity and accessibility. The frontend serves as a dynamic tool, empowering users with profound insights for robust RUL estimation and comprehensive battery analysis.

A snapshot of the frontend is provided in figure 3.3.

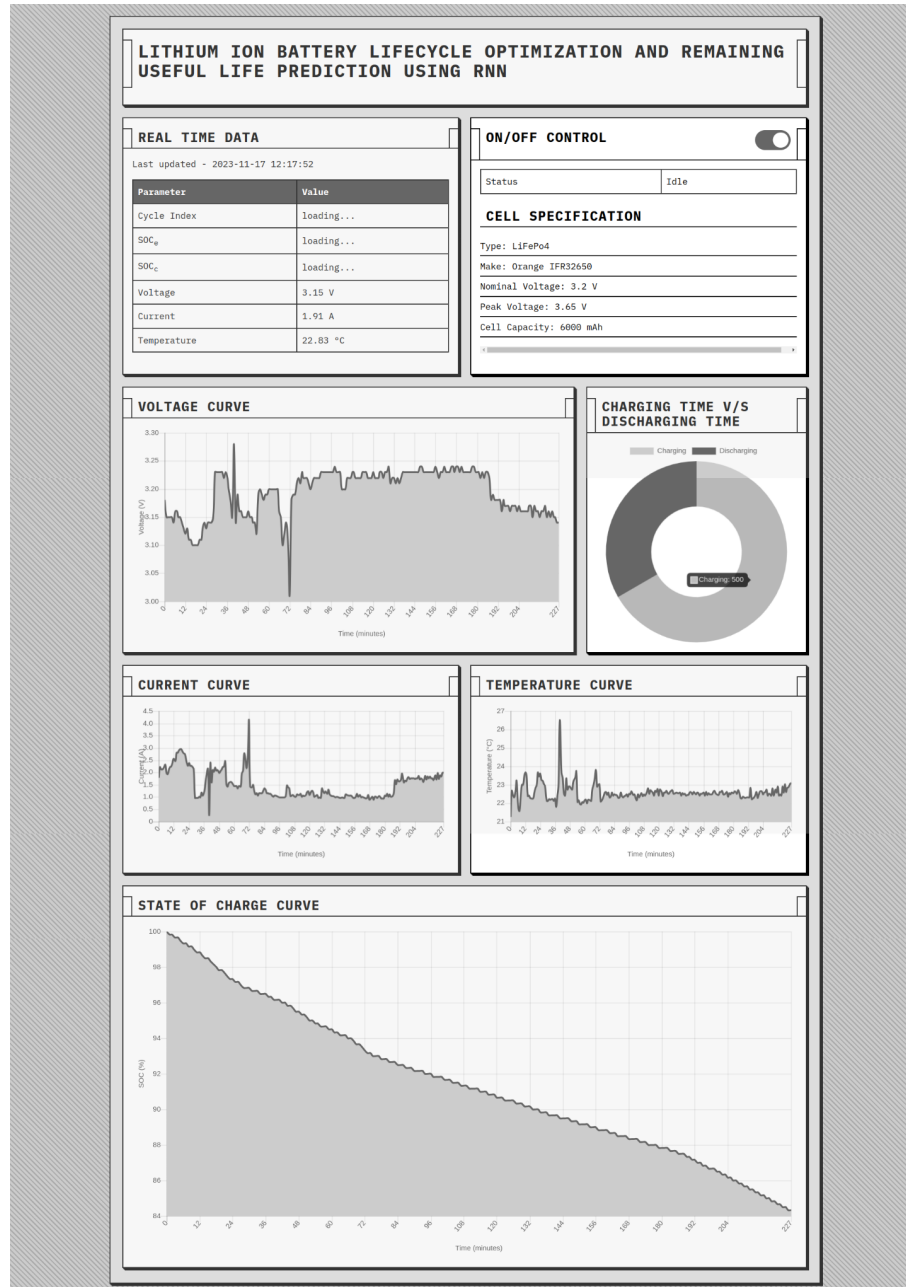


Figure 3.3 The frontend

CHAPTER FOUR

PREDICTIVE MAINTENANCE AND RUL

4.1 OVERVIEW

Predictive Maintenance is similar to having a knowledgeable companion for your machinery. Instead of waiting for issues to arise, it utilizes intelligent technology to forecast when equipment might require attention. It's like the foresight to fix a leaky roof before the storm hits rather than dealing with the aftermath.

Where is it used?

Predictive Maintenance is widely used in various industries such as manufacturing, energy, transportation, healthcare, and more. It is particularly beneficial for industries relying on critical equipment, where unexpected failures can result in significant downtime, production losses, and increased maintenance costs[26].

Why is it used?

- **Cost Reduction:** By predicting when equipment is likely to fail, maintenance activities can be scheduled precisely, minimizing downtime and reducing the need for unnecessary routine maintenance.
- **Increased Equipment Lifespan:** PdM allows for early detection of potential issues, enabling timely repairs or replacements, which can extend the lifespan of machinery and assets.
- **Enhanced Safety:** Predictive Maintenance helps identify potential safety hazards by addressing equipment issues before they lead to failures that could compromise safety.

How is it used?

- **Data Collection:** Sensors and other data sources collect information on equipment performance, temperature, vibration, and other relevant parameters.
- **Data Analysis:** Advanced analytics, machine learning algorithms, and artificial intelligence analyze the collected data to identify patterns, anomalies, and potential failure modes.
- **Predictive Models:** Predictive models are built based on historical data and continuously refined as new data becomes available. These models predict when equipment is likely to fail or require maintenance.

- **Alerts and Notifications:** When the predictive models indicate a potential issue, maintenance teams receive alerts and notifications, allowing them to take proactive measures.

4.2 ADVANTAGES OF PREDICTIVE MAINTENANCE

- **Reduced Downtime:** Predictive Maintenance minimizes unplanned downtime by addressing issues before they lead to equipment failures.
- **Cost Savings:** By optimizing maintenance schedules and avoiding unnecessary repairs, organizations can reduce maintenance costs and increase operational efficiency.
- **Improved Safety:** Early detection of potential equipment failures enhances workplace safety by preventing accidents and injuries.
- **Extended Asset Lifespan:** Timely maintenance and repairs based on predictive insights can extend the lifespan of equipment and assets, maximizing the return on investment.

In summary, Predictive Maintenance is a strategic approach that leverages data and technology to enhance the efficiency, safety, and longevity of industrial equipment, making it a valuable practice in modern maintenance management.

4.3 REMAINING USEFUL LIFE (RUL)

Remaining Useful Life (RUL) is a predictive metric used to estimate the amount of time a system or component is expected to remain functional before reaching the end of its operational life. It is a crucial parameter in predictive maintenance and reliability engineering, providing insights into the remaining lifespan of equipment.[19]

Where is it used?

RUL is employed across various engineering domains and industries, including manufacturing, energy, aerospace, and automotive. It finds application in systems ranging from machinery on a production line to critical components in aircraft engines.

Why is it used?

- **Proactive Maintenance Planning:** RUL allows for proactive maintenance planning by predicting when equipment is likely to reach the end of its useful life. This helps in scheduling timely maintenance activities, reducing downtime, and avoiding sudden failures.

- **Optimizing Asset Management:** In asset-intensive industries, understanding RUL aids in optimizing asset management strategies. It guides decisions on repair, replacement, or refurbishment, ensuring resources are allocated efficiently.
- **Cost Savings:** By predicting the remaining lifespan of components, RUL contributes to cost savings. Unplanned downtime and emergency repairs are minimized, reducing operational costs and enhancing overall cost-effectiveness.

How is it used?

- **Data Collection:** RUL calculations start with the collection of relevant data, including performance metrics, usage patterns, and environmental conditions. This data serves as the foundation for predicting the remaining lifespan of the equipment.
- **Model Development:** Advanced modeling techniques, such as machine learning algorithms or statistical methods, are employed to develop models that correlate historical data with the remaining life of the equipment. These models are trained to make accurate predictions.
- **Monitoring and Prediction:** The developed models are then used to monitor the current health and predict the future performance of the equipment. As new data is acquired, the models are continuously refined to enhance prediction accuracy.
- **Decision Support:** RUL information provides decision support for maintenance teams and asset managers. It helps in determining whether maintenance actions should be taken immediately, in the near future, or if the equipment is still within a safe operational range.

4.4 ADVANTAGES OF REMAINING USEFUL LIFE (RUL)

- **Preventive Maintenance Optimization:** RUL enables the optimization of preventive maintenance schedules, ensuring that maintenance activities are performed just in time to prevent failures & maximize useful life of equipment.
- **Resource Allocation Efficiency:** Knowing the remaining useful life of assets allows for efficient allocation of resources. Resources can be directed towards critical components or systems that are nearing the end of their lifespan, optimizing the overall maintenance strategy.
- **Cost-Effective Asset Management:** By aligning maintenance activities with the predicted remaining life of equipment, organizations can achieve cost-effective asset management. This includes minimizing unnecessary replacements and maximizing the return on investment.

4.5 REMAINING USEFUL LIFE (RUL) ESTIMATION OF LI-ION BATTERIES

Remaining Useful Life (RUL) estimation for Lithium-ion batteries is a critical aspect in the realm of energy storage systems. It involves predicting the time until a battery reaches the end of its reliable operational life, offering valuable insights for maintenance planning and sustainable utilization[32].

Factors Influencing RUL Estimation:

- **Cyclic Aging:** The number of charge and discharge cycles significantly impacts a battery's life. RUL estimation considers the cumulative effect of these cycles on the battery's health.
- **Operating Conditions:** Temperature, voltage levels, and charging rates influence a battery's degradation rate. RUL models take into account the impact of different operating conditions on the overall lifespan.
- **Capacity Fading:** As a battery ages, its ability to store and deliver energy decreases. RUL estimation involves monitoring capacity fade over time to predict when a battery might no longer meet performance requirements.

Advantages of RUL Estimation for Lithium-ion Batteries:

- **Optimized Maintenance Strategies:** RUL estimation enables proactive maintenance scheduling. By predicting when a battery is likely to reach the end of its life, maintenance activities can be planned in advance, reducing downtime and minimizing the risk of unexpected failures.
- **Extended Battery Lifespan:** The ability to predict RUL allows for precise management of charging and discharging cycles. This optimization contributes to extending the overall lifespan of Lithium-ion batteries, enhancing their economic and environmental sustainability.
- **Cost Efficiency:** Proactive maintenance and optimized usage contribute to cost efficiency. RUL estimation helps in avoiding unnecessary replacements and reduces the need for emergency repairs, resulting in overall cost savings.
- **Improved Reliability:** Knowing the RUL of batteries enhances the reliability of energy storage systems. It ensures a consistent and predictable performance, particularly in applications where a reliable power source is crucial.
- **Resource Planning:** RUL estimation aids in efficient resource planning, especially in scenarios where batteries are an integral part of larger systems. It guides decisions on when to invest in new batteries, ensuring resources are utilized wisely.

- **Environmental Impact:** By extending the lifespan of Lithium-ion batteries, RUL estimation contributes to reducing the environmental impact associated with battery disposal and manufacturing. It aligns with sustainable practices in the field of energy storage.

In conclusion, the RUL estimation of Lithium-ion batteries is a vital component of managing energy storage systems effectively. It provides a roadmap for maximizing performance, minimizing costs, and promoting sustainable practices in the context of battery technology.

4.6 LIFE CYCLE OPTIMIZATION OF LI-ION BATTERIES

In view of sustainable energy solutions, the life cycle optimization of lithium-ion batteries plays a pivotal role. This approach involves maximizing the efficiency, performance, and environmental sustainability of lithium-ion batteries throughout their entire life cycle[30].

1. Material Selection:

- **Raw Materials:** Choosing sustainable and ethically sourced materials for battery production reduces environmental impact and supports responsible resource management.
- **Recyclability:** Opting for materials with high recyclability facilitates the recovery of valuable components, minimizing waste and lowering the overall ecological footprint.

2. Manufacturing Process:

- **Energy Efficiency:** Implementing energy-efficient manufacturing processes reduces greenhouse gas emissions associated with battery production, contributing to a more sustainable life cycle.
- **Reduced Waste:** Minimizing manufacturing waste through efficient production techniques and recycling initiatives enhances the overall eco-friendliness of lithium-ion batteries.

3. Battery Usage Phase:

- **Optimized Charging Protocols:** Implementing intelligent charging algorithms and promoting user practices that extend battery life contribute to a longer and more efficient usage phase.
- **Advanced Battery Management Systems (BMS):** Utilizing sophisticated BMS enhances the overall performance and safety of lithium-ion batteries, ensuring optimal operation throughout their life cycle.

4. End-of-Life Considerations:

- **Recycling Programs:** Establishing effective recycling programs ensures the responsible disposal and recovery of materials from spent batteries, reducing environmental impact.
- **Second Life Applications:** Exploring opportunities for second-life applications, such as repurposing used batteries for less demanding tasks, extends their utility and delays the need for recycling.

5. Environmental Impact Assessment:

- **Life Cycle Assessment (LCA):** Conducting a comprehensive LCA helps quantify the environmental impact of lithium-ion batteries at each stage, guiding continuous improvement efforts.
- **Carbon Footprint Reduction:** Implementing strategies to minimize the carbon footprint of lithium-ion batteries contributes to a more sustainable and environmentally friendly energy storage solution.

Advantages Of Life Cycle Optimization

Sustainability: Life cycle optimization promotes environmentally sustainable practices, reducing the overall impact of lithium-ion batteries on ecosystems and resources.

- **Efficiency:** By maximizing efficiency at every stage, from manufacturing to end-of-life, life cycle optimization ensures that the energy storage system operates at its best throughout its entire lifespan.
- **Cost-effectiveness:** Effective recycling and second-life applications contribute to cost savings by recovering valuable materials and delaying the need for new battery production.
- **Positive Social Impact:** Ethical material sourcing and responsible manufacturing contribute to positive social impacts, fostering a sense of responsibility and accountability in the industry.

In conclusion, life cycle optimization of lithium-ion batteries is a holistic approach that aligns with sustainable engineering practices. It addresses environmental concerns, enhances operational efficiency, and supports a responsible and ethical approach to energy storage solutions.

CHAPTER FIVE

COMPARATIVE STUDY OF DIFFERENT MODELS USED FOR RUL PREDICTION

5.1 DATASET OVERVIEW

“The Hawaii Natural Energy Institute examined 14 NMC-LCO 18650 batteries with a nominal capacity of 2.8 Ah, which were cycled over 1000 times at 25°C with a CC-CV charge rate of C/2 rate and discharge rate of 1.5C.”

From that source dataset, we created features that showcase the voltage and current behavior over each cycle. Those features can be used to predict the remaining useful life (RUL) of the batteries. The dataset contains the summary of the 14 batteries.

Variables:

- Cycle Index: number of cycle
- F1: Discharge Time (s)
- F2: Time at 4.15V (s)
- F3: Time Constant Current (s)
- F4: Decrement 3.6-3.4V (s)
- F5: Max. Voltage Discharge (V)
- F6: Min. Voltage Charge (V)
- F7: Charging Time (s)
- Total time (s)
- RUL: target

You may check on GitHub how the dataset was built:

https://github.com/ignavinuales/Battery_RUL_Prediction

Link to the dataset :

<https://www.kaggle.com/datasets/ignaciovinuales/battery-remaining-useful-life-rul>

Creating a heatmap for the correlation among features is a useful visual tool to understand the relationships between variables in a dataset. It helps to detect the prominent features and eliminate the irrelevant features from the dataset.

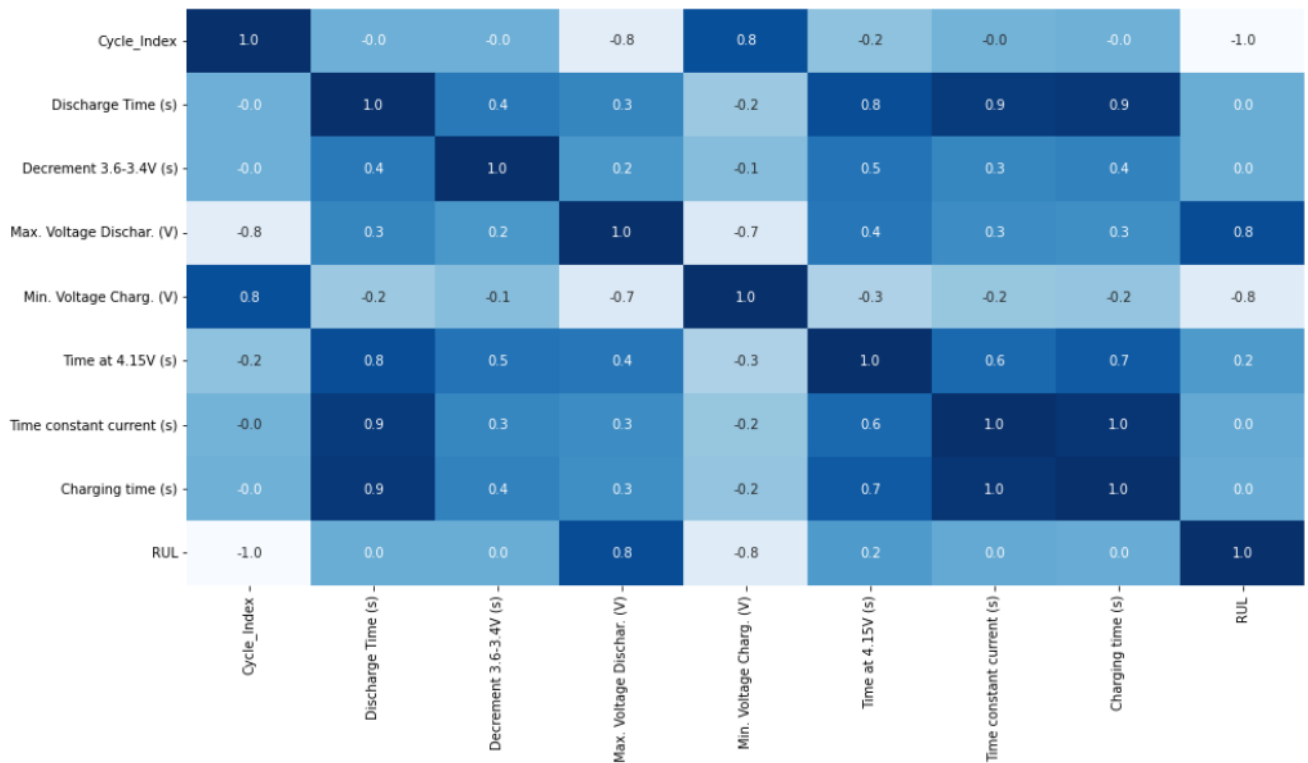


Figure 5.1 Correlation Among Features

Furthermore, we will use some inbuilt functions in python to study and get an in-depth overview of our dataset.

The `head()` function displays the first five rows of our DataFrame. The figure shows the implementation of the same.

```
df.head()
```

	Cycle_Index	Discharge Time (s)	Decrement 3.6-3.4V (s)	Max. Voltage Dischar. (V)	Min. Voltage Charg. (V)	Time at 4.15V (s)	Time constant current (s)	Charging time (s)	RUL
0	1	2595.30	1151.488500	3.670	3.211	5460.001	6755.01	10777.82	1112
1	2	7408.64	1172.512500	4.246	3.220	5508.992	6762.02	10500.35	1111
2	3	7393.76	1112.992000	4.249	3.224	5508.993	6762.02	10420.38	1110
3	4	7385.50	1080.320667	4.250	3.225	5502.016	6762.02	10322.81	1109
4	6	65022.75	29813.487000	4.290	3.398	5480.992	53213.54	56699.65	1107

Figure 5.2 First five rows of dataset

The `info()` function provides a concise summary of the DataFrame, including information about the data types, non-null values, and memory usage. The output of this function on our dataset is as shown below.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1076 entries, 0 to 1075
Data columns (total 9 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Cycle_Index                          1076 non-null   int64
1   Discharge Time (s)                  1076 non-null   float64
2   Decrement 3.6-3.4V (s)              1076 non-null   float64
3   Max. Voltage Dischar. (V)           1076 non-null   float64
4   Min. Voltage Charg. (V)             1076 non-null   float64
5   Time at 4.15V (s)                  1076 non-null   float64
6   Time constant current (s)           1076 non-null   float64
7   Charging time (s)                   1076 non-null   float64
8   RUL                                  1076 non-null   int64
```

Figure 5.3 Info of dataset

The describe() function generates descriptive statistics of our DataFrame, including measures of central tendency, dispersion, and shape of the distribution. The output of this function on our dataset has been attached.

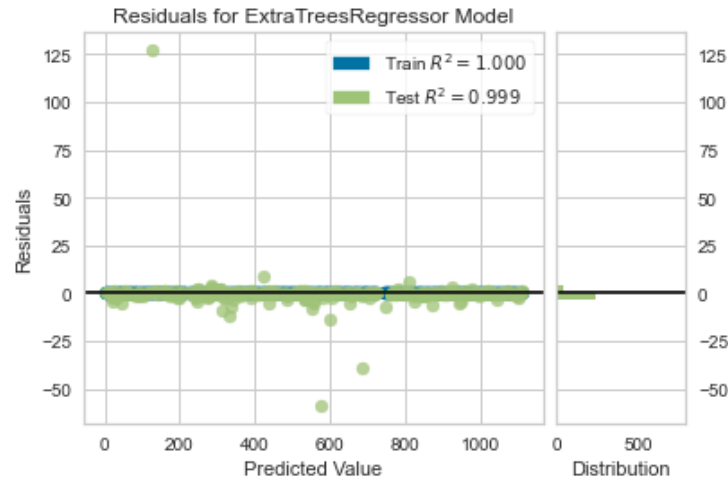
```
df.describe()
```

	Cycle_Index	Discharge Time (s)	Decrement 3.6-3.4V (s)	Max. Voltage Dischar. (V)	Min. Voltage Charg. (V)	Time at 4.15V (s)	Time constant current (s)	Charging time (s)	RUL
count	1076.000000	1076.000000	1076.000000	1076.000000	1076.000000	1076.000000	1076.000000	1076.000000	1076.000000
mean	556.878253	4899.191143	2062.10433	3.929328	3.566317	3863.478884	5972.033439	10089.222221	556.121747
std	323.594005	33317.514216	18051.02485	0.077341	0.121406	8892.439128	27000.077287	26671.212644	323.594005
min	1.000000	168.000000	-64192.71600	3.670000	3.086000	269.984000	1124.350000	5277.630000	0.000000
25%	271.750000	1195.347500	332.00000	3.869000	3.478000	1864.884179	2636.337500	7473.617500	276.750000
50%	559.500000	1617.405000	474.00000	3.928000	3.554000	3133.732000	4004.420000	8197.250000	553.500000
75%	836.250000	1934.392500	628.20000	3.982000	3.661000	4166.264000	5084.350000	8590.460000	841.250000
max	1113.000000	679307.970000	283252.12500	4.290000	4.053000	219923.996000	604521.090000	604521.090000	1112.000000

Figure 5.4 Description of dataset

5.2 EXTRA TREES REGRESSOR

- **Overview:** Extra Trees Regressor is an ensemble learning method that constructs a multitude of decision trees during training. It builds on the concept of Random Forests by introducing additional randomness in the feature selection process.[31]



5.5 Residuals Plot for ET

- **Advantages:**
 - Robust overfitting.
 - Reduced variance compared to individual decision trees.
 - Effective for complex, high-dimensional data.

5.3 LINEAR REGRESSION

- **Overview:** Linear Regression is a simple yet powerful regression algorithm that establishes a linear relationship between the input features and the target variable. It aims to find the best-fitting line through the data points[38].

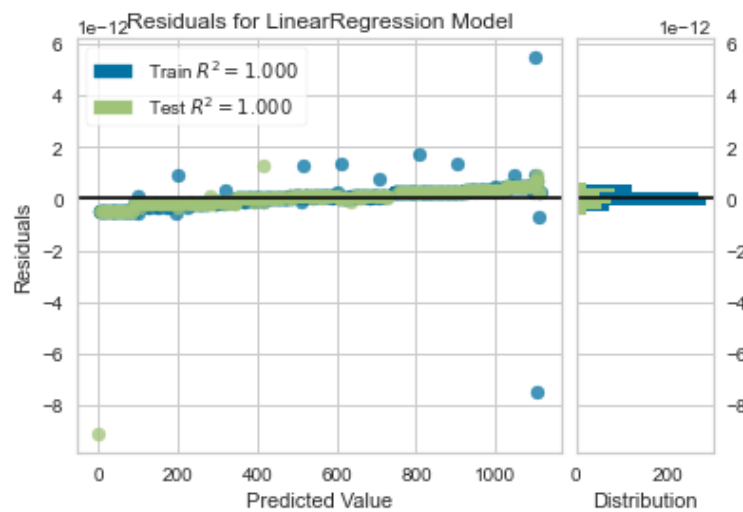


Figure 5.6 Residuals Plot for LR

- **Advantages:**
 - Interpretable and easy to implement.
 - Suitable for datasets with a linear relationship.

5.4 RIDGE REGRESSION

- **Overview:** Ridge Regression is a regularized linear regression technique that adds a penalty term to the cost function, preventing the model from becoming too complex. It's effective in handling multicollinearity[38].

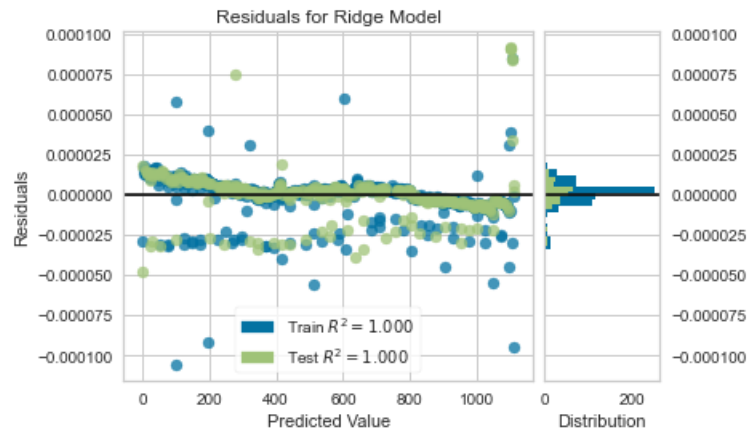


Figure 5.7 Residuals Plot RR

- **Advantages:**
 - Mitigates multicollinearity issues.
 - Provides a balance between simplicity and accuracy.

5.5 LASSO REGRESSION

```
reg_model_lasso = create_model('lasso')
plot_model(reg_model_lasso, plot='error')
plot_model(reg_model_lasso, plot='residuals')
plot_model(reg_model_lasso, plot='feature')
```

Figure 5.8 Code for Lasso Regression

- **Overview:** Lasso Regression, similar to Ridge, is a regularized linear regression method. However, it introduces L1 regularization, which encourages sparsity in the coefficient values, effectively performing feature selection.

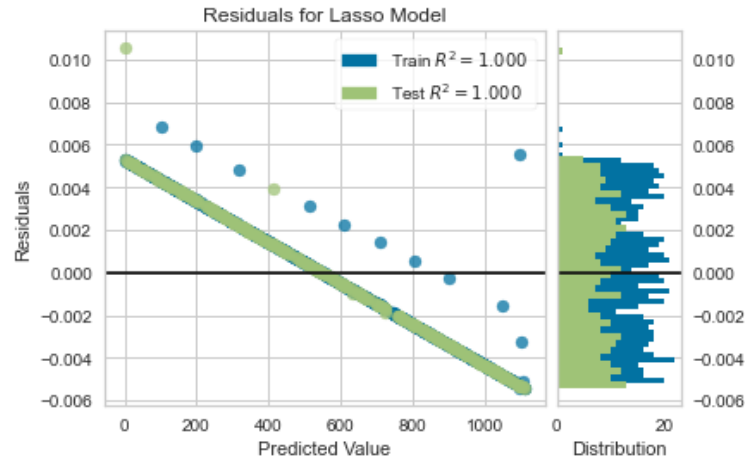
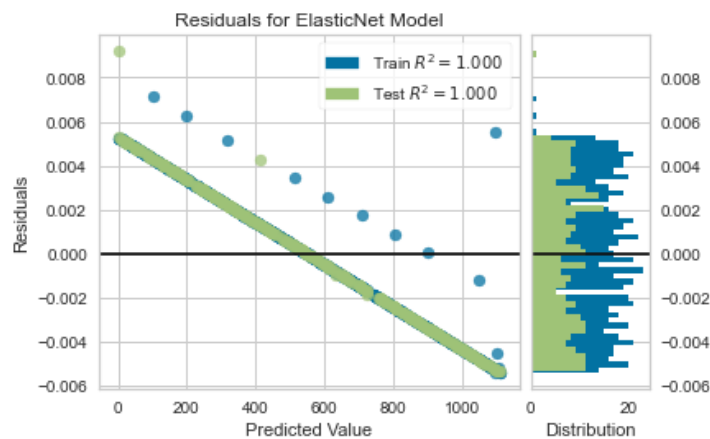


Figure 5.9 Residuals Plot Lasso Regression

- **Advantages:**
 - Automatic feature selection.
 - Useful when dealing with high-dimensional data.

5.6 ELASTIC NET

- **Overview:** Elastic Net is a hybrid of Ridge and Lasso Regression, combining both L1 and L2 regularization terms. It provides a balance between the advantages of Ridge and Lasso, addressing their individual limitations[38].



5.10 Residuals Plot EN

- **Advantages:**
 - Combines feature selection and handles multicollinearity.
 - Robust to a moderate degree of multicollinearity.

5.7 RANDOM FOREST REGRESSION

Overview: Random Forest Regression is an ensemble learning technique that constructs a multitude of decision trees during training. It aggregates the predictions of individual trees to improve overall accuracy and reduce overfitting.[35]

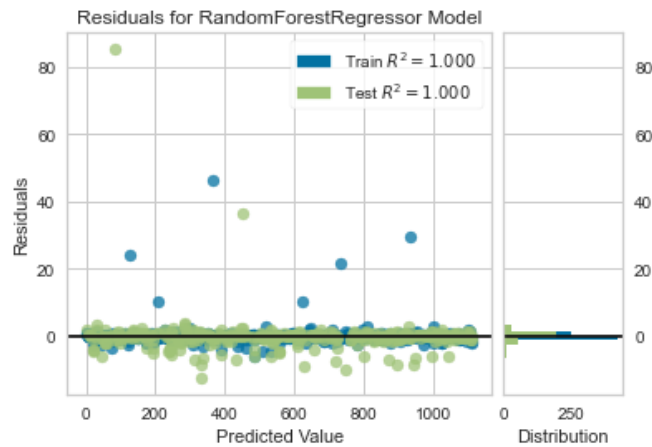


Figure 5.11 Residuals Plot for RFR

- **Advantages:**
 - High accuracy and robustness.
 - Handles complex relationships in data.

```
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error

# Assuming df is your DataFrame with features and target column
X = df.drop('RUL', axis=1)
y = df['RUL']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train the XGBoost model
xgb_model = xgb.XGBRegressor()
xgb_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = xgb_model.predict(X_test)

# Evaluate the model
mae = mean_absolute_error(y_test, y_pred)
print(f'Mean Absolute Error: {mae}')
```

Figure 5.12 Code for implementation of XGBoost Model

5.8 EXTREME GRADIENT BOOSTING (XGBOOST)

- **Overview:** XGBoost is an efficient and scalable implementation of gradient boosting. It builds a series of weak learners, typically decision trees, sequentially, each correcting the errors of the previous one.[37]

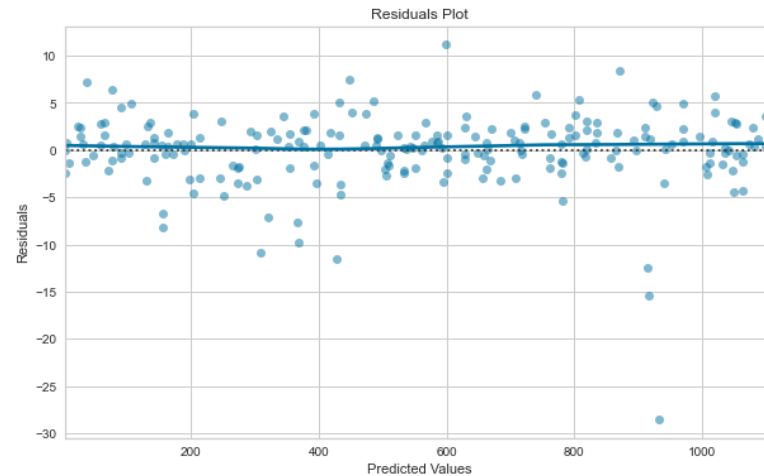


Figure 5.13 Residuals Plot for XGB

- **Advantages:**
 - Exceptional predictive performance.
 - Handles missing data and regularization.

```

from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split

# Assuming X_train, y_train, X_test, y_test are your training and testing data
# You should replace these with your actual data

# Step 1: Train the SVR model
svr_model = SVR(kernel='linear') # You can choose the appropriate kernel
svr_model.fit(X_train, y_train)

# Step 2: Make predictions on the test data
y_pred = svr_model.predict(X_test)

# Step 3: Calculate the Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)

print(f'Mean Squared Error (MSE): {mse}')
```

Figure 5.14 Code for implementation of SVR Model

5.9 SUPPORT VECTOR REGRESSOR (SVR)

- **Overview:** Support Vector Regressor is a regression technique that utilizes support vector machines to find the optimal hyperplane that best fits the data points while minimizing the error[28].

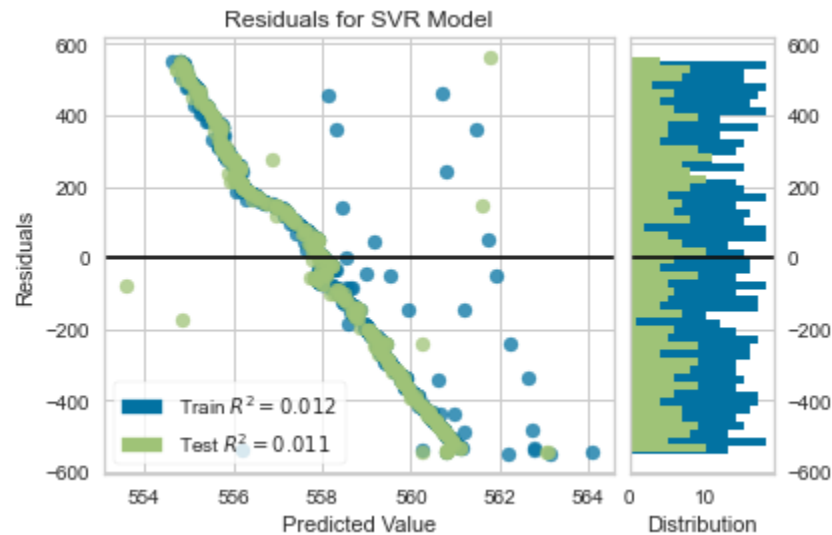


Figure 5.15 Residuals Plot for SVR

- **Advantages:**
 - Effective in high-dimensional spaces.
 - Versatile for different types of relationships in data.

After the study of all the eight models mentioned above, we have curated the predicted RUL values of all models in a tabular format in a dataframe. It contains a comparison of the predicted RUL values of each of the models alongside the actual RUL values present in our dataset.

The `describe()` function generates descriptive statistics of our result dataframe, including measures of central tendency, dispersion, and shape of the distribution. The output of this function on our dataset has been attached.

5.10 RESULTS

After studying all the models for our dataset, our main goal was to identify the best model among them. Mean Squared Error (MSE) is a common metric used to evaluate the performance of regression models. It measures the average squared difference between the predicted values and the actual values of the target variable. A smaller MSE indicates that the model's predictions are

closer to the actual values. Easy to calculate and widely used in optimization algorithms for model training.

So, after training our models on our dataset, we have plotted a bar plot for MSE comparison of all the models. The figure of plot is attached below:

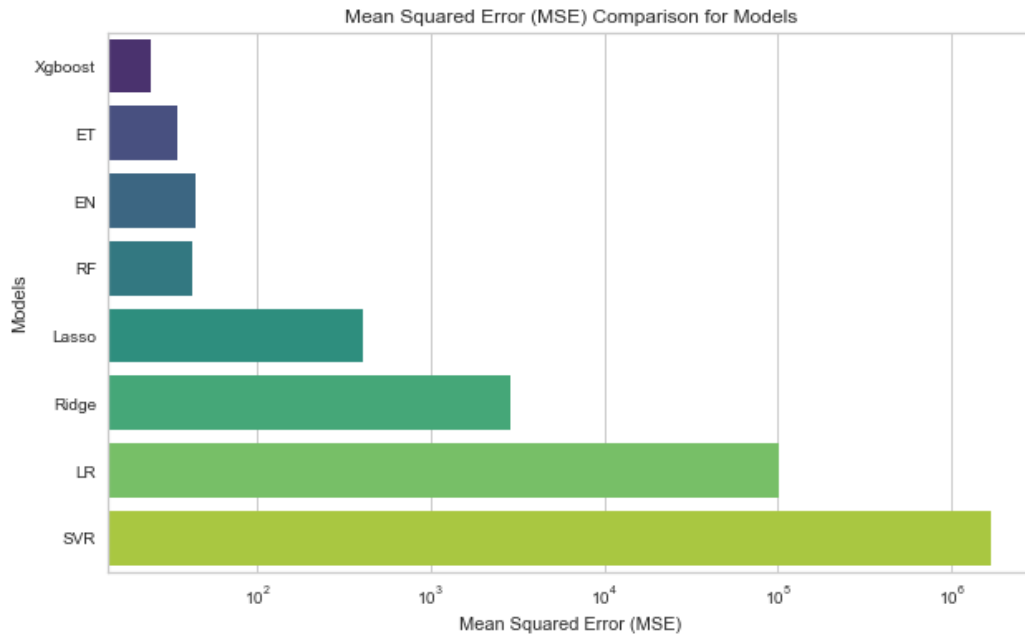


Figure 5.16 Box Plot for MSE Comparison of Models

As clearly visible, the Xgboost model has the least value of MSE and thus can be used for the prediction of RUL.

Final prediction of RUL After 216 Rows of Data

```
results_df.describe()
```

	Actual RUL	Model 1 Predicted RUL	Model 2 Predicted RUL	Model 3 Predicted RUL	Model 4 Predicted RUL	Model 5 Predicted RUL	Model 6 Predicted RUL	Model 7 Predicted RUL	Model 8 Predicted RUL
count	216.000000	216.000000	216.000000	216.000000	216.000000	216.000000	216.000000	216.000000	216.000000
mean	555.300926	554.981852	555.300926	555.300925	555.300903	555.300905	555.464444	556.419189	487.832023
std	327.762516	327.753226	327.762516	327.762513	327.759390	327.759399	327.922081	327.528168	1349.976242
min	1.000000	1.000000	1.000000	0.999971	1.005271	1.005258	2.090000	4.430970	-15006.682686
25%	271.500000	271.500000	271.500000	271.499996	271.502661	271.502654	271.462500	274.766739	328.249591
50%	556.500000	552.435000	556.500000	556.499988	556.499922	556.499920	554.050000	557.083466	551.376128
75%	820.250000	820.250000	820.250000	820.249999	820.247380	820.247382	820.225000	818.859558	859.901870
max	1110.000000	1110.210000	1110.000000	1110.000002	1109.994610	1109.994620	1109.100000	1110.830078	5491.743708

Figure 5.17 Final Predicted RUL

Some Insights from generated data

A typical discharging voltage vs time curve and current vs time graph

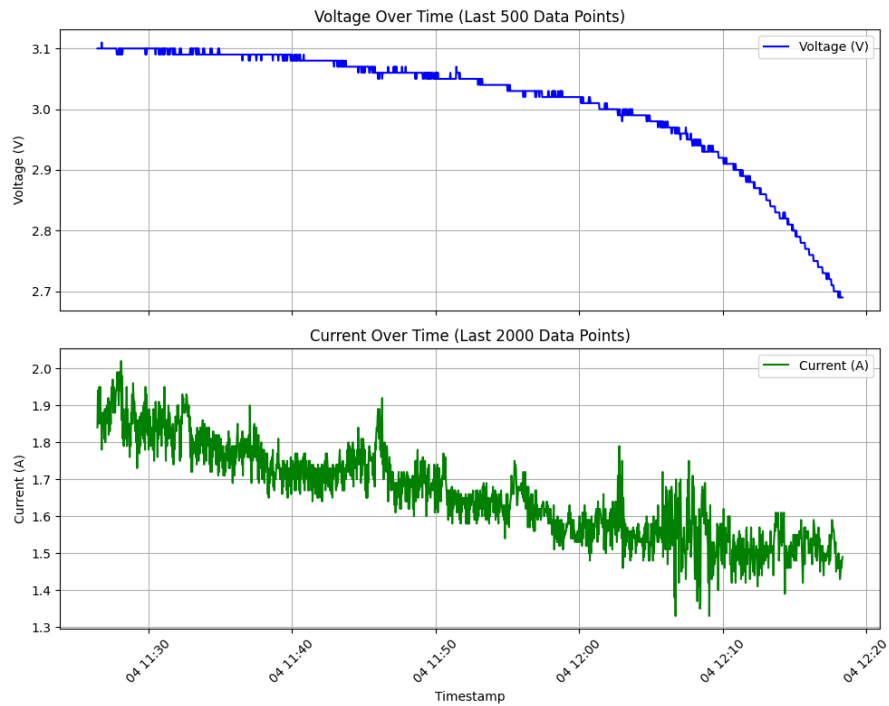


Figure 5.18 Discharging voltage vs time curve and current vs time graph

Charging voltage and current characteristics wrt time (passed through an averaging filter)

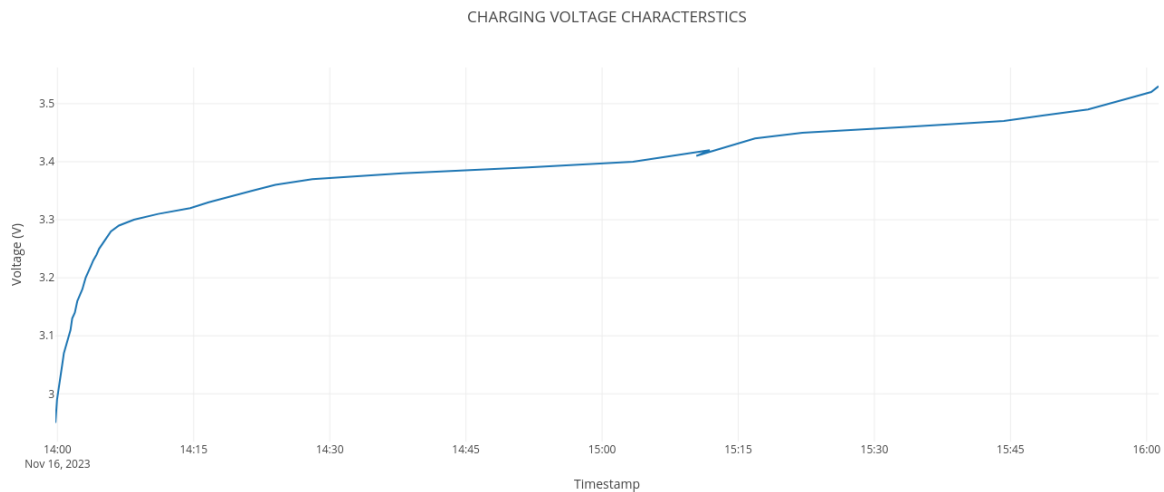


Figure 5.19 Charging voltage vs time

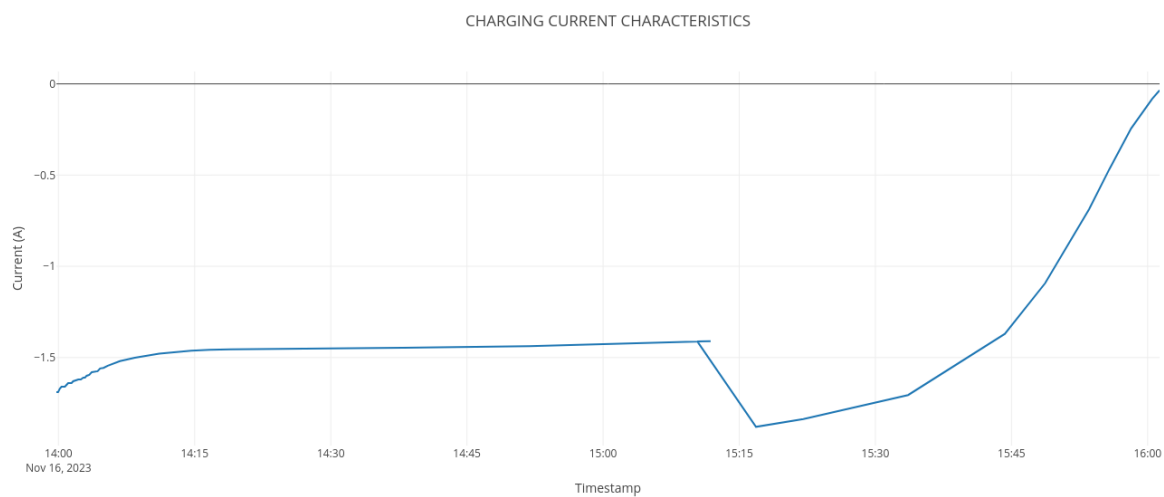


Figure 5.20 Charging Current vs time

CHAPTER SIX

CONCLUSION AND FUTURE SCOPES

6.1 CONCLUSION

The undertaking commenced with the overarching objective of enhancing the operational longevity of lithium-ion batteries through the judicious integration of real-time data and advanced machine learning methodologies. The foundational aspect of this initiative was the meticulous generation of a comprehensive and high-fidelity dataset, facilitated by a dependable data logging apparatus and a sophisticated communication infrastructure. This rigorous data collection process is instrumental in gaining nuanced insights into the dynamic behaviors exhibited by lithium-ion batteries across various lifecycle phases.

To enhance user accessibility and operational control, a web-based frontend application was strategically implemented. This interface not only facilitates user interaction with the data but also ensures a user-friendly experience, thereby democratizing the advantages of the project. The decision to imbue the device with functionalities extending beyond mere data logging, specifically encompassing protective measures and intelligent control mechanisms, underscores a commitment to multifaceted utility, addressing both monitoring imperatives and safety considerations.

Within the realm of machine learning, the project distinguishes itself through a systematic evaluation of diverse models. The discerning selection of the **XGBoost algorithm** for Remaining Useful Life (RUL) estimation reflects a methodical and analytical approach, acknowledging the algorithm's efficacy in capturing intricate relationships embedded in the dataset. The successful derivation of the battery's RUL after 216 life cycles constitutes a tangible validation of the selected methodology, affirming the predictive model's effectiveness.

The proof of concept demonstration constitutes a significant milestone, affirming the seamless synergy of project components toward the realization of intended objectives. Beyond the controlled environment of a test cell, the assertion that the device is scalable for application to larger batteries introduces a dimension of practicality and real-world relevance. This scalability elevates the project beyond conceptual merit, positioning it as a potentially transformative solution for optimizing the performance and lifespan of lithium-ion batteries in diverse operational contexts. As the project advances, further refinements, considerations for real-world deployment, and scalability concerns will inevitably guide the trajectory of subsequent developmental phases.

6.2 FUTURE SCOPES

The future applications of this idea are limitless and possibilities are for high commercial viabilities. Some of them are :

1. The device can be printed on small pcb with a more specialized and compact microprocessor and microcontrollers.
2. The device can be **incorporated in BMS** of an electric vehicle. It can be an added advantage increasing the range of the EV significantly.
3. **An user recommender system** can be expanded upon this project which will enable the users to get insights and commands to guide them towards better usage of their battery.
4. **Smart Grid Integration** : By communicating with the grid, the device can participate in demand response programs, helping to manage peak loads and contributing to a more sustainable and resilient energy infrastructure.

ANNEXURE 1

SETTING UP OF RASPBERRY PI

Requirements:-

Hardware

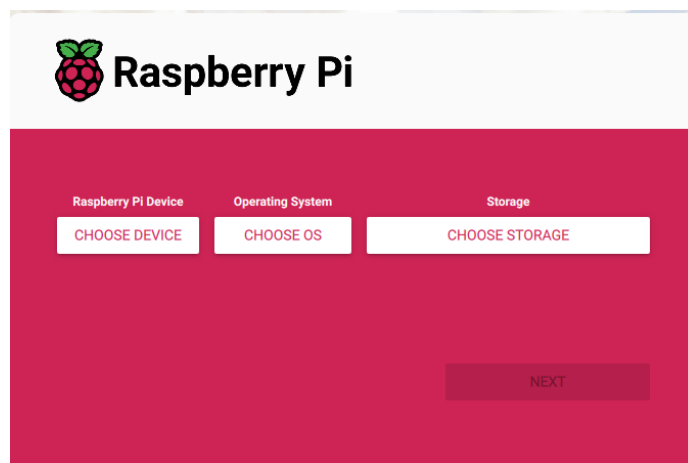
1. Raspberry Pi (3B or above)
2. SD Card/ USB Flash Drive
3. Micro Usb Cable
4. HDMI Cable
5. Monitor
6. External Mouse and Keyboard

Software

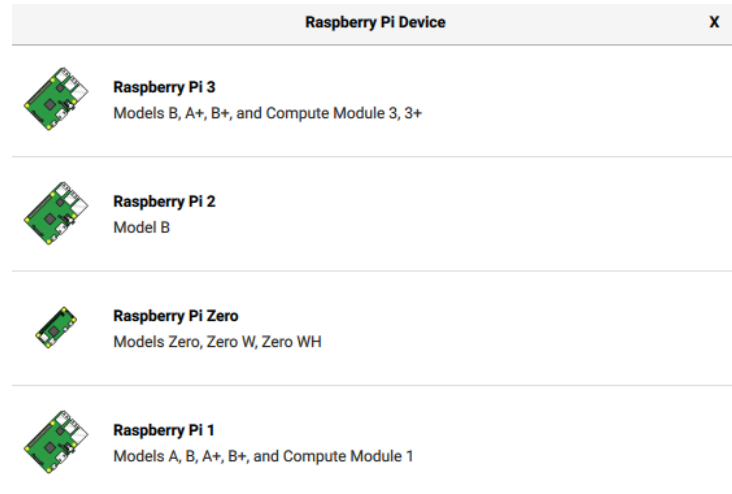
1. RPI Imager (<https://www.raspberrypi.com/software/>)

Instructions

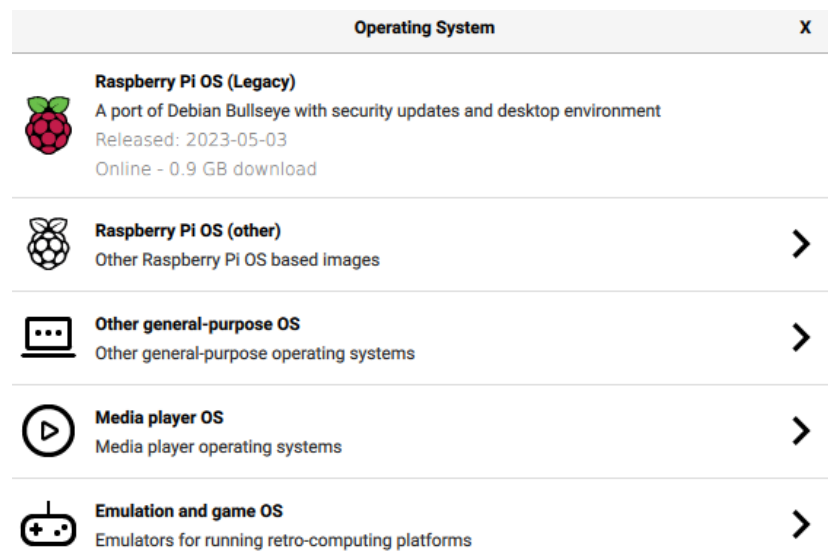
1. Download RPI Imager software form above link. You will be greeted with a screen as given below.



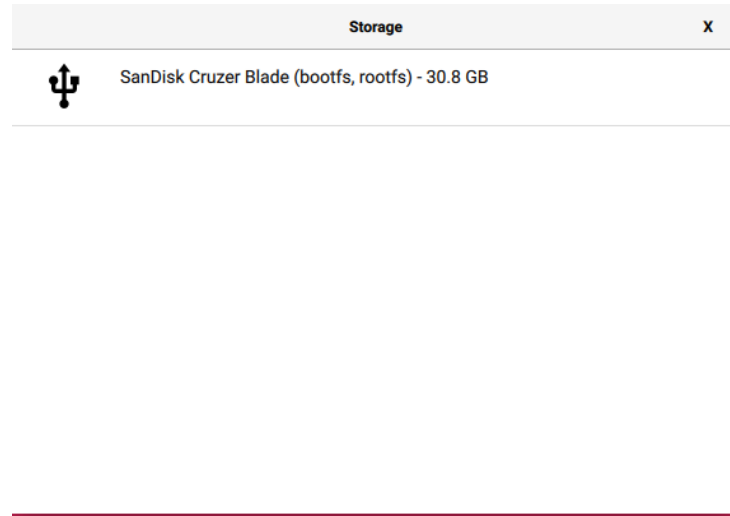
2. Choose your device.



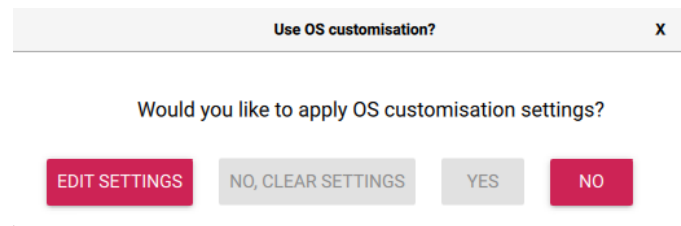
3. Choose the operating system you want to use. It is preferable to use lite versions for better performance on low end raspberry pi devices.



4. Connect your sd card/ usb flash drive to your system and click choose device button and select your device.



5. Click on the next button. A window will appear asking your for more options.



6. Select edit settings option and fill in your network details. This will allow the raspberry pi to connect to your network on boot itself.

 A screenshot of a settings window with three tabs: 'GENERAL' (active), 'SERVICES', and 'OPTIONS'. The 'GENERAL' tab contains several configuration options:

- ☐ Set hostname:
- ☐ Set username and password
 - Username:
 - Password:
- ☐ Configure wireless LAN
 - SSID:
 - Password:
 - ☐ Show password ☐ Hidden SSID
 - Wireless LAN country:
- ☐ Set locale settings
 - Time zone:
 - Keyboard layout:

 At the bottom right of the window is a red 'SAVE' button.

7. Click on save and write your disk to complete booting your OS to the storage device.

8. Disconnect the sd card/ usb drive after the process is completed and connect it to raspberry pi.
9. Power your raspberry pi using a micro usb cable and an adapter.
10. Connect your raspberry pi to a monitor using hdmi cable.
11. Now you can use your computer as a normal linux desktop.

ANNEXURE 2

ARDUINO UNO PROGRAM

```
#define voltagePin A1
#define currentPin A0
#define temperaturePin A3
#define switchPin 12

float ref_voltage = 5;
bool charging = false;

void setup()
{
    pinMode(switchPin, OUTPUT);
    pinMode(2, OUTPUT);
    pinMode(3, OUTPUT);
    digitalWrite(2, HIGH);
    digitalWrite(3, HIGH);
    digitalWrite(12, HIGH);
    Serial.begin(9600);
}

double computeTemperature()
{
    int adcData = analogRead(temperaturePin);
    float voltage = adcData * (5 / 1024);
    float temperature = (voltage) * 100;
    return temperature;
}

float computeVoltage()
{
    float R1 = 30000.0;
    float R2 = 7500.0;
    float adc_voltage = 0.0;
    float in_voltage = 0.0;
    int adc_value = 0;
    adc_value = analogRead(voltagePin);
```

```

    adc_voltage = (adc_value * 5) / 1024.0;
    in_voltage = adc_voltage / (R2 / (R1 + R2));
    return in_voltage;
}

float computeCurrent()
{
    float current_adc = 0;
    float current_voltage = 0;
    float in_current = 0;
    current_adc = analogRead(currentPin);
    current_voltage = (current_adc * ref_voltage * 1000) / 1024.0;
    in_current = (current_voltage - 2500) * 1000 / 100.00;
    return in_current;
}

void switchControl(float voltage)
{
    if (charging && voltage > 3.60)
    {
        digitalWrite(switchPin, HIGH);
        charging = false;
    }
    else if (!charging && voltage < 2.6)
    {
        digitalWrite(switchPin, LOW);
        charging = true;
    }
}

void displayData(float voltage, float current, float temperature)
{
    Serial.print(voltage);
    Serial.print(", ");
    Serial.print((current + 100) / 1000, 2);
    Serial.print(", ");
    Serial.print(temperature);
    Serial.print(", ");
    Serial.println(charging);
}

```

```
void loop()
{

    float voltage = 0;
    float current = 0;
    float temperature = 0;

    if (Serial.available() > 0)
    {
        String msg = Serial.readStringUntil('\n');
        if (msg == "Charging"){
            digitalWrite(switchPin, LOW);
            charging = true;
        }else{
            digitalWrite(switchPin,HIGH);
            charging = false;
        }
        Serial.readString();
        char command = Serial.read();
        if (command == '1')
        {
            digitalWrite(switchPin, HIGH);
            charging = false;
        }
        else if (command == '0')
        {
            digitalWrite(switchPin, LOW);
            charging = true;
        }

    }

    for (int i = 0; i < 50; i++)
    {
        voltage += computeVoltage();
        current += computeCurrent();
        temperature += computeTemperature();
        delay(5);
    }
}
```



```
voltage = voltage / 200;  
current /= -200;  
temperature /= 200;  
  
switchControl(voltage);  
displayData(voltage + 0.04, current, temperature);  
  
delay(6000);  
}
```

ANNEXURE 3

PYTHON PROGRAMS

serial_communication.py

```

import serial
from datetime import datetime
import time
from database import create_db, insert_to_instant_db, close_db, get_prev_mode
create_db()
from soc import get_soc
# from cycle_data import cycle_calculations
import requests
from variables import uri

def fetch_status():
    url = "http://192.168.1.8:5000/api/state"
    response = requests.get(url)

    if response.status_code == 200:
        data = response.json()
        return "Charging" if data[0][1] else "Discharging"
    else:
        print("Failed to fetch data. Status code:", response.status_code)
        print("Response content:", response.text)

if __name__ == '__main__':
    ser = serial.Serial('/dev/ttyACM0', 9600)
    while True:
        try:
            line = ser.readline().decode().strip()
            print(line)
            voltage, current, temperature, status = map(float, line.split(','))

            prev_mode = get_prev_mode()
            if prev_mode == None:
                prev_mode = 0
            cycle = 1 if prev_mode and not int(status) else 0

```

```

if cycle:
    print("hello")
    # if cycle is completed do feature calculation
    cycle_data = cycle_calculations()
    # insert cycle data row to database
    insert_to_cycle_db(cycle_data)

    # measuring the soc
    soc = get_soc(status,current)
    timestamp = datetime.now().strftime("%d/%m/%Y %H:%M:%S")

    # inserting instantaneous data to database
    insert_to_instant_db(timestamp,voltage,current,temperature,soc,status)
    print(f'{timestamp}: Voltage: {voltage} V, Current: {current} A, Temperature:
{temperature}, Mode: {status}')
    # check if cycle has completed

    # communicate form python to arduino
    command = fetch_status()
    ser.write(command.encode('utf-8'))

    time.sleep(10)

except KeyboardInterrupt:
    print("Communication Broken!")
    break

ser.close()
close_db()

```

database.py

```

import sqlite3
conn = sqlite3.connect('lion_dataset.db')
cursor = conn.cursor()

def create_db():
    try:

```

```

cursor.execute("""
    CREATE TABLE IF NOT EXISTS status (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        status INTEGER
    )
""")

```

```

cursor.execute("""
    INSERT INTO status (status)
    VALUES (?)
""", ("1"))

```

```

cursor.execute("""
    CREATE TABLE IF NOT EXISTS instant_data (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        timestamp TEXT,
        voltage REAL,
        current REAL,
        temperature REAL,
        soc REAL,
        status INTEGER
    )
""")

```

```

cursor.execute("""
    CREATE TABLE IF NOT EXISTS cycle_data (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        cycle_index INTEGER,
        discharge_time REAL,
        time_above_3_1v REAL,
        decrement_time_3_2v_3_1v REAL,
        max_voltage_discharge REAL,
        min_voltage_discharge REAL,
        average_discharge_current REAL,
        charging_time REAL,
        average_charging_current REAL,
        total_time REAL,
        start_timestamp TIMESTAMP,
        end_timestamp TIMESTAMP
    )
""")

```

```

conn.commit()
print("database created successfully...")

```

```

    print("Proceeding further...")
    return True
except sqlite3.Error as e:
    print("database creation failed...")
    print("Exiting program...")
    print(f"Error: {e}")
    return False

def insert_to_instant_db(timestamp,voltage,current,temperature,soc,status):
    cursor.execute("""
        INSERT INTO instant_data (timestamp, voltage, current, temperature,soc, status)
        VALUES (?, ?, ?, ?, ?, ?)
    """, (timestamp, voltage, current, temperature, soc, status))
    conn.commit()

def insert_to_cycle_db(data):
    cursor.execute("""
        INSERT INTO cycle_data (cycle_index,
            discharge_time,time_above_3_1v,
            decrement_time_3_2v_3_1v,
            max_voltage_discharge ,
            min_voltage_discharge ,
            average_discharge_current ,
            charging_time ,
            average_charging_current ,
            total_time)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?)
    """, (data.cycle_index,
        data.discharge_time,
        data.time_above_3_1v,
        data.decrement_time_3_2v_3_1v,
        data.max_voltage_discharge ,
        data.min_voltage_discharge ,
        data.average_discharge_current ,
        data.charging_time ,
        data.average_charging_current ,
        data.total_time))
    conn.commit()

def update_status(status):
    cursor.execute("""
        UPDATE status
        SET status = ?
        WHERE id = ?
    """)

```

```

    "", (status, 1))
    conn.commit()

def get_status():
    cursor.execute('SELECT * FROM instant_data')
    rows = cursor.fetchall()
    return rows[0][0]

def get_instant_db():
    cursor.execute('SELECT * FROM instant_data')
    rows = cursor.fetchall()
    return rows

def get_cycle_db():
    cursor.execute('SELECT * FROM cycle_data')
    rows = cursor.fetchall()
    return rows

def get_prev_soc():
    cursor.execute('SELECT * FROM instant_data ORDER BY timestamp DESC LIMIT 1')
    row = cursor.fetchone()
    return row[5] if row else None

def get_prev_mode():
    cursor.execute('SELECT * FROM instant_data ORDER BY timestamp DESC LIMIT 1')
    row = cursor.fetchone()
    return row[6] if row else None

def get_prev_cycle_index():
    cursor.execute('SELECT * FROM cycle_data ORDER BY cycle_index DESC LIMIT 1')
    row = cursor.fetchone()
    return row[0] if row else None

def close_db():
    conn.close()

```

cycle_data.py

```

from database import get_prev_cycle_index, get_cycle_db
from variables import dx

prev_index = get_prev_cycle_index()
db = get_cycle_db()

```

```

dx = dx()
data = {}

def cycle_calculations():
    charge_points = 0
    max_voltage_discharge, min_voltage_discharge = 0, 10
    above_voltage, between_voltage = 0, 0
    charging_current, discharging_current = 0, 0

    if not prev_index:
        index = 1
    else:
        index += prev_index

    for row in db:
        if row[6]:
            charge_points += 1
            charging_current += row[2]
        else:
            max_voltage_discharge = max(max_voltage_discharge, row[1])
            min_voltage_discharge = min(min_voltage_discharge, row[1])

            if row[1] > 3.1:
                above_voltage += 1
            if 3.2 > row[1] > 3:
                between_voltage += 1

            discharging_current += row[2]

    discharge_points = len(db) - charge_points

    data["prev_index"] = index
    data["discharge_time"] = dx * discharge_points
    data["charging_time"] = dx * charge_points
    data["total_time"] = len(db) * dx
    data["max_voltage_discharge"] = max_voltage_discharge
    data["min_voltage_discharge"] = min_voltage_discharge
    data["time_above_3_1v"] = dx * above_voltage
    data["decrement_time_3_2v_3_1v"] = dx * between_voltage
    data["average_discharge_current"] = discharging_current / len(db)

```

```

data["average_charging_current"] = charging_current / len(db)

return data

```

server.py

```

from flask import Flask, request, jsonify, render_template, g
from database import get_cycle_db, get_instant_db
from flask_cors import CORS
import sqlite3

```

```

app = Flask(__name__)
CORS(app)

```

```

DATABASE = 'lion_dataset.db'

```

```

def get_db():
    if 'db' not in g:
        g.db = sqlite3.connect(DATABASE)
    return g.db

```

```

@app.teardown_appcontext

```

```

def close_db(error):
    if hasattr(g, 'db'):
        g.db.close()

```

```

@app.route('/', methods=['GET'])

```

```

def serve_index():
    return render_template('index.html')

```

```

@app.route('/api/instant_data', methods=['GET'])

```

```

def get_instant_data():
    db = get_db()
    cursor = db.cursor()
    cursor.execute("SELECT * FROM instant_data")
    data = cursor.fetchall()
    cursor.close()
    return jsonify(data)

```

```

@app.route('/api/cycle_data', methods=['GET'])

```

```

def get_cycle_data():

```



```

db = get_db()
cursor = db.cursor()
cursor.execute("SELECT * FROM cycle_data")
data = cursor.fetchall()
cursor.close()
return jsonify(data)

```

```
@app.route('/api/state', methods=['GET'])
```

```

def get_status():
    db = get_db()
    cursor = db.cursor()
    cursor.execute("SELECT * FROM status")
    data = cursor.fetchall()
    cursor.close()
    return jsonify(data)

```

```
@app.route('/api/status', methods=['POST'])
```

```

def status_control():
    try:
        res = request.json
        status = "Charging" if res['status'] else "Discharging"
        msg = "1" if res['status'] else "0"

        print("status change request received")

```

```
print("status changed to ", status )
```

```

        return jsonify({"message": "Data added successfully"}), 201
    except Exception as e:
        return jsonify({"error": str(e)}), 400
    finally:
        db = get_db()
        cursor = db.cursor()
        # cursor.execute("SELECT * FROM status")
        cursor.execute("""
            UPDATE status
            SET status = ?
            WHERE id = 1
            """, (msg))

        db.commit()

```

```
cursor.close()
```

```
if __name__ == '__main__':  
    app.run(debug=True,host="0.0.0.0")
```

REFERENCES

- [1] T. Reddy-Linden's, "Handbook of Batteries" in , McGraw-Hill Professional, 2010.
- [2] M. S. H. Lipu, M. A. Hannan, A. Ayob, M. H. M. Saad and A. Hussain, "Review of lithium-ion battery state of charge estimation methodologies for electric vehicle application", Int. J. Eng. Technol., vol. 7, no. 3.17, pp. 219-224, Aug. 2018
- [3] R. Zhang, B. Xia, B. Li, L. Cao, Y. Lai, W. Zheng, et al., "State of the art of lithium-ion battery SoC estimation for electrical vehicles", Energies, vol. 11, no. 7, pp. 1820, Jul. 2018.
- [4] P. Sabine et al. Methods for state-of-charge determination and their applications J Power Sources (2001)
- [5] J. Garche et al. The influence of different operating conditions, especially over-discharge, on the lifetime and performance of lead/acid batteries for photovoltaic systems J Power Sources (1997)
- [6] X.Y. Li Remaining useful life prediction for lithium-ion batteries based on a hybrid model combining the long short-term memory and Elman neural networks J. Energy Storage (2019)
- [7] Y. Jiang State of health estimation of second-life LiFePO₄ batteries for energy storage applications J. Clean. Prod. (2018)
- [8] J. M. Tarascon, "Key challenges in future Li-battery research", Phil. Trans. Roy. Soc. A Math. Phys. Eng. Sci., vol. 368, no. 1923, pp. 3227-3241, Jul. 2010.
- [9] A. Perner et al.8 - Lithium-ion batteries for hybrid electric vehicles and battery electric vehicles
- [10] Ch. Ehret, S. Piller, W. Schroer, A. Jossen, State-of-charge determination for lead-acid batteries in PV-applications
- [11] Spotnitz R 2003 Simulation of capacity fade in lithium-ion batteries J. Power Sources 113 72-80
- [12] Eby R L 1978 Method and apparatus for determining the capacity of lead acid storage batteries US Patent 4,180,770 filed 1 March

- [13] Salkind A J, Fennie C, Singh P, Atwater T and Reisner D E 1999 Determination of state-of-charge and state-of-health of batteries by fuzzy logic methodology J. Power Sources 80 293-300
- [14] Texas Instruments 2002 High-performance battery monitor IC with Coulomb counter, voltage, and temperature measurements Doc. I.D. SLUS521A
- [15] S. Piller, M. Perrin and A. Jossen, "Methods for state-of-charge determination and their applications", Journal of Power Sources, pp. 113-120, 2001
- [16] Gregory L. Plett, "Extended Kalman filtering for battery management systems of LiPB-based HEV battery packs Part 1. Background", Journal of Power Sources, pp. 252-261, 2004
- [17] M. Dubarry Capacity loss in rechargeable lithium cells during cycle life testing: the importance of determining state-of-charge J Power Sources (2007)
- [18] M. Li, "Li-ion dynamics and state of charge estimation", Renewable Energy, vol. 100, pp. 44-52, 2017.
- [19] X. Wu, X. Li and J. Du, "State of charge estimation of lithium-ion batteries over wide temperature range using unscented Kalman filter", IEEE Access, vol. 6, pp. 41993-42003, 2018.
- [20] K.S. Ng, Y.F. Huang, C. Moo and Y. Hsieh, "An enhanced coulomb counting method for estimating state-of-charge and state-of-health of lead-acid batteries", 31st International Telecommunications Energy Conference (INTELEC 2009), 18-22 October 2009.
- [21] J. Yong-Min, C. Yong-Ki, A. Jung-Hoon and R. Seung-Hee, "Enhanced Coulomb counting method with adaptive SOC reset time for estimating OCV", IEEE Energy Conversion Congress and Exposition (ECCE), 2014.
- [22] O. Erdinc, B. Vural and M. Uzunoglu, "A dynamic lithium-ion battery model considering the effects of temperature and capacity fading", International Conference on Clean Electrical Power, 2009.
- [23] W Ahmad et al. A reliable technique for remaining useful life estimation of rolling element bearings using dynamic regression models Reliability Engineering & System Safety (2019)

- [24] X Si et al. Remaining useful life estimation - A review on the statistical data driven approaches European Journal of Operational Research (2011)
- [25] M Chang et al. Early stage data-based probabilistic wear life prediction and maintenance interval optimization of driving wheels Reliability Engineering & System Safety (2020)
- [26] C Hu et al. A new remaining useful life estimation method for equipment subjected to intervention of imperfect maintenance activities Chinese Journal of Aeronautics (2018)
- [27] P. C. Lopes Gerum, A. Altay and M. Baykal-Gürsoy, "Data-driven predictive maintenance scheduling policies for railways", Transp. Res. Pt. C-Emerg. Technol., vol. 107, pp. 137-154, Oct. 2019.
- [28] Y. Lei, N. Li, L. Guo, N. Li, T. Yan and J. Lin, "Machinery health prognostics: A systematic review from data acquisition to RUL prediction", Mech. Syst. Signal Proc., vol. 104, pp. 799-834, May 2018.
- [29] S. Lee et al. "State-of-charge and capacity estimation of lithium-ion battery using a new open-circuit voltage versus state-of-charge," J. Power Sources (2008)
- [30] K.S. Ng et al. Enhanced coulomb counting method for estimating state-of-charge and state-of-health of lithium-ion batteries Appl. Energy (2009)
- [31] Augmented model-based framework for battery remaining useful life prediction 2022, Applied Energy
- [32] Asselman,A.,Khaldi,M.,andAammou,S.(2021).Enhancing The Prediction of student performance based on machine learning xg boost algorithm.
- [33] Bian, S. (2006). DataMiningEnsembleHierarchy,DiversityandAccuracy. PhDthesis,UniversityofEastAnglia.
- [34] Bian, S. andWang,W. (2007). On Diversity And Accuracy Of Homogeneous and heterogeneous ensembles. International Journal ofHybrid Intelligent Systems,4(2):103–128.
- [35] Breiman,L. (2001).Randomforests.MachineLearning,45(1):5–32.
- [36] Buitinck, L., Louppe,G., Blondel,M., Pedregosa, F.,Mueller, A.,Grisel, O.,Niculae,V., Prettenhofer, P.,Gramfort,A.,Grobler, J., Layton,R., VanderPlas, J., Joly,A.,Holt,B., andVaroquaux,G. (2013)

- [37] Chen, T. and Guestrin, C. (2016). Xgboost: A Scalable Tree Boosting System. In Proceedings Of The 22nd international conference on knowledge discovery data mining
- [38] Demšar, J. (2006). Statistical comparison of classifiers over multiple datasets. Journal of Machine learning research, 7(Jan)
- [39] Freund, Y., Schapire, R., and Abe, N. (1999). A Short Introduction To Boosting. Journal-Japanese Society For Artificial Intelligence, 14(771-780)
- [40] Hoerl, A. E. and Kennard, R. W. (1970). Ridge Regression: Biased Estimation for nonorthogonal problems. Technometrics

Battery Technology

ORIGINALITY REPORT

15%

SIMILARITY INDEX

12%

INTERNET SOURCES

9%

PUBLICATIONS

12%

STUDENT PAPERS

PRIMARY SOURCES

1	st.robu.in Internet Source	1%
2	onlinelibrary.wiley.com Internet Source	1%
3	www.mdpi.com Internet Source	1%
4	Matúš Danko, Juraj Adamec, Michal Taraba, Peter Drgona. "Overview of batteries State of Charge estimation methods", Transportation Research Procedia, 2019 Publication	1%
5	www.hindawi.com Internet Source	1%
6	Submitted to University of Wollongong Student Paper	<1%
7	Submitted to Coventry University Student Paper	<1%
8	www.coursehero.com Internet Source	<1%

9	Submitted to University of Warwick Student Paper	<1 %
10	Submitted to Swinburne University of Technology Student Paper	<1 %
11	Submitted to National Institute of Technology Karnataka Surathkal Student Paper	<1 %
12	www.erckerala.org Internet Source	<1 %
13	vbn.aau.dk Internet Source	<1 %
14	"Sustainable Energy Storage in the Scope of Circular Economy", Wiley, 2023 Publication	<1 %
15	Submitted to RMIT University Student Paper	<1 %
16	Submitted to National Institute of Technology, Patna Student Paper	<1 %
17	stackoverflow.com Internet Source	<1 %
18	Submitted to University of Edinburgh Student Paper	<1 %
19	Submitted to Banaras Hindu University	

<1 %

20

www.mindg.cn

Internet Source

<1 %

21

Submitted to King's College

Student Paper

<1 %

22

Submitted to American University of Central Asia

Student Paper

<1 %

23

Goteti Sai Navyasri, Bandaru Nanda Krishna, Pamidi Sai Manideep, V.S.Kirthika Devi. "Smart Controller IoT based for Electric Vehicles", 2023 Third International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), 2023

Publication

<1 %

24

Submitted to University of Newcastle

Student Paper

<1 %

25

www.litime.com

Internet Source

<1 %

26

Submitted to Özyegin Üniversitesi

Student Paper

<1 %

27

"Proceedings of the FISITA 2012 World Automotive Congress", Springer Nature, 2013

Publication

<1 %

28	Submitted to Westminster School Student Paper	<1 %
29	mdpi-res.com Internet Source	<1 %
30	Submitted to American Public University System Student Paper	<1 %
31	Submitted to Nanyang Technological University Student Paper	<1 %
32	Xing, Yinjiao, Eden W. M. Ma, Kwok L. Tsui, and Michael Pecht. "Battery Management Systems in Electric and Hybrid Vehicles", Energies, 2011. Publication	<1 %
33	"Power Systems Research and Operation", Springer Science and Business Media LLC, 2024 Publication	<1 %
34	Wang, J.. "A fuel cell city bus with three drivetrain configurations", Journal of Power Sources, 20060922 Publication	<1 %
35	blogs.cuit.columbia.edu Internet Source	<1 %
36	vdocuments.site Internet Source	

<1 %

37

Submitted to Pacific Lutheran College

Student Paper

<1 %

38

Submitted to The University of the West of Scotland

Student Paper

<1 %

39

Submitted to UTC Heathrow

Student Paper

<1 %

40

Submitted to University of Surrey

Student Paper

<1 %

41

Submitted to Wayne State College

Student Paper

<1 %

42

b4ltazar.wordpress.com

Internet Source

<1 %

43

Submitted to Beykent Universitesi

Student Paper

<1 %

44

Submitted to Eastern University

Student Paper

<1 %

45

Submitted to University of Wales, Bangor

Student Paper

<1 %

46

hl-128-171-57-22.library.manoa.hawaii.edu

Internet Source

<1 %

47

worldwidescience.org

Internet Source

<1 %

48

Submitted to Tilburg University

Student Paper

<1 %

49

Submitted to University of Adelaide

Student Paper

<1 %

50

e-carsworld.blogspot.com

Internet Source

<1 %

51

ebin.pub

Internet Source

<1 %

52

rincon-mora.gatech.edu

Internet Source

<1 %

53

uu.diva-portal.org

Internet Source

<1 %

54

Jie Lu, Zhenlin Liu, Wangjie Zhang, Jialu Zheng, Chenhui Han. "Pressure Prediction Study of Coal Mining Working Face based on Nadam-LSTM", IEEE Access, 2023

Publication

<1 %

55

Submitted to Sabine Pass High School

Student Paper

<1 %

56

Submitted to University of Colorado, Denver

Student Paper

<1 %

57

Submitted to University of Dammam

Student Paper

<1 %

58	Submitted to University of Teesside Student Paper	<1 %
59	www.iberdrola.com Internet Source	<1 %
60	Submitted to Chester College of Higher Education Student Paper	<1 %
61	Submitted to IPS International Ltd Student Paper	<1 %
62	SpringerBriefs in Electrical and Computer Engineering, 2016. Publication	<1 %
63	Submitted to Technological University Dublin Student Paper	<1 %
64	Vikas Panit, Rajeev Kumar Chauhan. "Evaluation of Different State of Charge Estimation Techniques for Electric Vehicles", 2023 XIX International Scientific Technical Conference Alternating Current Electric Drives (ACED), 2023 Publication	<1 %
65	forum.arduino.cc Internet Source	<1 %
66	nanografi.com Internet Source	<1 %

67	Submitted to British University in Egypt Student Paper	<1 %
68	Submitted to Glyndwr University Student Paper	<1 %
69	skill-lync.com Internet Source	<1 %
70	Submitted to Wilmington University Student Paper	<1 %
71	keep.lib.asu.edu Internet Source	<1 %
72	Submitted to Cranfield University Student Paper	<1 %
73	Submitted to National University of Ireland, Maynooth Student Paper	<1 %
74	Submitted to SUNY, Binghamton Student Paper	<1 %
75	Submitted to University Tun Hussein Onn Malaysia Student Paper	<1 %
76	Submitted to University of Cape Town Student Paper	<1 %
77	git.cs.tu-dortmund.de Internet Source	<1 %

Exclude quotes On

Exclude bibliography On

Exclude matches < 12 words